

Contents

[Transform, shape, and model data in Power BI](#)

[Overview](#)

[Query Editor overview](#)

[Tutorials](#)

[Create your own measures](#)

[Create calculated columns](#)

[Concepts](#)

[Work with Modeling view](#)

[Understand model relationships](#)

[Many-to-many relationships](#)

[Enable bidirectional cross-filtering](#)

[Composite models](#)

[User-defined aggregations](#)

[Manage storage mode](#)

[Work with multidimensional models in Power BI](#)

[Dataflows](#)

[Introduction to dataflows](#)

[Create a dataflow](#)

[Configure and consume a dataflow](#)

[Configure Power BI Premium dataflow workloads](#)

[Dataflows and Azure Data Lake Gen2](#)

[Premium features of dataflows](#)

[Use Machine Learning and Cognitive Services with dataflows](#)

[Dataflows best practices](#)

[Understanding and optimizing dataflow refresh](#)

[Using DirectQuery with dataflows](#)

[Dataflows considerations and limitations](#)

[Streaming dataflows \(preview\)](#)

[Developing dataflows solutions](#)

Log analytics

- [Using Azure Log Analytics in Power BI \(preview\)](#)
- [Configuring Azure Log Analytics in Power BI \(preview\)](#)
- [Azure Log Analytics in Power BI - FAQ \(preview\)](#)
- [Installing the Log Analytics Template App \(preview\)](#)
- [Using the Log Analytics Template App \(preview\)](#)

How-to guides

- [Perform common query tasks](#)
- [Create and manage relationships](#)
- [Use the Analytics pane](#)
- [Work with Data view](#)
- [Apply DAX basics](#)
- [Work with Relationship view](#)
- [Transform and shape data](#)
 - [Combine files \(binaries\)](#)
 - [Use AI Insights](#)
- [Model your data](#)
 - [Create quick measures for common calculations](#)
 - [Create and use what-if parameters to visualize variables](#)
 - [Specify data categories in Power BI Desktop](#)
 - [Tag barcode fields in Power BI for the mobile app](#)
 - [Set geographic filters in Power BI for the mobile app](#)
 - [Create calculated columns](#)
 - [Create calculated tables](#)
 - [Create measures for data analysis](#)
 - [Import and display KPIs](#)
 - [Apply auto date/time](#)
- [External tools](#)
 - [External tools in Power BI Desktop](#)
 - [Register an external tool](#)
- [Reference](#)
 - [Use the Field list in Power BI Desktop](#)

Formula editor shortcuts

Query overview in Power BI Desktop

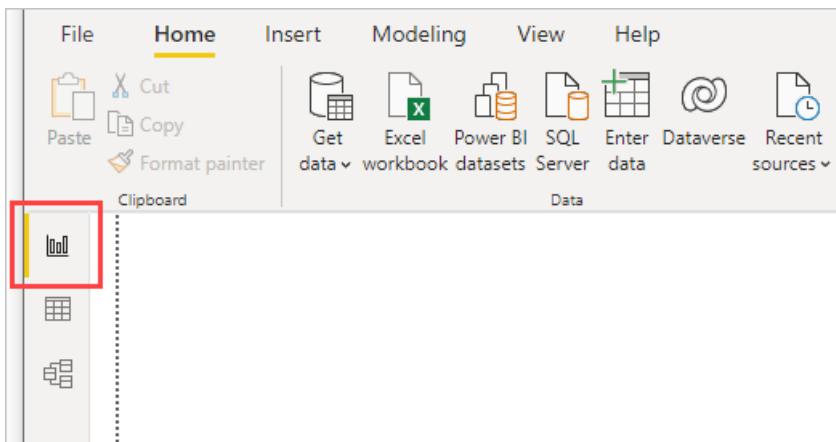
3/30/2022 • 5 minutes to read • [Edit Online](#)

With Power BI Desktop you can connect to the world of data, create compelling and foundational reports, and share your efforts with others – who can then build on your work, and expand their business intelligence efforts.

Power BI Desktop has three views:

- **Report view** – where you use queries you create to build compelling visualizations, arranged as you want them to appear, and with multiple pages, that you can share with others
- **Data view** – see the data in your report in data model format, where you can add measures, create new columns, and manage relationships
- **Relationships view** – get a graphical representation of the relationships that have been established in your data model, and manage or modify them as needed.

Access these views by selecting one of the three icons along the left side of Power BI Desktop. In the following image, **Report view** is selected, indicated by the yellow band beside the icon.



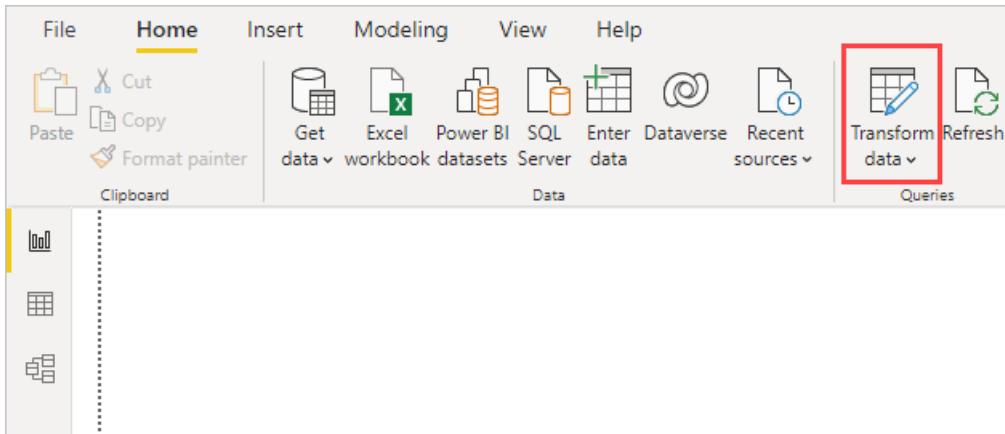
Power BI Desktop also comes with Power Query Editor. Use Power Query Editor to connect to one or many data sources, shape and transform the data to meet your needs, then load that model into Power BI Desktop.

This document provides an overview of the work with data in the Power Query Editor, but there's more to learn. At the end of this document, you'll find links to detailed guidance about supported data types. You'll also find guidance about connecting to data, shaping data, creating relationships, and how to get started.

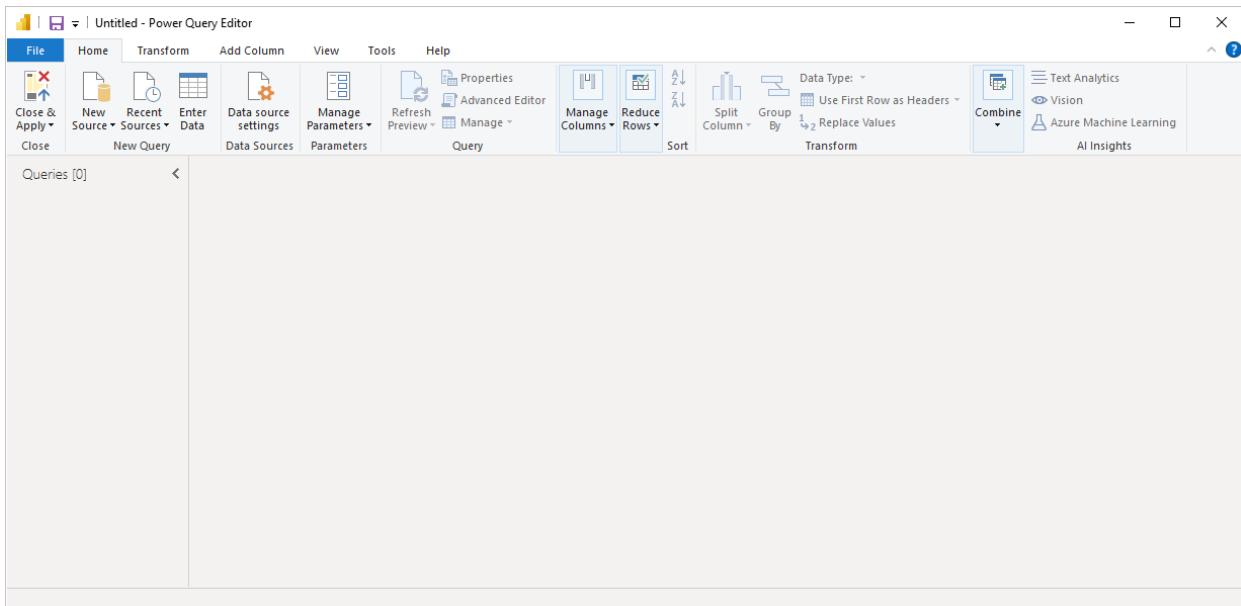
But first, let's see get acquainted with Power Query Editor.

Power Query Editor

To get to Power Query Editor, select **Transform data** from the **Home** tab of Power BI Desktop.



With no data connections, Power Query Editor appears as a blank pane, ready for data.



Once a query is loaded, Power Query Editor view becomes more interesting. If we connect to the following Web data source, Power Query Editor loads information about the data, which you can then begin to shape:

<https://www.bankrate.com/retirement/best-and-worst-states-for-retirement/>

Here's how Power Query Editor appears once a data connection is established:

1. In the ribbon, many buttons are now active to interact with the data in the query.
2. In the left pane, queries are listed and available for selection, viewing, and shaping.
3. In the center pane, data from the selected query is displayed and available for shaping.
4. The **Query Settings** pane appears, listing the query's properties and applied steps.

The screenshot shows the Power Query Editor interface. The ribbon is at the top with tabs: File, Home (highlighted), Transform, Add Column, View, Tools, Help. The Home tab has several icons: Close & Apply, New Source, Recent Sources, Enter Data, Data source settings, Manage Parameters, Refresh Preview, Advanced Editor, Properties, Manage, Manage Columns, Reduce Rows, Sort, Split Column, Group By, Replace Values, and Transform. The main area is the Data view, which displays a table titled 'Best States to Retire' with 17 rows of data. The Query Settings pane on the right shows the query name 'Best States to Retire' and the applied step 'Changed Type'.

We'll look at each of these four areas: the ribbon, the Queries pane, the Data view, and the Query Settings pane.

The query ribbon

The ribbon in Power Query Editor consists of four tabs: **Home**, **Transform**, **Add Column**, **View**, **Tools**, and **Help**.

The **Home** tab contains the common query tasks.

The screenshot shows the Power Query Editor ribbon with the Home tab selected. A red box highlights the 'New Source' button under the 'File' tab. The ribbon also includes File, Home, Transform, Add Column, View, Tools, and Help tabs. Below the ribbon is a dropdown menu for 'New Source' with options like Excel Workbook, SQL Server, Analysis Services, Text/CSV, Web, OData feed, Blank Query, and More... A preview pane shows a table titled 'Retire'.

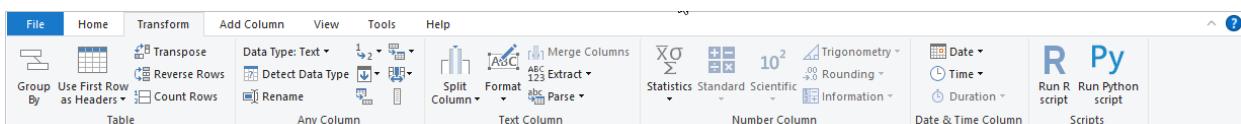
To connect to data and begin the query building process, select **New Source**. A menu appears, providing the most common data sources.

The screenshot shows the 'New Source' dropdown menu from the previous screenshot. It lists the most common data sources: Excel Workbook, SQL Server, Analysis Services, Text/CSV, Web, OData feed, Blank Query, and More... Each item has a corresponding icon next to it.

For more information about available data sources, see [Data Sources](#). For information about connecting to data, including examples and steps, see [Connect to Data](#).

The **Transform** tab provides access to common data transformation tasks, such as:

- Adding or removing columns
- Changing data types
- Splitting columns
- Other data-driven tasks

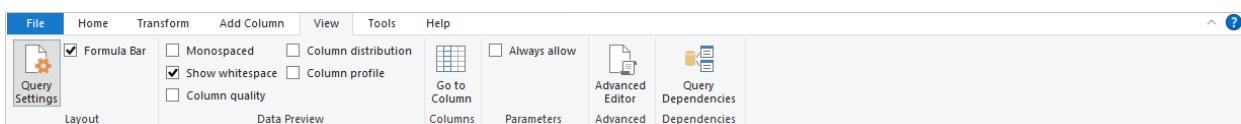


For more information about transforming data, including examples, see [Tutorial: Shape and combine data in Power BI Desktop](#).

The **Add Column** tab provides additional tasks associated with adding a column, formatting column data, and adding custom columns. The following image shows the **Add Column** tab.



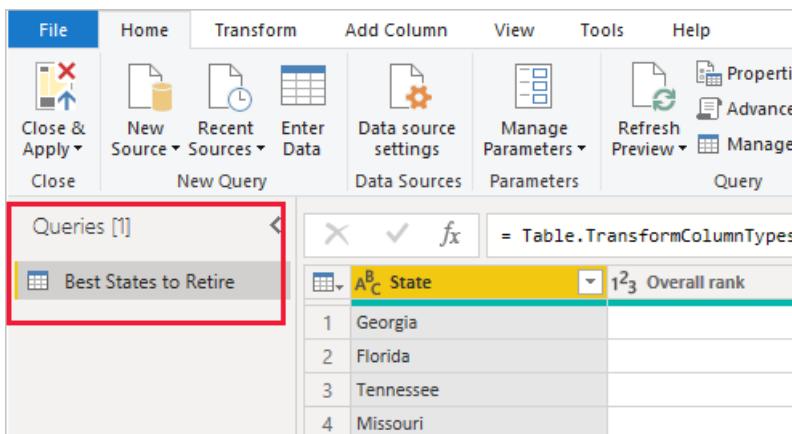
The **View** tab on the ribbon is used to toggle whether certain panes or windows are displayed. It's also used to display the Advanced Editor. The following image shows the **View** tab.



It's useful to know that many of the tasks available from the ribbon are also available by right-clicking a column, or other data, in the center pane.

The left (Queries) pane

The left pane, or **Queries** pane, displays the number of active queries and the name of the query. When you select a query from the left pane, its data is displayed in the center pane, where you can shape and transform the data to meet your needs. The following image shows the left pane with a query.



The center (Data) pane

In the center pane, or **Data** pane, data from the selected query is displayed. This pane is where much of the work of the **Query** view is accomplished.

In the following image shows the Web data connection established earlier. The **Overall score** column is selected, and its header is right-clicked to show the available menu items. Notice that many of these right-click

menu items are the same as buttons in the ribbon tabs.

The screenshot shows the Power Query Editor interface. A context menu is open over the 'Overall score' column in a table. The menu includes options like 'Remove', 'Remove Other Columns', 'Duplicate Column', 'Add Column From Examples...', 'Remove Duplicates', 'Remove Errors', 'Change Type' (which is currently set to 'Decimal Number'), 'Transform' (with sub-options like 'Replace Values...', 'Replace Errors...', 'Group By...', 'Fill', 'Unpivot Columns', 'Unpivot Other Columns', and 'Unpivot Only Selected Columns'), and 'Sort'. The 'Change Type' option is highlighted. To the right of the table, the 'Query Settings' pane is visible, showing the 'APPLIED STEPS' section which lists 'Source', 'Extracted Table From Html', 'Promoted Headers', and 'Changed Type' (which is also highlighted).

When you select a right-click menu item (or a ribbon button), the query applies the step to the data. It also saves step as part of the query itself. The steps are recorded in the **Query Settings** pane in sequential order, as described in the next section.

The right (Query Settings) pane

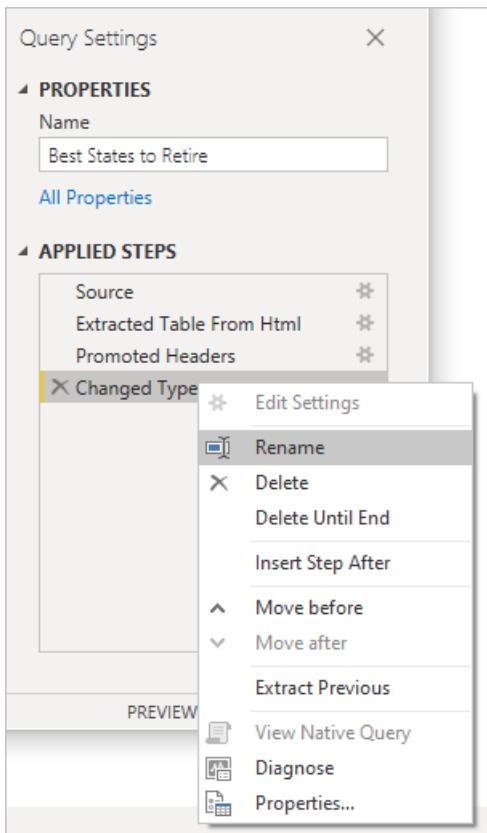
The right pane, or **Query Settings** pane, is where all steps associated with a query are displayed. For example, in the following image, the **Applied Steps** section of the **Query Settings** pane reflects the fact that we just changed the type of the **Overall score** column.

The screenshot shows the 'Query Settings' pane. The 'APPLIED STEPS' section contains the following steps listed sequentially: 'Source', 'Extracted Table From Html', 'Promoted Headers', and 'Changed Type' (which is highlighted with a yellow border). The 'Properties' section shows the query is named 'Best States to Retire'.

As additional shaping steps are applied to the query, they're captured in the **Applied Steps** section.

It's important to know that the underlying data *isn't* changed. Rather, Power Query Editor adjusts and shapes its view of the data. It also shapes and adjusts the view of any interaction with the underlying data that occurs based on Power Query Editor's shaped and modified view of that data.

In the **Query Settings** pane, you can rename steps, delete steps, or reorder the steps as you see fit. To do so, right-click the step in the **Applied Steps** section, and choose from the menu that appears. All query steps are carried out in the order they appear in the **Applied Steps** pane.



Advanced Editor

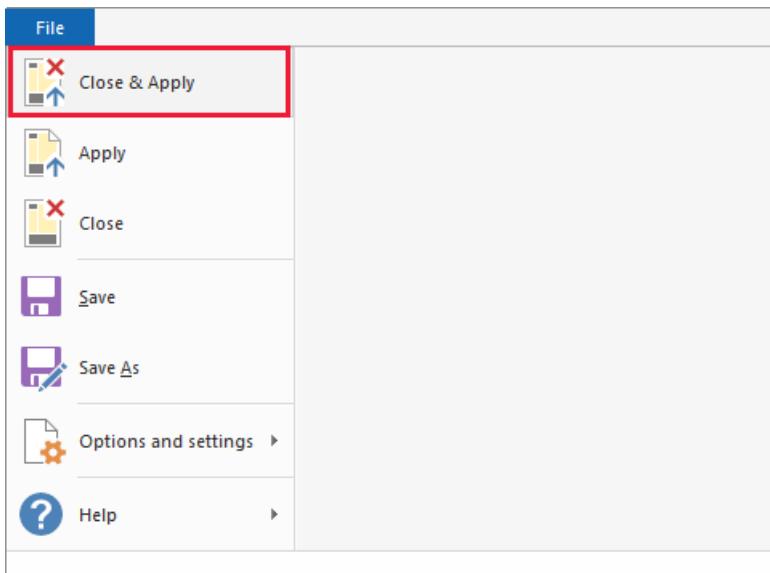
The **Advanced Editor** lets you see the code that Power Query Editor is creating with each step. It also lets you create your own shaping code. To launch the advanced editor, select **View** from the ribbon, then select **Advanced Editor**. A window appears, showing the existing query code.



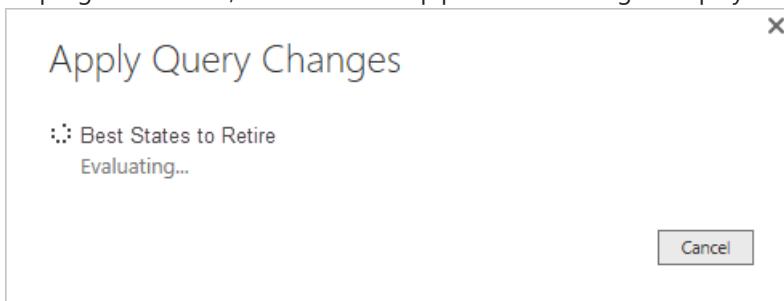
You can directly edit the code in the **Advanced Editor** window. To close the window, select the **Done** or **Cancel** button.

Saving your work

When your query is where you want it, select **Close & Apply** from Power Query Editor's **File** menu. This action applies the changes and closes the editor.

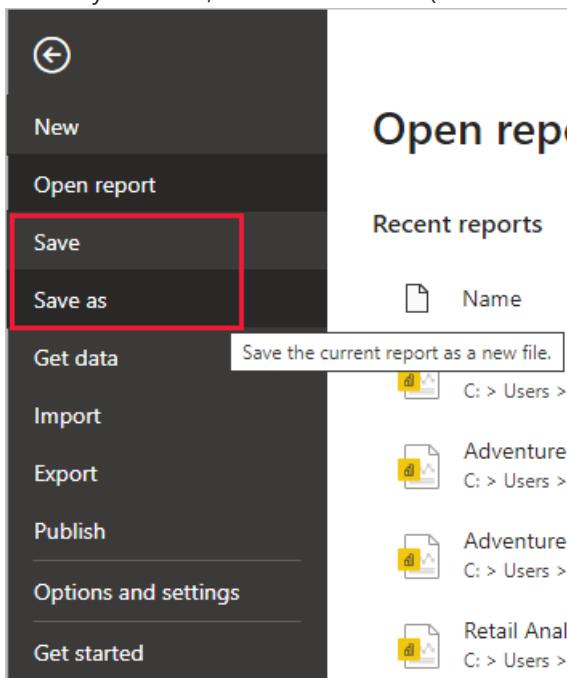


As progress is made, Power BI Desktop provides a dialog to display its status.



When you're ready, Power BI Desktop can save your work in the form of a .pbix file.

To save your work, select **File** > **Save** (or **File** > **Save As**), as shown in the following image.



Next steps

There are all sorts of things you can do with Power BI Desktop. For more information on its capabilities, check out the following resources:

- [What is Power BI Desktop?](#)
- [Data sources in Power BI Desktop](#)

- [Connect to data in Power BI Desktop](#)
- [Tutorial: Shape and combine data with Power BI Desktop](#)
- [Perform common query tasks in Power BI Desktop](#)

Tutorial: Create your own measures in Power BI Desktop

3/30/2022 • 11 minutes to read • [Edit Online](#)

By using measures, you can create some of the most powerful data analysis solutions in Power BI Desktop. Measures help you by performing calculations on your data as you interact with your reports. This tutorial will guide you through understanding measures and creating your own basic measures in Power BI Desktop.

Prerequisites

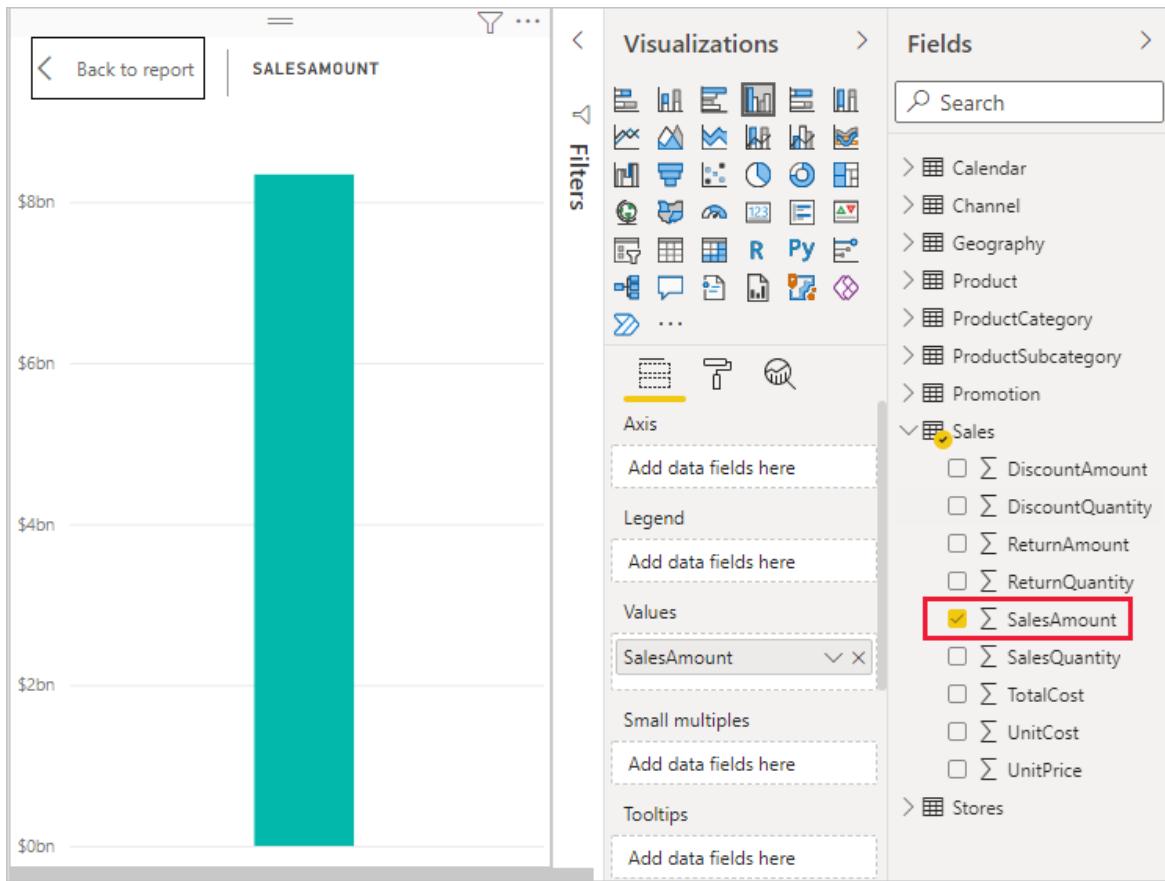
- This tutorial is intended for Power BI users already familiar with using Power BI Desktop to create more advanced models. You should already be familiar with using Get Data and Power Query Editor to import data, work with multiple related tables, and add fields to the report canvas. If you're new to Power BI Desktop, be sure to check out [Getting Started with Power BI Desktop](#).
- This tutorial uses the [Contoso Sales Sample for Power BI Desktop](#) file, which includes online sales data from the fictitious company, Contoso. Because this data is imported from a database, you can't connect to the datasource or view it in Power Query Editor. Download and extract the file on your computer.

Automatic measures

When Power BI Desktop creates a measure, it's most often created for you automatically. To see how Power BI Desktop creates a measure, follow these steps:

1. In Power BI Desktop, select **File** > **Open**, browse to the *Contoso Sales Sample for Power BI Desktop.pbix* file, and then select **Open**.
2. In the **Fields** pane, expand the **Sales** table. Then, either select the check box next to the **SalesAmount** field or drag **SalesAmount** onto the report canvas.

A new column chart visualization appears, showing the sum total of all values in the **SalesAmount** column of the **Sales** table.

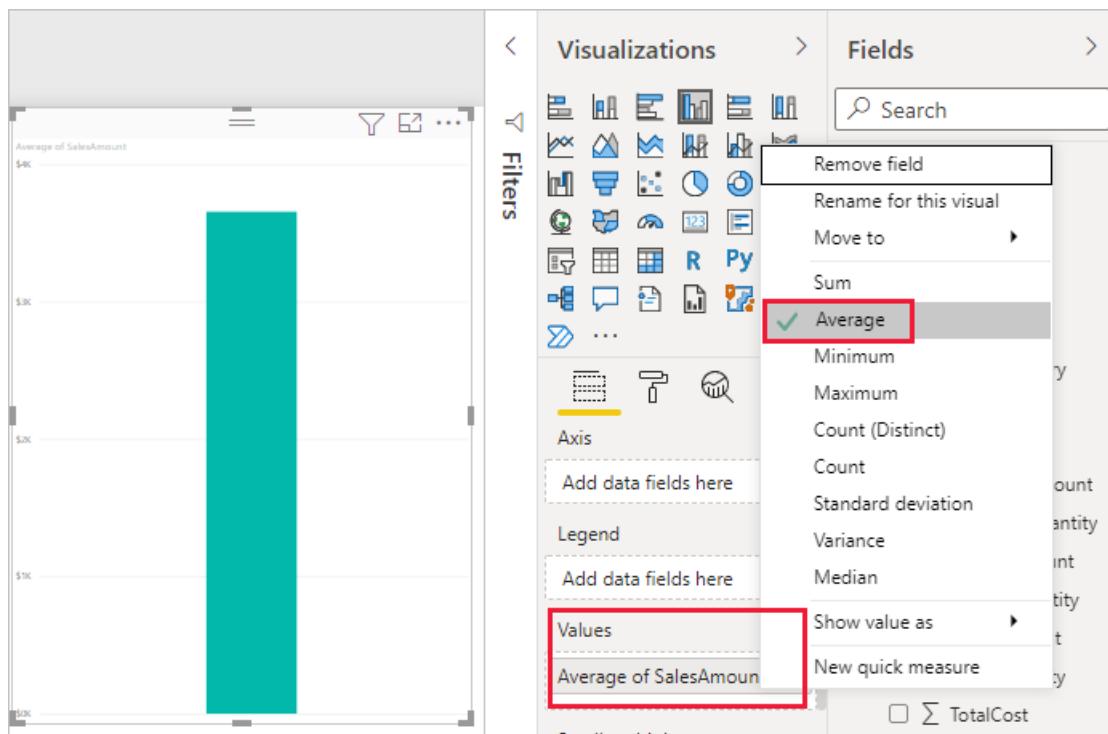


Any field (column) in the **Fields** pane with a sigma icon Σ is numeric, and its values can be aggregated. Rather than display a table with many values (two million rows for **SalesAmount**), Power BI Desktop automatically creates and calculates a measure to aggregate the data if it detects a numeric datatype. Sum is the default aggregation for a numeric datatype, but you can easily apply different aggregations like average or count. Understanding aggregations is fundamental to understanding measures, because every measure performs some type of aggregation.

To change the chart aggregation, follow these steps:

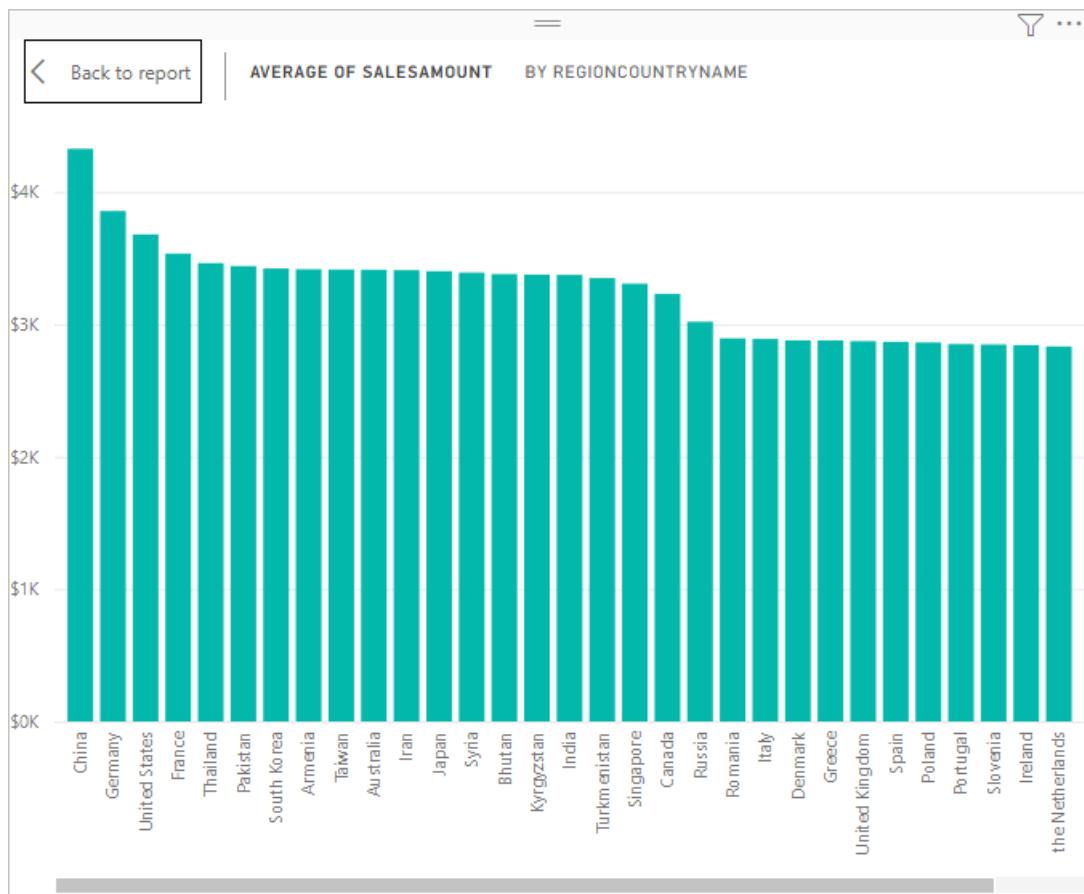
1. Select the **SalesAmount** visualization in the report canvas.
2. In the **Values** area of the **Visualizations** pane, select the down arrow to the right of **SalesAmount**.
3. From the menu that appears, select **Average**.

The visualization changes to an average of all sales values in the **SalesAmount** field.



Depending on the result you want, you can change the type of aggregation. However, not all types of aggregation apply to every numeric datatype. For example, for the **SalesAmount** field, Sum and Average are useful, and Minimum and Maximum have their place as well. However, Count doesn't make sense for the **SalesAmount** field, because while its values are numeric, they're really currency.

Values calculated from measures change in response to your interactions with your report. For example, if you drag the **RegionCountryName** field from the **Geography** table onto your existing **SalesAmount** chart, it changes to show the average sales amounts for each country.



When the result of a measure changes because of an interaction with your report, you've affected your measure's *context*. Every time you interact with your report visualizations, you're changing the context in which

a measure calculates and displays its results.

Create and use your own measures

In most cases, Power BI Desktop automatically calculates and returns values according to the types of fields and aggregations you choose. However, in some cases you might want to create your own measures to perform more complex, unique calculations. With Power BI Desktop, you can create your own measures with the Data Analysis Expressions (DAX) formula language.

DAX formulas use many of the same functions, operators, and syntax as Excel formulas. However, DAX functions are designed to work with relational data and perform more dynamic calculations as you interact with your reports. There are over 200 DAX functions that do everything from simple aggregations like sum and average to more complex statistical and filtering functions. There are many resources to help you learn more about DAX. After you've finished this tutorial, see [DAX basics in Power BI Desktop](#).

When you create your own measure, it's called a *model* measure, and it's added to the **Fields** list for the table you select. Some advantages of model measures are that you can name them whatever you want, making them more identifiable; you can use them as arguments in other DAX expressions; and you can make them perform complex calculations quickly.

Quick measures

Many common calculations are available as *quick measures*, which write the DAX formulas for you based on your inputs in a window. These quick, powerful calculations are also great for learning DAX or seeding your own customized measures.

Create a quick measure using one of these methods:

- From a table in the **Fields** pane, right-click or select **More options (...)**, and then select **New quick measure** from the list.
- Under **Calculations** in the **Home** tab of the Power BI Desktop ribbon, select **New Quick Measure**.

For more information about creating and using quick measures, see [Use quick measures](#).

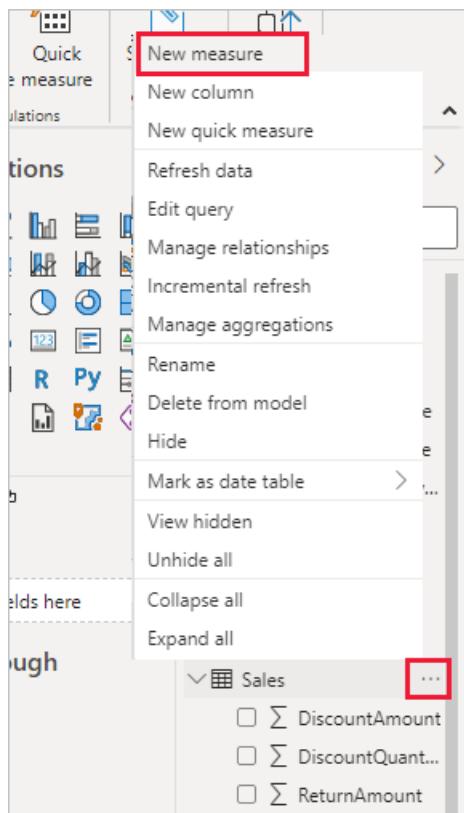
Create a measure

Suppose you want to analyze your net sales by subtracting discounts and returns from total sales amounts. For the context that exists in your visualization, you need a measure that subtracts the sum of **DiscountAmount** and **ReturnAmount** from the sum of **SalesAmount**. There's no field for Net Sales in the **Fields** list, but you have the building blocks to create your own measure to calculate net sales.

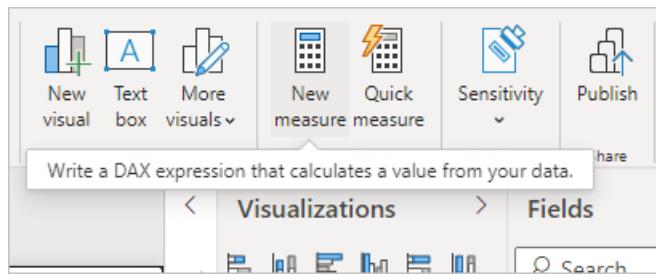
To create a measure, follow these steps:

- In the **Fields** pane, right-click the **Sales** table, or hover over the table and select **More options (...)**.
- From the menu that appears, select **New measure**.

This action saves your new measure in the **Sales** table, where it's easy to find.



You can also create a new measure by selecting **New Measure** in the **Calculations** group on the **Home** tab of the Power BI Desktop ribbon.



TIP

When you create a measure from the ribbon, you can create it in any of your tables, but it's easier to find if you create it where you plan to use it. In this case, select the **Sales** table first to make it active, and then select **New measure**.

The formula bar appears along the top of the report canvas, where you can rename your measure and enter a DAX formula.

The screenshot shows the Power BI Data Editor interface. The formula bar at the top has the text "1 Measure =". To the right is a search bar and a dropdown menu. Below the formula bar is a message: "Build visuals with your data. Select or drag fields from the Fields pane onto the report canvas." A small icon of a box with arrows is shown. On the right side, there's a "Fields" pane titled "Visualizations". It contains a tree view of data: Calendar, Channel, Geography (with ContinentName, GeographyType, RegionCountry...), Product (with ProductCategory, ProductSubcategory, Promotion), and Sales (with various sum and count measures). The "Measure" item under Sales is highlighted with a red rectangle.

3. By default, each new measure is named *Measure*. If you don't rename it, additional new measures are named *Measure 2*, *Measure 3*, and so on. Because we want this measure to be more identifiable, highlight *Measure* in the formula bar, and then change it to *Net Sales*.
4. Begin entering your formula. After the equals sign, start to type *Sum*. As you type, a drop-down suggestion list appears, showing all the DAX functions, beginning with the letters you type. Scroll down, if necessary, to select **SUM** from the list, and then press **Enter**.

The screenshot shows the Power BI Data Editor with the formula bar containing "1 Net Sales = SUM(". A dropdown menu is open, listing various DAX functions: ISSELECTEDMEASURE, ISSUBTOTAL, NONVISUAL, SELECTEDMEASURE, SELECTEDMEASUREFORMATSTRING, SELECTEDMEASURENAME, SUBSTITUTE, SUBSTITUTEWITHINDEX, SUM, SUMMARIZE, and SUMMARIZECOLUMNS. The "SUM" function is highlighted with a blue selection bar.

An opening parenthesis appears, along with a drop-down suggestion list of the available columns you can pass to the SUM function.

The screenshot shows the Power BI Data Editor with the formula bar containing "1 Net Sales = SUM(" followed by a dropdown menu. The menu title is "SUM(Column Name)" with the subtitle "Adds all the numbers in a column." Below the title is a list of columns from the 'Calendar' table: 'Calendar'[SalesAmount], 'Calendar'[DateInt], 'Calendar'[DateKey], 'Calendar'[DayOfMonth], 'Calendar'[DayOfWeek Name], and 'Calendar'[Month Name]. The 'Calendar'[SalesAmount] column is highlighted with a blue selection bar.

5. Expressions always appear between opening and closing parentheses. For this example, your expression contains a single argument to pass to the SUM function: the **SalesAmount** column. Begin typing **SalesAmount** until **Sales(SalesAmount)** is the only value left in the list.

The column name preceded by the table name is called the fully qualified name of the column. Fully qualified column names make your formulas easier to read.

The screenshot shows the Power BI formula bar. The formula is '1 Net Sales = SUM(salesA SUM(ColumnName))'. A tooltip for 'SUM(ColumnName)' states: 'Adds all the numbers in a column.' Below the tooltip is a small icon of a table labeled 'Sales[SalesAmount]'.

6. Select **Sales[SalesAmount]** from the list, and then enter a closing parenthesis.

TIP

Syntax errors are most often caused by a missing or misplaced closing parenthesis.

7. Subtract the other two columns inside the formula:

- After the closing parenthesis for the first expression, type a space, a minus operator (-), and then another space.
- Enter another SUM function, and start typing *DiscountAmount* until you can choose the **Sales[DiscountAmount]** column as the argument. Add a closing parenthesis.
- Type a space, a minus operator, a space, another SUM function with **Sales[ReturnAmount]** as the argument, and then a closing parenthesis.

The screenshot shows the Power BI formula bar with the completed formula: '1 Net Sales = SUM(Sales[SalesAmount]) - SUM(Sales[DiscountAmount]) - SUM(Sales[ReturnAmount])'.

8. Press **Enter** or select **Commit** (checkmark icon) in the formula bar to complete and validate the formula.

The validated **Net Sales** measure is now ready to use in the **Sales** table in the **Fields** pane.

The screenshot shows the Power BI Fields pane. The 'Sales' table is expanded, showing various measures like 'DiscountAmount', 'Net Sales', 'ReturnAmount', etc. The 'Net Sales' measure is selected, indicated by a highlighted row.

9. If you run out of room for entering a formula or want it on separate lines, select the down arrow on the right side of the formula bar to provide more space.

The down arrow turns into an up arrow and a large box appears.

The screenshot shows the Power BI formula bar. The down arrow on the right has been replaced by an up arrow, indicating a larger input area is available. The formula '1 Net Sales = SUM(Sales[SalesAmount]) - SUM(Sales[DiscountAmount]) - SUM(Sales[ReturnAmount])' is visible.

10. Separate parts of your formula by pressing **Alt + Enter** for separate lines, or pressing **Tab** to add tab spacing.

```

1 Net Sales = SUM(Sales[SalesAmount])
2   - SUM(Sales[DiscountAmount])
3   - SUM(Sales[ReturnAmount])

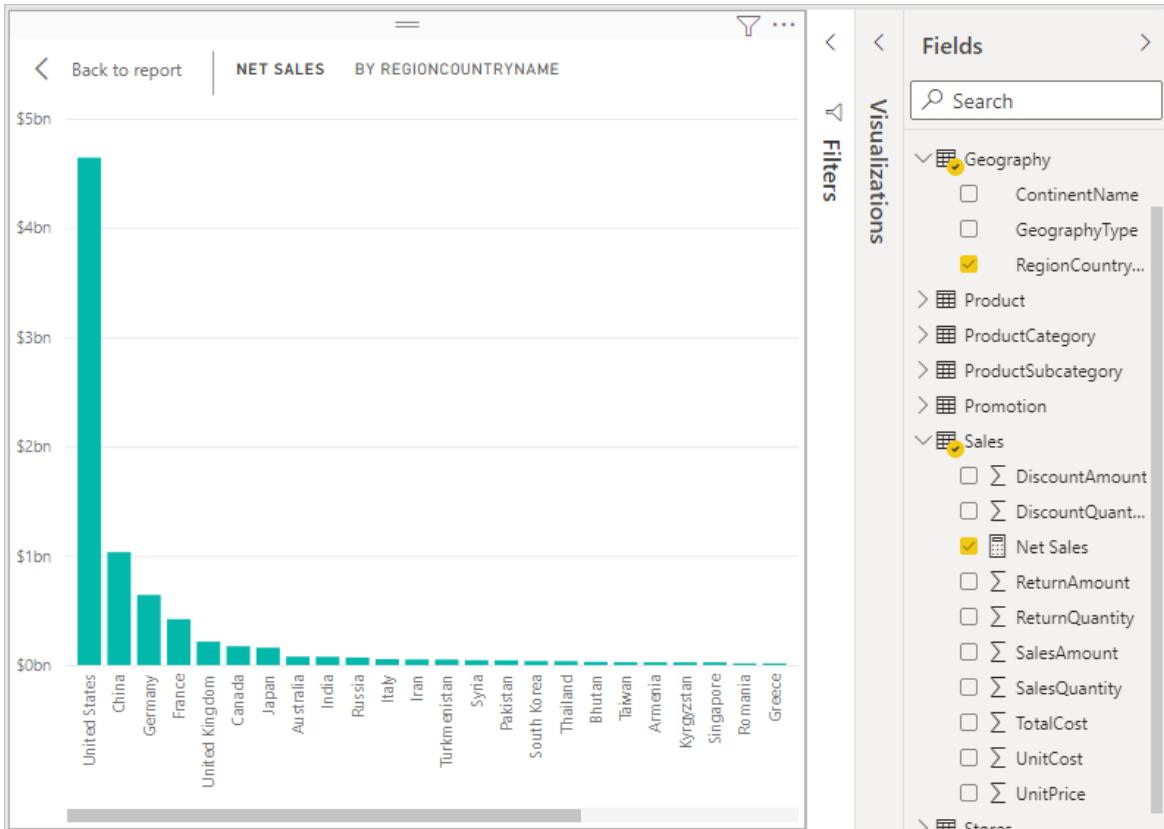
```

Use your measure in the report

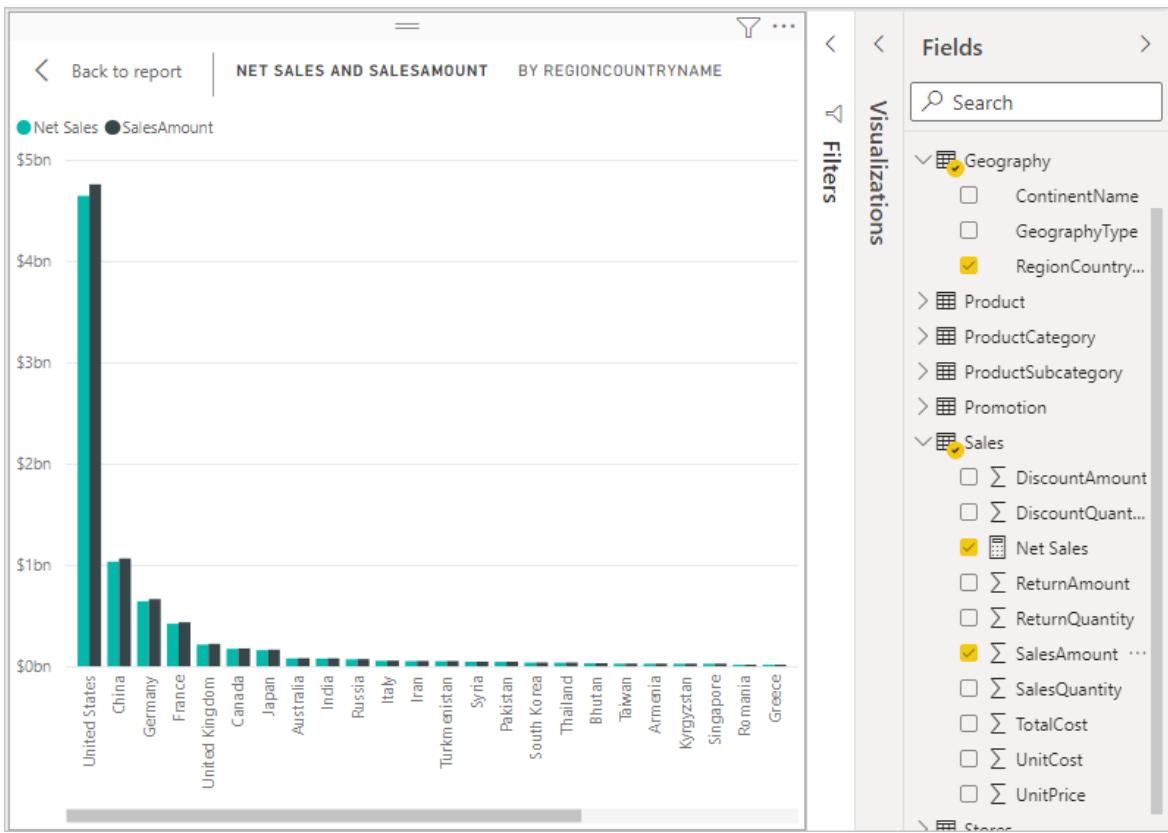
Add your new **Net Sales** measure to the report canvas, and calculate net sales for whatever other fields you add to the report.

To look at net sales by country:

1. Select the **Net Sales** measure from the **Sales** table, or drag it onto the report canvas.
2. Select the **RegionCountryName** field from the **Geography** table, or drag it onto the **Net Sales** chart.



3. To see the difference between net sales and total sales by country, select the **SalesAmount** field or drag it onto the chart.



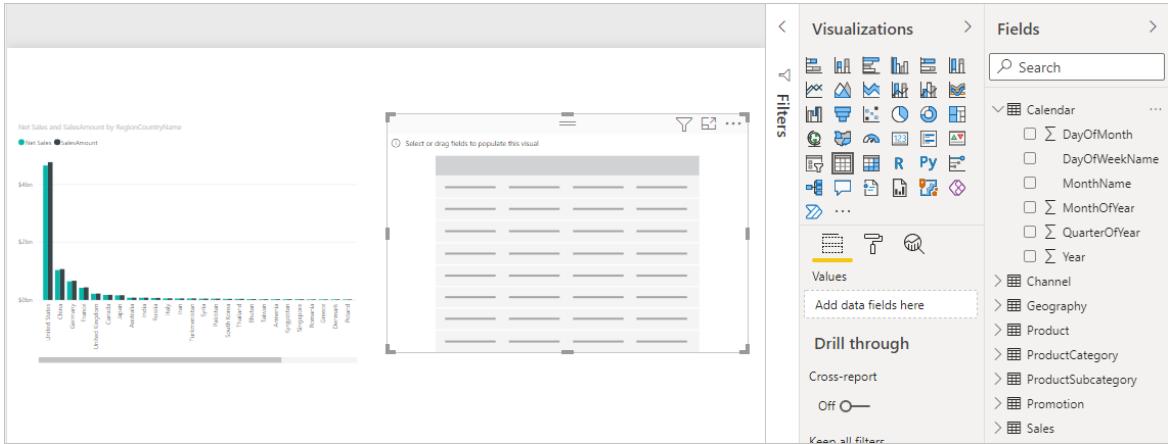
The chart now uses two measures: **SalesAmount**, which Power BI summed automatically, and the **Net Sales** measure, which you manually created. Each measure was calculated in the context of another field, **RegionCountryName**.

Use your measure with a slicer

Add a slicer to further filter net sales and sales amounts by calendar year:

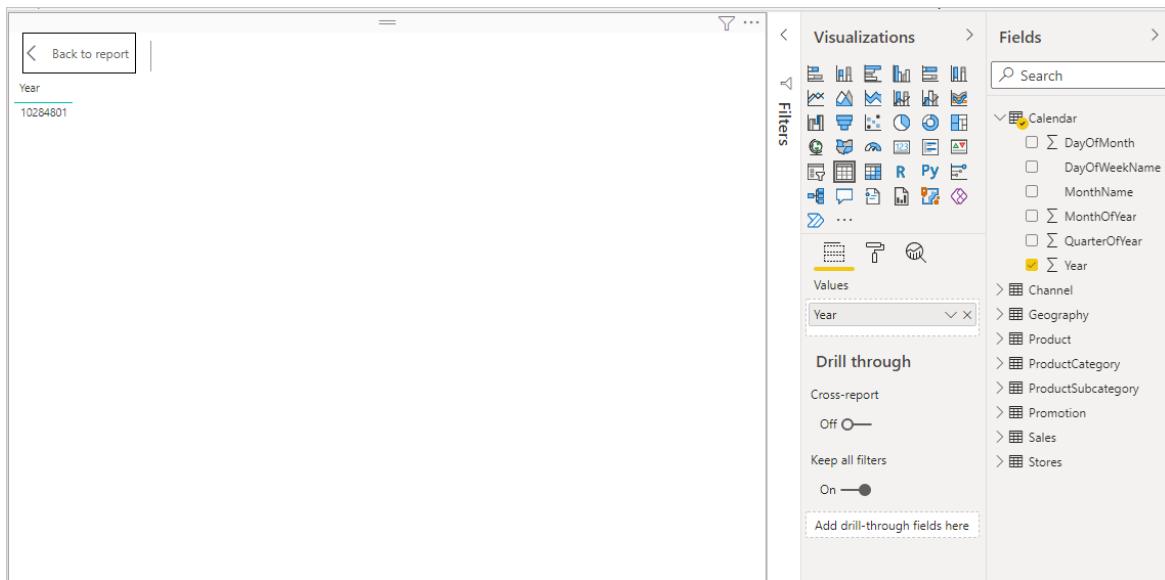
1. Select a blank area next to the chart. In the **Visualizations** pane, select the **Table** visualization.

This action creates a blank table visualization on the report canvas.

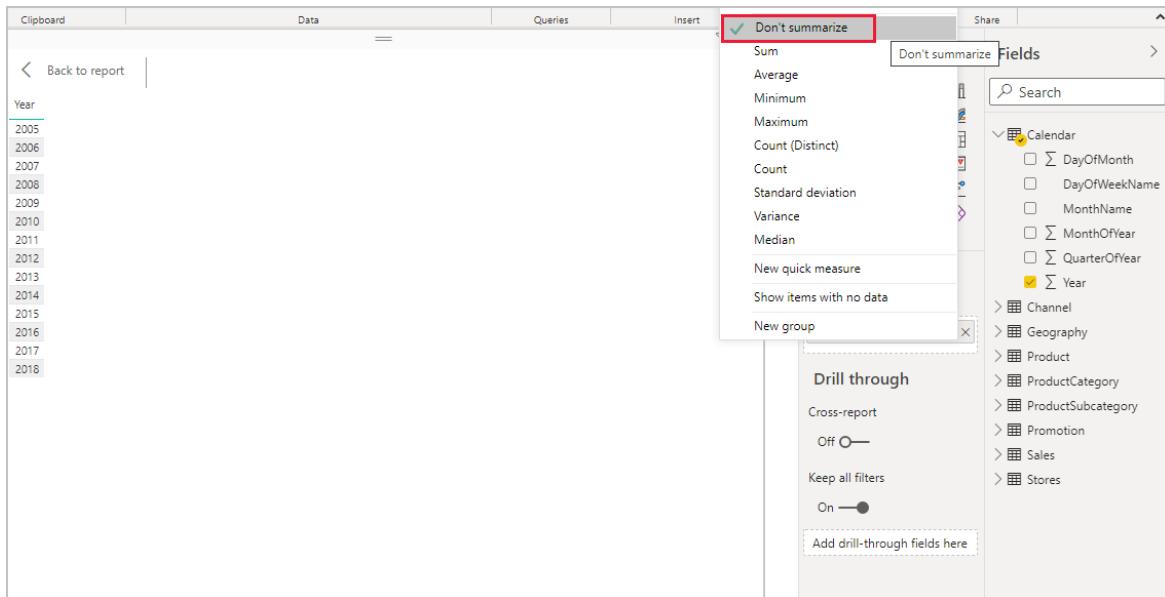


2. Drag the **Year** field from the **Calendar** table onto the new blank table visualization.

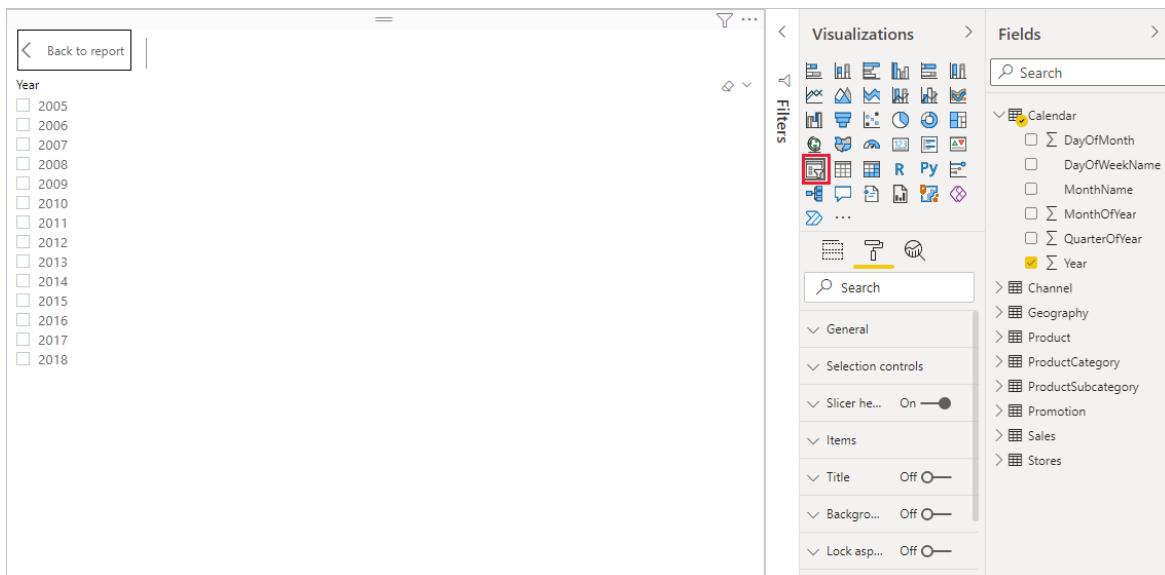
Because **Year** is a numeric field, Power BI Desktop sums up its values. This summation doesn't work well as an aggregation; we'll address that in the next step.



3. In the **Values** box in the **Visualizations** pane, select the down arrow next to **Year**, and then select **Don't summarize** from the list. The table now lists individual years.

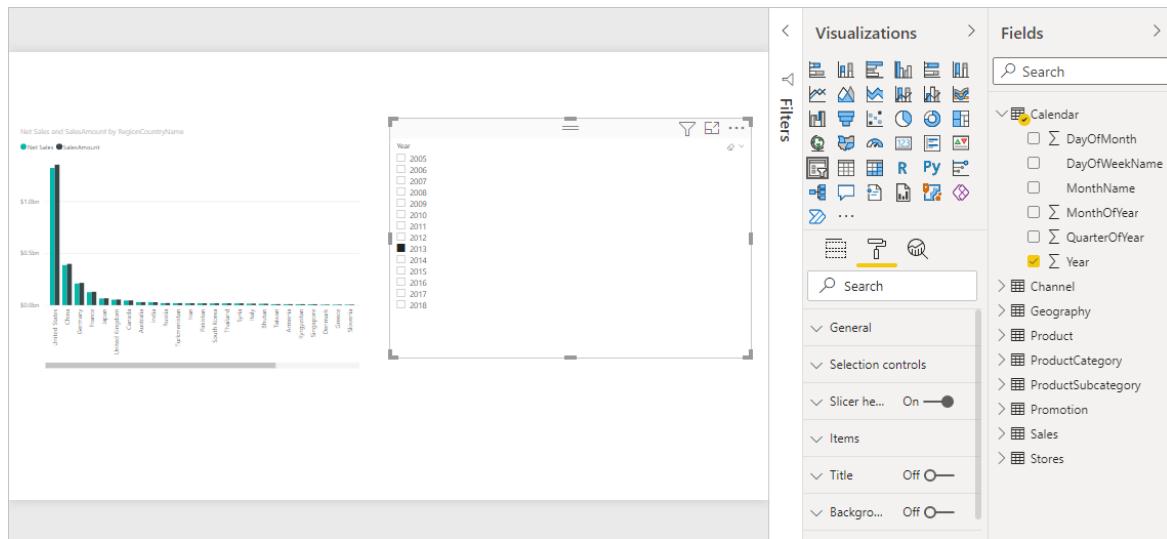


4. Select the **Slicer** icon in the **Visualizations** pane to convert the table to a slicer. If the visualization displays a slider instead of a list, select **List** from the down arrow in the slider.



5. Select any value in the **Year** slicer to filter the **Net Sales** and **Sales Amount** by

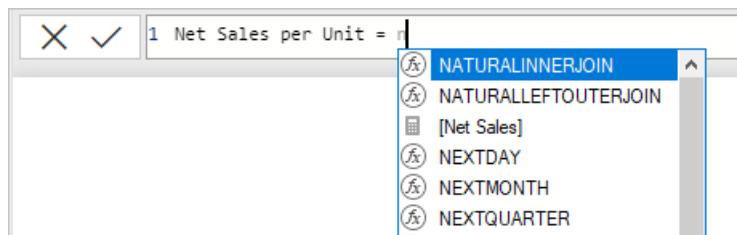
RegionCountryName chart accordingly. The **Net Sales** and **SalesAmount** measures recalculate and display results in the context of the selected **Year** field.



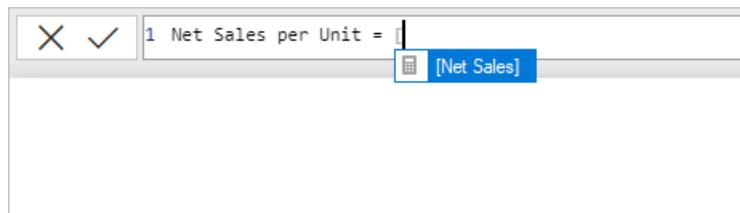
Use your measure in another measure

Suppose you want to find out which products have the highest net sales amount per unit sold. You'll need a measure that divides net sales by the quantity of units sold. Create a new measure that divides the result of your **Net Sales** measure by the sum of **Sales[SalesQuantity]**.

1. In the **Fields** pane, create a new measure named **Net Sales per Unit** in the **Sales** table.
2. In the formula bar, begin typing *Net Sales*. The suggestion list shows what you can add. Select [**Net Sales**].



3. You can also reference measures by just typing an opening bracket ([). The suggestion list shows only measures to add to your formula.



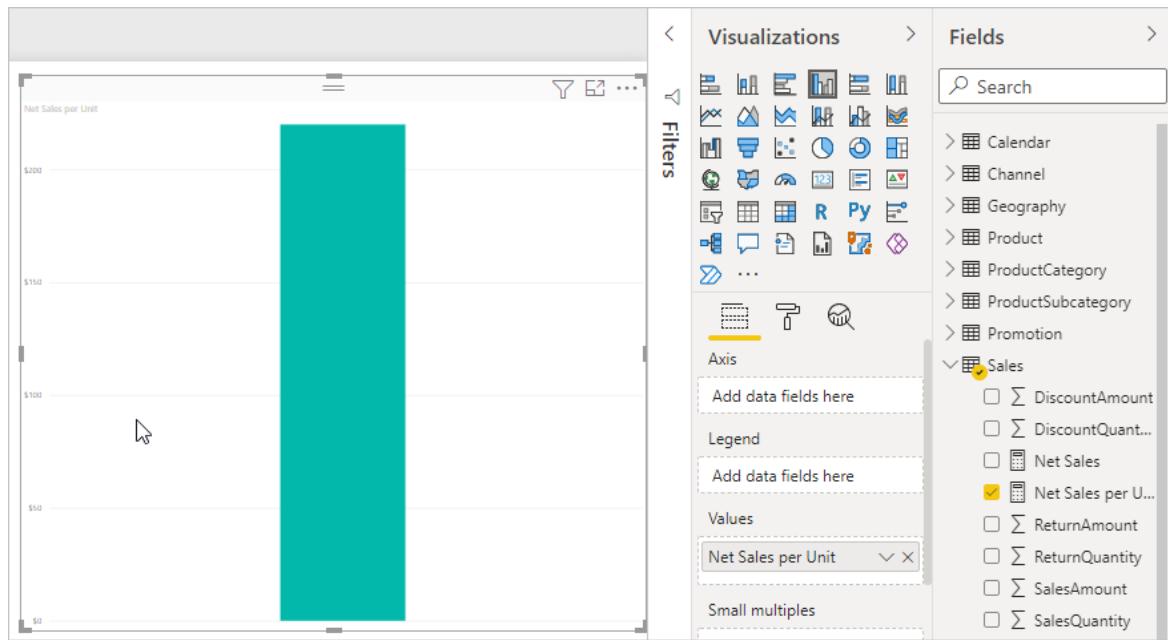
4. Enter a space, a divide operator (/), another space, a SUM function, and then type *Quantity*. The suggestion list shows all the columns with *Quantity* in the name. Select **Sales[SalesQuantity]**, type the closing parenthesis, and press **ENTER** or select **Commit** (checkmark icon) to validate your formula.

The resulting formula should appear as:

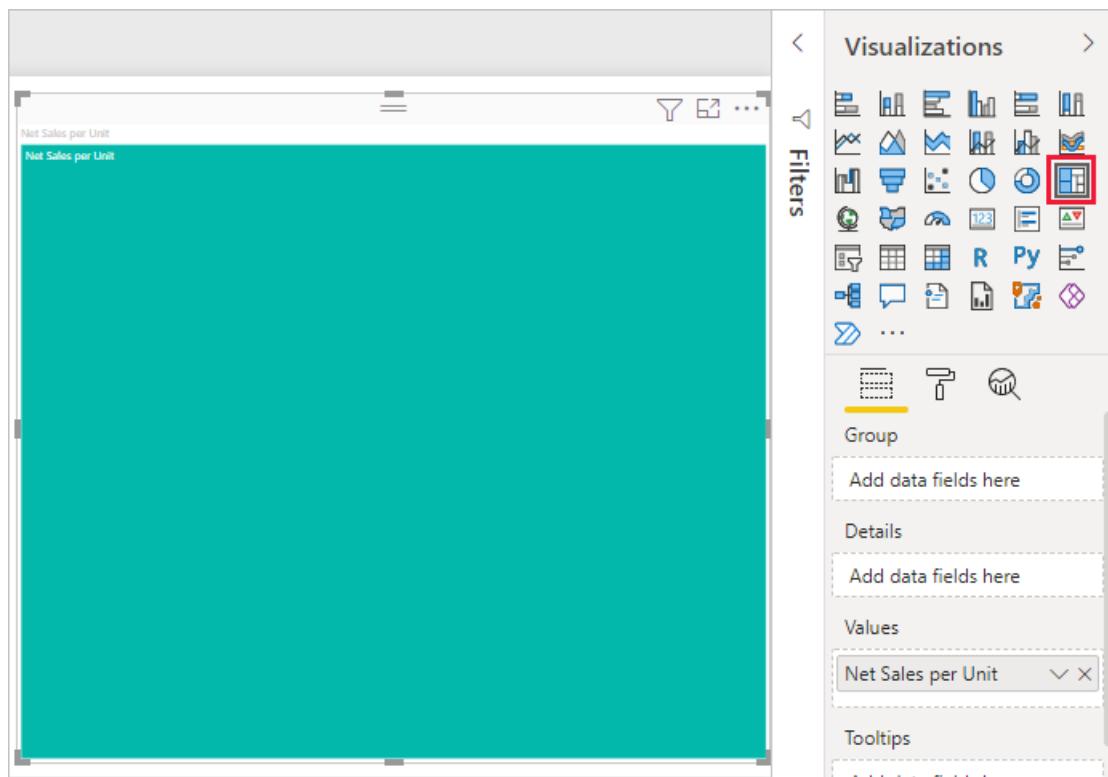
```
Net Sales per Unit = [Net Sales] / SUM(Sales[SalesQuantity])
```

5. Select the **Net Sales per Unit** measure from the **Sales** table, or drag it onto a blank area in the report canvas.

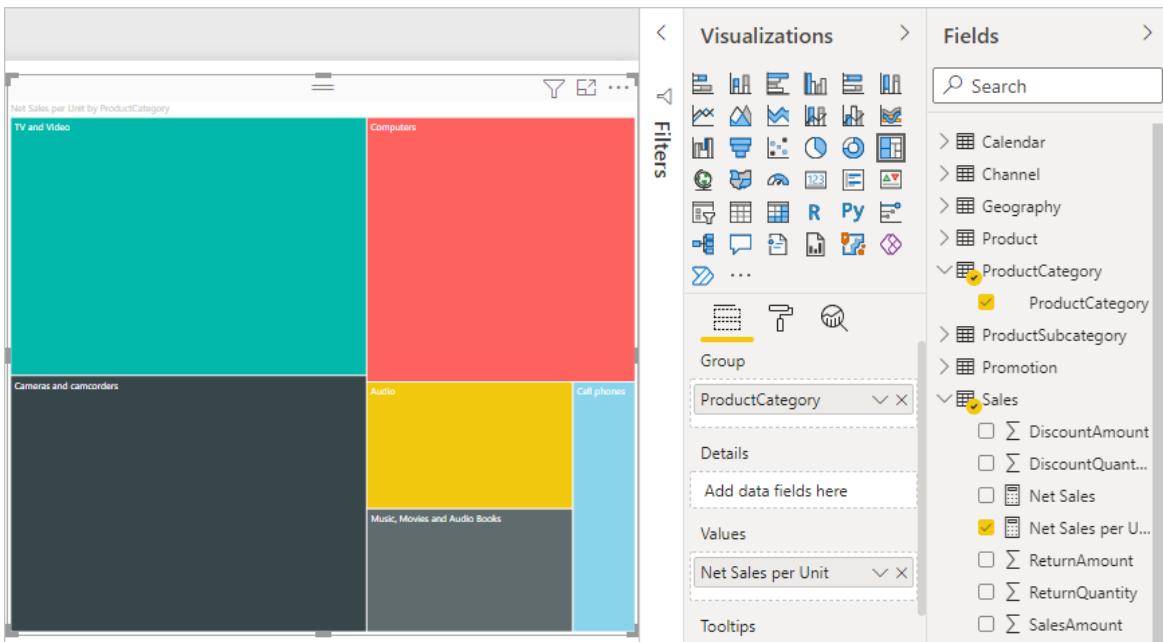
The chart shows the net sales amount per unit over all products sold. This chart isn't very informative; we'll address it in the next step.



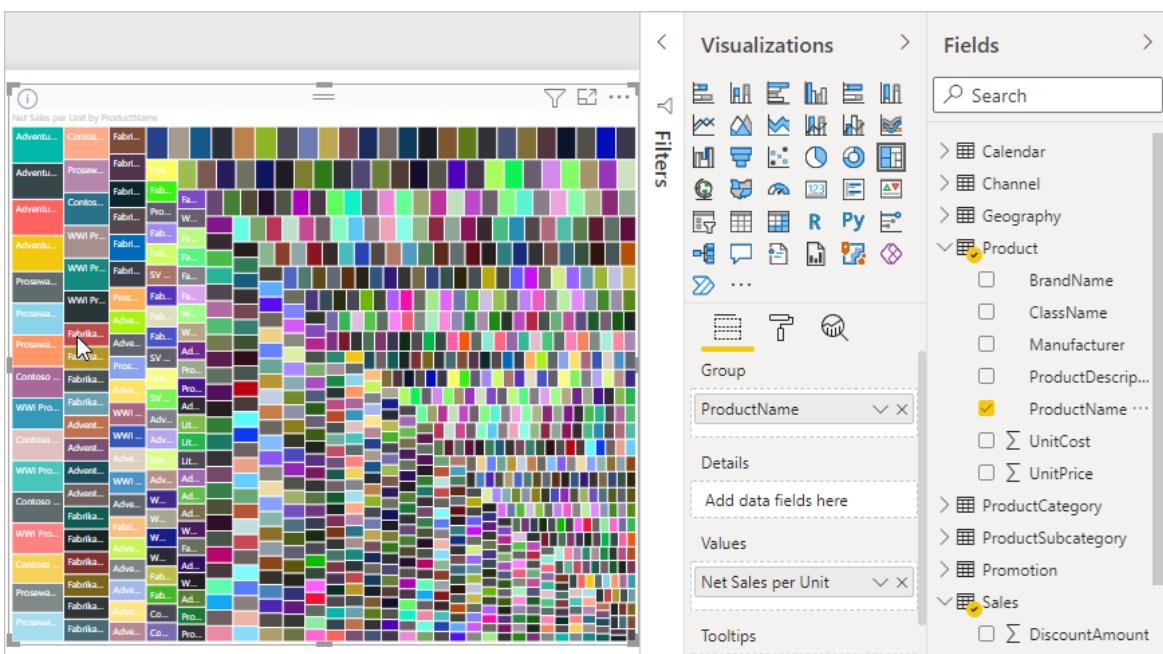
6. For a different look, change the chart visualization type to Treemap.



7. Select the **Product Category** field, or drag it onto the treemap or the **Group** field of the **Visualizations** pane. Now you have some good info!



8. Try removing the **ProductCategory** field, and dragging the **ProductName** field onto the chart instead.



Ok, now we're just playing, but you have to admit that's cool! Experiment with other ways to filter and format the visualization.

What you've learned

Measures give you the power to get the insights you want from your data. You've learned how to create measures by using the formula bar, name them whatever makes most sense, and find and select the right formula elements by using the DAX suggestion lists. You've also been introduced to context, where the results of calculations in measures change according to other fields or other expressions in your formula.

Next steps

- To learn more about Power BI Desktop quick measures, which provide many common measure calculations for you, see [Use quick measures to easily perform common and powerful calculations](#).
- If you want to take a deeper dive into DAX formulas and create some more advanced measures, see [DAX basics in Power BI Desktop](#). This article focuses on fundamental concepts in DAX, such as syntax,

functions, and a more thorough understanding of context.

- Be sure to add the [Data Analysis Expressions \(DAX\) Reference](#) to your favorites. This reference is where you'll find detailed info on DAX syntax, operators, and over 200 DAX functions.

Tutorial: Create calculated columns in Power BI Desktop

3/30/2022 • 7 minutes to read • [Edit Online](#)

Sometimes the data you're analyzing doesn't contain a particular field you need to get the results you're after. This is where *calculated columns* come in. Calculated columns use Data Analysis Expressions (DAX) formulas to define a column's values, anything from putting together text values from a couple of different columns to calculating a numeric value from other values. For example, let's say your data has **City** and **State** fields, but you want a single **Location** field that has both, like "Miami, FL". This is precisely what calculated columns are for.

Calculated columns are similar to [measures](#) in that both are based on DAX formulas, but they differ in how they are used. You often use measures in a visualization's **Values** area, to calculate results based on other fields. You use calculated columns as new **Fields** in the rows, axes, legends, and group areas of visualizations.

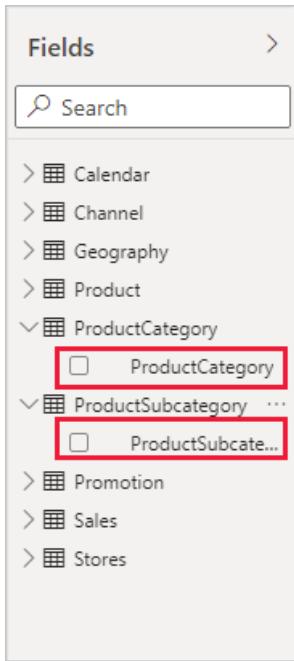
This tutorial will guide you through understanding and creating some calculated columns and using them in report visualizations in Power BI Desktop.

Prerequisites

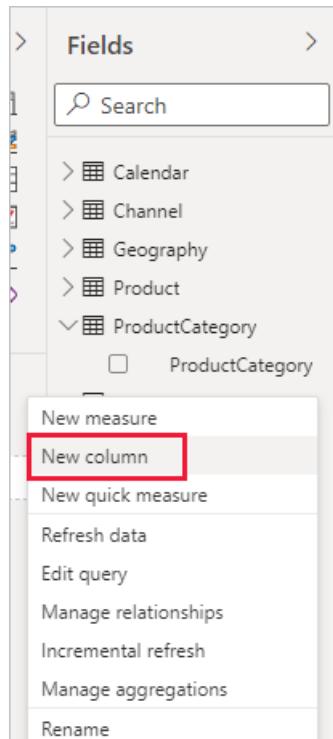
- This tutorial is intended for Power BI users already familiar with using Power BI Desktop to create more advanced models. You should already know how to use Get Data and the Power Query Editor to import data, work with multiple related tables, and add fields to the Report canvas. If you're new to Power BI Desktop, be sure to check out [Getting Started with Power BI Desktop](#).
- The tutorial uses the [Contoso Sales Sample for Power BI Desktop](#), the same sample used for the [Create your own measures in Power BI Desktop](#) tutorial. This sales data from the fictitious company Contoso, Inc. was imported from a database, so you won't be able to connect to the data source or view it in the Power Query Editor. Download and extract the file on your own computer, and then open it in Power BI Desktop.

Create a calculated column with values from related tables

In your Sales Report, you want to display product categories and subcategories as single values, like "Cell phones – Accessories", "Cell phones – Smartphones & PDAs", and so on. There's no field in the **Fields** list that gives you that data, but there is a **ProductCategory** field and a **ProductSubcategory** field, each in its own table. You can create a calculated column that combines values from these two columns. DAX formulas can leverage the full power of the model you already have, including relationships between different tables that already exist.



1. To create your new column in the **ProductSubcategory** table, right-click or select the ellipsis ... next to **ProductSubcategory** in the **Fields** pane, and select **New column** from the menu.



When you select **New column**, the **Formula bar** appears along the top of the Report canvas, ready for you to name your column and enter a DAX formula.

The screenshot shows the Power BI Desktop interface. On the left, there's a ribbon with icons for file, home, insert, transform, and modeling. Below the ribbon is a search bar with the text '1 Column ='. To the right of the search bar is a 'Build visuals with your data' section containing a small icon of a table with a checkmark. Further right is a 'Filters' section with a dropdown arrow. At the top right is a 'Fields' pane with a search bar and a list of fields. The 'ProductSubcategory' section is expanded, showing 'Column' which is highlighted with a red rectangle. Other fields listed include Calendar, Channel, Geography, Product, ProductCategory, ProductSubcategory, Promotion, Sales, and Stores. At the bottom, there are navigation buttons for 'Page 1' and a plus sign, and a status bar showing 'Page 1 of 1'.

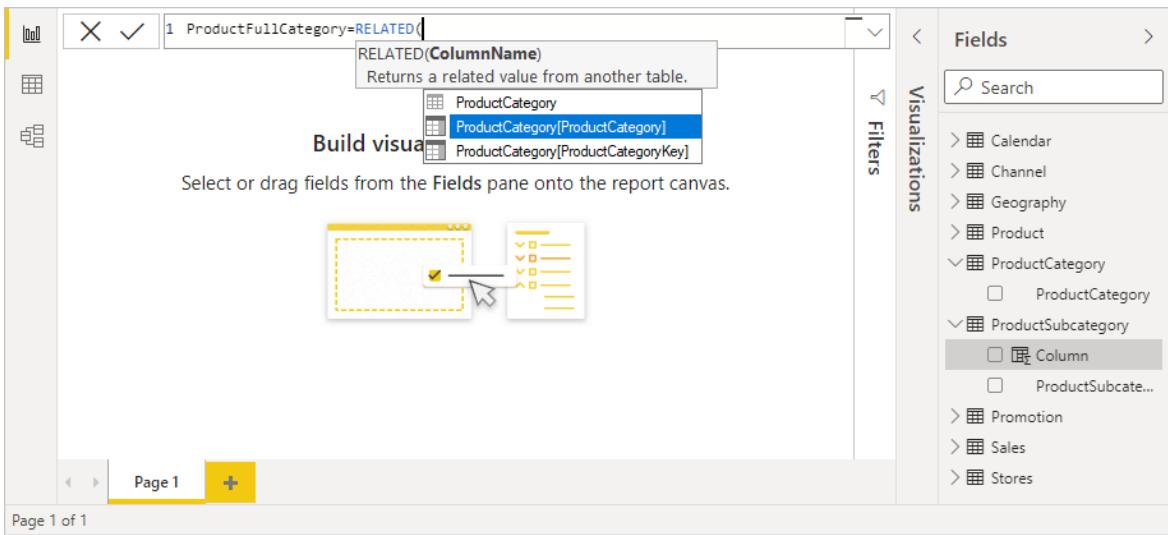
2. By default, a new calculated column is named **Column**. If you don't rename it, additional new columns will be named **Column 2**, **Column 3**, and so on. You want your column to be more identifiable, so while the **Column** name is already highlighted in the formula bar, rename it by typing **ProductFullCategory**, and then type an equals (=) sign.

3. You want the values in your new column to start with the name in the **ProductCategory** field. Because this column is in a different but related table, you can use the **RELATED** function to help you get it.

After the equals sign, type **r**. A dropdown suggestion list shows all of the DAX functions beginning with the letter R. Selecting each function shows a description of its effect. As you type, the suggestion list scales closer to the function you need. Select **RELATED**, and then press **Enter**.

This screenshot is similar to the previous one, showing the Power BI Desktop interface with the Fields pane expanded. In the formula bar, the text '1 ProductFullCategory=rel' is typed, and a dropdown menu is open over the word 'rel'. The 'RELATED' option is highlighted with a blue background and a tooltip 'Returns a related value from another table.' is visible. Below the formula bar is the 'Build visuals with your data' section. The 'Fields' pane on the right shows the same list of fields, with 'Column' still highlighted. The status bar at the bottom indicates 'Page 1 of 1'.

An opening parenthesis appears, along with another suggestion list of the related columns you can pass to the **RELATED** function, with descriptions and details of expected parameters.



4. You want the **ProductCategory** column from the **ProductCategory** table. Select **ProductCategory[ProductCategory]**, press **Enter**, and then type a closing parenthesis.

TIP

Syntax errors are most often caused by a missing or misplaced closing parenthesis, although sometimes Power BI Desktop will add it for you.

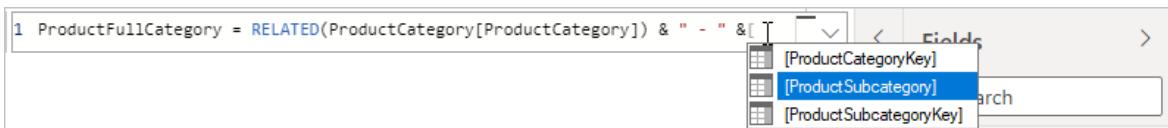
5. You want dashes and spaces to separate the **ProductCategories** and **ProductSubcategories** in the new values, so after the closing parenthesis of the first expression, type a space, ampersand (&), double-quote ("), space, dash (-), another space, another double-quote, and another ampersand. Your formula should now look like this:

```
ProductFullCategory = RELATED(ProductCategory[ProductCategory]) & " - " &
```

TIP

If you need more room, select the down chevron on the right side of the formula bar to expand the formula editor. In the editor, press **Alt + Enter** to move down a line, and **Tab** to move things over.

6. Enter an opening bracket ([), and then select the **[ProductSubcategory]** column to finish the formula.



You didn't need to use another RELATED function to call the **ProductSubcategory** table in the second expression, because you are creating the calculated column in this table. You can enter **[ProductSubcategory]** with the table name prefix (fully-qualified) or without (non-qualified).

7. Complete the formula by pressing **Enter** or selecting the checkmark in the formula bar. The formula validates, and the **ProductFullCategory** column name appears in the **ProductSubcategory** table in the **Fields** pane.

The screenshot shows the Power BI Desktop interface. At the top, there's a formula bar with the text: `1 ProductFullCategory = RELATED(ProductCategory[ProductCategory]) & " - " & [ProductSubcategory]`. Below the formula bar is a section titled "Build visuals with your data" with the sub-instruction "Select or drag fields from the Fields pane onto the report canvas." A cursor is shown dragging a field from the Fields pane into a dashed box on the report canvas. The Fields pane on the right lists various tables and their fields, with the "ProductFullCategory" field under the "ProductSubcategory" table highlighted and enclosed in a red box.

NOTE

In Power BI Desktop, calculated columns have a special icon in the **Fields** pane, showing that they contain formulas. In the Power BI service (your Power BI site), there's no way to change formulas, so calculated columns don't have icons.

Use your new column in a report

Now you can use your new **ProductFullCategory** column to look at **SalesAmount** by **ProductFullCategory**.

1. Select or drag the **ProductFullCategory** column from the **ProductSubcategory** table onto the Report canvas to create a table showing all of the **ProductFullCategory** names.

The screenshot shows the Power BI Desktop interface. The Fields pane on the right has the "ProductFullCategory" field under the "ProductSubcategory" table selected, indicated by a yellow checkmark icon. The report canvas shows a table visual with several rows of category names, such as "Audio - Audio Accessories", "Audio - Bluetooth Headphones", etc.

2. Select or drag the **SalesAmount** field from the **Sales** table into the table to show the **SalesAmount** for each **ProductFullCategory**.

The screenshot shows the Power BI Data View on the left and the Fields pane on the right. The Data View displays a table with columns 'ProductFullCategory' and 'SalesAmount'. The Fields pane shows a tree structure with 'ProductSubcategory' expanded, showing 'ProductFullCat...' and 'Sales' expanded, showing 'SalesAmount' selected. Both 'ProductFullCat...' and 'SalesAmount' are highlighted with red boxes.

Create a calculated column that uses an IF function

The Contoso Sales Sample contains sales data for both active and inactive stores. You want to ensure that active store sales are clearly separated from inactive store sales in your report by creating an **Active StoreName** field. In the new **Active StoreName** calculated column, each active store will appear with the store's full name, while the sales for inactive stores will be grouped together in one line item called **Inactive**.

Fortunately, the **Stores** table has a column named **Status**, with values of "On" for active stores and "Off" for inactive stores, which we can use to create values for our new **Active StoreName** column. Your DAX formula will use the logical **IF** function to test each store's **Status** and return a particular value depending on the result. If a store's **Status** is "On", the formula will return the store's name. If it's "Off", the formula will assign an **Active StoreName** of "Inactive".

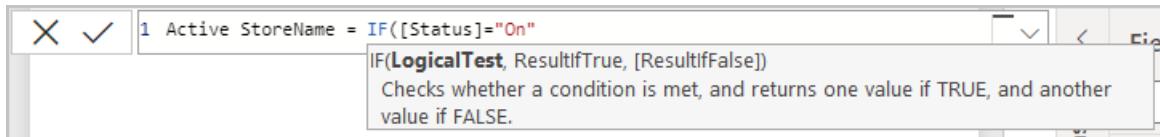
1. Create a new calculated column in the **Stores** table and name it **Active StoreName** in the formula bar.
2. After the = sign, begin typing **IF**. The suggestion list will show what you can add. Select **IF**.

The screenshot shows the Power BI formula bar with the text '1 Active StoreName = i'. A dropdown menu is open, showing 'IF' as the selected option. A tooltip for 'IF' says 'Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE.' To the right is the Fields pane, which shows the 'Stores' table expanded, with 'Status' selected.

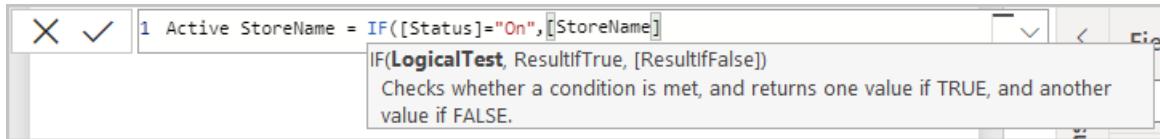
3. The first argument for **IF** is a logical test of whether a store's **Status** is "On". Type an opening bracket [, which lists columns from the **Stores** table, and select **[Status]**.

The screenshot shows the Power BI formula bar with the formula '1 Active StoreName = IF(['. A tooltip says 'Too few arguments were passed. The minimum argument count for the function is 2.' Below the formula bar is a dropdown menu showing columns from the 'Stores' table: [CloseReason], [EmployeeCount], [GeographyKey], [SellingAreaSize], [Status], [StoreKey], [StoreName], and [StoreType]. The 'Status' column is selected and highlighted.

4. Right after [Status], type = "On", and then type a comma (,) to end the argument. The tooltip suggests that you now need to add a value to return when the result is TRUE.



5. If the store's status is "On", you want to show the store's name. Type an opening bracket ([) and select the [StoreName] column, and then type another comma. The tooltip now indicates that you need to add a value to return when the result is FALSE.



6. You want the value to be "Inactive", so type "Inactive", and then complete the formula by pressing **Enter** or selecting the checkmark in the formula bar. The formula validates, and the new column's name appears in the **Stores** table in the **Fields** pane.

The screenshot shows the Power BI interface. The formula bar contains the validated formula: `1 Active StoreName = IF([Status] = "On", [StoreName], "Inactive")`. The Fields pane on the right lists various fields under the **Stores** table, with the **Active StoreName** field highlighted with a red border. The visualizations area shows a small preview of a chart with the text "Build visuals with your data".

7. You can use your new **Active StoreName** column in visualizations just like any other field. To show **SalesAmounts** by **Active StoreName**, select the **Active StoreName** field or drag it onto the Report canvas, and then select the **SalesAmount** field or drag it into the table. In this table, active stores appear individually by name, but inactive stores are grouped together at the end as **Inactive**.

The screenshot shows a Power BI report with a table view. The table has two columns: **Active StoreName** and **SalesAmount**. The data includes rows for various active stores and a single row for **Inactive** stores, which has a total sales amount of \$189,962,742.7355. To the right of the table is a **Visualizations** pane with a list of measures. Two measures are selected and highlighted with red boxes: **Σ SalesAmount** and **Active StoreName**.

Active StoreName	SalesAmount
Contoso Wapato Store	\$16,427,512.9295
Contoso Warsaw Store	\$15,142,181.7609
Contoso Waterbury Store	\$15,104,327.8925
Contoso Waukesha No.1 Store	\$16,032,441.5125
Contoso Waukesha No.2 Store	\$16,448,330.8045
Contoso West Yorkshire Store	\$15,165,663.891
Contoso Westminster Store	\$15,266,782.0765
Contoso Wheat Ridge Store	\$16,117,648.774
Contoso Winchester Store	\$15,563,992.0475
Contoso Worcester No.1 Store	\$15,388,242.957
Contoso Yakima Store	\$16,266,888.313
Contoso Yerevan Store	\$26,084,935.2425
Contoso Yokohama Store	\$25,311,723.6245
Contoso York Store	\$14,926,059.9838
Inactive	\$189,962,742.7355
Total	\$8,341,224,364.8324

What you've learned

Calculated columns can enrich your data and provide easier insights. You've learned how to create calculated

columns in the **Fields** pane and formula bar, use suggestion lists and tooltips to help construct your formulas, call DAX functions like RELATED and IF with the appropriate arguments, and use your calculated columns in report visualizations.

Next steps

If you want to take a deeper dive into DAX formulas and create calculated columns with more advanced formulas, see [DAX Basics in Power BI Desktop](#). This article focuses on fundamental concepts in DAX, such as syntax, functions, and a more thorough understanding of context.

Be sure to add the [Data Analysis Expressions \(DAX\) Reference](#) to your favorites. This is where you'll find detailed info on DAX syntax, operators, and over 200 DAX functions.

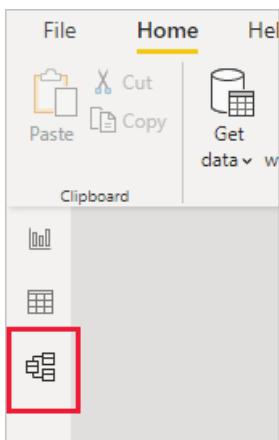
Work with Modeling view in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

With **Modeling view** in Power BI Desktop, you can view and work with complex datasets that contain many tables.

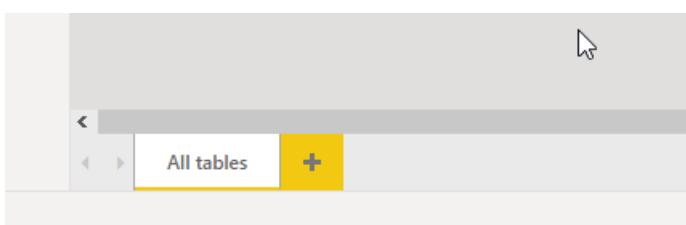
Using Modeling view

To access Modeling view, select the Model icon found on the left side of **Power BI Desktop**, as shown in the following image.

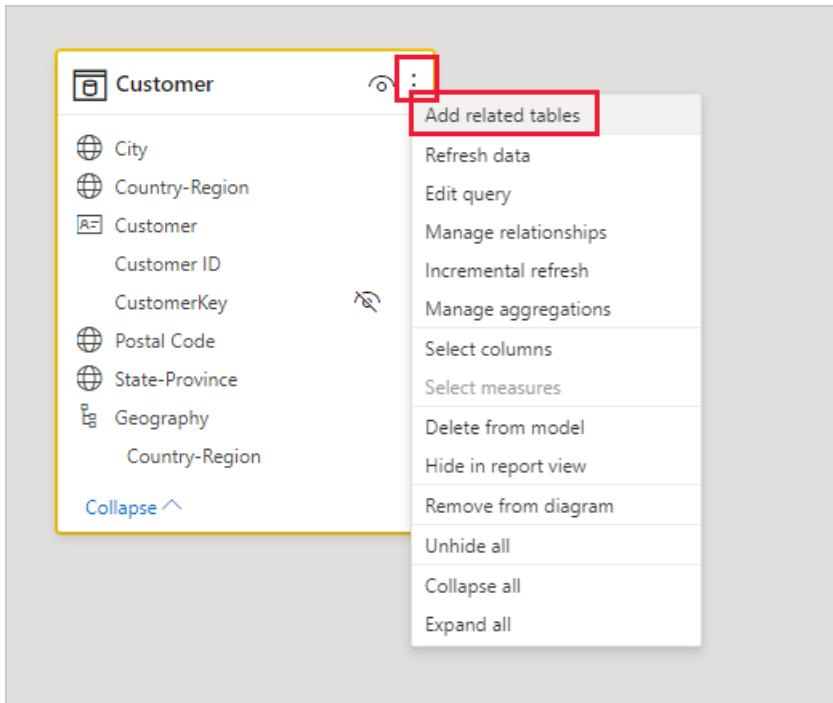


Creating separate diagrams

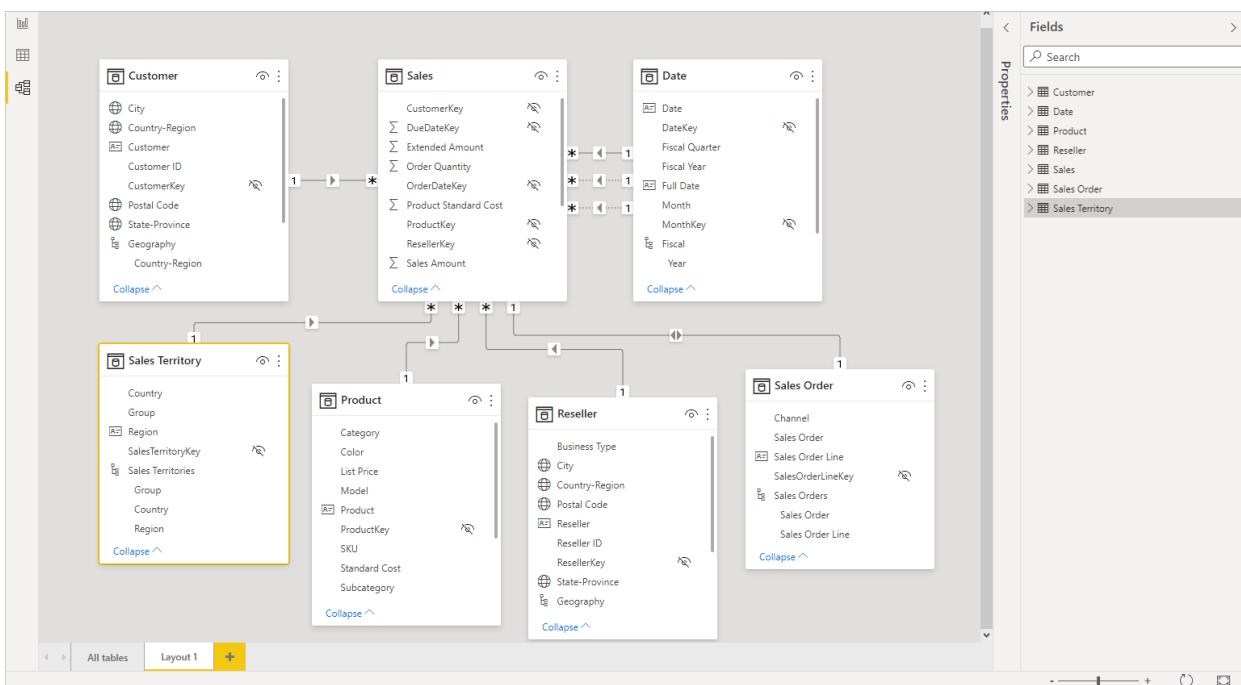
With Modeling view, you can create diagrams of your model that contain only a subset of the tables in your model. This can help provide a clearer view into the tables you want to work with, and make working with complex datasets easier. To create a new diagram with only a subset of the tables, click the + sign next to the **All tables** tab along the bottom of the Power BI Desktop window.



You can then drag a table from the **Fields** list onto the diagram surface. Right click the table, and then select **Add related tables** from the menu that appears.



When you do, tables that are related to the original table are displayed in the new diagram. The following image shows how related tables are displayed after selecting the **Add related tables** menu option.



NOTE

You can also find the 'Add related tables' option in the context menu on the background of the model view. When clicked, any table that has any relationship to any table already included in the layout will be added to the layout.

Setting common properties

You can select multiple objects at once in Modeling view by holding down the **Ctrl** key and clicking multiple tables. When you select multiple tables they become highlighted in Modeling view. When multiple tables are highlighted, changes applied in the **Properties** pane apply to all selected tables.

For example, you could change the **storage mode** for multiple tables in your diagram view by holding down the **Ctrl** key, selecting tables, then changing the storage mode setting in the **Properties** pane.



Next steps

The following articles describe more about data models, and also describe DirectQuery in detail.

- [Aggregations in Power BI Desktop \(Preview\)](#)
- [Composite models in Power BI Desktop](#)
- [Storage Mode in Power BI Desktop \(Preview\)](#)
- [Many-to-many relationships in Power BI Desktop](#)

DirectQuery articles:

- [Using DirectQuery in Power BI](#)
- [Data sources supported by DirectQuery in Power BI](#)

Model relationships in Power BI Desktop

3/30/2022 • 15 minutes to read • [Edit Online](#)

This article targets Import data modelers working with Power BI Desktop. It's an important model design topic that's essential to delivering intuitive, accurate, and optimal models.

For a deeper discussion on optimal model design, including table roles and relationships, see the [Understand star schema and the importance for Power BI](#) article.

Relationship purpose

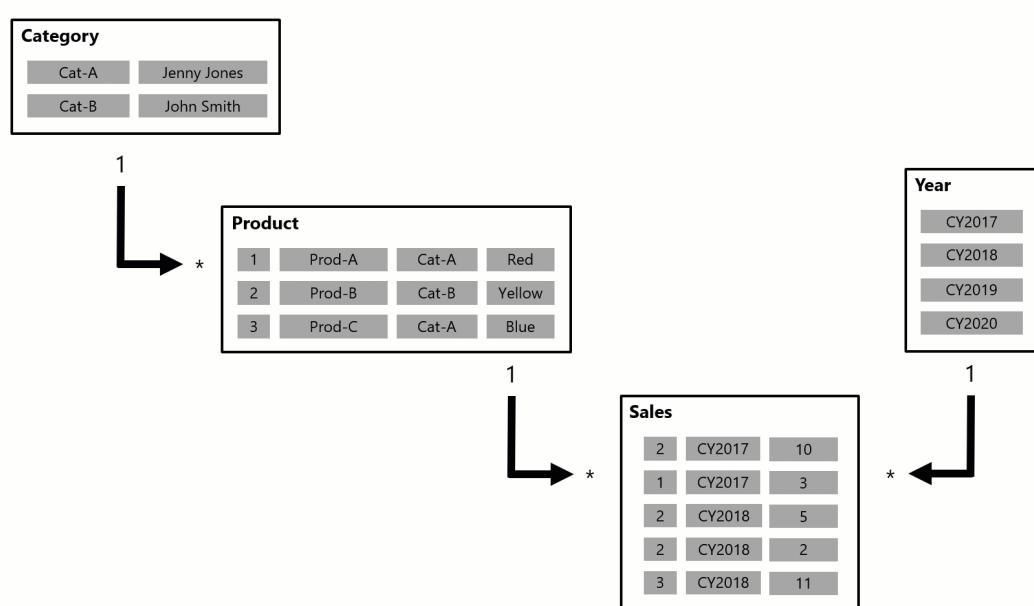
Put simply, Power BI relationships propagate filters applied on the columns of model tables to other model tables. Filters will propagate so long as there's a relationship path to follow, which can involve propagation to multiple tables.

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. For more information, see the [Relevant DAX functions](#) topic later in this article.

IMPORTANT

It's important to understand that model relationships do not enforce data integrity. For more information, see the [Relationship evaluation](#) topic later in this article. This topic explains how model relationships behave when there are data integrity issues with your data.

Let's see how relationships propagate filters with an animated example.



In this example, the model consists of four tables: **Category**, **Product**, **Year**, and **Sales**. The **Category** table relates to the **Product** table, and the **Product** table relates to the **Sales** table. The **Year** table also relates to the **Sales** table. All relationships are one-to-many (the details of which are described later in this article).

A query—possibly generated by a Power BI card visual—requests the total sales quantity for sales orders made

for a single category, **Cat-A**, and for a single year, **CY2018**. It's why you can see filters applied on the **Category** and **Year** tables. The filter on the **Category** table propagates to the **Product** table to isolate two products that are assigned to the category **Cat-A**. Then the **Product** table filters propagate to the **Sales** table to isolate just two sales rows for these products. These two sales rows represent the sales of products assigned to category **Cat-A**. Their combined quantity is 14 units. At the same time, the **Year** table filter propagates to further filter the **Sales** table, resulting in just the one sales row that is for products assigned to category **Cat-A** and that was ordered in year **CY2018**. The quantity value returned by the query is 11 units. Note that when multiple filters are applied to a table (like the **Sales** table in this example), it's always an AND operation, requiring that all conditions must be true.

Disconnected tables

It's unusual that a model table isn't related to another model table. Such a table in a valid model design can be described as a *disconnected table*. A disconnected table isn't intended to propagate filters to other model tables. Instead, it's used to accept "user input" (perhaps with a slicer visual), allowing model calculations to use the input value in a meaningful way. For example, consider a disconnected table that is loaded with a range of currency exchange rate values. As long as a filter is applied to filter by a single rate value, the value can be used by a measure expression to convert sales values.

The Power BI Desktop what-if parameter is a feature that creates a disconnected table. For more information, see the [Create and use a What if parameter to visualize variables in Power BI Desktop](#) article.

Relationship properties

A model relationship relates one column in a table to one column in a different table. (There's one specialized case where this requirement isn't true, and it applies only to multi-column relationships in DirectQuery models. For more information, see the [COMBINEVALUES DAX function](#) article.)

NOTE

It's not possible to relate a column to a different column *in the same table*. This is sometimes confused with the ability to define a relational database foreign key constraint that is table self-referencing. This relational database concept can be used to store parent-child relationships (for example, each employee record is related to a "reports to" employee). Generating a model hierarchy based on this type of relationship can't be solved by creating model relationships. To achieve this, see the [Parent and Child functions](#) article.

Cardinality

Each model relationship must be defined with 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.

NOTE

If a data refresh operation attempts to load duplicate values into a "one" side column, the entire data refresh will fail.

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 (*:*)

When you create a relationship in Power BI Desktop, the designer will automatically detect and set the cardinality type. The designer queries the model to know which columns contain unique values. For Import

models, it uses internal storage statistics; for DirectQuery models it sends profiling queries to the data source. Sometimes, however, it can get it wrong. It happens because tables are yet to be loaded with data, or because columns that you expect to contain duplicate values currently contain unique values. In either case, you can update the cardinality type as long as any "one" side columns contain unique values (or the table is yet to be loaded with rows of data).

The **One-to-many** and **Many-to-one** cardinality options are essentially the same, and they're also the most common cardinality types.

When configuring a One-to-many or Many-to-one relationship, you'll choose the one that matches the order in which you related the columns. Consider how you would configure the relationship from the **Product** table to the **Sales** table by using the **ProductID** column found in each table. The cardinality type would be *One-to-many*, as the **ProductID** column in the **Product** table contains unique values. If you related the tables in the reverse direction, **Sales** to **Product**, then the cardinality would be *Many-to-one*.

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. For more information on using this cardinality type, see [One-to-one relationship guidance](#).

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. For guidance on using this cardinality type, see [Many-to-many relationship guidance](#).

NOTE

The Many-to-many cardinality type isn't currently supported for models developed for Power BI Report Server.

TIP

In Power BI Desktop model view, you can interpret a relationship's cardinality type by looking at the indicators (1 or *) on either side of the relationship line. To determine which columns are related, you'll need to select—or hover the cursor over—the relationship line to highlight the columns.

Cross filter direction

Each model relationship must be defined with a cross filter direction. Your selection 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

Single cross filter direction means "single direction", and *Both* means "both directions". A relationship that filters in both directions is commonly described as *bi-directional*.

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 the 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.

When the cross filter direction is set to **Both**, an additional property is available. It can apply bi-directional filtering when row-level security (RLS) rules are enforced. For more information about RLS, see the [Row-level security \(RLS\) with Power BI Desktop](#) article.

Modifying the relationship cross filter direction—including the disabling of filter propagation—can also be done by a model calculation. It's achieved by using the [CROSSFILTER](#) DAX function.

Bi-directional relationships can impact negatively on performance. Further, attempting to configure a bi-directional relationship could result in ambiguous filter propagation paths. In this case, Power BI Desktop may fail to commit the relationship change and will alert you with an error message. Sometimes, however, Power BI Desktop may allow you to define ambiguous relationship paths between tables. Precedence rules that affect ambiguity detection and path resolution are described later in this article in the [Precedence rules](#) topic.

We recommend using bi-directional filtering only as needed. For more information, see [Bi-directional relationship guidance](#).

TIP

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; a double arrowhead represents a bi-directional relationship.

Make this relationship active

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

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

TIP

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.

Assume referential integrity

The *Assume referential integrity* property is available only for One-to-many and One-to-one relationships between two DirectQuery storage mode tables that are based on the same data source. When enabled, native queries sent to the data source will join the two tables together by using an INNER JOIN rather than an OUTER JOIN. Generally, enabling this property improves query performance, though it does depend on the specifics of the data source.

Always enable this property when a database foreign key constraint exists between the two tables. When a foreign key constraint doesn't exist, you can still enable the property as long as you're certain data integrity exists.

IMPORTANT

If data integrity should become compromised, the inner join will eliminate unmatched rows between the tables. For example, consider a model **Sales** table with a **ProductID** column value that did not exist in the related **Product** table. Filter propagation from the **Product** table to the **Sales** table will eliminate sales rows for unknown products. This would result in an understatement of the sales results.

For more information, see the [Assume referential integrity settings in Power BI Desktop](#) article.

Relevant DAX functions

There are several DAX functions that are relevant to model relationships. Each function is described briefly in the following bulleted list:

- **RELATED**: Retrieves the value from "one" side.
- **RELATEDTABLE**: Retrieve a table of rows from "many" side.
- **USERELATIONSHIP**: Forces the use of a specific inactive model relationship.
- **CROSSFILTER**: Modifies the relationship cross filter direction (to one or both), or it disables filter propagation (none).
- **COMBINEVALUES**: Joins two or more text strings into one text string. The purpose of this function is to support multi-column relationships in DirectQuery models.
- **TREATAS**: Applies the result of a table expression as filters to columns from an unrelated table.
- **Parent and Child functions**: A family of related functions that can be used to generate calculated columns to naturalize a parent-child hierarchy. These columns can then be used to create a fixed-level hierarchy.

