



TOC

Logi Symphony 24	50
Manipulate Data in Symphony 24 Managed Dashboards and Reports Data Stores	50
Quick Start Reference - Managed Dashboards and Reports Data Stores	51
Data Cubes	51
Working with Data	52
Transforms	53
Select Transforms	54
Common Transforms	54
Other	55
Output	56
Data Cubes	57
Design a Data Cube	59
The Data Cube Canvas	59
Data Cube Canvas - Toolbar	61
Create a New Data Cube	63
Data Cube Options	65



Edit a Data Cube	67
Data Cubes - Blank	69
Data Cubes - Manual Select	71
Data Cube - Data Input	74
Data Cube - Python Data Generator	77
Data Cube - R Data Generator	79
Data Cube Data Storage Options	81
Data Cube Storage Preparation and Use	82
Data Cube Preparation	82
Using Data Cube Storage	85
Check in Your Data Cube	86
Build the Data Cube	86
Schedule Data Cube Storage	87
Job Failure	89
Disable Memory Management	89
Administrator Options	90
Limitations	90
Incremental Data Update	90



Configuration Notes	93
Check the Storage Type and Status	93
In-Memory Build Performance	94
Raw Data	95
Optimize Storage Jobs	95
Create a Cube Perspective	97
Creating a Cube Perspective	97
Connect to an OLAP Data Source	97
Create a New Perspective	97
Drag and Drop a Native OLAP Cube	98
Select Measures and Dimensions	99
Check in Your Changes	100
Using a Cube Perspective	101
Using a Bridge Parameter	102
Examples	104
Multi-Level Hierarchy With Single Selection	104
Multi-Level Hierarchy With Single Selection On Multiple Levels	109
Single-Level Hierarchy With Multiple Selection	113



Single-Level Hierarchy With Inverted Multiple Selection	117
Single-Level Hierarchy With Single Selection Using MDX	119
Date Range Selection	120
Number Range Selection	123
	125
Handling All and Other Basic Tokens	125
Types	126
Using Manual Select Placeholders	129
Using a Placeholder With SQL	129
Defining a Placeholder	129
Using the Placeholder on a Dashboard	133
Using a Stored Procedure With a Placeholder	135
Using MDX With a Placeholder	135
Using TRACE Statements	137
TRACE Statements	137
Using #TRACE With a Calculated Element Transform	137
Viewing the Output	140
Writing Data Scripts with DundasScript	142



The DundasScript Engine	142
Syntax	142
Numeric Values	144
Features	145
Limitations	145
Standard .NET Classes	146
Dundas Classes	151
Debugging	151
Using Transforms	152
Input Transforms	152
Common Transforms	153
Other Transforms	154
Output Transforms	156
Aggregate	157
Input	157
Configure	157
Output	159
Calculated Element	162



Input	162
Add the Transform	163
Configure the Transform	165
Output	168
Working with Placeholders	169
Expression Samples	170
Limitations	172
Clustering predictions using R language analysis	173
Prepare the Data	173
Create the Data Cube Using Rserve	173
Using SQL Server R Services	177
Use the Results	179
R Resources:	180
Copy Element	181
Input	181
Configure	181
Output	182
Data Conversion	184



Input	184
Configure	185
Output	187
Data Cube Select	190
Add the Transform	190
Configure	190
Output	192
Data Input	193
Input	193
Add the Transform	193
Configure the Transform	196
Output	198
Display 'top N + other' Measures Using Data Cube Transforms	200
Set Up the Data Cube	200
Top/Bottom Transform	200
Union Transform	202
Sort Transform	202
Rank Transform	204



Filter Transform	206
Aggregate Transform	207
Preview the Data	209
Replace the Empty String	209
Calculated Element Transform	210
Display 'top N + other' Measures Using Stored Procedure	214
Set Up the Data	214
Create Stored Procedure	215
Create a Data Cube	215
Using the Data Cube On the Dashboard	218
Filter Data Cube Using Complex Logical Expression	222
Create the Calculated Element transform	222
Create the Filter Transform	225
Filter	229
Input	230
Configure	231
Filter Settings	232
Parameters	234



Output	235
Flatten JSON	238
Input	238
Add the Transform	238
Configure	241
Editing Mappings	242
Output	244
Flatten XML	246
Input	246
Add the Transform	246
Configure	249
Editing Mappings	250
Output	252
Fusing	254
Input	254
Configure	255
Output	257
Fuzzy Grouping	259



Input	259
Add the Transform	260
Configure	261
Output	262
Fuzzy Lookup	265
Input	265
Configure	268
Output	270
Give a Friendly Name to a Transform	272
Friendly Name and Description	273
Join	276
Input	276
Configure	276
Output	278
Lookup	281
Input	281
Configure	283
Output	285



Manual Select	288
Input	288
Add the Transform	288
Configure	290
Edit the Query	292
Enable Subquery Optimization	292
Placeholders	293
Session Schema	293
Output	294
Math	296
Input	297
Add the Transform	297
Configure the Transform	298
Output	302
MDX Select	304
Input	304
Add the Transform	304
Configure	305



Output	307
Merge	309
Input	309
Add the Transform	309
Configure	310
Output	313
Null Replacement	314
Input	314
Configure	315
Output	315
Passing Parameter Value From Dashboard to Stored Procedure	317
Data	318
Set up Data Cube Parameters	318
Connect Dashboard Filters	320
Testing the Dashboard	325
Passing Placeholder Value From Dashboard to Manual MDX Select	329
Example Query	329
Create a Data Cube	329



Edit the Dashboard	334
Testing	336
Percentage Sampling	340
Input	340
Add the transform	340
Configure	341
Output	343
Pivot	344
Input	344
Configure	344
Output	346
Process Result	348
Data Cube Elements	348
Hide Elements	349
Edit Elements	350
Replace With a Hierarchy	351
All Member Caption	355
Attribute Hierarchies	357



Security Options	358
Supported Aggregators	359
Formatting Measures	359
	360
Configure	360
Output	362
Python Analysis	365
Setup	365
Input	365
Add the Transform	366
Configure the Transform	366
Output	368
Example Python Script	368
Setup	368
Define Placeholders	369
Create the Script	370
Adjust the Column Names and the Output	370
View the Result	372



Geocoding Addresses	373
Python Data Generator	378
Setup	378
Input	378
Add the Transform	378
Configure the Transform	380
Output	382
Generate Data From a JSON File	383
Setup	383
Create the Script	384
Generate Data From a REST Call	384
Create the Script	384
R Data Generator	387
Setup	387
Input	387
Add the Transform	387
Configure the Transform	389
Output	391



Example R scripts	392
Random Number Generation	392
Pre-Defined Datasets	392
R Language Analysis	395
Setup	395
Input	395
Add the Transform	396
Configure the Transform	398
Output	400
Example R Scripts	401
Concatenate Text Columns	401
Moving Average	402
Rank	407
Input	407
Add the Rank Transform	410
Configure	412
Output	415
Record Sampling	418



Input	418
Add the Transform	418
Configure	419
Output	421
Remove Duplicates	422
Input	422
Add the Transform	423
Configure	424
Output	426
Replace '(Empty String)' Text With Empty Value	428
Replacing '(Empty string)' Text	428
Re-Wire the Data Flow	431
Replace a Select Transform	431
Edit the Input Elements	435
Re-wiring a Manual Select's Data Connector	439
Re-Wire Metric Set Elements	439
Sort	444
Input	444



Configure	445
Output	447
SQL Select	451
Input	451
Add the Transform	451
	452
Configure	452
Transform Element	455
Add Expression	457
Define Parameters	457
Edit Input Elements	461
Edit Output Elements	461
Output	461
Stored Procedure Select	463
Input	463
Add the Transform	463
Configure the Transform	465
Output	469



String	472
Input	473
Configure	473
Output	477
Tabular Select	480
Input	480
	481
Add the Transform	481
Configure	482
Output	484
Top/Bottom	486
Input	486
Add the Transform	487
Configure	489
Output	491
Transpose	493
Input	493
Configure	493



Output	496
Union	499
Input	499
Configure	500
Output	502
Unpivot	505
Input	505
Configure	505
Output	507
Update a Select Transform When The Data Source Changes	509
Refresh the Select Transform	509
Using the Filter Transform With Tokens	512
Current Display Name token	512
Connect to Data	512
Set Up User Accounts	512
Create a Data Cube	513
Create a Metric Set	516
Create a Dashboard	518



View the Dashboard	518
Filtering Out Null Values	518
Add a Relationship Between Tables	522
Review Existing Relationships	522
Create a Relationship	526
Using Relationships	529
Add a Slicer Comparison	534
Walkthrough	534
Create a Metric Set	534
Add Slicers	535
Add a Slicer Comparison	536
Add a Filter to Change While Viewing	540
Adding Contextual Measures	543
Adding a Contextual Measure	543
Editing the Measure Values	545
Different Contexts	547
Automatic Joins and Hierarchies	548
Automatic Joins	548



Display ProductID on Columns	549
Replace ProductID with Name	550
Edit Automatic Joins	552
Hierarchies	558
Create a Product Hierarchy	558
Replace ProductID with Product Hierarchy	561
Attributes	563
Add Attributes	563
Use Attributes	565
Create a Metric Set and Add a Filter	569
Create a New Metric Set	569
Set Up the Metric Set Based on a Data Cube	569
Complete the Metric Set	572
Add a Second Metric Set	575
Display the Metric Sets on a Dashboard	579
Using Filters on a Dashboard	581
Create a Ragged Hierarchy	587
Data	587



Steps	588
Using the Ragged Hierarchy	592
Visualizing Levels	592
Tables	592
Charts	593
Filters	594
Create a Time Dimension	596
Create a New Time Dimension	596
Gregorian	597
Fiscal	600
ISO 8601	602
Reporting (Marketing)	602
13 Period	604
Format Time Dimension Levels	606
Fiscal Month Formatting	609
Fiscal Year Formatting	612
Week Start/End Formatting	614
Quarter Start/End Formatting	614



Reporting and 13 Period Weeks	615
Sorting Time Dimension Levels	615
Preview Your Time Dimension	616
Time Hierarchies and Attributes	617
Define Custom Hierarchy Sorting	624
Defining Custom Hierarchy Sorting	624
Specify Members To Be on Top or Bottom in Sort Order	630
Define Custom Measure Sorting	635
Defining Custom Measure Sorting	635
Opposite Axis Hierarchies	639
Displaying Top/Bottom Records	643
Set the Top / Bottom Measure	643
Set the Other Group Measure	645
Exclude Data Points from the Visualization	649
Example preparation	649
Exclude Selected Data Points	650
Keep Only Selected Data Points	654
Clear All Filters	657



Expand and Collapse Hierarchy Members	660
Expand and Collapse	660
Charts	660
Treemaps	663
Tables	665
Expand/Collapse On Click	668
Collapse All	671
	674
Working with Hierarchies	674
Formatting Text	675
Metric Set Element Format	675
Formatting Currency	679
Formatting Large Values	682
Formatting Ranges	684
Time Dimension Formatting	686
Placeholder Keywords	687
Formats	689
Filter and Level Keywords	691



Visualization Keywords	693
Size and Color Rule Keywords	694
Format Properties	694
Group Members During Analysis	697
Ad Hoc Member Grouping	697
Working With Multiple Hierarchies	701
Using Custom Member Grouping	705
Handling Missing Data	707
Example Data	707
Missing Measure Values	707
Missing Dates or Hierarchy Values	709
Using a Missing Data Rule	711
Styling Missing Data	716
Table	716
Chart	718
How To Create a Custom User Hierarchy	722
Example Data	722
Steps	726



Set Up the First Level	726
Add the Second Level	730
Add the Third Level	732
Preview	735
	736
Using the Hierarchy	737
Promoting an Auto-Generated Metric Set	738
Creating a Metric Set in a Dashboard	738
Promote the Metric Set	740
Auto-Generated Data Cubes	742
Set or Reset the Default Time Dimension	746
View the Default Time Dimension	746
Set the Default Time Dimension	746
Reset the Default Time Dimension	748
Using the Default Time Dimension	749
Setting Up Data Input	751
Data in a Data Cube	751
Entering Data Into a New Metric Set	751



Entering Data When Editing a Dashboard	756
Modify Input Data	761
Pasting Data	761
Deleting Rows	761
Working With Data Input	762
Set Up States on a Metric Set	763
Setting Up States	763
Create a Metric Set	763
Add a State	765
Add Another State Group	770
Visualizing States	770
Examples	777
Comparing Against a Constant	777
Comparing Against a Hierarchy	780
Slicers Versus Columns and Rows	787
Data Analysis Panel	787
Measures	789
Rows	791



Slicers	793
Columns	795
Use Metric Set Query Hints	800
Using Query Hints	800
Prefer NON-EMPTY	802
Read Uncommitted	802
Adding Formulas	804
Accessing the formula bar	804
Simple Formulas	807
Constant Value	807
Placeholders	810
Placeholder Settings	813
Math	814
Functions	815
Advanced Mode	818
Formula Visualizations	820
Editing a Formula	820
Parameter Placeholders	822



Filtering	825
Accumulation Distribution	826
Syntax	826
Input	826
Parameters	826
Output	826
Aroon Oscillator	828
Syntax	828
Input	828
Parameters	828
Output	829
Average Directional Index	830
Syntax	830
Input	830
Parameters	831
Output	831
Average True Range	832
Syntax	832



Input	832
Parameters	832
Output	832
Bollinger Bands	833
Syntax	833
Input	834
Parameters	834
Output	834
Chaikin Money Flow	836
Syntax	836
Input	836
Parameters	836
Output	836
Chaikin Oscillator	838
Syntax	838
Input	838
Parameters	838
Output	839



Commodity Channel Index	840
Syntax	840
Input	840
Parameters	840
Output	840
Correlation	841
Syntax	843
Input	843
Parameters	843
Output	843
Correlation Matrix	845
Syntax	846
Input	846
Parameters	846
Output	846
Cumulative Total	848
Syntax	848
Input	849



Parameters	849
Output	849
Data Outliers	850
Syntax	850
Input	851
OUTIQ parameters	851
Output parameters	852
Output	852
Distance	853
Syntax	853
Input	853
Parameters	854
Output	854
Using a Custom Total Calculation	855
Creating a Custom Total Formula	855
Other Options	859
Exponential Moving Average	862
Syntax	864



Input	864
Parameters	864
Output	864
Exponential Smoothing	865
Syntax	866
Forecasting	866
Expsmooth	866
Input	867
Parameters	867
Output	868
Automatic Forecast Analysis	869
Examples	872
Single Exponential Smoothing	873
Double Exponential Smoothing	874
Triple Exponential Smoothing	875
Fast Stochastic	878
Syntax	878
Input	878



Parameters	878
Output	879
First Subset	880
Syntax	880
Input	880
Parameters	880
Output	880
Histogram	882
Syntax	884
Input	884
Parameters	884
Output	885
Historical Volatility	886
Syntax	887
Input	887
Parameters	887
Output	887
K-Means Clustering	888



Syntax	888
Input	888
Parameters	889
Output	889
Examples	889
Last Subset	895
Syntax	895
Input	895
Parameters	895
Output	895
List of Formulas	897
Standard Functions	897
Statistical Functions	899
Subset Functions	900
Financial Functions	901
Math Functions	905
DundasScript	906
Maximum Subset	908



Syntax	908
Input	908
Parameters	908
Output	908
Mean (Simple Average)	910
Syntax	910
Input	910
Parameters	911
Output	911
Median	912
	912
Syntax	912
Input	912
Parameters	913
Output	913
Middle Subset	914
Syntax	914
Input	914



Parameters	914
Output	914
Minimum Subset	916
Syntax	916
Input	916
Parameters	916
Output	916
Mode	918
Syntax	920
Input	920
Parameters	920
Output	920
Momentum	921
Syntax	921
Input	921
Parameters	921
Output	921
Money Flow Index	923



Syntax	923
Input	923
Parameters	923
Output	923
Moving Average	925
Syntax	927
Input	927
Parameters	927
Output	927
Moving Average Convergence-Divergence	928
Syntax	928
Input	928
Parameters	928
Output	929
Moving Average Envelopes	930
Syntax	931
Input	931
Parameters	932



Output	932
Negative Volume Index	933
Syntax	933
Input	933
Parameters	933
Output	933
Normal Distribution	934
Syntax	936
Input	936
Parameters	936
Output	936
On Balance Volume	937
Syntax	937
Input	937
Parameters	937
Output	937
On Balance Weighted Volume	938
Syntax	938



Input	938
Parameters	938
Output	938
Parabolic SAR (stop-and-reverse)	939
Syntax	939
Input	939
Parameters	939
Output	939
Percentile	940
Syntax	942
Input	942
Parameters	942
Output	942
Percent of Total	943
Syntax	943
Input	943
Parameters	943
Output	944



Example	944
	945
Percent Rank	946
Syntax	948
Input	948
Parameters	948
Output	948
Population Standard Deviation	949
Syntax	951
Input	951
Parameters	951
Output	951
Positive Volume Index	952
Syntax	952
Input	952
Parameters	952
Output	952
Rank Function	953



Syntax	954
Input	954
Parameters	954
Output	954
Rank Value	956
Syntax	956
Input	956
Parameters	957
Output	957
Rate of Change	958
Syntax	958
Input	958
Parameters	958
Output	958
Relative Strength Indicator	959
Syntax	959
Input	959
Parameters	959



Output	959
Sample Standard Deviation	960
Syntax	960
Input	960
Parameters	961
Output	961
Slow Stochastic	962
Syntax	962
Input	962
Parameters	962
Output	963
Stochastic Oscillator	964
Syntax	964
Input	964
Parameters	964
Output	965
Totals (Sum)	966
Syntax	966



Input	966
Parameters	966
Output	967
Trend and Forecasting	968
Syntax	969
Input	969
Parameters	970
Output	970
Examples	971
Trend and Forecast on a Time Series	971
Linear Regression on Scatter Plot	974
Trend Confirmation Indicator	978
Syntax	978
Input	978
Parameters	978
Output	978
Ultimate Oscillator	979
Syntax	979



Input	979
Parameters	979
Output	980
Using a Formula Visualization	980
Creating a Formula Visualization	980
Combining Data From Multiple Visualizations	984
Grouping and Aggregating	987
Subset Functions	991
Filtering the Formula Metric Set	992
Changing the Formula Metric Set	995
Removing Elements	995
Adding Elements	997
Advanced Options	998
Limitations	999
Variance	1000
Syntax	1001
Input	1001
Parameters	1001



Output	1001
Volume Oscillator	1002
Syntax	1002
Input	1002
Parameters	1002
Output	1003
Weighted Moving Average	1004
Syntax	1006
Input	1006
Parameters	1006
Output	1006
Williams %R (or Percent Range)	1007
Syntax	1007
Input	1007
Parameters	1007
Output	1007
Refresh Data	1008
Refreshing a Data Cube	1008



Change Data Caching	1008
Re-Build Manually	1009
Schedule Warehouse and In-Memory Builds	1009
Rebuild Using the API	1009
Refreshing a Data Visualization	1010
Bypass Data Cache	1010
Auto Refresh Interval	1011
Manual Refresh	1012
Result Cache Configuration	1013
Zig Zag Subset	1015
Syntax	1015
Input	1015
Parameters	1015
Output	1015
Writing Scripts Using Browser Developer Tools	1016
Debug using Chrome Developer Tools	1016
Watermarking Sanctioned/Non-Sanctioned Data	1020
Prerequisite	1020



Organizing Sanctioned Data	1020
Dashboard Setup	1020
Watermarking all Dashboards in the Instance	1023
Test the Dashboard	1023



- Archive of documentation for Logi Symphony v24

Logi Symphony 24

Manipulate Data in Symphony 24 Managed Dashboards and Reports Data Stores



Quick Start Reference - Managed Dashboards and Reports Data Stores

This applies to: *Managed Dashboards, Managed Reports*

Data Cubes

- [Data Cubes](#)
- [Design A Data Cube](#)
- [Create A New Data Cube](#)
- [Data Cubes - Blank](#)
- [Data Cubes - Manual Select](#)
- [Data Cube - Data Input](#)
- [Data Cube - Python Data Generator](#)
- [Data Cube - R Data Generator](#)
- [Edit A Data Cube](#)
- [Data Cube Data Storage Options](#)
- [Data Cube Storage Preparation And Use](#)
- [Refresh Data](#)
- [Using a Bridge Parameter](#)
- [Using Manual Select Placeholders](#)



- [Using TRACE Statements](#)
- [Writing Data Scripts with DundasScript](#)

Working with Data

- [Add A Relationship Between Tables](#)
- [Add A Slicer Comparison](#)
- [Adding Contextual Measures](#)
- [Automatic Joins And Hierarchies](#)
- [Create A Metric Set And Add A Filter](#)
- [Create A Ragged Hierarchy](#)
- [Create A Time Dimension](#)
- [Define Custom Hierarchy Sorting](#)
- [Define Custom Measure Sorting](#)
- [Displaying Top/Bottom Records](#)
- [Exclude Data Points From The Visualization](#)
- [Expand And Collapse Hierarchy Members](#)
- [Formatting Text](#)
- [Group Members During Analysis](#)



- [Handling Missing Data](#)
- [Handling Null Value In A State Indicator Column](#)
- [How To Create A Custom User Hierarchy](#)
- [Promoting An Auto-Generated Metric Set](#)
- [Set Or Reset The Default Time Dimension](#)
- [Setting Up Data Input](#)
- [Set Up States on a Metric Set](#)
- [Slicers Versus Columns And Rows](#)
- [Use Metric Set Query Hints](#)
- [Using A Custom Total Calculation](#)

Transforms

- [Using Transforms](#)
- [Give A Friendly Name To A Transform](#)
- [Clustering Predictions Using R Language Analysis](#)
- [Display 'top N + Other' Measures Using Data Cube Transforms](#)
- [Display 'top N + Other' Measures Using Stored Procedure](#)
- [Filter Data Cube Using Complex Logical Expression](#)



- [Passing Parameter Value From Dashboard To Stored Procedure](#)
- [Passing Placeholder Value From Dashboard To Manual MDX Select](#)
- [Replace '\(Empty String\)' Text With Empty Value](#)
- [Re-Wire The Data Flow](#)

Select Transforms

- [Data Cube Select](#)
- [Data Input](#)
- [Manual Select](#)
- [MDX Select](#)
- [SQL Select](#)
- [Stored Procedure Select](#)
- [Tabular Select](#)

Common Transforms

- [Aggregate](#)
- [Calculated Element](#)
- [Data Conversion](#)
- [Filter](#)



- Archive of documentation for Logi Symphony v24

- [Join](#)
- [Lookup](#)
- [Math](#)
- [String](#)
- [Union](#)

Other

- [Copy Element](#)
- [Flatten JSON](#)
- [Flatten XML](#)
- [Fusing](#)
- [Fuzzy Grouping](#)
- [Fuzzy Lookup](#)
- [Merge](#)
- [Null Replacement](#)
- [Percentage Sampling](#)
- [Pivot](#)
- [Install Python](#)



- Archive of documentation for Logi Symphony v24

- [Python Analysis](#)
- [Python Data Generator](#)
- [Install And Configure R](#)
- [R Data Generator](#)
- [R Language Analysis](#)
- [Rank](#)
- [Record Sampling](#)
- [Remove Duplicates](#)
- [Sort](#)
- [Top/Bottom](#)
- [Transpose](#)
- [Unpivot](#)

Output

- [Process Result](#)

Data Cubes

This applies to: *Managed Dashboards, Managed Reports*

A data cube is a reusable multidimensional model of your data that consists of measures, hierarchies, and metadata such as formatting. Data cubes work primarily with non-OLAP sources such as relational databases, letting you view your data as a cube without having to use an OLAP system like analysis services. Use the information pulled together in these data cubes to build our your dashboards and reports.

- Use data cubes to integrate data from different sources, [transform the data](#), and produce a reusable data model with optional [data storage](#) capabilities.
- Set up an ETL process when editing a data cube by choosing from our wide range of [transforms](#) and connecting them together to perform tasks such as [data cleansing](#) and combining data.
- The [MDX Select](#) transform lets you use an OLAP data source as input for selecting a tabular set of data to join with other data.
- Performance can be improved by using the optional [storage types](#): warehouse to store the cube's output in Symphony, or in-memory for faster analytics.
- There is always a cube behind a visualization and its metric set, so a data cube may be auto-generated by Symphony if you drag data directly from a data source so that you can start to work with it immediately.
- Once you create a new data cube, you can set up an ETL process consisting of transforms, use transforms to join database tables, and link the output columns of a data cube to existing hierarchies.



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

Use the [data cube](#) work area to create your data cube structures visually. Add structures, create joins, and define your ETL process.

For more information about Data Cubes and data cube options, see:

- [Design A Data Cube](#)
- [Create A New Data Cube](#)
 - [Data Cubes - Blank](#)
 - [Data Cubes - Manual Select](#)
 - [Data Cube - Data Input](#)



- Archive of documentation for Logi Symphony v24

- [Data Cube - Python Data Generator](#)
- [Data Cube - R Data Generator](#)

- [Edit A Data Cube](#)
- [Using Transforms](#)
- [Data Cube Data Storage Options](#)

For more information, see:

- Video: [Introduction to Data Cubes](#)
- Video: [Data Cleansing with Data Cubes](#)
- [Data Cubes](#)
- [Data Cube Storage Preparation and Use](#)
- [The Managed Dashboards Data Model](#)



Design a Data Cube

This applies to: *Managed Dashboards, Managed Reports*

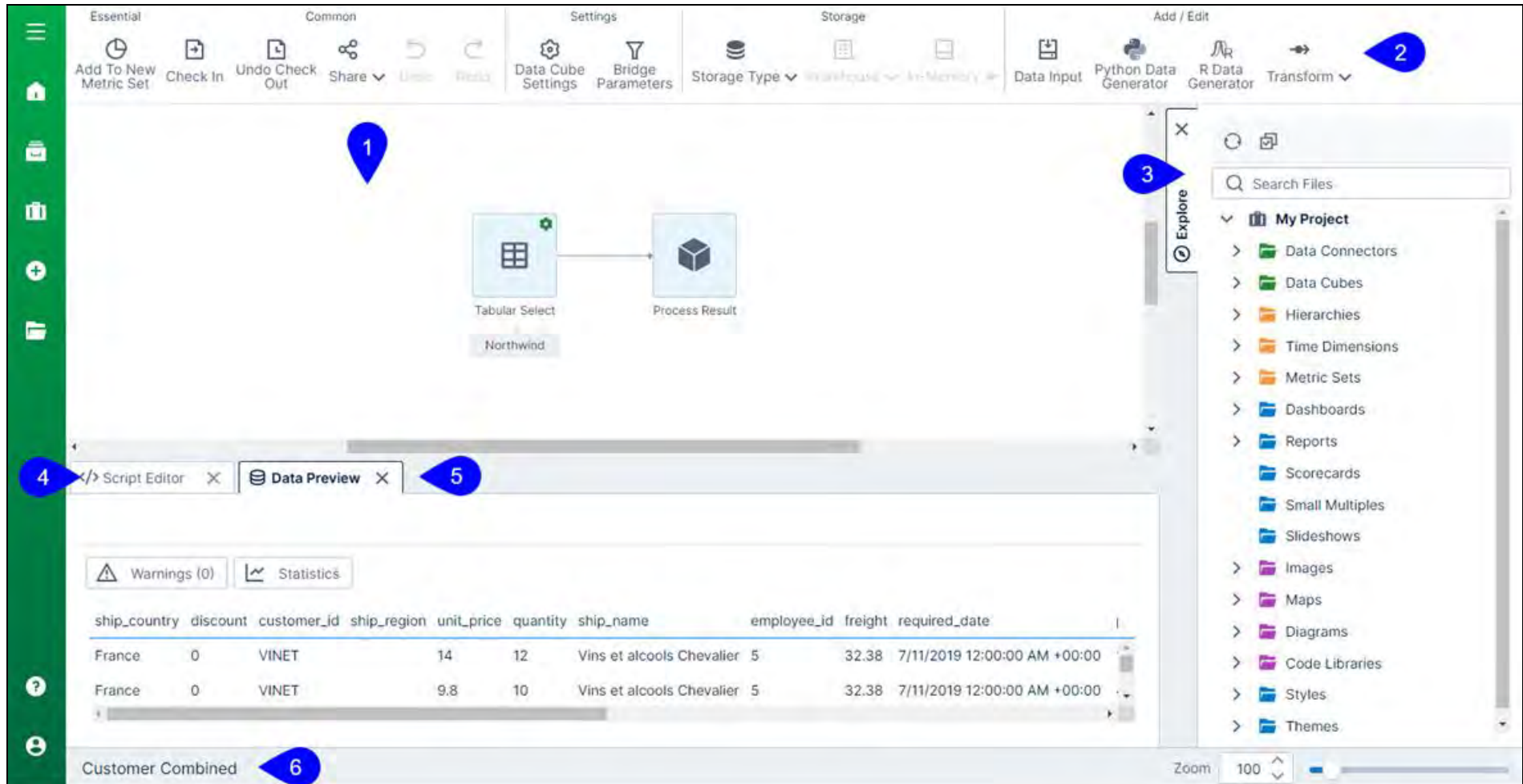


Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

Use the data cube canvas to create your data cube structures visually. Add structures, create joins, and define your ETL process.

The Data Cube Canvas

The data cube canvas is a large, scrollable work area you use to create your data cube.



1. The canvas. Drag data, structures, and transforms to create your data cube. Select elements and connect with joins to generate your process result. See [Design overview](#).
2. Toolbar. The options available in this toolbar change depending on the contents of the canvas and what elements you have selected.
3. Explore window. Explore the data you have available, and optionally select items to drop onto the data cube canvas.
4. Script Editor. Add custom queries to the data cube for [SQL or MDX](#), [Python or R](#), or [DundasScript](#).



5. Data Preview. Preview the data of a selected element in the canvas.
6. The status bar. Shows the name of the current data cube and other controls. See [Using the status bar](#).

Data Cube Canvas - Toolbar

The toolbar above the canvas changes depending on the nodes placed on your canvas, the items selected in the canvas, and if your data cube is checked in or checked out. Generally, most options are made unavailable when you view a checked in data cube.

Section	Menu Option	Select this option to...
Essential	Add to New Metric Set	Add this data cube to a new metric set.
Common	Check In	Check in a checked out data cube.
	Check Out	Check out a data cube for editing.
	Undo Check Out	Undo the check out of a data cube, returning it to a checked in state and removing all changes made since the last check in.
	Share	Open the Share work area and share this data cube as a link, adding users' access, or as an email.
	Undo	Undo your last action.
	Redo	Redo your last undone action.
Settings	Data Cube Settings	Configure settings for the overall data cube, unrelated to specific transforms. Options vary based on your data cube storage type .
	Bridge Parameters	Define a bridge parameter to promote a placeholder to a hierarchy.
Storage	Storage Type	Select a storage type for this data cube. Options include None, Warehouse, and In-Memory.
	Warehouse	Define settings for your Warehouse storage. Options include Build Warehouse, Schedule Warehouse, and Remove Schedule.
	In-Memory	Define settings for your In-Memory storage. Options include Build In-Memory, Schedule In-Memory, and Remove Schedule.
Add/Edit	Data Input	Add a Data Input transform to the data cube.
	Python Data Generator	Add a Python Data transform to the data cube.
	R Data Generator	Add an R Data Generator transform to the data cube.
	Transform	Add Common and Other transforms to the data cube. See Using Transforms .

For more information about Data Cubes and data cube options, see:



- Archive of documentation for Logi Symphony v24

- [Create A New Data Cube](#)
- [Edit A Data Cube](#)
- [Using Transforms](#)
- [Design a Data Cube](#)
- [The Managed Dashboards Data Model](#)

Create a New Data Cube

This applies to: *Managed Dashboards, Managed Reports*

Create a new data cube to organize your data from one source, or integrate data from multiple sources, starting from a blank cube or by using an available input transform to create your reusable data model. Once you've created and checked in a data cube, you and your users can create content in dashboards and reports that use this reusable [data model](#).

To create a cube, you'll connect content from your data sources as tables, SQL or MDX statements, transforms such as Data Input, Python Data Generator, and the R Data Generator. Use and configure joins, unions, lookups, or fusing to bring your data together, then review the process result.

Create a new data cube

1. Open the [data cube canvas](#). You can do this in one of several ways:
 - i. Select **Create Data Cubes** from the [home page](#)
 - ii. Select **Data** from the Managed Dashboards and Reports [main menu](#), then select **Create** and then select from **Blank**, **Data Input**, **Python Data Generator**, or **R Generator**
 - iii. Right-click **Data Cubes** in the Explore window, then select **New Data Cube** from the menu
2. Add a data structure to the data cube work area. Drag and drop a structure, such as a spreadsheet or table, from a data connector in the Explore window to the canvas, or select an option from the Add/Edit [toolbar](#).



Note: Data structures in your Symphony environment also include [data sources discovered](#) by Symphony in Managed Dashboard and Managed Reports data connectors, as well as shared from Data Discovery data connectors.
3. After you've added a first data structure, Symphony adds a connected Process Result to create a simple ETL process. The data cube is named based on the source of your first data structure: select the name in the data cube status bar to rename it.
4. Continue adding what you need: data structures and joins, scripts, and other transforms. Configure how they work together to complete your data cube structure. See [Adding, Connecting, And Configuring More Data Structures And Transforms](#).
5. Optionally, use the toolbar to perform tasks from the available menus (Essential, Common, Settings, [Storage](#), Add/Edit). The options available in the Contextual menu may vary based on what is selected on the canvas.
6. Configure your Process Result. Configure the output of the data users with [read access](#) see when using it as a data source in [metric sets](#).



7. When you've added, connected, and configured all elements, your options include:

- i. Select **Check In** from the toolbar to make your data cube available to for use in your project and sharing with other users.
- ii. Use the data cube in a metric set or dashboard before you check it in and make it available to others.

Adding, connecting, and configuring more data structures and transforms

1. When you add a new data structure, transform, or join, it may display on the canvas as a red icon until:

- i. You connect it to your ETL process.
- ii. You configure the connected transform.
- iii. Configure a join type, set up the unique key binding, or select and deselect columns for inclusion in the output.



Note: You can select a specific color for a structure on the canvas by changing it in the Configure dialog.

2. Insert or connect a new transform by one of several methods:

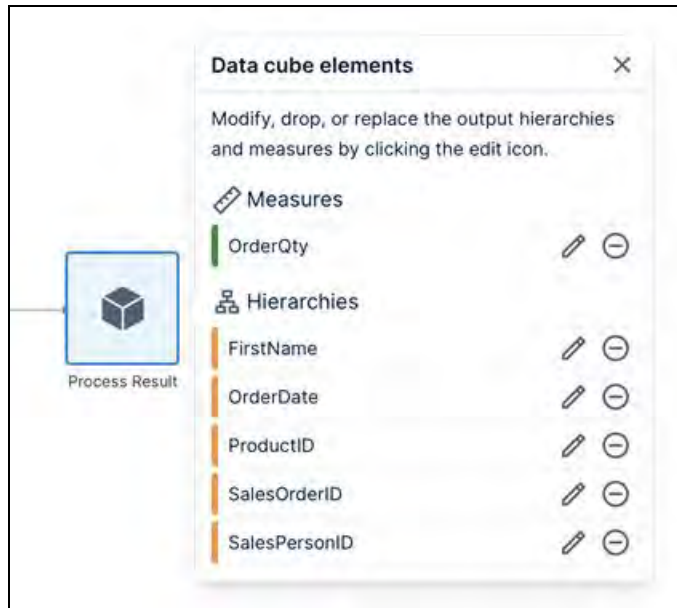
- i. Select an existing connection between two existing transforms to insert the new one.
- ii. Select an existing transform and drag it to a transform to connect the two.

3. Configure your newly added transforms. Select the transform, then **Configure** in the toolbar, double-click the transform, or select Configure from the menu when you right-click on the transform.

4. When you've configured all elements, right-click the Process Result element to configure the final output of the data users with [read access](#). See [Configure A Process Result](#).

Configure a Process Result

1. When you've configured all elements, right-click the Process Result element to configure the final output of the data users with [read access](#). The Data cube elements work area opens.



2. Customize each element's name, description, predefined formatting, and more. Any data not used as a measure becomes a hierarchy when output from the cube whether or not it was based on a hierarchy you defined ahead of time to contain multiple levels or other customizations. If not replaced with a predefined hierarchy, a column of data becomes an 'implicit' hierarchy.
3. To replace an implicit hierarchy with your own that's based on matching key data, select the option **Select a hierarchy or level to use as a replacement**. In the Open dialog that appears, select the hierarchy, or expand it and select a particular level that matches the column's values.
4. Select the **Open** button at the bottom to proceed and replace the implicit hierarchy, then **Save** your changes. See [Process Result](#) for more information.

Data Cube Options

When you create a data cube, Symphony provides several options to get you started adding your first input transform. If you don't have the appropriate [application privileges](#), some options may be hidden from you.

- **Blank.** Build a data cube on a blank canvas in the data cube work area.
- **Manual Select.** Enter a SQL or MDX statement to make a selection from a data connector instead of dragging structures onto the canvas.
- **Data Input.** Reference a warehouse data storage are containing user input directly, or through a [data input interaction](#).



- Archive of documentation for Logi Symphony v24

- [Python Data Generator](#). Generate data by writing scripts using the Python programming language.
- [R Data Generator](#). Generate output by running R scripts against the R server.

Each of these options available to you can also be added as additional input transforms to existing data cubes. See [Edit A Data Cube](#).



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

For more information, see:

- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)

Edit a Data Cube

This applies to: *Managed Dashboards, Managed Reports*

Edit an existing data cube quickly and easily to change the structure of your data cube, edit the properties, and make other changes as needed.

Edit a data cube

1. Find and open a data cube. You can do this in one of several ways:
 - i. Open Managed Dashboards or Managed Reports, then **Open > Data Cube** to browse or search the data cube content to which you have access. Navigate within your current open project, or use the Projects option in the main menu to open and search other shared projects.
 - ii. Select **Data Cubes** in the Explore window to browse or search the data cube content to which you have access. Navigate within your current open project, or use the Projects option in the main menu to open and search other shared projects.
 2. Right-click the data cube you want to edit. An options menu opens. Select an edit option to edit the data cube, or another option to perform other actions.
-  **Note:** If the data cube is currently checked in, you'll need to check it out to make changes.
3. When you've added, connected, and configured all elements, select **Check In** from the toolbar to make your data cube available to for use in your project and sharing with other users.

The menu options available from the right click menu vary based on the status of the data cube, and may include:

Menu Option	Select this option to...
Edit	Edit the selected data cube in this browser window.
Edit (Pop-up)	Edit the selected data cube in a new browser window.
Check In	Check in a checked out data cube. See Check In, Check Out and Auto Saving .
Check Out	Check out a data cube for editing. See Check In, Check Out and Auto Saving .
Undo Check Out	Undo the check out of a data cube, returning it to a checked in state and removing all changes made since the last check in. See Check In, Check Out and Auto Saving .
Publish	Publish the data cube to another project. Select a destination and options for publishing this cube. See Working With Projects .
Cut	Cut the selected data cube from its current location in the Explorer window. Paste the cube to a different location in the Explorer window.

Menu Option	Select this option to...
Copy	Copy the selected data cube in the Explorer window. Paste a copy of the cube to a different location in the Explorer window.
Paste	Paste a cut or copied data cube into a new location in the Explorer window.
Refresh	Refreshes the data cube's data, including name, description, database tables, and more, to ensure you have the most up-to-date information if it has changed since you last accessed it.
Rename	Rename the data cube.
Delete	Delete the data cube.
Properties	View and edit the properties for this data cube. Edit the associated Name, Description, and Tags. View the Revision history, References, as well as review and revise Security options for the cube. See File and Folder Properties .



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

For more information, see:

- [Data Cubes](#)
- [Edit a Data Cube](#)
- [The Managed Dashboards Data Model](#)



Data Cubes - Blank

This applies to: *Managed Dashboards, Managed Reports*

When you create a data cube, you can optionally start with a blank canvas to build the structures and ETL process your organization needs.

Create a new, blank data cube

1. Open the [data cube](#) work area. You can do this in one of several ways:
 - i. Select **Create Data Cubes** from the [home page](#).
 - ii. Select **Data** from the Managed Dashboards and Reports [main menu](#), then select **Create** and then select **Blank**.
 - iii. Right-click **Data Cubes** in the Explore window, then select **New Data Cube** from the menu .
2. Select and drop a data structure into the data cube work area. You can do this in one of several ways:
 - i. Locate an existing [data connector](#) in the Explore window and expand it to show data structures such as tables. Drag and drop a structure to the canvas.

 **Note:** Data structures in your Symphony environment also include [data sources discovered](#) by Symphony in Managed Dashboard and Managed Reports data connectors, as well as shared from Data Discovery data connectors.
 - ii. Select an option from the Add/Edit [toolbar](#).
3. After you've added a first data structure (node), Symphony adds a second node: a connected Process Result .
4. Add more data structures, joins, transforms, scripts, and other inputs as needed to the single output, the Process Result.
5. Optionally, use the toolbar to perform tasks from the available menus (Essential, Common, Settings, [Storage](#), Add/Edit). The options available in the Contextual menu may vary based on what is selected on the canvas.
6. Configure your Process Result. Configure the output of the data users with [read access](#) see when using it as a data source in [metric sets](#).
7. When you've added, connected, and configured all elements, select **Check In** from the toolbar to make your data cube available to for use in your project and sharing with other users.



- Archive of documentation for Logi Symphony v24



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

For more information, see:

- [Data Cubes](#)
- [Data Cubes - Blank](#)
- [The Managed Dashboards Data Model](#)

Data Cubes - Manual Select

This applies to: *Managed Dashboards, Managed Reports*

When you create a data cube, you can enter an SQL or MDX statement to make a selection from a data connector instead of dragging native structures onto the canvas.

Create a new data cube with manual select



Important: If you don't have the appropriate [application privileges](#), some options may be hidden from you.

1. Open the [data cube](#) work area. You can do this in one of several ways:
 - i. Select **Create Data Cubes** from the [home page](#), then **Manual Select** from the Add/Edit menu
 - ii. Select **Data** from the Managed Dashboards and Reports [main menu](#), then select **Create** and **Manual Select**
 - iii. Right-click **Data Cubes** in the Explore window, then select **New Data Cube** from the menu, then **Manual Select** from the Add/Edit menu

The Open dialog opens, allowing you to navigate to an appropriate data source.

2. Select a data connector. (For OLAP databases, select the cube under the data connector you want to query.)



Note: Data structures in your Symphony environment also include [data sources discovered](#) by Symphony in Managed Dashboard and Managed Reports data connectors, as well as shared from Data Discovery data connectors.

3. Enter your query in the script editor at the bottom of the data cube work area. Enter it manually, or use shortcuts to build it:



Note: Inserted comments in your query may cause an issue when [subquery optimization](#) is enabled for some data providers.

- i. The keyboard shortcut `CTRL + SPACE` opens a list of available SQL keywords and table names you can add to your query.
- ii. Drag a structure from the Explore window to the script editor (cube, column, measure, hierarchy level) to add it to your by name.
- iii. Drag a [stored procedure](#) from the **Explore** window to add it to the query.



4. Optionally, enter a placeholder in your parameterized SQL statement, with each parameter enclosed in dollar sign characters (\$). See [Using manual select placeholders](#).

5. **Save** and configure the transform you created.

Configure your manual select transform and preview your data

1. Select your transform, then **Configure** from the toolbar. The Manual Select configuration work area opens. Optionally, you can:

i. Update the **Name**, **Description**, and select a **Color** for the transform to display in the work area.

2. Configure the **Timeout** setting. Timeout (in seconds) controls how long to wait before terminating a long running command. The default is defined in [application configuration](#) settings.

3. In the **Element Discovery Execution Behavior** drop down, you can optionally temporarily change the behavior from **Default** to **Top 1/Limit 1** or **Single Row**, as appropriate for your data source. This temporary limit allows you to build and test your data cube, then still run it against the full data set when you've made all adjustments needed.

 **Note:** Oracle managed data sources use **ROWNUM = 1** or **FETCH FIRST 1 ROWS ONLY** behavior.

4. Enable or disable subquery optimization.

i. When enabled, your query can be incorporated automatically into a larger query of data cube transformations, metric set aggregations, and filtering, to optimize performance by doing this work in the database. Your query must be able to be enclosed in a `SELECT` statement, and not end in a semicolon (;).

ii. Disable if you are executing a stored procedure, performing advanced or database-specific SQL features such as creating temporary tables.

iii. Disable if your data connector uses an ODBC driver that incorrectly reports its capabilities.

When disabled, transforms connected to this manual select's output display an icon to indicate the data is brought into memory to perform the transformation instead of being performed in the database.

5. Open the Data Preview tab to see your output. If your query is unsuccessful, Symphony may display informative errors and warnings.

 **Note:** When you've added, connected, and configured all elements, select **Check In** from the toolbar to make your data cube available to for use in your project and sharing with other users.

Add a manual select to an existing data cube



- Archive of documentation for Logi Symphony v24

1. With your data cube open for editing, select Manual Select from the toolbar. The Open dialog opens, allowing you to navigate to an appropriate data source.
2. Select a data connector. (For OLAP databases, instead select the cube under the data connector you will be querying.)
3. Enter your query in the script editor at the bottom of the data cube work area. Enter it manually, or use shortcuts to build it:



Note: Inserted comments in your query may cause an issue when [subquery optimization](#) is enabled for some data providers.

- i. The keyboard shortcut `CTRL + SPACE` opens a list of available SQL keywords and table names you can add to your query.
 - ii. Drag a structure from the Explore window to the script editor (cube, column, measure, hierarchy level) to add it to your by name.
 - iii. Drag a [stored procedure](#) from the Explore window to add it to the query.
4. Optionally, enter a placeholder in your parameterized SQL statement, with each parameter enclosed in dollar sign characters (\$). See [Using manual select placeholders](#).
 5. **Save** and configure the transform you created.

For more information, see:

- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)

Data Cube - Data Input

This applies to: *Managed Dashboards, Managed Reports*

When you create a data cube, you can reference a warehouse data storage area that contains user input data. Use the Data Input transform to add and modify data from inside Symphony. Input data directly, or set up a [data interaction input](#) to allow users with the [Data Input privilege](#) to add data.



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

Create a new data cube with a data input transform

1. Open the [data cube](#) work area. You can do this in one of several ways:
 - i. Select **Create New Data Cube** from the [home page](#), then **Data Input** from the Add/Edit menu
 - ii. Select **New > Data Cube** from the Managed Dashboards and Reports [main menu](#), then select **Data Input**
 - iii. Right-click **Data Cubes** in the Explore window, then select **New Cube** from the menu, then **Data Input** from the Add/Edit menu

The data cube work area opens, with a Data Input transform and connected Process Result.

2. Select your transform, then **Configure** from the toolbar. The Data Input configuration work area opens.
3. Select **Add Column** to open the Data Input Column work area and add your columns to contain input data. Provide a **Name**, **Data Type**, **Max Length**, and optional **Default Value**. When you have saved all of your new columns, close the work area to continue creating your data cube.



Note: By default, several columns are created and populated, but not selected to be included in the output: `RecordID`, `RecordSequence`, `CreatedBy`, `CreatedTime`, `LastModifiedBy`, `LastModifiedTime`. Optionally, select a column to check it and add it to the data output.

4. View your included columns, and enter data or add more columns in the Data Preview tab at the bottom of the data cube canvas.
 - i. Enter your data, and use the Tab key to create or move to the next cell, or the Enter/Return key to create or move to the next column.
 - ii. Right-click a cell or row to add, delete, or insert columns and rows.
 - iii. If you delete a column, you can't re-add a column with the same name until you have checked the data cube in and then out again.



- iv. Changes to rows of data are not affected by check in or check out states, as viewers or end users can be allowed to make changes.
- v. Undo Check Out does not change or remove data.

Add a data input to an existing data cube

1. With your data cube open for editing, select Data Input from the toolbar.
2. Select a data connector. (For OLAP databases, instead select the cube under the data connector you will be querying.)



Note: Data structures in your Symphony environment also include [data sources discovered](#) by Symphony in Managed Dashboard and Managed Reports data connectors, as well as shared from Data Discovery data connectors.

3. Select your transform, then **Configure** from the toolbar. The Data Input configuration work area opens.
4. Select **Add Column** to open the Data Input Column work area and add your columns to contain input data. Provide a **Name**, **Data Type**, **Max Length**, and optional **Default Value**. When you have saved all of your new columns, close the work area to continue creating your data cube.



Note: By default, several columns are created and populated, but not selected to be included in the output: `RecordID`, `RecordSequence`, `CreatedBy`, `CreatedTime`, `LastModifiedBy`, `LastModifiedTime`. Optionally, select a column to check it and add it to the data output.

5. Connect the Data Input to your ETL process to view the data in the Data Preview tab.
6. View your included columns, and enter data or add more columns in the Data Preview tab at the bottom of the data cube canvas.
 - i. Enter your data, and use the Tab key to create or move to the next cell, or the Enter/Return key to create or move to the next column.
 - ii. Right-click a cell or row to add, delete, or insert columns and rows.
 - iii. If you delete a column, you can't re-add a column with the same name until you have checked the data cube in and then out again.
- iv. Changes to rows of data are not affected by check in or check out states, as viewers or end users can be allowed to make changes.
- v. Undo Check Out does not change or remove data.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Data Cubes](#)
- [Data Cube - Data Input](#)
- [The Managed Dashboards Data Model](#)

Data Cube - Python Data Generator

This applies to: *Managed Dashboards, Managed Reports*

A data cube is a multidimensional model of your data that consists of measures, hierarchies, and metadata such as formatting. The Python Data Generator transform lets you generate data by writing scripts using the Python programming language. This is similar to the [Python Analysis](#) transform except that it does not accept input from a preceding transform and generates its own output directly from Python.



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

You can use the Python Data Generator transform to provide data to be used or visualized in Symphony. For example, Python can connect to and manipulate REST API data into a usable format, or generate data for prototyping or developing proof-of-concept dashboards.

To learn more about the Python language, see python.org.



Important: Depending on your software configuration, Python transforms may not be available in your environment.

The Python programming environment must be installed in your Symphony environment for you to use the Python Data Generator transform. See [Install Python](#).

Create a data cube with a python data generator transform

1. When creating a [new data cube](#), you can add the Python Data Generator transform to an empty canvas from the toolbar.
 - i. You can also create a new data cube from the main menu, and choose Python Data Generator in the dialog of options that appears. The Python Data Generation transform is added to the data cube and connected to a Process Result transform automatically.
 - ii. You can also add the Python Data Generator transform from the toolbar to an existing data cube process, and use a Union transform or another transform that combines data from multiple inputs.
2. Configure your transform by selecting **Configure** from the right-click menu of the transform. Enter a Python Script that returns a result. Optionally select **Edit script** to use the Script Editor window for more space and to access [other helpful features](#).



Important: The result must be in a format that can be represented as a table, and data cubes will be unable to use Python outputs such as Print or Draw.



Note: Use the Print command to print variables to the application logs for debugging. To view these entries, ensure that the **Log Filter** configuration setting for Data Cubes is set to **Information**.



3. Set up Placeholders to insert into the script that pass in parameter values similar to when using a [manual select](#). Insert the placeholder's Identifier (beginning and ending with the \$ symbol) into your script to access the current parameter value. Type the \$ symbol or right-click for a popup list of the current placeholders, then use the keyboard or mouse to select and insert it.
4. Select the **Use Pooling** option to reuse Python processes between execution of Python scripts to improve performance compared to starting a new process each time. There are [corresponding configuration settings](#) for Symphony administrators in the Python Pool category to manage the number of processes and their memory consumption.
5. The output of the Python Data Generator depends on the script it is configured with. It can be a single value, a column of values, or multiple columns.



Note: You can also add a Python Data Generator input transform to an existing data cube as needed when structuring the ETL process in that data cube. For more information, see [Python Data Generator](#).

For more information, see:

- [Data Cubes](#)
- [Data Cube - Python Data Generator](#)
- [The Managed Dashboards Data Model](#)

Data Cube - R Data Generator

This applies to: *Managed Dashboards, Managed Reports*

A data cube is a multidimensional model of your data that consists of measures, hierarchies, and metadata such as formatting. The R Data Generator transform lets you generate data by writing scripts using the R statistical programming language. This is similar to the [R Language Analysis](#) transform except that it does not accept input from a preceding transform and generates its own output directly from R.

R is both a programming language and an environment for statistical computing, graphics, and predictive analysis. You can use the R Data Generator transform to generate data for prototyping or developing proof-of-concepts, or if you are using R to access a data source.

To learn more about the R language, see [The R Project for Statistical Computing](#).



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

Create a data cube with an R Data Generator transform

1. When creating a [new data cube](#), you can add the R Data Generator transform to an empty canvas from the toolbar.
 - i. You can also create a new data cube from the main menu, and choose R Data Generator in the dialog of options that appears. The R Data Generator transform is added to the data cube and connected to a Process Result transform automatically.
 - ii. You can also add the R Data Generator transform from the toolbar to an existing data cube process, and use a Union transform or another transform that combines data from multiple inputs.
2. Configure your transform by selecting Configure from the right-click menu of the transform. Enter an R Script that sets an output variable.
3. Set up Placeholders to insert into the script that pass in parameter values similar to when using a [manual select](#). Insert the placeholder's Identifier (beginning and ending with the \$ symbol) into your script to access the current parameter value. Type the \$ symbol or right-click for a popup list of the current placeholders, then use the keyboard or mouse to select and insert it. You can also set up Parameters to directly filter this transform's output as with [select transforms](#).
4. The output of the R Data Generator depends on the script it is configured with. It can be a single value, a column of values, or multiple columns.



Note: You can also add an R Data Generator input transform to an existing data cube as needed when structuring the ETL process in that data cube. For more information, see [R Data Generator](#).

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Data Cubes](#)
- [Data Cube - R Data Generator](#)
- [The Managed Dashboards Data Model](#)

Data Cube Data Storage Options

This applies to: *Managed Dashboards, Managed Reports*

A data cube is a multidimensional model of your data that can optionally include a Storage Type to potentially increase data retrieval performance or reduce the load on your database servers or other data sources. There are three storage options available from the toolbar when you [create](#) or [edit](#) a data cube:

- **None - (Default)** Connects to data sources directly when the data cube is used in a visualization. You can opt for [short term caching or real-time updates](#) of the live data.
- **Warehouse** - Stores or caches the transformed data in the Symphony Warehouse database when the cube is built.
- **In-Memory** - Stores data in server memory for even greater performance potential for data analysis queries.

When you select warehouse or in-memory storage, you can build the storage on-demand or schedule it to run periodically, in off-peak hours. Additionally, you can select to perform incremental updates, and retrieve only data added to the database since the last build. Incremental updates can potentially help you build large data cubes faster.



Important: A data cube that contains public parameters or user-specific filters cannot be built as a warehouse or in-memory cube. A [security hierarchy](#) can be used instead of filtering by custom attributes.

For more information, see:

- Video: [Introduction to Data Cubes](#)
- Video: [Data Cleansing with Data Cubes](#)
- [Data Cube Storage Preparation and Use](#)
- [The Managed Dashboards Data Model](#)

Data Cube Storage Preparation and Use

This applies to: *Managed Dashboards, Managed Reports*

When you select warehouse or in-memory storage, you can build the storage on-demand or schedule it to run periodically, in off-peak hours. Additionally, you can select to perform incremental updates, and retrieve only data added to the database since the last build. Incremental updates can potentially help you build large data cubes faster.



Important: A data cube that contains public parameters or user-specific filters cannot be built as a warehouse or in-memory cube. A [security hierarchy](#) can be used instead of filtering by custom attributes.

Data Cube Preparation

The in-memory storage option in particular can require a lot of server memory, so if you choose this type, we strongly recommend that you remove or hide elements (columns, hierarchies, or measures) that you do not need from your data cube's [Process Result](#) transform. This can also benefit the warehouse storage type by leaving out unnecessary data.

To completely remove columns from the data cube output, double-click the **Process Result** transform to open its configuration dialog, and then uncheck output columns that you don't need.

Alternatively, click on **Process Result** to open the **Data Cube Elements** pop-up where you can click the gray **minus** icon on the right to hide elements. Those elements will be moved and listed below under **Hidden**, where you can unhide it by clicking its **plus** icon. A hidden column will be excluded from storage but still available for linking in hierarchy keys (for example, if you need to use the data cube later to define a hierarchy by using the column as part of a composite key).



Process Result

Data cube elements



Modify, drop, or replace the output hierarchies and measures by clicking the edit icon.

Measures

OrderQty







Hide Element

Hierarchies

ProductID





SalesOrderID





SalesPersonID



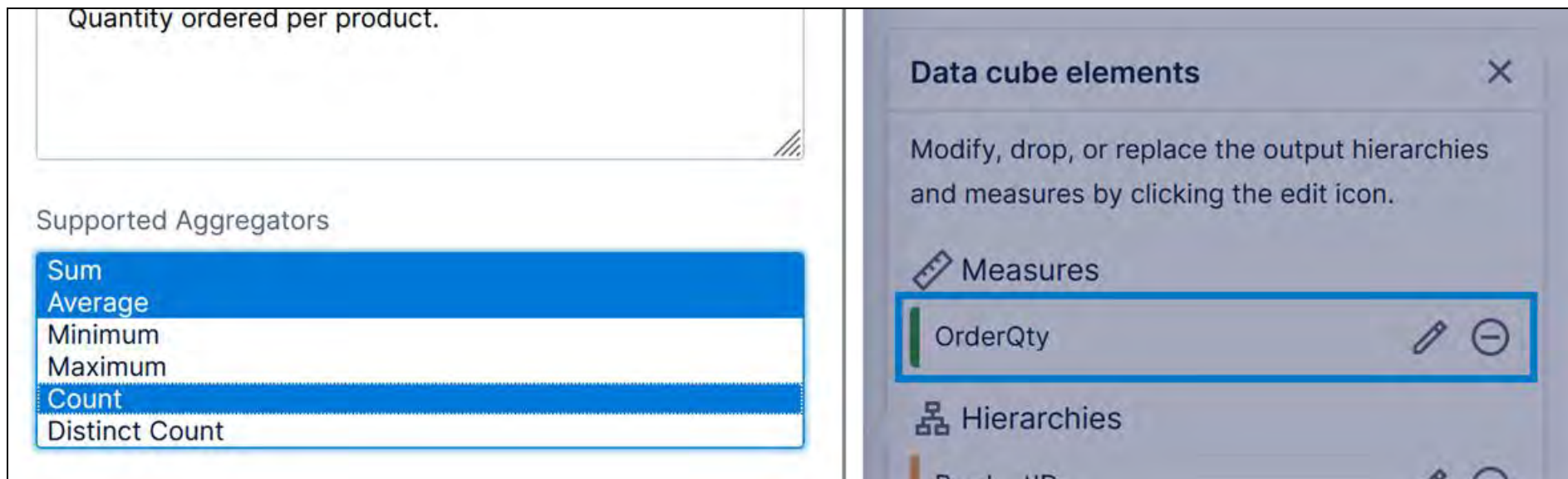


Hidden

OrderDate



In-memory storage is also affected by the number of **aggregators** enabled on each measure. For example, edit a measure from your data cube and check the list of **Supported Aggregators**: by default, only **Sum**, **Count**, and **Average** are selected in order to conserve memory usage, but you can adjust this setting as needed.



The screenshot displays two panels from the Logi Symphony interface. The left panel, titled 'Quantity ordered per product.', shows a list of 'Supported Aggregators' with the following options: Sum, Average, Minimum, Maximum, Count, and Distinct Count. The right panel, titled 'Data cube elements', provides instructions to 'Modify, drop, or replace the output hierarchies and measures by clicking the edit icon.' It features a 'Measures' section where 'OrderQty' is listed with an edit icon (pencil) and a minus icon. Below this is a 'Hierarchies' section.

Elements that provide details about or describe items identified by another hierarchy can be selected as **Attribute Hierarchies** for that hierarchy. This conserves the amount of in-memory storage required, while providing benefits when selecting data such as disabling unwanted totals and expand/collapse functionality by default.

Attribute Hierarchies

Class
Color
DiscontinuedDate
FinishedGoodsFlag

Close
Save

For example, with a **ProductID** that uniquely identifies each product, you could select various elements such as **Color**, **Class**, and **Size** as attribute hierarchies for **ProductID** that describe each product.

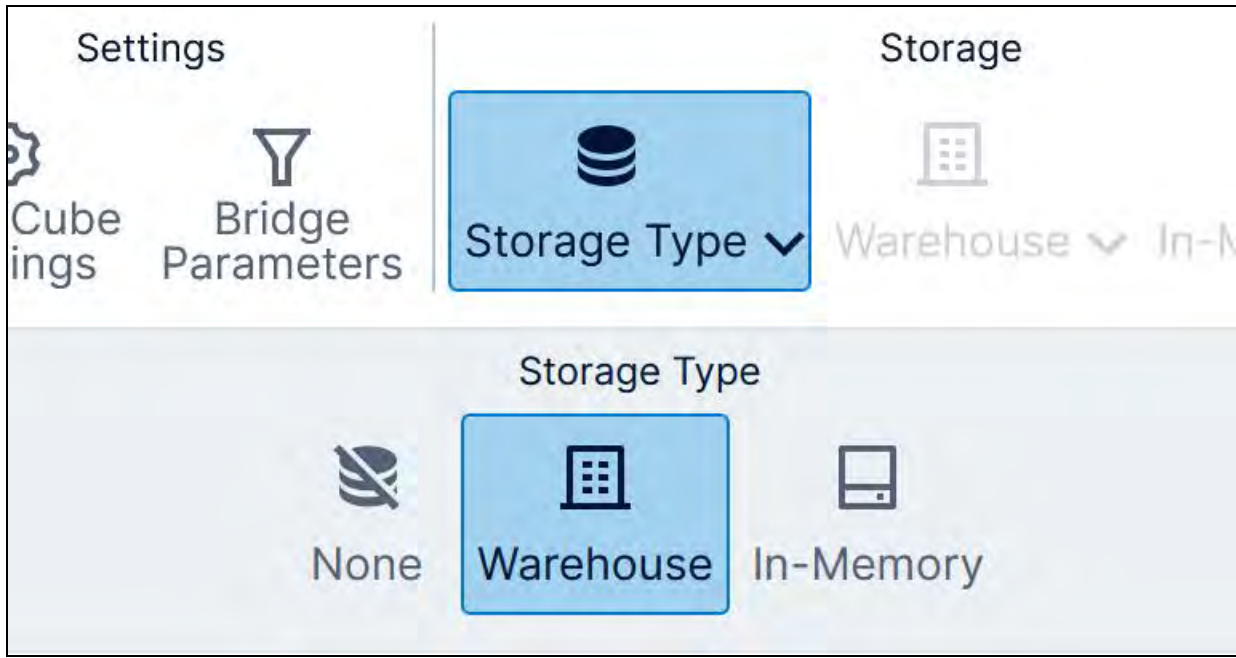
Attributes can also be set up when creating a predefined hierarchy as shown in Automatic joins and hierarchies.



Note: More details and tips for improving performance for in-memory cube storage are provided below.

Using Data Cube Storage

When editing a data cube that is checked out to you, click **Storage Type** in the toolbar and select the type you want.



The **Warehouse** storage type lets you store or cache data in the Symphony Warehouse database. The **In-Memory** storage type stores data in server memory for even greater performance gain potential for data analysis queries, and also stores its data in the Symphony Warehouse database so that it can be reloaded back into memory in the event of a server restart.

Check in Your Data Cube

Once you've chosen a storage type, you must **Check In** your data cube before you can build its storage or set up scheduling.

After checking in, you will be asked if you want to build the data cube. Click **Ok** to begin building the warehouse or in-memory storage, or click **Cancel** if you want to build it later as shown in the next section.

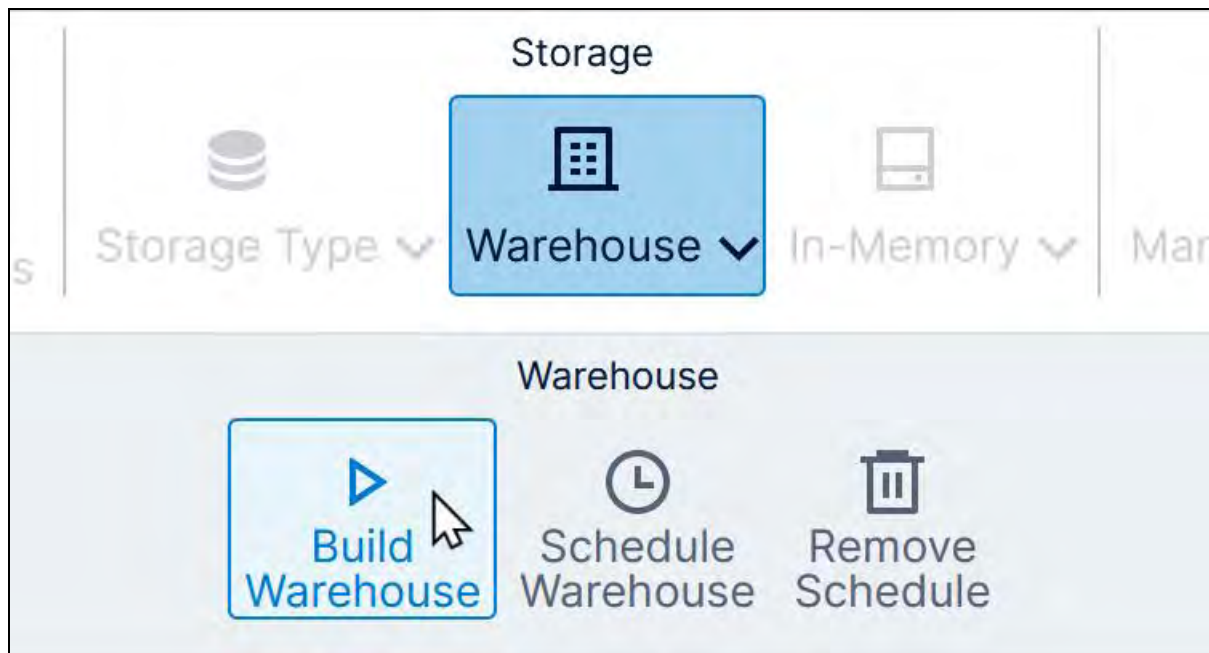
Next, you will be asked if you want to set up a schedule. Click **Ok** to open the **Schedule Rule** dialog, otherwise click **Cancel** and you can still choose to set up a schedule later.

Build the Data Cube

After checking in the data cube, you can also build its warehouse or in-memory storage on-demand any of these three ways:

- When the data cube is open for editing but still checked in, click **Warehouse** in the toolbar and then **Build Warehouse**, or click **In-Memory** and then **Build In-Memory**.

- Locate the data cube within the **Data Cubes** folder (navigate there from the **Data** option in the [main menu](#) or an [Explore window](#)), right-click the data cube, and choose **Build Warehouse** or **Build In-Memory**.
- If you are a Symphony administrator, you can use the [Managing Jobs](#) or [Managing Data Cubes](#) screens to run the data cube storage job.



You'll see a message indicating the build has started. The time required to build the cube depends on the amount of data and other factors described elsewhere in this article.

You can check on the status of the build from the **Data Cubes** dialog, which you can access from your [Data Cube Status](#).



Important: If you [check out](#) the data cube later, its data will be displayed for you live until you [undo your check out](#) or check in again and re-build the data cube with your changes. (This does not affect other users, who can only see the checked-in version.)

Schedule Data Cube Storage

To open the **Schedule Rule** dialog for scheduling, open the data cube for editing while it is still checked in.

Click **Warehouse** in the toolbar and then **Schedule Warehouse**, or click **In-Memory** and then **Schedule In-Memory**.

In the dialog, choose from a wide variety of scheduling options to have the data cube periodically re-build its storage automatically, then click **Submit** at the bottom.

Schedule Rule

Set up how this event should repeat including start and end dates.

Description

At 12:00 AM on day(s) 10 of every month, starting 2024-07-01, running time adjusted to (UTC-05:00) Eastern Time (US & Canada)

Enabled



Time Zone

Default - Server Time Zone (Eastern Standard Time) ▼

Daylight Savings Time



Administrators can also go to the [Managing Jobs](#) or [Managing Data Cubes](#) screens to schedule data cube builds.

Job Failure

You can confirm if the data cube storage processing was successful for data cubes you have access to from your [profile](#).

The [Configure Job Failure Emails](#) set up by your administrator may be configured so that someone is notified by email if processing the data cube storage fails. This can be configured so that you can choose which user to notify for each data cube using storage:

1. Click **Data Cube Settings** in the toolbar when editing a data cube that is checked out to you and is set to use storage.
2. If available in the dialog that opens, you can set **Notified On Failure** to a user that should receive these emails.

There is also a configuration setting that administrators can change to [Automatically Retry Jobs](#), including for data cube storage.

Disable Memory Management

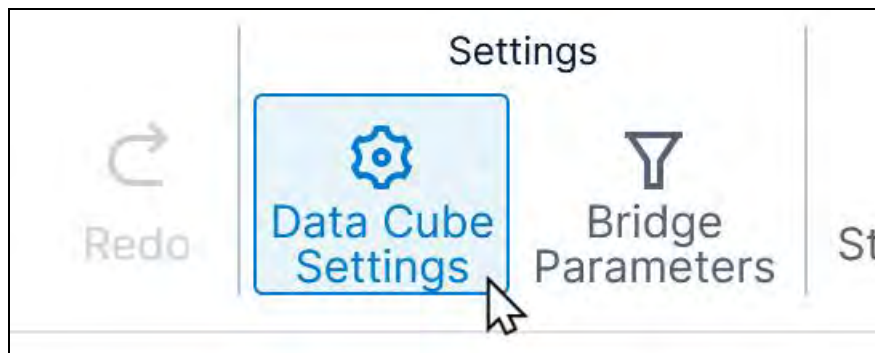
When using in-memory storage, after the storage exceeds a certain number of records, it begins compressing the records to maintain a stable use of RAM. When the records are used, they are decompressed and then recompressed again after some time.

You can increase the performance of the in-memory data cube by preventing it from recompressing records.



Important: This setting allows for unlimited use of RAM. Using it with an insufficient amount of RAM can result in memory paging, which would likely lower the performance rather than improve it.

Click to configure the **Data Cube Settings** in the toolbar when editing an in-memory data cube that is checked out to you.



In the **Data Cube Settings** dialog, tick the check box under **Disable Memory Management**. Then click **Submit**.

The new setting will take effect after the next time the data cube is built.



Important: As an alternative, you can change the number of records after which compression starts by navigating to **Config** settings as a system admin and editing the **Record Block Size** configuration. Navigate to **Admin > Setup > Config** to edit the Record Block Size.

Administrator Options

If you are a Symphony administrator, you can also access additional options that allow you to manage data cube storage for multiple data cubes at once, including running jobs, creating batched sequence jobs, changing storage types, scheduling, and automatically building storage for all referenced data cubes. Under **Projects / File System**, click **Data Cubes** to access these options. See [Manage Projects and the File System](#).

You can also manage each data cube's storage job from the **Jobs** page in Administration as shown in [Monitoring system health](#).

Limitations

The following limitations exist for in-memory warehouse storage:

- In-memory storage does not support binary data types.
- In-memory storage relies on [the double numeric type](#), which may approximate rather than guarantee accuracy when displaying a large number of significant digits (e.g., many digits to the right of the decimal point).

Incremental Data Update

Incrementally building data cubes will only retrieve data that was added to the database since the last build as kept track of by one or more **incremental element fields** in your data, such as a date or sequential ID. This has the potential to substantially decrease the time needed to build the data cube, however it will not update or remove existing data.

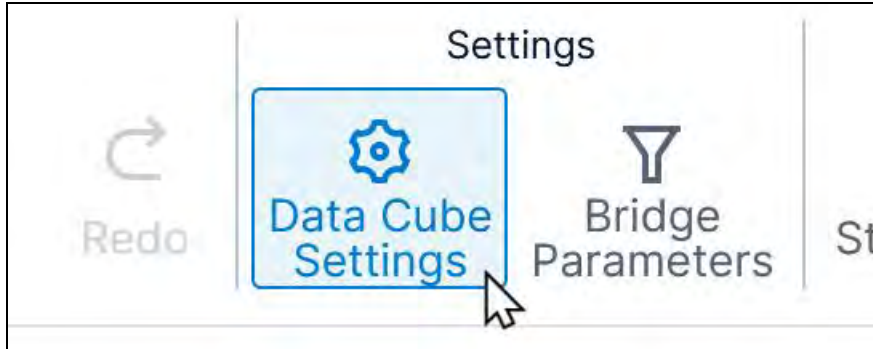
It is important to avoid or be cautious when using incremental updates on a data cube where changes to existing records or rows in the data are possible. For example, records deleted in the source will remain in the cube, records with the same or a smaller incremental element value will not be updated, and records updated with a larger incremental element value can appear as a new, separate record after the data cube has been re-built.



Important: Incremental data updates are not available for data structures that do not allow parameters, such as Excel.

You can enable incremental data updates when using either the warehouse or in-memory storage type, and then build the data cubes manually or via a schedule, as described above.

Click to configure the **Data Cube Settings** in the toolbar when the data cube is checked out to you.



Select **Perform Incremental Build**, and then select the **Incremental Elements** to use to identify new records. Each time the data cube builds, it will only load new records where the selected values are greater than those from the previous build.

Data Cube Settings

Configure the various available settings for this data cube that are not specific to any transform. These settings affect the entire data cube, and may change depending on the data cube's storage type.

Perform Incremental Build



Incremental Elements

- ▼

☐ [Sales].[SalesOrderDetail]

 - ☐ SalesOrderID (Int32)
 - ☐ OrderQty (Int16)
 - ☐ ProductID (Int32)
- ▼

☒ [Sales].[SalesOrderHeader]

 - ☒ SalesOrderID (Int32)
 - ☒ OrderQty (Int16)
 - ☒ ProductID (Int32)



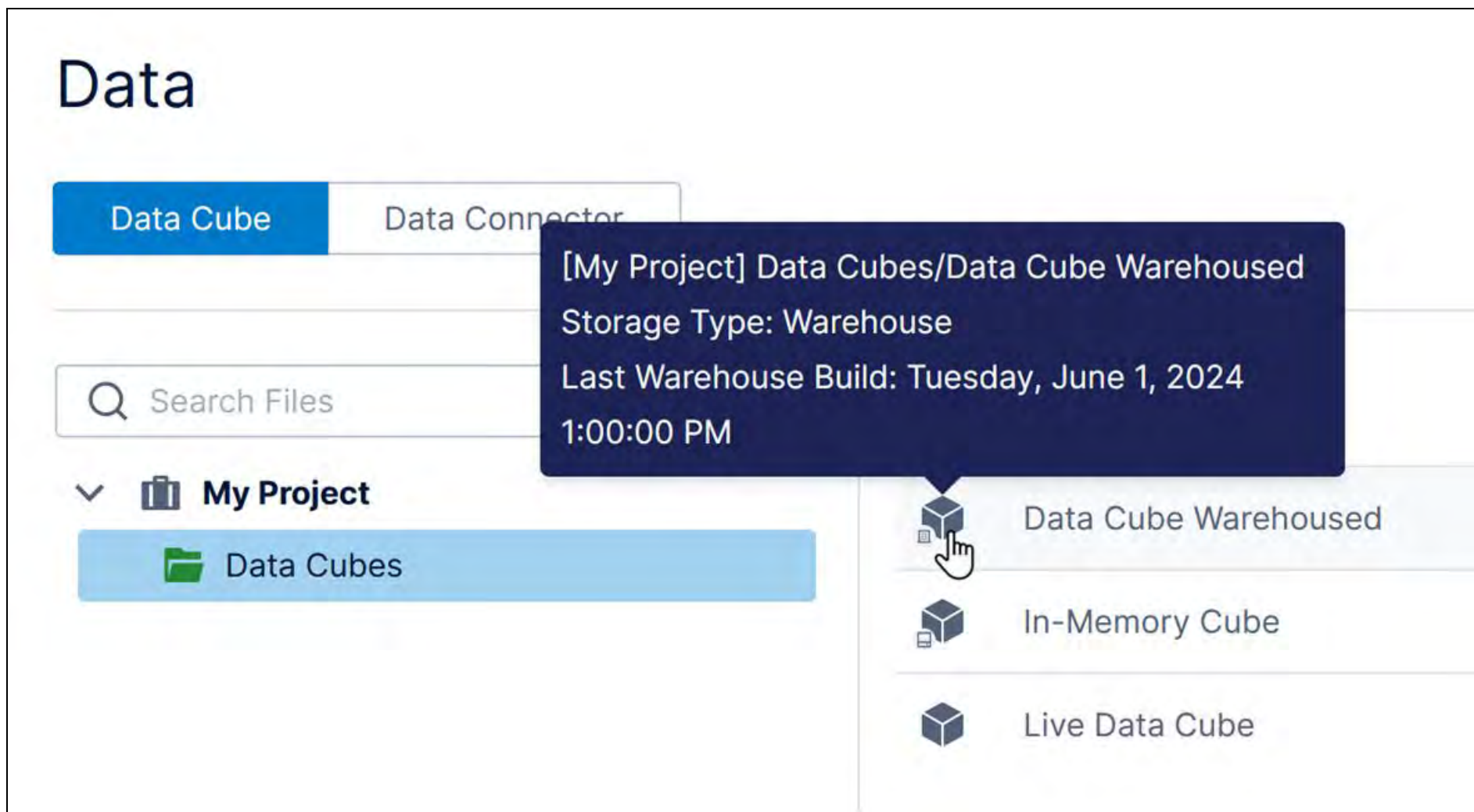
Configuration Notes

After you've built your data cube storage, you can check the storage type and status, review in-memory build performance, and work with your data as raw data.

Check the Storage Type and Status

If your data cube is checked in, you can identify its storage type from the [Explore window](#) or other similar file explorers.

Data cubes appear with a distinct icon indicating its storage type as shown below. The tooltip that appears when hovering over it (or long-tapping) also describes its storage type.



Data

Data Cube Data Connector

Search Files

My Project

Data Cubes

[My Project] Data Cubes/Data Cube Warehoused
Storage Type: Warehouse
Last Warehouse Build: Tuesday, June 1, 2024 1:00:00 PM

Data Cube Warehoused

In-Memory Cube

Live Data Cube

If available, the last build time is also displayed in the tooltip above, and in the 1's status bar when the data cube is open (not available for some data cubes on multi-tenant instances).

You can also view all the data cubes you have access to from the [Data Cube Status](#), along with their storage types, schedules, and build statuses.

In-Memory Build Performance

The total time required to build in-memory storage for a particular data cube, and the RAM memory required, can vary greatly depending on a range of factors. Estimating build times is therefore not as simple as saying building X records takes Y minutes, etc.



For example, memory consumption generally depends on:

- How memory is allocated by .NET
- The number of hierarchies and measures used in your data cube process (especially with distinct count)
- How sparse your fact table is (end result of your data cube process)

Ideally, if the in-memory storage result fits entirely in RAM, then the total build time will generally be divided as follows on a typical 8 core CPU box:

- 50% of time spent retrieving/loading data from your data sources
- 50% of time spent building the in-memory data structures

With a higher number of processing cores, the time to build the in-memory data structures can be reduced due to parallelization.

Here are some general guidelines for optimizing in-memory builds:

- De-select columns you don't need in your data cube as early in the process as possible.
- Minimize the number of hierarchies, measures, and aggregators in your data cube.
- Include columns as attributes of a hierarchy where appropriate instead of separate hierarchies.
- Minimize the complexity of your data cube process (e.g., type and number of transforms used).
- Use the [Filter transform](#) to restrict data to required ranges.

Raw Data

When the Data Retrieval Format on a metric set is set to **Raw** and **<Row Number>** is placed on Rows, the raw data may be retrieved directly from the original database, bypassing the data cube's storage.

See [Adding More Data](#).

Optimize Storage Jobs

If you have a very large number of data cubes using warehouse or in-memory storage, you could reach a limit to the number of their storage jobs that can successfully run simultaneously for multiple potential reasons, including the volume of data, database configurations, and your server's hardware. To avoid issues



when building data cube storage for many cubes at once, try one or more of the following:

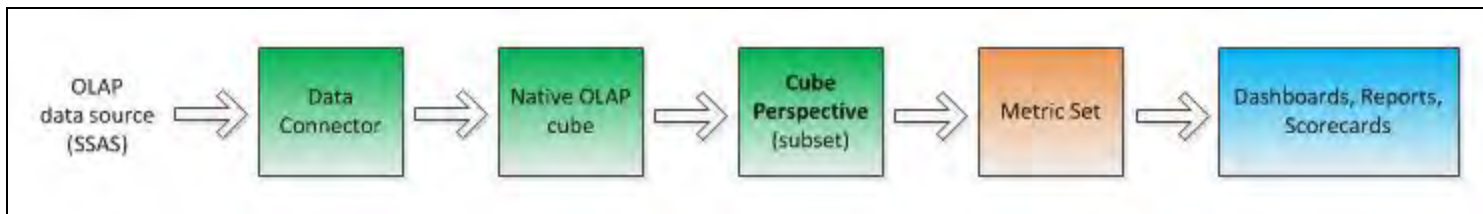
- Avoid scheduling or manually running all of the storage jobs at the same time: stagger each job's run time to allow time for some or all of the previous jobs to complete, and/or use [the batching feature](#) to schedule a special sequence job that will automatically build each selected cube in sequence.
- Increase the value of [the max_connections setting](#) in PostgreSQL for the warehouse database.
- Change the Data Warehouse Connection String in your [configuration settings](#) to specify a higher [Maximum Pool Size](#) parameter value.

Create a Cube Perspective

This applies to: *Managed Dashboards, Managed Reports*

When you create a data connector using an OLAP provider such as SQL Server Analysis Services, you gain access to native OLAP cubes and their measures and dimensions. However, you may not want to expose all of these elements (e.g., entire cubes) to other users.

As a *Developer* user, you can create a *cube perspective* which defines a subset of the available measures and dimensions from a native OLAP cube. You then give *Power users* access to this cube perspective instead of the full (native) cube so they can build metric sets and dashboards.



Related video: [Understanding Cube Perspectives](#).

Creating a Cube Perspective

Connect to an OLAP Data Source

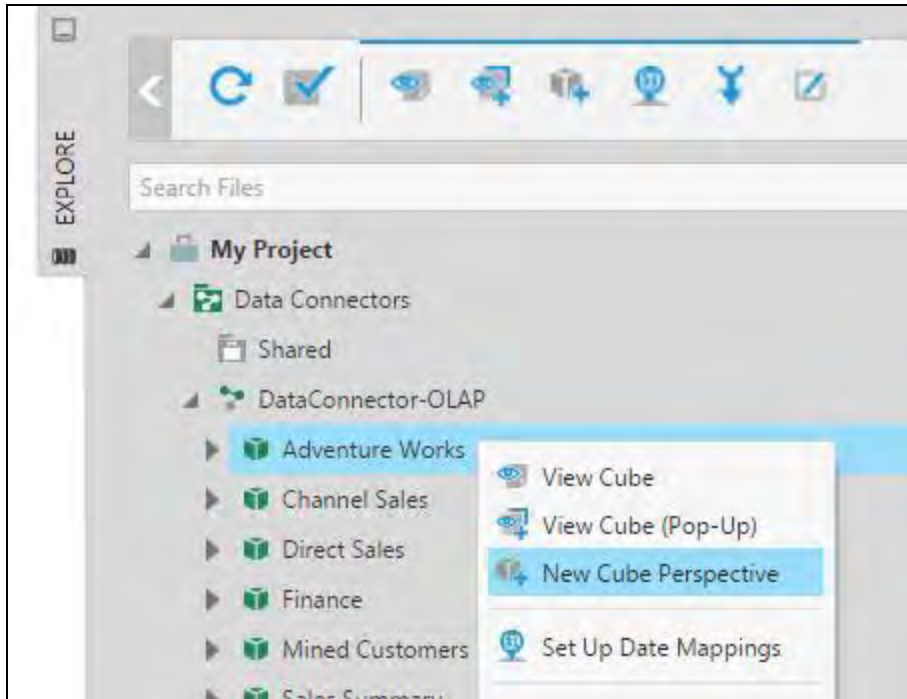
Before creating a cube perspective, you need to connect to an OLAP data source which gives you access to a native OLAP cube. See the first part of [Connect to OLAP data and apply a formula](#) for the steps.

Create a New Perspective

If available, you can use the [main menu](#) to create a new cube perspective under Data. (This option may not appear if there are no existing cube perspectives.)

You can also create a new cube perspective from the Explore window when you have another file open such as a dashboard or metric set:

1. Expand your OLAP data connector and right-click on a native OLAP cube (e.g., AdventureWorks).
2. From the context menu, select **New Cube Perspective**.
3. This opens the Cube Perspective editor with the selected OLAP cube already loaded (so you can skip the next step).



Drag and Drop a Native OLAP Cube

If the new cube perspective is blank because you didn't already choose a cube, go to the Explore window, find your OLAP data connector and expand it to see its available cubes.

Drag-and-drop a cube from the Explore window to the canvas.

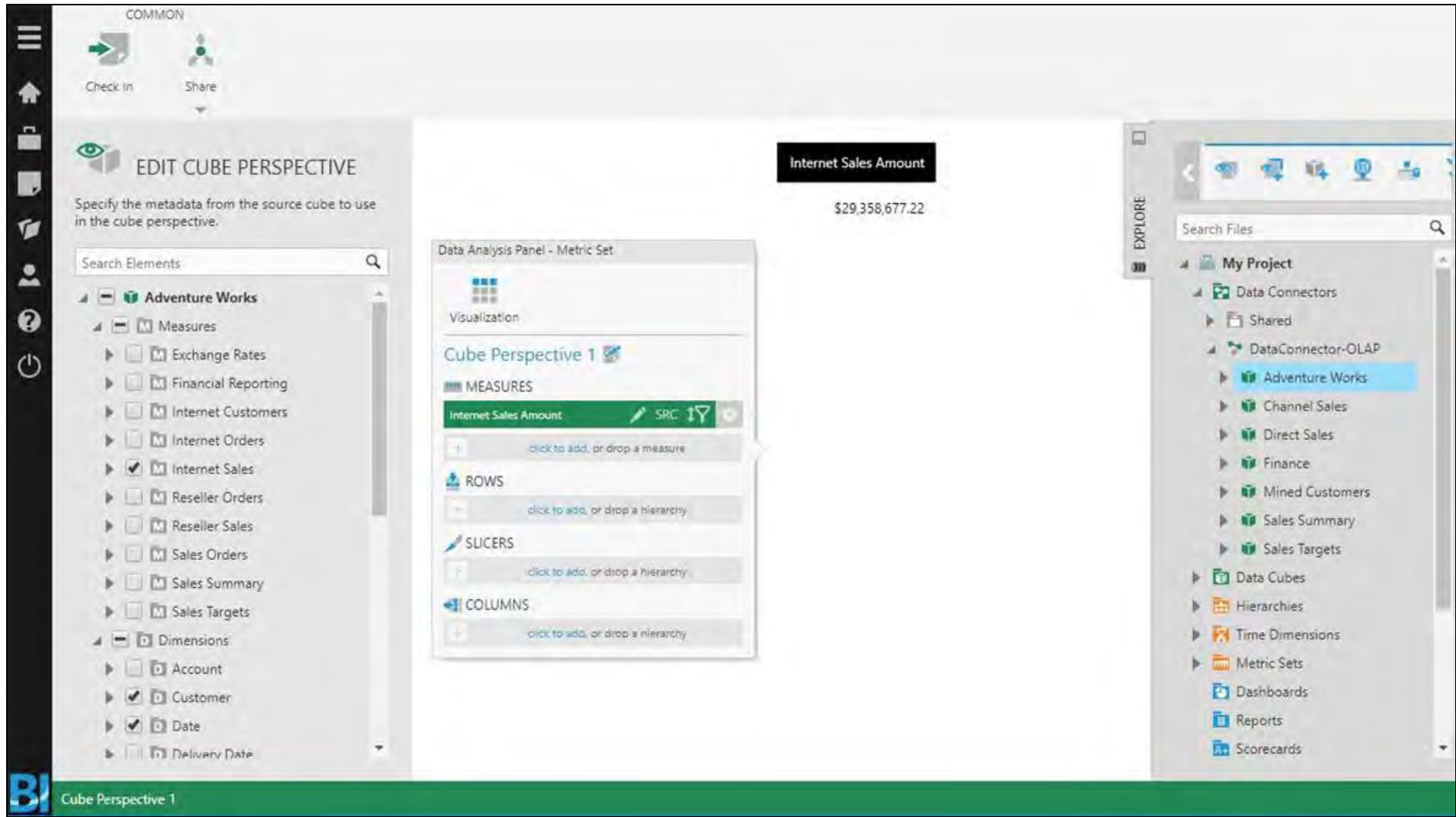


Select Measures and Dimensions

An Edit Cube Perspective panel on the left will list the measures, dimensions, and hierarchies of the native cube, often in folders that you can click to expand.

Use the check boxes to select and deselect the elements of the native cube that you want included in this cube perspective.

The Data Analysis Panel and table visualization in the middle part of the screen let you preview measures and dimensions while selecting them. This is for your convenience and is not saved. You can drag measures or hierarchies onto the Data Analysis Panel or **click to add** them to see your data.



COMMON

Check In Share

EDIT CUBE PERSPECTIVE

Specify the metadata from the source cube to use in the cube perspective.

Search Elements

- Adventure Works
 - Measures
 - Exchange Rates
 - Financial Reporting
 - Internet Customers
 - Internet Orders
 - ☒ Internet Sales
 - Reseller Orders
 - Reseller Sales
 - Sales Orders
 - Sales Summary
 - Sales Targets
 - Dimensions
 - Account
 - ☒ Customer
 - ☒ Date
 - Delivery Date

Data Analysis Panel - Metric Set

Visualization

Cube Perspective 1

MEASURES

Internet Sales Amount SRC: 1

+ click to add, or drop a measure

ROWS

+ click to add, or drop a hierarchy

SLICERS

+ click to add, or drop a hierarchy

COLUMNS

+ click to add, or drop a hierarchy

EXPLORE

Search Files

- My Project
 - Data Connectors
 - Shared
 - DataConnector-OLAP
 - Adventure Works
 - Channel Sales
 - Direct Sales
 - Finance
 - Mined Customers
 - Sales Summary
 - Sales Targets
 - Data Cubes
 - Hierarchies
 - Time Dimensions
 - Metric Sets
 - Dashboards
 - Reports
 - Scorecards

Internet Sales Amount

\$29,358,677.22

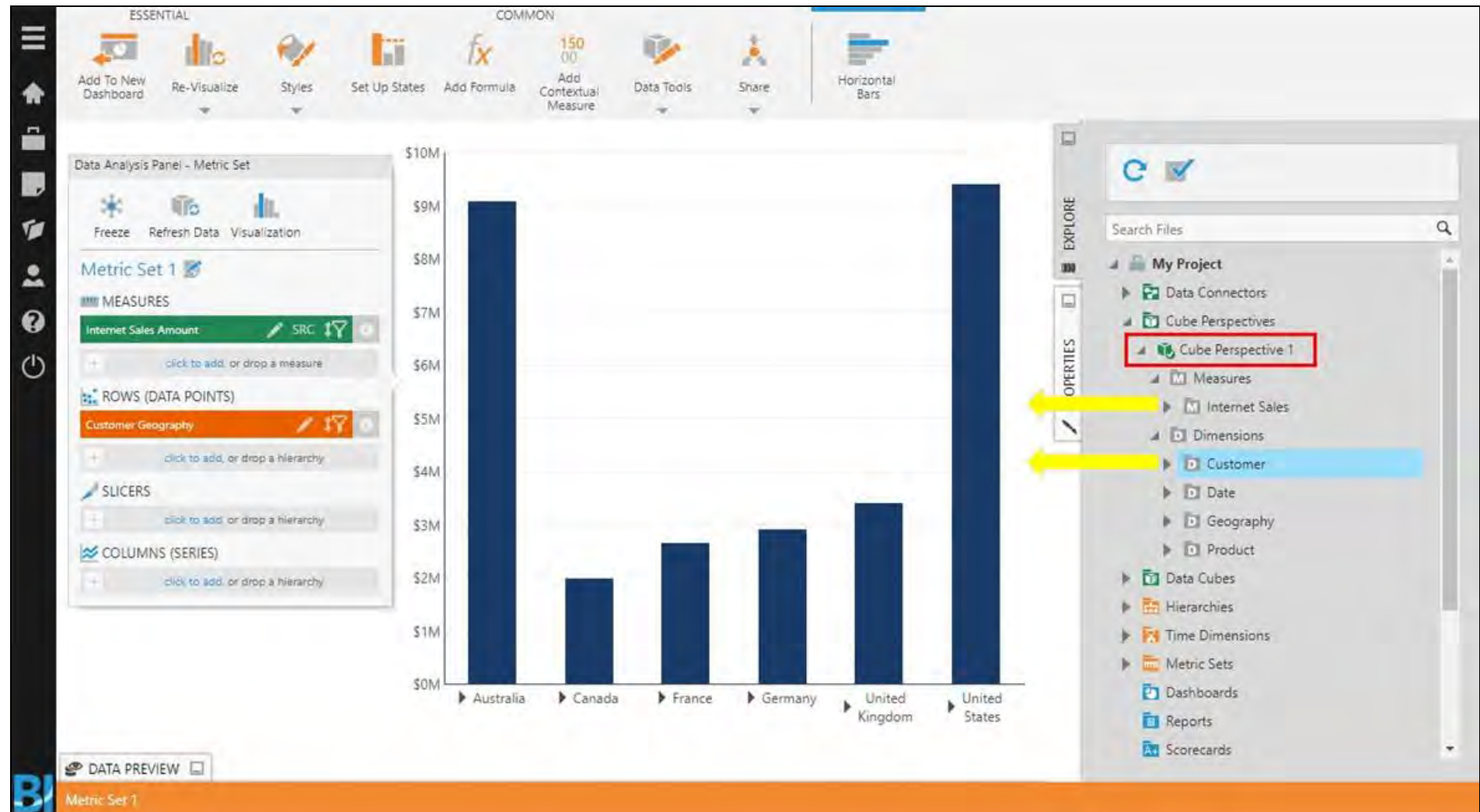
BI Cube Perspective 1

Check in Your Changes

Once you are done with the check box selections, go to the toolbar and click **Check In** to [let others see and use the cube perspective](#).

Using a Cube Perspective

Cube perspectives can be used in the same way as native cubes. For example, simply drag measures and hierarchies from a cube perspective in the Explore window to the canvas of a metric set or dashboard to display a table or chart.

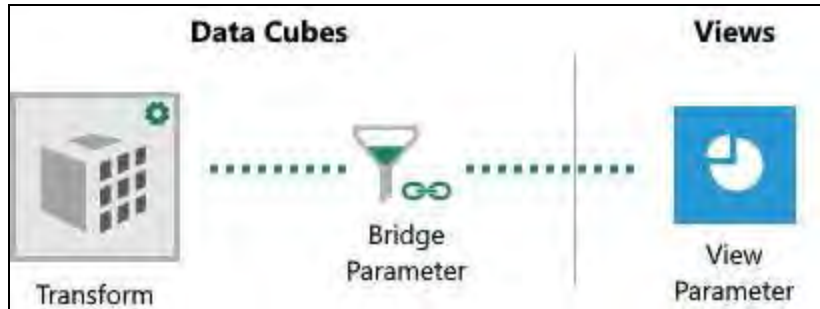


Using a Bridge Parameter

This applies to: *Managed Dashboards, Managed Reports*

When you want to pass a parameter value from a dashboard or view directly into one of the transforms in your data cube, bridge parameters are an optional intermediate step where you can convert or customize the parameter value.

You can use a bridge parameter to convert values from a hierarchy or another type not directly compatible with your query or transform. For example, a hierarchy value filter on a dashboard selects an entire hierarchy **member value**, which your database query or stored procedure won't be able to recognize unless it's first converted.



Once you click **Create parameter** in the dialog for a new bridge parameter, each transform parameter in your data cube is listed with a check box. Select a **check box** to open a dialog with a script editor where you can convert the bridge parameter input value type to the **type** used by the transform parameter and then return it. The types of this conversion are listed above the script editor (e.g., *SingleMember* -> *Sing/eNumber*).



Linked Parameter

Configure the link by writing the expression to convert the value for the target parameter. The expression is responsible for changing the bridged parameter type to the linked parameter type.

Parameter

Return Value Type

Default Value

Script (SingleMember -> SingleNumber)

Ln: 1 Col: 1

```

1 //if bridge parameter is passing its value to the attached parameter uncomment below
2 //return $input$;
3
4 //if bridge parameter needs to return a SingleNumberValue uncomment and modify below (yo
5 //return new SingleNumberValue($id$, $input$.Value);
6
7 //if bridge parameter needs to return a RangeDateTimeValue uncomment and modify below
8 //return new RangeDateTimeValue($id$, Convert.ToDateTime($input$.Value), true, Convert.T
9
10 //if bridge parameter needs to return a CollectionStringValue uncomment and modify below
11 //return new CollectionStringValue($id$, ["A","B","C"]);
12
13 //examples of SingleParameterValue properties
14 //$input$.Value
15 //$input$.Value.UniqueName

```



The script language used in bridge parameters is [DundasScript](#), which is also used in other parts of the application such as [calculated elements](#) and is similar to C#. The script receives one of the derived types of [ParameterValue](#) as the *\$input\$* placeholder, which will correspond to the Value Type you selected when creating the bridge parameter. Use the Test Value filter at the bottom to select a value, then click **Test script** to see a live result or an error message indicating where in the script there was a problem.

Examples

The following examples provide step-by-step instructions for some of the common uses of bridge parameters. Unless otherwise specified, all of these examples are based on the [Adventure Works sample SQL Server database](#).

Multi-Level Hierarchy With Single Selection

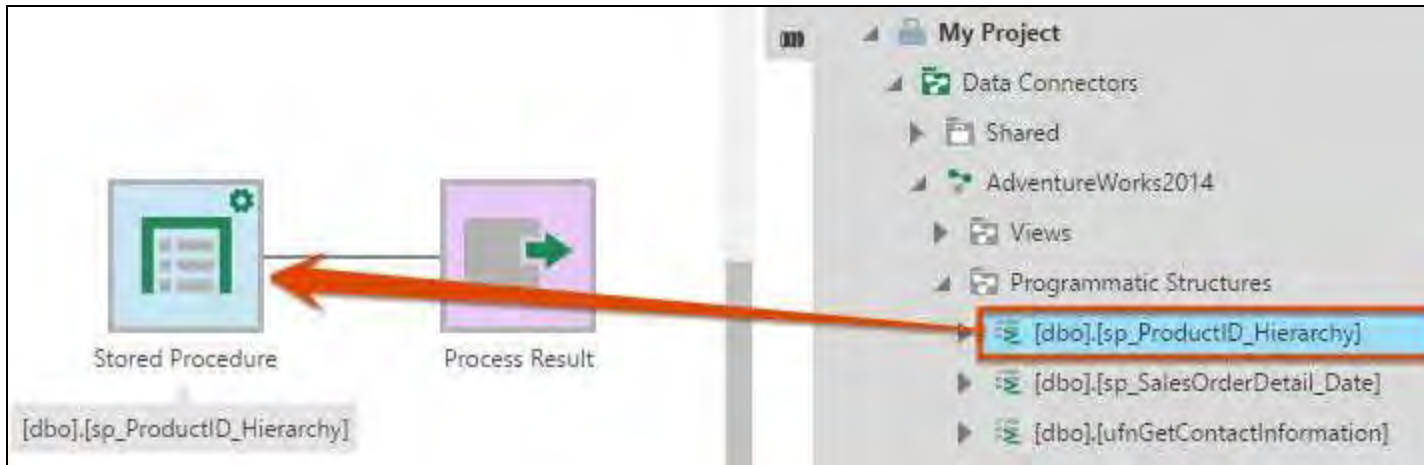
In this example, the bridge parameter is used to convert a single member from a hierarchy value filter into an int numeric value used by a stored procedure.

We have created this stored procedure using the following command in the Adventure Works database, which includes the parameter *@ProductID*.

```
CREATE PROCEDURE [dbo].[sp_ProductID_Hierarchy]
    @ProductID int
AS
BEGIN
    SET NOCOUNT ON;
    SELECT [ProductID] AS ProductID, [OrderQty] as OrderQty
    FROM [Sales].[SalesOrderDetail]
    WHERE [ProductID] = @ProductID OR @ProductID IS NULL
END
```

This example uses a Product hierarchy with three levels like the one demonstrated in [Automatic joins and hierarchies](#).

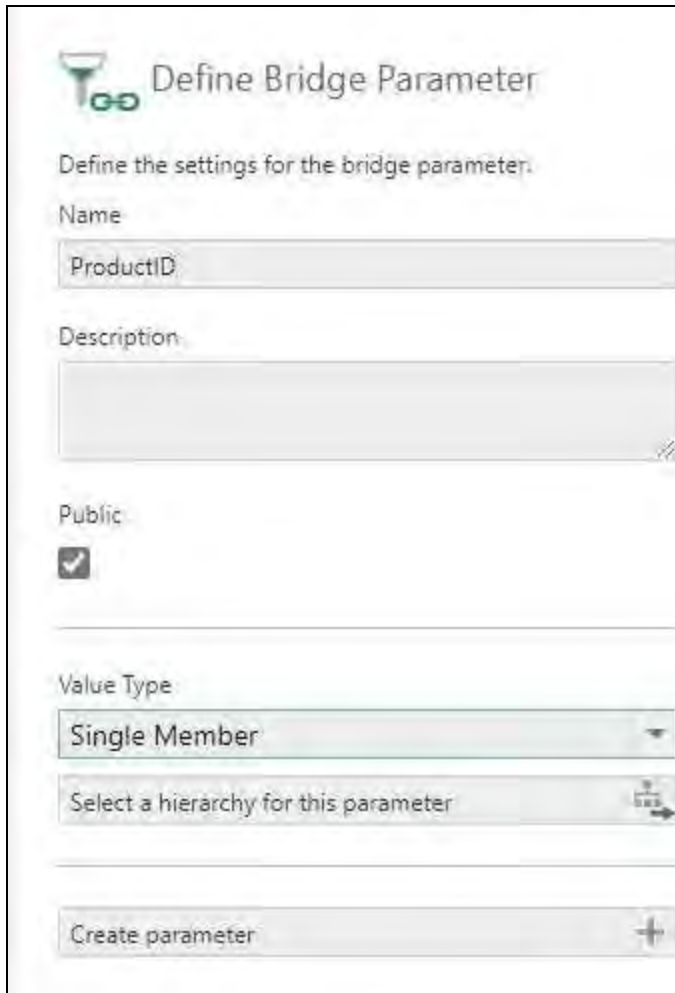
Create a new data cube and drag the *sp_ProductID_Hierarchy* stored procedure to the canvas. If you added the stored procedure after creating your data connector, right-click it and choose **Re-Discover**.



Click **Bridge Parameters** in the toolbar.



Click **Add parameter**, which opens the Define Bridge Parameter dialog.



Define Bridge Parameter

Define the settings for the bridge parameter.

Name
ProductID

Description

Public
☒

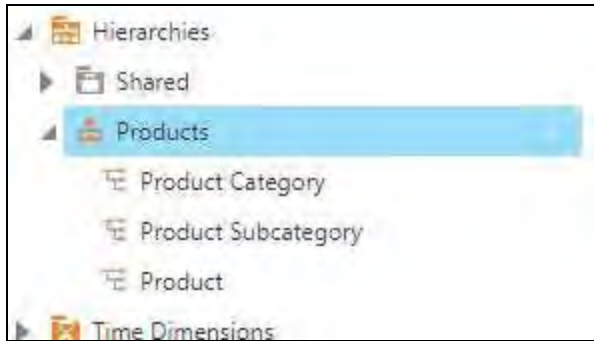
Value Type
Single Member

Select a hierarchy for this parameter

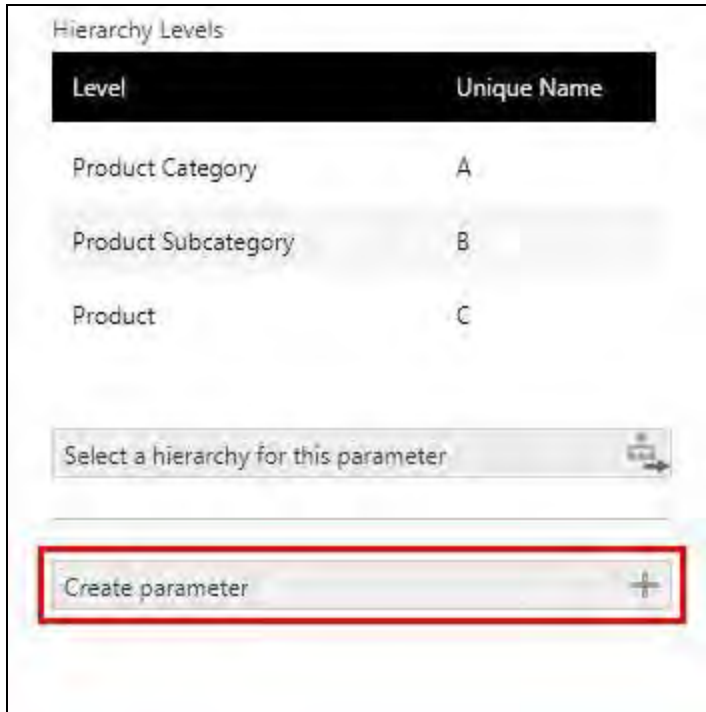
Create parameter

Enter a Name to identify this parameter on dashboards/views, and select **Public** to make it accessible to them. Set Value Type to **Single Member** for this example.

Click **Select a hierarchy for this parameter**, and in the Open dialog, locate and select your **hierarchy**. In this example, this is the *Product* hierarchy from the *Hierarchies* folder.



Returning to the Define Bridge Parameter dialog, click the **Create parameter** button. Then, you can select the stored procedure's **@ProductID** parameter to link to it.



Note: Notice the unique name *C* indicated for the *Product* level as this will be used in the unique names of the members we want to convert.

This opens a dialog where we can enter and test the script used to convert a hierarchy member into our required numeric value.



The following script calls the [Resolve](#) method on the [SingleMemberValue](#) *\$input\$* to ensure token values are resolved into a member, then extracts the numeric value from the member's unique name and returns it:

```
// Get SingleMemberValue with tokens resolved
SingleMemberValue parameterValue = $input$.Resolve();

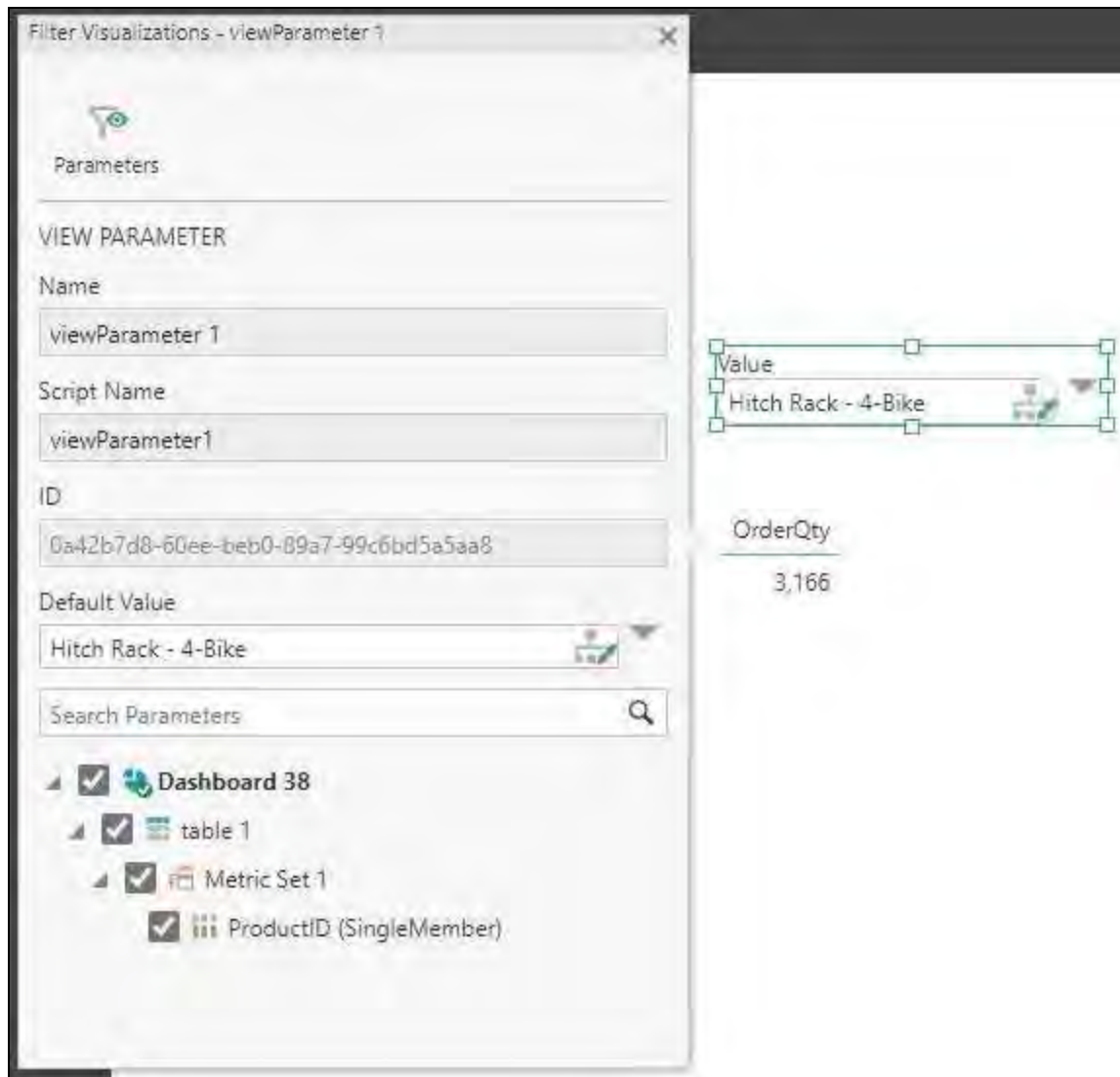
// Convert unique name into an integer ID
var productId = Convert.ToInt32(parameterValue.Value.UniqueName.Replace(".C", ""));

// Return as a SingleNumberValue
return new SingleNumberValue($id$, productId);
```

To test the script, select a **Test Value** from the hierarchy from its lowest level for this example (e.g., *Accessories> Bike Racks> Hitch Rack- 4-Bike*), then click **Test script**.

Now, we can connect a filter to our public bridge parameter. Create or edit a dashboard, and in the Explore window, expand the **Data Cubes** folder and drag its **OrderQty** measure to the dashboard canvas.

Choose **Filter** in the toolbar, and add a hierarchy **Value** filter. Connect it to our bridge parameter with type *SingleMember*, then you can filter to one of the products at the lowest level of the hierarchy and see the result.



The filter has options you can set in the Properties window to allow selecting only members from the bottom level.

Multi-Level Hierarchy With Single Selection On Multiple Levels

In this example, the bridge parameter is used to convert a single member from either of two levels of a hierarchy value filter into a string value used by a stored procedure.

This example is based on a table named `[dbo].[Sample_data]` created in a database with the following sample data:

	Continent	Country	Location	Sales
1	North America	USA	California	1200
2	North America	USA	Texas	2000
3	North America	Canada	Ontario	3000
4	North America	Canada	Quebec	7000
5	Asia	China	Beijing	5000
6	Asia	China	Shanghai	8000
7	Asia	Japan	Tokyo	3500
8	Asia	Japan	Narita	5000
9	Africa	Kenya	Nairobi	2600
10	Africa	Kenya	Mombasa	2800

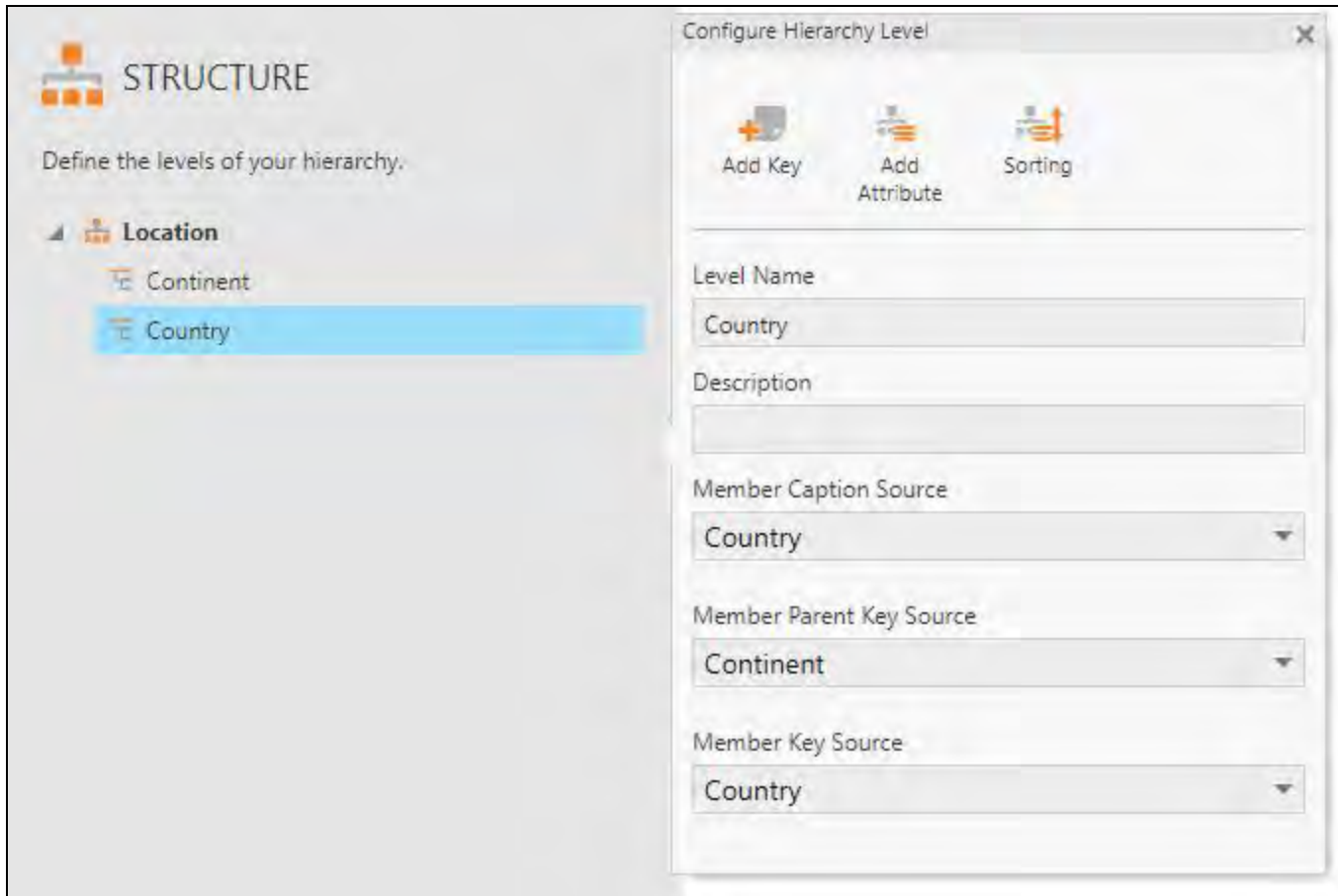
A stored procedure with two parameters is created in this example:

- *@Level* - The level of the selected member in the hierarchy filter.
- *@Name* - The selected member from either the *Continent* or the *Country* level.

```
CREATE PROCEDURE [dbo].[sp_Country_Sales]
    @Level nvarchar(2) = 'A',
    @Name nvarchar(1000) = 'North America'
AS
BEGIN
    SET NOCOUNT ON;
    IF (@Level = 'A')
        Begin
            SELECT [Location] , [Sales]
            FROM [dbo].[Sample_data]
            WHERE [Continent] = @Name
        End
    IF (@Level = 'B')
        Begin
            SELECT [Location] , [Sales]
            FROM [dbo].[Sample_data]
            WHERE [Country] = @Name
        End
End
```

END

Create a [hierarchy](#) from the main menu, and add two levels based on the *Sample_data* table: *Continent* and *Country*.



Create a new data cube, and drag the *sp_Country_Sales* stored procedure to the canvas. If you added the stored procedure after creating your data connector, right-click it and choose **Re-Discover**.

Click **Bridge Parameters** in the toolbar, then click **Add parameter** to open the Define Bridge Parameter dialog.



Enter a Name to identify this parameter on dashboards/Views, and select **Public** to make it accessible to them. Set Value Type to **Single Member** for this example.

Click **Select a hierarchy for this parameter**, and in the Open dialog, locate and select your **hierarchy**.

Returning to the Define Bridge Parameter dialog, click **Create parameter** and select the stored procedure's *@Level* parameter.



Note: Notice the unique names indicated for the two levels, as these are referred to in the stored procedure and may need to be updated.

This opens a dialog where we can enter and test script to convert the parameter values. The following script passes the level unique name from the input **SingleMemberValue** directly to our stored procedure:

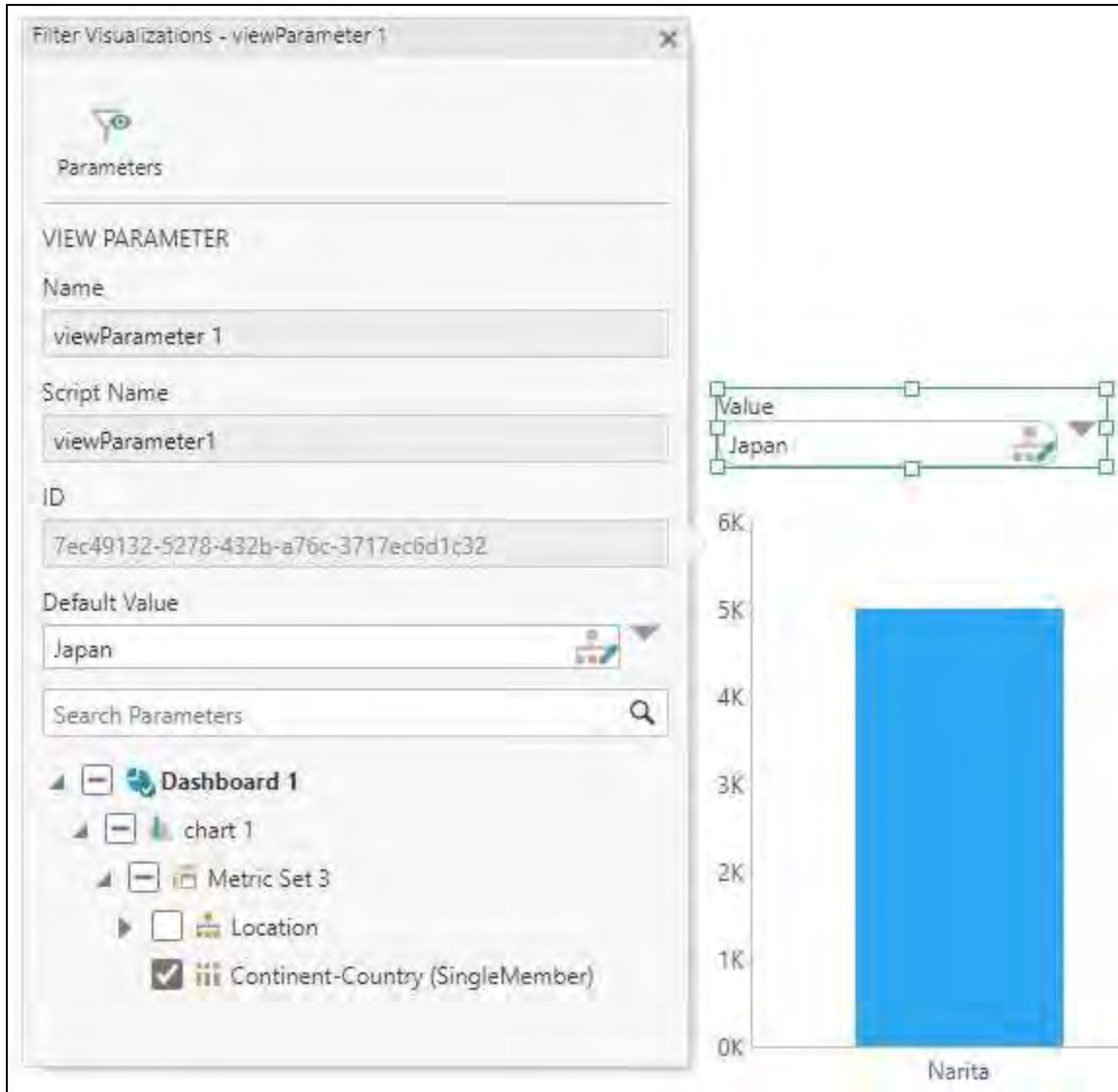
```
SingleMemberValue parameterValue = $input$.Resolve();
return new SingleStringValue($id$, parameterValue.Value.LevelUniqueName);
```

Select the stored procedure's **@Name** parameter. The following script removes the level unique name portion of the member unique name, leaving only the original key values:

```
SingleMemberValue parameterValue = $input$.Resolve();
var replacing = "." + parameterValue.Value.LevelUniqueName;
var keyValue = parameterValue.Value.UniqueName.Replace(replacing, "");
return new SingleStringValue($id$, keyValue);
```

Create or edit a dashboard, and drag the **Sales** measure and the **Location** dimension from the data cube to the dashboard canvas. Note that only North America results are shown by default as was indicated in the stored procedure.

Add a hierarchy value filter and connect it to the bridge parameter. First un-check **Location** if it was selected by default.



The filter is now connected to the hierarchy selected in the bridge parameter, and you can select either continent or country values.

Single-Level Hierarchy With Multiple Selection

In this example, a bridge parameter is used to select multiple hierarchy members from a filter connected to a single-level hierarchy.



- Archive of documentation for Logi Symphony v24

Create a new [hierarchy](#) from the main menu, and drag the **[Person]. [CountryRegion]** table to the canvas while holding the **Shift** key to add only a single level. Change the level's Member Key Source to **Name**, and rename the hierarchy and its level to **Country**.

Create a new Manual Query data cube. Select the **Adventure Works data connector** and use the following manual query:

```
SELECT B.Name as Country, SUM(A.SalesYTD) as SalesYTD
FROM [Sales].[SalesTerritory] A
INNER JOIN [Person].[CountryRegion] B
    ON A.CountryRegionCode = B.CountryRegionCode
WHERE $Country$
GROUP BY B.Name
```

Set up the *\$Country\$* placeholder with Identifier as **Country**, Value Type set to **Collection** and Data Type set to **String**.

Set Element Name to **B.Name**. This will replace the *\$Country\$* placeholder in the query with *B.Name IN <values>*. Add at least one default Value, such as *Australia*.

Configure Placeholder

Set up the settings of the placeholder being defined.

Identifier

Country

Name

Country

Public

☐

Value Type

Collection

Data Type

String

Element Name

B.Name

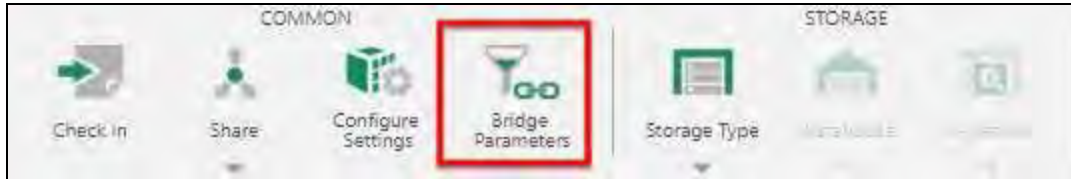
Value

Australia



For more information on setting up a placeholder for a manual select, see the article [Using manual select placeholders](#).

Click **Bridge Parameters** in the data cube's toolbar, then **Add parameter**.



Enter a Name to identify this parameter on dashboards/views, and select **Public** to make it accessible to them. Set Value Type to **Collection Member** for this example.

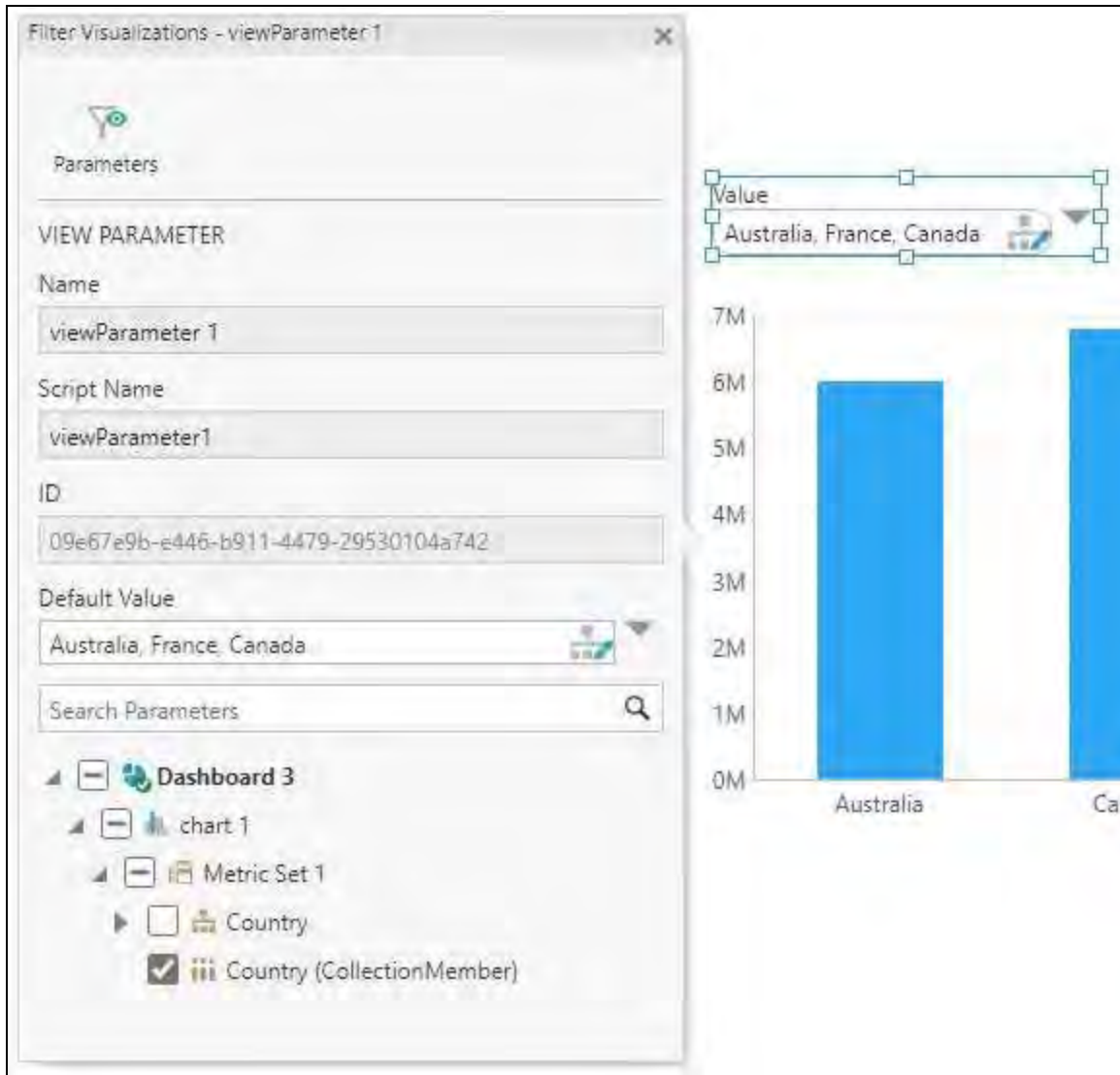
Click **Select a hierarchy for this parameter**, and locate and select your **hierarchy** in the Open dialog that appears.

Returning to the Define Bridge Parameter dialog, click **Create parameter** and select the **Country** parameter to link to it.

The following script extracts the member key values from the input [CollectionMemberValue](#) based on this hierarchy level's unique name and returns them as strings:

```
CollectionMemberValue parameterValue = $input$.Resolve();
var keyValues = new List<string>();
for (int i = 0; i < parameterValue.Values.Count; i++)
{
    keyValues.Add(parameterValue.Values[i].UniqueName.Replace(".A", ""));
}
return new CollectionStringValue($id$, keyValues);
```

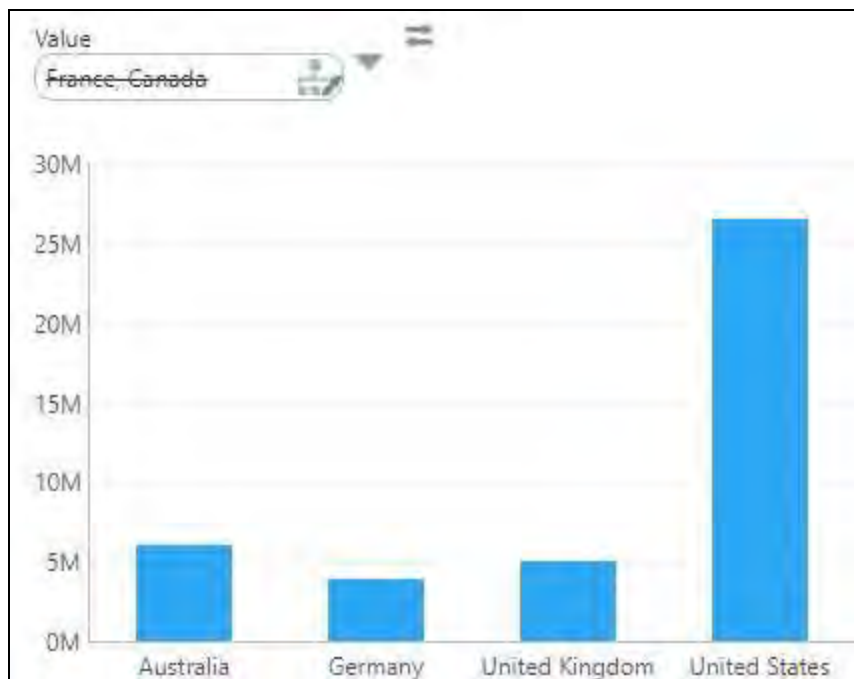
On a dashboard, drag **SalesYTD** and **Country** from your new data cube to the canvas and add a hierarchy value filter. Make sure the filter is connected to the *CollectionMember* bridge parameter.



Single-Level Hierarchy With Inverted Multiple Selection

To handle inverted selection, you need to check the parameter value property [IsInverted](#). You can pass it to your query as an additional parameter, for example.

This example modifies the previous one with the following manual query:



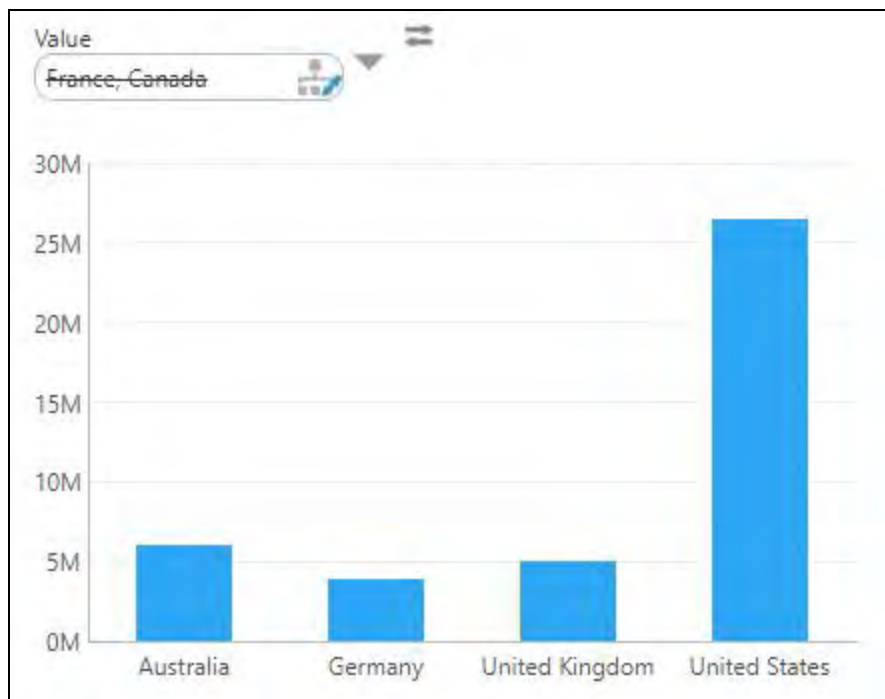
```
SELECT B.Name as Country, SUM(A.SalesYTD) as SalesYTD
FROM [Sales].[SalesTerritory] A
INNER JOIN [Person].[CountryRegion] B
    ON A.CountryRegionCode = B.CountryRegionCode
WHERE ($IsInverted$ = 0 AND $Country$)
    OR ($IsInverted$ = 1 AND NOT $Country$)
GROUP BY B.Name
```

This includes an additional placeholder *IsInverted* that should be set as **Single Boolean**.

Link the bridge parameter to this additional placeholder parameter and use script that checks whether the parameter value is inverted:

```
var parameterValue = $input$.Resolve();
return new SingleBooleanValue($id$, parameterValue.IsInverted);
```

On a dashboard, you can now choose the **Invert Selection** option from the filter's token menu.



Single-Level Hierarchy With Single Selection Using MDX

This example uses a bridge parameter to convert a single member from a single-level hierarchy to a value used in an MDX query.

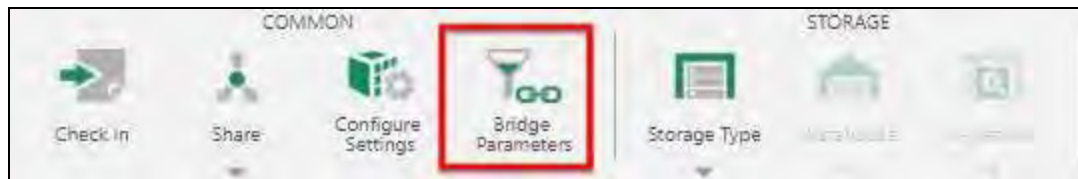
Use a single-level Country hierarchy like the one from the previous examples.

Create a new Manual Query data cube. This example uses a data connector connected to the SQL Server Analysis Services sample database Adventure Works with the following MDX query:

```
SELECT {[Measures].[Internet Sales Amount]} DIMENSION PROPERTIES MEMBER_TYPE ON COLUMNS,
{ DESCENDANTS(StrToMember($Country$),[Customer].[Customer Geography].[State-Province]) }
  DIMENSION PROPERTIES MEMBER_TYPE ON ROWS
FROM [Adventure Works]
```

Set up a *\$Country\$* placeholder as type **Single String**, and set a default value, such as *[Customer].[Customer Geography].[Country].&[Canada]*. See [Passing Placeholder Value From Dashboard to Manual MDX Select](#) for more details on placeholders.

Click **Bridge parameter** in the data cube's toolbar, then **Add parameter**.



Enter a Name to identify this parameter on dashboards/views, and select **Public** to make it accessible to them. Set Value Type to **Single Member** for this example.

Click **Select a hierarchy for this parameter**, and in the Open dialog, locate and select the **Country** hierarchy for this example.

Returning to the Define Bridge Parameter dialog, click **Create parameter** and select the **placeholder parameter** to link to it.

The following script forms the OLAP cube's unique name from the key extracted from Dundas Bi's hierarchy member unique name, accessed from the input [SingleMemberValue](#):

```
SingleMemberValue parameterValue = $input$.Resolve();
var keyValue = parameterValue.Value.UniqueName.Replace(".A", "");
var uniqueName = string.Format("[Customer].[Customer Geography].[Country].&[{0}]", keyValue);
return new SingleStringValue($id$, uniqueName);
```

Date Range Selection

In this example, a bridge parameter converts a range of dates into two date parameters. The following stored procedure is used:

```
CREATE PROCEDURE [dbo].[sp_SalesOrderDetail_Date]
    @DateFrom datetime,
    @DateTo datetime
AS
BEGIN
    SET NOCOUNT ON;
    SELECT
        [ModifiedDate] AS ModifiedDate,
        [LineTotal] AS LineTotal
    FROM [Sales].[SalesOrderDetail]
    WHERE [ModifiedDate] >= @DateFrom AND [ModifiedDate] < @DateTo
END
```



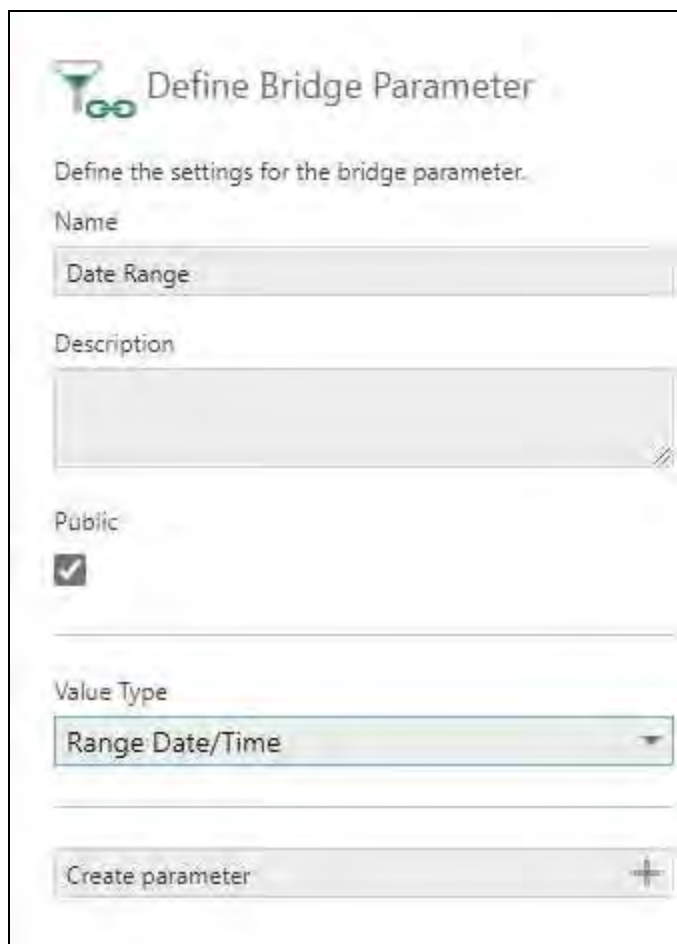
Note: This example uses an *exclusive* range end value.

Create a new data cube and drag the **sp_SalesOrderDetail_Date** stored procedure to the canvas.

Click **Bridge Parameters** in the toolbar, then click **Add parameter** to open the Define Bridge Parameter dialog.



Enter a Name to identify this parameter on dashboards/Views, and select **Public** to make it accessible to them.



The 'Define Bridge Parameter' dialog box is shown. It has a title bar with the Logi Analytics logo and the text 'Define Bridge Parameter'. Below the title bar, there is a subtitle 'Define the settings for the bridge parameter.' The form contains the following fields and controls:

- Name:** A text input field containing 'Date Range'.
- Description:** A large text area for describing the parameter.
- Public:** A checkbox that is checked.
- Value Type:** A dropdown menu currently set to 'Range Date/Time'.
- Create parameter:** A button with a plus sign icon to confirm the creation of the parameter.

Value Type can be set to **Range Date/Time** or **Range Member**, depending on which filter types you want to use. If you choose Range Member, click **Select a hierarchy for this parameter** and select a **hierarchy** under a time dimension to use in your filters.



Click **Create parameter** and select the stored procedure's **@DateFrom parameter** to link to it. This script returns the date value selected as the beginning of a Date/Time range from the input **RangeDateTimeValue**:

```
return new SingleDateTimeValue($id$, $input$.LowerBoundaryValue);
```

For the *Range Member* data type, the lower boundary value is a **member value**. The following script **resolves** the parameter value into members in case a token was selected, then returns the inclusive start date from the input **RangeMemberValue**:

```
var parameterValue = $input$.Resolve();
var datetime = parameterValue.LowerBoundaryValue.MemberTime;
if (!parameterValue.IncludeLowerBoundary) {
    datetime = parameterValue.LowerBoundaryValue.UpperBoundaryTime;
}
return new SingleDateTimeValue($id$, datetime);
```

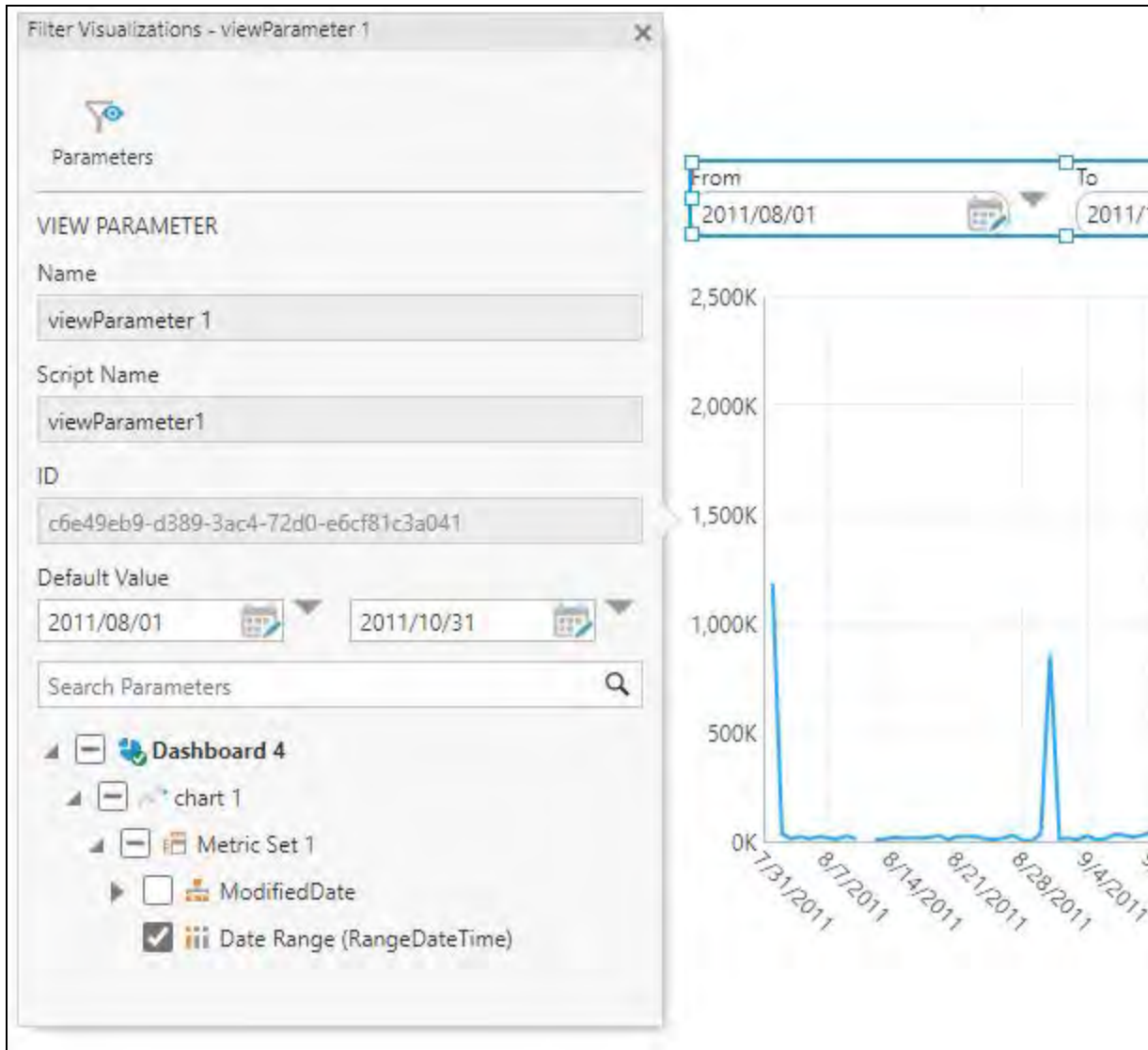
Select the **@DateTo parameter** to link to it and use script to return the date selected as the end of the range:

```
return new SingleDateTimeValue($id$, $input$.UpperBoundaryValue);
```

For *Range Member*, the following script would return the range end member value's exclusive upper boundary date:

```
var parameterValue = $input$.Resolve();
var datetime = parameterValue.UpperBoundaryValue.UpperBoundaryTime;
if (!parameterValue.IncludeUpperBoundary) {
    datetime = parameterValue.UpperBoundaryValue.MemberTime;
}
return new SingleDateTimeValue($id$, datetime);
```

Create or edit a dashboard, and drag the **LineTotal** measure and the **ModifiedDate** hierarchy to the dashboard canvas from your data cube in the Explore window. Add a filter, such as **Range Date** or **Calendar Range**, and connect it to your bridge parameter. De-select other parameters, such as your time hierarchy.

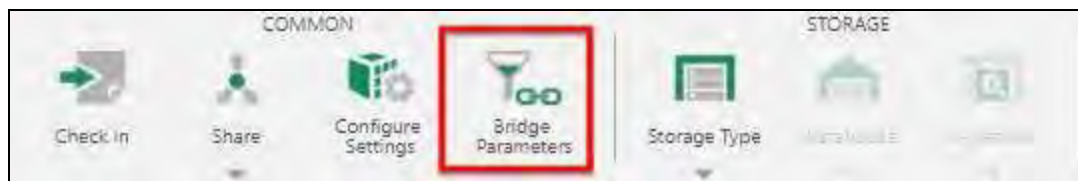


Number Range Selection

In this example, a range of numbers is passed to two parameters in a manual query:

```
SELECT OrderQty, OrderDate
FROM Sales.SalesOrderDetail INNER JOIN Sales.SalesOrderHeader
ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
WHERE ProductID >= $Start$ AND ProductID <= $End$
```

Click **Bridge Parameters** in the data cube's toolbar, then **Add parameter**.



Enter a Name to identify this parameter on dashboards/views, and select **Public** to make it accessible to them. Set Value Type to **Range Number**.

Click **Create parameter** and select the **start value parameter** to link to it. The following script returns the number selected as the beginning of the range from the input [RangeNumberValue](#):

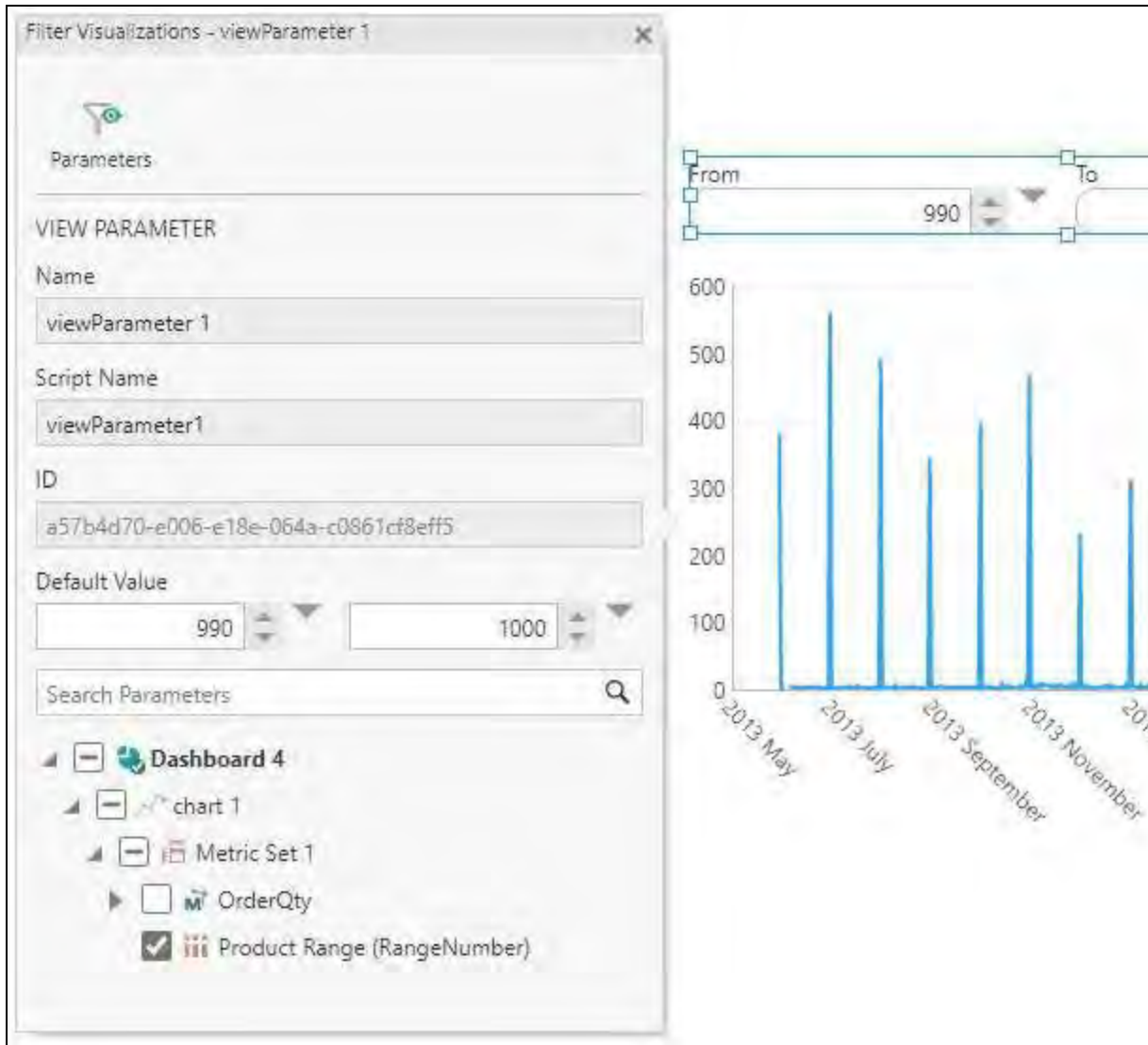
```
RangeNumberValue rangeNumberValue = $input$.Resolve();
return new SingleNumberValue($id$, rangeNumberValue.LowerBoundaryValue);
```

Select the **end value parameter** and use script to return the number selected as the end of the range:

```
RangeNumberValue rangeNumberValue = $input$.Resolve();
return new SingleNumberValue($id$, rangeNumberValue.UpperBoundaryValue);
```

Create or edit a dashboard, and drag the **OrderQty** measure and the **OrderDate** hierarchy to the dashboard canvas from your data cube in the Explore window.

Add a filter, such as **Range Number**, and connect it to your bridge parameter (de-select other parameters, such as the OrderQty measure).



Handling All and Other Basic Tokens

All types of [view parameters](#) and filters support choosing the built-in *All token values* by default. Range value types, such as *Range Date/Time* and *Range Number*, can also be set to the token value *Open Boundary Range* when only the *From* value or only the *To* value is defined.



You can [customize filters](#) to disable these tokens if not needed; otherwise, the bridge parameter's script should handle them. Calling the *Resolve* method like in the examples above does not work for these special token types because there is no specific value of type string, number, date, etc., that can correspond in all cases to *All* or *Open Range Value*. Bridge parameter script does not execute when the *Data Default* token is selected.

For example, you can insert script like the following before your other script, in case the [All token](#) is selected:

```
if ($input$.Token == ParameterToken.All) {  
    return new SingleStringValue($id$, $input$.Token);  
}
```

This assumes the transform parameter can accept the All token value. For some parameters, such as for a [manual select placeholder](#) or [stored procedure](#), you may need to instead return a special value compatible with your specific query or stored procedure. Otherwise, *SingleStringValue* should be replaced if necessary by whatever return [type](#) is expected for this transform parameter, which is listed above the script editor.

If you set the bridge parameter **Value Type** to a range so that *\$input\$* is a range value type, you can add script like the following to handle the [QRen Range Boundary token](#):

```
if ($input$.LowerBoundaryToken == ParameterToken.OpenRangeBoundary) {  
    return new SingleNumberValue($id$, 0);  
}  
// (and/or)  
if ($input$.UpperBoundaryToken == ParameterToken.OpenRangeBoundary) {  
    return new SingleNumberValue($id$, 1000000);  
}
```

Types

In addition to the usual types available to use in [DundasScript](#), the following can be used in bridge parameter script, many of which are used in examples above:

- [MemberValue](#)
- [SingleBooleanValue](#)
- [SingleDateTimeValue](#)
- [SingleMemberValue](#)
- [SingleNumberValue](#)



- [SingleStringValue](#)
- [CollectionBooleanValue](#)
- [CollectionDateTimeValue](#)
- [CollectionMemberValue](#)
- [CollectionNumberValue](#)
- [CollectionStringValue](#)
- [RangeDateTimeValue](#)
- [RangeMemberValue](#)
- [RangeNumberValue](#)
- [RangeStringValue](#)
- [LogicalValue](#)
- [ConnectorElementValue](#)
- [AnalysisElementValue](#)
- [FileValue](#)
- [HierarchyLevelValue](#)
- [TimeHierarchyOffsetValue](#)
- [Number](#)

- [PredefinedToken](#)

- [ParameterToken](#)



Note: When you create a bridge parameter and link to a transform parameter, the Default Value in the dialog that appears takes precedence for that parameter over the default value set when configuring its transform.

For more information, see:

- Video: [Filters](#)
- [Writing Data Scripts with DundasScript](#)
- [Using TRACE Statements](#)
- [Adding Filters](#)
- [Using Manual Select Placeholders](#)
- .NET API: [ParameterValue Class](#)
- [Stored Procedure Select](#)
- [Filter](#)



Using Manual Select Placeholders

This applies to: *Managed Dashboards, Managed Reports*

You can define a placeholder for a [Manual Select](#) transform and use it in your manual queries. A placeholder marks where in the query to insert the value from a parameter, which you can later connect to a filter or script on a dashboard, or set to an attribute value.

Using a Placeholder With SQL

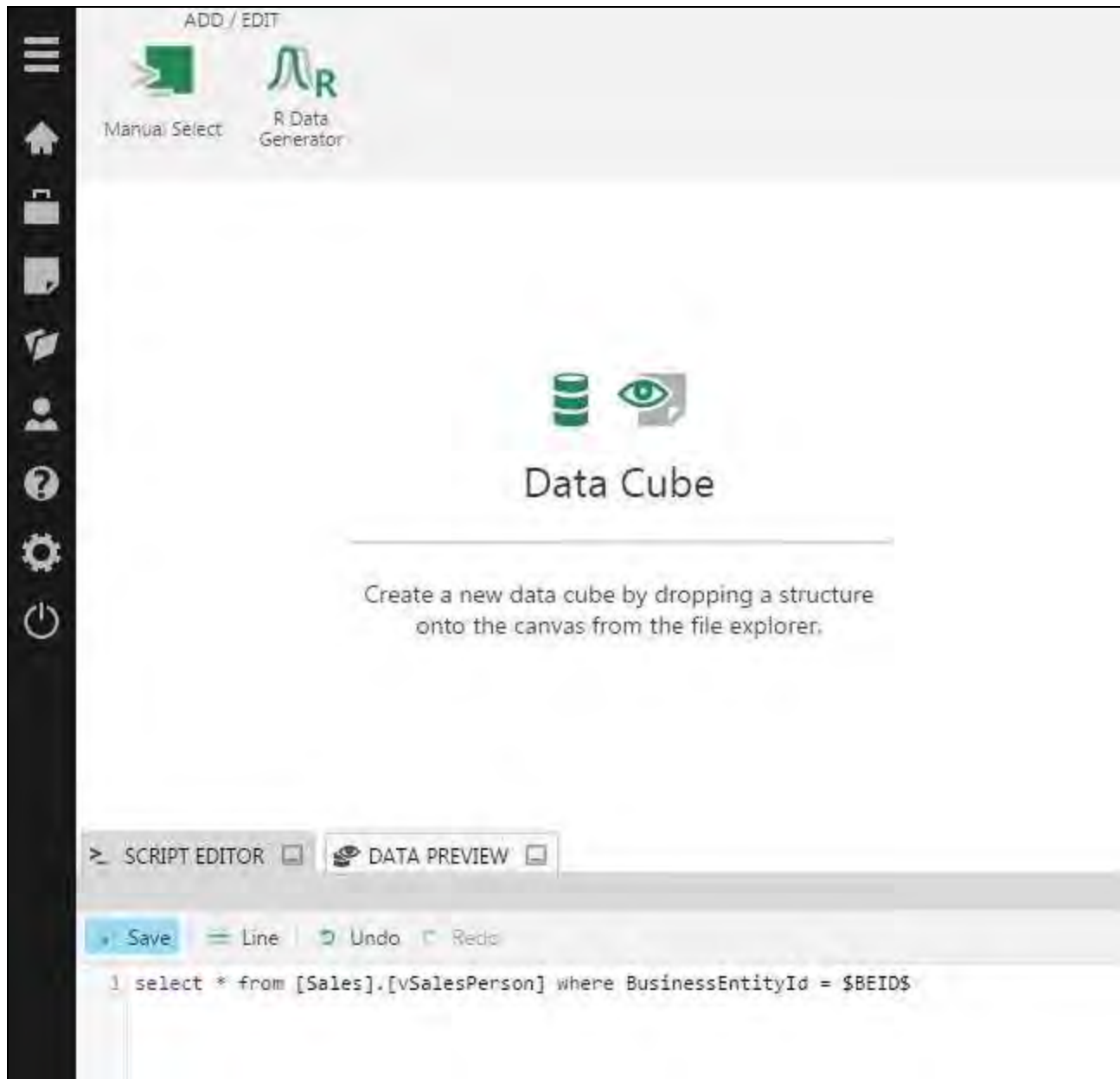
Defining a Placeholder

For this example, create a new data cube and select the **Manual Select** option.

In the Open dialog, select a **SQL Server data connector** (e.g., Adventure Works) and click the **submit** button at the bottom.

In the **Script Editor** window, enter a parameterized SQL statement where each parameter is enclosed between dollar sign(\$) characters. Then click **Save**. For example:

```
select * from [Sales].[vSalesPerson] where BusinessEntityId = $BEID$
```



The Placeholders dialog will appear. Click **Add placeholder** and then change the settings as needed. In our example, we set **Identifier** to `BEID` to match our query.

Select the **Public** check box to allow this parameter to be set outside the cube. Enter a default value for the parameter (e.g., 274).



Configure Placeholder

Set up the settings of the placeholder being defined.

Identifier

Name

Public

☒

Value Type

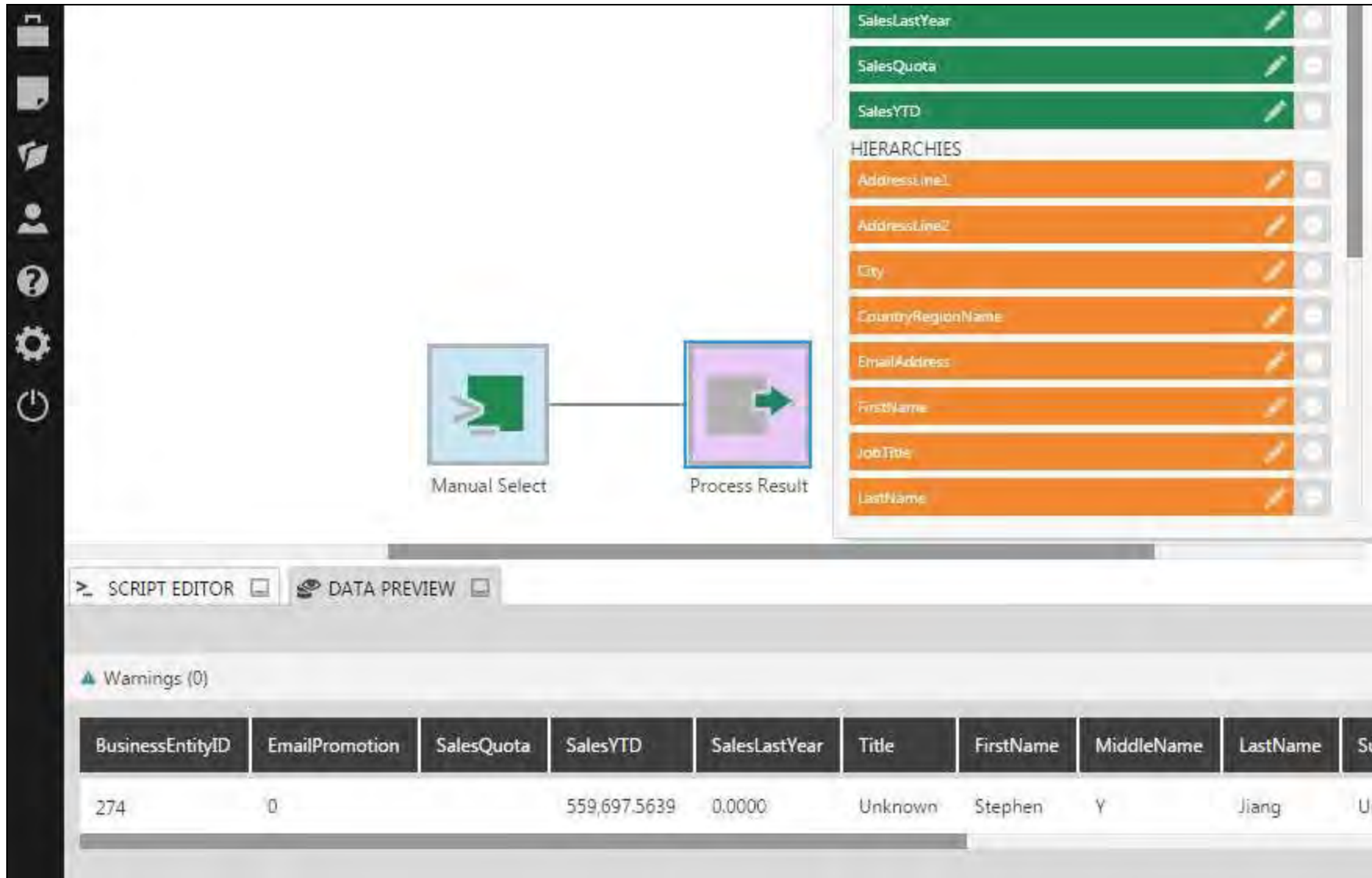
Data Type

Numeric Value



Important: When Value Type is *Collection*, the placeholder is replaced by the query text `<element> IN <values>`, so set the additional setting Element Name to your column name (e.g., *BusinessEntityID*). The sample query above would instead be: `select* from /Sales[/vSalesPerson] where $BEID$`

Click the **submit** button at the bottom and close the dialogs. A *Manual Select* and *Process Result* transform appear on the canvas. Select the **Process Result** transform and open the **Data Preview** window. You'll see that the records returned by the manual query have been filtered (e.g., `BusinessEntityID= 274`).



The screenshot displays the Logi Symphony v24 interface. On the left, a vertical toolbar contains icons for various functions. The main workspace shows a data pipeline with two transforms: 'Manual Select' (a blue square with a green arrow) and 'Process Result' (a purple square with a green arrow), connected by a line. To the right, a panel lists variables and hierarchies. The 'Sales' hierarchy includes 'SalesLastYear', 'SalesQuota', and 'SalesYTD'. The 'HIERARCHIES' section lists 'AddressLine1', 'AddressLine2', 'City', 'CountryRegionName', 'EmailAddress', 'FirstName', 'JobTitle', and 'LastName'. Below the workspace, there are tabs for 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table with columns: BusinessEntityID, EmailPromotion, SalesQuota, SalesYTD, SalesLastYear, Title, FirstName, MiddleName, LastName, and Su. The first row of data shows values: 274, 0, 559,697.5639, 0.0000, Unknown, Stephen, Y, Jiang, and Un.

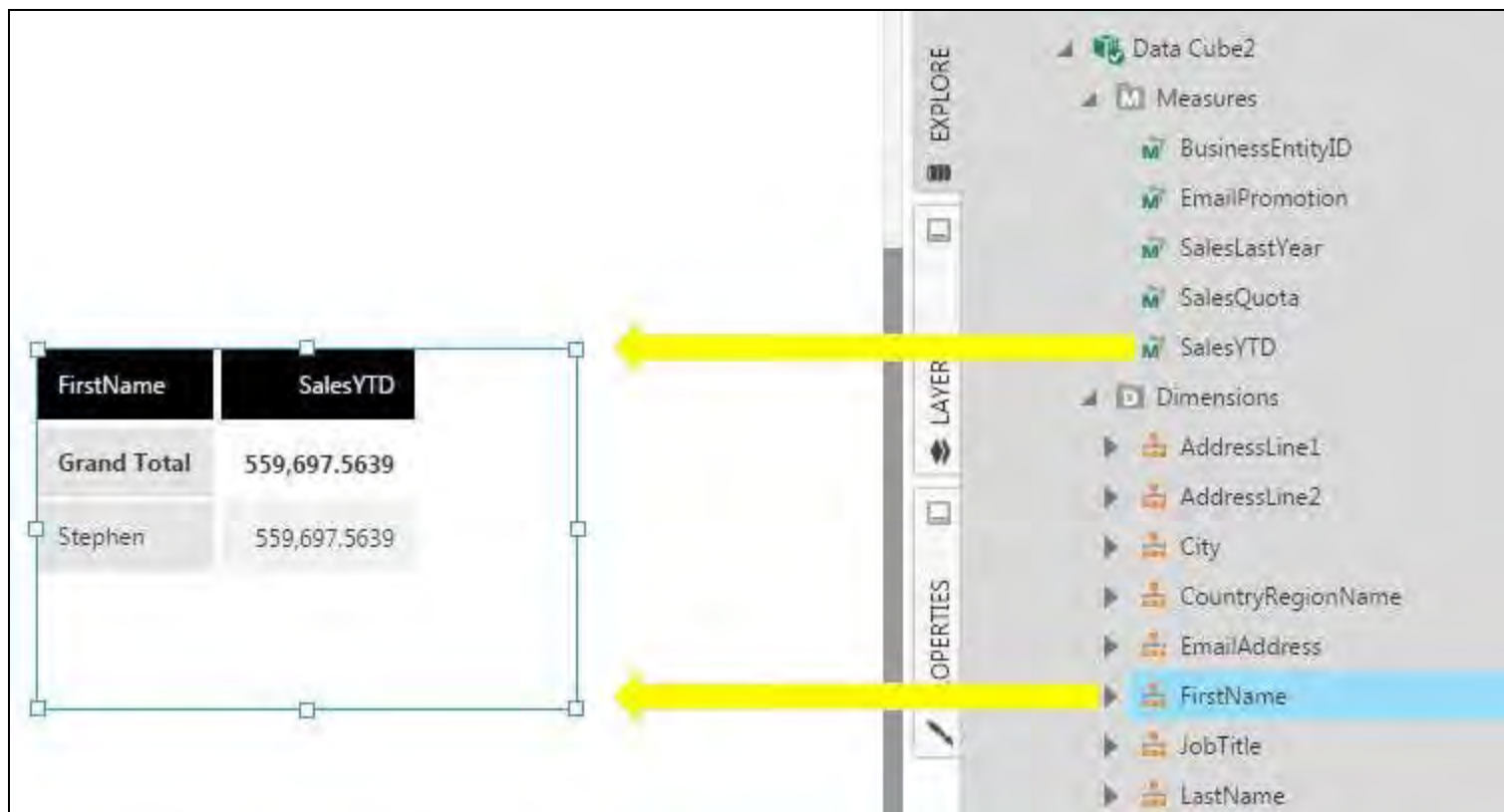


Note: Tokens, such as [built-in attributes](#) and [custom attributes](#), can be selected as the placeholder parameter's default value only after it's been created by reopening the Manual Select transform and editing its placeholders.

Using the Placeholder on a Dashboard

Next, create a new dashboard to see how to connect this placeholder parameter to a filter control.

Go to the Explore window and expand the data cube that was just created. Drag the **SalesYTD** measure to the canvas. It will appear as a table visualization. Then drag the **FirstName** dimension onto the table.

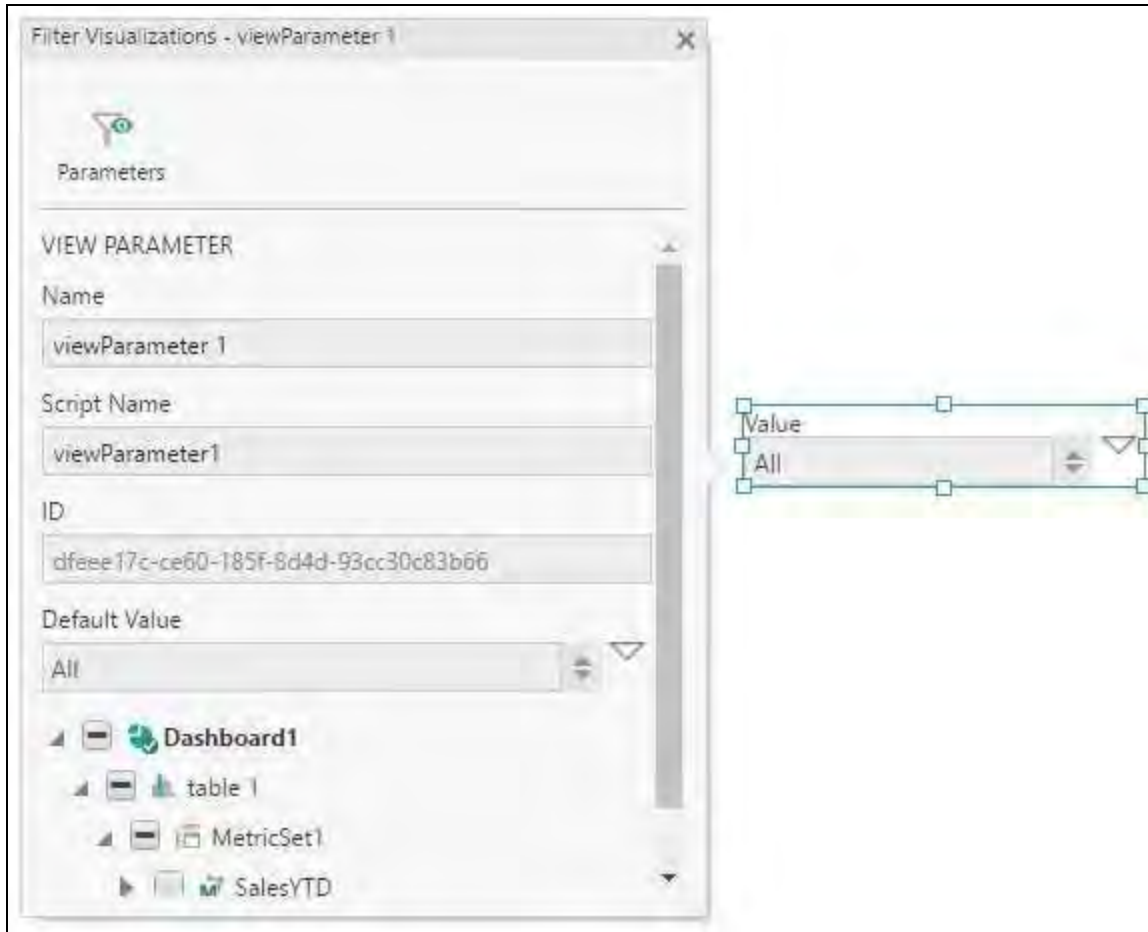


The screenshot shows the Logi Symphony interface. On the left, a table visualization is displayed with the following data:

FirstName	SalesYTD
Grand Total	559,697.5639
Stephen	559,697.5639

On the right, the 'EXPLORE' panel shows the 'Data Cube2' structure. The 'Measures' list includes 'SalesYTD', and the 'Dimensions' list includes 'FirstName'. Two yellow arrows indicate the drag-and-drop actions: one from 'SalesYTD' to the table header and another from 'FirstName' to the table header.

With the table selected, click **Filter** in the toolbar, and then select **Single Number**. In the Filter Visualizations panel, select the **Business Entity ID** item. This is the placeholder that was created earlier.



Finally, switch to **View** mode to use the filter.

FirstName	SalesYTD	Value
Grand Total	3,763,178.1787	275
Michael	3,763,178.1787	

Using a Stored Procedure With a Placeholder

In order to execute a stored procedure in the script editor, you need to uncheck the **Enable Subquery Optimization** check box in the *Manual Select* transform configuration.



Data Connector
DataConnector1

Timeout
30

The Query Statement
1 EXEC [dbo].[sp_Product] \$prodID\$

Edit Query

Define Placeholders

Enable Subquery Optimization ☐

See [Manual Select](#) for details on this option. Alternatively, you can use a [Stored Procedure Select](#) instead of executing the stored procedure with your own query in a Manual Select.

Using MDX With a Placeholder

See [Passing Placeholder Value From Dashboard to Manual MDX Select](#)

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Manual Select](#)
- [Data Cube Data Storage Options](#)
- [Create a Metric Set and Add a Filter](#)



Using TRACE Statements

This applies to: *Managed Dashboards, Managed Reports*

This article shows you how to use #TRACE statements in server-side [DundasScript](#), such as the ones used in calculated element transforms. The output from #TRACE statements goes to the [application logs](#) which can help you to debug your scripts.

DundasScript is used in several areas of the application, including:

- Data connectors (Generic Web, OLAP)
- Tokens
- Formulas
- Transforms (e.g., Calculated Element)
- Bridge Parameters

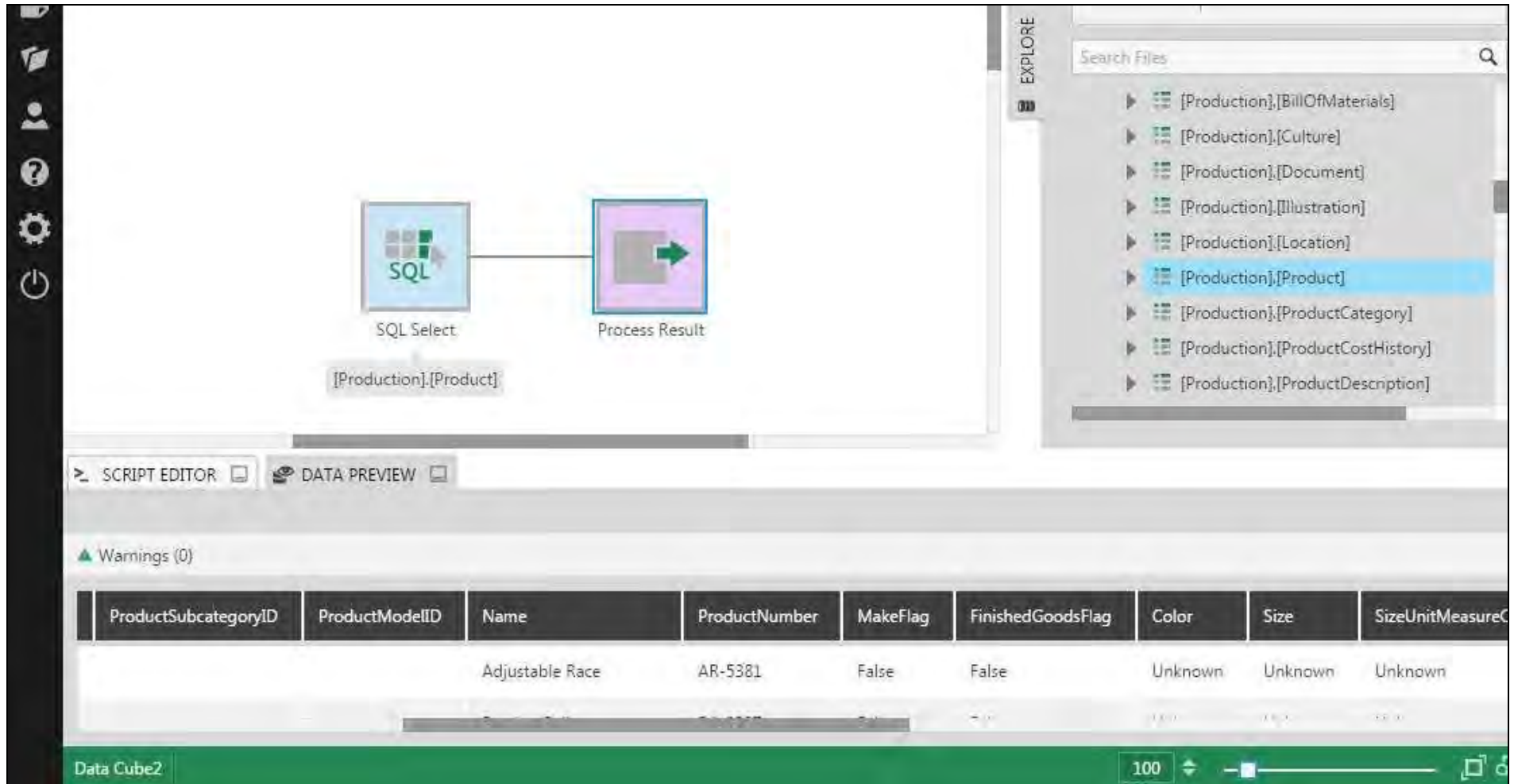
TRACE Statements

You can insert a #TRACE statement as a line of script anywhere DundasScript is used:

```
#TRACE <value to output>;  
  
// Optionally pass any number of values separated by commas:  
#TRACE <value 1>, <value 2>, <value 3>;
```

Using #TRACE With a Calculated Element Transform

For this example, create a new data cube and drag the *[Production].[Product]* table from a data connector in Explore to the canvas.



The screenshot shows the Logi Symphony v24 interface. At the top, there's a toolbar with icons for file operations, user management, help, settings, and a power button. The main workspace displays a data pipeline with two transforms: 'SQL Select' (blue square with a green arrow) and 'Process Result' (purple square with a green arrow). A connection link is visible between them. Below the transforms, the data source is identified as '[Production].[Product]'. On the right side, there's an 'EXPLORE' panel with a search bar and a list of files. The file '[Production].[Product]' is selected and highlighted in blue. At the bottom, there's a 'SCRIPT EDITOR' and 'DATA PREVIEW' tab. The 'DATA PREVIEW' tab is active, showing a table with the following columns: ProductSubcategoryID, ProductModelID, Name, ProductNumber, MakeFlag, FinishedGoodsFlag, Color, Size, and SizeUnitMeasureC. The first row of data shows: Adjustable Race, AR-5381, False, False, Unknown, Unknown, Unknown. At the bottom left, there's a 'Data Cube2' label. At the bottom right, there's a green bar with a '100' value and a slider control.

Select the **connection link** between the two transforms, go to the toolbar, click **Insert Common**, and then click **Calculated Element**.

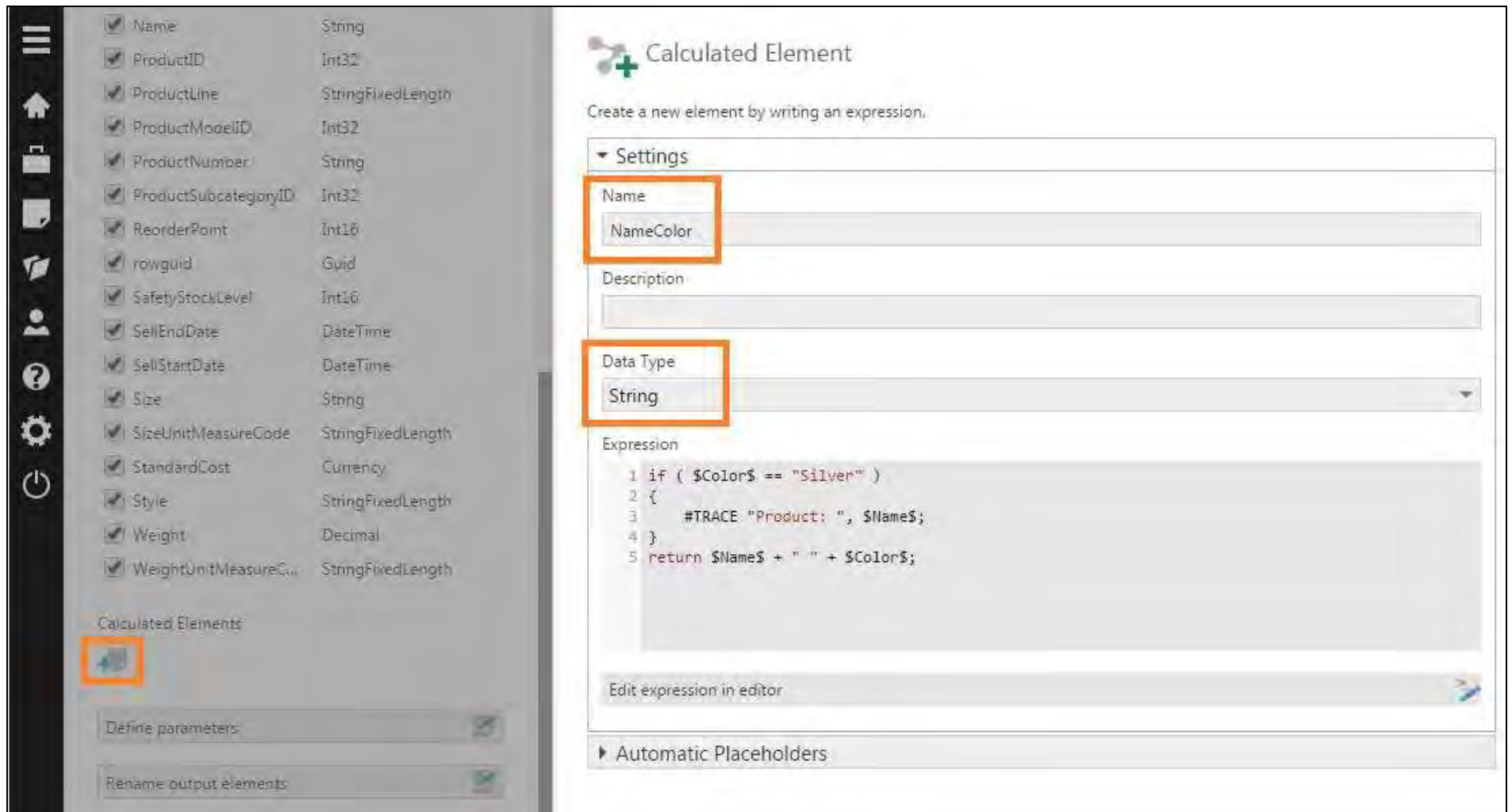
Double-click the **Calculated Element** transform to configure it. In the configuration dialog, scroll down and click the **plus** button under Calculated Elements to add a new calculated element (i.e., a new output column which will be defined using a script).

Enter the Name of the calculated element, set its Data Type to **String**, and then type in the following Expression (script).

```
if ( $Color$ == "Silver" )
{
```

```
#TRACE "Product: ", $Name$;
}
return $Name$ + " " + $Color$;
```

This script creates a new output column by concatenating two existing string columns. It also outputs a #TRACE statement to the application logs in administration for each product that has a Silver color.



Calculated Element

Create a new element by writing an expression.

Settings

Name:

Description:

Data Type:

Expression:

```
1 if ( $Color$ == "Silver" )
2 {
3     #TRACE "Product: ", $Name$;
4 }
5 return $Name$ + " " + $Color$;
```

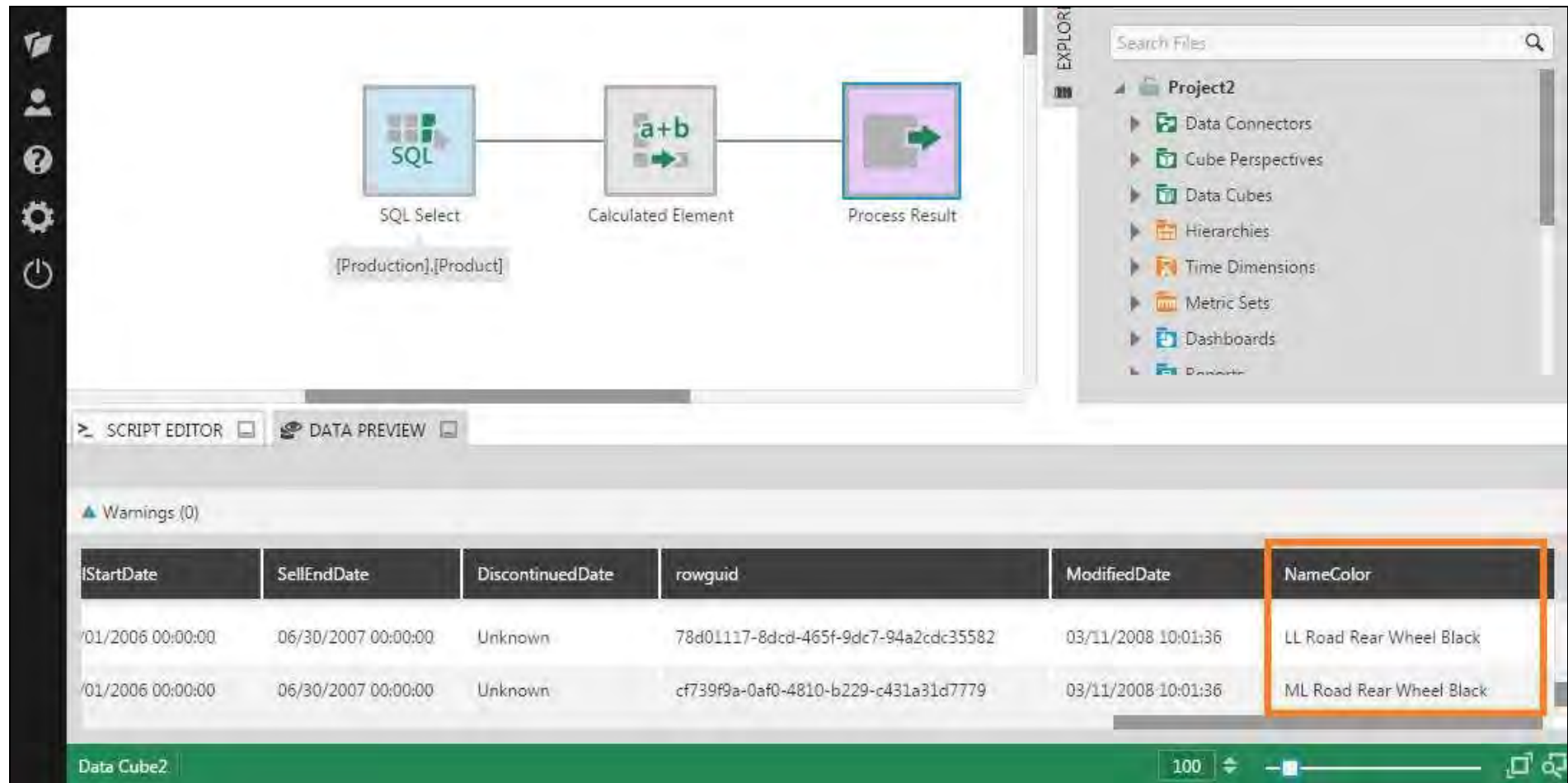
Edit expression in editor

Automatic Placeholders



Note: Calculated element transform scripts run for each row of data that is previewed or processed, which can produce many log entries if not restricted by your script.

Select the **Process Result** transform on the canvas and then click **Data Preview** to see the output from the data cube.



The screenshot shows the Logi Symphony interface. At the top, a canvas displays a data pipeline with three transforms: 'SQL Select' (labeled with '[Production].[Product]'), 'Calculated Element' (labeled with 'a+b'), and 'Process Result'. Below the canvas, there are tabs for 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table with the following data:

StartDate	SellEndDate	DiscontinuedDate	rowguid	ModifiedDate	NameColor
/01/2006 00:00:00	06/30/2007 00:00:00	Unknown	78d01117-8dcd-465f-9dc7-94a2cdc35582	03/11/2008 10:01:36	LL Road Rear Wheel Black
/01/2006 00:00:00	06/30/2007 00:00:00	Unknown	cf739f9a-0af0-4810-b229-c431a31d7779	03/11/2008 10:01:36	ML Road Rear Wheel Black

The 'NameColor' column is highlighted with an orange box. At the bottom of the interface, there is a green bar with the text 'Data Cube2' and a zoom slider set to 100.

Viewing the Output

Finally, if you're an administrator, go to the [Logs screen](#) to see the output of the #TRACE statements that ran when the script executed. Double-click an **entry** or select it and click **Details** in the toolbar to view its complete message.



You'll see some information about where the script was executed, the line number of the #TRACE statement in the script (helpful if you have multiple such statements), and lastly the actual output of the #TRACE statement itself. For example:

```
DataCube: [Production].[Product] Data Cube
Location: /Projects Root/Adventure Works/DataCubesRootFolder
Transform Name:
Transform Type: Calculated Element
Connector Element: Derived Column Output
Element Name: NameColor
(ln:3) Product: Mountain-500 Silver, 52
```

For more information, see:

- [Application Logs](#)
- [Writing Data Scripts with DundasScript](#)
- [Data Cubes](#)
- [Adding Formulas](#)
- [Using a Bridge Parameter](#)
- [Calculated Element](#)

Writing Data Scripts with DundasScript

This applies to: *Managed Dashboards, Managed Reports*

Symphony uses its own scripting language called *DundasScript* for transformations and other customizations at the data level, including formulas, calculated elements, bridge parameters, some data connectors, and script tokens. This article provides an overview of the DundasScript engine, including some examples and limitations to be aware of.



Note: Other languages are used for writing script in other areas of Dundas BI, such as [JavaScript](#) for advanced customization of dashboards and other views, and SQL when writing a [manual query](#).

The DundasScript Engine

DundasScript can be thought of as a subset of the C# programming language, but its syntax will be familiar if you have used other languages such as Java, JavaScript, or C, and it supports dynamic types.

Syntax

The DundasScript engine syntax is composed of the following elements:

- Keywords:

```
break case catch continue default do else false finally for foreach if in is new null return  
switch throw true try var while
```

- Built-in data types:

```
bool byte char decimal double float int long object string
```

- Operators (with the standard order of operations):

```
() ++ -- ! * / % + - < > <= >= == != | && || ?? = *= /= += -= |=
```

- Indexers, to access items in a collection, list, or array.
- Exceptions, to handle errors when the script does something unsupported:

```
string[] parts;  
try  
{  
    parts = someVariable.Split();  
}  
catch( /* NullReferenceException */ )  
{  
    parts = new string[];  
}
```

- Generics, if a generic type is registered with the engine ([see available types below](#)):

```
List<int> myIntList = new List<int>();  
myIntList.Add(3);  
myIntList.Add(2);  
myIntList.Add(1);  
myIntList.Sort();
```

- Access to all of an object's public methods, properties, and fields.
- Single dimensional arrays (note that inline array initialization isn't supported):

```
int[] bunchOfValues = new int[4];  
bunchOfValues[0] = 1;  
bunchOfValues[1] = 1;  
bunchOfValues[2] = 2;  
bunchOfValues[3] = 3;
```

- [IEnumerable<>](#) extension methods on lists and other collections (without lambda expressions):

```
int last = myIntList.Last();  
int avg = myIntList.Average();
```

- Strings, which can be multi-line if you use the `@` [verbatim identifier](#) for a string literal:

```
string line = "Regular string";  
string multiline = @"Multi  
line  
string";
```

- Both types of comments:

```
// Describe your code.  
  
/*  
Comments are useful.  
*/
```

Numeric Values

Numeric values typed into your code without any decimal places are interpreted as integral numbers, or the `int` type. Include the decimal point, or use *float* or *double* values:

- 12 - integer
- 12f - floating-point
- 12d - double-precision floating-point

The `0x` prefix can also be used to specify an integer in hexadecimal digits, for example `0xC` for 12.

As in most programming languages, the division of two integers always results in an integer. For example, the resulting value of *r* in the following script is *2.0*:

```
double r = 12 / 5;
```

Use the following script instead to get a result of *2.4*:

```
double r = 12.0 / 5;  
// Or:  
double r = 12d / 5;
```

Features

The DundasScript engine implements a custom interpreter. Although close in syntax with C#, it is a dynamically-typed script language. This results in the following features:

- You do not need to know the type of an object or cast it to access one of its properties.
- You can use the var keyword in place of any type name to simplify script.
- You can call static methods on an instance variable.
- There is no variable scoping. This means that if you redefine a variable, it is not an error but the previous value will be overwritten/lost. For example:

```
// Create a new string:  
string foo = "Hello world";  
if (someCondition)  
{  
    // From now on, foo is an int, not a string!  
    int foo = 10;  
}
```

Limitations

The following elements are **not** supported in DundasScript:

- Multidimensional arrays.
- Indexers with multiple parameters.
- Delegates.
- Multiple variable declarations per line.

- Creating user-defined events.
- Nullable types.
- *ref* and *out* parameters.
- Lambda expressions.
- The following operators:

```
& ^ &= ^= >> << >>= <<= typeof sizeof ~x ++x --x
```

- The conditional operator (x? y: z).
- Array creation using initializer:

```
int[] a = new int[] { 1, 2, 3 };
```

Standard .NET Classes

Only registered types can be used, providing a *sandbox* that makes it difficult for script to make your project misbehave or crash.

The following classes, structs, and enums are exposed in the DundasScript engine. They access the .NET types directly and have not been re-implemented in the engine. All public methods, properties and fields will work as documented by Microsoft:

- [Array](#) - Provides methods for creating, manipulating, searching, and sorting arrays, thereby serving as the base class for all arrays in the common language runtime.
- [bool](#) - Boolean value (true or false)
- [Boolean](#) - Same as bool.
- [byte](#) - An 8-bit unsigned integer.



- [Byte](#) - Same as byte.
- [CalendarWeekRule](#) - Defines different rules for determining the first week of the year.
- [Capture](#) - A single successful subexpression capture.
- [char](#) - A single UTF-8 character.
- [Char](#) - Same as char.
- [Collection<>](#) - A generic collection.
- [Color](#) - Describes a color in terms of alpha, red, green, and blue channels.
- [Colors](#) - A set of predefined colors such as CornFlowerBlue.
- [Convert](#) - Converts a base data type to another base data type.
- [CultureInfo](#) - Information about a specific culture, including the names for the culture, the writing system, the calendar used, the sort order of strings, and formatting for dates and numbers.
- [DateTime](#) - A date and time value.
- [DateTimeKind](#) - Specifies whether a DateTime object represents a local, UTC or unknown time.
- [DateTimeOffset](#) - A point in time, typically expressed as a date and time of day, relative to UTC.
- [DayOfWeek](#) - Specifies the day of the week.
- [decimal](#) - A fixed-precision decimal value.
- [Decimal](#) - Same as decimal.
- [Dictionary<,>](#) - A collection of keys and values.



- [double](#) - A double precision floating-point value.
- [Double](#) - Same as double.
- [Enum](#) - The base class for enumerations.
- [Exception](#) - Errors that occur during application execution.
- [float](#) - A single precision floating-point value (faster but less accurate).
- [Group](#) - The results from a single capturing group.
- [Guid](#) - Represents a globally unique identifier.
- [IConvertible](#) - Defines methods that convert the value of the implementing reference or value type to a common language runtime type that has an equivalent value.
- [ICollection<>](#) - Defines size, enumerators, and synchronization methods for all generic collections.
- [IDictionary<>](#) - A generic collection of key/value pairs.
- [IEnumerable<>](#) - Many collection types implement this interface, which provides various helpful extension methods.
- [int](#) - A 32-bit signed integer.
- [Int16](#) - A 16-bit signed integer.
- [Int32](#) - A 32-bit signed integer.
- [Int64](#) - A 64-bit signed integer.
- [IList<>](#) - A generic collection of objects that can be individually accessed by index.
- [IReadOnlyCollection<>](#) - A strongly-typed, read-only collection of elements.



- [IReadOnlyDictionary<>](#) - A generic read-only collection of key/value pairs.
- [IReadOnlyList<>](#) - A read-only collection of elements that can be accessed by index.
- [KeyValuePair<,>](#) - A key/value pair that can be set or retrieved.
- [List<>](#) - A list of objects that can be accessed by index.
- [long](#) - A 64-bit integer value.
- [Match](#) - Represents the results from a single regular expression match.
- [Math](#) - Provides constants and static methods for common mathematical functions.
- [MidpointRounding](#) - Specifies the strategy that mathematical rounding methods should use to round a number.
- [object](#) - The base class of all .NET class hierarchies.
- [Object](#) - Same as object.
- [Point](#) - an x- and y-coordinate pair in two-dimensional space.
- [Random](#) - A pseudo-random number generator.
- [Regex](#) - An immutable regular expression.
- [RegexOptions](#) - Enumerated values to use to set regular expression options.
- [sbyte](#) - An 8-bit signed integer.
- [sByte](#) - Same as sbyte.
- [short](#) - A 16-bit signed integer.



- [Single](#) - A single-precision floating-point number.
- [string](#) - A character string.
- [String](#) - Same as string.
- [StringComparison](#) - The culture, case, and sort rules to be used by certain overloads of the String.Compare and String.Equals methods.
- [StringSplitOptions](#) - Whether applicable String.Split method overloads include or omit empty substrings from the return value.
- [Timespan](#) - Represents a time interval.
- [TimeZoneInfo](#) - The time zone in the world.
- [uint](#) - A 32-bit signed integer.
- [UInt16](#) - A 16-bit unsigned integer.
- [UInt32](#) - A 32-bit unsigned integer.
- [UInt64](#) - A 64-bit unsigned integer.
- [ulong](#) - A 64-bit unsigned integer.
- [Uri](#) - An object representation of a uniform resource identifier (URI) and easy access to the parts of the URI.
- [UriBuilder](#) - Provides a custom constructor for uniform resource identifiers (URIs) and modifies URIs for the Uri class.
- [UriComponents](#) - Specifies the parts of a Uri.
- [UriFormat](#) - Controls how URI information is escaped.
- [UriHostNameType](#) - Defines the types of host names that can be specified in a URI.



- [UriKind](#) - Defines the different kinds of URIs.
- [UriPartial](#) - Defines which parts of a URI to get in certain methods.
- [ushort](#) - A 16-bit unsigned integer.

You can create new objects using the *new* keyword for any classes that have been registered with the DundasScript engine.

Dundas Classes

If you need information about the current user's session at the time the script is executing, you can access the variable `session`.

- `session` (type: [Session](#)) - Represents the current logon session when the script runs.

Debugging

To see live values and help with fixing issues with script, you can use the statement `#TRACE` in your script to pass a value to be entered in Symphony's [application logs](#).

For more details, see [Using TRACE statements](#).

For more information, see:

- [Using TRACE Statements](#)
- [Using a Bridge Parameter](#)
- [Calculated Element](#)
- [Adding Formulas](#)
- [Using a Custom Total Calculation](#)

Using Transforms

This applies to: *Managed Dashboards, Managed Reports*

A data cube is a multidimensional model of your data that is built of transforms that define your ETL process in each cube.

When you create or edit a [data cube](#), you can add most transforms the toolbar depending on your application settings and current canvas selection. For example, to insert a Join transform, select the connection link between two already connected transforms, then select Join from the toolbar. Alternatively, select an existing transform and drag it to another transform to connect the two, or select a connection between transforms and right-click to add other transforms or joins.

- **Input Transforms:** Use input transforms to bring data into the cube directly, or by reference. See [Input Transforms](#) and [Output Transforms](#).
- **Common Transforms:** Use these common transforms, such as Join, Filter, and Aggregate, to bring your data together or change the data, and more. See [Common Transforms](#).
- **Other Transforms:** For more extensive data cleansing, you can bring remove duplicate records, apply sort conditions to your data, or flatten your data, and more. See [Other Transforms](#).
- **Output Transforms:** This brings together the final output or result of your data cube ETL process. Configure the measures and hierarchies that are available to downstream consumers through metric sets and more. See [Input Transforms](#) and [Output Transforms](#).

When editing a data cube, most transforms can be accessed from the toolbar depending on your current canvas selection. For example, to insert a Join transform, first select the connection link between two already connected transforms.

To configure a transform, select the transform, then select **Configure** from the right-click menu to open a dialog that offers you information about the transform you can reference and in some cases, edit, such as the transform **Name**, **Color**, **Description**, and more.



Note: You need to be a user with a [Developer seat](#) to create or edit a data cube.

Input Transforms

Use input transforms to bring data into the cube directly, or by reference.

Transform	Description	Inputs
Data Cube	This transform uses the output of another data cube as an input, created when you drag another data cube onto the canvas.	N/A
Data Input	Reference a warehouse data storage area containing user input data. You can use the Data Input transform to	N/A



Transform	Description	Inputs
	add and modify data from inside Symphony.	
Manual Select	Enter a SQL or MDX statement to make a selection from a data connector instead of dragging native structures onto the canvas.	N/A
MDX Select	Drag and drop to select a tabular data result from an OLAP cube as input to a Symphony data cube.	N/A
Python Data Generator	Generate data by writing scripts using the Python programming language.	N/A
R Data Generator	Generate data by writing scripts using the R statistical programming language.	N/A
SQL Select	This transform is created when a structure is dragged onto the canvas from a data connector that supports relational queries. The columns to be used in the data cube can then be selected or unselected and aggregators can be set up per element. When retrieving data, these settings will then be translated into a SQL statement and sent to the data source.	N/A
Stored Procedure Select	This transform lets you retrieve data using a relational stored procedure. It is created automatically when you drag a stored procedure from a data connector in Explore to the data cube canvas.	N/A
Tabular Select	Similar to the SQL Select, this transform is created when dragging structures onto the canvas from a data connector that supports tabular data (e.g. XML, CSV).	N/A

Common Transforms

Transform	Description	Inputs
Aggregate	Apply aggregate functions to the columns of the previous transform such as SUM, AVG, COUNT, MIN, and MAX. Also allows setting GROUP BY functionality for the columns that are not aggregated.	1
Calculated Element	Create a new column by supplying an expression. The expression can contain placeholders representing columns using \$columnName\$ notation, and use functions such as MEDIAN and RATECHANGE.	1
Data Conversion	Change the data type of a column to another data type. For example, data coming from a CSV file may be recognized as a String but it is known to be an Integer.	1
Filter	Filter out rows that do not meet the configured criteria/settings. For example, use this transform when all of the data being read from a data source is not required and the data source doesn't allow querying (e.g. XML).	1

Transform	Description	Inputs
	Another example usage would be to place a filter on an ID column with the setting Less Than and the value 1000 which would result in taking only records with an ID value of less than 1000. Available filter operators are: Equals, Not Equals, Greater Than, Greater Than Or Equals, Less Than, and Less Than Or Equals.	
Join	Join two tables by defining the keys and specifying the join type. If relationships exist between the two tables the link is automatically created but this can be changed if necessary. If the data connectors of the transforms being joined are the same and the corresponding data provider supports joining then an optimized query will be generated and sent to the server.	2
Lookup	Join data from input columns with columns in a lookup table.	2
Math	Perform simple math functions such as Absolute, Ceiling, and Square Root on numeric input columns.	1
String	Manipulate string columns in data tables by applying functions such as To Upper, To Lower, Substring, Trim, Left, and Right.	1
Union	Combine data from multiple input structures by mapping columns onto one another. If the structures come from the same data connector and that data provider supports union statements then an optimized query will be generated and sent to the server.	2+

Other Transforms

Transform	Description	Inputs
Copy Element	Create new columns by copying selected input columns and adding the new columns to the output. This can be used in cases where you need to manipulate a column but want to keep the original column as well.	1
Flatten JSON	The Flatten JSON transform turns rows of data containing JSON text into separate columns for each of its values.	1
Flatten XML	The Flatten XML transform turns rows of data containing XML text into separate columns for each of its values.	1
Fusing	The Fusing transform combines the rows and columns of two tables together by appending them.	2
Fuzzy Grouping	Allows grouping of records by looking at the similarity between the values of various columns. Two records in which a possible misspelling occurs can be grouped together for further analysis, or duplicates can be removed by setting Output Top Level Records Only. The sensitivity can be set adjusted by setting the Probability Threshold.	1



Transform	Description	Inputs
Fuzzy Lookup	Search for a matching record from a secondary table when no relationship key fields are defined between two tables.	2
Merge	Merge values from multiple columns into one. This might be useful after a full outer join where one column may be null.	2+
Null Replacement	Replace input data with user-specified values when null values are found. Use this as an alternative to a Calculated Element transform which requires you to write a script.	1
Percentage Sampling	By specifying a rate, this transform reads in all of the data from the previous transform and generates a set of random indexes according to the rate input multiplied by the total record count, and then outputs the records according to those indexes.	1
Pivot	Allows for creation of new columns and transposing data into a new layout. Can be used to make results more compact by pivoting the input data on a column value.	1
Python Analysis	Write scripts using the Python programming language to perform statistical and predictive analysis on data.	1+
R Language Analysis	Write scripts using the R programming language to perform statistical and predictive analysis on data.	1+
Rank	Allow a rank number to be assigned to a new element created by the transform.	1
Record Sampling	Reads in all of the data from the previous transform and generates a set of random indexes according to the number input. Then output the records according to those indexes.	1
Remove Duplicates	Remove duplicated records by grouping all of the selected input columns and copying the results to the output	1
Sort	Apply sorting conditions to numerous columns. The data will be read in from the previous transform and sorted according to the options. It will then output the data to the next transform in the data cube. If the data provider of the previous transform supports sorting then an optimized query will be generated and sent to the server.	1
Top/Bottom	Set up rules on how data should be sorted and how many records are to be returned (combination of Sort transform and Record Sampling transform without the random generation).	1
Transpose	Turn columns into rows and rows into columns.	1
Unpivot	Combine multiple columns into a single column (opposite of Pivot). It expands a result into a more normalized form by expanding values from multiple columns in a single record into multiple records with the same values in a single column.	1



Output Transforms

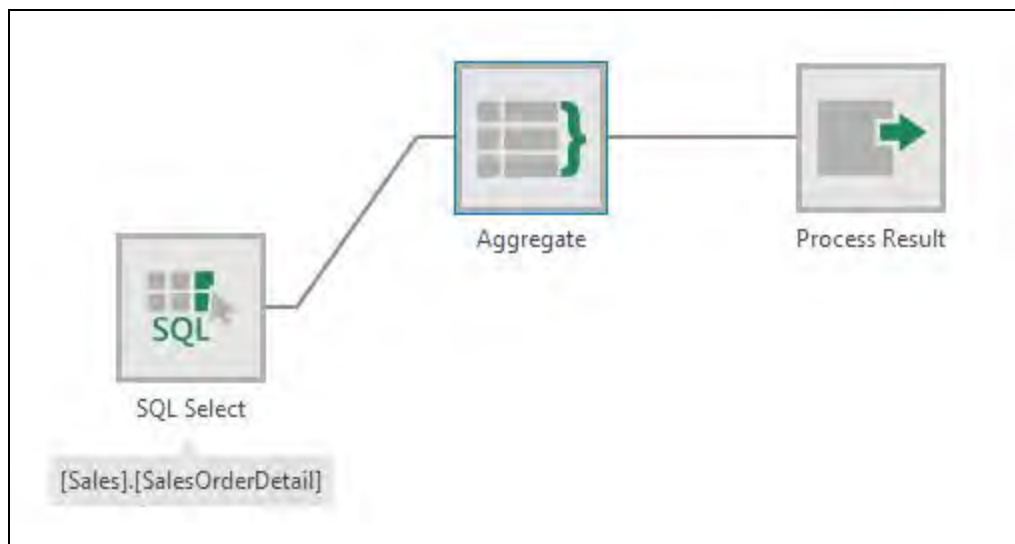
Output transforms bring together the final output or result of your data cube ETL process.

Transform	Description	Inputs
Process Result	Represents the final output or result of the data cube (ETL process). This transform doesn't do any data processing but it allows you to configure the measures and hierarchies that will be made available to downstream items such as metric sets.	1

Aggregate

This applies to: *Managed Dashboards, Managed Reports*

The Aggregate transform allows use of aggregate functions such as sum, average, count, minimum, maximum, etc.



Input

The Aggregate transform requires one input transform.

Configure

The transform will initially be configured to group by all non-numeric columns and sum the numeric columns.

Un-check the checkboxes for any columns you want to exclude from the output, and click to expand the remaining columns to configure them.

Aggregate

The transform allows aggregate functions to be applied to various input columns and copies the results to the output. It also allows setting Group By functionality for the columns that are not aggregated.

Name

Description

ID

SETTINGS

Choose the elements to use in this transform.



▸ ☐ SalesOrderID (Int32)

▾ ☒ OrderQty (Int16)

Aggregate Function

Group By

▸ ☒ ProductID (Int32)

▸ ☐ OrderDate (DateTime)

▸ ☐ SalesPersonID (Int32)

▸ ☐ FirstName (String(50))

▸ ☐ LastName (String(50))

For columns that are selected for output, either set **Aggregate Function**, or check **Group By**.

SETTINGS

Choose the elements to use in this transform.




☐ SalesOrderID (Int32)

☒ OrderQty (Int16)

☒ ProductID (Int32)

☐ OrderDate (DateTime)

☐ SalesPersonID (Int32)

☐ FirstName (String(50))

☐ LastName (String(50))

Aggregate Function

None

Group By

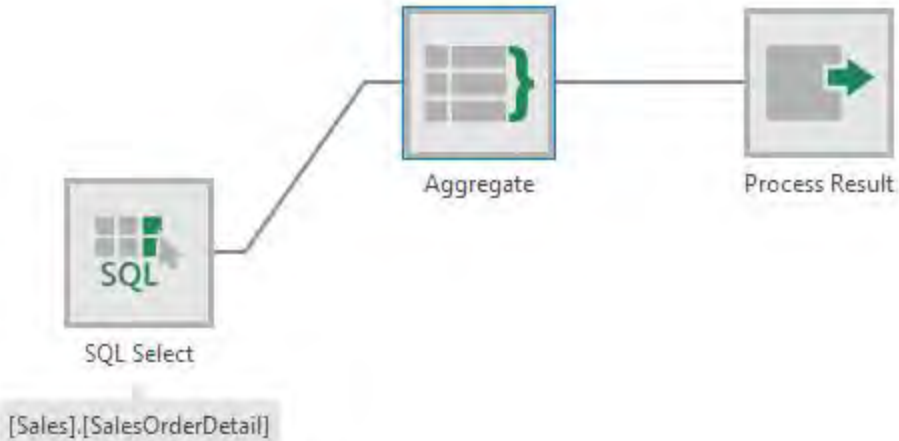
☒





Output

The figure below illustrates the output from the Aggregate transform as applied to the *SalesOrderDetails* table.



The workflow diagram illustrates a data processing pipeline. It begins with a 'SQL Select' component, which is connected to a data source labeled '[Sales].[SalesOrderDetail]'. The output of the SQL Select component flows into an 'Aggregate' component, represented by a grid icon with a green bracket. Finally, the output of the Aggregate component flows into a 'Process Result' component, represented by a grid icon with a green arrow.

Below the workflow diagram, the interface includes two tabs: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is currently active, displaying a table of data.

