

Power BI Training

Notes

Power BI Training

Notes

 Loading Data

 Transform Data

 Combining Data

 Append

 Merge

 Pivot and Unpivot

Data Modeling

 Cardinality

 Cross Filter Direction

 Resolve relationship path ambiguity

 Priority

 Weight

 Performance preference

 Example/Demo (Single vs Both Direction)

 Table/Column Properties

 Synonyms

 Custom Table

 Date Dimensions

 Creating a new Table Dimension in PowerBI using DAX

 Creating a new Table Dimension using M language

 Calculated Columns

 Calculated Measures

 Time Intelligence

 Optimization

Visualizations

R Visualizations

 R Scripts

 Data Transformation with R

 Parameters

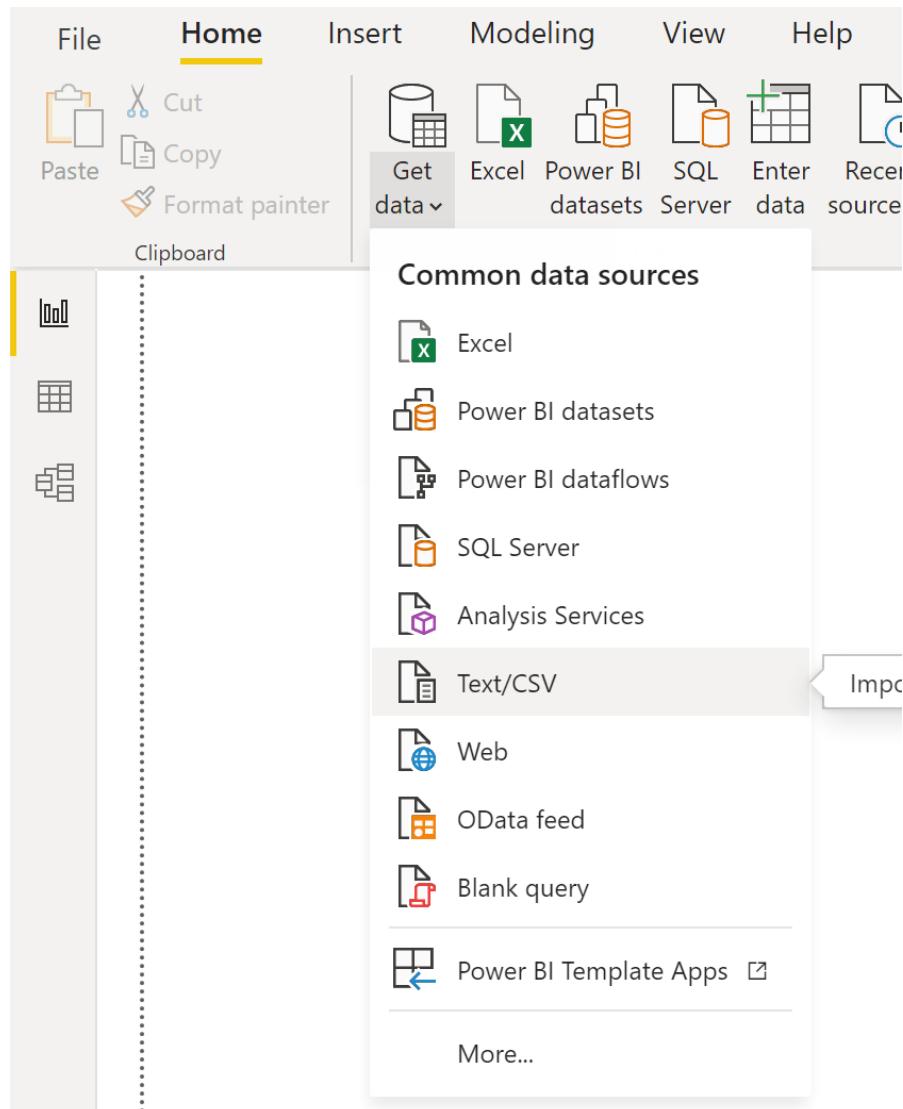
 List Parameteres

 Query Parameter

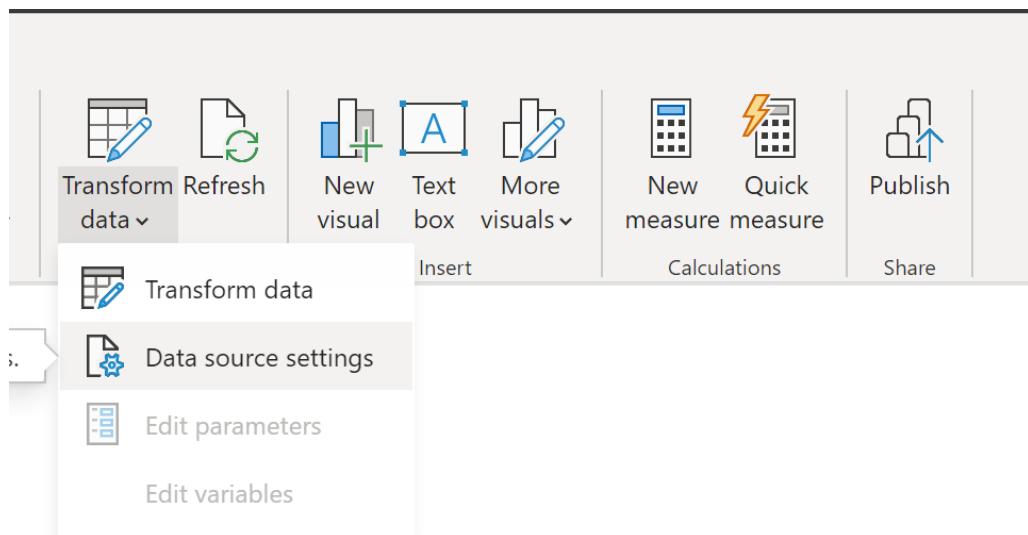
Loading Data

Demo

- Quick refresher on how to load/get data into PowerBI
- Connecting to a data source in Power BI
 - Example loading a CSV data set



- Data Source Setting
 - Example change data source



Data source settings

Manage settings for data sources that you have connected to using Power BI Desktop.

Data sources in current file Global permissions

Search data source settings A Z

z:\repos\coding-work\pluralsig...atasets\factcmpysales_2017.csv

[Change Source...](#) [Export PBIDS](#) [Edit Permissions...](#) [Clear Permissions ▾](#)

[Close](#)

Transform Data

Sales data from an e-commerce store // The company wants to have t...

	Column1	Column2	Column3	Column4	Column5	Column6	Column7
1		Sales data from an e-commerce store // The company wants to have t...					
2							
3	order_id (unique)	column3	order_date	sales	revenue	stock	pr
4	1		1/2/2017	0 sales	0	5	6..
6	NA		NA	NA	NA	NA	N/
7	NA	NA	NA	NA	NA	NA	N/
8	2		1/2/2017	0 sales	0	9	2..
9							
10	3		1/2/2017	5 sales	33.56	3	7..
11	3		1/2/2017	5 sales	33.56	3	7..
12	3		1/2/2017	5 sales	33.56	3	7..

- Enter 2 first rows

Data Type: Text ▾

Group By

Use First Row as Headers

Use First Row as Headers

Use Headers as

Promote the first row into column headers

- Remove Column 3

or

Choose Columns ▾

Remove Columns ▾

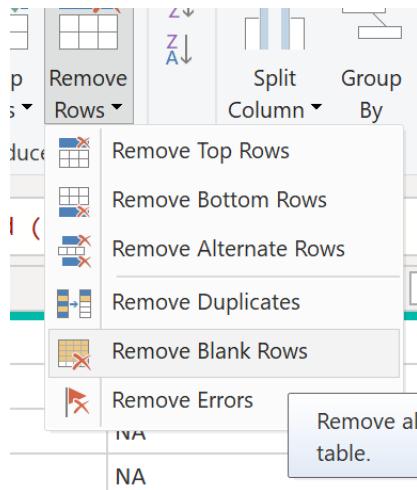
Keep Rows

Choose Columns

Go to Column

is(# Promoted Headers)

- Remove blank rows



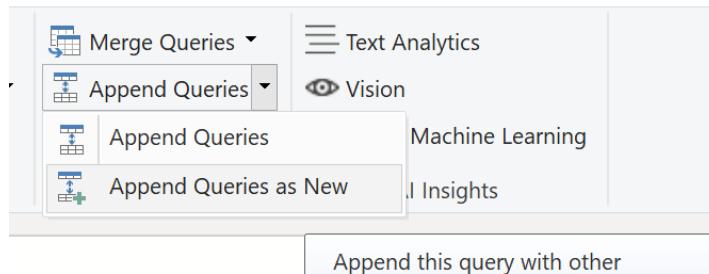
- Remove null rows by using filter from Order Date
- Remove unused columns

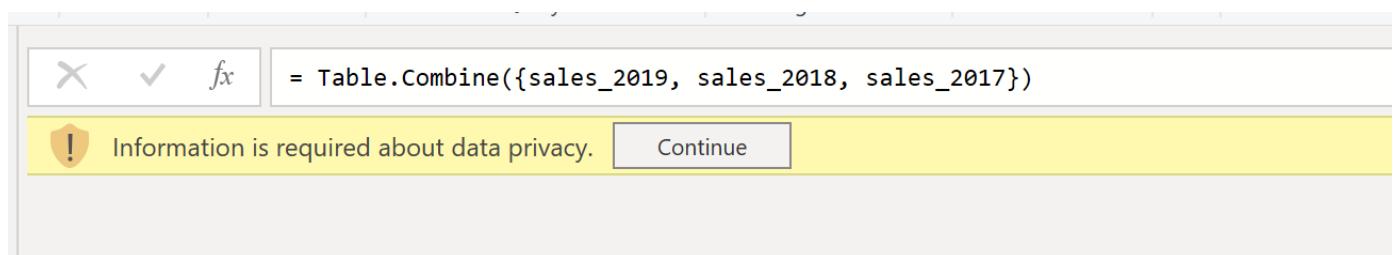
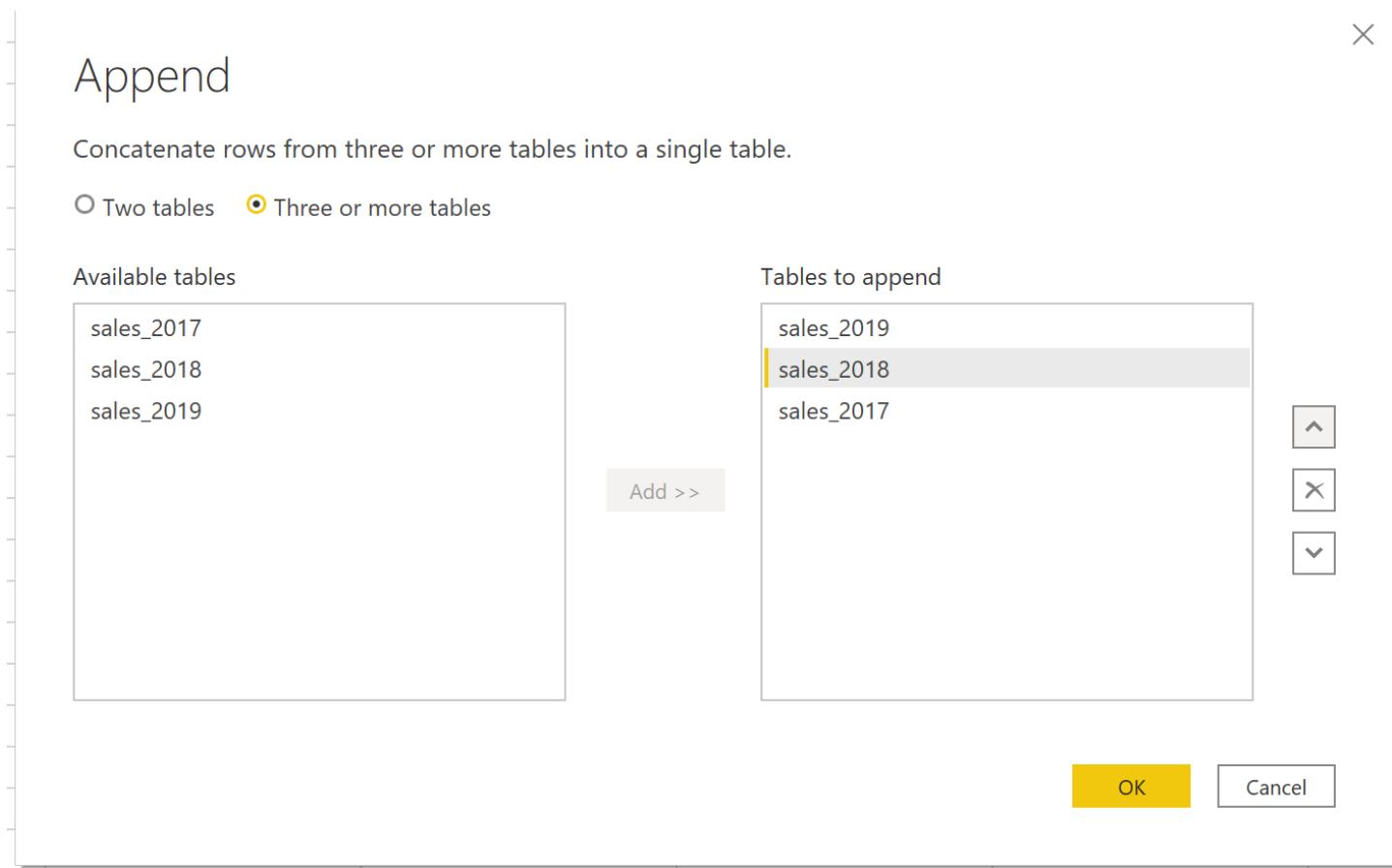
Combining Data

★ Demo

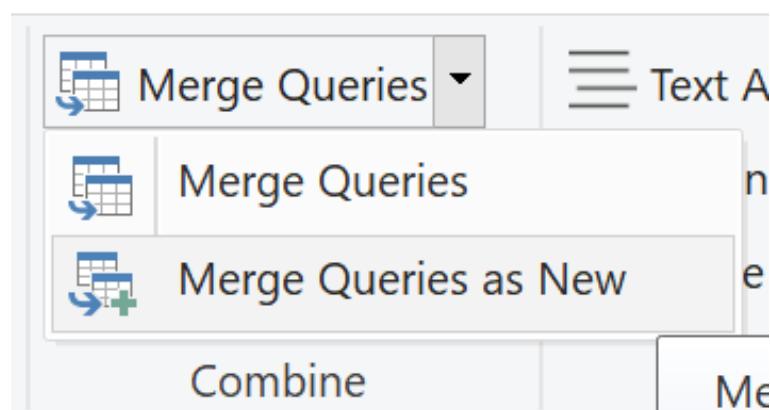
- When loading multiple files how you can combine two files
 - Show Merge Query
 - Show Append Query
- Append Queries vs Merge Queries

Append





Merge



- Merge as new vs merge queries

Append Queries vs Merge Queries

Merge Queries:

- Combines two or more queries by matching values in specified columns.
- Similar to SQL join operation.
- Creates a new table.
- Can combine tables horizontally.
- Requires at least one matching column in each table.
- Can combine tables with different numbers of columns

Append Queries:

- Adds rows of data from one query to another.
- Creates a new query.
- Can combine tables vertically.
- Requires matching columns with the same data types.
- Requires tables to have the same number of columns

Note

Merge Queries are used to combine tables **horizontally**, while **Append Queries** are used to combine tables **vertically**.