Relationship evaluation

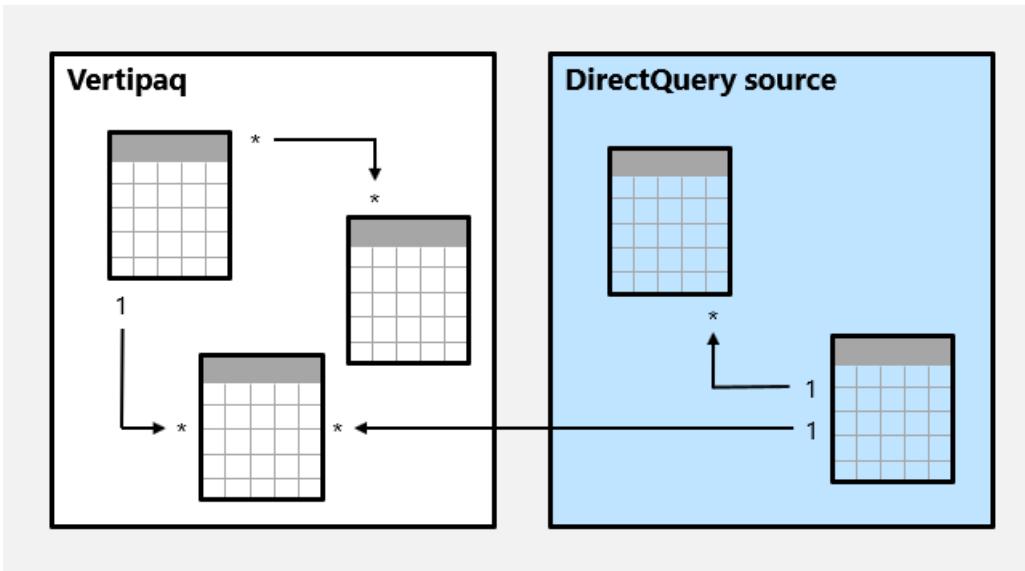
Model relationships, from an evaluation perspective, are classified as either *regular* or *limited*. It's not a configurable relationship property. It is in fact inferred from the cardinality type and the data source of the two related tables. It's important to understand the evaluation type because there may be performance implications or consequences should data integrity be compromised. These implications and integrity consequences are described in this topic.

First, some modeling theory is required to fully understand relationship evaluations.

An Import or DirectQuery model sources all of its data from either the Vertipaq cache or the source database. In both instances, Power BI is able to determine that a "one" side of a relationship exists.

A Composite model, however, can comprise tables using different storage modes (Import, DirectQuery or Dual), or multiple DirectQuery sources. Each source, including the Vertipaq cache of Import data, is considered to be a *source group*. Model relationships can then be classified as *intra source group* or *inter / cross source group*. An intra source group relationship is one that relates two tables within a source group, while a inter / cross source group relationship relates tables from different source group. Note that relationships in Import or DirectQuery models are always intra source group.

Let's see an example of a Composite model.

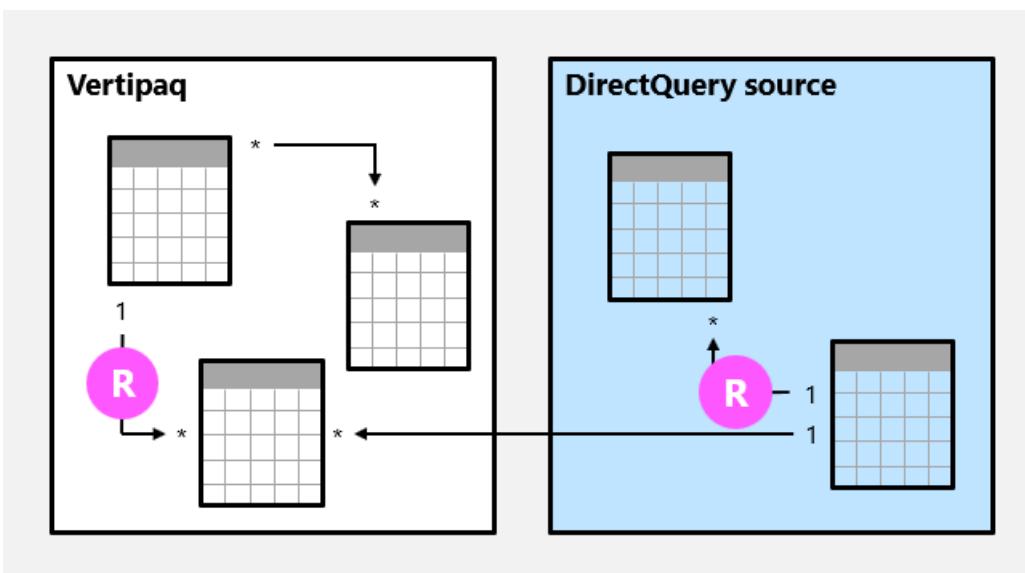


In this example, the Composite model consists of two source groups: a Vertipaq source group and a DirectQuery source group. The Vertipaq source group contains three tables, and the DirectQuery source group contains two tables. One cross source group relationship exists to relate a table in the Vertipaq source group to a table in the DirectQuery source group.

Regular relationships

A model relationship is *regular* when the query engine can determine the "one" side of relationship. It has confirmation that the "one" side column contains unique values. All One-to-many intra source group relationships are regular relationships.

In the following example, there are two regular relationships, both marked as R. Relationships include the One-to-many relationship contained within the Vertipaq source group, and the One-to-many relationship contained within the DirectQuery source.



For Import models, where all data is stored in the Vertipaq cache, a data structure is created for each regular relationship at data refresh time. The data structures consist of indexed mappings of all column-to-column values, and their purpose is to accelerate joining tables at query time.

At query time, regular relationships permit *table expansion* to take place. Table expansion results in the creation of a virtual table by including the native columns of the base table and then expanding into related tables. For Import tables, it's done in the query engine; for DirectQuery tables it is done in the native query sent to the source database (as long as the **Assume referential integrity** property isn't enabled). The query engine then acts upon the expanded table, applying filters and grouping by the values in the expanded table columns.

NOTE

Inactive relationships are expanded also, even when the relationship isn't used by a calculation. Bi-directional relationships have no impact on table expansion.

For One-to-many relationships, table expansion takes place from the "many" to the "one" sides by using LEFT OUTER JOIN semantics. When a matching value from the "many" to the "one" side doesn't exist, a blank virtual row is added to the "one" side table.

Table expansion also occurs for One-to-one intra source group relationships, but by using FULL OUTER JOIN semantics. It ensures that blank virtual rows are added on either side, when necessary.

The blank virtual rows are effectively *Unknown Members*. Unknown members represent referential integrity violations where the "many" side value has no corresponding "one" side value. Ideally these blanks should not exist, and they can be eliminated by cleansing or repairing the source data.

Let's see how table expansion works with an animated example.

In this example, the model consists of three tables: **Category**, **Product**, and **Sales**. The **Category** table relates to the **Product** table with a One-to-many relationship, and the **Product** table relates to the **Sales** table with a One-to-many relationship. The **Category** table contains two rows, the **Product** table contains three rows, and the **Sales** table contains five rows. There are matching values on both sides of all relationships meaning that there are no referential integrity violations. A query-time expanded table is revealed. The table consists of the columns from all three tables. It's effectively a denormalized perspective of the data contained in the three tables. A new row is added to the **Sales** table, and it has a production identifier value (9) that has no matching value in the **Product** table. It's a referential integrity violation. In the expanded table, the new row has (Blank) values for the **Category** and **Product** table columns.

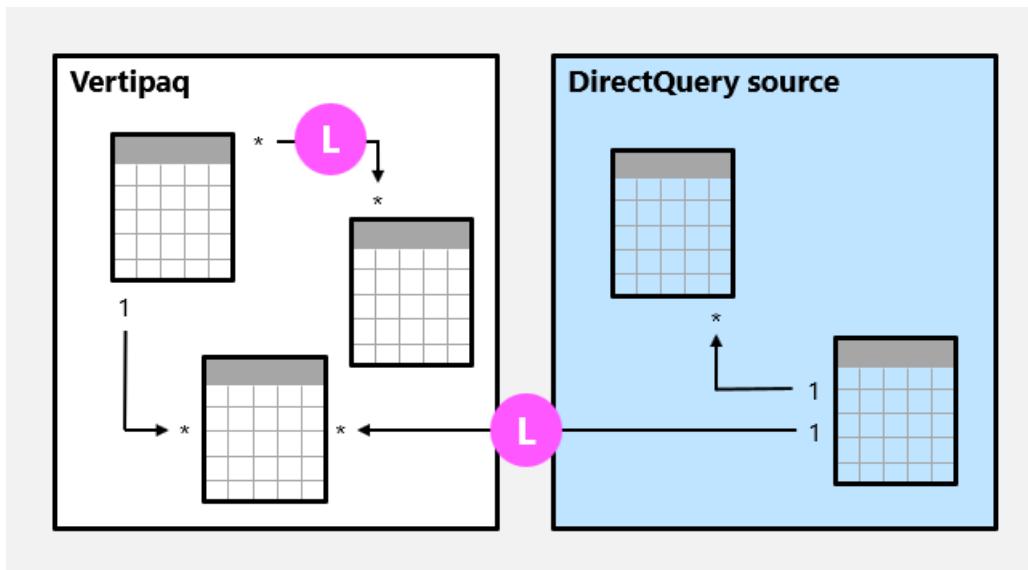
Limited relationships

A model relationship is *limited* when there's no guaranteed "one" side. It can be the case for two reasons:

- The relationship uses a Many-to-many cardinality type (even if one or both columns contain unique values)
- The relationship is cross source group (which can only ever be the case for Composite models)

In the following example, there are two limited relationships, both marked as L. The two relationships include the Many-to-many relationship contained within the Vertipaq source group, and the One-to-many cross source

group relationship.



For Import models, data structures are never created for limited relationships. This means table joins must be resolved at query time.

Table expansion never occurs for limited relationships. Table joins are achieved by using INNER JOIN semantics, and for this reason, blank virtual rows are not added to compensate for referential integrity violations.

There are additional restrictions related to limited relationships:

- The RELATED DAX function can't be used to retrieve the "one" side column values
- Enforcing RLS has topology restrictions

NOTE

In Power BI Desktop model view, it's not always possible to determine whether a model relationship is regular or limited. A Many-to-many relationship will always be limited, as is a One-to-many relationship when it's a cross source group relationship. To determine whether it's a cross source group relationship, you'll need to inspect the table storage modes and data sources to arrive at the correct determination.

Precedence rules

Bi-directional relationships can introduce multiple—and therefore ambiguous—filter propagation paths between model tables. The following list presents precedence rules that Power BI uses for ambiguity detection and path resolution:

1. Many-to-one and One-to-one relationships, including limited relationships
2. Many-to-many relationships
3. Bi-directional relationships, in the reverse direction (that is, from the "Many" side)

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 involves at least one bi-directional relationship
3. Many-to-many cardinality relationships
4. Cross source group relationships

Next steps

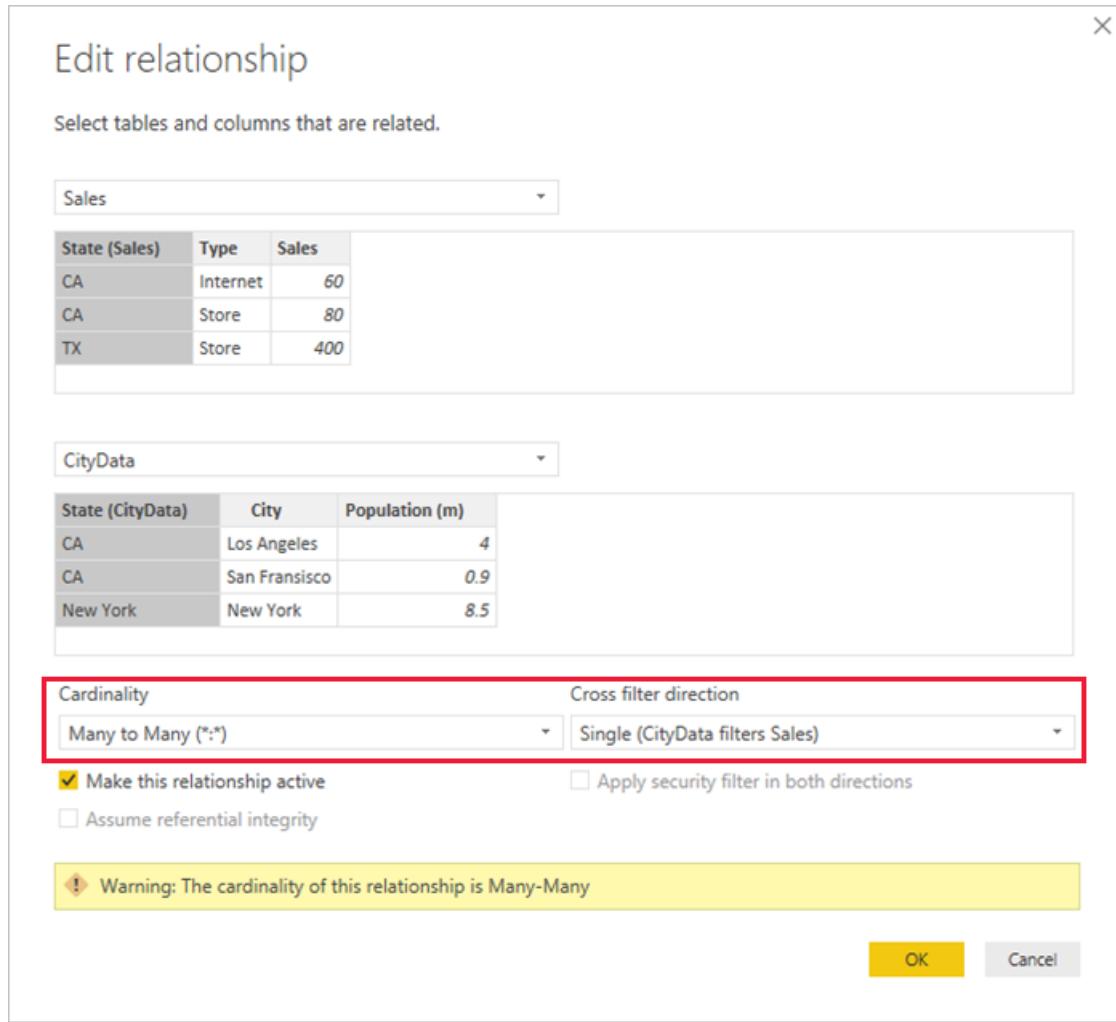
For more information about this article, check out the following resources:

- [Understand star schema and the importance for Power BI](#)
- [One-to-one relationship guidance](#)
- [Many-to-many relationship guidance](#)
- [Active vs inactive relationship guidance](#)
- [Bi-directional relationship guidance](#)
- [Relationship troubleshooting guidance](#)
- Video: [The Do's and Don'ts of Power BI Relationships](#)
- Questions? [Try asking the Power BI Community](#)
- Suggestions? [Contribute ideas to improve Power BI](#)

Apply many-to-many relationships in Power BI Desktop

3/30/2022 • 8 minutes to read • [Edit Online](#)

With *relationships with a many-to-many cardinality* in Power BI Desktop, you can join tables that use a cardinality of *many-to-many*. You can more easily and intuitively create data models that contain two or more data sources. *Relationships with a many-to-many cardinality* are part of the larger *composite models* capabilities in Power BI Desktop.



A *relationship with a many-to-many cardinality* in Power BI Desktop is composed of one of three related features:

- **Composite models:** A *composite model* allows a report to have two or more data connections, including DirectQuery connections or Import, in any combo. For more information, see [Use composite models in Power BI Desktop](#).
- **Relationships with a many-to-many cardinality:** With composite models, you can establish *relationships with a many-to-many cardinality* between tables. This approach removes requirements for unique values in tables. It also removes previous workarounds, such as introducing new tables only to establish relationships. The feature is described further in this article.
- **Storage mode:** You can now specify which visuals require a query to back-end data sources. Visuals that don't require a query are imported even if they're based on DirectQuery. This feature helps improve

performance and reduce back-end load. Previously, even simple visuals, such as slicers, began queries that were sent to back-end sources. For more information, see [Storage mode in Power BI Desktop](#).

What a relationship with a many-to-many cardinality solves

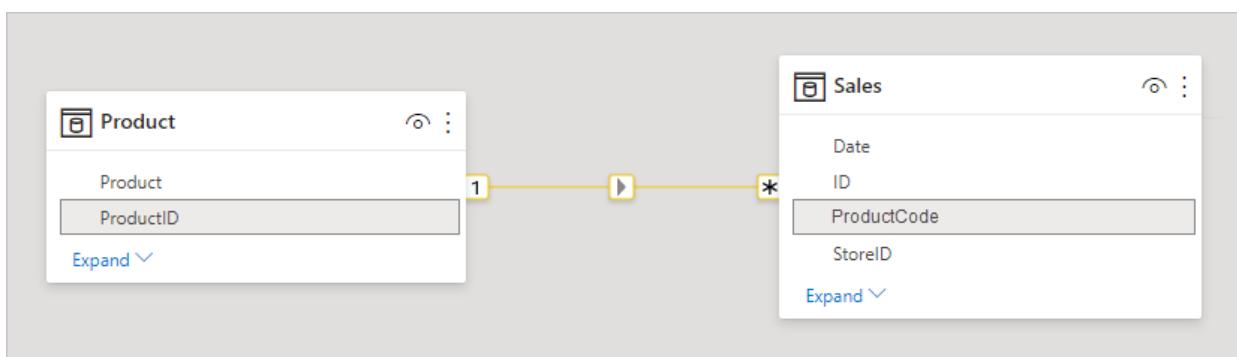
Before *relationships with a many-to-many cardinality* became available, the relationship between two tables was defined in Power BI. At least one of the table columns involved in the relationship had to contain unique values. Often, though, no columns contained unique values.

For example, two tables might have had a column labeled Country. The values of Country weren't unique in either table, though. To join such tables, you had to create a workaround. One workaround might be to introduce extra tables with the needed unique values. With *relationships with a many-to-many cardinality*, you can join such tables directly, if you use a relationship with a cardinality of *many-to-many*.

Use relationships with a many-to-many cardinality

When you define a relationship between two tables in Power BI, you must define the cardinality of the relationship. For example, the relationship between ProductSales and Product—using columns ProductSales[ProductCode] and Product[ProductCode]—would be defined as *Many-1*. We define the relationship in this way, because each product has many sales, and the column in the Product table (ProductCode) is unique. When you define a relationship cardinality as *Many-1*, *1-Many*, or *1-1*, Power BI validates it, so the cardinality that you select matches the actual data.

For example, take a look at the simple model in this image:



Now, imagine that the **Product** table displays just two rows, as shown:

ProductCode	ProductName	Price
A	Name for Product A	20
B	Name for Product B	23

Also imagine that the Sales table has just four rows, including a row for a product C. Because of a referential integrity error, the product C row doesn't exist in the **Product** table.

ProductCode	Date	Qty
A	1/1/2018	10
A	1/2/2018	20
B	1/1/2018	50
C	1/1/2018	1000

The **ProductName** and **Price** (from the **Product** table), along with the total **Qty** for each product (from the

ProductSales table), would be displayed as shown:

ProductName	Price	Qty
Name for Product B	23	50
Name for Product A	20	30
		1000
Total	1080	

As you can see in the preceding image, a blank **ProductName** row is associated with sales for product C. This blank row accounts for the following:

- Any rows in the **ProductSales** table for which no corresponding row exists in the **Product** table. There's a referential integrity issue, as we see for product C in this example.
- Any rows in the **ProductSales** table for which the foreign key column is null.

For these reasons, the blank row in both cases accounts for sales where the **ProductName** and **Price** are unknown.

Sometimes the tables are joined by two columns, yet neither column is unique. For example, consider these two tables:

- The **Sales** table displays sales data by **State**, and each row contains the sales amount for the type of sale in that state. The states include CA, WA, and TX.

State (Sales)	Type	Sales
CA	Internet	60
CA	Store	80
TX	Store	400
WA	Internet	150
WA	Store	100

- The **CityData** table displays data on cities, including the population and state (such as CA, WA, and New York).

State (CityData)	City	Population (m)
CA	Los Angeles	4.00
CA	San Francisco	0.90
New York	New York	8.50
WA	Seattle	0.70
WA	Spokane	0.20

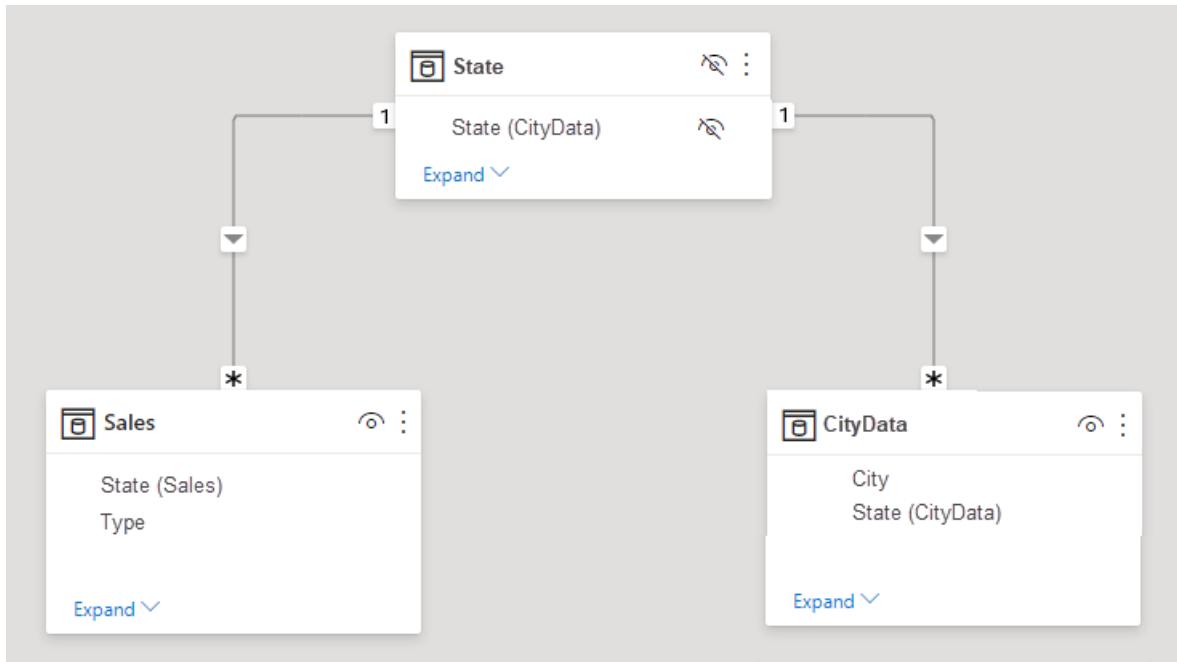
A column for **State** is now in both tables. It's reasonable to want to report on both total sales by state and total population of each state. However, a problem exists: the **State** column isn't unique in either table.

The previous workaround

Before the July 2018 release of Power BI Desktop, you couldn't create a direct relationship between these tables. A common workaround was to:

- Create a third table that contains only the unique State IDs. The table could be any or all of:
 - A calculated table (defined by using Data Analysis Expressions [DAX]).
 - A table based on a query that's defined in Power Query Editor, which could display the unique IDs drawn from one of the tables.
 - The combined full set.
- Then relate the two original tables to that new table by using common *Many-1* relationships.

You could leave the workaround table visible. Or you may hide the workaround table, so it doesn't appear in the **Fields** list. If you hide the table, the *Many-1* relationships would commonly be set to filter in both directions, and you could use the **State** field from either table. The later cross filtering would propagate to the other table. That approach is shown in the following image:



A visual that displays **State** (from the **CityData** table), along with total **Population** and total **Sales**, would then appear as follows:

State (CityData)	Population (m)	Sales
CA	4.90	140
New York	8.50	
WA	0.90	250
Total	14.30	790

NOTE

Because the state from the **CityData** table is used in this workaround, only the states in that table are listed, so TX is excluded. Also, unlike *Many-1* relationships, while the total row includes all **Sales** (including those of TX), the details don't include a blank row covering such mismatched rows. Similarly, no blank row would cover **Sales** for which there's a null value for the **State**.

Suppose you also add **City** to that visual. Although the population per **City** is known, the **Sales** shown for **City** simply repeats the **Sales** for the corresponding **State**. This scenario normally occurs when the column grouping is unrelated to some aggregate measure, as shown here:

State (CityData)	Population (m)	Sales
CA	4.90	140
Los Angeles	4.00	140
San Fransisco	0.90	140
New York	8.50	
New York	8.50	
WA	0.90	250
Seattle	0.70	250
Spokane	0.20	250
Total	14.30	790

Let's say you define the new Sales table as the combination of all States here, and we make it visible in the **Fields** list. The same visual would display **State** (on the new table), the total **Population**, and total **Sales**:

State	Population (m)	Sales
CA	4.90	140
New York	8.50	
TX		400
WA	0.90	250
Total	14.30	790

As you can see, TX—with **Sales** data but unknown *Population* data—and New York—with known *Population* data but no **Sales** data—would be included. This workaround isn't optimal, and it has many issues. For relationships with a many-to-many cardinality, the resulting issues are addressed, as described in the next section.

For more information about implementing this workaround, see [many-to-many relationship guidance](#).

Use a relationship with a many-to-many cardinality instead of the workaround

You can directly relate tables, such as the ones we described earlier, without having to resort to similar workarounds. It's now possible to set the relationship cardinality to *many-to-many*. This setting indicates that neither table contains unique values. For such relationships, you may still control which table filters the other table. Or you can apply bidirectional filtering, where each table filters the other.

In Power BI Desktop, the cardinality defaults to *many-to-many* when it determines neither table contains unique values for the relationship columns. In such cases, a warning message confirms you want to set a relationship, and the change isn't the unintended effect of a data issue.

For example, when you create a relationship directly between CityData and Sales—where filters should flow from CityData to Sales—Power BI Desktop displays the **Edit relationship** dialog box:

Edit relationship

Select tables and columns that are related.

Sales		
State (Sales)	Type	Sales
CA	Internet	60
CA	Store	80
TX	Store	400

CityData		
State (CityData)	City	Population (m)
CA	Los Angeles	4
CA	San Francisco	0.9
New York	New York	8.5

Cardinality

Many to Many (*:*)

Cross filter direction

Single (CityData filters Sales)

Make this relationship active

Apply security filter in both directions

Assume referential integrity

⚠ Warning: The cardinality of this relationship is Many-Many

OK **Cancel**

The resulting **Relationship** view would then display the direct, many-to-many relationship between the two tables. The tables' appearance in the **Fields** list, and their later behavior when the visuals are created, are similar to when we applied the workaround. In the workaround, the extra table that displays the distinct State data isn't made visible. As described earlier, a visual that shows **State**, **Population**, and **Sales** data would be displayed:

State (CityData)	Population (m)	Sales
CA	4.90	140
New York	8.50	
WA	0.90	250
Total	14.30	790

The major differences between *relationships with a many-to-many cardinality* and the more typical *Many-1* relationships are as follows:

- The values shown don't include a blank row that accounts for mismatched rows in the other table. Also, the values don't account for rows where the column used in the relationship in the other table is null.
- You can't use the `RELATED()` function, because more than one row could be related.
- Using the `ALL()` function on a table doesn't remove filters that are applied to other, related tables by a many-to-many relationship. In the preceding example, a measure that's defined as shown here wouldn't remove filters on columns in the related CityData table:

```
Sales total = CALCULATE(SUM('Sales'[Sales]), ALL('Sales'))
```

A visual showing **State**, **Sales**, and **Sales total** data would result in this graphic:

State (CityData)	Sales	Sales total
CA	140	140
WA	250	250
Total	790	790

With the preceding differences in mind, make sure the calculations that use `ALL(<Table>)`, such as *% of grand total*, are returning the intended results.

Considerations and limitations

There are a few limitations for this release of *relationships with a many-to-many cardinality* and composite models.

The following Live Connect (multidimensional) sources can't be used with composite models:

- SAP HANA
- SAP Business Warehouse
- SQL Server Analysis Services
- Power BI datasets
- Azure Analysis Services

When you connect to these multidimensional sources by using DirectQuery, you can't connect to another DirectQuery source or combine it with imported data.

The existing limitations of using DirectQuery still apply when you use *relationships with a many-to-many cardinality*. Many limitations are now per table, depending upon the storage mode of the table. For example, a calculated column on an imported table can refer to other tables, but a calculated column on a DirectQuery table can still refer only to columns on the same table. Other limitations apply to the whole model if any tables within the model are DirectQuery. For example, the QuickInsights and Q&A features are unavailable on a model if any table within it has a storage mode of DirectQuery.

Next steps

For more information about composite models and DirectQuery, see the following articles:

- [Use composite models in Power BI Desktop](#)
- [Storage mode in Power BI Desktop](#)
- [Use DirectQuery in Power BI](#)
- [Power BI data sources](#)

Enable bidirectional cross-filtering for DirectQuery in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

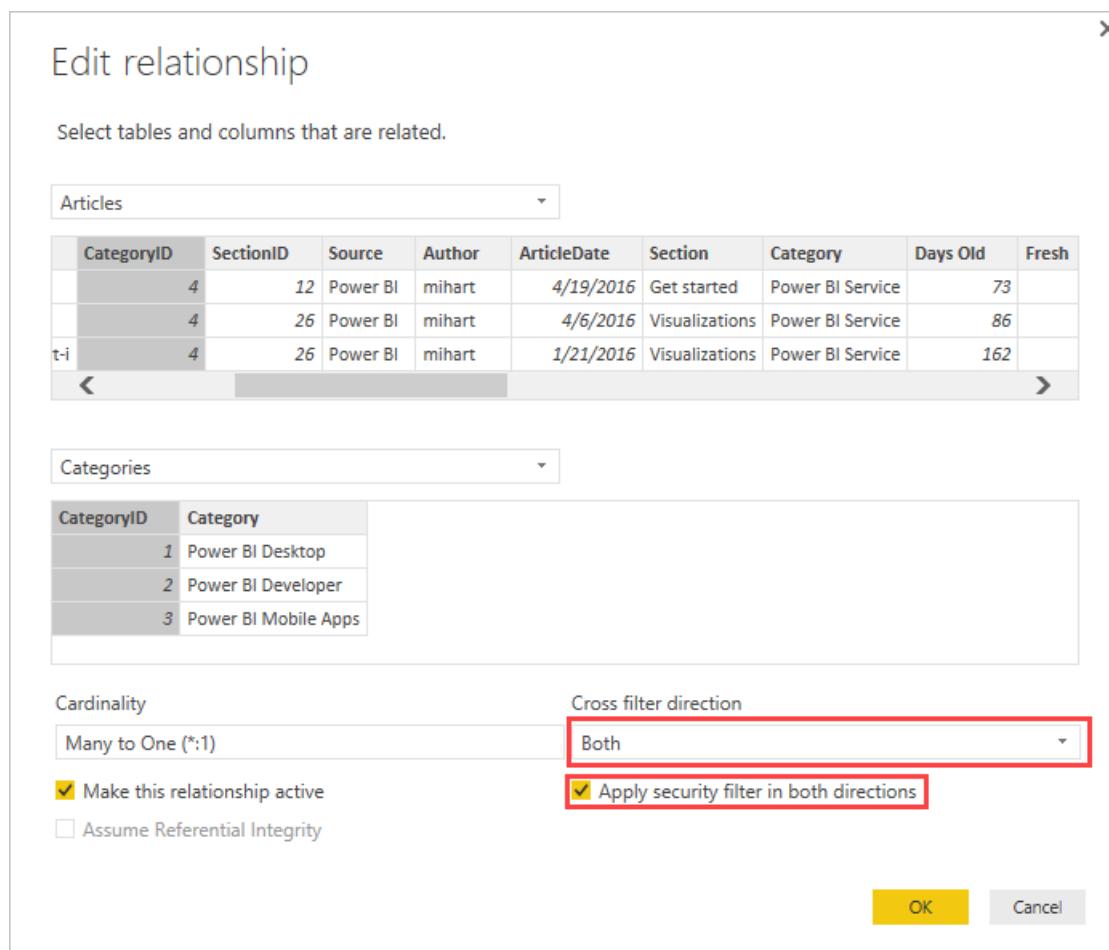
When filtering tables to create the appropriate view of data, report creators and data modelers face challenges determining how to apply filters to a report. Previously, the table's filter context was held on one side of the relationship, but not the other. This arrangement often required complex DAX formula to get the wanted results.

With bidirectional cross-filtering, report creators and data modelers now have more control over how they can apply filters when working with related tables. Bidirectional cross-filtering enables them to apply filters on *both* sides of a table relationship. You can apply the filters by propagating the filter context to a second related table on the other side of a table relationship.

Enable bidirectional cross-filtering for DirectQuery

You can enable cross-filtering in the **Edit relationship** dialog box. To enable cross-filtering for a relationship, you must configure the following options:

- Set **Cross filter direction** to **Both**.
- Select **Apply security filter in both directions**.



NOTE

When creating cross filtering DAX formulas in Power BI Desktop, use *UserPrincipalName*. This field is often the same as a user's login, for example *joe@contoso.com*, instead of *UserName*. As such, you may need to create a related table that maps *UserName* or *EmployeeID* to *UserPrincipalName*.

For more information and for examples of how bidirectional cross-filtering works, check out the [Bidirectional cross-filtering for Power BI Desktop whitepaper](#).

Use composite models in Power BI Desktop

3/30/2022 • 11 minutes to read • [Edit Online](#)

Previously in Power BI Desktop, when you used a DirectQuery in a report, no other data connections, whether DirectQuery or import, were allowed for that report. With composite models, that restriction is removed. A report can seamlessly include data connections from more than one DirectQuery or import data connection, in any combination you choose.

The composite models capability in Power BI Desktop consists of three related features:

- **Composite models:** Allows a report to have two or more data connections from different source groups, such as one or more DirectQuery connections and an import connection, two or more DirectQuery connections, or any combination thereof. This article describes composite models in detail.
- **Many-to-many relationships:** With composite models, you can establish *many-to-many relationships* between tables. This approach removes requirements for unique values in tables. It also removes previous workarounds, such as introducing new tables only to establish relationships. For more information, see [Apply many-many relationships in Power BI Desktop](#).
- **Storage mode:** You can now specify which visuals query back-end data sources. Visuals that don't require a query are imported even if they're based on DirectQuery. This feature helps improve performance and reduce back-end load. Previously, even simple visuals, such as slicers, initiated queries to back-end sources. For more information, see [Manage storage mode in Power BI Desktop](#).

Use composite models

With composite models, you can connect to different kinds of data sources when you use Power BI Desktop or the Power BI service. You can make those data connections in a couple of ways:

- By importing data to Power BI, which is the most common way to get data.
- By connecting directly to data in its original source repository by using DirectQuery. To learn more about DirectQuery, see [Use DirectQuery in Power BI](#).

When you use DirectQuery, composite models make it possible to create a Power BI model, such as a single .pbix Power BI Desktop file, that does either or both of the following actions:

- Combines data from one or more DirectQuery sources.
- Combines data from DirectQuery sources and import data.

For example, by using composite models, you can build a model that combines the following types of data:

- Sales data from an enterprise data warehouse.
- Sales-target data from a departmental SQL Server database.
- Data that's imported from a spreadsheet.

A model that combines data from more than one DirectQuery source or that combines DirectQuery with import data is called a composite model.

You can create relationships between tables as you always have, even when those tables come from different sources. Any relationships that are cross-source are created with a cardinality of many-to-many, regardless of their actual cardinality. You can change them to one-to-many, many-to-one, or one-to-one. Whichever cardinality you set, cross-source relationships have different behavior. You can't use Data Analysis Expressions (DAX) functions to retrieve values on the `one` side from the `many` side. You may also see a performance impact

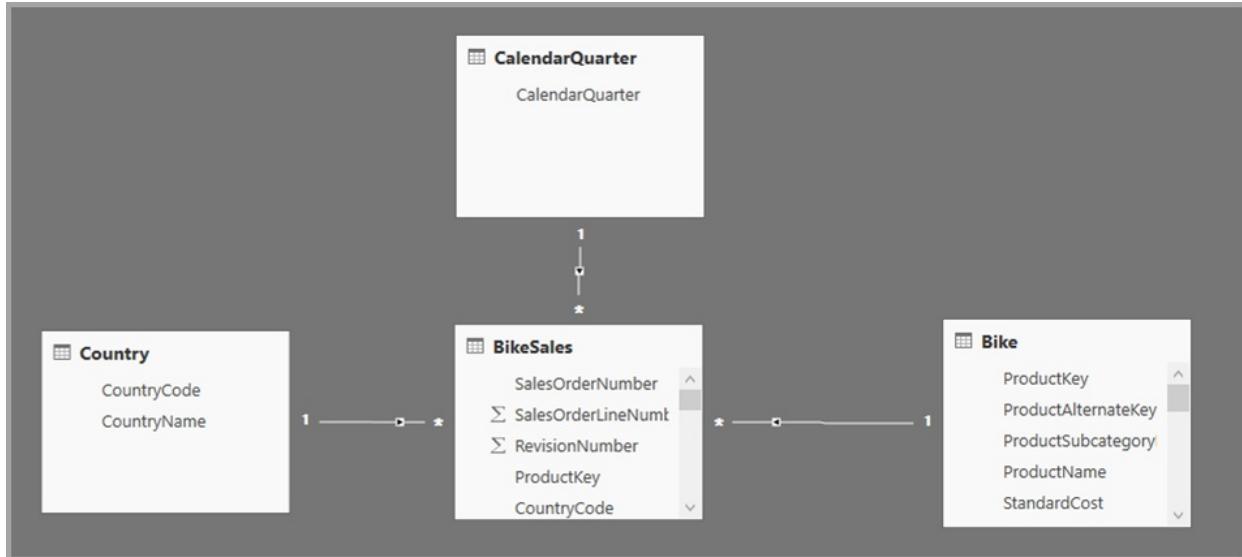
versus many-to-many relationships within the same source.

NOTE

Within the context of composite models, all imported tables are effectively a single source, regardless of the actual underlying data sources.

Example of a composite model

For an example of a composite model, consider a report that has connected to a corporate data warehouse in SQL Server by using DirectQuery. In this instance, the data warehouse contains **Sales by Country, Quarter**, and **Bike (Product)** data, as shown in the following image:



At this point, you could build simple visuals by using fields from this source. The following image shows total sales by *ProductName*, for a selected quarter.

The screenshot shows a report interface with the following details:

- CalendarQuarter** filter: Set to "2004 Q2".
- Report Data:**

ProductName	SalesAmount
Mountain-200 Black, 38	\$725,631.7742
Mountain-200 Black, 42	\$638,035.6779
Mountain-200 Black, 46	\$546,907.133
Mountain-200 Silver, 38	\$668,400.255
Mountain-200 Silver, 42	\$570,032.679
Mountain-200 Silver, 46	\$547,639.2075
Mountain-400-W Silver, 38	\$70,485.284
Mountain-400-W Silver, 40	\$82,951.022
Mountain-400-W Silver, 42	\$66,330.038
Mountain-400-W Silver, 46	\$72,024.264
Mountain-500 Black, 40	\$24,299.55
Mountain-500 Black, 42	\$28,511.472
Total	\$12,299,251.4178

But what if you have data in an Office Excel spreadsheet about the product manager who's assigned to each product, along with the marketing priority? If you want to view **Sales Amount** by **Product Manager**, it might not be possible to add this local data to the corporate data warehouse. Or it might take months at best.

It might be possible to import that sales data from the data warehouse, instead of using DirectQuery. And the sales data could then be combined with the data that you imported from the spreadsheet. However, that approach is unreasonable, for the reasons that lead to using DirectQuery in the first place. The reasons could include:

- Some combination of the security rules enforced in the underlying source.
- The need to be able to view the latest data.
- The sheer scale of the data.

Here's where composite models come in. Composite models let you connect to the data warehouse by using DirectQuery and then use **Get data** for additional sources. In this example, we first establish the DirectQuery connection to the corporate data warehouse. We use **Get data**, choose **Excel**, and then navigate to the spreadsheet that contains our local data. Finally, we import the spreadsheet that contains the *Product Names*, the assigned **Sales Manager**, and the **Priority**.

The screenshot shows the Power BI Navigator dialog. On the left, there is a tree view under 'Display Options' showing 'ProductManagers.xlsx [2]' with a checked checkbox next to 'ProductManagers'. Below it is 'Sheet1'. On the right, a table titled 'ProductManagers' is displayed with the following data:

ProductName	Product Manager	Priority
Mountain-200 Black, 38	Andersen	M
Mountain-200 Black, 42	Andersen	L
Mountain-200 Black, 46	Bell	L
Mountain-200 Silver, 38	Adams	L
Mountain-200 Silver, 42	Morris	H
Mountain-200 Silver, 46	Morris	M
Mountain-400-W Silver, 38	Morris	H
Mountain-400-W Silver, 40	Morris	H
Mountain-400-W Silver, 42	Nicholls	L
Mountain-400-W Silver, 46	Nicholls	M
Mountain-500 Black, 40	Nicholls	M
Mountain-500 Black, 42	Nicholls	H
Mountain-500 Black, 44	Nicholls	M
Mountain-500 Black, 48	Adams	L
Mountain-500 Black, 52	Bell	H
Mountain-500 Silver, 40	Hall	M
Mountain-500 Silver, 42	Hall	L
Mountain-500 Silver, 44	Hall	H
Mountain-500 Silver, 48	Hall	L
Mountain-500 Silver, 52	Hall	L
Road-250 Black, 44	Lee	L
Road-250 Black, 48	Lee	L
Road-250 Black, 52	Lee	L
Road-250 Black, 58	Lee	L

At the bottom of the dialog are three buttons: 'Load' (highlighted in yellow), 'Edit', and 'Cancel'.

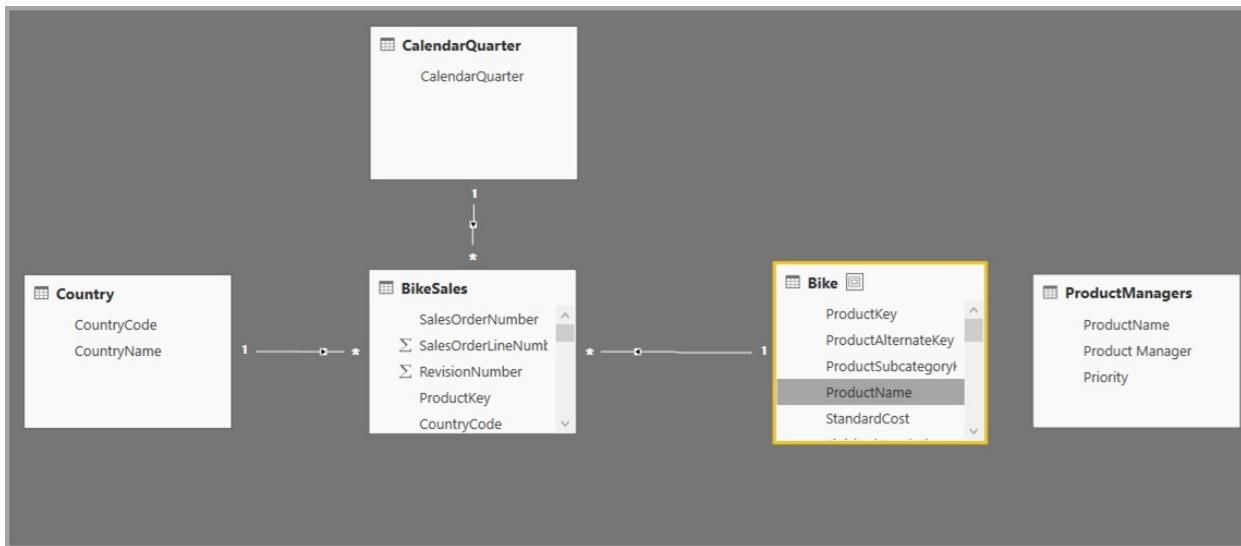
In the **Fields** list, you can see two tables: the original **Bike** table from SQL Server and a new **ProductManagers** table. The new table contains the data that's imported from Excel.

Fields

Search

- ✓ **Bike**
- ✓ **BikeSales**
- ✓ **CalendarQuarter**
- ✓ **Country**
- ✓ **ProductManagers**
- Priority
- Product Mana...
- ProductName
- Σ Size
- Style
- ✓ **SalesTargets**
- ✓ **TopSales**

Similarly, in the **Relationship** view in Power BI Desktop, we now see an additional table called **ProductManagers**.



We now need to relate these tables to the other tables in the model. As always, we create a relationship between the **Bike** table from SQL Server and the imported **ProductManagers** table. That is, the relationship is between **Bike[ProductName]** and **ProductManagers[ProductName]**. As discussed earlier, all relationships that go across source default to many-to-many cardinality.

Create relationship

Select tables and columns that are related.

Bike						
ProductKey	ProductAlternateKey	ProductSubcategoryKey	ProductName	StandardCost	FinishedGoodsFlag	
310	BK-R93R-62		2	Road-150 Red, 62	\$2,171.2942	1
311	BK-R93R-44		2	Road-150 Red, 44	\$2,171.2942	1
312	BK-R93R-48		2	Road-150 Red, 48	\$2,171.2942	1

ProductManagers		
ProductName	Product Manager	Priority
Mountain-200 Black, 38	Andersen	M
Mountain-200 Black, 42	Andersen	L
Mountain-200 Black, 46	Bell	L

Cardinality **Cross filter direction**

Many to Many (*:*) Single (ProductManagers filters Bike)

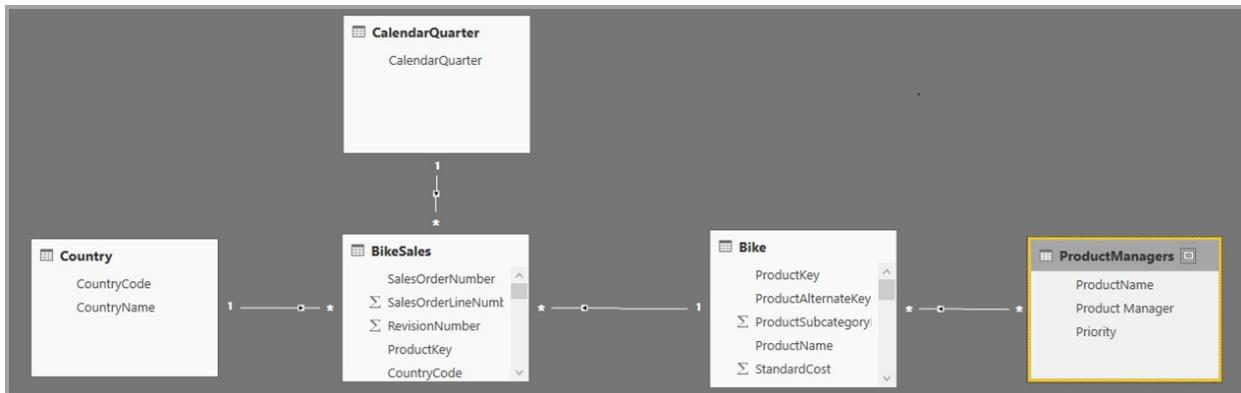
Make this relationship active Apply security filter in both directions

Assume referential integrity

Warning: The cardinality of this relationship is Many-Many

OK Cancel

Now that we've established this relationship, it's displayed in the **Relationship** view in Power BI Desktop, as we would expect.



We can now create visuals by using any of the fields in the **Fields** list. This approach seamlessly blends data from multiple sources. For example, the total *SalesAmount* for each *Product Manager* is displayed in the following image:

The screenshot shows the Power BI Data View interface. On the left, a table is displayed with columns 'Product Manager' and 'SalesAmount'. The last row, 'Total', has a value of '\$12,299,251.4178'. A red box highlights the 'SalesAmount' column header. On the right, the 'Fields' pane is open, showing a list of fields under the 'BikeSales' node. The 'SalesAmount' field is selected and highlighted with a red box. Other fields listed include 'CountryCode' and 'ProductKey'. Below the Fields pane, there are sections for 'Values', 'Drillthrough', 'Cross-report', and 'Keep all filters'.

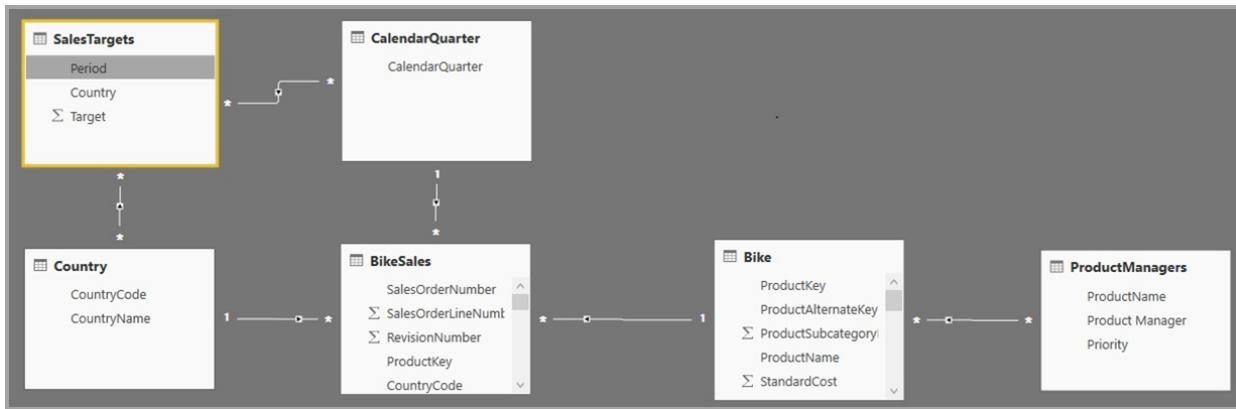
The following example displays a common case of a *dimension* table, such as **Product** or **Customer**, that's extended with some extra data imported from somewhere else. It's also possible to have tables use DirectQuery to connect to various sources. To continue with our example, imagine that **Sales Targets** per **Country** and **Period** are stored in a separate departmental database. As usual, you can use **Get data** to connect to that data, as shown in the following image:

The screenshot shows the Power BI Navigator window. The title bar says 'Navigator'. On the left, the 'Display Options' dropdown is set to 'Sales Targets [1]'. Underneath, 'SalesTargets' is selected and highlighted with a red box. In the center, a preview of the 'SalesTargets' table is shown, with the heading 'SalesTargets' and the note 'Preview downloaded on Friday, June 8, 2018'. The table has three columns: 'Period', 'Country', and 'Target'. The data consists of 24 rows of quarterly sales targets for various countries. At the bottom of the Navigator window, there are buttons for 'Select Related Tables', 'Load' (highlighted with a red box), 'Edit', and 'Cancel'.

Period	Country	Target
2001 Q1	AU	0
2001 Q1	CA	0
2001 Q1	DE	0
2001 Q1	FR	0
2001 Q1	GB	0
2001 Q1	US	0
2001 Q2	AU	0
2001 Q2	CA	0
2001 Q2	DE	0
2001 Q2	FR	0
2001 Q2	GB	0
2001 Q2	US	0
2001 Q3	AU	666803
2001 Q3	CA	901742
2001 Q3	DE	68973
2001 Q3	FR	37865
2001 Q3	GB	278499
2001 Q3	US	2499123
2001 Q4	AU	773149
2001 Q4	CA	1222830
2001 Q4	DE	97477
2001 Q4	FR	34364
2001 Q4	GB	246364

As we did earlier, we can create relationships between the new table and other tables in the model and then

create visuals that combine the table data. Let's look again at the **Relationships** view, where we've established the new relationships:



The next image is based on the new data and relationships we created. The visual at the lower left shows total *Sales Amount* versus *Target*, and the variance calculation shows the difference. The **Sales Amount** and **Target** data come from two different SQL Server databases.

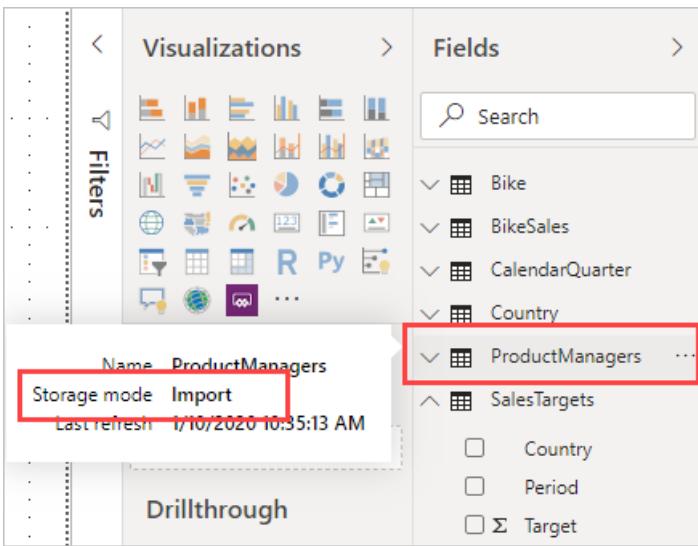
The screenshot shows the Power BI desktop interface with the following components:

- Table View:** Displays a table of sales data with columns **ProductName** and **SalesAmount**. The data includes various product models and their sales amounts.
- Visualizations View:** Shows a stacked bar chart titled "SalesAmount, Target and Variance by CountryCode". The chart displays SalesAmount (teal), Target (black), and Variance (red) for countries US, AU, CA, GB, FR, and DE. The Y-axis ranges from \$0M to \$5M.
- Fields pane:** On the right, the Fields pane lists available fields categorized by table. Selected fields include **CountryCode** (under **Country**), **Priority**, **ProductName**, **ProductManager**, **Size**, and **Style** (under **Bike**); and **Target**, **Variance**, and **SalesAmount** (under **SalesTargets**).

Set the storage mode

Each table in a composite model has a storage mode that indicates whether the table is based on DirectQuery or import. The storage mode can be viewed and modified in the **Property** pane. To display the storage mode, right-click a table in the **Fields** list, and then select **Properties**. The following image shows the storage mode for the **SalesTargets** table.

The storage mode can also be viewed on the tooltip for each table.



For any Power BI Desktop file (a `.pbix` file) that contains some tables from DirectQuery and some import tables, the status bar displays a storage mode called **Mixed**. You can click that term in the status bar and easily switch all tables to import.

For more information about storage mode, see [Manage storage mode in Power BI Desktop](#).

NOTE

You can use *Mixed* storage mode in Power BI Desktop and in the Power BI service.

Calculated tables

You can add calculated tables to a model that uses DirectQuery. The Data Analysis Expressions (DAX) that define the calculated table can reference either imported or DirectQuery tables or a combination of the two.

Calculated tables are always imported, and their data is refreshed when you refresh the tables. If a calculated table refers to a DirectQuery table, visuals that refer to the DirectQuery table always show the latest values in the underlying source. Alternatively, visuals that refer to the calculated table show the values at the time when the calculated table was last refreshed.

Security implications

Composite models have some security implications. A query sent to one data source can include data values that have been retrieved from another source. In the earlier example, the visual that shows (**Sales Amount**) by **Product Manager** sends an SQL query to the Sales relational database. That SQL query might contain the names of Product Managers and their associated Products.

```

SELECT ...
FROM ...
inner join (
(SELECT N'Andersen' AS [c52],N'Mountain-200 Black, 38' AS [c4] ) UNION ALL
(SELECT N'Andersen' AS [c52],N'Mountain-200 Black, 42' AS [c4] ) UNION ALL
(SELECT N'Bell' AS [c52],N'Mountain-200 Black, 46' AS [c4] ) UNION ALL
(SELECT N'Bell' AS [c52],N'Mountain-500 Black, 52' AS [c4] ) UNION ALL
...

```

Consequently, information that's stored in the spreadsheet is now included in a query that's sent to the relational database. If this information is confidential, you should consider the security implications. In particular, consider the following points:

- Any administrator of the database who can view traces or audit logs could view this information, even without permissions to the data in its original source. In this example, the administrator would need permissions to the Excel file.
- The encryption settings for each source should be considered. You want to avoid retrieving information from one source by an encrypted connection and then inadvertently including it in a query that's sent to another source by an unencrypted connection.

To allow confirmation that you've considered any security implications, Power BI Desktop displays a warning message when you create a composite model.

Additionally, if an author adds *Table1* from *Model A* to a Composite Model (we'll call it *Model C* for reference), then a user viewing a report built on *Model C* could query **any table** in *Model A* that is not protected by RLS.

For similar reasons, be careful when you open a Power BI Desktop file that's sent from an untrusted source. If the file contains composite models, information that someone retrieves from one source by using the credentials of the user who opens the file would be sent to another data source as part of the query. The information could be viewed by the malicious author of the Power BI Desktop file. When you initially open a Power BI Desktop file that contains multiple sources, Power BI Desktop displays a warning. The warning is similar to the one that's displayed when you open a file that contains native SQL queries.

Performance implications

When you use DirectQuery, you should always consider performance, primarily to ensure that the back-end source has sufficient resources to provide a good experience for users. A good experience means that the visuals refresh in five seconds or less. For more performance advice, see [About using DirectQuery in Power BI](#).

Using composite models adds additional performance considerations. A single visual can result in sending queries to multiple sources, which often pass the results from one query across to a second source. This situation can result in the following forms of execution:

- **An SQL query that includes a large number of literal values:** For example, a visual that requests total **Sales Amount** for a set of selected **Product Managers** would first need to find which **Products** were managed by those product managers. This sequence must happen before the visual sends an SQL query that includes all of the product IDs in a `WHERE` clause.
- **An SQL query that queries at a lower level of granularity, with the data later being aggregated locally:** As the number of **Products** that meet the filter criteria on **Product Manager** grows large, it can become inefficient or unfeasible to include all products in a `WHERE` clause. Instead, you can query the relational source at the lower level of **Products** and then aggregate the results locally. If the cardinality of **Products** exceeds a limit of 1 million, the query fails.
- **Multiple SQL queries, one per group by value:** When the aggregation uses **DistinctCount** and is grouped by a column from another source, and if the external source doesn't support efficient passing of many literal values that define the grouping, it's necessary to send one SQL query per group by value.

A visual that requests a distinct count of **CustomerAccountNumber** from the SQL Server table by **Product Managers** imported from the spreadsheet would need to pass in the details from the **Product Managers** table in the query that's sent to SQL Server. Over other sources, Redshift, for example, this action is unfeasible. Instead, there would be one SQL query sent per **Sales Manager**, up to some practical limit, at which point the query would fail.

Each of these cases has its own implications on performance, and the exact details vary for each data source.

Although the cardinality of the columns used in the relationship that joins the two sources remains low, a few thousand, performance shouldn't be affected. As this cardinality grows, you should pay more attention to the impact on the resulting performance.

Additionally, the use of many-to-many relationships means that separate queries must be sent to the underlying source for each total or subtotal level, rather than aggregating the detailed values locally. A simple table visual with totals would send two SQL queries, rather than one.

Considerations and limitations

This release of composite models presents a few limitations:

Currently, [incremental refresh](#) is supported for composite models connecting to SQL, Oracle, and Teradata data sources only.

The following Live Connect tabular sources can't be used with composite models:

- SAP HANA
- SAP Business Warehouse
- SQL Server Analysis Services
- Power BI datasets
- [Usage Metrics \(classic workspaces\)](#)
- Azure Analysis Services

The existing limitations of DirectQuery still apply when you use composite models. Many of these limitations are now per table, depending upon the storage mode of the table. For example, a calculated column on an import table can refer to other tables, but a calculated column on a DirectQuery table can still refer only to columns on the same table. Other limitations apply to the model as a whole, if any of the tables within the model are DirectQuery. For example, the QuickInsights feature isn't available on a model if any of the tables within it has a storage mode of DirectQuery.

Next steps

For more information about composite models and DirectQuery, see the following articles:

- [Many-to-many relationships in Power BI Desktop](#)
- [Storage mode in Power BI Desktop](#)
- [Use DirectQuery in Power BI](#)
- [Data sources supported by DirectQuery in Power BI](#)
- [Using DirectQuery for Power BI datasets and Azure Analysis Services \(preview\)](#)

User-defined aggregations

3/30/2022 • 14 minutes to read • [Edit Online](#)

Aggregations in Power BI can improve query performance over very large DirectQuery datasets. By using aggregations, you cache data at the aggregated level in-memory. Aggregations in Power BI can be manually configured in the data model, as described in this article, or for Premium subscriptions, automatically by enabling the [Automatic aggregations](#) feature in dataset Settings.

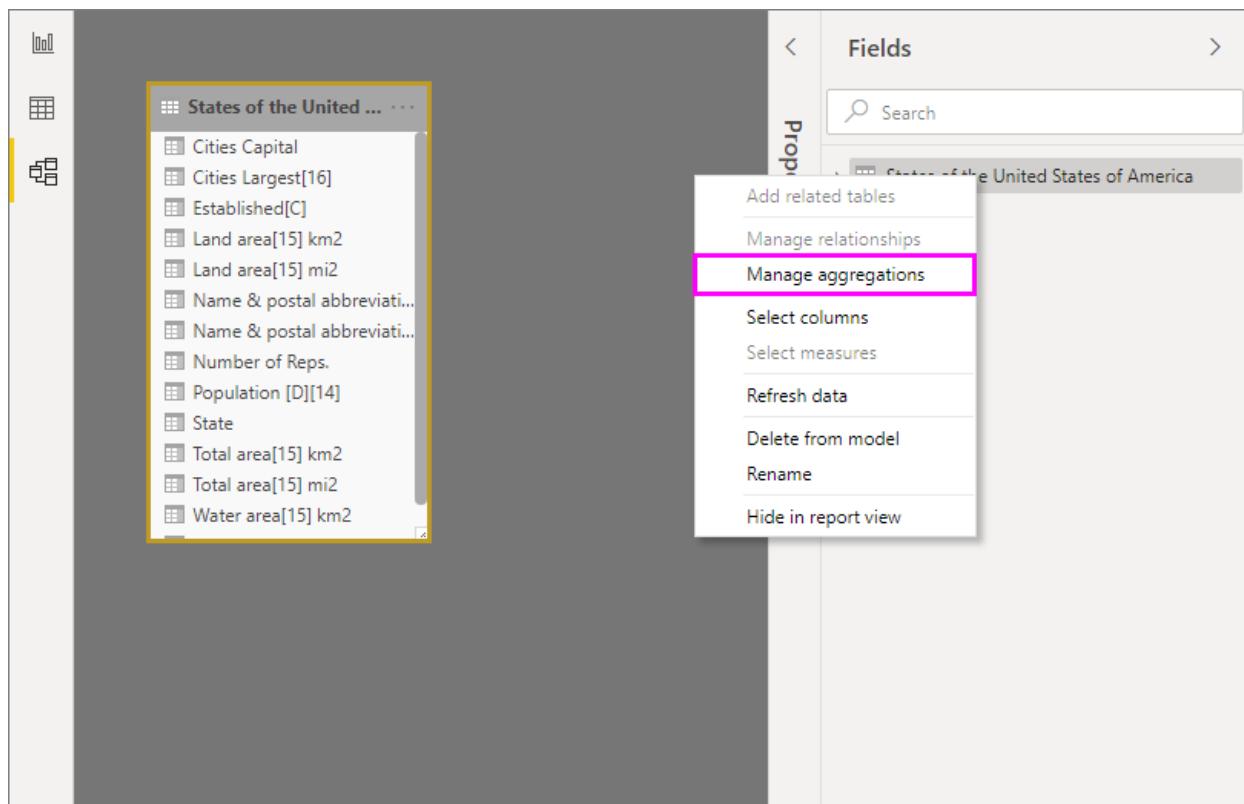
Creating aggregation tables

Depending on the data source type, an aggregations table can be created at the data source as a table or view, as a native query, or for the greatest performance, as an import table created in Power Query. You then use the Manage aggregations dialog in Power BI Desktop to define aggregations for aggregation columns with summarization, detail table, and detail column properties.

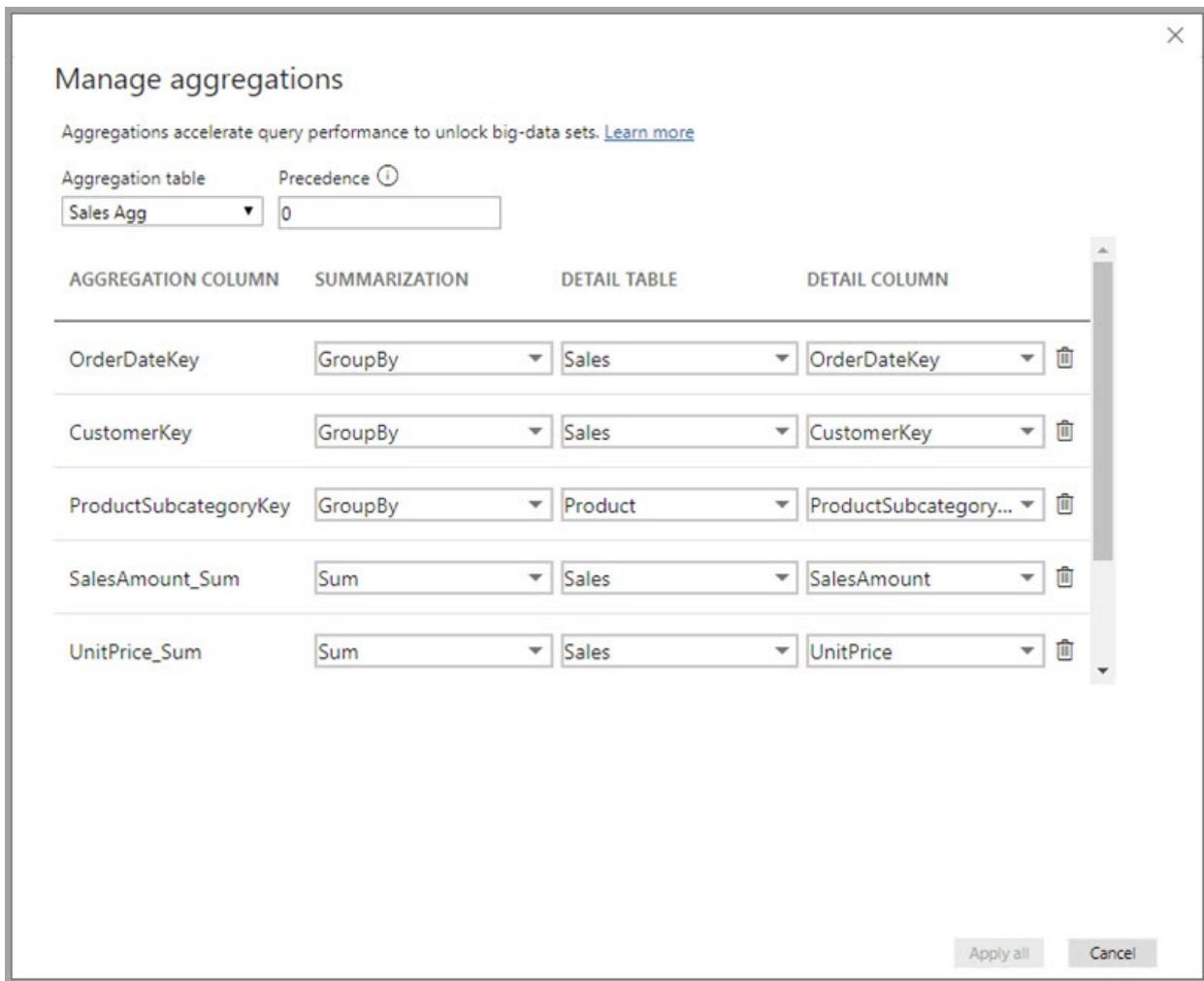
Dimensional data sources, like data warehouses and data marts, can use [relationship-based aggregations](#). Hadoop-based big-data sources often [base aggregations on GroupBy columns](#). This article describes typical Power BI data modeling differences for each type of data source.

Manage aggregations

In the Fields pane of any Power BI Desktop view, right-click the aggregations table, and then select **Manage aggregations**.



The **Manage aggregations** dialog shows a row for each column in the table, where you can specify the aggregation behavior. In the following example, queries to the **Sales** detail table are internally redirected to the **Sales Agg** aggregation table.



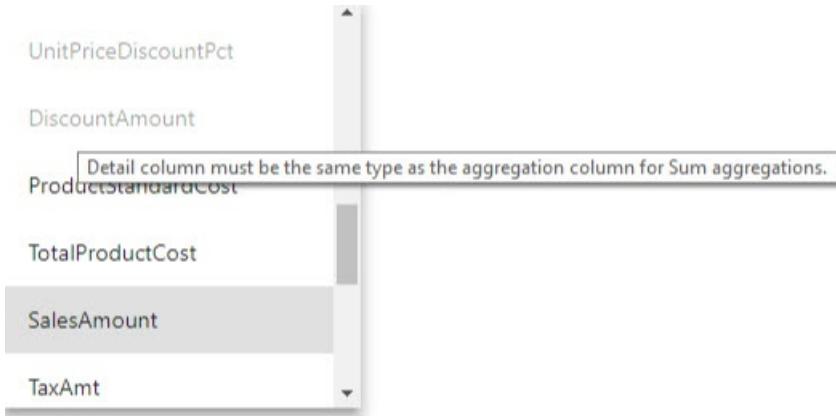
In this relationship-based aggregation example, the GroupBy entries are optional. Except for DISTINCTCOUNT, they don't affect aggregation behavior and are primarily for readability. Without the GroupBy entries, the aggregations would still get hit, based on the relationships. This is different from the [big data example](#) later in this article, where the GroupBy entries are required.

Validations

The Manage aggregations dialog enforces validations:

- The **Detail Column** must have the same datatype as the **Aggregation Column**, except for the Count and Count table rows **Summarization** functions. Count and Count table rows are only available for integer aggregation columns, and don't require a matching datatype.
- Chained aggregations covering three or more tables aren't allowed. For example, aggregations on **Table A** can't refer to a **Table B** that has aggregations referring to a **Table C**.
- Duplicate aggregations, where two entries use the same **Summarization** function and refer to the same **Detail Table** and **Detail Column**, aren't allowed.
- The **Detail Table** must use DirectQuery storage mode, not Import.
- Grouping by a foreign key column used by an inactive relationship, and relying on the USERELATIONSHIP function for aggregation hits, isn't supported.
- Aggregations based on GroupBy columns can leverage relationships between aggregation tables but authoring relationships between aggregation tables is not supported in Power BI Desktop. If necessary, you can create relationships between aggregation tables by using a third-party tool or a scripting solution through XMLA endpoints.

Most validations are enforced by disabling dropdown values and showing explanatory text in the tooltip.



Aggregation tables are hidden

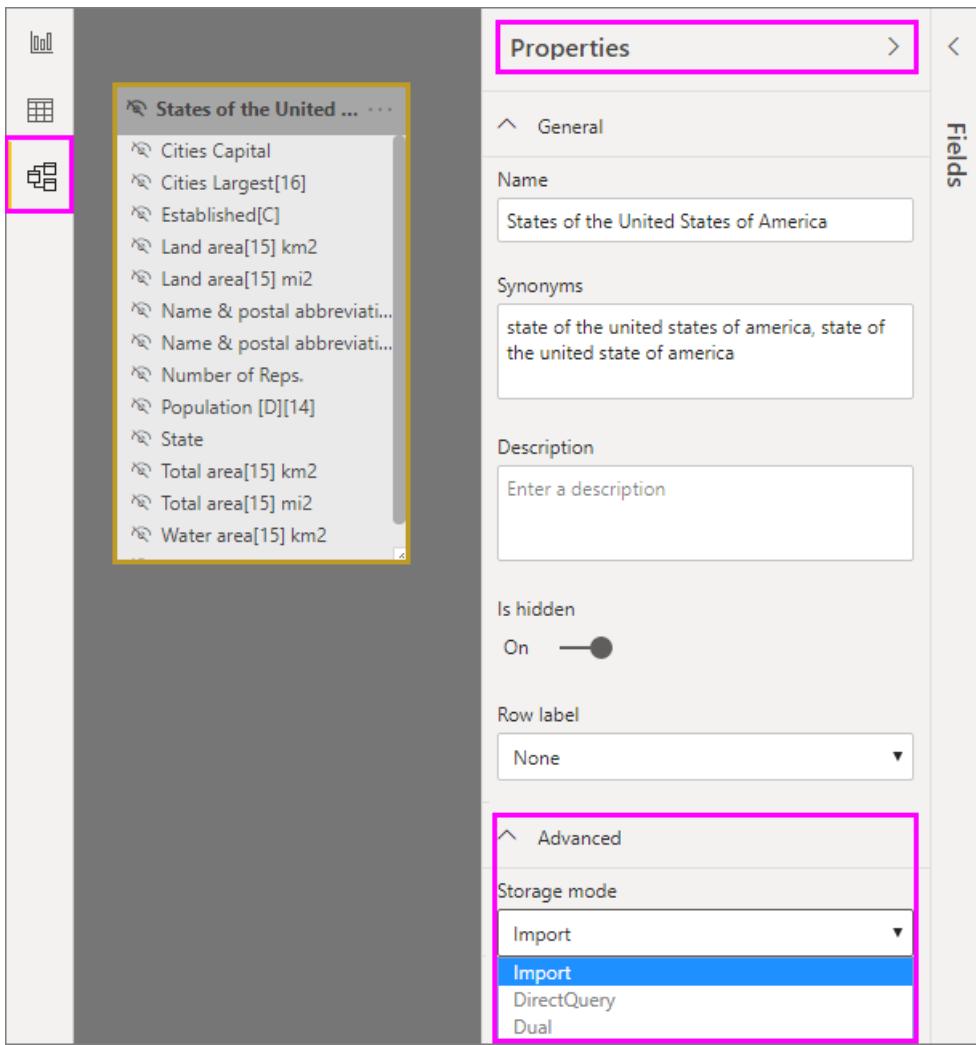
Users with read-only access to the dataset can't query aggregation tables. This avoids security concerns when used with *row-level security (RLS)*. Consumers and queries refer to the detail table, not the aggregation table, and don't need to know about the aggregation table.

For this reason, aggregation tables are hidden from **Report** view. If the table isn't already hidden, the **Manage aggregations** dialog will set it to hidden when you select **Apply all**.

Storage modes

The aggregation feature interacts with table-level storage modes. Power BI tables can use *DirectQuery*, *Import*, or *Dual* storage modes. DirectQuery queries the backend directly, while Import caches data in memory and sends queries to the cached data. All Power BI Import and non-multidimensional DirectQuery data sources can work with aggregations.

To set the storage mode of an aggregated table to Import to speed up queries, select the aggregated table in Power BI Desktop **Model** view. In the **Properties** pane, expand **Advanced**, drop down the selection under **Storage mode**, and select **Import**. Note that this action is irreversible.



To learn more about table storage modes, see [Manage storage mode in Power BI Desktop](#).

RLS for aggregations

To work correctly for aggregations, RLS expressions should filter both the aggregation table and the detail table.

In the following example, the RLS expression on the **Geography** table works for aggregations, because **Geography** is on the filtering side of relationships to both the **Sales** table and the **Sales Agg** table. Queries that hit the aggregation table and those that don't will both have RLS successfully applied.

An RLS expression on the **Product** table filters only the detail **Sales** table, not the aggregated **Sales Agg** table. Since the aggregation table is another representation of the data in the detail table, it would be insecure to answer queries from the aggregation table if the RLS filter can't be applied. Filtering only the detail table isn't recommended, because user queries from this role won't benefit from aggregation hits.

An RLS expression that filters only the **Sales Agg** aggregation table and not the **Sales** detail table isn't allowed.

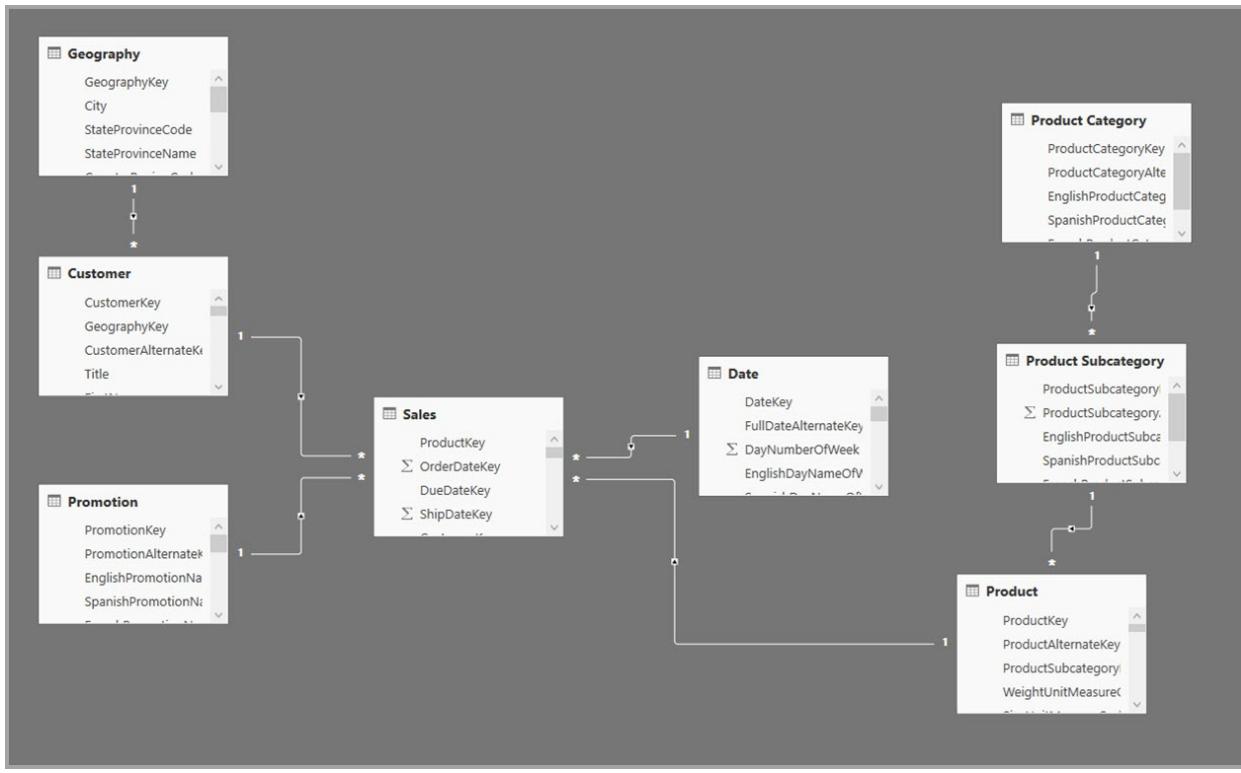


For [aggregations based on GroupBy columns](#), an RLS expression applied to the detail table can be used to filter the aggregation table, because all the GroupBy columns in the aggregation table are covered by the detail table. On the other hand, an RLS filter on the aggregation table can't be applied to the detail table, so is disallowed.

Aggregation based on relationships

Dimensional models typically use *aggregations based on relationships*. Power BI datasets from data warehouses and data marts resemble star/snowflake schemas, with relationships between dimension tables and fact tables.

In the following example, the model gets data from a single data source. Tables are using DirectQuery storage mode. The **Sales** fact table contains billions of rows. Setting the storage mode of **Sales** to Import for caching would consume considerable memory and resources overhead.

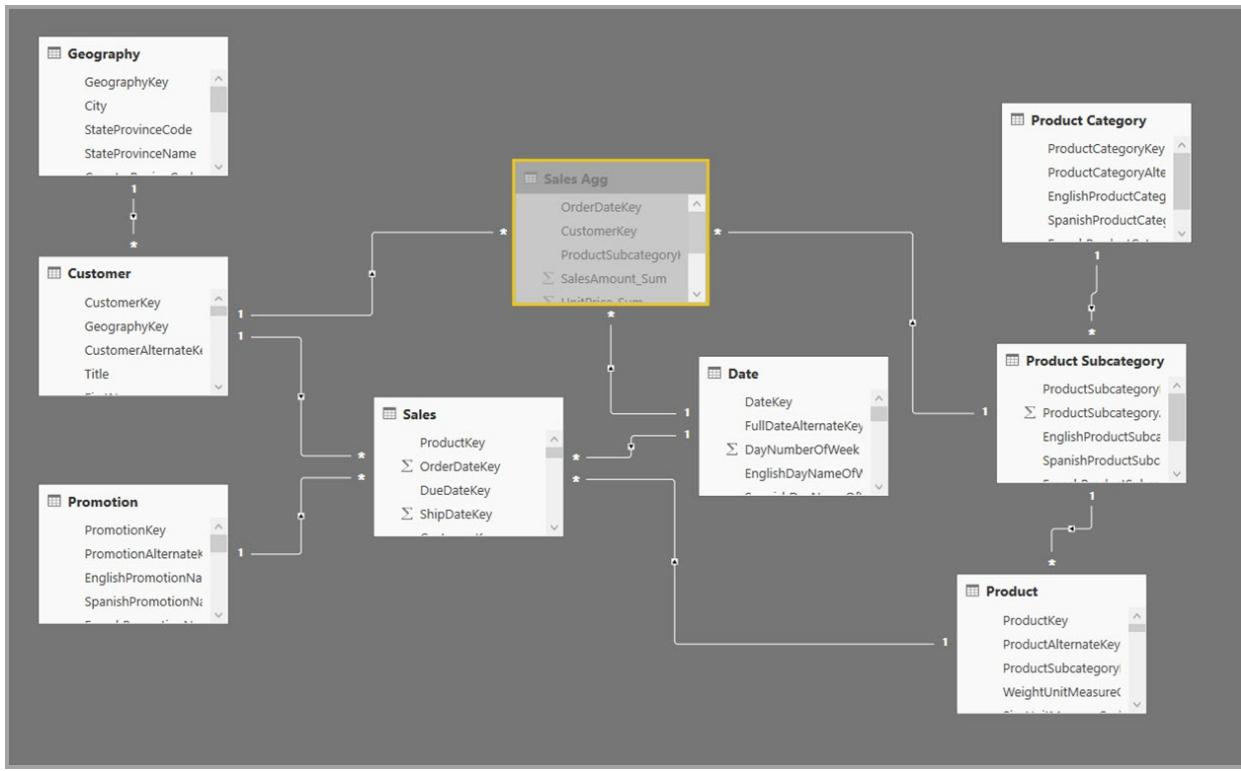


Instead, create the **Sales Agg** aggregation table. In the **Sales Agg** table, the number of rows equals the sum of **SalesAmount** grouped by **CustomerKey**, **DateKey**, and **ProductSubcategoryKey**. The **Sales Agg** table is at a higher granularity than **Sales**, so instead of billions, it might contain millions of rows, which are much easier to manage.

If the following dimension tables are the most commonly used for the queries with high business value, they can filter **Sales Agg**, using *one-to-many* or *many-to-one* relationships.

- Geography
- Customer
- Date
- Product Subcategory
- Product Category

The following image shows this model.



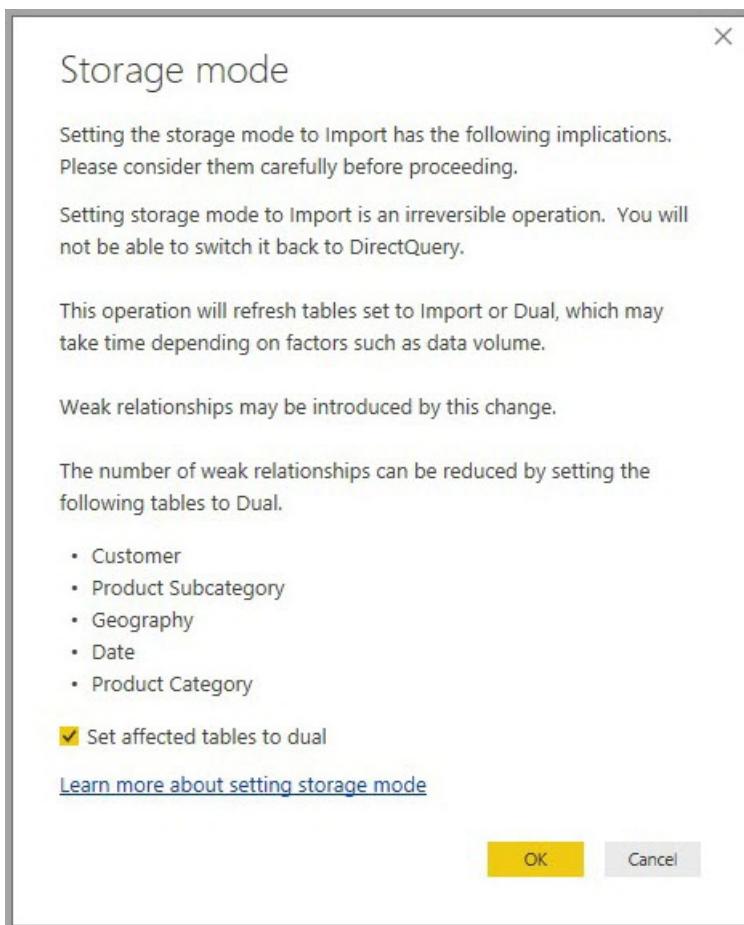
The following table shows the aggregations for the **Sales Agg** table.

Aggregation Column	Summarization	Detail Table	Detail Column
OrderDateKey	GroupBy	Sales	OrderDateKey
CustomerKey	GroupBy	Sales	CustomerKey
ProductSubcategoryKey	GroupBy	Product	ProductSubcategoryKey
SalesAmount_Sum	Sum	Sales	SalesAmount
UnitPrice_Sum	Sum	Sales	UnitPrice
UnitPrice_Count	Count	Sales	UnitPrice
FactInternetSales_Count	Count table rows	Sales	N/A

NOTE

The **Sales Agg** table, like any table, has the flexibility of being loaded in a variety of ways. The aggregation can be performed in the source database using ETL/ELT processes, or by the [M expression](#) for the table. The aggregated table can use Import storage mode, with or without [Incremental refresh for datasets](#), or it can use DirectQuery and be optimized for fast queries using [columnstore indexes](#). This flexibility enables balanced architectures that can spread query load to avoid bottlenecks.

Changing the storage mode of the aggregated **Sales Agg** table to **Import** opens a dialog box saying that the related dimension tables can be set to storage mode *Dual*.



Setting the related dimension tables to Dual lets them act as either Import or DirectQuery, depending on the subquery. In the example:

- Queries that aggregate metrics from the Import-mode **Sales** **Agg** table, and group by attribute(s) from the related Dual tables, can be returned from the in-memory cache.
- Queries that aggregate metrics from the DirectQuery **Sales** table, and group by attribute(s) from the related Dual tables, can be returned in DirectQuery mode. The query logic, including the GroupBy operation, is passed down to the source database.

For more information about Dual storage mode, see [Manage storage mode in Power BI Desktop](#).

Regular vs. limited relationships

Aggregation hits based on relationships require regular relationships.

Regular relationships include the following storage mode combinations, where both tables are from a single source:

TABLE ON THE MANY SIDES	TABLE ON THE 1 SIDE
Dual	Dual
Import	Import or Dual
DirectQuery	DirectQuery or Dual

The only case where a *cross-source* relationship is considered regular is if both tables are set to Import. Many-to-many relationships are always considered limited.

For *cross-source* aggregation hits that don't depend on relationships, see [Aggregations based on GroupBy columns](#).

Relationship-based aggregation query examples

The following query hits the aggregation, because columns in the **Date** table are at the granularity that can hit the aggregation. The **SalesAmount** column uses the **Sum** aggregation.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarYear],  
    "Sales", SUM('Sales'[SalesAmount])  
)
```

The following query doesn't hit the aggregation. Despite requesting the sum of **SalesAmount**, the query is performing a GroupBy operation on a column in the **Product** table, which isn't at the granularity that can hit the aggregation. If you observe the relationships in the model, a product subcategory can have multiple **Product** rows. The query wouldn't be able to determine which product to aggregate to. In this case, the query reverts to DirectQuery and submits a SQL query to the data source.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Product'[EnglishProductName],  
    "Sales", SUM('Sales'[SalesAmount])  
)
```

Aggregations aren't just for simple calculations that perform a straightforward sum. Complex calculations can also benefit. Conceptually, a complex calculation is broken down into subqueries for each **SUM**, **MIN**, **MAX**, and **COUNT**, and each subquery is evaluated to determine if it can hit the aggregation. This logic doesn't hold true in all cases due to query-plan optimization, but in general it should apply. The following example hits the aggregation:

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarYear],  
    "Sales Amount/Count", DIVIDE(SUM('Sales'[SalesAmount]), COUNTROWS('Sales'))  
)
```

The **COUNTROWS** function can benefit from aggregations. The following query hits the aggregation because there is a **Count table rows** aggregation defined for the **Sales** table.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarYear],  
    "Sales Count", COUNTROWS('Sales')  
)
```

The **AVERAGE** function can benefit from aggregations. The following query hits the aggregation because **AVERAGE** internally gets folded to a **SUM** divided by a **COUNT**. Since the **UnitPrice** column has aggregations

defined for both SUM and COUNT, the aggregation is hit.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarYear],  
    "Avg Unit Price", AVERAGE('Sales'[UnitPrice])  
)
```

In some cases, the DISTINCTCOUNT function can benefit from aggregations. The following query hits the aggregation because there is a GroupBy entry for **CustomerKey**, which maintains the distinctness of **CustomerKey** in the aggregation table. This technique might still hit the performance threshold where more than two to five million distinct values can affect query performance. However, it can be useful in scenarios where there are billions of rows in the detail table, but two to five million distinct values in the column. In this case, the DISTINCTCOUNT can perform faster than scanning the table with billions of rows, even if it were cached into memory.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarYear],  
    "Distinct Customer Count", DISTINCTCOUNT('Sales'[CustomerKey])  
)
```

DAX time-intelligence functions are aggregation aware. The following query hits the aggregation because the DATESYTD function generates a table of **CalendarDay** values, and the aggregation table is at a granularity that is covered for group-by columns in the **Date** table. This is an example of a table-valued filter to the CALCULATE function, which can work with aggregations.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarMonth],  
    'Product Category'[CategoryName],  
    "Sales", CALCULATE(SUM('Sales'[SalesAmount]), DATESYTD('Date'[CalendarDay]))  
)
```

Aggregation based on GroupBy columns

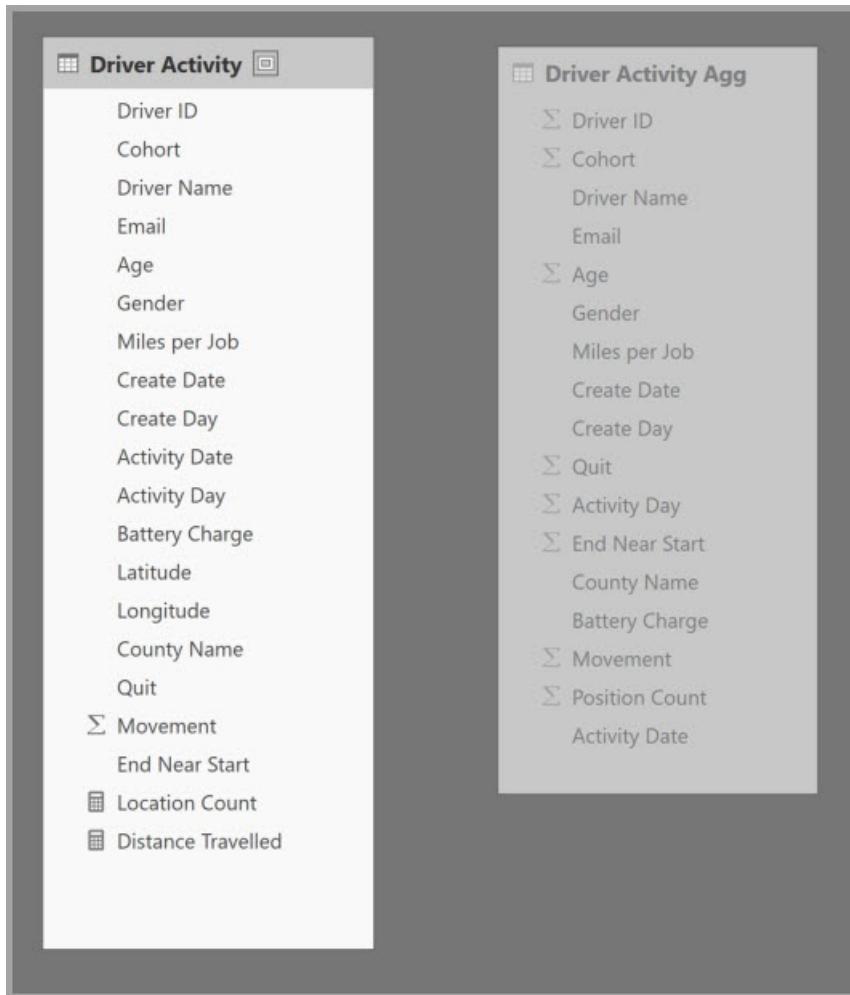
Hadoop-based big data models have different characteristics than dimensional models. To avoid joins between large tables, big data models often don't use relationships, but denormalize dimension attributes to fact tables. You can unlock such big data models for interactive analysis by using *aggregations based on GroupBy columns*.

The following table contains the **Movement** numeric column to be aggregated. All the other columns are attributes to group by. The table contains IoT data and a massive number of rows. The storage mode is DirectQuery. Queries on the data source that aggregate across the whole dataset are slow because of the sheer volume.

Driver Activity

Driver ID
Cohort
Driver Name
Email
Age
Gender
Miles per Job
Create Date
Create Day
Activity Date
Activity Day
Battery Charge
Latitude
Longitude
County Name
Quit
 Σ Movement
End Near Start
 Location Count
 Distance Travelled

To enable interactive analysis on this dataset, you can add an aggregation table that groups by most of the attributes, but excludes the high-cardinality attributes like longitude and latitude. This dramatically reduces the number of rows, and is small enough to comfortably fit into an in-memory cache.



You define the aggregation mappings for the **Driver Activity Agg** table in the [Manage aggregations](#) dialog.

Manage aggregations

Aggregations accelerate query performance to unlock big-data sets. [Learn more](#)

AGGREGATION COLUMN	SUMMARIZATION	DETAIL TABLE	DETAIL COLUMN
Driver ID	GroupBy	Driver Activity	Driver ID
Cohort	GroupBy	Driver Activity	Cohort
Driver Name	GroupBy	Driver Activity	Driver Name
Email	GroupBy	Driver Activity	Email
Age	GroupBy	Driver Activity	Age

[Apply all](#) [Cancel](#)

In aggregations based on GroupBy columns, the **GroupBy** entries aren't optional. Without them, the aggregations won't get hit. This is different from using aggregations based on relationships, where the GroupBy entries are optional.

The following table shows the aggregations for the **Driver Activity Agg** table.

Aggregation Column	Summarization	Detail Table	Detail Column
Driver ID	GroupBy	Driver Activity	Driver ID
Cohort	GroupBy	Driver Activity	Cohort
Driver Name	GroupBy	Driver Activity	Driver Name
Email	GroupBy	Driver Activity	Email
Age	GroupBy	Driver Activity	Age
Gender	GroupBy	Driver Activity	Gender
Jobs per Hour	GroupBy	Driver Activity	Jobs per Hour
Create Date	GroupBy	Driver Activity	Create Date
Create Day	GroupBy	Driver Activity	Create Day
Quit	GroupBy	Driver Activity	Quit
Activity Day	GroupBy	Driver Activity	Activity Day
Activity Date	GroupBy	Driver Activity	Activity Date
End Near Start	GroupBy	Driver Activity	End Near Start
County Name	GroupBy	Driver Activity	County Name
Battery Charge	GroupBy	Driver Activity	Battery Charge
Movement	Sum	Driver Activity	Movement
Position Count	Count table rows	Driver Activity	N/A

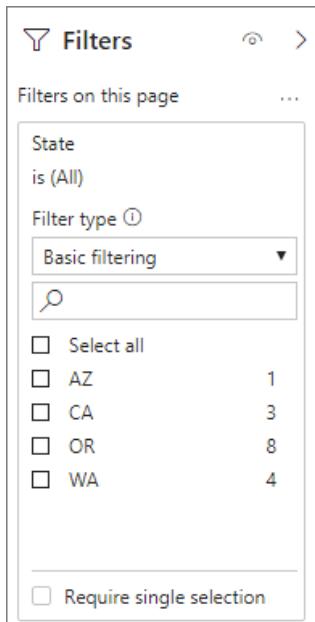
You can set the storage mode of the aggregated **Driver Activity Agg** table to Import.

GroupBy aggregation query example

The following query hits the aggregation, because the **Activity Date** column is covered by the aggregation table. The COUNTROWS function uses the **Count table rows** aggregation.

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Driver Activity'[Activity Date],  
    "Location Count", COUNTROWS('Driver Activity')  
)
```

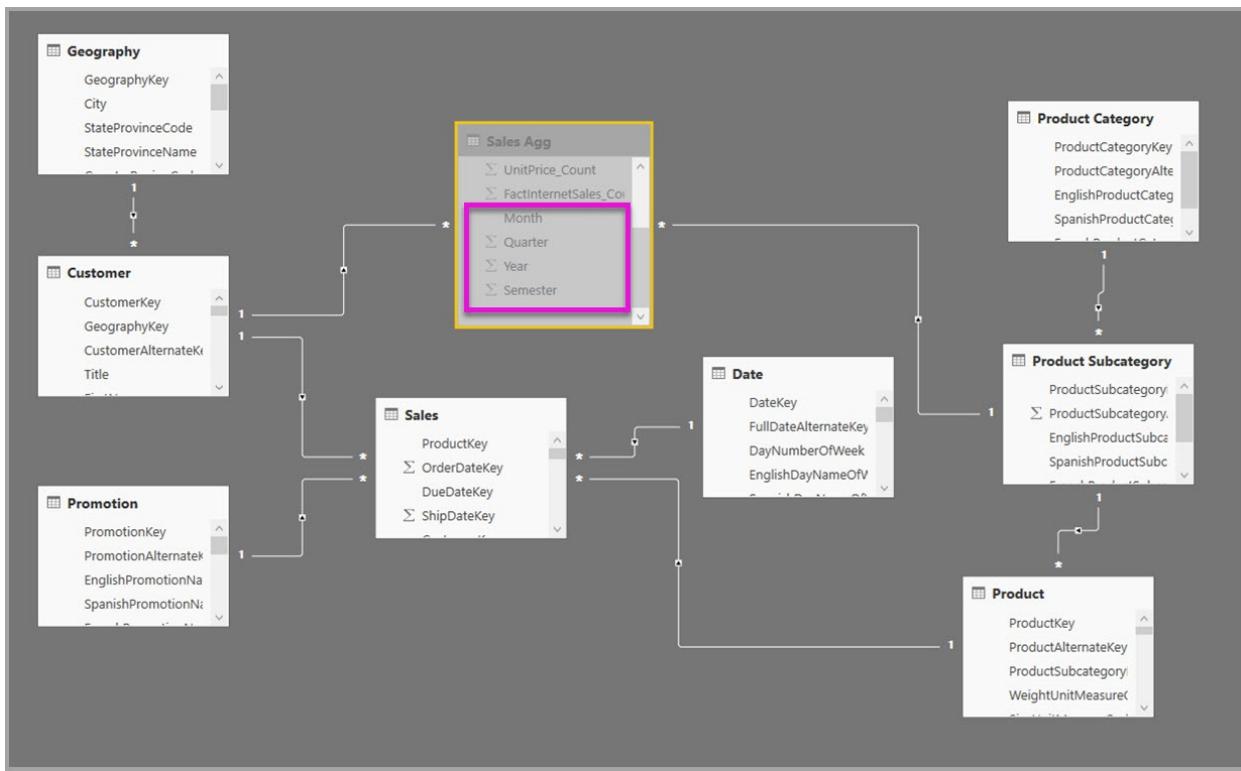
Especially for models that contain filter attributes in fact tables, it's a good idea to use **Count table rows** aggregations. Power BI may submit queries to the dataset using COUNTROWS in cases where it is not explicitly requested by the user. For example, the filter dialog shows the count of rows for each value.



Combined aggregation techniques

You can combine the relationships and GroupBy columns techniques for aggregations. Aggregations based on relationships may require the denormalized dimension tables to be split into multiple tables. If this is costly or impractical for certain dimension tables, you can replicate the necessary attributes in the aggregation table for those dimensions, and use relationships for others.

For example, the following model replicates **Month**, **Quarter**, **Semester**, and **Year** in the **Sales Agg** table. There is no relationship between **Sales Agg** and the **Date** table, but there are relationships to **Customer** and **Product Subcategory**. The storage mode of **Sales Agg** is Import.



The following table shows the entries set in the Manage aggregations dialog for the Sales Agg table. The GroupBy entries where **Date** is the detail table are mandatory, to hit aggregations for queries that group by the Date attributes. As in the previous example, the GroupBy entries for **CustomerKey** and **ProductSubcategoryKey** don't affect aggregation hits, except for DISTINCTCOUNT, because of the presence of relationships.

Aggregation Column	Summarization	Detail Table	Detail Column
Month	GroupBy	Date	CalendarMonth
Quarter	GroupBy	Date	CalendarQuarter
Semester	GroupBy	Date	CalendarSemester
Year	GroupBy	Date	CalendarYear
CustomerKey	GroupBy	Sales	CustomerKey
ProductSubcategoryKey	GroupBy	Product	ProductSubcategoryKey
SalesAmount_Sum	Sum	Sales	SalesAmount
UnitPrice_Sum	Sum	Sales	UnitPrice
UnitPrice_Count	Count	Sales	UnitPrice
FactInternetSales_Count	Count table rows	Sales	N/A

Combined aggregation query examples

The following query hits the aggregation, because the aggregation table covers **CalendarMonth**, and **CategoryName** is accessible via one-to-many relationships. **SalesAmount** uses the **SUM** aggregation.

```
EVALUATE
SUMMARIZECOLUMNS(
    'Date'[CalendarMonth],
    'Product Category'[CategoryName],
    "Sales", SUM('Sales'[SalesAmount])
)
```

The following query doesn't hit the aggregation, because the aggregation table doesn't cover **CalendarDay**.

EVALUATE

```
SUMMARIZECOLUMNS(  
    'Date'[CalendarDay],  
    'Product Category'[CategoryName],  
    "Sales", SUM('Sales'[SalesAmount])  
)
```

The following time-intelligence query doesn't hit the aggregation, because the DATESYTD function generates a table of **CalendarDay** values, and the aggregation table doesn't cover **CalendarDay**.

EVALUATE

```
SUMMARIZECOLUMNS(  
    'Date'[CalendarMonth],  
    'Product Category'[CategoryName],  
    "Sales", CALCULATE(SUM('Sales'[SalesAmount]), DATESYTD('Date'[CalendarDay]))  
)
```

Aggregation precedence

Aggregation precedence allows multiple aggregation tables to be considered by a single subquery.

The following example is a [composite model](#) containing multiple sources:

- The **Driver Activity** DirectQuery table contains over a trillion rows of IoT data sourced from a big-data system. It serves drillthrough queries to view individual IoT readings in controlled filter contexts.
- The **Driver Activity Agg** table is an intermediate aggregation table in DirectQuery mode. It contains over a billion rows in Azure Synapse Analytics (formerly SQL Data Warehouse) and is optimized at the source using columnstore indexes.
- The **Driver Activity Agg2 Import** table is at a high granularity, because the group-by attributes are few and low cardinality. The number of rows could be as low as thousands, so it can easily fit into an in-memory cache. These attributes happen to be used by a high-profile executive dashboard, so queries referring to them should be as fast as possible.

NOTE

DirectQuery aggregation tables that use a different data source from the detail table are only supported if the aggregation table is from a SQL Server, Azure SQL, or Azure Synapse Analytics (formerly SQL Data Warehouse) source.

The memory footprint of this model is relatively small, but it unlocks a huge dataset. It represents a balanced architecture because it spreads the query load across components of the architecture, utilizing them based on their strengths.

The screenshot shows the 'Manage aggregations' dialog with three tables listed:

- Driver Activity** (Leftmost table):
 - Driver ID
 - Cohort
 - Driver Name
 - Email
 - Age
 - Gender
 - Miles per Job
 - Create Date
 - Create Day
 - Activity Date
 - Activity Day
 - Battery Charge
 - Latitude
 - Longitude
 - County Name
 - Quit
 - Σ Movement
 - End Near Start
 - Location Count
 - Distance Travelled
- Driver Activity Agg** (Middle table):
 - Σ Driver ID
 - Σ Cohort
 - Driver Name
 - Email
 - Σ Age
 - Gender
 - Miles per Job
 - Create Date
 - Create Day
 - Σ Quit
 - Σ Activity Day
 - Σ End Near Start
 - County Name
 - Battery Charge
 - Σ Movement
 - Σ Position Count
 - Activity Date
- Driver Activity Agg2** (Rightmost table, highlighted with a yellow border):
 - Σ Quit
 - Σ Activity Day
 - Σ End Near Start
 - Σ Movement
 - Σ Position Count
 - Activity Date
 - Miles per Job

The **Manage aggregations** dialog for **Driver Activity Agg2** sets the **Precedence** field to **10**, which is higher than for **Driver Activity Agg**. The higher precedence setting means queries that use aggregations will consider **Driver Activity Agg2** first. Subqueries that aren't at the granularity that can be answered by **Driver Activity Agg2** will consider **Driver Activity Agg** instead. Detail queries that cannot be answered by either aggregation table will be directed to **Driver Activity**.

The table specified in the **Detail Table** column is **Driver Activity**, not **Driver Activity Agg**, because chained aggregations are not allowed.

Manage aggregations

Aggregations accelerate query performance to unlock big-data sets. [Learn more](#)

AGGREGATION COLUMN	SUMMARIZATION	DETAIL TABLE	DETAIL COLUMN
Driver Activity Agg2 ▾	Precedence ⓘ 10	Driver Activity	Quit
Quit	GroupBy	Driver Activity	Activity Day
Activity Day	GroupBy	Driver Activity	End Near Start
End Near Start	GroupBy	Driver Activity	Movement
Movement	Sum	Driver Activity	Position Count
Position Count	Count table rows	Driver Activity	

Apply all **Cancel**

The following table shows the aggregations for the Driver Activity Agg2 table.

Aggregation Column	Summarization	Detail Table	Detail Column
Jobs per Hour	GroupBy	Driver Activity	Jobs per Hour
Quit	GroupBy	Driver Activity	Quit
Activity Day	GroupBy	Driver Activity	Activity Day
Activity Date	GroupBy	Driver Activity	Activity Date
End Near Start	GroupBy	Driver Activity	End Near Start
Movement	Sum	Driver Activity	Movement
Position Count	Count table rows	Driver Activity	N/A

Detect whether queries hit or miss aggregations

SQL Profiler can detect whether queries are returned from the in-memory cache storage engine, or pushed to the data source by DirectQuery. You can use the same process to detect whether aggregations are being hit. For more information, see [Queries that hit or miss the cache](#).

SQL Profiler also provides the `Query Processing\Aggregate Table Rewrite Query` extended event.

The following JSON snippet shows an example of the output of the event when an aggregation is used.

- **matchingResult** shows that the subquery used an aggregation.
- **dataRequest** shows the GroupBy column(s) and aggregated column(s) the subquery used.
- **mapping** shows the columns in the aggregation table that were mapped to.

```
{  
  "table": "Sales",  
  "mapping": {  
    "table": "Sales Agg"  
  },  
  "matchingResult": "matchFound",  
  "dataRequest": [  
    {  
      "table": "Date",  
      "column": "CalendarYear",  
      "mapping": {  
        "table": "Date",  
        "column": "CalendarYear"  
      }  
    },  
    {  
      "aggregation": "sum",  
      "table": "Sales",  
      "column": "SalesAmount",  
      "mapping": {  
        "column": "SalesAmount_Sum"  
      }  
    }  
  ]  
}
```

Keep caches in sync

Aggregations that combine DirectQuery, Import, and/or Dual storage modes may return different data unless the in-memory cache is kept in sync with the source data. For example, query execution won't attempt to mask data issues by filtering DirectQuery results to match cached values. There are established techniques to handle such issues at the source, if necessary. Performance optimizations should be used only in ways that don't compromise your ability to meet business requirements. It's your responsibility to know your data flows and design accordingly.

