Module 4: Shaping and Combining Data

Contents:

Module overview

Power BI Desktop offers a self-service solution for creating visual, interactive reports and dashboards. Users can connect to a wide variety of data sources, combining data from on-premises databases, Software as a solution (SaaS) providers, cloud-based services, and local files such as Microsoft Excel®, into one report. The beauty of Power BI reports and dashboards is the ability to rapidly build reports to present this data so it is instantly readable—with clusters, outliers, and patterns in data visually brought to light. To achieve this, each report must have a dataset comprising tables and columns that are ready to add straight into visualizations. Data must be formatted for relevant currencies, numbers should have correct decimal places, additional columns and measures might be required, and data may have to be combined from multiple tables. With Power BI Desktop, you can do all of this, with powerful, built-in tools for shaping your data. This module introduces the tools that are available for preparing your data, and transforming it into a form ready for reporting.

Objectives

After completing this module, you will be able to:

- Perform a range of query editing tasks in Power BI.
- Shape data, using formatting and transformations.
- Combine data together from tables in your dataset.

Lesson 1: Power BI Desktop queries

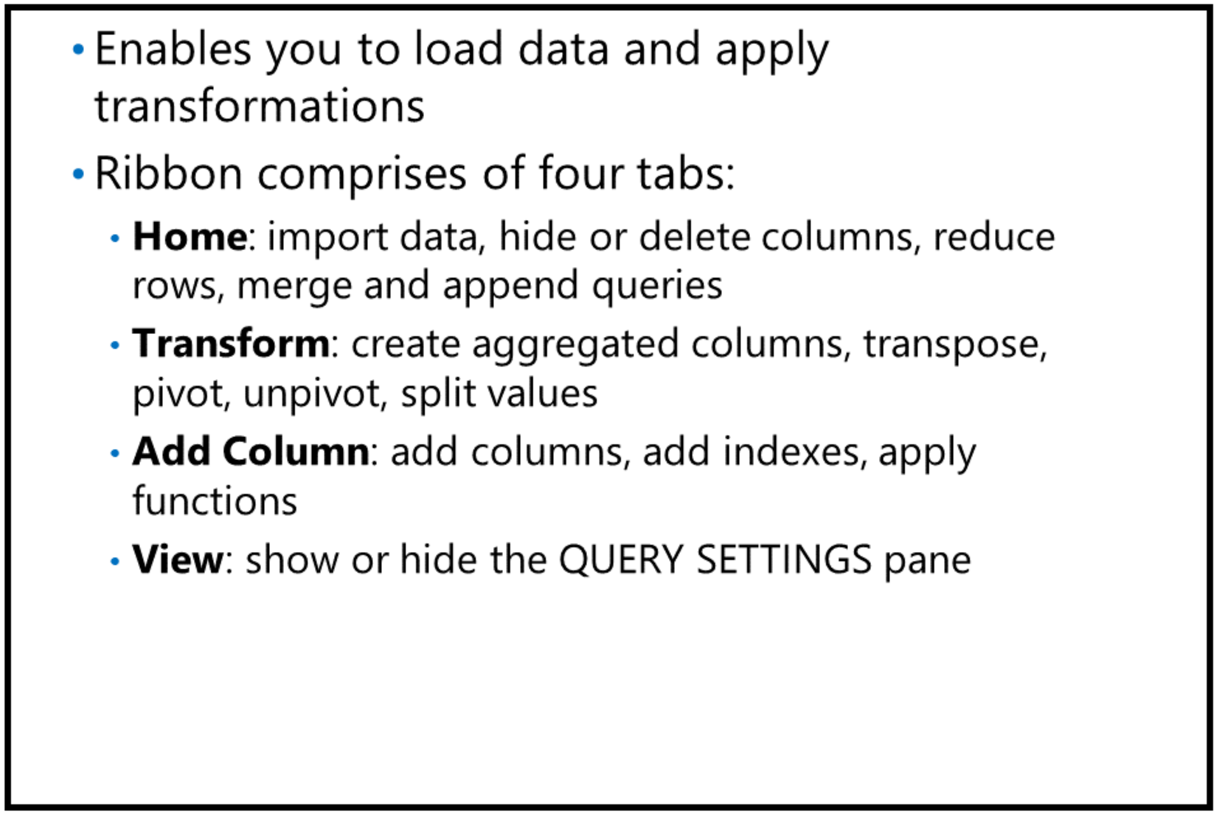
In this lesson, you will learn about the tools in Power BI that shape and transform your data, so that it is ready to use in reports. You will also explore the main features of the Advanced Editor.

Lesson objectives

After completing this lesson, you will be able to:

- Use the Power Query Editor to shape your data.
- Roll back your data shaping steps using the APPLIED STEPS feature.
- Change the code that Power BI uses to query the data sources.

The Power Query Editor



- Enables you to load data and apply transformations
- Ribbon comprises of four tabs:
  - **Home**: import data, hide or delete columns, reduce rows, merge and append queries
  - **Transform**: create aggregated columns, transpose, pivot, unpivot, split values
  - **Add Column**: add columns, add indexes, apply functions
  - **View**: show or hide the QUERY SETTINGS pane

By using the Power BI Power Query Editor, you can load data from a wide number of data sources, and apply transformations, including adding new columns and measures. There are two ways of accessing the Power Query Editor: you can click **Edit** when loading data using **Get Data**, or, from Power BI Desktop, on the **Home** menu, in the **External data** group, click **Edit Queries**. In the Power Query Editor window, click a table or view in the Queries pane, to display the data. The column names are prefixed with letters (ABC), numbers (123), currency symbols ($), or a calendar for datetime columns to represent the data type of the column. The Power Query Editor ribbon comprises four tabs for shaping your data:

Home

You can import data from the Power Query Editor using **New Source**, **Recent Sources**, or **Enter Data**, in the **New Query** group. These offer the same functionality as the **Get Data**, **Recent Sources**, or **Enter Data** options in the main Power BI design window.

In the **Data Sources** group, click **Data source settings** to change the properties of the data source. You cannot edit the query, but you can change the server and database, in addition to the login details. You can also choose to encrypt the connection and set the security level.

You use the **Parameters** group to manage and create parameters that can be used in a variety of ways within the report. Click **Manage Parameters** to edit the properties and data values of your parameters, or delete a parameter. Use **Edit Parameters** to select the current values for each of the parameters within the report. To add a new parameter, click **New Parameter**. You can explicitly specify the values for the parameters, or use a query, in addition to setting the data type and determining whether a value must be supplied.

The **Query** group includes a function to refresh the preview data for the current table, or all tables in the dataset. Click **Properties** to edit the name of the query, and the optional query description, and to configure whether to **Enable load to report** and **Include in report refresh**. Click **Advanced Editor** to view and edit the query code.

With the **Manage Columns** group, you can hide columns in your query using the **Choose Columns** function. You can also add them back in later if they are required. **Remove Columns** removes the currently hidden column(s) from the query.

Use the **Reduce Rows** group to keep or remove rows from the query. By using **Keep Rows**, you can retain the top or bottom specified number of rows, or use **Keep Range of Rows**, by entering the **First row** as the starting row, and then **Number of rows**. All other rows are removed. **Remove Rows** gives you the option to remove the top, or bottom specified number of rows, alternate rows, or blank rows. You can use **Remove Duplicates** to delete rows with identical values in one or more columns. You can also choose to **Remove Errors**, or **Keep Errors**.

The **Sort** group provides options for sorting data from **A-Z** or **Z-A**. You can apply a sort to multiple columns in a query, though you should always start with the column that has the less than unique values. For example, apply the sort in order of Country, Region, and then City.

For quick access, you will find the most common transformations on the **Transform** group. These are also included on the Transform tab.

The **Combine** group provides functions for merging and combining data between queries. You can use **Merge Queries** to combine columns, or use **Append Queries** to combine rows.

Transform

The **Table** group on the **Transform** tab includes a **Group By** function with which you can create a new column by applying an aggregation function to an existing column. This group also includes **Use First Row As Headers**, which is useful when importing data and Power BI has not detected that the first row contains the header. **Transpose** converts columns into rows, and rows into columns; **Reverse Rows** reorders the rows so that the last rows are at the top, inverting the order of the data. **Count Rows** returns a count of rows in the table.

The **Any Column** group functions can be applied to all columns in your table, regardless of data type. You can change the data type of columns, rename columns, replace values, and errors, and use the **Fill** function to fill empty cells with a neighboring value, going up or down. You can also move, pivot, and unpivot columns.

With the **Split Column** function in the **Text Column** group, you can split one column into one or more columns, based on a delimiter. This is very useful for extracting concatenated strings.

The **Structured Column** group gives you options for working with nested data such as tables or lists. The **Expand** function promotes nested data to become columns or rows in the top-level table. **Aggregate** summarizes data from a nested structure to reveal average, minimum, maximum, and count values.

You use the **Run R Script** function in the **Scripts** group to run R queries directly in the editor. You must have R installed to use this function.

The rest of the items included on this tab are also available on the **Add Column** tab.

