POWER BI

Power BI

Power BI is a standalone Microsoft business intelligence product, which includes both desktop and web-based applications for loading, modeling, and visualizing data

BUSINESS INTELLIGENCE PLATFORMS

Leading platforms include Power BI, Tableau and Qlik among others as shown below-

POWER BI Includes below main topics: -

1 **Introducing Power BI Desktop**	*Installing Power BI, exploring the Power BI workflow, comparing Power BI vs. Excel, etc.*
2 **Connecting & Shaping Data**	*Connecting to source data, shaping and transforming tables, editing, merging and appending queries, etc.*
3 **Creating a Data Model**	*Building relational models, creating table relationships, understanding cardinality, exploring filter flow, etc.*
4 **Adding Calculated Fields with DAX**	*Understanding DAX syntax, adding calculated columns and measures, writing common formulas and functions, etc.*
5 **Visualizing Data with Reports**	*Inserting charts and visuals, customizing formats, editing interactions, applying filters and bookmarks, etc.*

Use Power BI Desktop to:

- *Connect and transform the raw data*
- *Build a relational data model*
- *Create new calculated columns and DAX measures*
- *Design an interactive report to analyze and visualize the data*

NOTE: *Power BI is currently only compatible with PC/Windows (not available for Mac)*

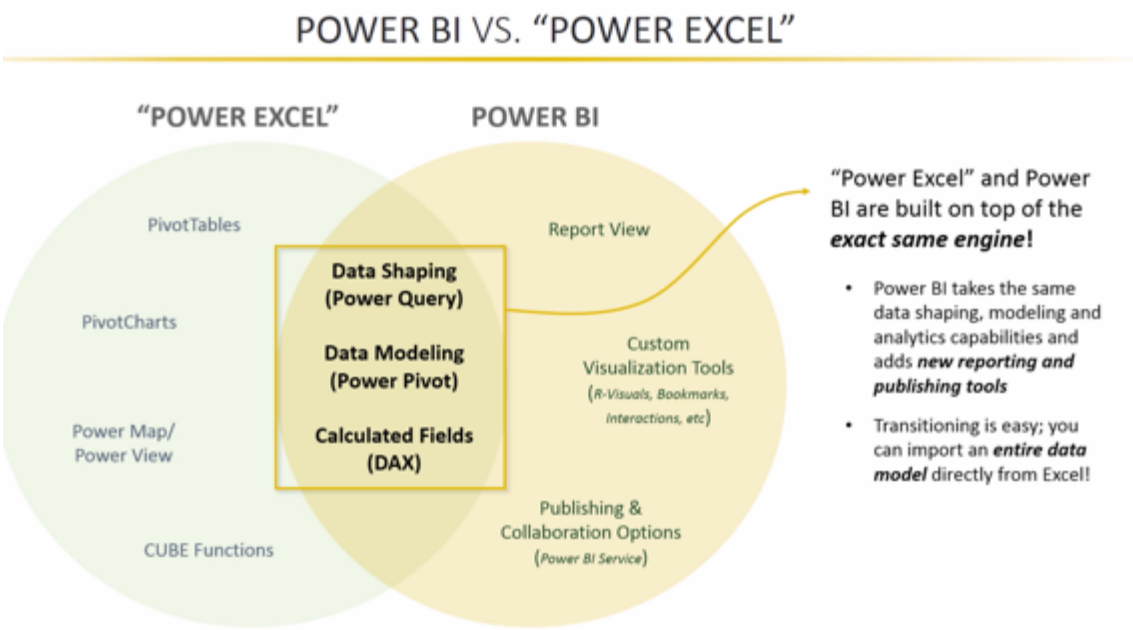
Microsoft Power BI Desktop version is free but Power BI Service is not free (online collaboration)

WHY POWER BI?

- **Connect, transform and analyze *millions* of rows of data**
  - *Access data from virtually anywhere (database tables, flat files, cloud services, folders, etc), and create fully automated data shaping and loading (ETL) procedures*
- **Build relational models to blend data from multiple sources**
  - *Create table relationships to analyze holistic performance across an entire data model*
- **Define complex calculations using Data Analysis Expressions (DAX)**
  - *Enhance datasets and enable advanced analytics with powerful and portable DAX expressions*
- **Visualize data with interactive reports & dashboards**
  - *Build custom business intelligence tools with best-in-class visualization and dashboard features*
- **Power BI is the industry leader among BI platforms**
  - *Microsoft Power BI is intuitive, powerful and absolutely FREE to get started*

Data Loading Tools will be called Get and transform in Excel vs Power Query in Power BI.

Data Modelling tools will be called Power Pivot in Excel vs Relationships view in Power BI.



How to Install Power BI Desktop?
Need not require to sign in for power BI Desktop which is free for use.
Only required sign in for paid version for Power BI Service.

3 CORE VIEWS IN POWER BI

1. Data View

Start with the Data View and the Query editor. Here you can connect Transform and shape raw data.

2. Relationships / Model View

Design data model and tie the tables together with relationships.

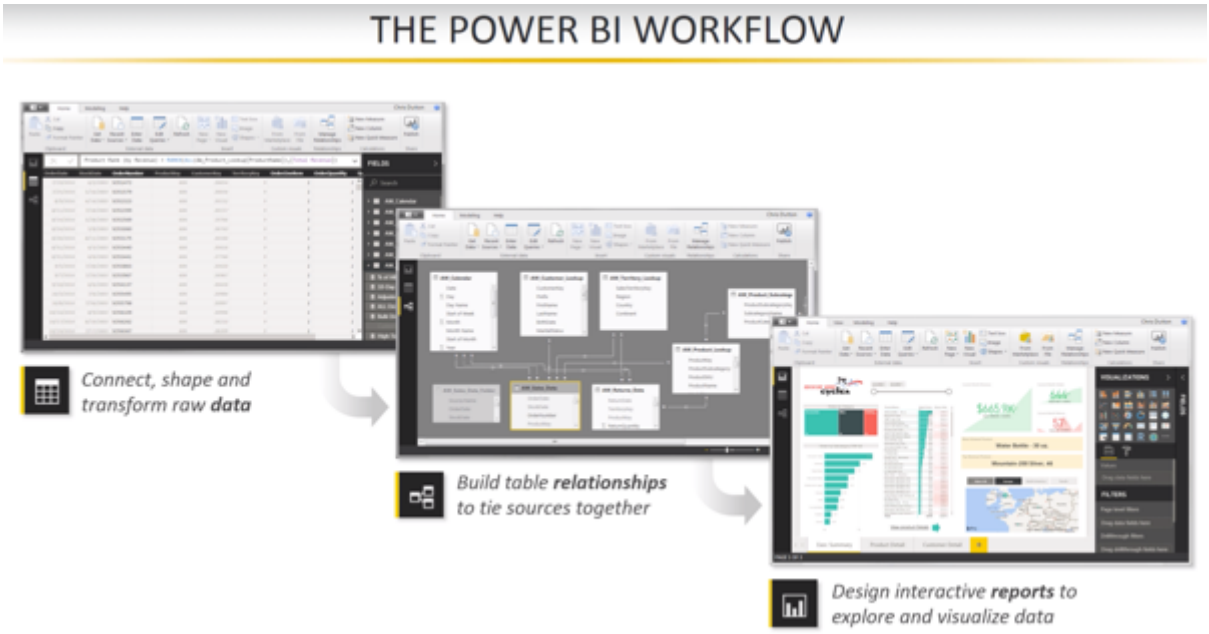
3. Report View

Designing reports and visualize data

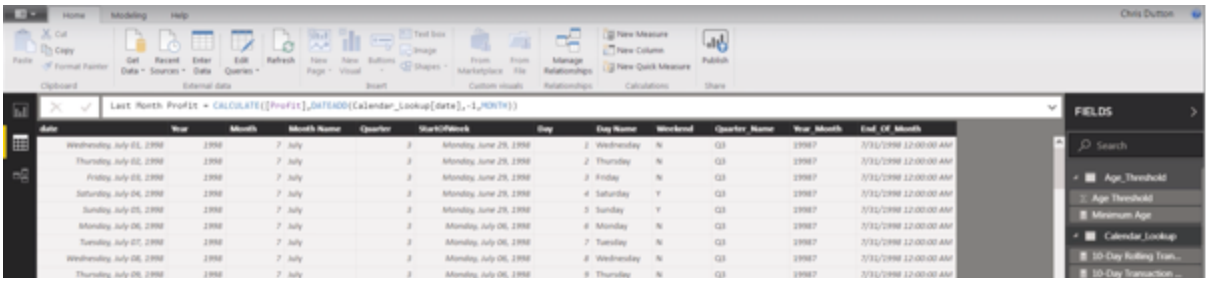
Symbols for 3 Views

Power BI WORKFLOW

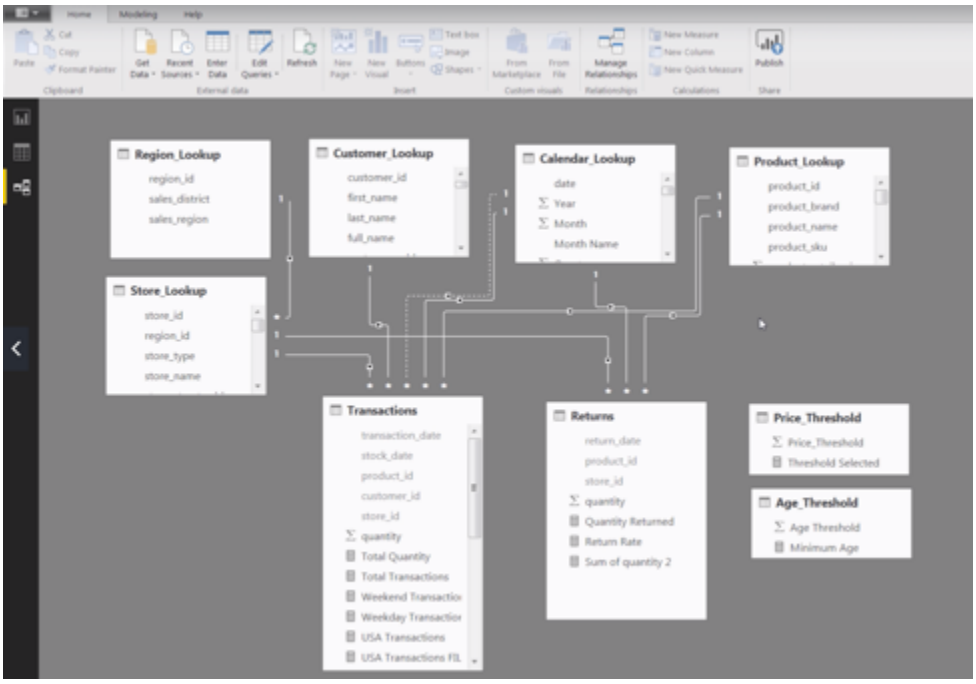
Data View to Relationships View and finally Report view



DATA VIEW

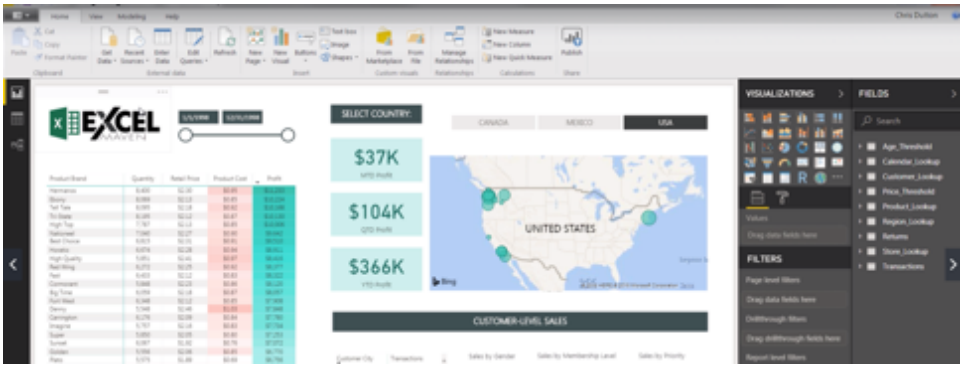


RELATIONSHIPS VIEW

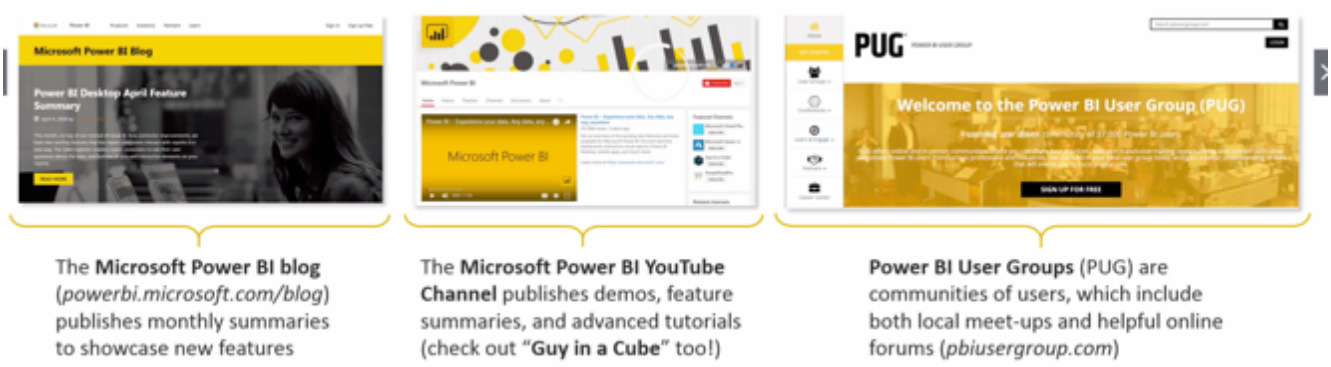


Relationships View is all about the data model. This is where you can see your tables as objects along with the relationships, cardinality, the filter flow.