Community

Power BI has a vibrant community where MVPs, BI pros, and peers share expertise in discussion groups, videos, blogs and more. When learning about aggregations, be sure to check out these additional resources:

- [Power BI Community](#)
- [Search "Power BI aggregations" on Bing](#)

See also

[Automatic aggregations](#)

[Composite models](#)

Manage storage mode in Power BI Desktop

3/30/2022 • 8 minutes to read • [Edit Online](#)

In Microsoft Power BI Desktop, you can specify the storage mode of a table. The storage mode lets you control whether Power BI Desktop caches table data in-memory for reports.

Setting the storage mode provides many advantages. You can set the storage mode for each table individually in your model. This action enables a single dataset, which provides the following benefits:

- **Query performance:** As users interact with visuals in Power BI reports, Data Analysis Expressions (DAX) queries are submitted to the dataset. Caching data into memory by properly setting the storage mode can boost the query performance and interactivity of your reports.
- **Large datasets:** Tables that aren't cached don't consume memory for caching purposes. You can enable interactive analysis over large datasets that are too large or expensive to completely cache into memory. You can choose which tables are worth caching, and which aren't.
- **Data refresh optimization:** You don't need to refresh tables that aren't cached. You can reduce refresh times by caching only the data that's necessary to meet your service level agreements and your business requirements.
- **Near-real time requirements:** Tables with near-real time requirements might benefit from not being cached, to reduce data latency.
- **Writeback:** Writeback enables business users to explore what-if scenarios by changing cell values. Custom applications can apply changes to the data source. Tables that aren't cached can display changes immediately, which allows instant analysis of the effects.

The storage mode setting in Power BI Desktop is one of three related features:

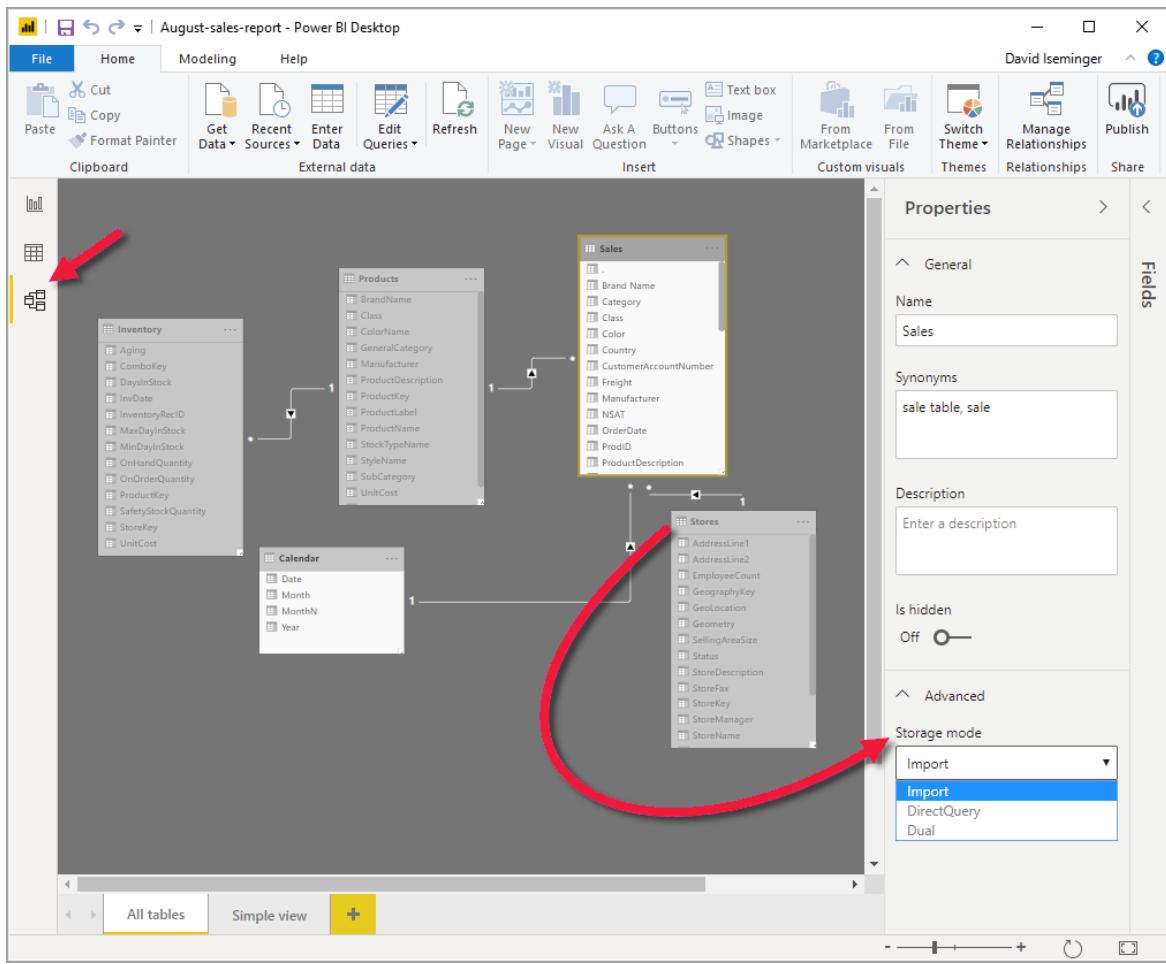
- **Composite models:** Allows a report to have two or more data connections, including DirectQuery connections or Import, in any combination. For more information, see [Use composite models in Power BI Desktop](#).
- **Many-to-many relationships:** With composite models, you can establish *many-to-many relationships* between tables. In a many-to-many relationship, requirements are removed for unique values in tables. It also removes prior workarounds, such as introducing new tables only to establish relationships. For more information, see [Many-to-many relationships in Power BI Desktop](#).
- **Storage mode:** With storage mode, you can now specify which visuals require a query to back-end data sources. Visuals that don't require a query are imported even if they're based on DirectQuery. This feature helps improve performance and reduce back-end load. Previously, even simple visuals, such as slicers, initiated queries that were sent to back-end sources.

Use the Storage mode property

The **Storage mode** property is a property that you can set on each table in your model and controls how Power BI caches the table data.

To set the **Storage mode** property, or view its current setting:

1. in **Model** view, select the table whose properties you want to view or set.
2. In the **Properties** pane, expand the **Advanced** section, and expand the **Storage mode** drop-down.



You set the **Storage mode** property to one of these three values:

- **Import:** Imported tables with this setting are cached. Queries submitted to the Power BI dataset that return data from Import tables can be fulfilled only from cached data.
- **DirectQuery:** Tables with this setting aren't cached. Queries that you submit to the Power BI dataset—for example, DAX queries—and that return data from DirectQuery tables can be fulfilled only by executing on-demand queries to the data source. Queries that you submit to the data source use the query language for that data source, for example, SQL.
- **Dual:** Tables with this setting can act as either cached or not cached, depending on the context of the query that's submitted to the Power BI dataset. In some cases, you fulfill queries from cached data. In other cases, you fulfill queries by executing an on-demand query to the data source.

Changing the **Storage mode** of a table to **Import** is an *irreversible* operation. Once set, this property can't later be changed to either **DirectQuery** or **Dual**.

NOTE

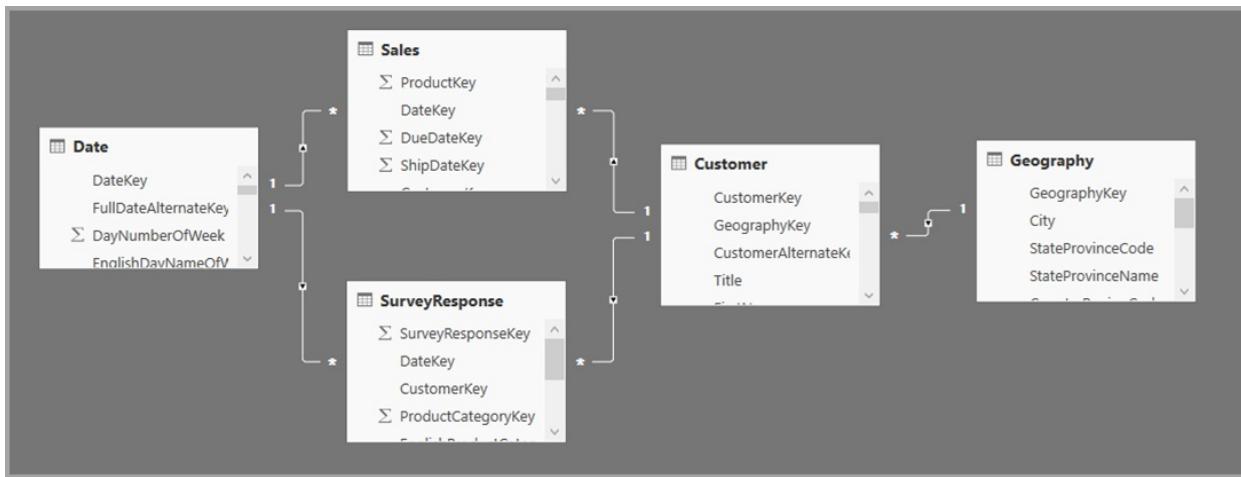
You can use **Dual** storage mode in both Power BI Desktop and the Power BI service.

Constraints on DirectQuery and Dual tables

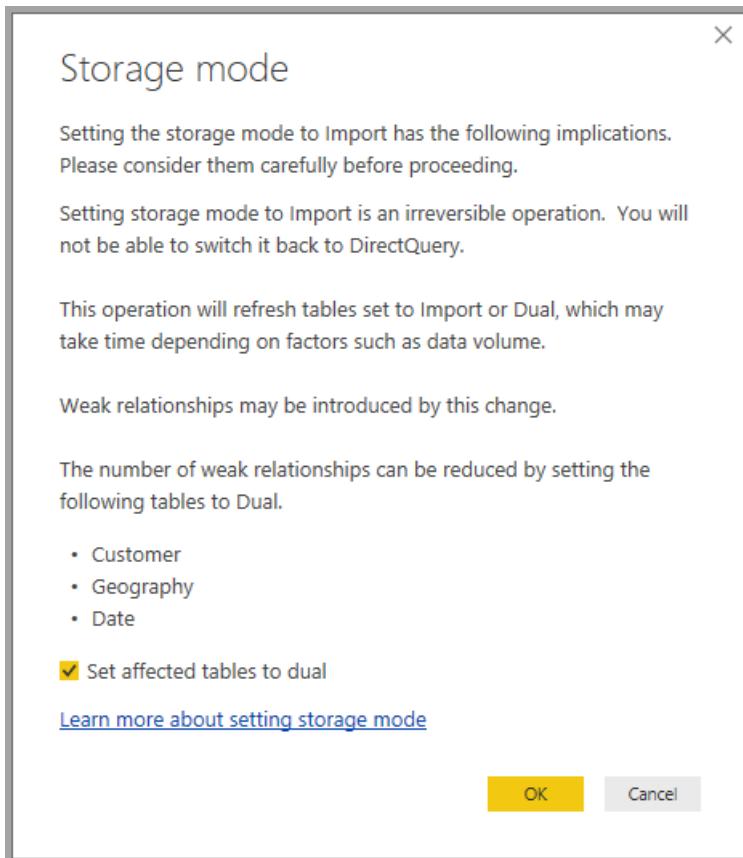
Dual tables have the same functional constraints as DirectQuery tables. These constraints include limited M transformations and restricted DAX functions in calculated columns. For more information, see [Implications of using DirectQuery](#).

Propagation of the Dual setting

Consider the following simple model, where all the tables are from a single source that supports Import and DirectQuery.



Let's say all tables in this model are initially set to **DirectQuery**. If you then change the **Storage mode** of the **SurveyResponse** table to **Import**, the following warning window is displayed:



You can set the dimension tables (**Customer**, **Geography**, and **Date**) to **Dual** to reduce the number of limited relationships in the dataset, and improve performance. Limited relationships normally involve at least one DirectQuery table where join logic can't be pushed to the source systems. Because Dual tables can act as either DirectQuery or Import tables, this situation is avoided.

The propagation logic is designed to help with models that contain many tables. Suppose you have a model with 50 tables and only certain fact (transactional) tables need to be cached. The logic in Power BI Desktop calculates the minimum set of dimension tables that must be set to **Dual**, so you don't have to.

The propagation logic traverses only to the one side of one-to-many relationships.

Storage mode usage example

Let's continue with the example from the previous section, and imagine applying the following storage mode property settings:

TABLE	STORAGE MODE
Sales	DirectQuery
SurveyResponse	Import
Date	Dual
Customer	Dual
Geography	Dual

Setting these storage mode properties results in the following behaviors, assuming that the **Sales** table has significant data volume:

- Power BI Desktop caches dimension tables, **Date**, **Customer**, and **Geography**, so load times of initial reports are fast when they retrieve slicer values to display.
- Power BI Desktop doesn't cache the **Sales** table. By not caching this table, Power BI Desktop provides the following results:
 - Data-refresh times are improved, and memory consumption is reduced.
 - Report queries that are based on the **Sales** table run in **DirectQuery** mode. These queries might take longer but are closer to real time because no caching latency is introduced.
- Report queries that are based on the **SurveyResponse** table are returned from the in-memory cache, and are therefore relatively fast.

Queries that hit or miss the cache

If you connect SQL Profiler to the diagnostics port for Power BI Desktop, you can see which queries hit or miss the in-memory cache by performing a trace that's based on the following events:

- Queries Events\Query Begin
- Query Processing\Vertipaq SE Query Begin
- Query Processing\DirectQuery Begin

For each *Query Begin* event, check other events with the same *ActivityID*. For example, if there isn't a *DirectQuery Begin* event, but there's a *Vertipaq SE Query Begin* event, the query is answered from the cache.

Queries that refer to Dual tables return data from the cache, if possible; otherwise, they revert to DirectQuery.

Continuing the previous example, the following query refers only to a column from the **Date** table, which is in **Dual** mode. Therefore, the query should hit the cache:

```
EVALUATE  
VALUES('Date'[CalendarYear])
```

The following query refers only to a column from the **Sales** table, which is in **DirectQuery** mode. Therefore, it should *not* hit the cache:

```
EVALUATE  
ROW("Sales", SUM('Sales'[SalesAmount]))
```

The following query is interesting because it combines both columns. This query doesn't hit the cache. You might initially expect it to retrieve **CalendarYear** values from the cache and **SalesAmount** values from the source and then combine the results, but this approach is less efficient than submitting the SUM/GROUP BY operation to the source system. If the operation is pushed down to the source, the number of rows returned will likely be far less:

```
EVALUATE  
SUMMARIZECOLUMNS(  
    'Date'[CalendarYear],  
    "Sales", SUM('Sales'[SalesAmount])  
)
```

NOTE

This behavior is different from [many-to-many relationships](#) in Power BI Desktop when cached and non-cached tables are combined.

Caches should be kept in sync

The queries displayed in the previous section show that Dual tables sometimes hit the cache and sometimes don't. As a result, if the cache is out of date, different values can be returned. Query execution won't attempt to mask data issues by, for example, filtering DirectQuery results to match cached values. It's your responsibility to know your data flows, and you should design accordingly. There are established techniques to handle such cases at the source, if necessary.

The **Dual** storage mode is a performance optimization. It should be used only in ways that don't compromise the ability to meet business requirements. For alternative behavior, consider using the techniques described in the [Many-to-many relationships in Power BI Desktop](#).

Data view

If at least one table in the dataset has its storage mode set to either **Import** or **Dual**, the **Data view** tab is displayable.

SalesOrderID	Country	OrderDate	SalesChannelCode	ProdID	Style
SO446221	Germany	12/1/2011	I	752	Pro
SO437511	France	2/21/2012	I	751	Pro
SO449681	Germany	3/9/2012	I	744	Pro
SO444371	Australia	3/24/2012	I	755	Pro
SO446681	United States	3/28/2012	I	743	Pro
SO446521	France	5/24/2012	I	754	Pro
SO441431	United States	6/24/2012	I	759	Pro
SO445951	Australia	7/1/2012	I	743	Pro
SO442081	United States	7/18/2012	I	760	Pro
SO437161	Australia	7/30/2012	I	744	Pro
SO437481	United States	8/20/2012	I	755	Pro
SO437711	United States	8/30/2012	I	748	Pro
SO437301	United States	9/11/2012	I	761	Pro
SO449811	Australia	9/13/2012	I	755	Pro
SO443961	United States	9/28/2012	I	748	Pro
SO448471	Australia	10/9/2012	I	766	Pro
SO446881	Australia	10/13/2012	I	743	Pro

When you select Dual and Import tables in Data view, they show cached data. DirectQuery tables don't show data, and a message is displayed that states that DirectQuery tables can't be shown.

Considerations and limitations

There are a few limitations for this release of storage mode and its correlation with composite models.

The following live connection (multi-dimensional) sources can't be used with composite models:

- SAP HANA
- SAP Business Warehouse
- SQL Server Analysis Services
- Power BI datasets
- Azure Analysis Services

When you connect to those multi-dimensional sources by using DirectQuery, you can't connect to another DirectQuery source or combine it with imported data.

The existing limitations of using DirectQuery still apply when you use composite models. Many of those limitations are now per table, depending upon the storage mode of the table. For example, a calculated column on an imported table can refer to other tables, but a calculated column on a DirectQuery table is still restricted to refer only to columns on the same table. Other limitations apply to the model as a whole, if any of the tables within the model are DirectQuery. For example, the QuickInsights and Q&A features aren't available on a model if any of the tables within it has a storage mode of DirectQuery.

Next steps

For more information about composite models and DirectQuery, see the following articles:

- [Composite models in Power BI Desktop](#)
- [Many-to-many relationships in Power BI Desktop](#)
- [Use DirectQuery in Power BI](#)
- [Data sources supported by DirectQuery in Power BI](#)

Work with multidimensional models in Power BI

3/30/2022 • 4 minutes to read • [Edit Online](#)

You can connect to multidimensional models in Power BI, and create reports that visualize all sorts of data within the model. When working with multidimensional models, Power BI applies rules to how it processes data, based on which column is defined as the *default member*.

When working with multidimensional models, Power BI handles data from the model based on where the column that contains the **DefaultMember** is used. The *DefaultMember* attribute is set in CSDL (Conceptual Schema Definition Language) for a particular column in a multidimensional model. You can learn more about the default member in its [attribute properties article](#). When a DAX query is executed, the default member specified in the model is applied automatically.

This article described how Power BI behaves under various circumstances when working with multidimensional models, based on where the *default member* is found.

Working with filter cards

When creating a filter card on a field with a default member, the default member field value is selected automatically in the filter card. The result is that all visuals that are affected by the filter card retain their default models in the database. The values in such filter cards reflect that default member.

If the default member is removed, de-selecting the value clears it for all visuals to which the filter card applies, and the values displayed do not reflect the default member.

For example, imagine we have a *Currency* column that has a default member set to *USD*.

- In this example case, if we have a card that shows *Total Sales*, the value will have the default member applied and we see sales that correspond to "USD".
- If we drag *Currency* to the filter card pane, we see *USD* as the default value selected. The value of *Total Sales* remains the same, since the default member is applied.
- However, if we deselect the *USD* value from the filter card, the default member for *Currency* is cleared, and now *Total Sales* reflects all currencies.
- Consequently, when we select another value in the filter card (let's say we select *Euro*), along the default member, the *Total Sales* reflects the filter *Currency IN {USD, Euro}*.

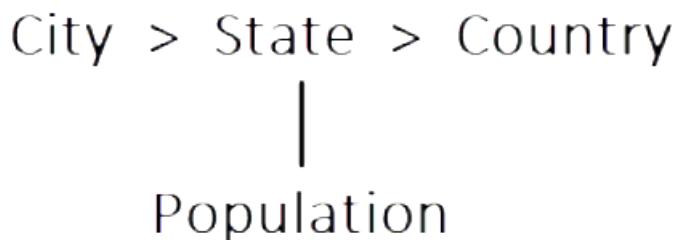
Grouping behavior

In Power BI, whenever you group a visual on a column that has a *default member*, Power BI clears the *default member* for that column and its attribute relationship path. This ensures the visual displays all values, rather than just the default values.

Attribute relationship paths (ARPs)

Attribute relationship paths (ARPs) provide *default members* with powerful capabilities, but also introduce a certain amount of complexity. When ARPs are encountered, Power BI follows the path of ARPs to clear additional default members for other columns, to provide consistent, and precise handling of data for visuals.

Let's look at an example to clarify the behavior. Consider the following configuration of ARPs:



Now let's imagine the following *default members* are set for these columns:

- City > Seattle
- State > WA
- Country > US
- Population > Large

Now let's examine what happens when each of the columns is used in Power BI. When visuals group on the following columns, here are the results:

- **City** - Power BI displays all the cities by clearing all the **default members** for *City, State, Country* but preserves the **default member** for *Population*; Power BI cleared the entire ARP for *City*.

NOTE

Population is not in the ARP path of *City*, it is solely related to *State* and thus Power BI doesn't clear it.

- **State** - Power BI displays all the *States* by clearing all **default members** for *City, State, Country* and *Population*.
- **Country** - Power BI displays all the countries by clearing all **default members** for *City, State* and *Country*, but preserves the **default member** for *Population*.
- **City and State** - Power BI clears all **default members** for all columns.

Groups displayed in the visual have their entire ARP path cleared.

If a group is not displayed in the visual, but is part of the ARP path of another grouped-on column, the following applies:

- Not all branches of the ARP path are cleared automatically.
- That group is still filtered by that uncleared **default member**.

Slicers and filter cards

When working with slicers or filter cards, the following behavior occurs:

- When a slicer or filter card is loaded with data, Power BI groups on the column in the visual, so the display behavior is the same as described in the previous section.

Since slicers and filter cards are often used to interact with other visuals, the logic of clearing **default members** for the affected visuals occurs as explained in the following table.

For this table, we use the same example data used earlier in this article:

Visual with groups	Filter card selection	Expected result in the visual
City	City = Default Member (Seattle)	Only the DM value of City shows up
City	City = Portland	Only Portland shows up
City	City - All	All cities, but from states that have Large population (due to the DM on Population)
City	City - All, Population - All	All cities
State	City = Default Member (Seattle)	WA, due to the default of Seattle
State	City - All	All states
State	Population - Default Member (Large)	Only states with large population
State	City = Default Member (Seattle) Country - All	WA In this case the clearing of the whole ARP path for Country (Country - State - City) due to Country - All But we will keep only City = Default The end result will be: All states, All countries but Cities = Seattle
State	Country - All Population - Default Member (Large)	Clear Country, State, City + Keep Population DM States that have the default population

The following rules apply for how Power BI behaves in these circumstances.

Power BI clears a **default member** for a given column if:

- Power BI groups on that column
- Power BI groups on a column related to that column (anywhere in the ARP, up or down)
- Power BI filters on a column that is in the ARP (up or down)
- The column has a filter card with *ALL* state
- The column has a filter card with any value selected (Power BI receives a filter for the column)

Power BI does not clear a **default member** for a given column if:

- The column has a filter card with default state, and Power BI is groupings on a column in its ARP.
- The column is above another column in the ARP, and Power BI has a filter card for that other column in default state.

Next steps

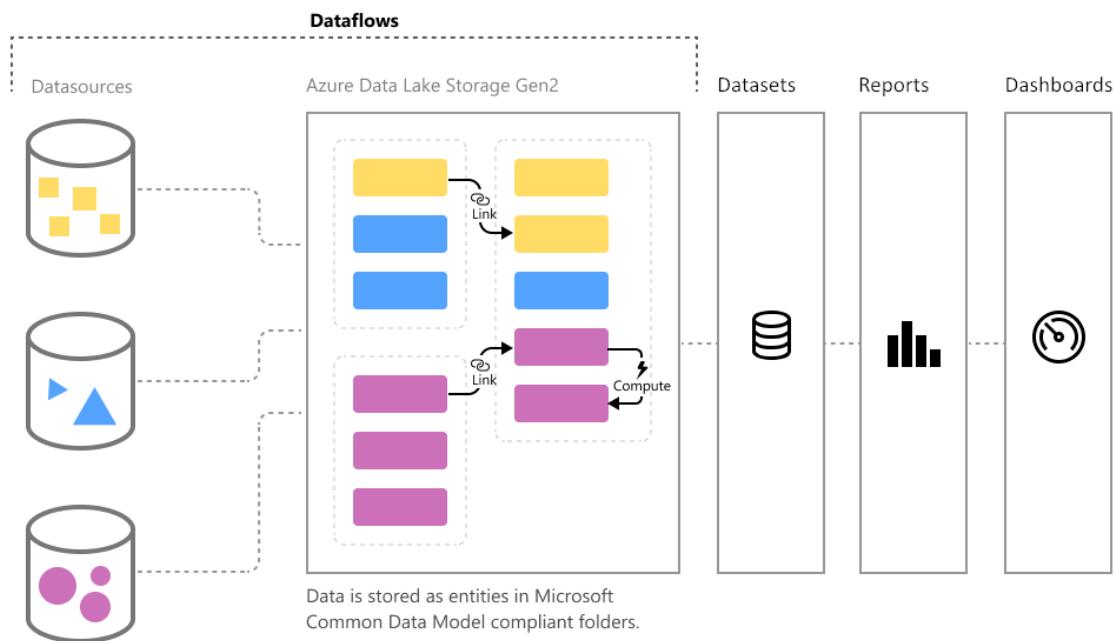
This article described the behavior of Power BI when working with default members in multidimensional models. You might also be interested in the following articles:

- [Show items with no data in Power BI](#)
- [Data sources in Power BI Desktop](#)

Introduction to dataflows and self-service data prep

3/30/2022 • 2 minutes to read • [Edit Online](#)

As data volume continues to grow, so does the challenge of wrangling that data into well-formed, actionable information. We want data that's ready for analytics, to populate visuals, reports, and dashboards, so we can quickly turn our volumes of data into actionable insights. With self-service data prep for big data in Power BI, you can go from data to Power BI insights with just a few clicks.



When to use dataflows

Dataflows are designed to support the following scenarios:

- Create reusable transformation logic that can be shared by many datasets and reports inside Power BI. Dataflows promote reusability of the underlying data elements, preventing the need to create separate connections with your cloud or on-premise data sources.
- Expose the data in your own Azure Data Lake Gen 2 storage, enabling you to connect other Azure services to the raw underlying data.
- Create a single source of the truth by forcing analysts to connect to the dataflows, rather than connecting to the underlying systems, providing you with control over which data is accessed, and how data is exposed to report creators. You can also map the data to industry standard definitions, enabling you to create tidy curated views, which can work with other services and products in the Power Platform.
- If you want to work with large data volumes and perform ETL at scale, dataflows with Power BI Premium scales more efficiently and gives you more flexibility. Dataflows supports a wide range of cloud and on-premise sources.
- Prevent analysts from having direct access to the underlying data source. Since report creators can build on top of dataflows, it may be more convenient for you to allow access to underlying data sources only to a few individuals, and then provide access to the dataflows for analysts to build on top of. This approach

reduces the load to the underlying systems, and gives administrators finer control of when the systems get loaded from refreshes.

Once you've created a dataflow, you can use Power BI Desktop and the Power BI service to create datasets, reports, dashboards, and apps that leverage the Common Data Model to drive deep insights into your business activities. Dataflow refresh scheduling is managed directly from the workspace in which your dataflow was created, just like your datasets.

Next steps

This article provided an overview of self-service data prep for big data in Power BI, and the many ways you can use it.

The following articles provide more information about dataflows and Power BI:

- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [Premium features of dataflows](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

For more information about the Common Data Model, you can read its overview article:

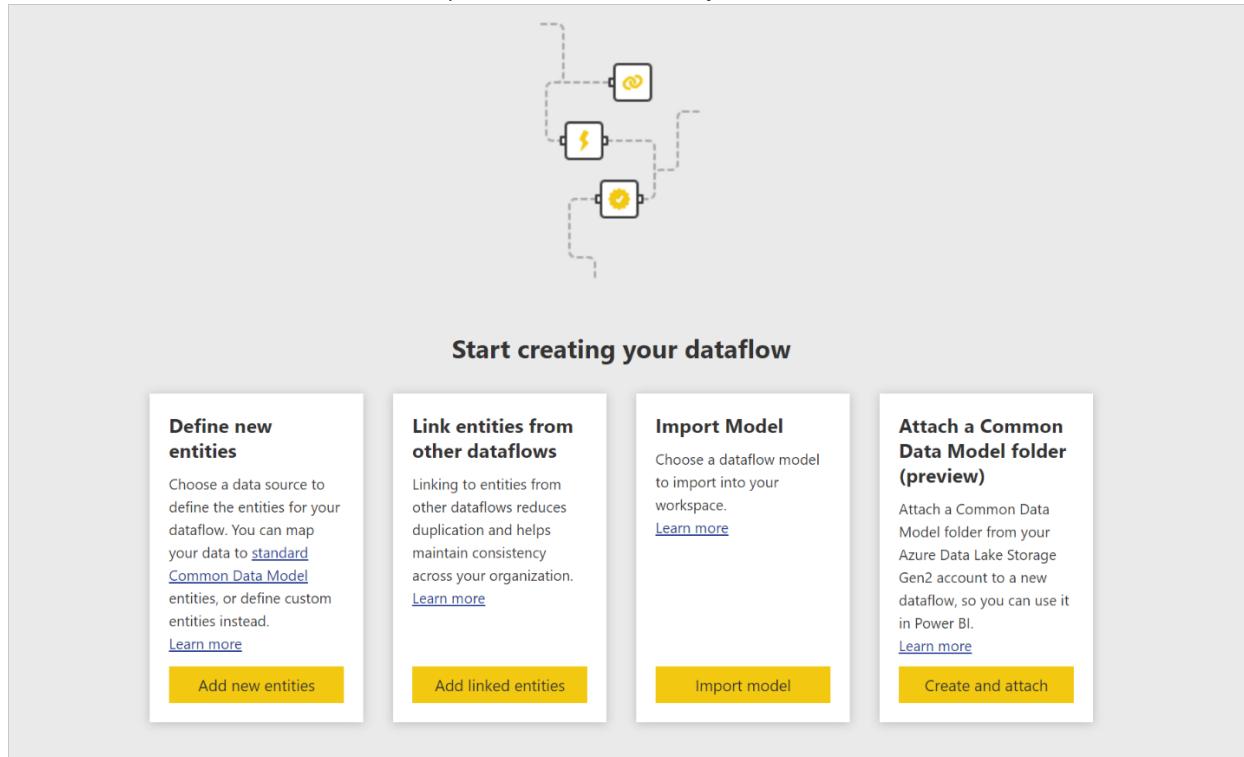
- [Common Data Model - overview](#)

Creating a dataflow

3/30/2022 • 6 minutes to read • [Edit Online](#)

A **dataflow** is a collection of tables that are created and managed in workspaces in the Power BI service. A **table** is a set of columns that are used to store data, much like a table within a database. You can add and edit tables in your dataflow, as well as manage data refresh schedules, directly from the workspace in which your dataflow was created.

To create a dataflow, launch the Power BI service in a browser then select a **workspace** (dataflows are not available in *my-workspace* in the Power BI service) from the nav pane on the left, as shown in the following screen. You can also create a new workspace in which to create your new dataflow.



The screenshot shows the 'Start creating your dataflow' screen in the Power BI service. It features four main options:

- Define new entities**: Choose a data source to define the entities for your dataflow. You can map your data to [standard Common Data Model](#) entities, or define custom entities instead. [Learn more](#). Button: Add new entities.
- Link entities from other dataflows**: Linking to entities from other dataflows reduces duplication and helps maintain consistency across your organization. [Learn more](#). Button: Add linked entities.
- Import Model**: Choose a dataflow model to import into your workspace. [Learn more](#). Button: Import model.
- Attach a Common Data Model folder (preview)**: Attach a Common Data Model folder from your Azure Data Lake Storage Gen2 account to a new dataflow, so you can use it in Power BI. [Learn more](#). Button: Create and attach.

There are multiple ways to create or build on top of a new dataflow:

- [Create a dataflow using define new tables](#)
- [Create a dataflow using linked tables](#)
- [Create a dataflow using a computed table](#)
- [Create a dataflow using import/export](#)

The following sections explore each of these ways to create a dataflow in detail.

NOTE

Dataflows can be created by user in a Premium workspace, users with a Pro license, and users with a Premium Per User (PPU) license.

Create a dataflow using define new tables

Using the Define new tables option lets you define a new table and connect to a new data source.

All categories File Database Power Platform Azure Online services Other

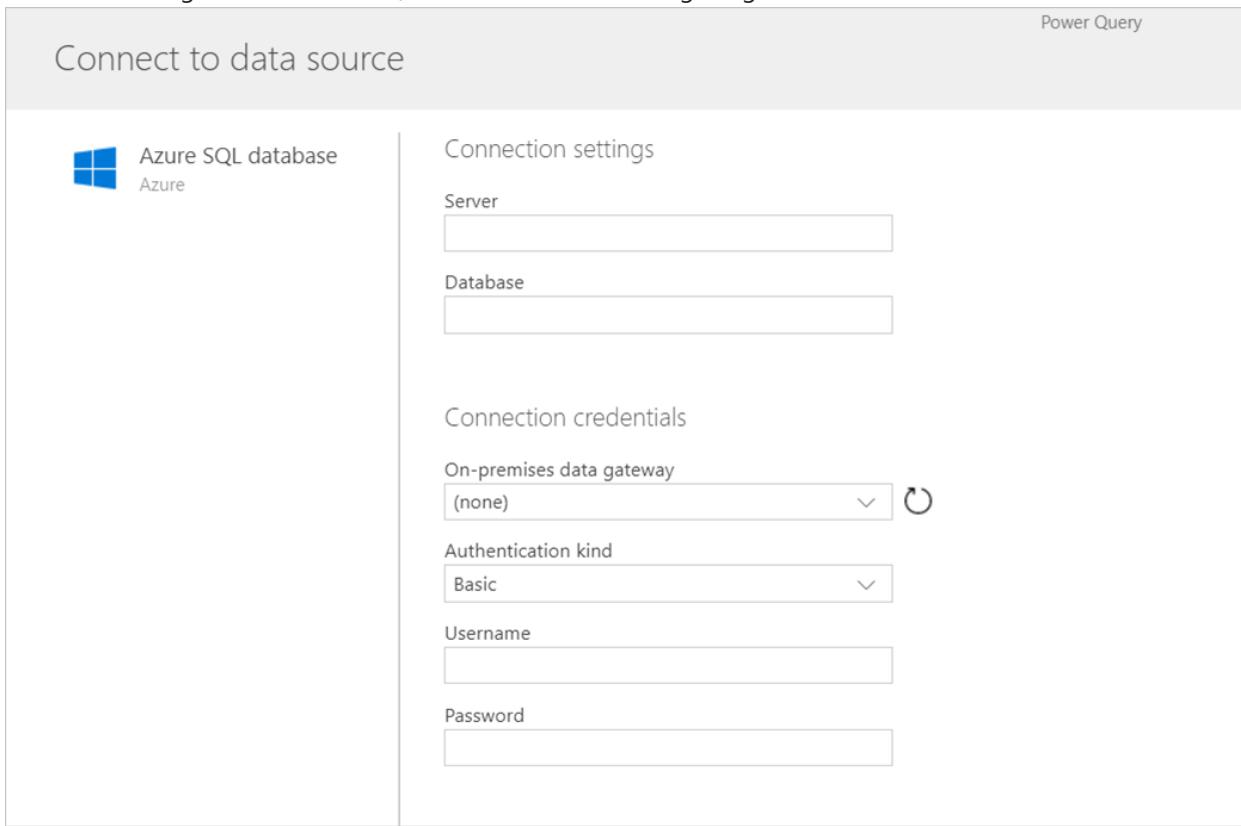
Data sources

Common Data Service Power Platform	Excel File	Folder File	JSON File	PDF File	SharePoint folder File	Text/CSV File
XML File	Access Database	Amazon Redshift Database	Google BigQuery Database	IBM DB2 database Database	Impala Database	MySQL database Database
Oracle database Database	PostgreSQL database Database	Power BI dataflows Power Platform	Power Platform dataflows Power Platform	SAP BW Application Server Database	SAP BW Message Server Database	SAP HANA database Database
SQL Server database Database	Sybase database Database	Teradata database Database	Vertica Database	Azure Blobs Azure	Azure Data Explorer (Kusto) Azure	Azure Data Lake Storage Gen2 Azure
Azure HDInsight Spark Azure	Azure SQL database Azure	Azure Synapse Analytics (SQL DW) Azure	Azure Tables Azure	Microsoft Exchange Online Online services	Salesforce objects Online services	Salesforce reports Online services
SharePoint Online list Online services	Smartsheet services	Active Directory Other	OData Other	OdBC Other	SharePoint list Other	Spark Other
Web API Other	Web page Other	Blank table Other	Blank query Other			

Templates

Accounts, leads, opportunities Salesforce Online services	Lead to cash Dynamics 365 Sales Online services	Leads, opportunities Dynamics 365 Sales Online services	Quotes, orders, invoices Dynamics 365 Sales Online services
---	---	---	---

When you select a data source, you're prompted to provide the connection settings, including the account to use when connecting to the data source, as shown in the following image.



Once connected, you can select which data to use for your table. When you choose data and a source, Power BI reconnects to the data source in order to keep the data in your dataflow refreshed, at the frequency you select later in the setup process.

The screenshot shows the 'Choose Data' dialog in Power Query. On the left, there's a tree view of data sources and tables. A yellow box highlights the 'pbist_twitter.mention_slicer' table under the 'pbist_twitter' node. The main area displays the contents of this table:

tweetid	facet	pbist_twitter.tweets_pr...
927141087180935168	MSPowerBI	[Record]
927118921479495680	MSPowerBI	[Record]
927086430395908096	MSPowerBI	[Record]
927029898538921984	DanMoorehead	[Record]
927029898538921984	willenfarm	[Record]
927029898538921984	FarmAtHand	[Record]
927029898538921984	PowerApps	[Record]
927029898538921984	SharePoint	[Record]
927029898538921984	SQLServer	[Record]
927029898538921984	MicrosoftFlow	[Record]
927029898538921984	Office365	[Record]
927029898538921984	MSPowerBI	[Record]
926997149484765184	MSPowerBI	[Record]
926977213794316288	MSPowerBI	[Record]
926977126535921665	MSPowerBI	[Record]
926950890795237376	PowerAccessSQL	[Record]
926950890795237376	PowerBI	[Record]
926950890795237376	MSPowerBI	[Record]
926950890795237376	JamesMPhillips	[Record]
926950890795237376	AmirNetz	[Record]
926950890795237376	mllopoulos	[Record]
926950890795237376	marcreguera	[Record]
926950890795237376	powerpivotpro	[Record]
926950890795237376	Will_MI77	[Record]

At the bottom, there are 'Back', 'Cancel', and 'Next' buttons.

Once you select the data for use in the table, you can use dataflow editor to shape or transform that data into the format necessary for use in your dataflow.

Create a dataflow using linked tables

Creating a dataflow using linked tables enables you to reference an existing table, defined in another dataflow, in a read-only fashion. The following list describes some of the reasons you may choose this approach:

- If you want to reuse a table across multiple dataflows, such as a date table or a static lookup table, you should create a table once and then reference it across the other dataflows.
- If you want to avoid creating multiple refreshes to a data source, it's better to use linked tables to store the data and act as a cache. Doing so allows every subsequent consumer to leverage that table, reducing the load to the underlying data source.
- If you need to perform a merge between two tables.

NOTE

Linked tables are available only with Power BI Premium.

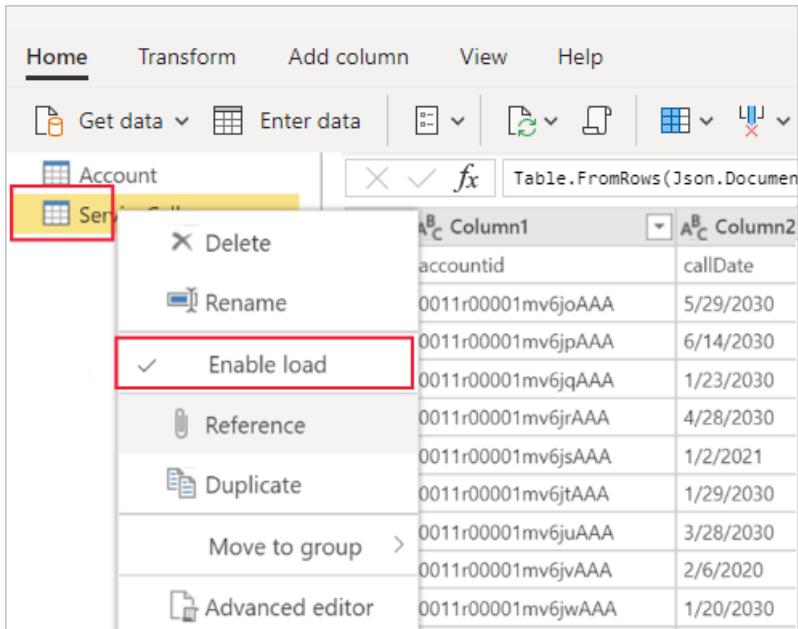
Create a dataflow using a computed table

Creating a dataflow using a computed table allows you to reference a linked table and perform operations on top of it in a write-only fashion. The result is a new table, which is part of the dataflow. To convert a linked table into a computed table, you can either create a new query from a merge operation, or if you want to edit or transform the table, create a reference or duplicate of the table.

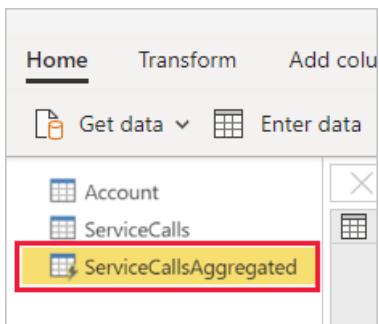
How to create computed tables

Once you have a dataflow with a list of tables, you can perform calculations on those tables. In the dataflow authoring tool in the Power BI service, select **Edit tables**, then right-click on the table you want to use as the basis for your computed table and on which you want to perform calculations. In the context menu, choose

Reference. For the table to be eligible as a computed table, the **Enable load** selection must be checked, as shown in the following image. Right-click on the table to display this context menu.



By selecting **Enable load**, you create a new table for which its source is the referenced table. The icon changes, and shows the **computed** icon, as shown in the following image.

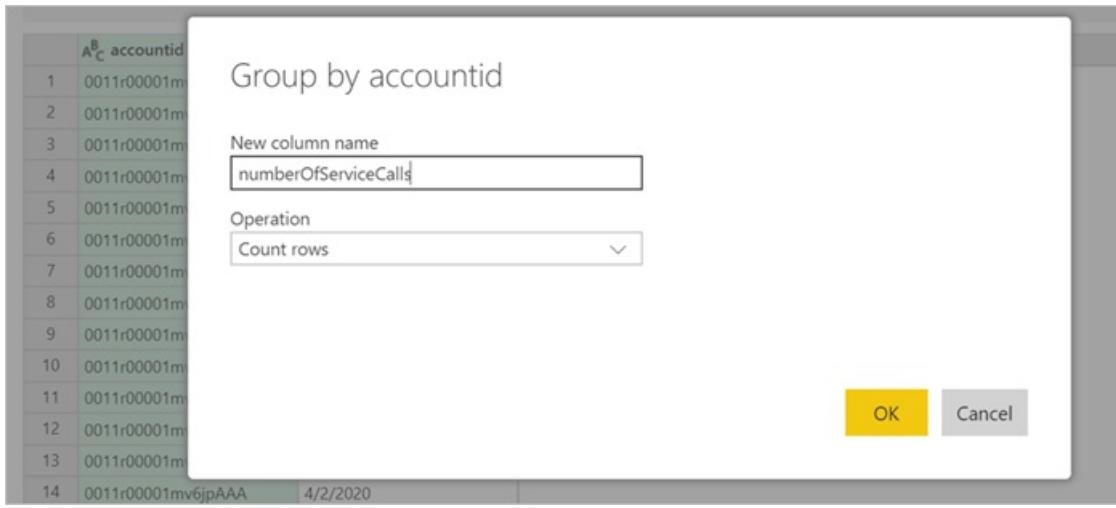


Any transformation you perform on this newly created table is run on the data that already resides in Power BI dataflow storage. That means that the query will not run against the external data source from which the data was imported (for example, the SQL database from which the data was pulled), but rather, is performed on the data that resides in the dataflow storage.

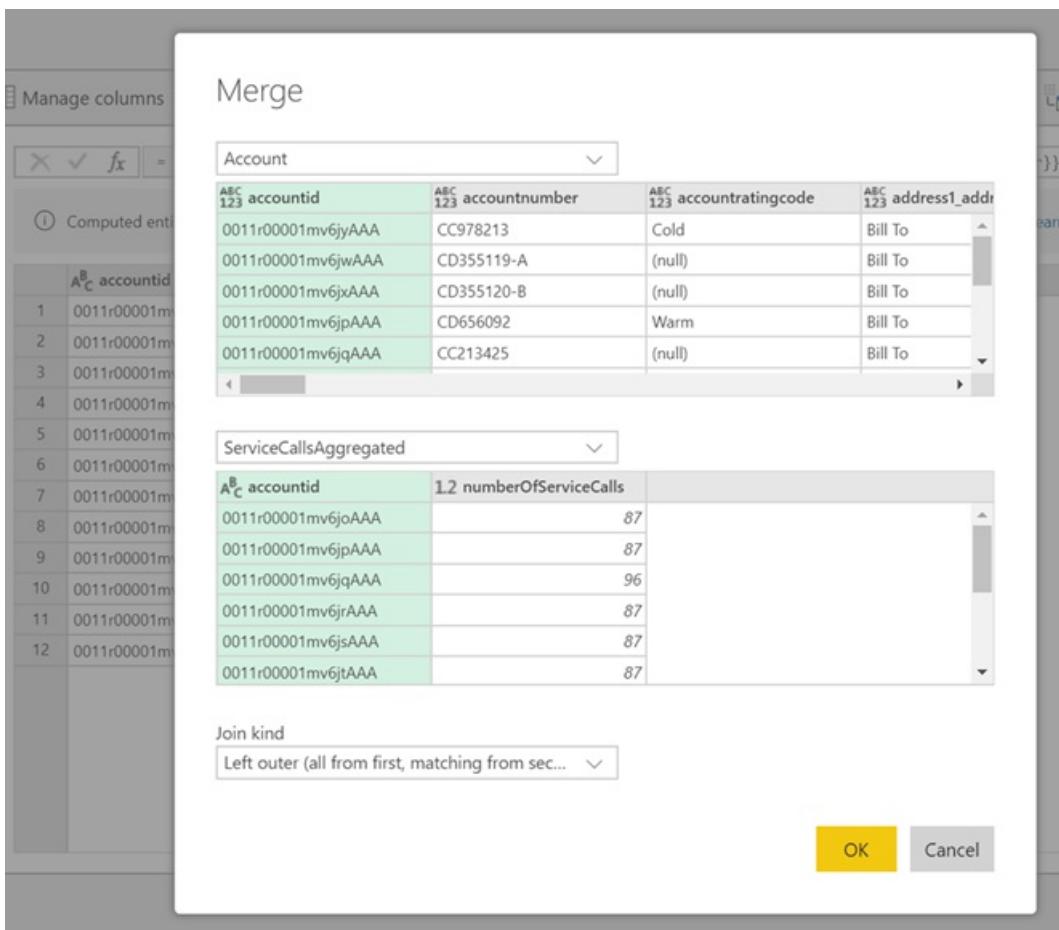
Example use cases What kind of transformations can be performed with computed tables? Any transformation that you usually specify using the transformation user interface in Power BI, or the M editor, are all supported when performing in-storage computation.

Consider the following example: you have an *Account* table that contains the raw data for all the customers from your Dynamics 365 subscription. You also have *ServiceCalls* raw data from the Service Center, with data from the support calls that were performed from the different account in each day of the year.

Imagine you want to enrich the *Account* table with data from the *ServiceCalls*. First you would need to aggregate the data from the *ServiceCalls* to calculate the number of support calls that were done for each account in the last year.



Next, you would want to merge the *Account* table with the *ServiceCallsAggregated* table to calculate the enriched *Account* table.



And then you can see the results, shown as *EnrichedAccount* in the following image.

Account	ServiceCalls	ServiceCallsAggregated	EnrichedAccount
			Name EnrichedAccount
			Entity type Account
			Applied steps
			Source X Expanded ServiceCalls...

The main table shows the following data:

ckexchange	ABC telephone1	ABC tickersymbol	ABC transactioncurrenc...	ABC websiteurl	1.2 numberofservicec...
1	(650) 667-3450	(null)	(null)	www.genepoint.com	83
2	(512) 757-6000	EDGE	(null)	http://edgecomm.com	87
3	+44 191 4956203	UOS	(null)	http://www.uos.com	86
4	(330) 222-7000	BTXT	(null)	www.burlington.com	87
5	(650) 450-8810	UOS	(null)	http://www.uos.com	83
6	(014) 427-4427	PYR	(null)	www.pyramid.com	96
7	(785) 241-6200	(null)	(null)	dickenson-consulting.com	87
8	(312) 596-1000	GHTL	(null)	www.grandhotels.com	87
9	(212) 842-5500	UOS	(null)	http://www.uos.com	87
10	(503) 421-7800	EXLT	(null)	www.expressl&t.net	97
11	(520) 773-9050	(null)	(null)	www.universityoffarizona.c...	87
12	(415) 901-7000	(null)	(null)	www.sforce.com	83

And that's it - the transformation is performed on the data in the dataflow that resides in your Power BI Premium subscription, not on the source data.

NOTE

Computed tables are a premium only feature

Create a dataflow using a CDM folder

Creating a dataflow from a CDM folder allows you to reference a table that has been written by another application in the Common Data Model (CDM) format. You are prompted to provide the complete path to the CDM format file stored in ADLS Gen 2.

The screenshot shows a dialog box titled "Attach an external CDM folder to a new dataflow". It contains fields for "Name" (set to "Dataflow from CDM folder"), "Description" (empty), and "CDM folder path" (with placeholder text "Enter the path to your CDM folder in Azure Data Lake Storage Gen2"). At the bottom are "Create and attach" and "Cancel" buttons.

Attach an external CDM folder to a new dataflow

Name *

Dataflow from CDM folder

Description

CDM folder path *

Enter the path to your CDM folder in Azure Data Lake Storage Gen2

Create and attach Cancel

There are a few requirements for creating dataflows from CDM folders, as the following list describes:

- The ADLS Gen 2 account must have the appropriate permissions set up in order for PBI to access the file
- The ADLS Gen 2 account must be accessible by the user trying to create the dataflow
- Creating dataflows from CDM folders is only available in the new workspace experience
- The URL must be a direct file path to the JSON file and use the ADLS Gen 2 endpoint; blob.core is not supported

Create a dataflow using import/export

Creating a dataflow using import/export lets you import a dataflow from a file. This is useful if you want to save a dataflow copy offline, or move a dataflow from one workspace to another.

To export a dataflow, select the dataflow you created and select the **More** menu item (the ellipsis) to expand the options, and then select **Export .json**. You are prompted to begin the download of the dataflow represented in CDM format.

The screenshot shows the Power BI Dataflows service interface. At the top, there's a navigation bar with 'New' and dropdown menus for 'All', 'Content', and 'Datasets + dataflows'. Below this is a table header with columns for 'Name', 'Type', and 'Owner'. A single row is visible, showing a thumbnail icon, the name 'Dataflow', its type as 'Dataflow', and the owner 'Mohammad Ali (MO...'. To the right of the table, a context menu is open, listing options: 'Delete', 'Edit', 'Export json', 'Properties', 'Refresh history', 'Settings', and 'View lineage'. The 'Edit' option is highlighted.

To import a dataflow, select the import box and upload the file. Power BI creates the dataflow for you, and allows you to save the dataflow as is, or to perform additional transformations.

Next steps

Once you create a dataflow, you can use Power BI Desktop and the Power BI service to create datasets, reports, dashboards, and apps that are based on the data you put into Power BI dataflows, and thereby gain insights into your business activities. The following articles go into more detail about common usage scenarios for dataflows:

- [Introduction to dataflows and self-service data prep](#)
- [Configure and consume a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [Premium features of dataflows](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

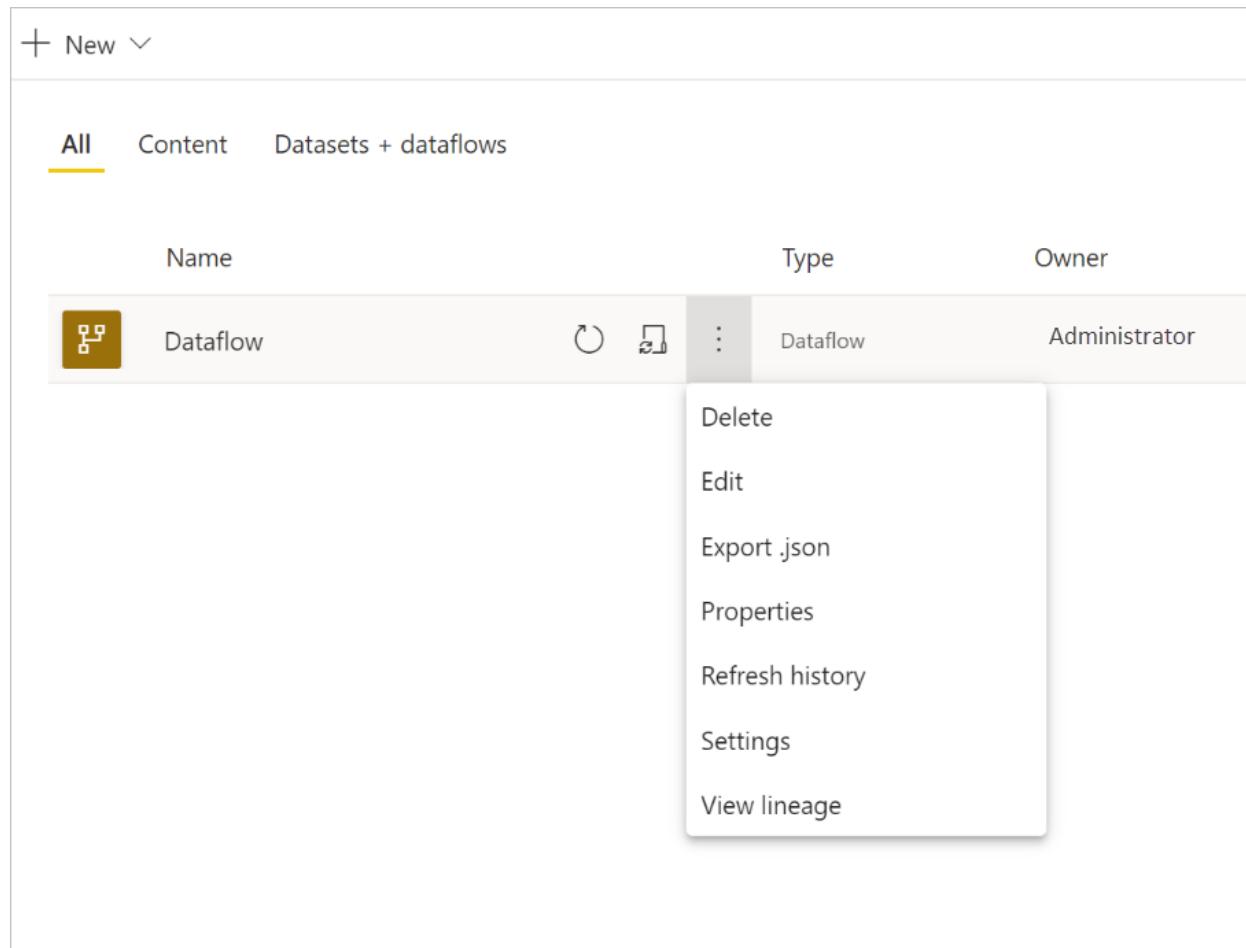
Configure and consume a dataflow

3/30/2022 • 4 minutes to read • [Edit Online](#)

With dataflows, you can unify data from multiple sources and prepare that unified data for modeling. Whenever you create a dataflow, you're prompted to refresh the data for the dataflow. Refreshing a dataflow is required before it can be consumed in a dataset inside Power BI Desktop, or referenced as a linked or computed table.

Configuring a dataflow

To configure the refresh of a dataflow, select the **More** menu (the ellipsis) and select **Settings**.



The screenshot shows the Power BI Dataflows blade. At the top, there's a header with a '+ New' button and dropdown menus for 'All', 'Content', and 'Datasets + dataflows'. Below this is a table with columns for 'Name', 'Type', and 'Owner'. A single row is visible, showing 'Dataflow' as the name, 'Dataflow' as the type, and 'Administrator' as the owner. To the right of this row is a context menu with the following options: Delete, Edit, Export .json, Properties, Refresh history, Settings, and View lineage. The 'Settings' option is highlighted.

Name	Type	Owner
Dataflow	Dataflow	Administrator

- Delete
- Edit
- Export .json
- Properties
- Refresh history
- Settings
- View lineage

The **Settings** options provide many options for your dataflow, as the following sections describe.

Settings for Dataflow

This dataflow has been configured by Administrator@Contoso.com

[Refresh history](#)

- ▶ Gateway connection
- ▶ Data source credentials
- ▶ Sensitivity label
- ▶ Scheduled refresh
- ▶ Enhanced compute engine settings (preview)
- ▶ Endorsement (preview)

- **Take ownership:** If you're not the owner of the dataflow, many of these settings are disabled. To take ownership of the dataflow, select **Take over** to take control. You are prompted to provide credentials to ensure you have the necessary access level.
- **Gateway Connection:** In this section, you can choose whether the dataflow uses a gateway, and select which gateway is used.
- **Data Source Credentials:** In this section you choose which credentials are being used, and can change how you authenticate to the data source.
- **Sensitivity Label:** Here you can define the sensitivity of the data in the dataflow. To learn more about sensitivity labels, see [how to apply sensitivity labels in Power BI](#).
- **Scheduled Refresh:** Here you can define the times of day the selected dataflow refreshes. A dataflow can be refreshed at the same frequency as a dataset.
- **Enhanced Compute Engine settings:** Here you can define whether the dataflow is stored inside the compute engine. The compute engine allows subsequent dataflows, which reference this dataflow, to perform merges and joins and other transformations much faster than you would otherwise. It also allows DirectQuery to be performed over the dataflow. Selecting **On** ensures the dataflow is always supported in DirectQuery mode, and any references benefit from the engine. Selecting **Optimized** means the engine is only used if there is a reference to this dataflow. Selecting **Off** disables the compute engine and DirectQuery capability for this dataflow.
- **Endorsements:** You can define whether the dataflow is certified or promoted.

NOTE

Dataflows can be created by user in a Premium workspace, users with a Pro license, and users with a Premium Per User (PPU) license.

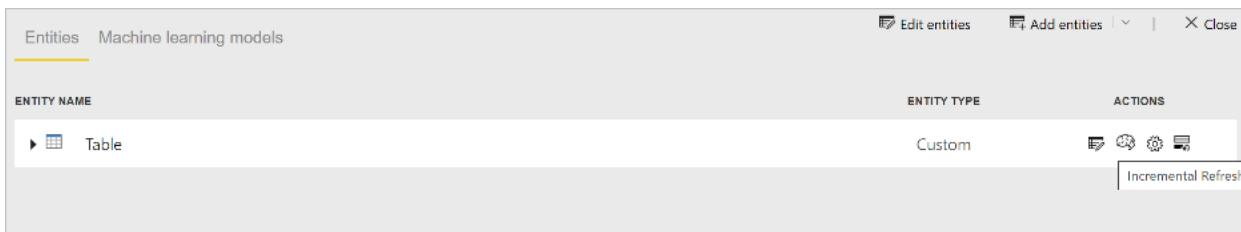
Refreshing a dataflow

Dataflows act as building blocks on top of one another. Suppose you have a dataflow called *Raw Data* and a linked table called *Transformed Data* which contains a linked table to the *Raw Data* dataflow. When the schedule refresh for the dataflow *Raw Data* triggers, it will trigger any dataflow that references it upon completion. This functionality creates a chain effect of refreshes, allowing you to avoid having to schedule dataflows manually. There are a few nuances to be aware of when dealing with linked tables refreshes:

- A linked table will be triggered by a refresh only if it exists in the same workspace
- A linked table will be locked for editing if a source table is being refreshed. If any of the dataflows in a reference chain fail to refresh, all the dataflows will roll back to the old data (dataflow refreshes are transactional within a workspace).
- Only referenced tables are refreshed when triggered by a source refresh completion. To schedule all the tables, you should set a schedule refresh on the linked table as well. Avoid setting a refresh schedule on linked dataflows to avoid double refresh.

Cancel Refresh Dataflows support the ability to cancel a refresh, unlike datasets. If a refresh is running a long time, you can select the dataflow options (the ellipses next to the dataflow) and then select **Cancel refresh**.

Incremental Refresh (Premium only) Dataflows can be also set to refresh incrementally. To do so, select the dataflow you wish to set up for incremental refresh, and then select the incremental refresh icon.



Setting incremental refresh adds parameters to the dataflow to specify the date range. For detailed information on how to set up incremental refresh, see the [incremental refresh in Power Query](#) article.

There are some circumstances under which you should not set incremental refresh:

- Linked tables should not use incremental refresh if they reference a dataflow. Dataflows do not support query folding (even if the table is Direct Query enabled).
- Datasets referencing dataflows should not use incremental refresh. Refreshes to dataflows are generally performant, so incremental refreshes shouldn't be necessary. If refreshes take too long, consider using the compute engine, or DirectQuery mode.

Consuming a dataflow

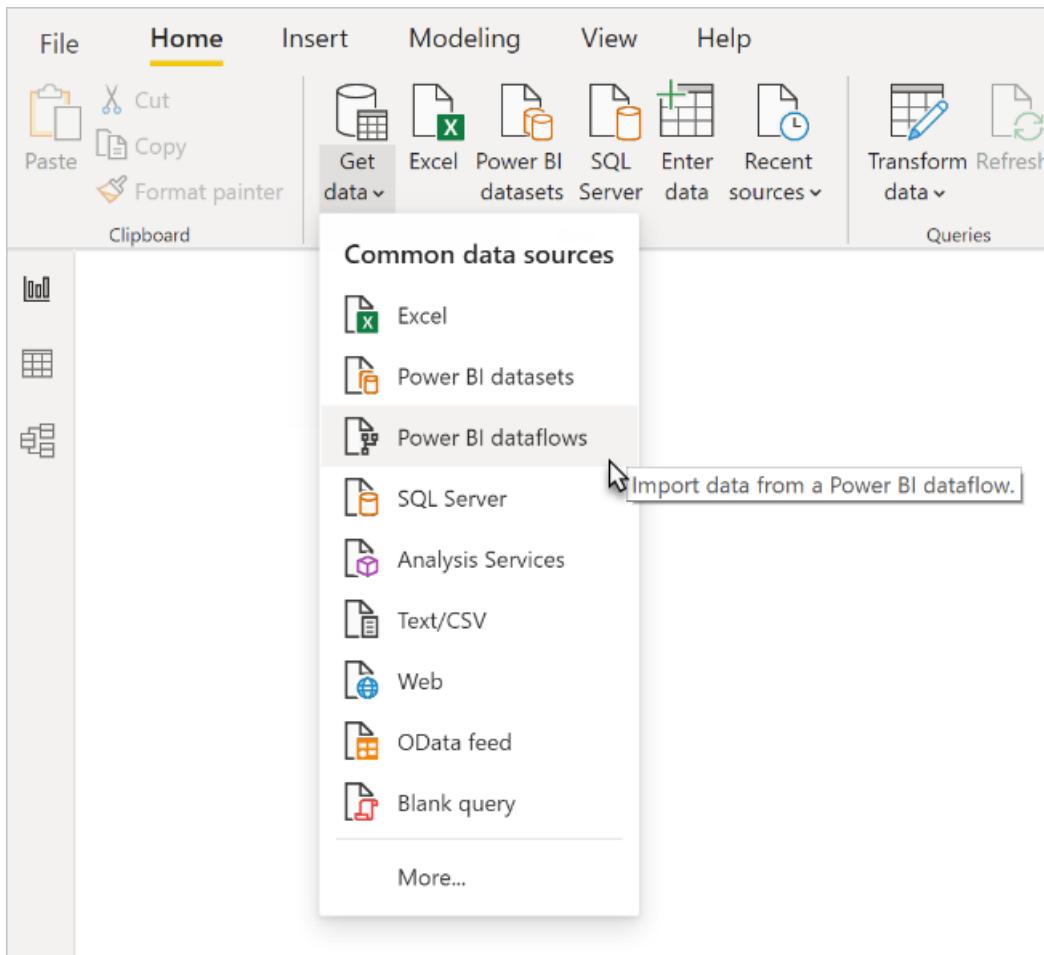
A dataflow can be consumed in the following three ways:

- Create a linked table from the dataflow to allow another dataflow author to use the data
- Create a dataset from the dataflow to allow a user to utilize the data to create reports
- Create a connection from external tools that can read from the CDM format

Consuming from Power BI Desktop To consume a dataflow, run Power BI Desktop and select the **Power BI dataflows connector** in the **Get Data** dialog.

NOTE

The Power BI dataflows connector uses a different set of credentials than the current logged in user. This is by design, to support multi-tenant users.



Select the dataflow and tables to which you want to connect.

NOTE

You can connect to any dataflow or table regardless of which workspace it resides in, and whether or not it was defined in a Premium or non-Premium workspace.

A screenshot of the Navigator pane in Microsoft Power BI desktop. The pane has a search bar at the top. Below it are 'Display Options' and a refresh icon. The main area is a tree view of dataflows and tables. Under 'Power BI dataflows [3]', there are three collapsed items: 'Dataflow Projects', 'Dataflow Team', and 'Dataflows Bugs'. Under 'TestDataflow [1]', there is one expanded item: 'Table'.

If DirectQuery is available, you're prompted to choose whether you want to connect to the tables through DirectQuery or Import.

In DirectQuery mode, you can quickly interrogate large-scale datasets locally. However, you cannot perform any additional transformations.

Using Import brings the data into Power BI, and requires the dataset to be refreshed independently of the dataflow.

Next steps

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [Premium features of dataflows](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

Configure Power BI Premium dataflow workloads

3/30/2022 • 12 minutes to read • [Edit Online](#)

You can create dataflow workloads in your Power BI Premium subscription. Power BI uses the concept of *workloads* to describe Premium content. Workloads include datasets, paginated reports, dataflows, and AI. The *dataflows* workload lets you use dataflows self-service data preparation to ingest, transform, integrate, and enrich data. Power BI Premium dataflows are managed in the **Admin portal**.

The following sections describe how to enable dataflows in your organization, how to refine their settings in your Premium capacity, and guidance for common usage.

Enabling dataflows in Power BI premium

The first requirement for using dataflows in your Power BI premium subscription is to enable the creation and use of dataflows for your organization. In the **Admin portal**, select **Tenant Settings** and switch the slider under **Dataflow settings** to **Enabled**, as shown in the following image.

The screenshot shows the 'Admin portal' interface. On the left, a sidebar lists various settings: Tenant settings (selected), Usage metrics, Users, Premium Per User, Audit logs, Capacity settings, Refresh summary, Embed Codes, Organizational visuals, Azure connections, Workspaces, Custom branding, Protection metrics, and Featured content. The main area is titled 'Dataflow settings'. It contains a section with a bullet point: 'Create and use dataflows' (Enabled for the entire organization). Below this is a note: 'Users in the organization can create and use dataflows. [Learn more](#)'. A large yellow 'Enabled' button is shown. At the bottom are 'Apply' and 'Cancel' buttons, and a note: 'This setting applies to the entire organization'.

After enabling the dataflows workload, it is configured with default settings. You might want to tweak these settings as you see fit. Next, we'll describe where these settings live, describe each, and help you understand when you might want to change the values to optimize your dataflow performance.

Refining dataflow settings

Once dataflows are enabled, you can use the **Admin portal** to change, or refine, how dataflows are created and how they use resources in your Power BI Premium subscription. The following steps show how to adjust your dataflow settings.

1. In the **Admin portal**, select **Tenant settings** to list all capacities that have been created. Select a capacity to manage its settings.

The screenshot shows the Power BI Admin portal interface. On the left is a navigation sidebar with various links like Home, Favorites, Recent, Create, Datasets, Goals, Apps, Shared with me, Learn, Workspaces, and My workspace. The 'Capacity settings' link under 'Workspaces' is highlighted with a grey background. The main content area is titled 'Admin portal' and contains a list of capacity-related options: Tenant settings, Usage metrics, Users, Premium Per User, Audit logs, Capacity settings (which is also highlighted with a grey background), Refresh summary, Embed Codes, Organizational visuals, Azure connections, Workspaces, Custom branding, Protection metrics, and Featured content.

2. Your Power BI Premium capacity reflects the resources available for your dataflows. You can change your capacity's size by selecting the **Change size** button, as shown in the following image.

The screenshot shows the 'Power BI Premium > Sample Capacity' page. On the left is a sidebar with links: Usage metrics, Users, Audit logs, Tenant settings, Capacity settings (which is highlighted with a grey background), Refresh summary, Embed Codes, Organizational visuals, Azure connections (preview), Workspaces, Custom branding, Protection metrics, and Featured content. The main content area is titled 'Power BI Premium > Sample Capacity'. It has two tabs at the top: 'Management' (which is selected) and 'Health'. Below the tabs is a large empty grey box. At the bottom of the page, there is a section titled 'CAPACITY SIZE' with the text 'The Premium SKU size you purchased is a P1, which gives you access to 8 v-cores.' Below this text is a yellow rectangular button with the text 'Change size' in black. This 'Change size' button is highlighted with a red border.

3. In Capacity settings, you can configure dataflow settings by expanding **Workloads**.

CAPACITY SIZE

The Premium SKU size you purchased is a P1, which gives you access to 8 v-cores.

[Change size](#)

REGION

East US 2

USER PERMISSIONS

- ▶ Capacity admins
- ▶ Users with assignment permissions
Disabled for the entire organization

MORE OPTIONS

- ▶ **Workloads**
- ▶ Advanced options

4. In the **Workloads** section, scroll to the **Dataflows** area. The following image shows the settings you can use to control or refine the dataflow workload's behavior for your capacity.

DATAFLOWS - Active

Your workload is ready to use.

On

Max Memory (%)

20

Enhanced Dataflows Compute Engine

Off

Container Size (Mb)

700

Compute engine memory (%)

30

The following table provides a basic description of the dataflows settings.

ADMIN SECTION	SETTING NAME	DESCRIPTION
Capacity Size	Change Size	The current selected capacity is listed, with options to change the capacity. Changing this setting allows for scale up or scale down of the capacity.
Workload	Max Memory (%)	The maximum percentage of available memory that dataflows can use in a capacity.
Workload	Enhanced Dataflows Compute Engine	Enable this option for up to 20x faster calculation of computed entities when working with large-scale data volumes. You must restart the capacity to activate the new engine. For more information, see Enhanced dataflows compute engine .
Workload	Container Size	The maximum size of the container that dataflows use for each table in the dataflow. The default value is 700 MB. For more information, see Container size .
Workload	Compute engine memory (%)	The maximum percentage of memory allocated to the compute engine. The default value is 30%.

In the following sections, we go into detail about each of the settings and how they affect your dataflow workload.

Understanding dataflow workload options

A simple way to think about dataflow workload options is to use an analogy. The *capacity size*, or the type of Power BI Premium Instance you have, can be thought of as your *restaurant*. Within your restaurant, you have your *workload memory*, which is your *kitchen*. The *compute engine* is your *oven*. And lastly, the *container* is the quality of your *chef*. To evaluate your dataflow workload options, imagine preparing a meal for a large, or very significant dinner. You have important guests coming over and you must have dinner ready to be served by the time they arrive.

We'll use this restaurant analogy as we explain and provide guidance on each of the setting. We'll start at the top level - your Premium capacity - as this is the first choice you make when using Power BI Premium.

Premium capacity SKUs - scale up the hardware

Power BI Premium workloads use a combination of front-end and backend cores to serve fast queries across the various workload types. The [capacity nodes](#) article includes a chart that illustrates the current specifications across each of the available workload offerings. Capacities of A3 and greater can take advantage of the compute engine, so when you want to use the enhanced compute engine, start there – [Capacity nodes](#).

In our restaurant analogy, choosing a capacity is like choosing a higher-quality restaurant. Though a higher cost, you can expect a higher level of performance due to the increase in front-end cores, backend cores, and more memory. When you go to a bigger restaurant, you get a bigger kitchen, and better chefs, which are akin to upgrading to a higher SKU in Power BI Premium, giving you the benefit of increasing CPU speed, increasing memory per operation, and adding more parallelism.

Max memory - dedicating a capacity for dataflows

The **Max Memory %** setting is the percentage of memory, out of the physical memory available to the Premium capacity, provided for dataflows workloads. You can effectively dedicate up to the full capacity for a dataflows workload, and the capacity will scale up dynamically as needed to the allocation you set. In our analogy, if you make your kitchen bigger you can cook more meals – similarly, you can increase your capacity's workload size for dataflows and allow more dataflows. While there is dynamic resource governance in place, the **Max Memory %** configuration option lets you dedicate 100% of the memory for the dataflows workload. This is for the rare situation when you want to ensure the capacity's memory is available for your dataflow workload, rather than relying on the Resource Governance. Using our analogy, this would be like making sure your kitchen is focusing on preparing a specific meal for a guest, dedicating the full kitchen to the task(s). Of course, the ability to dedicate more doesn't mean better food, or faster time to table – as the next section explains.

Container size - refresh or out of memory issues

Next let's discuss the **Container Size (Mb)** setting. Internally, dataflows use a process called *mashup containers* to evaluate your ETL processes. The engine splits your query logic into these containers and they can process in parallel. The number of containers effectively provides concurrent processing and increases performance. These containers are constrained by the **Capacity** first, the **Max Memory %** setting second, and then the amount of memory you specifically allocate to them in the container setting, which by default is 700Mb. So it's possible to increase the amount of hardware memory, and increase the container size, but doing so will decrease the parallel operations while dedicating more memory for specific ETL process within your containers. The number of containers is capped at three times the number of backend cores, which is important because you can't make the container very small, and have a lot of parallel containers beyond that point. The minimum size you can make a container is 200Mb. The container size is also scoped to a query level, which means each query gets executed in its own container except when queries are referencing other queries, in which case they are refreshed as part of the same container.

Going back to our analogy, having fewer but more focused cooks in the kitchen enables you to cook a meal much faster, depending on the types of orders that are placed and the complexity of the meal. The tradeoff here is having fewer chefs, but more focused time for preparation. Similarly, increasing the container size measure to 1200-1500 MB can mean that a smaller number of more complex ETL jobs – activities such as aggregations, joins, pivots, row or column manipulations – can see a performance increase as we provide more memory for each container, but in doing so, reduce the number of containers. Just as the analogy implies, too many orders can actually slow down the kitchen output, which is how you can think about container size – use this when you need complex table operations to complete, and you're willing to trade parallelism for performance, since increasing this resource divides the memory allocated to fewer containers.

To summarize, you want to optimize your container size based on the queries being used. For example, simply loading data from a source into a table does not need to pull in data and perform any operations and just loads the data to storage. You want as much parallelism as possible for this situation, as you want to increase the speed of the load and refresh operations. Conversely, when you add more transformation operations – complex filters, joins, aggregations, the memory may be much higher as we may need to process some of these transformation operations in memory. Be aware that if you have other dataflow operations running on the capacity, it can slow those operations and force them to queue up to have an execution slot. To this end, monitoring and management of dataflows, refresh performance, and the entire capacity is aided by the **Power BI Premium Capacity Metrics** app. You can use the [Power BI Premium Capacity Metrics](#) app to filter by capacity, and review performance metrics for workspace content. It's possible to review the performance metrics and resource usage by hour, for the past seven days, for all content stored within a Premium capacity – so for investigating dataflow performance, it's recommended to start with the app.

Enhanced compute engine - an opportunity to improve performance

In our analogy, the [enhanced compute engine](#) is like an oven. Power BI uses a compute engine to process your queries and refresh operations. The enhanced compute engine is an improvement over the standard engine, and works by loading data to a SQL Cache and uses SQL to accelerate table transformation, refresh operations and enables DirectQuery connectivity. If we compare the engines to ovens, as you leverage the enhanced oven, you may be able to cook meals faster and more effectively. When configured to **On** or **Optimized** for computed

entities, if your business logic allows for it, Power BI uses SQL speed up the performance. Having the engine On also provides for DirectQuery connectivity. As the analogy suggests – certain meals might not need an oven, nor take advantage of the oven. The enhanced compute engine can be thought of in a similar manner – make sure your dataflow usage is leveraging the enhanced compute engine properly. Users can configure the enhanced compute engine to be on, optimized, or off on a per-dataflow basis.

NOTE

The enhanced compute engine is not yet available in all regions.

Guidance for common scenarios

This section provides guidance for common scenarios when using dataflow workloads with Power BI Premium.

Slow refresh times

Slow refresh times are usually a parallelism issue. You should review the following options, in order:

1. A key concept for slow refresh times is the nature of your data preparation. In our restaurant analogy explained earlier in this article, imagine having prepared food already, waiting to be used. In this scenario, the food can be cooked much faster due to minimal prep time. Similarly, whenever you can optimize your slow refresh times by taking advantage of your data source actually doing the preparation and performing upfront query logic, you should do so. Specifically, when using a relational database such as SQL as your source, see if the initial query can be run on the source, and use that source query for your initial extraction dataflow for the data source. If you cannot use a native query in the source system, perform operations that the dataflows [engine can fold to the data source](#).
2. Evaluate spreading out refresh times on the same capacity. Refresh operations are a process that requires significant compute. Using our restaurant analogy, spreading out refresh times is akin to limiting the number of guests in your restaurant. Just as restaurants will schedule guests and plan for capacity, you also want to consider refresh operations during times when usage is not at its full peak. This can go a long way toward alleviating strain on the capacity.
3. Increase the overall amount of memory given to the workload. Think of this as the size of kitchen. Refining this resource is similar to adjusting how many chefs can you fit in the kitchen. This is done by adjusting the **Max Memory %** setting, and increasing it up to 100%.
4. Decrease the amount of memory to the container, which allows for more containers. You can think of this as: instead of hiring a famously capable chef like Gordon Ramsey, hire many competent but less expensive chefs. So you have more cooks in the kitchen, but those chefs can only perform smaller tasks. Then you have more containers, but less memory.
5. Do both of the previous steps, allowing for an even higher degree of parallelism, because you get more cooks and a bigger kitchen.
6. If the steps in this section don't provide the desired degree of parallelism, consider upgrading your capacity to a higher SKU. Then follow the previous steps in this sequence again.

Out of memory exceptions

When you experience **out of memory exceptions**, you need to increase performance of the containers and memory. Take the following steps:

1. Increase memory on the container. This is similar to having one star chef versus many chefs, as described in the previous section.
2. Increase memory to workload and more memory to container. In our analogy, it creates a bigger kitchen and higher-quality chefs.

3. If these changes don't give you the desired degree of parallelism, consider a higher Power BI Premium SKU.

Using the compute engine to improve performance

Take the following steps to enable workloads trigger the compute engine, and always improve performance:

For computed and linked entities in the same workspace:

1. For *ingestion* focus on getting the data into the storage as fast as possible, using filters only if they reduce the overall dataset size. It's best practice to keep your transformation logic separate from this step, and allow the engine to focus on the initial gathering of ingredients. Next, separate your transformation and business logic into a separate dataflow in the same workspace, using linked or computed entities; doing so allows for the engine to activate and accelerate your computations. In our analogy, it's like food preparation in the kitchen: food preparation is typically a separate and distinct step from gathering your raw ingredients, and a pre-requisite for putting the food in the oven. Similarly, your logic needs to be prepared separately before it can take advantage of the compute engine.
2. Ensure you perform the operations that fold, such as merges, joins, conversion, and [others](#).
3. Building dataflows [within published guidelines and limitations](#).

You can also use DirectQuery.

Compute engine is on but performance is slow

Take the following steps when investigating scenarios where the Compute engine is on, but you're seeing slower performance:

1. Limit computed and linked entities that exist across workspace.
2. When you perform your initial refresh with the compute engine turned on, then data gets written in the lake and in the cache. This double write means these refreshes will be slower.
3. If you have a dataflow linking to multiple dataflows, make sure you schedule refreshes of the source dataflows so that they do not all refresh at the same time.

Next steps

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)

Configuring dataflow storage to use Azure Data Lake Gen 2

3/30/2022 • 8 minutes to read • [Edit Online](#)

Data used with Power BI is stored in internal storage provided by Power BI by default. With the integration of dataflows and Azure Data Lake Storage Gen 2 (ADLS Gen2), you can store your dataflows in your organization's Azure Data Lake Storage Gen2 account. This essentially allows you to "bring your own storage" to Power BI dataflows, and establish a connection at the tenant or workspace level.

Reasons to use the ADLS Gen 2 workspace or tenant connection

After you attach your dataflow, Power BI configures and saves a reference so that you can now read and write data to your own ADLS Gen 2. Power BI stores the data in the CDM format, which captures metadata about your data in addition to the actual data generated by the dataflow itself. This unlocks many powerful capabilities and enables your data and the associated metadata in CDM format to now serve extensibility, automation, monitoring, and backup scenarios. By making this data available and widely accessible in your own environment, it enables you to democratize the insights and data created within the organization. It also unlocks the ability for you to create further solutions that are either CDM aware (such as custom applications and solutions in Power Platform, Azure, and those available through partner and ISV ecosystems) or simply able to read a CSV. Your data engineers, data scientists, and analysts can now work with, use, and reuse a common set of data that is curated in ADLS Gen 2.

There are two ways to configure which ADLS Gen 2 store to use: you can use a tenant-assigned ADLS Gen 2 account, or you can bring your own ADLS Gen 2 store at a workspace level.

Prerequisites

- To bring your own ADLS Gen 2 account, you must have [Owner](#) permission at the storage account layer. Permissions at the resource group or subscription level will *not* work. If you are an administrator, you still must assign yourself [Owner](#) permission. **Currently not supporting ADLS Gen2 Storage Accounts behind a firewall.**
- The storage account must be created with the [Hierarchical Namespace \(HNS\)](#) enabled.
- The storage account must be created in the same Azure Active Directory tenant as the Power BI tenant.
- The user must have Azure Blob Data Contributor role, and a Owner role at the storage account level.
- The Power BI workspace tenant region should be the same as the storage account region.
- TLS (Transport Layer Security) version 1.2 (or higher) is required to secure your endpoints. Web browsers and other client applications that use TLS versions earlier than TLS 1.2 won't be able to connect.
- Attaching a dataflow with ADLS Gen 2 behind multifactor authentication (MFA) is not supported.
- Finally, you can connect to any ADLS Gen 2 from the admin portal, but if you connect directly to a workspace, you must first ensure there are no dataflows in the workspace before connecting.

The following table describes the permissions for ADLS and for Power BI required for ADLS Gen 2 and Power BI:

Action	ADLS Permissions	Minimum Power BI Permissions
Connect ADLS Gen 2 to Power BI tenant	Owner	Power BI administrator
Connect ADLS Gen 2 to Workspace	Owner	Workspace Admin
Create Power BI dataflows writing back to connected ADLS account	Not applicable	Workspace contributor
Consume Power BI dataflow	Not applicable	Workspace viewer

Connecting to an Azure Data Lake Gen 2 at a workspace level

Navigate to a workspace that has no dataflows. Select **Workspace settings**. Select the **Azure Connections** tab and then select the **Storage** section.

The screenshot shows the 'Storage' configuration page within the 'Azure connections' tab of the Power BI workspace settings. It includes a 'Connect to Azure' button and a checkbox for 'Use the default Azure connection'.

The **Use default Azure connection** option is visible if admin has already configured a tenant-assigned ADLS Gen 2 account. You have two options:

- Use the tenant configured ADLS Gen 2 account by selecting the box called **Use the default Azure connection**, or
- Select **Connect to Azure** to point to a new Azure Storage account.

When you select **Connect to Azure**, Power BI retrieves a list of Azure subscriptions to which you have access. Fill in the dropdowns and select a valid Azure subscription, resource group, and storage account that has the hierarchical namespace option enabled, which is the ADLS Gen2 flag.

⚙️ Settings

Dataflow

About

Premium

Azure connections

◀ Storage

Connect an Azure Data Lake Gen2 storage account for dataflow storage. [Learn more](#)

[Connect to Azure](#)

Subscription

Microsoft Azure Internal Consump... ▾

Resource group

PlayPen ▾

Storage account

storage01 ▾

[Save](#)

[Cancel](#)

Use the default Azure connection

Once selected, select **Save** and you now have successfully connected the workspace to your own ADLS Gen2 account. Power BI automatically configures the storage account with the required permissions, and sets up the Power BI filesystem where the data will be written. At this point, every dataflow's data inside this workspace will write directly to this filesystem, which can be used with other Azure services, creating a single source for all of your organizational or departmental data.

Understanding configuration

Configuring Azure connections is an optional setting with additional properties that can optionally be set:

- Tenant Level storage, which lets you set a default, and/or
- Workspace-level storage, which lets you specify the connection per workspace

You can optionally configure tenant-level storage if you want to use a centralized data lake only, or want this to be the default option. We don't automatically start using the default to allow flexibility in your configuration, so you have flexibility to configure the workspaces that use this connection as you see fit. If you configure a tenant-assigned ADLS Gen 2 account, you still have to configure each workspace to use this default option.

You can optionally, or additionally, configure workspace-level storage permissions as a separate option, which

provides complete flexibility to set a specific ADLS Gen 2 account on a workspace by workspace basis.

To summarize, if tenant-level storage and workspace-level storage permissions are allowed, then workspace admins can optionally use the default ADLS connection, or opt to configure another storage account separate from the default. If tenant storage is not set, then workspace Admins can optionally configure ADLS accounts on a workspace by workspace basis. Finally, if tenant-level storage is selected and workspace-level storage is disallowed, then workspace admins can optionally configure their dataflows to use this connection.

Understanding the structure and format for ADLS Gen 2 workspace connections

In the ADLS Gen 2 storage account, all dataflows are stored in the **powerbi** container of the filesystem.

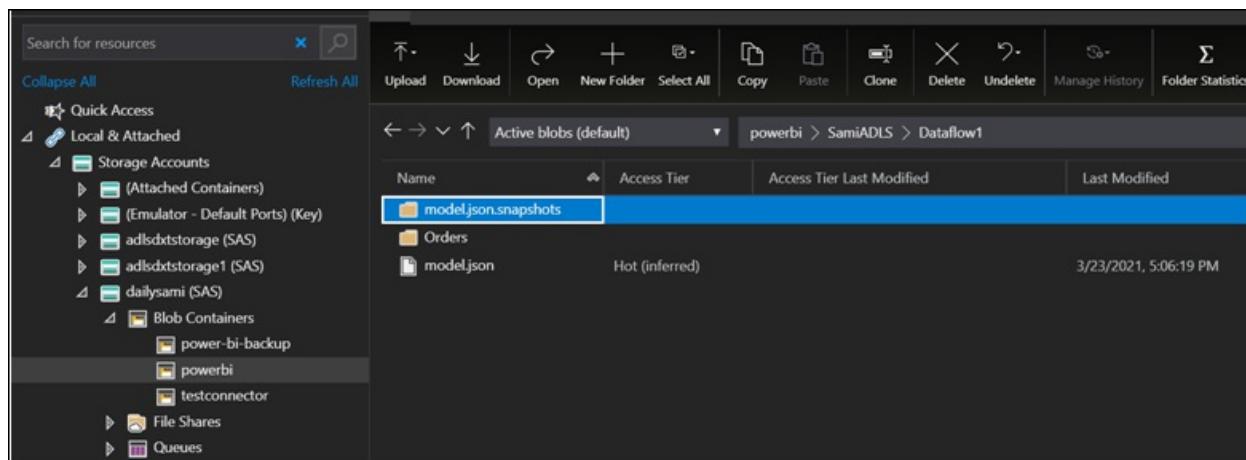
The structure of the **powerbi** container looks like this:

```
<workspace name>/<dataflow name>/model.json <workspace name>/<dataflow name>/model.json.snapshots/<all snapshots>
```

The location where dataflows store data in the folder hierarchy for ADLS Gen 2 is determined by whether the workspace is located in shared capacity or Premium capacity. The file structure after refresh for each capacity type is shown in the table below.

PREMIUM CAPACITY	SHARED CAPACITY
<workspace name>/<dataflow name>/<table name>/<tablesnapshots>	<workspace name>/<dataflow name>/<table name>/<tablesnapshots>

Below is an example using the Orders table of the Northwind Odata sample.



In the image above:

- The **model.json** is the most recent version of the dataflow.
- The **model.json.snapshots** are all previous versions of the dataflow. This is useful if you need a previous version of mashup, or incremental settings.
- The **table.snapshots.csv** is the data you got from a refresh. This is useful for incremental refreshes, and also for shared refreshes where a user is running into a refresh timeout issue because of data size. They can look at the most recent snapshot to see how much data is in the csv file.

We only write to this storage account and do not currently delete data. This means that even after detach, we don't delete from the ADLS account, so all of the above files are still stored.

NOTE

A model.json file can refer to another model.json that is another dataflow in the same workspace, or in a dataflow in another workspace. The only time where a model.json would refer to a table.snapshot.csv is for incremental refresh.

Extensibility for ADLS Gen 2 workspace connections

If you are connecting ADLS Gen 2 to Power BI, you can do this at the workspace or tenant level. Make sure you have the right access level. Learn more in [Prerequisites](#).

The storage structure adheres to the Common Data Model format. Learn more about the storage structure and CDM by visiting [What is the storage structure for analytical dataflows](#) and [Common Data Model and Azure Data Lake Storage Gen2](#).

Once properly configured, the data and metadata is in your control. A number of applications are aware of the CDM and the data can be extended using Azure, PowerApps, and PowerAutomate, as well as third-party ecosystems either by conforming to the format or by reading the raw data.

Detaching Azure Data Lake Gen 2 from a workspace or tenant

To remove a connection at a workspace level, you must first ensure all dataflows in the workspace are deleted. Once all the dataflows have been removed, select **Disconnect** in the workspace settings. The same applies for a tenant, but you must first ensure all workspaces have also been disconnected from the tenant storage account before you are able to disconnect at a tenant level.

Disabling Azure Data Lake Gen 2

In the [Admin portal](#), under **dataflows**, you can disable access for users to either use this feature, and can disallow workspace admins to bring their own Azure Storage.

Reverting from Azure Data Lake Gen 2

Once the dataflow storage has been configured to use Azure Data Lake Gen 2, there is no way to automatically revert. The process to return to Power BI-managed storage is manual.

To revert the migration that you made to Gen 2, you will need to delete your dataflows and recreate them in the same workspace. Then, since we don't delete data from ADLS Gen 2, go to the resource itself and clean up data. This would involve the following steps.

1. Export a copy of the dataflow from Power BI. Or, copy the model.json file. The model.json file is stored in ADLS.
2. Delete the dataflows.
3. Detach ADLS.
4. Recreate the dataflows using import. Note that incremental refresh data (if applicable) will need to be deleted prior to import. This can be done by deleting the relevant partitions in the model.json file.
5. Configure refresh / recreate incremental refresh policies.

Connecting to the data using the ADLS Gen 2 connector

The scope of this document describes ADLS Gen 2 dataflows connections and not the Power BI ADLS Gen 2 connector. Working with the ADLS Gen 2 connector is a separate, possibly additive, scenario. The ADLS connector simply uses ADLS as a datasource. This means that using PQL to query against that data doesn't

have to be in CDM format, it can be whatever data format the customer wants. Learn more about this scenario by visiting [Analyze data in Azure Data Lake Storage Gen2 by using Power BI](#).

Next steps

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Premium features of dataflows](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

Premium features of dataflows

3/30/2022 • 7 minutes to read • [Edit Online](#)

Dataflows are supported for Power BI Pro, Premium Per User (PPU), and Power BI Premium users. Some features are only available with a Power BI Premium subscription (which is either a Premium capacity or Premium Per User (PPU) license). This article describes and details the Premium Per User (PPU) and Premium-only features and their uses.

The following features are available only with Power BI Premium (PPU or a Premium capacity subscription):

- Enhanced compute engine
- Direct Query
- Computed entities
- Linked Entities
- Incremental refresh

The following sections described each of these features in detail.

The enhanced compute engine

The enhanced compute engine in Power BI enables Power BI Premium subscribers to use their capacity to optimize the use of dataflows. Using the enhanced compute engine provides the following advantages:

- Drastically reduces the refresh time required for long-running ETL steps over computed entities, such as performing *joins*, *distinct*, *filters*, and *group by*
- Performs DirectQuery queries over entities

NOTE

- The validation and refresh processes inform dataflows of the model schema. To set the schema of the tables yourself, use the PowerQuery Editor and set data types.
- This feature is available on all Power BI clusters except WABI-INDIA-CENTRAL-A-PRIMARY

Enable the enhanced compute engine

IMPORTANT

The enhanced compute engine works only for A3 or larger Power BI capacities.

- [Premium Gen2](#)
- [Premium Gen1](#)

In Premium Gen2, the enhanced compute engine is individually set for each dataflow. There are three configurations to choose from:

- **Disabled**
- **Optimized** (default) - The enhanced compute engine is turned off. It is automatically turned on when the dataflow is connected to another dataflow.
- **On**

To change the default setting and enable the enhanced compute engine, do the following:

1. In your workspace, next to the dataflow you want to change the settings for, select **More options**.
2. From the dataflow's *more options* menu, select **Settings**.

The screenshot shows the Microsoft Power BI Admin interface. At the top, there is a navigation bar with tabs: All (highlighted), Content, Datasets + dataflows, and Datamarts (Preview). Below the navigation bar is a table with columns: Name and Type. The table lists several items: Search Report (Report), LOG (Dataflow), test (Dataflow), Audit Log (Dataflow), Feature Adoption (Dataflow), Internal (Dataflow), Audit Log (Dataflow), and Feature Adoption (Dataflow). For the item 'test', a context menu is open, showing options: Delete, Edit, Export .json, Properties, Refresh history, Settings (which is highlighted with a red box), and View lineage.

Name	Type
Search Report	Report
LOG	Dataflow
test	Dataflow
Audit Log	Dataflow
Feature Adoption	Dataflow
Internal	Dataflow
Audit Log	Dataflow
Feature Adoption	Dataflow

3. Expand the Enhanced compute engine settings.

The screenshot shows the 'Dataflows' settings page for a dataflow named 'test'. On the left, there is a sidebar with a list of dataflows: LOG, test (selected), Audit Log, Feature Adoption, and Internal. The main area displays the 'Settings for danavotest' for the selected dataflow. It includes a message about last modification by 'davo@soft.com' and links for Refresh history, Gateway connection, Data source credentials, Sensitivity label, Scheduled refresh, Enhanced compute engine settings (which is highlighted with a red box), and Endorsement.

4. In the *Enhanced compute engine settings*, select **On** and then select **Apply**.

▪ Enhanced compute engine settings
Configure enhanced compute engine settings for this dataflow.

Disabled
Turn off the enhanced compute engine for this dataflow.

Optimized
We'll turn on the enhanced compute engine only when this dataflow is linked to another one, which will enhance performance.

On
Turn on the enhanced compute engine for this dataflow.

Apply **Discard**

Using the enhanced compute engine

Once the enhanced compute engine is on, return to **dataflows** and you should see a performance improvement in any computed table that performs complex operations, such as *joins* or *group by* operations for dataflows created from existing linked entities on the same capacity.

To make best use of the compute engine, split the ETL stage into two separate dataflows, in the following way:

- **Dataflow 1** - this dataflow should only be ingesting all of the required from a data source, and placing it into dataflow 2.
- **Dataflow 2** - perform all ETL operations in this second dataflow, but ensure you're referencing Dataflow 1, which should be on the same capacity. Also ensure you perform operations that can fold (filter, group by, distinct, join) first, before any other operation, to ensure the compute engine is utilized.

Common questions and answers

Question: I've enabled the enhanced compute engine, but my refreshes are slower. Why?

Answer: If you enable the enhanced compute engine, there are two possible explanations that could lead to slower refresh times:

- When the enhanced compute engine is enabled, it requires some memory to function properly. As such, memory available to perform a refresh is reduced and therefore increases the likelihood of refreshes to be queued, which in turn reduces the number of dataflows that can refresh concurrently. To address this, when enabling enhanced compute, increase the memory assigned for dataflows to ensure the memory available for concurrent dataflow refreshes remains the same.
- Another reason you may encounter slower refreshes is that the compute engine only works on top of existing entities. If your dataflow references a data source that's not a dataflow, you won't see an improvement. There will be no performance increase, since in some big data scenarios, the initial read from a data source would be slower because the data needs to be passed to the enhanced compute engine.

Question: I cannot see the enhanced compute engine toggle. Why?

Answer: The enhanced compute engine is being released in stages to regions around the world, but is not yet available in every region.

Question: What are the supported data types for the compute engine?

Answer: The enhanced compute engine and dataflows currently support the following data types. If your dataflow doesn't use one of the following data types, an error occurs during refresh:

- Date/Time
- Decimal Number

- Text
- Whole number
- Date/Time/Zone
- True/False
- Date
- Time

Use DirectQuery with dataflows in Power BI

You can use DirectQuery to connect directly to dataflows, and thereby connect directly to your dataflow without having to import its data.

Using DirectQuery with dataflows enables the following enhancements to your Power BI and dataflows processes:

- **Avoid separate refresh schedules** - DirectQuery connects directly to a dataflow, removing the need to create an imported dataset. As such, using DirectQuery with your dataflows means you no longer need separate refresh schedules for the dataflow and the dataset to ensure your data is synchronized.
- **Filtering data** - DirectQuery is useful for working on a filtered view of data inside a dataflow. If you want to filter data, and thereby work with a smaller subset of the data in your dataflow, you can use DirectQuery (and the compute engine) to filter dataflow data and work with the filtered subset you need.

Using DirectQuery for dataflows

Using DirectQuery with dataflows is available in Power BI Desktop.

There are also prerequisites for using DirectQuery with dataflows:

- Your dataflow must reside within a Power BI Premium enabled workspace
- The **compute engine** must be turned on

You can learn more about DirectQuery with dataflows in the [using DirectQuery with dataflows](#) article.

Enable DirectQuery for dataflows

To ensure your dataflow is available for DirectQuery access, the enhanced compute engine must be in its optimized state. To enable DirectQuery for dataflows, set the new **Enhanced compute engine settings** option to **On**. The following image shows the setting properly selected.

Enhanced compute engine settings (preview)
Configure enhanced compute engine settings for this dataflow.

Disabled
Turn off the enhanced compute engine for this dataflow.

Optimized
We'll turn on the enhanced compute engine only when this dataflow is linked to another one, which will enhance performance.

On
Turn on the enhanced compute engine for this dataflow.

Once you've applied that setting, refresh the dataflow for the optimization to take effect.

Considerations and limitations for DirectQuery

There are a few known limitations with DirectQuery and dataflows:

- Composite/mixed models that have import and DirectQuery data sources are currently not supported.
- Large dataflows may have trouble with timeout issues when viewing visualizations. Large dataflows that run into trouble with timeout issues should use Import mode.

- Under data source settings, the dataflow connector will show invalid credentials if you are using DirectQuery. This does not affect the behavior, and the dataset will work properly.

Computed entities

You can perform **in-storage computations** when using dataflows with a Power BI Premium subscription. This lets you perform calculations on your existing dataflows, and return results that enable you to focus on report creation and analytics.

The screenshot shows the 'Edit queries' interface in Power BI. On the left, there's a list of entities: 'Account', 'ServiceCalls', and 'ServiceCallsAggregated' (which is highlighted with a yellow box and a red border). On the right, there's a formula bar with 'fx' and a tooltip message: 'Computed entities require Premium to refresh. To enable refresh, switch to Premium license level.' Below the formula bar is a table view with two columns: 'accountid' and '1.2 numberOfServiceC...'. The table contains five rows of data:

	accountid	1.2 numberOfServiceC...
1	0011r00001mv6joAAA	87
2	0011r00001mv6jpAAA	87
3	0011r00001mv6jqAAA	96
4	0011r00001mv6jrAAA	87
5	0011r00001mv6jsAAA	87

To perform in-storage computations, you first must create the dataflow and bring data into that Power BI dataflow storage. Once you have a dataflow that contains data, you can create computed entities, which are entities that perform in-storage computations.

Considerations and limitations of computed entities

- When working with dataflows created in an organization's Azure Data Lake Storage Gen2 account, linked entities and computed entities only work properly when the entities reside in the same storage account.

As a best practice, when doing computations on data joined by on-premises and cloud data, create a new dataflow for each source (one for on-premises and one for cloud) and then create a third dataflow to merge/compute over these two data sources.

Linked entities

You can reference existing dataflows when using with a Power BI Premium subscription, which lets you either perform calculation on these entities using computed entities or allows you to create a "single source of the truth" table that you can reuse within multiple dataflows.

Incremental refresh

Dataflows can be set to refresh incrementally to avoid having to pull all the data on every refresh. To do so, select the dataflow then select the incremental refresh icon.

ENTITY NAME	ENTITY TYPE	ACTIONS
Table	Custom	Incremental Refresh

Setting incremental refresh adds parameters to the dataflow to specify the date range. For detailed information on how to set up incremental refresh, see the [incremental refresh](#) article.

Considerations for when not to set incremental refresh

Do not set a dataflow to incremental refresh in the following situations:

- Linked entities should not use incremental refresh if they reference a dataflow.

Next steps

The following articles provide more information about dataflows and Power BI:

[Dataflows best practices](#)

[Configure Power BI Premium dataflow workloads](#)

[Introduction to dataflows and self-service data prep](#)

[Creating a dataflow](#)

[Configure and consume a dataflow](#)

[Configuring Dataflow storage to use Azure Data Lake Gen 2](#)

[AI with dataflows](#)

[Dataflows considerations and limitations](#)

AI with dataflows

3/30/2022 • 27 minutes to read • [Edit Online](#)

In this article we discuss ways you can use artificial intelligence (AI) with dataflows. The areas described in this article are the following:

- Cognitive Services
- Automated Machine Learning
- Azure Machine Learning Integration

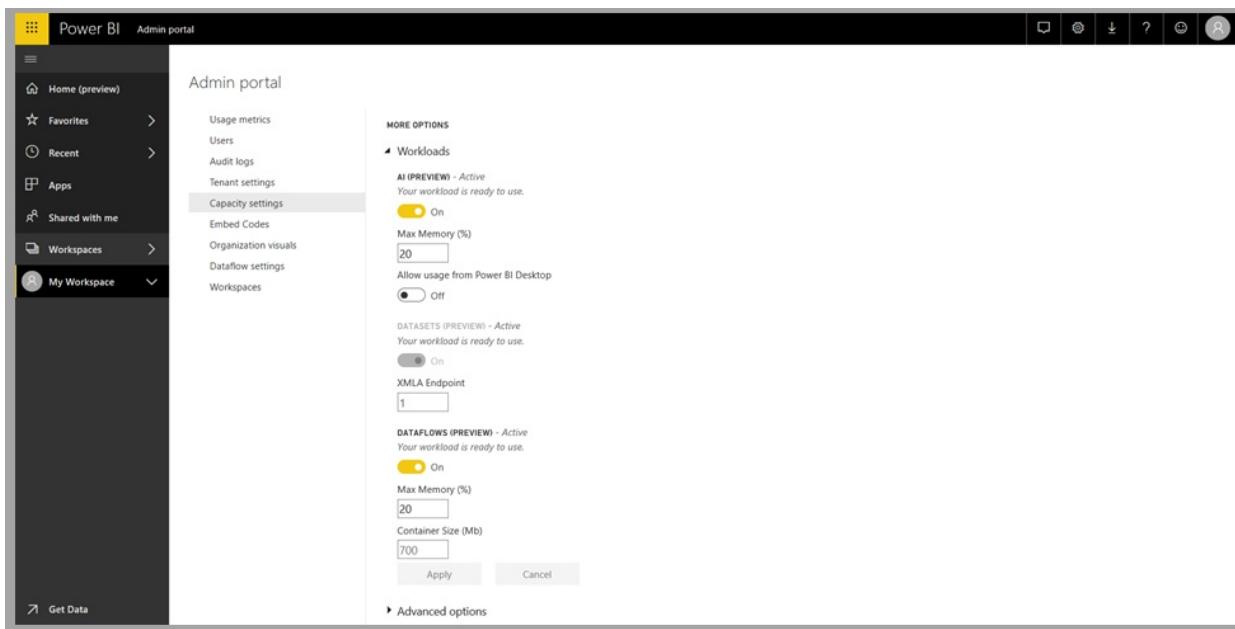
Cognitive Services in Power BI

With Cognitive Services in Power BI, you can apply different algorithms from [Azure Cognitive Services](#) to enrich your data in the self-service data prep for Dataflows.

The services that are supported today are [Sentiment Analysis](#), [Key Phrase Extraction](#), [Language Detection](#), and [Image Tagging](#). The transformations are executed on the Power BI Service and do not require an Azure Cognitive Services subscription. This feature requires Power BI Premium.

Enabling AI features

Cognitive services are supported for Premium capacity nodes EM2, A2, or P1 and above. Cognitive services are also available with a Premium Per User (PPU) license. A separate AI workload on the capacity is used to run cognitive services. Before using cognitive services in Power BI, the AI workload needs to be enabled in the capacity settings of the admin portal. You can turn on the AI workload in the workloads section, and define the maximum amount of memory you would like this workload to consume. The recommended memory limit is 20%. Exceeding this limit causes the query to slow down.



Getting started with Cognitive Services in Power BI

Cognitive Services transforms are part of the [Self-Service Data Prep for dataflows](#). To enrich your data with Cognitive Services, start by editing a dataflow.

Entities Machine learning models

ENTITY NAME	ENTITY TYPE	ACTIONS
Sales	Custom	
CRM Customers and Leads	Custom	
Hotel Reviews	Custom	
Service Calls	Custom	

Select the **AI Insights** button in the top ribbon of Power Query Editor.

