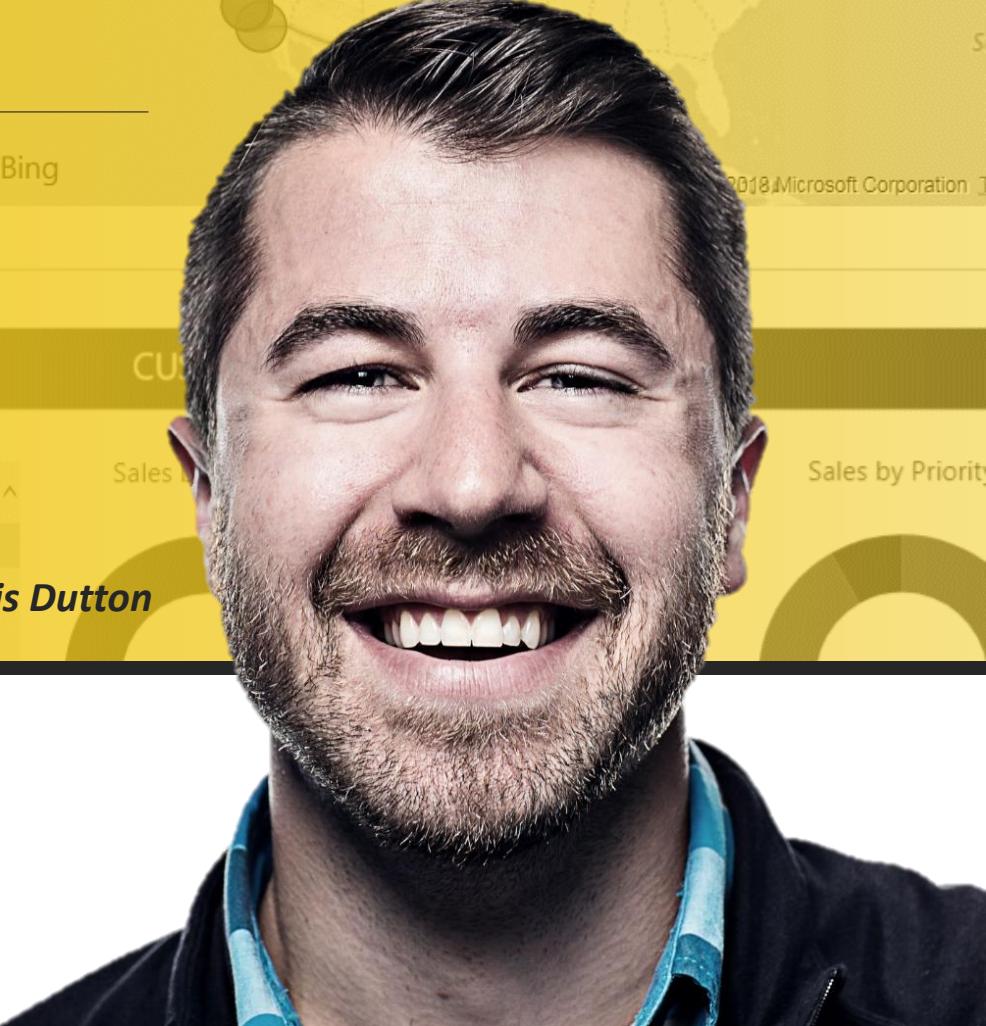


UP & RUNNING WITH POWER BI DESKTOP

★★★★★ *With Best-Selling Instructor **Chris Dutton***



COURSE STRUCTURE

This is a **project-based course**, designed for students looking for a *practical, hands-on,* and *highly engaging* approach to learning business intelligence with Power BI Desktop

Additional course resources include:

-  **Downloadable Power BI Ebook** (100+ pages) to serve as a helpful reference guide when you're offline or on the go (or just need a refresher!)
-  **Course Quizzes** and **Homework Exercises** to test and reinforce key concepts throughout the course, with detailed step-by-step solutions
-  A bonus **Final Project** to test your abilities and apply the skills developed throughout the course to a brand new data set

COURSE OUTLINE

1	Introducing Power BI Desktop	<i>Installing Power BI, exploring the Power BI workflow, comparing Power BI vs. Excel, etc.</i>
2	Connecting & Shaping Data	<i>Connecting to source data, shaping and transforming tables, editing, merging and appending queries, etc.</i>
3	Creating a Data Model	<i>Building relational models, creating table relationships, understanding cardinality, exploring filter flow, etc.</i>
4	Adding Calculated Fields with DAX	<i>Understanding DAX syntax, adding calculated columns and measures, writing common formulas and functions, etc.</i>
5	Visualizing Data with Reports	<i>Inserting charts and visuals, customizing formats, editing interactions, applying filters and bookmarks, etc.</i>
6	Final Course Project	<i>Applying all of the skills developed throughout the course to build a pro-quality B.I. report from a brand new dataset</i>

INTRODUCING THE COURSE PROJECT

THE SITUATION

You've just been hired by **Adventure Works Cycles***, a global manufacturing company, to design and deliver an end-to-end business intelligence solution – *from scratch!*

THE BRIEF

Your client needs a way to **track KPIs (sales, revenue, profit, returns), compare regional performance, analyze product-level trends and forecasts, and identify high-value customers**

All you've been given is a folder of **raw csv files**, containing information about transactions, returns, products, customers and territories

THE OBJECTIVE

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*

SETTING EXPECTATIONS

1 What you see on your screen **may not always match mine**

- *Power BI Desktop features are updated frequently (product updates released each month)*
- **NOTE:** *Power BI is currently only compatible with PC/Windows (not available for Mac)*

2 This course is designed to get you **up & running** with Power BI Desktop

- *The goal is to provide a **foundational understanding** of Power BI desktop; some concepts may be simplified, and we will not cover some of the more advanced tools (i.e. M code, custom R visuals, advanced DAX, etc)*

3 Power BI and Power Pivot in Excel are built on the **exact same engine**

- *If you've taken my **Power Query, Power Pivot & DAX** course, the first sections will review similar core concepts*
- *Feel free to skip ahead if you're already comfortable with Power Query and data modeling fundamentals*

4 We will not cover **Power BI Service** as part of this course

- *This course will focus on **Power BI Desktop** specifically; online sharing and collaboration features (app.powerbi.com) will be covered in depth in a separate course*

INTRODUCING POWER BI

MEET 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

More information at powerbi.microsoft.com



Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms

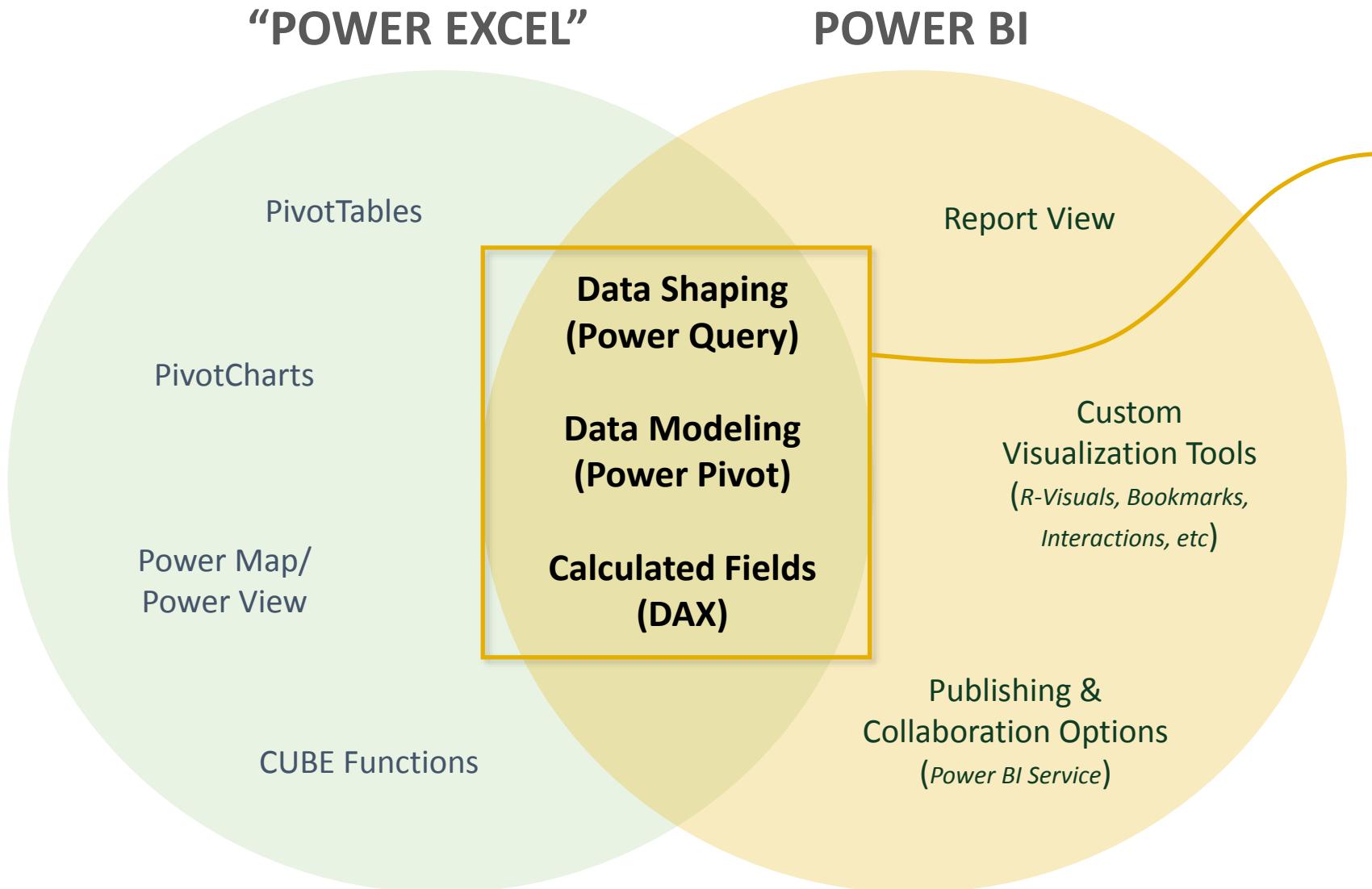


Source: Gartner (February 2018)

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*

POWER BI VS. “POWER EXCEL”

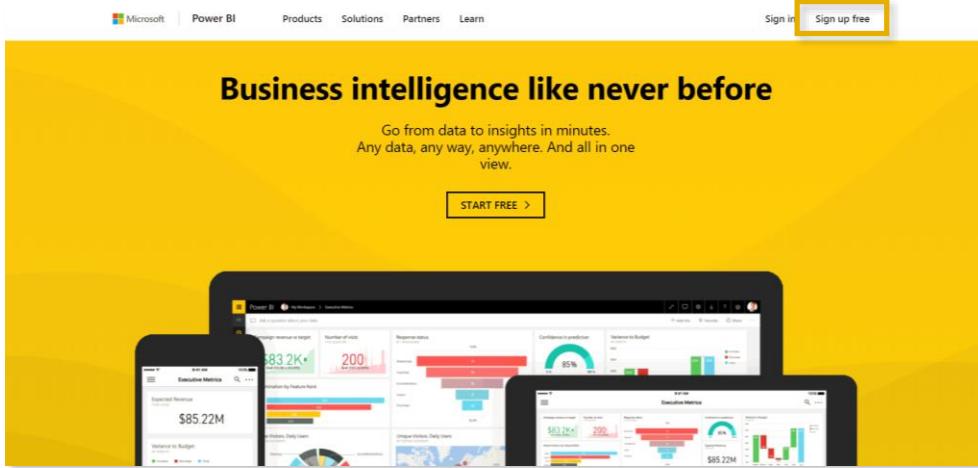


“Power Excel” and Power BI are built on top of the ***exact same engine!***

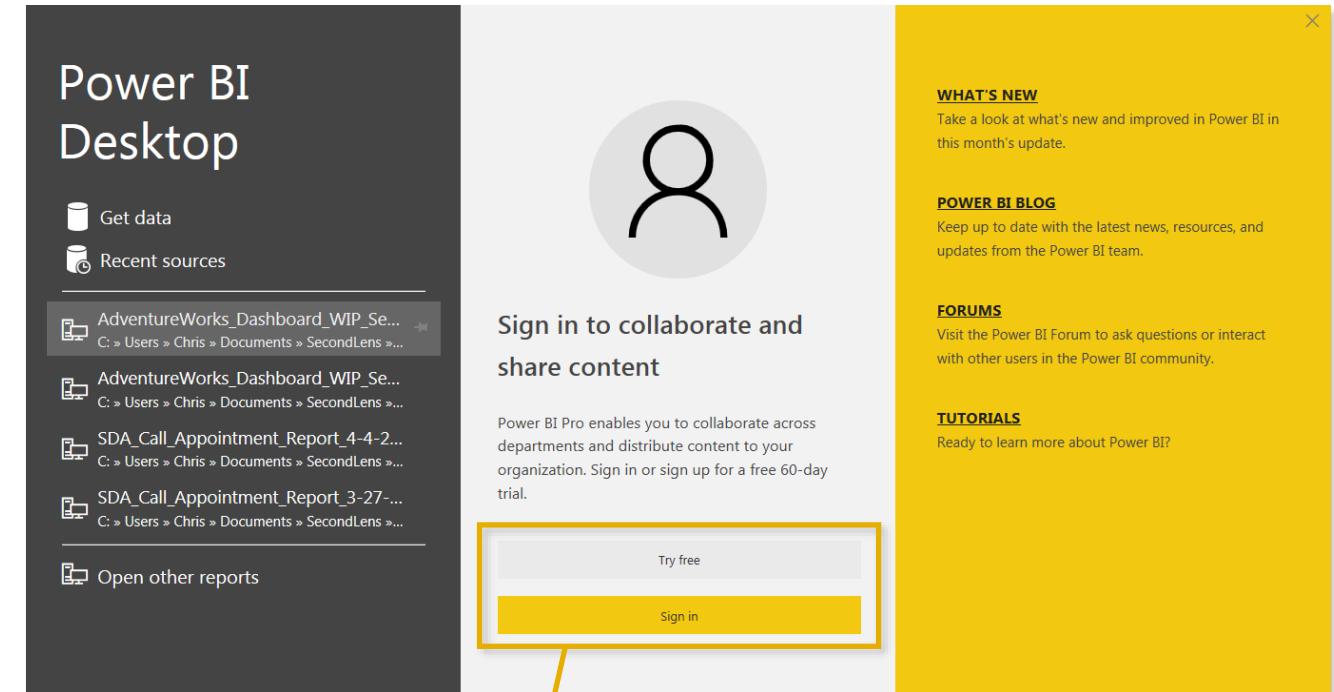
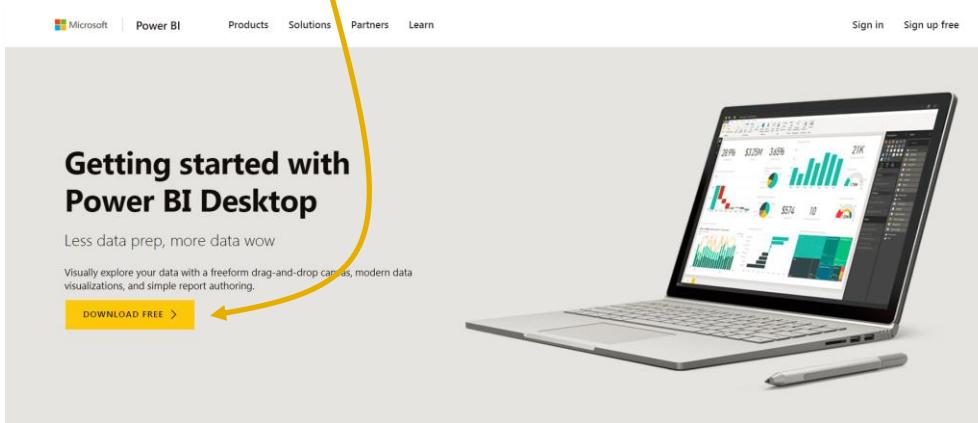
- Power BI takes the same data shaping, modeling and analytics capabilities and adds ***new reporting and publishing tools***
- Transitioning is easy; you can import an ***entire data model*** directly from Excel!

INSTALLING POWER BI DESKTOP

1) Head to powerbi.microsoft.com and click “Sign Up Free”



2) Click “Download Free” to start the Power BI Desktop download

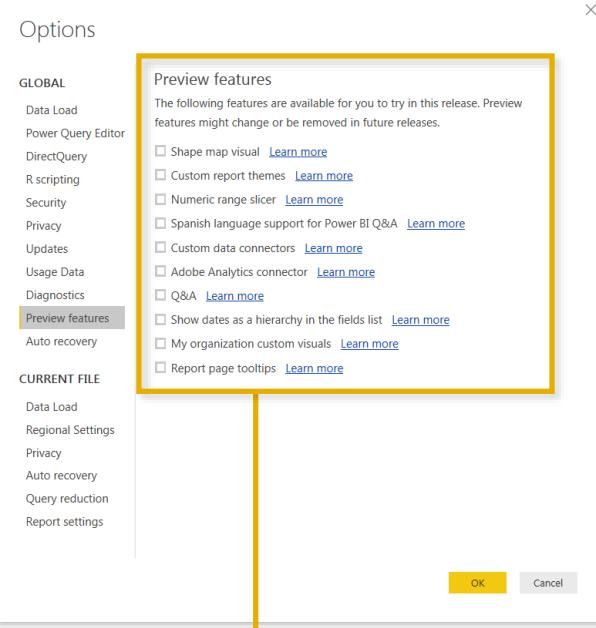


IMPORTANT: You do **not** need to sign in or register for a Power BI Pro account to access Power BI Desktop (*you can simply close this window*)

- Sign-in is only required to access the sharing and collaboration tools available through Power BI Service (app.powerbi.com)
- **Note:** Microsoft requires a **work or school e-mail address**

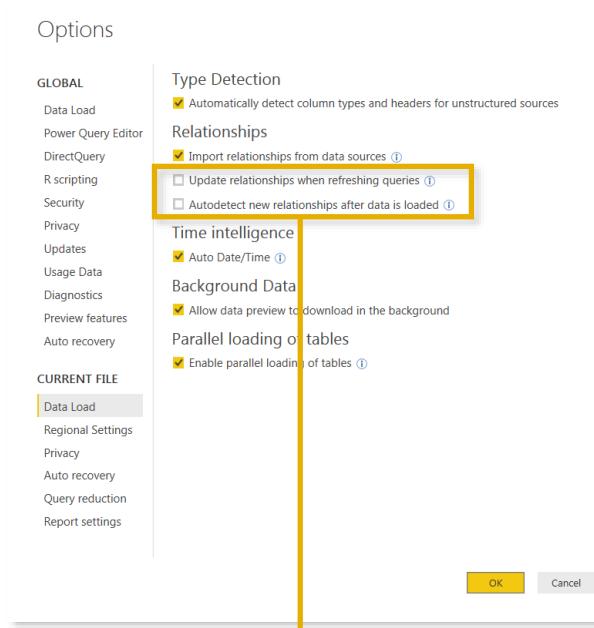
COURSE OPTIONS & SETTINGS

PREVIEW FEATURES



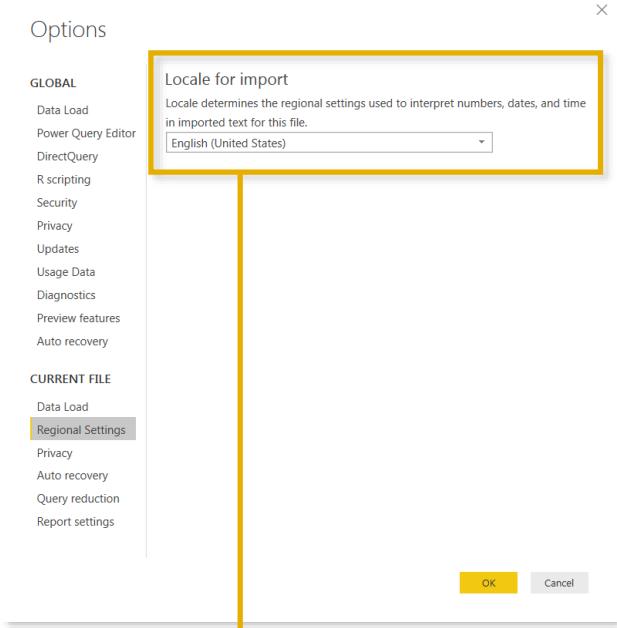
1) In the “Preview Features” tab, deselect any active features while you are taking the course

DATA LOAD



2) In the “Data Load” tab, deselect the “Update relationships” and “Autodetect new relationships after data is loaded” options

REGIONAL SETTINGS



3) In the “Regional Settings” tab, make sure to use the “English (United States)” locale for import

THE POWER BI INTERFACE

Three Core Views:

Report



Data



Relationships



The screenshot displays the Microsoft Power BI desktop application. On the left, a vertical ribbon menu is visible with icons for Report, Data, and Relationships. A large curly brace groups the 'Report' and 'Data' sections. The main workspace shows a report for 'ADVENTURE WORKS cycles' with various visualizations and data tables. At the bottom, tabs for 'Exec Summary', 'Product Detail', and 'Customer Detail' are present, along with a '+' button. The top navigation bar includes Home, View, Modeling, Help, and various ribbon tabs like Paste, Get Data, Insert, etc.