REPORT VIEW



HELPFUL RESOURCES



The **Microsoft Power BI blog** (*powerbi.microsoft.com/blog*) publishes monthly summaries to showcase new features

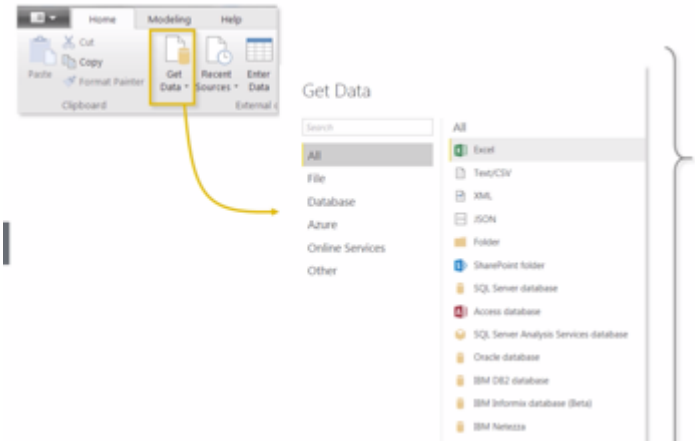
The **Microsoft Power BI YouTube Channel** publishes demos, feature summaries, and advanced tutorials (check out "**Guy in a Cube**" too!)

Power BI User Groups (PUG) are communities of users, which include both local meet-ups and helpful online forums (*pbiusergroup.com*)

Phases of Power BI are as follows –
Phase 1 – Connecting and Shaping data with Power BI Desktop
Phase 2 – Creating Table Relationships and Data Models in Power BI
Phase 3 – Analyzing data with DAX calculations in Power BI
Phase 4 – Visualizing data with Power BI Reports

Phase 1 of Power BI CONNECTING AND SHAPING DATA WITH POWER BI DESKTOP

You can access data from virtually anywhere. Below picture shows all sources from where you can get the data. You need to access via Get Data option in the Home tab and you can get data from files/databases/online services etc

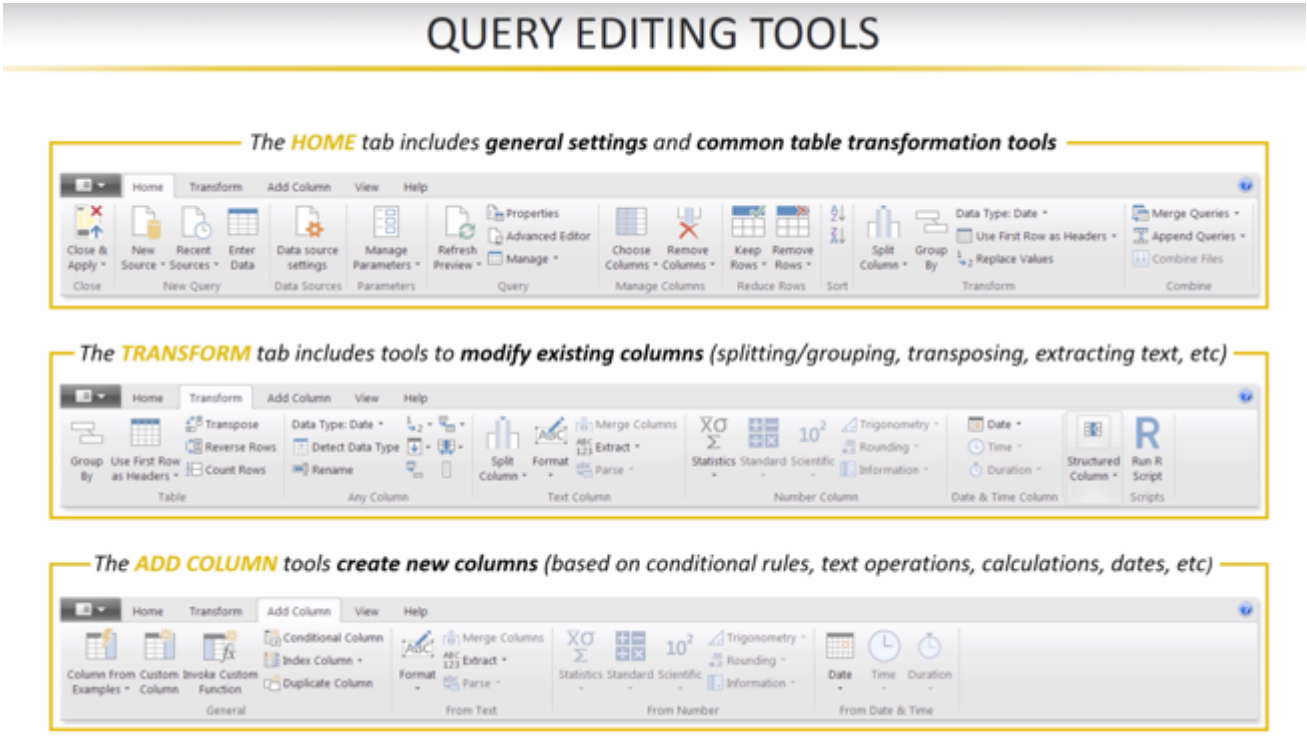


Power BI can connect to virtually **any** type of source data, including (*but not limited to*):

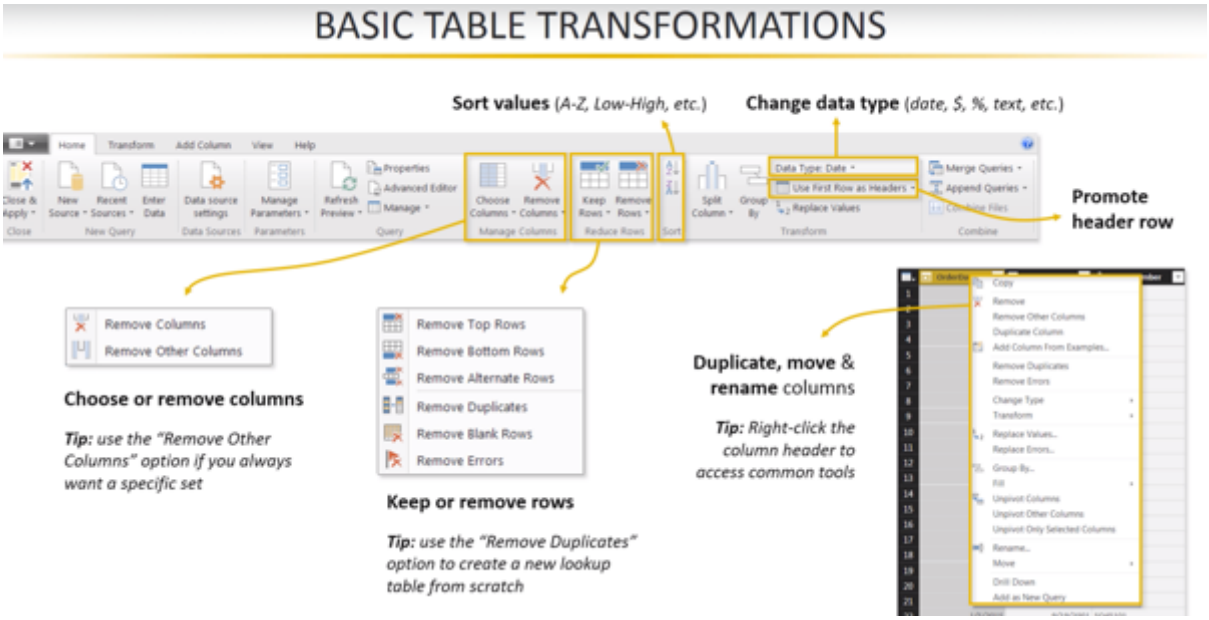
- **Flat files & Folders** (*csv, text, xls, etc*)
- **Databases** (*SQL, Access, Oracle, IBM, Azure, etc*)
- **Online Services** (*Sharepoint, GitHub, Dynamics 365, Google Analytics, Salesforce, Power BI Service, etc*)
- **Others** (*Web feeds, R scripts, Spark, Hadoop, etc*)

For connecting and transforming data the QUERY EDITOR is our cockpit. From home tab use Edit Queries to launch the Query Editor.

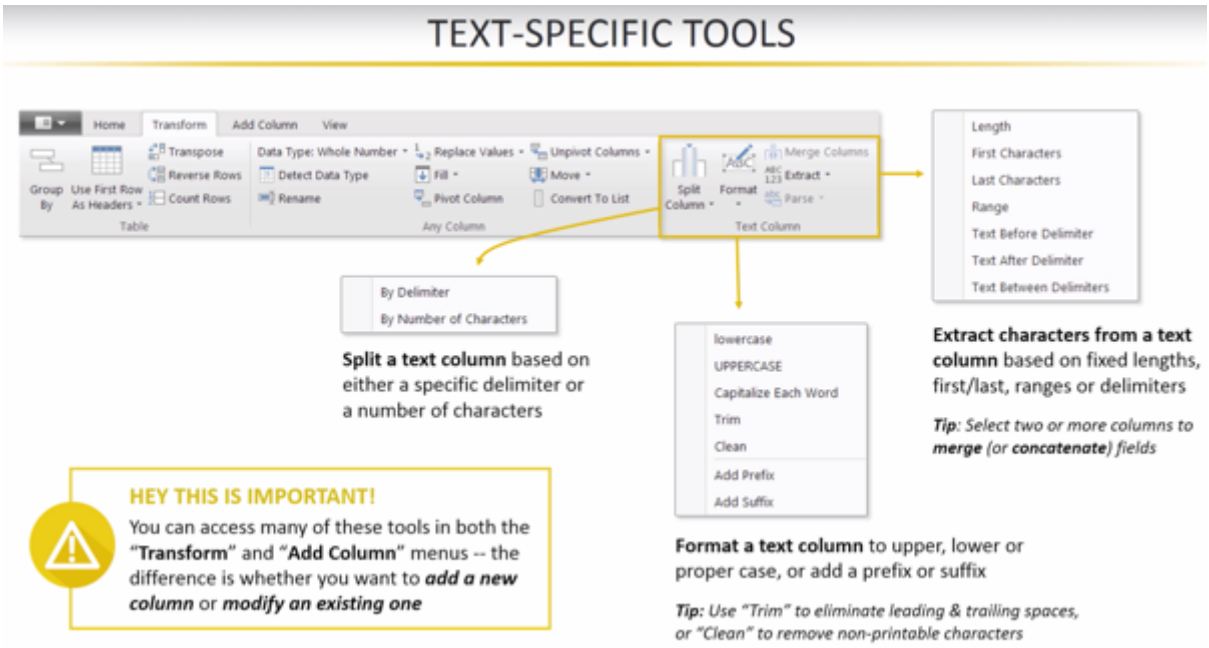
Home, Transform and Add Column



Some Basic Table Transformations including Add, Delete rows, add header, change data type etc

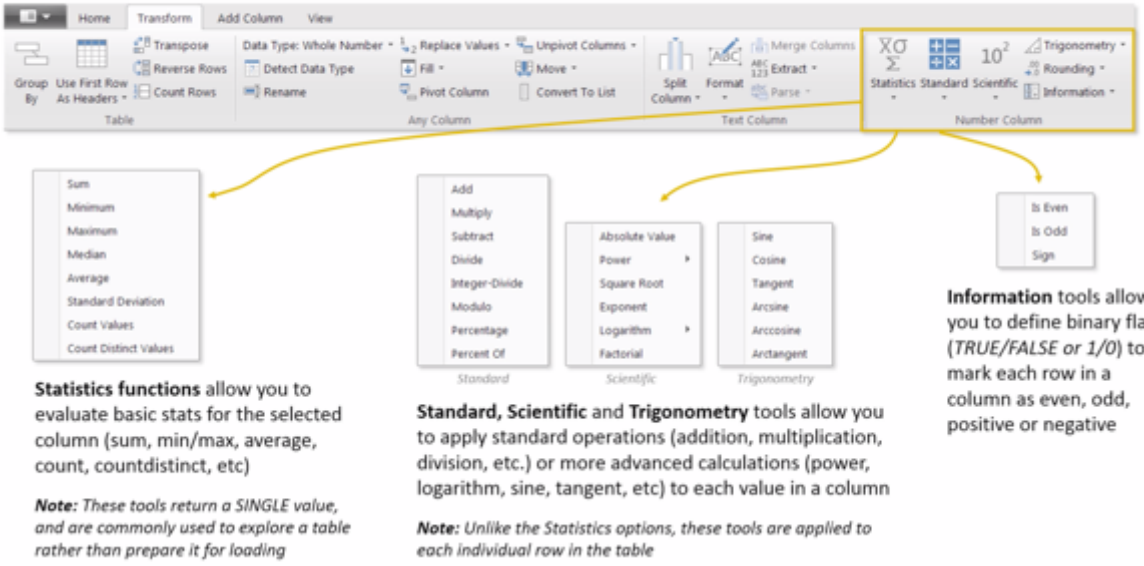


Some Text Specific Tools in Transform Tab



Number Specific Tools
Sum , Median, Mode, Square Root, Power, Round etc

NUMBER-SPECIFIC TOOLS



Statistics functions allow you to evaluate basic stats for the selected column (sum, min/max, average, count, countdistinct, etc)

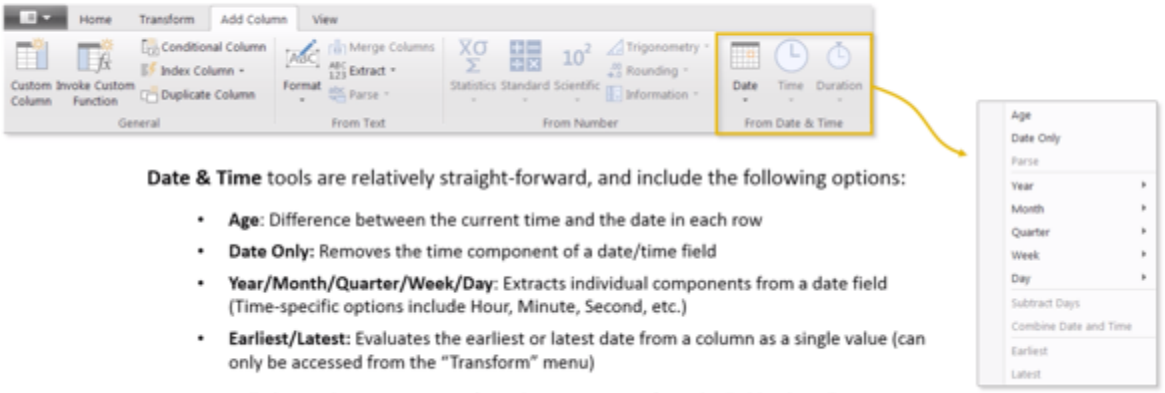
Note: *These tools return a SINGLE value, and are commonly used to explore a table rather than prepare it for loading*

Standard, Scientific and **Trigonometry** tools allow you to apply standard operations (addition, multiplication, division, etc.) or more advanced calculations (power, logarithm, sine, tangent, etc) to each value in a column

