

# INTRODUCTION

---

## What is Power BI?

Power BI is a collection of software services, apps, and connectors that work together to turn your unrelated sources of data into coherent, visually immersive, and interactive insights. Whether your data is a simple Excel spreadsheet, or a collection of cloud-based and on-premises hybrid data warehouses, Power BI lets you easily connect to your data sources, visualize (or discover) what's important, and share that with anyone or everyone you want.

Power BI can be simple and fast – capable of creating quick insights from an Excel spreadsheet or a local database. But Power BI is also robust and enterprise-grade, ready for extensive modeling and real-time analytics, as well as custom development. So it can be your personal report and visualization tool, and can also serve as the analytics and decision engine behind group projects, divisions, or entire corporations.

---

## Parts of Power BI

- Power BI consists of a Windows desktop application called Power BI Desktop, an online SaaS (Software as a Service) service called the Power BI service, and mobile Power BI apps available on Windows phones and tablets, as well as for iOS and Android devices.
- These three elements – the Desktop, the service, and Mobile – are designed to let people create, share, and consume business insights in the way that serves them, or their role, most effectively.

---

## Basic Comparison Power BI Desktop and Power BI Service

### POWER BI DESKTOP

- Power BI Desktop is an application that you download and install for free on your local computer.
- Desktop is a complete data analysis and report creation tool that is used to connect to, transform, visualize, and analyze your data.
- It includes the Query Editor, in which you can connect to many different sources of data, and combine them (often called modeling) into a data model. Then you design a report based on that data model.
- Reports can be shared with others directly or by publishing to the Power BI service. Sharing reports requires a Power BI Pro license.

### POWER BI SERVICE

- The Power BI service is a cloud-based service, or software as a service (SaaS).
- It supports report editing and collaboration for teams and organizations.
- You can connect to data sources in the Power BI service, too, but modeling is limited.
- The Power BI service is used to do things such as creating dashboards, creating and sharing apps, analyzing and exploring your data to uncover business insights, and much more.
- You can't sign up or purchase using email addresses provided by consumer email services or telecommunication providers. These services include outlook.com, hotmail.com, gmail.com, and others.
- Your license determines what you can do in the Power BI service. For more information about licenses, see Power BI licenses and subscriptions:  
<https://powerbi.microsoft.com/en-us/pricing/>

## Work in the Power Service

### COLLABORATE

- After you have created your reports, you can save them to a workspace in the Power BI service, where you and your colleagues collaborate.
- You can build dashboards on top of those reports or add them to apps.
- You can share those dashboards, reports and apps with others inside and outside your organization.
- When you share, you assign permissions that determine what the recipient can do with the dashboards, reports, apps, and underlying datasets.
- Sharing requires you to have a Power BI Pro license.
- Viewing shared reports requires a Pro license or for the report to be saved in Premium capacity.
- Consumers granted access to your report can view them in the Power BI service in Reading view, not Editing view.
- Consumers don't have access to all the features available to report creators.
- You can also share your datasets and let others build their own reports from them.

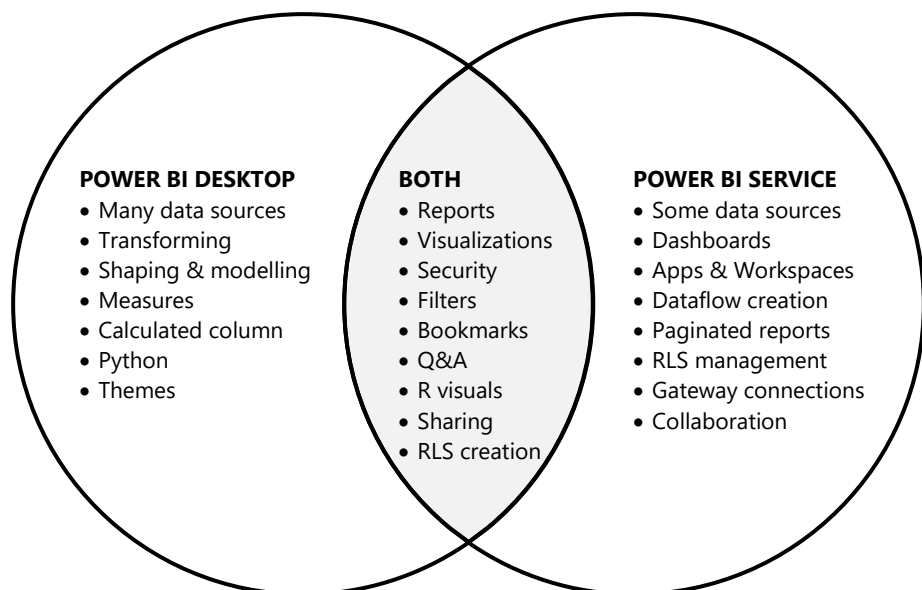
### SELF-SERVICE DATA PREP WITH DATAFLOWS

- Dataflows help organizations unify data from different sources and prepare it for modeling.
- Analysts can easily create dataflows using familiar self-service tools.
- Analysts use dataflows to ingest, transform, integrate, and enrich big data by defining data source connections, ETL logic, refresh schedules, and more.

