# POWER BI LAB - CONNECT TO DATA, QUERY DATA, DATA SHAPING

# THEORY SECTION

## Article: Dataset modes in Power BI

Article Source:  
 <https://learn.microsoft.com/en-us/power-bi/connect-data/service-dataset-modes-understand>

**In this article**

This article provides a technical explanation of Power BI dataset modes. The article emphasizes the rationale for each mode, and possible impacts on Power BI capacity resources.

The three dataset modes are:

* [Import](https://learn.microsoft.com/en-us/power-bi/connect-data/service-dataset-modes-understand#import-mode)
* [DirectQuery](https://learn.microsoft.com/en-us/power-bi/connect-data/service-dataset-modes-understand#directquery-mode)
* [Composite](https://learn.microsoft.com/en-us/power-bi/connect-data/service-dataset-modes-understand#composite-mode)

**Import mode**

*Import* mode is the most common mode used to develop datasets. This mode delivers fast performance thanks to in-memory querying. It also offers design flexibility to modelers, and support for specific Power BI service features (Q&A, Quick Insights, etc.). Because of these strengths, it's the default mode when creating a new Power BI Desktop solution.

It's important to understand that imported data is always stored on a disk. When queried or refreshed, the data must be fully loaded into memory of the Power BI capacity. Once in memory, Import models can then achieve very fast query results.

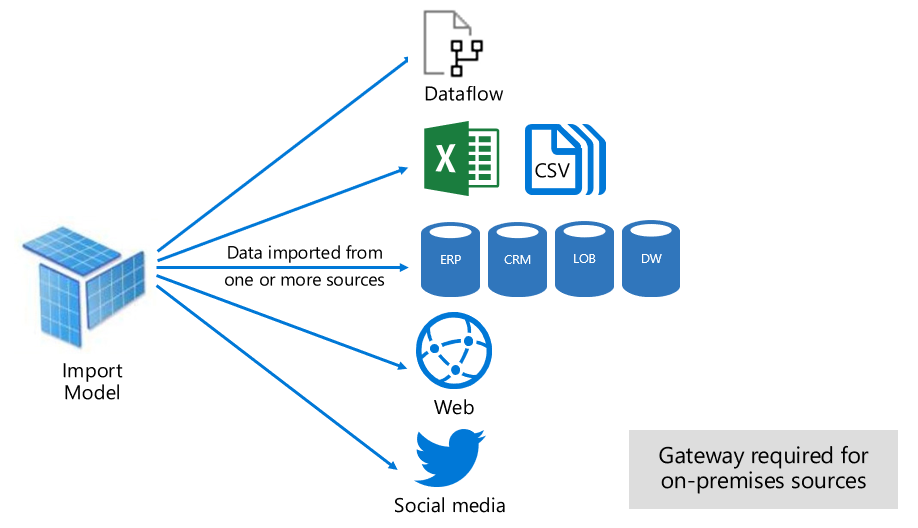
**Data Compression in Power BI**

When refreshed, data is compressed and optimized and then stored to disk by the **VertiPaq storage engine**. It is possible to see a 10% to 20% reduction from the compressed size.

Design flexibility can be achieved in three ways:

* Integrate data by caching data from dataflows, and external data sources, whatever the data source type or format.
* Use the entire set of [Power Query M formula language](https://learn.microsoft.com/en-us/powerquery-m/), referred to as *M*, functions when creating data preparation queries.
* Apply the entire set of [Data Analysis Expressions (DAX)](https://learn.microsoft.com/en-us/dax/) functions when enhancing the model with business logic. There's support for calculated columns, calculated tables, and measures.

As shown in the following image, an Import model can integrate data from any number of supported data source types.



However, while there are compelling advantages associated with Import models, there are disadvantages, too:

* The entire model must be loaded to memory before Power BI can query the model, which can place pressure on available capacity resources, especially as the number and size of Import models grow.
* Model data is only as current as the latest refresh, and so Import models need to be refreshed, usually on a scheduled basis.
* A full refresh removes all data from all tables and reloads it from the data source. This operation can be expensive in terms of time and resources for the Power BI service, and the data sources.

**Incremental Refreshes of Data**

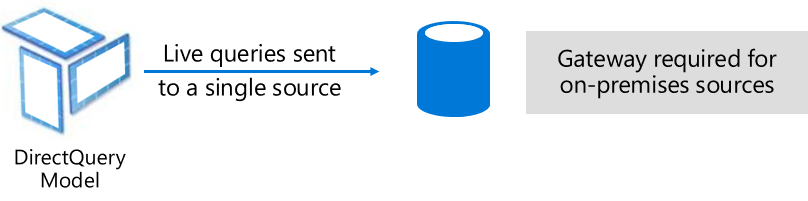
Power BI can achieve incremental refresh to avoid truncating and reloading entire tables. For more information, including supported plans and licensing, see [**Incremental refresh and real-time data for datasets**](https://learn.microsoft.com/en-us/power-bi/connect-data/incremental-refresh-overview).

From a Power BI service resource perspective, Import models require:

* Sufficient memory to load the model when it's queried or refreshed.
* Processing resources and extra memory resources to refresh data.

**DirectQuery mode**

*DirectQuery* mode is an alternative to Import mode. Models developed in DirectQuery mode don't import data. Instead, they consist only of metadata defining the model structure. When the model is queried, native queries are used to retrieve data from the underlying data source.



There are two main reasons to consider developing a DirectQuery model:

* When data volumes are too large, even when [data reduction methods](https://learn.microsoft.com/en-us/power-bi/guidance/import-modeling-data-reduction) are applied, to load into a model, or practically refresh.
* When reports and dashboards need to deliver *near real-time* data, beyond what can be achieved within scheduled refresh limits. Scheduled refresh limits are eight times a day for shared capacity, and 48 times a day for a Premium capacity.

There are several advantages associated with DirectQuery models:

* Import model size limits don't apply.
* Models don't require scheduled data refresh.
* Report users see the latest data when interacting with report filters and slicers. Also, report users can refresh the entire report to retrieve current data.
* Real-time reports can be developed by using the [Automatic page refresh](https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-automatic-page-refresh) feature.
* Dashboard tiles, when based on DirectQuery models, can update automatically as frequently as every 15 minutes.

However, there are some limitations associated with DirectQuery models:

* Power Query/Mashup expressions can only be functions that can be transposed to native queries understood by the data source.
* DAX formulas are limited to use only functions that can be transposed to native queries understood by the data source. Calculated tables aren't supported.
* Quick Insights features aren't supported.

From a Power BI service resource perspective, DirectQuery models require:

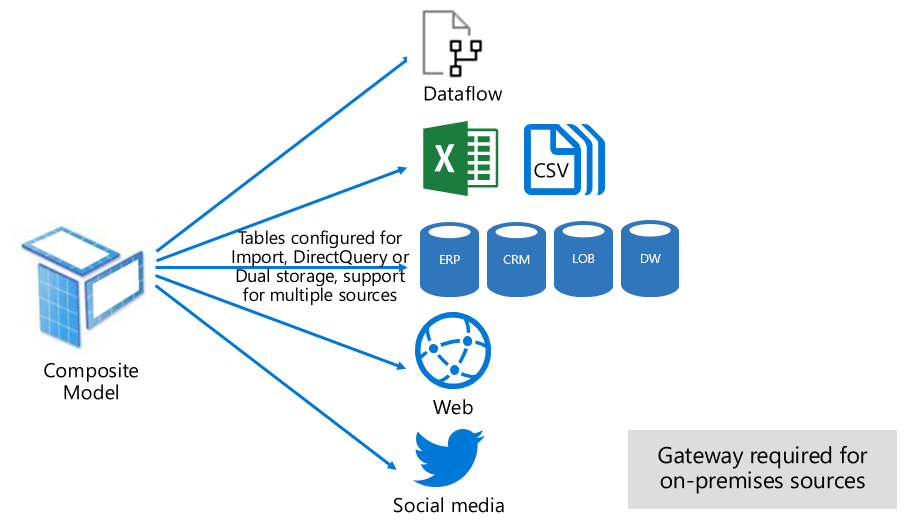
* Minimal memory to load the model (metadata only) when it's queried.
* Sometimes the Power BI service must use significant processor resources to generate and process queries sent to the data source. When this situation arises, it can affect throughput, especially when concurrent users are querying the model.

For more information, see [Use DirectQuery in Power BI Desktop](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-use-directquery).

**Composite mode**

*Composite* mode can mix Import and DirectQuery modes, or integrate multiple DirectQuery data sources. Models developed in Composite mode support configuring the storage mode for each model table. This mode also supports calculated tables, defined with DAX.

The table storage mode can be configured as Import, DirectQuery, or Dual. A table configured as Dual storage mode is both Import and DirectQuery, and this setting allows the Power BI service to determine the most efficient mode to use on a query-by-query basis.



Composite models strive to deliver the best of Import and DirectQuery modes. When configured appropriately, they can combine the high query performance of in-memory models with the ability to retrieve near real-time data from data sources. For more information, see [Use composite models in Power BI Desktop](https://learn.microsoft.com/en-us/power-bi/transform-model/desktop-composite-models).

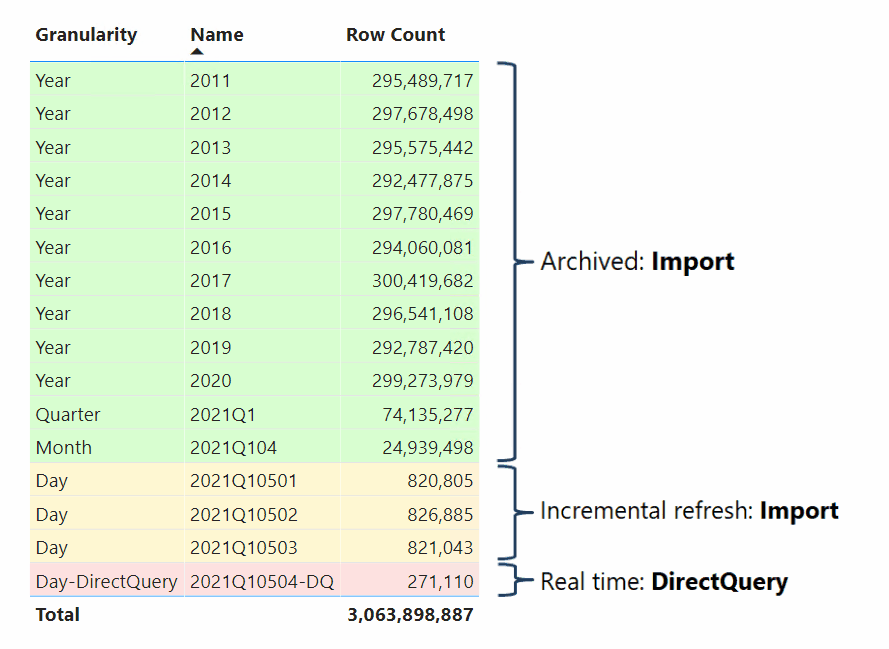
**Pure Import and DirectQuery tables**

Data modelers who develop Composite models are likely to configure dimension-type tables in Import or Dual storage mode, and fact-type tables in DirectQuery mode. For more information about model table roles, see [Understand star schema and the importance for Power BI](https://learn.microsoft.com/en-us/power-bi/guidance/star-schema).