Edit queries

Power Query

Get data Refresh Options Manage columns Transform table Reduce rows Add column AI Insights Map to standard Combine tables

AI insights [4]

- Sales
- CRM Customers an...
- Hotel Reviews**
- Service Calls

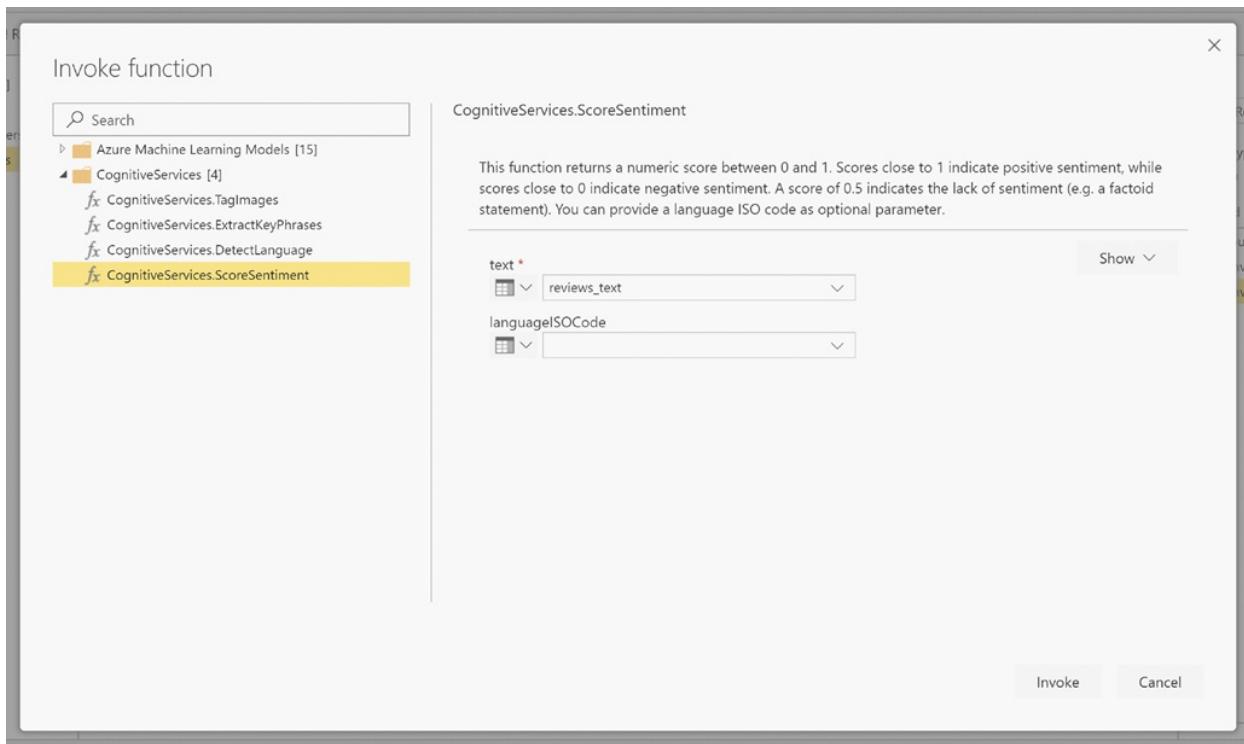
Navigation[[Schema = "dbo", Item = "Hotel Reviews"]][Data]

#	category	city	country	latitude	longitude	name
1	Hotels	Princeville	US	22.226	-159.481	Hotel 2
2	Hotels	Princeville	US	22.226	-159.481	Hotel 2
3	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
4	Hotels	Princeville	US	22.226	-159.481	Hotel 2
5	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
6	Hotels	Kapaa	US	22.043	-159.338	Hotel 5
7	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
8	Hotels	Princeville	US	22.226	-159.481	Hotel 2
9	Hotels	Princeville	US	22.226	-159.481	Hotel 2
10	Hotels	Princeville	US	22.226	-159.481	Hotel 2
11	Hotels	Princeville	US	22.226	-159.481	Hotel 2
12	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
13	Hotels	Princeville	US	22.226	-159.481	Hotel 2
14	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
15	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
16	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
17	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
18	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
19	Hotels	Princeville	US	22.226	-159.481	Hotel 2
20	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
21	Hotels	Kapaa	US	22.043	-159.338	Hotel 5
22	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4
23	Vacation Rentals,Resorts &...	Honolulu	US	21.282	-157.831	Hotel 4

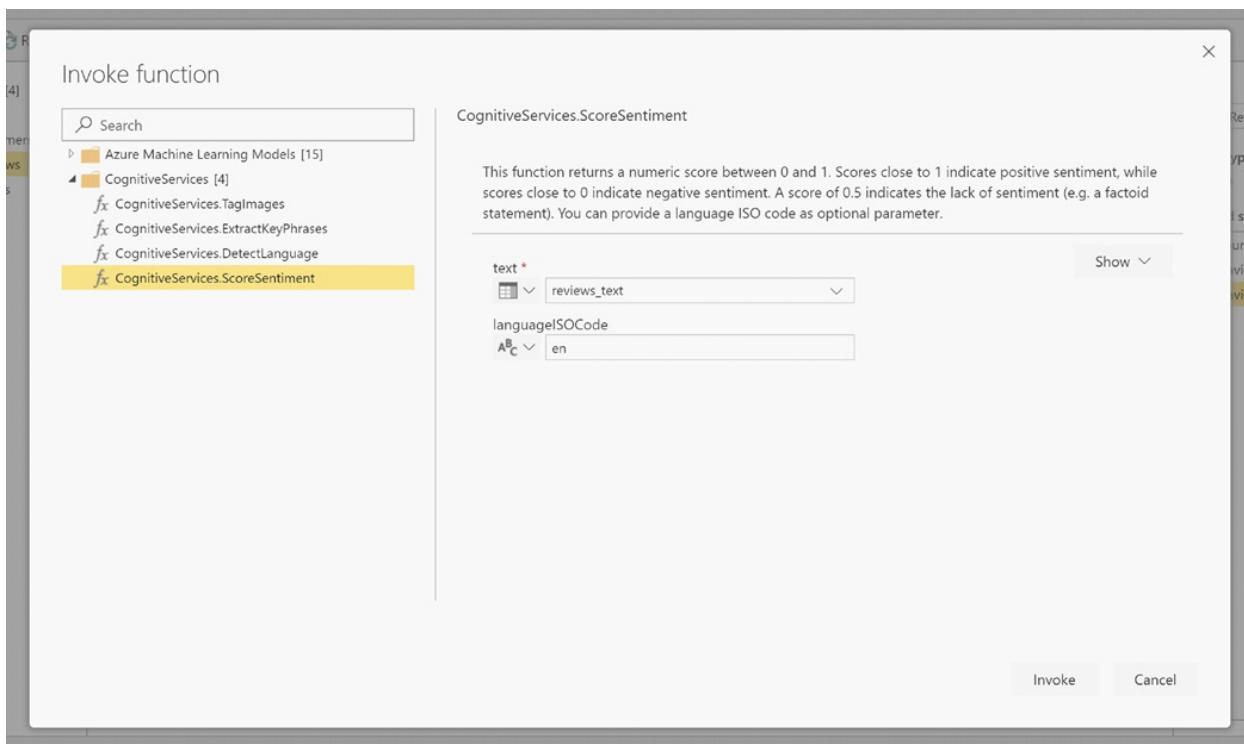
Name: Hotel Reviews
Entity type: Custom
Applied steps:
Source
Navigation
Navigation 1

Cancel Done

In the pop-up window, select the function you want to use and the data you want to transform. In this example, I'm scoring the sentiment of a column that contains review text.



Cultureinfo is an optional input to specify the language of the text. This column is expecting an ISO code. You can use a column as input for Cultureinfo, or a static column. In this example, the language is specified as English (en) for the whole column. If you leave this column blank, Power BI automatically detects the language before applying the function. Next, select **Invoke**.



After invoking the function, the result is added as a new column to the table. The transformation is also added as an applied step in the query.

The screenshot shows the 'Edit queries' interface in Power BI. A specific step in the query editor is highlighted: `Table.AddColumn(#"Navigation 1", "CognitiveServices.ScoreSentiment", each CognitiveServices.ScoreSentiment([reviews_text], "en"))`. A dropdown menu is open over the 'CognitiveServices.ScoreSentiment' function, listing four output columns: `reviews_text`, `reviews_title`, `Image`, and `CognitiveServices`. The 'Applied steps' pane on the right shows that the 'Invoked CognitiveServices.ScoreSentiment' step has been added.

If the function returns multiple output columns, invoking the function adds a new column with a row of the multiple output columns.

Use the expand option to add one or both values as columns to your data.

The screenshot shows the 'Edit queries' interface in Power BI. A specific step in the query editor is highlighted: `Table.AddColumn(#"Invoked CognitiveServices.ScoreSentiment", "CognitiveServices.DetectLanguage", each CognitiveServices.DetectLanguage([reviews_text]))`. A dropdown menu is open over the 'CognitiveServices.DetectLanguage' function, listing three options: 'Select all', 'Detected Language Name', and 'Detected Language ISO Code'. The 'OK' button is highlighted at the bottom of the dropdown.

Available functions

This section describes the available functions in Cognitive Services in Power BI.

Detect Language

The language detection function evaluates text input, and for each column, returns the language name and ISO identifier. This function is useful for data columns that collect arbitrary text, where language is unknown. The function expects data in text format as input.

Text Analytics recognizes up to 120 languages. For more information, see [supported languages](#).

Extract Key Phrases

The **Key Phrase Extraction** function evaluates unstructured text, and for each text column, returns a list of key phrases. The function requires a text column as input, and accepts an optional input for **Cultureinfo**. (See the [Getting Started](#) section earlier in this article).

Key phrase extraction works best when you give it bigger chunks of text to work on. This is opposite from sentiment analysis, which performs better on smaller blocks of text. To get the best results from both operations, consider restructuring the inputs accordingly.

Score Sentiment

The **Score Sentiment** function evaluates text input and returns a sentiment score for each document, ranging from 0 (negative) to 1 (positive). This function is useful for detecting positive and negative sentiment in social media, customer reviews, and discussion forums.

Text Analytics uses a machine learning classification algorithm to generate a sentiment score between 0 and 1. Scores closer to 1 indicate positive sentiment, scores closer to 0 indicate negative sentiment. The model is pre-trained with an extensive body of text with sentiment associations. Currently, it's not possible to provide your own training data. The model uses a combination of techniques during text analysis, including text processing, part-of-speech analysis, word placement, and word associations. For more information about the algorithm, see [Introducing Text Analytics](#).

Sentiment analysis is performed on the entire input column, as opposed to extracting sentiment for a particular table in the text. In practice, there's a tendency for scoring accuracy to improve when documents contain one or two sentences rather than a large block of text. During an objectivity assessment phase, the model determines whether an input column as a whole is objective or contains sentiment. An input column that is mostly objective does not progress to the sentiment detection phrase, resulting in a .50 score, with no further processing. For input columns continuing in the pipeline, the next phase generates a score above or below .50, depending on the degree of sentiment detected in the input column.

Currently, Sentiment Analysis supports English, German, Spanish, and French. Other languages are in preview. For more information, see [Supported languages](#).

Tag Images

The **Tag Images** function returns tags based on more than 2,000 recognizable objects, living beings, scenery, and actions. When tags are ambiguous or not common knowledge, the output provides 'hints' to clarify the meaning of the tag in context of a known setting. Tags are not organized as a taxonomy and no inheritance hierarchies exist. A collection of content tags forms the foundation for an image 'description' displayed as human readable language formatted in complete sentences.

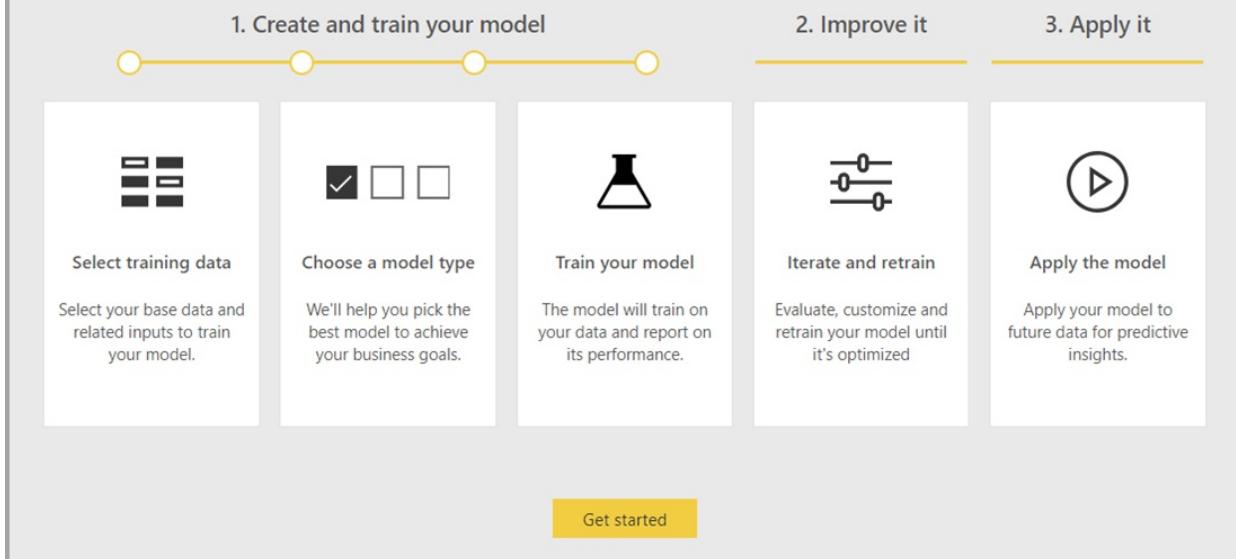
After uploading an image or specifying an image URL, Computer Vision algorithms output tags based on the objects, living beings, and actions identified in the image. Tagging is not limited to the main subject, such as a person in the foreground, but also includes the setting (indoor or outdoor), furniture, tools, plants, animals, accessories, gadgets, and so on.

This function requires an image URL or a base-64 column as input. At this time, image tagging supports English, Spanish, Japanese, Portuguese, and Simplified Chinese. For more information, see [Supported languages](#).

Automated Machine Learning in Power BI

Automated machine learning (AutoML) for dataflows enables business analysts to train, validate, and invoke Machine Learning (ML) models directly in Power BI. It includes a simple experience for creating a new ML model where analysts can use their dataflows to specify the input data for training the model. The service automatically extracts the most relevant features, selects an appropriate algorithm, and tunes and validates the ML model. After a model is trained, Power BI automatically generates a performance report that includes the results of the validation. The model can then be invoked on any new or updated data within the dataflow.

New to machine learning models? Here's what you'll be doing:



Automated machine learning is available for dataflows that are hosted on Power BI Premium and Embedded capacities only.

Working with AutoML

Dataflows offer self-serve data prep for big data. AutoML is integrated into dataflows and enables you to leverage your data prep effort for building machine learning models, right within Power BI.

AutoML in Power BI enables data analysts to use dataflows to build machine learning models with a simplified experience, using just Power BI skills. Most of the data science behind the creation of the ML models is automated by Power BI. It has guardrails to ensure that the model produced has good quality and provides visibility into the process used to create your ML model.

AutoML supports the creation of **Binary Prediction**, **Classification**, and **Regression Models** for dataflows. These are types of supervised machine learning techniques, which means that they learn from the known outcomes of past observations to predict the outcomes of other observations. The input dataset for training an AutoML model is a set of rows that are **labeled** with the known outcomes.

AutoML in Power BI integrates [automated ML](#) from [Azure Machine Learning](#) to create your ML models. However, you don't need an Azure subscription to use AutoML in Power BI. The process of training and hosting the ML models is managed entirely by the Power BI service.

After an ML model is trained, AutoML automatically generates a Power BI report that explains the likely performance of your ML model. AutoML emphasizes explainability by highlighting the key influencers among your inputs that influence the predictions returned by your model. The report also includes key metrics for the model.

Other pages of the generated report show the statistical summary of the model and the training details. The statistical summary is of interest to users who would like to see the standard data science measures of model performance. The training details summarize all the iterations that were run to create your model, with the associated modeling parameters. It also describes how each input was used to create the ML model.

You can then apply your ML model to your data for scoring. When the dataflow is refreshed, your data is updated with predictions from your ML model. Power BI also includes an individualized explanation for each specific prediction that the ML model produces.

Creating a machine learning model

This section describes how to create an AutoML model.

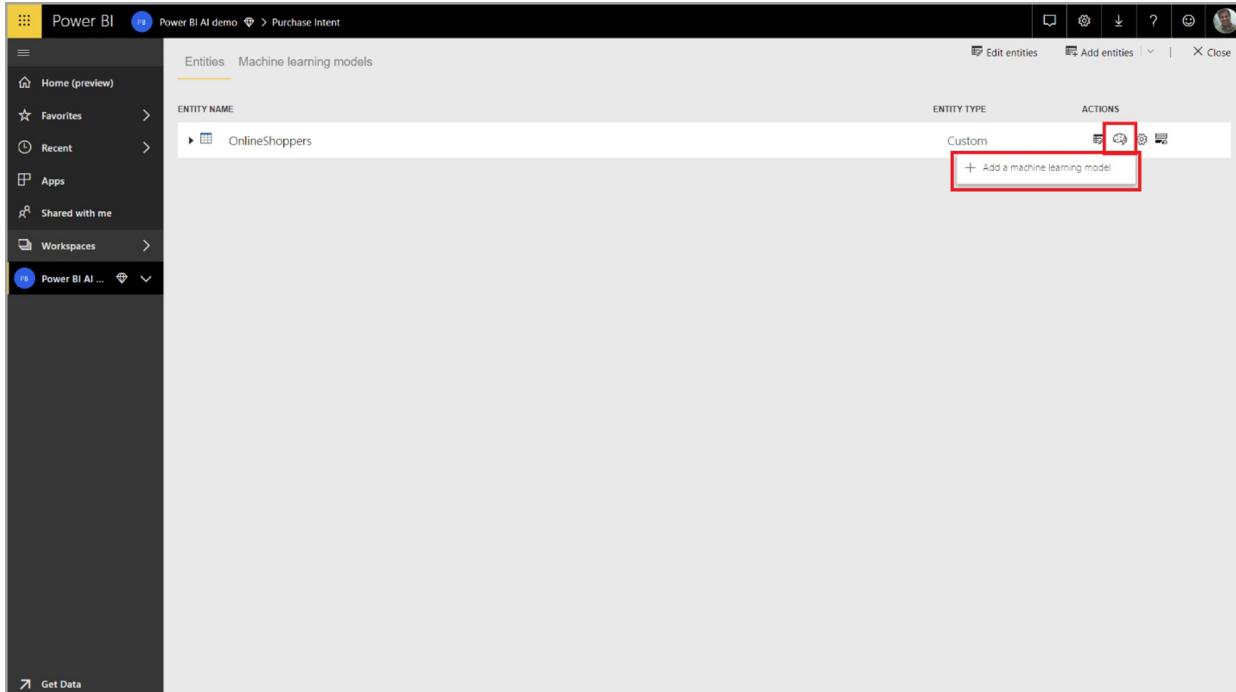
Data prep for creating an ML model

To create a machine learning model in Power BI, you must first create a dataflow for the data containing the historical outcome information, which is used for training the ML model. You should also add calculated columns for any business metrics that may be strong predictors for the outcome you're trying to predict. For details on configuring your dataflow, see [configure and consume a dataflow](#).

AutoML has specific data requirements for training a machine learning model. These requirements are described in sections below, based on respective model types.

Configuring the ML model inputs

To create an AutoML model, select the ML icon in the **Actions** column of the dataflow table, and select **Add a machine learning model**.

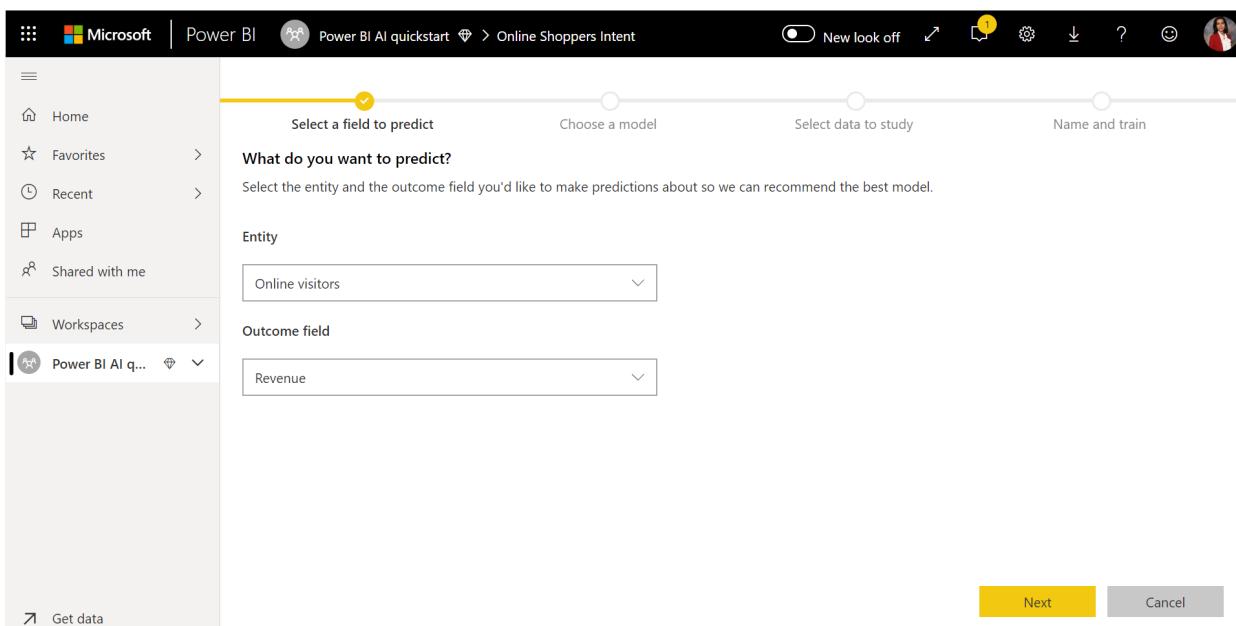


The screenshot shows the Power BI Entities screen. The 'Machine learning models' tab is selected. In the main area, there is a table with columns: ENTITY NAME, ENTITY TYPE, and ACTIONS. One row in the table is highlighted with a red box around the 'Actions' column, specifically around the '+ Add a machine learning model' button.

A simplified experience is launched, consisting of a wizard that guides you through the process of creating the ML model. The wizard includes the following simple steps.

1. Select the table with the historical data, and the outcome column for which you want a prediction

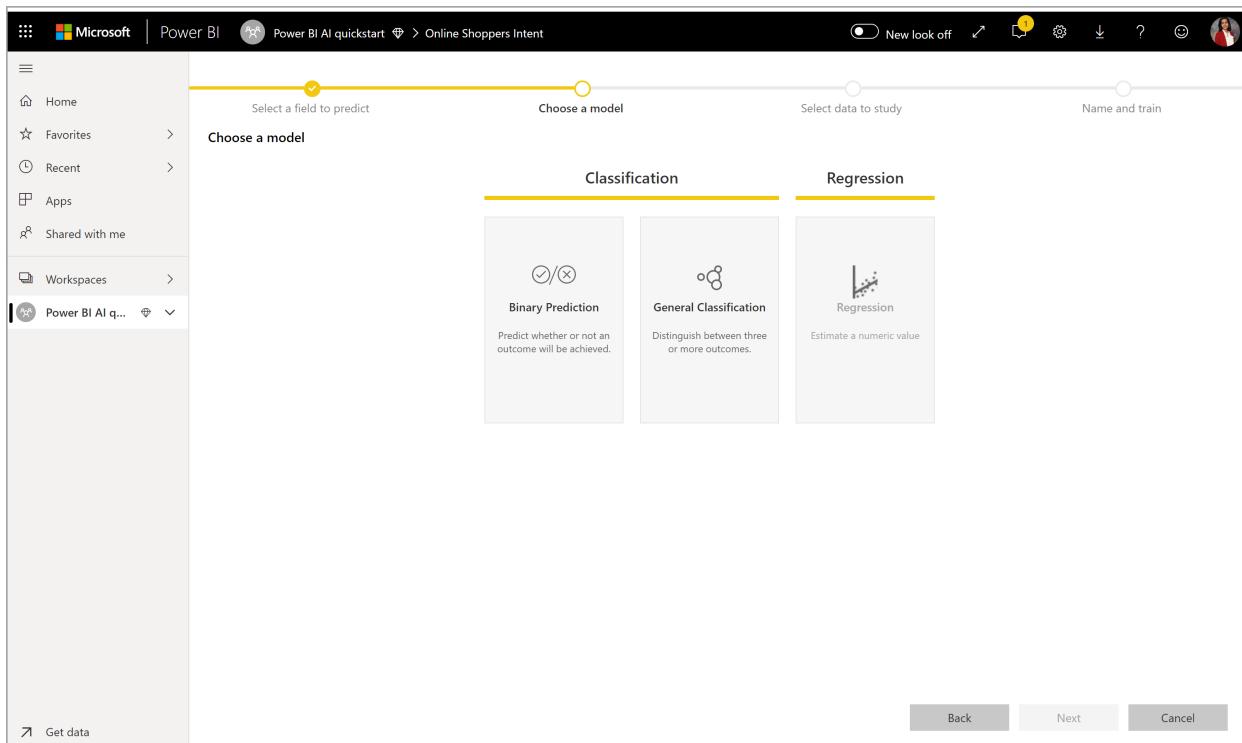
The outcome column identifies the label attribute for training the ML model, shown in the following image.



The screenshot shows the 'Select a field to predict' step of the Power BI AI quickstart wizard. On the left, there's a sidebar with navigation links like Home, Favorites, Recent, Apps, Shared with me, Workspaces, and a Power BI AI quickstart link. The main area has four circular progress steps: 'Select a field to predict' (yellow), 'Choose a model' (white), 'Select data to study' (white), and 'Name and train' (white). Below the steps, there's a section titled 'What do you want to predict?' with the instruction 'Select the entity and the outcome field you'd like to make predictions about so we can recommend the best model.' There are two dropdown menus: 'Entity' (set to 'Online visitors') and 'Outcome field' (set to 'Revenue'). At the bottom right, there are 'Next' and 'Cancel' buttons.

2. Choose a model type

When you specify the outcome column, AutoML analyzes the label data to recommend the most likely ML model type that can be trained. You can pick a different model type as shown below by clicking on "Select a different model".



NOTE

Some model types may not be supported for the data that you have selected and hence would be disabled. In the above example, Regression is disabled, as a text column is selected as outcome column.

3. Select the inputs you want the model to use as predictive signals

AutoML analyzes a sample of the selected table to suggest the inputs that can be used for training the ML model. Explanations would be provided next to columns that are not selected. If a particular column has too many distinct values or only one value, or low or high correlation with the output column, it would not be recommended.

Any inputs that are dependent on the outcome column (or the label column) should not be used for training the ML model, since they will affect its performance. Such columns would be flagged as having "suspiciously high correlation with output column". Introducing these columns into the training data causes label leakage, where the model performs well on the validation or test data but cannot match that performance when used in production for scoring. Label leakage could be a possible concern in AutoML models, when training model performance is too good to be true.

This feature recommendation is based on a sample of a data, so you should review the inputs used. You have the option to change the selections to include only the columns you want the model to study. You can also select all the columns by selecting the checkbox next to the table name.

4. Name your model and save your configuration

In the final step, you can name the model and select Save and train which begins training the ML model. You can choose to reduce the training time to see quick results or increase the amount of time spent in training to get the best model.

ML model training

Training of AutoML models is a part of the dataflow refresh. AutoML first prepares your data for training. AutoML splits the historical data you provide into training and testing datasets. The test dataset is a holdout set that is used for validating the model performance after training. These are realized as **Training** and **Testing** tables in the dataflow. AutoML uses cross-validation for the model validation.

Next, each input column is analyzed and imputation is applied, which replaces any missing values with substituted values. A couple of different imputation strategies are used by AutoML. For input attributes treated as numeric features, the mean of the column values is used for imputation. For input attributes treated as categorical features, AutoML uses the mode of the column values for imputation. The mean and mode of values used for imputation are calculated by the AutoML framework on the subsampled training dataset.

Then, sampling and normalization are applied to your data as required. For classification models, AutoML runs the input data through stratified sampling and balances the classes to ensure the row counts are equal for all.

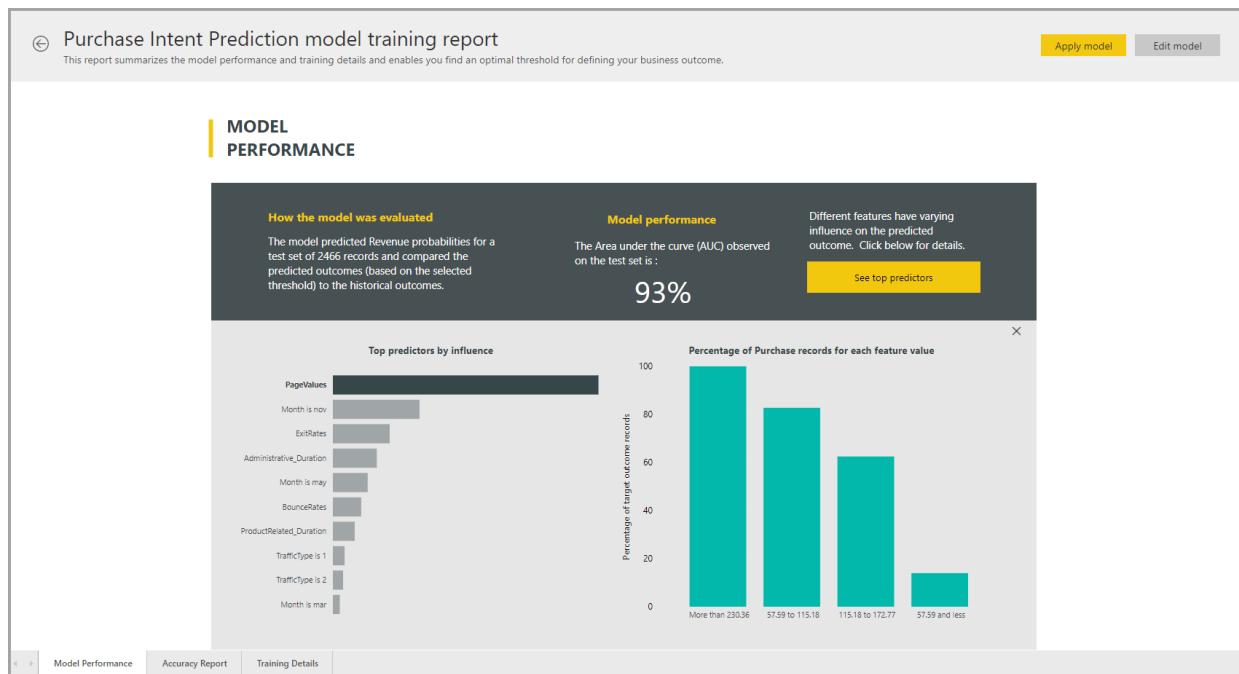
AutoML applies several transformations on each selected input column based on its data type, and its statistical properties. AutoML uses these transformations to extract features for use in training your ML model.

The training process for AutoML models consists of up to 50 iterations with different modeling algorithms and hyperparameter settings to find the model with the best performance. Training can end early with lesser iterations if AutoML notices that there is no performance improvement being observed. The performance of each of these models is assessed by validation with the holdout test dataset. During this training step, AutoML creates several pipelines for training and validation of these iterations. The process of assessing the performance of the models can take time, anywhere from several minutes to a couple of hours up-to the training time configured in the wizard, depending on the size of your dataset and the capacity resources available.

In some cases, the final model generated may use ensemble learning, where multiple models are used to deliver better predictive performance.

AutoML model explainability

After the model has been trained, AutoML analyzes the relationship between the input features and the model output. It assesses the magnitude of change to the model output for the holdout test dataset for each input feature. This is known as the *feature importance*. This happens as a part of the refresh once training is complete. Hence your refresh may take longer than the training time configured in the wizard.



AutoML model report

AutoML generates a Power BI report that summarizes the performance of the model during validation, along with the global feature importance. This report can be accessed from the Machine Learning Model tab once the dataflow refresh is successful. The report summarizes the results from applying the ML model to the holdout test data and comparing the predictions with the known outcome values.

You can review the model report to understand its performance. You can also validate that the key influencers of the model align with the business insights about the known outcomes.

The charts and measures used to describe the model performance in the report depend on the model type. These performance charts and measures are described in the following sections.

Additional pages in the report may describe statistical measures about the model from a data science perspective. For instance, the **Binary Prediction** report includes a gain chart and the ROC curve for the model.

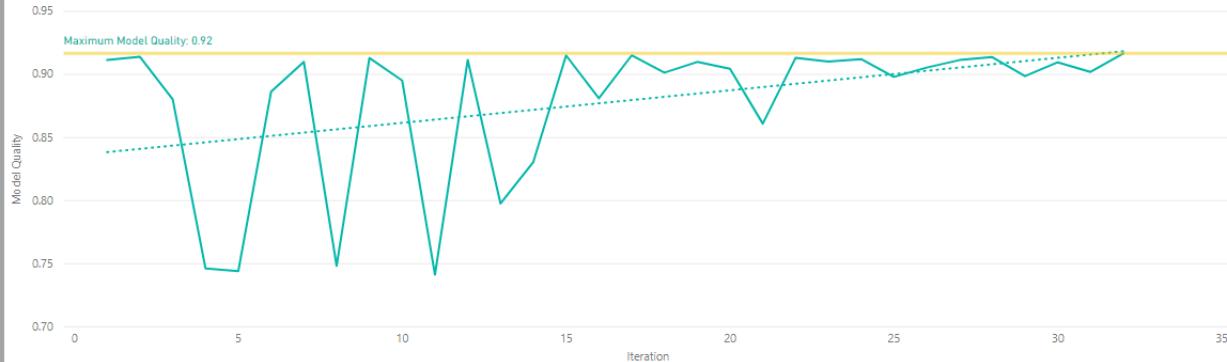
The reports also include a **Training Details** page that includes a description of how the model was trained, and a chart describing the model performance over each of the iterations run.

How the model was trained

Power BI used the automated ML capability in Azure Machine Learning to train your model. Automated ML was used to find the best way to prepare your data, determine the algorithms used and select the algorithm parameters likely to yield the best accuracy. These steps were used in the machine learning pipeline which generated your machine learning model.

Sampled rows	5504	Final model used	Pre-fitted Soft Voting Classifier
Training rows	3038	Iterations run	32

Model quality over iterations



Another section on this page describes the detected type of the input column and imputation method used for filling missing values. It also includes the parameters used by the final model.

Your machine learning model

The tables below contain the list of features extracted from the inputs you provided, and the final set of parameters that were used to create your machine learning model. This information can be used to recreate the machine learning model outside Power BI.

Data Featurization

Feature	Detected Column	Imputation Type
Informational	Categorical	
Month	Categorical	
OperatingSystems	Categorical	
TrafficType	Categorical	
Weekend	Categorical	
Administrative_Duration	Numeric	Mean
BounceRates	Numeric	Mean
ExitRates	Numeric	Mean
PageValues	Numeric	Mean
ProductRelated_Duration	Numeric	Mean
SpecialDay	Numeric	Mean

Pre-fitted Soft Voting Classifier final parameters selected

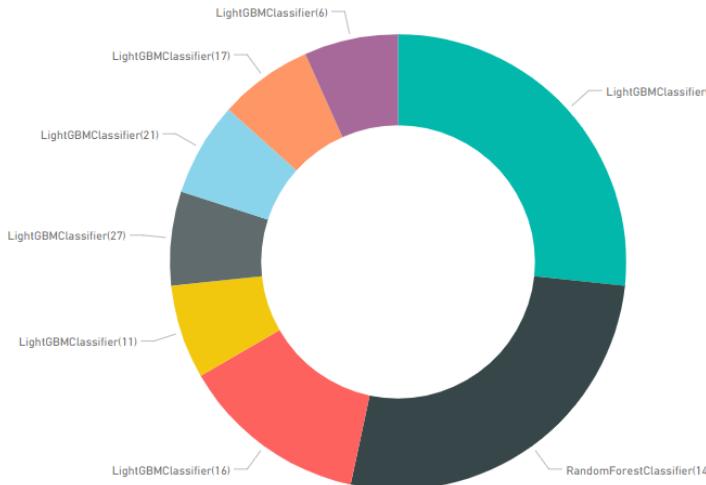
Parameter Name	Parameter Value
min_models	1
model_seed_threshold	0.05
max_models	15

If the model produced uses ensemble learning, then the **Training Details** page also includes a chart showing the weight of each constituent model in the ensemble, as well as its parameters.

Ensemble machine learning models

Ensemble models use multiple learning algorithms to obtain better predictive performance than may be obtained from a single learning algorithm. Ensemble models are useful for improving accuracy in certain cases.

Automated ML in Power BI generates ensemble models, if they are found to be optimal. If an ensemble model is used, then the constituent model details will be presented below.



Applying the AutoML model

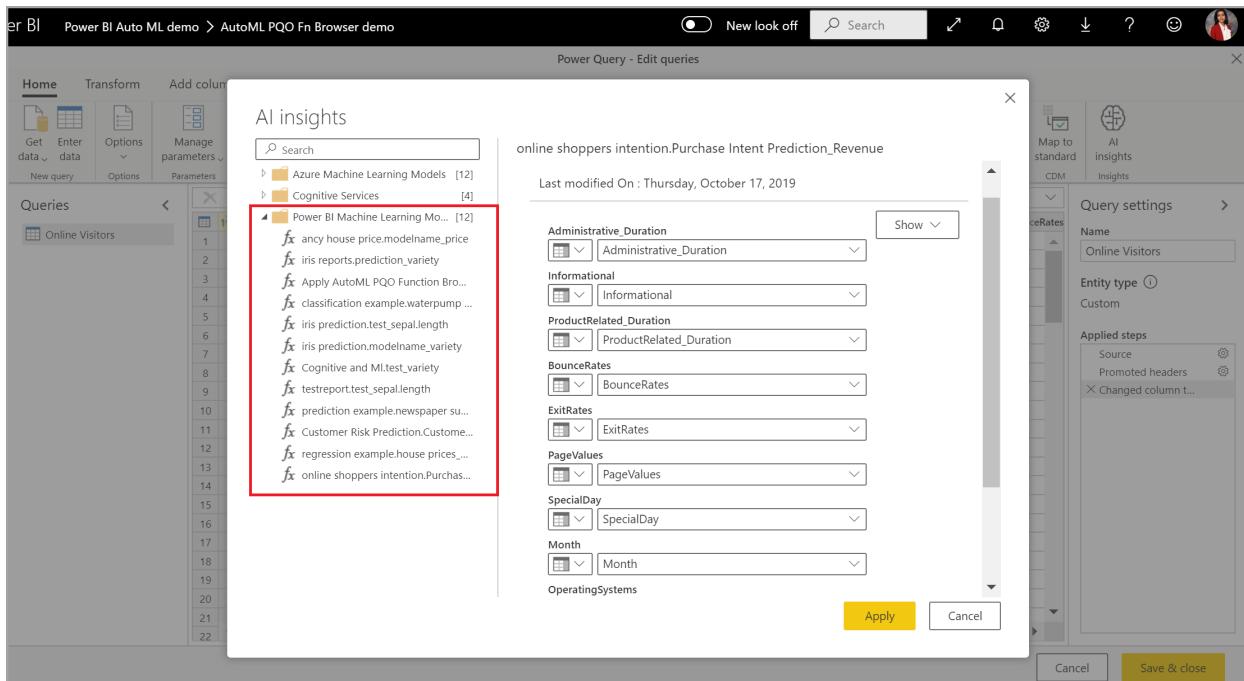
If you're satisfied with the performance of the ML model created, you can apply it to new or updated data when your dataflow is refreshed. You can do this from the model report, by selecting the **Apply** button in the top-right corner or the **Apply ML Model** button under actions in the Machine Learning Models tab.

To apply the ML model, you must specify the name of the table to which it must be applied, and a prefix for the columns that will be added to this table for the model output. The default prefix for the column names is the model name. The *Apply* function may include additional parameters specific to the model type.

Applying the ML model creates two new dataflow tables which contains the predictions and individualized explanations for each row that it scores in the output table. For instance, if you apply the *PurchaseIntent* model to the *OnlineShoppers* table, the output will generate the **OnlineShoppers enriched PurchaseIntent** and **OnlineShoppers enriched PurchaseIntent explanations** tables. For each row in the enriched table, The **Explanations** is broken down into multiple rows in the enriched explanations table based on the input feature. An **ExplanationIndex** helps map the rows from the enriched explanations table to the row in enriched table.

The screenshot shows the 'Edit queries' interface in Power Query. The main area displays a table with three columns: 'Purchase Intent Prediction....', 'Purchase Intent Prediction.PredictionScore', and 'Purchase Intent Prediction.Explanation...'. The 'Purchase Intent Prediction....' column contains rows 1 through 22, with values ranging from FALSE to TRUE. The 'Purchase Intent Prediction.PredictionScore' column also contains rows 1 through 22, with values such as '75.1401910139549', '60 "Month", "Categorical"', and '77 "Base Probability", "ExpectedValueType", "50.0658867995066...'. The 'Purchase Intent Prediction.Explanation...' column contains rows 1 through 22, with values like '88 "PageValues", "Numeric", "92.19429633232734", "", "PageValues..."'. The 'Applied steps' pane on the right lists various steps taken during the data flow, including 'Source', 'AddExplanationsIndex', 'Invoked Purchase Intent ...', 'DataflowPrecalculatedSo...', 'Workspace', 'Dataflow', 'EnrichedPreview', and 'Enriched results' (which is highlighted). At the bottom, there are 'Cancel' and 'Save & close' buttons.

You can also apply any Power BI AutoML model to tables in any dataflow in the same workspace using AI Insights in PQO function browser. This way, you can use models created by others in the same workspace without necessarily being an owner of the dataflow that has the model. Power Query discovers all the Power BI ML models in the workspace and exposes them as dynamic Power Query functions. You can invoke those functions by accessing them from the ribbon in Power Query Editor, or by invoking the M function directly. This functionality is currently only supported for Power BI dataflows, and for Power Query Online in the Power BI service. Note that this is very different from applying ML models within a dataflow using the AutoML wizard. There is no explanations table created using this method and unless you are the owner of the dataflow, you cannot access model training reports or retrain the model. If the source model is edited (adding or removing input columns) or, the model or source dataflow is deleted, then this dependent dataflow would break.



After you apply the model, AutoML always keeps your predictions up-to-date whenever the dataflow is refreshed.

To use the insights and predictions from the ML model in a Power BI report, you can connect to the output table from Power BI Desktop using the **dataflows** connector.

Binary Prediction models

Binary Prediction models, more formally known as **binary classification models**, are used to classify a dataset into two groups. They're used to predict events that can have a binary outcome. For instance, whether a sales opportunity will convert, whether an account will churn, whether an invoice will be paid on time, whether a transaction is fraudulent, and so on.

The output of a Binary Prediction model is a probability score, which identifies the likelihood that the target outcome will be achieved.

Training a Binary Prediction model

Pre-requisites:

- A minimum of 20 rows of historical data is required for each class of outcomes

The process of creation for a Binary Prediction model follows the same steps as other AutoML models, described in the section **Configuring the ML model inputs** above. The only difference is in the "Choose a model" step where you can select the target outcome value that you're most interested in. You can also provide friendly labels for the outcomes to be used in the automatically generated report that will summarize the results of the model validation.

Select a field to predict

Choose a model

Select data to study

Name and train

Choose a model

Based on the field you selected, we recommend a **Prediction** model. This model learns from your data to predict whether or not an outcome will be achieved. Not what you're looking for? [Select a different model](#)

Choose a target outcome
Enter or select the `status_group` outcome that you're most interested in.

()/()

Binary Prediction
Predict whether or not an outcome will be achieved.

functional

How should we label predictions in the model training report?

Match label
Enter the text you want to display when our prediction matches your target value.

functional

Mismatch label
Enter the text you want to display when our prediction does not match your target value.

Not functional

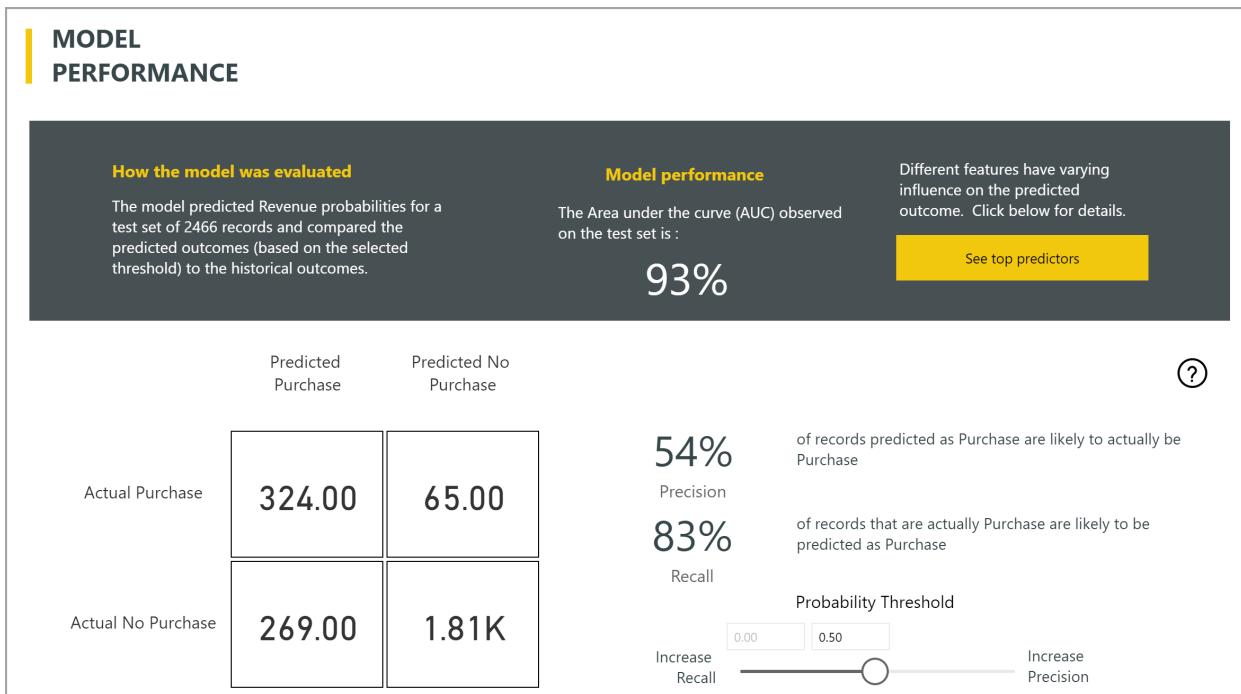
Back **Next** **Cancel**

Binary Prediction model report

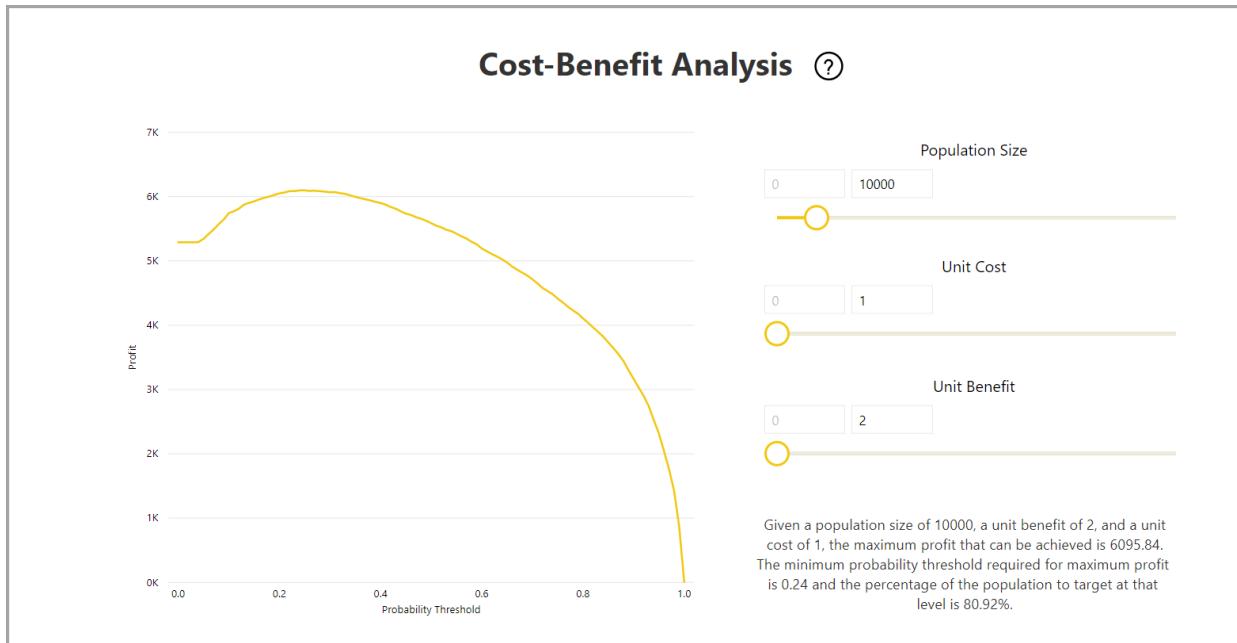
The Binary Prediction model produces as an output a probability that a row will achieve the target outcome. The report includes a slicer for the probability threshold, which influences how the scores above and below the probability threshold are interpreted.

The report describes the performance of the model in terms of *True Positives*, *False Positives*, *True Negatives*, and *False Negatives*. True Positives and True Negatives are correctly predicted outcomes for the two classes in the outcome data. False Positives are rows that were predicted to have Target outcome but actually did not. Conversely, False Negatives are rows that had Target outcome but were predicted as not having it.

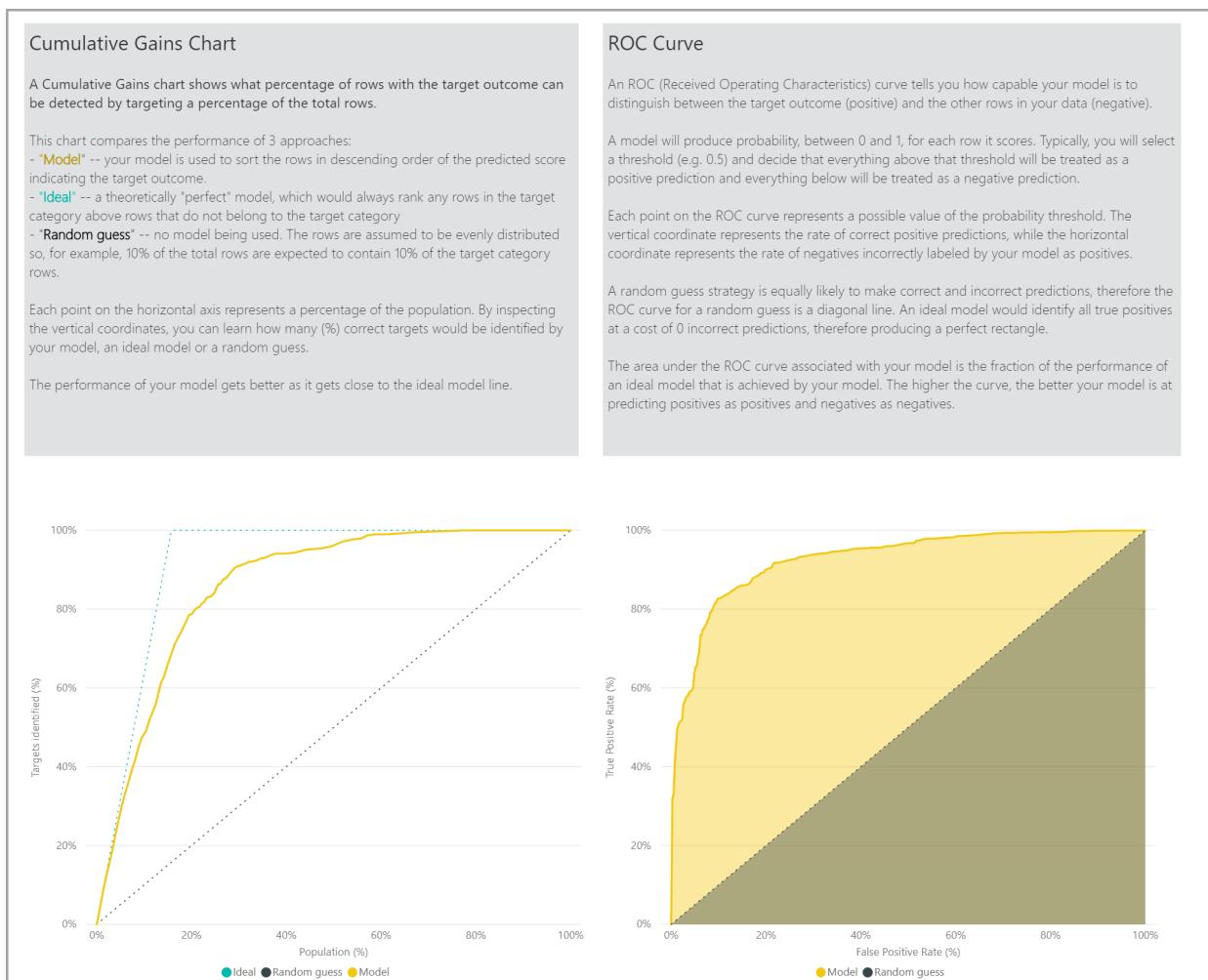
Measures, such as Precision and Recall, describe the effect of the probability threshold on the predicted outcomes. You can use the probability threshold slicer to select a threshold that achieves a balanced compromise between Precision and Recall.



The report also includes a Cost-Benefit analysis tool to help identify the subset of the population that should be targeted to yield the highest profit. Given an estimated unit cost of targeting and a unit benefit from achieving a target outcome, Cost-Benefit analysis attempts to maximize profit. You can use this tool to pick your probability threshold based on the maximum point in the graph to maximize profit. You can also use the graph to compute the profit or cost for your choice of probability threshold.

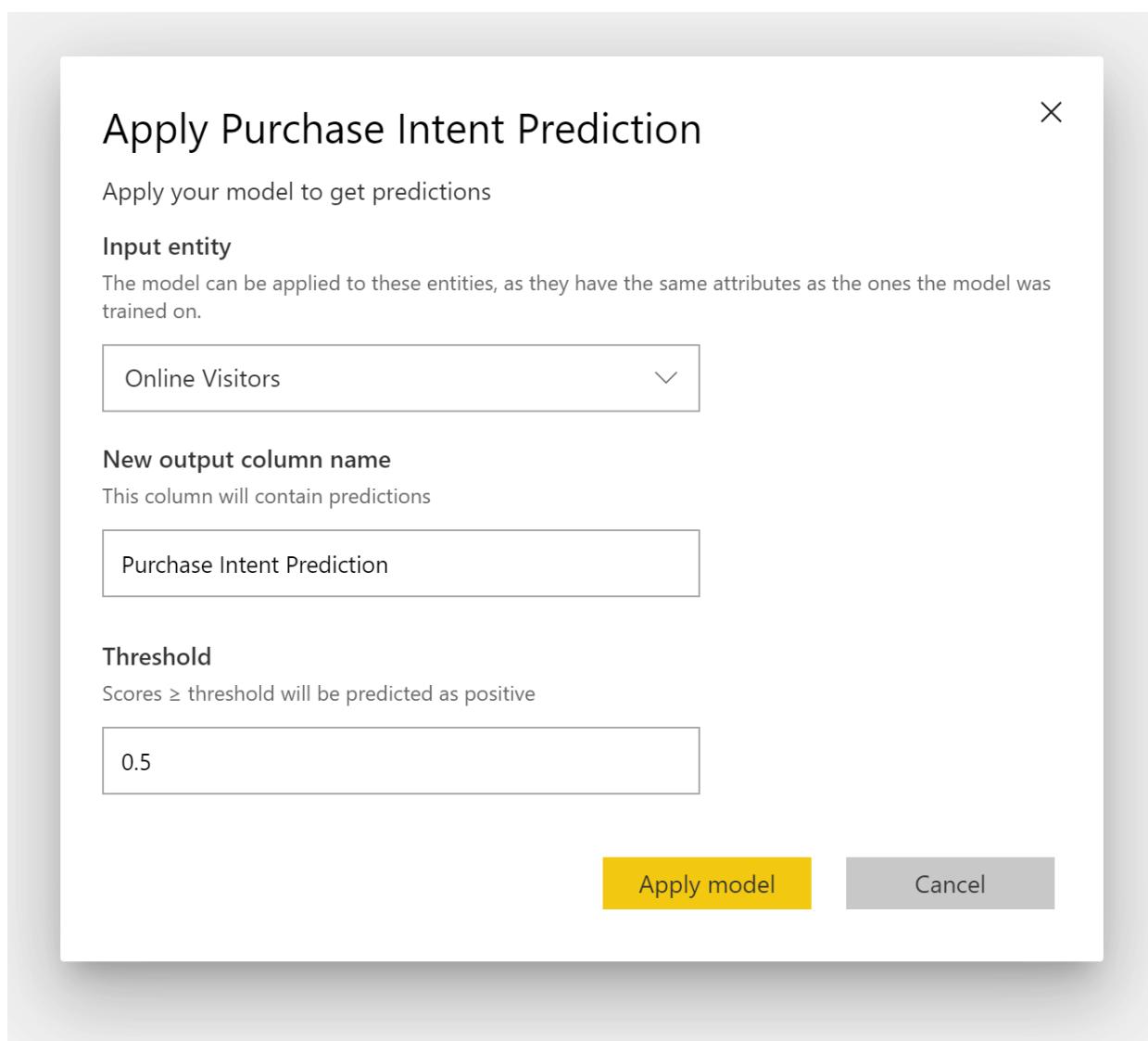


The **Accuracy Report** page of the model report includes the *Cumulative Gains* chart and the ROC curve for the model. These are statistical measures of model performance. The reports include descriptions of the charts shown.



Applying a Binary Prediction model

To apply a Binary Prediction model, you must specify the table with the data to which you want to apply the predictions from the ML model. Other parameters include the output column name prefix and the probability threshold for classifying the predicted outcome.



When a Binary Prediction model is applied, it adds four output columns to the enriched output table: **Outcome**, **PredictionScore**, **PredictionExplanation**, and **ExplanationIndex**. The column names in the table have the prefix specified when the model is applied.

PredictionScore is a percentage probability, which identifies the likelihood that the target outcome will be achieved.

The **Outcome** column contains the predicted outcome label. Records with probabilities exceeding the threshold are predicted as likely to achieve the target outcome and are labeled as True. Records below the threshold are predicted as unlikely to achieve the outcome and are labeled as False.

The **PredictionExplanation** column contains an explanation with the specific influence that the input features had on the **PredictionScore**.

Classification models

Classification models are used to classify a dataset into multiple groups or classes. They're used to predict events that can have one of the multiple possible outcomes. For instance, whether a customer is likely to have a very high, high, medium, or low Lifetime Value, whether the risk of default is High, Moderate, Low, or Very Low; and so on.

The output of a Classification model is a probability score, which identifies the likelihood that a row will achieve

the criteria for a given class.

Training a Classification model

The input table containing your training data for a Classification model must have a string or whole number column as the outcome column, which identifies the past known outcomes.

Pre-requisites:

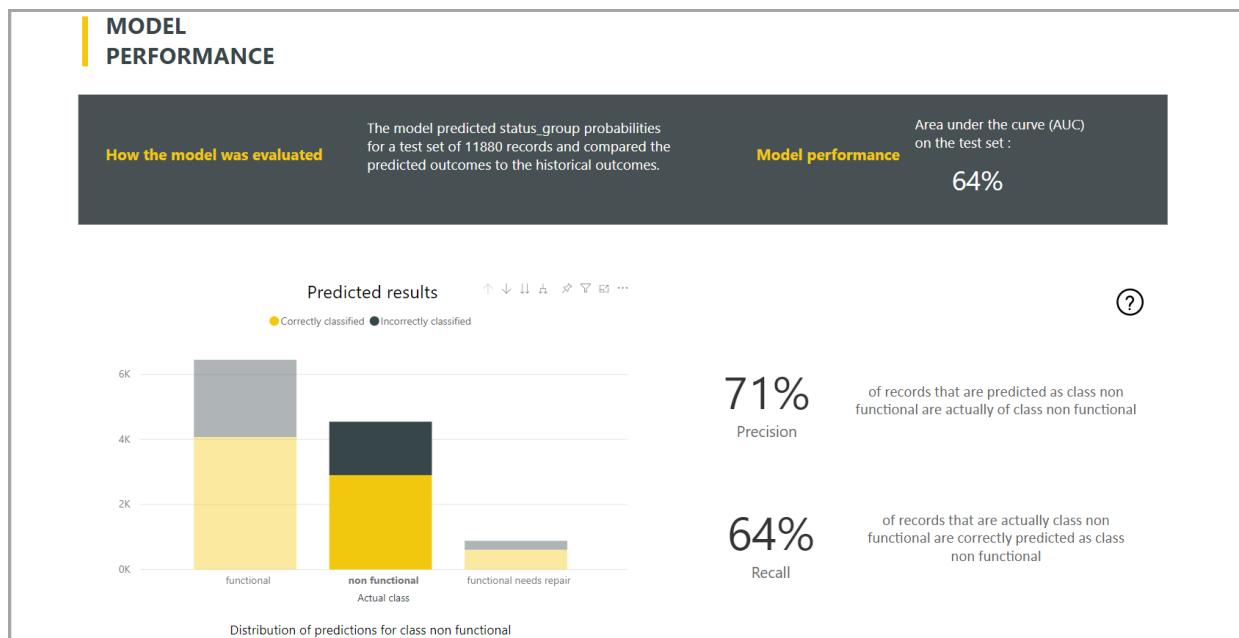
- A minimum of 20 rows of historical data is required for each class of outcomes

The process of creation for a Classification model follows the same steps as other AutoML models, described in the section **Configuring the ML model inputs** above.

Classification model report

The Classification model report is produced by applying the ML model to the holdout test data and comparing the predicted class for a row with the actual known class.

The model report includes a chart that includes the breakdown of the correctly and incorrectly classified rows for each known class.



A further class-specific drill-down action enables an analysis of how the predictions for a known class are distributed. This shows the other classes in which rows of that known class are likely to be misclassified.

The model explanation in the report also includes the top predictors for each class.

The Classification model report also includes a Training Details page similar to the pages for other model types, as described in the section **AutoML model report** earlier in this article.

Applying a classification model

To apply a Classification ML model, you must specify the table with the input data and the output column name prefix.

When a Classification model is applied, it adds five output columns to the enriched output table:

ClassificationScore, **ClassificationResult**, **ClassificationExplanation**, **ClassProbabilities**, and **ExplanationIndex**. The column names in the table have the prefix specified when the model is applied.

The **ClassProbabilities** column contains the list of probability scores for the row for each possible class.

The **ClassificationScore** is the percentage probability, which identifies the likelihood that a row will achieve the criteria for a given class.

The **ClassificationResult** column contains the most likely predicted class for the row.

The **ClassificationExplanation** column contains an explanation with the specific influence that the input features had on the **ClassificationScore**.

Regression models

Regression models are used to predict a numeric value. For instance: the revenue likely to be realized from a sales deal, the lifetime value of an account, the amount of a receivable invoice that is likely to be paid, the date on which an invoice may be paid, and so on.

The output of a Regression model is the predicted value.

Training a Regression model

The input table containing the training data for a Regression model must have a numeric column as the outcome column, which identifies the known outcome values.

Pre-requisites:

- A minimum of 100 rows of historical data is required for a Regression model

The process of creation for a Regression model follows the same steps as other AutoML models, described in the section **Configuring the ML model inputs** above.

Regression model report

Like the other AutoML model reports, the Regression report is based on the results from applying the model to the holdout test data.

The model report includes a chart that compares the predicted values to the actual values. In this chart, the distance from the diagonal indicates the error in the prediction.

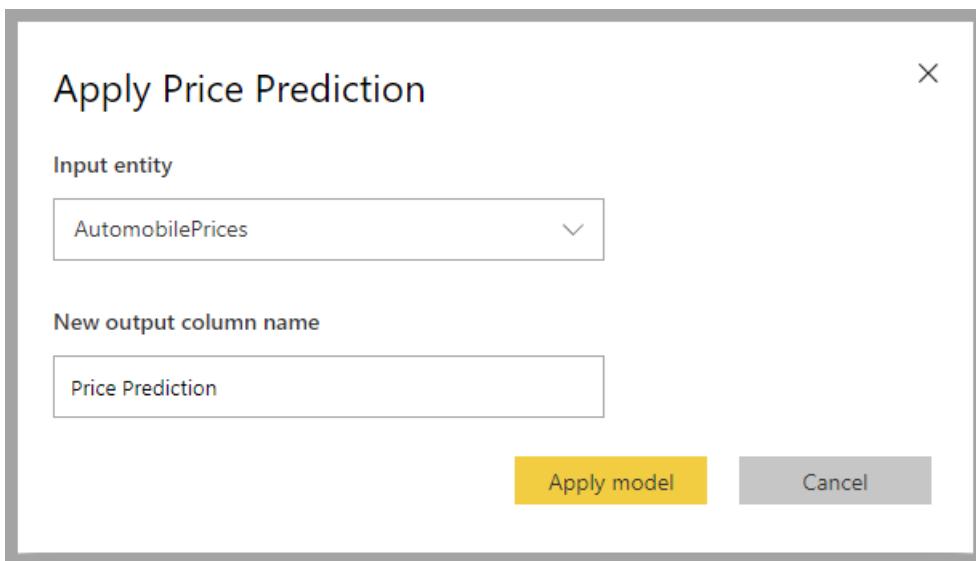
The residual error chart shows the distribution of the percentage of average error for different values in the holdout test dataset. The horizontal axis represents the mean of the actual value for the group, with the size of the bubble showing the frequency or count of values in that range. The vertical axis is the average residual error.



The Regression model report also includes a Training Details page like the reports for other model types, as described in the section **AutoML model report** above.

Applying a regression model

To apply a Regression ML model, you must specify the table with the input data and the output column name prefix.



When a Regression model is applied, it adds three output columns to the enriched output table: **RegressionResult**, **RegressionExplanation**, and **ExplanationIndex**. The column names in the table have the prefix specified when the model is applied.

The **RegressionResult** column contains the predicted value for the row based on the input columns. The **RegressionExplanation** column contains an explanation with the specific influence that the input features had on the **RegressionResult**.

Azure Machine Learning integration in Power BI

Numerous organizations use **Machine Learning** models for better insights and predictions about their business. The ability to visualize and invoke insights from these models, in your reports and dashboards and other analytics, can help disseminate these insights to the business users who need it the most. Power BI now makes it simple to incorporate the insights from models hosted on Azure Machine Learning, using straightforward point-and-click gestures.

To use this capability, a data scientist can simply grant access to the Azure ML model to the BI analyst using the Azure portal. Then, at the start of each session, Power Query discovers all the Azure ML models to which the user has access and exposes them as dynamic Power Query functions. The user can then invoke those functions by accessing them from the ribbon in Power Query Editor, or by invoking the M function directly. Power BI also automatically batches the access requests when invoking the Azure ML model for a set of rows to achieve better performance.

This functionality is currently only supported for Power BI dataflows, and for Power Query online in the Power BI service.

To learn more about dataflows, see [Introduction to dataflows and self-service data prep](#).

To learn more about Azure Machine Learning, please see:

- Overview: [What is Azure Machine Learning?](#)
- Quick Starts and Tutorials for Azure Machine Learning: [Azure Machine Learning Documentation](#)

Granting access to the Azure ML model to a Power BI user

To access an Azure ML model from Power BI, the user must have **Read** access to the Azure subscription and the Machine Learning workspace.

The steps in this article describe how to grant a Power BI user access to a model hosted on the Azure ML service, so they can access this model as a Power Query function. For further details, please see [Manage access using RBAC and the Azure portal](#).

1. Sign in to the Azure portal.
2. Go to the **Subscriptions** page. You can find the **Subscriptions** page through the **All Services** list in the nav pane menu of the Azure portal.

The screenshot shows the Microsoft Azure 'All services' page. On the left, there's a navigation pane with various service icons and links like Dashboard, Resource groups, App Services, etc. The main area is titled 'All services' and contains a grid of service tiles. In the top right corner of the grid, there are three buttons: 'Collapse all', 'Expand all', and a user profile icon. The grid is organized into sections: GENERAL (15), COMPUTE (20), and NETWORKING (20). The 'Subscriptions' tile is located in the GENERAL section. Other visible tiles include 'Management groups', 'Cost Management + Billing', 'Help + support', 'Tags', 'Recent', 'Resource groups', 'Marketplace', 'Templates', 'Quickstart Center', 'Virtual machines', 'Container services', 'Container instances', 'Cloud services (classic)', 'Disks', 'Images', 'Virtual networks', 'Application gateways', 'DNS zones', 'ExpressRoute circuits', 'Network security groups (classic)', 'Reserved IP addresses (classic)', 'Virtual machine scale sets', 'App Services', 'Service Fabric clusters', 'Availability sets', 'Snapshots', 'OS images (classic)', 'Citrix Virtual Apps Essentials', 'Load balancers', 'Local network gateways', 'CDN profiles', 'Traffic Manager profiles', 'Network security groups', 'Public IP addresses', 'On-premises Data Gateways', and 'Connections'.

3. Select your subscription.

The screenshot shows the Microsoft Azure 'Subscriptions' page. The left navigation pane is identical to the previous screenshot. The main content area has a header 'Home > Subscriptions'. Below the header, there's a message 'Showing subscriptions in Microsoft. Don't see a subscription? Switch directories'. A dropdown menu shows 'My role' with '7 selected' and 'Status' with '3 selected'. There are buttons for 'Apply' and 'Show only subscriptions selected in the global subscriptions filter'. A search bar is present with the placeholder 'Search to filter items...'. A table lists the available subscriptions. The first row shows 'Pay-As-You-Go' with a blue circular icon, 'Subscription ID' as '4x5c0b37-1baa-4b4e-bbf4-4c74443c1dba', 'MY ROLE' as 'Account admin', 'CURRENT COST' as '\$96.27', and 'STATUS' as 'Active'. There are three dots at the end of the row.

4. Select Access Control (IAM), and then select the Add button.

5. Select **Reader** as the Role. Select the Power BI user to whom you wish to grant access to the Azure ML model.

6. Select **Save**.

7. Repeat steps three through six to grant **Reader** access to the user for the specific Machine Learning workspace hosting the model.

Schema discovery for Machine Learning models

Data scientists primarily use Python to develop, and even deploy, their machine learning models for Machine Learning. The data scientist must explicitly generate the schema file using Python.

This schema file must be included in the deployed web service for Machine Learning models. To automatically generate the schema for web service, you must provide a sample of the input/output in the entry script for the deployed model. Please see the subsection on [\(Optional\) Automatic Swagger schema generation in the Deploy models with the Azure Machine Learning service documentation](#). The link includes the example entry script with the statements for the schema generation.

Specifically, the `@input_schema` and `@output_schema` functions in the entry script reference the input and output sample formats in the `input_sample` and `output_sample` variables, and use these samples to generate an OpenAPI (Swagger) specification for the web service during deployment.

These instructions for schema generation by updating the entry script must also be applied to models created

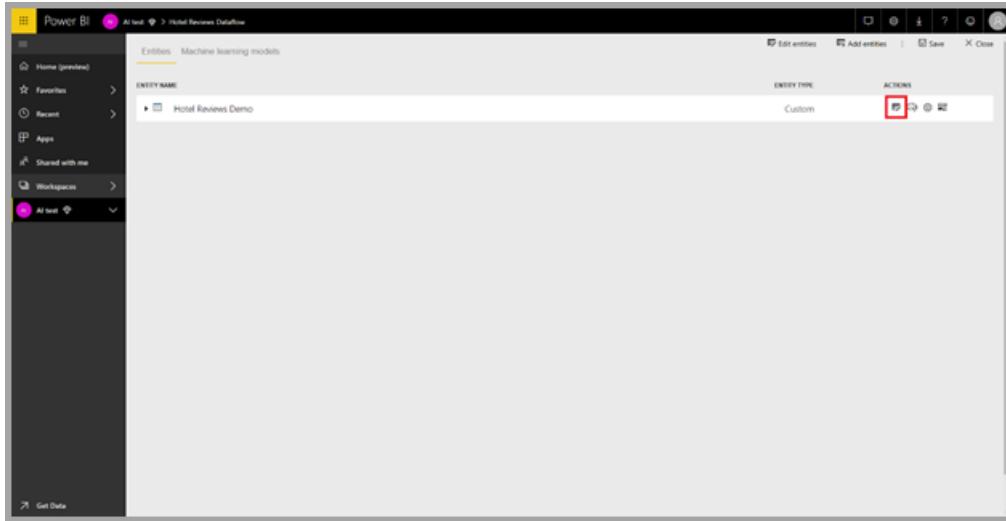
using automated machine learning experiments using the Azure Machine Learning SDK.

NOTE

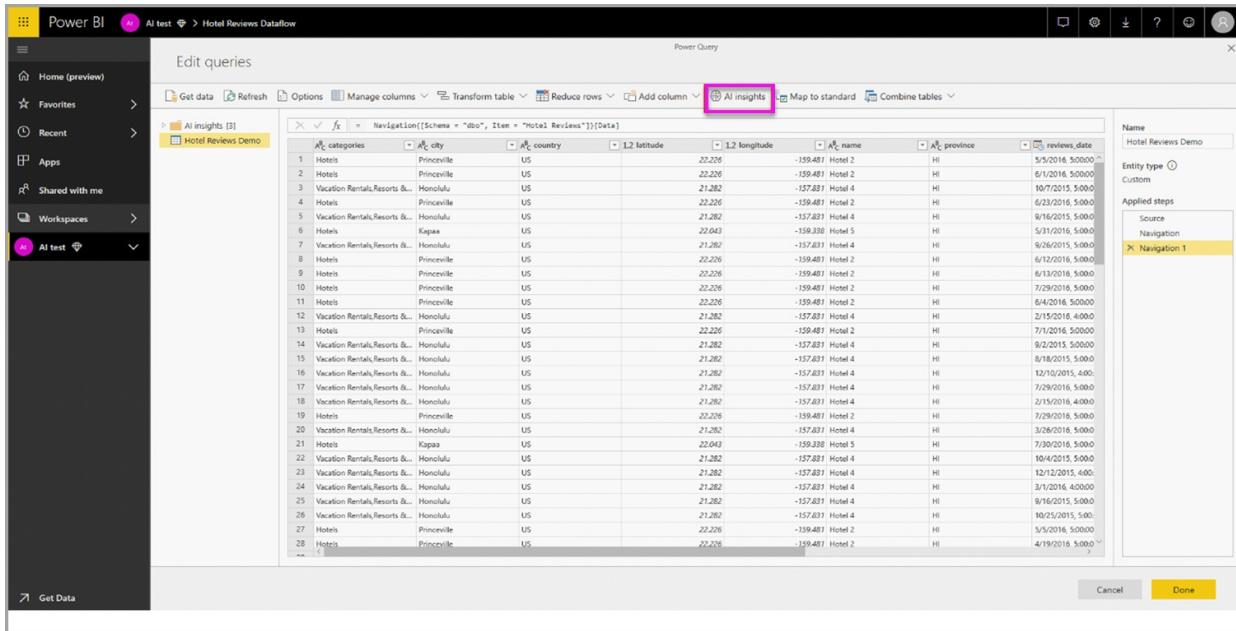
Models created using the Azure Machine Learning visual interface do not currently support schema generation, but will in subsequent releases.

Invoking the Azure ML model in Power BI

You can invoke any Azure ML model to which you have been granted access, directly from the Power Query Editor in your dataflow. To access the Azure ML models, select the **Edit** button for the table that you want to enrich with insights from your Azure ML model, as shown in the following image.

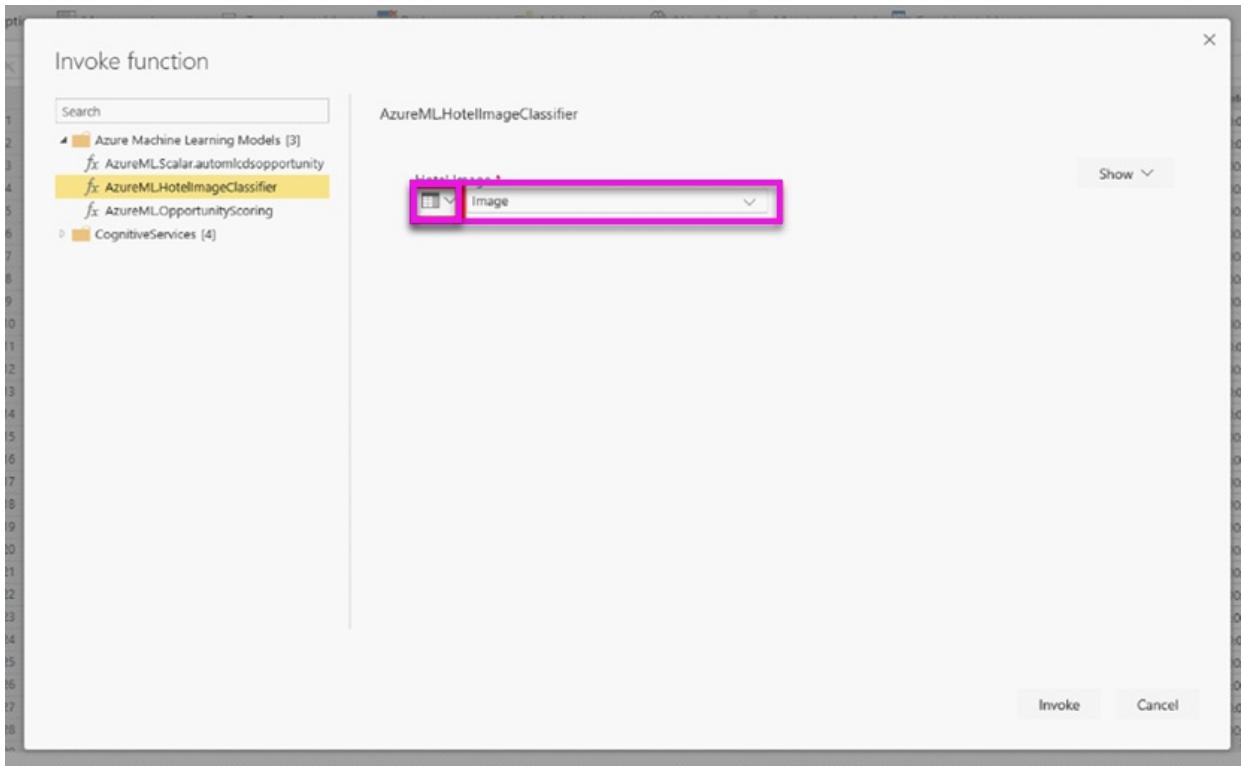


Selecting the **Edit** button opens the Power Query Editor for the tables in your dataflow.



Select the **AI Insights** button in the ribbon, and then select the *Azure Machine Learning Models* folder from the nav pane menu. All the Azure ML models to which you have access are listed here as Power Query functions. Also, the input parameters for the Azure ML model are automatically mapped as parameters of the corresponding Power Query function.

To invoke an Azure ML model, you can specify any of the selected table's columns as an input from the dropdown. You can also specify a constant value to be used as an input by toggling the column icon to the left of the input dialog.



Select **Invoke** to view the preview of the Azure ML model's output as a new column in the table table. You will also see the model invocation as an applied step for the query.

Index	Province	Review Date	Review Date Added	Review Text	Review Title	Image
1	HI	5/5/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	We had A/C issues at 3:30 ...	Please Keep Away Until Co...	Record
2	HI	6/1/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	A/C was broken. Hotel was...	Please Keep Away Until Co...	Record
3	HI	10/7/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	Elevator was broken.	Gutes Preis-Leistungsverh...	Record
4	HI	6/23/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	Elevator was broken.	Gutes Preis-Leistungsverh...	Record
5	HI	9/16/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	Unprepared for the unwele...	Very much a budget place	Record
6	HI	5/3/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	I expected that the Jesusc...	Did not live up to the Hilt...	Record
7	HI	9/26/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	For the price that I paid fo...	Rooms	Record
8	HI	6/12/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	At Night A/C very loud, ake...	AC in room Too loud	Record
9	HI	6/13/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	The A/C in my room broke...	Don't waste your money	Record
10	HI	7/29/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	Great beach park off the la...	Nice surprise!	Record
11	HI	6/4/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	Our room was on the bott...	Great staff, excellent getaw...	Record
12	HI	2/15/2016, 4:00:00 PM	2/25/2017, 12:32:57 PM	We spent 2 weeks in this ho...	IN SEVERE NEED OF UPDA...	Record
13	HI	7/1/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	Terrible view from my \$300...	Beautiful renovations locat...	Record
14	HI	9/2/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	Older property but it is su...	Combines great price with ...	Record
15	HI	8/18/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	We stayed here for over a ...	Affordable, Clean, Friendly ...	Record
16	HI	12/10/2015, 4:00:00 PM	2/25/2017, 12:32:57 PM	When we had booked this ...	Average	Record
17	HI	7/29/2016, 5:00:00 PM	2/25/2017, 12:32:57 PM	Loved the beach and service	Well the location is nice	Record
18	HI	2/15/2016, 4:00:00 PM	2/25/2017, 12:32:57 PM	I hesitate to share negative...	Advertisement Share	Record
19	HI	7/29/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	Beautiful renovation. The h...	Beautiful renovations locat...	Record
20	HI	3/26/2016, 5:00:00 PM	2/25/2017, 12:32:57 PM	Positives: Location! It is o...	Location!	Record
21	HI	7/30/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	This hotel is on the beach ...	Great Location	Record
22	HI	10/4/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	Clean room, old style. 196...	Location	Record
23	HI	12/12/2015, 4:00:00 PM	2/25/2017, 12:32:57 PM	The accommodation is bas...	Polyneian Plaza, Honolulu	Record
24	HI	3/1/2016, 4:00:00 PM	2/25/2017, 12:32:57 PM	The entrance to the hotel L...	Outdated but friendly	Record
25	HI	9/16/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	Rooms were nice, basic bu...	Close to everything in Wai...	Record
26	HI	10/25/2015, 5:00:00 PM	2/25/2017, 12:32:57 PM	I booked this hotel for mid...	Unprofessional	Record
27	HI	5/5/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	Loved the view from my roo...	(null)	Record
28	HI	4/19/2016, 5:00:00 PM	3/31/2017, 11:32:19 AM	A/C unit was disounting an...	(null)	Record

If the model returns multiple output parameters, they are grouped together as a row in the output column. You can expand the column to produce individual output parameters in separate columns.

Once you save your dataflow, the model is automatically invoked when the dataflow is refreshed, for any new or updated rows in the table table.

Considerations and limitations

- AI insights (Cognitive Services and Azure ML models) are not supported on machines with proxy authentication setup.
- AzureML models are not supported for Guest users.

Next steps

This article provided an overview of Automated Machine Learning for Dataflows in the Power BI service. The following articles may also be useful.

- [Tutorial: Build a Machine Learning model in Power BI](#)
- [Tutorial: Using Cognitive Services in Power BI](#)
- [Tutorial: Consume Azure Machine Learning models in Power BI](#)

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [Premium features of dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

Dataflows best practices

3/30/2022 • 2 minutes to read • [Edit Online](#)

Power BI **dataflows** are an enterprise-focused data prep solution, enabling an ecosystem of data that's ready for consumption, reuse, and integration. This article provides a list of best practices, with links to articles and other information that will help you understand and use dataflows to their full potential.

Dataflows across the Power Platform

Dataflows can be used across various Power Platform technologies, such as Power Query, Microsoft Dynamics 365, and other Microsoft offerings. For more information about how dataflows can work across the Power Platform, see [using dataflows across Microsoft products](#).

Dataflows best practices table and links

The following table provides a collection of links to articles that describe best practices when creating or working with dataflows. The links include information about developing business logic, developing complex dataflows, re-use of dataflows, and how to achieve enterprise-scale with your dataflows.

TOPIC	GUIDANCE AREA	LINK TO ARTICLE OR CONTENT
Power Query	Tips and tricks to get the most of your data wrangling experience	Power Query best practices
Leveraging Computed Tables	There are performance benefits for using computed tables in a dataflow	Computed Tables Scenarios
Developing complex dataflows	Patterns for developing large-scale, performant dataflows	Complex dataflows
Reusing dataflows	Patterns, guidance, and use cases	Reusing dataflows
Large-scale implementations	Large-scale use and guidance to complement enterprise architecture	Data warehousing using dataflows
Leveraging Enhanced Compute	Potentially improve dataflow performance up to 25x	Enhanced Compute Engine
Optimizing your workload settings	Get the most our of your dataflows infrastructure by understanding the levers you can pull to maximize performance	Dataflows workload configuration
Joining and expanding tables	Creating performant joins	Optimize expanding table operations
Query folding guidance	Speeding up transformations using the source system	Query folding
Using data profiling	Understand column quality, distribution, and profile	Data profiling tools

TOPIC	GUIDANCE AREA	LINK TO ARTICLE OR CONTENT
Implementing error handling	Developing robust dataflows resilient to refresh errors, with suggestions	Patterns for common errors Complex error handling
Use Schema view	Improve the authoring experience when working with a wide table and performing schema level operations	Schema view
Linked tables	Reusing and referencing transformations	Linked Tables
Incremental refresh	Load the latest or changed data versus a full reload	Incremental refresh

Next steps

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Premium features of dataflows](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)

Understanding and optimizing dataflows refresh

3/30/2022 • 16 minutes to read • [Edit Online](#)

Power BI dataflows enable you to connect to, transform, combine, and distribute data for downstream analytics. A key element in dataflows is the refresh process, which applies the transformation steps you authored in the dataflows and updates the data in the artifacts themselves.

To understand run times, performance, and whether you're getting the most out of your dataflow, you can download the refresh history after a dataflow has been refreshed.

Understanding refreshes

There are two types of refreshes applicable to dataflows:

- **Full**, which performs a complete flush and reload of your data.
- **Incremental (Premium only)**, which processes a subset of your data based on time-based rules, expressed as a filter, that you configure. The filter on the date column is used to dynamically partition the data into ranges in the Power BI service. After incremental refresh is configured, the dataflow automatically alters your query to include filtering by date. You can edit the automatically generated query by using the **Advanced Editor** in Power Query to fine-tune or customize your refresh. If you bring your own Azure Data Lake Storage, you can see time slices of your data based on the refresh policy you've set.

NOTE

You can read more about [incremental refresh](#) and how it works.

Incremental refresh enables large dataflows in Power BI with the following benefits:

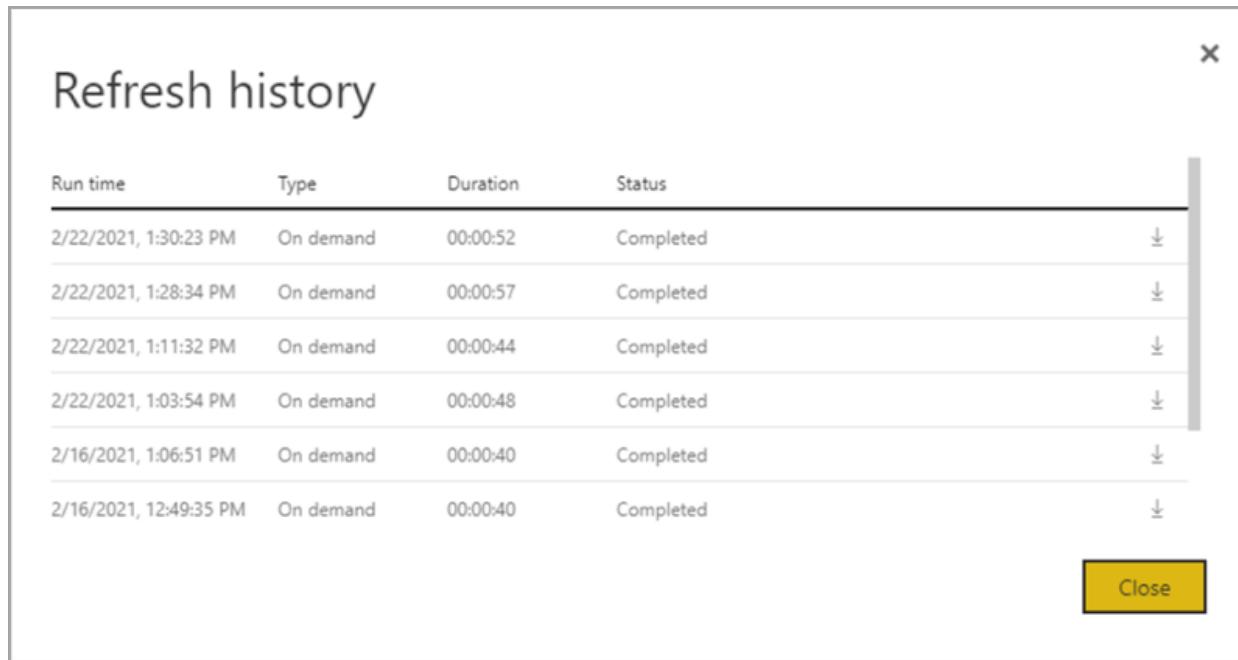
- Refreshes are faster after the first refresh, due to the following facts:
 - Power BI will refresh the last N partitions specified by the user (where partition is day/week/month, and so on), or:
 - Power BI will refresh only data that needs to be refreshed. For example, refreshing only the last five days of a 10-year dataset.
 - Power BI will only refresh only data that has changed, as long as you have specified the column to be checked for the change.
- Refreshes are more reliable - it's no longer necessary to maintain long-running connections to volatile source systems.
- Resource consumption is reduced - less data to refresh reduces overall consumption of memory and other resources.
- Wherever possible, Power BI employs parallel processing on partitions, which can lead to faster refreshes too.

In any of these refresh scenarios, if a refresh fails the data is not updated, which means that your data might be stale until the latest refresh completes, or you refresh it manually and it completes without error. Refresh occurs at a partition or entity, so if an incremental refresh fails, or an entity has an error, then the entire refresh transaction will not occur. Said another way, if a partition (incremental refresh policy) or entity fails for a

particular dataflow, the entire refresh operation is set to fail, and no data gets updated.

Understanding and optimizing refreshes

To better understand how a dataflow refresh operation performs, review the **Refresh History** for the dataflow by navigating to **Dataflow > Settings > Refresh History**. You can also select the dataflow in the **Workspace > context menu (...) > Refresh History**.



The screenshot shows a modal dialog titled "Refresh history". It contains a table with six rows of data, each representing a completed refresh. The columns are labeled "Run time", "Type", "Duration", and "Status". The data is as follows:

Run time	Type	Duration	Status
2/22/2021, 1:30:23 PM	On demand	00:00:52	Completed
2/22/2021, 1:28:34 PM	On demand	00:00:57	Completed
2/22/2021, 1:11:32 PM	On demand	00:00:44	Completed
2/22/2021, 1:03:54 PM	On demand	00:00:48	Completed
2/16/2021, 1:06:51 PM	On demand	00:00:40	Completed

In the bottom right corner of the dialog is a yellow "Close" button.

The **Refresh History** provides an overview of refreshes, including the type – *on demand* or *scheduled*, the duration, and the run status. To see details in the form of a CSV file, select the download icon on the far right of the refresh description's row. The downloaded CSV includes the attributes described in the following table. Premium refreshes provide more information based on the additional compute and dataflows capabilities, versus Pro based dataflows that reside on shared capacity. As such, some of the following metrics are available only in Premium.

ITEM	DESCRIPTION	PRO	PREMIUM
Requested on	Time refresh was scheduled or refresh now was clicked, in local time	✓	✓
Dataflow name	Name of your Dataflow	✓	✓
Dataflow refresh status	Completed, Failed, Skipped (for an entity) are possible statuses. Use cases like Linked Entities are reasons why one might see skipped.	✓	✓
Entity name	Table name	✓	✓

ITEM	DESCRIPTION	PRO	PREMIUM
Partition name	This is dependent on if the dataflow is premium or not, and if Pro will show as NA because it does not support incremental refreshes. Premium will show either FullRefreshPolicyPartition or IncrementalRefreshPolicyPartition-[DateRange]		✓
Refresh status	Refresh status of the individual entity or partition, which provides status for that time slice of data being refreshed.	✓	✓
Start time	In Premium, this is the time the dataflow was queued up to be processed for the entity or partition. This can differ if dataflows have dependencies and need to wait for the result set of an upstream dataflow to begin processing.	✓	✓
End time	This is the time the dataflow entity or partition completed, if applicable.	✓	✓
Duration	Total elapsed time for the dataflow to refresh expressed in HH:MM:SS	✓	✓
Rows processed	For a given entity or partition, # of rows scanned or written by the dataflows engine. This may not always contain data based on the operation you performed, such as when the Compute Engine is not used, or when using a gateway as the data is processed there.		✓
Bytes processed	For a given entity or partition, Data written by the dataflows engine, expressed in bytes. Note, when using a gateway on this particular dataflow this information will not be provided.		✓

ITEM	DESCRIPTION	PRO	PREMIUM
Max commit (KB)	<p>Max Commit is the peak commit memory useful for diagnosing out-of-memory failures when the M query is not optimized.</p> <p>When using a gateway on this particular dataflow, this information will not be provided.</p>		✓
Processor Time	<p>For a given entity or partition, Time, expressed in HH:MM:SS that the dataflows engine spent performing transformations.</p> <p>When using a gateway on this particular dataflow, this information will not be provided.</p>		✓
Wait time	For a given entity or partition, the time that an entity spent in wait status, based on workload on the Premium capacity.		✓
Compute engine	<p>For a given entity or partition, Details on how the Compute Engine was used in the refresh operation. Values are the following:</p> <ul style="list-style-type: none"> - NA - Folded - Cached - Cached + Folded <p>These elements are described in more detail later in this article.</p>		✓
Error	If applicable, the detailed error message is described per entity or partition	✓	✓

Dataflow refresh guidance

The refresh statistics provide valuable information you can use to optimize and speed up performance of your dataflows. In the following sections, we describe some scenarios, what to watch out for, and how to optimize based on the information provided.

Orchestration

Using dataflows in the same workspace allows straightforward orchestration. If you have dataflows A, B and C in a single workspace, and chaining like A > B > C, then if you refresh the source (A), the downstream entities also get refreshed. However, if you refresh C, then you'll have to refresh others independently. Also, if you add a

new data source in dataflow B (which is not included in A) that data will not be refreshed as a part of orchestration.

For scenarios where you want to chain items together that don't fit the managed orchestration Power BI performs, or if you want to mix and match, you can use the APIs and/or use PowerAutomate. You can refer to the [API documentation](#) and the [PowerShell script](#) for programmatic refresh. There is a Power Automate connector that enables doing this without writing any code. You can see [detailed samples](#), with specific walk-throughs for [sequential refreshes](#).

Monitoring

Using the enhanced refresh statistics described earlier in this article, you can get detailed per-dataflow refresh information. But if you would like to see dataflows with tenant-wide or workspace-wide overview of refreshes, perhaps to build a monitoring dashboard, you can use [the APIs](#) or [PowerAutomate templates](#). Similarly, for use cases such as [sending simple or complex notifications](#), you can use the PowerAutomate connector or build your own custom application using our APIs.

Timeout Errors

Optimizing the time it takes to perform extract, transform, and load scenarios is ideal. In Power BI, the following cases apply:

- Some connectors have explicit timeout settings you can configure. Check out the [Power Query Connector Reference](#) for more information
- Power BI dataflows, using Power BI Pro, can also experience timeouts for long running queries within an entity or dataflows themselves. That limitation does not exist in Power BI Premium workspaces.

Timeout guidance

Timeout thresholds for Power BI Pro dataflows are the following:

- Two hours at the individual entity level
- Three hours at the entire dataflow level

For example, if you have a dataflow with three tables, no individual table can take more than two hours and the entire dataflow will time out if the duration exceeds three hours.

If you are experiencing timeouts, consider optimizing your dataflow queries, and consider using [query folding](#) on your source systems.

Separately, consider upgrading to Premium Per User, which is not subject to these time-outs and offers increased performance due to many [Power BI Premium Per User features](#).

Long durations

Complex or large dataflows can take more time to refresh, as can poorly optimized dataflows. The following sections provide guidance on how to mitigate long refresh durations.

Guidance for long refresh durations

Dataflows best practices

The first step to improve long refresh durations for dataflows is to build dataflows according to our [best practices](#). Notable patterns include the following:

- Use [linked entities](#) for data that can be used later in other transformations
- Use [computed entities](#) to cache data, reducing data loading and data ingestion burden on source systems
- Split data into [staging dataflows](#) and [transformation dataflows](#), separating the ETL (extract, transform, load) into different dataflows
- Optimize expanding table operations
- Follow [guidance for complex dataflows](#)

Next, it can help to evaluate whether you can use incremental refresh.

Using [incremental refresh](#) can improve performance. It's important that the partition filters are pushed to the source system when queries are submitted for refresh operations. To push filtering down means the data source should support query folding, or you can express business logic through a function or other means that can help Power Query eliminate and filter files or folders. Most data sources that support SQL queries support query folding, and some OData feeds can also support filtering.

However, data sources like flat files, blobs, and APIs typically do not support filtering. In cases where the filter is not supported by the data source back-end, it cannot be pushed down. In such cases, the mash-up engine compensates and applies the filter locally, which may require retrieving the full dataset from the data source. This can cause incremental refresh to be very slow, and the process can run out of resources either in the Power BI service or in the on-premises data gateway, if used.

Given the various levels of query folding support for each data source, it's recommended that verification is performed to ensure the filter logic is included in the source queries. To make this easier, Power BI attempts to perform this verification for you, with [step folding indicators for Power Query Online](#). Many of these optimizations are design-time experiences, but after a refresh occurs, you have an opportunity to analyze and optimize your refresh performance.

Finally, consider optimizing your environment. You can optimize the Power BI environment by scaling up your capacity, right-sizing data gateways, and reducing network latency with the following optimizations:

- When using capacities available with Power BI Premium or Premium Per User, you can increase performance by increasing your Premium instance, or assigning the content to a different capacity.
- A gateway is required whenever Power BI needs to access data that isn't available directly over the Internet. You can install the [on-premises data gateway](#) on an on-premises server, or on a virtual machine.
* To understand gateway workloads and sizing recommendations, see [on-premises data gateway sizing](#). *
Also evaluate bringing the data first into a staging dataflow, and referencing it downstream using linked and computed entities.
- Network latency can impact refresh performance by increasing the time required for requests to reach the Power BI service, and for responses to be delivered. Tenants in Power BI are assigned to a specific region. To determine where your tenant is located, see [where is my Power BI tenant located?](#). When users from a tenant access the Power BI service, their requests always route to that region. As requests reach the Power BI service, the service may then send extra requests, for example, to the underlying data source, or a data gateway—which are also subject to network latency.
 - Tools such as [Azure Speed Test](#) provide an indication of network latency between the client and the Azure region. In general, to minimize the impact of network latency, strive to keep data sources, gateways, and your Power BI cluster as close as possible. Residing in the same region is preferable. If network latency is an issue, try locating gateways and data sources closer to your Power BI cluster by placing them inside cloud-hosted virtual machines.

High Processor Time

If you see high processor time, you likely have expensive transformations that aren't being folded, either because of the number of applied steps you have, or the type of transformations you're making, each of which can result in higher refresh times.

Guidance for high processor time

There are two options for optimizing high processor time.

First, use query folding within the data source itself, which should reduce the load on the dataflow Compute Engine directly. Query folding within the data source allows the source system to do most of the work, allowing the dataflow to pass through queries in the native language of the source, rather than having to perform all the computations in memory after the initial query.

Not all data sources can perform query folding, and even when query folding is possible there may be dataflows that perform certain transformations that cannot fold to the source. In such cases, the [enhanced Compute Engine](#) is a capability introduced by Power BI to potentially improve performance by up to 25 times, for transformations specifically.

Using the Compute Engine to maximize performance

While Power Query has design-time visibility into query folding, the Compute Engine column provides details about whether the internal engine itself is used. The Compute Engine is helpful when you have a complex dataflow and you're performing transformations in memory. This is where the enhanced refresh statistics can be helpful, since the Compute Engine column provides details about whether or not the engine itself was used.

The following sections provide guidance about using the Compute Engine, and its statistics.

Guidance on Compute Engine Statuses

Turning on the Enhanced Compute Engine and understanding the various statuses is helpful. Internally, the enhanced Compute Engine uses a SQL database to read and store data. It's best to have your transformations execute against the query engine here. The paragraphs below provide various situations, and guidance about what to do for each.

NA - This status means that the Compute Engine was not used, either because: you are using Power BI Pro dataflows; you have it explicitly turned off; you are using query folding on the data source; or you are performing complex transformations that cannot make use of the SQL engine used to speed up queries.

If you're experiencing long durations and still get a status of **NA**, make sure that it is [turned on](#) and not accidentally turned off. One recommended pattern is to use [staging dataflows to initially get your data into the Power BI service, then build dataflows on top of this data, once it is in a staging dataflow](#). That pattern can reduce load on source systems and, together with the Compute Engine, provide a speed boost for transformations and improve performance.

Cached - If you see **cached** status, the dataflow data was stored in the Compute Engine and available to be referenced as part of another query. This is ideal if you're using it as a Linked Entity, because that data is cached for use downstream and doesn't need to be refreshed multiple times in the same dataflow. This is also potentially ideal if you want to use it for DirectQuery.

When cached, the performance impact on initial ingestion will pay off later, in the same dataflow or different dataflow in same workspace.

If you have a large duration for the entity, consider turning off the Compute Engine. To cache the entity, Power BI writes it to storage and to SQL. If it's a single-use entity, the performance benefit for users may not be worth the penalty of the double-ingestion.

Folded - Folded means that the dataflow was able to use SQL Compute to read data. The calculated entity used the table from SQL to read data, and the SQL used is related to the constructs of their query.

Folded status appears if, when you're using on-premises or cloud data sources, you first loaded data into a staging dataflow and referenced that in this dataflow. This status applies only to entities that reference another entity. It means your queries were run on top of the SQL engine, and thus have the potential to be improved with SQL compute. To ensure your transformations are processed by the SQL engine, use transformations that support SQL folding, such as merge (join), group by (aggregation), and append (union) actions in the Query Editor.

Cached + Folded - When you see **cached + folded**, it's likely that the data refresh has been optimized, as you have an entity that both references another entity and is referred to by another entity upstream. This will also run on top of the SQL and, as such, also has the potential to be improved with SQL compute. To be sure you're getting the best performance possible, use transformations that support SQL folding, like merge (join), group by (aggregation), and append (union) actions in the Query Editor.

Guidance for Compute Engine performance optimization

The following steps will enable workloads to trigger the Compute Engine, and thereby, always improve performance:

Computed and Linked Entities in the same workspace:

For *ingestion*, focus on getting the data into the storage as fast as possible, using filters only if they reduce the overall dataset size. Keep your transformation logic separate from this step. Next, separate your transformation and business logic into a separate dataflow in the same workspace, using linked or computed entities; doing so allows for the engine to activate and accelerate your computations. For a simple analogy, it's like food preparation in a kitchen: food preparation is typically a separate and distinct step from gathering your raw ingredients, and a pre-requisite for putting the food in the oven. Similarly, your logic needs to be prepared separately before it can take advantage of the Compute Engine.

Ensure you perform the operations that fold, such as merges, joins, conversion, and [others](#).

Also, build dataflows [within published guidelines and limitations](#).

When the Compute engine is on, but performance is slow:

Take the following steps when investigating scenarios where the Compute Engine is on, but you're seeing poor performance:

- Limit computed and linked entities that exist across the workspace.
- If your initial refresh is performed with the Compute Engine turned on, data gets written in the lake *and* in the cache. This double-write results in refreshes being slower.
- If you have a dataflow linking to multiple dataflows, make sure you schedule refreshes of the source dataflows so that they don't all refresh at the same time.

Considerations and limitations

When using a Power BI Pro license, dataflows refreshes are limited to 8 refreshes per day.

Next steps

- [Using incremental refresh with dataflows](#)
- [Incremental refresh for datasets](#)
- [Troubleshooting refresh scenarios](#)
- [Dataflows best practices](#)
- [Premium features of dataflows](#)
- [Dataflows Limitations, restrictions and supported connectors and features](#)
- [Troubleshooting refresh scenarios](#)

Using DirectQuery with dataflows

3/30/2022 • 2 minutes to read • [Edit Online](#)

Using DirectQuery with Power BI dataflows lets you connect directly to a dataflow without the need to import the data into a dataset. There are many reasons why using DirectQuery with dataflows, rather than importing data, is useful and helpful. The following are a few examples:

- Working with large dataflows
- Decreasing orchestration needs for dataflows
- Serving data to customers in a managed and performance-minded way
- Preventing the need to duplicate data in a dataflow and an imported dataset

Configuration

To use DirectQuery with dataflows, you must explicitly toggle the **enhanced compute engine** to **On** in dataflow settings. You must then refresh the dataflow before it can be consumed in DirectQuery mode.

If you're using the original version of Power BI Premium (rather than using Premium Gen2), there are four items you must validate:

- The **enhanced compute engine** must be enabled for the Premium capacity *and* the specific dataflow.
- You must be running the latest version of Power BI Desktop.
- You must specifically connect to the data source using the **Power BI dataflows** connector.
- You must also take the following steps to connect using **Power BI Desktop**:
 1. Sign out of Power BI Desktop
 2. Clear the dataflows connection, which requires you sign in
 - Select **File > Options and settings > Data source settings > Delete Power BI dataflows**
 3. Make your connection using the **Power BI dataflows** connector, ensuring that the **enhanced compute engine** is on, and the connection has been refreshed

If you're using Premium Gen2, the following ordered steps are much simpler:

1. Navigate to the Premium dataflow, and set **enhanced compute engine** to **On**.
2. Navigate to the dataflow settings section for the target dataflow, and turn on **enhanced compute engine** for the dataflow.
3. Refresh the dataflow.

Once the steps are completed, the dataflow will be accessible in Power BI Desktop with DirectQuery mode.

Consumption

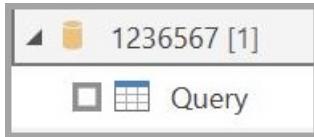
When DirectQuery is available for a dataflow, connecting to a dataflow using the **Power BI dataflows** connector prompts you to choose whether to connect to tables through DirectQuery or Import.

Dataflow entities that support DirectQuery display the **View** icon in Power BI Desktop, rather than the **Table** icon. The View icon appears as two boxes overlaid on each other, the Table icon is a single table with a grid.

The following image shows the **View** icon, indicating that the *Orders* table supports DirectQuery:



This image shows the **Table** icon, indicating that the *Query* table only supports import:



In DirectQuery mode, you can quickly interrogate large-scale datasets locally. However, you cannot currently perform any other transformations.

Next steps

This article provided an overview of using DirectQuery with dataflows. The following articles may also be useful.

- [About using DirectQuery in Power BI](#)
- [DirectQuery model guidance in Power BI Desktop](#)

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [Premium features of dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

Dataflows considerations and limitations

3/30/2022 • 6 minutes to read • [Edit Online](#)

There are a few dataflow limitations across authoring, refreshes, and capacity management that users should keep in mind, as described in the following sections.

General limitations

- Feature parity across government environments can be found in the [Power BI feature availability for government](#) article.
- Deleted datasources are not removed from the dataflow datasource page. This is a benign behavior and does not impact the refresh or editing of dataflows. In [Lineage View](#), deleted data sources appear as lineage for a dataflow.
- Deleted datasources will still appear in the Setting page in the gateway drop-down.
- *Depth* equates to dataflows linked to other dataflows. The current maximum depth is 32.
- *Breadth* equates to entities within a dataflow.
 - There is no guidance or limits for the optimal number of entities in a dataflow, however, shared dataflows have a refresh limit of two hours per entity, and three per dataflow. So if you have two entities, and each takes two hours, you shouldn't put them in the same dataflow.
 - For Power BI Premium, guidance and limits are driven by individual use cases rather than specific requirements. The only limit for Power BI Premium is a 24-hour refresh per dataflow.
- A Power BI Premium subscription is required in order to refresh more than ten dataflows cross workspace.
- PowerQuery limitations are found in the [PowerQuery Online usage limits](#) article.
- Power BI dataflows do not support use of global variables in a URL argument.
- Multi-Geo is currently not supported.
- Vnet support is achieved by using a gateway.
- When using *Computed entities* with gateway data sources, the data ingestion should be performed in different data sources than the computations. The computed entities should build upon entities that are only used for ingestion, and not ingest data within their own mash-up steps.
- In Power BI dataflows, you can use parameters but you cannot edit them unless you edit the entire dataflow. In this regard parameters in dataflows behave similar to declared constants.

Dataflow authoring

When authoring dataflows, users should be mindful of the following considerations:

- Authoring in Dataflows is done in the Power Query Online (PQO) environment; see the limitations described in [Power Query limits](#). Because dataflows authoring is done in the Power Query Online (PQO) environment, updates performed on the Dataflows workload configurations only impact refreshes, and will not have an impact on the authoring experience
- Dataflows can only be modified by their owners
- Dataflows are not available in *My Workspace*
- Dataflows using gateway data sources do not support multiple credentials for the same data source
- Using the Web.Page connector requires a gateway

API considerations

More about supported Dataflows REST APIs can be found in the [REST API reference](#). Here are some considerations to keep in mind:

- Exporting and Importing a dataflow gives that dataflow a new ID
- Importing dataflows that contain linked tables will not fix the existing references within the dataflow (these queries should be fixed manually before importing the dataflow)
- Dataflows can be overwritten with the *CreateOrOverwrite* parameter, if they have initially been created using the import API

Dataflows in shared capacities

There are limitations for Dataflows in shared capacities (non-Premium capacities):

- When refreshing Dataflows, timeouts in a shared capacity are 2 hours per table, and 3 hours per Dataflow
- Linked tables cannot be created in shared Dataflows, although they can exist within the Dataflow as long as the *Load Enabled* property on the query is disabled
- Computed tables cannot be created in shared Dataflows
- AutoML and Cognitive services are not available in shared Dataflows
- Incremental refresh does not work in shared Dataflows

Dataflows in Premium

Dataflows that exist in Premium have the following considerations and limitations.

