

# Power BI

## What is Power BI?

*Power BI is a cloud-based business analysis and intelligence service by Microsoft. It is a collection of business intelligence and data visualization tools such as software services, apps and data connectors. We can use the datasets imported in Power BI for data visualization and analysis by making sharable reports, dashboards, and apps. Power BI is a user-friendly tool offering impressive drag-and-drop features and self-service capabilities.*

## Why Power BI?

Microsoft offers three types of Power BI platforms:

- Power BI Desktop (A desktop application)
- Power BI Service (SaaS i.e., Software as a Service)
- Power BI Mobile (For iOS and Android devices)

Power BI is an umbrella term having several different kinds of services.

1. There is a cloud-based BI service called **Power BI Services** used to view and share dashboards.
2. A desktop-based reporting interface known as **Power BI Desktop**. Another useful service is **Power BI Embedded** that runs on an Azure cloud platform and we can use it for *report creation, ETL and data analysis*.
3. **Real-time analysis** in Power BI can be done by establishing direct connections to the data sources. Also, it keeps data updated to the latest second by data refreshing.
4. You can use **custom visualizations** from a custom visuals gallery. Custom visuals are divided into many options and categories.

5. You can quickly search for important insights and datasets within your data by using the **Quick Insights** option.
6. Establish a live or non-live connection to on-premises data sources like SQL Server, and use a secure channel to access data through **data gateways**. This makes Power BI enterprise-ready as on-premises connections make data transfer secure and the technology scalable and reliable.
7. You can connect to other services through Power BI such as **SQL Server Analysis Services (SSAS)**, **Microsoft Excel**, etc.

## Power BI Features

These are some of the most important and interesting features of Power BI:

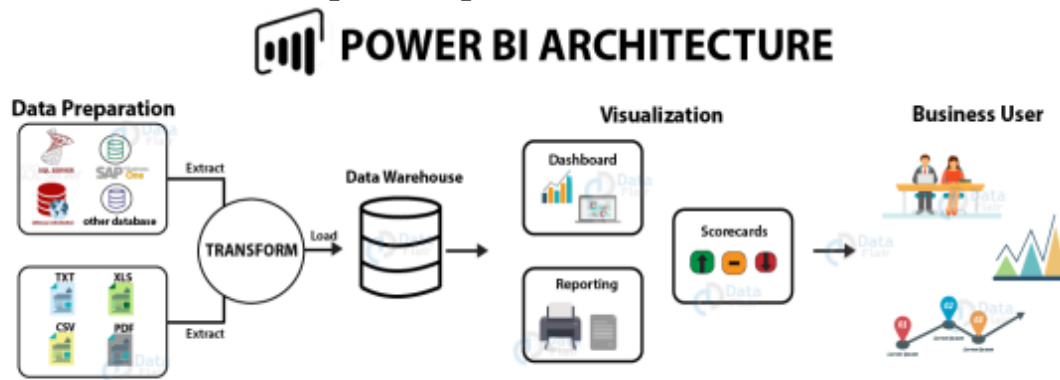
- Attractive Visualizations/ custom visualizations
- Get Data (Data sources)/data connections
- Datasets
- Dashboards
- Filters
- Reports
- Trend indicators
- Online Analytical Processing (OLAP)
- Navigation pane
- Natural language Q & A box
- DAX functions and formula
- Authoring interactive reports

## Power BI Components

Power BI is a business intelligence and data mining software suite which is a collection of different kinds of services by Microsoft. These services play a specific role and work in coordination with each other, to make Power BI function.

- **Power Query:** We use this service to access, search and transform data from public or local/internal data sources.
- **Power Pivot:** This service provides tools to model data taken from the in-memory data source to use it for analytics.
- **Power View:** This service has many tools to graphically represent data using visuals and use them for analysis.
- **Power Map:** It comes with tools and capabilities to visualize Geo-spatial data or information in the 3D model in a map. You can use these maps in a Power BI report.
- **Power BI Desktop:** It is a companion development tool for *Power View*, *Power Query*, and *Power Pivot*. You can import data from a data source, prepare and transform it and use it in visualizations to create reports in Power BI Desktop.
- **Power BI Website:** It is a web platform to view and share Power BI apps or solutions. Using Power BI Website, you can create dashboards from reports, share the dashboards with other Power BI users and slice and dice data within a report.
- **Power Service:** The Power Service enables the sharing of workbooks and data views with other users. The data gets refreshed at regular intervals from the on-premises or/and cloud-based data sources.
- **Power Q&A:** Using the Power Q&A option, you can search for your data or discover insights by entering queries in natural language. It instantly understands your query and returns relevant results.

To have a better understanding of Power BI, we can divide the architecture into three parts or phases:



## 1.Data Integration

In Power BI, we can import data from different kinds of data sources in different formats. In the data integration step, Power BI brings data together (extracted) from different data sources and converts it into a standard format. After data is integrated into Power BI, it is stored in a common storage area known as the staging area.

## 2. Data Processing

Once Power BI integrates and stores data at a secure place, the raw data requires some processing. Several processing or cleansing operations transform the raw data such as removing redundant values, etc. Later, we apply relevant business rules on the processed data that transforms it according to our business needs. This transformed data is loaded into the data warehouses. This completes a full process of ETL.

## 3. Data Presentation

In this final phase, the processed data moves from the warehouse and goes into the Power BI platforms like Power BI Desktop to *create reports, dashboards, and scorecards*. Power BI offers a wide range of visualizations. We can also import custom visualization from the marketplace. From the report development platforms, we can publish

the reports on the web or mobile apps to share it with other business users.

## **Users of Power BI**

Power BI users are categorized into four sections according to the purpose of the usage of Power BI. These four types of users are *Analysts, Business users, IT professionals and Developers*. Let's learn some more about them.

### **1. Analysts**

Analysts use Power BI to develop reports, dashboards, data models and study them to discover valuable insights in the data. Power BI offers a wide range of data sources from which an analyst can *extract data, make a common dataset, cleanse and prepare that data to make reports and conduct analysis*.

### **2. Business Users**

*The business users are the common users who study the reports and dashboards available to share with them on the Power BI website or mobile app.* Business users remain updated with the latest information which helps in taking important decisions in time.

### **3. IT professionals**

The IT professionals are mainly concerned with the *scalability, availability, and security of data*. They also centrally manage all the Power BI services and users.

### **4. Developers**

Developers are responsible for all the technical work. Their key roles are to *create custom visuals to be used in Power BI, embedding Power BI into other applications, creating reports, etc.*

## Data Connections in Power BI

There are a plenty of data sources from which you can extract data into Power BI. You can connect to data files on your local system, Excel files, Azure SQL Database, Facebook, Google Analytics, Power BI datasets, etc.

You can connect to cloud-based sources, on-premises data sources using gateways, online services, direct connections, etc. We have listed some commonly used data sources below.

- **File:** Excel, Text/CSV, XML, PDF, JSON, Folder, SharePoint.
- **Database:** SQL Server database, Access database, Oracle database, IBM, MySQL, Teradata, Impala, Amazon Redshift, Google BigQuery, etc.
- **Power BI:** Power BI datasets and Power BI dataflows.
- **Azure:** Azure SQL, Azure SQL Data Warehouse, Azure Analysis Services, Azure Data Lake, Azure Cosmos DB, etc.
- **Online Services:** Salesforce, Azure DevOps, Google Analytics, Adobe Analytics, Dynamics 365, Facebook, GitHub, etc.
- **Others:** Python script, R script, Web, Spark, Hadoop File (HDFS), ODBC, OLE DB, Active Directory, etc.

## Power BI Pricing

Now, we are sure you must have started liking Power BI after learning what all it has to offer. And so, you would also want to know its pricing and licensing costs. Microsoft has put out three pricing plans for Power BI:

- The **basic version**, Power BI Desktop is *free of cost* and includes tools for data visualization, data preparation, data modeling, data cleansing and publishing reports to Power BI Service.

- **Power BI Pro** is available at a subscription price of \$9.99 per user per month. You can try a 60 day free trial before purchasing the subscription. This plan for Power BI Pro includes tools for data collaboration, a 360 real-time view for dashboards, data governance, and the freedom to publish reports anywhere.
- The **Power BI Premium** is available at a price of \$4,995 per month for one dedicated storage resource and cloud computing facility.

## Power BI Building Blocks

We are going to explore the components of Power

BI: Visualizations, Datasets, Reports, Dashboards, and Tiles.