OrderQty	ProductID
625	925
36	902
90	710
249	879
90	733
1616	856
346	756
2394	779
190	802
322	971



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)



Calculated Element

This applies to: *Managed Dashboards, Managed Reports*

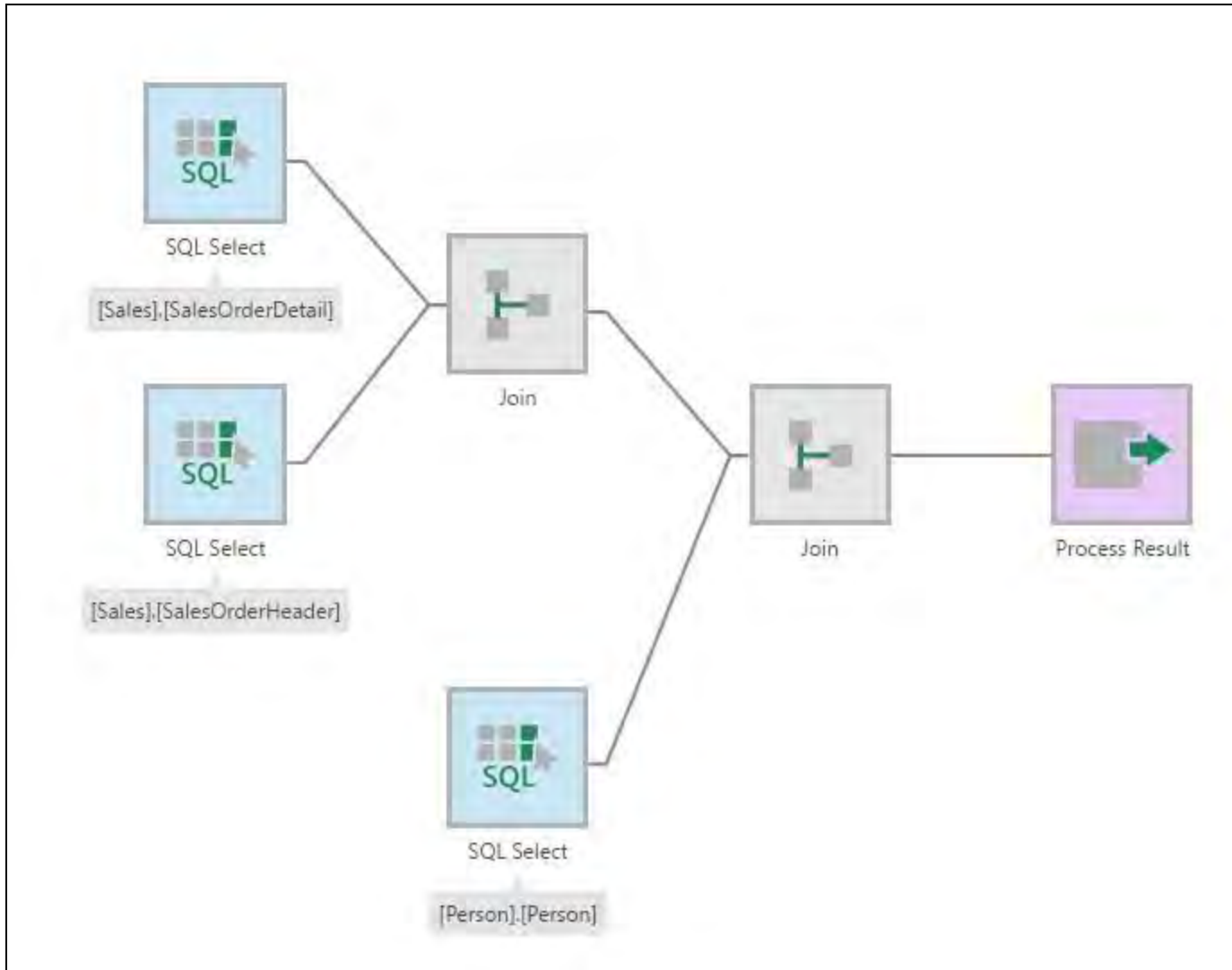
The **Calculated Element** transform lets you create new elements by writing [DundasScript expressions](#). In addition to using built-in operators and methods, you can pass data into a Symphony [formula](#) function. The result of each expression is a new output element.

Related video: [Data Cleansing with Data Cubes](#)

Input

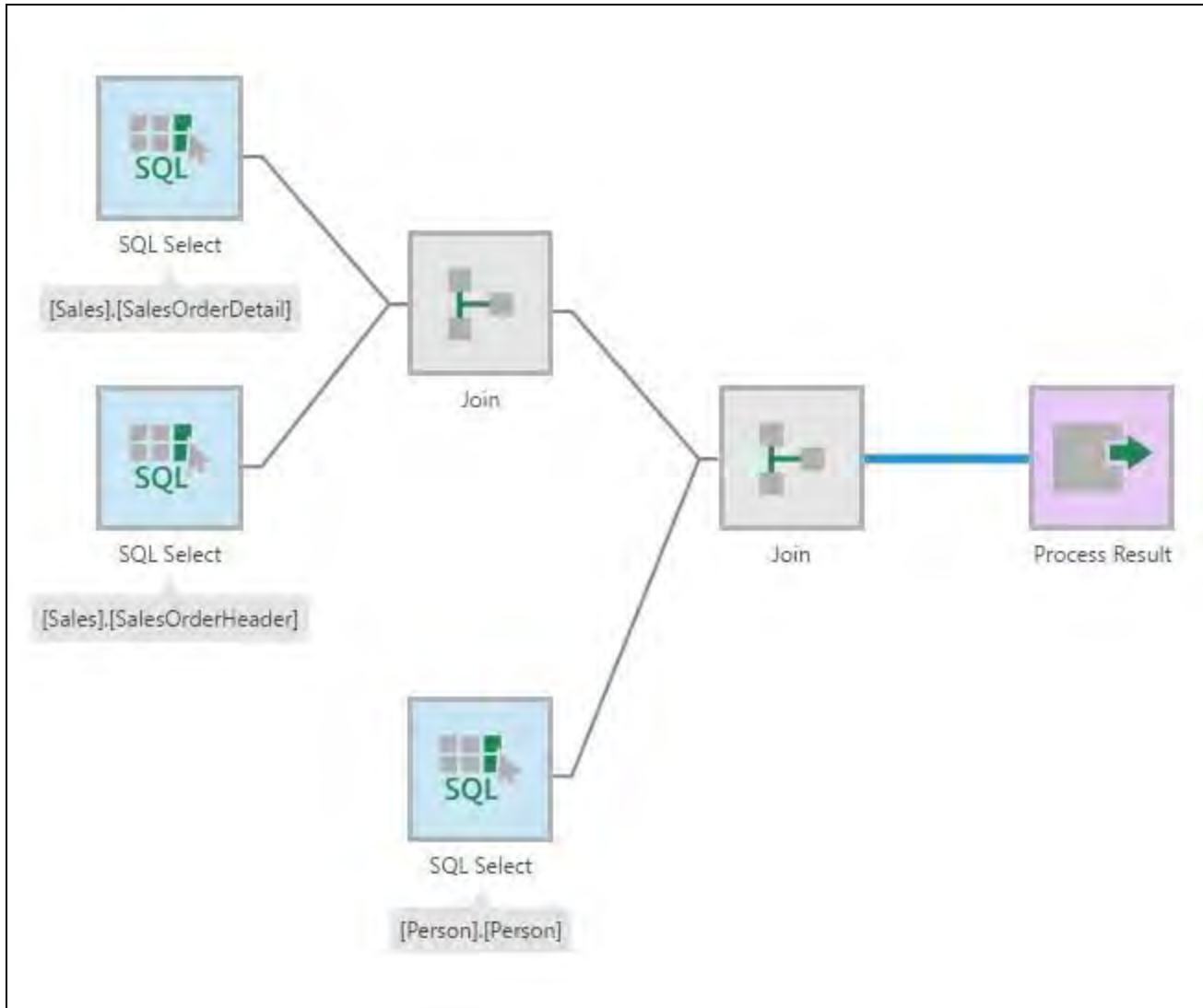
The **Calculated Element** transform requires input data from one transform.

For example, we will use the data cube created in the article [Create a New Data Cube](#).

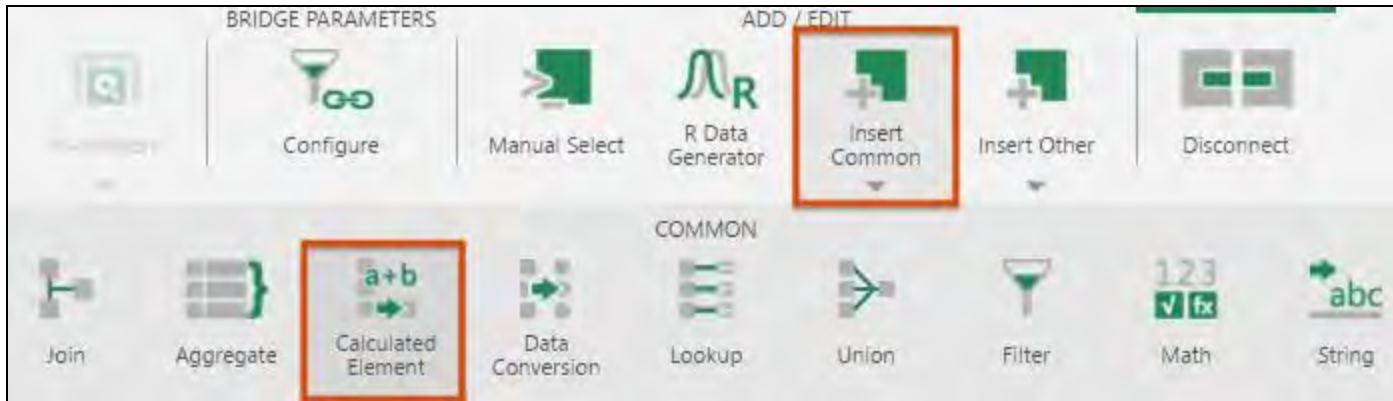


Add the Transform

To add the transform, select the connection between the second Join and the Process Result.



Click Insert Common in the toolbar and select Calculated Element.



Note: When you add certain transforms like a Calculated Element transform, an icon with an exclamation mark appears on the top-left of the transform indicating that data will be taken in-memory for processing and could impede performance in some cases.

Configure the Transform

- Double-click the Calculated Element transform to configure it.
- In the Calculated Element transform configuration dialog, you can optionally provide a Name and Description for the transform for your reference. The name will be displayed below this transform on the canvas.
- De-select any columns you want to omit from the output. Click Add Calculated Element.



Calculated Element

[more help](#)

The transform allows for creation of new elements by applying user-defined expressions. The result of each expression is a new output element.

Name

Description

ID

SETTINGS

Input Elements




☒	SalesOrderID	Int32
☒	OrderQty	Int16
☒	ProductID	Int32
☒	OrderDate	DateTime
☒	SalesPersonID	Int32
☐	FirstName	String(50)
☐	LastName	String(50)

Calculated Elements



Note: You can add multiple elements using a single transform.

In the *Calculated Element* dialog, provide a name and optional description for the new element, and set Data Type to the value your script will return. When you choose String as the data type, you can specify a value for Maximum Length. The default value is 200.

When using the Decimal data type, two additional optional fields become available. If set, your values cannot exceed these limits when using [cube storage](#):

- Precision - Maximum count of digits in each number.
- Scale - Maximum count of digits after the decimal separator.

Write your [DundasScript](#) in the expression field, which must include a return statement. For example:

```
1return $FirstName$ + " " + $LastName$;
```

Type the dollar sign ('\$') to begin entering a placeholder for another column's values, and choose from the popup menu that appears.



Calculated Element

Create a new element by writing an expression.

Settings

Name

Full Name

Description

First name, followed by a space and the last name

Data Type

String

Expression

```
1 return $FirstName$ + " " + $LastName$;
```

Edit expression in editor

Automatic Placeholders

Define Placeholders

Click the submit button at the bottom of the dialog when finished.

Output

Select the Process Result transform and expand the **Data Preview** window to see the new columns in the output.

SalesOrderID	OrderQty	ProductID	SalesPersonID	LineTotal	OrderDate	Full Name
43,659	1	776	279	2,024.994000	05/31/2011 00:00:00	Tsvi Reiter
43,659	3	777	279	6,074.982000	05/31/2011 00:00:00	Tsvi Reiter
43,659	1	778	279	2,024.994000	05/31/2011 00:00:00	Tsvi Reiter
43,659	1	771	279	2,039.994000	05/31/2011 00:00:00	Tsvi Reiter
43,659	1	772	279	2,039.994000	05/31/2011 00:00:00	Tsvi Reiter
43,659	2	773	279	4,079.988000	05/31/2011 00:00:00	Tsvi Reiter

Working with Placeholders

The transform automatically defines a placeholder for each input element. The above example makes use of two such placeholders (FirstName and LastName).

To view a list of automatically defined placeholders, expand the Automatic Placeholders section.



Calculated Element

Create a new element by writing an expression. To retrieve values from other elements, use an identifier from the list of automatic placeholders or define new placeholders. If you modify an automatic placeholder it will be moved into the normal list of placeholders. Also, if you do not use it in the expression, the changes will be lost.

Settings

Automatic Placeholders

Identifier	Offset Value	Edit
\$SalesOrderID\$	0	
\$OrderQty\$	0	
\$ProductID\$	0	
\$OrderDate\$	0	
\$SalesPersonID\$	0	
\$FirstName\$	0	
\$LastName\$	0	

Define placeholders 

Note the offset value next to each placeholder, which is set to a default of 0. The offset is a number of records to go back or forward when retrieving a value used in the script. You can introduce an offset to the input element used for a placeholder by clicking the Edit icon and changing the value.

For example, to use the value of the ProductID from the previous row, set the Offset Value for the \$ProductID\$ placeholder to -1.



Note: The order of some data may change depending on how it is queried based on its use in a metric set. To ensure the sort order is as you expect, you can insert a [sort](#) transform before the Calculated Element.

To use values from a single column/input element multiple times but with different offsets, click Define Placeholders and add additional placeholders. You can add more than one for the same element but with different identifiers and offsets to refer to in your script.

Expression Samples

The following are some common expressions for calculated elements:



Get the year part from a date column:

```
DateTime date = $ModifiedDate$;  
return date.Year;
```

Add 3 months to a date column:

```
DateTime date = $StartDate$;  
return date.AddMonths(3);
```

Check the Year and return a string:

```
DateTime date = $Date$;  
string text;  
if (date.Year == DateTime.Today.Year)  
{  
    text = "This Year";  
}  
else  
{  
    text = "Other Year";  
}  
return text;
```

Get the lowercase equivalent of a string:

```
string s = $FirstName$;  
return s.ToLower();
```

Apply title case capitalization to a string:

```
string s = $Title$;  
return CultureInfo.CurrentCulture.TextInfo.ToTitleCase(s);
```

Split a string up by commas into multiple values and return the first:



```
var values = $CommaSeparated$.Split(',');  
return values[0];
```

Change a UNIX timestamp to the DateTime type:

```
DateTime dateTime = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);  
dateTime = dateTime.AddSeconds($Timestamp$).ToLocalTime();  
return dateTime;
```

Use the median function from Dundas BI formulas (/support/learning/documentation/analyze-data/formulas/list-of-formulas):

```
return MEDIAN($Sales Amount$);
```

Limitations

Column names that contain \$ (the dollar symbol) are not allowed to be used with Calculated Elements. You can rename this column to refer to it in script.

For more information, see:

- Video: [Introduction to Data Cubes](#)
- Video: [Data Cleansing with Data Cubes](#)
- [List of Formulas](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)

Clustering predictions using R language analysis

This applies to: *Managed Dashboards, Managed Reports*

This article shows an example of using R for analysis, creating clusters using a K-means model. K-means clustering aims to partition a number (n) of observations into a number (k) of clusters, in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. Clustering can help identify different groups in your data that should receive special treatment based on their characteristics (for example, a defined custom marketing campaign for a certain cluster).



Note: This article is intended to show how you can use any R package or function for additional advanced analytics tasks. Symphony already includes a [K-Means Clustering](#) function you can use in metric set formulas, and can [recommend](#) the option to automatically identify clusters in a scatter plot.

Prepare the Data

The following example uses the [Iris data set](#). This data set contains sepal and petal measurements, which are used to predict the species of the iris plant by classifying the data points into different clusters.

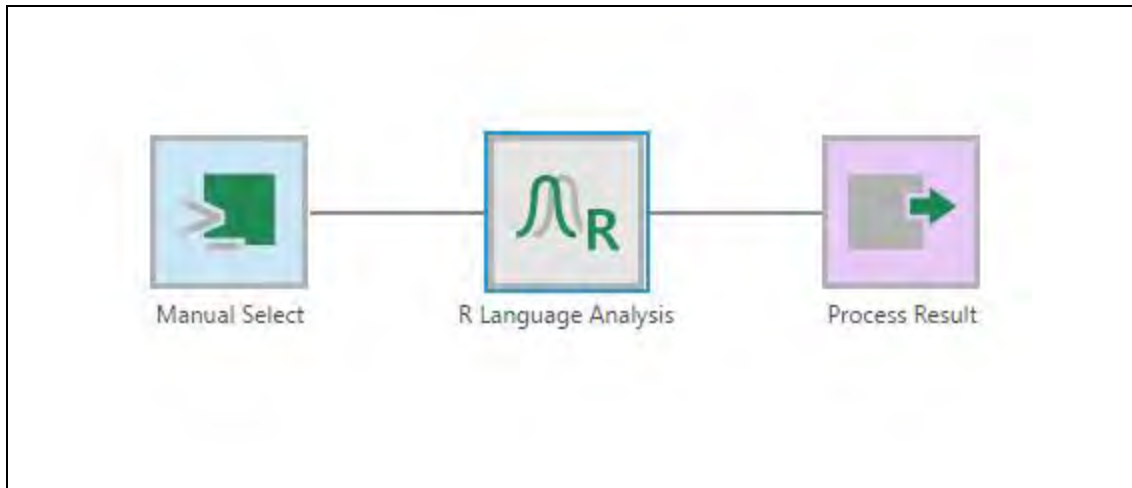
If necessary, first [install and configure R and Rserve](#).

Create the Data Cube Using Rserve

Create a data cube that retrieves your data from any source, such as a database, Excel file, or a [flat file](#) such as the iris data file provided in CSV format. The following shows the iris data in our example with column names assigned:

Sepal length	Sepal width	Petal length	Petal width	Sample ID	Species
5.1	3.5	1.4	0.2	1	I. setosa
4.9	3.0	1.4	0.2	2	I. setosa
4.7	3.2	1.3	0.2	3	I. setosa
4.6	3.1	1.5	0.2	4	I. setosa
5.0	3.6	1.4	0.2	5	I. setosa
5.4	3.9	1.7	0.4	6	I. setosa

Using the **Insert Other** option in the toolbar, add an **R Language Analysis** transform after the Select transform.



Configure the R Language Analysis transform and add the following script:

```
#Define an argument for each column in the data set.
arg1 <- $PetalLength$
arg2 <- $PetalWidth$
arg3 <- $SepalLength$
arg4 <- $SepalWidth$
arg5 <- $Species$
arg6 <- $SampleID$

#Use the K-means algorithm to classify the Petal Length, Petal Width, Sepal Length, and Sepal Width into three clusters
result <- kmeans(data.frame(arg1, arg2, arg3, arg4), 3);

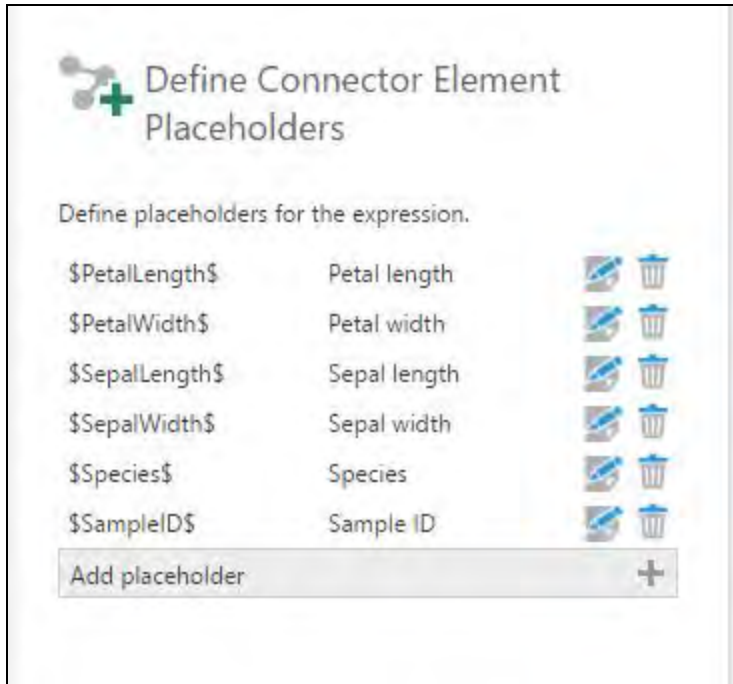
#Dundas BI requires that the output be in the form of a data frame
output <- data.frame(arg1, arg2, arg3, arg4, result$cluster, arg5, arg6);
```

This script is using the [K-means clustering algorithm](#), which requires multiple arguments and defines the number of clusters to generate out of those variables.



Note: The number of clusters is an input parameter (in this example we used 3 as the number of clusters). An inappropriate choice of k may yield poor results. That is why, when using the K-means model, it is important to run diagnostic checks to determine the optimal number of clusters in the data set.

In the same configuration panel, click **Define Element Placeholders** and add the placeholders for all the arguments defined in the script. Make sure the column names match with something that the R script recognizes.



Open the data preview. You can see that the new result.cluster column, which is classifying each data point into clusters 1, 2, or 3.

arg1	arg2	arg3	arg4	result.cluster	arg6	arg5
1.40	0.20	5.10	3.50	1	1	I. setosa
1.40	0.20	4.90	3.00	3	2	I. setosa
1.30	0.20	4.70	3.20	3	3	I. setosa
1.50	0.20	4.60	3.10	3	4	I. setosa
1.40	0.20	5.00	3.60	1	5	I. setosa
1.70	0.40	5.40	3.90	1	6	I. setosa
1.40	0.30	4.60	3.40	1	7	I. setosa

Note: arg5, the species we are trying to predict, isn't necessary for this example but is used here to help us assess the accuracy of the cluster detection by comparing it to known clusters - in this case the species of the iris flower.

Using SQL Server R Services

Alternatively, if you are using a database such as SQL Server or Azure SQL with R Services or [Machine Learning Services](#), you may be able to run R script as part of the SQL query in a Manual Select transform.

Create a [Manual Select](#) data cube using the following script:

```
exec sp_execute_external_script
    @language = N'R'
    , @script = N'
        df <- data.frame(InputDataSet)
        arg1 <- df$SepalLength
        arg2 <- df$SepalWidth
        arg3 <- df$PetalLength
        arg4 <- df$PetalWidth
```

```

arg5 <- df$Species
arg6 <- df$SampleID

result <- kmeans(data.frame(arg1, arg2, arg3, arg4), 3);
OutputDataSet <- data.frame(arg1, arg2, arg3, arg4, result$cluster, arg5, arg6);
, @input_data_1 = N'
Select 5.1 As [SepalLength], 3.5 As [SepalWidth], 1.4 As [PetalLength], 0.2 As [PetalWidth], 'I. setosa' as [Species], 1
As [SampleID] UNION ALL
Select 4.9, 3, 1.4, 0.2, 'I. setosa', 2 UNION ALL
Select 4.7, 3.2, 1.3, 0.2, 'I. setosa', 3 UNION ALL
.....
Select 6.5, 3, 5.2, 2, 'I. virginica', 148 UNION ALL
Select 6.2, 3.4, 5.4, 2.3, 'I. virginica', 149 UNION ALL
Select 5.9, 3, 5.1, 1.8, 'I. virginica', 150'
WITH RESULT SETS (([Sepal Length] float, [Sepal width] float, [Petal length] float, [Petal width] float, [Cluster Result]
int, [Species] varchar(MAX),[Sample ID] int));

```

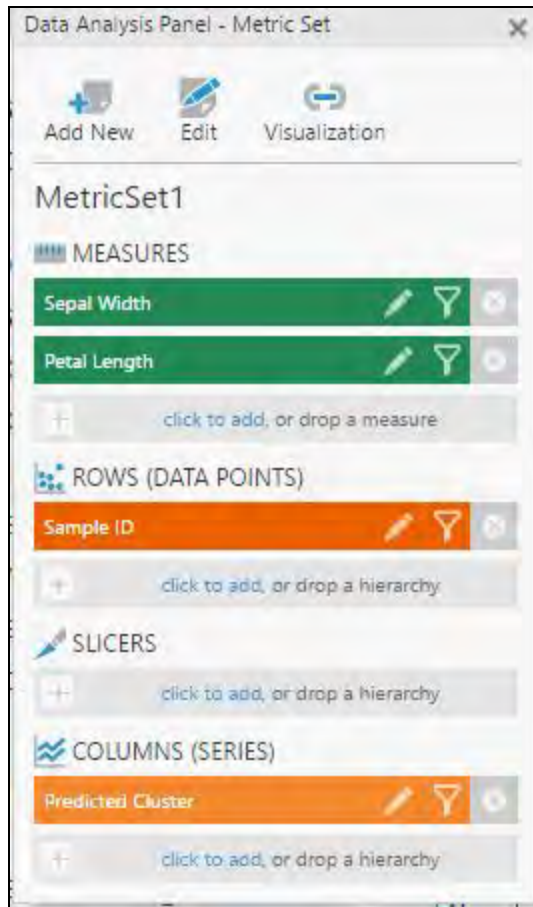
This script specifies the language (i.e., R) and executes the K-means algorithm as a standard stored procedure. It also includes the Select statements for retrieving the data points from the database.

The same result is obtained as with using Rserve:

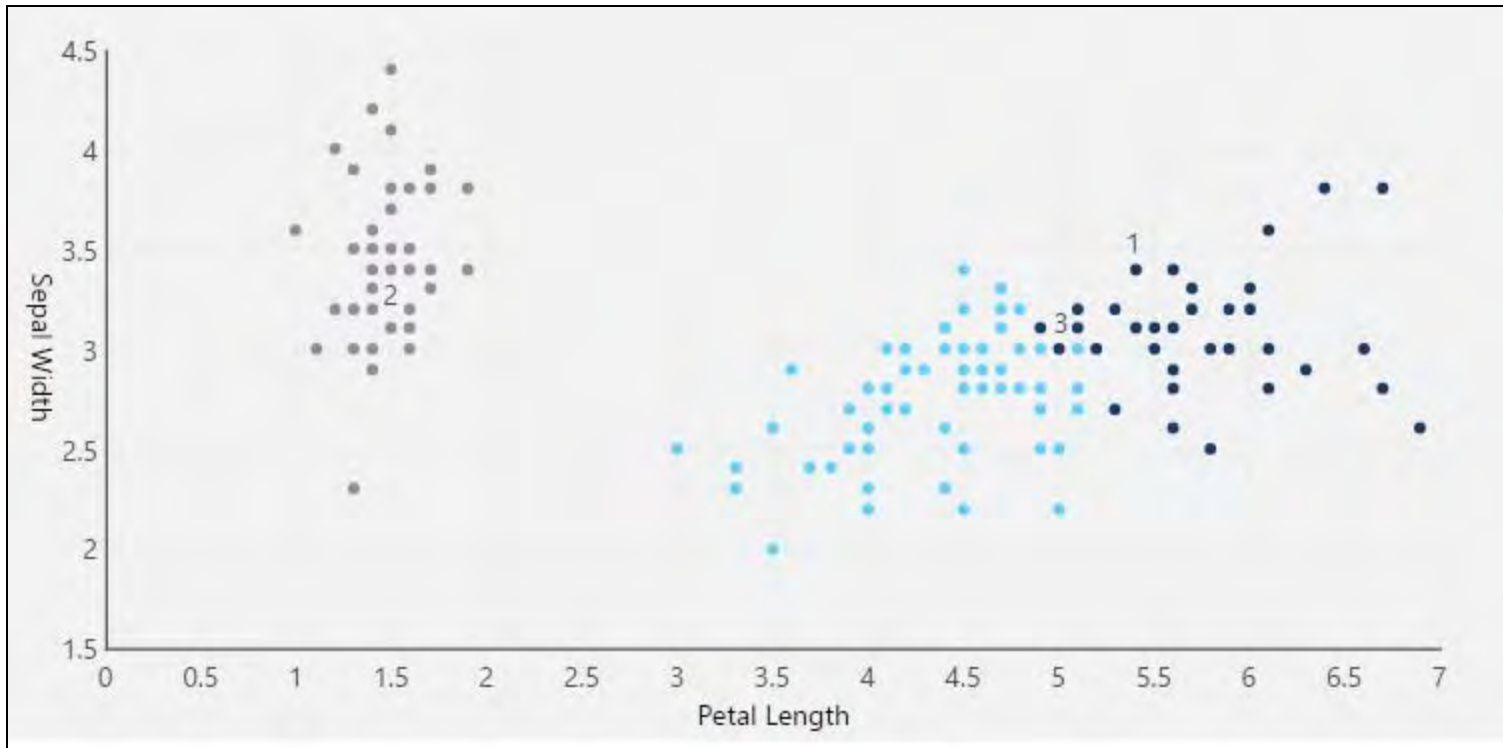
Sepal width	Petal length	Petal width	Sample ID	Sepal Length	Cluster Result	Species
3.5	1.4	0.2	1	5.10	2	I.?setosa
3.0	1.4	0.2	2	4.90	2	I.?setosa
3.2	1.3	0.2	3	4.70	2	I.?setosa
3.1	1.5	0.2	4	4.60	2	I.?setosa
3.6	1.4	0.2	5	5.00	2	I.?setosa
3.9	1.7	0.4	6	5.40	2	I.?setosa

Use the Results

From the data cube created above, select the two measures, and add Sample ID to Rows in the Data Analysis Panel of a metric set. Add the Predicted Cluster dimension to Columns, so that each Sample ID is grouped by the cluster.



Revisualize to a Scatter Plot. You will see that all the samples are grouped by cluster.



Note: The clusters may change every time the data is retrieved (assuming that data is not cached). This may happen if the particular K-means algorithm used isn't deterministic and has a random component used to define the clusters.

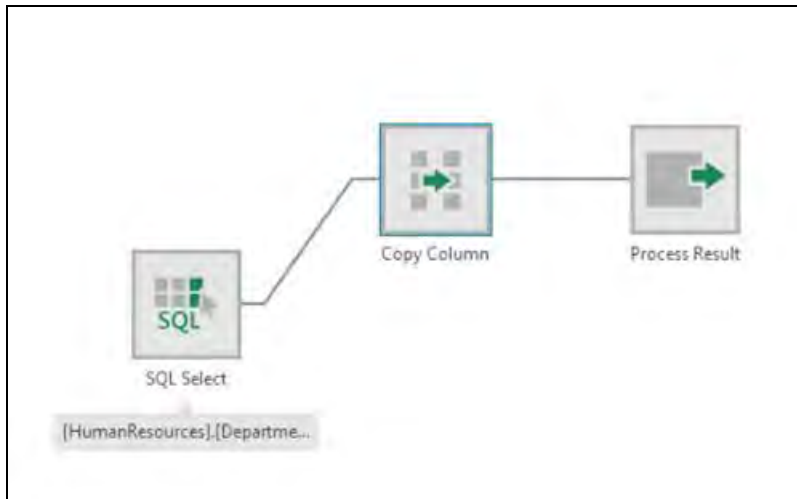
R Resources:

- The R Project for [Statistical Computing](#)
- The Comprehensive R [Archive Network](#)
- [An Introduction to R](#)
- [The R Datasets Package](#)

Copy Element

This applies to: *Managed Dashboards, Managed Reports*

The Copy Element transform creates new columns by copying selected input columns and adding the new columns to the output. Use in cases where you need to manipulate a column but want to keep the original column as well.



Input

The Copy Element transform requires one input transform that has at least one column.

Configure

In the Copy Element transform configuration dialog, click the Copy icon next to a column to add a copy. It will be listed below under Copied Elements.

SETTINGS

Choose the elements to use in this transform. Click the copy button to create a copy of the element.




☒	DepartmentID	Int16	
☒	Name	String(50)	
☒	GroupName	String(50)	
☒	ModifiedDate	DateTime	

COPIED ELEMENTS

Name	Original Element	Edit
NameCopied	Name	 

You can use this same transform to add more copies for the same column or for other columns.

Elements that are unchecked under Input Elements won't be included in the output. You can rename the elements by clicking Edit output elements.

Output

The figure below illustrates the output where a Name column was copied.



The screenshot displays the Logi Symphony v24 interface. At the top, a workflow diagram shows three steps: 'SQL Select', 'Copy Column', and 'Process Result', connected by arrows. Below the workflow, a data preview table is visible. The table has five columns: 'DepartmentID', 'Name', 'GroupName', 'ModifiedDate', and 'NameCopied'. It contains 10 rows of data, including departments like Engineering, Tool Design, Sales, Marketing, Purchasing, Research and Development, Production, Production Control, Human Resources, and Finance.

DepartmentID	Name	GroupName	ModifiedDate	NameCopied
1	Engineering	Research and Development	06/01/2002 00:00:00	Engineering
2	Tool Design	Research and Development	06/01/2002 00:00:00	Tool Design
3	Sales	Sales and Marketing	06/01/2002 00:00:00	Sales
4	Marketing	Sales and Marketing	06/01/2002 00:00:00	Marketing
5	Purchasing	Inventory Management	06/01/2002 00:00:00	Purchasing
6	Research and Development	Research and Development	06/01/2002 00:00:00	Research and Development
7	Production	Manufacturing	06/01/2002 00:00:00	Production
8	Production Control	Manufacturing	06/01/2002 00:00:00	Production Control
9	Human Resources	Executive General and Administration	06/01/2002 00:00:00	Human Resources
10	Finance	Executive General and Administration	06/01/2002 00:00:00	Finance

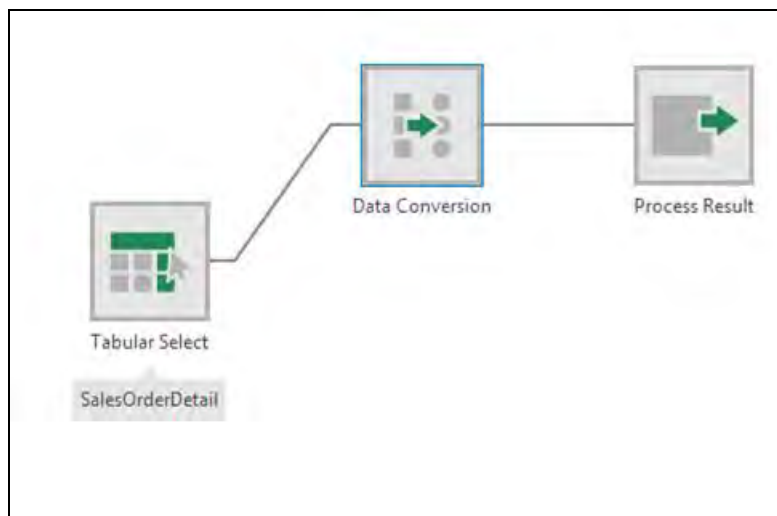
For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Data Conversion

This applies to: *Managed Dashboards, Managed Reports*

The Data Conversion transform allows you to change the data type of a column.



Input

This transform requires one input transform with at least one column.

As an example, data coming from a CSV file may have been recognized as a string of text when it's known to contain dates.

	SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal	rowguid	ModifiedDate
1	43659	1	4911-403C-98	1	776	1	2024.994	0	2024.994	B207C96D-D9E6-402B-8470-2CC176C42283	7/1/2005
2	43659	2	4911-403C-98	3	777	1	2024.994	0	6074.982	7ABB600D-1E77-41BE-9FE5-B9142CFC08FA	7/1/2005
3	43659	3	4911-403C-98	1	778	1	2024.994	0	2024.994	475CF8C6-49F6-486E-B0AD-AFC6A50CDD2F	7/1/2005
4	43659	4	4911-403C-98	1	771	1	2039.994	0	2039.994	04C4DE91-5815-45D6-8670-F462719FBCE3	7/1/2005
5	43659	5	4911-403C-98	1	772	1	2039.994	0	2039.994	5A74C7D2-E641-438E-A7AC-37BF23280301	7/1/2005
6	43659	6	4911-403C-98	2	773	1	2039.994	0	4079.988	CE472532-A4C0-45BA-816E-EEFD3FDS48B3	7/1/2005
7	43659	7	4911-403C-98	1	774	1	2039.994	0	2039.994	80667840-F962-4EE3-96E0-AECA108E0D4F	7/1/2005
8	43659	8	4911-403C-98	3	714	1	28.8404	0	86.5212	E9D54907-E7B7-4969-80D9-76BA69F8A836	7/1/2005
9	43659	9	4911-403C-98	1	716	1	28.8404	0	28.8404	AA542630-BDCD-4CE5-89A0-C1BF82747725	7/1/2005
10	43659	10	4911-403C-98	6	709	1	5.7	0	34.2	AC769034-3C2F-495C-A5A7-3B71CDB25D4E	7/1/2005
11	43659	11	4911-403C-98	2	712	1	5.1865	0	10.373	06A66921-6B9F-4199-A912-DDAFD383472B	7/1/2005
12	43659	12	4911-403C-98	4	711	1	20.1865	0	80.746	0E371EE3-253E-4BB0-B813-83CF4224F972	7/1/2005
13	43660	13	6431-4D57-83	1	762	1	419.4589	0	419.4589	419A1302-AC7A-4044-97B2-66D9D14CD02E	7/1/2005
14	43660	14	6431-4D57-83	1	758	1	874.794	0	874.794	5D0B2B03-1D4C-4C34-9696-C14C58E7301C	7/1/2005
15	43661	15	4E0A-4F89-AE	1	745	1	809.76	0	809.76	EDE1759E-6733-4C7B-A43F-DC6F48002D8A	7/1/2005
16	43661	16	4E0A-4F89-AE	1	743	1	714.7043	0	714.7043	FE10BF09-D477-485B-9541-27AE8053A6D4	7/1/2005
17	43661	17	4E0A-4F89-AE	2	747	1	714.7043	0	1429.4086	B136852E-24C9-4006-804E-B14AEFE6C337	7/1/2005
18	43661	18	4E0A-4F89-AE	4	712	1	5.1865	0	20.746	F0F410B6-BC93-4D77-B70D-E4E2465D98EE	7/1/2005
19	43661	19	4E0A-4F89-AE	4	715	1	28.8404	0	115.3616	AE4CDA7D-9B6E-4ABB-A391-E9E9DD810125	7/1/2005
20	43661	20	4E0A-4F89-AE	2	742	1	722.5949	0	1445.1898	054A9D3B-A178-4656-B2B0-9B9BDDF1C345	7/1/2005



Note: If you have access, you can change the data type for a flat file by [editing the data connector](#).

Configure

In the Data Conversion configuration dialog, you can choose a new data type for each column from the corresponding drop-down list:



Data Conversion

[more help](#)

The transform converts the data in the selected input element(s) to a different data type.

Name

Description

ID

SETTINGS

Choose the elements to use in this transform, and select the data conversion.



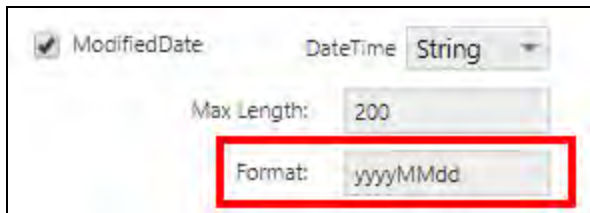

☒	CarrierTrackingNum...	String(25)	DateTime
		Format:	
☒	LineTotal	Double	--
☒	ModifiedDate	DateTime	String
		Max Length:	200
		Format:	
☒	OrderQty	Int16	--
☒	ProductID	Int32	--

When you choose String as the data type, you can specify a value for **Maximum Length**. The default value is 200.



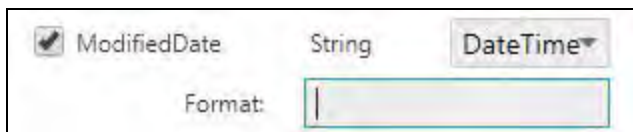
A screenshot of the Logi Symphony configuration interface. It shows a field named 'Name' with a data type dropdown set to 'String'. Below the field name, there is a 'Max Length' input field containing the value '200'.

When converting a DateTime value to a string, you can specify a **Format**.



A screenshot of the Logi Symphony configuration interface. It shows a field named 'ModifiedDate' with a data type dropdown set to 'String'. Below the field name, there is a 'Max Length' input field containing the value '200'. A red box highlights the 'Format' input field, which contains the value 'yyyyMMdd'.

When choosing DateTime, a Format field will appear, allowing you to enter the format to be used to correctly identify each part of the text as a part of the date and time.

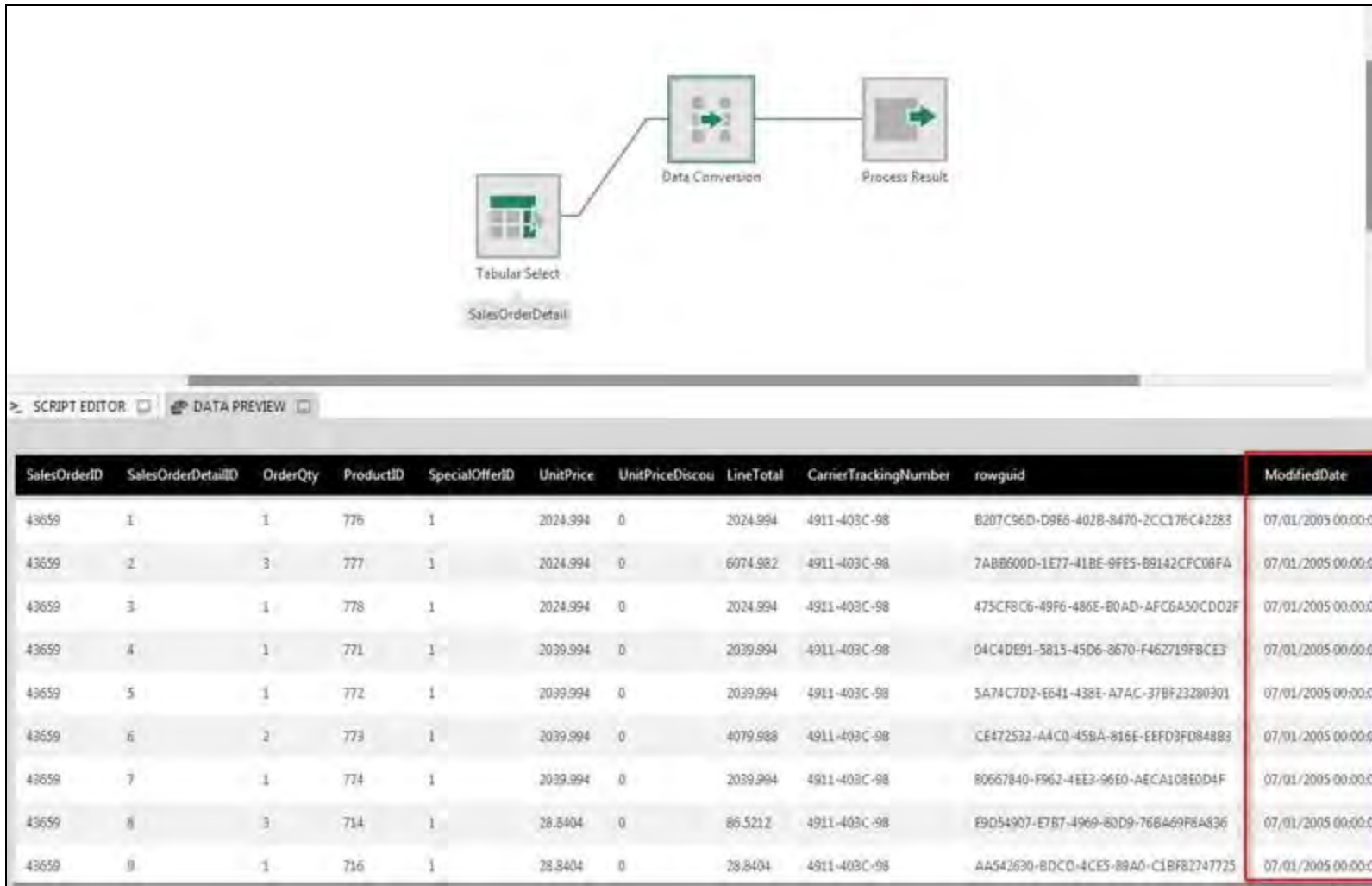


A screenshot of the Logi Symphony configuration interface. It shows a field named 'ModifiedDate' with a data type dropdown set to 'DateTime'. Below the field name, there is a 'Format' input field, which is currently empty.

For example, the format for the text 2014-01-21T20:28:21.703 would be yyyy-MM-ddTHH:mm:ss.fff. See [.NET Custom Date and Time Format Strings](#) for details.

Output

The figure below illustrates the output from the transform in our example.



The screenshot displays the Logi Symphony v24 interface. At the top, a data flow diagram shows a sequence of steps: 'SalesOrderDetail' (a data source icon) connects to 'Tabular Select' (a filter icon), which then connects to 'Data Conversion' (a conversion icon), and finally to 'Process Result' (a result icon). Below the diagram, the 'DATA PREVIEW' tab is active, showing a table with 11 columns and 9 rows of data. The columns are: SalesOrderID, SalesOrderDetailID, OrderQty, ProductID, SpecialOfferID, UnitPrice, UnitPriceDiscou, LineTotal, CarrierTrackingNumber, rowguid, and ModifiedDate. The 'ModifiedDate' column is highlighted with a red border. The data in the table is as follows:

SalesOrderID	SalesOrderDetailID	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscou	LineTotal	CarrierTrackingNumber	rowguid	ModifiedDate
43659	1	1	776	1	2024.994	0	2024.994	4911-403C-98	B207C96D-D966-402B-8470-2CC176C42283	07/01/2005 00:00:00
43659	2	3	777	1	2024.994	0	6074.982	4911-403C-98	7AB8600D-1E77-41BE-9FE5-89142CFC08FA	07/01/2005 00:00:00
43659	3	1	778	1	2024.994	0	2024.994	4911-403C-98	475CF8C6-49F6-486E-B0AD-AFC6A30CDD2F	07/01/2005 00:00:00
43659	4	1	771	1	2039.994	0	2039.994	4911-403C-98	04C4DE91-5815-45D6-8670-F462719FBCF3	07/01/2005 00:00:00
43659	5	1	772	1	2039.994	0	2039.994	4911-403C-98	5A74C7D2-E641-438E-A7AC-37BF23280301	07/01/2005 00:00:00
43659	6	2	773	1	2039.994	0	4079.988	4911-403C-98	CE472532-A4C0-45BA-816E-EEFD3FD848B3	07/01/2005 00:00:00
43659	7	1	774	1	2039.994	0	2039.994	4911-403C-98	80667840-F962-4EE3-96E0-AECA108E0D4F	07/01/2005 00:00:00
43659	8	3	714	1	28.8404	0	86.5212	4911-403C-98	F9D54907-E7B7-4969-80D9-768A69F8A836	07/01/2005 00:00:00
43659	9	1	716	1	28.8404	0	28.8404	4911-403C-98	AA542630-BDC0-4CE5-89A0-C1BF82747725	07/01/2005 00:00:00

For more information, see:

- [Microsoft: Custom Date and Time Format Strings](#)
- [Using Transforms](#)



- Archive of documentation for Logi Symphony v24

- [Create a New Data Cube](#)
- [Data Cubes](#)

Data Cube Select

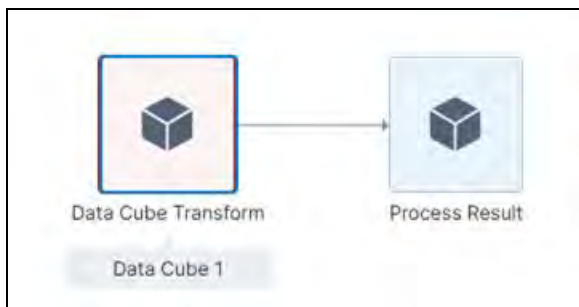
This applies to: *Managed Dashboards, Managed Reports*

The Data Cube transform is created when another data cube is dragged onto the canvas. It allows you to reuse that data cube's process flow as an input to your data cube.

The input to the other cube's [process result](#) transform is used as an input to this data cube.

Add the Transform

Expand the data connector in the Data Cubes folder in the Explore window, then drag a different data cube to the canvas. A Data Cube transform is added.



You can optionally configure the transform by selecting the transform and clicking **Configure** in the toolbar or the context menu.

Configure

The **Data Cube Transform** dialog shows the original data cube's name as a hyperlink you can click to open the source cube.

Help 

Data Cube Transform

The node responsible for retrieving data from a Data Cube.

Name

Color

Data Cube 1



Description

ID

f732b8b9-8d0a-4fc2-b8f7-df065ae422f2

Settings

Timeout

30

Data Cube

[Data Cube](#)

The elements discovered from the Data Cube.



Search Input Elements

Parameters





In the data cube transform itself, you can de-select input elements, set up parameters, and rename output elements, similar to other [select transforms](#). In Symphony v24.3 and later, you can add expressions if the data cube you're selecting from uses [warehouse storage](#).

Click the submit button at the bottom of the dialog when finished.

Output

The Data Preview window will show the same output as shown when previewing the process result transform of the source data cube.



Note: A single source data cube cannot be used twice in the same data cube, including the entire chain of referenced data cubes (e.g., two different data cubes that reuse the same data cube cannot be combined together).



Note: If the source data cube is warehouse-cached, the target data cube will also be using data from the warehouse, unless a different storage type is selected for the target data cube. In contrast, if the source data cube is in-memory-cached, the target data cube will not be pulling data from memory.



Note: Settings in the source data cube's process result are carried over when selected in another data cube, such as hierarchies that were selected to replace 'implicit' generated hierarchies.

For more information, see:

- [Using Transforms](#)
- [SQL Select](#)
- [Process Result](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)



Data Input

This applies to: *Managed Dashboards, Managed Reports*

The Data Input transform lets you reference a warehouse data storage area containing user input data.

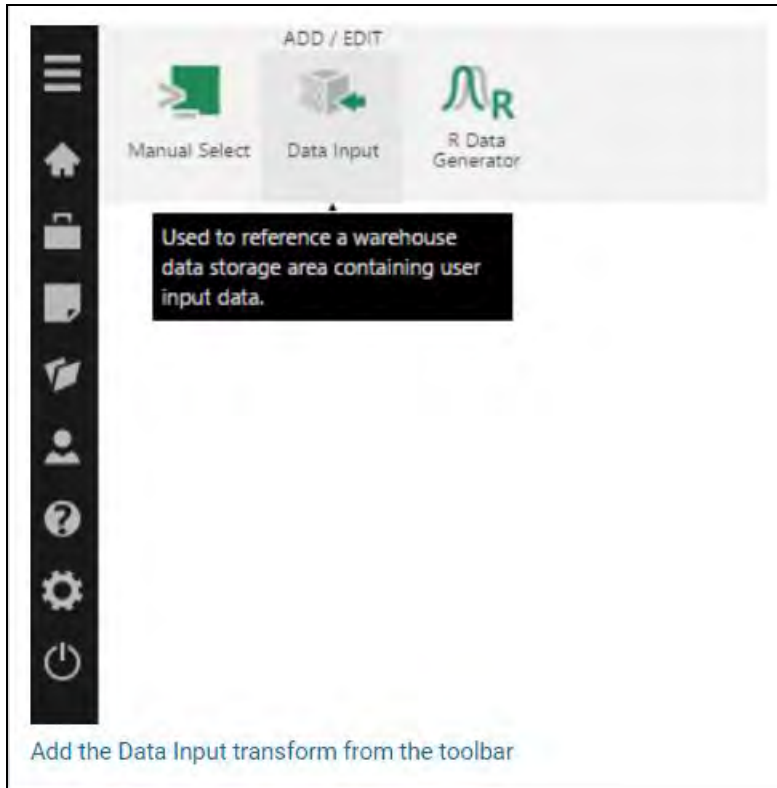
You can use the Data Input transform to add and modify data from inside Symphony. You can input data yourself, or set up a [data input interaction](#) for viewers to add to it.

Input

Input into the Data Input transform is done manually after the transform has been configured. It does not require another transform for input and generates output based on user-provided data.

Add the Transform

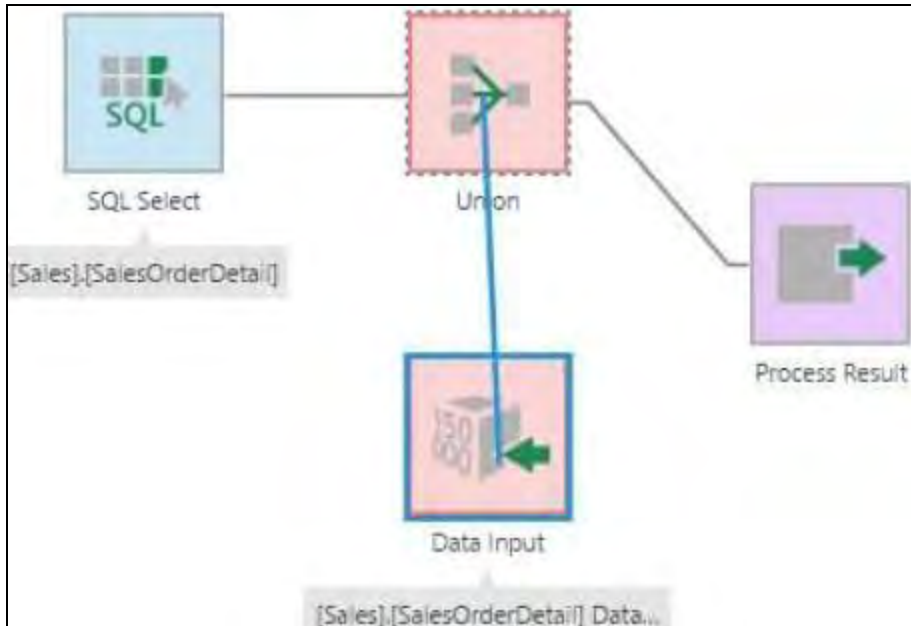
When creating a new data cube, you can add the Data Input transform to an empty canvas from the toolbar.



The Data Input transform is added to the data cube and connected to a Process Result transform automatically.



You can also add the Data Input transform from the toolbar to an existing data cube process. A typical example is to connect the Data Input transform to a [Union](#) transform that merges data from multiple inputs.



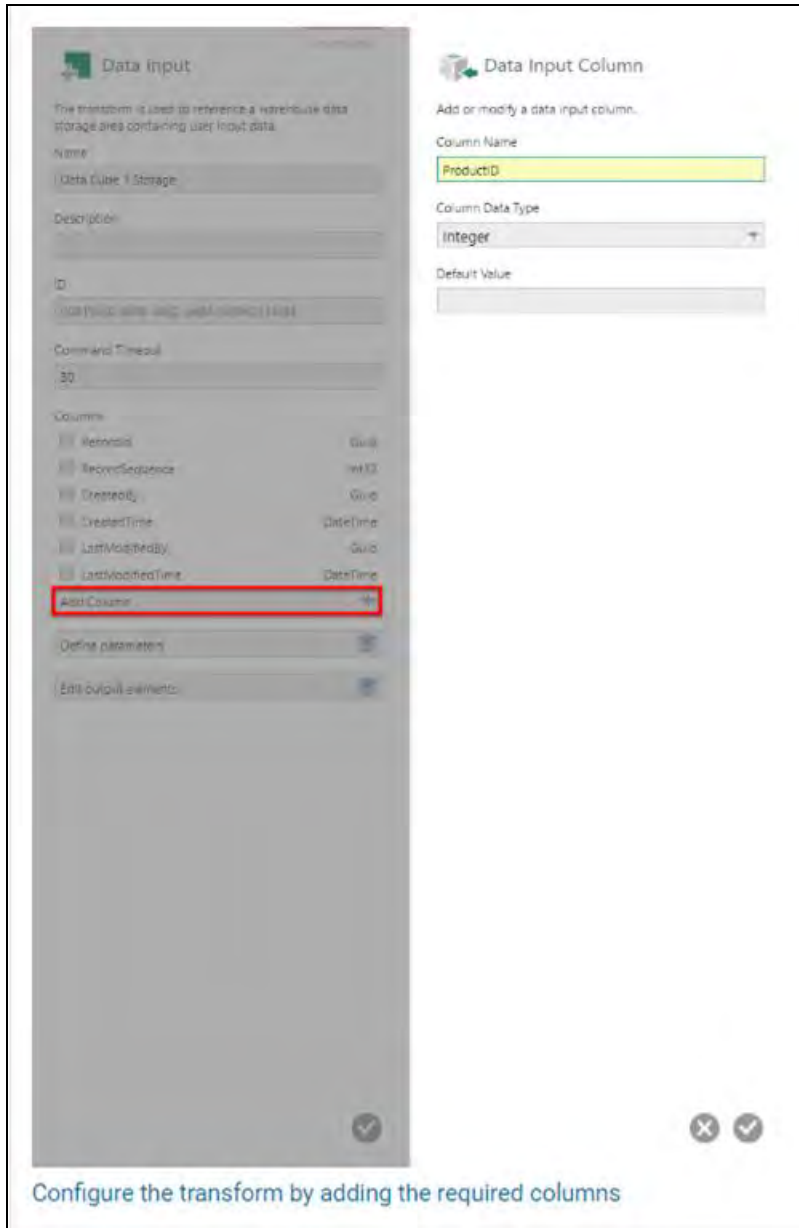
Configure the Transform

Double click the Data Input transform, or select or right-click it and choose the Configure option in the toolbar or its right-click menu.

The Data Input transform has six automatically created and populated columns. By default, these columns are not selected to be included in the output:

- RecordID
- RecordSequence
- CreatedBy
- CreatedTime
- LastModifiedBy
- LastModifiedTime

In the configuration dialog for the transform, the key task is to add new columns that will contain the input data. Click **Add Column** and complete the *Data Input Column* dialog.

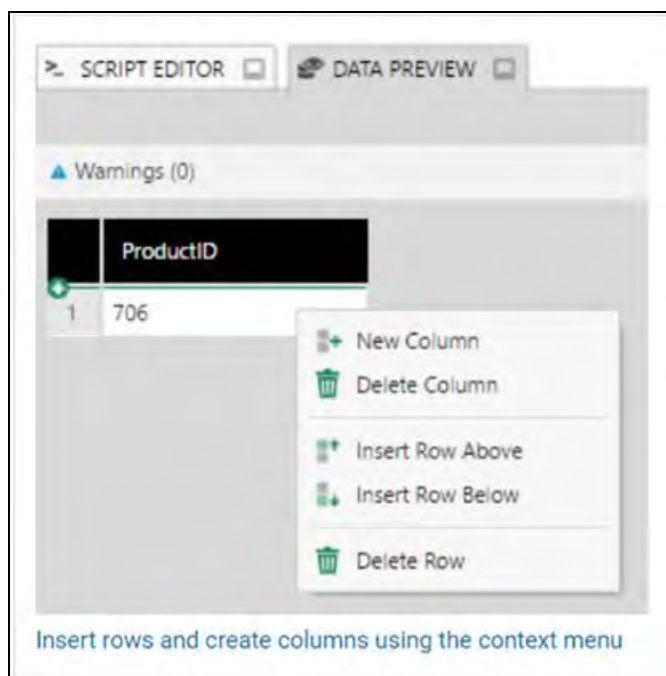


Output

The output is user-input data, which you can both view and edit by selecting the Data Input transform and opening the Data Preview window.

When entering data, type **Enter** to create or move to the next row, and **Tab** to create or move to the next cell.

You can right-click a cell to insert a row or create a new column, or move the mouse and click the **+** icon when it appears where you want to add a row or column.



-  **Note:** Deleting a column from the Data Input transform is postponed until after you check in. This means you will be unable to create a column with the same title as the deleted one until you check in and check out the transform.
-  **Note:** Changes to rows of data are not affected by check in or check out states, as viewers or end users can be allowed to make changes. Undo Check Out does not change or remove data.
-  **Note:** Users require the [Data Input privilege](#) to be able to input data.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Setting Up Data Input](#)
- [Set Up States on a Metric Set](#)
- [Configuration settings](#)

Display 'top N + other' Measures Using Data Cube Transforms

This applies to: *Managed Dashboards, Managed Reports*

This article shows how to display the top/bottom *N* records and the sum of the remaining records using data cube transforms.



Note: Metric sets have settings that can perform the same task: see [Displaying Top/Bottom Records](#).

Set Up the Data Cube

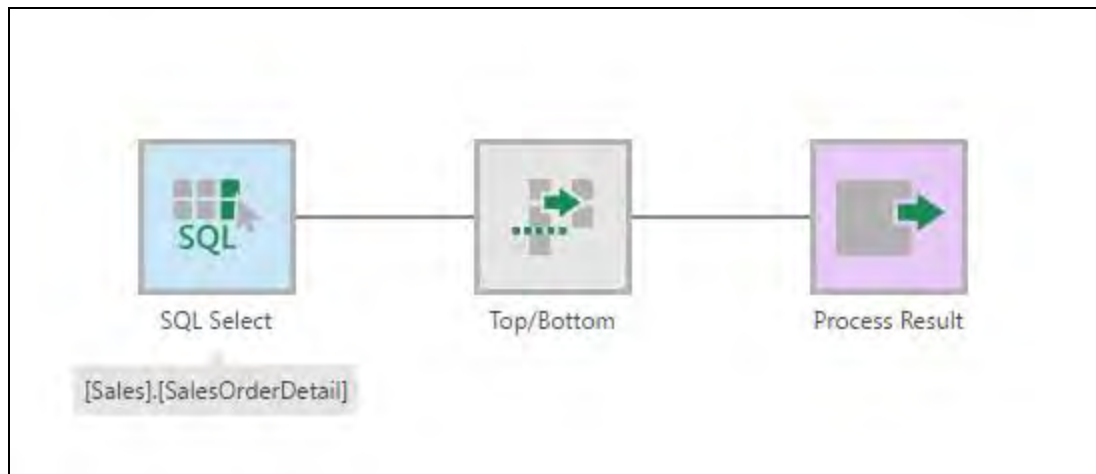
This example uses the Adventure Works database to display the top 4 Product IDs by *LineTotal* price, and the sum of the rest of the *LineTotal* values.

Under the data connector, find the *[Sales].[SalesOrderDetail]* table, and drag it onto the data cube canvas.

An automatic ETL process consisting of **SQL Select transform** and the **Process Result** is created.

Top/Bottom Transform

Select the connection link and insert a **Top/Bottom** transform from the toolbar.



Right-click on the **Top/Bottom** Transform and select **Configure**.

In the configuration dialog, insert the number of records you want to see in the **Record Count**. From the **Sorting Transform Input**, drag the measure being sorted for the Top/bottom records (*LineTotal*).

Top/Bottom

[more help](#)

The transform allows various sorting rules to be applied to the input columns and returns the specified quantity of results to the output.

Name

Description

Record Count

Starting Point

Click on the sort icon of each transform element to toggle its sorting method. Drag and drop within the list to set the priority.



Sorting Transform Input

☐	CarrierTrackingNumber
☒	LineTotal
☐	ModifiedDate
☐	OrderQty
☒	ProductID
☐	rowguid
☐	SalesOrderDetailID
☐	SalesOrderID
☐	SpecialOfferID
☐	UnitPrice
☐	UnitPriceDiscount

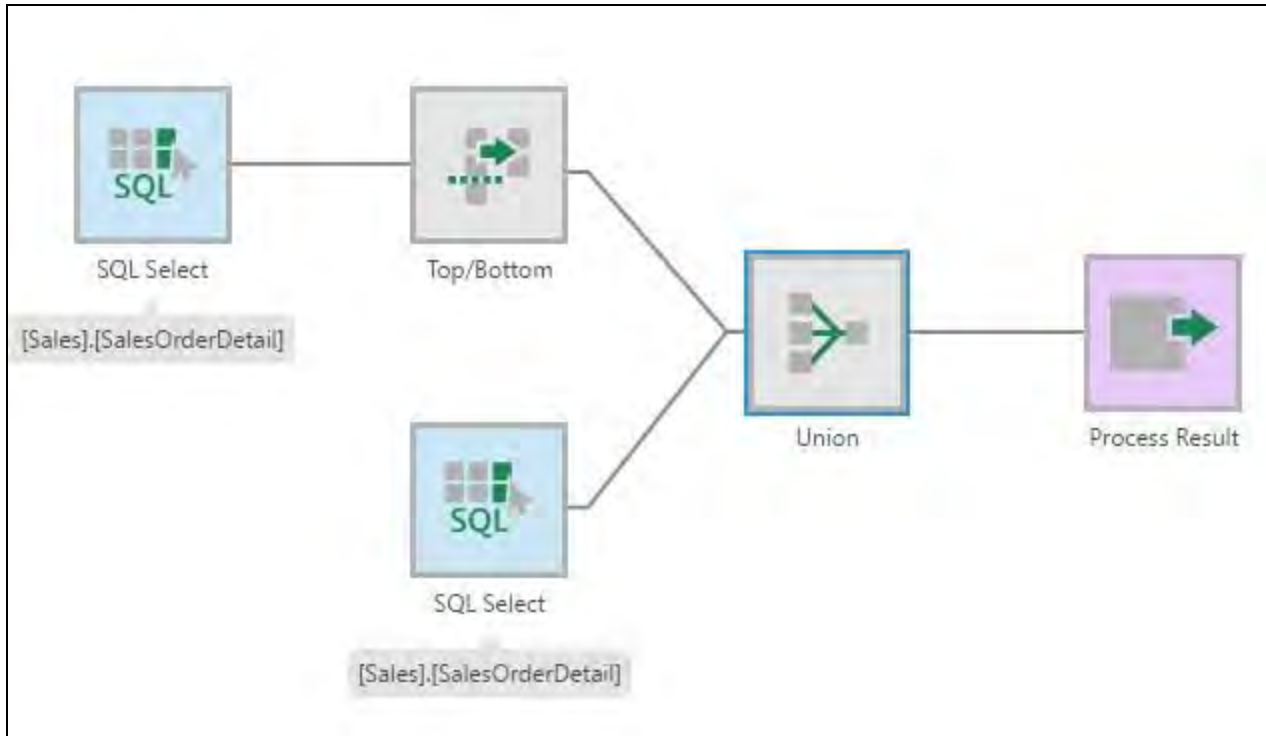


Sorting Elements

LineTotal		

Union Transform

Drag and drop the *[Sales].[SalesOrderDetail]* table again onto the canvas to create another SQL Select transform. Combine the outputs of the Top/Bottom transform and the new SQL Select using a **Union** transform.



Sort Transform

Select the connection link between the second SQL Select and the Union transform, and insert a Sort transform.

Right-click the transform and configure it. In the configuration dialog, drag *LineTotal* from the **Sorting Transform Input** to the **Sorting Elements** and click its arrow button to sort it in Descending order.

Sort

[more help](#)

The transform allows various sorting rules to be applied to the input columns and copies the results to the output. The sorting applied is only applicable inside the data cube. The sort transform should never be the last transform before the process result.

Name

Description

Drag and drop the elements you want to sort your data on into the 'Sorting Elements' section. Click on the sort icon of each transform element to change its sorting method. Drag and drop within the list to set the priority.



Sorting Transform Input

☒ LineTotal

☒ ProductID



Sorting Elements

LineTotal
 


Define parameters



Rename output elements





Rank Transform

Select the connection link between the Sort transform and the Union transform, and insert a Rank transform.

Right-click on the transform and configure it. In the configuration dialog, *Linetotal* from Rank Transform Input to Ranking Elements.

[more help](#)

Rank

The transform assigns a rank to each row. The data needs to be sorted according to the ranking elements by the transform before it. The data can be sorted from the source or else a sort transform can be used.

Name

Description

Rank Element Name

Rank Element Description

Rank Type

Continue Sequence

☒


Rank Transform Input

☒	LineTotal
☒	ProductID



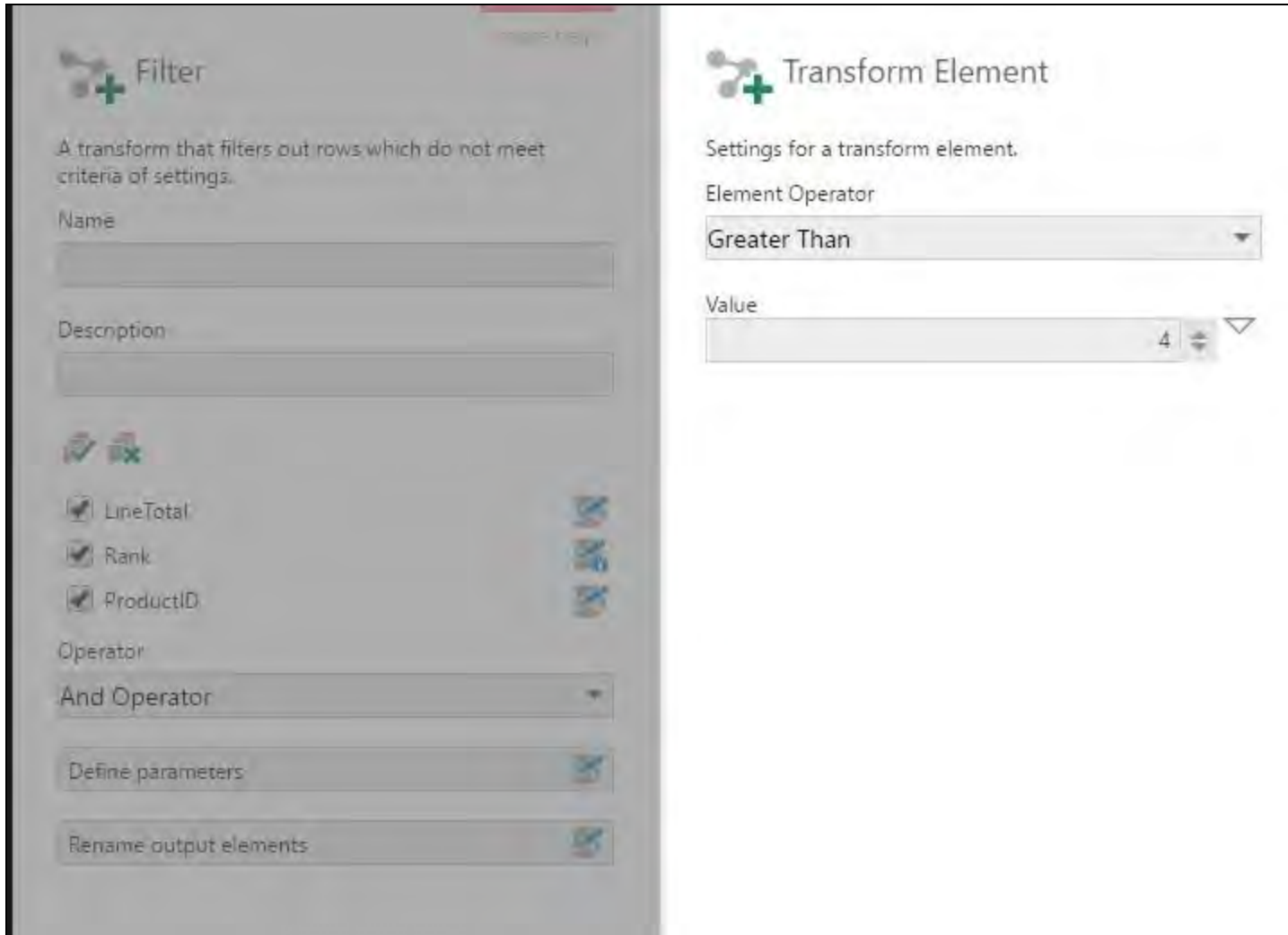
Ranking Elements

LineTotal	

Filter Transform

Select the connection link between **Rank** transform and the **Union** transform and insert a **Filter** transform.

Right-click on the transform and configure it. In the configuration dialog, click the **Edit** icon beside the Rank element. Choose **Greater Than** from the **Element Operator** drop-down and set the value as 4. This returns all the records that have a rank greater than 4 (i.e., all but the top 4 records).



The screenshot displays two side-by-side configuration panels. The left panel, titled 'Filter', describes the transform and provides fields for 'Name' and 'Description'. It lists three elements: 'LineTotal', 'Rank', and 'ProductID', each with a checked checkbox. Below this, there are dropdowns for 'Operator' and 'And Operator', and buttons for 'Define parameters' and 'Rename output elements'. The right panel, titled 'Transform Element', shows settings for a specific element. It includes a dropdown for 'Element Operator' set to 'Greater Than' and a 'Value' field set to '4'.



Aggregate Transform

Next, click on the connection link between the **Filter** transform and the **Union** transform and insert the Aggregate transform.

In its configuration dialog, uncheck all the measures or dimensions that are not needed. Set the **Aggregate** function for *LineTotal* as *Sum* and uncheck the *ProductID*.

Aggregate

[more help](#)

The transform allows aggregate functions to be applied to various input columns and copies the results to the output. It also allows setting Group By functionality for the columns that are not aggregated.

Name

Description

Choose the elements to use in this transform.

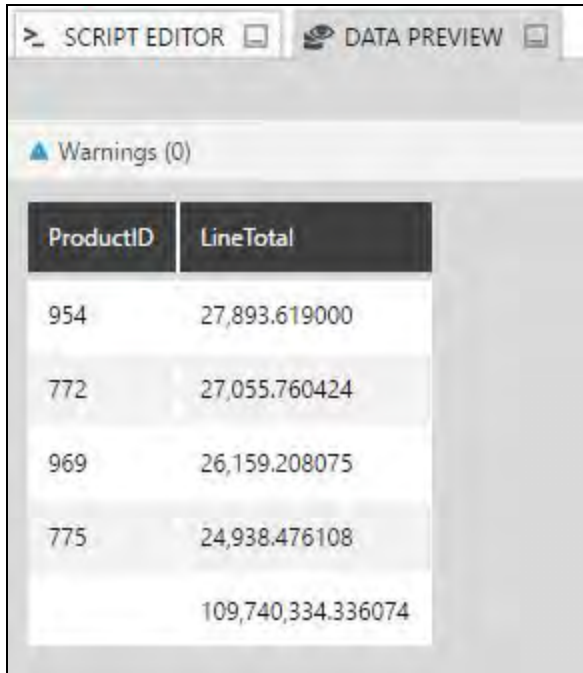


☒	LineTotal	Double
Aggregate Function Sum		
Group By ☐		
☐	Rank	Double
☐	ProductID	Int32
Define parameters		
Rename output elements		

Preview the Data

Select the **Union** transform and open the **Data Preview** window.

You can see the *ProductIDs* with the top 4 *LineTotal* values and an additional row with the sum of the rest of the *LineTotal* values.



ProductID	LineTotal
954	27,893.619000
772	27,055.760424
969	26,159.208075
775	24,938.476108
	109,740,334.336074

Notice that the sum of the rest of the *LineTotals* has an empty string for the *ProductID*.

Replace the Empty String

Click on the connection link between Union transform and the **Process Result** node.

In our case, *ProductID* contains numbers and not yet strings (text), so we will insert a Data Conversion transform and configure it. Change the type for *ProductID* from *Int32* to *String*.



Data Conversion

[more help](#)

The transform converts the data in the selected input element(s) to a different data type.

Name

Description

Choose the elements to use in this transform, and select the data conversion




☒ ProductID	Int32	String
☒ LineTotal	Double	--

Define parameters 

Rename output elements 

Calculated Element Transform

Click on the connection link between the **Data Conversion** transform and the **Process Result** node, and insert a **Calculated Element** transform. In its configuration dialog, uncheck *ProductID* and add a new calculated element.

Name this calculated element as *ProductID*. Change the **Data Type** to **String** and add the following code in the expression editor:



Calculated Element

The transform allows for creation of new elements by applying user-defined expressions. The result of each expression is a new output element.

Name

Description




☐ ProductID	String
☒ LineTotal	Double

Calculated Elements

ProductID	if (\$ProductID\$ == null)		







Calculated Element

Create a new element by writing an expression.

Settings

Name

Description

Data Type

Expression

```

1 if ($ProductID$ == null) {
2   return "Others";
3 }
4 else{
5   return $ProductID$;
6 }

```

Edit expression in editor 

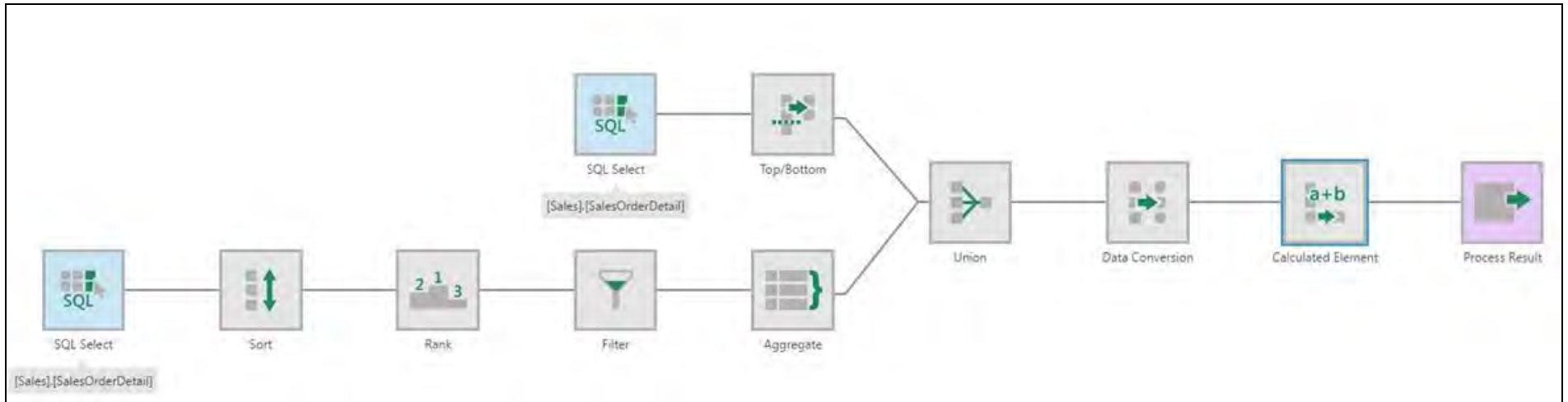
Automatic Placeholders



Note: A [Null Replacement](#) transform can also be used instead.

Submit the changes.

The final data cube looks like this:



Click on **Process Result** and preview the data in the **Data Preview** window.

SCRIPT EDITOR

DATA PREVIEW

Warnings (0)

LineTotal	ProductId
27,893.619000	954
27,055.760424	772
26,159.208075	969
24,938.476108	775
109,740,334.336074	Others



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Top/Bottom](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Display 'top N + other' Measures Using Stored Procedure](#)

Display 'top N + other' Measures Using Stored Procedure

This applies to: *Managed Dashboards, Managed Reports*

This article shows an example of how to use a stored procedure in a data cube with a parameter for selecting the top *N* records and grouping the rest as *Others* for an SQL data source.



Note: Metric sets have settings that can perform the same task: see [Displaying Top/Bottom Records](#).

Set Up the Data

SalesRepID	SalesRep	SalesActual
1	Eden Dillon	6,381
2	Tamara Landry	5,515
3	Gannon Wall	2,225
4	Georgia Hays	5,822
5	Sasha Price	7,049
6	Joe Barnes	3,487
7	Christopher Burkhart	1,463
8	Frank Smith	2,378
9	Alice Stamper	8,452
10	Maria Jones	4,892

The data in this example is a SQL relational database, with the goal of showing the **Top N Sales** using a *Single Number* filter on the dashboard to determine the number, with the rest of the values summed as a single *Others* item.

Create Stored Procedure

Create a new stored procedure on the relational SQL data source with a parameter which receives the number of top records we want to display. A stored procedure for the sample data above can be created like below:

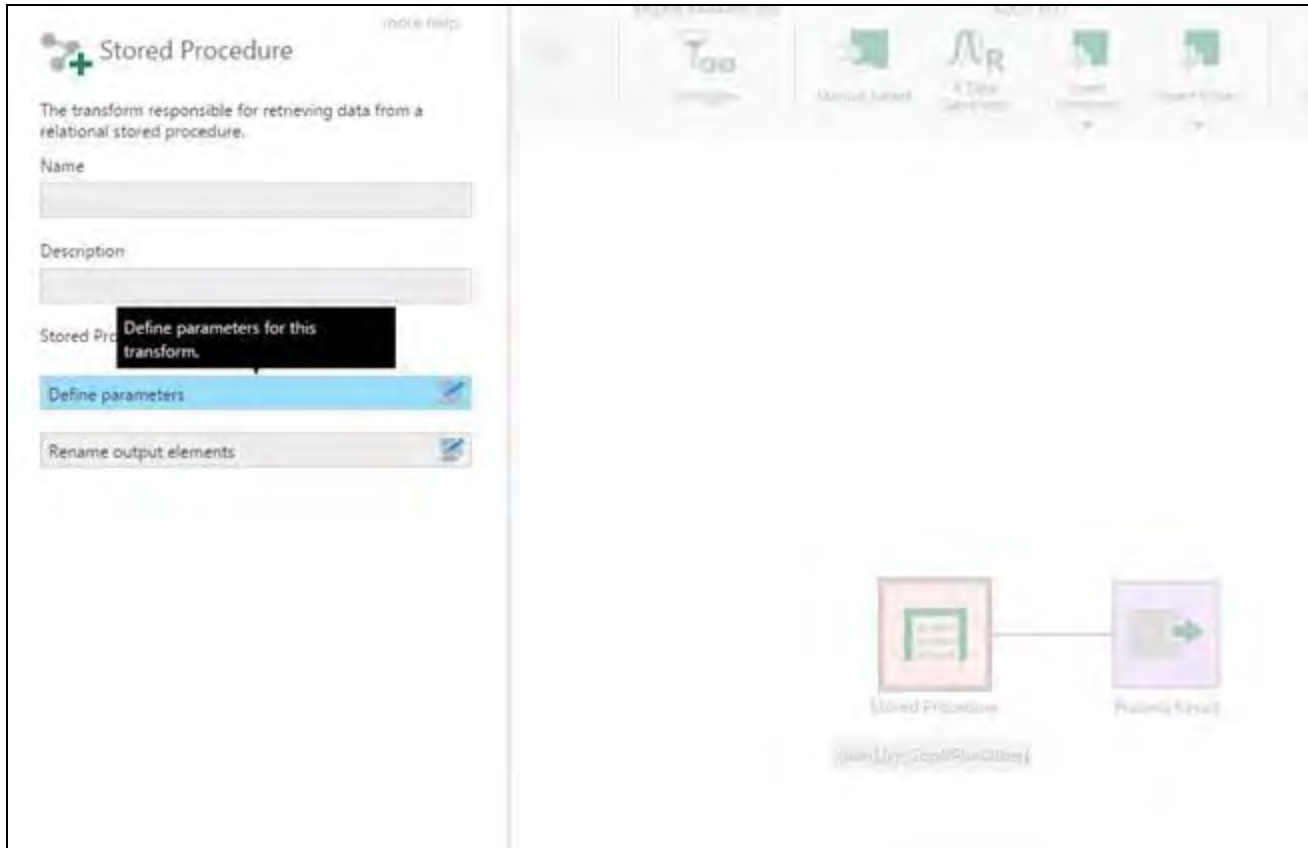
```
create procedure [dbo].[sp_TopNPlusOther]
    @topn int
as
begin
select [SalesRep], [SalesActual]
from (
    select top (@topn) [SalesRep], sum([SalesActual]) as [SalesActual]
    from [dbo].[myTable]
    group by
        [SalesRep]
    order by [SalesActual] desc
) as alias
union all
select '(OTHER)', sum([SalesActual])
from [dbo].[myTable]
where
[SalesRep] not in (
    select top (@topn) [SalesRep]
    from [dbo].[myTable]
    group by
        [SalesRep]
    order by SUM([SalesActual]) desc
)
end
```

Run the query to save this Stored Procedure.

Create a Data Cube

After creating a new data connector to the database with the above created stored procedure, create a new data cube. Drag and drop the stored procedure **sp_TopNPlusOther** onto the data cube canvas.

Double click the stored procedure transform, then click **Define Parameters**.



Select the **@topn** parameter defined in the stored procedure.

- Set the Parameter to **Public** to allow it be set from a dashboard
- Set the default **Value** to 3



Transform Parameters

Define transform parameters. Once 'Create Parameter' is clicked, the parameter will be created on the server.

@topn 

Add parameter 



Define Transform Parameter

Define the settings for the transform parameter.

Name

@topn

Public

☒

Value Type

☒ Single

Value

3 

Preview the data from the process result.

SCRIPT EDITOR DATA PREVIEW

Warnings (0)

SalesActual	SalesRep
8,452	Alice Stamper
7,049	Sasha Price
6,381	Eden Dillon
25,782	(OTHER)

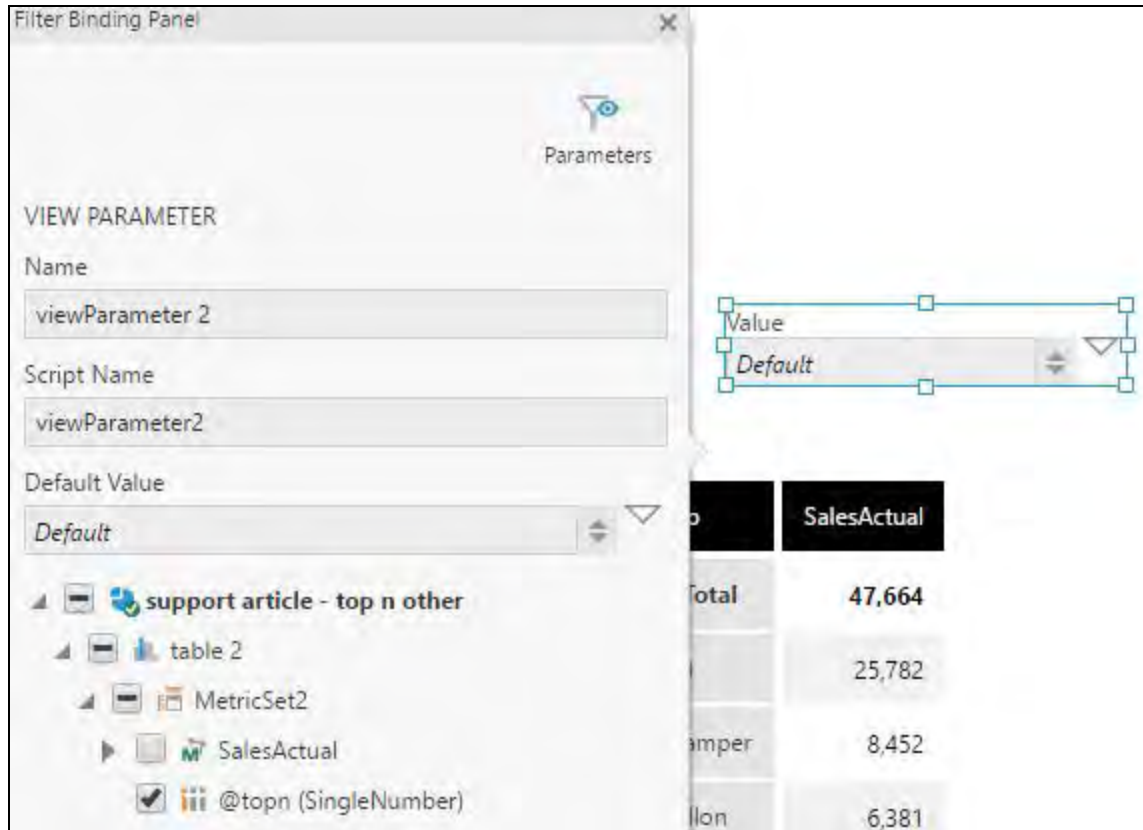
Check in the data cube.

Using the Data Cube On the Dashboard

On an existing or new dashboard, drag and drop the above created data cube.

Choose **Filters** in the toolbar, then select a **Single Number** filter.

Connect the Single Number filter to the **@topn** parameter under the table.



The screenshot shows the Logi Symphony interface. On the left, the 'Filter Binding Panel' is open, displaying the 'VIEW PARAMETER' section. The 'Name' field is set to 'viewParameter 2', the 'Script Name' is 'viewParameter2', and the 'Default Value' is 'Default'. Below this, a tree view shows the hierarchy: 'support article - top n other' (expanded), 'table 2' (expanded), 'MetricSet2' (expanded), 'SalesActual' (expanded), and '@topn (SingleNumber)' (checked). On the right, a data table is visible with the following data:

	SalesActual
Total	47,664
	25,782
amper	8,452
llon	6,381

By default, the table will show top 3 Sales and the sum of the rest of the Sales. Switch to View mode and change the value in the Single Number filter to change the number of top Sales that are visible.

Value ▼

SalesRep	SalesActual
Alice Stamper	8,452
Eden Dillon	6,381
Georgia Hays	5,822
Sasha Price	7,049
Tamara Landry	5,515
(OTHER)	14,445

For more information, see:

- [Top/Bottom](#)
- [Stored Procedure Select](#)
- [Passing Parameter Value From Dashboard to Stored Procedure](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)



- Archive of documentation for Logi Symphony v24

- [Display 'top N + other' Measures Using Data Cube Transforms](#)
- [Top/Bottom](#)

Filter Data Cube Using Complex Logical Expression

This applies to: *Managed Dashboards, Managed Reports*

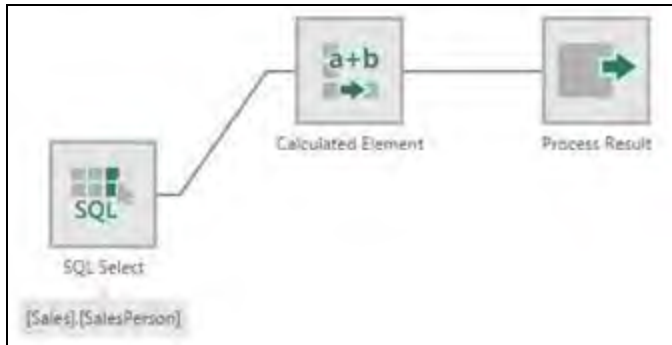
This article shows you how to filter the data cube with complex logical expressions.

For many filter operations, the [Filter](#) transform can be used, which allows you to use either an **And** or **Or** operator between multiple conditions. If you need to combine both **And** and **Or** operators, you may need to use an expression such as the ones supported by the [Calculated Element](#) transform.

Create the Calculated Element transform

In this example, we have dragged the *[Sales].[SalesPerson]* table from the Adventure Works database onto a new data cube.

Add the Calculated Element transform from Insert Common in the toolbar.



Configure the transform and click to add a calculated element.



Calculated Element

The transform allows for creation of new elements by applying user-defined expressions. The result of each expression is a new output element.

Name

Description

ID

1e0da516-1ee2-4acf-ae99-cdfe29cb93f5




☒ BusinessEntityID	int32
☒ TerritoryID	int32
☒ SalesQuota	Currency
☒ Bonus	Currency
☒ CommissionPct	Currency
☒ SalesYTD	Currency
☒ SalesLastYear	Currency
☒ rowguid	Guid
☒ ModifiedDate	DateTime

Calculated Elements



Define parameters

Edit output elements

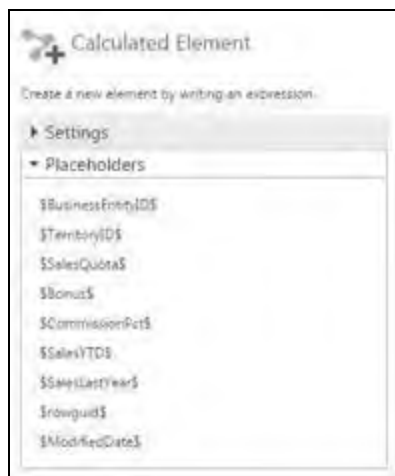
Set the **Name** and **Data** Type of the element. Our expression will return a data type of **Boolean**.

Enter an expression using [DundasScript](#) that includes your desired combination of filter conditions. Our example returns true for rows where *(COL_A > 5 AND COL_B > 5000) OR (COL_A = 1 AND COL_C > 0.01)*.

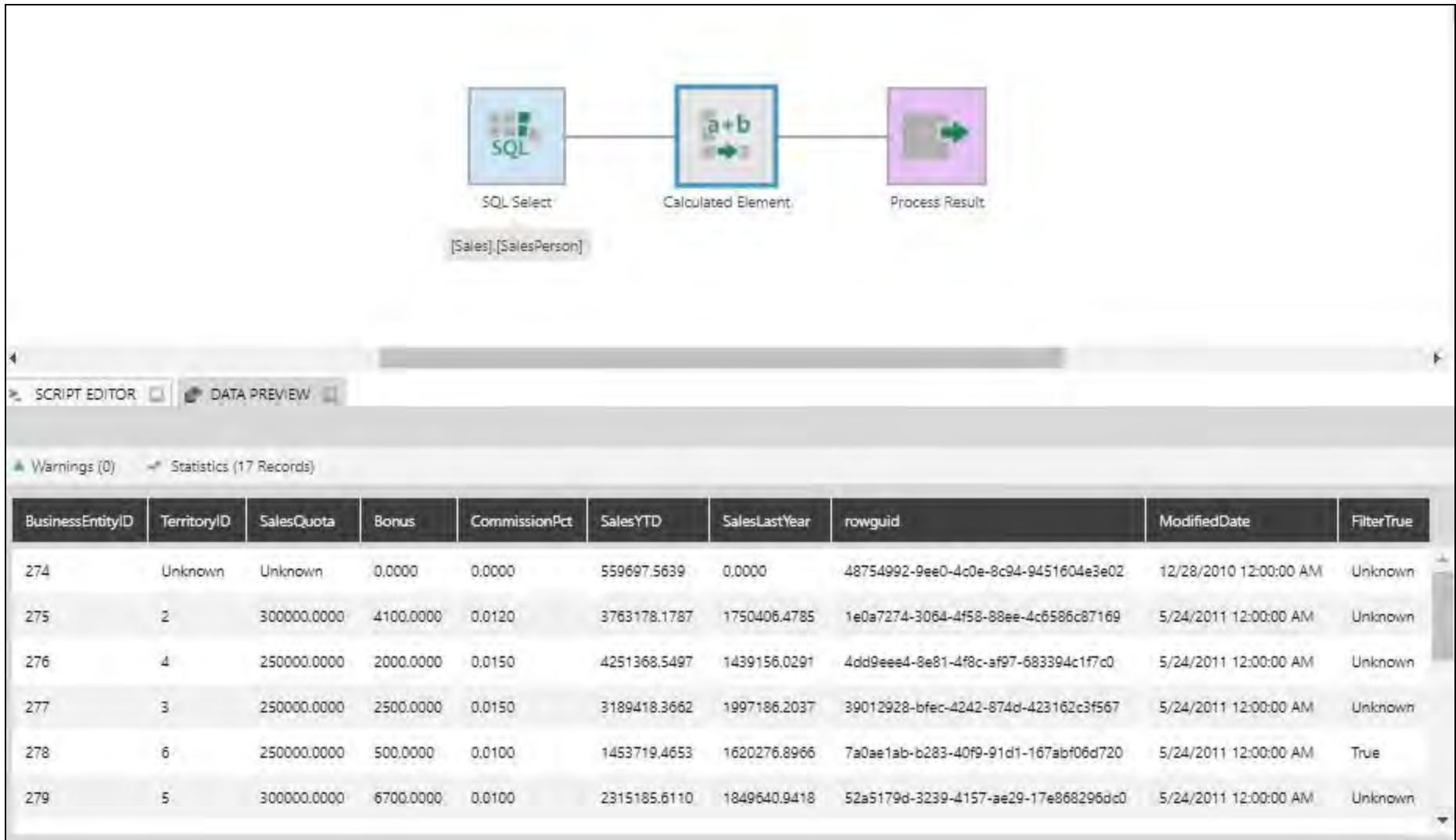
In our case, we will use the following script:

```
if (($TerritoryID$ > 5 && $SalesYTD$ > 5000) || ($TerritoryID$ == 1 && $CommissionPct$ > 0.01)){  
    return true;  
}
```

Type a dollar sign (\$) for a popup to suggest the placeholders that are available to refer to columns in your data, or click to expand the placeholders section for a list.



After submitting your changes, expand the Data Preview to see the results. They should include your new column.



The screenshot displays the Logi Symphony v24 interface. At the top, a workflow diagram shows three components connected in sequence: 'SQL Select' (blue box with 'SQL' text), 'Calculated Element' (blue box with 'a+b' text), and 'Process Result' (purple box with a green arrow). Below the 'SQL Select' component, the text '[Sales].[SalesPerson]' is visible. Below the workflow diagram, there are two tabs: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table with 10 columns and 6 rows of data. The columns are: BusinessEntityID, TerritoryID, SalesQuota, Bonus, CommissionPct, SalesYTD, SalesLastYear, rowguid, ModifiedDate, and FilterTrue. The rows contain numerical and string data, with the 'FilterTrue' column showing 'Unknown' for most rows and 'True' for one row (row 278).

BusinessEntityID	TerritoryID	SalesQuota	Bonus	CommissionPct	SalesYTD	SalesLastYear	rowguid	ModifiedDate	FilterTrue
274	Unknown	Unknown	0.0000	0.0000	559697.5639	0.0000	48754992-9ee0-4c0e-8c94-9451604e3e02	12/28/2010 12:00:00 AM	Unknown
275	2	300000.0000	4100.0000	0.0120	3763178.1787	1750406.4785	1e0a7274-3064-4f58-88ee-4c6586c87169	5/24/2011 12:00:00 AM	Unknown
276	4	250000.0000	2000.0000	0.0150	4251368.5497	1439156.0291	4dd9eee4-8e81-4f8c-af97-683394c1f7c0	5/24/2011 12:00:00 AM	Unknown
277	3	250000.0000	2500.0000	0.0150	3189418.3662	1997186.2037	39012928-bfec-4242-874d-423162c3f567	5/24/2011 12:00:00 AM	Unknown
278	6	250000.0000	500.0000	0.0100	1453719.4653	1620276.8966	7a0ae1ab-b283-40f9-91d1-167abf06d720	5/24/2011 12:00:00 AM	True
279	5	300000.0000	6700.0000	0.0100	2315185.6110	1849640.9418	52a5179d-3239-4157-ae29-17e868296dc0	5/24/2011 12:00:00 AM	Unknown

Create the Filter Transform

Add a Filter transform after the calculated element and configure it.

In the Filter configuration dialog, click Edit next to the created Calculated Element.



Filter

A transform that filters out rows which do not meet criteria of settings.

Name

Description

ID




☒ BusinessEntityID
 
☒ TerritoryID
 
☒ SalesQuota
 
☒ Bonus
 
☒ CommissionPct
 
☒ SalesYTD
 
☒ SalesLastYear
 
☒ rowguid
 
☒ ModifiedDate
 
☒ FilterTrue
 

Operator

And Operator

Define parameters

Edit output elements

STORAGE

450 / 537







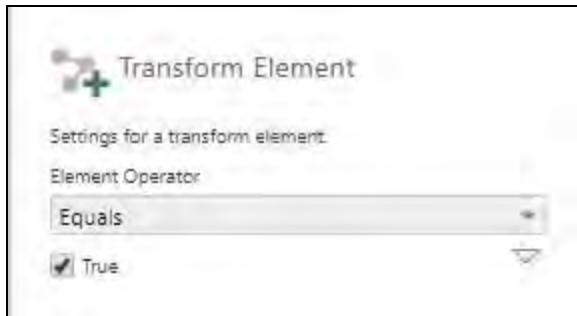




Calculated Element

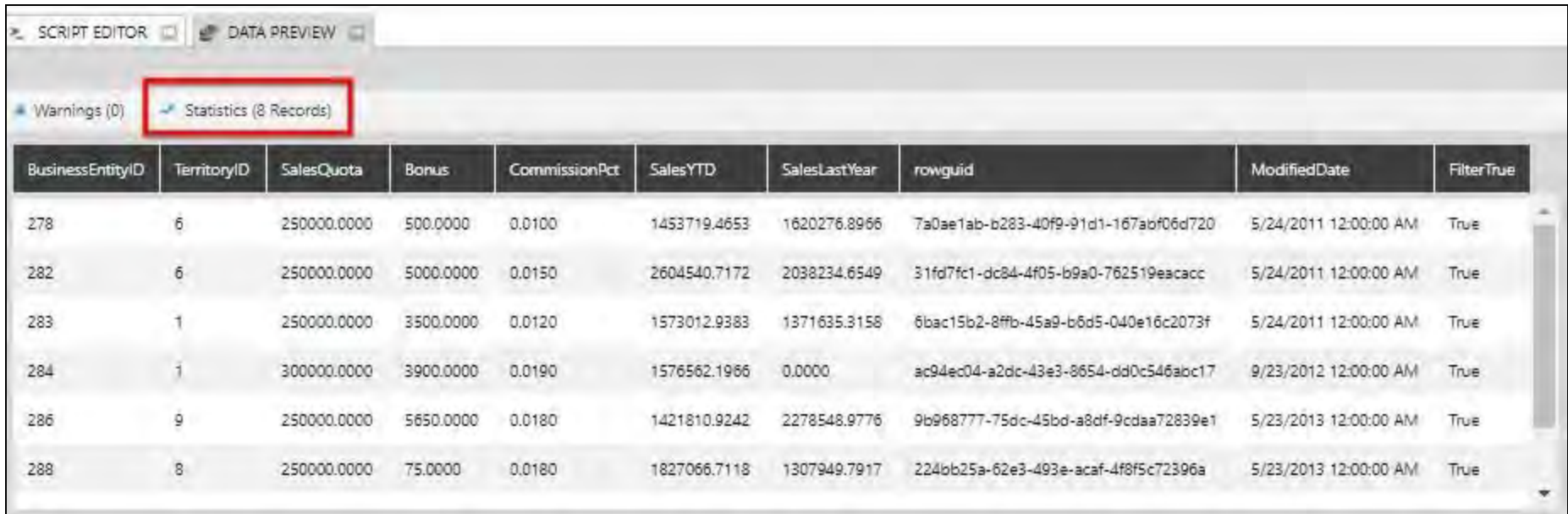
Filter

Project & Export



Submit the Transform Element dialog and uncheck the checkbox next to this calculated element to remove it from the filter transform's output (but still use it for filtering).

Submit the dialog and return to the Data Preview window to see the filtered results.



The image shows the 'DATA PREVIEW' window. At the top, there are tabs for 'SCRIPT EDITOR' and 'DATA PREVIEW'. Below the tabs, there is a section for 'Warnings (0)' and a section for 'Statistics (8 Records)' which is highlighted with a red box. Below this, there is a table with 10 columns: BusinessEntityID, TerritoryID, SalesQuota, Bonus, CommissionPct, SalesYTD, SalesLastYear, rowguid, ModifiedDate, and FilterTrue. The table contains 6 rows of data.

BusinessEntityID	TerritoryID	SalesQuota	Bonus	CommissionPct	SalesYTD	SalesLastYear	rowguid	ModifiedDate	FilterTrue
278	6	250000.0000	500.0000	0.0100	1453719.4653	1620276.8966	7a0ae1ab-b283-40f9-91d1-167abf06d720	5/24/2011 12:00:00 AM	True
282	6	250000.0000	5000.0000	0.0150	2604540.7172	2038234.6549	31fd7fc1-dc84-4f05-b9a0-762519eacacc	5/24/2011 12:00:00 AM	True
283	1	250000.0000	3500.0000	0.0120	1573012.9383	1371635.3158	6bac15b2-8ffb-45a9-b6d5-040e16c2073f	5/24/2011 12:00:00 AM	True
284	1	300000.0000	3900.0000	0.0190	1576562.1966	0.0000	ac94ec04-a2dc-43e3-8654-dd0c546abc17	9/23/2012 12:00:00 AM	True
286	9	250000.0000	5650.0000	0.0180	1421810.9242	2278548.9776	9b968777-75dc-45bd-a8df-9cdaa72839e1	5/23/2013 12:00:00 AM	True
288	8	250000.0000	75.0000	0.0180	1827066.7118	1307949.7917	2240b25a-62e3-493e-acaf-4f85c72396a	5/23/2013 12:00:00 AM	True

For more information, see:

- [Filter](#)
- [Calculated Element](#)



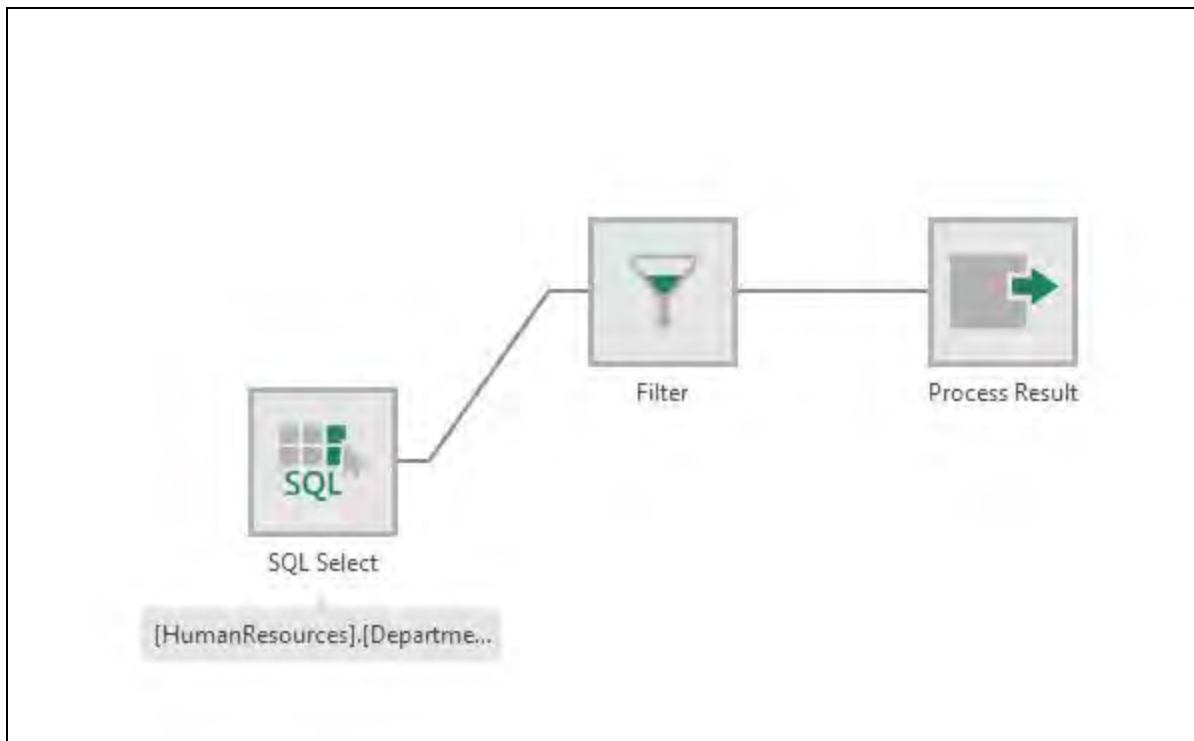
- Archive of documentation for Logi Symphony v24

- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Display 'top N + other' Measures Using Data Cube Transforms](#)

Filter

This applies to: *Managed Dashboards, Managed Reports*

The **Filter** transform filters out rows that do not meet the configured criteria/settings.



There are a variety of filter operators available when configuring the transform's settings, depending on the value type. Some examples:

- Equals
- Does Not Equal
- Not Equals Value Or Is Null
- Less Than

- Greater Than Or Equal
- Is A Number
- Is Not A Number
- Starts With
- Ends With
- Equals Any
- Contains
- Does Not Contain
- Contains All
- Contains Any
- Matches Regular Expression
- (And more, depending on the value type)

For some operators, you can also choose the Invert Selection option for your filter value to reverse the operator above.



Note: [Many select transforms](#) also support parameters you can configure to filter their output elements, which in some cases can improve query performance.

Input

The Filter transform requires one input transform that has at least one column.

For example, this could be a [SQL Select](#), or the result of another transform such as the [Aggregate](#) transform, as seen below:



Configure

The filter transform can filter either according to specific filter settings, or through parameters.

Filter Settings

Uncheck any columns you don't want to be included in the output. These can still be used to filter the records. Click the **Edit** icon for a column to filter by its values.



Filter

more help

A transform that filters out rows which do not meet criteria of settings.

Name

Description

ID

SETTINGS




☒ OrderQty
 

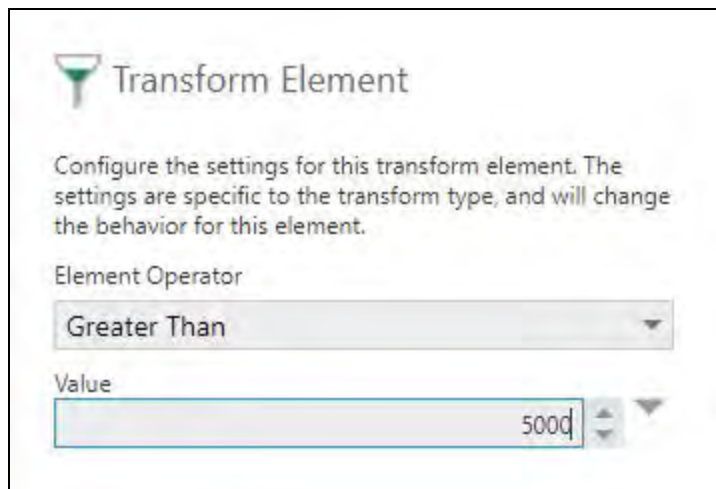
☒ ProductID
 

Operator

In the Transform Element dialog, select the Element Operator and fill in the fields that follow (e.g., Value) or use their [token menu](#).

to set their value using options including [attributes](#) and [custom attributes](#).

After entering a Value, you can choose the Invert Selection option from the menu on the right (if applicable) when you want to filter by the reverse of whatever is selected as the operator and value.



Transform Element

Configure the settings for this transform element. The settings are specific to the transform type, and will change the behavior for this element.

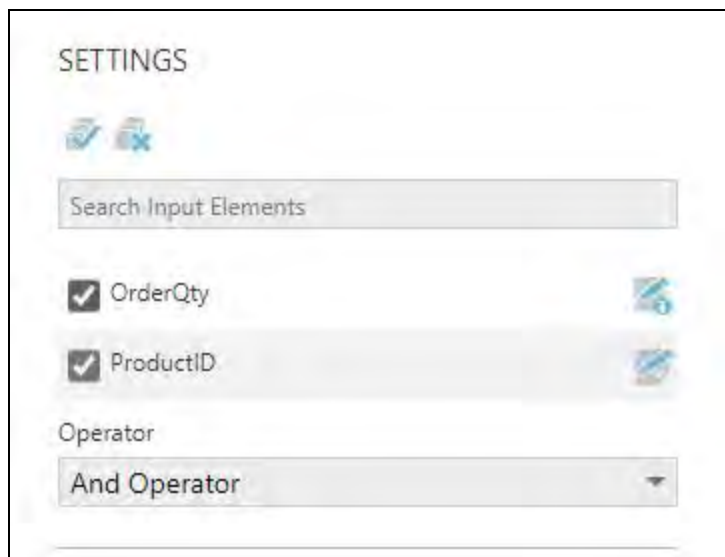
Element Operator

Greater Than

Value

5000

Click the **Submit** button at the bottom of the dialog. The icon will now indicate its filtering with a distinct icon.



SETTINGS

Search Input Elements

☒ OrderQty 

☒ ProductID 

Operator

And Operator

When filtering by more than one column's values, use the **Operator** setting in the **Filter** dialog to determine whether all of the conditions must be met (**And Operator**) or any of the conditions can be met (**Or Operator**) for records to be included.

Parameters

You can set up an output element filter parameter for when you want to connect the filtering to a [view parameters](#) or [filter](#) on a dashboard/view, or use it with a [bridge parameter](#).

Click **Parameters**, then **Add parameter**.



In the *Configure Transform Parameter* dialog, set **Parameter Type** to *Output Element Filter*.

Set **Connector Element** to the data you want to filter by.

Transform Parameters

Create, edit and delete the parameters on this transform. Parameters can be created on various settings, such as transform settings for this transform and output element filters.

Add parameter

Configure Transform Parameter

Set up this transform parameter by specifying what it is linked to when creating it, and then the name, value and type of the parameter value after it has been created. If a parameter is marked as public, then it will be accessible outside of this data cube.

Parameter Type

Output Element Filter

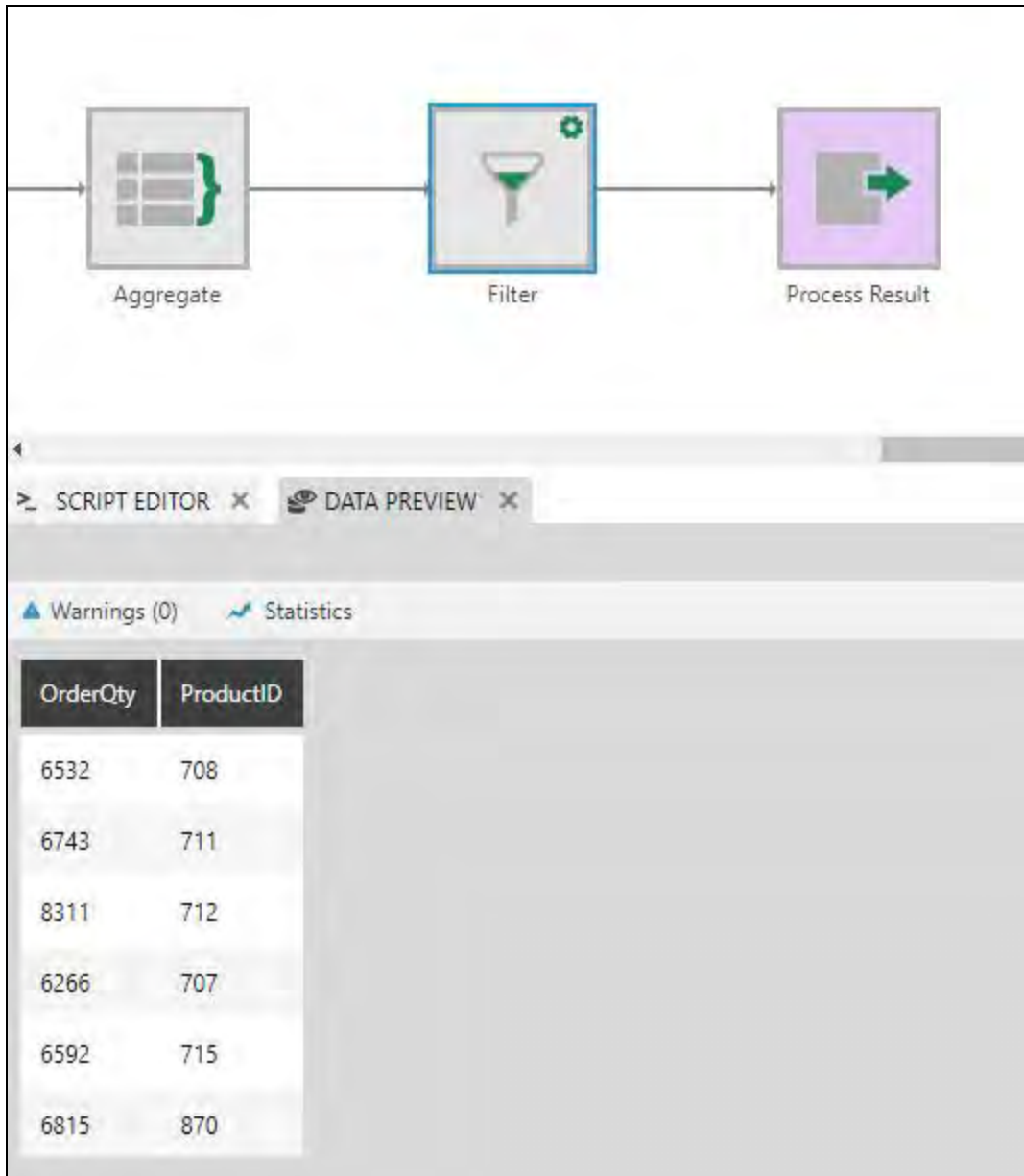
Connector Element

ProductID (Int32)

Click the Submit button at the bottom to continue configuring the parameter. This process is the same as defining parameters on a [select transform](#).

Output

The figure below illustrates the output for our filter settings example. Only rows with OrderQty greater than 5000 are included.



The screenshot displays the Logi Symphony v24 interface. At the top, a workflow diagram shows three steps: 'Aggregate' (represented by a grid icon), 'Filter' (represented by a funnel icon), and 'Process Result' (represented by a square with a right-pointing arrow). Below the workflow, the interface is divided into two main sections: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' section is active, showing a table with two columns: 'OrderQty' and 'ProductID'. The table contains six rows of data. Above the table, there are tabs for 'Warnings (0)' and 'Statistics'.

OrderQty	ProductID
6532	708
6743	711
8311	712
6266	707
6592	715
6815	870

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Managing Tokens](#)
- [Using Custom Attributes to Filter Data by User](#)
- [The Managed Dashboards Data Model](#)

Flatten JSON

This applies to: *Managed Dashboards, Managed Reports*

The Flatten JSON transform turns rows of data containing JSON text into separate columns for each of its values. JSON is a common result of using the [Generic Web data provider](#) to connect to online data sources, for example.

Input

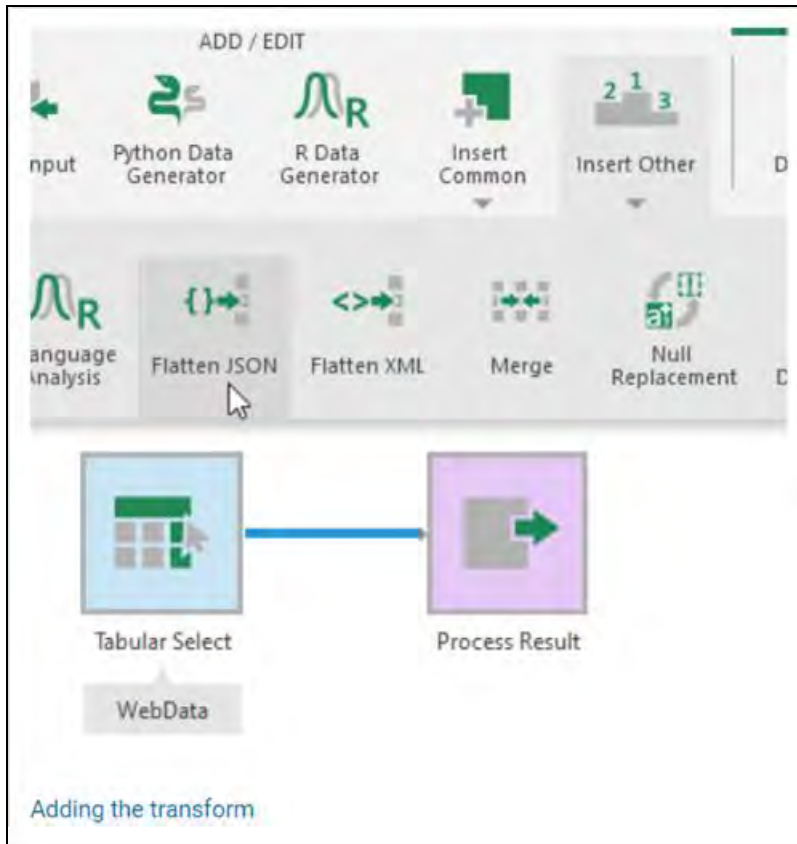
The Flatten JSON transform requires one column containing JSON text.

There might be only one row containing all of the JSON, or it could be split into multiple rows.

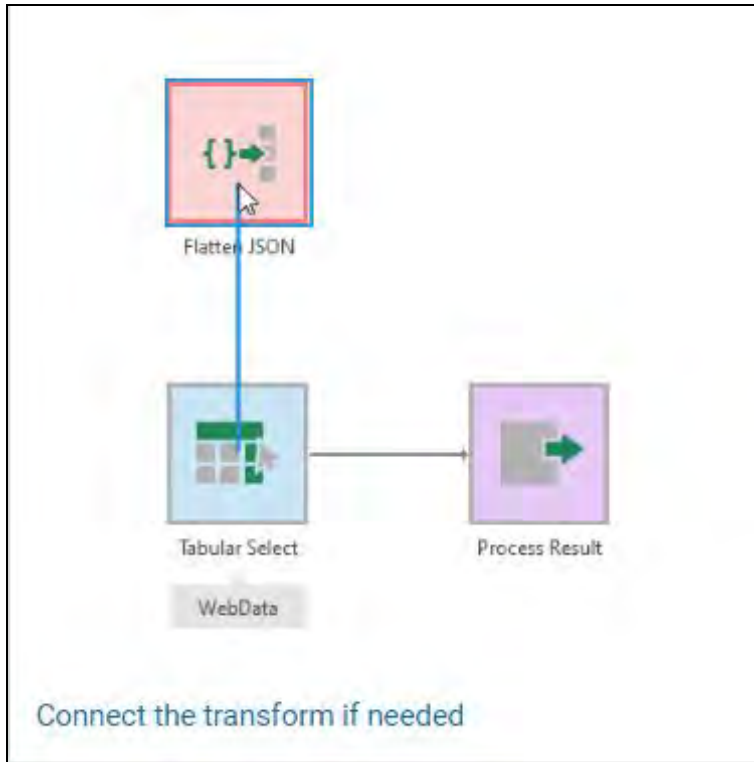


Add the Transform

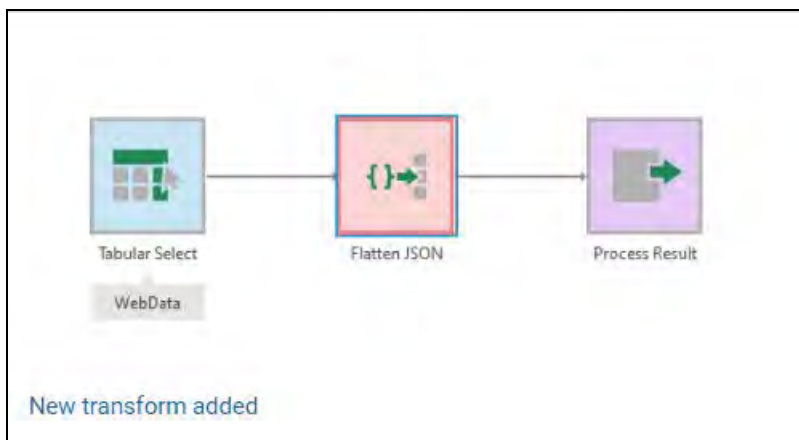
Optionally, click to select an existing connection between two of your transforms where you want to insert one. In the toolbar, choose **Insert Other**, then **Flatten JSON**.



If you didn't select an insertion point earlier, click and drag from the transform you want to use as input onto the new transform to connect it.

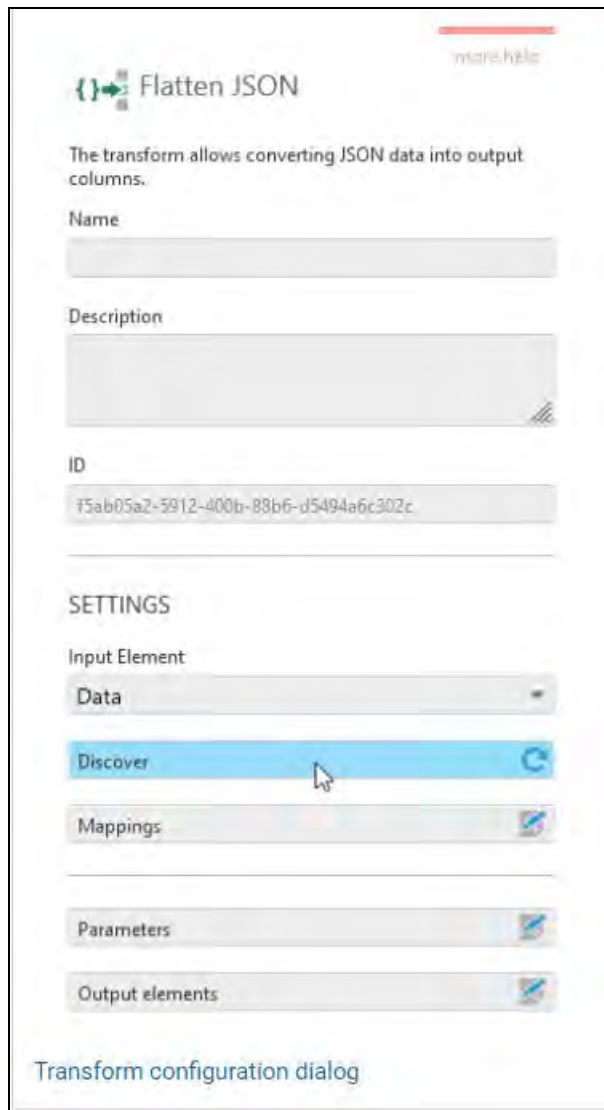


The transform initially appears red because it requires configuration.



Configure

Open the configuration dialog by clicking or right-clicking the new transform and choosing **Configure** in the toolbar or menu (or double-clicking the transform).



Flatten JSON

The transform allows converting JSON data into output columns.

Name

Description

ID

f5ab05a2-5912-400b-88b6-d5494a6c302c

SETTINGS

Input Element

Data

Discover

Mappings

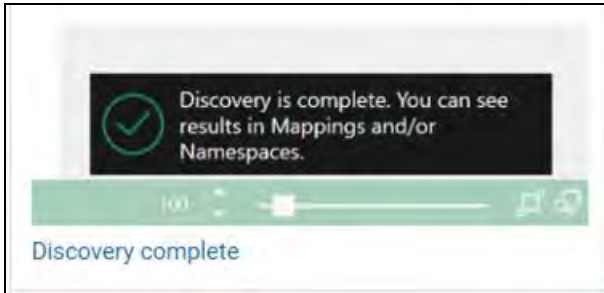
Parameters

Output elements

Transform configuration dialog

If your input has more than one column, set the **Input Element** drop-down to the one containing JSON text.

Next, click **Discover** to automatically populate mappings defining the structure and data types of your columns based on the current input JSON. A notification will appear in the lower right corner of the screen when complete, often right away.

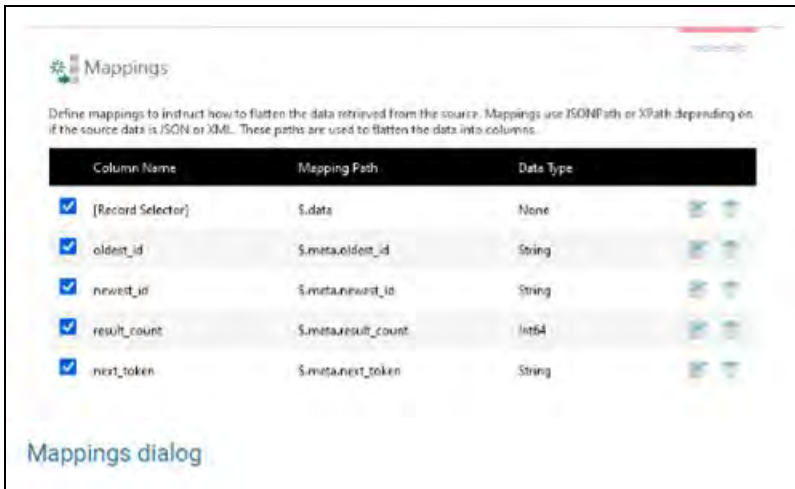


Note: Only one property containing an array of multiple objects is included in the output at a time per parent object. You can edit these record selector mappings as shown below to choose which one to include.

Editing Mappings

Click **Mappings** to confirm or change the discovered mappings, although this is typically not required.

The initial *Mappings* dialog shows the properties for the top level of discovered object(s) and their data types, which will be transformed into columns.

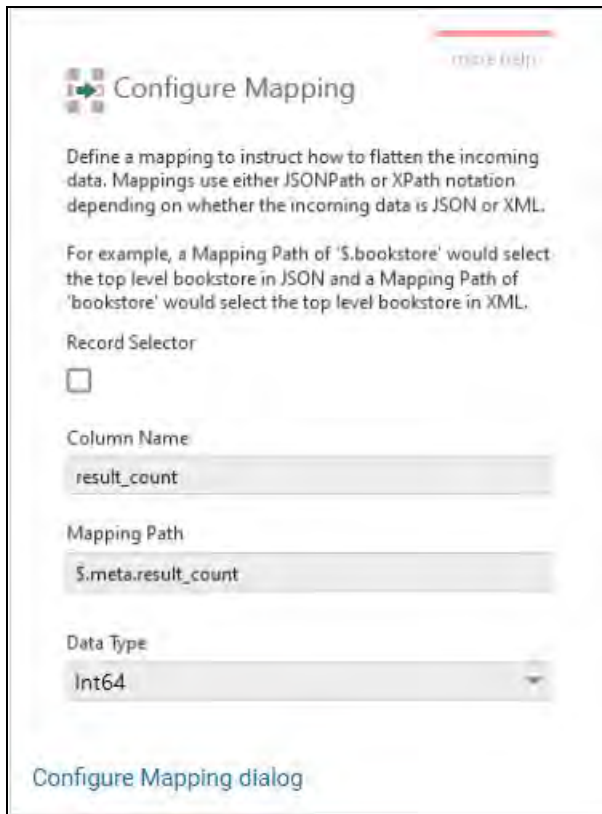


Note: If no mappings were discovered, check that the Input Element drop-down was set correctly, or open the Data Preview with the preceding transform selected to confirm the input is JSON text that can be converted into columns.

You can exclude a mapping from the transform output by unchecking its checkbox along the left, or delete it using the icon along the right.

A mapping with *[Record Selector]* as its column name is describing a property containing multiple child objects that will be turned into records (rows of data) and then combined/joined with the parent record. Only one of these can be enabled at the same level of mappings at a time to prevent excessive duplication of the parent record's values.

Click the **Edit** icon for a mapping to open its details.



Configure Mapping [more help](#)

Define a mapping to instruct how to flatten the incoming data. Mappings use either JSONPath or XPath notation depending on whether the incoming data is JSON or XML.

For example, a Mapping Path of '\$.bookstore' would select the top level bookstore in JSON and a Mapping Path of 'bookstore' would select the top level bookstore in XML.

Record Selector
☐

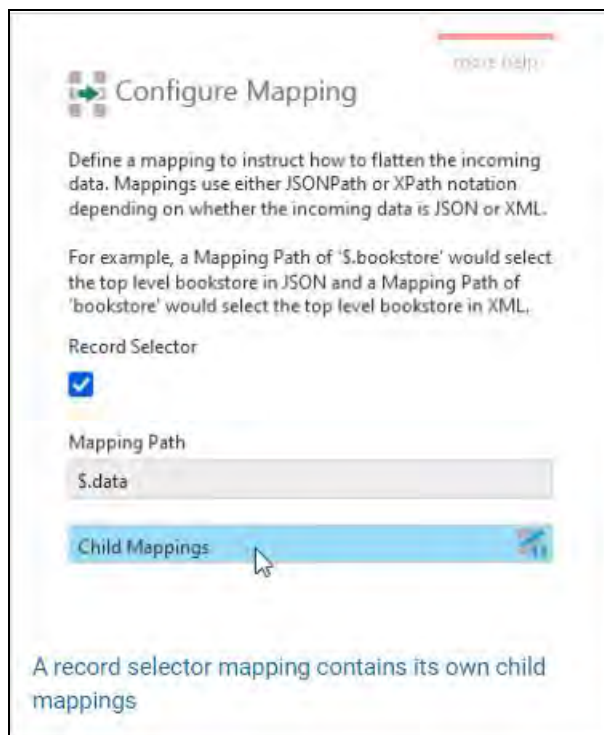
Column Name

Mapping Path

Data Type

[Configure Mapping dialog](#)

The Mapping Path is set using JSONPath syntax to refer to a property in the JSON text. For simple property values, you can change the Data Type from what was automatically discovered to another type that will be compatible with that property's values.

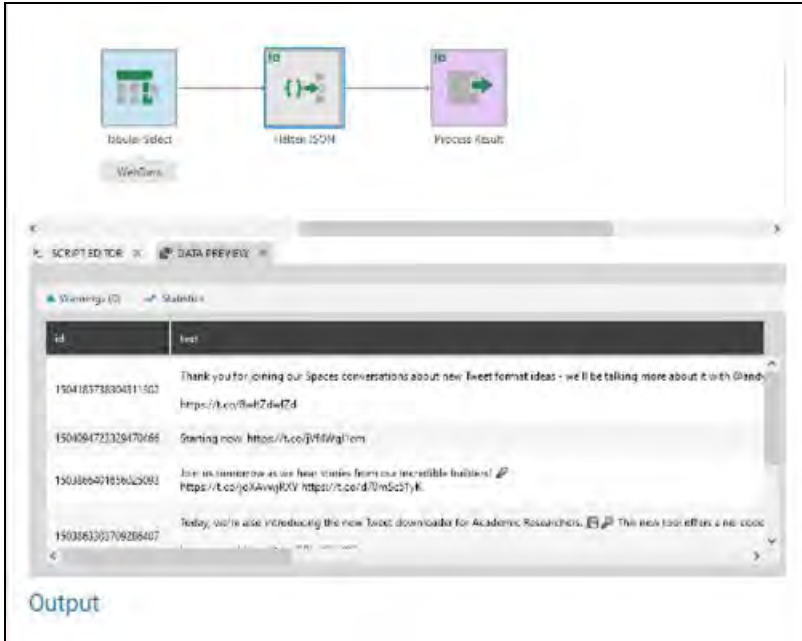


A selected Record Selector option indicates that the value of this property in JSON is an array containing child objects that will be transformed into multiple records.

Click on **Child Mappings** for this type of mapping to open another *Mappings* dialog like the first one above, this time for the child object's properties.

Output

After submitting the configuration dialogs, you can preview the output by opening the **Data Preview** window with the transform selected.



The screenshot displays the Logi Symphony v24 interface. At the top, a data pipeline is visible with three components: 'Table Select', 'Filter SQL', and 'Process Result', connected by arrows. Below the pipeline, a 'WebCore' button is present. The main area shows a 'DATA PREVIEW' window with a table of data. The table has two columns: 'id' and 'text'. The data rows are as follows:

id	text
150418378804311503	Thank you for joining our Spaces conversations about new tweet format ideas - we'll be talking more about it with @andrew...
150406472329470666	Starting now: https://t.co/9Vf6wgl1em
1503866491616425093	Join us tomorrow as we hear stories from our incredible builders! https://t.co/pXAvyRXY https://t.co/d70mSc5YkI
1503863303708286407	Today, we're also introducing the new Tweet downloader for Academic Researchers. This new tool offers a full suite...

Below the data preview, there is an 'Output' section.

For more information, see:

- [Using Transforms](#)
- [Flatten XML](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Flatten XML

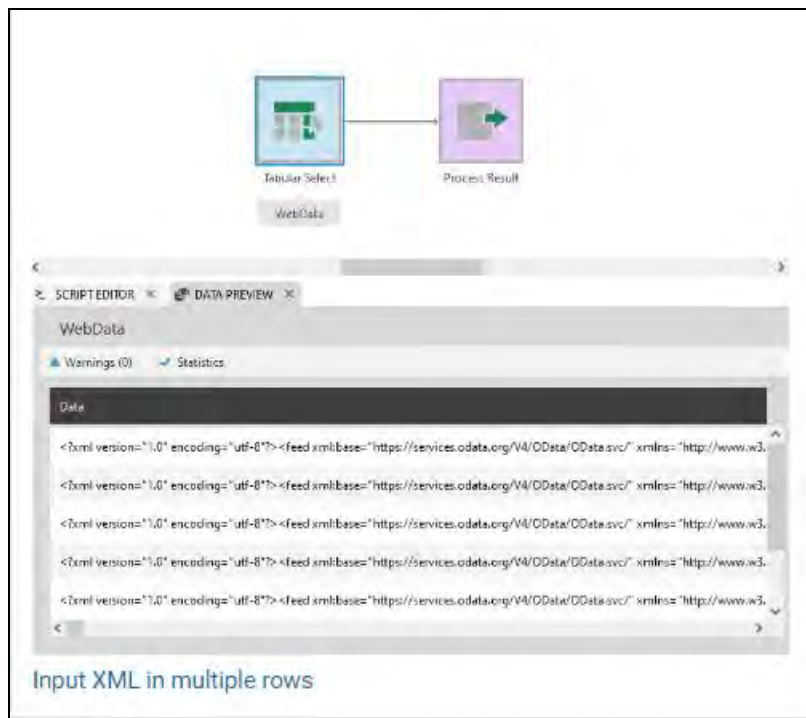
This applies to: *Managed Dashboards, Managed Reports*

The Flatten XML transform turns rows of data containing XML text into separate columns for each of its values. XML is a common result of using the [Generic Web data provider](#) to connect to online data sources, for example.

Input

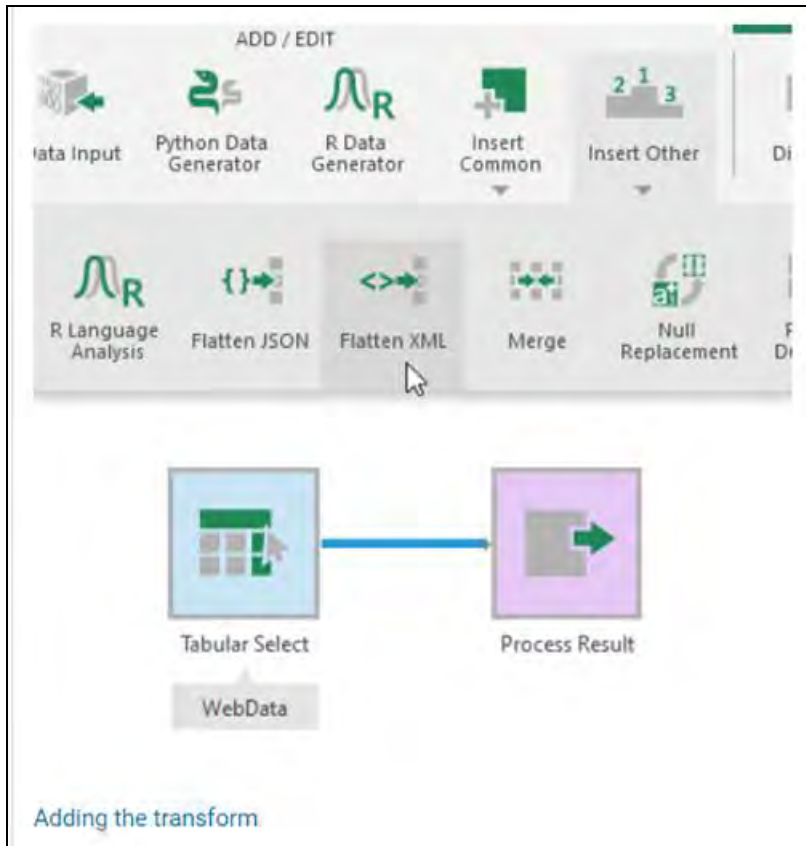
The Flatten XML transform requires one column containing XML text.

There might be only one row containing all of the XML, or it could be split into multiple rows.

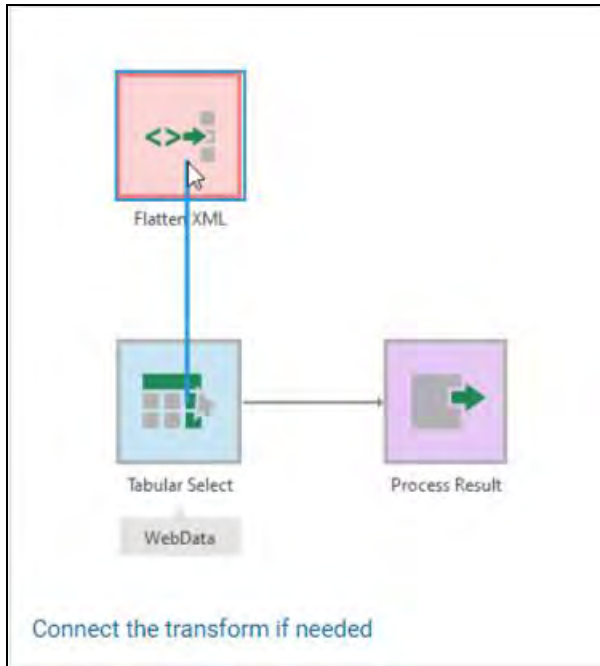


Add the Transform

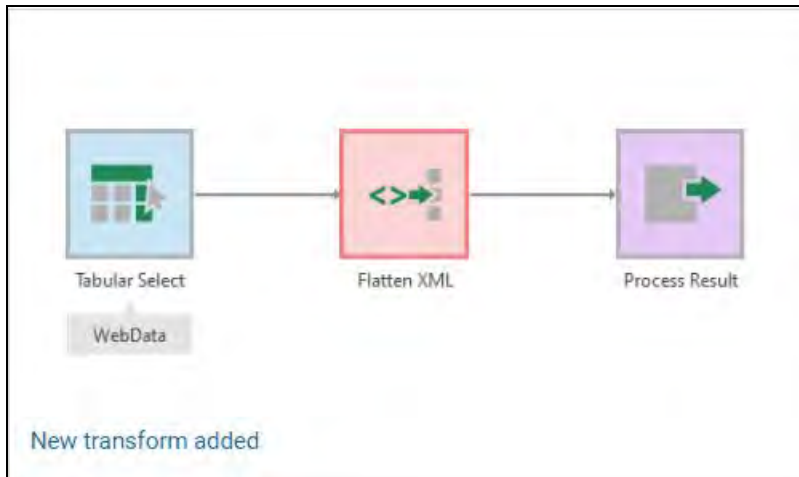
Optionally, click to select an existing connection between two of your transforms where you want to insert one. In the toolbar, choose **Insert Other**, then **Flatten XML**.



If you didn't select an insertion point earlier, click and drag from the transform you want to use as input onto the new transform to connect it.



The transform initially appears red because it requires configuration.



Configure

Open the configuration dialog by clicking or right-clicking the new transform and choosing Configure in the toolbar or menu (or double-clicking the transform).


Flatten XML
more help

The transform allows converting XML data into output columns.

Name

Description

ID

SETTINGS

Input Element

Data

Discover

Mappings

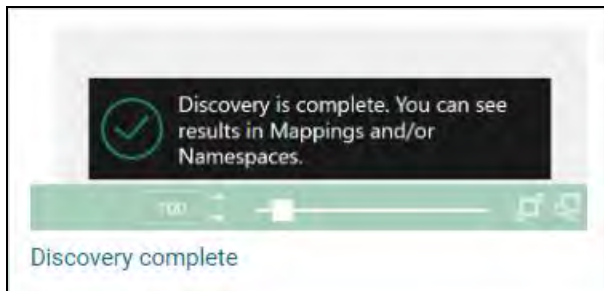
NAMESPACES

Add Namespace

Transform configuration dialog

If your input has more than one column, set the **Input Element** drop-down to the one containing XML text.

Next, click **Discover** to automatically populate mappings and other details defining the structure and data types of your columns based on the current input XML. A notification will appear in the lower right corner of the screen when complete, often right away.



Note: Only one type of child element can be included in the output at a time per parent element. You can edit these record selector mappings as shown below to choose which one to include.

If the XML contains prefixes (e.g., <prefix:element>...), these may have been populated under the Namespaces heading along with their corresponding URLs in order to parse the XML correctly.

Editing Mappings

Click **Mappings** to confirm or change the discovered mappings, although this is typically not required.

The initial *Mappings* dialog shows the top level of the discovered elements in your XML.





Note: If no mappings were discovered, check that the Input Element drop-down was set correctly, or open the Data Preview with the preceding transform selected to confirm the input is XML that can be converted into columns.

Your top-level element might only contain child or sub-elements like the example above rather than directly containing data.

Click the **Edit** icon for a mapping to open its details.



Configure Mapping

Define a mapping to instruct how to flatten the incoming data. Mappings use either JSONPath or XPath notation depending on whether the incoming data is JSON or XML.

For example, a Mapping Path of '\$.bookstore' would select the top level bookstore in JSON and a Mapping Path of 'bookstore' would select the top level bookstore in XML.

Record Selector
☒

Mapping Path

Child Mappings

A record selector mapping contains its own child mappings

The **Mapping Path** is set using XPath syntax to refer to an element or attribute value in the XML.

A selected **Record Selector** option indicates that this mapping represents multiple child elements that will each be transformed into a record.

Click on **Child Mappings** for this type of mapping to open another *Mappings* dialog like the first one above, this time for the child element.



Mappings

Define mappings to instruct how to flatten the data retrieved from the source. Mappings use JSONPath or XPath depending on if the source data is JSON or XML. These paths are used to flatten the data into columns.

Column Name	Mapping Path	Data Type		
☒ xml:base	@xml:base	String		
☒ mc:context	@mc:context	String		
☒ id	id	String		
☒ title	title	String		
☒ type	@type	String		
☒ updated	updated	DateTime		
☒ link	link	String		
☒ rel	@rel	String		
☒ title	@title	String		
☒ href	@href	String		
☒ [Record Selector]	entry	None		

Editing mappings containing data

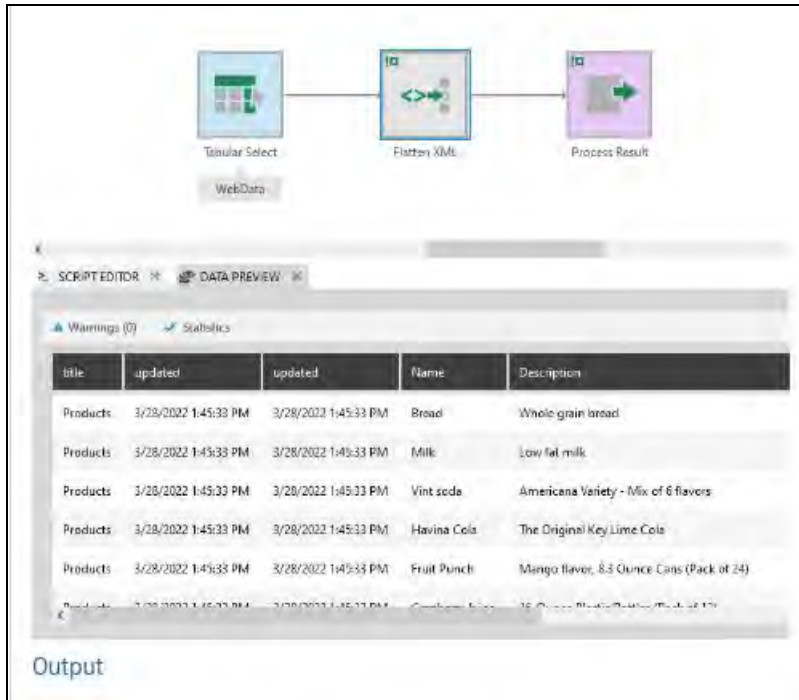
Mappings representing data that will be turned into columns will have a type listed in the *Data Type* column. You can edit these mappings if you want to change the data type to something else that will be compatible with that property's values.

A mapping with *[Record Selector]* as its column name indicates there can be multiple child elements of that type within each of the current parent element type, so the child elements will be turned into multiple records (rows of data) and then combined/joined with the parent record. Only one of these can be enabled at each level of mappings at a time to prevent excessive duplication of the parent record's values.

You can exclude a mapping from the transform output by unchecking its checkbox along the left, or delete it using the icon along the right.

Output

After submitting the configuration dialogs, you can preview the output by opening the **Data Preview** window with the transform selected.



You can configure other [standard transform options](#) such as parameters and output elements by configuring the transform again.

For more information, see:

- [Using Transforms](#)
- [Flatten JSON](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Override Config Settings Using an XML File](#)
- [Connecting to XML Files](#)
- [Connecting to Generic Web Data Sources](#)

Fusing

This applies to: *Managed Dashboards, Managed Reports*

The Fusing transform combines the rows and columns of two tables together by appending them.

Its behavior is a combination of different aspects of the **Union** and **Join** transforms. Rows are appended like a Union without requiring the columns of the two tables to match. It can combine different columns from two tables like a Join, but without matching and combining rows by key values.

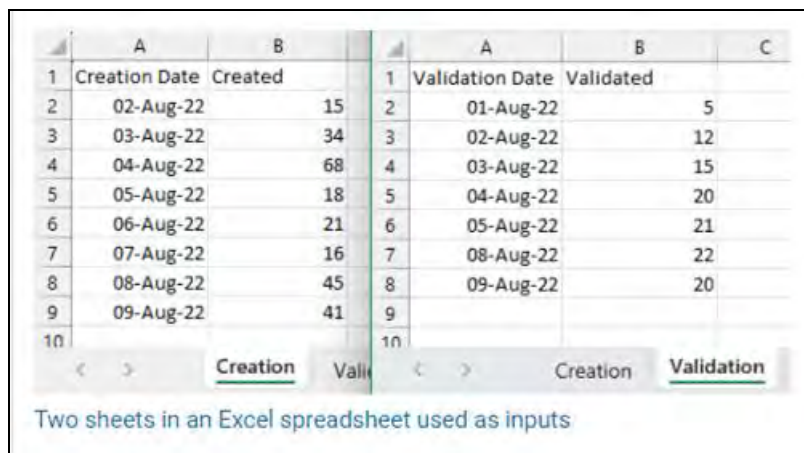


Note: When [automatically joining data](#), the data will actually be combined using fusing if the matched columns are *DateTime* type.

Input

Choose the **Fusing** transform from the toolbar, which requires two inputs.

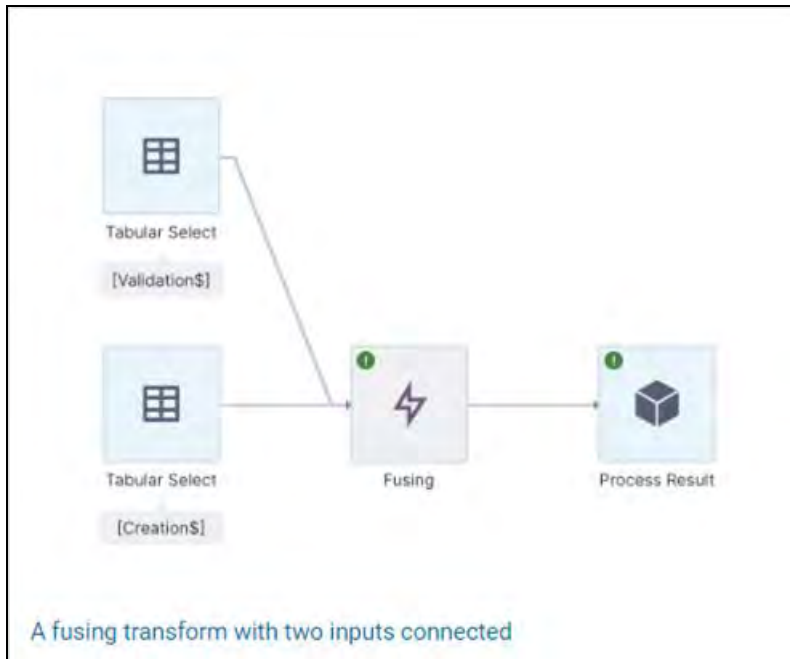
As an example, the following two sheets from Excel have been dragged onto the data cube canvas. They each contain numeric counts tracked by dates or timestamps, where it is common and expected for them to not match:



	A	B		A	B	C
1	Creation Date	Created	1	Validation Date	Validated	
2	02-Aug-22	15	2	01-Aug-22	5	
3	03-Aug-22	34	3	02-Aug-22	12	
4	04-Aug-22	68	4	03-Aug-22	15	
5	05-Aug-22	18	5	04-Aug-22	20	
6	06-Aug-22	21	6	05-Aug-22	21	
7	07-Aug-22	16	7	08-Aug-22	22	
8	08-Aug-22	45	8	09-Aug-22	20	
9	09-Aug-22	41	9			
10			10			

Two sheets in an Excel spreadsheet used as inputs

Click and drag from your inputs to the fusing transform to connect them. You can also click to select an existing connection from an input before adding the transform to insert it there.



Configure

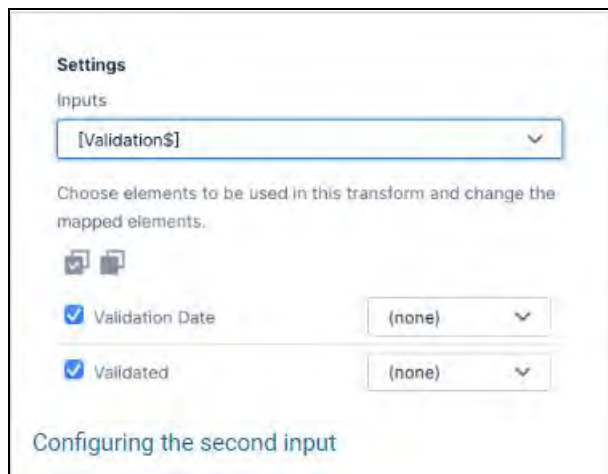
Click or right-click the transform and choose **Configure** in the toolbar or menu, or double-click as a shortcut.

Under **Settings**, uncheck any elements (columns) from the first input that you do not want included in the output.



The screenshot shows the 'Settings' dialog for the 'Fusing' transform. Under the 'Inputs' section, the first input is selected, showing '[Creation\$]' in a dropdown menu. Below this, there is a section titled 'Choose elements to be used in this transform:' which contains two checkboxes. The first checkbox is checked and labeled 'Creation Date' with a 'DateTime' data type. The second checkbox is also checked and labeled 'Created' with a 'Double' data type. At the bottom of the dialog, the text 'Configuring the first input' is displayed.

Choose the second input from the **Inputs** drop-down, then update the checkboxes the same way for its elements.



Settings

Inputs

[Validation\$] ▼

Choose elements to be used in this transform and change the mapped elements.

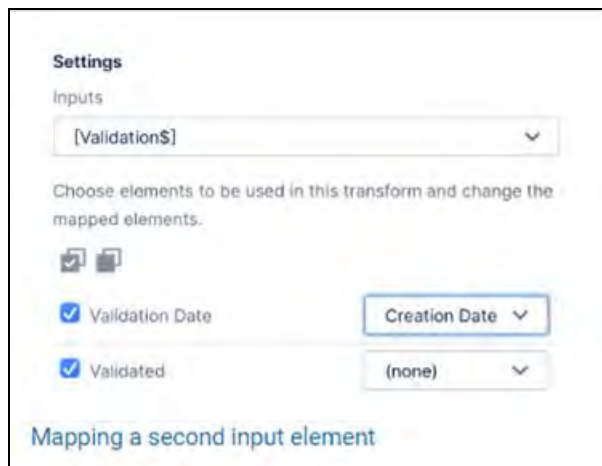
☒ Validation Date (none) ▼

☒ Validated (none) ▼

[Configuring the second input](#)

The drop-downs next to each input element allow you to choose whether they should be mapped with any of the elements from the first input, which will combine them into a single column in the output. Some of these may have been set for you automatically based on matching names. Any element that you want to remain a separate column can be set to (none).

For our example data, if we want all the dates to be combined into a single column, we can map the **Validation Date** column to the **Creation Date** column.



Settings

Inputs

[Validation\$] ▼

Choose elements to be used in this transform and change the mapped elements.

☒ Validation Date Creation Date ▼

☒ Validated (none) ▼

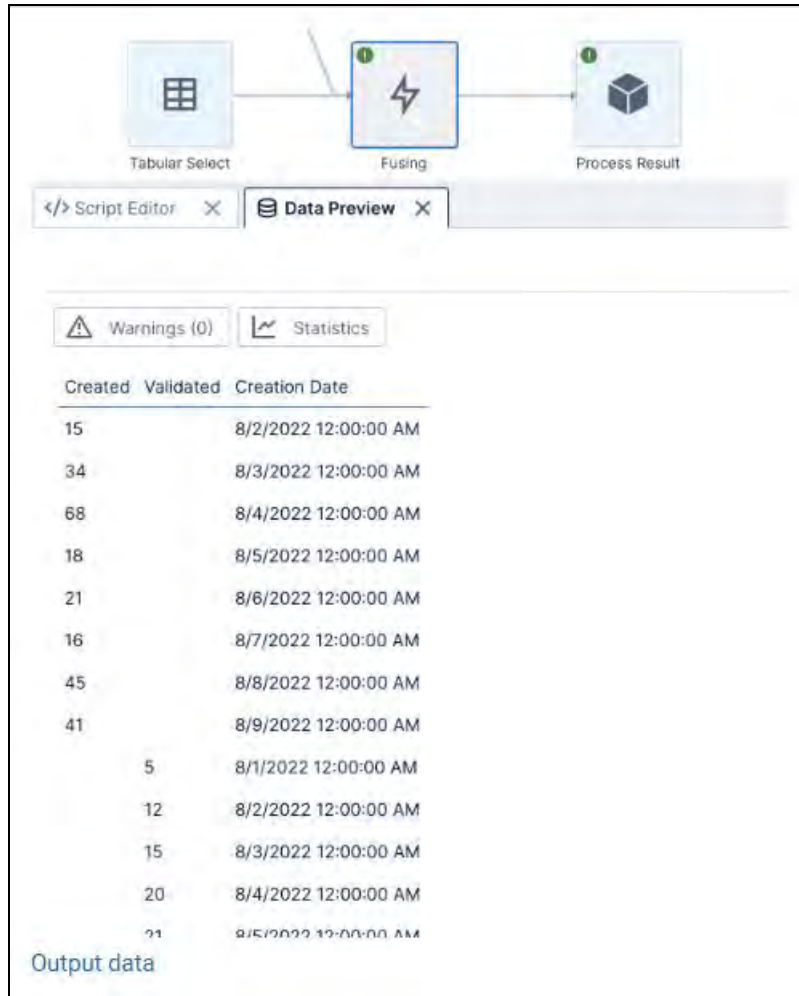
[Mapping a second input element](#)



Note: Submit the configuration dialog and then reopen it if you want changes to be reflected when editing output elements, for example.

Output

After submitting the configuration dialog, you can preview the output by opening the Data Preview window with the transform selected.



The screenshot shows the Logi Symphony interface with a workflow consisting of three transforms: Tabular Select, Fusing, and Process Result. The Fusing transform is selected, and the Data Preview window is open, displaying a table of output data.

Created	Validated	Creation Date
15		8/2/2022 12:00:00 AM
34		8/3/2022 12:00:00 AM
68		8/4/2022 12:00:00 AM
18		8/5/2022 12:00:00 AM
21		8/6/2022 12:00:00 AM
16		8/7/2022 12:00:00 AM
45		8/8/2022 12:00:00 AM
41		8/9/2022 12:00:00 AM
	5	8/1/2022 12:00:00 AM
	12	8/2/2022 12:00:00 AM
	15	8/3/2022 12:00:00 AM
	20	8/4/2022 12:00:00 AM
	21	8/5/2022 12:00:00 AM

Output data

In the example above, the rows from the two inputs have been appended, the two numeric columns have been appended, and the two date columns have been combined. We could re-open the configuration dialog, click Output elements, and edit the date column's name to simply Date.

Data is commonly grouped and aggregated once used in a metric set in Symphony, which can combine the appended rows. For example, clicking **Add To New Metric Set** in the toolbar and adding all three elements automatically visualizes the data like the following:



The Fusing transform accomplishes this result when it could be more difficult with alternatives:

- A **Union** transform (like a UNION in SQL) matches and combines every column from the first input with a column from the second, and could not output a **Created** column separately from the **Validated** column.
- A **Join** transform (like a JOIN in SQL) is based on matching key values, but our non-numeric data consists of only dates that do not match and are not expected to. A full (outer) join could return all of the rows from each input, but beware of duplicated key values: if there were any duplicated dates in one of our example inputs, the matched row of data in the other input would be duplicated including its numeric values.

For more information, see:

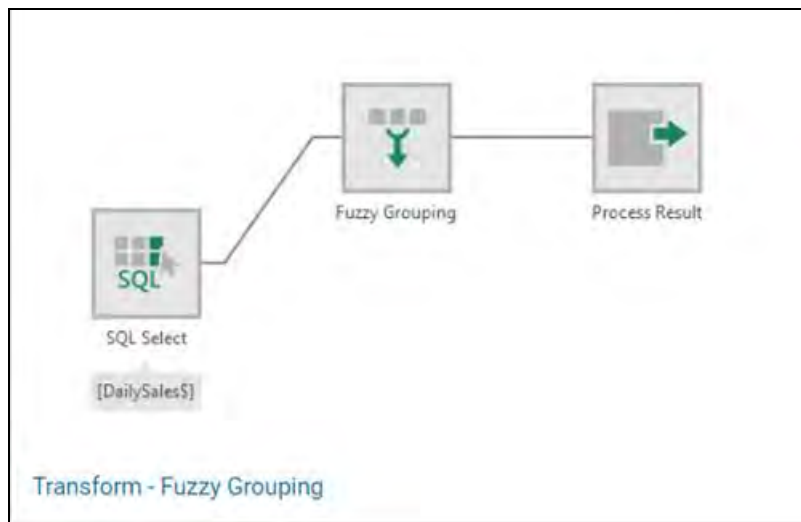
- [Automatic Joins and Hierarchies](#)
- [Using Transforms](#)
- [Union](#)
- [Join](#)
- [Create a New Data Cube](#)

Fuzzy Grouping

This applies to: *Managed Dashboards, Managed Reports*

The Fuzzy Grouping transform allows grouping of records by looking at the similarity between the values of various columns.

Two records in which a possible misspelling has occurred can be grouped together for further analysis, or found duplicates can be removed. The sensitivity toward differences between values can be adjusted.



Input

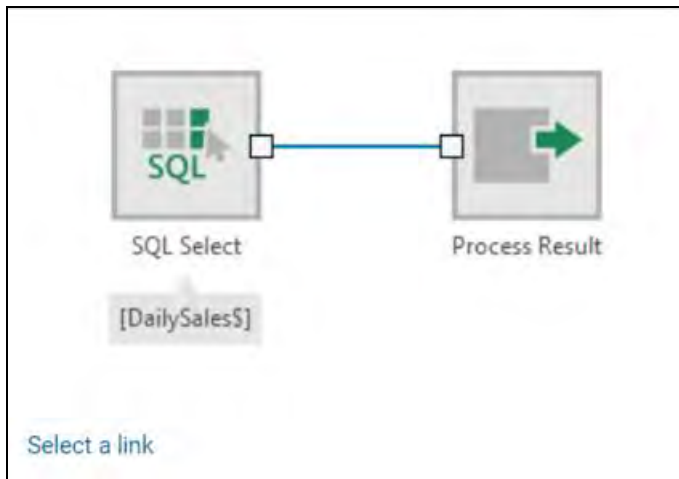
The Fuzzy Grouping transform requires one input transform that has at least one column. Consider the following input as an example:

Sales	Name	Surname
50000	John	Doe
80000	Kevin	Smith
60000	David	Jones
35000	Jon	Doe
25000	kevin	Smiht

Example data

Add the Transform

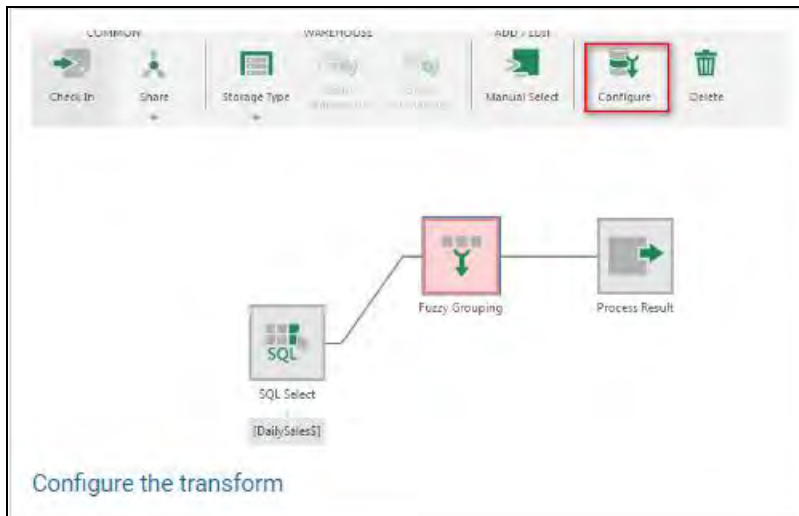
Click the connector link between two transforms to select it.



In the toolbar, choose **Insert Other**, then **Fuzzy Grouping**.

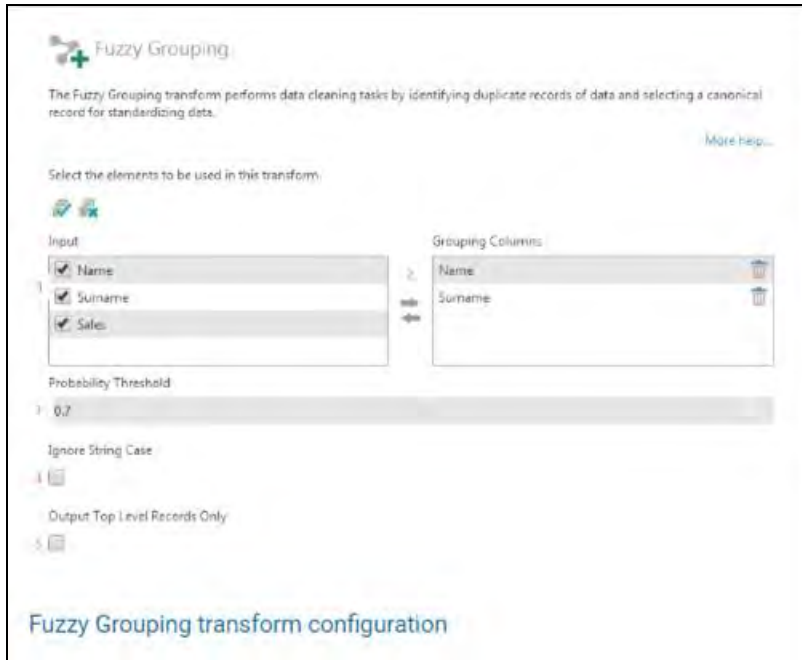


To edit/configure the transform, select it and choose **Configure** in the toolbar.



Configure

Steps to configure the Fuzzy Grouping transform:



1. Uncheck any columns that should be excluded from the output.
2. Drag and drop the columns that should be grouped from under **Input** to **Grouping Columns**.
3. Enter the **Probability Threshold**. Valid values are from .0001 to 1.0, where a value of 1 will require input data to be an exact match for them to be grouped together.
4. Select **Ignore String Case** if you want a non-case sensitive match.
5. Select **Output Top Level Records Only** if you want to omit the records found to be duplicates.

Output

The figure below illustrates the output for our example.

With **Output Top Level Records Only** unchecked:

Sales	Name	Surname
60000	David	Jones
50000	John	Doe
35000	Jon	Doe
25000	kevin	Smiht
80000	Kevin	Smith

All records output

With **Output Top Level Records Only** checked:

Sales	Name	Surname
60000	David	Jones
50000	John	Doe
25000	kevin	Smiht

Only top level records output

For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)



- Archive of documentation for Logi Symphony v24

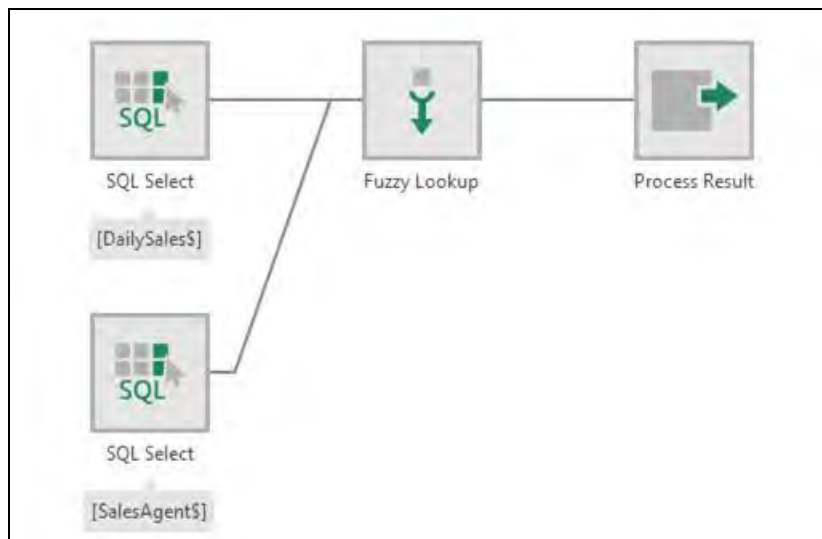
- [Data Cubes](#)

Fuzzy Lookup

This applies to: *Managed Dashboards, Managed Reports*

The Fuzzy Lookup transform joins the columns of two tables into one table by matching key values, where there may not be an exact match between the two tables.

This is similar to the [Lookup](#) and [Join](#) transforms.



Input

The Fuzzy Lookup transform requires two inputs.

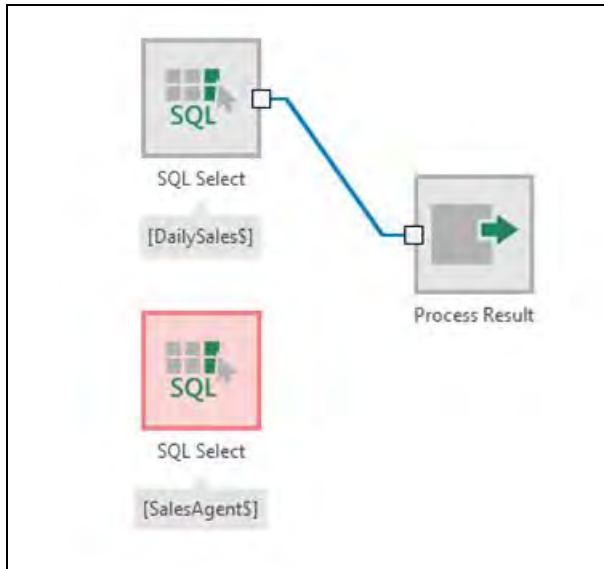
The following two tables will be used as an example:

Sales	Name	Surname	AgentID	Name	Surname
50000	John	Doe	1	Jane	Doe
80000	Kevin	Smith	15	John	Doe
60000	David	Jones	12	Bill	Howard
35000	Jon	Doe	17	Kevin	Smith
25000	kevin	Smiht	14	David	Jones

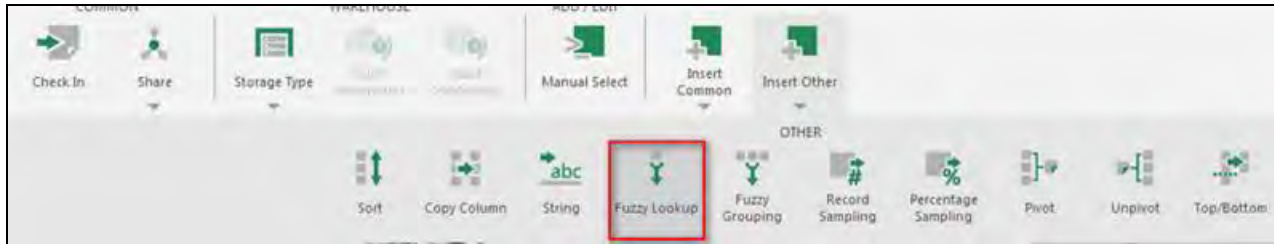
DailySales input data **SalesAgent** input data

2. Add the transform

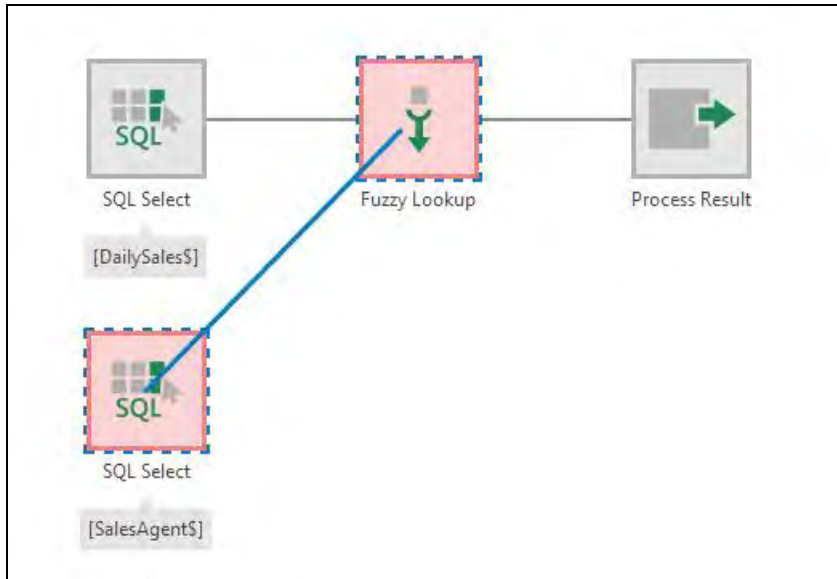
Click to select the connector link between two transforms.



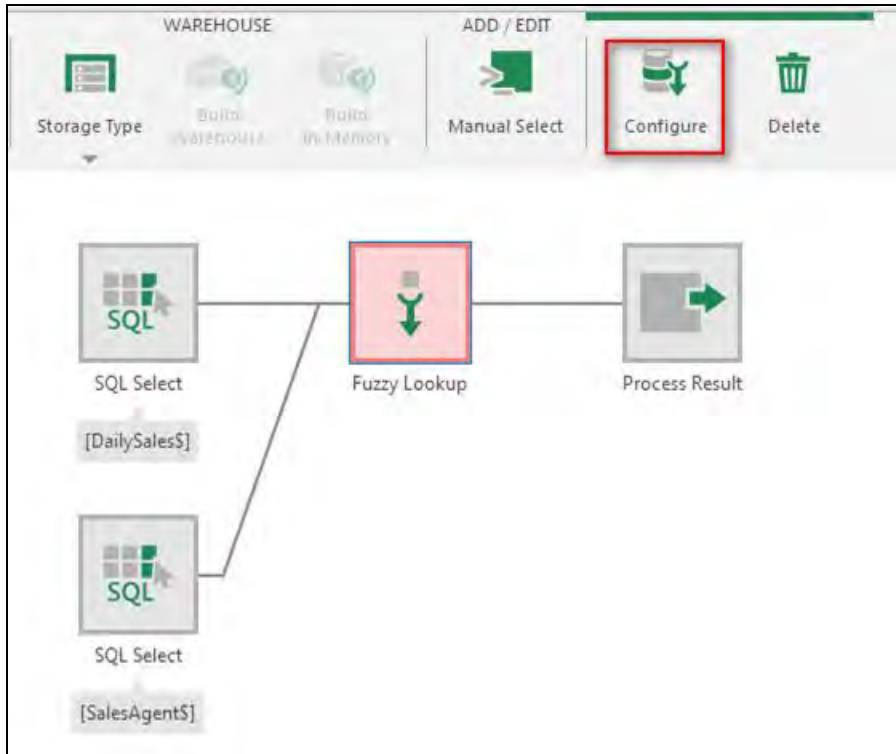
In the toolbar, choose **Insert Other**, then **Fuzzy Lookup**.



Click the other input transform and drag a connection to the Fuzzy Lookup transform.



To edit/configure the transform, select it and choose **Configure** in the toolbar.



Configure

The **Fuzzy Lookup** configuration dialog lists the columns from the two input tables.

Choose elements to be used in this transform. Drag an element from a table to an element of the other table to create a unique key binding.

Input Table

☒ Name	 
☒ Surname	 
☒ Sales	

Lookup Table

☒ AgentID	
☒ Name	 
☒ Surname	 

Probability Threshold

0.7

Maximum Matches

1

Ignore String Case

☐

Define parameters 

Edit output elements 

You will need to designate one or more columns as keys in each table, which will be matched together.

If a [relationship](#) between the two tables was already defined, the key elements may already have been set. This will be indicated by a key icon shown next to one or more items in each list, like in the figure above. If you want to remove an existing key binding for your lookup, click the **Delete** icon next to the key icon.

To define a key binding, drag a column from one input table and drop it onto the corresponding column in the other input table.

The following settings determine how to handle differences between the key values:

- Enter the **Probability Threshold**. Valid values are from .0001 to 1.0, where a value of 1 will require input data to be an exact match for them to be looked up successfully.
- Set **Maximum Matches** to determine the maximum number of output records that can be returned when multiple matches are found.
- Select **Ignore String Case** if you want a non-case-sensitive match.

Uncheck an element in either list to exclude them from the output. You can click Edit output elements at the bottom to rename an element.

Output

The figure below illustrates the output in our example.

Sales	AgentID	Name	Surname	Name	Surname
50000	15	John	Doe	John	Doe
80000	17	Kevin	Smith	Kevin	Smith
60000	14	David	Jones	David	Jones
35000	15	Jon	Doe	John	Doe
25000	17	kevin	Smih	Kevin	Smith

For more information, see:

- [Using Transforms](#)
- [Join](#)
- [Lookup](#)
- [Create a New Data Cube](#)



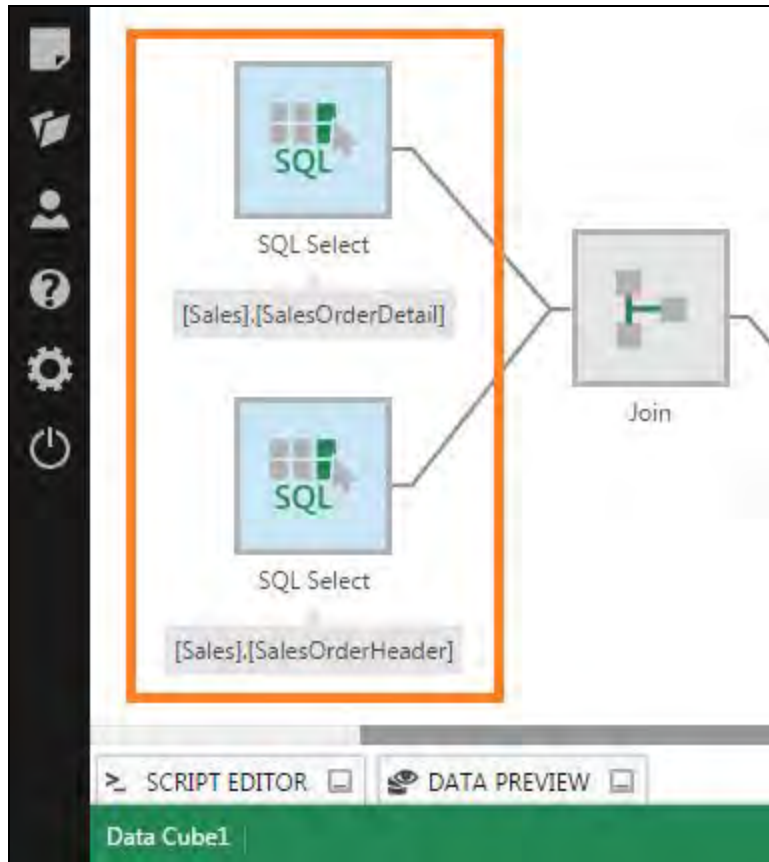
- Archive of documentation for Logi Symphony v24

- [Data Cubes](#)
- [Add a Relationship Between Tables](#)

Give a Friendly Name to a Transform

This applies to: *Managed Dashboards, Managed Reports*

When you add multiple [SQL Select](#) transforms to your data cube, each transform node on the canvas displays the name of the associated SQL Server database table. In this case, it is easy to identify the transforms on the canvas.

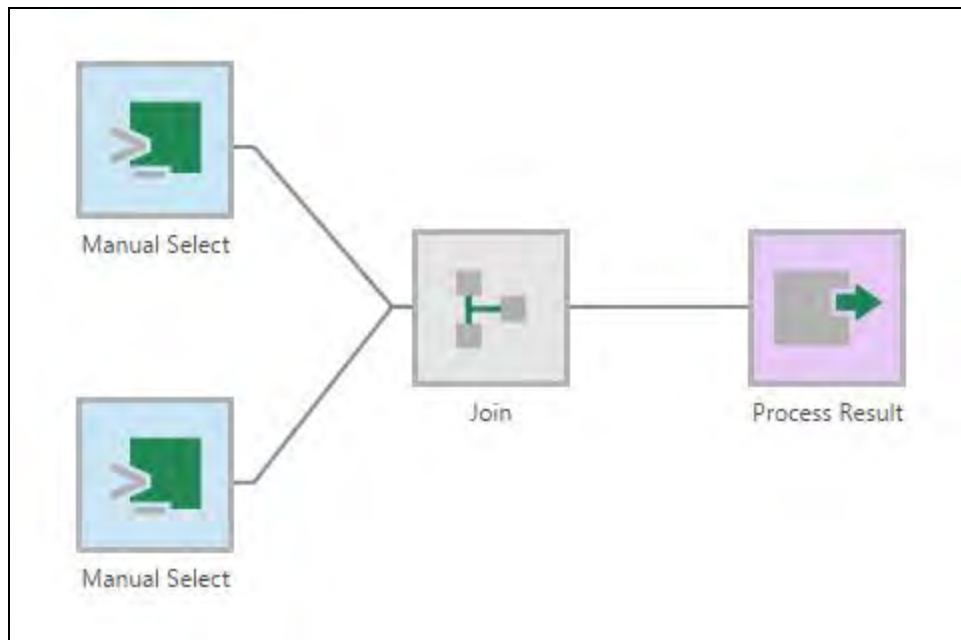


However, when you are working with multiple [Manual Select](#) transforms, there may be no suitable default name for them and they all appear identical on the canvas, making it difficult to identify them.

The solution in this case is to give a friendly name and optionally a description to each of these transforms.

Friendly Name and Description

As an example, consider a data cube process consisting of two Manual Select transforms.



Double-click the first Manual Select transform to configure it. In the Manual Select dialog, set the Name and Description fields. Optionally, select a color for a transform to differentiate it for other users.

Help 

Manual Select

The manual select transform. The user is responsible for entering a valid select statement for the data connector of his choice.

Name

Color

Products



Description

Manual query selection from Production.Product

ID

1dbb61d7-c7df-460c-a590-e5faf5ab4ef1

Settings

Data Connector

ee

Timeout

30

Element Discovery Execution Behavior

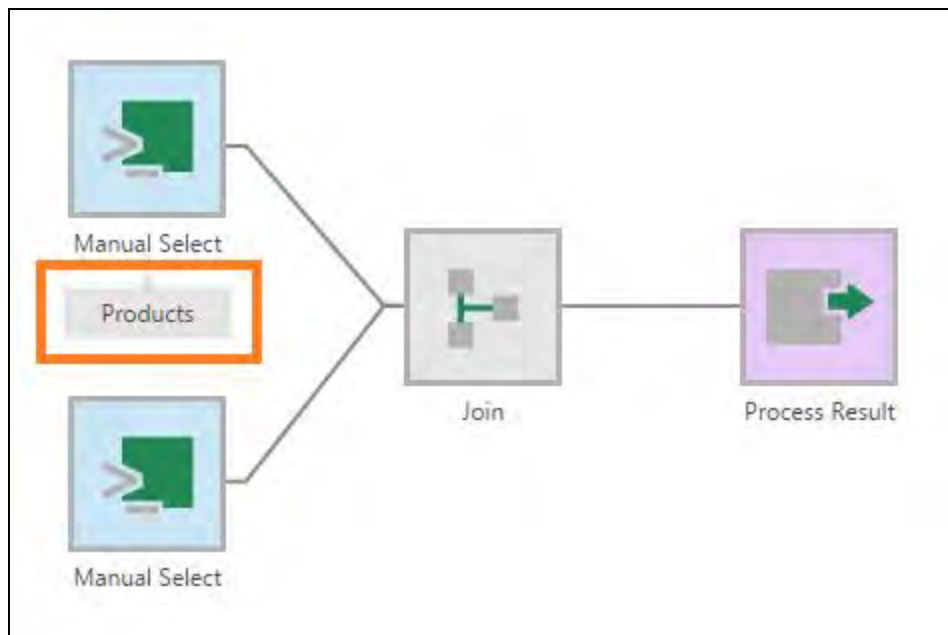
Default



Enable Subquery Optimization



Returning to the data cube's canvas, you'll see that the Manual Select transform now appears with its friendly name displayed. To see the description in a tooltip, hover over the transform.



Note: Friendly names are not displayed for native selection transforms such as SQL Select because these transforms display the table name instead.

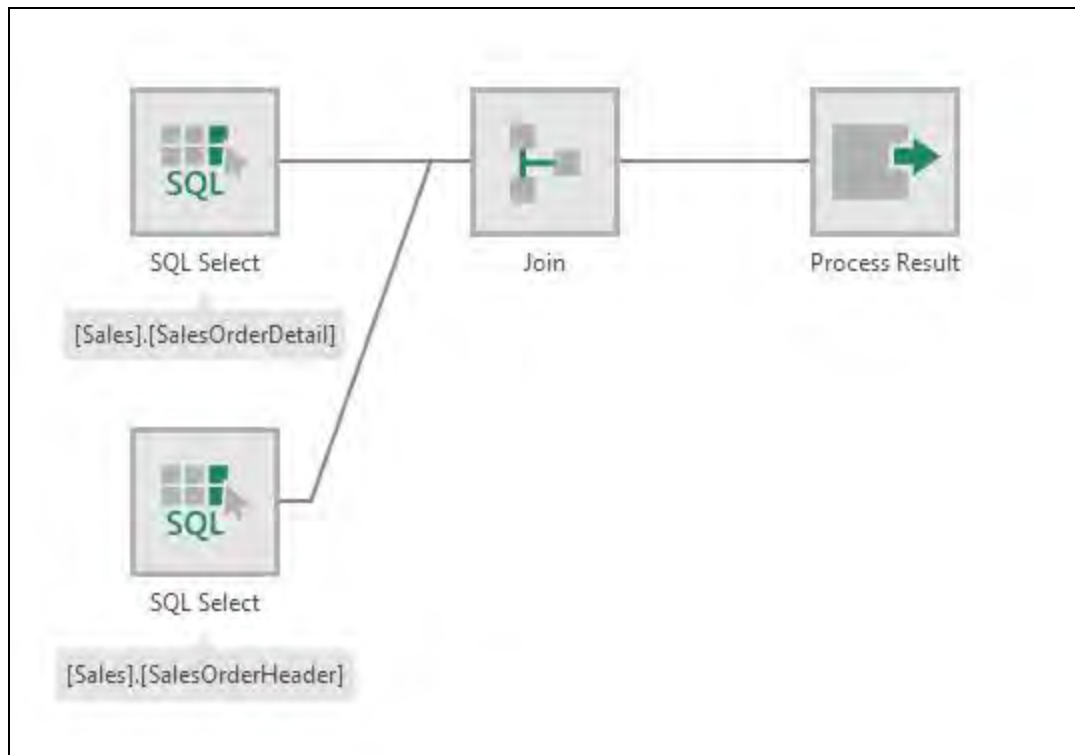
For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Join

This applies to: *Managed Dashboards, Managed Reports*

The Join transform combines two separate tables together by matching up their rows. All of the columns from both tables can be included.



If the data connector of the transforms being joined are the same, and that provider supports joining, the joining can be automatically included as part of the query sent to the database if no preceding transform is added that must be run directly by Symphony.

Input

The Join transform requires two input transforms.

Configure

The **Join** configuration dialog lists the columns of the two input tables, and includes settings to determine how they are joined.

Join Type
Inner

Select the elements to be used in this transform. Drag an element from a table to an element of the other table to create a unique key binding.

Left Input
[Sales].[SalesOrderHeader]

Right Input
[Sales].[SalesOrderDetail]

☒ AccountNumber
☒ BillToAddressID
☒ Comment
☒ CreditCardApprovalCode
☒ CreditCardID
☒ CurrencyRateID
☒ CustomerID
☒ DueDate
☒ Freight
☒ ModifiedDate

☐ SalesOrderID
☒ SalesOrderDetailID
☒ CarrierTrackingNumber
☒ OrderQty
☒ ProductID
☒ SpecialOfferID
☒ UnitPrice
☒ UnitPriceDiscount
☒ LineTotal
☒ Rowguid

Define parameters

Choose the **Join Type** to determine which records should be included:



- **Inner Join** - Only records with matching key values in both input tables.
- **Left Join** - Only records with a key value in **Left Input**.
- **Right Join** - Only records with a key value in **Right Input**.
- **Full Join** - All original records will be included.



Note:



Note: Any setting other than **Inner Join** can result in *null* or missing values in certain columns.

A join requires you to designate one or more columns as keys in each table. Rows from the two tables with matching key values will be combined into a single row.

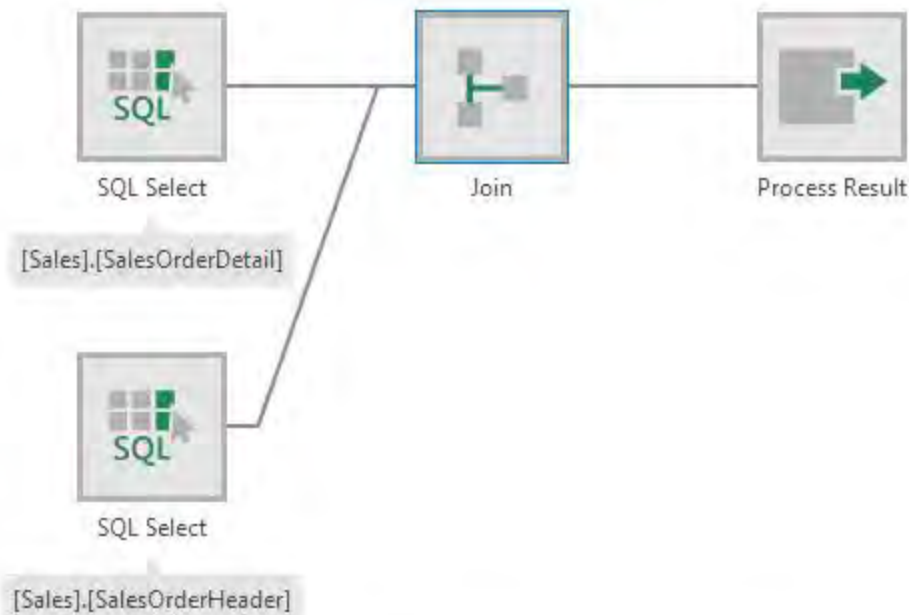
If a [relationship](#) between the two tables was already defined, the key elements may have already been set. This will be indicated by a key icon shown next to one or more items in each list, like in the figure above. If you want to remove an existing key binding for your join, click the Delete icon next to the key icon.

To define a key binding, drag a column from one input table and drop it onto the corresponding column in the other input table.

Use the checkboxes to select or de-select columns for inclusion in the output. For example, you might choose to de-select a column in one list that contains the same values as another column already selected.

Output

The figure below illustrates the output of a join between two tables that each contained a *SalesOrderID* column used as the key values.



> SCRIPT EDITOR

DATA PREVIEW

SalesOrderID	OrderQty	ProductID	SalesOrderID	SalesPersonID	ModifiedDate
43659	1	776	43659	279	07/01/2005 00:00:00
43659	3	777	43659	279	07/01/2005 00:00:00
43659	1	778	43659	279	07/01/2005 00:00:00
43659	1	771	43659	279	07/01/2005 00:00:00
43659	1	772	43659	279	07/01/2005 00:00:00
43659	2	773	43659	279	07/01/2005 00:00:00
43659	1	774	43659	279	07/01/2005 00:00:00



- Archive of documentation for Logi Symphony v24

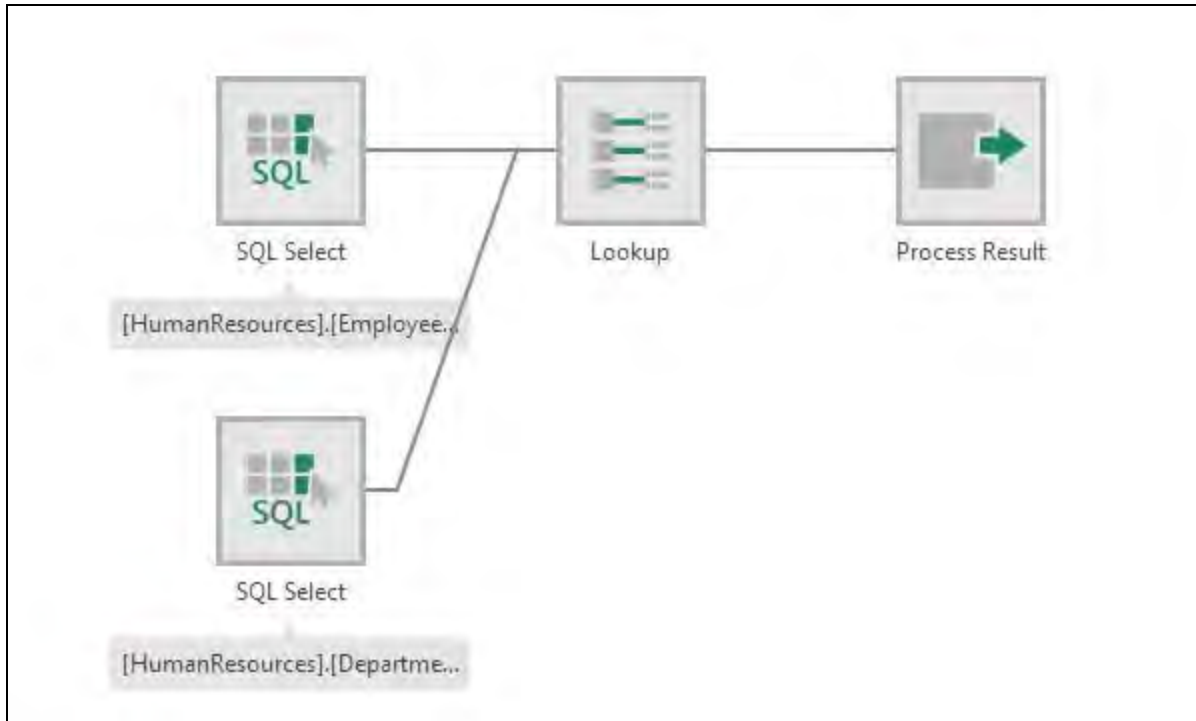
For more information, see:

- Video: [Introduction to Data Cubes](#)
- Video: [Data Cleansing with Data Cubes](#)
- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Add a Relationship Between Tables](#)

Lookup

This applies to: *Managed Dashboards, Managed Reports*

The **Lookup** transform replaces the values of one or more key columns with the values you choose in another table.



This can be used to replace IDs with more descriptive text values, for example.

Input

The Lookup transform requires two input transforms.

For example, the main input could be the following data, which includes a *DepartmentID*:

```
SELECT [BusinessEntityID]
      ,[DepartmentID]
      ,[ShiftID]
      ,[StartDate]
      ,[EndDate]
      ,[ModifiedDate]
FROM [AdventureWorks 2012].[HumanResources].[EmployeeDepartmentHistory]
```

Results		Messages				
BusinessEntityID	DepartmentID	ShiftID	StartDate	EndDate	ModifiedDate	
1	16	1	2003-02-15	NULL	2003-02-14 00:00:00.000	
2	1	1	2002-03-03	NULL	2002-03-02 00:00:00.000	
3	1	1	2001-12-12	NULL	2001-12-11 00:00:00.000	
4	1	1	2002-01-05	2004-06-30	2004-06-28 00:00:00.000	
4	2	1	2004-07-01	NULL	2004-06-30 00:00:00.000	
5	1	1	2002-02-06	NULL	2002-02-05 00:00:00.000	
6	1	1	2002-02-24	NULL	2002-02-23 00:00:00.000	
7	6	1	2003-03-12	NULL	2003-03-11 00:00:00.000	
8	6	1	2003-01-30	NULL	2003-01-29 00:00:00.000	
9	6	1	2003-02-17	NULL	2003-02-16 00:00:00.000	
10	6	1	2003-06-04	NULL	2003-06-03 00:00:00.000	
11	2	1	2005-01-05	NULL	2005-01-04 00:00:00.000	
12	2	1	2002-01-11	NULL	2002-01-10 00:00:00.000	
13	2	1	2005-01-23	NULL	2005-01-22 00:00:00.000	
14	1	1	2005-01-30	NULL	2005-01-29 00:00:00.000	
15	1	1	2005-02-18	NULL	2005-02-17 00:00:00.000	
16	5	1	2002-01-20	2003-08-15	2003-08-13 00:00:00.000	

With the following department details used as a lookup input:

```
SELECT [DepartmentID]
      , [Name]
      , [GroupName]
      , [ModifiedDate]
FROM [AdventureWorks 2012].[HumanResources].[Department]
```

DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and Development	2002-06-01 00:00:00.000
2	Tool Design	Research and Development	2002-06-01 00:00:00.000
3	Sales	Sales and Marketing	2002-06-01 00:00:00.000
4	Marketing	Sales and Marketing	2002-06-01 00:00:00.000
5	Purchasing	Inventory Management	2002-06-01 00:00:00.000
6	Research and Development	Research and Development	2002-06-01 00:00:00.000
7	Production	Manufacturing	2002-06-01 00:00:00.000
8	Production Control	Manufacturing	2002-06-01 00:00:00.000
9	Human Resources	Executive General and Administration	2002-06-01 00:00:00.000
10	Finance	Executive General and Administration	2002-06-01 00:00:00.000
11	Information Services	Executive General and Administration	2002-06-01 00:00:00.000
12	Document Control	Quality Assurance	2002-06-01 00:00:00.000
13	Quality Assurance	Quality Assurance	2002-06-01 00:00:00.000
14	Facilities and Maintenance	Executive General and Administration	2002-06-01 00:00:00.000
15	Shipping and Receiving	Inventory Management	2002-06-01 00:00:00.000
16	Executive	Executive General and Administration	2002-06-01 00:00:00.000

Configure

The **Lookup** configuration dialog lists the columns from the two input tables, and allows you to choose the replacement values.

Drag an element from a table to an element of the other table to create a unique key binding.

Input Table
[HumanResources].[EmployeeDepartmentHistory]

☒ BusinessEntityID
☒ DepartmentID  
☒ ShiftID
☒ StartDate
☒ EndDate
☒ ModifiedDate

Lookup Table
[HumanResources].[Department]

DepartmentID  
Name
GroupName
ModifiedDate

⇌

Lookup Element
Name

Omit Non Matches
☐

Define parameters 

Edit output elements 

The lookup requires you to designate one or more columns as keys in each table. Rows from the lookup table with matching key values will be used to replace the key values in the output.

If a [relationship](#) between the two tables was already defined, the key elements may already have been set. This will be indicated by a key icon shown next to one or more items in each list, like in the figure above. If you want to remove an existing key binding for your lookup, click the **Delete** icon next to the key icon.

To define a key binding, drag a column from one input table and drop it onto the corresponding column in the other input table. Set **Lookup Element** to the column under **Lookup Input** that will be used to replace the key columns in the output.

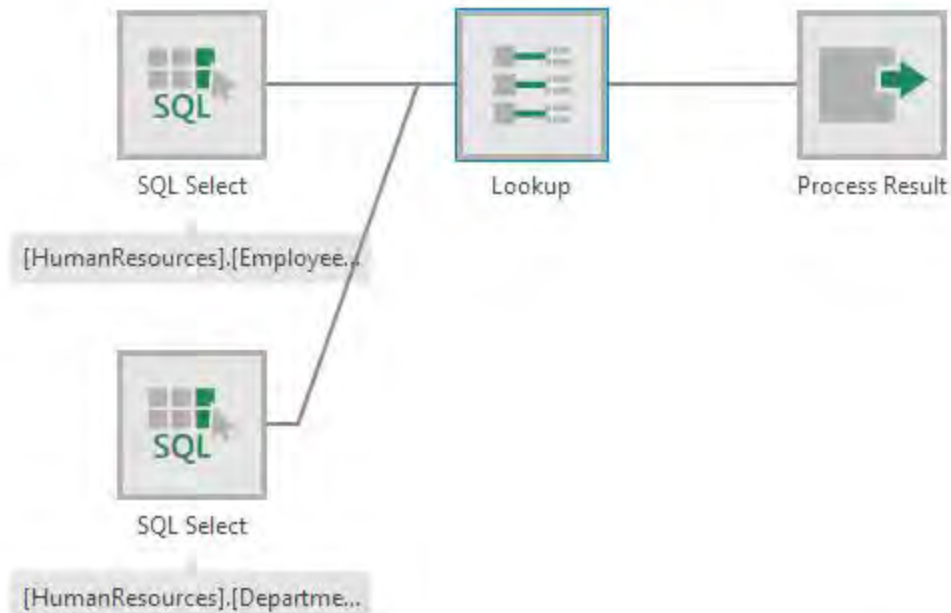


- Archive of documentation for Logi Symphony v24

Selecting the **Omit Non Matches** checkbox will exclude rows where a match was not found in the lookup table. Use the checkboxes to select or de-select columns for inclusion in the output.

Output

The figure below illustrates the output in our example.



> SCRIPT EDITOR

DATA PREVIEW

BusinessEntityID	DepartmentID	ShiftID	StartDate	EndDate	ModifiedDate	Name
2	1	1	03/03/2002 00:00:00		03/02/2002 00:00:00	Engineering
3	1	1	12/12/2001 00:00:00		12/11/2001 00:00:00	Engineering
4	1	1	01/05/2002 00:00:00	06/30/2004 00:00:00	06/28/2004 00:00:00	Engineering
5	1	1	02/06/2002 00:00:00		02/05/2002 00:00:00	Engineering
6	1	1	02/24/2002 00:00:00		02/23/2002 00:00:00	Engineering
14	1	1	01/30/2005 00:00:00		01/29/2005 00:00:00	Engineering



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Using Transforms](#)
- [Fuzzy Lookup](#)
- [Join](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Add a Relationship Between Tables](#)

Manual Select

This applies to: *Managed Dashboards, Managed Reports*

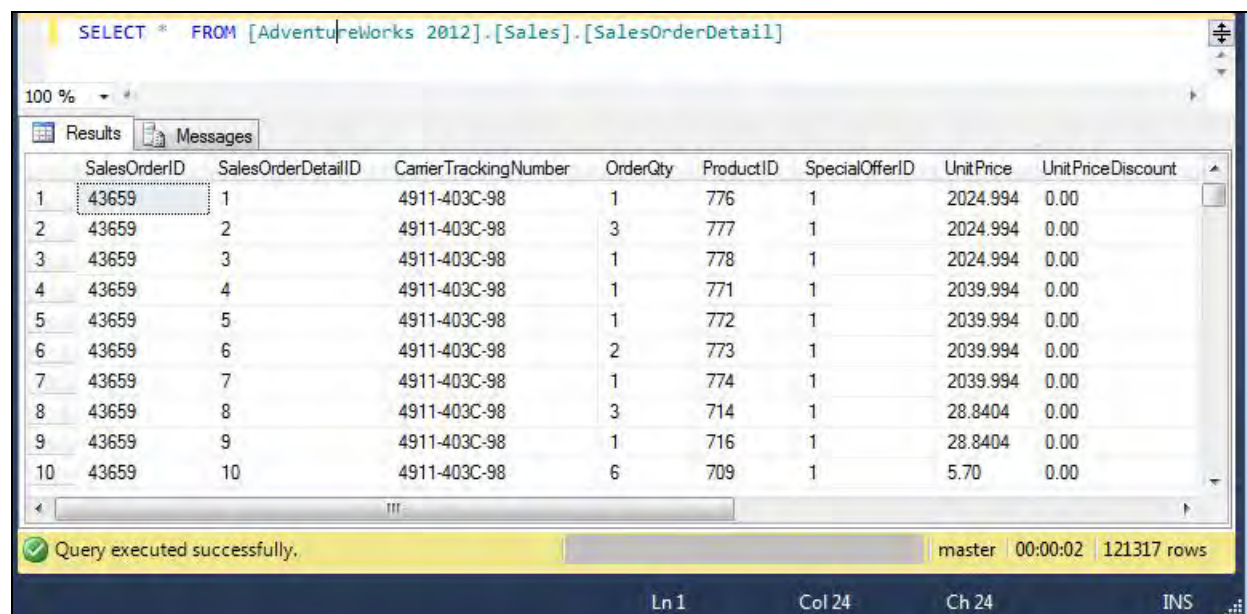
Enter a SQL or MDX statement to make a selection from a data connector instead of dragging native structures onto the canvas.

Input

The input to a Manual Select is a custom query executed against a data connector that you select.

You can write your own query, or paste one obtained from another tool to retrieve the same results.

For example, the figure below shows the data retrieved by a simple query:



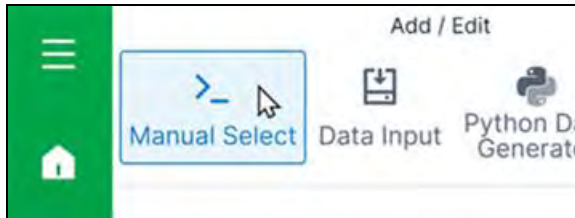
The screenshot shows the Logi Analytics interface with a SQL query editor at the top and a results table below. The query is: `SELECT * FROM [AdventureWorks 2012].[Sales].[SalesOrderDetail]`. The results table displays 10 rows of data with columns: SalesOrderID, SalesOrderDetailID, CarrierTrackingNumber, OrderQty, ProductID, SpecialOfferID, UnitPrice, and UnitPriceDiscount. A status bar at the bottom indicates 'Query executed successfully.' and '121317 rows'.

	SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount
1	43659	1	4911-403C-98	1	776	1	2024.994	0.00
2	43659	2	4911-403C-98	3	777	1	2024.994	0.00
3	43659	3	4911-403C-98	1	778	1	2024.994	0.00
4	43659	4	4911-403C-98	1	771	1	2039.994	0.00
5	43659	5	4911-403C-98	1	772	1	2039.994	0.00
6	43659	6	4911-403C-98	2	773	1	2039.994	0.00
7	43659	7	4911-403C-98	1	774	1	2039.994	0.00
8	43659	8	4911-403C-98	3	714	1	28.8404	0.00
9	43659	9	4911-403C-98	1	716	1	28.8404	0.00
10	43659	10	4911-403C-98	6	709	1	5.70	0.00

Add the Transform

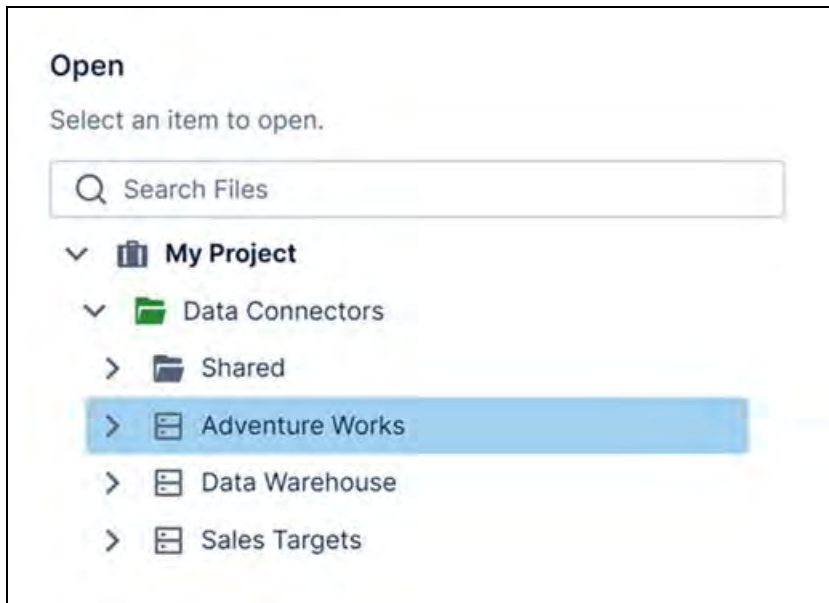
If you already have a blank or existing data cube open, select the Manual Select transform from the toolbar.

Alternatively, create a data cube and choose Manual Select from the menu that appears. This transform will be added automatically when the new data cube opens.



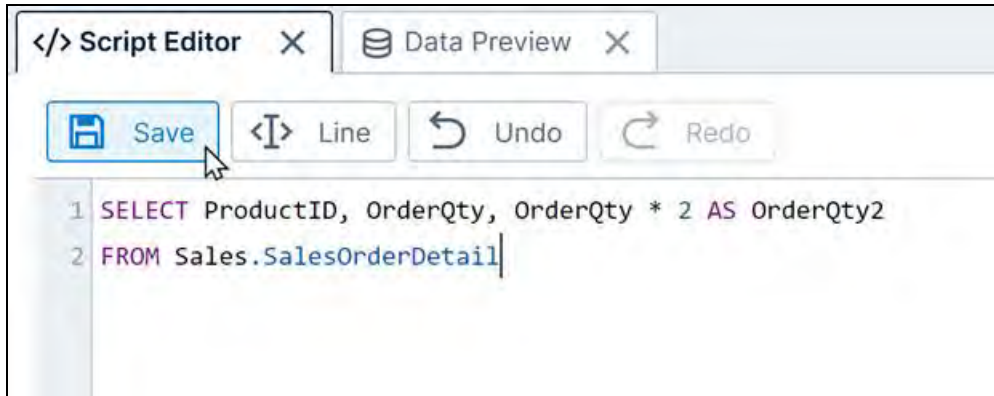
Note: You may need to be assigned an [application privilege](#) by your administrator before you can access this transform.

Select a **data connector**. For OLAP databases, you can also expand the data connector and select a cube to query.



Enter the query in the **Script Editor**.

Note: For some data providers, inserting comments into your query may cause an issue when subquery optimization is enabled.



Use the keyboard shortcut **Ctrl + Space** for a list of available SQL keywords and table names. You can also drag the native structure (cube, table, column, measure, or hierarchy level) from the Explore window to the Script Editor to add the respective unique name.

Select **Save**.

Note: You can also [drag a stored procedure](#) from the Explore window onto the canvas instead of typing it into a query.

Note: Subquery optimization is enabled by default, which can mean inserting comments into your query, ending your query with a semicolon, or certain statements such as executing a stored procedure may result in an error unless the option is disabled. You can simply drag a [Stored Procedure from the Explore window](#) onto the data cube canvas instead of using a **Manual Select**.

Note: MDX queries must return a cellset or multidimensional data set (**DRILLTHROUGH** is not supported).

Configure

To configure the transform, select it and choose Configure in the toolbar.

Settings

Data Connector

Adventure Works

Timeout

30

Element Discovery Execution Behavior

Default

Enable Subquery Optimization



Search Input Elements



ProductID

Int32



OrderQty

Int16



OrderQty2

Int32

Query Statement

```
1 SELECT ProductID, OrderQty, OrderQty * 2 AS OrderQty2
2 FROM Sales.SalesOrderDetail
```

Edit query in editor



The **Timeout** (in seconds) can be customized to determine how long to wait before terminating a long-running command.



Note: The **Default Query Timeout** [application configuration setting](#) determines the initial timeout setting.

Edit the Query

Edit the **Query Statement** directly from the configuration dialog, or click **Edit query in editor** to reopen the Script Editor window.

Enable Subquery Optimization

Subquery optimization allows your query to be incorporated automatically into a larger query that also performs data cube transformations, metric set aggregations, and filtering, so that this work is done in the database to optimize performance when possible.