Add Column

You can use the **Custom Column** button to create a new column using a formula. A list of available columns to work with is included, and the syntax checker ensures your formula is correct before applying to create the new column. You use the **Index Column** button to create a new index column on a table. The index might be an incremental value, starting at 1, and incrementing by 1—you can also set the starting value to be 1 or 0. With the custom index column, you set the starting value and increment to any value. You can also duplicate columns using existing values, applying built-in string functions such as uppercase, lowercase, or capitalizing each first letter.

You can use **Conditional Column** to add a column based on the values in another column. For example, if you have a Title column, and want to create a new Gender column, you could specify that if the value in the Title column was Mr, then the value in the new Gender column would be Male—for all other values, it would be Female.

Instead of explicitly specifying the comparison value of Mr, you could choose a parameter to supply this value. If you create the new column based on a date or datetime column, you can use the date picker for the comparison value. Data can also be cleaned and trimmed, and you can add a suffix or prefix. You can extract a substring from a column value to create a new column, and parse JSON, and XML column data into columns.

You use the **From Number** group to apply statistical, standard, and scientific functions to numerical columns. Statistical functions include aggregations such as sum, minimum, maximum, and average; you can also count values and distinct values. Standard functions you can apply include add, subtract, multiply, and divide. The scientific functions include absolute value, square root, exponent, and factorial. You can also apply trigonometry, rounding, and information such as **Is Even**, or **Is Odd**.

The **From Date & Time** group offers useful functions for extracting dates, times, and durations from existing datetime columns. You create a new column by extracting the year, month, day, or quarter from a value, and compare two columns to extract the date or time difference.

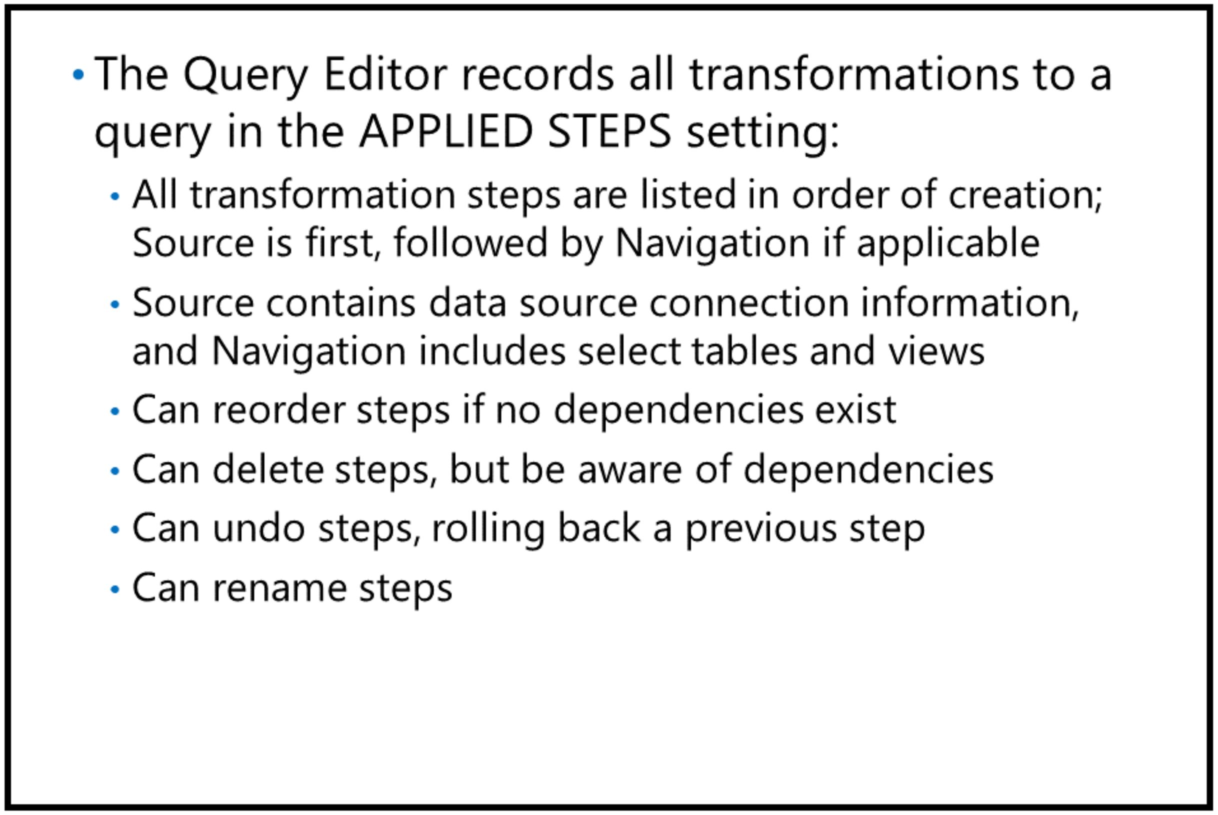
View

With the **View** tab, you can show or hide the QUERY SETTINGS pane. Settings include the name of the query, or table, and the list of steps, which are the transformations performed on the query. From here, you can also open the Advanced Editor window to view and edit the query code. You can also show or hide the

Formula Bar—and toggle to display the data as monospaced—and decide whether to show or hide white space.

Note: When you click a column in the Power Query Editor, Power BI determines the data type from the values, and enables the relevant features on the tabs, so you can only apply formatting to columns with applicable data. You use **Determine Data Type** to run automatic data type detection against select columns.

APPLIED STEPS



- The Query Editor records all transformations to a query in the APPLIED STEPS setting:
  - All transformation steps are listed in order of creation; Source is first, followed by Navigation if applicable
  - Source contains data source connection information, and Navigation includes select tables and views
  - Can reorder steps if no dependencies exist
  - Can delete steps, but be aware of dependencies
  - Can undo steps, rolling back a previous step
  - Can rename steps

When you shape your data using Power Query Editor, Power BI saves a list of the transformations you applied to your data, such as rename a column, delete a column, or change a data type. This list is displayed in the APPLIED STEPS section of the QUERY SETTINGS pane. Each time you connect to the data source to run the query, Power Query Editor applies these steps to the data, so it is always presented exactly how you shaped it.

> **Note:** When you shape the data in Power BI using the Power Query Editor, you are only amending the query, and reflecting these changes in the data that has been imported. The data in the data source remains unaffected.

The steps order

The Power Query Editor saves each change you make sequentially in date order, so the first transformation you made will always be applied to the data first. If you share a query with a colleague, these steps are included as part of the query, so the shaping can be applied again and your colleague sees the data exactly as you have specified. The **Source** step is first in the list, followed by **Navigation** if you selected the data from a list of tables or views. The **Source** step is the data source you connected to, and **Navigation** is the table you selected from that source. You cannot delete the **Source** step, but you can delete the **Navigation** step. However, this breaks any following steps.

If you click the gear icon to the right of **Source**, this opens the connection dialog box. For example, if your connection is to an Azure SQL Server database, click **Source** to open the connection dialog box. This shows the server name and database, and the code if you entered a query. If you included a query at this stage, no **Navigation** step is included. However, if you did not include a query, and instead selected from a list of available tables and views, you can use **Navigation** to change the source table or view.

Reordering steps

You can change the order of the steps in the APPLIED STEPS list, but you must be careful that this does not break the query. For example, if you move the step **Renamed Columns** above **Navigation**, you will break the query, so be aware of the dependencies between the steps. To move a step, right-click the step in the APPLIED STEPS section, and choose **Move Up**, or **Move Down**, or drag the step.

Deleting steps

You can delete steps in the list, effectively rolling back and undoing an action, but only if there are no later dependencies on the steps. If the step is isolated and has no later transformations that are dependent on the previous step, you can probably delete it. This is a useful—and fast—method for undoing transformations, such as removing a column that you later realize you need to include. To remove a step, click the **X** to the left of it.

Undoing steps

In addition to undoing deleted columns, if you hide a column using the **Choose Column** option, you can click the gear icon to the right of a step, and it opens the list box that was used for selecting the column you wanted to hide. You then select or clear columns.