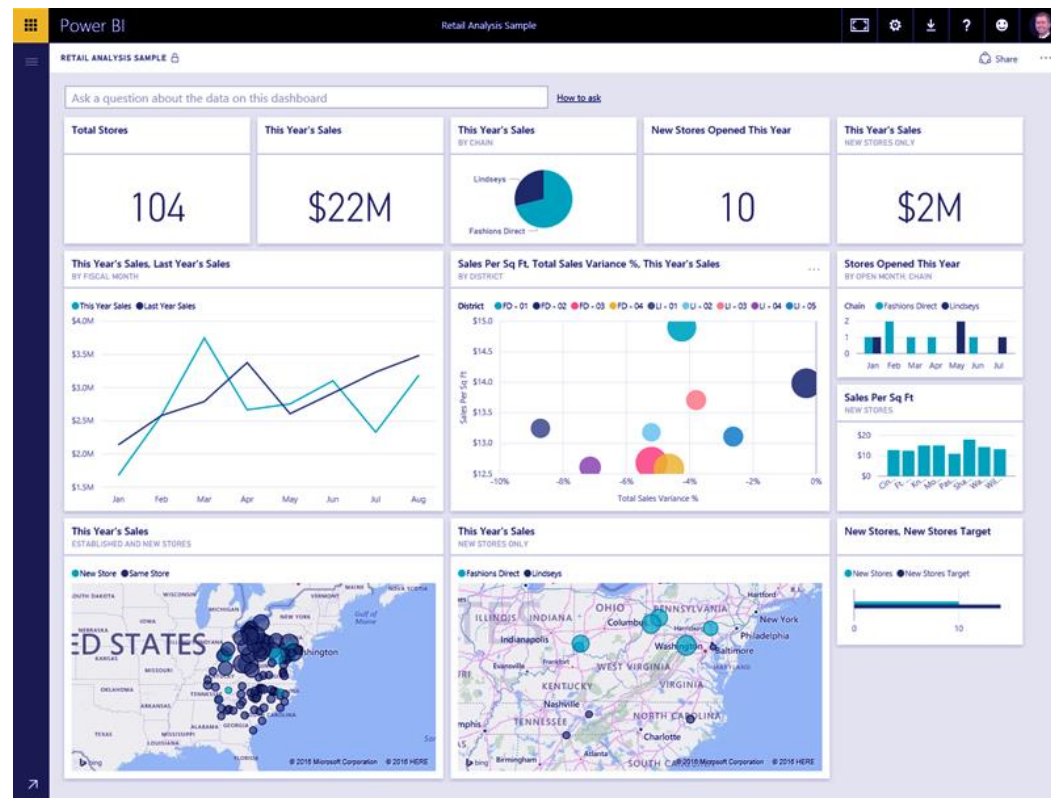
The fundamental Power BI building block are:

- Visualizations
- Datasets
- Reports
- Dashboards
- Tiles

### a. Visualization

A perception is a visual portrayal of information. For example, a diagram, chart, shading coded outline, other intriguing things you can make to

speak to your information outwardly. Power BI has a wide range of various perception writes, and additionally coming constantly. The accompanying picture demonstrates a gathering of various visualizations that was made in the Power BI benefit.



## b. Datasets

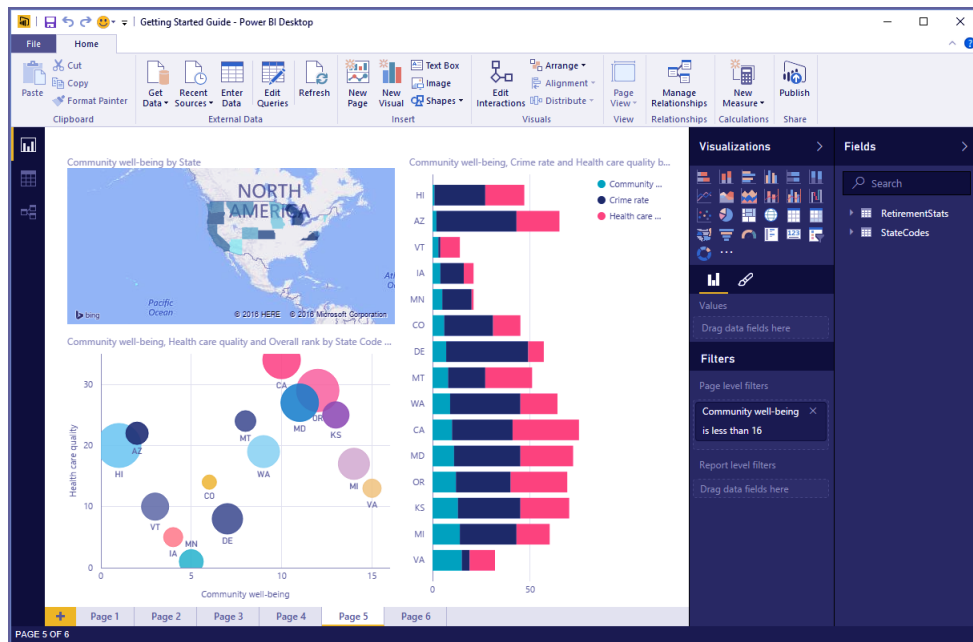
A dataset is an accumulation of information that Power BI uses to make its representations. You can have a basic dataset in light of a solitary table from Excel exercise manual, like what's appeared in the accompanying picture



C2132							2
	B	C	D	E	F	G	H
1	Year	Month	Month Name	Calendar Month	Births	Births Per Day	Births (Normalized)
2119	2004	1	January	1/1/2004	2,937	94.7	2842
2120	2004	2	February	2/1/2004	2,824	97.4	2921
2121	2004	3	March	3/1/2004	3,128	100.9	3027
2122	2004	4	April	4/1/2004	2,896	96.5	2896
2123	2004	5	May	5/1/2004	3,008	97.0	2911
2124	2004	6	June	6/1/2004	3,047	101.6	3047
2125	2004	7	July	7/1/2004	2,981	96.2	2885
2126	2004	8	August	8/1/2004	3,079	99.3	2980
2127	2004	9	September	9/1/2004	3,219	107.3	3219
2128	2004	10	October	10/1/2004	3,547	114.4	3433
2129	2004	11	November	11/1/2004	3,365	112.2	3365
2130	2004	12	December	12/1/2004	3,143	101.4	3042
2131	2005	1	January	1/1/2005	2,921	94.2	2827
2132	2005	2	February	2/1/2005	2,699	96.4	2892
2133	2005	3	March	3/1/2005	3,024	97.5	2926
2134	2005	4	April	4/1/2005	3,037	101.2	3037
2135	2005	5	May	5/1/2005	3,231	104.2	3127
2136	2005	6	June	6/1/2005	3,163	105.4	3163
2137	2005	7	July	7/1/2005	3,119	100.6	3018
2138	2005	8	August	8/1/2005	3,156	101.8	3054
2139	2005	9	September	9/1/2005	3,439	114.6	3439

### c. Reports

In Power BI, a report is a gathering of perceptions that seem together on at least one page. Much the same as some other report you may make for a business introduction, or a report you would compose for a school task, in Power BI a report is an accumulation of things that identify with each other. The accompanying picture demonstrates a report in Power BI Desktop



## d. Dashboards

When you prepare to share a solitary page from a report or offer an accumulation of perceptions, you make a dashboard. Much like the dashboard in an auto. A Power BI dashboard is a gathering of visuals from a solitary page that you can impart to others. Frequently, it's a chosen gathering of visuals that give snappy understanding into the information or story you're attempting to exhibit.

## e. Tiles

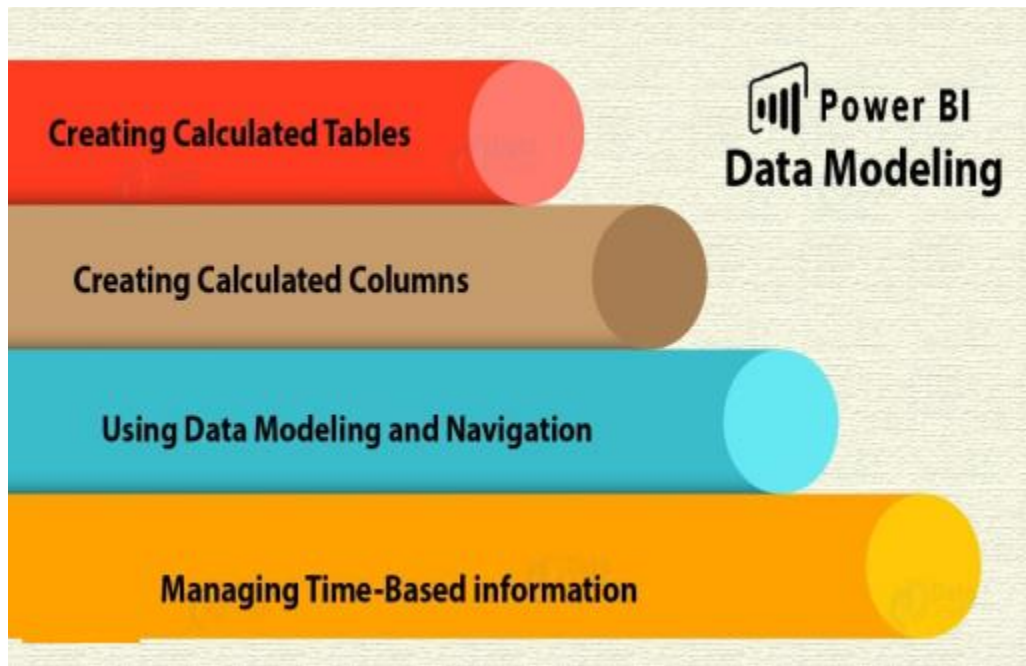
In Power BI, a tile is a solitary representation found in a report or on a dashboard. It's the rectangular box that contains every individual visual. In the accompanying picture, you see one tile (featured by a splendid box) which additionally encompass by different tiles.