Refreshes and data considerations:

- When refreshing Dataflows, timeouts are 24 hours (no distinction for tables and/or dataflows)
- Changing a dataflow from an incremental refresh policy to a normal refresh, or vice versa, will drop all data
- Modifying a dataflow's schema will drop all data
- When using a Premium Per User (PPU) license with dataflows, data is cleared when moving data out of a PPU environment
- When a dataflow is refreshed in a Premium Per User (PPU) context, the data is not visible to non-PPU users
- Incremental refresh works with dataflows only when the enhanced compute engine is enabled

Linked and Computed tables:

- Linked tables can go down to a depth of 32 references
- Cyclic dependencies of linked tables are not allowed
- A linked table can't be joined with a regular table that gets its data from an on-premises data source
- When a query (query *A*, for example) is used in the calculation of another query (query *B*) in dataflows, query *B* becomes a calculated table. Calculated tables cannot refer to on-premises sources.

Compute Engine:

- While using the Compute engine, there is an approximate 10% to 20% initial increase in time for data ingestion.

- This only applied to the first dataflow that is on the compute engine, and reads data from the data source
- Subsequent dataflows that use the source dataflow will not incur the same penalty
- Only certain operations make use of the compute engine, and only when used through a linked table or as a computed table. A full list of operations is available in [this blog post](#).

Capacity Management:

- By design, the Premium Power BI Capacities have an internal Resource Manager which throttles the workloads in different ways when the capacity is running on low memory.
 1. For Dataflows, this throttling pressure reduces the number of available M Containers
 2. The memory for Dataflows can be set to 100%, with an appropriately sized container for your data sizes, and the workload will manage the number of containers appropriately
- The approximate number of containers can be found out by dividing the total memory allocated to the workload by the amount of memory allocated to a container

Dataflow usage in datasets

- When creating a dataset in Power BI Desktop, and then publishing it to the Power BI service, ensure the credentials used in Power BI Desktop for the Dataflows data source are the same credentials used when the dataset is published to the service.
 1. Failing to ensure those credentials are the same results in a *Key not found* error upon dataset refresh

ADLS limitations

- ADLS is not available in GCC, GCC High or DOD environments. See [Power BI for US government customers](#) for more information.
- You must be assigned as an owner of the resource, due to changes in the ADLS Gen 2 APIs.
- Azure subscription migration is not supported, but there are two alternatives to do so:
 - First approach: after migration, the user can detach workspaces and reattach them. If using the tenant level account, you must detach all workspaces then detach at the tenant level, and reattach. This can be undesirable for customers who don't want to delete all of their dataflows, or have many workspaces.
 - Second approach: if the previous approach isn't feasible, submit a support request to change the subscription ID in the database.
- ADLS doesn't support most elements in the list in the [Directories and file names](#) section of the article for workspace naming and dataflow naming, due to the following limitations:
 - Power BI either returns an unhelpful error, or allows the process to happen but the refresh will fail.
- Cross tenant ADLS subscriptions are not supported. The ADLS attached to Power BI must be part of the same Azure tenant that Power BI uses for Azure Active Directory (Azure AD).

Dataflow data types

The data types supported in dataflows are the following:

MASHUP DATA TYPE	DATAFLOW DATA TYPE
Time	Time
Date	Date
DateTime	DateTime

MASHUP DATA TYPE	DATAFLOW DATA TYPE
DateTimeZone	DateTimeOffset
Logical	Boolean
Text	String
Any	String
Currency	Decimal
Int8	Int64
Int16	Int64
Int32	Int64
Int64	Int64
Double	Double
Percentage	Double
Single	Double
Decimal	Double
Number	Double
Duration	Not Supported
Binary	Not Supported
Function	Not Supported
Table	Not Supported
List	Not Supported
Record	Not Supported
Type	Not Supported
Action	Not Supported
None	Not Supported
Null	Not Supported

Next steps

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Creating a dataflow](#)
- [Configure and consume a dataflow](#)
- [Configuring Dataflow storage to use Azure Data Lake Gen 2](#)
- [Premium features of dataflows](#)
- [AI with dataflows](#)
- [Dataflows best practices](#)

Streaming dataflows (preview)

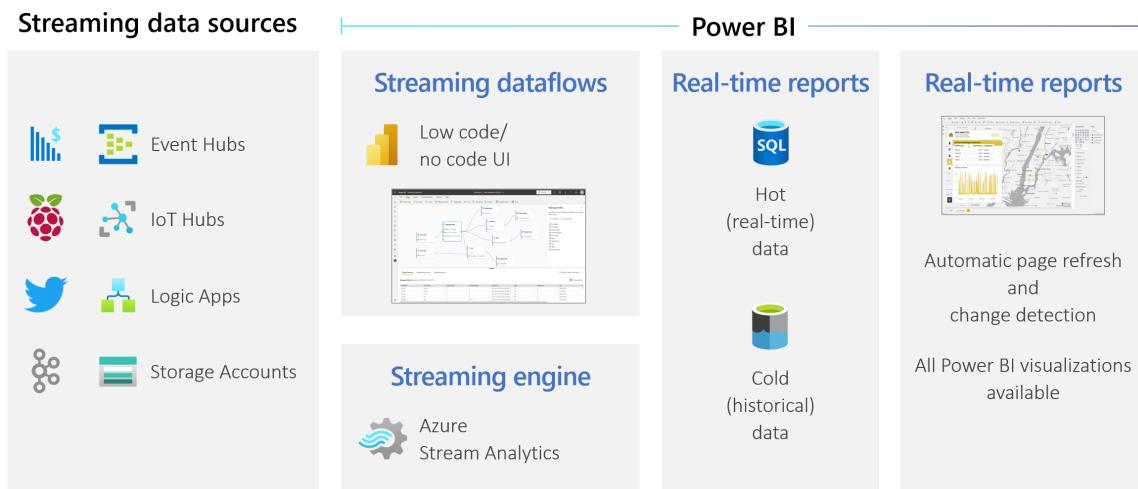
3/30/2022 • 30 minutes to read • [Edit Online](#)

Organizations want to work with data as it comes in, not days or weeks later. The vision of Power BI is simple: the distinctions between batch, real-time, and streaming data today will disappear. Users should be able to work with all data as soon as it's available.

Analysts usually need technical help to deal with streaming data sources, data preparation, complex time-based operations, and real-time data visualization. IT departments often rely on custom-built systems, and a combination of technologies from various vendors, to perform timely analyses on the data. Without this complexity, they can't provide decision makers with information in near real time.

Streaming dataflows allow authors to connect to, ingest, mash up, model, and build reports based on streaming, near real-time data directly in the Power BI service. The service enables drag-and-drop, no-code experiences.

Users can mix and match streaming data with batch data if they need to. This is done through a UI that includes a *diagram view* for easy data mashup. The final artifact produced is a dataflow, which can be consumed in real time to create highly interactive, near real-time reporting. All of the data visualization capabilities in Power BI work with streaming data, just as they do with batch data.



Users can perform data preparation operations like joins and filters. They can also perform time-window aggregations (such as tumbling, hopping, and session windows) for group-by operations.

Streaming dataflows in Power BI empower organizations to:

- Make confident decisions in near real time. Organizations can be more agile and take meaningful actions based on the most up-to-date insights.
- Democratize streaming data. Organizations can make data more accessible and easier to interpret with a no-code solution, and reduce IT resources.
- Accelerate time to insight by using an end-to-end streaming analytics solution with integrated data storage and BI.

Streaming dataflows support DirectQuery and [automatic page refresh/change detection](#). This support allows users to build reports that update in near real time, up to every second, by using any visual available in Power BI.

Requirements

Before you create your first streaming dataflow, make sure that you meet all the following requirements:

- To create and run a streaming dataflow, you need a workspace that's part of a *Premium capacity* or *Premium Per User (PPU)* license.

IMPORTANT

If you're using a PPU license and you want other users to consume reports created with streaming dataflows that are updated in real time, they'll also need a PPU license. They can then consume the reports with the same refresh frequency that you set up, if that refresh is faster than every 30 minutes.

- Enable dataflows for your tenant. For more information, see [Enabling dataflows in Power BI Premium](#).
- To make sure streaming dataflows work in your Premium capacity, the [enhanced compute engine](#) needs to be turned on. The engine is turned on by default, but Power BI capacity admins can turn it off. If this is the case, contact your admin to turn it on.

The [enhanced compute engine](#) is available only in Premium P or Embedded A3 and larger capacities. To use streaming dataflows, you need either PPU, a Premium P capacity of any size, or an Embedded A3 or larger capacity. For more information about Premium SKUs and their specifications, see [Capacity and SKUs in Power BI embedded analytics](#).

- To create reports that are updated in real time, make sure that your admin (capacity and/or Power BI for PPU) has enabled automatic page refresh. Also make sure that the admin has allowed a minimum refresh interval that matches your needs. For more information, see [Automatic page refresh in Power BI](#).

Create a streaming dataflow

A streaming dataflow, like its dataflow relative, is a collection of entities (tables) created and managed in workspaces in the Power BI service. A table is a set of fields that are used to store data, much like a table within a database.

You can add and edit tables in your streaming dataflow directly from the workspace in which your dataflow was created. The main difference with regular dataflows is that you don't need to worry about refreshes or frequency. Because of the nature of streaming data, there's a continuous stream coming in. The refresh is constant or infinite unless you stop it.

NOTE

You can have only one type of dataflow per workspace. If you already have a regular dataflow in your Premium workspace, you won't be able to create a streaming dataflow (and vice versa).

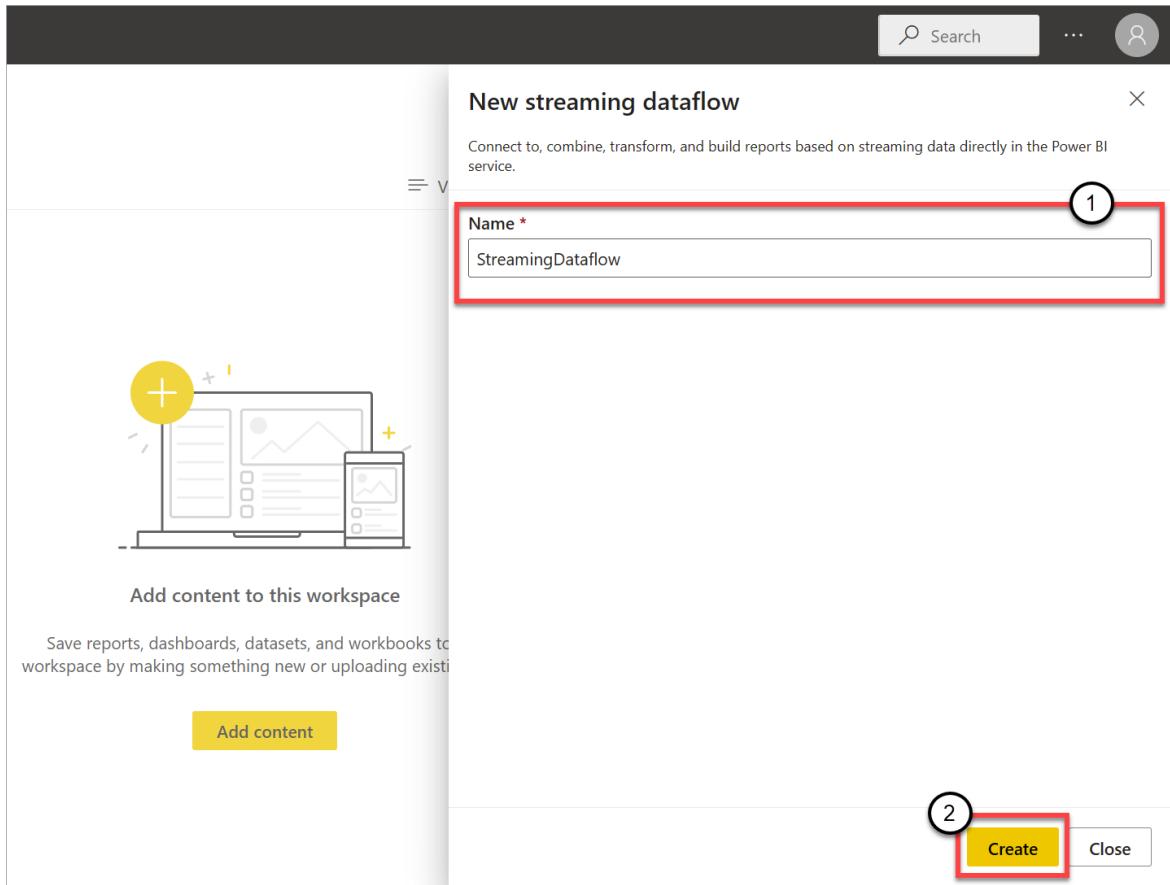
To create a streaming dataflow:

1. Open the Power BI service in a browser, and then select a Premium-enabled workspace. (Streaming dataflows, like regular dataflows, are not available in **My Workspace**.)
2. Select the **New** dropdown menu, and then select **Streaming dataflow**.

The screenshot shows the Microsoft Power BI interface, specifically the 'Streaming dataflows' workspace. The top navigation bar includes the Microsoft logo, 'Power BI', and 'Streaming dataflows'. On the left, there's a vertical sidebar with various icons for different types of content. The main area displays a title 'Streaming dataflows' with a lightning bolt icon and a subtitle 'Workspace to test and demo streaming dataflows'. Below this, there's a button labeled '+ New' with a dropdown arrow, followed by a link 'Create a pipeline'. A list of items follows:

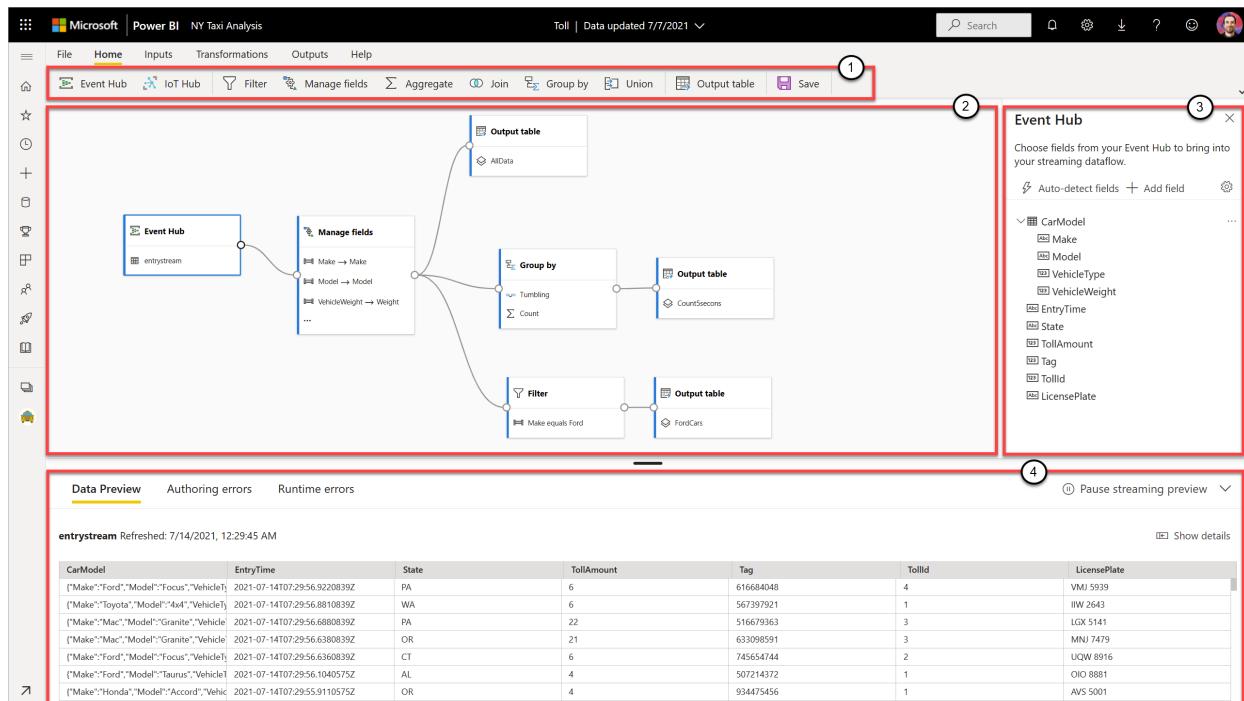
- Report: Visualize your data
- Paginated Report: Build a paginated report
- Scorecard: Track related goals together
- Dashboard: Build a single-page data story
- Dataset: Create a dataset to use in a report
- Dataflow: Prep, clean, and transform data
- Streaming dataset: Build visuals from real-time data
- Streaming dataflow**: Combine and transform streaming ... (This item is highlighted with a red box)
- Upload a file: Open a .pbix, .rdl, .xlsx, or .csv in Po...

3. On the side pane that opens, you must name your streaming dataflow. Enter a name in the **Name** box (1), and then select **Create** (2).



The empty diagram view for streaming dataflows appears.

The following screenshot shows a finished dataflow. It highlights all the sections available to you for authoring in the streaming dataflow UI.



- Ribbon:** On the ribbon, sections follow the order of a "classic" analytics process: inputs (also known as data sources), transformations (streaming ETL operations), outputs, and a button to save your progress.
- Diagram view:** This is a graphical representation of your dataflow, from inputs to operations to outputs.
- Side pane:** Depending on which component you selected in the diagram view, you'll have settings to modify each input, transformation, or output.

4. **Tabs for data preview, authoring errors, and runtime errors:** For each card shown, the data preview will show you results for that step (live for inputs and on-demand for transformations and outputs).

This section also summarizes any authoring errors or warnings that you might have in your dataflows. Selecting each error or warning will select that transform. In addition, you have access to runtime errors after the dataflow is running, such as dropped messages.

You can always minimize this section of streaming dataflows by selecting the arrow in the upper-right corner.

A streaming dataflow is built on three main components: *streaming inputs*, *transformations*, and *outputs*. You can have as many components as you want, including multiple inputs, parallel branches with multiple transformations, and multiple outputs.

Add a streaming input

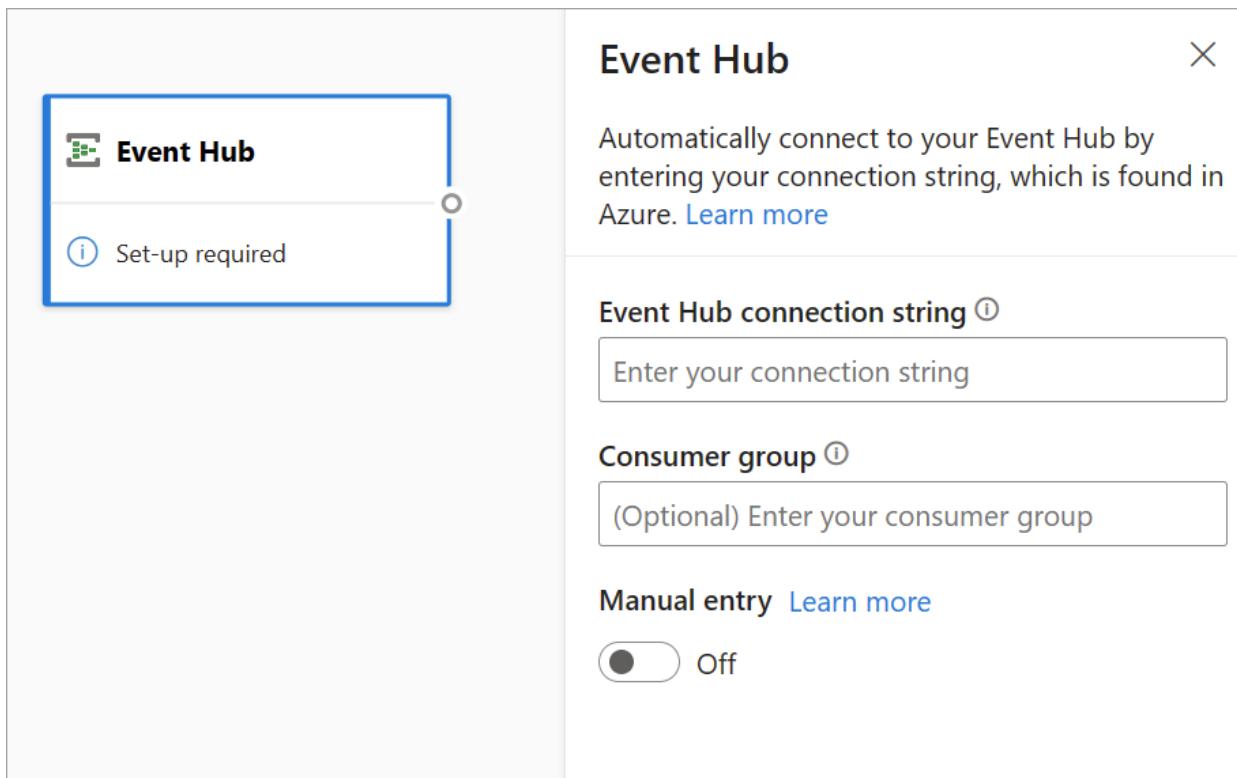
To add a streaming input, select the icon on the ribbon and provide the information needed on the side pane to set it up. As of July 2021, the preview of streaming dataflows supports [Azure Event Hubs](#) and [Azure IoT Hub](#) as inputs.

The Azure Event Hubs and Azure IoT Hub services are built on a common architecture to facilitate the fast and scalable ingestion and consumption of events. IoT Hub in particular is tailored as a central message hub for communications in both directions between an IoT application and its attached devices.

Azure Event Hubs

Azure Event Hubs is a big-data streaming platform and event ingestion service. It can receive and process millions of events per second. Data sent to an event hub can be transformed and stored by using any real-time analytics provider or batching/storage adapters.

To configure an event hub as an input for streaming dataflows, select the **Event Hub** icon. A card appears in the diagram view, including a side pane for its configuration.



You have the option of pasting the Event Hubs connection string. Streaming dataflows fill out all the necessary information, including the optional consumer group (which by default is **\$Default**). If you want to enter all

fields manually, you can turn on the manual-entry toggle to expose them. You can learn more about Event Hubs connection strings in [Get an Event Hubs connection string](#).

After you set up your Event Hubs credentials and select **Connect**, you can add fields manually by using **+ Add field** if you know the field names. To instead detect fields and data types automatically based on a sample of the incoming messages, select **Autodetect fields**. Selecting the gear icon allows you to edit the credentials if needed.

The screenshot shows a user interface for managing data fields. At the top, there are three buttons: 'Auto-detect fields' (with a lightning bolt icon), '+ Add field', and a gear icon for settings. Below this, a section titled 'CarModel' is expanded, showing four fields: 'Make', 'Model', 'VehicleType', and 'VehicleWeight', each with a small icon and 'Abc' prefix. Below this section, another section is partially visible with the title 'EntryTime' and a small calendar icon. A 'More options' button with three dots is located to the right of the 'EntryTime' section. A list of other detected fields follows: 'State', 'TollAmount', 'Tag', 'TollId', and 'LicensePlate'. Each field has a small icon and 'Abc' prefix.

When streaming dataflows detect the fields, you'll see them in the list. You'll also see a live preview of the incoming messages in the **Data Preview** table under the diagram view.

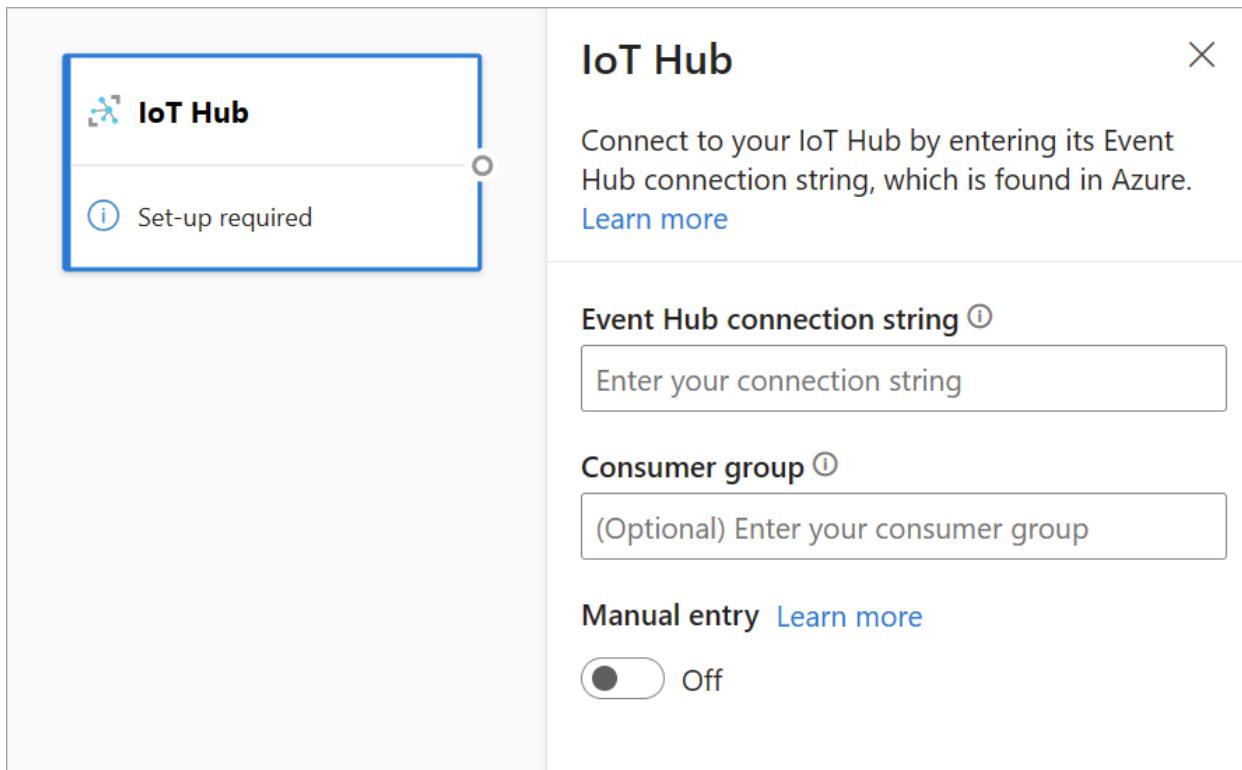
You can always edit the field names, or remove or change the data type, by selecting the three dots (...) next to each field. You can also expand, select, and edit any nested fields from the incoming messages, as shown in the following image.

The screenshot shows a context menu for the 'EntryTime' field. The menu items are: 'Remove' (with a trash bin icon), 'Rename' (with a rename icon), and 'Data type' (with a pencil icon). The 'Data type' option is currently selected. Below the menu, the list of detected fields is shown again, with 'EntryTime' now expanded to reveal its nested fields: 'DateTime', 'Float', 'Int', 'Record', and 'String'. The 'DateTime' field is highlighted with a checkmark and a calendar icon.

Azure IoT Hub

IoT Hub is a managed service hosted in the cloud. It acts as a central message hub for communications in both directions between an IoT application and its attached devices. You can connect millions of devices and their back-end solutions reliably and securely. Almost any device can be connected to an IoT hub.

IoT Hub configuration is similar to Event Hubs configuration because of their common architecture. But there are some differences, including where to find the Event Hubs-compatible connection string for the built-in endpoint. You can learn more about the IoT Hub built-in endpoint in [Read device-to-cloud messages from the built-in endpoint](#).



After you paste the connection string for the built-in endpoint, all functionality for selecting, adding, autodetecting, and editing fields coming in from IoT Hub is the same as in Event Hubs. You can also edit the credentials by selecting the gear icon.

TIP

If you have access to Event Hubs or IoT Hub in your organization's Azure portal and you want to use it as an input for your streaming dataflow, you can find the connection strings in the following locations:

For Event Hubs:

1. In the **Analytics** section, select **All Services > Event Hubs**.
2. Select **Event Hubs Namespace > Entities/Event Hubs**, and then select the event hub name.
3. In the **Shared Access Policies** list, select a policy.
4. Select the **Copy to clipboard** button next to the **Connection string-primary key** field.

For IoT Hub:

1. In the **Internet of Things** section, select **All Services > IoT Hubs**.
2. Select the IoT hub that you want to connect to, and then select **Built-in endpoints**.
3. Select the **Copy to clipboard** button next to the Event Hubs-compatible endpoint.

When you use stream data from Event Hubs or IoT Hub, you have access to the following metadata time fields in your streaming dataflow:

- **EventProcessedUtcTime**: The date and time that the event was processed.
- **EventEnqueuedUtcTime**: The date and time that the event was received.

Neither of these fields will appear in the input preview. You need to add them manually.

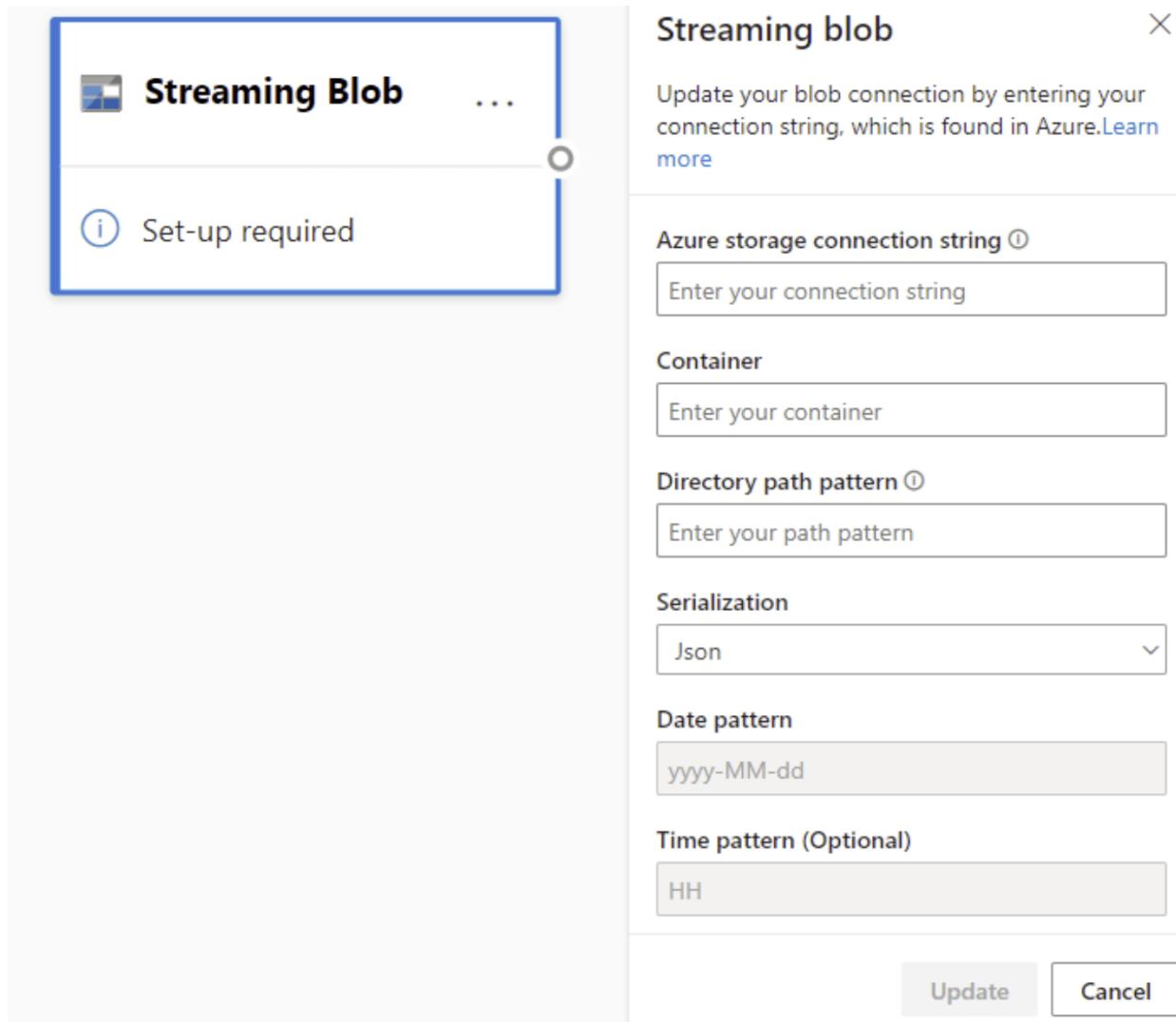
Blob storage

Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data. Unstructured data is data that doesn't adhere to a particular data model or definition, such as text or binary data.

We can use Azure Blobs as a streaming/reference input. Streaming blobs are generally checked every second for updates. Unlike a streaming blob, a reference blob is only loaded at the beginning of the refresh. It is static data that isn't expected to change and the recommended limit for which is 50MB or less.

Reference blobs are expected to be used alongside streaming sources (eg. Through a JOIN). Hence, a streaming dataflow with a reference blob must also have a streaming source.

The configuration for Azure Blobs is slightly different to that of an Azure Event Hub node. To find your Azure Blob connection string follow the directions under the 'View account access keys' section of this article [Manage account access keys - Azure Storage](#).



Once you've entered the Blob connection string, you will also need to enter the name of your container as well as the path pattern within your directory to access the files you want to set as the source for your dataflow.

For streaming blobs, the directory path pattern is expected to be a dynamic value. It is required for the date to be a part of the filepath for the blob – referenced as {date}. Furthermore, an asterisk (*) in the path pattern – eg. {date}/{time}/json will not be supported.

For example, if you have a blob called ExampleContainer within which you are storing nested json files – where the first level is the date of creation and the second level is the hour of creation (eg. 2021-10-21/16), then your Container input would be "ExampleContainer", the Directory path pattern would be "{date}/{time}" where you could modify the date and time pattern.

Container

ExampleContainer

Directory path pattern ⓘ

{date}/{time}

Serialization

Json



Date pattern

yyyy-MM-dd

After your blob is connected to the endpoint, all functionality for selecting, adding, autodetecting, and editing fields coming in from Azure Blob is the same as in Event Hubs. You can also edit the credentials by selecting the gear icon.

Often, when working with real time data, data will be condensed, and Identifiers are used to represent the object. A possible use case for blobs could also be as reference data for your streaming sources. Reference data allows you to join static data to streaming data to enrich your streams for analysis. Let's go through a quick example of when this would be helpful. Imagine you install sensors at different department stores to measure how many people are entering the store at a given time. Usually, the sensor ID needs to be joined onto a static table to indicate which department store and which location the sensor is located at. Now with reference data, it is possible to join this data during the ingestion phase to make it easy to see which store has the highest output of users.

NOTE

A Streaming Dataflows job pulls data from Azure Blob storage or ADLS Gen2 input every second if the blob file is available. If the blob file is unavailable, there is an exponential backoff with a maximum time delay of 90 seconds.

Data types

The available data types for streaming dataflows fields are:

- **DateTime**: Date and time field in ISO format.
- **Float**: Decimal number.
- **Int**: Integer number.
- **Record**: Nested object with multiple records.

- **String:** Text.

IMPORTANT

The data types selected for a streaming input have important implications downstream for your streaming dataflow. Select the data type as early as you can in your dataflow, to avoid having to stop it later for edits.

Add a streaming data transformation

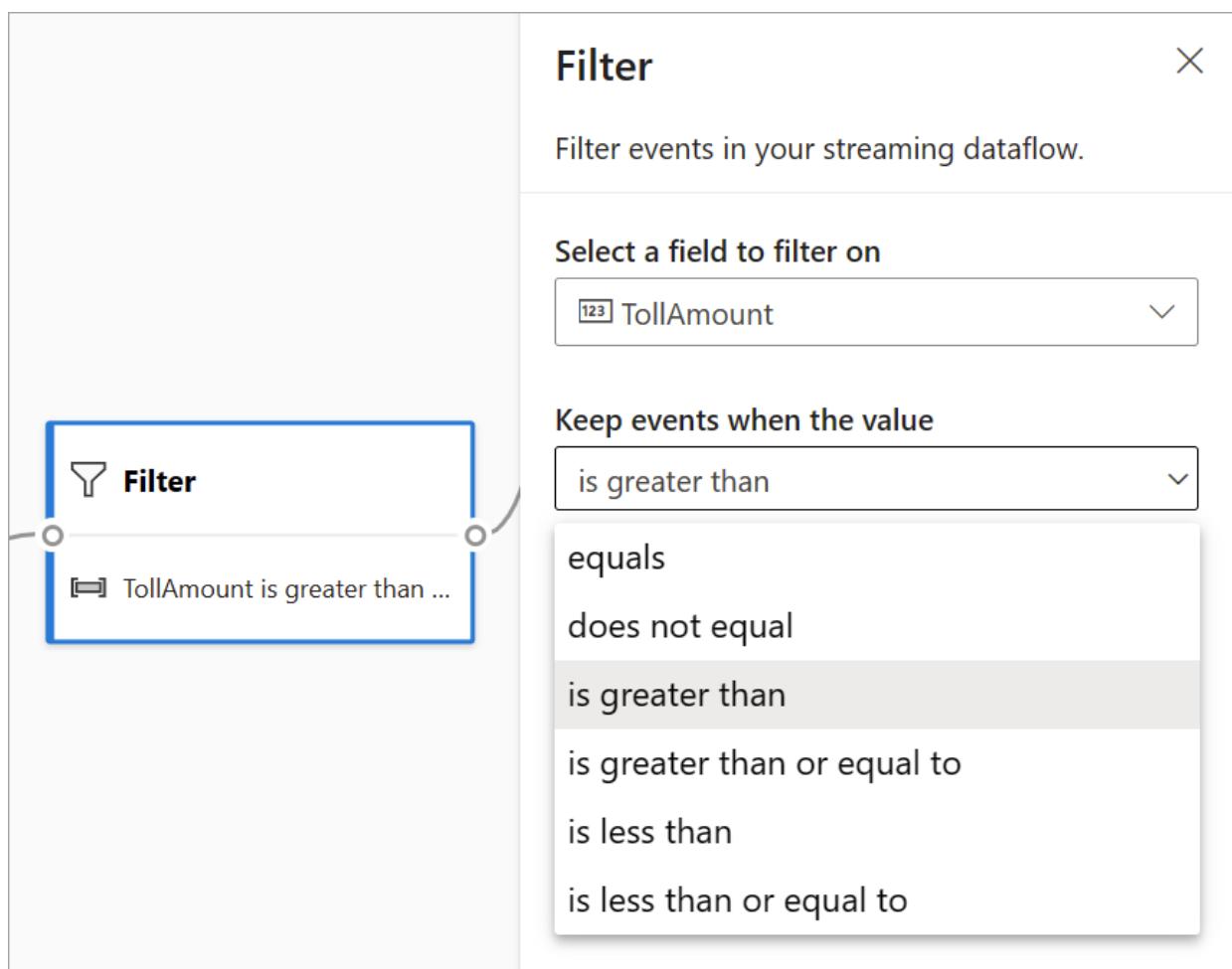
Streaming data transformations are inherently different from batch data transformations. Almost all streaming data has a time component, which affects any data preparation tasks involved.

To add a streaming data transformation to your dataflow, select the transformation icon on the ribbon for that transformation. The respective card will be dropped in the diagram view. After you select it, you'll see the side pane for that transformation to configure it.

As of July 2021, streaming dataflows support the following streaming transformations.

Filter

Use the **Filter** transformation to filter events based on the value of a field in the input. Depending on the data type (number or text), the transformation will keep the values that match the selected condition.

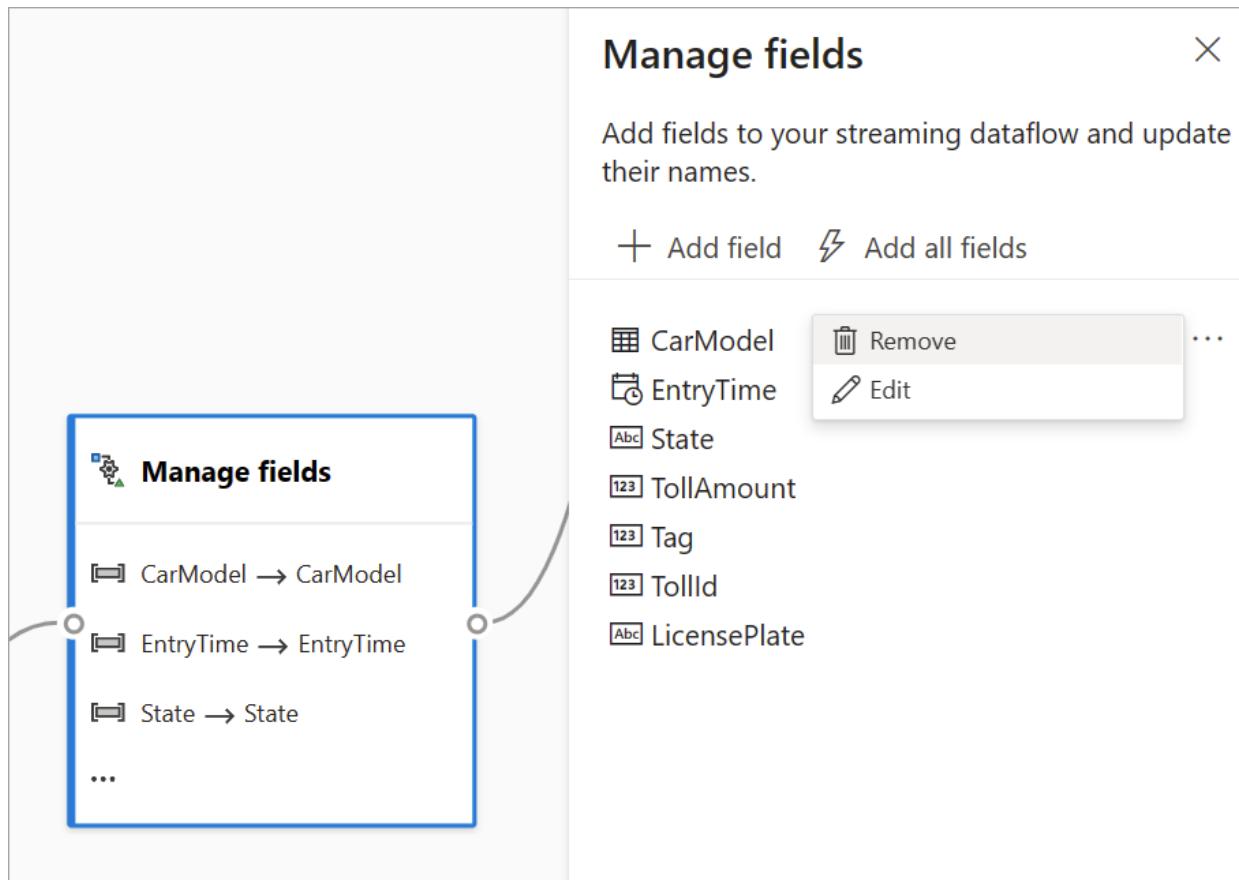


NOTE

Inside every card, you'll see information about what else is needed for the transformation to be ready. For example, when you're adding a new card, you'll see a "Set-up required" message. If you're missing a node connector, you'll see either an "Error" or a "Warning" message.

Manage fields

The **Manage fields** transformation allows you to add, remove, or rename fields coming in from an input or another transformation. The settings on the side pane give you the option of adding a new one by selecting **Add field** or adding all fields at once.



TIP

After you configure a card, the diagram view gives you a glimpse of the settings within the card itself. For example, in the **Manage fields** area of the preceding image, you can see the first three fields being managed and the new names assigned to them. Each card has information relevant to it.

Aggregate

You can use the **Aggregate** transformation to calculate an aggregation (**Sum**, **Minimum**, **Maximum**, or **Average**) every time a new event occurs over a period of time. This operation also allows you to filter or slice the aggregation based on other dimensions in your data. You can have one or more aggregations in the same transformation.

To add an aggregation, select the transformation icon. Then connect an input, select the aggregation, add any filter or slice dimensions, and select the period of time over which the aggregation will be calculated. In this example, we're calculating the sum of the toll value by the state where the vehicle is from over the last 10 seconds.

The screenshot shows the Azure Data Factory designer interface. On the left, there is a transformation node labeled "Aggregate" with a blue border. Inside the node, there is a sub-node labeled "Sum of TollAmount". A curved arrow points from the main "Aggregate" node to the configuration pane on the right. The configuration pane has a header "Aggregate" with a close button "X". Below the header is a description: "Calculate an aggregation (like sum or average) each time a new event occurs." There is a button "+ Add aggregate function" with a plus sign icon. The main configuration area is titled "Sum of TollAmount" with a close button "X". It contains the following settings:

- Aggregation:** A dropdown menu set to "Sum".
- Field:** A dropdown menu set to "TollAmount".
- Filter by:** A dropdown menu set to "State".
- Aggregate values within the last:** Two input fields: "10" and "Second".

To add another aggregation to the same transformation, select **Add aggregate function**. Keep in mind that the filter or slice will apply to all aggregations in the transformation.

Join

Use the **Join** transformation to combine events from two inputs based on the field pairs that you select. If you don't select a field pair, the join will be based on time by default. The default is what makes this transformation different from a batch one.

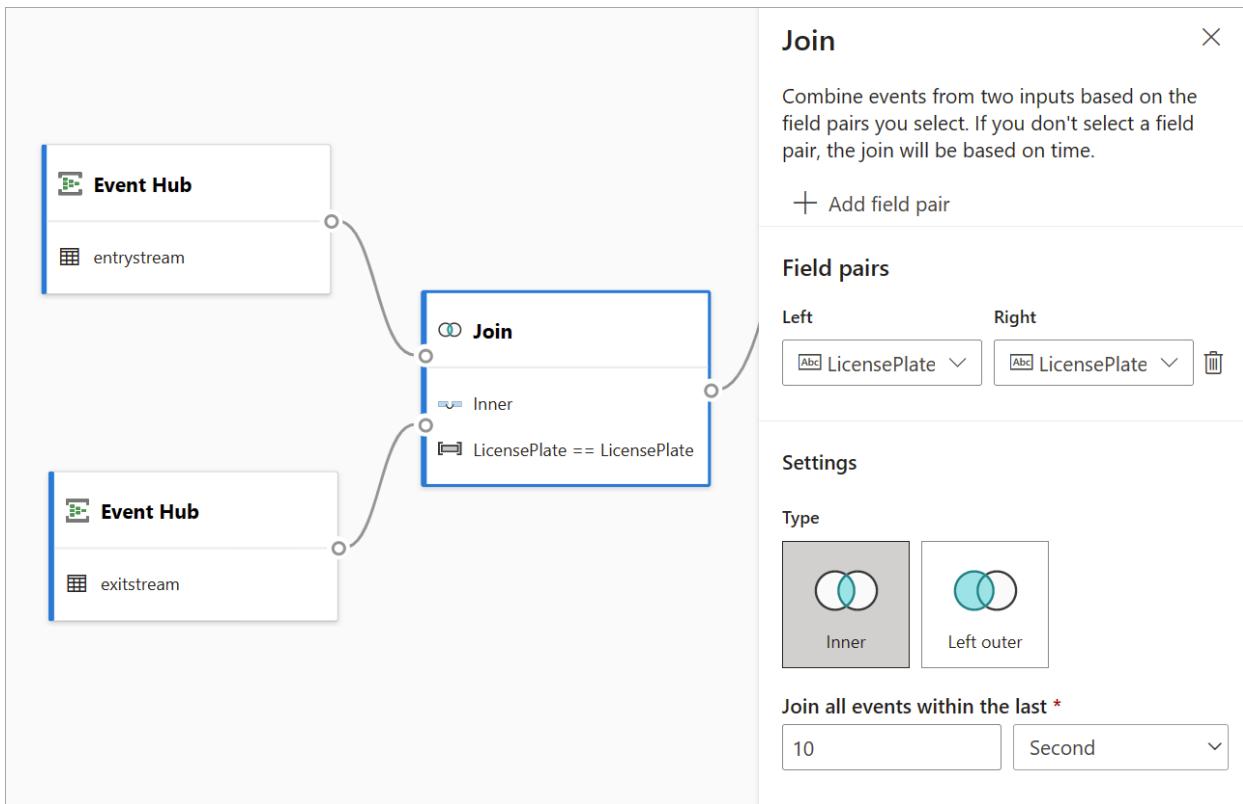
As with regular joins, you have different options for your join logic:

- **Inner join:** Include only records from both tables where the pair matches. In this example, that's where the license plate matches both inputs.
- **Left outer join:** Include all records from the left (first) table and only the records from the second one that match the pair of fields. If there's no match, the fields from the second input will be blank.

To select the type of join, select the icon for the preferred type on the side pane.

Finally, select over what period of time you want the join to be calculated. In this example, the join looks at the last 10 seconds. Keep in mind that longer the period is, the less frequent the output is--and the more processing resources you'll use for the transformation.

By default, all fields from both tables are included. Prefixes left (first node) and right (second node) in the output help you differentiate the source.



Group by

Use the **Group by** transformation to calculate aggregations across all events within a certain time window. You have the option to group by the values in one or more fields. It's similar to the **Aggregate** transformation but provides more options for aggregations. It also includes more complex time-window options. Also similar to **Aggregate**, you can add more than one aggregation per transformation.

The aggregations available in this transformation are: **Average**, **Count**, **Maximum**, **Minimum**, **Percentile** (continuous and discrete), **Standard Deviation**, **Sum**, and **Variance**.

To configure this transformation:

1. Select your preferred aggregation.
2. Select the field that you want to aggregate on.
3. Select an optional group-by field if you want to get the aggregate calculation over another dimension or category (for example, **State**).
4. Select your function for time windows.

To add another aggregation to the same transformation, select **Add aggregate function**. Keep in mind that the **Group by** field and the windowing function will apply to all aggregations in the transformation.

The screenshot shows the 'Group by' transformation configuration in the Azure Data Factory designer. On the left, there's a preview pane showing a single row of data with a 'Group by' operation applied. The main configuration area has the following sections:

- Aggregations:** A section for defining aggregate functions. It currently shows 'Count of LicensePlate' with 'Count' as the aggregate type and 'LicensePlate' as the field.
- Settings:** A section for configuring time windows and offsets.
 - Group aggregations by (optional):** A dropdown menu set to 'Select field'.
 - Time window:** Set to 'Tumbling'.
 - Duration:** Set to '10' with a unit of 'Second'.
 - Offset:** Set to 'Time interval' with a unit of 'Microsecond'.

A time stamp for the end of the time window is provided as part of the transformation output for reference.

A section later in this article explains each type of time window available for this transformation.

Union

Use the **Union** transformation to connect two or more inputs to add events with shared fields (with the same name and data type) into one table. Fields that don't match will be dropped and not included in the output.

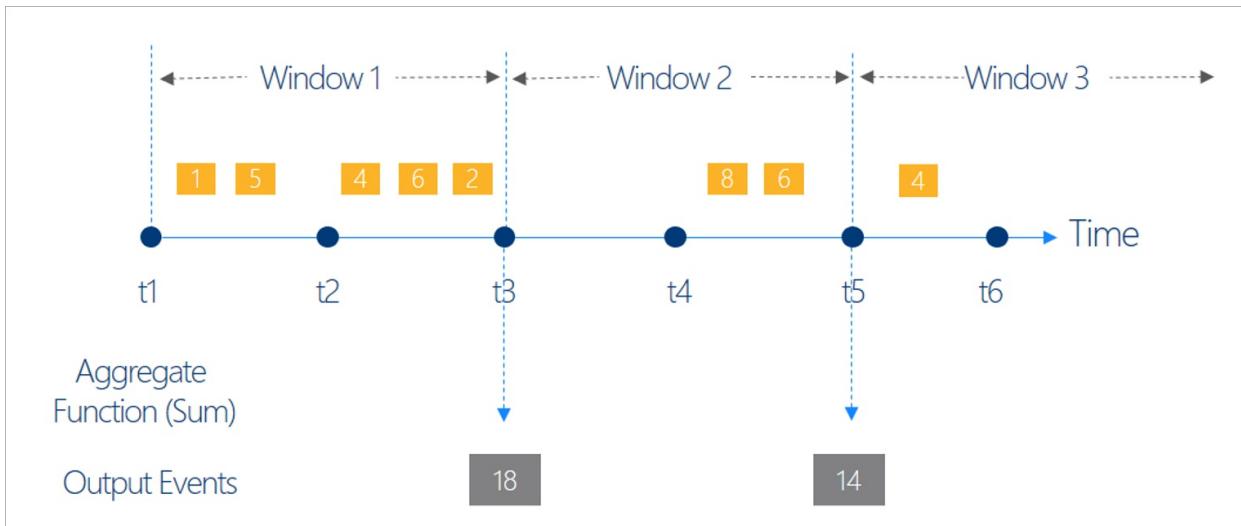
Set up time-window functions

Time windows are one of the most complex concepts in streaming data. This concept sits at the core of streaming analytics.

With streaming dataflows, you can set up time windows when you're aggregating data as an option for the **Group by** transformation.

NOTE

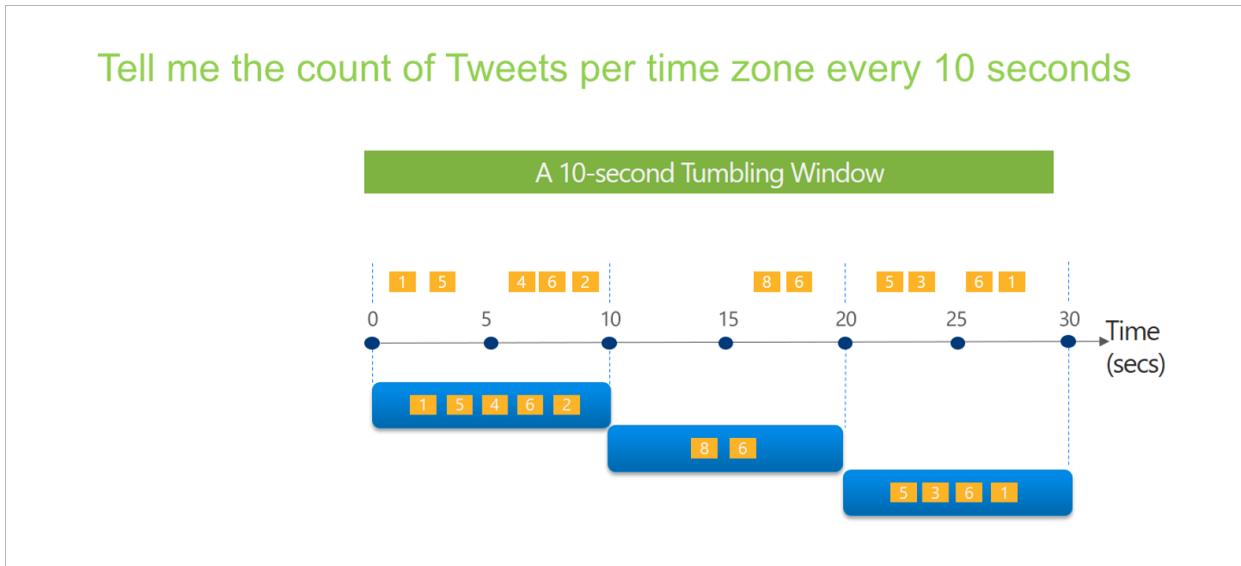
Keep in mind that all the output results for windowing operations are calculated at the end of the time window. The output of the window will be a single event that's based on the aggregate function. This event will have the time stamp of the end of the window, and all window functions are defined with a fixed length.



There are five kinds of time windows to choose from: tumbling, hopping, sliding, session, and snapshot.

Tumbling window

Tumbling is the most common type of time window. The key characteristics of tumbling windows are that they repeat, have the same time length, and don't overlap. An event can't belong to more than one tumbling window.



When you're setting up a tumbling window in streaming dataflows, you need to provide the duration of the window (same for all windows in this case). You also can provide an optional offset. By default, tumbling windows include the end of the window and exclude the beginning. You can use this parameter to change this behavior and include the events in the beginning of the window and exclude the ones in the end.

Time window [Learn more](#)

Tumbling

Duration

5

Second

Offset

Time interval

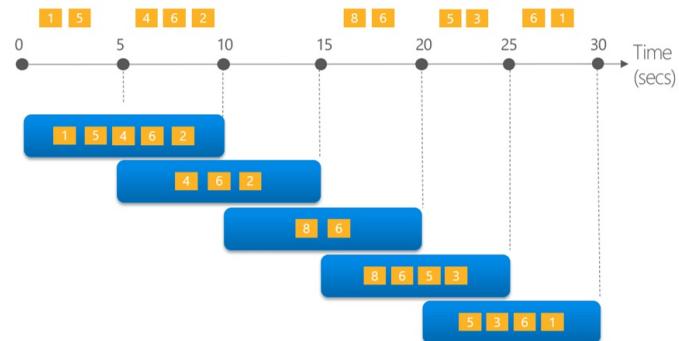
Microsecond

Hopping window

Hopping windows "hop" forward in time by a fixed period. You can think of them as tumbling windows that can overlap and be emitted more often than the window size. Events can belong to more than one result set for a hopping window. To make a hopping window the same as a tumbling window, you can specify the hop size to be the same as the window size.

Every 5 seconds give me the count of Tweets over the last 10 seconds

A 10-second Hopping Window with a 5-second "Hop"



When you're setting up a hopping window in streaming dataflows, you need to provide the duration of the window (same as with tumbling windows). You also need to provide the hop size, which tells streaming dataflows how often you want the aggregation to be calculated for the defined duration.

The offset parameter is also available in hopping windows for the same reason as in tumbling windows: to define the logic for including and excluding events for the beginning and end of the hopping window.

Time window [Learn more](#)

Hopping

Hop size

5

Second

Duration

10

Second

Offset

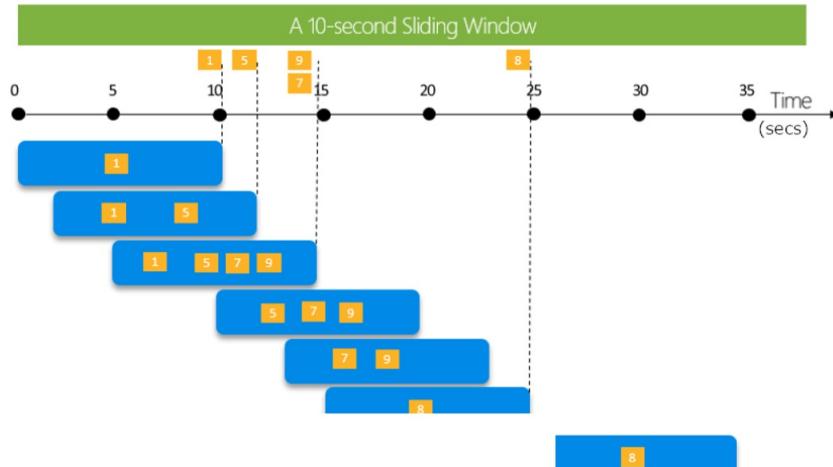
0

Second

Sliding window

Sliding windows, unlike tumbling or hopping windows, calculate the aggregation only for points in time when the content of the window actually changes. When an event enters or exits the window, the aggregation is calculated. So, every window has at least one event. Similar to hopping windows, events can belong to more than one sliding window.

Give me the count of Tweets for all topics which are Tweeted more than 10 times in the last 10 seconds



The only parameter that you need for a sliding window is the duration, because events themselves define when the window starts. No offset logic is necessary.

Time window [Learn more](#)

Sliding

Duration

10

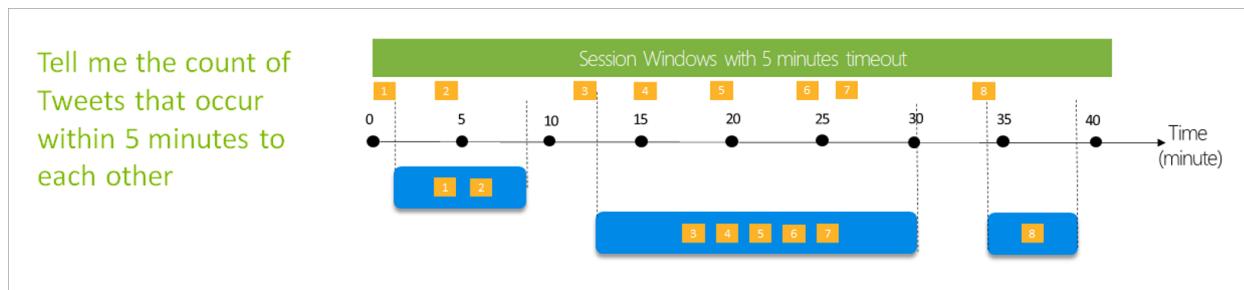
Second

Session window

Session windows are the most complex type. They group events that arrive at similar times, filtering out periods of time where there's no data. For this, it's necessary to provide:

- A timeout: how long to wait if there's no new data.
- A maximum duration: the longest time that the aggregation will be calculated if data keeps coming.

You can also define a partition, if you want.



You set up a session window directly on the side pane for the transformation. If you provide a partition, the aggregation will only group events together for the same key.

Time window [Learn more](#)

Session

Max duration

10 Minute

Timeout

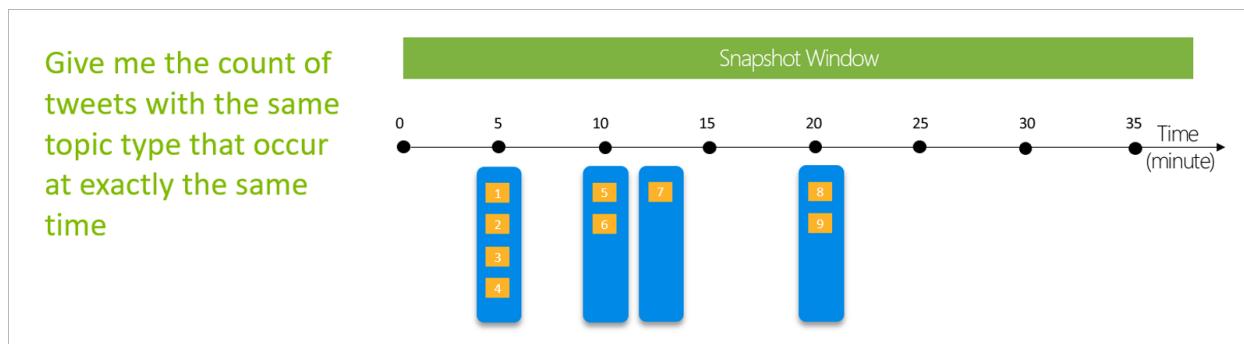
5 Minute

Partition by [Learn more](#)

None

Snapshot window

Snapshot windows groups events that have the same time stamp. Unlike other windows, a snapshot doesn't require any parameters because it uses the time from the system.



Define outputs

After you're ready with inputs and transformations, it's time to define one or more outputs. As of July of 2021, streaming dataflows support only one type of output: a Power BI table.

This output will be a dataflow table (that is, an entity) that you can use to create reports in Power BI Desktop. You need to join the nodes of the previous step with the output that you're creating to make it work. After that, all you need to do is name the table.

Output table

Name *

OutputTable

After you connect to your dataflow, this table will be available for you to create visuals that are updated in real time for your reports.

Data preview and errors

Streaming dataflows provide tools to help you author, troubleshoot, and evaluate the performance of your analytics pipeline for streaming data.

Let's start with the data preview.

Live data preview for inputs

When you're connecting to an event hub or IoT hub and selecting its card in the diagram view (the **Data Preview** tab), you'll get a live preview of data coming in if all the following are true:

- Data is being pushed.
- The input is configured correctly.
- Fields have been added.

As shown in the following screenshot, if you want to see or drill down into something specific, you can pause the preview (1). Or you can start it again if you're done.

You can also see the details of a specific record (a "cell" in the table) by selecting it and then selecting **Show/Hide details** (2). The screenshot shows the detailed view of a nested object in a record.

CarModel	EntryTime	State	TollAmount	Tag	TollId	LicensePlate	CarModel
{"Make": "Honda", "Model": "Accord", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:08.8255183Z	PA	4	527106791	3	WCE 7883	{"Make": "Honda", "Model": "Accord", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Peterbilt", "Model": "389", "CarModel": "C30", "VehicleType": "Truck", "VehicleWeight": 1}	2021-07-14T12:25:08.4555183Z	PA	18	857383105	1	SVA 6115	{"Make": "Peterbilt", "Model": "389", "CarModel": "C30", "VehicleType": "Truck", "VehicleWeight": 1}
{"Make": "Toyota", "Model": "Corolla", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:08.2165183Z	CA	5	891439315	3	SWF 3204	{"Make": "Toyota", "Model": "Corolla", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Keweenaw", "Model": "T680", "CarModel": "C30", "VehicleType": "Truck", "VehicleWeight": 1}	2021-07-14T12:25:08.0835183Z	OR	19	616574637	4	QEG 8107	{"Make": "Keweenaw", "Model": "T680", "CarModel": "C30", "VehicleType": "Truck", "VehicleWeight": 1}
{"Make": "Toyota", "Model": "RAV4", "CarModel": "C30", "VehicleType": "SUV", "VehicleWeight": 0}	2021-07-14T12:25:08.0825183Z	AL	5	505017174	4	SAV 3979	{"Make": "Toyota", "Model": "RAV4", "CarModel": "C30", "VehicleType": "SUV", "VehicleWeight": 0}
{"Make": "Volvo", "Model": "C30", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:07.1975179Z	AL	4	575867108	3	AEP 6554	{"Make": "Volvo", "Model": "C30", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Volvo", "Model": "C30", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:07.1835179Z	OR	4	977516738	2	GFE 1132	{"Make": "Volvo", "Model": "C30", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Ford", "Model": "Focus", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:07.0335179Z	AL	5	349011475	1	HJK 3255	{"Make": "Ford", "Model": "Focus", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Ford", "Model": "Mustang", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:06.9445179Z	NJ	6	801365828	3	KNL 1547	{"Make": "Ford", "Model": "Mustang", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Toyota", "Model": "Corolla", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:06.6655179Z	CA	5	253840721	3	KOE 4640	{"Make": "Toyota", "Model": "Corolla", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Ford", "Model": "Focus", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:05.9873207Z	TX	6	770159141	1	JYF 6978	{"Make": "Ford", "Model": "Focus", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Chevy", "Model": "Malibu", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:05.5363207Z	OR	4	967743115	4	KHU 6249	{"Make": "Chevy", "Model": "Malibu", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Peterbilt", "Model": "389", "CarModel": "C30", "VehicleType": "Truck", "VehicleWeight": 1}	2021-07-14T12:25:05.5373207Z	OR	26	245670976	4	UAO 4914	{"Make": "Peterbilt", "Model": "389", "CarModel": "C30", "VehicleType": "Truck", "VehicleWeight": 1}
{"Make": "Honda", "Model": "Accord", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:05.3593207Z	TX	4	152037239	2	DAK 7082	{"Make": "Honda", "Model": "Accord", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}
{"Make": "Chevy", "Model": "Malibu", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}	2021-07-14T12:25:05.3493207Z	CT	4	399702271	2	RVJ 4957	{"Make": "Chevy", "Model": "Malibu", "CarModel": "C30", "VehicleType": "Sedan", "VehicleWeight": 0}

Static preview for transformations and outputs

After you add and set up any steps in the diagram view, you can test their behavior by selecting the static data



After you do, streaming dataflows evaluate all transformation and outputs that are configured correctly. Streaming dataflows then display the results in the static data preview, as shown in the following image.

Make	Model	Weight	Time	License	Revenue
Honda	CRV	0	2021-07-14T12:36:28.061522Z	QDL 1438	6
Chevy	Malibu	0	2021-07-14T12:36:27.887522Z	WRY 1650	4
Peterbilt	389	2.675	2021-07-14T12:36:27.707522Z	UCO 7778	18
Toyota	Rav4	0	2021-07-14T12:36:27.549522Z	TIN 2948	6
Toyota	Rav4	0	2021-07-14T12:36:27.467522Z	UGR 8248	6
Toyota	4x4	0	2021-07-14T12:36:27.0179118Z	AIP 4184	5
Peterbilt	389	2.675	2021-07-14T12:36:27.0179118Z	IQN 1075	24
Honda	Civic	0	2021-07-14T12:36:26.7399118Z	HXM 8363	5
Kenworth	T680	4.32	2021-07-14T12:36:26.7269118Z	JFR 4978	23
Kenworth	T680	4.32	2021-07-14T12:36:26.1339118Z	IAG 2643	33
Toyota	Rav4	0	2021-07-14T12:36:25.4145366Z	CYB 1657	4
Volvo	V70	0	2021-07-14T12:36:25.2915366Z	SPC 6542	4
Volvo	C30	0	2021-07-14T12:36:25.2175366Z	DLR 4575	4
Volvo	S80	0	2021-07-14T12:36:25.2125366Z	LKK 7040	5

You can refresh the preview by selecting **Refresh static preview** (1). When you do this, streaming dataflows take new data from the input and evaluate all transformations and outputs again with any updates that you might have performed. The **Show/Hide details** option is also available (2).

Authoring errors

If you have any authoring errors or warnings, the **Authoring errors** tab (1) will list them, as shown in the following screenshot. The list includes details of the error or warning, the type of card (input, transformation, or output), the error level, and a description of the error or warning (2). When you select any of the errors or warnings, the respective card will be selected and the configuration side pane will open for you to make the needed changes.

The screenshot shows the Power BI Streaming Dataflows interface with the 'Authoring errors' tab selected (1). The main workspace displays a dataflow diagram with nodes like Event Hub, Group by, Union, and Output table. An IoT Hub node is shown with a 'Set-up required' status. The right pane shows the 'IoT Hub' configuration section with fields for connection string, consumer group, and manual entry. The bottom section shows a table of errors (2), with one specific error for an IoT Hub node highlighted.

Node ID	Node Type	Level	Error
04f6bfe-f5d2-07ef-8945-3e22841fc24	Union	Fatal	This operation requires 2 or more inputs to work.
04f6bfe-5d92-07ef-8945-3e22841fc24	Union	Fatal	This operation is missing an output to work.
0abc1cd13-a694-256a-e378-553710ad57cf	Output table	Fatal	You need to configure this operation or data source.
0abc1cd13-a694-256a-e378-553710ad57cf	Output table	Fatal	This operation is missing an input to work.
4b44bc44-b777-0ec1-f58b-d4903741a877	IoT Hub	Fatal	You need to configure this operation or data source. (2)
4b44bc44-b777-0ec1-f58b-d4903741a877	IoT Hub	Fatal	This operation is missing an output to work.

Runtime errors

The last available tab in the preview is **Runtime errors** (1), as shown in the following screenshot. This tab lists any errors in the process of ingesting and analyzing the streaming dataflow after you start it. For example, you might get a runtime error if a message came in corrupted, and the dataflow couldn't ingest it and perform the defined transformations.

Because dataflows might run for a long period of time, this tab offers the option to filter by time span and to download the list of errors and refresh it if needed (2).



Modify settings for streaming dataflows

As with regular dataflows, settings for streaming dataflows can be modified depending on the needs of owners and authors. The following settings are unique to streaming dataflows. For the rest of the settings, because of the shared infrastructure between the two types of dataflows, you can assume that the use is the same.

The screenshot shows the 'Dataflows' tab selected in the navigation bar. The main content area is titled 'Settings for Toll'. It displays the last modified user ('Miguel Martinez'), the last refresh time ('Wed Jul 07 2021 16:46:08 GMT-0700 (Pacific Daylight Time)'), and a link to 'Refresh history'. The settings are organized into sections: 'Gateway connection', 'Data source credentials' (EventHub, Edit credentials, Show in lineage view), 'Sensitivity label', 'Enhanced compute engine settings', 'Endorsement', 'Retention Duration' (set to 1 Day), and 'Notifications'. Buttons for 'Apply' and 'Discard' are at the bottom.

- **Refresh history:** Because streaming dataflows run continuously, the refresh history shows only information about when the dataflow was started, when it was canceled, or when it failed (with details and error codes when applicable). This information is similar to what appears for regular dataflows. You can use this information to troubleshoot issues or to provide Power BI support with requested details.
- **Data source credentials:** This setting shows the inputs that have been configured for the specific streaming dataflow.
- **Enhanced compute engine settings:** Streaming dataflows need the enhanced compute engine to provide real-time visuals, so this setting is turned on by default and can't be changed.
- **Retention duration:** This setting is specific to streaming dataflows. Here you can define how long you want to keep real-time data to visualize in reports. Historical data is saved by default in Azure Blob Storage. This setting is specific to the real-time side of your data (hot storage). The minimum value here is 1 day or 24 hours.

IMPORTANT

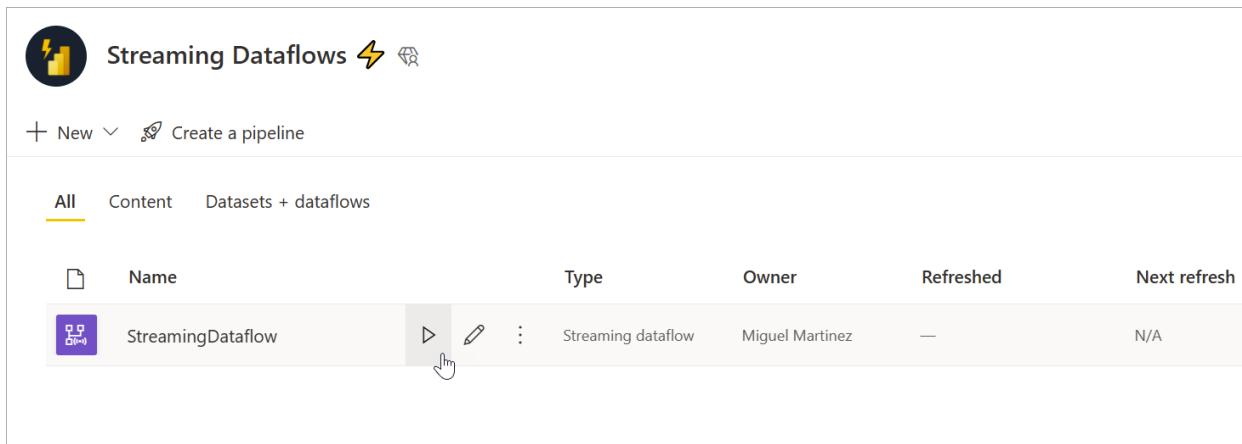
The amount of hot data stored by this retention duration directly influences the performance of your real-time visuals when you're creating reports on top of this data. The more retention you have here, the more your real-time visuals in reports can be affected by low performance. If you need to perform historical analysis, we recommend that you use the cold storage provided for streaming dataflows.

Run and edit a streaming dataflow

After you save and configure your streaming dataflow, everything is ready for you to run it. You can then start ingesting data into Power BI with the streaming analytics logic that you've defined.

Run your streaming dataflow

To start your streaming dataflow, first save your dataflow and go to the workspace where you created it. Hover over the streaming dataflow and select the play button that appears. A pop-up message tells you that the streaming dataflow is being started.



The screenshot shows the 'Streaming Dataflows' workspace in Power BI. At the top, there's a navigation bar with a plus icon for 'New', a 'Create a pipeline' button, and tabs for 'All', 'Content', and 'Datasets + dataflows'. Below the navigation bar, a table lists a single dataflow:

Name	Type	Owner	Refreshed	Next refresh
StreamingDataflow	Streaming dataflow	Miguel Martinez	—	N/A

A hand cursor is hovering over the play button icon next to the dataflow name. The play button is a grey circle with a white triangle pointing right, and there's a small blue progress bar underneath it.

NOTE

It might take up to five minutes for data to start being ingested and for you to see data coming in to create reports and dashboards in Power BI Desktop.

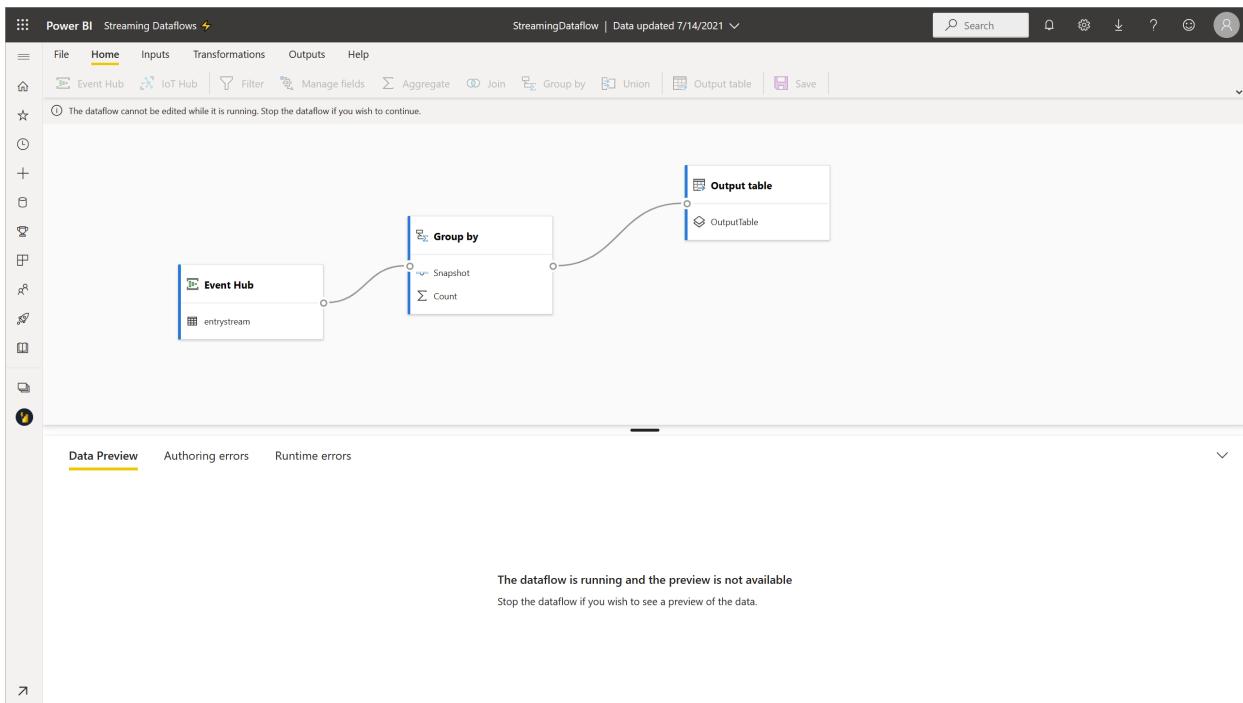
Edit your streaming dataflow

While a streaming dataflow is running, it *can't be edited*. But you can go into a streaming dataflow that's in a running state and see the analytics logic that the dataflow is built on.

When you go into a running streaming dataflow, all edit options are disabled and a message is displayed: "The dataflow cannot be edited while it is running. Stop the dataflow if you wish to continue." The data preview is disabled too.

To edit your streaming dataflow, you have to stop it. *A stopped dataflow will result in missing data.*

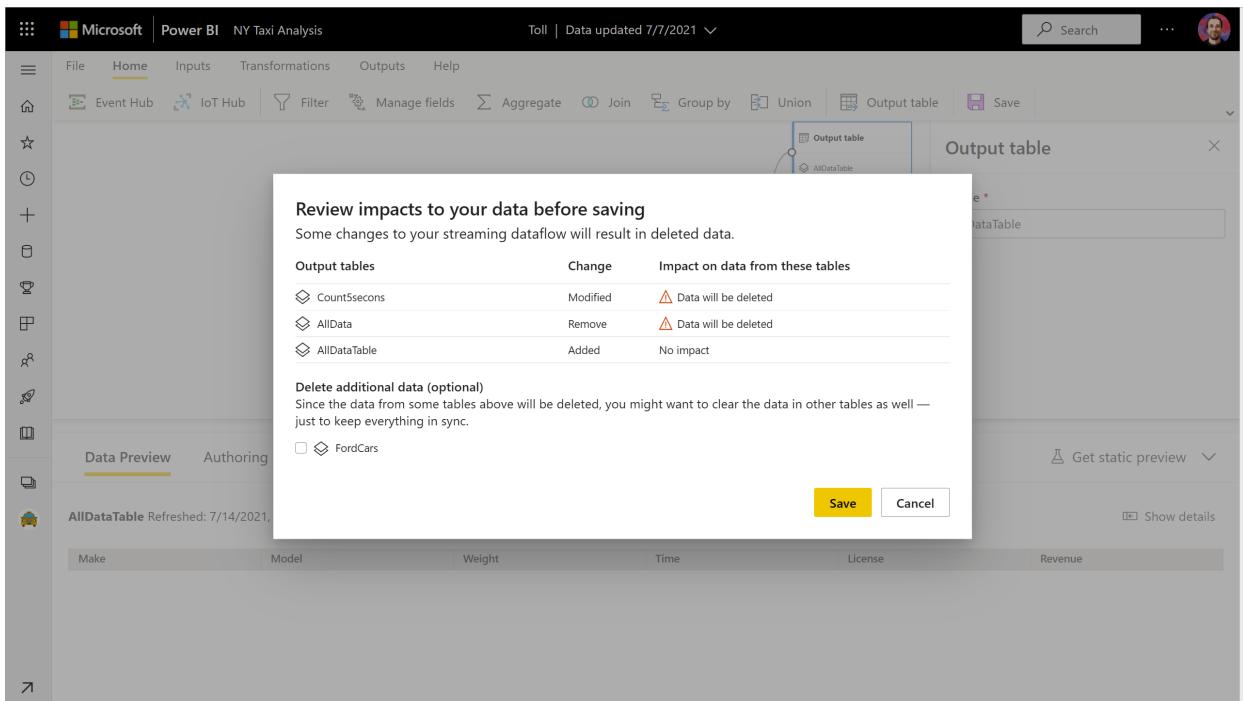
The only experience available while a streaming dataflow is running is the **Runtime errors** tab, where you can monitor the behavior of your dataflow for any dropped messages and similar situations.



Consider data storage when editing your dataflow

When you're editing a dataflow, you need to account for other considerations. Similar to any changes in a schema for regular dataflows, if you make changes to an output table, you'll lose data that has already been pushed and saved to Power BI. The interface provides clear information about the consequences of any of these changes in your streaming dataflow, along with choices for changes that you make before saving.

This experience is better shown with an example. The following screenshot shows the message you would get after adding a column to one table, changing the name for a second table, and leaving a third table the same as it was before.



In this example, the data already saved in both tables that had schema and name changes will be deleted if you save the changes. For the table that stayed the same, you get the option to delete any old data and start from scratch, or save it for later analysis together with new data that comes in.

Keep in mind these nuances when editing your streaming dataflow, especially if you need historical data available later for further analysis.

Consume a streaming dataflow

After your streaming dataflow is running, you're ready to start creating content on top of your streaming data. There are no structural changes compared to what you have to currently do to create reports that are updated in real time. But there are some nuances and updates to consider, so you can take advantage of this new type of data preparation for streaming data.

Set up data storage

As we mentioned before, streaming dataflows save data in the following two locations. The use of these sources depends on what type of analysis you're trying to do.

- **Hot storage (real-time analysis):** As data comes in to Power BI from streaming dataflows, data is stored in a hot location for you to access with real-time visuals. How much data is saved in this storage depends on the value that you defined for **Retention duration** in the streaming dataflow settings. The default (and minimum) is 24 hours.
- **Cold storage (historical analysis):** Any time period that doesn't fall in the period that you defined for **Retention duration** is saved in cold storage (blobs) in Power BI for you to consume if needed.

NOTE

There is overlap between these two data storage locations. If you need to use both locations in conjunction (for example, day-over-day percentage change), you might have to deduplicate your records. It depends on the time intelligence calculations that you're making and the retention policy.

Connect to streaming dataflows from Power BI Desktop

With the July 2021 release of Power BI Desktop, a new connector called **Dataflows** is available for you to use. As part of this new connector, for streaming dataflows, you'll see two tables that match the data storage previously described.

To connect to your data for streaming dataflows:

1. Go to **Get Data**, search for **power platform**, and then select the **Dataflows** connector.

Get Data

X

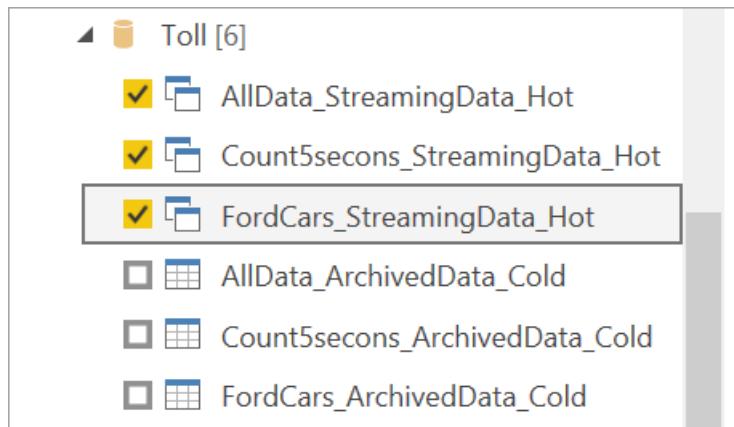
The screenshot shows the 'Get Data' interface in Power BI. On the left, there's a sidebar with a search bar at the top. Below it are categories: All, File, Database, Power Platform, Azure, Online Services, and Other. 'Power Platform' is currently selected. Under 'Power Platform', there are sub-options: Power BI datasets, Power BI dataflows, Common Data Service (Legacy), Dataverse, and Dataflows. 'Dataflows' is also highlighted. At the bottom of the sidebar, there are links for 'Certified Connectors' and 'Template Apps', and buttons for 'Connect' and 'Cancel'.

2. Sign in with your Power BI credentials.
3. Select workspaces. Look for the one that contains your streaming dataflow and select that dataflow. (In this example, the streaming dataflow is called **Toll**.)
4. Notice that all your output tables appear twice: one for streaming data (hot) and one for archived data (cold). You can differentiate them by the labels added after the table names and by the icons.

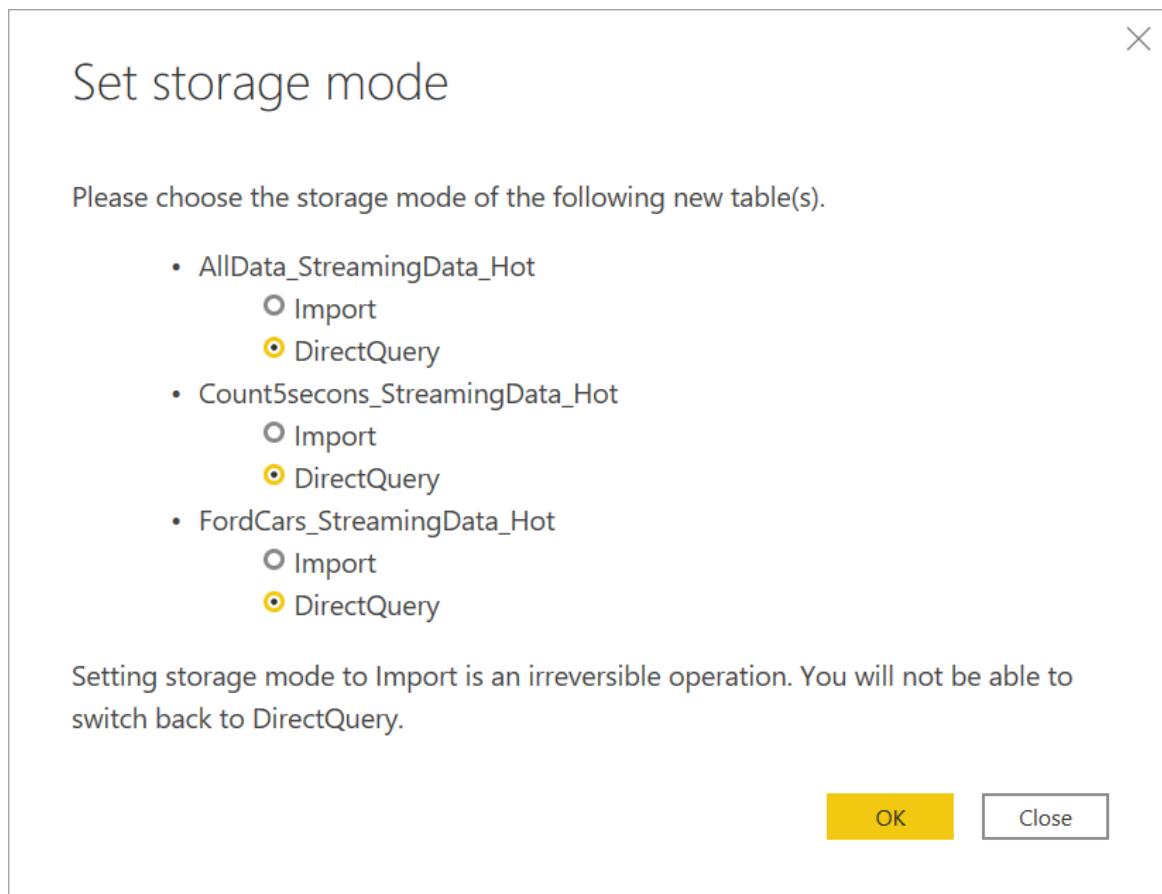
The screenshot shows the Power BI data browser. A dataflow named 'Toll [6]' is expanded, revealing six tables:

- AllData_StreamingData_Hot
- Count5secs_StreamingData_Hot
- FordCars_StreamingData_Hot
- AllData_ArchivedData_Cold
- Count5secs_ArchivedData_Cold
- FordCars_ArchivedData_Cold

5. Connect to the streaming data. The archived data case is the same, only available in import mode. Select the tables that include the labels **Streaming** and **Hot**, and then select **Load**.



6. When you're asked to choose a storage mode, select **DirectQuery** if your goal is to create real-time visuals.



Now you can create visuals, measures, and more, by using the features available in Power BI Desktop.

NOTE

The regular Power BI dataflow connector is still available and will work with streaming dataflows with two caveats:

- It only allows you to connect to hot storage.
- The data preview in the connector does not work with streaming dataflows.

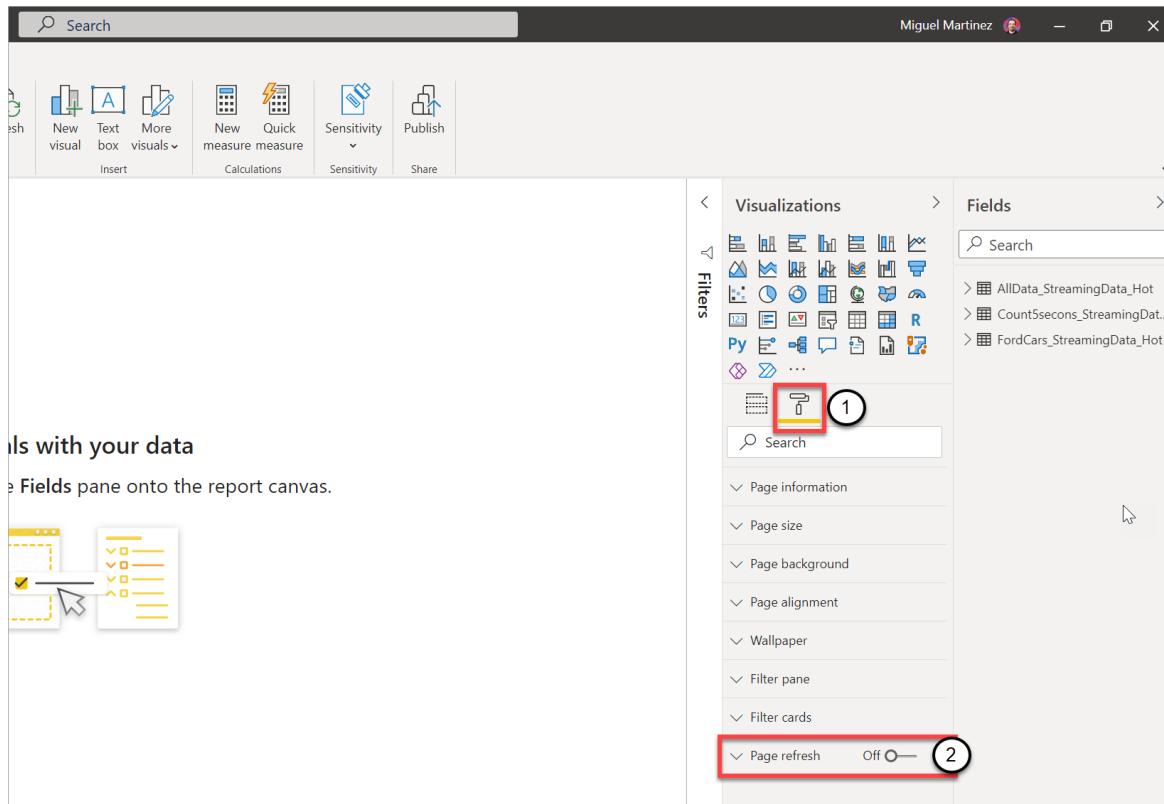
Turn on automatic page refresh for real-time visuals

After your report is ready and you've added all the content that you want to share, the only step left is to make sure your visuals are updated in real time. For this, you can use a feature called *automatic page refresh*. This feature allows you to refresh visuals from a DirectQuery source as often as one second.

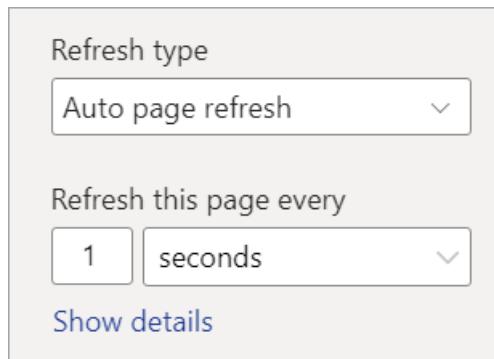
For more information about the feature, see [Automatic page refresh in Power BI](#). That information includes how to use it, how to set it up, and how to contact your admin if you're having trouble. Here are the basics on how to

set it up:

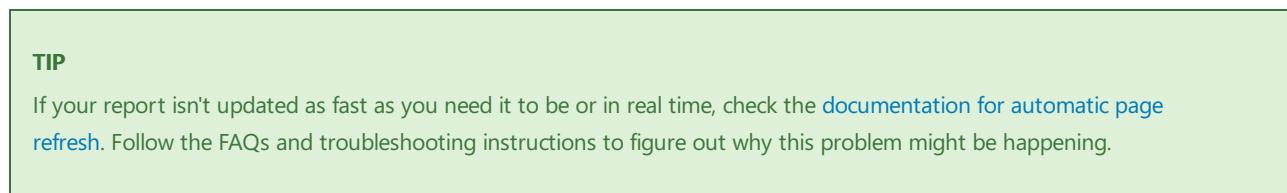
1. Go to the report page where you want the visuals to be updated in real time.
2. Clear any visual on the page. If possible, select the background of the page.
3. Go to the format pane (1) and turn on the **Page refresh** toggle (2).



4. Set up your desired frequency (up to every second if your admin has allowed it) and enjoy the real-time updates to your visuals.



5. To share a real-time report, first publish back to the Power BI service. Then you can set up your dataflow credentials for the dataset and share.



Considerations and limitations

General limitations

- A Power BI Premium subscription (capacity or PPU) is required for creating and running streaming dataflows.

- Only one type of dataflow is allowed per workspace.
- Linking regular and streaming dataflows is not possible.
- Capacities smaller than A3 don't allow the use of streaming dataflows.
- If dataflows or the enhanced calculation engine is not enabled in a tenant, you can't create or run streaming dataflows.
- Workspaces connected to a storage account are not supported.
- Each streaming dataflow can provide up to 1 megabyte per second of throughput.

Availability

The preview of streaming dataflows is not available in the following regions:

- Central India
- Germany North
- Norway East
- Norway West
- UAE Central
- South Africa North
- South Africa West
- Switzerland North
- Switzerland West
- Brazil Southeast

Licensing

The number of streaming dataflows allowed per tenant depends on the license being used:

- For regular capacities, use the following formula to calculate the maximum number of streaming dataflows allowed in a capacity:

Maximum number of streaming dataflows per capacity = vCores in the capacity x 5

For example, P1 has 8 vCores: $8 * 5 = 40$ streaming dataflows.

- For Premium Per User, one streaming dataflow is allowed per user. If another user wants to consume a streaming dataflow in a PPU workspace, they'll need a PPU license too.

Dataflow authoring

When you're authoring streaming dataflows, be mindful of the following considerations:

- Streaming dataflows can be modified only by their owners, and only if they're not running.
- Streaming dataflows are not available in **My Workspace**.

Connecting from Power BI Desktop

You can access cold storage only by using the **Power Platform dataflows (Beta)** connector available starting in the July 2021 Power BI Desktop update. The existing Power BI dataflow connector allows only connections to streaming data (hot) storage. The connector's data preview doesn't work.

Next steps

This article provided an overview of self-service streaming data preparation by using streaming dataflows. The following articles provide information about how to test this capability and how to use other streaming data features in Power BI:

- [Build a sample IoT solution to test streaming dataflows with one click](#)
- [Use the Azure Raspberry Pi simulator and the Free tier of IoT Hub to test streaming dataflows](#)

- Set up push and streaming datasets in Power BI
- Learn more about automatic page refresh

Develop solutions with dataflows

3/30/2022 • 11 minutes to read • [Edit Online](#)

Power BI *dataflows* are an enterprise-focused data prep solution that enables an ecosystem of data that's ready for consumption, reuse, and integration. This article presents some common scenarios. Links to articles and other information help you to understand and use dataflows to their full potential.

Get access to Premium features of dataflows

Power BI dataflows in Premium capacities provide many key features that help achieve greater scale and performance for your dataflows, such as:

- Advanced compute, which accelerates ETL performance and provides DirectQuery capabilities.
- Incremental refresh, which lets you load data that's changed from a source.
- Linked entities, which you can use to reference other dataflows.
- Computed entities, which you can use to build composable building blocks of dataflows that contain more business logic.

For these reasons, we recommend that you use dataflows in a Premium capacity whenever possible. Dataflows used in a Power BI Pro license can be used for simple, small-scale use cases.

Solution

Getting access to these [Premium features of dataflows](#) is possible in two ways:

- Designate a **Premium capacity** to a given workspace and bring your own Pro license to author dataflows here.
- Bring your own **Premium per user (PPU)** license, which requires other members of the workspace to also possess a PPU license.

You can't consume PPU dataflows (or any other content) outside the PPU environment (such as in Premium or other SKUs or licenses).

For Premium capacities, your consumers of dataflows in Power BI Desktop don't need explicit licenses to consume and publish to Power BI. But to publish to a workspace or share a resulting dataset, you'll need at least a Pro license.

For PPU, everyone who creates or consumes PPU content must have a PPU license. This requirement varies from the rest of Power BI in that you need to explicitly license everyone with PPU and you can't mix Free, Pro, or even Premium capacities with PPU content unless you migrate the workspace to a Premium capacity.

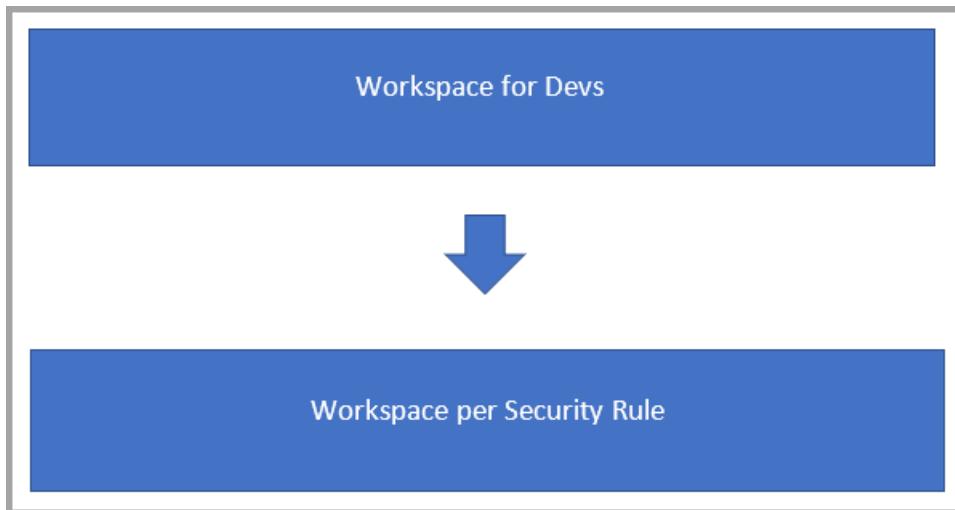
Choosing a model typically depends on your organization's size and goals, but the following guidelines apply.

TEAM TYPE	PREMIUM PER CAPACITY	PREMIUM PER USER
>5,000 users	✓	
<5,000 users		✓

For small teams, PPU can bridge the gap between Free, Pro, and Premium per capacity. If you have larger needs, using a Premium capacity with users who have Pro licenses is the best approach.

Create user dataflows with security applied

Imagine that you need to create dataflows for consumption but have security requirements:



In this scenario, you likely have two types of workspaces:

- Back-end workspaces where you develop dataflows and build out the business logic.
- User workspaces where you want to expose some dataflows or tables to a particular group of users for consumption:
 - The user workspace contains linked tables that point to the dataflows in the back-end workspace.
 - Users have viewer access to the consumer workspace and no access to the back-end workspace.
 - When a user uses Power BI Desktop to access a dataflow in the user workspace, they can see the dataflow. But because the dataflow appears empty in the Navigator, the linked tables don't show.

Understand linked tables

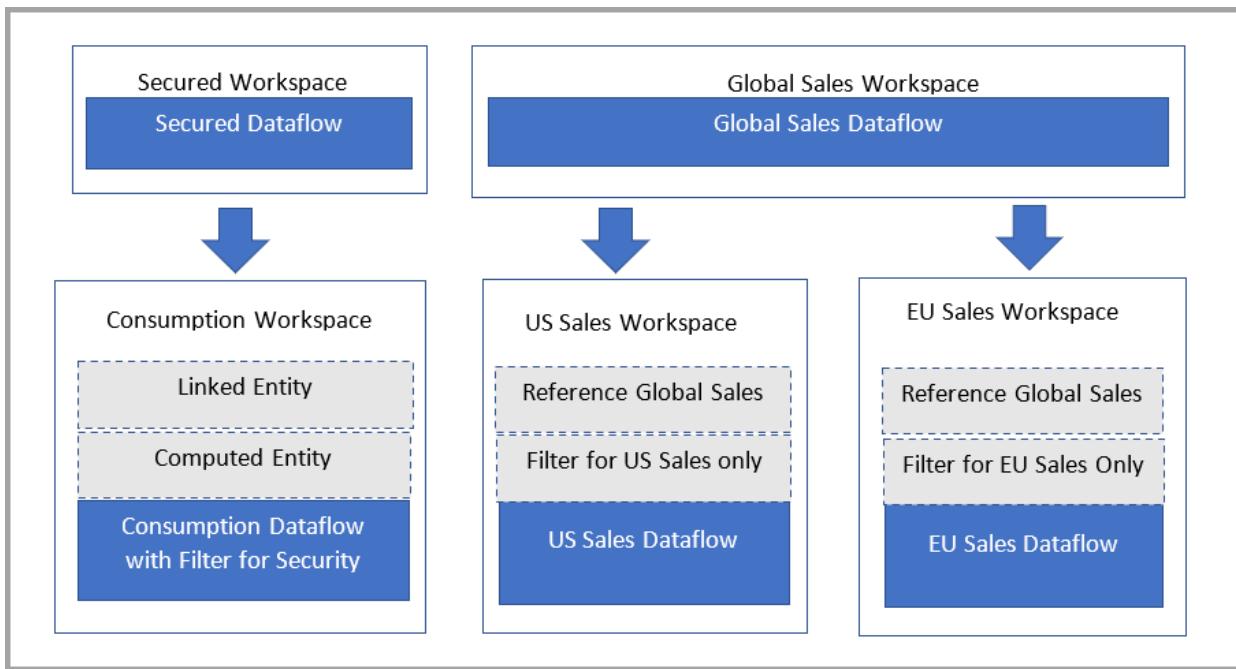
Linked tables are simply a pointer to the original dataflow tables, and they inherit the permission of the source. If Power BI allowed the linked table to use the destination permission, any user might circumvent the source permission by creating a linked table in the destination that points to the source.

Solution: Use computed tables

If you have access to Power BI Premium, you can create a computed table in the destination that refers to the linked table, which has a copy of the data from the linked table. You can remove columns through projections and remove rows through filters. The user with permission on the destination workspace can access data through this table.

Lineage for privileged individuals also shows the referenced workspace and allows users to link back to fully understand the parent dataflow. For those users who aren't privileged, privacy is still respected. Only the name of the workspace is shown.

The following diagram illustrates this setup. On the left is the architectural pattern. On the right is an example that shows sales data split and secured by region.



Reduce refresh times for dataflows

Imagine you have a large dataflow, but you want to build datasets off of that dataflow and decrease the time required to refresh it. Typically, refreshes take a long time to complete from the data source to dataflows to the dataset. Lengthy refreshes are difficult to manage or maintain.

Solution: Use tables with Enable Load explicitly configured for referenced tables and don't disable load

Power BI supports simple orchestration for dataflows, as defined in [understanding and optimizing dataflows refresh](#). Taking advantage of orchestration requires explicitly having any downstream dataflows configured to *Enable Load*.

Disabling load typically is appropriate only when the overhead of loading more queries cancels the benefit of the entity with which you're developing.

While disabling load means Power BI doesn't evaluate that given query, when used as ingredients, that is, referenced in other dataflows, it also means that Power BI doesn't treat it as an existing table where we can provide a pointer to and perform folding and query optimizations. In this sense, performing transformations such as a join or merge is merely a join or merge of two data source queries. Such operations can have a negative effect on performance because Power BI must fully reload already computed logic again and then apply any more logic.

To simplify the query processing of your dataflow and ensure any engine optimizations are taking place, enable load and ensure that the compute engine in Power BI Premium dataflows is set at the default setting, which is **Optimized**.

Enabling load also enables you to keep the complete view of lineage because Power BI considers a non-enabled load dataflow as a new item. If lineage is important to you, don't disable load for entities or dataflows connected to other dataflows.

Reduce refresh times for datasets

Imagine you have a dataflow that's large, but you want to build datasets off of it and decrease the orchestration. Refreshes take a long time to complete from the data source to dataflows to datasets, which adds increased latency.

Solution: Use DirectQuery dataflows

DirectQuery can be used whenever a workspace's enhanced compute engine (ECE) setting is configured

explicitly to **On**. This setting is helpful when you have data that doesn't need to be loaded directly into a Power BI model. If you're configuring the ECE to be **On** for the first time, the changes that allow DirectQuery will occur during the next refresh. You'll need to refresh it when you enable it to have changes take place immediately. Refreshes on the initial dataflow load can be slower because Power BI writes data to both storage and a managed SQL engine.

To summarize, using DirectQuery with dataflows enables the following enhancements to your Power BI and dataflows processes:

- **Avoid separate refresh schedules:** DirectQuery connects directly to a dataflow, which removes the need to create an imported dataset. As such, using DirectQuery with your dataflows means you no longer need separate refresh schedules for the dataflow and the dataset to ensure your data is synchronized.
- **Filtering data:** DirectQuery is useful for working on a filtered view of data inside a dataflow. If you want to filter data, and in this way work with a smaller subset of the data in your dataflow, you can use DirectQuery (and the ECE) to filter dataflow data and work with the filtered subset you need.