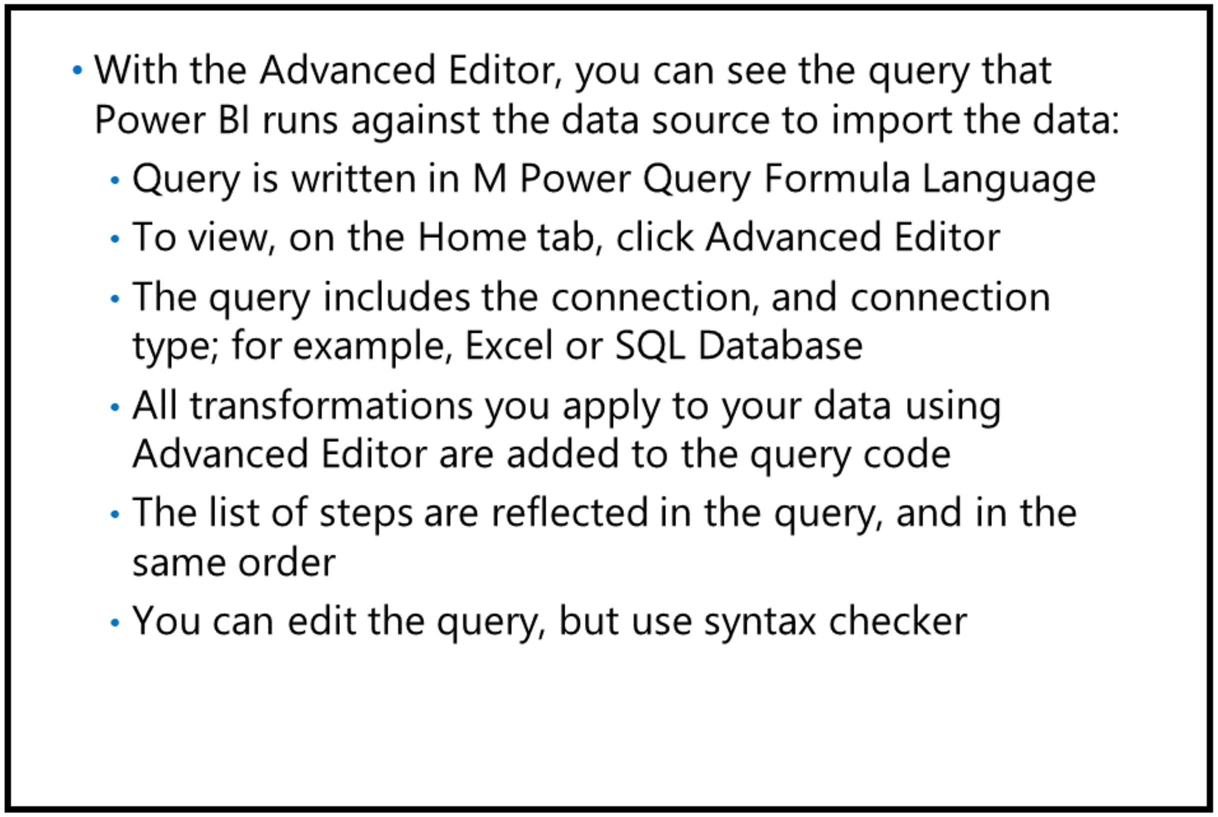
Renaming steps

Each step in the list is given a generic name, such as **Removed Column**, **Removed Other Columns1**, **Removed Duplicates**. This is not helpful if you have a long list of steps and want to go back and adjust the order, or roll back a step. However, you can rename a step. In the APPLIED STEPS pane, right-click a step, and click **Rename**. Type in the name of the step; for example, **CustomerID renamed Customer Code**. Providing sensible names for your steps helps you make future amendments, and assists colleagues with whom you share queries, as they see which transformations are applied.

Best Practice: Providing sensible names for the steps in your queries helps if you return to the data after a long time, and have forgotten exactly what transformations are applied. This is particularly helpful if you want to stop halfway through shaping your data, and return later. You can see the list of transformations, and pick up from where you finished. This will be helpful if you share the query with colleagues.

You can also add a description to each of the steps. Right-click a step and click **Properties** to open the **Step Properties** dialog. In the **Description** box, type the description of the transformation, and click **OK**. When you hover your mouse over the steps in the APPLIED STEPS pane, a tooltip displays the name of the step and the description.

The Advanced Editor



- With the Advanced Editor, you can see the query that Power BI runs against the data source to import the data:
  - Query is written in M Power Query Formula Language
  - To view, on the Home tab, click Advanced Editor
  - The query includes the connection, and connection type; for example, Excel or SQL Database
  - All transformations you apply to your data using Advanced Editor are added to the query code
  - The list of steps are reflected in the query, and in the same order
  - You can edit the query, but use syntax checker

You can use the Advanced Editor to see the query that is run against the data source. The query is written in M, the Power Query Formula Language. To view the query code from Power BI Desktop, on the **Home** tab, click **Advanced Editor**.

Advanced Editor Query to Return Unfiltered Data from the SalesLT.SalesOrderDetail Table

The following code connects to an Azure SQL Database, and returns all columns and rows in the SalesLT.SalesOrderDetail table, without any filtering applied:

```
let
    Source = Sql.Database("sqlazure.database.windows.net",
"AdventureWorksLT"),
    SalesLT_SalesOrderDetail =
Source{[Schema="SalesLT",Item="SalesOrderDetail"]}[Data]
in
    SalesLT_SalesOrderDetail
```

When you make transformations to your data in the Power Query Editor, the steps are saved to the APPLIED STEPS in the QUERY SETTINGS. These steps are also applied to the code in the Advanced Editor. For example, the following code shows the steps that have been applied to the SalesOrderDetail query. First, the SalesOrderDetailID column was removed, and then the OrderQty column was renamed Order Quantity. Finally, the rowguid and ModifiedDate columns were removed.

Advanced Editor Query to Return Filtered Data from the SalesLT.SalesOrderDetail Table

The following code shows the connection to the AdventureWorksLT database on Azure, with filtering applied using the Power Query Editor:

```
let
    Source = Sql.Database("sqlazure.database.windows.net",
"AdventureWorksLT"),
    SalesLT_SalesOrderDetail =
Source{[Schema="SalesLT",Item="SalesOrderDetail"]}[Data],
    #"Removed SalesOrderDetailID" =
Table.RemoveColumns(SalesLT_SalesOrderDetail,
{"SalesOrderDetailID"}),
    #"Rename OrderQty" = Table.RenameColumns(#"Removed
SalesOrderDetailID",{{"OrderQty", "Order Quantity"}}),
    #"Removed rowguid and ModifiedDate" =
Table.RemoveColumns(#"Rename OrderQty",{"rowguid", "ModifiedDate"})
in
    #"Removed rowguid and ModifiedDate"
```

The transformations in the code reflect the order in the APPLIED STEPS—these must be in the correct order when run against the data source. You can alter the code in the Advanced Editor, but you should use the syntax checker to ensure you do not break the code.

Demonstration: Using APPLIED STEPS

In this demonstration, you will see how to:

- Add transformations to a query, and see the steps in APPLIED STEPS.
- Rename steps in the APPLIED STEPS list.
-

See the steps reflected in Advanced Editor.

- Delete steps, and change the source table in the Navigation step.

Check your knowledge

Select the best answer

Which of the following statements about APPLIED STEPS is false?

Steps are added in sequential order.

You can rename the steps.

The Source step is always the first step.

The Navigation step only shows if you have selected tables or views from the data source, instead of using a query.

You can move a step between the Source step, and the Navigation step.

Check answer Show solution Reset

Lesson 2: Shaping data

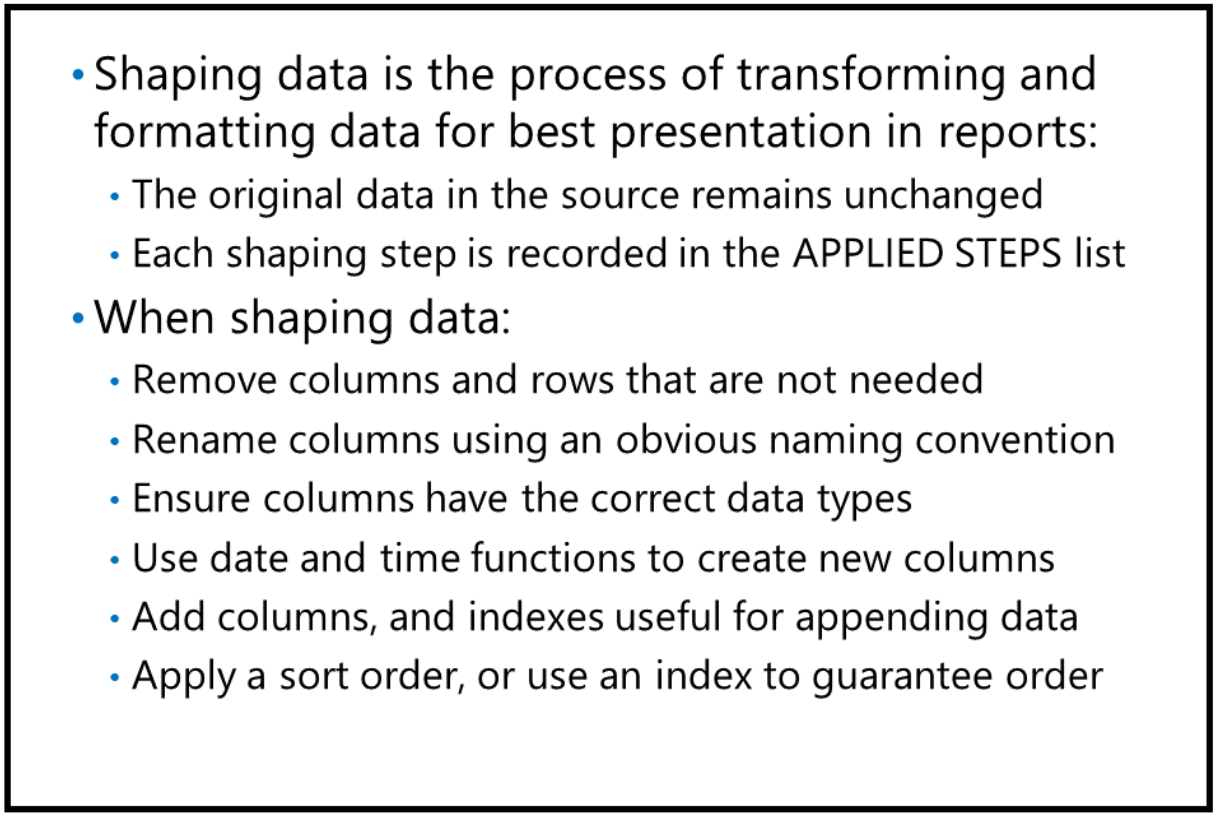
This lesson explores the powerful features in Power BI with which you can shape your data ready for use in reports. You will learn how to shape your data, by applying formatting to make your data better for Power BI to handle in visuals, and how to apply transformations.