Note: *Unlike the Statistics options, these tools are applied to each individual row in the table*

Information tools allow you to define binary flags (*TRUE/FALSE or 1/0*) to mark each row in a column as even, odd, positive or negative

DATE-SPECIFIC TOOLS

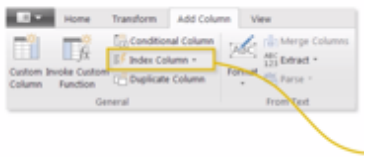


Date & Time tools are relatively straight-forward, and include the following options:

- **Age**: Difference between the current time and the date in each row
- **Date Only:** Removes the time component of a date/time field
- **Year/Month/Quarter/Week/Day**: Extracts individual components from a date field (Time-specific options include Hour, Minute, Second, etc.)
- **Earliest/Latest:** Evaluates the earliest or latest date from a column as a single value (can only be accessed from the "Transform" menu)

Note: *You will almost always want to perform these operations from the "Add Column" menu to build out new fields, rather than transforming an individual date/time column*

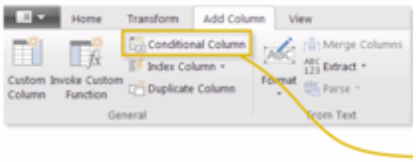
Adding Index Column



Index Columns contain a list of sequential values that can be used to identify each unique row in a table (*typically starting from 0 or 1*)

These columns are often used to create **unique IDs** that can be used to form relationships between tables

Adding Conditional Columns

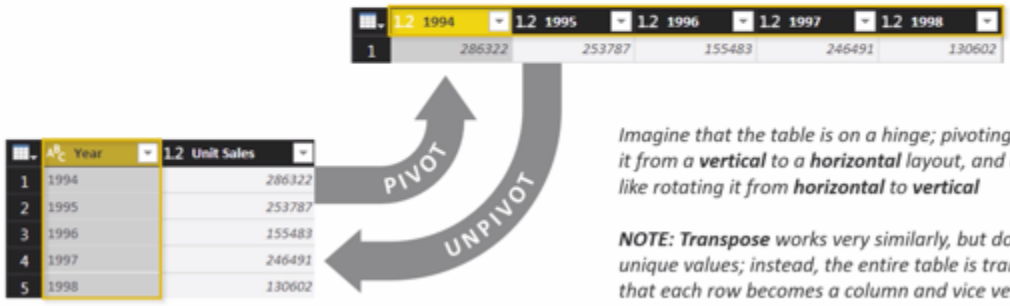


*In this case we're creating a new conditional column called "**QuantityType**", which depends on the values in the "**OrderQuantity**" column, as follows:*

- *If OrderQuantity =1, QuantityType = "**Single Item**"*
- *If OrderQuantity >1, QuantityType = "**Multiple Items**"*
- *Otherwise QuantityType = "**Other**"*

PIVOTING & UNPIVOTING

"Pivoting" is a fancy way to describe the process of turning **distinct row values** into **columns** (*"pivoting"*) or turning **columns** into **rows** (*"unpivoting"*)



*Imagine that the table is on a hinge; pivoting is like rotating it from a **vertical** to a **horizontal** layout, and unpivoting is like rotating it from **horizontal** to **vertical***

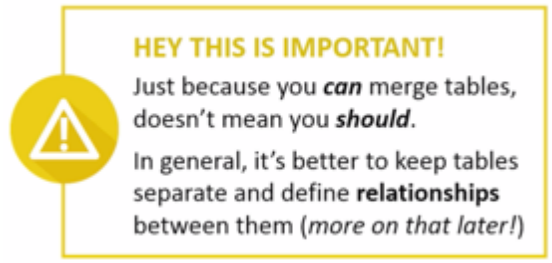
***NOTE: Transpose** works very similarly, but doesn't recognize unique values; instead, the entire table is transformed so that each row becomes a column and vice versa*

Merge Queries

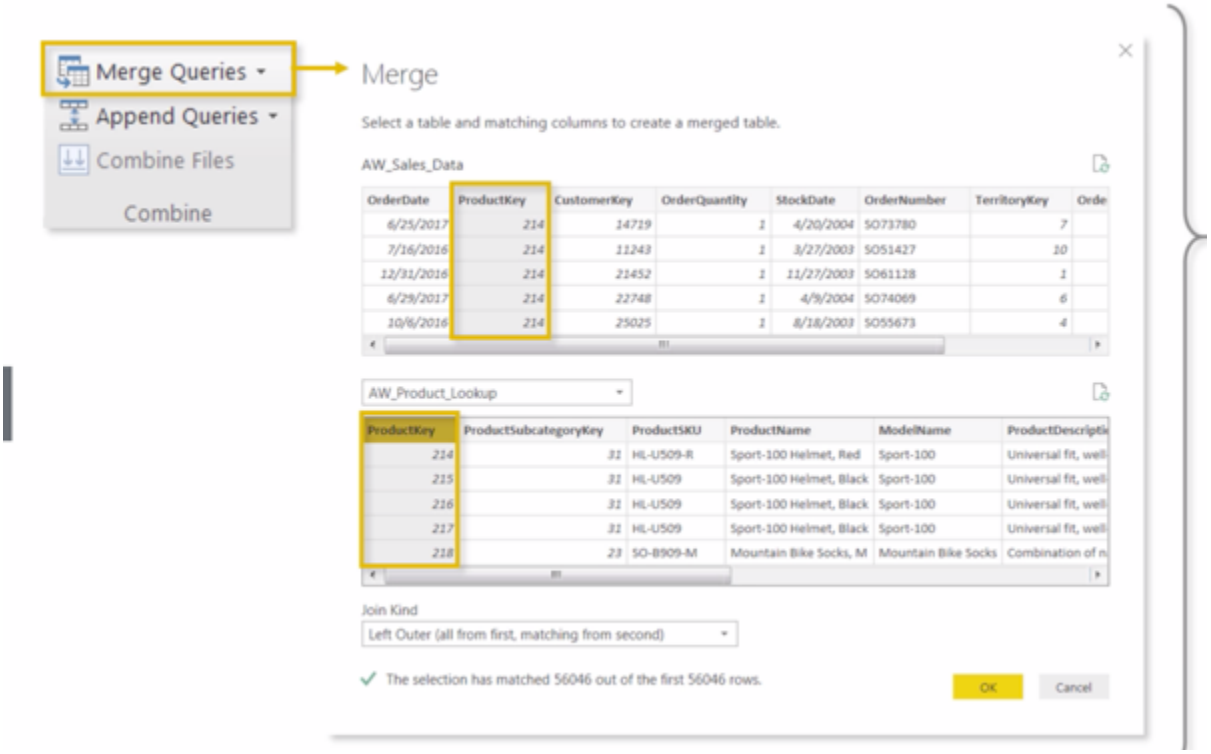
Merging queries allows you to **join tables** based on a common column (like VLOOKUP)

In this case we're merging the **AW_Sales_Data** table with the **AW_Product_Lookup** table, which share a common "*ProductKey*" column

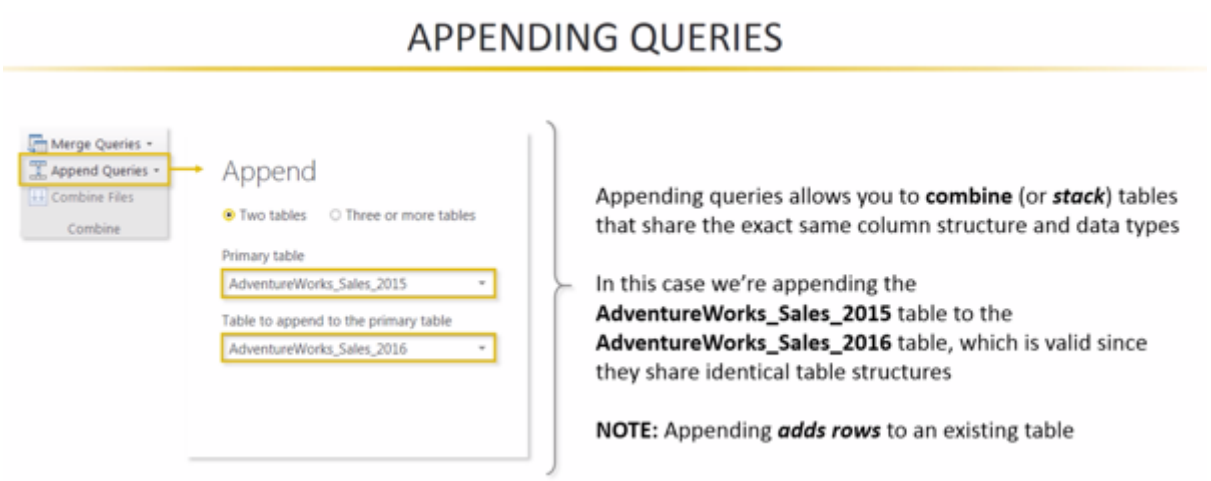
NOTE: Merging ***adds columns*** to an existing table



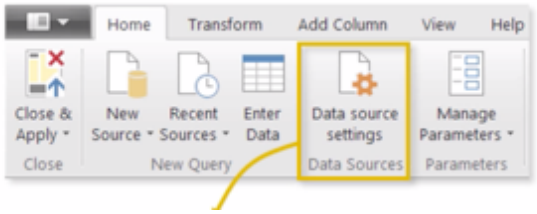
Also you can choose how you want to merge – Left Outer Join, Right Outer Join, Full outer join etc



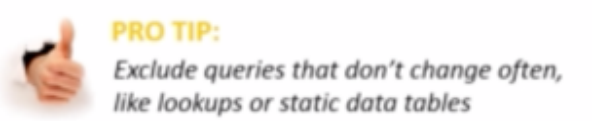
Append Queries

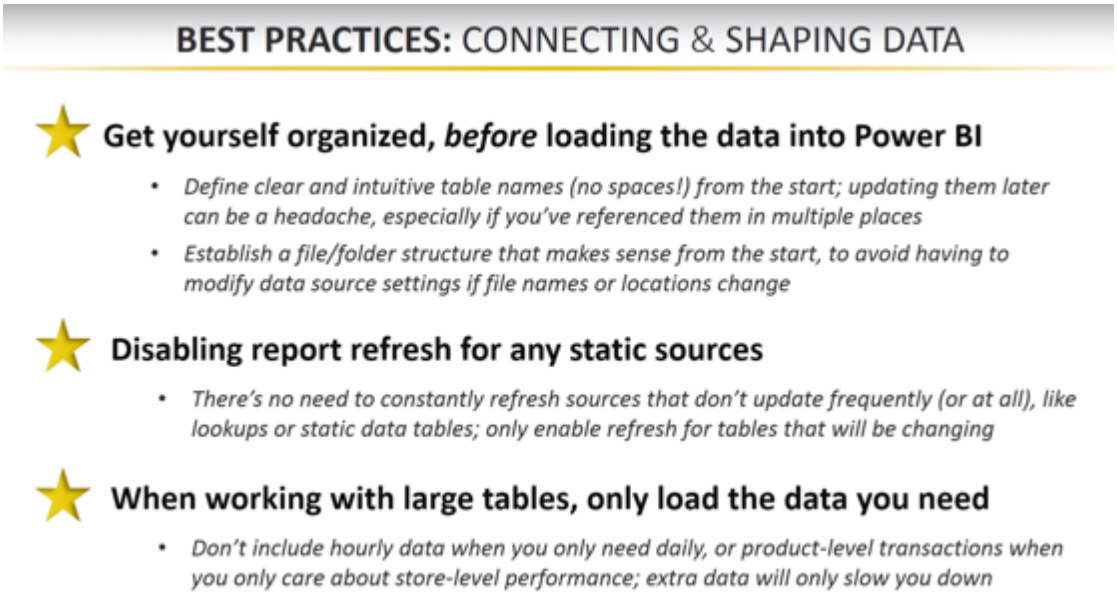


Data Source Settings allows you to specify/change location (for eg) for source file on your machine which you want to work with in Power BI



Refresh data – Can refresh all tables automatically or can only refresh required tables individually.
Should not include Master / Lookup files for auto refresh as these do not change often however transaction data would update frequently so would require refresh.

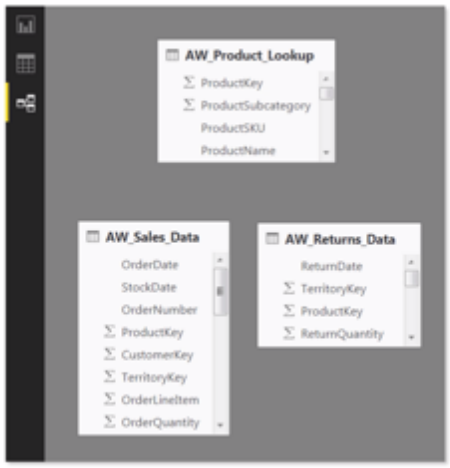




After connecting and loading all data sources into Power BI, the goal for next phase is to create table relationships so we can blend all this information together within a single normalized data model.

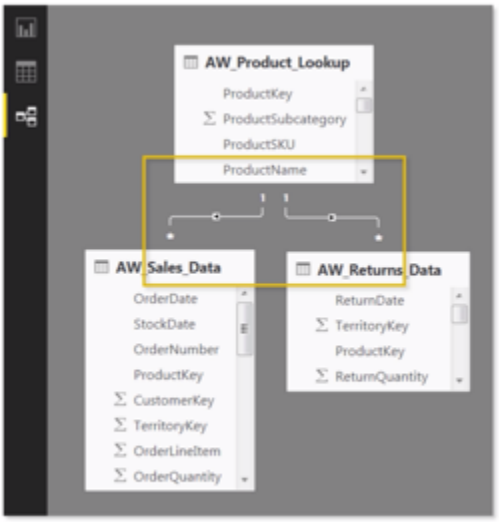
Phase 2 of Power BI CREATING TABLE RELATIONSHIPS AND DATA MODELS

What is Data Model?