For example, consider a model with a **Product** dimension-type table in Dual mode, and a **Sales** fact-type table in DirectQuery mode. The **Product** table could be efficiently and quickly queried from in-memory to render a report slicer. The **Sales** table could also be queried in DirectQuery mode with the related **Product** table. The latter query could enable the generation of a single efficient native SQL query that joins **Product** and **Sales** tables, and filters by the slicer values.

**Hybrid tables**

Data modelers who develop Composite models can also configure fact tables as hybrid tables. A hybrid table is a table with one or multiple Import partitions and one DirectQuery partition. The advantage of a hybrid table is it could be efficiently and quickly queried from in-memory while at the same time including the latest data changes from the data source that occurred after the last import cycle, as the following visualization illustrates.



The easiest way to create a hybrid table is to configure an incremental refresh policy in Power BI Desktop and enable the option **Get the latest data in real time with DirectQuery (Premium only)**. When Power BI applies an incremental refresh policy that has this option enabled, it partitions the table like the partitioning scheme displayed in the previous diagram. To ensure good performance, configure your dimension-type tables in Dual storage mode so that Power BI can generate efficient native SQL queries when querying the DirectQuery partition.

**Note**

Power BI supports hybrid tables only when the dataset is hosted in workspaces on Premium capacities. Accordingly, you must upload your dataset to a Premium workspace if you configure an incremental refresh policy with the option to get the latest data in real time with DirectQuery. For more information, see [**Incremental refresh and real-time data for datasets**](https://learn.microsoft.com/en-us/power-bi/connect-data/incremental-refresh-overview).

It's also possible to convert an Import table to a hybrid table by adding a DirectQuery partition using Tabular Model Scripting Language (TMSL) or the Tabular Object Model (TOM) or by using a third-party tool. For example, you can partition a fact table such that the bulk of the data is left in the data warehouse while only a fraction of the most recent data is imported. This approach can help to optimize performance if the bulk of this data is historical data that is infrequently accessed. A hybrid table can have multiple Import partitions, but only one DirectQuery partition.

# LAB SECTION

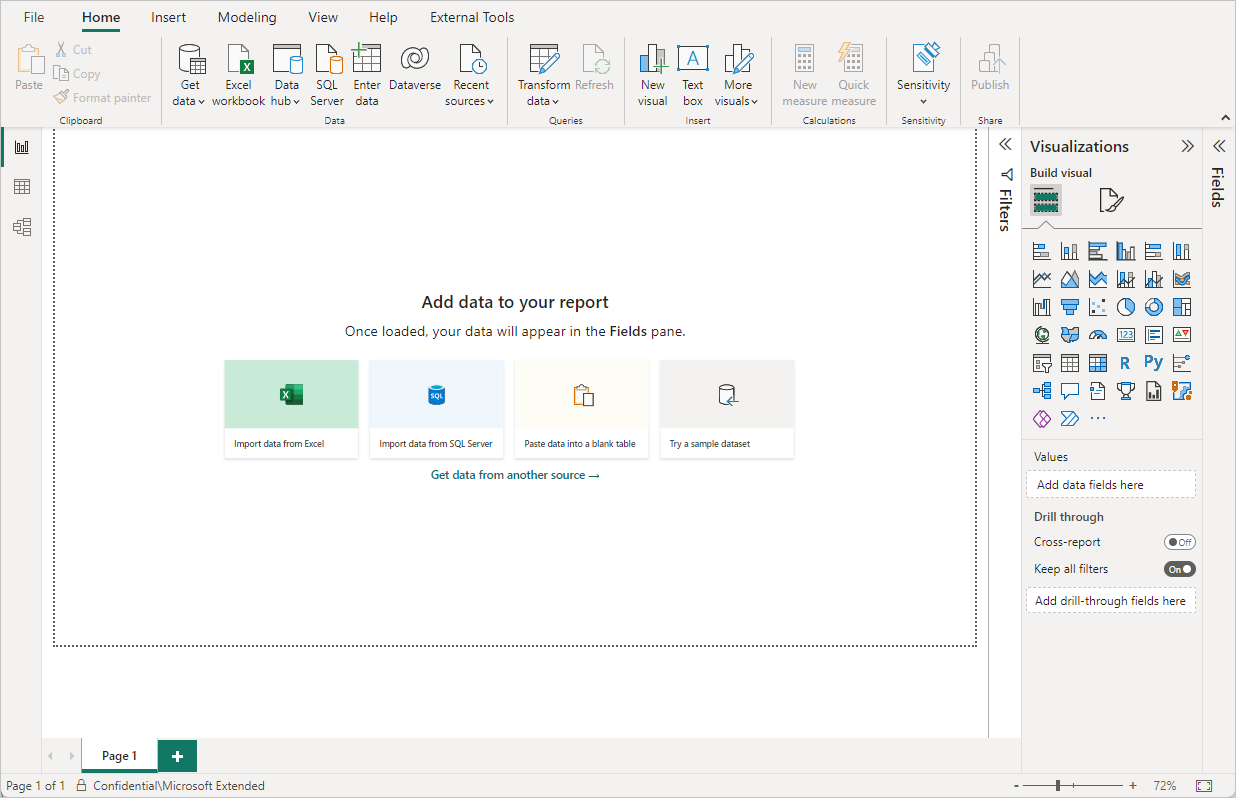
# LAB 1A: Connect to data in Power BI Desktop

LAB Source: <https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-quickstart-connect-to-data>

In this lab, you connect to data using Power BI Desktop, which is the first step in building data models and creating reports.

## Launch Power BI Desktop

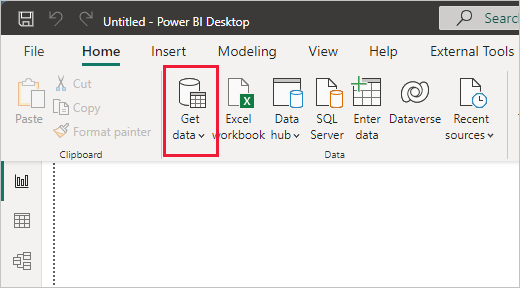
Once you install Power BI Desktop, launch the application so it's running on your local computer. You're presented with a Power BI tutorial. Follow the tutorial or close the dialog to start with a blank canvas. The canvas is where you create visuals and reports from your data.



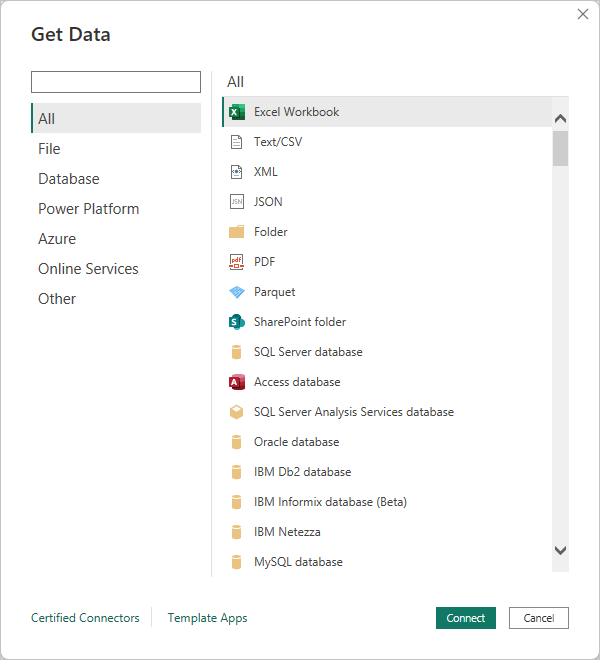
## Connect to data

With Power BI Desktop, you can connect to many different types of data. These sources include basic data sources, such as a Microsoft Excel file. You can connect to online services that contain all sorts of data, such as Salesforce, Microsoft Dynamics, Azure Blob Storage, and many more.

To connect to data, from the **Home** ribbon select **Get data**.



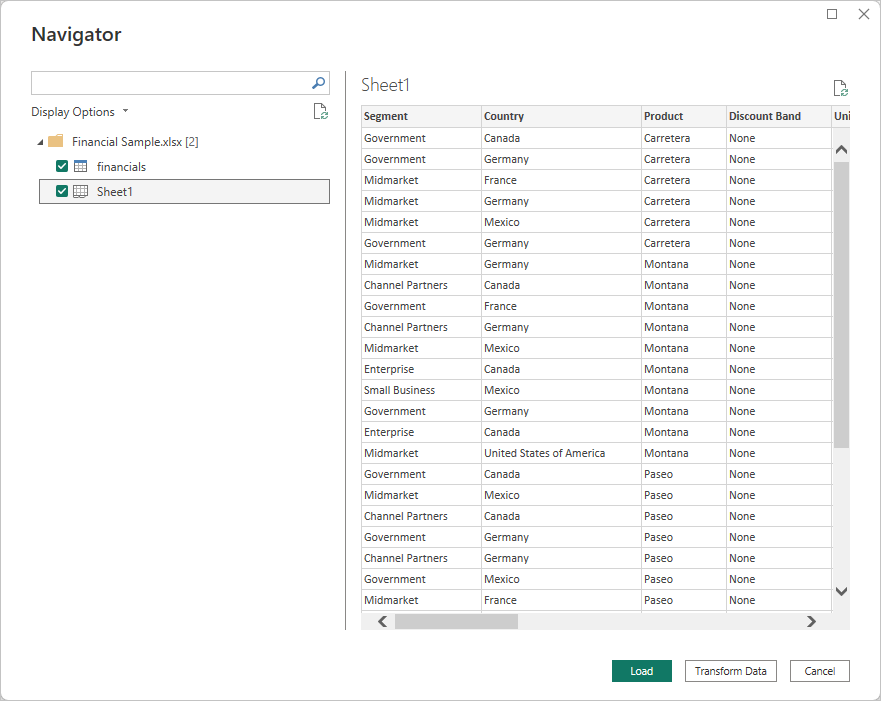
The **Get Data** window appears. You can choose from the many different data sources to which Power BI Desktop can connect. Since this data source is an Excel file, select **Excel** from the **Get Data** window, then select the **Connect** button.



Power BI prompts you to provide the location of the Excel file to which to connect. The file is called Financial Sample and it is available in datasets folder. Select that file, and then select **Open**.