## Venn Diagram comparison

This Venn diagram compares Power BI Desktop and the Power BI service.

- The middle shows some of the areas where they overlap. Some tasks you can do in either Power BI Desktop or the service.
- The two outer sides of the Venn diagram show the features that are unique to either the Desktop application or to the Power BI service.



---

## What is POWER QUERY

Power Query is a data transformation and data preparation engine.

The transformation engine in Power Query includes many prebuilt transformation functions that can be used through the graphical interface of the Power Query Editor.

These transformations can be as simple as removing a column or filtering rows, or as common as using the first row as a table header.

There are also advanced transformation options such as merge, append, group by, pivot, and unpivot.

We use POWER QUERY for:

Data Connection:

Power Query allows users to connect to a wide variety of data sources, including databases, spreadsheets, APIs, and more.

Data Transformation:

It offers a rich set of features for transforming data, such as filtering, sorting, pivoting, unpivoting, merging, appending, and changing data types.

Data Cleaning:

Power Query helps clean data by handling missing values, inconsistencies, and errors, ensuring that the data loaded into Power BI is clean and accurate.

Data Combination:

It allows users to combine data from multiple sources into a single, integrated dataset, making it easier to analyze and visualize.

Data Preparation:

Power Query prepares the data for analysis and visualization in Power BI, making it easier to create reports and dashboards.

## M Language in Power Query

### Power Query M formula language

In any data transformation scenario, there are some transformations that can't be done in the best way by using the graphical editor.

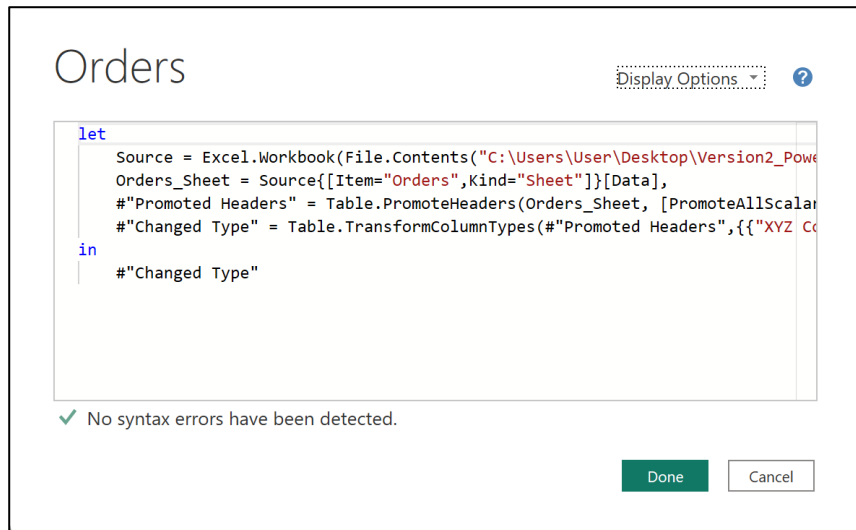
Some of these transformations might require special configurations and settings that the graphical interface doesn't currently support.

The Power Query engine uses a scripting language behind the scenes for all Power Query transformations: the Power Query M formula language, also known as M.

The M language is the data transformation language of Power Query. Anything that happens in the query is ultimately written in M.

You can use the Advanced Editor to access the script of the query and to fine-tune your functions and transformations:

1. Go to HOME menu → ADVANCED EDITOR button:



# DAX (DATA ANALYSIS EXPRESSIONS)

---

## About DAX

DAX (Data Analysis Expressions) is a collection of functions, operators, and constants that can be used in a formula, or expression, to calculate and return one or more values. Stated more simply, DAX helps you create new information from data already in your model. It is the programming language of Power Pivot that is used inside the Power Pivot for Excel, Power BI and SQL Server Analysis Services Tabular (SSAS Tabular).

---

## Calculated Column vs Measure

When you start using Power BI, you will realize that often, there is more than one way to accomplish the same goal. For example, when you perform an analysis of your data in Power BI, you will likely need to enrich your data model with calculations, many of which could be done by using calculated columns, measures, or both. Both calculated columns and measures are created using DAX (Data Analysis Expressions).