This **IS NOT** a data model ☹

- This is a collection of independent tables, which share no connections or relationships



This **IS** a data model! ☺

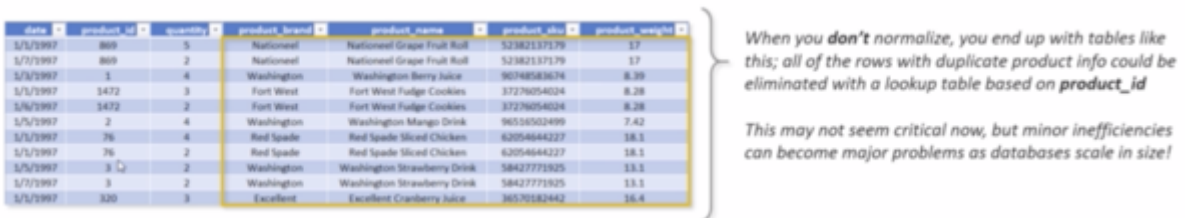
- The tables are connected via relationships, based on the common *ProductKey* field

DATABASE NORMALIZATION

Normalization is the process of organizing the tables and columns in a relational database to reduce redundancy and preserve data integrity. It's commonly used to:

- **Eliminate redundant data** to decrease table sizes and improve processing speed & efficiency
- **Minimize errors and anomalies** from data modifications (inserting, updating or deleting records)
- **Simplify queries** and structure the database for meaningful analysis

TIP: In a normalized database, each table should serve a ***distinct*** and ***specific*** purpose (*i.e. product information, dates, transaction records, customer attributes, etc.*)



date	product_id	quantity	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	Excellent	Excellent Cranberry Juice	36570182442	16.4

*When you **don't** normalize, you end up with tables like this; all of the rows with duplicate product info could be eliminated with a lookup table based on **product_id***

This may not seem critical now, but minor inefficiencies can become major problems as databases scale in size!

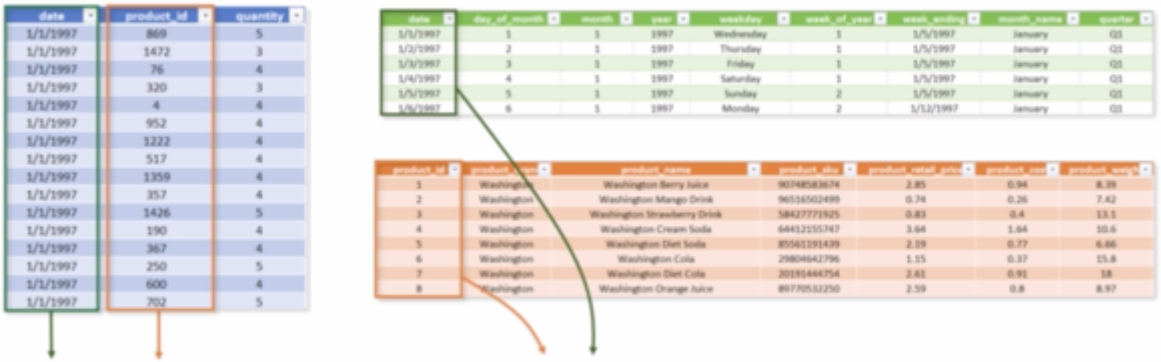
DATA TABLES VS. LOOKUP TABLES

Models generally contain two types of tables: **data** (or "*fact*") tables, and **lookup** (or "*dimension*") tables

- **Data tables** contain *numbers* or *values*, typically at a granular level, with ID or "*key*" columns that can be used to create table relationships
- **Lookup tables** provide descriptive, often text-based *attributes* about each dimension in a table

Primary Keys are Unique, Foreign Keys often contain duplicates. Example as below –

PRIMARY VS. FOREIGN KEYS



date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

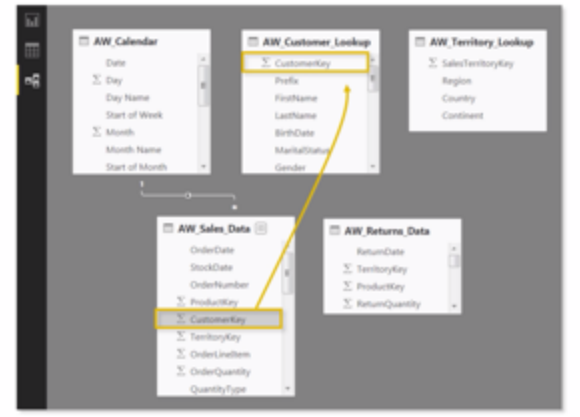
product_id	product_brand	product_name	product_sku	product_retail_price	product_cost	product_weight
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20193444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

These columns are **foreign keys**; they contain *multiple* instances of each value, and are used to match the **primary keys** in related lookup tables

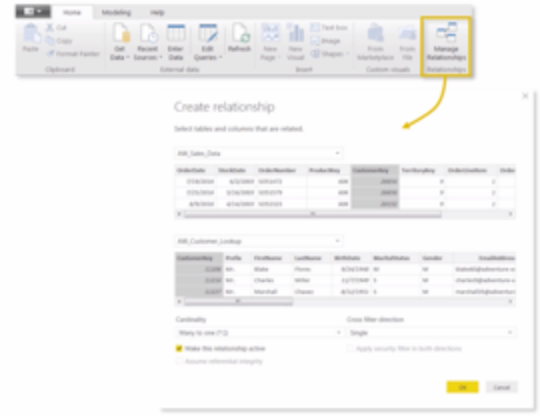
These columns are **primary keys**; they *uniquely* identify each row of a table, and match the **foreign keys** in related data tables

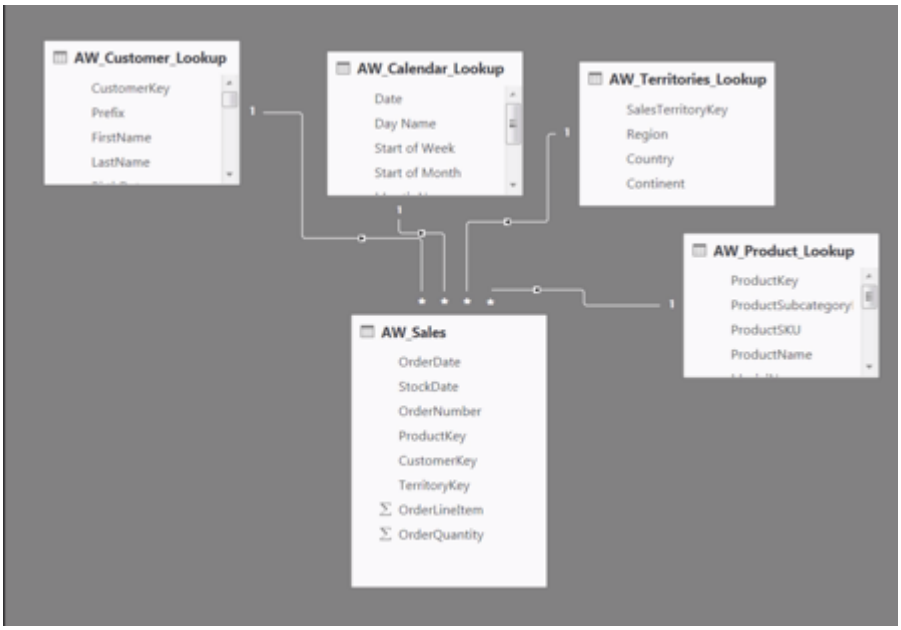
CREATING TABLE RELATIONSHIPS

Option 1: Click and drag to connect primary and foreign keys within the **Relationships** pane

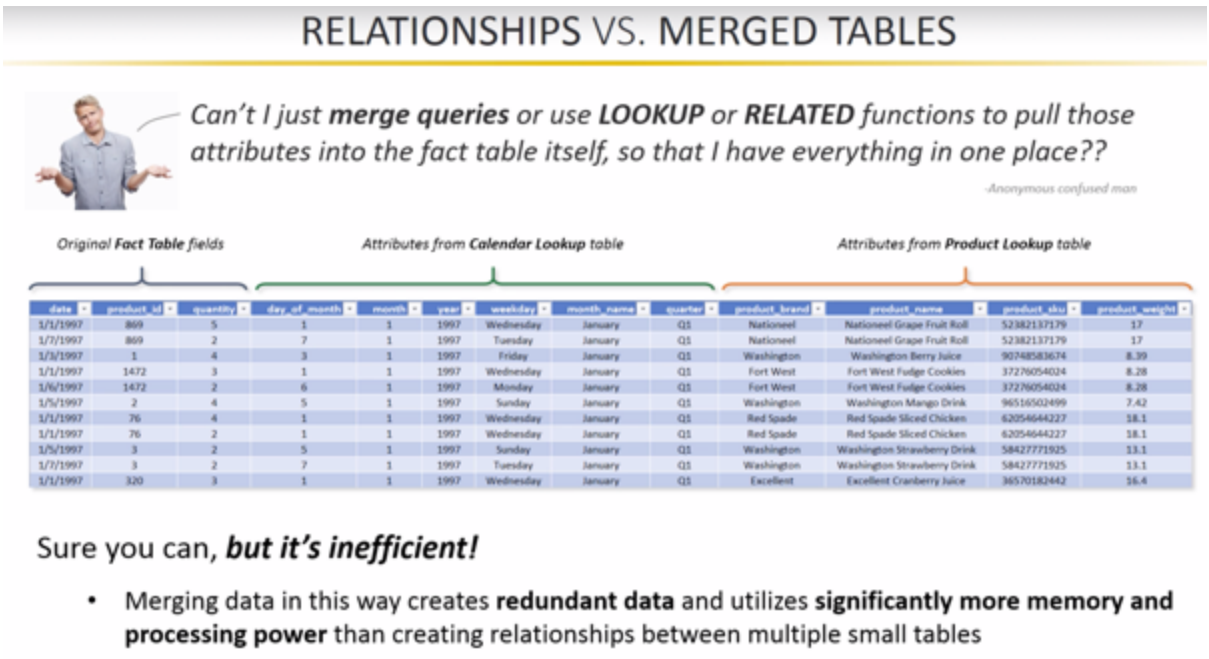


Option 2: Add or detect relationships using the "**Manage Relationships**" dialog box

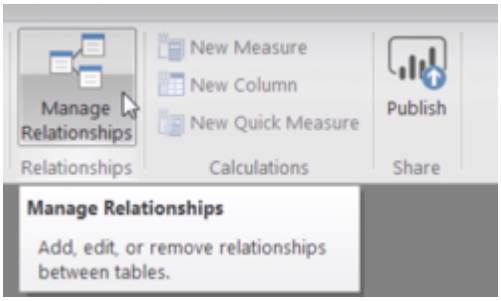




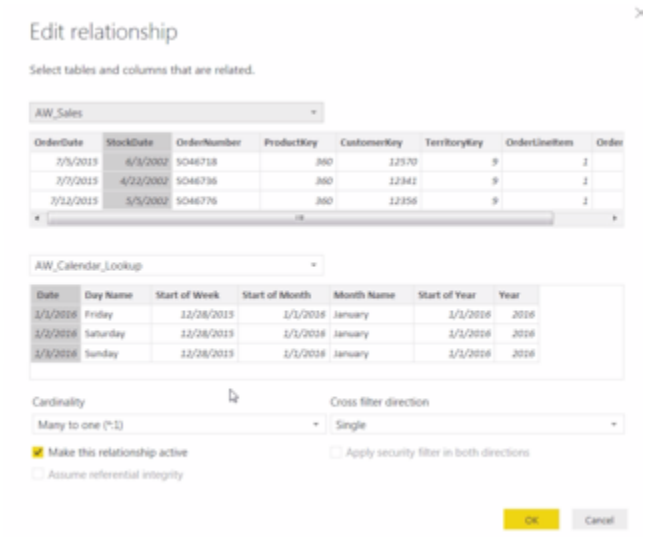
Tables need not essentially have to have just the Primary key or Foreign key. They can have both.



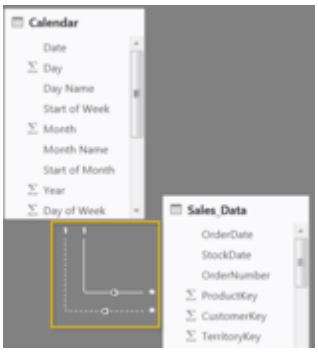
If you want to add/ delete / modify relationships can do via Manage Relationships



Another way is just to click on that relationship line and u will see the Edit relationship menu.



Active vs Inactive Relationships



Currently as seen above Date field has 2 relationships but at a time only 1 can be active.
The dotted one is inactive while the solid line is active one.
In Edit relationship you can make relationship active / inactive. You can toggle these.

RELATIONSHIP CARDINALITY

Cardinality refers to the *uniqueness of values* in a column

- For our purposes, all relationships in the data model should follow a "**one-to-many**" cardinality; **one** instance of each *primary key*, but potentially **many** instances of each *foreign key*

Other Types of Cardinality apart from 1 to Many Cardinality are –

Many to Many
In this u have multiple instances in both the tables & Power BI would give error as u cannot create a relationship because one of the columns must have unique values.

One to One
Not as bad as Many to Many. Can create 1 to 1 relationships but they are just a bit inefficient as shown below rather they can be merged together

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

product_id	product_price
4	$3.64
5	$2.19
7	$2.61
8	$2.59

To eliminate the inefficiency, you could simply ***merge the two tables*** *into a single, valid lookup*

NOTE: *this still respects the laws of normalization, since all rows are unique and capture attributes related to the primary key*

product_id	product_name	product_sku	product_price
4	Washington Cream Soda	64412155747	$3.64
5	Washington Diet Soda	85561191439	$2.19
7	Washington Diet Cola	20191444754	$2.61
8	Washington Orange Juice	89770532250	$2.59

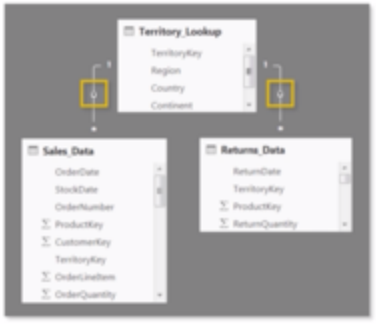
HEY THIS IS IMPORTANT!

In general, never create ***direct relationships*** *between data tables; instead,* ***connect them through shared lookups***

This way u can see both Order Quantity and Return quantity side by side broken down by product name as both Sales and Return table are linked to Product lookup table.
Instead of directly connecting Sales and Return table which are both transaction tables you connect them via common lookup Product table. Below is matrix view in viusliazation.