## Load Data

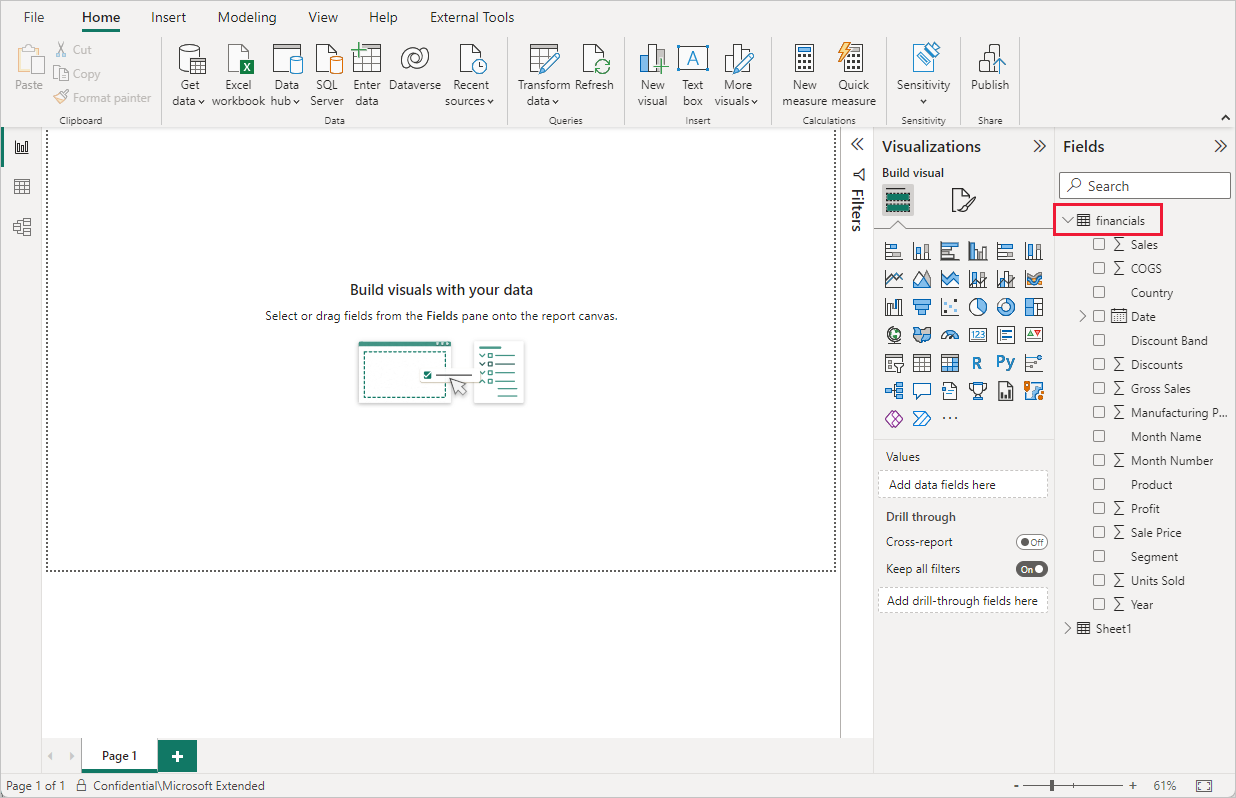
Power BI Desktop then loads the workbook, reads its contents, and shows you the available data in the file using the **Navigator** window. In that window, you can choose which data you would like to load into Power BI Desktop. Select the tables by marking the checkboxes beside each table you want to import. Import both available tables.



Once you've made your selections, select **Load** to import the data into Power BI Desktop.

## View data in the Fields pane

Once you've loaded the tables, the **Fields** pane shows you the data. You can expand each table by selecting the arrow beside its name. In the following image, the financials table is expanded, showing each of its fields.



And that's it! You've connected to data in Power BI Desktop, loaded that data, and now you can see all the available fields within those tables.

# LAB 1B: Shape and combine data in Power BI Desktop

LAB Source:  
<https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-shape-and-combine-data>

**In this tutorial**

1. [Shaping/Transforming data](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-shape-and-combine-data#shape-data)
2. [Adjust the data](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-shape-and-combine-data#adjust-the-data)
3. [Combine data](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-shape-and-combine-data#combine-data)
4. [Combine queries](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-shape-and-combine-data#combine-queries)
5. [Next steps](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-shape-and-combine-data#next-steps)

With Power BI Desktop, you can connect to many different types of data sources, then shape the data to meet your needs, enabling you to create visual reports to share with others. *Shaping* data means transforming the data: renaming columns or tables, changing text to numbers, removing rows, setting the first row as headers, and so on. *Combining* data means connecting to two or more data sources, shaping them as needed, then consolidating them into a single query.

In this tutorial, you'll learn how to:

* Shape data by using Power Query Editor.
* Connect to different data sources.
* Combine those data sources and create a data model to use in reports.

This tutorial demonstrates how to shape a query by using Power BI Desktop, highlighting the most common tasks. The query used here is described in more detail, including how to create the query from scratch, in [Getting Started with Power BI Desktop](https://learn.microsoft.com/en-us/power-bi/fundamentals/desktop-getting-started).

Power Query Editor in Power BI Desktop uses the right-click menus, and the **Transform** ribbon. Most of what you can select in the ribbon is also available by right-clicking an item, such as a column, and choosing from the menu that appears.

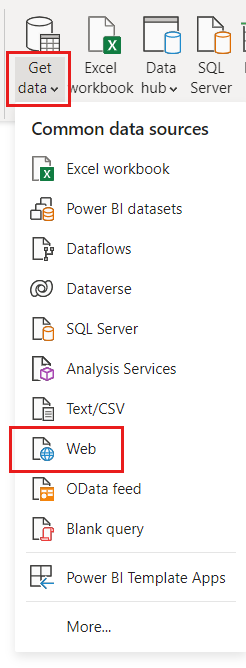
## Shape data

To shape data in Power Query Editor, you provide step-by-step instructions for Power Query Editor to adjust the data as it loads and presents the data. The original data source isn't affected; only this particular view of the data is adjusted, or *shaped*.

The steps you specify (such as rename a table, transform a data type, or delete a column) are recorded by Power Query Editor. Each time this query connects to the data source, Power Query Editor carries out those steps so that the data is always shaped the way you specify. This process occurs whenever you use Power Query Editor, or for anyone who uses your shared query, such as on the Power BI service. Those steps are captured, sequentially, in the **Query Settings** pane, under **APPLIED STEPS**. We’ll go through each of those steps in this article.

## Import the data from a web source.

1. Select the **Get data** dropdown, then choose **Web**.



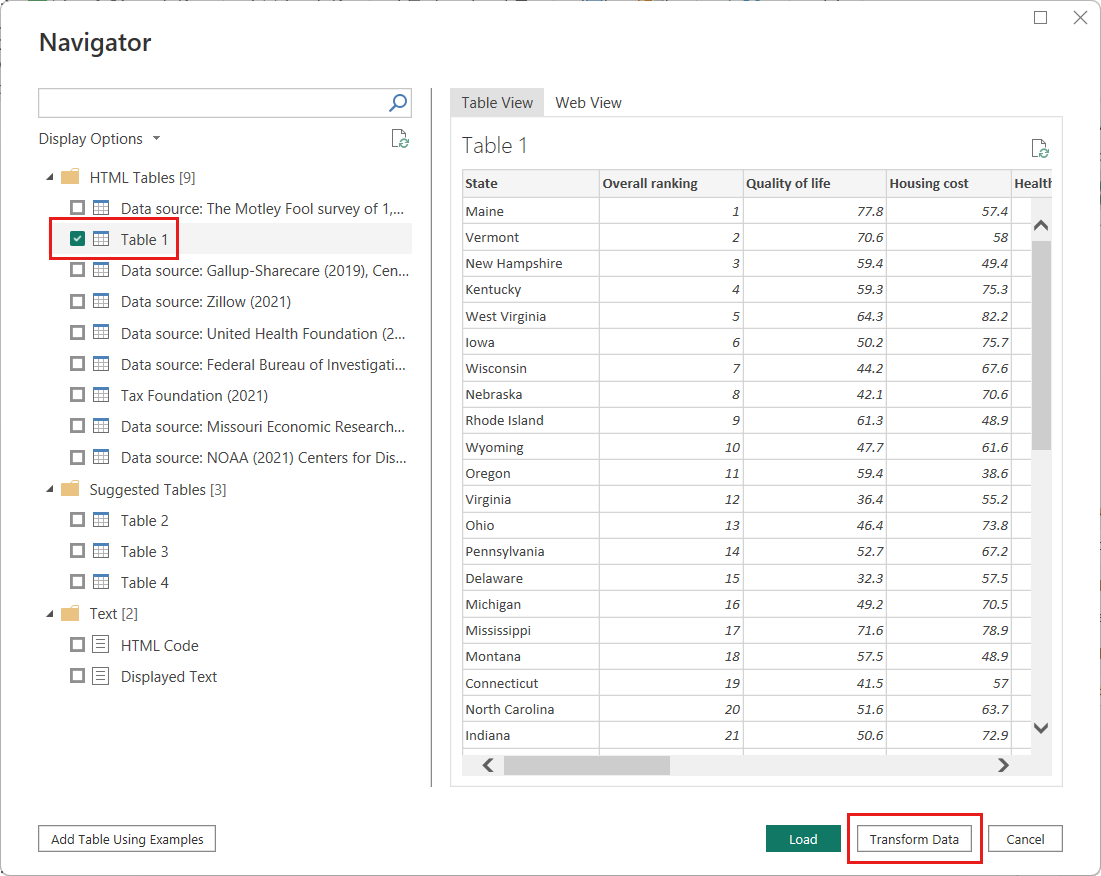
Paste this URL into the **From Web** dialog and select **OK** and then click **Connect**.