Generally, using DirectQuery trades up-to-date data in your dataset with slower report performance compared to import mode. Consider this approach only when:

- Your use case requires low latency data coming from your dataflow.
- The dataflow data is large.
- An import would be too time consuming.
- You're willing to trade cached performance for up-to-date data.

Solution: Use the dataflow connector to enable query folding and incremental refresh for import

The unified Dataflows connector can significantly reduce evaluation time for steps performed over computed entities, such as performing joins, distinct, filters, and group by operations. There are two specific benefits:

- Downstream users connecting to the Dataflows connector in Power BI Desktop can take advantage of better performance in authoring scenarios because the new connector supports query folding.
- Dataset refresh operations can also fold to the enhanced compute engine, which means even incremental refresh from a dataset can fold to a dataflow. This capability improves refresh performance and potentially decreases latency between refresh cycles.

To enable this feature for any Premium dataflow, make sure the [compute engine](#) is explicitly set to **On**. Then use the Dataflows connector in Power BI Desktop. You must use the August 2021 version of Power BI Desktop or later to take advantage of this feature.

To use this feature for existing solutions, you must be on a Premium or Premium Per User subscription. You might also need to make some changes to your dataflow as described in [Using the enhanced compute engine](#). You must update any existing Power Query queries to use the new connector by replacing `PowerBI.Dataflows` in the **Source** section with `PowerPlatform.Dataflows`.

Complex dataflow authoring in Power Query

Imagine you have a dataflow that's millions of rows of data, but you want to build complex business logic and transformations with it. You want to follow best practices for working with large dataflows. You also need the dataflow previews to perform quickly. But, you have dozens of columns and millions of rows of data.

Solution: Use Schema view

You can [use Schema view](#), which is designed to optimize your flow when you work on schema-level operations by putting your query's column information front and center. Schema view provides contextual interactions to shape your data structure. Schema view also provides lower latency operations because it only requires the column metadata to be computed and not the complete data results.

Work with bigger data sources

Imagine you run a query on the source system, but you don't want to provide direct access to the system or democratize access. You plan to put it in a dataflow.

Solution 1: Use a view for the query or optimize the query

Using an optimized data source and query is your best option. Often, the data source operates best with queries intended for it. Power Query has advanced query-folding capabilities to delegate these workloads. Power BI also provides step-folding indicators in Power Query Online. Read more about types of indicators in the [step-folding indicators documentation](#).

Solution 2: Use Native Query

You can also use the `Value.NativeQuery()` M function. You set `EnableFolding=true` in the third parameter. Native Query is documented on [this website](#) for the Postgres connector. It also works for the SQL Server connector.

Solution 3: Break the dataflow into ingestion and consumption dataflows to take advantage of the ECE and Linked Entities

By breaking a dataflow into separate ingestion and consumption dataflows, you can take advantage of the ECE and Linked Entities. You can learn more about this pattern and others in the [best practices documentation](#).

Ensure customers use dataflows whenever possible

Imagine you have many dataflows that serve common purposes, such as conformed dimensions like customers, data tables, products, and geographies. Dataflows are already available in the ribbon for Power BI. Ideally, you want customers to primarily use the dataflows you've created.

Solution: Use endorsement to certify and promote dataflows

To learn more about how endorsement works, see [Endorsement: Promoting and certifying Power BI content](#).

Programmability and automation in Power BI dataflows

Imagine you have business requirements to automate imports, exports, or refreshes, and more orchestration and actions outside of Power BI. You have a few options to enable doing so, as described in the following table.

TYPE	MECHANISM
Use the PowerAutomate templates .	No-code
Use automation scripts in PowerShell .	Automation scripts
Build your own business logic by using the APIs .	Rest API

For more information about refresh, see [Understanding and optimizing dataflows refresh](#).

Ensure you protect data assets downstream

You can use sensitivity labels to apply a data classification and any rules you configured on downstream artifacts that connect to your dataflows. To learn more about sensitivity labels, see [Microsoft Information Protection sensitivity labels in Power BI](#). To review inheritance, see [Sensitivity label downstream inheritance in Power BI](#).

Multi-geo support

Many customers today have a need to meet data sovereignty and residency requirements. You can complete a manual configuration to your dataflows workspace to be multi-geo.

Dataflows support multi-geo when they use the bring-your-own-storage-account feature. This feature is described in [Configuring dataflow storage to use Azure Data Lake Gen 2](#). The workspace must be empty prior to attach for this capability. With this specific configuration, you can store dataflow data in specific geo regions of your choice.

Ensure you protect data assets behind a virtual network

Many customers today have a need to secure your data assets behind a private endpoint. To do so, use virtual networks and a gateway to stay compliant. The following table describes the current virtual network support and explains how to use dataflows to stay compliant and protect your data assets.

SCENARIO	STATUS
Read virtual network data sources through an on-premises gateway.	Supported through an on-premises gateway
Write data to a sensitivity label account behind a virtual network by using an on-premises gateway.	Not yet supported

Next steps

The following articles provide more information about dataflows and Power BI:

- [Introduction to dataflows and self-service data prep](#)
- [Create a dataflow](#)
- [Configure and consume a dataflow](#)
- [Premium features of dataflows](#)
- [AI with dataflows](#)
- [Dataflows considerations and limitations](#)
- [Dataflows best practices](#)

Using Azure Log Analytics in Power BI (Preview)

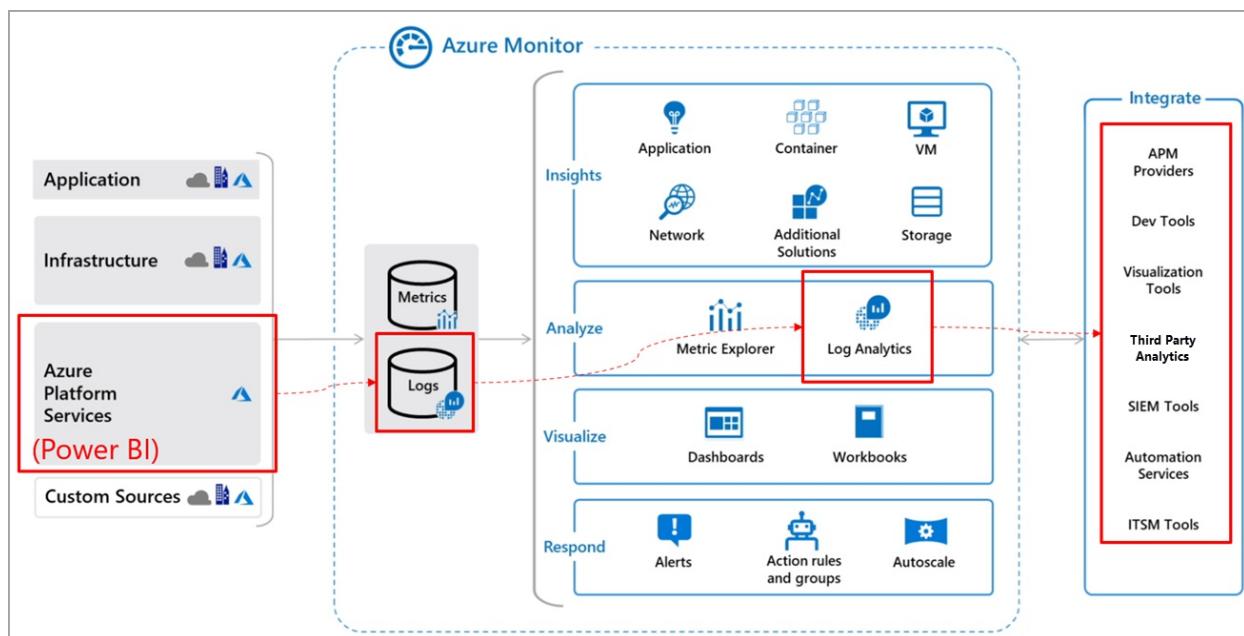
3/30/2022 • 3 minutes to read • [Edit Online](#)

Power BI is integrating with Azure Log Analytics (LA) to enable administrators and Premium workspace owners to configure a Log Analytics connection to their Power BI subscription. This article describes how the integration between Log Analytics and Power BI works, and provides examples of how you can use Azure Log Analytics in your Power BI Premium subscription.

Azure Log Analytics (LA) is a service within [Azure Monitor](#) which Power BI uses to save activity logs. The Azure Monitor suite lets you collect, analyze, and act on telemetry data from your Azure and on-premises environments. It offers long-term storage, an ad-hoc query interface and API access to allow data export and integration with other systems.

The Power BI integration with Log Analytics exposes events from the Analysis Services engine. The events are derived from existing [diagnostic logs available for Azure Analysis Services](#).

Once connected to Power BI, data is sent continuously and is available in Log Analytics in approximately 5 minutes. The following diagram shows how Azure Monitor operates, with the path taken by Power BI highlighted.



The following sections describe the integration of Azure Log Analytics with Power BI, the requirements necessary to connect Azure Log Analytics to Power BI, and considerations to keep in mind.

Available scopes for logging

An Azure Log Analytics connection is currently supported for Premium workspaces in Power BI. The following table provides more information about the Workspace-level Log Analytics configuration.

CONFIGURATION LEVEL	ROLE / PERMISSION	DESTINATION TABLE IN LOG ANALYTICS	DETAILS
---------------------	-------------------	------------------------------------	---------

CONFIGURATION LEVEL	ROLE / PERMISSION	DESTINATION TABLE IN LOG ANALYTICS	DETAILS
Workspace	Power BI workspace owner, or Log Analytics workspace owner	PowerBIDatasetsWorkspace	<ul style="list-style-type: none"> - Must be allowed by the tenant administrator - Logs activity from the workspace only - Premium workspaces only - Workspace v2 only

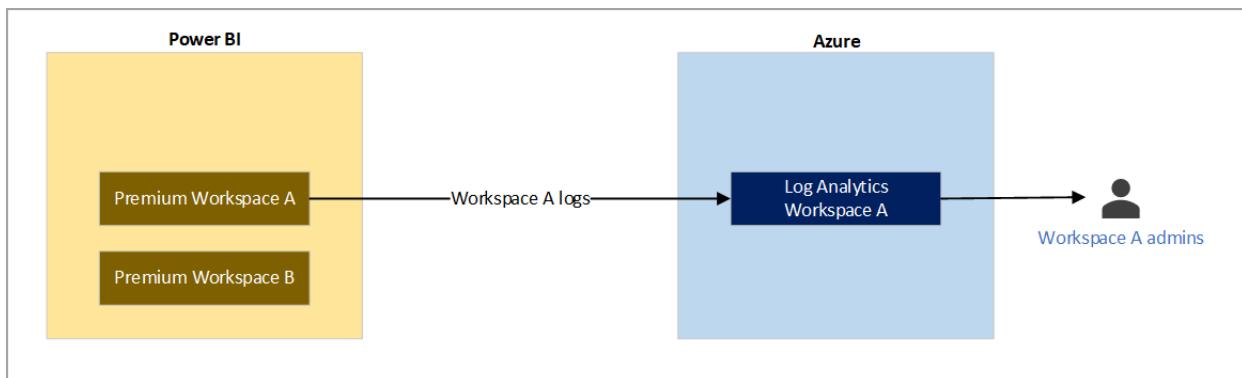
The following section provides examples of how you might put logging to use in Power BI.

Examples of logging scenarios

This section provides some examples of how you might configure Log Analytics for Power BI, and how selections you make will impact what is logged, and how the information is provided.

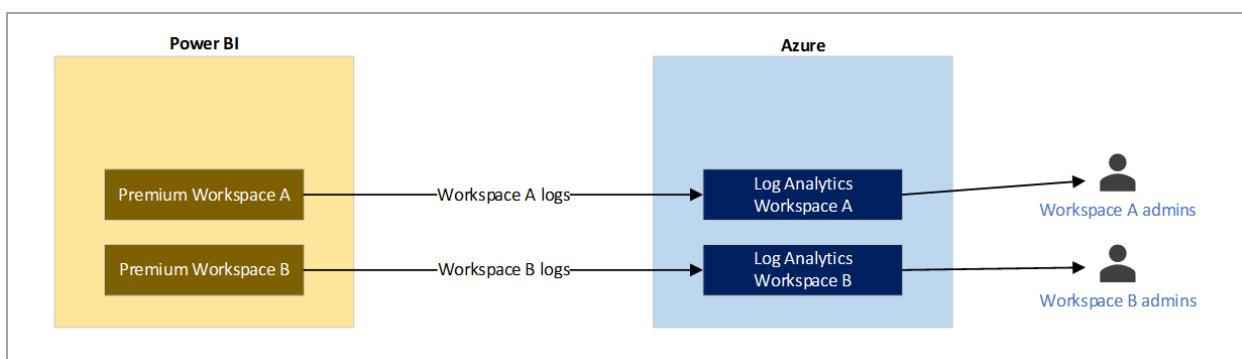
Example 1: Workspace logs for one workspace only

In this example, only workspace logs from *Workspace A* are sent to a dedicated Log Analytics workspace:



Example 2: Workspace logs sent to dedicated Log Analytics workspaces

In this example, workspace logs from two different Power BI workspaces are each sent to separate, dedicated Log Analytics workspaces:



These examples highlight the various ways you can use Azure Log Analytics with Power BI, and get the log information you need.

In another article, you can see how to configure Azure Log Analytics to work with Power BI, with specific steps and requirements to get your logging working properly.

Considerations and limitations

Keep the following considerations and limitations in mind when working with Azure Log Analytics and Power BI:

- [Sovereign cloud](#) support is currently limited to US Department of Defense and US Government Community

Cloud High.

- Only Premium workspaces are supported.
- Only Workspace v2 support Log Analytics connections.
- Activities are only captured for datasets physically hosted within the Premium workspace where you configure logging. For example, if you configure logging for Premium workspace A, you will not see logs for any reports within that use datasets hosted in [Azure Analysis Services](#). You will also not see any logs for [shared datasets](#) that are not in Premium workspace A. To capture activities for shared datasets, configure logging on the workspace that contains the shared dataset, not the workspace that contains the report.
- You cannot connect multiple Power BI workspaces to the same Log Analytics workspace at this time.
- Datasets created on the web by uploading a CSV file do not generate logs.
- If you have Multi-Factor Auth (MFA) in place for Azure but not Power BI, the configuration screens will give general Azure errors. A workaround is to first log in to the [Azure portal](#), complete the MFA challenge and then log into Power BI in the same browser session.
- The following events are intentionally excluded temporarily. This will affect the Query Detail page in the template report (that is, no storage engine subqueries will be visible for now)
 - ProgressReportCurrent
 - ProgressReportBegin
 - ProgressReportError
 - VertipaqSEQueryBegin
 - VertipaqSEQueryEnd

Next steps

The following articles provide more information about Power BI and its many features:

- [Configuring Azure Log Analytics for Power BI \(Preview\)](#)
- [Azure Log Analytics in Power BI FAQ](#)
- [What is Power BI Premium?](#)
- [Organize work in the new workspaces in Power BI](#)
- [Creating a dataflow](#)
- [Dataflows best practices](#)

Configuring Azure Log Analytics for Power BI (Preview)

3/30/2022 • 8 minutes to read • [Edit Online](#)

Power BI is integrating with Azure Log Analytics (LA) to enable administrators and Premium workspace owners to configure a Log Analytics connection to their Power BI subscription. This article describes how the integration between Log Analytics and Power BI works, and how to configure it for your environment.

There are two elements to getting Azure Log Analytics working for Power BI: configuring your Azure subscription in the Azure portal, and enabling Log analytics for Power BI in the Power BI Admin portal. The following sections take you through the steps in each, in turn.

The screenshot shows the 'Settings' page for a user named 'admin'. The 'Azure connections (preview)' tab is selected. Under 'Log Analytics', there is a section for connecting to an Azure Log Analytics workspace. It shows a 'Subscription' (a1), a 'Resource group' (rg-byola), and a 'Log Analytics workspace' (admin). The configuration was made by 'Admin Admin' on 2020-10-12T12:00:00Z. A 'Disconnect from Azure' button is visible at the bottom.

Pre-requisites

Before you can configure Log Analytics integration from Power BI, you need to [create a Log Analytics Workspace](#) in the Azure portal. You must also give permission in Azure for the Power BI service to write logs. The exact requirements are:

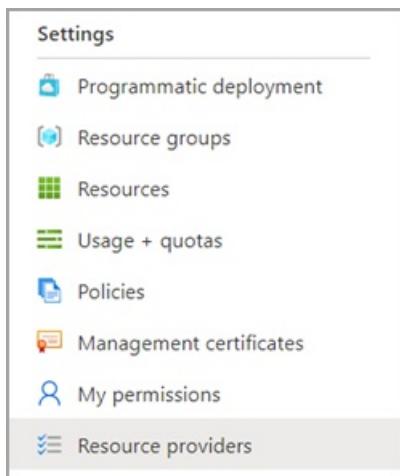
- Register the 'microsoft.insights' resource provider in the Azure subscription where you will collect Power BI log data
- The user who will set up Log Analytics integration in Power BI must be in the Owner role for the Log Analytics Workspace. See FAQ for workarounds if Owner cannot be given.
- The service principal 'Power BI Service' must be in the Owner role for the Log Analytics Workspace.

The following section shows you how to meet these three requirements.

Enable the 'microsoft.insights' Resource Provider

Log Analytics requires the 'microsoft.insights' resource provider enabled at the Azure subscription level. The following steps take you through the process.

1. Log into the Azure portal, select the subscription you want to use with Log Analytics and that contains your Log Analytics workspaces. In the **Settings** section, select **Resource providers** as shown in the following image.



2. Search for **microsoft.insights** under **Resource providers** and then select **Register**.

A screenshot of the Azure Resource providers page. The top navigation bar shows 'Home > Subscriptions > [Subscription] | Resource providers'. On the left, there's a sidebar with links: 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', and 'Security'. The main area has a search bar with 'Search (Ctrl+ /)' and buttons for 'Re-register', 'Unregister', and 'Refresh'. A search term 'insight' is entered in the search bar. A table lists resource providers: 'Microsoft.OperationalInsights' with status 'Registered' and 'microsoft.insights' with status 'Registered'. The 'microsoft.insights' row is highlighted with a red border. The table has columns 'Provider' and 'Status'.

Set Permissions

1. Make sure the user configuring Log Analytics integration and the Power BI Service principal are in the **Owner** role of the Log Analytics workspace. When you select **Access control (IAM)** for the subscription in the Azure portal, and then select **Role assignments** from the top selections in the panel, the current user must see at least two entries: **Owner** for the user who will configure Log Analytics (1, in the following image), and **Owner** for the Power BI service (2, in the following image).

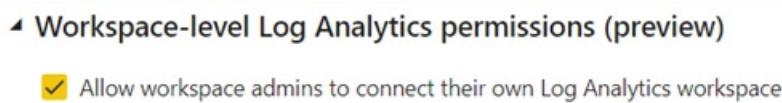
The screenshot shows the 'Access control (IAM)' section of the Azure Log Analytics workspace. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Locks, Agents management, Network Isolation, Advanced settings), General, and Workspace summary. The main area displays 'Role assignments' with 3 assignments for this subscription, up to a limit of 2000. It includes search, type, and scope filters. Two entries are shown: 'Admin Admin' (User, Owner) and 'Power BI Service' (App, Owner). Both entries are highlighted with red boxes.

Once you've completed those steps, the Azure Log Analytics configuration portion is complete. The next section shows you how to continue and complete the configuration in the Power BI Admin portal.

Allow Workspace-level Logging from the Admin Portal

A Power BI administrator must complete the following step to enable Azure Log Analytics for Power BI Premium workspaces. This will allow Power BI Premium workspace administrators to send their workspace logs to Azure Log Analytics when the pre-requisites have been met.

1. In the **Power BI Admin portal** navigate to **Tenant Settings > Azure connections** and expand the **Workspace-level Log Analytics permissions (preview)**. To allow workspace admins to enable Log Analytics, select the checkbox as shown in the following image.



Configure Logging in a Premium Workspace

1. In the **Premium workspace**, workspace admins can enable Log Analytics. To do so, navigate to **Settings** as shown in the following image.

The screenshot shows the Power BI service dashboard. At the top, there is a user profile for 'admin' and a 'Create app' button. Below the header, there are navigation links: '+ New', 'Create a pipeline', 'View', 'Filters', 'Settings' (which is highlighted with a red box), 'Access', and 'Search'. Under the 'Content' section, there are three items listed: 'Customer360 TEST' (Report), 'Customer360 TEST' (Dataset), and 'Customer360 TEST.pbix' (Dashboard). Each item has columns for Name, Type, Owner, Refreshed, and Next refresh.

2. In the **Settings** pane, select **Azure connections (preview)**, then expand **Log Analytics** as shown in the following image.

The screenshot shows the 'Settings' pane with the 'Azure connections (preview)' tab selected. Under the 'Log Analytics' section, it says: 'Connect an Azure Log Analytics workspace to collect usage and performance logs for this workspace.' A 'Learn more' link and a 'Connect to Azure' button are present. Below this, there are dropdown menus for 'Subscription', 'Resource group', and 'Log Analytics workspace', each with a 'Select an Option' placeholder. At the bottom are 'Save' and 'Cancel' buttons.

3. Select the Azure subscription, Resource group, and then the Log Analytics workspace configured in the previous section, then select **Save**. When successfully completed, the expanded **Tenant-level Log Analytics (Preview)** section will look similar to the following image.

 **Settings**

admin [REDACTED]

About Premium Azure connections (preview)

▶ Storage

◀ Log Analytics

Connect an Azure Log Analytics workspace to collect usage and performance logs for this workspace. [Learn more](#)

Subscription a1 [REDACTED] 7e

Resource group rg-byola

Log Analytics workspace admir [REDACTED]

Configured by Admin Admin on 2020-10-12T12:00:00Z

Disconnect from Azure

Disconnecting will stop usage and performance data from flowing into your Azure Log Analytics workspace.

Disconnect from Azure

Disconnect Azure Log Analytics

You can disconnect from Azure Log Analytics to stop sending logs to Azure. To disconnect, in the **Power BI Workspace Settings**, navigate to the **Log Analytics** settings. The following image shows the warning displayed when the **Disconnect from Azure** button is selected.

◀ Tenant-level Log Analytics (preview)

Connect an Azure Log Analytics workspace to collect usage and performance logs for your entire organization. [Learn more](#)

Subscription a1 [REDACTED] 7e

Resource group rg-byola

Log Analytics workspace ms [REDACTED]

Configured by Admin Admin on 2020-10-09T12:00:00Z

Disconnect from Azure

ⓘ Disconnecting will stop usage and performance data from flowing into your Azure Log Analytics workspace.

Save **Cancel**

Select **Save** to disconnect.

NOTE

When you disconnect a Power BI workspace from Azure Log Analytics, logs are not deleted. Your data remains in the Azure and follows the storage and retention policies you set there.

Usage scenarios

There are many ways that Azure Log Analytics and Power BI can help solve real-world challenges for your organization. Here are a few to consider:

- Identifying periods of high or unusual Analysis Services engine activity by capacity, workspace, report or user
- Analyzing query performance and trends, including external DirectQuery operations
- Analyzing dataset refresh duration, overlaps and processing steps
- Analyzing custom operations sent using the Premium XMLA endpoint

Send us feedback in the Power BI Community for how you're using logging, and how it has helped your organization.

Error conditions and resolutions

The following table provides a collection of common errors, the events or configurations that triggered them, and suggested resolutions.

Trigger Condition	Type	Message
User doesn't have permission to write to the Log Analytics Workspace	Error - cannot proceed	You need write permissions on this Log Analytics workspace to connect it to Power BI. Contact the person in your organization who manages Azure subscriptions to fix this.
User doesn't have permission to write to the Log Analytics workspace account	Error - cannot proceed	You need write permissions on this Log Analytics workspace to connect it to Power BI.
Does not have access to any Azure subscriptions	Error - cannot proceed	You don't have access to any Azure subscriptions. Ask the person who manages Azure subscriptions in your organization to grant you contributor access or higher.
Does not have access to any Azure Log Analytics workspaces within that subscription	Error - cannot proceed	You don't have access to an Azure Log Analytics workspace. Ask the person who manages Azure subscriptions in your organization to add you to the Log Analytics owner or contributor role.
Workspace-level Log Analytics disabled when trying to connect	Information	Ask your tenant admin to grant workspace admins permission to connect Log Analytics workspaces.

Trigger Condition	Type	Message
Workspace-level Log Analytics disabled when trying to disconnect	Information	Your tenant admin revoked permission for workspace admins to connect their own Azure Log Analytics workspaces. If you disconnect, you won't be able to connect to another one.

Events and schema

Once enabled, Azure Log Analytics will log the following **event categories**. For more information on these events, see [Analysis Services trace events](#).

- AggregateTableRewriteQuery
- Command
- Deadlock
- DirectQuery
- Discover
- Error
- ProgressReport
- Query
- Session Initialize
- VertiPaqSEQuery

NOTE

The following events are currently not available in the Preview:

- ProgressReportBegin
- ProgressReportCurrent
- ProgressReportError
- VertipaqSEQueryBegin
- VertipaqSEQueryEnd

The following table describes the **schema**.

Property	Existing Azure Analysis Services Property	Description
ApplicationContext	ApplicationContext_s	Property bag of unique identifiers providing details about the application executing the request. for example, report ID.
ApplicationName	ApplicationName_s	Contains the name of the client application that created the connection to the server. This column is populated with the values passed by the application rather than the displayed name of the program.
ArtifactId		Unique identifier of the resource logging the data.

PROPERTY	EXISTING AZURE ANALYSIS SERVICES PROPERTY	DESCRIPTION
ArtifactKind		Type of artifact logging the operation, for example, Dataset
ArtifactName	DatabaseName_s	The name of the Power BI artifact logging this operation.
LogAnalyticsCategory	Unique	Category of the events, like Audit/Security/Request.
CorrelationId		The ID for correlated events. Can be used to identify correlated events between multiple tables.
CpuTimeMs	CPUTime_s	Amount of CPU time (in milliseconds) used by the event.
CustomerTenantId		Customer's Power BI tenant identifier.
DatasetMode		The mode of the dataset. Import, DirectQuery, or Composite.
DurationMs	Duration_s	Amount of time (in milliseconds) taken by the operation.
EventText	TextData_s	Contains verbose information associated with the operation, for example, DAX Query
ExecutingUser	EffectiveUsername_s	The user executing the operation.
Identity		Information about user and claims.
Level	Severity_s	Contains the severity level of the operation being logged. Success, Informational, Warning, or Error.
OperationDetailName	EventSubclass_s	More details about the operation.
OperationName	EventClass_s	The operation associated with the log record.
PremiumCapacityId		Unique identifier of the Premium capacity hosting the artifact being operated on.
ProgressCounter	ProgressTotal_s	Progress counter.
Status		Status of the operation.
StatusCode	Error_s	Status code of the operation. It covers success and failure.

PROPERTY	EXISTING AZURE ANALYSIS SERVICES PROPERTY	DESCRIPTION
TenantId		Unique identifier of Microsoft's Power BI tenant. This does not refer to the customer tenant.
TimeGenerated		The timestamp (UTC) of when the log was generated.
User	User_s	The user on whose behalf the operation is running. Used when an end-user identity must be impersonated on the server.
WorkspaceId		Unique identifier of the workspace containing the artifact being operated on.
WorkspaceName	ServerName_s	Name of the workspace containing the artifact.
XmlaObjectPath	ObjectPath_s	Object path. A comma-separated list of parents, starting with the object's parent.
XmlaProperties	RequestProperties_s	Properties of the XMLA request.
XmlaRequestId	RootActivityId_g	Unique Identifier of request.
XmlaSessionId	SPID_s	Analysis Services session identifier.

Sample Log Analytics KQL queries

The following collection of sample queries may be helpful when using Azure Log Analytics with Power BI. They can be executed directly in the Azure portal or through APIs to query the latest data, typically about 5-10 minutes old.

```

// log count per day for last 30d
PowerBIDatasetsWorkspace
| where TimeGenerated > ago(30d)
| summarize count() by format_datetime(TimeGenerated, 'yyyy-MM-dd')

// average query duration by day for last 30d
PowerBIDatasetsWorkspace
| where TimeGenerated > ago(30d)
| where OperationName == 'QueryEnd'
| summarize avg(DurationMs) by format_datetime(TimeGenerated, 'yyyy-MM-dd')

//query duration percentiles for a single day in 1 hour bins
PowerBIDatasetsWorkspace
| where TimeGenerated >= todatetime('2021-04-28') and TimeGenerated <= todatetime('2021-04-29')
| where OperationName == 'QueryEnd'
| summarize percentiles(DurationMs, 0.5, 0.9) by bin(TimeGenerated, 1h)

// refresh durations by workspace and dataset for last 7d
PowerBIDatasetsWorkspace
| where TimeGenerated > ago(30d)
| where OperationName == 'CommandEnd'
| where ExecutingUser contains 'system'
| where EventText contains 'refresh'
| project WorkspaceName, DatasetName = ArtifactName, DurationMs

```

Next steps

The following articles can help you learn more about Power BI, and about its integration with Azure Log Analytics.

- [Using Azure Log Analytics in Power BI \(Preview\)](#)
- [Azure Log Analytics in Power BI FAQ](#)
- [What is Power BI Premium?](#)
- [The new workspace experience in Power BI](#)

Azure Log Analytics in Power BI - FAQ (Preview)

3/30/2022 • 3 minutes to read • [Edit Online](#)

Power BI is integrating with Azure Log Analytics (LA) to enable administrators and Premium workspace owners to configure a Log Analytics connection to their Power BI subscription.

This article is a collection of Frequently Asked Questions (FAQ) for reference during the preview period. When **Azure Log Analytics for Power BI** becomes generally available, this FAQ will be incorporated into other articles about the feature.

Frequently asked questions

Question: What areas of Power BI are available for Log Analytics integration?

Answer: Dataset activity logs (such as Analysis Services Engine traces) are currently available in the Preview.

Question: When should I use Log Analytics for the Analysis Services Engine?

Answer: Engine logs are detailed and can be high volume and large, averaging 3-4 KB each for complex datasets. Therefore we recommend carefully considering when to use logging for the Analysis Service engine.

Typical use cases for logging are performance investigations, scale/load testing or pre-release validation.

Question: Which Analysis Services events are supported? What will the logs look like?

Answer: Refer to [events and schema](#) for details.

Question: I cannot get Owner permissions for Azure Log Analytics in my organization, is there a workaround?

Answer: Yes, you will need some help from administrators:

OPTION 1:

An Azure admin can grant you Owner rights in Log Analytics only to perform the initial configuration in Power BI. After this, they can reduce your access to Contributor or lower as required.

OPTION 2:

For workspace level configuration, you can add an Azure admin as a Power BI Workspace admin and ask them to configure logging for your workspace. Once logging is configured, you can remove their access to your workspace.

Question: I cannot get Workspace Admin permissions for Power BI in my organization, is there a workaround?

Answer: Yes. Refer to option 2 in the previous question.

Question: The schema only contains some GUID identifiers, can you include the item names?

Answer: We hope to expose Report Name, Workspace Name and others as the feature progresses towards General Availability.

Question: What happens if I send logs from many Power BI workspaces to the same Log Analytics Workspace? How do I differentiate?

Answer: This configuration is currently not supported.

Question: Can we configure Log Analytics for non-Premium workspaces?

Answer: No, currently only Premium workspaces are supported.

Question: How long does it take for logs to appear in Log Analytics?

Answer: Typically within 5 minutes of the activity being generated in Power BI. The logs are sent continuously.

Question: What happens when I disconnect Log Analytics? Will I lose my data?

Answer: Disconnecting is a non-destructive operation. Logs will stop flowing to Log Analytics, but everything else remains unchanged. Power BI will not alter permissions or delete data.

Question: How much data is retained in Log Analytics?

Answer: The default retention period is 31 days. You can adjust data retention period within the Azure portal, which currently can be increased to 730 days (two years).

Question: What if the tenant administrator disables workspace-level logging?

Answer: No new Log Analytics configurations can be made at workspace-level if that occurs. Any existing workspaces that have Log Analytics already configured will continue to send logs.

Question: Do you support Blob Store and Event Hub destinations in Log Analytics?

Answer: These are not currently supported, but your feedback is welcomed on how useful you would find those destinations.

Question: What happens if I move my workspace out of a Premium capacity?

Answer: Currently the Log Analytics configuration will not be deleted, but logs will stop flowing when the dataset is not in a Premium capacity. If you move it back to Premium capacity, logs will begin to flow again.

Question: Do you support Workspace v1 for Log Analytics?

Answer: There is no ability to configure Log Analytics for individual v1 workspaces.

Question: There are numerous events logged from the AS Engine. Can I choose which ones I want?

Answer: Currently you cannot choose which events to log.

Question: How much will Log Analytics cost?

Answer: Azure Log Analytics bills storage, ingestion, and analytical queries independently. Cost also depends on the geographic region and will vary depending on how much activity is generated, how long you choose to store the data and how often you query it. An average Premium capacity generates about 35 GB of logs monthly, but this can be higher for heavily utilized capacities. Refer to the cost calculator for more information.

Next steps

The following articles can help you learn more about Power BI, and about its integration with Azure Log Analytics.

- [Using Azure Log Analytics in Power BI \(Preview\)](#)
- [Configuring Azure Log Analytics for Power BI \(Preview\)](#)

Installing the Log Analytics for Power BI Datasets Template App (preview)

3/30/2022 • 2 minutes to read • [Edit Online](#)

The Power BI Log Analytics for Analysis Services Engine app is designed to analyze AS Engine behavior in general, and to help isolate and debug specific problems in depth. Use this guide to install the app. Once the app is installed, you can [learn how to use it](#).

NOTE

The app is updated regularly with new features and functionalities. If you see there's a pending update in the notifications center, we recommend that you update the app.

Prerequisites

Before you install the AS Engine app, review these requirements:

- You need to have a Log Analytics Workspace

Install the app

To install the AS Engine app, follow these steps:

1. Go to [AppSource > Power BI Log Analytics for Analysis Services Engine](#) and select **Get it now**.
2. If prompted, sign in to AppSource using your work or school account and complete the registration screen. The app will take you to the Power BI Service to complete the process. Select **Install** to continue.
3. Open the app and when you see the message *You have to connect to your own data to view this report*, select **Connect**.



You have to connect to your own data to view this report.

Connect

4. In the first window, fill in the following fields:

- **Days Ago to Start** - How many days ago to start loading log data. Maximum value you can

select here is 30 days.

- **Days Ago to Finish** - How many days ago to stop loading log data. Use 0 for today.
- **Log Analytics Table** - * PowerBIDatasetsWorkspace
 - * PowerBIDatasetsTenant (Currently not supported.)
- **Log Analytics WorkspaceId** - GUID of the Azure Log Analytics Workspace containing the AS Engine data.
- **UTC Offset Hours** - An hourly offset used to convert the data from UTC to a local time zone
- **Pagination Hours** - This is an optional parameter. It describes the time window for each log analytics call from Power BI. You only need to update this if you're running into failures while fetching data due to data size exceeding Log Analytics limits.

5. Select **Next**.

6. In the second window, fill in the following fields:

* **Authentication method** - Select your authentication method. The default authentication method is *OAuth2*.

* **Privacy level setting for this data source** - Select *Organizational* to enable app access to all the data sources in your organization.

7. Select **Sign in and continue**.

Next steps

- [Using the Log Analytics for Power BI Datasets Template App \(preview\)](#)

Using the Log Analytics for Power BI Datasets Template App (preview)

3/30/2022 • 16 minutes to read • [Edit Online](#)

Power BI is integrating with Azure Log Analytics (LA) to enable administrators and Premium workspace owners to configure a connection from Power BI to a Log Analytics workspace in Azure subscriptions that they control. We published a template app to give you a head start with your analysis. This article describes the app so you can understand how to use the parameters, filters, navigation, and drillthrough paths to answer questions about dataset operations from the perspective of the Analysis Services (AS) engine. We will describe each page so you can understand its purpose and the typical use cases they support.

To [install the AS Engine app](#), you must have a Log Analytics workspace. Once installed, anyone in the organization with the right permissions can view the app.

App Goals

We wanted to build an app that can be used to analyze AS engine behavior in general, and to help isolate and debug specific problems in depth. Any operation can be sliced by CapacityId, Workspace Name, Dataset Name, and ReportId to give you the necessary context. We are looking into providing you with more item names, and not just the ID.

- ▶ Some examples of questions that can be answered

App Data Source

The app loads data from a single Azure Log Analytics workspace.

It doesn't matter if the Log Analytics workspace contains data from many Power BI workspaces. It also doesn't matter which level of administrator configured logging. The log schemas are exactly the same for every role, so there is only one version of the app. We included different levels of aggregation to accommodate a range of use cases. [Using Log Analytics in Power BI](#) goes into detail on this.

App Data Model

The app has the following tables and relationships:

- User
- Query Duration Segment
- Scenario
- Calendar
- Time
- Operation
- Suboperation - Aggregations
- Suboperation - Query
- Suboperation - Refresh

- ▶ Screenshot of ER Diagram

App Parameters

The following parameters are defined in the template:

PARAMETER	DESCRIPTION
Days Ago To Start	Load data from the specified number of days ago. Maximum value you can select here is 30 days.
Days Ago To Finish	Load data up to the specified number of days ago. Use 0 for today.
Log Analytics Table	Preset values corresponding to the Log Analytics source table: - PowerBIDatasetsWorkspace - PowerBIDatasetsTenant Currently only PowerBIDatasetsWorkspace is supported.
Log Analytics WorkspaceId	GUID of the Azure Log Analytics workspace containing the AS Engine data.
UTC Offset	An hourly offset used to convert the data from UTC to a local time zone.
Pagination Hours	This is an optional parameter. It describes the time window for each log analytics call from Power BI. You only need to update this if you're running into failures while fetching data due to data size exceeding Log Analytics limits.

- ▶ Screenshot of AS Engine Parameters

App Usage

App Workflow

- ▶ Diagram showing the major pages of the app and some important available drillthroughs

Workspace Summary

Shows engine activities and load at a workspace level, focusing on identifying interesting datasets, reports, or users. You can use this to identify a high-level issue to analyze further by navigating or drilling through to other pages of the app.

Engine Activities

Provides engine load and activity trends by day and hour, with the ability to select a scenario such as Refresh Completed or Query Completed. You can drill through to the Engine Activity Detail page to get a look at a detailed list of each activity within the selected context.

Engine Activity Detail

This is a drillthrough page showing event-level data. For example, you can list all queries that ran in a particular time range.

Dataset Refresh

Provides a Gantt chart for style view of refreshes to observe duration and parallelism. You can drill through to the dataset refresh details for more details.

Dataset Refresh Detail

A drillthrough page showing the operations that occurred within a single dataset refresh. You can use this view to identify the longest running operation in a refresh and to see if there are any dependencies between activities.

Query Statistics

An overview of query performance, highlighting typical and low performing durations and letting you see how variable each unique query is.

Query Detail

A drillthrough page showing visuals such as a detailed table for the query, a table for related queries etc. For Import tables, the page will show you the internal Vertipaq storage engine queries and durations. For DirectQuery models, the page will show you the external queries, for example T-SQL sent to a SQL Server.

Query History

Shows you every execution of a query, provides CPU and duration stats and trend visuals to see if there are any spikes.

User Activities

A summary view that focuses on users, helping you identify the most active users and those users who are experiencing worse performance relative to others.

User Detail

A drillthrough page that provides details of the activities performed by a single user.

Error Summary

Helps identify errors and spot any error trends.

Error Details

Allows you to zoom in on a specific error by viewing the detailed event.

Navigating in The App

- ▶ The app contains a navigation bar at top of the page to navigate to the expected page.
- ▶ Also, there is a back button on top-left corner to go back to the previous page and an info icon that provides information about the page.

Filtering and Understanding Current Context

- ▶ Every page has a filter button below the navigation bar that you can click to open the pop-up filter panel and make selections.
- ▶ The current values of the filters are displayed in the smart narrative next to the filters button. You can clear all the slicers using the **Clear** button on the top-left corner or close the window using the **X** button on top-right corner.

App Pages

- [Workspace Summary](#)
- [Engine Activities](#)
- [Engine Activity Details](#)
- [Dataset Refreshes](#)
- [Dataset Refresh Detail](#)
- [Query Statistics](#)
- [Query Detail](#)
- [Query History](#)
- [User Activities](#)
- [User Detail](#)
- [Error Summary](#)
- [Error Detail](#)

- [Help](#)

Page: Workspace Summary

This page is targeted at a Workspace Administrator and shows activities and statistics related to datasets and queries. It also identifies top reports by load, details popular datasets by operations or users, and allows drillthrough to various pages to get further details.

- ▶ [Workspace Summary](#)

Page: Engine Activities (also a drillthrough)

This page provides a trend overview of AS Engine activities by day and by hour. It allows you to identify peaks or outliers on a day and then see how that activity was distributed by hour when you cross-highlight by selecting a day.

- ▶ [Engine Activities](#)

Drillthrough Page: Engine Activity Details

This page allows you to focus on a narrow time range and see the individual activities at a granular level of detail. The example below shows all the DAX queries that were executed in a minute, sorted by longest duration.

- ▶ [Engine Activity Details](#)

Page: Dataset Refreshes (also a drillthrough)

This page provides an overview of dataset refreshes occurring over a selected period. It allows you to identify long running refreshes and visualize which ones are happening in parallel. This page allows you to select any data refresh and drill through to a page called Dataset Refresh Detail.

- ▶ [Dataset Refreshes](#)

Drillthrough Page: Dataset Refresh Detail

This page allows you to visualize a single dataset refresh in detail. You can see all the internal operations that the engine performed such as executing queries and compressing data. It allows you to determine the longest running operations, which are parallel, and which may have dependencies.

- ▶ [Dataset Refresh Detail](#)

Page: Query Statistics (also a drillthrough)

This page focuses on queries in bulk. The goal is to identify which queries are common, and which queries have high variability. We provide percentiles and deviations to give you an idea of both typical and more extreme measurements.

Any query can be drilled through to a page called *Query Details* to see details about its execution, such as Storage Engine and Formulae Engine time. You can also see the internal Vertipaq Queries or external DirectQuery text and duration depending on the model type.

You can also drill through to a page called *Query History* that will show you all the execution of that query over a period, and its performance trend.

- ▶ [Query Statistics](#)

Drillthrough Page: Query Detail

This page provides a detailed look at a single execution of a DAX query. Depending on whether the query was for an Import or DQ model, you will either see the internal Vertipaq Storage Engine queries or the external DQ source queries (for example, T-SQL for SQL Server). It also identifies which aggregations were used, if any.

- ▶ [Query Detail](#)

Drillthrough Page: Query History

This is a historical view of a single unique query. It shows metrics over time and introduces the Storage Engine and Formula Engine time. You can use it to determine how consistent a query is over time and identify if issues are isolated to particular users or time frames.

► [Query History](#)

Page: User Activities

This page gives an overview of the user activities across the workspace. It also gives information about the most active users for a period by capturing their CPU time usage, query usage, and operations performed.

► [User Activities](#)

Drillthrough Page: User Detail

This page provides a detailed historical view of activities for a single user.

► [User Detail](#)

Page: Error Summary

This page provides an overview of errors or failed executions over time, allowing you to view individual operations that reported an error status.

► [Error Summary](#)

The cards on the right display overall operations count, query failure count, refresh failure count, and user count.

Drillthrough Page: Error Detail

This page provides details of errors generated by the engine. It also provides the information about failed operations due to query failures.

► [Error Detail](#)

Help

This page provides a help summary of different features throughout the app. It also has support links that can be used for any support assistance.

► [Help](#)

Considerations and Limitations

- [Log Analytics Query Limits](#)

- Kusto has limits in terms of number of records returned and the overall size of the data based on the query. The app has been built to work around these limits by pulling data in sequential chunks. However, you might still exceed these limits, resulting in a refresh failure in this template app.

[Query Limits](#)

- If template app refresh is failing due to above data limits, you can configure the Pagination Hours parameter. Setting a lower value here will lower the amount of data retrieved from Log Analytics per call by increasing the number of calls.

- The following events are intentionally excluded from Log Analytics during the Preview. Due to this, storage engine sub-queries are not visible for now on the Query Detail page.

- ProgressReportCurrent
 - ProgressReportBegin
 - ProgressReportError
 - VertipaqSEQueryBegin
 - VertipaqSEQueryEnd

Next steps

The following articles provide more information about Power BI and its many features:

- [Log Analytics for Analysis Services Template App](#)
- [Install Log Analytics Template App](#)
- [Configuring Azure Log Analytics for Power BI \(Preview\)](#)
- [Azure Log Analytics in Power BI FAQ](#)
- [What is Power BI Premium?](#)

Perform common query tasks in Power BI Desktop

3/30/2022 • 8 minutes to read • [Edit Online](#)

In the Power Query Editor window of Power BI Desktop, there are a handful of commonly used tasks. This article demonstrates those common tasks and provides links for additional information.

The common query tasks demonstrated here are:

- Connect to data
- Shape and combine data
- Group rows
- Pivot columns
- Create custom columns
- Query formulas

We'll use a few data connections to complete these tasks. The data is available for you to download or connect to, in case you want to step through these tasks yourself.

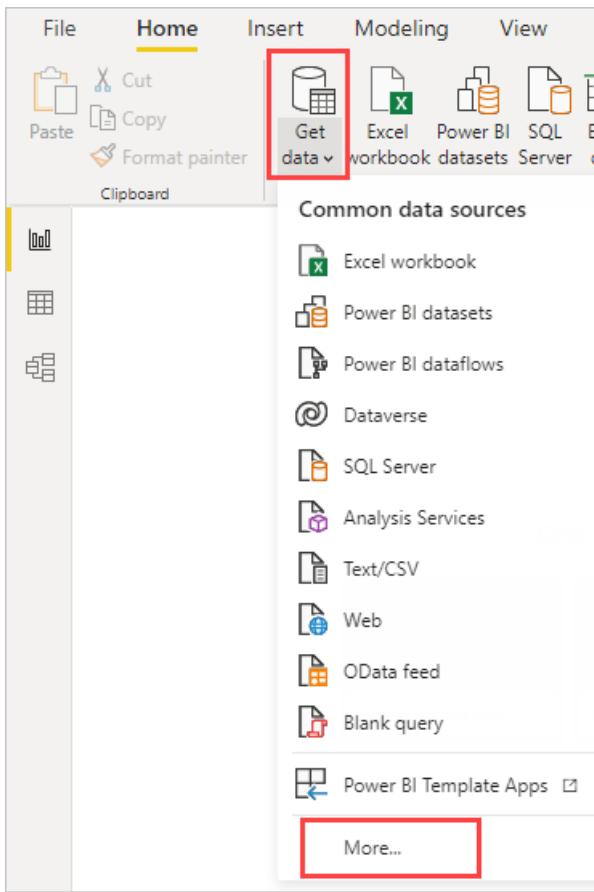
The first data connection is an [Excel workbook](#), which you can download and save locally. The other is a Web resource that's also used in other Power BI Desktop articles:

<https://www.bankrate.com/retirement/best-and-worst-states-for-retirement/>

Common query tasks begin at the steps necessary to connect to both of those data sources.

Connect to data

To connect to data in Power BI Desktop, select **Home** and then **Get data**. Power BI Desktop presents a menu with the most common data sources. For a complete list of data sources to which Power BI Desktop can connect, select **More** at the end of the menu. For more information, see [Data sources in Power BI Desktop](#).



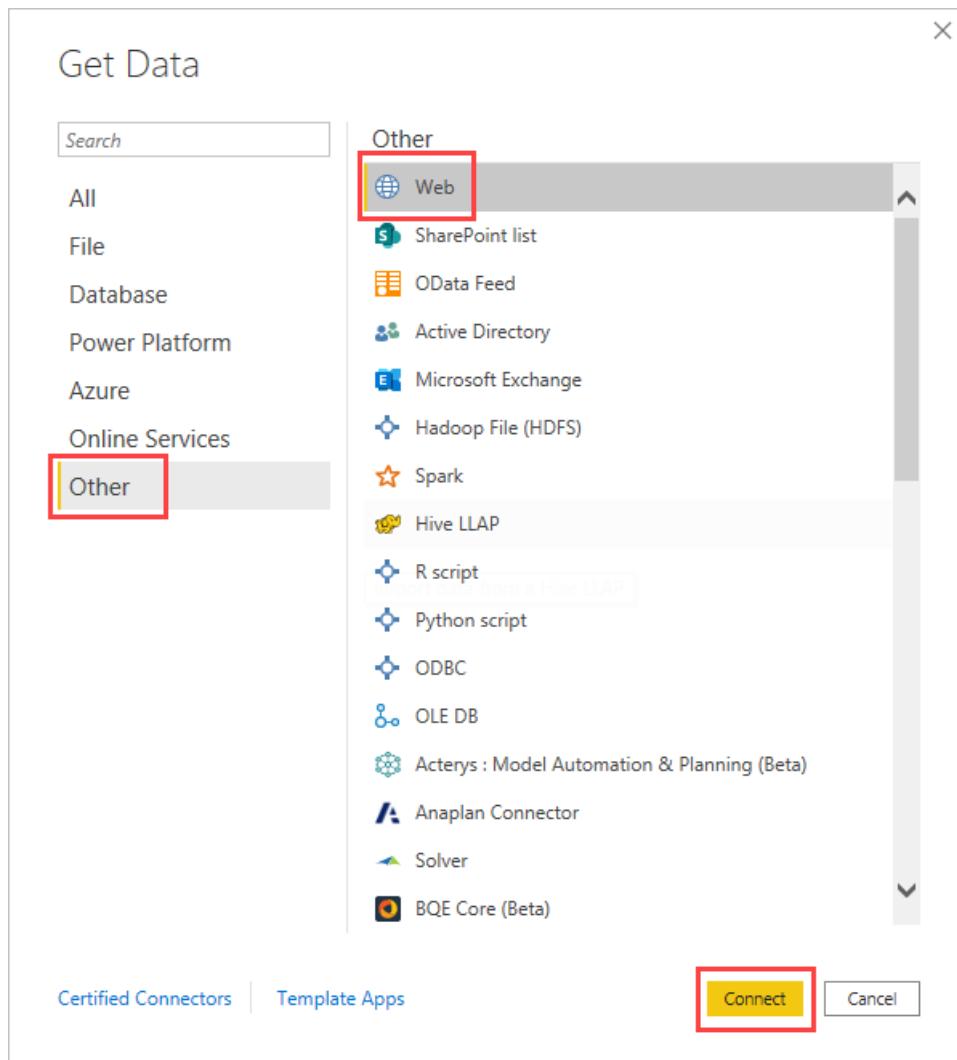
To start, select **Excel**, specify the Excel workbook mentioned earlier, and then select **Open**. Query inspects the workbook, then presents the data it found in the **Navigator** dialog box after you select a table.

Agency Name	State Name [District] Latest available year
"LIFELONG LEARNING RESEARCH INSTITUTE INC."	Arizona
100 LEGACY ACADEMY CHARTER SCHOOL	New Jersey
21ST CENTURY CHARTER SCH OF GARY	Indiana
21ST CENTURY CYBER CS	Pennsylvania
21ST CENTURY LEARNING ACADEMY	Ohio
21ST CENTURY PREPARATORY SCHOOL AGENCY	Wisconsin
21ST CENTURY SCHOOL	Ohio
4-WINDS ACADEMY INCORPORATED DBA 4-WINDS ACAD	Arizona
A CENTER FOR CREATIVE EDUCATION	Arizona
A CHILD'S VIEW SCHOOL INC.	Arizona
A E R O SPEC EDUC COOP	Illinois
A W BEATTIE CAREER CENTER	Pennsylvania
A W BROWN-FELLOWSHIP LEADERSHIP ACADEMY	Texas

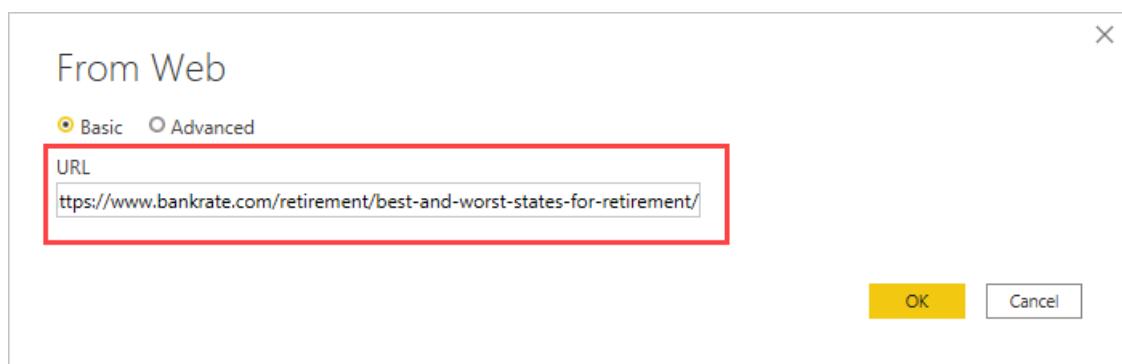
You can select **Transform Data** to edit, adjust, or *shape*, the data before you load it into Power BI Desktop.

Editing is especially useful when you work with large datasets that you want to pare down before loading.

Connecting to different types of data is as easy. You also want to connect to a Web resource. Choose **Get data > More**, and then select **Other > Web > Connect**.



The **From Web** dialog box appears, where you can type in the URL of the Web page.



Select **OK**. Like before, Power BI Desktop inspects the Web page data and shows preview options in the **Navigator** dialog box. When you select a table, it displays a preview of the data.

Other data connections are similar. If authentication is required to make a data connection, Power BI Desktop prompts you for the appropriate credentials.

For a step-by-step demonstration of connecting to data in Power BI Desktop, see [Connect to data in Power BI Desktop](#).

Shape and combine data

You can easily shape and combine data with Power Query Editor. This section includes a few examples of how you can shape data. For a more complete demonstration of shaping and combining data, see [Shape and combine Data with Power BI Desktop](#).

In the previous section, you connected to two sets of data: an Excel workbook and a Web resource. After the data is loaded in Power Query Editor, select the Web page query from the available queries in the **Queries** pane, as shown here:

The screenshot shows the Power Query Editor window with the following details:

- Home Tab:** Selected tab.
- Queries Pane:** Shows two queries: "Table1" and "Best States to Retire".
- Table View:** Displays a table with columns: "State", "Overall rank", and "Overall score". The table contains 20 rows of data.
- Query Settings Panel:** Shows the query name "Best States to Retire" and the applied steps:
 - Source
 - Extracted Table From Html
 - Promoted Headers
 - Changed Type** (highlighted)
- Ribbon:** Shows tabs like File, Home, Transform, Add Column, View, Tools, Help, and various buttons for managing data sources, columns, and rows.

When you shape data, you transform a data source into the form and format that meets your needs.

In Power Query Editor, many commands can be found in the ribbon, and in context menus. For example, when you right-click a column, the context menu lets you remove the column. You may also select a column and then select the **Manage Columns > Remove Columns** button from the **Home** tab in the ribbon.

The screenshot shows the Power Query Editor interface. In the center, there is a table with 20 rows and three columns. The first column is labeled '1', the second '2', and the third '3'. The data in the table consists of various numbers. On the right side of the table, a context menu is open for the third column, with the 'Remove' option highlighted by a red box. The menu also includes options like 'Copy', 'Remove Other Columns', 'Duplicate Column', 'Add Column From Examples...', 'Remove Duplicates', 'Remove Errors', 'Change Type', 'Transform', 'Replace Values...', 'Replace Errors...', 'Group By...', 'Fill', 'Unpivot Columns', 'Unpivot Other Columns', 'Unpivot Only Selected Columns', 'Rename...', and 'Move'.

You can shape the data in many other ways in this query. You may remove any number of rows from the top or bottom. Or you may add columns, split columns, replace values, and do other shaping tasks. With these features, you can direct Power Query Editor to get the data how you want it.

Group rows

In Power Query Editor, you can group the values from many rows into a single value. This feature can be useful when summarizing the number of products offered, the total sales, or the count of students.

In this example, you group rows in an education enrollment dataset. The data is from the Excel workbook. It's been shaped in Power Query Editor to get just the columns you need, rename the table, and make a few other transforms.

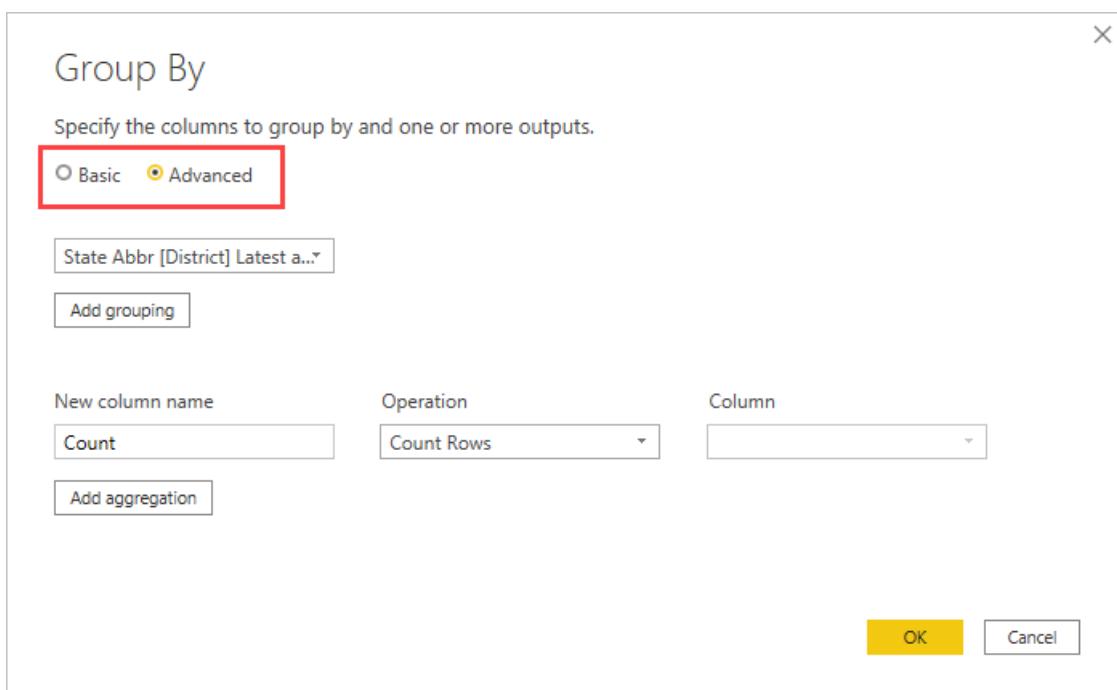
Let's find out how many Agencies each state has. (Agencies can include school districts, other education agencies such as regional service districts, and more.) Select the **Agency ID - NCES Assigned [District]** **Latest available year** column, then select the **Group By** button in the **Transform** tab or the **Home** tab of the ribbon. (**Group By** is available in both tabs.)

The screenshot shows the Power Query Editor interface. The ribbon has tabs like Tools and Help. In the top right, there are various icons for managing data: Refresh, Preview, Advanced Editor, Manage, Query, Choose Columns, Remove Columns, Keep Rows, Remove Rows, Sort, Split Column, Group By (which is highlighted with a red box), Transform, Data Type: Text, Use First Row as Head, and Replace Values. Below the ribbon is a table with three columns. Column A is labeled 'State Name [District]...', Column B is 'State Abbr [District] Latest available year', and Column C is 'Agency ID - NCES Assigned [District] La'. The 'State Abbr [District] Latest a...' column is currently selected.

The **Group By** dialog box appears. When Power Query Editor groups rows, it creates a new column into which it places the **Group By** results. You can adjust the **Group By** operation in the following ways:

1. The unlabeled dropdown list specifies the column to be grouped. Power Query Editor defaults this value to the selected column, but you can change it to be any column in the table.
2. **New column name:** Power Query Editor suggests a name for the new column, based on the operation it applies to the column being grouped. You can name the new column anything you want, though.
3. **Operation:** You may choose the operation that Power Query Editor applies, such as **Sum**, **Median**, or **Count Distinct Rows**. The default value is **Count Rows**.
4. **Add grouping and Add aggregation:** These buttons are available only if you select the **Advanced** option. In a single operation, you can make grouping operations (**Group By** actions) on many columns and create several aggregations using these buttons. Based on your selections in this dialog box, Power Query Editor creates a new column that operates on multiple columns.

Select **Add grouping** or **Add aggregation** to add more groupings or aggregations to a **Group By** operation. To remove a grouping or aggregation, select the ellipsis icon (...) to the right of the row, and then **Delete**. Go ahead and try the **Group By** operation using the default values to see what occurs.



When you select **OK**, Query does the **Group By** operation and returns the results. Whew, look at that – Ohio, Illinois, Texas, and California now each have over a thousand agencies!

	A B C State Abbr [District] Latest available year	1 2 3 Count
1	AZ	673
2	NJ	698
3	IN	394
4	PA	773
5	OH	1091
6	WI	461
7	IL	1078
8	TX	1277
9	IA	368
10	VA	225
11	MN	555
12	LA	126
13	SC	105
14	NY	952
15	MA	403
16	CA	1193
17	ID	151
18	MS	164

2 COLUMNS, 51 ROWS Column profiling based on top 1000 rows

And with Power Query Editor, you can always remove the last shaping operation. In the **Query Settings** pane, under **Applied Steps**, just select the X next to the step recently completed. So go ahead and experiment. If you don't like the results, redo the step until Power Query Editor shapes your data the way you want.

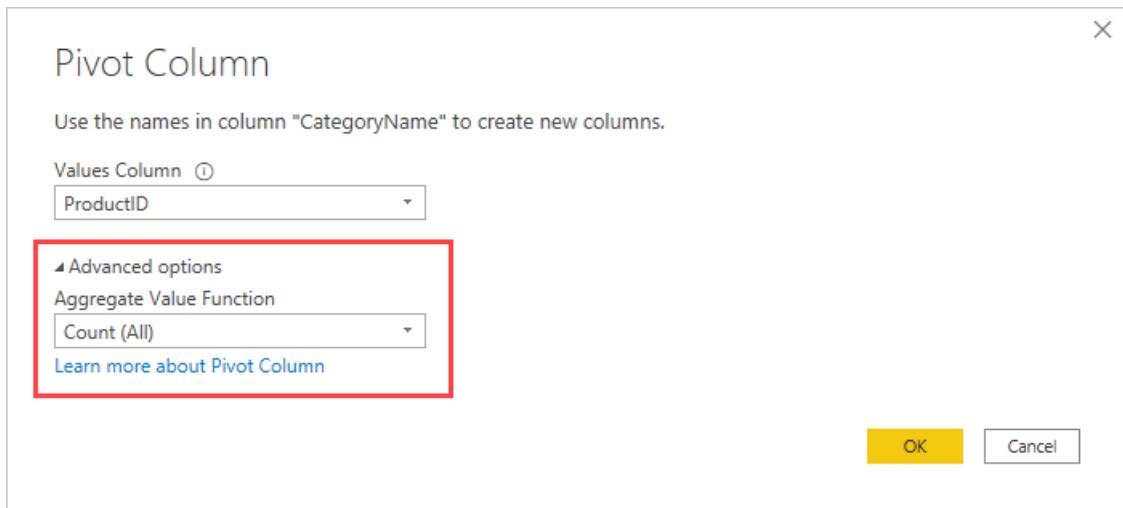
Pivot columns

You can pivot columns and create a table that contains aggregated values for each unique value in a column. For example, to find out how many different products are in each product category, you can quickly create a table to do that.

Let's look at an example. The following **Products_by_Categories** table has been shaped to only show each unique product (by name), and which category each product falls under. To create a new table that shows a count of products for each category (based on the **CategoryName** column), select the column, then select **Transform > Pivot Column**.

ABC 123 ProductName	ABC 123 ProductID	ABC 123 SupplierID
Alice Mutton		17
Aniseed Syrup		3
Boston Crab Meat		40
Camembert Pierrot		60
Carnarvon Tigers		18
Chai		1
Chang		2

The **Pivot Column** dialog box appears, letting you know which column's values will be used to create new columns. (If the wanted column name of **CategoryName** isn't shown, select it from the dropdown list.) When you expand **Advanced options**, you can select the function that will be applied to the aggregated values.



When you select **OK**, Query displays the table according to the transform instructions provided in the **Pivot Column** dialog box.

	Meat	1.2 Condiments	1.2 Seafood	1.2 Cheese	1.2 Beverages
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	0	1
5	0	0	1	0	0
6	0	0	0	0	0

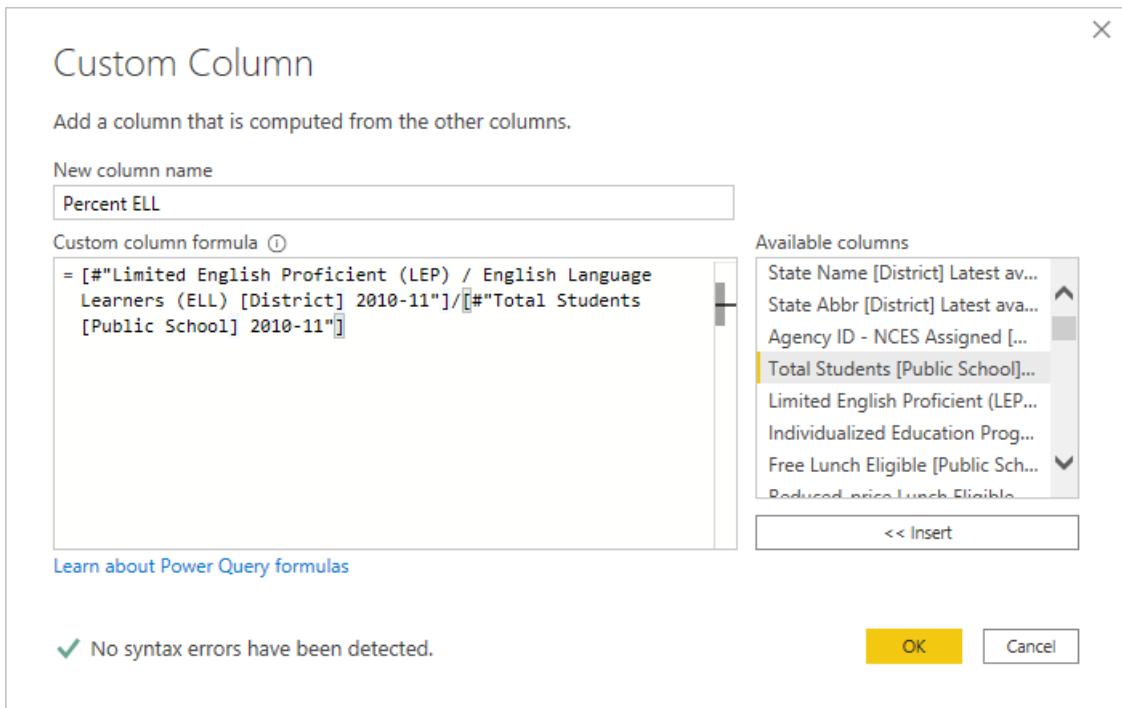
Create custom columns

In Power Query Editor, you can create custom formulas that operate on multiple columns in your table. Then you may place the results of such formulas into a new (custom) column. Power Query Editor makes it easy to create custom columns.

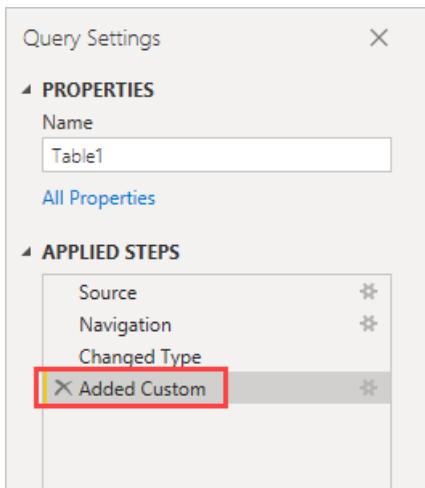
With the Excel workbook data in Power Query Editor, go to the **Add Column** tab on the ribbon, and then select **Custom Column**.

Agency Name	State Name [District] Latest available year	State Abbr [
"LIFELONG LEARNING RESEARCH INSTITUTE INC."	Arizona	AZ
100 LEGACY ACADEMY CHARTER SCHOOL	New Jersey	NJ
21ST CENTURY CHARTER SCH OF GARY	Indiana	IN
21ST CENTURY CYBER CS	Pennsylvania	PA
21ST CENTURY LEARNING ACADEMY	Ohio	OH

The following dialog box appears. In this example, create a custom column called *Percent ELL* that calculates the percentage of total students that are English Language Learners (ELL).



As with any other applied step in Power Query Editor, if the new custom column doesn't provide the data you're looking for, you can delete the step. In the **Query Settings** pane, under **Applied Steps**, just select the X next to the **Added Custom** step.



Query formulas

You can edit the steps that Power Query Editor generates. You can also create custom formulas, which let you connect to and shape your data more precisely. Whenever Power Query Editor does an action on data, the formula associated with the action is displayed in the formula bar. To view the formula bar, go to the **View** tab of the ribbon, and then select **Formula Bar**.

Column	Data
A ^B Agency Name	Arizona
1 "LIFELONG LEARNING RESEARCH INSTITUTE INC."	AZ
2 100 LEGACY ACADEMY CHARTER SCHOOL	New Jersey
3 21ST CENTURY CHARTER SCH OF GARY	Indiana

Power Query Editor keeps all applied steps for each query as text that you can view or modify. You can view or modify the text for any query using the **Advanced Editor**. Just select **View** and then **Advanced Editor**.

The screenshot shows the Power BI Desktop ribbon with the 'View' tab selected. In the 'Tools' group, the 'Advanced Editor' button is highlighted with a red box. Below the ribbon, there's a 'Queries [2]' section with 'Table1' selected. To the right is a data preview grid showing three rows of data:

	A ^B Agency Name	A ^B State Name [District] Latest available year	A ^B State Abbr [District] Latest av
1	"LIFELONG LEARNING RESEARCH INSTITUTE INC."	Arizona	AZ
2	100 LEGACY ACADEMY CHARTER SCHOOL	New Jersey	NJ
3	21ST CENTURY CHARTER SCH OF GARY	Indiana	IN

Here's a look at the **Advanced Editor**, with the query steps associated with the **USA_StudentEnrollment** query displayed. These steps are created in the Power Query Formula Language, often referred to as *M*. For more information, see [Learn about Power Query formulas](#). To view the language specification itself, see [Power Query M language specification](#).

Table1

Display Options

```

let
    Source = Excel.Workbook(File.Contents("C:\Users\v-tishe\OneDrive - Microsoft\Documents\AzureTechnicalContent\powerbi\PBI_Edu_ESLi_Enrolm"),
    Table1_Table = Source[[Item="Table1",Kind="Table"]][Data],
    #"Changed Type" = Table.TransformColumnTypes(Table1_Table,{{"Agency Name", type text}, {"State Name [District] Latest available year", ty
    #"Added Custom" = Table.AddColumn(#"Changed Type", "Percent ELL", each [#"Limited English Proficient (LEP) / English Language Learners (ELL)"]),
    #"Grouped Rows" = Table.Group(#"Added Custom", {"State Abbr [District] Latest available year"}, {"Count", each Table.RowCount(_), Int64}),
    in
    #"Grouped Rows"

```

No syntax errors have been detected.

Done Cancel

Power BI Desktop provides an extensive set of formula categories. For more information, and a complete reference of all Power Query Editor formulas, see [Power Query M function reference](#).

Next steps

You can do all sorts of things with Power BI Desktop. For more information on its capabilities, see the following resources:

- [What is Power BI Desktop?](#)
- [Query overview with Power BI Desktop](#)
- [Data sources in Power BI Desktop](#)
- [Connect to data in Power BI Desktop](#)
- [Shape and combine data with Power BI Desktop](#)

Create and manage relationships in Power BI Desktop

3/30/2022 • 24 minutes to read • [Edit Online](#)

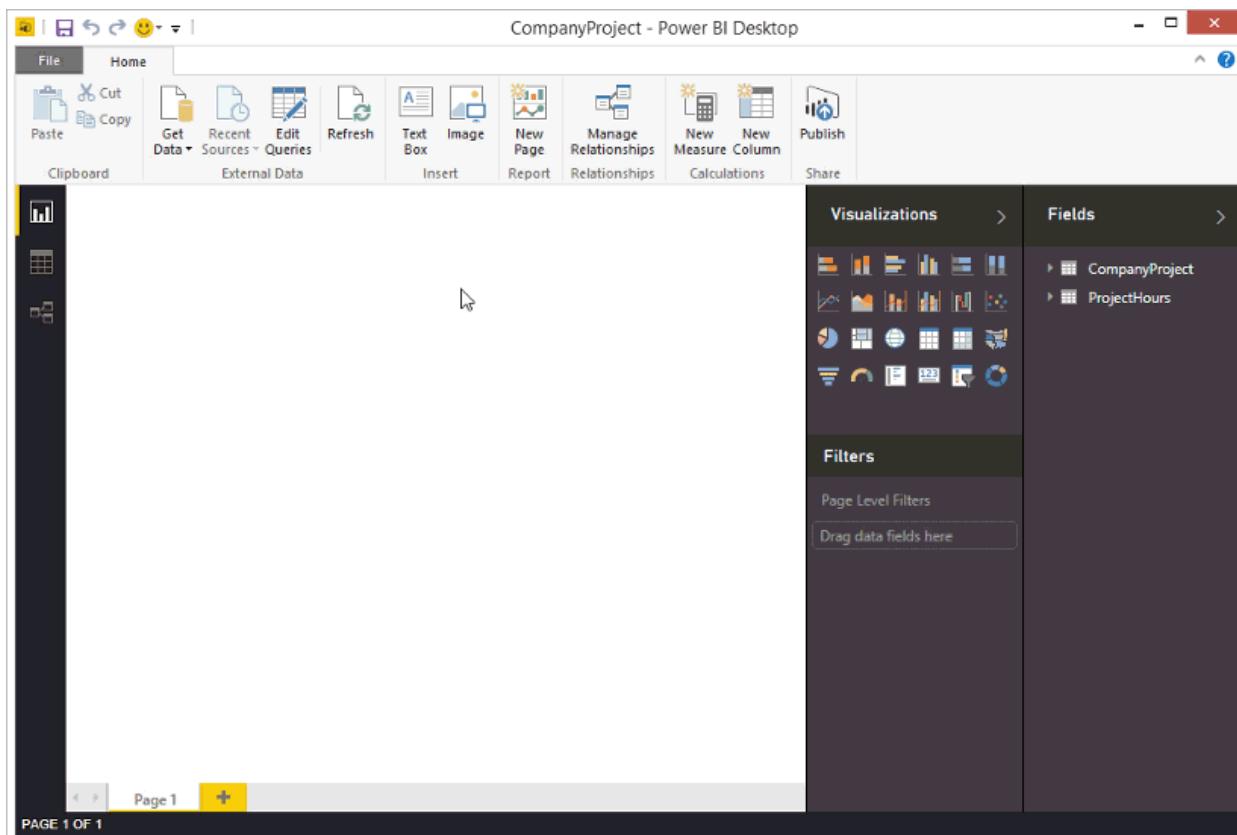
When you import multiple tables, chances are you'll do some analysis using data from all those tables. Relationships between those tables are necessary to accurately calculate results and display the correct information in your reports. Power BI Desktop makes creating those relationships easy. In fact, in most cases you won't have to do anything, the autodetect feature does it for you. However, sometimes you might have to create relationships yourself, or need to make changes to a relationship. Either way, it's important to understand relationships in Power BI Desktop and how to create and edit them.

Autodetect during load

If you query two or more tables at the same time, when the data is loaded, Power BI Desktop attempts to find and create relationships for you. The relationship options **Cardinality**, **Cross filter direction**, and **Make this relationship active** are automatically set. Power BI Desktop looks at column names in the tables you're querying to determine if there are any potential relationships. If there are, those relationships are created automatically. If Power BI Desktop can't determine with a high level of confidence there's a match, it doesn't create the relationship. However, you can still use the **Manage relationships** dialog box to manually create or edit relationships.

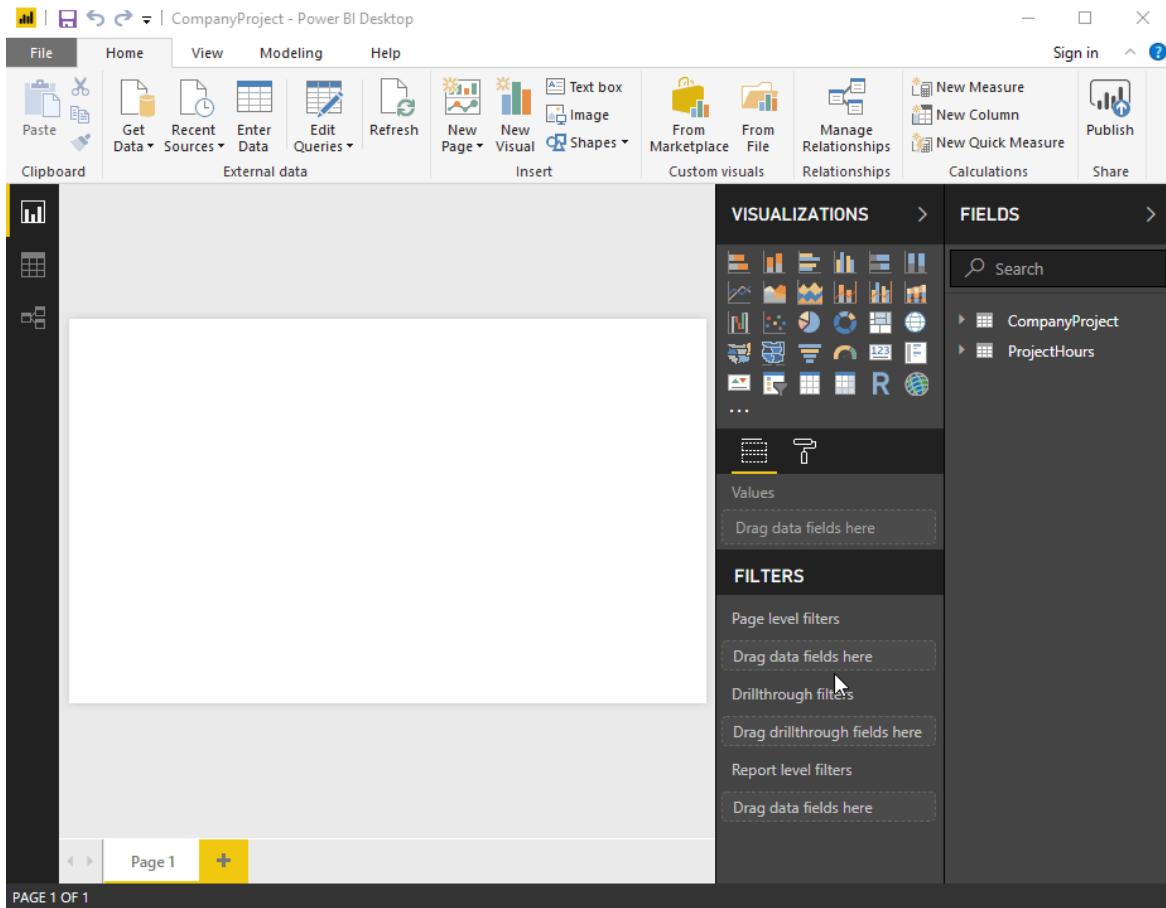
Create a relationship with autodetect

On the Modeling tab, select **Manage relationships** > **Autodetect**.



Create a relationship manually

1. On the **Modeling** tab, select **Manage relationships** > **New**.
2. In the **Create relationship** dialog box, in the first table drop-down list, select a table. Select the column you want to use in the relationship.
3. In the second table drop-down list, select the other table you want in the relationship. Select the other column you want to use, and then select **OK**.



By default, Power BI Desktop automatically configures the options **Cardinality** (direction), **Cross filter direction**, and **Make this relationship active** for your new relationship. However, you can change these settings if necessary. For more information, see [Understanding additional options](#).

If none of the tables selected for the relationship has unique values, you'll see the following error: *One of the columns must have unique values*. At least one table in a relationship *must* have a distinct, unique list of key values, which is a common requirement for all relational database technologies.

If you encounter that error, there are a couple ways to fix the issue:

- Use **Remove Duplicates** to create a column with unique values. The drawback to this approach is that you might lose information when duplicate rows are removed; often a key (row) is duplicated for good reason.
- Add an intermediary table made of the list of distinct key values to the model, which will then be linked to both original columns in the relationship.

For more information, see this [blog post](#).

Edit a relationship

1. On the **Modeling** tab, select **Manage relationships**.
2. In the **Manage relationships** dialog box, select the relationship, then select **Edit**.

Configure additional options

When you create or edit a relationship, you can configure additional options. By default, Power BI Desktop automatically configures additional options based on its best guess, which can be different for each relationship based on the data in the columns.

Cardinality

The **Cardinality** option can have one of the following settings:

Many to one (*:1): A many-to-one relationship is the most common, default type of relationship. It means the column in a given table can have more than one instance of a value, and the other related table, often known as the lookup table, has only one instance of a value.

One to one (1:1): In a one-to-one relationship, the column in one table has only one instance of a particular value, and the other related table has only one instance of a particular value.

One to many (1:*): In a one-to-many relationship, the column in one table has only one instance of a particular value, and the other related table can have more than one instance of a value.

Many to many (*:*): With composite models, you can establish a many-to-many relationship between tables, which removes requirements for unique values in tables. It also removes previous workarounds, such as introducing new tables only to establish relationships. For more information, see [Relationships with a many-many cardinality](#).

For more information about when to change cardinality, see [Understanding additional options](#).

Cross filter direction

The **Cross filter direction** option can have one of the following settings:

Both: For filtering purposes, both tables are treated as if they're a single table. The **Both** setting works well with a single table that has many lookup tables that surround it. An example is a sales actuals table with a lookup table for its department. This configuration is often called a star schema configuration (a central table with several lookup tables). However, if you have two or more tables that also have lookup tables (with some in common) then you wouldn't want to use the **Both** setting. To continue the previous example, in this case, you also have a budget sales table that records target budget for each department. And, the department table is connected to both the sales and the budget table. Avoid the **Both** setting for this kind of configuration.

Single: The most common, default direction, which means filtering choices in connected tables work on the table where values are being aggregated. If you import a Power Pivot in Excel 2013 or earlier data model, all relationships will have a single direction.

For more information about when to change cross filter direction, see [Understanding additional options](#).

Make this relationship active

When checked, the relationship serves as the active, default relationship. In cases where there is more than one relationship between two tables, the active relationship provides a way for Power BI Desktop to automatically create visualizations that include both tables.

For more information about when to make a particular relationship active, see [Understanding additional options](#).

Understanding relationships

Once you've connected two tables together with a relationship, you can work with the data in both tables as if they were a single table, freeing you from having to worry about relationship details, or flattening those tables into a single table before importing them. In many situations, Power BI Desktop can automatically create relationships for you. However, if Power BI Desktop can't determine with a high-degree of certainty that a

relationship between two tables should exist, it doesn't automatically create the relationship. In that case, you must do so.

Let's go through a quick tutorial, to better show you how relationships work in Power BI Desktop.

TIP

You can complete this lesson yourself:

1. Copy the following **ProjectHours** table into an Excel worksheet (excluding the title), select all of the cells, and then select **Insert > Table**.
2. In the **Create Table** dialog box, select **OK**.
3. Select any table cell, select **Table Design > Table Name**, and then enter *ProjectHours*.
4. Do the same for the **CompanyProject** table.
5. Import the data by using **Get Data** in Power BI Desktop. Select the two tables as a data source, and then select **Load**.

The first table, **ProjectHours**, is a record of work tickets that record the number of hours a person has worked on a particular project.

ProjectHours

TICKET	SUBMITTEDBY	HOURS	PROJECT	DATESUBMIT
1001	Brewer, Alan	22	Blue	1/1/2013
1002	Brewer, Alan	26	Red	2/1/2013
1003	Ito, Shu	34	Yellow	12/4/2012
1004	Brewer, Alan	13	Orange	1/2/2012
1005	Bowen, Eli	29	Purple	10/1/2013
1006	Bento, Nuno	35	Green	2/1/2013
1007	Hamilton, David	10	Yellow	10/1/2013
1008	Han, Mu	28	Orange	1/2/2012
1009	Ito, Shu	22	Purple	2/1/2013
1010	Bowen, Eli	28	Green	10/1/2013
1011	Bowen, Eli	9	Blue	10/15/2013

This second table, **CompanyProject**, is a list of projects with an assigned priority: A, B, or C.

CompanyProject

PROJNAME	PRIORITY
Blue	A
Red	B

PROJNAME	PRIORITY
Green	C
Yellow	C
Purple	B
Orange	C

Notice that each table has a project column. Each is named slightly different, but the values look like they're the same. That's important, and we'll get back to it in soon.

Now that we have our two tables imported into a model, let's create a report. The first thing we want to get is the number of hours submitted by project priority, so we select **Priority** and **Hours** from the **Fields** pane.

The screenshot shows the Power BI Fields pane on the right side of the interface. On the left, there is a preview of a table with columns 'Priority' and 'Hours'. The data in the preview table is as follows:

	Priority	Hours
A	256	
B	256	
C	256	
Total	256	

The 'Priority' field is selected in the preview. In the Fields pane, under the 'CompanyProject' table, the 'Priority' field is checked. Under the 'ProjectHours' table, the 'Hours' field is checked. Other fields like 'ProjName' and 'DateSubmit' are also listed but not selected.

If we look at our table in the report canvas, you'll see the number of hours is 256 for each project, which is also the total. Clearly this number isn't correct. Why? It's because we can't calculate a sum total of values from one table (**Hours** in the **Project** table), sliced by values in another table (**Priority** in the **CompanyProject** table) without a relationship between these two tables.

So, let's create a relationship between these two tables.

Remember those columns we saw in both tables with a project name, but with values that look alike? We'll use these two columns to create a relationship between our tables.

Why these columns? Well, if we look at the **Project** column in the **ProjectHours** table, we see values like Blue, Red, Yellow, Orange, and so on. In fact, we see several rows that have the same value. In effect, we have many color values for **Project**.

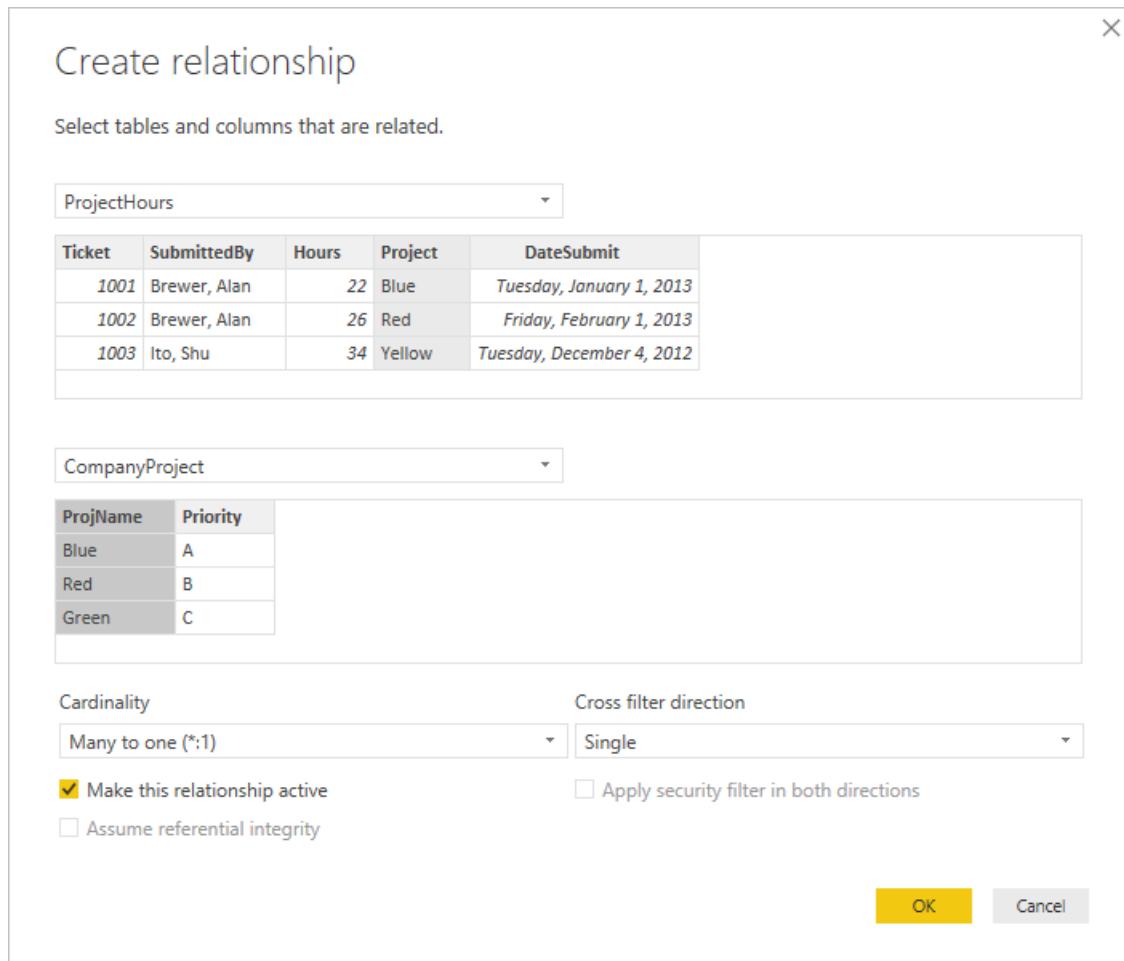
If we look at the **ProjName** column in the **CompanyProject** table, we see there's only one of each of the color values for the project name. Each color value in this table is unique, and that's important, because we can create a relationship between these two tables. In this case, a many-to-one relationship. In a many-to-one relationship, at least one column in one of the tables must contain unique values. There are some additional options for some relationships, which we'll look at later. For now, let's create a relationship between the project columns in each of our two tables.

To create the new relationship

1. Select **Manage relationships** from the **Modeling** tab.
2. In **Manage relationships**, select **New** to open the **Create relationship** dialog box, where we can

select the tables, columns, and any additional settings we want for our relationship.

3. In the first drop-down list, select **ProjectHours** as the first table, then select the **Project** column. This side is the *many* side of our relationship.
4. In the second drop-down list, **CompanyProject** is preselected as the second table. Select the **ProjName** column. This side is the *one* side of our relationship.
5. Accept the defaults for the relationship options, and then select **OK**.



6. In the **Manage relationships** dialog box, select **Close**.

In the interest of full disclosure, you just created this relationship the hard way. You could have just selected **Autodetect** in the **Manage relationships** dialog box. In fact, autodetect would have automatically created the relationship for you when you loaded the data if both columns had the same name. But, what's the challenge in that?

Now, let's look at the table in our report canvas again.

The screenshot shows the Power BI Desktop interface. On the left is a report visual displaying a table with columns 'Priority' and 'Hours'. The data rows are A (31), B (77), C (148), and a total row (Total 256). On the right is the 'Fields' pane, which includes sections for 'Visualizations' and 'Filters'. Under 'Visualizations', there are two tables: 'CompanyProject' and 'ProjectHours'. 'CompanyProject' has a checked checkbox for 'Priority'. 'ProjectHours' has a checked checkbox for 'Σ Hours'. There is also a section for 'DateSubmit' with checkboxes for 'Project', 'SubmittedBy', and 'Ticket'.

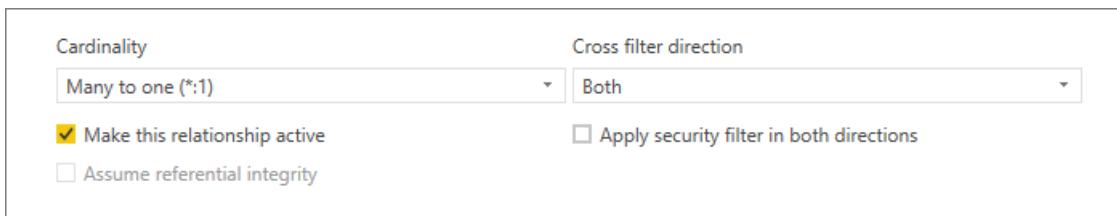
That looks a whole lot better, doesn't it?

When we sum up hours by **Priority**, Power BI Desktop looks for every instance of the unique color values in the **CompanyProject** lookup table, looks for every instance of each of those values in the **ProjectHours** table, and then calculates a sum total for each unique value.

That was easy. In fact, with autodetect, you might not even have to do that much.

Understanding additional options

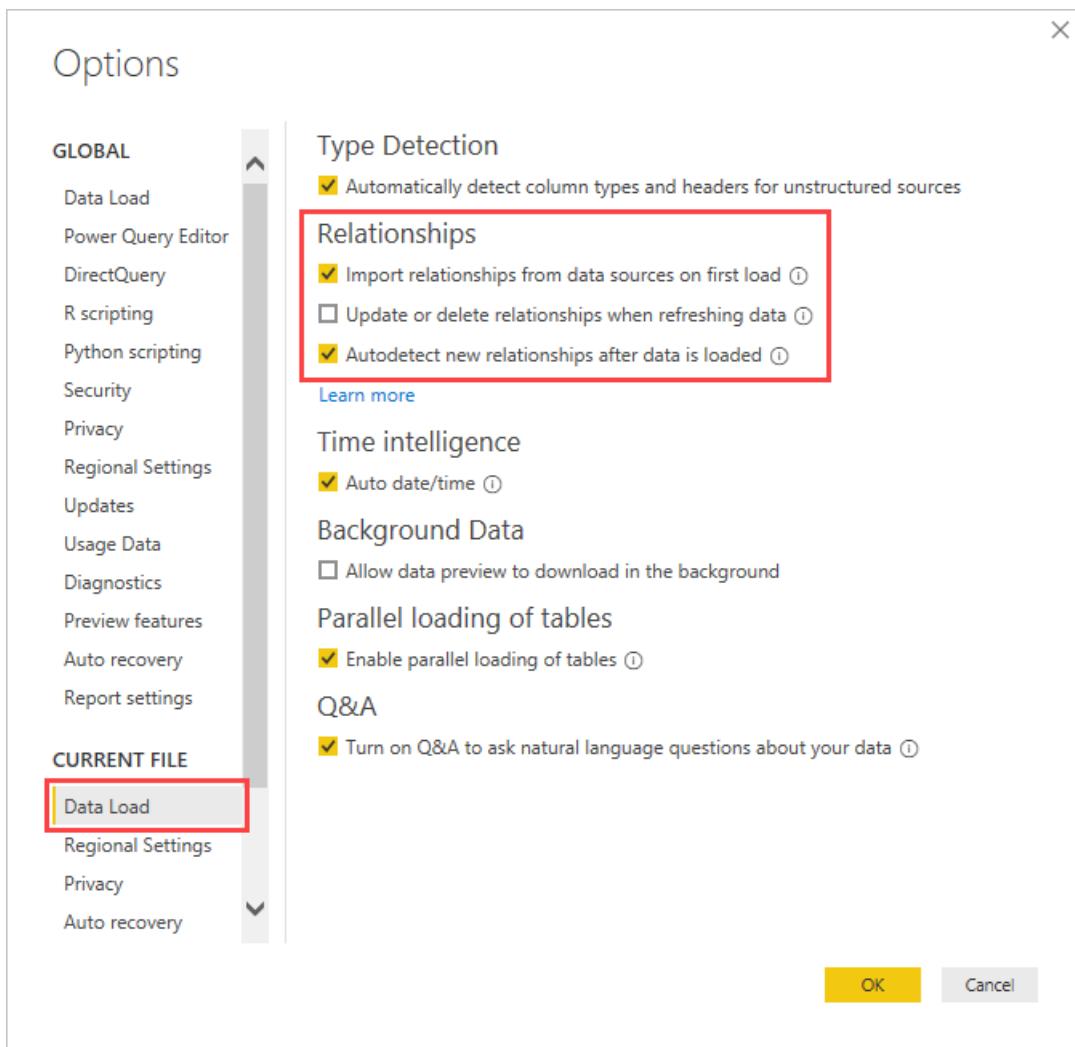
When a relationship is created, either with autodetect or one you create manually, Power BI Desktop automatically configures additional options based on the data in your tables. These additional relationship options are located in the lower portion of the **Create relationship** and **Edit relationship** dialog boxes.



Power BI typically sets these options automatically and you won't need to adjust them; however, there are several situations where you might want to configure these options yourself.

Automatic relationship updates

You can manage how Power BI treats and automatically adjusts relationships in your reports and models. To specify how Power BI handles relationships options, select **File > Options and settings > Options** from Power BI Desktop, and then select **Data Load** in the left pane. The options for **Relationships** appear.



There are three options that can be selected and enabled:

- **Import relationships from data sources on first load:** This option is selected by default. When it's selected, Power BI checks for relationships defined in your data source, such as foreign key/primary key relationships in your data warehouse. If such relationships exist, they're mirrored into the Power BI data model when you initially load data. This option enables you to quickly begin working with your model, rather than requiring you find or define those relationships yourself.
- **Update or delete relationships when refreshing data:** This option is unselected by default. If you select it, Power BI checks for changes in data source relationships when your dataset is refreshed. If those relationships changed or are removed, Power BI mirrors those changes in its own data model, updating or deleting them to match.

WARNING

If you're using row-level security that relies on the defined relationships, we don't recommend selecting this option. If you remove a relationship that your RLS settings rely on, your model might become less secure.

- **Autodetect new relationships after data is loaded:** This option is described in [Autodetect during load](#).

Future updates to the data require a different cardinality

Normally, Power BI Desktop can automatically determine the best cardinality for the relationship. If you do need to override the automatic setting, because you know the data will change in the future, you can change it with the **Cardinality** control. Let's look at an example where we need to select a different cardinality.

The **CompanyProjectPriority** table is a list of all company projects and their priority. The **ProjectBudget** table is the set of projects for which a budget has been approved.

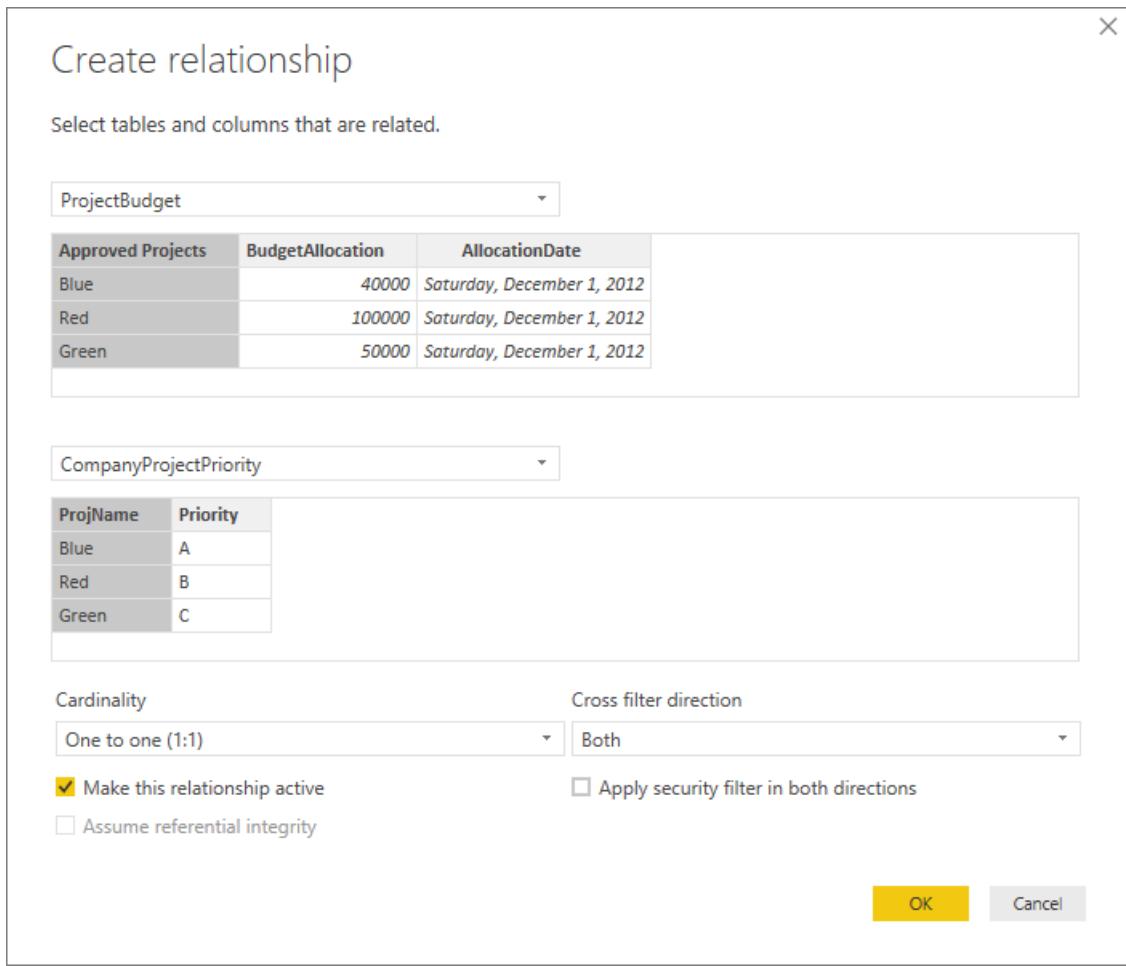
CompanyProjectPriority

PROJNAME	PRIORITY
Blue	A
Red	B
Green	C
Yellow	C
Purple	B
Orange	C

ProjectBudget

APPROVED PROJECTS	BUDGETALLOCATION	ALLOCATIONDATE
Blue	40,000	12/1/2012
Red	100,000	12/1/2012
Green	50,000	12/1/2012

If we create a relationship between the **Approved Projects** column in the **ProjectBudget** table and the **ProjectName** column in the **CompanyProjectPriority** table, Power BI automatically sets **Cardinality** to **One to one (1:1)** and **Cross filter direction** to **Both**.



The reason Power BI makes these settings is because, to Power BI Desktop, the best combination of the two tables is as follows:

PROJNAME	PRIORITY	BUDGETALLOCATION	ALLOCATIONDATE
Blue	A	40,000	12/1/2012
Red	B	100,000	12/1/2012
Green	C	50,000	12/1/2012
Yellow	C		
Purple	B		
Orange	C		

There's a one-to-one relationship between our two tables because there are no repeating values in the combined table's **ProjName** column. The **ProjName** column is unique, because each value occurs only once; therefore, the rows from the two tables can be combined directly without any duplication.

But, let's say you know the data will change the next time you refresh it. A refreshed version of the **ProjectBudget** table now has additional rows for the Blue and Red projects:

ProjectBudget

APPROVED PROJECTS	BUDGETALLOCATION	ALLOCATIONDATE
Blue	40,000	12/1/2012
Red	100,000	12/1/2012
Green	50,000	12/1/2012
Blue	80,000	6/1/2013
Red	90,000	6/1/2013

These additional rows mean the best combination of the two tables now looks like this:

PROJNAME	PRIORITY	BUDGETALLOCATION	ALLOCATIONDATE
Blue	A	40,000	12/1/2012
Red	B	100,000	12/1/2012
Green	C	50,000	12/1/2012
Yellow	C		
Purple	B		
Orange	C		
Blue	A	80000	6/1/2013
Red	B	90000	6/1/2013

In this new combined table, the **ProjName** column has repeating values. The two original tables won't have a one-to-one relationship once the table is refreshed. In this case, because we know those future updates will cause the **ProjName** column to have duplicates, we want to set the **Cardinality** to be **Many to one (*:1)**, with the *many* side on **ProjectBudget** and the *one* side on **CompanyProjectPriority**.

Adjusting Cross filter direction for a complex set of tables and relationships

For most relationships, the cross filter direction is set to **Both**. There are, however, some more uncommon circumstances where you might need to set this option differently from the default, like if you're importing a model from an older version of Power Pivot, where every relationship is set to a single direction.

The **Both** setting enables Power BI Desktop to treat all aspects of connected tables as if they're a single table. There are some situations, however, where Power BI Desktop can't set a relationship's cross filter direction to **Both** and also keep an unambiguous set of defaults available for reporting purposes. If a relationship cross filter direction isn't set to **Both**, then it's usually because it would create ambiguity. If the default cross filter setting isn't working for you, try setting it to a particular table or to **Both**.

Single direction cross filtering works for many situations. In fact, if you've imported a model from Power Pivot in Excel 2013 or earlier, all of the relationships will be set to single direction. Single direction means that filtering choices in connected tables work on the table where aggregation work is happening. Sometimes, understanding

cross filtering can be a little difficult, so let's look at an example.

With single direction cross filtering, if you create a report that summarizes the project hours, you can then choose to summarize (or filter) by the **CompanyProject** table and its **Priority** column or the **CompanyEmployee** table and its **City** column. If however, you want to count the number of employees per projects (a less common question), it won't work. You'll get a column of values that are all the same. In the following example, both relationship's cross filtering direction is set to a single direction: towards the **ProjectHours** table. In the **Values** well, the **Project** field is set to **Count**:

The screenshot shows the Power BI Data View interface. On the left is a table visualization titled "Employee Count of Project". It contains the following data:

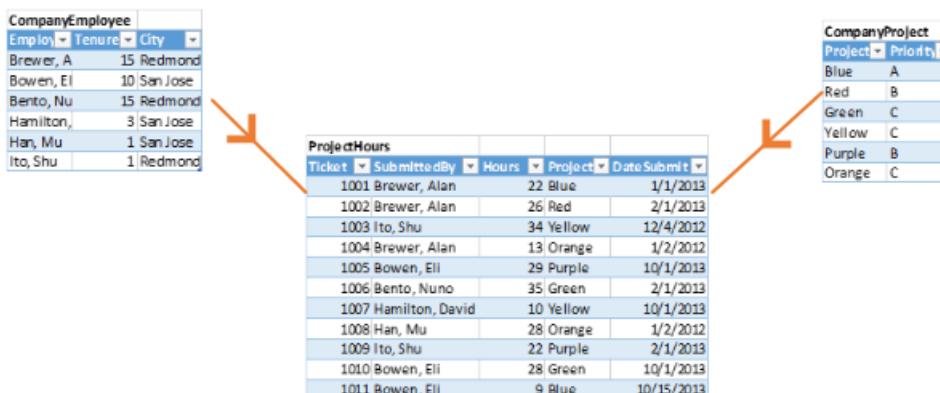
Employee	Count of Project
Bento, Nuno	6
Bowen, Eli	6
Brewer, Alan	6
Hamilton, David	6
Han, Mu	6
Ito, Shu	6
Total	6

At the bottom of the table are three filter icons: a magnifying glass, a checkmark, and an ellipsis.

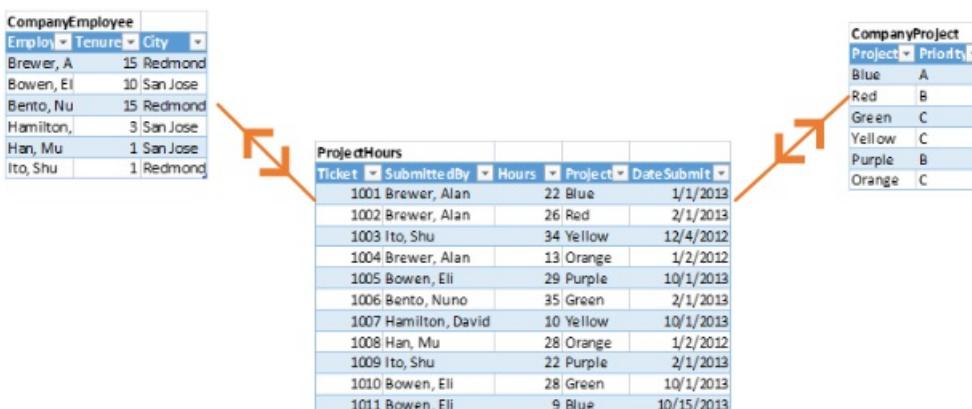
In the center is the "Visualizations" pane, which displays various chart and report icons.