## Power BI Components

### Power BI Data Modeling – Creating Calculated Columns

We will study Power BI data modeling. Moreover, we will see how we use Data Modeling in Power BI, and how to Create Calculated Columns in Data Modeling in Power BI. In addition, we will talk about how to

Create a Calculated table in Power BI Data Modeling and Use information Modeling and Navigation.



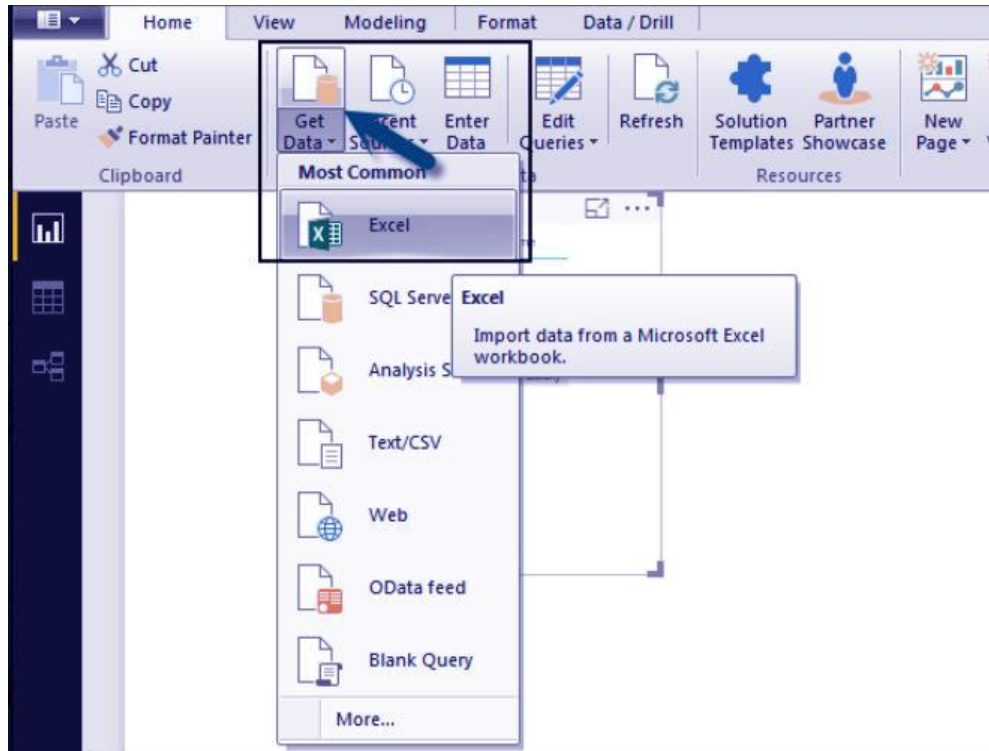
## What is Power BI Data Modeling?

One of **Power BI strengths** is that you just do not get to flatten your information into one table. Instead, you'll use multiple tables from multiple sources, and outline the link between them. You'll conjointly produce your own custom calculations and assign new metrics to look at specific segments of your information and use these new measures in visualizations for straightforward modeling.

## Using information Modeling and Navigation

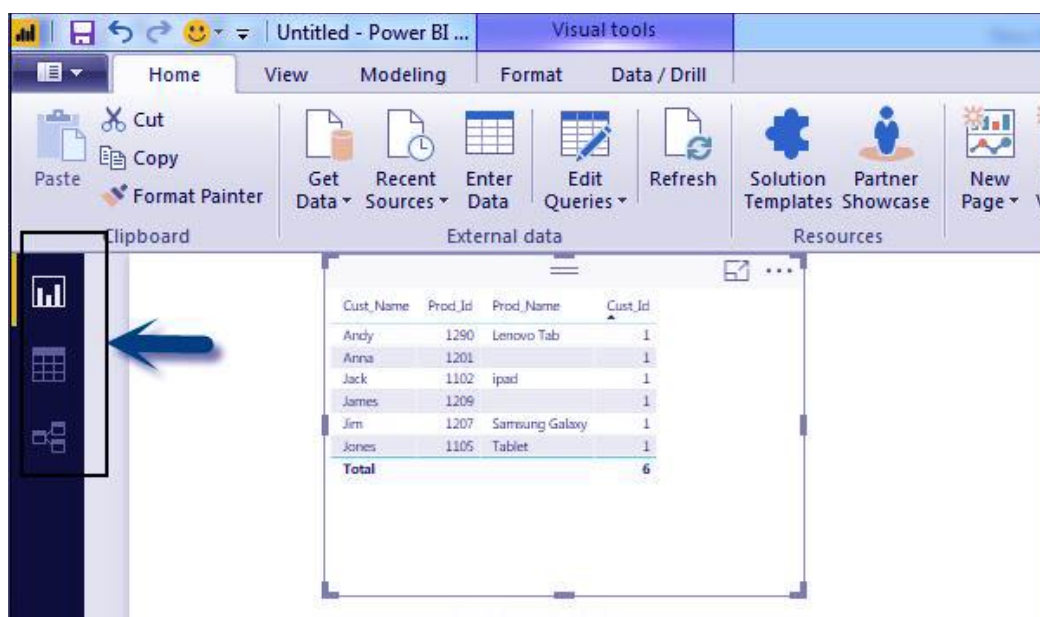
Power Bi Data Modeling is one in all the options that wont to connect multiple information sources. A relationship defines however, information sources are connected with one another, and you'll be able to produce attention-grabbing information visualizations on multiple information sources.

To feature a knowledge supply, attend the Get information possibility. Then, choose the information supply you wish to attach and click on the Connect button.

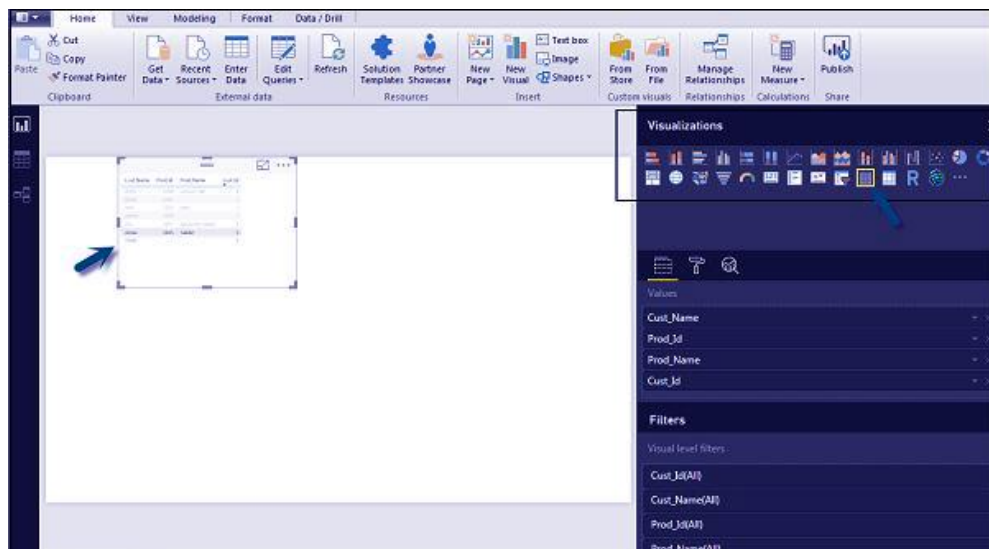


In Power BI on the left aspect of the screen, you've got the subsequent 3 tabs –

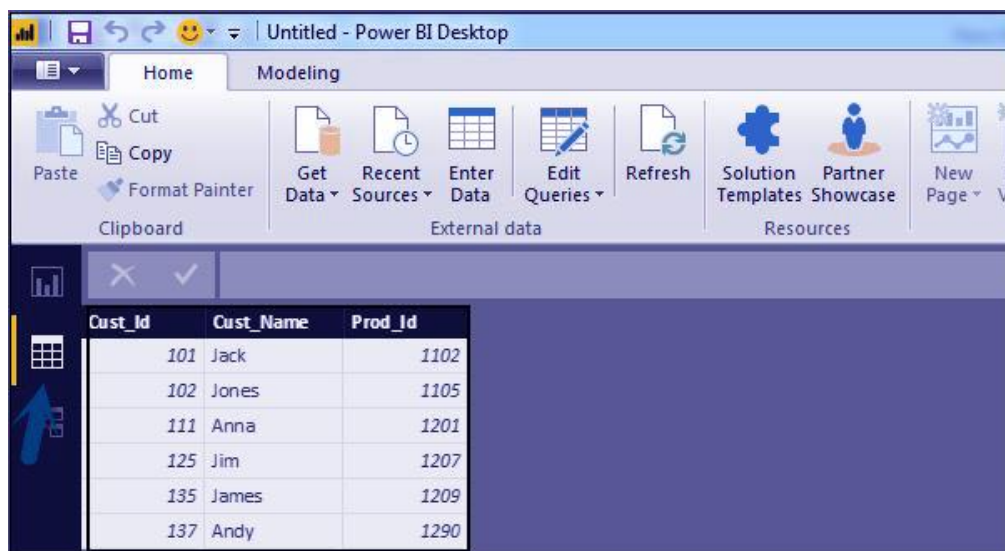
**Report**  
**Data**  
**Relationships**



When you navigate to the Report tab, you'll be able to see a dashboard and a chart hand-picked for information visual image. You'll be able to choose totally different chart varieties as per you want. In our example, we've hand-picked a Table sort from offered Visualizations.