ProductName	OrderQuantity	ReturnQuantity
All-Purpose Bike Stand	234	8
AWC Logo Cap	4151	46
Bike Wash - Dissolver	1706	25
Classic Vest, L	182	4
Classic Vest, M	182	7
Classic Vest, S	157	8
Fender Set - Mountain	3960	54
Half-Finger Gloves, L	840	18
Half-Finger Gloves, M	918	16
Half-Finger Gloves, S	886	15
Hitch Rack - 4-Bike	302	8
HL Mountain Tire	1305	49

FILTER FLOW



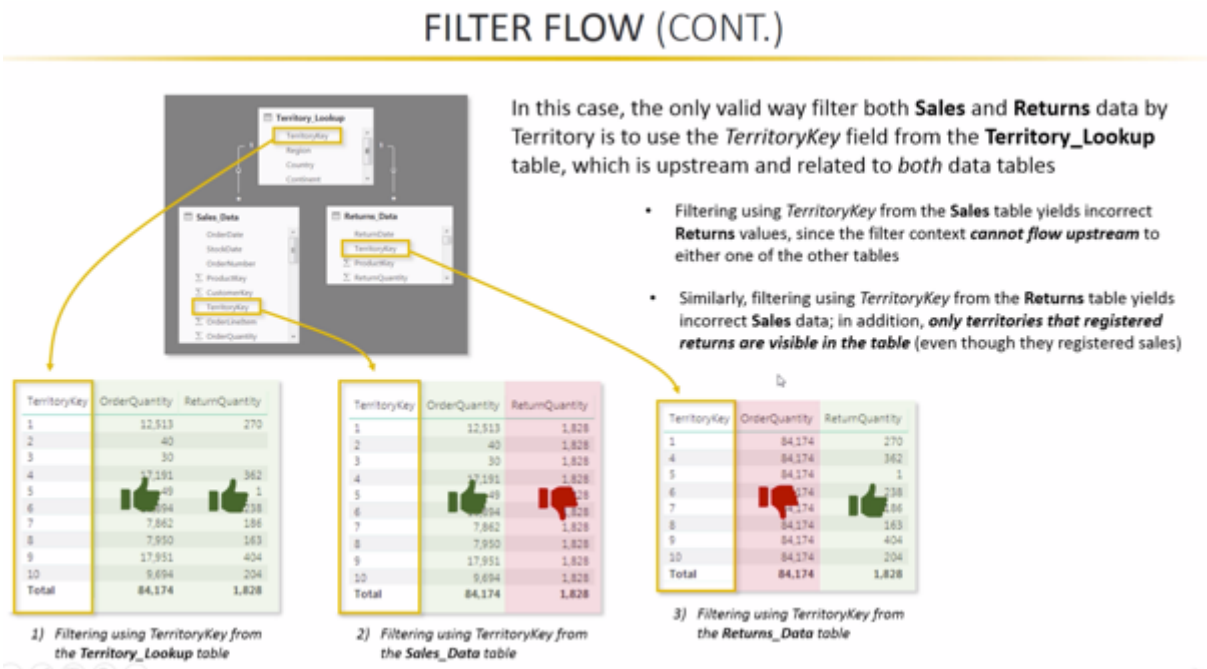
Here we have two data tables (**Sales_Data** and **Returns_Data**), connected to **Territory_Lookup**

Note the filter directions (shown as arrows) in each relationship; by default, **these will point from the "one" side of the relationship** *(lookups)* **to the "many" side** *(data)*

- When you filter a table, that filter context is passed along to all related "*downstream*" tables (following the direction of the arrow)
- Filters **cannot** flow "*upstream*" (against the direction of the arrow)

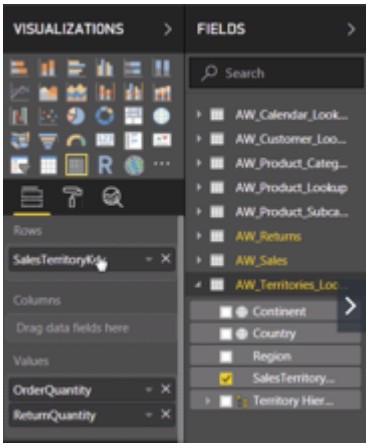
PRO TIP:

*Arrange your lookup tables **above** your data tables in your model as a visual reminder that filters flow "**downstream**"*



In Table 3 we do not get to even see Territory 2 and 3 although they registered orders because in 3rd case table is struck in Return Quantity (cannot flow upward) which does not have data for territories 2 and 3. So filter should go via lookup to data table.

Below shows how u filter from Territories look up table to view Order Quantity and Sales Quantity in Values. If u filter from territory from data tables results will be incorrect as explained above.



To avoid Filter problem

PRO TIP:

*Hide the **foreign key columns** in your data tables to force users to filter using the **primary keys** in the lookup tables*

BEST PRACTICES: DATA MODELING

Focus on building a normalized model from the start

- *Make sure that each table in your model serves a single, distinct purpose*
- *Use relationships vs. merged tables; long & narrow tables are better than short & wide*

Organize lookup tables *above* data tables in the diagram view

- *This serves as a visual reminder that filters flow "**downstream**"*

Avoid complex cross-filtering unless absolutely necessary

- *Don't use two-way filters when 1-way filters will get the job done*

Hide fields from report view to prevent invalid filter context

- *Recommend hiding foreign keys from data tables, so that users can only access valid fields*

Phase 3 of Power BI ANALYSING DATA WITH DAX CALCULATIONS IN POWER BI

Using Power BI Formula Language which is Data Analysis Expressions or DAX to build calculated columns and measures- Metrics or Quantitative values to analyze further using visualization.

Data Analysis Expressions, commonly known as **DAX**, is the formula language that drives Power BI. With DAX, you can:

- Add ***calculated columns*** and ***measures*** to your model, using intuitive syntax
- Go beyond the capabilities of traditional "grid-style" formulas, with powerful and flexible functions built specifically to work with relational data models

Calculated Columns live inside your tables while Measures do not, they are mainly used for Visualization purpose.

CALCULATED COLUMNS

CALCULATED COLUMNS

Calculated columns allow you to add new, formula-based columns to tables

- No "A1-style" references; calculated columns refer to **entire tables** or **columns**
- Calculated columns generate values for each row, which are **visible within tables in the Data view**
- Calculated columns understand **row context**; they're great for defining properties based on information in each row, but generally useless for aggregation (*SUM, COUNT, etc*)

HEY THIS IS IMPORTANT!

As a rule of thumb, use calculated columns when you want to "stamp" static, fixed values to each row in a table (*or use the Query Editor!*)

DO NOT use calculated columns for aggregation formulas, or to calculate fields for the "Values" area of a visualization (use **measures** instead)

PRO TIP:

*Calculated columns are typically used for **filtering** data, rather than creating numerical values*

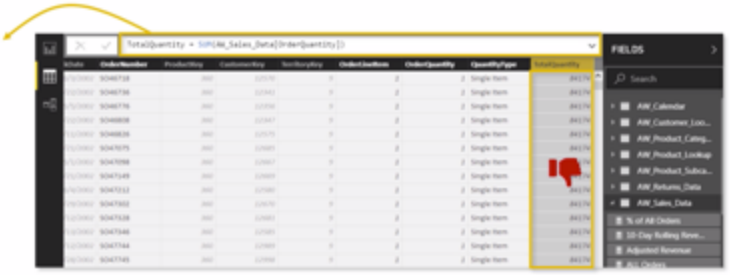
CALCULATED COLUMNS (EXAMPLES)

In this case we've added a **calculated column** named **"Parent"**, which equals "***Yes***" if the [TotalChildren] field is greater than 0, and "***No***" otherwise (*just like Excel!*)

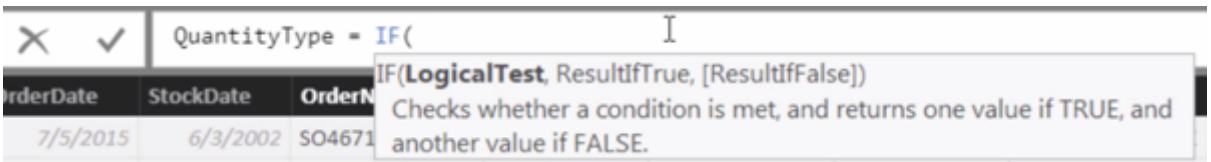
- Since calculated columns understand **row context**, a new value is calculated in each row based on the value in the [TotalChildren] column
- This is a valid use of calculated columns; it creates a new row "property" that we can now use to filter or segment any related data within the model

Here we're using an aggregation function (SUM) to calculate a new column named **TotalQuantity**

- Since calculated columns do not understand **filter context**, the same grand total is returned in *every single row* of the table
- This is not a valid use of calculated columns; these values are statically "stamped" onto the table and can't be filtered, sliced, subdivided, etc.



DAX Formula Example – Creating a new calculated column



```
QuantityType = IF(AW_Sales[OrderQuantity]>1,"Multiple Items","Single Item")
```

You get a new calculated column

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	QuantityType
7/5/2015	6/3/2002	SO46718	360	12570	9	1	1	Single Item
7/7/2015	4/22/2002	SO46736	360	12341	9	1	1	Single Item

MEASURES

Measures are DAX formulas used to generate new calculated values

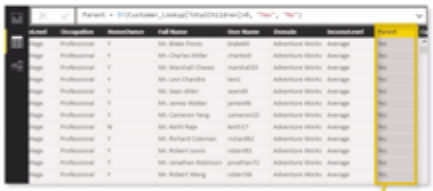
- Like calculated columns, measures reference **entire tables** or **columns** (*no A1-style or "grid" references*)
- *Unlike* calculated columns, **measure** values aren't visible within tables; they can only be "*seen*" within a visualization like a chart or matrix (*similar to a calculated field in an Excel pivot*)
- Measures are evaluated based on **filter context**, which means they recalculate when the fields or filters around them change (*like when new row or column labels are pulled into a matrix or when new filters are applied to a report*)

HEY THIS IS IMPORTANT!
As a rule of thumb, use measures (*vs. calculated columns*) when a single row can't give you the answer (*in other words, when you need to **aggregate***)

PRO TIP:
*Use measures to create **numerical, calculated values** that can be analyzed in the "**values**" field of a report visual*

CALCULATED COLUMNS

- Values are calculated based on information from each row of a table (has **row context**)
- Appends static values to each row in a table and stores them in the model (*which increases file size*)
- Recalculate on data source refresh or when changes are made to component columns
- Primarily used as **rows**, **columns**, **slicers** or **filters**



*Calculated columns "live" in **tables***

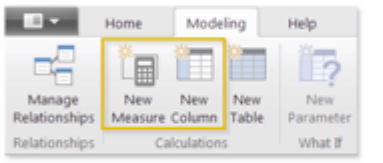
MEASURES

- Values are calculated based on information from any filters in the report (has **filter context**)
- Does not create new data in the tables themselves (*doesn't increase file size*)
- Recalculate in response to any change to filters within the report
- Almost *always* used within the **values** field of a visual

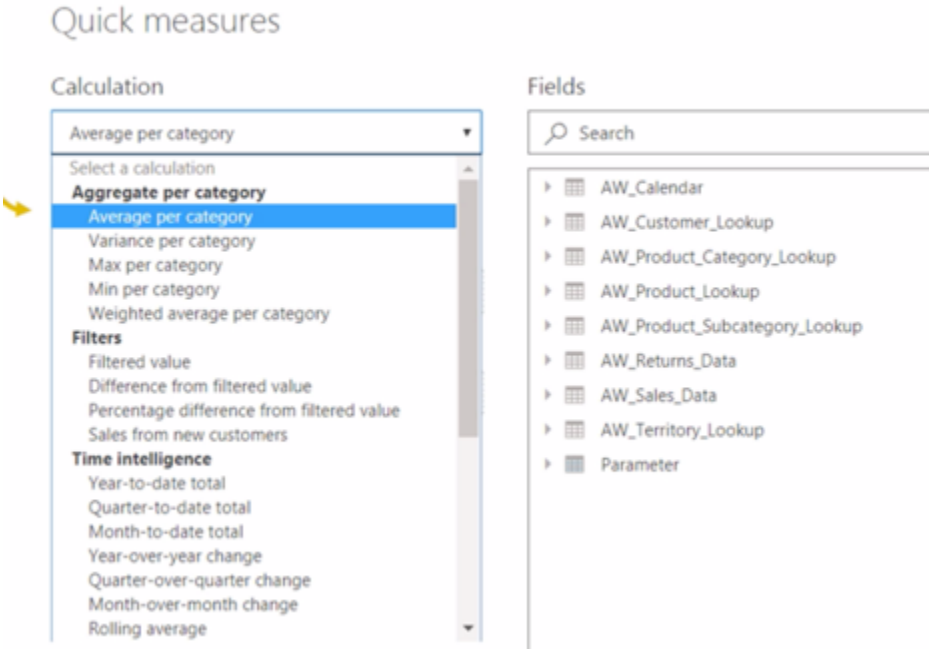
*Measures "live" in **visuals***

ADDING COLUMNS & MEASURES

Option 1: Select "**New Measure**" or "**New Column**" from the **Modeling** tab



Option 2: Right-click within the **table** (in the **Data** view) or the **Field List** (in either the **Data** or **Report** view)

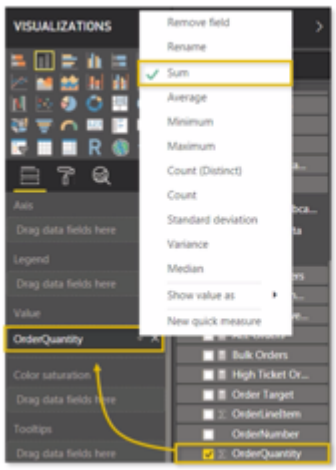


Quick Measures are pre-built formula templates that allow you to drag and drop fields, rather than write DAX from scratch

While these tools can be helpful for defining more complex measures (*like weighted averages or time intelligence formulas*), they encourage laziness and don't help you understand the fundamentals of DAX

There are 2 types of Measures

IMPLICIT MEASURE
EXPLICIT MEASURE



*Example of an **implicit measure***

Implicit measures are created when you drag raw numerical fields (like "*OrderQuantity*") into the values pane of a visual and manually select the aggregation mode (*Sum, Average, Min/Max, etc*)

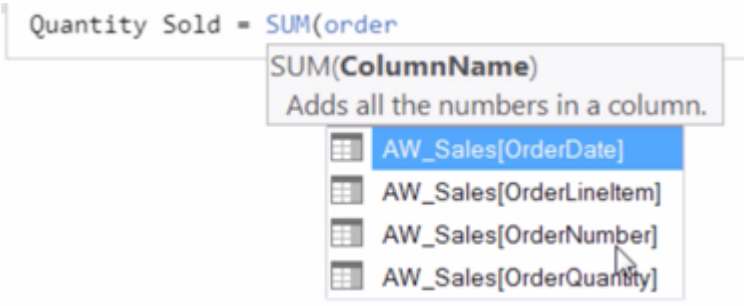
Explicit measures are created by actually entering DAX functions (or adding "*quick measures*") to define calculated columns or measures

HEY THIS IS IMPORTANT!