Report View Components:

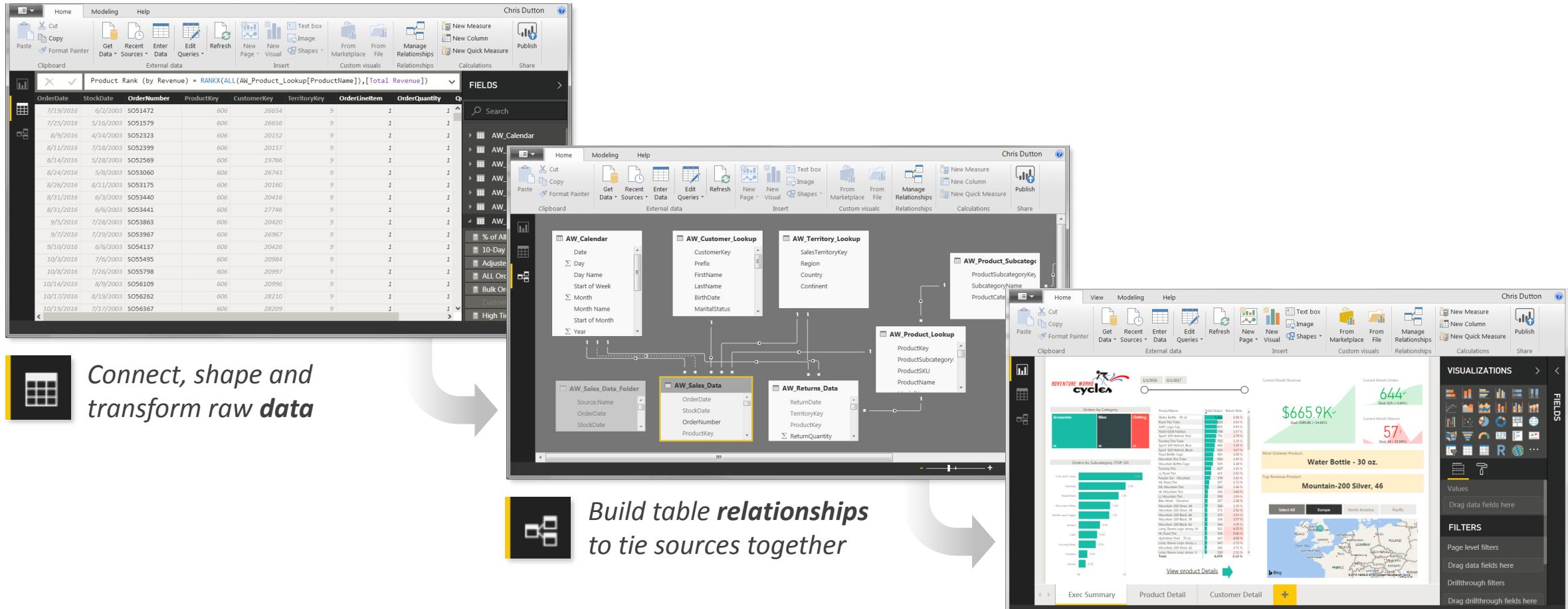
- Top Navigation:** Home, View, Modeling, Help, Clipboard, External data, Insert, Custom visuals, Relationships, Calculations, Share.
- Report Header:** ADVENTURE WORKS cycles, Date range (1/1/2016 - 6/1/2017).
- Visualizations:**
 - Current Month Revenue: \$665.9K (Goal: \$580.8K +14.66%).
 - Current Month Orders: 644 (Goal: 625 +3.04%).
 - Current Month Returns: 57 (Goal: 43 -32.56%).
 - Most Ordered Product: Water Bottle - 30 oz.
 - Top Revenue Product: Mountain-200 Silver, 46.
- Data Tables:**
 - Orders by Category:

Category	Value
Accessories	5K
Bikes	4K
Clothing	2K
 - Orders by Subcategory (TOP 10):

Subcategory	Value
Tires and Tubes	2.8K
Helmets	2.0K
Road Bikes	1.7K
Mountain Bikes	1.4K
Bottles and Cages	1.3K
Jerseys	0.9K
Caps	0.8K
Touring Bikes	0.7K
Fenders	0.4K
Gloves	0.3K
 - Product Name, Total Orders, Return Rate table:

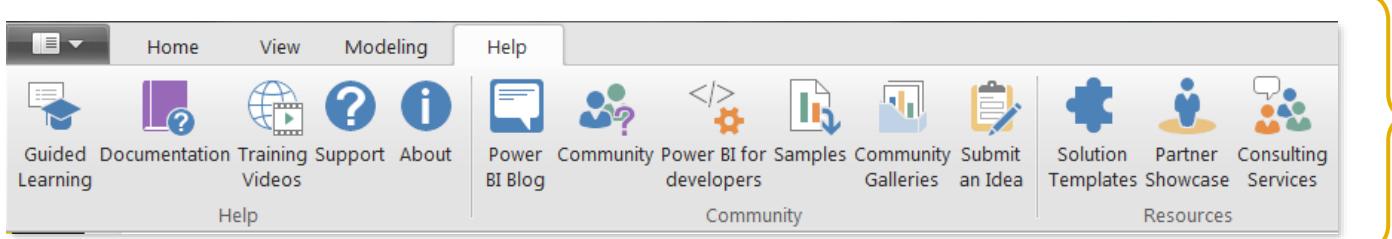
ProductName	Total Orders	Return Rate
Water Bottle - 30 oz.	1,164	1.96 %
Road Tire Tube	829	1.63 %
AWC Logo Cap	603	0.93 %
Patch Kit/8 Patches	798	1.57 %
Sport-100 Helmet, Red	753	2.79 %
Touring Tire Tube	702	1.35 %
Sport-100 Helmet, Blue	666	3.15 %
Sport-100 Helmet, Black	626	3.67 %
Road Bottle Cage	560	1.58 %
Mountain Tire Tube	554	1.95 %
Mountain Bottle Cage	539	1.38 %
Touring Tire	427	1.16 %
LL Road Tire	421	2.02 %
Fender Set - Mountain	378	1.82 %
ML Road Tire	297	1.72 %
ML Mountain Tire	266	1.94 %
HL Mountain Tire	206	3.40 %
LL Mountain Tire	195	2.09 %
Bike Wash - Discover	187	2.38 %
Mountain-200 Silver, 46	184	1.09 %
Mountain-200 Silver, 38	171	2.92 %
Mountain-200 Black, 46	170	3.53 %
Mountain-200 Black, 38	168	3.57 %
Mountain-200 Black, 42	164	3.05 %
Long-Sleeve Logo Jersey, M	161	4.35 %
HL Road Tire	158	5.06 %
Hydration Pack - 70 oz.	147	4.08 %
Long-Sleeve Logo Jersey, L	147	2.72 %
Mountain-200 Silver, 42	141	0.71 %
Long-Sleeve Logo Jersey, S	130	2.31 %
Total	6,670	2.13 %
- Map:** A Bing map of Europe and North America showing various cities and regions.
- Bottom Navigation:** PAGE 1 OF 3, Exec Summary, Product Detail, Customer Detail, +.

THE POWER BI WORKFLOW



Design interactive **reports** to explore and visualize data

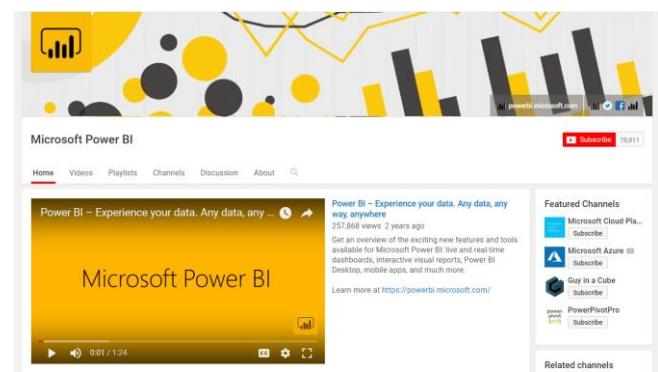
HELPFUL RESOURCES



The “**Help**” tab includes documentation, training videos, sample files, templates, and links to support blogs and communities – all within Power BI Desktop



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



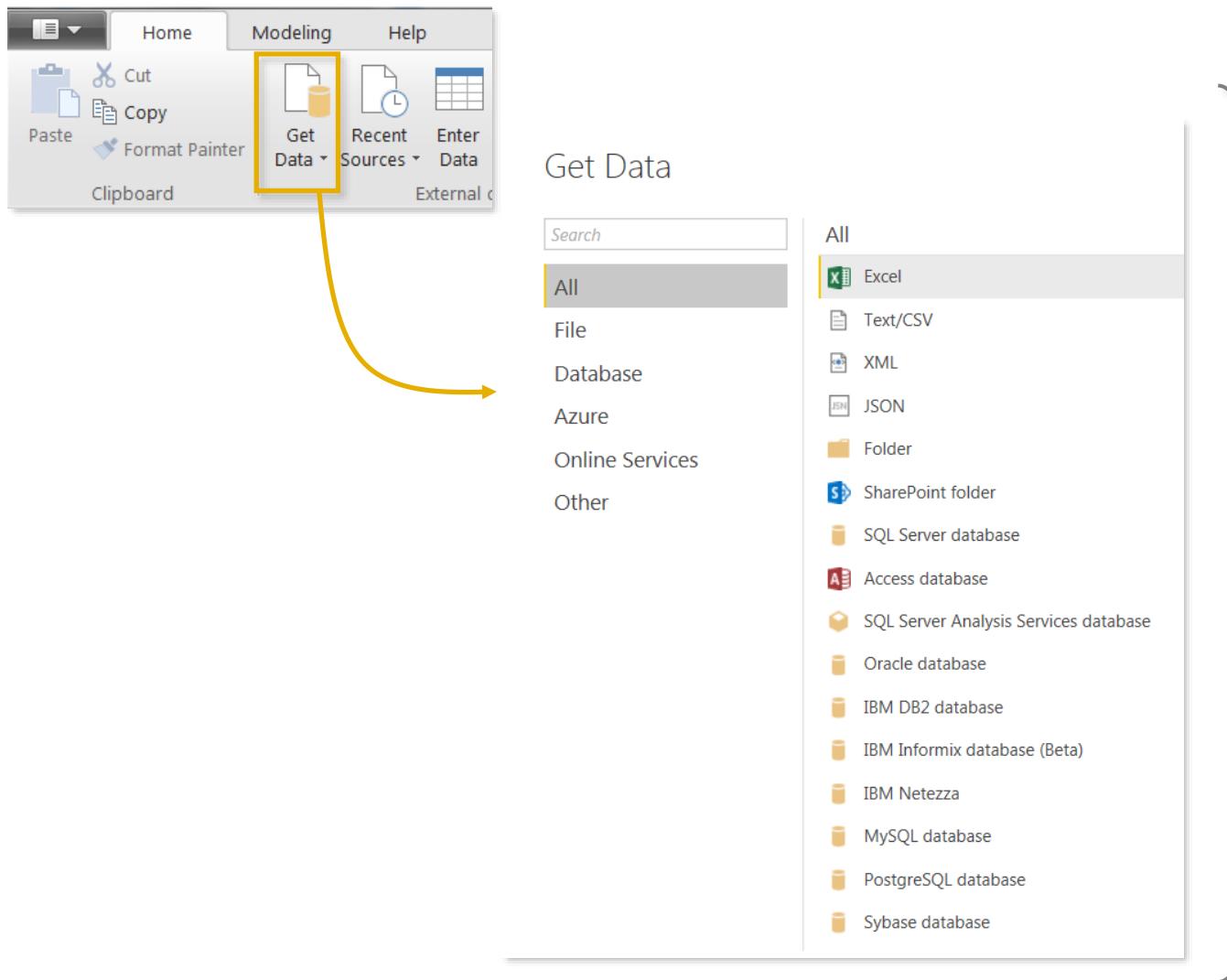
The **Microsoft Power BI YouTube Channel** publishes demos, feature summaries, and advanced tutorials (check out “**Guy in a Cube**” too!)



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

CONNECTING & SHAPING DATA

TYPES OF DATA CONNECTORS



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*)

THE QUERY EDITOR

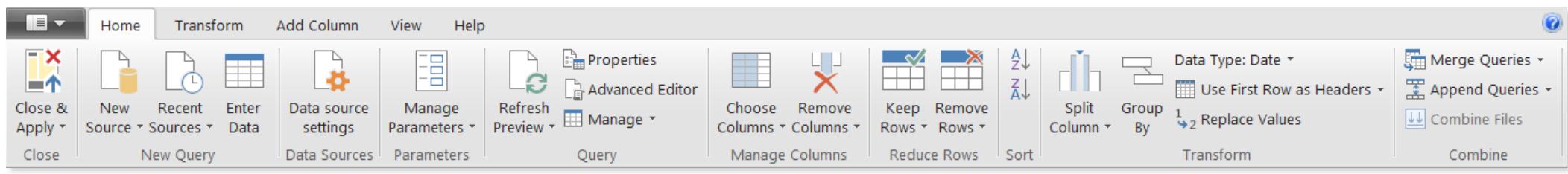
The screenshot shows the Microsoft Power Query Editor interface with several key components highlighted:

- Formula Bar (this is "M" code)**: Located at the top left, showing the formula `= Table.RemoveColumns("#Changed Type2", {"OrderID"})`.
- Query List**: A list of queries on the left side, including "Transform File from AW_Sales_Data...", "Sample Query [2]", and "Other Queries [9]" (which includes "AW_Sales_Data").
- Table**: The main workspace displaying a table with columns: OrderDate, StockDate, OrderNumber, ProductKey, CustomerKey. The table has 21 rows of data.
- Query Editing Tools**: A toolbar at the top with various icons for data transformations like Remove Columns, Keep Rows, Sort, and Transform.
- QUERY SETTINGS**: A panel on the right showing:
 - PROPERTIES**: Name: AW_Sales_Data, All Properties
 - APPLIED STEPS**: A list of steps including Source, Promoted Headers, Changed Type, Added Index, Reordered Columns, Renamed Columns, Changed Type1, Added Conditional Column, Filtered Rows, Changed Type2, and Removed Columns.

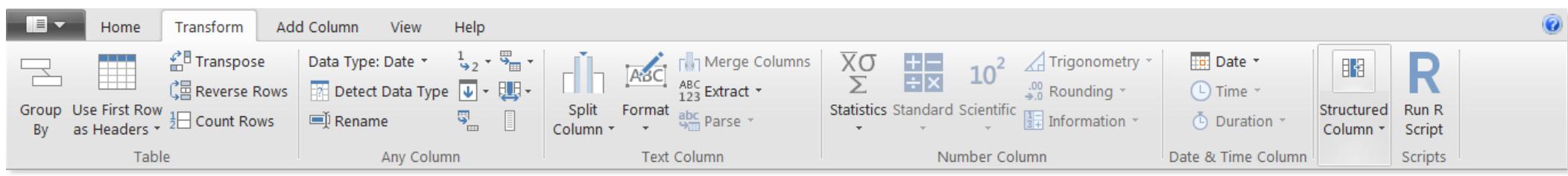
At the bottom of the editor, it says "PREVIEW DOWNLOADED AT 1:27 PM".

QUERY EDITING TOOLS

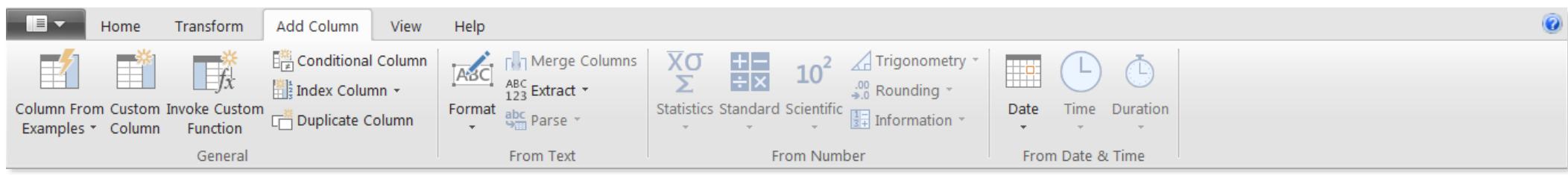
The **HOME** tab includes **general settings** and **common table transformation tools**



The **TRANSFORM** tab includes tools to **modify existing columns** (splitting/grouping, transposing, extracting text, etc)



The **ADD COLUMN** tools **create new columns** (based on conditional rules, text operations, calculations, dates, etc)



BASIC TABLE TRANSFORMATIONS

The screenshot shows the Microsoft Power BI desktop application's ribbon and context menus for transforming data.

Sort values (A-Z, Low-High, etc.)

Change data type (date, \$, %, text, etc.)

Promote header row

Choose or remove columns

Tip: use the “Remove Other Columns” option if you always want a specific set

Keep or remove rows

Tip: use the “Remove Duplicates” option to create a new lookup table from scratch

Duplicate, move & rename columns

Tip: Right-click the column header to access common tools

Column 1	Column 2	Column 3
1/5/2015	9/19/2001	SO45101
1/5/2015	11/21/2001	SO45100

TEXT-SPECIFIC TOOLS

The screenshot shows the Power BI ribbon with the "Transform" tab selected. A context menu is open over a "Text Column". The menu items are:

- Split Column (highlighted with a yellow box)
- Format
- Merge Columns
- Extract
- Parse

Below the "Split Column" item, a secondary menu is displayed with two options:

- By Delimiter
- By Number of Characters

To the right of the "Text Column" menu, a vertical list of options is shown:

- Length
- First Characters
- Last Characters
- Range
- Text Before Delimiter
- Text After Delimiter
- Text Between Delimiters

Below the "Text Column" menu, another vertical list of options is shown:

- lowercase
- UPPERCASE
- Capitalize Each Word
- Trim
- Clean
- Add Prefix
- Add Suffix

HEY THIS IS IMPORTANT!

You can access many of these tools in both the “Transform” and “Add Column” menus -- the difference is whether you want to ***add a new column*** or ***modify an existing one***

Split a text column based on either a specific delimiter or a number of characters

Extract characters from a text column based on fixed lengths, first/last, ranges or delimiters

Format a text column to upper, lower or proper case, or add a prefix or suffix

Tip: Select two or more columns to merge (or concatenate) fields

Tip: Use “Trim” to eliminate leading & trailing spaces, or “Clean” to remove non-printable characters

NUMBER-SPECIFIC TOOLS

The screenshot shows the Power BI Transform ribbon with several groups of tools highlighted by yellow arrows:

- Statistics**: Sum, Minimum, Maximum, Median, Average, Standard Deviation, Count Values, Count Distinct Values.
- Standard**: Add, Multiply, Subtract, Divide, Integer-Divide, Modulo, Percentage, Percent Of.
- Scientific**: Absolute Value, Power, Square Root, Exponent, Logarithm, Factorial.
- Trigonometry**: Sine, Cosine, Tangent, Arcsine, Arccosine, Arctangent.
- Number Column**: Is Even, Is Odd, Sign.

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

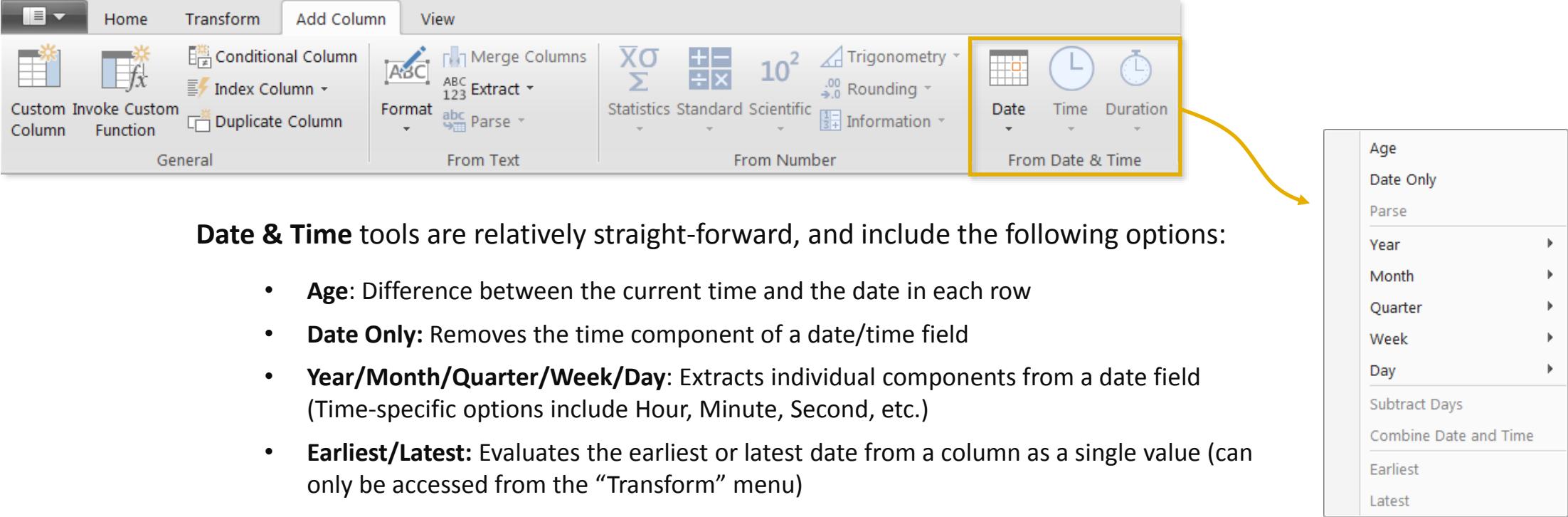
Note: These tools return a *SINGLE* value, and are commonly used to explore a table rather than prepare it for loading

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



The screenshot shows the Power BI ribbon with the 'Transform' tab selected. In the 'From Date & Time' section of the ribbon, there is a group of three icons: 'Date', 'Time', and 'Duration'. This group is highlighted with a yellow box, and a yellow arrow points from it to a detailed list of options on the right side of the screen.