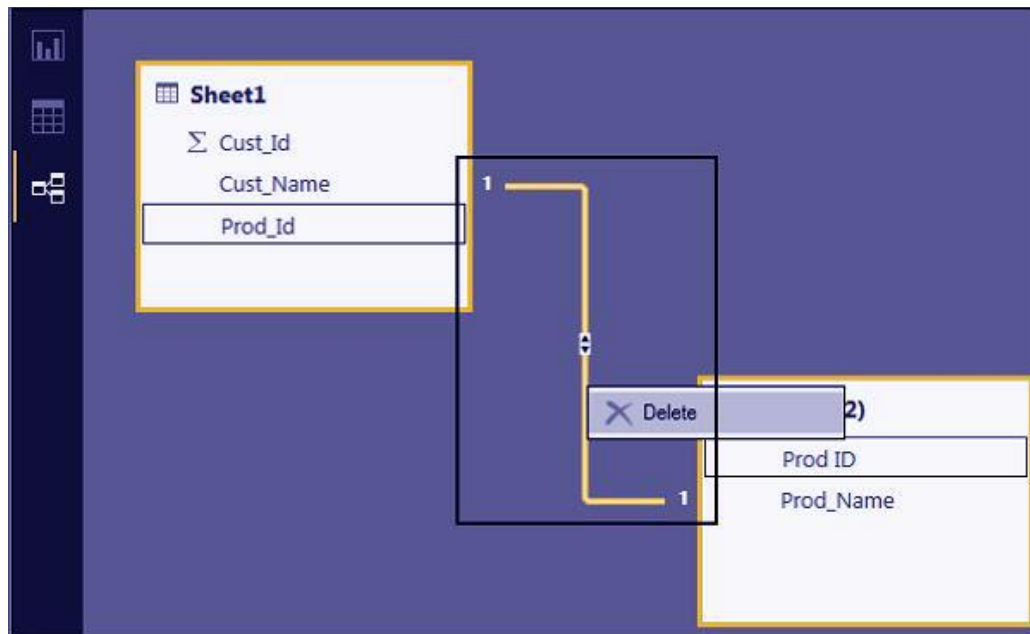


When you attend the information tab, you'll be able to see all the information as per the outlined Relationship from the information sources.



In the Relationship tab, you'll be able to see the connection between information sources. Once you add multiple information sources to Power atomic number 83 visual image, the tool mechanically tries to

discover the connection between the columns. Once you navigate to the connection tab, you'll be able to read the connection. you'll be able to conjointly produce a Relationship between the columns victimization produce Relationships possibility.



## How to Create Calculated Columns in Data Modeling?

You can produce calculated columns in Power BI by combining 2 or a lot of components of the present information. you'll be able to conjointly apply calculation on an associate existing column to outline a replacement metric or mix 2 columns to form one **new column**.

The screenshot shows the Microsoft Excel interface with the 'Table tools' and 'Column tools' tabs active. The 'Column tools' tab is selected, showing options for 'Name', 'Format', 'Summarization', 'Data type', 'Data category', 'Sort by column', 'Data groups', 'Manage relationships', and 'New column'. The 'Name' field is set to 'Updated Price', and the 'Format' is set to 'Whole number'. The 'Summarization' dropdown is set to 'Sum', and the 'Data category' dropdown is set to 'Uncategorized'. The 'Sort by column' dropdown is set to 'Sort', and the 'Data groups' dropdown is set to 'Groups'. The 'Manage relationships' dropdown is set to 'Relationships', and the 'New column' dropdown is set to 'Calculations'. The table below shows various financial metrics, including 'Units Sold', 'Manufacturing Price', 'Sale Price', 'Gross Sales', 'Discounts', 'Sales', 'COGS', 'Profit', 'Date', 'Month Number', 'Month Name', 'Year', and 'Updated Price'. The 'Updated Price' column is calculated as 'financials[Sale Price] \* 25'.

Units Sold	Manufacturing Price	Sale Price	Gross Sales	Discounts	Sales	COGS	Profit	Date	Month Number	Month Name	Year	Updated Price
1513	3	350	529550	0	529550	393380	136170	01 December 2014	12	December	2014	375
1006	10	350	352100	0	352100	261560	90540	01 June 2014	6	June	2014	375
1725	10	350	603750	0	603750	448500	155250	01 November 2013	11	November	2013	375
1513	10	350	529550	0	529550	393380	136170	01 December 2014	12	December	2014	375
1006	120	350	352100	0	352100	261560	90540	01 June 2014	6	June	2014	375
1527	250	350	534450	0	534450	397020	137430	01 September 2013	9	September	2013	375
2750	260	350	962500	0	962500	715000	247500	01 February 2014	2	February	2014	375
1210	3	350	423500	4235	419265	314600	104665	01 March 2014	3	March	2014	375
1397	3	350	488950	4889.5	484060.5	363220	120840.5	01 October 2014	10	October	2014	375
2155	3	350	754250	7542.5	746707.5	560300	186407.5	01 December 2014	12	December	2014	375
2155	10	350	754250	7542.5	746707.5	560300	186407.5	01 December 2014	12	December	2014	375
943.5	250	350	330225	3302.25	326922.75	245310	81612.75	01 April 2014	4	April	2014	375
1397	250	350	488950	4889.5	484060.5	363220	120840.5	01 October 2014	10	October	2014	375
2852	3	350	998200	19964	978236	741520	236716	01 December 2014	12	December	2014	375
2852	10	350	998200	19964	978236	741520	236716	01 December 2014	12	December	2014	375
2966	120	350	1038100	20762	1017338	771160	246178	01 October 2013	10	October	2013	375
2877	120	350	1006950	20139	986811	748020	238791	01 October 2014	10	October	2014	375
2877	250	350	1006950	20139	986811	748020	238791	01 October 2014	10	October	2014	375
266	250	350	93100	1862	91238	69160	22078	01 December 2013	12	December	2013	375
1940	250	350	679000	13580	665420	504400	161020	01 December 2013	12	December	2013	375
2966	260	350	1038100	20762	1017338	771160	246178	01 October 2013	10	October	2013	375
1797	5	350	628950	18868.5	610081.5	467220	142861.5	01 September 2013	9	September	2013	375

## Applied:

Updated Price = `financials[Sale Price] +25`

## Creating Calculated Table in Data Modeling

You can conjointly produce a replacement calculated table in information modeling in Power BI. To form a replacement table, navigate to the information read tab on the left aspect of the screen. So, attend the Modeling possibility at the highest of the screen.

DAX expression employs forming the new table. You've got to enter the name of a replacement table on the left aspect of the equal sign and DAX formula to perform the calculation to make that table on the proper. Once the calculation is complete, the new table seems within the Fields pane in your model.

The screenshot shows the Power BI Desktop interface. The 'Table tools' ribbon is active, displaying options like 'Mark as date table', 'Manage relationships', and 'New measure'. Below the ribbon, the 'Structure' pane shows a table named 'tab3' with the DAX formula: `tab3 = GROUPBY(financials,financials[Country], financials[ Sales],financials[Date], financials[Discount Band])`. The 'Fields' pane on the right shows the 'financials' table with columns: Sales, COGS, Country, Date, Discount Band, Discounts, Gross Sales, Manufacturing Price, Month Name, Month Number, Product, Profit, Sale Price, Segment, Units Sold, Updated Price, Year, and 'tab3'.