Your query must be able to be enclosed in a SELECT statement when this optimization is enabled. For statements such as executing a stored procedure, or advanced or database-specific SQL features such as creating temporary tables, uncheck the **Enable Subquery Optimization** option in the transform configuration dialog. You may also need to uncheck this option if your data connector uses an ODBC driver that incorrectly reports its capabilities.

Enable Subquery Optimization

☐

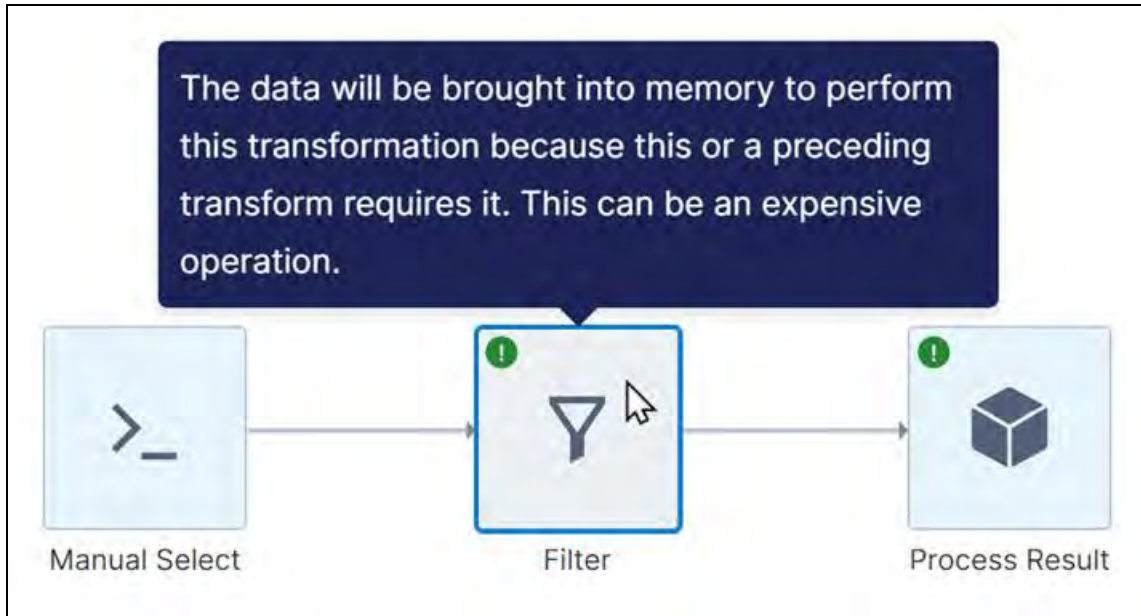
Search Input Elements

☒	ProductID	Int32
☒	OrderQty	Int16

Query Statement

```
1 EXEC [dbo].[sp_ProductID_Hierarchy] 777
```


When this option is unchecked, transforms connected to the manual select transform's output will display an icon indicating that the data is being brought into memory in Symphony to perform the transformation rather than using the database.



Placeholders

The Manual Select transform also lets you define a *placeholder* and use it in your manual queries. A placeholder inserts a parameter into the query, which you can set from outside the transform or optionally the data cube, for example by connecting it to a filter on a dashboard.

See [Using Manual Select Placeholders](#).

Session Schema

Some data providers support an **Allow Session Schemas** option in Symphony v24.2 and later. If supported, enable when [creating or editing a data connector](#), such as PostgreSQL, MySQL, and Oracle.

When enabled in the data connector you're using, a **Session Schema** option is available when you configure the Manual Select transform. Use this to enter the name of a schema to use.

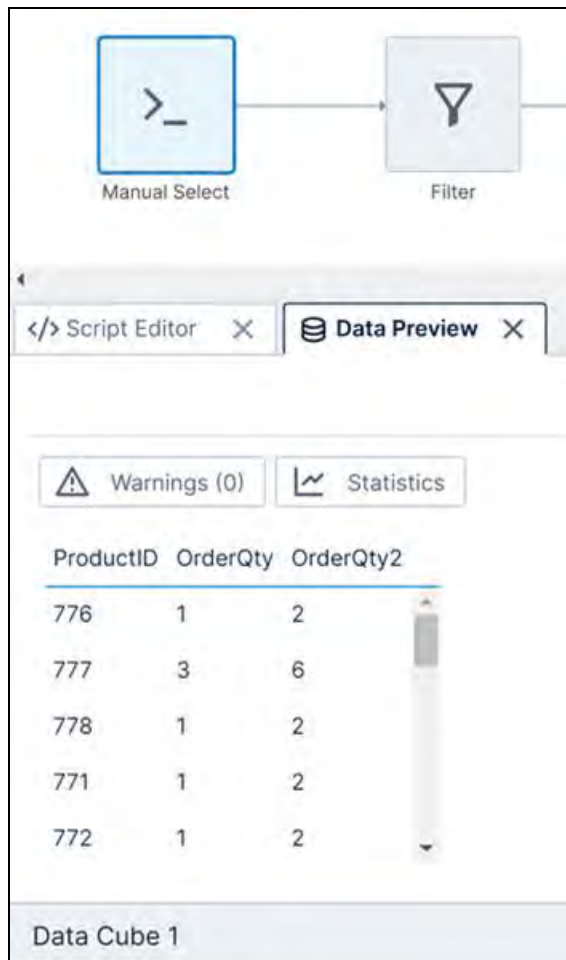
If you want to allow the schema to be changed via a parameter on a dashboard, report, or other view, you can also create a public parameter:

1. Select **Parameters**.
2. Select **Add parameter** in the new dialog that opens.

3. Set **Parameter Type** to **Transform Setting** and **Transform Setting** to **Session Schema**, then select **Save**.
4. Select the **Public** option in the next set of options that appears, then **Save**.

Output

Open the **Data Preview** window to see the output. If the query did not succeed, this may include errors or warnings explaining why.



The screenshot shows a data pipeline with two components: 'Manual Select' and 'Filter'. Below the pipeline, there is a 'Data Preview' window. The window has tabs for 'Script Editor' and 'Data Preview'. The 'Data Preview' tab is active, showing a table with columns 'ProductID', 'OrderQty', and 'OrderQty2'. The table contains five rows of data. At the bottom of the window, it says 'Data Cube 1'.

ProductID	OrderQty	OrderQty2
776	1	2
777	3	6
778	1	2
771	1	2
772	1	2

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Stored Procedure Select](#)
- [Passing Placeholder Value From Dashboard to Manual MDX Select](#)
- [Data Cubes - Manual Select](#)
- [The Managed Dashboards Data Model](#)
- [Using Manual Select Placeholders](#)

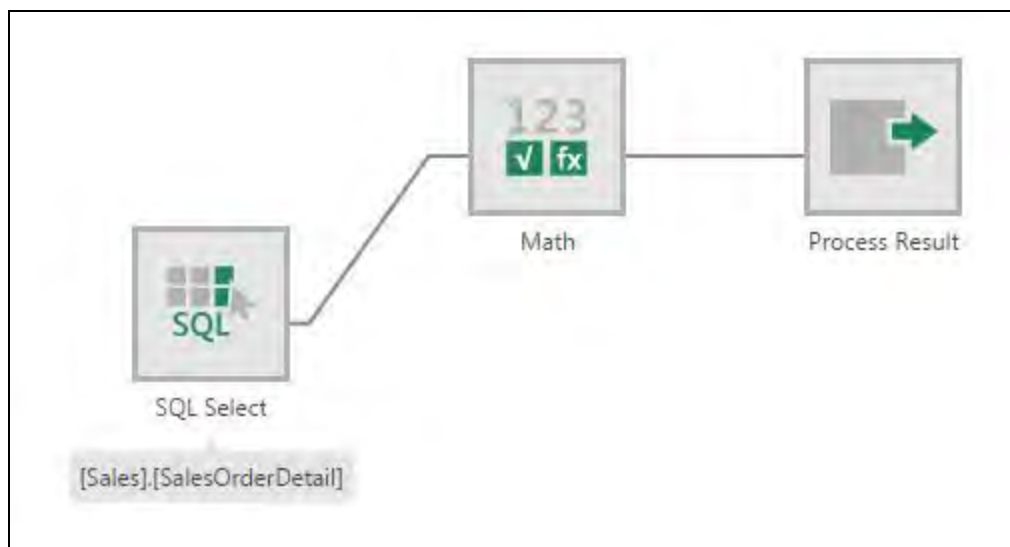
Math

This applies to: *Managed Dashboards, Managed Reports*

The **Math** transform lets you perform simple math functions such as Absolute, Round, and Square Root on numeric input columns. It is similar to the [String](#) transform except that it offers math functions.

The following functions are available:

- Round
- Absolute
- Ceiling
- Floor
- Truncate
- Square Root
- Multiply by 100





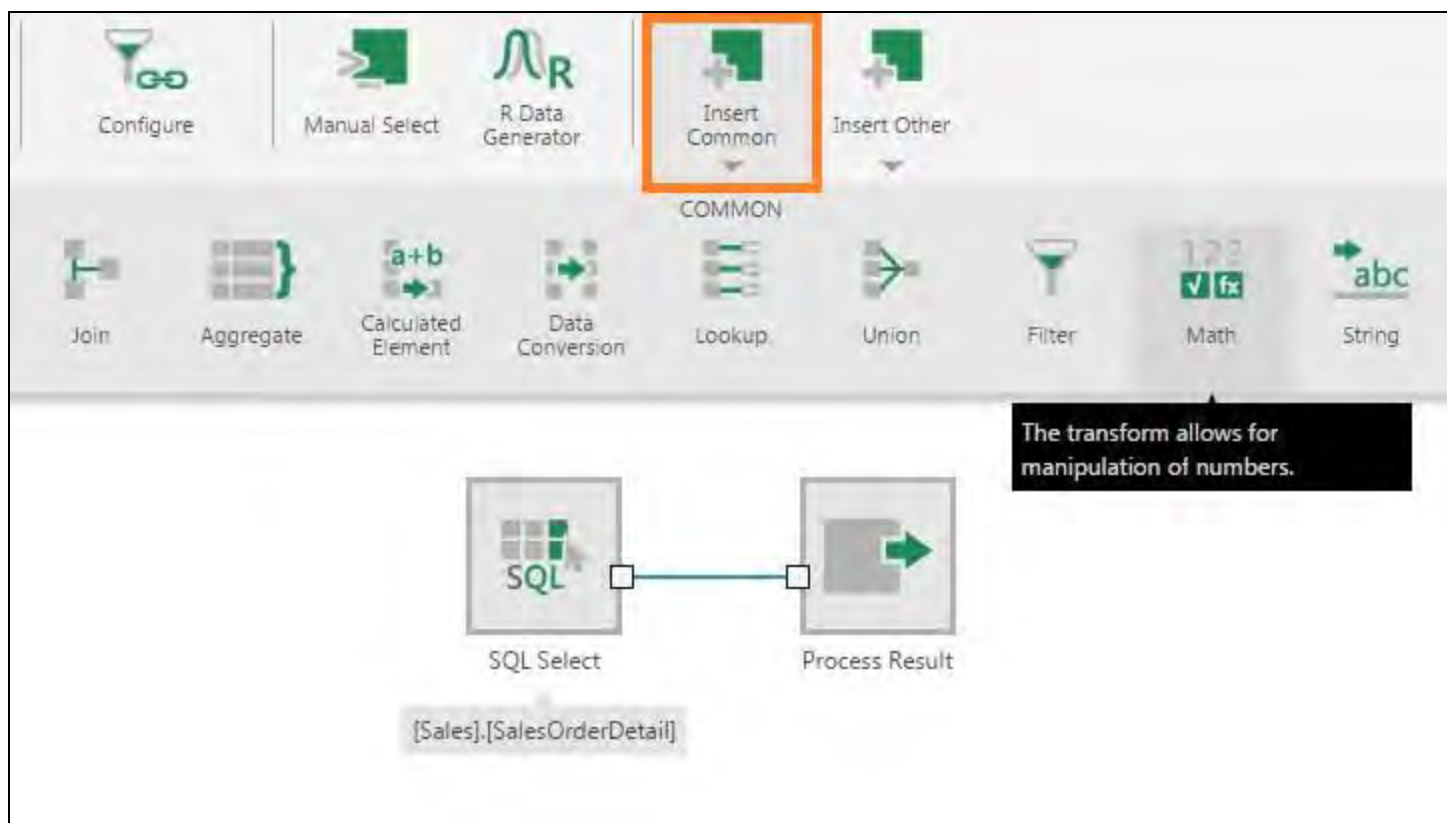
Note: You can also apply [formula functions](#) by using a [Calculated Element](#) transform.

Input

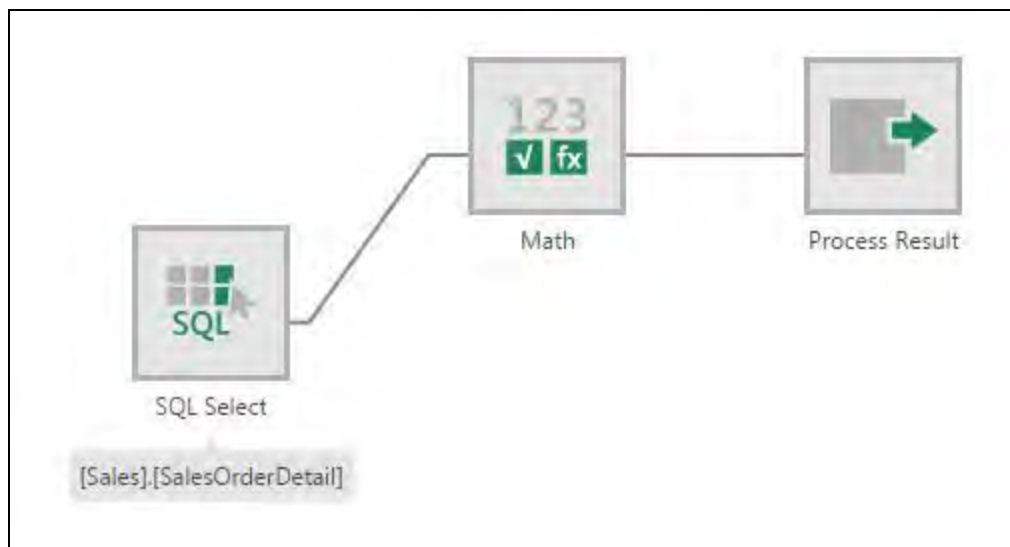
The Math transform requires one input transform that has one or more numeric columns.

Add the Transform

To add this transform to an existing data cube process, first select the connection link between two connected transforms. Go to the toolbar, click **Insert Common**, and then select **Math**.



The Math transform is inserted between the two transforms.



Configure the Transform

Double-click the Math transform or select the **Configure** option from its right-click menu.

In the configuration dialog for the transform, decide which input column you want to apply a math function on, and then click the column's **Edit** icon. Also uncheck any columns that should be excluded from the output.

[more help](#)

Math

The transform allows for manipulation of numbers.

Name

Description

		Edit the element.
☒ CarrierTrackingNumber	String	
☒ LineTotal	Double	
☒ ModifiedDate	DateTime	
☒ OrderQty	Int16	
☒ ProductID	Int32	
☒ rowguid	Guid	
☒ SalesOrderDetailID	Int32	
☒ SalesOrderID	Int32	
☒ SpecialOfferID	Int32	
☒ UnitPrice	Currency	
☒ UnitPriceDiscount	Currency	

Define parameters

Rename output elements



- Archive of documentation for Logi Symphony v24

In the **Transform Element** dialog, use the drop-down to select the Math function you want (e.g., Round). Then click the submit button at the bottom of the dialog.

You can apply math functions to more than one numeric column with a single transform.

Math

The transform allows for manipulation of numbers.

Name

Description

☒ CarrierTrackingNumber String 

☒ LineTotal Double 

☒ ModifiedDate DateTime

☒ OrderQty Int16 

☒ ProductID Int32 

☒ rowguid Guid

☒ SalesOrderDetailID Int32 

☒ SalesOrderID Int32 

☒ SpecialOfferID Int32 

☒ UnitPrice Currency 

☒ UnitPriceDiscount Currency 

Define parameters 

Rename output elements 

Transform Element

Settings for a transform element.

Math Function

Round

None

Round

Absolute

Ceiling

Floor

Truncate

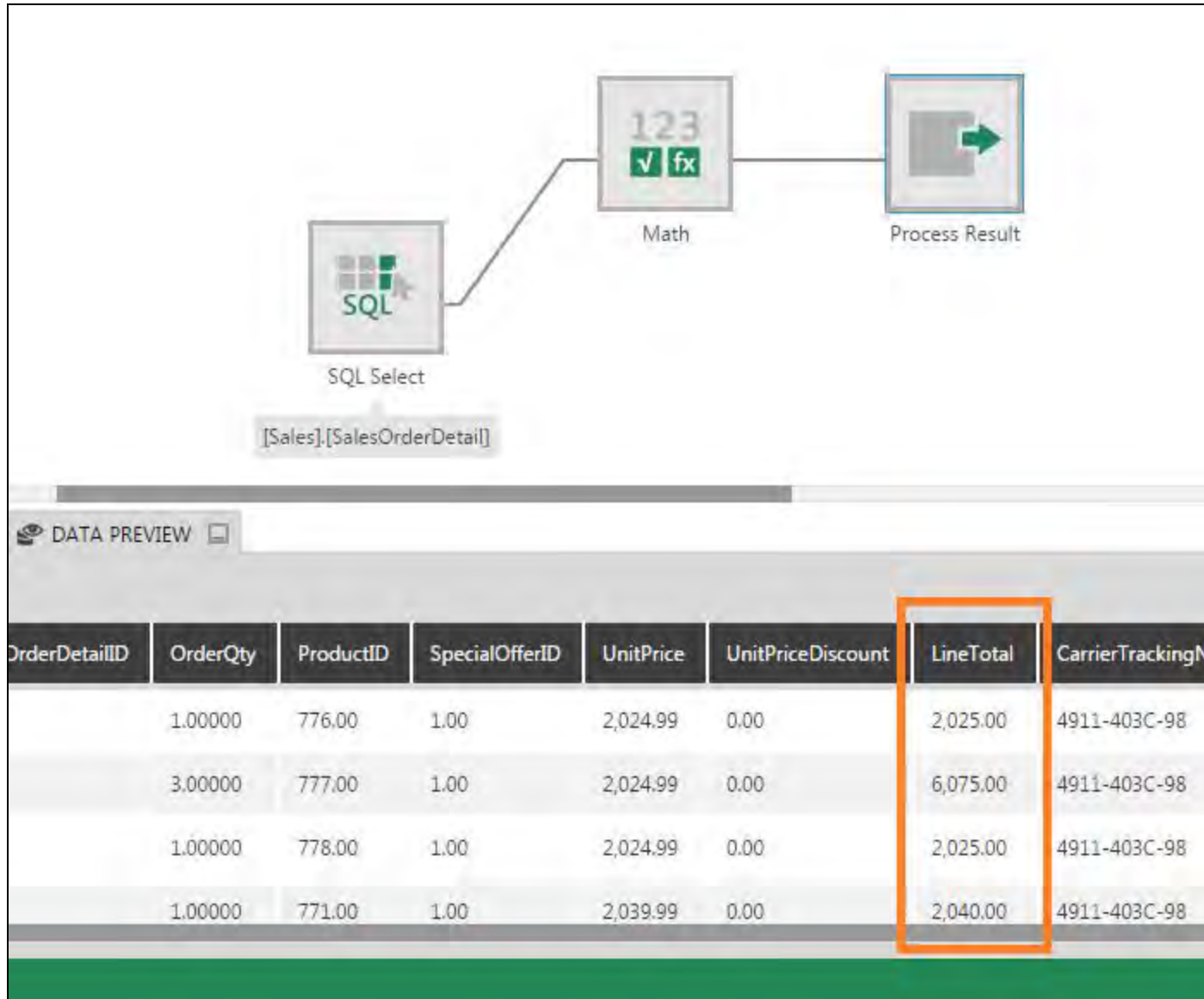
Square Root

Multiply By 100

Output

The output of the Math transform consists of the columns that were checked, where the values in each numeric column with a math function selected are changed.

Click the **Process Result** transform and open the **Data Preview** to see the output of the transform.



The screenshot displays a data flow diagram with three transforms: **SQL Select**, **Math**, and **Process Result**. The **SQL Select** transform is connected to the **Math** transform, which is then connected to the **Process Result** transform. Below the **SQL Select** transform, the data source is specified as `[Sales].[SalesOrderDetail]`. The **Math** transform icon shows a calculator and the text "123 √ fx". The **Process Result** transform icon shows a green arrow pointing right.

Below the diagram, the **DATA PREVIEW** tab is active, showing a table with the following columns: **OrderDetailID**, **OrderQty**, **ProductID**, **SpecialOfferID**, **UnitPrice**, **UnitPriceDiscount**, **LineTotal**, and **CarrierTrackingN**. The **LineTotal** column is highlighted with an orange box.

OrderDetailID	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal	CarrierTrackingN
1.00000	776.00	1.00	2,024.99	0.00	2,025.00	4911-403C-98	
3.00000	777.00	1.00	2,024.99	0.00	6,075.00	4911-403C-98	
1.00000	778.00	1.00	2,024.99	0.00	2,025.00	4911-403C-98	
1.00000	771.00	1.00	2,039.99	0.00	2,040.00	4911-403C-98	



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Using Transforms](#)
- [String](#)
- [Calculated Element](#)

MDX Select

This applies to: *Managed Dashboards, Managed Reports*

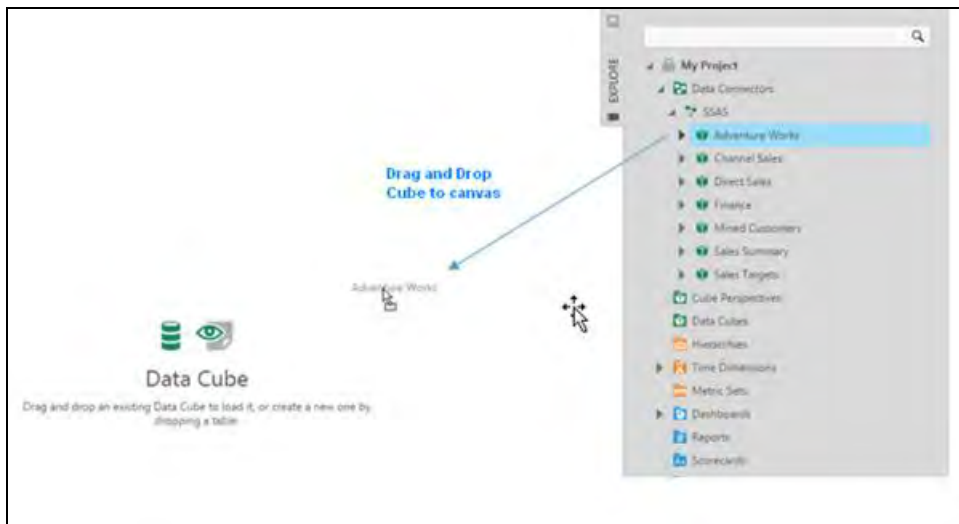
The MDX Select transform is created when a cube from an OLAP database (e.g., Microsoft SQL Server Analysis Services) is dragged onto the canvas from its data connector.

Input

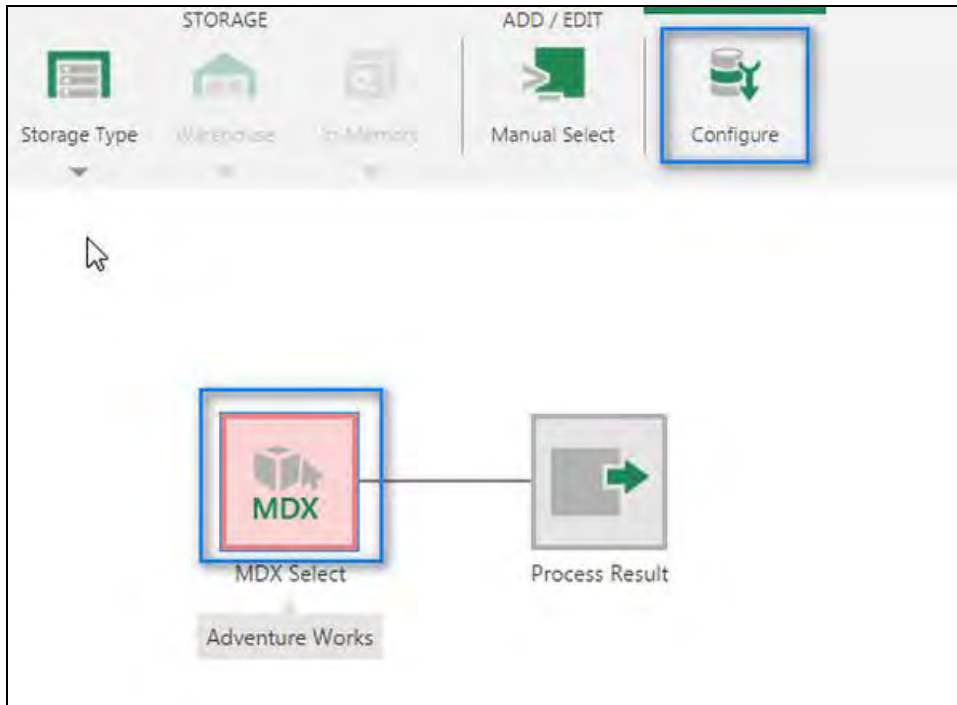
Configure an MDX Select transform to select specific measures and dimensions to return a table of data. This tabular result becomes an input to your data cube in Symphony.

Add the Transform

Expand the data connector in the **Data Connectors** folder in the **Explore** window, then drag one of its cubes to the canvas.



Configure the transform by selecting the transform and clicking **Configure** in the toolbar.



Configure

To configure the MDX Select transform, drag and drop the measures and dimensions to the right pane.

For a dimension hierarchy, only one hierarchy level can be selected. Use the **Hierarchy Level Display Property** drop-down to indicate whether you want the captions (display values) or the unique names (keys) of the hierarchy's members.

Name

Adventure Works

Description

ID

3262a2b5-1b5d-47d0-8ae6-15385d1e9b7c

Native Structure

Adventure Works

Data Connector

AdventureWorksOLAP

Measures

Measures

- Exchange Rates
- Financial Reporting
- Internet Customers
- Internet Orders
- Internet Sales
- Reseller Orders
- Reseller Sales
- Sales Orders
- Order Count
- Sales Summary
- Sales Targets

Selected Measures

Order Count

Drop items here

Dimensions

Sales Territory

Scenario

Ship Date

Calendar

- Ship Date, Calendar
- Calendar Year
- Calendar Semester
- Calendar Quarter
- Month
- Date
- Ship Date, Calendar Quarter of Year
- Ship Date, Calendar Semester of Year

Selected Dimensions

Date

Drop items here

Hierarchy Level Display Property

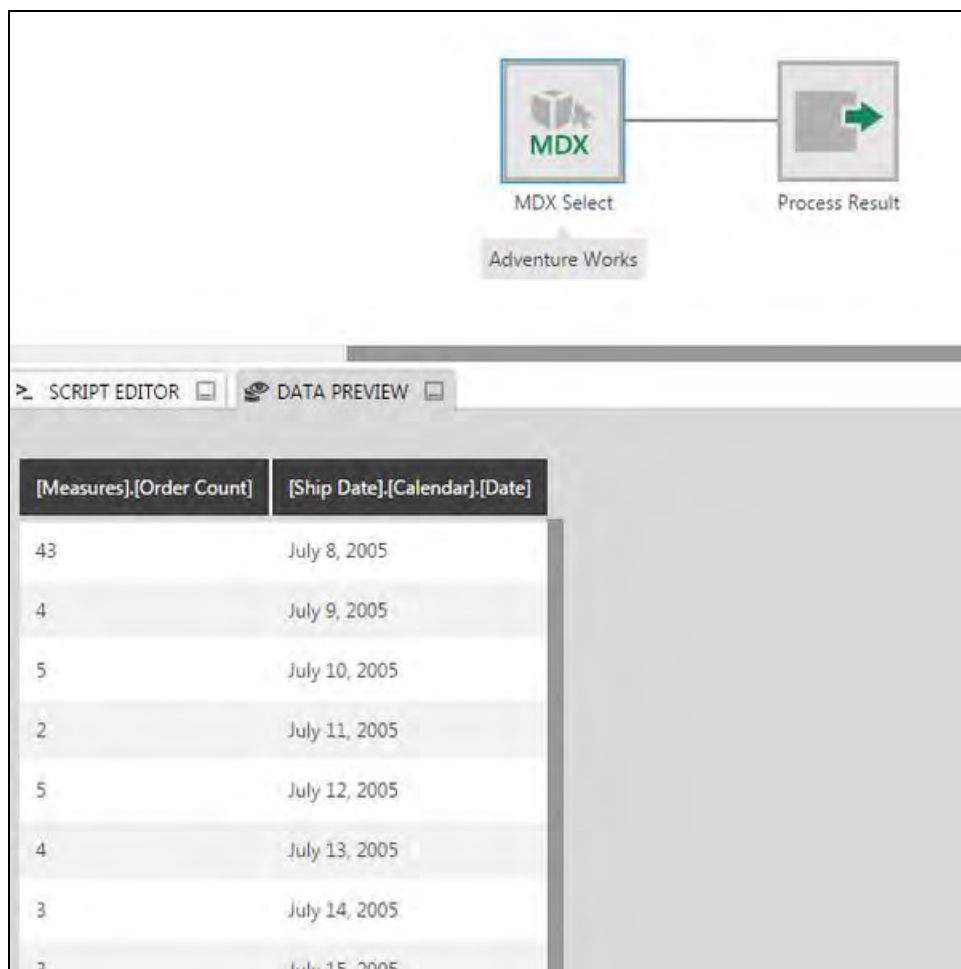
Hierarchy Level Caption

Edit output elements

Click the **Submit** button at the bottom of the dialog when finished.

Output

Open the **Data Preview** window to view the output.



The screenshot shows the Logi Analytics interface. At the top, there is a diagram with two components: 'MDX Select' (represented by a cube icon) and 'Process Result' (represented by a square icon with a green arrow). A line connects them, and a label 'Adventure Works' is positioned below the 'MDX Select' component. Below the diagram, there is a tabbed interface with two tabs: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table with two columns: '[Measures].[Order Count]' and '[Ship Date].[Calendar].[Date]'. The table contains data for dates from July 8, 2005, to July 15, 2005.

[Measures].[Order Count]	[Ship Date].[Calendar].[Date]
43	July 8, 2005
4	July 9, 2005
5	July 10, 2005
2	July 11, 2005
5	July 12, 2005
4	July 13, 2005
3	July 14, 2005
3	July 15, 2005

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Manual Select](#)
- [Using Transforms](#)
- [Data Cubes](#)
- [Connect to OLAP Data and Apply a Formula](#)
- [The Managed Dashboards Data Model](#)

Merge

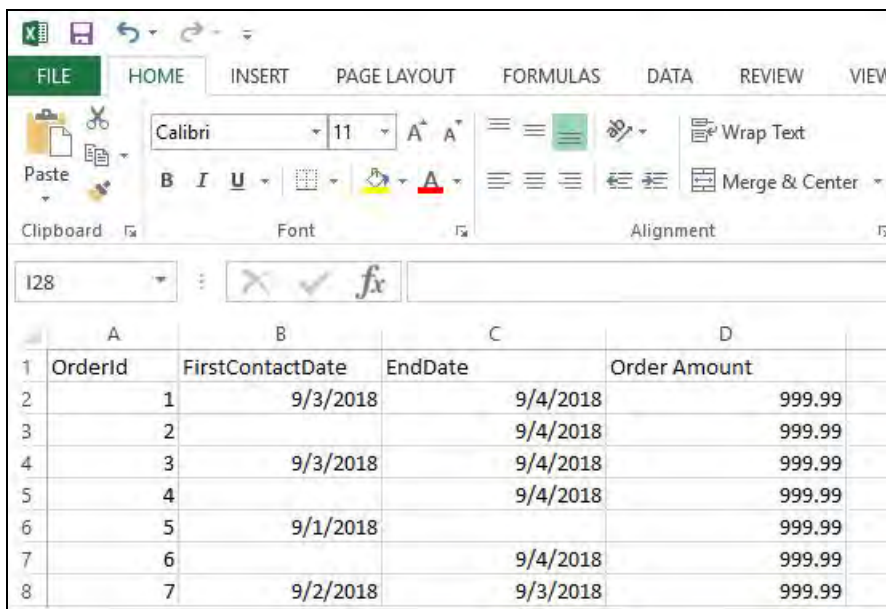
This applies to: *Managed Dashboards, Managed Reports*

The Merge transform allows values from two or more columns to be merged into a single column. This might be useful in cases such as after a full outer join where one column may be null.

This article provides a simple example of merging two date columns.

Input

The Merge transform requires two or more columns. This example will merge the **FirstContactDate** and **EndDate** columns from this data:



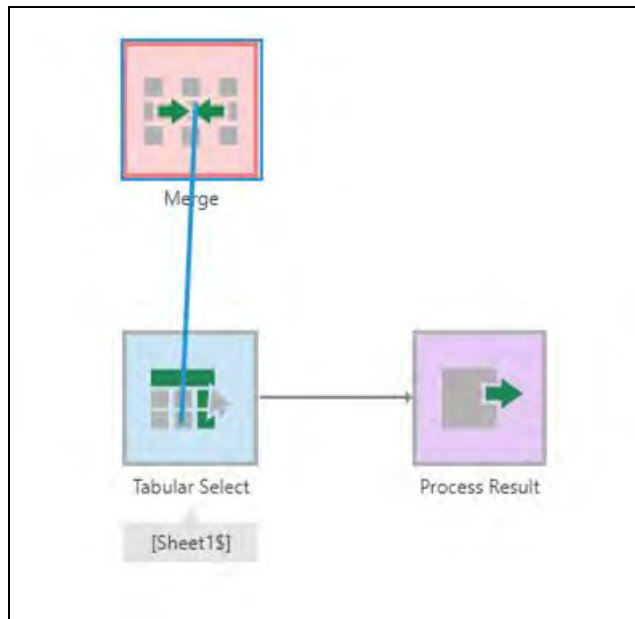
	A	B	C	D
1	OrderId	FirstContactDate	EndDate	Order Amount
2	1	9/3/2018	9/4/2018	999.99
3	2		9/4/2018	999.99
4	3	9/3/2018	9/4/2018	999.99
5	4		9/4/2018	999.99
6	5	9/1/2018		999.99
7	6		9/4/2018	999.99
8	7	9/2/2018	9/3/2018	999.99

Add the Transform

In the toolbar, choose **Insert Other**, then **Merge**.

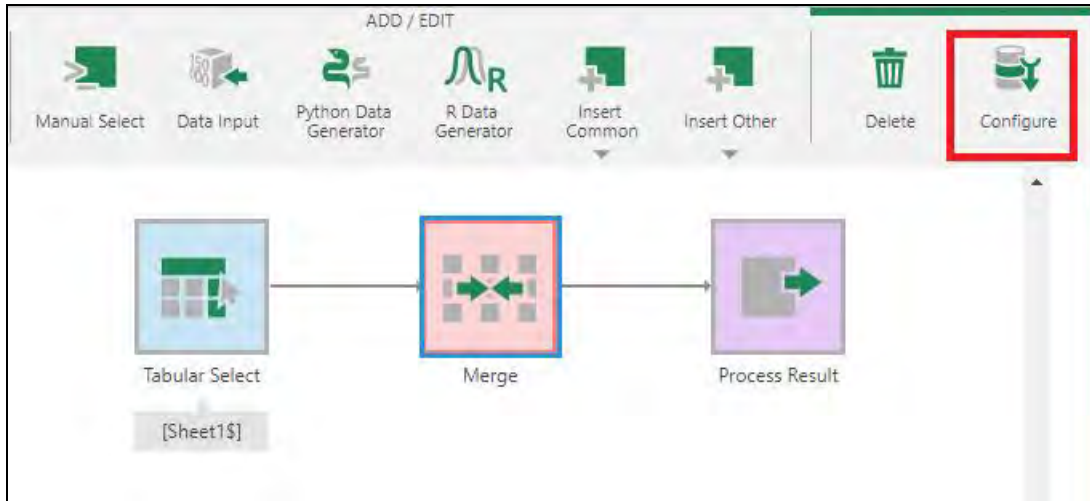


Click and drag from the input transform (e.g., Tabular Select) onto the **Merge** transform.



Configure

Next we need to configure the transform. Click to select it, and choose **Configure** in the toolbar.



The **Merge** configuration dialog will appear. Here you need to drag the elements you want to merge and drop them under **Elements To Merge**.

The **Merge Rule** is initially set to **Priority**, which takes the first non-null value found based on the order of the Elements to Merge list. The other options are **Highest** (highest value will be merged), and **Lowest** (lowest value will be merged).



Merge

[more help](#)

The transform allows values from two or more columns to be merged into a single column.

Name

Description

ID

Merged Element Name

Merge Rule

Priority

Drag and drop the elements you want to merge into the 'Elements to Merge' section. Drag and drop within the list to set the prioritization.




Merge Transform Input

☒ OrderId
 ☒ FirstContactDate
 ☒ EndDate
 ☒ Order Amount

Elements to Merge

FirstContactDate
 

 EndDate
 

Define parameters

Edit output elements

Click the submit button at the bottom of the dialog when finished.

Output

To preview the output, open the **Data Preview** window with the merge transform selected.

The figure below illustrates the output for our example. Note the added **Merged Element** column with dates from the **FirstContactDate** and **EndDate** columns, with the priority based on the set order of the elements.

OrderId	FirstContactDate	EndDate	Order Amount	Merged Element
1	9/3/2018 12:00:00 AM	9/4/2018 12:00:00 AM	999.99	9/3/2018 12:00:00 AM
2	Unknown	9/4/2018 12:00:00 AM	999.99	9/4/2018 12:00:00 AM
3	9/3/2018 12:00:00 AM	9/4/2018 12:00:00 AM	999.99	9/3/2018 12:00:00 AM
4	Unknown	9/4/2018 12:00:00 AM	999.99	9/4/2018 12:00:00 AM
5	9/1/2018 12:00:00 AM	Unknown	999.99	9/1/2018 12:00:00 AM
6	Unknown	9/4/2018 12:00:00 AM	999.99	9/4/2018 12:00:00 AM

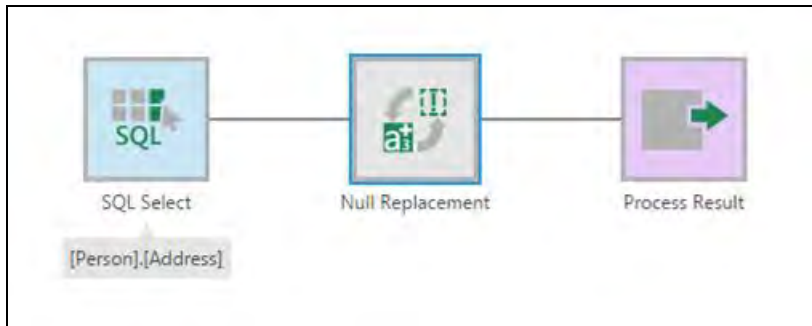
For more information, see:

- [Using Transforms](#)
- [Join](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Null Replacement

This applies to: *Managed Dashboards, Managed Reports*

The Null Replacement transform allows for the replacing of input data with user-specified values when **null** values are found. Use this as an alternative to a Calculated Element transform, which requires you to write a script.



Input

The **Null Replacement** transform requires one input transform that has at least one column.

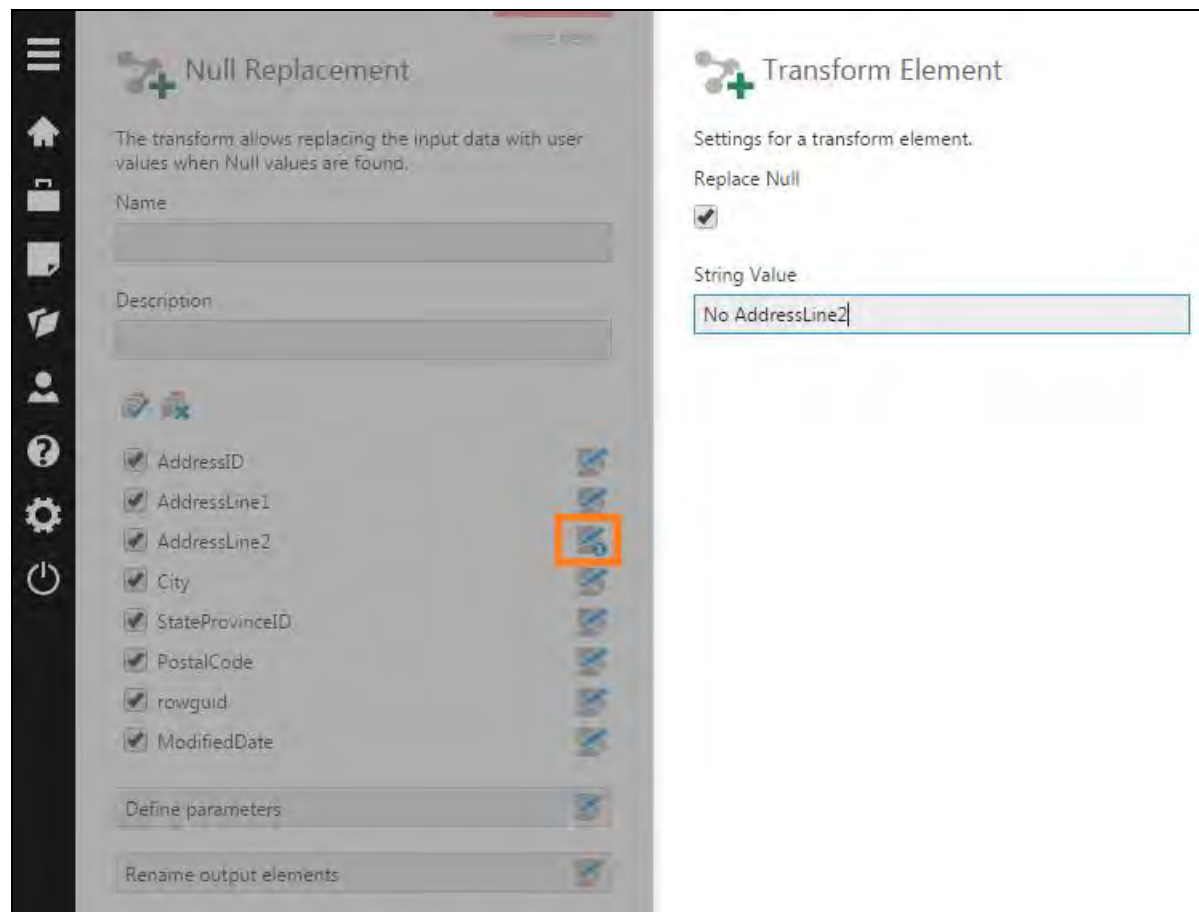
For example, the input could be a SQL Select transform that corresponds to the following SQL Server table from AdventureWorks 2012:

	AddressID	AddressLine1	AddressLine2	City	StateProvinceID	PostalCode
	83	5379 Treasure Island ..	# 14	Duvall	79	98019
	84	3421 Bouncing Road	NULL	Duvall	79	98019
	85	991 Vista Verde	NULL	Duvall	79	98019
	86	390 Ridgewood Ct.	NULL	Carnation	79	98014
	87	1411 Ranch Drive	NULL	Carnation	79	98014
	88	9666 Northridge Ct.	NULL	Carnation	79	98014
	89	3074 Arbor Drive	NULL	Carnation	79	98014
	90	9752 Jeanne Circle	NULL	Carnation	79	98014
	91	7166 Brock Lane	NULL	Seattle	79	98104
	92	7126 Ending Ct.	NULL	Seattle	79	98104
	93	4598 Manila Avenue	NULL	Seattle	79	98104
	94	5666 Hazelnut Lane	NULL	Seattle	79	98104
	95	1220 Bradford Way	NULL	Seattle	79	98104

Configure

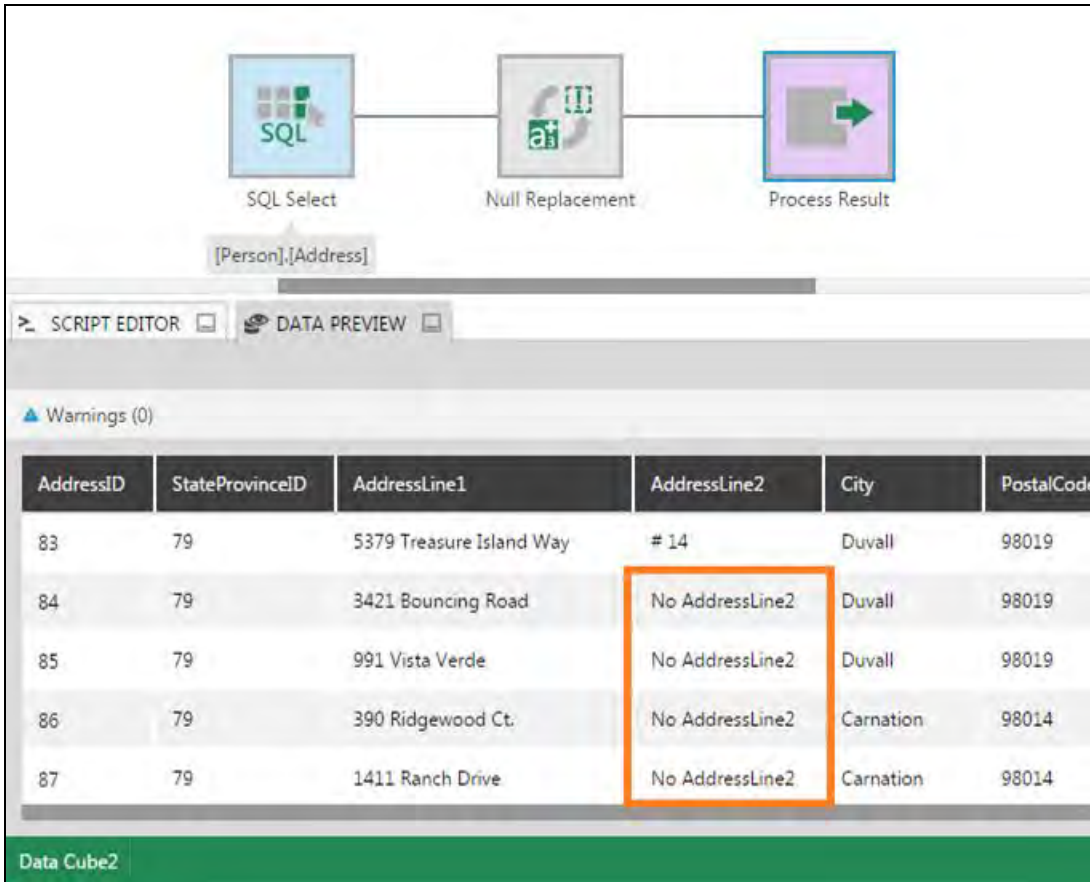
In the Null Replacement configuration dialog, **Edit** the column that may contain Null values.

Select the **Replace Null** option and then enter the replacement **String Value**.



Output

The output of the Null Replacement transform is the same as the input data except Null values in one or more columns may have been replaced with specified values.



The screenshot displays the Logi Symphony v24 interface. At the top, a data pipeline is shown with three steps: 'SQL Select' (blue icon), 'Null Replacement' (grey icon), and 'Process Result' (purple icon). Below the pipeline, the input data is labeled '[Person].[Address]'. The interface includes tabs for 'SCRIPT EDITOR' and 'DATA PREVIEW'. Below the tabs, a 'Warnings (0)' section is visible. The 'DATA PREVIEW' tab shows a table with the following data:

AddressID	StateProvinceID	AddressLine1	AddressLine2	City	PostalCode
83	79	5379 Treasure Island Way	# 14	Duvall	98019
84	79	3421 Bouncing Road	No AddressLine2	Duvall	98019
85	79	991 Vista Verde	No AddressLine2	Duvall	98019
86	79	390 Ridgewood Ct.	No AddressLine2	Carnation	98014
87	79	1411 Ranch Drive	No AddressLine2	Carnation	98014

The 'AddressLine2' column for rows 84, 85, 86, and 87 is highlighted with an orange box. At the bottom of the interface, a green bar indicates 'Data Cube2'.

For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)



- Archive of documentation for Logi Symphony v24

Passing Parameter Value From Dashboard to Stored Procedure

This applies to: *Managed Dashboards, Managed Reports*

This article demonstrates how to pass parameter values from a dashboard to a stored procedure. In this sample, a numeric filter is used in the dashboard, but the steps are similar for most data types. If you prefer to use a hierarchy/member filter rather than a textbox to enter string values, you would need [a bridge parameter](#).

Data

The Stored Procedure **uspGetBillOfMaterials** from [Adventure Works for SQL Server 2012](#) database is used in this example.

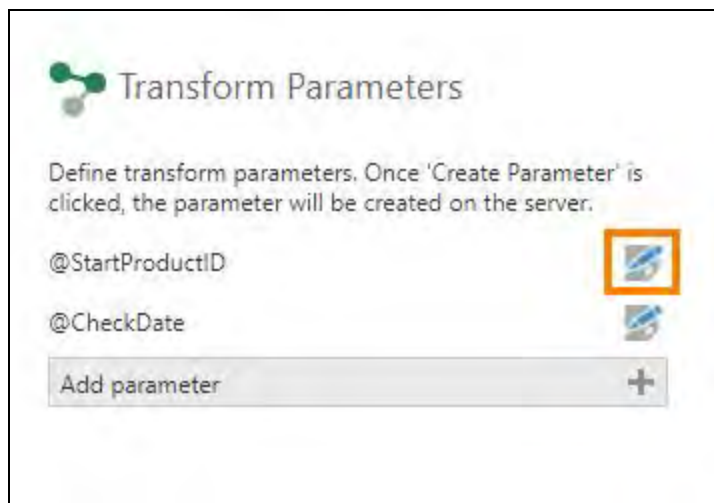
```
CREATE PROCEDURE [dbo].[uspGetBillofMaterials]
    @StartProductID [int],
    @CheckDate [datetime]
AS
BEGIN
    ...
```

Set up Data Cube Parameters

In a new data cube, go to the **Explore** window and locate your data connector. Expand its

Programmatic Structures folder, then drag and drop the **[dbo].[uspGetBillOfMaterials]** stored procedure (in our example) to the canvas. Select the **Stored Procedure** transform and choose **Configure** in the toolbar.

In the configuration dialog, click **Define Parameters**, and click the **Edit** icon next to each parameter that's being used.



Set the parameter to be public to be able to set its value from a dashboard.

The icon for the "Define Transform Parameter" window, showing three green circles connected by lines in a triangular shape.

Define Transform Parameter

Define the settings for the transform parameter.

Name

Description

Public
☒

Value type

☒ Single

Value



Define Transform Parameter

Define the settings for the transform parameter.

Name

Description

Public

☒

Value type

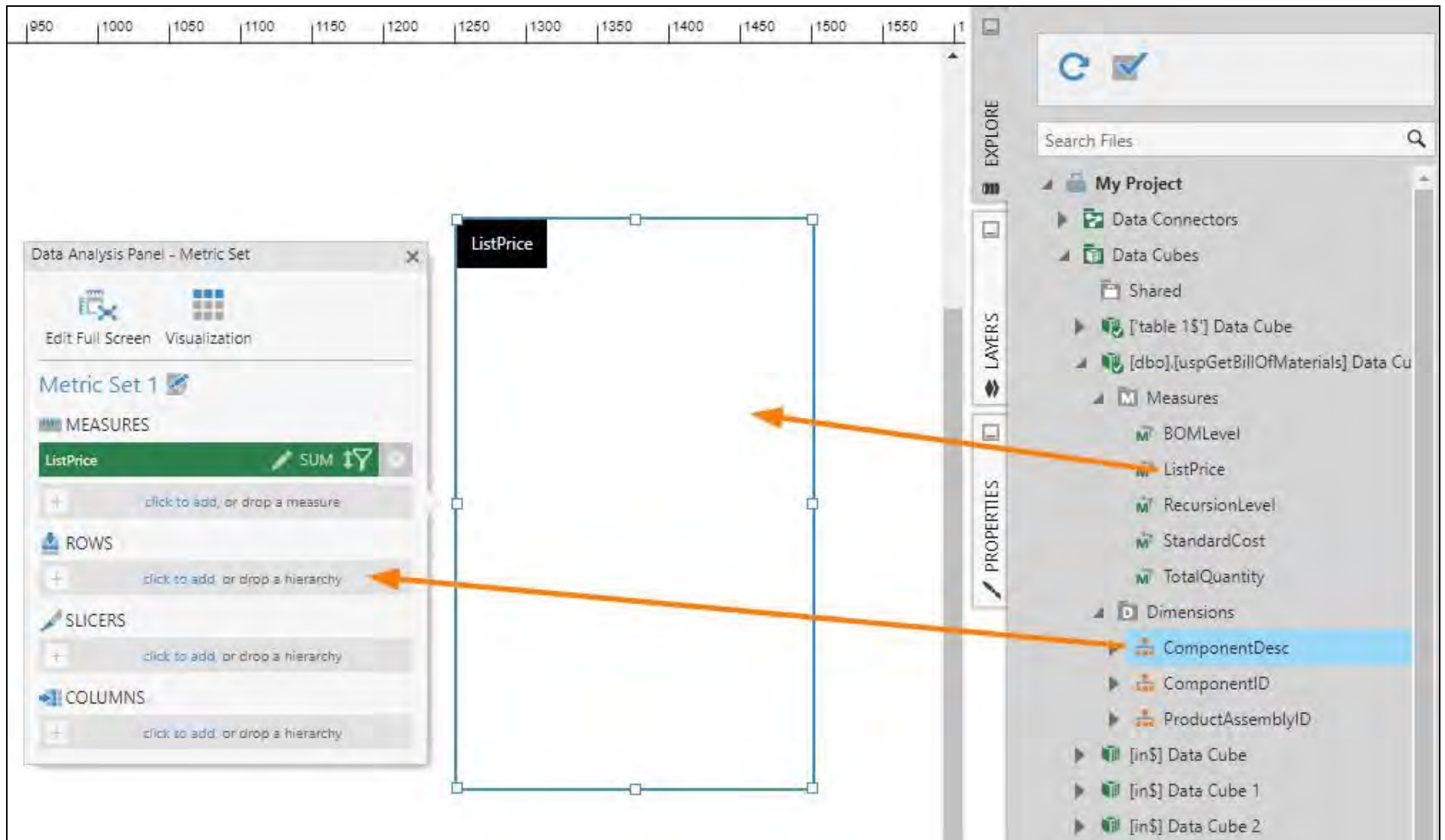
☒ Single

Value

Connect Dashboard Filters

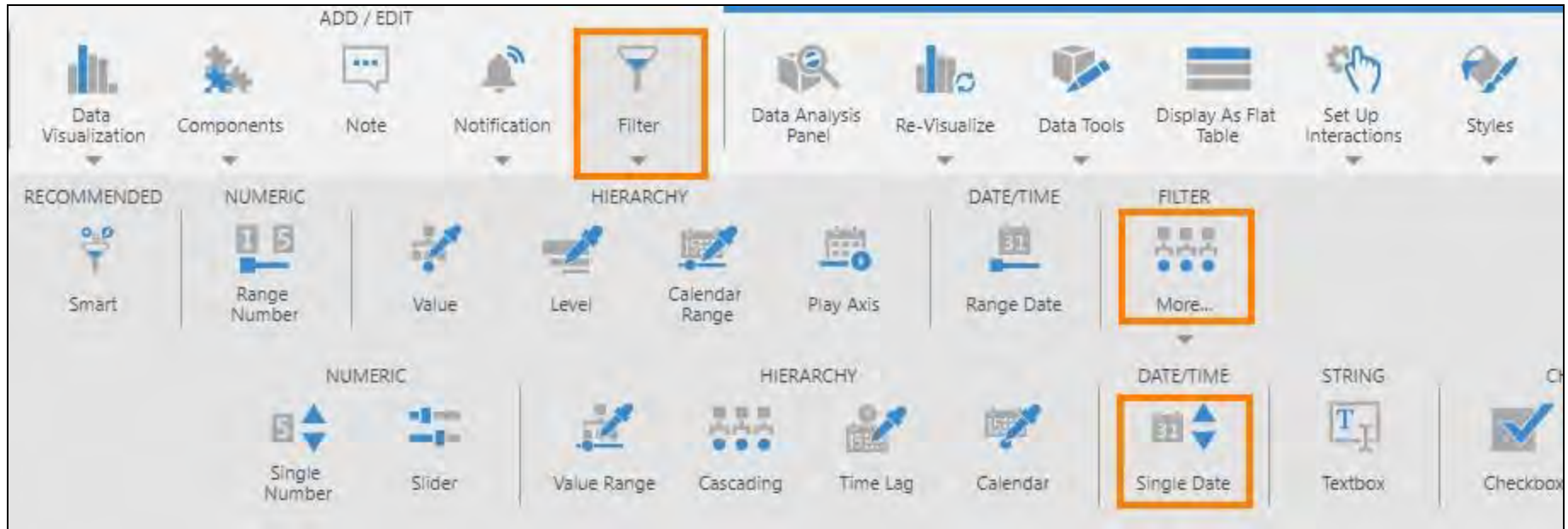
Create or edit a dashboard, then find and expand the data cube we created in the **Explore** window.

In our example, we will drag and drop the measure **ListPrice** to the dashboard canvas, then drag and drop the dimension **ComponentDesc** to **Rows** in the Data Analysis Panel.



The screenshot displays the Logi Symphony v24 interface. On the left, the 'Data Analysis Panel - Metric Set' is visible, showing a 'Metric Set 1' with a 'ListPrice' measure. The 'ROWS' section is currently empty, with a prompt to 'click to add, or drop a hierarchy'. On the right, the 'EXPLORE' panel shows a file tree under 'My Project'. The 'Data Cubes' section is expanded, showing a list of measures: 'BOMLevel', 'ListPrice', 'RecursionLevel', 'StandardCost', and 'TotalQuantity'. The 'Dimensions' section is also expanded, showing 'ComponentDesc', 'ComponentID', and 'ProductAssemblyID'. Two orange arrows point from the 'ListPrice' measure in the 'Data Cubes' list to the 'ListPrice' measure in the 'Data Analysis Panel'. Another orange arrow points from the 'ComponentDesc' dimension in the 'Dimensions' list to the 'ROWS' section of the 'Data Analysis Panel'.

Click **Filter** in the toolbar, click **More**, and then select **Single Date**. This will be used to pass a date value to the **@CheckDate** parameter.



In the pop-up that appears, you can select which measures, dimensions, or and/or stored procedure parameters you want associated with this filter. In our example, **@CheckDate** was checked automatically.

Filter Visualizations - viewParameter 1

Parameters

VIEW PARAMETER

Name
viewParameter 1

Script Name
viewParameter1

ID
43466028-9323-81ae-518c-17036eaea958

Default Value
Data Default

Search Parameters

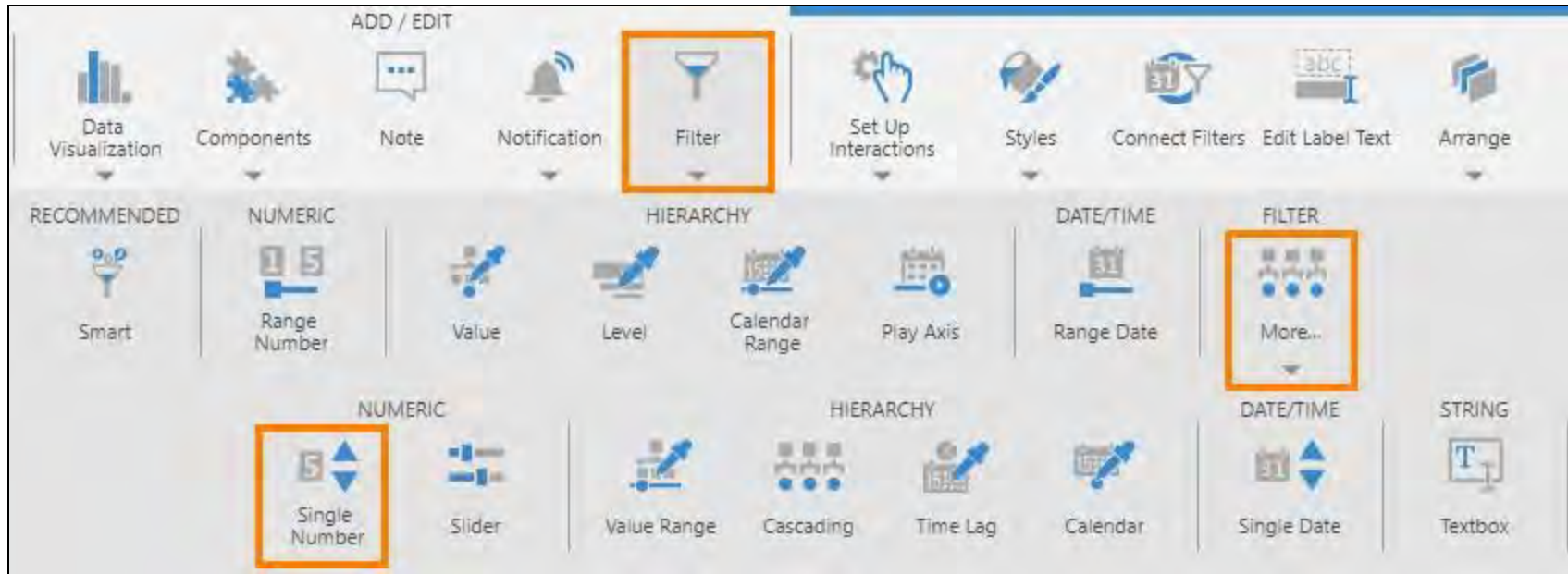
Dashboard 5

table

Metric Set 1

@CheckDate (SingleDateTime)

Click **Filter** in the toolbar, click **More**, and then click **Single Number**. This will pass a numeric value to the **@StartProductID** parameter.



Connect the filter to parameters in the popup that appears. In our example, we select **@StartProductID**.

Filter Visualizations - viewParameter 2

Parameters

VIEW PARAMETER

Name
viewParameter 2

Script Name
viewParameter2

ID
ec355052-9a2c-43ad-df55-c952d8828b3b

Default Value
Data Default

Search Parameters

Dashboard 54

table 1

Metric Set 1

ListPrice

☒ @StartProductID (SingleNumber)

Testing the Dashboard

Click View in the toolbar. Nothing is retrieved with the initial parameter values we have set.

Value

800

Date

2014/30/10

ComponentDesc	ListPrice
Grand Total	2,521.30
Adjustable Race	0.00
BB Ball Bearing	0.00
Bearing Ball	0.00
Blade	0.00
Chain	20.24
Chain Stays	0.00
Chainring	0.00
Chainring Bolts	0.00
Chainring Nut	0.00

Set the filters to values, and observe the updated data retrieved.

Value

800

Date

2014/30/10

ComponentDesc	ListPrice
Grand Total	2,521.30
Adjustable Race	0.00
BB Ball Bearing	0.00
Bearing Ball	0.00
Blade	0.00
Chain	20.24
Chain Stays	0.00
Chainring	0.00
Chainring Bolts	0.00
Chainring Nut	0.00

For more information, see:

- [Add a Relationship Between Tables](#)
- [The Managed Dashboards Data Model](#)
- [Connect to Data and View it on a Dashboard](#)



- Archive of documentation for Logi Symphony v24

- [Automatic Joins and Hierarchies](#)
- [Passing Placeholder Value From Dashboard to Manual MDX Select](#)
- [Using a Bridge Parameter](#)



Passing Placeholder Value From Dashboard to Manual MDX Select

This applies to: *Managed Dashboards, Managed Reports*

This article demonstrates how to pass a value from a dashboard to a placeholder in a manual MDX select.

Example Query

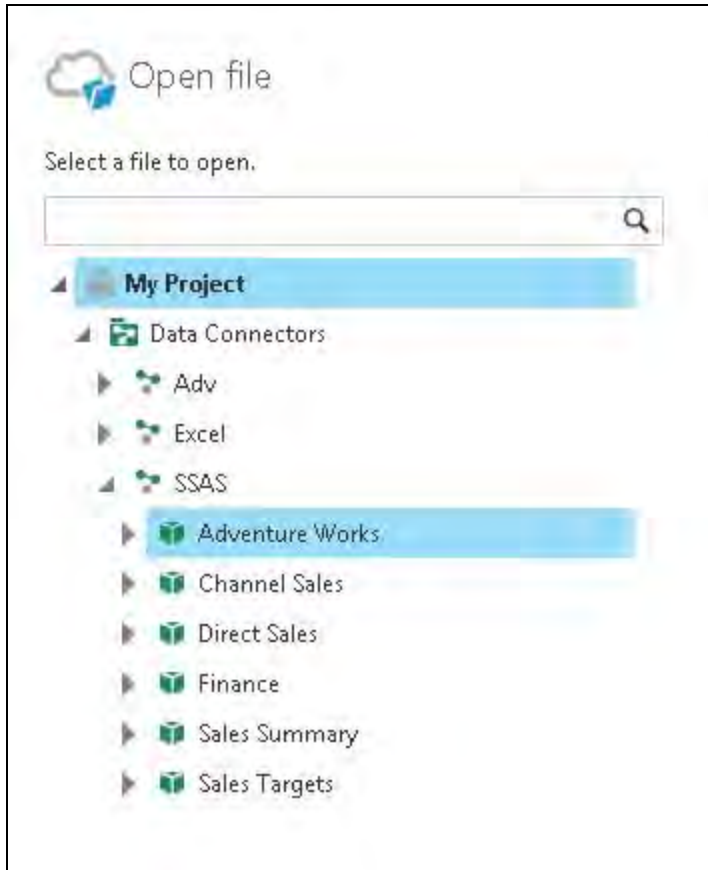
Here is a sample MDX Select with **\$country\$** as a placeholder using SSAS AdventureWorks cube:

```
SELECT {[Measures].[Internet Sales Amount]} DIMENSION PROPERTIES MEMBER_TYPE ON COLUMNS,  
{ DESCENDANTS(StrToMember($country$) ,  
[Customer].[Customer Geography].[State-Province]) } DIMENSION PROPERTIES MEMBER_TYPE ON ROWS  
FROM [Adventure Works]
```

Create a Data Cube

Create a new data cube using the **Manual Query** option, or choose the **Manual Select** option in the toolbar.

In the **Open** dialog, select the particular cube under your data connector you are querying (e.g., Adventure Works).



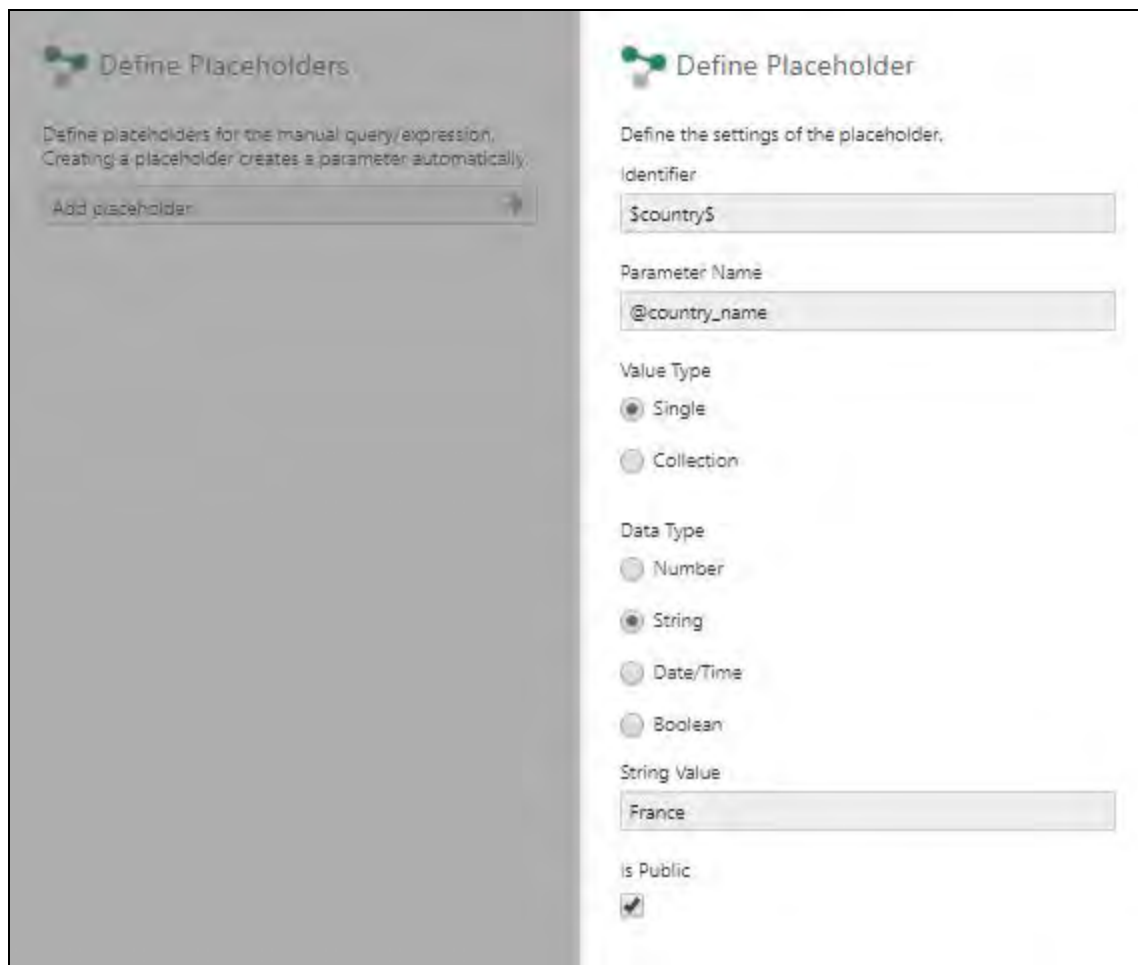
In the **Script Editor** window that appears, add your MDX query that includes a placeholder enclosed in dollar signs (\$) and click **Save**.



The **Define Placeholders** dialog will appear automatically if your query includes a placeholder. Click **Add placeholder**.

The **Identifier** should match your placeholder in the MDX query (e.g., \$country\$). The **Parameter Name** will identify this parameter when you use it on the dashboard.

In our example, we set **Data Type** to **String** and the default **String Value** to **[Customer].[Customer Geography].[Country].&[France]**.




Note: The Value Type **Collection** is not compatible with MDX queries.

After submitting the dialog, your Manual MDX transform should not be red. If it is, there is something wrong with your query, how you have set up the placeholder, or it still needs to be connected to another transform such as the Process Result. You can also open **Data Preview** to confirm data is retrieved using your default placeholder value.

You can configure the Manual MDX transform by double-clicking it, or from its toolbar or context menu. For example, you can change the **Timeout**, make changes to the MDX query, or define additional placeholders.



Manual MDX

The manual MDX transform. The user is responsible for entering a valid select statement for the OLAP data connector of his choice.

Name

Description

ID

Timeout

OLAP Data Provider

AdventureWorksOLAP

The MDX Query Statement

```

1 SELECT {[Measures].[Internet Sales Amount]} DIMENSION PROPERTIES MEMBER_TYPE ON COLUMNS,
2 { { DESCENDANTS(StrToMember($country$) ,
3 [Customer].[Customer Geography].[State-Province]} } DIMENSION PROPERTIES MEMBER_TYPE ON ROWS
4 FROM [Adventure Works]

```

Edit Query

Define Placeholders

Define parameters

Edit output elements



- Archive of documentation for Logi Symphony v24



Note: The greater value between the command timeout on the transform and the command timeout on the data connector will be used.



Note: The Default Query Timeout [application configuration setting](#) determines the initial timeout setting.

Edit the Dashboard

Create or edit a dashboard, and find your data cube in the **Explore** window. Drag its data to the dashboard canvas to select it in a metric set. From the **Filter** option in the toolbar, choose a **Textbox** filter and connect it to your parameter.

C:[Customer].[Customer Geography]

M:[Measures].[Internet Sales Amount]

Grand Total	2,644,017.71
Aaron L. Wright	10,813.63
Abby P. Gonzalez	
Abby P. Rana	
Abby Perez	
Abby Sandberg	
Abigail A. Griffin	
Abigail D. Rivera	
Abigail G. Diaz	

Parameters

VIEW PARAMETER

Name

viewParameter1

Script Name

viewParameter1

Default Value

Value

France

Dashboard2

table 1

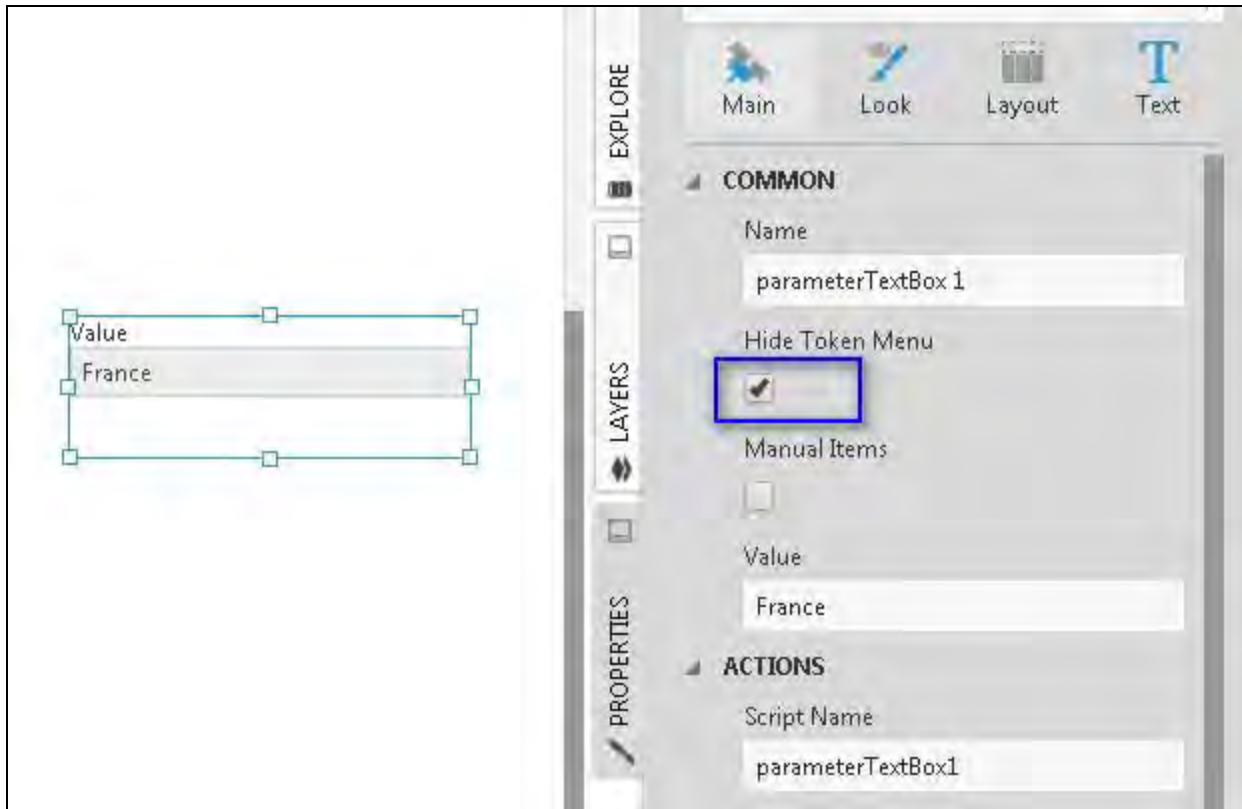
MetricSet1

@country_name (SingleString)

Value

France

In this example, we'll open the **Properties** window for the filter and hide the token menu.



Testing

Click View in the toolbar.

C:[Customer].[Customer Geography]	M:[Measures].[Internet Sales Amount]	Value
Grand Total	2,644,017.71	France
Aaron L. Wright	10,813.63	
Abby P. Gonzalez	616.95	
Abby P. Rana	36.96	
Abby Perez	92.48	
Abby Sandberg	5,948.23	
Abigail A. Griffin	2,518.97	
Abigail D. Rivera	113.96	
Abigail G. Diaz	588.96	
Abigail Ross	1,155.48	
Adam A. Hall	74.98	

After changing the filter value, new data should be retrieved:

C:[Customer].[Customer Geography]	M:[Measures].[Internet Sales Amount]	Value
		Germany
Grand Total	2,894,312.34	
Abby A. Garcia	1,293.33	
Abby Arthur	4,390.69	
Abby E. Chandra	58.98	
Abby Madan	37.27	
Abby Subram	7,494.88	
Abigail H. Russell	12.94	
Abigail R. Henderson	5,918.24	
Adam D. Perez	39.98	
Adam H. Diaz	548.98	
Adrian F. Morris	36.59	

For more information, see:

- [The Managed Dashboards Data Model](#)
- [Connect to Data and View it on a Dashboard](#)



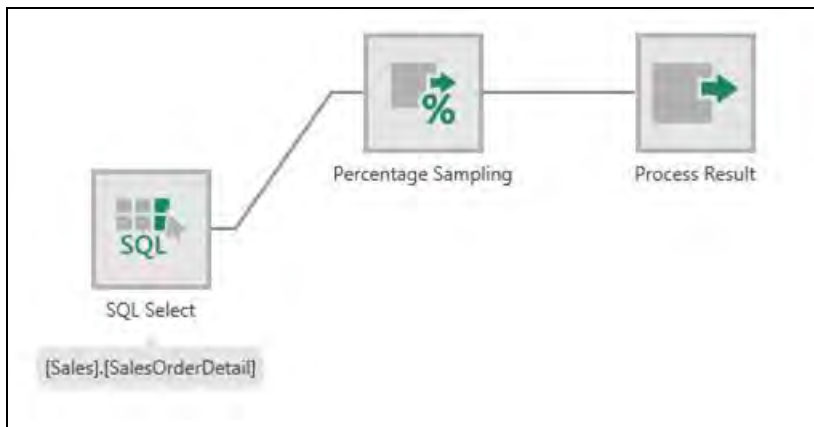
- Archive of documentation for Logi Symphony v24

- [Using Manual Select Placeholders](#)
- [Manual Select](#)
- [MDX Select](#)
- [Passing Parameter Value From Dashboard to Stored Procedure](#)

Percentage Sampling

This applies to: *Managed Dashboards, Managed Reports*

This transform returns the specified percentage out of the full set of input records using random selection.

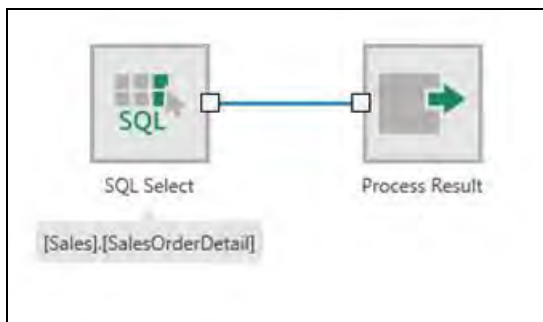


Input

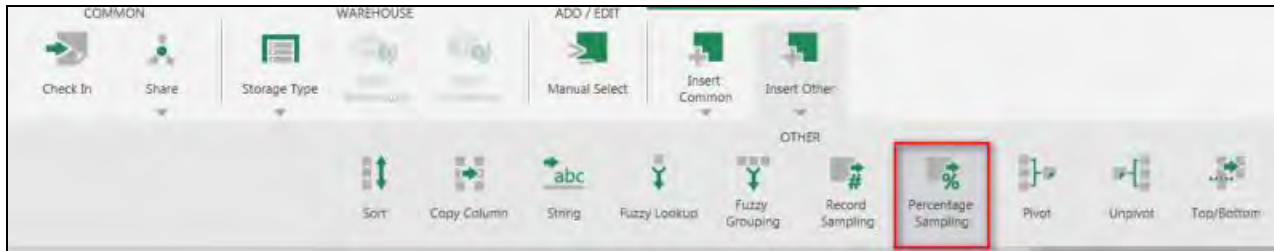
The **Percentage Sampling** transform requires one input transform that has at least one column.

Add the transform

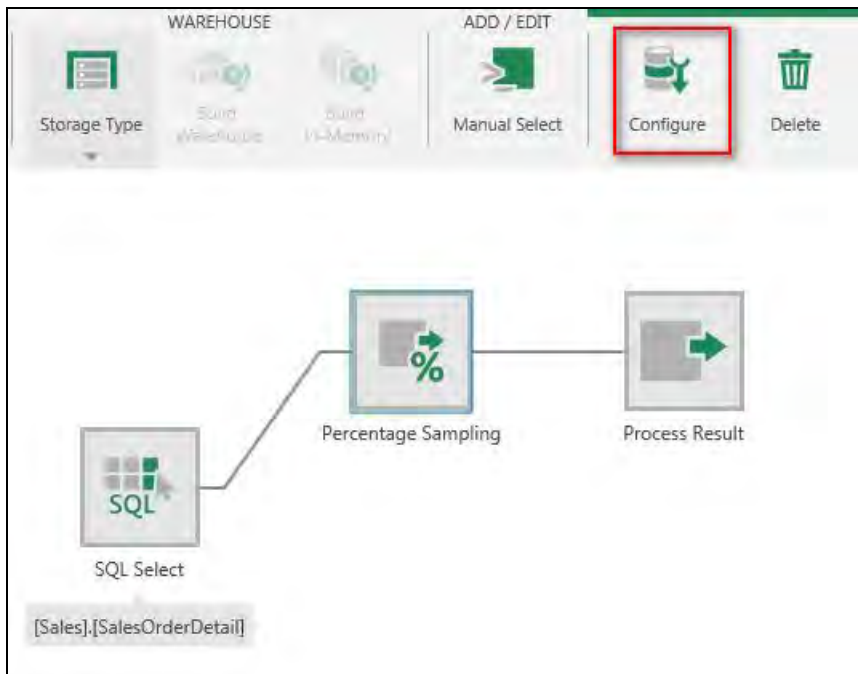
Click to select the connector link between two transforms.



In the toolbar, choose **Insert Other**, then **Percentage Sampling**.



To edit/configure the transform, select it and choose **Configure** in the toolbar.



Configure

In the **Percentage Sampling** transform configuration dialog:



Percentage Sampling

Percentage Sampling node, responsible for returning a portion of the input according to percentage input by user.

[More help...](#)

Sampling Percentage

10

Random Seed

0

☐ SalesOrderID
☐ SalesOrderDetailID
☐ CarrierTrackingNumber
☒ OrderQty
☒ ProductID
☐ SpecialOfferID
☐ UnitPrice
☐ UnitPriceDiscount
☐ LineTotal
☐ rowguid
☐ ModifiedDate

1. Set the **Sampling Percentage** - The percentage of the input records that you want included in the output.
2. Optionally, set the **Random Seed** - An arbitrary number no greater than 2,147,483,647, used to initialize a pseudorandom number generator. If this is set to zero, the current time is used to randomly determine which records will be sampled; otherwise, the same records will be sampled given the same input data and sampling percentage.
3. Select the columns to be included in the output.

As with other transforms, these settings can be turned into a parameter to be changed outside of the transform or the data cube (e.g., with a filter on a dashboard) by clicking **Define parameters**.



- Archive of documentation for Logi Symphony v24

Output

Submit the configuration dialog, and with the transform selected, open the **Data Preview** to view the output.

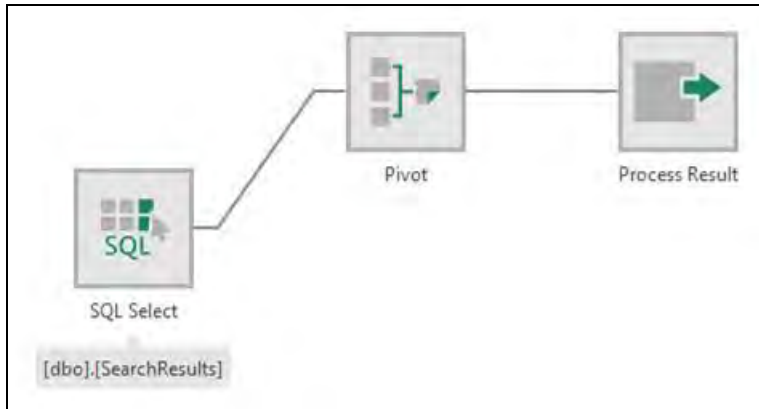
For more information, see:

- [Using Transforms](#)
- [Record Sampling](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Pivot

This applies to: *Managed Dashboards, Managed Reports*

The Pivot transform allows you to create new columns out of rows in your input that contain a corresponding set of values (opposite of [Unpivot](#)).



Input

The **Pivot** transform requires one input transform that has at least three columns.

For example, the input could be a SQL Select transform that corresponds to the following table:

	ID	Source	Type	Value
▶	1	Search Engine 1	Hits	1000
	2	Search Engine 1	Unique IPs	800
	3	Search Engine 2	Hits	500
	4	Search Engine 2	Unique IPs	479
★	NULL	NULL	NULL	NULL

Configure

In the Pivot configuration dialog, you need to configure the following settings:

Grouping Elements

☐ ID
 ☒ Source
 ☐ Type
 ☐ Value

Pivot Element

Type

Value Element

Value

Automatically Generate Column Names

☐

The Names Of The New Columns The Pivot Transform Will Generate (Comma Delimited).

Hits,Unique IPs

Ignore Case

☐

Check the **Grouping Elements** that can be used to group a set of rows together (e.g., the **Source** column in our example), which will be turned into a single record containing a value for each new column. The selected column(s) will be part of the output as well.

Set **Pivot Element** to the names in the rows that will correspond with the names of the new columns (e.g., the **Type** column in our example).

The **Value Element** should be set to the values that will be moved from separate rows into the values of the new columns (e.g., the **Value** column in our example).

To automatically generate the new column names from your data, select **Automatically Generate Column Names**, and the field below will be automatically filled in with names after you close the dialog.

Grouping Elements

☐ ID
 ☒ Source
 ☐ Type
 ☐ Value

Pivot Element

Type

Value Element

Type

Automatically Generate Column Names

☒

Sort The Automatically Generated Column Names

☐

The Names Of The New Columns The Pivot Transform Will Generate (Comma Delimited).

Ignore Case

☐

When automatically generating column names, select **Sort The Automatically Generated Column Names** before submitting the dialog if you want the new columns to be sorted alphabetically.

If you prefer to enter the names of the new columns yourself, they must match the values of the **Pivot Element** in your data (e.g., **Hits**, **Unique IPs** in our example).

Output

The output of the Pivot transform consists of:

- A single row for each Grouping Elements value or value combination.
- New columns that correspond to the values of the Pivot Element column.

- The values for the new columns that came from the Value Element column.

The figure below illustrates the output from the Pivot transform as applied to our example input data.

Hits	Unique IPs	Source
1,000.00	800.00	Search Engine 1
500.00	479.00	Search Engine 2

For more information, see:

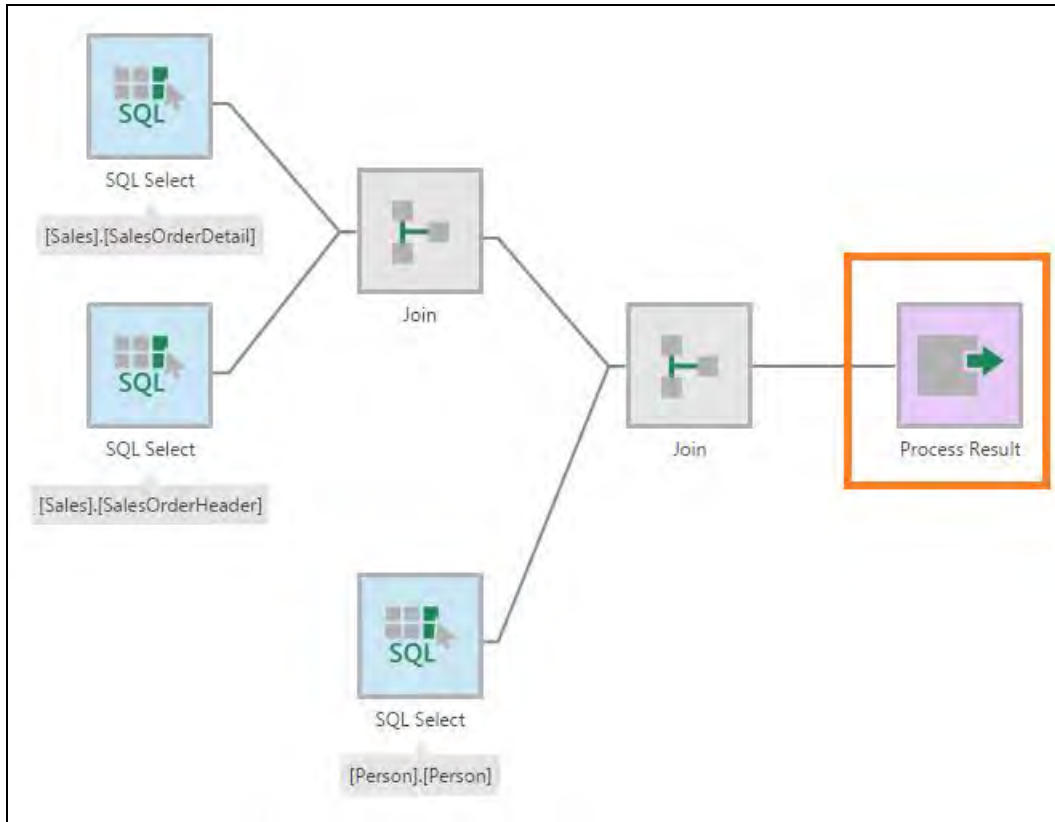
- [Using Transforms](#)
- [Unpivot](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Process Result

This applies to: *Managed Dashboards, Managed Reports*

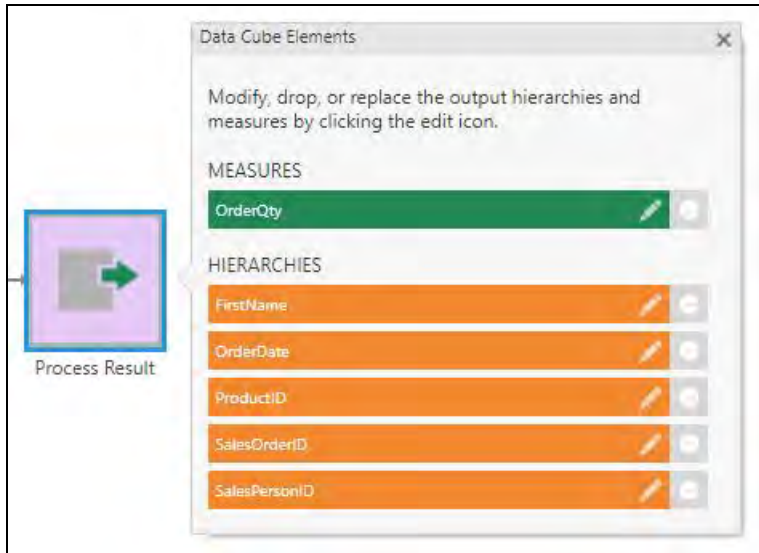
The **Process Result** transform represents the final output or result of the data cube (ETL process).

This transform doesn't do any data processing but it allows you to configure the measures and hierarchies that will be made available to downstream items such as metric sets.



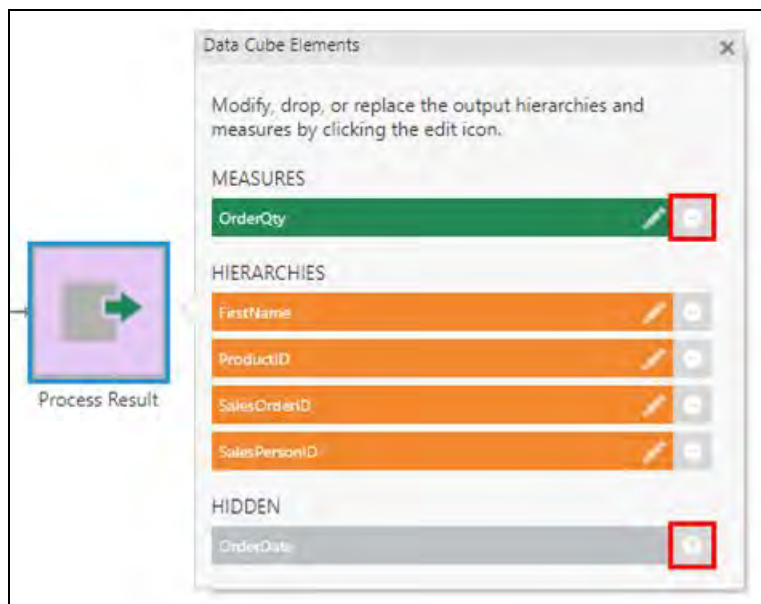
Data Cube Elements

Select the **Process Result** transform to see the **Data Cube Elements** panel, which shows the list of output measures and hierarchies for this data cube (ETL process). Measures and hierarchies are color-coded the same way as in metric sets.



Hide Elements

You can hide a measure or hierarchy from the output by clicking its gray **minus** icon on the very right. The measure or hierarchy will be listed under a **Hidden** section and you can easily unhide it by clicking its **plus** icon.



A hidden measure or hierarchy is hidden from the data cube output but will still be available for subsequent linking in hierarchy keys (e.g., if you need to use the data cube later to define a hierarchy).

Edit Elements

You can access additional options related to a specific measure or hierarchy by clicking its green or orange tile to edit it.

The **Output Element** dialog will open, where you can change the **Caption**, set a **Description**, and customize various settings including [supported aggregators](#) or a measure's [number formatting](#).



Output Element

[more help](#)

See and edit the settings of the data cube's output measure or hierarchy.

This output element is currently a hierarchy.

[Switch to a measure.](#)

Current Hierarchy

Implicit Hierarchy (generated)

Select a hierarchy or level to use as a replacement 

Caption

ProductID

Unique Name

ProductID

Description

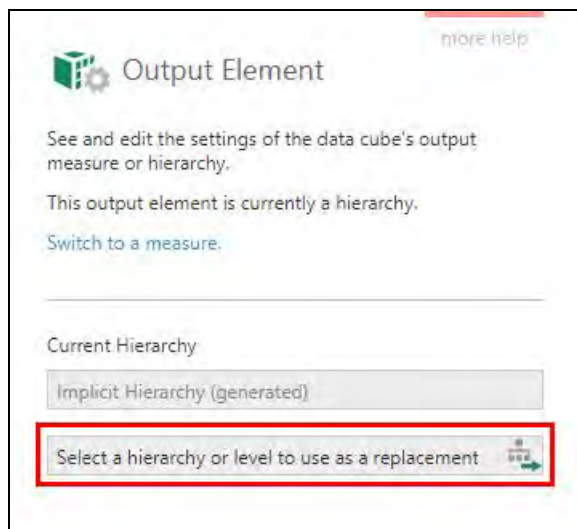
Product sold to customer. Foreign key to Product.ProductID.

For elements that could be either a measure or a hierarchy, you can switch between them by clicking [Switch to a hierarchy](#) or [Switch to a measure](#). Non-numeric type data can also be switched to a measure in order to perform a count or distinct count of that element's values, similar to adding a [hierarchy count measure](#) in a metric set.

Replace With a Hierarchy

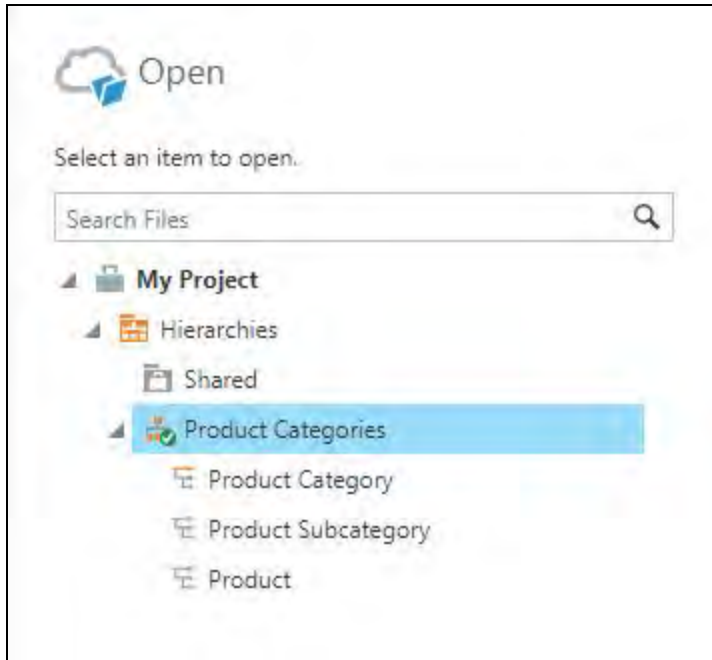
Simple columns of text, dates/times, or numeric IDs each automatically become an 'implicit' or generated hierarchy when output by the cube and used in a [metric set](#). You can replace such columns with a [hierarchy](#) you've created and set up with features such as multiple levels and custom sorting, or with a [time dimension](#) hierarchy for date/time data. For example, you could replace a Product/D column with an existing multi-level Product hierarchy.

To do this, click **Select a hierarchy or level to use as a replacement** in the **Output Element** dialog.



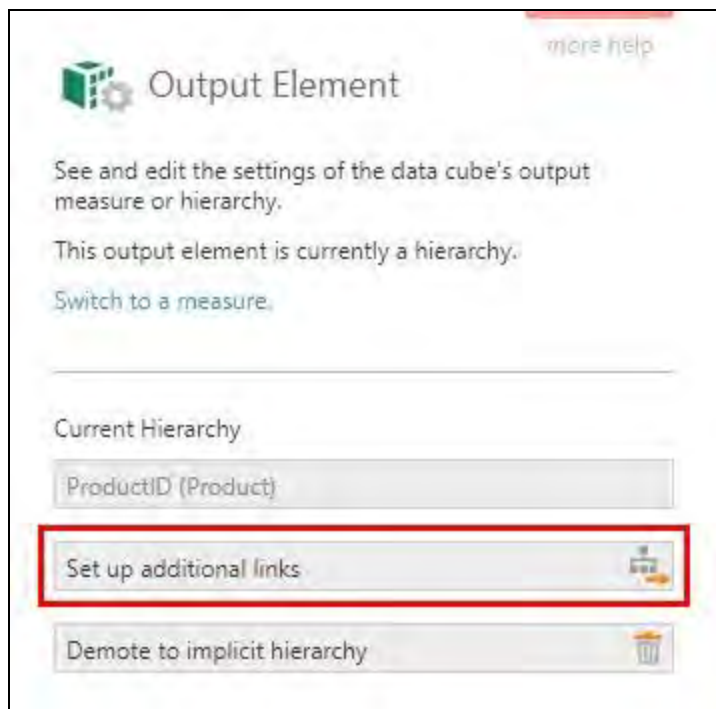
In the **Open** dialog that appears, select your hierarchy from the **Hierarchies** folder, or for time dimensions, click to expand one from the **Time Dimensions** folder and select one of its hierarchies.

Selecting the hierarchy will link the column to the lowest level of the hierarchy by default, or you can expand the hierarchy and select a specific hierarchy level instead. For text or numeric columns, you must choose the level with keys that were set up to match your column's values.



Click the submit button at the bottom when finished.

Returning to the **Output Element** dialog, you can optionally click to **Set up additional links** between the hierarchy and the cube's data at other levels if your data also matches with their keys, which can improve performance. You should also set up additional links if you chose a hierarchy level that has multiple keys so that each key is linked to your cube's data.



In the dialog that appears, select a **Hierarchy Level** from the drop-down. To add a link, drag an item under **Hierarchy Keys** and drop it onto the corresponding item under **Result Elements**.



Set Up Hierarchy Links

Set up how each individual level of the hierarchy is linked to the backing structure.

Hierarchy Level

Product Category

Hierarchy Keys

ProductCategoryID

Result Elements

SalesOrderID
SalesOrderDetailID
CarrierTrackingNumber
OrderQty
ProductID
SpecialOfferID
UnitPrice
UnitPriceDiscount
LineTotal
rowguid
ModifiedDate

All Member Caption

The **"All" Member Caption** can be set to customize the text identifying the top node of the hierarchy. If this value is not set, it will display the value located in the localization file for the current language.

Caption

ProductID (Product)

Unique Name

ProductID

Description

"All" Member Caption

All Products

Supported Aggregators

Count

This caption can appear in visualizations and filters connected to this hierarchy.

Value

All Products

Search Values

- ☒ All Products
 - ☒ Accessories
 - ☒ Bikes
 - ☒ Clothing
 - ☒ Components

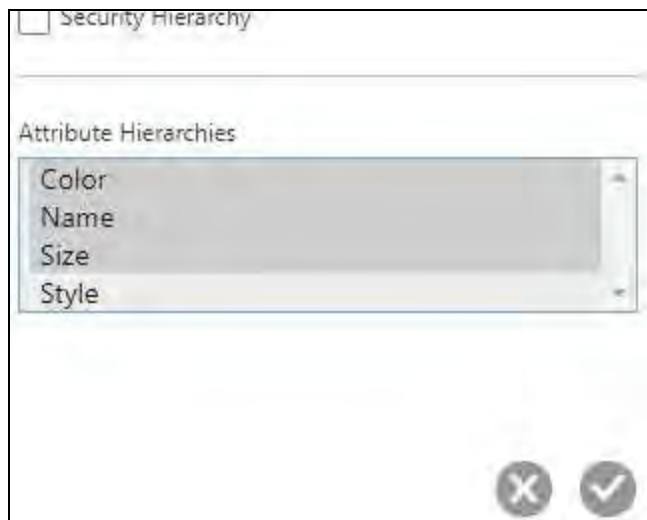


Note: Implicit date hierarchies used with a Calendar filter will use the caption from default time dimension.

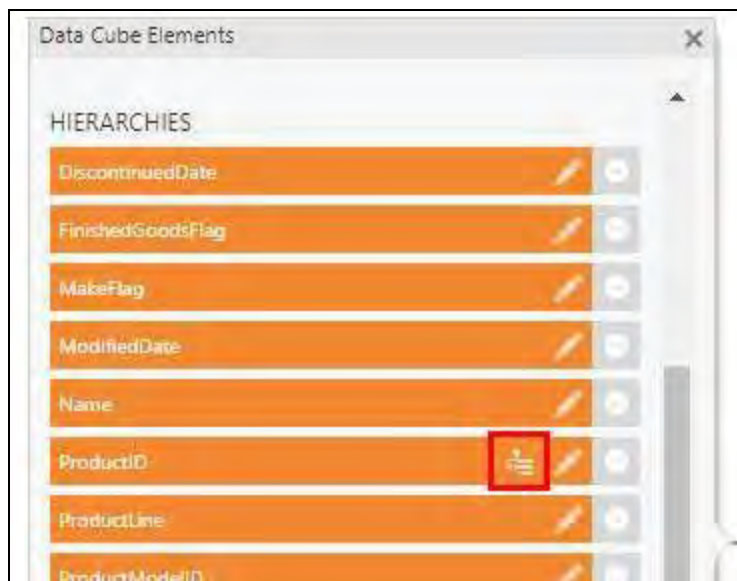
Attribute Hierarchies

If the element is an implicit hierarchy, you can define other elements that should appear as attributes for this hierarchy, the same way as attributes defined in a [hierarchy](#) created from the main menu.

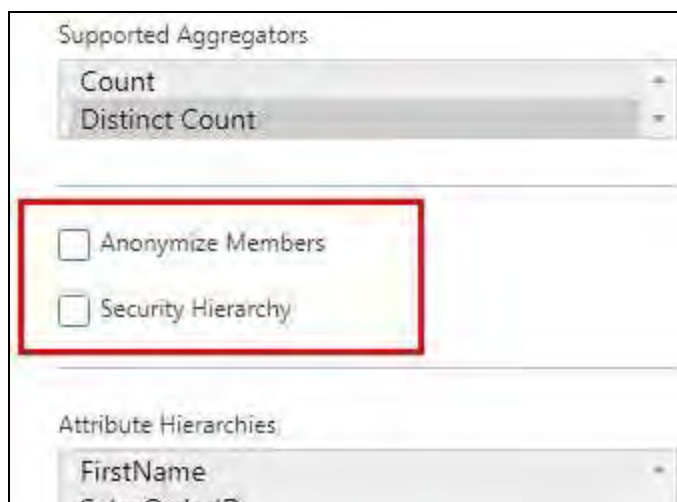
You can select one or more elements under Attribute Hierarchies: to select multiple, click and drag, or hold the Ctrl or Shift keys while selecting.



After submitting the dialog, the hierarchy with attributes defined will appear with a special icon, and the elements selected as attribute hierarchies will no longer be listed as separate hierarchies in the **Data Cube Elements** popup.



Security Options



The **Anonymize Members** option is available for implicit hierarchies (except for DateTime columns), and when enabled will generate a new value to replace each unique original value in the column so that it does not directly identify the original. The generated values may change over time.

OrderDate	Customer	TotalDue
1/16/2012	Customer 2564	2,738.5657
1/16/2012	Customer 2603	2,694.0563
1/16/2012	Customer 1164	2,634.3974
1/16/2012	Customer 2975	2,618.8058
1/16/2012	Customer 994	2,574.6279
1/16/2012	Customer 765	2,566.1194

The **Security Hierarchy** option allows you to accomplish row-level security by filtering the data by that hierarchy's values where they must match the custom attribute value for each logged-in user. For details, see [Using a security hierarchy to filter data by user](#).

Supported Aggregators

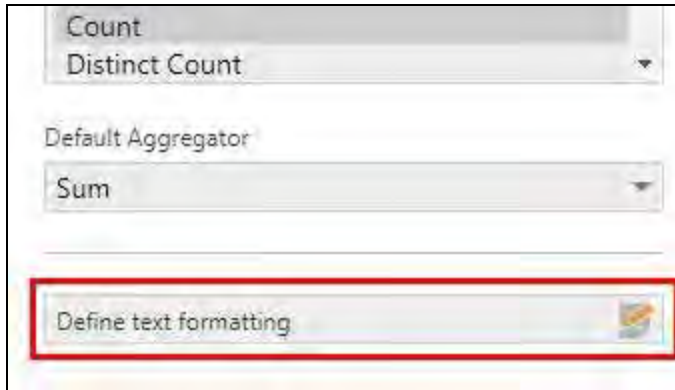
Both numeric data and non-numeric data elements have a **Supported Aggregators** setting, which determines which aggregators are available when this data is added under **Measures** in a metric set. To select multiple with a mouse, click and drag across the options, or click while holding the Ctrl/ or Shift keys on your keyboard. The number of available aggregators is a consideration for the amount of memory consumed when using in-memory storage.

You can drag a hierarchy or non-numeric to [Measures](#) in the Data Analysis Panel to perform a count or distinct count of its values, so the Supported Aggregators setting determines how many of these two are available.

Formatting Measures

The [formatting options](#) available on a metric set can also be defined ahead of time for a measure in a data cube, which will be the initial settings used whenever that measure is added to a metric set.

Click Define text formatting to open the Configure Formats dialog, which contains a variety of options for formatting numbers as text. For details on these, see [Formatting Text](#).



Configure

Double-click the Process Result transform to see its configuration dialog, or select **Configure** from the toolbar or context menu.

Uncheck any columns which you don't need to be part of the data cube output. This will completely remove the column from the data cube and will make the column unavailable for subsequent linking in hierarchy keys (e.g., if you need to use the data cube to build a [hierarchy](#)).

Help 

Process Result

The transform is used to represent the final process output, which translates to the data cube entity.

Name

Color

Description

ID

Settings





LineTotal

Double



Margin

Double



Product ProductID

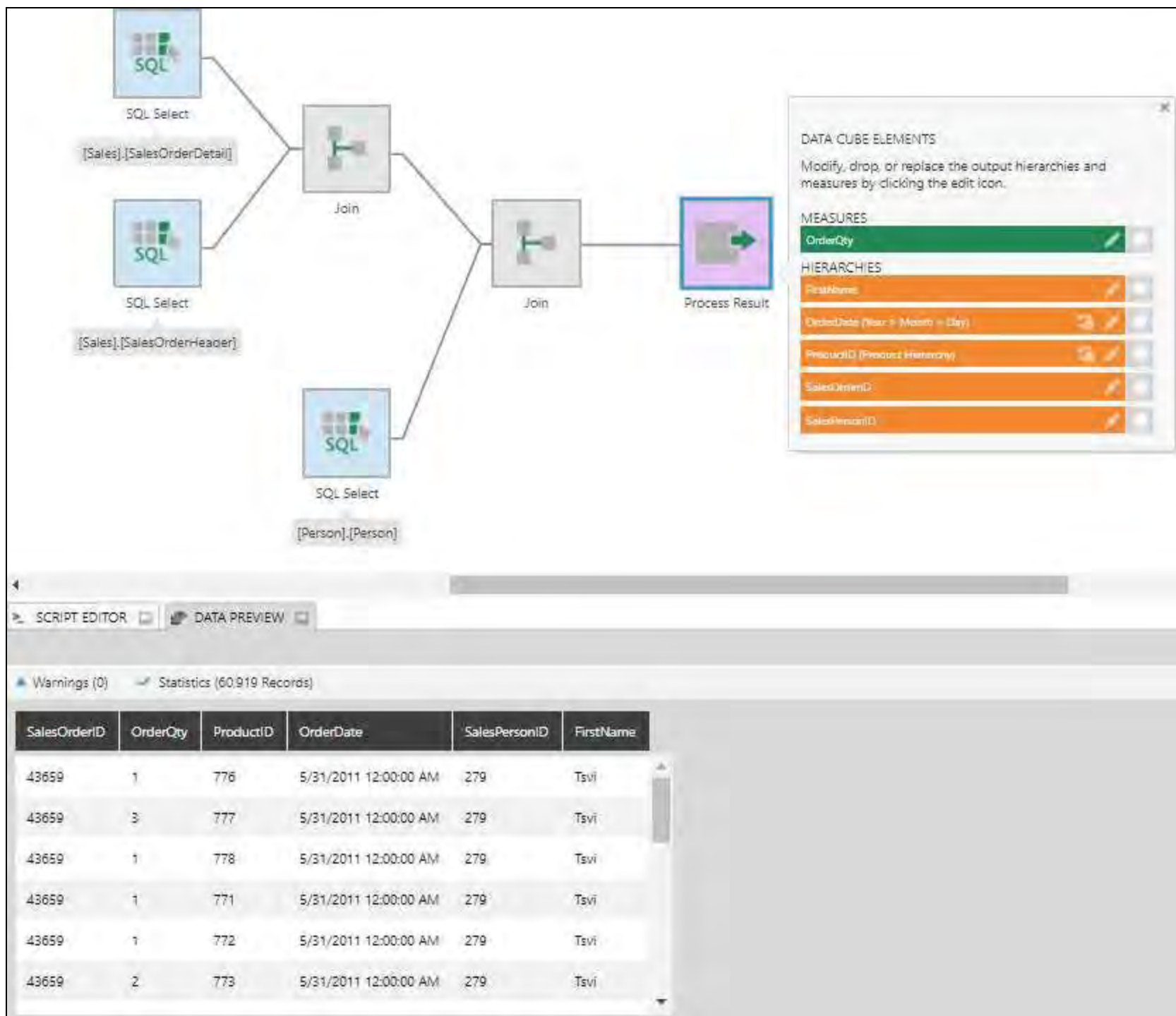
Int32



- Archive of documentation for Logi Symphony v24

Output

Select the Process Result transform on the canvas and then click Data Preview to see a preview of the data output generated by this data cube.



The screenshot displays the Logi Symphony v24 interface. At the top, a data pipeline is visible, consisting of three SQL Select components connected by two Join components, leading to a Process Result component. The SQL Select components are labeled with their respective data sources: [Sales].[SalesOrderDetail], [Sales].[SalesOrderHeader], and [Person].[Person].

On the right side, a 'DATA CUBE ELEMENTS' panel is open, showing the following elements:

- MEASURES:** OrderQty
- HIERARCHIES:**
 - FirstName
 - OrderDate (Year > Month > Day)
 - ProductID (Product Hierarchy)
 - SalesOrderID
 - SalesPersonID

Below the pipeline, the 'SCRIPT EDITOR' and 'DATA PREVIEW' tabs are visible. The 'DATA PREVIEW' tab is active, showing a table with the following data:

SalesOrderID	OrderQty	ProductID	OrderDate	SalesPersonID	FirstName
43659	1	776	5/31/2011 12:00:00 AM	279	Tsvi
43659	3	777	5/31/2011 12:00:00 AM	279	Tsvi
43659	1	778	5/31/2011 12:00:00 AM	279	Tsvi
43659	1	771	5/31/2011 12:00:00 AM	279	Tsvi
43659	1	772	5/31/2011 12:00:00 AM	279	Tsvi
43659	2	773	5/31/2011 12:00:00 AM	279	Tsvi



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)
- [Formatting Text](#)
- [Automatic Joins and Hierarchies](#)
- [Symphony Metric Sets](#)

Python Analysis

This applies to: *Managed Dashboards, Managed Reports*

The Python Analysis transform lets you write scripts using the Python programming language to perform statistical and predictive analysis on data or perform other data manipulation. This is similar to the [Python Data Generator](#) transform except that it operates on input data from a preceding transform instead of generating its own output.

To learn more about the Python language, see python.org.



Note: If your organization did not install Symphony, Python transforms may not be available depending on how your instance is hosted.

Setup

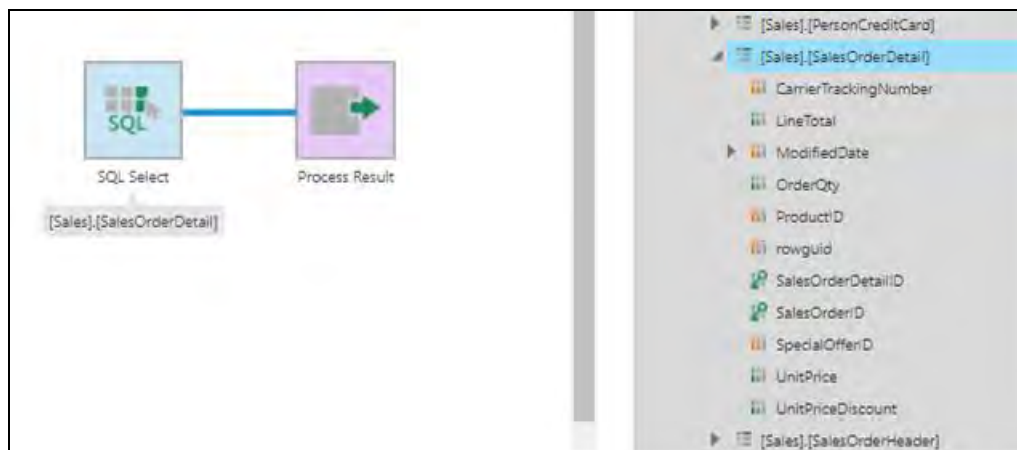
The Python programming environment must be installed on the Symphony server for you to use the Python Analysis transform.

This is normally installed for you automatically as a prerequisite. See [Install Python](#) for more details, and for examples of how to install additional Python packages to use in your transforms.

Input

The Python Analysis transform requires one or more input transforms (each of which may produce multiple columns of data).

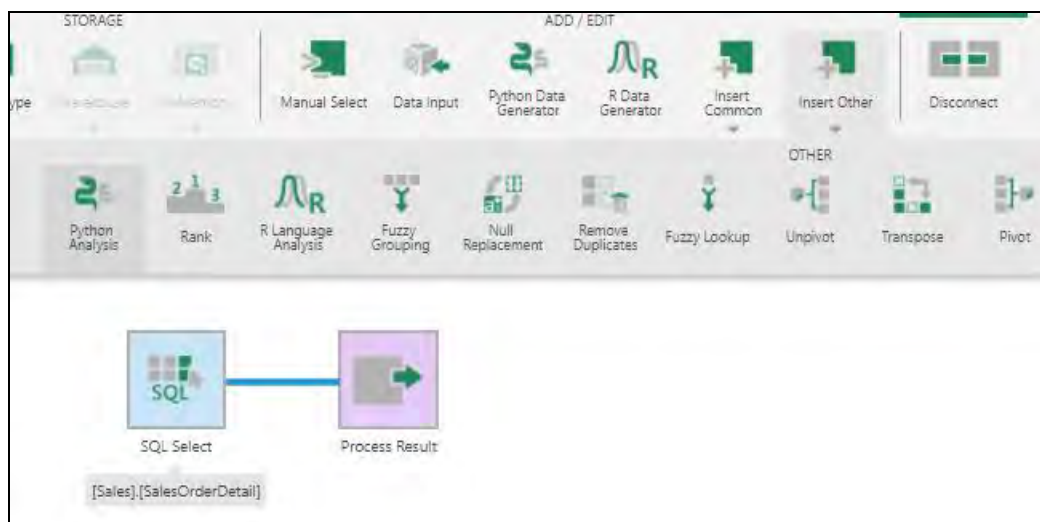
For example, the input could be a [SQL Select](#) transform that reads data from the AdventureWorks table **[Sales].[SalesOrderDetail]**.



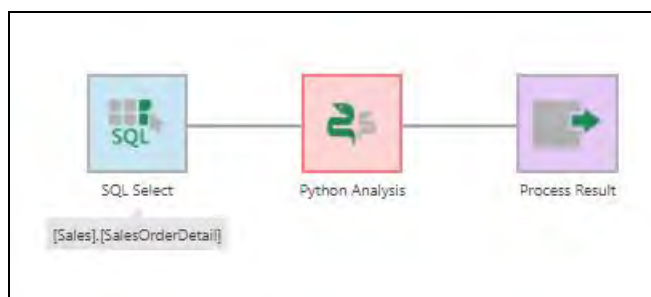
Add the Transform

To add this transform to an existing data cube, first select the connection link between two connected transforms.

Go to the toolbar, click **Insert Other**, and then select **Python Analysis**.



The Python Analysis transform is inserted between the two transforms and appears red because it needs further configuration.



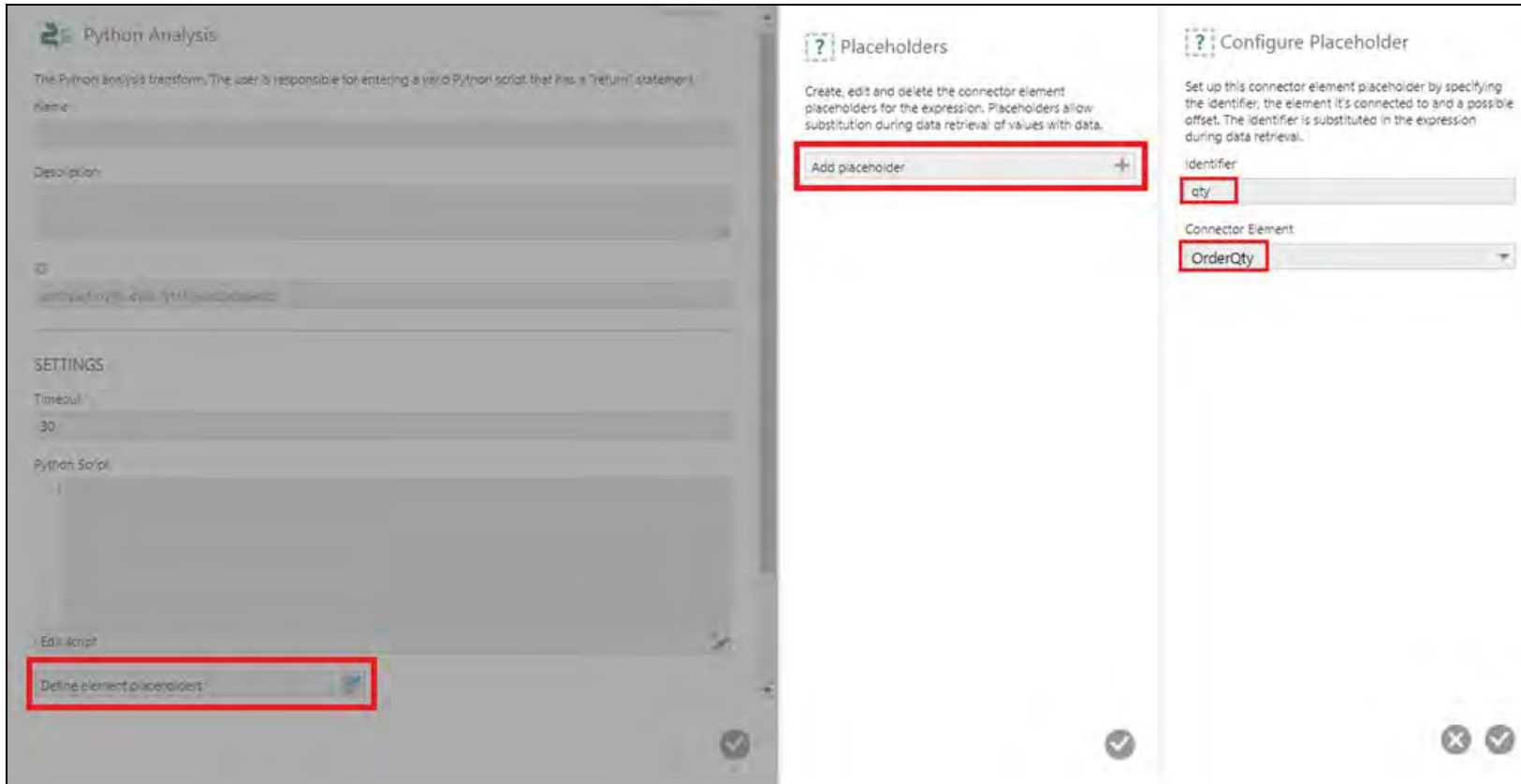
Configure the Transform

Double-click the Python Analysis transform (You can also select the **Configure** option from the context menu or the toolbar).

In the configuration dialog for the transform, the objective is to decide which input column(s) you want to analyze or apply processing to, and then enter a script that performs this analysis or processing on the input data.

For example, suppose you want to calculate the total sum of the **OrderQty** column from the input transform, and send this sum as the output.

To begin, click **Define element placeholders** in the configuration dialog. Click **Add placeholder**. In the **Identifier** text box, enter a variable name (for example, **qty**) for the input column which you will use in the Python script. Use the drop-down to select the corresponding input column (for example, **OrderQty**).



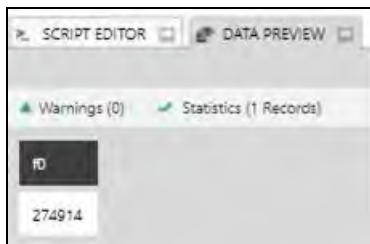
You can now write a Python script that references the **qty** variable by enclosing it between dollar sign characters. For example:

```
return sum($qty$)
```



Note: You can use the **Print** command to print variables to the [application logs](#) for debugging. To view these entries, ensure that the Log Filter [configuration setting](#) for Data Cubes is set to Information.

This script calculates the sum of the **OrderQty** input column and returns the result as the output.



Note: Click **Edit script** to open the script in the Script Editor window for more space and [other helpful features](#).

Select the **Use Pooling** option to reuse Python processes between execution of Python scripts to improve performance compared to starting a new process each time. There are corresponding [configuration settings](#) for Symphony administrators in the **Python Pool** category to manage the number of processes and their memory consumption.

Output

The output of the Python Analysis transform depends on the Python script it is configured with. It can be a single value, a column of values, or multiple columns.

Example Python Script

The following example uses the **polyfit** function to find the least squares polynomial fit for the input columns.

Setup

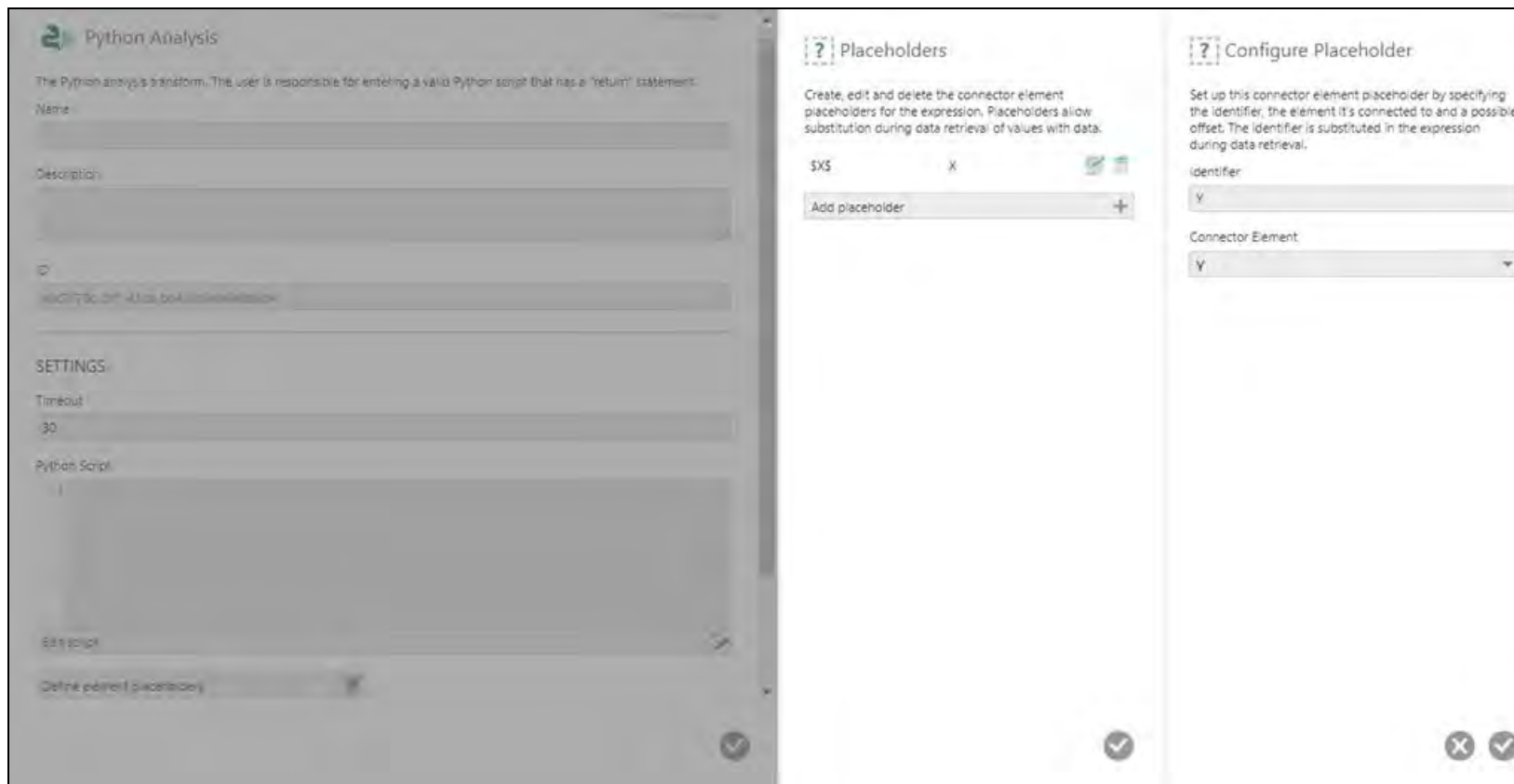
This particular example relies on the NumPy package in Python and some input values:

- If the NumPy package is not installed, the Symphony server administrator can install it as described in the article [Install Python](#).
- For this example, in a new data cube, create a [Data Input transform](#) with the following two columns using the Double data type:
 - X: [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
 - Y: [0.0, 0.8, 0.9, 0.1, -0.8, -1.0]

	X	Y
1	0.0	0.0
2	1.0	0.8
3	2.0	0.9
4	3.0	0.1
5	4.0	-0.8
6	5.0	-1.0

Define Placeholders

Add the Python Analysis transform and double-click to configure it. Click **Define element placeholders**, and then define the two placeholders **X** and **Y**.



Python Analysis

The Python analysis transform. The user is responsible for entering a valid Python script that has a "return" statement.

Name:

Description:

SETTINGS

Timeout: 30

Python Script:

Edit script

Define element placeholders

Placeholders

Create, edit and delete the connector element placeholders for the expression. Placeholders allow substitution during data retrieval of values with data.

SX\$ X

Add placeholder +

Configure Placeholder

Set up this connector element placeholder by specifying the identifier, the element it's connected to and a possible offset. The identifier is substituted in the expression during data retrieval.

Identifier: Y

Connector Element: Y

Create the Script

To fit the input values to a third-power polynomial, add the following script to the Python Analysis transform:

```
import numpy as np

Z = np.polyfit($X$, $Y$, 3)
P = np.poly1d(Z)

return ($X$, $Y$, P($X$))
```

The generated output is in the form of a table consisting of the X and Y inputs, as well as the polynomial values for each entry.

#0	f1	f2
0	0	-0.0396825396825446
1	0.8	0.926984126984127
2	0.9	0.788888888888891
3	0.1	0.0682539682539707
4	-0.8	-0.712698412698409
5	-1	-1.03174603174603

Adjust the Column Names and the Output

Configure the transform again and click **Edit output elements**.

Edit each output element and provide a relevant column name.



Python Analysis

The Python analysis transform. The user is responsible for entering a valid Python script that has a "return" statement.

Name

Description

ID

SETTINGS

Timeout

Python Script

```

import numpy as np
Z = np.polyfit(X, Y, 4)
P = np.polyval(Z)
return f(X), Y, P

```

Edit script

Define element placeholders

Parameters

Output elements

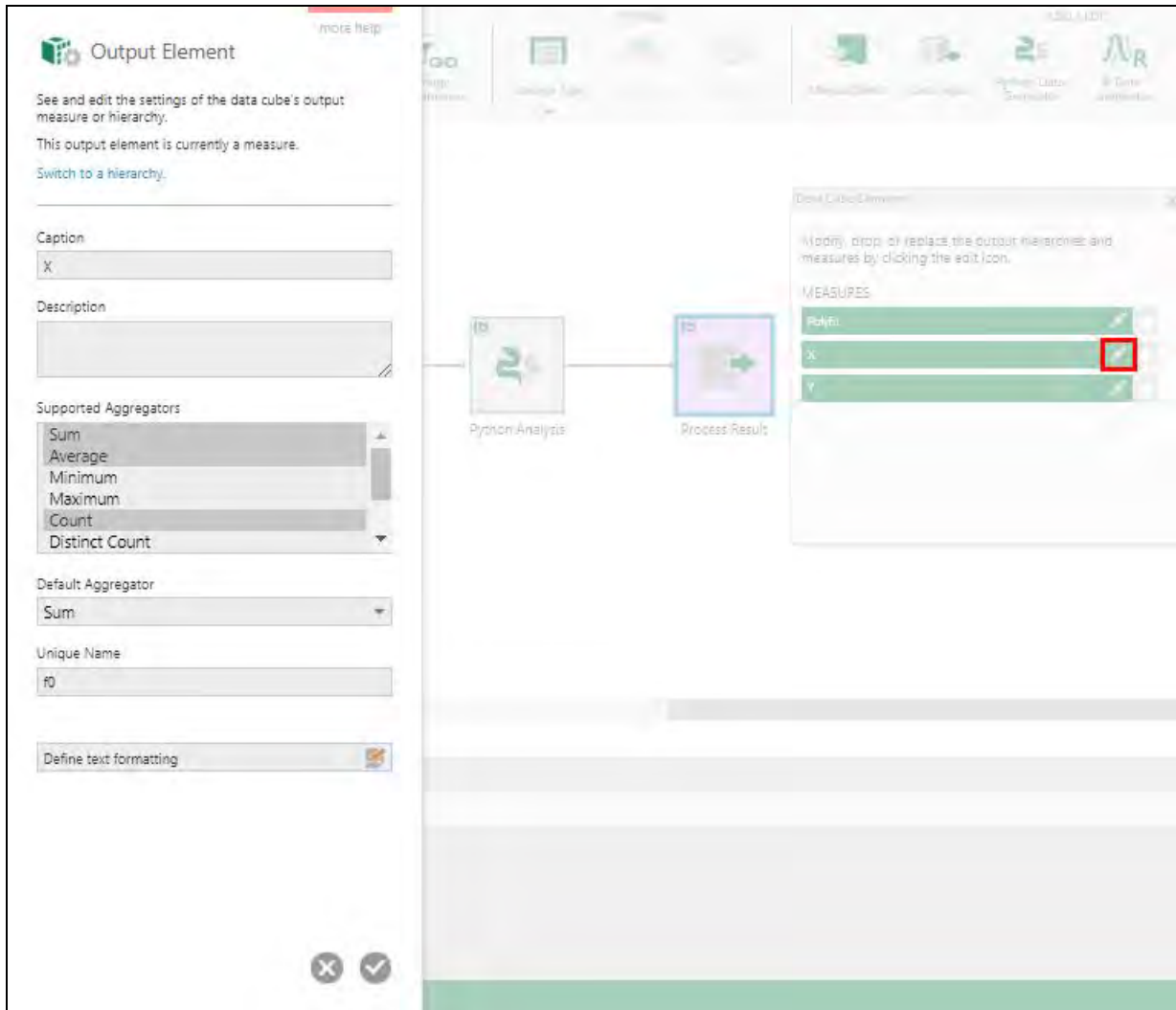


Output Elements

Edit the transform's output elements to change their name.

Name	Connector Name	Input Element Name
X		
Y		
Polyfit		

Select the **Process Result** transform and edit the **X** measure. Click **Switch to hierarchy** and then select **Count** from the supported aggregators.



Output Element

See and edit the settings of the data cube's output measure or hierarchy.
This output element is currently a measure.
[Switch to a hierarchy.](#)

Caption
X

Description

Supported Aggregators
Sum
Average
Minimum
Maximum
Count
Distinct Count

Default Aggregator
Sum

Unique Name
r0

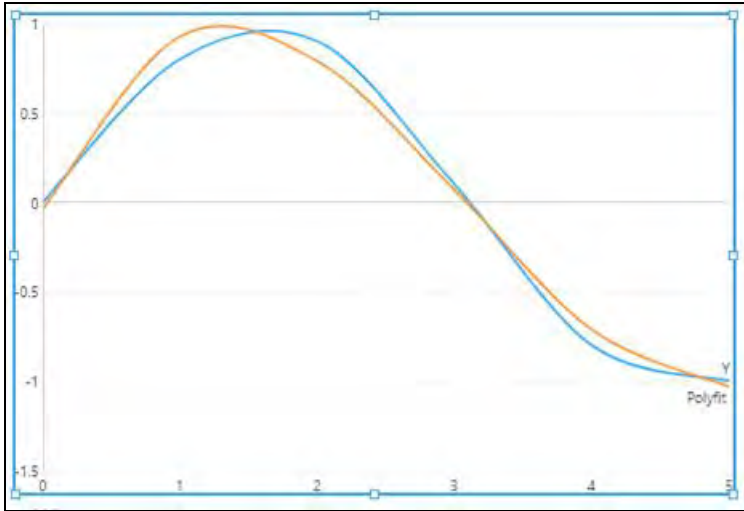
Define text formatting

Process Result

MEASURES
Polyfit
X
Y

View the Result

On a new dashboard, drag all three elements from the data cube and Re-Visualize as a Curved Line.



Geocoding Addresses

A Python Analysis transform is one way you can convert street addresses in your data into latitude & longitude coordinates (also called 'geocoding'), which can then be plotted on a [map visualization](#).

The following example uses the [geopy](#) package in Python to connect to the [Mapbox](#) geocoding service (using the 'temporary' geocoding API, meaning results are on-demand and not stored). The administrator of the Symphony server may need to [install](#) this package first, entering `pip install geopy` for example in a command prompt running as administrator.

After [adding](#) a Python Analysis transform connected to your data source, click to [configure](#) it and click Define element placeholders. Add a placeholder with Identifier set to Address and Connector Element set to your address data.



Configure Placeholder

Set up this connector element placeholder by specifying the identifier, the element it's connected to and a possible offset. The identifier is substituted in the expression during data retrieval.

Identifier

Connector Element

You can also add other placeholders if you have other data you want to return and use in the transforms that follow this one.

Enter **Python Script** like the following. Edit the `api_key` value to your [Mapbox access token](#), and edit the last line if you have additional placeholders you want to return:

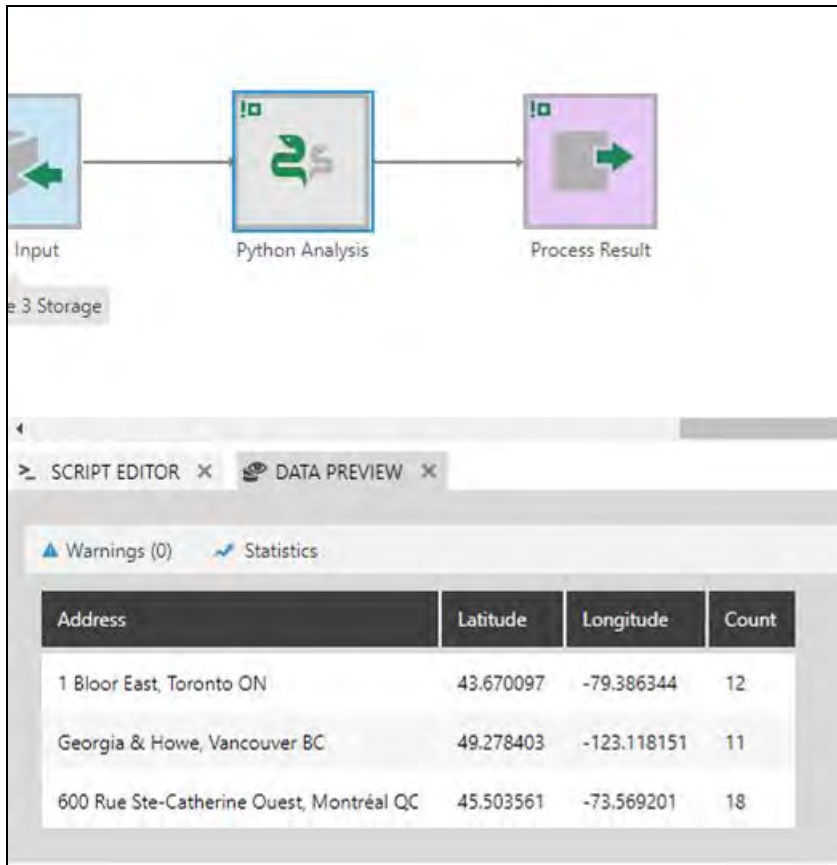
```
import pandas
from geopy.geocoders import MapBox

geolocator = MapBox(api_key="<your MapBox API access token>")
locations = list(map(lambda address: geolocator.geocode(address), $Address$))

latitudes = list(map(lambda location: location.latitude if location != None else None, locations))
longitudes = list(map(lambda location: location.longitude if location != None else None, locations))

return pandas.DataFrame({"Address": $Address$, "Latitude": latitudes, "Longitude": longitudes})
```

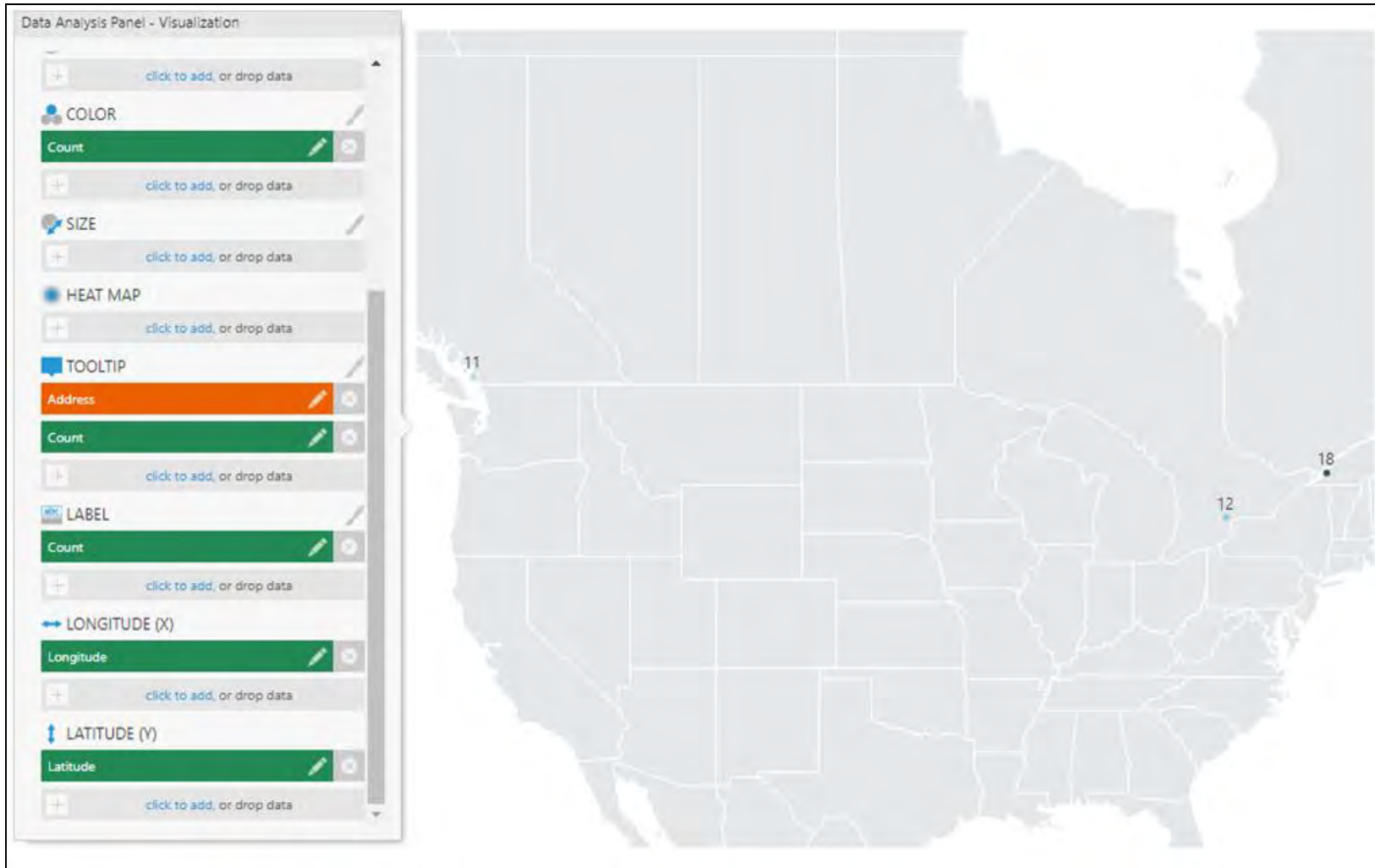
Your script will run when you submit the transform configuration dialog. You can expand **Data Preview** to view the output, including longitude and latitude coordinates:



Note: Web services like the one in this example may charge per use or for heavier usage. You may want to consider using [data cube storage](#) to reduce the frequency of requests to the service.

Now you can plot your addresses on a map using the steps shown in the article [Displaying symbols on a map](#), using your data cube as the data source. You can start by dragging **Address** from your data cube in the **Explore** window onto a new [metric set](#), for example, and re-visualizing to a **Map**.

In the Data Analysis Panel's **Visualization** tab, expand **Symbols** and assign the **Longitude (X)** and **Latitude (Y)** to the corresponding coordinates from your data, and assign other data to be visualized as you prefer.



Depending on your needs and how closely you are zoomed into your addresses, you can also set up the map to [display street-level maps](#) behind the symbols as context. Symphony maps support [Mapbox static tiles](#) as well as other map providers.

For more information, see:

- [Install Python](#)
- Transform: [Python Data Generator](#)



- Archive of documentation for Logi Symphony v24

- Transform: [R Language Analysis](#)
- [Using Transforms](#)
- Off the Charts: [Adding Weather Data is a Breeze](#)
- Off the Charts: [Advanced Sentiment Analysis](#)



Python Data Generator

This applies to: *Managed Dashboards, Managed Reports*

The **Python Data Generator** transform lets you generate data by writing scripts using the Python programming language. This is similar to the [Python Analysis](#) transform except that it does not accept input from a preceding transform and generates its own output directly from Python.

You can use the Python Data Generator transform to provide data to be used or visualized in Symphony. For example, Python can connect to and manipulate REST API data into a usable format, or generate data for prototyping or developing proof-of-concept dashboards.

To learn more about the Python language, see python.org.



Note: If your organization did not install Symphony, Python transforms may not be available depending on how your instance is hosted.

Setup

The Python programming environment must be installed on the Symphony server for you to use the Python Data Generator transform.

This is normally installed for you automatically as a prerequisite. See [Install Python](#) for more details, and for examples of how to install additional Python packages to use in your transforms.

Input

The Python Data Generator transform does not have any inputs. It generates output by running Python scripts.

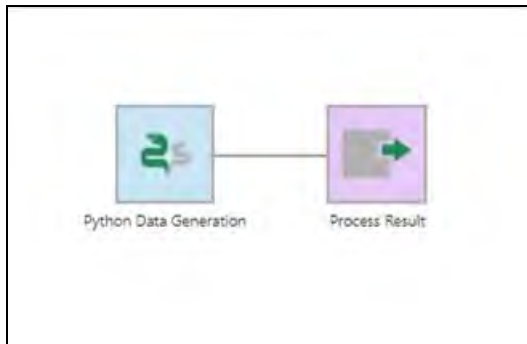
Add the Transform

When creating a new data cube, you can add the Python Data Generator transform to an empty canvas from the toolbar.

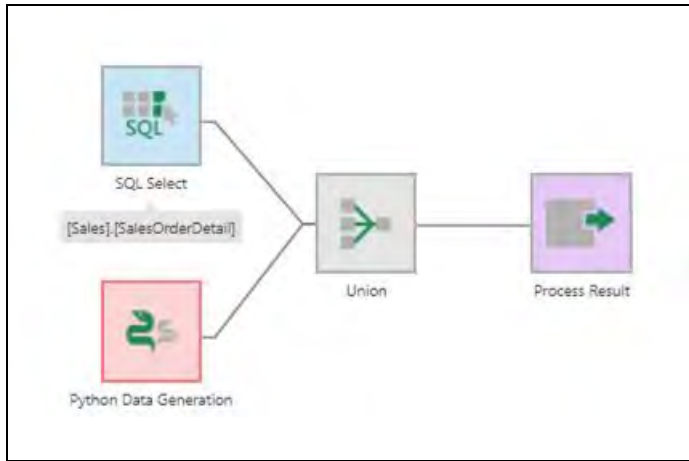


You can also create a new data cube from the left menu, and choose **Python Data Generator** in the dialog of options that appears.

The Python Data Generation transform is added to the data cube and connected to a Process Result transform automatically.



You can also add the Python Data Generator transform from the toolbar to an existing data cube process, and use a [Union](#) transform or another transform that combines data from multiple inputs.



Configure the Transform

Double-click the Python Data Generation transform, or select **Configure** from its right-click menu or the toolbar.

In the configuration dialog for the transform, the key task is to enter a **Python Script** that returns a result. Optionally click Edit script to use the Script Editor window for more space and to access [other helpful features](#).

As a simple example, a script for generating a column of numbers from 1 to 5 looks like this:

```
return (1,2,3,4,5)
```



Note: You can use the Print command to print variables to the application logs for debugging. To view these entries, ensure that the Log Filter configuration setting for Data Cubes is set to Information.



Python Data Generation

[more help](#)

The generator node. The user is responsible for entering a valid Python script that has a "return" statement.

Name

Description

ID

SETTINGS

Python Script

```
1 return (1,2,3,4,5)
```

Edit script

Placeholders

Timeout

Use Pooling

☐

Parameters

Output elements



Important: The result must be in a format that can be represented as a table, and data cubes will be unable to use Python outputs such as **Print** or **Draw**.

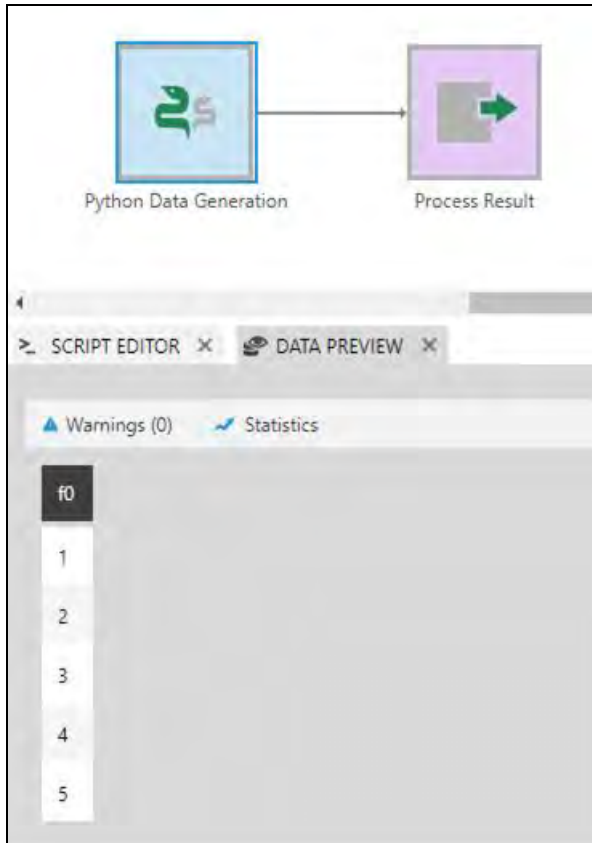
In this dialog, you can set up **Placeholders** to insert into the script that pass in parameter values similar to when using a [manual select](#). Insert the Identifier you entered for the placeholder (beginning and ending with the \$ symbol) into your script to access the current parameter value. When typing into the Python Script box shown in the image above, you can type the \$ symbol or right-click for a popup list of the current placeholders, then use the keyboard or mouse to select and insert it.

Select the **Use Pooling** option to reuse Python processes between execution of Python scripts to improve performance compared to starting a new process each time. There are corresponding [configuration settings](#) for Symphony administrators in the Python Pool category to manage the number of processes and their memory consumption.

Output

The output of the Python Data Generator depends on the script it is configured with. It can be a single value, a column of values, or multiple columns.

In the case of the simple script for generating numbers from 1 to 5, you can see an output column named **f0** in the Data Preview window.



Generate Data From a JSON File

Another example Python script for generating data is by connecting to a JSON file. This essentially uses a Python Data Generator transform in a data cube as a JSON data connector.

Setup

This example relies on four packages to be [installed](#) in Python on the Symphony server. If needed, open a command prompt as an administrator if the server runs Windows, for example, and use commands like the following for the four packages:

- `pip install numpy`
- `pip install pandas`



- `pip install jsonlib-python3`
- `pip install requests`

Create the Script

To generate the JSON data, configure the Python Data Generator transform and use script like the following, which uses `json_normalize`:

```
import json, requests
from pandas.io.json import json_normalize
url = "http://example.domain.com/data.json"
resp = requests.get(url)
data = json.loads(resp.text)
return json_normalize(data)
```

This can create a table reflecting all of the data in the referenced JSON file.

Generate Data From a REST Call

This example relies on the same four packages in Python as the previous section. An administrator with access to the Symphony server may need to install these packages as described above.

This example will log onto Symphony using [REST](#) in order to get a session ID. Next, it calls the Symphony file system query API with that session ID to retrieve all the dashboards that exist in a specific project. Finally it logs off, and then returns the results.

Create the Script

```
import json, requests
from pandas.io.json import json_normalize

host = "<<Insert host address>>"
username = "<<Insert user name>>"
password = "<<Insert password>>"
projectId = "<<Insert project id>>"

# authenticate with DBI
response = requests.post(host + "/api/logon/", json={"accountName":username, "password":password})
sessionId = response.json()["sessionId"]
```

```
requestBody = {
  "queryFileSystemEntriesOptions":{
    "entryIds":[
      projectId
    ],
    "pageNumber":0,
    "pageSize":0,
    "queryOptions":"RecursiveQuery",
    "orderBy":[
      {
        "fileSystemQueryField":"CreatedTime",
        "sortDirection":"Descending"
      }
    ],
    "filter":[
      {
        "field":"ObjectType",
        "operator":"Equals",
        "value":"Dashboard",
        "options":"None"
      }
    ]
  }
}

# query the file system api
response = requests.post(host + "/api/file/query", json=requestBody, headers={"Authorization":"Bearer " + sessionId})

# logout
requests.delete(host + "/api/session/current", headers={"Authorization":"Bearer " + sessionId})
result = json.loads(response.text)

# return the result
return json_normalize(result)
```



Important: Passing the session ID in an authorization header requires Dundas BI version 10 or higher. In earlier versions, you can pass it as a query string parameter in the request: `params={"sessionId":sessionId}`

For more information, see:



- Archive of documentation for Logi Symphony v24

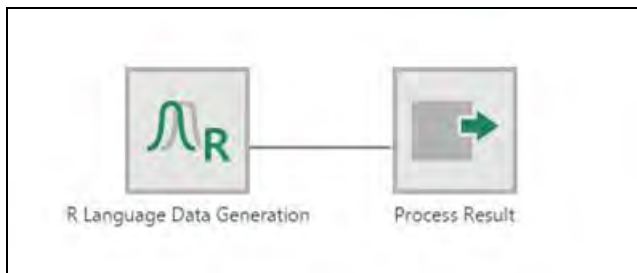
- [Install Python](#)
- Transform: [Python Analysis](#)
- Transform: [R Data Generator](#)
- Transform: [Flatten JSON](#)
- [Connecting to Generic Web Data Sources](#)
- [Using Transforms](#)
- [Using Manual Select Placeholders](#)
- [Creating a Data Provider](#)
- Off the Charts: [Adding Weather Data is a Breeze](#)
- Off the Charts: [Advanced Sentiment Analysis](#)

R Data Generator

This applies to: *Managed Dashboards, Managed Reports*

The **R Data Generator** transform lets you generate data by writing scripts using the R statistical programming language. This is similar to the [R Language Analysis](#) transform except that it does not accept input from a preceding transform and generates its own output directly from R.

R is both a programming language and an environment for statistical computing, graphics, and predictive analysis. You can use the R Data Generator transform to generate data for prototyping or developing proof-of-concepts, or if you are using R to access a data source.



To learn more about the R language, see [The R Project for Statistical Computing](#).

Setup

Before you can use the R Data Generator transform in Symphony, the R programming environment must be installed on a server.

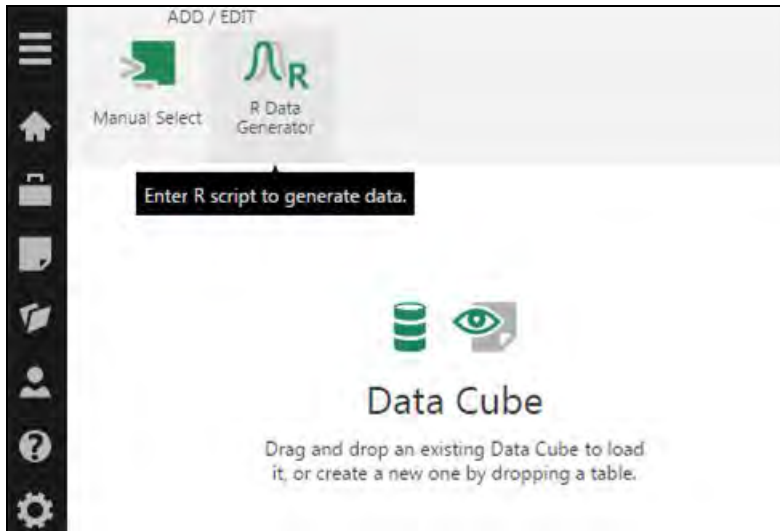
See [Install and Configure R](#).

Input

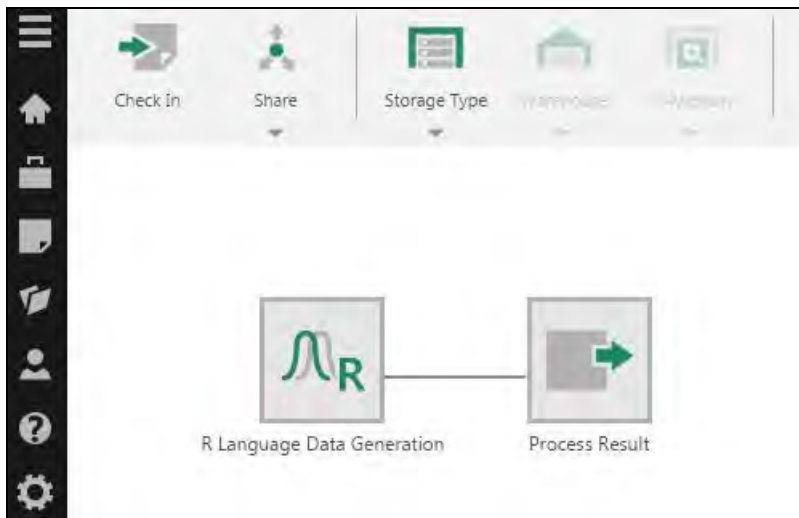
The R Data Generator transform does not have any inputs. It just generates output by running R scripts against the R server.

Add the Transform

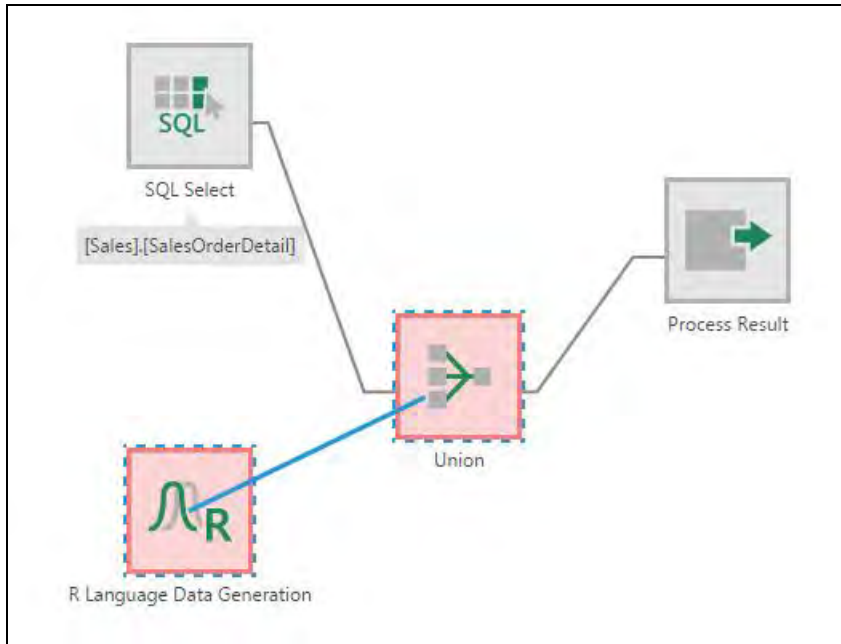
When creating a new data cube, you can add the R Data Generator transform to an empty canvas from the toolbar.



The **R Language Data Generation** transform is added to the data cube and connected to a Process Result transform automatically.



You can also add the R Data Generator transform from the toolbar to an existing data cube process. A typical example is to connect the R Language Data Generation instance to a [Union](#) transform which merges data from multiple inputs.



Configure the Transform

Double click the R Language Data Generation transform or select the **Configure** option from its right-click menu.

In the configuration dialog for the transform, the key task is to enter an R script that sets the output variable.


R Language Data Generation

The inception transform. The user is responsible for entering a valid R script that sets the "output" variable.

Name

Description

ID

2942742f-d138-4117-ac68-74f9e0ebb989

SETTINGS

Timeout

30

R Script

```
1 output=c(1,2,3,4,5)
```

Edit script

Placeholders

Parameters

Output elements

For example, a simple script for generating a column of numbers from 1 to 5 looks like this:

```
output=c(1,2,3,4,5)
```

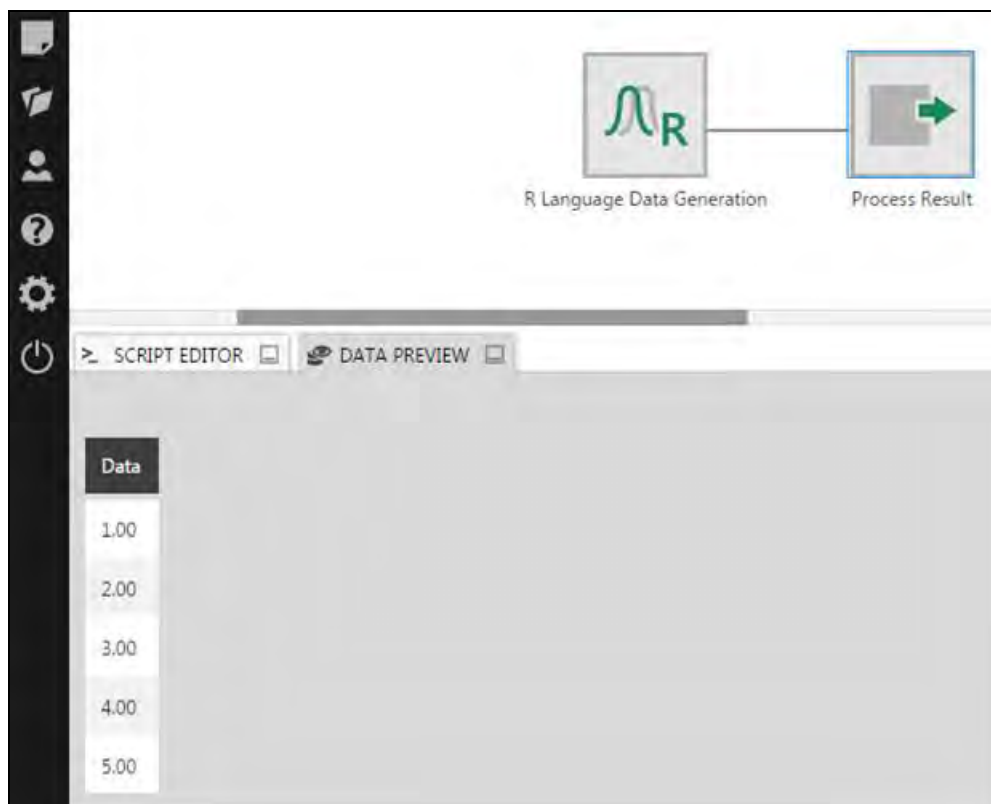
In this dialog, you can set up Placeholders to insert into the script that pass in parameter values similar to when using a [manual select](#).

You can also set up Parameters to directly filter this transform's output like with [select transforms](#).

Output

The output of the R Data Generator depends on the R script it is configured with. It can be a single value, a column of values, or multiple columns.

In the case of the simple script for generating numbers from 1 to 5, you can see an output column named **Data** by selecting the Process Result transform and then clicking on **Data Preview**.



Example R scripts

Here are some example R scripts for generating data.

Random Number Generation

Generate 10 random numbers between 200.5 and 300.5:

```
output=runif(10, 200.5, 300.5)
```

Generate 5 random integers between 1 and 1000:

```
output=sample(1:1000, 5)
```

Generate two columns of data. The first column contains integers from 1 to 5 in order. The second column contains 5 random integers between 50 and 100:

```
x=c(1,2,3,4,5)
y=sample(50:100, 5)
output=data.frame(x,y)
```

Generate two columns, the first column with 12 random dates between 2017/01/01 and 2018/01/01, and the 2nd column with 12 random integers between 1 and 1000:

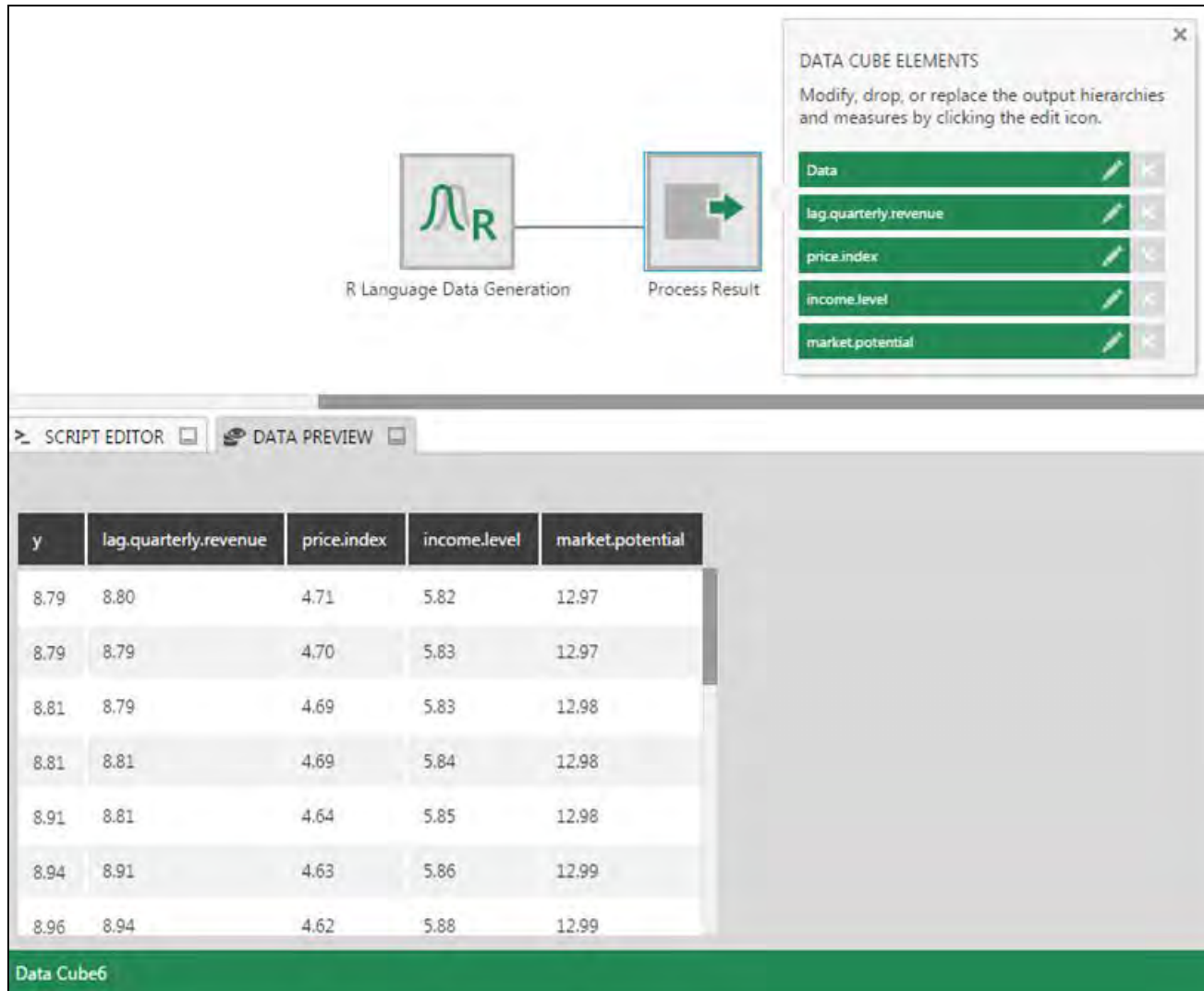
```
x=sample(1:1000, 12)
y=sample(seq(as.Date('2017/01/01'), as.Date('2018/01/01'), by="day"), 12)
output=data.frame(x,y)
```

Pre-Defined Datasets

Load pre-defined data from the [R Datasets Package](#). For example, [Freeny's Revenue Data](#):

```
output=datasets::freeny
```

Here's the resulting Data Preview:



The screenshot displays the Logi Symphony v24 interface. At the top, a workflow diagram shows 'R Language Data Generation' (represented by an R logo) connected to 'Process Result' (represented by a green arrow icon). To the right, a 'DATA CUBE ELEMENTS' panel is open, listing five data elements: 'Data', 'lag.quarterly.revenue', 'price.index', 'income.level', and 'market.potential'. Each element has an edit icon (pencil) and a delete icon (X). Below the workflow diagram, the 'SCRIPT EDITOR' and 'DATA PREVIEW' tabs are visible. The 'DATA PREVIEW' tab is active, showing a table with five columns: 'y', 'lag.quarterly.revenue', 'price.index', 'income.level', and 'market.potential'. The table contains eight rows of data. At the bottom left, the text 'Data Cube6' is displayed.

y	lag.quarterly.revenue	price.index	income.level	market.potential
8.79	8.80	4.71	5.82	12.97
8.79	8.79	4.70	5.83	12.97
8.81	8.79	4.69	5.83	12.98
8.81	8.81	4.69	5.84	12.98
8.91	8.81	4.64	5.85	12.98
8.94	8.91	4.63	5.86	12.99
8.96	8.94	4.62	5.88	12.99

Data Cube6

For more information, see:

- [Install and Configure R](#)
- Transform: [R Language Analysis](#)



- Archive of documentation for Logi Symphony v24

- Transform: [Python Data Generator](#)
- [Using Transforms](#)
- [Configuration Settings](#)

R resources:

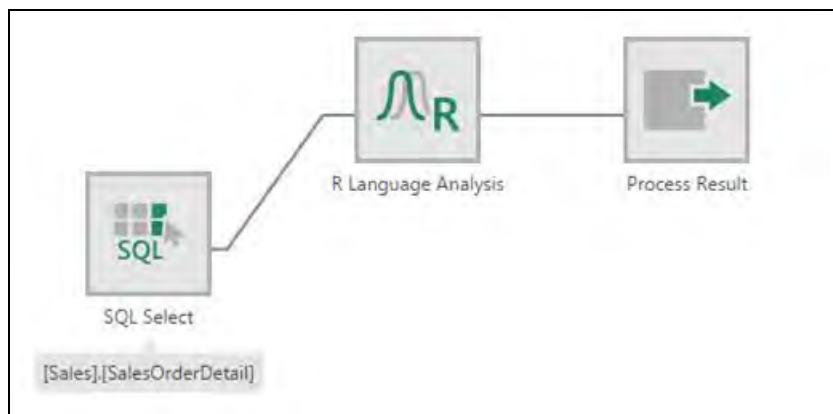
- [The R Project for Statistical Computing](#)
- [The Comprehensive R Archive Network](#)
- [An Introduction to R](#)
- [The R Datasets Package](#)

R Language Analysis

This applies to: *Managed Dashboards, Managed Reports*

The **R Language Analysis** transform lets you write scripts using the R programming language to perform statistical and predictive analysis on data. This is similar to the [R Data Generator](#) transform except that it operates on input data from a preceding transform instead of generating its own output.

R is both a programming language and an environment for statistical computing and graphics. To learn more about the R language, see [The R Project for Statistical Computing](#).



Setup

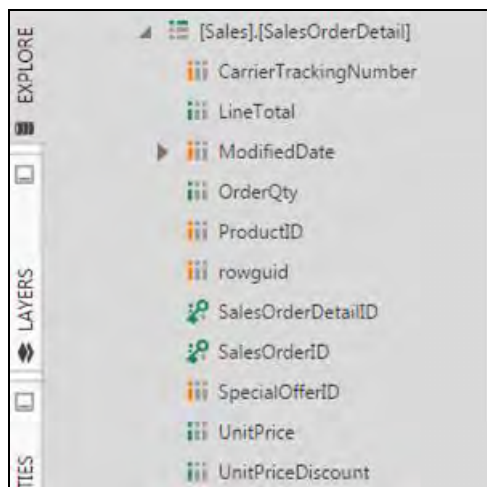
Before you can use the **R Language Analysis** transform in Symphony, the R programming environment must be installed on a server.

See [Install and Configure R](#).

Input

The R Language Analysis transform requires one or more input transforms, each of which may produce multiple columns of data.

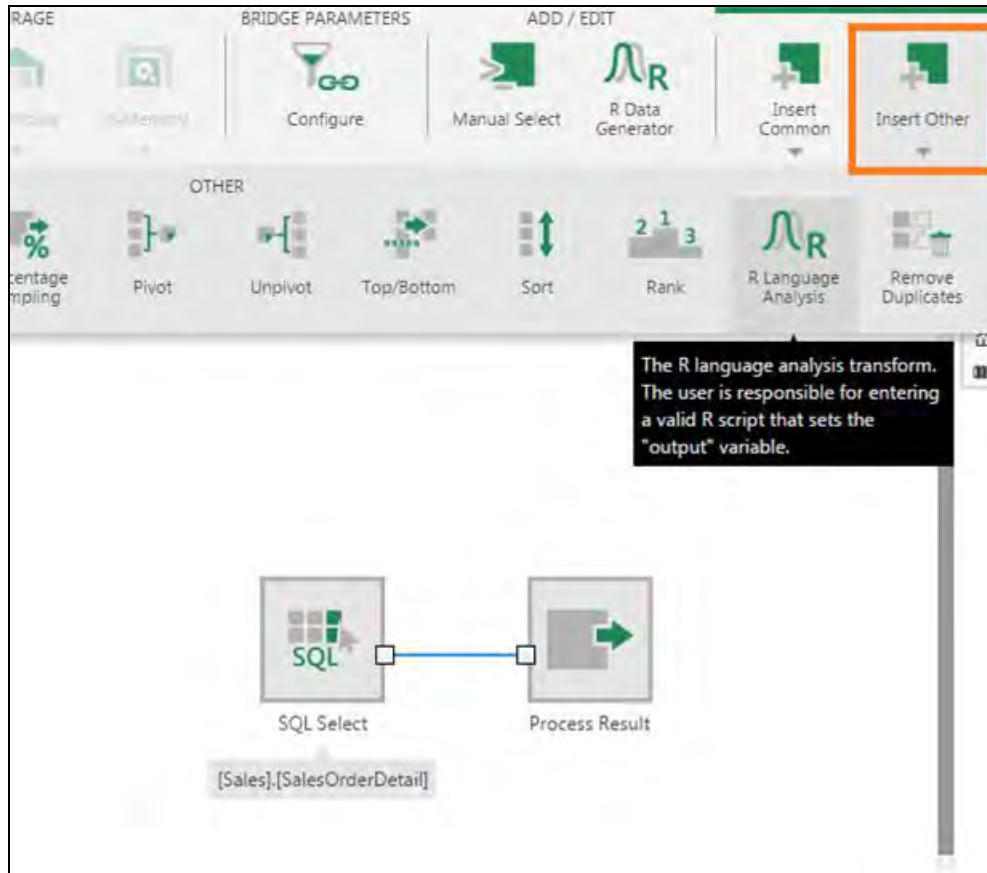
For example, the input could be a [SQL Select](#) transform that reads data from the [AdventureWorks](#) 2012 table `[Sales].[SalesOrderDetail]`.



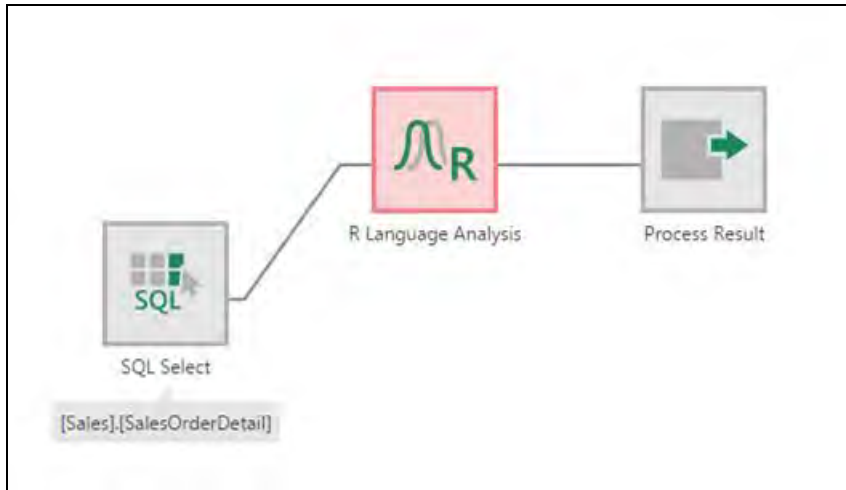
Add the Transform

To add this transform to an existing data cube process, first select the connection link between two connected transforms.

Go to the toolbar, click **Insert Other**, and then select **R Language Analysis**.



The R Language Analysis transform is inserted between the two transforms and appears highlighted because it needs further configuration.



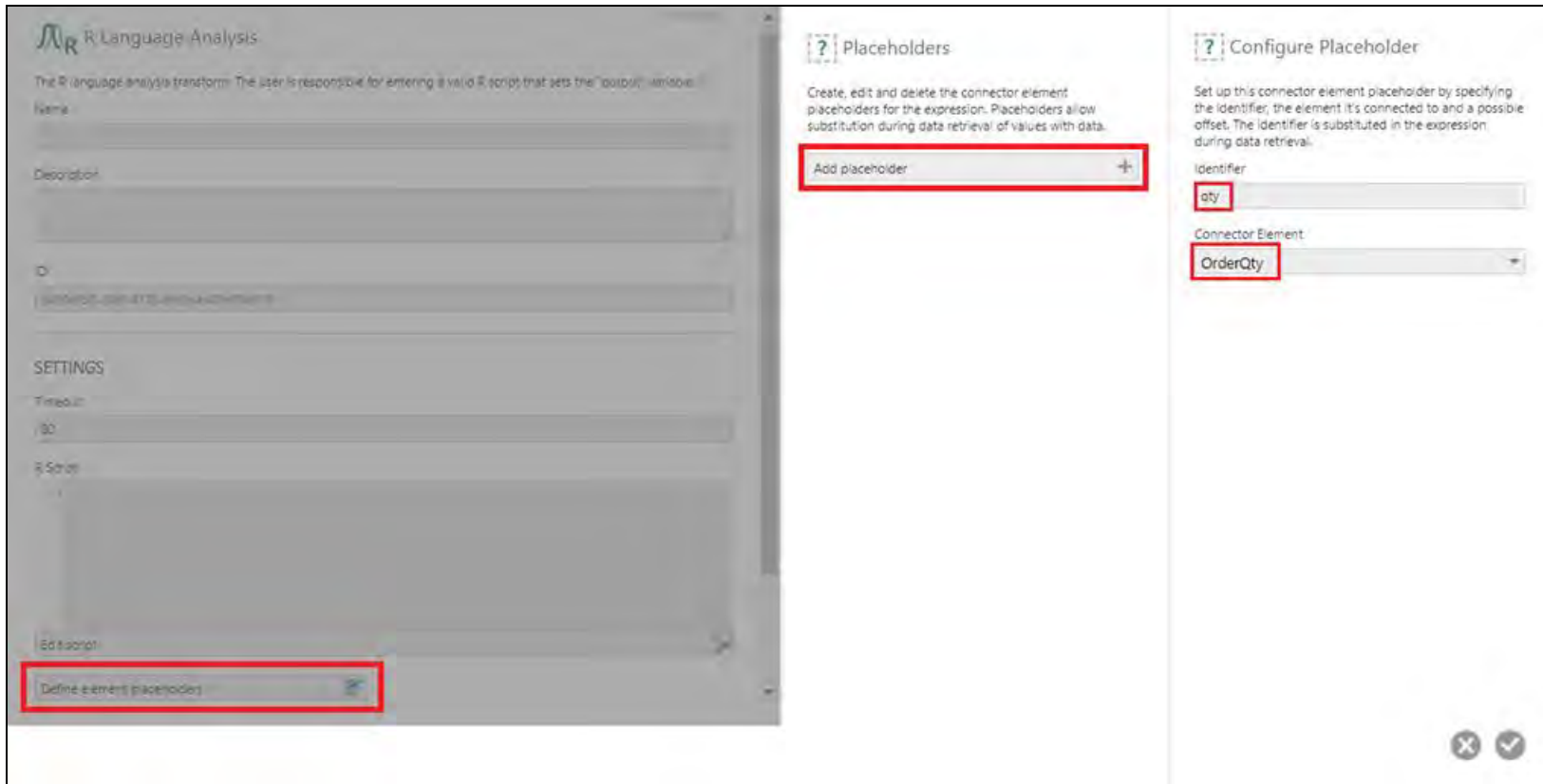
Configure the Transform

Double-click the R Language Analysis transform or select the **Configure** option from its right-click menu.

In the configuration dialog for the transform, the objective is to decide which input column(s) you want to analyze or apply processing to, and then enter an R script that performs this analysis or processing on the input data.

For example, suppose you want to calculate the total sum of the **OrderQty** column from the input transform, and send this sum as the output.

To begin, click **Define element placeholders** in the configuration dialog. Click **Add placeholder**. In the **Identifier** box, enter a variable name (e.g., **qty**) for the input column which you will use in the R script. Use the drop-down to select the corresponding input column (e.g., **OrderQty**).



R Language Analysis

The R language analysis transform: The user is responsible for entering a valid R script that sets the "output" variable.

Item:

Description:

ID:

SETTINGS

Timeout:

90

R Script:

Edit script

Define element placeholders

Placeholders

Create, edit and delete the connector element placeholders for the expression. Placeholders allow substitution during data retrieval of values with data.

Add placeholder

Configure Placeholder

Set up this connector element placeholder by specifying the identifier, the element it's connected to and a possible offset. The identifier is substituted in the expression during data retrieval.

Identifier:

qty

Connector Element:

OrderQty

You can now write an R script that references the qty variable by enclosing it between dollar sign characters. For example:

```
output <- sum($qty$)
```

This script calculates the sum of the **OrderQty** input column and assigns the result to the output variable.



R Language Analysis

more help

The R language analysis transform. The user is responsible for entering a valid R script that sets the "output" variable.

Name

Description

ID

5490c957-206f-4176-b66b-e408e79e11ff

SETTINGS

Timeout

30

R Script

```
1. output <- sum(qty)
```

Edit script

Define element placeholders

✓

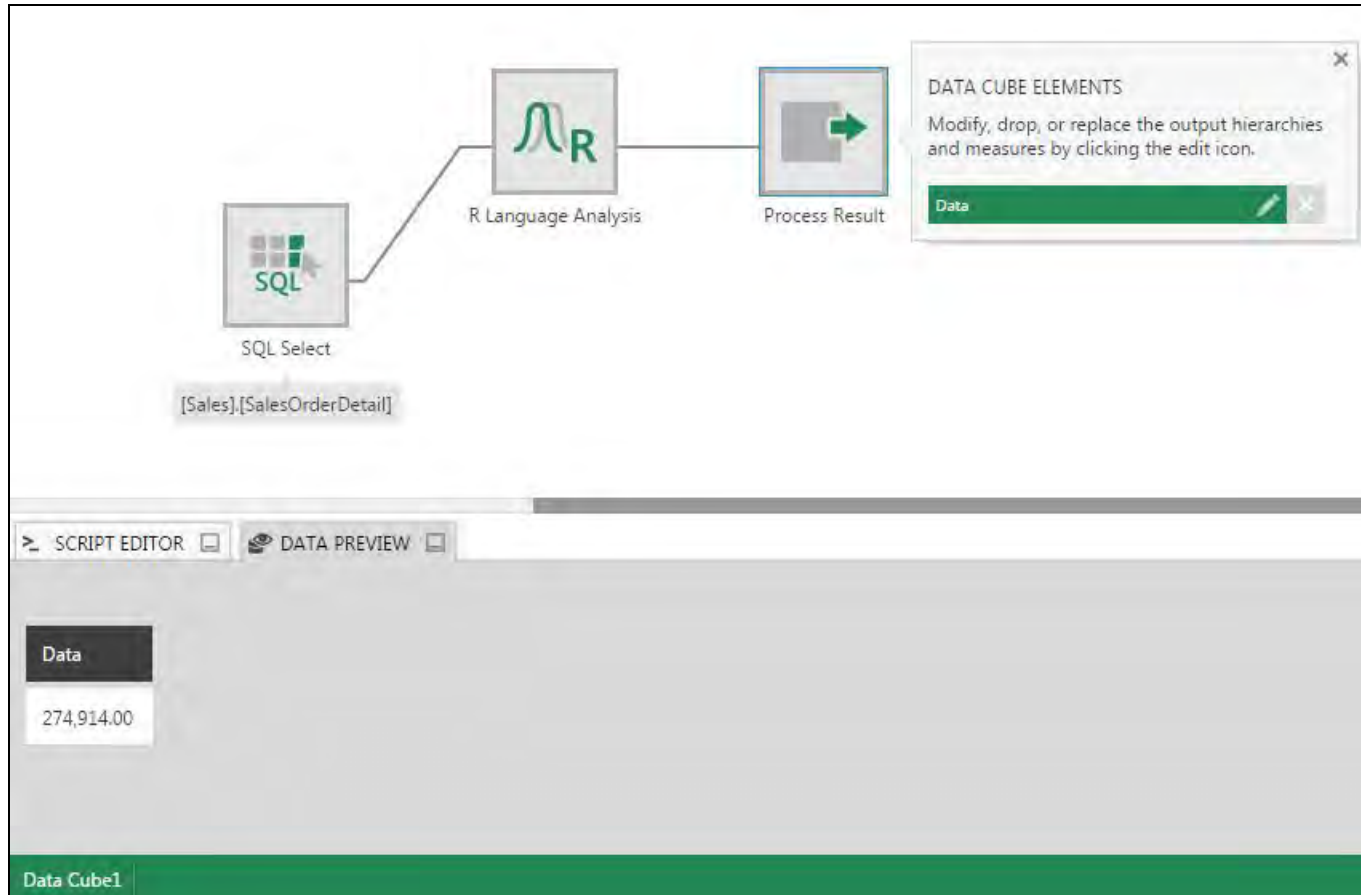


Note: Click **Edit script** to open the script in the Script Editor window for editing. This is useful for longer R script because it offers undo/redo, a go-to-line option, and other functionality.

Output

The output of the R Language Analysis transform depends on the R script it is configured with. It can be a single value, a column of values, or multiple columns.

In the case of the script that calculates the sum of the **OrderQty** column, you can see an output column named 'Data' by selecting the **Process Result** transform and then clicking on **Data Preview**.



The screenshot displays the Logi Symphony v24 interface. At the top, a data pipeline is visible with three transforms: 'SQL Select' (inputting '[Sales].[SalesOrderDetail]'), 'R Language Analysis', and 'Process Result'. A 'DATA CUBE ELEMENTS' dialog box is open, showing a table with one row and one column named 'Data'. The value in the 'Data' column is 274,914.00. The bottom of the interface shows a 'SCRIPT EDITOR' tab and a 'DATA PREVIEW' tab. The 'DATA PREVIEW' tab is active, showing the same data as the dialog box. The bottom status bar indicates 'Data Cube1'.

Example R Scripts

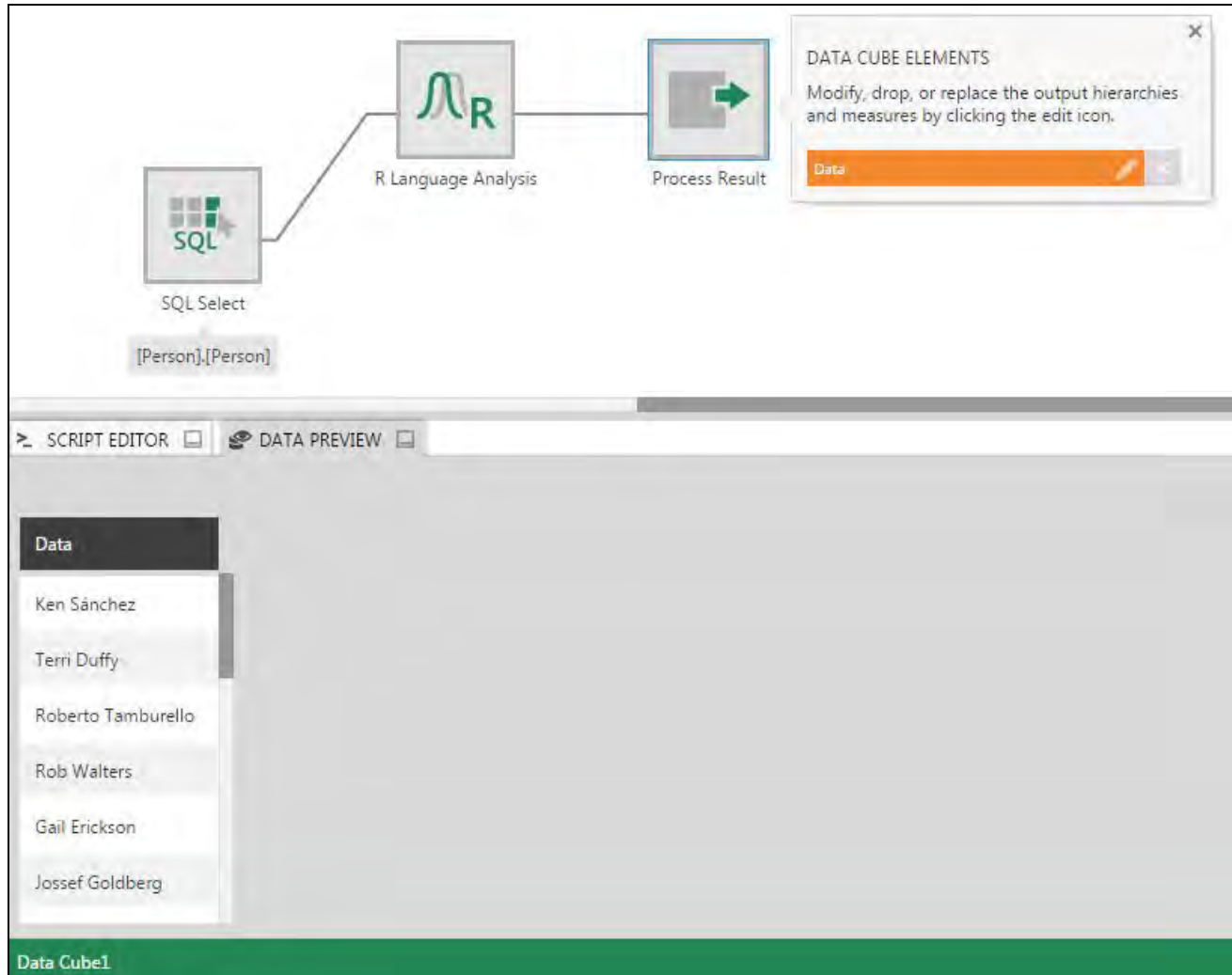
Concatenate Text Columns

For this example, use the `[Person].[Person]` table from AdventureWorks 2012 as input to the R Language Analysis transform.

Define element placeholders for the `FirstName` and `LastName` columns. Then enter the following R script to concatenate the two text columns into a single output column:

```
output <- paste($FirstName$, $LastName$, sep=" ")
```

Here's the resulting output:



The screenshot displays the Logi Symphony v24 interface. At the top, a workflow diagram shows three steps: 'SQL Select' (with a green SQL icon), 'R Language Analysis' (with a green R icon), and 'Process Result' (with a green arrow icon). Below the 'SQL Select' step, the data source is identified as '[Person].[Person]'. To the right of the workflow, a 'DATA CUBE ELEMENTS' panel is visible, containing the text: 'Modify, drop, or replace the output hierarchies and measures by clicking the edit icon.' Below this text, there is an orange bar labeled 'Data' with an edit icon. The main interface area is divided into two tabs: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a list of names under the 'Data' header: Ken Sánchez, Terri Duffy, Roberto Tamburello, Rob Walters, Gail Erickson, and Jossef Goldberg. At the bottom of the interface, a green bar displays 'Data Cube1'.

Moving Average

Suppose you have a Sales table with a Sales column containing figures for 11 months.

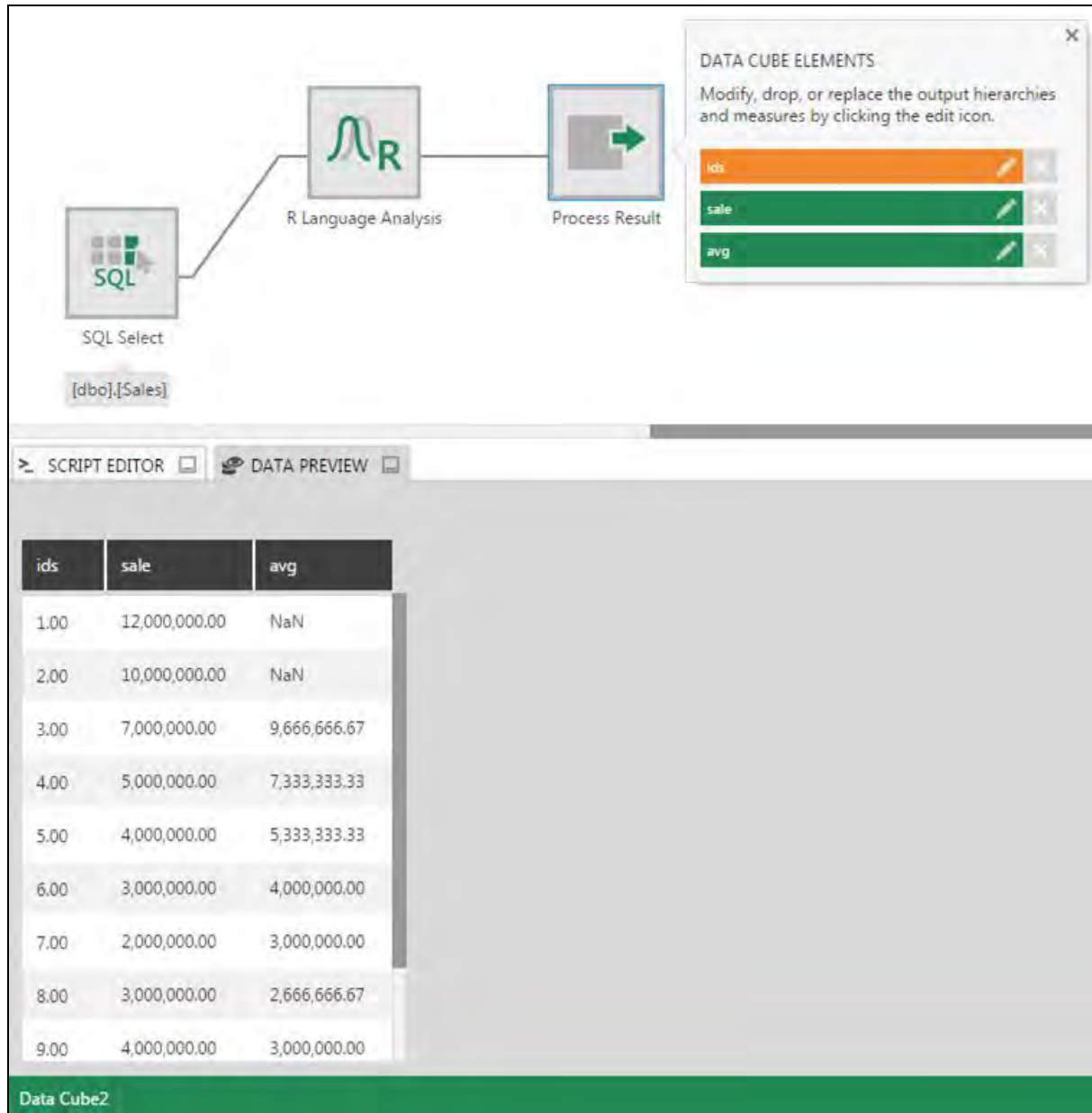


- Archive of documentation for Logi Symphony v24

Use this as input to a R Language Analysis transform, define a placeholder for the Sales column, and then enter the following R script:

```
movavg <- function(x,n=3){filter(x,rep(1/n,n), sides=1)}  
sale <- $Sales$  
avg <- movavg($Sales$)  
ids <- 1:11  
output <- data.frame(ids, sale, avg)
```

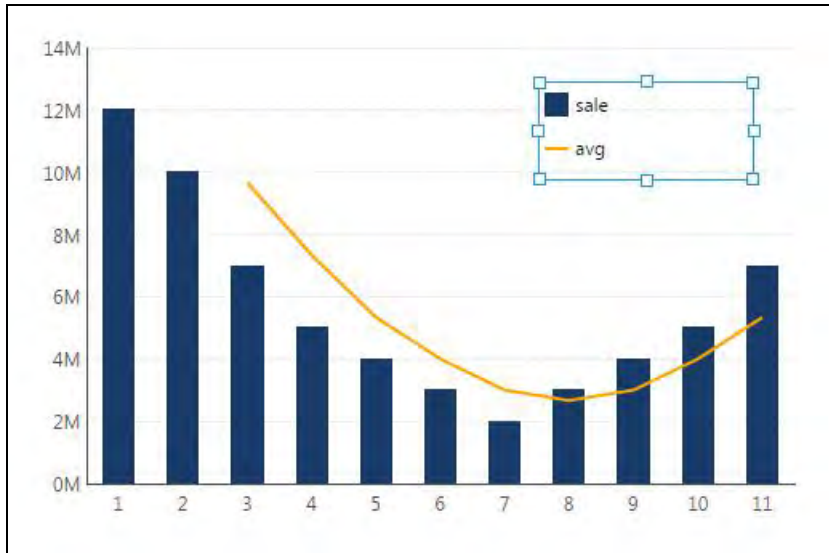
Here's the resulting Data Preview showing the output:



The screenshot displays the Logi Symphony v24 interface. At the top, a data pipeline is shown with three components: 'SQL Select', 'R Language Analysis', and 'Process Result'. The 'SQL Select' component is connected to the 'R Language Analysis' component, which is then connected to the 'Process Result' component. Below the 'SQL Select' component, the database schema '[dbo].[Sales]' is visible. To the right of the pipeline, a 'DATA CUBE ELEMENTS' panel is open, showing a list of elements: 'ids' (orange bar), 'sale' (green bar), and 'avg' (green bar). Each element has an edit icon (pencil) and a delete icon (X). Below the pipeline, there are two tabs: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table with three columns: 'ids', 'sale', and 'avg'. The table contains 9 rows of data. At the bottom of the interface, a green bar displays 'Data Cube2'.

ids	sale	avg
1.00	12,000,000.00	NaN
2.00	10,000,000.00	NaN
3.00	7,000,000.00	9,666,666.67
4.00	5,000,000.00	7,333,333.33
5.00	4,000,000.00	5,333,333.33
6.00	3,000,000.00	4,000,000.00
7.00	2,000,000.00	3,000,000.00
8.00	3,000,000.00	2,666,666.67
9.00	4,000,000.00	3,000,000.00

And the corresponding visualization on a dashboard:



For more information, see:

- [Install and Configure R](#)
- Transform: [R Language Analysis](#)
- Transform: [Python Data Generator](#)
- [Using Transforms](#)
- [Configuration Settings](#)

R resources:

- [The R Project for Statistical Computing](#)
- [The Comprehensive R Archive Network](#)



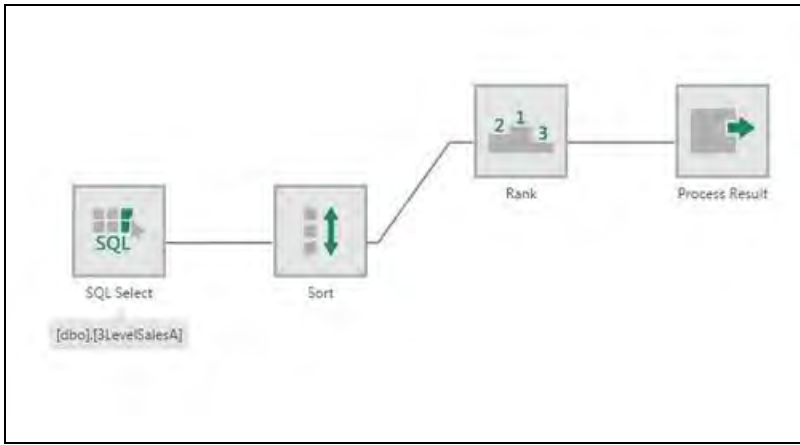
- Archive of documentation for Logi Symphony v24

- [An Introduction to R](#)
- [The R Datasets Package](#)

Rank

This applies to: *Managed Dashboards, Managed Reports*

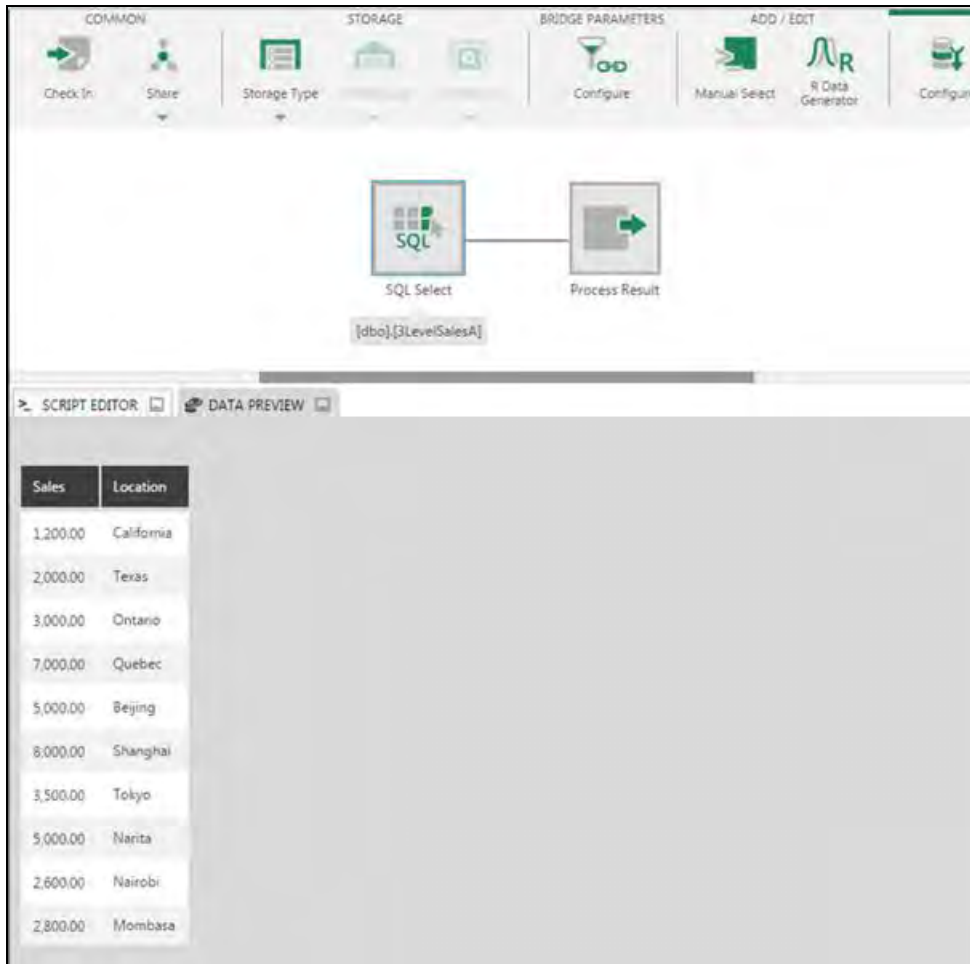
The **Rank** transform ranks your data and adds the rank number to your input. This transform should be used right after a [Sort](#) transform, so that the data will be ranked according to your sorting rule.



Input

The Rank transform requires one input transform that has at least one column.

For our example, we'll use the unsorted data below:



The screenshot shows the Logi Symphony v24 interface. At the top, there are tabs for COMMON, STORAGE, BRIDGE PARAMETERS, and ADD / EDIT. Below these tabs, there are icons for Check In, Share, Storage Type, Configure, Manual Select, R Data Generator, and another Configure. In the center, there is a workflow diagram with two components: 'SQL Select' and 'Process Result'. The 'SQL Select' component is connected to the 'Process Result' component. Below the 'SQL Select' component, the query '[dbo].[3LevelSalesA]' is displayed. At the bottom, there is a 'SCRIPT EDITOR' and a 'DATA PREVIEW' tab. The 'DATA PREVIEW' tab is active, showing a table with two columns: 'Sales' and 'Location'. The table contains 10 rows of data, sorted by Sales in descending order.

Sales	Location
1,200.00	California
2,000.00	Texas
3,000.00	Ontario
7,000.00	Quebec
5,000.00	Beijing
8,000.00	Shanghai
3,500.00	Tokyo
5,000.00	Narita
2,600.00	Nairobi
2,800.00	Mombasa

In order rank the data according to our Sales numbers, we will first need to add a [Sort](#) transform. In this example, the sorting rule applied is by **Sales** (descending order), followed by **Location** (ascending order).



Sort

The transform allows various sorting rules to be applied to the input columns and copies the results to the output. The sorting applied is only applicable inside the data cube.

Name

Description

Drag and drop the elements you want to sort your data on into the 'Sorting Elements' section. Click on the sort icon of each transform element to change its sorting method. Drag and drop within the list to set the priority.

Sorting Transform Input

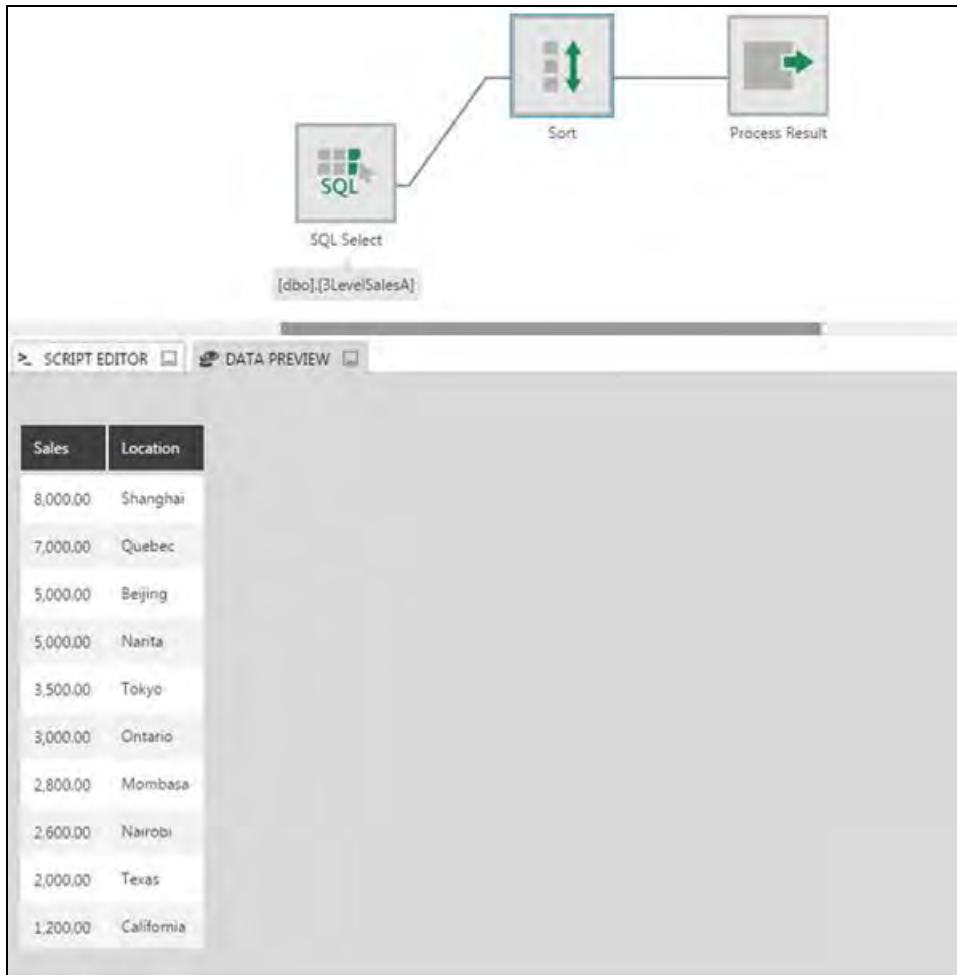
- ☒ Sales
- ☒ Location

Sorting Elements

- Sales
- Location

Define parameters

Rename output elements



SQL Select

[dbo].[3LevelSalesA]

Sort

Process Result

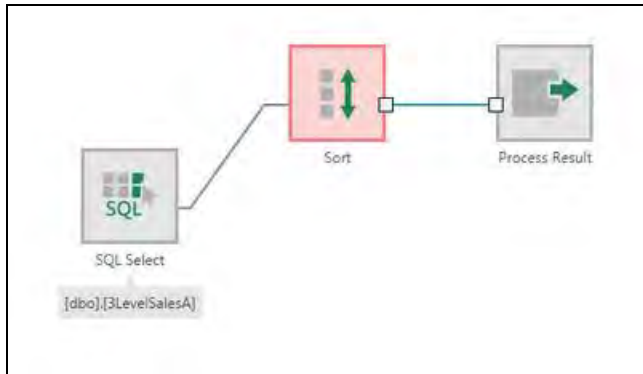
SCRIPT EDITOR

DATA PREVIEW

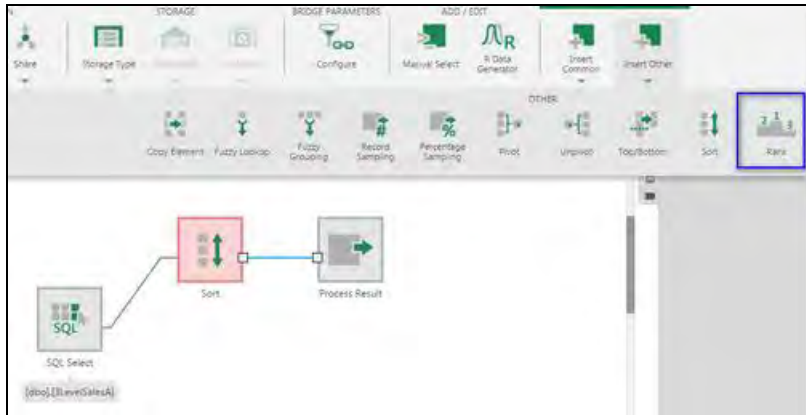
Sales	Location
8,000.00	Shanghai
7,000.00	Quebec
5,000.00	Beijing
5,000.00	Nanta
3,500.00	Tokyo
3,000.00	Ontario
2,800.00	Mombasa
2,600.00	Nairobi
2,000.00	Texas
1,200.00	California

Add the Rank Transform

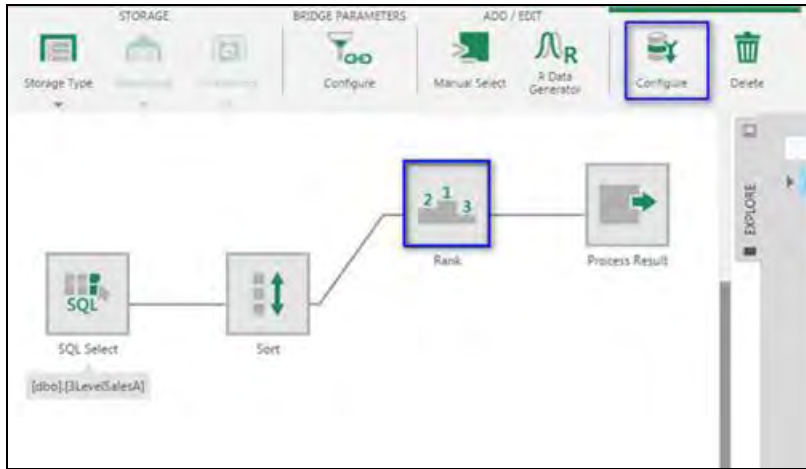
Select the connector link between two transforms.