Merge Queries create a new table, while **Append Queries** create a new query.

Merge Queries require at least one matching column in each table, while **Append Queries** require matching columns with the same data types and the same number of columns.

X

Merge

Select a table and matching columns to create a merged table.

Sales



order_id	order_date	sales	revenue	stock	price	promo_type_1	promo_bin_1	promo_type_2
1	1/2/2017	0	0	5	6.5	PR14		PR03
2	1/2/2017	0	0	9	2.1	PR05	verylow	PR03
3	1/2/2017	5	33.56	3	7.25	PR14		PR03
4	1/2/2017	0	0	1	59.9	PR05	moderate	PR03
5	1/2/2017	1	1.00	5	1.00	PR14		PR03



sales_details



order_id	product_id	store_id
1	P0258	S0008
2	P0348	S0110
3	P0219	S0026
3	P0219	S0026
3	P0219	S0026

Join Kind

Left Outer (all from first, matching from second)



Use fuzzy matching to perform the merge

> Fuzzy matching options

* Estimating matches based on data previews

OK

Cancel

	 sales_details	
	Table	

Pivot and Unpivot

	A ^B _C Year	1.2 2021	1.2 2022	A ^B _C Type
1	January	113272	228433	Costs for employees
2	February	144611	213682	Costs for employees
3	March	189012	207938	Costs for employees
4	April	200317	239486	Costs for employees
5	May	184961	251093	Costs for employees
6	June	199864	202772	Costs for employees
7	July	207086	250205	Costs for employees
8	August	188931	232672	Costs for employees
9	September	229585	135287	Costs for employees
10	October	229630	156158	Costs for employees
11	November	214745	224940	Costs for employees
12	December	210958	152437	Costs for employees
13	January	26293.54957	222139.6243	Costs for rent
14	February	95668.19211	170426.8469	Costs for rent
15	March	173548.4992	7304.058054	Costs for rent
16	April	117398.5157	183101.9371	Costs for rent
17	May	120092.4774	66002.41072	Costs for rent
18	June	12085.48478	36836.63619	Costs for rent
19	July	50434.47319	185493.1607	Costs for rent
20	August	46323.80319	71576.65226	Costs for rent
21	September	8219.619201	27977.82764	Costs for rent
22	October	89370.59485	44134.72878	Costs for rent
23	November	128244.99	184002.1745	Costs for rent
24	December	135563.9076	52743.35507	Costs for rent

- We need to unpivot

Transform Add Column View Tools Help

Transpose

Reverse Rows

Count Rows

Data Type: Decimal Number
Replace Values
Unpivot Columns

Detect Data Type
Fill
Move

Rename
Pivot Column
Convert to List

Any Column

- Let's split up the type column into two columns

X

Pivot Column

Use the names in column "Type" to create new columns.

Values Column (i)

Cost_Amount

Advanced options

Aggregate Value Function

Sum

[Learn more about Pivot Column](#)

OK

Cancel

	A ^B _C Month	1 ² ₃ Year	1.2 Costs for employees	1.2 Costs for rent
1	April	2021	200317	117398.5157
2	April	2022	239486	183101.9371
3	August	2021	188931	46323.80319
4	August	2022	232672	71576.65226
5	December	2021	210958	135563.9076
6	December	2022	152437	52743.35507
7	February	2021	144611	95668.19211
8	February	2022	213682	170426.8469
9	January	2021	113272	26293.54957
10	January	2022	228433	222139.6243
11	July	2021	207086	50434.47319
12	July	2022	250205	185493.1607
13	June	2021	199864	12085.48478
14	June	2022	202772	36836.63619
15	March	2021	189012	173548.4992
16	March	2022	207938	7304.058054
17	May	2021	184961	120092.4774
18	May	2022	251093	66002.41072
19	November	2021	214745	128244.99
20	November	2022	224940	184002.1745
21	October	2021	229630	89370.59485
22	October	2022	156158	44134.72878
23	September	2021	229585	8219.619201
24	September	2022	135287	27977.82764

Data Modeling

★ Demo

- Walk through the data modeling process
 - Discuss why data modeling is critical
 - How it is done

💡 Note

- A model relationship propagates filters applied on the column of one model table to a different model table. Filters will propagate so long as there's a relationship path to follow, which can involve propagation to multiple tables.
Relationship paths are deterministic, meaning that filters are always propagated in the same way and without random variation. Relationships can, however, be disabled, or have filter context modified by model calculations that use particular DAX functions
- Model relationships don't enforce data integrity

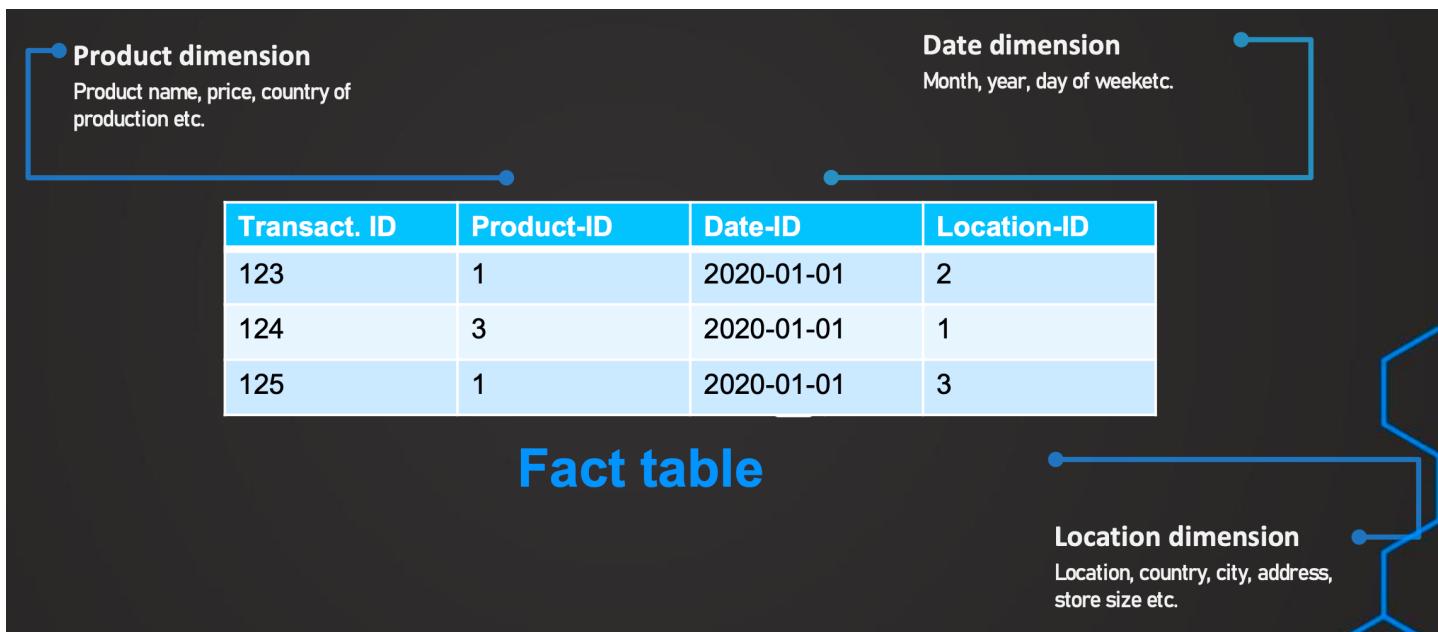
★ Best Practices from Microsoft

- We recommend you apply [star schema](#) design principles to produce a model comprising dimension and fact tables. It's common to set up Power BI to enforce rules that [filter dimension tables](#), allowing model relationships to efficiently propagate those filters to fact tables.
- The data type for both the "from" and "to" column of the relationship should be the same
- Working with relationships defined on **DateTime** columns might not behave as expected. The engine that stores Power BI data, only uses *DateTimedata* types; *Date*, *Time* and *Date/Time/Timezone* data types are Power BI formatting constructs implemented on top. Any model-dependent objects will still appear as *DateTime* in the engine (such as relationships, groups, and so on). As such, if a user selects **Date** from the **Modeling** tab for such columns, they still don't register as being the same date, because the time portion of the data is still being considered by the engine. [Read more about how Date/time types are handled](#). To correct the behavior, the column data types should be updated in the **Power Query Editor** to remove the *Time* portion from the imported data, so when the engine is handling the data, the values will appear the same

Disconnected Tables

- It's unusual that a model table isn't related to another model table. Such a table in a valid model design is described as a *disconnected table*. A disconnected table isn't intended to propagate filters to other model tables. Instead, it accepts "user input" (perhaps with a slicer visual), allowing model calculations to use the input value in a meaningful way.

- Relationship between tables
- Star Schema
 - Fact**
 - Observational or event data values
 - Dimensions**
 - Details about the data in the fact table
 - Foreign Key and Primary Keys**



- Autodetect relationship option

X

Options

GLOBAL

- Data Load
- Power Query Editor
- DirectQuery
- R scripting
- Python scripting
- Security
- Privacy
- Regional Settings
- Updates
- Usage Data
- Diagnostics
- Preview features
- Auto recovery
- Report settings

CURRENT FILE

- Data Load
- Regional Settings
- Privacy
- Auto recovery

Type Detection

- Detect column types and headers for unstructured sources

Relationships

- Import relationships from data sources on first load (i)
- Update or delete relationships when refreshing data (i)
- Autodetect new relationships after data is loaded (i)

[Learn more](#)

Time intelligence

- Auto date/time (i) [Learn more](#)

Background Data

- Allow data previews to download in the background

Parallel loading of tables

- Enable parallel loading of tables (i)

Q&A

- Turn on Q&A to ask natural language questions about your data (i) [Learn more](#)

OK

Cancel

- You can hide tables from the end-users so they do not need to see them

Cardinality

★ Demo

- What is Cardinality
- Cardinality and Data Modeling

Cardinality

- Each model relationship is defined by a cardinality type. There are four cardinality type options, representing the data characteristics of the "from" and "to" related columns. The "one" side means the column contains unique values; the "many" side means the column can contain duplicate values
- The four options, together with their shorthand notations, are described in the following bulleted list:
 - One-to-many (1:*)
 - Many-to-one (*:1)
 - One-to-one (1:1)
 - Many-to-many (:)
- The **one-to-many** and **many-to-one** cardinality options are essentially the same, and they're also the most common cardinality types.
- A **one-to-one** relationship means both columns contain unique values. This cardinality type isn't common, and it likely represents a suboptimal model design because of the storage of redundant data.
- A **many-to-many** relationship means both columns can contain duplicate values. This cardinality type is infrequently used. It's typically useful when designing complex model requirements. You can use it to relate many-to-many facts or to relate higher grain facts. For example, when sales target facts are stored at product category level and the product dimension table is stored at product level.

X

Edit relationship

Select tables and columns that are related.

Sales	▼
-------	---

promo_type_1	promo_bin_1	promo_type_2	promo_bin_2	promo_discount_2	product_id	store_id
PR14		PR03		NA	P0400	S0054
PR14		PR03		NA	P0400	S0094
PR14		PR03		NA	P0400	S0108

◀ ▶

dim_stores	▼
------------	---

store id	storetype id	store size	city_id	state - state abr - city	lat	long	
S0091	ST04	19	C013	AR - Arkansas - Hot Springs National Park	34.5137	-92.9685	
S0012	ST04	28	C005	TX - Texas - Huntsville	30.7813	-95.5953	
S0045	ST04	17	C008	NC - North Carolina - Asheville	35.6004	-82.4918	

Cardinality

- Many to one (*:1)
- Many to one (*:1)
- One to one (1:1)
- One to many (1:*)
- Many to Many (*:*)

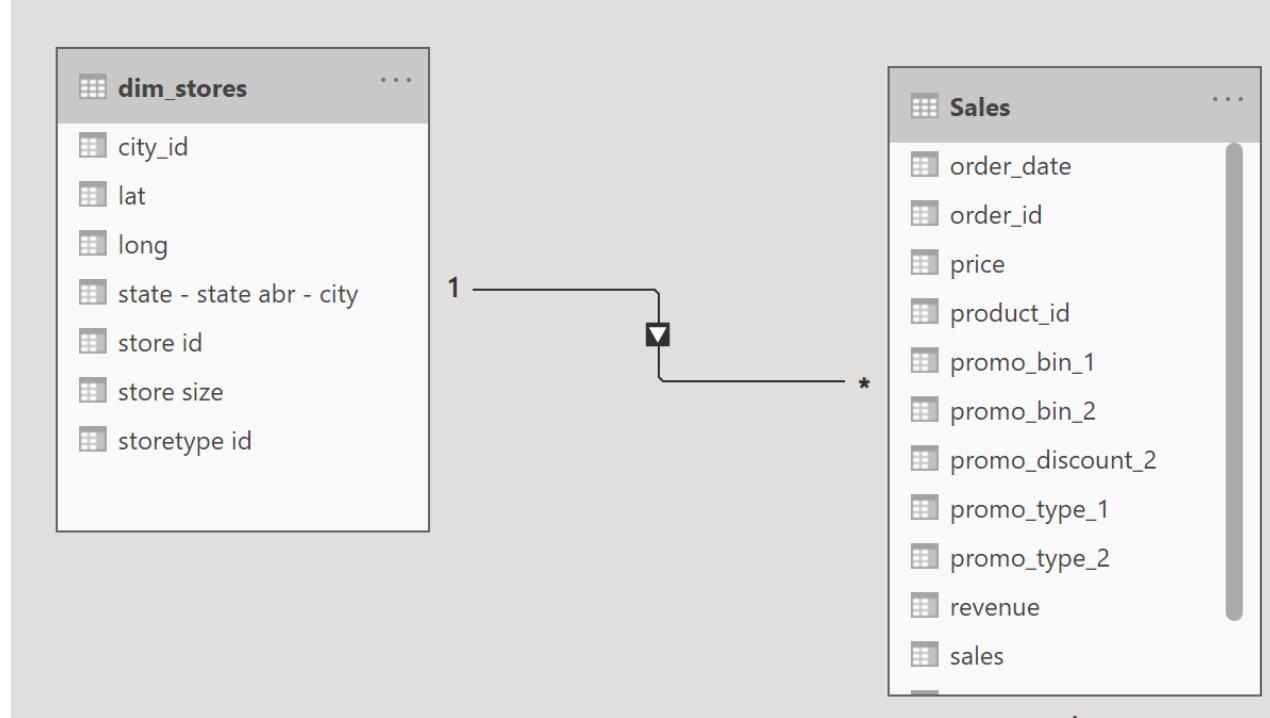
Cross filter direction

- Single

Apply security filter in both directions

OK

Cancel



- The **1** refers to the primary key values being unique in the dim_stores table, and the many indicates that the store_id can exist many times (not unique) in the fact table and thus acting as a foreign key.