Lesson objectives

After completing this lesson, you will be able to:

- Describe how data shaping makes your data easier for reporting.
- Format your data so that Power BI manages it correctly within charting visuals.
- Transform your data using techniques such as adding new columns and changing data types.

What is shaping data?



- Shaping data is the process of transforming and formatting data for best presentation in reports:
  - The original data in the source remains unchanged
  - Each shaping step is recorded in the APPLIED STEPS list
- When shaping data:
  - Remove columns and rows that are not needed
  - Rename columns using an obvious naming convention
  - Ensure columns have the correct data types
  - Use date and time functions to create new columns
  - Add columns, and indexes useful for appending data
  - Apply a sort order, or use an index to guarantee order

Shaping data is the process of transforming data, so the Power Query Editor loads and presents it in the best way for your reports. The original data source remains unchanged, because it is just the view in Power BI that you are adjusting. Each of the transformation steps is captured in the APPLIED STEPS under QUERY SETTINGS. You use the Power Query Editor to shape your data, using features such as adding or removing columns, renaming columns, combining data, changing data types, transposing columns, and applying functions. Ideally, you want to shape your data before working with it in visuals. The most common data shaping techniques are described here.

Removing columns and rows

You should always remove data that isn’t required. The dataset should be as succinct as possible, so you do not have redundant data that is loaded unnecessarily. If you have a large dataset, remove everything that isn’t required to make it as small as possible and improve the performance of handling the data in Power BI. This means less data is transferred from the source to Power BI; there is less data to be processed as the Power Query Editor applies the transformations; and you have less extraneous data when creating reports.

Renaming columns

Your columns should have names that make it easy to work with them when creating reports and viewing dashboards. Each column name should give the data in that column an adequate description. This is particularly relevant when working with datasets containing several tables and columns—it makes it easier to find the right fields to add to report visuals. Power BI Q&A, which uses the natural query language, also returns more accurate results if it can find the data needed to answer the question it’s being asked.

Data types

The Power Query Editor makes a best guess at the data type of each column when loading in the data. It is a good idea to check the given column types are as you would expect, and then format any that are incorrect. This can be critically important for decimal columns, where changing the data types between a decimal and a whole number could potentially give false results in calculations. Currency columns that don’t contain a currency symbol in the source are not typed as currency, so checking these columns and formatting them as your local currency will give better results in aggregations—in addition to formatting the data so it presents better in data labels.

Datetime columns

You should also format datetime columns with the correct data type. You use the Date and Time transformations to add additional columns to extract the year, quarter, month, week, and day from a date column, and hours, minutes, and seconds from a time column. You calculate the difference between two data columns to create a new column. For example, you could subtract the **Delivery Date** from the **Order Date** to create a **Days to Fulfill** column, showing how long it took to deliver an order after it had been placed. If you have a **Date of Birth** column, you can use the **Age** function to create a new column for the person's current age.

Adding columns

There are many options and functions to help you create new columns in your queries. You can duplicate an existing column, split a column into multiple columns, use the data and time functions described previously, or concatenate values in multiple columns to create **Full Name** or **Full Address** fields. You can also use math functions to create calculated columns; for example, to subtract **Manufacturing Costs** from **Retail Price** to create **Profit**. You merge data from a date column with data from a time column to create a new datetime column.

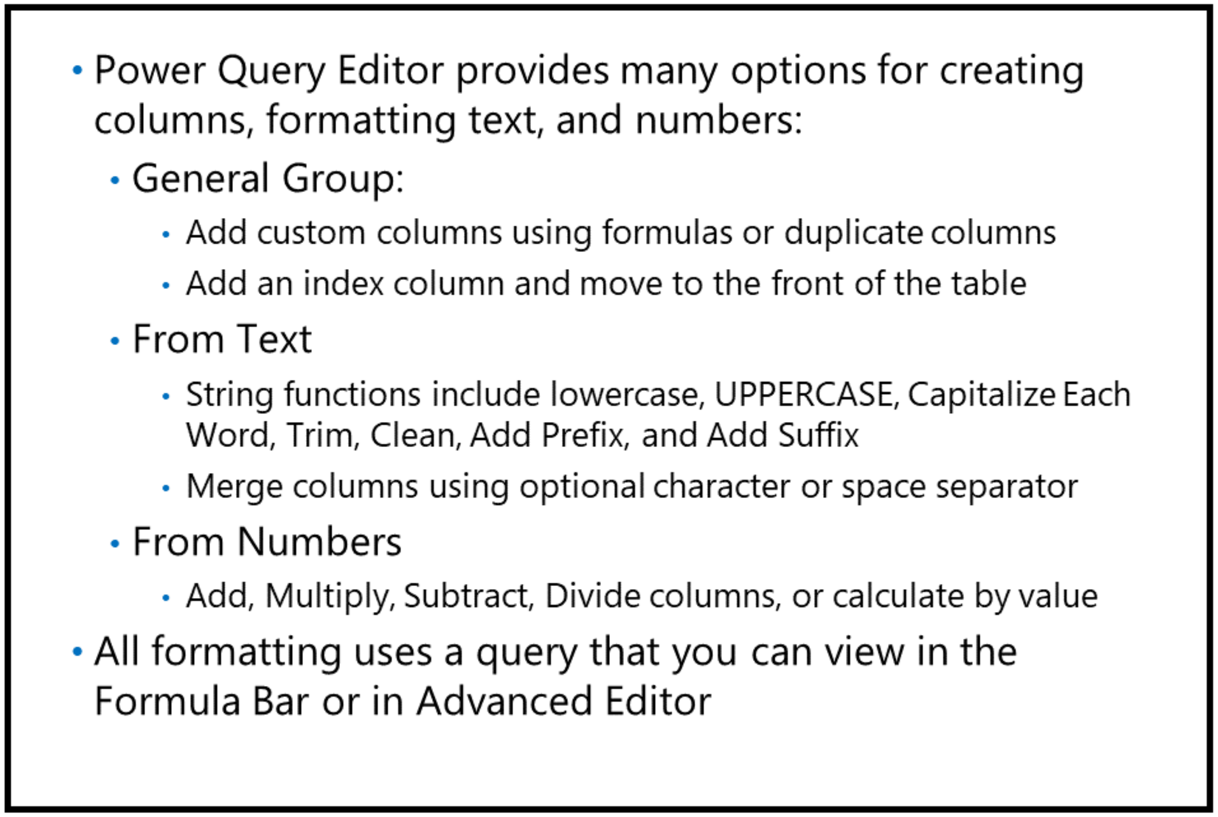
Adding indexes

You can add indexes to your tables, with the seed value starting at 1 or 0, or you can create a custom index by defining the start number and the increment. If you combine data from different systems, you may find that there are overlaps in the index key columns, meaning you wouldn’t have unique values when merged together. In the Power Query Editor, you can add an index as a surrogate key column to the two tables you are appending, so the index value is always unique.

Apply a sort order

By default, Power BI sorts data in alphabetical order in visualizations. While this may be desirable in some instances, you might want to order by a month column or by another categorization. The Home tab in the Power Query Editor includes a Sort group, with which you can sort A-Z, or Z-A. These may not fulfill your criteria, in which case you can add an index column and use this to sort by in the visualizations.

Formatting data



- Power Query Editor provides many options for creating columns, formatting text, and numbers:
  - General Group:
    - Add custom columns using formulas or duplicate columns
    - Add an index column and move to the front of the table
  - From Text
    - String functions include lowercase, UPPERCASE, Capitalize Each Word, Trim, Clean, Add Prefix, and Add Suffix
    - Merge columns using optional character or space separator
  - From Numbers
    - Add, Multiply, Subtract, Divide columns, or calculate by value
- All formatting uses a query that you can view in the Formula Bar or in Advanced Editor

By formatting your data, you help Power BI categorize and identify the data, making it much easier to work with. In Power Query Editor, you apply string functions to your text columns to create consistency, ensuring that data is well presented.

General formatting

The **Custom Column** button on the **Add Column** ribbon enables you to enter a custom formula to create a new column, including calculations using values from other columns. The syntax checker indicates when you have an error and does not allow you to save a formula with errors. To create a new column, click **Custom Column**. In the **New column name** box, type the name of the column, and add your formula to the **Custom column formula** box; for example, **[ShipDate] - [OrderDate]**. To avoid errors in the column names, you select a column in the **Available columns** list and click **Insert** to add it to the Custom column formula text box. When your formula is complete, click **OK**. The new column is appended to the end of the table and the formula is visible in the Formula Bar. If you open Advanced Editor, you will see the formula appended to the query.

The following example shows the code in the Formula Bar, which subtracts the **OrderDate** from the **ShipDate** to return the number of days as the new column **DaysOrderToShip**.