Implicit measures are *only accessible* within the specific visualization in which it was created, and cannot be referenced elsewhere

Explicit measures can be used anywhere in the report, and referenced within other DAX calculations to create "measure trees"

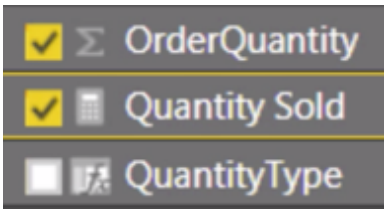
Example of MEASURE




```
Quantity Sold = SUM(AW_Sales[OrderQuantity])
```

Symbols as shown below to identify where a raw field, calculated column or measure?

Sigma for raw field
Calculator for measure
Fx for calculated Column



DAX SYNTAX

MEASURE NAME

- ***Note:*** *Measures are always surrounded in brackets (i.e.* ***[Total Quantity]****) when referenced in formulas, so spaces are OK*

Referenced TABLE NAME

Referenced COLUMN NAME

Total Quantity: =SUM(Transactions[quantity])

FUNCTION NAME

- Calculated columns don't always use functions, but measures do:

Note: *This is a* ***"fully qualified"*** *column, since it's preceeded by the table name -- table names with spaces must be surrounded by* ***single quotes****:*

- *Without a space:* ***Transactions****[quantity]*
- *With a space:* ***'Transactions Table'****[quantity]*

OPERATORS – Arithmetic, Comparison and Logical/Text Operators

DAX OPERATORS

Arithmetic Operator	Meaning	Example
+	Addition	2 + 7
-	Subtraction	5 – 3
*	Multiplication	2 * 6
/	Division	4 / 2
^	Exponent	2 ^ 5

Comparison Operator	Meaning	Example
=	Equal to	[City]="Boston"
>	Greater than	[Quantity]>10
<	Less than	[Quantity]<10
>=	Greater than or equal to	[Unit_Price]>=2.5
<=	Less than or equal to	[Unit_Price]<=2.5
<>	Not equal to	[Country]<>"Mexico"

Text/Logical Operator	Meaning	Example
&	Concatenates two values to produce one text string	[City] & " " & [State]
&&	Create an AND condition between two logical expressions	([State]="MA") && ([Quantity]>10)
|| (double pipe)	Create an OR condition between two logical expressions	([State]="MA") || ([State]="CT")
IN	Creates a logical OR condition based on a given list (using curly brackets)	'Store Lookup'[State] IN { "MA", "CT", "NY" }

COMMON FUNCTION CATEGORIES

MATH & STATS Functions

*Basic **aggregation** functions as well as **"iterators"** evaluated at the row-level*

Common Examples:

- SUM
- AVERAGE
- MAX/MIN
- DIVIDE
- COUNT/COUNTA
- COUNTROWS
- DISTINCTCOUNT

Iterator Functions:

- SUMX
- AVERAGEX
- MAXX/MINX
- RANKX
- COUNTX

LOGICAL Functions

*Functions for returning information about values in a given **conditional expression***

Common Examples:

- IF
- IFERROR
- AND
- OR
- NOT
- SWITCH
- TRUE
- FALSE

TEXT Functions

*Functions to manipulate **text strings** or **control formats** for dates, times or numbers*

Common Examples:

- CONCATENATE
- FORMAT
- LEFT/MID/RIGHT
- UPPER/LOWER
- PROPER
- LEN
- SEARCH/FIND
- REPLACE
- REPT
- SUBSTITUTE
- TRIM
- UNICHAR

FILTER Functions

***Lookup** functions based on related tables and **filtering** functions for dynamic calculations*

Common Examples:

- CALCULATE
- FILTER
- ALL
- ALLEXCEPT
- RELATED
- RELATEDTABLE
- DISTINCT
- VALUES
- EARLIER/EARLIEST
- HASONEVALUE
- HASONEFILTER
- ISFILTERED
- USERELATIONSHIP

DATE & TIME Functions

*Basic **date and time** functions as well as advanced **time intelligence** operations*

Common Examples:

- DATEDIFF
- YEARFRAC
- YEAR/MONTH/DAY
- HOUR/MINUTE/SECOND
- TODAY/NOW
- WEEKDAY/WEEKNUM

Time Intelligence Functions:

- DATESYTD
- DATESQTD
- DATESMTD
- DATEADD
- DATESINPERIOD

BASIC DATE & TIME FUNCTIONS

DAY/MONTH/ YEAR()	*Returns the day of the month (1-31), month of the year (1-12), or year of a given date*	=DAY/MONTH/YEAR(Date)
HOUR/MINUTE/ SECOND()	*Returns the hour (0-23), minute (0-59), or second (0-59) of a given datetime value*	=HOUR/MINUTE/SECOND(Datetime)
TODAY/NOW()	*Returns the current date or exact time*	=TODAY/NOW()
WEEKDAY/ WEEKNUM()	*Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year*	=WEEKDAY/WEEKNUM(Date, *[ReturnType]*)
EOMONTH()	*Returns the date of the last day of the month, +/- a specified number of months*	=EOMONTH(StartDate, Months)
DATEDIFF()	*Returns the difference between two dates, based on a selected interval*	=DATEDIFF(Date1, Date2, Interval)

Example

```
Current Age = DATEDIFF(AW_Customer_Lookup[BirthDate],TODAY(),YEAR)
```

BASIC LOGICAL FUNCTIONS (IF/AND/OR)

IF()	*Checks if a given condition is met, and returns one value if the condition is TRUE, and another if the condition is FALSE*	=IF(LogicalTest, ResultIfTrue, *[ResultIfFalse]*)
IFERROR()	*Evaluates an expression and returns a specified value if the expression returns an error, otherwise returns the expression itself*	=IFERROR(Value, ValueIfError)
AND()	*Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE, otherwise returns FALSE*	=AND(Logical1, Logical2)
OR()	*Checks whether one of the arguments is TRUE to return TRUE, and returns FALSE if both arguments are FALSE*	=OR(Logical1, Logical2)

***Note:** Use the **&&** and **||** operators if you want to include more than two conditions!*

Some EXAMPLES-

```
Parent = IF(AW_Customer_Lookup[TotalChildren]>0,"Yes","No")
```

```
Weekend = IF(OR(AW_Calendar_Lookup[Day of Week]=6,AW_Calendar_Lookup[Day of Week]=7),"Weekend","Weekday")
```

```
Weekend = IF(AW_Calendar_Lookup[Day of Week]=6 || AW_Calendar_Lookup[Day of Week]=7,"Weekend","Weekday")
```

TEXT FUNCTIONS

LEN()	*Returns the number of characters in a string*	=LEN(Text)
CONCATENATE()	*Joins two text strings into one*	=CONCATENATE(Text1, Text2)
LEFT/MID/ RIGHT()	*Returns a number of characters from the start/middle/end of a text string*	=LEFT/RIGHT(Text, *[NumChars]*) =MID(Text, StartPosition, NumChars)
UPPER/LOWER/ PROPER()	*Converts letters in a string to upper/lower/proper case*	=UPPER/LOWER/PROPER(Text)
SUBSTITUTE()	*Replaces an instance of existing text with new text in a string*	=SUBSTITUTE(Text, OldText, NewText, *[InstanceNumber]*)
SEARCH()	*Returns the position where a specified string or character is found, reading left to right*	=SEARCH(FindText, WithinText, *[StartPosition], [NotFoundValue]*)

***Note:** Use the **&** operator as a shortcut, or to combine more than two strings!*

Example

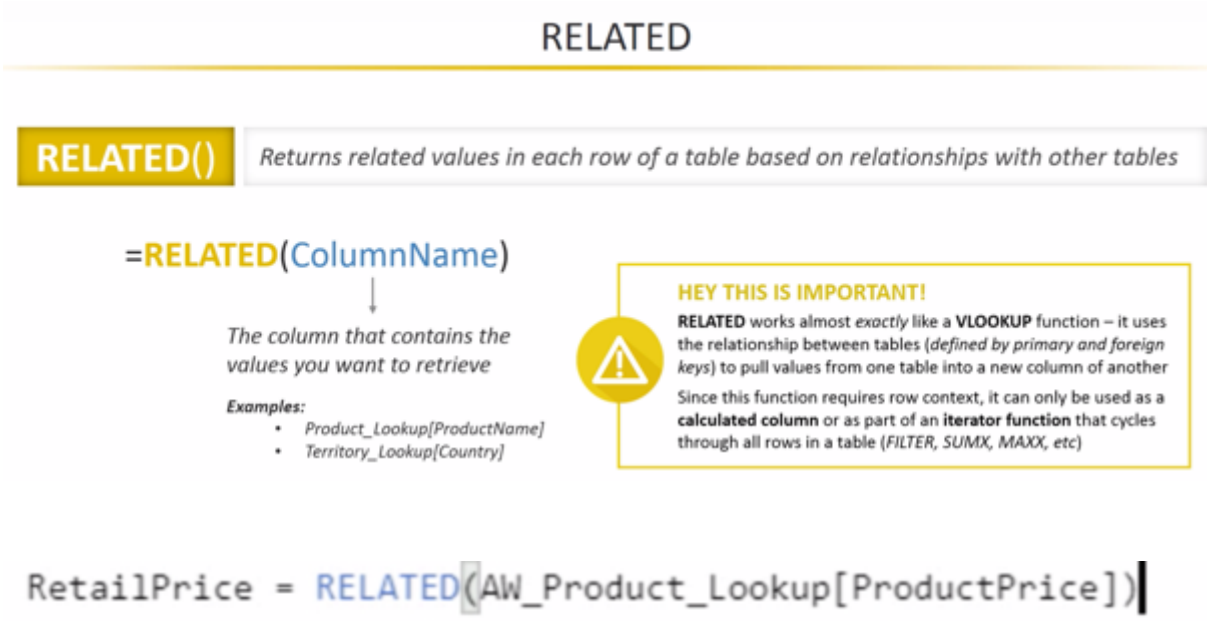
```
FullName_CC = AW_Customer_Lookup[Prefix] & " " & AW_Customer_Lookup[FirstName] & " " & AW_Customer_Lookup[LastName]
```

```
Short Month = LEFT(AW_Calendar_Lookup[Month Name],3)
```

To change to Uppercase'

```
Short Month = UPPER(LEFT(AW_Calendar_Lookup[Month Name],3))
```

Joining Data with RELATED




```
RetailPrice = RELATED(AW_Product_Lookup[ProductPrice])
```

Now using this field added in Sales Table we can create another new column Revenue here

```
Revenue = AW_Sales[RetailPrice] * AW_Sales[OrderQuantity]
```

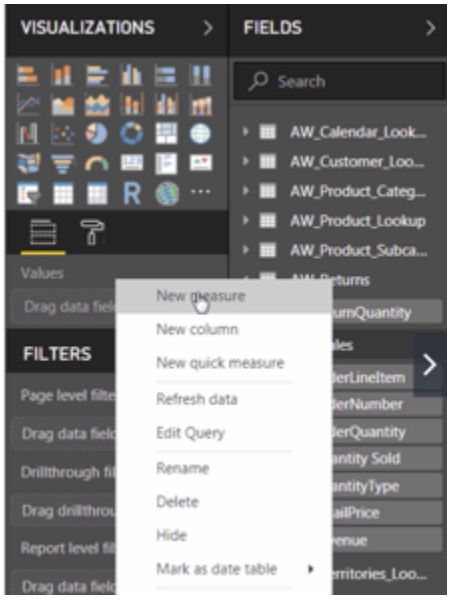
OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	QuantityType	RetailPrice	Revenue
9/9/2016	7/31/2003	SO54042	466	13983	9	7	3	Multiple Items	$23.55	$70.65
6/14/2017	5/2/2004	SO72945	223	25493	10	6	3	Multiple Items	$8.64	$25.92
5/25/2017	3/20/2004	SO71336	223	23841	4	6	3	Multiple Items	$8.64	$25.92

After CALCULATED COLUMNS we now move to MEASURES
Some Functions for creating Measures

BASIC MATH & STATS FUNCTIONS

SUM()	*Evaluates the sum of a column*	=SUM(ColumnName)
AVERAGE()	*Returns the average (arithmetic mean) of all the numbers in a column*	=AVERAGE(ColumnName)
MAX()	*Returns the largest value in a column or between two scalar expressions*	=MAX(ColumnName) *or* =MAX(Scalar1, *[Scalar2]*)
MIN()	*Returns the smallest value in a column or between two scalar expressions*	=MIN(ColumnName) *or* =MIN(Scalar1, *[Scalar2]*)
DIVIDE()	*Performs division and returns the alternate result (or blank) if div/0*	=DIVIDE(Numerator, Denominator, *[AlternateResult]*)

Can add Measures in either Data View or Report View, preferable add in Report view
Select that table in which you want to create the measure, then right click New Measure



Example

```
Quantity Returned = SUM(AW_Returns[ReturnQuantity])
```

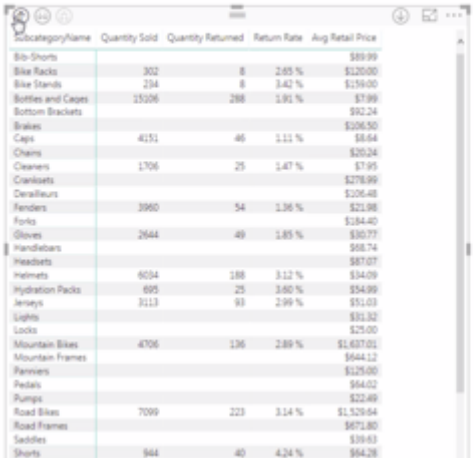
```
Return Rate = DIVIDE([Quantity Returned],[Quantity Sold],
```