Cross Filter Direction

Each model relationship is defined with a cross filter direction. Your setting determines the direction(s) that filters will propagate. The possible cross filter options are dependent on the cardinality type.

Cardinality type	Cross filter options
One-to-many (or Many-to-one)	Single Both
One-to-one	Both
Many-to-many	Single (Table1 to Table2) Single (Table2 to Table1) Both

Note

Cross filter direction in Power BI refers to the direction in which filters are applied between two tables in a relationship. There are two options for cross filter direction: single and both

Single: This means that the filter flows in one direction only, from the table on the "one" side of the relationship to the table on the "many" side

Both: This means that the filter flows in both directions, from the "one" side to the "many" side and vice versa

- When the cross filter direction is set to both, an additional property becomes available. This property allows for bi-directional filtering when row-level security (RLS) rules are enforced
- It's important to note that bi-directional filtering can introduce complex joins between tables and affect background processing performance, so it should only be used when necessary
- In Power BI Desktop model view, you can interpret a relationship's cross filter direction by noticing the arrowhead(s) along the relationship line. A single arrowhead represents a single-direction filter in the direction of the arrowhead, while two arrowheads represent bi-directional filtering

For **one-to-many** relationships, the cross filter direction **is always from the "one" side**, and optionally from the "many" side (bi-directional). For **one-to-one** relationships, the cross filter direction **is always from both tables**. Lastly, for **many-to-many** relationships, cross filter direction can be from **either one of the tables, or from both tables**. Notice that when the cardinality type includes a "one" side, that filters will always propagate from that side

Edit relationship

Select tables and columns that are related.

Sales						
promo_type_1	promo_bin_1	promo_type_2	promo_bin_2	promo_discount_2	product_id	store_id
PR14		PR03		NA	P0400	S0054
PR14		PR03		NA	P0400	S0094
PR14		PR03		NA	P0400	S0108

dim_stores							
store id	storetype id	store size	city_id	state - state abr - city	lat	long	
S0091	ST04	19	C013	AR - Arkansas - Hot Springs National Park	34.5137	-92.9685	
S0012	ST04	28	C005	TX - Texas - Huntsville	30.7813	-95.5953	
S0045	ST04	17	C008	NC - North Carolina - Asheville	35.6004	-82.4918	

Cardinality

Many to one (*:1)

Make this relationship active

Assume referential integrity

Cross filter direction

Single

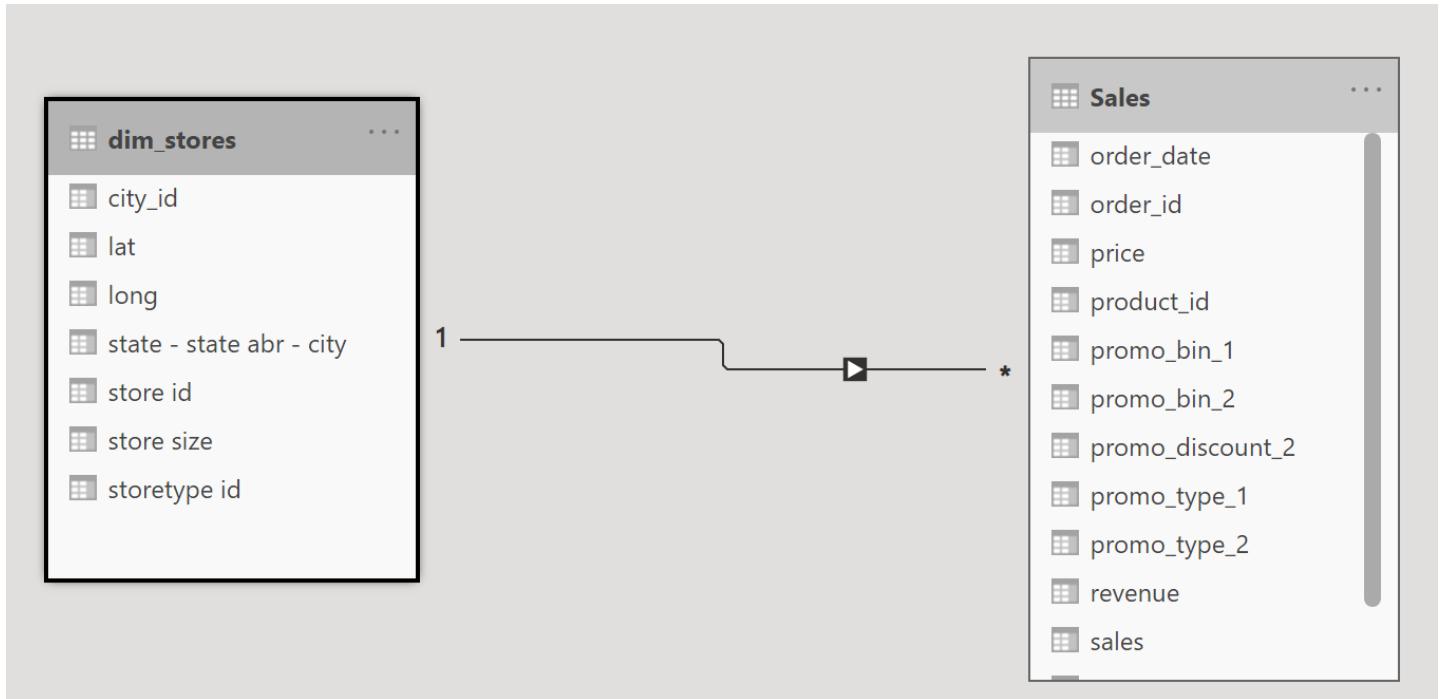
Single

Both

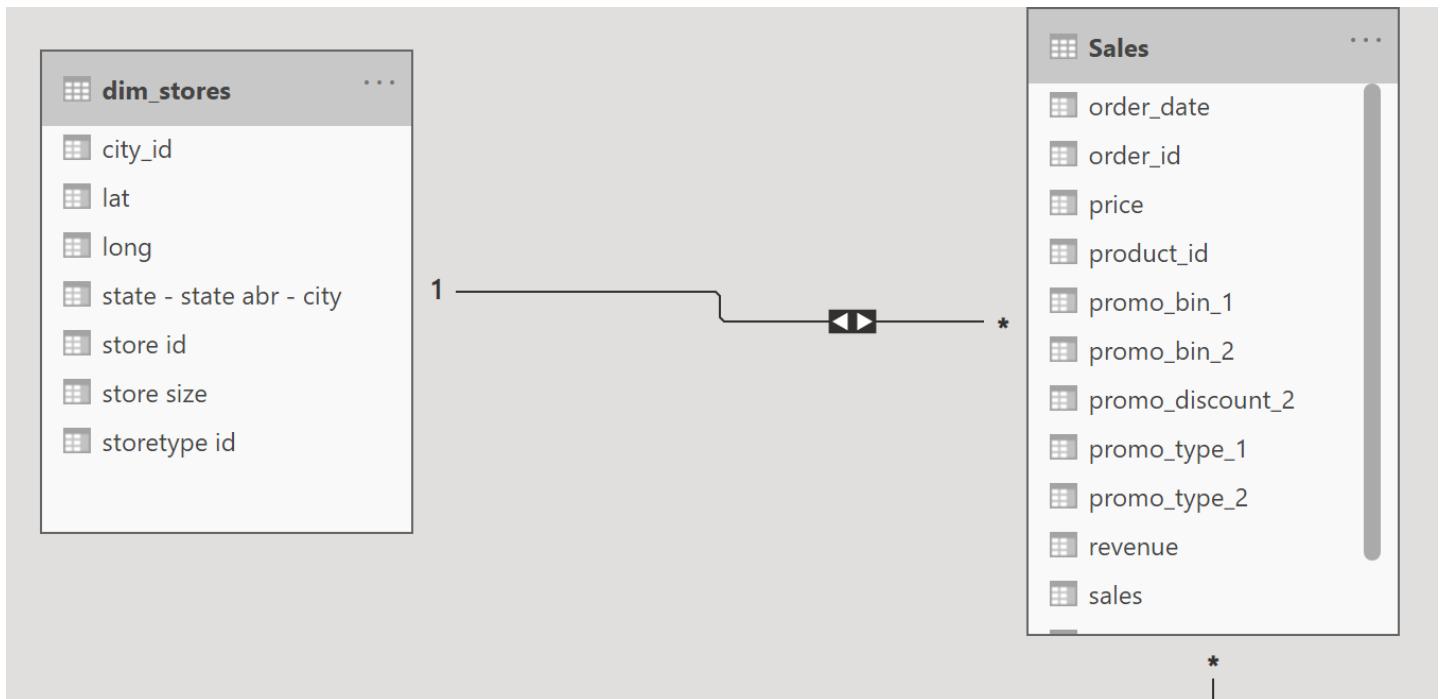
OK

Cancel

- Single (one way direction from dim to fact) - select from dim side to be applied as filter on other (fact) side
 - The arrow representation shows the direction (1 to many)
 - The icon shows the direction
 - From the dimension table (one side) to filter on the sales (fact) side



- Both (two way direction)



Note

In Power BI Desktop model view, you can interpret a relationship's active vs inactive status. An active relationship is represented by a solid line; an inactive relationship is represented as a dashed line.

There can only be one active filter propagation path between two model tables. However, it's possible to introduce additional relationship paths, though you must set these relationships as *inactive*. Inactive relationships can only be made active during the evaluation of a model calculation. It's achieved by using the `USERELATIONSHIP` DAX function.

Generally, it is recommended to define active relationships whenever possible. They widen the scope and potential of how report authors can use your model. Using only active relationships means that role-playing dimension tables should be duplicated in your model.

In specific circumstances, however, you can define one or more inactive relationships for a role-playing dimension table. You can consider this design when:

- There's no requirement for report visuals to simultaneously filter by different roles.
- You use the `USERELATIONSHIP` DAX function to activate a specific relationship for relevant model calculations.

For more information, see [Active vs inactive relationship guidance](#).

Edit relationship

Select tables and columns that are related.

Sales						
promo_type_1	promo_bin_1	promo_type_2	promo_bin_2	promo_discount_2	product_id	store_id
PR14		PRO3		NA	P0400	S0054
PR14		PRO3		NA	P0400	S0094
PR14		PRO3		NA	P0400	S0108

dim_stores						
store id	storetype id	store size	city_id	state - state abr - city	lat	long
S0091	ST04	19	C013	AR - Arkansas - Hot Springs National Park	34.5137	-92.9685
S0012	ST04	28	C005	TX - Texas - Huntsville	30.7813	-95.5953
S0045	ST04	17	C008	NC - North Carolina - Asheville	35.6004	-82.4918

Cardinality

Many to one (*:1)

Cross filter direction

Single

Make this relationship active

Apply security filter in both directions

Assume referential integrity

OK

Cancel

Resolve relationship path ambiguity

Bi-directional relationships can introduce multiple, and therefore ambiguous, filter propagation paths between model tables. When evaluating ambiguity, Power BI chooses the filter propagation path according to its [priority](#) and [weight](#).

Priority

Priority tiers define a sequence of rules that Power BI uses to resolve relationship path ambiguity. The first rule match determines the path Power BI will follow. Each rule below describes how filters flow from a source table to a target table.

1. A path consisting of one-to-many relationships.

2. A path consisting of one-to-many or many-to-many relationships.
3. A path consisting of many-to-one relationships.
4. A path consisting of one-to-many relationships from the source table to an intermediate table followed by many-to-one relationships from the intermediate table to the target table.
5. A path consisting of one-to-many or many-to-many relationships from the source table to an intermediate table followed by many-to-one or many-to-many relationships from the intermediate table to the target table.
6. Any other path.

When a relationship is included in all available paths, it's removed from consideration from all paths.

Weight

Each relationship in a path has a weight. By default, each relationship weight is equal unless the [USERELATIONSHIP](#) function is used. The *path weight* is the maximum of all relationship weights along the path. Power BI uses the path weights to resolve ambiguity between multiple paths in the same priority tier. It won't choose a path with a lower priority but it will choose the path with the higher weight. The number of relationships in the path doesn't affect the weight.

You can influence the weight of a relationship by using the [USERELATIONSHIP](#) function. The weight is determined by the nesting level of the call to this function, where the innermost call receives the highest weight.

Consider the following example. The **Product Sales** measure assigns a higher weight to the relationship between **Sales[ProductID]** and **Product[ProductID]**, followed by the relationship between **Inventory[ProductID]** and **Product[ProductID]**.

```

1 | Product Sales = 
2 | CALCULATE(
3 |   CALCULATE(
4 |     SUM(Sales[SalesAmount]),
5 |     USERELATIONSHIP(Sales[ProductID], Product[ProductID])
6 |   ),
7 |   USERELATIONSHIP(Inventory[ProductID], Product[ProductID]))

```

Note

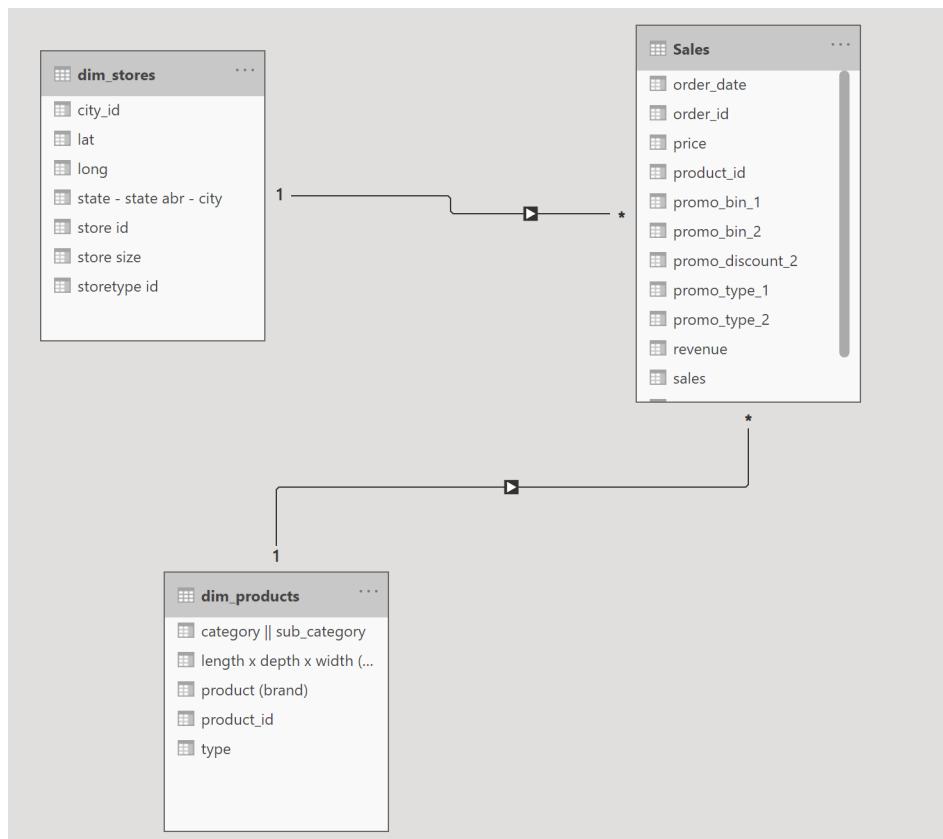
If Power BI detects multiple paths that have the same priority and the same weight, it will return an ambiguous path error. In this case, you must resolve the ambiguity by influencing the relationship weights by using the [USERELATIONSHIP](#) function, or by removing or modifying model relationships.