<https://www.fool.com/research/best-states-to-retire>

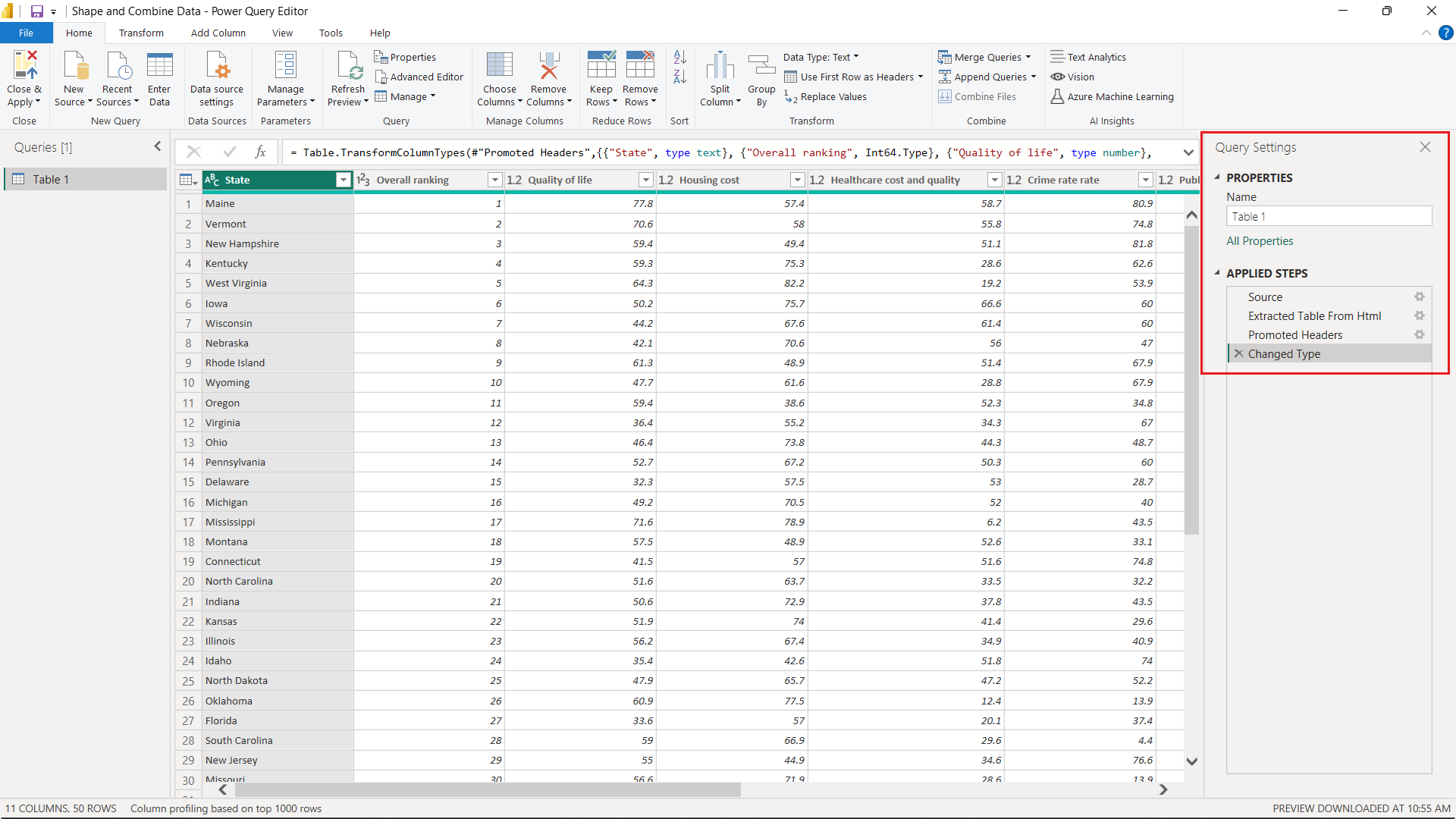


## Transform Data Using Power Query

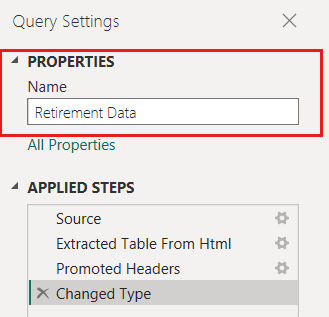
1. In the **Navigator** dialog, select Table 1, then choose **Transform Data**.



1. The Power Query Editor window opens. You can see the default steps applied so far, in the **Query Settings** pane under **APPLIED STEPS**.
   * **Source**: Connecting to the website.
   * **Extracted Table from Html**: Selecting the table.
   * **Promoted Headers**: Changing the top row of data into column headers.
   * **Changed Type**: Changing the column types, which are imported as text, to their inferred types.

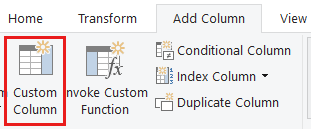
[](https://learn.microsoft.com/en-us/power-bi/connect-data/media/desktop-shape-and-combine-data/power-query-editor-query-settings-dialog.png#lightbox)

1. At the upper right part of the screen, change the table name from the default "Table 1" to "Retirement Data." Then press **Enter**.



## Add Custom Columns

1. Right now, the data is ordered by a weighted score, as described on the source web page under [Methodology](https://www.fool.com/research/best-states-to-retire/#:%7E:text=Methodology). Let's add a custom column to calculate a different score for ranking. We'll then sort the table on this column to compare the custom score's ranking to the existing **Rank**.
2. From the **Add Column** ribbon (menu item), select **Custom Column**.



1. In the **Custom Column** dialog, in **New column name**, enter *New score*. For the **Custom column formula**, enter the following data:

( [Quality of life] + [Housing cost] + [Healthcare cost and quality] + [Crime rate] + [#"Public health/Covid response"] + [Taxes] + [Cost of living] + [Weather] ) / 8

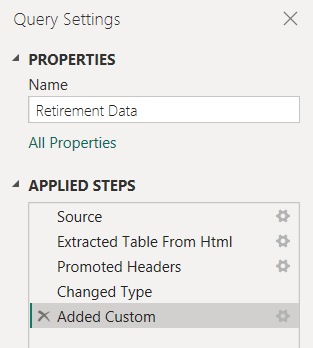
## Working with Column formulas

1. Make sure the status message is *No syntax errors have been detected*, and select **OK**.

A screenshot of a computer

Description automatically generated

1. In **Query Settings**, the **APPLIED STEPS** list now shows the new **Added Custom** step we just defined.



## Adjusting data

Before we work with this query, let's make a few changes to adjust its data:

## Remove columns

For example, assume **Weather** isn't a factor in our results. Removing this column from the query doesn't affect the other data.

## Fix errors.

Because we removed a column, we need to adjust our calculations in the **New score** column by changing its formula.

## Sort data.

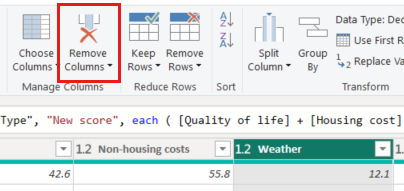
Sort the data based on the **New score** column, and compare to the existing **Rank** column.

## Replace data.

We'll highlight how to replace a specific value and how to insert an applied step.

These changes are described in the following steps.

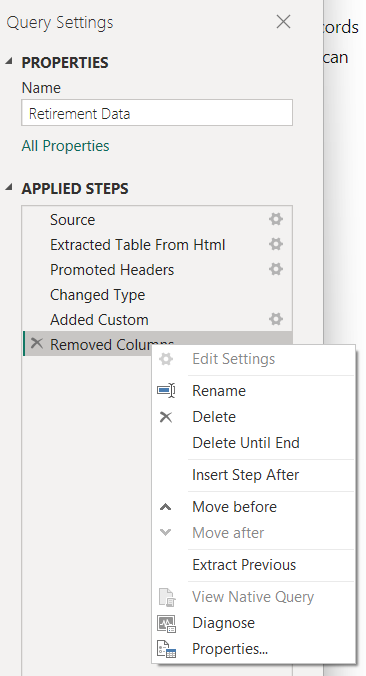
1. To remove the **Weather** column, select the column, choose the **Home** tab from the ribbon, and then choose **Remove Columns**. Alternatively**, right click** on the column and select **Remove**.



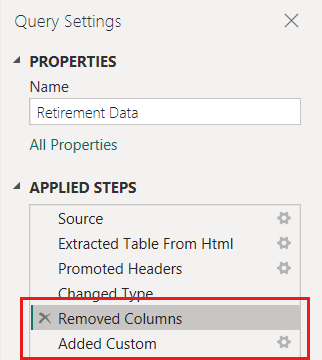
**Note**

The **New score** values haven't changed, due to the ordering of the steps. Power Query Editor records the steps sequentially, yet independently, of each other. To apply actions in a different sequence, you can move each applied step up or down.

1. Right-click a step to see its context menu.



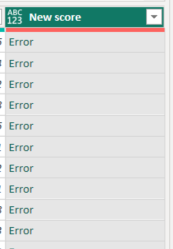
1. Move up the last step, **Removed Columns**, to just above the **Added Custom** step.



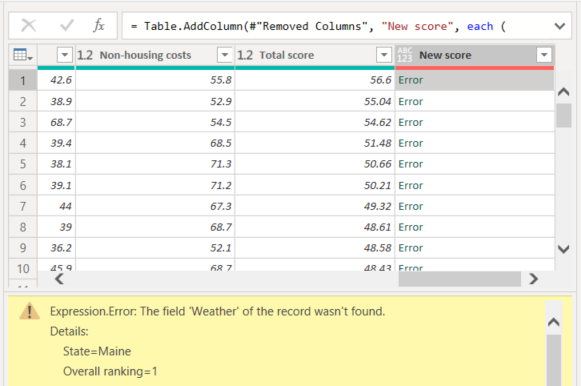
## Handling Errors in Formulas

1. Select the **Added Custom** step.

Notice the **New score** column now shows *Error* rather than the calculated value.



To get more information about this error, click on the blank space of any of the cells (not the word "Error" directly, and Power Query Editor displays the error information at the bottom of the window.



If you click on the word *Error* directly, Power Query Editor creates an **Applied Step** in the **Query Settings** pane and displays information about the error. Because we don't need to display error information anywhere else, select **Cancel**.

1. To fix the errors, there are two changes needed, removing the *Weather* column name and changing the divisor from 8 to 7.

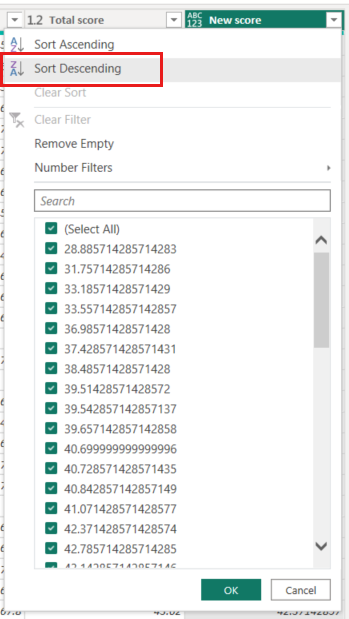
Right-click the **Added Custom** step and select **Edit Settings**. This brings up the **Custom Column** dialog you used to create the **New score** column. Edit the formula as described previously, until it looks like this:

A screenshot of a computer

Description automatically generated

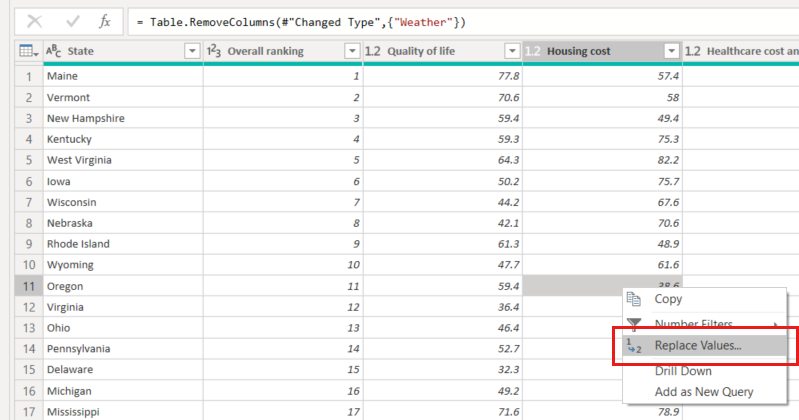
## Sorting Data

1. Sort the data based on the **New score** column. Select the drop-down located next to the **New score** column header and choose **Sort Descending**.