Age
Date Only
Parse
Year
Month
Quarter
Week
Day
Subtract Days
Combine Date and Time
Earliest
Latest

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

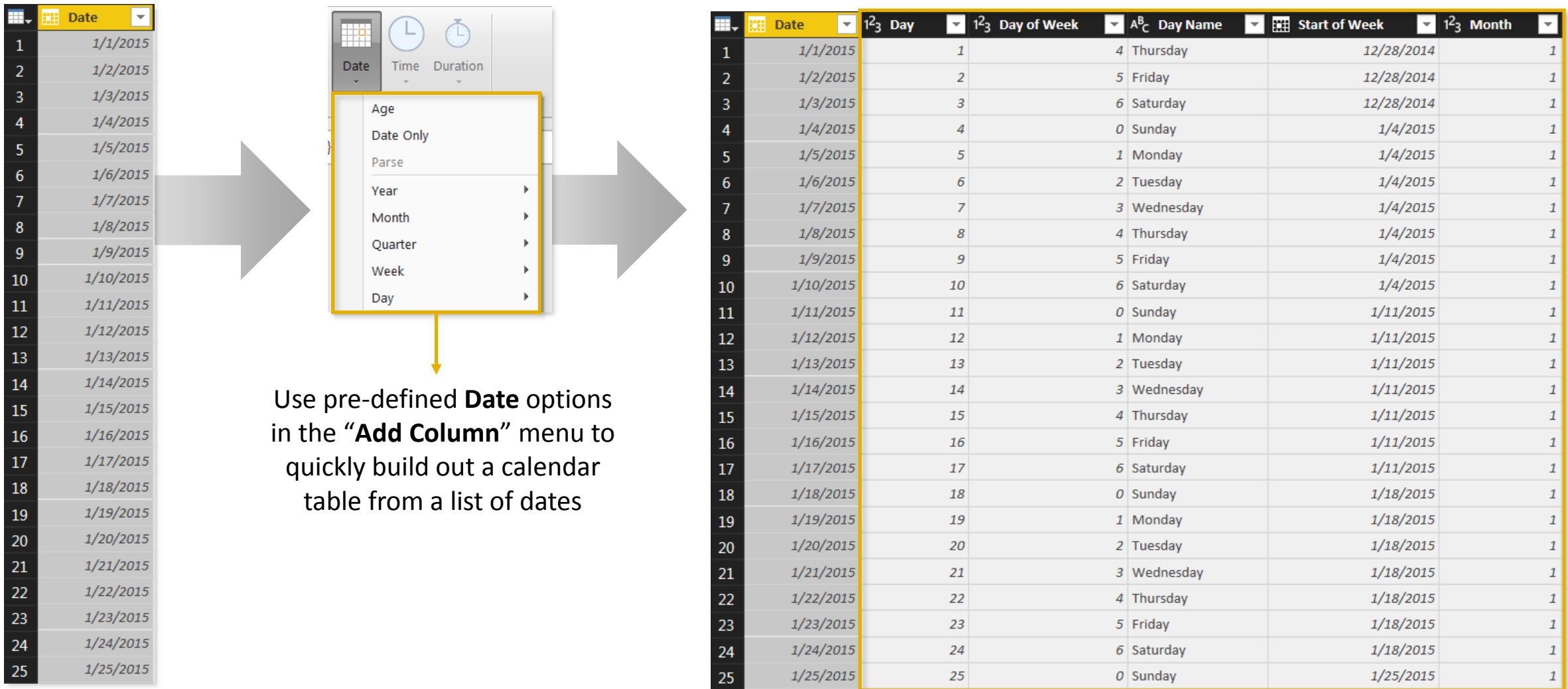
- **Age:** Difference between the current time and the date in each row
- **Date Only:** Removes the time component of a date/time field
- **Year/Month/Quarter/Week/Day:** Extracts individual components from a date field
(Time-specific options include Hour, Minute, Second, etc.)
- **Earliest/Latest:** 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

PRO TIP:

Load up a table containing a **single date column** and use Date tools to build out an **entire calendar table**

CREATING A BASIC CALENDAR TABLE

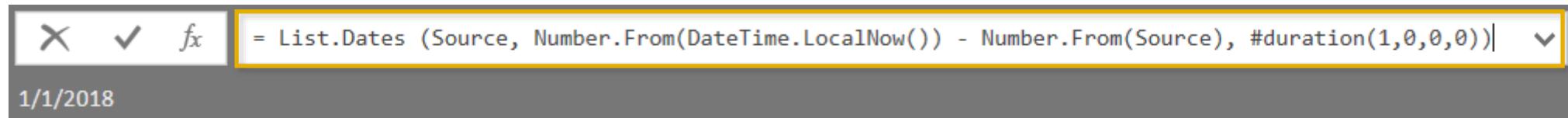


PRO TIP: CREATING A ROLLING CALENDAR

- 1) Create a new, blank query (**Get Data > Blank Query or New Source > Blank Query**)
- 2) In the formula bar, generate a starting date by entering a “literal” (in YYYY, MM, DD format):

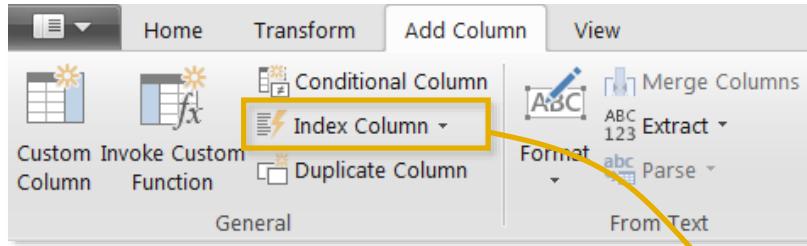


- 3) Click the fx icon to add a new custom step, and enter the following formula exactly as shown:



- 4) Convert the resulting list into a Table (**List Tools > To Table**) and format the column as a Date
- 5) Add calculated Date columns (Year, Month, Week, etc.) as necessary using the **Add Column** tools

ADDING INDEX COLUMNS

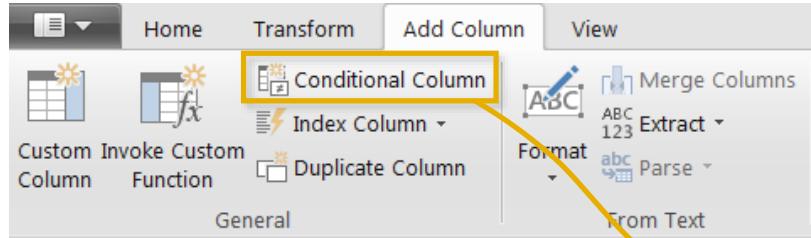


Index Columns contain a list of sequential values that can be used to identify each unique row in a table (*typically starting from 0 or 1*)

These columns are often used to create **unique IDs** that can be used to form relationships between tables (*more on that later!*)

Index	OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey
1	1/1/2015	9/21/2001	SO45080	332	14657
2	1/1/2015	12/5/2001	SO45079	312	29255
3	1/1/2015	10/29/2001	SO45082	350	11455
4	1/1/2015	11/16/2001	SO45081	338	26782
5	1/2/2015	12/15/2001	SO45083	312	14947
6	1/2/2015	10/12/2001	SO45084	310	29143
7	1/2/2015	12/18/2001	SO45086	314	18747
8	1/2/2015	10/9/2001	SO45085	312	18746
9	1/3/2015	10/3/2001	SO45093	312	18906
10	1/3/2015	9/29/2001	SO45090	310	29170
11	1/3/2015	12/11/2001	SO45088	345	11398
12	1/3/2015	10/24/2001	SO45092	313	18899
13	1/3/2015	12/16/2001	SO45089	351	25977
14	1/3/2015	10/26/2001	SO45091	314	18909
15	1/3/2015	9/11/2001	SO45087	350	11388
16	1/3/2015	9/11/2001	SO45094	310	22785
17	1/4/2015	10/30/2001	SO45096	312	12483
18	1/4/2015	10/30/2001	SO45097	313	29151

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"*

Conditional Columns allow you to define new fields based on logical rules and conditions (*IF/THEN statements*)

Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name

QuantityType

	Column Name	Operator	Value	Output
If	OrderQuantity	equals	ABC 123	1
Else If	OrderQuantity	is greater than	ABC 123	1

Add rule

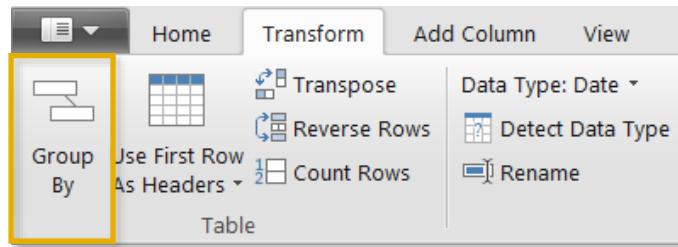
Otherwise

ABC 123

Other

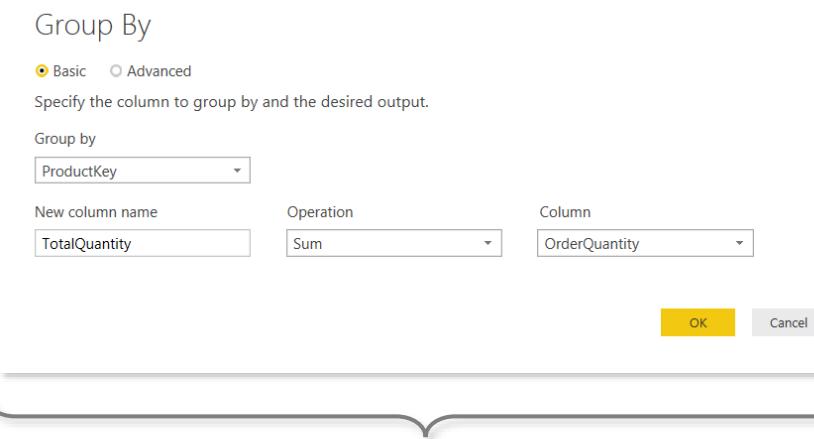
OK Cancel

GROUPING & AGGREGATING DATA



Group By allows you to aggregate your data at a different level
(i.e. transform daily data into monthly, roll up transaction-level data by store, etc)

	OrderDate	ProductKey	CustomerKey	OrderQuantity
1	6/25/2017	214	14719	1
2	7/16/2016	214	11243	1
3	12/31/2016	214	21452	1
4	6/29/2017	214	22748	1
5	10/6/2016	214	25025	1
6	10/7/2016	214	16504	1
7	10/13/2016	214	13043	1
8	1/19/2017	214	23101	1
9	9/7/2016	214	24900	1
10	1/19/2017	214	24196	1
11	6/29/2017	214	12963	1
12	11/6/2016	214	14570	1
13	11/13/2016	214	16999	1
14	7/31/2016	214	12281	1
15	10/9/2016	214	15685	1
16	8/1/2016	214	16982	1
17	12/4/2016	214	12835	1

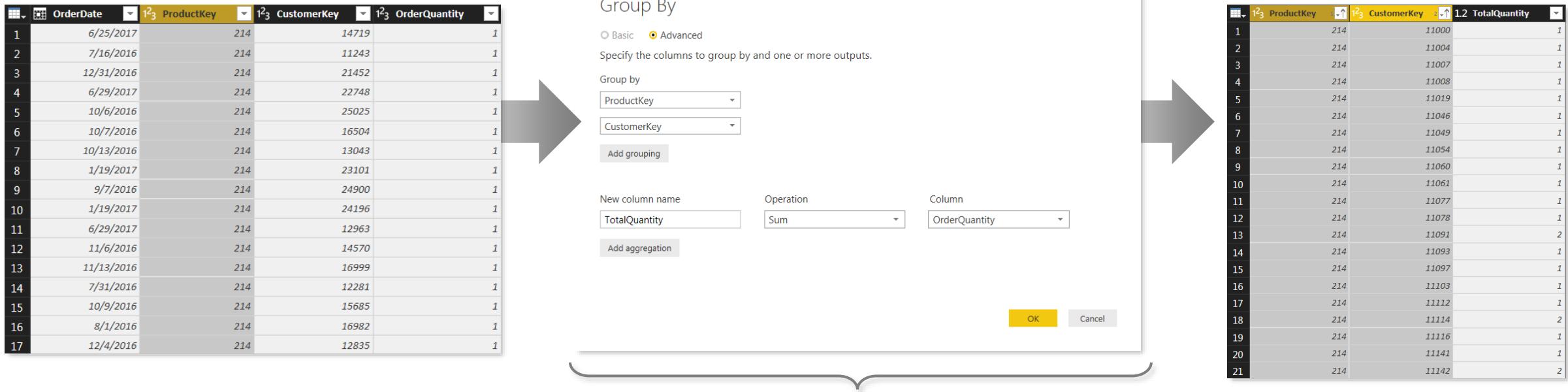


	ProductKey	TotalQuantity
1	214	2099
2	217	1940
3	222	1995
4	225	4151
5	228	392
6	231	408
7	234	424
8	237	381
9	310	169
10	311	139
11	312	179
12	313	168
13	314	157
14	320	10
15	321	55
16	322	5
17	323	34

In this case we're transforming a daily, transaction-level table into a summary of "TotalQuantity" rolled up by "ProductKey"

NOTE: Any fields not specified in the Group By settings are lost

GROUPING & AGGREGATING DATA (ADVANCED)

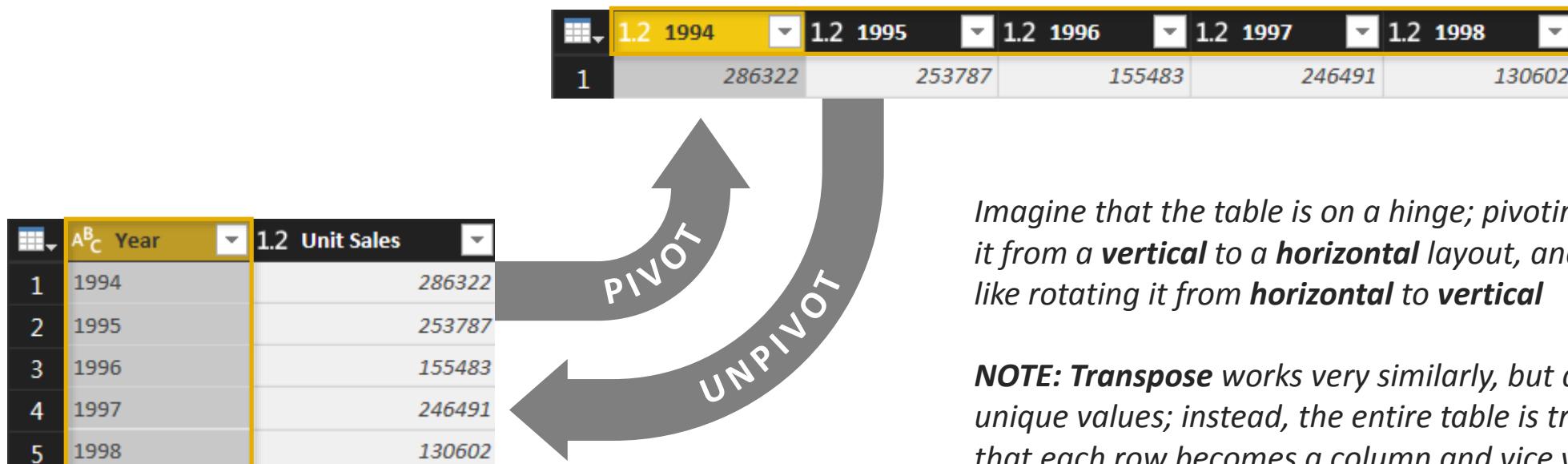


This time we're transforming the daily, transaction-level table into a summary of "**TotalQuantity**" aggregated by both "**ProductKey**" and "**CustomerKey**" (using the advanced option in the dialog box)

NOTE: This is similar to creating a PivotTable in Excel and pulling in "**Sum of OrderQuantity**" with **ProductKey** and **CustomerKey** as row labels

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

MERGING QUERIES

The screenshot shows the 'Merge Queries' dialog box from Microsoft Power Query. On the left, a vertical menu lists 'Merge Queries' (highlighted with a yellow box), 'Append Queries', 'Combine Files', and 'Combine'. An arrow points from the 'Merge' option in the menu to the 'Merge' button in the dialog. The main area contains two tables: 'AW_Sales_Data' and 'AW_Product_Lookup'. The 'AW_Sales_Data' table has columns: OrderDate, ProductKey, CustomerKey, OrderQuantity, StockDate, OrderNumber, TerritoryKey, and Order. The 'AW_Product_Lookup' table has columns: ProductKey, ProductSubcategoryKey, ProductSKU, ProductName, ModelName, and ProductDescription. Both tables have a row selected with the value '214'. Below the tables, a 'Join Kind' dropdown is set to 'Left Outer (all from first, matching from second)'. A note at the bottom says 'The selection has matched 56046 out of the first 56046 rows.' At the bottom right are 'OK' and 'Cancel' buttons.

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

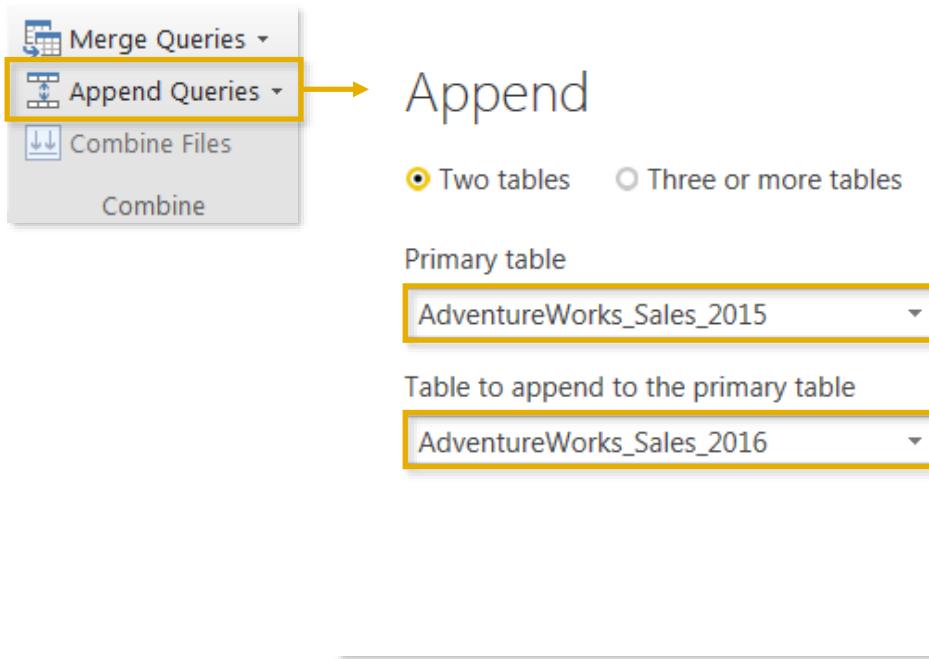
NOTE: Merging **adds columns** to an existing table

HEY THIS IS IMPORTANT!

Just because you **can** merge tables, doesn't mean you **should**.

In general, it's better to keep tables separate and define **relationships** between them (*more on that later!*)

APPENDING QUERIES



Append

Two tables Three or more tables

Primary table
AdventureWorks_Sales_2015

Table to append to the primary table
AdventureWorks_Sales_2016

Appending queries allows you to **combine** (or **stack**) tables that share the exact same column structure and data types

In this case we're appending the **AdventureWorks_Sales_2015** table to the **AdventureWorks_Sales_2016** table, which is valid since they share identical table structures

NOTE: Appending **adds rows** to an existing table



PRO TIP:

Use the “**Folder**” option (*Get Data > More > Folder*) to append all files within a folder (assuming they share the same structure); as you add new files, simply refresh the query and they will automatically append!

DATA SOURCE SETTINGS

The screenshot shows the Power BI Desktop interface with the 'Data Source Settings' dialog box open. The dialog box has a ribbon bar at the top with tabs: Home, Transform, Add Column, View, Help, Close & Apply, New Source, Recent Sources, Enter Data, Data source settings (which is selected and highlighted with a yellow box), Manage Parameters, and Data Sources. Below the ribbon is a toolbar with Close, New Query, and a 'Data source settings' button. A search bar labeled 'Search data source settings' is followed by a list of local file connections. The 'c:\users\chris\documents\secon...ks\adventureworks_calendar.csv' connection is currently selected and highlighted with a yellow box. At the bottom of the dialog are buttons for 'Change Source...', 'Edit Permissions...', 'Clear Permissions...', and 'Close'. A yellow arrow points from the 'Data source settings' tab in the ribbon to the 'Data source settings' button in the toolbar, and another yellow arrow points from the 'Change Source...' button in the dialog to the 'Change Source...' button in the toolbar.

Data source settings

Manage settings for data sources that you have connected to using Power BI Desktop.