Performance preference

The following list orders filter propagation performance, from fastest to slowest performance:

1. One-to-many intra source group relationships
2. Many-to-many model relationships achieved with an intermediary table and that involve at least one bi-directional relationship
3. Many-to-many cardinality relationships
4. Cross source group relationships

Example/Demo (Single vs Both Direction)



The screenshot shows the Power BI desktop interface. On the left, there is a table visual titled "Count of product_id store_id" displaying a list of 20 rows, each containing a value of 699 followed by a store ID from S0001 to S0020. The ribbon menu at the top has tabs for Home, Insert, Modeling, View, Help, Format, and Data / Drill. The "Format" tab is selected. The right side of the screen shows the "Filters", "Visualizations", and "Fields" panes.

Filters

- Search: Count of product_id is (All)
- store_id is (All)
- Add data fields here

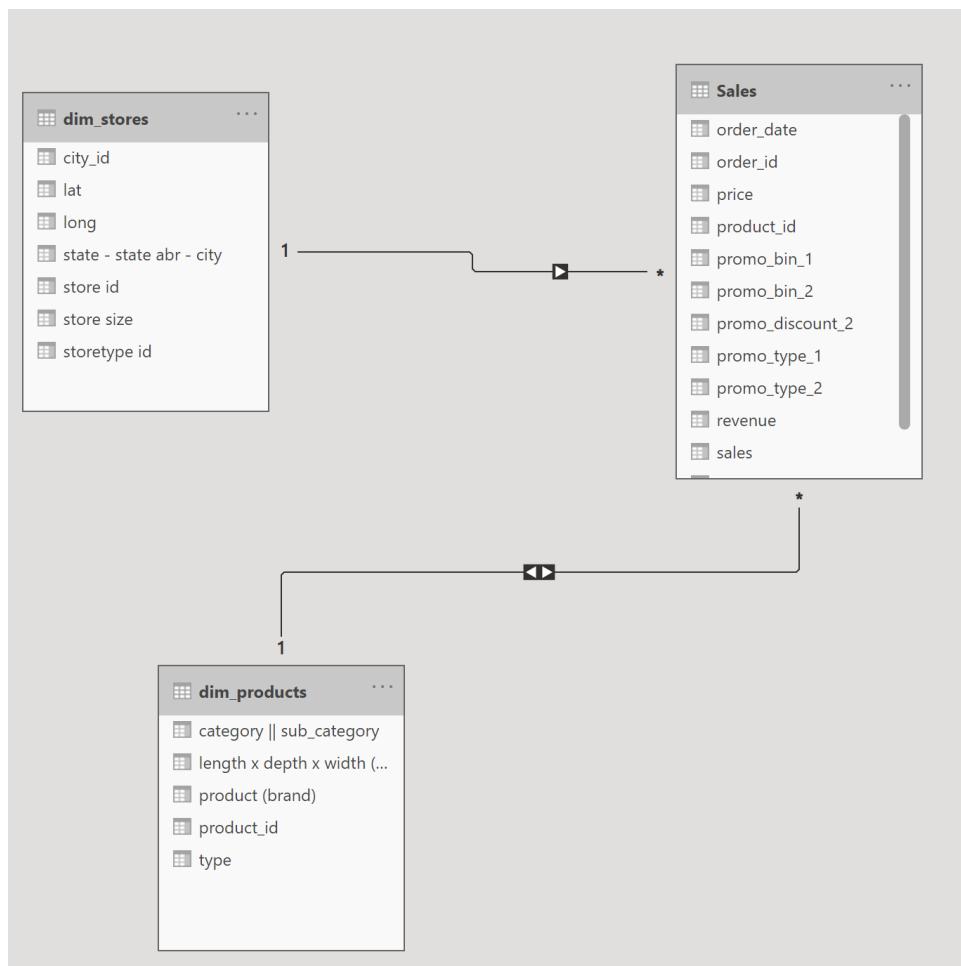
Visualizations

- Count of product_id
- store_id

Fields

- dim_products
 - category || sub_category
 - length x depth x width (...)
 - product (brand)
 - product_id
 - type
- dim_stores
 - city_id
 - lat
 - long
 - state - state abr - city
 - store id
 - store size
 - storetype id
- Sales
 - order_date
 - order_id
 - price
 - product_id
 - promo_bin_1
 - promo_bin_2
 - promo_discount_2
 - promo_type_1
 - promo_type_2
 - revenue
 - sales

Changing to Both will change the results



The screenshot shows the Power BI Desktop interface. On the left, there's a ribbon with Home, Insert, Modeling, View, Help, Format, and Data / Drill tabs. Under the Home tab, there are icons for Paste, Cut, Copy, Format painter, Clipboard, Get data (with options for Excel, Power BI datasets, SQL Server, Enter data, Recent sources), Transform data (with options for Refresh, New visual, Text box, More visuals, Quick measure, Calculations, Share). Below these are sections for Queries, Insert, and Data.

The main area displays a table titled "Count of product_id store_id". The table has two columns: "Count of product_id" and "store_id". The data is as follows:

Count of product_id	store_id
357	S0001
304	S0002
91	S0003
165	S0004
129	S0005
46	S0006
39	S0007
248	S0008
88	S0009
182	S0010
78	S0011
236	S0012
285	S0013
93	S0014
177	S0015
87	S0016
120	S0017
199	S0018
48	S0019
342	S0020
699	

To the right of the table is the Fields pane. It contains sections for Filters, Visualizations, and Fields. The Fields section shows a tree view of data models:

- dim_products
 - category
 - length
 - product
 - product_id
 - type
- dim_stores
- Sales
 - order_date
 - order_id
 - price
 - product_id
 - promo_bin_1
 - promo_bin_2
 - promo_disc
 - promo_type
 - revenue
 - sales
 - stock
 - store_id

Note

This is a good example to demonstrate the different in results when changing the table relationship type

Table/Column Properties

- Hide tables from end user view
- Show hidden tables

The screenshot shows the Power BI Fields pane. On the left, there's a toolbar with various icons for data modeling. Below it, two dashed boxes highlight specific areas: one labeled "ds here" and another labeled "ugh fields here". The main pane lists fields categorized by table:

- dim_products**:
 - category || ...
 - length x de...
 - product (br...
 - product_id
 - type
- dim_stores**:
 - order_date
 - Σ order_id
 - Σ price
 - product_id
 - promo_bin_1
 - promo_bin_2
 - promo_disc...
 - promo_typ...
 - promo_typ...
 - Σ revenue
 - Σ sales
 - Σ stock
 - store_id
- Sales**:
 - order_date
 - Σ order_id
 - Σ price
 - product_id
 - promo_bin_1
 - promo_bin_2
 - promo_disc...
 - promo_typ...
 - promo_typ...
 - Σ revenue
 - Σ sales
 - Σ stock
 - store_id

A context menu is open at the bottom right of the pane, with the "View hidden" option highlighted. Other options in the menu are: Unhide all, Collapse all, and Expand all.

- We can also hide/unhide specific columns
- **Table Properties**

Properties

General

Name: Sales

Description: Enter a description

Synonyms: sale

Row label: Select a row label

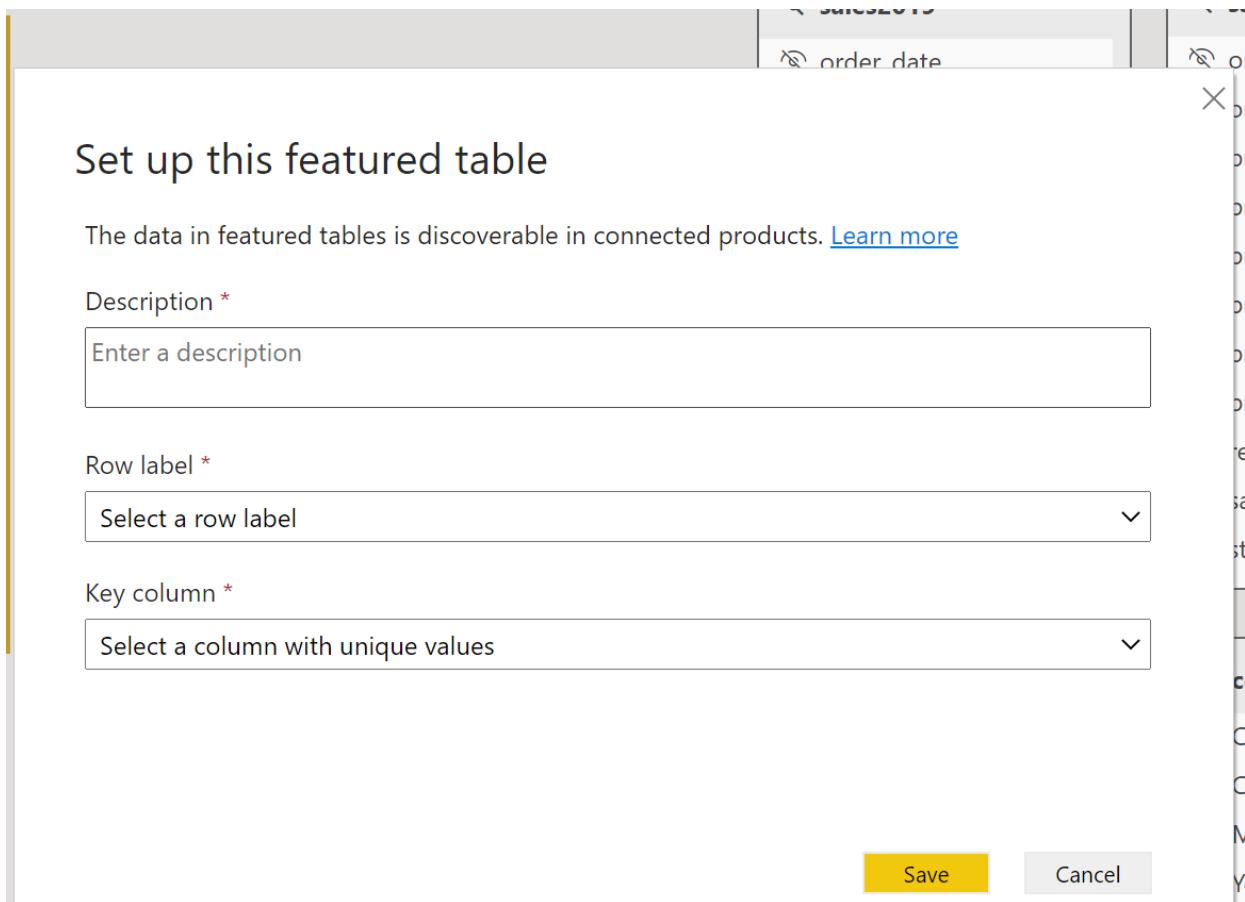
Key column: Select a column with unique values

Is hidden: No

Is featured table: No Edit

Advanced

- Row Label and Key Column are related to the featured tables Is featured table
 - In summary, the "Is Featured Table" option in Power BI is used to set a table as a featured table, which allows it to appear in Excel's Data Types Gallery. This feature is useful for linking data in Excel to data from Power BI, making it easier to add enterprise data to Excel.



1. In the **Set up this featured table** dialog box, provide the required fields:

- o A *Description*

Tip

Start the description with "Featured table" to help Power BI report creators identify it.

- o A **Row label**. The Row label field value is used in Excel so users can easily identify the row. It appears as the cell value for a linked cell, in the **Data Selector** pane, and in the **Information** card.
- o A **Key column**. The Key column field value provides the unique ID for the row. This value enables Excel to link a cell to a specific row in the table.

Set up this featured table

The data in featured tables is discoverable in connected products. [Learn more](#)

Description *

List of customers and their contact information.

Row label *

CompanyName

Key column *

CustomerID

Save

Cancel

- **Column** properties

Properties >

General

Name
price

Description
Enter a description

Synonyms
price

Display folder
Enter the display folder

Is hidden
No

^ Formatting

Data type
Decimal number

Format
General

Percentage format
No

Thousands separator
No

Decimal places
Auto

▼ Advanced

^ Advanced

Sort by column
price (Default)

Data category
Uncategorized

Summarize by
Sum

Is nullable
Yes

- The display folder allows us to organize our columns into folders

Synonyms

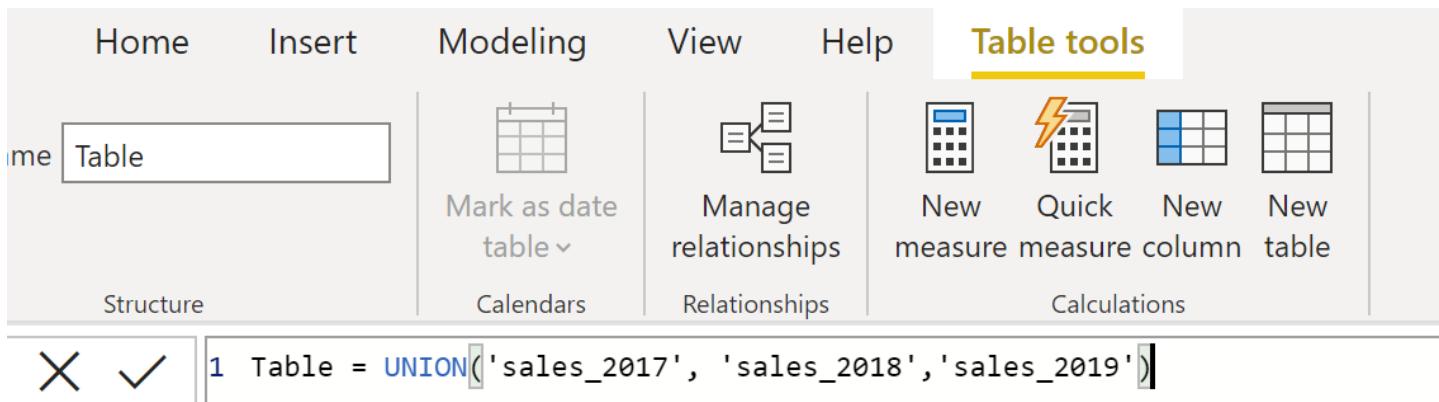
- Synonyms are used for NLP you can have multiple using a comma `sale, sales, invoices`
- In Power BI, synonyms are used to improve the user experience with the Q&A (Question and Answer) functionality. Synonyms are alternate names for tables, columns, or measures that can be added to the model view of Power BI Desktop

Here are some key points to keep in mind about synonyms in Power BI:

- Synonyms can be added to any field, including measures, columns, and tables
- Multiple synonyms can be added to a single field, separated by **commas**
- Synonyms can be added manually in the model view of Power BI Desktop, or they can be added automatically by using the training functionality of the Q&A visual
- Synonyms can be especially useful when working with large groups of report consumers who may have different backgrounds or speak different languages
- Synonyms can be exported from Power BI using the Synonyms API

By adding synonyms to tables, columns, and measures in Power BI, users can ask questions using the vocabulary that first comes to them, rather than choosing from a predefined list of columns. This can improve the user's experience with the report and make it easier for them to find what they are looking for

Custom Table



Date Dimensions

A date dimension table is a table that contains a range of dates with attributes such as month name, year, financial quarter, financial semester, and financial year. It is an integral part of a data warehouse and is used to visualize facts and figures over some time in Power BI.