In the toolbar, choose **Insert Other**, then **Rank**.



To edit/configure the transform, select it and choose **Configure** in the toolbar.



Configure

In the **Rank** transform configuration dialog:



Rank

The transform allows various sorting rules to be applied to the input columns and assigns a rank to each row. The sorting applies only inside the data cube.

Name

Description

Rank Element Name

Rank

Rank Element Description

Rank of record

Rank Type

Standard

Rank Transform Input

☒ Sales

☒ Location

Ranking Elements

Sales

Define parameters

Rename output elements

Optionally, set the **Name** and **Description** for your reference.



- Archive of documentation for Logi Symphony v24

Set **Rank Element Name** to name the new column that will contain the rank number, and an optional **Rank Element Description**.

Set **Rank Type** to one of the following:

- **Standard** - Consecutive values are assigned based on the sorting order (e.g., 1, 2, 3, 4).
- **Podium** - If the value of consecutive records are the same, then the same rank is given (e.g., 1, 2, 3, 3).



Rank

The transform assigns a rank to each row. The data needs to be sorted according to the ranking elements by the transform before it. The data can be sorted from the source or else a sort transform can be used.

Name

Description

Rank Element Name

Rank Element Description

Rank Type

Continue Sequence

EVALUATION SOFTWARE

Rank Transform Input

Ranking Elements

Define parameters

Rename output elements

If you chose **Podium**, a **Continue Sequence** option appears. If selected, the sequence of numbers will not skip when consecutive records contain the same values in the rank elements. For example, if unchecked, the sequence could be 1, 2, 3, 3, 5.

Under **Rank Transform Input**, uncheck any columns you don't want to be included in the output.

Drag and drop the input columns to **Ranking Elements** that you want to include in the sorting key. The input data should already be sorted by these elements.

Output

The following is the output for our example with **Rank Type** set to **Standard**:

Sales	Rank	Location
8,000.00	1.00	Shanghai
7,000.00	2.00	Quebec
5,000.00	3.00	Beijing
5,000.00	4.00	Narita
3,500.00	5.00	Tokyo
3,000.00	6.00	Ontario
2,800.00	7.00	Mombasa
2,600.00	8.00	Nairobi
2,000.00	9.00	Texas
1,200.00	10.00	California

With **Rank Type** set to **Podium**, with **Continue Sequence** selected:

Sales	Rank	Location
8,000.00	1.00	Shanghai
7,000.00	2.00	Quebec
5,000.00	3.00	Beijing
5,000.00	3.00	Narita
3,500.00	4.00	Tokyo
3,000.00	5.00	Ontario
2,800.00	6.00	Mombasa
2,600.00	7.00	Nairobi
2,000.00	8.00	Texas
1,200.00	9.00	California

With **Continue Sequence** unchecked:

Sales	Rank	Location
8,000.00	1.00	Shanghai
7,000.00	2.00	Quebec
5,000.00	3.00	Beijing
5,000.00	3.00	Narita
3,500.00	5.00	Tokyo
3,000.00	6.00	Ontario
2,800.00	7.00	Mombasa
2,600.00	8.00	Nairobi
2,000.00	9.00	Texas
1,200.00	10.00	California

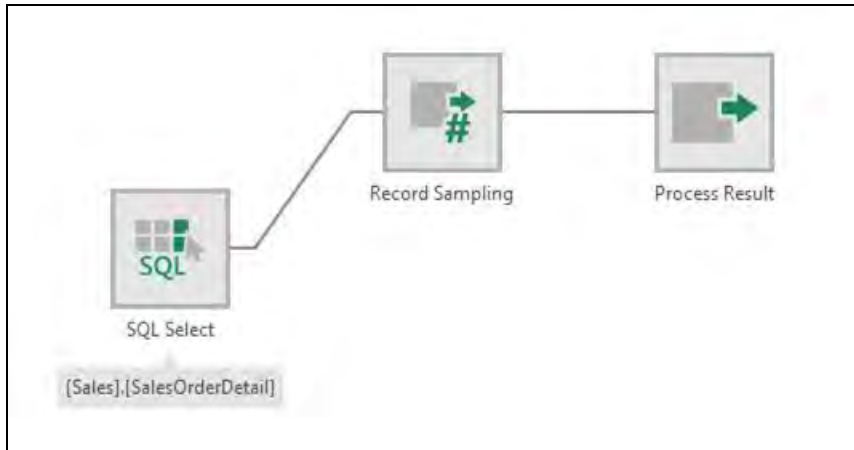
For more information, see:

- [Using Transforms](#)
- [Sort](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Record Sampling

This applies to: *Managed Dashboards, Managed Reports*

The **Record Sampling** transform returns the specified number of records out of the full set of records by random selection.

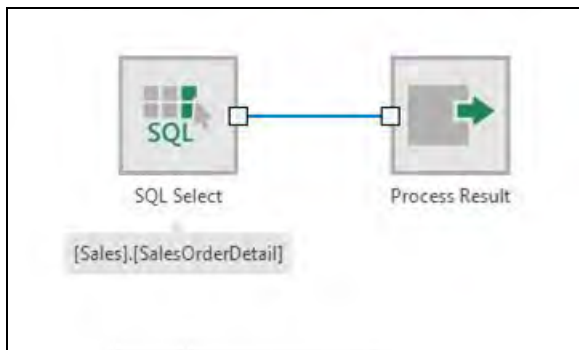


Input

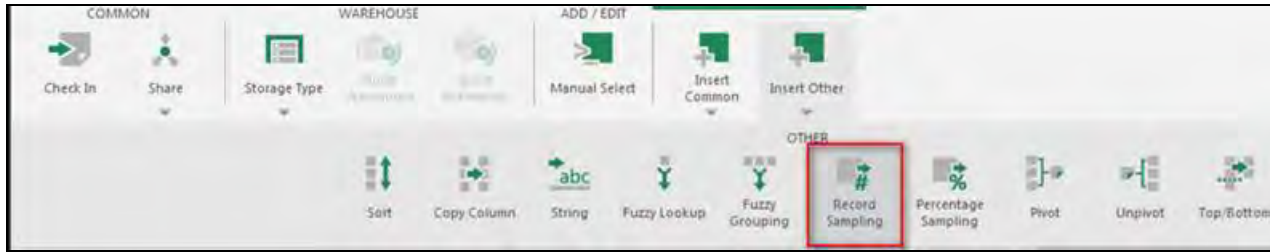
The Record Sampling transform requires one input transform that has at least one column.

Add the Transform

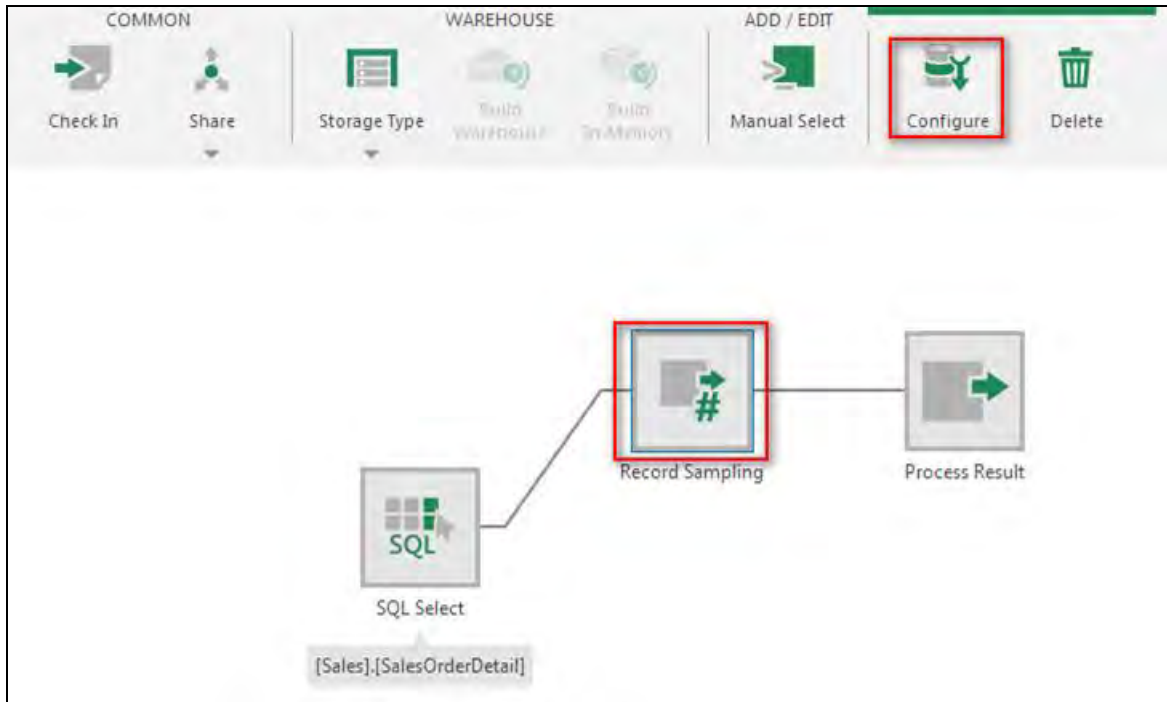
Select the connector link between two transforms.



In the toolbar, choose **Insert Other**, then **Record Sampling**.



To edit/configure the transform, select it and choose **Configure** in the toolbar.



Configure

In the **Record Sampling** transform configuration dialog:



Record Sampling

Record Sampling node, responsible for returning a portion of the input according to the count input by user.

[More help...](#)

Sampling Record Quantity

10

Random Seed

0




☐ SalesOrderID
☐ SalesOrderDetailID
☐ CarrierTrackingNumber
☒ OrderQty
☒ ProductID
☐ SpecialOfferID
☐ UnitPrice
☐ UnitPriceDiscount
☐ LineTotal
☐ rowguid
☐ ModifiedDate

1. Set **Sampling Record Quantity** - The number of records in the output.
2. Optionally, set **Random Seed** - An arbitrary number no greater than 2,147,483,647, used to used to initialize a pseudorandom number generator. If set to zero, the current time is used to randomly determine which records will be sampled; otherwise, the same records will be sampled given the same input data and sampling record quantity.
3. Select the columns to be included in the output.



- Archive of documentation for Logi Symphony v24

As with other transforms, these settings can be turned into a parameter to be changed outside of the transform or the data cube (e.g., with a filter on a dashboard) by clicking **Define parameters**.

Output

Submit the configuration dialog, and with the transform selected, open the **Data Preview** window to view the output.

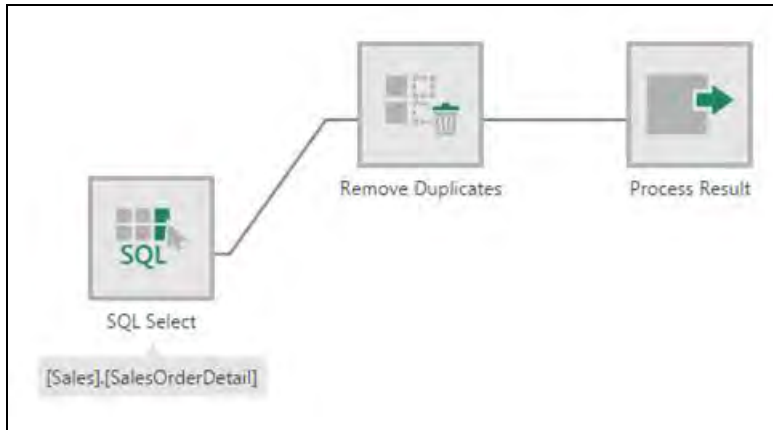
For more information, see:

- [Using Transforms](#)
- [Percentage Sampling](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Remove Duplicates

This applies to: *Managed Dashboards, Managed Reports*

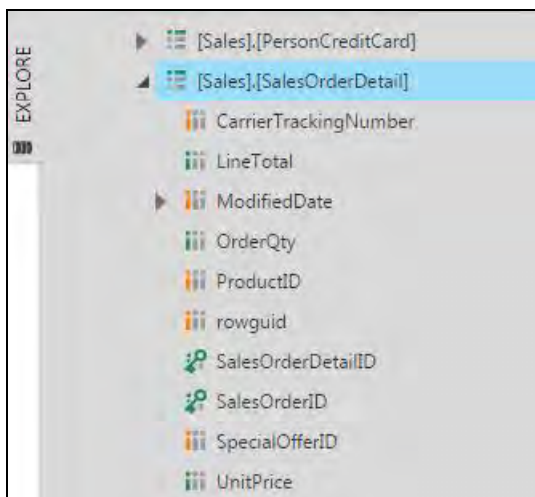
The Record Sampling transform returns the specified number of records out of the full set of records by random selection.



Input

The Remove Duplicates transform requires one input transform.

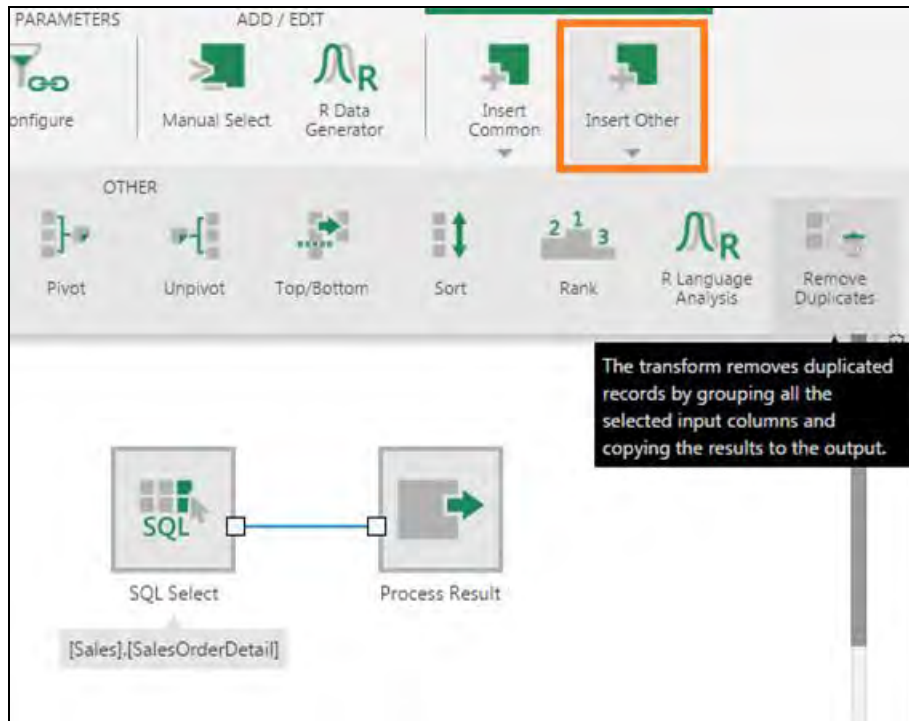
For example, the input could be a [SQL Select](#) transform that reads data from the [AdventureWorks 2012](#) table `[Sales].[SalesOrderDetail]`.



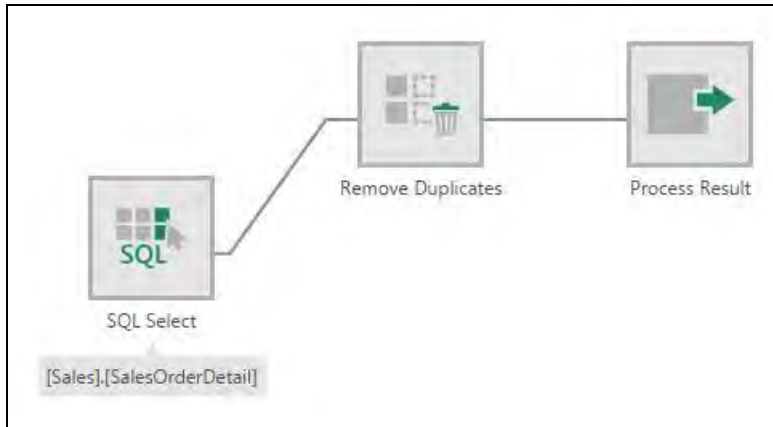
Add the Transform

To add this transform to an existing data cube process, first select the connection link between two connected transforms.

Go to the toolbar, click **Insert Other**, and then select **Remove Duplicates**.



The Remove Duplicates transform is inserted between the two transforms.



Configure

Double-click the **Remove Duplicates** transform, or select the **Configure** option from its right-click menu.

In the configuration dialog for the transform, select the input columns to be grouped and sent to the output. For example, if you simply select a **ProductID** column for the **SalesOrderDetail** table, the expectation will be that the resulting output will consist of a column of unique product IDs.

more help



Remove Duplicates

The transform removes duplicated records by grouping all the selected input columns and copying the results to the output.

Name

Description

Input

Remove Duplicates Transform Input




☐ CarrierTrackingNumber	String
☐ LineTotal	Double
☐ ModifiedDate	DateTime
☐ OrderQty	Int16
☒ ProductID	Int32
☐ rowguid	Guid
☐ SalesOrderDetailID	Int32
☐ SalesOrderID	Int32
☐ SpecialOfferID	Int32
☐ UnitPrice	Currency
☐ UnitPriceDiscount	Currency

Define parameters

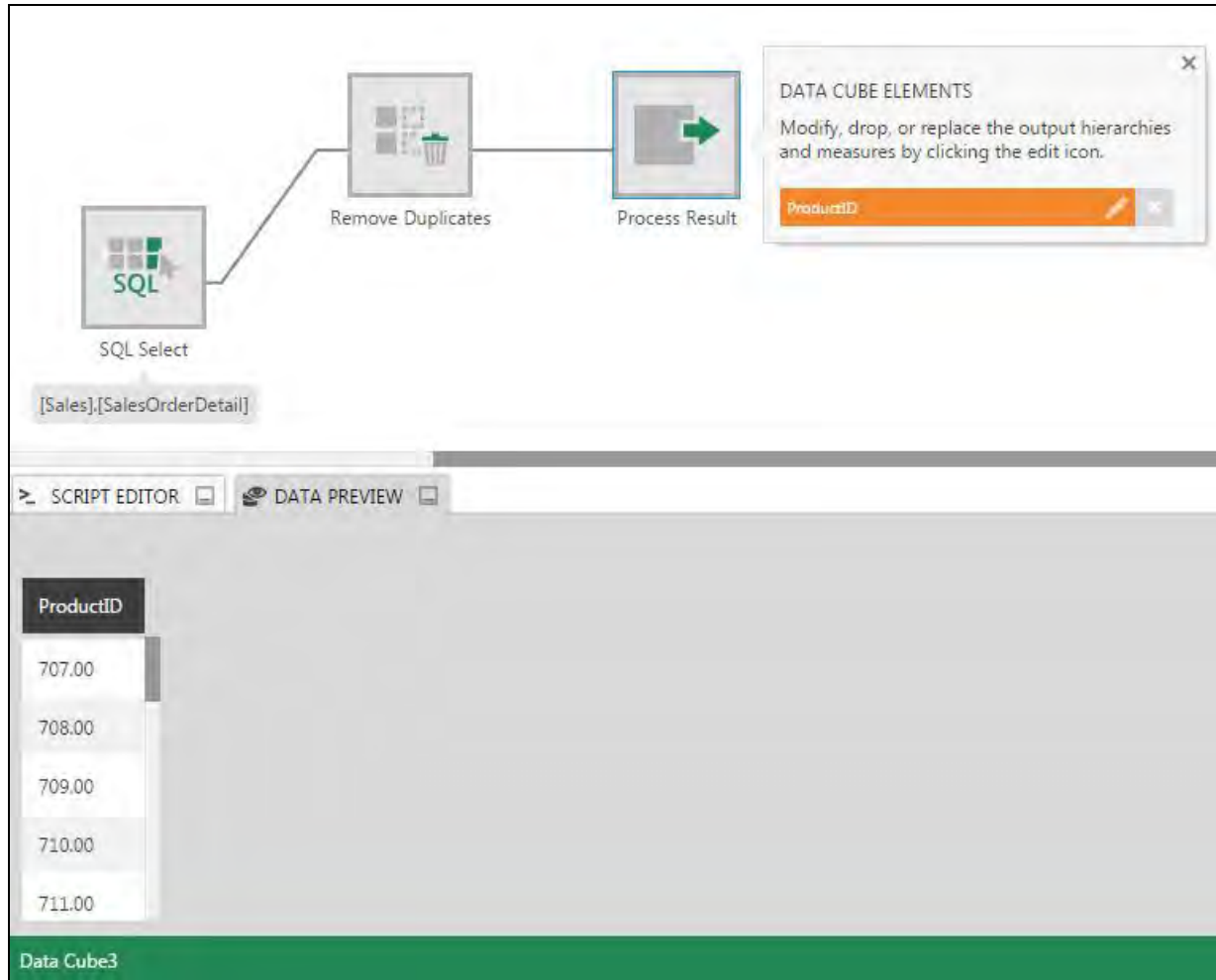
Rename output elements



Output

The output of the **Remove Duplicates** transform consists of the column(s) you selected, where the records have been grouped in order to remove duplicates.

Select the **Process Result** transform and open the **Data Preview** to see the output of the data cube.



The screenshot displays the Logi Symphony interface. At the top, a data pipeline is visible with three transforms: **SQL Select**, **Remove Duplicates**, and **Process Result**. The **SQL Select** transform is connected to the **Remove Duplicates** transform, which is then connected to the **Process Result** transform. Below the **SQL Select** transform, the query `[Sales].[SalesOrderDetail]` is shown. To the right of the **Process Result** transform, a **DATA CUBE ELEMENTS** window is open, showing a list of elements with **ProductID** selected. Below the pipeline, the **SCRIPT EDITOR** and **DATA PREVIEW** tabs are visible. The **DATA PREVIEW** tab is active, showing a table with the following data:

ProductID
707.00
708.00
709.00
710.00
711.00

At the bottom of the interface, a green bar indicates the **Data Cube3** is active.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Percentage Sampling](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Replace '(Empty String)' Text With Empty Value

This applies to: *Managed Dashboards, Managed Reports*

When the string manipulation transform is used in the data cube, empty string values (but not *null* values) are replaced by the text (*Empty string*). This article shows how to customize this text.

Replacing '(Empty string)' Text

The '(Empty string)' text can be replaced for the whole Symphony instance, which will affect all data containing an empty string. This can be done by using the [localization](#) functionality of Symphony.

1. Access [Administration](#) from your Profile as a system administrator, then expand **Setup** and then click **Localization**.
2. Select the localization Dundas.Core and click **Copy** in the toolbar. Enter a **Target Culture**, such as en to override the built-in localization provided for **English**, then submit the dialog.
3. Select the new Dundas.Core localization override and click **Details** in the toolbar.
4. Change the <localization/> tag to have separate open and close tags: <localization ...> and </localization>. In the middle, add a new element:

```
<string key="GS_ENT_HC_EmptyString"><sp/></string>
```

This displays a space character to identify an empty string, or you can customize the text.



Localization

[more help](#)

Update a localization for a specific culture and module. A localization will modify all strings defined in it when a session with the matching culture is used. Any modifications to existing localizations may not be immediately reflected in the UI due to caching.

Culture Name

Name

Module ID

Override

☒

Upload a file to use for the contents, or copy and paste the contents as text below.

No file chosen

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <localization xmlns="http://dundas.com/schemas/BI/localization.xsd" culture="en" moduleId="EC0
3   <string key="GS_ENT_HC_EmptyString"><sp/></string>
4 </localization>

```

5. Submit the dialog. Because of caching, you may need to [recycle the application pool](#) if you want to see the changes take effect immediately.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [String](#)
- [Create a New Data Cube](#)
- [Localization and Multi-Language Support](#)

Re-Wire the Data Flow

This applies to: *Managed Dashboards, Managed Reports*

This article shows how to change the data sources and structures used behind existing data cubes and metric sets. This allows you, for example, to migrate to a new database without having to rebuild the dashboards.

The following examples show how you can:

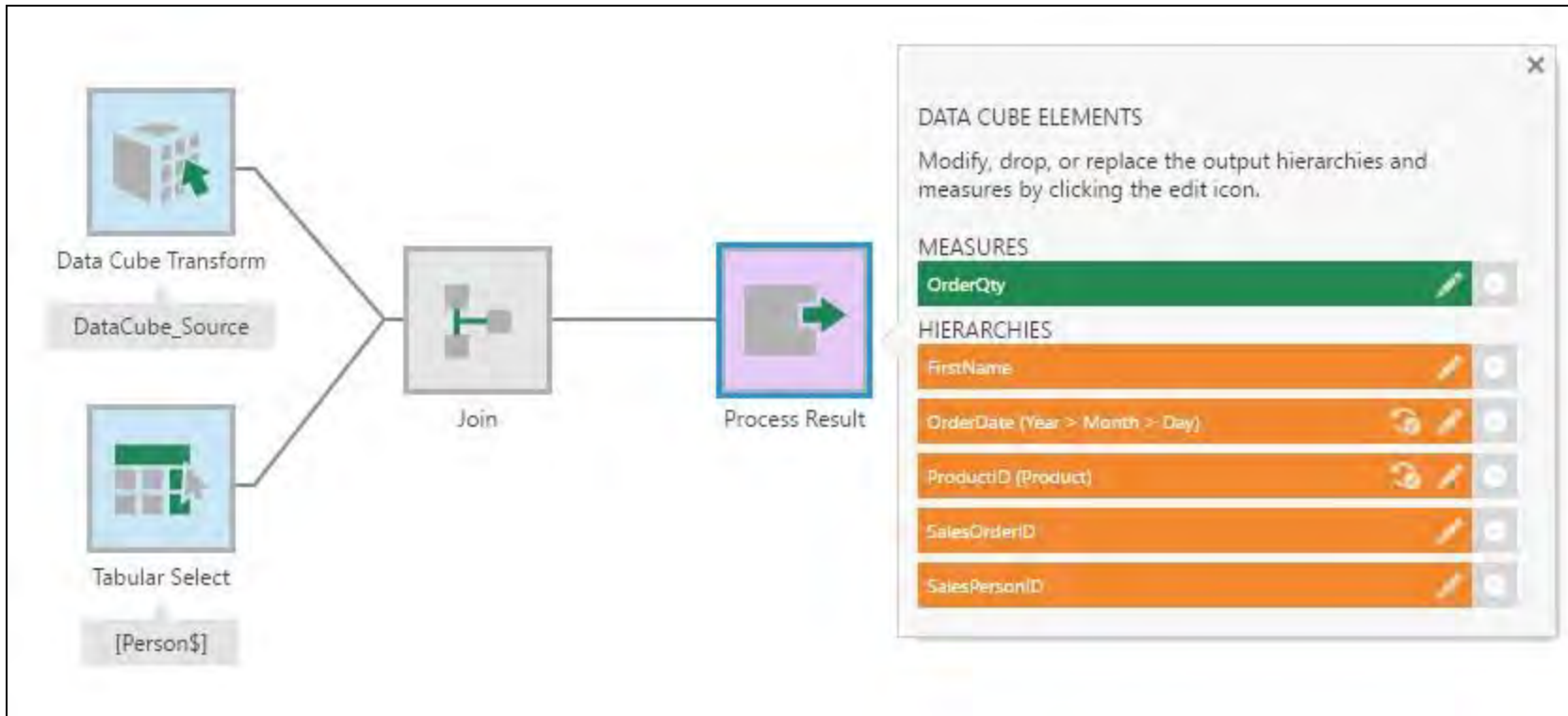
- Drag data from a different data connector onto a data cube select transform to replace it.
- Replace the data cube or data connector used as a data source for a metric set.



Note: You can also open and edit a [data connector](#) to point to a different version of the data source, but be sure its structure including table and column names matches so that data cubes and metric sets continue to work or click to [Change Data Provider](#) if needed.

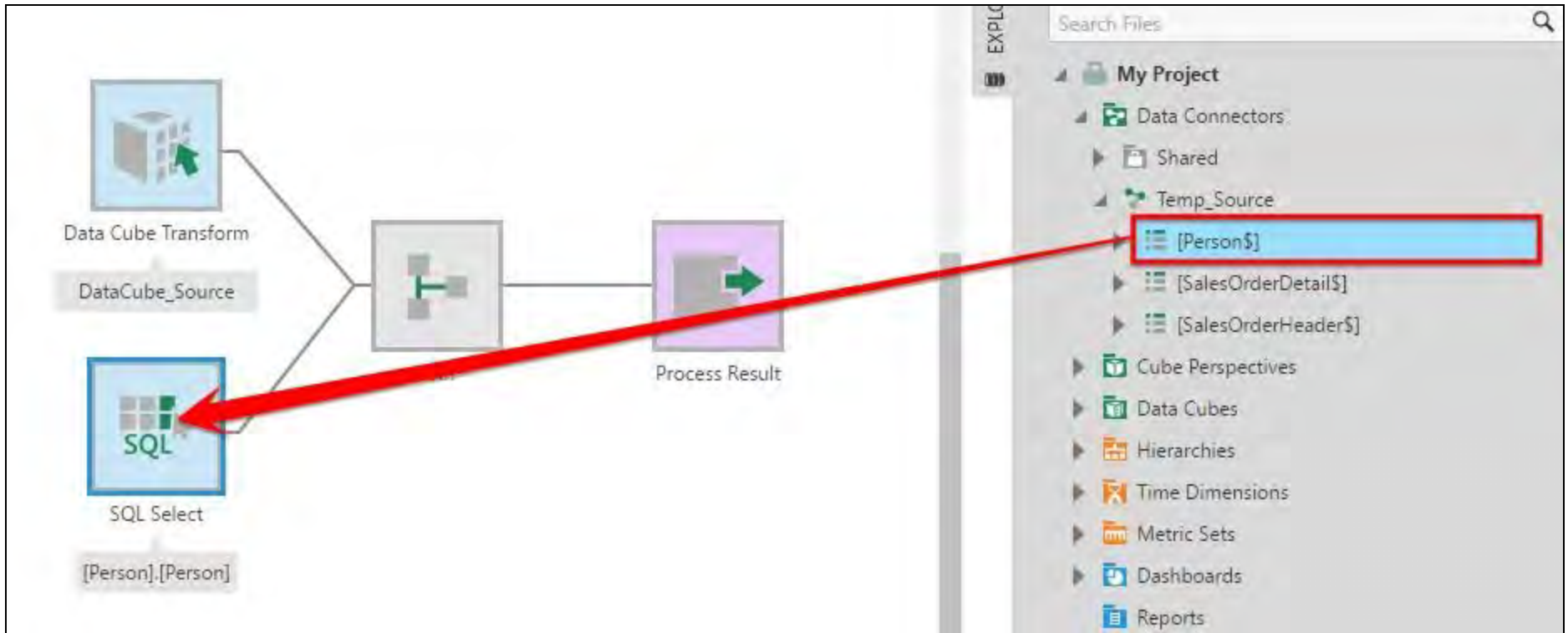
Replace a Select Transform

Open the data cube you would like to rewire. As an example, the data cube pictured below contains a [SQL Select](#) referring to a database table.



In the Explore window, drag a table or other data structure from a different data connector that contains similar data, then drop it onto the select transform you want to replace it with.

In this example, we are dragging a sheet from under an [Excel data connector](#).



In the **Re-Wire Elements** dialog that appears, use the drop-downs on the right to select the matching element from your new data source for each existing element, or confirm that their existing selections are correct.

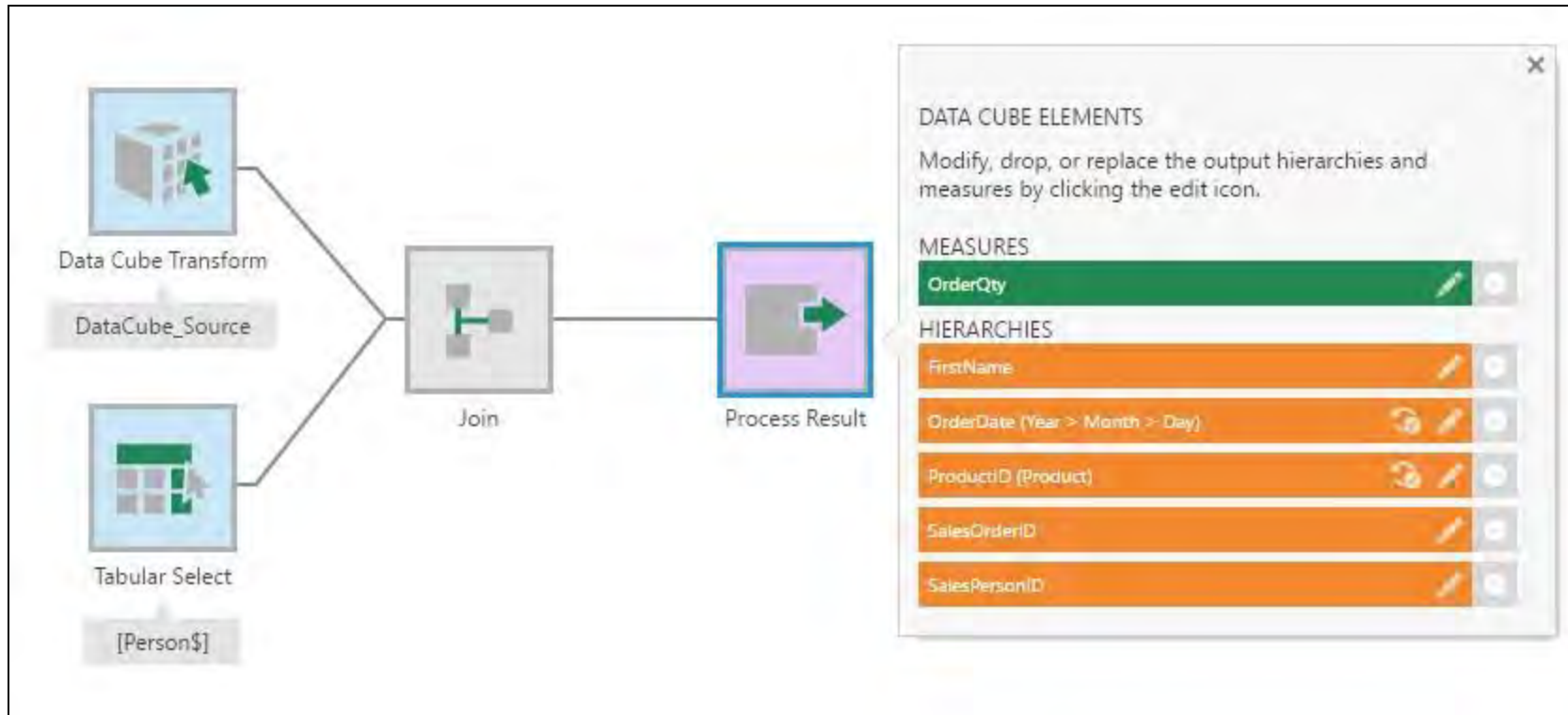
Re-Wire Elements

Replace the existing select transform node with the new native structure. To do this, map each of the existing input connectors to the new native structure source element name.

Connector Name	Data Type	Native Source Element Name
FirstName	String(50)	FirstName (String)
MiddleName	String(50)	MiddleName (String)
LastName	String(50)	LastName (String)
Suffix	String(10)	Suffix (String)
EmailPromotion	Int32	EmailPromotion (Double)
AdditionalContactInfo	String	AdditionalContactInfo (String)
Demographics	String	Demographics (String)
rowguid	Guid	rowguid (Guid)
ModifiedDate	DateTime	ModifiedDate (DateTime)

Click the submit button at the bottom of the dialog when done.

In our example, the SQL Select is now replaced with a [Tabular Select](#) referring to the Excel file data. The Join and Process Result transforms to the right remain intact along with their previously applied settings.



The Process Result's element names remain the same for compatibility with metric sets using them.

Edit the Input Elements

You can edit the matching of elements afterward.

Double-click the new select transform to configure it, then scroll down and click **Input elements**.

Name
[Person\$]

Description

Input
Tabular Select Input

☒ AdditionalContactInfo	String
☒ BusinessEntityID	Double
☒ Demographics	String
☒ EmailPromotion	Double
☒ FirstName	String
☒ LastName	String
☐ LegalFirstName	String
☒ MiddleName	String
☒ ModifiedDate	DateTime
☒ NameStyle	Boolean
☒ PersonType	String
☒ rowguid	Guid
☒ Suffix	String
☒ Title	String

▼ Settings

Command Timeout
30

Define parameters 



Note: Some transform types such as Data Cube and Stored Procedure do not allow for this option after re-wiring.

In our example from the previous section, the Excel file had an extra **LegalFirstName** column not present in the original database table, which is why it is initially unchecked in the select transform configuration as shown above.

We can locate the row for **FirstName** in the Input Elements dialog and change its drop-down to match it instead with **LegalFirstName**.

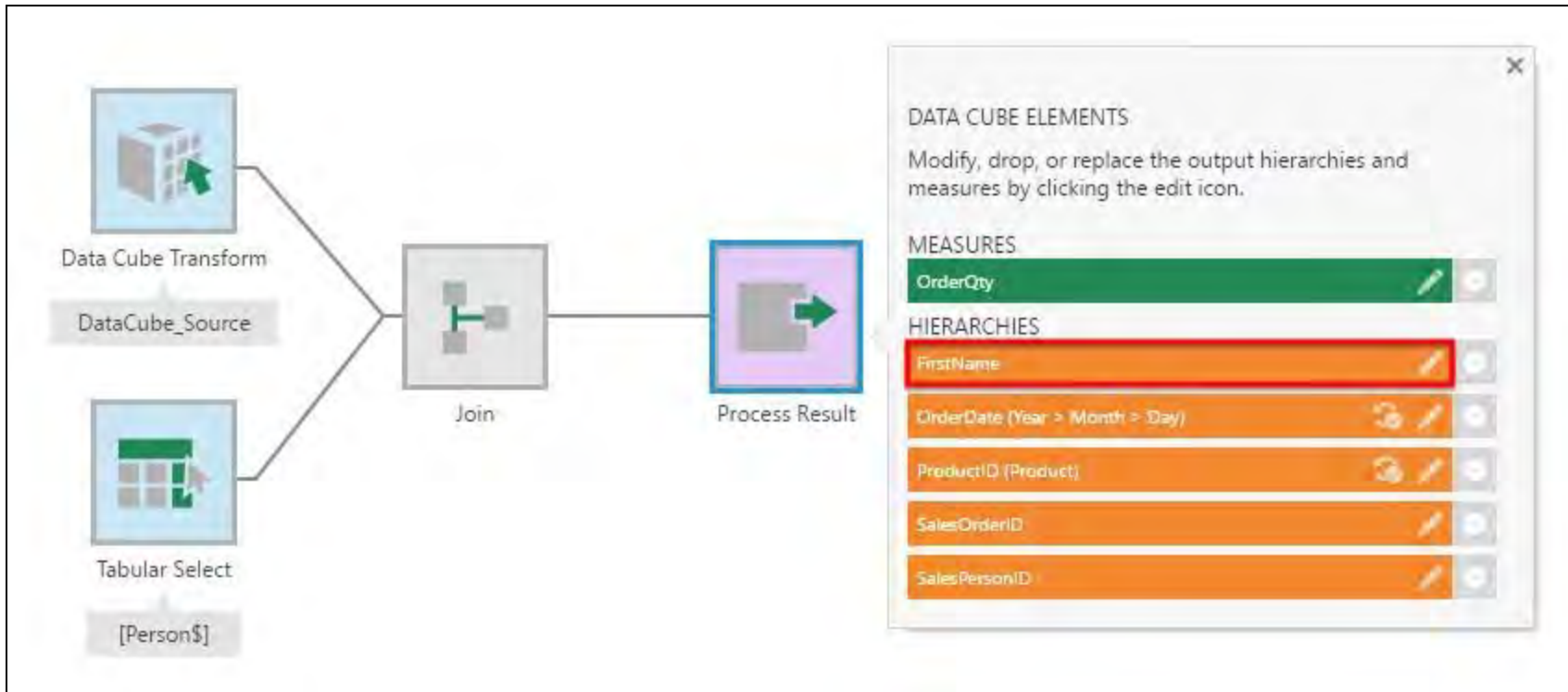
Input Elements

Edit the transform's input elements to change which source native element they are using, if possible. For some sources, such as stored procedures, this is not possible.

Connector Name	Data Type	Native Source Element Name
BusinessEntityID	Double	BusinessEntityID (Double)
PersonType	String	PersonType (String)
NameStyle	Boolean	NameStyle (Boolean)
Title	String	Title (String)
FirstName	String	LegalFirstName (String)
MiddleName	String	MiddleName (String)
LastName	String	LastName (String)
Suffix	String	Suffix (String)
EmailPromotion	Double	EmailPromotion (Double)

Submit the dialog.

The hierarchy is still named FirstName although the data now comes from the LegalFirstName column.



Re-wiring a Manual Select's Data Connector

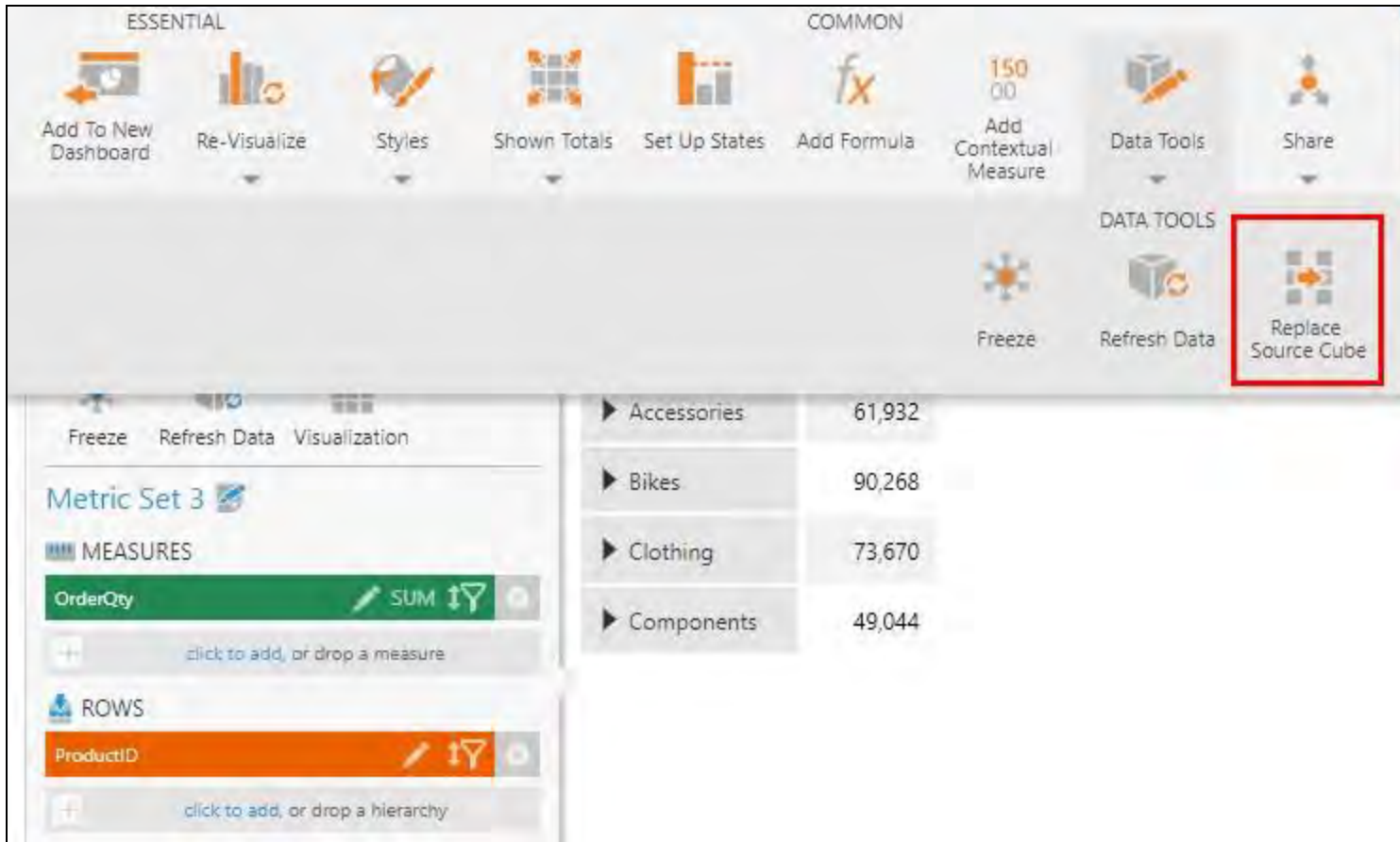
If you are currently using a [Manual Select](#) in your data cube with your own SQL or MDX query, dragging a table, sheet, or OLAP cube onto the Manual Select as shown above will replace it and your query with a [SQL Select](#) or [MDX Select](#) that provides the specific data you dragged as input.

You can also maintain your Manual Select transform and its query, but change the data connector it executes against: drag a data connector from the Explore window and drop it onto the Manual Select transform. This operation requires that the same elements be returned when running the query against the new data connector as with the original data connector.

Re-Wire Metric Set Elements

To replace the data cube or source used behind a metric set, open it in the full screen editor from the [main menu](#) or [Explore window](#), or choose this option from the Data Analysis Panel or from Data Tools in the toolbar when using the metric set on a dashboard.

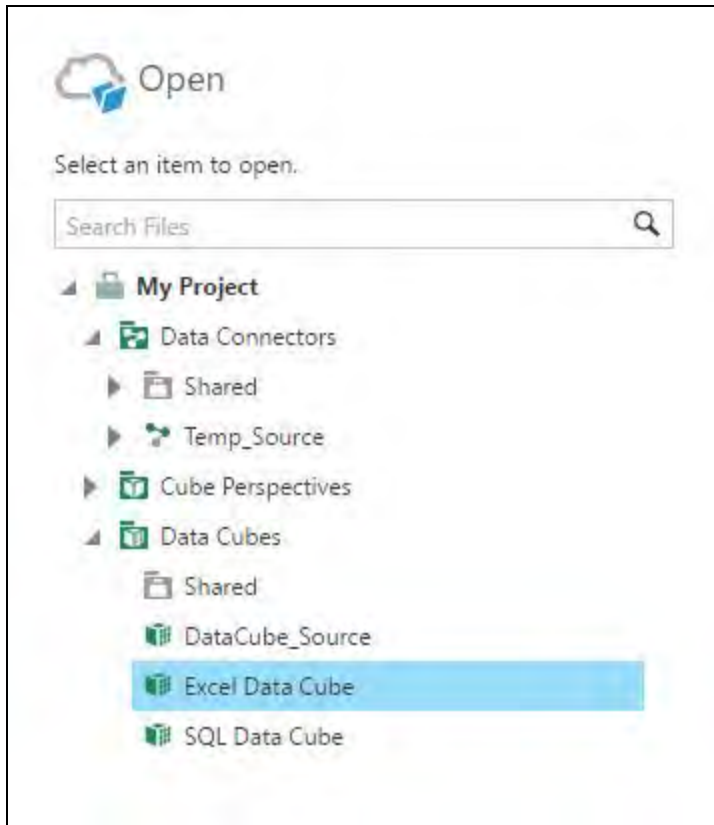
In the metric set's toolbar, choose **Data Tools**, then **Replace Source Cube**.



The screenshot displays the Logi Symphony v24 interface. The top navigation bar is divided into 'ESSENTIAL' and 'COMMON' sections. The 'ESSENTIAL' section includes icons for 'Add To New Dashboard', 'Re-Visualize', 'Styles', 'Shown Totals', 'Set Up States', 'Add Formula', 'Add Contextual Measure', 'Data Tools', and 'Share'. The 'COMMON' section includes 'Freeze', 'Refresh Data', and 'Replace Source Cube'. The 'Replace Source Cube' button is highlighted with a red box. Below the navigation bar, the 'Metric Set 3' panel is visible, showing a table with columns for 'Accessories', 'Bikes', 'Clothing', and 'Components'. The 'MEASURES' section shows 'OrderQty' with a 'SUM' aggregation. The 'ROWS' section shows 'ProductID'.

Category	Count
Accessories	61,932
Bikes	90,268
Clothing	73,670
Components	49,044

Select a data cube, or you can choose another data source such as database table or spreadsheet and a data cube will be auto-generated for it. Click the submit button at the bottom of the dialog.



In the **Re-Wire Source Cube** dialog, use the Replacement Structure Element Name drop-downs to match the existing elements to the new source elements.

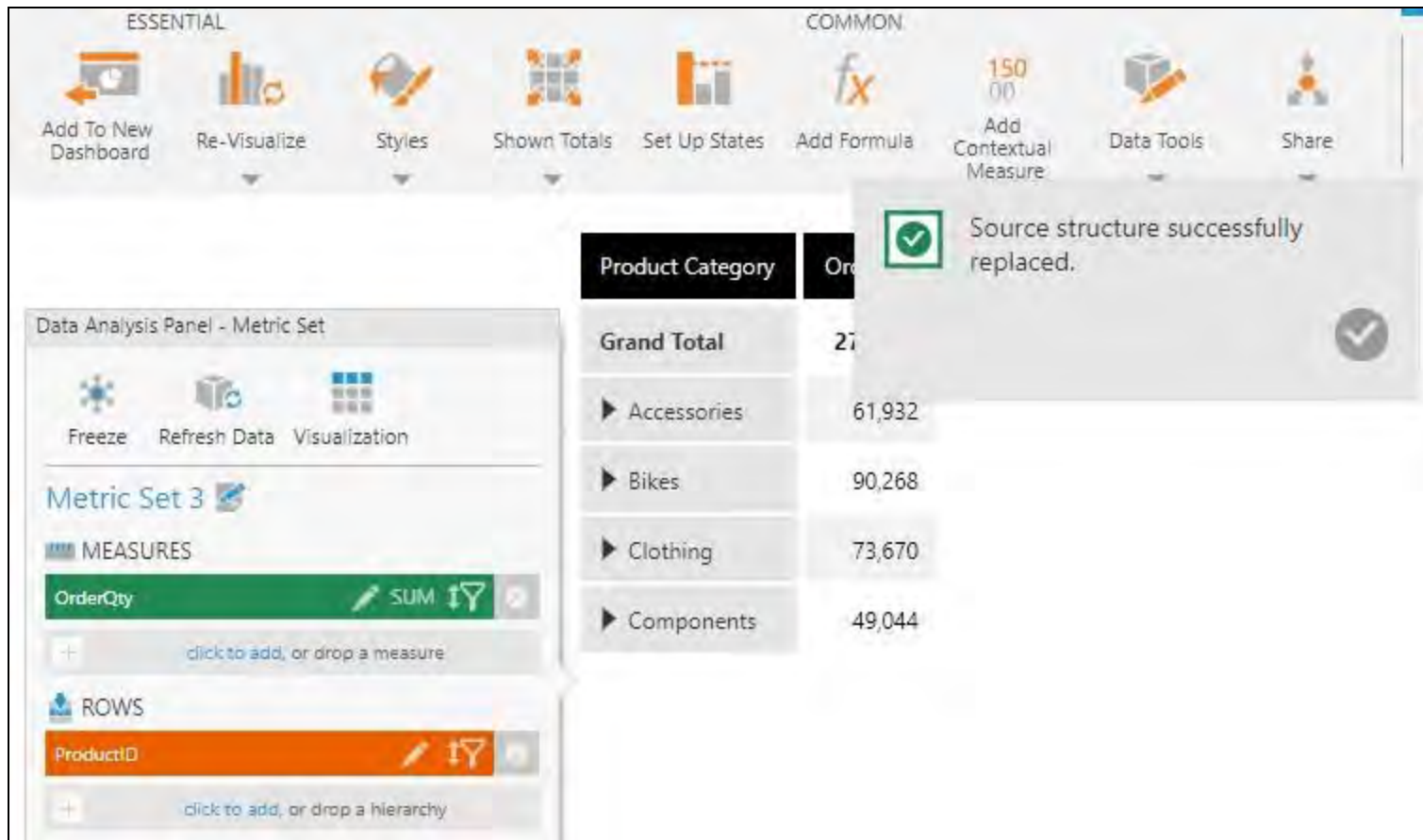
Re-Wire Source Cube

Replace the existing source cube that is powering the metric set with a new structure. To do this, map each of the existing measures and hierarchies to the new structure source element name.

Element Name	Position	Replacement Structure Element Name
OrderQty	Measure	OrderQty (OrderQty) ▼
ProductID (Product)	Rows	ProductID (Product) (ProductID) ▼

Submit the dialog when done.

A dialog will appear confirming that the source was replaced. The metric set's elements and their settings remain intact.



The screenshot displays the Logi Symphony v24 interface. At the top, there are two tabs: 'ESSENTIAL' and 'COMMON'. The 'ESSENTIAL' tab contains icons for 'Add To New Dashboard', 'Re-Visualize', 'Styles', 'Shown Totals', 'Set Up States', 'Add Formula', 'Add Contextual Measure', 'Data Tools', and 'Share'. The 'COMMON' tab contains a '15000' icon. Below the tabs, a 'Data Analysis Panel - Metric Set' is visible on the left. It includes a 'Metric Set 3' section with a 'MEASURES' list containing 'OrderQty' (SUM) and a 'ROWS' list containing 'ProductID'. A table is displayed in the center, showing 'Product Category' and 'OrderQty' columns. The table data is as follows:

Product Category	OrderQty
Grand Total	27
Accessories	61,932
Bikes	90,268
Clothing	73,670
Components	49,044

A notification box in the top right corner states: 'Source structure successfully replaced.' with a green checkmark icon.

For more information, see:

- [The Managed Dashboards Data Model](#)
- [Data Cubes](#)
- [Symphony Metric Sets](#)
- [Create a Metric Set and Add a Filter](#)
- [Promoting an Auto-Generated Metric Set](#)

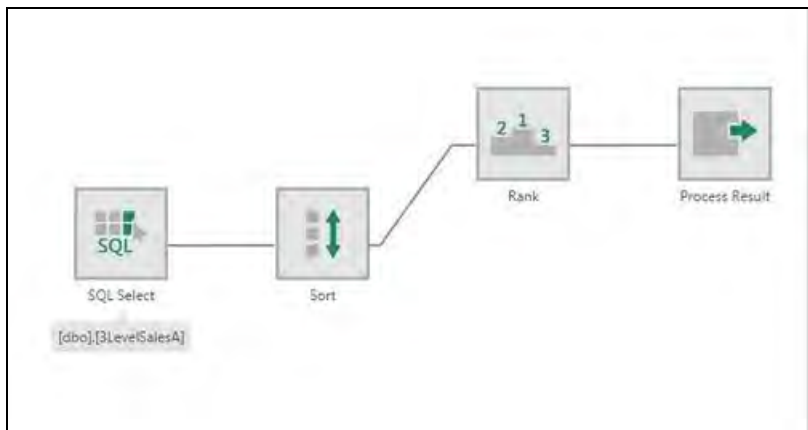
Sort

This applies to: *Managed Dashboards, Managed Reports*

The **Sort** transform allows various sorting rules to be applied sort the input data.

The sorting is only applicable inside the data cube, since **metric sets** have their own sorting. (The **Sort** transform should never be the last transform before the **Process Result** as this will result in unnecessary processing.)

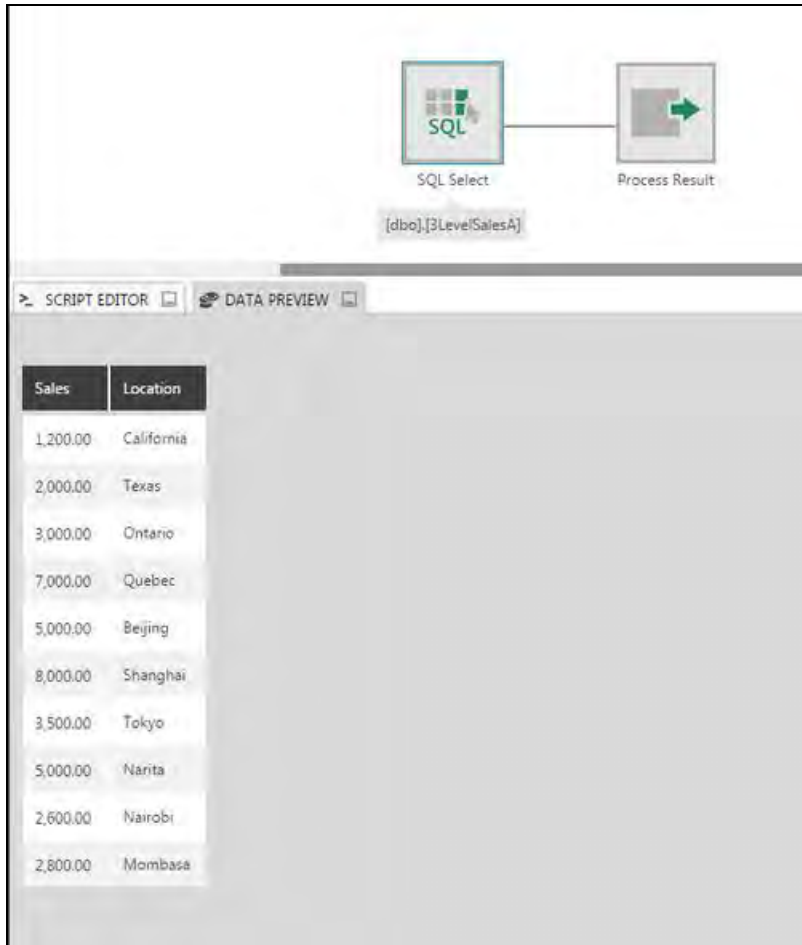
If the data provider used to select data supports sorting, and no preceding transform must execute within Symphony, the sorting will be automatically included in the query sent to the database.



Input

The **Sort** transform requires one input transform that has at least one column.

The input could be a SQL Select transform, or the result of another transform. For example, a table showing **Location** and **Sales**.



The screenshot shows the Logi Symphony v24 interface. At the top, there is a workflow diagram with two components: 'SQL Select' and 'Process Result', connected by a line. Below the 'SQL Select' component, the data source '[dbo].[3LevelSalesA]' is specified. The interface has two tabs: 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table with two columns: 'Sales' and 'Location'. The table contains the following data:

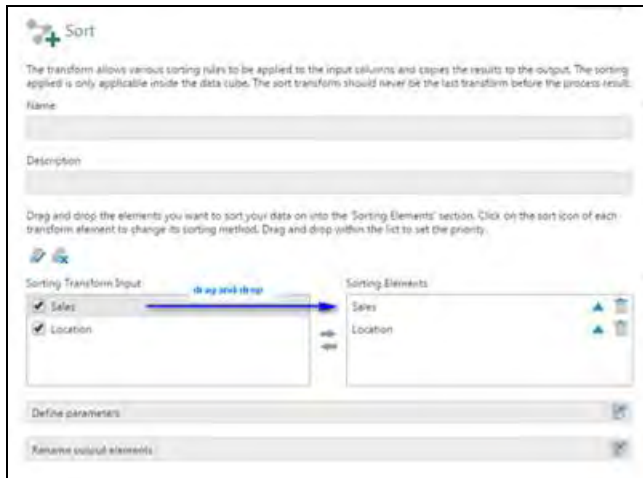
Sales	Location
1,200.00	California
2,000.00	Texas
3,000.00	Ontario
7,000.00	Quebec
5,000.00	Beijing
8,000.00	Shanghai
3,500.00	Tokyo
5,000.00	Narita
2,600.00	Nairobi
2,800.00	Mombasa

Configure

In the **Sort** transform configuration dialog, uncheck any columns you don't want included in the output.



Drag and drop the elements containing the values you want to determine the sorting from under **Sorting Transform Input** to **Sorting Elements**.



Click the arrow icon to toggle between **Ascending** (up arrow) and **Descending** (down arrow).


Sort

The transform allows various sorting rules to be applied to the input columns and copies the results to the output. The sorting applied is only applicable inside the data cube. The sort transform should never be the last transform before the process result.

Name

Description

Drag and drop the elements you want to sort your data on into the 'Sorting Elements' section. Click on the sort icon of each transform element to change its sorting method. Drag and drop within the list to set the priority.

Sorting Transform Input

- ☒ Sales
- ☒ Location

Sorting Elements

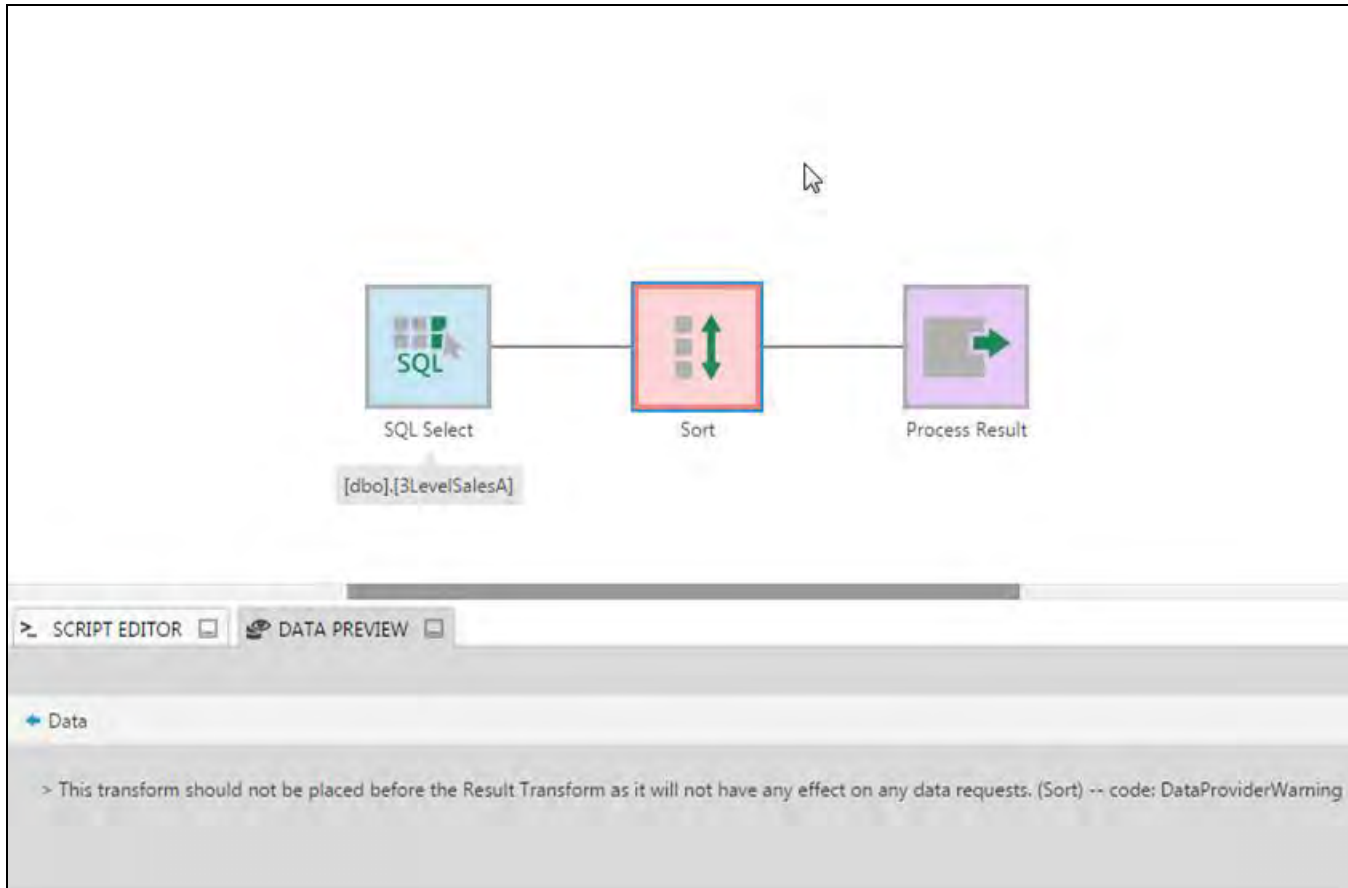
- Sales
- Location

Define parameters

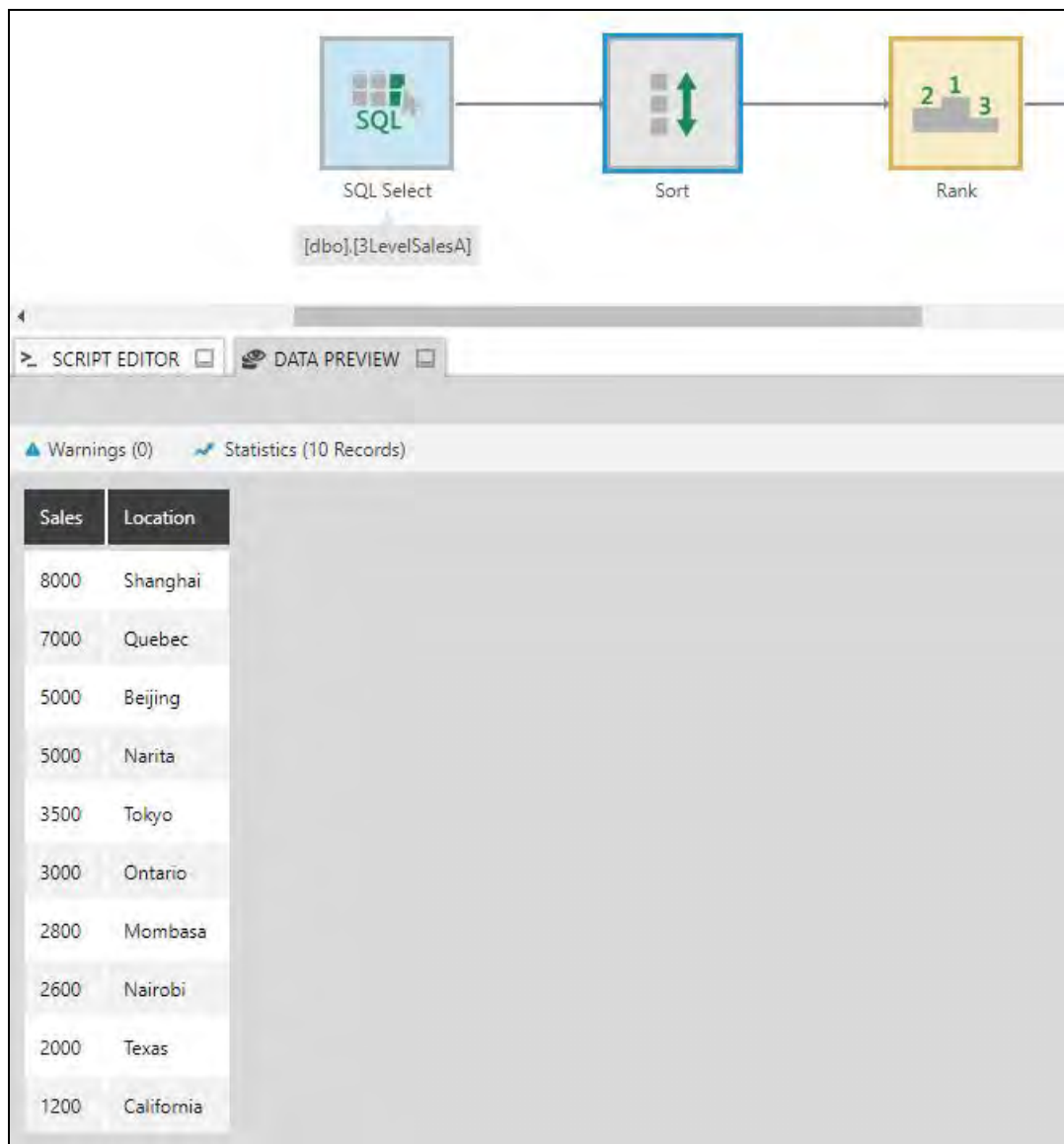
Rename output elements

Output

If you don't have another transform after the Sort transform other than the Process Result, the Sort transform will appear in red and you will not be able to preview its output. This is because metric sets have their own sorting, and sorting within the data cube has no effect other than to change the input for some other transform.



If there is another transform besides Process Result connected to the output, you can select the Sort transform and open **Data Preview** to view its output. The figure below illustrates the data sorted by **Sales** in descending order, and then by **Location** in ascending order.



The screenshot displays the Logi Symphony v24 interface. At the top, a workflow diagram shows three steps: 'SQL Select' (blue box), 'Sort' (blue box), and 'Rank' (yellow box). The 'SQL Select' step is connected to a data source labeled '[dbo].[3LevelSalesA]'. The 'Sort' step is connected to the 'SQL Select' step, and the 'Rank' step is connected to the 'Sort' step. Below the workflow, the 'DATA PREVIEW' tab is active, showing a table with two columns: 'Sales' and 'Location'. The table contains 10 rows of data, sorted by the 'Sales' column in descending order. The 'Warnings (0)' and 'Statistics (10 Records)' indicators are visible above the table.

Sales	Location
8000	Shanghai
7000	Quebec
5000	Beijing
5000	Narita
3500	Tokyo
3000	Ontario
2800	Mombasa
2600	Nairobi
2000	Texas
1200	California

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)
- [Select and Group Data for a Metric Set](#)

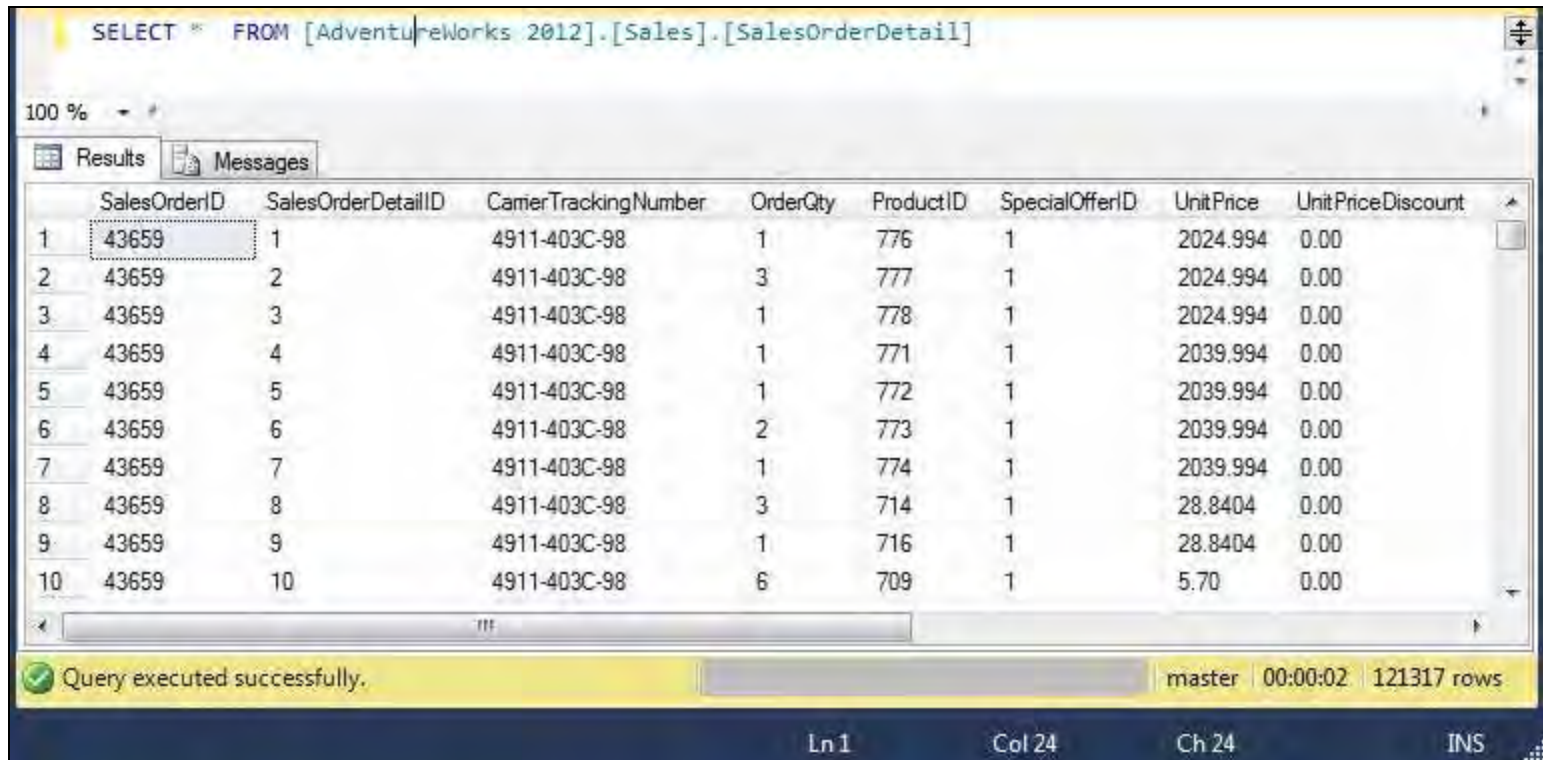
SQL Select

This applies to: *Managed Dashboards, Managed Reports*

The **SQL Select** transform is created when a structure is dragged onto the canvas from a data connector that supports relational queries. You can select the columns to be used in the data cube and set up aggregators. When retrieving data, these settings will then be translated into a SOL query and sent to the data source.

Input

The structure of the table dropped on the data cube canvas is used as input. The examples in this article are based on the following data:

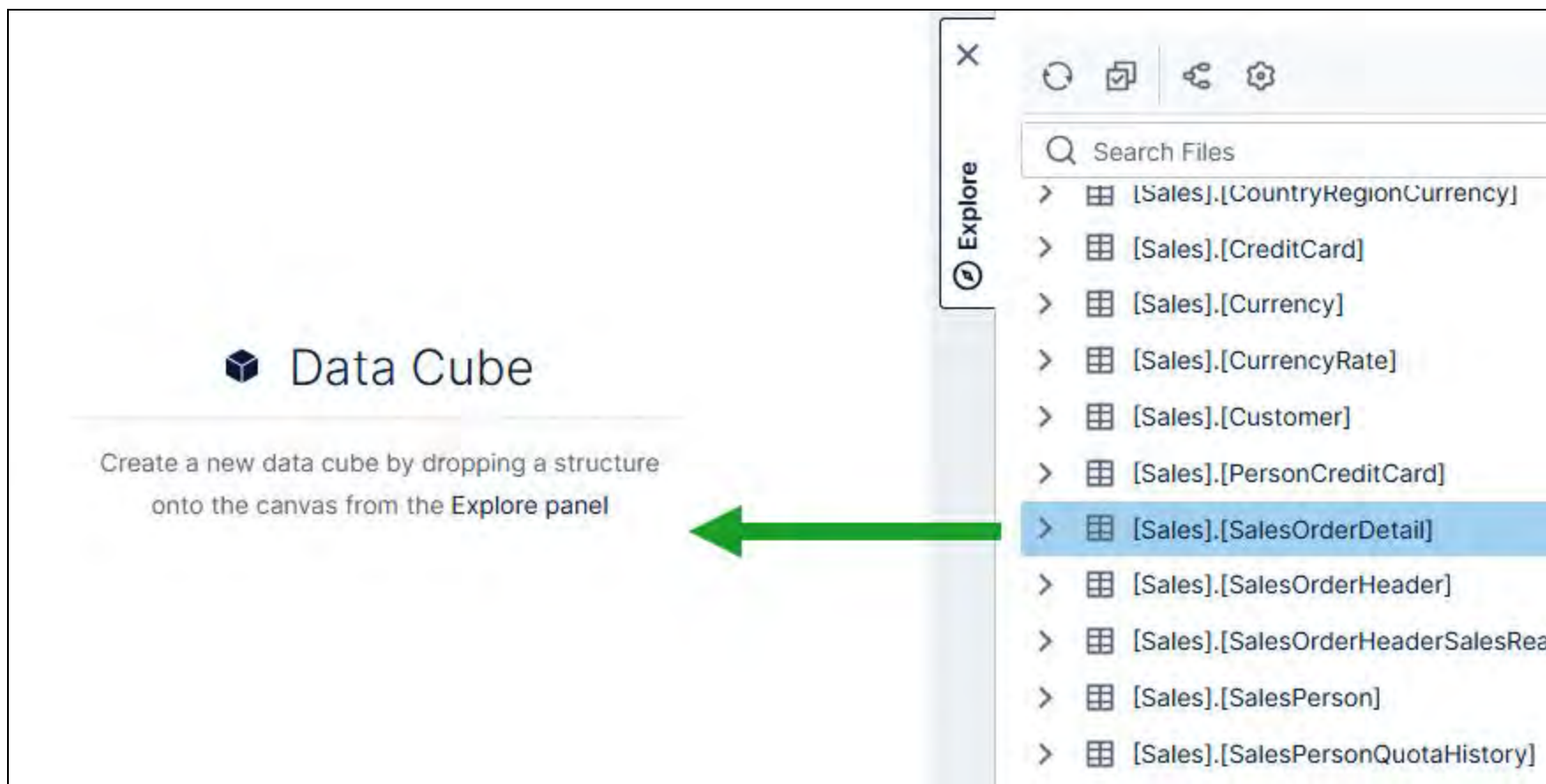


	SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount
1	43659	1	4911-403C-98	1	776	1	2024.994	0.00
2	43659	2	4911-403C-98	3	777	1	2024.994	0.00
3	43659	3	4911-403C-98	1	778	1	2024.994	0.00
4	43659	4	4911-403C-98	1	771	1	2039.994	0.00
5	43659	5	4911-403C-98	1	772	1	2039.994	0.00
6	43659	6	4911-403C-98	2	773	1	2039.994	0.00
7	43659	7	4911-403C-98	1	774	1	2039.994	0.00
8	43659	8	4911-403C-98	3	714	1	28.8404	0.00
9	43659	9	4911-403C-98	1	716	1	28.8404	0.00
10	43659	10	4911-403C-98	6	709	1	5.70	0.00

Query executed successfully. master 00:00:02 121317 rows

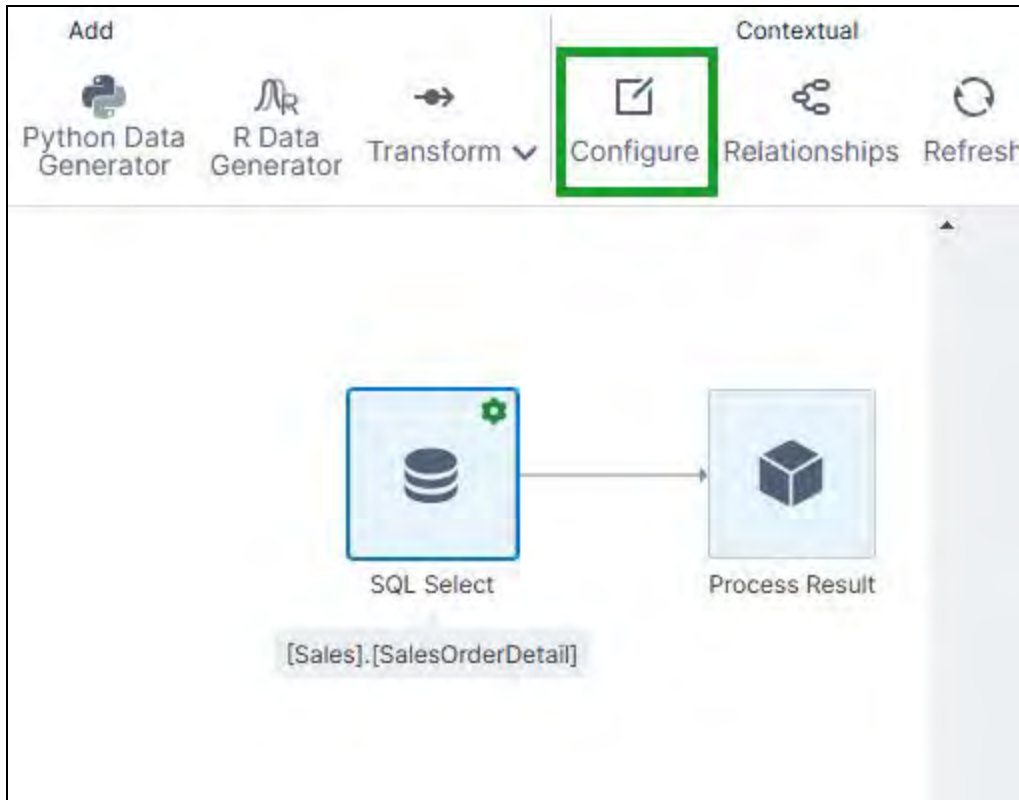
Add the Transform

Expand the data connector in the **Data Connectors** folder in the **Explore** window, then drag a structure (e.g., table or view) to the canvas.



Configure

To configure the transform, double-click it, or select it and click **Configure** in the toolbar.



The **SQL Select** configuration dialog appears.

Help 

SQL Select

The relational SQL select transform, responsible for basic data querying.

Name

Color

[Sales].[SalesOrderDetail]



Description

ID

a6e62017-ecdd-486f-8b19-939e047799d5

Settings

Table Name

[Sales].[SalesOrderDetail]

Data Connector

AdventureWorks16

Add Counter Column

☐

Timeout

30



- Archive of documentation for Logi Symphony v24

Optionally update the **Name**, select a **Color** for the transform, and add a **Description**.

Select the **Add Counter Column** option to group by the values in each existing column and add a new column containing the corresponding number of records for each set of values.

The **Timeout** (in seconds) can be customized to determine how long to wait before terminating a long-running database command.



Note: The Default Query Timeout application configuration setting determines the initial timeout setting.

Transform Element

Click the Edit icon next to a column name to configure it and access different options that may be available depending on the data type and data provider.



Transform Element

Configure the settings for this transform element. The settings are specific to the transform type, and will change the behavior for this element.

Group By

☐

Aggregate Function

Date Trimming Functions

- No TRIM
- TRIM(minute)
- TRIM(hour)
- TRIM(day)
- TRIM(month)
- TRIM(year)

The list of available **Aggregate Function** options reflects the functions supported by the data provider. For example, numeric columns from SQL Server support STDEV, STDEVP, VAR, and VARP in addition to the Symphony built-in aggregate functions.

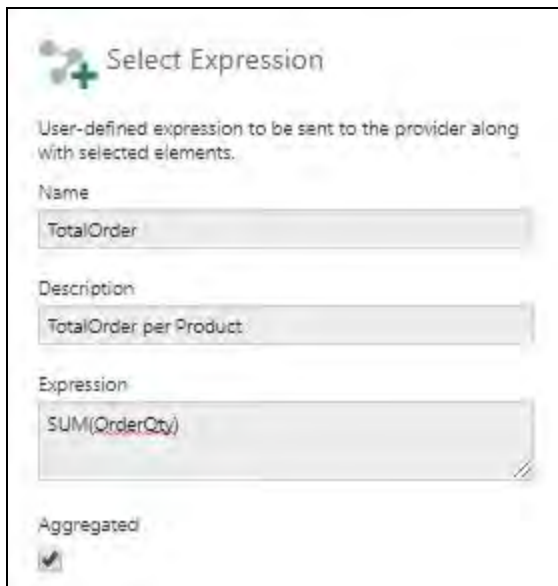


Note: When a numeric column is assigned an aggregator, the output will be grouped by other included columns even if their **Group By** options are not checked unless they are aggregated.

There may also be **Date Trimming Functions** available for date/time data. For example, these can be used along with the Group By & Aggregate Function options or an [Aggregate](#) transform to group by the selected time period.

Add Expression

Add a new element by inputting a provider expression.



You can also use a Scalar-Valued function in the Expression:

```
dbo. ISOweek (DATEADD (month, 3, (ModifiedDate)))
```

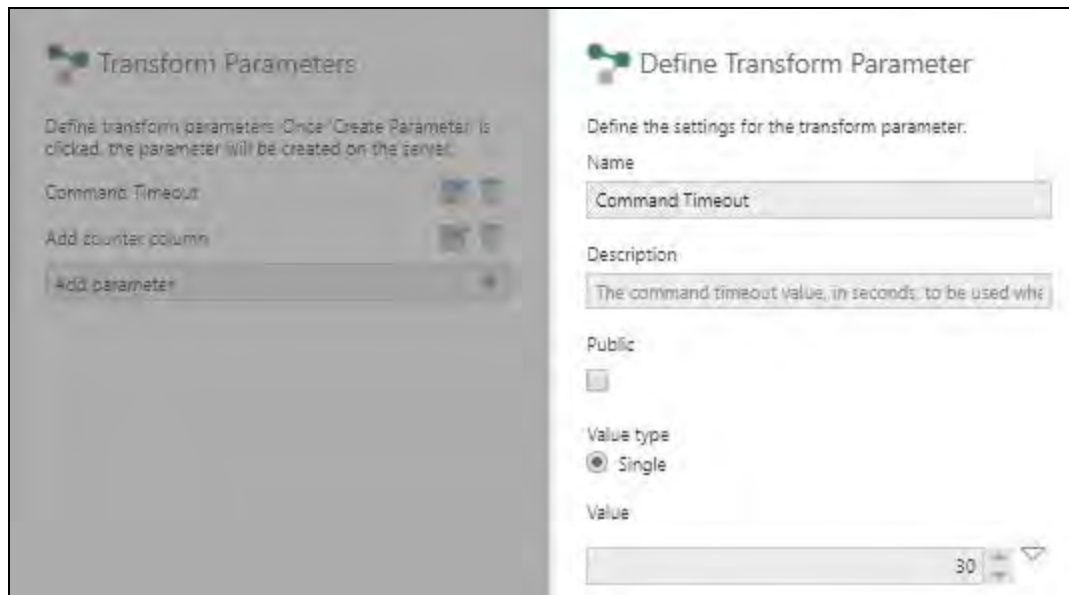
If you're running Symphony v24.4 and later, you can insert placeholders similar to those available for a Manual Select to insert values via parameters. Include a name in your expression enclosed by dollar signs (\$), then select **Placeholders** to set up the corresponding parameter with a matching Identifier. See [Using Manual Select Placeholders](#).

Define Parameters

Parameters allow you to filter the data, or for transform options to be set from outside of the transform. The parameter's value can be changed by a [bridge parameter](#) in this data cube, or if you make the parameter public you can set it from outside the data cube, for example using a filter on a dashboard.

Click **Parameters** in the transform configuration dialog. In the Transform Parameters dialog that opens, click **Add Parameter** and select a parameter type:

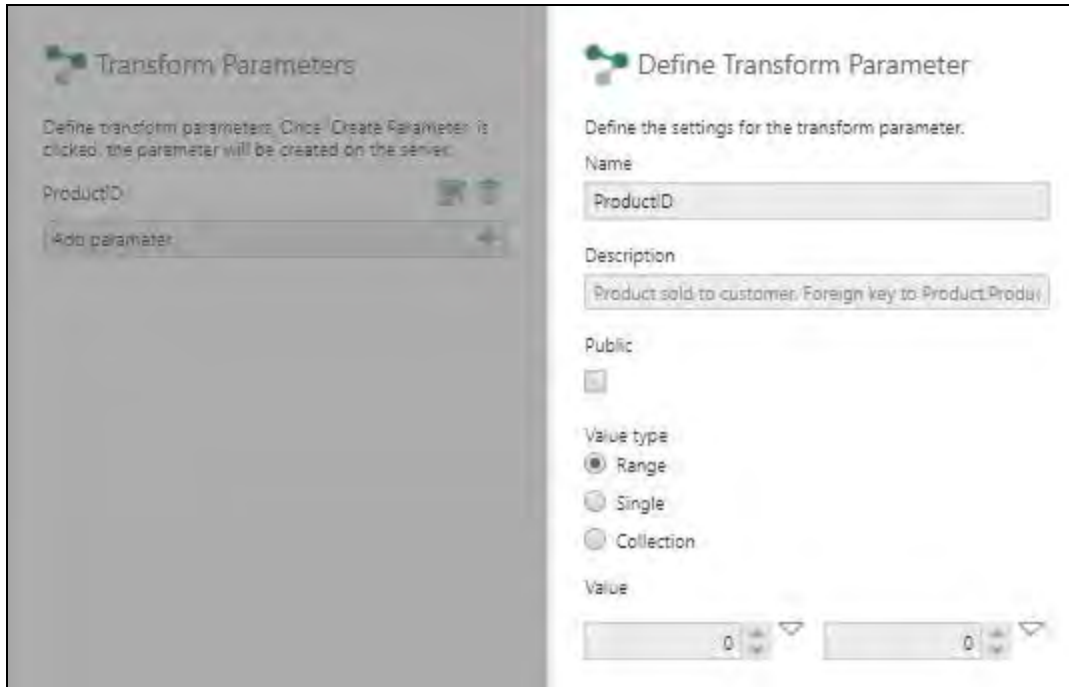
- **Transform Setting** - Add a counter column or command timeout parameter to customize the configuration options above from outside the transform.



- **Output Element Filter** - Add a filter to a column in the transform.



Note: The token menu can be used to select options including [attributes](#) or custom [attributes](#) describing the logged-in user, or whether ranges are inclusive or exclusive.



Select Structure Parameter - Allows you to switch between multiple structures such as tables from the same data connector, provided they have the same column names and types. When creating this type of parameter, click to select which data structures (e.g., tables) are compatible and should be accepted as valid values, then this parameter can be set to the ID of the data structure you want to provide data. To find the structure's ID, find and expand its data connector in the [Explore window](#) to list its structures below, then right-click the structure and open its Properties [dialog](#) to find its ID field.



Transform Parameters

Create, edit and delete the parameters on this transform. Parameters can be created on various settings, such as transform settings for this transform and output element filters.

Select Structure Parameter

Add parameter



Configure Transform Parameter

Set up this transform parameter by specifying what it is linked to when creating it, and then the name, value and type of the parameter value after it has been created. If a parameter is marked as public, then it will be accessible outside of this data cube.

Name

Select Structure Parameter

Description

Parameter that allows the dynamic change of a select transform data source.

Public

☒

Parameter Value And Type

Single

VALID VALUES

[dbo].[DateTime1]

[dbo].[DateTime2]

When parameters are defined, a gear icon will appear in the corner of the transform when viewing the data cube process.

Edit Input Elements

This option in the configuration dialog allows you to change the [source](#) element behind each transform output element.

Edit Output Elements

Use this option to rename the output elements of this transform.

Output

Open the **Data Preview** window to view the transform output.

OrderQty	ProductID	Total_Order
625	925	625
36	902	36
90	710	90
249	879	249
90	733	90
1616	856	1616
346	756	346
2394	779	2394
190	802	190
322	971	322



Note: Certain data types such as `Array` from PostgreSQL or Vertica are not available in any following transforms or the data cube output, but may be available to use in this transform's configuration to filter by its output values using parameters.

For more information, see:

- Transform: [Stored Procedure Select](#)
- Transform: [Manual Select](#)
- [Using Transforms](#)
- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)
- [Re-Wire the Data Flow](#)



Stored Procedure Select

This applies to: *Managed Dashboards, Managed Reports*

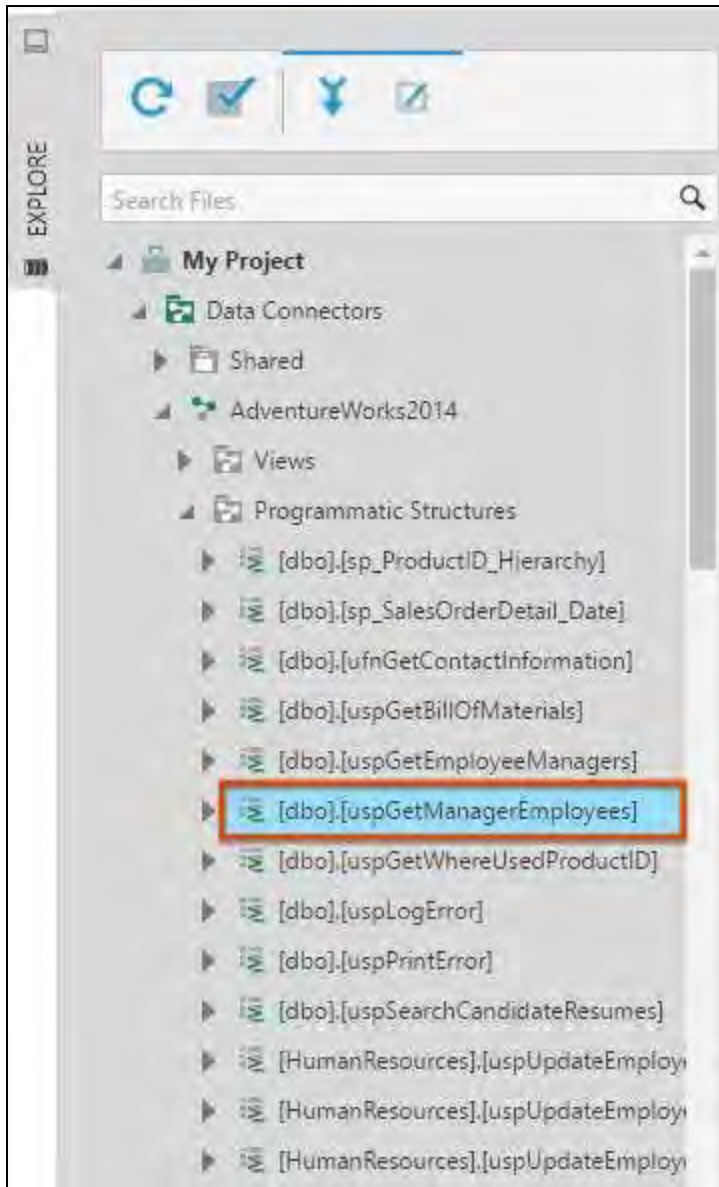
The **Stored Procedure Select** transform lets you retrieve data using a relational database stored procedure. This transform is created automatically when you drag a stored procedure from a data connector in Explore to the data cube canvas.

Input

The Stored Procedure Select transform extracts data via a stored procedure such as **uspGetManagerEmployees** from the [AdventureWorks 2014 sample database](#).

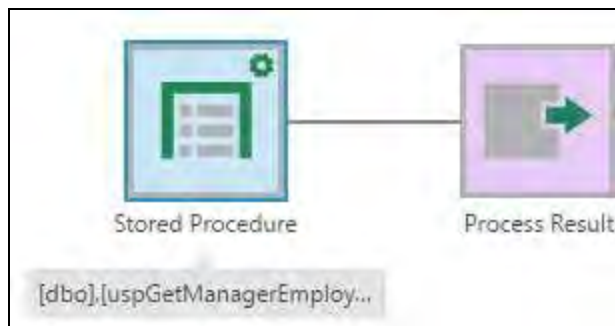
Add the Transform

When editing a data cube, find your data connector in the **Data Connectors** folder in the **Explore** window. Expand it and its **Programmatic Structures** folder, then drag a stored procedure onto the canvas.



Note: The stored procedure must return data in order to use it in a data cube.

A **Stored Procedure** transform appears on the canvas and is connected to a **Process Result** transform. The icon at the top-right of the Stored Procedure transform indicates that it includes a parameter.



Important: The Stored Procedure transform has a default timeout of 30 seconds. If your stored procedure command takes longer to execute, the transform will appear red and will not work. You can configure the transform and increase the timeout.

Configure the Transform

Double-click the Stored Procedure transform, or select it and then choose **Configure** in the toolbar.

In the configuration dialog, the names and data types of the stored procedure's input columns are listed.



Stored Procedure

The transform responsible for retrieving data from a relational stored procedure.

more help

Name

Description

Timeout

Stored Procedure Transform Input

RecursionLevel	Int32
OrganizationNode	String(4000)
ManagerFirstName	String(50)
ManagerLastName	String(50)
BusinessEntityID	Int32
FirstName	String(50)
LastName	String(50)

Define parameters



Rename output elements



STORED PROCEDURE

BRIDGE PARAMETERS

Configure



Stored Procedure

GetManagerEmployees



Process Result

The **Timeout** determines how long in seconds to allow execution to take before terminating it.



- Archive of documentation for Logi Symphony v24



Note: The Default Query Timeout [application configuration setting](#) determines the initial timeout setting.

Click **Define parameters** to configure the stored procedure's parameters.



Stored Procedure

The transform responsible for retrieving data from a relational stored procedure.

Name

Description

Timeout

Stored Procedure Transform Input

RecursionLevel	Int32
OrganizationNode	String(4000)
ManagerFirstName	String(50)
ManagerLastName	String(50)
BusinessEntityID	Int32
FirstName	String(50)
LastName	String(50)

Define parameters

Rename output elements



Transform Parameters

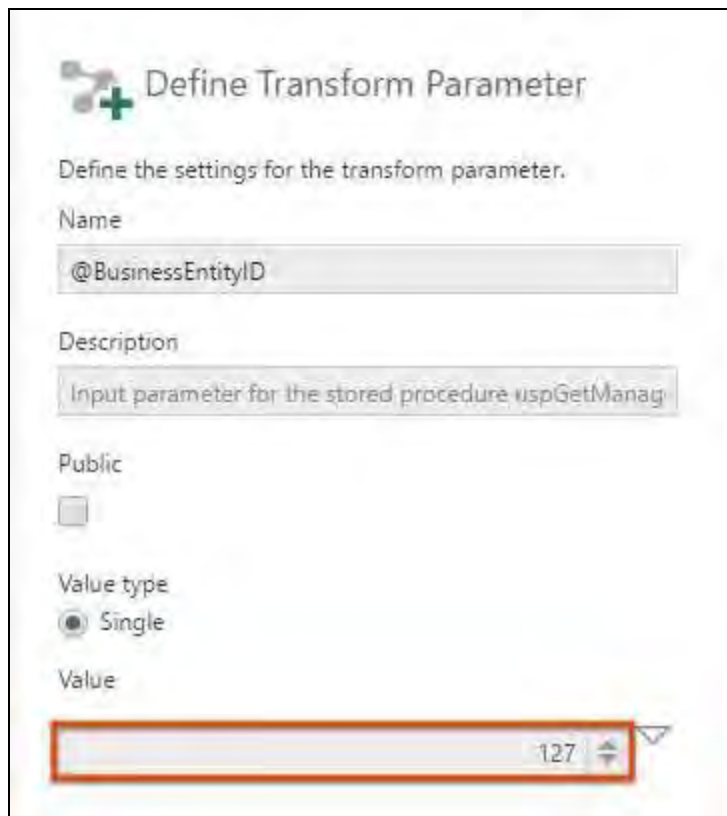
Define transform parameters. Once 'Create Parameter' is clicked, the parameter will be created on the server.

@BusinessEntityID

Add parameter

In the **Transform Parameters** dialog, you will see that parameters are automatically detected and added to the list. Click the **Edit** button for a parameter to configure it.

In the **Define Transform Parameter** dialog, use the **Value** field to indicate a default value for the parameter.



The image shows the 'Define Transform Parameter' dialog box. It has a title bar with a plus icon and the text 'Define Transform Parameter'. Below the title bar, there is a subtitle 'Define the settings for the transform parameter.' followed by several input fields: 'Name' with the value '@BusinessEntityID', 'Description' with the value 'Input parameter for the stored procedure uspGetManag', 'Public' with an unchecked checkbox, 'Value type' with 'Single' selected, and 'Value' with a numeric input field containing '127'. The 'Value' field is highlighted with a red border.



Note: Like in other transform parameters, you can use the menu to the right of the Value field to choose to pass the value of an attribute or [custom attribute](#) associated with each user.

To allow the parameter to be set or attached to filters in metric sets, dashboards, etc., check **Public**.

Output

Select the Process Result transform on the canvas and then go to **Data Preview** to see the output from the stored procedure.



Stored Procedure

[dbo].[uspGetManagerEmploy

X

DATA CUBE ELEMENTS

Modify, drop, or replace the output hierarchies and measures by clicking the edit icon.

MEASURES

- BusinessEntityID
- RecursionLevel

HIERARCHIES

- FirstName
- LastName
- ManagerFirstName
- ManagerLastName
- OrganizationNode

SCRIPT EDITOR

DATA PREVIEW

Warnings (0)

RecursionLevel	BusinessEntityID	OrganizationNode	ManagerFirstName	ManagerLastName	FirstName	LastName
0	127	/3/1/13/	Peter	Krebs	David	Hamilton
1	128	/3/1/13/1/	David	Hamilton	Paul	Komosinski
1	129	/3/1/13/2/	David	Hamilton	Gary	Yukish
1	130	/3/1/13/3/	David	Hamilton	Rob	Caron
1	131	/3/1/13/4/	David	Hamilton	Baris	Cetinok
1	132	/3/1/13/5/	David	Hamilton	Nicole	Holliday



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)
- [Passing Parameter Value From Dashboard to Stored Procedure](#)
- [Using a Bridge Parameter](#)

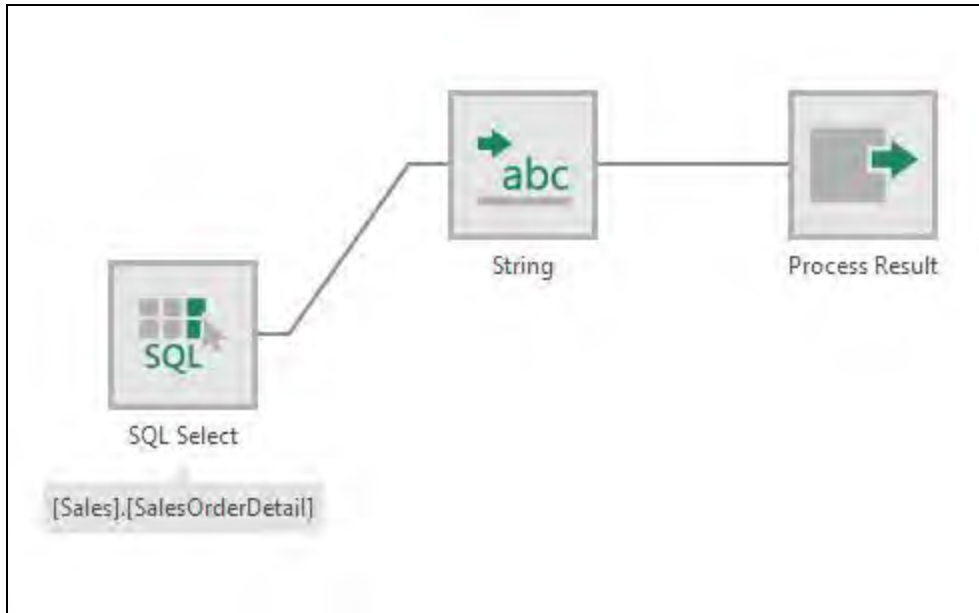
String

This applies to: *Managed Dashboards, Managed Reports*

The **String** transform manipulates string columns by applying string functions.

The following string functions are available, which are applied to the values in each row:

- **Trim** - Removes whitespace characters at the beginning and end.
- **TrimStart** - Removes whitespace characters at the beginning.
- **TrimEnd** - Removes whitespace characters at the end.
- **Remove** - Removes the specified number of characters starting at the specified position.
- **Replace** - Replaces all instances of the specified text with the specified replacement text.
- **Substring** - Keeps only the text at the specified position up to the specified length.
- **Left** - Keeps only the specified number of characters starting from the left/beginning.
- **Right** - Keeps only the specified number of characters starting from the right/end.
- **ToUpper** - Replaces all lowercase letters with uppercase letters.
- **ToLower** - Replaces all uppercase letters with lowercase letters.



Related video: [Data Cleansing with Data Cubes](#)

Input

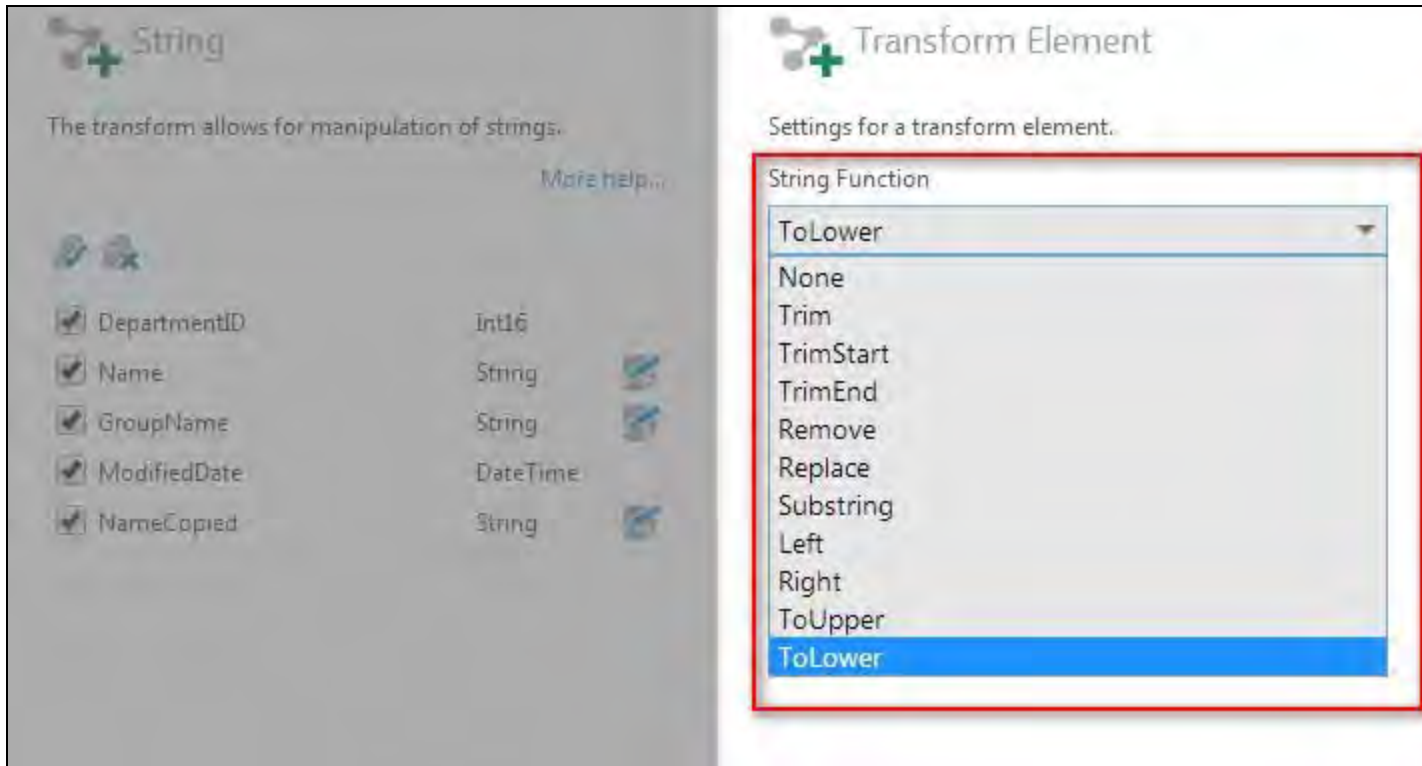
The String transform requires one input transform that has at least one column.

Configure

In the **String** transform configuration dialog, click the **Edit** icon next to a column with a **String** data type:



In the **Transform Element** dialog, choose a **String Function**.



Click the submit button at the bottom of the dialog when finished.

In the **String** transform configuration dialog, de-select any columns you don't want to be included in the output.



String

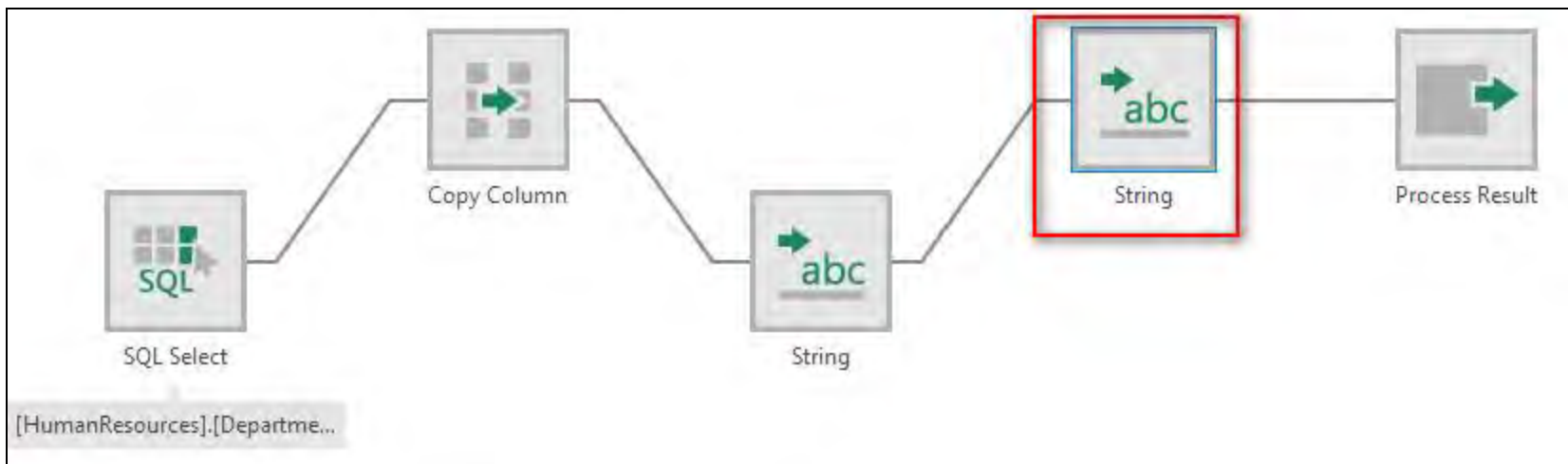
The transform allows for manipulation of strings.

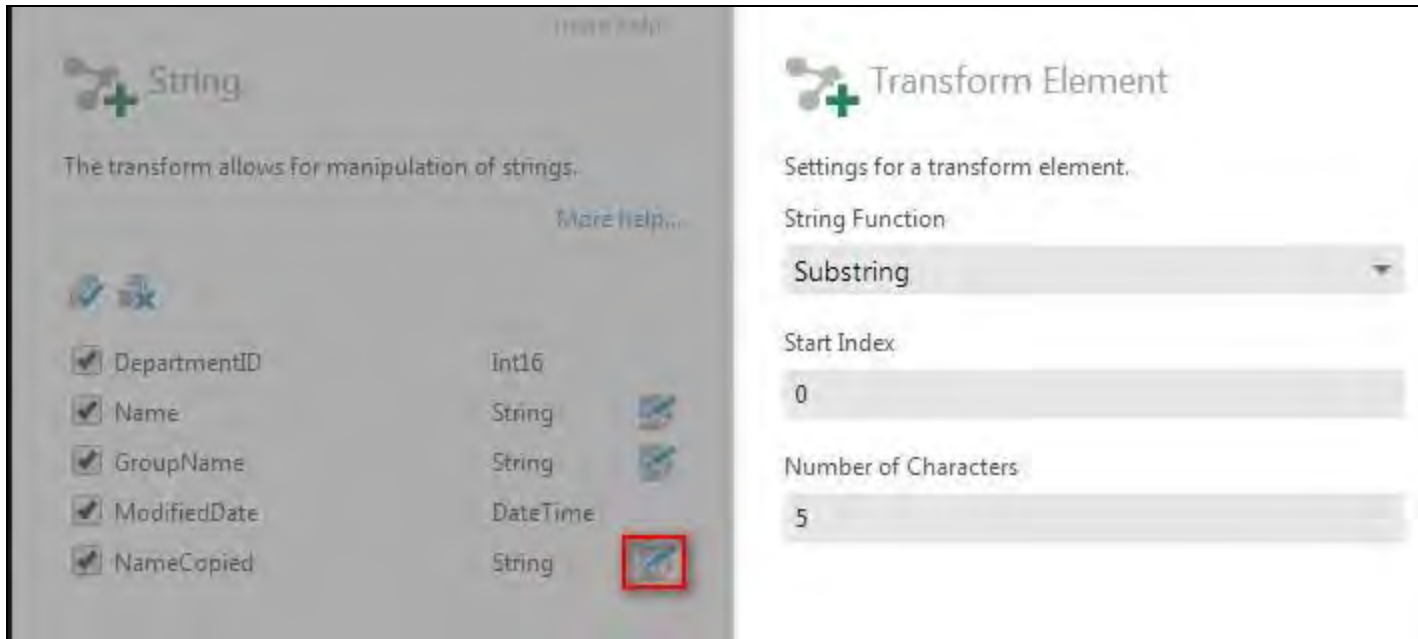
[More help...](#)




☒ DepartmentID	Int16	
☒ Name	String	
☒ GroupName	String	
☒ ModifiedDate	DateTime	
☒ NameCopied	String	

You can apply string functions to multiple columns/elements in the input with a single transform. To apply multiple string functions to the same column, you can add multiple String transforms.





Output

The figure below illustrates the output from the two string transform examples pictured above.

SCRIPT EDITOR DATA PREVIEW

DepartmentID	Name	GroupName	ModifiedDate	NameCopied
1	Engineering	Research and Development	06/01/2002 00:00:00	engin
2	Tool Design	Research and Development	06/01/2002 00:00:00	tool
3	Sales	Sales and Marketing	06/01/2002 00:00:00	sales
4	Marketing	Sales and Marketing	06/01/2002 00:00:00	marke
5	Purchasing	Inventory Management	06/01/2002 00:00:00	purch
6	Research and Development	Research and Development	06/01/2002 00:00:00	resea
7	Production	Manufacturing	06/01/2002 00:00:00	produ
8	Production Control	Manufacturing	06/01/2002 00:00:00	produ
9	Human Resources	Executive General and Administration	06/01/2002 00:00:00	human
10	Finance	Executive General and Administration	06/01/2002 00:00:00	finan

For more information, see:

- Video: [Introduction to Data Cubes](#)
- Video: [Data Cleansing with Data Cubes](#)
- [Using Transforms](#)



- Archive of documentation for Logi Symphony v24

- [Create a New Data Cube](#)
- [Data Cubes](#)



Tabular Select

This applies to: *Managed Dashboards, Managed Reports*

The **Tabular Select** transform is created when you drag a structure onto the data cube canvas from a data connector that supports tabular data (e.g., Excel, XML, SQL Server table-valued functions).

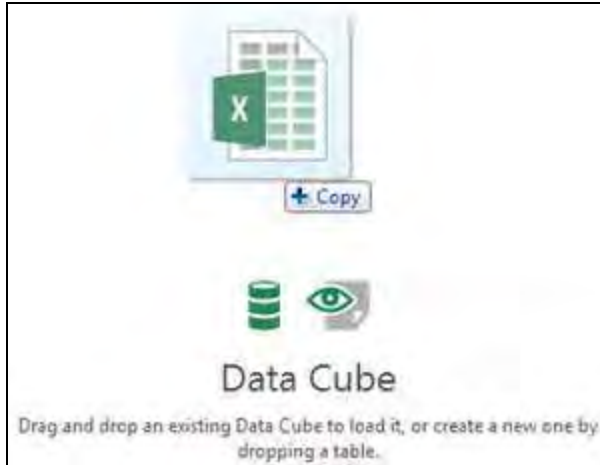
Input

The structure of the table dropped on the data cube canvas is used as input. The following example is used in this article:

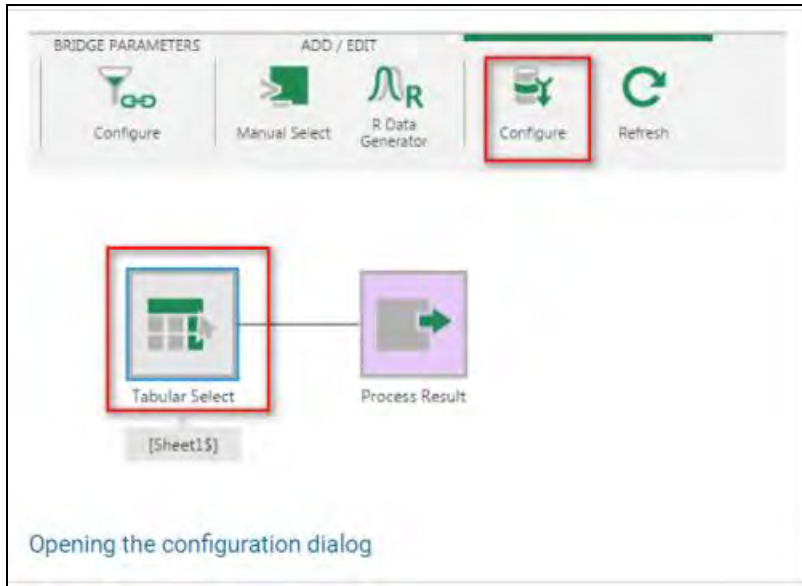
	A	B	C	D
1	Continent	Country	State	Sales
2	North America	USA	California	9999
3	North America	USA	Texas	2000
4	North America	Canada	Ontario	3000
5	North America	Canada	Quebec	4000
6	Asia	China	Beijing	5000
7	Asia	China	Shanghai	6000
8	Asia	Japan	Tokyo	7000
9	Asia	Japan	Narita	15000
10				

Add the Transform

Find an existing data connector in the **Explore** window and drag and drop one of its tables. You can also drag and drop a file to the canvas from Windows Explorer or Finder.



To configure the transform, select it and choose **Configure** in the toolbar or from the context menu.



Configure

The **Tabular Select** configuration dialog appears.



Tabular Select

[more help](#)

The transform responsible for retrieving data from table-valued functions, flat files or other tabular structures.

Name

[Sheet1\$]

Description

ID

54a49ae1-cb5c-46e0-8d21-bc7bf114151c

SETTINGS

Command Timeout

30

Search Input Elements

☒ Continent String

☒ Country String

☒ Date DateTime

☒ Sales Amount Double

Parameters 

Input elements 

Output elements 

Tabular Select transform configuration

Select which columns from the table should be included in the output of this transform.

The **Timeout** can be customized to change the length of time in seconds before terminating a long-running query.



Note: The Default Query Timeout application configuration setting determines the initial timeout setting.

Click **Parameters** to set up parameters. You can use these for filtering if your data connector is based on uploaded data or a database, or for changing settings. Your data source may also define its own parameters that you can set, for example an [OData connector's](#) query options, which will be indicated by a gear icon displayed on the transform when first added.

Many options are common to all select transforms. You can find more details about these in the [SQL Select](#) transform article.

Output

With the transform selected, open the **Data Preview** window to see the output.

Sales	Continent	Country	State	SalesDouble	Sales	Continent	Country	State	SalesDouble
9999	North America	USA	California	19998	9999	North America	USA	California	19998
2000	North America	USA	Texas	4000	2000	North America	USA	Texas	4000
3000	North America	Canada	Ontario	6000	3000	North America	Canada	Ontario	6000
4000	North America	Canada	Quebec	8000	4000	North America	Canada	Quebec	8000
5000	Asia	China	Beijing	10000	5000	Asia	China	Beijing	10000
6000	Asia	China	Shanghai	12000	6000	Asia	China	Shanghai	12000
7000	Asia	Japan	Tokyo	14000	7000	Asia	Japan	Tokyo	14000
15000	Asia	Japan	Narita	30000	15000	Asia	Japan	Narita	30000

For more information, see:



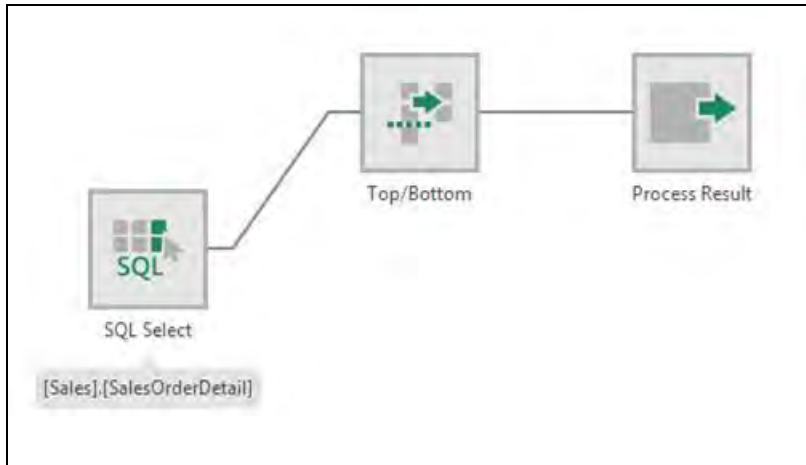
- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Top/Bottom

This applies to: *Managed Dashboards, Managed Reports*

The **Top/Bottom** transform filters the records to contain the top or bottom 'N' number of records as ranked by the columns you select.

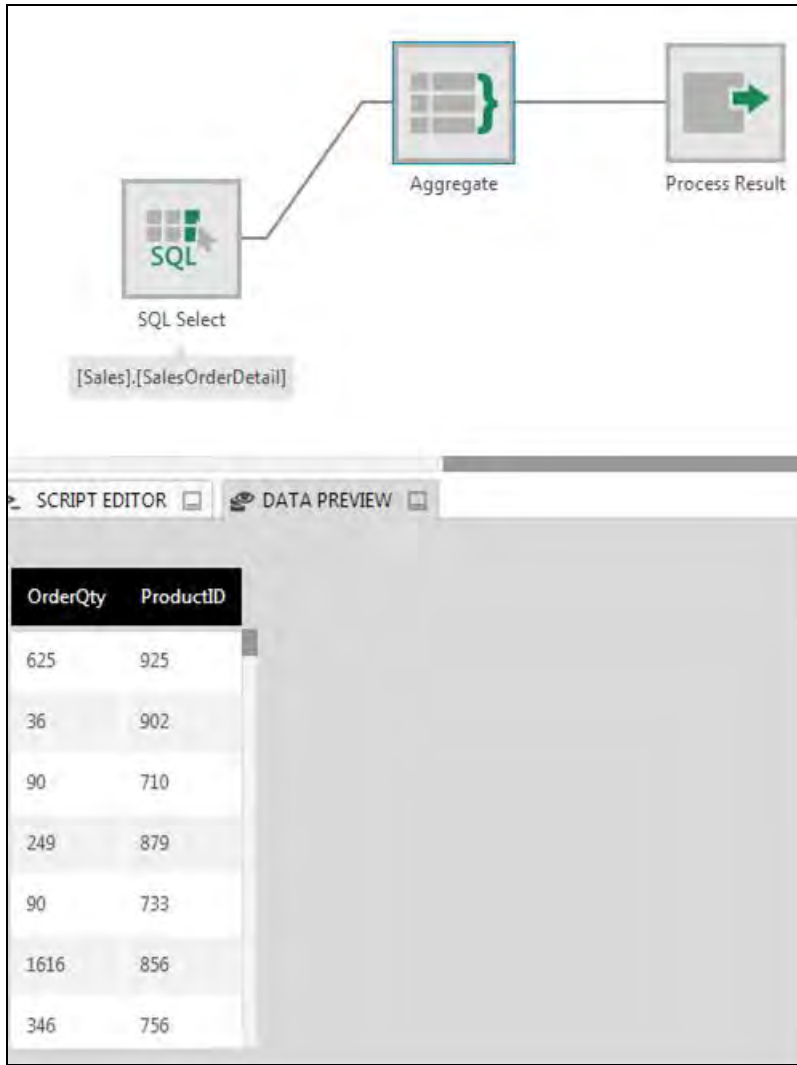


Note: The metric set can also display [top/bottom records](#).

Input

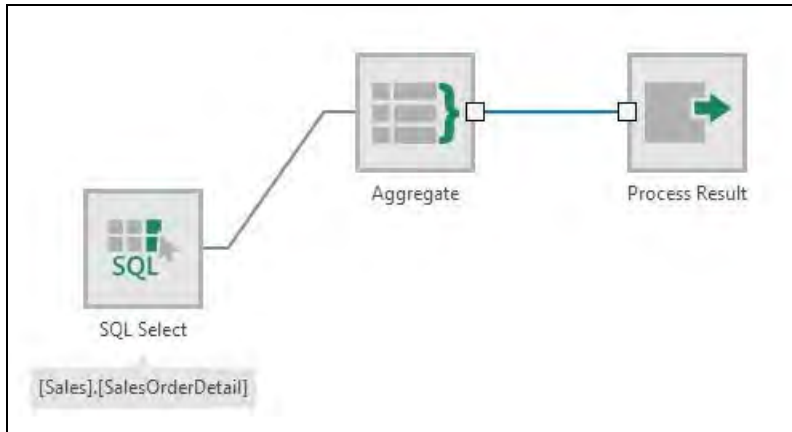
The Top/Bottom transform requires one input transform that has at least one column.

For our example, we'll use the output of the [Aggregate](#) transform, as seen below:

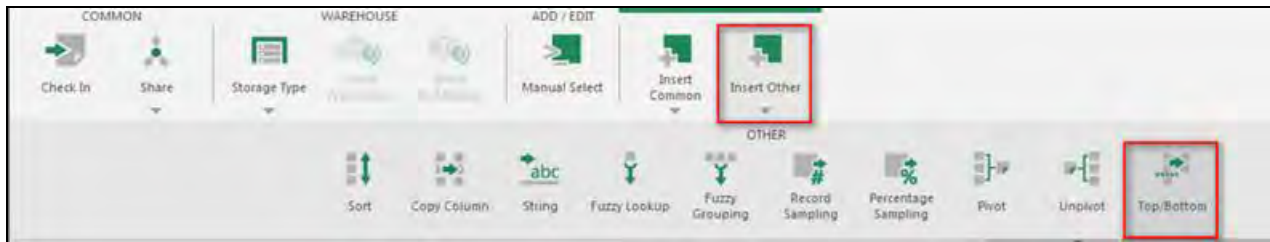


Add the Transform

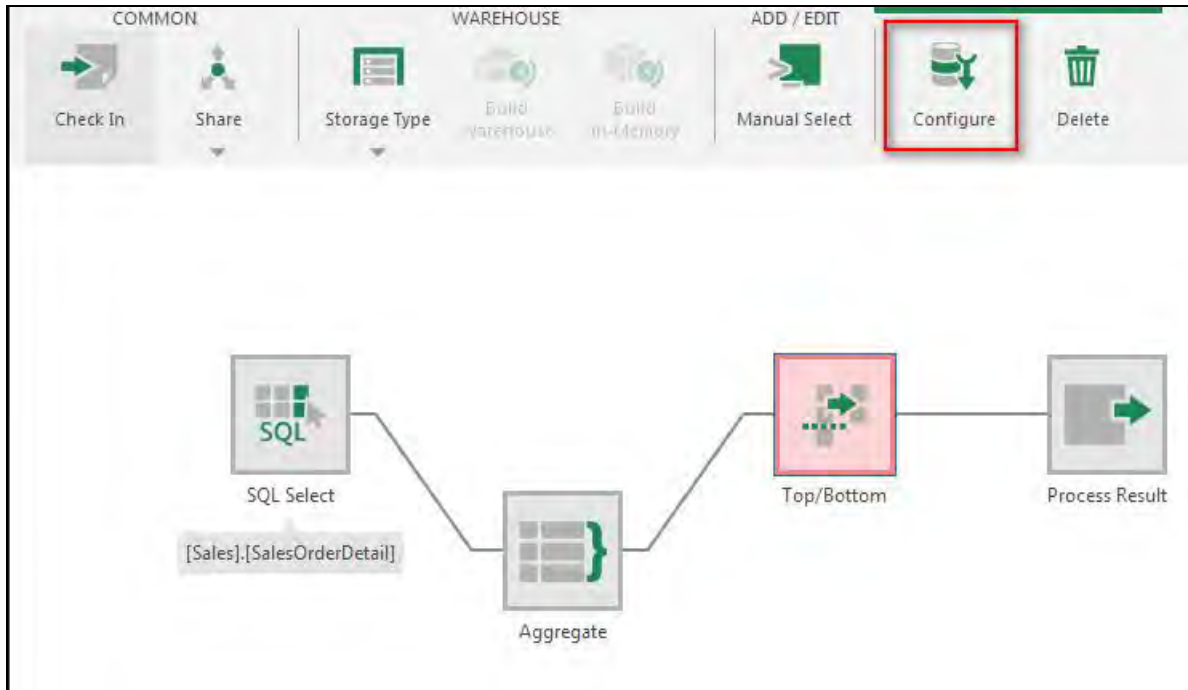
Select the connector link between two transforms.



In the toolbar, choose **Insert Other**, then **Top/Bottom**.



To edit/configure the transform, select it and choose **Configure** in the toolbar.



Configure

Steps to configure the Top/Bottom transform:


Top/Bottom

The transform allows various sorting rules to be applied to the input columns and returns the specified quantity of results to the output.

Name

Description

ID

Record Count

Starting Point

Click on the sort icon of each transform element to toggle its sorting method. Drag and drop within the list to set the priority.

Sorting Transform Input

☒ OrderQty

☒ ProductID

Sorting Elements

OrderQty

Use Record Grouping

Define parameters

Edit output elements

1. Enter a **Record Count**.
2. Choose a **Starting Point** to select either the records at the **Top** or **Bottom** of the result after sorting.
3. Uncheck any columns you don't want included in the output.



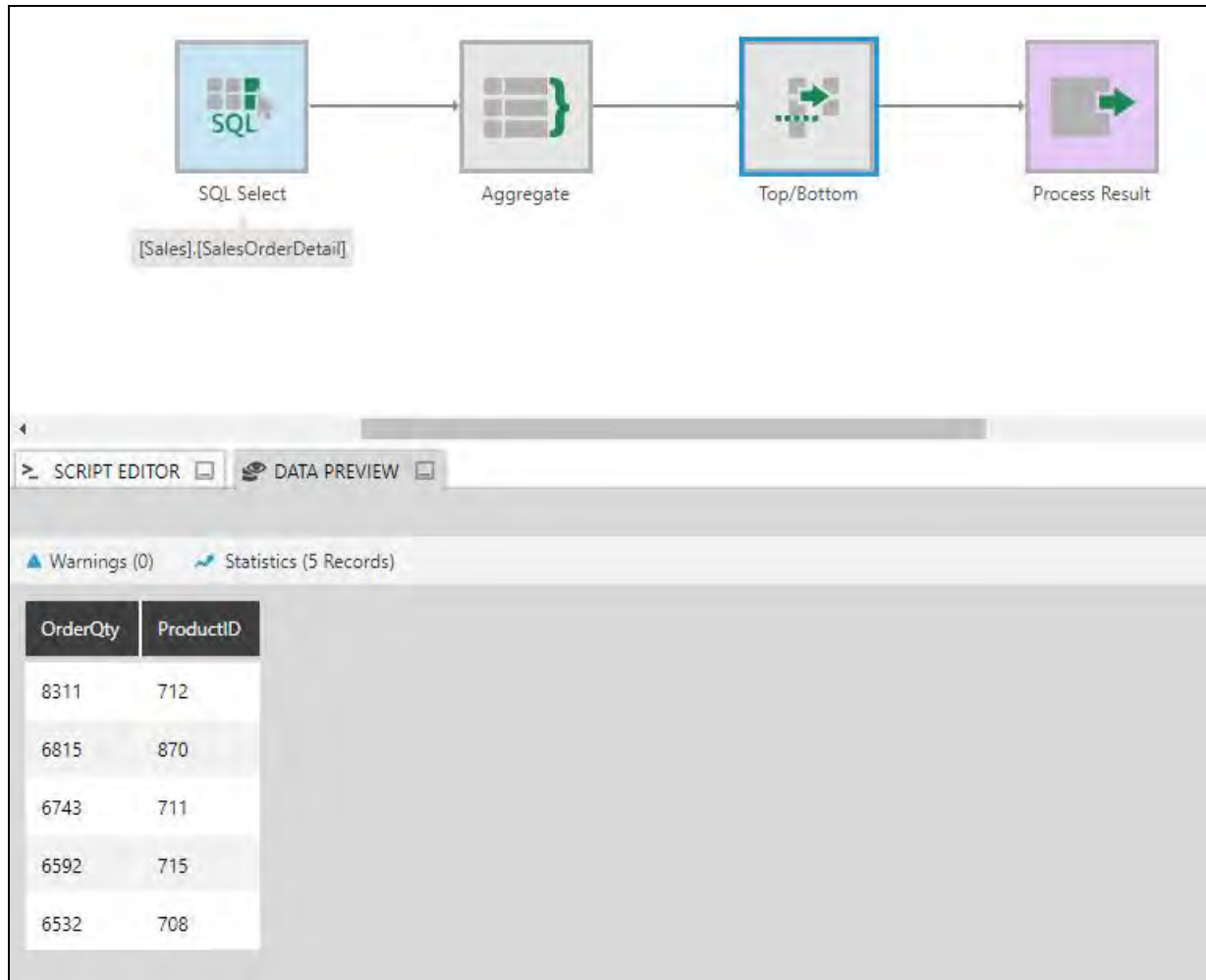
- Archive of documentation for Logi Symphony v24

4. Drag and drop the columns from the left that you want used to sort the data, and drop them under **Sorting Elements**. Data will be sorted first by the first element listed, then by the following elements.
5. Click the arrow icon for each sorting element to determine whether to sort by its values **Ascending** (up arrow) or **Descending** (down arrow).

Select **Use Record Grouping** if you want to return the number of records specified above multiple times for groups of records. A list of **Grouping Elements** will appear: select elements that should have a separate top/bottom list for each of their values.

Output

The figure below illustrates the output from the Top/Bottom transform for the **Top 5** records as sorted **Descending** by **OrderQty**:



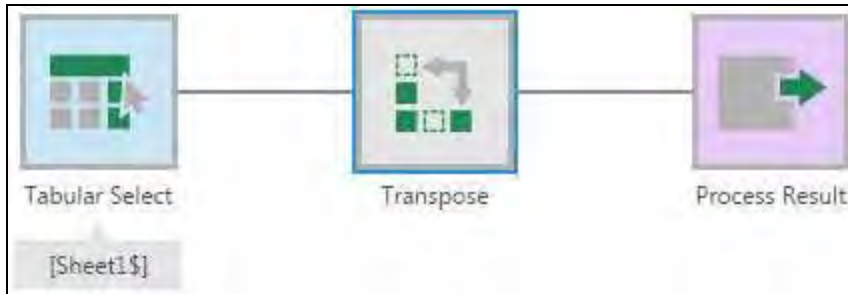
For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Transpose

This applies to: *Managed Dashboards, Managed Reports*

The Transpose transform lets you swap columns to rows and rows to columns.



Input

The **Transpose** transform requires one input transform that has at least one column.

For example, the input could be a Tabular Select transform that corresponds to the following Excel sheet:

salestarget.xlsx			
	A	B	C
1	Location	Sales	Target
2	Vancouver	42300	40000
3	Calgary	27500	30000
4	Toronto	65100	60000
5	Montreal	53600	50000
6	Ottawa	26400	30000
7			

Configure

In the Transpose configuration dialog, the **Elements** representing the columns to be transposed into rows are listed. This list is read-only.

Enter the **Number of Rows** that will be transposed into columns (i.e., the number of input rows).



- Archive of documentation for Logi Symphony v24

Set the **New Header Element** drop-down to the input column containing the values that will become the output column headings.



Transpose

[more help](#)

The transform handles swapping columns to rows and rows to columns.

Name

Description

Elements

Location

Sales

Target

Number Of Rows

New Header Element

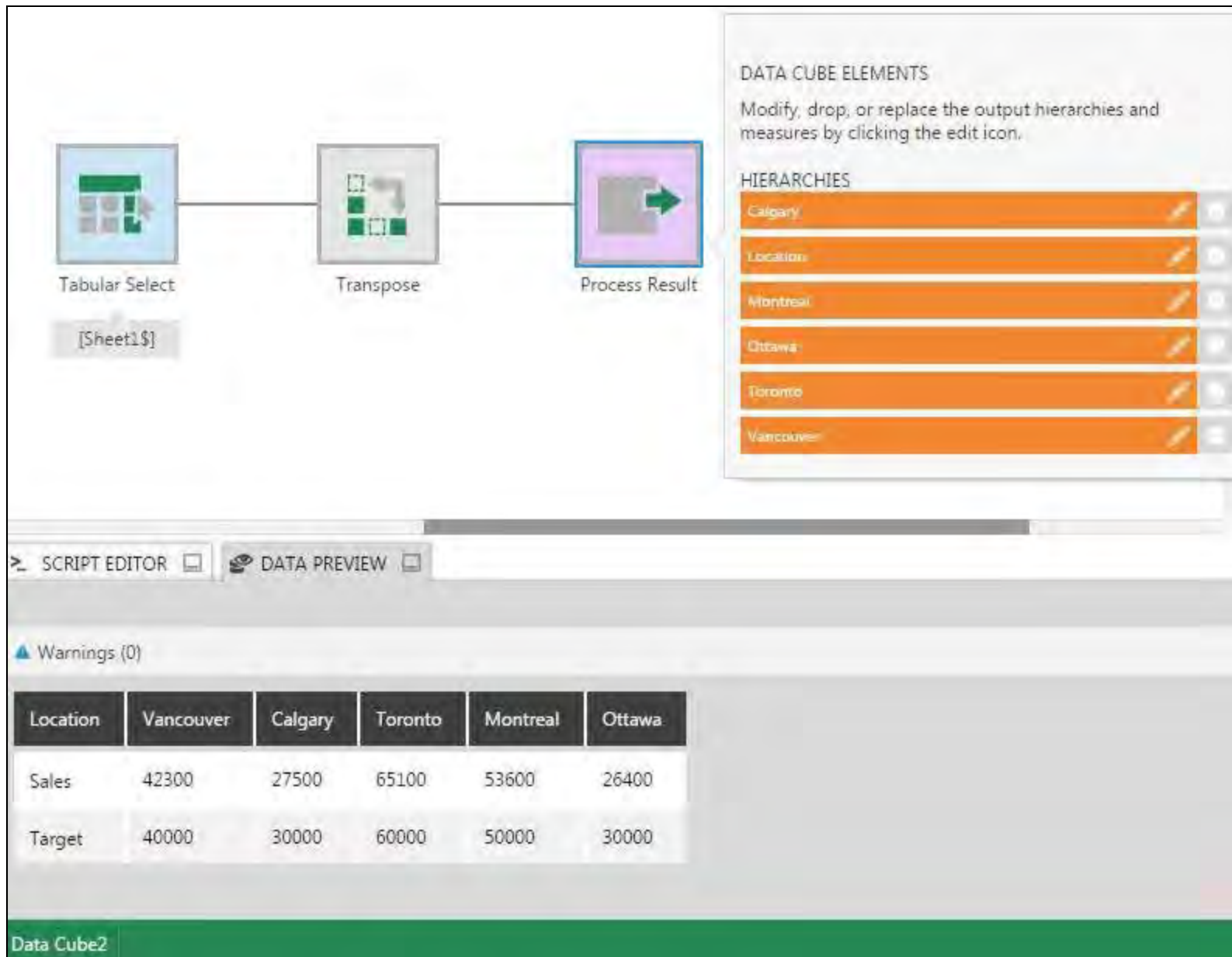






Output

The output of the Transpose transform is the swapping of columns to rows and rows to columns from the input data.



The screenshot displays the Logi Symphony v24 interface. At the top, the Logi Analytics logo is visible. Below it, a workflow diagram shows three steps: 'Tabular Select' (with a reference to '[Sheet1\$]'), 'Transpose', and 'Process Result'. To the right of the workflow, a panel titled 'DATA CUBE ELEMENTS' provides instructions: 'Modify, drop, or replace the output hierarchies and measures by clicking the edit icon.' Below this, a list of 'HIERARCHIES' is shown, including 'Calgary', 'Location', 'Montreal', 'Ottawa', 'Toronto', and 'Vancouver', each with an edit icon.

At the bottom of the interface, there are tabs for 'SCRIPT EDITOR' and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is active, showing a table of data. Above the table, a 'Warnings (0)' indicator is present. The table has columns for 'Location' and five cities: 'Vancouver', 'Calgary', 'Toronto', 'Montreal', and 'Ottawa'. The rows represent 'Sales' and 'Target' values for each city.

Location	Vancouver	Calgary	Toronto	Montreal	Ottawa
Sales	42300	27500	65100	53600	26400
Target	40000	30000	60000	50000	30000

The bottom of the interface features a green bar with the text 'Data Cube2'.

For more information, see:



- Archive of documentation for Logi Symphony v24

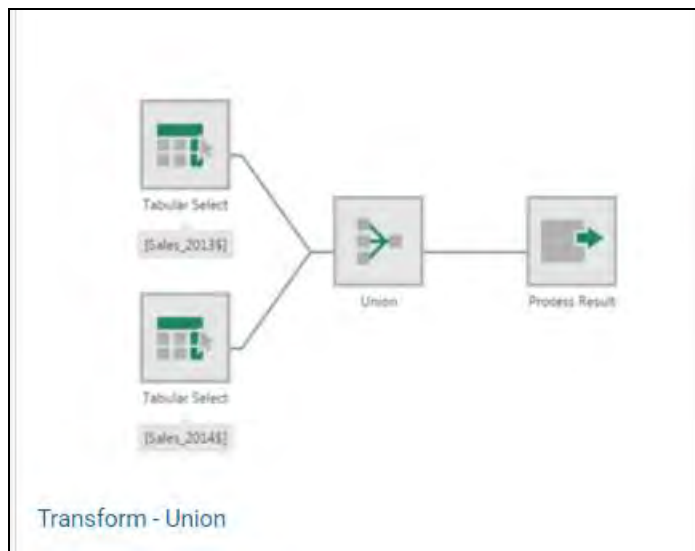
- [Using Transforms](#)
- [Data Cubes](#)
- [The Managed Dashboards Data Model](#)

Union

This applies to: *Managed Dashboards, Managed Reports*

The **Union** transform combines the rows from two tables that have matching columns.

If the two inputs come from the same data connector, and the data provider supports union statements, the union will be automatically included as part of the query sent to the database, unless the transform is preceded by another that must be executed by Symphony.



Input

The Union transform requires at least two inputs, and more than two are supported.

For example, the input could be data coming from two different Excel sheets, which have corresponding columns:

Input Table 1: Sales_2013

	A	B	C	D
1	Region	Sales		
2	East	2500		
3	West	4000		
4	South	4500		
5	North	5500		
		Sales_2013	Sales_2014	

Input table 2: Sales_2014

	A	B	C	D
1	Region	Sales		
2	East	3000		
3	West	4000		
4	South	5000		
5	North	6000		
		Sales_2013	Sales_2014	

Configure

In the **Union** configuration dialog, there are separate settings for each input.

For the first input, select which elements should be included in this transform. You can exclude elements that don't have matching columns in the other inputs.



Union

The Union transform, responsible for merging input from numerous sources into one output.

Name

Description

Inputs

Include Duplicate Rows

☒

Choose elements to be used in this transform.

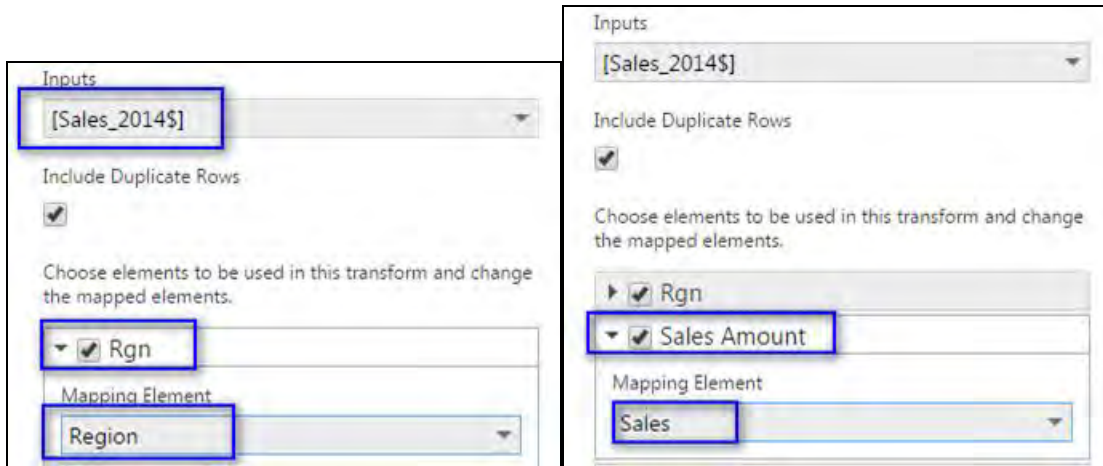
☒  

☒ Region
 ☒ Sales

Define parameters

Rename output elements

Change the **Inputs** drop-down to select each other input, and click to expand each column's **Mapping Element**. Ensure each column is mapped correctly to the matching column in the other input. This is especially important if the column names did not match.



Inputs

[Sales_2014\$]

Include Duplicate Rows ☒

Choose elements to be used in this transform and change the mapped elements.

☒ Rgn

Mapping Element

Region

Inputs

[Sales_2013\$]

Include Duplicate Rows ☒

Choose elements to be used in this transform and change the mapped elements.

☒ Rgn

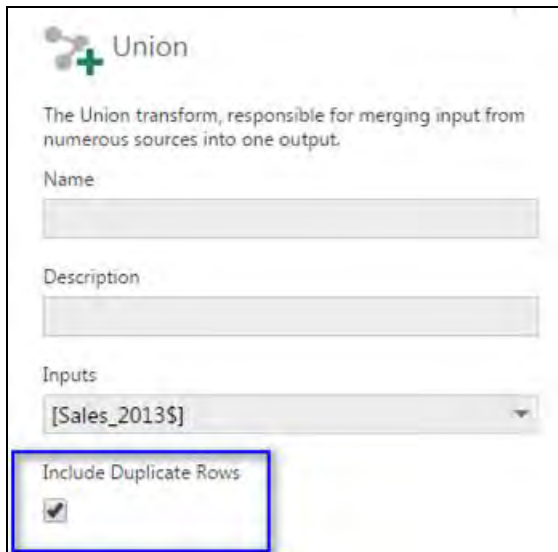
☒ Sales Amount

Mapping Element

Sales

Extra elements that don't have matching columns in the other inputs can be un-checked to exclude them.

There is an **Include Duplicate Rows** option, which applies to all inputs.



 Union

The Union transform, responsible for merging input from numerous sources into one output.

Name

Description

Inputs

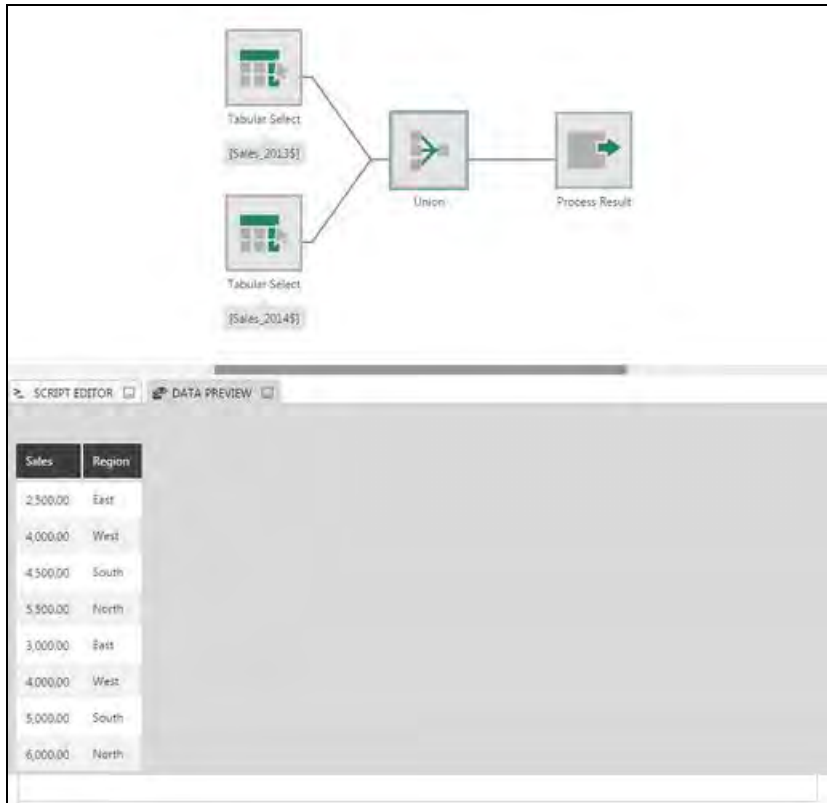
[Sales_2013\$]

Include Duplicate Rows ☒

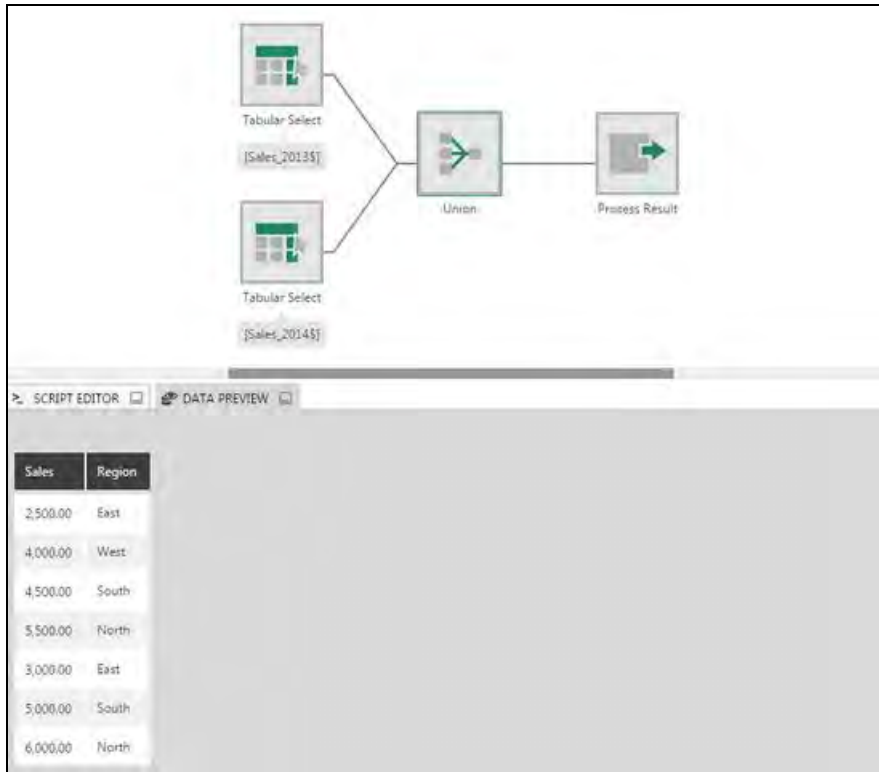
Output

The output combines the rows from all of the inputs.

The figure below illustrates the output in our example:



The figure below illustrates the output when the duplicate rows are excluded:



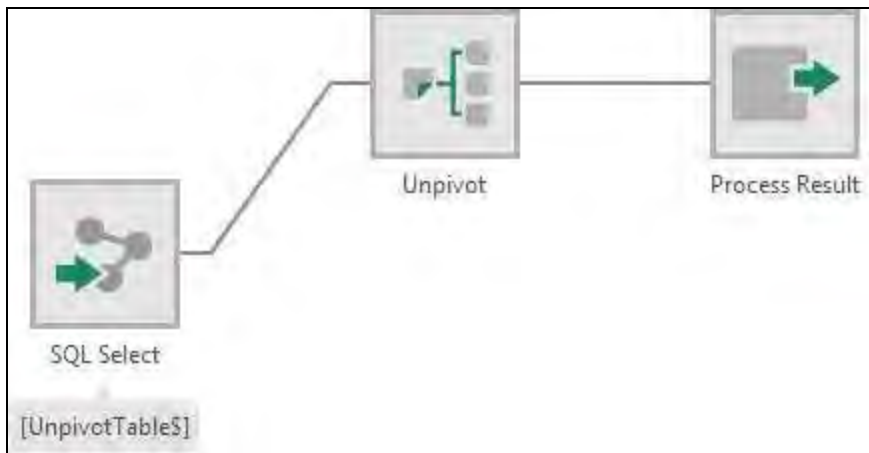
For more information, see:

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Unpivot

This applies to: *Managed Dashboards, Managed Reports*

The **Unpivot** transform combines multiple columns into a single column containing new rows ([opposite of Pivot](#)).



Input

The Unpivot transform requires one input transform that has at least three columns.

For example, the input could be a SQL Select transform that corresponds to the following Excel sheet

	A	B	C	D	E	F
1	City	Monday	Tuesday	Wednesday	Thursday	Friday
2	Dallas	52000	48000	36000	25000	34000
3	Detroit	28000	42000	32000	26000	40500
4	New York	65000	24000	28000	32000	40000
5						

Configure

In the **Unpivot** configuration dialog, you need to configure the following settings:



Unpivot

more help

The transform expands a result into a more normalized version by expanding values from multiple columns in a single record into multiple records with the same values in a single column.

Name

DayOfWeek

Description

Unpivot separate day columns into one

ID

4c466018-0fc5-4c26-a6aa-4a1c65d9552d

Unpivot Column Name

DayOfWeek

Unpivot Column Maximum Length

200

Value Column Name

Sales

Unpivot Columns




☐ City
 ☒ Monday
 ☒ Tuesday
 ☒ Wednesday
 ☒ Thursday
 ☒ Friday

Define parameters

Edit output elements



Unpivot transform configuration dialog

You can optionally provide a **Name** and **Description** for the transform for your reference.

Enter a name in **Unpivot Column Name** for the new column that will contain the names of all the unpivoted columns (e.g., the days of the week in our example).

Set the **Unpivot Column Maximum Length** as required. The default value is 200.

Set **Value Column Name** to name the new column that will contain all of the unpivoted column values.

- Select the **Unpivot columns**, which will be combined into a single column.

Output

The figure below illustrates the output from the Unpivot transform in our example.

Sales	City	DayOfWeek
52000	Dallas	Monday
48000	Dallas	Tuesday
36000	Dallas	Wednesday
25000	Dallas	Thursday
34000	Dallas	Friday
28000	Detroit	Monday
42000	Detroit	Tuesday
32000	Detroit	Wednesday
26000	Detroit	Thursday
40500	Detroit	Friday

For more information, see:



- Archive of documentation for Logi Symphony v24

- Transform: [Pivot](#)
- [Using Transforms](#)
- [Data Cube Data Storage Options](#)
- [The Managed Dashboards Data Model](#)

Update a Select Transform When The Data Source Changes

This applies to: *Managed Dashboards, Managed Reports*

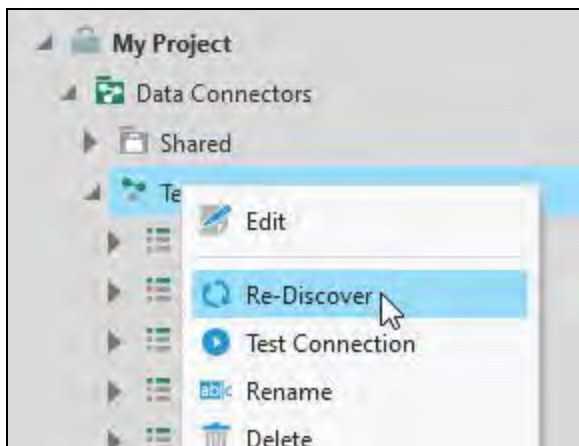
If the structure of the data used as an input to a data cube changes, new elements (columns) that were added may not initially be selected for input, and old elements that were removed will initially remain.

You may be shown an error when trying to preview a transform from your data cube if it still contains a reference to an element that was removed from your data source, such as **Invalid column name**. If you need this missing element, you may want to look into why it was removed and try to re-add it.

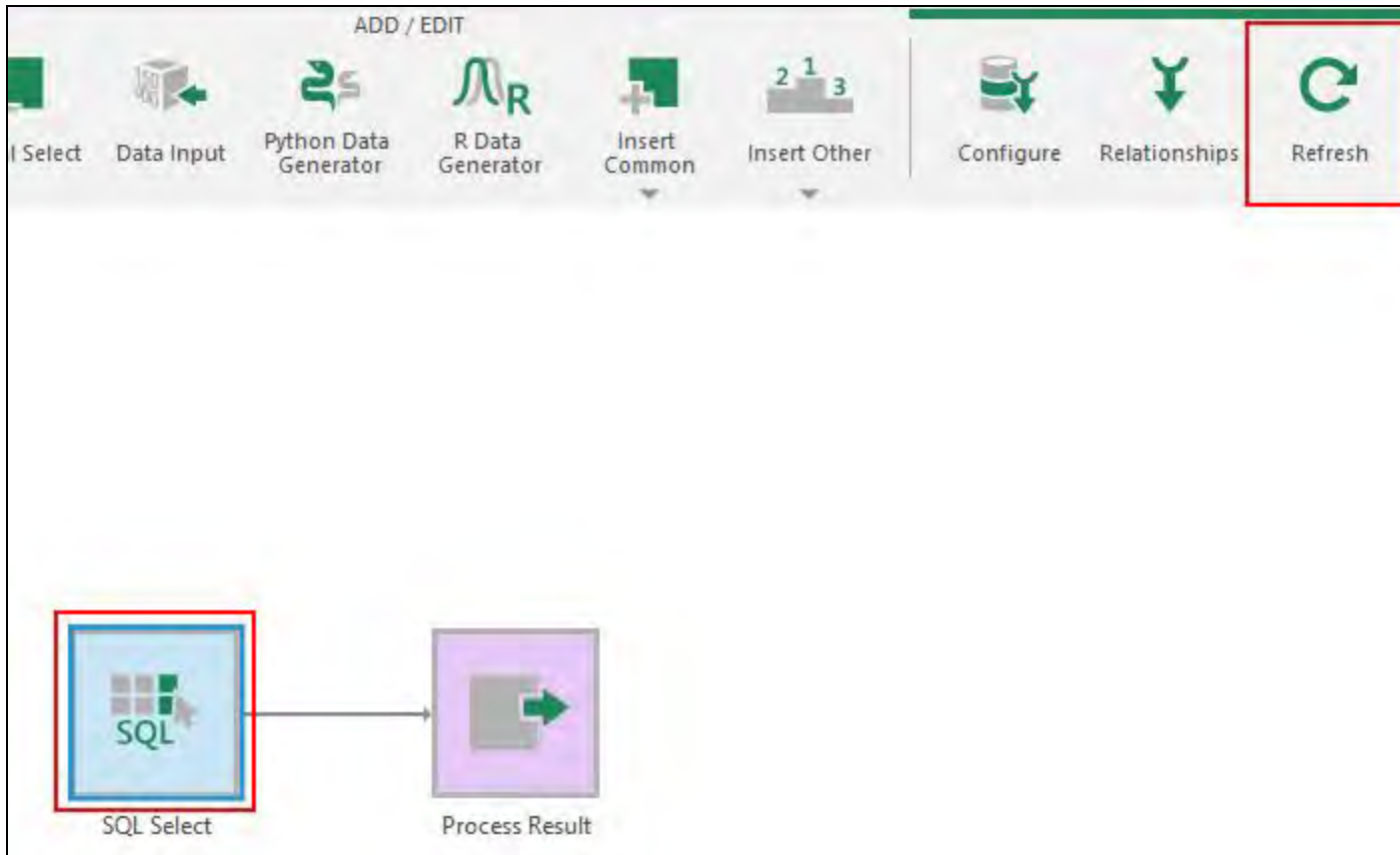
Refresh the Select Transform

If you want your select transform to update to reflect changes in the data source, you can refresh the select transform.

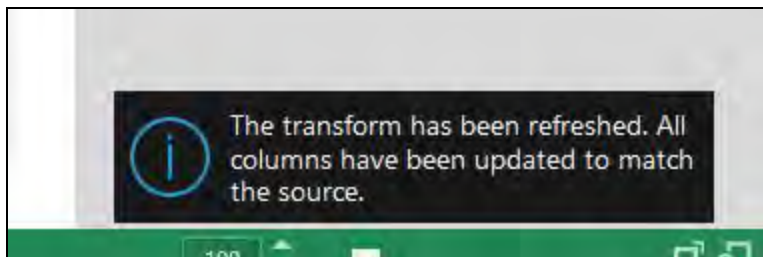
If the data source is an external data source such as a database, changes to its structure first need to be re-discovered. In the **Explore** window, find and right-click the data connector under **Data Connectors**, and choose **Re-Discover**.



Click to select your select transform and choose **Refresh** in the toolbar or from the context menu.



A popup notification will appear confirming that the select transform has been updated to match the data source.



Next, choose **Configure** in the toolbar or context menu. You can select any new elements as inputs to include in your process flow, and you can confirm that old elements have been removed.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Transforms](#)
- [Create a New Data Cube](#)
- [Data Cubes](#)

Using the Filter Transform With Tokens

This applies to: *Managed Dashboards, Managed Reports*

This article shows examples of filtering data using the [Filter](#) transform with [tokens](#) as filter values, which can include attributes or [custom attributes](#) describing the current user for user-dependent filtering.

Note: You can also add parameters to a [SQL Select](#) or other select [transforms](#) and set them using tokens in a similar way.

Current Display Name token

This walkthrough shows you how to filter data per user by using the Filter transform and the Current Display Name [attribute token](#).

Connect to Data

dbo.Employees X				
	EmployeeID	EmployeeName	EmployeePict...	Salary
	100	June Chou	5b84ef92-5052-...	60000.00
	101	Tom Grant	3bd77669-6374-...	65000.00
	102	Sam Preston	0f079222-a7ca-...	70000.00
	103	Alex Simpson	15fe8301-0636-...	75000.00
▶▶	NULL	NULL	NULL	NULL

Create a new data connector to the following SQL Server database table which contains employee and salary information. The goal will be to create a dashboard that lets employees see their own salary only.

Set Up User Accounts

Set up user accounts in Symphony as shown in the figure below. Note that the account display names match the **EmployeeName** column from the SQL table.





Accounts

 Locked Account
 Reserved Seat
 Floating Seat
 Disabled Account

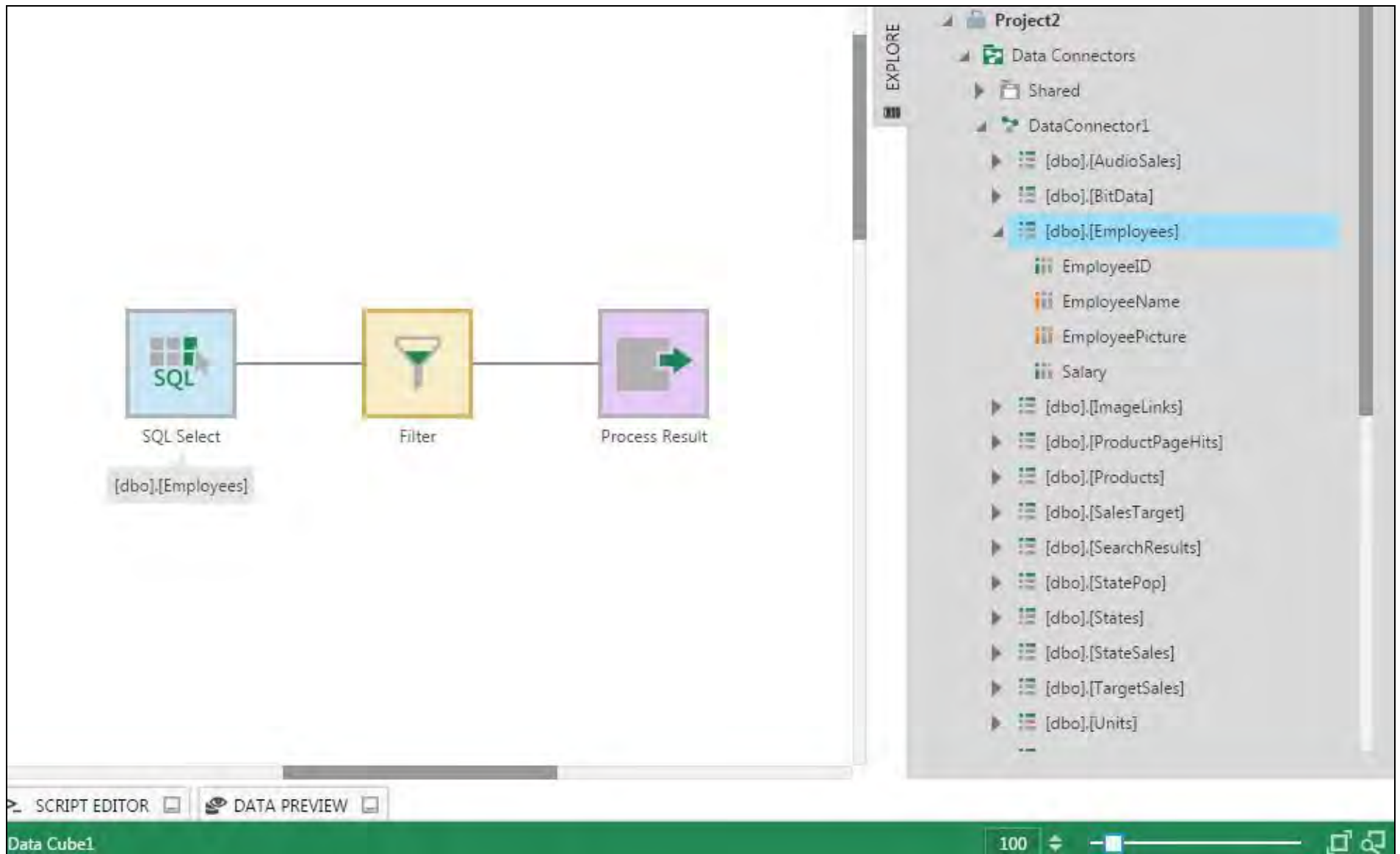
Name	Display name	Type	Se Last L...	Attributes	Edit / Delete
Admin	System Administrator	Local user	Di Wed...		
alex	Alex Simpson	Local user	St Wed...		 
junc	June Chou	Local user	St Wed...		 
samp	Sam Preston	Local user	St Wed...		 
tomg	Tom Grant	Local user	St Wed...		 

Create a Data Cube

Go to the **Explore** window and expand the data connector. Locate the **Employees** table.

Create a new data cube and drag the **Employees** table to the canvas.

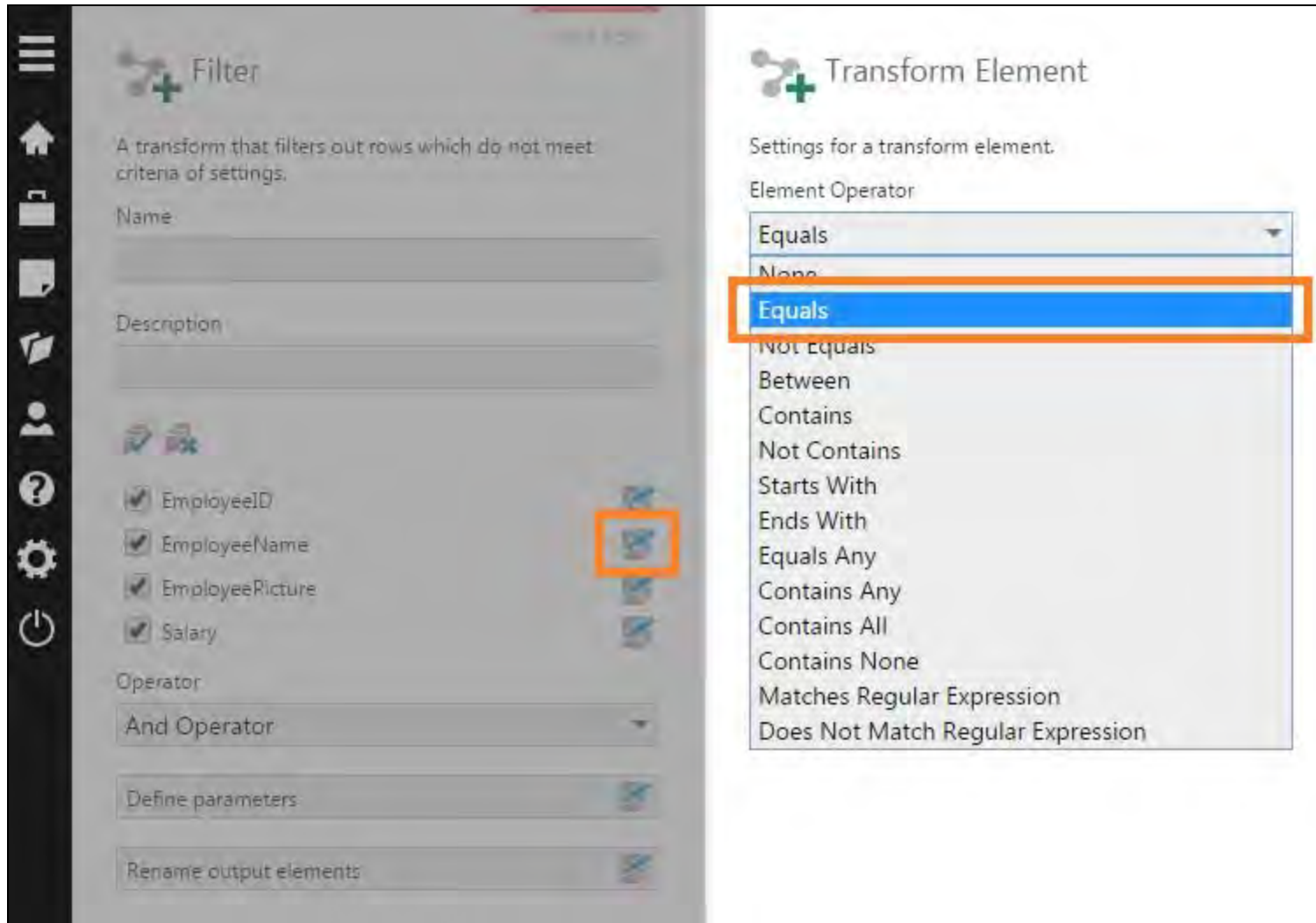
Select the connection link and insert a **Filter** transform.



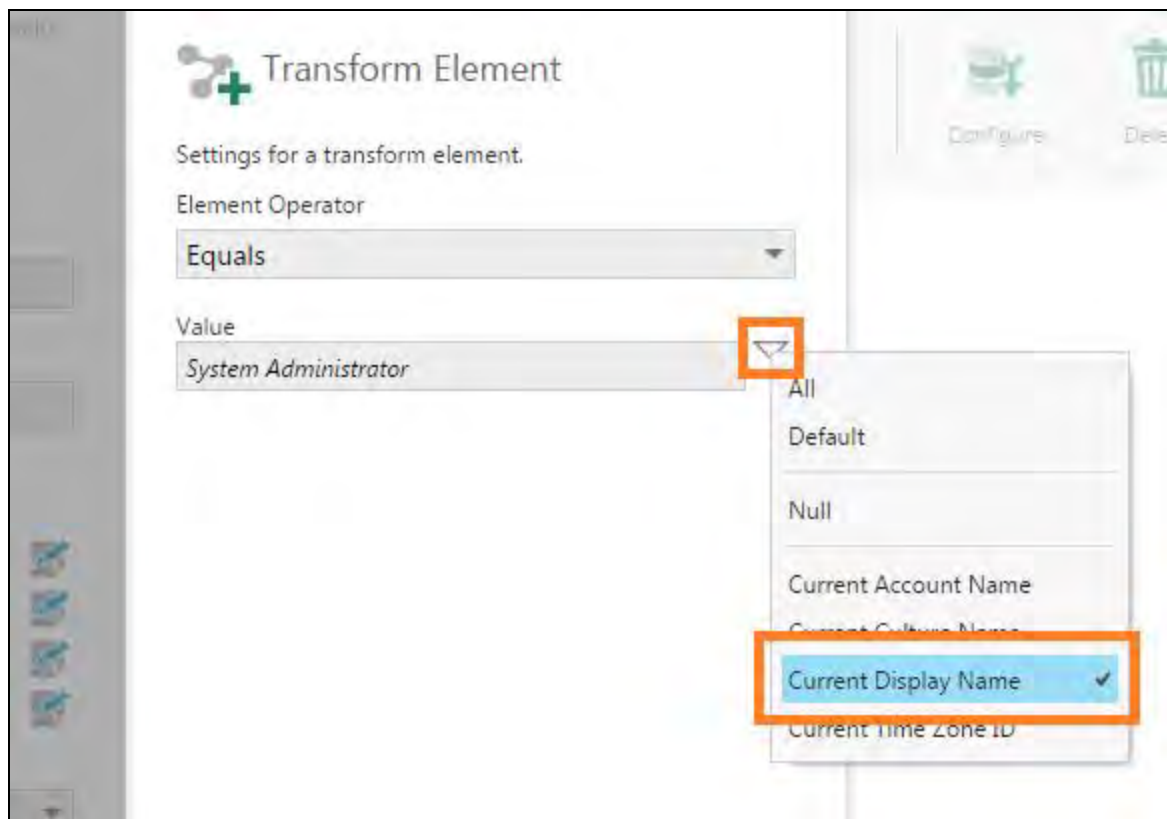
Right-click on the filter transform and select **Configure**.

In the **Filter** dialog, click the **Edit** button beside the **EmployeeName** column.

In the **Transform Element** dialog, set the **Element Operator** to **Equals**.



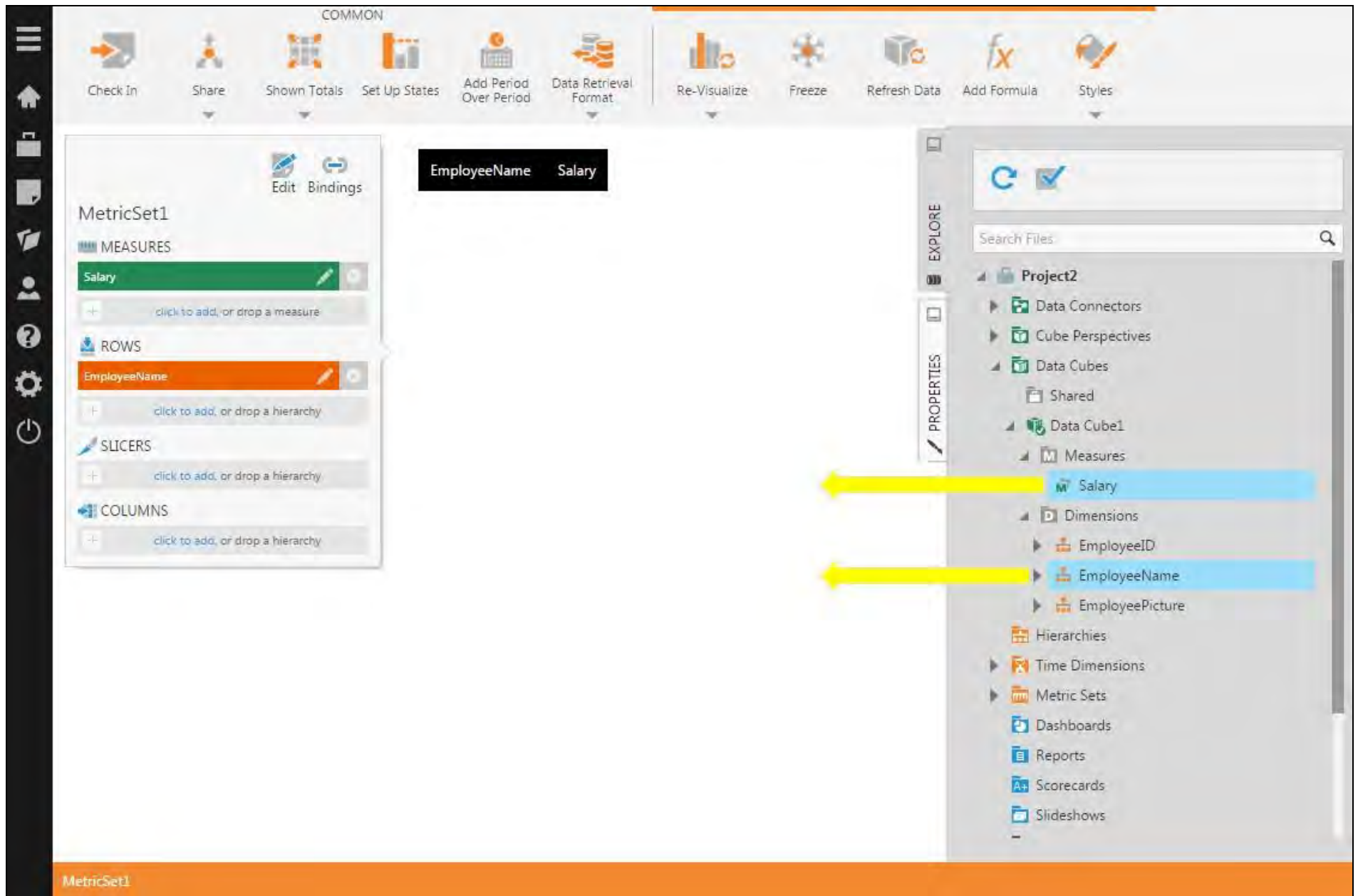
Instead of hardcoding a filter value, open the **Value** token menu and select **Current Display Name**. This token value resolves to the **Display Name** of the current Symphony user account.



The data cube is now complete and you can check it in.

Create a Metric Set

Next, create a new metric set by dragging the **Salary** measure and **EmployeeName** hierarchy from the data cube to the canvas. The resulting table visualization won't display any data though unless you happen to be one of the employee users.



The screenshot displays the Logi Symphony v24 interface. The top toolbar contains various actions: Check In, Share, Shown Totals, Set Up States, Add Period Over Period, Data Retrieval Format, Re-Visualize, Freeze, Refresh Data, Add Formula, and Styles. The left sidebar shows the 'MetricSet1' configuration panel with sections for MEASURES (Salary), ROWS (EmployeeName), SLICERS, and COLUMNS. The right sidebar shows the 'EXPLORE' pane with a tree view of the data model. Two yellow arrows point from the 'Salary' and 'EmployeeName' items in the 'EXPLORE' pane to the 'MetricSet1' panel. The bottom status bar shows 'MetricSet1'.

Check in the metric set.

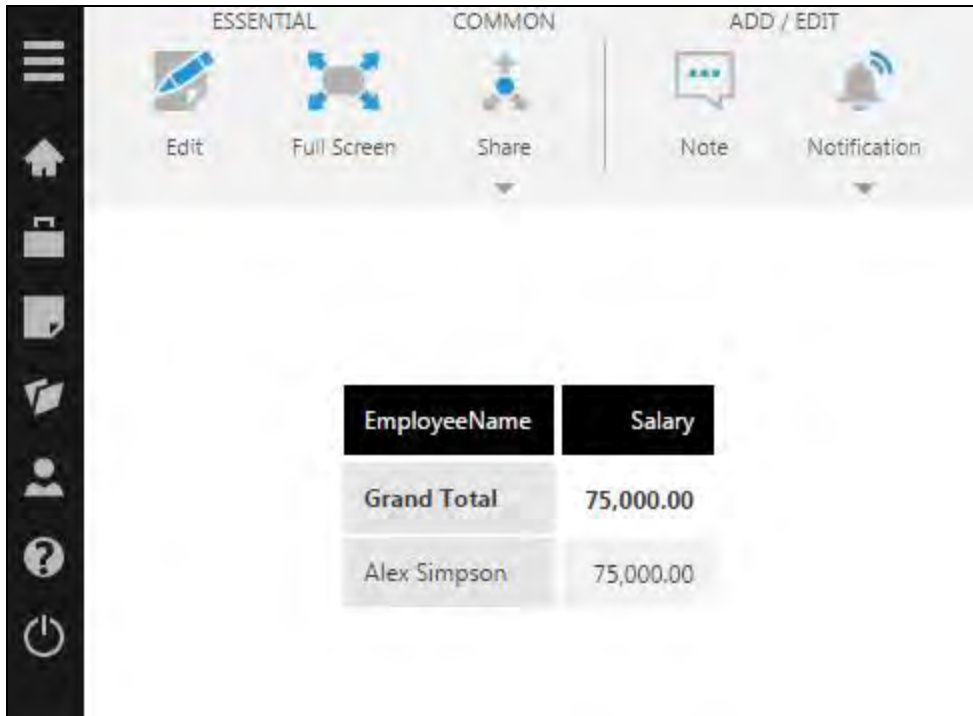
Create a Dashboard

Create a new dashboard by dragging the metric set to the canvas; it will appear as a table visualization without any rows.

Check in the dashboard.

View the Dashboard

Log out of Symphony and log on as one of the employee users. Open the dashboard for viewing and verify the table visualization shows the salary for this user only.



Filtering Out Null Values

There are a couple of ways to filter out Null values from your data using the Filter transform and the Null token.

The first method is to open the **Filter** configuration dialog, and edit the column containing possible Null values. In the **Transform Element** dialog, set the Element Operator to **Does Not Equal**. Then open the **Value** token menu and set it to **Null**.



Filter

A transform that filters out rows which do not meet criteria of settings.

Name

Description

☒ EmployeeID
☒ EmployeeName
☒ EmployeePicture
☒ Salary

Operator: And Operator

Define parameters 

Rename output elements 

Transform Element

Settings for a transform element.

Element Operator

Does Not Equal

Value: Null

All

Default

Null

Current Account Name

Current Culture Name

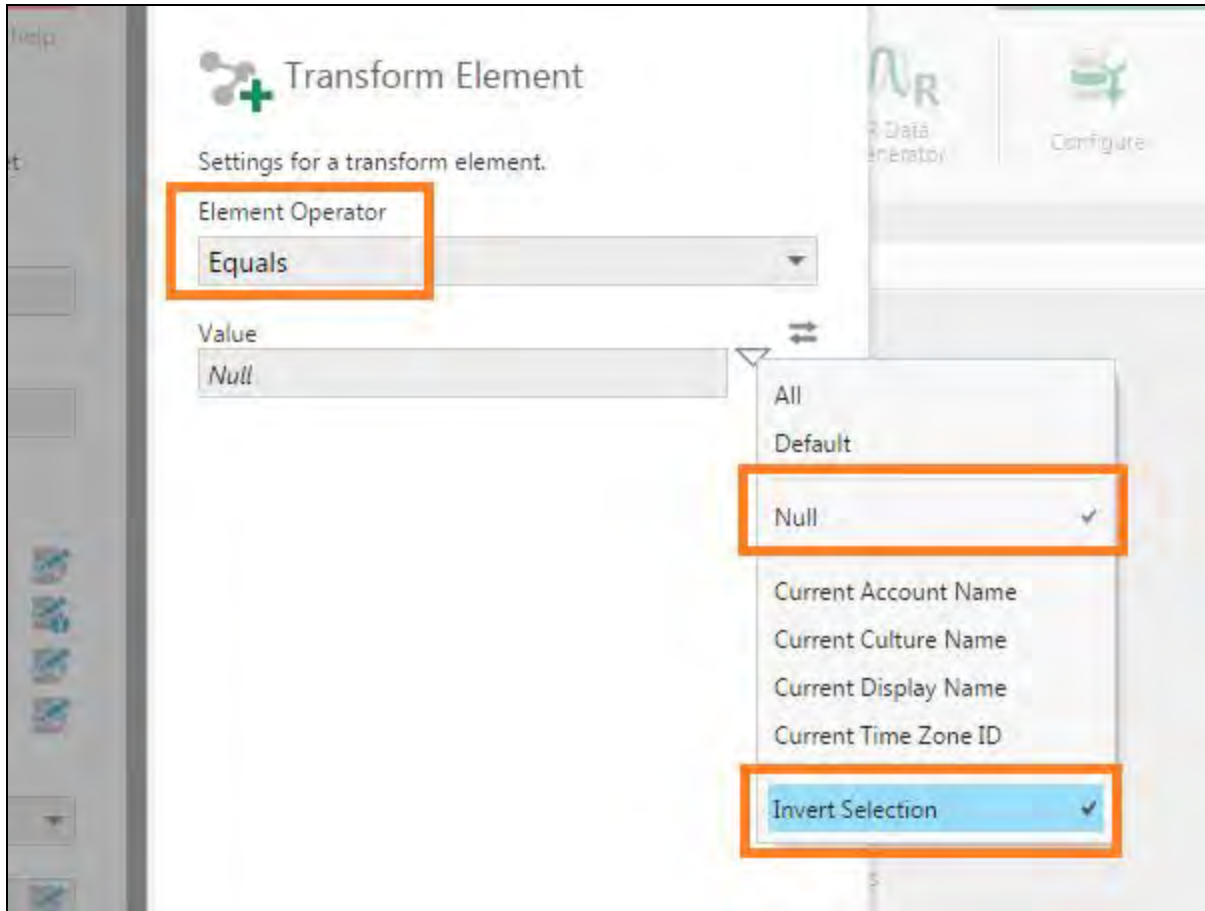
Current Display Name

Current Time Zone ID

Invert Selection

X ✓

The second method is to set the **Element Operator** to **Equals**. Then open the **Value** token menu and set it to **Null** and also select the **Invert Selection** option.



For more information, see:

- Transform: [Filter](#)
- Transform: [SQL Select](#)
- [Managing Tokens](#)
- [Using custom attributes to Filter data by user](#)



- Archive of documentation for Logi Symphony v24

- [Creating Custom Tokens](#)
- [Data Cubes](#)
- [Create a New Data Cube](#)

Add a Relationship Between Tables

This applies to: *Managed Dashboards, Managed Reports*

Relationships allow you to work with data from multiple data sources as if they had already been prepared into a single data cube for you. For example, with relationships defined, data from an Excel file and a SQL table can be selected together and automatically joined, even if the matching columns have different names.

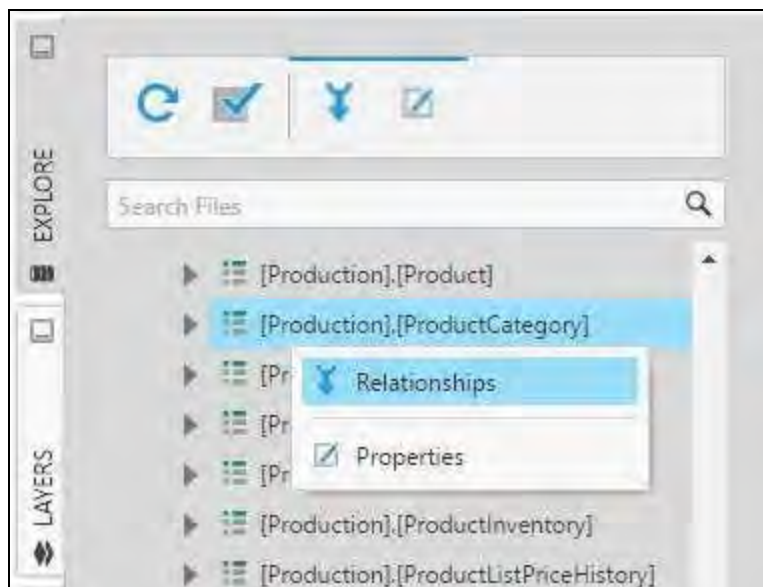
You can use the *Relationships* dialog to review existing relationships and add new ones. These are used when selecting data in a metric set, or when creating a multi-level hierarchy.



Note: If there is no relationship defined, the relationship will be guessed for you when you use unrelated data in a metric set. You can then modify how the data was joined as shown in [Automatic Joins and Hierarchies](#).

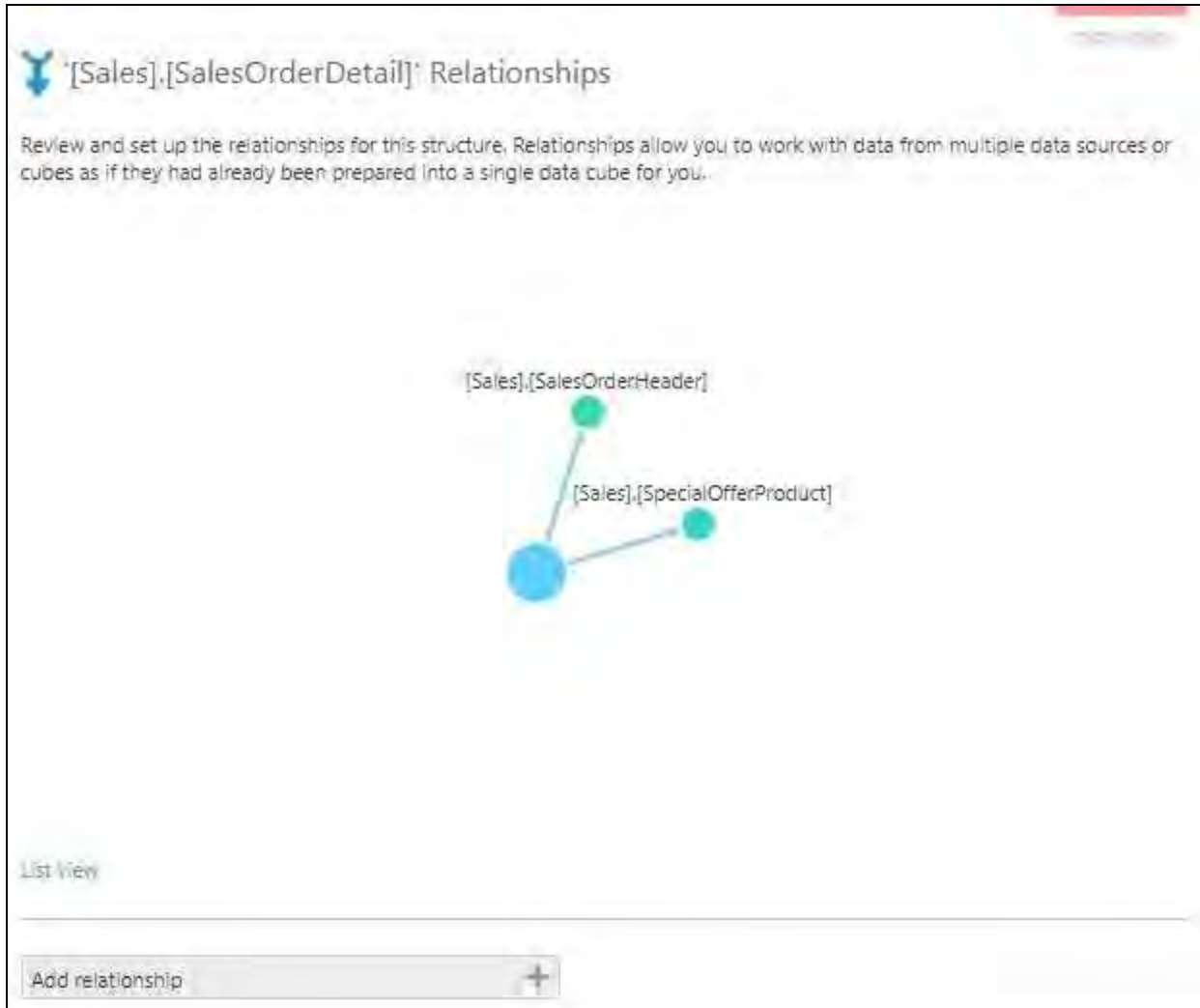
Review Existing Relationships

Go to the **Explore** window, right-click on a table and choose **Relationships**.



Note: For data cubes, you cannot define a relationship between its data and another data structure before combining its data with other data in a metric set, but you can [edit the automatic join](#) afterward.

In the *Relationships* dialog, you will see a diagram displaying the existing relationships between that table and other tables/data structures.



Hover over or long-tap a node (circle) to view details about how it's related in a tooltip.

'[Sales].[SalesOrderDetail]' Relationships

Review and set up the relationships for this structure. Relationships allow you to work with data from multiple data sources or cubes as if they had already been prepared into a single data cube for you.



For each table node in the diagram, you can right-click (or long-tap) and choose **Expand** to add all of the additional tables related to that table to the diagram. This allows you to explore and discover other relevant data that can be made available by the intermediate tables.

'[Sales].[SalesOrderDetail]' Relationships

Review and set up the relationships for this structure. Relationships allow you to work with data from multiple data sources or cubes as if they had already been prepared into a single data cube for you.



If needed, to view the relationships as a list, click **List view**. The *Category* column displaying in list view indicates how each relationship was created:

- *Native* relationships are defined in the database (e.g., foreign keys).
- *System-defined* relationships were created automatically.
- *User-defined* relationships were created by a user.

Return to **Diagram view**, then right-click a node and click **Edit** to see its details.



Note: Native relationships can only be modified in the source database.



Create a Relationship

By creating a relationship ahead of time, you can define how two tables are related rather than allow this to be chosen for you when you or others drag data from those tables onto one metric set.

Start in the *Relationships* dialog opened for one of the tables as described above, and click **Add relationship**. The *Set Up a Relationship* dialog is displayed.

Enter a **Name** for the relationship.

'[Sales].[SalesOrderDetail]' Relationships

Review and set up the relationships for this structure. Relationships allow you to work with data from multiple data sources or cubes as if they had already been prepared into a single data cube for you.



Logi Symphony

Add relationship

Set Up Relationship

A relationship specifies how two structures are related to each other. Select another data structure related to this one, then drag items from one list of elements and drop on the matching element in the other list.

Name

Description

Category

UserDefined

Target Structure

- Getting Started with Dundas BI
 - Data Connectors
 - Shared
 - Rich data
 - Simple data

[Sales].[SalesOrderDetail] Elements

- CarrierTrackingNumber
- LineTotal
- ModifiedDate
- OrderQty
- ProductID
- rowguid
- SalesOrderDetailID
- SalesOrderID
- SpecialOfferID
- UnitPrice

Target Elements

Under **Target Structure**, find the data connector containing related data and select the related table.

This data can be from a different data source. Match elements from each structure by dragging one from the left under **Source Elements** to the equivalent under **Referenced Elements**.

Set Up A Relationship

A relationship specifies how two structures are related to each other. Select another data structure related to this one, then drag items from one list of elements and drop on the matching element in the other list.

Name

[Production].[ProductCategory] - ProductSubcategory

Description

A user-defined relationship between the SQL table [ProductCategory] and the Excel table ProductSubcategory

Category

UserDefined

Target Structure

-  My Project
 -  Data Connectors
 -  Shared
 -  Adventure Works 2014
 -  ProductName
 -  ProductSubcategory
 -  Subcategory
 -  Cube Perspectives
 -  Data Cubes

Source Elements

ModifiedDate

Name

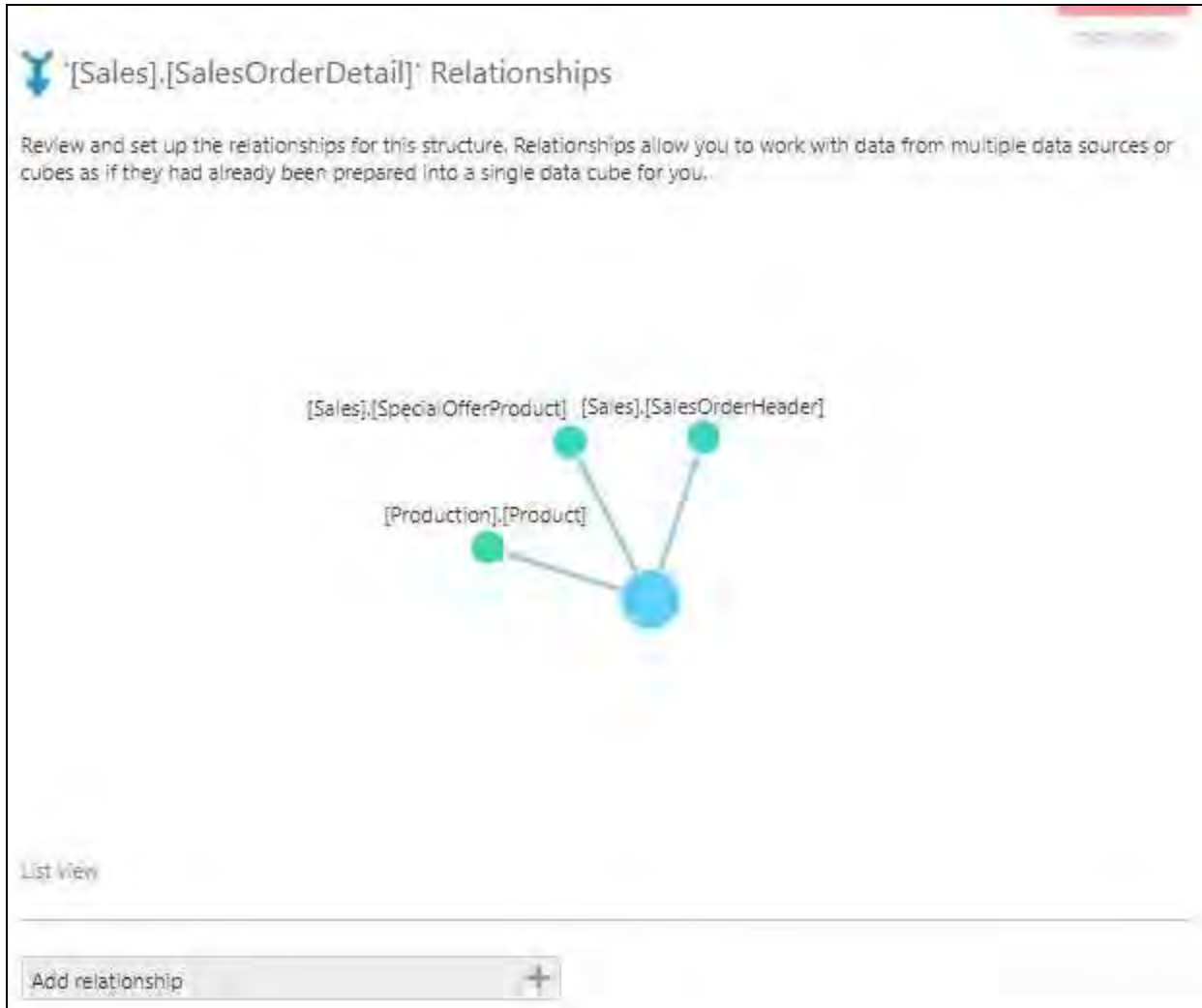
Referenced Elements

ProductSubcategoryID

ProductCategoryID 

Click the submit button at the bottom of the dialog to add the relationship.

The new relationship will now be add to the diagram in the *Relationships* dialog as a user-defined relationship.

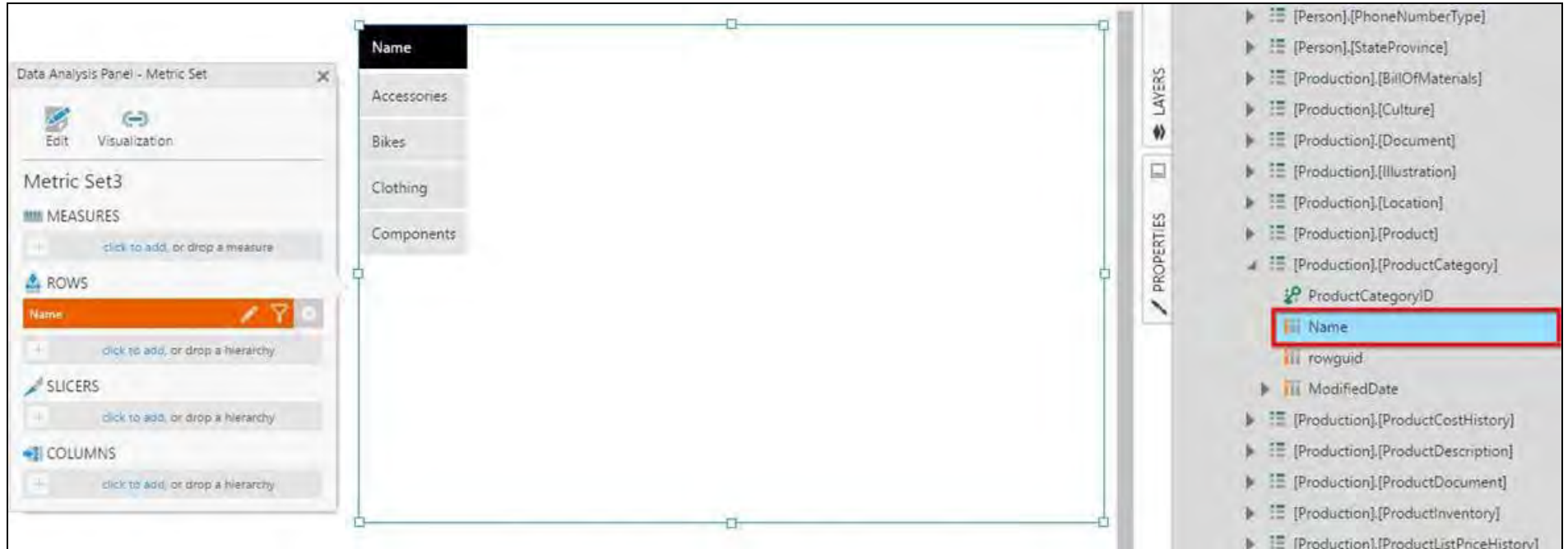


Using Relationships

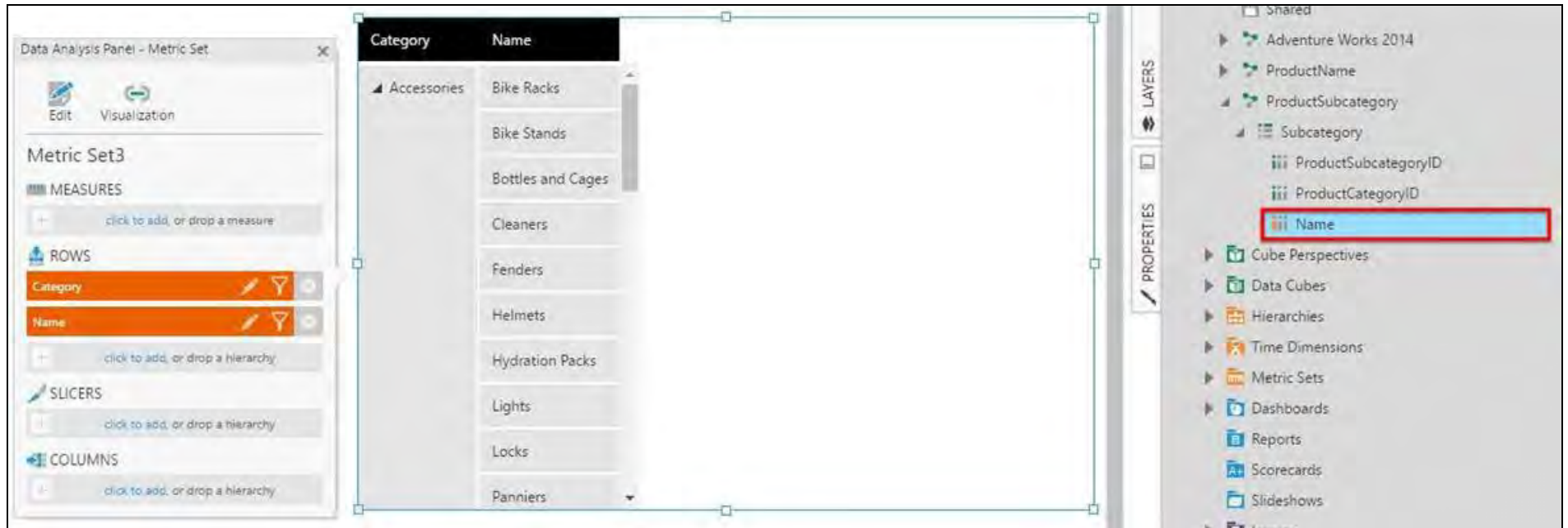
Relationships are used when selecting data from different tables and structures in a metric set, whether in the full-screen metric set editor or when working with metric sets on a dashboard or another view. They are also used when creating a [hierarchy](#) that contains multiple levels.

To demonstrate, we will create a single metric set as shown below based on a database table and two Excel spreadsheet files, after relationships have been defined.

In a new dashboard, drag a column from the table *[Production].[ProductCategory]* to the canvas.



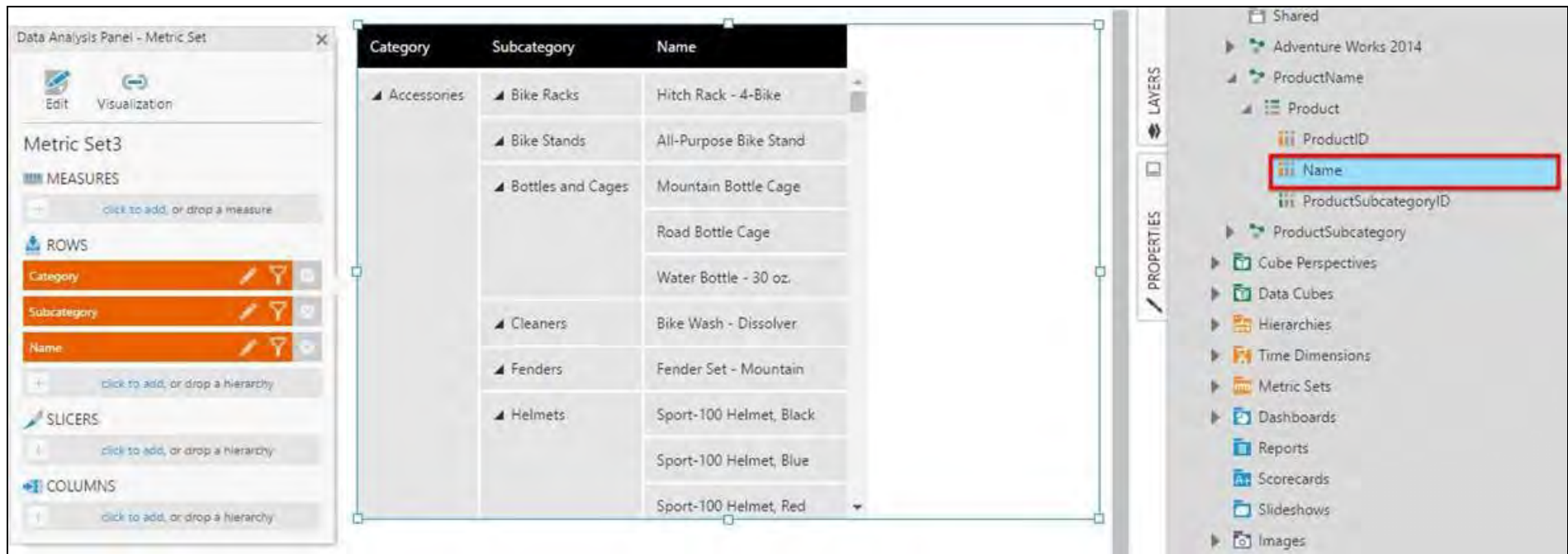
Drag a column from an Excel sheet to **Rows** in the Data Analysis Panel.



The screenshot displays the Logi Symphony v24 Data Analysis Panel. On the left, the 'Metric Set3' panel shows the configuration for a metric set. The 'ROWS' section contains two fields: 'Category' and 'Name'. The 'COLUMNS' section is empty. The 'MEASURES' section is also empty. The 'SLICERS' section is empty. The 'LAYERS' panel on the right shows a list of data sources, including 'Adventure Works 2014', 'ProductName', 'ProductSubcategory', 'Subcategory', 'ProductSubcategoryID', 'ProductCategoryID', and 'Name'. The 'Name' field is highlighted with a red box. The 'PROPERTIES' panel on the right shows the properties of the selected 'Name' field, including 'Name', 'ProductCategoryID', and 'ProductSubcategoryID'.

Category	Name
Accessories	Bike Racks
	Bike Stands
	Bottles and Cages
	Cleaners
	Fenders
	Helmets
	Hydration Packs
	Lights
	Locks
	Panniers

Drag a column containing additional details from a different Excel data connector.

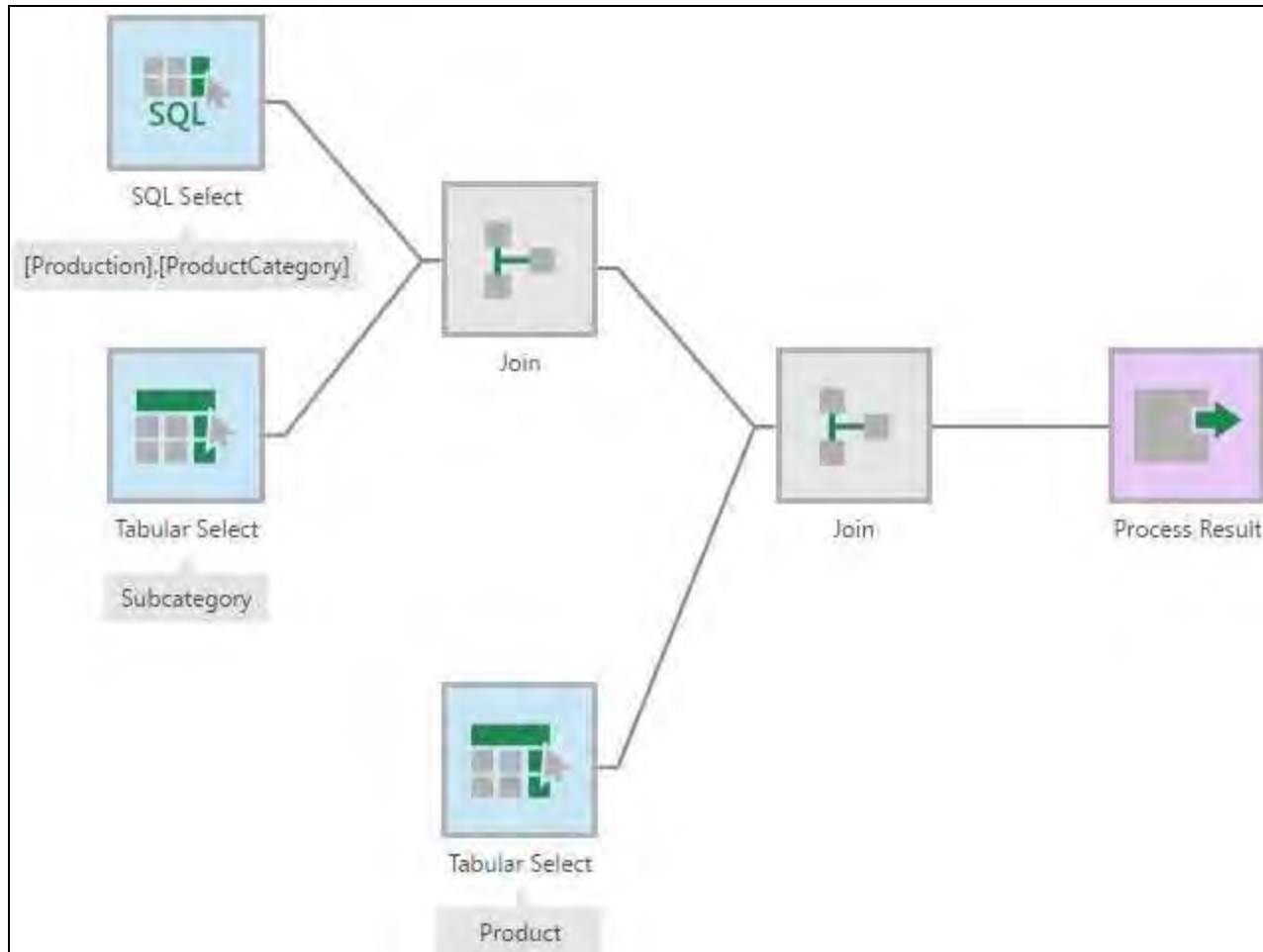


The screenshot displays the 'Data Analysis Panel - Metric Set' interface. On the left, the 'Metric Set3' panel shows a table with columns 'Category', 'Subcategory', and 'Name'. The 'ROWS' section lists 'Category', 'Subcategory', and 'Name'. The 'SLICERS' and 'COLUMNS' sections are empty. The main table displays a list of products under the 'Accessories' category, including 'Bike Racks', 'Bike Stands', 'Bottles and Cages', 'Cleaners', 'Fenders', and 'Helmets'. On the right, the 'LAYERS' and 'PROPERTIES' panels are visible. The 'LAYERS' panel shows a tree structure with 'Shared' at the top, followed by 'Adventure Works 2014', 'ProductName', 'Product', 'ProductID', 'Name' (highlighted with a red box), and 'ProductSubcategoryID'. The 'PROPERTIES' panel shows a list of data cubes, including 'Cube Perspectives', 'Data Cubes', 'Hierarchies', 'Time Dimensions', 'Metric Sets', 'Dashboards', 'Reports', 'Scorecards', 'Slideshows', and 'Images'.

Category	Subcategory	Name
Accessories	Bike Racks	Hitch Rack - 4-Bike
	Bike Stands	All-Purpose Bike Stand
	Bottles and Cages	Mountain Bottle Cage
		Road Bottle Cage
		Water Bottle - 30 oz.
	Cleaners	Bike Wash - Dissolver
	Fenders	Fender Set - Mountain
	Helmets	Sport-100 Helmet, Black
		Sport-100 Helmet, Blue
		Sport-100 Helmet, Red

In the background, this created automatic joins between the three tables based on the created relationships. [Automatic joins and hierarchies](#) shows you how to view and edit the automatic joins, along with more examples of combining data.

You can also view these joins by [promoting the auto-generated metric set](#) if applicable, and then promoting the auto-generated source data cube behind the metric set. Edit this data cube (as a [Developer user](#)) to see the join transforms.



For more information, see

- [Symphony Metric Sets](#)
- [Create a Metric Set and Add a Filter](#)



Add a Slicer Comparison

This applies to: *Managed Dashboards, Managed Reports*

This walkthrough shows you how to add a slicer comparison, which lets you apply different filter values to a measure and compare the results together.

Walkthrough

A slicer comparison measure can be added if you have numeric data (a measure) and a slicer to filter it by.

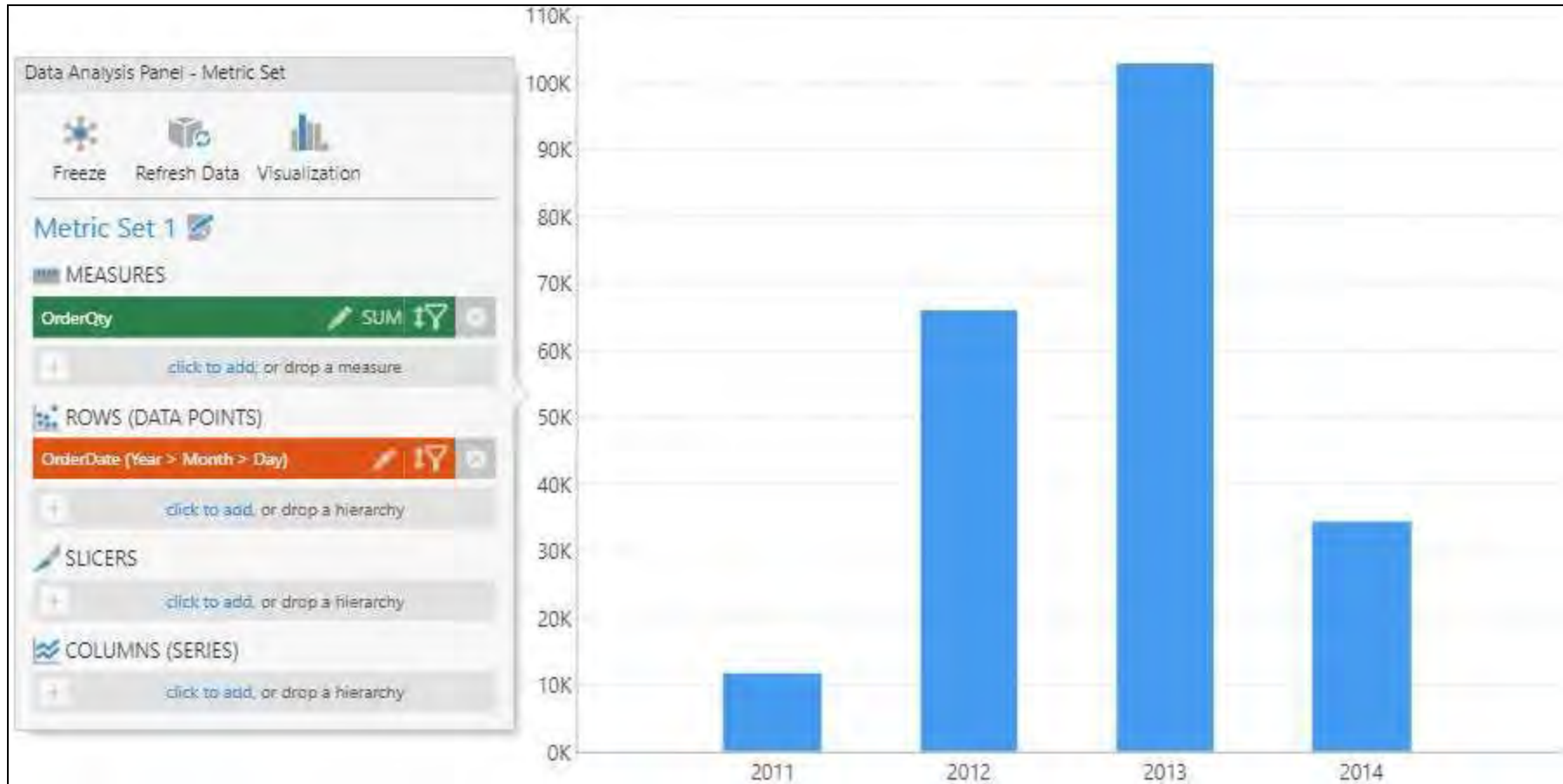
This example starts with a new metric set, but the same steps can be followed when editing a metric set checked out to you on a dashboard or another view.

Create a Metric Set

To create a new full screen [metric set](#), choose **Business** in the [main menu](#) and click **Create**.

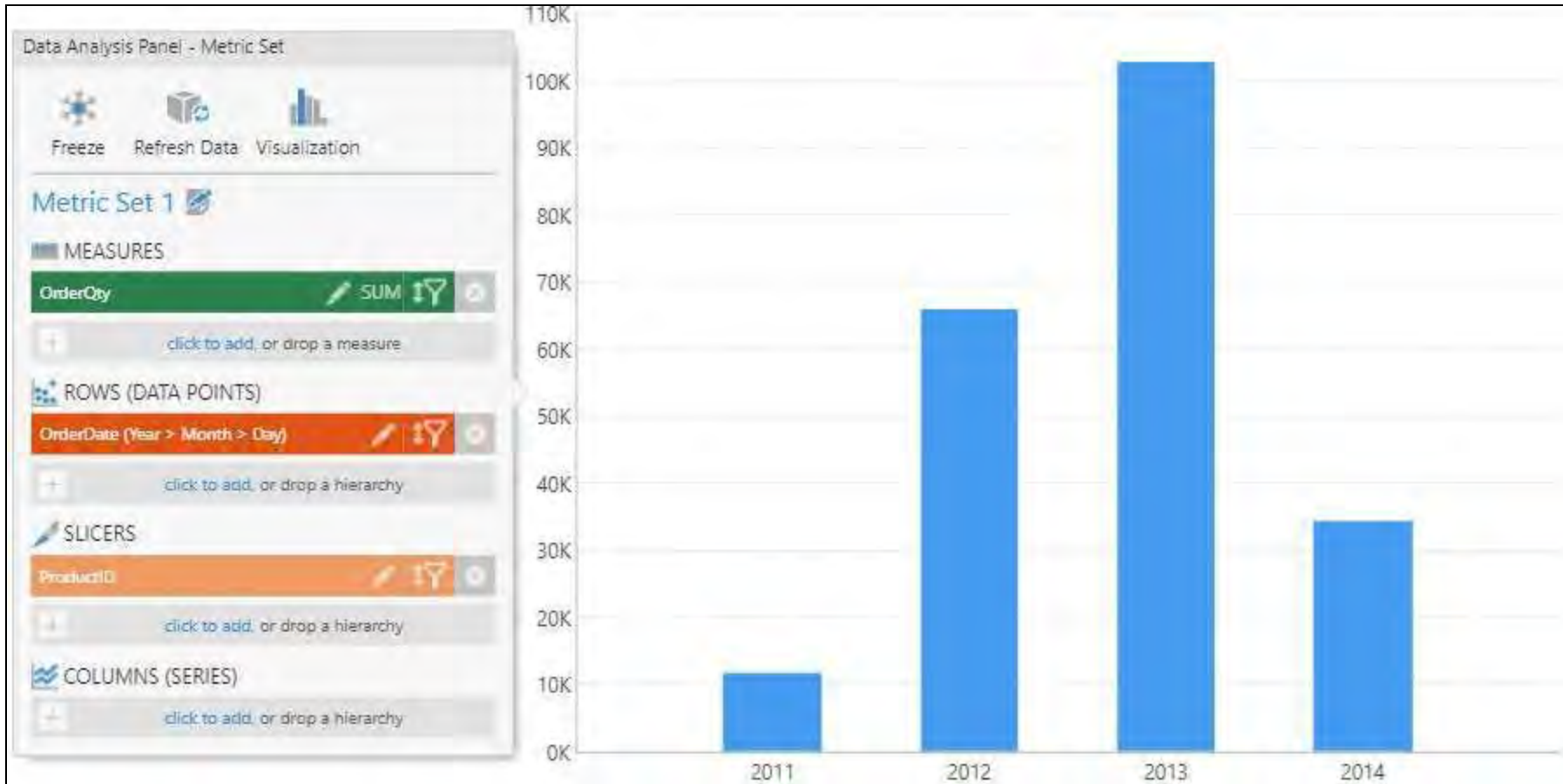
Choose a data source containing numeric data to use as a measure.

In our example, we chose a data cube containing the measure *OrderQty*, then dragged an extra time dimension hierarchy *OrderDate* from the **Explore** window to **Rows** (adding a hierarchy other than to Slicers is not necessary for a slicer comparison). After re-visualizing to a bar chart in the toolbar, this displays a bar for each year.



Add Slicers

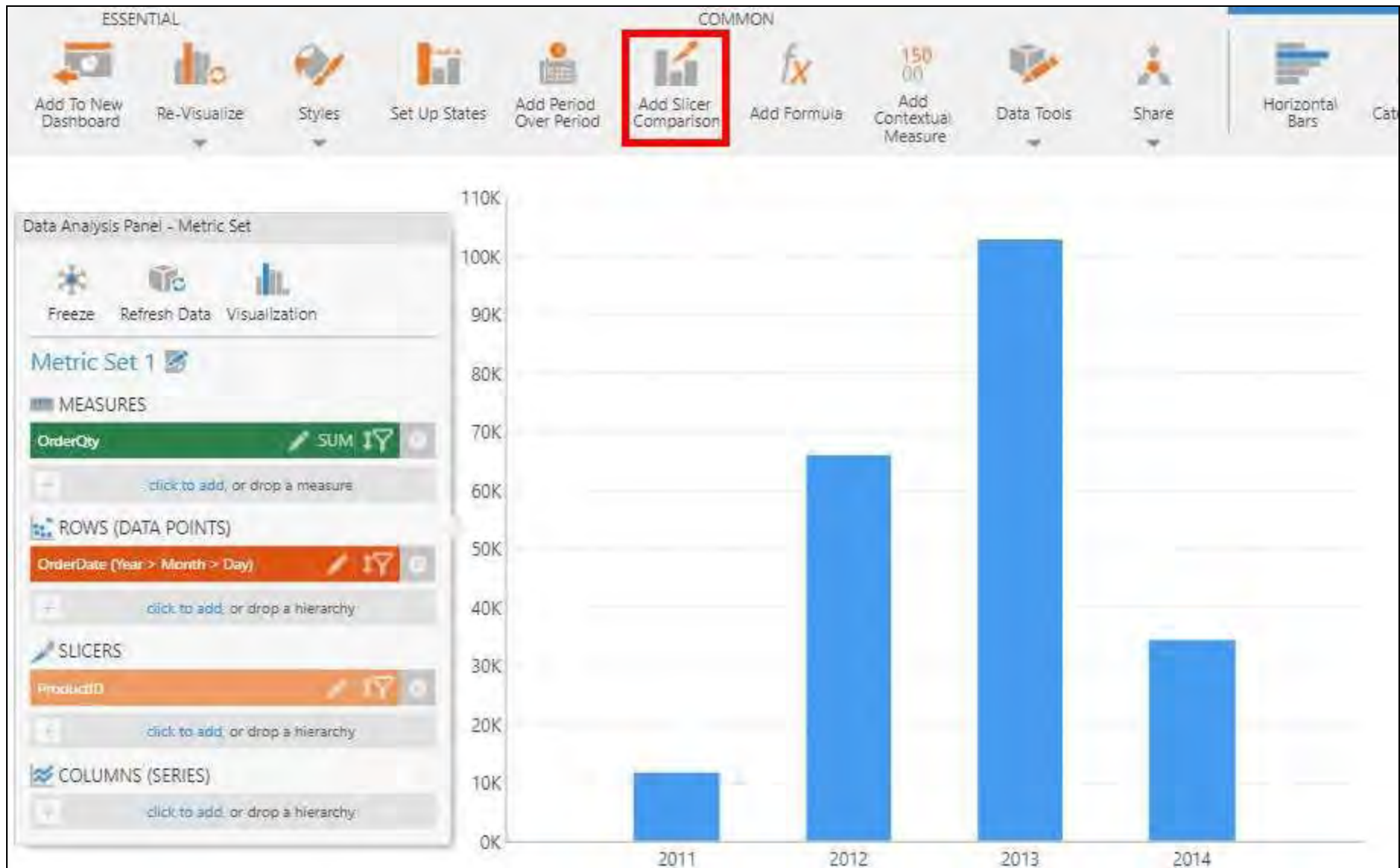
Add data under **Slicers**. In this example we dragged the *Product* hierarchy under Slicers.



A [slicer](#) allows you to filter the measure values without changing how the data is grouped into rows, columns, or data points. You can choose to filter your slicer, but in this example we will leave its filter value as the default setting *All*.

Add a Slicer Comparison

The slicer comparison adds a 'parallel' version of the measure that's displayed alongside the original but can be filtered differently. In the toolbar, click **Add Slicer Comparison**. This can be found under **Data Tools** if you are editing a view such as a dashboard.



In the *Slicer Comparison* dialog, confirm the **Measure** that you want to compare with different filtering (in our example there is only one).

Optionally provide a **Parallel Slicer Measure Name**, or leave it blank to generate one automatically.



Slicer Comparison

[more help](#)

Add a new measure to your data with different filter values applied for the slicer hierarchies. The new measure will be added to your metric set and the visualization.

Measure

OrderQty

Parallel Slicer Measure Name

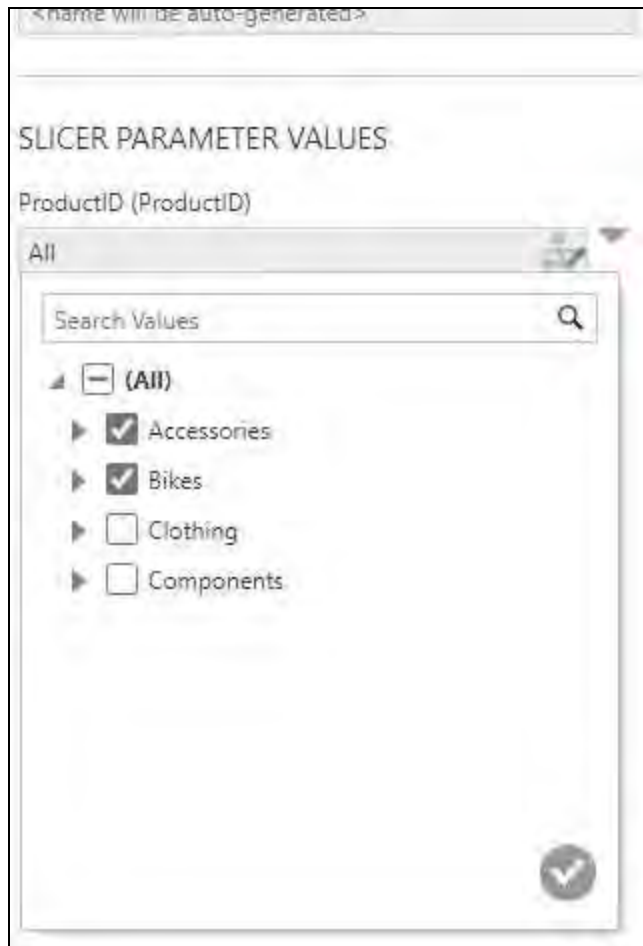
<name will be auto-generated>

SLICER PARAMETER VALUES

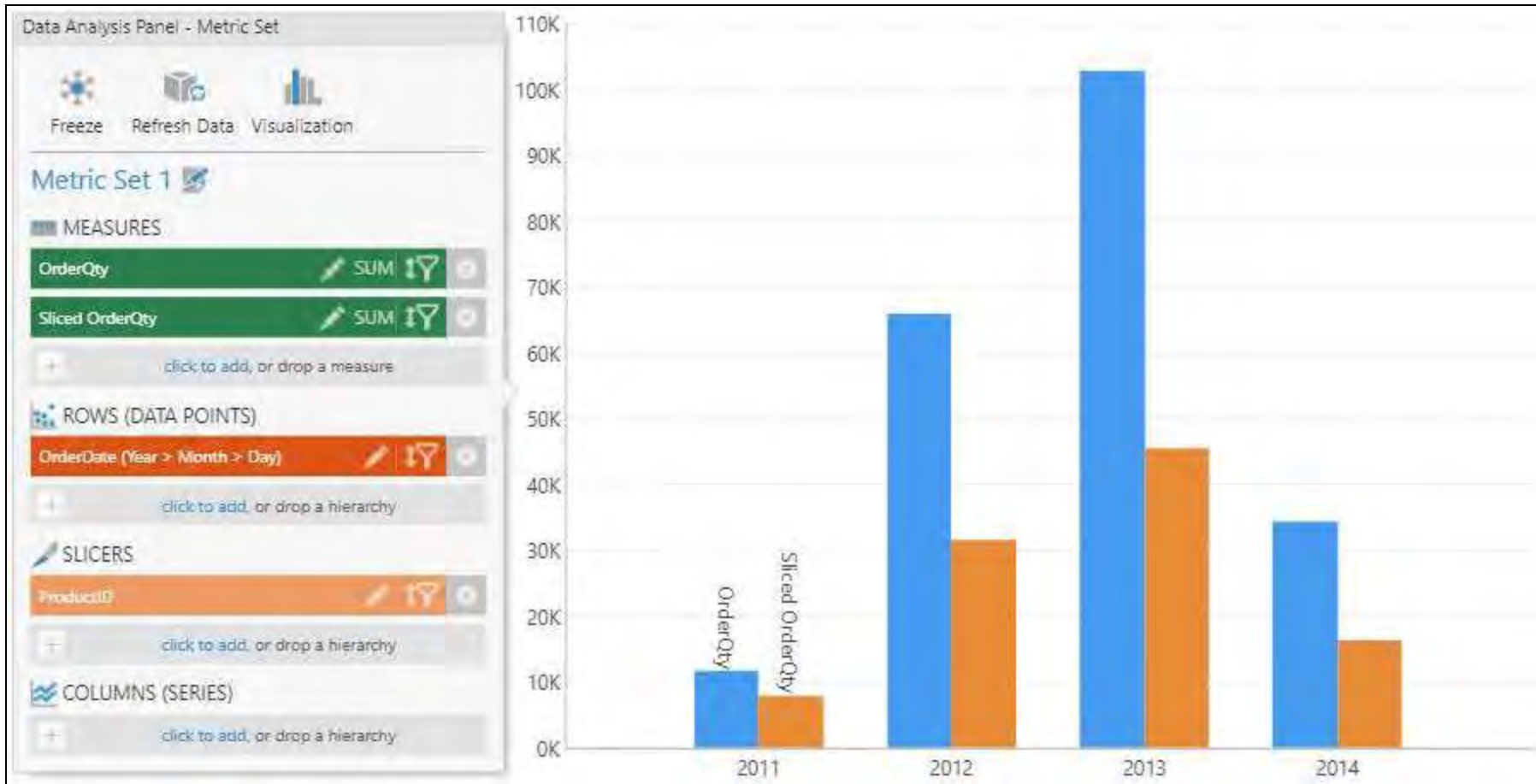
ProductID (ProductID)

All

Specify the **Slicer Parameter Values** for each slicer that you added. In this example, the orders are filtered by the product values *Accessories and Bikes*. Click the **Submit** button at the bottom of the dialog when finished.



A new slicer measure is added to the metric set and can be seen in the Data Analysis Panel. The visualization is updated to show the newly added slicer measure for comparison.



In this example, the slicer comparison allowed us to easily compare the order quantity for Accessories and Bikes (in orange) against the total order quantity for all products (in blue) for each year.

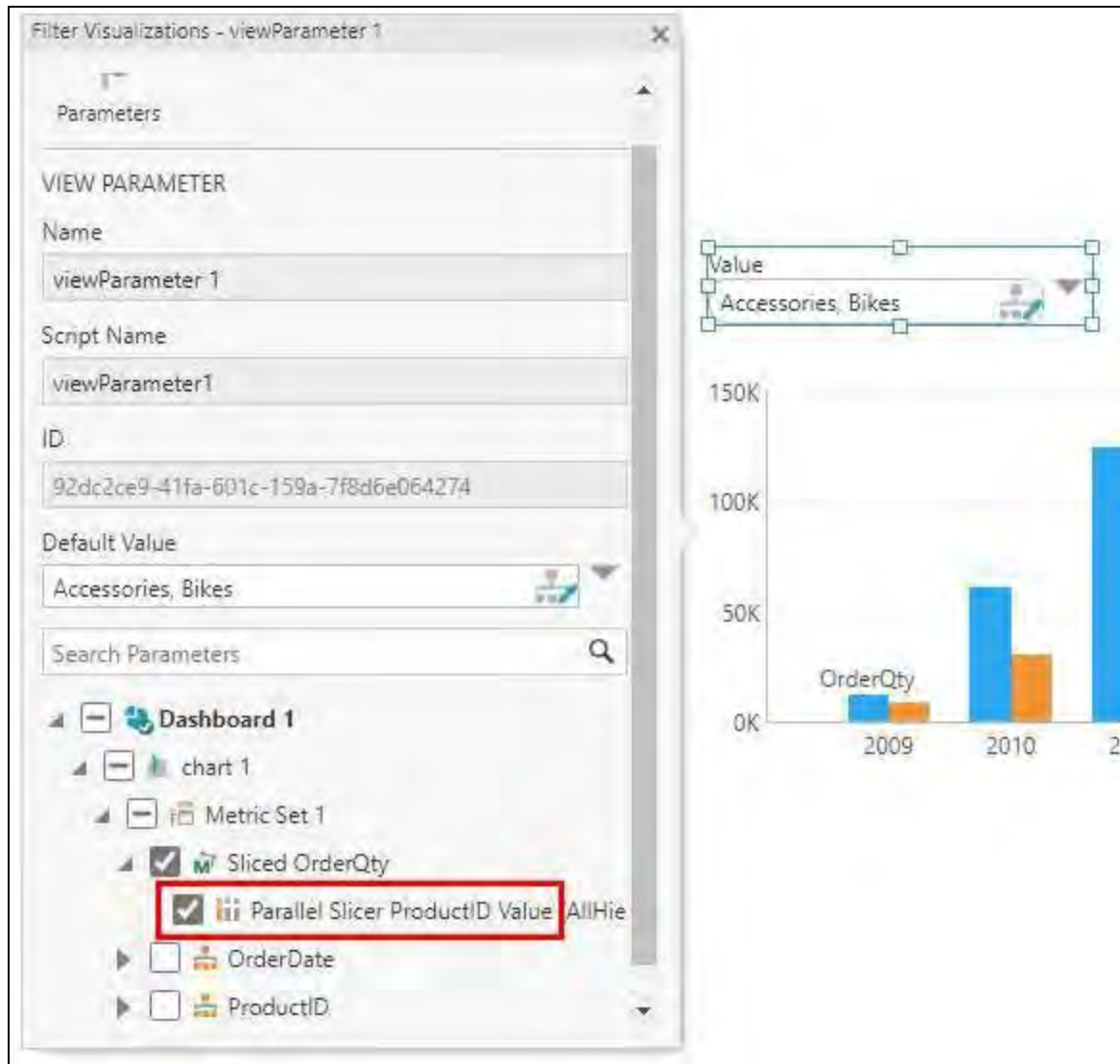
Add a Filter to Change While Viewing

The slicer parameter values specified above can be changed while viewing the metric set on a dashboard or another view by [adding a filter](#) or a [view parameter](#) and connecting it to a parallel slicer parameter.



Note: If you are editing the metric set on a dashboard when you add the slicer comparison, you can use the **Automatically Add Parameter Controls** option in the dialog as a shortcut.

For example, add a hierarchy **Value** filter and connect it to one of the parallel slicer parameters under the slicer measure. (Disconnect it from other parameters, such as the original slicer hierarchy *ProductID* in this example).



For more information, see:



- Archive of documentation for Logi Symphony v24

- Video: [Using The Data Analysis Panel](#)
- Video: [Filters](#)
- [Symphony Metric Sets](#)
- [Setting Up the Visualization](#)
- [Slicers Versus Columns and Rows](#)
- [Adding Filters](#)

Adding Contextual Measures

This applies to: *Managed Dashboards, Managed Reports*

A contextual measure lets you enter your own values to augment existing data in your metric set within Symphony. Contextual measures are part of the [metric set](#) and you can enter or change contextual measure values when it is checked out to you.

Contextual measure data is not affected by the check in or check out state of the metric set. Undo Check Out does not change or remove data.

Different values can be entered and displayed if you have multi-level hierarchies or slicer filters.

Note: You can also use a [formula](#) to enter a single constant value. which will instead apply for all hierarchy levels and filtering.

Adding a Contextual Measure

With the metric set checked out to you, click **Add Contextual Measure** in the toolbar. (When editing a dashboard or another view, you can find this option within **Data Tools** in the toolbar.)



Alternatively, you can **click to add** in the *Data Analysis Panel* and select **Contextual Measure**.

A Measure Value bar appears. For a single value that applies to every row or column (e.g., every sales person), enter it and click **apply**.



Enter a value and click apply

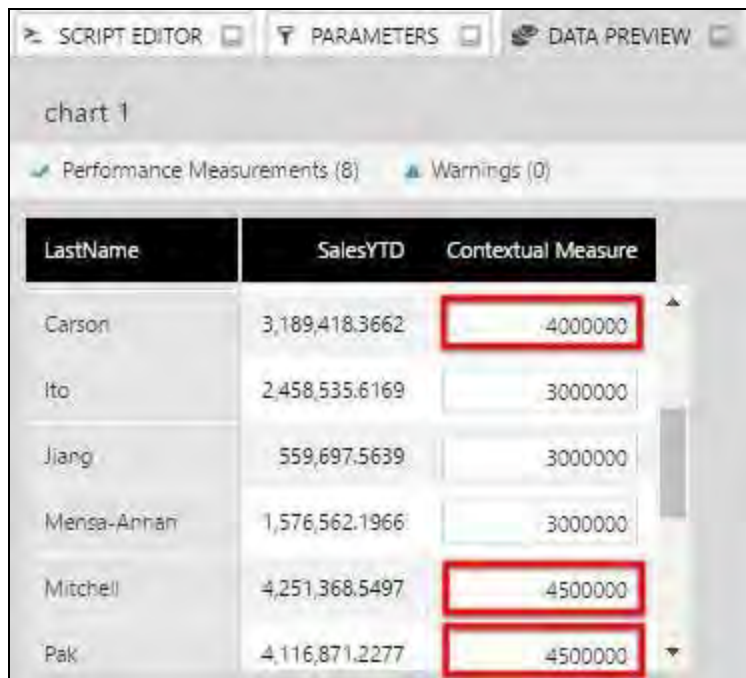
Note: If your visualization is a chart, you can click a position on the chart to enter the corresponding value into the measure value bar.

To enter different values for each row or column, click **multiple** at the far right of the measure value bar instead.



This opens the Data Preview window with editable fields for your contextual measure.

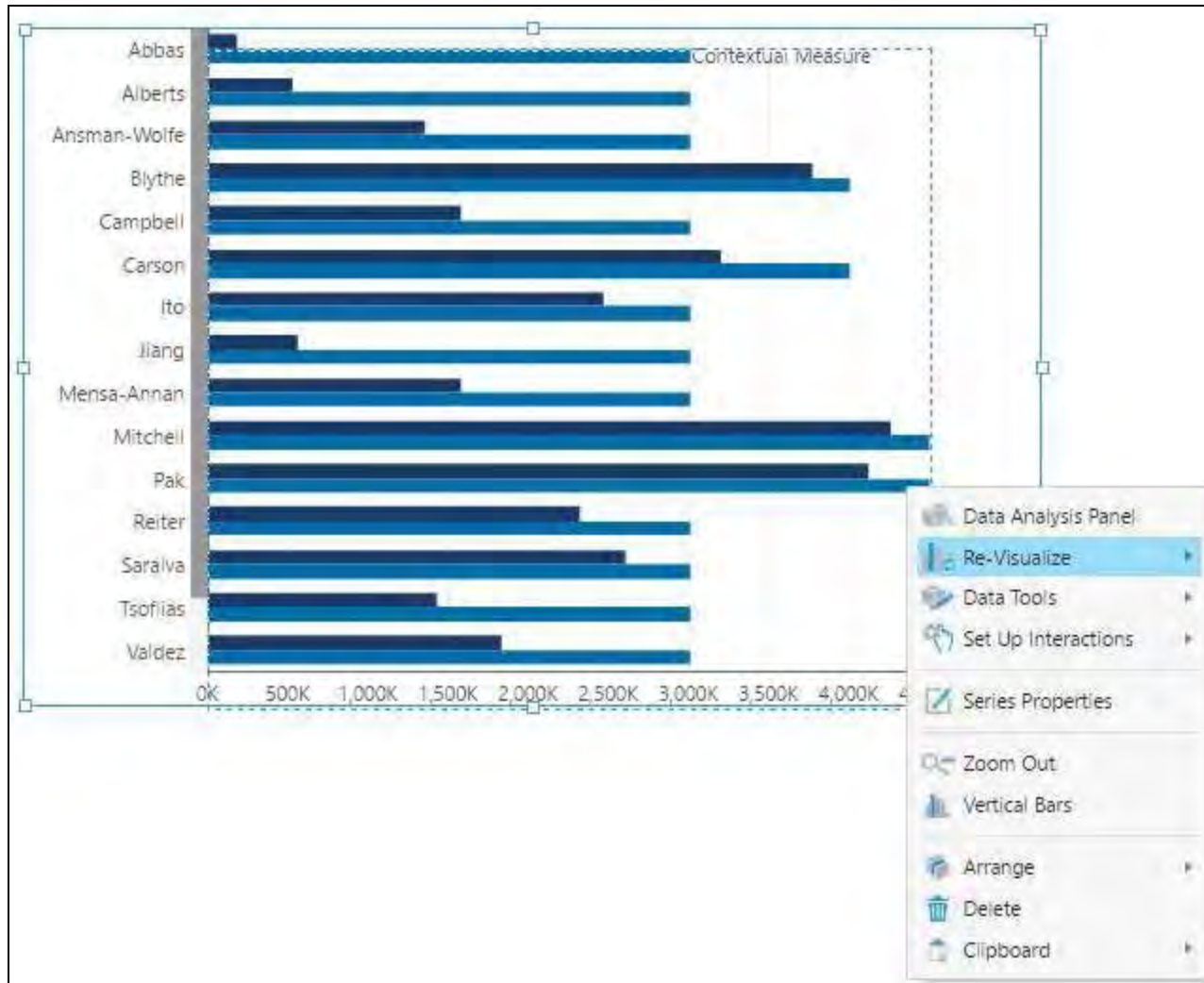
Click inside a field and type a value. You can type *Enter* on your keyboard to finish entering a value and move to the next row (or *Tab* to move to the next column if the metric set is [transposed](#)).



LastName	SalesYTD	Contextual Measure
Carson	3,189,418.3662	4000000
Ito	2,458,535.6169	3000000
Jiang	559,697.5639	3000000
Mensa-Annan	1,576,562.1966	3000000
Mitchell	4,251,368.5497	4500000
Pak	4,116,871.2277	4500000

Values you enter for a specific row or column will take precedence over a single value entered into the measure value bar, if you have entered both.

In a chart or table, you might now choose to change how the new measure is visualized, for example by right-clicking it and using the **Re-Visualize** option.



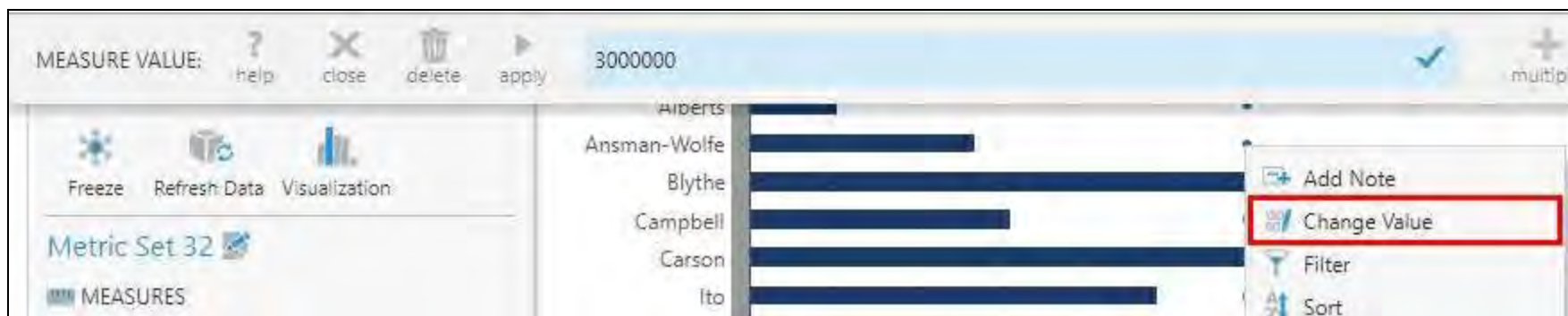
Editing the Measure Values

To enter or change values after the contextual measure was already added, click the number icon shown for the measure in the Data Analysis Panel.



This re-opens the measure value bar, where you can enter or change the single value here, or click **multiple** to change or enter values for each row or column.

Alternatively, you can right-click (or long-tap) a single value in the visualization and choose **Change Value**. (When editing a dashboard or another view, first choose **View** to interact with data in the visualization. The metric set must still be checked out to you to edit the contextual measure values directly.)



This opens the measure value bar for a specific value, where you can enter a new value for this specific row or column, or click **delete** to remove a value you previously entered.

Different Contexts

To allow appropriate values to be displayed for different contexts, the values you enter apply for whatever the current level of each hierarchy is and whatever the slicer filtering is on the metric set.

To enter values that will apply at other levels of the hierarchy, you can right-click (or long-tap) the visualization and choose **Change Level**.



Now you can re-open the measure value bar as described above to enter a value for this level. For example, you can change the level from *Year to Month* and enter a monthly goal that is different from your yearly goal.

All values you enter are saved, and will be displayed later when the level is changed back.

Similarly, you can filter any hierarchies placed on **Slicers** in the Data Analysis Panel and enter a different value into the measure value bar.

- When working with hierarchies, the values you provide are applied only to the current level. These values will not be distributed to lower hierarchy levels or aggregated into higher hierarchy levels. The highest-level values (for example Years) will be the only ones contributing to the Grand Total.
- When entering values for different hierarchy levels or filtering, the metric set settings are used when storing the values. When editing a metric set on a dashboard with filters attached, you can change these settings by opening the hierarchy's settings dialog from the Data Analysis Panel, or by editing the metric set full-screen.

For more information, see:

- [Adding Formulas](#)
- [Symphony Metric Sets](#)
- [Collaboration Using Data Annotations](#)
- [Data Input](#)

Automatic Joins and Hierarchies

This applies to: *Managed Dashboards, Managed Reports*

This article demonstrates how to use automatic joining and hierarchies to combine related data together while working with a [metric set](#).

In keeping with the self-serve paradigm, some things are done automatically to save you time as you drag-and-drop items from the **Explore** window to the canvas, such as joining two tables of data together when you drag a column from each into one metric set.

You can also drag a *hierarchy* that was set up ahead of time onto a regular column of matching data, to augment it with additional capabilities such as multiple hierarchy levels and additional details made available as *attributes*.

The [Adventure Works 2014](#) database is used as an example.

Related video: [Creating Custom Categorical Hierarchies](#)



Note: Automatic joining in a metric set is an alternative to editing a data cube to [join data together](#) and [replace columns with hierarchies](#).

Automatic Joins

This example begins with a table visualization like the one in [View Data With a Chart and Drill Down](#), which displays *OrderQty* as a measure, and a date-time hierarchy on the **Rows** axis.

Data Analysis Panel - Metric Set

 Freeze
 Refresh Data
 Visualization

Metric Set 3

MEASURES

OrderQty SUM

+ click to add, or drop a measure

ROWS

ModifiedDate

+ click to add, or drop a hierarchy

SLICERS

+ click to add, or drop a hierarchy

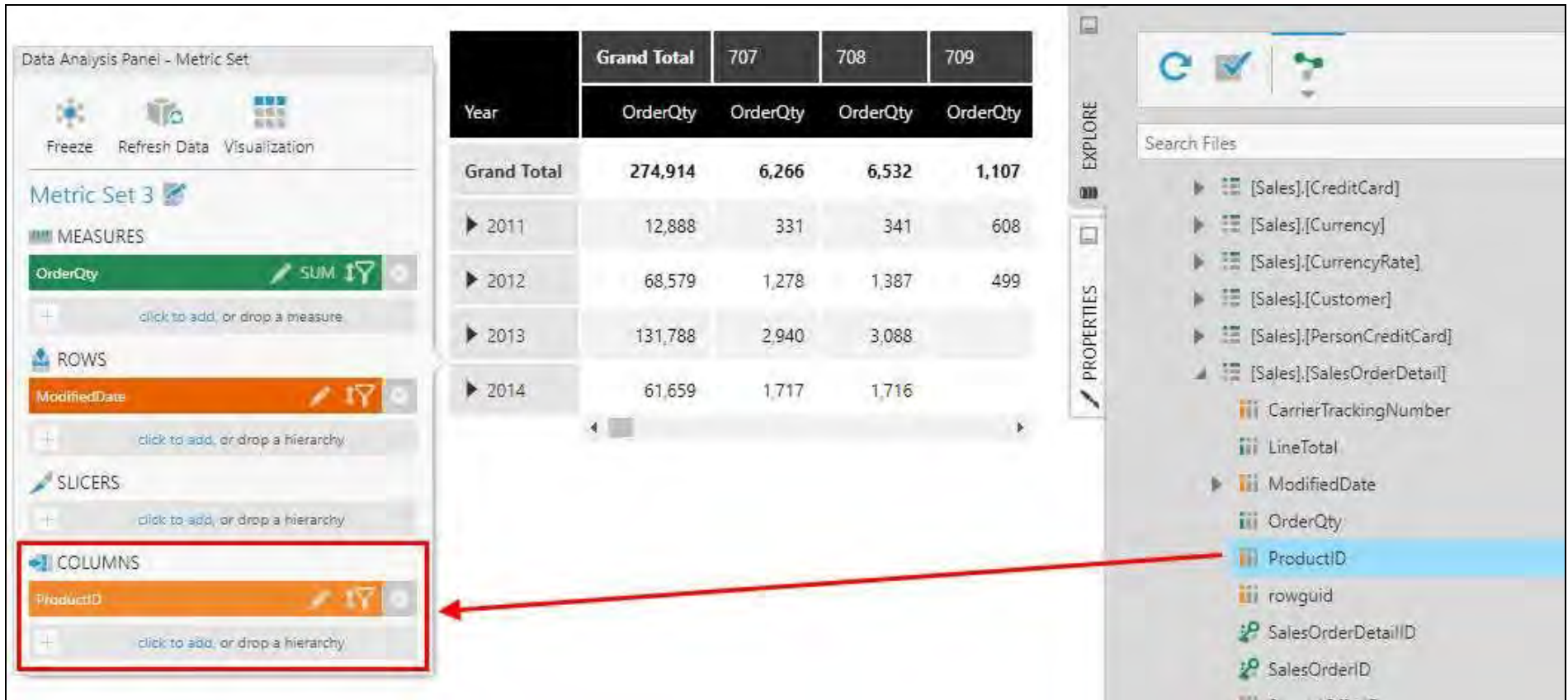
COLUMNS

+ click to add, or drop a hierarchy

Year	OrderQty
Grand Total	274,914
▶ 2011	12,888
▶ 2012	68,579
▶ 2013	131,788
▶ 2014	61,659

Display ProductID on Columns

In the **Explore** window, expand the data connector to find the table *[Sales].[SalesOrderDetail]*, and expand it as well. Drag the *ProductID* column onto **Columns** in the *Data Analysis Panel*.



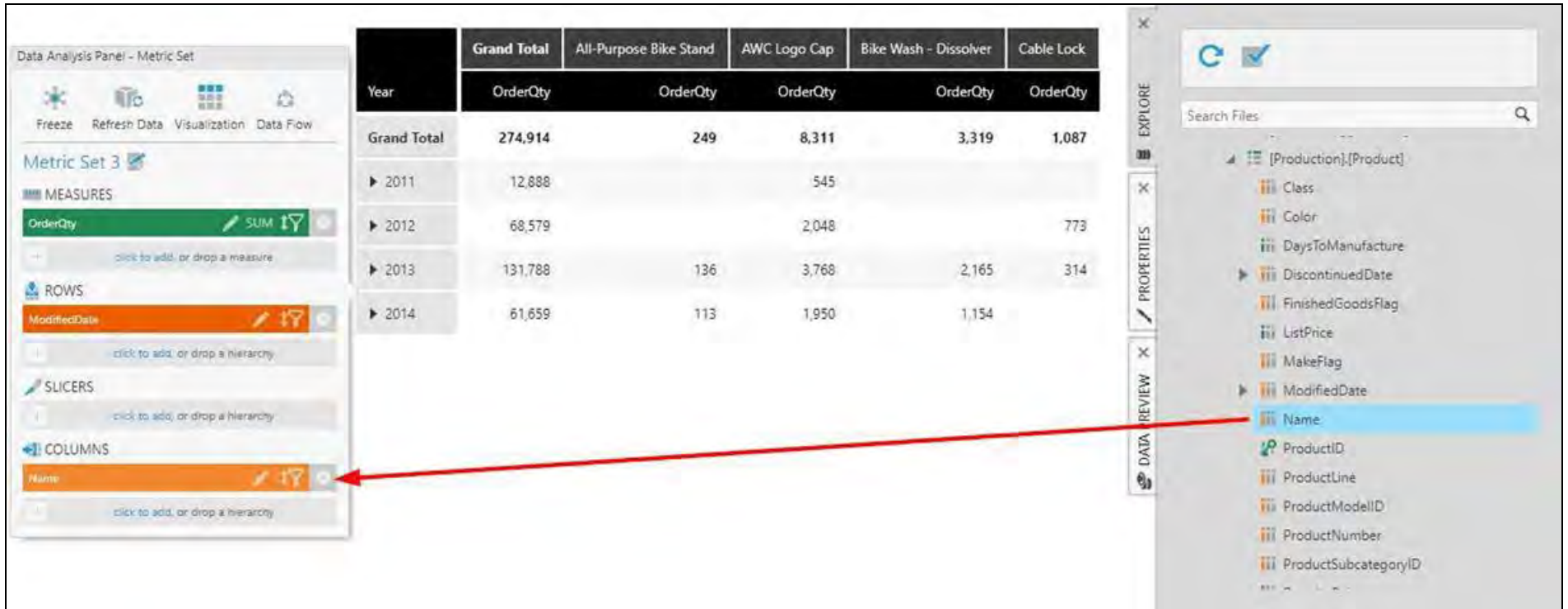
The screenshot shows the Logi Symphony v24 interface. On the left is the 'Data Analysis Panel - Metric Set'. It has tabs for 'Freeze', 'Refresh Data', and 'Visualization'. Below these are sections for 'MEASURES', 'ROWS', 'SLICERS', and 'COLUMNS'. The 'COLUMNS' section is highlighted with a red box, and 'ProductID' is listed as a column. In the center is a table visualization. The table has columns for 'Year', 'Grand Total', and three columns labeled '707', '708', and '709'. The rows represent years from 2011 to 2014. On the right is the 'EXPLORE' window, which shows a search bar and a list of tables and columns. 'ProductID' is highlighted in the list. A red arrow points from the 'ProductID' in the EXPLORE window to the 'ProductID' in the COLUMNS section of the Data Analysis Panel.

	Grand Total	707	708	709
Year	OrderQty	OrderQty	OrderQty	OrderQty
Grand Total	274,914	6,266	6,532	1,107
2011	12,888	331	341	608
2012	68,579	1,278	1,387	499
2013	131,788	2,940	3,088	
2014	61,659	1,717	1,716	

The table visualization is reconfigured to additionally display Product IDs along the columns axis.

Replace ProductID with Name

Since showing product ID values is not very meaningful, find the *[Production].[Product]* table in the **Explore** window and expand it. Drag the Name column onto **Columns** in the *Data Analysis Panel*, replacing the ProductID column.



The screenshot displays the Logi Symphony v24 interface. On the left, the 'Data Analysis Panel - Metric Set' is visible, showing a table visualization. The table has columns for 'Year', 'Grand Total', 'All-Purpose Bike Stand', 'AWC Logo Cap', 'Bike Wash - Dissolver', and 'Cable Lock'. The rows represent years from 2011 to 2014. A red arrow points from the 'Name' field in the 'COLUMNS' section of the panel to the 'Name' field in the table's column headers.

Year	Grand Total	All-Purpose Bike Stand	AWC Logo Cap	Bike Wash - Dissolver	Cable Lock
Grand Total	274,914	249	8,311	3,319	1,087
2011	12,888		545		
2012	68,579		2,048		773
2013	131,788	136	3,768	2,165	314
2014	61,659	113	1,950	1,154	

On the right, the 'EXPLORE' panel shows a list of fields. The 'Name' field is highlighted in blue. A red arrow points from the 'Name' field in the 'COLUMNS' section of the 'Data Analysis Panel' to the 'Name' field in the 'EXPLORE' panel.

The table visualization is reconfigured to show product names on the columns axis instead of product IDs (in alphabetical order, by default).

Data Analysis Panel - Metric Set

Freeze Refresh Data Visualization Data Flow

Metric Set 3

MEASURES

OrderQty SUM

click to add, or drop a measure

ROWS

ModifiedDate

click to add, or drop a hierarchy

SLICERS

click to add, or drop a hierarchy

COLUMNS

Name

click to add, or drop a hierarchy

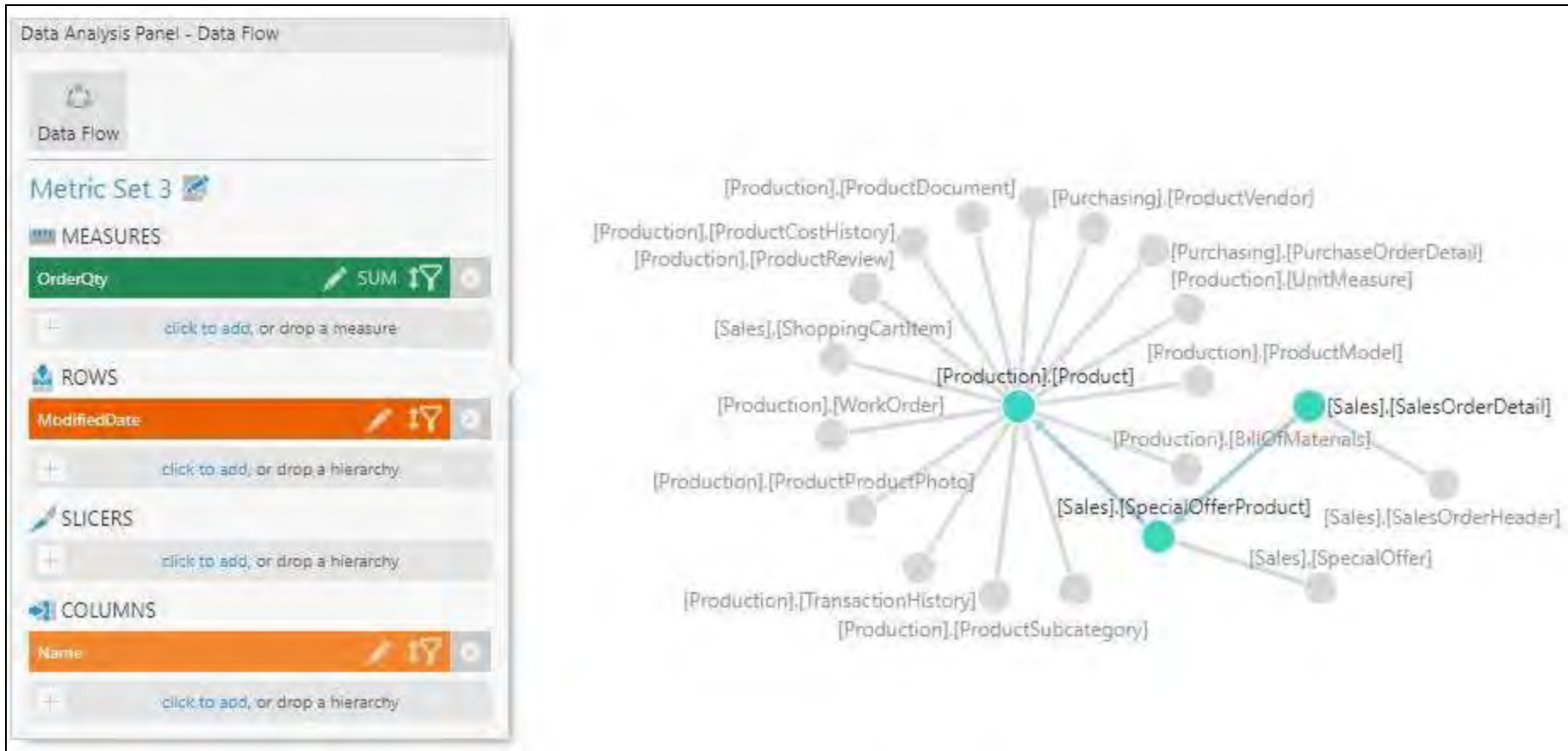
	Grand Total	All-Purpose Bike Stand	AWC Logo Cap	Bike Wash - Dissolver	Cable Lock
Year	OrderQty	OrderQty	OrderQty	OrderQty	OrderQty
Grand Total	274,914	249	8,311	3,319	1,087
▶ 2011	12,888		545		
▶ 2012	68,579		2,048		773
▶ 2013	131,788	136	3,768	2,165	314
▶ 2014	61,659	113	1,950	1,154	

Behind the scenes, this automatically performed a join between the *SalesOrderDetail* and *Product* tables based on their common column, *ProductID*, using the native relationships defined in the database. You can also create your own relationships for joining between any two data sources as described in the article [Add a relationship between tables](#), or matching columns will be identified for you automatically if no relationship exists.

You could choose to drag additional data from this table to take advantage of auto-joining, for example by dragging *Color* to **Slicers** in the Data Analysis Panel.

Edit Automatic Joins

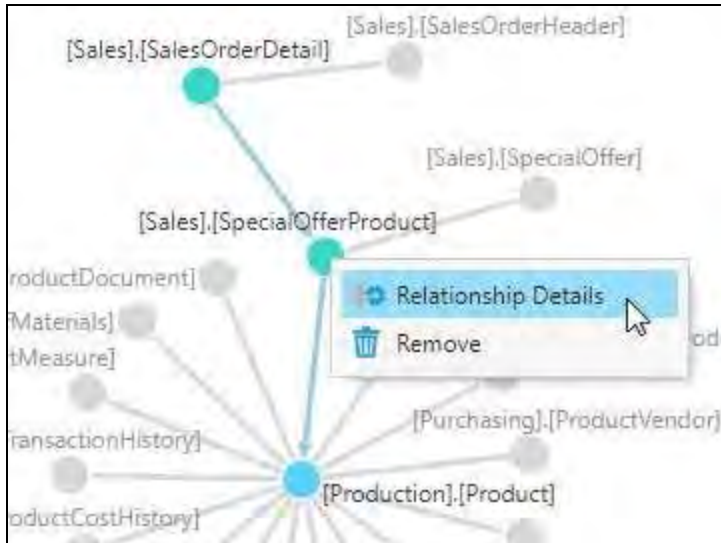
After adding data from multiple tables or data sources, you can see and change how the data is automatically joined. While editing the metric set full screen, click the **Data Flow** button that now appears in the Data Analysis Panel.



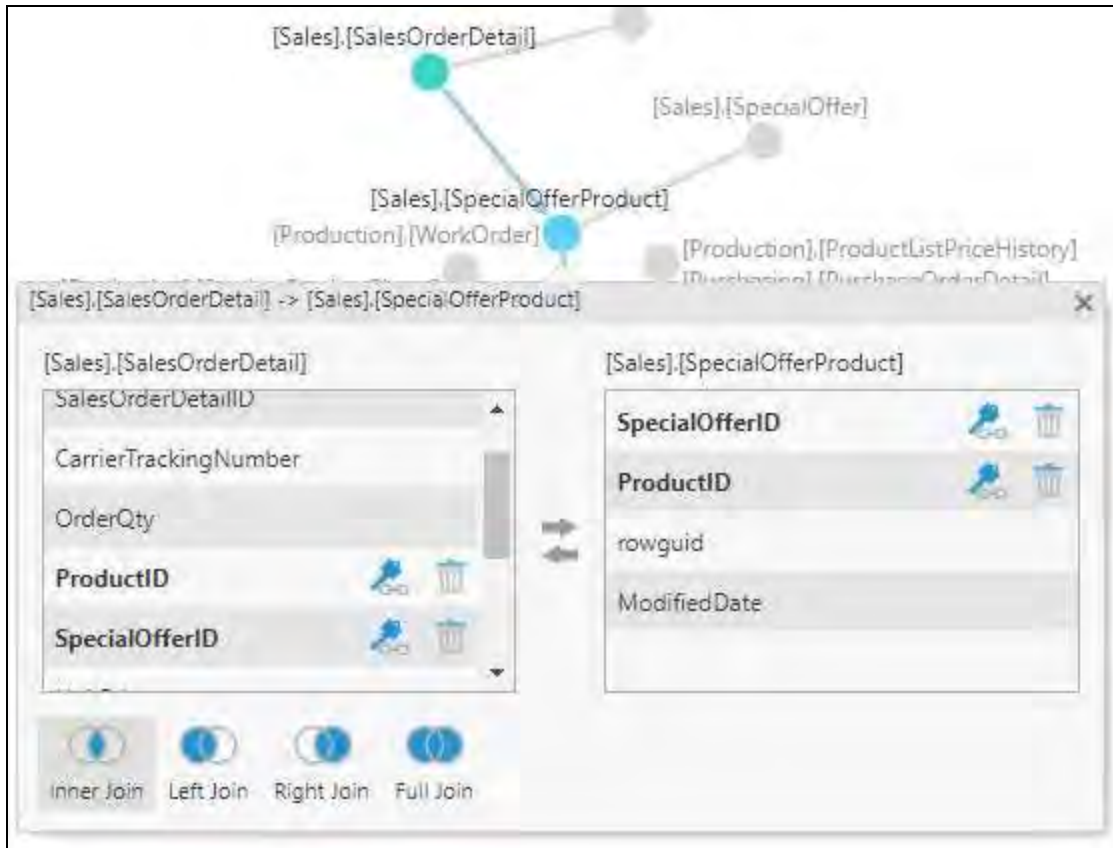
Note: When editing a metric set on a dashboard, you can edit the metric set full screen from either the Data Analysis Panel or under **Data Tools** in the toolbar.

The green and blue nodes in the diagram represent tables that have been joined, and gray nodes represent [additional related data](#) that is not used but available to add.

The arrows indicate the order in which the tables were joined. View the join relationship details for any node that an arrow points to by clicking it, or by right-clicking it and selecting **Relationship Details**.



The relationship popup lists the columns of the two input tables and includes settings to determine how they are joined.



Note: The left table in the relationship popup may be listed as Data Flow if two or more tables were already joined before connecting to the one you clicked.

The existing key elements are indicated by key icons shown next to the column in each table that matches with the other.

To define a key binding, drag a column from one input table and drop it onto the column containing matching values in the other table. You can click the **Delete** icon to remove a key binding when you have multiple.

Change the join type if you want to change which records should be included:

- *Inner Join* - Only records with matching key values in both input tables.
- *Left Join* - Only records with a key value in Left Input.



- Archive of documentation for Logi Symphony v24

- *Right Join* - Only records with a key value in Right Input.
- *Full Join* - All original records will be included.

When a node is selected (shown as blue), you can open **Data Preview** to see the data that has been joined up to that point and verify that it looks correct.

Data Analysis Panel - Data Flow

Data Flow

Metric Set 5

MEASURES

OrderQty SUM

click to add, or drop a measure

ROWS

ModifiedDate

click to add, or drop a hierarchy

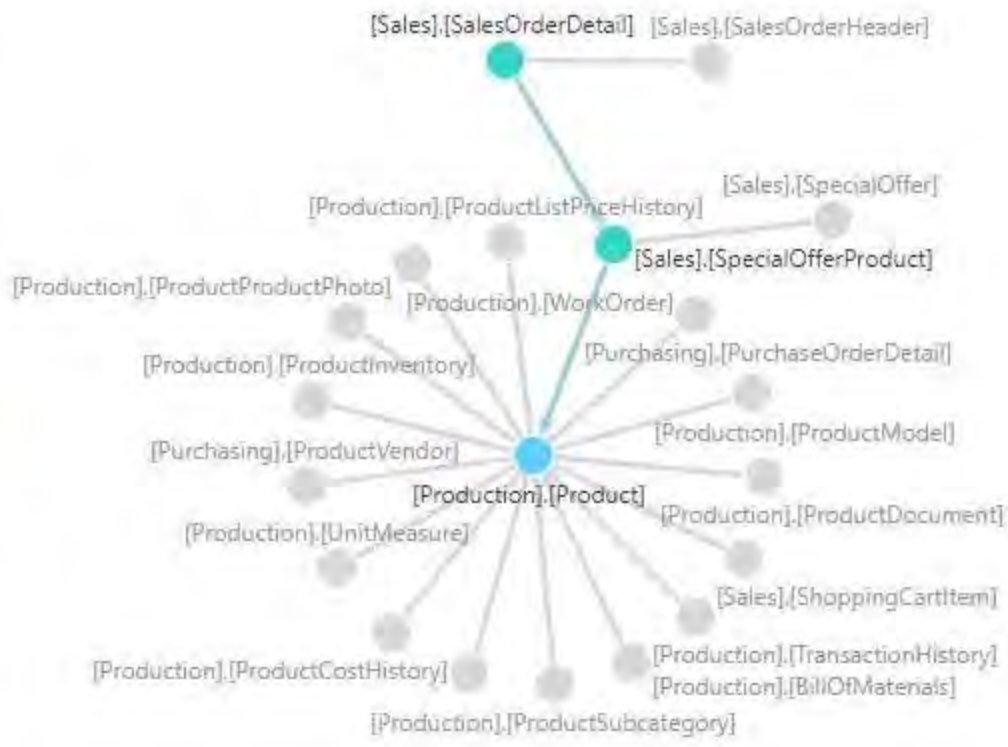
SLICERS

click to add, or drop a hierarchy

COLUMNS

Name

click to add, or drop a hierarchy



DATA PREVIEW

[Sales].[SalesOrderDetail] -> [Sales].[SpecialOfferProduct] -> [Production].[Product]

Warnings (0) Statistics

SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal
57058	61157	4A45-4678-8D	1	958	1	445.4100	0.0000	445.41
57058	61158	4A45-4678-8D	1	968	1	1430.4420	0.0000	1430.442
57058	61159	4A45-4678-8D	2	916	1	31.5840	0.0000	63.168



When [relationships](#) have been created to your current data by other users or in your database, the related tables are displayed as gray nodes. You can right-click one and select **Expand** to view all the other data related to that table in the diagram to explore.

To join with a table represented by a gray node and add it to the visualization, right-click it and select **Add to Visualization**.

To join with other data not in the diagram, drag a table from the Explore window. This will add a new node in the diagram and a new join relationship that you can configure.

Click the **Data Flow** button again to return to your own visualization and the regular metric set options on the Data Analysis Panel.

Hierarchies

Now we will create a hierarchy to allow us to do more with this column of products. Hierarchies that are set up ahead of time like these are sometimes called 'user' hierarchies.

Create a Product Hierarchy

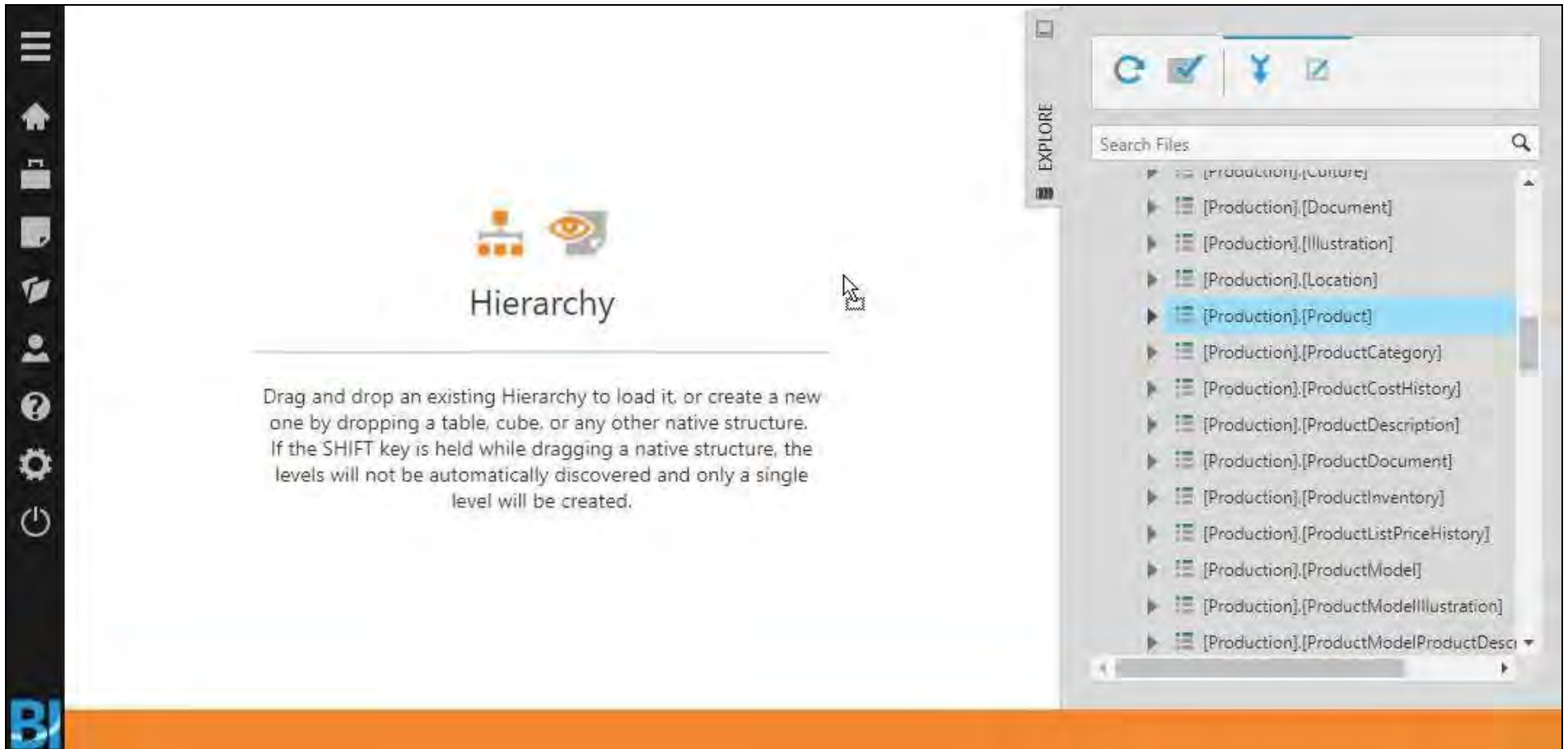
Choose **Business** in the [main menu](#), click to choose the **Hierarchy** file type, then click **Create** and choose a **Standard** hierarchy.

This opens the *Hierarchy* editor, which lets you define a hierarchy by dragging items from the Explore window or a data file from your desktop.

As an example, expand your data connector in the **Explore** window and find the *[Production].[Product]* table. Drag the table (or one of its columns) to the canvas.

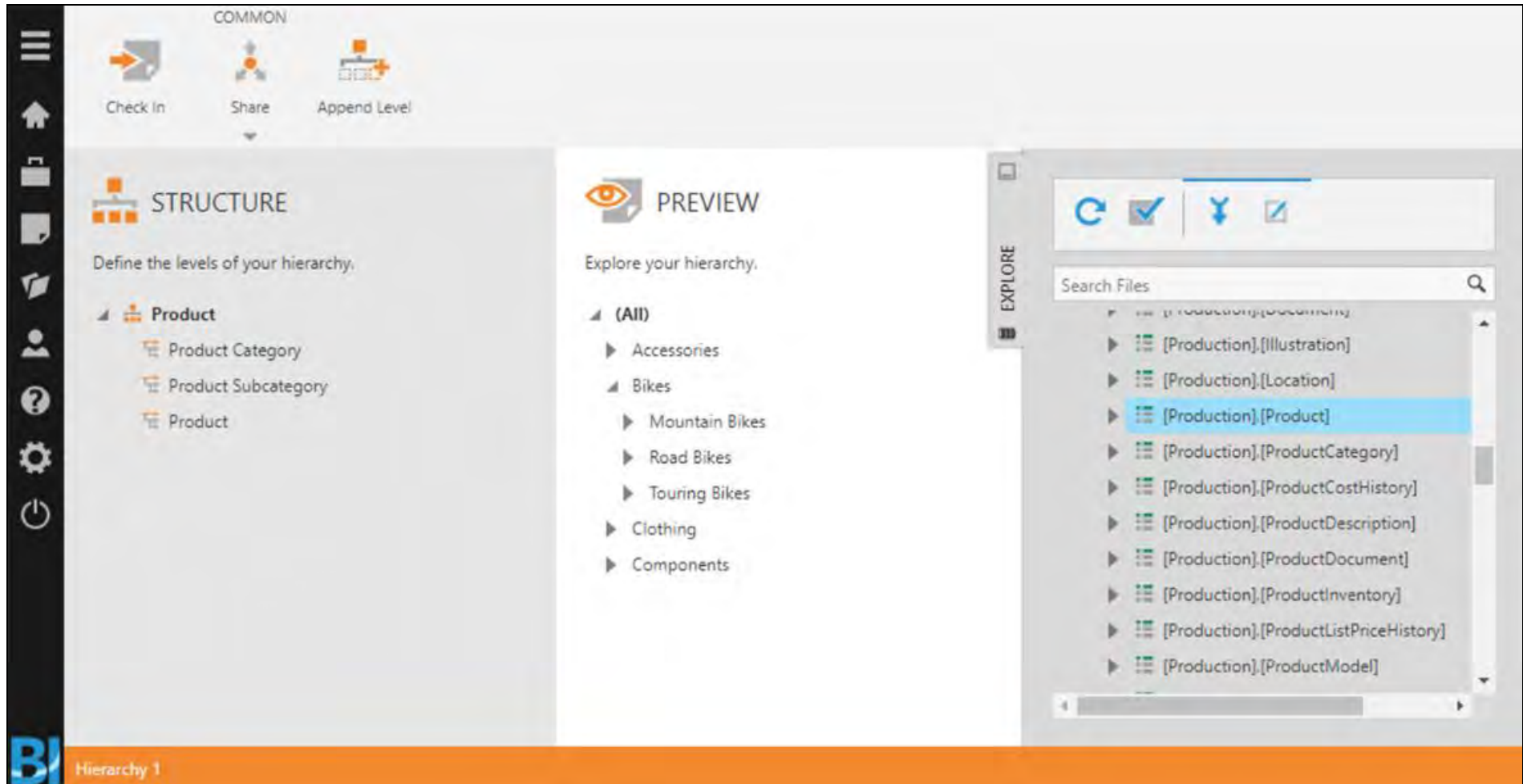


Note: You can also drag a data cube or one of its elements to the canvas.



An automatically defined hierarchy *Structure* is displayed with a *Preview* to the right, allowing you to explore the resulting hierarchy as a set of members arranged in a tree view.

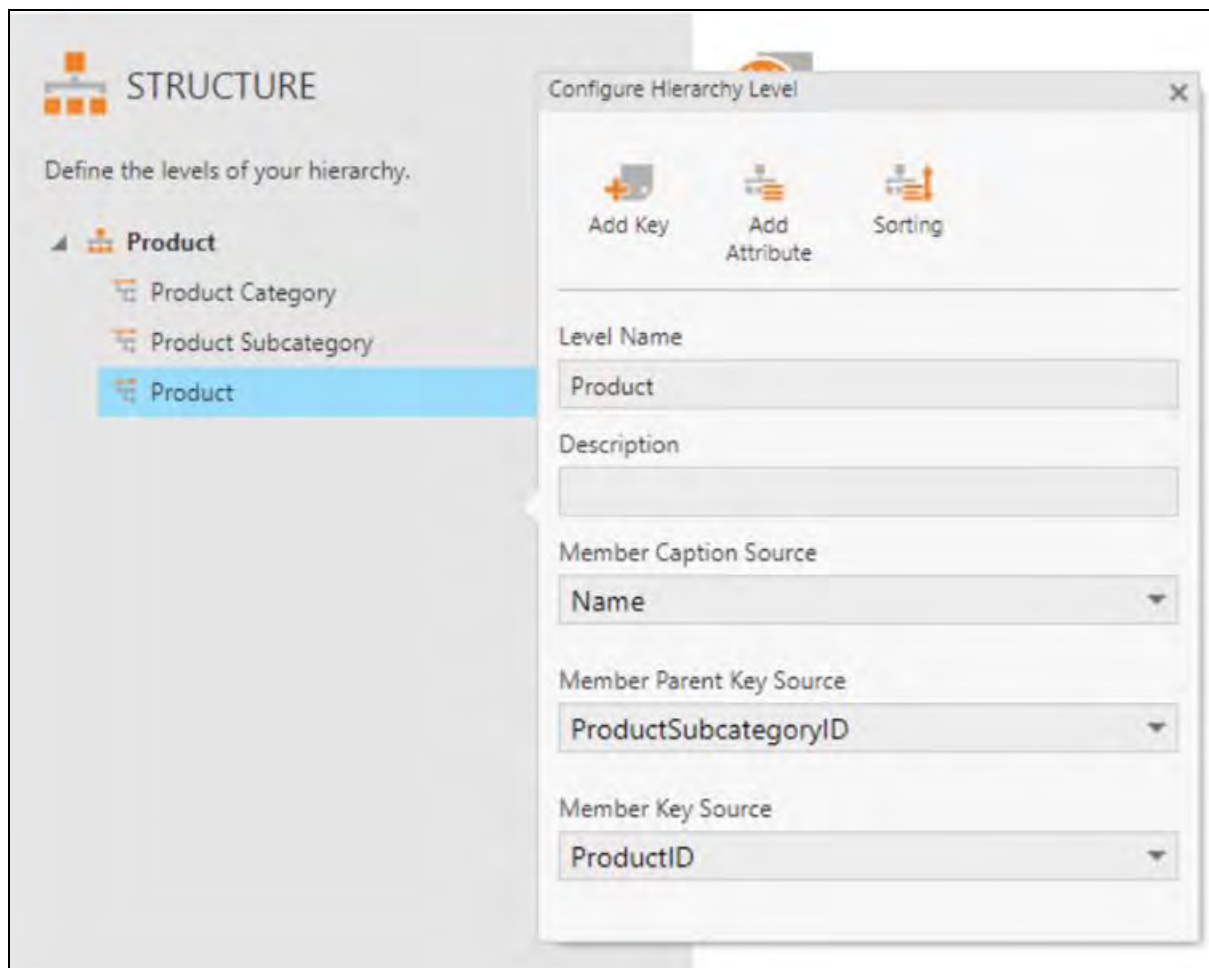
The name of your new hierarchy is displayed in the status bar at the bottom, where you can double-click to [rename it](#).



Note: If you want to define the levels yourself and don't want multiple levels created automatically, hold the **Shift** key while dropping your first data source. This way only a single level will be created and you can append or insert more.

In this example, three levels of a Product hierarchy have been automatically generated based on existing relationships between the Product table and other tables.

Under *Structure*, you can select a level if you want to customize its options in the *Configure Hierarchy Level* popup, including its [keys and display captions](#) and [default sorting](#). After selecting a level, you can click to add a new one from the toolbar either above or below it.



Note: Click **Check Warnings** in the toolbar to check for problems such as duplicate key values within a level that could cause an error or other problems if not avoided.

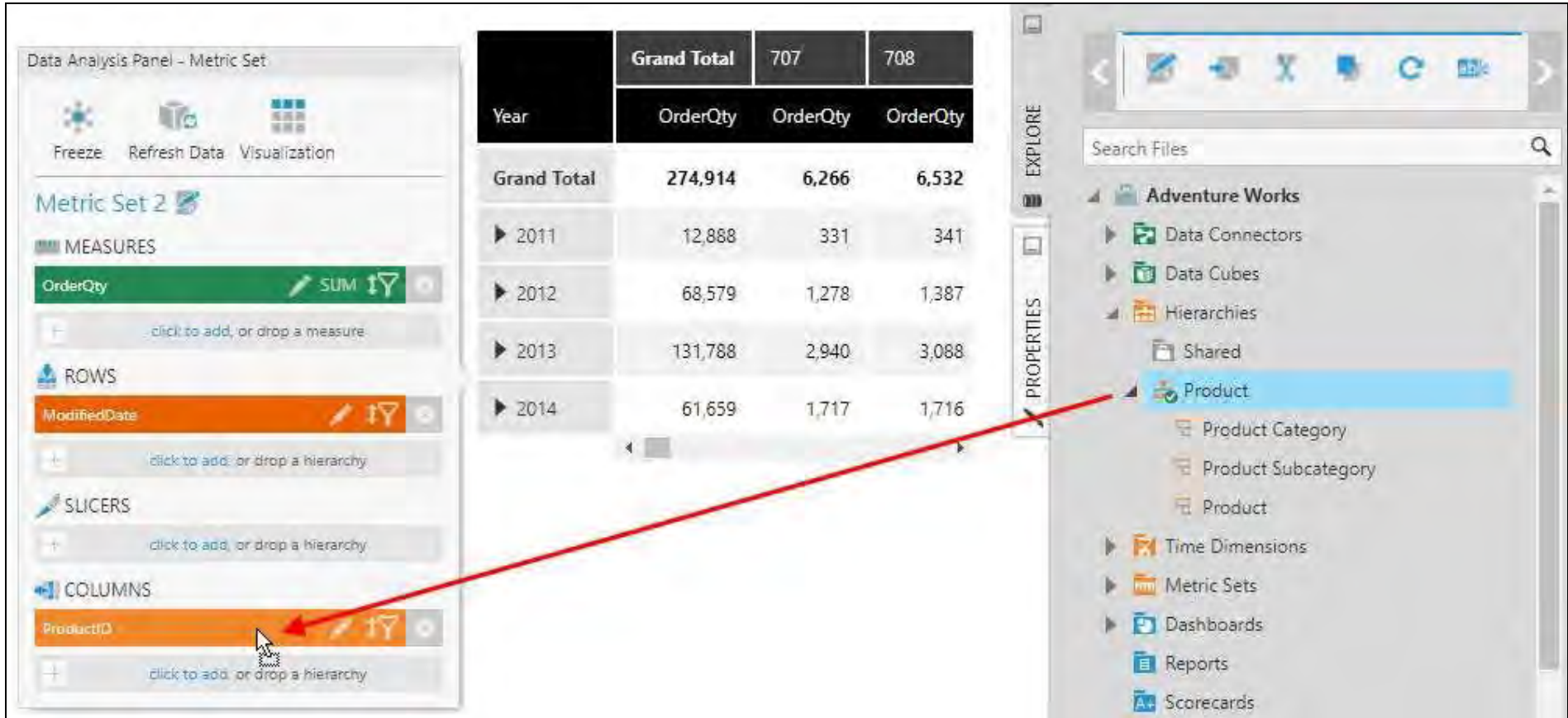
In this example, *ProductID* was automatically set as the **Member Key Source** of the Product level, which is used to uniquely identify each product member. These key values match our original *ProductID* column's values, so we will be able to use these together.

Replace ProductID with Product Hierarchy

Next, go back to the metric set from [Automatic Joins](#).

Instead of replacing the product IDs with a simple *Name* column, we will replace it with our new hierarchy: remove *Name* from under **Columns** and re-add *ProductID* in the Data Analysis Panel.

From the **Explore** window, drag the hierarchy you created onto *ProductID* to replace it.



The screenshot shows the Logi Analytics interface. On the left is the 'Data Analysis Panel - Metric Set' with sections for Measures (OrderQty), Rows (ModifiedDate), Slicers, and Columns (ProductID). In the center is a table visualization with columns: Year, Grand Total, OrderQty, OrderQty, and OrderQty. The table data is as follows:

Year	Grand Total	OrderQty	OrderQty	OrderQty
Grand Total	274,914	6,266	6,532	
2011	12,888	331	341	
2012	68,579	1,278	1,387	
2013	131,788	2,940	3,088	
2014	61,659	1,717	1,716	

On the right is the 'EXPLORE' window showing a file tree. The 'Product' hierarchy is selected and highlighted in blue. A red arrow points from this 'Product' hierarchy to the 'ProductID' field in the Columns section of the Data Analysis Panel.



Note: Hierarchy levels that use multiple key values to uniquely identify their members need multiple columns that correspond with those keys. You can use hierarchies like these when editing a data cube and its [Process Result](#).

The Columns axis of the table visualization now shows product categories, which you can expand to see product subcategories and expand again to see products. Since the *Name* column was indicated as the *Member Caption Source* for the hierarchy, the captions are taken from the *Name* column of each hierarchy level.

Year	Grand Total	▶ Accessories	▲ Bikes	▲ Mountain Bikes	
	OrderQty	OrderQty	OrderQty	OrderQty	OrderQty
Grand Total	274,914	61,932	90,268	28,321	682
▶ 2011	12,888	1,032	7,963	2,621	356
▶ 2012	68,579	5,750	28,494	9,034	326
▶ 2013	131,788	32,153	37,748	11,734	
▶ 2014	61,659	22,997	16,063	4,932	

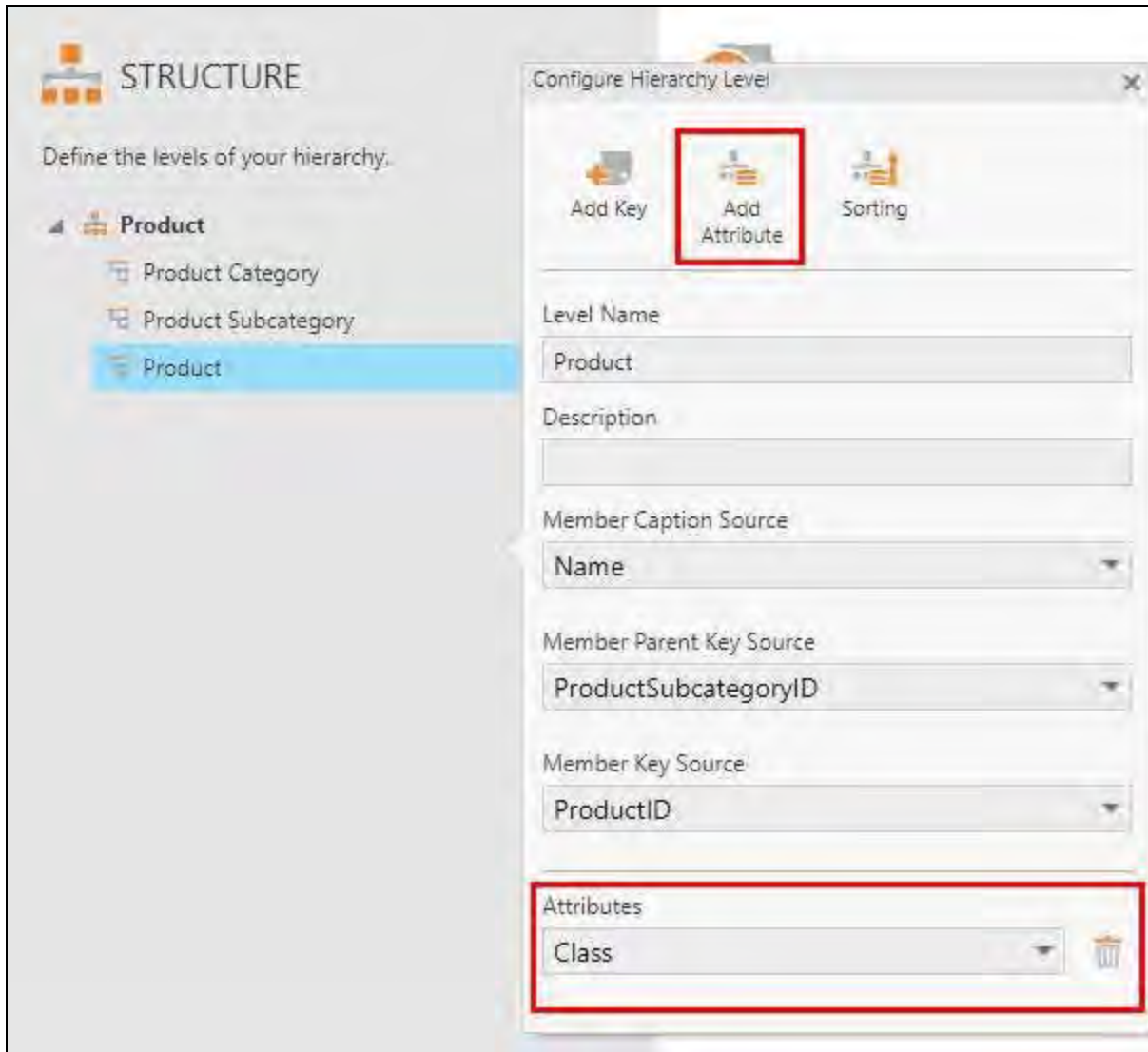
In this case the hierarchy's lowest level *Product* was set up with keys that matched our *ProductID* column. If you are working with a column of values that matches the keys from a different level of the hierarchy than the bottom one, expand the hierarchy in the Explore window and drag that specific level instead on top of your existing column to replace it with the hierarchy from that level up (e.g., drag the *Product Subcategory* level in our example above on top of a column containing subcategory IDs).

Attributes

Attributes are optional additional details from your hierarchy level's data source that automatically become available to add to the metric set once the hierarchy is used. Attributes can describe the members of a hierarchy level, like the color or size of each product in the *Product* level of the hierarchy.

Add Attributes

To add attributes to our *Product* hierarchy, re-open it from the main menu. For this example, click to select the *Product* level to open its details.



Click **Add Attribute** in the Configure Hierarchy Level popup for each attribute that you want to add, and under **Attributes**, choose one of the columns from this level's data source. From our example data source, you can choose *Color*, *Class*, *Size*, *Style*, and others.



Note: When editing a data cube, another way to add attributes is in the cube's [process result](#).

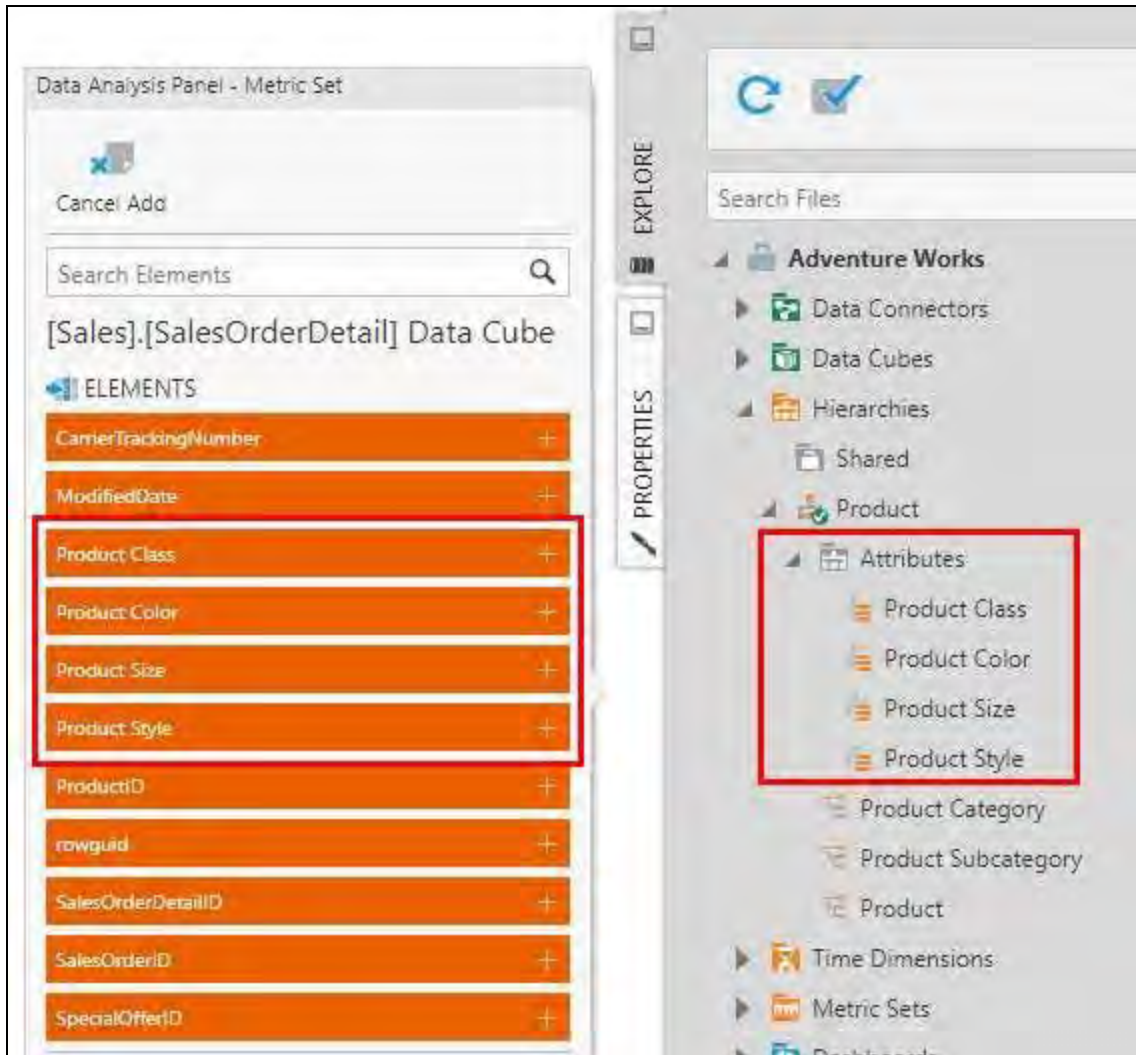
Use Attributes

Navigate back to the metric set to use the attributes in a visualization.

To have a better look at our products, we can click the x button next to *ModifiedDate* in the Data Analysis Panel to remove it, then drag *ProductID* from **Columns** to **Rows** in the Data Analysis Panel.



Next, we can **click to add** an attribute of the Product hierarchy.



Select an attribute from the list. Alternatively, you can find these attributes in the *Attributes* folder under the hierarchy in the **Explore** window, which you can then drag to add.

Since we added attributes to the *Product* level in our example, click to expand or right-click to change the level down to *Product* to see these product details we added as attributes.

Product Category	Product Subcategory	Product	Product Color	OrderQty
Grand Total				274,914
▶ Accessories			N/A	61,932
▲ Bikes	▲ Mountain Bikes	Mountain-100 Black, 38	Black	682
		Mountain-100 Black, 42	Black	634
		Mountain-100 Black, 44	Black	678
		Mountain-100 Black, 48	Black	616
		Mountain-100 Silver, 38	Silver	642
		Mountain-100 Silver, 42	Silver	593

These attributes are known to describe the values of the hierarchy, e.g., each *Product* can have only one *Color*. Additional totals and expander buttons are unnecessary and will be left out automatically for the attribute when selected after your hierarchy.

If you choose to, you can also place an attribute before its hierarchy, or on its own. In this case, totals and expanding/collapsing work as they normally do for other hierarchies. The figure below shows the result of dragging to re-order Product Color first and displaying the *Product* level of the hierarchy afterward.

Product Color	Product	OrderQty
Grand Total		274,914
Black	(All)	81,937
	Full-Finger Gloves, L	3,378
	Full-Finger Gloves, M	2,206
	Full-Finger Gloves, S	500
	Half-Finger Gloves, L	1,276
	Half-Finger Gloves, M	3,464

For more information, see:

- Video: [Creating Custom Categorical Hierarchies](#)
- [Symphony Metric Sets](#)
- [Add a Relationship Between Tables](#)
- [Create a Ragged Hierarchy](#)
- [How To Create a Custom User Hierarchy](#)
- [Process Result](#)



Create a Metric Set and Add a Filter

This applies to: *Managed Dashboards, Managed Reports*

This walkthrough shows you how to create [metric sets](#), add them to a dashboard, and connect filters.

Specifically, you will learn how to:

- Create metric sets to analyze data.
- Change the visualization.
- Combine the metric sets on a dashboard.
- Connect a filter to both visualizations.

This walkthrough uses a data cube like the one from [used here](#), but can be followed with any data source with similar data.



Note: You can also create metric sets directly in a dashboard or other view, and these are saved as part of the view rather than in the *Metric Sets* folder. See [View data with a chart and drill down](#) for an example.

Related video: [Introduction to Metric Set Designer](#)

Create a New Metric Set

From the main menu, click **Business**, and then click **Create**.

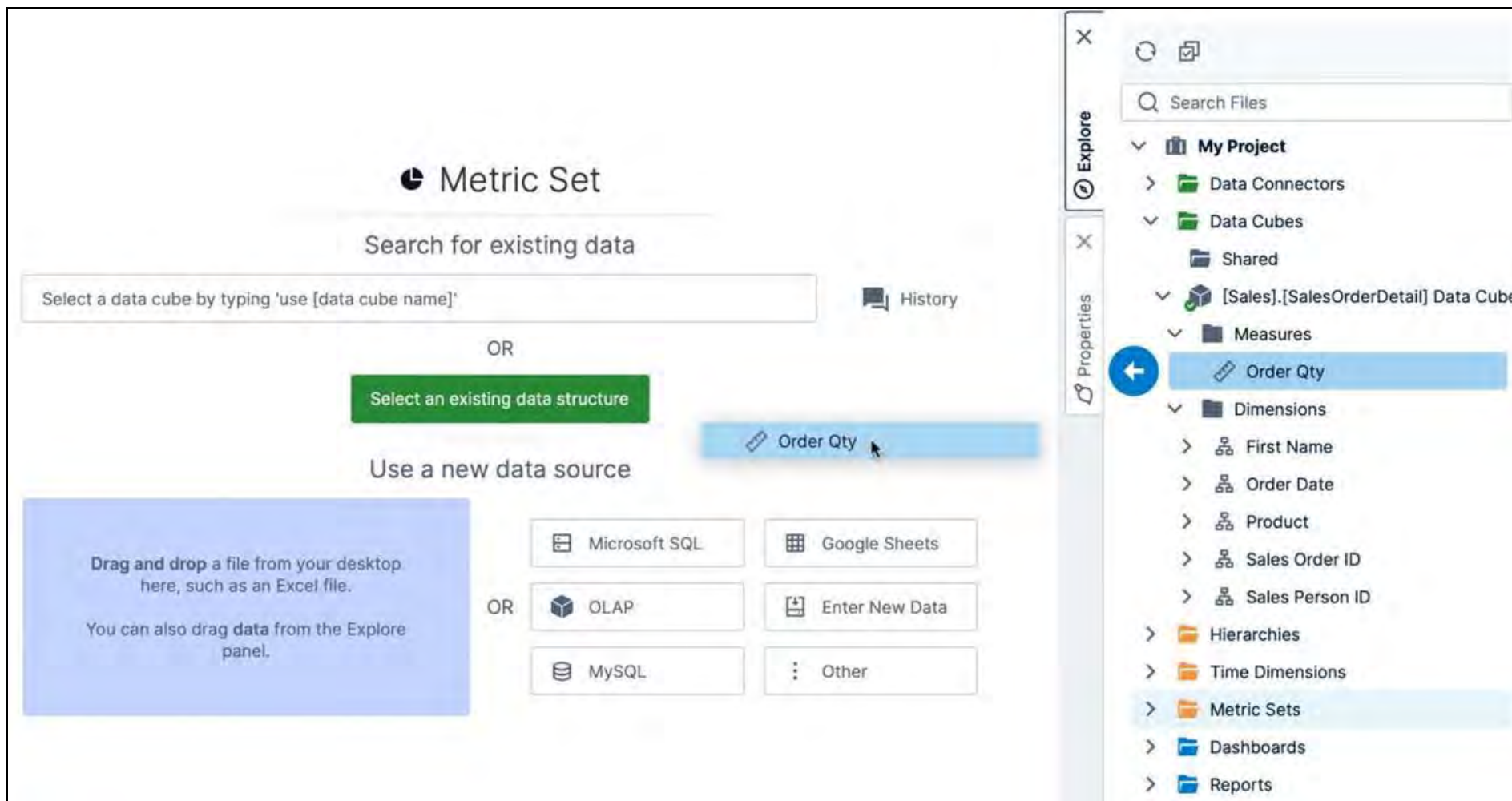
The full-screen metric set editor is displayed.

Set Up the Metric Set Based on a Data Cube

There are multiple options available from this screen for choosing your data source such as a data cube or data connector.

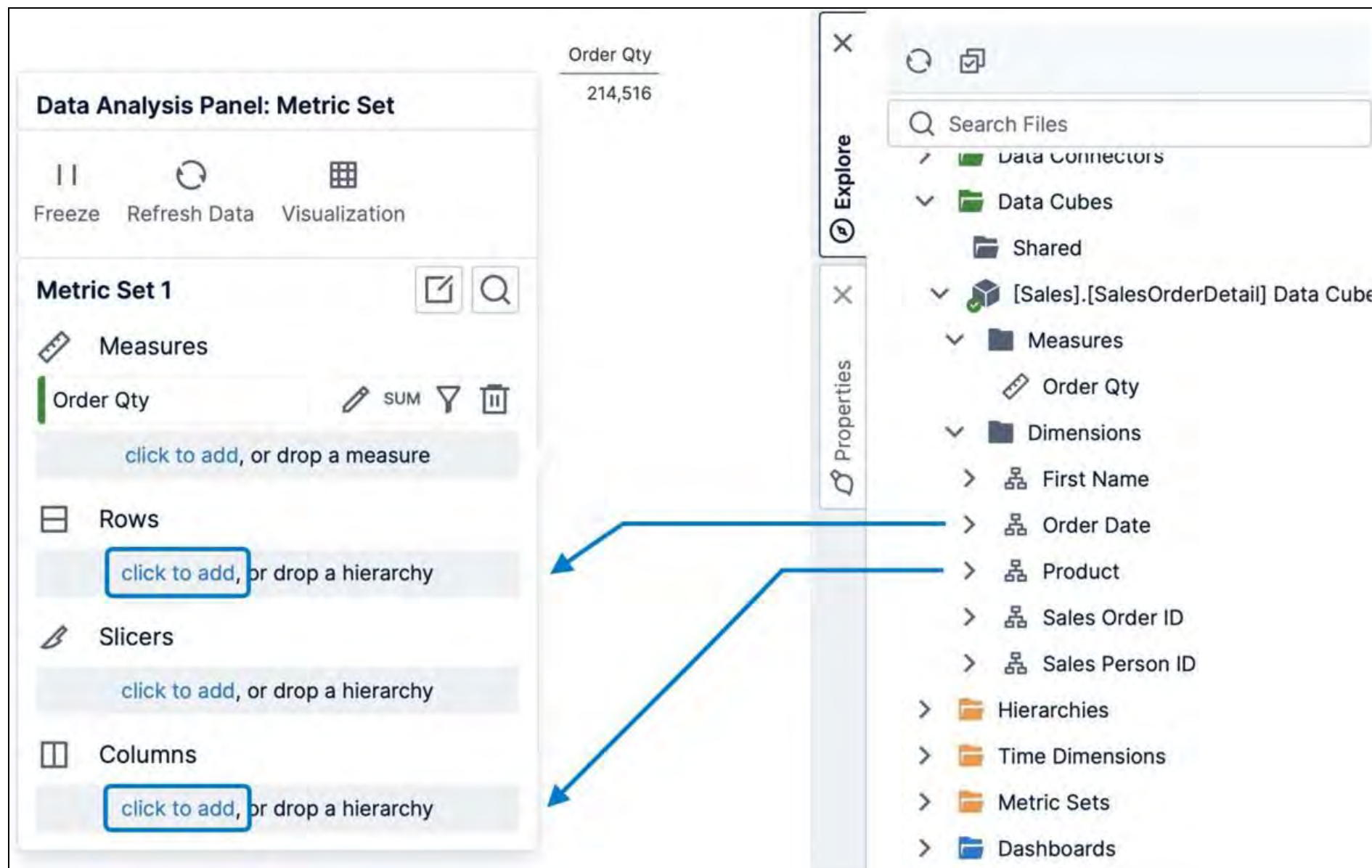
In this example, we located a data cube in the **Explore** window and clicked to expand it and its **Measures** subfolder, then dragged the *Order Qty* measure to the canvas.

You can also click to **Select an existing data structure**, then find and select the same measure in the dialog that opens.



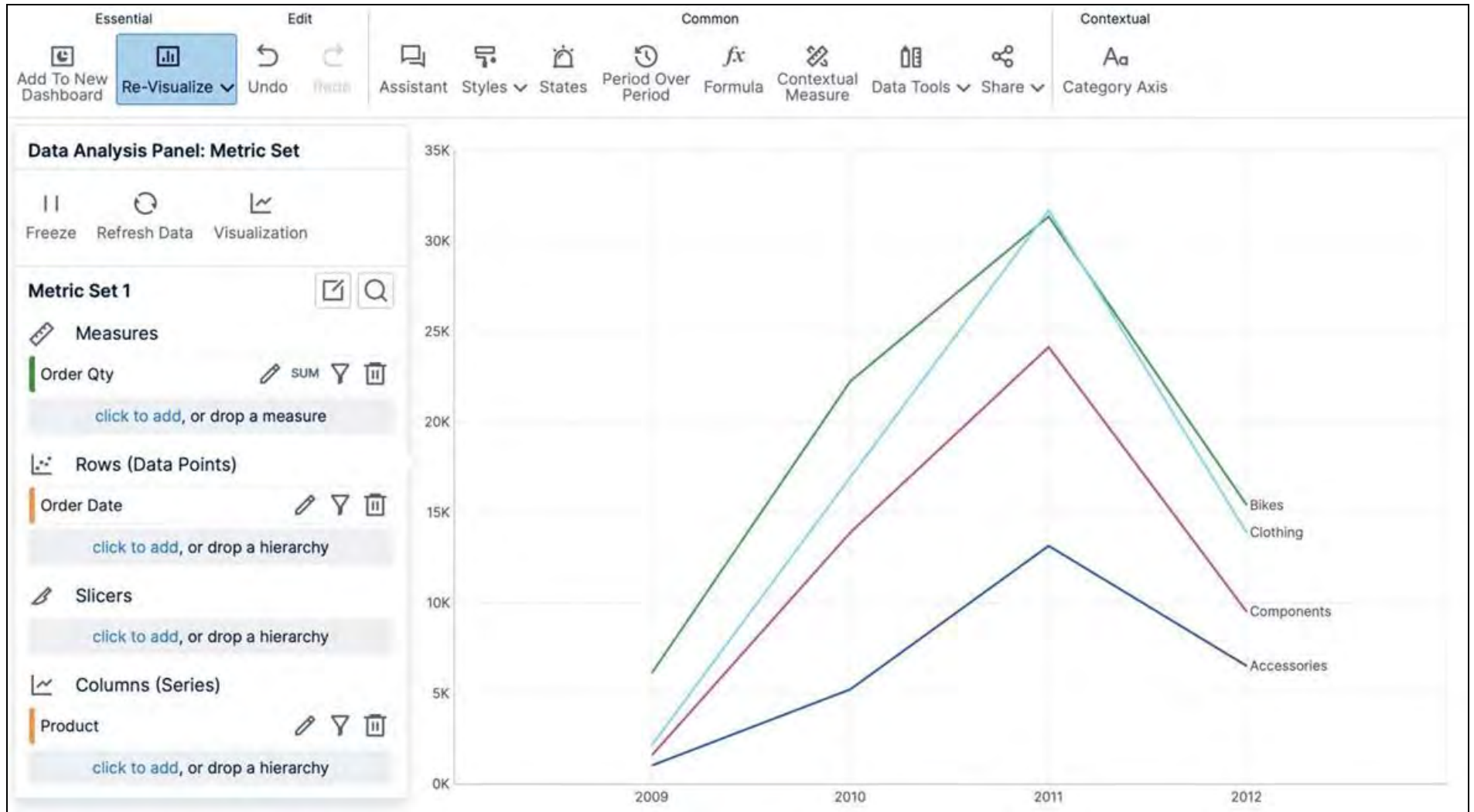
By default, the first data you choose is displayed in a table visualization beside a *Data Analysis Panel*. There is a single aggregated total value because no hierarchies have been added to group it.

In the **Explore** window, expand the data cube's *Dimensions* folder, then drag *Order Date* to **Rows** and *Product* to **Columns** in the Data Analysis Panel. Alternatively, you can **click to add** them in the Data Analysis Panel, choosing them from a list of data from your existing data source.



The screenshot displays the Logi Symphony v24 interface. On the left, the **Data Analysis Panel: Metric Set** is visible, featuring controls for **Freeze**, **Refresh Data**, and **Visualization**. Below these, the **Metric Set 1** section includes **Measures** (with **Order Qty** selected and a **SUM** aggregation), **Rows**, **Slicers**, and **Columns**. Each of these sections has a **click to add, or drop a hierarchy** button. On the right, the **Explore** sidebar shows a hierarchical tree of data sources, including **Data Connectors**, **Data Cubes**, and a specific **[Sales].[SalesOrderDetail] Data Cube**. This cube is expanded to show **Measures** (including **Order Qty**) and **Dimensions** (including **First Name**, **Order Date**, **Product**, **Sales Order ID**, and **Sales Person ID**). A **Properties** panel is also visible at the bottom of the sidebar. Two blue arrows point from the **Order Qty** measure in the **Measures** list of the **Explore** sidebar to the **click to add, or drop a hierarchy** buttons in the **Rows** and **Columns** sections of the **Data Analysis Panel**.

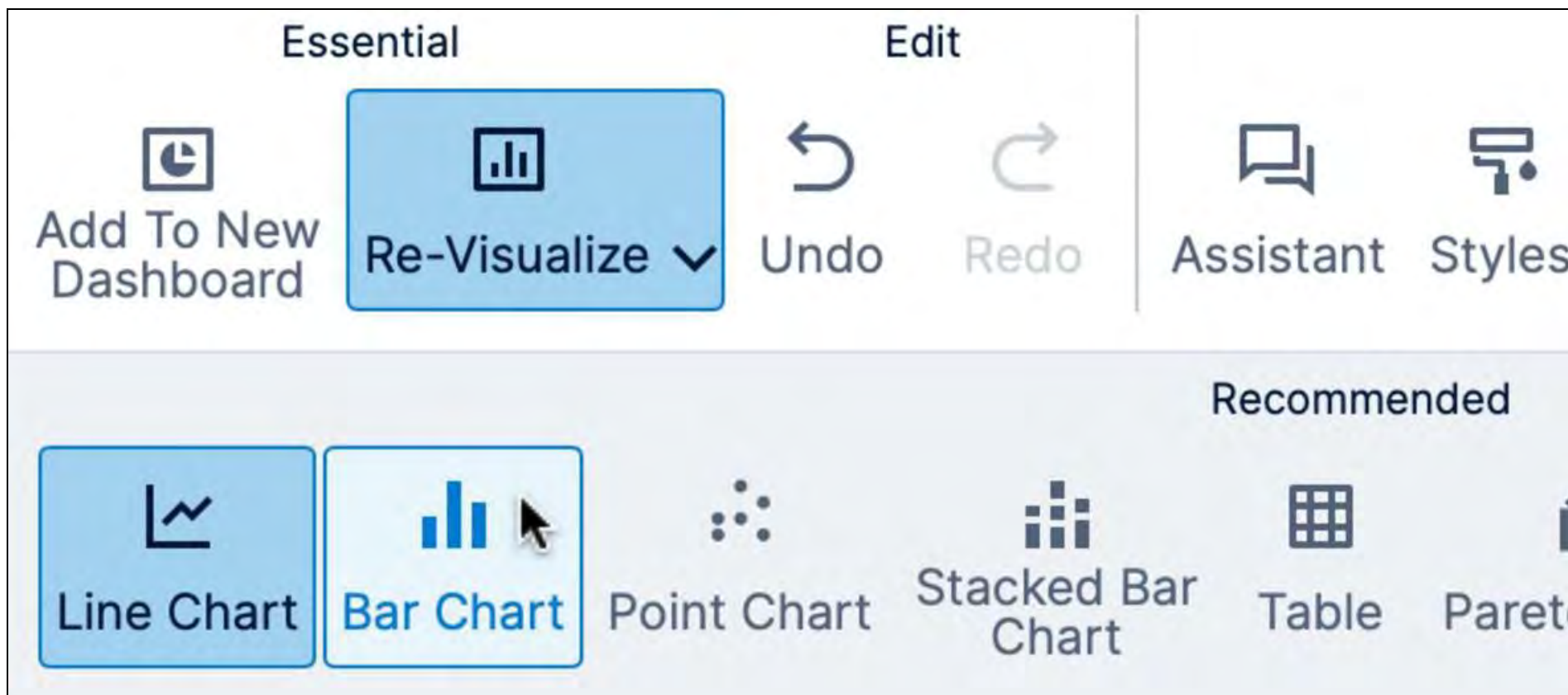
If you haven't chosen or customized a visualization yet, the table visualization is automatically re-visualized to a line chart because this is usually the best choice for visualizing changes to measure values overtime.



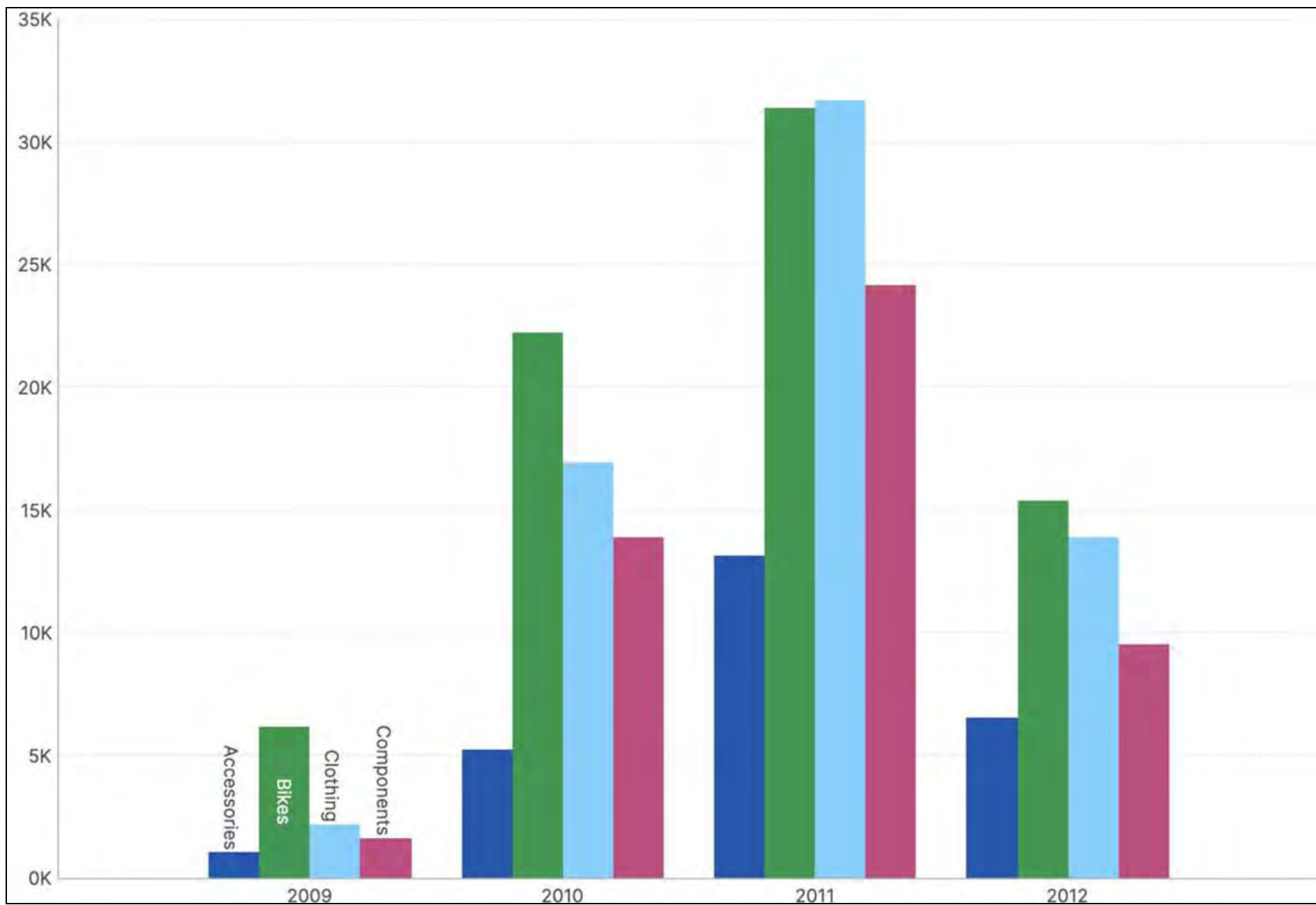
At any time, you can click **Re-Visualize** in the toolbar and make your own selection: it will not be re-visualized automatically after you have chosen or customized a visualization yourself.

Complete the Metric Set

To complete the setup of this metric set, click **Re-Visualize** in the toolbar, and then click **Bar Chart** (or **Bar** under *Change Chart Type*).



The visualization is now changed to a bar chart where each series (i.e., bars of the same color) corresponds to a specific product category.



If you want to allow others to use this metric set, click **Share** in the toolbar and [check it in](#).

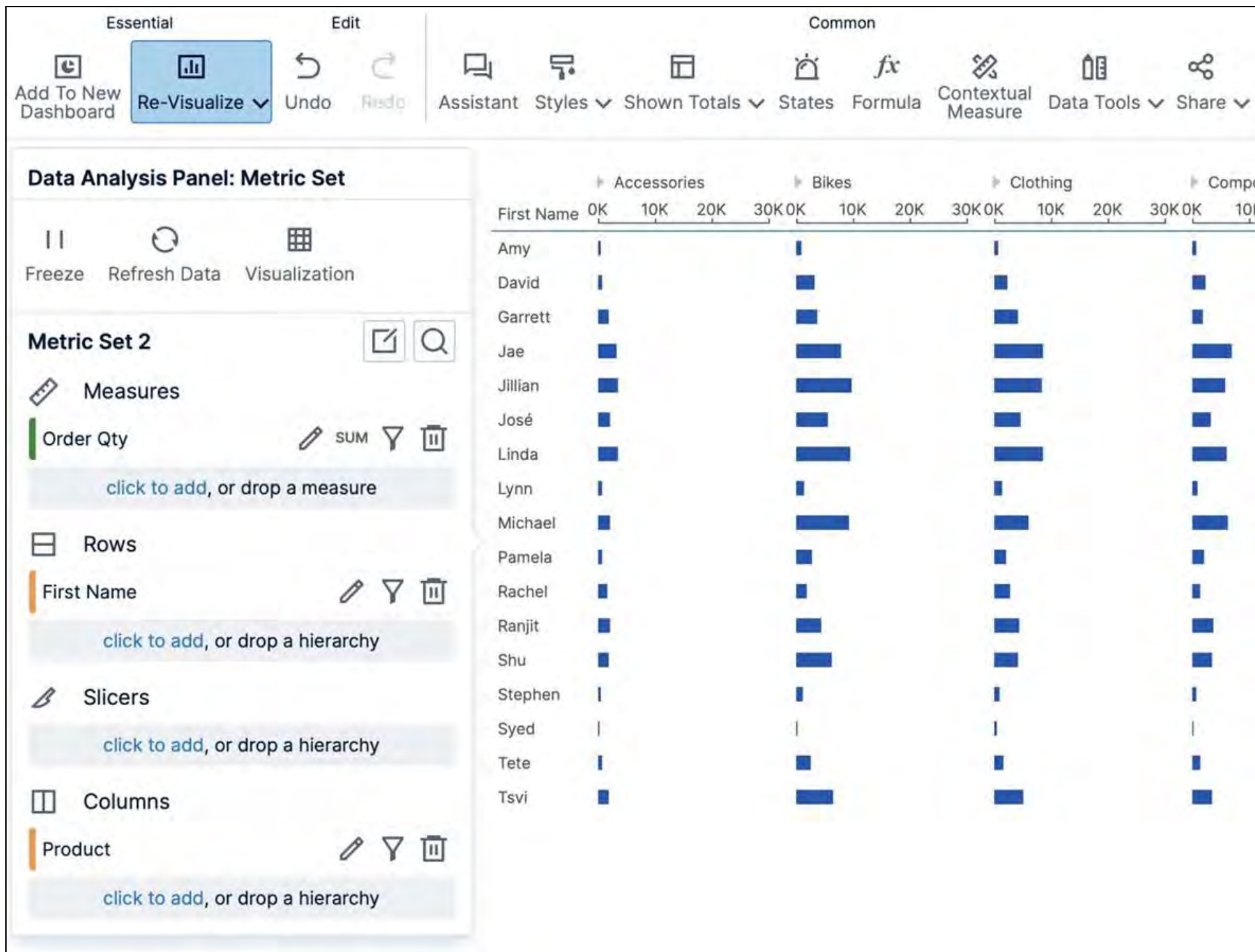


Add a Second Metric Set

Using similar steps as before, create a second metric set as follows:

- Expand the data cube's *Measures* folder and drag the *Order Qty* measure to the canvas.
- Expand the data cube's *Dimensions* folder and drag the *First Name* hierarchy to **Rows** in the Data Analysis Panel.
- Drag the *Product* hierarchy to **Columns** in the Data Analysis Panel.

A data bar table may be chosen as the visualization automatically, showing person names along the rows axis, and expandable product categories along the columns axis.



Data Analysis Panel: Metric Set

Freeze

Refresh Data

Visualization

Metric Set 2

Measures

Order Qty

click to add, or drop a measure

Rows

First Name

click to add, or drop a hierarchy

Slicers

click to add, or drop a hierarchy

Columns

Product

click to add, or drop a hierarchy

Accessories

Bikes

Clothing

Compo

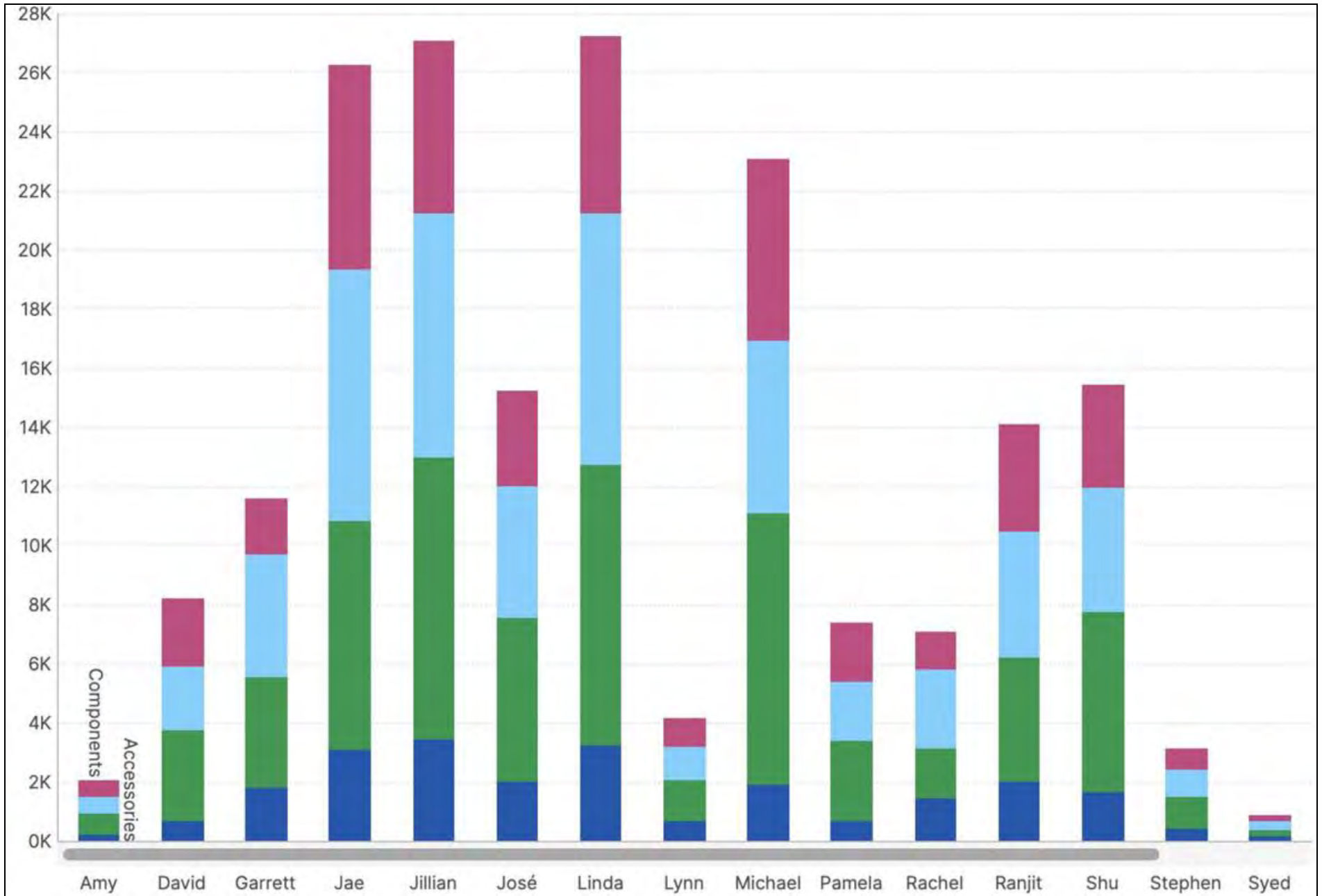
First Name	OK	10K	20K	30K	OK	10K	20K	30K	OK	10K	20K	30K	OK	10K
Amy														
David														
Garrett														
Jae														
Jillian														
José														
Linda														
Lynn														
Michael														
Pamela														
Rachel														
Ranjit														
Shu														
Stephen														
Syed														
Tete														
Tsvi														



- Archive of documentation for Logi Symphony v24

Click **Re-Visualize** in the toolbar, and then click **Stacked Bar** under *Charts*.

The table visualization is changed to a stacked bar chart showing person names along the bottom axis and measure values on the left axis. Each color in the stacked bars corresponds to a specific product category.



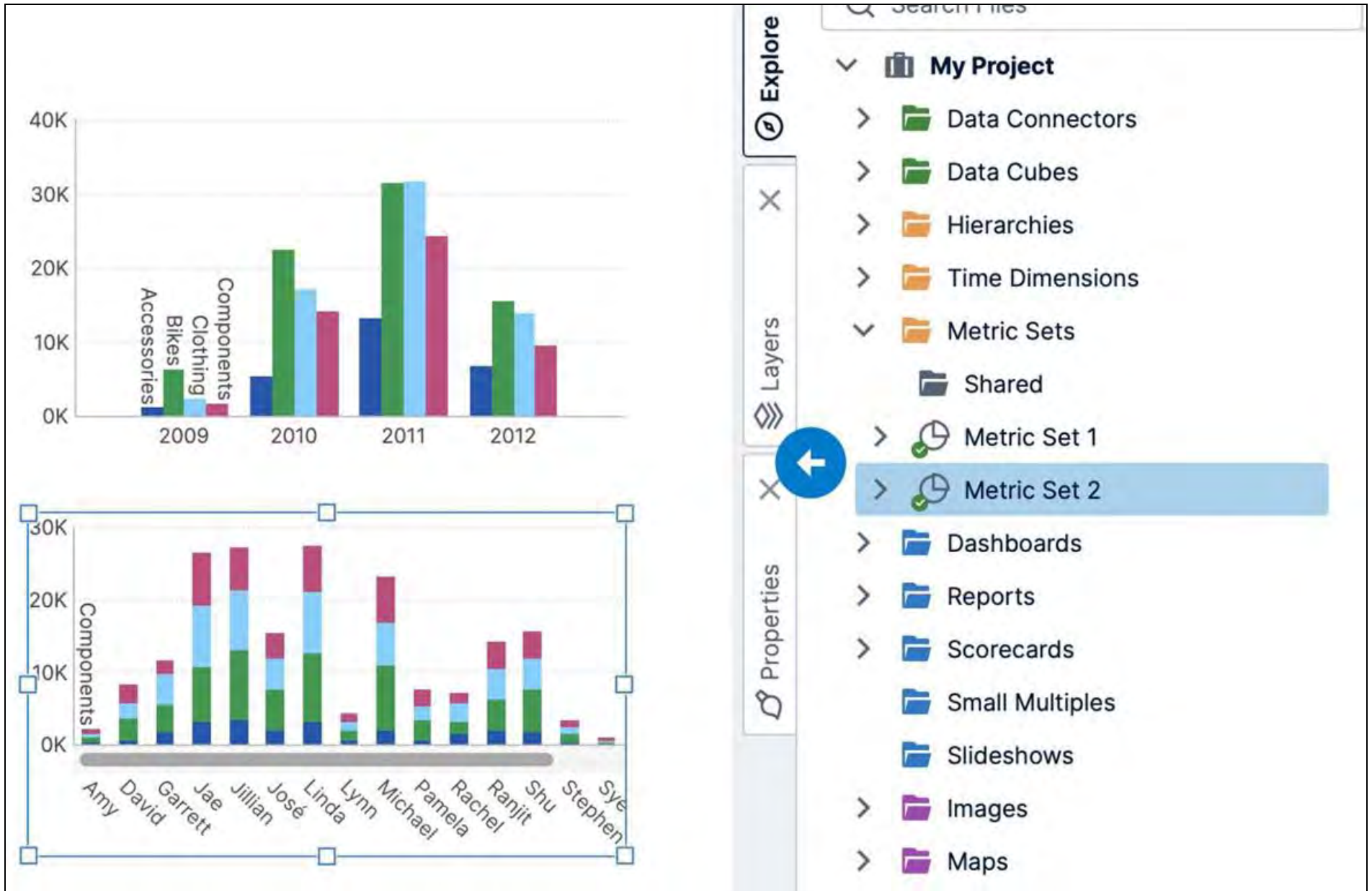
Click **Share** in the toolbar and check in this metric set too if you want to.



Display the Metric Sets on a Dashboard

To automatically create a new dashboard and put the current metric set on it, click **Add To New Dashboard** in the toolbar of a metric set. Otherwise, click [main menu](#) and then click **Create** to create a dashboard.

Go to the **Explore** window, expand the *Metric Sets* folder, and drag the metric sets from earlier or the remaining one onto the dashboard canvas. Observe that they are visualized using the same options you chose.



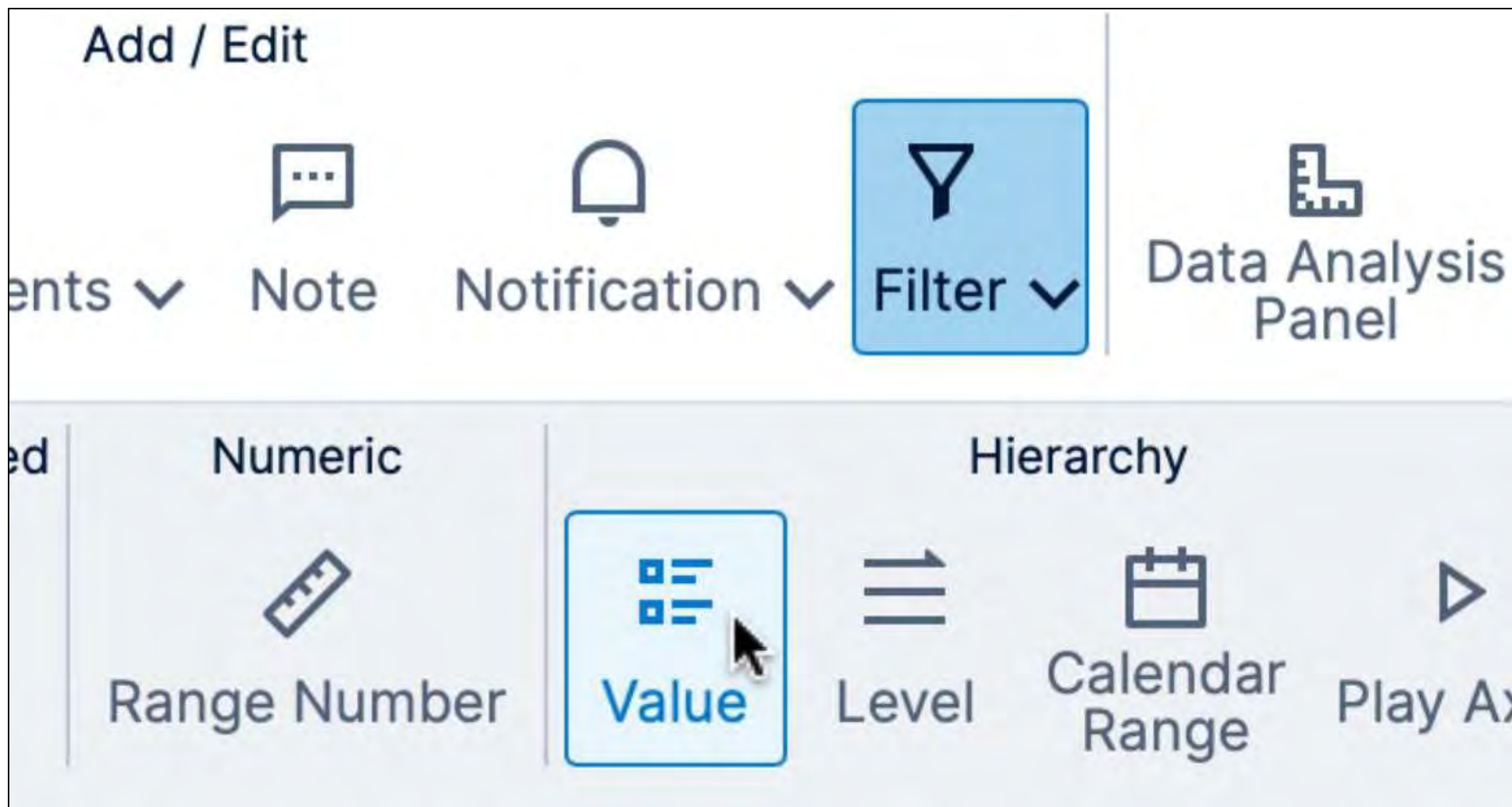


Note: A single metric set can be visualized multiple times and in different ways. The visualizations above are now independent from the one set up in the metric set editor, and you can choose to customize or re-visualize the ones just on this dashboard.

Using Filters on a Dashboard

With the metric sets added to a dashboard, you can add dedicated [filters](#) that can be used to more easily filter the data while viewing. For example, you can add one filter used to choose the products displayed in both charts at once.

1. If one of the charts is selected, click an empty area of the canvas to de-select it if you don't want to filter only that chart (as in this example).
2. Click **Filter** in the toolbar, and then choose **Value** under Hierarchy.



3. In the filter connections popup that appears next to the new filter, scroll down if necessary and confirm that *Product* in both charts is selected for filtering.
4. You can choose the default selection for this filter by setting the **Default Value** as shown in the image below.

Filters: viewParameter 1

Default Value

Bikes, Components

Exclude From Export

☐

Search Parameters

Dashboard 1

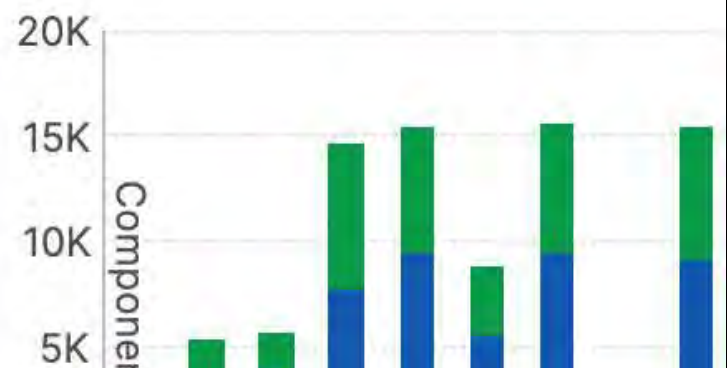
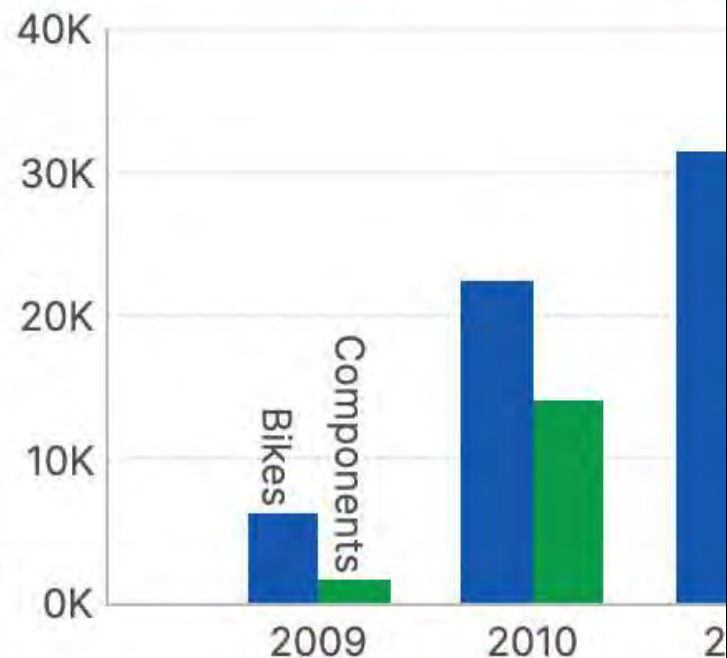
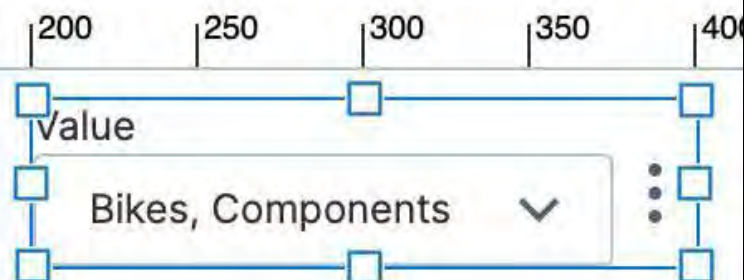
chart 1

Metric Set 1

Product

Order Date

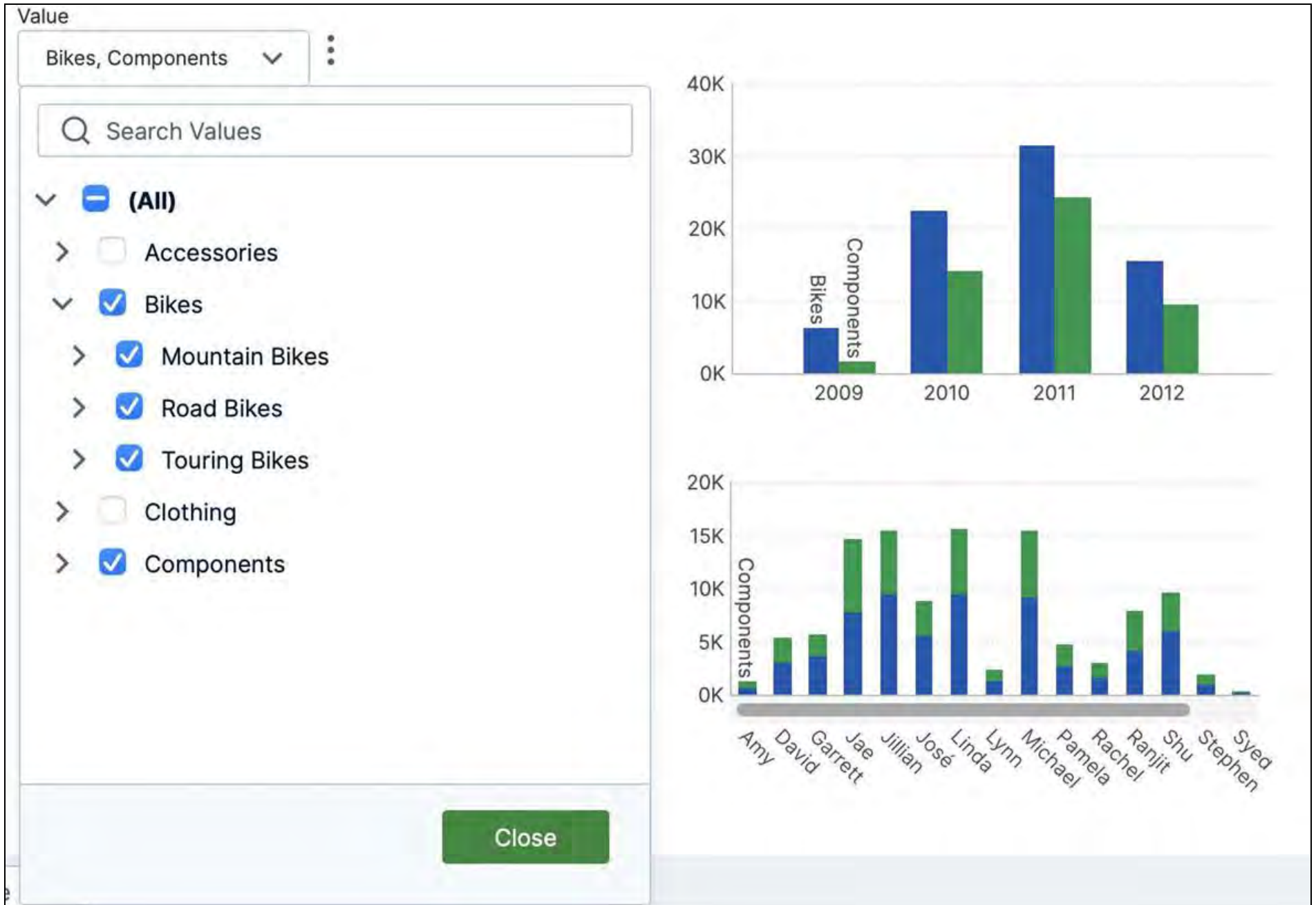
chart 2





- Archive of documentation for Logi Symphony v24

To use the filter directly, click **View** in the toolbar.





- Archive of documentation for Logi Symphony v24

For more information, see:

- Video: [Introduction to Metric Set Designer](#)
- Video: [Filters](#)
- [Metric set analysis tools](#)
- [Adding Filters](#)
- [Data Cubes](#)

Create a Ragged Hierarchy

This applies to: *Managed Dashboards, Managed Reports*

This article demonstrates how to create a ragged, unbalanced, or self-referencing hierarchy. A ragged hierarchy can skip levels, as shown in the image below.

Country	State	City	Value
US	New York		2,456.55
	Texas		1,779.42
	Washington		891.64

Ragged hierarchies that extend to different levels in each branch are also known as "unbalanced", as shown in the image below:

Country	State	City	Order Amount
United States	Florida		3,477.57
	Michigan	Detroit	1,650.04

The number of levels is determined by the data, and may change over time.

Data

One structure is used to define a ragged hierarchy. The top level is determined by the rows where the parent ID is empty, while the rest will be based on their parent ID and will appear in one of the levels below the parent.

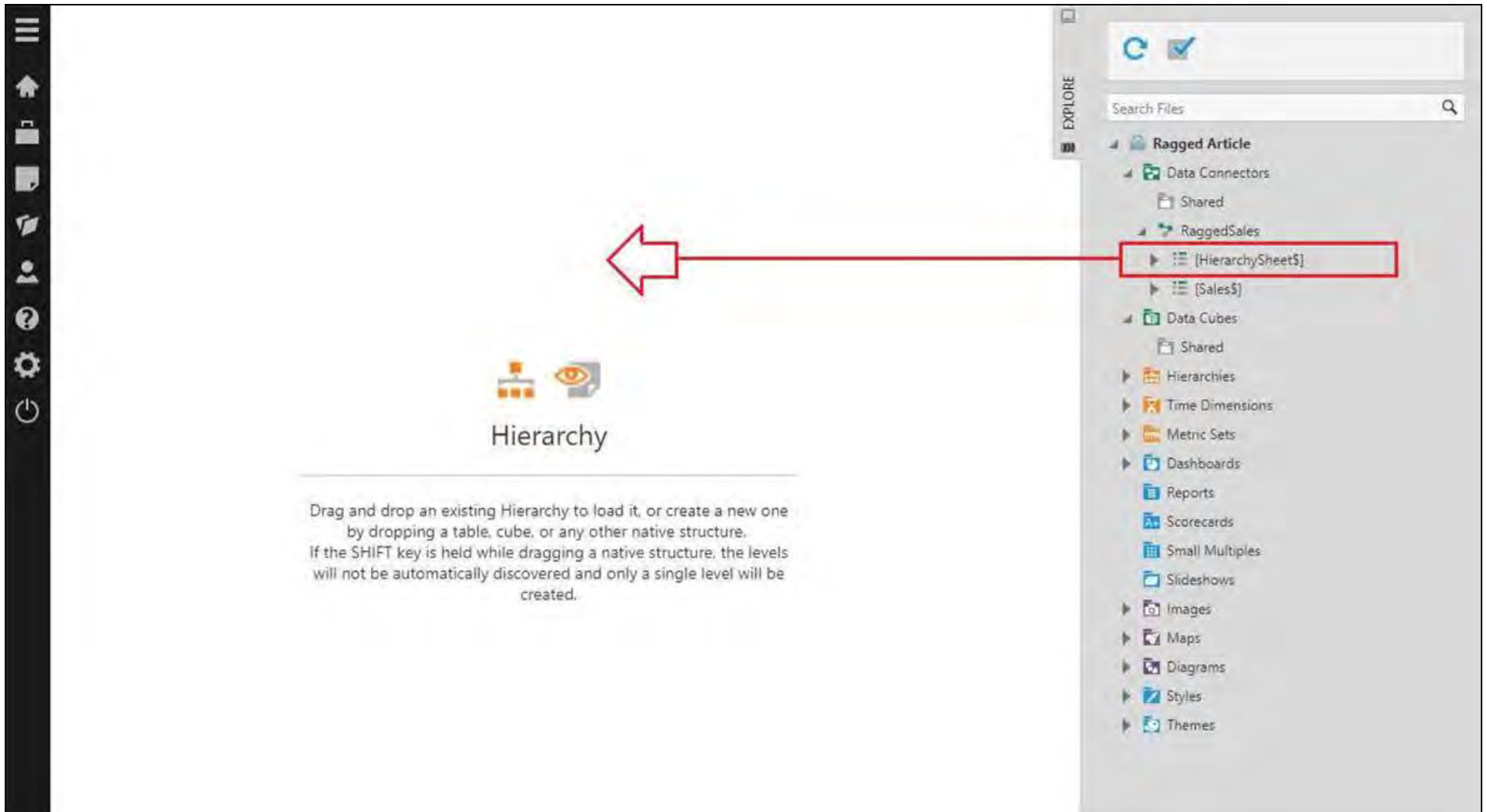
	A	B	C	D	E
1	Id	Name	HierarchyParentId	RegionType	Level
2	1	Global Sales		Region	1
3	2	North America	1	Continent	2
4	3	Europe	1	Continent	2
5	4	Canada	2	Country	3
6	5	United States	2	Country	3
7	6	Mexico	2	Country	3
8	7	Greece	3	Country	3
9	8	Ontario	4	StateProvince	4
10	9	Michigan	5	StateProvince	4
11	10	California	5	StateProvince	4
12	11	Toronto	8	City	5
13	12	Detroit	9	City	5
14	13	Los Angeles	10	City	5
15	14	Mexico City	6	City	5
16	15	Athens	7	City	5
17	16	Bill Bonds	11	SalesPerson	6
18	17	Adela Walls	12	SalesPerson	6
19	18	Tiya Keeling	13	SalesPerson	6
20	19	Gertrude Valenzuela	14	SalesPerson	6
21	20	Keiren Finnegan	15	SalesPerson	6
22	21	Sunny Esparza	11	SalesPerson	6
23	22	Online	1	Online	2
24	23	Website	22	Website	6

You may also have an optional column of values defining the level of each row, which is what makes it possible to "skip" a level, by placing a value two or more levels below its parent. In the example data above, Mexico City's parent is *Mexico* in the *Country* level, even though other cities have a parent value in the *StateProvince* level.

Steps

Create a new Hierarchy from the [main menu](#) by choosing **Business** and the **Hierarchy** file type, then clicking **Create** and choosing the **Ragged** option.

The hierarchy canvas will appear. Drag the data defining the hierarchy onto the canvas from the **Explore** window.



On the left side of the editor under *Definition*, set up the following:

- **Member Caption Source** - The values used displayed as the captions to users.
- **Member Parent Key Source** - The values identifying the parent member's key values.
- **Member Key Source** - The values that uniquely identify each member. The child member's parent keys must match these.

- **Level Source** (*Optional*) - Values that identify the level. This can be un-set, and the levels will be generated based on each member's parent keys. If set, the level will be determined by and named for these values, and it may be possible for a member's parent to be two or more levels above. Level values can be text (string) or numeric.

The figure below shows how our example source data maps to our ragged hierarchy definition.

	A	B	C	D	E
1	Id	Name	HierarchyParentId	RegionType	Level
2	1	Global Sales		Region	1
3	2	North America		1 Continent	2
4	3	Europe		1 Continent	2
5	4	Canada		2 Country	3
6	5	United States		2 Country	3
7	6	Mexico		2 Country	3
8	7	Greece		3 Country	3
9	8	Ontario		4 StateProvince	4
10	9	Michigan		5 StateProvince	4
11	10	California		5 StateProvince	4
12	11	Toronto		8 City	5
13	12	Detroit		9 City	5
14	13	Los Angeles		10 City	5
15	14	Mexico City		6 City	5
16	15	Athens		7 City	5
17	16	Bill Bonds		11 SalesPerson	6
18	17	Adela Walls		12 SalesPerson	6
19	18	Tiya Keeling		13 SalesPerson	6
20	19	Gertrude Valenzuela		14 SalesPerson	6
21	20	Keiren Finnegan		15 SalesPerson	6
22	21	Sunny Esparza		11 SalesPerson	6
23	22	Online		1 Online	2
24	23	Website		22 Website	6

DEFINITION

Define how the hierarchy is created from the underlying data, including which columns are used for the caption and keys and if multiple keys are used.

Add Key

Member Caption Source

Name

Member Parent Key Source

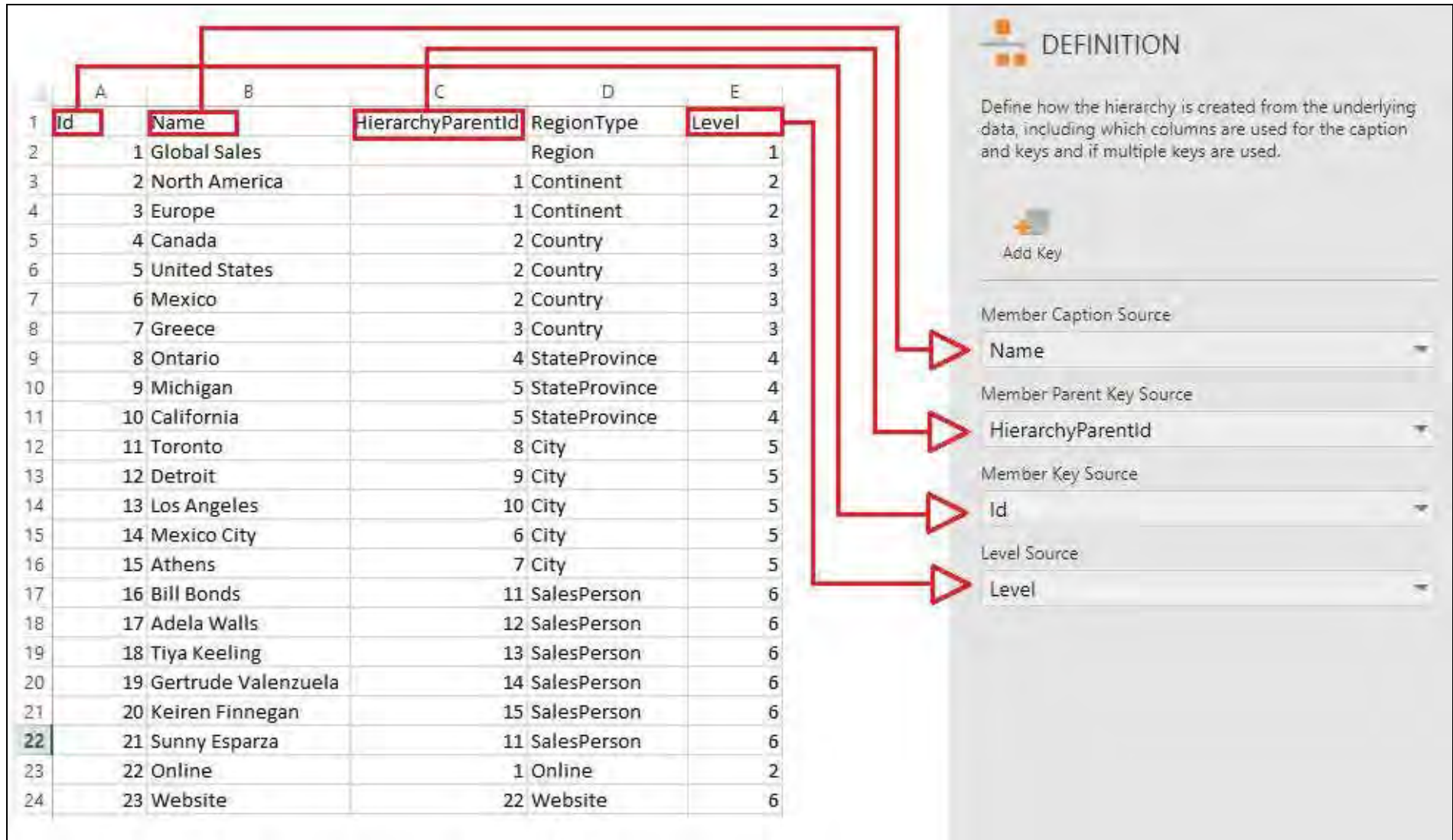
HierarchyParentId

Member Key Source

Id

Level Source

Level



The following image shows two versions of a hierarchy defined with our sample data and used in a table visualization. The first result sets the optional Level Source, while the second leaves it un-set.

Level Source

Level

Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Global Sales	Europe	Greece		Athens	Keiren Finnegan
	North America	Canada	Ontario	Toronto	Bill Bonds
					Sunny Esparza
		Mexico		Mexico City	Gertrude Valenzuela
		United States	California	Los Angeles	Tiya Keeling
		Michigan	Detroit	Adela Walls	
Online	Website				

Level Source

(not set)

Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Global Sales	Europe	Greece	Athens	Keiren Finnegan	
	North America	Canada	Ontario	Toronto	Bill Bonds
					Sunny Esparza
		Mexico	Mexico City	Gertrude Valenzuela	
		United States	California	Los Angeles	Tiya Keeling
		Michigan	Detroit	Adela Walls	
Online	Website				

You can review the hierarchy in the preview section.

Click **Check Warnings** in the toolbar to check for issues such as circular references that could cause an error or other problem later if not avoided.

Using the Ragged Hierarchy

You can use a ragged hierarchy the same way as for a standard hierarchy, replacing a column that can be matched with your hierarchy's keys. See [Automatic Joins and Hierarchies](#) for details on replacing a column with a hierarchy when editing a metric set, dashboard, or other view. You can also promote a column to your hierarchy in the [Process Result](#) of a data cube.



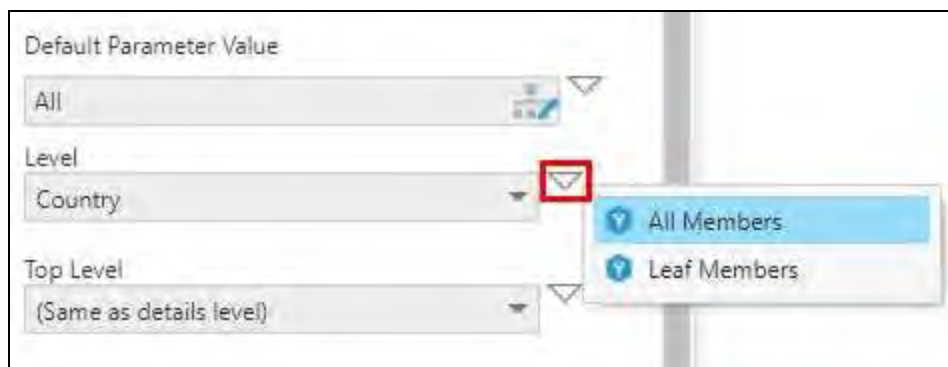
Note: When replacing key values with your ragged hierarchy, the key values must match only "leaf" members, which are members with no children.

The following sections demonstrate how you can visualize and work with a ragged hierarchy selected in your metric set.

Visualizing Levels

There are special options available when deciding what level(s) to display from a ragged hierarchy in addition to the usual options. To access these, click the hierarchy in the *Data Analysis Panel* to edit it, and click the token menu to the right of the level drop-down:

- *All Members* - This option selects all members from all levels at once, applicable for visualizations that can display multiple levels. This can be useful since the number of levels might change over time according to the data.
- *Leaf Members* - This option displays only the members with no children. This is similar to selecting the lowest level, except that for a ragged hierarchy, these members might be in different levels.



You can also add a [Level filter](#) to your dashboard or other view and access these options from its token menu.

Tables

The table visualization has a layout property called **Row Header Layout** that can be used to show a ragged hierarchy structure in different ways:

- *Separate Columns* - Each level's values are shown in their own column.

Country	State	City	Order Amount
United States		Washington DC	2,456.55
	Florida		1,779.42
	Michigan		891.64

- *Single Indented Column* - All levels are displayed in a single column where each child member is shown indented below its parent.

Row Labels	Order Amount
Grand Total	5,127.61
United States	5,127.61
Washington DC	2,456.55
Florida	1,779.42
Michigan	891.64

Charts

The figure below demonstrates a chart's expander buttons that can be used to expand or collapse the ragged hierarchy's members:



You can choose to set a chart up to initially display multiple levels at once along its axis, with each level in a separate row, as described in [Visualization tab examples](#). By default, an expanded axis label always displays its children directly adjacent.

Filters

A hierarchy value filter allows you to expand each member to view its children, and expand down to different depths depending on whether each member has any members below it.

Value

Washington DC, Florida

Search Values

(All)

United States

Florida

Orlando

Michigan

Washington DC

Order Amount

4,235.97

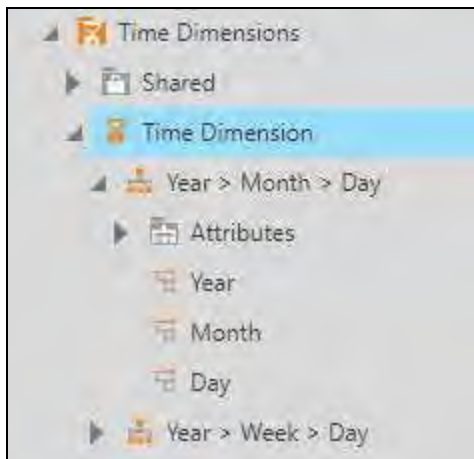


Note: If you choose a cascading member filter instead, it will be unable to list members that skip a level.

Create a Time Dimension

This applies to: *Managed Dashboards, Managed Reports*

Each project includes a time dimension with hierarchies ready to use for filtering, sorting, grouping, and drill down.



This time dimension has various settings you are able to customize. You can also create new time dimensions based on typical *Gregorian* calendars, or other calendar systems such as *13 Period, Fiscal, ISO 8601, and Reporting (Marketing)*.

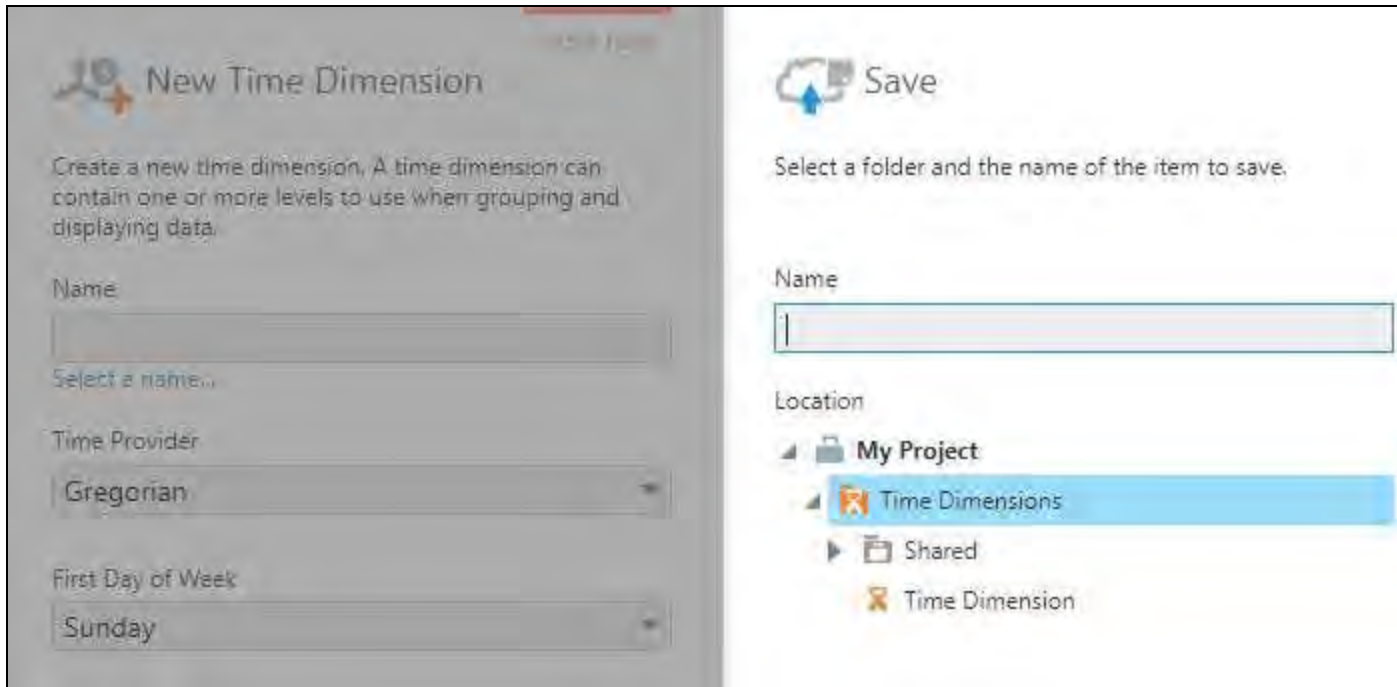
Related video: [Introduction to Time Dimensions](#)

Create a New Time Dimension

There are multiple ways you can open the *New Time Dimension* dialog:

- In the [main menu](#), choose **Business** and then **Time Dimension**, then click **Create**.
- When you have another file open with an [Explore window](#), right-click the *Time Dimensions* folder and choose **New Time Dimension**.

Click inside the **Name** text box and enter a name for your time dimension in the Save dialog, then click **Save**.



The screenshot shows two side-by-side panels from the Logi Symphony v24 interface. The left panel is titled 'New Time Dimension' and contains the following fields: 'Name' (with a placeholder 'Select a name...'), 'Time Provider' (set to 'Gregorian'), and 'First Day of Week' (set to 'Sunday'). The right panel is titled 'Save' and contains a 'Name' field (with a placeholder '|') and a 'Location' tree view. The 'Location' tree view shows a hierarchy: 'My Project' (expanded) -> 'Time Dimensions' (selected) -> 'Shared' -> 'Time Dimension'.

Returning to the *New Time Dimension* dialog, set **Time Provider** to the type of calendar you want: *Gregorian*, *Fiscal*, *ISO 8601*, *Reporting (Marketing)*, or *13 Period*. The remaining fields are different depending on your choice.

Gregorian

A Gregorian calendar is also known as a Western calendar and is the most widely used general purpose calendar.

The **First Day of Week** can be changed from the default value of *Sunday* if needed.

Next, select the **Time Periods**. The time dimension hierarchies and their levels will be automatically created based on this selection.

Select a name...

Gregorian

First Day of Week

Sunday

Time Periods

☒ Year

☐ Half Year

☒ Quarter

☒ Month

☐ Week

☒ Day

☐ Hour

☐ 15 minutes

☐ 5 minutes

☐ Minute

☐ 15 seconds

☐ 5 seconds

☐ Second

Date Range

From

2009/01/01

To

2017

Check In

☒

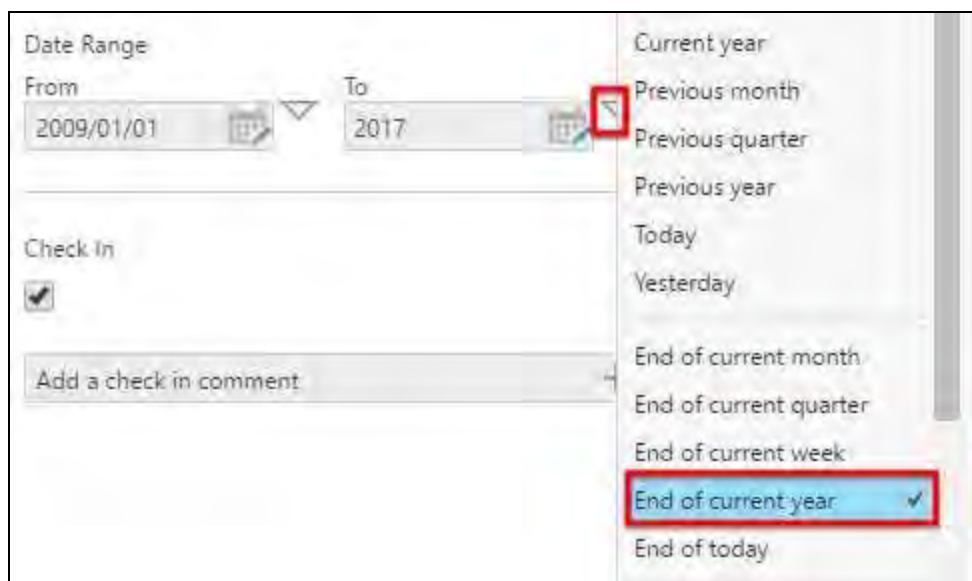
Add a check in comment



Note: Adding time periods at lower levels increases the number of members. For example, going from days to seconds increases the number by a factor of 86,400 (24x60x60). Choosing a short time period over data spanning a wide range of dates can increase memory requirements if used in a data cube stored in-memory.

Adjust the **Date Range** of the time dimension according to the data you're working with, which can determine the range of dates selectable in filters.

By default, the start date is set to a fixed value equal to eight years before the beginning of the current year. The **To** or end date is not fixed by default and is instead set to the token value *End of current year*. By using a token, the end date will change automatically as time advances from year to year.



Date Range

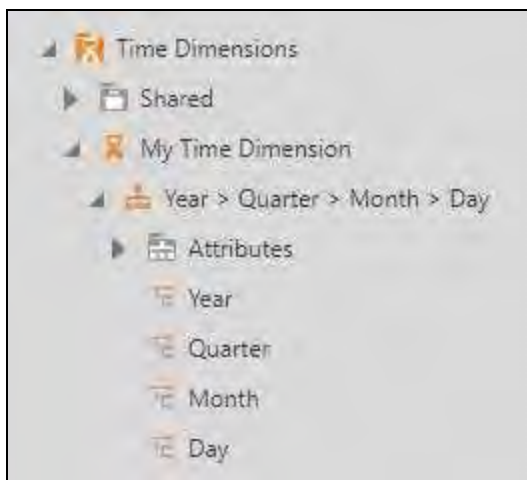
From 2009/01/01 To 2017

Check In ☒

Add a check in comment

Current year
Previous month
Previous quarter
Previous year
Today
Yesterday
End of current month
End of current quarter
End of current week
End of current year ✓
End of today

Click **Save** at the bottom to create the time dimension.



You can now find and expand your time dimension in the **Explore** window in metric sets and views, and find one or more hierarchies and attributes generated based on the time periods you chose. These are described in more detail in a later section.

Fiscal

A *Fiscal* calendar corresponds to the 12-month accounting period (or fiscal year) for an organization. This period may be different from the typical Gregorian calendar year.

To set up a fiscal calendar, choose the **First Day of Week**.

Next, you can optionally set **Fiscal Calendar Naming Convention** to *Calendar year+ 1* if the start of a fiscal year should be named after the following calendar year.

Then choose the starting month, day, and hour of the fiscal year.

Time Provider

Fiscal

First Day Of Week

Sunday

Fiscal Calendar Naming Convention

Calendar year name + 1

Fiscal Year Start Month

January

Fiscal Month Start Day

1

Fiscal Day Start Hour

Midnight

Time Periods

☐ Year
☐ Half Year
☐ Quarter
☐ Month
☐ Week
☐ Day
☐ Hour
☐ 15 minutes
☐ 5 minutes
☐ Minute
☐ 15 seconds
☐ 5 seconds
☐ Second

Date Range

From
To

Select the **Time Periods**, which will define the time dimension hierarchies and their levels.

Update the **Date Range** if needed as described in the Gregorian case, and then click **Save** at the bottom to create and check in the time dimension.

ISO 8601

The *ISO 8601* calendar is an international standard based on week numbers (e.g., *2016-W01* represents the first week of 2016).

To set up the ISO 8601 calendar, select the **Time Periods**, which will define the time dimension hierarchies and their levels.

Update the **Date Range** if needed as described in the Gregorian case, and then click **Submit** to create and check in the time dimension.



The screenshot shows a configuration window for the ISO 8601 calendar. At the top, the 'Time Provider' is set to 'ISO 8601'. Below this, the 'Time Periods' section contains a list of time intervals with checkboxes: Year, Week, Day, Hour, 15 minutes, 5 minutes, Minute, 15 seconds, 5 seconds, and Second. The 'Date Range' section at the bottom has 'From' and 'To' date pickers. The 'From' date is set to '2012/01/01' and the 'To' date is set to '2020'.

Reporting (Marketing)

A *Reporting* calendar (also known as *Marketing*, 4-4-5, 4-5-4, etc.) is commonly used in retail and manufacturing industries. This calendar divides a year into 4 quarters, where each quarter is comprised of 13 weeks.

Time Provider
Reporting (Marketing)

First Day Of Week
Sunday

Start Month
January

Week Rule
First 4 day week

Month Rule
4-5-4

Fiscal Calendar Naming Convention
Calendar year name

Month With Extra Week
Last

Combine Weeks 52-53 Into 14 Day Week
☐

Time Periods
☐ Year

The 13 weeks within each quarter may be further divided into a combination of 4-week months and 5-week months. For example, a 4-5-4 reporting calendar defines a quarter using the combination of a 4-week month, followed by a 5-week month, and then followed by a 4-week month. The other possible **Month Rule** options you can select are 4-4-5 and 5-4-4. The reporting calendar effectively lets you define a *month-week* hierarchy, whereas weeks do not divide evenly into Gregorian calendar months.

Since a reporting year consists of 52 weeks, totaling 364 days, after a certain number of years an extra week is inserted into a year to ensure it remains close to Gregorian calendar years that are typically 365 days. The **Week Rule** options determine when each reporting year starts based on the day of the week on which



- Archive of documentation for Logi Symphony v24

the Gregorian year starts, which results in most years having 52 weeks but some having 53. The reporting year can start on the first day of the week of the *First 4 day week*, the *First full week*, or the *First day* that occurs within the Gregorian year.

When a year is made up of 53 weeks as described above, the **Month With Extra Week** setting determines whether this week is added to the *First* or *Last* month of the year (e.g., turning the Last 4-5-4 month into a 4-5-5 month). You can also choose **Combine Weeks 52-53 Into 14-day Week** if preferred.

13 Period

The *13 Period* calendar divides each typical year into 13 periods of 4 weeks each, making each *period* similar to a month but with an equal number of days for easier 'monthly' comparisons.

Time Provider
13 Period

First Day Of Week
Sunday

Start Month
January

Period With Extra Week
P13

Quarter With Four Periods
Q4

Week Rule
First 4 day week

Year Naming Convention
Calendar year name

Time Periods
☐ Year
☐ Quarter
☐ Period
☐ Week
☐ Day
☐ Hour
☐ 15 minutes

As with the reporting calendar, a 13 Period year consists of 52 weeks, totaling 364 days, and an extra week must be inserted into certain years to ensure it remains close to Gregorian calendar years. The **Week Rule** options determine this as described above for the reporting calendar.



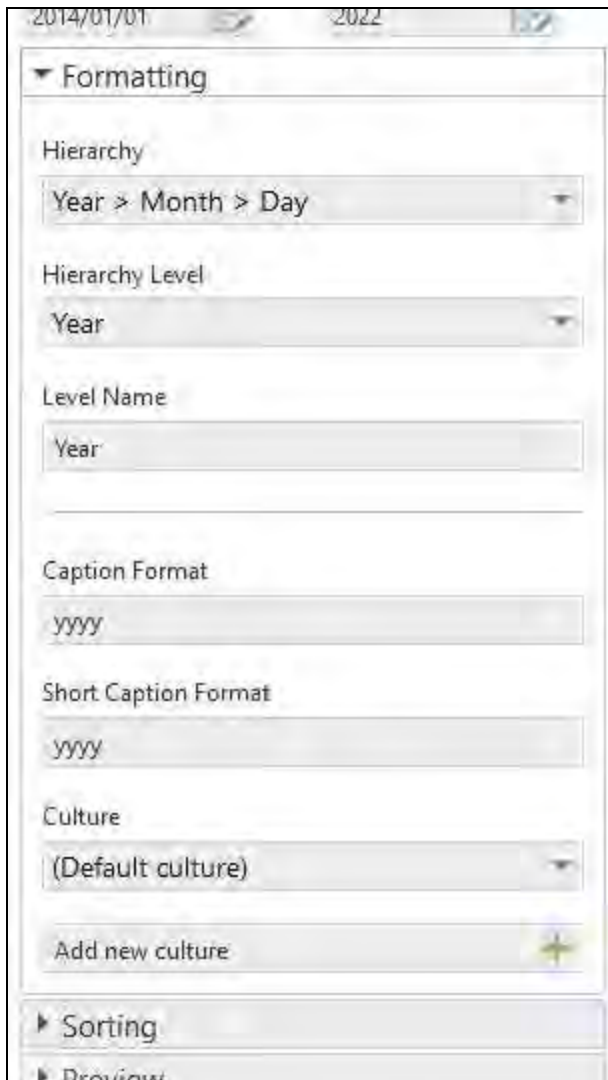
When this causes a year to have 53 weeks, the extra week is typically assigned to the 13th period of that year, but you can customize this with the **Period With Extra Week** setting.

If you select to include the Quarter time period in your time dimension, one of your four quarters must be made up of four periods rather than three in order to have 13 periods per year. This is customizable with the **Quarter With Four Periods** setting.

Format Time Dimension Levels

You can change the formatting of a time dimension's levels once it's been created (click to **Save** a new time dimension first). Find and open it in the [main menu](#) under **Business**, or right-click it in the [Explore window](#) and choose **Edit**.

In the *Time Dimension* dialog, scroll down and click to expand the **Formatting** section, which lets you choose different formatting options for each hierarchy level as well as for different cultures.



The screenshot shows the 'Formatting' panel in Logi Symphony v24. At the top, there are date range selectors showing '2014/01/01' and '2022'. The 'Formatting' section is expanded, showing the following settings:

- Hierarchy:** A dropdown menu set to 'Year > Month > Day'.
- Hierarchy Level:** A dropdown menu set to 'Year'.
- Level Name:** A text input field containing 'Year'.
- Caption Format:** A text input field containing 'yyyy'.
- Short Caption Format:** A text input field containing 'yyyy'.
- Culture:** A dropdown menu set to '(Default culture)'.
- Add new culture:** A button with a plus icon.

Below the 'Formatting' section, there is a 'Sorting' section which is currently collapsed.

If the time dimension you're editing has multiple hierarchies, first select a **Hierarchy** for formatting.

Next, choose a **Hierarchy Level** to format. You can optionally customize the **Level Name**.

Set the desired formats in the **Caption Format** and **Short Caption Format** boxes. These can be .NET [standard date and time](#) formats or [custom date and time](#) formats, or see the sections below for additional options for certain calendar types and levels.

Standard date and time formats are specified by a single character and place their date and time components automatically according to the culture of each logged-in user. Below are some examples:

Standard Format	Specifier	Sample Text
Short date	d	6/12/2015
Long date	D	Friday, June 5, 2015
Short time	t	8:31 AM
Long time	T	8:31:05AM
Long date, short time	f	Friday, June 5, 2015 8:31 AM
Long date, long time	F	Friday, June 5, 2015 8:31:05 AM
Month/day	M	June 05
Month/year	Y	2015 June

Custom formats allow you to place each date and time component yourself. An example of a complete custom format is *yyyy-MM-dd*. Some common custom format specifiers:

Custom Format	Specifier	Sample Text
Year	y, yy, yyyy	5, 05, 2005
Month	M, MM, MMM, MMMM	4, 04, Apr, April
Day	d, dd, ddd, dddd	9, 09, Wed, Wednesday
12-hour clock	h, hh	9, 09
24-hour clock	H, HH	9, 09 & 21, 21
Minutes	m, mm	6, 06
Seconds	s, ss	8, 08
AM/PM	t, tt	a, AM
Fractional seconds	f, ff, fff	6, 61, 611
Literal text	'My Text'	My Text



Note: Short captions are used where upper level details are not needed, such as when [displaying each level as a separate row of axis labels on a chart](#).

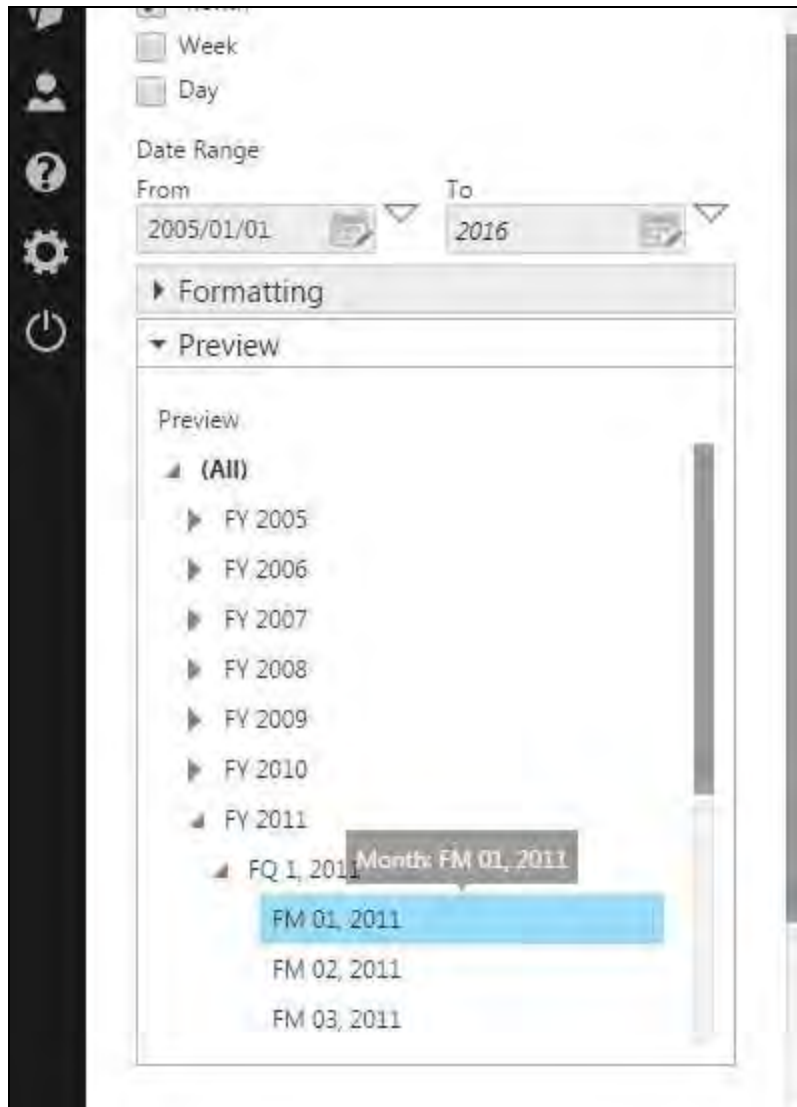
The **Culture** drop-down optionally allows you to set up different format settings for users logged in with different [cultures \(languages\)](#). You can click **Add new culture** and enter a localization [culture code/tag](#) such as *fr-Fr* or *de-AT*, then switch between options in the culture drop-down while setting up your formats.

Click the **Save** button at the bottom of the dialog when finished.

Fiscal Month Formatting

Fiscal time dimensions provide the option of formatting months using either fiscal or calendar values, or a combination of both.

After opening your fiscal time dimension, scroll down and expand the **Preview** section, and observe that the month level is formatted to display Fiscal Month Number and Fiscal Year values.

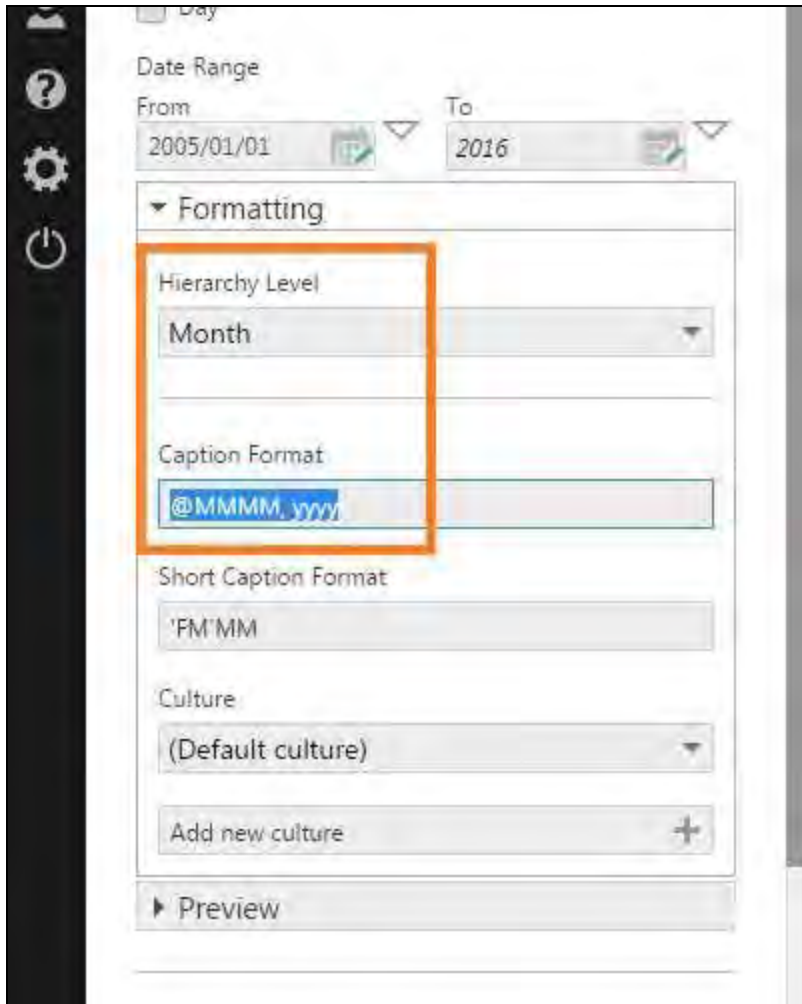




The following table summarizes the formats that are supported when setting up the captions in the Formatting section:

First Character of Format	Displays	Options
Not @ or !	Fiscal Month Number and Fiscal Year	■ Use MM to show Fiscal Month Number ■ Use yy or yyyy to show Fiscal Year Example: 'FM' MM, yyyy
@	Calendar Month and Fiscal Year	■ Use MM, MMM, or MMMM to show Calendar Month ■ Use yy or yyyy to show Fiscal Year Example: @MMMM, yyyy
!	Calendar Month and Calendar Year	■ Use MM, MMM, or MMMM to show Calendar Month ■ Use yy or yyyy to show Calendar Year Example: !MMMM, yyyy

As an example of displaying the Calendar Month Name and Fiscal Year values, set the **Hierarchy Level** to *Month* and change the **Caption Format** from 'FM' MM, yyyy to @MMMM, yyyy.



Day

Date Range

From 2005/01/01 To 2016

▼ Formatting

Hierarchy Level

Month

Caption Format

@MMMM, yyy

Short Caption Format

FM/MM

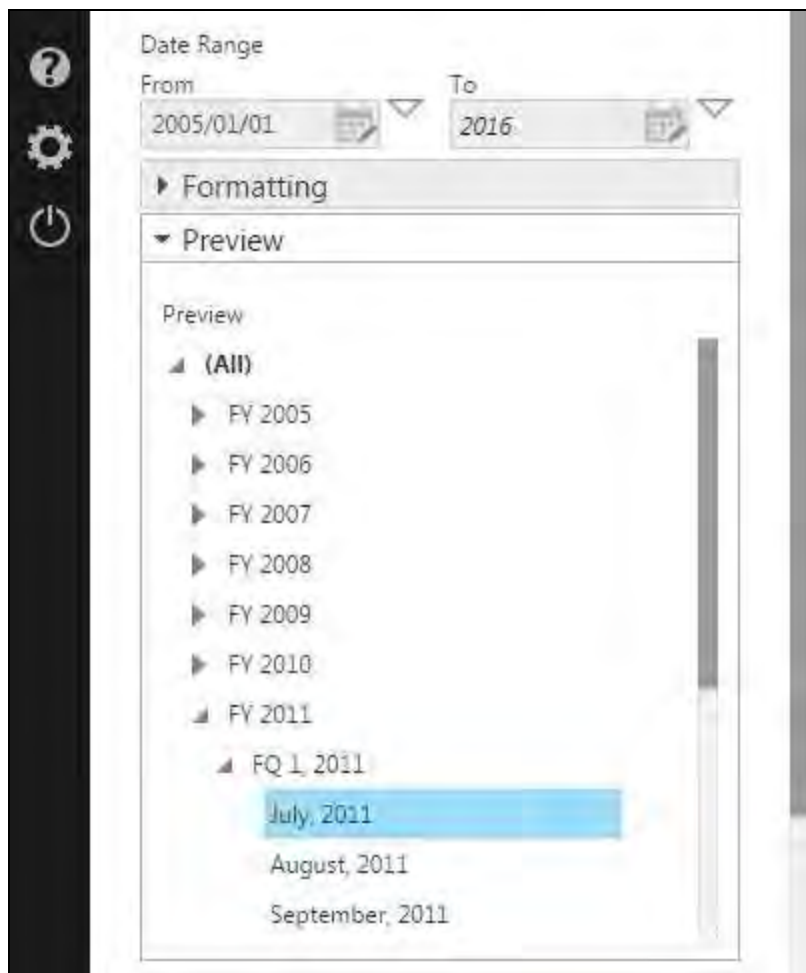
Culture

(Default culture)

Add new culture +

► Preview

Click **Save** when finished. Re-opening the time dimension again and going to the **Preview** section then shows different results:



Fiscal Year Formatting

You can change the default fiscal time dimension formatting to include both calendar years at the Year level by using `{CYStart:<format>}` and `{CYEnd:<format>}` placeholders, where you can specify the formats for each.

For example, with the **Hierarchy Level** set to Year, change the **Caption Format** to `'FY' {CYStart:yyyy}/{CYEnd:yy}`.

▼ Formatting

Hierarchy

Year > Quarter > Month

Hierarchy Level

Year

Caption Format

'FY' (CYStart:yyy)/(CYEnd:yy)

Short Caption Format

'FY'yyyy

Culture

(Default culture)

Add new culture +

After clicking to **Save** your changes, opening the time dimension again and expanding the **Preview** section shows these results:



Week Start/End Formatting

When **Time Provider** is set to *Gregorian*, *Fiscal*, *Reporting (Marketing)* or *13 Period*, you can format the Week level to include the start and/or end of the week. Use `{CWStart:<format>}` and `{CWEnd:<format>}` in the caption settings.

For example, if the **Caption Format** is set to `{CWStart:MMM/d/yyyy} 'to' {CWEnd:MMM/d/yyyy}`, weeks will be displayed in the style *May/26/2019 to Jun/1/2019*.

Quarter Start/End Formatting

When **Time Provider** is set to *Gregorian*, *Fiscal*, *Reporting (Marketing)* or *13 Period*, you can format the Quarter level to include the start and/or end of the quarter. Use `{CQStart:<format>}` and `{CQEnd:<format>}` in the caption settings.

For example, if the **Caption Format** is set to `{CQStart:MMM/yyyy} 'to' {CQEnd:MMM/yyyy}`, quarters will be displayed in the style *Apr/2019 to Jun/2019*.

Reporting and 13 Period Weeks

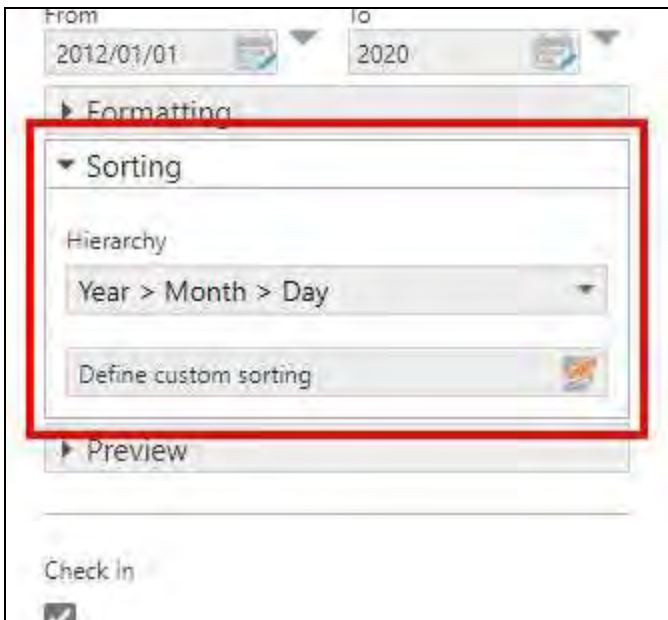
Weeks in a *Reporting (Marketing)* or *13 Period* time dimension are identified by default by week numbers relative to the year. Since weeks are evenly divisible into 'months' or 'periods' in these time dimensions, you can choose to format them instead to display the week numbers relative to each month or period.

When setting captions for the *Week* level, you can type `{WeekOfMonth}` for a reporting time dimension, and `{WeekOfPeriod}` for a 13 period time dimension.

Sorting Time Dimension Levels

You can optionally change the sorting of the time dimension's hierarchy levels. This will determine the default sorting of the time dimension's members when used in metric sets and when listed in filters connected to the time dimension.

Open your time dimension from the [main menu](#) or [Explore window](#), and click to expand the **Sorting** section of the *Time Dimension* dialog.



If your time dimension has multiple hierarchies, choose one in the **Hierarchy** drop-down.

Click **Define custom sorting** to open the *Set Up Hierarchy Sorting* dialog, where you can click the **Edit** button next to each of the hierarchy's levels to set up its sorting.

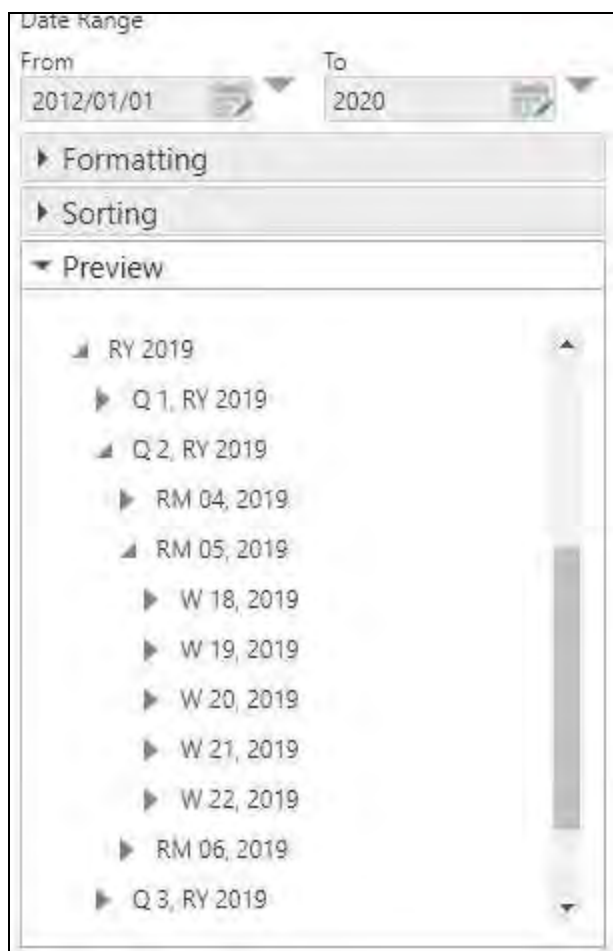
Sorting each level of a time dimension is similar to how you can configure each hierarchy level's sorting in a metric set, allowing you choose between *Ascending*, *Descending*, and specific members to display at the top & bottom. For details on these settings, see [Define custom hierarchy sorting](#).

Preview Your Time Dimension

Open an existing time dimension after it's been created or saved to preview how its members are organized, formatted, and sorted.

In the *Time Dimension* dialog, scroll down to find the **Preview** section which lets you preview the hierarchy levels and members of your time dimension.

For example, the figure below shows the preview for a Reporting calendar used by a large retail company.





Time Hierarchies and Attributes

Hierarchies with multiple levels will be automatically available from a time dimension based on the indicated type of calendar and the selected time periods. For example, the built-in default time dimension has a Gregorian calendar with the *Year*, *Month*, *Week*, and *Day* time periods selected. Since weeks are not divisible into months, there will be two hierarchies available from the same time dimension:

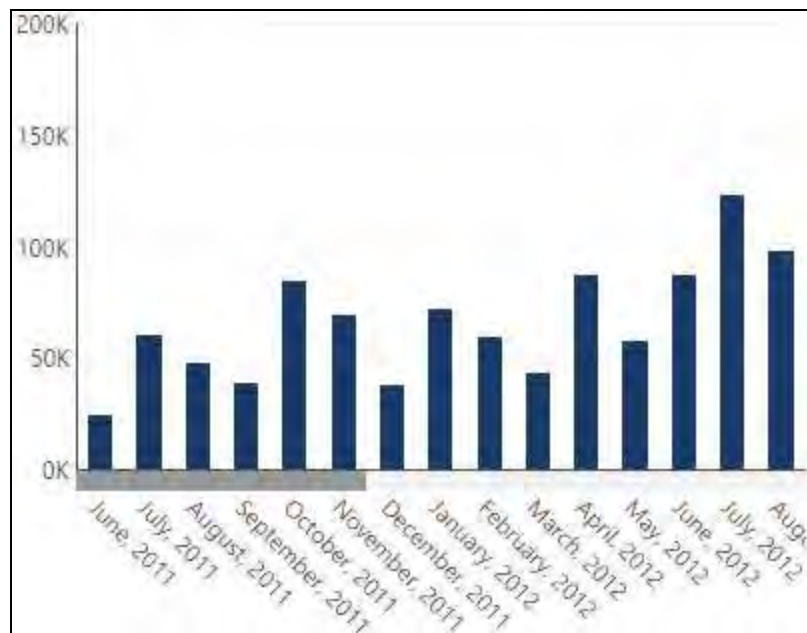
- Year > Month > Day
- Year > Week > Day

Within each hierarchy there may also be attributes available to use in metric sets:

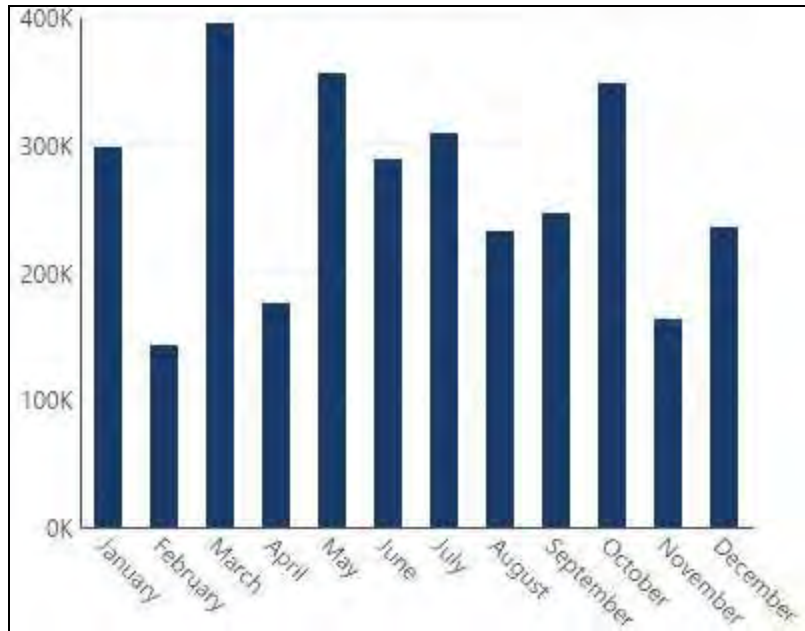
- Quarter of year
- Month of year
- Week of year
- Day of week
- Day of month
- Hour of day

These attributes are available for any calendar type and behave differently from the *Month* and *Day* levels of hierarchies as shown below:

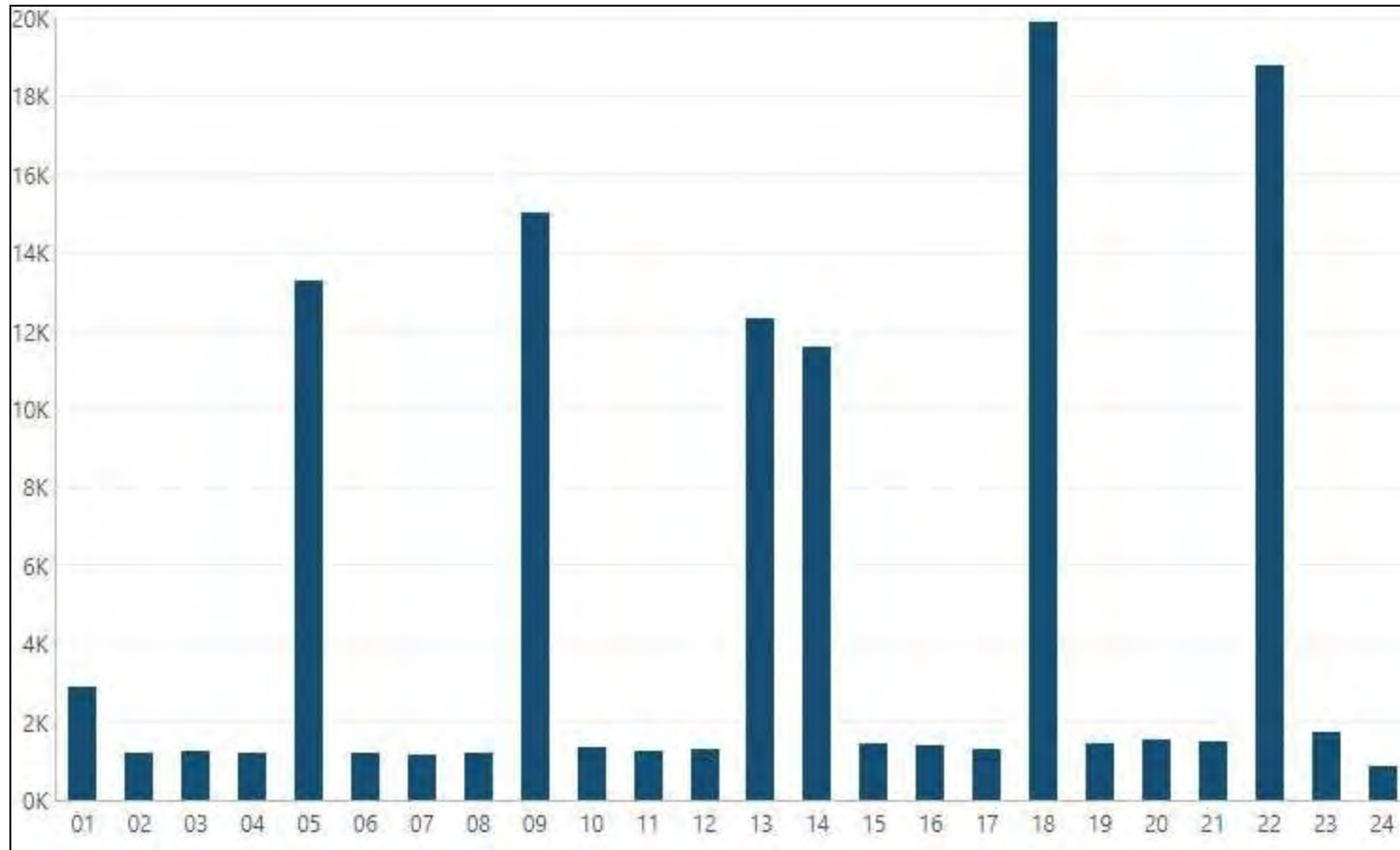
- The *Month* and *Day* levels group values occurring during a single month occurring in a particular year, or a single day occurring during a particular week:



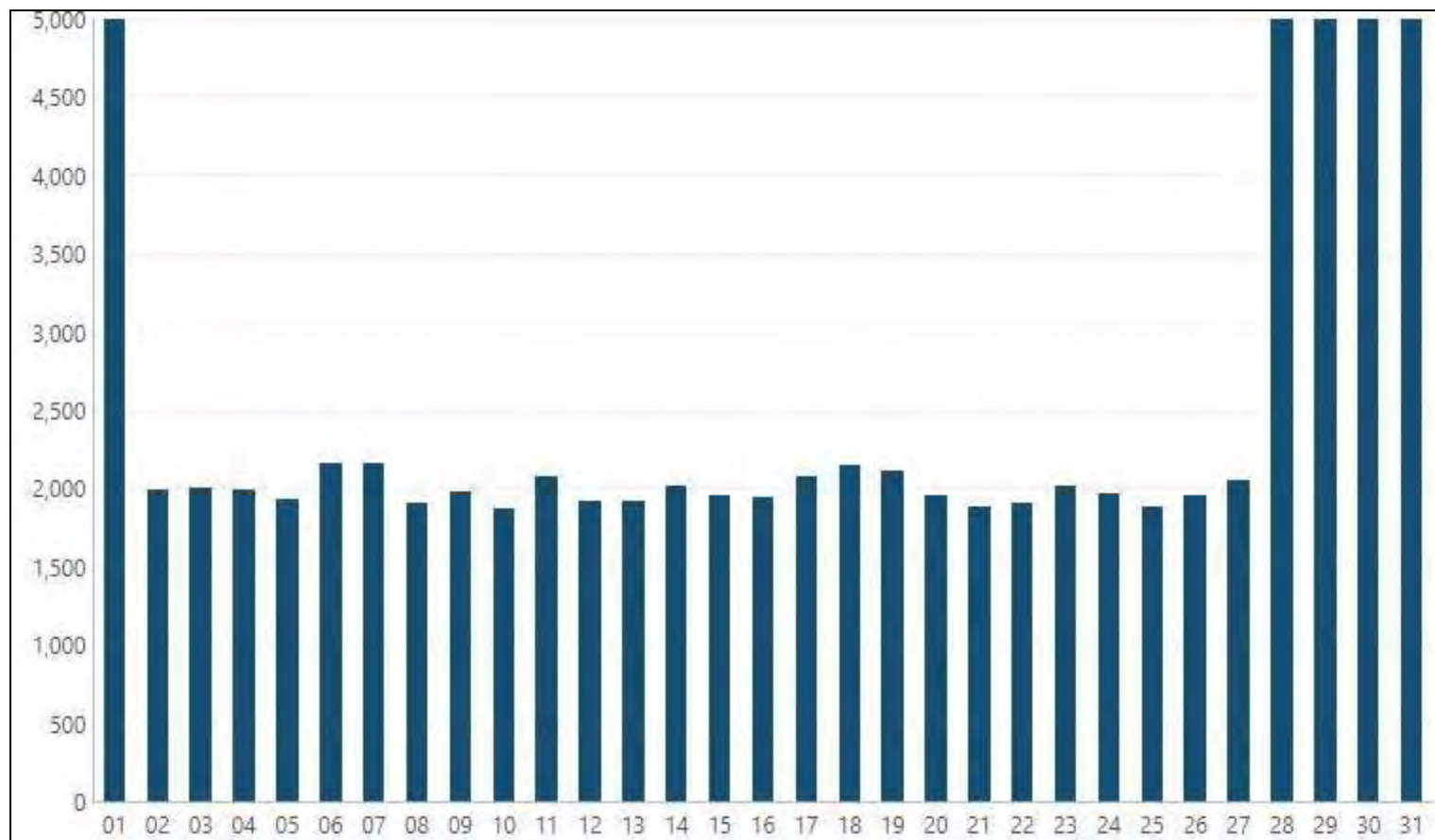
- *Month of year* groups the values by month from all years, and *Day of week* groups the values occurring during a single day of any week:



- *Week of year* groups the values by week number for all years:

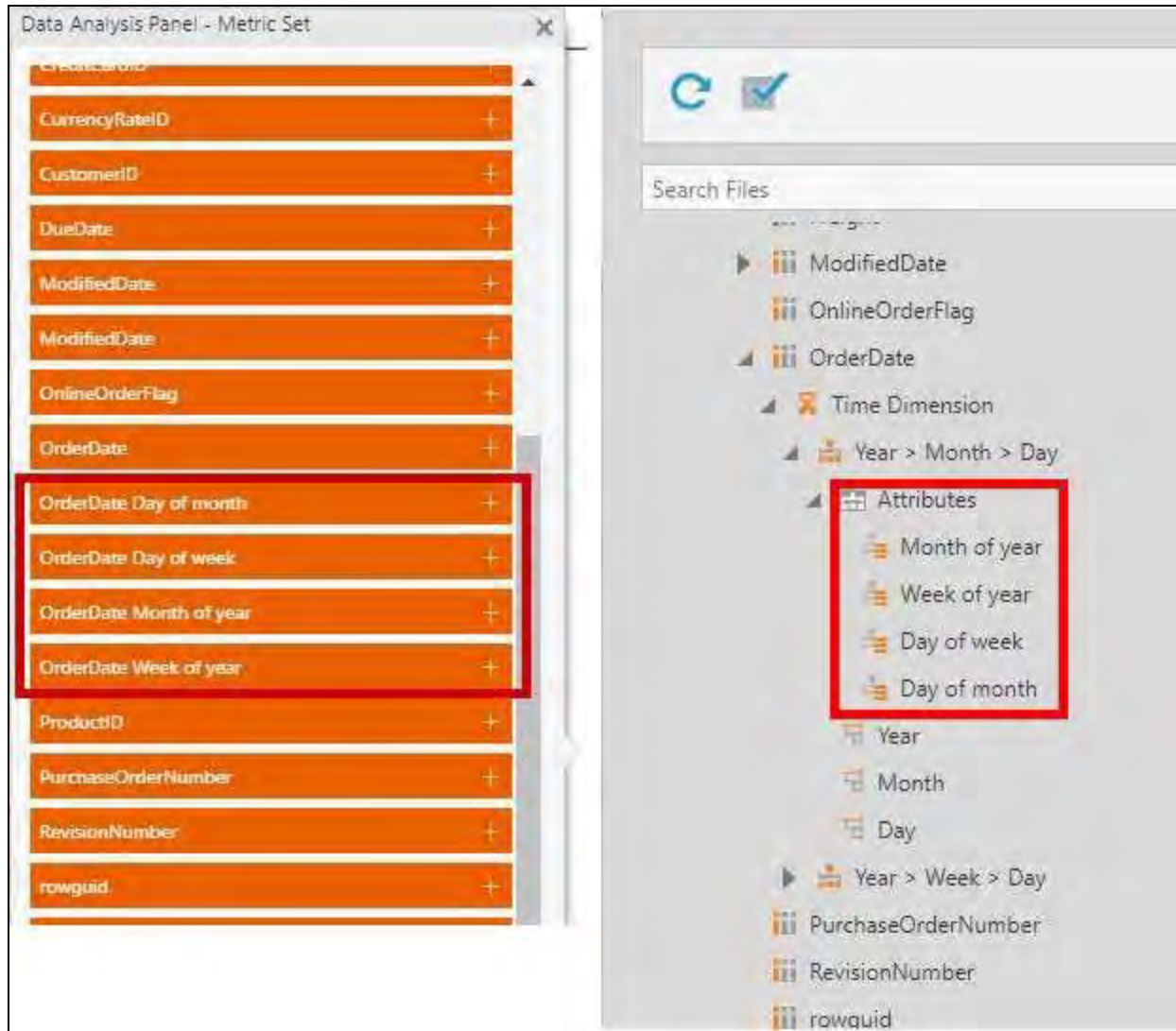


- *Day of month* groups the values by day number for all months and years:



To apply a time dimension hierarchy or attribute to your data to group by a level like Year or attribute like *Month of year*, drag it from the Explore window to the Data Analysis Panel. You can drag it on top of some existing 'raw' date and time data in the Data Analysis Panel previously dragged directly from a data connector, or you can drop it onto the area labeled *drop a hierarchy* to add this data and apply the time dimension in one step.

For details, see [Symphony Metric Sets](#) or the walkthrough [View data with a chart and drill down](#). For data that was prepared as a data cube, the data cube can be edited and its [Process Result](#) configured to use a time dimension hierarchy.



If you already used a time dimension hierarchy in your metric set or if one was used in the data cube you're using, you can also **click to add** its attributes in the Data Analysis Panel from the list that's displayed. You can also drag them from the Explore window, and time dimension attributes can be dragged on their own if you only want the attribute and not its hierarchy in your metric set.

For more information, see:



- Archive of documentation for Logi Symphony v24

- Video: [Introduction to Time Dimensions](#)
- [Set or Reset the Default Time Dimension](#)
- [View Data With a Chart and Drill Down](#)
- [Automatic Joins and Hierarchies](#)
- [The Managed Dashboards Data Model](#)
- [Adding Filters](#)

Define Custom Hierarchy Sorting

This applies to: *Managed Dashboards, Managed Reports*

You can set up more customized sorting for a hierarchy's values in your metric set using the Data Analysis Panel. For each level, choose a sort direction and optional members to appear on top or the bottom.

Similar sorting options are also available when editing a [hierarchy](#) or [time dimension](#) to define their default sorting when added to a metric set or used in a filter.

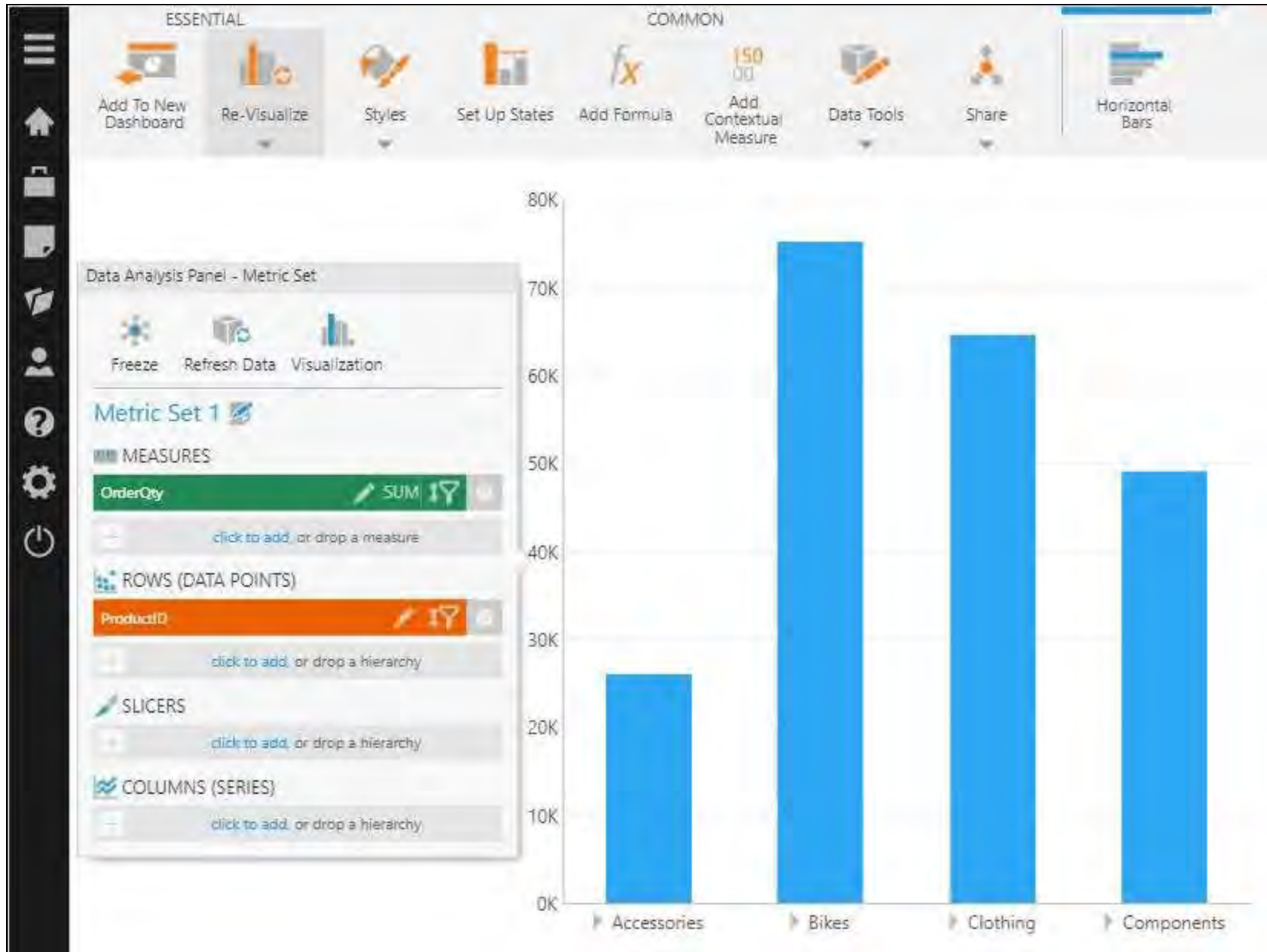


Note: The metric set can also be quickly sorted directly from the Data Analysis Panel or the context menu of a visualization. See [Sort and Filter Metric Data](#) and [Select and Group Data for a Metric Set](#).

Defining Custom Hierarchy Sorting

For this example, create or edit a metric set that is configured with one measure and a hierarchy under **Rows**.

From the toolbar, re-visualize the metric set as a bar chart and observe that the bars are sorted according to *Product Category* names in ascending or alphabetical order. This is the default sorting behavior for the *Product* hierarchy and its levels.



In the data analysis panel, click the **Edit** icon for the *Product* hierarchy.



In the *Configure Metric Set Element* dialog, click **data**.



Configure 'ProductID'

Here you can quickly access the visual settings associated with this element, or configure specific settings within the metric set.

VISUALIZATION

 ORDERQTY SERIES: HORIZONTAL AXIS
 ORDERQTY SERIES: TOOLTIP
 AXIS: BOTTOM AXIS LABEL

METRIC SET DEFAULT VALUES

Caption

Unique Name

Analysis Element Name

▼ Element Settings

Unknown Member Caption

Collapse All

☐

Shown Totals

Retrieve Unknown Members

☒

Exclude From Data Export

☐

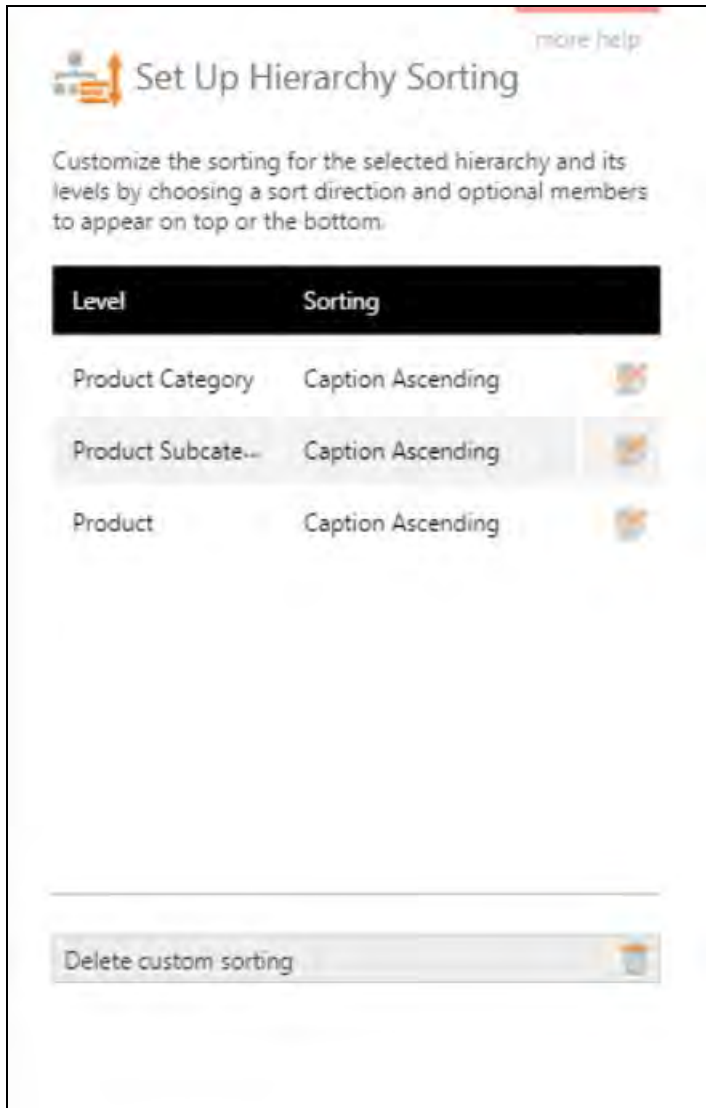
▶ Parameter Values

Define custom hierarchy sorting 

Delete this element 




In the *Set Up Hierarchy Sorting* dialog, you'll see list of hierarchy levels (e.g., Product Category, Product Subcategory, and Product).



Click the edit icon for the *Product Category* level to configure the sorting for this level.

In the *Configure Hierarchy Level Sorting* dialog, set the following options:

- Choose a **Category**. This is the member property you want to use for sorting (e.g., Caption or UniqueName).
- Choose a **Sort Direction**. For example, set it to *Descending*.



Set Up Hierarchy Sorting

Customize the sorting for the selected hierarchy and its levels by choosing a sort direction and optional members to appear on top or the bottom

Level	Sorting
Product Category	Caption Ascending
Product Subcate...	Caption Ascending
Product	Caption Ascending

Delete custom sorting



Configure Hierarchy Level Sorting

Customize how the values will be sorted for this hierarchy level. Specify the field to sort by (category), the direction and any optional top or bottom overrides. Overrides always appear at the set position and do not follow the configured sorting.

Category
Caption

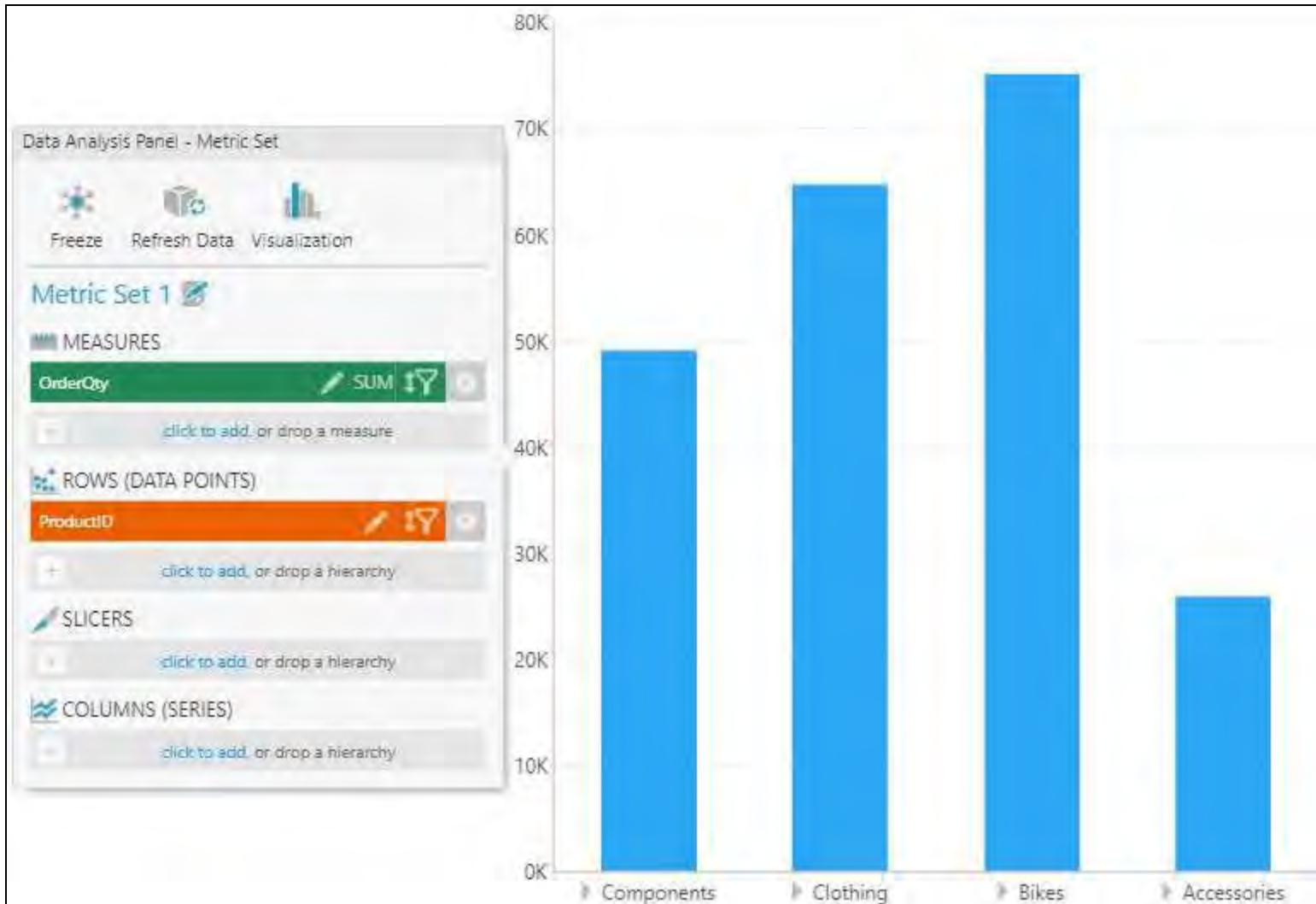
Sort Direction
Descending

Top Overrides
Add top override

Bottom Overrides
Add bottom override

Click **Submit** to save your changes.

Observe that the bars are now sorted according to *Product Category* name in reverse alphabetical order.



Specify Members To Be on Top or Bottom in Sort Order

In the *Configure Hierarchy Level Sorting* dialog, there are two additional options which let you choose one or more hierarchy members to always appear on top or bottom in the sort order.

For example, extend the previous example by clicking **Add top override**.



Configure Hierarchy Level Sorting

Customize how the values will be sorted for this hierarchy level. Specify the field to sort by (category), the direction and any optional top or bottom overrides. Overrides always appear at the set position and do not follow the configured sorting.

Category

Caption

Sort Direction

Descending

Top Overrides

Add top override

Bottom Overrides

Add bottom override

Click the drop-down that appears and select the member (e.g., Accessories) you always want to appear on top.

Configure Hierarchy Level Sorting

Customize how the values will be sorted for this hierarchy level. Specify the field to sort by (category), the direction and any optional top or bottom overrides. Overrides always appear at the set position and do not follow the configured sorting.

Category

Sort Direction

Top Overrides

- ▶ (All)
- ▶ Accessories
- ▶ Bikes
- ▶ Clothing
- ▶ Components

Click **Submit** to save your changes.

Observe that the bars are now sorted the same as before (reverse alphabetical order) except that the *Accessories* member appears first in the sort order.



You can specify which members appear at the bottom by adding a bottom override in a similar way.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Define Custom Measure Sorting](#)
- [Sorting a Table Visualization](#)

Define Custom Measure Sorting

This applies to: *Managed Dashboards, Managed Reports*

You can sort the data in your metric set by measure values using the dialog available in the Data Analysis Panel. For example, you could sort your *Sales* measure's figures for each *Product* from largest to smallest.

Sort one specific hierarchy level by a measure, and when applicable you can set the opposite axis members from a drop-down.

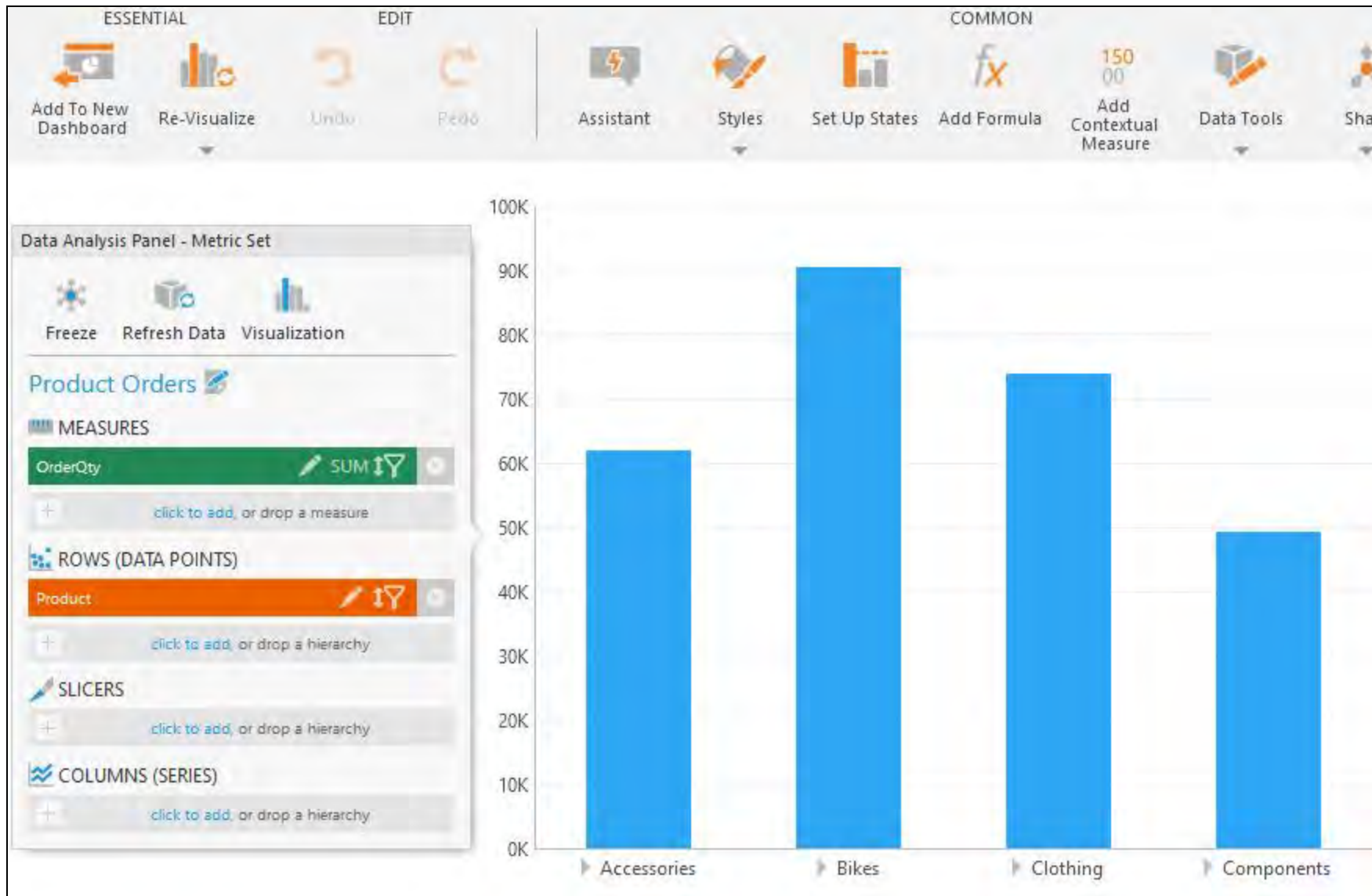


Note: You can also sort your data from the Sort & Filter popup in the Data Analysis Panel or from your visualization, and those options allow you to sort more than one hierarchy level at a time by a measure. See [Sort and Filter Metric Data](#) and [Select and Group Data for a Metric Set](#).

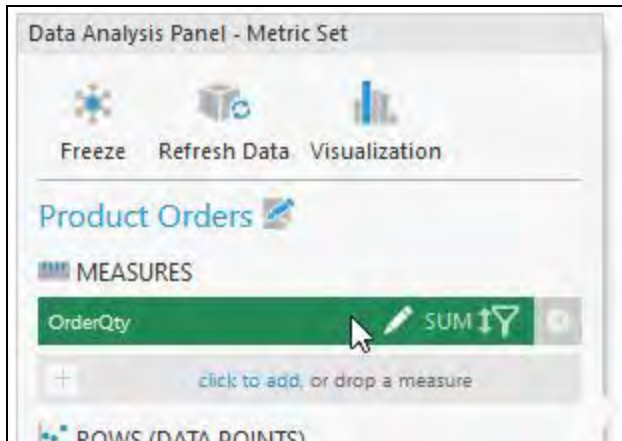
Defining Custom Measure Sorting

As an example, we have a metric set configured with one measure *OrderQty*, and a *Product* hierarchy under **Rows**.

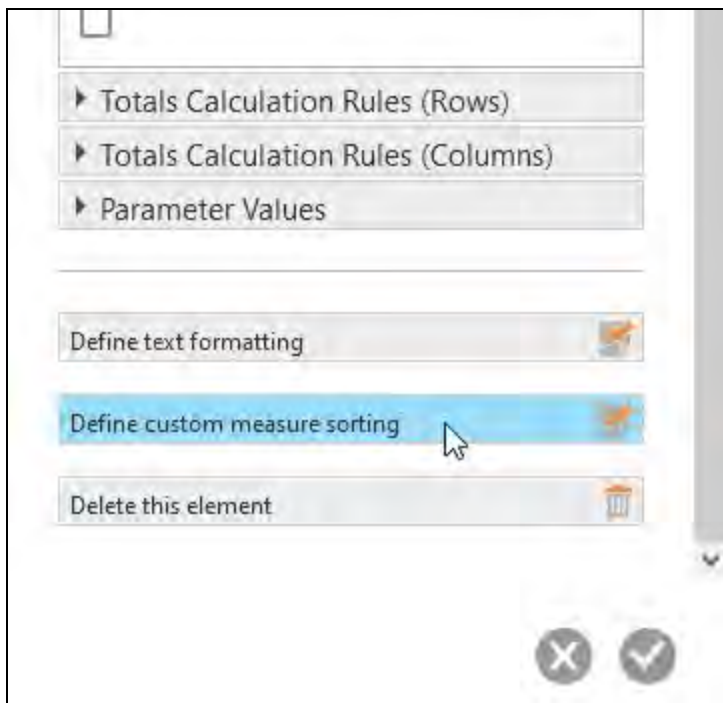
From the toolbar, we re-visualized the metric set as a bar chart and observe that the *OrderQty* measure values are unsorted along the *Product* hierarchy.



In the Data Analysis Panel, click on the *OrderQty* measure to edit it.

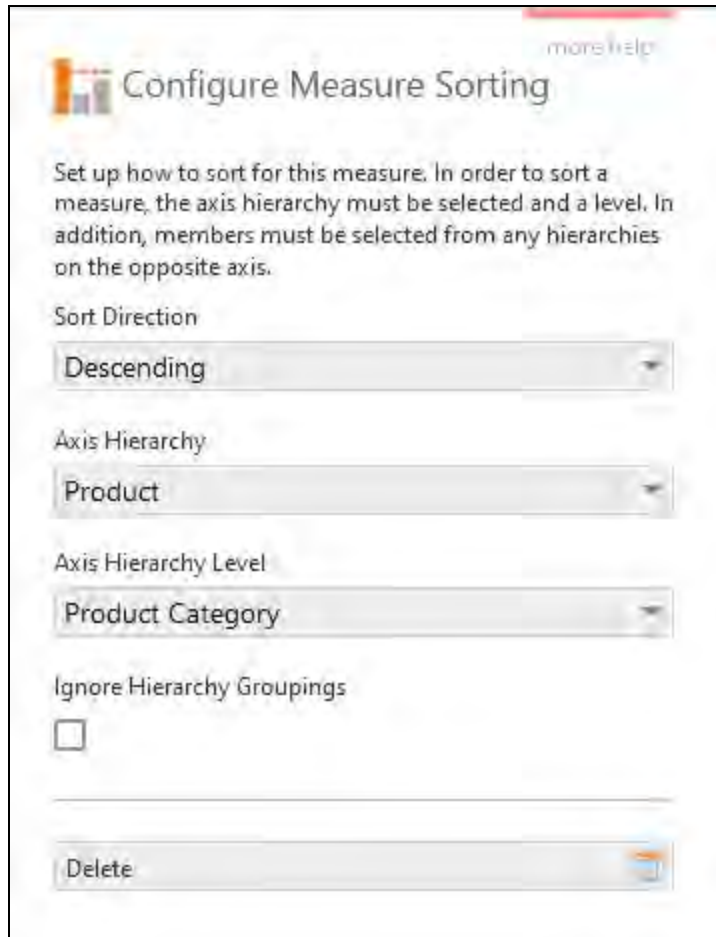


In the *Configure Element* dialog that opens, scroll down and click **Define custom measure sorting**.



In the *Configure Measure Sorting* dialog, set the following options:

- Choose a **Sort Direction**. For this example, set it to *Descending* so that measure values will go from largest to smallest.
- Select the **Axis Hierarchy**. This is the hierarchy values your want to be sorted by your measure, such as *Product*.
- Select the **Axis Hierarchy Level**. Choose a which level of the hierarchy to sort by the measure when there are multiple, such as *Product Category*.
- If you need to sort all rows by this measure independently of any groupings (including totals), select **Ignore Hierarchy Groupings**.



[more help](#)

Configure Measure Sorting

Set up how to sort for this measure. In order to sort a measure, the axis hierarchy must be selected and a level. In addition, members must be selected from any hierarchies on the opposite axis.

Sort Direction
Descending

Axis Hierarchy
Product

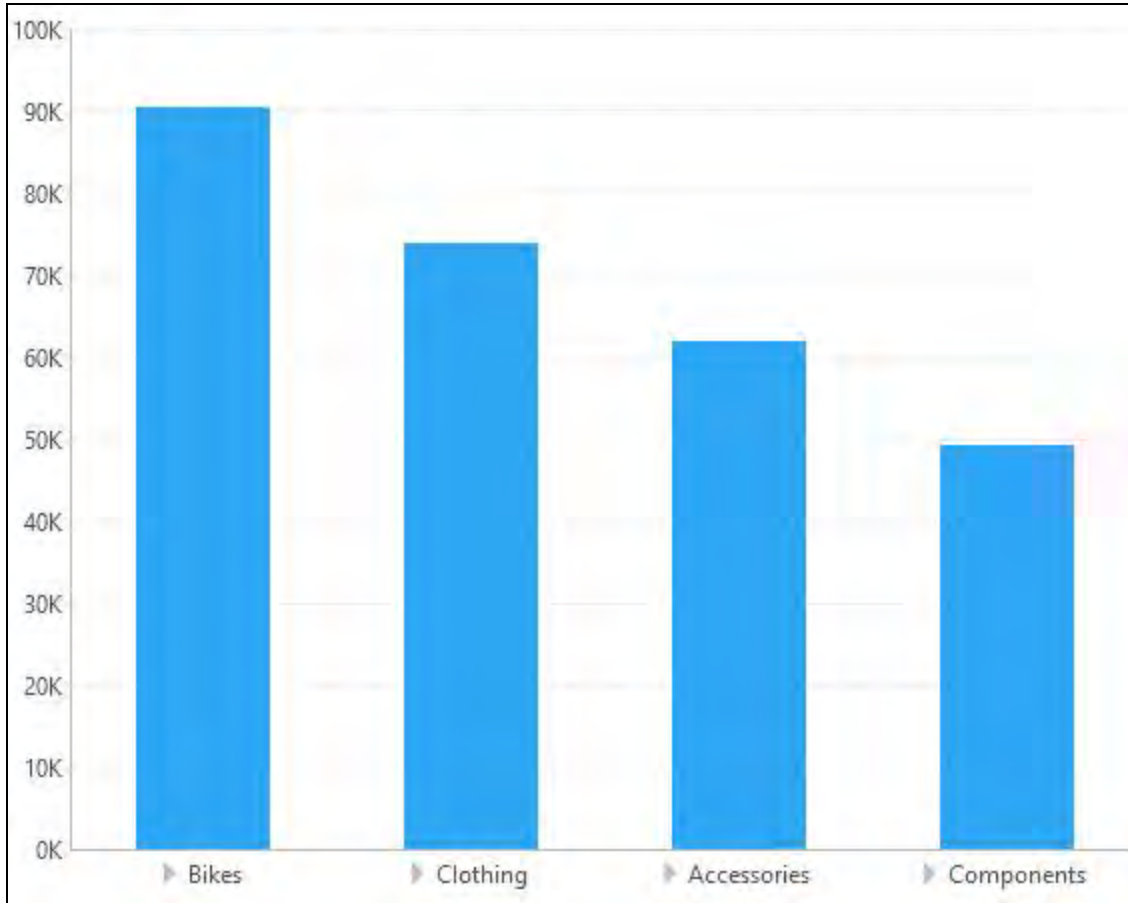
Axis Hierarchy Level
Product Category

Ignore Hierarchy Groupings
☐

Delete

Click the **Submit** button at the bottom of the dialogs to save your changes.

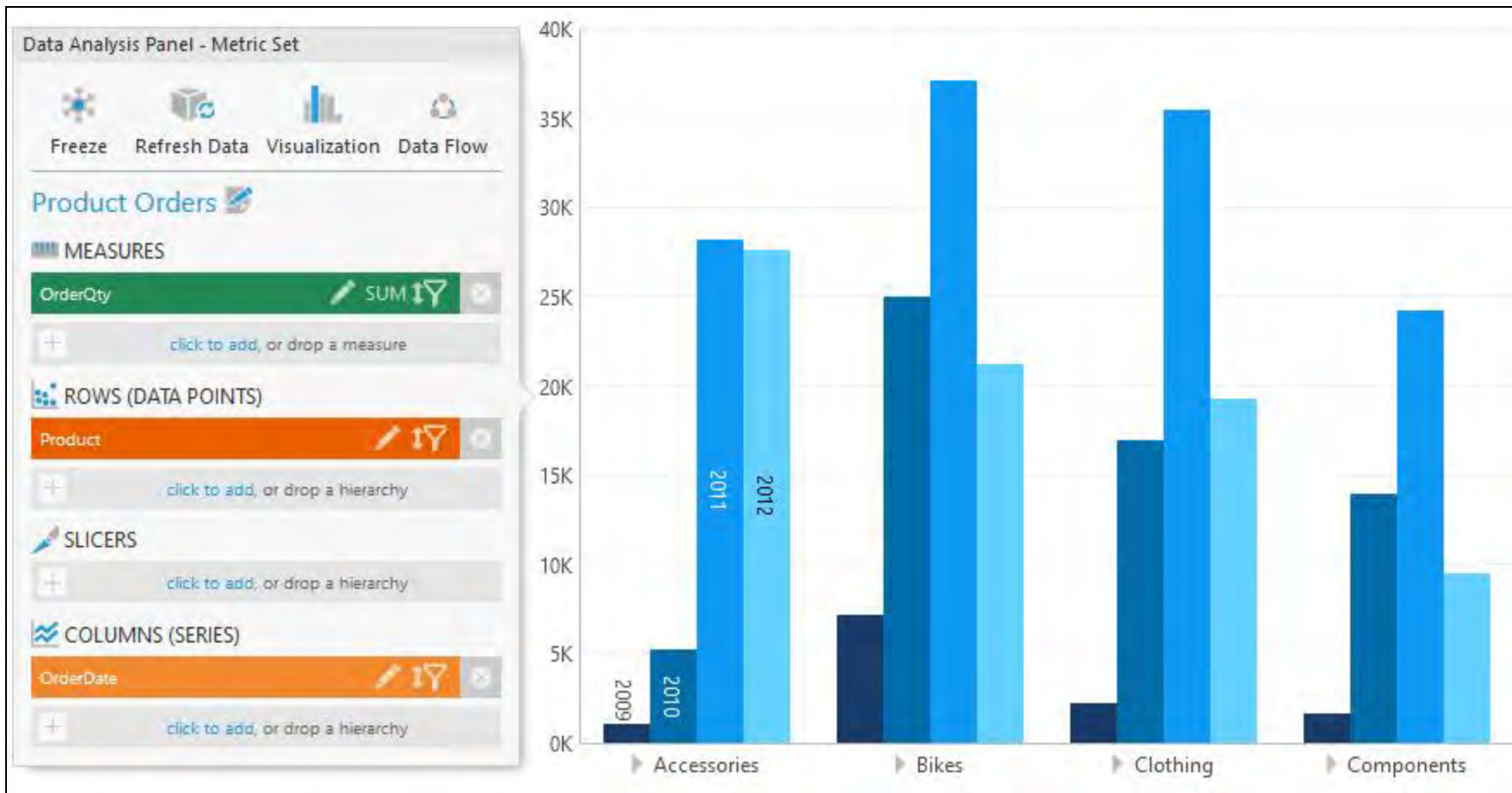
Observe that the *OrderQty* values for the *Product Category* data points in the chart are now sorted from largest to smallest.



Opposite Axis Hierarchies

When sorting the rows or data points of your metric set by a measure, for example, you might have a hierarchy under Columns in the Data Analysis Panel that splits your measure values into multiple columns or series. In this case, you would need to decide which column or series to sort by.

As an extension of the above example, consider a second hierarchy *OrderDate* added to the metric set on the opposite axis from Rows, which is **Columns (Series)**. This produces a separate data point series for each year in a bar chart.



Now when configuring the measure sorting you also need to choose which *OrderDate* (year) values to sort by under **Opposite axis members**.



Configure Measure Sorting

more help

Set up how to sort for this measure. In order to sort a measure, the axis hierarchy must be selected and a level. In addition, members must be selected from any hierarchies on the opposite axis.

Sort Direction

Descending

Axis Hierarchy

Product

Axis Hierarchy Level

Product Category

Opposite axis members:

OrderDate:

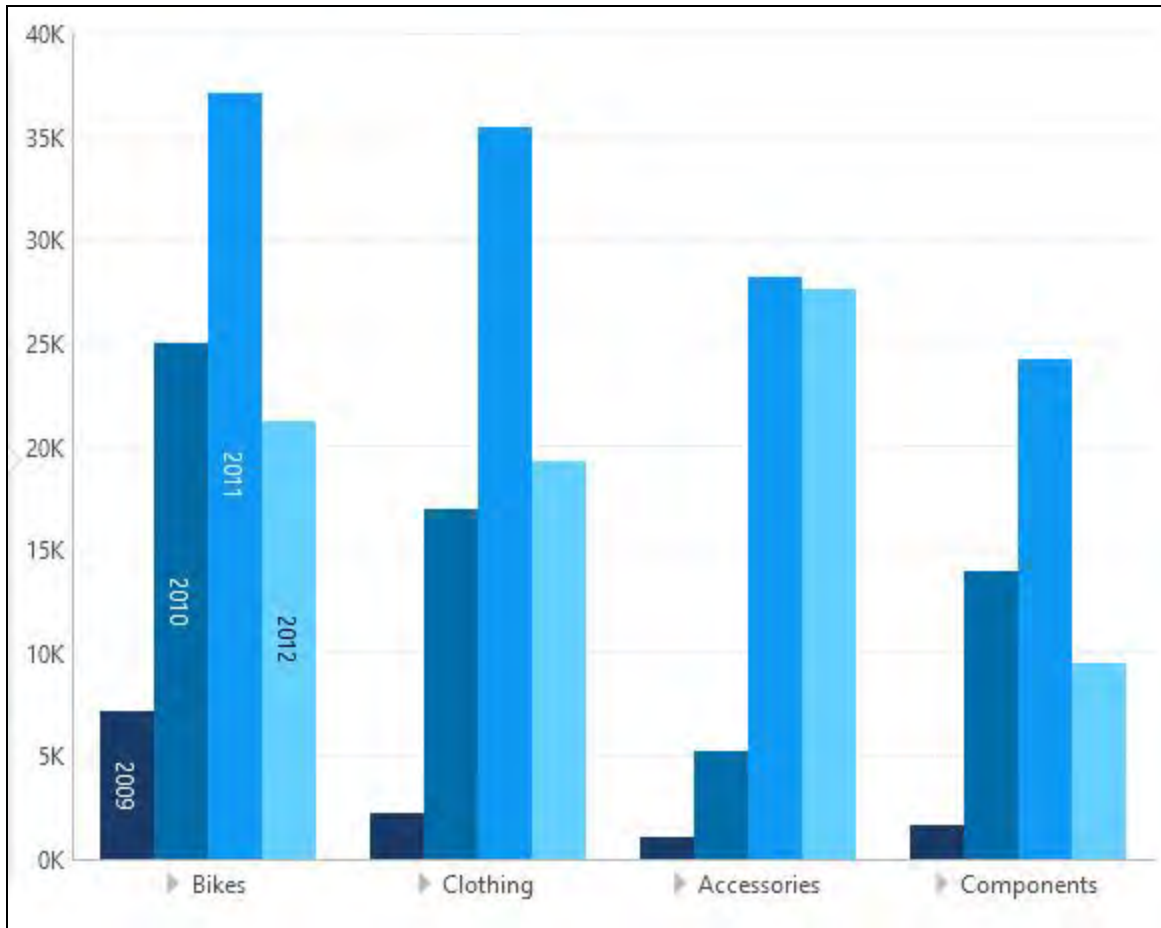
2011

Delete



Note: You can select *(All)* to sort by the measure value total for all opposite axis members even if those totals are not being visualized directly. For example: the total of all years for each product category.

The resulting metric set shows measure values sorted in descending order for the selected year.



For more information, see:

- [Sorting a Table Visualization](#)
- [Symphony Metric Sets](#)

Displaying Top/Bottom Records

This applies to: *Managed Dashboards, Managed Reports*

You can filter your [metric set](#) to display the top or bottom 'N' number of items, or group the smallest or largest items together into an 'other' group (with a customizable caption). Any other filters are applied first before filtering or grouping the top/bottom items.

For example, you can decide to display only the top 5 values, or only the top 4 values plus an *Other* value that groups and aggregates all the smaller values together.

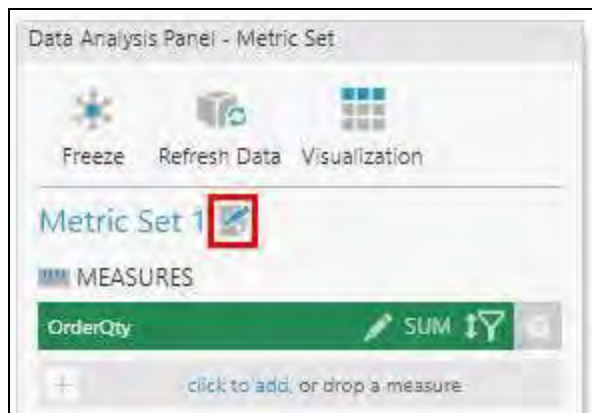


Note: Another option is to use the [TOP](#) or [BOTTOM](#) subset function in a [formula metric set](#).

Set the Top / Bottom Measure

If you want to limit or filter your result to a certain number of items based on a measure's values, you can set up a top/bottom measure.

In the *Data Analysis Panel*, click the **Edit** icon.



Set **Top/ Bottom Measure** to the measure you want to determine the top or bottom records.

Set **Top and/or Bottom** to the number of top or bottom items or values you want to display.

Metric Set Default Values

Transpose Result

☐

Prevent Data Export

☐

Shown Totals

Both

Data Retrieval Format

Standard

Top / Bottom Measure

SalesYTD

Top

5

Bottom

0

Tie Policy

None

Other Group Measure

(None)

Items with the same value are "tied". You can decide how to handle these values:

- Select *Exclude Ties* to display exactly the number of items you set, even if some values are tied.
- Select *Include All Ties* to display all of the items with the top/bottom 'N' values, e.g., the top three values, even if there are four items because two items are

tied.

- Select *Include All Ties In Last Position* to display only the number of items you set, plus any other items with the same value as the last item.

Click the submit button at the bottom of the dialog to apply the settings.

Name	SalesYTD
Grand Total	36,160,590.0807
Australia	5,977,814.9154
Canada	6,771,829.1376
Northwest	7,887,186.7882
Southwest	10,510,853.8739
United Kingdom	5,012,905.3656

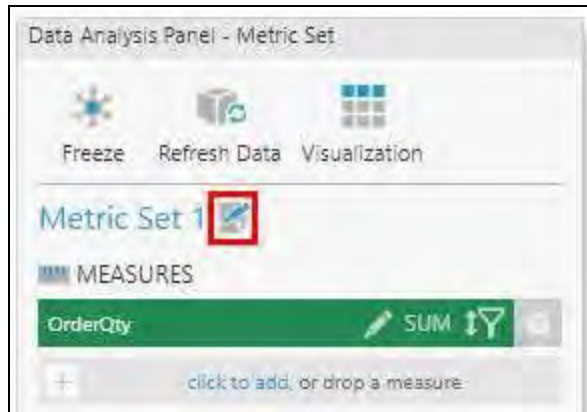


Note: This setting does not sort the values. You may now want to [sort](#) by the same measure.

Set the Other Group Measure

You can set up an 'other' group to group multiple rows, columns, or data points together into one if their measure value meets certain criteria. This can be set up together with a top/bottom measure as described above, or on its own.

In the *Data Analysis Panel*, click the **Edit** icon.



Set **Other Group Measure** to the values you want to determine the items to group together.

Other Group Measure	
OrderQty	
Hierarchy	
Description	
Caption	
Category	
Bottom	
Upper Boundary Category	
Percentage of Total	
Upper Boundary	
	2%

In case there are multiple hierarchies in your metric set, set **Hierarchy** to whichever displayed hierarchy's values should be grouped together into one.

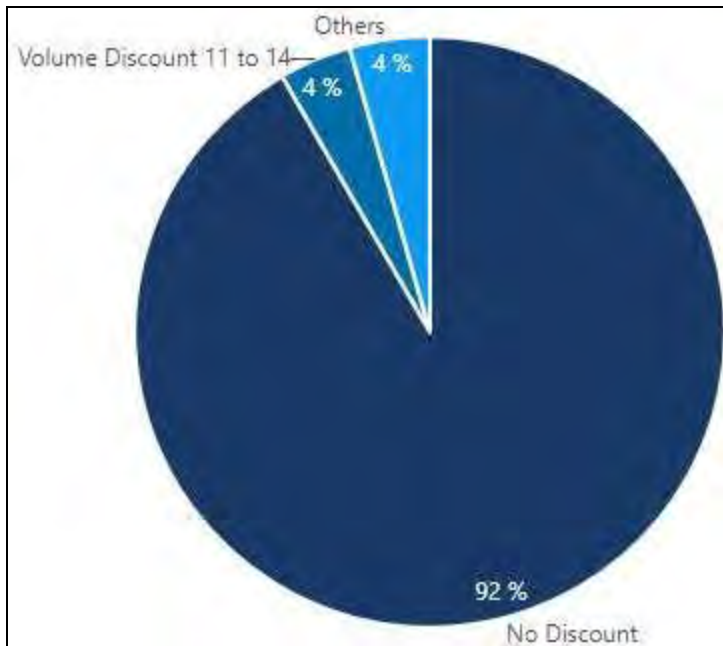
Set **Caption** to whatever the new group should be called, e.g., *Other*.

Set **Category** to determine if the smallest (*Bottom*), largest (*Top*), or values closest to the middle should be grouped together.

Upper/Lower Boundary Category allows you to choose whether to group items together based on:

- Items with values above or below a *Fixed Value*
- The items with values within the top or bottom *Percentage of Total*
- A specified *Number of Items*

Click the submit button at the bottom of the dialog to apply the settings.



When using an OLAP data source, the grouped value may be missing until you assign the measure an [aggregator](#).

For more information, see:

- [Select and Group Data for a Metric Set](#)
- [Group Members During Analysis](#)



- Archive of documentation for Logi Symphony v24

- [Sort and Filter Metric Data](#)
- [List of Formulas](#)
- [Symphony Metric Sets](#)

Exclude Data Points from the Visualization

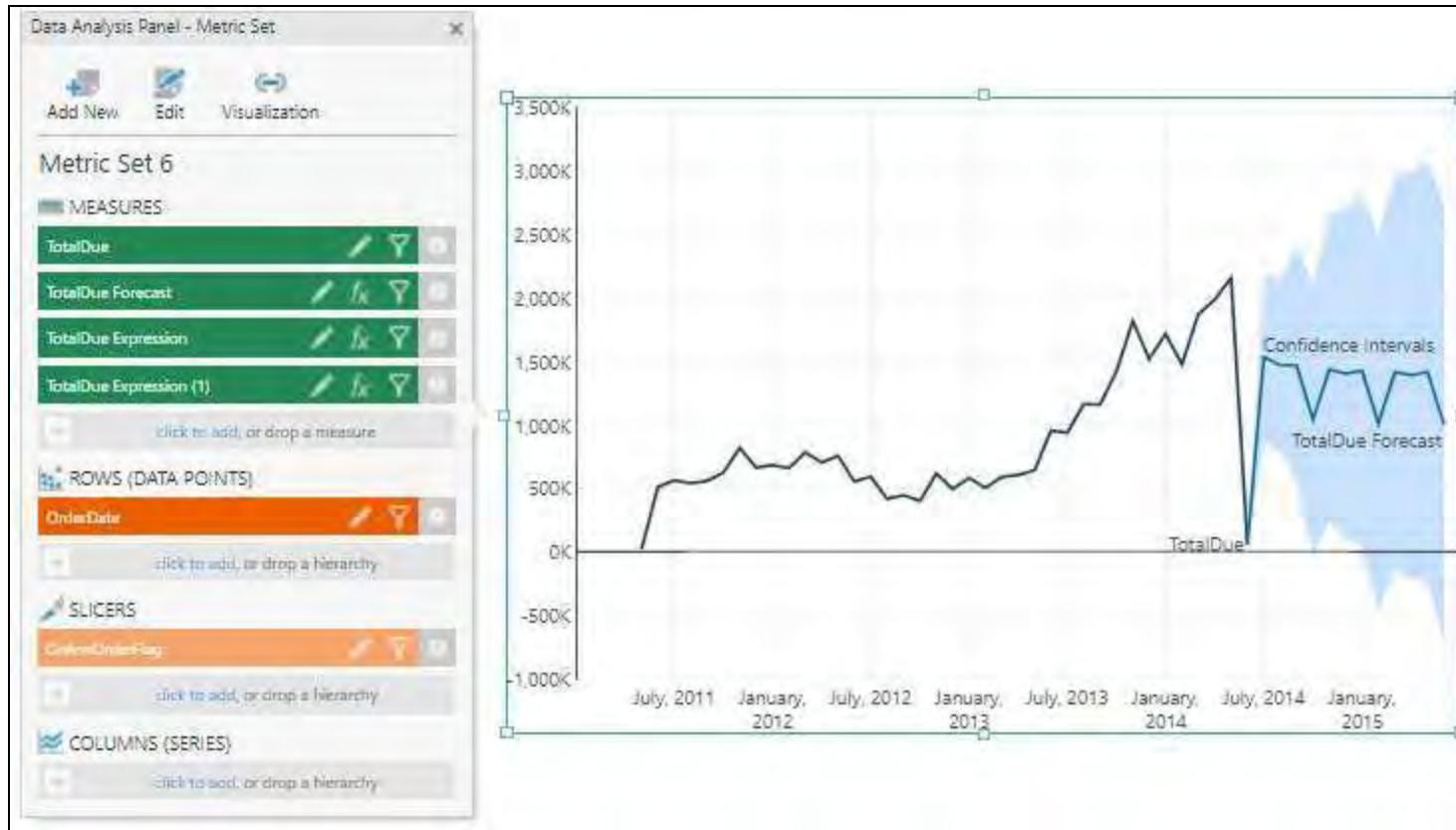
This applies to: *Managed Dashboards, Managed Reports*

Exclude select data points from the visualization without introducing filters or parameters, and optionally select multiple data points to keep or filter out at the same time. One common use case for this functionality is when you may want to omit some outlying points.

Example preparation

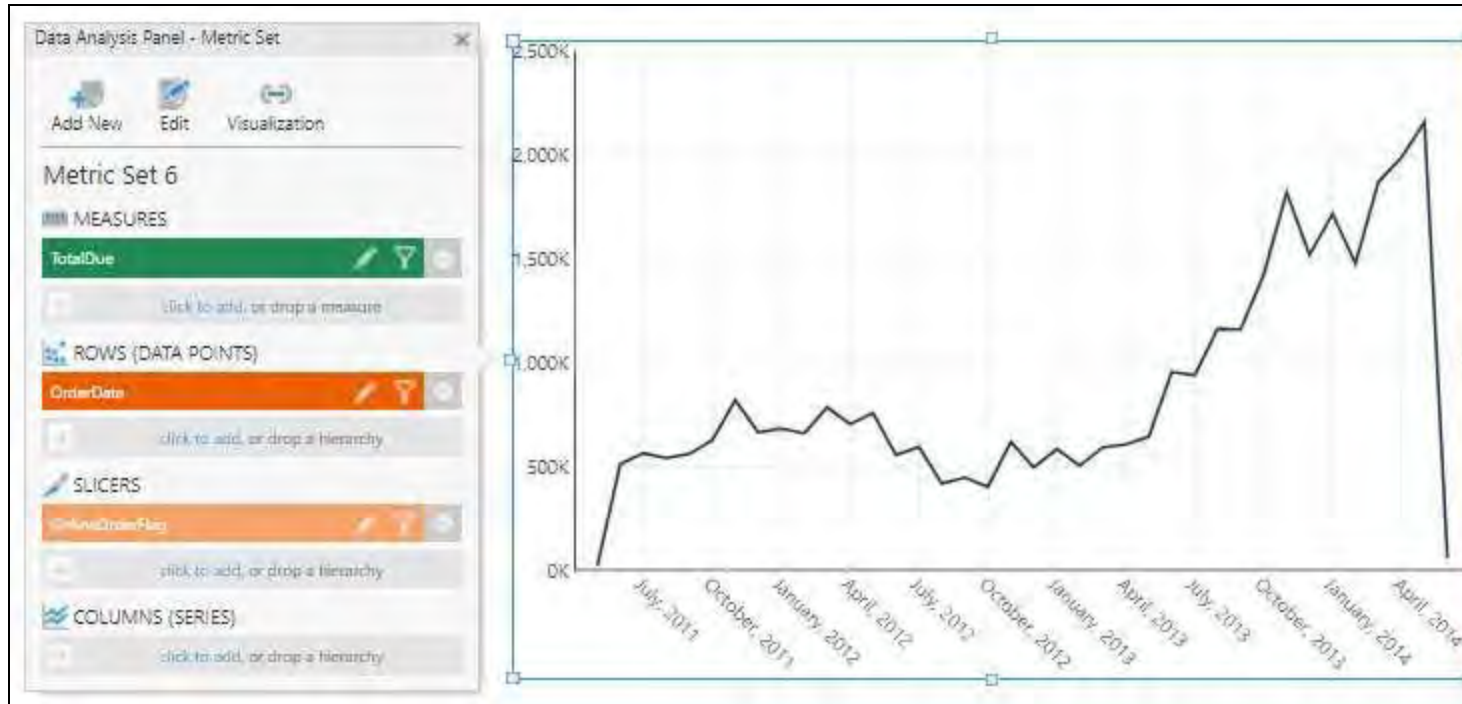
Any data displayed by any visualization can be filtered the same way as shown in this article, but for the example shown, a line chart was added to a dashboard by choosing **Data Visualization** from the toolbar and then **Line** (under Charts).

From the *[Sales].[SalesOrderHeader]* table of the [Adventure Works database](#), *TotalDue* and *OrderDate* (expanded to a time dimension's Month level) were dragged onto the chart, and *OnlineOrderFlag* to [Slicers](#) in the Data Analysis Panel. The slicer was then filtered to show only *True* values (online only).



The **Selected Point Style** property on the TotalDue series has also been customized in the screenshots below to a distinct *DimGray* fill.

A forecast was added by choosing **Re-Visualize** in the toolbar and then the recommended **Forecast Analysis** option.



Exclude Selected Data Points

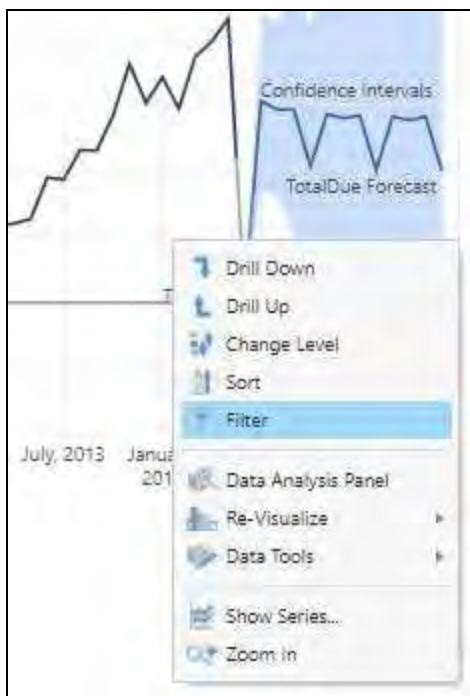
Switch to **View** mode if you are currently editing a view such as a dashboard, so that you can interact with the data directly.

In this example using a chart, we can optionally select multiple *TotalDue* points at once by dragging a selection box around them, for example outlying points that are likely misrepresenting the trend.

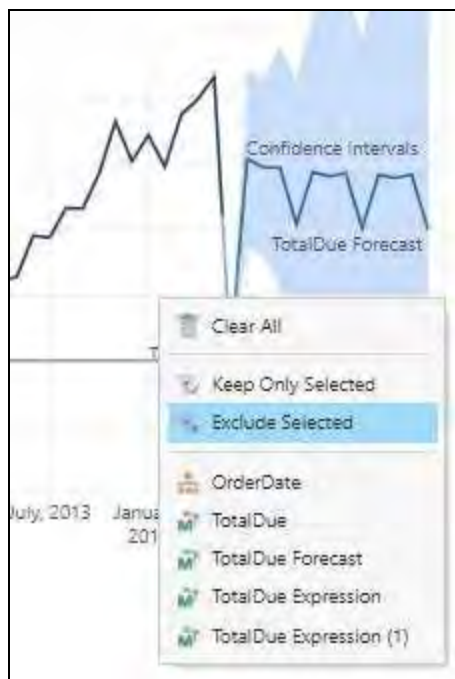


Note: You can select multiple groups of points in a chart by holding the [Ctrl or Shift key](#) on the keyboard when dragging additional selection boxes. For a table, hold one of these keys while clicking multiple rows or columns to select.

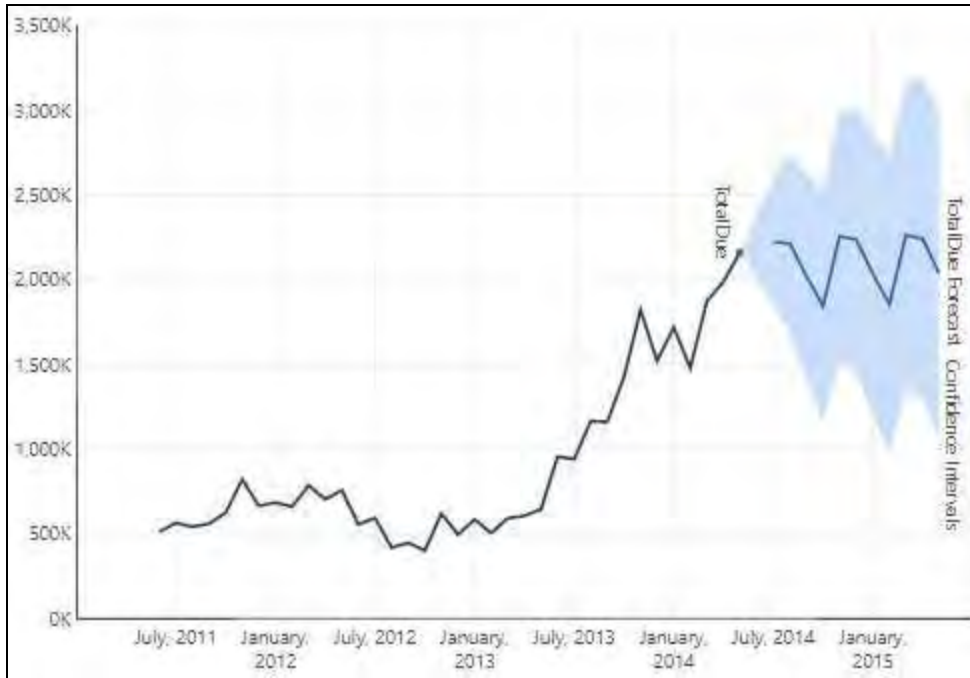
Right-click (or long-tap) the visualization and select **Filter**. If you have just selected data as described above, this will present filtering options based on the selected data. Otherwise, the filter options will be based on the data you just right-clicked.



Select **Exclude Selected**.

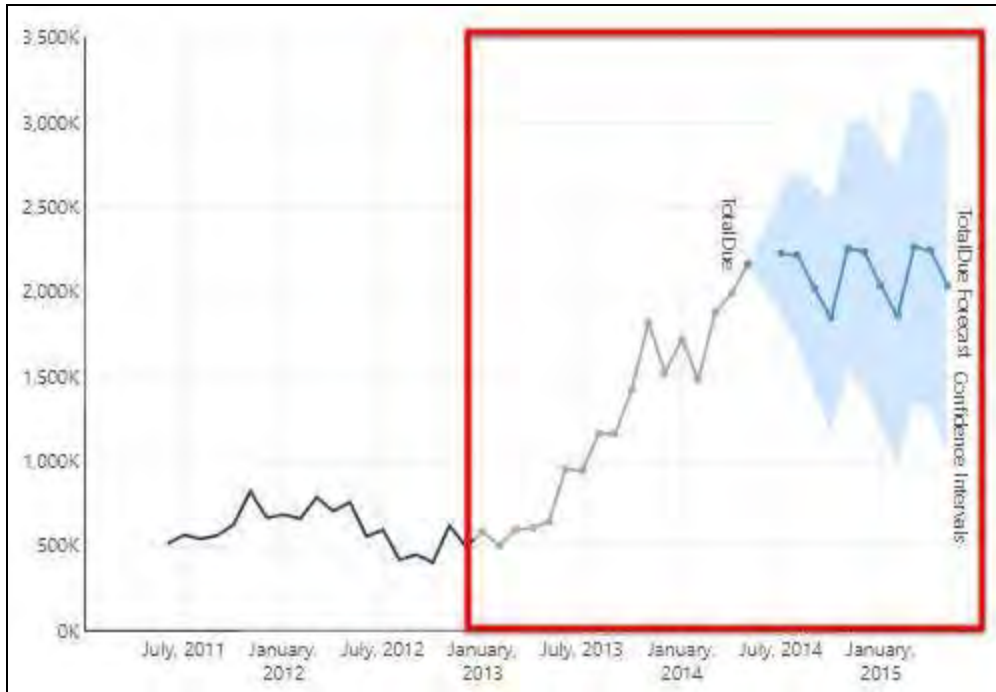


The selected points are excluded from the data visualization and the forecasting is recalculated.



Keep Only Selected Data Points

You can optionally select a group of multiple data points if you want to focus on them, for example dragging a selection box all the way from *January 2013* to the top right corner of the chart in this example.

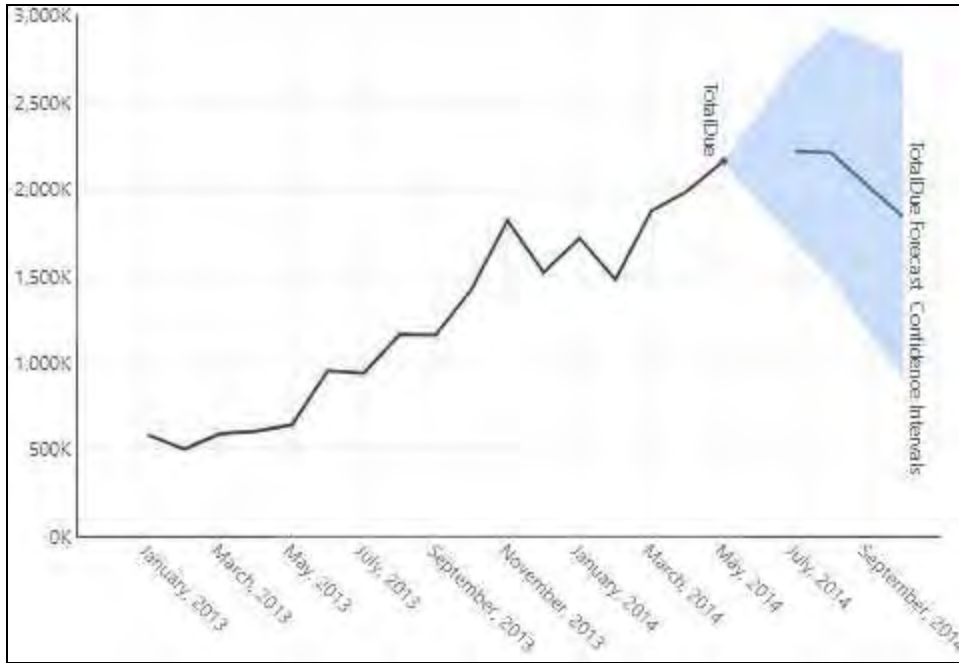


Note: You also need to include the forecasting if you don't want it excluded.

Right-click (or long-tap) the data visualization, select **Filter**, and select **Keep Only Selected**.

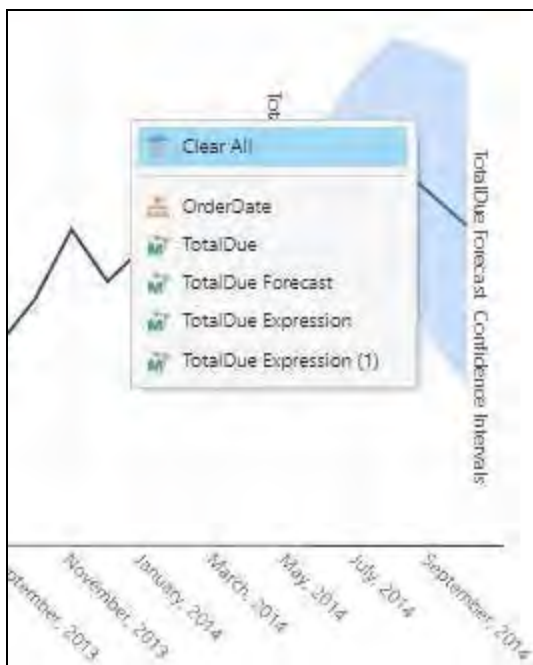


Everything other than the selected points (or else the one you right-clicked) is excluded from the data visualization and the forecasting is recalculated.

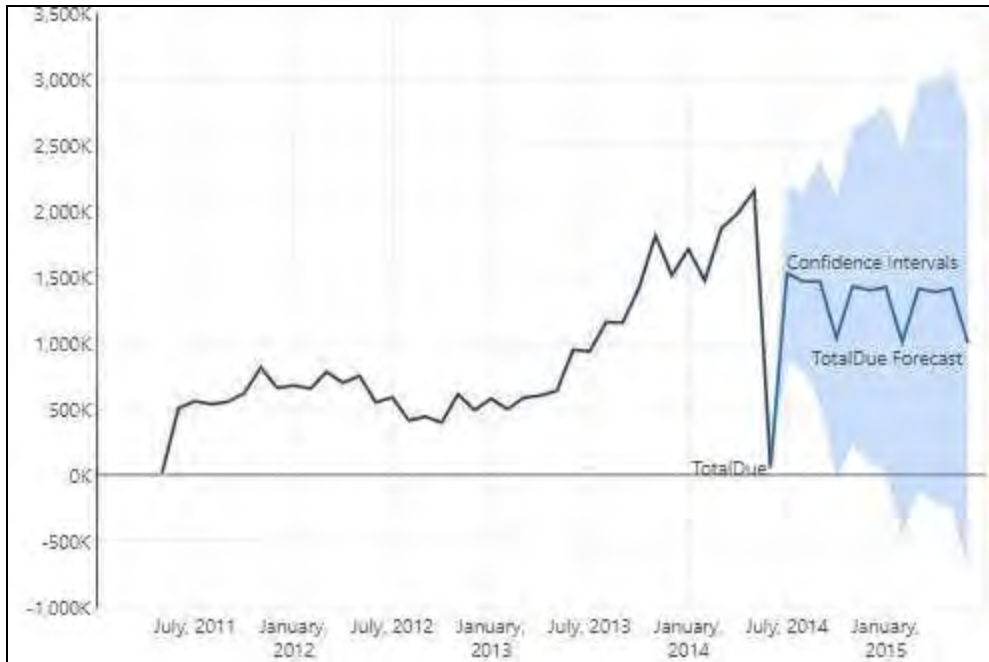


Clear All Filters

Right-click the data visualization, select **Filter**, and select **Clear All**.



All of the filtering has reverted back to the originally prepared data.



For more information, see:

- [Symphony Metric Sets](#)
- [Create a Metric Set and Add a Filter](#)



Expand and Collapse Hierarchy Members

This applies to: *Managed Dashboards, Managed Reports*

This article shows you how to use the expand and collapse functionality to see more detailed breakdowns or step back to a summary view.

These options are introduced along with other functionality in [Symphony Metric Sets](#). Additional examples are shown here for tables, charts, and treemaps, and the same functionality is available for relationship diagrams, including the [tree diagram](#).

Expand and Collapse

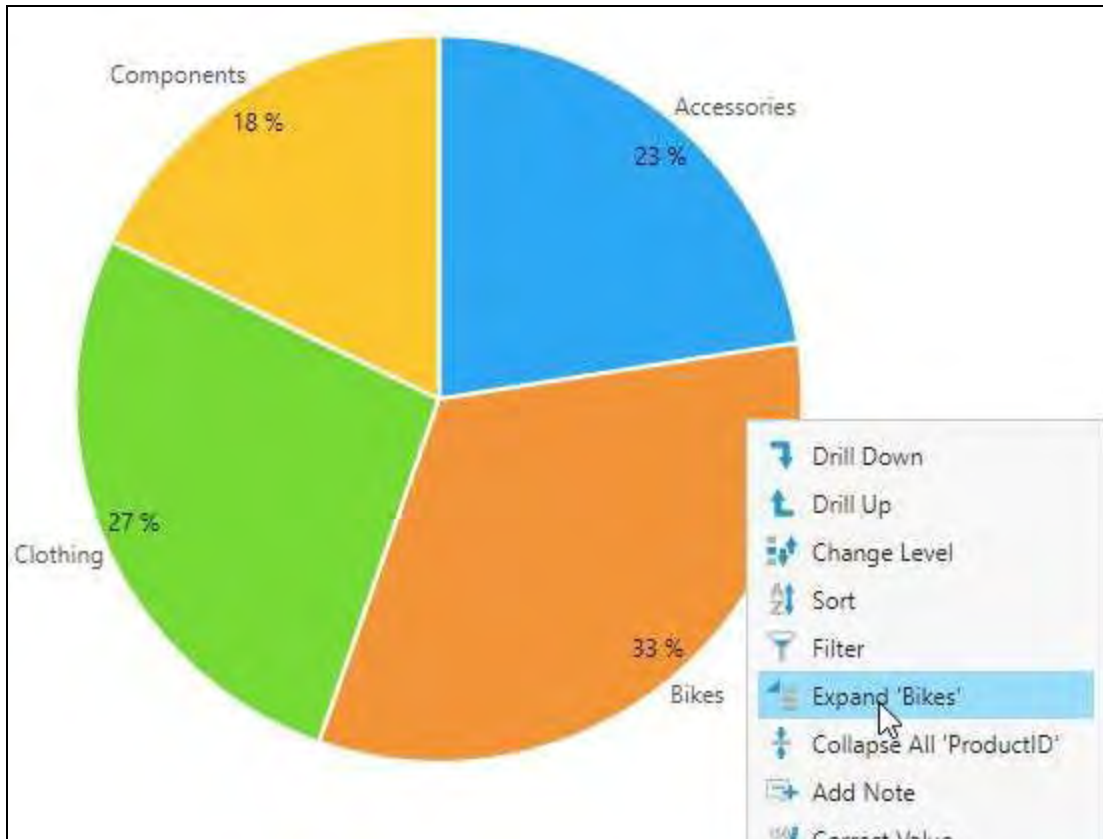
Expanding a member is similar to drilling down in-place but without filtering out other members. In the same manner, collapsing a member is similar to drilling up in-place.

These options are available when selecting data using a multi-level hierarchy, or when multiple hierarchies are selected in the Data Analysis Panel.

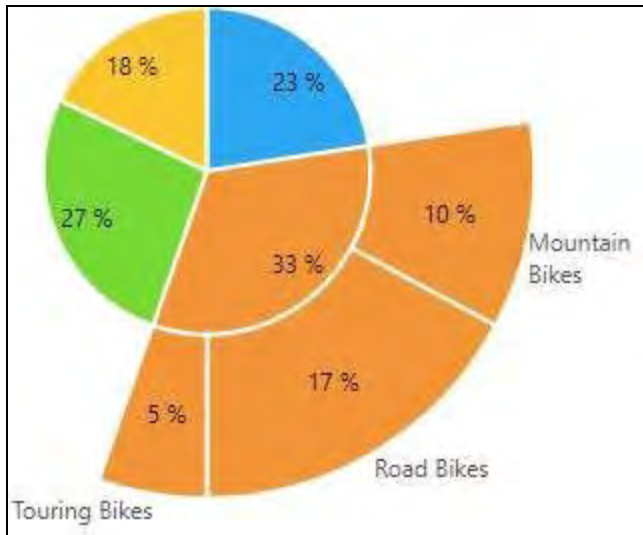
Charts

All chart types support the Expand & Collapse options, which are available when you right-click (or long-tap) a data point to open the context menu.

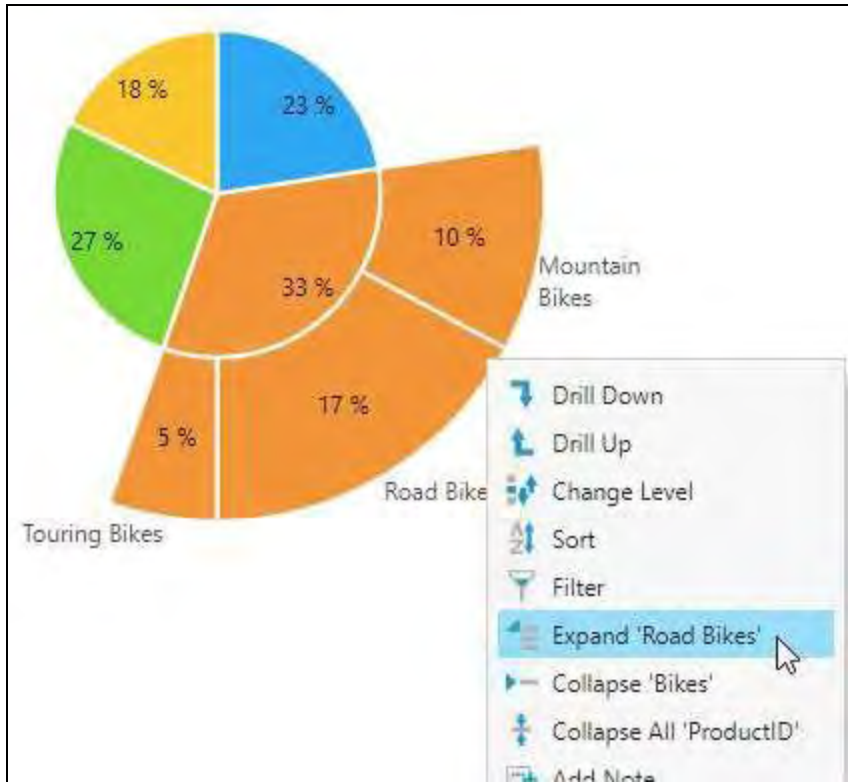
For example, right-click the *Bikes* member in this pie/sunburst chart, and then select **Expand**. (Switch to view mode on a dashboard first.)



The chart now shows an exploded view of that member but the other members are not filtered out. They are still displayed in order to provide context.



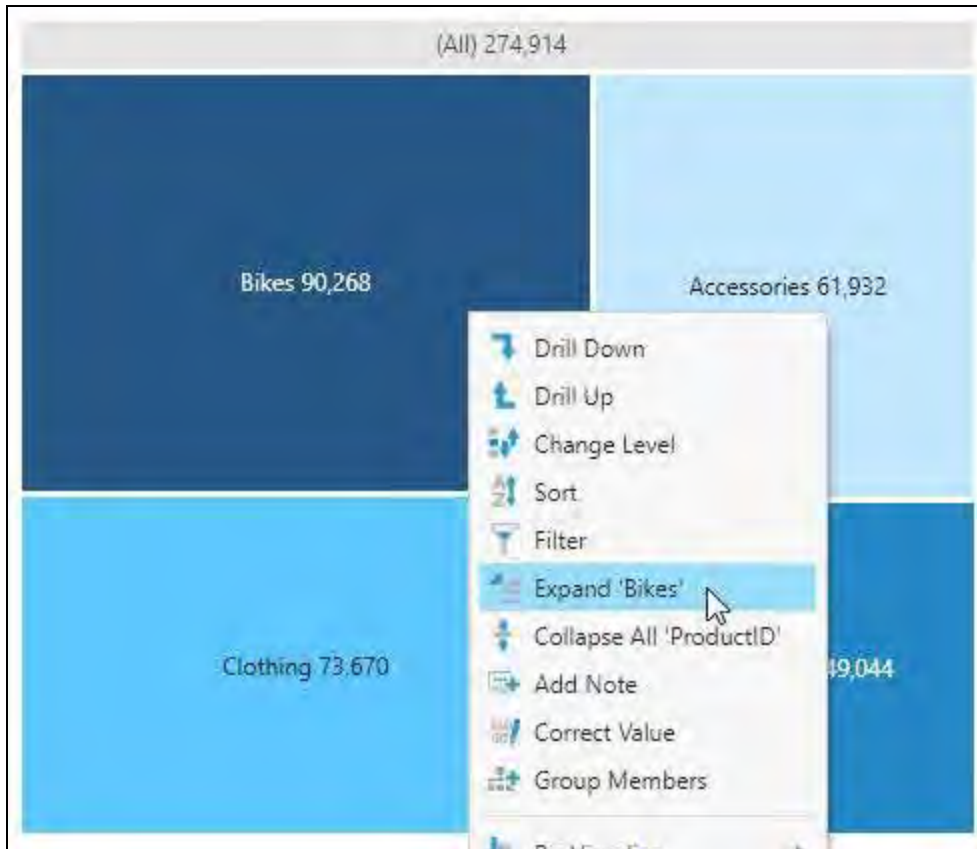
You can continue expanding members, or collapse members using the **Collapse** option.



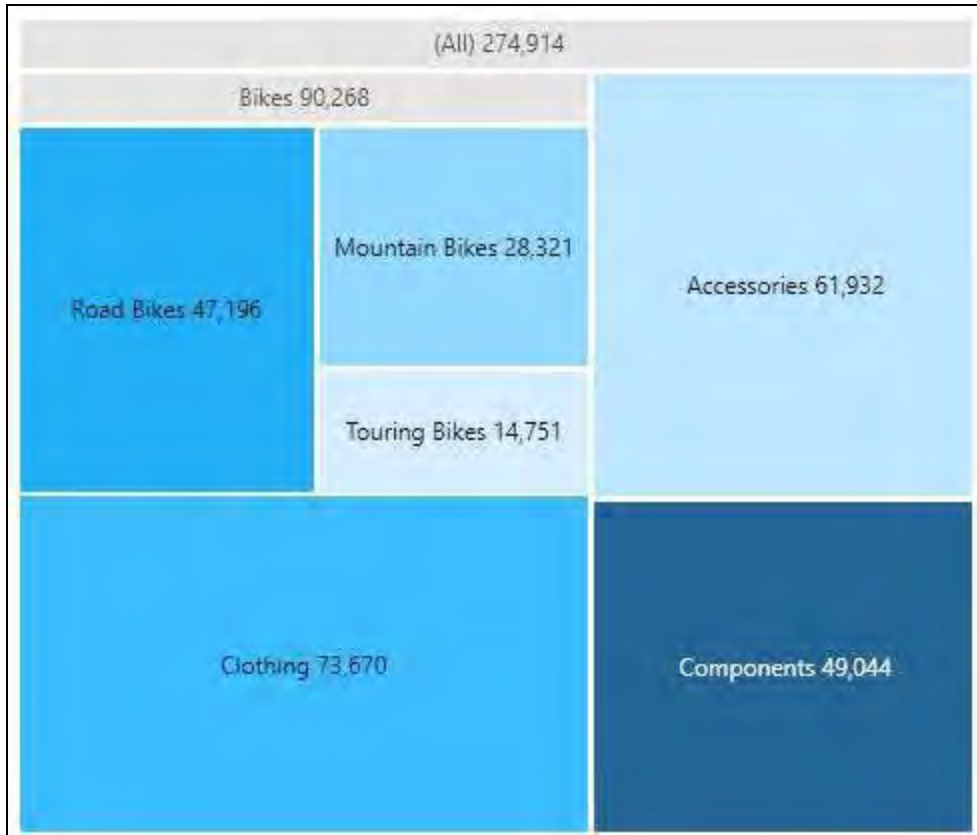
Charts can also display expander buttons similar to those in tables when there is a [categorical axis](#) displayed.

Treemaps

Expand and collapse works similarly for a treemap visualization. Just right-click a treemap rectangle and select **Expand**.



This breaks that value down into the next level's values while leaving the other rectangles unchanged.



You can right-click again and expand further, or click **Collapse** to go back to the original result. See [Using a Treemap](#).

Tables

Table visualizations support expand and collapse functionality through the use of expander buttons (triangles) in row header and column header cells. These are toggle buttons that can be used to expand or collapse. Alternatively, you can right-click (or long-tap) a cell and use **Expand** or **Collapse** from the context menu.

		2011	2012				
		January, 2012	February, 2012	March, 2012	April, 2012	May, 2012	
Product Category	Product Subcategory	OrderQty	OrderQty	OrderQty	OrderQty	OrderQty	OrderQty
Accessories		1,032	216	88	277	212	721
Bikes	Mountain Bikes	2,621	925	229	631	501	744
	Road Bikes	5,342	1,882	718	1,291	932	1,840
	Touring Bikes						
Clothing				947	1,922	1,433	2,584
Components				234	555	429	2,455
				173	430	331	1,963
Grand Total				1,442	3,184	2,405	7,723

Drill Down

Drill Up

Change Level

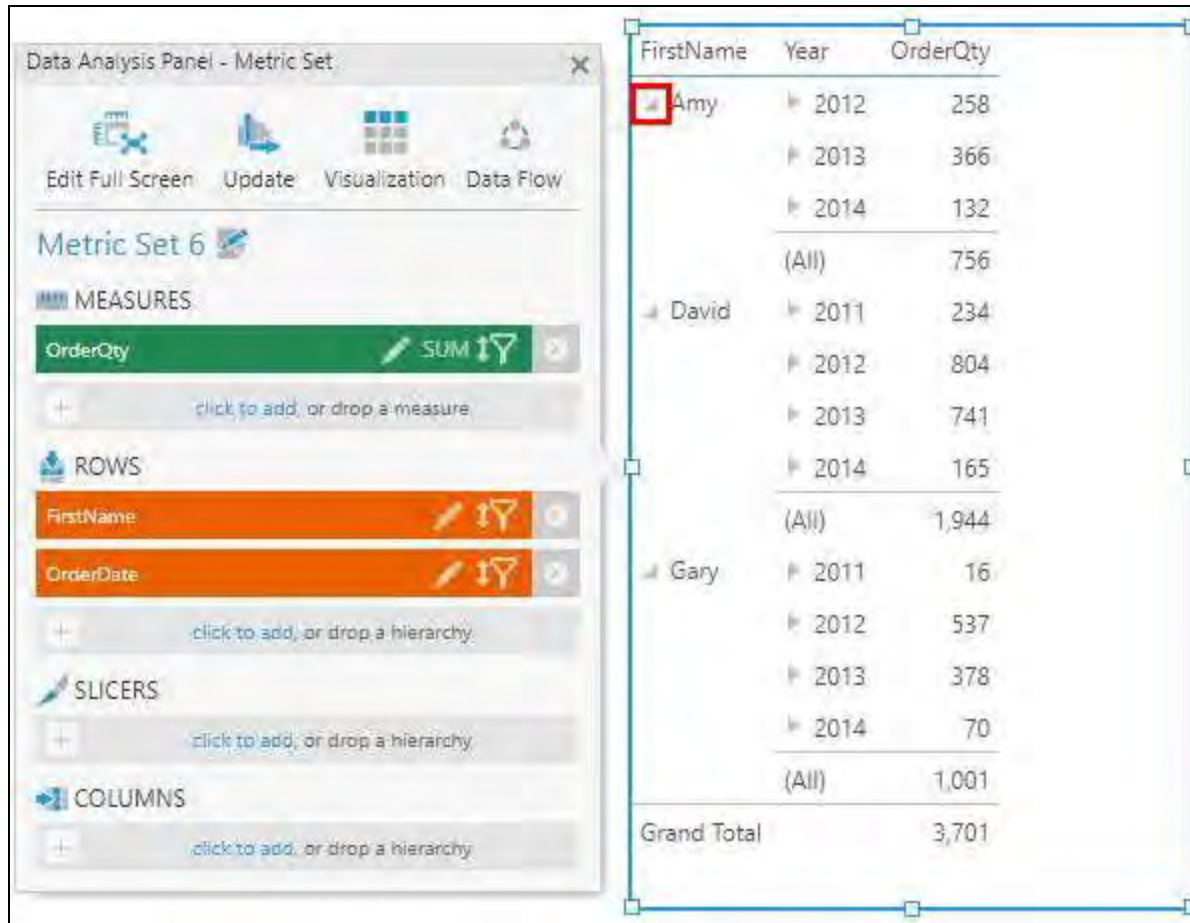
Sort

Filter

Expand 'Road Bikes'

Collapse All 'ProductID'

Group Members



These expander buttons allow you to collapse the members below/to the right even though they belong to a separate hierarchy.

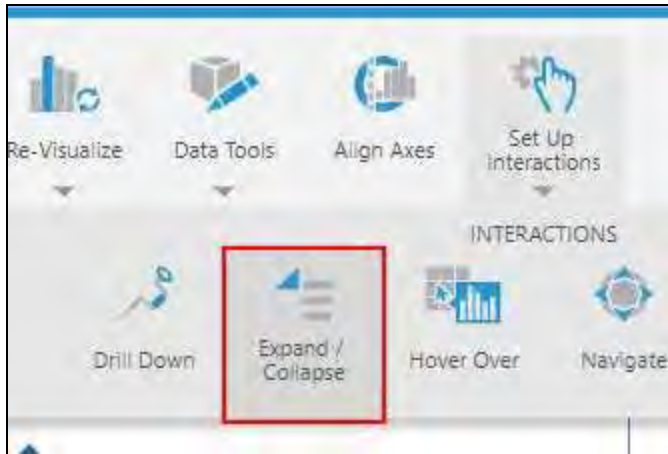
FirstName	Year	OrderQty
▶ Amy	(All)	756
▶ David	▶ 2011	234
	▶ 2012	804
	▶ 2013	741
	▶ 2014	165
	(All)	1,944
▶ Gary	▶ 2011	16
	▶ 2012	537
	▶ 2013	378
	▶ 2014	70
	(All)	1,001
Grand Total		3,701

You can customize how or whether expander buttons are shown in the [table's properties](#).

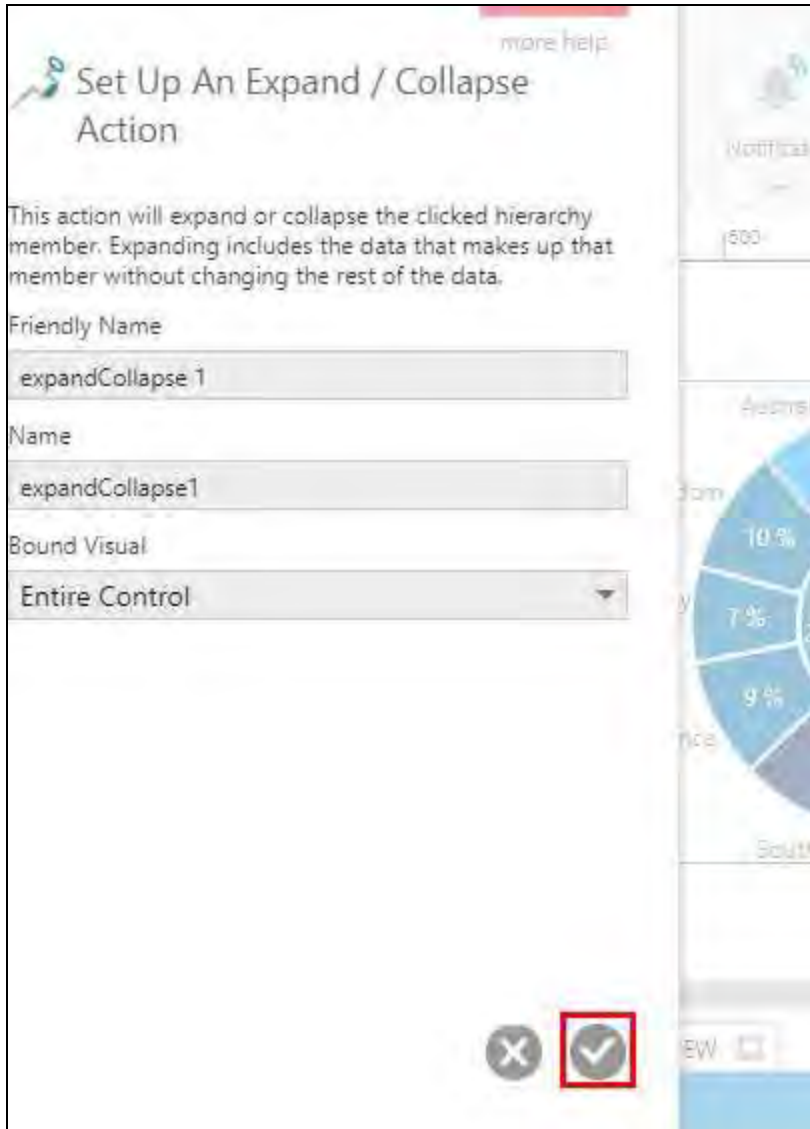
Expand/Collapse On Click

If you want to be able to simply click on data to expand or collapse it, you can easily set up this interaction for any visualization that supports expanding and collapsing.

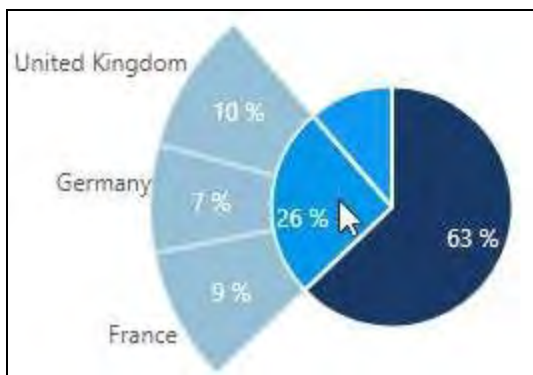
When editing a dashboard or another view with your metric set added, select the visualization and choose **Set Up Interactions** in the toolbar, then choose **Expand/ Collapse**.



In the dialog that appears, you typically don't need to change any settings, and you can click the **Submit** button at the bottom to confirm.



Choose **View** in the toolbar, and click a data point. If there is data in a hierarchy or level below, it will be expanded; otherwise, it will be collapsed into the hierarchy or level above.



Collapse All

Every hierarchy has an *All* or *Grand total* value above the first level of the hierarchy. You can use the **Collapse All** option of the context menu to collapse all of the values of the hierarchy you clicked all the way up to this level. This can be used to leave out the details of that hierarchy and view the data broken down only by other hierarchy values.

When editing, you can also set a hierarchy to be collapsed this way initially, as follows:

From the *Data Analysis Panel*, edit the hierarchy.





Configure 'ProductID'

Here you can quickly access the visual settings associated with this element, or configure specific settings within the metric set.

VISUALIZATION

 TABLE: COLUMN HEADER

 TOOLTIPS: ORDERQTY COLUMN

METRIC SET DEFAULT VALUES

Caption

Unique Name

Analysis Element Name

▼ Element Settings

Unknown Member Caption

Collapse All

☐

Shown Totals

Click **Submit**. These values will now initially be collapsed to into a single *All* or *Grand Total* value. You or viewers can use the **Expand All** option from the context menu to view the individual values again.

Grand Total	
Year	OrderQty
▶ 2011	11,687
▶ 2012	65,836
▶ 2013	102,829
▶ 2014	34,164
Grand Total	214,516

Drill Down
 Drill Up
 Change Level
 Sort
 Filter
 Expand '2012'
Expand All 'ProductID'
 Collapse All 'OrderDate'
 Group Members

Working with Hierarchies

- Expanding and collapsing requires the [Change Data privilege](#). This can be disabled [for users or groups, on the dashboard/view, or on the visualization](#).
- The **Collapse All** option may cause a pie chart to disappear if **Hide Grand Total** is selected in the chart properties. You will still be able to use **Expand All** from the context menu.
- Unknown** indicates data outside of the hierarchy. This is not a member of the hierarchy and cannot be expanded. You can see data under Unknown by selecting an additional hierarchy or by changing the hierarchy level.
- Filters applied to the standard hierarchy will be ignored on the expanded members. The filtering will still be maintained on the aggregated hierarchy levels.



Formatting Text

This applies to: *Managed Dashboards, Managed Reports*

You can format how data is displayed as text in visualizations or their tooltips:

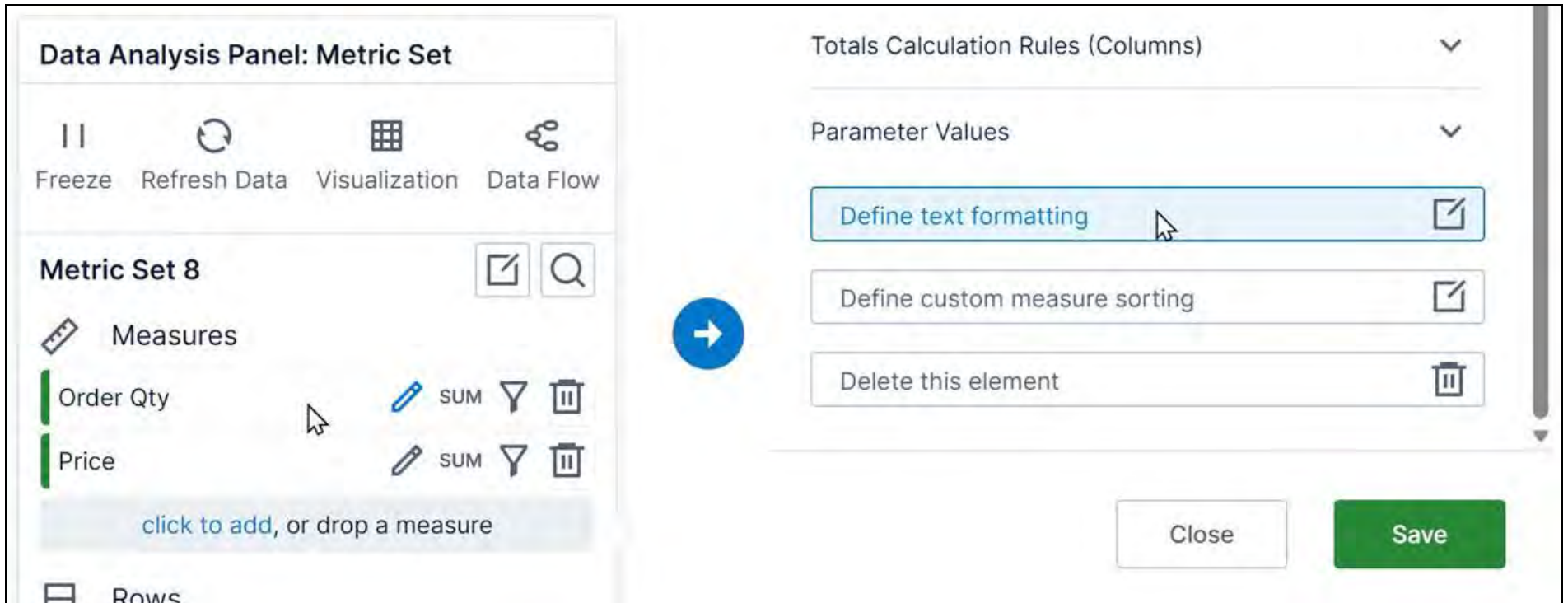
- [Metric sets](#) determine how to format numbers (measures) and unknown members, for example the number of decimal places.
- [Time dimensions](#) have customizable formatting for each time period/level.
- [Visualizations](#) can combine multiple values together with predefined text and also change number and date/time formats.

Related video: [Using The Data Analysis Pan](#)

Metric Set Element Format

You can format a measure displaying numeric data by changing the metric set while it's [checked out](#) to you.

In the Data Analysis Panel, click the green tile for your measure to edit its settings, then scroll down and click **Define text formatting**.



This opens the *Configure Formats* dialog, containing multiple formatting options.

To set up the main formatting, click to **Format** button, which displays the measure's current format as its text (e.g., *N2*).

Help 

Configure formats

You can customize how the numbers are formatted into text. Click to edit the main Format, or choose from other options.

Use Source Formatting



 Format will be determined by the data source.

Format

Set format 

Symbol

Symbol Placement

Before 

Additional Format Settings 

Thousands Format



Help 

Configure format

Choose from standard format types or set a custom format for more flexibility.

Format Type

Number 

General

Currency

Exponential

Number 

Percent

Custom

0  

1000s Separator



Sample Preview 

Using the **Format Type** drop-down in the **Configure Format** dialog:

- Choose a standard format type like **Number** or **Currency** with options such as the number of decimal places. These are designed to automatically display numbers as expected for each user's culture.
- Or choose **Custom** for more advanced formatting with more control over the result. Enter any custom format or choose one of the common examples from the drop-down menu to the right. (Thousands and decimal separator characters are always determined by the user's culture and not changed by any format settings.)



Configure Format

Here you can quickly configure the format to be applied to the value.

Format Type

Custom

Displays the value using the custom format entered below. This allows for greater control due to many customization options.

Custom Format

#,0,K

Sample Preview

Positive format	1K
Negative format	-1K
Thousands format	14K
Millions format	13 868K
Billions format	13 868 200K
Trillions format	13 868 200 000K

One non-zero decimal place (0,0,#)

Thousands, no decimal places (#,0,K)

Millions, one decimal place (#,0,.0M)

No thousands separator (0)

Negative parentheses (0#,0,00;(#,0,00))

Plus and minus signs (+#,0;-#,0;0)

more help

Data Analysis Pane

Re-Measures

Other Tools

700 750 800 850 900

Product Category	LineTotal
Accessories	\$1 272 072.88
Bikes	594 651 172.70

Some common examples of formats are listed below. For more details, see .NET [standard numeric format strings](#) and [custom numeric format strings](#).

Format Type	Format	Sample Text
Numeric with 5 decimal places	N5	80,487,704.17919
Currency with 2 decimal places	C2	\$80,487,704.18
Percent (multiplied by 100) with 0 decimal places	P0	8,048,770,418%
Exponential/scientific with 2 decimal places	E2	8.0SE+007
Custom numeric (1 non-zero decimal place)	0,0.#	80,487,704.2
Custom numeric (thousands)	#,0,K	80,488K
Custom numeric (millions with 1 decimal place)	#,0,,.0M	80.5M



Note: You can also set the format for a measure in a [data cube](#) ahead of time so that it takes effect by default in every metric set where it's used.

Formatting Currency

When using a currency format on its own, the user's [language and culture settings](#) determine which currency symbol to use and where to place it.

Use Source Formatting

☐

Format

C2 

Symbol

Symbol Placement

Before 

 Placement will be determined by the session's culture.

Price
962 716,74 €
2 044 600,00 €
1 639 840,11 €
1 358 050,47 €
2 868 129,20 €
2 458 472,43 €
1 309 863,25 €
2 451 605,62 €
2 099 415,62 €
1 546 592,23 €
2 942 672,91 €
1 678 567,42 €
2 894 054,68 €



Important: Allowing the currency symbol to change automatically in format settings does not change the numeric values. Unless you store or convert to different currencies elsewhere, users with different cultures may see the same numeric value with different currency symbols unless Symbol is set.

Normally, your values are in a specific currency and you should enter the corresponding **Symbol**. It will be placed according to each user's culture, but will be displayed for all users.

Format	
C2	1 639 840,11 \$
	1 358 050,47 \$
	2 868 129,20 \$
	2 458 472,43 \$
	1 309 863,25 \$
	2 451 605,62 \$

If you prefer to format a value with a specific symbol placement that does not change based on culture, use a non-currency format such as Numeric or 0.00, and specify the placement using the **Symbol Placement** setting.

Format	
N2	\$1 639 840,11
	\$1 358 050,47
	\$2 868 129,20
Symbol	\$2 458 472,43
\$	\$1 309 863,25
	\$2 451 605,62
Symbol Placement	\$2 099 415,62
Before	\$1 546 592,23
	\$2 942 672,91

Formatting Large Values

Numeric values can be further formatted into thousands, millions, billions, or trillions units automatically according to the magnitude of the values as they change. For example, to display a value larger than one million with an 'M' symbol following an appropriately divided value, enter *OM* into the **Millions Format** field. The '0' is a placeholder for the corresponding number of millions.

Symbol Placement

Before

i Placement will be determined by the session's culture.

Additional Format Settings

Thousands Format

OK

Millions Format

0 Million

Billions Format

Trillions Format

Format Large Values Separately



Qty	Price
366	\$962.72K
209	\$2.04 Million
358	\$1.64 Million
403	\$1.36 Million
132	\$2.87 Million
480	\$2.46 Million
340	\$1.31 Million
303	\$2.45 Million
841	\$2.10 Million
467	\$1.55 Million
179	\$2.94 Million
418	\$1.68 Million
755	\$2.89 Million
325	\$4.15 Million
366	\$3.24 Million
584	\$2.22 Million
268	\$3.39 Million
672	\$2.76 Million
914	\$109.85 Million



Note: Chart axes and gauge scales do not display measure values directly but their own reference values. They have their own versions of these properties that can be set in the Properties window.

If the **Format Large Values Separately** option is selected, numbers in different rows or columns can be formatted into different magnitude units as they vary. When this option is unchecked, all numbers displayed at one time use the same unit for easier comparison.

Trillions Format

Format Large Values Separately
☐

Range Format Overrides

Price

\$962.72K

\$2,044.60K

\$1,639.84K

\$1,358.05K

\$2,868.13K

\$2,458.47K

Formatting Ranges

You can assign different formats to different ranges of values, for example to reduce the number of decimal places for smaller numbers.

To format a specific range, click to expand **Range Format Overrides** in the *Configure Formats* dialog, then click **Add range format override**.

Use Source Formatting

☐

Format

C2

Symbol

\$

Symbol Placement

Before

Placement will be determined by the session's culture.

Additional Format Settings

Range Format Overrides

Add range format override

+

Close

Save

Configure range override format

Configure the formats to be applied to the values matching the specified range.

Format

C0

Symbol

\$

Symbol Placement

Placement Will Be Determined By The Session's Culture.

Before

From

1000

:

To

Open Range Boun...

Additional Format Settings

Close

Save

In the *Configure Range Override Format* dialog for each range, all of the format options described in previous sections are available. Set **From** and **To** to determine the value range this formatting is applied to.

Element Settings

Unknown Member Caption

N/A

Collapse All

☐

Shown Totals

Millions Format

Billions Format

Trillions Format

Format Large Values Separately ☒

Name	Discontinued
Adjustable Race	N/A
All-Purpose Bike Stand	N/A
AWC Logo Cap	N/A
BB Ball Bearing	N/A
Bearing Ball	N/A
Bike Wash - Dissolver	N/A
Blade	N/A

Time Dimension Formatting

When working with date & time data, it's common to apply a time dimension to group by time periods such as *Year* and *Month*, drill down, and more. Time dimensions can be applied to your data [in a data cube](#) or [in a metric set](#).

Time dimensions also have formatting options you can customize to change how the dates & times are formatted as text when displayed in your metric sets and visualizations. See [Create a Time Dimension](#).



Placeholder Keywords

Various text and tooltip properties of data visualizations can be set with placeholder keywords that will be replaced with actual values from your data when displayed. These property settings are often prepopulated for you using the data assigned in [Properties](#), but you can optionally customize this property yourself for more options, more control over the formatting, or to combine the keywords with descriptive text.

When you click into one of these properties in the Properties window, a popup will appear suggesting all of the available keywords described below. Click one to insert it into the property at your current position in the text.

X

Explore

X

Layers

X

Sales Amount

< chart 1

Q Search Properties

⚙ Main

🎯 Look

📐 Layout

T Text

Text

Series Label

Tooltip Text

Date: [Date]

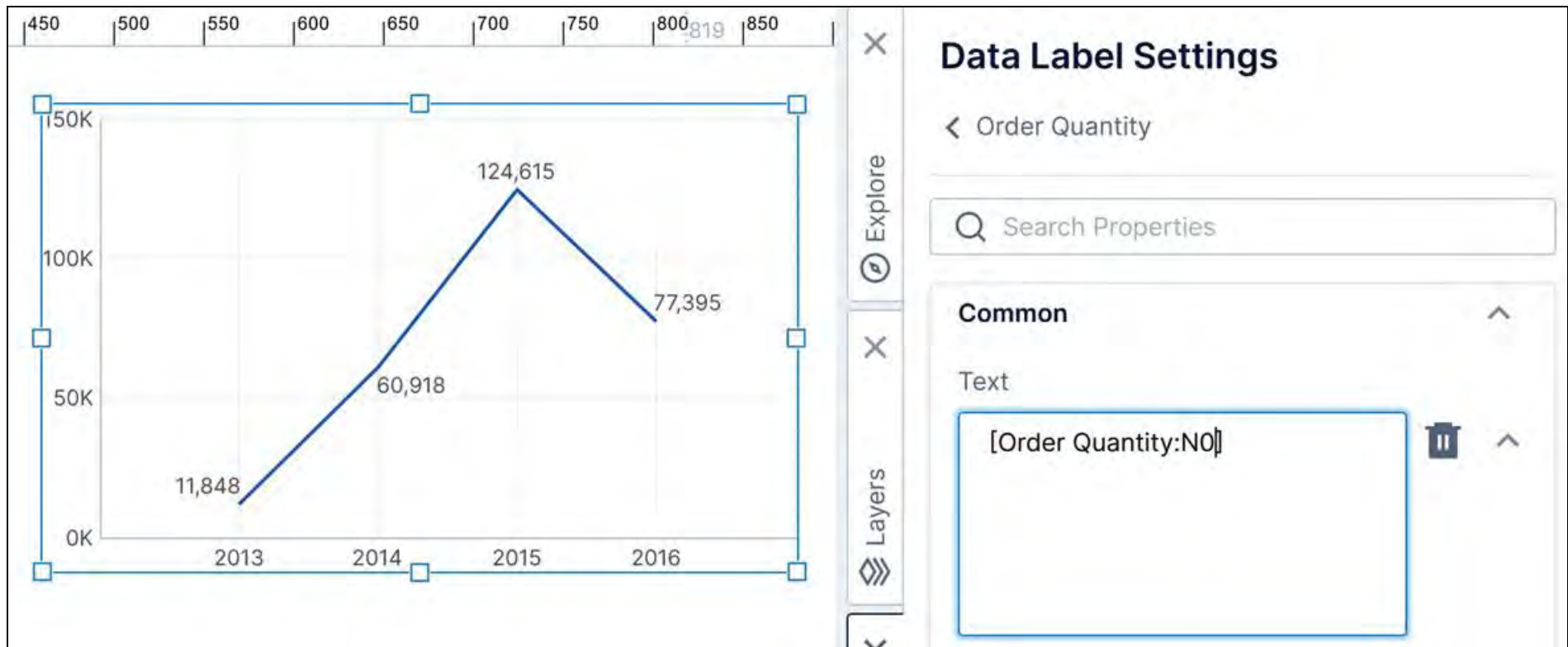
Sales Amount: [Sales Amount]

You can insert multiple keywords and combine them with any other text or characters you choose to type in.

The most common keywords refer to a measure or hierarchy from the Data Analysis Panel to display its values, enclosing its caption in square brackets. For example: [Order Quantity], using the format defined in the [metric set](#) or the [time dimension](#) by default.

Formats

You can optionally edit the keywords used in the Properties window to customize the formatting for numbers or dates/times to be different than set up in the metric set or [time dimension](#). The figure below shows how to format the Order Quantity measure displayed in a chart's data point labels:



Note: You can usually format the data in the [metric set](#) or by using a [time dimension](#) rather than formatting measure and hierarchy keywords.

To format a keyword, add a colon and your format before the square closing bracket. For example, to format [Order Quantity] as numeric with no decimal places, change it to [Order Quantity:N0].



When you include the optional format, placeholder keywords support most [standard date and time format strings](#), [custom date and time format strings](#), and [standard numeric format strings](#). (Custom numeric format strings are not supported in visualization keywords.) To customize a hierarchy's formatting, its data type needs to be numeric, date/time, or a time dimension hierarchy, and it should not be placed on Slicers in the Data Analysis Panel.

Examples of numeric formats you can use:

Format Type	Format
Numeric with 2 decimal places	N2
Currency with no decimal places	C0
Percent (multiplied by 100) with 1 decimal place	P1
Whole numbers, 3 digits	D3

Standard date and time formats are specified by a single character and place their date and time components automatically according to the culture of each logged-in user:

Standard Format	Specifier	Sample Text
Short date	d	6/12/2015
Long date	D	Friday, June 5, 2015
Short time	t	8:31 AM
Long time	T	8:31:05AM
Long date, short time	f	Friday, June 5, 2015 8:31 AM
Long date, long time	F	Friday, June 5, 2015 8:31:05 AM
Month/day	M	June 05
Month/year	y	2015 June

Custom formats allow you to place each date and time component yourself. An example of a complete custom format is *yyyy-MM-dd*.

Custom Format	Specifier	Sample Text
Year	y, yy, yyyy	5, 05, 2005
Month	M, MM, MMM, MMMM	4, 04, Apr, April
Day	d, dd, ddd, dddd	9, 09, Wed, Wednesday
12-hour clock	h, hh	9, 09
24-hour clock	H, HH	9, 09 & 21, 21
Minutes	m, mm	6, 06

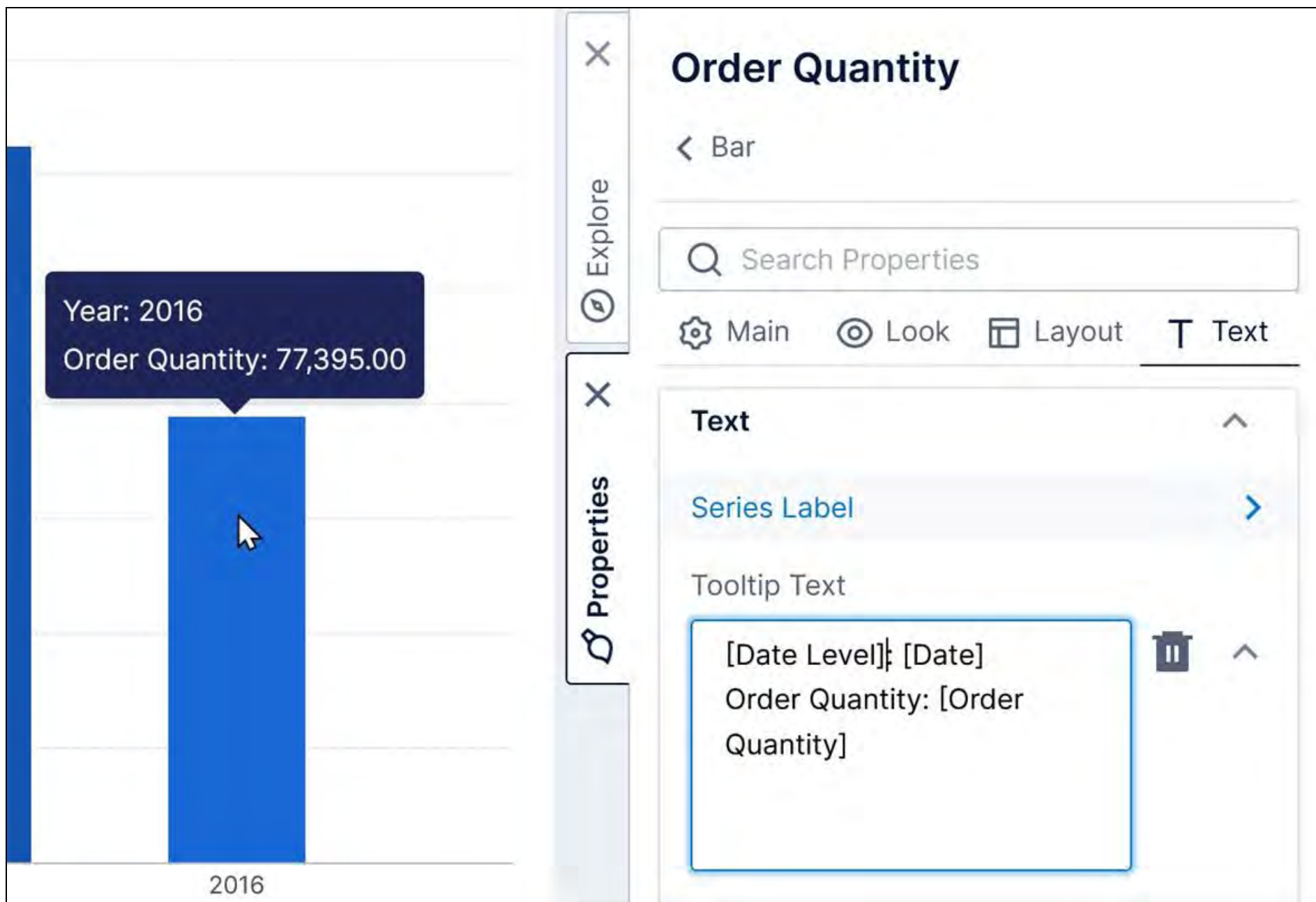


Custom Format	Specifier	Sample Text
Seconds	s, ss	8, 08
AM/PM	t, tt	a, AM
Fractional seconds	f, ff, fff	6, 61, 611
Literal text	'My Text'	My Text

Filter and Level Keywords

By default, visualization tooltips identify the data by its hierarchy caption, for example the caption Date regardless of whether the hierarchy level *Year* or *Month* is displayed from a time dimension.

If you want to display the caption of the hierarchy level, use the keyword [*<Hierarchy> Level*], replacing *<Hierarchy>* with your hierarchy name. For example: [*Date Level*].





You can display filter values for data placed on [slicers](#) as a [regular keyword](#).

When using a [slicer comparison](#) measure, you can also display the filter values determining the 'sliced' measure values using the keyword [*<Sliced Measure> <Slicer Hierarchy> Filter*]. For a slicer comparison measure with the caption *Sliced Sales Amount* filtered by the slicer Country, for example, you can display the particular country used to filter the sales amounts using [*Sliced Sales Amount Country Filter*].

The following keywords can be used in text properties when using a [Dynamic Element filter](#), replacing *<Dynamic Measure>* or *<Dynamic Hierarchy>* with the name of your particular dynamic measure or dynamic hierarchy:

Keyword	Description
[<Dynamic Measure> Caption]	The caption of the actual measure currently displayed by the dynamic measure.
[<Dynamic Measure>]	Represents the numeric values of the current dynamic measure.
[<Dynamic Hierarchy> Level]	The caption of the actual hierarchy or hierarchy level displayed by the dynamic hierarchy.
[<Dynamic Hierarchy>]	Represents the captions/values of the current dynamic hierarchy.

Visualization Keywords

In addition to measure and hierarchy keywords, the following keywords may be available depending on the visualization, and will be suggested in the [popup](#) when you edit a text property:

Keyword	Visualization	Description
[Total]	Stacked and 100% stacked charts, pie charts	Displays the stacked total up to and including that data point from zero or a pie's start angle.
[Percent]	100% stacked charts, pie charts	Displays the percent of the data point's value out of its stack total.
[Measure Name], [Measure Value]	Charts or tables displaying a transposed metric set	The measure name or value displayed by each data point or in each row.
[Member]	Pie chart, waterfall chart. charts with Hide Total Values unchecked, tables (custom total text), treemaps, tree diagrams, relationship/Sankey/chord diagrams (nodes)	The hierarchy member caption. In a table's custom total/subtotal text, displays the hierarchy's <i>All</i> caption, which can be set in a data cube's Process Result .
[Level]	Treemaps, tree diagrams, relationship/Sankey diagrams (nodes)	The hierarchy level caption.
[Size]	Treemaps	The value of the measure used to determine the node size.
[SizeMeasure]	Treemaps	The name of the measure used to determine the node size.
[Flow]	Sankey diagrams (nodes)	The total quantity flowing in or out of the node.



Keyword	Visualization	Description
[Source]	Sankey diagrams (links), chord diagrams (links)	The source hierarchy member caption.
[SourceValue]	Chord diagrams (links)	The quantity of the flow from the source to the target.
[Target]	Sankey diagrams (links), chord diagrams (links)	The target hierarchy member caption.
[TargetValue]	Chord diagrams (links)	The quantity of the flow from the target to the source (if applicable).
[Value]	Chord diagrams (nodes)	The total quantity flowing out of the node.

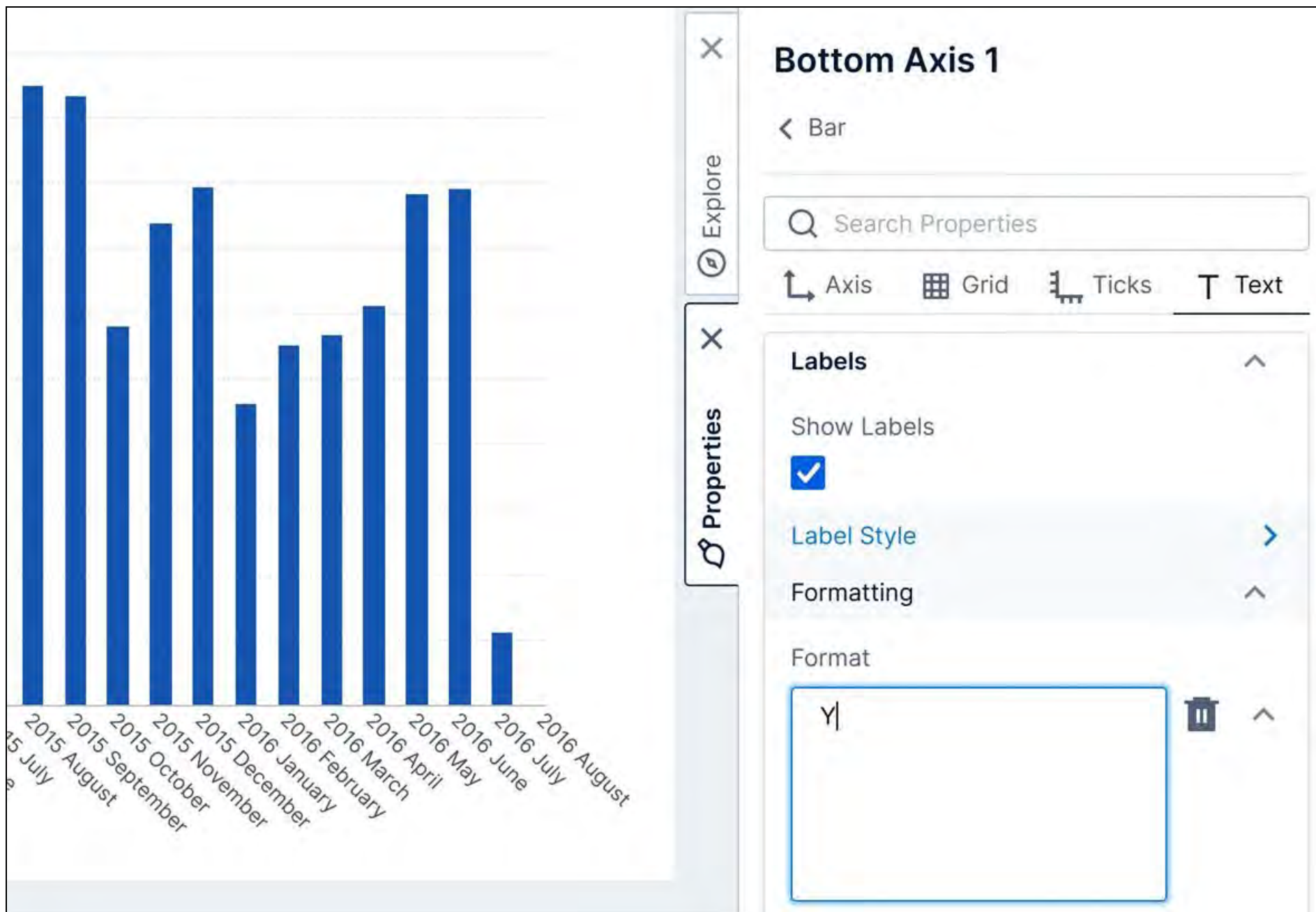
Size and Color Rule Keywords

The following keywords are supported for the legend text property of size and color rules (i.e., Range Size Rule, Auto Size Rule, Range Color Rule, and Auto Color Rule) in any visualization.

Keyword	Description
[From]	The starting value in the range.
[To]	The ending value in the range.

Format Properties

Dedicated format properties in data visualizations such as on chart axes and gauge scales support the same formats as [visualization keywords](#) (and have the same restrictions).





- Archive of documentation for Logi Symphony v24

Axes and scales have a **Symbol** property below the format similar to the settings on measures. When applying currency formatting, it is best to set this to the currency symbol for your data as shown above for measures so that it is displayed consistently for users logged in with different [language or culture settings](#).

For more information, see:

- Video: [Using The Data Analysis Pan](#)
- [Create a Time Dimension](#)
- [Symphony Metric Sets](#)



Group Members During Analysis

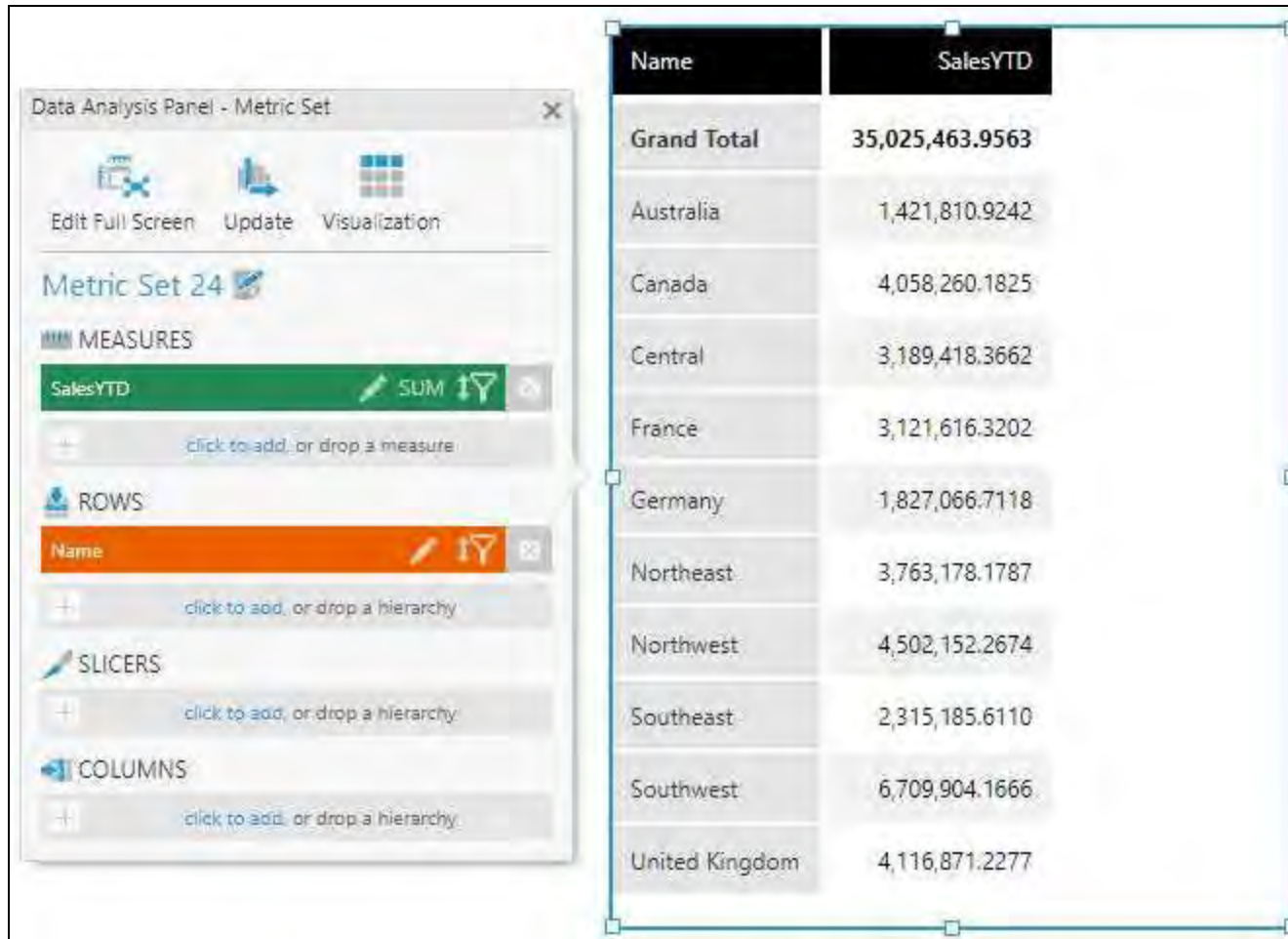
This applies to: *Managed Dashboards, Managed Reports*

You can group multiple values into one when working with a [metric set](#). Do as part of analysis, and undo if needed. Use help an analysis when data is not grouped the way it's needed.

This option is available when working with a metric set that is checked out to you.

Ad Hoc Member Grouping

While working with a checked-out metric set in the full-screen editor, or in a dashboard or another view, select data so that at least one element is placed under Rows or Columns in the Data Analysis Panel.



In our example above, the territories representing the United States are separated into members *Central*, *Northeast*, *Northwest*, *Southeast*, and *Southwest*.

Right-click one of the members and select **Group Members**. (If you are editing a dashboard or another view, click **View** in the toolbar first before interacting with a visualization's data.)

Name	SalesYTD
Grand Total	35,025,463.9563
Australia	1,421,810.9242
Canada	4,058,260.1825
Central	3,189,418.3662
France	3,121,616.3202
Germany	1,827,066.7118
Northeast	3,763,178.1787
Northwest	
Southeast	
Southwest	
United Kingdom	

Data Analysis Panel

Re-Visualize

Data Tools

Display As Flat Table

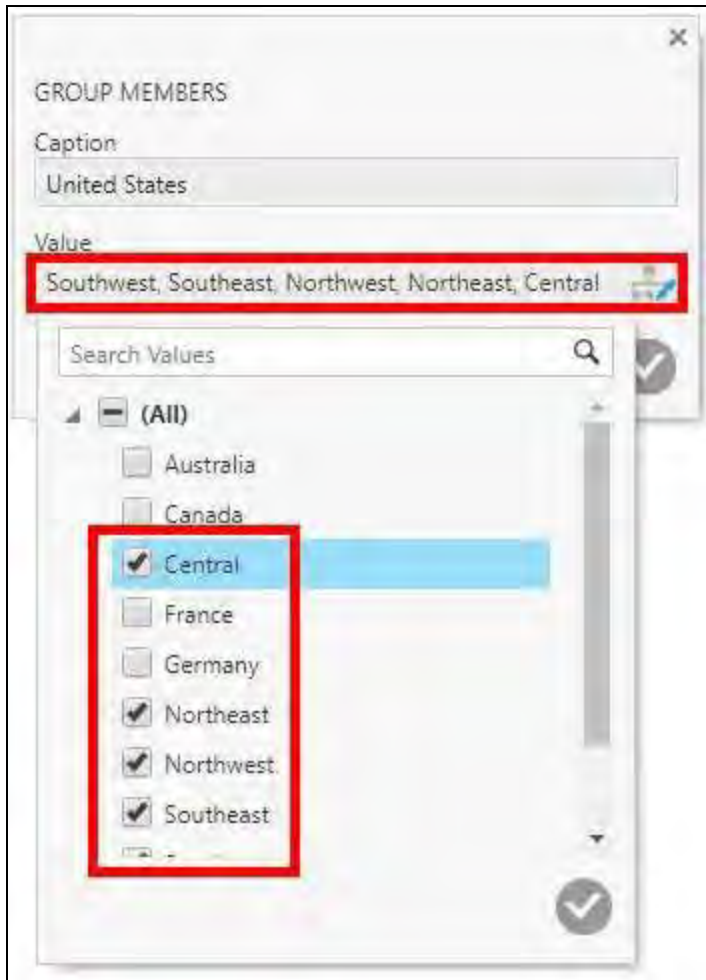
Sort

Filter

Collapse All

Group Members

In the *Group Members* popup, provide a **Caption** for the group and use the **Value** drop-down to select the members that should be grouped together if you didn't select them earlier.



Note: In tables or charts, you can select multiple [rows](#), [columns](#), or [data points](#) ahead of time and right-click them to group them together instead of choosing them in this popup.

Submit the drop-down and the *Group Members* popup.

Your group will now be displayed as a single member with measure values aggregated together according to the measure's aggregator (e.g., sum).

Name	SalesYTD
Grand Total	35,025,463.9563
Australia	1,421,810.9242
Canada	4,058,260.1825
France	3,121,616.3202
Germany	1,827,066.7118
United Kingdom	4,116,871.2277
United States	20,479,838.5899

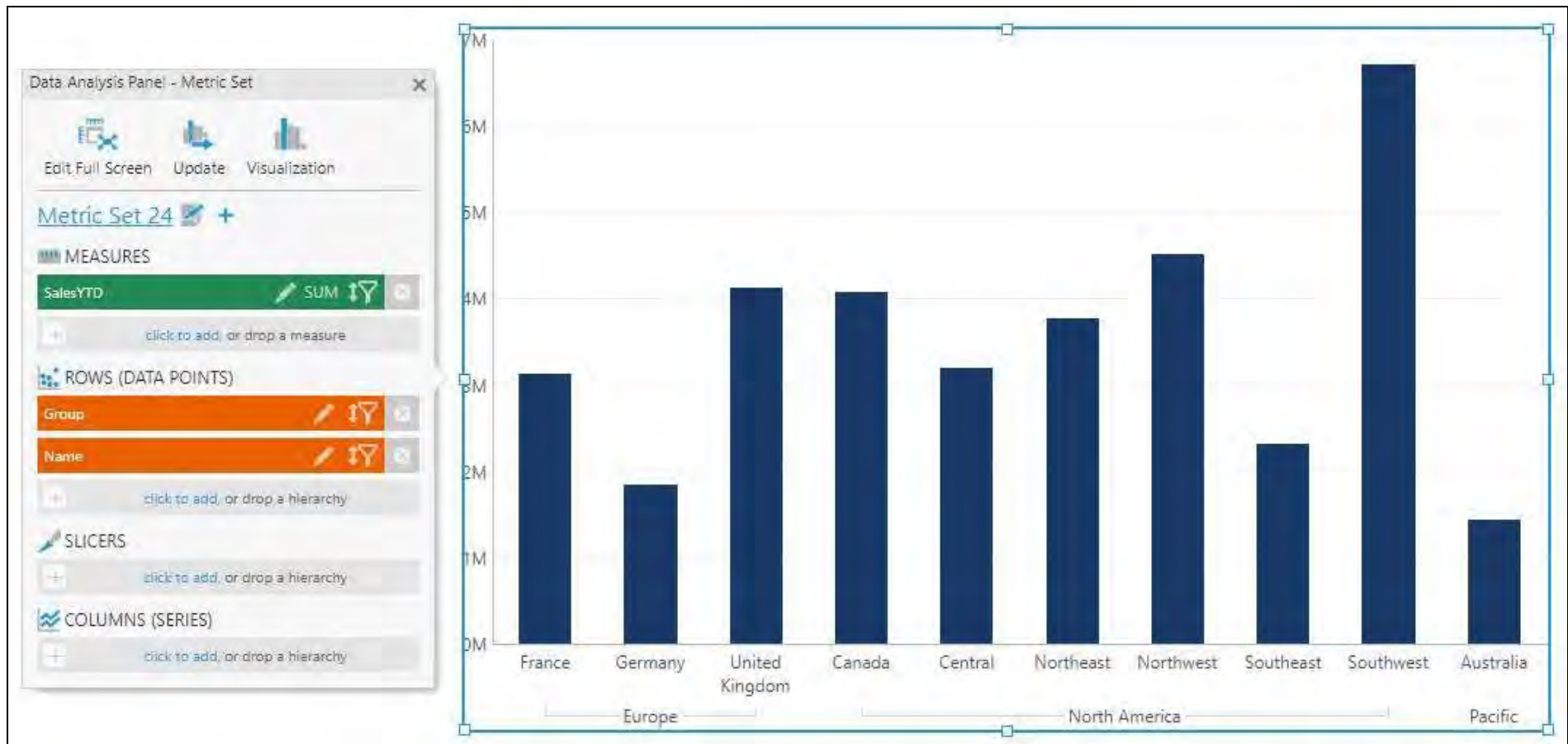
To undo this grouping, right-click the group and select the **Ungroup** option to ungroup it.

A group can't be changed after it's created, but you can ungroup the members and then group them again.

Working With Multiple Hierarchies

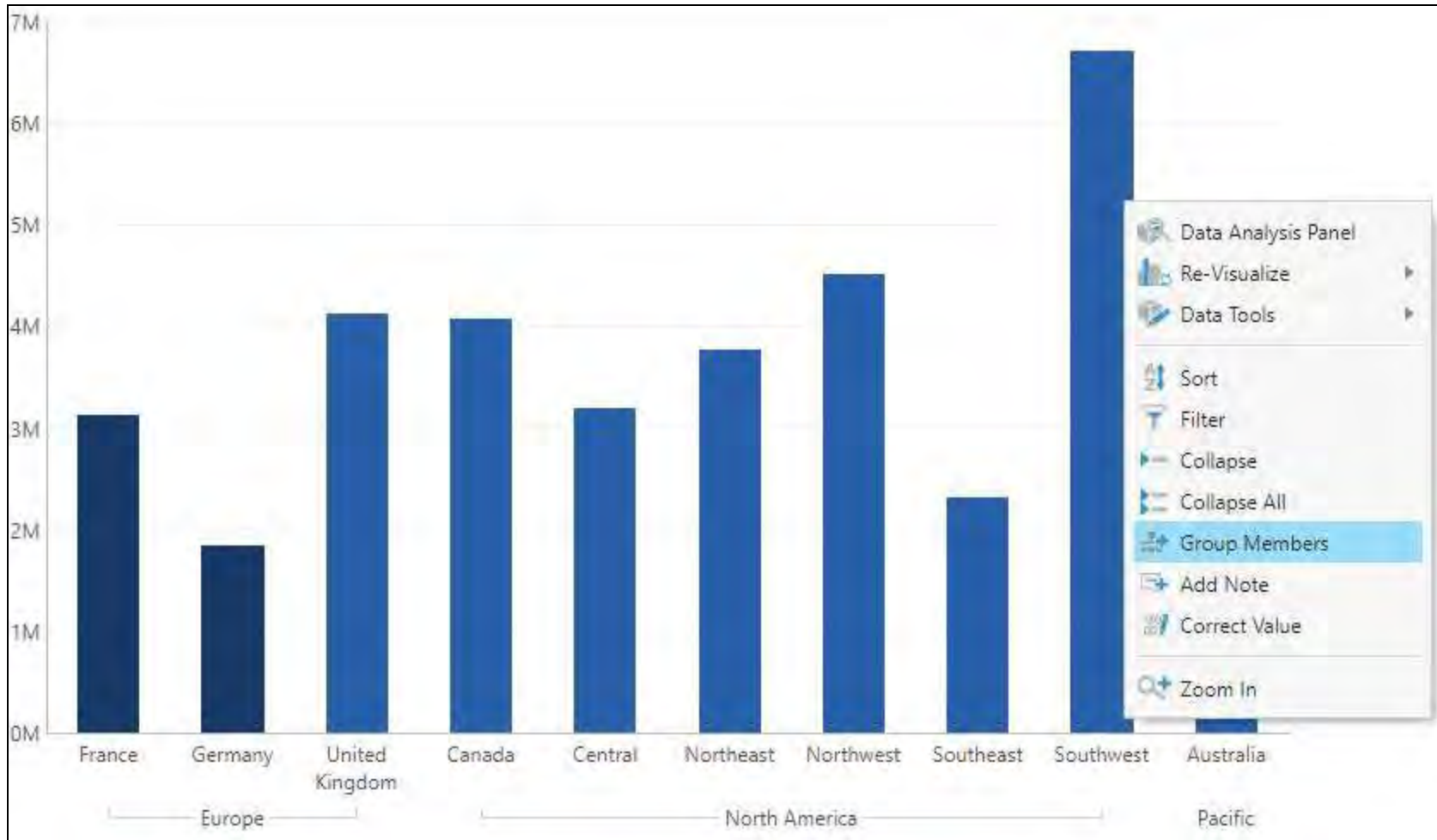
When your data is grouped by multiple hierarchies, a group you created might be split up by the members of another hierarchy.

In our example from above, we have added the *Group* hierarchy to the metric set and placed it first under **Rows**, then we re-visualized into a bar chart.

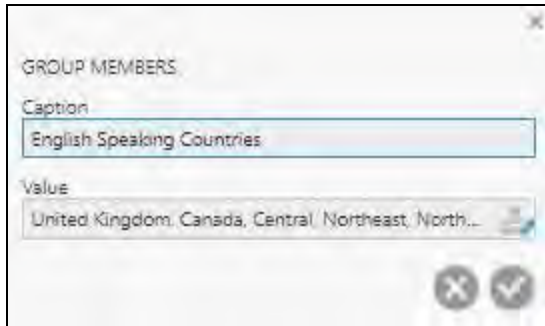


Right-click on one or more data points and choose **Group Members** to create a group. (Because there are multiple hierarchies, you may be asked which one you want to group values for.)

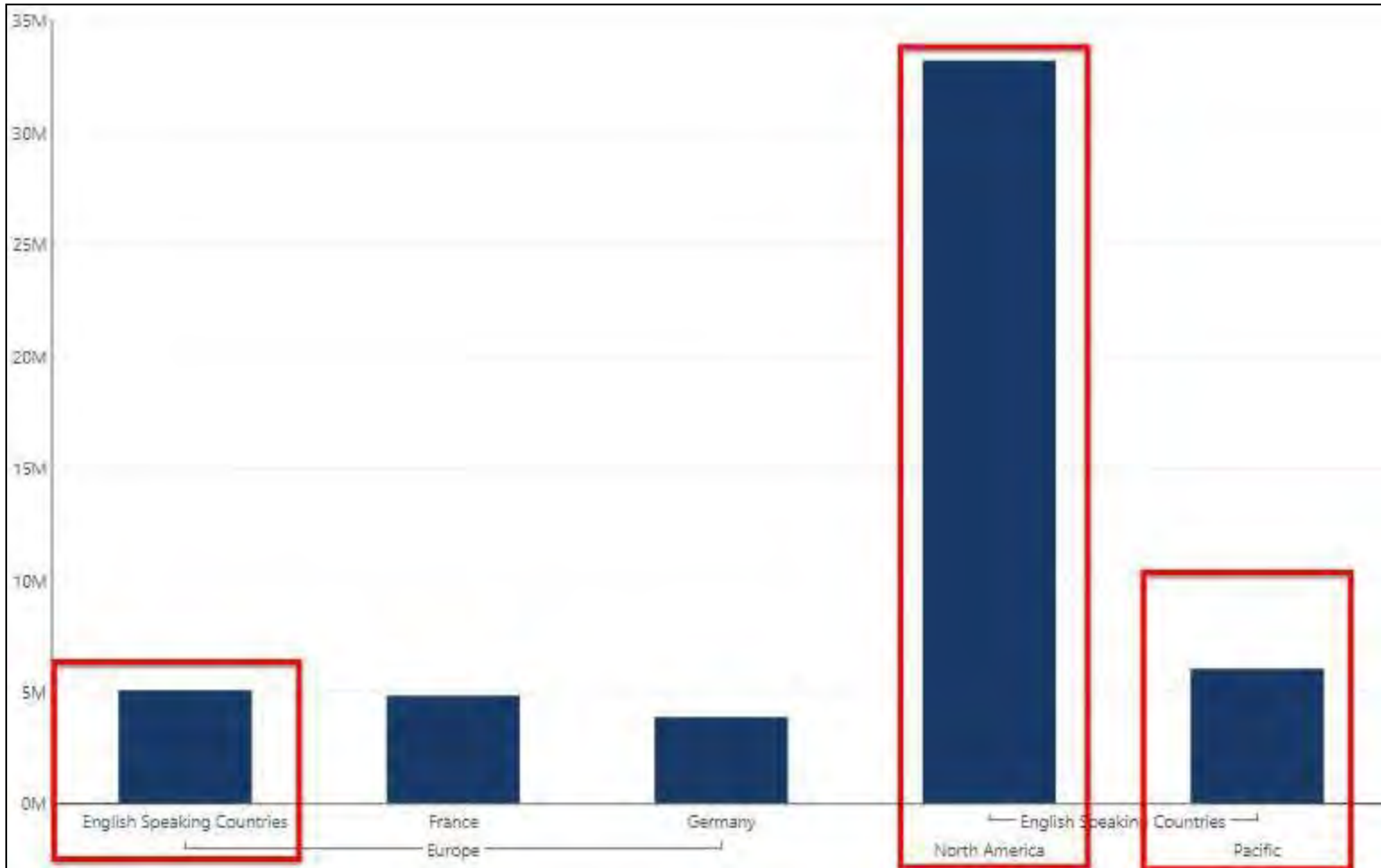
In our example, we will select the English-speaking data points in the chart by clicking and dragging across it to select points, and hold the *Ctrl* key on the keyboard while selecting additional points.



In the *Group Members* popup, provide a **Caption** for the new group, confirm the **Value** members and submit.



The English Speaking Countries group is now displayed in each of the *Group* hierarchy elements where it has members, and aggregated accordingly.



Using Custom Member Grouping

- Custom member grouping is not available for the following types of hierarchies:
 - Dynamic
 - Time-based
 - OLAP



- Archive of documentation for Logi Symphony v24

- When working with user-defined hierarchies, grouping can be done only within the same hierarchy level and for members with the same parent.

For more information, see:

- [Displaying Top/Bottom Records](#)
- [Symphony Metric Sets](#)



Handling Missing Data

This applies to: *Managed Dashboards, Managed Reports*

This article shows you how you can handle null values and missing data in your metric set. If rows are missing from your data or have measure values that are null or missing, you can set up a missing data rule to display them and optionally fill in their measure values.

By default, measure values that are null or missing are not displayed by the metric set or its visualizations.

Example Data

Missing Measure Values

This first example uses the **Sales.SalesPerson** table from the [AdventureWorks sample database](#), which contains rows where the **SalesQuota** column is null (missing).

First, drop the **BusinessEntityID** column onto a metric set or dashboard:

BusinessEntityID
274
275
276
277
278
279
280
281
282
283

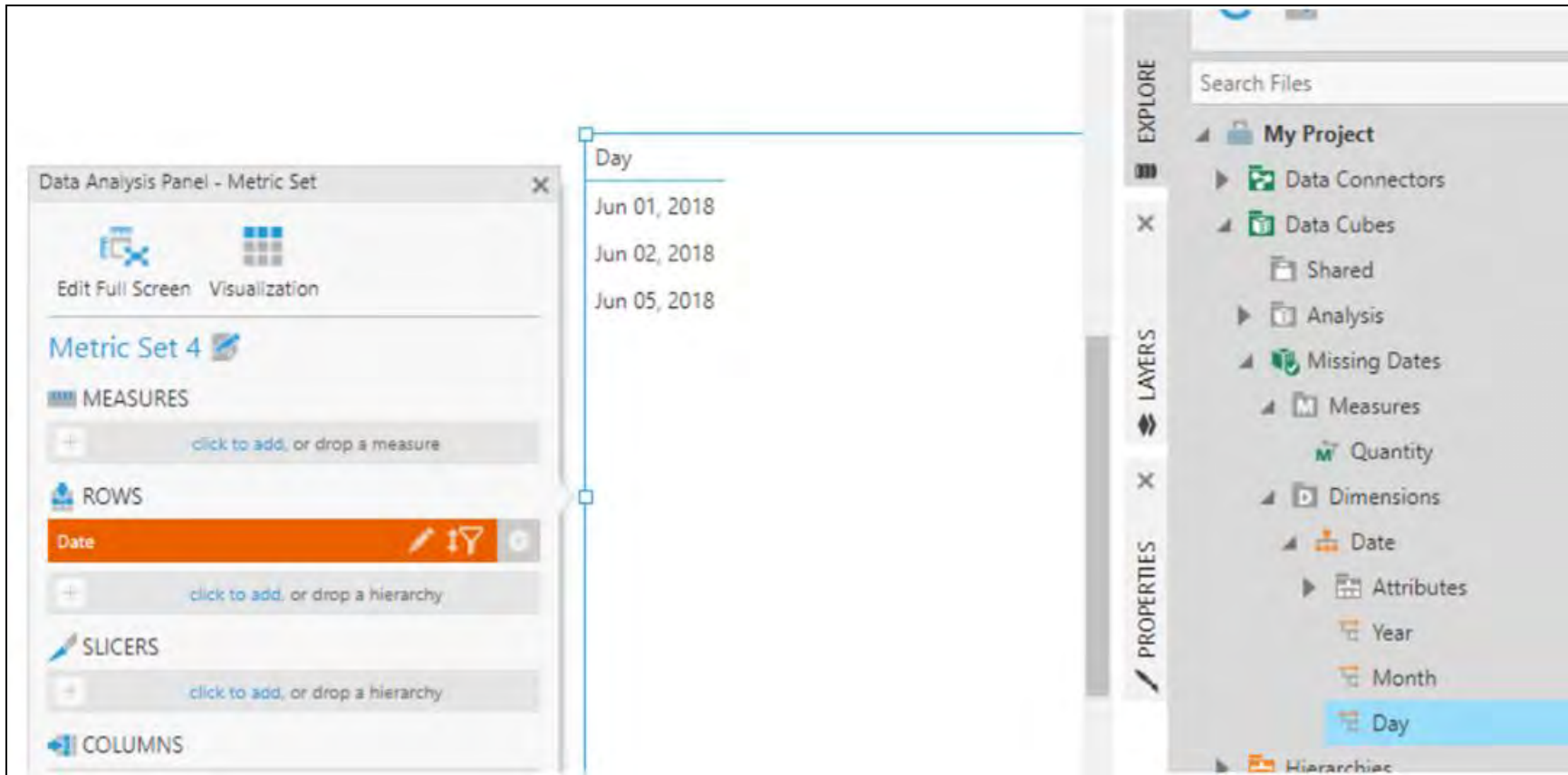
Then, drop **SalesQuota** onto the area over the table visualization labeled **Column**:

BusinessEntityID	SalesQuota
Grand Total	3,650,000.00
275	300,000.00
276	250,000.00
277	250,000.00
278	250,000.00
279	300,000.00
280	250,000.00
281	250,000.00
282	250,000.00
283	250,000.00

Notice that **BusinessEntityID 274** is now missing in the list after the **SalesQuota** was added. This is because there is no **SalesQuota** value in that row, but you can display it using a missing data rule.

Missing Dates or Hierarchy Values

As another example, the following data source is missing entire rows corresponding with June 3rd & 4th, and contains rows only for the 1st, 2nd, and 5th:



This column of dates was promoted to a time dimension hierarchy in a data cube ahead of time, and its **Day** level was dragged onto the canvas to create a metric set. When working with a column of dates directly from a data connector, you can [expand it to drag a time dimension hierarchy level](#) without leaving your metric set or dashboard.

Use a [time dimension](#) to add in entire rows of data for dates that are missing from your data source according to the current level of the hierarchy, such as Day, Month, Hour, etc. Similarly, a predefined [hierarchy](#) can fill in entire rows missing for other types of values.



Note: If you do not use a time dimension or predefined hierarchy, missing data rules can only fill in measure values that are missing in existing rows where other values are already present in the data source, like for the **BusinessEntityID** example earlier.

After adding a measure to the Data Analysis Panel, the data is automatically visualized as a line chart showing gaps for the missing dates. In this example, [markers](#) were also enabled to clearly show each data point along the line:



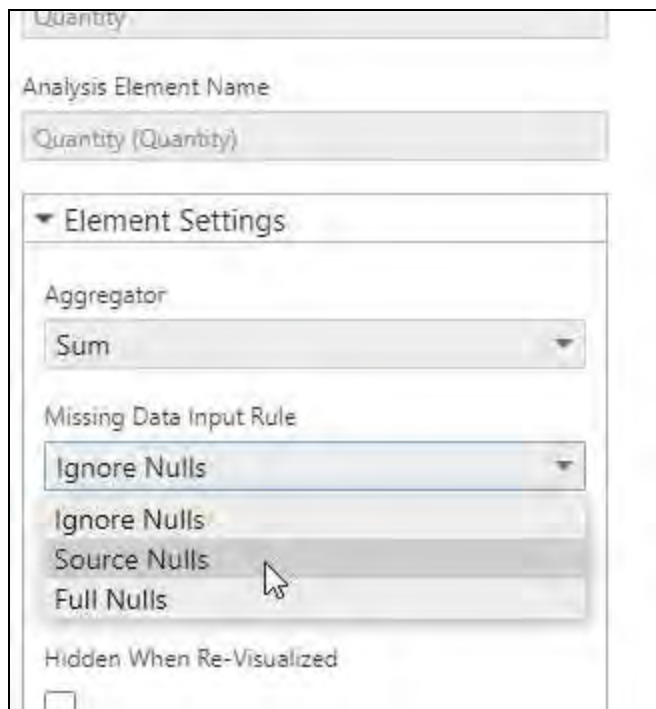
Note: If you have a chart like the above and only want to fill in the missing axis labels, you can also remove the time dimension from under **Bottom Axis Label** in the Data Analysis Panel's [Visualization tab](#) and allow the chart to label the axis on its own.

Using a Missing Data Rule

To include the missing rows or measure values, select to edit the measure in the Data Analysis Panel.



In the dialog under **Element Settings**, set the **Missing Data Input Rule** to determine which missing data to include:



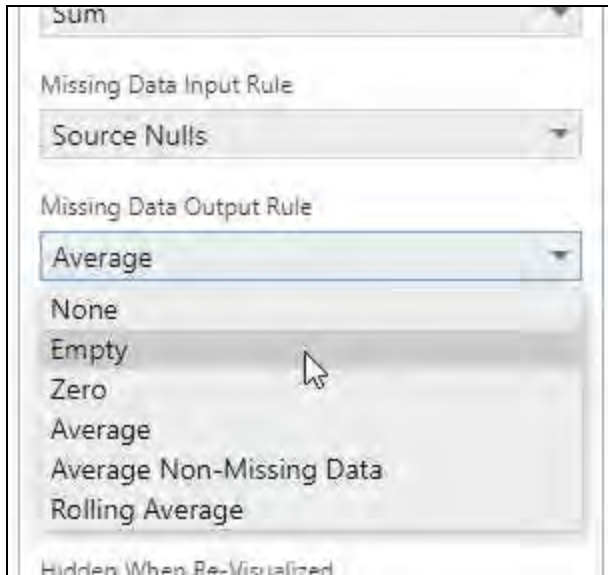
- **Ignore Nulls** (default) - does not include any measure values that were null or missing.
- **Source Nulls** - includes only the null measure values present in the records in the data source.
- **Full Nulls** - includes all possible null values even if records did not exist for them in the data source.



Note: If you have multiple hierarchies, the **Full Nulls** option includes all possible combinations of values and may consume a lot of memory for large data sources.

If we choose **Source Nulls** for our first example, **BusinessEntityID 274** can be displayed. When using a time dimension or predefined hierarchy like in the second example, the **Full Nulls** option can fill in missing hierarchy values.

Next, set the **Missing Data Output Rule** to determine how missing measure values should be displayed:



- **None** (default) - does not display data where the measure value was null or missing, and excludes those rows or columns.
- **Empty** - a row, column, or data point is included but the measure value is left null/blank.
- **Zero** - inserts zeros where the measure value was missing.
- **Average** - displays the average between the previous and next data points.



- **Average Non-Missing Data** - displays the average between the previous and next data points that were not missing measure values.



- **Rolling Average** - displays the average of a certain number of previous data points determined by the **Rolling Average Period** setting.



For the options that calculate an average, also set **Axis The Average Will Affect**. This is typically **Rows** to use next and previous rows in a table or data points in a series, but can be **Columns**.

If you choose **Empty**, the **Empty Value Text** field will allow you to optionally insert text to be displayed in place of the measure value (e.g., **N/A**).

Missing Data Input Rule
Source Nulls

Missing Data Output Rule
Empty

Empty Value Text
N/A

In our first example, **BusinessEntityID 274** can now show **N/A** for **SalesQuota**.

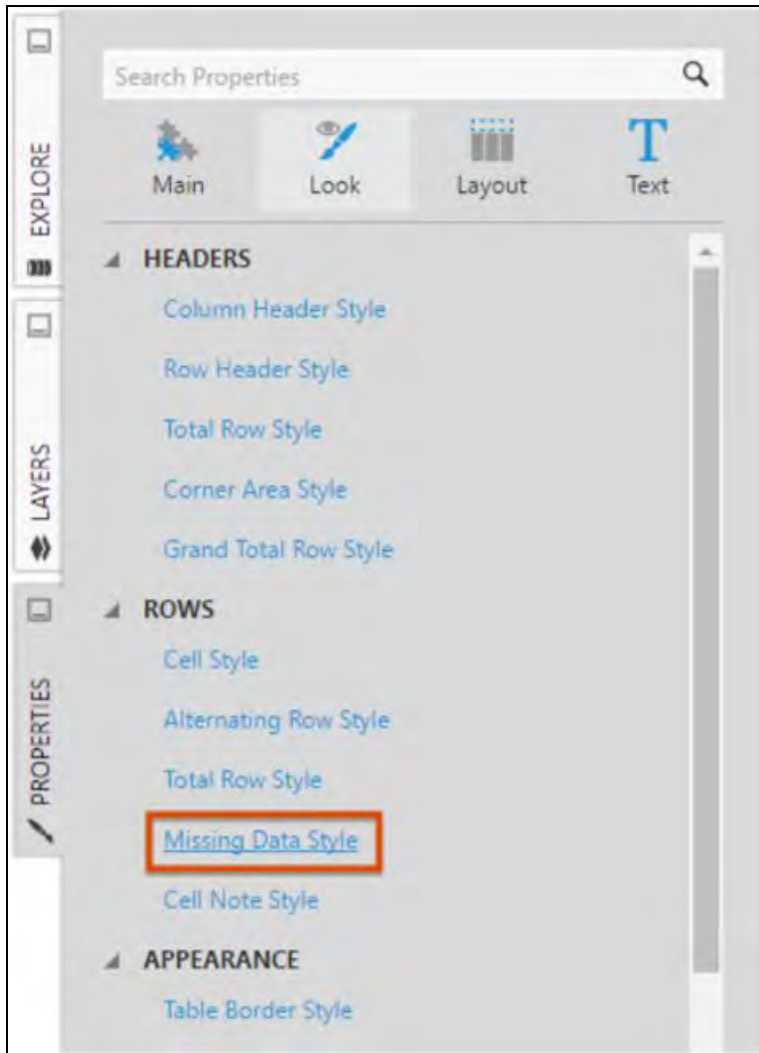
BusinessEntityID	SalesQuota
Grand Total	3,650,000.0000
274	N/A
275	300,000.0000
276	250,000.0000
277	250,000.0000
278	250,000.0000
279	300,000.0000
280	250,000.0000
281	250,000.0000
282	250,000.0000

Styling Missing Data

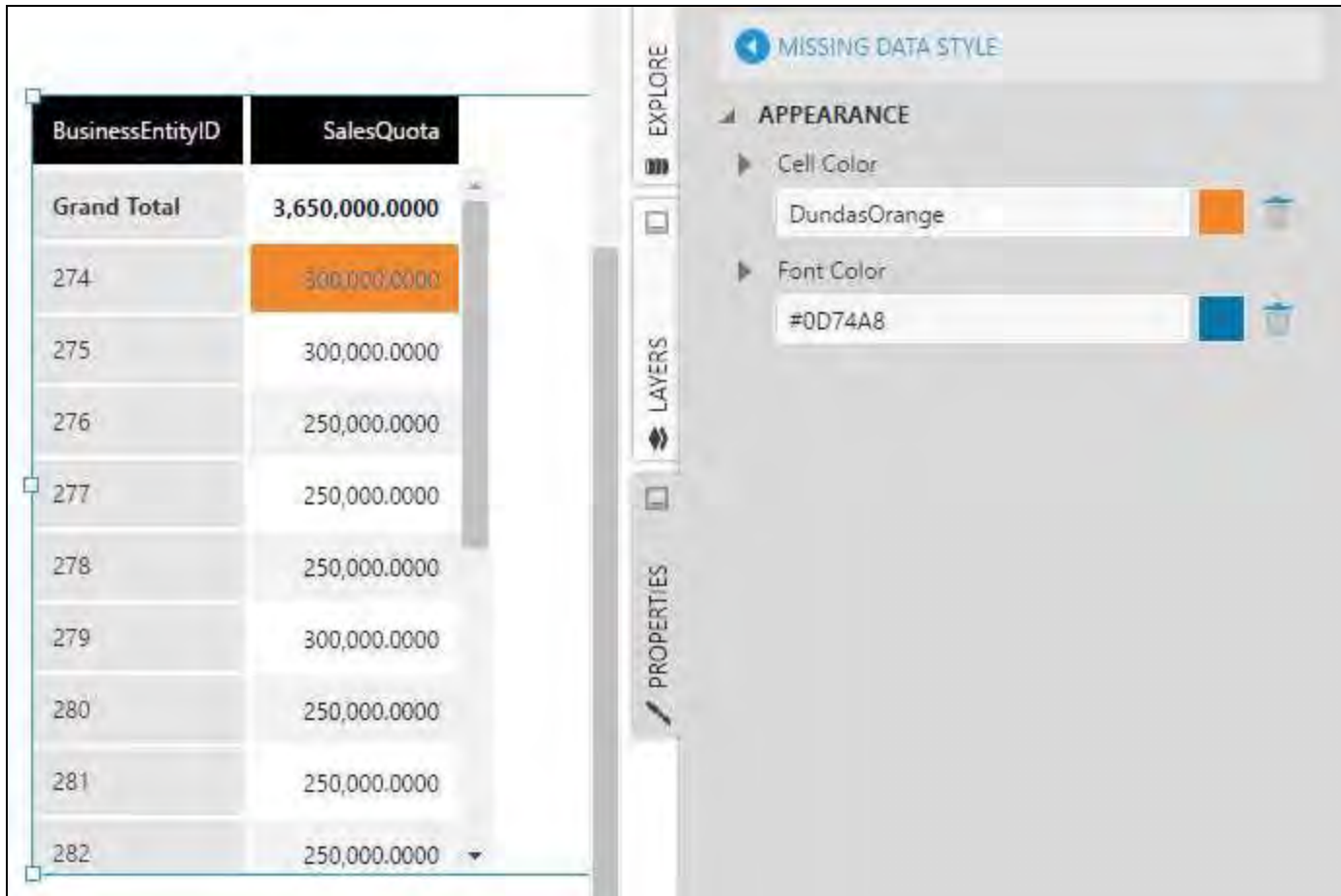
You can style cells or data points differently for missing data using the **Properties** window.

Table

In the **Look** tab of the [properties for a table](#) select **Missing Data Style**.



The **Cell Color** and **Font Color** properties can be set to distinguish cells showing data calculated by the missing data rule.



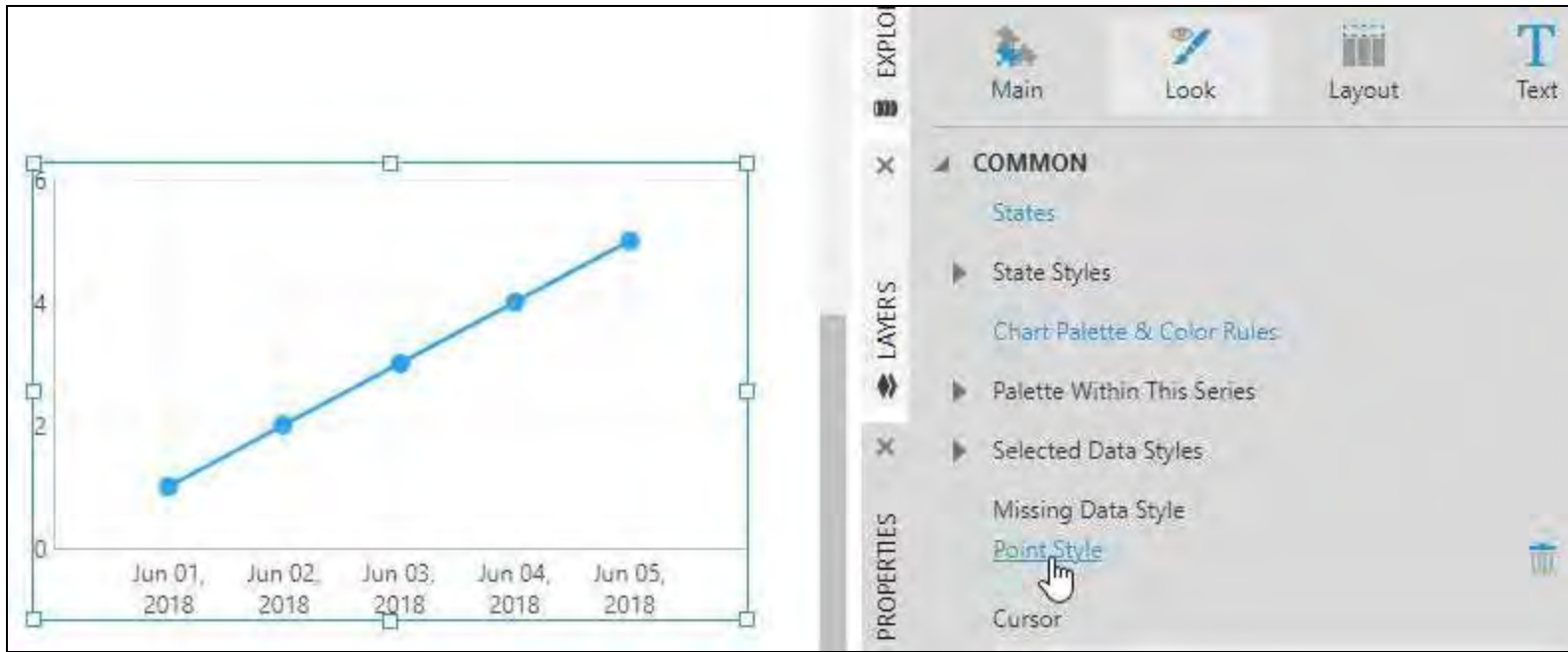
The screenshot displays the Logi Symphony interface. On the left, a table is shown with two columns: 'BusinessEntityID' and 'SalesQuota'. The table includes a 'Grand Total' row and several data rows (274-282). The 'SalesQuota' values range from 250,000.0000 to 3,650,000.0000. The row for '274' is highlighted in orange. On the right, the 'MISSING DATA STYLE' panel is open, showing the 'APPEARANCE' section. Under 'Cell Color', the value 'DundasOrange' is selected with an orange color swatch. Under 'Font Color', the value '#0D74A8' is selected with a blue color swatch. The interface also features a vertical sidebar with 'EXPLORE', 'LAYERS', and 'PROPERTIES' tabs.

BusinessEntityID	SalesQuota
Grand Total	3,650,000.0000
274	3,000,000.0000
275	300,000.0000
276	250,000.0000
277	250,000.0000
278	250,000.0000
279	300,000.0000
280	250,000.0000
281	250,000.0000
282	250,000.0000

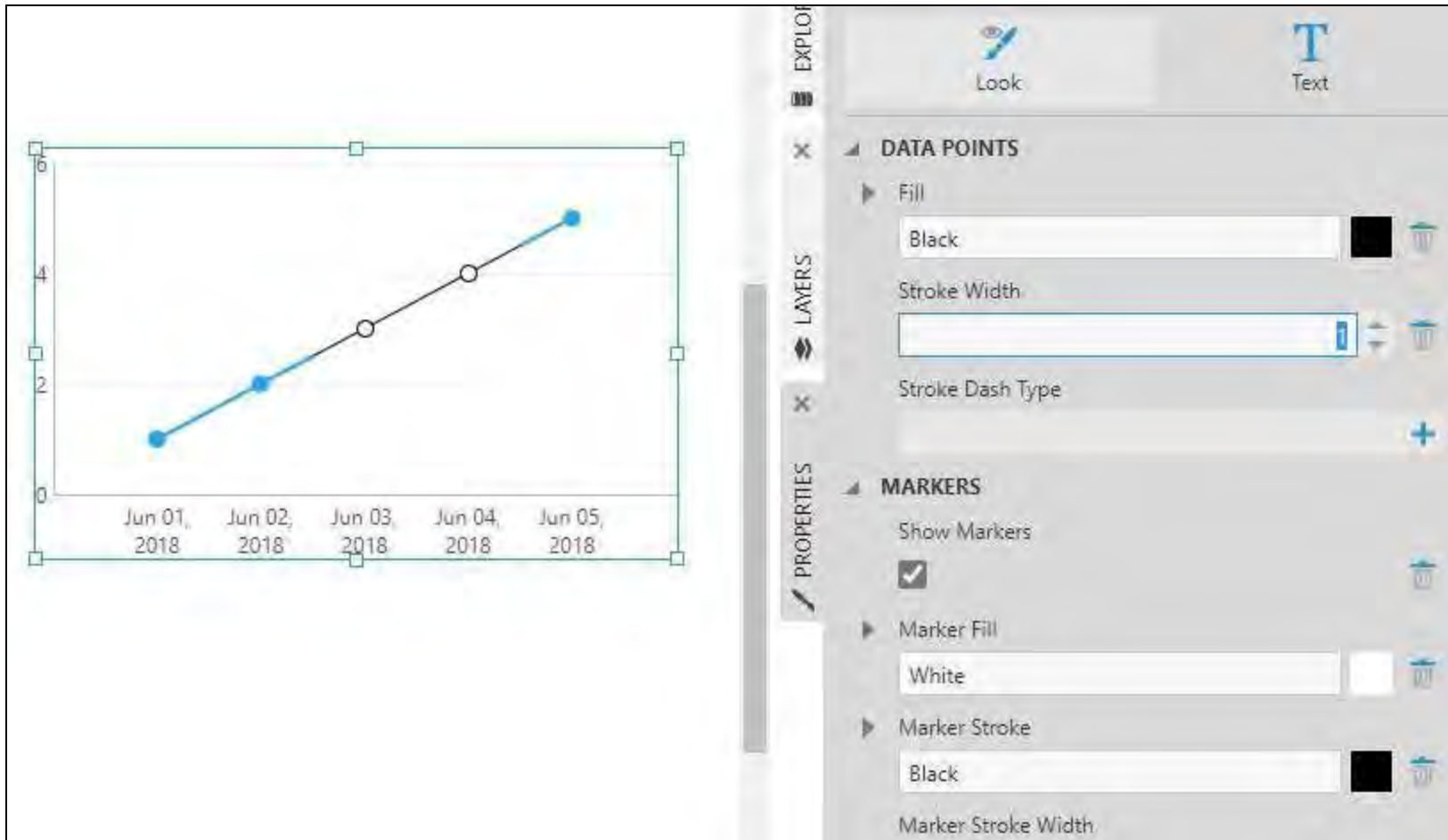
Chart

In a chart, go to the [properties](#) for its data point series. You can right-click a chart's data points when editing and choose **Series Properties**.

Go to the **Look** tab to find `Missing Data Style`, select **+** to add one if there is none, then select to navigate to the style settings.



In this example, data points that were filled in using a missing data rule are distinguished using a thin black line stroke and hollow markers:



For more information, see:

- [Symphony Metric Sets](#)
- [Other Metric Set Tools](#)
- [Create a Time Dimension](#)
- [Using a Table Visualization](#)



- Archive of documentation for Logi Symphony v24

- [Using Chart Properties](#)
- [Handling Null Value in a State Indicator Column](#)

How To Create a Custom User Hierarchy

This applies to: *Managed Dashboards, Managed Reports*

As shown in [Automatic Joins and Hierarchies](#), you can create a hierarchy in Symphony from the main menu (sometimes called a 'user hierarchy' as it is user-defined) with settings such as multiple levels, attributes, and sorting. In many cases, such as when the data source contains multiple columns or there are [relationships](#) defined, the levels are automatically detected and added.

This article is a continuation that shows in more detail how to customize the hierarchy's levels in case they are not automatically detected as needed or you want to change them.

Example Data

In the example depicted below, an Excel file with three different sheets is used as the data source:

- ProductCategory

	A	B
1	ProductCategoryID	Name
2	1	Bikes
3	2	Components
4	3	Clothing
5	4	Accessories
6		

- ProductSubcategory

	A	B	C
1	ProductSubcategoryID	ProductCategoryID	Name
2	1	1	Mountain Bikes
3	2	1	Road Bikes
4	3	1	Touring Bikes
5	4	2	Handlebars
6	5	2	Bottom Brackets
7	6	2	Brakes
8	7	2	Chains
9	8	2	Cranksets
10	9	2	Derailleurs
11	10	2	Forks
12	11	2	Headsets
13	12	2	Mountain Frames
14	13	2	Pedals
15	14	2	Road Frames
16	15	2	Saddles
17	16	2	Touring Frames
18	17	2	Wheels
19	18	3	Bib-Shorts
20	19	3	Caps
21	20	3	Gloves
22	21	3	Jerseys
23	22	3	Shorts
24	23	3	Socks
25	24	3	Tights
26	25	3	Vests
27	26	4	Bike Racks
28	27	4	Bike Stands
29	28	4	Bottles and Cages
30	29	4	Cleaners
31	30	4	Fenders
32	31	4	Helmets
33	32	4	Hydration Packs
34	33	4	Lights
35	34	4	Locks
36	35	4	Panniers
37	36	4	Pumps
38	37	4	Tires and Tubes
39			



- Archive of documentation for Logi Symphony v24

- Product

	A	B	C
1	ProductID	Name	ProductSubcategoryID
211	680	HL Road Frame - Black, 58	14
212	706	HL Road Frame - Red, 58	14
213	707	Sport-100 Helmet, Red	31
214	708	Sport-100 Helmet, Black	31
215	709	Mountain Bike Socks, M	23
216	710	Mountain Bike Socks, L	23
217	711	Sport-100 Helmet, Blue	31
218	712	AWC Logo Cap	19
219	713	Long-Sleeve Logo Jersey, S	21
220	714	Long-Sleeve Logo Jersey, M	21
221	715	Long-Sleeve Logo Jersey, L	21
222	716	Long-Sleeve Logo Jersey, XL	21
223	717	HL Road Frame - Red, 62	14
224	718	HL Road Frame - Red, 44	14
225	719	HL Road Frame - Red, 48	14
226	720	HL Road Frame - Red, 52	14
227	721	HL Road Frame - Red, 56	14
228	722	LL Road Frame - Black, 58	14
229	723	LL Road Frame - Black, 60	14
230	724	LL Road Frame - Black, 62	14
231	725	LL Road Frame - Red, 44	14
232	726	LL Road Frame - Red, 48	14
233	727	LL Road Frame - Red, 52	14
234	728	LL Road Frame - Red, 58	14
235	729	LL Road Frame - Red, 60	14
236	730	LL Road Frame - Red, 62	14
237	731	ML Road Frame - Red, 44	14
238	732	ML Road Frame - Red, 48	14
239	733	ML Road Frame - Red, 52	14
240	734	ML Road Frame - Red, 58	14
241	735	ML Road Frame - Red, 60	14
242	736	LL Road Frame - Black, 44	14
243	737	LL Road Frame - Black, 48	14
244	738	LL Road Frame - Black, 52	14
245	739	HL Mountain Frame - Silver	12
246	740	HL Mountain Frame - Silver	12
247	741	HL Mountain Frame - Silver	12
248	742	HL Mountain Frame - Silver	12
249	743	HL Mountain Frame - Black	12
250	744	HL Mountain Frame - Black	12
251	745	HL Mountain Frame - Black	12
252	746	HL Mountain Frame - Black	12
253	747	HL Mountain Frame - Black	12



In this example, each sheet contains an ID column with unique values, so that one of these ID values in a lower level or another data source will be enough to find the corresponding hierarchy member.

Steps

Set Up the First Level

If you are starting with a new hierarchy, create a new Hierarchy from the main menu as shown in [Automatic Joins and Hierarchies](#). Otherwise, your existing level can be edited as shown further down.

Expand your data connector in the **Explore** window and find the data with the first level's values (e.g., the Excel sheet named *ProductCategory\$*), then drag it to the canvas.

Your data source can be different for each level or the same for all levels, but either way each level below the first must include parent key values that match the keys of the level above.

If you want to ensure additional levels are not added automatically, press *Shift* on the keyboard while dropping. Otherwise, you can select unwanted levels to delete them from the toolbar.



Hierarchy

Drag and drop an existing Hierarchy to load it, or create a new one by dropping a table, cube, or any other native structure. If the SHIFT key is held while dragging a native structure, the levels will not be automatically discovered and only a single level will be created.

[ProductCategory\$]

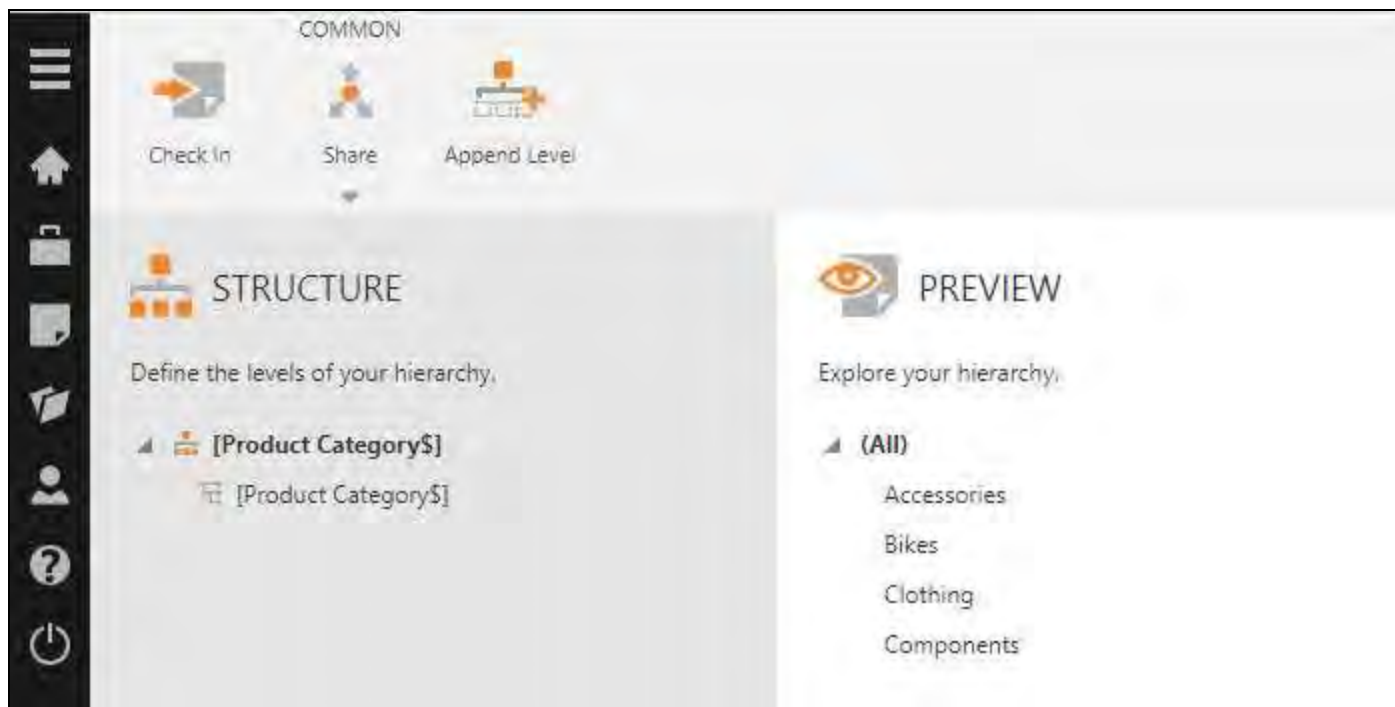
EXPLORE

Search Files

My Project

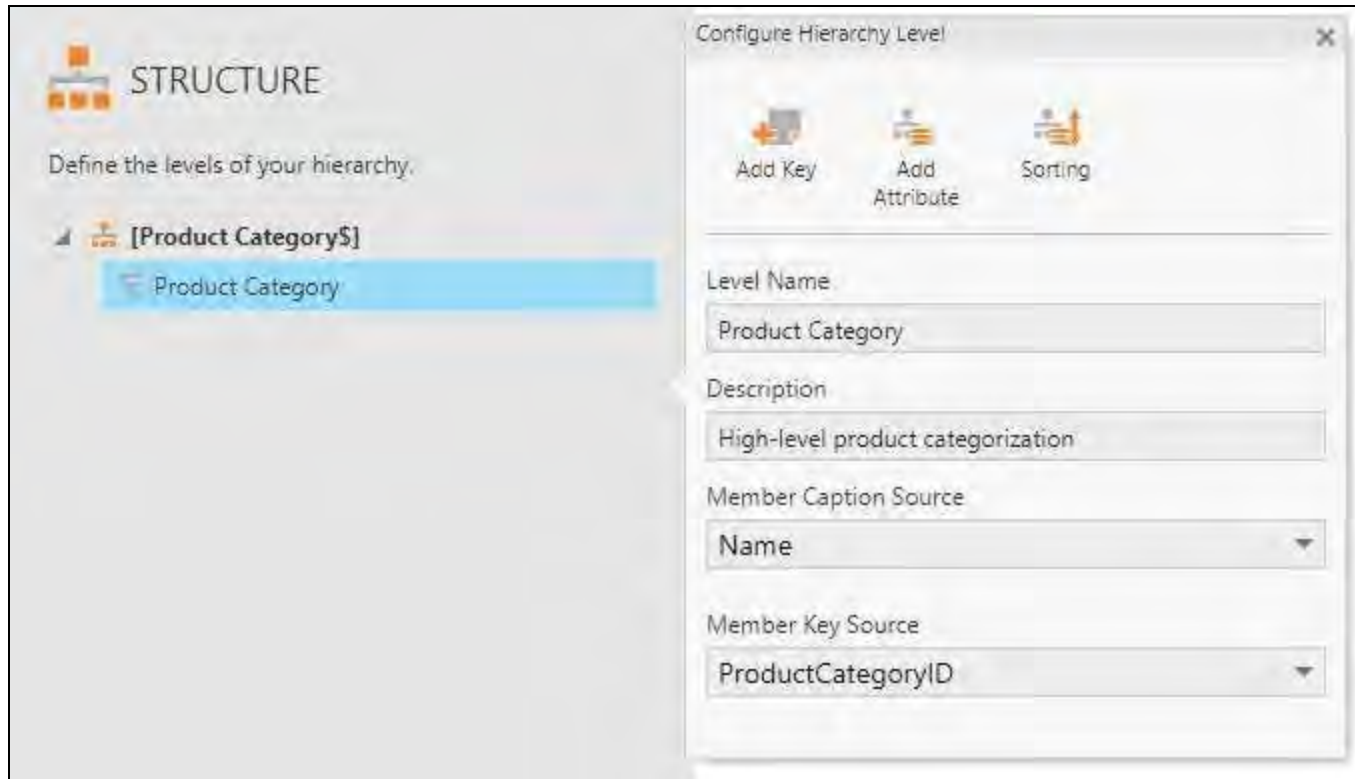
- Data Connectors
 - Shared
 - Products_Excel
 - [Product\$]
 - [ProductCategory\$]**
 - [ProductSubcategory\$]
- Data Cubes
- Hierarchies
- Time Dimensions
- Metric Sets
- Dashboards
- Reports
- Scorecards

The first level can now be previewed.



Click on the first level under *Structure* to open its settings, used to determine the values or 'members' of the hierarchy at this level. Confirm or change these details:

- **Level Name** - The name used to identify this level in the Explore window, metric sets, and filters.
- **Description** - (*Optional*) Displayed in a tooltip popup when hovering over this level in the Explore window.
- **Member Caption Source** - The values used to identify the hierarchy members in this level when displayed in visualizations and filters.
- **Member Key Source** - The values used to uniquely identify the members in this level and match them with other data sources. These may be different than the caption values if you have dedicated IDs, for example.

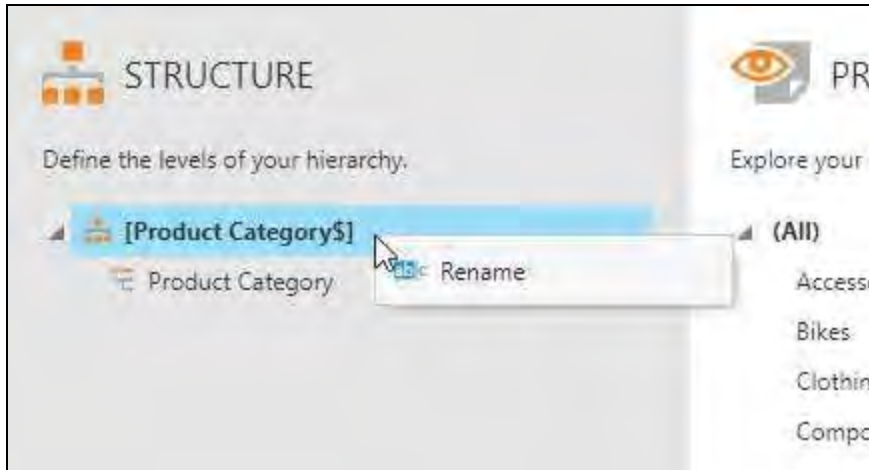


In our example, *Name* values such as *Accessories* will be displayed in filters and visualizations as captions, while *ProductCategoryID* values are meant to uniquely identify each category and will be used to match with product category IDs in the lower level.



Important: The **Member Key Source** values must be unique within each level in order to know which member corresponds with a matching key value. You can use two or more values to uniquely identify each member (composite key) if needed by clicking **Add Key** and using the additional drop-down(s) to select the other key values.

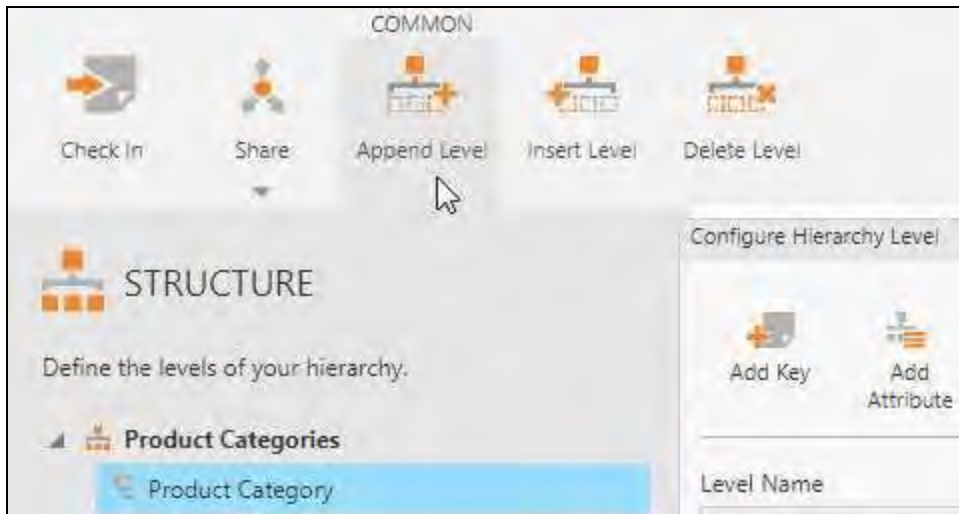
You can right-click the hierarchy and choose **Rename** to give the hierarchy a different name within this editor.



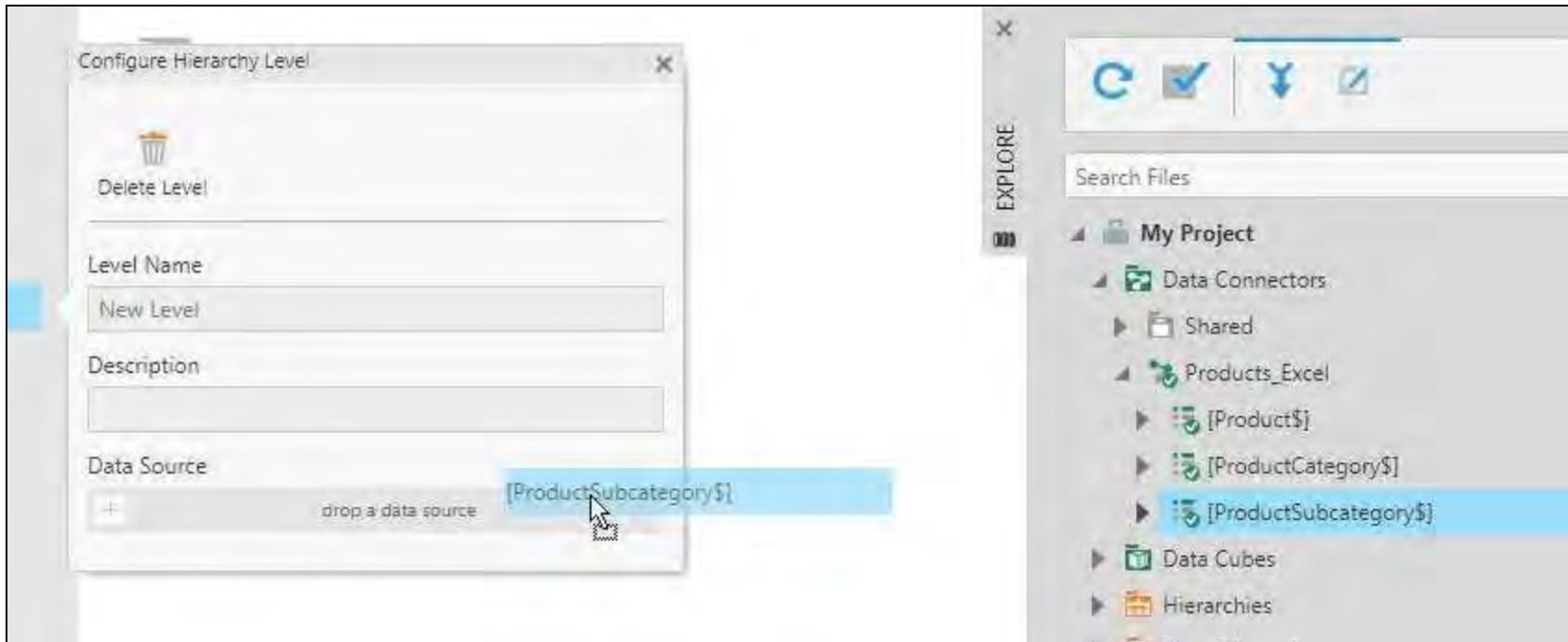
To rename the hierarchy file as shown in dialogs and the Explore window, [double-click the name](#) in the status bar at the bottom of the screen.

Add the Second Level

To add another level, select the first level and click **Append Level** in the toolbar.

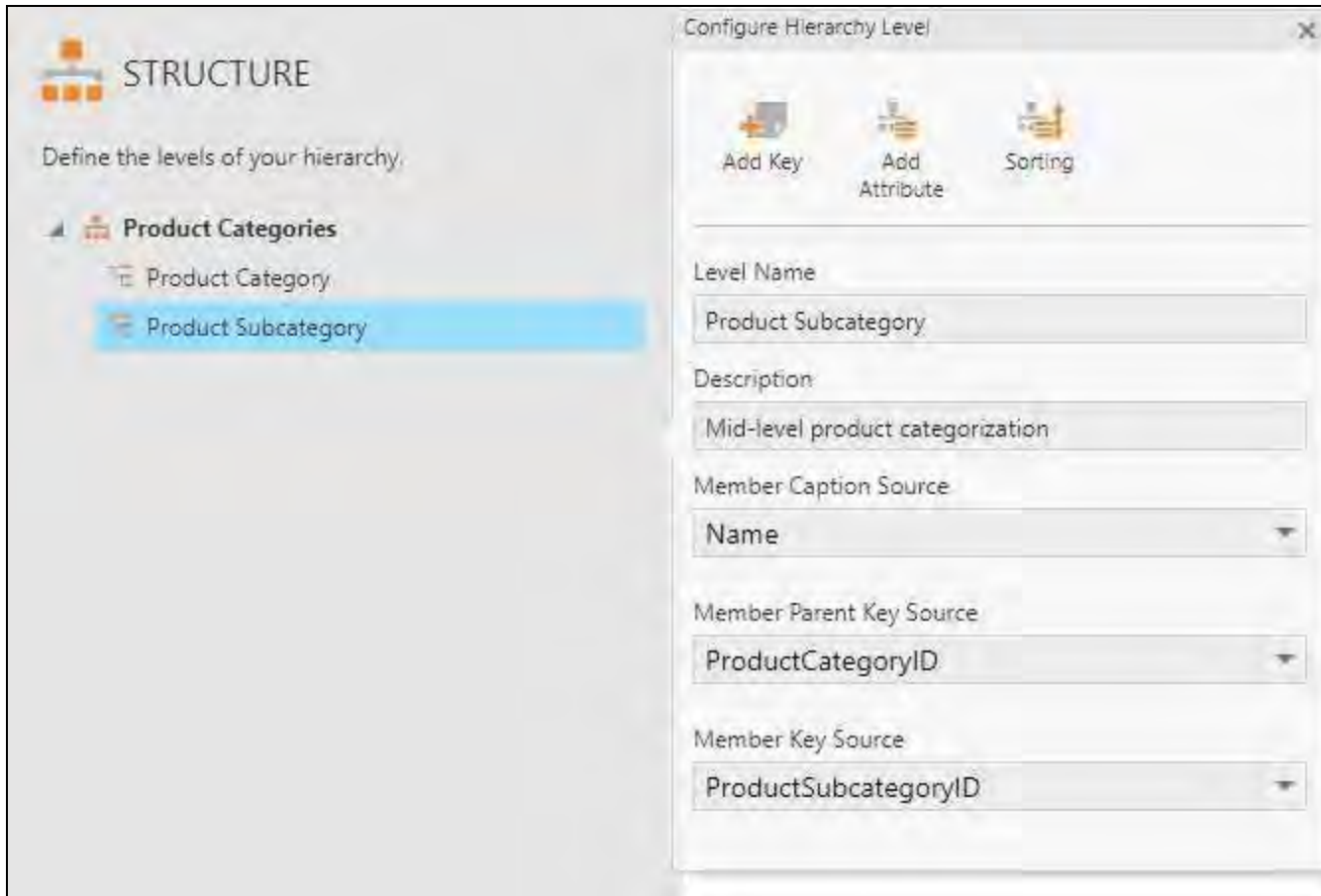


Drag and drop the data source for the second level (e.g., *ProductSubcategory\$*) onto *drop a data source*.



Confirm or update the level's details. The settings are similar to those of the top level but there is an additional one:

- **Member Parent Key Source** - The values in this level's data source that can be used to find the matching member in the level above. The values you choose should match the values of the **Member Key Source** of the upper level.

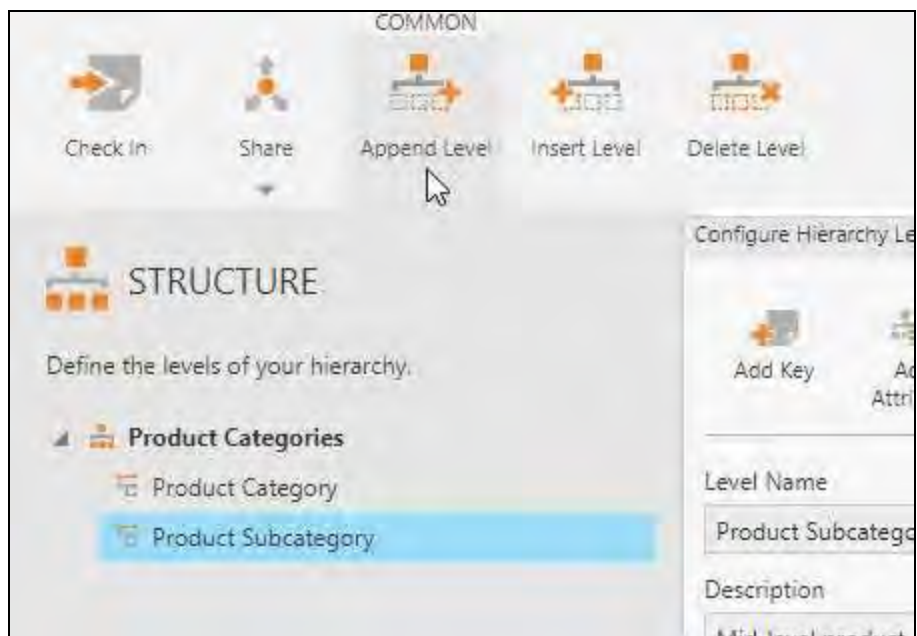


In our example, the data source for the subcategory level contains *ProductCategoryID* values that match the category level's member key values. These are selected as the **Member Parent Key Source**.

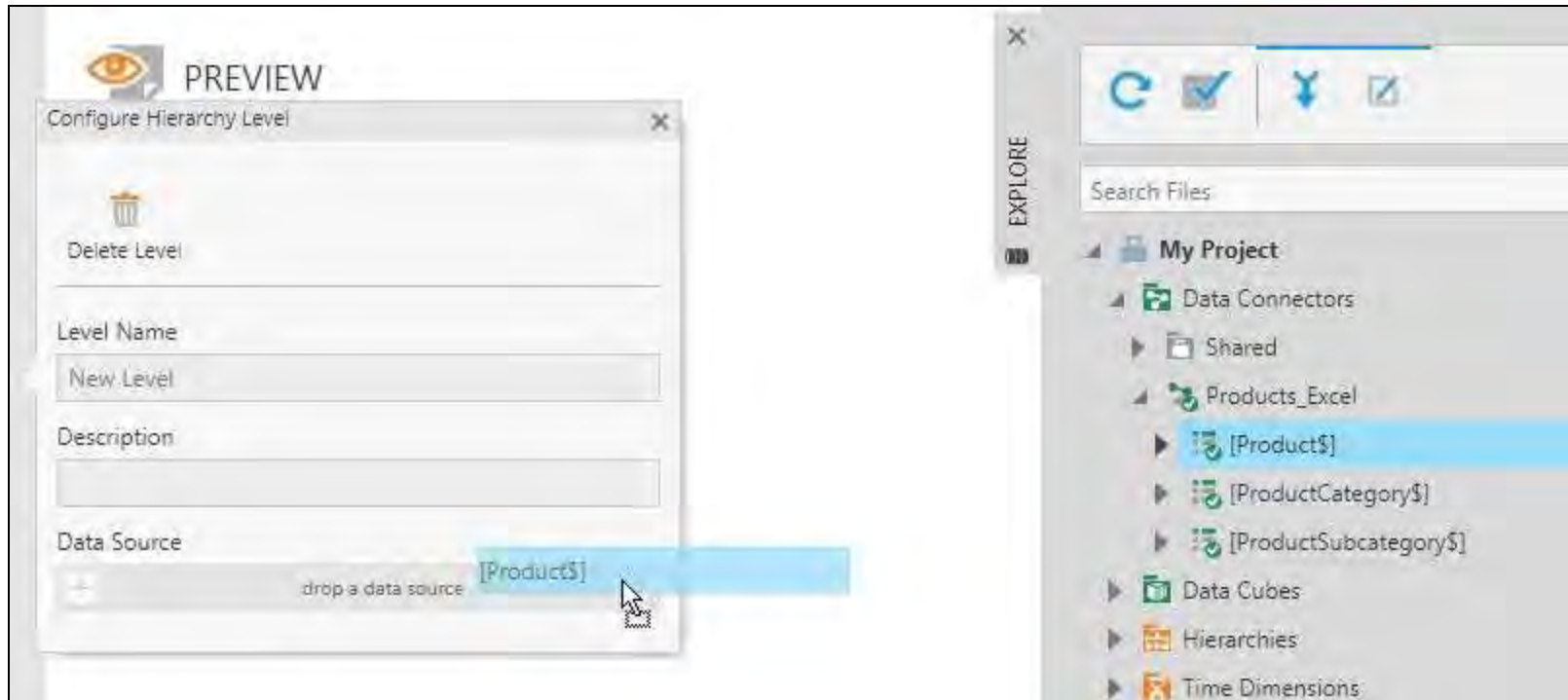
We use *ProductSubcategoryID* values as **Member Key Source** to uniquely identify each subcategory and match with data from the lower level.

Add the Third Level

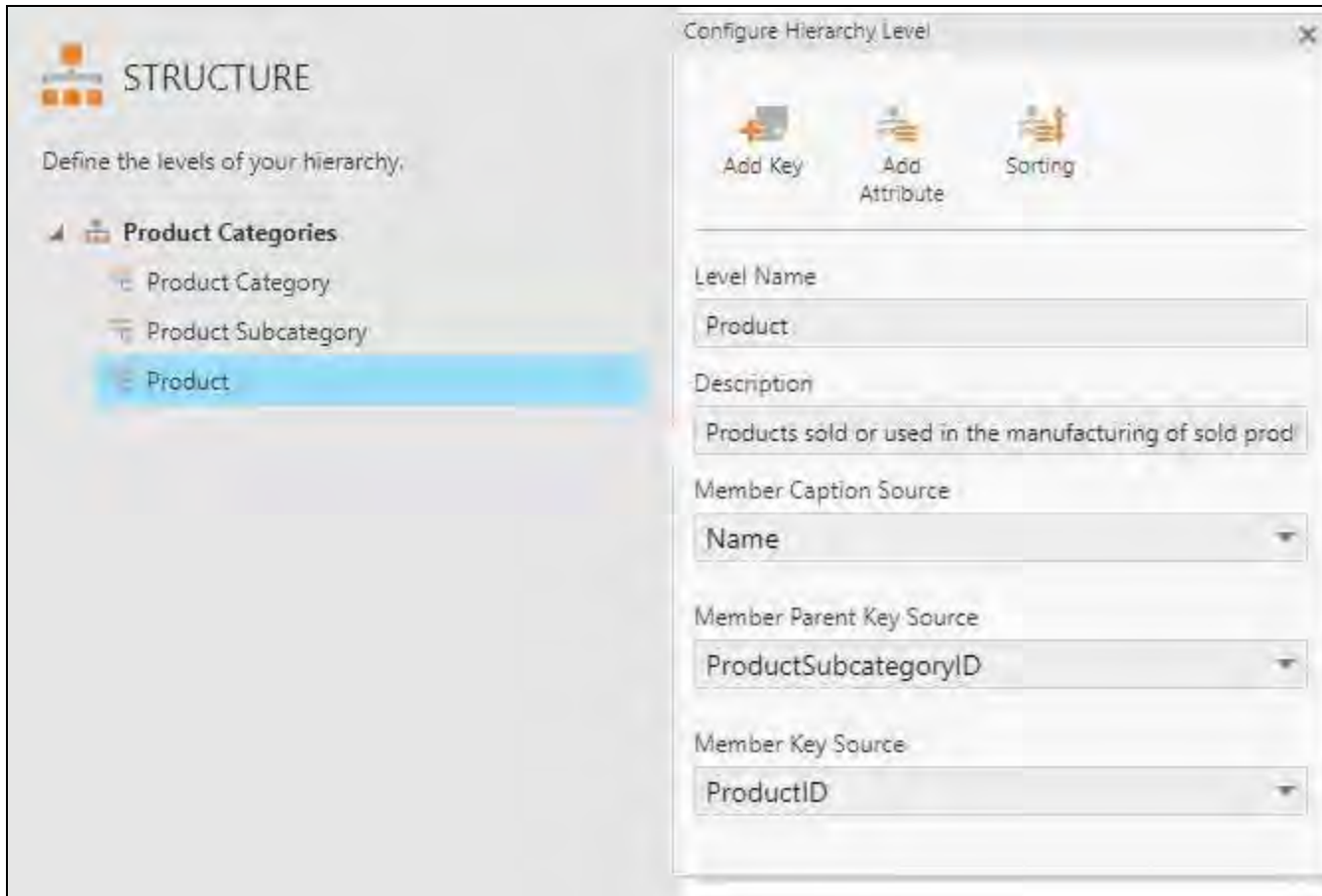
To add a third level, select the second and then choose **Append Level** in the toolbar.



In our example, we drag and drop the *Product\$* sheet onto *drop a data source*.



Set up the level as described previously but for this level's data source.



In our example, the data source for the product level contains *ProductSubcategoryID* values that match the subcategory level's member key values, which are chosen as the **Member Parent Key Source**.

The unique set of *ProductID* values are selected as the **Member Key Source**, allowing this hierarchy to be used with any data source that has matching product IDs.

Preview

Click the **x** button to close the level popup if one is open. Under *Preview*, expand the members to preview the multi-level hierarchy.



STRUCTURE

Define the levels of your hierarchy.

-  **Product Categories**
 -  Product Category
 -  Product Subcategory
 -  Product



PREVIEW

Explore your hierarchy.

-  **(All)**
 -  Accessories
 -  Bike Racks
 -  Bike Stands
 -  Bottles and Cages
 - Mountain Bottle Cage
 - Road Bottle Cage
 - Water Bottle - 30 oz.
 -  Cleaners
 -  Fenders
 -  Helmets
 -  Hydration Packs
 -  Lights
 -  Locks
 -  Panniers
 -  Pumps
 -  Tires and Tubes
 -  Bikes
 -  Clothing
 -  Components

Using the Hierarchy

This hierarchy can now be used to replace a regular column of data with values that match the **Member Key Source** of a level, such as product IDs, so that you can take advantage of your multiple levels or other settings applied in the hierarchy. You can do this [in a metric set](#) when working with data directly from a data connector, or in the [process result](#) of a data cube.

Note: When replacing data with a hierarchy level that uses multiple member key source values to uniquely identify its members, use the hierarchy in the [process result](#) of a data cube where you are able to choose which values correspond to each member key.

For more information, see:

- [Automatic Joins and Hierarchies](#)
- [Add a Relationship Between Tables](#)
- Transform: [Process Result](#)



Promoting an Auto-Generated Metric Set

This applies to: *Managed Dashboards, Managed Reports*

You can analyze data by simply dragging data from data connectors or data cubes directly onto the canvas of a dashboard or other view, so that you don't need to create a metric set ahead of time as a separate step, and to easily work with multiple metric sets together. This is handled by creating automatically generating a metric set for you.

These metric sets are part of the dashboard or view itself, so they can only be found under that view in the Explore window rather than the *Metric Sets* folder, and they share a [revision history](#) with the rest of the dashboard/view. You can promote a metric set to the Metric Sets folder to make it easier to share and reuse it.

If you drag data directly from a data connector rather than a data cube, this automatically generates a data cube behind the metric set in a similar way. This way, you can always choose to take advantage of all of the options in the full [data model](#) later even if you start out by quickly dragging some data onto a dashboard or metric set.

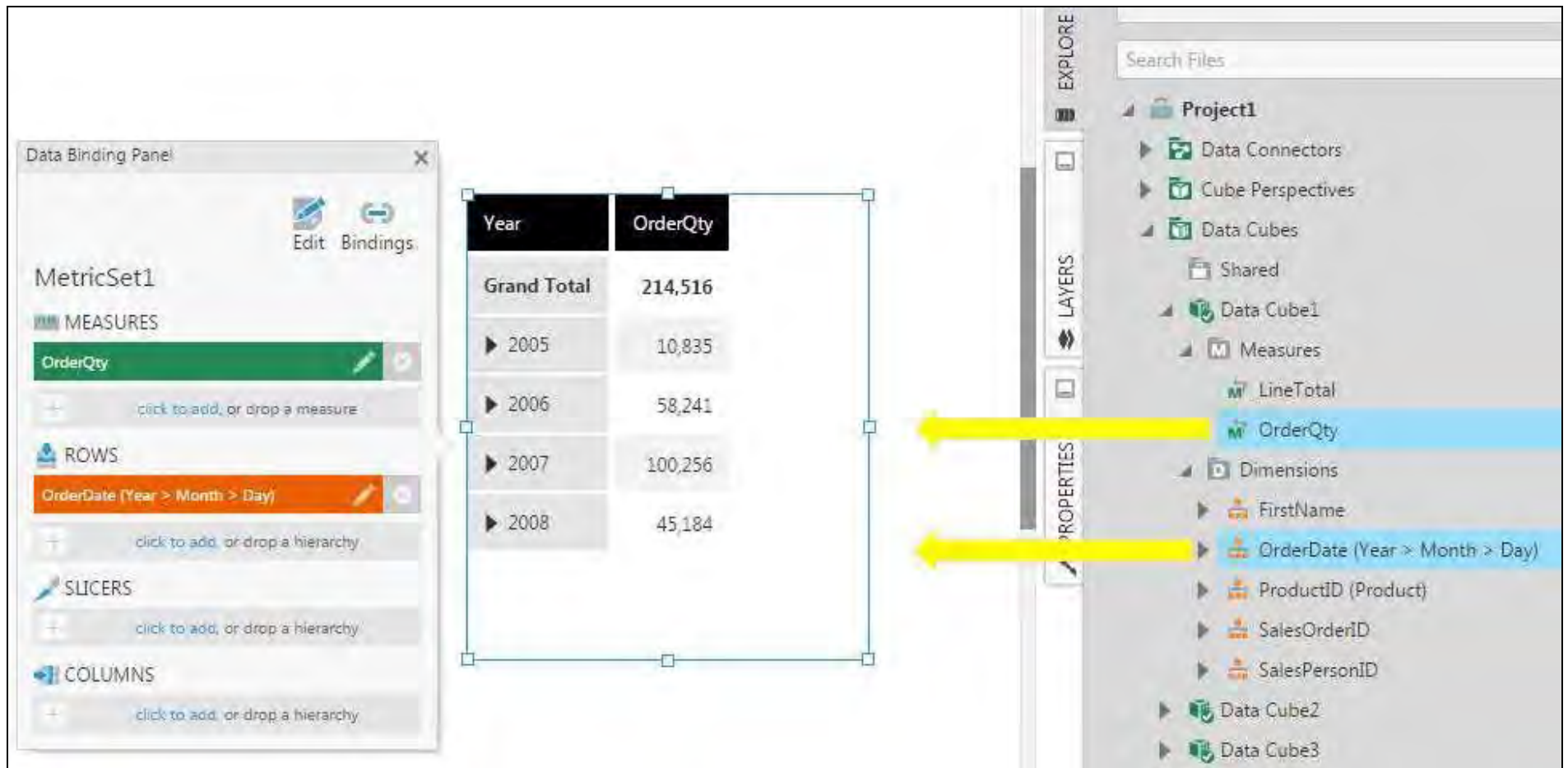
Creating a Metric Set in a Dashboard

As an example, first create a new dashboard from the main menu. Then perform some basic analysis as follows.

Go to the **Explore** window and drag a measure column from a data cube (or data connector) to the canvas. A table visualization appears which shows a single aggregated measure value.

Drag a dimension column from the data cube to the table visualization. The table is reconfigured to show the measure value for each row.

This visualization is powered by an underlying, auto-generated metric set.

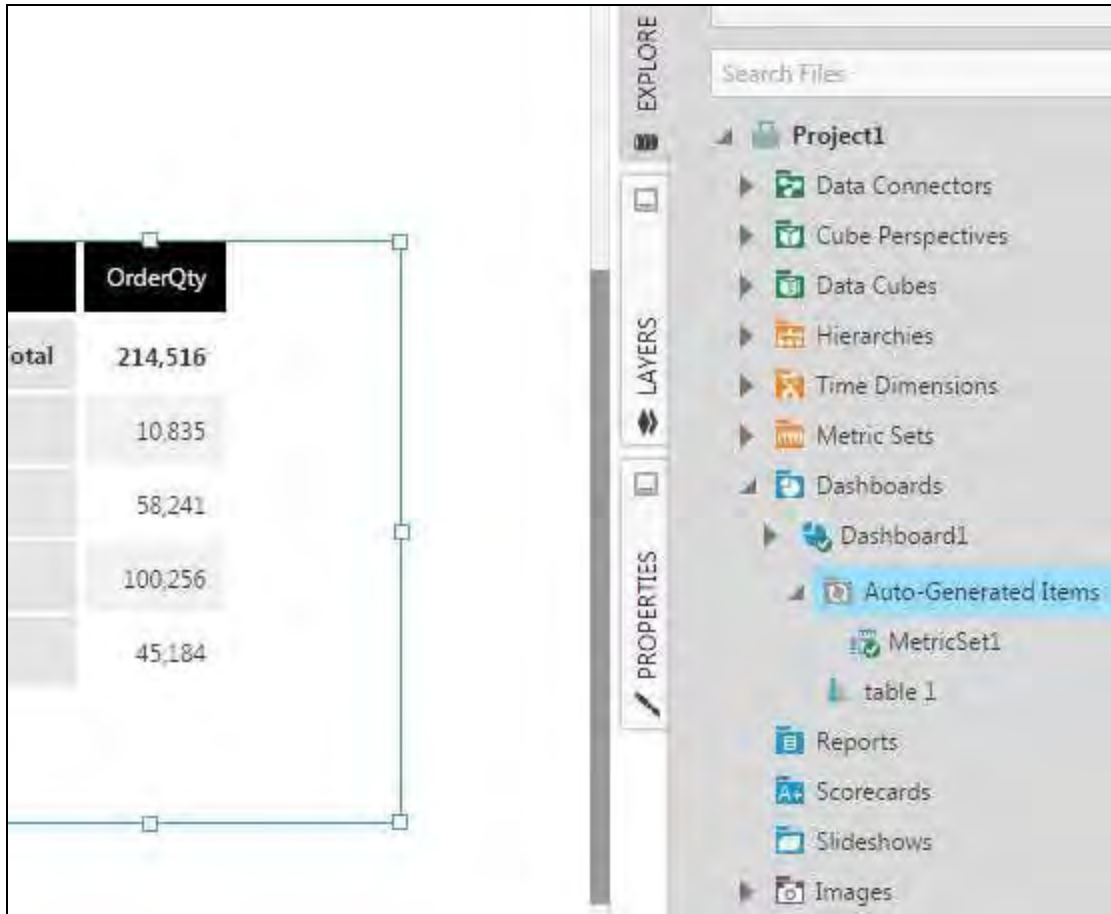


The screenshot displays the Logi Symphony v24 interface. On the left, the **Data Binding Panel** is open, showing the configuration for **MetricSet1**. It includes sections for **MEASURES** (with **OrderQty** selected), **ROWS** (with **OrderDate (Year > Month > Day)** selected), **SLICERS**, and **COLUMNS**. In the center, a table displays the data:

Year	OrderQty
Grand Total	214,516
▶ 2005	10,835
▶ 2006	58,241
▶ 2007	100,256
▶ 2008	45,184

On the right, the **EXPLORE** window shows a tree view of the project structure. The **OrderQty** measure and the **OrderDate (Year > Month > Day)** dimension are highlighted with blue bars and yellow arrows pointing to the table headers, indicating their binding to the table columns.

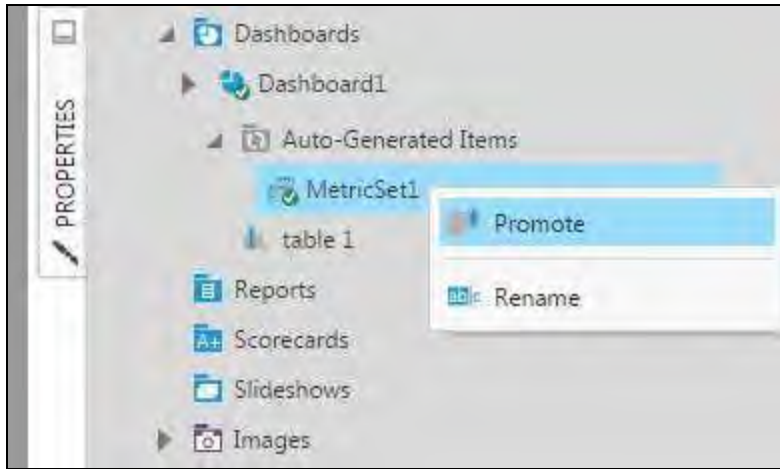
Next, go to the **Explore** window, locate your dashboard, and expand it. You should see an **Auto-Generated Items** folder appear under your dashboard which contains *Metric Set 1*.



Promote the Metric Set

This auto-generated metric set means you can always choose later to move it to the regular *Metric Sets* folder to make it easier for you or your team to find it, even if you started by quickly dragging some data onto the canvas.

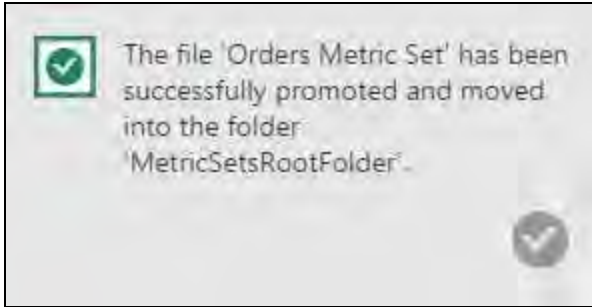
To promote the metric set, right-click (long-tap) on it in the **Auto-Generated Items** folder under its dashboard or view. From the menu, choose **Promote**.



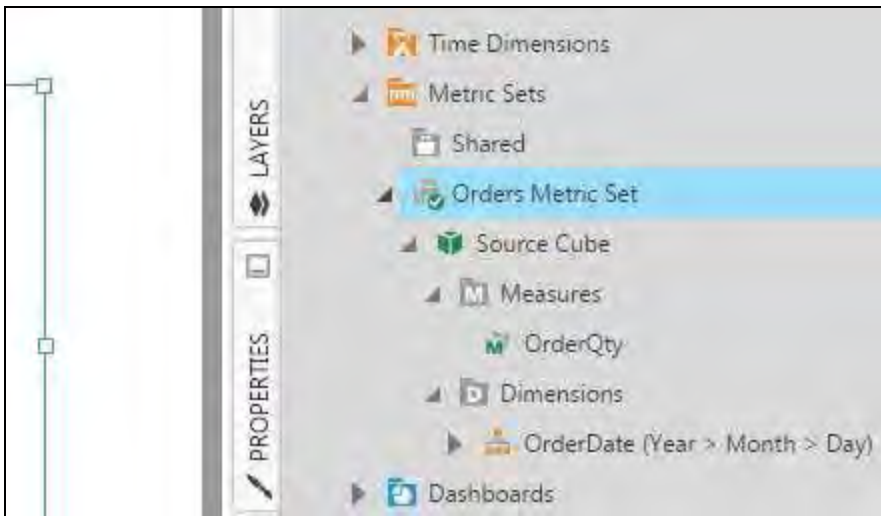
In the *Save File* dialog, optionally edit the **File Name** for the promoted metric set, and select the **Location** where it should be placed. Then click the submit button at the bottom of the dialog.



You'll see a message indicating the metric set has been promoted (and moved).



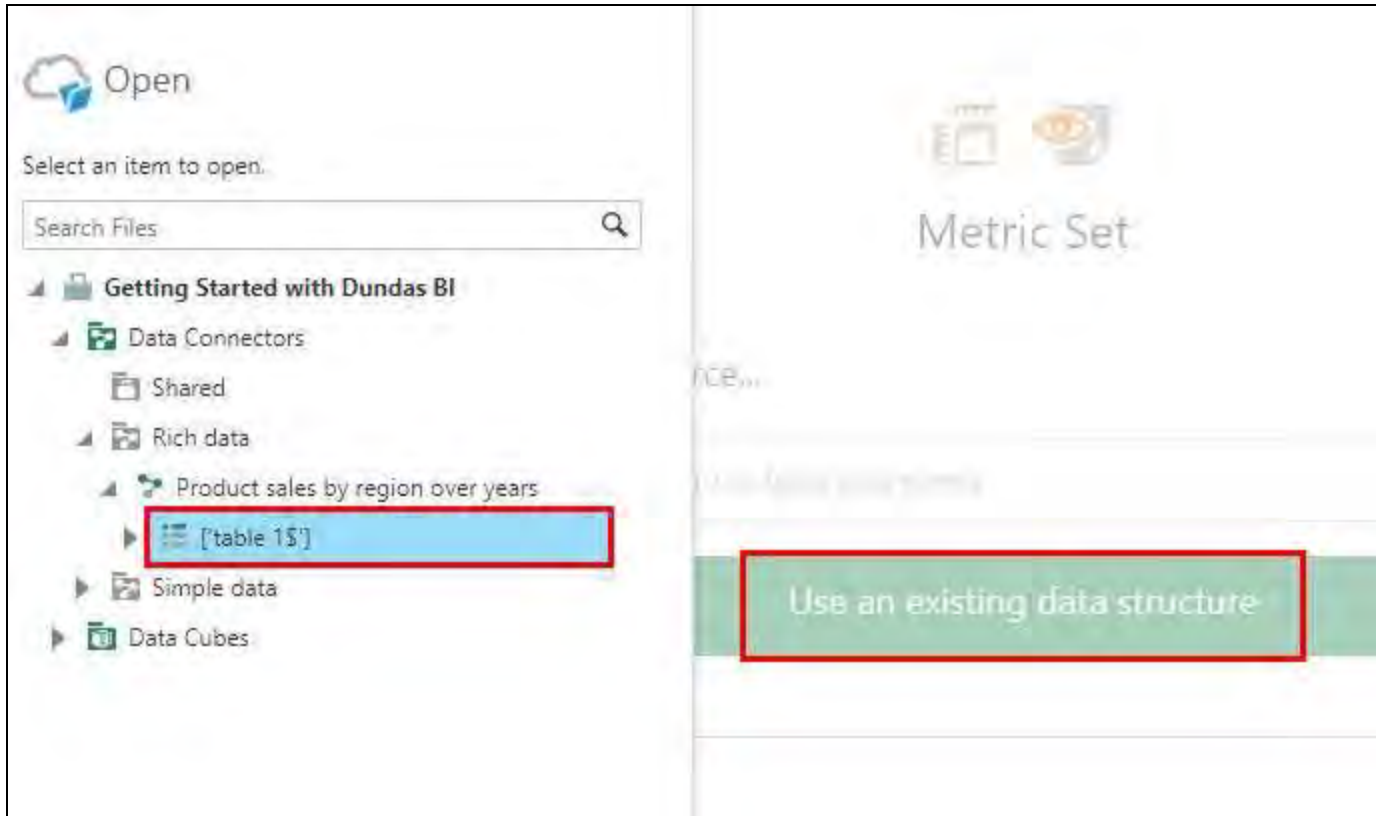
Go to **Metric Sets** in the main menu, or expand the *Metric Sets* folder in the **Explore** window to see the promoted metric set, which is now easier to find and reuse. You can now check in this metric set to give others access to it as well. Also note that the dashboard in this example no longer has an Auto-Generated Items folder.



Auto-Generated Data Cubes

In the same way, when dragging data directly from a data connector from a table or another structure besides a cube, a data cube is generated automatically.

For example, create a new metric set and drag or use data directly from a data connector or an Excel spreadsheet.

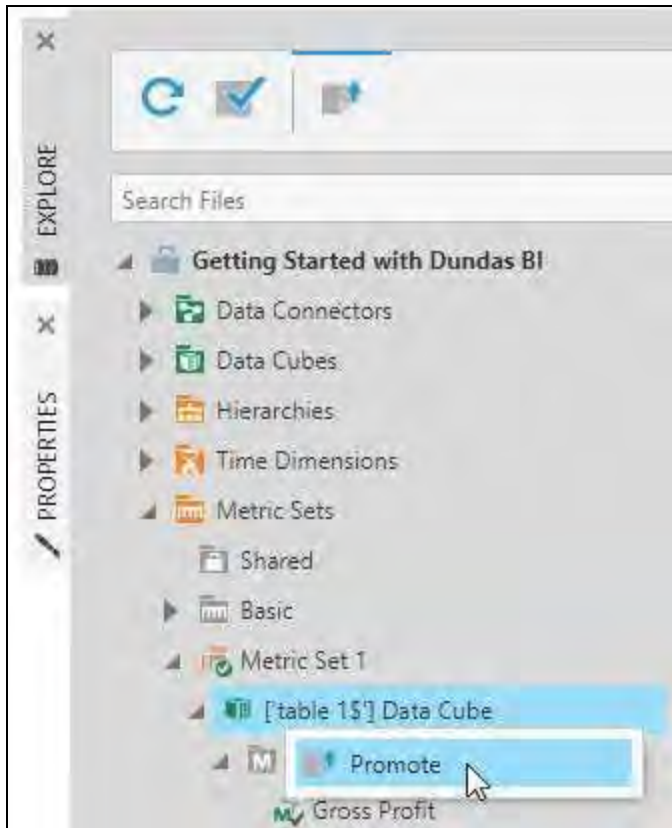


If you clicked **Use an existing data structure**, the **Explore** window now displays the auto-generated data cube based on your data source. All metric sets have a data cube like this one behind the scenes if you dragged data from a data connector without an existing cube.

Click the project icon in the Explore window toolbar to view the current project if it currently displays only your new cube.



Expand your metric set under the project's *Metric Sets* folder in the Explore window to find the data cube. You can right-click this cube to **Promote** it.



Follow steps just like when promoting a metric set to promote this data cube to the *Data Cubes* folder. This makes the data cube easier to share and reuse, or you can then [edit the data cube](#) to transform your data or use other data cube options.

When dragging data onto a metric set directly from multiple data connectors, they are automatically joined within the auto-generated data cube based on relationships, using a [Join](#) transform. If you promote this data cube afterward, you can see and edit this join as shown in the article [Add a relationship between tables](#).

For more information, see:

- [The Managed Dashboards Data Model](#)

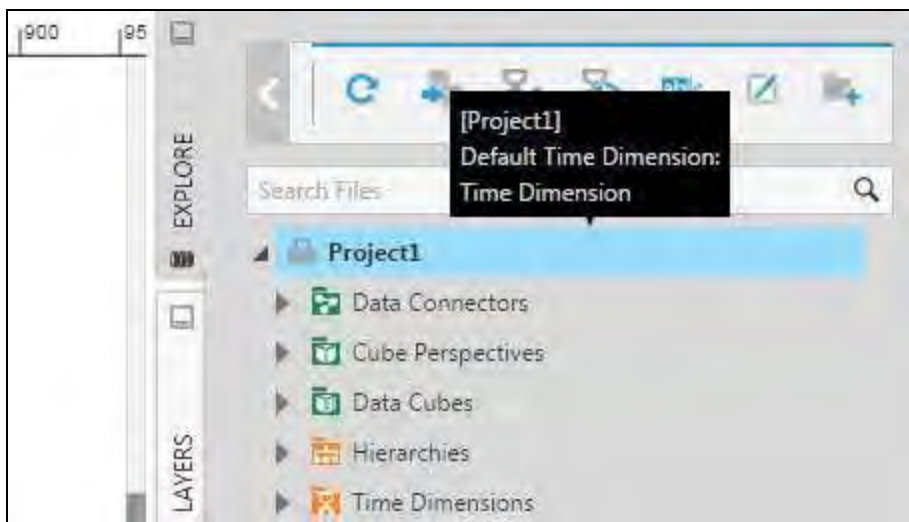
Set or Reset the Default Time Dimension

This applies to: *Managed Dashboards, Managed Reports*

Every project in Symphony has a built-in time dimension, named simply *Time Dimension*, which is initially set as the default for the project. If you create additional time dimensions in a project (e.g., using a fiscal calendar system), you can set one of them as the default for the project instead.

View the Default Time Dimension

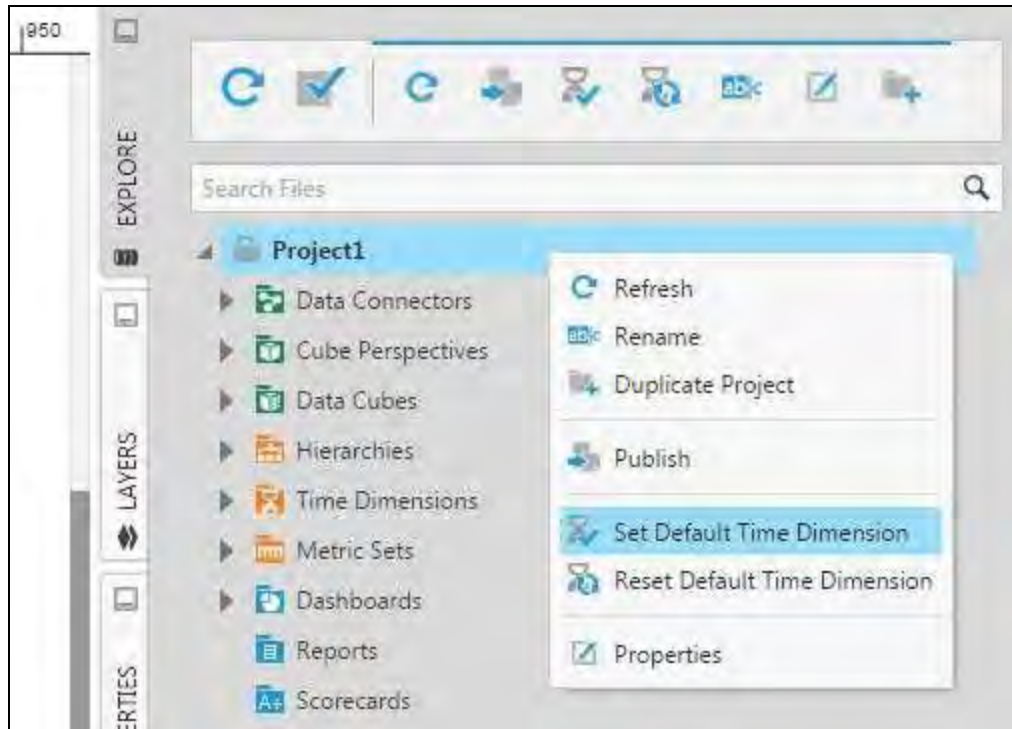
To see which time dimension is the default for the active project, open the **Explore** window and hover over the project name. The project's tooltip shows the name of the default time dimension.



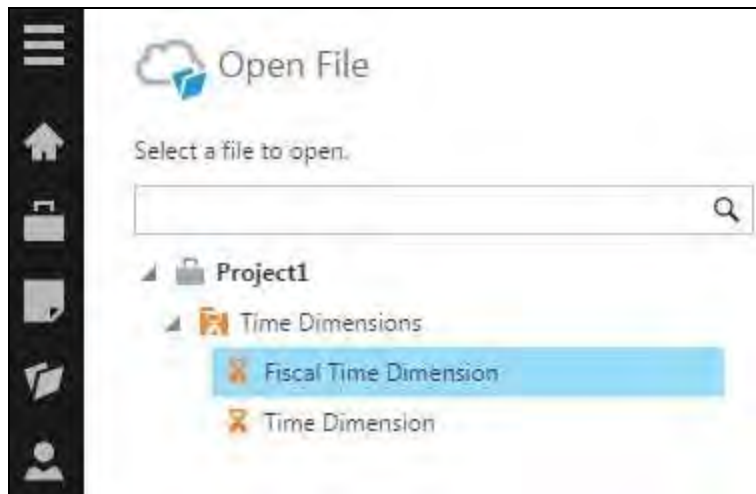
Note: If the tooltip does not show the default time dimension, it means that it was reset previously. When the default time dimension for a project is reset, the default from the global project will be used instead.

Set the Default Time Dimension

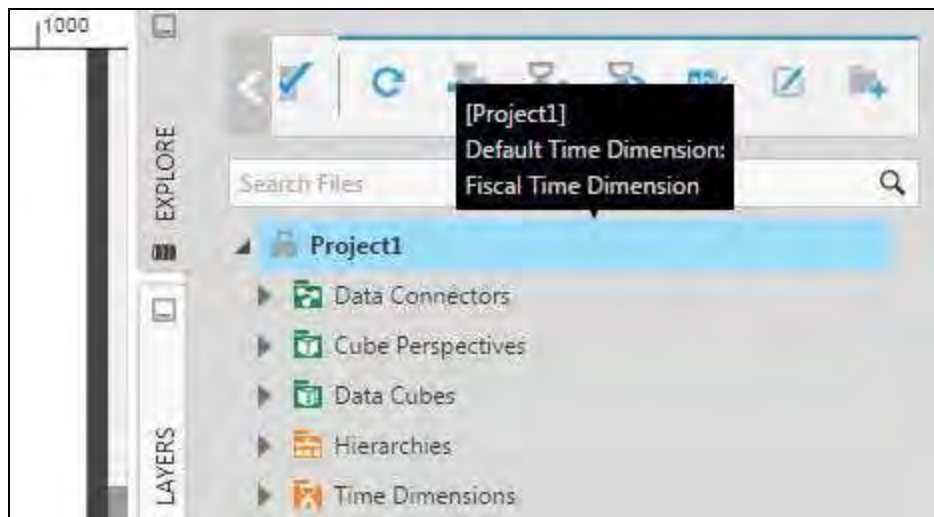
To set the default time dimension for a project, right-click on the project name in the **Explore** window, and then select **Set Default Time Dimension**.



In the *Open File* dialog, select the time dimension you want as the new default, and then click **Submit**.



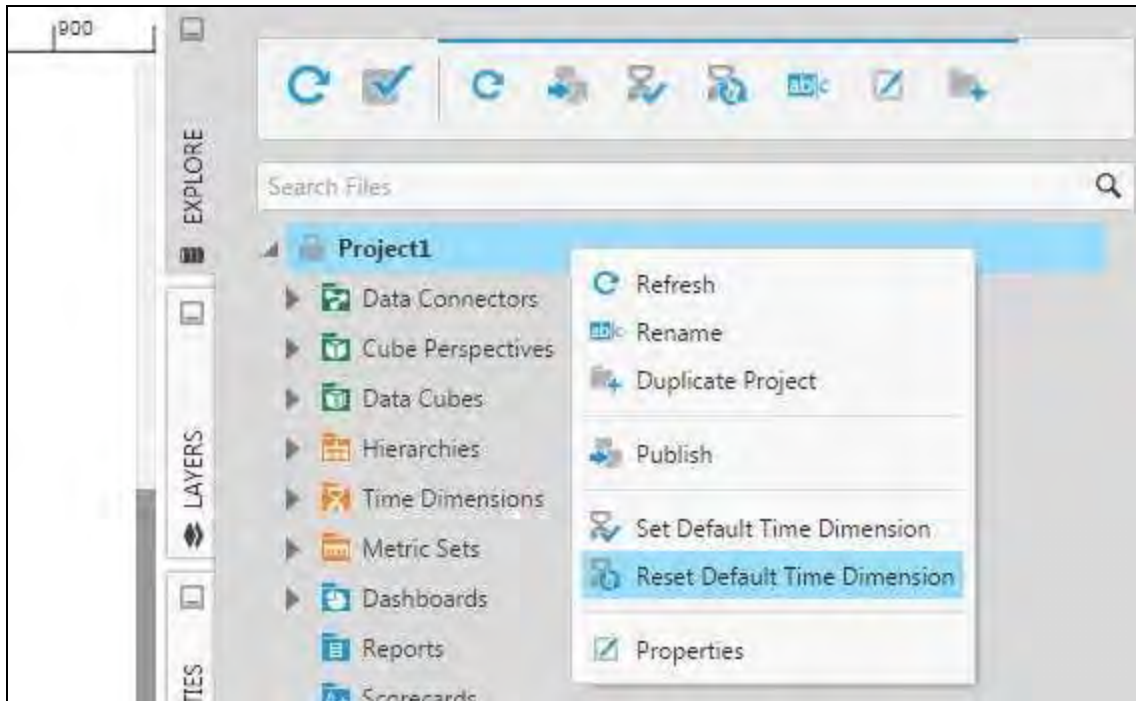
View the tooltip for the project name in the Explore window and verify that the default time dimension has been changed.



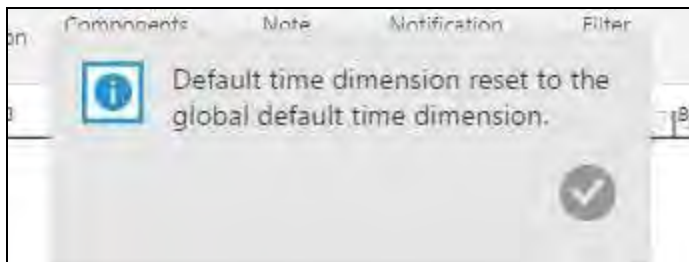
Reset the Default Time Dimension

To reset the default time dimension for a project, right-click on the project name in the **Explore** window, and then select **Reset Default Time Dimension**.

Note that this option is not available for the global project.



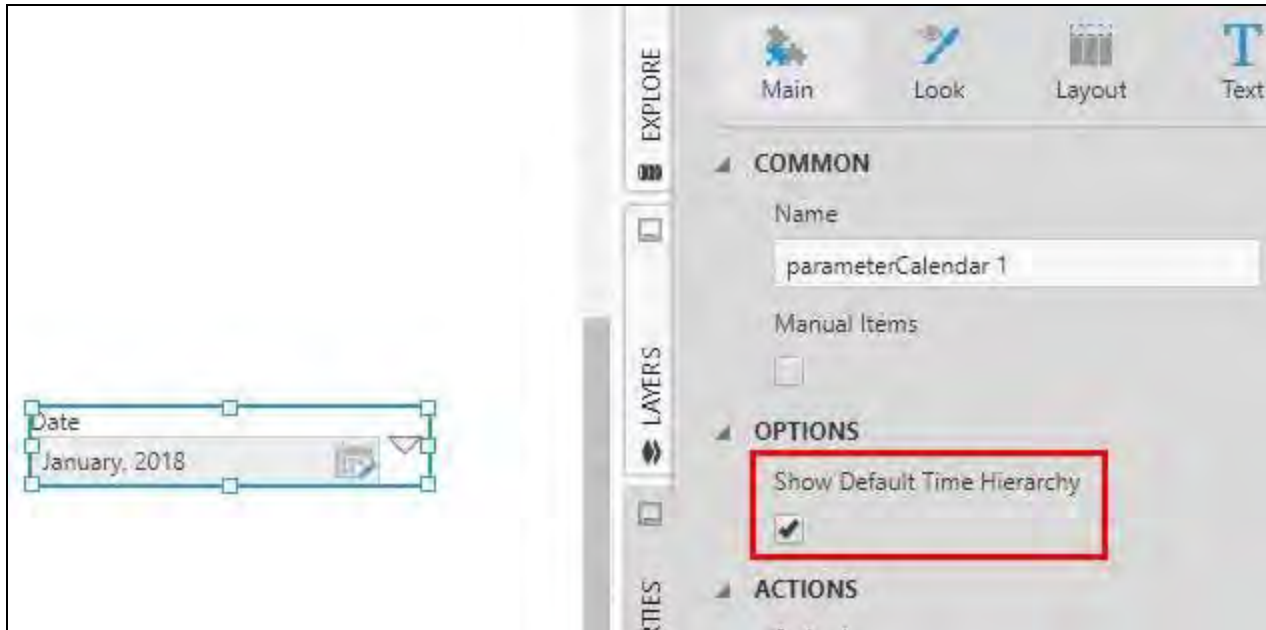
You'll see a message indicating the default time dimension was reset to the default time dimension from the global project.



View the tooltip for the project name in the Explore window and verify that the default time dimension is no longer indicated. This means that the default from the global project will be used.

Using the Default Time Dimension

The default time dimension for a project is used by the Calendar and Calendar Range filters which have a property **Show Default Time Hierarchy**. When this property is enabled, the filter displays the hierarchy from the project's default time dimension instead of the hierarchy passed from the visualization.



For more information, see:

- [Create a Time Dimension](#)
- [Working With Projects](#)
- [Automatic Joins and Hierarchies](#)

Setting Up Data Input

This applies to: *Managed Dashboards, Managed Reports*

Symphony includes the ability to add and modify data within the application, and to store it in its warehouse data storage area. This data can then be used as a data source.

Since you are inputting the data and it is stored in the Symphony warehouse, you do not need to use a data connector for this functionality. Other than the lack of an explicit data connector, data input follows the usual [Data Model](#) that relies on data cubes and metric sets to display data on dashboards, reports, and scorecards.

When you want to input a new set of data, you have three options:

- Create a data cube with a [Data input transform](#). You can then use the data cube to create metric sets, or as a storage area for data input by viewers using a [data input interaction](#).
- Create a new metric set based on data that you enter. This will auto-generate the data cube with a Data Input transform. You can then use the metric set in a view such as a dashboard.
- Create a new visualization in a dashboard or other view and enter new data into it. This will auto-generate a metric set and a data cube with a Data Input transform.

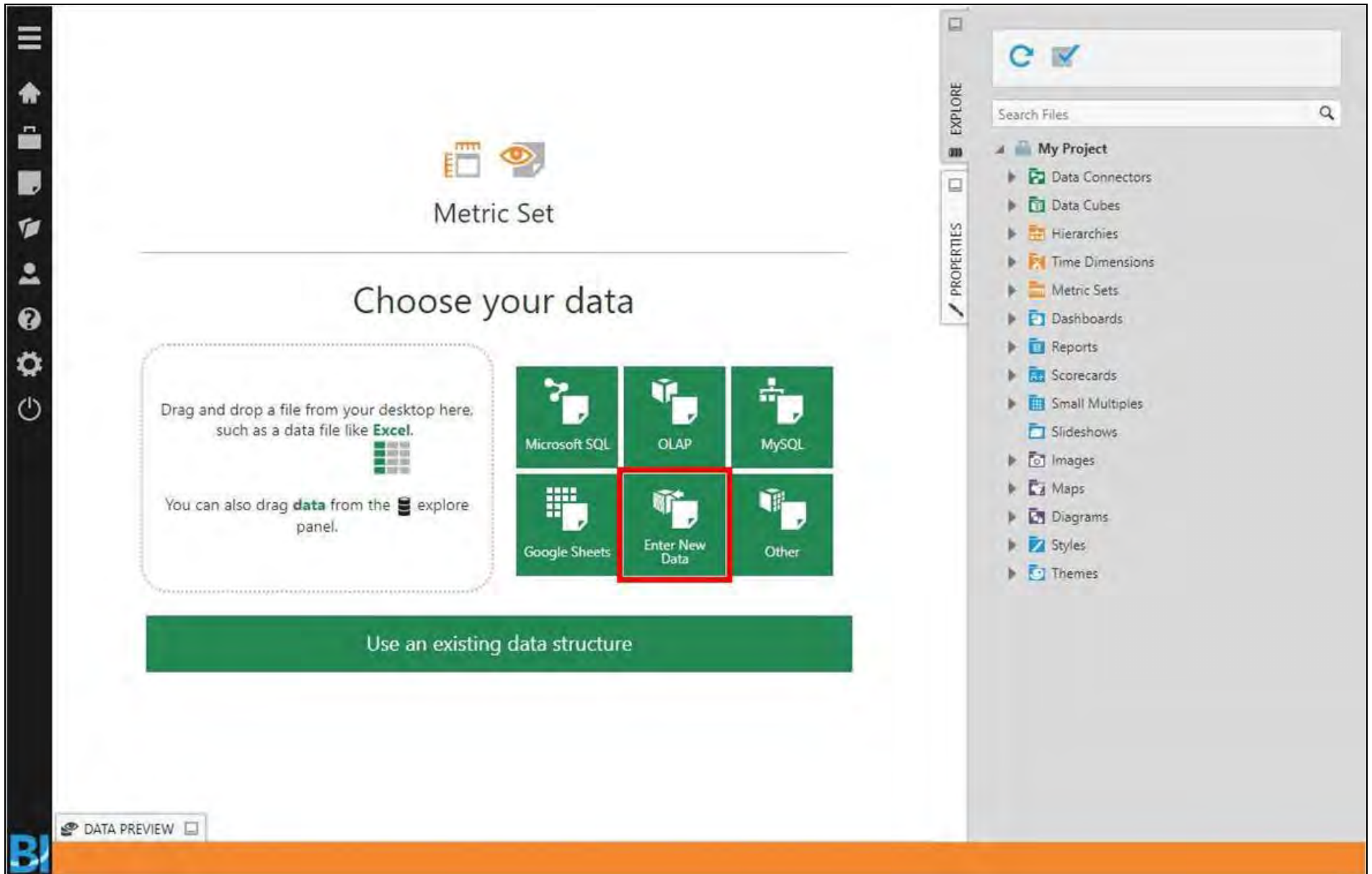
 **Note:** You can always find and [promote auto-generated metric sets and data cubes](#) to the *Data Cubes* or *Metric Sets* folder later.

Data in a Data Cube

You can add a Data Input transform directly to a data cube when editing it. For details, see [Data Input Transform](#). You can then add data to this transform yourself, or set up a [data input interaction](#) for viewers to add to it.

Entering Data Into a New Metric Set

On a new metric set, click **Enter New Data**.



The Data Preview window automatically expands to show the **Data Input Column** dialog. Provide a *Column Name* and select the *Column Data Type*, and then click **Submit**.

Metric Set

Choose your data

Drag and drop a file from your desktop here, such as a data file like **Excel**.



You can also drag **data** from the  explore panel.



EXPLORE
PROPERTIES

DATA PREVIEW

Data Input Column

Add a new column of values of the selected type.

Name

Data Type

Max Length

Default Value



Note: If you indicate a *Default Value*, that value will be automatically populated in that column for each new row.

The Data Input Column dialog is replaced with a single-column table in the Data Preview window. Type data into the cells to input data. Type Enter to create or move to the next row, and Tab to create or move to the next cell to the right. Right-click to insert a row or create a new column, or move the mouse and click the + icon when it appears where you want to add a row or column.

Data Analysis Panel - Metric Set

Freeze

Refresh Data

Visualization

Metric Set 3

MEASURES

click to add, or drop a measure

ROWS

ProductID

click to add, or drop a hierarchy

SLICERS

click to add, or drop a hierarchy

COLUMNS

click to add, or drop a hierarchy

ProductID

706

DATA PREVIEW

Metric Set 3

Data

	ProductID
1	706

New Column

Insert Row Above

Insert Row Below

Delete Row

Like any other metric set, you can now re-visualize and customize this metric set.

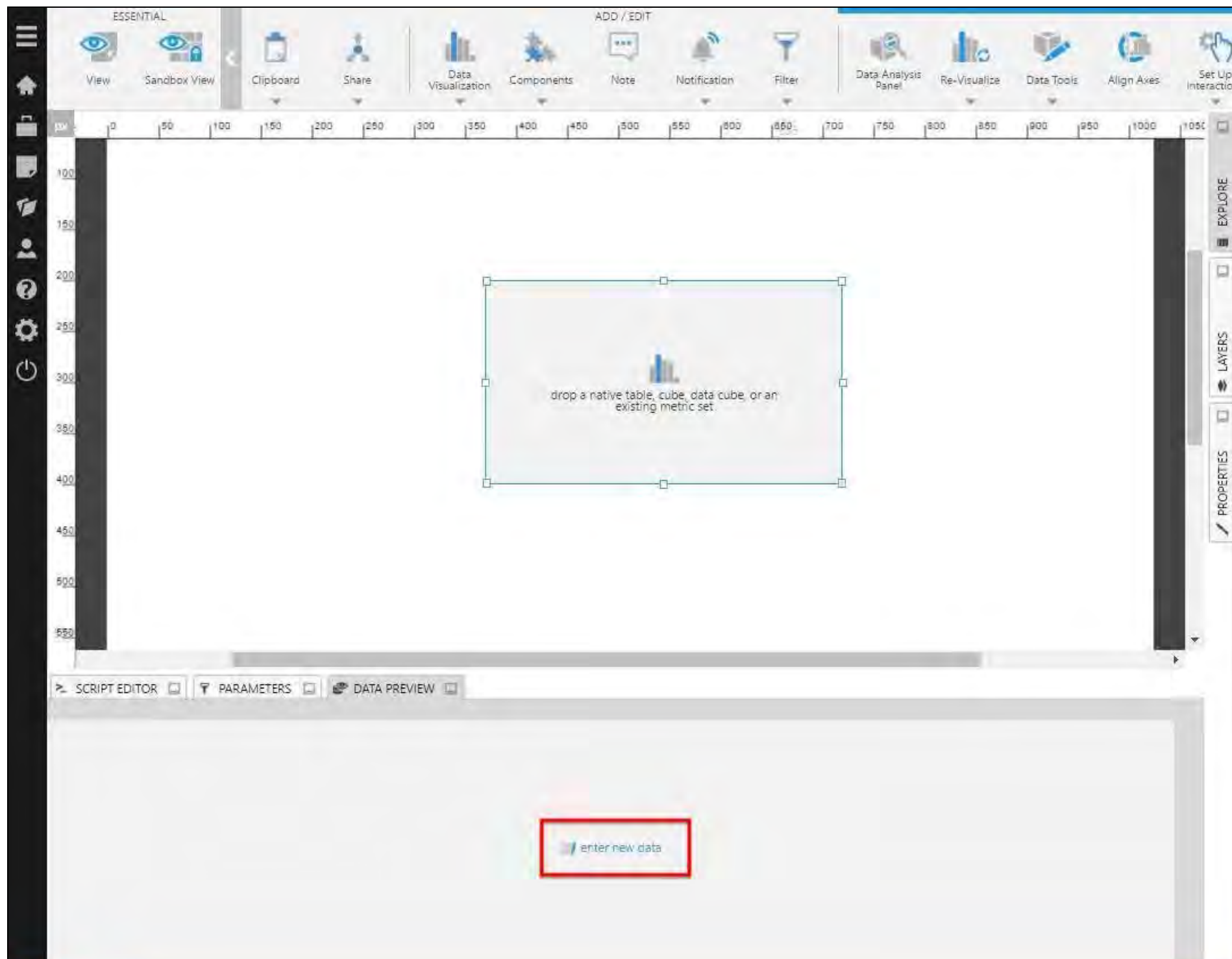


- Archive of documentation for Logi Symphony v24

Entering Data When Editing a Dashboard

When editing a dashboard, add a new data visualization from the toolbar (for example, a bar chart).

Open the Data Preview window and click **enter new data**.

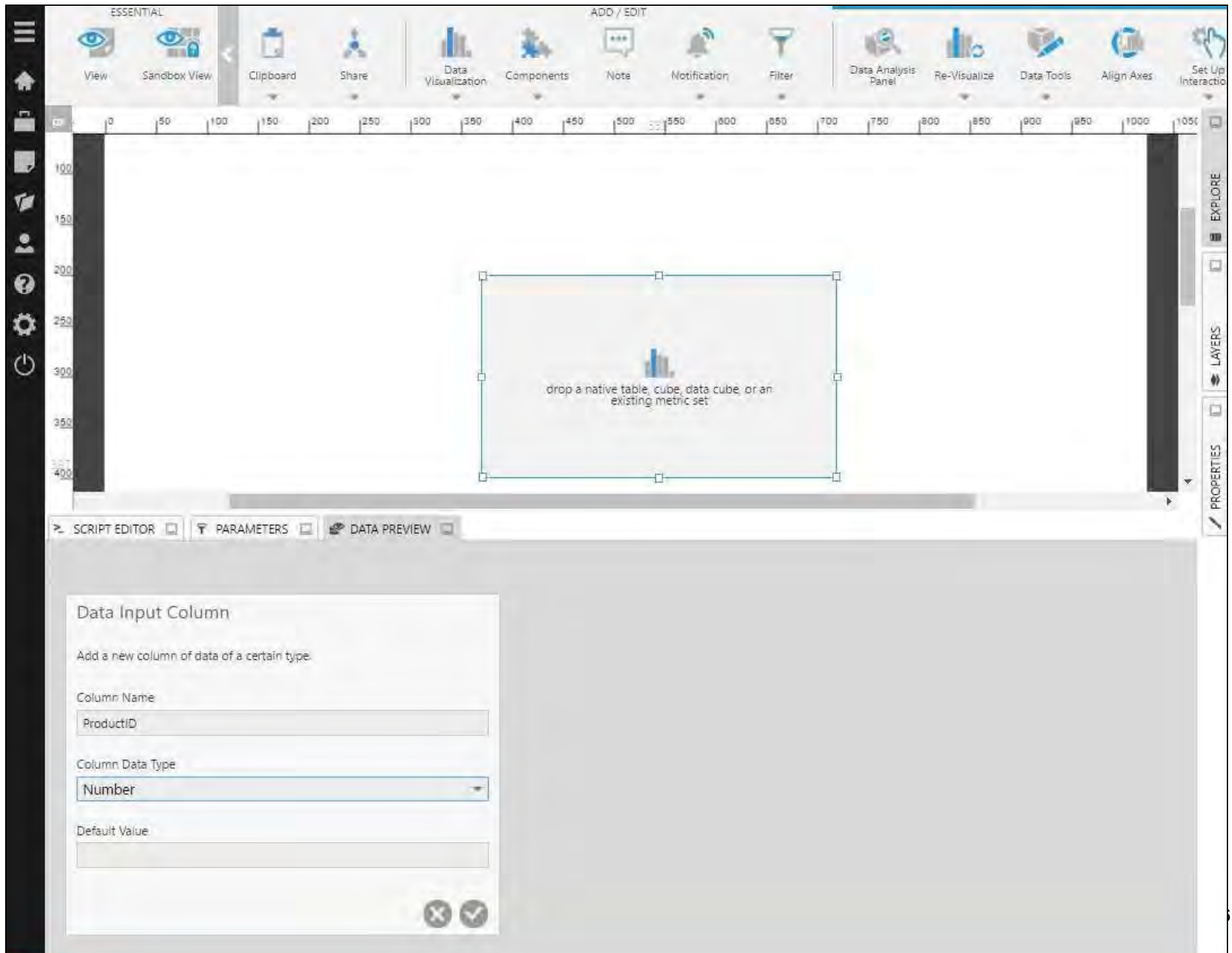


The screenshot displays the Logi Symphony v24 interface. At the top, there is a toolbar with two main sections: 'ESSENTIAL' and 'ADD / EDIT'. The 'ESSENTIAL' section includes 'View' and 'Sandbox View'. The 'ADD / EDIT' section includes 'Clipboard', 'Share', 'Data Visualization', 'Components', 'Note', 'Notification', 'Filter', 'Data Analysis Panel', 'Re-Visualize', 'Data Tools', 'Align Axes', and 'Set Up Interaction'. Below the toolbar is a horizontal axis with numerical values from 0 to 1050. The main canvas area is a large white rectangle with a central placeholder for a visualization. The placeholder contains a small bar chart icon and the text: 'drop a native table, cube, data cube, or an existing metric set'. On the right side of the canvas, there is a vertical axis with numerical values from 100 to 550. At the bottom of the interface, there is a panel with three tabs: 'SCRIPT EDITOR', 'PARAMETERS', and 'DATA PREVIEW'. The 'DATA PREVIEW' tab is currently selected, and it contains a red-bordered button labeled 'enter new data'.



- Archive of documentation for Logi Symphony v24

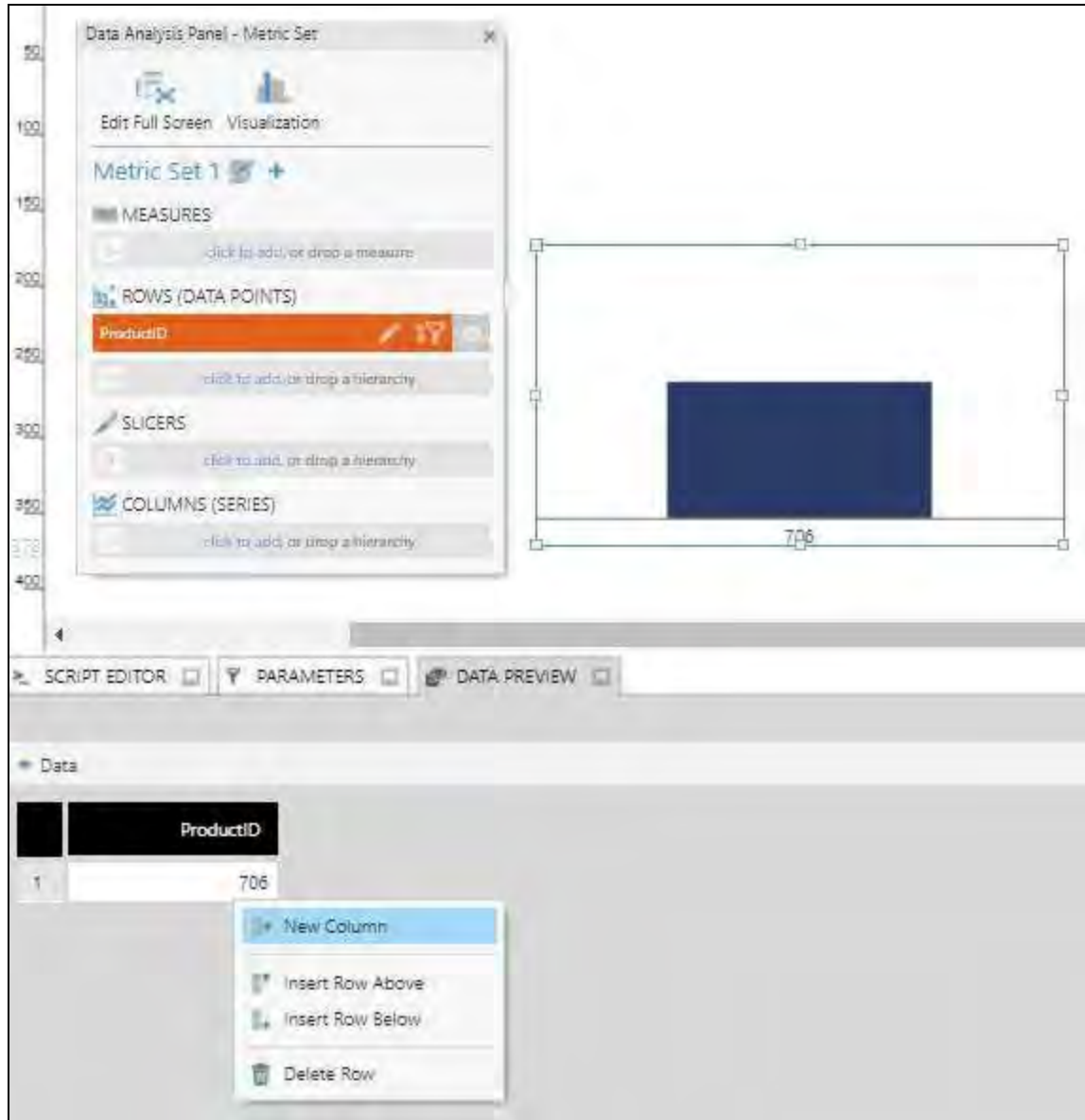
The Data Preview window automatically expands to show the **Data Input Column** dialog. Provide a *Column Name* and select the *Column Data Type*, and then click **Submit**.



The screenshot displays the Logi Symphony v24 interface. At the top, there is a toolbar with two main sections: 'ESSENTIAL' and 'ADD / EDIT'. The 'ESSENTIAL' section includes icons for 'View', 'Sandbox View', 'Clipboard', and 'Share'. The 'ADD / EDIT' section includes icons for 'Data Visualization', 'Components', 'Note', 'Notification', 'Filter', 'Data Analysis Panel', 'Re-Visualize', 'Data Tools', 'Align Axes', and 'Set Up Interaction'. Below the toolbar is a horizontal axis with numerical values from 0 to 1050. The main workspace is a large white area with a central gray box containing a bar chart icon and the text 'drop a native table, cube, data cube, or an existing metric set'. On the right side, there is a vertical sidebar with buttons for 'EXPLORE', 'LAYERS', and 'PROPERTIES'. At the bottom, there is a tabbed interface with 'SCRIPT EDITOR', 'PARAMETERS', and 'DATA PREVIEW' tabs. The 'DATA PREVIEW' tab is active, showing a 'Data Input Column' dialog box. This dialog box has a title 'Data Input Column' and a subtitle 'Add a new column of data of a certain type.' It contains three input fields: 'Column Name' with the value 'ProductID', 'Column Data Type' with a dropdown menu showing 'Number', and 'Default Value' with an empty text box. At the bottom right of the dialog box are two circular buttons, one with an 'X' and one with a checkmark.

The Data Input Column dialog is replaced with a single-column table in the Data Preview window. Type data into the cells to input data. Type Enter to create or move to the next row, and Tab to create or move to the next cell to the right.

Right-click to insert a row or create a new column, or move the mouse and click the + icon when it appears where you want to add a row or column.



The screenshot displays the Logi Symphony v24 interface. On the left, the 'Data Analysis Panel - Metric Set' is visible, showing a 'Metric Set 1' with a '+ ' icon. Below this, there are sections for 'MEASURES', 'ROWS (DATA POINTS)', 'SLICERS', and 'COLUMNS (SERIES)'. The 'ROWS (DATA POINTS)' section is currently selected, showing 'ProductID' as the data point. The 'COLUMNS (SERIES)' section is also visible, showing '706' as the series.

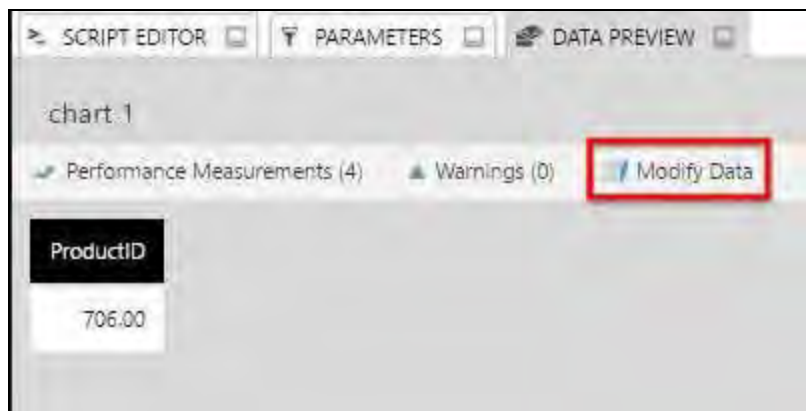
On the right, a visualization area shows a dark blue rectangle representing the data. Below the visualization, a 'Data Preview' window is open, showing a table with the following structure:

	ProductID
1	706

A context menu is open over the table, showing options: 'New Column', 'Insert Row Above', 'Insert Row Below', and 'Delete Row'.

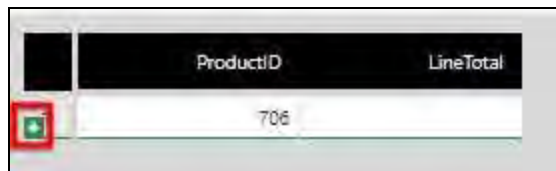
Modify Input Data

No matter where you initially set up data input, when working with this data you can access the editable table again to make changes. To modify input data, open the **Data Preview** window, and click **Modify Data**.



Pasting Data

If you have existing data in another application, you can copy multiple values into the data input table. Copy (press Ctrl+C) some columns and rows of data from the other application, then click a cell in Symphony's Data Preview window and paste (Ctrl+V). The pasting action works from left to right and from the top down, which means the cell you selected should correspond to the top-left cell of your pasted data. If that cell does not exist yet, you can quickly add a new line by hovering over the last cell and clicking the green plus.



Important: Pasting values from more columns than those set up for data input will not create new columns.

Deleting Rows

You can right-click a row and choose **Delete Row**. To delete multiple rows, select them by clicking the numbers on the left while holding Ctrl or Shift on your keyboard (tap the row numbers on touch devices). Right-click (or long-tap) the table when rows are selected and choose **Delete Selected Rows**.

	ProductID	LineTotal
1	706	
2	707	157772
3	708	160869
4	709	6060
5	710	513

- New Column
- Delete Column
- Delete Selected Rows

Working With Data Input

- Deleting a column from the Data Input is postponed until after you check in. This means you will be unable to create a column with the same title as the deleted one until you check in and check out.
- Changes to rows of data are not affected by check in or check out states, since viewers or end users can be allowed to make changes. Undo Check Out does not change or remove data.
- When using the data input feature, you cannot enter data using tokens. Disable the token picker when using it for these purposes.

For more information, see:

- [Database Write Back](#)
- [The Managed Dashboards Data Model](#)
- [Data Input](#)
- [Set Up a Data Input Interaction](#)
- [Promoting an Auto-Generated Metric Set](#)



Set Up States on a Metric Set

This applies to: *Managed Dashboards, Managed Reports*

You can set up states on a metric set to display some data differently when it falls within that state based on comparisons with other values.

They can be used to call attention to certain data or to otherwise customize visualization settings for certain data by setting up its **State Styles**.

States are defined as part of a *state group*, which is a set of related states for evaluating the same measure, such as *Good Sales* and *Bad Sales*. You might have multiple state groups if a single value could represent multiple states, such as a value that is both above the sales target ("Good") and lower than this time last year ("Trending Down").

Related video: [States](#)

Setting Up States

States are saved as part of your [metric set](#) so that they are reused when switching between visualizations or when visualizing a metric set multiple times. (The metric set will need to be [checked out](#) to you to set up states if it has been checked in.)

Create a Metric Set

As an example, create a new metric set from the **main menu**. You can also set up states when editing a metric set on a dashboard or on any other view.

Drag two numeric columns (measures) for comparison purposes.

Drag a column of text or date values (a hierarchy) to **Rows** in the Data Analysis Panel.

	FirstName	SalesLastYear	SalesYTD
	Grand Total	23,685,963.6244	36,277,591.9034
	Amy	0.0000	519,905.9320
	David	1,371,635.3158	1,573,012.9383
	Garrett	1,620,276.8966	1,453,719.4653
	Jae	1,635,823.3967	4,116,871.2277
	Jillian	1,997,186.2037	3,189,418.3662
	José	2,038,234.6549	2,604,540.7172
	Linda	1,439,156.0291	4,251,368.5497
	Lynn	2,278,548.9776	1,421,810.9242
	Michael	1,750,406.4785	3,763,178.1787
	Pamela	1,927,059.1780	1,352,577.1325
	Rachel	1,307,949.7917	1,827,066.7118
	Ranjit	2,396,539.7601	3,121,616.3202
	Shu	2,073,505.9999	2,458,535.6169
	Stephen	0.0000	559,697.5639
	Syed	0.0000	172,524.4512
	Tete	0.0000	1,576,562.1966
	Tsvi	1,849,640.9418	2,315,185.6110

Data Analysis Panel - Metric Set

 Freeze
 Refresh Data
 Visualization
 Data Flow

Metric Set 39

MEASURES

SalesLastYear SUM

SalesYTD SUM

+ click to add, or drop a measure

ROWS

FirstName

+ click to add, or drop a hierarchy

SLICERS

+ click to add, or drop a hierarchy

COLUMNS

+ click to add, or drop a hierarchy

Add a State

To create a state, click **Set Up States** in the toolbar. When editing a dashboard or another view, you can find this option under **Data Tools**.



If this is the first state in this metric set, the *Configure State* dialog appears immediately.

In the **Measure** drop-down, choose which numbers determine your conditions. You can leave this as *(None)* if you are only comparing hierarchy values.

You can enter a preferred **Name** to identify this state.

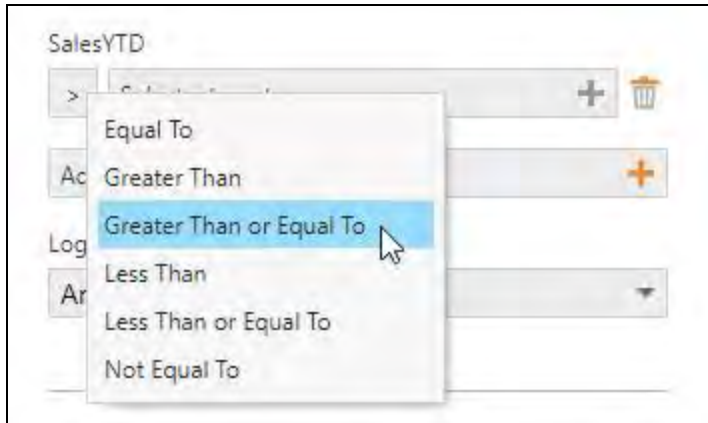
Click **Add condition**.



The 'Configure State' dialog box is shown. It has a title bar with a bar chart icon and the text 'Configure State'. Below the title, there is a description: 'Set up the conditions for this state. Compare against another measure or value, or compare the values of a hierarchy against a value you specify.' The form contains the following fields:

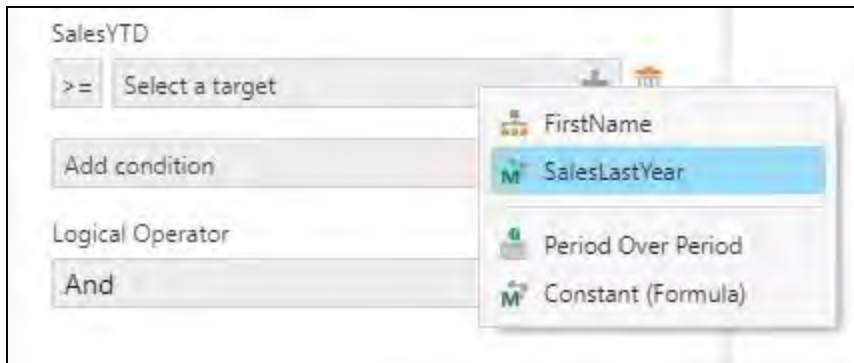
- Measure:** A dropdown menu with 'SalesYTD' selected.
- Name:** A text input field containing 'Good'.
- Description:** A large text area that is currently empty.
- Add condition:** A button with a red border and a plus icon, highlighted with a red rectangular box.
- Logical Operator:** A label for the next field, which is partially visible at the bottom.

To change the operator used to compare against your measure, click the **Greater Than** sign button and choose another option such as *Greater Than* or *Equal To* from the drop-down menu.



Click **Select a target** and choose what to compare against in your data.

This can be another measure or hierarchy you already added to the metric set, including a [formula](#). If you select a constant or a hierarchy (orange icon), you will also enter a specific value to compare against - see the [examples](#) below for details. The [Period Over Period](#) option will open a dialog to add a new comparison against another time period to use for this state.



Click **Submit** to create a new state group with your new state.

For our example below, we will repeat these steps with a *Less Than* comparison operator to create a *Bad* state.

In the *Set Up States* dialog, click **Add a state** near the bottom of the dialog to add another state to this state group.

Set Up States

You can set up states on a metric set to display some data differently when it falls within that state, based on comparisons with other values. This can be used to call attention to certain data, or to otherwise customize visualization settings for certain data by setting up its State Styles.

SalesYTD State Group 1  

 
Preview Visual

State	Condition	
Good	> = SalesLastYear	 

Add a state 

Repeat the steps to configure the state.



Set Up States [more fields](#)

You can set up states on a metric set to display some data differently when it falls within that state, based on comparisons with other values. This can be used to call attention to certain data, or to otherwise customize visualization settings for certain data by setting up its State Styles.

SalesYTD State Group 1   Preview Visual

State	Condition	
Good	>= SalesLastYear	 
Bad	< SalesLastYear	 

Close the **Set Up States** dialog (or select the **Preview** button) to see your visualization updated with default settings reflecting your states.

	FirstName	SalesLastYear	SalesYTD
	Grand Total	23,685,963.6244	36,277,591.9034
	Amy	0.0000	519,905.9320
	David	1,371,635.3158	1,573,012.9383
	Garrett	1,620,276.8966	1,453,719.4653
	Jae	1,635,823.3967	4,116,871.2277
	Jillian	1,997,186.2037	3,189,418.3662
	José	2,038,234.6549	2,604,540.7172
	Linda	1,439,156.0291	4,251,368.5497
	Lynn	2,278,548.9776	1,421,810.9242
	Michael	1,750,406.4785	3,763,178.1787
	Pamela	1,927,059.1780	1,352,577.1325
	Rachel	1,307,949.7917	1,827,066.7118
	Ranjit	2,396,539.7601	3,121,616.3202
	Shu	2,073,505.9999	2,458,535.6169
	Stephen	0.0000	559,697.5639
	Syed	0.0000	172,524.4512
	Tete	0.0000	1,576,562.1966
	Tsvi	1,849,640.9418	2,315,185.6110

Data Analysis Panel - Metric Set

 Freeze

 Refresh Data

 Visualization

 Data Flow

Metric Set 39

MEASURES

SalesLastYear
SUM

SalesYTD
SUM

+
click to add, or drop a measure

ROWS

FirstName

+
click to add, or drop a hierarchy

SLICERS

+
click to add, or drop a hierarchy

COLUMNS

+
click to add, or drop a hierarchy

Add Another State Group

You can set up another state group to check for an independent set of conditions. A value could then be considered to be in more than one state at a time: for example, one state from a group named *Vs. Target* and another from the group *Vs. Last Year*. In the *Set Up States* dialog, click the **+** button next to your existing state group name to add a new one.



After creating the new state group, you can click **Add a state**.

Visualizing States

Each visualization can use states in different ways, so the visual-related settings are located in the **Properties** window for the visualization. Some initial settings will be applied for you, or you can find the **State Styles** property that corresponds with what is displaying those values and customize the settings.

 **Note:** The *Set Up States* dialog has a **Visual** button you can use as a shortcut to go to the related visualization properties.

For example, a table visualization has state styles that can be applied in each column: click on a column in the **Main** tab.

FirstName	SalesLastYear	SalesYTD
Grand Total	23,685,963.6244	36,277,591.9034
Amy	0.0000	519,905.9320
David	1,371,635.3158	1,573,012.9383
Garrett	1,620,276.8966	1,453,719.4653
Jae	1,635,823.3967	4,116,871.2277
Jillian	1,997,186.2037	3,189,418.3662
José	2,038,234.6549	2,604,540.7172
Linda	1,439,156.0291	4,251,368.5497
Lynn	2,278,548.9776	1,421,810.9242
Michael	1,750,406.4785	3,763,178.1787
Pamela	1,927,059.1780	1,352,577.1325
Rachel	1,307,949.7917	1,827,066.7118
Ranjit	2,396,539.7601	3,121,616.3202
Shu	2,073,505.9999	2,458,535.6169
Stephen	0.0000	559,697.5639
Syed	0.0000	172,524.4512

Search Properties

Main Look Layout Text

PROPERTIES

TABLE

Columns

SalesLastYear

SalesYTD

Row Header Columns

FirstName

Choose data to display

Column Headers

Interactivity

Sorting & Filtering

Freeze Total Rows

Show Totals At Bottom

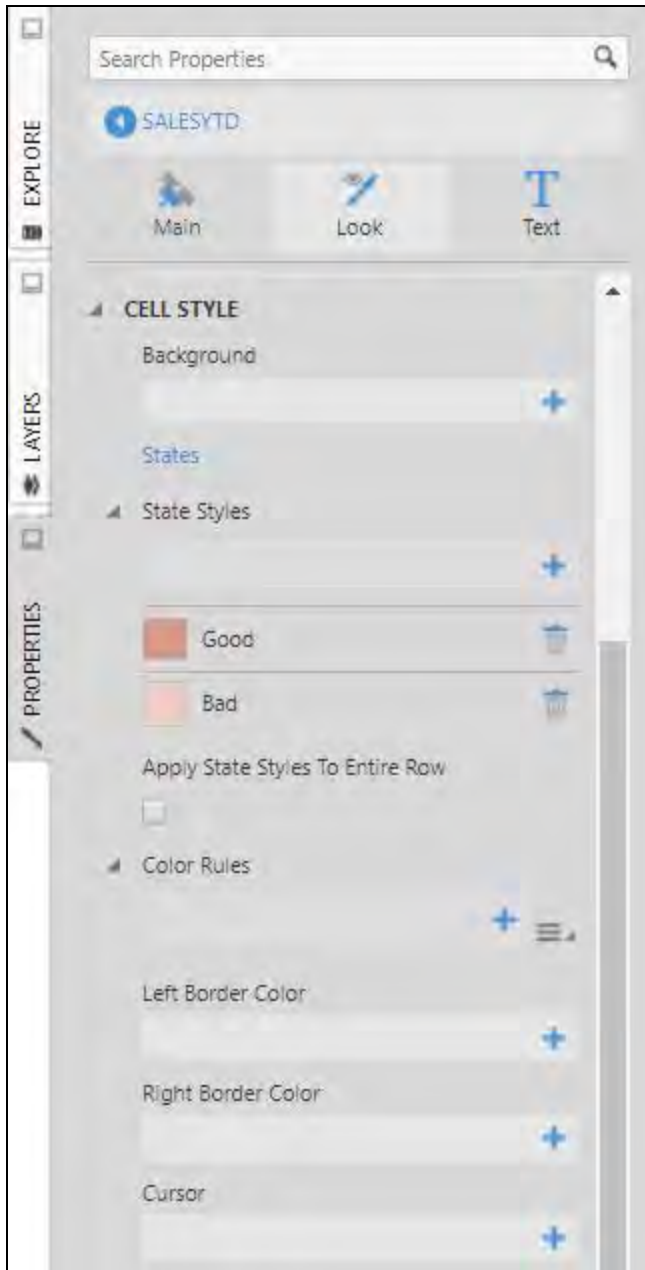
Show Row Numbers

OPTIONS



- Archive of documentation for Logi Symphony v24

In the **Look** tab, find **State Styles**. If your state group applies to this column's data, you will see state styles added automatically. You can also add your own and assign their **State** property to your state.



Select a **state style** to edit its properties. For example, change the `Text` for the **Good** state.

FirstName	SalesLastYear	SalesYTD
Grand Total	23,685,963.6244	Great!
Amy	0.0000	Great!
David	1,371,635.3158	Great!
Garrett	1,620,276.8966	1,453,719.4653
Jae	1,635,823.3967	Great!
Jillian	1,997,186.2037	Great!
José	2,038,234.6549	Great!
Linda	1,439,156.0291	Great!
Lynn	2,278,548.9776	1,421,810.9242
Michael	1,750,406.4785	Great!
Pamela	1,927,059.1780	1,352,577.1325
Rachel	1,307,949.7917	Great!
Ranjit	2,396,539.7601	Great!
Shu	2,073,505.9999	Great!
Stephen	0.0000	Great!
Syed	0.0000	Great!
Tete	0.0000	Great!
Tsvi	1,849,640.9418	Great!

EXPLORE

LAYERS

PROPERTIES

GOOD

COMMON

State

State Group 1: Good

Legend Text

Good

Legend Category

Text

Great!

APPEARANCE

Background

#D99B8D

Font Color

Font Family

Font Style

Font Size

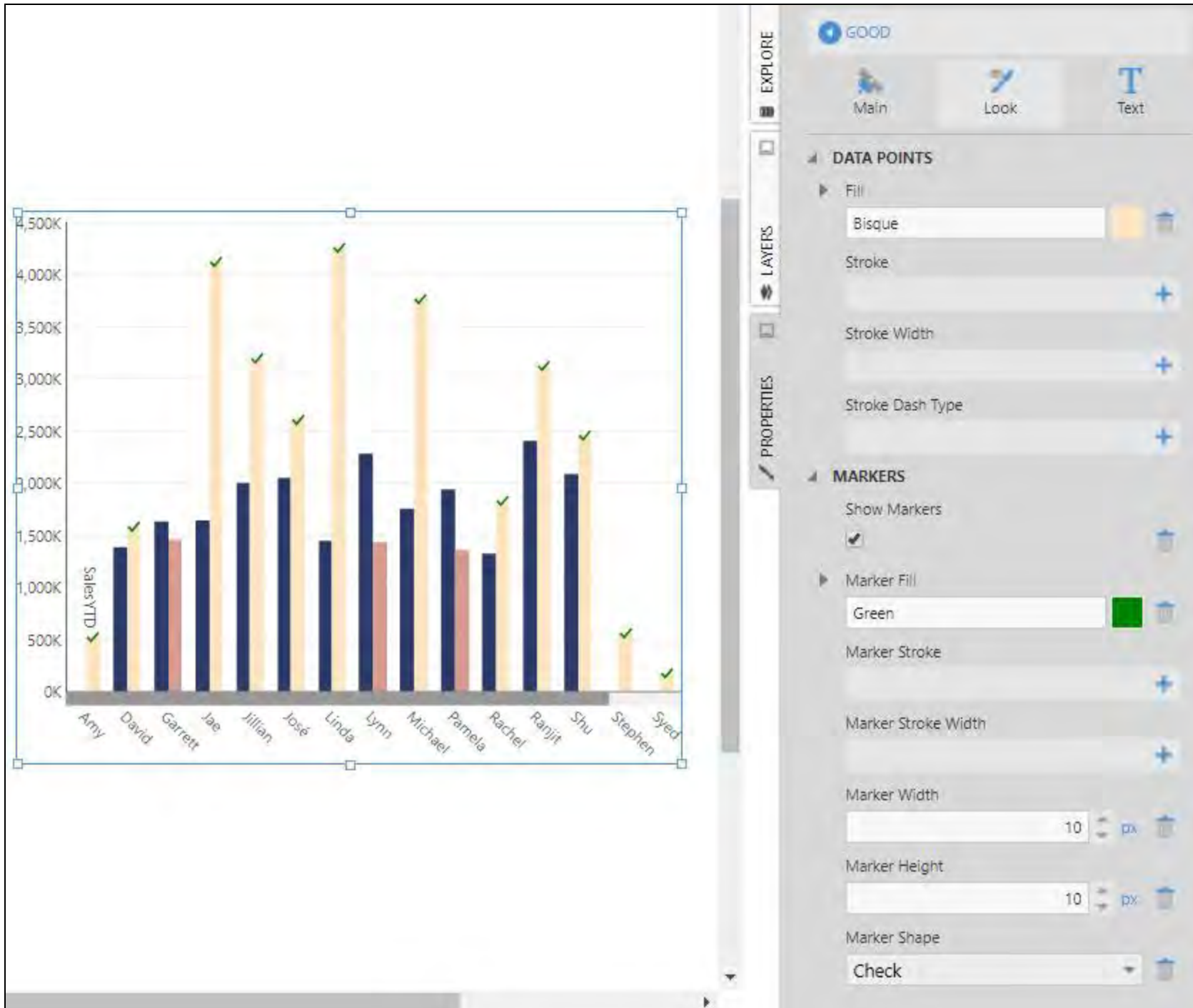
Font Variant

Font Weight



- Archive of documentation for Logi Symphony v24

When re-visualizing, notice that the state style options are visualization-specific and have changed. For example, here is our table re-visualized to a **Bar** chart. In the `State Styles` found in the data point series, the fill color has been changed and markers added for the **Good** state.

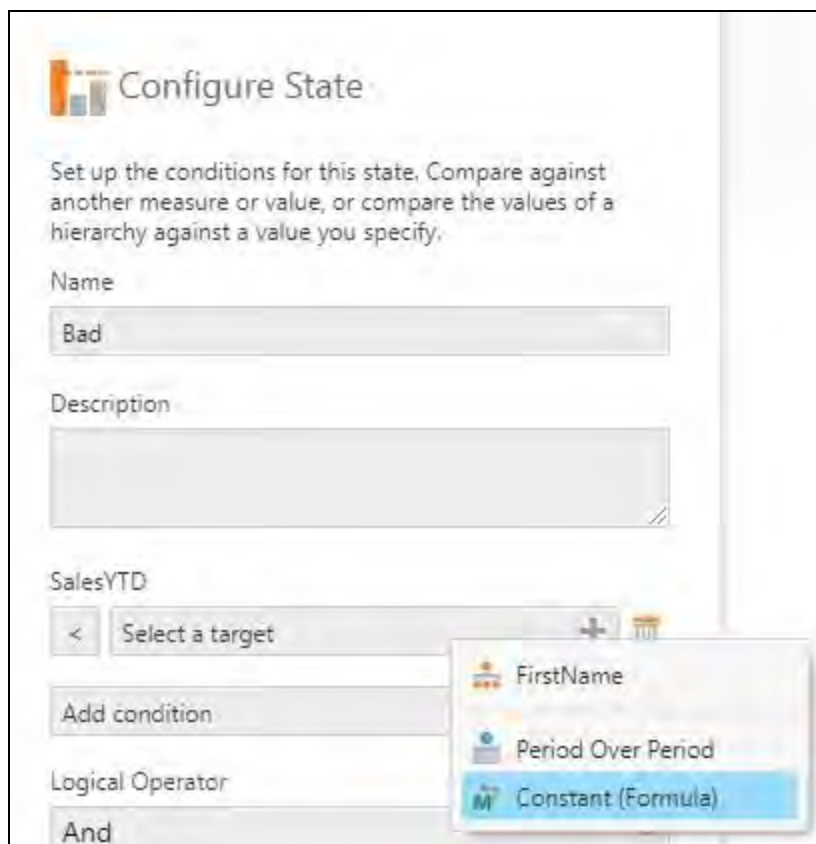


Examples

Comparing Against a Constant

In the **Configure State** dialog, you can add a condition that compares your measure against a constant numeric value.

Click **Select a target** and choose the **Constant (Formula)** option.



Type the constant value in the numeric text box and then click **Submit**.



Configure State

Set up the conditions for this state. Compare against another measure or value, or compare the values of a hierarchy against a value you specify.

Name

Description

SalesYTD

Add condition 

A new measure is added to the metric set with the caption **Constant (value)**.

FirstName	SalesYTD
Grand Total	36,277,591.9034
Amy	519,905.9320
David	1,573,012.9383
Garrett	1,453,719.4653
Jae	4,116,871.2277
Jillian	3,189,418.3662
José	2,604,540.7172
Linda	4,251,368.5497
Lynn	1,421,810.9242
Michael	3,763,178.1787
Pamela	1,352,577.1325
Rachel	1,827,066.7118
Ranjit	3,121,616.3202
Shu	2,458,535.6169
Stephen	559,697.5639
Syed	172,524.4512
Tete	1,576,562.1966
Tsvi	2,315,185.6110

Data Analysis Panel - Metric Set

Freeze Refresh Data Visualization Data Flow

Metric Set 39

MEASURES

SalesYTD SUM

Constant (2000000) AVG

click to add, or drop a measure

ROWS

FirstName

click to add, or drop a hierarchy

SLICERS

click to add, or drop a hierarchy

COLUMNS

click to add, or drop a hierarchy



The constant is created as a formula measure. If you need to change the constant value you can do so by clicking its **fx** button in the data analysis panel. You can also update its caption by editing the measure from the data analysis panel.

Comparing Against a Hierarchy

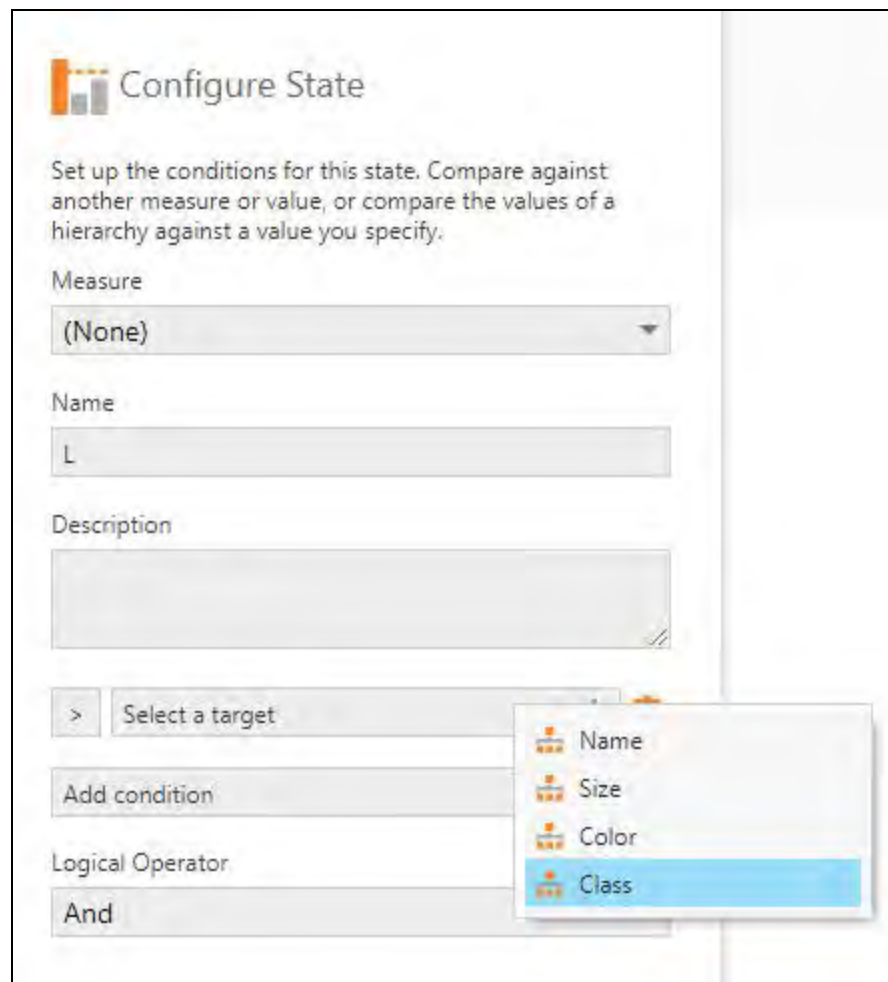
You can also add conditions in the **Configure State** dialog to check against text or date/time data, even if you associated your state group with a measure. For example, add two conditions that compare the **Date** value in each row of data to highlight all values between two dates. Or, you can set up a state to highlight **Sales** values of products that have a **Color** hierarchy value equal to **Silver**.

For the following example, create a table that displays various data made up of text values and dates (measures are not required for this example).

		Name	Size	Color	Class
Data Analysis Panel - Metric Set  Freeze  Refresh Data  Visualization Metric Set 1 MEASURES + click to add, or drop a measure ROWS Name Size Color Class click to add, or drop a hierarchy SLICERS click to add, or drop a hierarchy COLUMNS click to add, or drop a hierarchy		HL Mountain Frame - Black, 38	38	Black	H
		HL Mountain Frame - Black, 42	42	Black	H
		HL Mountain Frame - Black, 44	44	Black	H
		HL Mountain Frame - Black, 46	46	Black	H
		HL Mountain Frame - Black, 48	48	Black	H
		HL Mountain Frame - Silver, 38	38	Silver	H
		HL Mountain Frame - Silver, 42	42	Silver	H
		HL Mountain Frame - Silver, 44	44	Silver	H
		HL Mountain Frame - Silver, 46	46	Silver	H
		HL Mountain Frame - Silver, 48	48	Silver	H
		HL Road Frame - Black, 44	44	Black	H
		HL Road Frame - Black, 48	48	Black	H
		HL Road Frame - Black, 52	52	Black	H
		HL Road Frame - Black, 58	58	Black	H
		HL Road Frame - Black, 62	62	Black	H
		HL Road Frame - Red, 44	44	Red	H

Add a state. Since we are not checking measure values, leave the measure as **(None)**.

In the **Configure State** dialog, click **Add condition**, then click **Select a target** and choose the values you want to compare.



Configure State

Set up the conditions for this state. Compare against another measure or value, or compare the values of a hierarchy against a value you specify.

Measure
(None)

Name
L

Description

> Select a target

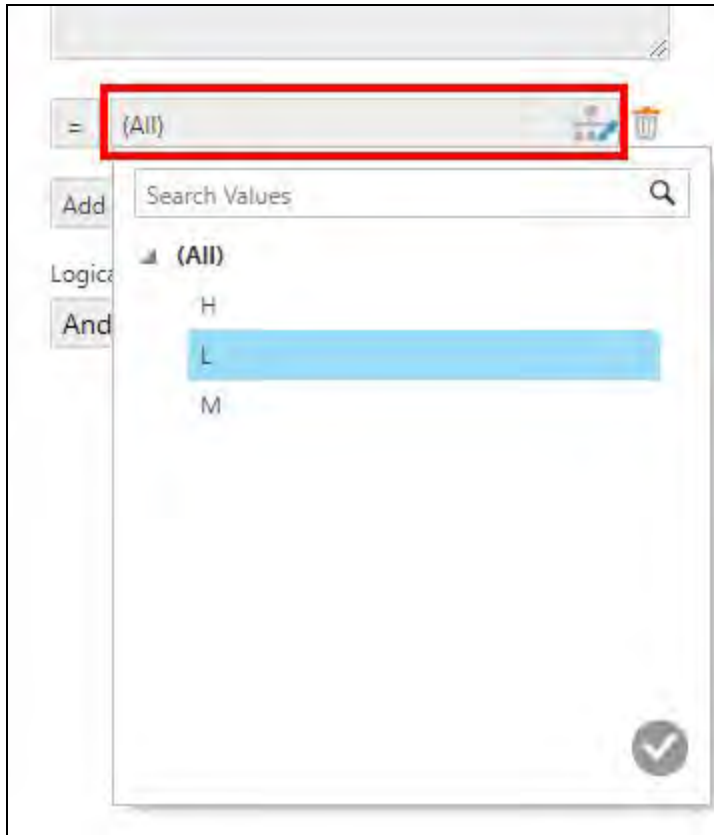
Add condition

Logical Operator
And

- Name
- Size
- Color
- Class

Next, the first button displays an operator such as an equals sign, which you can click to change. Date/time data can be compared using additional operators such as **Greater Than**.

Click the button next to it to choose which specific value to compare against.



You can add more conditions. In the example shown below, we're checking for rows with a particular **Class** value **And** a particular **Color** value according to the **Logical Operator** setting.



Configure State

Set up the conditions for this state. Compare against another measure or value, or compare the values of a hierarchy against a value you specify.

Measure

(None)

Name

L Black

Description

= L

= Black

Add condition

Logical Operator

And

The resulting metric set highlights values matching both conditions.

Data Analysis Panel - Metric Set

Freeze Refresh Data Visualization

Metric Set 1

MEASURES

click to add, or drop a measure

ROWS

Name Size Color Class

click to add, or drop a hierarchy

SLICERS

click to add, or drop a hierarchy

COLUMNS

click to add, or drop a hierarchy

Name	Size	Color	Class
HL Mountain Frame - Black, 38	38	Black	H
HL Mountain Frame - Silver, 38	38	Silver	H
ML Mountain Frame - Black, 38	38	Black	M
ML Mountain Frame-W - Silver, 38	38	Silver	M
ML Road Frame-W - Yellow, 38	38	Yellow	M
Mountain-100 Black, 38	38	Black	H
Mountain-100 Silver, 38	38	Silver	H
Mountain-200 Black, 38	38	Black	H
Mountain-200 Silver, 38	38	Silver	H
Mountain-300 Black, 38	38	Black	M
Mountain-400-W Silver, 38	38	Silver	M
Road-550-W Yellow, 38	38	Yellow	M
LL Mountain Frame - Black, 40	40	Black	L
LL Mountain Frame - Silver, 40	40	Silver	L
ML Mountain Frame - Black, 40	40	Black	M
ML Mountain Frame-W - Silver, 40	40	Silver	M
ML Road Frame-W - Yellow, 40	40	Yellow	M



- Archive of documentation for Logi Symphony v24

For more information, see:

- [Using a State Indicator](#)
- [Symphony Metric Sets](#)
- [Other Metric Set Tools](#)
- [Create a Metric Set and Add a Filter](#)



Slicers Versus Columns and Rows

This applies to: *Managed Dashboards, Managed Reports*

This article describes the purpose of slicers, columns, rows, and measures in the *Data Analysis Panel*.

The Data Analysis Panel appears when you drag and drop data onto a metric set, or the canvas of a dashboard or other view. If closed, you can re-open it from the toolbar or context menu for a data visualization.

Data Analysis Panel

The Data Analysis Panel lets you build a view of your data simply by dragging data from the Explore window.

Type	Purpose	Visible
Measures	Display numeric/fact values	Yes
Rows	Group by values into rows/data points	Yes
Slicers	Filtering only	Limited
Columns	Group by values into columns/series	Yes

The figure below shows the Data Analysis Panel for a table visualization.

Data Analysis Panel: Metric Set

Freeze
Refresh Data
Visualization

Metric Set 1

Measures

Total

Target

click to add, or drop a measure

Rows

OrderDate Month of year

click to add, or drop a hierarchy

Slicers

Online Orders

click to add, or drop a hierarchy

Columns

Product

click to add, or drop a hierarchy

	Accessories		Bikes		Clothing
Month of year	Total	Target	Total	Target	Total
January	\$88,440.55	\$87,556.15	\$9,581,530.97	\$9,485,715.66	\$128,721.15
February	\$66,067.74	\$65,407.06	\$4,622,222.89	\$4,576,000.66	\$82,800.00
March	\$151,774.46	\$150,256.71	\$11,711,823.06	\$11,594,704.83	\$268,480.00
April	\$83,971.83	\$83,132.11	\$5,264,823.65	\$5,212,175.41	\$100,990.00
May	\$149,042.33	\$147,551.91	\$10,017,469.32	\$9,917,294.63	\$278,070.00
June	\$120,958.39	\$119,748.80	\$7,583,083.85	\$7,507,253.02	\$255,570.00
July	\$124,917.75	\$123,668.57	\$8,288,750.92	\$8,205,863.41	\$237,640.00
August	\$98,681.65	\$97,694.83	\$7,000,279.33	\$6,930,276.53	\$157,890.00
September	\$104,665.15	\$103,618.50	\$7,175,476.70	\$7,103,721.93	\$188,480.00
October	\$116,291.87	\$115,128.95	\$10,404,879.89	\$10,300,831.09	\$192,370.00
November	\$83,975.79	\$83,136.03	\$5,475,466.21	\$5,420,711.55	\$103,970.00
December	\$83,285.37	\$82,452.52	\$7,525,365.91	\$7,450,112.25	\$125,510.00
Grand Total	\$1,272,072.88	\$1,259,352.16	\$94,651,172.70	\$93,704,660.98	\$2,120,540.00



Note: If you're not sure where in the Data Analysis Panel to drag data, just drop it onto the visualization instead.

Measures

Add data under **Measures** in the Data Analysis Panel to display the numeric values you are interested in viewing (such as sales or stock levels). Measure values will be grouped into the Rows and Columns data that have been added, and filtered by Slicers, if any.

In a table visualization, each measure appears as a column. When selecting measures together with data under **Columns**, the measure columns will reoccur for each column hierarchy value (as seen under *Accessories* and *Bikes* below).

Data Analysis Panel: Metric Set					
Freeze	Refresh Data	Visualization			
Metric Set 1					
Measures					
Total	SUM				
Target	SUM				
click to add, or drop a measure					

Accessories		Bikes		Clothing	
Month of year	Total	Target	Total	Target	Total
January	\$88,440.55	\$87,556.15	\$9,581,530.97	\$9,485,715.66	\$128,721.15
February	\$66,067.74	\$65,407.06	\$4,622,222.89	\$4,576,000.66	\$82,800.00
March	\$151,774.46	\$150,256.71	\$11,711,823.06	\$11,594,704.83	\$268,480.00
April	\$83,971.83	\$83,132.11	\$5,264,823.65	\$5,212,175.41	\$100,990.00
May	\$149,042.33	\$147,551.91	\$10,017,469.32	\$9,917,294.63	\$278,070.00
June	\$120,958.39	\$119,748.80	\$7,583,083.85	\$7,507,253.02	\$255,570.00
July	\$124,917.75	\$123,668.57	\$8,288,750.92	\$8,205,863.41	\$237,640.00
August	\$98,681.65	\$97,694.83	\$7,000,279.33	\$6,930,276.53	\$157,890.00
September	\$104,665.15	\$103,618.50	\$7,175,476.70	\$7,103,721.93	\$188,480.00
October	\$116,291.87	\$115,128.95	\$10,404,879.89	\$10,300,831.09	\$192,370.00

In a chart, each measure is displayed as a series of data points, each with a different color by default.

Data Analysis Panel: Metric Set

 Freeze
  Refresh Data
  Visualization

Metric Set 1

Measures

Total  SUM  

 Target  SUM  

[click to add](#), or drop a measure

Rows (Data Points)

OrderDate Month of year   

[click to add](#), or drop a hierarchy

Slicers

Online Orders   

[click to add](#), or drop a hierarchy

Columns (Series)

[click to add](#), or drop a hierarchy



 **Note:** To display a row or data point for each measure instead of a column or series, you can [transpose](#) the metric set.

Rows

Data can be added under **Rows** in the Data Analysis Panel to group the measures by these values into rows, such as sales broken down by year instead of just a single total sales value.

This data often consists of text or date/time values, and is called a hierarchy (or dimension).

Data Analysis Panel: Metric Set

Freeze
Refresh Data
Visualization

Metric Set 1

Measures

Total

Target

[click to add](#), or drop a measure

Rows

Order Date

[click to add](#), or drop a hierarchy

Year	Total	Target
▶ 2020	\$12,641,672.21	\$12,515,255.49
▶ 2021	\$33,524,301.32	\$33,189,058.31
▶ 2022	\$43,622,479.05	\$43,186,254.26
▶ 2023	\$20,057,928.81	\$19,857,349.52
Grand Total	\$109,846,381.40	\$108,747,917.59

If you add multiple hierarchies, a row will be displayed for each combination of their values in the data, for example for each year and for each country.

In a bar chart, rows of data are displayed as data points (bars). These will often be displayed along a chart axis with a label corresponding with each data point.



Slicers

Data can be added under **Slicers** in the Data Analysis Panel when you only want to filter by those values. In our example shown below, all the data displayed in the table is filtered to only online orders by filtering a column of true/false values named *Online Orders* to the value *True*.

The slicer values selected for filtering can be displayed in tooltips and labels (a limited number of filter values can be displayed at a time if many are selected at once). Slicer values apply to the entire visualization: if you want to display different values in each row, column, or data point, you can add this data elsewhere on the metric set instead of Slicers.

After adding data under Slicers, click its filter icon and choose the values to filter by.

Data Analysis Panel: Metric Set



Freeze



Refresh Data



Visualization

Metric Set 1



Measures



Total



SUM



Target



SUM



[click to add](#), or drop a measure



Rows



Order Date



[click to add](#), or drop a hierarchy



Slicers



Online Orders



[click to add](#), or drop a hierarchy

▶ 2021	\$33,524,301.32	\$33,189,058.31
▶ 2022	\$43,622,479.05	\$43,186,254.26
▶ 2023	\$20,057,928.81	\$19,857,349.52
Grand Total	\$109,846,381.40	\$108,747,917.59

Sort



Ascending



Descending



Unspecified

Filter



Note: You can also filter data added as measures, rows, or columns, when you want to visualize and filter by those values at the same time.

In a dashboard or another view, you can also [add a filter](#) and connect it to this slicer.

Columns

Data can be added under **Columns** in the Data Analysis Panel to group the measures by the selected values into columns. Since measures are each displayed as a column, a column hierarchy will repeat the measures for each of its values.

Column hierarchy values appear as another row of column headers in a table visualization.

Data Analysis Panel: Metric Set

||

Freeze

↺

Refresh Data

📊

Visualization

Metric Set 1

📏

Measures

Total

✎ SUM

📏

🗑

Target

✎ SUM

📏

🗑

click to add, or drop a measure

📅

Rows

Order Date Month of year

✎

📏

🗑

click to add, or drop a hierarchy

✎

Slicers

Online Orders

✎

📏

🗑

click to add, or drop a hierarchy

📑

Columns

Product

✎

📏

🗑

click to add, or drop a hierarchy



- Archive of documentation for Logi Symphony v24

In a bar chart, each series of bars is split into separate series for each column hierarchy value, where each series has a different color by default.

Data Analysis Panel: Metric Set

 Freeze
  Refresh Data
  Visualization

Metric Set 1

Measures

 Total
  SUM
 


[click to add](#), or drop a measure

Rows (Data Points)

 Order Date Month of year
 



[click to add](#), or drop a hierarchy

Slicers

 Online Orders
 

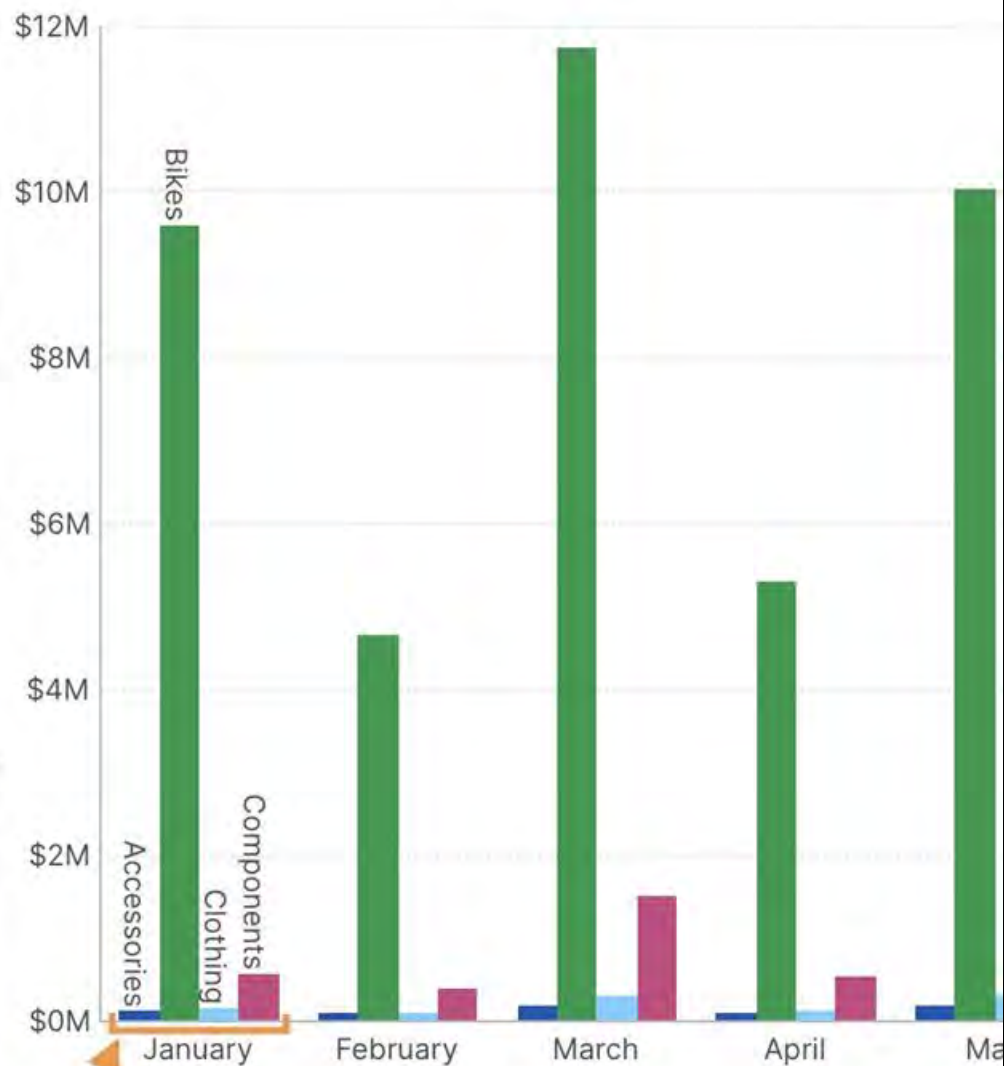


[click to add](#), or drop a hierarchy

Columns (Series)

 Product
 







- Archive of documentation for Logi Symphony v24

For more information, see:

- Video: [Using The Data Analysis Panel](#)
- Video: [Filters](#)
- [Symphony Metric Sets](#)
- [Other Metric Set Tools](#)
- [Create a Metric Set and Add a Filter](#)
- [The Managed Dashboards Data Model](#)
- [Setting Up the Visualization](#)

Use Metric Set Query Hints

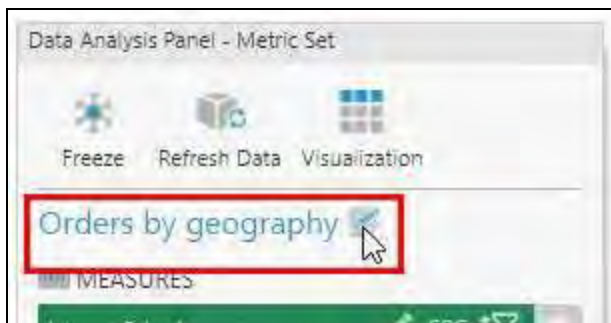
This applies to: *Managed Dashboards, Managed Reports*

The *Advanced (Query Hints)* section of the metric set's settings dialog provides advanced settings that are relevant to the current metric set depending on the data connector it uses.

Using Query Hints

To find and use query hints in your metric set, first ensure that the metric set is [checked out](#) to you.

Find or open the **Data Analysis Panel**, and click the metric set name or **Edit** icon to edit its settings.



In the dialog that appears, expand the **Advanced (Query Hints)** section at the bottom.

Metric Set Configuration

This is where you can change how we retrieve data for this metric set.

Metric Set Full Name
[My Project] Metric Sets Root Folder/Orders by geography

Metric Set ID
1d0164f2-7020-4c59-b16c-77b56c164283

Analysis Structure Name
[<GLOBAL>] Data Connector Root Folder/SSAS/Advent

▼ Metric Set Settings

Transpose Result
☐

Prevent Data Export
☐

Shown Totals
Columns

Data Retrieval Format
Standard

Top / Bottom Measure
(None)

Other Group Measure
(None)

Request Options

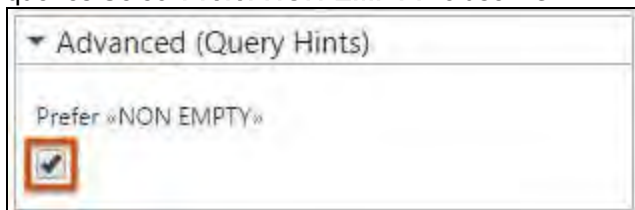
Inherited Parameters

Advanced (Query Hints)

Depending on what data connector provides the data for this metric set, options may appear as described in the following sections. Query hints may not be available when combining data from multiple data connectors.

Prefer NON-EMPTY

When the data comes from an OLAP database, you can set a preference for using *NON EMPTY* instead of the default *Nonempty()* when generating MDX queries. Select **Prefer NON-EMPTY** to use *NON EMPTY* instead of *Nonempty()* when generating MDX queries.



Read Uncommitted

When data comes from Microsoft SQL Server or IBM Db2, you can use a query hint to specify that statements can read rows that have been modified by other transactions but not yet committed. This option enables dirty reads.

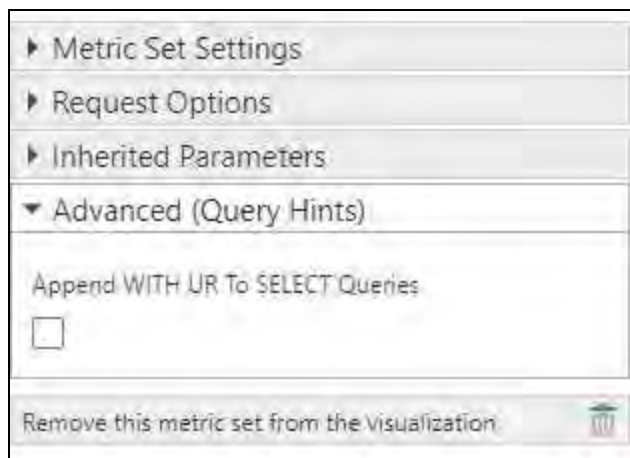


Important: Ensure that you are aware of the features and limitations of the *uncommitted read* isolation level before attempting to use it.

For SQL Server, select the **Read Uncommitted** query hint to enable this option.



For IBM Db2, select **Append WITH UR To SELECT Queries**.



For more information, see:

- [Symphony Metric Sets](#)
- [Connect to OLAP Data and Apply a Formula](#)
- Microsoft Docs: [SET TRANSACTION ISOLATION LEVEL](#)
- IBM Documentation: [Isolation levels](#)



Adding Formulas

This applies to: *Managed Dashboards, Managed Reports*

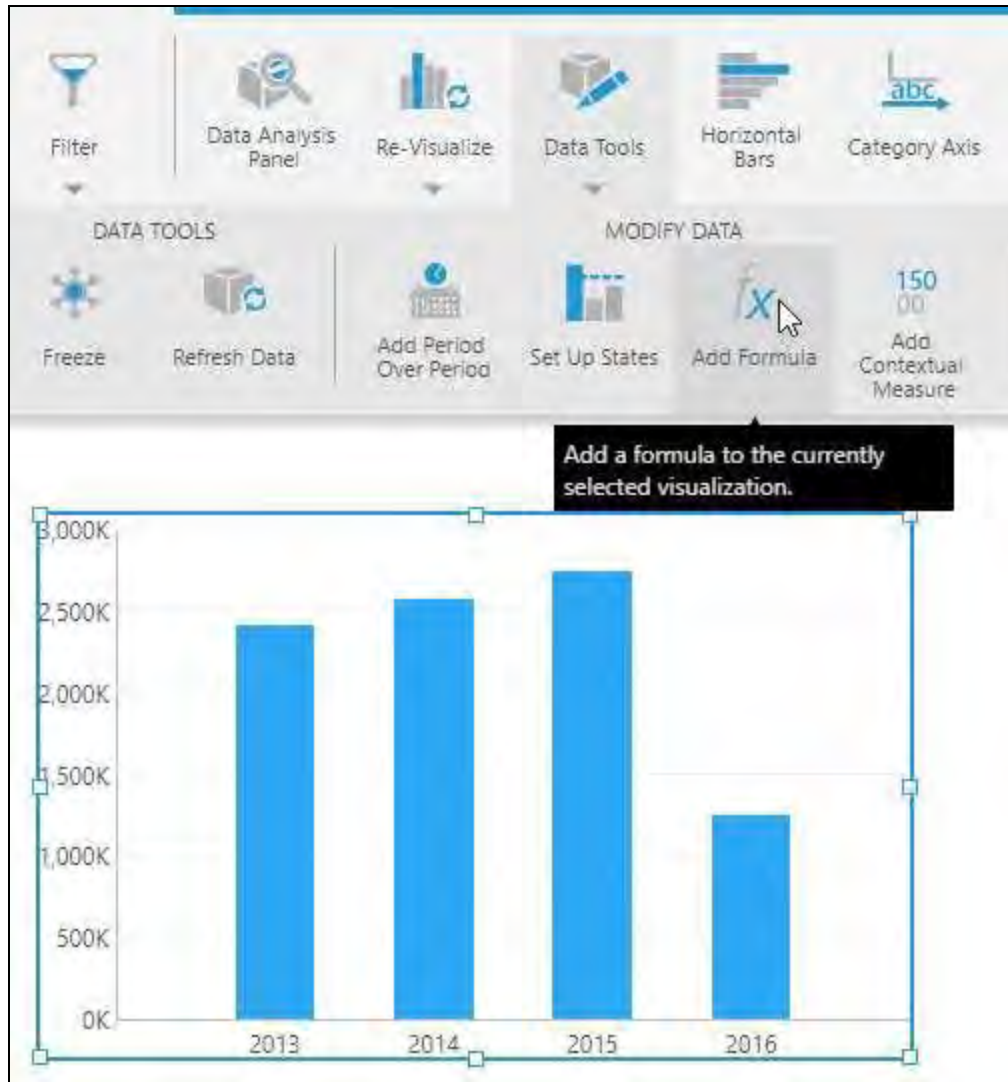
You can apply formulas to data by writing small expressions, much like you would in a spreadsheet program. A formula expression can be a simple constant value, arithmetic or other math functions, a call to a built-in [formula function](#) such as AVG or TOP, or any combination of these. For more complex formulas, you can write [DundasScript](#).

Although this article demonstrates the various ways of using formulas while editing a dashboard or another view, you can add formula measures when editing a metric set full-screen as well.

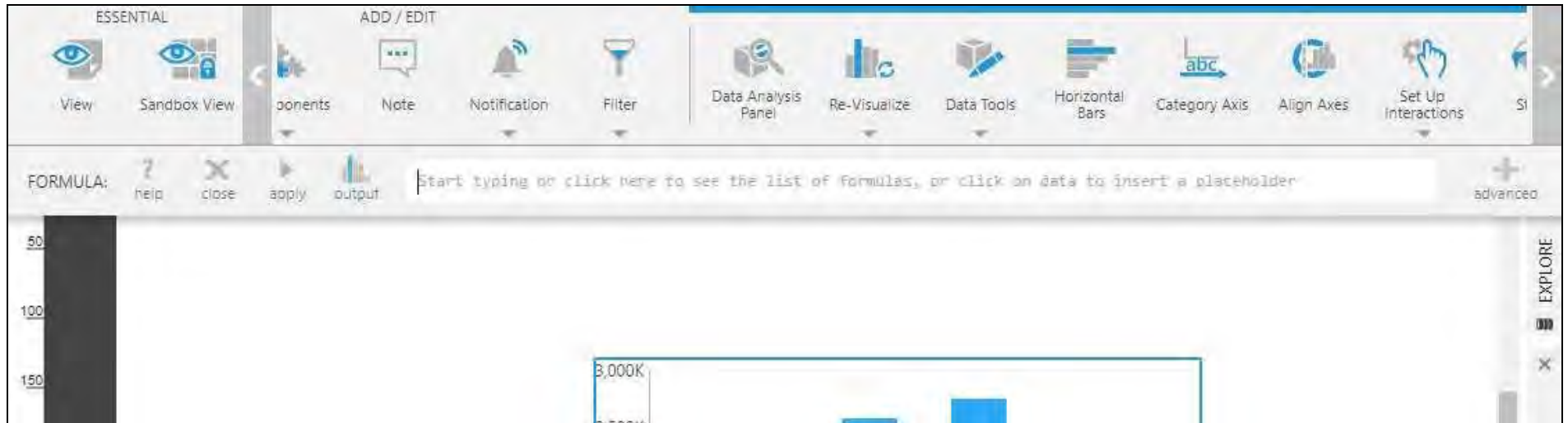
Related video: [Formulas](#)

Accessing the formula bar

Select a data visualization on the canvas, click **Data Tools** in the toolbar and then **Add Formula**. When editing a metric set full-screen, simply choose **Add Formula** in the toolbar.



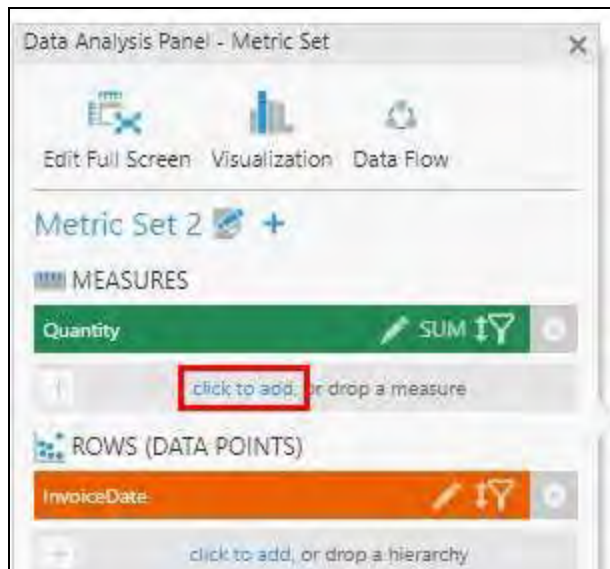
A formula bar appears below the main toolbar.



There is an **output** option for charts, for example, to choose what chart type to use to visualize the new formula measure. There is also an **advanced** mode for entering complex, multi-line formula expressions.

Once you are finished entering your formula, click **apply** or type Enter.

An alternative way of opening the formula bar is from the Data Analysis Panel: **click to add** a measure under *Measures*.



Then click **<Formula>** at the bottom under *Calculated Elements*.



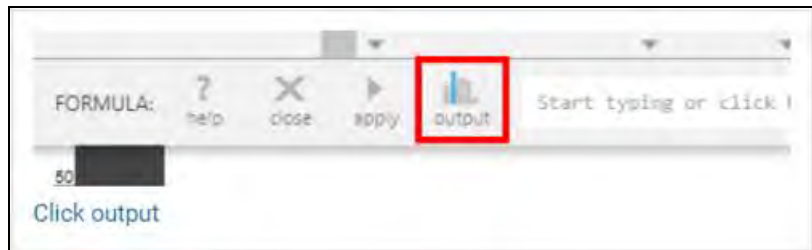
Simple Formulas

By default, the formula bar starts off in simple editing mode which allows you to enter a one line expression.

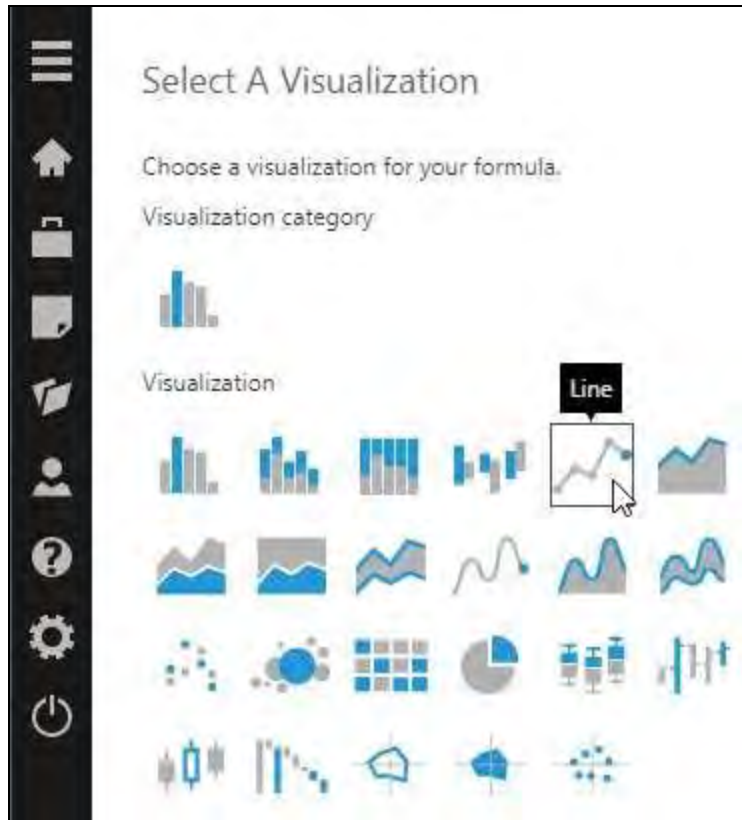
Constant Value

The simplest formula expression you can have is a constant number. As an example, consider a bar chart showing order quantity as a measure by invoice date.

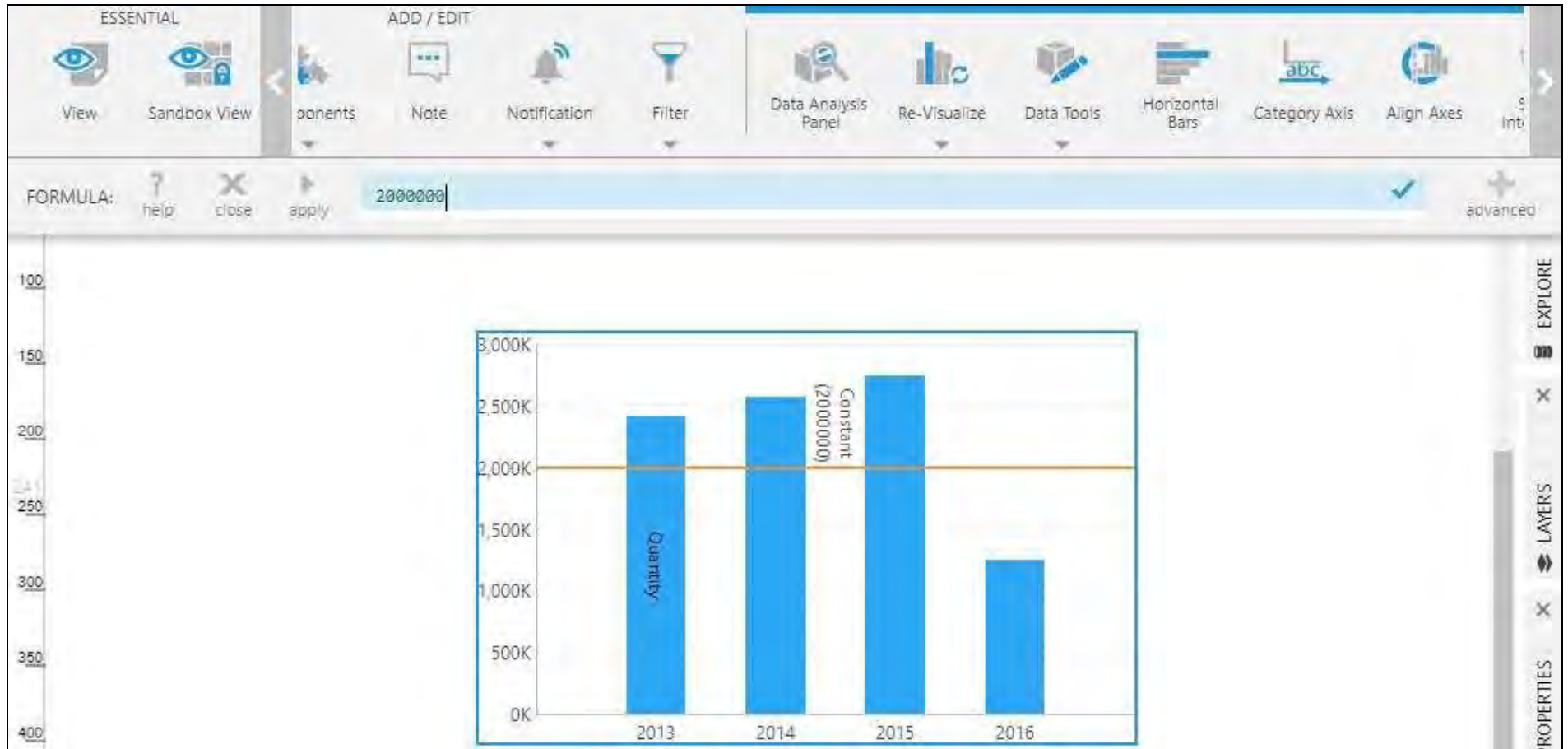
Open the formula bar for this chart and click the **output** button to change it from a bar chart series to another type.



Select the **Line** chart option and then click **Submit**.



Notice that the icon for the output button changes to indicate a Line chart visualization. Type a number into the formula bar and click **apply** or type Enter. The text box is highlighted in blue and a checkmark appears on the right to indicate the formula syntax is correct, and the formula result is added as a line series.

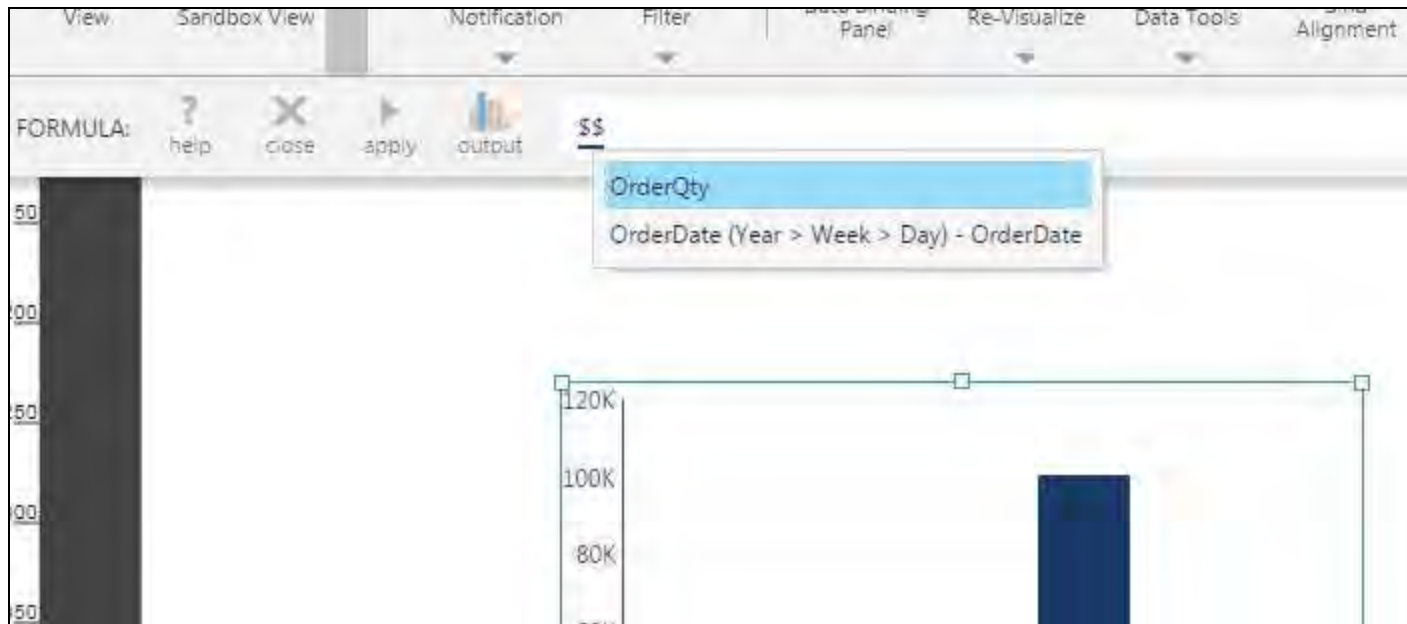


Open the *Data Analysis Panel* for the chart to see the formula result has been added as a second measure to the underlying metric set, appearing with an *fx* icon.



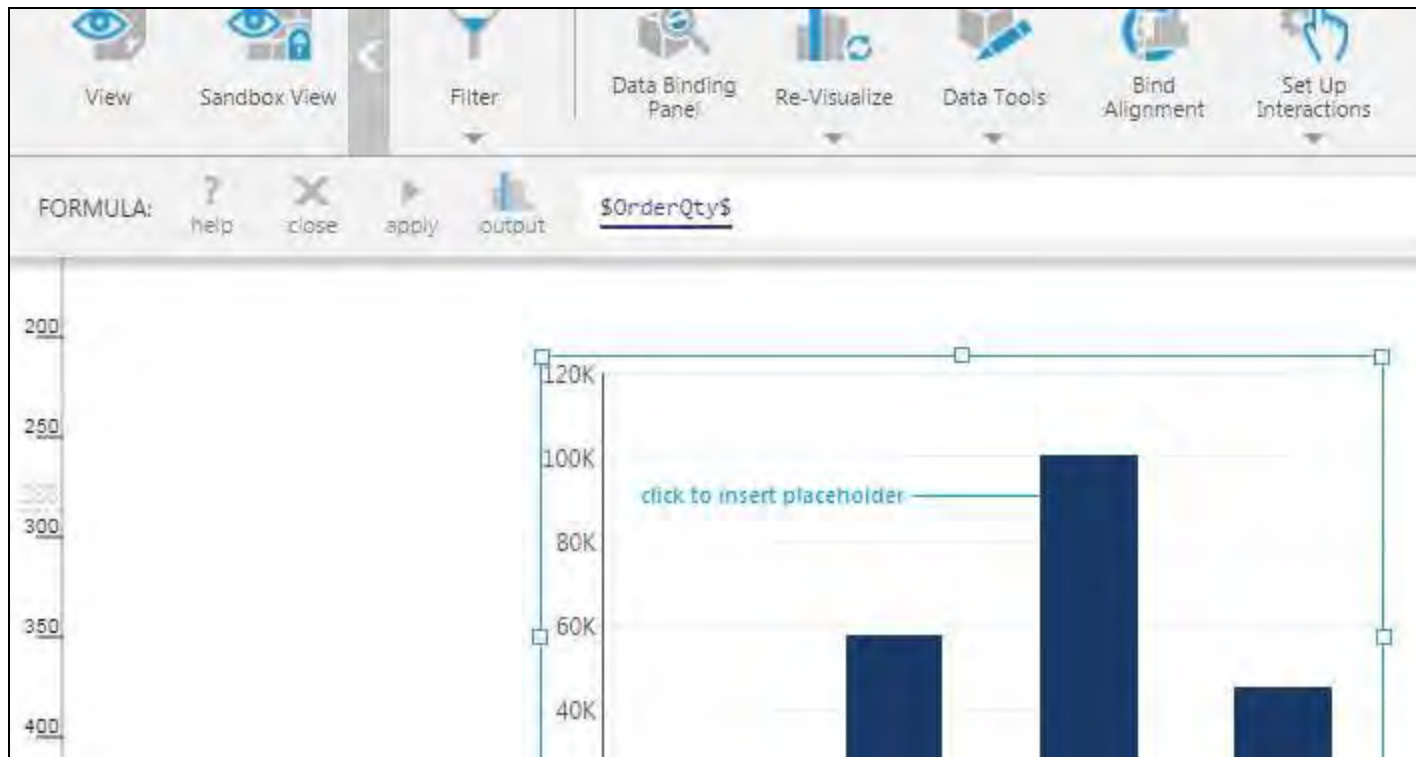
Placeholders

A formula expression can include placeholders representing the other data in your metric set. To insert a placeholder, type a dollar sign (\$) character in the formula bar. You will see a menu that lists available placeholders you can insert.

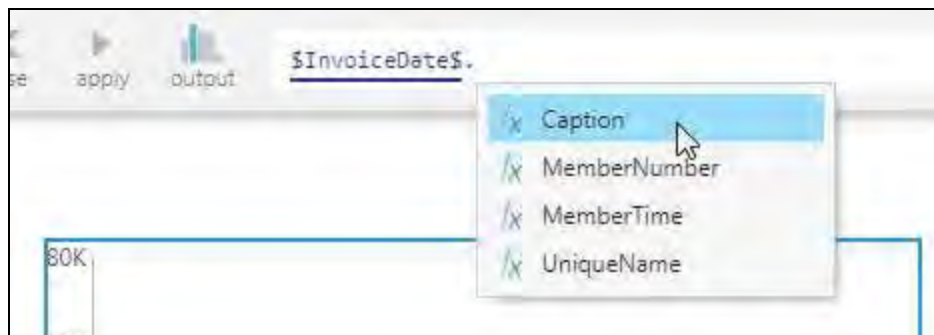


The inserted placeholder consists of the measure or hierarchy name enclosed between dollar sign characters (for example, *\$OrderQty\$*).

Another way to insert a placeholder is to click the relevant part of a data visualization. For example, click a column in a table, click a data point in a chart to insert a placeholder for the corresponding measure, or click an axis label.



If you are working directly with hierarchy values in a formula, you can type a period (.) and one of the properties *MemberTime*, *MemberNumber*, *UniqueName*, or *Caption*, or choose it from the popup. (If no property is specified, *Caption* is used by default.)

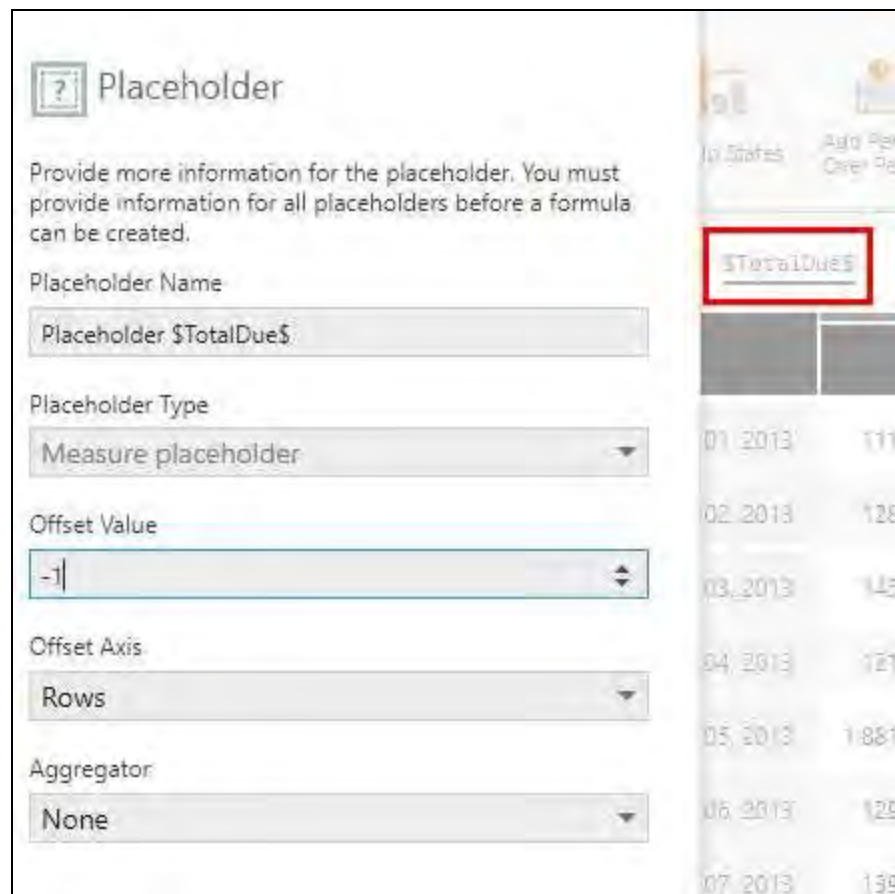


Formulas are measures so they should return numeric values. Some examples of referring to hierarchy values are *\$Product/D\$.MemberNumber* if the hierarchy contains numeric values, or for text values:


```
if ($HierarchyMember$.Caption == "Test") { return 1; }
```

Placeholder Settings

For some additional options, after adding a measure placeholder to your formula, click it to open the *Placeholder* dialog.



The image shows the 'Placeholder' dialog box on the left and a data table on the right. The dialog box has the following fields:

- Placeholder Name:** Placeholder \$TotalDue\$
- Placeholder Type:** Measure placeholder
- Offset Value:** -1
- Offset Axis:** Rows
- Aggregator:** None

The data table on the right has columns 'In States' and 'Avg Per Capita'. The 'Avg Per Capita' column contains values for months 01 through 07 of 2013. The value for 01_2013 is 111, and for 07_2013 is 139. The cell containing '01_2013' is highlighted with a red box.

In States	Avg Per Capita
	111
	128
	145
	121
	1881
	129
	139

When first adding the placeholder, click **apply** before clicking on the placeholder.

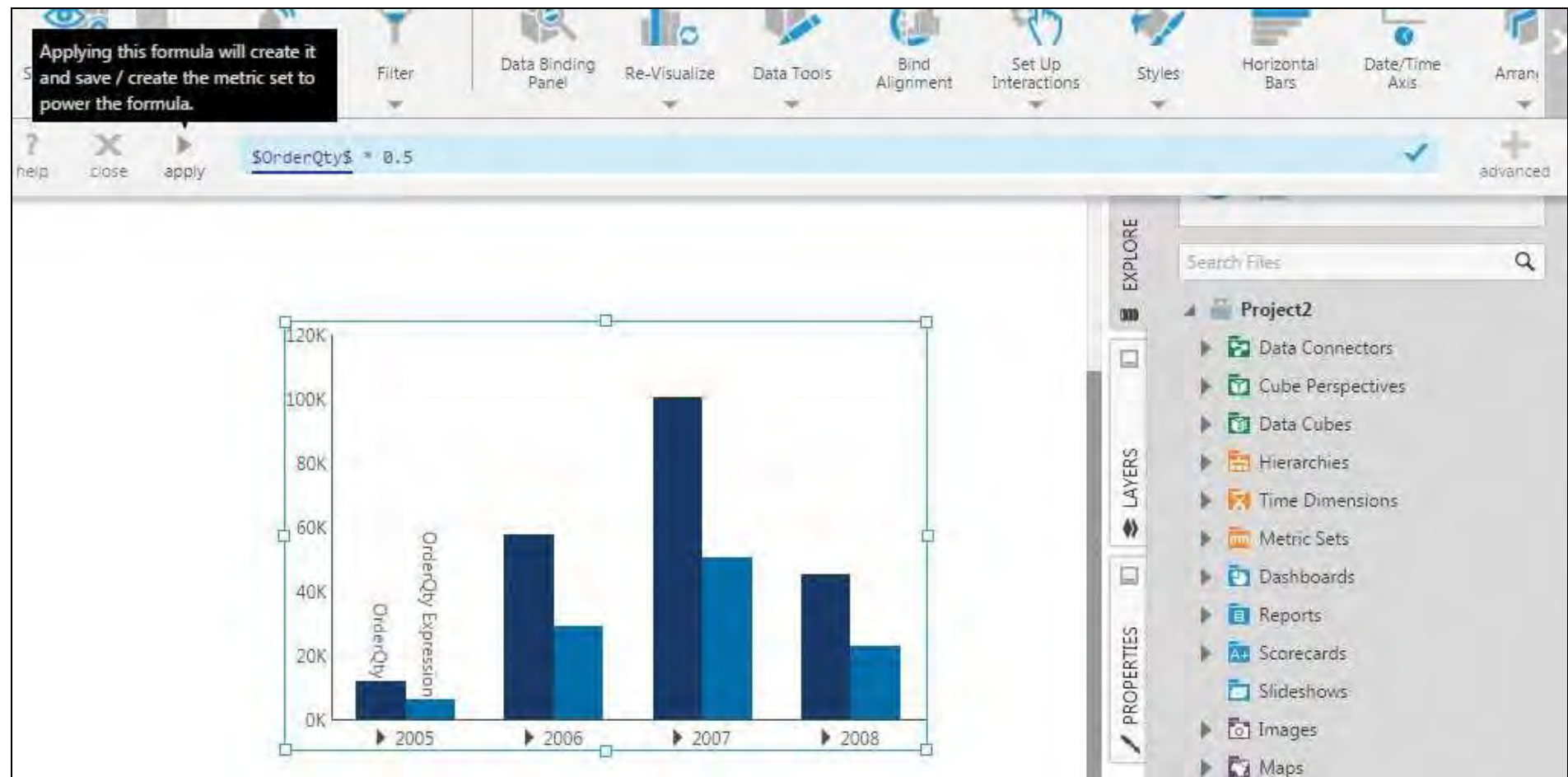
To refer to values from other rows or columns, set **Offset Value** and **Offset Axis**. For example, an offset of -7 Rows can refer to the previous row's value. The **Aggregator** can optionally be set to get the overall aggregated sum, average, etc., just for the values inserted by this placeholder into the formula.

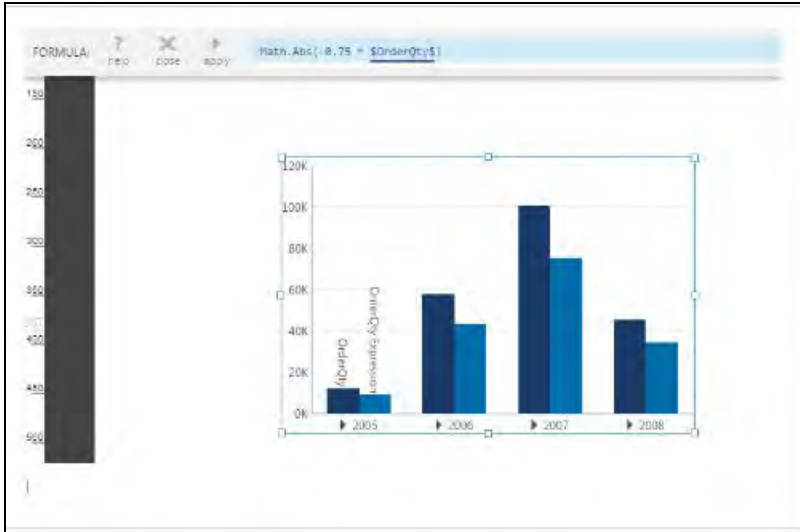
Math

You can use arithmetic operators within a formula expression as well as [math functions](#) such as `Math.Max()` and `Math.Abs()`. These arithmetic operations and math functions are calculated separately in each row.

Examples:

- `$OrderQty$ * 0.5`
- `Math.Abs(-0.75 * $OrderQty$)`

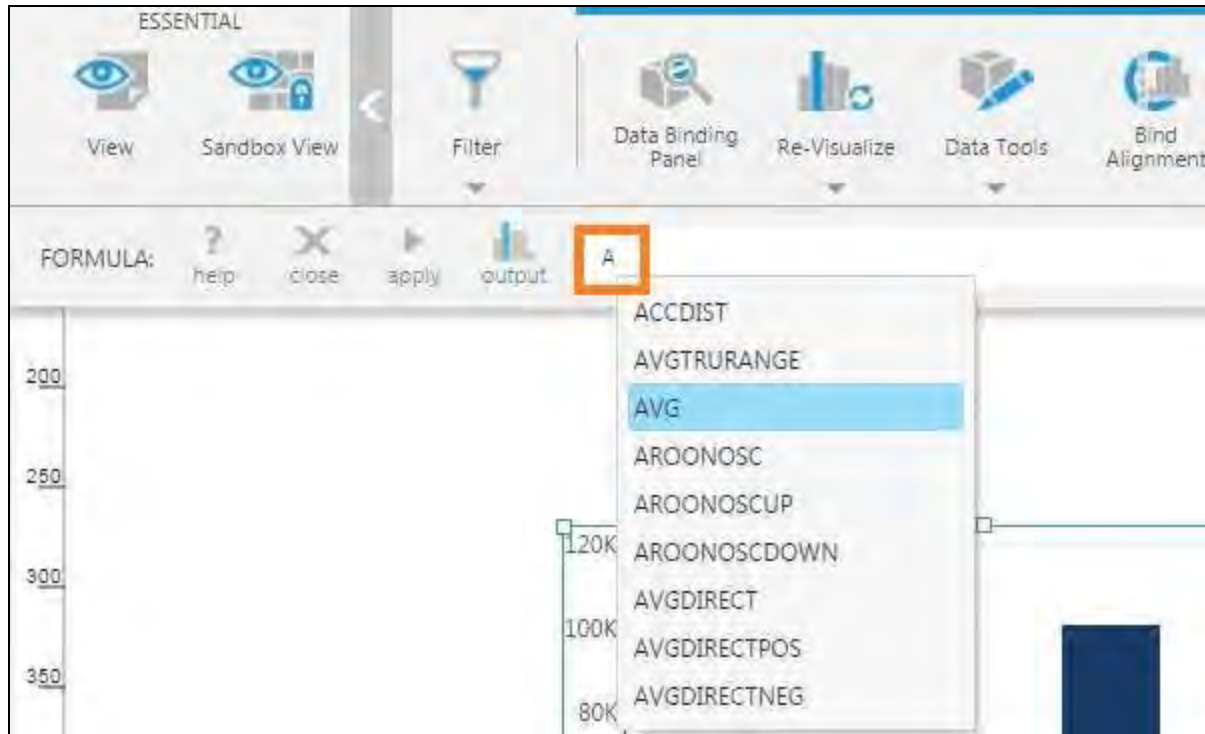




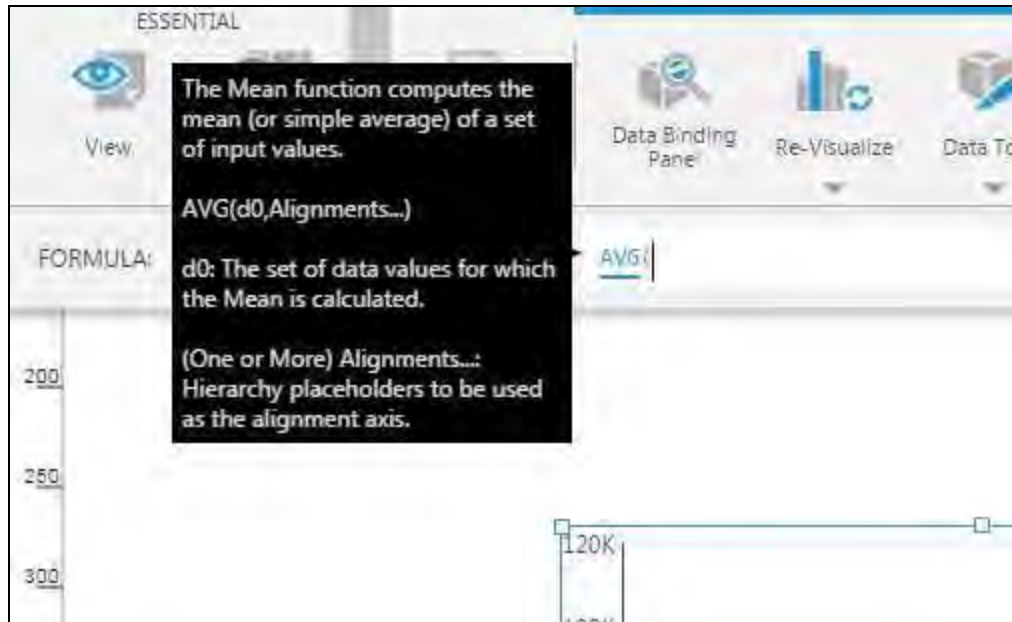
Functions

There are over 50 [built-in formula functions](#) that you can apply to your data. These functions work with the entire series of values you specify when calculating their result.

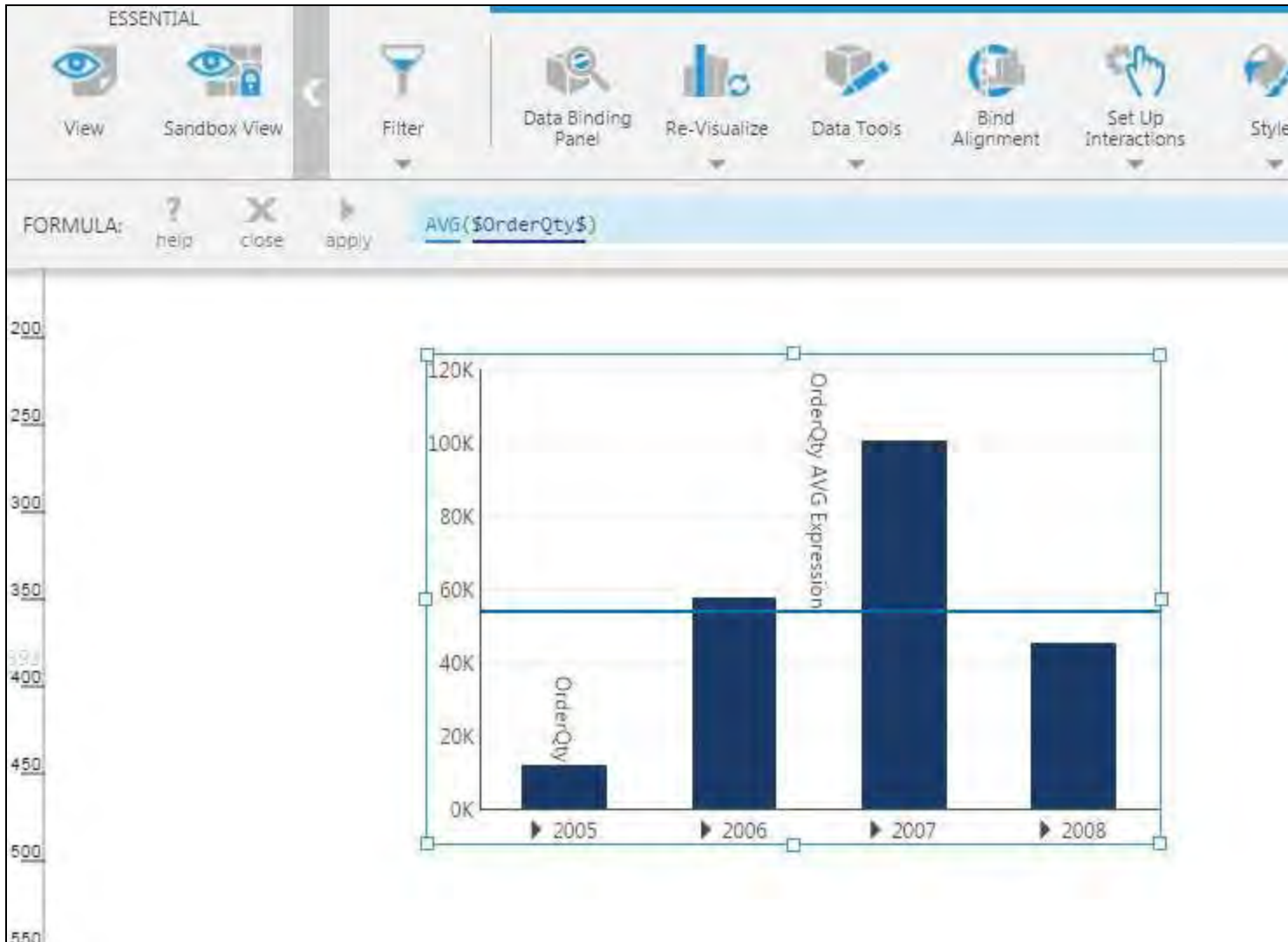
Type a letter in the formula bar and you'll see an auto-complete menu showing a list of matching functions you can choose from.



Once you've chosen a function (or typed it in manually along with the open bracket), a tooltip appears which shows the required and optional arguments for the function.



Most functions take in numeric values (measures) as inputs. The first argument for the AVG function is a measure placeholder which you can insert by typing a dollar sign character or clicking on a data point.



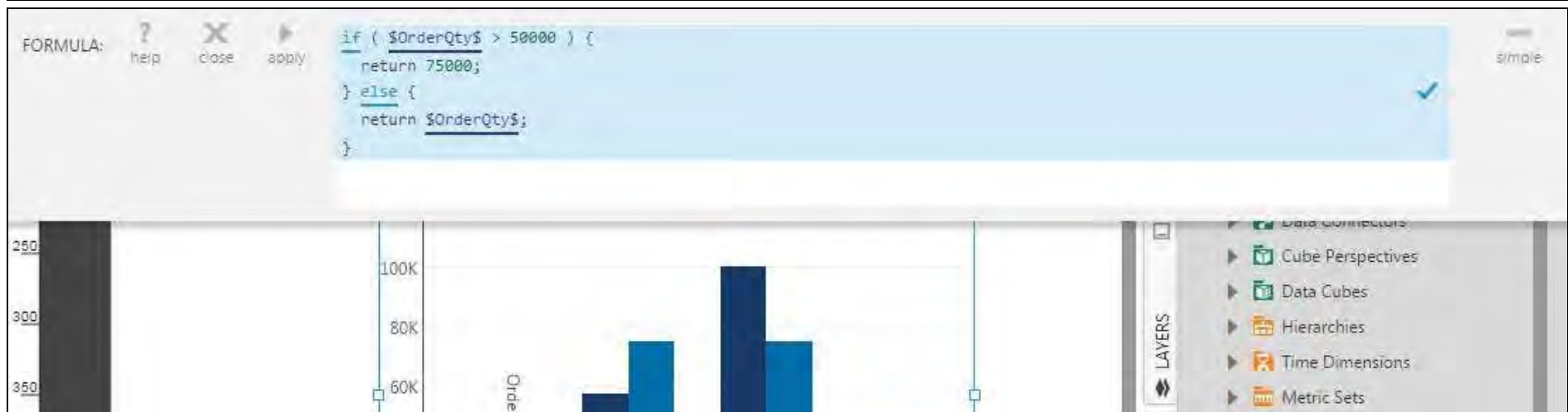
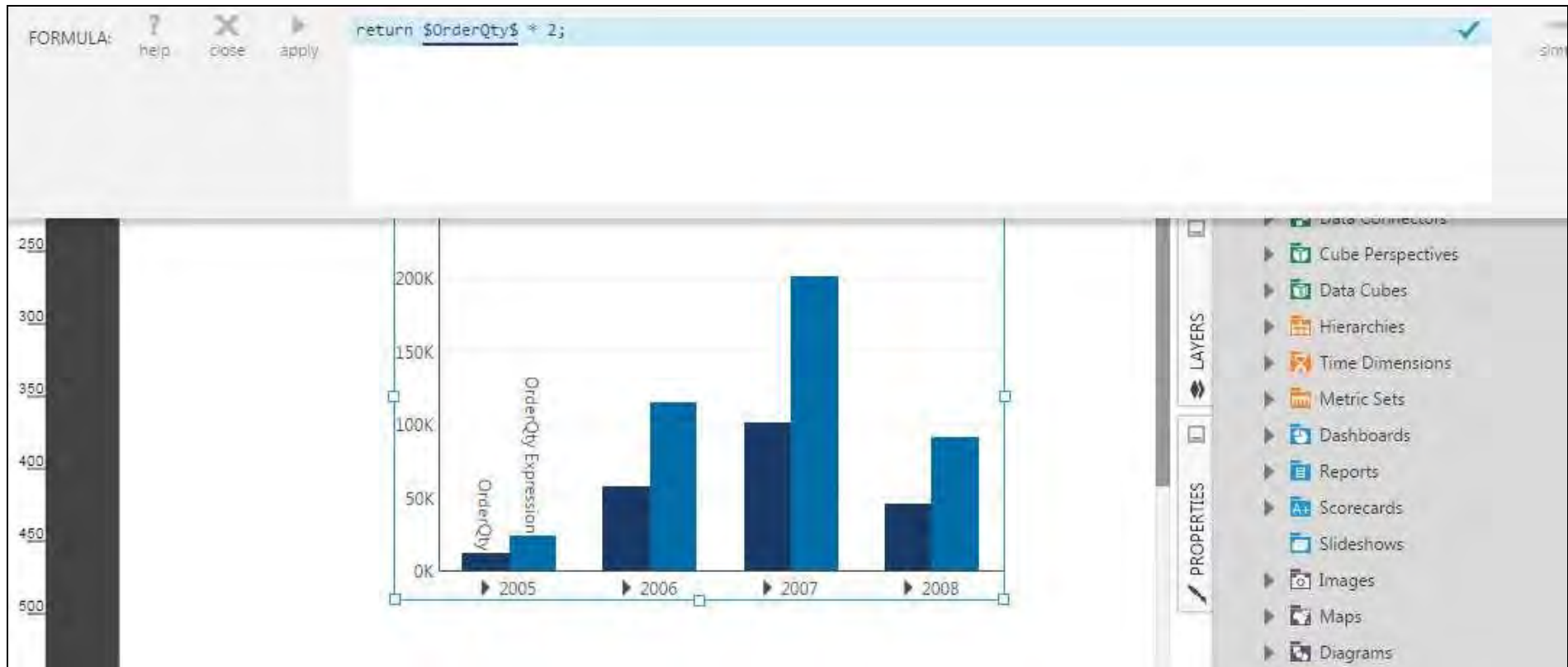
Advanced Mode

Click the advanced button on the right of the formula bar to go into advanced editing mode. This button works like a toggle allowing you to flip between simple and advanced modes.

In advanced mode, you can enter multiple lines of script to more easily include if statements and other syntax available when using [DundasScript](#). The formula should return a value using the return statement for advanced mode, even if it is just a single line.

For example, a constant formula must be entered as:

```
return 40000;
```





Note: You can also write if/else statements on a single line in simple mode. For example: *if (\$Measure\$ > 100) return 7; else if (\$Measure\$ < 0) return -7; else return 0;*

Formula Visualizations

When editing a dashboard or another view, click Data Visualization, and then click Formula Visualization under the Formula category.

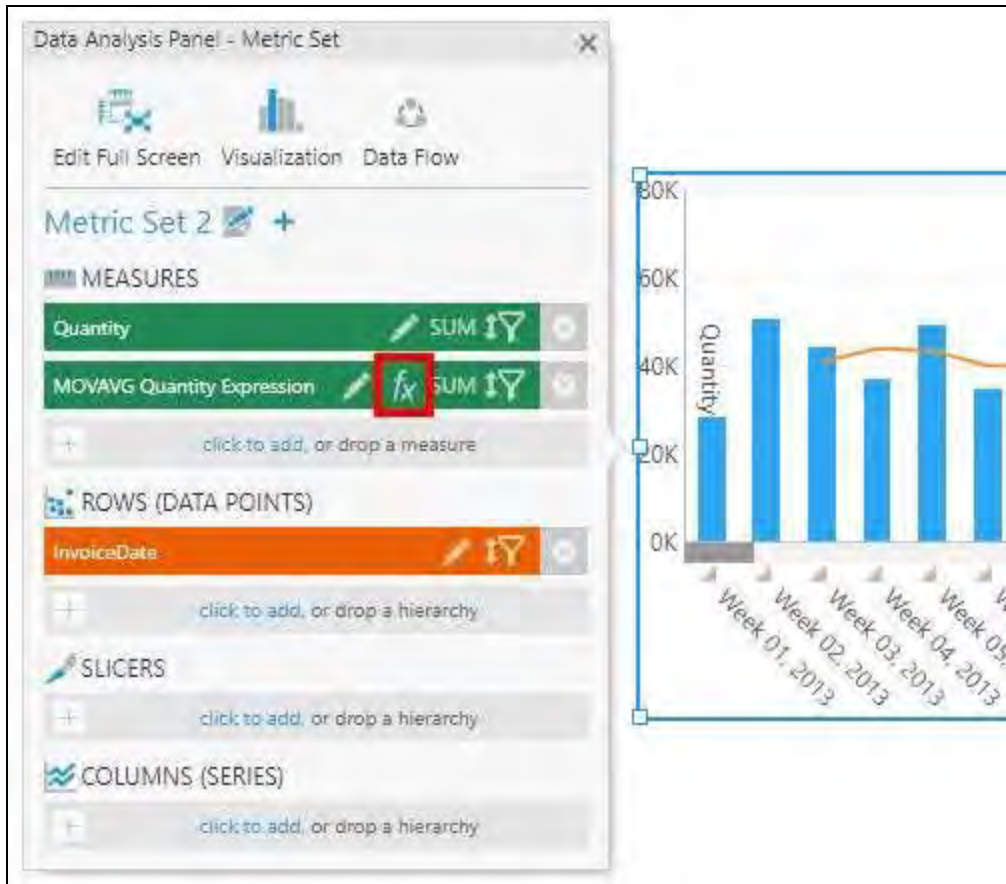


This opens the formula bar just like for adding a formula measure, except that you are able to optionally refer to data from multiple visualizations or metric sets rather than just one. When you click **apply**, an entirely new metric set is created and displayed in a new visualization instead of modifying an existing visualization and metric set. This new metric set is called a formula metric set, and it is able to produce an entirely different output than the metric set(s) it was based on, potentially consisting of different rows or columns as well as different measure values.

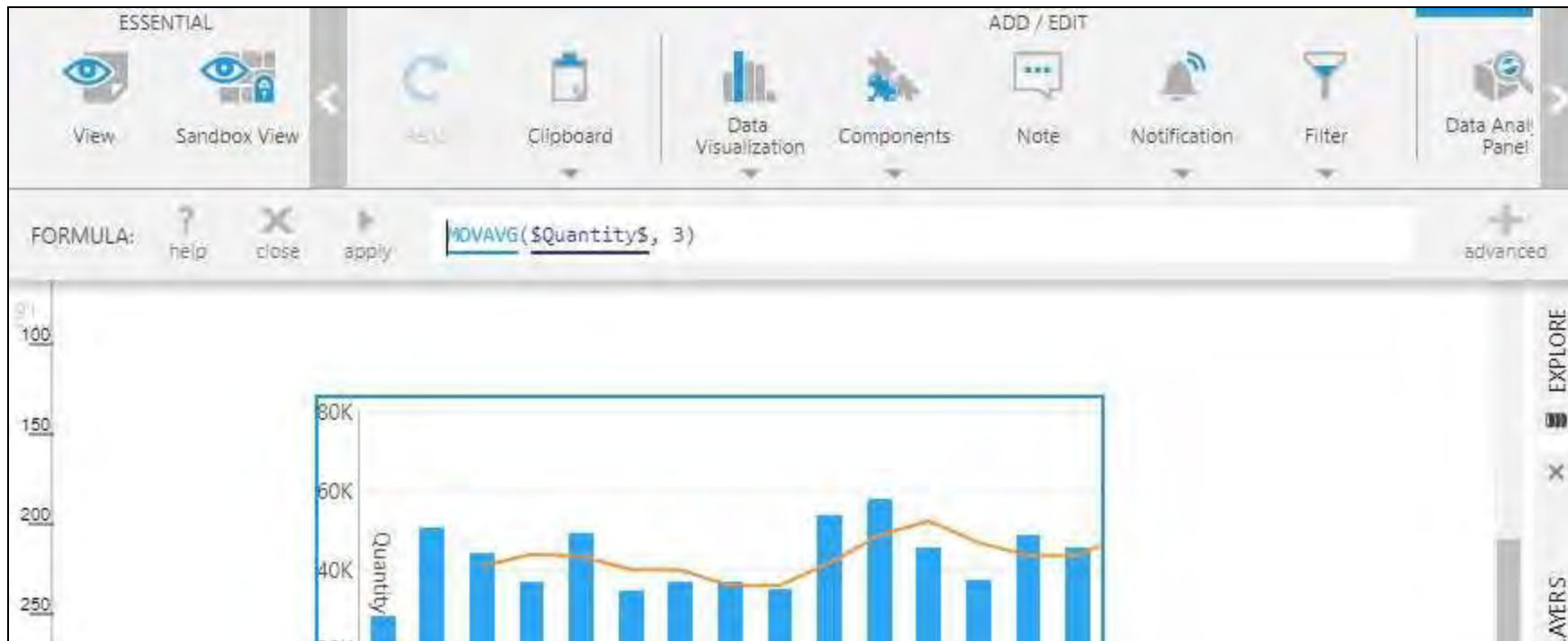
For a walkthrough on adding formula visualizations, see [Using a Formula Visualization](#).

Editing a Formula

Once you've added a formula measure, you can go back and edit it by clicking the *fx* button in the Data Analysis Panel.



The formula bar opens and the corresponding formula is loaded into the formula bar.

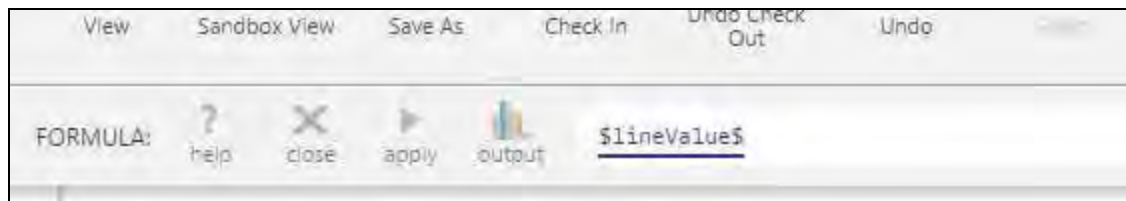


You can make changes to the formula and then click **apply**.

Parameter Placeholders

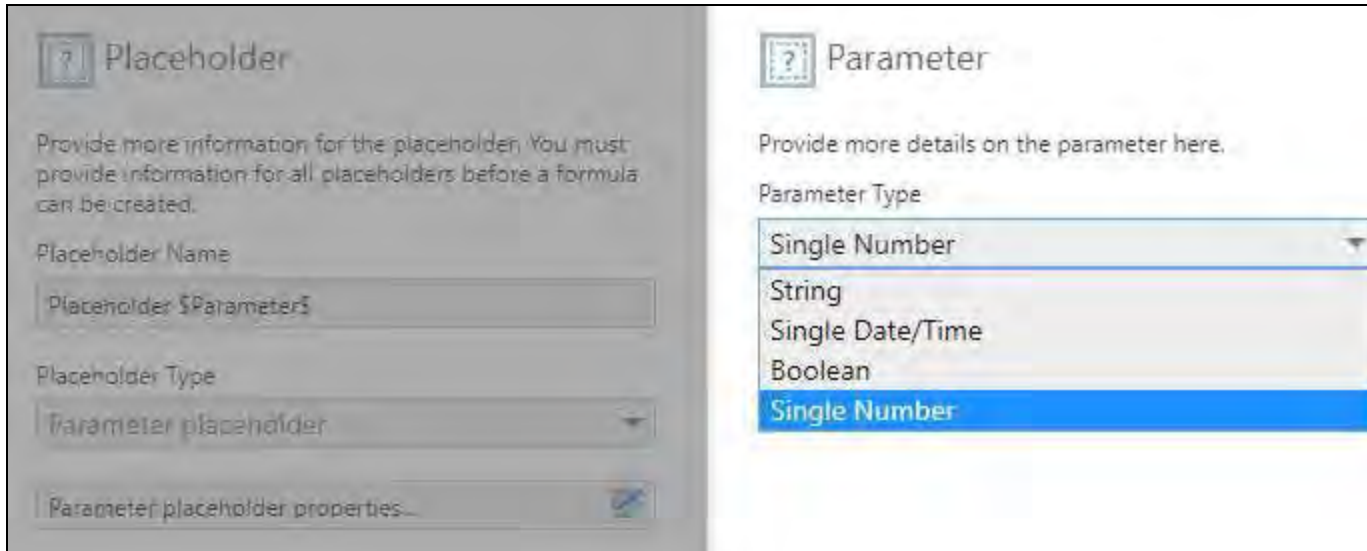
In addition to placeholders for measures and hierarchies, formulas support parameter placeholders, which allow you to refer to the value of a parameter connected to a filter or set in script. This can allow viewers of dashboards and other views to enter their own value into the formula, pass in a value through an [interaction](#), or for custom functionality implemented in script to affect the formula when viewing.

To create a parameter placeholder, simply type one in when adding or editing any formula. Choose a name that's not already taken by an element in your metric set, and surround it with dollar signs (\$):



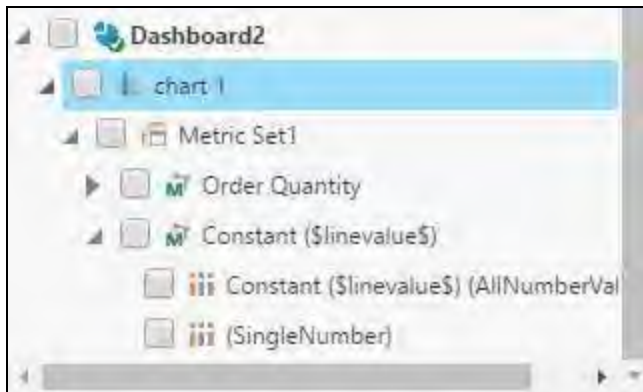
After typing a dollar sign, a popup menu will list the names that are already taken. The placeholder can either appear by itself, or be used anywhere in the formula you would like to be able to change the value later. Click **apply** when done.

This creates a parameter that is numeric. You may want a parameter that isn't numeric, especially if using the advanced mode of the formula bar to write script into your formula. You can change its type by clicking on the placeholder you just typed, which is now underlined, then clicking **Parameter placeholder properties....**

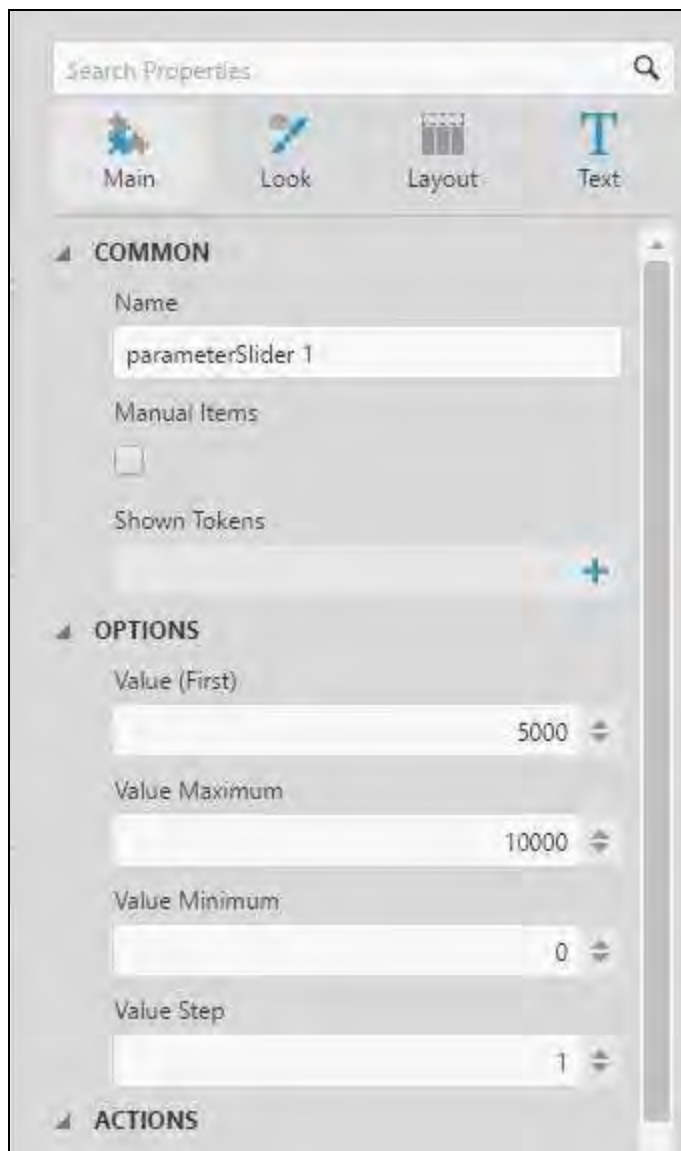


Go to **Filters** in the toolbar and add a filter. Or, to set the parameter value through an interaction or through script, you can open the **Parameters** window and add a new view parameter.

In the connections list that appears, un-check any existing connections, which are for regular filter parameters, and then find and expand your formula measure. Connect to the (*Single Number*) parameter under the formula measure. (If you changed the type of the parameter, look for the single value of that type instead).



For a numeric parameter, you might have chosen a *Slider* filter. You may now want to change its minimum and maximum settings in the **Properties** window to accommodate a particular range of values.



The screenshot shows the 'Properties' window for a 'parameterSlider 1' filter. The window has a search bar at the top and four tabs: 'Main', 'Look', 'Layout', and 'Text'. The 'COMMON' section is expanded, showing the filter's name and a 'Manual Items' checkbox. The 'OPTIONS' section is also expanded, showing four numeric input fields: 'Value (First)' set to 5000, 'Value Maximum' set to 10000, 'Value Minimum' set to 0, and 'Value Step' set to 1. The 'ACTIONS' section is partially visible at the bottom.

You and other viewers can now change the value used in this formula.

Filtering

When filtering a metric set that contains formula measures, formulas are calculated *after* filtering is applied, except for filtering by formula measure values. These operations occur in this order:

1. The metric set's data is filtered by hierarchy values and by measures that are not formulas
2. Formula functions run their calculations based on this data
3. The resulting data is then filtered by formula measure values

For example, the following table shows a formula measure using the [Percent of Total](#) function, which was calculated based on the entire set of rows. Filtering this formula measure to a range of 0.15 (15%) and up preserves and filters by these calculations, displaying only one row:

StoreID	Revenue	PERCENTTOTAL Revenue Expression
4	\$17,479,223.41	16.35%



Note: When you need formulas to be calculated before some filtering is applied, add a formula measure to filter by instead. This formula can compare against hierarchy values or a Qarameter's value.

When formulas are calculated for the members of a [time dimension hierarchy](#) at the Week level, and the results are visualized in a chart set to use a [Date/Time scale](#), the **Week 53** member of one year and **Week 1** member of the following year often must be combined together into a single member representing the full seven days. In this case, the [aggregator](#) set on the formula measure is used to combine the two formula results together in most cases such as for aggregators **Sum**, **Average**, **Minimum**, **Maximum**, and **Count**. For other aggregators, one of the two week's values may be displayed.

For more information, see:

- Video: [Formulas](#)
- Video: [Advanced Formulas](#)
- [List of Formulas](#)
- [Connect to OLAP Data and Apply a Formula](#)
- [Using a Formula Visualization](#)

Accumulation Distribution

This applies to: *Managed Dashboards, Managed Reports*

The **Accumulation Distribution** function is a technical indicator that considers both stock price and volume. When the indicator increases, it implies an accumulation or buying of the related stock. When the indicator decreases, it implies distribution or selling of the related stock. Accumulation Distribution can be used to detect divergences between stock price movement and volume movement. Other applications include confirmation of price trends and providing advance warning of a possible reversal in stock price. A formula bar appears below the main toolbar.

Syntax

```
ACCDIST(d0,d1,d2,d3,d4,Alignment)
```

Input

The Accumulation Distribution function requires the following input series:

- d0 - High data values - The first set of data values for which the Accumulation Distribution is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Accumulation Distribution is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Accumulation Distribution is calculated, usually the daily close price of a stock.
- d3 - Volume data values - The fourth set of data values for which the Accumulation Distribution is calculated, usually the daily volume.

Parameters

The Accumulation Distribution function requires the following parameters:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Accumulation Distribution function generates the following output:

- Accumulation Distribution - The Accumulation Distribution result set.



- Archive of documentation for Logi Symphony v24

For more information, see [Adding Formulas](#).

Aroon Oscillator

This applies to: *Managed Dashboards, Managed Reports*

The **Aroon Oscillator** function is calculated as the difference between the Aroon Up and Aroon Down indicators, with resulting values ranging from -100 to 100. An uptrend in stock price is indicated when the Aroon Oscillator value is positive. A downtrend is indicated when the oscillator value is negative. The further away the Aroon Oscillator value is from the zero line, the stronger the trend. Applications of the Aroon Oscillator include determining whether a stock is trending or not, and gauging the strength of the current trend.

Syntax

Aroon Oscillator:

```
AROONOSC (d0,s0,Alignment)
```

Aroon Up indicator:

```
AROONOSCUP (d0,s0,Alignment)
```

Aroon Down indicator:

```
AROONOSCDOWN (d0,s0,Alignment)
```

Input

The Aroon Oscillator functions require the following input series:

- d0 - Input data values - The set of data values for which the Aroon Oscillator is calculated, usually the daily close price of a stock.

Parameters

The Aroon Oscillator function requires the following parameters:

- s0 - Period - The number of time periods to use in the calculation. Default value is 14.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.



Output

The AROONOSC function generates the following output:

- Aroon Oscillator - The Aroon Oscillator result set, which consists of values between -100 and 100, with the zero value representing a crossover point.

The AROONOSCUP function generates the following output:

- Aroon Up - The Aroon Up indicator, which consists of values between 0 and 100. A higher value indicates a stronger uptrend. The AROONOSCDOWN function generates the following output:
- Aroon Down -The Aroon Down indicator, which consists of values between 0 and 100. A higher value indicates a stronger downtrend.

Average Directional Index

This applies to: *Managed Dashboards, Managed Reports*

The **Average Directional Index** function is based on the indicator developed by J. Welles Wilder. It combines two other indicators, the Positive Directional Indicator and the Negative Directional Indicator, which were also developed by Wilder. Applications of ADX include indicating the strength of the current trend in stock price (but not the trend direction).

Syntax

Average Directional Index:

```
AVGDIRECT ( d0, d1, d2, s0, Alignment)
```

Positive Directional Indicator:

```
AVGDIRECTPOS ( d0, d1, d2, s0, Alignment)
```

Negative Directional Indicator:

```
AVGDIRECTNEG( d0, d1, d2, 50, Alignment)
```

Input

The Average Directional Index functions require the following input series:

- d0 - High data values - The first set of data values for which the Average Directional Index is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Average Directional Index is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Average Directional Index is calculated, usually the daily close price of a stock.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Parameters

The Average Directional Index function requires the following parameters:

- s0 - Period - The number of time periods to use in the calculation. Default value is 14.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The AVGDIRECT function generates the following output:

- Average Directional Index - The Average Directional Index result set. Values range between 0 and 100. The AVGDIRECTPOS function generates the following output:
- Positive Directional Indicator - The Positive Directional Indicator result set. The AVGDIRECTNEG function generates the following output:
- Negative Directional Indicator - The Negative Directional Indicator result set.

Average True Range

This applies to: *Managed Dashboards, Managed Reports*

The **Average True Range** function computes an average of the True Range, which represents the range in price of a stock or commodity over the course of one day, or from one day to the next. Average True Range is used to indicate price volatility.

Syntax

```
AVGTRURANGE (d0,d1,d2,s0,Alignment)
```

Input

The Average True Range function requires the following input series:

- d0 - High data values - The first set of data values for which the Average True Range is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Average True Range is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Average True Range is calculated, usually the daily close price of a stock.

Parameters

The Average True Range function has the following parameter:

- s0 - Period - The number of time periods to use in the calculation. Default value is 14.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Average True Range function generates the following output

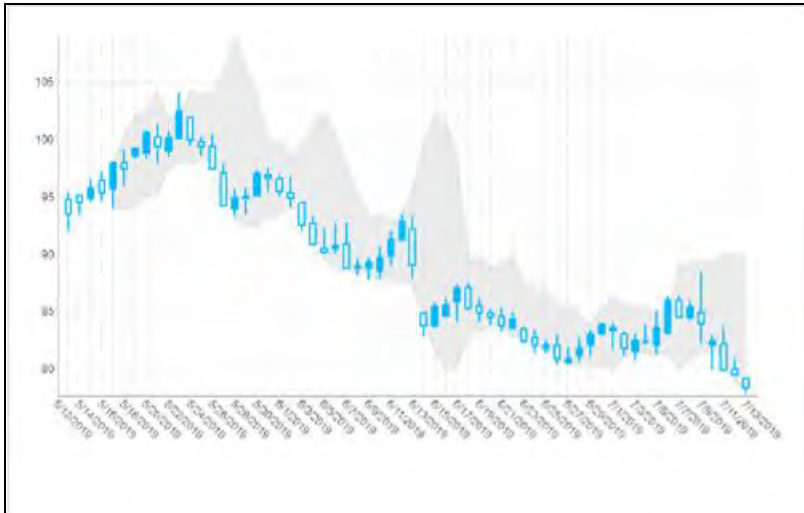
- Average True Range - The Average True Range result set.

Bollinger Bands

This applies to: *Managed Dashboards, Managed Reports*

The **Bollinger Bands** functions are based on methods developed by John Bollinger. They compute a pair of data values based on the [moving average](#) of the input value series.

Each value is derived by calculating a specified number of standard deviations based on the moving average of the selected input value series. Applications of Bollinger Bands include the measurement of stock price volatility.



Note: To use an exponential moving average instead of the simple one, see [Moving Average Envelopes](#).

Syntax

Upper Bollinger Band:

```
BOLLINGERUPPER(d0,s0,s1,Alignment)
```

Lower Bollinger Band:

```
BOLLINGERLOWER(d0,s0,s1,Alignment)
```



Input

The Bollinger Bands functions require the following input:

- d0 - The input data values for which the Bollinger Bands function is calculated.

Parameters

The Bollinger Bands functions accept the following parameters:

- s0 - Period - The number of time periods to use in the calculation. Default value is 10.
- s1 - Standard Deviation Shift - The number of standard deviations that is used to determine the values of the Upper and Lower Bollinger Bands in relation to the simple moving average. Default value is 2.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Bollinger Bands functions generate the following output:



- Archive of documentation for Logi Symphony v24

- Upper Bollinger Band - The Upper Bollinger Band result set.
- Lower Bollinger Band - The Lower Bollinger Band result set.

For more information, see:

- [Adding Formulas](#)
- [Moving Average](#)
- [Moving Average Envelopes](#)

Chaikin Money Flow

This applies to: *Managed Dashboards, Managed Reports*

The **Chaikin Money Flow** function is based on an indicator developed by Marc Chaikin. It computes the sum of the Accumulation Distribution line values for a stock over a specified number of time periods, and divides this by the total stock volume over the same duration. When the CMF value is above zero, this indicates buying pressure. When the CMF value is below zero, this indicates selling pressure. Applications of Chaikin Money Flow include determining if a stock is under accumulation or distribution. For example, if the CMF value remains above zero for an extended period of time, this indicates accumulation.

Syntax

```
CHKMONEYFLO(d0,d1,d2,d3,s0,Alignment)
```

Input

The Chaikin Money Flow function requires the following input series:

- d0 - High data values - The first set of data values for which the Chaikin Money Flow is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Chaikin Money Flow is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Chaikin Money Flow is calculated, usually the daily close price of a stock.
- d3 - Volume data values - The fourth set of data values for which the Chaikin Money Flow is calculated, usually the daily volume.

Parameters

The Chaikin Money Flow function has the following parameter:

- s0 - Short Period - The short time period to use in the calculation. Default value is 21.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Chaikin Money Flow function generates the following output:



- Archive of documentation for Logi Symphony v24

- Chaikin Money Flow - The Chaikin Money Flow result set.

Chaikin Oscillator

This applies to: *Managed Dashboards, Managed Reports*

The **Chaikin Oscillator** function is based on an indicator developed by Marc Chaikin. It is calculated by subtracting the 10-day exponential moving average from the 3-day exponential moving average of the Accumulation Distribution line. Resulting values oscillate around the zero level. Applications of Chaikin Oscillator include confirmation of price trends and providing advance warning of price reversals.

Syntax

```
CHKOSC(d0,d1,d2,d3,s0,s1,Alignment)
```

Input

The Chaikin Oscillator function requires the following input series:

- d0 - High data values - The first set of data values for which the Chaikin Oscillator is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Chaikin Oscillator is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Chaikin Oscillator is calculated, usually the daily close price of a stock.
- d3 - Volume data values - The fourth set of data values for which the Chaikin Oscillator is calculated, usually the daily volume.

Parameters

The Chaikin Oscillator function has the following parameters:

- d0 - Short Period - The short time period to use in the calculation. Default value is 3.
- d1 - Long Period - The long time period to use in the calculation. Default value is 10.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.



- Archive of documentation for Logi Symphony v24

Output

The Chaikin Oscillator function generates the following output

- Chaikin Oscillator - The Chaikin Oscillator result set.

Commodity Channel Index

This applies to: *Managed Dashboards, Managed Reports*

The **Commodity Channel Index** function is based on an indicator developed by Donald Lambert. It compares a stock price to its average over a specified number of time periods. Applications of CCI include identification of overbought or oversold conditions, and signaling when a price trend may be weakening.

Syntax

```
COMMCHANNEL(d0,d1,d2,s0,Alignment)
```

Input

The Commodity Channel Index function requires the following input series:

- d0 - High data values - The first set of data values for which the Commodity Channel Index is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Commodity Channel Index is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Commodity Channel Index is calculated, usually the daily close price of a stock.

Parameters

The Commodity Channel Index function has the following parameter:

- s0 - Period - The number of time periods to use in the calculation.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Commodity Channel Index function generates the following output:

- Commodity Channel Index - The Commodity Channel Index result set.

Correlation

This applies to: *Managed Dashboards, Managed Reports*

The **Correlation** function calculates the correlation, or the strength of the relationship between two input data series.

The resulting correlation value ranges between -1 and 1, where a value close to -1 indicates a negative correlation, a value close to 1 indicates a positive correlation, and a value close to zero indicates no correlation.

A negative correlation means that when the first input is large, the second input is small (and vice-versa) whereas a positive correlation means the both input values become larger and smaller at the same time.

close

apply

CORRELATION (\$Sales\$, \$Sales2\$)

Day	Sales	Sales2	Script Measure
Grand Total	842.00	849.00	
Jan 01, 2014	65.00	62.00	-0.30
Jan 08, 2014	60.00	65.00	-0.30
Jan 15, 2014	55.00	70.00	-0.30
Jan 22, 2014	52.00	65.00	-0.30
Jan 29, 2014	56.00	60.00	-0.30
Feb 05, 2014	58.00	58.00	-0.30
Feb 12, 2014	59.00	56.00	-0.30
Feb 19, 2014	62.00	57.00	-0.30
Feb 26, 2014	63.00	62.00	-0.30
Mar 05, 2014	65.00	63.00	-0.30
Mar 12, 2014	61.00	65.00	-0.30
Mar 19, 2014	63.00	60.00	-0.30
Mar 26, 2014	62.00	54.00	-0.30
Apr 02, 2014	61.00	52.00	-0.30



The **Correlation Group** function can optionally group the input data and calculate separate correlation values for each.

Syntax

Correlation:

```
CORRELATION(d0,d1,Alignments...)
```

Correlation Group:

```
CORRELATIONGROUP(d0,d1,Alignments...)
```

Input

The Correlation functions require the following input series:

- d0: First Input data values - The first set of data values for which the Correlation is calculated.
- d1: Second Input data values - The second set of data values for which the Correlation is calculated.

Parameters

The Correlation functions accept the following parameters:

- Alignments... (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.
 - For CORRELATION, specifying a hierarchy calculates the correlation for the corresponding input values separately each time they occur: e.g., **CORRELATION(\$Measure1\$, \$Measure2\$, \$Year\$)** can calculate separate correlation values within each **Country** group.
 - For CORRELATIONGROUP, specify the hierarchies by which to group the input values and calculate the correlation within each group: e.g., **CORRELATIONGROUP(\$Measure1\$, \$Measure2\$, \$Country\$)** will calculate the correlation between **Measure1** and **Measure2** values separately within each country.

Output

The Correlation functions generate the following outputs:



- Archive of documentation for Logi Symphony v24

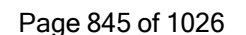
- Correlation - A single value representing the correlation between the two sets of input values (or for each group when using alignment parameters).

For more information, see:

- [Correlation Matrix](#)
- [Adding Formulas](#)
- [Using a Formula Visualization](#)

This applies to: *Managed Dashboards, Managed Reports*

Zero implies that the two series are independent; a value close to 1 implies they are positively correlated, and a value close to -1 implies they are negatively correlated. A negative correlation means that when the first input is large, the second input is small, and vice-versa.





Note: With multiple measures and many rows of data selected, re-visualize to a Correlation Matrix to quickly produce a preconfigured correlation matrix visualization. See Using [recommendations](#) for details.



Note: The correlation matrix function can only be used in a [formula visualization](#).

Syntax

```
CORRMATRIX (d0...)
```

Input

The Correlation Matrix requires the following inputs:

- d0 (One or More) - The data series (i.e., measures) for which the Correlation Matrix is calculated. Must contain at least one data series, but should contain multiple for calculating the correlation between each pair.

Parameters

The Correlation Matrix does not use any parameters.

Output

The Correlation Matrix function generates the following output:

Correlation Matrix - the Correlation Matrix result set containing a measure with the correlation values between each pair of inputs, a hierarchy of measure names placed on Rows, and a hierarchy of measure names placed on Columns in the metric set.



Note: You are not allowed to change the placement of the hierarchies in the resulting metric set.

For more information, see:

- [Using a Formula Visualization](#)
- [List of Formulas](#)



- Archive of documentation for Logi Symphony v24

- [Using Symphony Recommendations](#)
- [Correlation](#)

Cumulative Total

This applies to: *Managed Dashboards, Managed Reports*

The **Cumulative Total** function computes the cumulative total (or running sum) of a set of input values.



The **Cumulative Total Group** function can be optionally used to compute the cumulative total (or running sum) of a set of input values grouped by a specific hierarchy.

Syntax

Cumulative Total:

```
CMLTOTAL(d0,Alignments...)
```

Cumulative Total Group:

```
CMLTOTALGROUP(d0,Alignments...)
```

Input

The Cumulative Total functions require the following input:

- d0 - The set of data values for which the Cumulative Total is calculated.

Parameters

The Cumulative Total functions accept the following parameters:

- Alignments... (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.
 - For CMLTOTAL, specifying a hierarchy calculates cumulative totals for the corresponding input values separately each time they occur: e.g., **CMLTOTAL(\$Measure\$, \$Year\$)** can calculate separate running totals of yearly values within each **Country** group.
 - For CMLTOTALGROUP, specify the hierarchies by which to group the input values and calculate cumulative totals within each group: e.g., **CMLTOTALGROUP(\$Measure\$, \$Country\$)** will calculate a running total of the values within each country.

Output

The Cumulative Total function generates the following output:

- Cumulative Total - The Cumulative Total result set.

The Cumulative Total Group function generates the following output:

- Cumulative Total Group - The Cumulative Total Group result set.

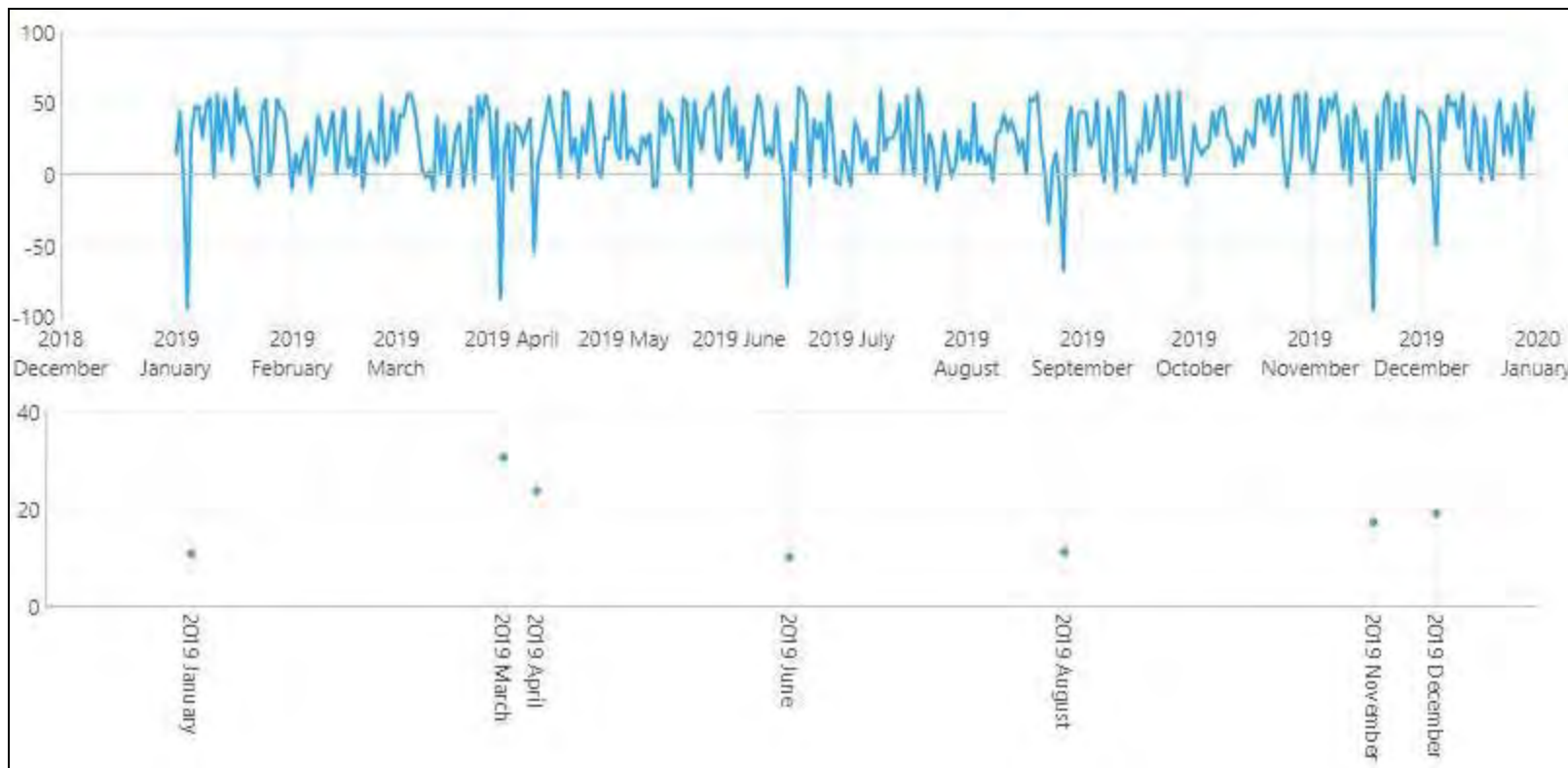
For more information, see:

- [Adding Formulas](#)
- [List of Formulas](#)

Data Outliers

This applies to: *Managed Dashboards, Managed Reports*

The **Data Outlier** functions identify the outliers in a set of input values. You can hide or replace the outliers to display only non-outliers, or the result can include only the outliers to display them or to format them [using states](#).



The **Outlier Interquartile** function uses a multiple of the interquartile range (IQR) to determine which values are outliers, typically with a multiple of 1.5. The interquartile range is the difference between the 75th and 25th percentiles. A multiple of 3 is sometimes used to further identify extreme or 'major' outliers, and you can use the parameters of this function to show or hide major and minor outliers separately if you choose.

The **Outlier Tau** function uses the modified Thompson Tau test to determine the outliers.

Syntax

Outlier Interquartile:

```
OUTIQ(d0,s0,s1,s2,s3,s4,s5,Alignments)
```

Outlier Tau:

```
OUTTAU(d0,s0,s1,s2,Alignments)
```

Input

- d0 - The set of data values for which the outliers is calculated.

OUTIQ parameters

- s0 - The value used to determine the position of the minor outliers. The default value is 1.5 and the supported range is from 1 to 5.
- s1 - The value used to determine the position of the major outliers. The default value is 3 and the supported range is from 1 to 10.
- s2 - Indicates that all non-outlier data points should be hidden in the result. The default value is False.
- s3 - Indicates that all minor outliers should be hidden in the result. The default value is True.
- s4 - Indicates that all major outlier data points should be hidden in the result. The default value is True.
- s5 - Indicates what value should be used to replace any hidden outliers. Available choices include:
 - 0 - null values
 - 1 - zero
 - 2 - neighbor's average
 - 3 - global average
- Alignments (optional) - A hierarchy placeholder to be used as the alignment axis.

Output parameters

- s0 - Indicates that all non-outlier data points should be hidden in the result. The default value is False.
- s1 - Indicates that all outliers should be hidden in the result. The default value is True.
- s2 - Indicates what value should be used to replace any hidden outliers. Available choices include:
 - 0 - null values
 - 1 - zero
 - 2 - neighbor's average
 - 3 - global average
- Alignments (optional) -A hierarchy placeholder to be used as the alignment axis.

Output

The Data Outlier functions generate the following output:

- The Data Outliers result set, consisting of the input values with non-outlier values hidden and/or with outlier values hidden or replaced as specified in the parameters.

For more information, see:

- [Creating a Box Plot](#)
- [Adding Formulas](#)
- [List of Formulas](#)

Distance

This applies to: *Managed Dashboards, Managed Reports*

The **Distance** function computes the as-the-crow-flies distance between a source and destination identified by latitude and longitude.

close apply DISTANCE(\$SourceLatitude\$, \$SourceLongitude\$, \$TargetLatitude\$, \$TargetLongitude\$)

SourceCity	SourceLatitude	SourceLongitude	TargetCity	TargetLatitude	TargetLongitude	DISTANCE (KM)
Aberdeen, Scotland	57.00	2.00	Beijing, China	39.00	116.00	7,691.541645
Bangkok, Thailand	13.00	100.00	Cairo, Egypt	30.00	31.00	7,282.412910
Kingsville, Canada	42.00	82.00	Munich, Germany	48.00	11.00	5,422.704955

Syntax

```
DISTANCE ( d0, d1, d2, d3, s0)
```

Input

The Distance function requires the following input series:



- d0 - The latitude of the starting location.
- d1 - The longitude of the starting location.
- d2 - The latitude of the ending location.
- d3 - The longitude of the ending location.

Parameters

The Distance function accepts the following parameters:

- s0 - Specifies unit to use for the result. 0 for kilometers, 1 for miles. The default is 0.

Output

The Distance function generates the following output:

- The Distance result set, consisting of distances between each pair of coordinates in kilometers or miles as specified above.

Using a Custom Total Calculation

This applies to: *Managed Dashboards, Managed Reports*

In some cases, it is useful to customize the total displayed for a measure. For instance, when calculating the percentage that a measure value makes up of the total of several measures, the sum or the average of all the calculated values in the rows may not be as helpful as the calculating the percentage again based on the measures' overall totals.

Group	A	B	A / (A+ B)		
Apple	1	7	12.50%		
Apple	2	11	15.38%		
Apple	3	13	18.75%		
Apple	5	17	22.73%		
Apple	7	19	26.92%		
Apple	11	23	32.35%		
Total:	29	90	X	SUM	128.64%
Banana	13	29	30.95%	AVG	21.44%
Banana	17	31	35.42%	Calc.	24.37%
Banana	19	37	33.93%		
Banana	23	39	37.10%		
Banana	29	41	41.43%		
Banana	31	43	41.89%		
Total:	132	220	X	SUM	220.71%
AVG36.79%					
Calc.37.50%					

Totals calculation rules allow you to customize how the totals are calculated.

Creating a Custom Total Formula

This example will add the formula $A/(A+B)$ as the total calculation.

Group	Index	A	B	A / (A+ B)
Grand Total		409.00	510.00	1,079.92 %
▲ Apple	(All)	29.00	90.00	128.64 %
	1	1.00	7.00	12.50 %
	2	2.00	11.00	15.38 %
	3	3.00	13.00	18.75 %
	4	5.00	17.00	22.73 %
	5	7.00	19.00	26.92 %
	6	11.00	23.00	32.35 %
▲ Banana	(All)	132.00	220.00	220.71 %
	1	13.00	29.00	30.95 %
	2	17.00	31.00	35.42 %
	3	19.00	37.00	33.93 %
	4	23.00	39.00	37.10 %
	5	29.00	41.00	41.43 %
	6	31.00	43.00	41.89 %

To create a totals calculation rule, click on the measure in the Data Analysis Panel to open its measure configuration dialog, click to expand **Totals Calculation Rules** for either rows or columns, then click **Define totals calculation rule formula**.

Analysis Element Name

▶ Element Settings

▼ Totals Calculation Rules (Rows)



▶ Totals Calculation Rules (Columns)

▶ Parameter Values

The formula bar will appear.

FORMULA:

? help
X close
▶ apply
+ advanced

Formulas are entered in a similar way as for a regular [formula measure](#): you can use [formula functions](#) such as *SUM*, and measure placeholders such as *\$A\$* for our measure A.



Note: You can click a measure in your visualization to insert its placeholder into the formula bar.

There are multiple ways you can determine your totals:

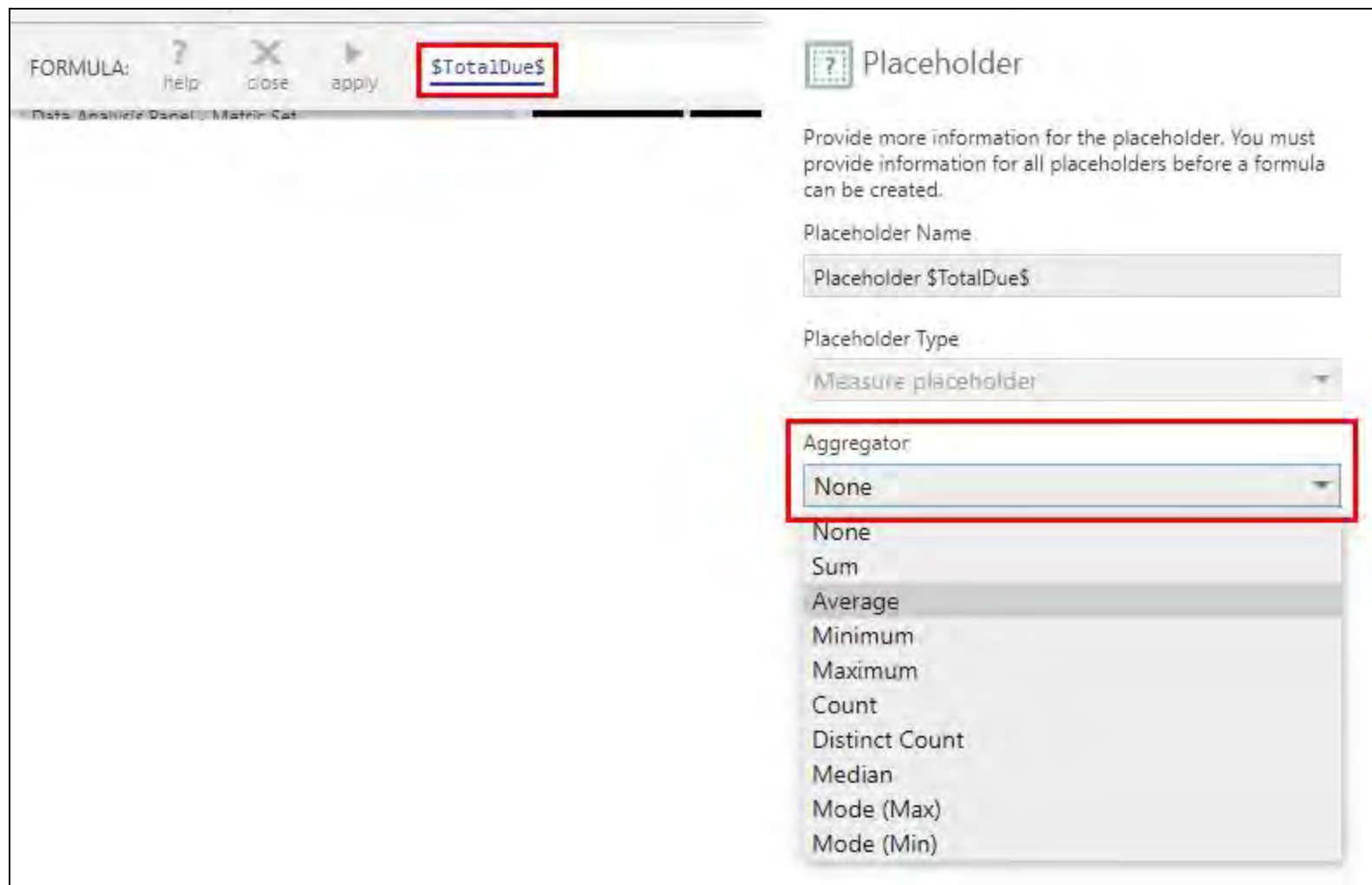
- You can pass measures as arguments to [functions](#) such as *AVG* or *SUM* to determine how the individual values should be totaled. In the example above, the measures A and B both aggregate using **Sum**, and we will want to use the corresponding *SUM* function for each measure in order to calculate the overall percentage in this case. You are free to choose functions that don't correspond with the measure's aggregator if needed for your purposes.

FORMULA:

? help
X close
▶ apply
☒
+ advanced

- Or, you can instead refer to measures directly without calling a function, to use their original totals as determined by their aggregator. When you do this, you have the option of customizing the aggregator used just for this total by clicking on the underlined measure placeholder to open the *Placeholder* dialog, then

setting **Aggregator**.



FORMULA: ? help close apply \$TotalDue\$

Data Analysis Panel: Matrix Set

Placeholder

Provide more information for the placeholder. You must provide information for all placeholders before a formula can be created.

Placeholder Name
Placeholder \$TotalDue\$

Placeholder Type
Measure placeholder

Aggregator
None

- None
- Sum
- Average
- Minimum
- Maximum
- Count
- Distinct Count
- Median
- Mode (Max)
- Mode (Min)



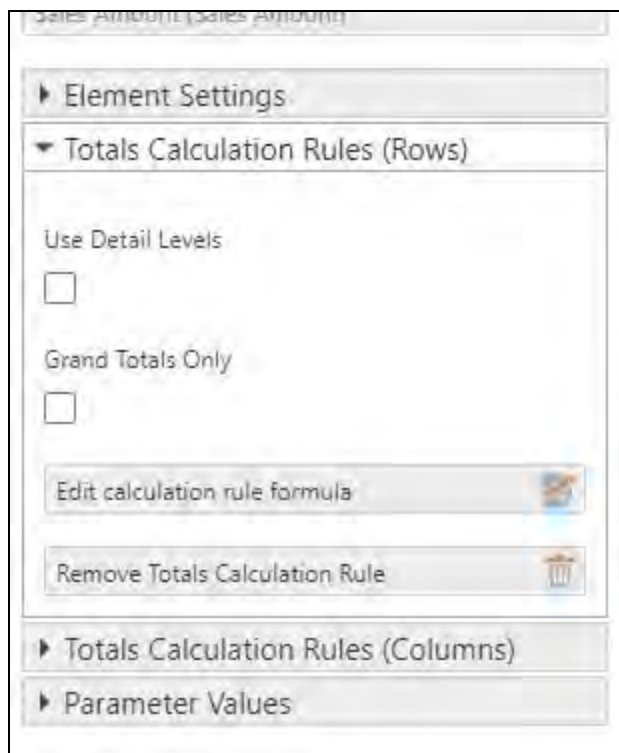
Note: Use measure placeholders without calling functions or customizing the aggregator for optimizations such as avoiding the retrieving of collapsed data from the data source.

Once you complete the formula, click **apply** to return to the measure configuration dialog. You can then submit the dialog when finished.

Group	Index	A	B	A / (A+ B)
Grand Total		409.00	510.00	44.50 %
▲ Apple	(All)	29.00	90.00	24.37 %
	1	1.00	7.00	12.50 %
	2	2.00	11.00	15.38 %
	3	3.00	13.00	18.75 %
	4	5.00	17.00	22.73 %
	5	7.00	19.00	26.92 %
	6	11.00	23.00	32.35 %

Other Options

There are options available that can influence how the custom total formula is used.



Use Detail Levels - If checked, grand totals use the set of data created from the details level, i.e., the lowest-selected level of the last hierarchy selected. If unchecked, grand totals are calculated using the set of values created from the subtotals.

Grand Totals Only - only applies the formula to the grand total at the beginning or end, and not the subtotals displayed when there are multiple hierarchies selected under Rows or Columns in the metric set.

For more information, see:

- [Create a Metric Set and Add a Filter](#)
- [Slicers Versus Columns and Rows](#)
- [Adding Formulas](#)



- Archive of documentation for Logi Symphony v24

- [List of Formulas](#)
- [Writing Data Scripts with DundasScript](#)

Exponential Moving Average

This applies to: *Managed Dashboards, Managed Reports*

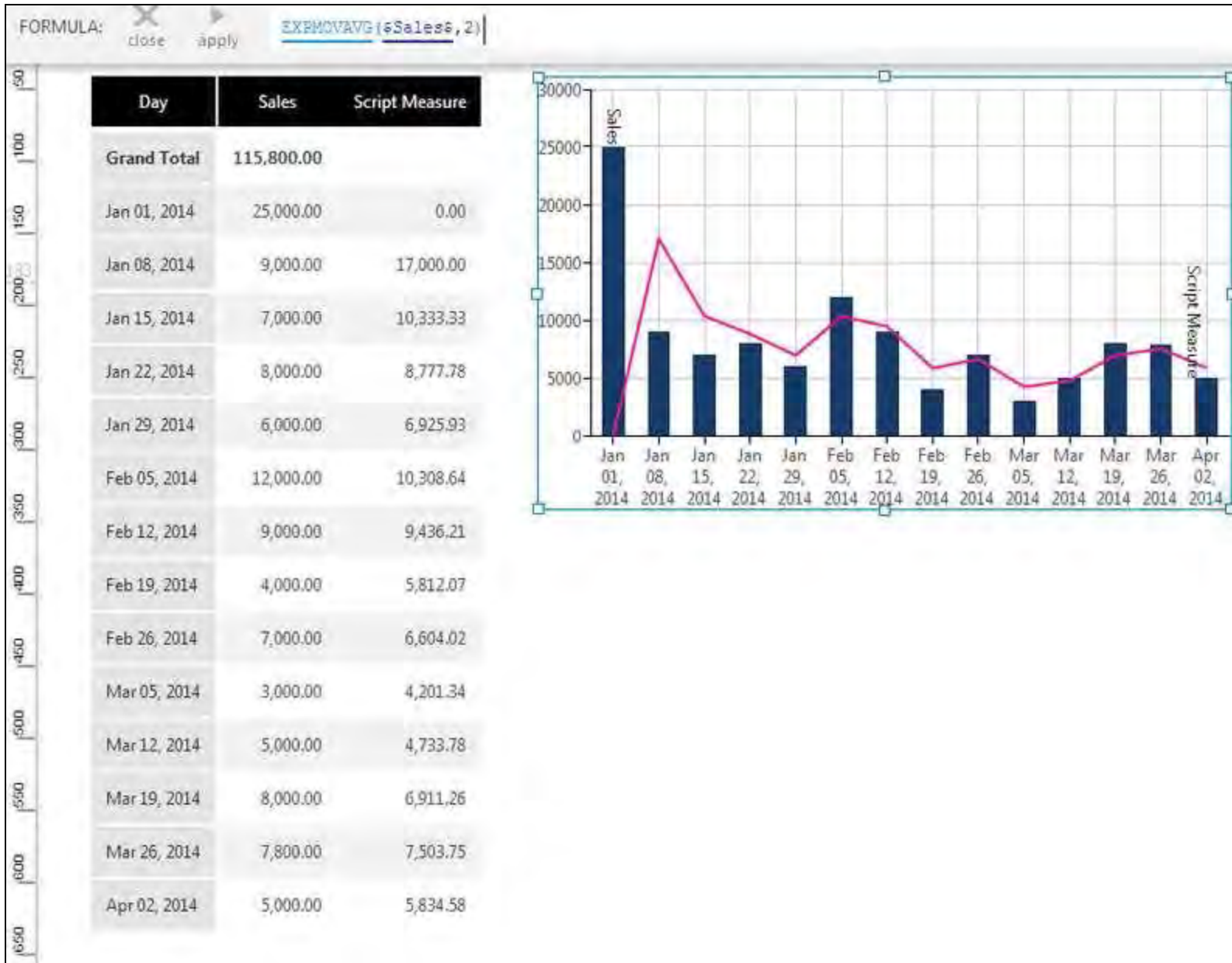
The **Exponential Moving Average**, or exponentially weighted moving average (EWMA), function computes the average of a set of input values over a specified number of periods. In this function, a greater weight is given to more recent data. This function can be used to smooth a data series, which helps to reduce noise and make it easier to spot data trends.

The mathematical formula being calculated is as follows:

$$EWMA_t = \lambda Y_t + (1 - \lambda)EWMA_{t-1}$$

$$for\ t = 1, 2, \dots, n.$$

Where EWMA₀ is the mean of historical data, Y is the value, n is the number of periods, and λ is the weight constant, which is set to $2 / (n + 1)$. You can find more information from the [National Institute of Standards and Technology](#).



Syntax

```
EXPMOAVG(d0,s0,Alignment)
```

Input

The Exponential Moving Average function requires the following input:

- d0 - The set of data values for which the Exponential Moving Average is calculated.

Parameters

The Exponential Moving Average function requires the following parameters:

- s0 - The number of periods to use in the calculation. The default value is 10.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Exponential Moving Average function generates the following output:

- Exponential Moving Average - The Exponential Moving Average result set.

For more information, see:

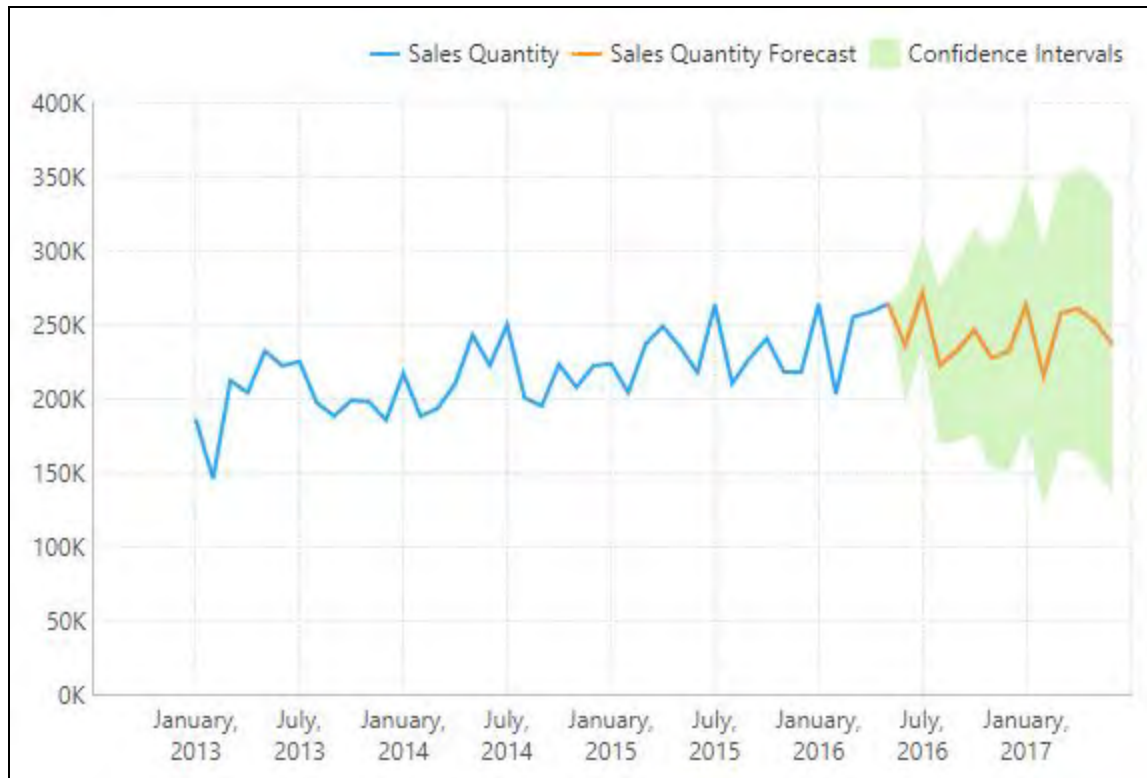
- [Adding Formulas](#)
- [Moving Average](#)
- [Weighted Moving Average](#)

Exponential Smoothing

This applies to: *Managed Dashboards, Managed Reports*

The **Forecasting and Exponential Smoothing** functions use exponential smoothing models to predict future values based on an analysis of historical time series data. These models apply an exponentially-decreasing weight to historical data in order to forecast future values based on emerging trends and can consider seasonal effects.

These functions are typically used with a date/time axis but will also work with a numeric hierarchy or measure as the first input. Also look for the [Forecast Analysis](#) option recommended in the Re-Visualize menu to apply these functions automatically.



Note: For background information, see the OTexts article about [exponential smoothing taxonomy](#) and the [Nelder-Mead method for optimization fitting](#).



Syntax

You can use two functions to perform forecasting using exponential smoothing in Symphony: EXPSMOOTH, or the simplified FORECASTING function.

The FORECASTING function leaves out many of the parameters available for EXPSMOOTH so that their default values take effect. By default, many of these parameters choose an effective setting for you automatically after analyzing your data.

Forecasting

```
FORECASTING(d0,d1,s0,s1)
```

Upper Band Error:

```
FORECASTING_UPPER(d0,d1,s0,s1)
```

Lower Band Error:

```
FORECASTING_LOWER(d0,d1,s0,s1)
```

The same syntax can also be used to return the axis with FORECASTINGAXIS and the season length with FORECASTINGSEASONLENGTH.

Expsmooth

```
EXPSMOOTH(d0,d1,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10)
```

Upper Band Error:

```
EXPSMOOTH_UPPER(d0,d1,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10)
```

Lower Band Error:

```
EXPSMOOTH_LOWER(d0,d1,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10)
```



The same syntax can be used to return the alpha (EXPSMOOTHALPHA), axis (EXPSMOOTHAXIS), beta (EXPSMOOTHBETA), gamma (EXPSMOOTHGAMMA), mean squared error (EXPSMOOTHMSE), phi (EXPSMOOTHPHI), season length (EXPSMOOTHINGSEASONLENGTH), season type (EXPSMOOTHSEASONTYPE), and trend type (EXPSMOOTHTRENDTYPE).

Input

The Exponential Smoothing functions require the following input series:

- d0 - Trend hierarchy - The hierarchy over which to perform the forecast (e.g. datetime axis).
- d1 - Base values - The set of historical data values to be used in the formula.

Parameters

- s0 - Forecast Length - The number of periods to forecast after the end of historical data. A value of -1 indicates the default value, which is 1/3 of the length of the historical data.
- s1 - Season Length - The length of the season, in periods, over which to find seasonal patterns. A value of -1 indicates to use the default, which is automatically selected by Dundas BI.
- s2 - Confidence Level - The probability that the future data will fall within the upper and lower band errors. The default value is 0.95.
- s3 - Show Whole Result - Indicates whether to show only the forecast or the results across all periods. The default is 0 (false).
- s4 - Optimizer Passes - The number of data passes the optimizer should make to find optimal the forecasting values alpha, beta, gamma, and phi. The default value is 10.
- s5 - Trend Smoothing Type - The trend smoothing method to be used. The following options are available:
 - -1 - Choose automatically (default)
 - 0 - No trend smoothing
 - 1 - Additive
 - 2 - Additive Damped



- 3 - Multiplicative
- 4 - Multiplicative Damped
- s6 - Season Smoothing Type - The season smoothing method to be used. The following options are available:
 - -1 - Choose automatically (default)
 - 0 - No trend smoothing
 - 1 - Additive
 - 2 - Multiplicative
- s7 - Alpha - The level smoothing factor. The default value is optimized (0.25 with optimization disabled). Higher values emphasize more recent data, lower values increase the impact of historical data.
- s8 - Beta - The trend smoothing factor. The default value is optimized (0.25 with optimization disabled). Higher values emphasize more recent trend, lower values increase the impact of broader historical trends.
- s9 - Gamma - The seasonal smoothing factor. The default value is optimized (0.25 with optimization disabled). Higher values emphasize more recent seasonal patterns, lower values increase the impact of historical seasonal patterns.
- s10 - Phi - The trend damping factor. The default value is optimized (0.25 with optimization disabled). Lower values decrease the severity of the forecasted trend when using Additive Damped or Multiplicative Damped trend smoothing types.

Output

The Exponential Smoothing functions generate the following outputs:

- Exponential Smoothing - The Exponential Smoothing result set.
- Upper Band Error - The upper error boundary based on standard deviation and the forecasting error.
- Lower Band Error - The lower error boundary based on standard deviation and the forecasting error.
- Alpha - The smoothing factor.

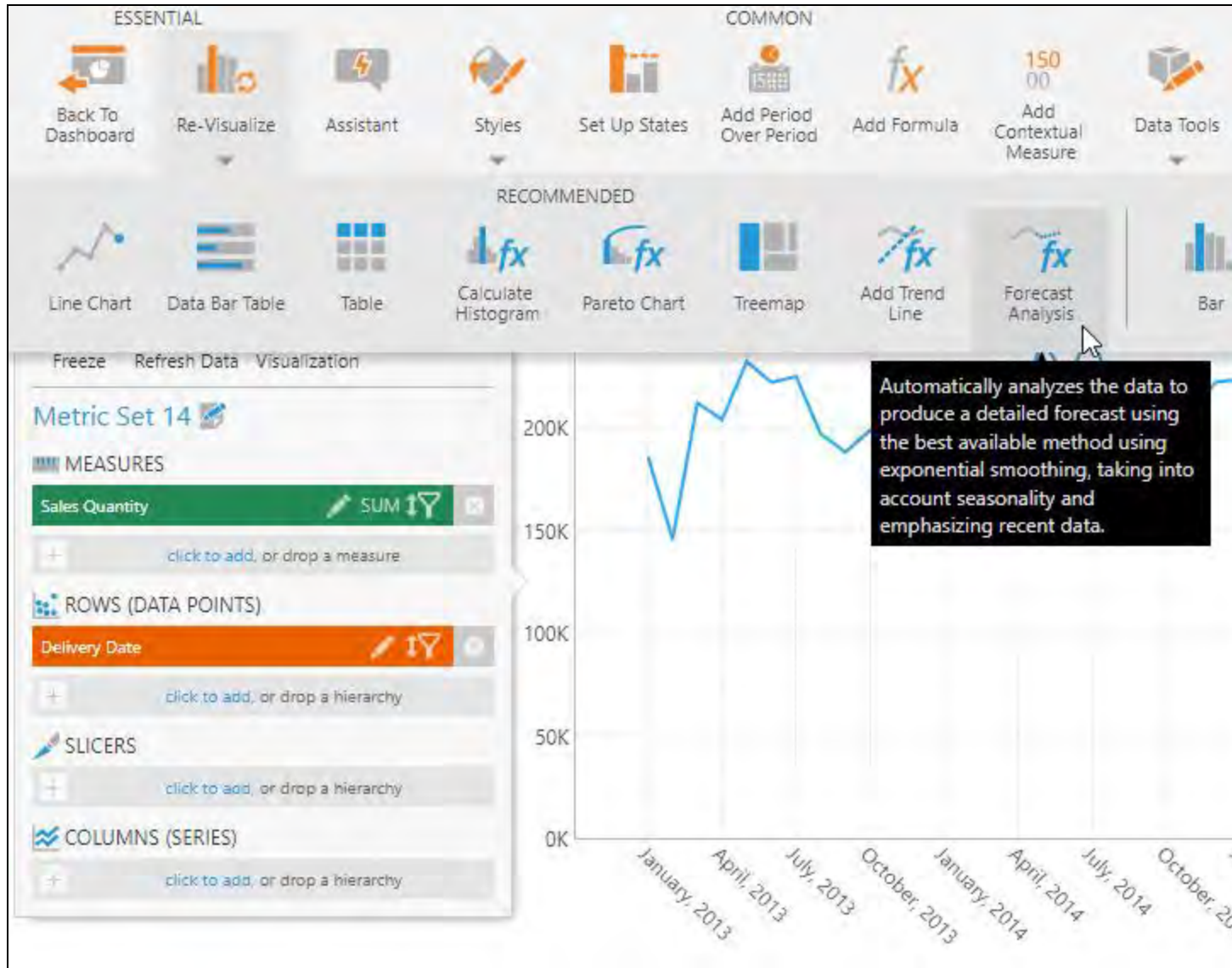


- Archive of documentation for Logi Symphony v24

- Axis - The forecasting axis.
- Beta - The trend smoothing factor.
- Gamma - The seasonal smoothing factor.
- Mean squared error - The average deviation from historical data.
- Phi - The trend damping factor.
- Season Length - The season length.
- Season type - The season smoothing method used.
- Trend type - The trend smoothing method used.

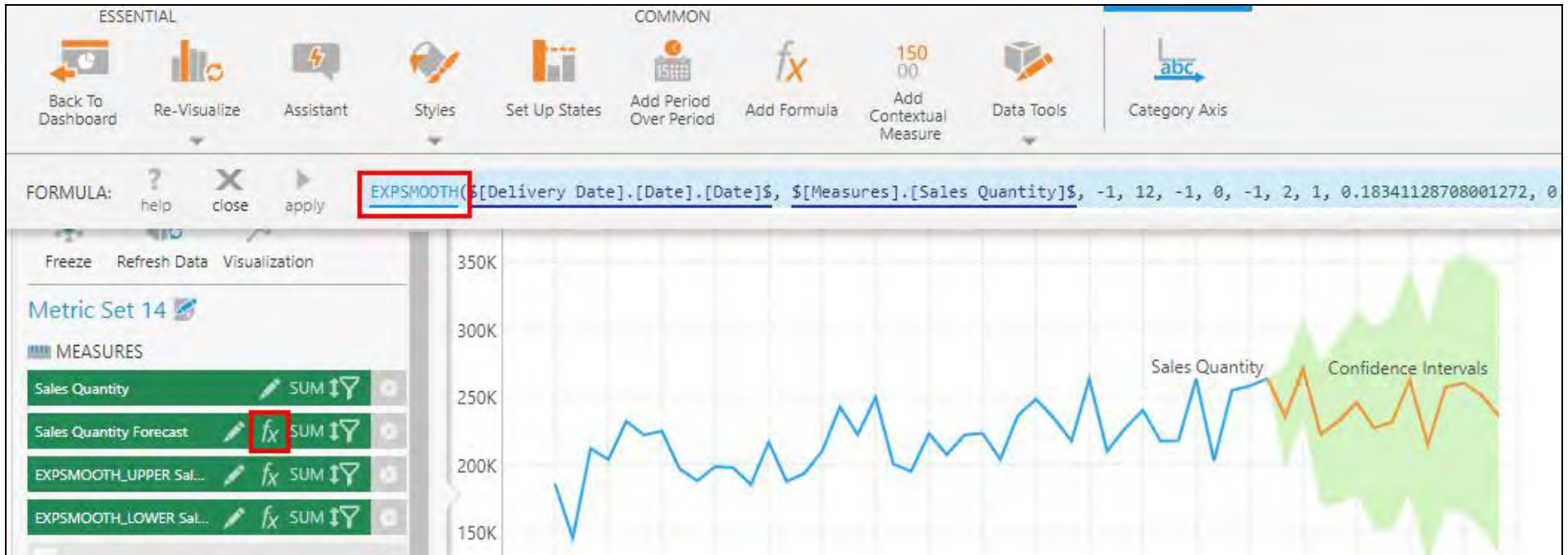
Automatic Forecast Analysis

When displaying a chart with date/time data along the bottom axis, choose **Re-Visualize** in the toolbar and find Forecast Analysis under [Recommended](#).



This option automatically applies the EXPSMOOTH function with default parameters so that your data is analyzed to find the best fitting forecast options, and then the optimal parameters are saved into the formula so that the result is calculated more efficiently next time.

To see this, click the **fx** button on the new forecast measure in the Data Analysis Panel.



To read these parameters more easily, click the EXPSMOOTH function name to open the **Formula** dialog, then click to expand the section for each parameter to see its description and the current value.

 Formula

[more help](#)

EXPSTMOOTH

An exponential smoothing function designed to forecast future data values based on existing patterns, with an emphasis on recent data.