Data sources in current file Global permissions

Search data source settings

- c:\users\chris\documents\secon...ks\adventureworks_calendar.csv
- c:\users\chris\documents\secon...s\adventureworks_customers.csv
- c:\users\chris\documents\secon...reworks_product_categories.csv
- c:\users\chris\documents\secon...orks_product_subcategories.csv
- c:\users\chris\documents\secon...ks\adventureworks_products.csv
- c:\users\chris\documents\secon...rks\adventureworks_returns.csv
- c:\users\chris\documents\secon...works\adventureworks_sales.csv
- c:\users\chris\documents\secon...adventureworks_territories.csv
- c:\users\chris\documents\secon...i\data\adventureworks\aw_sales

Change Source... Edit Permissions... Clear Permissions... Close

Comma-Separated Values

Basic Advanced

File path: C:\Users\Chris\Desktop\Power BI Course Files\Adventure Works\Adventure [Browse...](#)

Open file as: Csv Document

File origin: 1252: Western European (Windows)

Line breaks: Apply all line breaks

Delimiter: Comma

The **Data Source Settings** in the Query Editor allow you to manage data connections and permissions

HEY THIS IS IMPORTANT!

Connections to local files reference the *exact* path
If the file name or location changes, **you will need to change the source and browse to the current version**

MODIFYING QUERIES

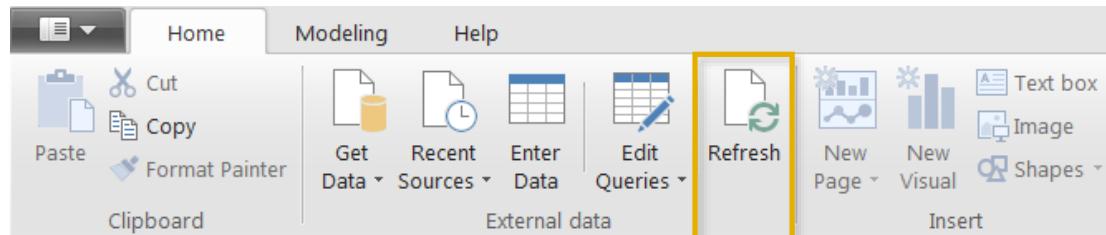
Select “Edit Queries” from the **Home** tab to launch the Query Editor

Within the editor, view or modify existing queries in the “**Queries**” pane

Within each query, you can click each item within the “**Applied Steps**” pane to view each stage of the transformation, add new steps or delete existing ones, or modify individual steps by clicking the gear icons

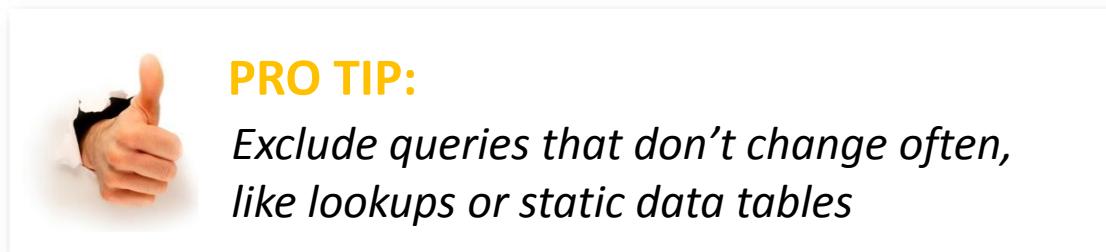
The screenshot shows the Microsoft Power Query Editor interface. The Home tab is selected, with the 'Edit Queries' button highlighted. The Queries pane on the left shows a list of 13 queries, with 'AW_Customer_Lookup' selected. The main area displays a preview of the data with columns: CustomerKey, Prefix, FirstName, LastName, and Full Name. The Query Settings pane on the right shows properties for the selected query, including its name ('AW_Customer_Lookup') and a list of applied steps. The Applied Steps pane lists various transformations such as 'Capitalized Each Word', 'Inserted Merged Column', and 'Reordered Columns'. Arrows point from the text instructions to the corresponding UI elements.

REFRESHING QUERIES



By default, **ALL** queries in the model will refresh when you use the “*Refresh*” command from the **Home** tab

From the Query Editor, uncheck “***Include in report refresh***” to exclude individual queries from the refresh



A screenshot of the Power BI 'Queries [13]' pane. A context menu is open over the 'AW_Territory_Lookup' query, specifically over the 'Include in report refresh' option, which is highlighted with a yellow box and a yellow arrow pointing from the text above. The menu also includes options like 'Enable load', 'Delete', 'Rename', 'Copy', and 'Paste'. To the right of the pane, a preview of a table is shown with 10 rows of data.

	SalesTerritoryKey	Region	Country
1	Northwest	United States	
2	Northeast	United States	
3	Central	United States	
4	Southwest	United States	
5	Southeast	United States	
6	Canada	Canada	
7	France	France	
8	Germany	Germany	
9	Australia	Australia	
10	United Kingdom	United Kingdom	

DEFINING DATA CATEGORIES

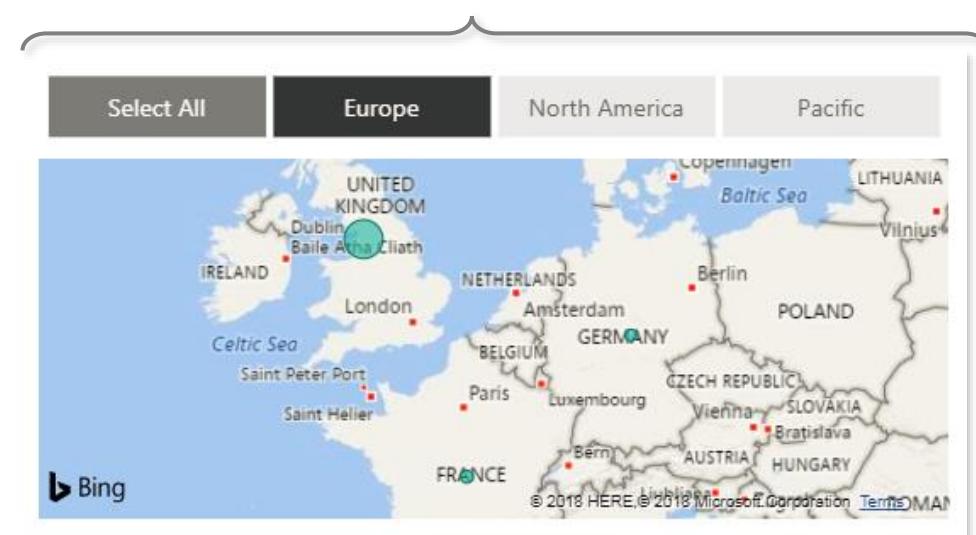
The screenshot shows the Power BI Data view interface. The top navigation bar includes Home, Modeling, and Help tabs. Below the navigation bar are several icons: Manage Relationships, New Measure, New Column, New Table, New Parameter, What If, Sort by Column, and Sort. The 'Modeling' tab is currently selected. On the left, there's a ribbon with icons for Sales Territory, Region, Country, and Continent. The main area displays a table with the following data:

SalesTerritoryKey	Region	Country	Continent
1	Northwest	United States	North America
2	Northeast	United States	North America
3	Central	United States	North America
4	Southwest	United States	North America
5	Southeast	United States	North America
6	Canada	Canada	North America
7	France	France	Europe
8	Germany	Germany	Europe
9	Australia	Australia	Pacific
10	United Kingdom	United Kingdom	Europe

A context menu is open over the 'Country' column header. The menu title is 'Data Category: Country/Region'. The menu items include: Uncategorized, Address, City, Continent, and Country/Region (which is checked). The entire context menu is highlighted with a yellow box.

From the “**Modeling**” tab in the **Data** view, you can edit field properties to define specific categories

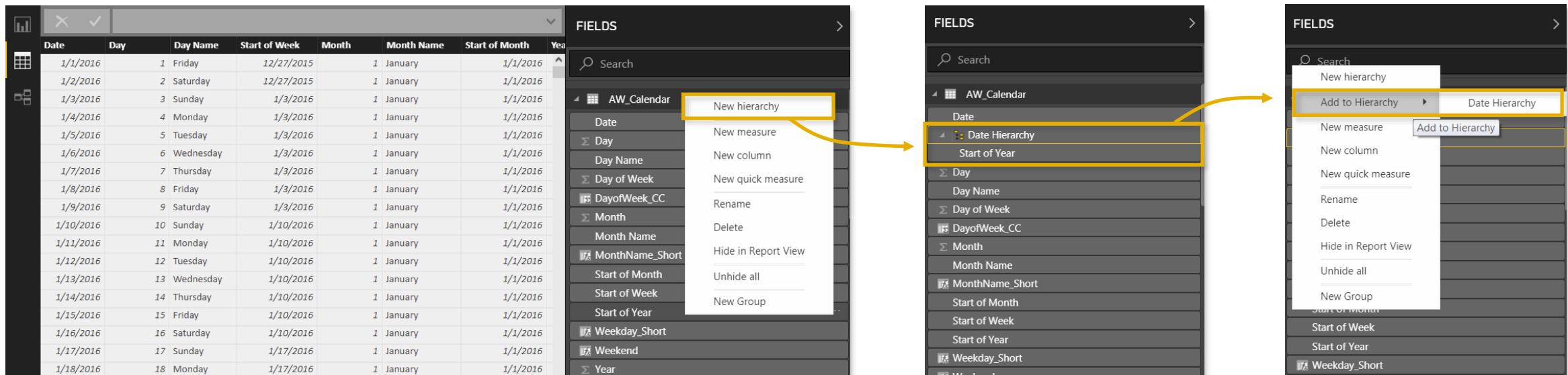
*This is commonly used to help Power BI accurately map location-based fields like **addresses**, **countries**, **cities**, **latitude/longitude coordinates**, **zip codes**, etc*



DEFINING HIERARCHIES

Hierarchies are groups of nested columns that reflect multiple levels of granularity

- For example, a “**Geography**” hierarchy might include **Country**, **State**, and **City** columns
- Each hierarchy can be treated as a **single item** in tables and reports, allowing users to “drill up” and “drill down” through different levels of the hierarchy in a meaningful way

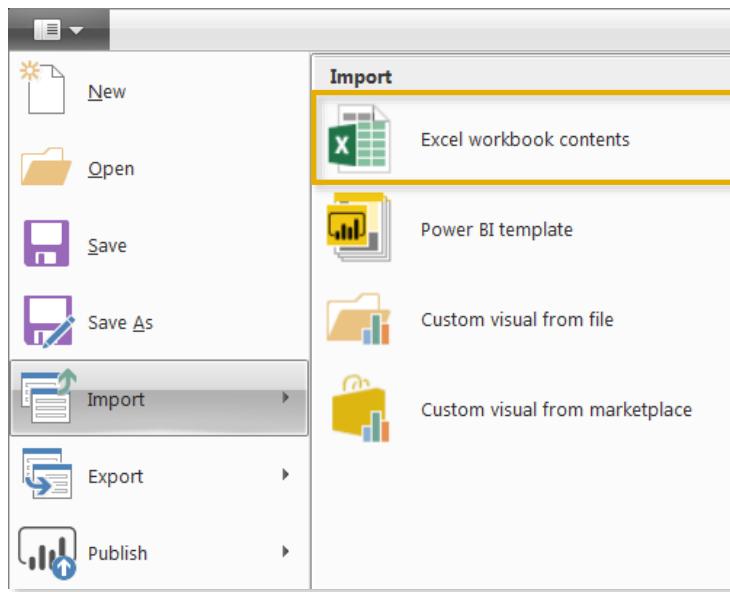


1) From within the **Data** view, right-click a field (or click the ellipsis) and select “**New hierarchy**” (here we’ve selected “*Start of Year*”)

2) This creates a hierarchy field containing “*Start of Year*”, which we’ve renamed “**Date Hierarchy**”

3) Right-click other fields (like “*Start of Month*”) and select “**Add to Hierarchy**”

PRO TIP: IMPORTING MODELS FROM EXCEL



Already have a fully-built model in Excel?

Import files built with Power Query/Power Pivot directly into Power BI Desktop using ***Import > Excel Workbook Contents***

Imported models retain the following:

- Data source connections and queries
- Query editing procedures and applied steps
- Table relationships, hierarchies, field settings, etc.
- All calculated columns and DAX measures

PRO TIP:

Power Pivot includes some features that Power BI does not (filtering options, DAX function help, etc); if you are more comfortable in the Excel environment, build your models there and then import to Power BI!

BEST PRACTICES: CONNECTING & SHAPING DATA



Get yourself organized, *before* loading the data into Power BI

- *Define clear and intuitive table names (no spaces!) from the start; updating them later can be a headache, especially if you've referenced them in multiple places*
- *Establish a file/folder structure that makes sense from the start, to avoid having to modify data source settings if file names or locations change*



Disabling report refresh for any static sources

- *There's no need to constantly refresh sources that don't update frequently (or at all), like lookups or static data tables; only enable refresh for tables that will be changing*



When working with large tables, only load the data you need

- *Don't include hourly data when you only need daily, or product-level transactions when you only care about store-level performance; extra data will only slow you down*

CREATING A DATA MODEL

WHAT'S A "DATA MODEL"?

The screenshot shows the Power BI Data Model view. On the left, there's a navigation bar with icons for Home, Reports, and Data. The main area displays three tables:

- AW_Product_Lookup**: Contains columns ProductKey, ProductSubcategory, ProductSKU, and ProductName.
- AW_Sales_Data**: Contains columns OrderDate, StockDate, OrderNumber, ProductKey, CustomerKey, TerritoryKey, OrderLineItem, and OrderQuantity.
- AW_Returns_Data**: Contains columns ReturnDate, TerritoryKey, ProductKey, and ReturnQuantity.



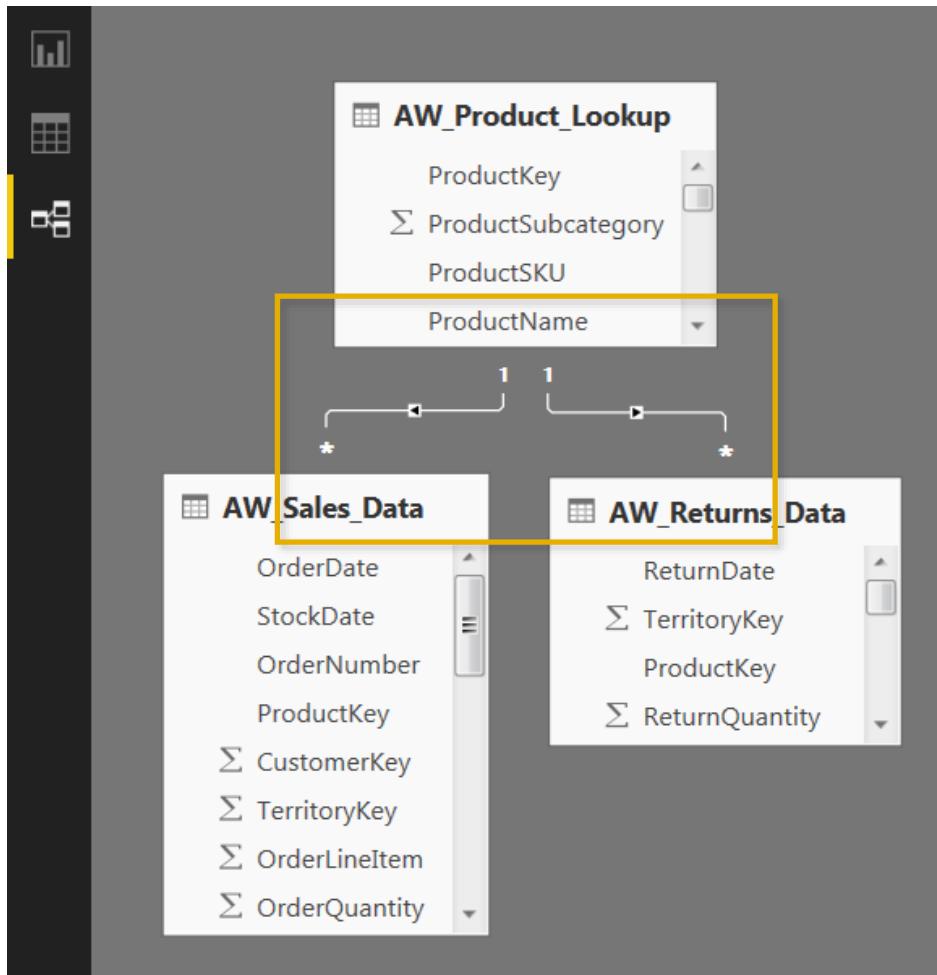
This IS NOT a data model



- This is a collection of independent tables, which share no connections or relationships
- If you tried to visualize **Orders** and **Returns** by **Product**, this is what you'd get

ProductName	OrderQuantity	ReturnQuantity
All-Purpose Bike Stand	84,174	1,828
AWC Logo Cap	84,174	1,828
Bike Wash - Dissolver	84,174	1,828
Cable Lock	84,174	1,828
Chain	84,174	1,828
Classic Vest, L	84,174	1,828
Classic Vest, M	84,174	1,828
Classic Vest, S	84,174	1,828
Fender Set - Mountain	84,174	1,828
Total	84,174	1,828

WHAT'S A "DATA MODEL"?



This **IS** a data model! 😊

- The tables are connected via relationships, based on the common *ProductKey* field
- Now the **Sales** and **Returns** tables know how to filter using fields from the **Product** table!

ProductName	OrderQuantity	ReturnQuantity
All-Purpose Bike Stand	234	8
AWC Logo Cap	4,151	46
Bike Wash - Dissolver	1,706	25
Classic Vest, L	182	4
Classic Vest, M	182	7
Classic Vest, S	157	8
Fender Set - Mountain	3,960	54
Half-Finger Gloves, L	840	18
Half-Finger Gloves, M	918	16
Total	84,174	1,828

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

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
1/1/1997
1/2/1997
1/3/1997
1/4/1997
1/5/1997
1/6/1997

This **Calendar Lookup** table provides additional attributes about each **date** (month, year, weekday, quarter, etc.)

product_id
1
2
3
4
5
6
7
8

This **Product Lookup** table provides additional attributes about each **product** (brand, product name, sku, price, etc.)

This **Data Table** contains “*quantity*” values, and connects to lookup tables via the “*date*” and “*product_id*” columns

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	20191444754	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

RELATIONSHIPS VS. MERGED TABLES



*Can't I just **merge queries** or use **LOOKUP** or **RELATED** functions to pull those attributes into the fact table itself, so that I have everything in one place??*

-Anonymous confused man

Original **Fact Table** fields

Attributes from **Calendar Lookup** table

Attributes from **Product Lookup** table

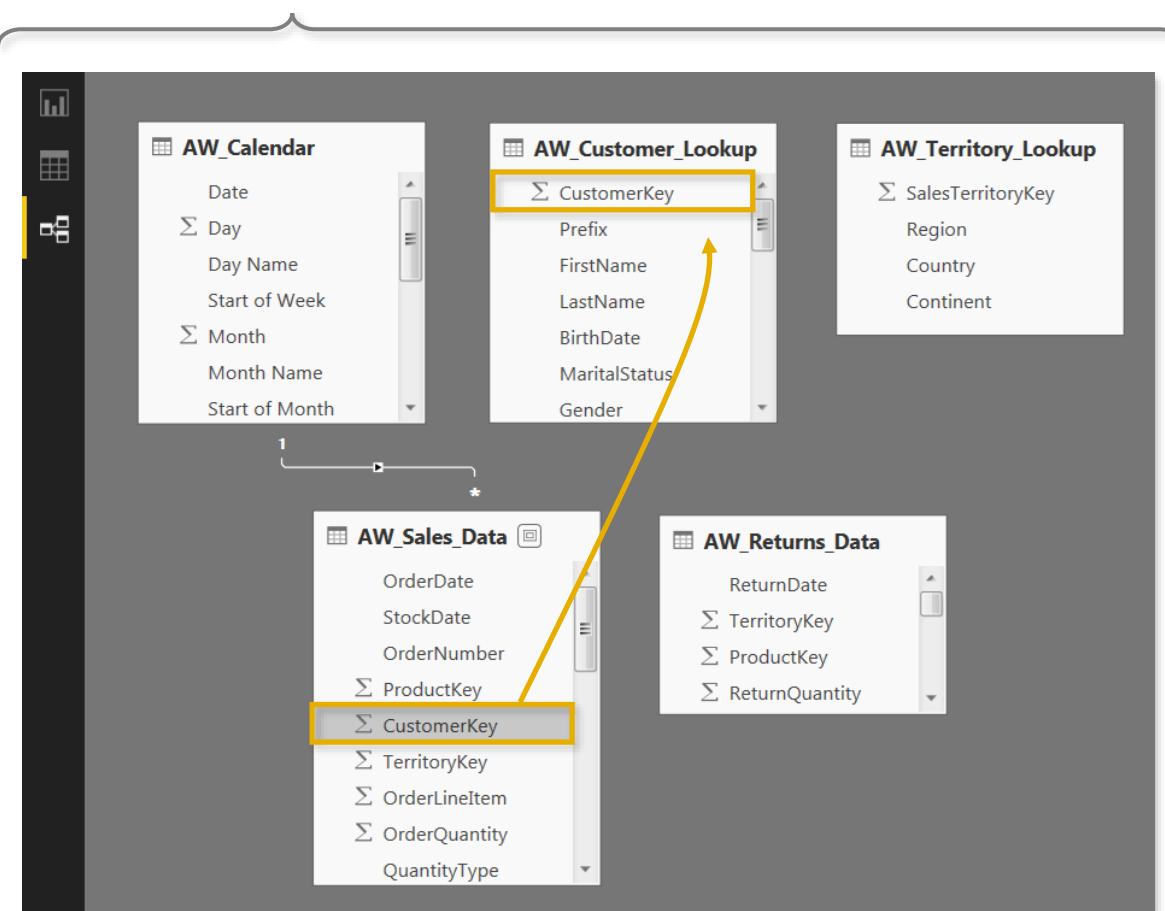
date	product_id	quantity	day_of_month	month	year	weekday	month_name	quarter	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	1	1	1997	Wednesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	7	1	1997	Tuesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	3	1	1997	Friday	January	Q1	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	1	1	1997	Wednesday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	6	1	1997	Monday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	5	1	1997	Sunday	January	Q1	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	5	1	1997	Sunday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	7	1	1997	Tuesday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	1	1	1997	Wednesday	January	Q1	Excellent	Excellent Cranberry Juice	36570182442	16.4