Sales	Date	Country	Discount Band
728595	01-05-2014 00:00:00	United States of America	High
201285	01-07-2014 00:00:00	France	High
884205	01-08-2014 00:00:00	United States of America	High
589050	01-12-2013 00:00:00	Germany	High
705600	01-02-2014 00:00:00	Canada	High
222705	01-09-2014 00:00:00	Canada	High
246708	01-05-2014 00:00:00	Germany	High
238609	01-01-2014 00:00:00	Germany	High
655551.75	01-07-2014 00:00:00	Canada	High
107156	01-10-2013 00:00:00	Mexico	High
272888	01-06-2014 00:00:00	Mexico	High
490952	01-11-2014 00:00:00	France	High
368676	01-11-2014 00:00:00	Mexico	High
429660	01-07-2014 00:00:00	Mexico	High
303688	01-10-2014 00:00:00	United States of America	High
191884	01-09-2013 00:00:00	Canada	High
83160	01-02-2014 00:00:00	United States of America	High
281053.5	01-03-2014 00:00:00	Canada	High
545055	01-03-2014 00:00:00	France	High
299171.25	01-01-2014 00:00:00	United States of America	High
801444	01-06-2014 00:00:00	Canada	High
108706.5	01-11-2014 00:00:00	Germany	High
670477.5	01-01-2014 00:00:00	Canada	High

CalCtable = `CALCULATETABLE(financials,financials[Units Sold]>2500)`

## DAX in Power BI

DAX stands for **Data Analysis Expressions** i.e. such expressions or formulas that are used for data analysis and calculations. These expressions are a collection and combination of *functions*, *operators*, and *constants* that are evaluated as one formula to yield results (value or values). DAX formulas are very useful in BI tools like *Power BI* as



they help data analysts to use the data sets, they have to the fullest potential.

With the help of the DAX language, analysts can discover new ways to calculate data values they have and come up with fresh insights.

Have a look at some key points about DAX which will help you understand the concept better.

- DAX is a functional language i.e. its complete code is always a function. An executable DAX expression may contain *conditional statements, nested functions, value references, etc.*
- DAX formulas have two primary data types; **Numeric** and **Non-numeric** or Others. The numeric data type includes *integers, decimals, currency, etc.* Whereas, the non-numeric consists of *strings and binary objects.*
- DAX expressions are evaluated from the innermost function going to the outermost one at the last. This makes formulating of a DAX formula important.

You can use values of mixed data types as inputs in a DAX formula and the conversion will take place automatically during execution of the formula. The output values will be converted into the data type you instructed for the DAX formula.

## DAX Calculation Types

So, apparently, the DAX formulas can also be called as calculations as they calculate an input value and return a resultant value. You can create two types of expressions or calculations using DAX in Power BI; calculated columns and calculated measures.

**Calculated Columns:** The calculated columns create a new column in your existing table. The only difference between a regular column and a calculated column is that it is necessary to have at



least one function in the calculated column. These are used when you want to create a column with filtered or sorted information.

To create a calculated column:

1. Go to the **Modeling** tab in Power BI Desktop.
2. Then select **the New Column** option. A Formula bar will open showing “Column =”. You can replace the “Column” word with the column name you want.
3. After this, enter the expression for the calculated column on the right of the equals to sign.

**Calculated Measures:** A calculated measure creates a field having aggregated values such as sum, *ratios*, *percentages*, *averages*, etc.

To create a calculated measure:

1. Go to the **Modeling** tab in Power BI Desktop.
2. Then select **New Measure** option. A Formula bar will open showing “Measure =”. You can replace the “Measure” word with the measure name you want.
3. After this, enter the expression for the calculated measure on the right of the equals to sign.
4. Once you create the measure, you can modify your measure name with a calculator icon next to it, under the table name you created the measure in.

## **DAX Functions**

A DAX function is a predefined formula which performs calculations on values provided to it in arguments. The arguments in a function need to be in a particular order and can be a *column reference*, *numbers*, *text*, *constants*, *another formula or function*, or a *logical value* such as TRUE or FALSE. Every function performs a particular operation on the values enclosed in an argument. You can use more than one argument in a DAX formula.

## Key Points about DAX Functions

Here are some unique facts about DAX functions that you must know to understand them better:

- Any DAX function always refers to a complete column/field or a table. It will never refer to individual values. If you want to use the functions on separate values within a column, you need to apply filters in a DAX formula.
- DAX functions provide the flexibility to create a formula that is applied on a row-by-row basis. The calculations or formulas get applied as per the context of the values in each row.
- In some cases, DAX functions return a full table which can be used in other DAX formulas that need a complete set of values. However, you cannot display this table's contents.
- DAX functions have a category known as time intelligence functions. Such functions are used to calculate time/date ranges and periods.

## Aggregation functions

### Syntax

`Column=AVERAGE(<column>)`

Returns a column

Average Sale = `AVERAGE(financials[ Sales])`

Average Death Cases = `AVERAGE(day_wise[Deaths / 100 Cases])`

.....

### COUNT

## Syntax

Column=COUNT(<column>)

SaleUpdated = COUNT(financials[Units Sold])

... • • • • •

## COUNTA

Counts the number of rows in the specified column that contain non-blank values.

## Syntax

Column=COUNTA(<column>)

SaleUpdated = COUNTA(financials[Discounts])

... • • • • •

## DISTINCTCOUNT

Counts the number of distinct values in a column.

## Syntax

Column=DISTINCTCOUNT(<column>)

SaleUpdated = DISTINCTCOUNT(financials[Month Name])

DistinctYearCount = DISTINCTCOUNT(financials[Year])

country Count = DISTINCTCOUNT(financials[Country])

... • • • • •

## DISTINCTCOUNTNOBLANK

Counts the number of distinct values in a column.

## Syntax

Column=DISTINCTCOUNTNOBLANK (<column>)

SaleUpdated = DISTINCTCOUNTNOBLANK(financials[Month Name])

... • • • • •

## MAX

Returns the largest value in a column, or between two scalar expressions.

## Syntax

```
Column=MAX(<column>)
Max Sale = MAX(financials[ Sales])
... • ..... •
```

## MAXA

## Syntax

```
Column=MAXA(<column>)
Max Date = MAXA(financials[Date])
... • ..... •
```

## MIN

## Syntax

```
Column=MIN(<column>)
Min Sale = MIN(financials[ Sales])
... • .....
```

## PRODUCT

Returns the product of the numbers in a column.

## Syntax

```
Column=PRODUCT(<column>)
column = PRODUCT(financials[Month Number])
... • .....
```

## SUM

## Syntax

```
Column=SUM(<column>)
Total Sale = SUM(financials[ Sales])
... • .....
```

# SUMX

Returns the sum of an expression evaluated for each row in a table.

`SUMX(<table>, <expression>)`

`column = SUMX(financials,financials[Sale Price]+2)`

# CALENDAR

Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.

## Syntax

`Table=CALENDAR(<start_date>, <end_date>)`

## Return value

Returns a table with a single column named "Date" containing a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.

## Syntax

`CALENDAR(<start_date>, <end_date>)`

`Table = CALENDAR(DATE(2014,1,1),DATE(2014,12,30))`

Home -> New Table

DAX

`Table= CALENDAR ( DATE ( 2005, 1, 1 ), DATE ( 2015, 12, 31 ) )`

By specifying start and ending date from day\_wise table:

`Table= CALENDAR ( DATE ( YEAR ( MIN ( day_wise[Date] ) ), 1, 1 ), DATE ( YEAR ( MAX ( day_wise[Date] ) ), 12, 31 ) )`

`Table= CALENDAR (MINX (day_wise, day_wise[Date]), MAXX (day_wise, day_wise[Date]))`

Table= `CALENDAR` ( 1, 100 )

# DATE

Returns the specified date in datetime format.

## Syntax

`Column=DATE(<year>, <month>, <day>)`

Column 2 = `DATE(YEAR(day_wise[Date]),MONTH(day_wise[Date]),DAY(day_wise[Date]))`

## Simple Date

The following formula returns the date July 8, 2009:

column= `DATE`(2009,7,8)

## Years before 1899

If the value that you enter for the **year** argument is between 0 (zero) and 1899 (inclusive), that value is added to 1900 to calculate the year. The following formula returns January 2, 1908: (1900+08).

Column= `DATE`(08,1,2)

## Years after 1899

If **year** is between 1900 and 9999 (inclusive), that value is used as the year. The following formula returns January 2, 2008:

Column= `DATE`(2008,1,2)

## Months

If **month** is greater than 12, **month** adds that number of months to the first month in the year specified. The following formula returns the date February 2, 2009:

Column= `DATE`(2008,14,2)

## Days

If **day** is greater than the number of days in the month specified, **day** adds that number of days to the first day in the month. The following formula returns the date February 4, 2008:

```
Column=DATE(2008,1,35)
```

## DAY

Returns the day of the month, a number from 1 to 31.

### Syntax

```
DAY(<date>)
```

```
column=DAY("3-4-2007")
```

```
Column= DAY("March 4 2007")
```

```
column= DAY(day_wise[Date])
```

## HOUR

Returns the hour as a number from 0 (12:00 A.M.) to 23 (11:00 P.M.).

### Syntax

```
HOUR(<datetime>)
```

```
Column = HOUR("March 3, 2008 3:00 PM")
```

```
column= HOUR(day_wise[Date])
```

## MINUTE

Returns the minute as a number from 0 to 59, given a date and time value.