The date dimension table can be used to create time intelligence calculations, such as year-to-date, month-to-date, and quarter-to-date

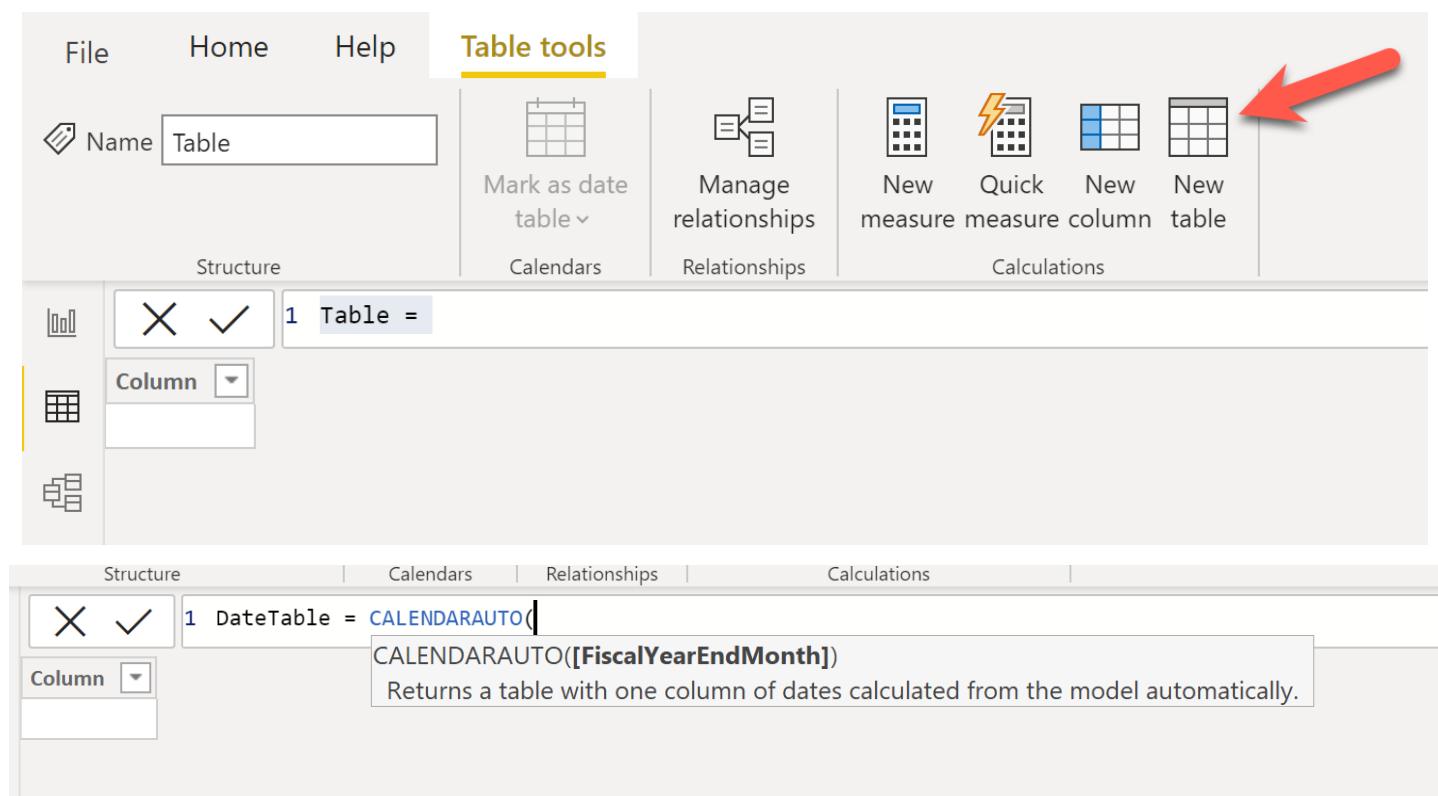
The date dimension table can be configured with start and end dates, fiscal calendar columns, and public holidays fetched live

It is a best practice to always have a Date Dimension included in our data model

Different methods to get a data dimension table in our data:

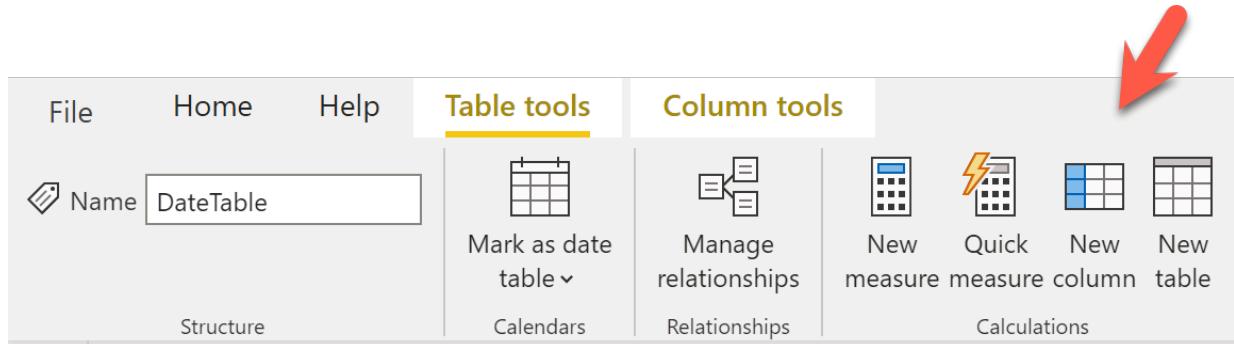
- The date dimension already exists in the database as a separate table
- A date dimension table can be created in Power BI Desktop by creating a calculated table using either the CALENDAR or CALENDARAUTO DAX function
- The date dimension table can be created using DAX to create a date dimension in Power B
- The date dimension table can be marked as a date table in Power BI by selecting the table in Power BI Desktop's model view and then setting the "Mark as Date Table" option to "Yes"

Creating a new Table Dimension in PowerBI using DAX



- All dates in the entire fiscal year will be included
- The CALENDARAUTO DAX function in Power BI returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is calculated automatically based on data in the model

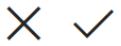
- The `CALENDARAUTO` function is useful for creating a date dimension table in Power BI without having to manually enter all the dates
- By default, the fiscal year ends in December (month 12)



Or using the `ADDCOLUMNS` function

```

1 DateTable = ADDCOLUMNS(CALENDARAUTO(),
2 "Month No", MONTH([Date]),
3 "Month Name", FORMAT([Date], "MMMM"),
4 "Quarter", QUARTER([Date]),
5 "Year", YEAR([Date]),
6 "Weekday Name", FORMAT([Date], "DDDD"),
7 "Weekday No", WEEKDAY([Date]))
```



```
1 DateTable = ADDCOLUMNS(CALENDARAUTO(),
2 "Month No", MONTH([Date]),
3 "Month Name", FORMAT([Date], "MMMM"),
4 "Quarter", QUARTER([Date]),
5 "Year", YEAR([Date]),
6 "Weekday Name", FORMAT([Date], "DDDD"),
7 "Weekday No", WEEKDAY([Date]))
```

Date	Month No	Month Name	Quarter	Year	Weekday Name	Weekday No
7/1/2017 12:00:00 AM	7	July	3	2017	Saturday	7
7/2/2017 12:00:00 AM	7	July	3	2017	Sunday	1
7/3/2017 12:00:00 AM	7	July	3	2017	Monday	2
7/4/2017 12:00:00 AM	7	July	3	2017	Tuesday	3
7/5/2017 12:00:00 AM	7	July	3	2017	Wednesday	4
7/6/2017 12:00:00 AM	7	July	3	2017	Thursday	5
7/7/2017 12:00:00 AM	7	July	3	2017	Friday	6
7/8/2017 12:00:00 AM	7	July	3	2017	Saturday	7
7/9/2017 12:00:00 AM	7	July	3	2017	Sunday	1
7/10/2017 12:00:00 AM	7	July	3	2017	Monday	2
7/11/2017 12:00:00 AM	7	July	3	2017	Tuesday	3
7/12/2017 12:00:00 AM	7	July	3	2017	Wednesday	4
7/13/2017 12:00:00 AM	7	July	3	2017	Thursday	5
7/14/2017 12:00:00 AM	7	July	3	2017	Friday	6
7/15/2017 12:00:00 AM	7	July	3	2017	Saturday	7
7/16/2017 12:00:00 AM	7	July	3	2017	Sunday	1
7/17/2017 12:00:00 AM	7	July	3	2017	Monday	2
7/18/2017 12:00:00 AM	7	July	3	2017	Tuesday	3
7/19/2017 12:00:00 AM	7	July	3	2017	Wednesday	4

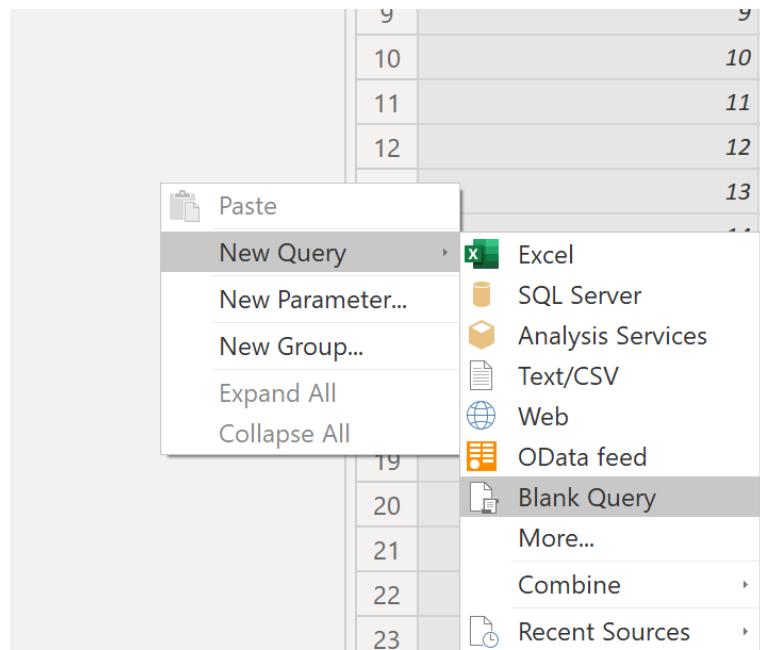
- Using the CALENDAR to give us more control

```
1 DateTable = ADDCOLUMNS(CALENDAR(DATE(2017,1,1),DATE(2019,12,31)),
2 "Month No", MONTH([Date]),
3 "Month Name", FORMAT([Date], "MMMM"),
4 "Quarter", QUARTER([Date]),
5 "Year", YEAR([Date]),
6 "Weekday Name", FORMAT([Date], "DDDD"),
7 "Weekday No", WEEKDAY([Date]))
```

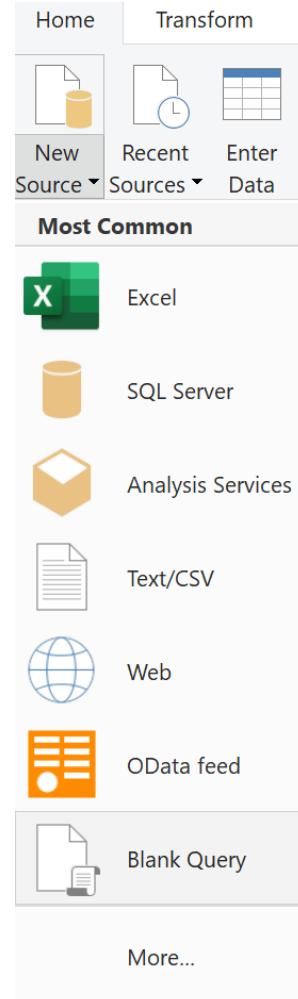
Creating a new Table Dimension using M language

Two ways to navigate

1.