Sure you can, **but it's inefficient!**

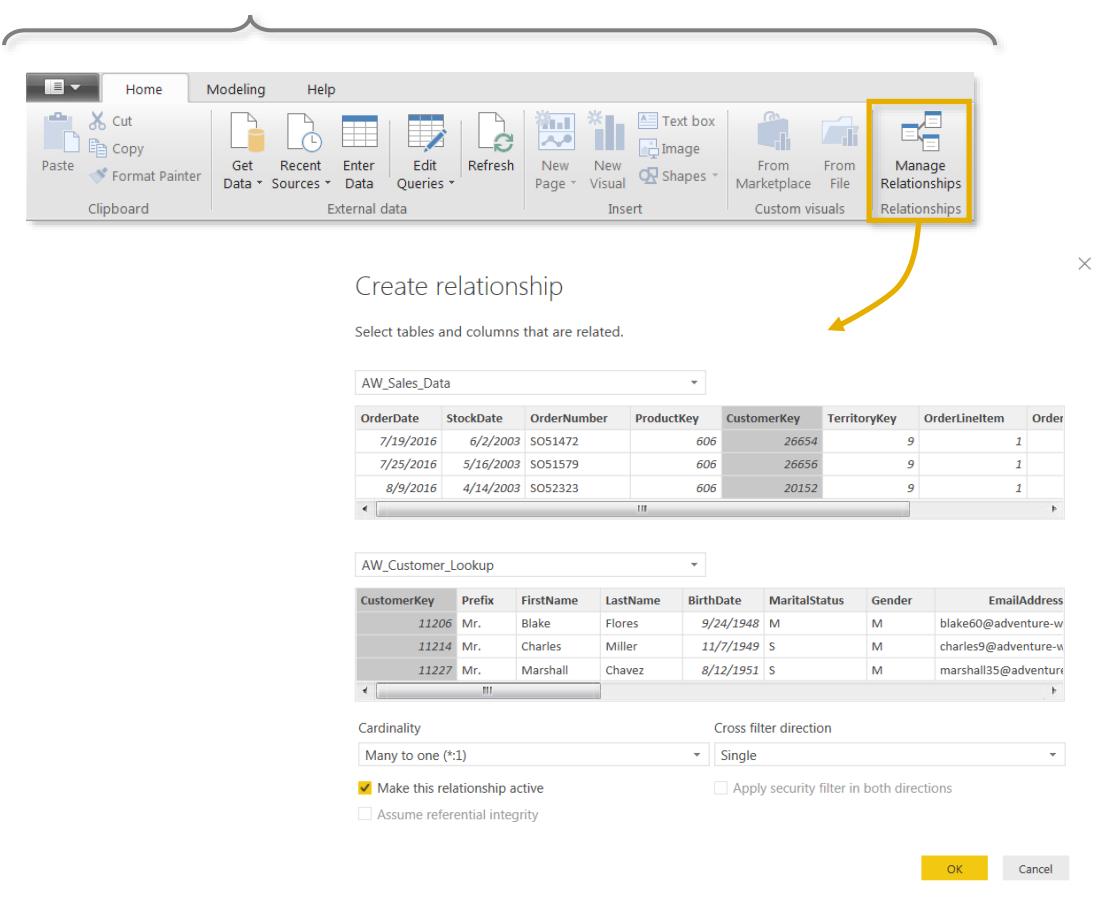
- Merging data in this way creates **redundant data** and utilizes **significantly more memory and processing power** than creating relationships between multiple small tables

CREATING TABLE RELATIONSHIPS

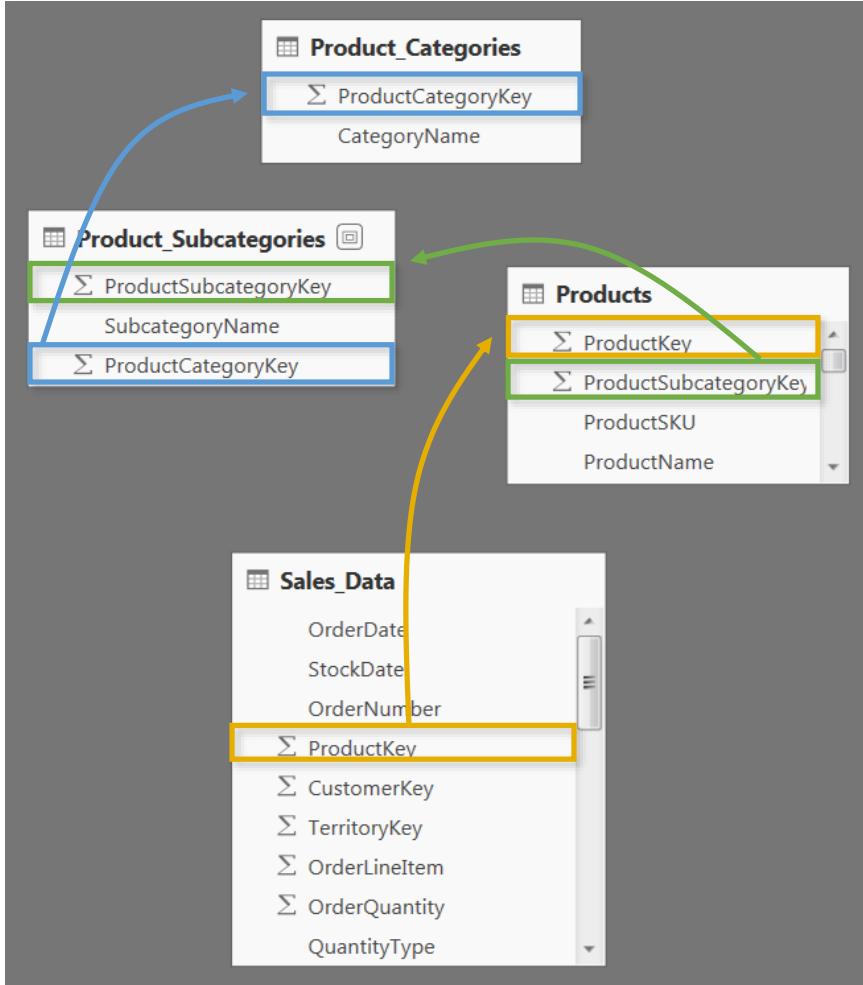
Option 1: Click and drag to connect primary and foreign keys within the **Relationships** pane



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



CREATING “SNOWFLAKE” SCHEMAS



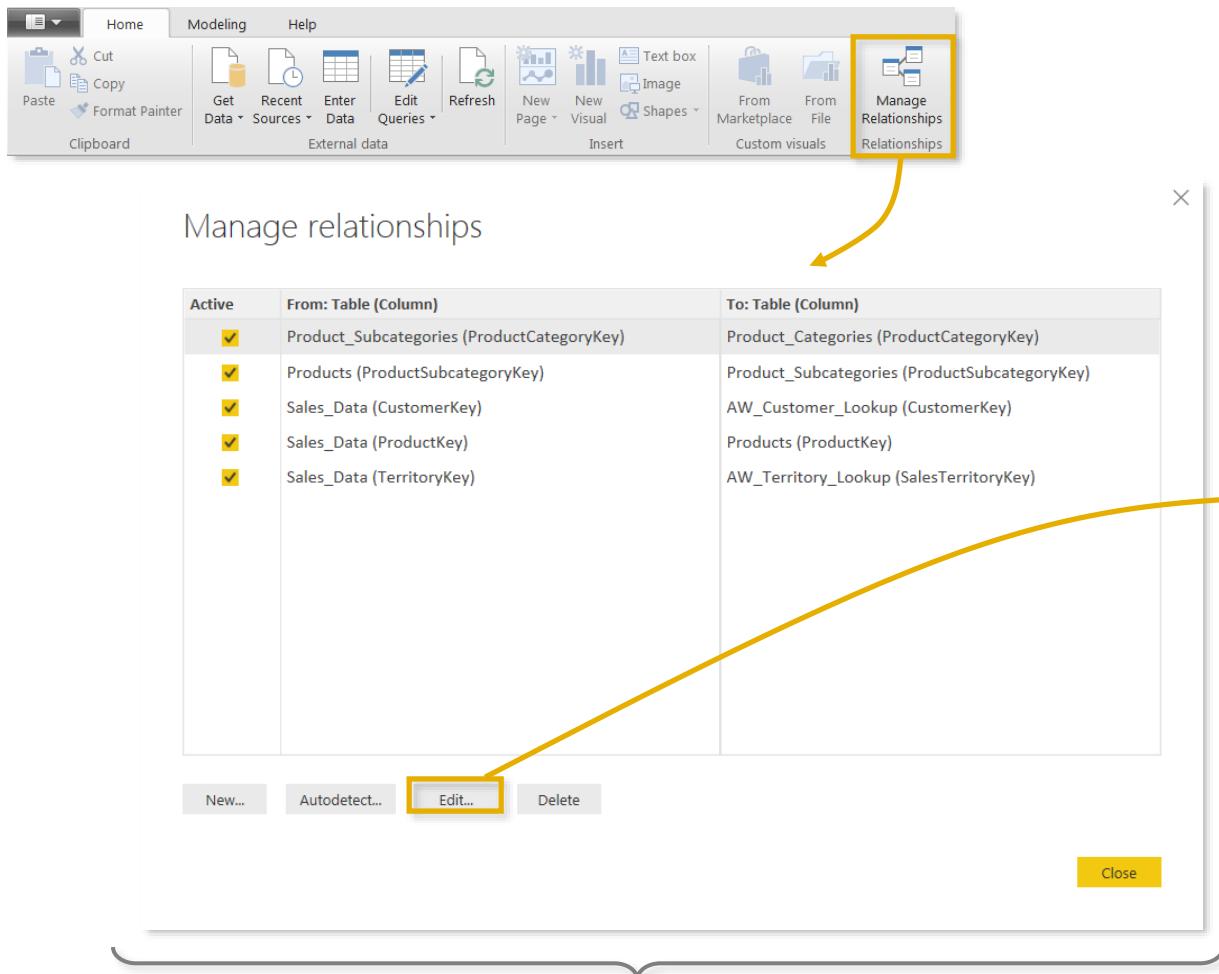
The **Sales_Data** table can connect to **Products** using the **ProductKey** field, but cannot connect directly to the **Subcategories** or **Categories** tables

By creating relationships from **Products** to **Subcategories** (using **ProductSubcategoryKey**) and **Subcategories** to **Categories** (using **ProductCategoryKey**), we have essentially connected **Sales_Data** to each lookup table; filter context will now flow all the way down the chain

PRO TIP:

Models with chains of dimension tables are often called “snowflake” schemas (whereas “star” schemas usually have individual lookup tables surrounding a central data table)

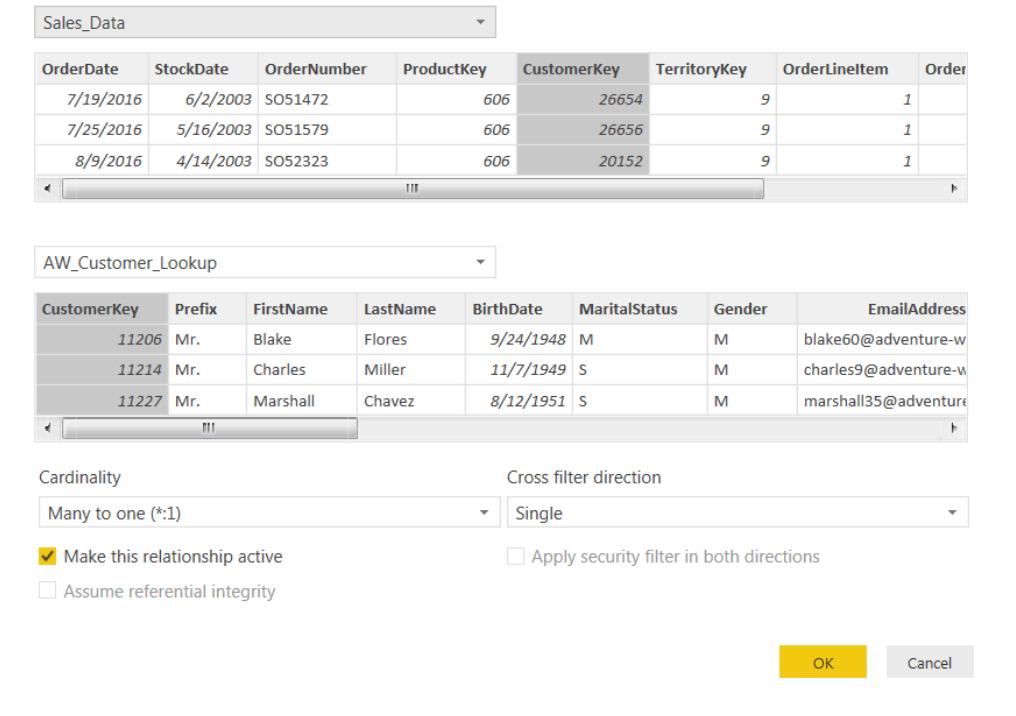
MANAGING & EDITING RELATIONSHIPS



The “**Manage Relationships**” dialog box allows you to **add, edit, or delete** table relationships

Edit relationship

Select tables and columns that are related.



Editing tools allow you to **activate/deactivate** relationships, view **cardinality**, and modify the **cross filter direction** (stay tuned!)

ACTIVE VS. INACTIVE RELATIONSHIPS

The diagram illustrates the creation of relationships between the **Sales_Data** and **Calendar** tables in a data model.

Relationship Diagram: On the left, the **Calendar** table is shown with various date-related columns. A relationship line connects the **Date** column in the **Calendar** table to the **OrderDate** column in the **Sales_Data** table. This relationship is highlighted with a yellow box. The **Sales_Data** table contains columns: OrderDate, StockDate, OrderNumber, ProductKey, CustomerKey, TerritoryKey, OrderLineItem, and OrderQuantity. The **OrderDate** column is also highlighted with a yellow box.

Edit Relationship Dialog (Left): The "Edit relationship" dialog shows the **Sales_Data** table selected. The **OrderDate** column is highlighted. The "Cardinality" section shows "Many to one (*:1)" and the "Cross filter direction" is set to "Single". The "Make this relationship active" checkbox is checked (highlighted with a yellow box). The "Assume referential integrity" checkbox is unchecked.

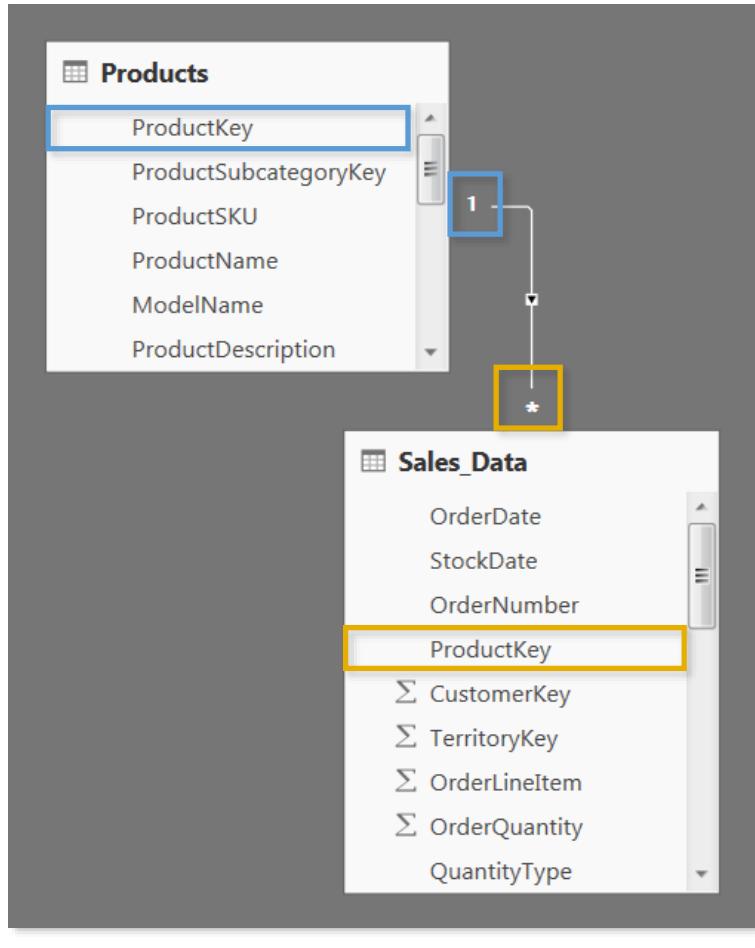
Edit Relationship Dialog (Right): The "Edit relationship" dialog shows the **Sales_Data** table selected. The **StockDate** column is highlighted. The "Cardinality" section shows "Many to one (*:1)" and the "Cross filter direction" is set to "Single". The "Make this relationship active" checkbox is unchecked (highlighted with a yellow box). The "Assume referential integrity" checkbox is unchecked.

Text:

The **Sales_Data** table contains two date fields (**OrderDate & StockDate**), but there can only be one *active* relationship to the Date field in the **Calendar** table

Double-click the relationship line, and check the “**Make this relationship active**” box to toggle (note that you have to deactivate one in order to activate another)

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*

*In this case, there is only **ONE instance of each ProductKey** in the **Products** table (noted by the “1”), since each row contains **attributes of a single product** (Name, SKU, Description, Retail Price, etc)*

*There are **MANY instances of each ProductKey** in the **Sales_Data** table (noted by the asterisk *), since there are **multiple sales associated with each product***

CARDINALITY CASE STUDY: MANY-TO-MANY

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
4	Washington Diet Cream Soda	81727382373
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

date	product_id	transactions
1/1/2017	4	12
1/2/2017	4	9
1/3/2017	4	11
1/1/2017	5	16
1/2/2017	5	19
1/1/2017	7	11



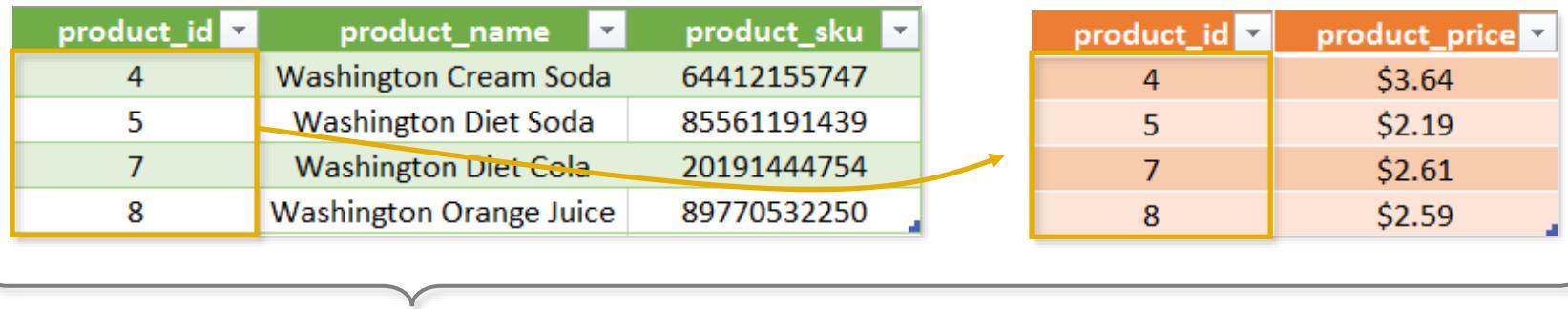
Create relationship

You can't create a relationship between these two columns because one of the columns must have unique values.

OK

- If we try to connect these tables using **product_id**, we'll get a "**many-to-many relationship**" error since there are multiple instances of each ID in both tables
- Even if we *could* create this relationship, how would you know which product was actually sold on each date – *Cream Soda* or *Diet Cream Soda*?