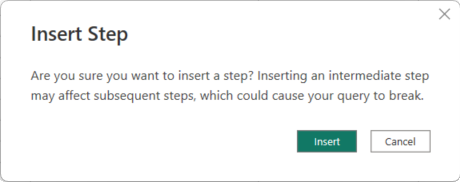
The data is now sorted according to **New score**. You can select an applied step anywhere in the list, and continue shaping the data at that point in the sequence. Power Query Editor automatically inserts a new step directly after the currently selected applied step.

1. In **APPLIED STEPS**, select the step preceding the custom column, which is the **Removed Columns** step. Here we'll replace the value of the **Housing cost** ranking in Oregon. Right-click the appropriate cell that contains Oregon's **Housing cost** value, and then select **Replace Values**. Note which **Applied Step** is currently selected.



1. Select **Insert**.

Because we're inserting a step, Power Query Editor reminds us that subsequent steps could cause the query to break.



1. Change the data value to *100.0*.

Power Query Editor replaces the data for Oregon. When you create a new applied step, Power Query Editor names it based on the action, in this case, **Replaced Value**. If you have more than one step with the same name in your query, Power Query Editor appends an increasing number to each subsequent applied step's name.

1. Select the step **Sorted Rows**.

Notice the data has changed regarding Oregon's new ranking. This change occurs because we inserted the **Replaced Value** step in the correct location, before the **Added Custom** step.

We’ve now shaped our data to the extent we need to. Next let’s connect to another data source, and combine data.

## Combining datasets

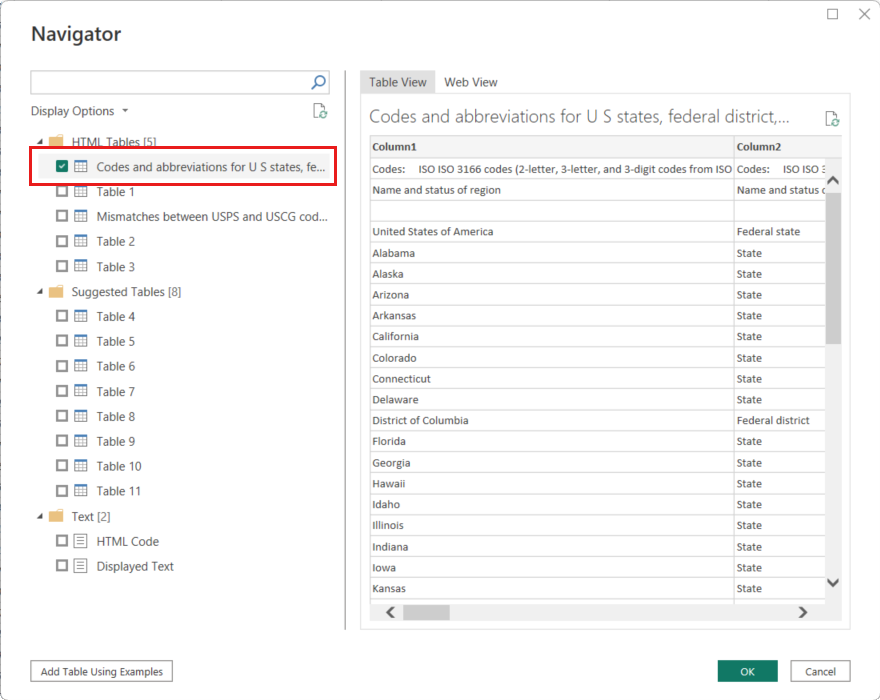
The data about various states is interesting and will be useful for building further analysis efforts and queries. However, most data about states uses a two-letter abbreviation for state codes, not the full name of the state. We need a way to associate state names with their abbreviations.

## Add a new dataset from the web

There's another public data source that provides that association, but it needs a fair amount of shaping before we can connect it to our retirement table. To shape the data, follow these steps:

1. From the **Home** ribbon in Power Query Editor, select **New Source > Web**.
2. Enter the address of the website for state abbreviations, <https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations> , and then select **Connect**.

The Navigator displays the content of the website.



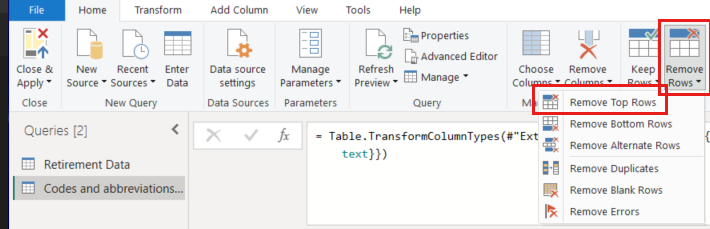
1. Select **Codes and abbreviations for U.S. states, federal district, territories, and other regions**.

**Tip**

It will take a bit of shaping to pare this table’s data down to what we want. Is there a faster or easier way to accomplish the following steps? Yes, we could create a *relationship* between the two tables, and shape the data based on that relationship. The following example steps are helpful to learn for working with tables. However, relationships can help you quickly use data from multiple tables.

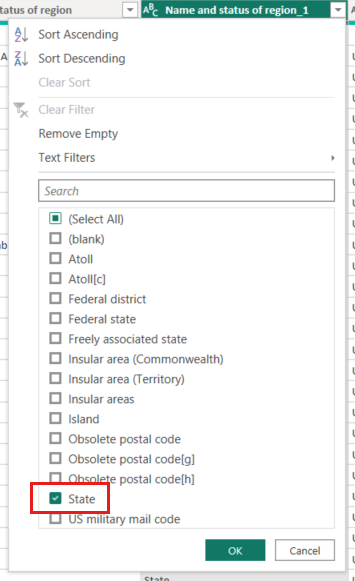
To get the data into shape, follow these steps:  
  
Removing Rows

1. Remove the top row. Because it's a result of the way that the web page’s table was created, we don’t need it. From the **Home** ribbon, select **Remove Rows > Remove Top Rows**.



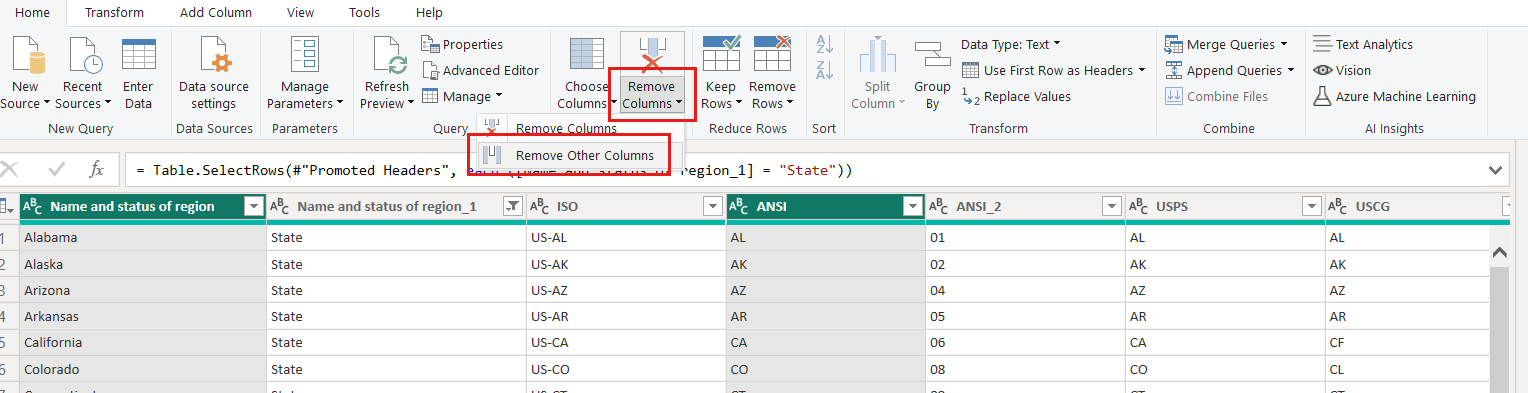
The **Remove Top Rows** dialog appears. Specify 1 row to remove.

1. Promote the new top row to headers with **Use First Row As Headers** from the **Home** tab, or from the **Transform** tab in the ribbon.
2. Because the **Retirement Data** table doesn't have information for Washington DC or territories, we need to filter them from our list.   
   Select the **Name and status of region\_1** column's drop-down, then clear all checkboxes except **State**.



## Removing Columns

1. Remove all unneeded columns. Because we need only the mapping of each state to its official two-letter abbreviation (**Name and status of region** and **ANSI** columns), we can remove the other columns.   
   First select the **Name and status of region** column, then hold down the **CTRL** key and select the **ANSI** column. At this point, two columns must be selected. From the **Home** tab on the ribbon, select **Remove Columns > Remove Other Columns**. Now, only two columns must exist in the data source.