### **CALCULATED COLUMN**

A calculated column is an extension of a table using a DAX formula that is evaluated for each row. These columns are distinguished from columns that you obtain from your data source or using Power Query Editor because calculated columns are computed based on data that has already been loaded into your data model.

When you write a calculated column formula, it is automatically applied to the whole table and evaluated individually for each row. The values in calculated columns are evaluated when you first define them or when you refresh your dataset. Once evaluated, the values are stored in your data model, which means your data model size increases and it consumes more RAM.

### **MEASURE**

A measure is a formula that is evaluated in the context in which it is used. For example, if you use a measure in a matrix visual, the formula will be evaluated for each matrix row and column combination separately. The closest equivalent of a measure from the Excel world is a PivotTable calculated field: you can see its result only when you use it in the PivotTable.

Because measures do not store their values directly in the data model, it is safe to say that they do not use any RAM for storage purposes. Instead, because they are evaluated with every interaction – for example, slicing or cross-filtering visuals – they use CPU at query time.

---

## DAX Function Reference

You may explore more information of DAX function reference in:

<https://learn.microsoft.com/en-us/dax/dax-function-reference>

## Quick Comparison Between Calculated Column and Measure

Calculated Column	Measure
Stored in memory	Is not stored in memory
New Column is added in the table	Do not need any new column; need a table to keep the measure
Calculates at the time of Refresh Report	Calculated on the fly
Consumes Memory	Consumes CPU
Pre-calculated and stored	Never pre-calculated
Row by row calculation usually	Usually is a result of an aggregation
Slicers, row/ column, axis	Only in values area
Use when result is text	Use when result is numeric
Value can be seen in the column in DATA tab/view	Value can be seen when adding in the report
In the majority of the cases can be done in Power Query	DAX usually is the best place for this calculation
Increases file size	Does not increase file size
Example: Profit <i>Profit = Sales - Cost</i>	Example: Sales Year to Date <i>Sales YTD = TotalYTD(Sum(Sales), DateField.[Date])</i>

### Note:

Data aggregation is any process whereby data is gathered and expressed in a summary form. When data is aggregated, atomic data rows typically gathered from multiple sources are replaced with totals or summary statistics. In a data warehouse, atomic data is the lowest level of detail.

Aggregation is often done on a large scale, through software tools known as data aggregators. Data aggregators typically include features for collecting, processing and presenting aggregate data.

Data aggregation is often used to provide statistical analysis for groups of people and to create useful summary data for business analysis.

---

## Add New Column in Power Query or DAX?

- Generally, creating new columns in Power Query Editor is the preferred approach in Power BI.
- This is because Power Query is designed for data transformation, and adding columns there allows you to manipulate the data before it's loaded into the Power BI model.
- This can lead to a cleaner and more efficient data model.

Here's a more detailed breakdown:

- Why Power Query is often better:
  - Data Transformation Focus: Power Query is specifically designed for transforming and cleaning data before it's used in reports and visualizations.
  - Query Performance: Adding columns in Power Query can improve query performance, especially when dealing with large datasets, as it can optimize data storage and processing within the model.
  - Customization and Flexibility: Power Query provides extensive options for customizing column calculations and logic using the M language.
  - Maintainability: Modifying columns in Power Query is easier than modifying DAX calculations, especially when the logic is complex or requires adjustments.
- Why DAX Calculated Columns are used:
  - Dynamic Calculations: DAX calculated columns can be used to create calculations that depend on data in different rows or tables within the Power BI model.
  - Row-Level Logic: DAX calculated columns allow for calculations that are performed at the row level, providing more flexibility than Power Query.

# TRANSFORMING DATA

---

## Common Transformation Tasks

We often receive data that is unpolished, or raw. That is to say, the data may have duplicates or blank fields or inconsistent text, for example.

Data transformation generally entails certain actions that are meant to “clean” your data — actions such as establishing a table structure, removing duplicates, cleaning text, removing blanks, and even adding your own calculations.

---

## REMOVING DUPLICATE RECORDS

Duplicate records are absolute analysis killers. The effect that duplicate records have on your analysis can be far-reaching, corrupting almost every metric, summary, and analytical assessment you produce. It is for this reason that finding and removing duplicate records should be your first priority when you receive a new dataset.

---

## FILING IN BLANK FIELDS

There are two kinds of blank values: null and empty string. A null is essentially a numerical value of nothing, whereas an empty string is equivalent to entering two quotation marks (“”) in a cell.