CARDINALITY CASE STUDY: ONE-TO-ONE



- Connecting the two tables above using the **product_id** field creates a **one-to-one relationship**, since each ID only appears once in each table
- Unlike many-to-many, there is nothing *illegal* about this relationship; it's just **inefficient**

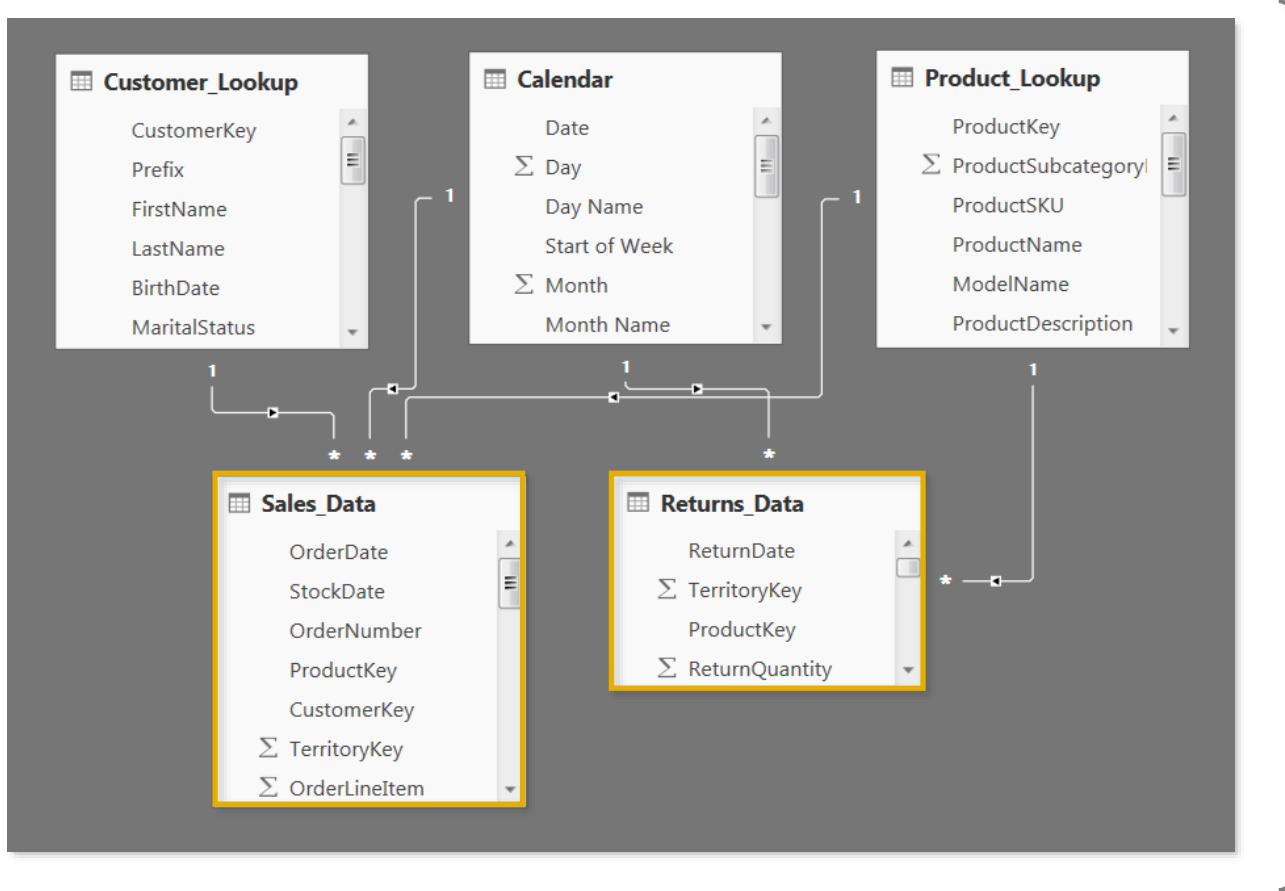
To eliminate the inefficiency, you could simply **merge the two tables** into a single, valid lookup

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

The diagram shows a single merged table with four rows, where the data from both original tables has been combined. The table includes columns for product_id, product_name, product_sku, and product_price.

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

CONNECTING MULTIPLE DATA TABLES



This model contains two data tables:
Sales_Data and **Returns_Data**

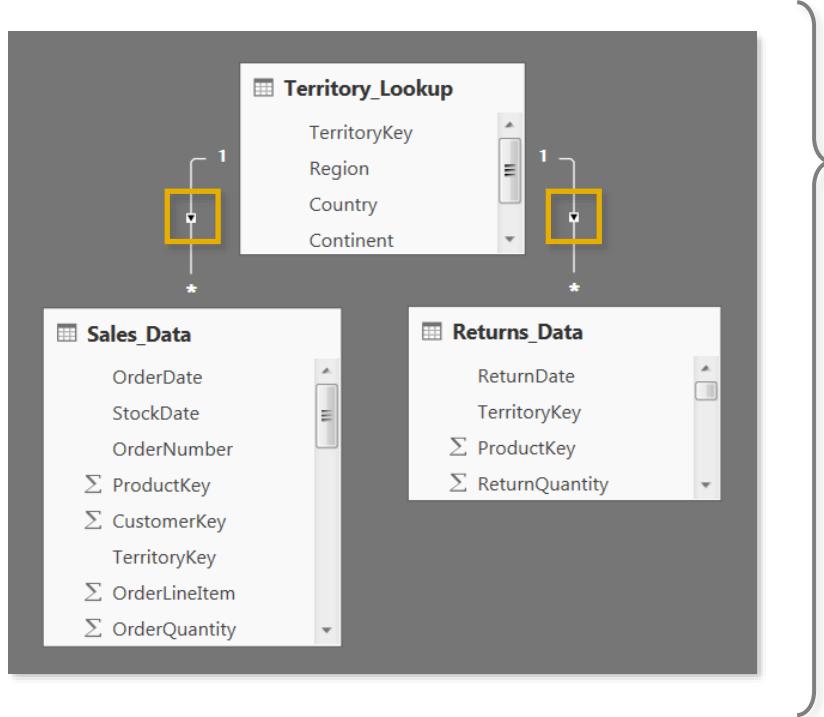
- Note that the **Returns** table connects to **Calendar** and **Product_Lookup** just like the **Sales** table, but without a *CustomerKey* field it cannot be joined to **Customer_Lookup**
- This allows us to analyze sales and returns within the same view, **but only if we filter or segment the data using shared lookups**
 - In other words, we know which **product** was returned and on which **date**, but nothing about which **customer** made the return



HEY THIS IS IMPORTANT!

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

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)**

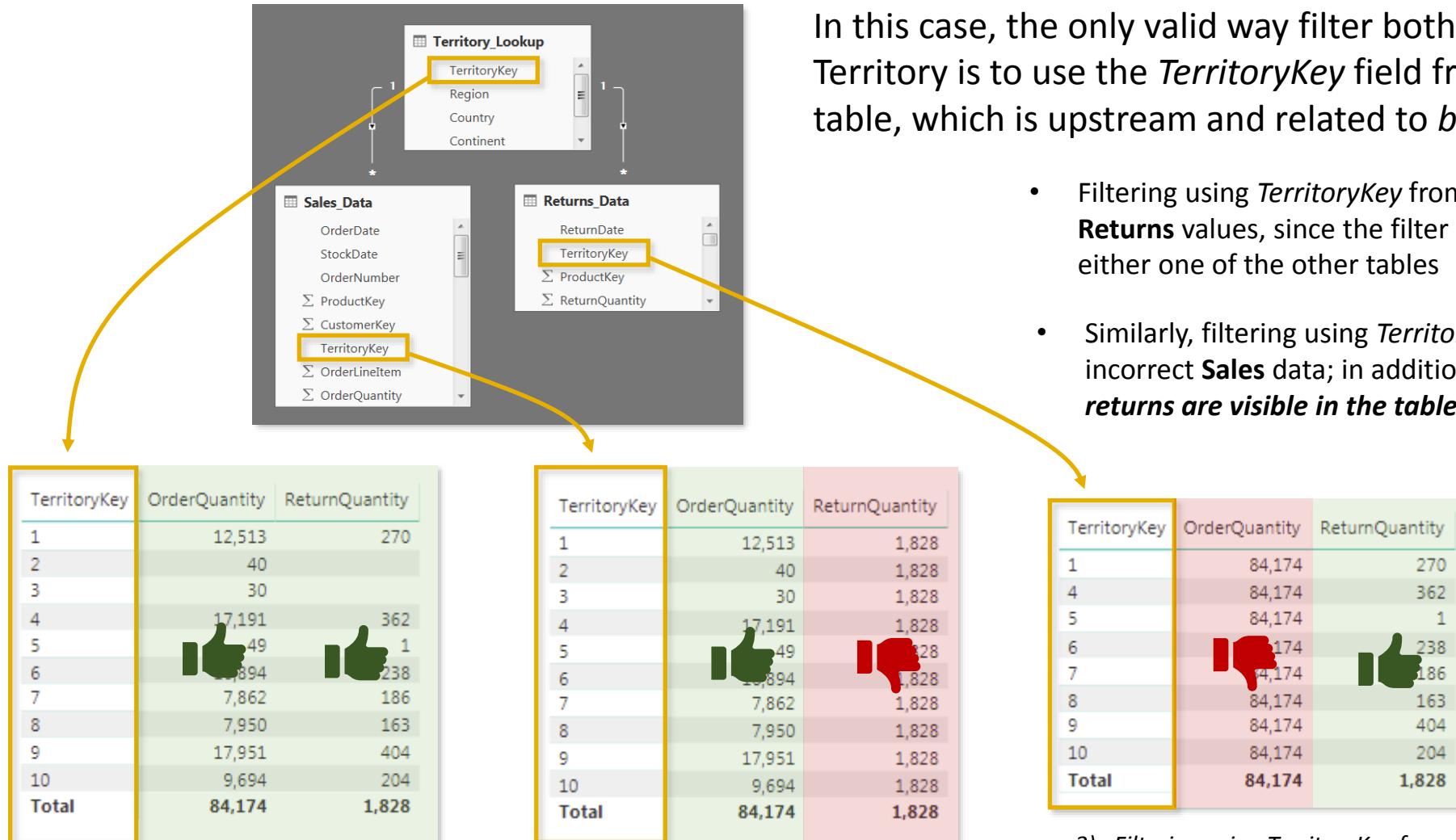
- When you filter a table, that filter context is passed along to all related “*downstream*” tables (following the direction of the arrow)
- Filters **cannot** flow “*upstream*” (against the direction of the arrow)



PRO TIP:

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

FILTER FLOW (CONT.)



In this case, the only valid way filter both **Sales** and **Returns** data by Territory is to use the **TerritoryKey** field from the **Territory_Lookup** table, which is upstream and related to *both* data tables

- Filtering using **TerritoryKey** from the **Sales** table yields incorrect **Returns** values, since the filter context *cannot flow upstream* to either one of the other tables
- Similarly, filtering using **TerritoryKey** from the **Returns** table yields incorrect **Sales** data; in addition, **only territories that registered returns are visible in the table** (even though they registered sales)

1) Filtering using TerritoryKey from the Territory_Lookup table

2) Filtering using TerritoryKey from the Sales_Data table

3) Filtering using TerritoryKey from the Returns_Data table

TWO-WAY FILTERS

Edit relationship

Select tables and columns that are related.

Sales Data

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity
7/19/2016	6/2/2003	SO51472	606	26654	9	1	1
7/25/2016	5/16/2003	SO51579	606	26656	9	1	1
8/9/2016	4/14/2003	SO52323	606	20152	9	1	1

Territory_Lookup

TerritoryKey	Region	Country	Continent
1	Northwest	United States	North America
2	Northeast	United States	North America
3	Central	United States	North America

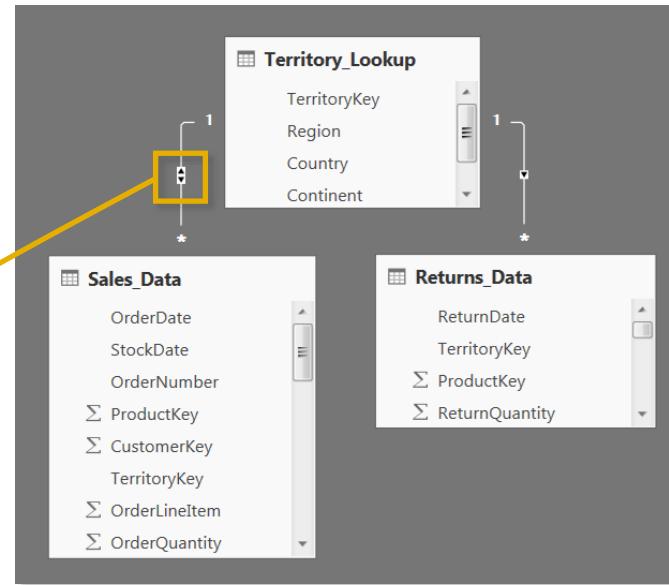
Cardinality: Many to one (*:1)

Cross filter direction: Both

Make this relationship active

Assume referential integrity

OK Cancel

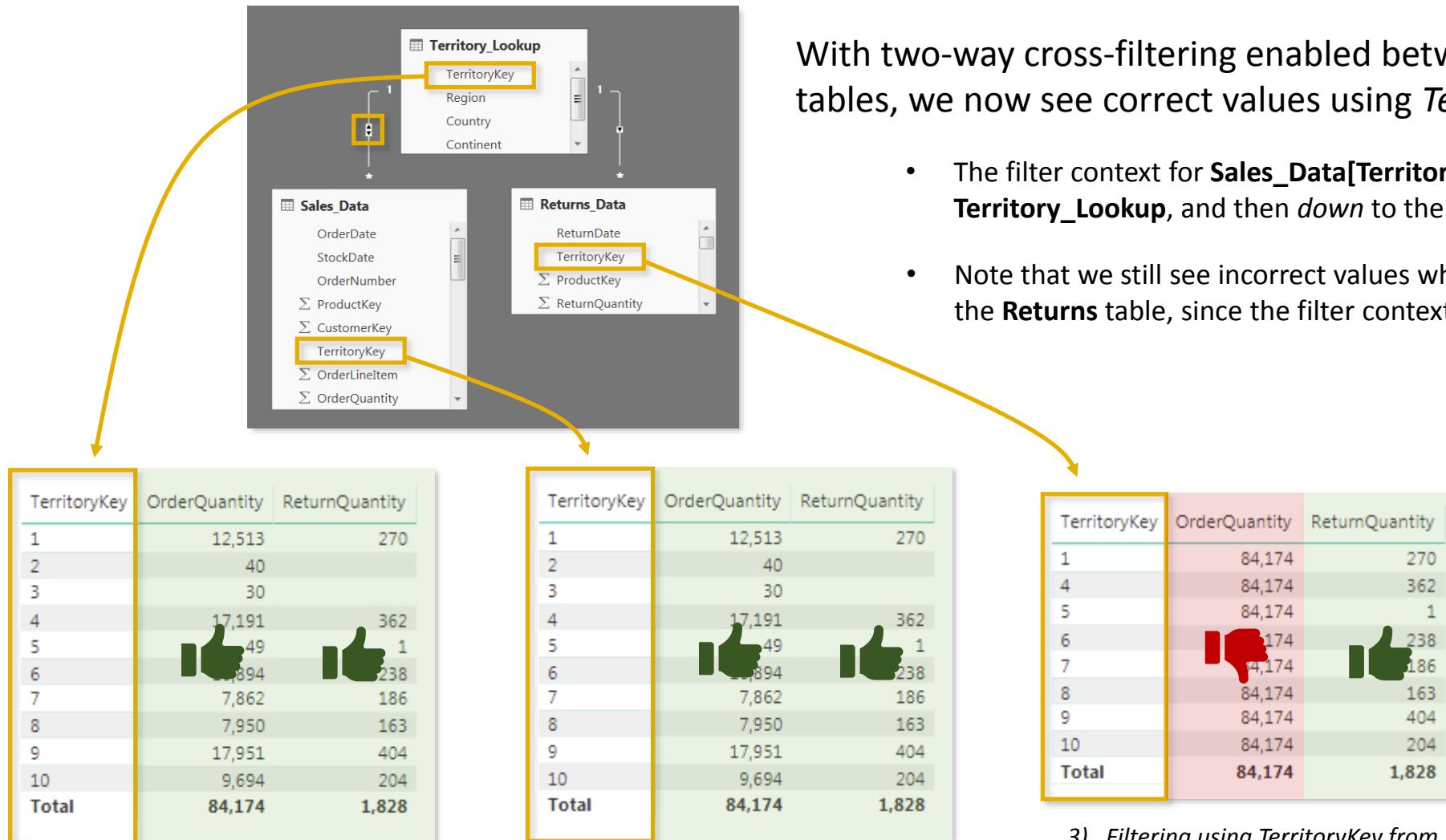


Updating the filter direction between **Sales** and **Territory** from “Single” to “Both” allows filter context to flow both ways

- This means that filters applied to the **Sales_Data** table will pass to the lookup, and then down to the **Returns_Data** table

NOTE: The “Apply security filter in both directions” option relates to row-level security (RLS) settings, which are not covered in this course

TWO-WAY FILTERS (CONT.)



With two-way cross-filtering enabled between the **Sales** and **Territory** tables, we now see correct values using **TerritoryKey** from either table

- The filter context for **Sales_Data[TerritoryKey]** now passes *up* to the **Territory_Lookup**, and then *down* to the **Returns_Data** table
- Note that we still see incorrect values when filtering using **TerritoryKey** from the **Returns** table, since the filter context is isolated to that single table

TWO-WAY FILTERS (CONT.)

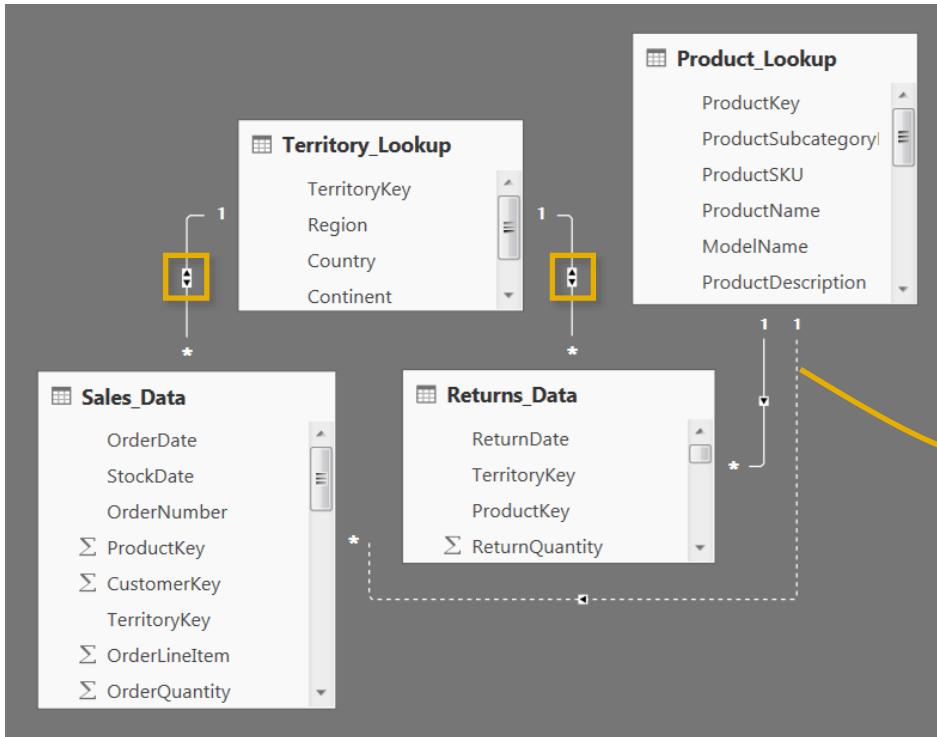


In this case, we've enabled two-way cross-filtering between the **Returns** and **Territory** tables

- As expected, we now see incorrect values when filtering using **TerritoryKey** from the **Sales** table, since the filter context is isolated to that single table
- While the values *appear* to be correct when filtering using **TerritoryKey** from the **Returns** table, we're **missing sales data** from any territories that didn't register returns (*specifically Territories 2 & 3*)

Since no information about Territory 2 or 3 is passed from the **Returns_Data** table to **Territory_Lookup**, they get filtered out of the lookup, and subsequently filtered out of the **Sales_Data**

TWO-WAY FILTERS: A WORD OF WARNING



Use two-way filters carefully, and **only when necessary***

- If you try to use multiple two-way filters in a more complex model, you run the risk of creating “**ambiguous relationships**” by introducing multiple filter paths between tables:

! You can't create a direct active relationship between Sales_Data and Product_Lookup because that would introduce ambiguity between tables Product_Lookup and Territory_Lookup. To make this relationship active, deactivate or delete one of the relationships between Product_Lookup and Territory_Lookup first.