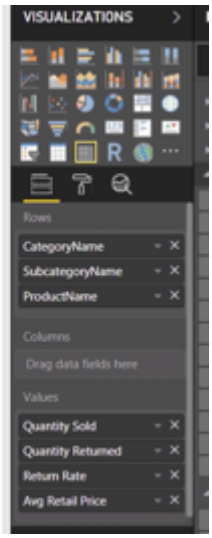
DIVIDE(Numerator, Denominator, **[AlternateResult]**)
Safe Divide function with ability to handle divide by zero case.

```
Return Rate = DIVIDE([Quantity Returned],[Quantity Sold],"No Sales")
```

```
Avg Retail Price = AVERAGE(AW_Product_Lookup[ProductPrice])
```

Drill through Options eg Category, Subcategory, Product Name





COUNTING Measures

COUNT()	*Counts the number of cells in a column that contain numbers*	=COUNT(ColumnName)
COUNTA()	*Counts the number of non-empty cells in a column (numerical and non-numerical)*	=COUNTA(ColumnName)
DISTINCTCOUNT()	*Counts the number of distinct or unique values in a column*	=DISTINCTCOUNT(ColumnName)
COUNTROWS()	*Counts the number of rows in the specified table, or a table defined by an expression*	=COUNTROWS(Table)

Example

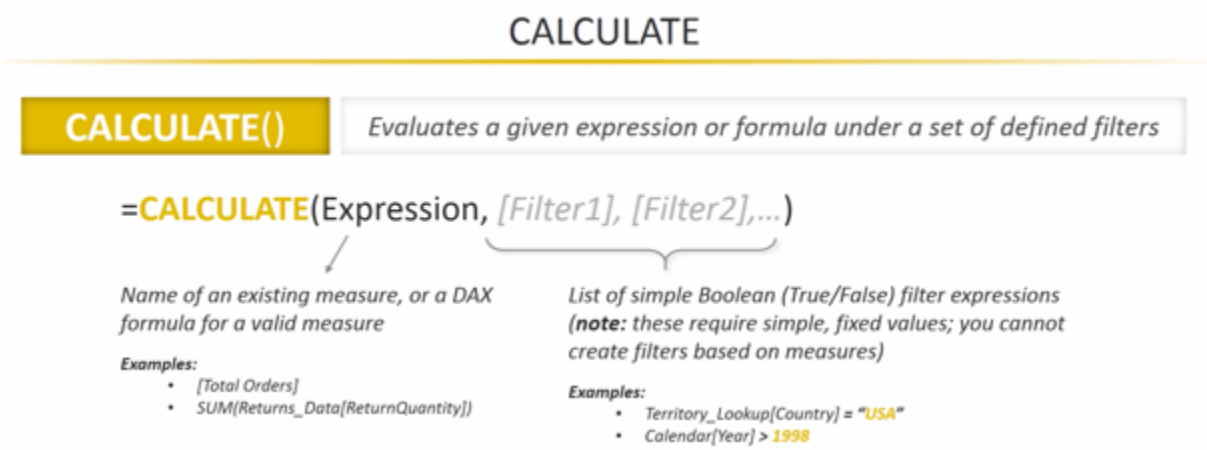
```
Total Returns = COUNT(AW_Returns[ReturnQuantity])
```

COUNTA also accounts for non-numerical values like Text

```
Total Orders = DISTINCTCOUNT(AW_Sales[OrderNumber])
```

Many orders contained multiple line items

CALCULATE function

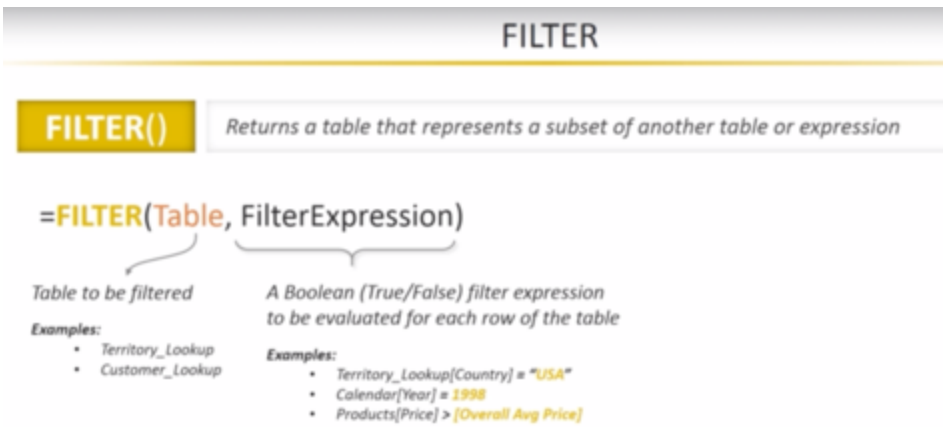


PRO TIP:
*CALCULATE works just like **SUMIF** or **COUNTIF** in Excel, except it can evaluate measures based on ANY sort of calculation (not just a sum, count, etc); it may help to think of it like **"CALCULATEIF"***

Example

```
Weekend Orders = CALCULATE([Total Orders],AW_Calendar_Lookup[Weekend] = "Weekend")
```

FILETR FUNCTION



Example

```
High Ticket Orders = CALCULATE([Total Orders], FILTER(AW_Product_Lookup,AW_Product_Lookup[ProductPrice] > [Overall Avg Price]))
```

FILTER(Table, **FilterExpression**)
Returns a table that has been filtered.

Iterator Functions

ITERATOR ("X") FUNCTIONS

Iterator (or "**X**") **functions** allow you to loop through the same calculation or expression on *each row of a table*, and then apply some sort of aggregation to the results (*SUM*, *MAX*, etc)

=SUMX(Table, Expression)

*Aggregation to apply to calculated rows**

Examples:
- *SUMX*
- *COUNTX*
- *AVERAGEX*
- *RANKX*
- *MAXX/MINX*

Table in which the expression will be evaluated

Examples:
- *Sales*
- *FILTER(Sales, RELATED(Products[Category])="Clothing")*

Expression to be evaluated for each row of the given table

Examples:
- *[Total Orders]*
- *Sales[RetailPrice] * Sales[Quantity]*

Example

```
Total Revenue_Measure = SUMX(AW_Sales, AW_Sales[OrderQuantity] * AW_Sales[RetailPrice])
```

```
Total Revenue = SUMX(AW_Sales, AW_Sales[OrderQuantity] * RELATED(AW_Product_Lookup[ProductPrice]))
```

Some Time Intelligence Functions in Power BI

TIME INTELLIGENCE FORMULAS

Time Intelligence functions allow you to easily calculate common time comparisons:

Performance To-Date
=CALCULATE(Measure, DATESYTD(Calendar[Date]))
Use **DATESQTD** *for Quarters or* **DATESMTD** *for Months*

Previous Period
=CALCULATE(Measure, DATEADD(Calendar[Date], -1, MONTH))
*Select an interval (***DAY***,* **MONTH***,* **QUARTER***, or* **YEAR***) and the # of intervals to compare (i.e. previous month, rolling 10-day)*

Running Total
=CALCULATE(Measure,
DATESINPERIOD(Calendar[Date], MAX(Calendar[Date]), -10, DAY))

BEST PRACTICES: SPEED & PERFORMANCE

Eliminate redundant columns; keep data tables narrow

- *Data tables should ideally only contain only quantitative values and foreign keys; any extra descriptive columns can usually live in a related lookup table*

Imported columns are better than calculated columns

- *When possible, create calculated columns at the source (i.e. in your raw database) or within the Query Editor; this is more efficient than processing those calculations in the Data Model*

Minimize iterator functions (FILTER, SUMX, etc.)

- *Functions that cycle through each row in a table are "expensive", meaning that they take time and consume processing power*

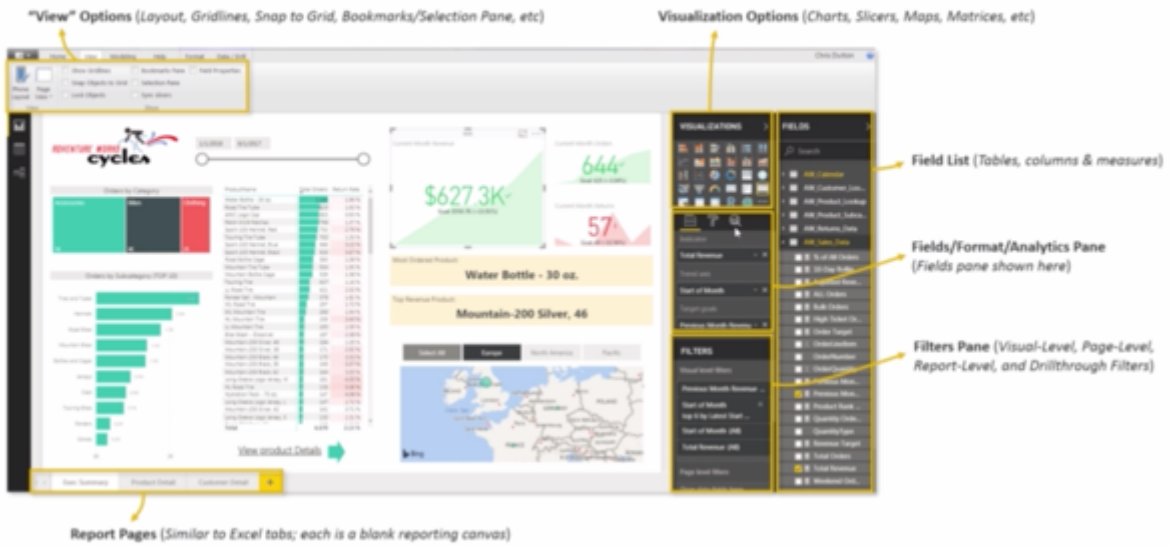
Phase 4 – VISUALIZING DATA WITH POWER BI REPORTS

Visualizing data with Reports and Dashboards

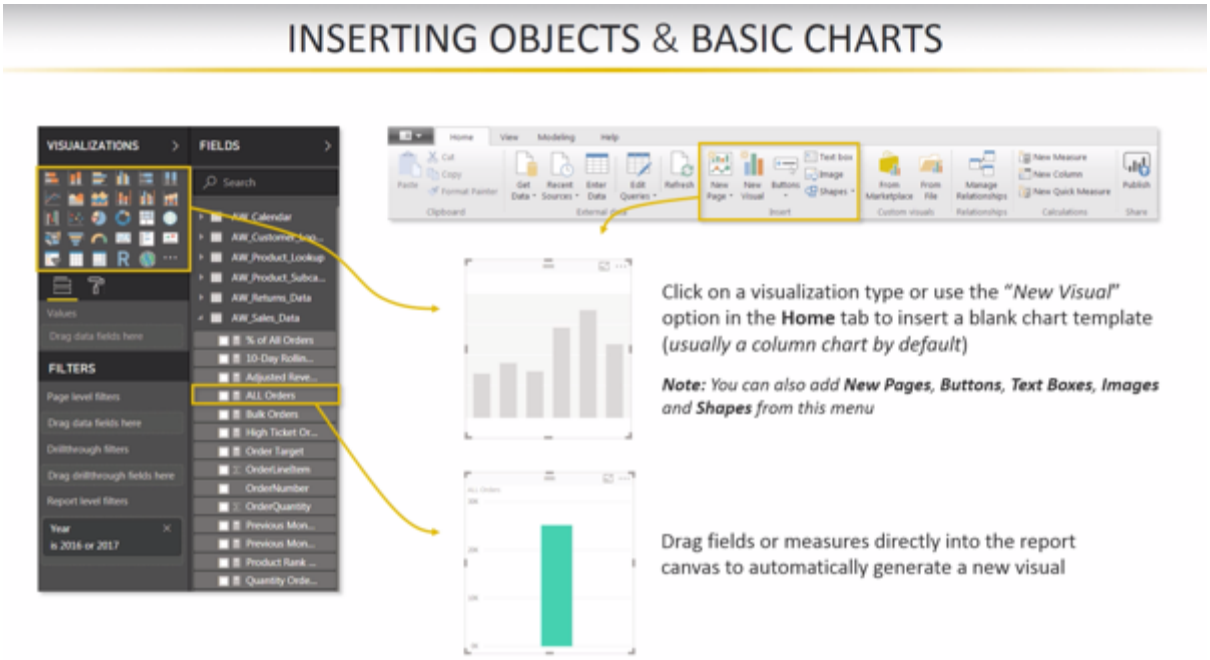
Matrix Visuals, Slicers, Maps, KPI cards, Forecasting tools etc

A Report can have many pages just like there are tabs in Excel

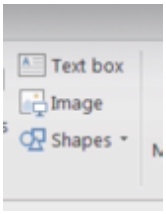
THE POWER BI REPORT VIEW



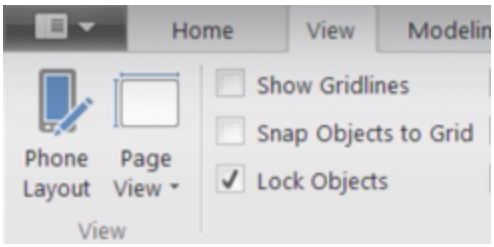
Adding Object to the Report



You can insert any object from your machine like logo into Dashboard using below Image option

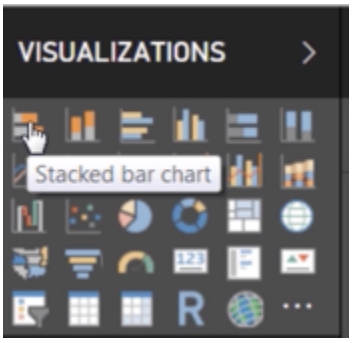


Lock objects option is helpful when you have a number of visuals on your Dashboard so they do not move

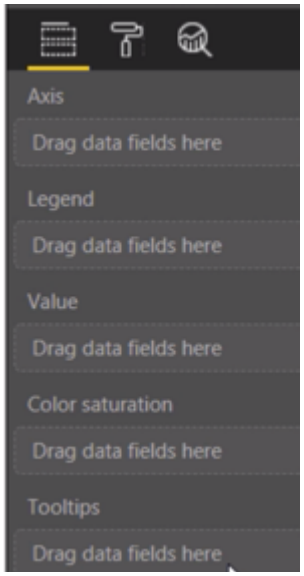


Inserting Basic charts and visuals in Power BI

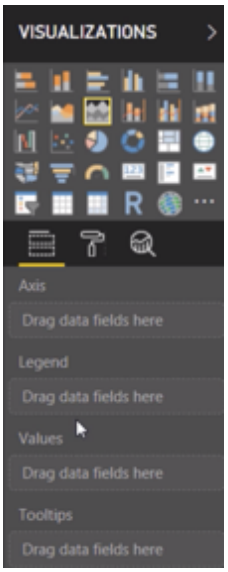
Can choose the visual u want directly from the Visualizations menu for eg u want Stacked Bar just choose that option as below -