### Syntax

```
Column=MINUTE(<datetime>)
```

```
Column=MINUTE("March 23, 2008 1:45 PM")
```

```
Column = MINUTE(day_wise[Date])
```

# YEAR

## Syntax

```
Column=YEAR(<date>)
CurYear = YEAR(NOW())
CurYear = YEAR("March 2007")
CurYear = YEAR(financials[Date])
age = YEAR(TODAY())-1985
```

# MONTH

## Syntax

```
MONTH(<datetime>)
Column=MONTH("March 3, 2008 3:45 PM")
Column = MONTH(financials[Date])

Column=MONTH(Orders[TransactionDate])
```

# WEEKDAY

Returns a number from 1 to 7 identifying the day of the week of a date. By default the day ranges from 1 (Sunday) to 7 (Saturday).

## Syntax

```
WEEKDAY(<date>)
Column = WEEKDAY("29-11-2022")
```

# TODAY

## Syntax

```
TODAY()
Column=TODAY()
```



# NOW

Returns the current date and time in **datetime** format.

The NOW function is useful when you need to display the current date and time on a worksheet or calculate a value based on the current date and time,

## Syntax

```
Column=NOW()  
Column = NOW()+28
```

# TIME

Converts hours, minutes, and seconds given as numbers to a time in **datetime** format.

## Syntax

```
TIME(hour, minute, second)  
Column = TIME(12,59,58)  
Column= TIME(10,34,56)  
Column=TIME(HOUR(day_wise[Date]),MINUTE(day_wise[Date]),SECOND(day_wise[Date]))  
... • ..... •
```

# FILTER

Returns a table that represents a subset of another table or expression.

## Syntax

```
Table=FILTER(<table>,<filter>)  
  
Table = FILTER(financials,financials[Country]="France")  
Table = FILTER(financials,financials[Country]="France" && financials[Manufacturing  
Price] >200)  
  
Table3 = FILTER(financials,financials[Country]="Germany" &&  
financials[Manufacturing Price] >200)  
  
Table4 =FILTER(financials,financials[Country]="Germany" ||  
financials[Country]="France")  
  
... • ..... •
```

# FIND

Returns the starting position of one text string within another text string. FIND is case-sensitive.

## Syntax

```
Column=FIND(<find_text>, <within_text>[, [<start_num>][, <NotFoundValue>]])
```

```
Column = FIND("an",financials[Country],,-2)
```

# Statistical functions

Returns the median of numbers in a column.

## Syntax

```
MEDIAN(<column>)
```

```
Column = MEDIAN(financials[Units Sold])
```

# STDEV.P

Returns the standard deviation of the entire population.

```
STDEV.P(<ColumnName>)
```

```
Column = STDEV.P(day_wise[Deaths / 100 Cases])
```

# STDEV.S

Returns the standard deviation of a sample population.

## Syntax

```
STDEV.S(<ColumnName>)
```

```
Column = STDEV.S(day_wise[Deaths / 100 Cases])
```

# VAR.P

Returns the variance of the entire population.

## Syntax

`VAR.P(<columnName>)`

Column = `VAR.P(day_wise[Deaths / 100 Cases])`

# VAR.S

Returns the variance of a sample population.

## Syntax

`VAR.S(<columnName>)`

Column = `VAR.S(day_wise[Deaths / 100 Cases])`

... • ..... •

# DATEDIFF

Returns the number of interval boundaries between two dates.

## Syntax

`DATEDIFF(<Date1>, <Date2>, <Interval>)`

Column = `DATEDIFF(day_wise[Date], TODAY(), YEAR)`

Column1 = `DATEDIFF(day_wise[Date], TODAY(), MONTH)`

# SECOND

Returns the seconds of a time value, as a number from 0 to 59.

## Syntax

`SECOND(<time>)`

```
Column = SECOND("March 23, 2008 1:45 PM")
```

```
Column = SECOND(day_wise[Date])
```

## COUNTBLANK

Counts the number of blank cells in a column.

### Syntax

```
Column=COUNTBLANK(<column>)
```

```
Column = COUNTBLANK(day_wise[Deaths / 100 Cases])
```

## COUNTROWS

The COUNTROWS function counts the number of rows in the specified table, or in a table defined by an expression.

### Syntax

```
Column=COUNTROWS([<table>])
```

```
Column 3 = COUNTROWS(day_wise)
```

or

```
Column 3 = COUNTROWS()
```

... • ..... •

## String Functions

## LEN

Returns the number of characters in a text string.

```
LEN(<text>)
```

```
Column = LEN(financials[Country])
```

## LOWER

Converts all letters in a text string to lowercase.

### Syntax

```
LOWER(<text>)
```

```
Column = LOWER(financials[Country])
```

```
Column = LOWER('P1-UK-Bank-Customers' [Surname])
```

## UPPER

Converts a text string to all uppercase letters.

## Syntax

```
UPPER(<text>)  
Column = UPPER(financials[Country])  
Column = UPPER('P1-UK-Bank-Customers'[Surname])
```

## REPT

Repeats text a given number of times. Use REPT to fill a cell with a number of instances of a text string.

## Syntax

```
REPT(<text>, <num_times>)  
Column = REPT(financials[Country],2)  
Column = REPT('P1-UK-Bank-Customers'[Surname],3)
```

## SEARCH

Returns the number of the character at which a specific character or text string is first found, reading left to right.

## Syntax

```
SEARCH(<find_text>, <within_text>[, [<start_num>][, <NotFoundValue>]])  
Column = SEARCH("is", "India is my country", 1, -2)  
Column = SEARCH("EN", 'P1-UK-Bank-Customers'[Region],1,-5)
```

## CONCATENATE

Joins two text strings into one text string.

## Syntax

```
CONCATENATE(<text1>, <text2>)  
  
Column = CONCATENATE('P1-UK-Bank-Customers'[Name], 'P1-UK-Bank-Customers'[Surname])
```

```
Column = 'P1-UK-Bank-Customers'[Name] & " " & 'P1-UK-Bank-Customers'[Surname]  
CountryProduct Column = CONCATENATE(financials[Country],CONCATENATE("  
",financials[Product]))  
  
Full Name = CONCATENATE(CONCATENATE('P1-UK-Bank-Customers'[Name], " "), 'P1-UK-Bank-  
Customers'[Surname])
```

## LEFT

Returns the specified number of characters from the start of a text string.

Syntax

```
LEFT(<text>, <num_chars>)
```

```
Column = LEFT('P1-UK-Bank-Customers'[Surname],3)
```

## RIGHT

RIGHT returns the last character or characters in a text string, based on the number of characters you specify.

### Syntax

```
RIGHT(<text>, <num_chars>)
```

```
Column = RIGHT('P1-UK-Bank-Customers'[Surname],3)
```

## REPLACE

REPLACE replaces part of a text string, based on the number of characters you specify, with a different text string.

### Syntax

```
REPLACE(<old_text>, <start_num>, <num_chars>, <new_text>)
```

```
Column = REPLACE("New Products",1,2,"OB")
```

```
Column = REPLACE('P1-UK-Bank-Customers'[Surname],1,2,"EKM")
```

## TRIM

Removes all spaces from text except for single spaces between words.

## Syntax

```
TRIM(<text>)  
Column = TRIM(" A column with trailing spaces. ")  
Column = TRIM('P1-UK-Bank-Customers'[Name])
```

# Math functions

## ABS

Returns the absolute value of a number.

## Syntax

```
ABS(<number>)  
Column = ABS(-19)  
Column = abs(day_wise[Recovered]-day_wise[Active])
```

... • ..... •

## CEILING

Rounds a number up, to the nearest integer or to the nearest multiple of significance.

## Syntax

```
CEILING(<number>, <significance>)  
Column = CEILING(4.41,0.22) returns 4.62  
  
Column = CEILING(4.42,0.05)
```

... • ..... •

## CONVERT

Converts an expression of one data type to another.

## Syntax

```
CONVERT(<Expression>, <Datatype>)
```

```
Column = CONVERT(20.8,INTEGER) returns 21
```

```
Column = CONVERT(20.4,INTEGER) returns 20
```

## DIVIDE

Performs division and returns alternate result or BLANK() on division by 0.

## Syntax

```
DIVIDE(<numerator>, <denominator> [,<alternateresult>])
```

```
Column = DIVIDE(6,2)
```

```
Column 2 = DIVIDE('P1-UK-Bank-Customers'[Balance], 'P1-UK-Bank-Customers'[Age])
```

## EVEN

Returns number rounded up to the nearest even integer.

## Syntax

```
EVEN(number)
```

```
Column=EVEN(1.5)
```

```
Column 2 = EVEN(2.1) returns 4
```

```
Column 2 = EVEN('P1-UK-Bank-Customers'[Age])
```

## FACT

Returns the factorial of a number, equal to the series  $1*2*3*...*$ , ending in the given number.

## Syntax

```
FACT(<number>)
```

```
Column 2 = FACT('P1-UK-Bank-Customers'[Age])
```

```
Column 2 = FACT(5)
```



# FLOOR

Rounds a number down, toward zero, to the nearest multiple of significance.