In this model, filter context from the **Product_Lookup** table can pass down to **Returns_Data** and up to **Territory_Lookup**, which would filter accordingly based on the TerritoryKey values passed from the Returns table

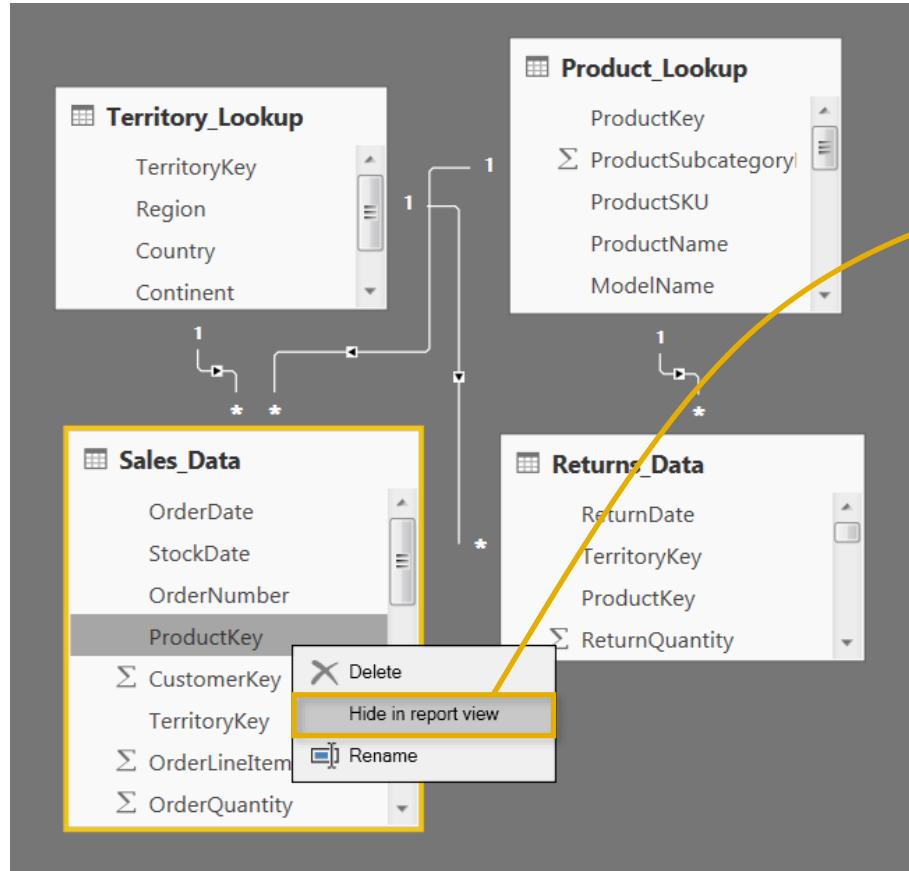
If we were able to activate the relationship between **Product_Lookup** and **Sales_Data** as well, filters could pass from the **Product_Lookup** table through EITHER the Sales or Returns table to reach the **Territory_Lookup**, which could yield conflicting filter context

PRO TIP:

Design your models with **one-way filters** and **1-to-Many cardinality**, unless more complex relationships are necessary

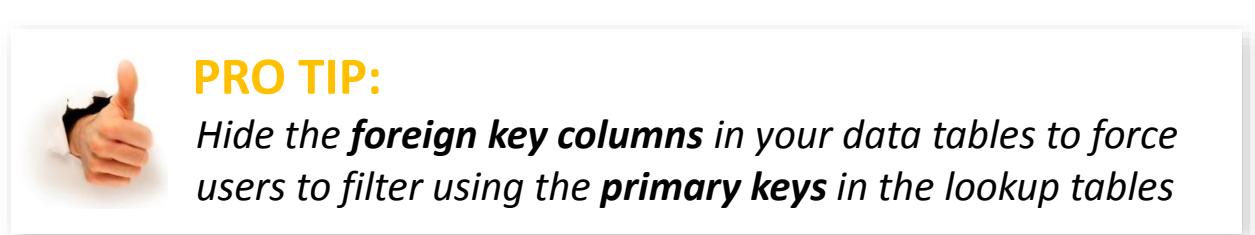


HIDING FIELDS FROM REPORT VIEW



Hiding fields from Report View makes them inaccessible from the Report tab (*although they can still be accessed within the **Data** or **Relationships** views*)

This is commonly used to prevent users from filtering using invalid fields, or to hide irrelevant metrics from view



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*

CALCULATED FIELDS WITH DAX

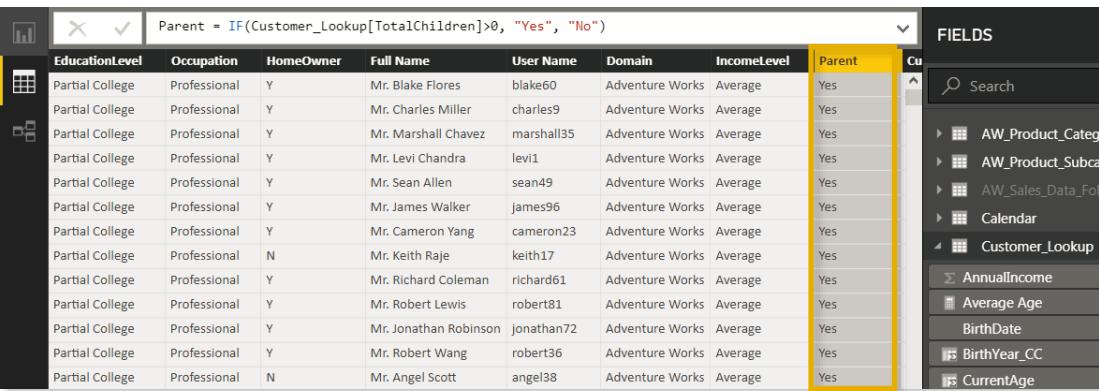
MEET DAX

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

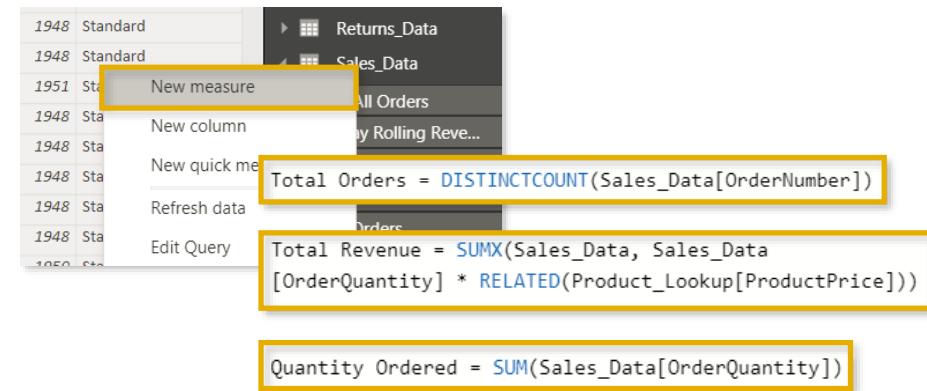
- Add *calculated columns* and *measures* to your model, using intuitive syntax
- Go beyond the capabilities of traditional “grid-style” formulas, with powerful and flexible functions built specifically to work with relational data models

Two ways to use DAX

1) *Calculated Columns*



2) *Measures*



The image displays two screenshots from the Power BI interface. The left screenshot shows the 'Data View' where a new calculated column 'Parent' is being defined using the DAX formula: `Parent = IF(Customer_Lookup[TotalChildren]>0, "Yes", "No")`. The 'Customer_Lookup' table is highlighted with a yellow box. The right screenshot shows the 'Model View' where three measures are being created: `Total Orders = DISTINCTCOUNT(Sales_Data[OrderNumber])`, `Total Revenue = SUMX(Sales_Data, Sales_Data[OrderQuantity] * RELATED(Product_Lookup[ProductPrice]))`, and `Quantity Ordered = SUM(Sales_Data[OrderQuantity])`. The 'Sales_Data' table is highlighted with a yellow box.

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)

The screenshot shows the Power BI Data Editor interface. A calculated column named "Parent" is defined with the formula: `Parent = IF(Customer_Lookup[TotalChildren]>0, "Yes", "No")`. The "FIELDS" pane on the right lists various tables and columns, including "Customer_Lookup", "AnnualIncome", "Average Age", "BirthDate", and "BirthYear_CC". A green thumbs-up icon is placed over the "Parent" column.

Level	Occupation	HomeOwner	Full Name	User Name	Domain	IncomeLevel	Parent
College	Professional	Y	Mr. Blake Flores	blake60	Adventure Works	Average	Yes
College	Professional	Y	Mr. Charles Miller	charles9	Adventure Works	Average	Yes
College	Professional	Y	Mr. Marshall Chavez	marshall35	Adventure Works	Average	Yes
College	Professional	Y	Mr. Levi Chandra	levi1	Adventure Works	Average	Yes
College	Professional	Y	Mr. Sean Allen	sean49	Adventure Works	Average	Yes
College	Professional	Y	Mr. James Walker	james96	Adventure Works	Average	Yes
College	Professional	Y	Mr. Cameron Yang	cameron23	Adventure Works	Average	Yes
College	Professional	N	Mr. Keith Raje	keith17	Adventure Works	Average	Yes
College	Professional	Y	Mr. Richard Coleman	richard61	Adventure Works	Average	Yes
College	Professional	Y	Mr. Robert Lewis	robert81	Adventure Works	Average	Yes
College	Professional	Y	Mr. Jonathan Robinson	jonathan72	Adventure Works	Average	Yes
College	Professional	Y	Mr. Robert Wang	robert36	Adventure Works	Average	Yes

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.

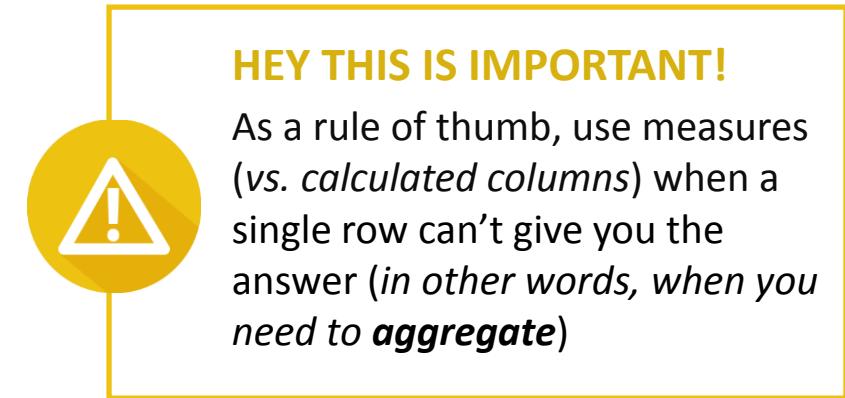
The screenshot shows the Power BI Data Editor interface. A calculated column named "TotalQuantity" is defined with the formula: `TotalQuantity = SUM(AW_Sales_Data[OrderQuantity])`. The "FIELDS" pane on the right lists various tables and columns, including "AW_Sales_Data", "% of All Orders", "10-Day Rolling Rev...", "Adjusted Revenue", and "ALL Orders". A red thumbs-down icon is placed over the "TotalQuantity" column.

OrderDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	QuantityType	TotalQuantity
6/3/2002	SO46718	360	12570	9	1	1	Single Item	84174
6/22/2002	SO46736	360	12341	9	1	1	Single Item	84174
6/5/2002	SO46776	360	12356	9	1	1	Single Item	84174
6/22/2002	SO46808	360	12347	9	1	1	Single Item	84174
6/11/2002	SO46826	360	12575	9	1	1	Single Item	84174
6/21/2002	SO47075	360	12685	9	1	1	Single Item	84174
6/1/2002	SO47098	360	12667	9	1	1	Single Item	84174
6/21/2002	SO47149	360	12669	9	1	1	Single Item	84174
6/4/2002	SO47212	360	12580	9	1	1	Single Item	84174
6/29/2002	SO47302	360	12670	9	1	1	Single Item	84174
6/12/2002	SO47328	360	12681	9	1	1	Single Item	84174
6/13/2002	SO47346	360	12585	9	1	1	Single Item	84174
6/12/2002	SO47744	360	12989	9	1	1	Single Item	84174
6/28/2002	SO47745	360	12998	9	1	1	Single Item	84174

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*)



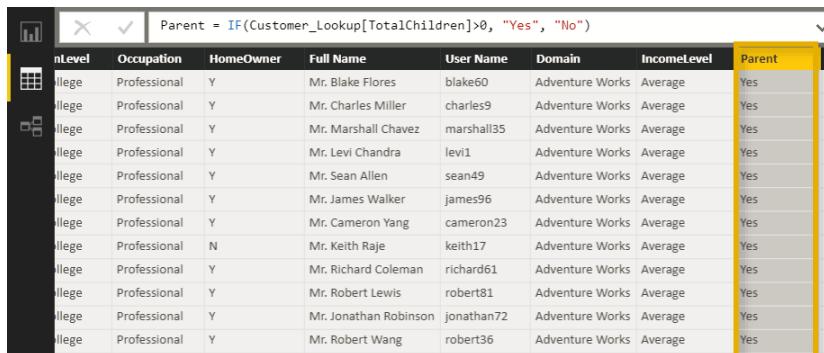
PRO TIP:

*Use measures to create **numerical, calculated values** that can be analyzed in the “**values**” field of a report visual*

RECAP: CALCULATED COLUMNS VS. MEASURES

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**



A screenshot of the Power BI Data View interface. A table is displayed with columns: nLevel, Occupation, HomeOwner, Full Name, User Name, Domain, IncomeLevel, and Parent. The Parent column contains the formula: `Parent = IF(Customer_Lookup[TotalChildren]>0, "Yes", "No")`. The table shows several rows of data, such as Mr. Blake Flores, Mr. Charles Miller, Mr. Marshall Chavez, Mr. Levi Chandra, Mr. Sean Allen, Mr. James Walker, Mr. Cameron Yang, Mr. Keith Raje, Mr. Richard Coleman, Mr. Robert Lewis, Mr. Jonathan Robinson, and Mr. Robert Wang.

Calculated columns “live” in tables

MEASURES

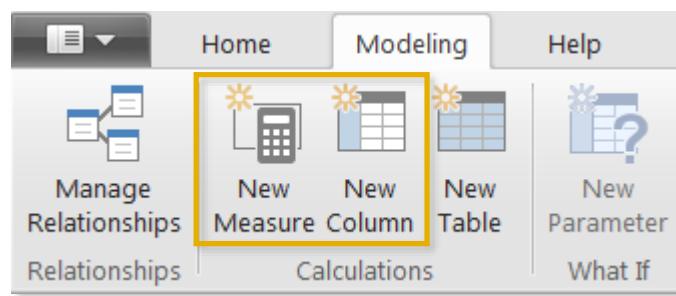
- Values are calculated based on information from any filters in the report (**has filter context**)
- Does not create new data in the tables themselves (*doesn’t increase file size*)
- Recalculate in response to any change to filters within the report
- Almost *always* used within the **values** field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



Measures “live” in visuals

ADDING COLUMNS & MEASURES

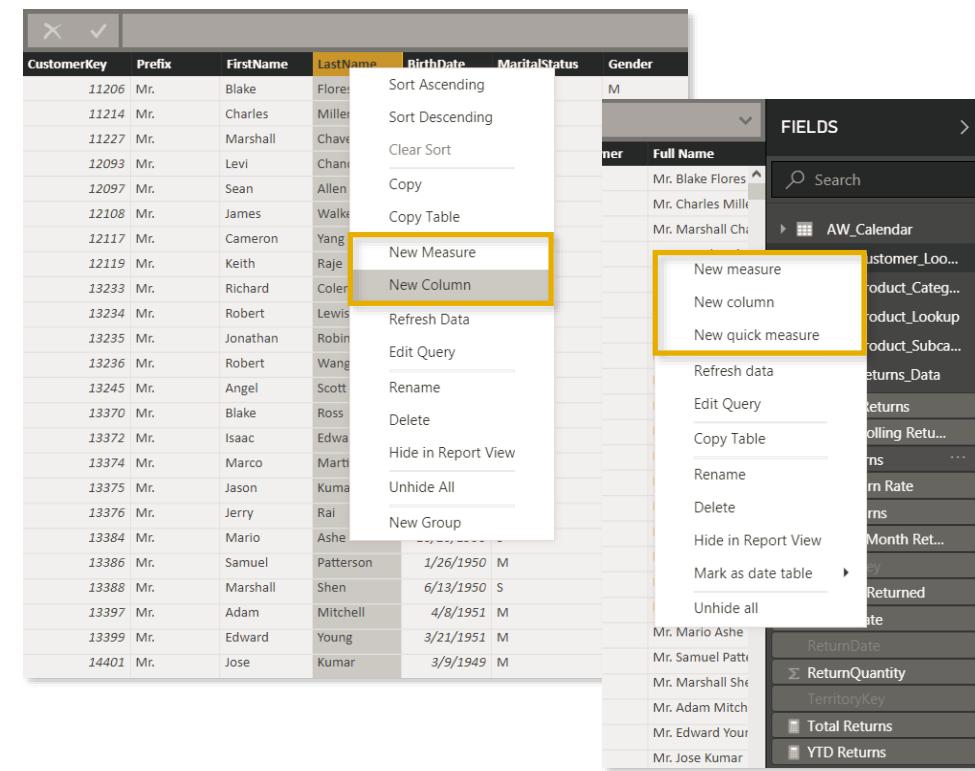
Option 1: Select “New Measure” or “New Column” from the **Modeling** tab



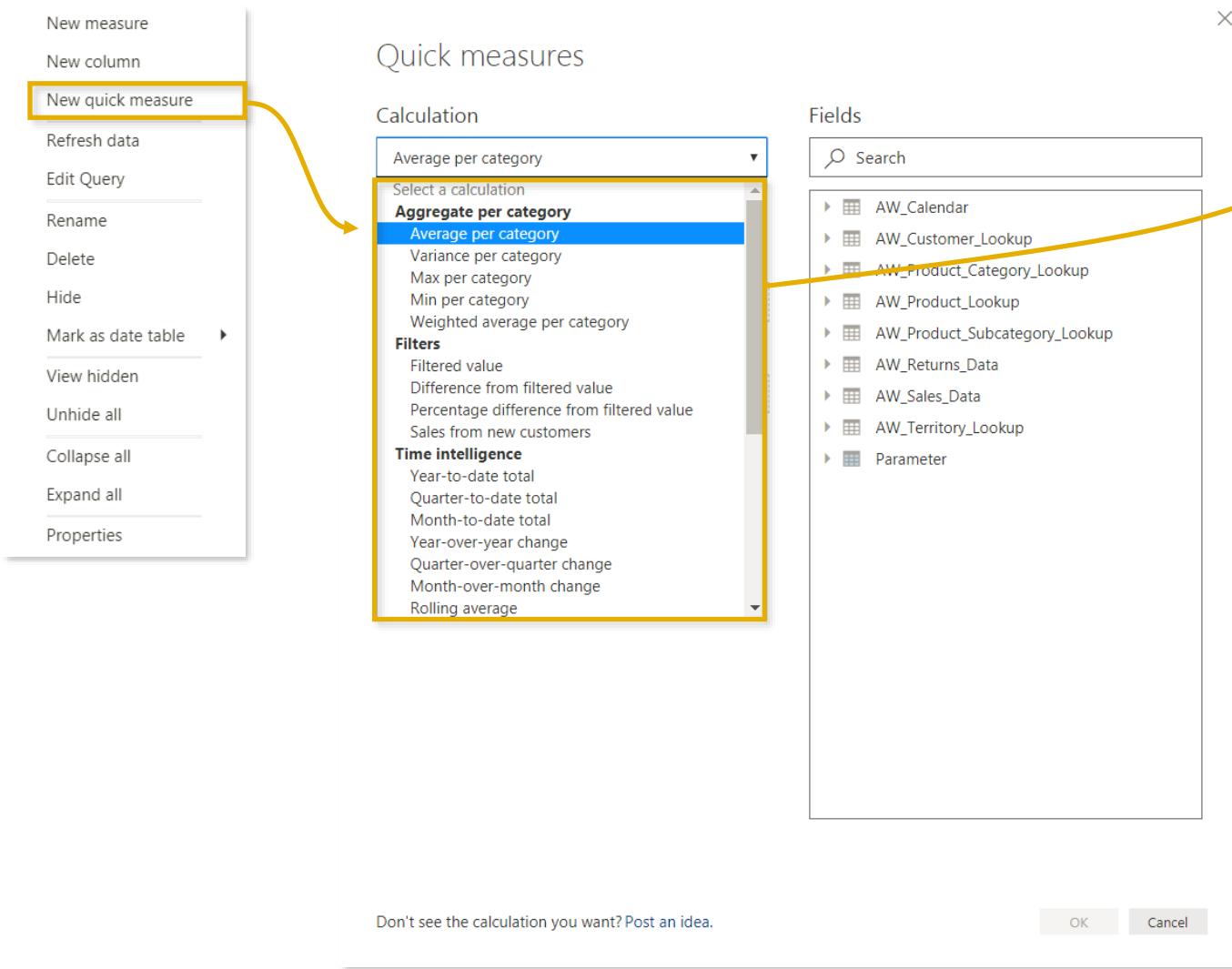
When you insert Columns or Measures using the **Modeling** tab (Option 1), they are assigned to whichever table is *currently selected*, or the *first table in the field list* by default

- Measures can be reassigned to new “Home” tables (under the “Properties” options in the **Modeling** tab), but the Option 2 allows you to be more deliberate about placing them
 - **Note:** Assigning measures to specific tables doesn’t have any impact on functionality – it’s just a way to keep them organized

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

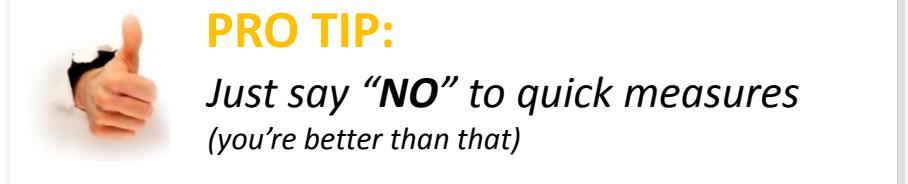


QUICK MEASURES



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



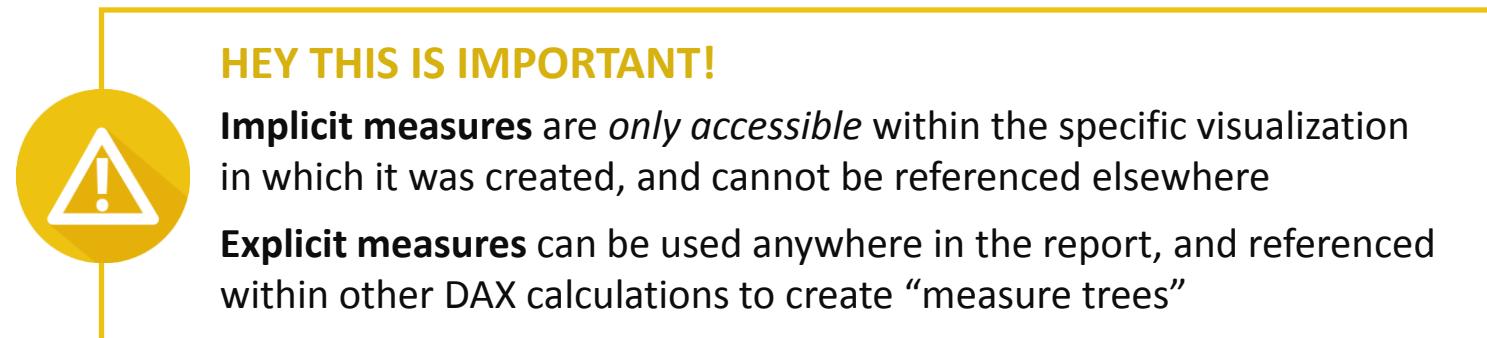
IMPLICIT VS. EXPLICIT MEASURES

The screenshot shows the Power BI desktop interface. On the left, there's a visualizations pane with various chart icons. Below it are sections for Axis, Drag data fields here, Legend, and Value. In the Value section, the field 'OrderQuantity' is selected, highlighted with a yellow box. A context menu is open over this field, listing options: Remove field, Rename, Sum (which is selected and highlighted with a yellow box), Average, Minimum, Maximum, Count (Distinct), Count, Standard deviation, Variance, Median, Show value as, and New quick measure. At the bottom of the Value section, there's a list of other fields: ALL Orders, Bulk Orders, High Ticket Or..., Order Target, OrderLineItem, OrderNumber, and OrderQuantity, with the last one also highlighted with a yellow box.