```
EXPSTMOOTH([Delivery Date],[Date],[Date]$, [Measures],[Sales Quantity]$, -1, 12, -1, 0, -1, 2, 1, 0.18341128708001272, 0.35089757584698866, 0.6060869404101886, 0.365058460745421)
```

▶ Parameter 1 - Time Axis Values

▶ Parameter 2 - Base Values

▶ Parameter 3 - Forecast Length

▼ Parameter 4 - Season Length

(Optional) Length of the season in periods over which to find patterns, or -1 for auto.

Parameter Expression

12

▶ Parameter 5 - Confidence Level

▶ Parameter 6 - Show Whole Result

▶ Parameter 7 - Optimizer Passes

▶ Parameter 8 - Trend Smoothing Type

✓

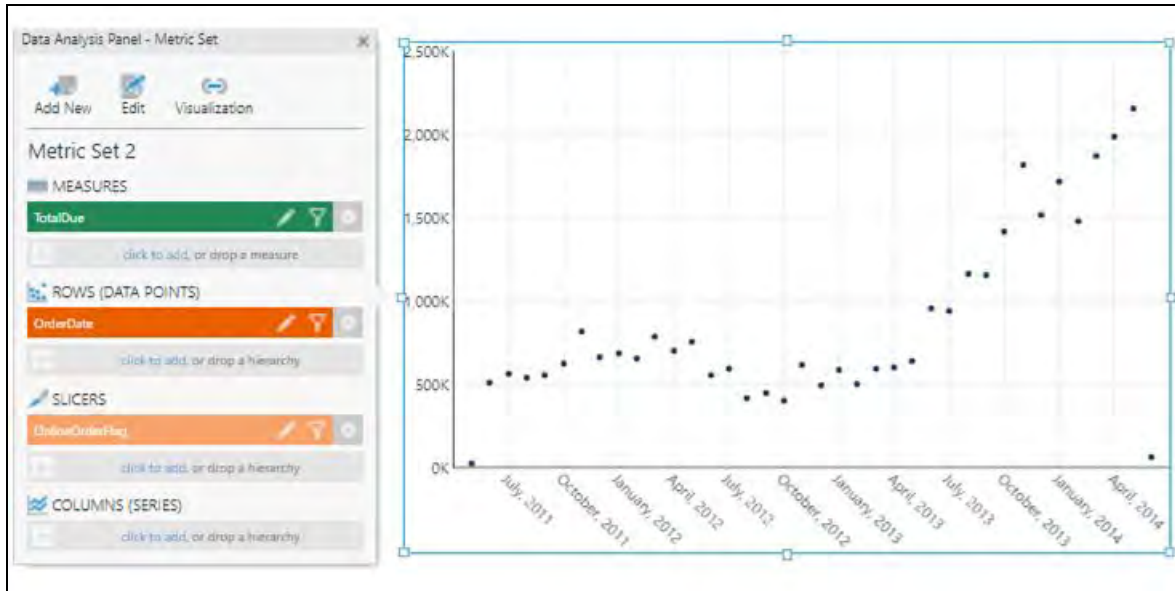
Examples

For the following examples of applying the exponential smoothing functions manually, first create a Point chart.

Add a measure (**TotalDue**), a time hierarchy (**OrderDate**), and a slicer (**OnlineOrderFlag**).

Edit the **OrderDate** hierarchy and change its Level to **Month**.

Filter the slicer to show only **True** values.



Single Exponential Smoothing

For some background information on single exponential smoothing, view [this article](#) from the National Institute of Standards and Technology.

With the chart selected on the canvas, go to the toolbar, click **Data Tools**, and then select **Add Formula**.

In the formula bar, click Visualization and select the line chart, then type the following and click **apply**:

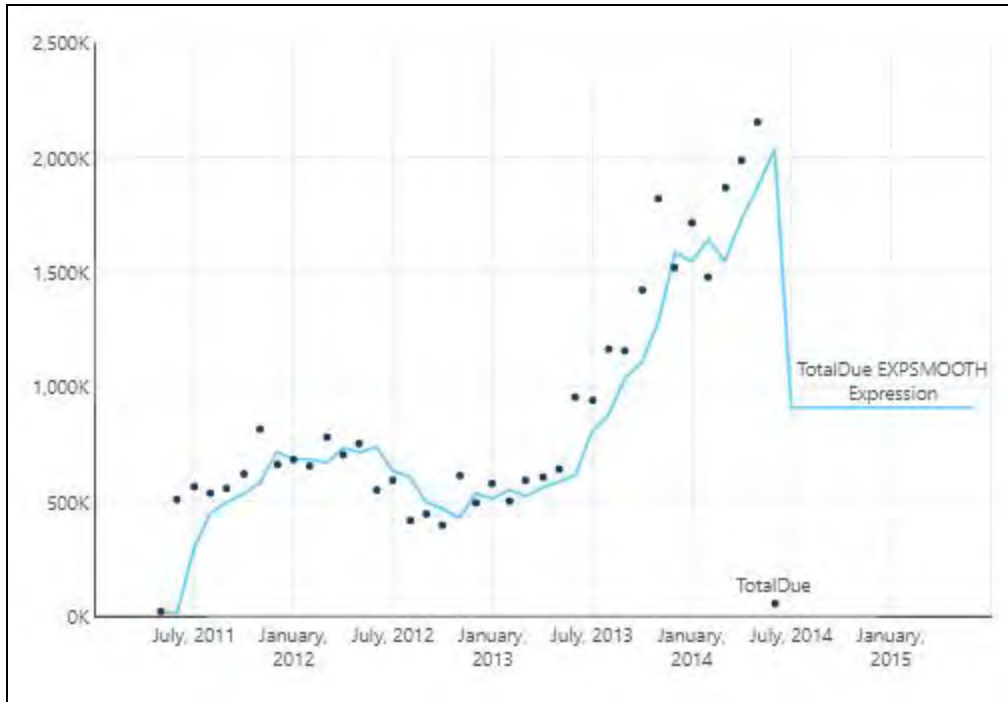
```
EXPSMOOTH($OrderDate$, $TotalDue$, -1, -1, 0.95, 1, 10, 0, 0)
```



Here all the parameters are set to their default values, other than the following three:

- **s3** is set to **1** to show the whole result
- **s5** is set to **0**, which means there is no trend smoothing
- **s6** is set to **0**, which means there is no season smoothing

The result is a forecasting extension of the [Exponential Moving Average](#).



Double Exponential Smoothing

For some background information on double exponential smoothing, view [this article](#) from the National Institute of Standards and Technology.

Open the Data Analysis Panel for the chart and edit the formula for the new measure.

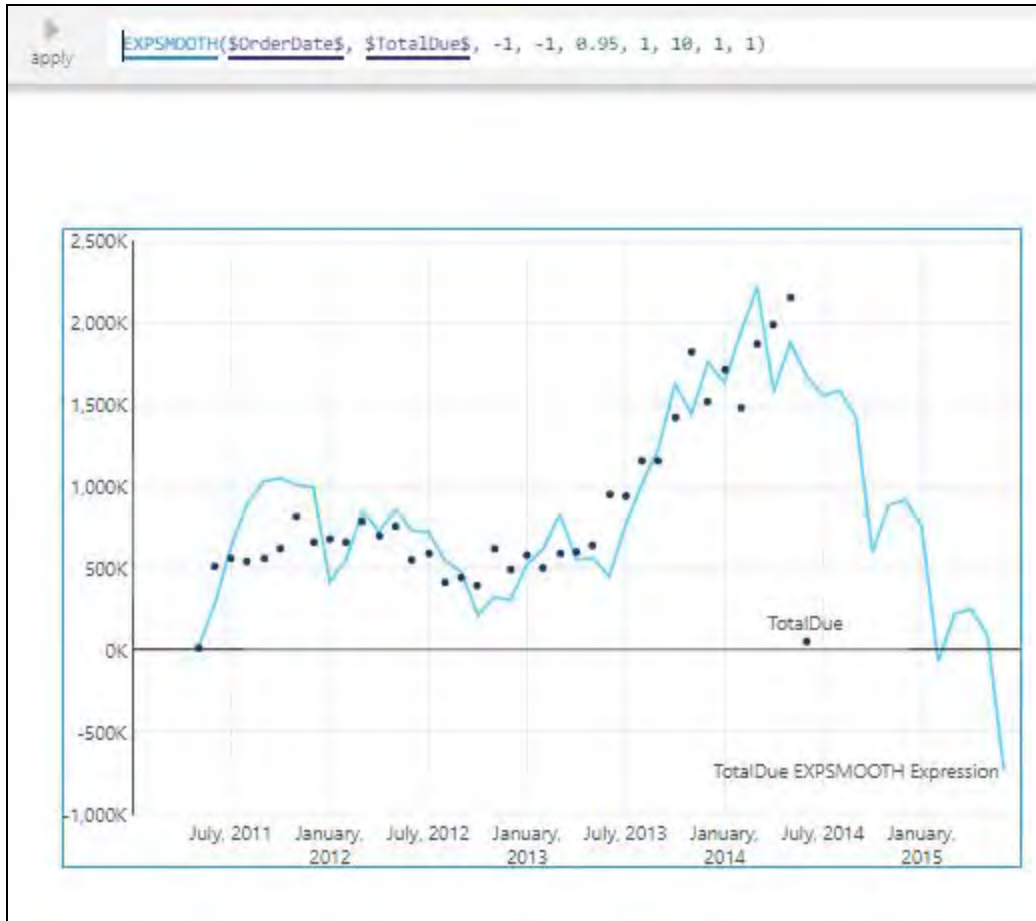
Change the **s5** parameter to **1**, which will add additive trend smoothing to the single exponential smoothing result. Trend smoothing is an exponential moving average of the slope between points.



Triple Exponential Smoothing

For some background information on triple exponential smoothing, view [this article](#) from the National Institute of Standards and Technology.

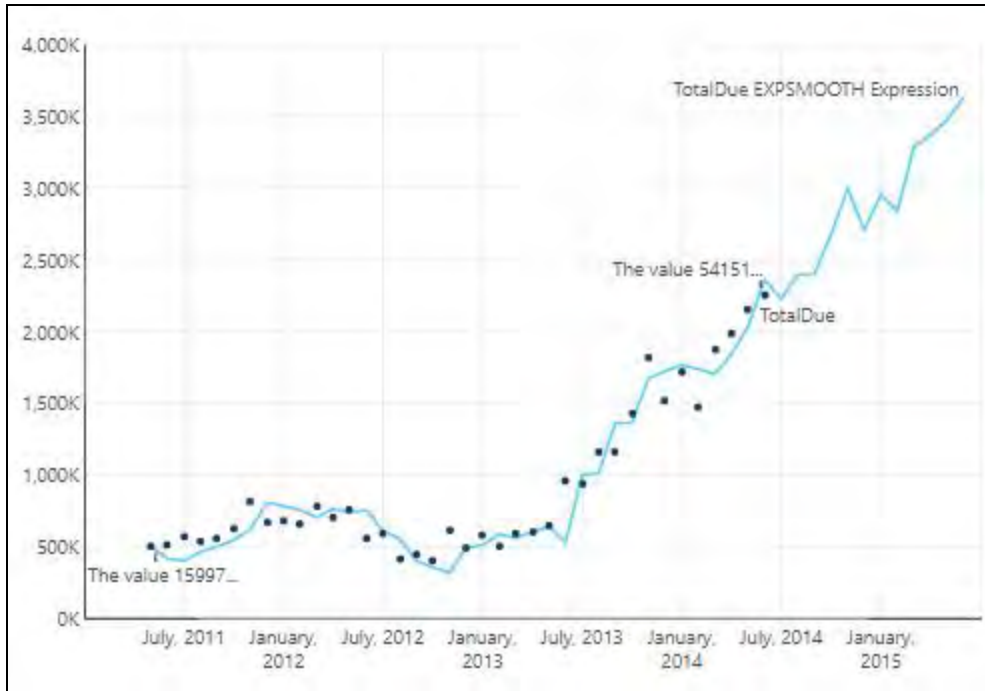
Edit the formula again and change the **s6** parameter to **1**. This will add additive season smoothing to the double exponential smoothing result. Season smoothing is an exponential moving average of the values over repeating seasons.



Notice that the first and last data points are very low and likely throw the forecasting off.

To correct that, switch to view mode, right-click the first data point and select **Correct Value**. Set the new value to **500000**.

Similarly, correct the last data point to **2250000**.



Note: The forecast will only go until the end of the time dimension or until the end of the OLAP time hierarchy.

For more information, see:

- [Using Symphony Recommendations](#)
- [Adding Formulas](#)
- [List of Formulas](#)
- [Exponential Moving Average](#)
- [Trend and Forecasting](#)

Fast Stochastic

This applies to: *Managed Dashboards, Managed Reports*

The **Fast Stochastic** function compares the close price of a stock against its price range (high-low) over a specified number of time periods. Applications of this formula include the generation of buy and sell signals.

Syntax

Fast Stochastic:

```
FASTSTOC (d0,d1,d2,s0,Alignment)
```

Trigger Line:

```
FASTSTOCTRG (d0,d1,d2,s0,Alignment)
```

Input

The Fast Stochastic functions require the following input series:

- d0 - High data values - The first set of data values for which the Fast Stochastic formula is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Fast Stochastic formula is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Fast Stochastic formula is calculated, usually the daily close price of a stock.

Parameters

The Fast Stochastic function has the following parameters:

- s0 - Look Back Period - The number of time periods for determining the overall high and low values. Default value is 14.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.



Output

The FASTSTOC function generates the following output:

- Fast Stochastic - The Fast Stochastic result set.

The FASTSTOCTRG function generates the following output:

- Trigger Line - The trigger line is a 3-period, simple moving average that is used to smooth out the Fast Stochastic values.

First Subset

This applies to: *Managed Dashboards, Managed Reports*

The **First Subset** function returns the value(s) appearing first in an input data series.

When used in a [formula visualization](#), you can specify a count of more than one, for example, return the first 3 values.

Syntax

```
FIRST(d0,s0,Alignment)
```

Input

The First Subset function requires the following input:

- d0 - Input data values - The set of data values for which the First Subset is calculated.

Parameters

The First Subset function accepts the following parameters:

- s0 - First Subset Count (Optional) - The number of values to be returned by the formula. Default value is 1.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis. Specify a hierarchy to return the first input value(s) corresponding to that hierarchy's values: e.g., **FIRST(\$Measure\$, 1, \$Year\$)** will return the first year within each grouping of years under another hierarchy's values such as **Country**.

When you use the FIRST function within a metric set, it can only return a single value (s0 must be 1). In order to return the first 3 values, for example, you must use a formula visualization. This applies to the other subset functions as well.

Output

The First Subset function generates the following output:

- First Subset - The First Subset result set.



Note: When used in a formula visualization, you can effectively [filter the result](#) to the first value(s), optionally [grouped](#) by an alignment hierarchy.

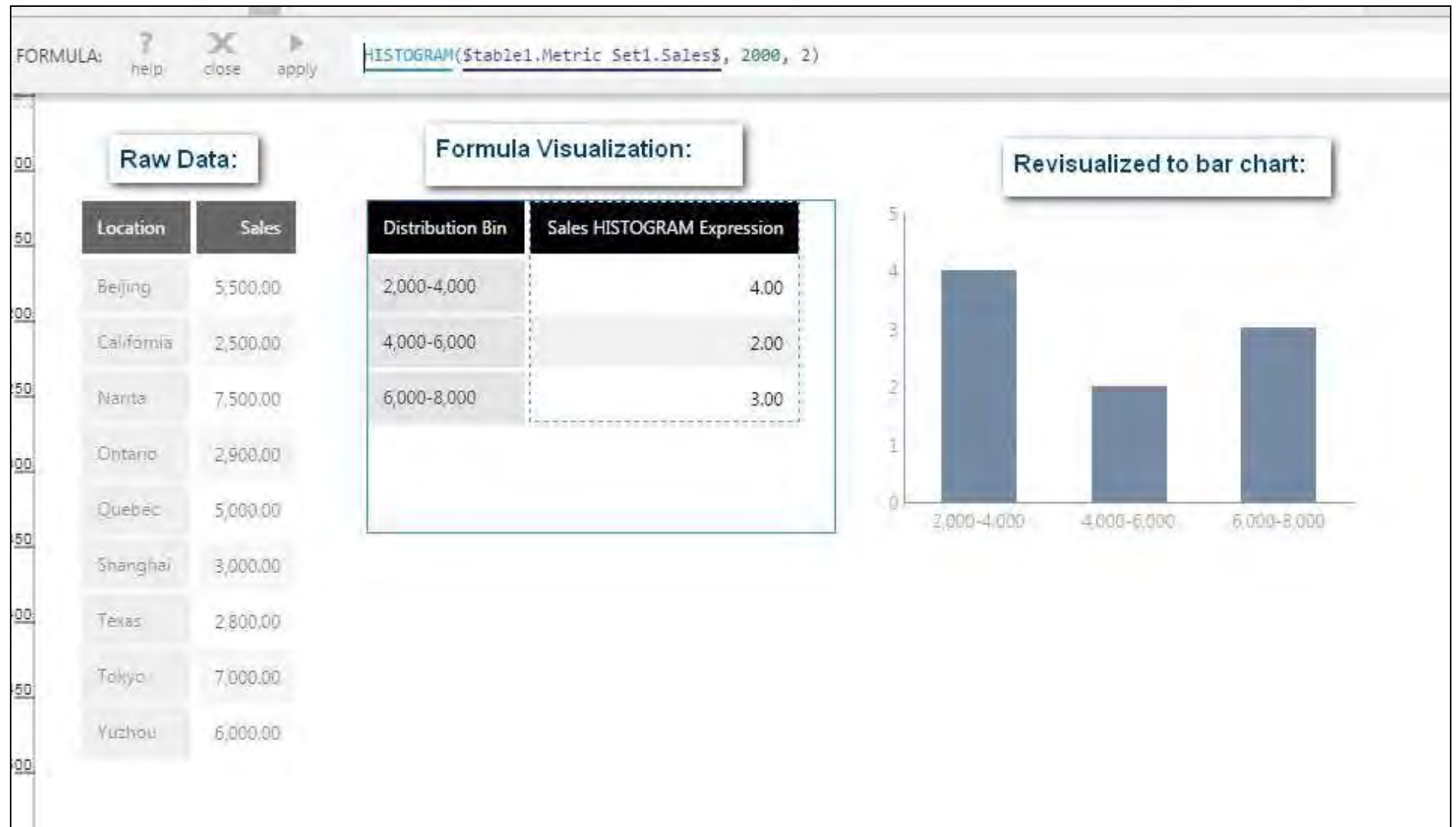
For more information, see:

- [Using a Formula Visualization](#)
- [List of Formulas](#)
- [Adding Formulas](#)

Histogram

This applies to: *Managed Dashboards, Managed Reports*

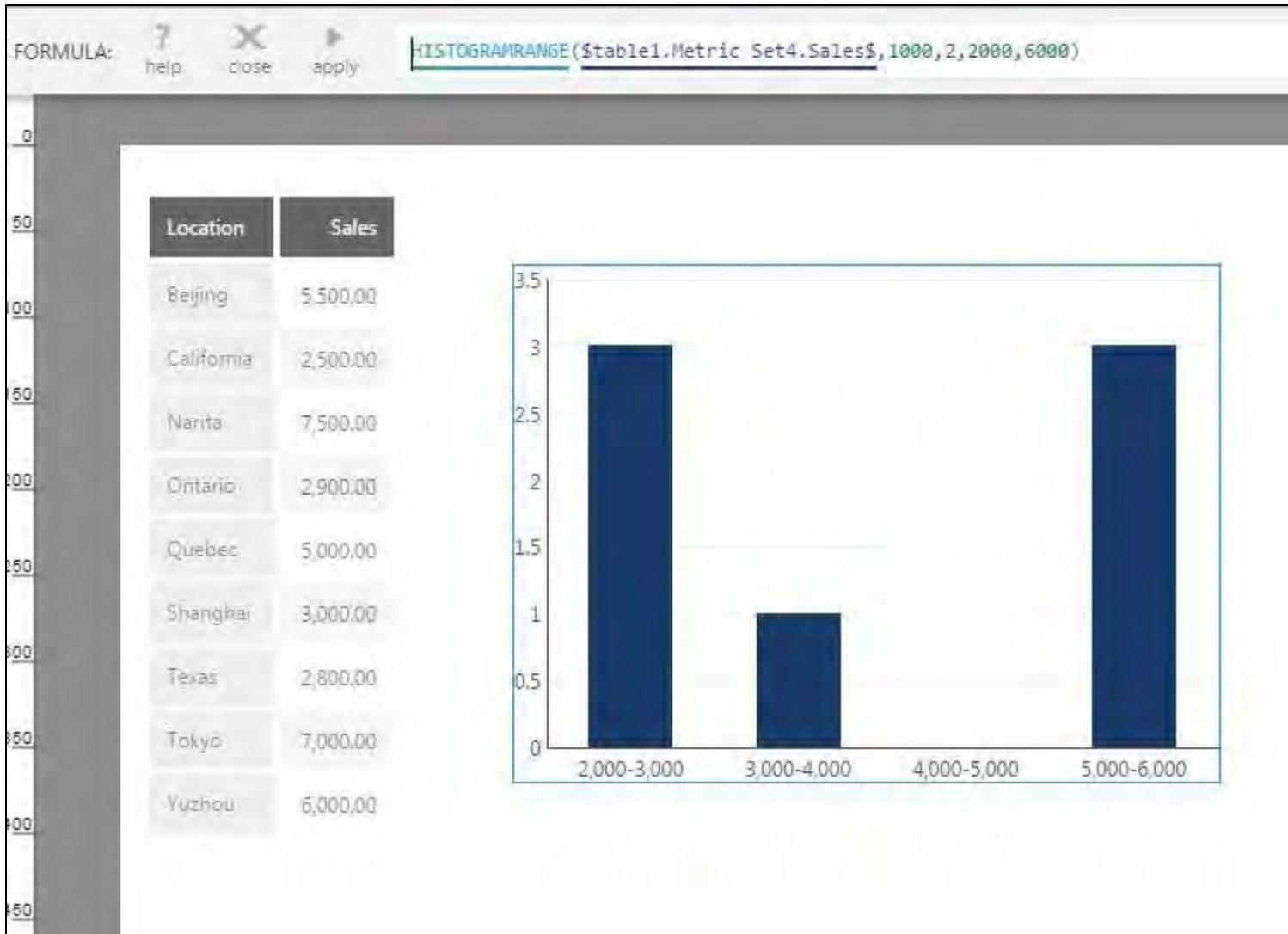
The **Histogram** function assigns numeric values to bins and returns the counts. This analyzes the numeric values by themselves, and can make it possible to visualize how a set of numeric values are distributed, see which are the most or least common, and look for patterns.



Note: You can also re-visualize Calculate Histogram to quickly generate a histogram chart. See [Calculate Histogram](#).

Note: The histogram functions can only be used in a [formula visualization](#).

The **Histogram Range** function can optionally be used instead when you want to specify the start and/or end of the range of values included in the result.





Syntax

Histogram:

```
HISTOGRAM(d0,s0,s1,Alignments...)
```

Histogram Range:

```
HISTOGRAMRANGE(d0,s0,s1,s2,s3,Alignments...)
```

Input

The Histogram function requires the following input:

- d0 - The set of data values for which the histogram is calculated.

Parameters

The Histogram functions use the following parameters:

- s0 - The size of the bins for the numeric values to be assigned to, optional. Enter 0 or null to calculate the bin size automatically.
- s1 - A number representing the type of captions to describe the ranges, optional.
 - 0: The default value is 0 (Auto)
 - 1: ">=0 & <10, >=10 & <20"
 - 2: "0-10, 10-20"
- s2 - (For Histogram Range) The value used to start the range of the first bin. Input values smaller than this will not be counted in the result.
- s3 - (For Histogram Range) The value used to end the range of the last bin. Input values greater or equal to this will not be counted in the result.
- Alignments... (One or More, Optional): Hierarchy placeholders to be used as the alignment axis, for grouping histogram results by the specified values rather than producing a single result.



Output

The histogram functions generate the following output:

- Histogram - The Histogram result set, containing only a **Distribution Bin** hierarchy and a count of the number of measure values from the input data within each bin

Historical Volatility

This applies to: *Managed Dashboards, Managed Reports*

The **Historical Volatility** function calculates the [standard deviation](#) of a set of values over a period. For example, the volatility of stock prices over time.

apply HISTVOL(\$Close\$)

Date Values	Close	Script Measure
08/14/2014 00:00:00	36.36	0.01
08/15/2014 00:00:00	36.47	0.01
08/18/2014 00:00:00	37.38	0.01
08/19/2014 00:00:00	37.83	0.01
08/20/2014 00:00:00	37.50	0.01
08/21/2014 00:00:00	37.64	0.01
08/22/2014 00:00:00	38.01	0.01
08/25/2014 00:00:00	37.71	0.01
08/26/2014 00:00:00	37.79	0.01
08/27/2014 00:00:00	38.18	0.01

Syntax

```
HISTVOL ( d0, Alignment)
```

Input

The Historical Volatility function requires the following input series:

- d0 - The set of data values for which the Historical Volatility is calculated.

Parameters

The Historical Volatility function requires the following parameter:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Historical Volatility function generates the following output:

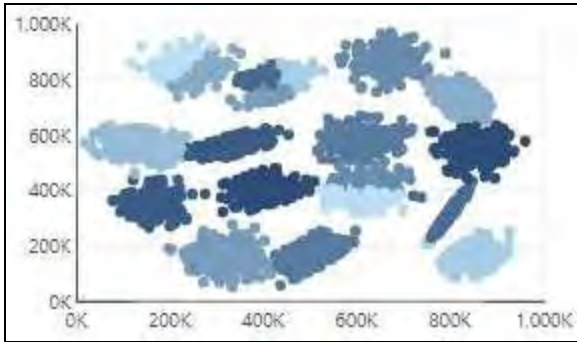
- Historical Volatility - A single value representing the Historical Volatility of the input values.)

K-Means Clustering

This applies to: *Managed Dashboards, Managed Reports*

The **Clustering** function uses the K-Means algorithm to group data points based on similarity of the measures provided. Clustering can help identify different groups in your data that should receive special treatment (for example, a defined custom marketing campaign for a certain cluster).

The K-means clustering model partitions a number (n) of observations into a number (k) of clusters, in which each observation belongs to the cluster with the nearest mean.



Note: Symphony can apply this formula automatically to a scatter plot as one of its [recommendations](#).

For more information about the methods used for clustering, see [k-Means Clustering](#), [Bisecting kMeans](#) and the [Calinski-Harabasz criterion index](#).

Syntax

```
CLUSTERING (d0,d1,d2,...,s0,s1)
```

The same syntax can also be used to return the K value with CLUSTERINGK.

Input

The Clustering function requires at least one input series, but can take any number of input series:

- d0,d1,d2,... - Measures to be used to compute similarity for clustering.



Parameters

- s0 - K - The number of clusters to be created. If no K is provided (or is less than 1), Symphony will attempt to automatically select a fitting value.
- s1 - MaxPasses - The number of passes after which to stop if convergence is not reached. The default is 20.

Output

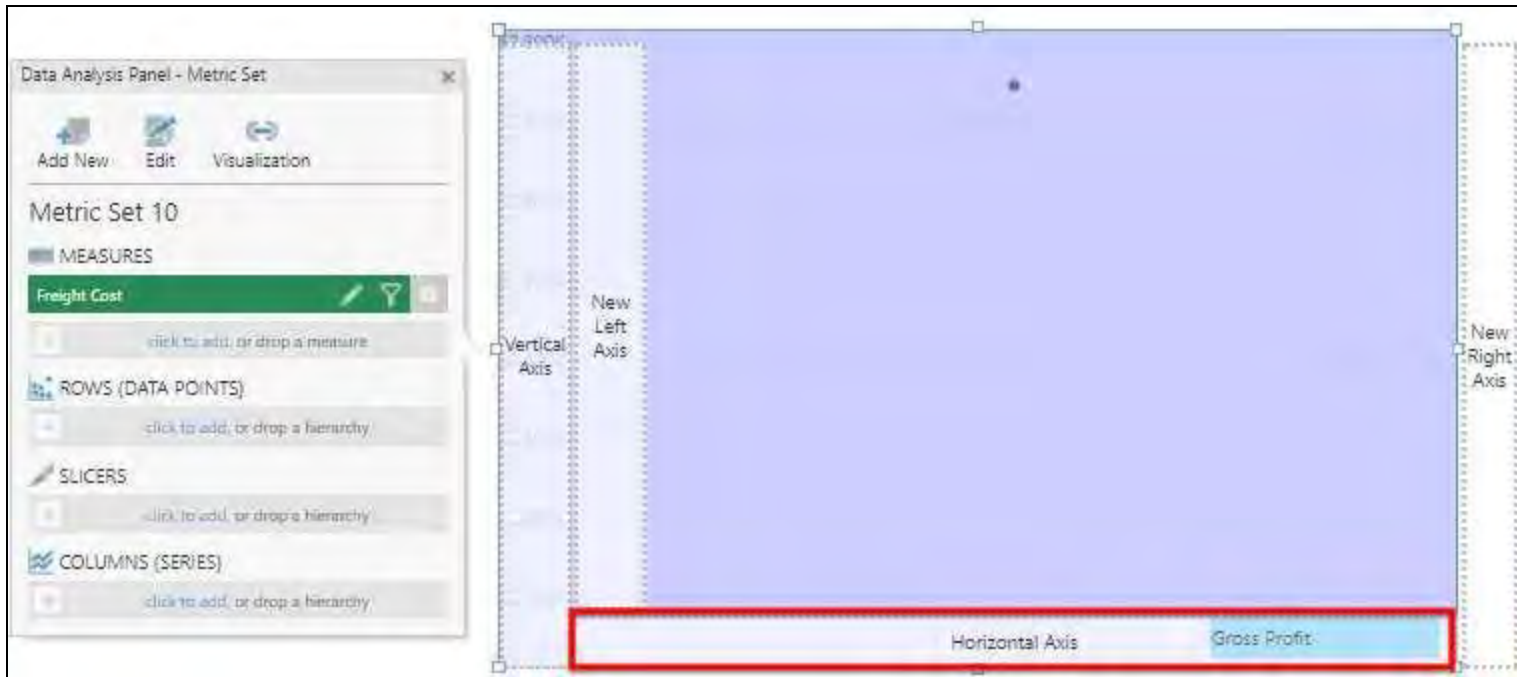
The Clustering function generate the following outputs:

- Clustering - The Clustering result set, where each row is assigned a value between 1 and K.
- K - The K factor.

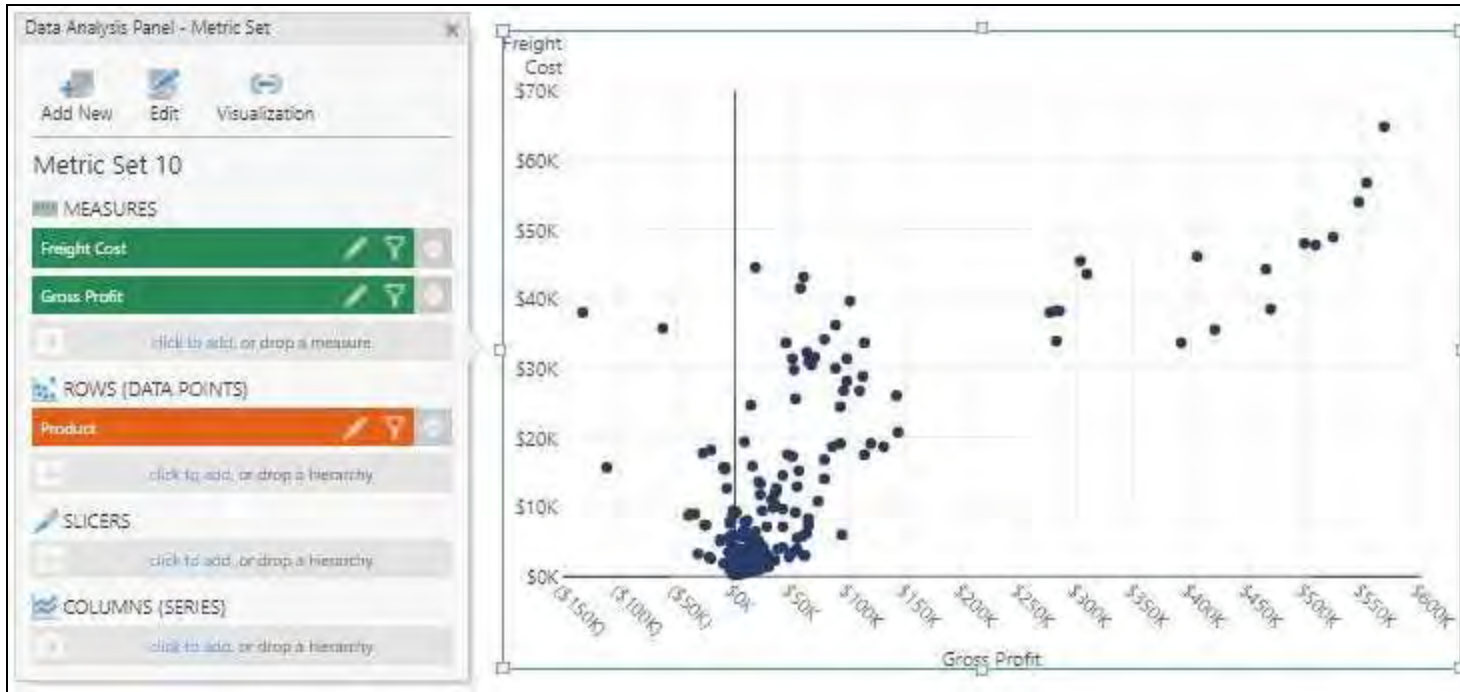
Examples

For the following example of using K-Means Clustering, first create a Point chart.

From the Adventure Works OLAP database, drag the **Freight Cost** measure to the **Measures** field and the **Gross Profit** measure to the **Horizontal Axis** area.



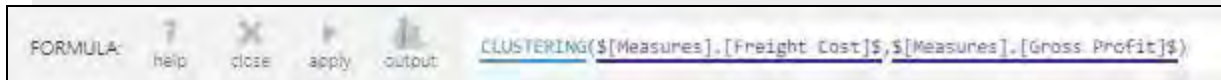
Drag the **Product** dimension to the **Rows** field.



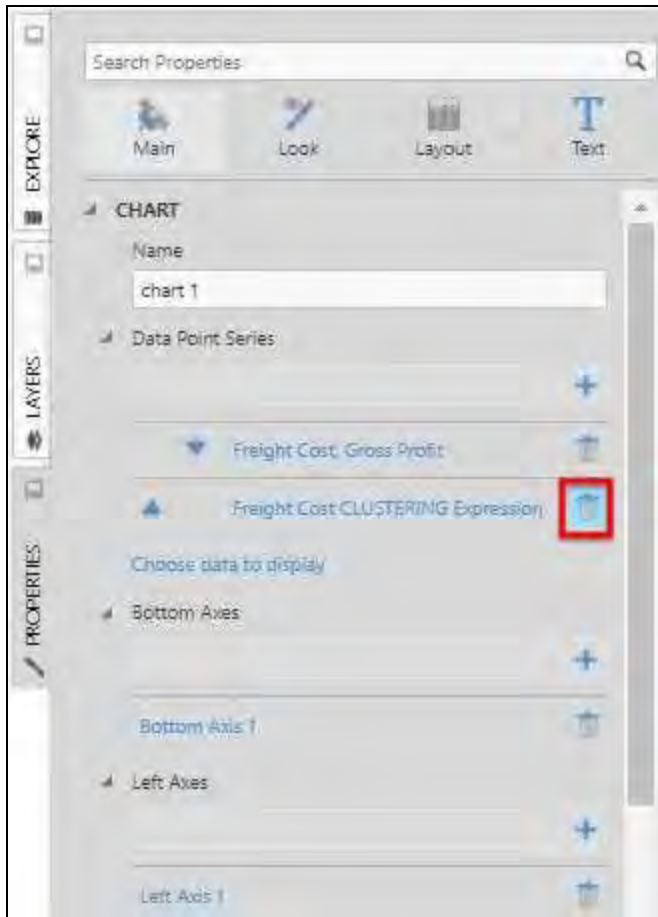
From the toolbar, click **Data Tools** and then select **Add Formula**.

In the formula bar, type the following and click apply:

```
CLUSTERING($[Measures].[Freight Cost],$[Measures].[Gross Profit])
```



Open the Properties panel and remove the **Freight Cost CLUSTERING** Expression series for the visualization.



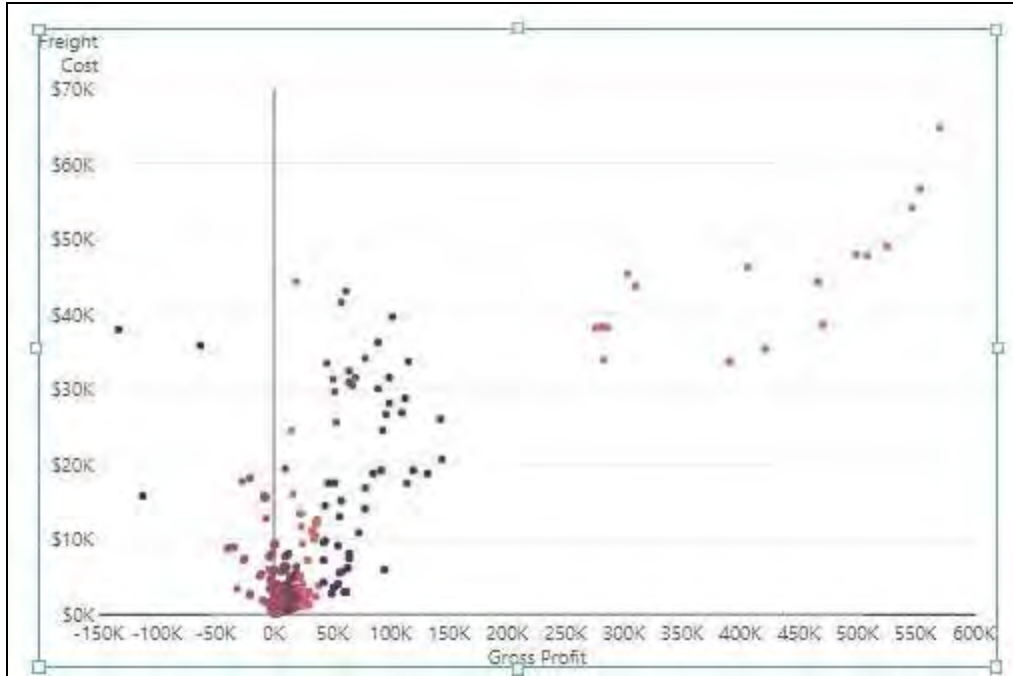
Open the Data Analysis Panel and click **Visualization**.

Click **More...** to see more options for the **Freight Cost, Gross Profit** series.

Under **Color**, click **click to add** and select **Freight Cost CLUSTERING Expression**.



The data in the visualization is colored based on the results of the clustering formula. Click the brush icon next to the Color field to adjust the generated Auto Color Rule.



For more information, see:

- [Using Symphony Recommendations](#)
- [Adding Formulas](#)
- [List of Formulas](#)
- [Clustering predictions using R language analysis](#)



Last Subset

This applies to: *Managed Dashboards, Managed Reports*

The **Last Subset** function returns the value(s) appearing last in an input data series.

When used in a [formula visualization](#), you can specify a count of more than one, for example, to return the last 3 values.

Syntax

```
LAST(d0,s0,Alignment)
```

Input

The Last Subset function requires the following input:

- d0 - Input data values - The set of data values for which the Last Subset is calculated.

Parameters

The Last Subset function accepts the following parameters:

- s0 - Last Subset Count (Optional) - The number of values to be returned by the function. Default value is 1.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis. Specify a hierarchy to return the last input value(s) corresponding to that hierarchy's values: e.g., **LAST(\$Measure\$, 1, \$Year\$)** will return the last year within each grouping of years under another hierarchy's values such as **Country**.

When you use the LAST function within a metric set, it can only return a single value (s0 must be 1). In order to return the last 3 values, for example, you must use a formula visualization. This applies to the other subset functions as well.

Output

The Last Subset function generates the following output:

- Last Subset - The Last Subset result set.

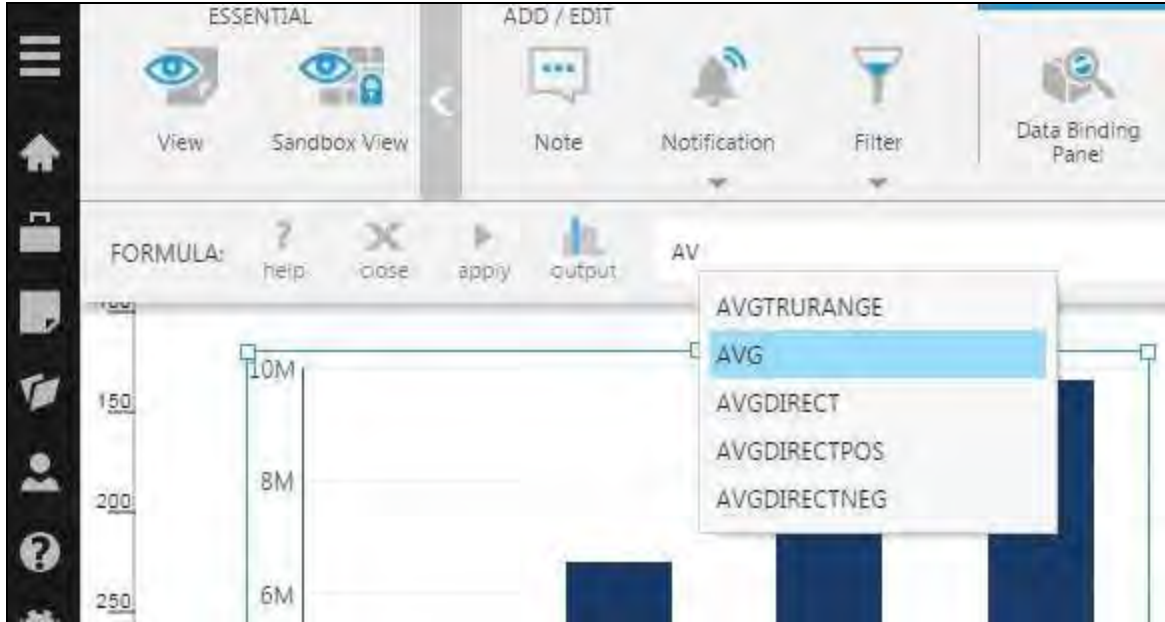


Note: When used in a formula visualization, you can effectively [filter the result](#) to the last value(s), optionally [grouped](#) by an alignment hierarchy.

List of Formulas

This applies to: *Managed Dashboards, Managed Reports*

You can apply mathematical formulas to data by entering small expressions into the formula bar, similar to a spreadsheet program. These formula expressions typically include functions such as AVG and SUM, arithmetic operators or other math functions, or a combination of these. For more complex formulas, you can [write scripts](#).



This article lists the functions that are available for use in formula expressions.

Related video: [Formulas](#)

Standard Functions

These are the most common functions for statistical or financial calculations, such as calculating the mean or generating a moving average. Additional statistical and financial functions can be found in subsequent sections.

Function	Name	Description
AVG	Mean (Simple Average)	The Mean function computes the mean (or simple average) of a set of input values.
BOLLINGERUPPER	Bollinger Bands	The Bollinger Bands functions are based on methods developed by John Bollinger. They compute a pair of

Function	Name	Description
BOLLINGERLOWER		data bands that envelop the moving average of the input value series. Each data band is positioned a specified number of standard deviations based on the moving average of the selected input value series. Applications of Bollinger Bands include the measurement of stock price volatility.
CMLTOTAL CMLTOTALGROUP	Cumulative Total	The Cumulative Total functions compute a running sum or total of a set of input values which can be optionally grouped by specific hierarchies.
CLUSTERING	K-Means Clustering	The Clustering function uses the K-Means algorithm to split data points into clusters based on similarity of the measures provided.
DISTANCE	Distance	The Distance function computes the as-the-crow-flies distance between a source and destination identified by latitude and longitude.
EXPMOAVG	Exponential Moving Average	The Exponential Moving Average function computes the average of a set of input values over a specified number of time periods. In this function, a greater weight is given to more recent data. This function can be used to smooth a data series, which helps to reduce noise and make it easier to spot data trends.
HISTVOL	Historic Volatility	The Historical Volatility function calculates the volatility of a set of input data values, such as stock prices over a period of time.
MEDIAN	Median	The Median function computes the median (e.g., middle) value from a set of input values.
MODE	Mode	The Mode function finds the value that is repeated more often than any other in a value series.
MOAVG	Moving Average	The Moving Average function computes the average of a set of input values over a specified number of time periods. The smaller the number of time periods, the faster the SMA responds to changes in the input values.
MOAVGEVPUPPER MOAVGEVLOWER	Moving Average Envelopes	The Moving Average Envelopes functions compute a pair of data bands that envelop a moving average of the input data values. Each data band is positioned a percentage-based magnitude away from the moving average line. The average line can be calculated using a simple moving average or an exponential moving average. Applications of the Moving Average Envelopes function include indication of overbought/oversold conditions and price trends for a security.
OUTIQ OUTTAU	Data Outliers	The data outlier functions identify the outliers in a set of input values.
PERCENTTOTAL PERCENTTOTALGROUP	Percent of Total	The Percent of Total functions return the percentage of each value in the selected series out of the total sum.
RANK	Rank	The Rank function calculates the rank for each value in a set of input values.
RANKVAL	Rank Value	The Rank Value function returns the value found at the specified position, either starting from the highest or

Function	Name	Description
		the lowest value.
SUM	Totals	The Totals function calculates the sum total for an input value series.
TRENDFORE TRENDFOREUPPER TRENDFORELOWER	Trend and Forecasting	The Trend and Forecasting functions apply a regression function to historical data in order to produce a line of best fit and optionally forecast future values, with optional upper and lower error boundaries.
WGHTMOVAVG	Weighted Moving Average	The Weighted Moving Average function computes the average of a set of input values over a specified number of time periods. In this function, a greater weight is given to more recent data. The function can be used to smooth a data series, which helps to reduce noise and make it easier to spot data trends.

Statistical Functions

Use statistical functions to perform statistical analysis on your data.

Function	Name	Description
CORRELATION CORRELATIONGROUP	Correlation	The Correlation functions calculate the correlation between two input data series. It measures the strength of the relationship between the two series. The resulting correlation value ranges between -1 and 1. A correlation value of zero implies that the two series are independent and have no relationship. A correlation value that is close to 1 implies that the two series are positively correlated. Similarly, a correlation value that is close to -1 implies that the two series are negatively correlated. If the correlation value is negative, this means that when the first input is large, the second input is small (and vice-versa).
CORRMATRIX CORRMATRIXCOLUMN CORRMATRIXROW	Correlation Matrix	The Correlation Matrix function calculates the strength of the relationships between all possible pairings of the specified data series as a value between -1 and 1. Zero implies that the two series are independent; a value close to 1 implies they are positively correlated, and a value close to -1 implies they are negatively correlated. A negative correlation means that when the first input is large, the second input is small (and vice-versa).
FORECASTING EXPSMOOTH	Exponential Smoothing	The Exponential Smoothing functions apply an exponentially-decreasing weight to historical data in order to forecast future values based on emerging trends.
HISTOGRAM HISTOGRAM RANGE	Histogram	The Histogram functions assign numeric values to bins and returns the counts.
NORMALDIST	Normal Distribution	The Normal Distribution functions calculate the standard normal cumulative distribution for an input data

Function	Name	Description
NORMALDISTGROUP		series. Specifically, the function computes the probability that a series, which has a mean of 0 and a standard deviation of 1, contains a value less than the input value.
PERCENTILE	Percentile	The Percentile function returns a scalar value below which the specified percentage of input data values fall.
PERCENTRANK PERCENTRANKGROUP	Percent Rank	The Percent Rank functions calculate the percent rank for each value in a set of input values. Specifically, the function computes the percentage of all input values that are less than a given input value. Percent Rank values range between 0 and 1.
STDEV	Sample Standard Deviation	The Sample Standard Deviation function measures volatility as the difference between values and their moving average. A larger difference implies a larger standard deviation value.
STDEVPOP	Population Standard Deviation	The Population Standard Deviation function measures volatility as the difference between values and their average. A larger difference implies a larger standard deviation value.
VARIANCE	Variance	The Variance function calculates the variance of a set of input values. It measures how spread out a set of values are from each other. Variance is defined as the average of the squared differences of each value from the mean of the input values.

Subset Functions

Subset functions let you return a subset of your input data based on various criteria.

Function	Name	Description
BOTTOM	Minimum Subset	The Minimum Subset function returns a specified number of the smallest values in an input data series. For example, the function can be used to return the 3 smallest values.
FIRST	First Subset	The First Subset function returns a specified number of values that appear first in an input data series. For example, the function can be used to return the first 3 values.
LAST	Last Subset	The Last Subset function returns a specified number of values which appear last in an input data series. For example, the function can be used to return the last 3 values.
MIDDLE	Middle Subset	The Middle Subset function returns a specified number of the middle values in an input data series.

Function	Name	Description
		For example, the function can be used to return the 3 middle values in a set of input values.
TOP	Maximum Subset	The Maximum Subset function returns a specified number of the largest values in an input data series. For example, the function can be used to return the 3 largest values.
ZIGZAG	Zig Zag Subset	The Zig Zag Subset function returns the input data values that show movement greater than the specified sensitivity percentage value.

Financial Functions

Financial functions are used in financial applications such as technical analysis. Generally, these functions accept the price and/or volume of a security (e.g., a stock) as input and generate a result set that may help you to confirm price trends, spot reversals, or determine when to buy or sell.

Function	Name	Description
ACCDIST	Accumulation Distribution	The Accumulation Distribution function is a technical indicator that considers both stock price and volume. When the indicator increases, it implies an accumulation or buying of the related stock. When the indicator decreases, it implies distribution or selling of the related stock. Accumulation Distribution can be used to detect divergences between stock price movement and volume movement. Other applications include confirmation of price trends and providing advance warning of a possible reversal in stock price.
AROONOSC AROONOSCUP AROONOSCDOWN	Aroon Oscillator	The Aroon Oscillator function is calculated as the difference between the Aroon Up and Aroon Down indicators, with resulting values ranging from -100 to 100. An uptrend in stock price is indicated when the Aroon Oscillator value is positive. A downtrend is indicated when the oscillator value is negative. The further away the Aroon Oscillator value is from the zero line, the stronger the trend. Applications of the Aroon Oscillator include determining whether a stock is trending or not, and gauging the strength of the current trend.
AVGDIRECT AVGDIRECTPOS AVGDIRECTNEG	Average Directional Index	The Average Directional Index was developed by J. Welles Wilder. It combines two other indicators, the Positive Directional Indicator and the Negative Directional Indicator, which were also developed by Wilder. Applications include indicating the strength of the current trend in stock price (but not the trend direction).
AVGTRURANGE	Average True Range	The Average True Range function computes an average of the True Range, which represents the range in

Function	Name	Description
		price of a stock or commodity over the course of one day, or from one day to the next. Average True Range is used to indicate price volatility.
CHKOSC	Chaikin Oscillator	The Chaikin Oscillator was developed by Marc Chaikin. It is calculated by subtracting the 10-day exponential moving average from the 3-day exponential moving average of the Accumulation Distribution line. Resulting values oscillate around the zero level. Applications of Chaikin Oscillator include confirmation of price trends and providing advance warning of price reversals.
CHKMONEYFLO	Chaikin Money Flow	The Chaikin Money Flow was developed by Marc Chaikin. It computes the sum of the Accumulation Distribution line values for a stock over a specified number of time periods, and divides this by the total stock volume over the same duration. When the CMF value is above zero, this indicates buying pressure. When the CMF value is below zero, this indicates selling pressure. Applications of Chaikin Money Flow include determining if a stock is under accumulation or distribution. For example, if the CMF value remains above zero for an extended period of time, this indicates accumulation.
COMMCHANNEL	Commodity Channel Index	The Commodity Channel Index was developed by Donald Lambert. It compares a stock price to its average over a specified number of time periods. Applications of CCI include identification of overbought or oversold conditions, and signaling when a price trend may be weakening.
FASTSTOC FASTSTOCTRG	Fast Stochastic	The Fast Stochastic function compares the close price of a stock against its price range (high-low) over a specified number of time periods. Applications of this function include the generation of buy and sell signals.
MOMENTUM	Momentum	Momentum is a trend-following indicator that measures the speed of price movements by comparing the current stock price against its value from a specified number of time periods in the past. A stock price has upward momentum when the indicator value is positive, and downward momentum when the indicator value is negative. Applications of Momentum include indication of possible trend reversals.
MONEYFLO	Money Flow Index	The Money Flow Index function computes a relationship between the typical stock price and stock volume over a specified number of time periods. Resulting values range between 0 and 100. MFI is similar to the Relative Strength Indicator except that it incorporates volume.

Function	Name	Description
		Applications of Money Flow Index include identifying market tops and market bottoms, and indicating price reversals.
MOVAVGCD MOVAVGCD SIG	Moving Average Convergence-Divergence	Moving Average Convergence-Divergence was developed by Gerald Appel. It computes the difference between a short-term and a long-term exponential moving average of the stock price. Applications of MACO include generation of buy and sell signals. Signals are generated when the MACO line crosses the zero line or its signal line.
NEGVOLUME	Negative Volume Index	The Negative Volume Index function calculates a cumulative value based on days where the trading volume has decreased from the previous day. On days where the volume remained the same or increased, the NVI remains unchanged. Applications of NVI include indication of market trends. For example, a bull market may be indicated when the NVI rises above its one-year moving average.
ONBALVOLUME	On Balance Volume	The On Balance Volume was developed by Joseph Granville. It calculates a cumulative volume that depends on stock price. When the closing stock price for the day is greater than the previous day's close, the volume for the day is added to the cumulative result. If the stock price decreases, volume is subtracted from the result. Applications of On Balance Volume include confirmation of price movements.
ONBALVOLWGHT	On Balance Weighted Volume	The On Balance Weighted Volume function calculates a cumulative volume that depends on stock price. The daily volume is weighted by the difference in stock price between the day's close and the previous day's close before it is added to the cumulative result. Applications of On Balance Weighted Volume include confirmation of price movements and providing indication of possible trend reversals.
PARABOLSAR	Parabolic SAR	Parabolic SAR (stop-and-reverse) was developed by J. Welles Wilder, Jr. It is a lagging technical indicator that can help you to determine when a stock is set to have a change in trend. Applications of Parabolic SAR include generation of buy-sell signals and, in particular, determining when to exit from trades.
POSVOLUME	Positive Volume Index	The Positive Volume Index function calculates a cumulative value based on days where the trading volume has increased from the previous day. On days where the volume remained the same or decreased, the PVI remains unchanged.

Function	Name	Description
		Applications of PVI include indication of market trends. For example, a bear market may be indicated when the PVI drops below its one-year moving average.
RATECHANGE	Rate of Change	The Rate of Change function is a variation on the Momentum function. ROC calculates the percentage difference between the day's closing stock price and the price from a specified number of time periods in the past. Applications of ROC include indication of price reversals. For example, if the price is trending upward while the ROC is moving downward, this may indicate a large drop in price is forthcoming.
RELSTRENGTH	Relative Strength Indicator	The Relative Strength Indicator was developed by J. Welles Wilder. It compares the average of up closes against the average of down closes over a specified number of time periods. Resulting values range from 0 to 100. Applications of RSI include identifying overbought or oversold situations. For example, if the RSI line stalls at a level above 70, this may indicate the stock is overbought. Or, if the RSI line drops to a level below 30, this may indicate the stock is oversold and becoming undervalued.
SLOSTOC SLOSTOCTRG	Slow Stochastic	The Slow Stochastic function compares the close price of a stock against its price range (high-low) over a specified number of time periods. It is derived by applying a 3-period, simple moving average to the Fast Stochastic line. Applications of Slow Stochastic include the generation of buy and sell signals.
STOCOSC STOCOSCTRG	Stochastic Oscillator	The Stochastic Oscillator function compares the close price of a stock against its price range (high-low) over a specified number of time periods. Applications of Stochastic Oscillator include the generation of buy and sell signals.
TRENDCONF	Trend Confirmation Indicator	The Trend Confirmation Indicator function computes the ratio between a short-term moving average and a longer-term moving average of the stock price. Resulting values oscillate around the 100 level. Applications of TCI include generation and confirmation of buy and sell signals. For example, when the TCI advances above 100, this may be interpreted as a buy-signal.
ULTIMOSC	Ultimate Oscillator	The Ultimate Oscillator was developed by Larry Williams. It computes a weighted total of three oscillators, each of which is calculated using a different time period. Resulting values range from 0 to 100. Applications of Ultimate Oscillator include buy and sell signals, and providing indication of overbought or

Function	Name	Description
		oversold conditions.
VOLUMEOSC VOLUMEOSCSIG VOLUMEOSCHIST	Volume Oscillator	The Volume Oscillator function computes the relationship between a short-term moving average and a long-term moving average of the stock volume. Resulting values oscillate around the zero line. Applications of Volume Oscillator include indication of volume surges, which may be due to panic selling or greedy buying.
WILLIAMSR	Williams%R	Williams %R (or Percent Range) was developed by Larry Williams. It is similar to the Stochastic Oscillator function in that it relates the close price of a stock to its price range (high-low) over a specified number of time periods. Applications of Williams %R include indication of price reversals and determining overbought or oversold conditions.

Math Functions

In addition to calling the formula functions listed above, you can use arithmetic operators and math functions like the following.

While formula functions listed above work with the entire series of input values you pass to them, arithmetic operators and math functions operate on each row's values separately. For example, \$Measure 1\$ / \$Measure 2\$ divides each row's Measure 1 value by its Measure 2 value,

while SUM(\$Measure 1\$) takes an entire set of measure values and outputs a single sum. The math function call Math.Round(\$Measure 1\$) takes each row's measure value and rounds it to the nearest whole number.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
()	Specify order of operations
%	Calculate the remainder after dividing

The following are some of the .NET [Math fields](#) and [methods](#) you can access (use the links for the complete list and details):

Field	Description
Math.E	Represents the natural logarithmic base, specified by the constant, e .

Field	Description
Math.PI	Represents the ratio of the circumference of a circle to its diameter, specified by the constant, π .

Method	Description
Math.Abs	Returns the absolute value of a decimal or a number.
Math.Acos	Returns the angle whose cosine is the specified number.
Math.Asin	Returns the angle whose sine is the specified number.
Math.Atan2	Returns the angle whose tangent is the quotient of two specified numbers.
Math.Ceiling	Returns the smallest integral value that is greater than or equal to the specified number.
Math.Cos	Returns the cosine of the specified angle.
Math.Cosh	Returns the hyperbolic cosine of the specified angle.
Math.Exp	Returns e raised to the specified power.
Math.Floor	Returns the largest integer less than or equal to the specified number.
Math.Log	Returns the natural logarithm of the specified number.
Math.Log10	Returns the logarithm of the specified number.
Math.Max	Returns the larger of two numbers.
Math.Min	Returns the smaller of two numbers.
Math.Pow	Returns the specified number raised to the specified power.
Math.Round	Rounds a decimal or double-precision value to the nearest integer or to the specified number of decimal places.
Math.Sign	Returns an integer that indicates the sign of a number.
Math.Sin	Returns the sine of the specified angle.
Math.Sinh	Returns the hyperbolic sine of the specified angle.
Math.Sqrt	Returns the square root of the specified number.
Math.Tan	Returns the tangent of the specified angle.
Math.Tanh	Returns the hyperbolic tangent of the specified angle.
Math.Truncate	Returns the integral part of the specified number.

DundasScript

Formulas run as DundasScript, so you have access to additional operators such as if/else statements, and other functions and types. It can be easier to write more complex formula scripts in advanced mode, described in detail in [Adding Formulas](#).



- Archive of documentation for Logi Symphony v24

For details about script syntax and accessible classes and their methods, see [Writing Data Scripts with DundasScript](#).

Maximum Subset

This applies to: *Managed Dashboards, Managed Reports*

The **Maximum Subset** function returns the largest or top value(s) in an input data series.

When used in a [formula visualization](#), you can specify a count of more than one, for example, to return the 3 largest values.

Syntax

```
TOP(d0,s0,Alignment)
```

Input

The Maximum Subset function requires the following input:

- d0 - Input data values - The set of data values for which the Maximum Subset is calculated.

Parameters

The Maximum Subset function accepts the following parameters:

- Maximum Subset Count (Optional) - The number of values to be returned by the function. Default value is 1.
- Alignment (Optional) -A hierarchy placeholder to be used as the alignment axis. Specify a hierarchy to return the top input value(s) corresponding to that hierarchy's values: e.g., **TOP(\$Measure\$, 1, \$Year\$)** will return the year with the largest value within each grouping of years under another hierarchy's values such as Country. If an alignment hierarchy is not indicated, the subset is applied to the whole result.

When you use the TOP function within a metric set, it can only return a single value (s0 must be 1). In order to return the top 3 values, for example, you must use a formula visualization. This applies to the other subset functions as well.

Output

The Maximum Subset formula generates the following output:

- Maximum Subset - The Maximum Subset result set.



Note: When used in a formula visualization, you can effectively filter the [result](#) to the topmost value(s), optionally [grouped](#) by an alignment hierarchy.

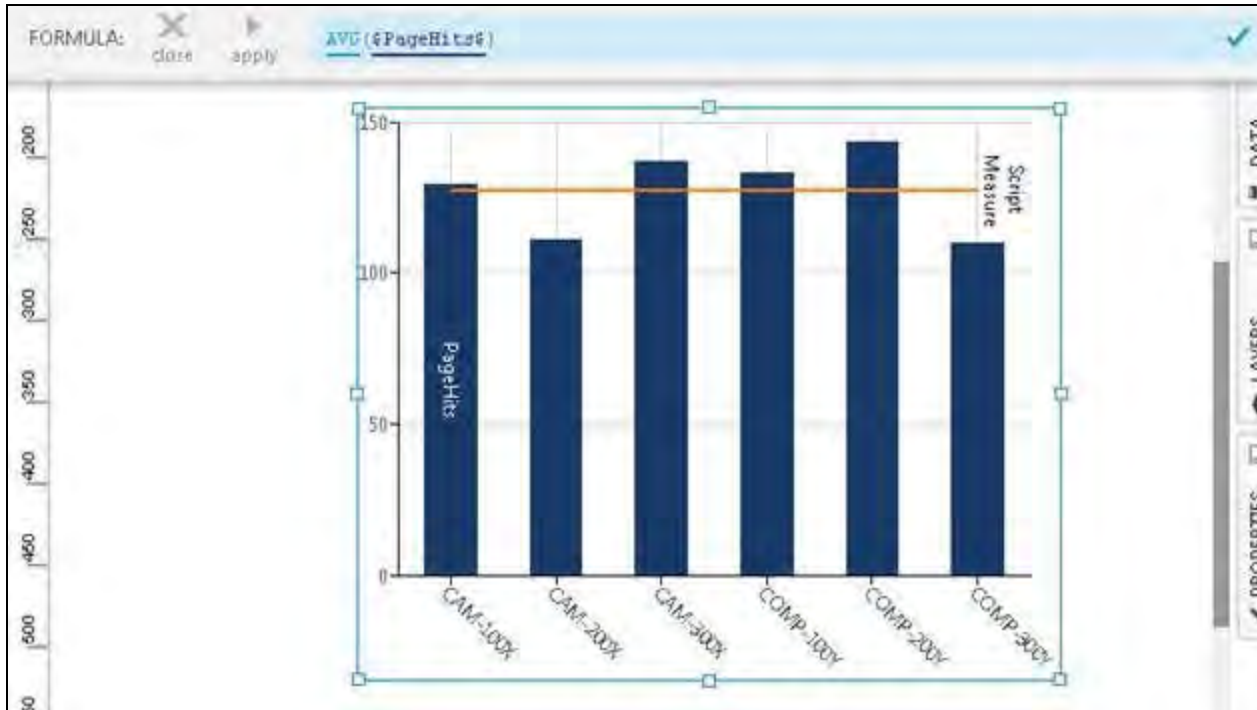
For more information, see:

- [Adding Formulas](#)
- [List of Formulas](#)
- [Using a Formula Visualization](#)

Mean (Simple Average)

This applies to: *Managed Dashboards, Managed Reports*

The **Average** function computes the mean (or simple average) of a set of input values by calculating the **total (sum)** and dividing it by the number of values.



Syntax

```
AVG(d0,Alignments...)
```

Input

The Average function requires the following input:

- d0 - The set of data values for which the mean is calculated.

Parameters

The Average function accepts the following parameters:

- Alignments... (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis. Input values are grouped by the specified hierarchy values, and averages calculated for each group: e.g., `AVG($Quantity$, $Country$)` calculates the average quantity for each country.



Note: When using alignment parameters in a [formula visualization](#), you can choose to display these median values grouped by only the alignment hierarchy values, for example to display just the median for each country without the original input values.

Output

The Average function generates the following output:

- A single value representing the mean of the input data values (or for each group when using alignment parameters).

For more information, see:

- [Adding Formulas](#)
- [List of Formulas](#)
- [Using a Formula Visualization](#)

Median

This applies to: *Managed Dashboards, Managed Reports*

The **Median** function computes the middle value from a set of input values. If the number of values is even, the function computes the average of the two middle values.



Syntax

```
MEDIAN( d0,Alignments ... )
```

Input

The Median function requires the following input:

- d0 - The set of data values for which the Median is calculated.

Parameters

The Median function accepts the following parameters:

- Alignments... (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis. Input values are grouped by the specified hierarchy values, and the median calculated for each group: e.g., `MEDIAN($Quantity$, $Country$)` calculates the median quantity for each country.



Note: When using alignment parameters in a [formula visualization](#), you can choose to display these median values grouped by only the alignment hierarchy values, for example to display just the median for each country without the original input values.

Output

The Median function generates the following output:

- Median - A single value representing the median (or for each group when using alignment parameters).

Middle Subset

This applies to: *Managed Dashboards, Managed Reports*

The **Middle Subset** function returns the middle value(s) in an input data series.

When used in a [formula visualization](#), you can specify a count of more than one, for example, to return the 3 values found in the middle of a series.

Syntax

```
MIDDLE(d0,s0,Alignment)
```

Input

The Middle Subset function requires the following input:

- d0 - Input data values - The set of data values for which the Middle Subset is calculated.

Parameters

The Middle Subset function accepts the following parameters:

- s0 - Middle Subset Count (Optional) - The number of values to be returned by the function. Default value is 1.
- Alignment (Optional) -A hierarchy placeholder to be used as the alignment axis. Specify a hierarchy to return the middle input value(s) corresponding to that hierarchy's values: e.g., **MIDDLE(\$Measure\$, 1, \$Year\$)** will return the year in the middle of each grouping of years under another hierarchy's values such as **Country**. If an alignment hierarchy is not indicated, the subset is applied to the whole result.

When you use the MIDDLE function within a metric set, it can only return a single value (s0 must be 1). In order to return the middle 3 values, for example, you must use a formula visualization. This applies to the other subset functions as well.

Output

The Middle Subset function generates the following output:

- Middle Subset - The Middle Subset result set.



Note: When used in a formula visualization, you can effectively [filter the result](#) to the middle value(s), optionally [grouped](#) by an alignment hierarchy.

For more information, see:

- [Adding Formulas](#)
- [List of Formulas](#)
- [Using a Formula Visualization](#)

Minimum Subset

This applies to: *Managed Dashboards, Managed Reports*

The Minimum Subset function returns the smallest or bottom value(s) in an input data series.

When used in a [formula visualization](#), you can specify a count of more than one, for example to return the 3 smallest values.

Syntax

```
BOTTOM(d0,s0,Alignment)
```

Input

The Minimum Subset function requires the following input:

- d0 - Input data values - The set of data values for which the Minimum Subset is calculated.

Parameters

The Minimum Subset function accepts the following parameters:

- s0 - Minimum Subset Count (Optional) - The number of values to be returned by the formula. Default value is 1.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis. Specify a hierarchy to return the bottom input value(s) corresponding to that hierarchy's values: e.g., **BOTTOM(\$Measure\$, 1, \$Year\$)** will return the year with the smallest value within each grouping of years under another hierarchy's values such as **Country**. If an alignment hierarchy is not indicated, the subset is applied to the whole result.

When you use the BOTTOM function within a metric set, it can only return a single value (s0 must be 1). In order to return the bottom 3 values, for example, you must use a formula visualization. This applies to the other subset functions as well.

Output

The Minimum Subset function generates the following output:

- Minimum Subset - The Minimum Subset result set.



Note: When used in a formula visualization, you can effectively [filter the result](#) to the bottommost value(s), optionally [grouped](#) by an alignment hierarchy.

For more information, see:

- [Adding Formulas](#)
- [List of Formulas](#)
- [Using a Formula Visualization](#)



- Archive of documentation for Logi Symphony v24

Mode

This applies to: *Managed Dashboards, Managed Reports*

The Mode function finds the value (or values) repeated more often than any other in a value series.



Syntax

```
MODE (d0,Alignments...)
```

Input

The Mode function requires the following input:

- d0 - The set of data values for which the Mode is calculated.

Parameters

The Mode function accepts the following parameters:

- Alignments... (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis. Input values are grouped by the specified hierarchy values, and the mode calculated for each group: e.g., **MODE(\$Quantity\$, \$Country\$)** calculates the mode for each country.

Output

The Mode function generates the following output:

- Mode - One or more values representing the mode of the input values (for each group when using alignment parameters).

This can include multiple values if they each occur the same number of times, and mode values are therefore not repeated in every row when used in a formula measure.



Note: When used in a formula visualization where there are multiple mode values, they will be aggregated according to the aggregator setting of the [measure](#), which is **Sum** by default. You can change the aggregator to **Minimum** or **Maximum** instead, for example, to choose which mode to display.

For more information, see:

- [Adding Formulas](#)
- [List of Formulas](#)
- [Using a Formula Visualization](#)

Momentum

This applies to: *Managed Dashboards, Managed Reports*

The Momentum function is a trend-following indicator that measures the speed of price movements by comparing the current stock price against its value from a specified number of time periods in the past. A stock price has upward momentum when the indicator value is positive, and downward momentum when the indicator value is negative. Applications of Momentum include indication of possible trend reversals.

Syntax

```
MOMENTUM(d0,s0,Alignment)
```

Input

The Momentum function requires the following input series:

- d0 - The set of data values for which the Momentum is calculated, usually the daily close price of a stock.

Parameters

The Momentum function has the following parameter:

- s0 - Period - The number of time periods to use in the calculation. Default value is 5.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Momentum function generates the following outputs:

- Momentum - The Momentum result set.

For more information, see:



- Archive of documentation for Logi Symphony v24

- [Adding Formulas](#)
- [List of Formulas](#)
- [Using a Formula Visualization](#)

Money Flow Index

This applies to: *Managed Dashboards, Managed Reports*

The Money Flow Index function computes a relationship between the typical stock price and stock volume over a specified number of time periods. Resulting values range between 0 and 100. MFI is similar to the Relative Strength Indicator except that it incorporates volume. Applications of Money Flow Index include identifying market tops and market bottoms, and indicating price reversals.

Syntax

```
MONEYFLO(d0,d1,d2,d3,s0,Alignment)
```

Input

The Money Flow Index function requires the following input series:

- d0 - High data values - The first set of data values for which the Money Flow Index is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Money Flow Index is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Money Flow Index is calculated, usually the daily close price of a stock.
- d3 - Volume data values - The fourth set of data values for which the Money Flow Index is calculated, usually the daily volume.

Parameters

The Money Flow Index function has the following parameter:

- so - Period - The number of time periods to use in the calculation. Default value is 5.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Money Flow Index function generates the following output:



- Archive of documentation for Logi Symphony v24

- Money Flow Index - The Money Flow Index result set.

Moving Average

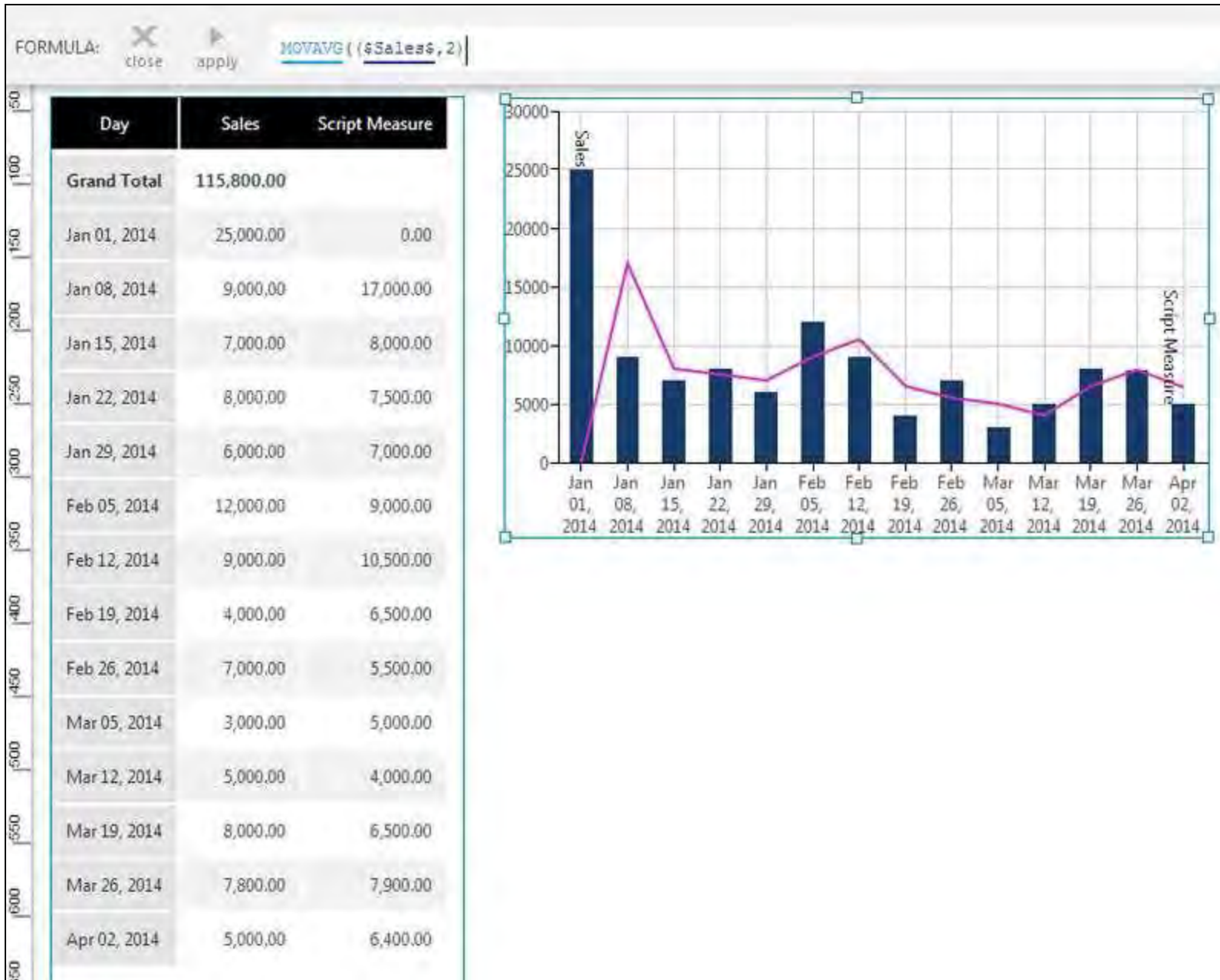
This applies to: *Managed Dashboards, Managed Reports*

The **Moving Average** (MA) function computes the average of a set of input values over a specified number of periods. The smaller the number of periods, the faster the moving average responds to changes in the input values.

The mathematical expression being calculated is as follows:

$$MA_t = (Y_t + Y_{t-1} + Y_{t-2} + \dots + Y_{t-N+1})/N$$

Where N is the number of time periods and Y is the value. You can find further details about the calculation of a moving average from Wolfram [MathWorld](#).



Syntax

```
MOVAVG (d0, s0, Alignment)
```

Input

The Moving Average function requires the following input:

- d0 - Input data values - The set of data values for which the Moving Average is calculated.

Parameters

The Moving Average function requires the following parameters.

- s0 - The number of periods to use in the calculation. The default value is 10.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Moving Average function generates the following output

- Moving Average - The Moving Average result set.

For more information, see:

- [Adding Formulas](#)
- [Using a Formula Visualization](#)
- [Weighted Moving Average](#)
- [Exponential Moving Average](#)

Moving Average Convergence-Divergence

This applies to: *Managed Dashboards, Managed Reports*

The **Moving Average Convergence-Divergence (MACD)** function is based on the indicator developed by Gerald Appel. It computes the difference between a short-term and a long-term exponential moving average of the stock price. Applications of MACD include generation of buy and sell signals. Signals are generated when the MACD line crosses the zero line or its signal line.

Syntax

Moving Average Convergence-Divergence:

```
MOVAVGCD(d0,s0,s1,s2,Alignment)
```

Signal Line:

```
MOVAVGCDSIG(d0,s0,s1,s2,Alignment)
```

Histogram:

```
MOVAVGCDHIST(d0,s0,s1,s2,Alignment)
```

Input

The Moving Average Convergence-Divergence functions require the following input series:

- d0 - Input data values - The set of data values for which the MACD is calculated, usually the daily close price of a stock.

Parameters

The Moving Average Convergence-Divergence function has the following parameters:

- s0 - Long Period - The larger number of time periods to use in the calculation of MACD. Default value is 26.
- s1 - Short Period - The smaller number of time periods to use in the calculation of MACD. Default value is 12.



- s2 - Signal Period - The number of time periods to use in the calculation of the Signal Line. Default value is 9.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The MOVAVGCD function generates the following output:

- Moving Average Convergence-Divergence - The MACO result set. The MOVAVGCDSIG function generates the following output:

The MOVAVGCDSIG function generates the following output:

- Signal Line - This is the result of smoothing out the MACO line via the application of an exponential moving average. The Signal Line can help to identify turns in the MACO line.

The MOVAVGCDHIST function generates the following output:

- Histogram - The Histogram is computed as the difference between the MACD line and the Signal Line. This makes it easy to see when the two lines converge or diverge.

Moving Average Envelopes

This applies to: *Managed Dashboards, Managed Reports*

The **Moving Average Envelope** functions compute a pair of data bands that envelop a moving average of the input data values.

Each data band is positioned a percentage-based magnitude away from the moving average line. The average line can be calculated using a [simple moving average](#) or an [exponential moving average](#). Applications of the Moving Average Envelope functions include indication of overbought/oversold conditions and price trends for a security.

The mathematical expressions being calculated are as follows:

$$\text{Upper} = \text{Moving Average} * (1 + K)$$

$$\text{Lower} = \text{Moving Average} * (1 - K)$$

Where K is the indicated percentage-based factor.

Using **MOVAVGEPUPPER**:



Using **MOVAVGEVFLOWER**:



Note: A similar envelope can be generated by the [Bollinger Bands](#) functions, which uses standard deviations instead of a percentage-based factor.

Syntax

Upper Envelope:

`MOVAVGEVPUPPER(d0, 50, 51, 52, Alignment)`

Lower Envelope:

`MOVAVGEVPLOWER(d0, 50, 51, 52, Alignment)`

Input

The Moving Average Envelope functions require the following input:

- d0 - Input data values - The set of data values for which the MOVAVGEVPUPPER and MOVAVGEVPLOWER formulas are calculated.

Parameters

The Moving Average Envelope functions require the following parameters.

- s0 - Moving Average Type - The type of moving average to use in the calculation. Valid values are:
 - 0: Simple Moving Average (default)
 - 1 : Exponential Moving Average
- s1 - Period - The number of time periods to use in the calculation. Default value is 10.
- s2 - Envelope factor- The factor used to shift the bands away from the simple moving average. Default value is 2.5.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Moving Average Envelope functions generate the following output:

- Upper Envelope - The Upper Envelope result set.
- Lower Envelope - The Lower Envelope result set.

For more information, see:

- [Adding Formulas](#)
- [Bollinger Bands](#)
- [Moving Average](#)
- [Exponential Moving Average](#)

Negative Volume Index

This applies to: *Managed Dashboards, Managed Reports*

The **Negative Volume Index** function calculates a cumulative value based on days where the trading volume has decreased from the previous day. On days where the volume remained the same or increased, the NVI remains unchanged. Applications of NVI include indication of market trends. For example, a bull market may be indicated when the NVI rises above its one-year moving average.

Syntax

```
NEGVOLUME (d0,d1,Alignment)
```

Input

The Negative Volume Index function requires the following input series:

- d0 - Volume data values - The first set of data values for which the Negative Volume Index is calculated, usually the daily volume for a stock.
- d1 - Price data values - The second set of data values for which the Negative Volume Index is calculated, usually the daily close price of a stock.

Parameters

The Negative Volume Index function requires the following parameters:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Negative Volume Index function generates the following output:

- Negative Volume Index - The Negative Volume Index result set.



Normal Distribution

This applies to: *Managed Dashboards, Managed Reports*

The **Normal Distribution** function calculates the standard normal distribution for an input data series.

Specifically, the function computes the relative likelihood that a continuous random variable would equal the input value if the series had a symmetrical "bell-curved" distribution.

apply NORMALDIST (\$Sales\$)

Day	Sales	Script Measure
Grand Total	842.00	
Jan 01, 2014	65.00	0.05
Jan 08, 2014	60.00	0.11
Jan 15, 2014	55.00	0.04
Jan 22, 2014	52.00	0.01
Jan 29, 2014	56.00	0.06
Feb 05, 2014	58.00	0.09
Feb 12, 2014	59.00	0.10
Feb 19, 2014	62.00	0.10
Feb 26, 2014	63.00	0.08
Mar 05, 2014	65.00	0.05
Mar 12, 2014	61.00	0.11
Mar 19, 2014	63.00	0.08
Mar 26, 2014	62.00	0.10
Apr 02, 2014	61.00	0.11



The Normal Distribution Group function can be optionally used to calculate the standard normal distribution for each input data series grouped by the specified hierarchies.

Syntax

Normal Distribution:

```
NORMALDIST (d0, Alignments...)
```

Normal Distribution Group:

```
NORMALDISTGROUP (d0, Alignments...)
```

Input

The Normal Distribution functions require the following input series:

- d0 - Input data values - The set of data values for which the Normal Distribution is calculated.

Parameters

The Normal Distribution functions accept the following parameters:

- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.
 - For NORMALDIST, specifying a hierarchy calculates standard normal distribution for the corresponding input values separately each time they occur: e.g., **NORMALDIST(\$Measure\$, \$Month\$)** can calculate results based on monthly values grouped by another hierarchy such as Country.
 - For NORMALDISTGROUP, specify the hierarchies by which to group the input values and calculate the standard normal distribution within each group: e.g., **NORMALDISTGROUP(\$Measure\$, \$Country\$)** will calculate results separately for the values within each country.

Output

The Normal Distribution functions generate the following outputs:

- Normal Distribution - The Normal Distribution result set(s).



On Balance Volume

This applies to: *Managed Dashboards, Managed Reports*

The **On Balance Volume** function is based on the indicator developed by Joseph Granville. It calculates a cumulative volume that depends on stock price. When the closing stock price for the day is greater than the previous day's close, the volume for the day is added to the cumulative result. If the stock price decreases, volume is subtracted from the result. Applications of On Balance Volume include confirmation of price movements.

Syntax

```
ONBALVOLUME (d0,d1,Alignment)
```

Input

The On Balance Volume function requires the following input series:

- d0 - Volume data values - The first set of data values for which the On Balance Volume is calculated, usually the daily volume for a stock.
- d1 - Price data values - The second set of data values for which the On Balance Volume is calculated, usually the daily close price of a stock.

Parameters

The On Balance Volume function requires the following parameters:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The On Balance Volume function generates the following output:

- On Balance Volume - The On Balance Volume result set.

On Balance Weighted Volume

This applies to: *Managed Dashboards, Managed Reports*

The **On Balance Weighted Volume** function calculates a cumulative volume that depends on stock price. The daily volume is weighted by the difference in stock price between the day's close and the previous day's close before it is added to the cumulative result. Applications of On Balance Weighted Volume include confirmation of price movements and providing indication of possible trend reversals.

Syntax

```
ONBALVOLWGHT (d0,d1,Alignment)
```

Input

The On Balance Weighted Volume function requires the following input series:

- d0 - Volume data values - The first set of data values for which the On Balance Weighted Volume is calculated, usually the daily volume for a stock.
- d1 - Price data values - The second set of data values for which the On Balance Weighted Volume is calculated, usually the daily close price of a stock.

Parameters

The On Balance Weighted Volume function requires the following parameters:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The On Balance Weighted Volume function generates the following output:

- On Balance Weighted Volume - The On Balance Weighted Volume result set.

Parabolic SAR (stop-and-reverse)

This applies to: *Managed Dashboards, Managed Reports*

The **Parabolic SAR (stop-and-reverse)** function is based on the indicator developed by J. Welles Wilder, Jr. It is a lagging technical indicator that can help you to determine when a stock is set to have a change in trend.

Applications of Parabolic SAR include generation of buy-sell signals and, in particular, determining when to exit from trades.

Syntax

```
PARABOLSAR(d0,d1,s0,s1,Alignment)
```

Input

The Parabolic SAR formula requires the following input series:

- d0 - High data values - The first set of data values for which the Parabolic SAR is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Parabolic SAR is calculated, usually the daily low price of a stock.

Parameters

The Parabolic SAR function has the following parameter:

- s0 - Acceleration Factor - The acceleration factor used in the calculations.
- s1 - Maximum Acceleration - The maximum value that the acceleration is allowed to reach.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Parabolic SAR function generates the following output:

- Parabolic SAR - The Parabolic SAR result set.



Percentile

This applies to: *Managed Dashboards, Managed Reports*

The **Percentile** function returns a scalar value below which the specified percentage of input data values fall. The closest input value that satisfies the condition is returned.

apply PERCENTILE (\$Sales\$, 80, "TRUE")

Day	Sales	Script Measure
Grand Total	842.00	
Jan 01, 2014	65.00	63.00
Jan 08, 2014	60.00	63.00
Jan 15, 2014	55.00	63.00
Jan 22, 2014	52.00	63.00
Jan 29, 2014	56.00	63.00
Feb 05, 2014	58.00	63.00
Feb 12, 2014	59.00	63.00
Feb 19, 2014	62.00	63.00
Feb 26, 2014	63.00	63.00
Mar 05, 2014	65.00	63.00
Mar 12, 2014	61.00	63.00
Mar 19, 2014	63.00	63.00
Mar 26, 2014	62.00	63.00
Apr 02, 2014	61.00	63.00

Syntax

```
PERCENTILE (d0,s0,s1,s2,Alignments...)
```

Input

The Percentile function requires the following input:

- d0 - The set of data values for which the Percentile is calculated.

Parameters

The Percentile function accepts the following parameters, which may be left out to use their default values:

- s0 - A percentage value ranging between 0-100 to determine the calculated percentile. Default value is 80.
- s1 - Indicates whether to include the two extreme percent values in the calculation. Default value is true.
- s2 - The methodology used for calculating the percentile. Obsolete in Dundas BI version 11 and higher as direct computation is always used.

Available choices:

- 0 - The percent rank computation method (default)
- 1 - The direct computation method
- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.

Output

The Percentile formula generates the following output:

- Percentile - A single value representing the specified percentile.



Percent of Total

This applies to: *Managed Dashboards, Managed Reports*

The **Percent of Total** function returns the percentage of each value in the input series out of the total sum of the values in the input series.

The **Percent of Total Group** function can be optionally used to return the percentage of each value out of the total sum of its group, where the values are grouped by the specified hierarchies.

Syntax

Percent of Total:

```
PERCENTTOTAL(d0,Alignments...)
```

Percent of Total Group:

```
PERCENTTOTALGROUP(d0,Alignments...)
```

Input

The Percent of Total functions require the following input series:

- d0 - The set of data values for which the formula is calculated.

Parameters

The Percent of Total functions accept the following parameters:

- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.
 - For PERCENTTOTAL, specifying a hierarchy calculates the percentage for each of its corresponding values out of their total separately each time those hierarchy values occur: e.g., **PERCENTTOTAL(\$Measure\$, \$Year\$)** can calculate the percentage for each year out of the total within each Country group.
 - For PERCENTTOTALGROUP, specify the hierarchies by which to group the input values and calculate the percentages of the totals within each group: e.g., **PERCENTTOTAL(\$Measure\$, \$Country\$)** will calculate the percentage each value represents out of each country's total.

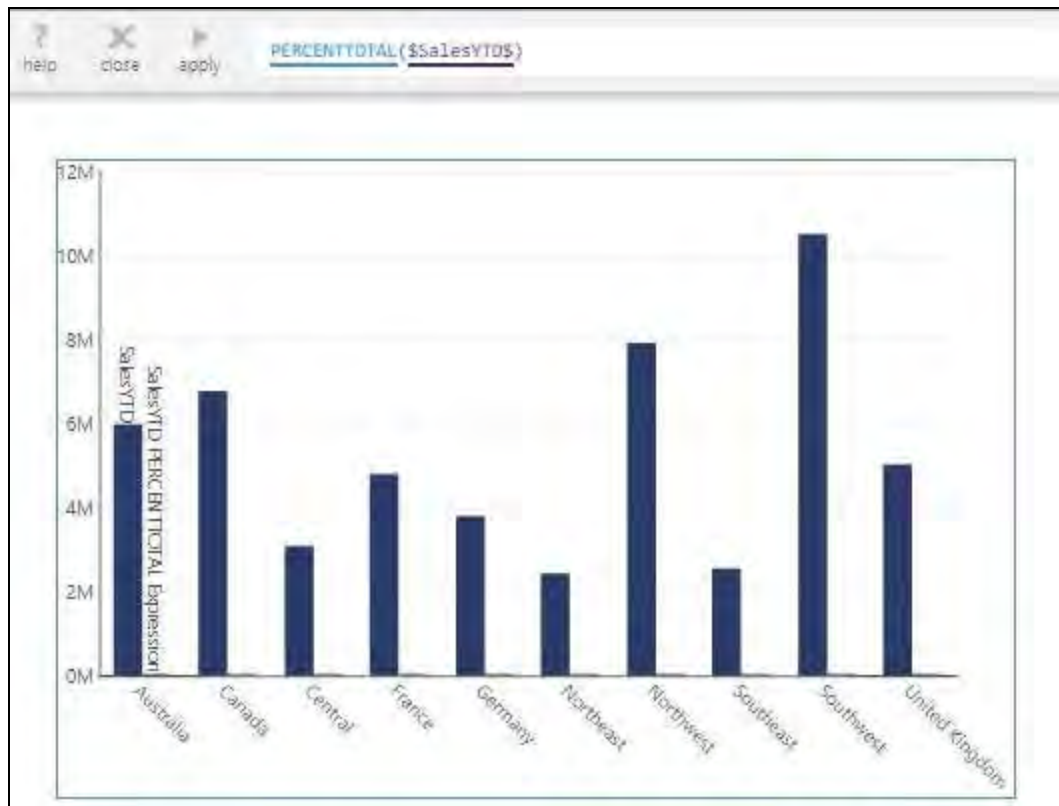
Output

The Percent of Total functions generate the following output:

- Percent of Total - The Percent of Total result set consisting of values between 0-1 (inclusive) assuming positive input values, representing each input value's percentage of the total.

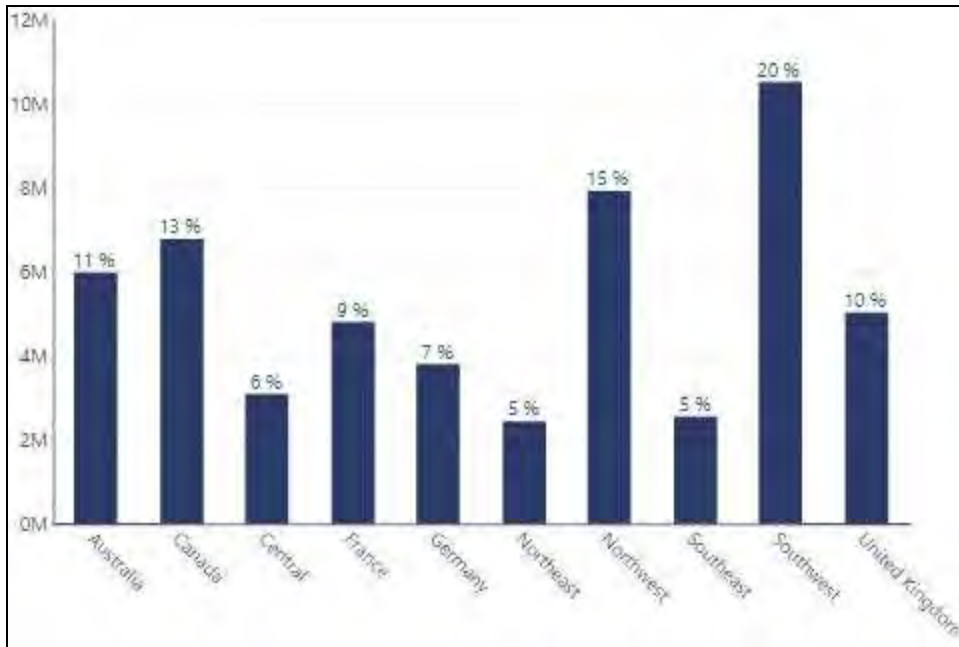
Example

As an example, the figure below shows the initial result of applying the Percent of Total function to a bar chart displaying sales figures for each country, visualized initially as a second bar chart series:



The formula's output is a percentage, which is not useful for comparison against the same chart axis as the original measure that uses a currency as its unit. You can click to remove this formula from the second series from the Data Analysis Panel's [Visualization tab](#), or remove the second series from the chart's [properties](#) in the Properties window.

A common example of how to use this formula's output is to display the percentages as labels next to each data point or in tooltips. You can add the formula under **Data Point Label** in the Data Analysis Panel's Visualization tab, or in the properties for the first chart series. See [Setting Up the Visualization](#).





Percent Rank

This applies to: *Managed Dashboards, Managed Reports*

The **Percent Rank** function calculates the percent rank for each value in a set of input values. Specifically, the function computes the percentage of all input values that are less than a given input value. Percent Rank values range between 0 and 1.

apply `PERCENTRANK($Sales$, "TRUE")`

Day	Sales	Script Measure
Grand Total	842.00	
Jan 01, 2014	65.00	0.92
Jan 08, 2014	60.00	0.38
Jan 15, 2014	55.00	0.08
Jan 22, 2014	52.00	0.00
Jan 29, 2014	56.00	0.15
Feb 05, 2014	58.00	0.23
Feb 12, 2014	59.00	0.31
Feb 19, 2014	62.00	0.62
Feb 26, 2014	63.00	0.77
Mar 05, 2014	65.00	0.92
Mar 12, 2014	61.00	0.46
Mar 19, 2014	63.00	0.77
Mar 26, 2014	62.00	0.62
Apr 02, 2014	61.00	0.46



The **Percent Rank Group** function can be optionally used to calculate the percent rank for each value in sets of input values grouped by the specified hierarchies.

Syntax

Percent Rank:

```
PERCENTRANK(d0,s0,Alignments...)
```

Percent Rank Group:

```
PERCENTRANKGROUP(d0,s0,Alignments...)
```

Input

The Percent Rank function requires the following input:

- d0 - The set of data values for which the Percent Rank is calculated.

Parameters

The Percent Rank functions require the following parameters.

- s0 - Indicates whether to include the two extreme percent values in the calculation. Default value is True.
- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.
 - For PERCENTRANK, specifying a hierarchy calculates percentage ranks for the corresponding input values separately each time they occur: e.g., **PERCENTRANK(\$Measure\$, \$Year\$)** can calculate separate percent ranks for yearly values within groupings for another hierarchy such as **Country**.
 - For PERCENTRANKGROUP, specify the hierarchies by which to group the input values and calculate percentage ranks within each group: e.g., **PERCENTRANKGROUP(\$Measure\$, \$Country\$)** will calculate the percentage ranking for each value within each country.

Output

The Percent Rank functions generate the following output:

- Percent Rank - The Percent Rank result set.



Population Standard Deviation

This applies to: *Managed Dashboards, Managed Reports*

The **Population Standard Deviation** function measures volatility as the difference between values and their average. A larger difference implies a larger standard deviation value.

apply SIDEVPOP(\$Sales\$)

Day	Sales	Script Measure
Grand Total	842.00	
Jan 01, 2014	65.00	3.66
Jan 08, 2014	60.00	3.66
Jan 15, 2014	55.00	3.66
Jan 22, 2014	52.00	3.66
Jan 29, 2014	56.00	3.66
Feb 05, 2014	58.00	3.66
Feb 12, 2014	59.00	3.66
Feb 19, 2014	62.00	3.66
Feb 26, 2014	63.00	3.66
Mar 05, 2014	65.00	3.66
Mar 12, 2014	61.00	3.66
Mar 19, 2014	63.00	3.66
Mar 26, 2014	62.00	3.66
Apr 02, 2014	61.00	3.66

Syntax

```
STDEVPOP (d0,Alignment)
```

Input

The Population Standard Deviation function requires the following input:

- d0 - The set of data values for which the function is calculated.

Parameters

The Population Standard Deviation function accepts the following parameter:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Population Standard Deviation function generates the following output:

- Population Standard Deviation - The Population Standard Deviation result set.



Positive Volume Index

This applies to: *Managed Dashboards, Managed Reports*

The **Positive Volume Index** function calculates a cumulative value based on days where the trading volume has increased from the previous day. On days where the volume remained the same or decreased, the PVI remains unchanged. Applications of PVI include indication of market trends. For example, a bear market may be indicated when the PVI drops below its one-year moving average.

Syntax

```
POSVOLUME (d0,d1,Alignment)
```

Input

The Positive Volume Index function requires the following input series:

- d0 - Volume data values - The first set of data values for which the Positive Volume Index is calculated, usually the daily volume for a stock.
- d1 - Price data values - The second set of data values for which the Positive Volume Index is calculated, usually the daily close price of a stock.

Parameters

The Positive Volume Index function requires the following parameters:

- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Positive Volume Index function generates the following output:

- Positive Volume Index - The Positive Volume Index result set.

Rank Function

This applies to: *Managed Dashboards, Managed Reports*

The **Rank** function calculates the rank for each value in a set of input values.

 <u>RANK(\$SalesYTD\$,0,0,0)</u>		
Name	SalesYTD	RANK SalesYTD Expres...
Southwest	10,510,853.8739	1.00
Northwest	7,887,186.7882	2.00
Canada	6,771,829.1376	3.00
Australia	5,977,814.9154	4.00
United Kingdom	5,012,905.3656	5.00
France	4,772,398.3078	6.00
Germany	3,805,202.3478	7.00
Central	3,072,175.1180	8.00
Southeast	2,538,667.2515	9.00
Northeast	2,402,176.8476	10.00

Syntax

```
RANK(d0,s0,s1,s2,Alignment)
```

Input

The Rank function requires the following input:

- d0 - The set of data values for which the rank is calculated.

Parameters

The Rank function requires the following parameters.

- s0 - Indicates which way the values should be sorted for ranking. 0 for descending, 1 for ascending. The default value is 0.
- s1 - Indicates how ties should be handled. The default value is 0.
 - 0 - Order of occurrence breaks tie
 - 1 - Podium
 - 2 - Lowest possible
 - 3 - Highest possible
- s2 - If greater than 0, ranks will be scaled so that the highest rank is the value. The default value is 0.
- Alignment (optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Rank function generates the following output:

- Rank - The Rank result set, consisting of rank values corresponding with each input value.

For more information, see:



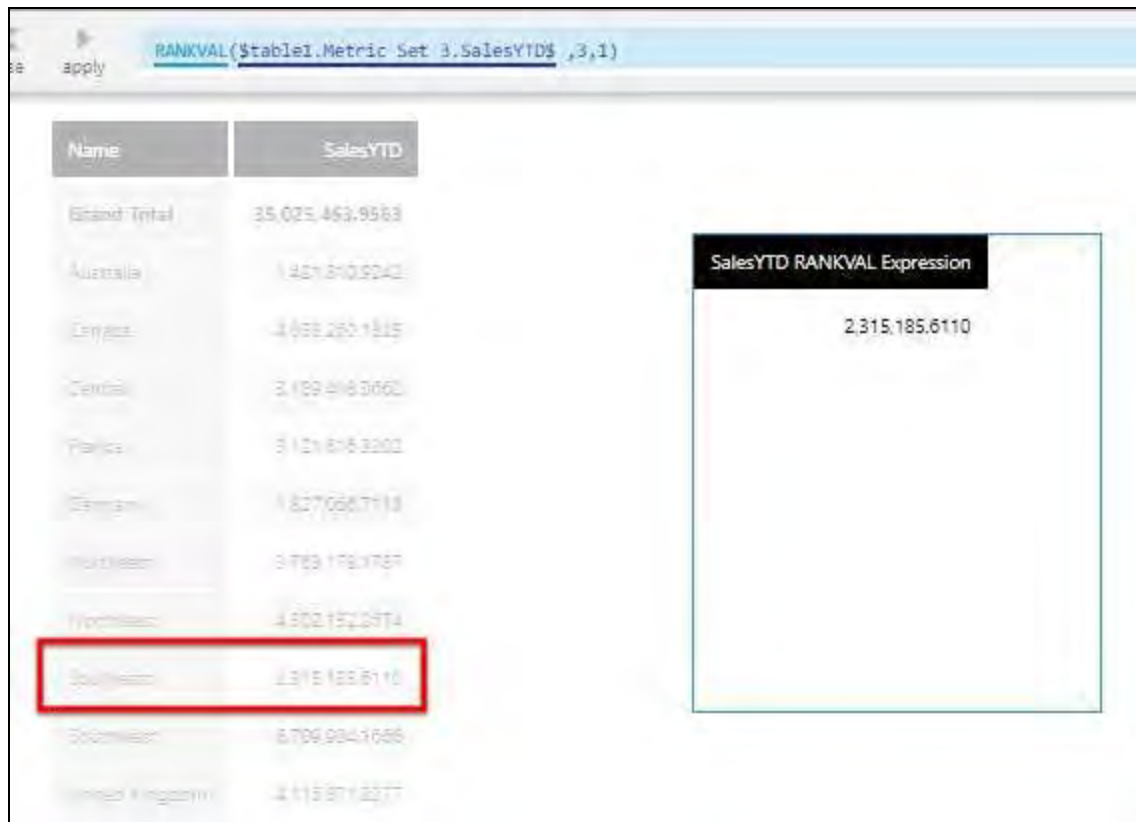
- Archive of documentation for Logi Symphony v24

- [Rank](#)
- [Adding Formulas](#)

Rank Value

This applies to: *Managed Dashboards, Managed Reports*

The **Rank Value** function returns the value found at the specified position, either starting from the highest or the lowest value.



Syntax

```
RANKVAL(d0, s0,s1 s2,Alignments ... )
```

Input

The Rank Value function requires the following input:

- d0 - The set of data values for which the Rank Value is calculated.

Parameters

The Rank Value function requires the following parameters:

- s0 - The rank position in the list of values.
- s1 - The sorting type. 0 ranks from the highest value to the lowest, while 1 ranks from the lowest to the highest.
- s2 - Whether duplicate values are considered as multiple values in the ranking. The default value is false.
- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis. Input values are grouped by the specified hierarchy values before ranking them: e.g., `RANKVAL($Quantity$, 2, null, null, $Country$)` calculates the 2nd highest quantity for each country.



Note: When using alignment parameters in a [formula visualization](#), you can choose to display these values grouped by only the alignment hierarchy values, for example to display just the N-ranked value for each country without the original input values.

Output

The Rank Value function generates a single value from the data set that corresponds to the indicated rank (or for each group when using alignment parameters).

For more information, see:

- [Using a Formula Visualization](#)
- [Adding Formulas](#)
- [Percent Rank](#)
- [Rank](#)

Rate of Change

This applies to: *Managed Dashboards, Managed Reports*

The **Rate of Change** function is a variation on the [Momentum](#) formula. ROC calculates the percentage difference between the day's closing stock price and the price from a specified number of time periods in the past. Applications of ROC include indication of price reversals. For example, if the price is trending upward while the ROC is moving downward, this may indicate a large drop in price is forthcoming.

Syntax

```
RATECHANGE (d0,s0,Alignment)
```

Input

The Rate of Change function requires the following input series:

- d0 - The set of data values for which the Rate of Change is calculated, usually the daily close price of a stock.

Parameters

The Rate of Change function has the following parameters:

- s0 - Period - The number of time periods to use in the calculation. Default value is 5.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Rate of Change formula generates the following output:

- Rate of Change - The Rate of Change result set.

Relative Strength Indicator

This applies to: *Managed Dashboards, Managed Reports*

The **Relative Strength Indicator** function is based on the indicator developed by J. Welles Wilder. It compares the average of up closes against the average of down closes over a specified number of time periods. Resulting values range from 0 to 100. Applications of RSI include identifying overbought or oversold situations. For example, if the RSI line stalls at a level above 70, this may indicate the stock is overbought. Or, if the RSI line drops to a level below 30, this may indicate the stock is oversold and becoming undervalued.

Syntax

```
RELSTRENGTH(d0,s0,Alignment)
```

Input

The Relative Strength Indicator function requires the following input series:

- d0 - Input data values - The set of data values for which the Relative Strength Indicator is calculated, usually the daily close price of a stock

Parameters

The Relative Strength Indicator function has the following parameter:

- s0 - Period - The number of time periods to use in the calculation.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

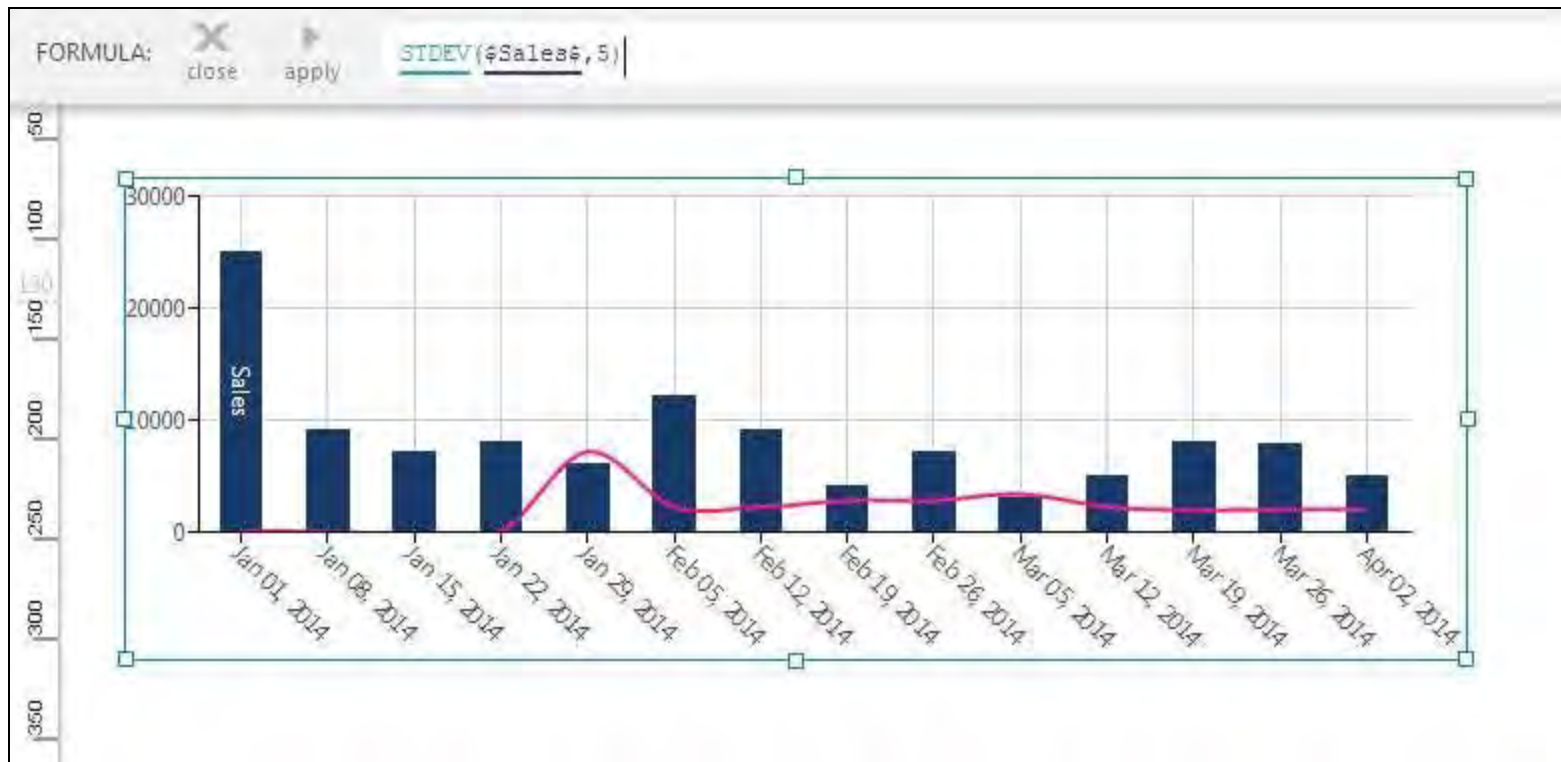
The Relative Strength Indicator function generates the following output

- Relative Strength Indicator - The Relative Strength Indicator result set, which consists of values ranging between 0 and 100.

Sample Standard Deviation

This applies to: *Managed Dashboards, Managed Reports*

The **Sample Standard Deviation** function measures volatility as the difference between values and their moving average. A larger difference implies a larger standard deviation value.



Syntax

```
STDEV(d0,s0,Alignment)
```

Input

The Sample Standard Deviation function requires the following input:



- d0 - The set of data values for which the formula is calculated.

Parameters

The Sample Standard Deviation function requires the following parameters:

- s0 - The number of time periods to use in the calculation. The default value is 10.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Sample Standard Deviation function generates the following output:

- Sample Standard Deviation - The Sample Standard Deviation result set.

Slow Stochastic

This applies to: *Managed Dashboards, Managed Reports*

The **Slow Stochastic** function compares the close price of a stock against its price range (high-low) over a specified number of time periods. It is derived by applying a 3-period, simple moving average to the Fast Stochastic line. Applications of Slow Stochastic include the generation of buy and sell signals.

Syntax

Slow Stochastic:

```
SLOSTOC(d0,d1,d2,s0,Alignment)
```

Trigger Line:

```
SLOSTOCTRG(d0,d1,d2,s0,Alignment)
```

Input

The Slow Stochastic functions require the following input series:

- d0 - High data values - The first set of data values for which the Slow Stochastic formula is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Slow Stochastic formula is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Slow Stochastic formula is calculated, usually the daily close price of a stock.

Parameters

The Slow Stochastic function has the following parameters:

- s0 - Look Back Period - The number of time periods for determining the overall high and low values. Default value is 14.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.



Output

The SLOSTOC function generates the following output

- Slow Stochastic - The Slow Stochastic result set.

The SLOSTOCTRG function generates the following output

- Trigger Line - The trigger line is a 3-period, simple moving average that is used to smooth out the Slow Stochastic values.

Stochastic Oscillator

This applies to: *Managed Dashboards, Managed Reports*

The **Stochastic Oscillator** function compares the close price of a stock against its price range (high-low) over a specified number of time periods. Applications of Stochastic Oscillator include the generation of buy and sell signals.

Syntax

Stochastic Oscillator:

```
STOCOSC(d0,d1,d2,s0,s1,Alignment)
```

Trigger Line:

```
STOCOSCTRG(d0,d1,d2,s0,s1,Alignment)
```

Input

The Stochastic Oscillator functions require the following input series:

- d0 - High data values - The first set of data values for which the Stochastic Oscillator formula is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Stochastic Oscillator formula is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Stochastic Oscillator formula is calculated, usually the daily close price of a stock.

Parameters

The Stochastic Oscillator function has the following parameters:

- s0 - Look Back Period - The number of time periods for determining the overall high and low values. Default value is 14.
- s1 - SMA Period - The number of time periods for the simple moving average calculation which is used to generate the trigger line. Default value is
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.



Output

The STOCOSC function generates the following output:

- Stochastic Oscillator - The Stochastic Oscillator result set. The STOCOSCTRG function generates the following output
- Trigger Line - The trigger line is a 3-period, simple moving average that is used to smooth out the Stochastic Oscillator values.

Totals (Sum)

This applies to: *Managed Dashboards, Managed Reports*

The Sum function calculates the sum total for a set of input values.

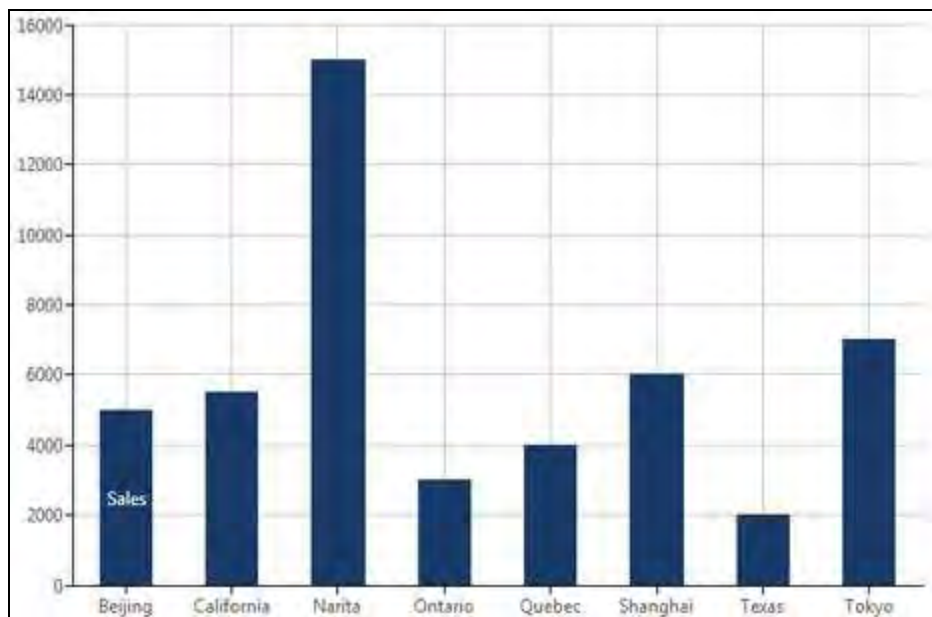
Syntax

```
SUM(d0,Alignments...)
```

Input

The Sum function requires the following input:

- d0 - The set of data values for which the Sum is calculated.



Parameters

The Sum function accepts the following parameters:

- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis. Input values are grouped by the specified hierarchy values, and the sum calculated for each group: e.g., **SUM(\$Quantity\$, \$Country\$)** calculates the total for each country.

Output

The Sum function generates the following output:

- Total - A single value representing the sum total of the input values (or for each group when using alignment parameters).



Note: To display the output in a separate visualization such as a [data label](#), you can use a [formula visualization](#).



Trend and Forecasting

This applies to: *Managed Dashboards, Managed Reports*

The **Trend and Forecasting** function calculates a regression line or line of best fit. When applied to a time series, it can forecast future values.

This function can be used with a measure and date/time values, or with two measures where each is typically plotted along a separate chart axis.



There are four regression functions available, which you can choose from when setting the function parameters:

- Linear (or polynomial)
- Exponential

- [Logarithmic](#)
- [Power](#)

The **Upper Band Error** and **Lower Band Error** functions can be optionally used to add upper and lower error boundaries, which increase as the forecast portion of the regression line progresses.

Syntax

Trend and Forecasting:

```
TRENDFORE d0, d1,s0,s1,s2)
```

Upper Band Error:

```
TRENDFOREUPPER(d0,d1,s0,s1,s2)
```

Lower Band Error:

```
TRENDFORELOWER(d0,d1,s0,s1 s2)
```

Input

The Trend and Forecasting functions require the following input series:

- d0 - Trend Axis values - The set of values on the trending axis (e.g., the date/time values, or the measure plotted on the X/bottom axis opposite from the input data values).
- d1 - Input data values - The measure for which to determine the trend.

Parameters

- s0 - Regression Type - The type of regression function used by the Trend and Forecasting formula. Available choices include:
 - 0 - Linear (default)
 - 1 - Exponential
 - 2 - Logarithmic
 - 3- Power
- s1 - Regression Degree - The degree used by the regression function in calculating the Trend and Forecasting formula. Default value is 2.
- s2 - Forecasting Period - The length of the forecasted data series as returned by the Trend and Forecasting formula. Available choices include:
 - 0 - No Forecast
 - 1 - Tenth Input
 - 2 - Quarter Input
 - 3 - Half Input (default)
 - 4 - Full Input

Output

The Trend and Forecasting functions generate the following outputs:

- Trend and Forecasting - The Trend and Forecasting result set.
- Upper Band Error - The upper error boundary based on standard deviation and the forecasting error.
- Lower Band Error - The lower error boundary based on standard deviation and the forecasting error.

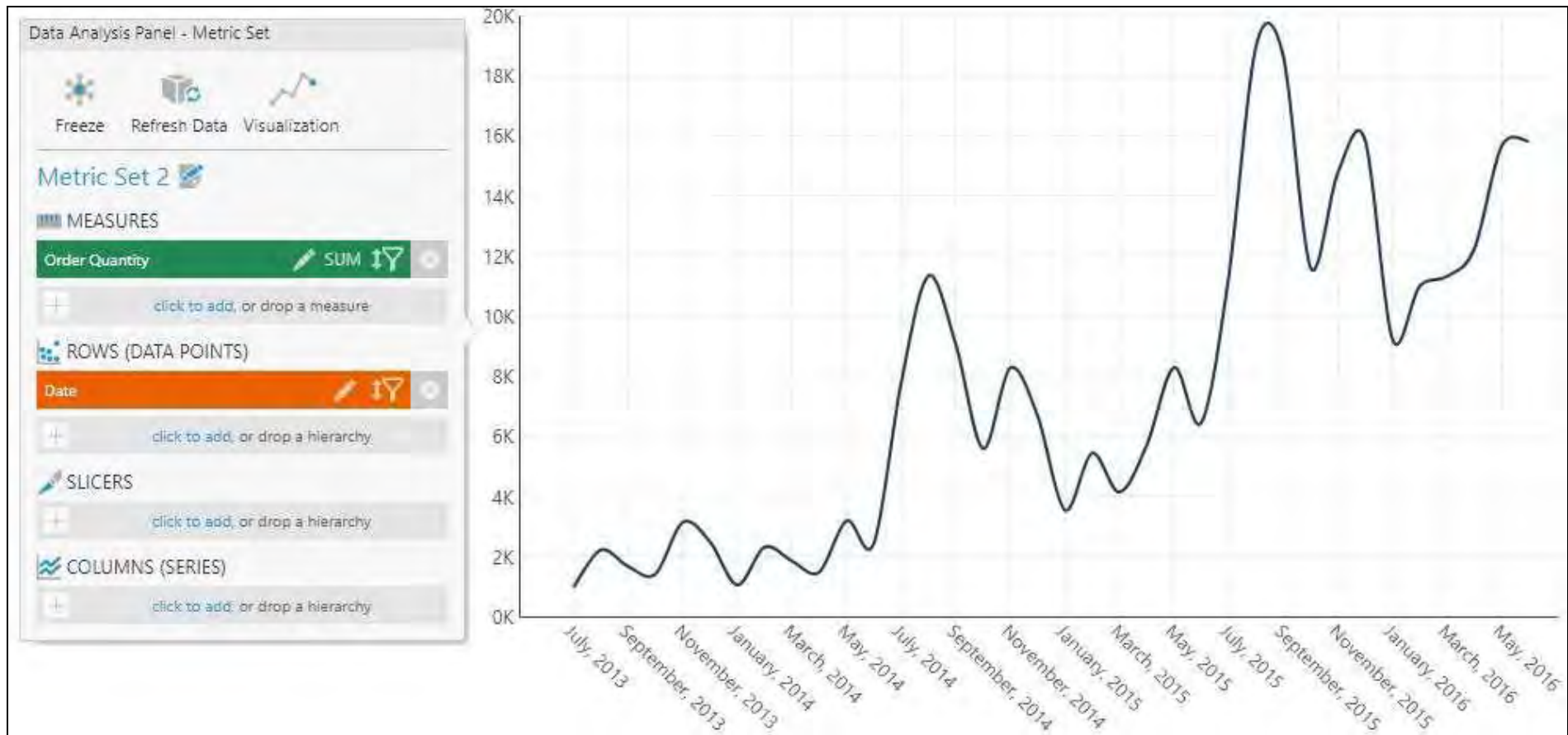
The upper and lower error boundaries are calculated as two standard deviations away from the trend/forecast, plus an additional standard deviation for every step past the original input values when forecasting.

Examples

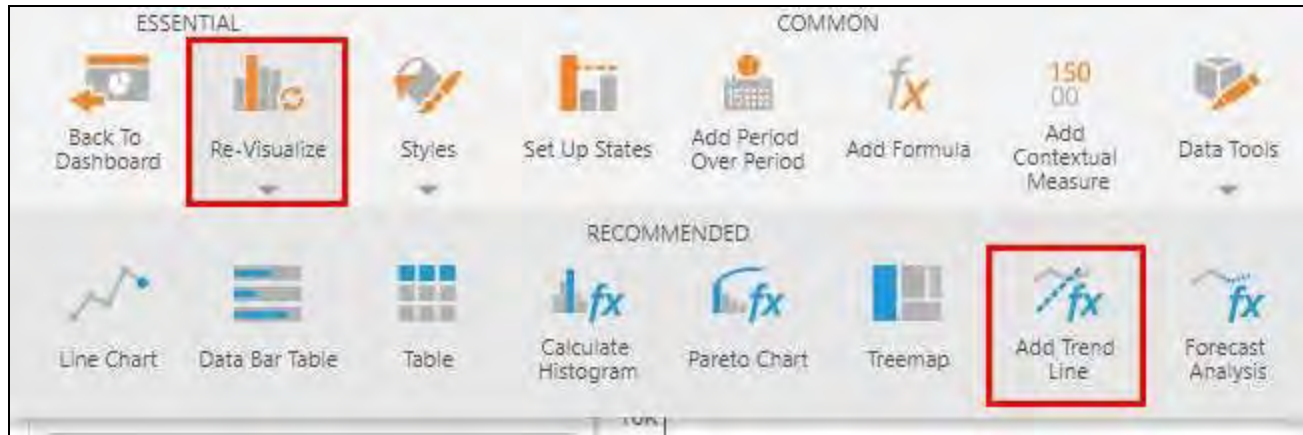
Trend and Forecast on a Time Series

This example adds a trend line including a forecast to a line chart.

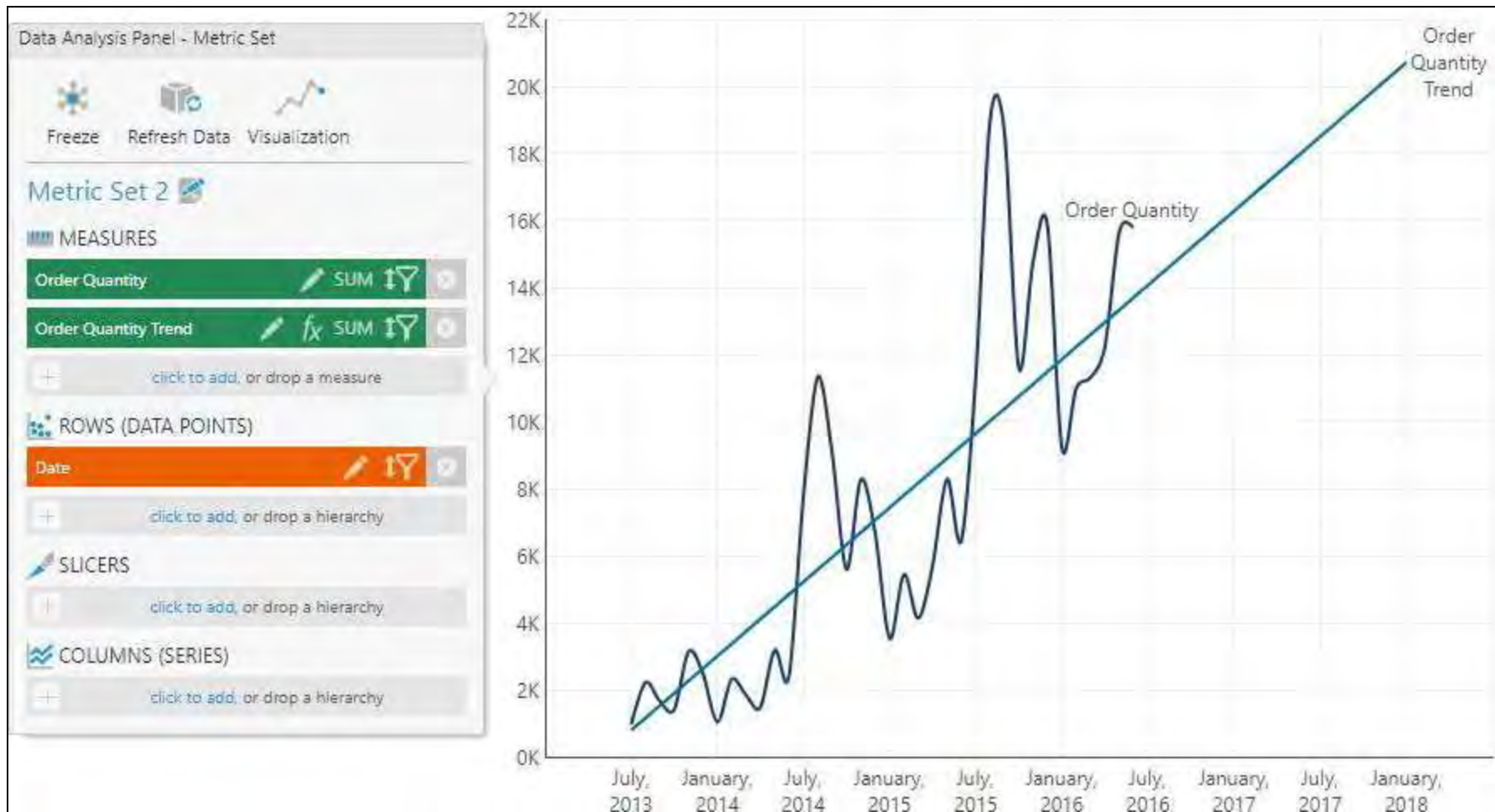
With a measure and time dimension hierarchy displayed in a chart like the following:



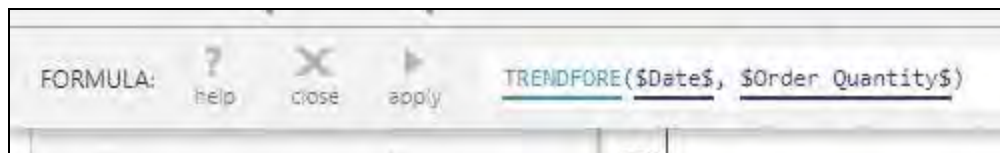
You can quickly add a trend line by choosing **Re-Visualize** in the toolbar, then **Add Trend Line**.



This adds a formula measure that uses the TRENDFORE function with the default forecasting settings, visualized as a new line series.



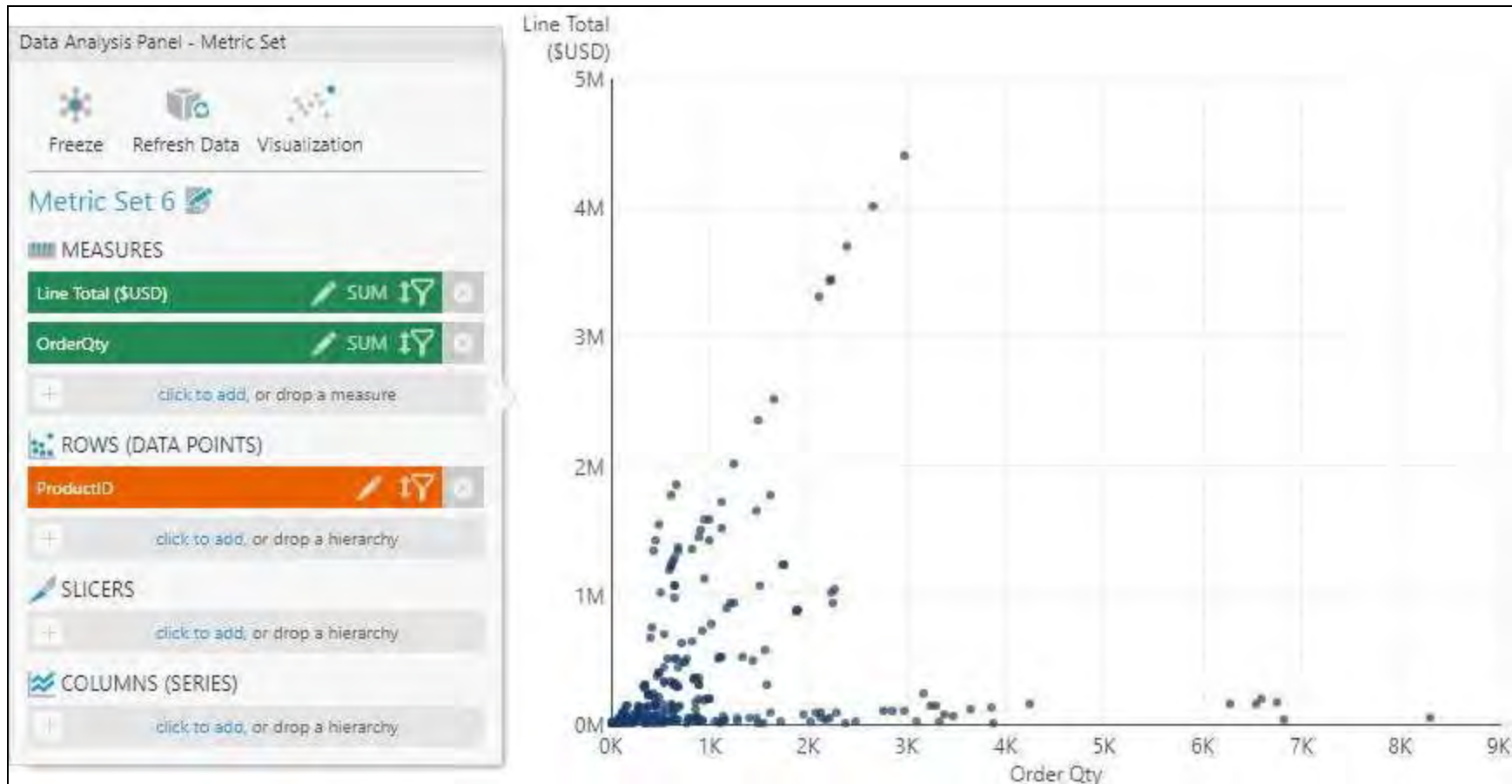
You can now edit the formula if you choose to.



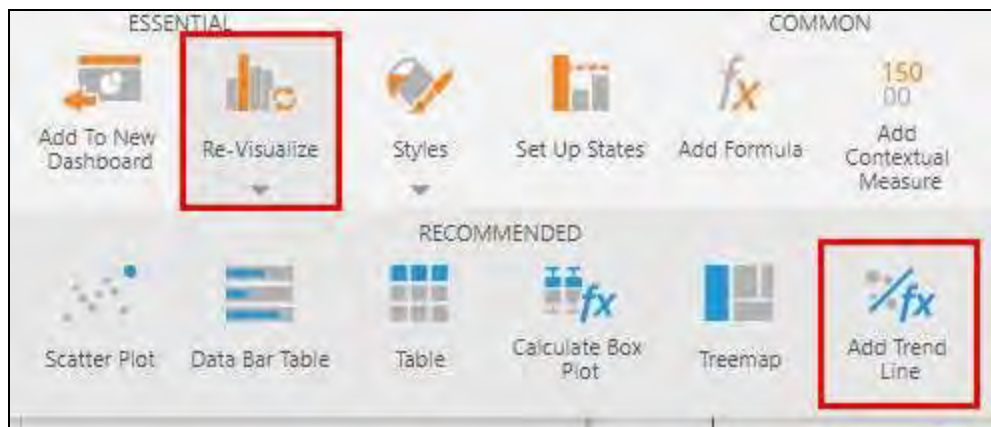
The screenshot shows the 'FORMULA:' field with the formula: `TRENDFORE($Date$, $Order Quantity$)`. The formula is displayed in a blue font. Above the formula field are three icons: a question mark (help), a close button (X), and an apply button (play icon). Below the formula field are three buttons: 'help', 'close', and 'apply'.

Linear Regression on Scatter Plot

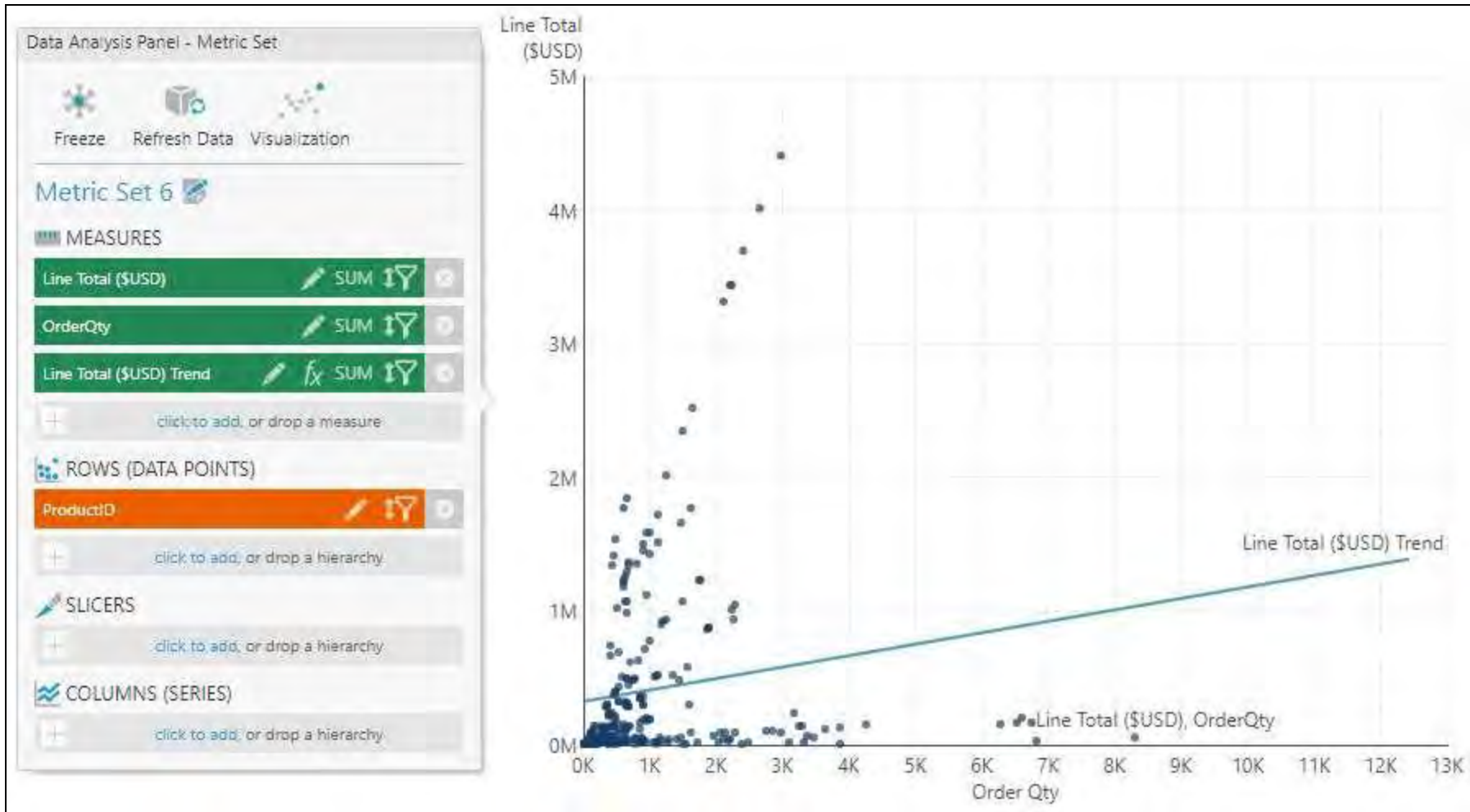
This example shows how to add a regression line, or line of best fit to a scatter plot. With a scatter plot chart displaying two measures grouped by one or more hierarchies:



You can quickly add a trend line by choosing **Re-Visualize** in the toolbar, then **Add Trend Line**.



This adds a formula measure that uses the TRENDFORE function, visualized as a new line series.



You can now edit the formula if you choose to.



For more information, see:



- Archive of documentation for Logi Symphony v24

- [Using Symphony Recommendations](#)
- [Adding Formulas](#)
- [List of Formulas](#)
- [Exponential Smoothing](#)



Trend Confirmation Indicator

This applies to: *Managed Dashboards, Managed Reports*

The **Trend Confirmation Indicator** function computes the ratio between a short-term moving average and a longer-term moving average of the stock price. Resulting values oscillate around the 100 level. Applications of TCI include generation and confirmation of buy and sell signals. For example, when the TCI advances above 100, this may be interpreted as a buy-signal.

Syntax

```
TRENDCONF(d0,s0,s1,Alignment)
```

Input

The Trend Confirmation Indicator function requires the following input series:

- d0 - Input data values - The set of data values for which the Trend Confirmation Indicator is calculated, usually the daily close price of a stock.

Parameters

The Trend Confirmation Indicator function has the following parameters:

- s0 - Short Period - The short time period for which the Trend Confirmation Indicator is calculated.
- s1 - Long Period - The long time period for which the Trend Confirmation Indicator is calculated.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Trend Confirmation Indicator function generates the following output:

- Trend Confirmation Indicator - The Trend Confirmation Indicator result set.



Ultimate Oscillator

This applies to: *Managed Dashboards, Managed Reports*

The Ultimate Oscillator function is based on the indicator developed by Larry Williams. It computes a weighted total of three oscillators, each of which is calculated using a different time period. Resulting values range from 0 to 100. Applications of Ultimate Oscillator include buy and sell signals, and providing indication of overbought or oversold conditions.

Syntax

```
ULTIMOSC (d0,d1,d2,s0,s1,s2,Alignment)
```

Input

The Ultimate Oscillator function requires the following input series:

- d0 - High data values - The first set of data values for which the Ultimate Oscillator is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Ultimate Oscillator is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Ultimate Oscillator is calculated, usually the daily close price of a stock.

Parameters

The Ultimate Oscillator function has the following parameter:

- s0 - Short Period - The short time period to use in the calculation. Default value is 7.
- s1 - Intermediate Period - The intermediate time period to use in the calculation. Default value is 14.
- s2 - Long Period - The long time period to use in the calculation. Default value is 28.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Ultimate Oscillator function generates the following output:

- Ultimate Oscillator - The Ultimate Oscillator result set, with values ranging from 0 to 100.

Using a Formula Visualization

This applies to: *Managed Dashboards, Managed Reports*

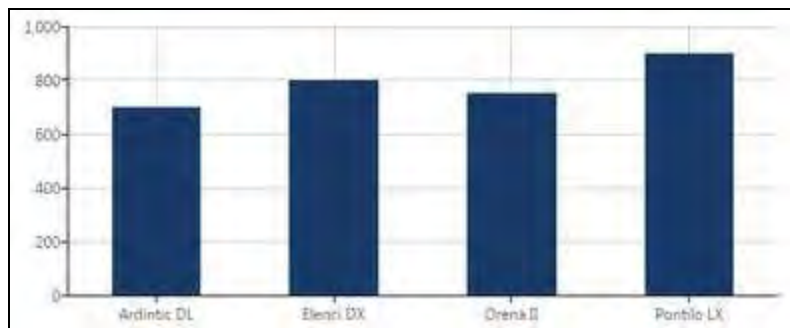
Formula visualizations allow you to use data from one or more visualizations to produce a new metric set (called a formula metric set) in a new visualization. These can be useful for a variety of analysis tasks, and can be created without leaving the editor of your dashboard or other view.

Related video: [Formulas](#)

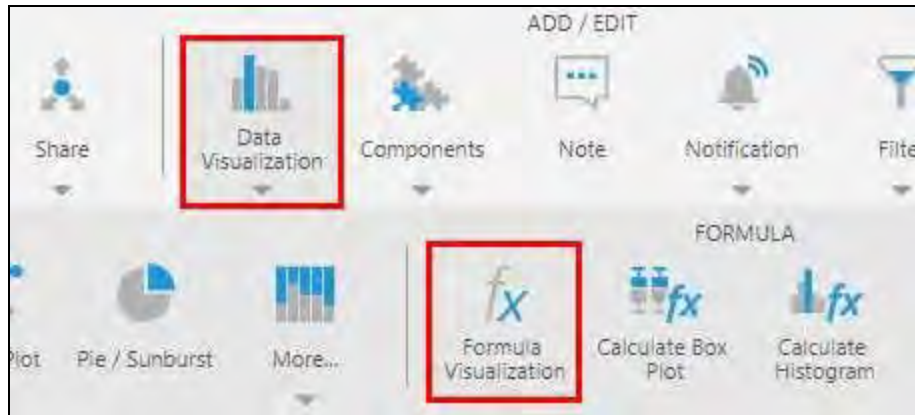
Creating a Formula Visualization

Formulas used to create a formula visualization are the same as formulas used when [adding a formula measure](#), except the result is displayed in a separate metric set.

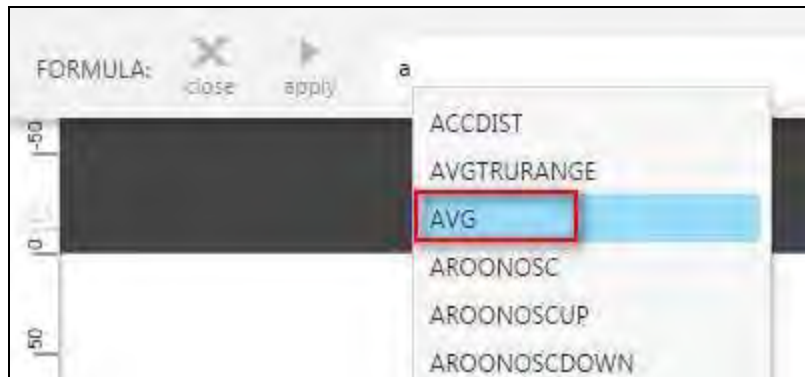
As a simple example, suppose you have a bar chart on a dashboard that shows the **List Price** of each product, and you want to display the **Average List Price** in a data label.



In the toolbar, click **Data Visualization** and select **Formula Visualization** under **Formula**.

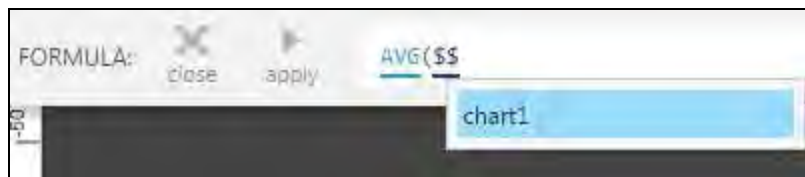


Enter a [formula](#) in the formula bar. In our example, we will type the first letter of the [formula function](#) AVG and select it from the list of available functions.

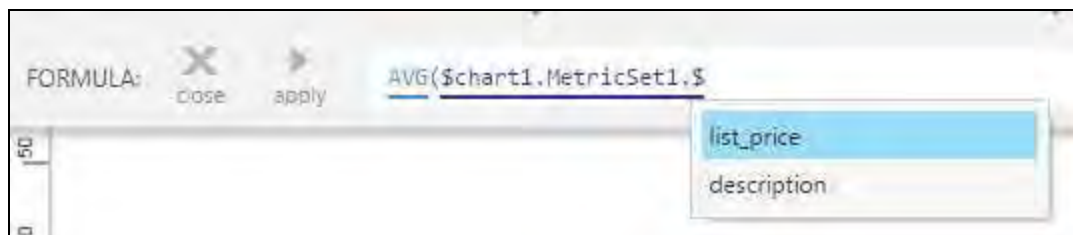


Click on data values in a visualization to insert that data into your formula. This can be from any visualization, and its placeholder in the formula will include its visualization and metric set names.

Alternatively, you can enter the placeholder for some data yourself, beginning with a dollar sign(\$), then choosing the visualization name.



Type a period character(.) and then from the menu, select the metric set from the list of metric sets on that visualization (in some cases there may be multiple).

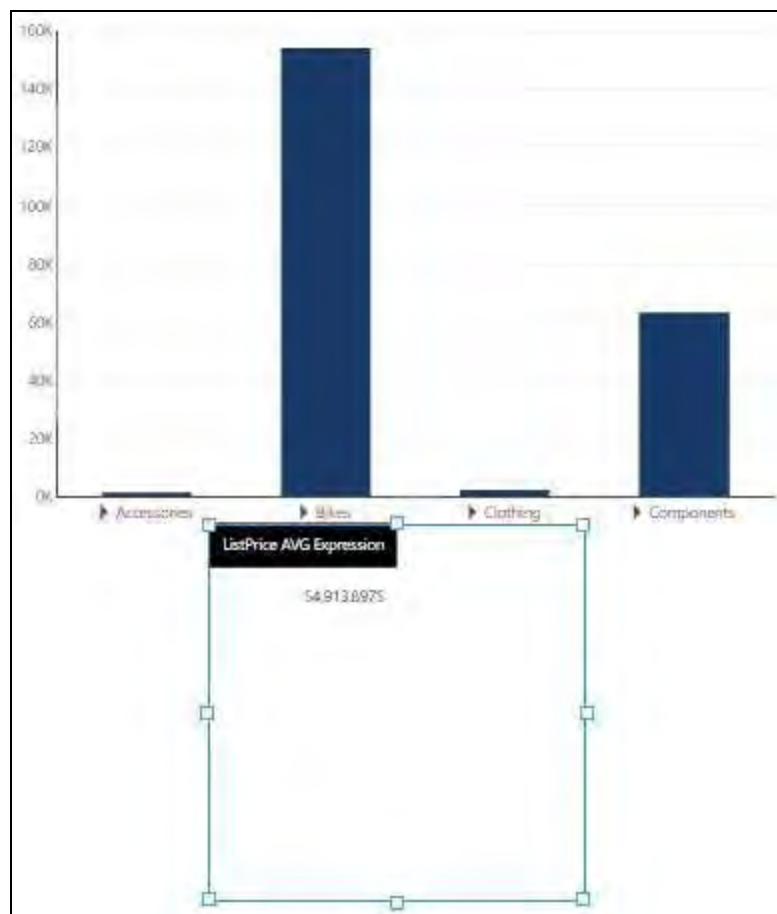


Type a closing parenthesis to complete the formula and press **Enter** or click **apply**.



The result of this formula is a new metric set displayed in a new visualization, which is initially a table.

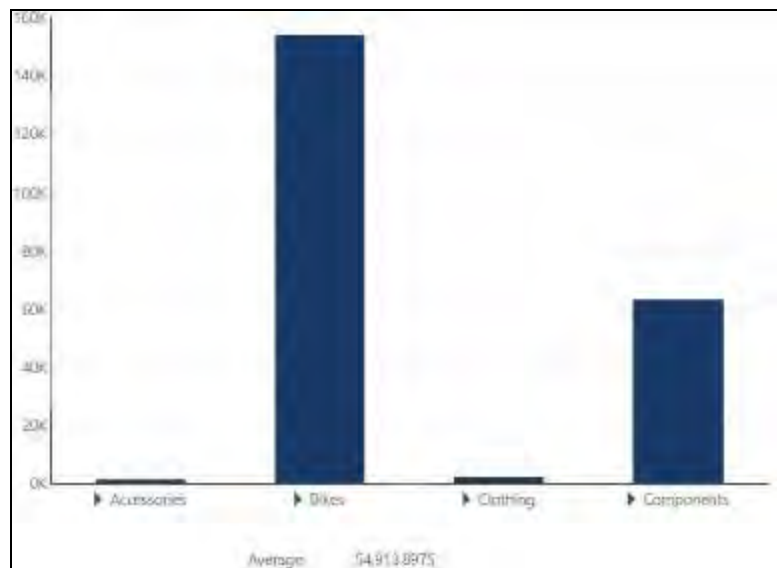
In our example, a single measure value in a single row is included, because that is all that is output by the AVG formula we entered.



You can choose to re-visualize this new metric set. In our case, with the table selected, we will choose **Re-Visualize** in the toolbar and then **Data Label**. You might then want to choose **Components** and then **Label**, and set its text to "Average:" to display to the left of the data label.



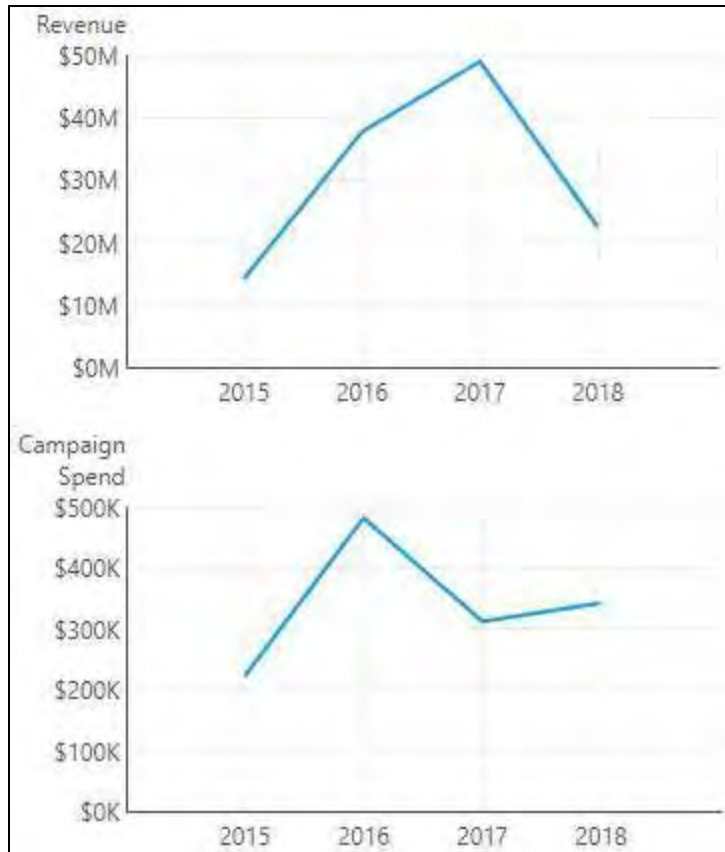
Note: Once the formula metric set has been created, it is safe to delete the original visualization from the canvas. The metric set it is based on will continue being referenced as needed. (If the original metric set was auto-generated, you will still be able to find it in the [Auto-Generated Items folder](#) under the view.)



Combining Data From Multiple Visualizations

Formula metric sets can also be created based on the data from multiple metric sets. Normally, this will only make sense when those metric sets are [using the same hierarchy](#) or a [time dimension hierarchy](#) - the rows will be matched up when possible, and the data combined.

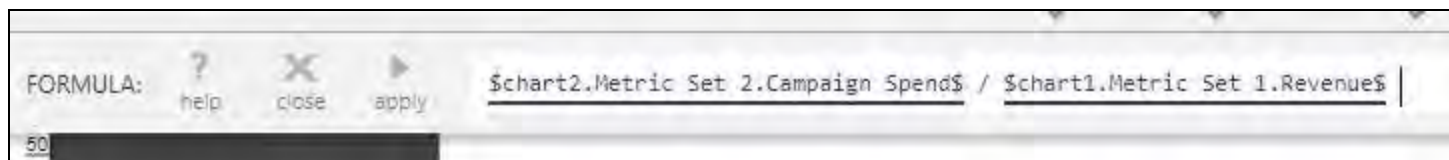
As an example, suppose we have two line charts on a dashboard: one displaying revenue over time from a database, and another with campaign spending over time from an Excel spreadsheet. We have dragged the same default time dimension hierarchy onto the dates in both charts.



Suppose we want to calculate what percentage of our revenue each year is spent on marketing campaigns.

Choose **Data Visualization** in the toolbar and then **Formula Visualization** to open the formula bar, and enter a formula that refers to a measure from each metric set to combine them.

In our example, we click the Campaign Spend line in the chart to insert it into the formula, type a forward slash (/) to [divide it](#), then click the Revenue line in the other chart to insert it as the divisor.



FORMULA: ? X ▶ help close apply \$chart2.Metric Set 2.Campaign Spend\$ / \$chart1.Metric Set 1.Revenue\$

Click **apply** to create the new formula metric set and visualization.

Data Analysis Panel - Metric Set

Unique Name:
Formula Metric Set
Description:
A metric set entity based on one or more measure expressions.

Formula Metric Set   SUM 

Campaign Spend   SUM 

Revenue   SUM 

click to add, or drop a measure

ROWS

Date  

click to add, or drop a hierarchy

SLICERS

click to add, or drop a hierarchy

COLUMNS

click to add, or drop a hierarchy

Year	Formula Metric Set	Campaign Spend	Revenue
Grand Total	\$0.05	\$1,357,532.00	\$123,216,786.12
2015	\$0.02	\$221,784.00	\$14,155,699.53
2016	\$0.01	\$481,941.00	\$37,675,700.31
2017	\$0.01	\$311,965.00	\$48,965,887.96
2018	\$0.02	\$341,842.00	\$22,419,498.32

Since we have calculated a percentage in our example, we will want to click to edit the formula measure in the Data Analysis Panel. We can change the **Caption** and also set its Format to P1, to [format it as a percentage with one decimal place](#).

Year	Percent	Campaign Spend	Revenue
Grand Total	5.0%	\$1,357,532.00	\$123,216,786.12
2015	1.6%	\$221,784.00	\$14,155,699.53
2016	1.3%	\$481,941.00	\$37,675,700.31
2017	0.6%	\$311,965.00	\$48,965,887.96
2018	1.5%	\$341,842.00	\$22,419,498.32

Grouping and Aggregating

Many formula functions have optional alignment arguments, which can change how the function groups data, and can result in multiple rows or columns of data instead of just one.

To make use of this ability, select a measure with multiple hierarchies. For example, this table lists products grouped by Color:

Color	Product	List Price
Grand Total		\$216,905.47
Black	(All)	\$67,436.26
	Chainring	\$0.00
	Full-Finger Gloves, L	\$37.99
	Full-Finger Gloves, M	\$37.99
	Full-Finger Gloves, S	\$37.99
	Half-Finger Gloves, L	\$24.49

While editing a view such as a dashboard displaying this metric set, choose **Data Visualization** in the toolbar and then **Formula Visualization** to open the formula bar.

After entering a function such as [AVG](#) or [SUM](#) and the input values, type a comma, then click a hierarchy value in a visualization or type in the hierarchy as the second, *alignment* argument, before the closing parenthesis:

The Mean function computes the mean (or simple average) of a set of input values.

AVG(d0,Alignments...)

d0: The set of data values for which the Mean is calculated.

(One or More, Optional) Alignments...: Hierarchy placeholders to be used as the alignment axis.

Check In
Undo
Clipboard
Share
Data Visualization

AVG(\$table1.Metric Set 1.ListPrice\$, \$table1.Metric Set 1.Color\$)

Color	Product	List Price
Grand Total		\$216,905.47

In our example, we are calculating the average list price for each color. Click apply to create the new formula metric set and visualization.

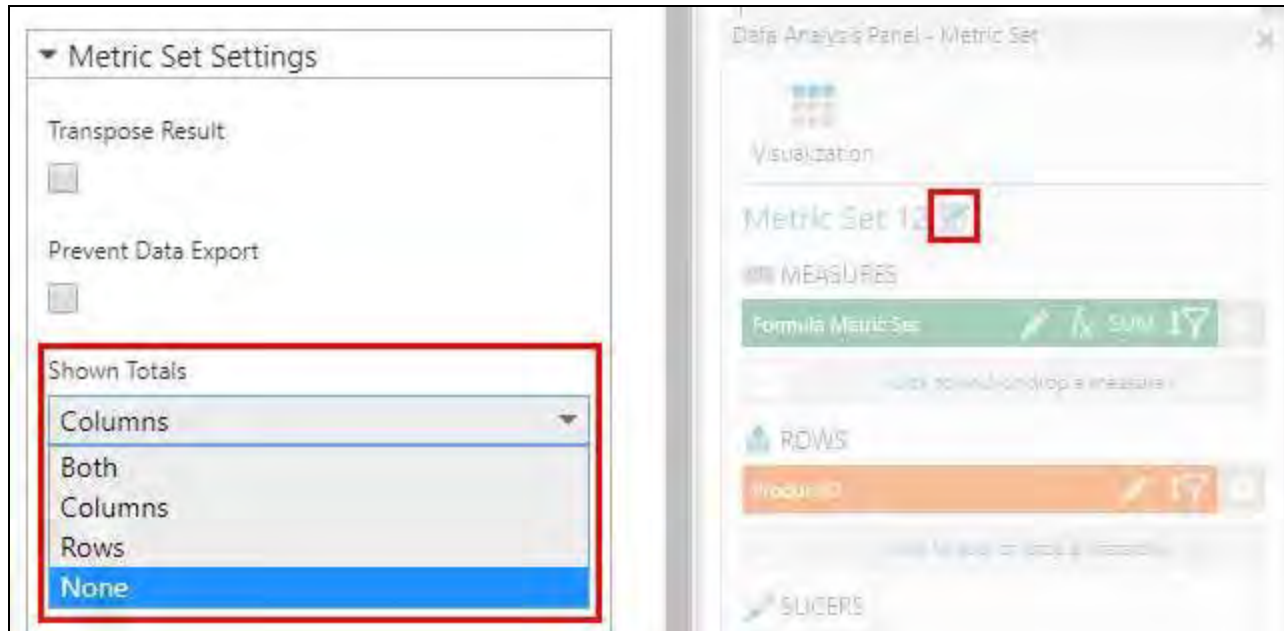
Color	Formula Metric Set
Grand Total	\$5,118.28
Black	\$725.12
Blue	\$923.68
Grey	\$125.00
Multi	\$59.87
Red	\$1,401.95
Silver	\$850.31

Another helpful function is the TOP or **maximum** subset function, which we can use with an alignment argument to provide the top (1) product for each color. Other [subset functions](#) can be used the same way, such as BOTTOM, FIRST and LAST.

TOP(\$table1.Metric Set 1.ListPrice\$, 1, \$table1.Metric Set 1.ProductID\$)

Color	Product	Formula Metric Set
Black	Mountain-100 Black, 48	\$3,374.99
Blue	Touring-1000 Blue, 60	\$2,384.07
Grey	Touring-Panniers, Large	\$125.00
Multi	Men's Bib-Shorts, S	\$89.99
Red	Road-150 Red, 62	\$3,578.27
Silver	Mountain-100 Silver, 48	\$3,399.99
Silver/Black	Touring Dadal	\$80.00

If totals are included that you don't want, you can disable them the same way as for any metric set: click the metric set's edit button in the Data Analysis Panel, then set **Shown Totals**.



Subset Functions

When used in formula visualizations, subset functions can be used to reduce the full set of data to a particular subset of its rows or columns. This isn't possible when just adding a new formula measure column to an existing metric set.

For example, the TOP function that we used above with an alignment argument can also be used to ask for the top 3 list price values:

`TOP($table1.Metric Set 1.ListPrice$, 3)`

Product Subcategory	List Price
Grand Total	\$219,655.59
▶ E-bike Shirts	\$329.97
▶ E-bike Pants	\$120.00
▶ E-bike Diapers	\$155.00
▶ E-bike and Socks	\$23.97
▶ E-bike Socks	\$25.70

Product Subcategory	Formula Metric Set
Grand Total	\$153,913.49
Road Bikes	\$68,690.35
Mountain Bikes	\$53,867.68
Touring Bikes	\$31,355.46

Because a formula visualization produces a completely new metric set and result, formulas can be used in this way to change the rows or columns that are displayed compared with the original metric set.

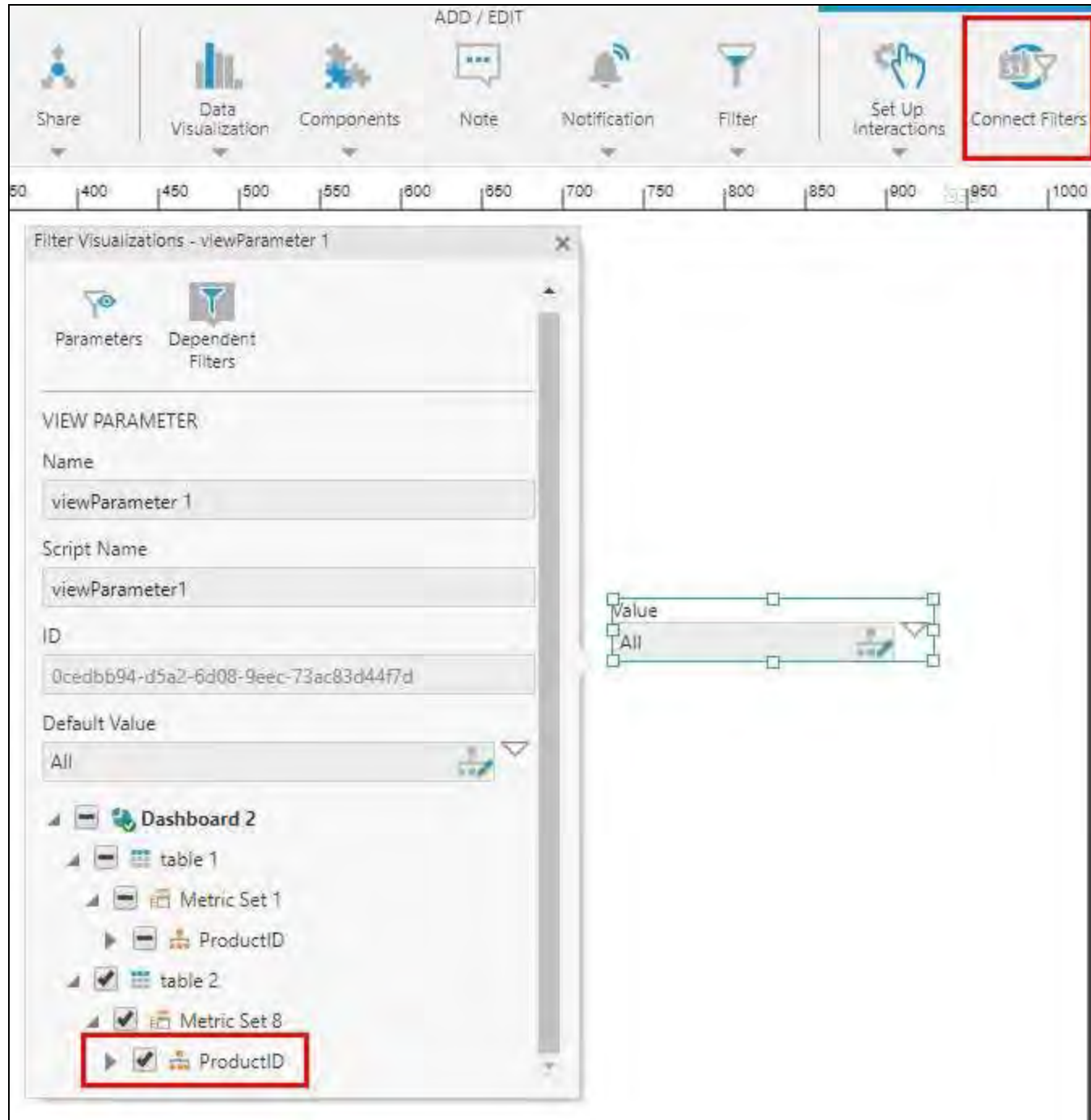


Note: When using formulas to select the top N items, a parameter placeholder could be used with a filter to allow viewers to change how many items are displayed. See [Adding Formulas](#) for details on parameter placeholders.

Filtering the Formula Metric Set

The result of creating a formula visualization is a separate metric set. This means it's safe to delete the original visualization if you don't need it; otherwise, the two metric sets now request their data independently, including their sorting and filtering. If you have placed filters on your dashboard canvas or in a [parameter bar](#), you will need to connect them to the new metric set for it to be affected.

To connect an existing filter on the canvas to the formula visualization, click to select it while editing, then choose **Connect Filters** in the toolbar. When adding a new filter, the same **Filter Visualizations** popup appears automatically:



The screenshot displays the Logi Symphony v24 interface. At the top, a toolbar contains several icons: Share, Data Visualization, Components, Note, Notification, Filter, Set Up Interactions, and **Connect Filters** (highlighted with a red box). Below the toolbar is a horizontal axis with numerical values from 50 to 1000.

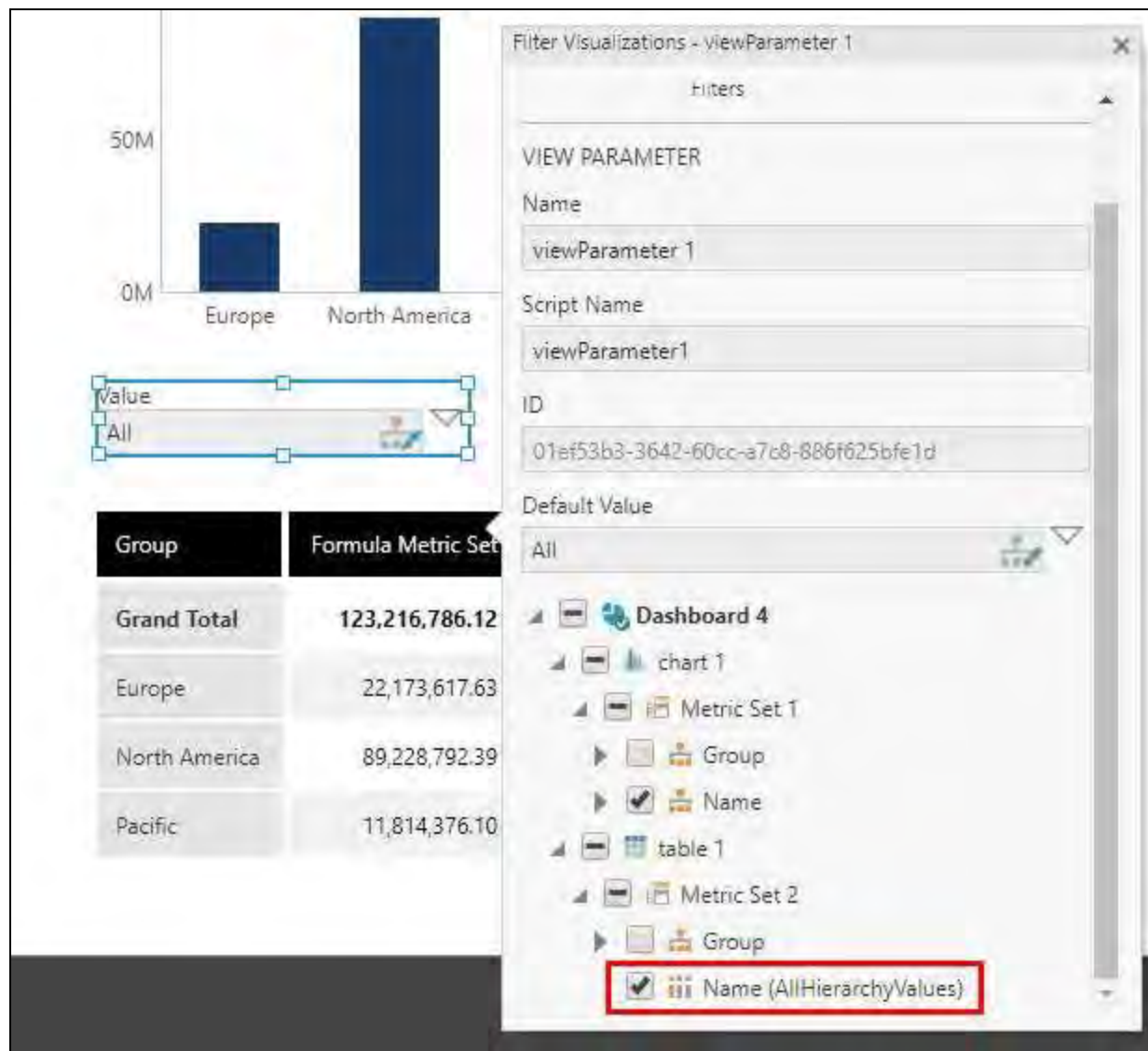
The main area features a panel titled "Filter Visualizations - viewParameter 1". This panel has two tabs: "Parameters" and "Dependent Filters". Under the "Parameters" tab, there are fields for "Name" (viewParameter 1), "Script Name" (viewParameter1), "ID" (0cedbb94-d5a2-6d08-9eec-73ac83d44f7d), and "Default Value" (All). Below these fields is a tree view showing the dashboard structure:

- Dashboard 2
 - table 1
 - Metric Set 1
 - ProductID
 - table 2
 - Metric Set 8
 - ProductID** (highlighted with a red box)

To the right of the tree view, there is a visual representation of a filter slider. It shows a horizontal bar with a "Value" label and a range from "All" to a specific value, with a small icon indicating the current selection.

Check the box next to the hierarchy under the new visualization and its metric set.

Slicers in the original metric set can be filtered using the parameter that appears under the formula metric set:



The screenshot displays the Logi Symphony interface. On the left, a bar chart shows values for 'Europe' and 'North America'. Below the chart is a table with the following data:

Group	Formula Metric Set
Grand Total	123,216,786.12
Europe	22,173,617.63
North America	89,228,792.39
Pacific	11,814,376.10

On the right, the 'Filter Visualizations - viewParameter 1' panel is open. It shows a tree view of the dashboard structure. The 'Name (AllHierarchyValues)' parameter is highlighted with a red box, indicating it is selected for filtering.

There may be other parameters as well for other hierarchies in the original metric set, and you can connect your filters to these parameters. If a parameter is not available, you can add the hierarchy to the formula metric set as described in the following section, and then its parameter will be available.



- Archive of documentation for Logi Symphony v24

If you are not using [filter controls](#) or your filters are in a parameters bar, you can also open the **Parameters** window and edit [a view parameter](#) to connect it to the formula metric set the same way as shown above.

Changing the Formula Metric Set

When a formula visualization is created, its new metric set initially has certain measures or hierarchies based on the particular formula you entered. This determines how the data is grouped, as well as which information can be displayed within the new visualization.

If you want to change which hierarchies and measures are included, you can do so. This does not affect the original metric set, but only how this formula metric set groups and displays its result.

Removing Elements

You can remove any element from the Data Analysis Panel that you don't want (including the measure containing the original formula).

Data Analysis Panel - Metric Set

Visualization

Metric Set 12

MEASURES

Formula Metric Set

List Price

click to add, or drop a measure

ROWS

Color

ProductID

click to add, or drop a hierarchy

SLICERS

click to add, or drop a hierarchy

COLUMNS

click to add, or drop a hierarchy

Color	Product Subcategory	Formula Metric Set	List Price
Grand Total		\$219,655.59	\$219,655.59
Black	(All)	\$67,436.26	\$67,436.26
	Cranksets	\$836.97	\$836.97
	Gloves	\$187.44	\$187.44
	Helmets	\$34.99	\$34.99
	Mountain Bikes	\$27,404.84	\$27,404.84
	Mountain Frames	\$9,391.99	\$9,391.99
	Road Bikes	\$16,631.30	\$16,631.30
	Road Frames	\$9,180.82	\$9,180.82
	Shorts	\$449.93	\$449.93
	Tights	\$224.97	\$224.97
	Wheels	\$3,093.01	\$3,093.01

When a hierarchy is removed, this affects how the result is grouped into rows or columns, and the remaining data is re-aggregated accordingly.

Data Analysis Panel - Metric Set

Visualization

Metric Set 12

MEASURES

Formula Metric Set ✎ fx SUM ↑ ↓ ⊙

+ [click to add](#) or drop a measure

ROWS

Color ✎ ↑ ↓ ⊙

+ [click to add](#) or drop a hierarchy

SLICERS

+ [click to add](#) or drop a hierarchy

COLUMNS

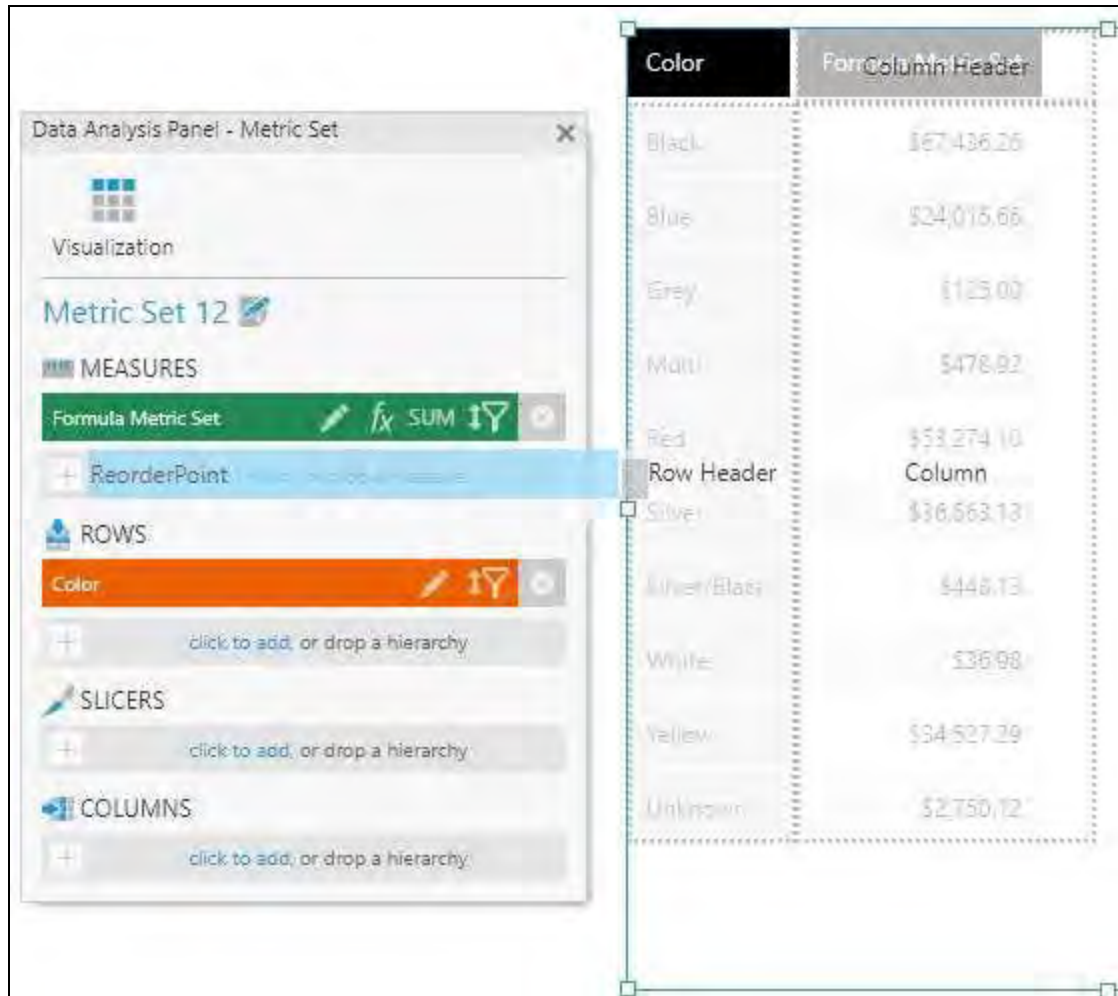
+ [click to add](#) or drop a hierarchy

Color	Formula Metric Set
Black	\$67,436.26
Blue	\$24,015.66
Grey	\$125.00
Multi	\$478.92
Red	\$53,274.10
Silver	\$36,563.13
Silver/Black	\$448.13
White	\$36.98
Yellow	\$34,527.29
Unknown	\$2,750.12

Adding Elements

You can add any elements that are missing from the formula metric set, as long as they were already selected in the original metric set(s).

While you can't use the **click to add** links, you can drag the same elements from the Explore window that were added to the original metric set, and drop them onto the Data Analysis Panel of the new formula metric set. They must be in the same place they were added in the original metric set (e.g., under Rows, if originally added there).



The screenshot displays the Logi Symphony interface. On the left, the 'Data Analysis Panel - Metric Set' is open, showing a 'Metric Set 12' with a 'Formula Metric Set' containing the formula 'SUM 1Y'. Below this, there are sections for 'ROWS' (with 'Color' selected) and 'COLUMNS'. The main area on the right shows a table with two columns: 'Color' and 'Form Column Header'. The table contains data for various colors and their corresponding values.

Color	Form Column Header
Black	\$67,436.25
Blue	\$24,015.66
Grey	\$125.00
Multi	\$478.92
Red	\$53,274.10
Row Header	Column
Silver	\$36,563.13
Silver/Black	\$448.13
White	\$36.98
Yellow	\$34,527.29
Unknown	\$2,750.12

If you need to add an element that is not already present in the original metric set, first add it to the original metric set. You will then be able to drag it onto the formula metric set, even if you created it earlier.

Advanced Options

When using multiple metric sets in your formula, there are settings available that determine how the data is combined.

To access these settings, click the fix button of a measure in your formula metric set, and click any of the underlined [formula functions](#) (If you haven't used one of these functions yet, add a [new formula measure](#) to the formula metric set that uses one (you can remove it later).



In the formula dialog that appears, there are two options at the bottom:

- **Alignment Behavior** determines whether the rows of data in each metric set are matched up by their hierarchy's values (Hierarchical or Auto), or are combined arbitrarily by sequence (Positional).
- **Enforce Hierarchy Cube Check** is disabled by default. When un-checked and metric sets select data from different data cubes, hierarchies with the same unique name are assumed to be the same, allowing their rows of data to be matched up. If hierarchies should not be considered the same based only on their names for the data cubes you are using, you can check this option.

Limitations

Formula visualizations have some limitations compared with the original metric sets:

- Like other formulas, they can't be created based on a metric set in raw data mode. Remove the **<Row Number>** element from **Rows** in the metric set's Data Analysis Panel before creating a formula visualization.
- Period over period measures can't be added to a formula metric set. They can instead be added to the original metric before creating the formula visualization.
- Formula metric sets don't support expanding or collapsing individual members/rows.
- The level of a dynamic hierarchy in a formula metric set should be set to the same level as in the original metric set.

Variance

This applies to: *Managed Dashboards, Managed Reports*

The **Variance** function calculates the volatility of a set of input data values, such as stock prices over a period of time.

VARIANCE (\$Close\$)		
Date Values	Close	Script Measure
08/14/2014 00:00:00	36.36	0.34
08/15/2014 00:00:00	36.47	0.34
08/18/2014 00:00:00	37.38	0.34
08/19/2014 00:00:00	37.83	0.34
08/20/2014 00:00:00	37.50	0.34
08/21/2014 00:00:00	37.64	0.34
08/22/2014 00:00:00	38.01	0.34
08/25/2014 00:00:00	37.71	0.34
08/26/2014 00:00:00	37.79	0.34
08/27/2014 00:00:00	38.18	0.34

Syntax

```
VARIANCE(d0,Alignments...)
```

Input

The Variance function requires the following input:

- d0 - Input data values - The set of data values for which the Variance is calculated.

Parameters

The Variance function requires the following parameters:

- Alignments (One or More, Optional) - Hierarchy placeholders to be used as the alignment axis.

Output

The Variance function generates the following output:

- Variance - A single value representing the variance of the input values.

Volume Oscillator

This applies to: *Managed Dashboards, Managed Reports*

The Volume Oscillator function computes the relationship between a short-term moving average and a long-term moving average of the stock volume. Resulting values oscillate around the zero line. Applications of Volume Oscillator include indication of volume surges, which may be due to panic selling or greedy buying.

Syntax

Volume Oscillator:

```
VOLUMEOSC(d0,s0,s1,s2,Alignment)
```

Histogram:

```
VOLUMEOSCHIST(d0,s0,s1,s2,Alignment)
```

Signal Line:

```
VOLUMEOSCSIG(d0,s0,s1,s2,Alignment)
```

Input

The Volume Oscillator functions require the following input series:

- d0 - Input data values - The set of data values for which the Volume Oscillator is calculated, usually the daily volume of a stock.

Parameters

The Volume Oscillator function has the following parameters:

- s0 - Short Period - The short time period to use in the calculation. Default value is 7.
- s1 - Long Period - The long time period to use in the calculation. Default value is 14.



- s2 - Signal Period - The period of time to use in the calculation of the signal line. Default value is 7.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The VOLUMEOSC function generates the following output:

- Volume Oscillator - The Volume Oscillator result set. Values oscillate about the zero line.

The VOLUMEOSCHIST function generates the following output:

- Histogram - The difference between the Volume Oscillator and the Signal Line.

The VOLUMEOSCHIST function generates the following output:

- Signal Line - A smoothing out of the Volume Oscillator that is created by applying an exponential moving average.

Weighted Moving Average

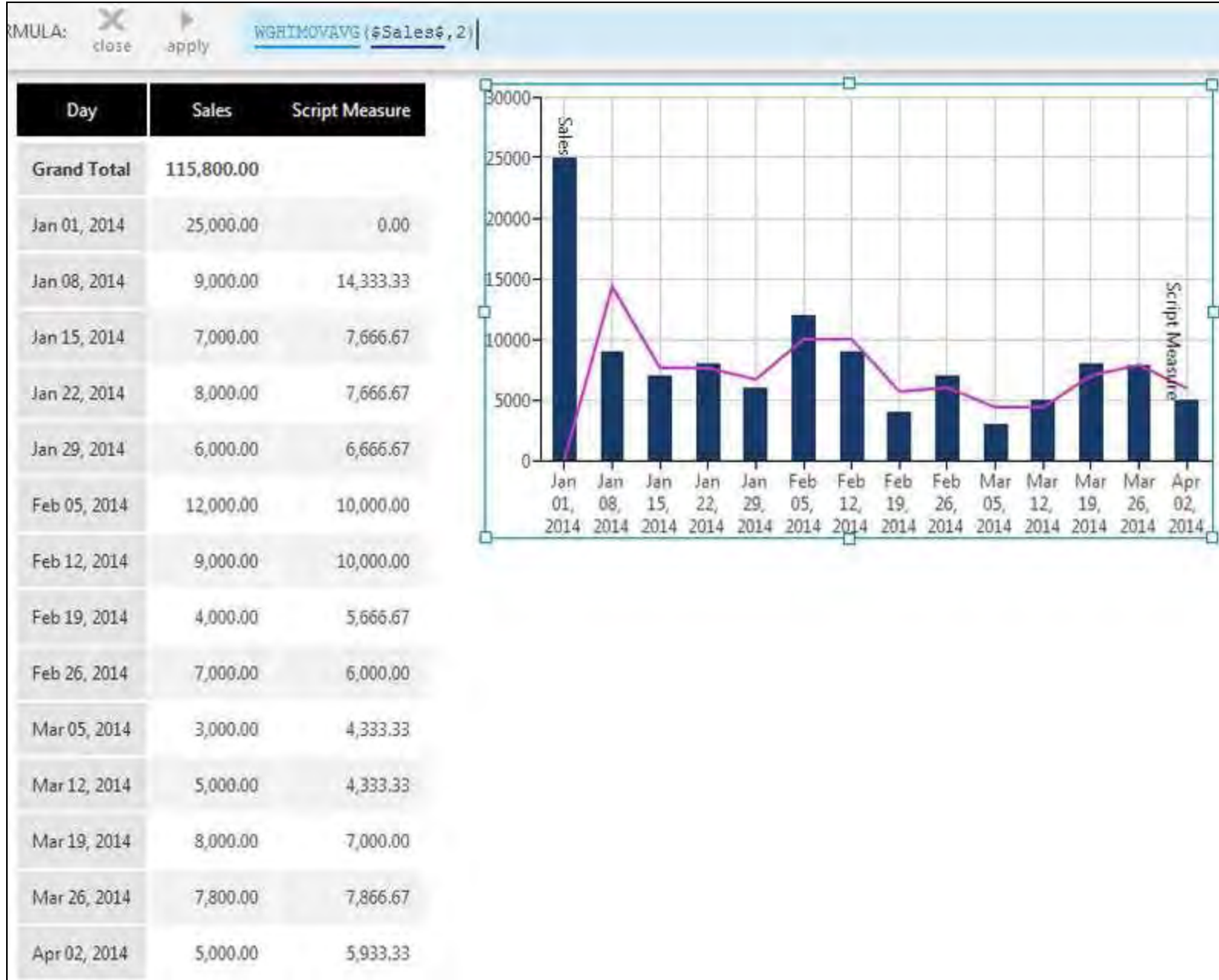
This applies to: *Managed Dashboards, Managed Reports*

The **Weighted Moving Average** (WMA) function computes the average of a set of input values over a specified number of time periods. In this function, a greater weight is given to more recent data. The function can be used to smooth a data series, which helps to reduce noise and make it easier to spot data trends.

The mathematical formula being calculated is as follows:

$$WMA_t = (n * Y_t / K) + ((n - 1) * Y_{t-1} / K)$$

Where Y is the value, n is the number of periods, and K is the sum of n-based multipliers (for example, for n=3, K=3+2+1=6).



Syntax

```
WGHTMOVAVG (d0,s0,Alignment)
```

Input

The Weighted Moving Average function requires the following input:

- d0 - Input data values - The set of data values for which the Weighted Moving Average is calculated.

Parameters

The Weighted Moving Average function requires the following parameters:

- s0 Period - The number of time periods to use in the calculation. The default value is 10.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Weighted Moving Average function generates the following output:

- Weighted Moving Average - The Weighted Moving Average result set.

For more information, see:

- [Moving Average](#)
- [Exponential Moving Average](#)

Williams %R (or Percent Range)

This applies to: *Managed Dashboards, Managed Reports*

The Williams %R (or Percent Range) function is based on the indicator developed by Larry Williams. It is similar to the Stochastic Oscillator function in that it relates the close price of a stock to its price range (high-low) over a specified number of time periods. Applications of Williams %R include indication of price reversals and determining overbought or oversold conditions.

Syntax

```
WILLIAMSR(d0,d1,d2,s0,Alignment)
```

Input

The Williams %R function requires the following input series:

- d0 - High data values - The first set of data values for which the Williams %R formula is calculated, usually the daily high price of a stock.
- d1 - Low data values - The second set of data values for which the Williams %R formula is calculated, usually the daily low price of a stock.
- d2 - Close data values - The third set of data values for which the Williams %R formula is calculated, usually the daily close price of a stock.

Parameters

The Williams %R function has the following parameter:

- s0 - Period - The number of time periods used to determine the overall high and low price values.
- Alignment (Optional) - Hierarchy placeholder to be used as the alignment axis.

Output

The Williams %R function generates the following output:

- Williams %R - The Williams %R result set.

Refresh Data

When data changes frequently, you can set up visualizations to automatically update at a regular interval to keep them open and monitor them. When using a data cube storage option, you may need the cube to update to include the latest changes from the original data source.

This article describes ways you can update and auto-refresh the data displayed.

Refreshing a Data Cube

Data cubes and other data displayed are not warehoused or stored in-memory by default, which means they retrieve data directly from the data source except for some temporary caching.

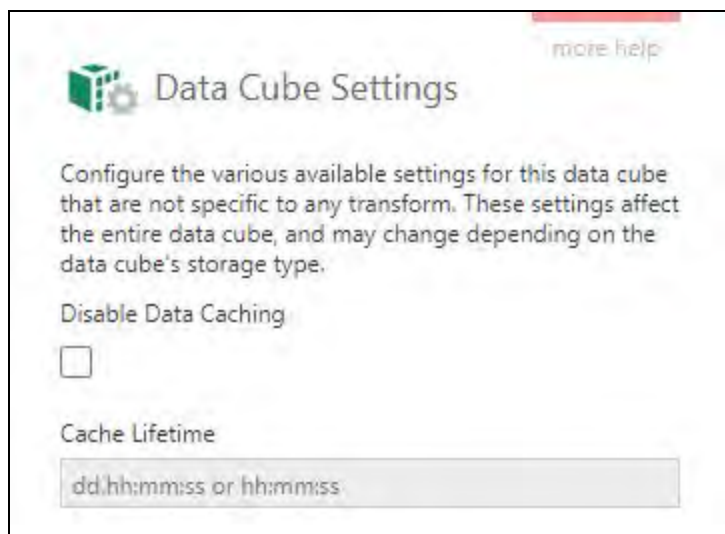
If the data cube's storage type has been set to warehouse or in-memory, it can be updated on-demand or on a schedule.

Change Data Caching

For data cubes that are not using warehouse or in-memory storage, data can still be temporarily cached, normally for a short time according to the application's result caching settings.

You can customize data caching for all metric sets and visualizations displaying data from a particular data cube. Alternatively, you can also bypass data caching on individual visualizations as shown later in this article.

Click **Data Cube Settings** in the toolbar when editing a data cube, then you can change the **Cache Lifetime** for the data retrieved from this cube. You can also choose the **Disable Data Caching** option.



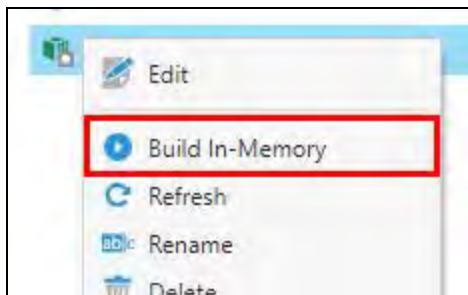
The screenshot shows a dialog box titled "Data Cube Settings" with a "more help" link in the top right corner. Below the title is a green cube icon. The main text reads: "Configure the various available settings for this data cube that are not specific to any transform. These settings affect the entire data cube, and may change depending on the data cube's storage type." There are two settings: "Disable Data Caching" with an unchecked checkbox, and "Cache Lifetime" with a text input field containing the placeholder "dd.hh:mm:ss or hh:mm:ss".



Important: Use caution when disabling caching completely, as the data retrieval process can be slower depending on the data source and the data cube's transforms. It also means that scrolling tables or zoomed-in charts may result in querying the data source again rather than using data cached from when first loading.

Re-Build Manually

Data cubes using warehouse or in-memory storage can be re-built on-demand. Find the data cube by choosing **Open** in the main menu, or in the Explore window. Right-click the data cube and choose **Build Warehouse** or **Build In-Memory**.



Schedule Warehouse and In-Memory Builds

A data cube that is stored in the warehouse or in-memory can be configured to refresh itself automatically. After the data cube is checked in, you can [set up a schedule rule](#).

Rebuild Using the API

If the data cube is warehoused or stored in-memory, you can trigger the data cube build using the API. For example, this script can be added to a button's click actions:

```
// Get the data cube service
var dataCubeService = this.getService("DataCubeService");

// Build the data cube with the specified ID
dataCubeService.buildStorage("4b22e502-6dac-4227-bc93-1cdb8491dec6");
```

See [File and folder properties](#) for details on finding the data cube's ID.

Refreshing a Data Visualization

A data visualization can be refreshed manually or automatically. To always retrieve 'live' data including the latest changes from the data source, you can also bypass or customize result caching.

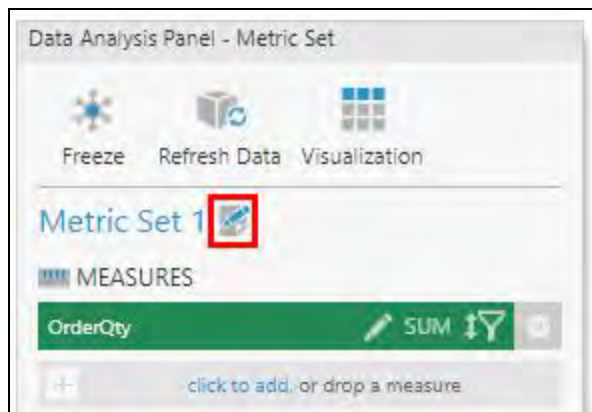
Important

When the underlying data cube is using warehouse or in- memory storage, you can only refresh a visualization with data from the last cube build.

Bypass Data Cache

Data results are cached by default when data is retrieved again a short time later for the same metric set, filtering, and other settings. To display live or real-time data with more frequent updates than the result cache, you can bypass the data cache for a visualization.

In the *Data Analysis Panel*, click the **Edit** icon.



In the **Request Options** section, select the **Bypass Data Cache** option.



Metric Set Configuration

This is where you can change how we retrieve data for this metric set.

Metric Set Full Name

Metric Set ID

Analysis Structure Full Name

▶ Metric Set Settings

▼ Request Options

Bypass Data Cache

☒

Include Notes

☒

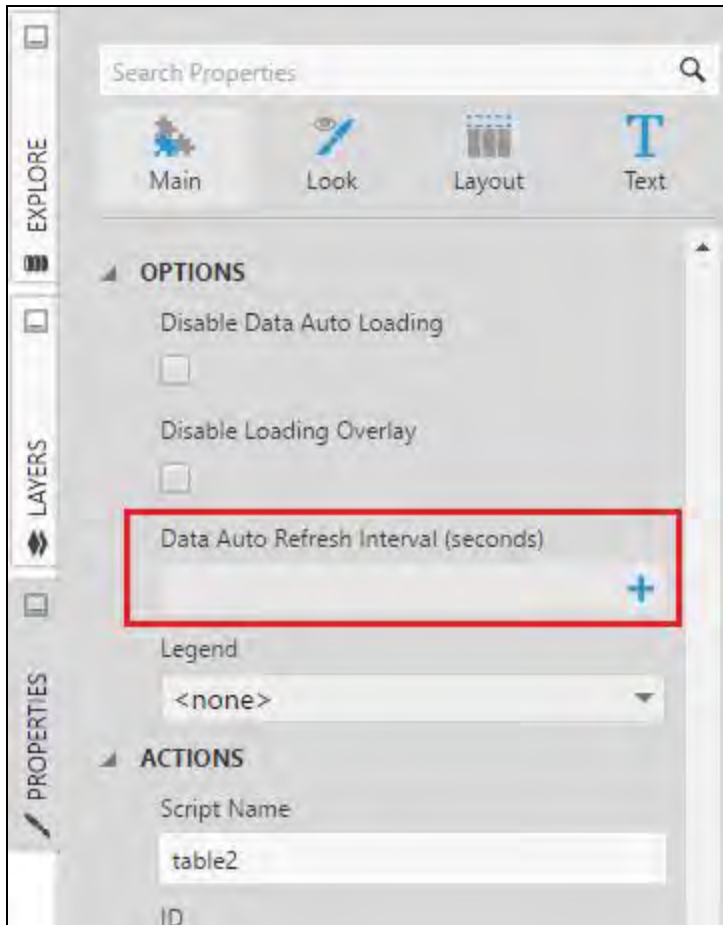


Note: Caching may already have been disabled or customized in the data cube, which affects all visualizations displaying its data regardless of their *bypass* setting. Use this *bypass* option rather than disabling caching in the data cube to allow cached data to be retrieved when scrolling tables or charts.

Auto Refresh Interval

Every visualization has the property **Data Auto Refresh Interval**. When this is set, the visualization will automatically update its data regularly at the specified interval for as long as it's being viewed.

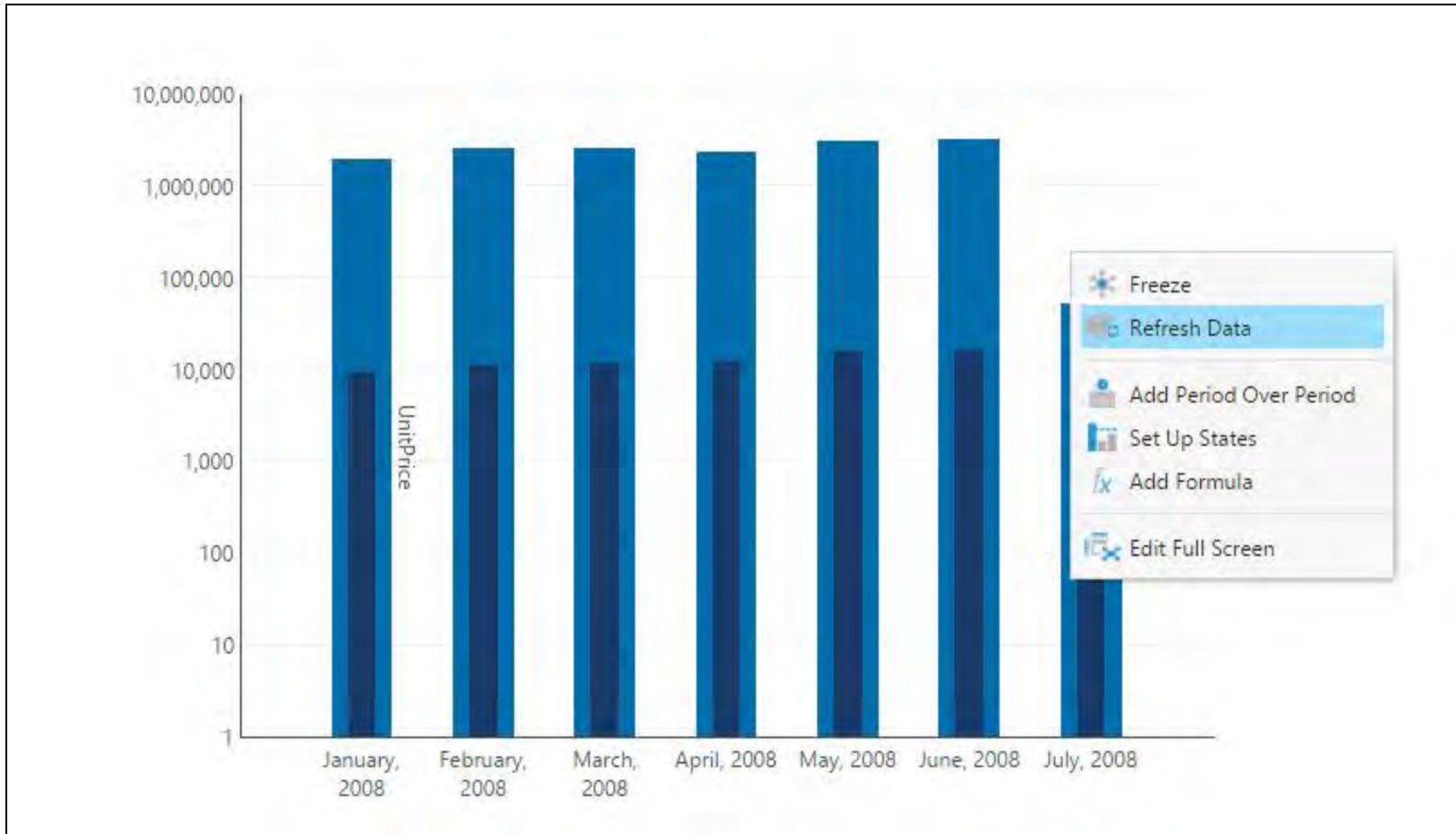
You can find this property in the **Main** tab of the visualization's properties.



Manual Refresh

In [View mode](#), the data visualization can be refreshed manually whenever you want to see the latest data. To do this, right-click (or long-tap) on the visualization to bring up the context menu.

Select **Data Tools** and choose **Refresh Data**.



Result Cache Configuration

Rather than changing or bypassing caching as described above, you can choose to change the result caching for the entire application by changing the *Result Cache Absolute Expiration* [configuration setting](#) in Administration.

You can change this to a shorter length of time or set to zero to disable caching. Setting to zero is not recommended and should generally not be done in production environments.



Configure Result Cache Absolute Expiration

The approximate time interval for which the in-memory result cache is allowed to store a new result. A value of zero indicates that the cache should be disabled. (e.g., dd.hh:mm:ss or hh:mm:ss)

☒ Reset to default
☐ Edit value

00:20:00

more help

GENERAL

Refresh Advanced Settings Configured Settings Edit Details

cache x Setting Scope Global (A)

Category	Name	Value
Caching	Hierarchy Cache Absolute Expiration	00:00:00
Caching	Hierarchy Cache Size	1000
Caching	Result Cache Absolute Expiration	00:20:00
Caching	Result Cache Size	1000

For more information, see:

- [Data Cube Data Storage Options](#)
- JavaScript API: [DataCubeService](#)
- [Configuration Settings](#)

Zig Zag Subset

This applies to: *Managed Dashboards, Managed Reports*

The Zig Zag Subset function returns the input data values that show movement greater than the specified sensitivity percentage value.

Syntax

```
ZIGZAG(d0,s0,Alignment)
```

Input

The Zig Zag Subset function requires the following input:

- d0 - Input data values - The set of data values for which the Zig Zag Subset is calculated.

Parameters

The Zig Zag Subset function requires the following parameters:

- s0 - Sensitivity - The Sensitivity parameter is a percentage that is used to filter out smaller data movements. Default value is 10.
- Alignment (Optional) - A hierarchy placeholder to be used as the alignment axis. If an alignment hierarchy is not indicated, the subset is applied to the whole result.

Output

The Zig Zag Subset function generates the following output:

- Zig Zag Subset - The Zig Zag Subset result set.



Writing Scripts Using Browser Developer Tools

This applies to: *Managed Dashboards, Managed Reports*

When writing script in dashboards, reports, and other views, you may find it helpful to use your web browser's developer tools when testing and debugging. This article shows you how to use Google Chrome's Developer Tools in conjunction with the script editor.

Symphony provides a built-in [script editor](#) that pops up with a list of each object's properties and their descriptions as you type, with direct links available to the applicable API documentation. The [JavaScript API reference](#) lists and describes all supported API properties and methods, and allows you to easily navigate between different types of objects. The browser's developer tools complements the other tools by allowing you to step through your script while it's running live, and inspect the values of properties and variables.

Debug using Chrome Developer Tools

To debug script using Chrome's Developer Tools, simply insert the following line on its own at the point in your script where you want to start debugging:

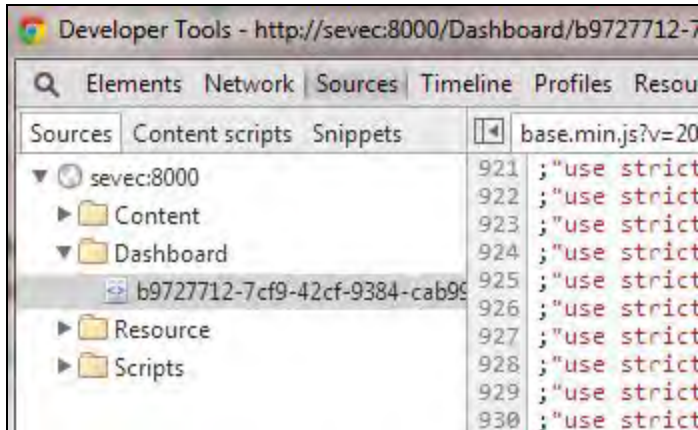
```
debugger;
```

For example, insert it at the beginning as the first line. This is a statement that various browsers recognize when their developer tools window is open. When the statement is reached during script execution, the browser's developer tools window becomes the active window and it automatically pauses at this line, allowing you to debug the rest of the script.

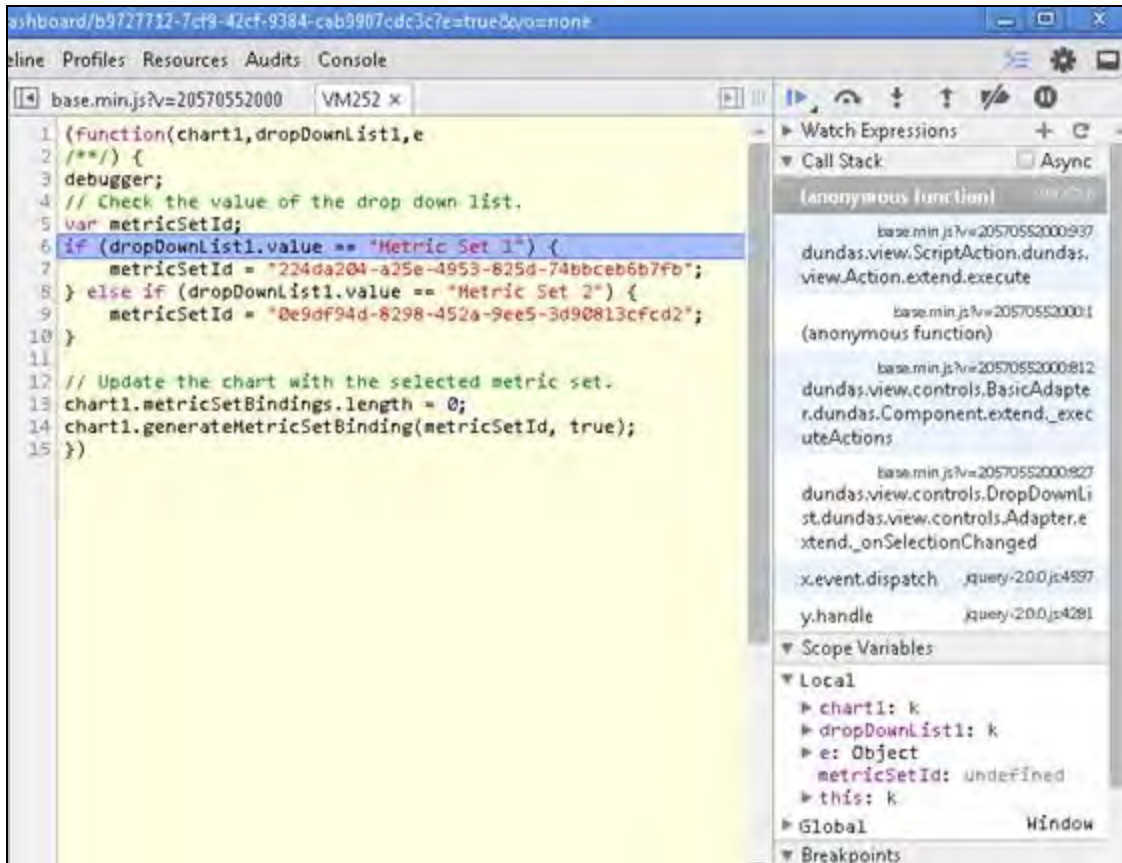
Next, click **Build**.

In the toolbar, choose **Sandbox View** in order to test the script in a separate tab, without saving the changes it makes as part of the dashboard.

Next, launch Chrome Developer Tools from the Chrome menu in the top-right corner, or press CTRL+SHIFT+I. This opens the developer tools in a separate window.



Now perform the action that will trigger your script. When script execution reaches the debugger statement, the browser's developer tools will become the active window and you can begin debugging.



For example, you can:

- Hover over a variable or property with the mouse to see its current value
- Write expressions or statements into the Console to evaluate it and/or see its results
- Step over each line one-at-a-time to see where values are changing, by choosing this option in the toolbar in the top-right, or pressing F10
- Add a Watch Expression that will update automatically as you step through lines of script

Once you've finished debugging your script, go to the toolbar in the top-right corner of the developer tools and click **Resume script execution**, or press F8.



While you can name controls in any way you would like, JavaScript does not allow leading digits in function parameter names. To avoid errors in certain scripts, Symphony prefixes an underscore (_) when the adapter name starts with a number. This means, if the parameter name is 1Label, you will have to use _1Label to target it in script.

For more information, see:

- [Using the Script Editor](#)
- [API Reference](#)
- [White Labeling the Application](#)

Watermarking Sanctioned/Non-Sanctioned Data

This applies to: *Managed Dashboards, Managed Reports*

A requirement that may be associated with data governance is the ability to watermark content based on approved/non-approved data, for example when power users are doing self-service analysis with IT-certified data as well as personal data files, and sharing it with others.

With the provided open APIs, you can use script to watermark datasets and content based on whether or not sanctioned data sources are used. This article provides a sample of how the visualizations in a dashboard can be watermarked to certify data using JavaScript and CSS overrides.

Prerequisite

This sample requires basic familiarity with JavaScript and CSS.

Organizing Sanctioned Data

As a best practice, sanctioned or company approved data should reside in a repository that can be controlled by an administrator. Any metric set that is based on said data can then be watermarked when they are created using the certified data sources or [data model](#) in the repository.

Symphony provides users with a **Global project** where content that is meant to be shared across all projects would reside. Every project other than Global project has a **Shared** folder for project items such as data connectors, data cubes, metric sets etc. Any item created in the Global project will be accessible from the Shared folder of any other project. This ensures that the data is consistent and valid across all dashboards and reports built regardless of the projects.

The sample below has a dashboard for ad-hoc analysis created by a power user. Any metric sets the analyst uses from the **Shared** folders will be watermarked as certified official data, however any metric sets that the analyst creates on the fly for analysis will not be watermarked.

Dashboard Setup

Open a dashboard ready for sharing in **Edit** mode.

Add the following script to the **Ready** actions of the dashboard. This script gets all the metric sets on the dashboard and adds a watermark to them.



Note: This script can be modified to change the watermark content, format etc.

```
var metricSetService = this.getService('MetricSetService');
var baseViewService = this.getService('BaseViewService');

// Get all the adapters of the dashboard:
baseViewService.currentView.control.adapters.forEach(function (adapter) {
```



```

if (adapter.metricSetBindings) {
  adapter.metricSetBindings.forEach(function (metricSetBinding) {
    var metricSetDef = metricSetService.getMetricSetById(metricSetBinding.metricSetId);
    $.when(metricSetDef).done(function (metricSet) {
      // If the metric set belongs to the specified project:
      if (metricSet.projectId == 'a3a952c9-5092-444b-bc9a-28e1d013ffa4'){
        var rect = dundas.Utility.getAdapterBoundingClientRect(adapter.id);
        var watermark = document.createElement('div');
        $(watermark).text('Sanctioned Data');
        $(adapter.control.container).append(watermark);
        $(watermark).css('border', '#ff7f7f solid 2px');
        $(watermark).css('position', 'absolute');
        $(watermark).css('font-size', '13px');
        $(watermark).css('font-family', 'Arial');
        $(watermark).css('color', '#A9A9A9');
        $(watermark).css({ left: (rect.width / 3), top: (rect.height / 3) });

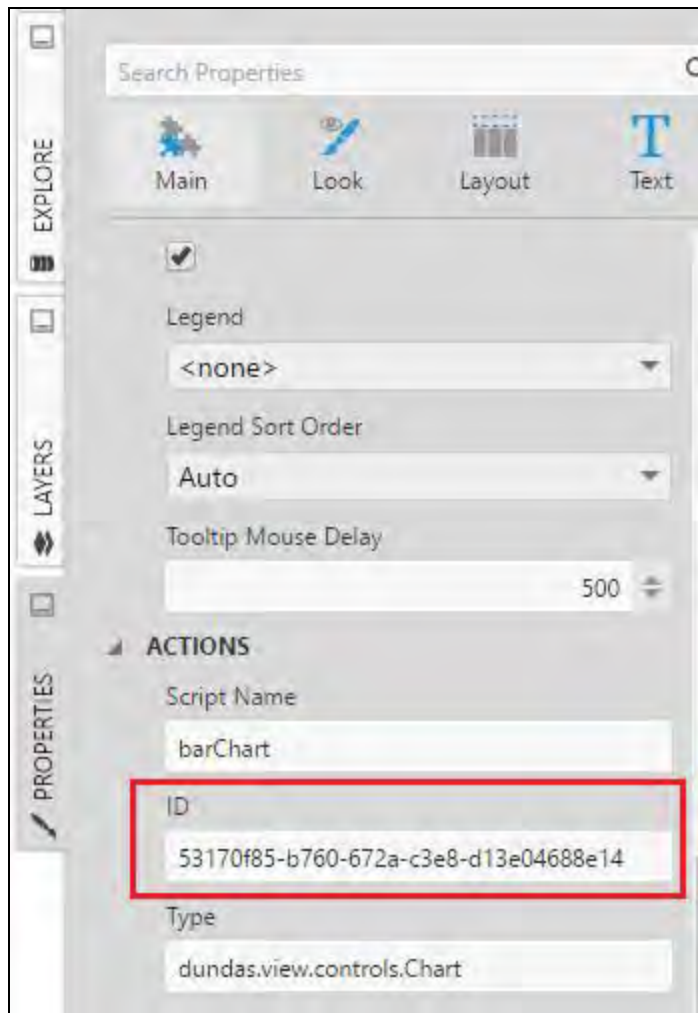
        var watermarkBackground = document.createElement('div');
        $(watermarkBackground).css('background', '#FF0000');
        $(watermarkBackground).css('opacity', 0.5);
        $(watermark).append(watermarkBackground);
        $(watermark).css('z-Index', 9999999999999999);

      }
    });
  });
}
});

```

To add the watermark on only **specific visualizations** regardless of which project they belong to, or to add different watermark comment on different visualizations, you can fetch a metric set by its visualization ID instead of getting all the metric sets of the project as in the script above.

To get the ID of the visualization, click on it and go to the **Properties** window. In the **Main** tab, find **ID** under **Actions**. This is the visualization *adapter* ID (adapters plug controls such as visualizations into Symphony). Copy and paste this adapter ID into the script below:



```
var metricSetService = this.getService('MetricSetService');
var baseViewService = this.getService('BaseViewService');

// Get the metric set to be watermarked by its visualization adapter ID
var adapter = baseViewService.currentView.control.getAdapterById("60f85a18-8e65-c287-e813-ec311156ddd4");
adapter.metricSetBindings.forEach(function (metricSetBinding) {
    var metricSetDef = metricSetService.getMetricSetById(metricSetBinding.metricSetId);
    $.when(metricSetDef).done(function (metricSet) {
```

```
var rect = dundas.Utility.getAdapterBoundingClientRect(adapter.id);
var watermark = document.createElement('div');
$(watermark).text('Non-Official Data');
$(adapter.control.container).append(watermark);
$(watermark).css('border', '#ff7f7f solid 2px');
$(watermark).css('position', 'absolute');
$(watermark).css('font-size', '13px');
$(watermark).css('font-family', 'Arial');
$(watermark).css('color', '#A9A9A9');
$(watermark).css({ left: (rect.width / 3), top: (rect.height / 3) });

var watermarkBackground = document.createElement('div');
$(watermarkBackground).css('background', '#FF0000');
$(watermarkBackground).css('opacity', 0.5);
$(watermark).append(watermarkBackground);
$(watermark).css('z-Index', 9999999999999999);

});
```

Watermarking all Dashboards in the Instance

The above script can also be added to the [application styling](#) so it runs for all the dashboards in that instance and not just a specific dashboard.

The following changes have been made to the script if you want to add it to the override file:

```
$(document).ready(function () {
    this.ready(function () {
        ..
        ..
        ..
        //add the script from the section above
        ..
        ..
        ..
    });
});
```

Test the Dashboard

Open the dashboard in **Sandbox View** mode.

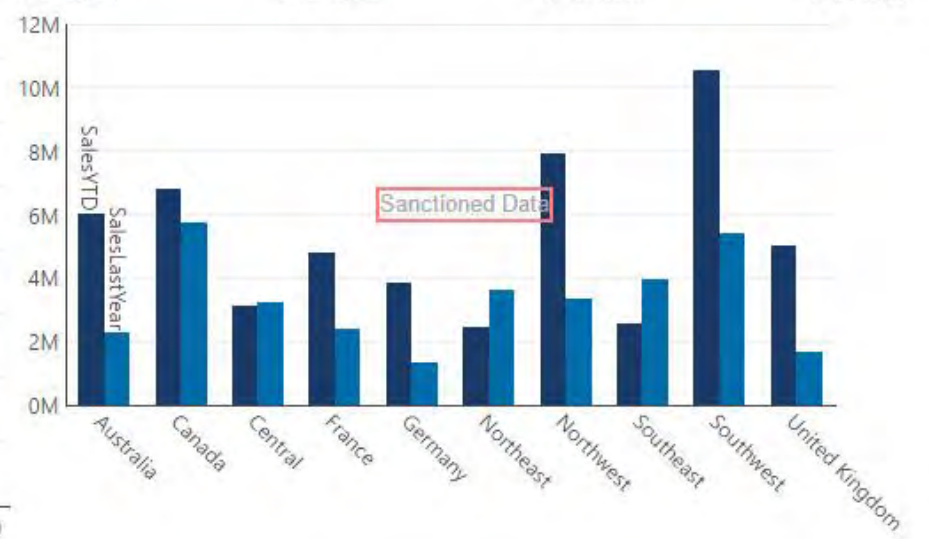
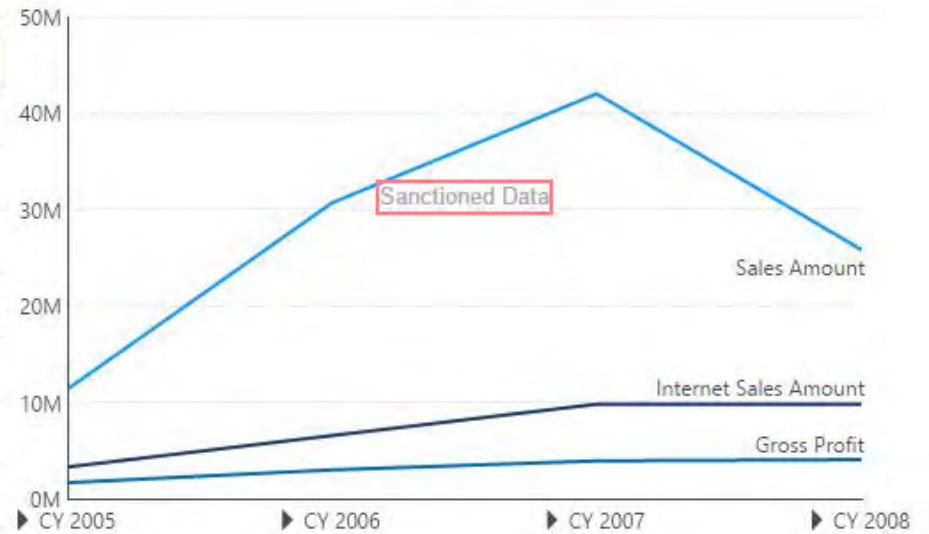
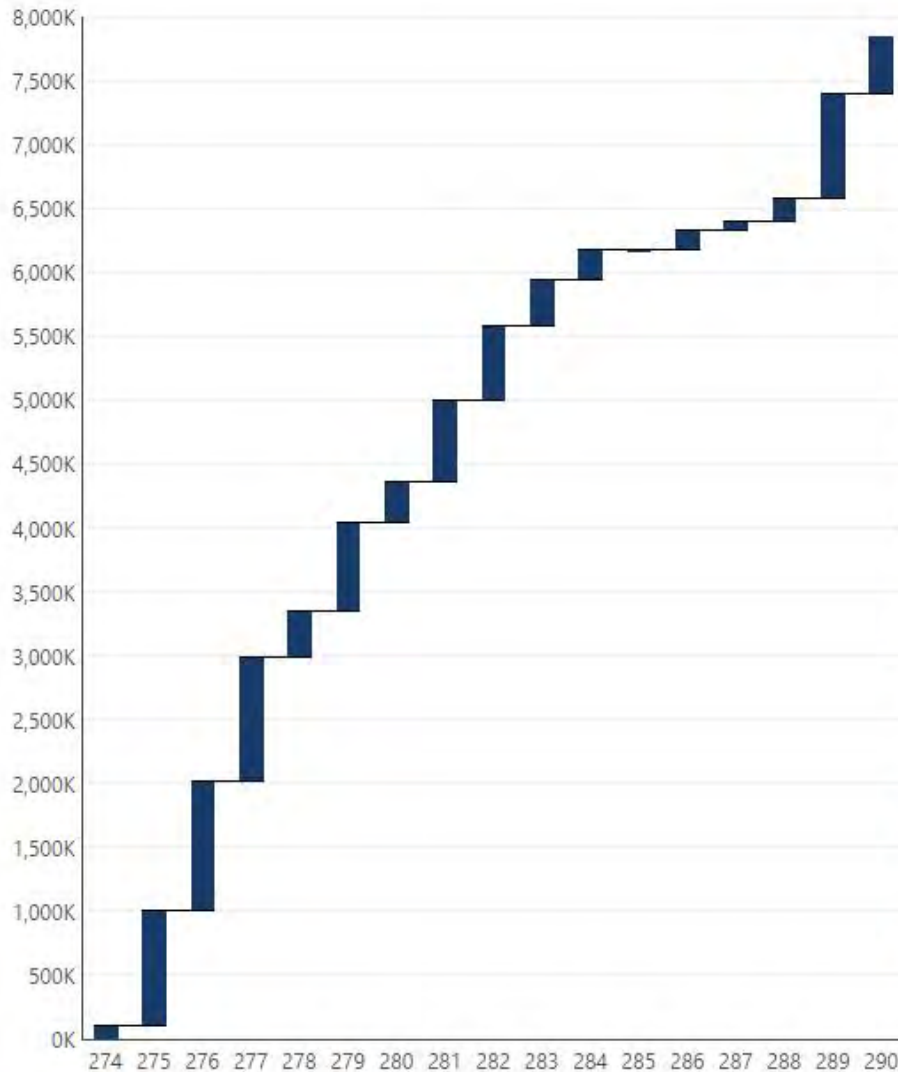


- Archive of documentation for Logi Symphony v24

 **Note:** Switching a dashboard that is already open from edit mode to view mode will not trigger a script added to the Ready event.

The dashboard will have the watermarks on metric sets that have been used for the **Shared folder** (i.e., belong to the Global project) containing the IT developed and approved **official data**.

My Analysis Dashboard





- Archive of documentation for Logi Symphony v24

For more information, see:

- [Working With Projects](#)
- [White Labeling the Application](#)
- [Edit Versus View Mode and Sandbox View](#)