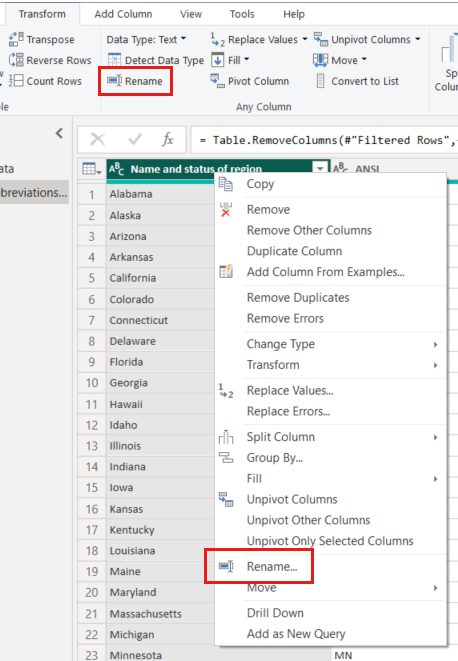
[](https://learn.microsoft.com/en-us/power-bi/connect-data/media/desktop-shape-and-combine-data/state-table-remove-other-columns.png#lightbox)

## The role of applied steps in Power Query Editor

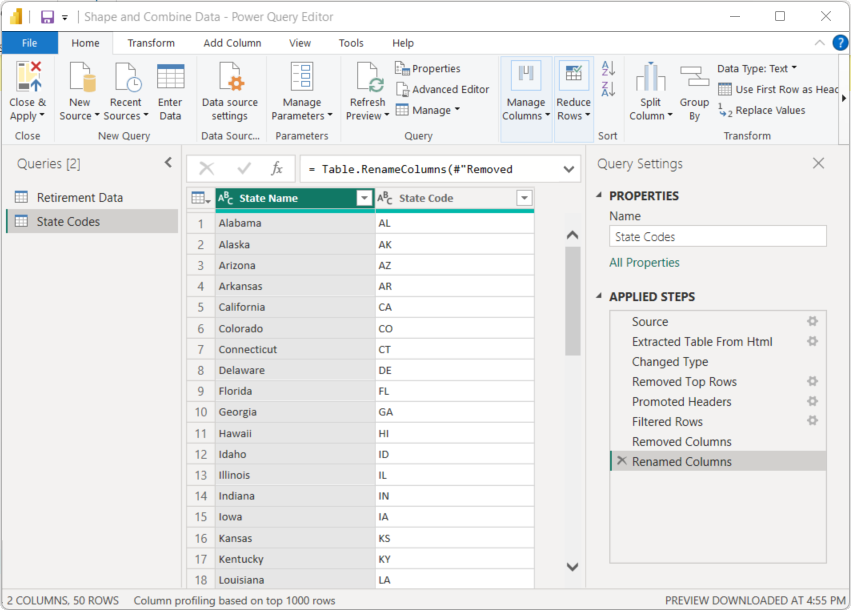
The *sequence* of applied steps in Power Query Editor is important, and affects how the data is shaped. It’s also important to consider how one step might impact another subsequent step. For example, if you remove a step from the applied steps, subsequent steps might not behave as originally intended.

## Rename Columns

1. Rename the columns and the table. There are a few ways to rename a column: First select the column, then either select **Rename** from the **Transform** tab on the ribbon, or right-click and select **Rename**. The following image shows both options, but you only need to choose one.



1. Rename the columns to *State Name* and *State Code*. To rename the table, enter the **Name** *State Codes* in the **Query Settings** pane.



## Combine queries

Now that we’ve shaped the *State Codes* table the way we want, let’s combine these two tables, or queries, into one. Because the tables we now have are a result of the queries we applied to the data, they’re often referred to as *queries*.

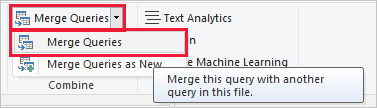
## Merging and Appending Query Data

There are two primary ways of combining queries: *merging* and *appending*.

* For one or more *columns* that you’d like to add to another query, you *merge* the queries.
* For one or more *rows* of data that you’d like to add to an existing query, you *append* the query.

In this case, we want to merge the queries:

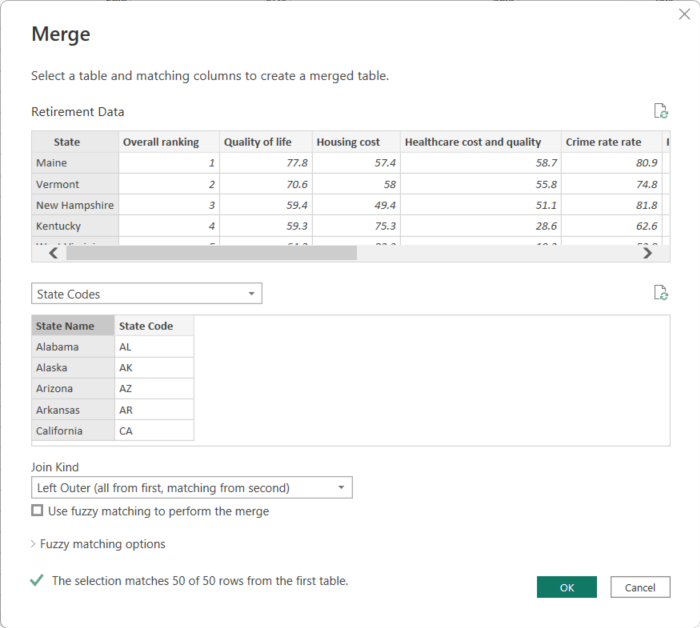
1. From the left pane of Power Query Editor, select the query *into which* you want the other query to merge. In this case, it's **Retirement Data**.
2. Select **Merge Queries > Merge Queries** from the **Home** tab on the ribbon.



The **Merge** window appears. It prompts you to select which table you'd like merged into the selected table, and the matching columns to use for the merge.

1. Select **State** from the *Retirement Data* table, then select the **State Codes** query.

When you select a matching columns, the **OK** button is enabled.

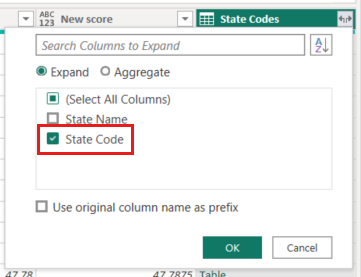
  
  
You might be prompted to set the privacy levels, to ensure the data is combined without including or transferring data you don't want transferred.

1. Select public for both data sources
2. Select **OK**.

Power Query Editor creates a new column at the end of the query, which contains the contents of the table (query) that was merged with the existing query. All columns from the merged query are condensed into this column, but you can **Expand** the table and include whichever columns you want.

1. To expand the merged table, and select which columns to include, select the expand icon ( ).

The **Expand** window appears.



1. In this case, we want only the **State Code** column. Select that column, clear **Use original column name as prefix**, and then select **OK**.

If we had left the checkbox selected for **Use original column name as prefix**, the merged column would be named **State Codes.State Code**.

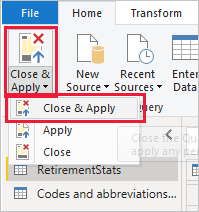
**Note**

If you want to explore how to bring in the **State Codes** table, you can experiment a bit. If you don’t like the results, just delete that step from the **APPLIED STEPS** list in the **Query Settings** pane, and your query returns to the state prior to applying that **Expand** step. You can do this as many times as you like until the expand process looks the way you want it.

We now have a single query (table) that combines two data sources, each of which was shaped to meet our needs. This query can be a basis for interesting data connections, such as housing cost statistics, quality of life, or crime rate in any state.

1. To apply your changes and close Power Query Editor, select **Close & Apply** from the **Home** ribbon tab.

The transformed dataset appears in Power BI Desktop, ready to be used for creating reports.



# LAB 1C: Analyze Sales Data from Excel and OData feeds

It is common to have data in multiple data sources. For example, you could have two databases, one for product information, and another for sales information. With *Power BI Desktop*, you can combine data from different data sources to create interesting, compelling data analyses and visualizations.

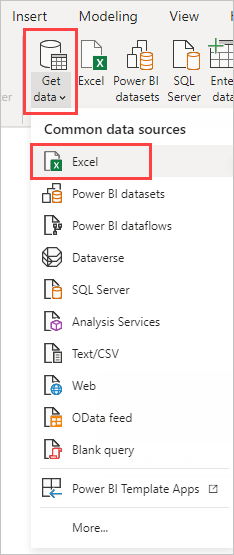
In this tutorial, you combine data from two data sources:

* An Excel workbook with product information
* An OData feed containing data about customer orders

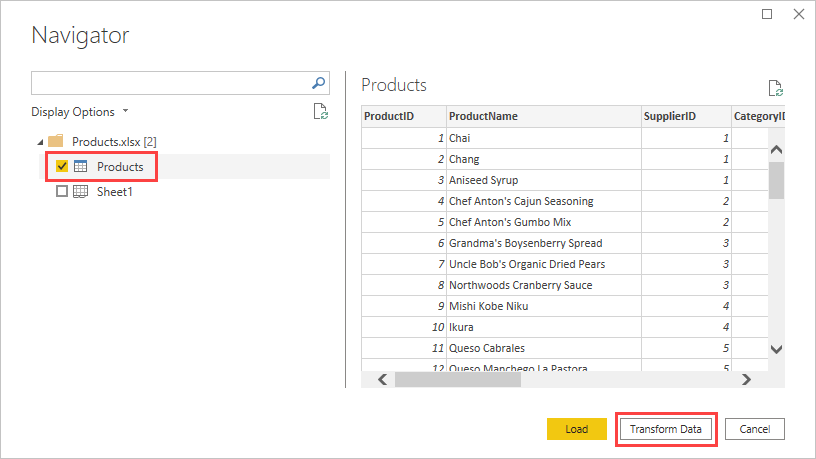
You're going to import each dataset and do transformation and aggregation operations. Then, you can use the data from the two sources to produce a sales analysis report with interactive visualizations.

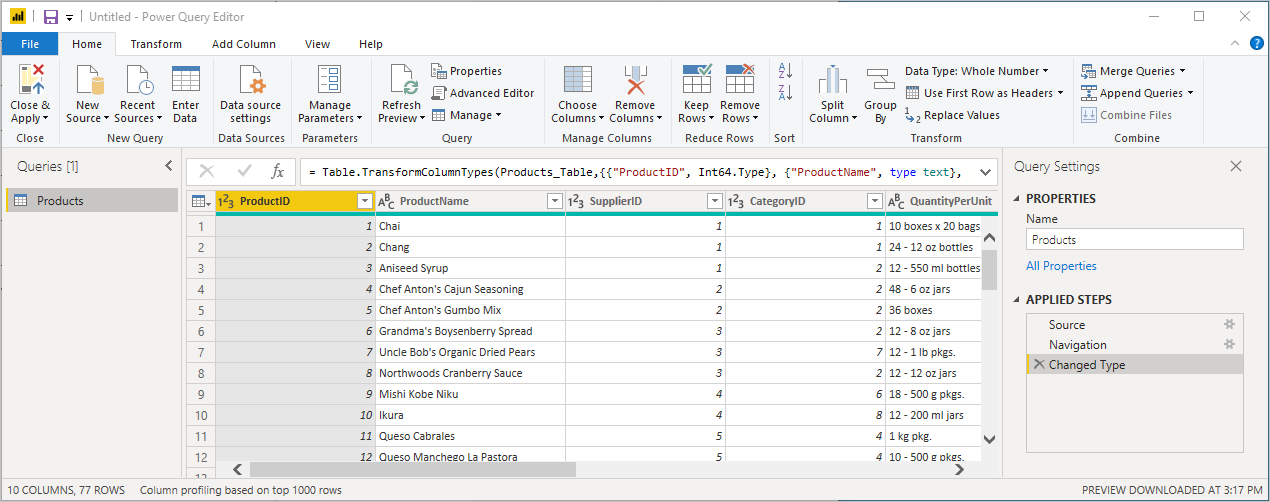
## Import the Excel Data File

1. Download the **Products.xlsx** file from Datasets Folder.
2. Select the arrow next to **Get data** in the Power BI Desktop ribbon's **Home** tab, and then select **Excel** from the **Common data sources** menu.