On the right is the "Fields" pane. It includes a search bar and sections for "CompanyEmployee", "CompanyProject", and "ProjectHours". Under "CompanyEmployee", "Employee" is checked. Under "CompanyProject", "Project" is checked. Under "ProjectHours", "Count of Project" is selected in the "Values" section.

Filter specification will flow from **CompanyProject** to **ProjectHours** (as shown in the following image), but it won't flow up to **CompanyEmployee**.

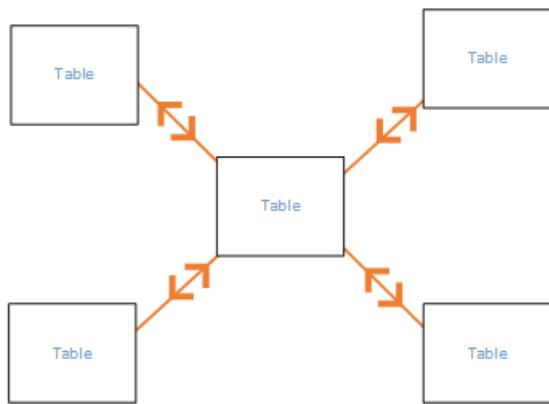


However, if you set the cross filtering direction to **Both**, it will work. The **Both** setting allows the filter specification to flow up to **CompanyEmployee**.

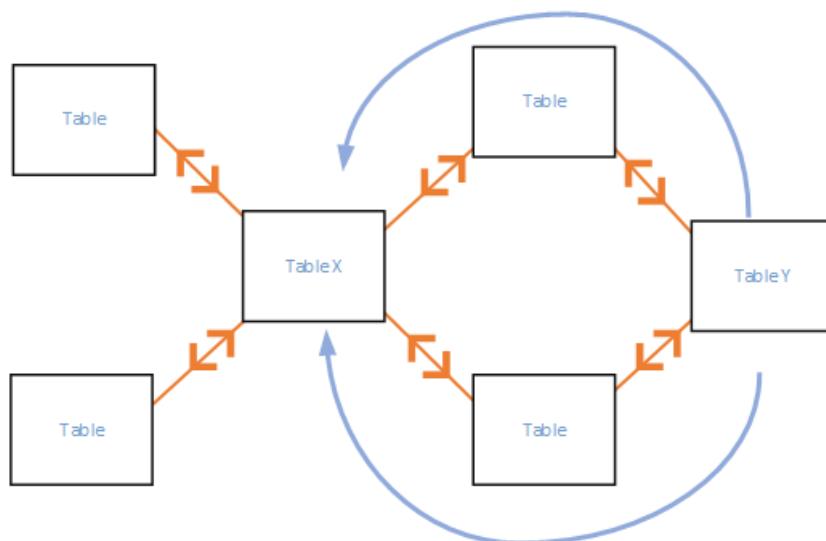


With the cross filtering direction set to **Both**, our report now appears correct:

Cross filtering both directions works well for a pattern of table relationships such as the pattern above. This schema is most commonly called a star schema, like this:



Cross filtering direction does not work well with a more general pattern often found in databases, like in this diagram:



If you have a table pattern like this, with loops, then cross filtering can create an ambiguous set of relationships. For instance, if you sum up a field from TableX and then choose to filter by a field on TableY, then it's not clear how the filter should travel, through the top table or the bottom table. A common example of this kind of pattern is with TableX as a sales table with actuals data and for TableY to be budget data. Then, the tables in the middle are lookup tables that both tables use, such as division or region.

As with active/inactive relationships, Power BI Desktop won't allow a relationship to be set to **Both** if it will

create ambiguity in reports. There are several different ways you can handle this situation. Here are the two most common:

- Delete or mark relationships as inactive to reduce ambiguity. Then, you might be able to set a relationship cross filtering as **Both**.
- Bring in a table twice (with a different name the second time) to eliminate loops. Doing so makes the pattern of relationships like a star schema. With a star schema, all of the relationships can be set to **Both**.

Wrong active relationship

When Power BI Desktop automatically creates relationships, it sometimes encounters more than one relationship between two tables. When this situation happens, only one of the relationships is set to be active. The active relationship serves as the default relationship, so that when you choose fields from two different tables, Power BI Desktop can automatically create a visualization for you. However, in some cases the automatically selected relationship can be wrong. Use the **Manage relationships** dialog box to set a relationship as active or inactive, or set the active relationship in the **Edit relationship** dialog box.

To ensure there's a default relationship, Power BI Desktop allows only a single active relationship between two tables at a given time. Therefore, you must first set the current relationship as inactive and then set the relationship you want to be active.

Let's look at an example. The first table is **ProjectTickets**, and the second table is **EmployeeRole**.

ProjectTickets

TICKET	OPENEDBY	SUBMITTEDBY	HOURS	PROJECT	DATESUBMIT
1001	Perham, Tom	Brewer, Alan	22	Blue	1/1/2013
1002	Roman, Daniel	Brewer, Alan	26	Red	2/1/2013
1003	Roth, Daniel	Ito, Shu	34	Yellow	12/4/2012
1004	Perham, Tom	Brewer, Alan	13	Orange	1/2/2012
1005	Roman, Daniel	Bowen, Eli	29	Purple	10/1/2013
1006	Roth, Daniel	Bento, Nuno	35	Green	2/1/2013
1007	Roth, Daniel	Hamilton, David	10	Yellow	10/1/2013
1008	Perham, Tom	Han, Mu	28	Orange	1/2/2012
1009	Roman, Daniel	Ito, Shu	22	Purple	2/1/2013
1010	Roth, Daniel	Bowen, Eli	28	Green	10/1/2013
1011	Perham, Tom	Bowen, Eli	9	Blue	10/15/2013

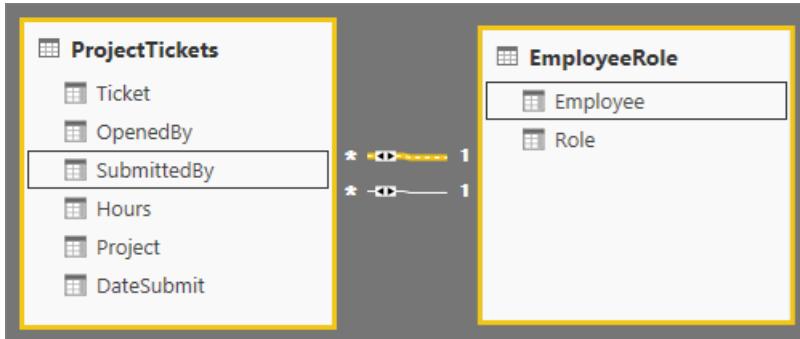
EmployeeRole

EMPLOYEE	ROLE
Bento, Nuno	Project Manager

EMPLOYEE	ROLE
Bowen, Eli	Project Lead
Brewer, Alan	Project Manager
Hamilton, David	Project Lead
Han, Mu	Project Lead
Ito, Shu	Project Lead
Perham, Tom	Project Sponsor
Roman, Daniel	Project Sponsor
Roth, Daniel	Project Sponsor

There are actually two relationships here:

- Between **Employee** in the **EmployeeRole** table and **SubmittedBy** in the **ProjectTickets** table.
- Between **OpenedBy** in the **ProjectTickets** table and **Employee** in the **EmployeeRole** table.



If we add both relationships to the model (**OpenedBy** first), then the **Manage relationships** dialog box shows that **OpenedBy** is active:

Manage relationships

Active	From: Table (Column)	To: Table (Column)
<input type="checkbox"/>	EmployeeRole (Employee)	ProjectTickets (SubmittedBy)
<input checked="" type="checkbox"/>	ProjectTickets (OpenedBy)	EmployeeRole (Employee)

New... Autodetect... Edit... Delete Close

Now, if we create a report that uses **Role** and **Employee** fields from **EmployeeRole**, and the **Hours** field from **ProjectTickets** in a table visualization in the report canvas, we see only project sponsors because they're the only ones that opened a project ticket.

The screenshot shows the Power BI Report view. On the left, there is a table visualization with three columns: Role, Employee, and Hours. The data in the table is:

Role	Employee	Hours
Project Sponsor	Perham, Tom	72
Project Sponsor	Roman, Daniel	77
Project Sponsor	Roth, Daniel	107
Total		256

On the right, the Fields pane is open, showing the available fields categorized by table:

- EmployeeRole** (checked):
 - Employee
 - Role
- ProjectTickets** (checked):
 - DateSubmit
 - Hours (summarized)
 - OpenedBy
 - Project
 - SubmittedBy
 - Ticket (summarized)

We can change the active relationship and get **SubmittedBy** instead of **OpenedBy**. In **Manage relationships**, uncheck the **ProjectTickets(OpenedBy)** to **EmployeeRole(Employee)** relationship, and then check the **EmployeeRole(Employee)** to **Project Tickets(SubmittedBy)** relationship.

Manage relationships

Active	From: Table (Column)	To: Table (Column)
<input checked="" type="checkbox"/>	EmployeeRole (Employee)	ProjectTickets (SubmittedBy)
<input type="checkbox"/>	ProjectTickets (OpenedBy)	EmployeeRole (Employee)

New... Autodetect... Edit... Delete Close

See all of your relationships in Relationship view

Sometimes your model has multiple tables and complex relationships between them. **Relationship view** in Power BI Desktop shows all of the relationships in your model, their direction, and cardinality in an easy to understand and customizable diagram.

To learn more, see [Work with Relationship view in Power BI Desktop](#).

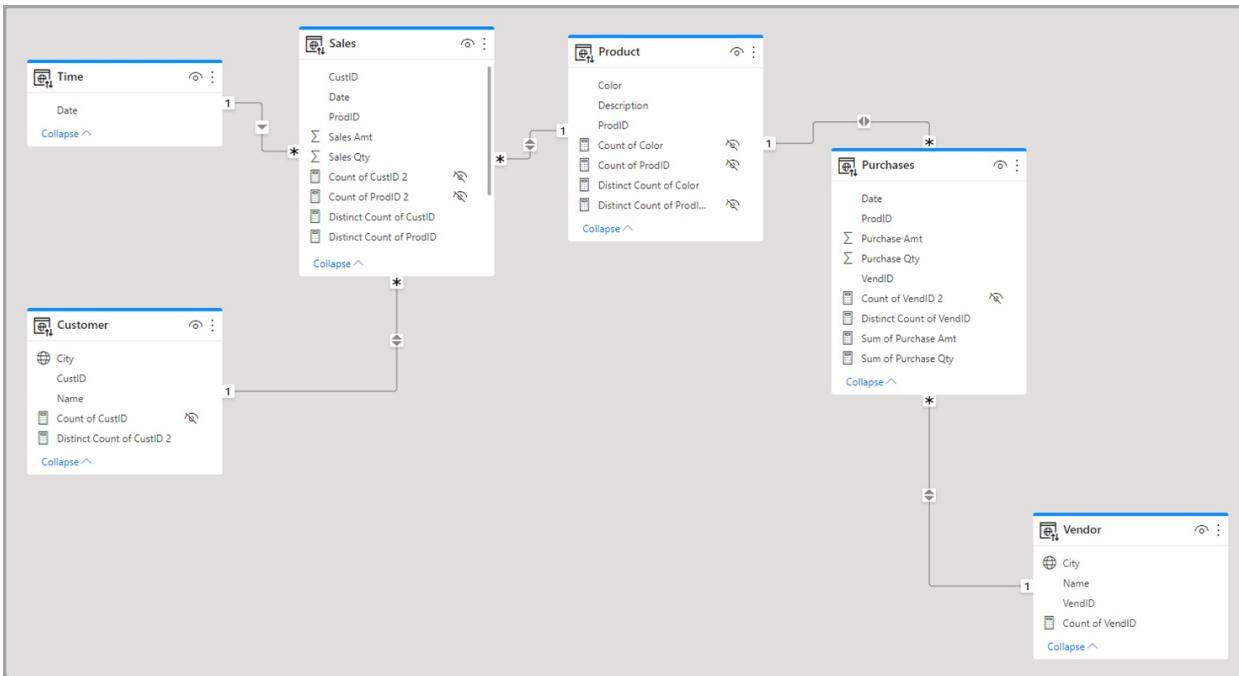
Troubleshooting

This section provides guidance and troubleshooting information when working with relationships in Power BI.

Relationships between fields cannot be determined

Power BI attempts to show relevant data in visuals by inferring the relationships from the model being used. Sometimes such inferences aren't obvious, and you might be surprised to see an error in your visual, indicating there is no relationship between certain columns.

To explain how Power BI determines whether fields are related, let's use an example model to illustrate a few scenarios in the following sections. The following image shows the sample model we'll use in the example scenarios.



Scenario 1: Traditional star schema and no measure constraint provided. Referring to the sample model in the previous image, let's look first at the right half of the image with the *Vendor - Purchases - Product* tables. This is a traditional star schema with the Fact table (*Purchases*) and two Dimension tables (*Product* and *Vendor*). The relationship between the dimension tables and the fact table is *1 to Many* (one product corresponds to many purchases, one vendor corresponds to many purchases). In this type of schema, we can answer questions like *What sales do we have for product X?* and *What sales do we have for Vendor Y?* and *What products does Vendor Y sell?*

If we want to correlate *Products* and *Vendors*, we can do so by looking at the *Purchases* table to see if there is an entry with the same product and vendor. A sample query might look like the following:

```
Correlate Product[Color] with Vendor[Name] where CountRows(Purchases)>0
```

The `where CountRows(Purchases)>0` is an implicit constraint that Power BI would add to ensure relevant data is returned. By doing this correlation through the *Purchases* table, we can return pairings of Product-Vendor that have at least one entry in a fact table, pairings that make sense from the data perspective. You can expect any nonsensical combinations of Product-Vendor for which there has never been a sale (which would be useless for analysis) will not be displayed.

Scenario 2: Traditional star schema and measure constraint provided. In the previous example in Scenario 1, if the user provides a constraint in the form of summarized column (Sum/Average/Count of Purchase Qty, for example) or a model measure (Distinct Count of VendID), Power BI can generate a query in the form of the following:

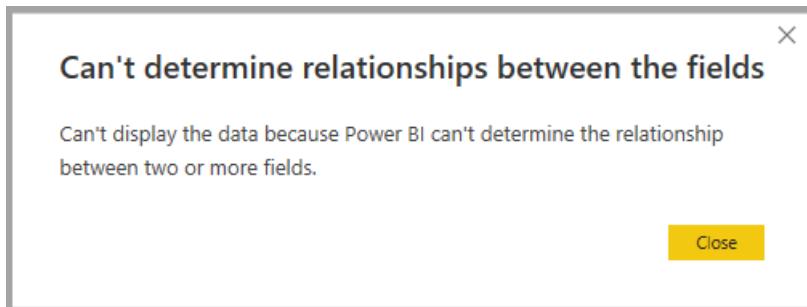
```
Correlate Product[Color] with Vendor[Name] where MeasureConstraint is not blank
```

In such a case, Power BI attempts to return combinations that have meaningful values for the constraint provided by the user (non-blank). Power BI does not need to also add its own implicit constraint of `CountRows(Purchases)>0`, such as what was done like in the previous Scenario 1, because the constraint provided by the user is sufficient.

Scenario 3: Non-star schema and no measure constraint provided. In this scenario, we focus our attention to the center of the model, where we have the *Sales - Product - Purchases* tables, where we have one dimension table (*Product*) and two Fact Tables (*Sales*, *Purchases*). Since this is not a star schema, we can't answer the same kind of questions as we had in Scenario 1. Let's say we try to correlate *Purchases* and *Sales*; since *Purchases* has a *Many to 1* relationship with *Product*, and *Product* has a *1 to Many* relationship with *Sales*, *Sales* and *Purchases* are indirectly *Many to Many*. We can link one *Product* to many *Purchases* and one *Product* to

many sales, but we cannot link one *Sale* to many *Purchases* or vice versa. We can only link many *Purchases* to many *Sales*.

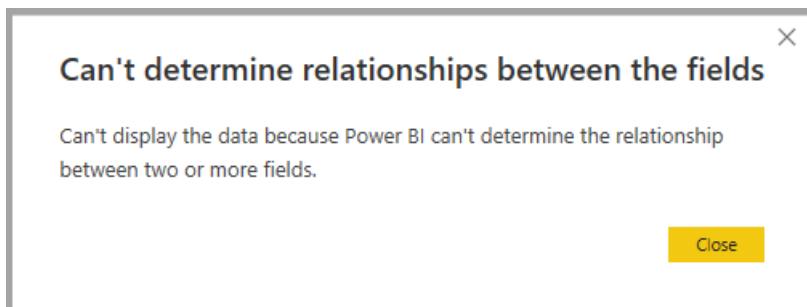
In this situation, if we try to combine *Purchase[VenID]* and *Sales[CustID]* in a visual, Power BI does not have a concrete constraint it can apply, due to the *Many to Many* relationship between those tables. Though there may be custom constraints (not necessarily stemming from the relationships established in the model) that can be applied for various scenarios, Power BI cannot infer a default constraint solely based on the relationships. If Power BI attempted to return all combinations of the two tables, it would create a large cross join and return non-relevant data. Instead of this, Power BI raises an error in the visual, such as the following.



Scenario 4: Non-star schema and measure constraint provided. If we take the example from Scenario 3 and add a user provided constraint in the form of a summarized column (*Count of Product[ProdID]* for example) or a model measure (*Sales[Total Qty]*) Power BI can generate a query in the form of *Correlate Purchase[VenID]* and *Sales[CustID]* where *MeasureConstraint* is not blank.

In this case, Power BI respects the user's constraint as being the sole constraint Power BI needs to apply, and return the combinations that produce non-blank values for it. The user has guided Power BI to the scenario it wants, and Power BI applies the guidance.

Scenario 5: When a measure constraint is provided but it is partially related to the columns. There are cases where the measure constraint provided by the user is not entirely related to all the columns in the visual. A model measure always relates everything; Power BI treats this as a black box when attempting to find relationships between columns in the visual, and assume the user knows what they are doing by using it. However, summarized columns in the form of *Sum*, *Average*, and similar summaries chosen from the user interface can be related to only a subset of the columns/tables used in the visual based on the relationships of the table to which that column belongs. As such, the constraint applies to some pairings of columns, but not to all, in which case Power BI attempts to find default constraints it can apply for the columns that are not related by the user provided constraint (such as in Scenario 1). If Power BI cannot find any, the following error is returned.



Resolving relationship errors

When you see the **Can't determine relationships between the fields** error, you can take the following steps to attempt to resolve the error:

1. Check your model. Is it set up appropriately for the types of questions you want answered from your analysis? Can you change some of the relationships between tables? Can you avoid creating an indirect *Many to Many*?

Consider converting your reversed Vshape schema to two tables, and use a direct *Many to Many* relationship between them as described in [apply many-many relationships in Power BI Desktop](#).

2. Add a constraint to the visual in the form of a summarized column or a model measure.
3. If a summarized column is added and there still is an error, consider using a model measure.

Next steps

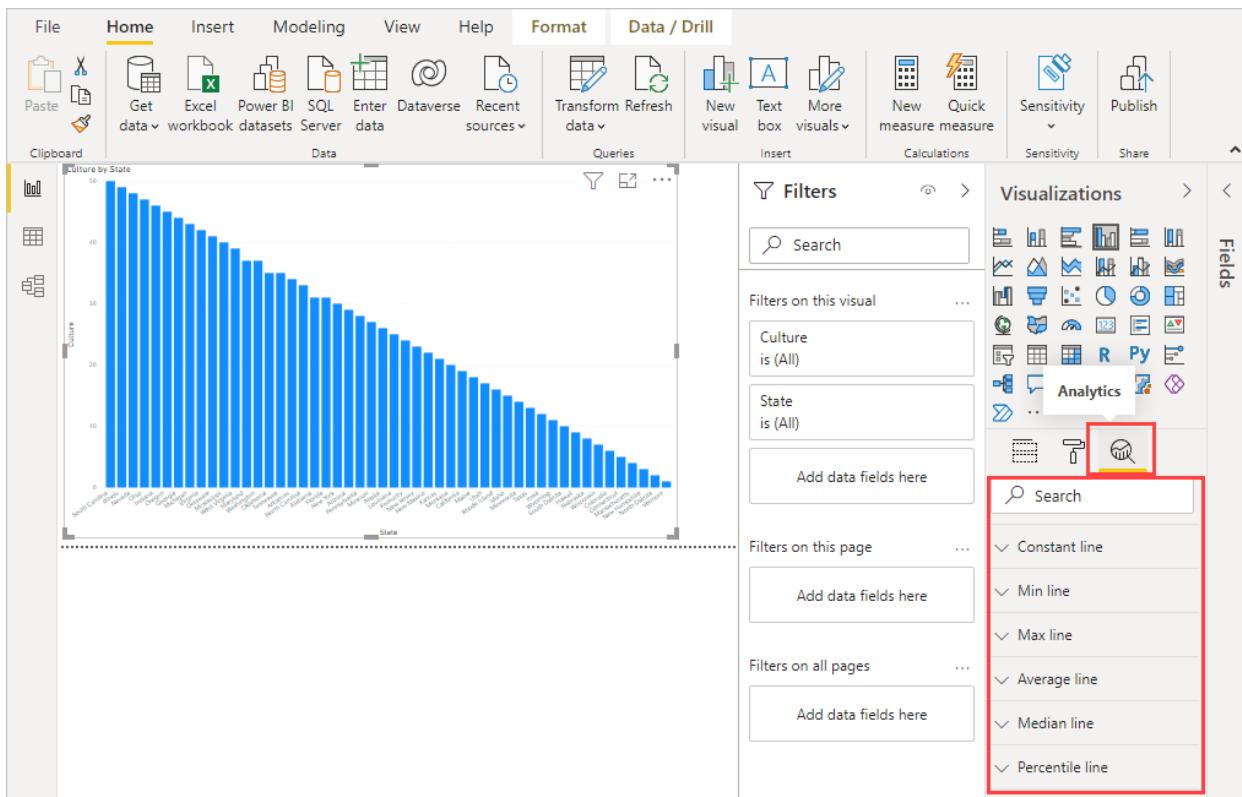
For more information about models and relationships, see the following articles:

- [Use composite models in Power BI Desktop](#)
- [Storage mode in Power BI Desktop](#)
- [Use DirectQuery in Power BI](#)
- [Power BI data sources](#)

Use the Analytics pane in Power BI Desktop

3/30/2022 • 4 minutes to read • [Edit Online](#)

With the **Analytics** pane in Power BI Desktop, you can add dynamic *reference lines* to visuals, and provide focus for important trends or insights. The **Analytics** icon and pane is found in the **Visualizations** area of Power BI Desktop.

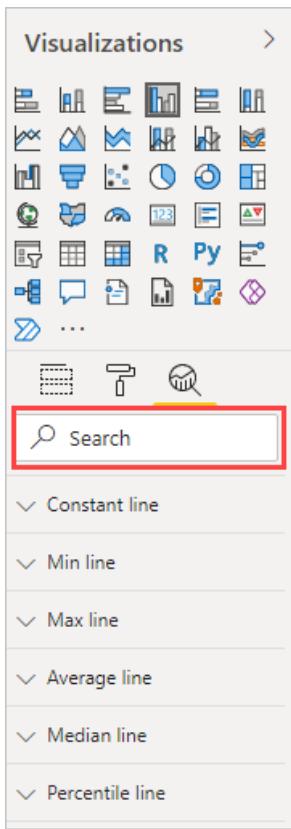


NOTE

The **Analytics** pane only appears when you select a visual on the Power BI Desktop canvas.

Search within the Analytics pane

You can search within the **Analytics** pane, which is a subsection of the **Visualizations** pane. The search box appears when you select the **Analytics** icon.



Use the Analytics pane

With the **Analytics** pane, you can create the following types of dynamic reference lines:

- X-Axis constant line
- Y-Axis constant line
- Min line
- Max line
- Average line
- Median line
- Percentile line
- Symmetry shading

NOTE

Not all lines are available for all visual types.

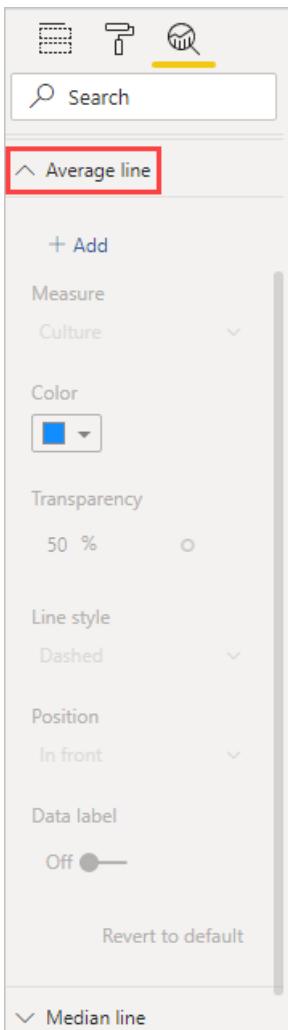
The following sections show how you can use the **Analytics** pane and dynamic reference lines in your visualizations.

To view the available dynamic reference lines for a visual, follow these steps:

1. Select or create a visual, then select the **Analytics** icon from the **Visualizations** section.

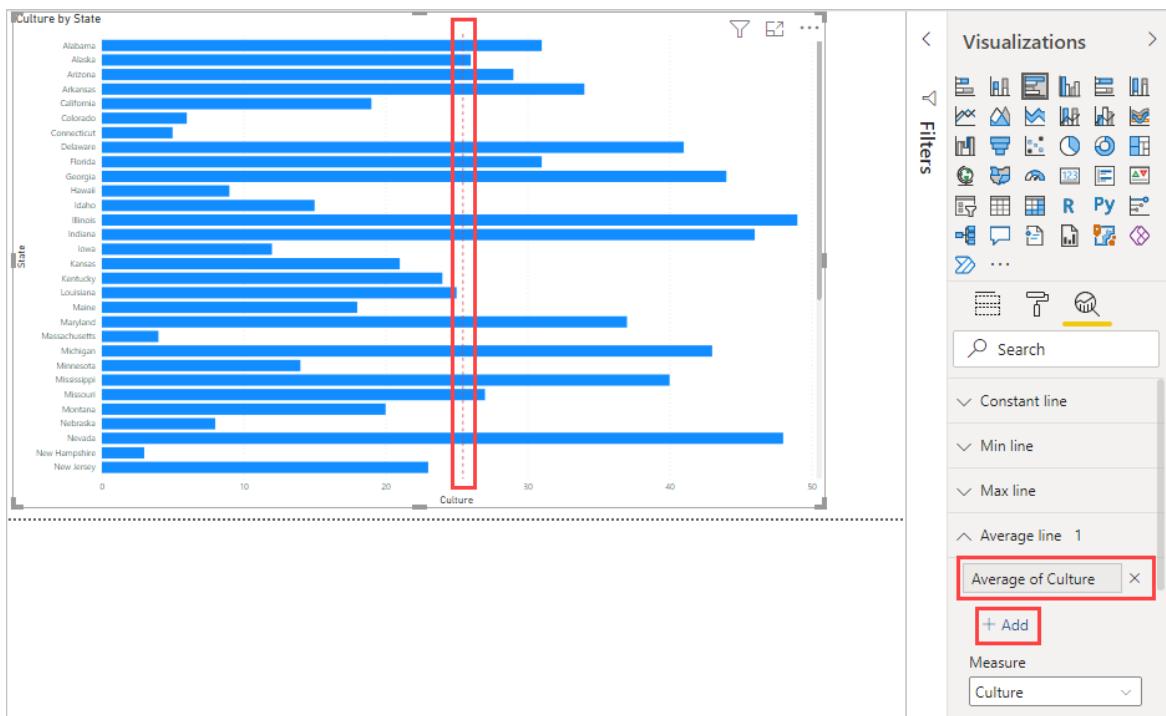


2. Select the type of line you want to create to expand its options. In this case, we'll select **Average line**.

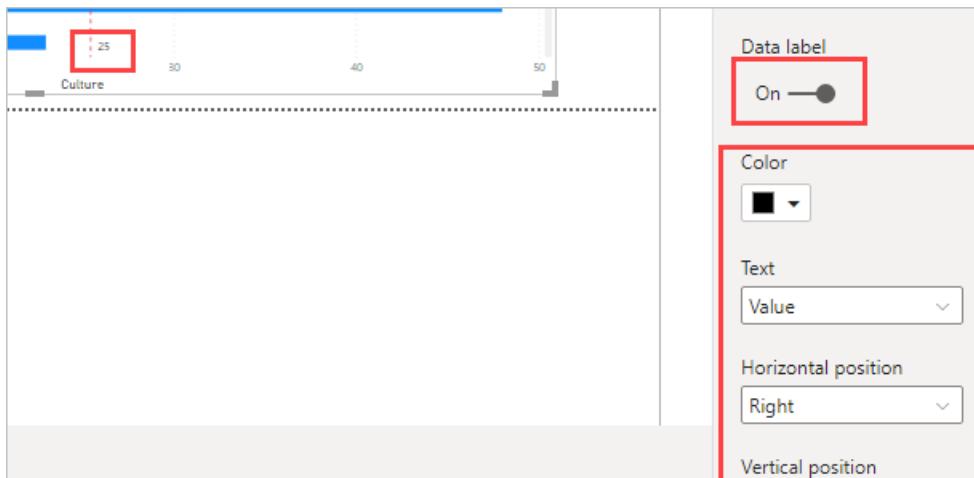


3. To create a new line, select **+ Add**. Then you can name the line. Double-click the text box and enter your name.

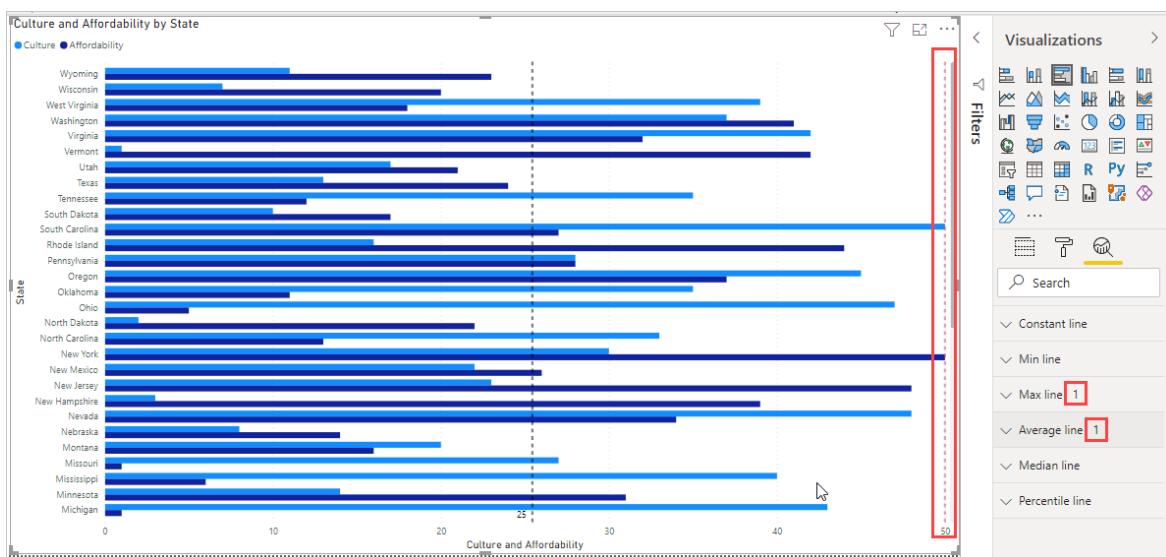
Now you have all sorts of options for your line. You can specify its **Color**, **Transparency** percentage, **Line style**, and **Position** (compared to the visual's data elements). You may also choose whether to include the **Data label**. To specify the visual measure to base your line upon, select the **Measure** dropdown list, which is automatically populated with data elements from the visual. Here we'll select **Culture** as the measure, label it *Average of Culture*, and customize a few of the other options.



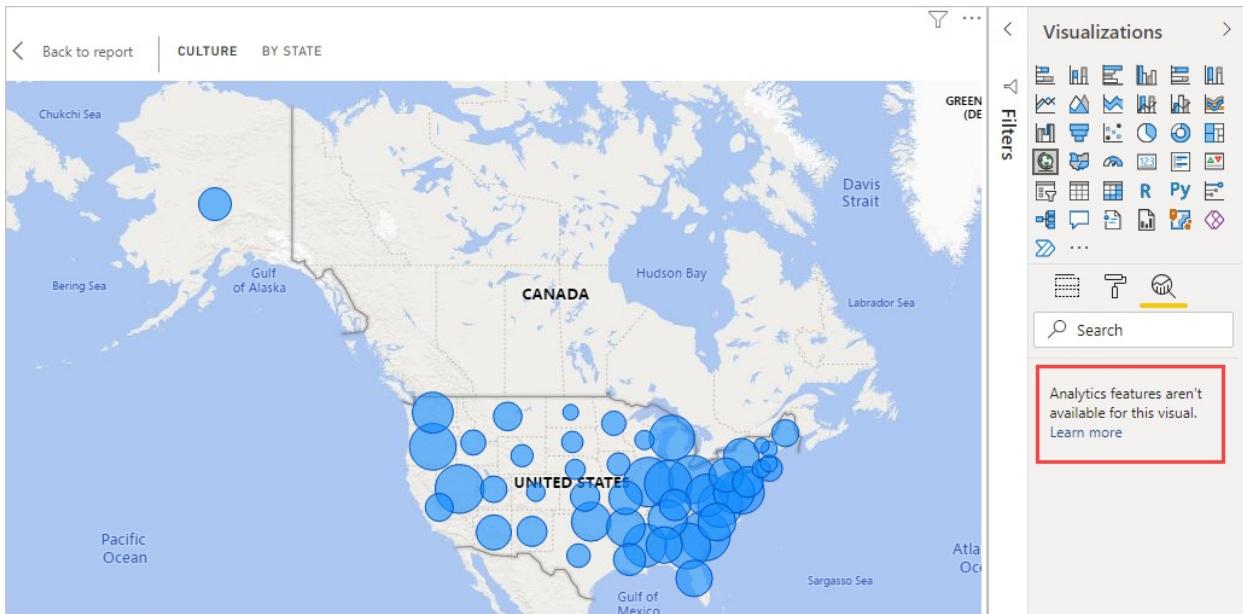
4. If you want to have a data label appear, change **Data label** from **Off** to **On**. When you do so, you get a whole host of additional options for your data label.



5. Notice the number that appears next to the **Average line** item in the **Analytics** pane. That tells you how many dynamic lines you currently have on your visual, and of which type. If we add a **Max line** for **Affordability**, the **Analytics** pane shows that we now also have a **Max line** dynamic reference line applied to this visual.



If the visual you've selected can't have dynamic reference lines applied to it (in this case, a **Map** visual), you'll see the following message when you select the **Analytics** pane.

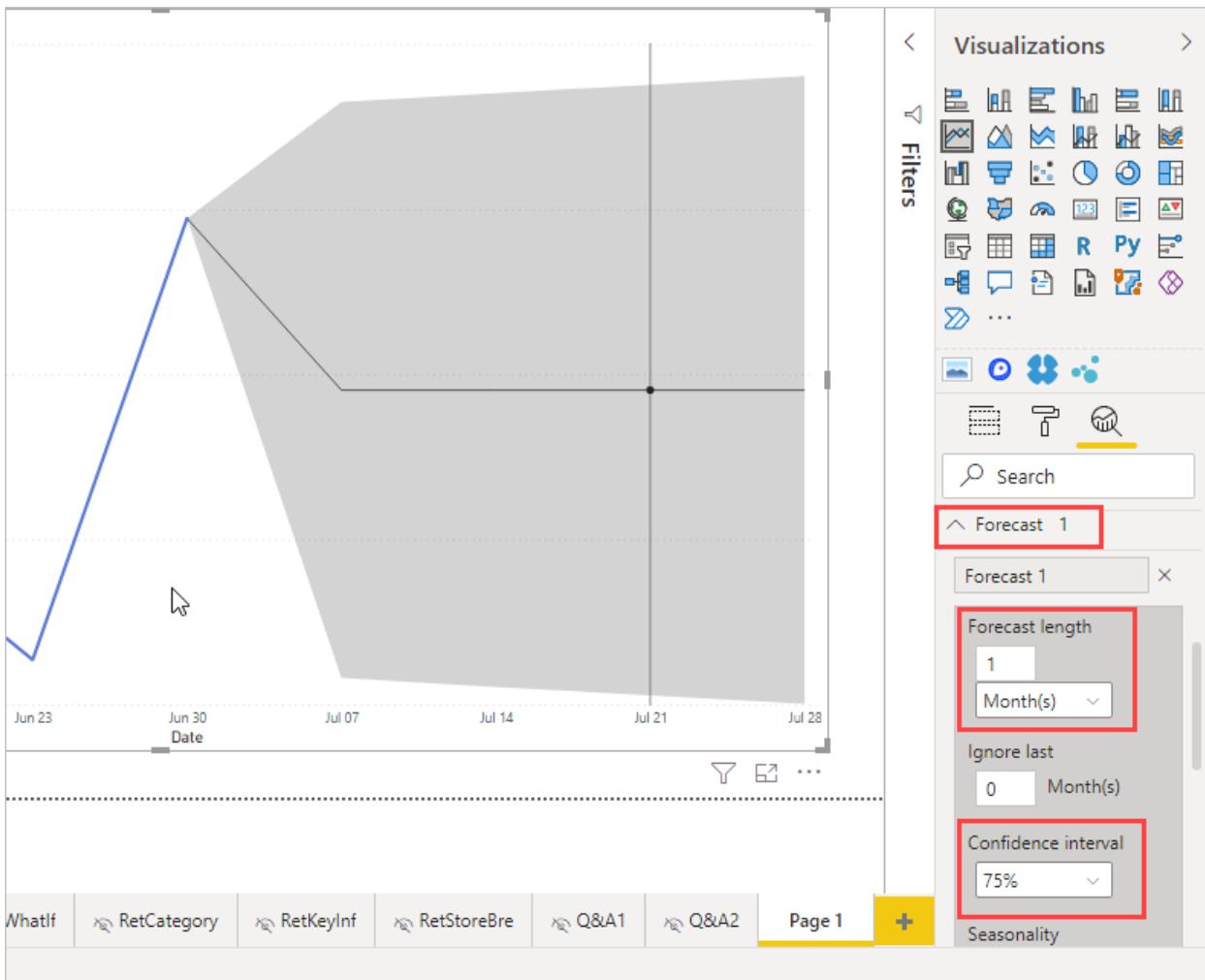


You can highlight many interesting insights by creating dynamic reference lines with the **Analytics** pane.

We're planning more features and capabilities, including expanding which visuals can have dynamic reference lines applied to them. Check back often to see what's new.

Apply forecasting

If you have time data in your data source, you can use the *forecasting* feature. Just select a visual, then expand the **Forecast** section of the **Analytics** pane. You may specify many inputs to modify the forecast, such as the **Forecast length** or the **Confidence interval**. The following image shows a basic line visual with forecasting applied. Use your imagination (and play around with forecasting) to see how it may apply to your models.



NOTE

The forecasting feature is only available for line chart visuals.

For an example of how forecasting can be applied, see the (somewhat dated, but still relevant) article about [forecasting capabilities](#).

Considerations and limitations

The ability to use dynamic reference lines is based on the type of visual being used. The following lists describe these limitations more specifically.

You may use *x-axis constant line*, *y-axis constant line*, and *symmetry shading* on the following visual:

- Scatter chart

Use of *constant line*, *min line*, *max line*, *average line*, *median line*, and *percentile line* is available on these visuals:

- Area chart
- Clustered bar chart
- Clustered column chart
- Line chart
- Scatter chart

The following visuals can use only a *constant line* from the **Analytics** pane:

- Stacked area chart

- Stacked bar chart
- Stacked column chart
- Waterfall chart
- 100% Stacked bar chart
- 100% Stacked column chart

The following visuals can use a *trend line* if there's time data:

- Area chart
- Clustered column chart
- Line chart
- Line and clustered column chart
- Scatter chart

Lastly, you can't currently apply any dynamic lines to many visuals, including (but not limited to):

- Funnel
- Line and clustered column chart
- Line and stacked column chart
- Ribbon chart
- Non-Cartesian visuals, such as Donut chart, Gauge, Matrix, Pie chart, and Table

The *percentile line* is only available when using imported data in Power BI Desktop or when connected live to a model on a server that's running **Analysis Service 2016** or later, **Azure Analysis Services**, or a dataset on the Power BI service.

Next steps

You can do all sorts of things with Power BI Desktop. For more information on its capabilities, check out the following resources:

- [What's new in Power BI?](#)
- [Get Power BI Desktop](#)
- [What is Power BI Desktop?](#)
- [Query overview with Power BI Desktop](#)
- [Data types in Power BI Desktop](#)
- [Shape and combine data with Power BI Desktop](#)
- [Perform common tasks in Power BI Desktop](#)

Work with Data view in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

Data view helps you inspect, explore, and understand data in your *Power BI Desktop* model. It's different from how you view tables, columns, and data in *Power Query Editor*. With Data view, you're looking at your data *after* it has been loaded into the model.

NOTE

Since Data view shows data after it's loaded into the model, the Data view icon is not visible if all data sources are based on DirectQuery.

When you're modeling your data, sometimes you want to see what's actually in a table or column without creating a visual on the report canvas. You might want to see right down to the row level. This ability is especially useful when you're creating measures and calculated columns, or you need to identify a data type or data category.

Let's take a closer look at some of the elements found in Data view.

The screenshot shows the Power BI Desktop interface with the 'Table tools' ribbon selected. A table named 'Sheet1' is displayed in the data grid. Several UI elements are highlighted with red circles and numbers:

- 1**: Data view icon (grid icon) located in the top-left corner of the data grid.
- 2**: Data Grid: The main area showing rows of data for 'Units Sold' (e.g., 1513, 1006, 1725) and various financial columns like Manufacturing Price, Sale Price, Gross Sales, Discounts, Sales, COGS, Profit, Date, and Month.
- 3**: Formula bar: A search bar at the top of the grid.
- 4**: Fields list: A sidebar on the right containing a search bar and a list of tables: 'financials' and 'Sheet1'.
- 5**: Fields list expanded view: Shows the contents of the 'Sheet1' table, including columns for Units Sold, Manufacturing Price, Sale Price, Gross Sales, Discounts, Sales, COGS, Profit, Date, and Month, with specific row values like 1513, 350, 529550, etc.

Table: Sheet1 (700 rows)

- 1. Data view icon.** Select this icon to enter Data view.
- 2. Data Grid.** This area shows the selected table and all columns and rows in it. Columns hidden from *Reportview* are greyed out. You can right-click on a column for options.
- 3. Formula bar.** Enter Data Analysis Expression (DAX) formulas for Measures and Calculated columns.
- 4. Search.** Search for a table or column in your model.
- 5. Fields list.** Select a table or column to view in the data grid.

Filtering in Data view

You can also filter and sort data in Data view. Each column shows an icon that identifies the sort direction, if

applied.

A screenshot of the Power BI Data view interface. On the left, there's a table with columns 'LocationID' (yellow header) and 'Sum_Markdown_Sales_Dollars'. A context menu is open over the 'LocationID' column, with several options: 'Sort ascending' (unchecked), 'Sort descending' (checked with a green checkmark), 'Clear sort', 'Clear filter', 'Clear all filters', and 'Number filters'. The 'Number filters' option is highlighted with a red box. Below it is a search bar and a list of numerical values from 2 to 14, each preceded by a yellow checkmark. To the right of the menu is a list of filtering operators: 'Equals...', 'Does not equal...', 'Greater than...', 'Greater than or equal to...', 'Less than...', 'Less than or equal to...', 'Between...', and 'Custom filter...'. At the bottom of the menu are 'OK' and 'Cancel' buttons.

You can filter individual values, or use advanced filtering based on the data in the column.

NOTE

When a Power BI model is created in a different culture than your current user interface, the search box will not appear in the Data view user interface for anything other than text fields. For example, this would apply for a model created in US English that you view in Spanish.

Next steps

You can do all sorts of things with Power BI Desktop. For more information on its capabilities, check out the following resources:

- [What is Power BI Desktop?](#)
- [Query overview with Power BI Desktop](#)
- [Data types in Power BI Desktop](#)
- [Shape and combine data with Power BI Desktop](#)
- [Common query tasks in Power BI Desktop](#)

Apply DAX basics in Power BI Desktop

3/30/2022 • 13 minutes to read • [Edit Online](#)

This article is for users new to Power BI Desktop. It gives you a quick and easy introduction on how you can use Data Analysis Expressions (DAX) to solve a number of basic calculation and data analysis problems. We'll go over some conceptual information, a series of tasks you can complete, and a knowledge check to test what you've learned. After completing this article, you should have a good understanding of the most important fundamental concepts in DAX.

What is DAX?

DAX is a collection of functions, operators, and constants that can be used in a formula, or expression, to calculate and return one or more values. Stated more simply, DAX helps you create new information from data already in your model.

Why is DAX so important?

It's easy to create a new Power BI Desktop file and import some data into it. You can even create reports that show valuable insights without using any DAX formulas at all. But, what if you need to analyze growth percentage across product categories and for different date ranges? Or, you need to calculate year-over-year growth compared to market trends? DAX formulas provide this capability and many other important capabilities as well. Learning how to create effective DAX formulas will help you get the most out of your data. When you get the information you need, you can begin to solve real business problems that affect your bottom line. This is the power of Power BI, and DAX will help you get there.

Prerequisites

You might already be familiar with creating formulas in Microsoft Excel. That knowledge will be helpful in understanding DAX, but even if you have no experience with Excel formulas, the concepts described here will help you get started creating DAX formulas and solving real-world BI problems right away.

We'll focus on understanding DAX formulas used in calculations, more specifically, in measures and calculated columns. You should already be familiar with using Power BI Desktop to import data and add fields to a report, and you should also be familiar with fundamental concepts of [Measures](#) and [Calculated columns](#).

Example workbook

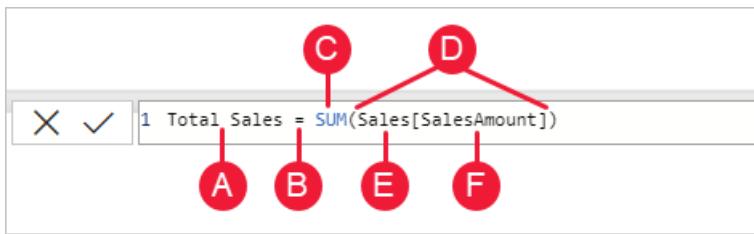
The best way to learn DAX is to create some basic formulas, use them with actual data, and see the results for yourself. The examples and tasks here use the [Contoso Sales Sample for Power BI Desktop file](#). This sample file is the same one used in the [Tutorial: Create your own measures in Power BI Desktop](#) article.

Let's begin!

We'll frame our understanding of DAX around three fundamental concepts: *Syntax*, *Functions*, and *Context*. There are other important concepts in DAX, but understanding these three concepts will provide the best foundation on which to build your DAX skills.

Syntax

Before you create your own formulas, let's take a look at DAX formula syntax. Syntax includes the various elements that make up a formula, or more simply, how the formula is written. For example, here's a simple DAX formula for a measure:



This formula includes the following syntax elements:

- A. The measure name, **Total Sales**.
- B. The equals sign operator (=), which indicates the beginning of the formula. When calculated, it will return a result.
- C. The DAX function **SUM**, which adds up all of the numbers in the **Sales[SalesAmount]** column. You'll learn more about functions later.
- D. Parenthesis (), which surround an expression that contains one or more arguments. Most functions require at least one argument. An argument passes a value to a function.
- E. The referenced table, **Sales**.
- F. The referenced column, **[SalesAmount]**, in the **Sales** table. With this argument, the **SUM** function knows on which column to aggregate a **SUM**.

When trying to understand a DAX formula, it's often helpful to break down each of the elements into a language you think and speak every day. For example, you can read this formula as:

For the measure named Total Sales, calculate (=) the SUM of values in the [SalesAmount] column in the Sales table.

When added to a report, this measure calculates and returns values by summing up sales amounts for each of the other fields we include, for example, Cell Phones in the USA.

You might be thinking, "Isn't this measure doing the same thing as if I were to just add the **SalesAmount** field to my report?" Well, yes. But, there's a good reason to create our own measure that sums up values from the **SalesAmount** field: We can use it as an argument in other formulas. This may seem a little confusing now, but as your DAX formula skills grow, knowing this measure will make your formulas and your model more efficient. In fact, you'll see the **Total Sales** measure showing up as an argument in other formulas later on.

Let's go over a few more things about this formula. In particular, we introduced a function, **SUM**. Functions are pre-written formulas that make it easier to do complex calculations and manipulations with numbers, dates, time, text, and more. You'll learn more about functions later.

You also see that the column name **[SalesAmount]** was preceded by the **Sales** table in which the column belongs. This name is known as a fully qualified column name in that it includes the column name preceded by the table name. Columns referenced in the same table don't require the table name be included in the formula, which can make long formulas that reference many columns shorter and easier to read. However, it's a good practice to include the table name in your measure formulas, even when in the same table.

NOTE

If a table name contains spaces, reserved keywords, or disallowed characters, you must enclose the table name in single quotation marks. You'll also need to enclose table names in quotation marks if the name contains any characters outside the ANSI alphanumeric character range, regardless of whether your locale supports the character set or not.

It's important your formulas have the correct syntax. In most cases, if the syntax isn't correct, a syntax error is

returned. In other cases, the syntax may be correct, but the values returned might not be what you're expecting. The DAX editor in Power BI Desktop includes a suggestions feature, used to create syntactically correct formulas by helping you select the correct elements.

Let's create a simple formula. This task will help you further understand formula syntax and how the suggestions feature in the formula bar can help you.

Task: Create a measure formula

1. [Download](#) and open the Contoso Sales Sample Power BI Desktop file.
2. In Report view, in the field list, right-click the **Sales** table, and then select **New Measure**.
3. In the formula bar, replace **Measure** by entering a new measure name, *Previous Quarter Sales*.
4. After the equals sign, type the first few letters *CAL*, and then double-click the function you want to use. In this formula, you want to use the **CALCULATE** function.

You'll use the **CALCULATE** function to filter the amounts we want to sum by an argument we pass to the **CALCULATE** function. This is referred to as nesting functions. The **CALCULATE** function has at least two arguments. The first is the expression to be evaluated, and the second is a filter.

5. After the opening parenthesis (for the **CALCULATE** function, type *SUM* followed by another opening parenthesis (.

Next, we'll pass an argument to the **SUM** function.

6. Begin typing *Sal*, and then select **Sales[SalesAmount]**, followed by a closing parenthesis).

This is the first expression argument for our **CALCULATE** function.

7. Type a comma (,) followed by a space to specify the first filter, and then type *PREVIOUSQUARTER*.

You'll use the **PREVIOUSQUARTER** time intelligence function to filter **SUM** results by the previous quarter.

8. After the opening parenthesis (for the **PREVIOUSQUARTER** function, type *Calendar[DateKey]*.

The **PREVIOUSQUARTER** function has one argument, a column containing a contiguous range of dates. In our case, that's the **DateKey** column in the **Calendar** table.

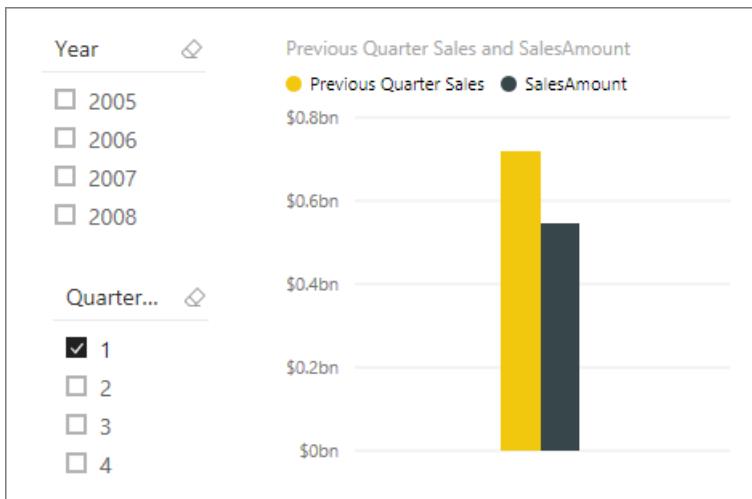
9. Close both the arguments being passed to the **PREVIOUSQUARTER** function and the **CALCULATE** function by typing two closing parenthesis)).

Your formula should now look like this:

**Previous Quarter Sales = CALCULATE(SUM(Sales[SalesAmount]),
PREVIOUSQUARTER(Calendar[DateKey]))**

10. Select the checkmark ✓ in the formula bar or press Enter to validate the formula and add it to the model.

You did it! You just created a complex measure by using DAX, and not an easy one at that. What this formula will do is calculate the total sales for the previous quarter, depending on the filters applied in a report. For example, if we put **SalesAmount** and our new **Previous Quarter Sales** measure in a chart, and then add **Year** and **QuarterOfYear** as Slicers, we'd get something like this:



You were just introduced to several important aspects of DAX formulas:

- This formula included two functions. `PREVIOUSQUARTER`, a time intelligence function, is nested as an argument passed to `CALCULATE`, a filter function.
DAX formulas can contain up to 64 nested functions. It's unlikely a formula would ever contain so many nested functions. In fact, such a formula would be difficult to create and debug, and it probably wouldn't be very fast either.
- In this formula, you also used filters. Filters narrow down what will be calculated. In this case, you selected one filter as an argument, which is actually the result of another function. You will learn more about filters later.
- You used the `CALCULATE` function. This function is one of the most powerful functions in DAX. As you author models and create more complex formulas, you'll likely use this function many times. Although further discussion about the `CALCULATE` function is outside the scope of this article, as your knowledge of DAX grows, pay special attention to it.

Syntax QuickQuiz

1. What does this button on the formula bar do?



2. What always surrounds a column name in a DAX formula?

Answers are provided at the end of this article.

Functions

Functions are predefined formulas that perform calculations by using specific values, called arguments, in a particular order or structure. Arguments can be other functions, another formula, expression, column references, numbers, text, logical values such as TRUE or FALSE, or constants.

DAX includes the following categories of functions: [Date and Time](#), [Time Intelligence](#), [Information](#), [Logical](#), [Mathematical](#), [Statistical](#), [Text](#), [Parent/Child](#), and [Other](#) functions. If you're familiar with functions in Excel formulas, many of the functions in DAX will appear similar to you; however, DAX functions are unique in the following ways:

- A DAX function always references a complete column or a table. If you want to use only particular values from a table or column, you can add filters to the formula.
- If you need to customize calculations on a row-by-row basis, DAX provides functions that let you use the current row value or a related value as a kind of argument, to perform calculations that vary by context.

You'll learn more about context later.

- DAX includes many functions that return a table rather than a value. The table isn't displayed, but is used to provide input to other functions. For example, you can retrieve a table and then count the distinct values in it, or calculate dynamic sums across filtered tables or columns.
- DAX includes a variety of time intelligence functions. These functions let you define or select date ranges, and perform dynamic calculations based on them. For example, you can compare sums across parallel periods.
- Excel has a popular function, VLOOKUP. DAX functions don't take a cell or cell range as a reference like VLOOKUP does in Excel. DAX functions take a column or a table as a reference. Keep in mind, in Power BI Desktop you're working with a relational data model. Looking up values in another table is easy, and in most cases you don't need to create any formulas at all.

As you can see, functions in DAX can help you create powerful formulas. We really only touched on the basics of functions. As your DAX skills grow, you'll create formulas by using many different functions. One of the best places to learn details about each of the DAX functions is in the [DAX Function Reference](#).

Functions QuickQuiz

1. What does a function always reference?
2. Can a formula contain more than one function?
3. What category of functions would you use to concatenate two text strings into one string?

Answers are provided at the end of this article.

Context

Context is one of the most important DAX concepts to understand. There are two types of context in DAX: row context and filter context. We'll first look at row context.

Row context

Row context is most easily thought of as the current row. It applies whenever a formula has a function that applies filters to identify a single row in a table. The function will inherently apply a row context for each row of the table over which it is filtering. This type of row context most often applies to measures.

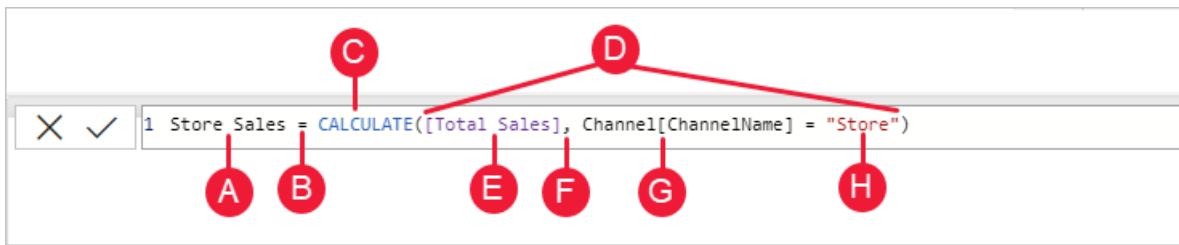
Filter context

Filter context is a little more difficult to understand than row context. You can most easily think of filter context as: One or more filters applied in a calculation that determines a result or value.

Filter context doesn't exist in place of row context; rather, it applies in addition to row context. For example, to further narrow down the values to include in a calculation, you can apply a filter context, which not only specifies the row context, but also specifies a particular value (filter) in that row context.

Filter context is easily seen in your reports. For example, when you add TotalCost to a visualization, and then add Year and Region, you are defining a filter context that selects a subset of data based on a given year and region.

Why is filter context so important to DAX? Because while filter context can most easily be applied by adding fields to a visualization, filter context can also be applied in a DAX formula by defining a filter using functions such as ALL, RELATED, FILTER, CALCULATE, by relationships, and by other measures and columns. For example, let's look at the following formula in a measure named Store Sales:



To better understand this formula, we can break it down, much like with other formulas.

This formula includes the following syntax elements:

- A. The measure name, **Store Sales**.
- B. The equals sign operator (=), which indicates the beginning of the formula.
- C. The **CALCULATE** function, which evaluates an expression, as an argument, in a context that is modified by the specified filters.
- D. Parenthesis (), which surround an expression containing one or more arguments.
- E. A measure [**Total Sales**] in the same table as an expression. The Total Sales measure has the formula: =SUM(Sales[SalesAmount]).
- F. A comma (,), which separates the first expression argument from the filter argument.
- G. The fully qualified referenced column, **Channel[ChannelName]**. This is our row context. Each row in this column specifies a channel, such as Store or Online.
- H. The particular value, **Store**, as a filter. This is our filter context.

This formula ensures only sales values defined by the Total Sales measure are calculated only for rows in the Channel[ChannelName] column, with the value *Store* used as a filter.

As you can imagine, being able to define filter context within a formula has immense and powerful capabilities. The ability to reference only a particular value in a related table is just one such example. Don't worry if you do not completely understand context right away. As you create your own formulas, you will better understand context and why it's so important in DAX.

Context QuickQuiz

1. What are the two types of context?
2. What is filter context?
3. What is row context?

Answers are provided at the end of this article.

Summary

Now that you have a basic understanding of the most important concepts in DAX, you can begin creating DAX formulas for measures on your own. DAX can indeed be a little tricky to learn, but there are many resources available to you. After reading through this article and experimenting with a few of your own formulas, you can learn more about other DAX concepts and formulas that can help you solve your own business problems. There are many DAX resources available to you; most important is the [Data Analysis Expressions \(DAX\) Reference](#).

Because DAX has been around for several years in other Microsoft BI tools such as Power Pivot and Analysis Services Tabular models, there's a lot of great information out there. You can find more information in books, whitepapers, and blogs from both Microsoft and leading BI professionals. The [DAX Resource Center Wiki on TechNet](#) is also a great place to start.

QuickQuiz answers

Syntax:

1. Validates and enters the measure into the model.
2. Brackets [].

Functions:

1. A table and a column.
2. Yes. A formula can contain up to 64 nested functions.
3. [Text functions](#).

Context:

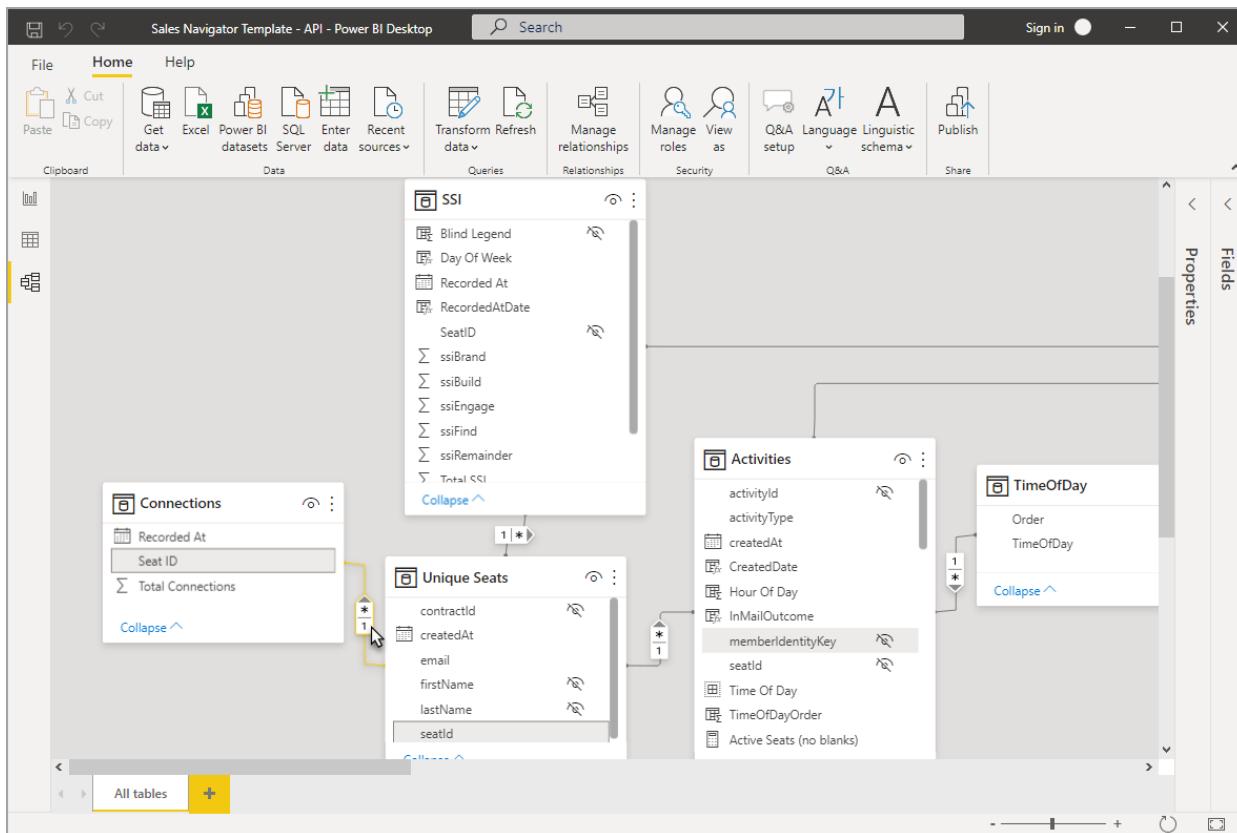
1. Row context and filter context.
2. One or more filters in a calculation that determines a single value.
3. The current row.

Work with Model view in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

Model view shows all of the tables, columns, and relationships in your model. This view can be especially helpful when your model has complex relationships between many tables.

Select the **Model** icon near the side of the window to see a view of the existing model. Hover your cursor over a relationship line to show the columns that are used.

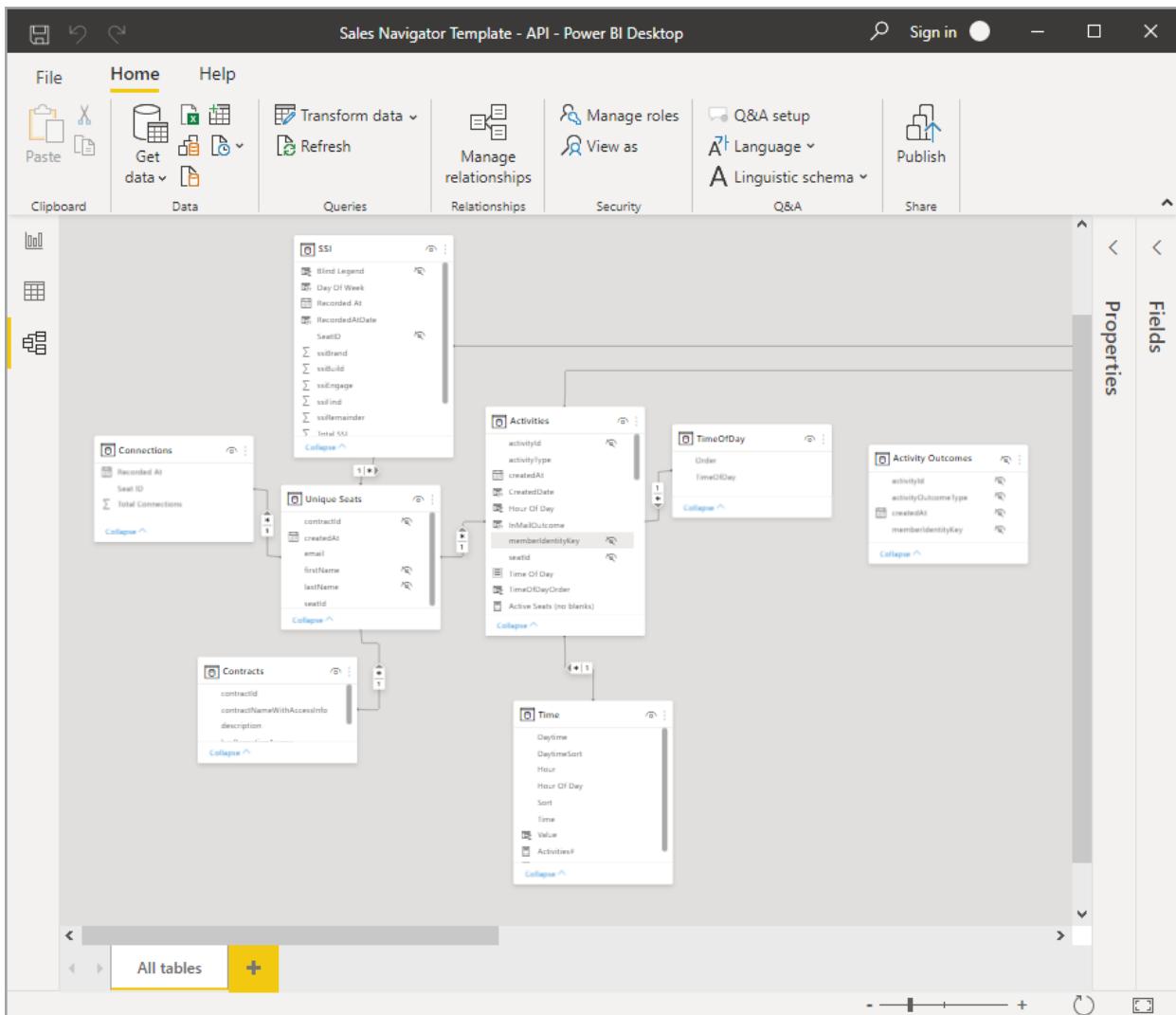


In the image, the *Connections* table has a *Seat ID* column that's related to the *Unique Seats* table, which also has a *seatId* column. The two tables have a *Many to One* (*:1) relationship. An arrow in the middle of the line shows the direction of the filter context flow. Double arrows would mean the cross-filter direction is set to *Both*.

You can double-click a relationship to open it in the **Edit Relationship** dialog box. To learn more about relationships, see [Create and manage relationships in Power BI Desktop](#).

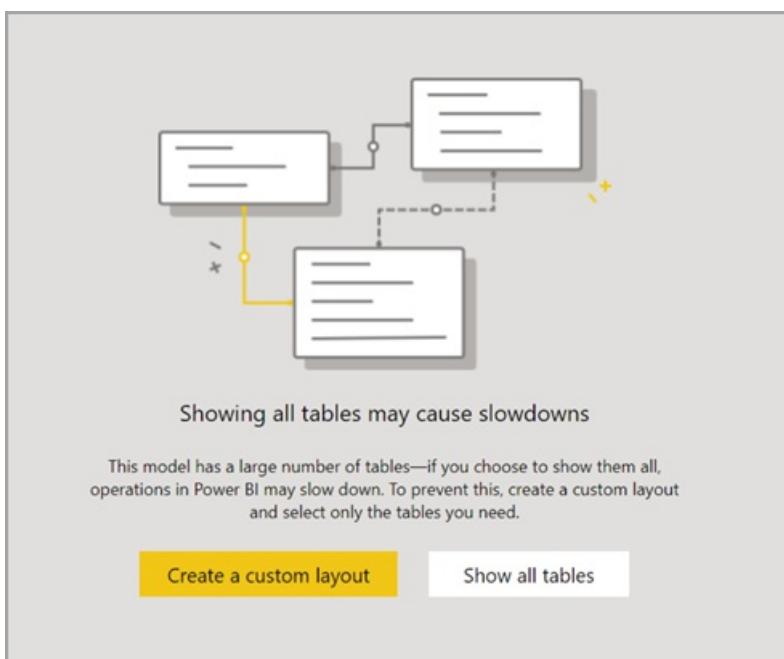
Updated Model View

Current releases of Power BI Desktop have the updated **Model view** enabled.



The colors in the table card headers will automatically match the colors you've selected in any report theme you're using. If the color is too close to white, it won't be used in the Model view theme headers to avoid situations where it's difficult to differentiate tables in dual mode.

If your model has less than 75 tables, all tables will be shown in Model view. If your model has more than 75 tables, instead of showing all tables you see the following image:



We recommend you create a custom layout when your model has more than 75 tables (select the *Create a*

custom layout button) to reduce the significant CPU and memory used when more than 75 tables are displayed.

Next steps

There are all sorts of things you can do with Power BI Desktop. For more information on data sources, check out the following resources:

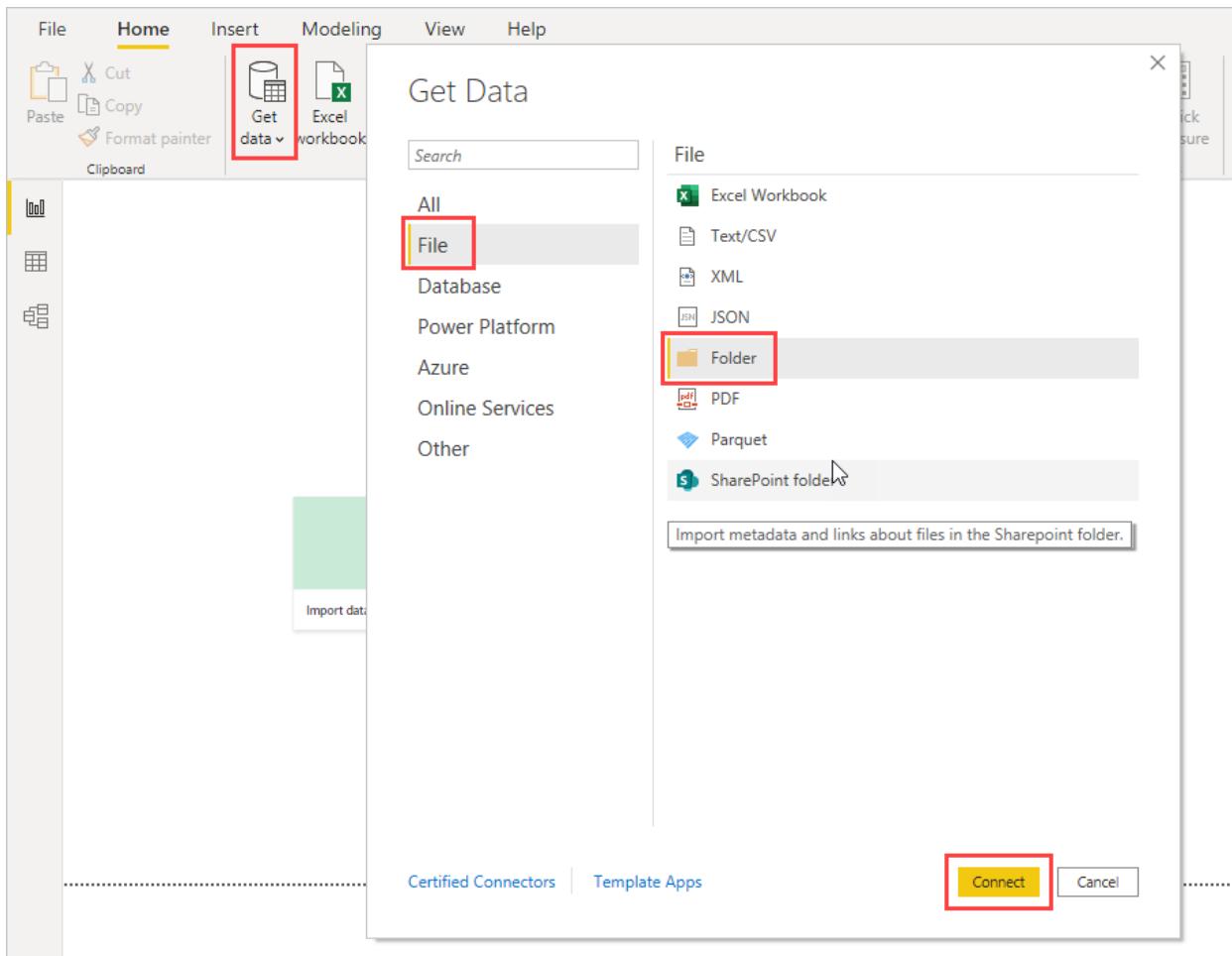
- [What is Power BI Desktop?](#)
- [Data Sources in Power BI Desktop](#)
- [Shape and Combine Data with Power BI Desktop](#)
- [Connect to Excel workbooks in Power BI Desktop](#)
- [Enter data directly into Power BI Desktop](#)

Combine files (binaries) in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

Here's a powerful approach to importing data into **Power BI Desktop**: If you have multiple files that have the same schema, combine them into a single logical table. This popular technique has been made more convenient and more expansive.

To start the process of combining files from the same folder, select **Get data**, choose **File > Folder**, and then select **Connect**.



Enter the folder path, select **OK**, and then select **Transform data** to see the folder's files in Power Query Editor.

Combine files behavior

To combine binary files in Power Query Editor, select **Content** (the first column label) and select **Home > Combine Files**. Or you can just select the **Combine Files** icon next to **Content**.

The screenshot shows the Power Query Editor interface. The 'Home' tab is selected. In the center, there's a table view titled '= Folder.Files("C:\Statistics")' with columns 'Content', 'Name', and 'Extension'. The first row, 'Content', is highlighted with a red box. A yellow box highlights the 'Content' column header. In the top right, the 'Transform' ribbon has a 'Combine' button highlighted with a red box. On the right side, there's a 'Query Settings' pane with sections for 'PROPERTIES' (Name: Statistics) and 'APPLIED STEPS' (Source).

The *combine files* transform behaves as follows:

- The combine files transform analyzes each input file to determine the correct file format to use, such as *text*, *Excel workbook*, or *JSON file*.
- The transform allows you to select a specific object from the first file, such as an Excel workbook, to extract.

The screenshot shows the 'Combine Files' dialog box. At the top, it says 'Combine Files'. Below that, it says 'Select the object to be extracted from each file. [Learn more](#)'. A dropdown menu 'Sample File:' is set to 'First file'. To the right, there's a preview area titled 'ABCDE' showing a table of financial data. At the bottom left, there's a 'Display Options' section with a tree view showing 'Parameter1 [1]' expanded, with 'ABCDE' selected. At the bottom right, there are 'OK' and 'Cancel' buttons.

Symbol	Date	Open	High	Low	Close	Ad
ABCDE	1/11/2019	348.559998	354.359985	348.089996	352.899994	
ABCDE	1/14/2019	348.200012	352.809998	347.01001	350.359985	
ABCDE	1/15/2019	352	353.320007	347.980011	352.23999	
ABCDE	1/16/2019	352.5	355	351.559998	352.059998	
ABCDE	1/17/2019	350.75	363.829987	350.730011	359.089996	
ABCDE	1/18/2019	363.880005	367.320007	361.320007	364.730011	
ABCDE	1/22/2019	362.859985	364.200012	354.230011	357.899994	
ABCDE	1/23/2019	361.910004	362.200012	353.670013	358.609985	
ABCDE	1/24/2019	358.910004	363.179993	356.899994	358.269989	
ABCDE	1/25/2019	362.48999	366.940002	360.329987	364.200012	
ABCDE	1/28/2019	360.649994	363.170013	357.5	362.970001	
ABCDE	1/29/2019	363.070007	367.779999	362.410004	364.910004	
ABCDE	1/30/2019	387.399994	391.970001	380.5	387.720001	
ABCDE	1/31/2019	387.160004	388.98999	382.079987	385.619995	
ABCDE	2/1/2019	386.109985	392.799988	384.730011	387.429993	
ABCDE	2/4/2019	388.970001	397.070007	388.119995	397	
ABCDE	2/5/2019	400.75	410.75	399.549988	410.179993	
ABCDE	2/6/2019	411.51001	413.880005	405.660004	411.109985	
ABCDE	2/7/2019	407.929993	410.350006	402.290009	405.170013	
ABCDE	2/8/2019	400	404.959991	397.799988	404.910004	

- The combine files transform then automatically takes these actions:
 - Creates an example query that performs all the required extraction steps in a single file.
 - Creates a *function query* that parameterizes the file/binary input to the *exemplar query*. The exemplar query and the function query are linked, so that changes to the exemplar query are reflected in the function query.
 - Applies the *function query* to the original query with input binaries, such as the *Folder* query. It applies the function query for binary inputs on each row, then expands the resulting data extraction as top-level columns.

NOTE

The scope of your selection in an Excel workbook will affect the behavior of combine binaries. For example, you can select a specific worksheet to combine that worksheet, or select the root to combine the full file. Selecting a folder combines the files found in that folder.

With the behavior of combine files, you can easily combine all files within a given folder if they have the same file type and structure (such as the same columns).

In addition, you can easily apply additional transformation or extraction steps by modifying the automatically created exemplar query, without having to worry about modifying or creating additional function query steps. Any changes to the exemplar query are automatically generated in the linked function query.

Next steps

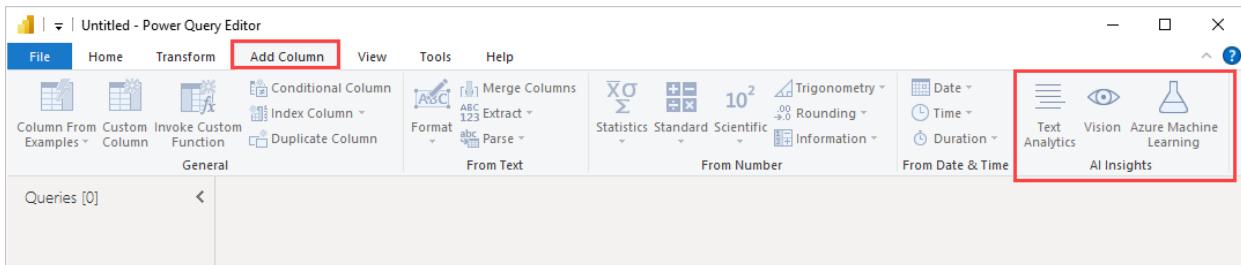
You can connect to all sorts of data using Power BI Desktop. For more information on data sources, see the following resources:

- [What is Power BI Desktop?](#)
- [Data sources in Power BI Desktop](#)
- [Shape and combine data with Power BI Desktop](#)
- [Connect to CSV files in Power BI Desktop](#)
- [Enter data directly into Power BI Desktop](#)

Use AI Insights in Power BI Desktop

3/30/2022 • 12 minutes to read • [Edit Online](#)

In Power BI, you can use AI Insights to gain access to a collection of pre-trained machine learning models that enhance your data preparation efforts. AI Insights is accessed in the **Power Query Editor**, and its associated features and functions are accessed through the **Home** and **Add Column** tabs in **Power Query Editor**.



This article describes functions for Text Analytics and Vision functions, both from Azure Cognitive Services. Also in this article is a section that describes the custom functions available in Power BI from Azure Machine Learning.

Using Text Analytics and Vision

With Text Analytics and Vision in Power BI, you can apply different algorithms from [Azure Cognitive Services](#) to enrich your data in Power Query.

The services that are supported today are the following:

- [Sentiment Analysis](#)
- [Key Phrase Extraction](#)
- [Language Detection](#)
- [Image Tagging](#).

The transformations are executed on the Power BI service and do not require an Azure Cognitive Services subscription.

IMPORTANT

Using the Text Analytics or Vision features requires Power BI Premium.

Enabling Text Analytics and Vision on Premium capacities

Cognitive Services are supported for Premium capacity nodes EM2, A2, or P1 and above. Cognitive services are also available with a Premium Per User (PPU) license. A separate AI workload on the capacity is used to run Cognitive Services. Before using Cognitive Services in Power BI, the AI workload must be enabled in the **capacity settings** of the admin portal. You can turn on the **AI workload** in the **workloads** section and define the maximum amount of memory you would like this workload to consume. The recommended memory limit is 20%. Exceeding this limit causes the query to slow down.

Available functions

This section describes the available functions in Cognitive Services in Power BI.

Detect language

The language detection function evaluates text input, and for each field, returns the language name and ISO

identifier. This function is useful for data columns that collect arbitrary text, where language is unknown. The function expects data in text format as input.

Text Analytics recognizes up to 120 languages. For more information, see [supported languages](#).

Extract key phrases

The **Key Phrase Extraction** function evaluates unstructured text, and for each text field, returns a list of key phrases. The function requires a text field as input, and accepts an optional input for **Culture info**.

Key phrase extraction works best when you give it bigger chunks of text to work on. This is opposite from sentiment analysis, which performs better on smaller blocks of text. To get the best results from both operations, consider restructuring the inputs accordingly.

Score sentiment

The **Score Sentiment** function evaluates text input and returns a sentiment score for each document, ranging from 0 (negative) to 1 (positive). This function is useful for detecting positive and negative sentiment in social media, customer reviews, and discussion forums.

Text Analytics uses a machine learning classification algorithm to generate a sentiment score between 0 and 1. Scores closer to 1 indicate positive sentiment, scores closer to 0 indicate negative sentiment. The model is pre-trained with an extensive body of text with sentiment associations. Currently, it's not possible to provide your own training data. The model uses a combination of techniques during text analysis, including text processing, part-of-speech analysis, word placement, and word associations. For more information about the algorithm, see [Introducing Text Analytics](#).

Sentiment analysis is performed on the entire input field, as opposed to extracting sentiment for a particular entity in the text. In practice, there's a tendency for scoring accuracy to improve when documents contain one or two sentences rather than a large block of text. During an objectivity assessment phase, the model determines whether an input field as a whole is objective or contains sentiment. An input field that is mostly objective does not progress to the sentiment detection phase, resulting in a .50 score, with no further processing. For input fields continuing in the pipeline, the next phase generates a score above or below .50, depending on the degree of sentiment detected in the input field.

Currently, Sentiment Analysis supports English, German, Spanish, and French. Other languages are in preview. For more information, see [supported languages](#).

Tag images

The **Tag Images** function returns tags based on more than two thousand recognizable objects, living beings, scenery, and actions. When tags are ambiguous or not common knowledge, the output provides *hints* to clarify the meaning of the tag in context of a known setting. Tags are not organized as a taxonomy and no inheritance hierarchies exist. A collection of content tags forms the foundation for an image *description* displayed as human readable language formatted in complete sentences.

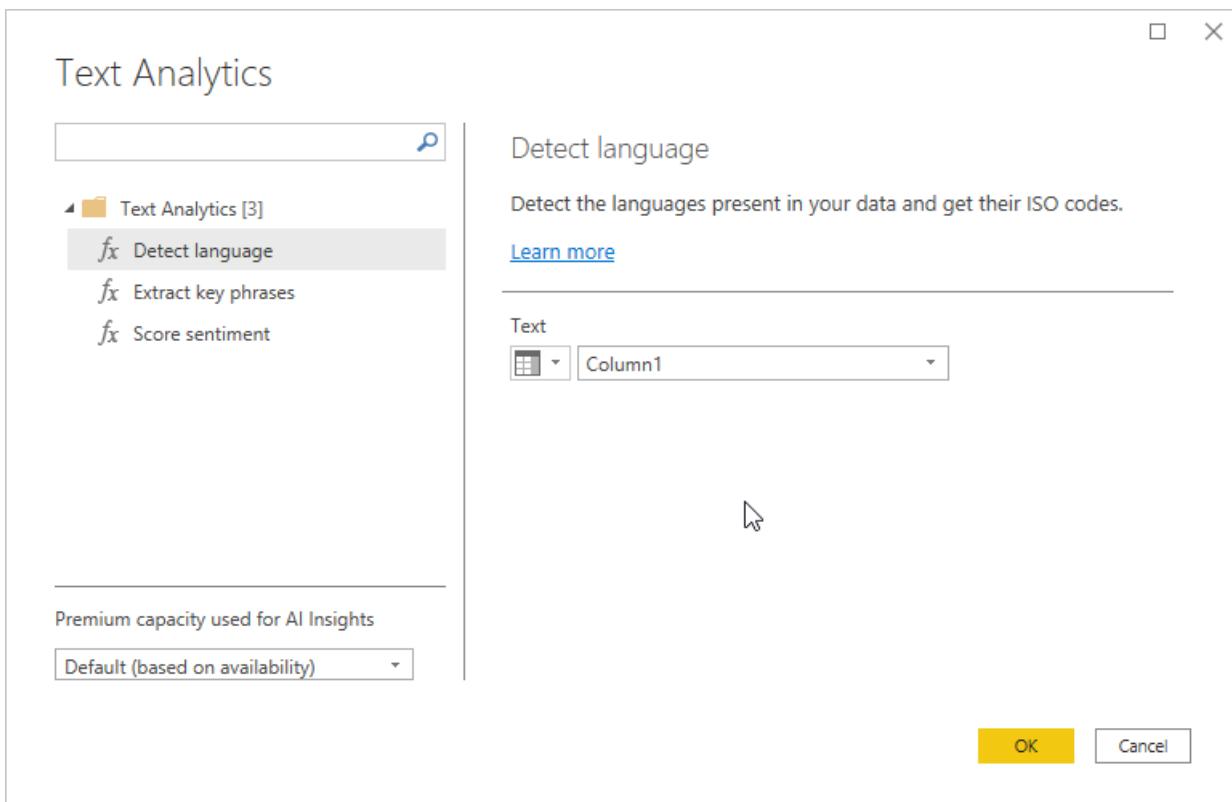
After uploading an image or specifying an image URL, Computer Vision algorithms output tags based on the objects, living beings, and actions identified in the image. Tagging is not limited to the main subject, such as a person in the foreground, but also includes the setting (indoor or outdoor), furniture, tools, plants, animals, accessories, gadgets, and so on.

This function requires an image URL or a base-64 field as input. At this time, image tagging supports English, Spanish, Japanese, Portuguese, and Simplified Chinese. For more information, see [supported languages](#).

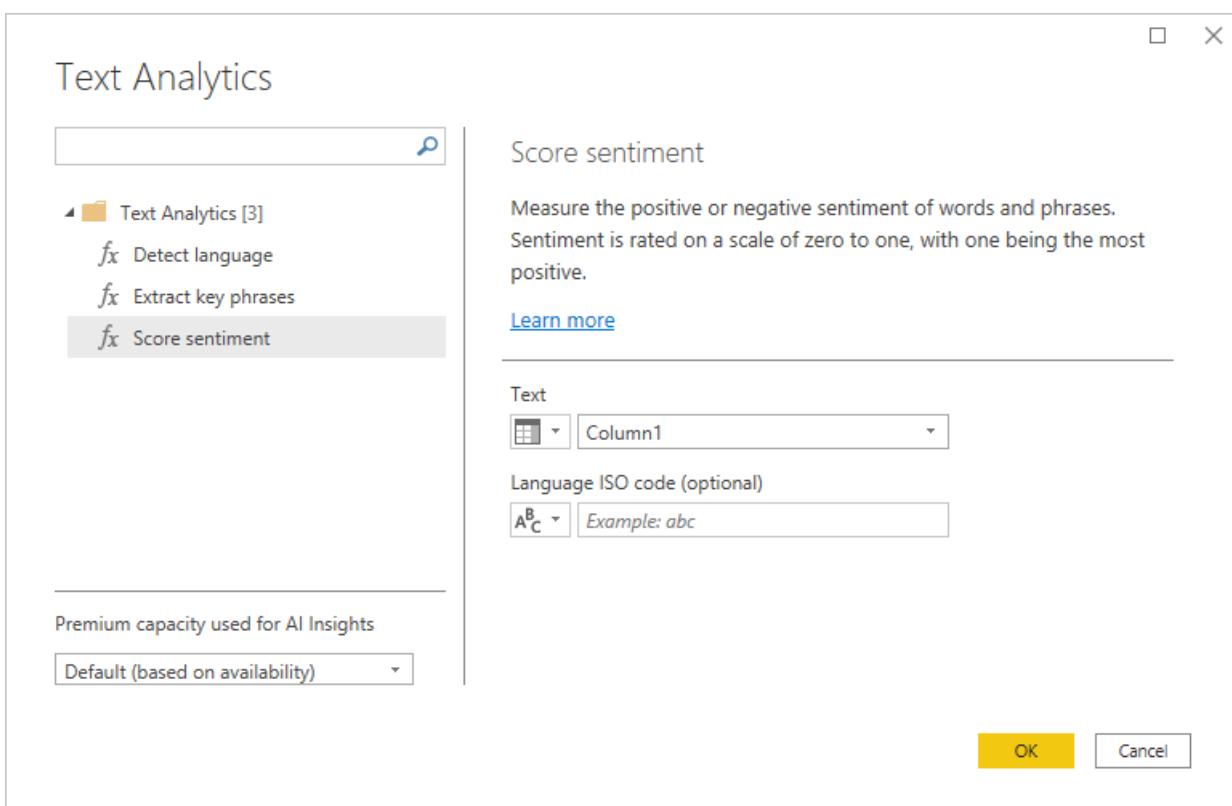
Invoking Text Analytics or Vision functions in Power Query

To enrich your data with Text Analytics or Vision functions, open **Power Query Editor**. This example walks through scoring the sentiment of a text. The same steps can be used to extract key phrases, detect language, and tag images.

Select the **Text analytics** button in the **Home** or **Add Column** ribbon. You'll be prompted to sign in.



After signing in, select the function you want to use and the data column you want to transform in the pop-up window.

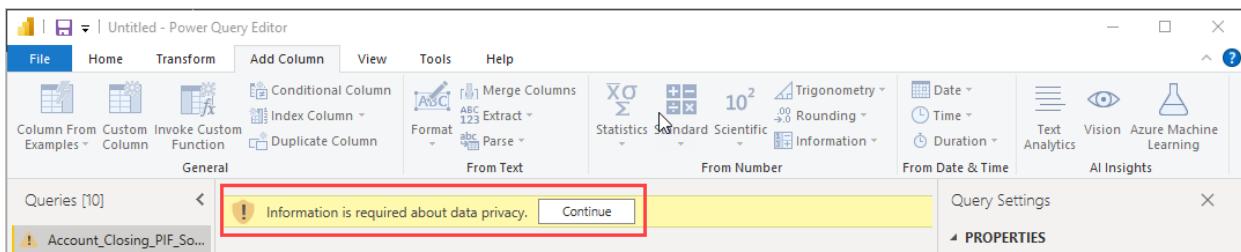


Power BI selects a Premium capacity to run the function on and send the results back to Power BI Desktop. The selected capacity is only used for Text Analytics and Vision function during application and refreshes in Power BI Desktop. Once the report is published, refreshes run on the Premium capacity of the workspace the report is published to. You can change the capacity used for all Cognitive Services in the dropdown in the lower left corner of the popup window.



Cultureinfo is an optional input to specify the language of the text. This field is an ISO code. You can use a column as input for Cultureinfo, or a static field. In this example, the language is specified as English (en) for the whole column. If you leave this field blank, Power BI automatically detects the language before applying the function. Next, select **Apply**.

The first time you use AI Insights on a new data source, you're prompted to set the privacy level of your data.



NOTE

Refreshes of the dataset in Power BI will only work for data sources where the privacy level is set to public or organizational.

After invoking the function, the result is added as a new column to the table. The transformation is also added as an applied step in the query.

In the cases of image tagging and key phrase extraction, the results can return multiple values. Each individual result is returned on a duplicate of the original row.

Publishing a report with Text Analytics or Vision functions

While editing in Power Query and performing refreshes in Power BI Desktop, Text Analytics and Vision use the Premium capacity that was selected in Power Query Editor. After publishing the report to Power BI, it uses the Premium capacity of the workspace into which it was published.

Reports with applied Text Analytics and Vision functions should be published to a workspace that is not on a Premium capacity, otherwise refreshing the dataset will fail.

Managing impact on a Premium capacity

The following sections describe how you can manage the impacts of Text Analytics and Vision on capacity.

Selecting a capacity

Report authors can select which Premium capacity on which to run AI Insights. By default, Power BI selects the first created capacity to which the user has access.

Monitoring with the Capacity Metrics app

Premium capacity owners can monitor the impact of Text Analytics and Vision functions on a capacity with the [Power BI Premium Capacity Metrics app](#). The app provides detailed metrics on the health of the AI workloads within your capacity. The top chart shows the memory consumption by AI workloads. Premium capacity admins can set the memory limit for the AI workload per capacity. When memory usage reaches the memory limit, you can consider increasing the memory limit or moving some workspaces to a different capacity.

Comparing Power Query and Power Query Online

The Text Analytics and Vision functions used in Power Query and Power Query Online are the same. The only

differences between the experiences are the following:

- Power Query has separate buttons for Text Analytics, Vision, and Azure Machine Learning. In Power Query Online, these are combined in one menu.
- In Power Query, the report author can select the Premium capacity that is used to run the functions. This is not required in Power Query Online, since a dataflow is already on a specific capacity.

Considerations and limitations of Text Analytics

There are a few considerations and limitations to keep in mind when using Text Analytics.

- Incremental refresh is supported but can cause performance issues when used on queries with AI insights.
- Direct Query is not supported.

Using Azure ML

Numerous organizations use **Machine Learning** models for better insights and predictions about their business. The ability to visualize and invoke insights from these models, in your reports and dashboards and other analytics, can help disseminate these insights to the business users who need it the most. Power BI makes it simple to incorporate the insights from models hosted on Azure Machine Learning, using straightforward point-and-click gestures.

To use this capability, a data scientist can simply grant access to the Azure ML model to the BI analyst using the Azure portal. Then, at the start of each session, Power Query discovers all the Azure ML models to which the user has access and exposes them as dynamic Power Query functions. The user can then invoke those functions by accessing them from the ribbon in Power Query Editor, or by invoking the M function directly. Power BI also automatically batches the access requests when invoking the Azure ML model for a set of rows to achieve better performance.

This functionality is supported in Power BI Desktop, Power BI dataflows, and for Power Query Online in the Power BI service.

To learn more about dataflows, see [Self-service data prep in Power BI](#).

To learn more about Azure Machine Learning, see the following articles:

- Overview: [What is Azure Machine Learning?](#)
- Quick Starts and Tutorials for Azure Machine Learning: [Azure Machine Learning Documentation](#)

Granting access to an Azure ML model

To access an Azure ML model from Power BI, the user must have **Read** access to the Azure subscription. In addition, they must also have **Read** access to the Machine Learning workspace.

The steps in this section describe how to grant a Power BI user access to a model hosted on the Azure ML service, so they can access this model as a Power Query function. For further details, please see [Manage access using RBAC and the Azure portal](#).

1. Sign in to the [Azure portal](#).
2. Go to the **Subscriptions** page. You can find the **Subscriptions** page through the **All Services** list in the left navigation menu of the Azure portal.
3. Select your subscription
4. Select **Access control (IAM)**, and then select the **Add** button.
5. Select **Reader** as the Role. Select the Power BI user to whom you wish to grant access to the Azure ML model.
6. Select **Save**
7. Repeat steps three through six to grant **Reader** access to the user for the specific Machine Learning workspace hosting the model.

Schema discovery for Machine Learning models

Data scientists primarily use Python to develop, and even deploy, their machine learning models for Machine Learning. The data scientist must explicitly generate the schema file using Python.

This schema file must be included in the deployed web service for Machine Learning models. To automatically generate the schema for web service, you must provide a sample of the input/output in the entry script for the deployed model. Please see the subsection on [\(Optional\) Automatic Swagger schema generation in the Deploy models with the Azure Machine Learning](#) service documentation. The link includes the example entry script with the statements for the schema generation.

Specifically, the `@input_schema` and `@output_schema` functions in the entry script reference the input and output sample formats in the `input_sample` and `output_sample` variables, and use these samples to generate an OpenAPI (Swagger) specification for the web service during deployment.

These instructions for schema generation by updating the entry script must also be applied to models created using automated machine learning experiments using the Azure Machine Learning SDK.

NOTE

Models created using the Azure Machine Learning visual interface do not currently support schema generation, but will in subsequent releases.

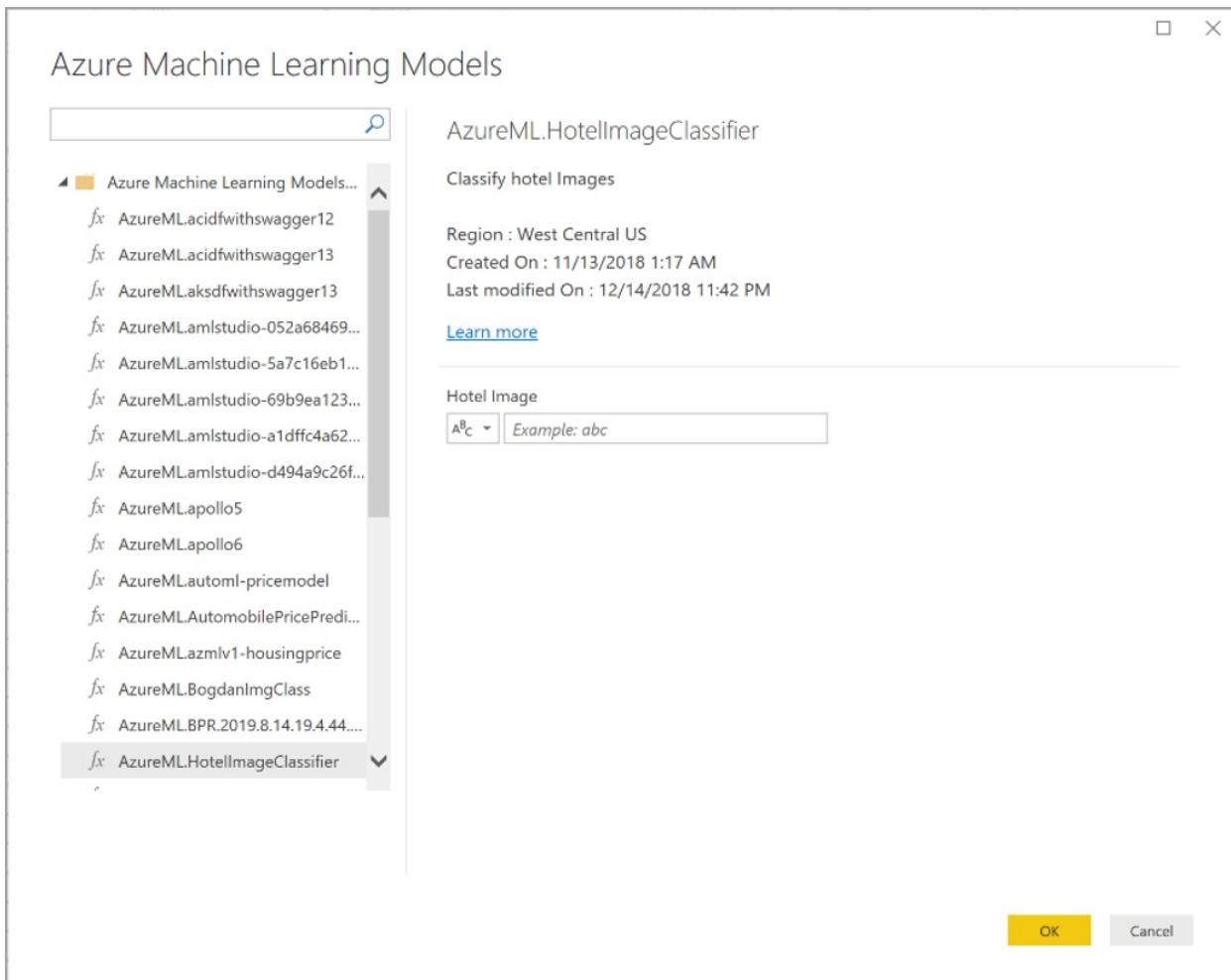
Invoking an Azure ML model in Power Query

You can invoke any Azure ML model to which you have been granted access, directly from the Power Query Editor. To access the Azure ML models, select **Azure Machine Learning** button in the **Home** or **Add Column** ribbon in the Power Query Editor.



All Azure ML models to which you have access are listed here as Power Query functions. Also, the input parameters for the Azure ML model are automatically mapped as parameters of the corresponding Power Query function.

To invoke an Azure ML model, you can specify any of the selected entity's columns as an input from the drop-down. You can also specify a constant value to be used as an input by toggling the column icon to the left of the input dialog.



Select **OK** to view the preview of the Azure ML model's output as a new column in the entity table. You will also see the model invocation as an applied step for the query.

If the model returns multiple output parameters, they are grouped together as a record in the output column. You can expand the column to produce individual output parameters in separate columns.

Considerations and limitations of Azure ML

The following considerations and limitations apply to Azure ML in Power BI Desktop.

- Models created using the Azure Machine Learning visual interface do not currently support schema generation. Support is anticipated in subsequent releases.
- Incremental refresh is supported but can cause performance issues when used on queries with AI insights.
- Direct Query is not supported.

Next steps

This article provided an overview of integrating Machine Learning into Power BI Desktop. The following articles might also be interesting and useful.

- [Tutorial: Consume Azure Machine Learning models in Power BI](#)
- [Tutorial: Using Cognitive Services in Power BI](#)
- [Cognitive Services in Power BI](#)
- [Azure Machine Learning integration in Power BI](#)
- [Monitoring Premium capacities with the app](#)
- [AI metrics in the Premium capacity metrics app](#)

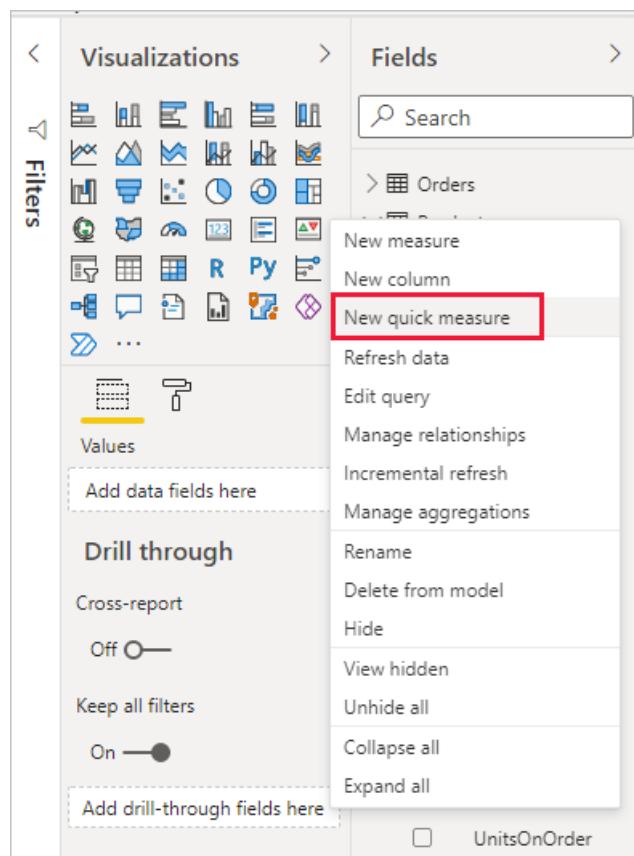
Use quick measures for common calculations

3/30/2022 • 5 minutes to read • [Edit Online](#)

You can use *quick measures* to quickly and easily perform common, powerful calculations. A quick measure runs a set of Data Analysis Expressions (DAX) commands behind the scenes, then presents the results for you to use in your report. You don't have to write the DAX, it's done for you based on input you provide in a dialog box. There are many available categories of calculations and ways to modify each calculation to fit your needs. Perhaps best of all, you can see the DAX that's executed by the quick measure and jump-start or expand your own DAX knowledge.

Create a quick measure

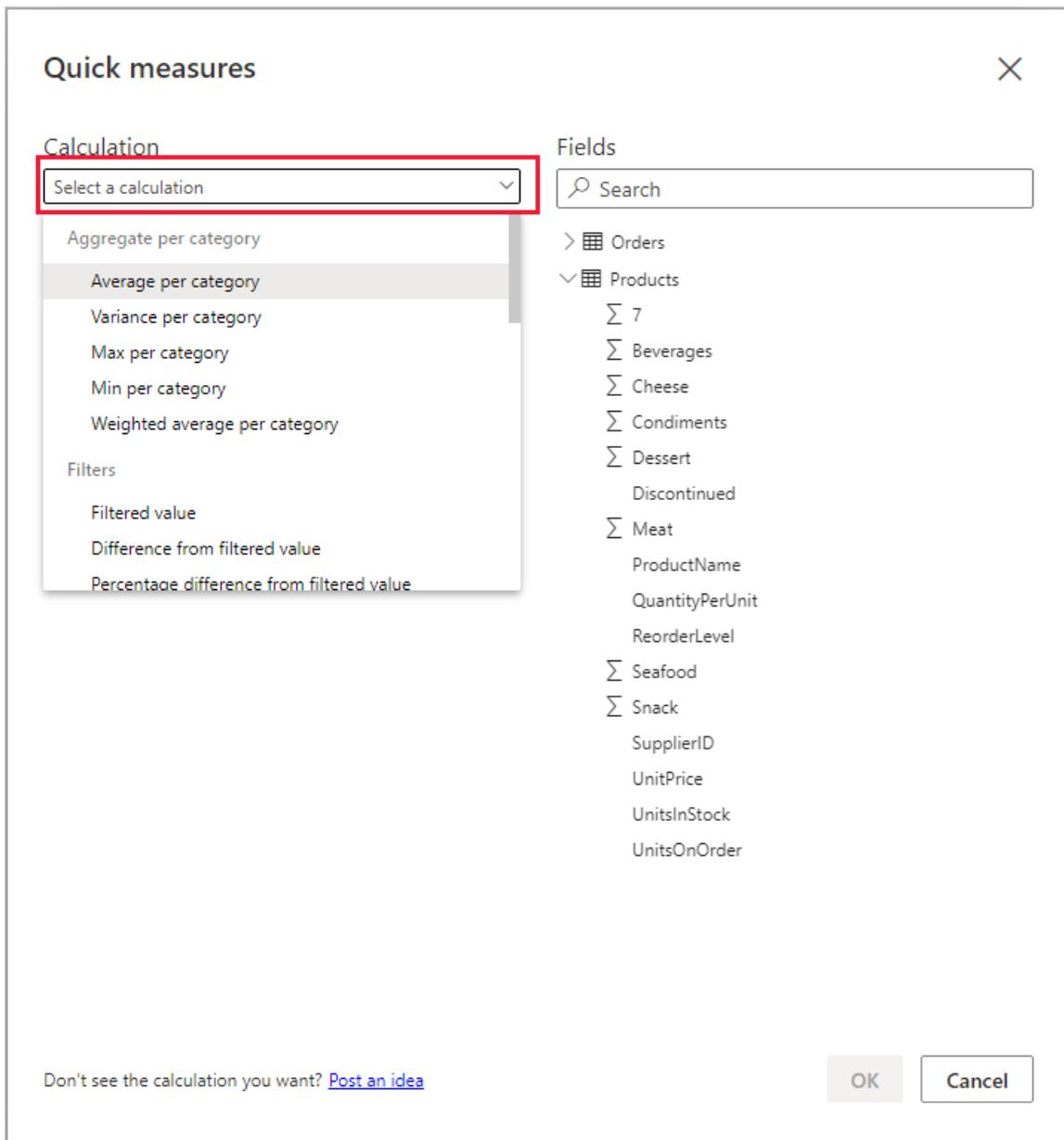
To create a quick measure in Power BI Desktop, right-click or select the ellipsis ... next to any item in the **Fields** pane, and select **New quick measure** from the menu that appears.