Blank fields aren’t necessarily a bad thing, but having an excessive number of blanks in your data can lead to unexpected problems when analyzing it.

Your job is to decide whether to leave the blanks in the dataset or fill them with actual values. Consider the following best practices:

Use blanks sparingly:

Working with a dataset is a much less daunting task when you don’t have to test continually for blank values.

Use alternatives whenever possible:

Represent missing values with some logical missing-value code whenever possible.

Never use null values in number fields:

Use zero instead of null in a currency or a number field that will be used in calculations.

---

## CONCATENATING COLUMNS

You can easily concatenate (join) the values in two or more columns. In Power Query, you do this by using the Merge Columns command. The Merge Columns command concatenates the values in two or more fields and outputs the newly merged values into a new column.



---

## CHANGING CASE

Making sure that the text in your data has the correct capitalization may sound trivial, but it's important. Imagine that you receive a customer table that has an address field where all addresses are lowercase. How will that look on labels, form letters, or invoices? Fortunately, Power Query has a few built-in functions that make changing the case of your text a snap.

---

## FINDING AND REPLACING SPECIFIC TEXT

Imagine that you work in a company named ABC Company. One day, the president of your company informs you that the abbreviation AC on all addresses is now deemed an infringement of your company's trademarked name and must be changed to Boulevard as soon as possible. How would you go about meeting this new requirement? The Replace Values function is ideal in a situation like this. Select the Address field, and then click the Replace Values command on the Home tab.

- **Match entire cell contents:**  
Selecting this option tells Power Query to replace values that contain only the text entered into the Value to Find field. This option comes in handy when you want to replace zeros (0) with n/a but not affect any zeros that are part of a number
  - only those that are alone in a cell.
- **Replace Using Special Characters:**  
Selecting this option allows you to use special invisible characters such as line feed, carriage return, or tab as replacement text. This option is useful when you want to force an indent or reposition the text so that it shows up on two lines.

---

## TRIMMING AND CLEANING TEXT

When you receive a dataset from a mainframe system, a data warehouse, or even a text file, it isn't uncommon to have field values that contain leading and trailing spaces. These spaces can cause some abnormal results, especially when you're appending values with leading and trailing spaces to other values that are clean.

---

## EXTRACTING THE LEFT, RIGHT, AND MIDDLE VALUES

In Excel, the RIGHT function, the LEFT function, and the MID function allow you to extract portions of a string starting from different positions:

**Left:**

Returns a specified number of characters, starting from the leftmost character of the string.

**Right:** Returns a specified number of characters starting from the rightmost character of the string. The required arguments for the Right function are the text you're evaluating and the number of characters you want returned.

**Mid:** Returns a specified number of characters starting from a specified character position. The required arguments for the Mid function are the text you're evaluating, the starting position, and the number of characters you want returned.

---

## PIVOTING AND UNPIVOTING FIELDS

Power Pivot offers an easy way to unpivot and pivot columns, allowing you to quickly convert matrix-style tables to tabular datasets (and vice versa).

### **Unpivot Columns command**

The Unpivot Columns command lets you select a set of columns and convert those columns into two columns: one column consisting of the old column labels and another containing the old column data.

### **Unpivot Other Columns command**

As helpful as the Unpivot Columns command is, it has a flaw: You have to explicitly select the months that you want unpivoted. But what if the number of columns is ever growing? What if you unpivot January through June, but next month a new dataset will arrive with July and then August and then September? Because the Unpivot Columns command forces you to essentially hard-code the columns you want unpivoted, you have to redo the unpivot each and every month. Fortunately, you can avoid this problem with the Unpivot Other Columns command. This nifty command allows you to unpivot by selecting the columns that you want to remain static and telling Power Query to unpivot all other columns.

### **Pivot Columns command**

If you find that you need to transform your data from a tabular layout to a matrix-style layout, you can use the Pivot Columns command. Simply select the columns that will make up the header labels and values for the new matrix columns, and then select the Pivot Column command.

# CREATING CUSTOM COLUMNS

---

## Intro

When transforming your data, you sometimes have to add your own columns to extract key data points, create new dimensions, or even create your own calculations. The inputs are:

- **New column name:**  
An input box where you enter a name for the column you're creating.
- **Available columns:**  
A list box that contains the names of all columns in the query. Double-click any column name in this list box to automatically place it in the formula area.
- **Custom column formula:**  
The area where you type the formula.

Before diving into building Power Query formulas, you should understand how Power Query formulas differ from those in Excel. Here are some high-level differences to be aware of:

No cell references.

You can't reach outside the Add Custom Column dialog box to select a range of cells. Power Query formulas work by referencing columns, not cells.