2.



```
1 | = List.Dates(#date(2017, 1, 1), 365*3, #duration(1, 0, 0, 0))
```

List	
1	1/1/2017
2	1/2/2017
3	1/3/2017
4	1/4/2017
5	1/5/2017

Convert to table

Practice_1 - Power Query Editor

Transform

To Table

Keep Items ▾ Remove Items ▾

Remove Duplicates

Reverse Items

Statistics ▾

Convert

Manage Items

Sort

Numeric List

Queries [8]

List

1	1/1/2017
2	1/2/2017

Column1

1	1/1/2017
2	1/2/2017
3	1/3/2017
4	1/4/2017
5	1/5/2017

Add additional features

File Home Transform Add Column View Tools Help

Column From Examples ▾ Custom Column Invoke Custom Function General

Conditional Column Index Column ▾ Duplicate Column

Format Parse

Merge Columns Extract

Trigonometry ▾ Statistics Standard Scientific Information ▾

Date Time Duration

Text Analytics Vision Azure Machine Learning

AI Insights

Age Date Only Parse

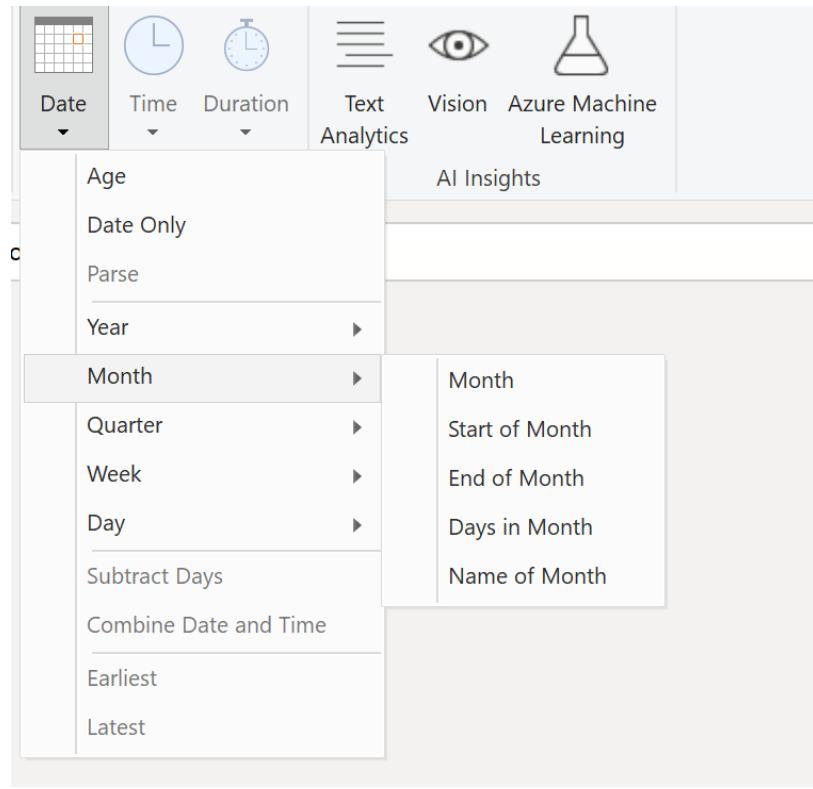
Year Month Quarter Week Day

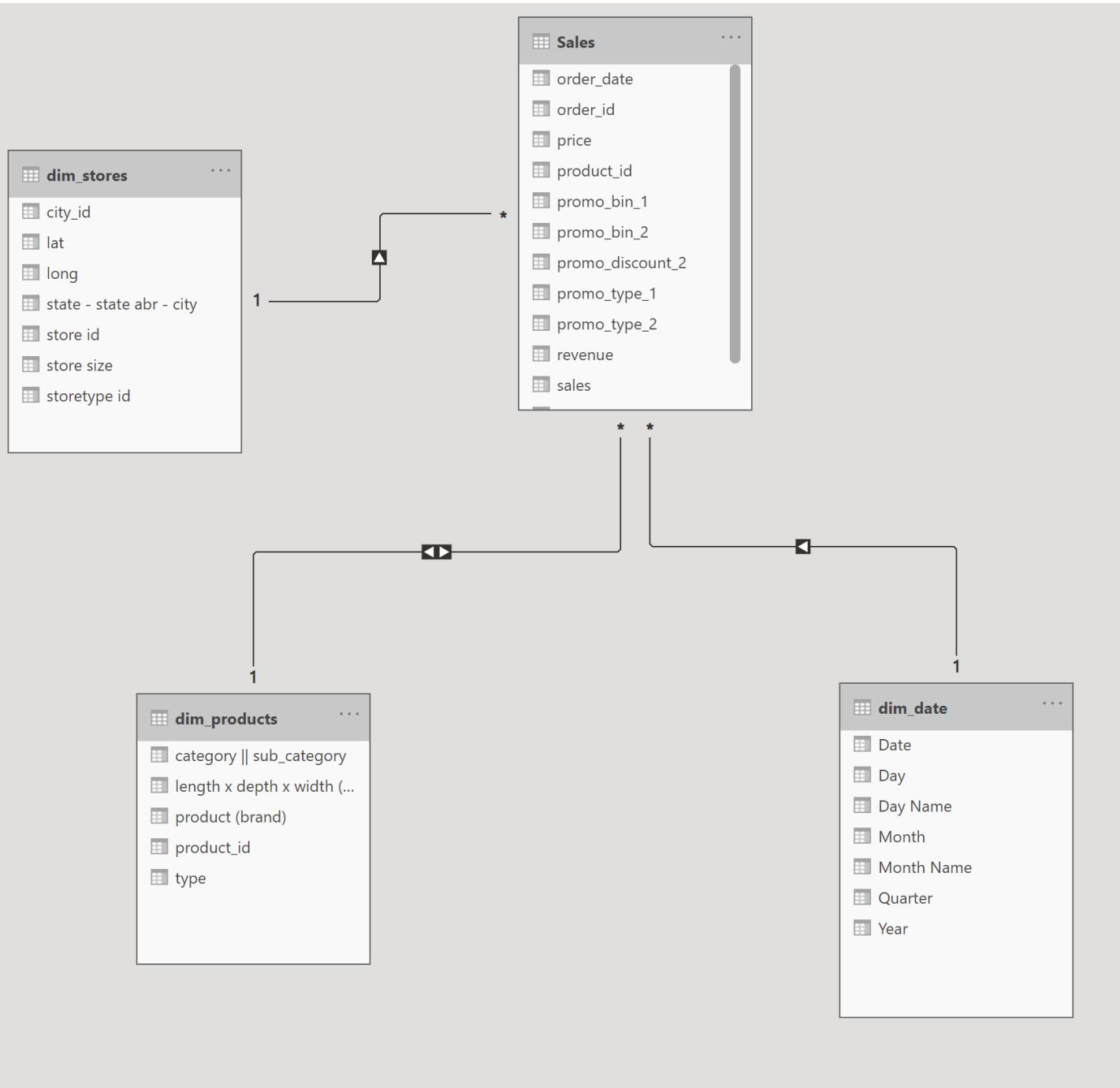
Subtract Days Combine Date and Time Earliest Latest

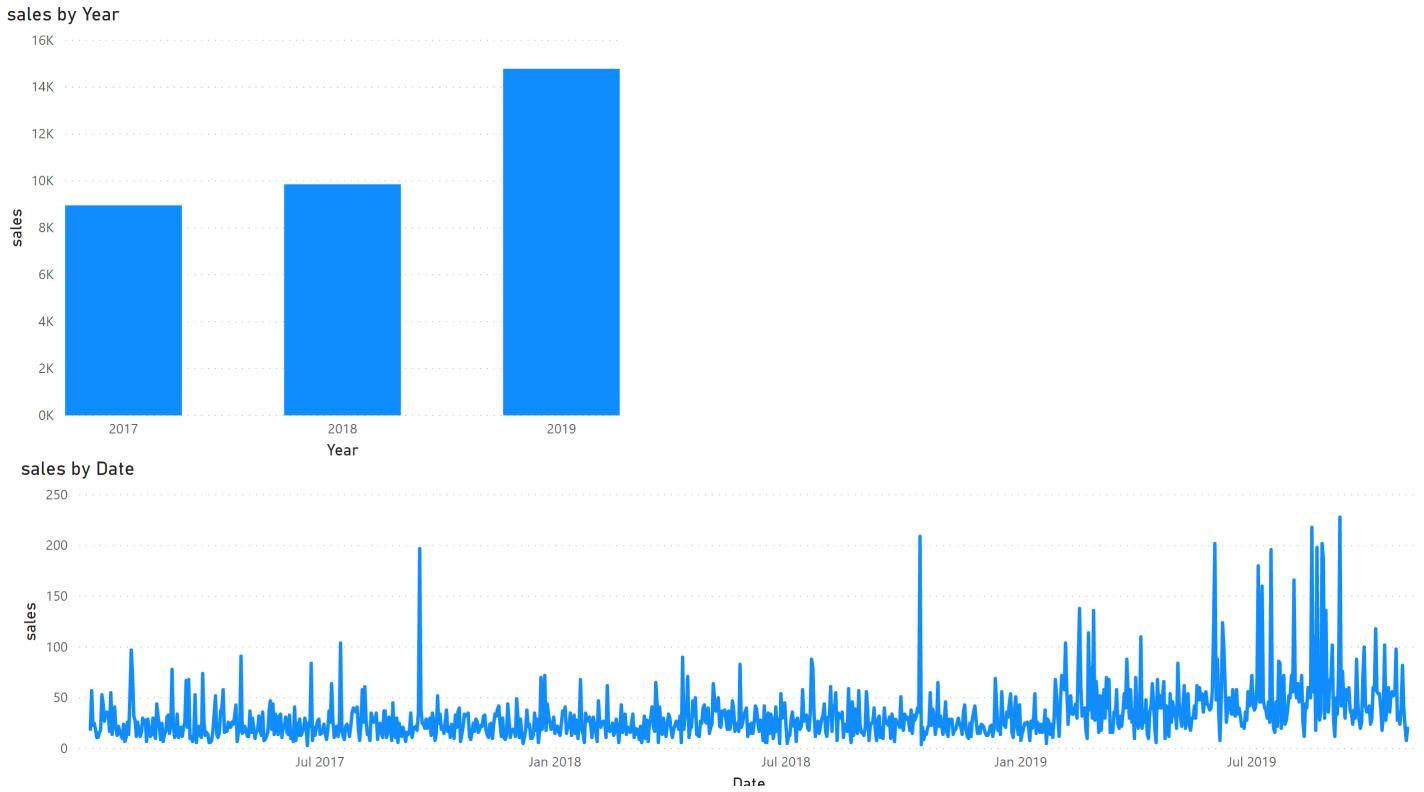
Queries [8]

Column1

1	1/1/2017
2	1/2/2017
3	1/3/2017
4	1/4/2017
5	1/5/2017
6	1/6/2017
7	1/7/2017
8	1/8/2017
9	1/9/2017
10	1/10/2017

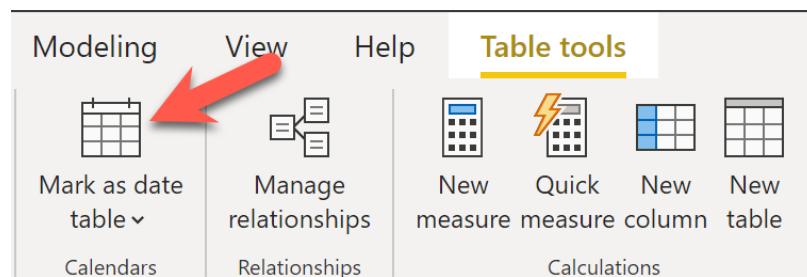






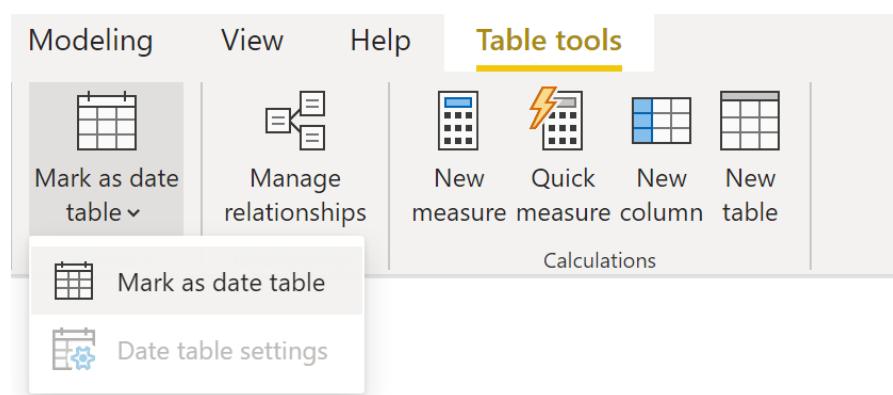
- Marking our date table as a Date Table in our model

-



- When you mark a table as a date table in Power BI, it tells Power BI that the table contains date values and enables time intelligence features such as time-based calculations and visualizations

-



-

X

Mark as date table

Select a column to be used for the date. The column must be of the data type 'date' and must contain only unique values. [Learn more](#)

Date column

▼

 Validated successfully

- ⓘ When you mark this as a date table, the built-in date tables that were associated with this table are removed. Visuals or DAX expressions referring to them may break.

[Learn how to fix visuals and DAX expressions](#)

OK

Cancel

- Notice the date hierarchy got removed - removed redundancy

Calculated Columns

•

The screenshot shows the Power BI desktop interface. A context menu is open for the 'Sales' table, listing options like 'New measure', 'New column', 'Refresh data', and 'Edit query'. Below the table name, there are additional options: 'Mark as date table', 'View hidden', 'Unhide all', 'Collapse all', and 'Expand all'. The 'Format' tab is selected in the ribbon, and the 'Table tools' section is active. In the 'Calculations' group, the 'New column' button is highlighted with a red arrow. The ribbon tabs shown are File, Home, Insert, Modeling, View, Help, Format, Data / Drill, and Table tools.

```
1 | Revenue_calculated = Sales[sales] * Sales[price]
```

- Notice the icon for a calculated column

Sales	
<input type="checkbox"/>	order_date
<input type="checkbox"/>	Σ order_id
<input type="checkbox"/>	Σ price
<input type="checkbox"/>	product_id
<input type="checkbox"/>	promo_bin_1
<input type="checkbox"/>	promo_bin_2
<input type="checkbox"/>	promo_discount_2
<input type="checkbox"/>	promo_type_1
<input type="checkbox"/>	promo_type_2
<input type="checkbox"/>	Σ revenue
<input type="checkbox"/>	Revenue_calculated
<input type="checkbox"/>	Σ sales
<input type="checkbox"/>	Σ stock
<input type="checkbox"/>	store_id

Note

- In general, it's recommended to use calculated columns judiciously and to consider using measures or calculated tables instead when possible, as they can be more efficient for certain types of calculations
- Calculated columns are computed at the row level within the table, which means that they can slow down query performance, especially when dealing with large datasets
- Calculated columns are stored in the data model
- Calculated columns can be used for calculations that require row-level context, such as concatenating columns or performing calculations based on a specific column value

Better options when possible is to:

- Have them created/available at the source
- Create in Power Query
- Alternatively, use DAX **Measure** instead

Calculated Measures

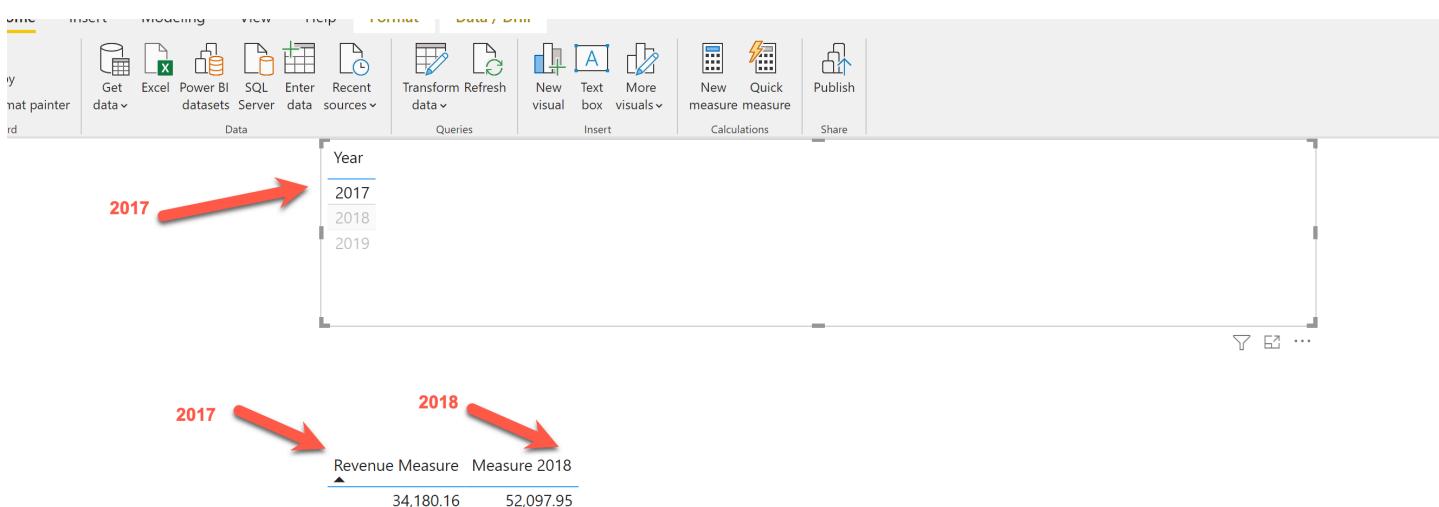
Note

- Calculated measures are computed at query time, which means that they can be more efficient for certain types of calculations
- Calculated measures are not stored in the data model
 - Just contains the logic for the calculation
- Calculated measures can be used for calculations that require aggregation, such as calculating the sum or average of a column

```
1 | Revenue Measure = SUMX(Sales, Sales[sales]* Sales[price])
```