You can also right-click or select the drop-down arrow next to any value in the **Values** well for an existing visual, and select **New quick measure** from the menu.

When you select **New quick measure**, the **Quick measures** window appears, letting you select the calculation you want and the fields to run the calculation against.

Select the **Select a calculation** field to see a long list of available quick measures.



The five quick measure calculation types, with their calculations, are:

- **Aggregate per category**
 - Average per category
 - Variance per category
 - Max per category
 - Min per category
 - Weighted average per category
- **Filters**
 - Filtered value
 - Difference from filtered value
 - Percentage difference from filtered value
 - Sales from new customers
- **Time intelligence**
 - Year-to-date total
 - Quarter-to-date total
 - Month-to-date total
 - Year-over-year change
 - Quarter-over-quarter change

- Month-over-month change
- Rolling average
- **Totals**
 - Running total
 - Total for category (filters applied)
 - Total for category (filters not applied)
- **Mathematical operations**
 - Addition
 - Subtraction
 - Multiplication
 - Division
 - Percentage difference
 - Correlation coefficient
- **Text**
 - Star rating
 - Concatenated list of values

To submit your ideas about new quick measures you'd like to see, underlying DAX formulas, or other quick measures ideas for consideration, see the end of this article.

NOTE

When using SQL Server Analysis Services (SSAS) live connections, some quick measures are available. Power BI Desktop displays only the quick measures that are supported for the version of SSAS you're connecting to. If you're connected to a SSAS live data source and don't see certain quick measures in the list, it's because the SSAS version you're connected to doesn't support the DAX commands used to implement those quick measures.

After you select the calculations and fields you want for your quick measure, select OK. The new quick measure appears in the **Fields** pane, and the underlying DAX formula appears in the formula bar.

Quick measure example

Let's take a look at a quick measure in action.

The following matrix visual shows a sales table for various products. It's a basic table that includes the sales totals for each category.

The screenshot shows the Microsoft Power BI desktop interface. On the left, a matrix visual displays sales data with columns for Category, Fashions Direct, Lindsey's, and Total. The data includes various product categories like Womens, Mens, Kids, etc., with their respective sales figures. On the right, the Fields pane is open, showing filters for Rows (Category), Columns (Chain), and Values (TotalSales). The 'TotalSales' field is selected. In the Fields pane, there are also sections for Visualizations, Insert, Calculations, Sensitivity, and Share. A red box highlights the 'TotalSales' field in the Fields pane.

With the matrix visual selected, select the drop-down arrow next to **TotalSales** in the **Values** well, and select **New quick measure**.

In the **Quick measures** window, under **Calculation**, select **Average per category**.

Drag **Average Unit Price** from the **Fields** pane into the **Base value** field. Leave **Category** in the **Category** field, and select **OK**.

Quick measures

Calculation

Average per category

Calculate the average of the base value within the category. [Learn more](#)

Base value (1)

Add data fields here Average Unit Price

Category (1)

Category

Fields

Search

Sales

- Average Unit Price (highlighted with a red box)
- Average Unit Price Last Year
- Avg \$/Unit LY
- Avg \$/Unit TY
- Gross Margin Last Year
- Gross Margin Last Year %
- Gross Margin This Year
- Gross Margin This Year %
- ItemID
- Last Year Sales
- LocationID
- Markdown_Sales_Dollars
- Markdown_Sales_Units
- MonthID
- Regular_Sales_Dollars
- Regular_Sales_Units
- ReportingPeriodID
- Sales Per Sq Ft
- Scenarioid
- Store Count
- Sum_GrossMarginAmount

Don't see the calculation you want? [Post an idea](#)

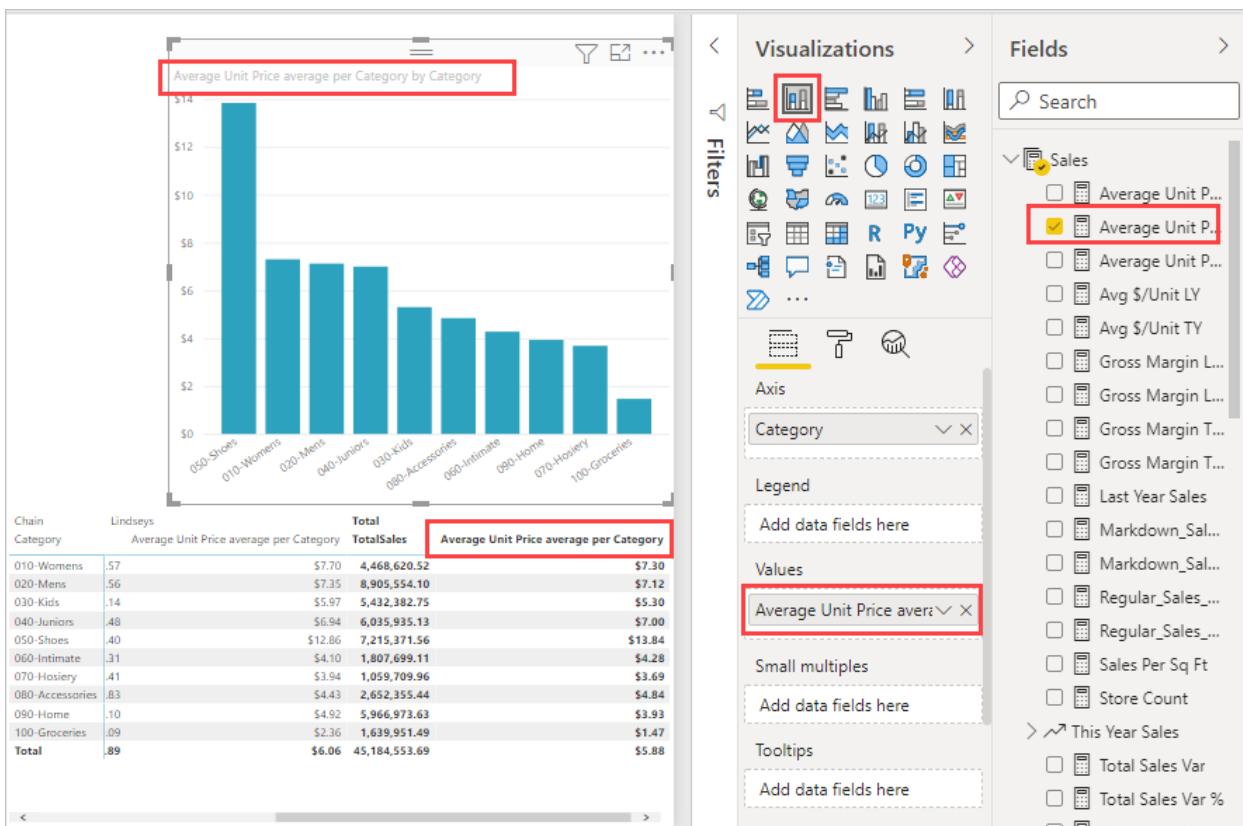
OK (highlighted with a red box) **Cancel**

When you select **OK**, several interesting things happen.

Category	Sub-Category	Sales	Profit	Margin	Average Unit Price average per Category
010-Women	3,549,729.95	\$7.21	918,890.57	\$7.70	\$4,468,620.52
020-Mens	5,125,078.54	\$6.97	3,780,475.56	\$7.35	\$8,905,554.10
030-Kids	4,456,189.61	\$5.18	976,193.14	\$5.97	\$5,432,382.75
040-Juniors	3,506,367.65	\$7.04	2,529,567.48	\$6.94	\$6,035,935.13
050-Shoes	4,367,200.16	\$14.54	2,848,171.40	\$12.86	\$7,215,371.56
060-Intimate	1,369,482.80	\$4.34	438,216.31	\$4.10	\$1,807,699.41
070-Hosiery	772,433.55	\$3.60	287,276.41	\$3.94	\$1,059,709.96
080-Accessories	1,695,255.61	\$5.11	957,099.83	\$4.43	\$2,652,355.44
090-Home	5,335,141.53	\$5.86	431,832.10	\$4.92	\$5,966,973.63
100-Groceries	1,633,661.40	\$1.47	6,290.09	\$2.36	\$1,639,951.49
Total		\$5.93	13,174,012.89	\$6.06	\$45,184,553.69

- The matrix visual has a new column that shows the calculated **Average Unit Price average per Category**.
- The DAX formula for the new quick measure appears in the formula bar. See the [next section](#) for more about the DAX formula.
- The new quick measure appears selected and highlighted in the **Fields** pane.

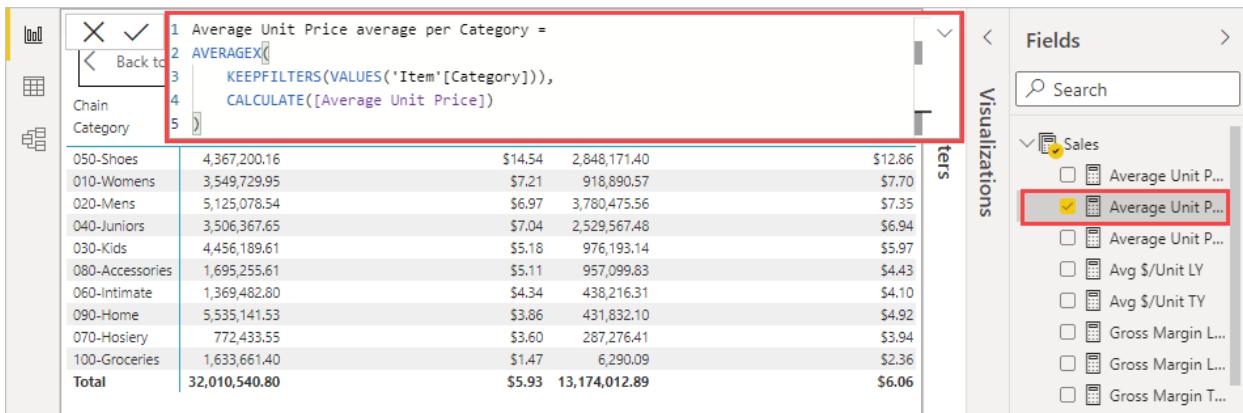
The new quick measure is available to any visual in the report, not just the visual you created it for. The following image shows a quick column chart visual created by using the new quick measure field.



Learn DAX by using quick measures

A great advantage of quick measures is that they show you the DAX formula that implements the measure.

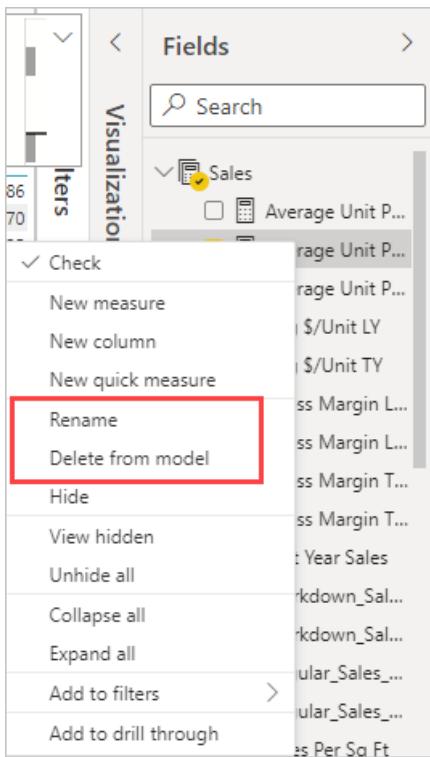
When you select a quick measure in the **Fields** pane, the **Formula bar** appears, showing the DAX formula that Power BI created to implement the measure.



The formula bar not only shows you the formula behind the measure, but perhaps more importantly, lets you see how to create the DAX formulas underlying quick measures.

Imagine you need to do a year-over-year calculation, but you're not sure how to structure the DAX formula, or you have no idea where to start. Instead of banging your head on the desk, you can create a quick measure using the **Year-over-year change** calculation, and see how it appears in your visual and how the DAX formula works. Then you can either make changes directly to the DAX formula, or create a similar measure that meets your needs and expectations. It's like having a teacher that immediately responds to what-if questions you ask with a few clicks.

You can always delete quick measures from your model if you don't like them. That's as easy as right-clicking or selecting the ... next to the measure and selecting **Delete from model**. You can also rename a quick measure whatever you like by selecting **Rename** from the menu.



Considerations and limitations

There are a few considerations and limitations to keep in mind.

- You can use quick measures added to the **Fields** pane with any visual in the report.
- You can always see the DAX associated with a quick measure by selecting the measure in the **Fields** list and looking at the formula in the formula bar.
- Quick measures are only available if you can modify the model. That isn't the case when you're working with some Live connections. SSAS tabular live connections are supported, as previously described.
- You can't create time intelligence quick measures when working in DirectQuery mode. The DAX functions used in these quick measures have performance implications when translated into the T-SQL statements that are sent to your data source.

IMPORTANT

DAX statements for quick measures use only commas for argument separators. If your version of Power BI Desktop is in a language that uses commas as decimal separators, quick measures will not work properly.

Time intelligence and quick measures

You can use your own custom date tables with time intelligence quick measures. If you're using an external tabular model, make sure that when the model was built, the primary date column in the table was marked as a date table, as described in [Specify Mark as Date Table for use with time-intelligence](#). If you're importing your own date table, make sure to mark it as a date table, as described in [Set and use date tables in Power BI Desktop](#).

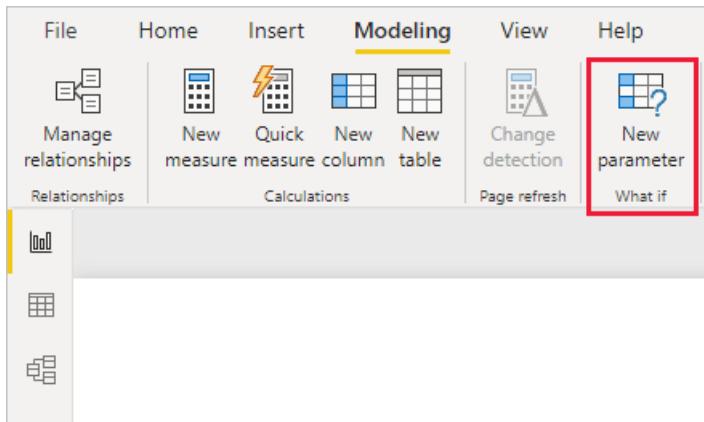
Additional information and examples

Have an idea for a quick measure that isn't already provided? Great! Check out the [Power BI Ideas](#) page, and submit your ideas and DAX formulas for quick measures you'd like to see in Power BI Desktop. We'll consider adding them to the quick measures list in a future release.

Create and use what-if parameters to visualize variables in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

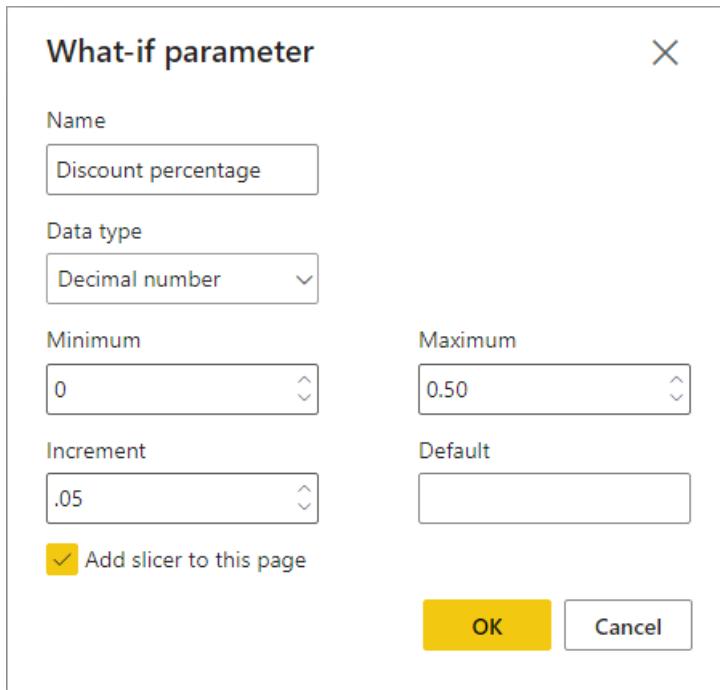
You can create *what-if* variables for your reports, interact with the variable as a slicer, and visualize and quantify different key values in your reports.



Create a *what-if* parameter on the **Modeling** tab in Power BI Desktop. When you select it, a dialog box appears where you can configure the parameter.

Creating a what-if parameter

To create a what-if parameter, select **New Parameter** from the **Modeling** tab in Power BI Desktop. In the following image, we've created a parameter called *Discount percentage* and set its data type to **Decimal number**. The **Minimum** value is zero. The **Maximum** is 0.50 (50 percent). We've also set the **Increment** to 0.05, or five percent. That's how much the parameter will adjust when interacted with in a report.



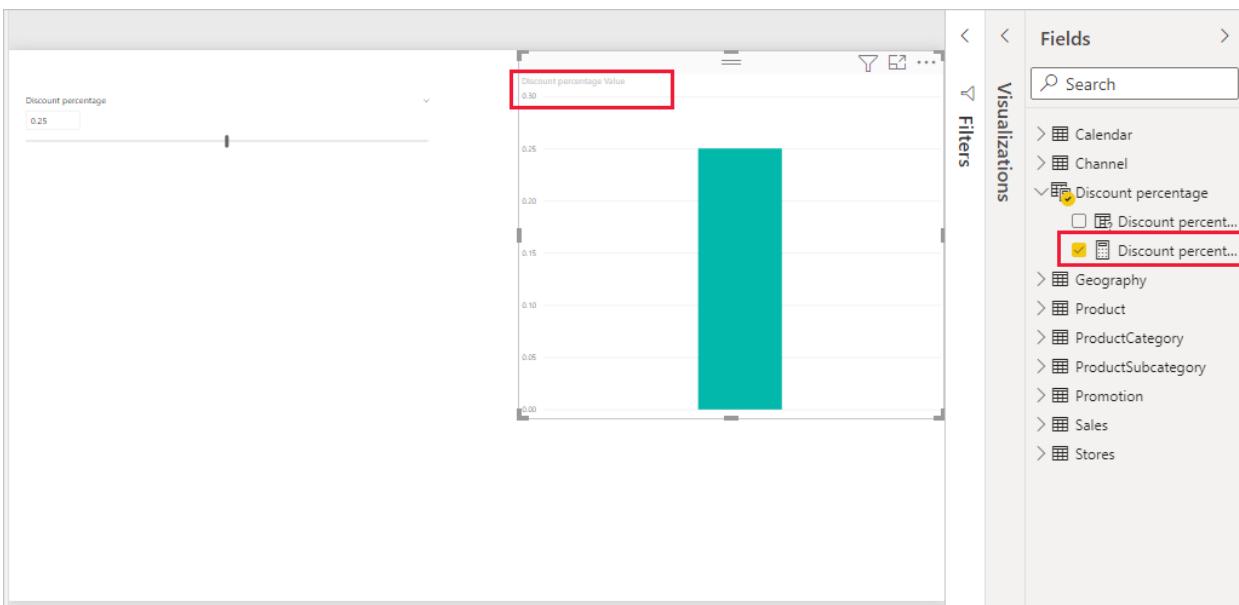
NOTE

For decimal numbers, make sure you precede the value with a zero, as in 0.50 versus just .50. Otherwise, the number won't validate and the OK button won't be selectable.

For your convenience, the **Add slicer to this page** checkbox automatically puts a slicer with your what-if parameter onto the current report page.



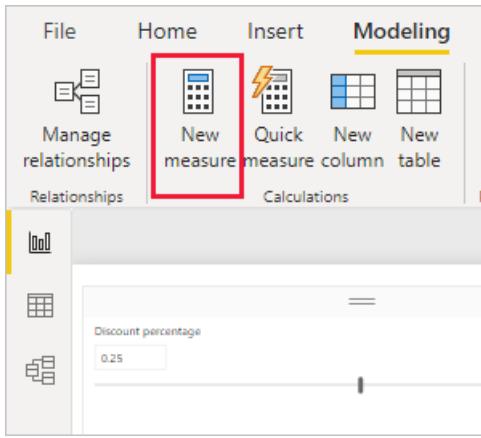
In addition to creating the parameter, creating a what-if parameter also creates a measure, which you can use to visualize the current value of the what-if parameter.



It's important and useful to note that once you create a what-if parameter, both the parameter and the measure become part of your model. So, they're available throughout the report and can be used on other report pages. And, since they're part of the model, you can delete the slicer from the report page. If you want it back, just grab the what-if parameter from the **Fields** list and drag it onto the canvas, then change the visual to a slicer.

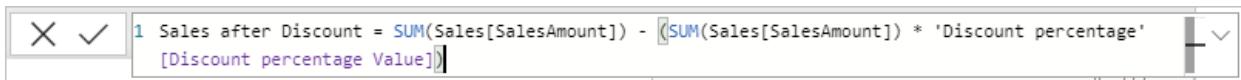
Using a what-if parameter

Let's create a simple example of using a what-if parameter. We created the what-if parameter in the previous section. Now we'll put it to use by creating a new measure whose value adjusts with the slider.

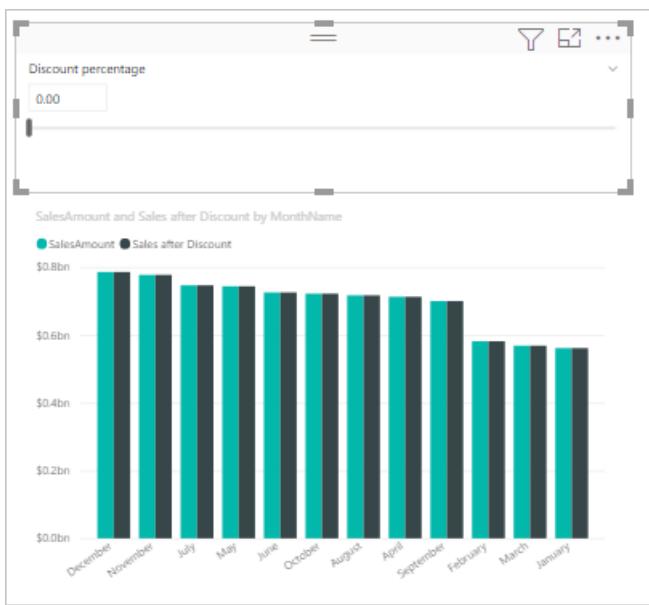


The new measure is simply going to be the total sales amount, with the discount rate applied. You can create complex and interesting measures that let the consumers of your reports visualize the variable of your what-if parameter. For example, you could create a report that lets sales people see their compensation if they meet certain sales goals or percentages, or see the effect of increased sales to deeper discounts.

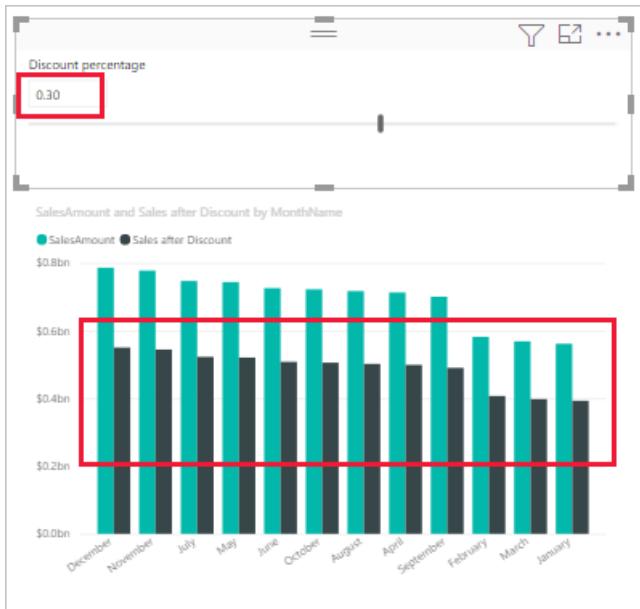
Enter the measure formula into the formula bar, and name the formula *Sales after Discount*.



Then, we create a column visual with **OrderDate** on the axis, and both **SalesAmount** and the just-created measure, **Sales after Discount** as the values.



Then, as we move the slider, we see that the **Sales after Discount** column reflects the discounted sales amount.



And, that's all there is to it. You can use what-if parameters in all sorts of situations. These parameters enable the consumers of reports to interact with different scenarios that you create in your reports.

Considerations and limitations

There are a few considerations and limitations for what-if variables to keep in mind.

- What-if parameters can only be used with value ranges between 0 and 1,000. For ranges greater than 1,000, the parameter value will be sampled.
- What if parameters are designed for measures within visuals, and may not calculate properly when used in a dimension calculation.

Next steps

You might also be interested in the following articles:

- [Use quick measures for common calculations](#)
- [Create calculated columns in Power BI Desktop](#)
- [Create calculated tables in Power BI Desktop](#)

Specify data categories in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

In Power BI Desktop, you can specify the *data category* for a column so Power BI Desktop knows how it should treat its values when in a visualization.

When Power BI Desktop imports data, it gets other information than the data itself, like the table and column names, and whether the data is a primary key. With that information, Power BI Desktop makes some assumptions about how to give you a good default experience when creating a visualization. For example, when a column has numeric values, you'll probably want to aggregate it in some way, so Power BI Desktop places it in the **Values** area of the **Visualizations** pane. Or, for a column with date-time values on a line chart, Power BI Desktop assumes you'll probably use it as a time hierarchy axis.

But, there are some cases that are a bit more challenging, like geography. Consider the following table from an Excel worksheet:

GeoCode	Sales Amount
AL	\$ 10,175,870.00
AR	\$ 4,351,530.00
AZ	\$ 6,114,241.00
CA	\$ 6,688,589.00
KY	\$ 53,832,611.00

Should Power BI Desktop treat the codes in the **GeoCode** column as an abbreviation for a Country or a US State? That's not clear because a code like this can mean either one. For instance, AL can mean Alabama or Albania, AR can mean Arkansas or Argentina, or CA can mean California or Canada. It makes a difference when we go to chart our **GeoCode** field on a map.

Should Power BI Desktop show a picture of the world with countries highlighted? Or should it show a picture of the United States with states highlighted? You can specify a data category for data just like this. Data categorization further refines the information Power BI Desktop can use to provide the best visualizations.

To specify a data category

1. In **Report View** or **Data View**, in the **Fields** list, select the field you want to be sorted by a different categorization.
2. On the ribbon, in the **Properties** area of the **Column tools** tab, select the drop-down arrow next to **Data Category**. This list shows the data categories you can choose for your column. Some selections might be disabled if they won't work with the current data type of your column. For example, if a column is a date or time data type, Power BI Desktop won't let you choose geographic data categories.
3. Select the category you want.

Column tools

Text Summarization Don't summarize Data category Uncategorized

Formatting Auto Sort by column Sort Data groups

Address	I_Pic
Place	sharepoint.com/SiteAssets/image
City	sharepoint.com/SiteAssets/image
County	sharepoint.com/SiteAssets/image
State or Province	sharepoint.com/SiteAssets/image
Postal code	sharepoint.com/SiteAssets/image
Country	sharepoint.com/SiteAssets/image
Continent	sharepoint.com/SiteAssets/image
Latitude	sharepoint.com/SiteAssets/image

DM_Pic_f1
://farm6.staticflickr.com/5502/11550929204_d49a13:
://farm3.staticflickr.com/2811/11551022076_9260a3:
://farm4.staticflickr.com/3682/11550895504_4cfa795:
://farm6.staticflickr.com/5537/11550895544_2f1fc49:
://farm4.staticflickr.com/3672/11550895574_542fb0b4:
://farm8.staticflickr.com/7428/11549627473_2a071cc:
://farm4.staticflickr.com/3833/11549608016_e1c7d0e:
://farm6.staticflickr.com/5506/11549608026_8af4822:
://farm8.staticflickr.com/7358/11549627523_ccf0967:

You might also be interested in learning about geographic filtering for Power BI mobile apps.

Tag barcode fields in Power BI Desktop to enable barcode-scan filtering in the mobile apps

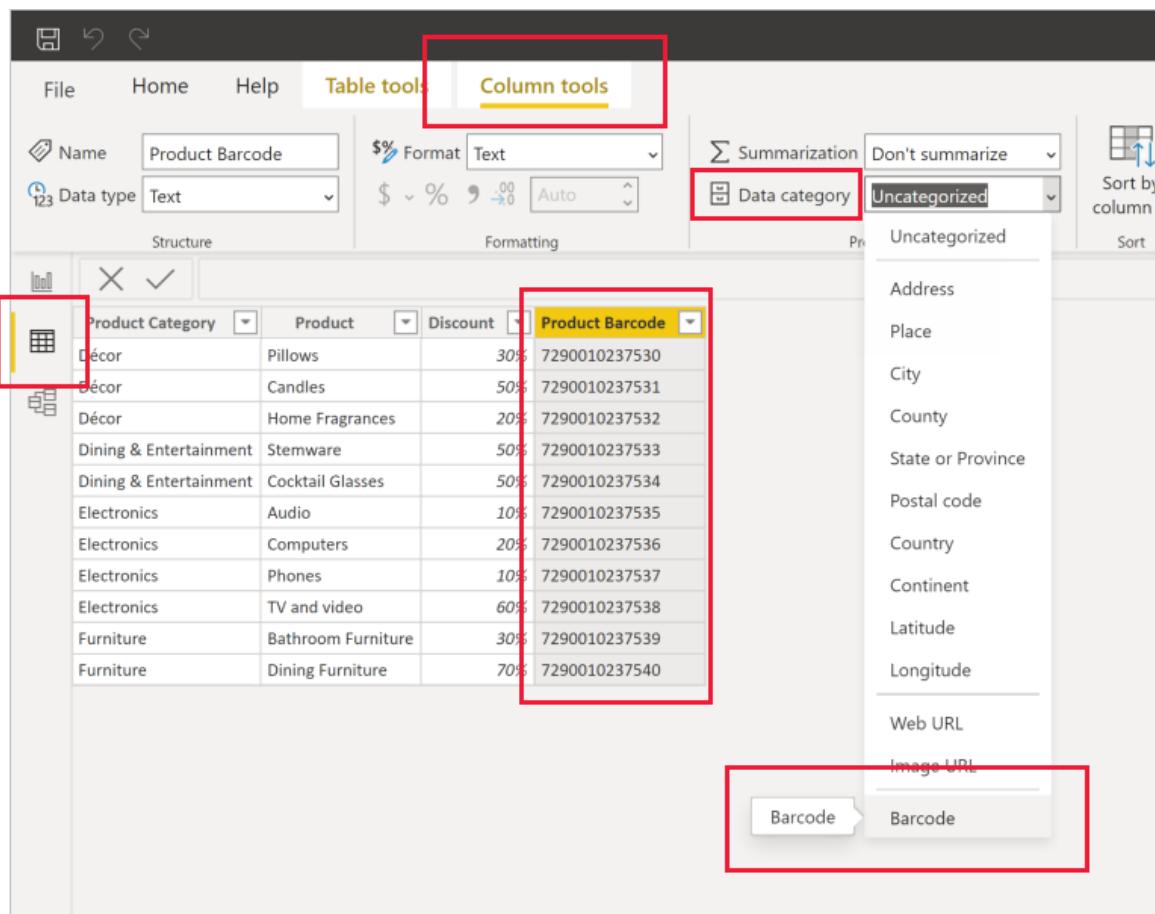
3/30/2022 • 2 minutes to read • [Edit Online](#)

In Power BI Desktop, you can [categorize data](#) in a column, so Power BI Desktop knows how to treat values in visuals in a report. You can also categorize a column as **Barcode**. Then, when someone in your company or organization [scans a barcode](#) on a product using the Power BI mobile app on their iOS or Android phone or tablet, they'll see any report that includes that barcode. When they open the report, it will automatically be filtered to the data related to that barcode.

Categorize barcode data

Assuming you have a report that includes barcodes:

1. In Power BI Desktop, switch to Data view.
2. Select the column that contains the barcode data. See the list of [supported barcode formats](#) below.
3. On the **Column tools** tab, select **Data Category > Barcode**.



WARNING

Do not categorize more than one column across all data tables in a report as **Barcode**. The mobile apps support Barcode filtering only for reports that have only one barcode column across all report data tables. If a report has more than one barcode column, no filtering takes place.

4. In Report view, add the barcode field to the visuals you want filtered by the barcode.

5. Save the report and publish it to the Power BI service.

Now when you open the scanner on the Power BI apps for iOS and Android phones and tablets and scan a barcode, you'll see this report in the list of reports that have barcodes. When you open the report, its visuals will be filtered by the product barcode you scanned.

Supported barcode formats

These are the barcode formats Power BI recognizes if you can tag them in a Power BI report:

- UPCECode
- Code39Code
- A39Mod43Code
- EAN13Code
- EAN8Code
- 93Code
- 128Code
- PDF417Code
- Interleaved2of5Code
- ITF14Code

Next steps

- [Scan a barcode from the Power BI app on your iOS or Android phone or tablet](#)
- [Issues with scanning barcodes](#)
- [Data categorization in Power BI Desktop](#)
- Questions? [Try asking the Power BI Community](#)

Set geographic filters in Power BI Desktop for use in the mobile app

3/30/2022 • 2 minutes to read • [Edit Online](#)

In Power BI Desktop, you can [categorize geographical data](#) for a column, so Power BI Desktop knows how to treat values in visuals in a report. As an added benefit, when you or your colleagues view that report in the Power BI mobile apps, Power BI automatically provides geographical filters that match where you are.

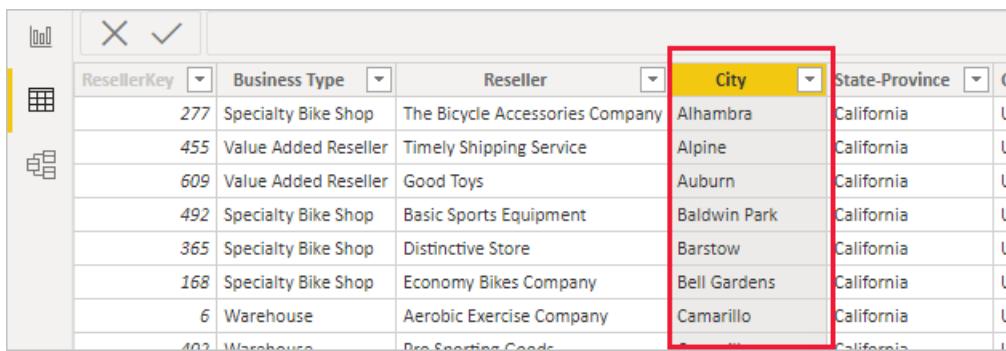
For example, say you're a sales manager traveling to meet customers, and you'd like to quickly filter the total sales and revenue for the specific customer you're planning to visit. You want to break out the data for your current location, whether by state, city, or an actual address. Later, if you have time left, you'd like to visit other customers located nearby. You can [filter the report by your location to find those customers](#).

NOTE

You can only filter by location in the mobile app if the geographic names in the report are in English for example, "New York City" or "Germany".

Identify geographic data in your report

1. In Power BI Desktop, switch to Data View 
2. Select a column with geographic data for example, a City column.



ResellerKey	Business Type	Reseller	City	State-Province	Others
277	Specialty Bike Shop	The Bicycle Accessories Company	Alhambra	California	L
455	Value Added Reseller	Timely Shipping Service	Alpine	California	L
609	Value Added Reseller	Good Toys	Auburn	California	L
492	Specialty Bike Shop	Basic Sports Equipment	Baldwin Park	California	L
365	Specialty Bike Shop	Distinctive Store	Barstow	California	L
168	Specialty Bike Shop	Economy Bikes Company	Bell Gardens	California	L
6	Warehouse	Aerobic Exercise Company	Camarillo	California	L
402	Warehouse	Pro Sporting Goods		California	L

3. On the **Modeling** tab, select **Data Category**, then the correct category in this example, **City**.

Column tools

Text	Summarization	Don't summarize	Sort by column
Auto	Data category	City	Sort
Formatting	Properties	Uncategorized	
Reseller	City	State-Prov	Address
Le Accessories Company	Alhambra	California	Place
ipping Service	Alpine	California	City
s	Auburn	California	County
rts Equipment	Baldwin Park	California	State or Province
e Store	Barstow	California	Postal code
Bikes Company	Bell Gardens	California	Country
xercise Company	Camarillo	California	Continent
ng Goods	Camarillo	California	Latitude
Bike Store	Camarillo	California	91801
g Supplies	Canoga Park	California	91901
			95603
			91706
			92311
			90201
			93010
			93010
			93010
			91303

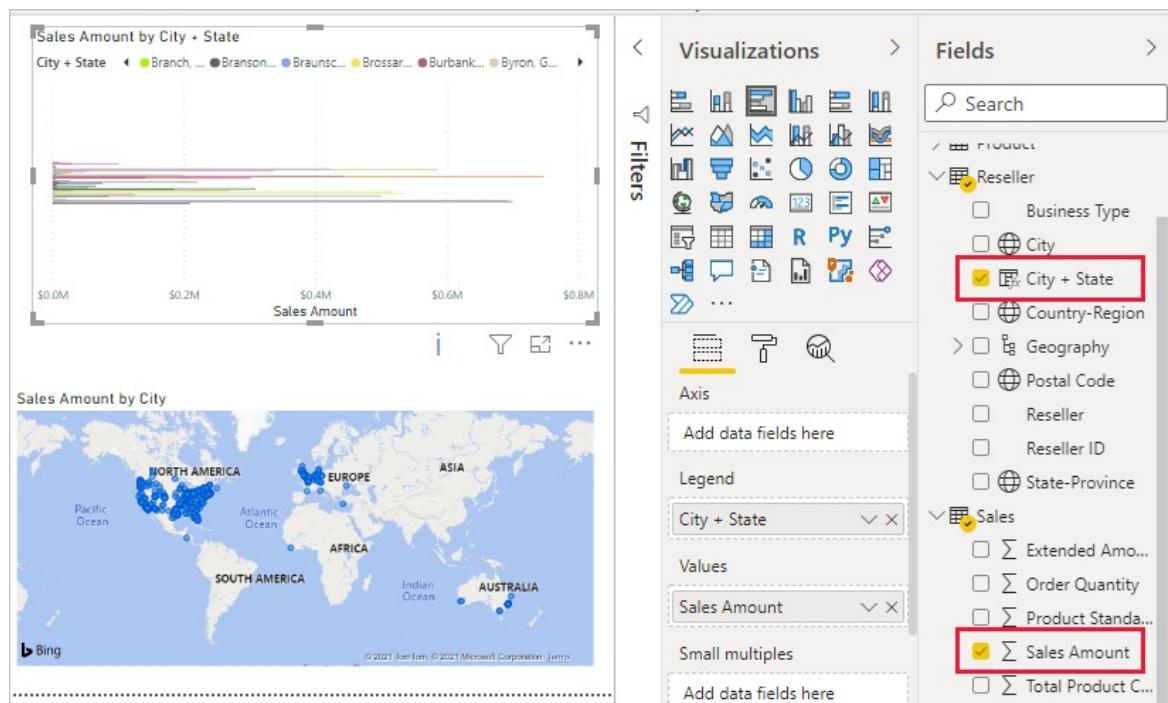
4. Continue setting geographic data categories for any other fields in the model.

NOTE

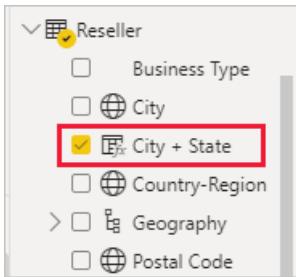
You can set multiple columns for each data category in a model, but if you do the model can't filter for geography in the Power BI mobile app. To use geographic filtering in the mobile apps, set only one column for each data category for example, only one **City** column, one **State or Province** column, and one **Country** column.

Create visuals with your geographic data

1. Switch to Report view  , and create visuals that use the geographic fields in your data.



In this example, the model also contains a calculated column that brings city and state together in one column. Read about [creating calculated columns in Power BI Desktop](#).

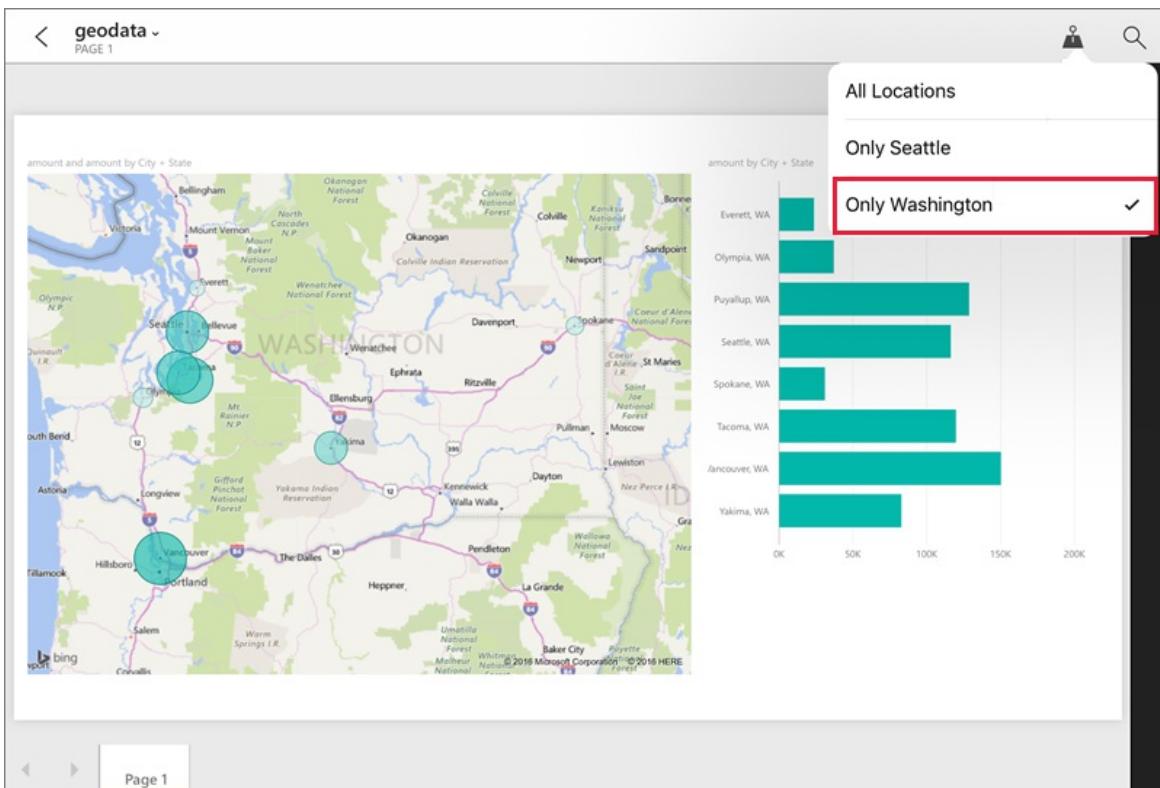


2. Publish the report to the Power BI service.

View the report in Power BI mobile app

1. Open the report in any of the [Power BI mobile apps](#).

2. If you're in a geographic location with data in the report, you can filter it automatically to your location.



Read more about [filtering a report by location in the Power BI mobile apps](#).

Next steps

- [Data categorization in Power BI Desktop](#)
- Questions? [Try asking the Power BI Community](#)

Create calculated columns in Power BI Desktop

3/30/2022 • 3 minutes to read • [Edit Online](#)

With calculated columns, you can add new data to a table already in your model. But instead of querying and loading values into your new column from a data source, you create a Data Analysis Expressions (DAX) formula that defines the column's values. In Power BI Desktop, calculated columns are created by using the new column feature in **Report** view.

Unlike custom columns that are created as part of a query by using **Add Custom Column** in Power Query Editor, calculated columns that are created in **Report** view or **Data** view are based on data you've already loaded into the model. For example, you might choose to concatenate values from two different columns in two different but related tables, do addition, or extract substrings.

Calculated columns you create appear in the **Fields** list just like any other field, but they'll have a special icon showing its values are the result of a formula. You can name your columns whatever you want, and add them to a report visualization just like other fields.

The screenshot shows the 'Fields' list in Power BI. At the top, there is a search bar with a magnifying glass icon. Below it, a tree view shows a 'Calendar' table expanded, revealing six calculated columns: DayOfMonth, DayOfWeekName, MonthName, MonthOfYear, QuarterOfYear, and Year. Each column has a small icon next to its name, indicating it is a calculated column.

Calculated columns calculate results by using DAX, a formula language meant to work with relational data like in Power BI Desktop. DAX includes a library of over 200 functions, operators, and constructs. It provides immense flexibility in creating formulas to calculate results for just about any data analysis need. To learn more about DAX, see [DAX basics in Power BI Desktop](#).

DAX formulas are similar to Excel formulas. In fact, DAX has many of the same functions as Excel. DAX functions, however, are meant to work over data interactively sliced or filtered in a report, like in Power BI Desktop. In Excel, you can have a different formula for each row in a table. In Power BI, when you create a DAX formula for a new column, it will calculate a result for every row in the table. Column values are recalculated as necessary, like when the underlying data is refreshed and values have changed.

Let's look at an example

Jeff is a shipping manager at Contoso, and wants to create a report showing the number of shipments to different cities. Jeff has a **Geography** table with separate fields for city and state. But, Jeff wants their reports to show the city and state values as a single value on the same row. Right now, Jeff's **Geography** table doesn't have the wanted field.

Fields

Search

Geography

- City
- State

Product

But with a calculated column, Jeff can put together the cities from the **City** column with the states from the **State** column.

Jeff right clicks on the **Geography** table and then selects **New Column**. Jeff then enters the following DAX formula into the formula bar:

```
X ✓ CityState = [City] & ", " & [State]
```

This formula simply creates a new column named **CityState**. For each row in the **Geography** table, it takes values from the **City** column, adds a comma and a space, and then concatenates values from the **State** column.

Now Jeff has the wanted field.

Fields

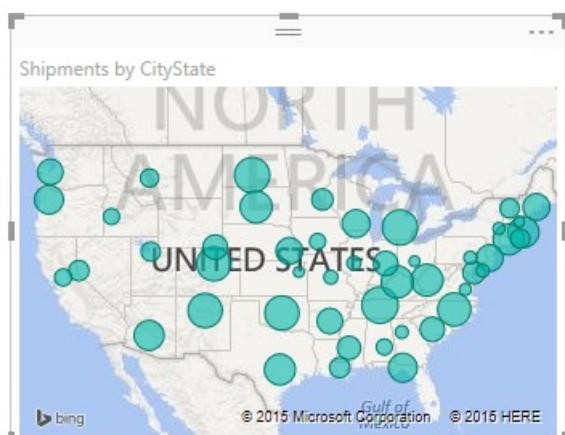
Search

Geography

- City
- CityState
- State

Product

Jeff can now add it to the report canvas along with the number of shipments. With minimal effort, Jeff now has a **CityState** field that can be added to just about any type of visualization. When Jeff creates a new map, Power BI Desktop already knows how to read the city and state values in the new column.



Next steps

We've only provided a quick introduction to calculated columns here. For more information, check out the following resources:

- To download a sample file and get step-by-step lessons on how to create more columns, see [Tutorial: Create calculated columns in Power BI Desktop](#)
- To learn more about DAX, see [DAX basics in Power BI Desktop](#).
- To learn more about columns you create as part of a query, see the **Create custom columns** section in [Common query tasks in Power BI Desktop](#).

Create calculated tables in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

Most of the time, you create tables by importing data into your model from an external data source. But *calculated tables* let you add new tables based on data you've already loaded into the model. Instead of querying and loading values into your new table's columns from a data source, you create a [Data Analysis Expressions \(DAX\)](#) formula to define the table's values.

DAX is a formula language for working with relational data, like in Power BI Desktop. DAX includes a library of over 200 functions, operators, and constructs, providing immense flexibility in creating formulas to calculate results for just about any data analysis need. Calculated tables are best for intermediate calculations and data you want to store as part of the model, rather than calculating on the fly or as query results. For example, you might choose to *union* or *cross join* two existing tables.

Just like other Power BI Desktop tables, calculated tables can have relationships with other tables. Calculated table columns have data types, formatting, and can belong to a data category. You can name your columns whatever you want, and add them to report visualizations just like other fields. Calculated tables are recalculated if any of the tables they pull data from are refreshed or updated, unless the table uses data from a table that uses DirectQuery; in the case with DirectQuery, the table will only reflect the changes once the dataset has been refreshed. If a table needs to use DirectQuery, it's best to have the calculated table in DirectQuery as well.

Create a calculated table

You create calculated tables by using the **New table** feature in Report View or Data View of Power BI Desktop.

For example, imagine you're a personnel manager who has a table of **Northwest Employees** and another table of **Southwest Employees**. You want to combine the two tables into a single table called **Western Region Employees**.

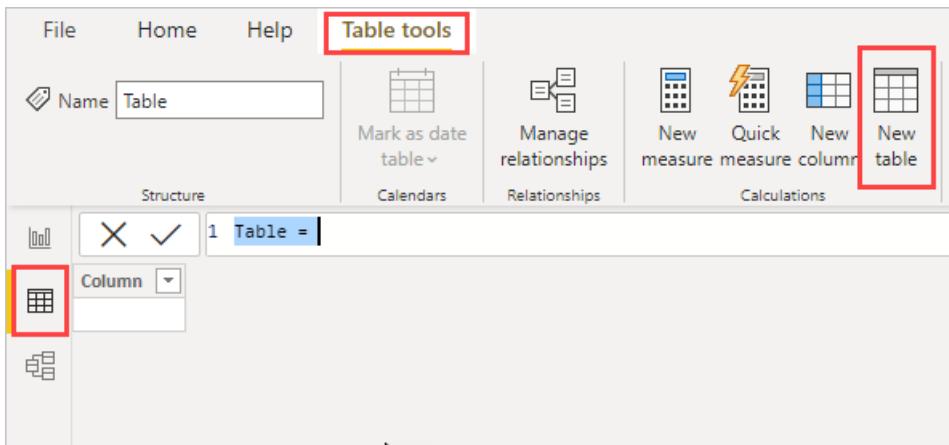
Northwest Employees

Employee	City	State	Tenure
Allen, Kerry	Eugene	OR	1
Baker, Cameron	Portland	OR	15
Morin, Max	Redmond	WA	10
Ramirez, Riley	Portland	OR	3
Rocha, Kim	Redmond	WA	15
Smith, Avery	Redmond	WA	15

Southwest Employees

Employee	City	State	Tenure
Connors, Morgan	San Diego	CA	10
Irwin, Jesse	Phoenix	AZ	3
Nguyen, Rory	Los Angeles	CA	3
Torres, Devon	Los Angeles	CA	2

In Report View or Data View of Power BI Desktop, in the **Calculations** group of the **Modeling** tab, select **New table**. It's a bit easier to do in **Table tools** in the Data View, because then you can immediately see your new calculated table.



Enter the following formula in the formula bar:

```
Western Region Employees = UNION('Northwest Employees', 'Southwest Employees')
```

A new table named **Western Region Employees** is created, and appears just like any other table in the **Fields** pane. You can create relationships to other tables, add measures and calculated columns, and add the fields to reports just like with any other table.

	X	✓	1 Western Region Employees = UNION('Northwest Employees', 'Southwest Employees')	
	Employee	City	State	Tenure
	Allen, Kerry	Eugene	OR	1
	Baker, Cameron	Portland	OR	15
	Morin, Max	Redmond	WA	10
	Ramirez, Riley	Portland	OR	3
	Rocha, Kim	Redmond	WA	15
	Smith, Avery	Redmond	WA	15
	Connors, Morgan	San Diego	CA	10
	Irwin, Jesse	Phoenix	AZ	3
	Nguyen, Rory	Los Angeles	CA	3
	Torres, Devon	Los Angeles	CA	2

Fields

Search

- Northwest Employees
- Southwest Employees
- Western Region Employees
 - City
 - Employee
 - State
 - Tenure

Functions for calculated tables

You can define a calculated table by any DAX expression that returns a table, including a simple reference to another table. For example:

```
New Western Region Employees = 'Western Region Employees'
```

This article provides only a quick introduction to calculated tables. You can use calculated tables with DAX to

solve many analytical problems. Here are some of the more common DAX table functions you might use:

- DISTINCT
- VALUES
- CROSSJOIN
- UNION
- NATURALINNERJOIN
- NATURALLEFTOUTERJOIN
- INTERSECT
- CALENDAR
- CALENDARAUTO

See the [DAX Function Reference](#) for these and other DAX functions that return tables.

Create measures for data analysis in Power BI Desktop

3/30/2022 • 5 minutes to read • [Edit Online](#)

Power BI Desktop helps you create insights into your data with just a few clicks. But sometimes that data just doesn't include everything you need to answer some of your most important questions. Measures can help you get there.

Measures are used in some of the most common data analyses. Simple summarizations such as sums, averages, minimum, maximum and counts can be set through the **Fields** well. The calculated results of measures are always changing in response to your interaction with your reports, allowing for fast and dynamic ad-hoc data exploration. Let's take a closer look. For more information, see [Create measures](#).

Understanding measures

In Power BI Desktop, measures are created and displayed in *Report View* or *Data View*. Measures you create yourself appear in the **Fields** list with a calculator icon. You can name measures whatever you want, and add them to a new or existing visualization just like any other field.

The screenshot shows the 'Fields' list in Power BI Desktop. At the top, there is a search bar labeled 'Search'. Below it, a tree view lists several categories and their contents:

- financials**:
 - \sum Sales
 - \sum COGS
 - Country
- Date**:
 - Discount Band
 - \sum Discounts
 - \sum Gross Sales
 - \sum Manufacturing P...
 - Month Name
 - \sum Month Number
- Net Sales**:
 - Net Sales
 - Net Sales per M...
- Total Sales**:
 - Product
 - \sum Profit
 - \sum Sale Price
 - Segment
- Sheet1**:
 - \sum Units Sold
 - \sum Year

Two specific measures are highlighted with red boxes: 'Net Sales' and 'Total Sales'.

NOTE

You might also be interested in *quick measures*, which are ready-made measures you can select from dialog boxes. They're a good way to quickly create measures, and also a good way to learn Data Analysis Expressions (DAX) syntax, since their automatically created DAX formulas are available to review. For more information, see [quick measures](#).

Data Analysis Expressions

Measures calculate a result from an expression formula. When you create your own measures, you'll use the **Data Analysis Expressions (DAX)** formula language. DAX includes a library of over 200 functions, operators, and constructs. Its library provides immense flexibility in creating measures to calculate results for just about any data analysis need.

DAX formulas are a lot like Excel formulas. DAX even has many of the same functions as Excel, such like `DATE`, `SUM`, and `LEFT`. But the DAX functions are meant to work with relational data like we have in Power BI Desktop.

Let's look at an example

Jan is a sales manager at Contoso. Jan has been asked to provide reseller sales projections over the next fiscal year. Jan decides to base the estimates on last year's sales amounts, with a six percent annual increase resulting from various promotions that are scheduled over the next six months.

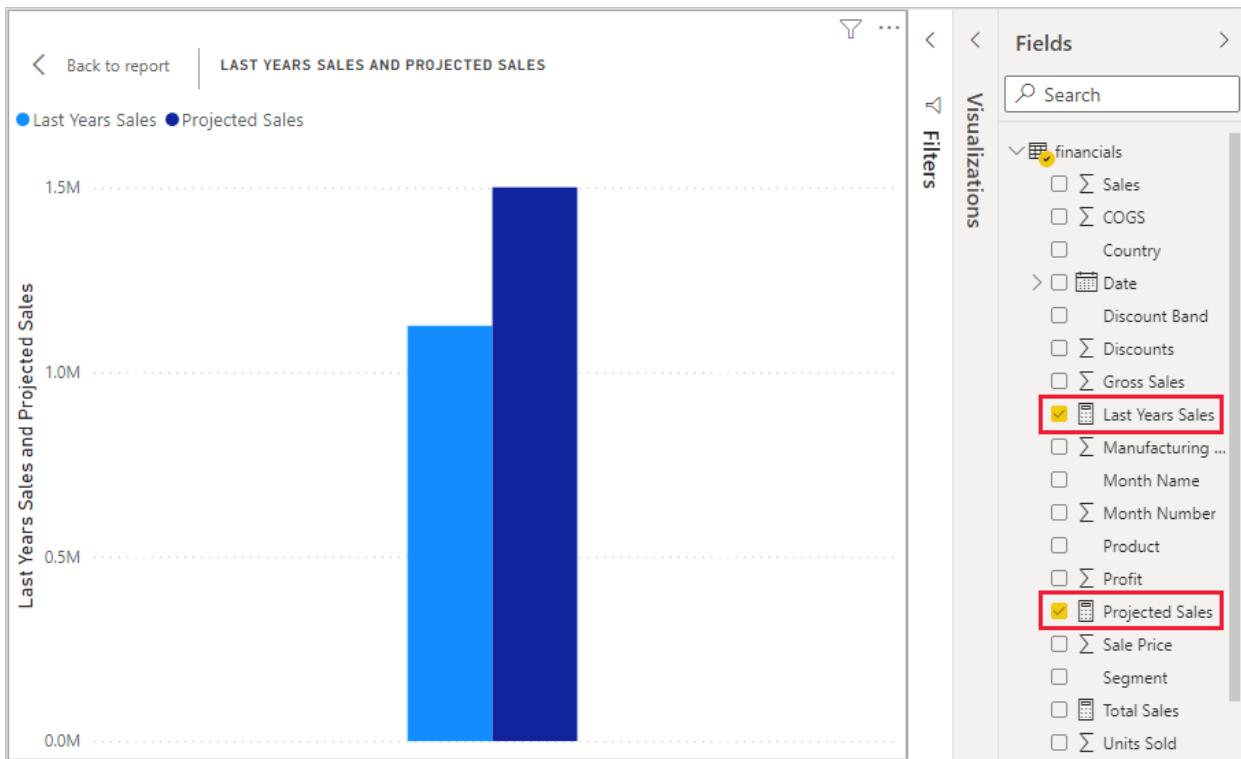
To report the estimates, Jan imports last year's sales data into Power BI Desktop. Jan finds the **SalesAmount** field in the **Reseller Sales** table. Because the imported data only contains sales amounts for last year, Jan renames the **SalesAmount** field to *Last Years Sales*. Jan then drags **Last Years Sales** onto the report canvas. It appears in a chart visualization as single value that is the sum of all reseller sales from last year.

Jan notices that even without specifying a calculation, one has been provided automatically. Power BI Desktop created its own measure by summing up all of the values in **Last Years Sales**.

But Jan needs a measure to calculate sales projections for the coming year, which will be based on last year's sales multiplied by 1.06 to account for the expected 6 percent increase in business. For this calculation, Jan will create a measure. Using the *New Measure* feature, Jan creates a new measure, then enters the following DAX formula:

```
Projected Sales = SUM('Sales'[Last Years Sales])*1.06
```

Jan then drags the new Projected Sales measure into the chart.



Quickly and with minimal effort, Jan now has a measure to calculate projected sales. Jan can further analyze the projections by filtering on specific resellers or by adding other fields to the report.

Data categories for measures

You can also pick data categories for measures.

Among other things, data categories allow you to use measures to dynamically create URLs, and mark the data category as a Web URL.

You could create tables that display the measures as Web URLs, and be able to click on the URL that's created based on your selection. This approach is especially useful when you want to link to other Power BI reports with [URL filter parameters](#).

Organizing your measures

Measures have a *Home* table that defines where they're found in the field list. You can change their location by choosing a location from the tables in your model.

You can also organize fields in a table into *Display Folders*. Select **Model** from the left edge of the Power BI Desktop. In the **Properties** pane, select the field you want to move from the list of available fields. Enter a name for a new folder in **Display folder** to create a folder. Creating a folder moves the selected field into that folder.

The screenshot shows the Power BI Fields pane. On the left, there are three configuration sections: 'Description' (with a placeholder 'Enter a description'), 'Display folder' (with a placeholder 'Enter the display folder'), and 'Is hidden' (set to 'Off'). On the right, a tree view shows the 'Sales' folder expanded, containing the following fields:

- channelKey
- DateKey
- DiscountAmount
- DiscountQuantity
- ProductKey
- Profit** (highlighted)
- PromotionKey
- ReturnAmount
- ReturnQuantity

You can create subfolders by using a backslash character. For example, *Finance\urrencies* creates a *Finance* folder and within it, a *Currencies* folder.

You can make a field appear in multiple folders by using a semicolon to separate the folder names. For example, *Products\Names;Departments* results in the field appearing in a *Departments* folder as well as a *Names* folder inside a *Products* folder.

You can create a special table that contains only measures. That table always appears at the top of the **Fields**. To do so, create a table with just one column. You can use **Enter data** to create that table. Then move your measures to that table. Finally, hide the column, but not the table, that you created. Select the arrow at the top of **Fields** to close and reopen the fields list to see your changes.

The screenshot shows the Power BI Fields pane. On the left, there is a 'Fields' section with a search bar. Below it, the 'Calculations' section is expanded, showing three measures: 'Last Years Sales', 'Projected Sales', and 'Total Sales'. Underneath these, the 'Channel' and 'Sales' sections are collapsed.

TIP

Hidden measures are displayed and accessible in Power BI Desktop, however, you will not see hidden measures in Excel or the Power BI services, since Excel and the Power BI service are considered client tools.

Learn more

We've only provided you with a quick introduction to measures here. There's a lot more to help you learn how to create your own. For more information, see [Tutorial: Create your own measures in Power BI Desktop](#). You can download a sample file and get step-by-step lessons on how to create more measures.

To dive a little deeper into DAX, see [DAX basics in Power BI Desktop](#). The [Data Analysis Expressions Reference](#) provides detailed articles on each of the functions, syntax, operators, and naming conventions. DAX has been around for several years in Power Pivot in Excel and SQL Server Analysis Services. There are many other great resources available, too. Be sure to check out the [DAX Resource Center Wiki](#), where influential members of the BI community share their knowledge of DAX.

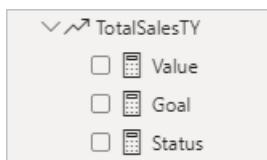
Import and display KPIs in Power BI

3/30/2022 • 2 minutes to read • [Edit Online](#)

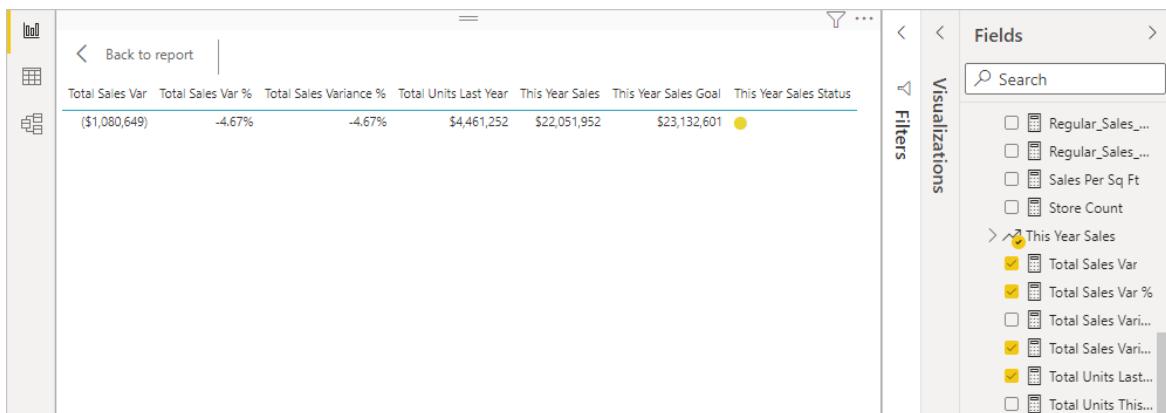
With **Power BI Desktop**, you can import and display KPIs in tables, matrices, and cards.

Follow these steps to import and display KPIs.

1. Start with an Excel workbook that has a Power Pivot model and KPIs.
2. Import the Excel workbook into Power BI, using **File -> Import -> Excel workbook contents**. You can also [learn how to import workbooks](#).
3. After import into Power BI, your KPI will appear in the **Fields** pane, marked with the  icon. To use a KPI in your report, be sure to expand its contents, exposing the **Value**, **Goal**, and **Status** fields.



4. Imported KPIs are best used in standard visualization types, such as the **Table** type. Power BI also includes the **KPI** visualization type, which should only be used to create new KPIs.



The screenshot shows the Power BI desktop interface. On the left, there's a navigation bar with icons for Home, Reports, and Data. In the center, there's a 'Table' visualization showing sales data for the current year. The columns include Total Sales Var, Total Sales Var %, Total Sales Variance %, Total Units Last Year, This Year Sales, This Year Sales Goal, and This Year Sales Status. The data row shows values like (\$1,080,649), -4.67%, -4.67%, \$4,461,252, \$22,051,952, \$23,132,601, and a yellow status indicator. On the right, the 'Fields' pane is open. It has a search bar at the top. Below it, there are sections for 'Visualizations' and 'Filters'. Under 'Visualizations', there's a list of fields: Regular_Sales_..., Regular_Sales_..., Sales Per Sq Ft, Store Count, and a section for 'This Year Sales' which includes 'Total Sales Var' (selected), 'Total Sales Var %' (selected), 'Total Sales Variance %' (unchecked), 'Total Units Last Year' (selected), and 'Total Units This Year' (unchecked). The 'Fields' pane title bar says 'Fields' with arrows for navigation.

That's all there is to it. You can use KPIs to highlight trends, progress, or other important indicators.

Apply auto date/time in Power BI Desktop

3/30/2022 • 5 minutes to read • [Edit Online](#)

This article targets data modelers developing Import or Composite models in Power BI Desktop. It introduces and describes the *Auto date/time* option.

The Auto date/time is a data load option in Power BI Desktop. The purpose of this option is to support convenient time intelligence reporting based on date columns loaded into a model. Specifically, it allows report authors using your data model to filter, group, and drill down by using calendar time periods (years, quarters, months, and days). What's important is that you don't need to explicitly develop these time intelligence capabilities.

When the option is enabled, Power BI Desktop creates a hidden auto date/time table for each date column, providing all of the following conditions are true:

- The table storage mode is Import
- The column data type is date or date/time
- The column isn't the "many" side of a model relationship

How it works

Each auto date/time table is in fact a [calculated table](#) that generates rows of data by using the DAX [CALENDAR](#) function. Each table also includes six calculated columns: **Day**, **MonthNo**, **Month**, **QuarterNo**, **Quarter**, and **Year**.

NOTE

Power BI translates and formats column names and values according to the [model language](#). For example, if the model was created by using English, it will still show month names, and so on, in English, even if viewed with a Korean client.

Power BI Desktop also creates a relationship between the auto date/time table's **Date** column and the model date column.

The auto date/time table contains full calendar years encompassing all date values stored in the model date column. For example, if the earliest value in a date column is March 20, 2016 and the latest value is October 23, 2019, the table will contain 1,461 rows. It represents one row for each date in the four calendar years 2016 to 2019. When Power BI refreshes the model, each auto date/time table is also refreshed. This way, the model always contains dates that encompass the date column values.

If it were possible to see the rows of an auto date/time table, they would look like this:

Date	Day	MonthNo	Month	QuarterNo	Quarter	Year
01/01/2019 00:00:00	1	1	January	1	Qtr 1	2019
01/02/2019 00:00:00	2	1	January	1	Qtr 1	2019
01/03/2019 00:00:00	3	1	January	1	Qtr 1	2019
01/04/2019 00:00:00	4	1	January	1	Qtr 1	2019
01/05/2019 00:00:00	5	1	January	1	Qtr 1	2019
01/06/2019 00:00:00	6	1	January	1	Qtr 1	2019
01/07/2019 00:00:00	7	1	January	1	Qtr 1	2019
01/08/2019 00:00:00	8	1	January	1	Qtr 1	2019
01/09/2019 00:00:00	9	1	January	1	Qtr 1	2019
01/10/2019 00:00:00	10	1	January	1	Qtr 1	2019

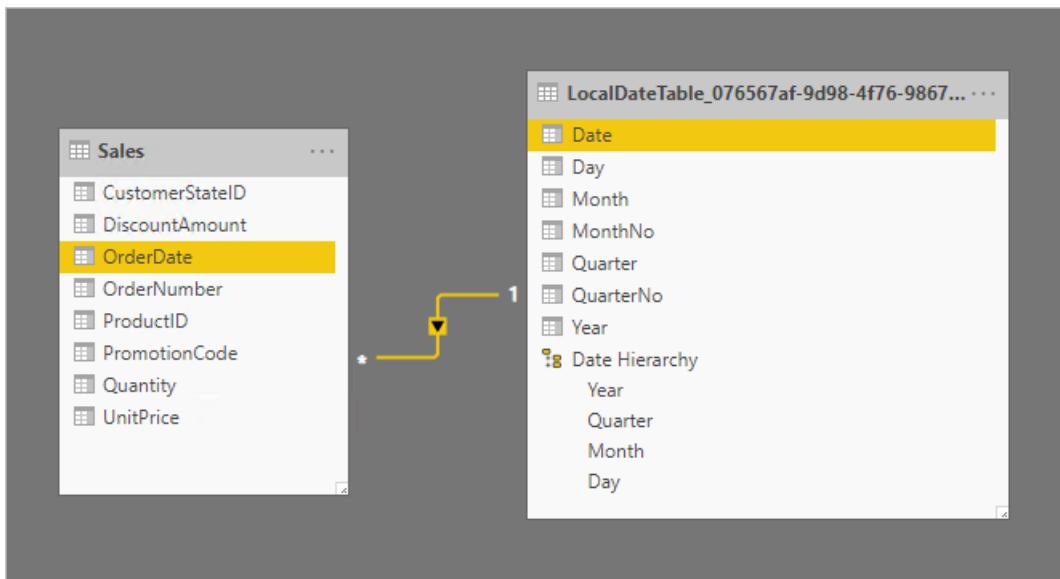
NOTE

Auto date/time tables are permanently hidden, even from modelers. They cannot be seen in the **Fields** pane or the Model view diagram, and its rows cannot be seen in Data view. Also, the table and its column cannot be directly referenced by DAX expressions.

Further, it's not possible to work with them when using [Analyze in Excel](#), or connecting to the model using non-Power BI report designers.

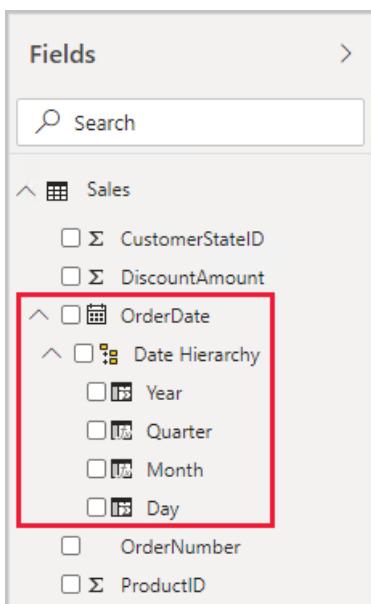
The table also defines a hierarchy, providing visuals with a drill-down path through year, quarter, month, and day levels.

If it were possible to see an auto date/time table in the Model view diagram, it would look like this (related columns are highlighted):



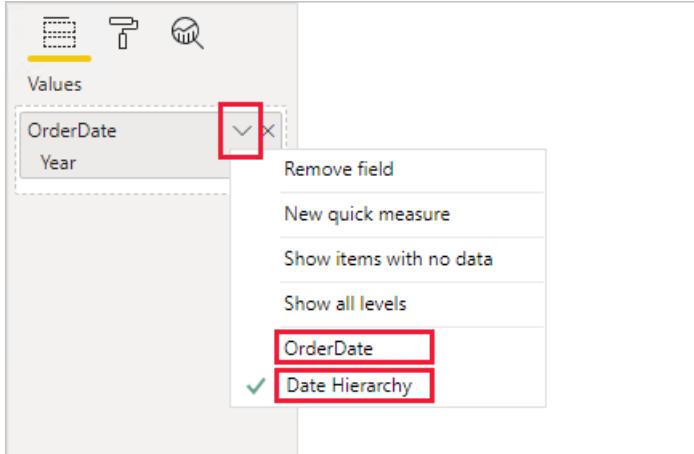
Work with auto date/time

When an auto date/time table exists for a date column (and that column is visible), report authors won't find that column as a field in the **Fields** pane. Instead, they find an expandable object that has the name of the date column. You can easily identify it because it's adorned with a calendar icon. When report authors expand the calendar object, they find a hierarchy named **Date Hierarchy**. After they expand the hierarchy, they find four levels: **Year**, **Quarter**, **Month**, and **Day**.



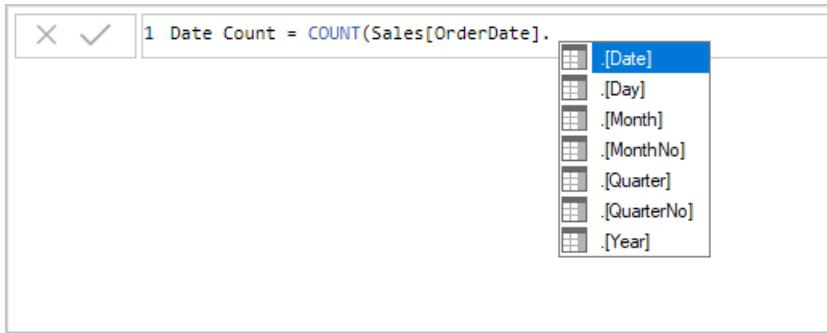
The auto date/time generated hierarchy can be used to configure a visual in exactly the same way that regular hierarchies can be used. Visuals can be configured by using the entire **Date Hierarchy** hierarchy, or specific levels of the hierarchy.

There is, however, one added capability not supported by regular hierarchies. When the auto date/time hierarchy—or a level from the hierarchy—is added to a visual well, report authors can toggle between using the hierarchy or the date column. This approach makes sense for some visuals, when all they require is the date column, not the hierarchy and its levels. They start by configuring the visual field (right-click the visual field, or click the down-arrow), and then using the context menu to switch between the date column or the date hierarchy.



Lastly, model calculations, written in DAX, can reference a date column *directly*, or the hidden auto date/time table columns *indirectly*.

Formula written in Power BI Desktop can reference a date column in the usual way. The auto date/time table columns, however, must be referenced by using a special extended syntax. You start by first referencing the date column, and then following it by a period (.). The formula bar auto complete will then allow you to select a column from the auto date/time table.



In Power BI Desktop, a valid measure expression could read:

```
Date Count = COUNT(Sales[OrderDate].[Date])
```

NOTE

While this measure expression is valid in Power BI Desktop, it's not correct DAX syntax. Internally, Power BI Desktop transposes your expression to reference the true (hidden) auto date/time table column.

Configure auto date/time option

Auto date/time can be configured *globally* or for the *current file*. The global option applies to new Power BI

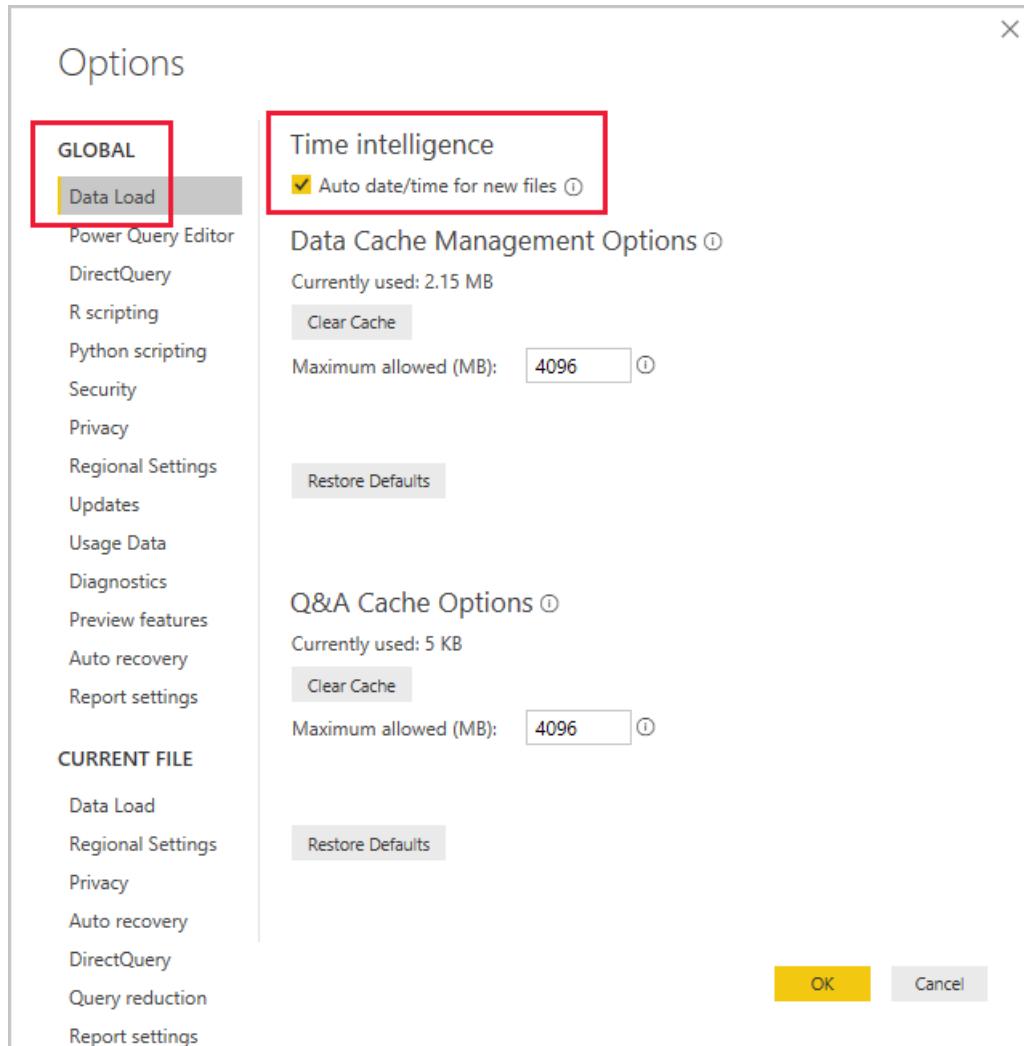
Desktop files, and it can be turned on or off at any time. For a new installation of Power BI Desktop, both options default to on.

The current file option, too, can also be turned on or off at any time. When turned on, auto date/time tables are created. When turned off, any auto date/time tables are removed from the model.

Caution

Take care when you turn the current file option off, as this will remove the auto date/time tables. Be sure to fix any broken report filters or visuals that had been configured to use them.

In Power BI Desktop, you select *File > Options and settings > Options*, and then select either the **Global** or **Current File** page. On either page, the option exists in the **Time intelligence** section.



Considerations and limitations

While using Analysis Services data sources, having relationships defined on the *DateTime* columns may not behave as expected. Analysis Services only uses *DateTime* data types; *Date* and *Time* data types are Power BI formatting constructs implemented on top of Analysis Services, so any model-dependent objects will still appear as *DateTime* to Analysis Services (such as relationships, groups, and so on). As such, if a user selects *Date* from the **Modeling** tab for such columns, they still do not register as being the same date, because the time portion of the data is still being considered by Analysis Services. 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 Analysis Services is handling the data, the values will appear the same.

Next steps

For more information related to this article, check out the following resources:

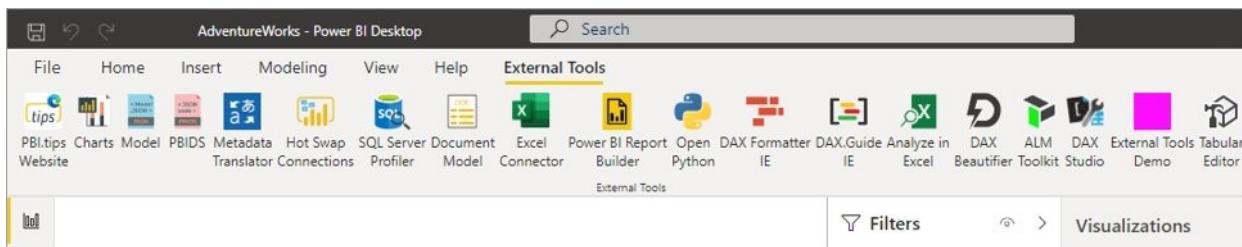
- Auto date/time guidance in Power BI Desktop
- Create date tables in Power BI Desktop
- Set and use date tables in Power BI Desktop
- Questions? [Try asking the Power BI Community](#)
- Suggestions? [Contribute ideas to improve Power BI](#)

External tools in Power BI Desktop

3/30/2022 • 6 minutes to read • [Edit Online](#)

Power BI has a vibrant community of BI professionals and developers. A vital part of that community are contributors that create free tools that use Power BI and Analysis Services APIs to extend and integrate with Power BI Desktop's data modeling and reporting features.

The **External Tools** ribbon provides easy access to external tools that have been installed locally and *registered* with Power BI Desktop. When launched from the External Tools ribbon, Power BI Desktop passes the name and port number of its internal data model engine instance to the tool, and the current model name. The tool then automatically connects, providing a seamless connection experience.



External tools generally fall into one of the following categories:

Semantic modeling - Open-source tools such as DAX Studio, ALM Toolkit, Tabular Editor, and Metadata Translator extend Power BI Desktop functionality for specific data modeling scenarios such as DAX query and expression optimization, application lifecycle management (ALM), and metadata translation.

Data analysis - Tools for connecting to a model in read-only to query data and perform other analysis tasks. For example, tools that launch Python, Excel, and Power BI Report Builder and connect the client application to the model in Power BI Desktop for testing and analysis without having to first publish the Power BI Desktop (.pbix) file to the Power BI service. Tools to document a Power BI dataset also fall into this category.

Miscellaneous - Some external tools don't connect to a model at all, but instead extend Power BI Desktop to make helpful tips and make helpful content more readily accessible. For example, PBI.tips tutorials, DAX Guide from sqlbi.com, and the PowerBI.tips Product Business Ops community tool, which makes installation of a large selection of external tools and their registration with Power BI Desktop, including DAX Studio, ALM Toolkit, Tabular Editor, and many others easy.

Custom - Integrate your own scripts and tools by adding a *.pbitool.json document to the Power BI Desktop\External Tools folder.

Before installing external tools, keep the following in mind:

- External tools are not supported in Power BI Desktop for Power BI Report Server.
- External tools are provided by external, third-party contributors. Microsoft does not provide support or documentation for external tools.

Featured open-source tools

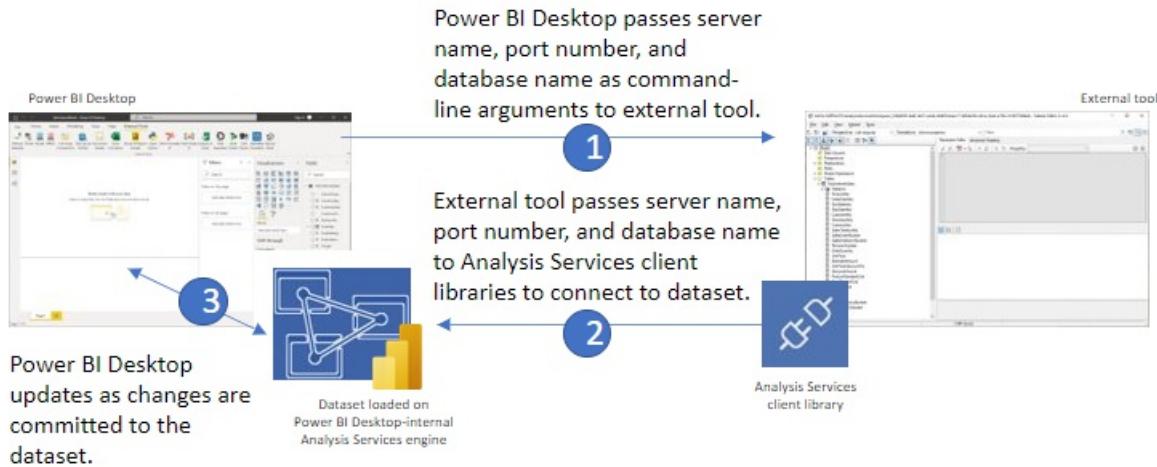
There are many external tools out there. Here are some of the most popular and belong in every Power BI Desktop data modeler's toolbox:

Tool	Description
PowerBI.tips - Business Ops	An easy to use deployment tool for adding external tools extensions to Power BI Desktop. The Business Ops goal is to provide a one stop shop for installing all the latest versions of external tools. To learn more, go to the PowerBI.tips - Business Ops website.
Tabular Editor	Model creators can easily build, maintain, and manage tabular models using an intuitive and lightweight editor. A hierarchical view shows all objects in your tabular model organized by display folders, with support for multi-select property editing and DAX syntax highlighting. To learn more, go to tabulareditor.com .
DAX Studio	A feature-rich tool for DAX authoring, diagnosis, performance tuning, and analysis. Features include object browsing, integrated tracing, query execution breakdowns with detailed statistics, DAX syntax highlighting and formatting. To get the latest, go to DAX Studio on GitHub.
ALM Toolkit	A schema compare tool for Power BI models and datasets, used for application lifecycle management (ALM) scenarios. You can perform straightforward deployment across environments and retain incremental refresh historical data. You can diff and merge metadata files, branches, and repos. You can also reuse common definitions between datasets. To get the latest, go to alm-toolkit.com .
Metadata Translator	Streamlines localization of Power BI models and datasets. The tool can automatically translate captions, descriptions, and display folder names of tables, columns, measures, and hierarchies by using the machine translation technology of Azure Cognitive Services. You can also export and import translations via Comma Separated Values (.csv) files for convenient bulk editing in Excel or a localization tool. To get the latest, go to Metadata Translator on GitHub.

External tools integration architecture

Power BI Desktop (pbix) files consist of multiple components including the report canvas, visuals, model metadata, and any data that has already been loaded from data sources. When Power BI Desktop opens a pbix file, it launches an Analysis Services process in the background to load the model so that the data modeling features and the report visuals can access model metadata and query model data.

When Power BI Desktop launches Analysis Services as its analytical data engine, it dynamically assigns a random port number and loads the model with a randomly generated name in the form of a globally unique identifier (GUID). Because these connection parameters change with every Power BI Desktop session, it's difficult for external tools to discover on their own the correct Analysis Services instance and model to connect to. External tools integration solves this problem by allowing Power BI Desktop to communicate the Analysis Services server name, port number, and model name to the tool as command-line parameters when starting the external tool from the External Tools ribbon, as shown in the following diagram.



With the Analysis Services Server name, port number, and model name, the tool uses Analysis Services client libraries to establish a connection to the model, retrieve metadata, and execute DAX or MDX queries. Whenever an external data modeling tool updates the metadata, Power BI Desktop synchronizes the changes so that the Power BI Desktop user interface reflects the current state of the model accurately. Keep in mind there are some limitations to the synchronization capabilities as described below.

Data modeling operations

External data modeling tools can apply modifications and have Power BI synchronize those changes with the report canvas. This synchronization is so those modifications are consistently applied in Power BI visuals. For example, external data modeling tools can override the original format string expression of a measure, and edit any of the measure properties including KPIs and detail rows. External tools can also create new roles for object and row-level security, and add translations.

Supported write operations

- Define and edit [measures](#) for calculations, including format string, KPI, and detail rows settings.
- Add [calculation groups](#) for calculation reusability in complex models.
- Create [perspectives](#) to define focused, business-domain specific views of dataset metadata.
- Apply [metadata translations](#) to support multi-language versions within a single dataset.
- Add dataset roles for [row-level security \(RLS\)](#) and [object-level security \(OLS\)](#) rules to restrict data access.

Data modeling limitations

All Tabular Object Model (TOM) metadata can be accessed for read-only. Write operations are limited because Power BI Desktop must remain in-sync with the external modifications, therefore the following operations are not supported:

- Any TOM object types not covered in Supported write operations, such as tables and columns.
- Editing a Power BI Desktop template (PBIT) file.
- Report-level or data-level translations.
- Renaming tables and columns is not yet supported
- Sending processing commands to a dataset loaded in Power BI Desktop

Registering external tools

External tools are *registered* with Power BI Desktop when the tool includes a *.pbitool.json registration file in the `C:\Program Files (x86)\Common Files\Microsoft Shared\Power BI Desktop\External Tools` folder. When a tool is registered, and includes an icon, the tool appears in the External Tools ribbon. Some tools, like ALM Toolkit and DAX Studio create the registration file automatically when you install the tool. However, many tools, like SQL Profiler typically do not because the installer they do have does not include creating a registration file for Power

BI Desktop. Tools that don't automatically register with Power BI Desktop can be registered manually by creating a *.pbitool.json registration file.

To learn more, including json examples, see [Register an external tool](#).

Disabling the External Tools ribbon

The External Tools ribbon is enabled by default, but can be disabled by using Group Policy or editing the `EnableExternalTools` registry key directly.

- Registry key: `Software\Policies\Microsoft\Power BI Desktop\`
- Registry value: `EnableExternalTools`

A value of 1 (decimal) enables the External Tools ribbon, which is also the default value.

A value of 0 (decimal) disable the External Tools ribbon.

See also

[Register an external tool](#)

Register an external tool

3/30/2022 • 2 minutes to read • [Edit Online](#)

Some tools must be manually registered with Power BI Desktop. To register an external tool, create a JSON file with the following:

```
{  
    "name": "<tool name>",  
    "description": "<tool description>",  
    "path": "<tool executable path>",  
    "arguments": "<optional command line arguments>",  
    "iconData": "image/png;base64,<encoded png icon data>"  
}
```

The pbitool.json file includes the following elements:

- **name:** Provide a name for the tool, which will appear as a button caption in the External Tools ribbon within Power BI Desktop.
- **description:** (optional) Provide a description, which will appear as a tooltip on the External Tools ribbon button within Power BI Desktop.
- **path:** Provide the fully qualified path to the tool executable.
- **arguments:** (optional) Provide a string of command-line arguments that the tool executable should be launched with. You may use any of the following placeholders:
 - **%server%:** Replaced with the server name and portnumber of the local instance of Analysis Services Tabular for imported/DirectQuery data models.
 - **%database%:** Replaced with the database name of the model hosted in the local instance of Analysis Services Tabular for imported/DirectQuery data models.
- **iconData:** Provide image data, which will be rendered as a button icon in the External Tools ribbon within Power BI Desktop. The string should be formatted according to the syntax for Data URIs without the "data:" prefix.

Name the file `<tool name>.pbitool.json` and place it in the following folder:

- `%commonprogramfiles%\Microsoft Shared\Power BI Desktop\External Tools`

For 64-bit environments, place the files in the following folder:

- `Program Files (x86)\Common Files\Microsoft Shared\Power BI Desktop\External Tools`

Files in that specified location with the `.pbitool.json` extension are loaded by Power BI Desktop upon startup.

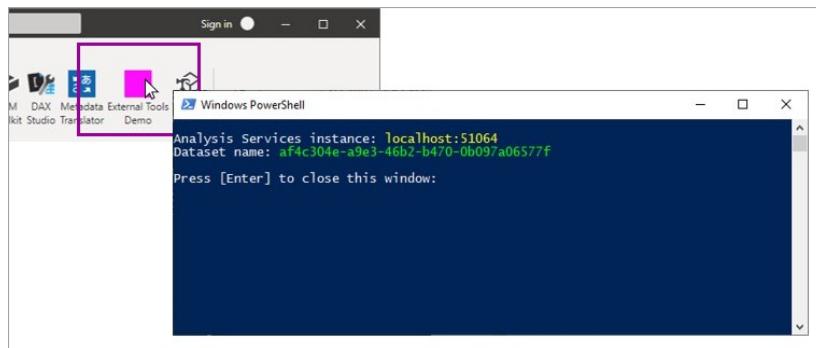
Example

The following `*.pbitool.json` file launches `powershell.exe` from the External Tools ribbon and runs a script called `pbiToolsDemo.ps1`, passing the server name and port number in the `-Server` parameter and the dataset name in the `-Database` parameter.

```
{  
    "version": "1.0.0",  
    "name": "External Tools Demo",  
    "description": "Launches PowerShell and runs a script that outputs server and database parameters.  
(Requires elevated PowerShell permissions.)",  
    "path": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",  
    "arguments": "C:\\pbiToolsDemo.ps1 -Server \"%server%\" -Database \"%database%\"",  
    "iconData":  
        "image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAFcSJAAAAAXNSR0IAr4c6QAAAARnQU1BAACxjwv8YQUAAAJcE  
hZcwAADsEAAA7BAbiRa+0AAAANSURBVhXY/jH9+8/AAcIAwpql70kAAAAAE1FTkSuQmC"  
}
```

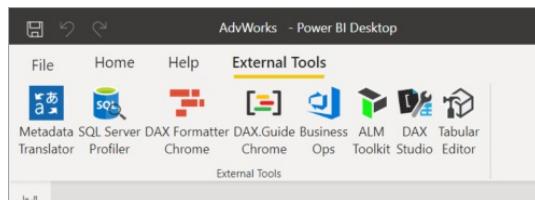
The corresponding `pbiToolsDemo.ps1` script outputs the Server and Database parameters to the console.

```
[CmdletBinding()]  
param  
(  
    [Parameter(Mandatory = $true)]  
    [string] $Server,  
    [Parameter(Mandatory = $true)]  
    [string] $Database  
)  
Write-Host ""  
Write-Host "Analysis Services instance: " -NoNewline  
Write-Host "$Server" -ForegroundColor Yellow  
Write-Host "Dataset name: " -NoNewline  
Write-Host "$Database" -ForegroundColor Green  
Write-Host ""  
Read-Host -Prompt 'Press [Enter] to close this window'
```



Icon data URIs

To include an icon in the External Tools ribbon, the pbbitool.json registration file must include an iconData element.



The iconData element takes a data URI without the **data:** prefix. For example, the data URI of a one pixel magenta png image is:

```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAAAABCAYAAAAFFcSJAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAJcEhZcwAADsEAA7Bab1Ra+0AAAANSURBVbXY/jH9+
```

Be sure to remove the **data:** prefix, as shown in the pbbitool.json example above.

To convert a .png or other image file type to a data URI, use an online tool or a custom tool such as the one shown in the following C# code snippet:

```
string ImageDataUri;
OpenFileDialog openFileDialog1 = new OpenFileDialog();
openFileDialog1.Filter = "PNG Files (.png)|*.png|All Files (*.*)|*.*";
openFileDialog1.FilterIndex = 1;
openFileDialog1.Multiselect = false;
openFileDialog1.CheckFileExists = true;
bool? userClickedOK = openFileDialog1.ShowDialog();
if (userClickedOK == true)
{
    var fileName = openFileDialog1.FileName;
    var sb = new StringBuilder();
    sb.Append("image/");
    .Append((System.IO.Path.GetExtension(fileName) ?? "png").Replace(".", ""));
    .Append(";base64,");
    .Append(Convert.ToBase64String(File.ReadAllBytes(fileName)));
    ImageDataUri = sb.ToString();
}
```

See also

- [External tools in Power BI Desktop](#)
- [Client libraries for connecting to Analysis Services](#)
- [Tabular Object Model \(TOM\)](#)

Using the Field list in Power BI Desktop

3/30/2022 • 3 minutes to read • [Edit Online](#)

The **Field** lists are being unified across Model view, Data view and Report view in Power BI Desktop. Unifying these views will create consistency for functionality and the user interface (UI) across views, and addresses customer feedback.

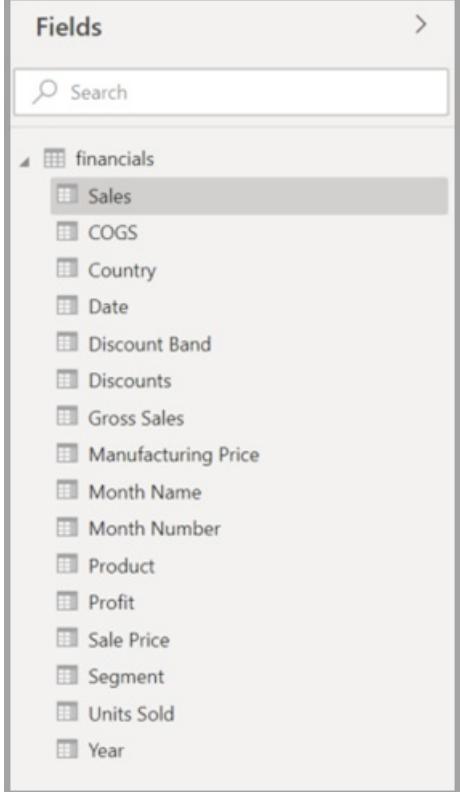
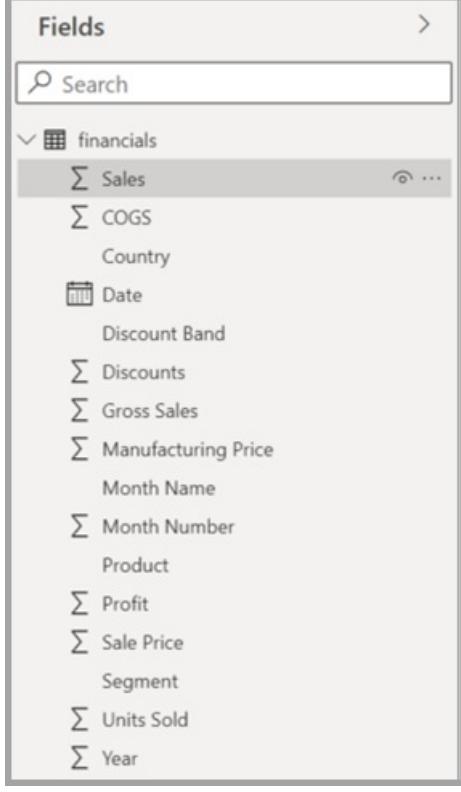
Changes you'll notice across views include the following:

- Iconography
- Search functionality
- Context menu items
- Similar drag-drop behavior
- Tooltips
- Accessibility improvements

The intent is to improve Power BI Desktop usability. The changes should have minimal impact on your typical data workflow.

Field list changes

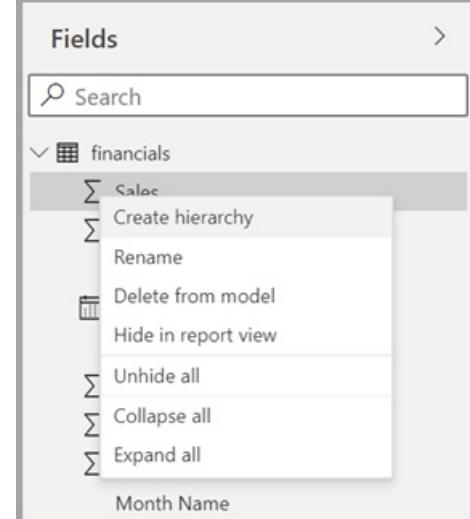
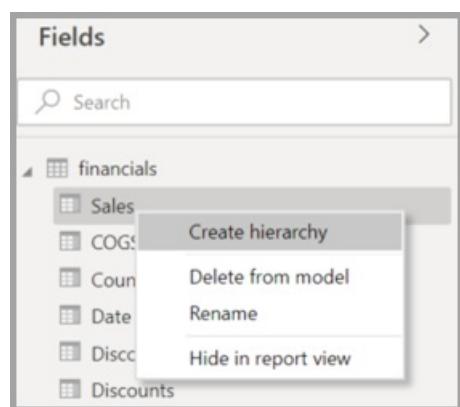
The following tables show the field list updates.

ORIGINAL FIELD LIST (MODEL VIEW)	NEW FIELD LIST (MODEL VIEW)
Original	New
Icons and UI	
	

ORIGINAL FIELD LIST (MODEL VIEW)

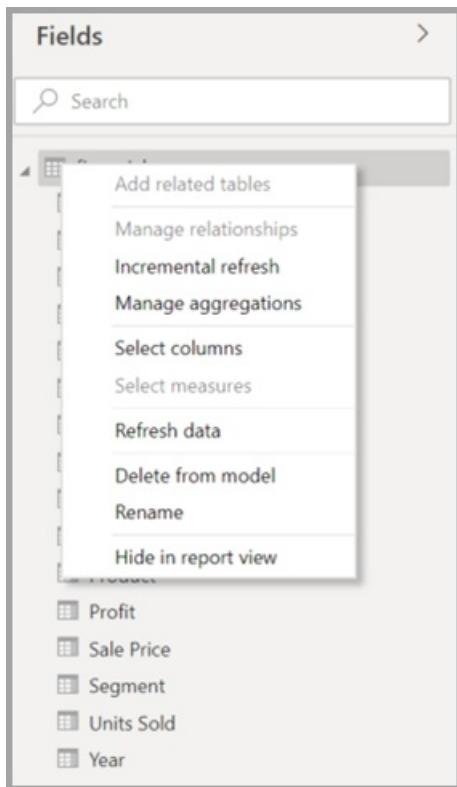
NEW FIELD LIST (MODEL VIEW)

Context menu - Field

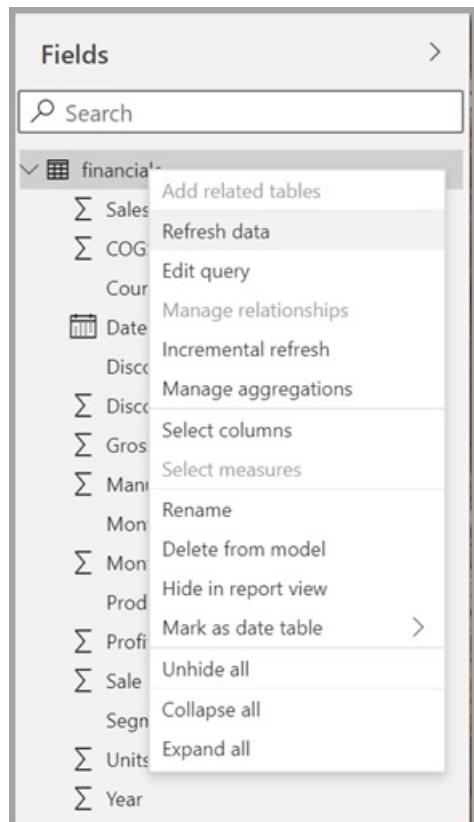


Context menu - Table

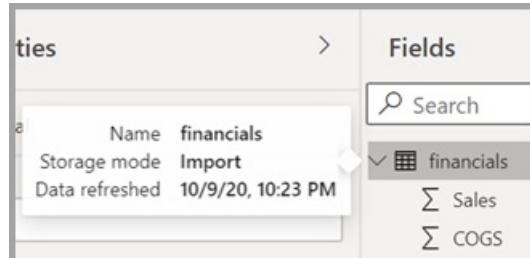
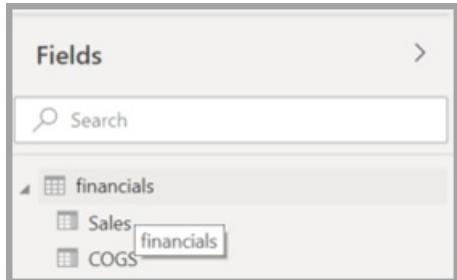
ORIGINAL FIELD LIST (MODEL VIEW)



NEW FIELD LIST (MODEL VIEW)



Tooltips



Field list icons

There are new Field list icons as well. The following table shows the original icons and their new equivalent, and provides a brief description of each.

ORIGINAL ICON	NEW ICON	DESCRIPTION
		Folder in the Fields list
		Numeric field: Numeric fields are aggregates that can be summed or averaged, for example. Aggregates are imported with the data and defined in the data model your report is based on. For more information, see Aggregates in Power BI reports .

ORIGINAL ICON	NEW ICON	DESCRIPTION
		Calculated column with a non-numeric data type: A new non-numeric column you create with a Data Analysis Expressions (DAX) formula that defines the column's values. Read more about calculated columns .
		Numeric calculated column: A new column you create with a Data Analysis Expressions (DAX) formula that defines the column's values. Read more about calculated columns .
		Measure: A measure has its own hard-coded formula. Report viewers can't change the calculation, for example, if it's a sum, it can only be a sum. The values aren't stored in a column. They're calculated on the fly, depending solely on their location in a visual. For more information, read Understanding measures .
		Measure group.
		KPI: A visual cue that communicates the amount of progress made toward a measurable goal. Read more about Key Performance Indicator (KPI) visuals .
		Hierarchy of fields: Select the arrow to see the fields that make up the hierarchy. Watch this Power BI video on YouTube about Creating and working with hierarchies for more information.
		Geo data: These location fields can be used to create map visualizations.
		Identity field: Fields with this icon are unique fields, set to show all values, even if they have duplicates. For example, your data might have records for two different people named 'Robin Smith', and each will be treated as unique. They won't be summed.
		Parameter: Set parameters to make parts of your reports and data models (such as a query filter, a data source reference, a measure definition, etc.) depend on one or more parameter values. See this Power BI blog post about query parameters for more information.

ORIGINAL ICON	NEW ICON	DESCRIPTION
		Calendar date field with a built-in date table.
		Calculated table: A table created with a Data Analysis Expressions (DAX) formula based on data already loaded into the model. These are best used for intermediate calculations and you want to store as part of the model.
		Warning: A calculated field with an error. For example, the syntax of the DAX expression might be incorrect.
		Group: Values in this column are based on grouping values from another column, by using the groups and bins feature. You can read how to Use grouping and binning .
no original icon		Change detection measure: When you configure a page for automatic page refresh, you can configure a change detection measure that is queried to determine if the rest of a page's visuals should be updated.

Next steps

You might also be interested in the following articles:

- [Create calculated columns in Power BI Desktop](#)
- [Use grouping and binning in Power BI Desktop](#)
- [Use gridlines and snap-to-grid in Power BI Desktop reports](#)

Formula editor in Power BI Desktop

3/30/2022 • 2 minutes to read • [Edit Online](#)

The formula editor (often referred to as the DAX editor) includes robust editing and shortcut enhancements to make authoring and editing formulas easy and intuitive.

Using the formula editor

You can use the following keyboard shortcuts to increase your productivity and to streamline creating formulas in the formula editor.

KEYBOARD COMMAND	RESULT
Ctrl+C	Copy line (empty selection)
Ctrl+G	Go to line...
Ctrl+I	Select current line
Ctrl+M	Toggle Tab moves focus
Ctrl+U	Undo last cursor operation
Ctrl+X	Cut line (empty selection)
Shift+Enter	Insert line below
Ctrl+Shift+Enter	Insert line above
Ctrl+Shift+\	Jump to matching bracket
Ctrl+Shift+K	Delete line
Ctrl+] / [Indent/outdent line
Ctrl+Home	Go to beginning of file
Ctrl+End	Go to end of file
Ctrl+↑ / ↓	Scroll line up/down
Ctrl+Shift+Alt + (arrow key)	Column (box) selection
Ctrl+Shift+Alt + PgUp/PgDn	Column (box) selection page up/down
Ctrl+Shift+L	Select all occurrences of current selection
Ctrl+Alt+ ↑ / ↓	Insert cursor above / below

KEYBOARD COMMAND	RESULT
Ctrl+F2	Select all occurrences of current word
Shift+Alt + (drag mouse)	Column (box) selection
Shift+Alt + ↓ / ↑	Copy line up/down
Shift+Alt+→	Expand selection
Shift+Alt+←	Shrink selection
Shift+Alt+I	Insert cursor at end of each line selected
Alt+↑ / ↓	Move line up/down
Alt+PgUp / PgDn	Scroll page up/down
Alt+Click	Insert cursor
Home / End	Go to beginning/end of line

Next steps

The following articles provide more information about formulas and DAX in Power BI Desktop.

- [DAX basics in Power BI Desktop](#)
- [DAX in Power BI Desktop Microsoft Learn course](#)
- [DAX reference](#)