Custom Column Formula

The following code is the formula to create a custom column, which calculates the days from when the order was placed to when it shipped:

```
 = Table.AddColumn(#"Sorted Rows", "DaysOrderToShip", each [ShipDate]
 - [OrderDate])
```

The column is created with a data type of Any, and values are in the format of 7:00:00:00. To change the type, right-click the column, click **Change Type**, and then select the appropriate type. In this example, you could convert the data to a Whole Number type.

You can also add an index column to the end of your table. You can start the index at 1 or 0, or choose the starting value. By default, the index increments by 1, but you can change this using the custom index option. To add an index, click the down arrow on the **Index Column** button on the **Add Column** ribbon. Click **From 0** or **From 1** to immediately add an index using the chosen starting value. If you click **Custom**, this opens the **Add Index Column** dialog box. In the **Starting Index** box, type your starting number; for example, 100. In the **Increment** box, type the number you want the index to increase by with each row; for example, 10. The index in this case would be 100, 110, 120, 130, and so on. It is common practice to have your index as the first column in the table, so right-click the new index column, point to **Move**, and then click **To Beginning**. You can also select multiple columns to move them together.

The **Duplicate Column** function is useful when you have a string column that will be split, but you want to retain the original value. You click to select the column and choose **Duplicate Column** from the **General** group or right-click and select **Duplicate Column**. The new column is appended to the end of the table, and given a name such as **SalesOrderNumber - Copy**. You then work with this column to split the values, or perform other operations, such as replacing substrings or trimming repeated characters.

Formatting text

The **From Text** group functions provide options for formatting string values, merging columns, extracting values, and parsing to other formats. The **Format** function converts strings to **lowercase**, **UPPERCASE**, **Capitalize Each Word**, **Trim**, **Clean**, **Add Prefix**, and **Add Suffix**. You can use these to convert your string data into consistent formats, which is particularly helpful when importing raw data that has not been cleaned. If you import data from an e-commerce website, where customers have entered their names and addresses, and no formatting was applied before the data was saved to the database, it is likely to be inconsistent—with mixed casing across the various fields. You can use **Capitalize Each Word** so the columns all have the correct casing, and apply **UPPERCASE** to state codes, such as MA, NJ, WA, or country or area names depending on your reporting requirements.

You can create a new column by merging two or more columns together. To do this, click to select a column, and hold down the Ctrl key and click the other columns you want to merge. On the **From Text** group, click **Merge Columns**. In the **Merge Columns** dialog box, choose how you want the values to be separated, from **Colon**, **Comma**, **Equals Sign**, **Semicolon**, **Space**, **Tab**, or **Custom**. For Custom, enter the symbol or character, such as a dash. In the **New column name (optional)** box, give your column an appropriate name, and click **OK**.

Note: The column values are concatenated in the order in which you click the columns to select them. This gives you full control over the end result.

Best Practice: The Merge Columns function can be used on address fields to quickly create a full address column. Highlight your address columns in order, and click **Merge Columns**. For the separator, choose **Custom**, and enter “, ” (comma and a space). This concatenates all the values together in a comma separated list.
However, you are likely to have null values or empty strings in some columns, perhaps Address2, which results in double commas. You then use the **Replace Values** function on the **Any Column** group of the **Transform** tab, to replace “, ,”, with “, ”.

You use the **Extract** function to copy a substring value from one column, to create a new column. You also use **Extract** to count the length of a string. Select a column in your table, and click **Extract** from the **From Text** group. Click **Length** to create a new column that counts the number of characters in the column. Spaces are included in the length. To extract a fixed number of characters from the beginning or the end of the column value, use **First Characters**, or **Last Characters**. Select the column and click **Extract**, and then either **First Characters**, or **Last Characters**. Enter a value for the **Count**, and click **OK**. This is useful if you want to split a **PostalCode** column to extract the first few characters to create a map based on area, rather than exact postal code. To extract a specific number of characters from the middle of a string, you use **Range**. Select the column, and click **Extract**, **Range**. Provide a **Starting Index**, and a **Number of Characters** value, and click **OK**. If you type 2 for the Starting Index, the extract starts on the third character.

The **Parse** function takes a column that is an XML or JSON format, and parses it into a table. Select the column with your XML or JSON data, and click **Parse** from the **From Text** group. Select **XML**, or **JSON**, and the Power Query Editor adds a table column to the current table. Click the double-arrow icon to expand the new table, and choose the attributes you want to include in the table. This is a very quick way to parse and extract data provided in XML or JSON format.

Formatting numbers

There is a wide range of formatting functions that you can apply to your numeric columns. The **From Number** group includes functions for **Statistics**, **Standard**,

Scientific, **Trigonometry**, **Rounding**, and **Information**. This section focuses on the more common standard number functions. Choose from **Add**, **Multiply**, **Subtract**, **Divide**, **Divide (Integer)**, **Modulo**, or **Percentage**.

Add

To add two or more columns together, click the first column, hold the Ctrl key, and click the other columns. On the **From Number** group, click **Standard**, **Add**. This creates a new column with a default name of **Inserted Addition**. You can also add a whole or decimal number to a column. Select a single column, and then click **Standard**, **Add**. Enter the number that you want to be added to the existing column value.

Multiply

If you want to multiply two or more columns together, click the first column, hold the Ctrl key, and click the other columns. On the **From Number** group, click **Standard**, **Multiply**. This creates a new column with a default name, **Inserted Multiplication**. To multiply a column by a whole or decimal number, select a single column, and then click **Standard**, **Multiply**. Type the number that you want the existing column value to be multiplied by. For example, to calculate a net value to include 20 percent tax, click the **net** column, click **Standard**, **Multiply**, and type **1.2**. This creates a column with the additional tax amount.

Subtract

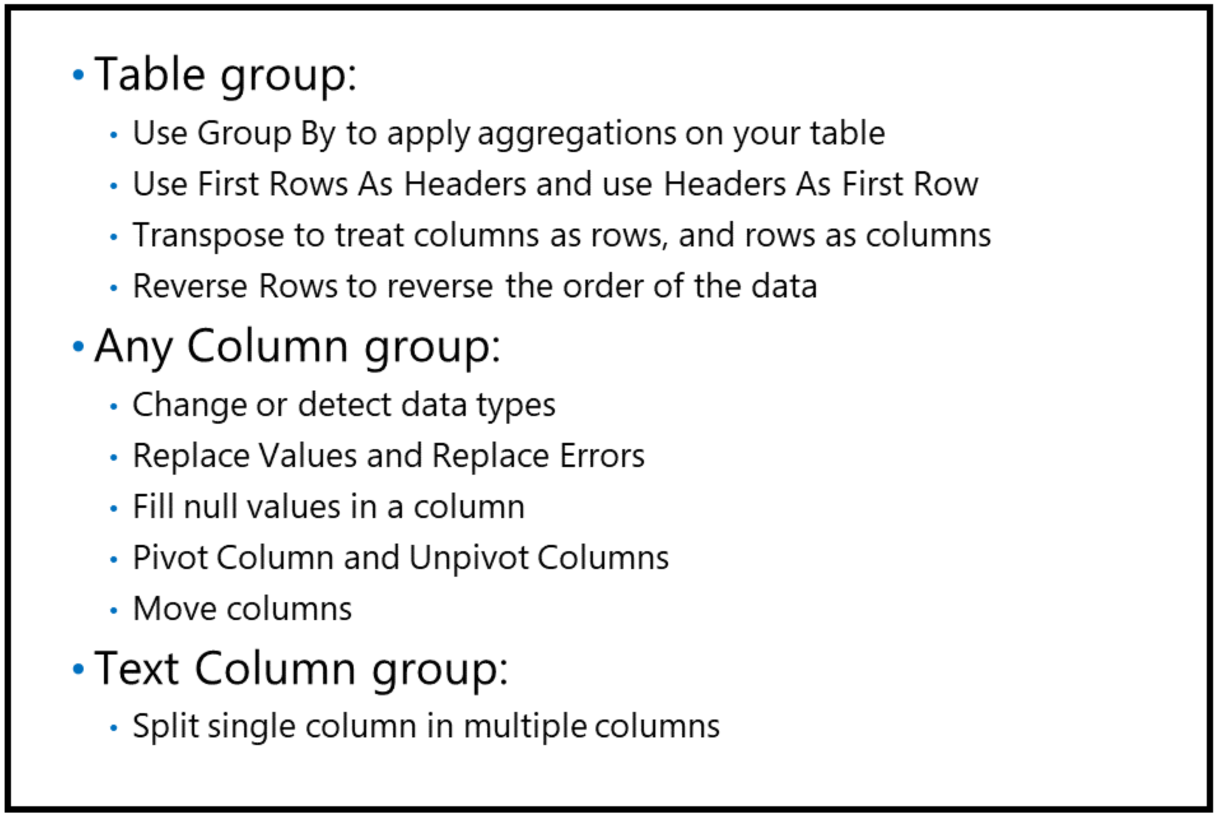
Subtract works in much the same way as the Add and Multiply functions; however, you can only use two columns in the calculation. Select the first column you want to use in the calculation, and then click the column to subtract from the first column, and click **Standard**, **Subtract**. This creates a new column named **Inserted Subtraction** by default. If you wanted to use more than two columns, you could use a custom column, with a formula such as **[RetailPrice] - [ManufacturingCost] - [StoreCommission]**. You can also select a single column, click **Standard**, **Subtract**, and then enter a whole or decimal number.