- Example using `SUMX` an iterator function to calculate row by row
 - Understand why `SUM` will not get the correct aggregation
 - Example when removing `Order_ID`
 - `SUMX` is an iterative calculation function that calculates a sum of values for each value in a table
 - `SUMX` is capable of performing row-by-row calculations and iterates through every row of a specified table to complete the calculation
- Demo the use `CALCULATE` function
 - Example Revenue for 2018
 - The `CALCULATE` function in Power BI is used to evaluate an expression in a modified filter context

```
1 | Measure 2018 = CALCULATE([Revenue Measure], dim_date[Year] = 2018)
```



For more control show the use of `FILTER` function

```
1 | Measure 2018 = CALCULATE([Revenue Measure], FILTER(dim_date, dim_date[Year] = 2018))
```

Compare the results

Measure 2018 = CALCULATE([Revenue Measure], dim_date[Year] = 2018)	
2017	52,097.95
2018	52,097.95
2019	52,097.95
Total	52,097.95

to this

Measure 2018 = CALCULATE([Revenue Measure], FILTER(dim_date, dim_date[Year] = 2018))	
2018	52,097.95
Total	52,097.95

Show adding additional columns

Measure 2018 = CALCULATE([Revenue Measure], FILTER(dim_date, dim_date[Year] = 2018))		
2017	8960	
2018	52,097.95	9858
2019		14787
Total	52,097.95	33605

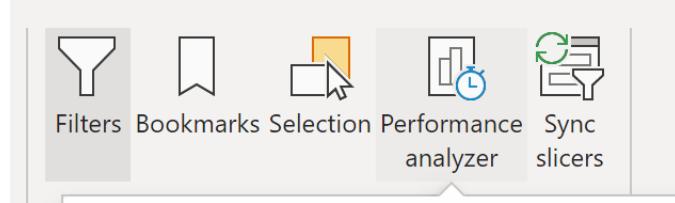
Time Intelligence

- Example calculate year-to-date
- Example calculate Previous Revenue

Optimization

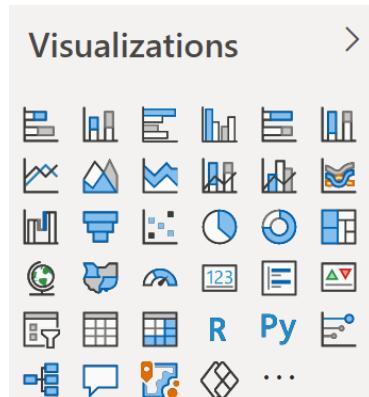
- Best practices to optimize model performance
 - Examples
 - Low level of cardinality
 - Ensure correct data types are used
 - Remove unnecessary columns and rows

- Star Schema
 - Dim tables
 - Leverage calculated measures
- Analyze performance using the Performance Analyzer



Visualizations

- Cover some basic visualizations available
 - Examples of how to customize and configure visuals from the visualization pane



- Demo Tables, Matrix, and Slices



- **Table** visualization
 - A two-dimensional display

The screenshot shows a Power BI interface with a matrix visualization on the left and a filter context pane on the right.

Matrix Visualization Data:

store_id	revenue
S0001	4,608.59
S0002	2,208.87
S0003	674.27
S0004	1,069.54
S0005	1,427.27
S0006	105.11
S0007	79.60
S0008	1,511.09
S0009	234.19
S0010	1,549.81
Total	171,291.00

Filter Context Pane:

- Filters on this visual:**
 - revenue is (All)
 - store_id is (All)
 - Add data fields here
- Filters on this page:**
 - Add data fields here
- Filters on all pages:**
 - Add data fields here
- Values:** store_id, revenue
- Drill through:**
 - Cross-report: Off (switch is off)
 - Keep all filters: On (switch is on)
 - Add drill-through fields here

- **Matrix visualization**

- Similar to table but supports multiple dimensions
- Matrix visualization in Power BI is useful when you want to display data meaningfully across multiple dimensions
- When you want to specify multiple variables in rows and columns and take advantage of Power BI's drill-down functionality

Example adding year

The screenshot shows a table visualization on the left and its corresponding filter context pane on the right.

Table Visualization:

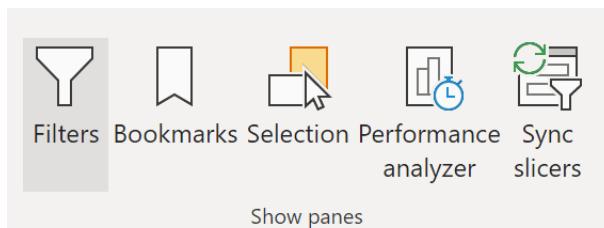
store_id	2017	2018	2019	Total
S0001	666.36	1,611.51	2,330.72	4,608.59
S0002	424.46	528.35	1,256.06	2,208.87
S0003	162.96	128.60	382.71	674.27
S0004	113.15	290.18	666.21	1,069.54
S0005		1,286.85	140.42	1,427.27
S0006	32.75	38.72	33.64	105.11
S0007			79.60	79.60
S0008	166.99	447.70	896.40	1,511.09
S0009	48.94	66.50	118.75	234.19
S0010	373.75	403.93	772.13	1,549.81
S0011	86.46	85.68	148.56	320.70
Total	31,765.00	45,387.05	94,138.95	171,291.00

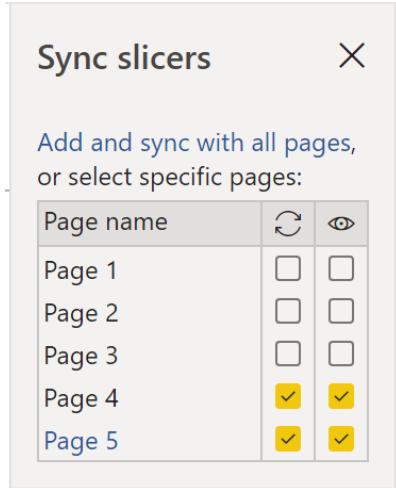
Filter Context Pane:

- Filters on this visual:**
 - revenue is (All)
 - store_id is (All)
 - Year is (All) (highlighted with a red arrow)
- Filters on this page:**
 - Add data fields here
 - Year (highlighted with a red arrow)
- Filters on all pages:**
 - Add data fields here
- Drill through:**
 - Cross-report Off
 - Keep all filters On (highlighted with a red arrow)

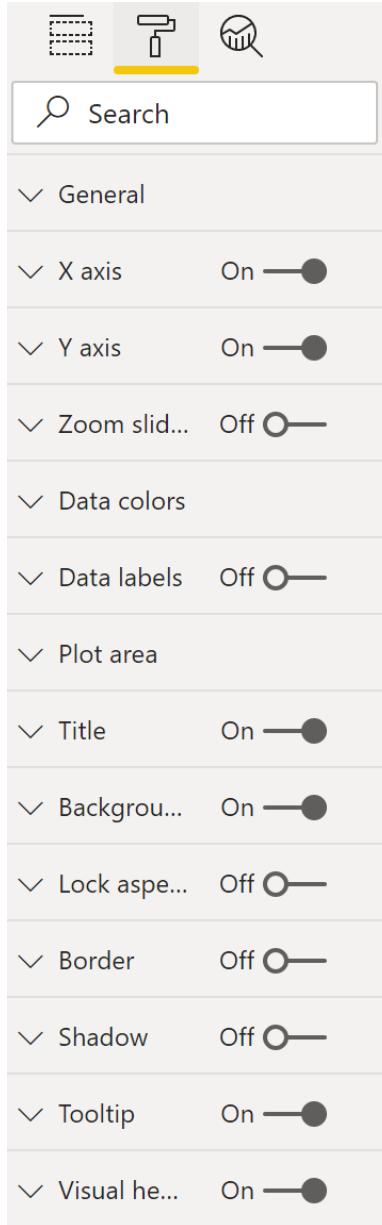
- **Slicer visualization**

- An alternative way to filtering
- Show the **Sync Slicers** options





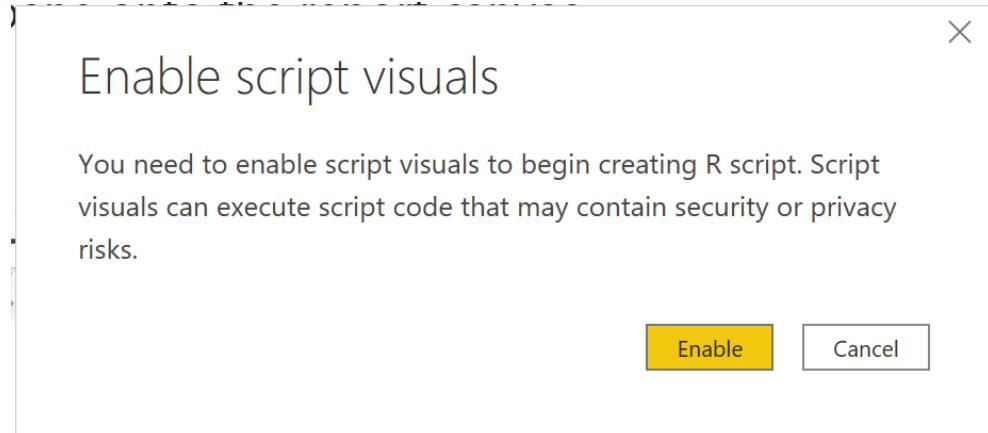
- Other charts
 - Discuss them briefly including card and multi-row card
- Show the different formatting options for some visuals
 - Example bar chart



- Custom Tool tip
- Show the use of Drillthrough

R Visualizations

- Demonstrate how to use the R visualization
- Demonstrate how do add data using R-Scripts
- Discuss the “Enable Script Visuals” alter box/option



R Scripts

- Install R
- Installation Path

```

1 | R.home()
2 | file.path(R.home("bin"), "R")
3 | sys.getenv('R_HOME')

```



Options

GLOBAL

- Data Load
- Power Query Editor
- DirectQuery
- R scripting
- Python scripting
- Security
- Privacy
- Regional Settings
- Updates
- Usage Data
- Diagnostics
- Preview features
- Auto recovery
- Report settings

CURRENT FILE

- Data Load
- Regional Settings
- Privacy
- Auto recovery

R script options

To choose a home directory for R, select a detected R installation from the drop-down list, or select Other and browse to the location you want.

Detected R home directories:

Other

Set an R home directory:

C:\Program Files (x86)\R\R-4.3.1

[Browse](#)

[How to install R](#)

To choose which R integrated development environment (IDE) you want Power BI Desktop to launch, select a detected IDE from the drop-down list, or select Other to browse to another IDE on your machine.

Detected R IDEs:

R Studio

[Learn more about R IDEs](#)

[Change temporary storage location](#)

Note: Sometimes, R custom visuals automatically install additional packages. For those to work, the temporary storage folder name must be written in Latin characters (letters in the English alphabet).

[OK](#)

[Cancel](#)

X

Get Data

r sc X

All

Other

All

R script

Python script

Certified Connectors

Template Apps

Connect

Cancel

X

R script

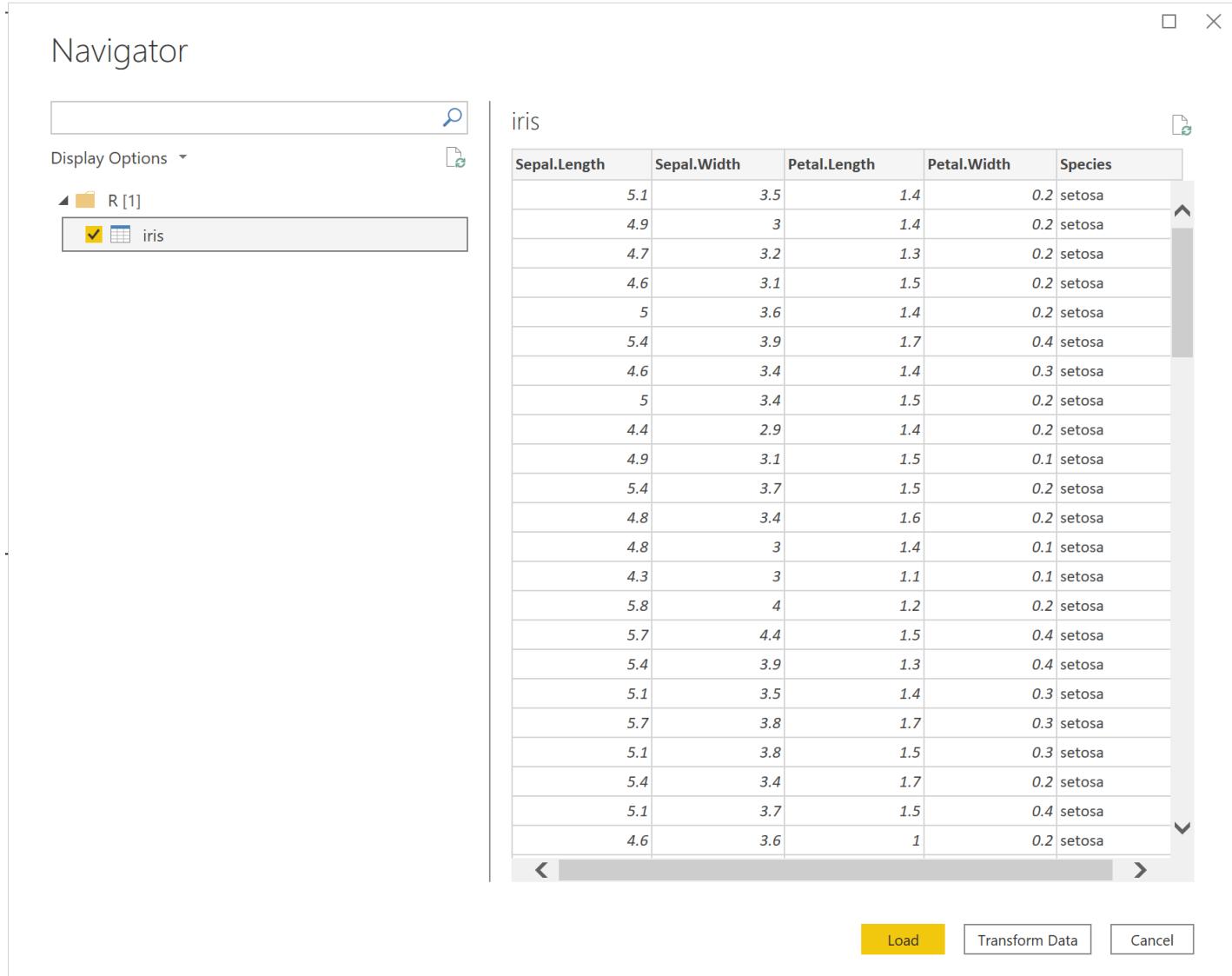
Script

```
data(iris)
```

The script will run with the following R installation C:\Program Files\Microsoft\MRO-for-RRE\8.0\R-3.2.2.
To configure your settings and change which R installation you want to run, go to Options and settings.