Below pane is dynamic and will change according to the visualization you choose
For eg Stacked bar chart can take below options Axis, Value, legend etc



If we choose Stacked area chart we see below pane also changes according to the values it can take



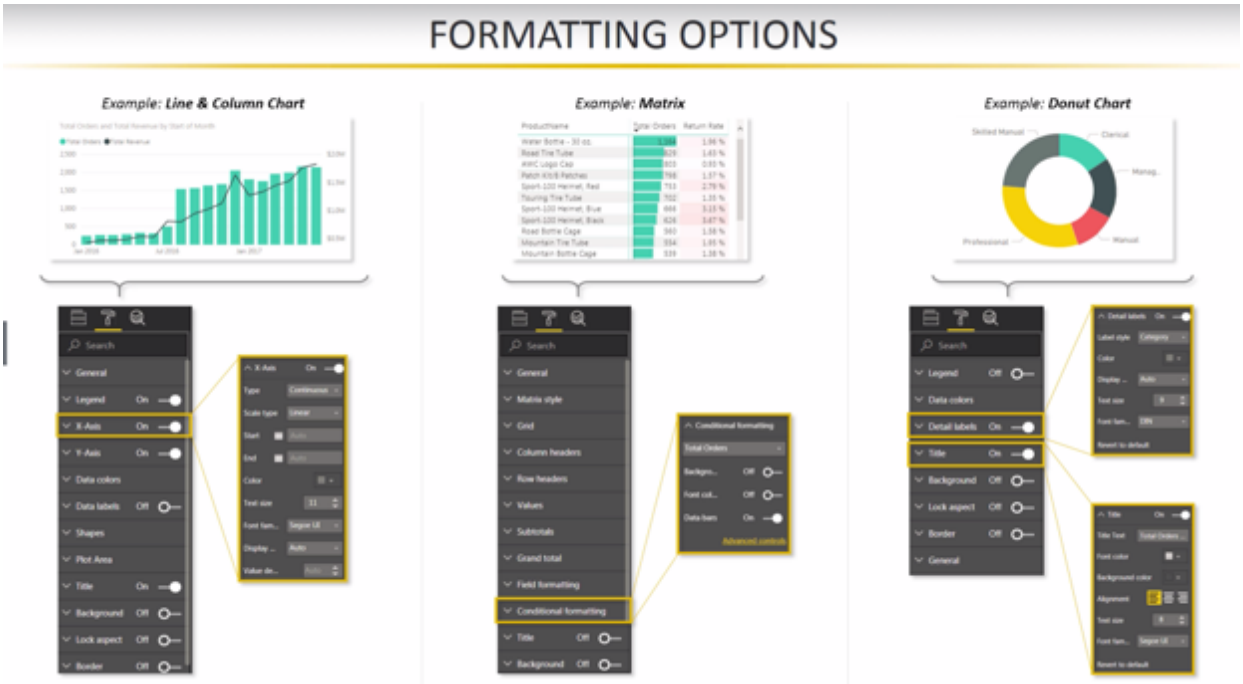
We always start with Fields option because that tells what data we want to visualize and how we want to visualize it.
First Option as below is for Fields

STACKED BAR CHART Visual

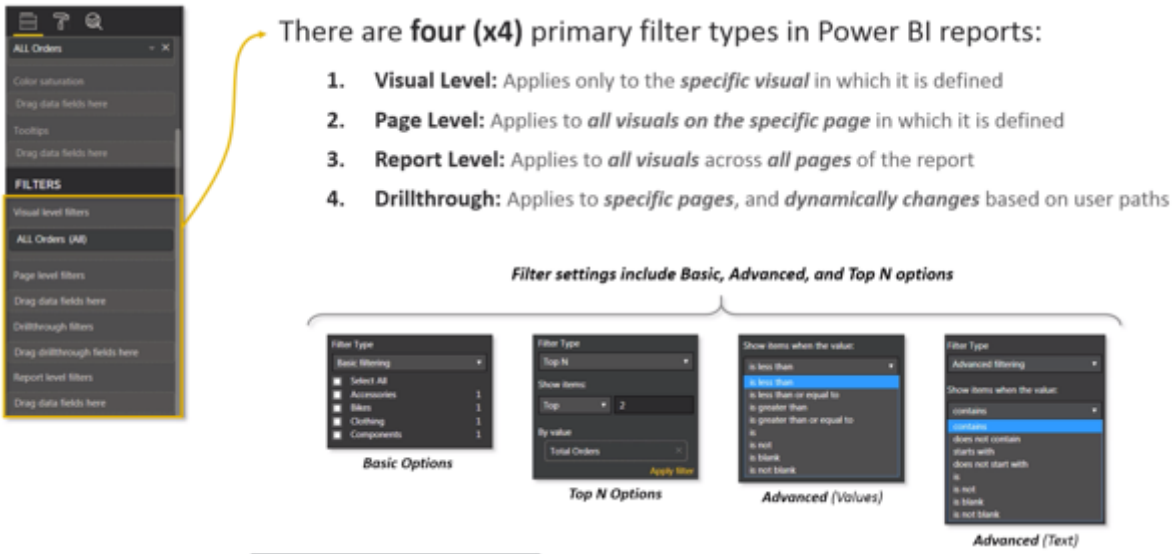
Example for a populated stacked bar chart with Axis as Subcategory Name and Values as Total orders. Values is where you are going to pull your measures in (the aggregations or results you want to visualize)

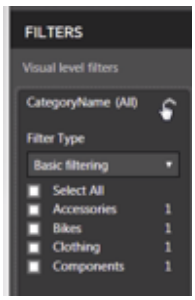
Power BI Report Formatting Options – Customizing the look and feel of your visuals.

Below is from where u will get all the formatting options for visuals you choose

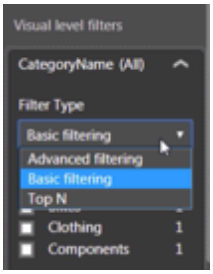


Power BI Report Filtering Options



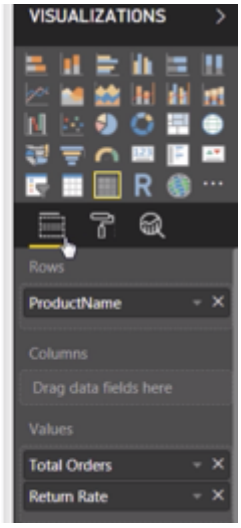


Apart from basic filtering options u can use the Top N Filters to view top 5 products for example



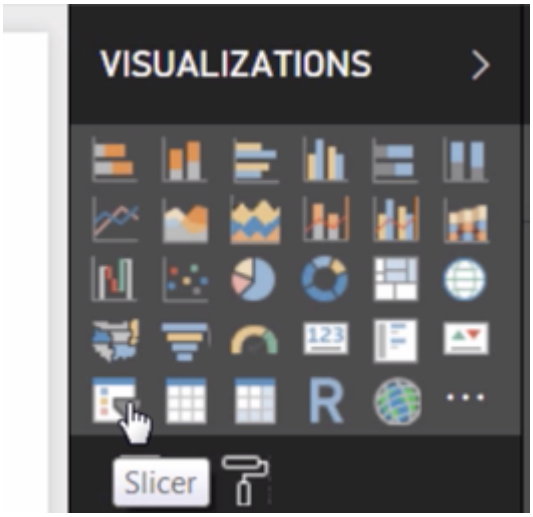
MATRIX Visual

ProductName	Total Orders	Return Rate
Water Bottle - 30 oz.	3,983	1.95 %
Patch Kit/8 Patches	2,952	1.61 %
Mountain Tire Tube	2,846	1.64 %
Road Tire Tube	2,173	1.55 %
Sport-100 Helmet, Red	2,099	3.33 %
AWC Logo Cap	2,062	1.11 %
Sport-100 Helmet, Blue	1,995	3.31 %
Fender Set - Mountain	1,975	1.36 %
Sport-100 Helmet, Black	1,940	2.68 %
Mountain Bottle Cage	1,896	2.02 %
Road Bottle Cage	1,668	1.68 %
Touring Tire Tube	1,364	1.64 %
HL Mountain Tire	1,305	3.75 %
ML Mountain Tire	1,059	1.32 %
LL Road Tire	957	2.26 %
ML Road Tire	868	1.51 %
Touring Tire	863	1.22 %
Bike Wash - Dissolver	850	1.47 %
HL Road Tire	795	3.52 %
LL Mountain Tire	788	2.50 %

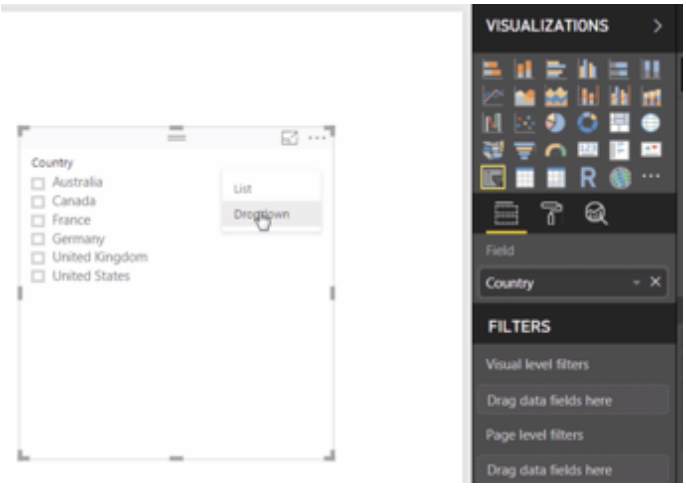


SLICER Visual

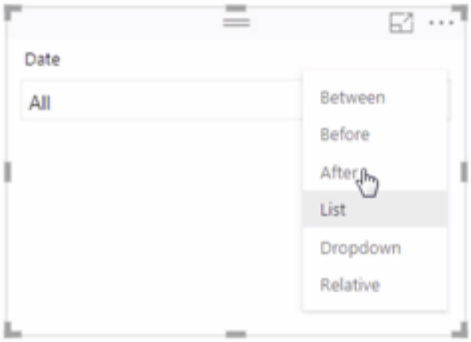
Slicer is just a fancy word for visual filter. It only takes 1 input which is some field.



Example of Slicer, like here for slicer for country
If you choose a specific country it will show all other visualizations on Dashboard corresponding to that country only.



Slicer for Date

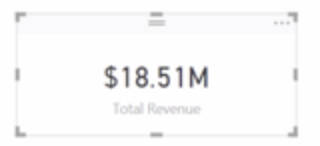


CARDS VISUALS

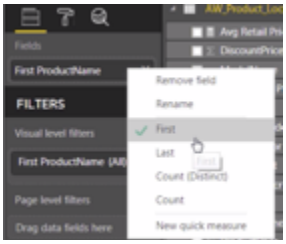
This is a good way to draw attention to very big / important numbers or KPI's – Key Performance Indicators

There are 3 types of cards –
Singular row or regular card
Multi-row card and
KPI cards

Symbols for all 3 Card Visualizations

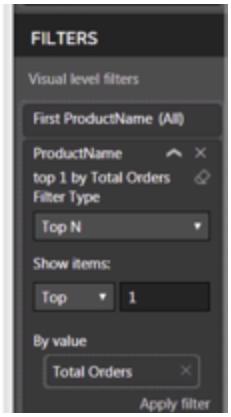


TEXT CARDS Visual

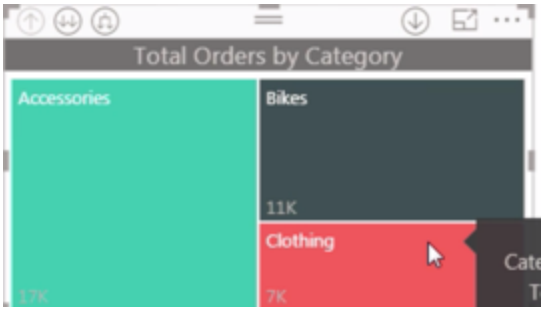


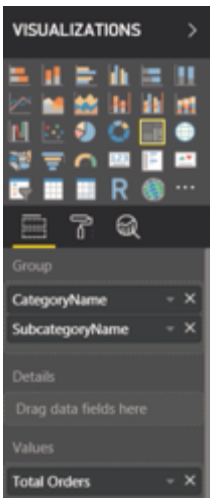
Use Top N Filters to display in Text Card

Top product by Order Quantity

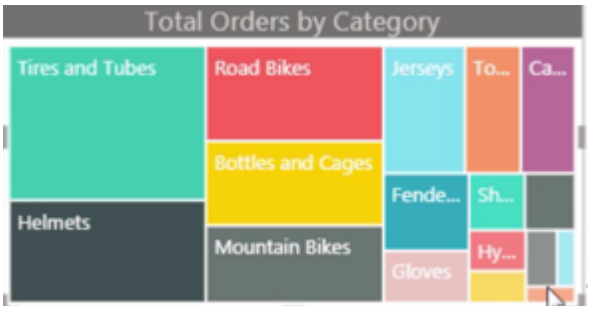


TREEMAP Visual





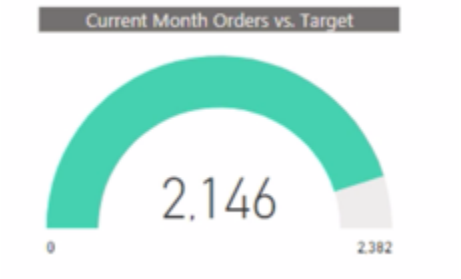
TreeMap is designed to show composition. For eg above how total order volume breaks down into different categories.



LINE and AREA chart Visuals
To show Trends

Can also add trend Line on the chart

Gauge Chart Visual



DATA VISUALIZATION **BEST PRACTICES**

Strive for clarity & simplicity, above all else

- *Aim to maximize impact and minimize noise; it's all about balancing design and function*

Don't just build charts and graphs; create a *narrative*

- *Without context, data is meaningless; use filters, bookmarks, and effective visualizations to translate raw data into powerful insights and implications*

Always ask yourself the three key questions:

1. *What **type of data** are you visualizing? (Integer, categorical, time-series, geo-spatial, etc)*
2. *What are you trying to **communicate**? (Relationships, compositions, trending, etc)*
3. *Who is the **end user** consuming this information? (Analyst, CEO, client, intern, etc)*