Note: The order in which you select your columns affects the calculation. For example, if you want to calculate **Profit**, click **RetailPrice**, click **ManufacturingCost**, and then click **Standard**, **Subtract**. The calculation is displayed in the Formula Bar, so if you have incorrectly ordered the columns, you can manually change the query. In this case, the query **Table.AddColumn(#"Changed Type", "Profit", each [ManufacturingCost] - [RetailPrice], type number)** is incorrect, because the ManufacturingCost should be subtracted from the RetailPrice.

Divide

The Divide function can also only operate on two columns. You should be aware of the order in which you select them, because this affects the calculation. Select the first column for the calculation, hold down the Ctrl key, and then click the second column for the number to divide by. Click **Standard**, **Divide**—a new column is created and by default is named **Inserted Division**. This returns a whole or decimal value. You can divide a single column by a specific value. Click the column, and then click **Standard**, **Divide**, and enter a whole or decimal number. Click **OK**.

Transforming data



- Table group:
  - Use Group By to apply aggregations on your table
  - Use First Rows As Headers and use Headers As First Row
  - Transpose to treat columns as rows, and rows as columns
  - Reverse Rows to reverse the order of the data
- Any Column group:
  - Change or detect data types
  - Replace Values and Replace Errors
  - Fill null values in a column
  - Pivot Column and Unpivot Columns
  - Move columns
- Text Column group:
  - Split single column in multiple columns

While Power BI is flexible in the variety of data sources that you can import data from, visualizations work best with data that is in a columnar format. For example, data that is imported from Excel may be easy for the human eye to digest visually, but the data might not be structurally appropriate for Power BI to translate the values in a bar chart. The Power Query Editor offers plenty of functions for you to transform data into a structure that Power BI can use effectively in reports. This lesson explores the functions available on the **Transformations** tab.

Table

The Table group offers some useful functions that you can quickly use to transform your data. Each of the functions is described next.

Group By

You can aggregate one or more columns in your table. Click **Group By** on the **Table** group, and select the columns you want to include. Ensure you include all columns that you want to include in the table, or they will be removed. In the **New column name** box, give the column a useful name, and choose the **Operation** from **Sum**, **Average**, **Median**, **Min**, **Max**, **Count Rows**, **Count Distinct Rows**, or **All Rows**. If the operation such as Sum requires a column to aggregate, select this from the list, and then click **OK**.

Use First Row As Headers

This function is useful when data has been imported and it already has a header row, but Power BI has not detected this. If you import columns with numeric values that include a header, Power BI can detect that the first row is a string compared to the other values—and guess that it is the header. This is not so obvious when all of the columns contain string values. To apply this function, click **Use First Row As Headers**, and select **Use First Row As Headers**. The values of the first row are promoted to the column header. You can also perform this operation in reverse. Click **Use First Row As Headers**, and select **Use Headers As First Row**. The headers become the first row in your table, and you can then rename the columns.

Transpose

With Transpose, you can treat rows as columns, and columns as rows. This is useful if you import a table from a spreadsheet with columns and rows that are readable to the user in a matrix format, but don't translate into a format that Power BI can use easily. Select the table that you want to apply this function to, and then click **Transpose** from the **Table** group. You can then begin applying other functions, such as unpivot, to give your data a columnar format.

Reverse rows

This function reverses the order of the rows in the table, so that the bottom rows are at the top, and the top rows are at the bottom.

Count rows

Use this function to return the number of rows in the current table. The rows are replaced with the count of rows.

Any Column

The functions in the **Any Column** group can be applied to columns, regardless of the data type or format. Each of the functions is described next.

Data Type

You can use the **Data Type** function to select from a list of data types—this is useful for converting columns where Power BI has incorrectly guessed the type. Select one or more columns in the table, click **Data Type**, and then select the data type for your conversion. Types include **Decimal Number**, **Fixed Decimal Number**, **Whole Number**, **Date/Time**, **Date**, **Time**, **Date/Time/Timezone**, **Duration**, **Text**, **True/False**, and **Binary**.

Detect Data Type

You can select one or more columns and use the built-in data type detection function. Select a column, hold down the Ctrl key, and then click any additional columns you want to add. From the **Any Column** group, select **Detect Data Type**. Power BI automatically corrects any columns it guesses to be wrong.

Rename columns

To rename a column, select the column in the table, and then click **Rename** from the **Any Column** group, or you can right-click the column and then click **Rename**. Type in the new name of the column, and press Enter—the name is updated.

Replace Values and Replace Errors

With these two functions, you can very quickly replace a value or an error in a column with another value. Both functions work on one or more columns, so select a single column, or hold down the Ctrl key to click and select multiple columns. Click **Replace Values** from the **Any Column** group, and in the dialog box, type a **Value To Find**, and a value to **Replace With**. If you are working with a text column, you can also click **Advanced Options** to **Match entire cell contents** and/or **Replace using**

special characters. Special characters include **Tab**, **Carriage Return**, **Line Feed**, and **Carriage Return and Line Feed**. Click **OK**. The values in the column are replaced. To replace an error, click **Replace Errors** instead of Replace Values, and type a replacement value in the **Value** box of the **Replace Errors** dialog box.

Fill

You can use the fill function to fill in null values with the value from an adjacent cell. Click the cell you want to use to fill the adjacent cells, and click **Fill**, and then click **Up** or **Down**, depending on which direction you want to fill. This works at the column level.

Pivot Column and Unpivot Columns

The **Pivot Column** function takes the values in the selected column, and uses them to create new columns. This is particularly helpful if you import data that has a matrix format from Excel, and you want to convert it to a columnar format for reporting. **Unpivot** can also help with this, by converting selected columns into attribute-value pairs.

Move

The **Move** function moves one or more columns to another location in the table. Click the column you want to move, or hold down the Ctrl key to select multiple columns, and click **Move** from the **Any Column** group, or right-click. You can move **Left**,

Right, **To Beginning**, or **To End**. The **To Beginning** option is useful if you add in an index column, because this is always appended to the right.

Text Column

The **Split Column** function splits a column based on a delimiter, or a specified number of characters. Much like the **Extract** function discussed in the previous topic, you can select from the list of delimiters, or use a custom delimiter. To split a column, select the column in the table, and from the **Text Column** group, click **Split Column**. Click **By Delimiter**, to open the **Split Column by Delimiter** dialog box. You can select a delimiter from **Colon**, **Comma**, **Equals Sign**, **Semicolon**, **Space**, **Tab**, or **Custom**. To use a custom character, click **Custom**, and enter the character, or symbol, such as a hyphen. You can split at the **Left-most delimiter**, **Right-most delimiter**, or **Each occurrence of the delimiter**. The number of new columns created will depend on which split option you choose.

To have further control over the split, click **Advanced options**. You specify the number of columns to split into, and the **Quote Style**, which is **CSV**, or **None**. You can also split using special characters, and choose from **Tab**, **Carriage Return**, **Line Feed**, and **Carriage Return and Line Feed**. Click **OK**. The column splits the values, and the original column is replaced. Use the **Duplicate Column** function on the **Add Column** tab if you want to retain this value.

Demonstration: Transforming data with the Power Query Editor

In this demonstration, you will see how to:

- Import data from Excel.
- Apply transformations to the table.

Check your knowledge

Select the best answer

Which of the following is not good advice for shaping your data?

Remove all columns and rows that are not used in the reports.

Rename columns to provide names that represent the column data, and can be used by Power BI Q&A.

Let Power BI guess the data types of your columns because it will always be correct.

Create an index column if you want to guarantee the sort order in a visual, or if you are appending data.

Use the Age function on a Date of Birth column to calculate the current age.