## Syntax

```
FLOOR(<number>, <significance>)
```

```
Column = FLOOR(10.5323,.6)
```

```
Column 2 = FLOOR('P1-UK-Bank-Customers'[Balance],.6)
```

# GCD

Returns the greatest common divisor of two or more integers.

## Syntax

```
GCD(number1, [number2], ...)
```

```
Column 2 = GCD(5,10)
```

# INT

Rounds a number down to the nearest integer.

```
INT(<number>)
```

```
column= INT(1.5)
```

```
Column 2 = INT('P1-UK-Bank-Customers'[Balance])
```

# LCM

Returns the least common multiple of integers.

## Syntax

```
LCM(number1, [number2], ...)
```

```
Column 2 = INT('P1-UK-Bank-Customers'[Balance])
```

```
Column 2 = LCM(5,10)
```

# MOD

Returns the remainder after a number is divided by a divisor.

## Syntax

```
MOD(<number>, <divisor>)
```

```
Column = mod(10,4)
```

## ODD

Returns number rounded up to the nearest odd integer.

## Syntax

```
ODD(number)
```

```
Column = ODD(45.6)
```

```
Column 2 = ODD('P1-UK-Bank-Customers'[Balance])
```

## POWER

Returns the result of a number raised to a power.

## Syntax

```
POWER(<number>, <power>)
```

```
Column= POWER(5,2)
```

```
Column 2 = POWER(8,2)
```

```
Column = POWER(country_wise_latest[New recovered],2)
```

## RAND

Returns a random number greater than or equal to 0 and less than 1

## Syntax

```
RAND()
```

```
Column = RAND()
```

## RANDBETWEEN

Returns a random number in the range between two numbers you specify.

## Syntax

```
RANDBETWEEN(<bottom>,<top>)  
Column = RANDBETWEEN(1,10)
```

# ROUND

Rounds a number to the specified number of digits.

## Syntax

```
ROUND(<number>, <num_digits>)  
Column = ROUND(2.15,1)
```

# SQRT

Returns the square root of a number.

## Syntax

```
SQRT(<number>)  
  
Column = SQRT(64)
```

# Logical functions

## IF

Checks a condition, and returns one value when it's TRUE, otherwise it returns a second value.

## Syntax

```
IF(<logical_test>, <value_if_true>[, <value_if_false>])  
  
Column = IF(25< 500, "Low")  
Column = IF(country_wise_latest[Confirmed last week]< 500, "Low count")  
Column = IF(country_wise_latest[Confirmed last week]< 500, "Low count", "High Count")
```

```
Column = IF( country_wise_latest[Confirmed last week]< 500, "Low",  
column=IF( country_wise_latest[Confirmed last week] < 1500, "Medium", "High" )
```

## AND

Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE. Otherwise returns false.

### Syntax

```
AND(<logical1>,<logical2>)
```

```
Column = IF(AND(10 > 9, -10 < -1), "All true", "One or more false" )
```

Because both conditions, passed as arguments, to the AND function are true, the formula returns "All True".

## NOT

Changes FALSE to TRUE, or TRUE to FALSE.

### Syntax

```
NOT(<logical>)
```

```
Column = NOT(10>20)
```

## OR

Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.

### Syntax

```
OR(<logical1>,<logical2>)
```

```
Column = OR(10>5, 9<5)
```

## SWITCH

Evaluates an expression against a list of values and returns one of multiple possible result expressions.

## Syntax

```
SWITCH(<expression>, <value>, <result>[, <value>, <result>]...[, <else>])  
Column 2 = SWITCH(7, 1, "January", 2, "February", 3, "March", 4, "April"  
    , 5, "May", 6, "June", 7, "July", 8, "August"  
    , 9, "September", 10, "October", 11, "November", 12, "December"  
    , "Unknown month number" )
```

# Table manipulation functions

## ADDCOLUMNS

Adds calculated columns to the given table or table expression.

## Syntax

```
ADDCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]...)  
  
Table = ADDCOLUMNS(financials, "New Price", financials[Manufacturing Price]+100)
```

## DISTINCT (column)

Returns a one-column table that contains the distinct values from the specified column. In other words, duplicate values are removed and only unique values are returned.

## Syntax

```
DISTINCT(<column>)  
Table4 = DISTINCT(financials[Country])
```

## DISTINCT (table)

Returns a table by removing duplicate rows from another table or expression.

# Syntax

```
DISTINCT(<table>)  
Table = DISTINCT(financials)
```

# CROSSJOIN

Returns a table that contains the Cartesian product of all rows from all tables in the arguments. The columns in the new table are all the columns in all the argument tables.

# Syntax

```
CROSSJOIN(<table>, <table>[, <table>]...)
```

**Colors**

**Color**

Red  
Green  
Blue

**Pattern**

Horizontal Stripe  
Vertical Stripe  
Crosshatch

**Stationery**

**Font**

serif  
sans-serif

**Presentation**

embossed  
engraved

```
Table = CROSSJOIN( Colors, Stationery)
```

Color	Pattern	Font	Presentation
Red	Horizontal Stripe	serif	embossed
Green	Vertical Stripe	serif	embossed
Blue	Crosshatch	serif	embossed
Red	Horizontal Stripe	sans-serif	engraved
Green	Vertical Stripe	sans-serif	engraved
Blue	Crosshatch	sans-serif	engraved

**Power Query** is a Data Transformation and Data Preparation tool included with Microsoft Excel and Microsoft Power BI. Power Query streamlines the process of importing data from a variety of file types including Excel tables, CSV files, Database tables, Webpages, and so on, and allows you to transform your data into the proper shape and condition for better analysis.

Once you've configured your Power Query procedures, you won't have to repeat the same steps with new data. You can quickly set up and automate the same data transformation procedures and produce the same data outputs with Power Query.

Microsoft Power BI's Power Query feature allows you to do sophisticated data transformations such as:

- Removing any extraneous columns, rows, or blanks
- Conversions of data types – text, numbers, and dates
- Columns can be split or merged.
- New computed columns are being added.
- **Data aggregation** or summarization, among other things.

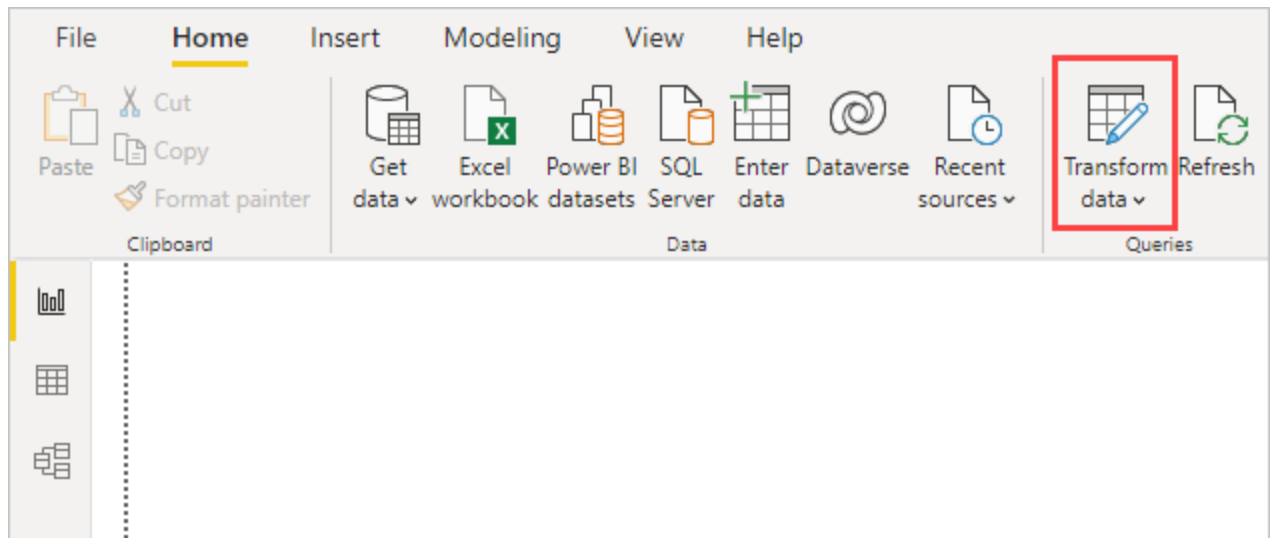
### Key Features of Power Query

The following are some of Power Query's most notable features.

- **Connectivity to a variety of Data Sources:** Power Query has been designed to export data from a variety of sources, including but not limited to text files, Excel Workbooks, and CSV files.
- **Combining Tables:** The Merge option in Power Query replaces Excel's VLOOKUP function. The latter is a handy method for looking up corresponding values, but it gets slightly problematic when applied to a huge dataset with thousands of rows.
- **Tables for Combining:** When modifications to the same source data must be imported at regular periods
- **Make Your Own Functions:** Power Query was created in such a way that you do not need to know how to code to utilize it. It is really simple to use because you simply click buttons and apply filters as you would in Excel.
- **Power Query keeps track of your actions and automates processes:** Power Query not only makes all of these activities easier but also logs your steps so you don't have to repeat them. If you prepare the same data every day, week, or month, it will save you a lot of time.