OK

Cancel

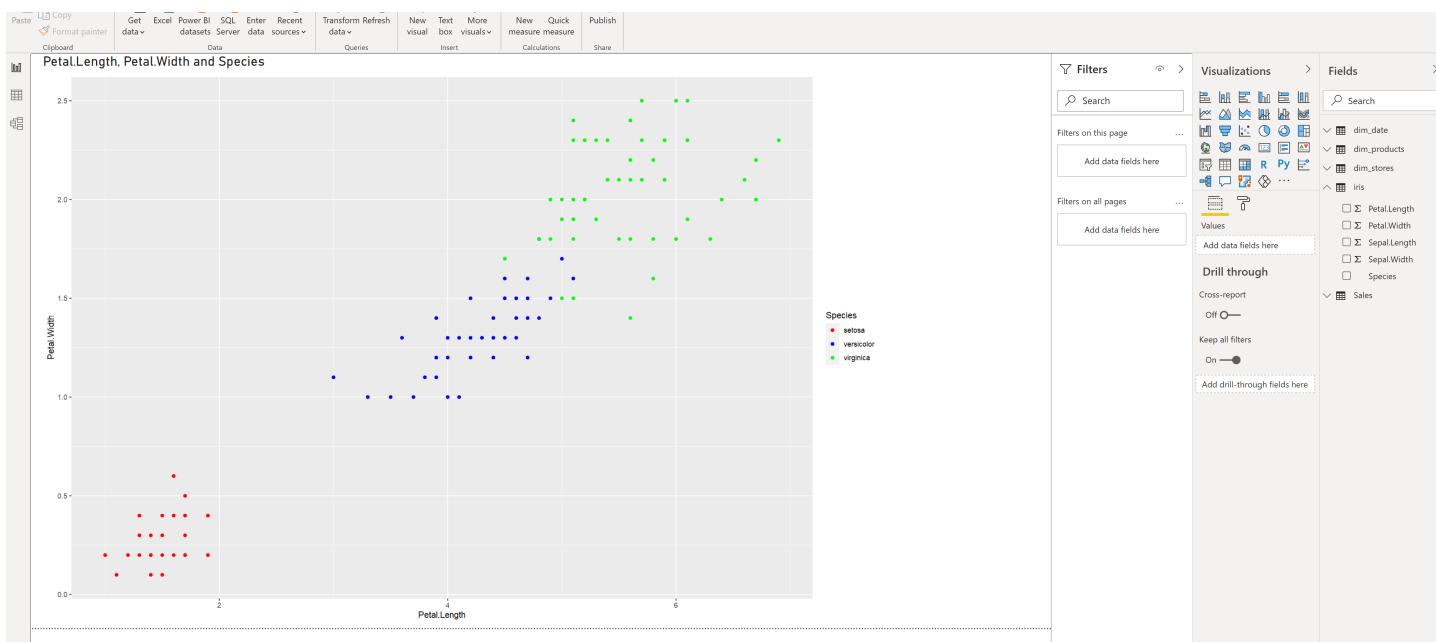


```
1 # The following code to create a dataframe and remove duplicated rows is always executed  
and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(Petal.Length, Petal.Width)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:  
7  
8 library(ggplot2)  
9 ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width)) + geom_point()
```

```

1 # The following code to create a dataframe and remove duplicated rows is always executed
2 # and acts as a preamble for your script:
3
4 # dataset <- data.frame(Petal.Length, Petal.Width, Species)
5 # dataset <- unique(dataset)
6
7 # Paste or type your script code here:
8
9 library(ggplot2)
10 ggplot(data = dataset, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
11   geom_point() +
    scale_color_manual(values = c("red", "blue", "green"))

```



```

1 library(ggplot2)
2 ggplot(data = dataset, aes(x = Petal.Length, y = Petal.Width)) +
3   geom_point(aes(color=Species), size=5) +
4   ggtitle("Petal Width and Length") +
5   labs(x="Petal Width", y= "Petal Length") +
6   theme_bw() +
7   theme(title=element_text(size=24, color="grey8"))

```

Data Transformation with R

Run R script

Enter R scripts into the editor to transform and shape your data.

Script

```
# 'dataset' holds the input data for this script  
  
output <- dataset  
  
output$rev_new <- output$price * output$sales|
```

The script will run with the following R installation C:\Program Files (x86)\R\R-4.3.1.

To configure your settings and change which R installation you want to run, go to Options and settings.

Run R script

Enter R scripts into the editor to transform and shape your data.

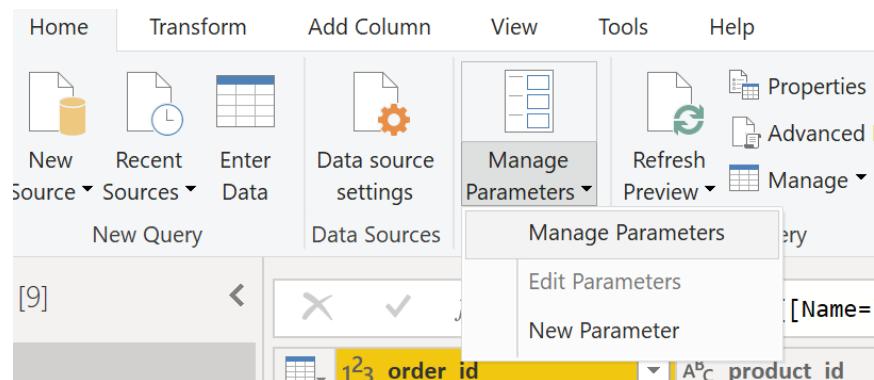
Script

```
# 'dataset' holds the input data for this script  
  
output <- dataset  
  
output$rev_new <- output$price * output$sales  
  
output$sales <- replace(output$sales, is.na(output$sales), 0)|
```

The script will run with the following R installation C:\Program Files (x86)\R\R-4.3.1.

To configure your settings and change which R installation you want to run, go to Options and settings.

Parameters



Manage Parameters

New

1 ² 3 rate	X
-----------------------	---

Name: rate

Description: Enter an exchange rate

Required:

Type: Decimal Number

Suggested Values: Any value

Current Value: 0.071

OK Cancel

Queries [10]

- Sales
- sales2018
- sales2019
- sales_details
- costs table
- dim_products
- dim_stores
- dim_date
- iris
- rate (0.071)

Current Value

0.071

Manage Parameter

X

Custom Column

Add a column that is computed from the other columns.

New column name

new_revenue

Custom column formula ⓘ

= [revenue] * rate

Available columns

order_id
product_id
store_id
order_date
sales
revenue
stock
.

<< Insert

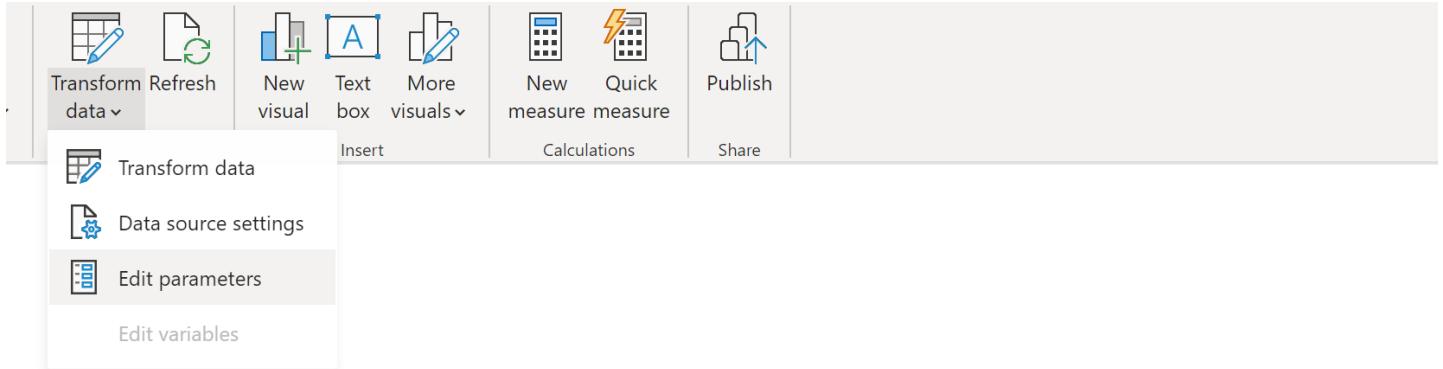
[Learn about Power Query formulas](#)

✓ No syntax errors have been detected.

OK

Cancel

You can change in the report for end user



121.62K

new_revenue

171.29K

revenue



Manage Parameters

New

1²3 rate X

Name

rate

Description

Enter an exchange rate

Required

Type

Decimal Number ▼

Suggested Values

Any value ▼

Any value

List of values

Query

List of values ▼

OK

Cancel

List Parameters

X

Manage Parameters

New

1 ² 3	rate
A ^B C	states

Name: states

Description:

Required

Type: Any

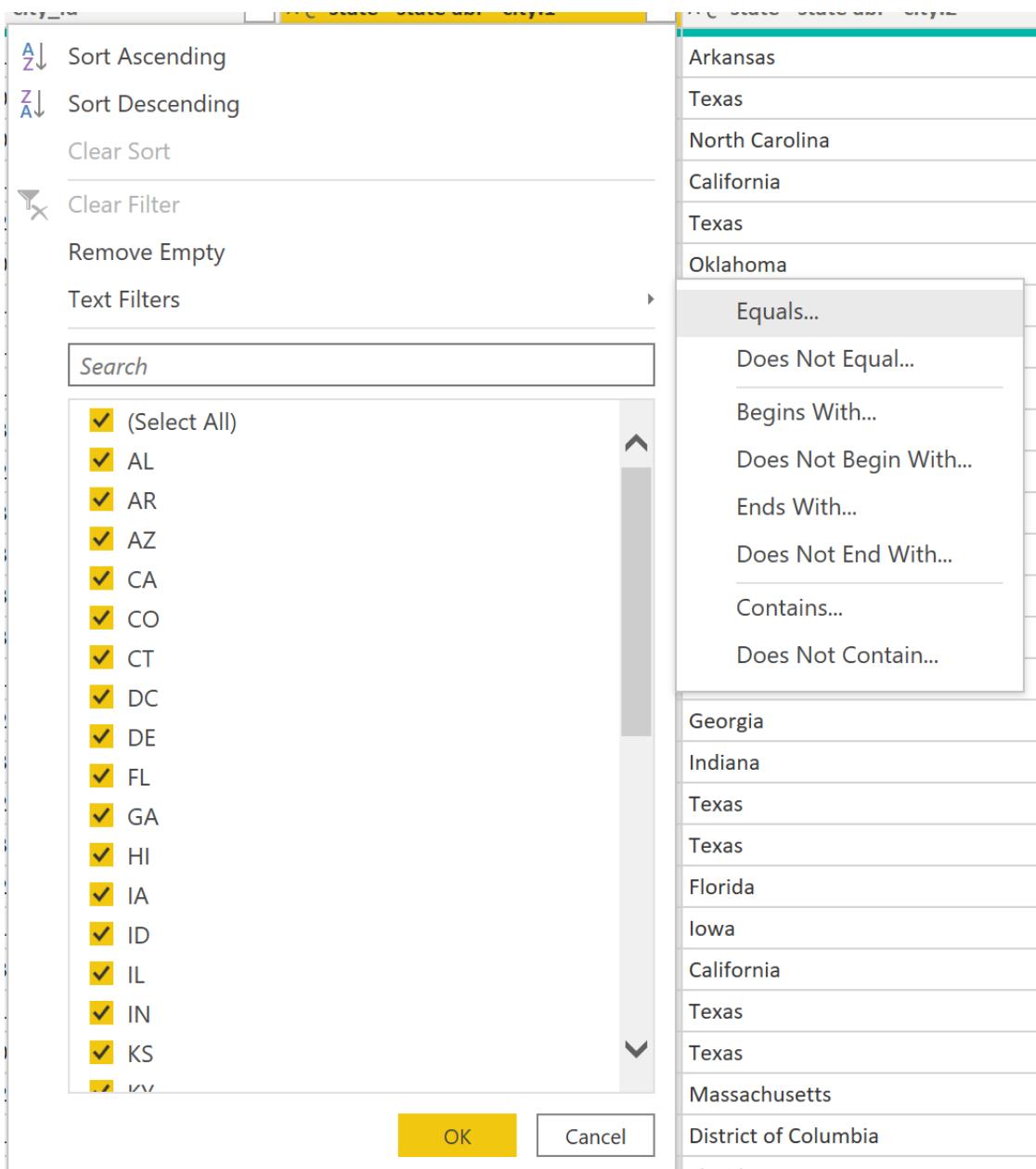
Suggested Values: List of values

1	CA
2	NY
*	

Default Value: CA

Current Value: CA

We can use to filter our State columns for example



Filter Rows

X

Apply one or more filter conditions to the rows in this table.

Basic Advanced

Keep rows where 'state - state abr - city.1'

equals	A ^B C	Enter or select a value
<input checked="" type="radio"/> And	<input type="radio"/> Or	

A^BC Text
Parameter
New Parameter... value

OK Cancel

Query Parameter

Create a List Query

A ^B C state - state abr - city.1	A ^B C state - state abr - city.2
AR	Copy
TX	Remove
NC	Remove Other Columns
CA	Duplicate Column
TX	Add Column From Examples...
OK	Remove Duplicates
OH	Remove Errors
FL	Change Type
DE	Transform
GA	1 ↪ 2 Replace Values...
CO	Replace Errors...
GA	Split Column
CA	Group By...
FL	Fill
AL	Unpivot Columns
TX	Unpivot Other Columns
GA	Unpivot Only Selected Columns
IN	Rename...
TX	Move
TX	Drill Down
TX	Add as New Query

X

Manage Parameters

New

1²3 rateA^BC statesA^BC states_list X

Name

states_list

Description

 Required

Type

Any

Suggested Values

Query

Query ⓘ

state - state abr - city 1

Current Value

CA

OK

Cancel