Check answer Show solution Reset

Lesson 3: Combining data

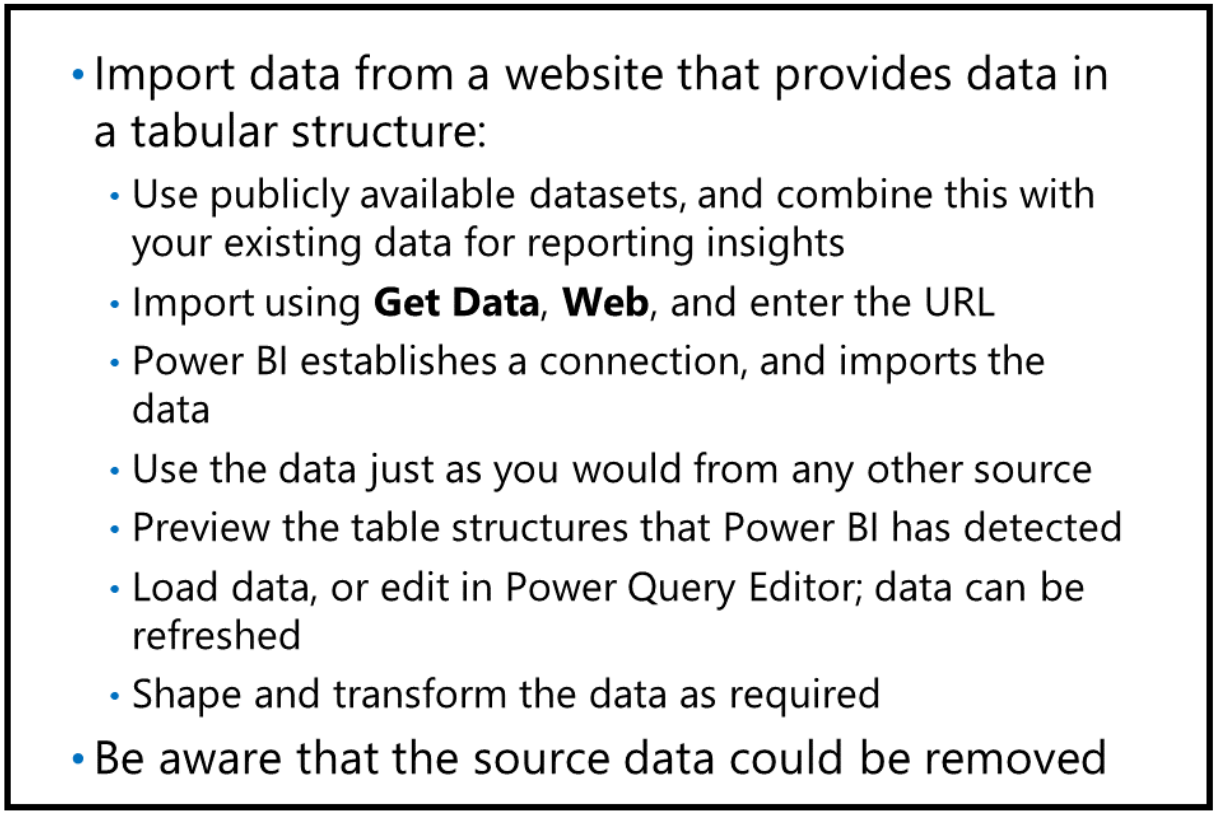
In this lesson, you will learn how to import data using an internet address as a data source, how to shape that data, and how to merge it with existing data in your dataset.

Lesson objectives

After completing this lesson, you will be able to:

- Import data into your dataset from the internet.
- Apply shaping to data you have imported from the internet.
- Merge data from different tables within your dataset.

Adding data from the internet



- Import data from a website that provides data in a tabular structure:
  - Use publicly available datasets, and combine this with your existing data for reporting insights
  - Import using **Get Data**, **Web**, and enter the URL
  - Power BI establishes a connection, and imports the data
  - Use the data just as you would from any other source
  - Preview the table structures that Power BI has detected
  - Load data, or edit in Power Query Editor; data can be refreshed
  - Shape and transform the data as required
- Be aware that the source data could be removed

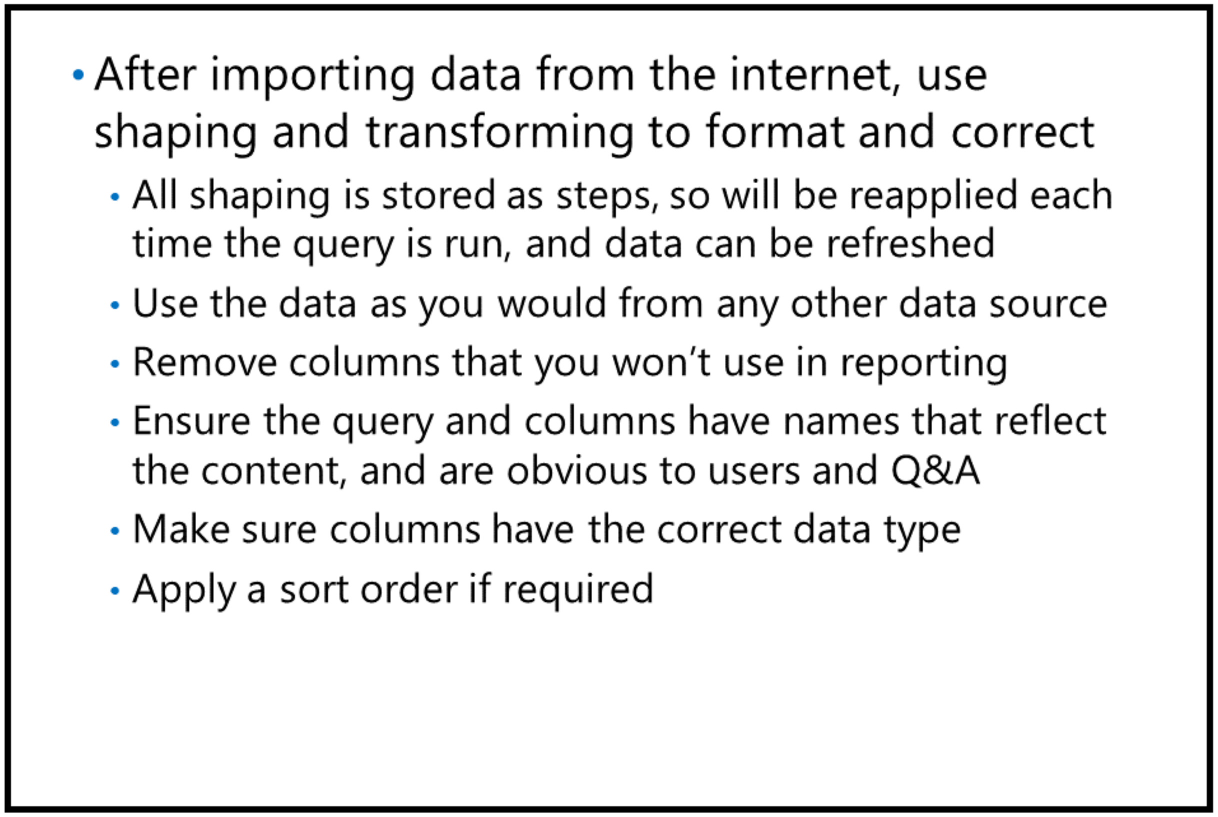
Power BI offers great flexibility for importing data. You use the web data source to pass a URL to Power BI so it can scrape the data into a new table. Data in the webpage you want to scrape should be in a tabular layout, so Power BI can determine the shape and import the data into a table structure. This is a useful way to import publicly available data, such as government statistics, or information gathered by organizations such as those monitoring climate change, or population socio-economics—you can combine this with your existing data to show trends or demographics.

Importing data

To import data from a webpage, open Power BI Desktop. From the **Getting Started** dialog box, or from the **Home** tab, click **Get Data**, and select **Web**. In the **From Web** dialog box, type or paste the web address into the **URL** box, and click **OK**. Power BI establishes a connection to the webpage, and determines the data that can be imported. In the **Navigator** dialog box, you are presented with a list of tables for the data that can be imported. You select the tables and preview them as you would any other data source. Click **Edit** to load the data into Power Query Editor and begin shaping the data. Alternatively, you can click **Load** to import the data into Power BI designer, where you can use it immediately in visualizations, or later apply transformations and shaping.

Note: Public websites such as Wikipedia offer a wealth of information that you can freely use in your reporting. However, you should be aware that you have no control over when the data is updated, whether or not it is accurate, or even if the page or data is retained or removed.

Shaping the new data



- After importing data from the internet, use shaping and transforming to format and correct
  - All shaping is stored as steps, so will be reapplied each time the query is run, and data can be refreshed
  - Use the data as you would from any other data source
  - Remove columns that you won't use in reporting
  - Ensure the query and columns have names that reflect the content, and are obvious to users and Q&A
  - Make sure columns have the correct data type
  - Apply a sort order if required

When you import data from the internet, it is unlikely that you will know how the data will initially be shaped until Power BI has established a connection to the page, and determined the format and possible tables that it can scrape. If you regularly import or refresh from the same source, and this doesn't change, you can have some confidence in the end results. However, after first importing internet data, you are very likely to want to perform some shaping and transformations. The transformations that you apply to the data are stored in the QUERY SETTINGS under APPLIED STEPS, so each time you refresh the data, the query includes the code to shape and transform the data from the web—you should always see the results you expect.