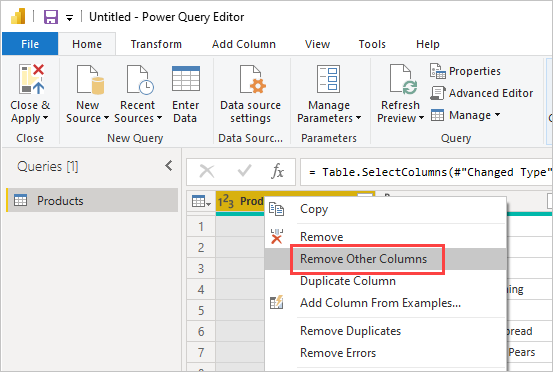
1. In the **Open** dialog box, navigate to and select the **Products.xlsx** file, and then select **Open**.
2. In the **Navigator**, select the **Products** table and then select **Transform Data**.



A table preview opens in the Power Query Editor, where you can apply transformations to clean up the data.  
  
  


**Remove Other Columns from the Query Editor**

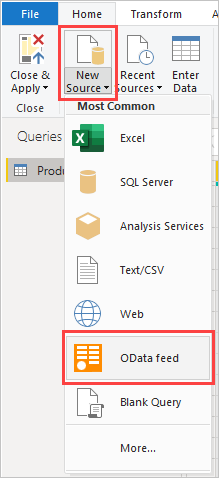
Your combined report uses the Excel workbook's ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns. You will now remove the other columns.

1. In Power Query Editor, select the **ProductID**, **ProductName**, **QuantityPerUnit**, and **UnitsInStock** columns. You can use Ctrl to select more than one column, or Shift to select columns next to each other.
2. Right-click any of the selected column headers. Select **Remove Other Columns** from the dropdown menu. You can also select **Remove Columns** > **Remove Other Columns** from the **Manage Columns** group in the **Home** ribbon tab.  
     
   

**Connecting to OData Feeds**

Next, import the order data from the sample Northwind sales system OData feed.

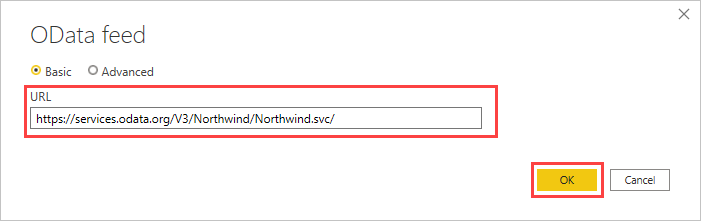
1. In Power Query Editor, select **New Source** and then select the **OData feed**.



1. In the **OData feed** dialog box, paste the Northwind OData feed URL:

<https://services.odata.org/V3/Northwind/Northwind.svc/>

1. Click **OK**.



1. In **Navigator**, select the **Orders** table, and then select **Load/OK** to load the data into Power Query Editor.

A screenshot of a computer

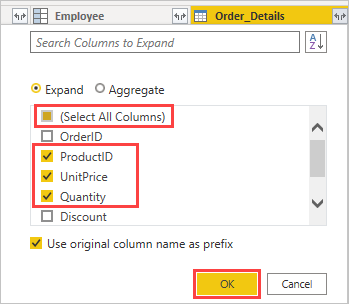
Description automatically generated

In **Navigator**, you can select any table name, without selecting the checkbox, to see a preview.

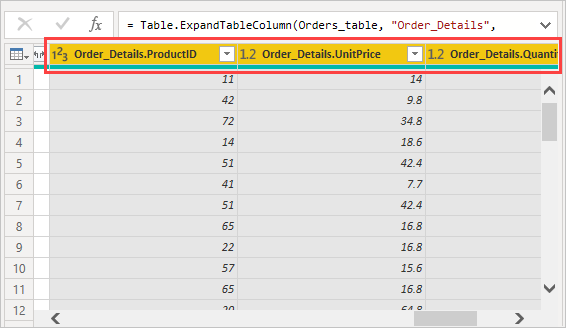
**Working with Related Data**

You can use table references to build queries when connecting to data sources with multiple tables, such as relational databases or the Northwind OData feed.   
  
The **Orders** table contains references to several related tables. The table **Order\_Details** in the OData Feed contains rows related to the main **Orders** table.

1. Scroll to the right in the **Orders** table until you see the **Order\_Details** column. It contains references to another table and not data.
2. Select the **Expand** icon (  ) in the **Order\_Details** column header.
3. In the dropdown menu:
   1. Select **(Select All Columns)** to clear all columns.
   2. Select **ProductID**, **UnitPrice**, and **Quantity**, and then select **OK**.



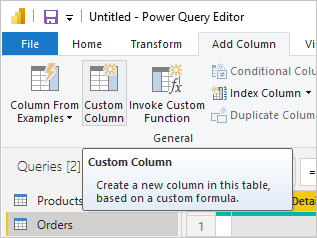
After you expand the **Order\_Details** table, three new nested table columns replace the **Order\_Details** column. There are new rows in the table for each order's added data.



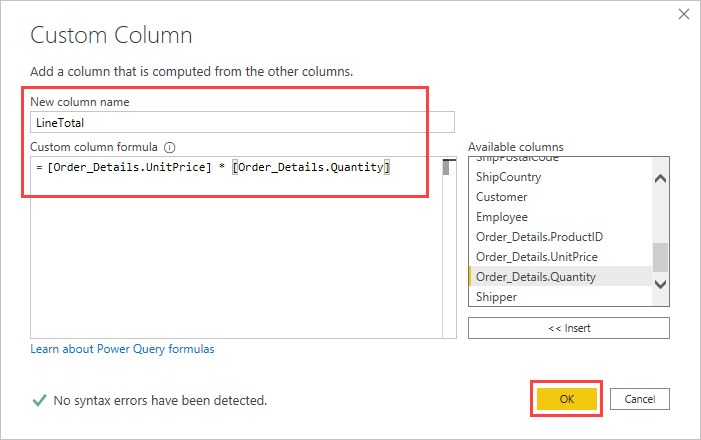
## Add a Calculated Column

We will add a column that multiplies the unit price by quantity to calculate the total price for each order's product.

1. In the Power Query Editor's **Add Column** ribbon tab, select **Custom Column**.



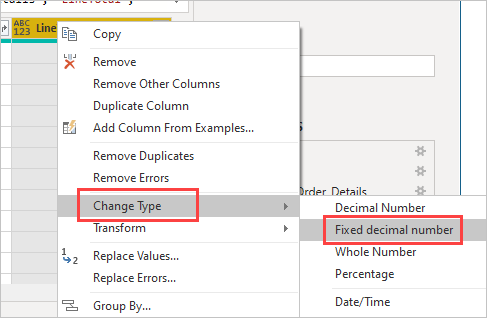
1. In the **Custom Column** dialog box, type **LineTotal** in the **New column name** field.
2. In the **Custom column formula** field after the **=**, enter **[Order\_Details.UnitPrice]** \* **[Order\_Details.Quantity]**.
3. Click **OK**. The new **LineTotal** field appears as the last column in the **Orders** table.



**Setting Field Data Types**

When Power Query Editor connects to data, it makes a best guess as to each field's data type for display purposes. A header icon indicates each field's assigned data type.

Your new **LineTotal** column has an **Any** data type, but it has currency values. To assign a data type, right-click the **LineTotal** column header, select **Change Type** from the dropdown menu, and then select **Fixed decimal number**.



**Delete, Rename, and Reorder Columns in Power Query Datasets**

To make your model easier to work with in reports, you can delete, rename, and reorder some columns.

Your report is going to use the following columns:

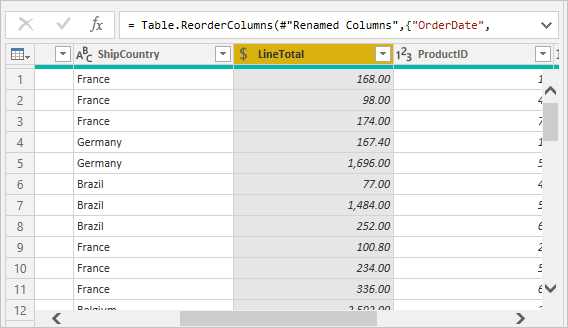
* **OrderDate**
* **ShipCity**
* **ShipCountry**
* **Order\_Details.ProductID**
* **Order\_Details.UnitPrice**
* **Order\_Details.Quantity**
* **LineTotal**

Select these columns and use **Remove Other Columns** as you did with the Excel data.

Rename the columns prefixed with "**Order\_Details.**" to make them easier to read:

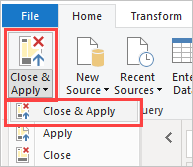
1. Double-click or tap and hold each column header, or right-click the column header, and select **Rename** from the dropdown menu.
2. Delete the **Order\_Details.** prefix from each name.

Finally, to make the **LineTotal** column easier to access, drag and drop it to the left, just to the right of the **ShipCountry** column.

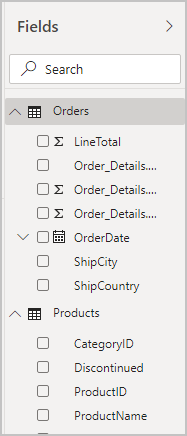


**Use the Edited Dataset In the PowerBI Report**

At this point we are done with the data transformation and we will import this data into the Power BI **Report** view. Select **Close & Apply** > **Close & Apply** in the **Home** ribbon tab's **Close** group.



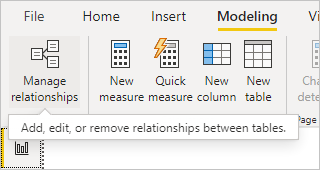
Once the data is loaded, the queries appear in the **Fields** list in the Power BI Desktop **Report** view at the right part of the Power BI window, in the data pane. Expand both sources to see the fields they contain. Fields with a Σ in front of them are numeric fields.



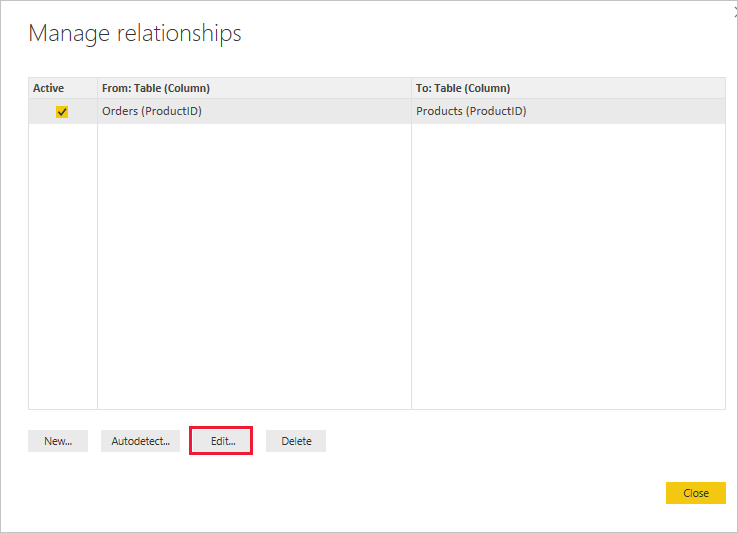
**Manage Relationship Between Datasets**

In Power BI we can create relationships between datasets based on common fields. Power BI may detect relationships automatically, or we can create them ourselves. We will now create a relationship between this lab's Orders and Products datasets.