### 1) Power Query Power BI: Power Query Editor

To access Power Query Editor, go to the **Home tab** of Power BI Desktop and click Transform data.



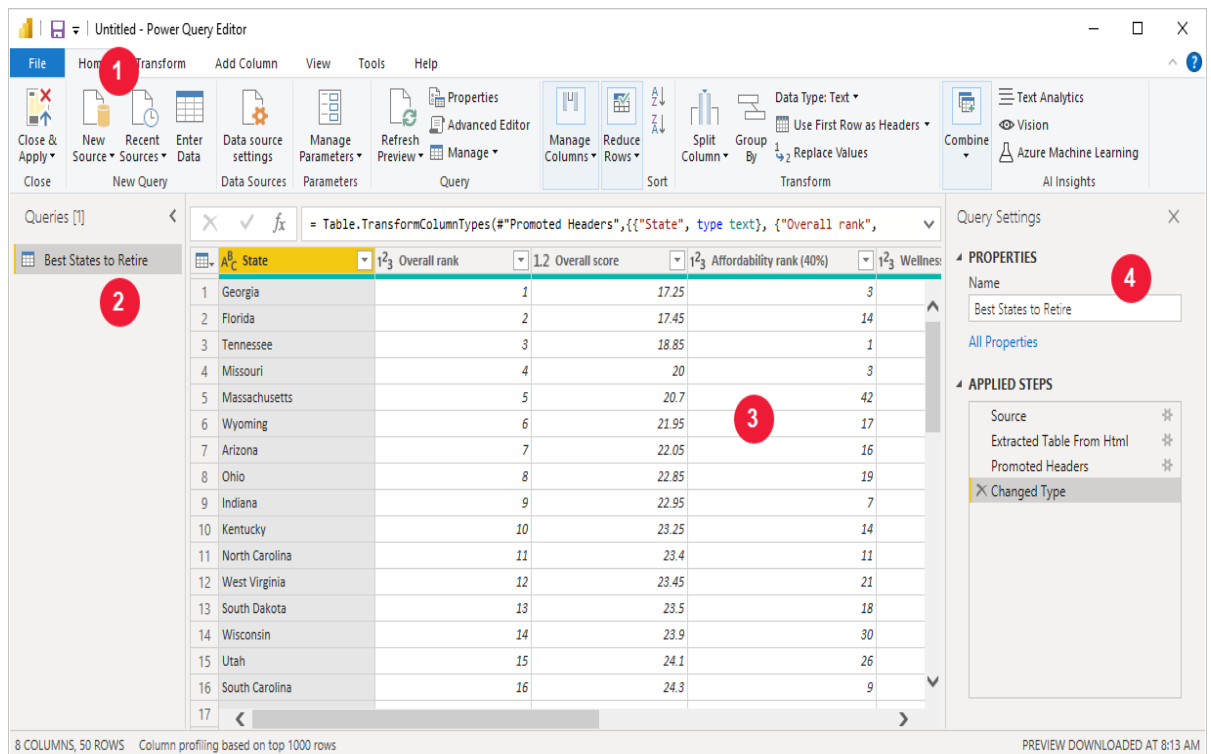
When there are no data connections, Power Query Editor displays as a blank window, ready to be filled with data.

The Power Query Editor window becomes more intriguing as a query is loaded. When we connect to the following Web data source, Power Query Editor loads data information, which you can then shape:

Here's how Power Query Editor looks once you've made a data connection:

- Many buttons in the ribbon are now active to interact with the data in the query.
- Queries are listed in the left pane and can be selected, viewed, and shaped.
- The data from the selected query is shown and ready for shaping in the center pane.
- The Query Settings window displays, listing the query's properties and steps that have been applied.

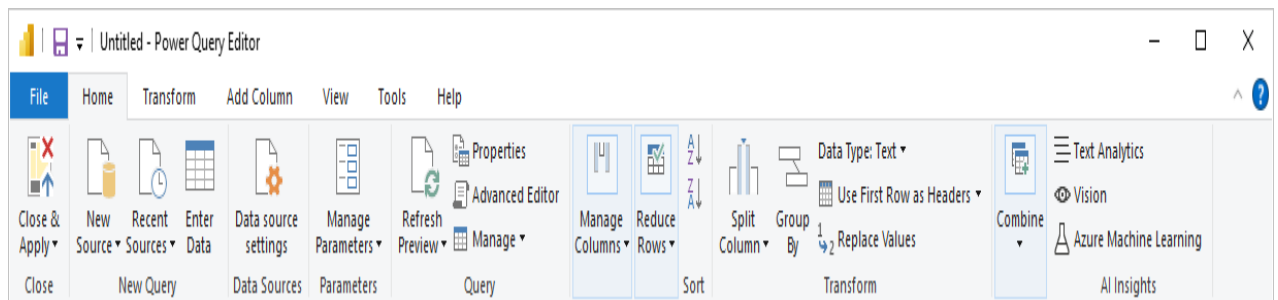




## 2) Power Query Power BI: The Query Tab

Power Query Editor's ribbon is divided into four Tabs: Home, Transform, Add Column, View, Tools, and Help.

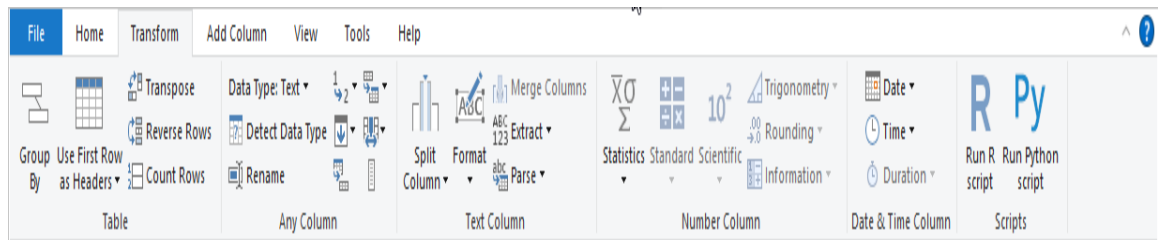
The common query tasks are located on the **Home** tab.



Select **New Source** to connect to data and begin the query-building process. A menu with the most frequent data sources is shown.

The **Transform** tab gives you access to popular data transformation activities like:

1. Columns can be added or removed.
2. Modifying data types
3. Column division
4. Other data-driven tasks include



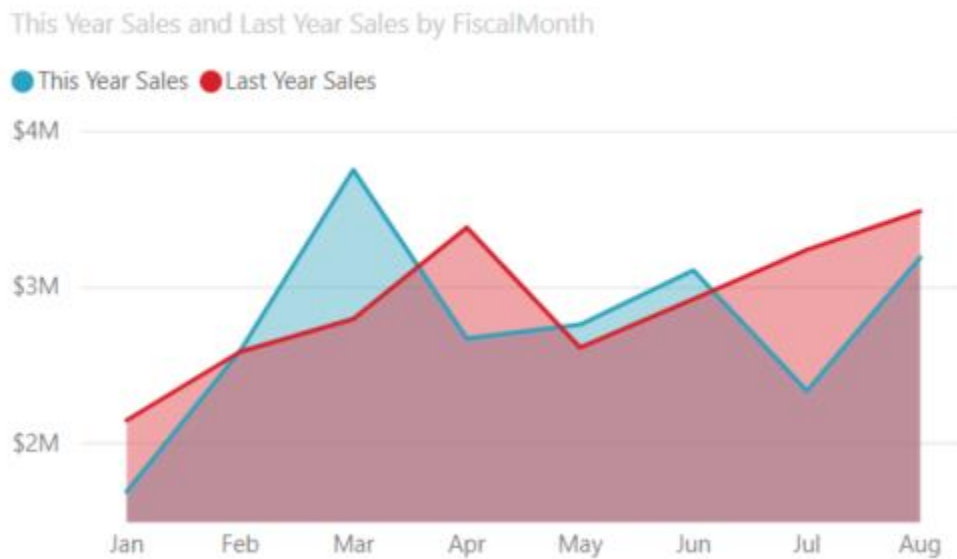
PowerQuery aggregate function

`=List.Sum({10, 20, 30})`

`=List.Sum({[Deaths], [Recovered]})`

## Visualization in Power BI

### Area charts: Basic (Layered) and Stacked




## Create and use basic area charts

Create a basic area chart


These steps will help you create an area chart that displays this year's sales and last year's sales by month.

1.

In Power BI Desktop, open the **Retail Analysis Sample PBIX file** in report view . In the Power BI service, open the **Retail Analysis Sample PBIX file** and select **Edit**.

2.

3.

Select  to add a new page.

4.

5.

From the Fields pane, select **Sales > Last Year Sales**, and **This Year Sales > Value**.

6.