Shaping the data

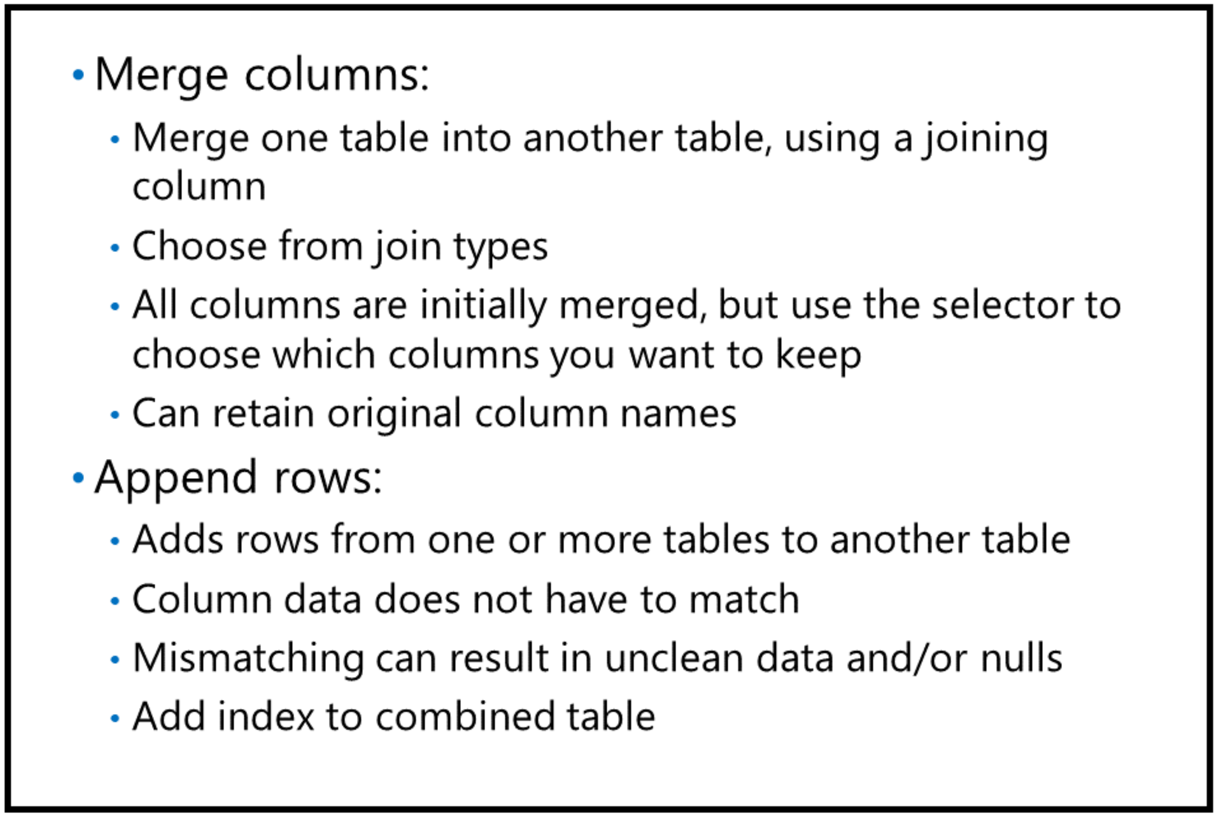
You can shape data from the internet exactly as you would with data from any other data source. As with any dataset, it is a good idea to remove the columns that will not be used in your reports and analysis, keeping the data succinct, and more efficient to work with. This reduces the size of the data, and does not present extraneous columns to colleagues who might share the query.

It is also important to ensure the name of the query (table) and the names of the columns are something obvious. Again, this keeps clarity within the dataset, and has the added advantage that you or other colleagues can understand the data just by looking at the names of the queries and columns. Furthermore, Power BI Q&A uses the natural query language and relies on being able to find answers to questions, based on relevant column names. The names should accurately describe the data.

When importing the data, the Power Query Editor makes a best guess at the data types for each of the columns. You will want to check the columns to ensure the type matches the data. Check that datetime columns have been detected correctly, especially if you want to use dates and times for drilldown. The Power Query Editor does not always recognize currencies unless there is a symbol included in the data—you should update any currency columns. Check that numerical columns have the correct data types, and include whole numbers and decimals, which you require for aggregations.

If the data needs a particular sort order, you can set this to be A-Z, or Z-A, or add the month number to a query that includes month in text format—so you can order on the numeric value in your visualizations.

Merging data



- Merge columns:
  - Merge one table into another table, using a joining column
  - Choose from join types
  - All columns are initially merged, but use the selector to choose which columns you want to keep
  - Can retain original column names
- Append rows:
  - Adds rows from one or more tables to another table
  - Column data does not have to match
  - Mismatching can result in unclean data and/or nulls
  - Add index to combined table

By using Power BI, you can gather data from different sources and of different types into a single dataset. You then combine the data in one report. You can import data from SaaS providers such as Bing and Salesforce, combine it with data from your Azure SQL Database in the cloud, an on-premises SQL Server Analysis Services (SSAS) data warehouse, and data from Excel. After importing into a single set of data, you can merge columns using tables from different sources and append rows.

Merging columns

To merge columns, the two tables must have a joining column, where the value will match the order to combine the values. From the Power BI Desktop designer, click **Edit Queries** to open the Power Query Editor window. Click the query (or table) into which you want to merge the other columns. From the **Home** tab, select **Merge Queries** from the **Combine** group. This opens the **Merge** dialog box. The top table is the one you elected as the destination table for merging the second table. Click to choose the column where you want to join. You select more than one column by holding the Ctrl key down while using your mouse to select. In the list, select the table from where you want to merge. In the second table, click to select the column, or columns, you are joining. The label at the bottom of the dialog box counts the matches, so you can usually determine if the match is correct. For example, if you are expecting all rows to match, and the label says, “The selection has matched 36 out of the first 48 rows”, then something is wrong.

You choose the type of join used to connect the two tables, by selecting from the **Join Kind** list. Types of join include **Left Outer** (all from first, matching from second), **Right Outer** (all from second, matching from first), **Full Outer** (all rows from both), **Inner** (only matching rows), **Left Anti** (rows only in first), or **Right Anti** (rows only in second). Use the default **Left Outer**, or select another join, and click **OK**. The second query is merged as a single column, with a value of **Table**. Click the double-arrow icon in the column header, and select the columns you want to include from the second table. You might not want to include the joining column, if all your rows matched or partially matched as expected. Clear the **Use original column name as prefix** if you want the columns to keep their original names, otherwise the column is named **NewColumn.<original name>**. After making your selection, click **OK**. The second table columns now appear as columns in the first table—though you may need to rename them.

Appending rows

When you append rows, you take rows from one or more tables, and add them to the first table. In most situations, the columns and data types will match. However, you can append rows between two tables that have all different columns—but the result is unclean data and no values when the number of columns between the tables does not match. From Power BI Desktop designer, click **Edit Queries** to open the Power Query Editor window. Click the query (table) into which you want to append the rows, and click **Append Queries** from the **Combine** group on the **Home** tab. This opens the **Append** dialog box. From the **Select the table to append** list, choose the table you want to add in, and then toggle **Two tables**, or **Three or more tables**. If you are appending two tables, click **OK**. If you have clicked **Three or more tables**, in the **Available Table(s) list**, select each table you want to append, and click **Add**. You can append a table to itself if you need to. Click **OK**.

> **Best Practice:** If you are appending rows from multiple sources, and the table contains index values that overlap when the data is combined, combine the data, and then create a new index column on the table into which the rows have been appended.

Demonstration: Adding and shaping data from the internet

In this demonstration, you will see how to:

- Import data from the internet.
- Shape the data that is imported.

Check your knowledge

Select the best answer

Which of the following is not a true join type for merging columns?

Left Outer (all from first, matching from second).

Right Outer (all from second, matching from first).

Full Outer (all rows from both).

Inner (matching rows only).

Random (let Power BI decide).

Check answer Show solution Reset

Lab: Shaping and combining data

Scenario

Adventure Works employees are becoming increasingly frustrated by the time it takes to implement managed BI services. The existing managed BI infrastructure, including a data warehouse, enterprise data models, and reports and dashboards, are valued sources of decision-making information. However, users increasingly want to explore relationships with other, currently unmanaged data. It takes too long for the IT department to include these requirements in the corporate BI solution.

As a BI professional, you have been asked to explore ways in which Adventure Works can empower business users to augment their managed enterprise BI solution with self-service BI.

Objectives

After completing this lab, you will be able to:

- Connect to a SQL Server database and import data.
- Apply formatting to the data you have imported to shape it, ready for reporting.
- Combine related data to the shaped data.

Note: Because of updates to Microsoft Power BI, the lab steps for this course change frequently. Microsoft Learning regularly updates the lab steps, so they are not available in this manual − but you can access them on GitHub.

Lab setup

Estimated time: 60 minutes

Virtual machine: **20778C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

All the lab steps are contained in 20778C_LAB_04.md.

Exercise 1: Shape Power BI data

Scenario

You are exploring how Power BI can help shape and combine data that comes from multiple sources. Currently, much of the data is exported from SQL Server into Excel. You have two worksheets, one for sample sales data for the North America territory, and one for the European territory. After importing the data into Power BI, you will shape the data using transformations and formatting.

The main tasks for this exercise are as follows:

1. Preparing the environment.
2. Import data from Excel.
3. Apply formatting to the existing data.

Result: At the end of this exercise, the data will be imported from Excel, and shaped ready to be combined.

Exercise 2: Combine Power BI data

Scenario

You have imported the two worksheets for sales in Europe and North America, and applied some shaping. You now want to combine the rows from the North America query, into the Europe query. You also want to include a Country Code column.

The main task for this exercise is as follows:

- Add related data to the shaped data

Result: At the end of this exercise, the Europe and North America data will be appended, and the Country Code column will be added to the query.

Review question(s)

Check your knowledge

Discovery

Discuss the types of different data in your organization that could be combined using the Power Query Editor. Do you have data stored across locations that could be appended, or lookup data that could be merged into other tables to make it more useful for reporting?

Show solution Reset

Module review and takeaways

In this module, you have learned how to:

- Perform a range of query editing skills in Power BI.
- Shape data, using formatting and transformations.
- Combine data together from tables in your dataset.

Review question(s)

Check your knowledge

Discovery

Discuss the benefits of using Power BI, rather than Excel, to shape and transform your data. Are there any disadvantages? What can Power BI do that Excel cannot, and vice versa? Which tool do you think is most straightforward to use?

Show solution Reset