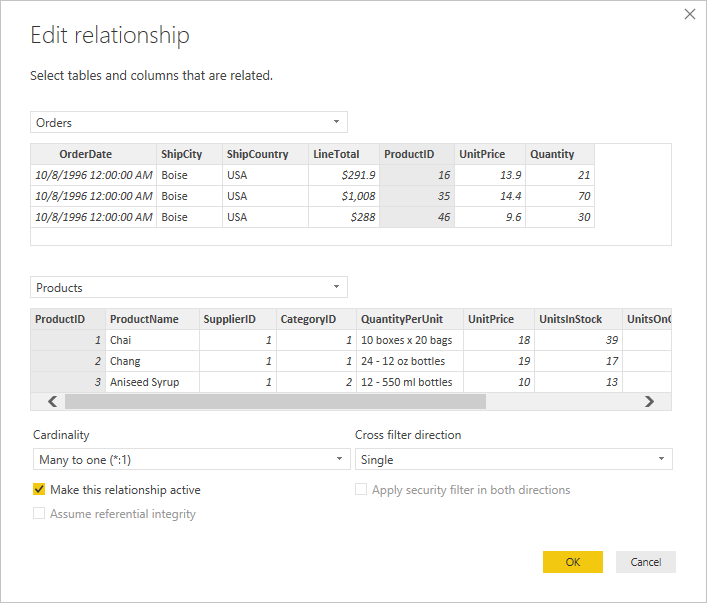
1. Make sure you are in the Power BI **Report** view. Click on **Modeling** in the menu ribbon and click **Manage relationships**.



1. In the **Manage relationships** dialog box, you can see that Power BI Desktop has already detected and listed an active relationship between the **Products** and **Orders** tables. To view the relationship, select **Edit**.

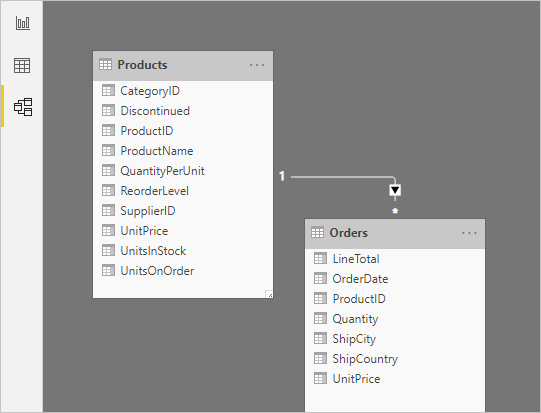


The **Edit relationship** opens, showing details about the relationship.



1. Power BI Desktop has auto-detected the relationship correctly, so you can select **Cancel** and then **Close**.

In Power BI Desktop, on the left side, select **Model** to view and manage query relationships. You can double-click the arrow on the line connecting the two queries to edit the relationship.

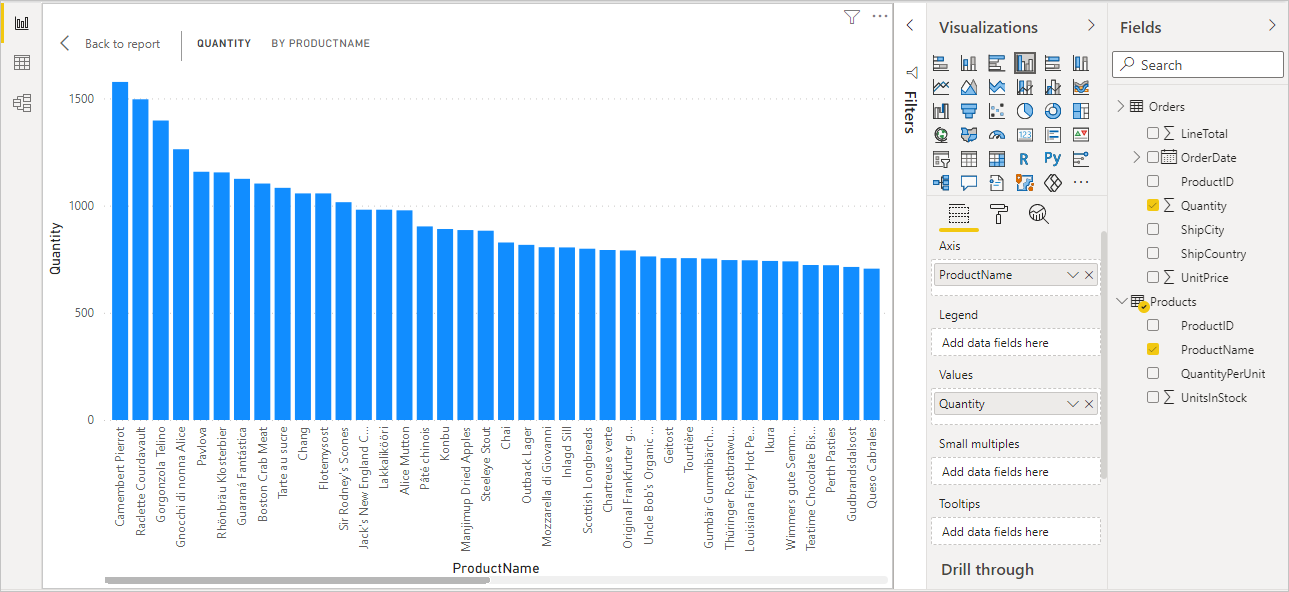


To get back to **Report** view from **Model** view, select the **Report** icon.

## Create a Stacked Column Chart

First, create a stacked column chart that uses fields from both queries to show the quantity of each product ordered.

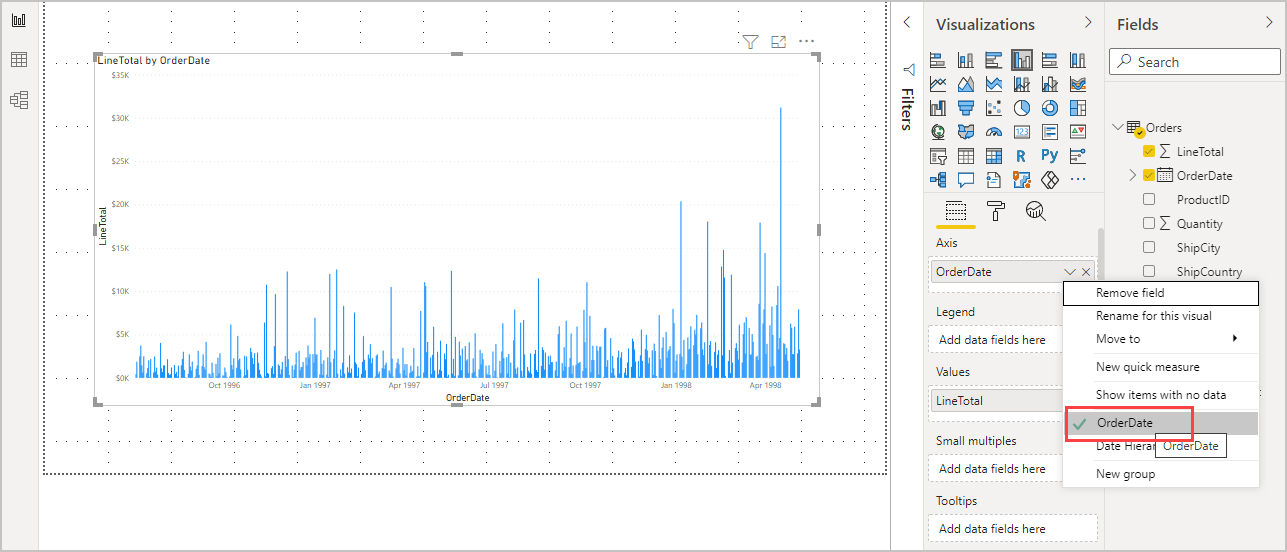
1. Select the **Quantity** field from **Orders** in the **Fields** pane at the right or drag it onto a blank space on the canvas. A stacked column chart is created showing the total quantity of all products ordered.
2. To show the quantity of each product ordered, select **ProductName** from **Products** in the **Fields** pane, or drag it onto the stacked chart you have just crteated.
3. To sort the products by most to least ordered, select the **More options** ellipsis (**...**) at the visualization's upper right, and then select **Sort By** > **Quantity**.
4. You can use the handles at the corners of the chart to increase its size or use the  focus icon to maximize the size of the report.
5. At the bottom of the report, right click the "Page" tab and rename the report to "**LAB1C**"



## Create a Line Chart

Next in the same page in Power BI, create a **Line Chart** visualization showing order dollar amounts (**LineTotal**) over time (**OrderDate**).

1. Select **LineTotal** from **Orders** in the **Fields** pane or drag it to a blank space on the canvas. From the visualizations pane to the right, make sure that the line chart is selected.
2. Select the line chart, then select **OrderDate** from **Orders**, or drag it onto the chart. The chart now shows line totals for each order date.
3. Drag the corners to resize the visualization and see more data.



If you only see **Years** on the chart and only three data points, select the arrow next to **OrderDate** in the **Axis** field of the **Visualizations** pane, and select **OrderDate** instead of **Date Hierarchy**. Alternatively, you might need to select **Options and settings > Options** from the **File** menu, and under **Data Load**, clear the *Auto date/time for new files* option.

Your report to this point should look like the following:

A screenshot of a graph

Description automatically generated

## Create a Map Visualization

Finally, in the same page, create a map visualization showing order amounts from each country or region.

1. Click on a blank space on the report below the two charts. Select **ShipCountry** from **Orders** in the **Fields** pane or drag it to a blank space on the canvas. Power BI Desktop detects that the data is country or region names. It then automatically creates a map visualization, with a data point for each country or region with orders. If you get a warning that maps are disabled, go to File/Options/Security and enable "**Use Map and Filled Map Visuals**"
2. To make the data point sizes reflect each country's/region's order amounts, click the **LineTotal** field. You can also drag it to **Add data fields here** under **Size** in the **Visualizations** pane. The sizes of the circles on the map now reflect the dollar amounts of the orders from each country or region. Hover your mouse point over any of the countries to see the country name and the total amounts of products sold in that country. Your report should now look like the following:  
     
   A screenshot of a graph

   Description automatically generated