Excel functions don't work.

The Excel functions you're used to don't work in Power Query. Power Query has many of the same kinds of functions as Excel, but it has its own formula language.

Everything is case sensitive.

In Excel, you can type in all lowercase or all uppercase letters and your formulas will work. Not so in Power Query. To Power Query, sum, Sum, and SUM are three different items, and only one of them is acceptable.

Data types matter.

Some fields are text fields, other fields are number fields, and still others are date fields. Excel does a good job of handling formulas that mix fields of differing data types. The Power Query formula language, which is extremely sensitive to data types, doesn't have the built-in intelligence to gracefully handle data type mismatches. Data type issues are resolved with conversion functions.

No tool tips or intelligence help.

Excel is quick to throw up a tool tip or a menu of options when you start entering a new formula. Power Query has none of that. As of this writing, Power Query offers only a Learn About Power Query Formulas link to a Microsoft site dedicated to Power Query.

## Understanding data type conversions

When working with formulas in Power Query, you inevitably need to perform some action on fields that have differing data types, as in the exercise in the previous section, where I show you how to merge the Type column (a text field) with the Code column (a numeric field). In that example, you use a conversion function to change the data type of the Code field so that it can be temporarily treated as a text field. A conversion function does exactly what it sounds like: It converts data from one data type to another.

<i>Convert From</i>	<i>To</i>	<i>Function</i>
Date	Text	Date.ToText()
Time	Text	Time.ToText()
Number	Text	Number.ToText()
Text	Number	Number.FromText()
Text Dates	Date	Date.FromText()
Numeric Dates	Date	Date.From()

## Spicing up custom columns with functions

With a few basic fundamentals and a little knowledge of Power Query functions, you can create transformations that go beyond what you can do by using the Query Editor.

<i>Excel Function</i>	<i>Power Query Function</i>
LEFT([Text], [Number])	Text.Start([Text], [Number])
RIGHT([Text], [Number])	Text.End([Text], [Number])
MID([Text], [StartPosition], [Number])	Text.Range([Text], [StartPosition], [Number])
FIND([Find], [Within])	Text.PositionOf([Within], [Find])+1
IF([Expression], [Result1], [Result2])	if [Expression] then [Result1] else [Result2]
IFERROR([Procedure], [FailResult])	try [Procedure] otherwise [FailResult]

## Adding conditional logic to custom columns

Power Query has a built-in if function. The if function is designed to test for conditions and provide different outcomes based on the results of those tests. In this section, you'll see how you can control the output of your custom columns by utilizing Power Query's if function.

As in Excel, Power Query's if function evaluates a specific condition and returns a result based on a true or false determination.

You can also use the if function to save steps in your analytical processes and, ultimately, save time. For example, you may need to tag customers as either large customers or small customers, based on their dollar potential. You decide to add a custom column that contains either "LARGE" or "SMALL" based on the revenue potential of the customer.

# UNDERSTANDING JOIN TYPES IN POWER QUERY

---

## Intro

Data is frequently analyzed in layers, with each layer of analysis using or building on the previous layer. You may not know it, but you already build layers all the time. For instance, when you build a pivot table using the results of a Power Query output, you're layering your analysis. When you build a query based on a table created by a SQL Server view, you're also creating a layered analysis.

Sure, you would probably love to be able to analyze a single data source and call it a day. But that's not how data analysis works. You often find the need to build queries on top of other queries to get the results you're looking for. That's what this chapter is all about. In this chapter, I help you examine a few ways you can advance your data analysis by making your queries work together.

---

## Understanding the Merge Feature

In your data adventures, you often find the need to build queries that join the data between two tables. For example, you may want to join an employee table to a transaction table to create a view that contains both transaction details and information on the employees who logged those transactions.

Similar to VLOOKUP in Excel, the Merge feature joins the records from one query to the records in another by matching on a unique identifier. An example of a unique identifier is Customer ID or Invoice Number. You can join two datasets in one of several ways. The kind of join you apply is important because it determines which records are returned from each dataset.

Power Query supports six kinds of joins:

Left Outer:

Tells Power Query to return all records from the first query, regardless of matching, and only those records from the second query that have matching values in the joined field.

Right Outer: Tells Power Query to return all records from the second query, regardless of matching, and only those records from the first query that have matching values in the joined field.

Full Outer: Tells Power Query to return all records from both queries, regardless of matching.

Inner: Tells Power Query to return only those records from both queries that have matching values.

Left Anti: Tells Power Query to return only those records from the first query that don't match any of the records from the second query.

Right Anti: Tells Power Query to return only those records from the first query that don't match any of the records from the second query.