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



UNDERSTANDING FILTER CONTEXT

Remember that measures are evaluated based on **filter context**, which means that they recalculate whenever the fields or filters around them change

ProductName	Total Orders	Return Rate
Water Bottle - 30 oz.	1,164	1.96 %
Road Tire Tube	829	1.62 %
AWC Logo Cap	803	0.93 %
Patch Kit/8 Patches	798	1.57 %
Sport-100 Helmet, Red	753	2.79 %
Touring Tire Tube	702	1.35 %
Sport-100 Helmet, Blue	666	3.15 %
Sport-100 Helmet, Black	626	3.67 %
Road Bottle Cage	560	1.58 %
Mountain Tire Tube	554	1.95 %
Mountain Bottle Cage	539	1.38 %
Touring Tire	427	1.16 %
LL Road Tire	421	2.02 %
Fender Set - Mountain	378	1.82 %
ML Road Tire	297	1.72 %
ML Mountain Tire	266	1.94 %
HL Mountain Tire	206	3.40 %
Mountain-200 Silver, 46	199	1.51 %
Mountain-200 Black, 46	196	3.06 %
LL Mountain Tire	195	2.09 %
Mountain-200 Silver, 38	189	2.65 %
Bike Wash - Dissolver	187	2.38 %
Mountain-200 Black, 42	182	3.85 %
Mountain-200 Black, 38	180	3.33 %
Long-Sleeve Logo Jersey, M	161	4.35 %
HL Road Tire	158	5.06 %
Mountain-200 Silver, 42	153	1.28 %
Hydration Pack - 70 oz.	147	4.08 %
Long-Sleeve Logo Jersey, L	147	2.72 %
Long-Sleeve Logo Jersey, S	130	2.31 %
Total	7,380	2.17 %

For this particular value in the matrix, the **Total Orders** measure is calculated based on the following filter context: *Products[ProductName] = “Touring Tire Tube”*

- This allows the measure to return the total order quantity for each product specifically (or whatever the row and column labels dictate – years, countries, product categories, customer names, etc)

This Total is **not** calculated by summing the values above; it evaluates as its own measure, with **no filter context** (*since we aren’t calculating orders for a specific product*)



HEY THIS IS IMPORTANT!

Each measure value in a report is **like an island**, and calculates according to its own filter context (*even Totals and Grand Totals*)

FILTER CONTEXT (EXAMPLES)

MEASURE: Total Revenue

FILTER CONTEXT:

- *Calendar[Year]* = 2016 or 2017
- *Customers[Full Name]* = Mr. Larry Munoz

Full Name	Total Orders	Total Revenue
Mr. Maurice Shan	6	\$12,407.95
Mrs. Janet Munoz	6	\$12,015.39
Mrs. Lisa Choi	7	\$11,330.44
Mrs. Lacey Zheng	7	\$11,085.74
Mr. Jordan Turner	7	\$11,022.38
Mr. Larry Munoz	7	\$10,952.04
Mrs. Ariana Gray	6	\$10,391.42
Mr. Marco Lopez	6	\$10,289.68
Mr. Franklin Xu	5	\$10,164.34
Mrs. Margaret He	4	\$9,266.74
Mrs. Kaitlyn Henderson	4	\$9,258.92
Mrs. Nichole Nara	4	\$9,234.66
Mr. Randall Dominguez	4	\$9,210.36
Mrs. Rosa Hu	4	\$9,201.2
Adriana Gonzalez	4	\$9,195.69
Mrs. Dominique Prasad	6	\$9,180.93
Mrs. Brandi Gill	4	\$9,166.18
Mr. Brad She	4	\$9,161.01
Mr. Francisco Sara	4	\$9,125.54
Mr. Kevin Coleman	4	\$7,750.53
Mr. Johnathan Suri	4	\$7,721.33
Mrs. Crystal Zeng	4	\$7,706.81
Mrs. Felicia Blanco	4	\$7,669.66
Mrs. Jill Suarez	4	\$7,652.61
Mr. Preston Raman	4	\$7,599.49
Mr. Willie Xu	4	\$7,553.55
Mrs. Abby Subram	4	\$7,308.39
Mr. Lance Blanco	4	\$7,207.07
Mrs. Audrey Blanco	4	\$7,139.17
Mr. Ricky Navarro	3	\$7,119.63
Mr. Eddie Dominguez	3	\$7,044.38
Mrs. Molly Madan	3	\$7,043.25
Mr. Jarrod Mehta	3	\$7,038.37
Ms. Susan Zhou	3	\$7,027.98
Mr. Brent Zhang	3	\$7,018.99
Ms. Alyssa Bradley	3	\$7,018.84
Total	22,534	\$18,509,633.2

MEASURE: Total Orders

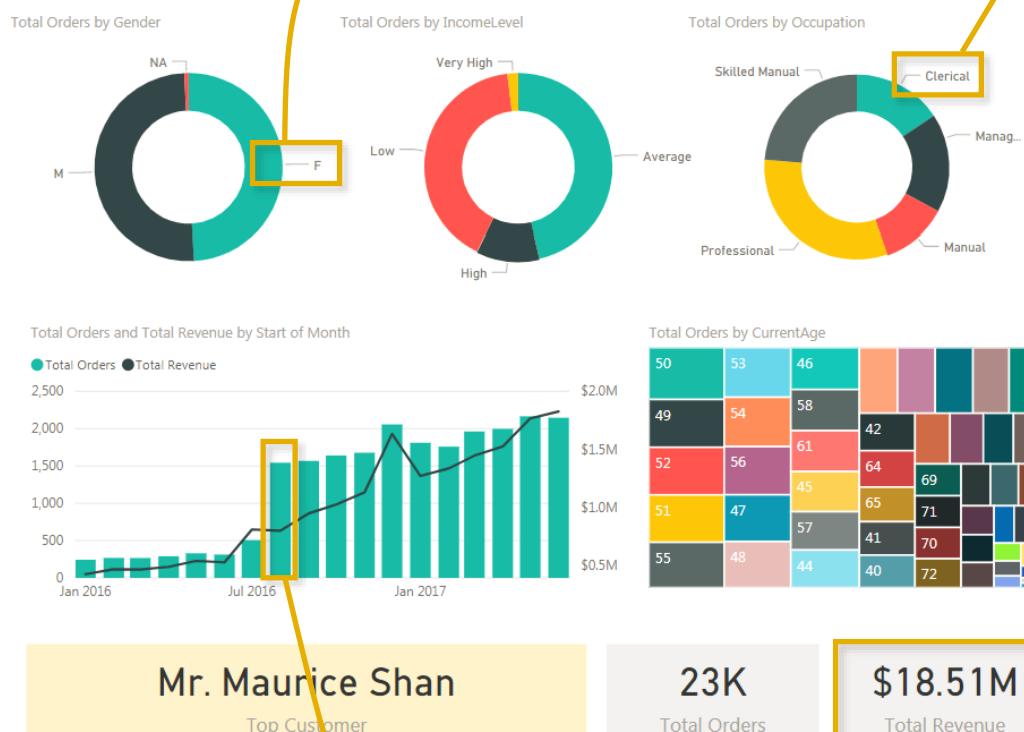
FILTER CONTEXT:

- *Calendar[Year]* = 2016 or 2017

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[Year]* = 2016 or 2017
- *Customers[Gender]* = F (Female)



MEASURE: Total Orders

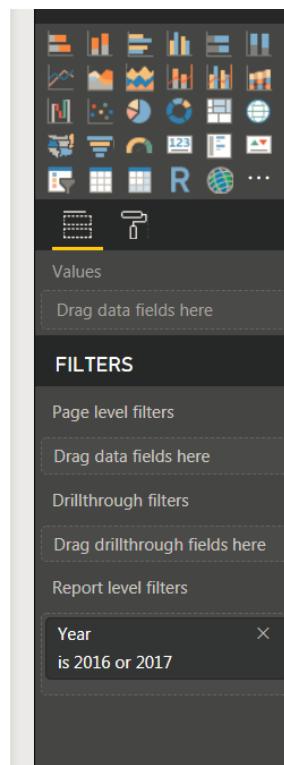
FILTER CONTEXT:

- *Calendar[Year]* = 2016 or 2017
- *Calendar[Month]* = August 2016

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[Year]* = 2016 or 2017
- *Customers[Occupation]* = Clerical



MEASURE: Total Revenue

FILTER CONTEXT:

- *Calendar[Year]* = 2016 or 2017

STEP-BY-STEP MEASURE CALCULATION

CategoryName	Total Returns
Accessories	1,115
Bikes	342
Clothing	267

How *exactly* is this measure calculated?

- **REMEMBER:** This all happens *instantly* behind the scenes, every time the filter context changes

STEP 1

Filter context is detected & applied



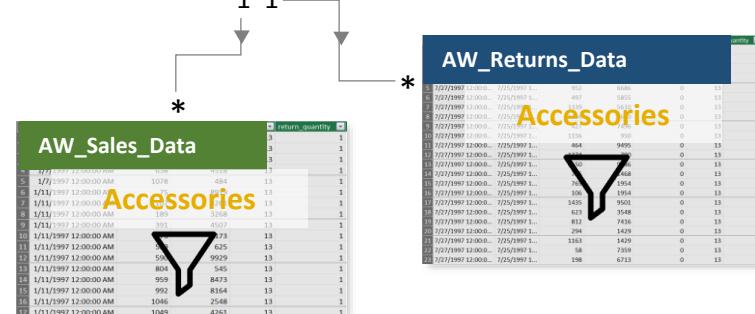
CategoryName	Total Returns
Accessories	1,115
Bikes	342
Clothing	267

Product[**CategoryName**] = “**Accessories**”



STEP 2

Filters flow “downstream” to all related tables



STEP 3

Measure formula evaluates against the filtered table



Total Returns = COUNTROWS(AW Returns Data)

Count of rows in the AW_Returns_Data table, filtered down to only rows where the product category is “Accessories”

= 1,115

COMMON DAX FUNCTIONS

DAX SYNTAX

MEASURE NAME

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

Total Quantity: =**SUM(Transactions[quantity])**

FUNCTION NAME

- Calculated columns don't always use functions, but measures do:
 - In a **Calculated Column**, **=Transactions[quantity]** returns the value from the quantity column in each row (*since it evaluates one row at a time*)
 - In a **Measure**, **=Transactions[quantity]** will return an **error** since Power BI doesn't know how to translate that as a single value (*you need some sort of aggregation*)

Referenced
TABLE NAME

Referenced
COLUMN NAME

Note: This is a “fully qualified” column, since it’s preceded 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]**

PRO TIP:

For **column** references, use the fully qualified name (i.e. **Table[Column]**)
For **measure** references, just use the measure name (i.e. **[Measure]**)



DAX OPERATORS

Arithmetic Operator	Meaning	Example
+	Addition	$2 + 7$
-	Subtraction	$5 - 3$
*	Multiplication	$2 * 6$
/	Division	$4 / 2$
\wedge	Exponent	$2 \wedge 5$

Pay attention to these!

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

*Head to www.microsoft.com for more information about DAX syntax, operators, troubleshooting, etc

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

**Note: This is NOT a comprehensive list (does not include trigonometry functions, parent/child functions, information functions, or other less common functions)*

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)

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)**

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

OR()

Checks whether one of the arguments is TRUE to return TRUE, and returns FALSE if both arguments are FALSE

=**OR(Logical1, Logical2)**

TEXT FUNCTIONS

LEN()	Returns the number of characters in a string	= LEN (Text)	<i>Note: Use the & operator as a shortcut, or to combine more than two strings!</i>
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])	

RELATED

RELATED()

Returns related values in each row of a table based on relationships with other tables

=RELATED(Column**Name**)



The column that contains the values you want to retrieve

Examples:

- Product_Lookup[ProductName]
- Territory_Lookup[Country]



HEY THIS IS IMPORTANT!

RELATED works almost *exactly* like a **VLOOKUP** function – it uses the relationship between tables (*defined by primary and foreign keys*) to pull values from one table into a new column of another. Since this function requires row context, it can only be used as a **calculated column** or as part of an **iterator function** that cycles through all rows in a table (*FILTER, SUMX, MAXX, etc*)

PRO TIP:

Avoid using **RELATED** to create redundant calculated columns unless you absolutely need them, since those extra columns increase file size; instead, use **RELATED** within a measure like **FILTER** or **SUMX**



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])

COUNT, COUNTA, DISTINCTCOUNT & COUNTROWS

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)

CALCULATE

CALCULATE()

Evaluates a given expression or formula under a set of defined filters

=CALCULATE(Expression, [Filter1], [Filter2],...)



Name of an existing measure, or a DAX formula for a valid measure

Examples:

- [Total Orders]
- SUM(Returns_Data[ReturnQuantity])



*List of simple Boolean (True/False) filter expressions
(note: these require simple, fixed values; you cannot create filters based on measures)*

Examples:

- Territory_Lookup[Country] = "USA"
- Calendar[Year] > 1998



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**"

CALCULATE (EXAMPLE)

```
X ✓ Bike Returns = CALCULATE([Total Returns], Products[CategoryName] = "Bikes") ▾
```

CategoryName	Total Returns	Bike Returns
Accessories	1,115	342
Bikes	342	342
Clothing	267	342
Components		342
Total	1,724	342

Here we've defined a new measure named "**Bike Returns**", which evaluates the "**Total Returns**" measure when the *CategoryName* in the **Products** table equals "**Bikes**"

*Wait, why do we see the **same repeating values** when we view a matrix with different categories on rows?*

*Shouldn't these cells have different filter contexts for **Accessories**, **Clothing**, **Components**, etc?*



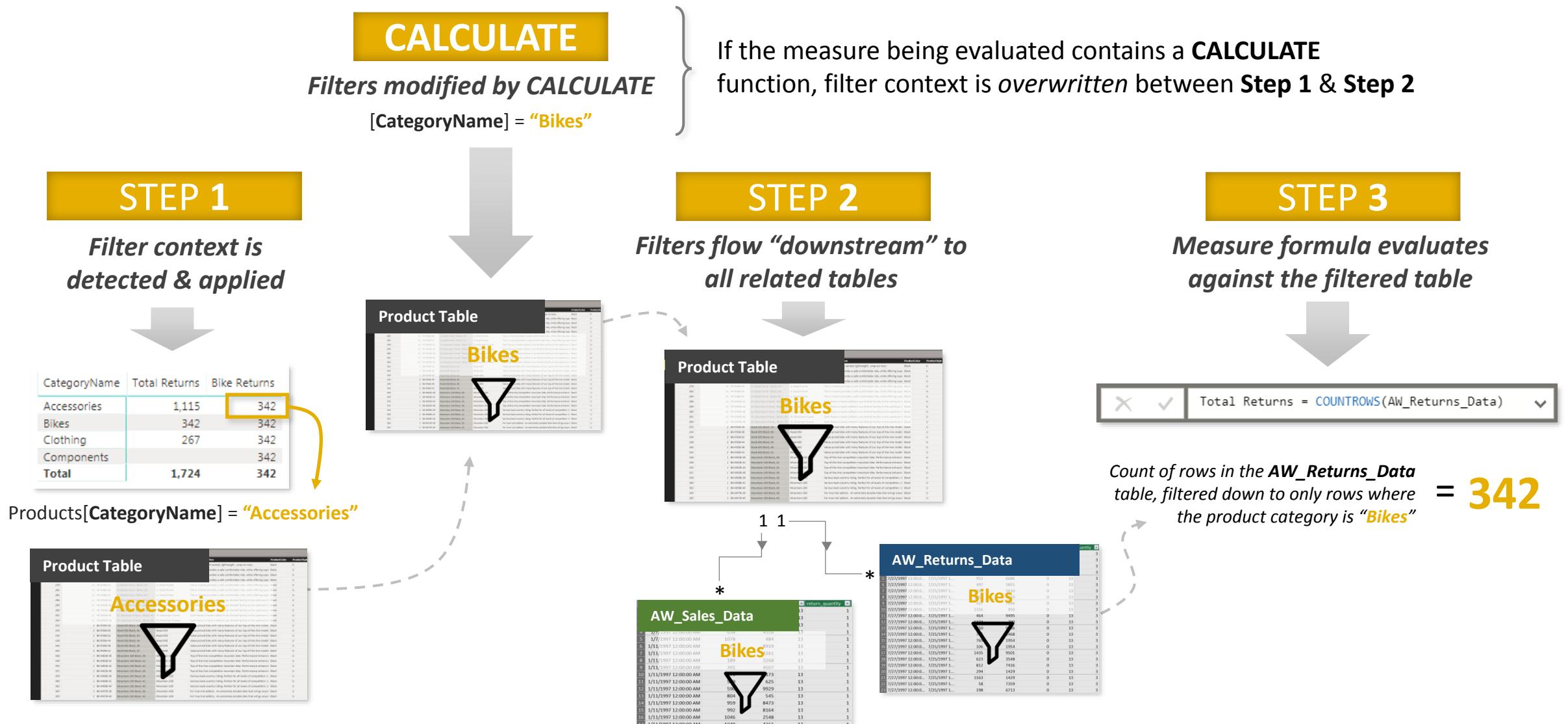
HEY THIS IS IMPORTANT!

CALCULATE **modifies** and **overrules** any competing filter context!

In this example, the "Clothing" row has filter context of CategoryName = "**Clothing**" (*defined by the row label*) **and** CategoryName= "**Bikes**" (*defined by the CALCULATE function*)

Both cannot be true at the same time, so the "**Clothing**" filter is overwritten and the "**Bikes**" filter (from CALCULATE) takes priority

CALCULATE CHANGES THE FILTER CONTEXT



ALL

ALL()

Returns all rows in a table, or all values in a column, ignoring any filters that have been applied

=ALL(Table or ColumnName, [ColumnName1], [ColumnName2],...)

The table or column that you want to clear filters on

Examples:

- Transactions
- Products[ProductCategory]

List of columns that you want to clear filters on (optional)

Notes:

- If your first parameter is a table, you can't specify additional columns
- All columns must include the table name, and come from the same table

Examples:

- Customer_Lookup[CustomerCity], Customer_Lookup[CustomerCountry]
- Products[ProductName]

PRO TIP:

Instead of adding filter context, ALL removes it. This is often used when you need unfiltered values that won't react to changes in filter context (i.e. % of Total, where the denominator needs to remain fixed)



FILTER

FILTER()

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

=FILTER(Table, FilterExpression)

Table to be filtered

Examples:

- Territory_Lookup
- Customer_Lookup

A Boolean (True/False) filter expression to be evaluated for each row of the table

Examples:

- Territory_Lookup[Country] = "USA"
- Calendar[Year] = 1998
- Products[Price] > [Overall Avg Price]

HEY THIS IS IMPORTANT!

FILTER is used to add new filter context, and can handle **more complex filter expressions** than CALCULATE (by referencing measures, for example)

Since FILTER returns an entire table, it's almost always used as an *input* to other functions, like CALCULATE or SUMX



PRO TIP:

Since FILTER iterates through each row in a table, it can be slow and processor-intensive; don't use FILTER if a CALCULATE function will accomplish the same thing



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]



PRO TIP:

Imagine the function **adding a temporary new column** to the table, calculating the value in each row (based on the expression) and then applying the aggregation to that new column (like SUMPRODUCT)

*In this example we’re looking at **SUMX**, but other “X” functions follow a similar syntax

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**))

Running
Total

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

*Select an interval (DAY, MONTH, QUARTER, or YEAR) and the
of intervals to compare (i.e. previous month, rolling 10-day)*



PRO TIP:

To calculate a *moving average*, use the running total calculation above and divide by the number of intervals

BEST PRACTICES: CALCULATED COLUMNS & MEASURES



Don't use a calculated column when a measure will do the trick

- *Only use calculated columns to “stamp” static, fixed values to each row in a table*
- *Use measures when aggregation is necessary, or to create dynamic values in a report*



Write measures for even the simplest calculations (i.e. Sum of Sales)

- *Once you create a measure it can be used anywhere in the report and as an input to other, more complex calculations (no implicit measures!)*



Break measures down into simple, component parts

- *DAX is a difficult language to master; focus on practicing and understanding simple components at first, then assemble them into more advanced formulas*



Reference columns with the table name, and measures alone

- *Using “fully qualified” column references (preceded by the table name) helps make formulas more readable and intuitive, and differentiates them from measure references*

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*



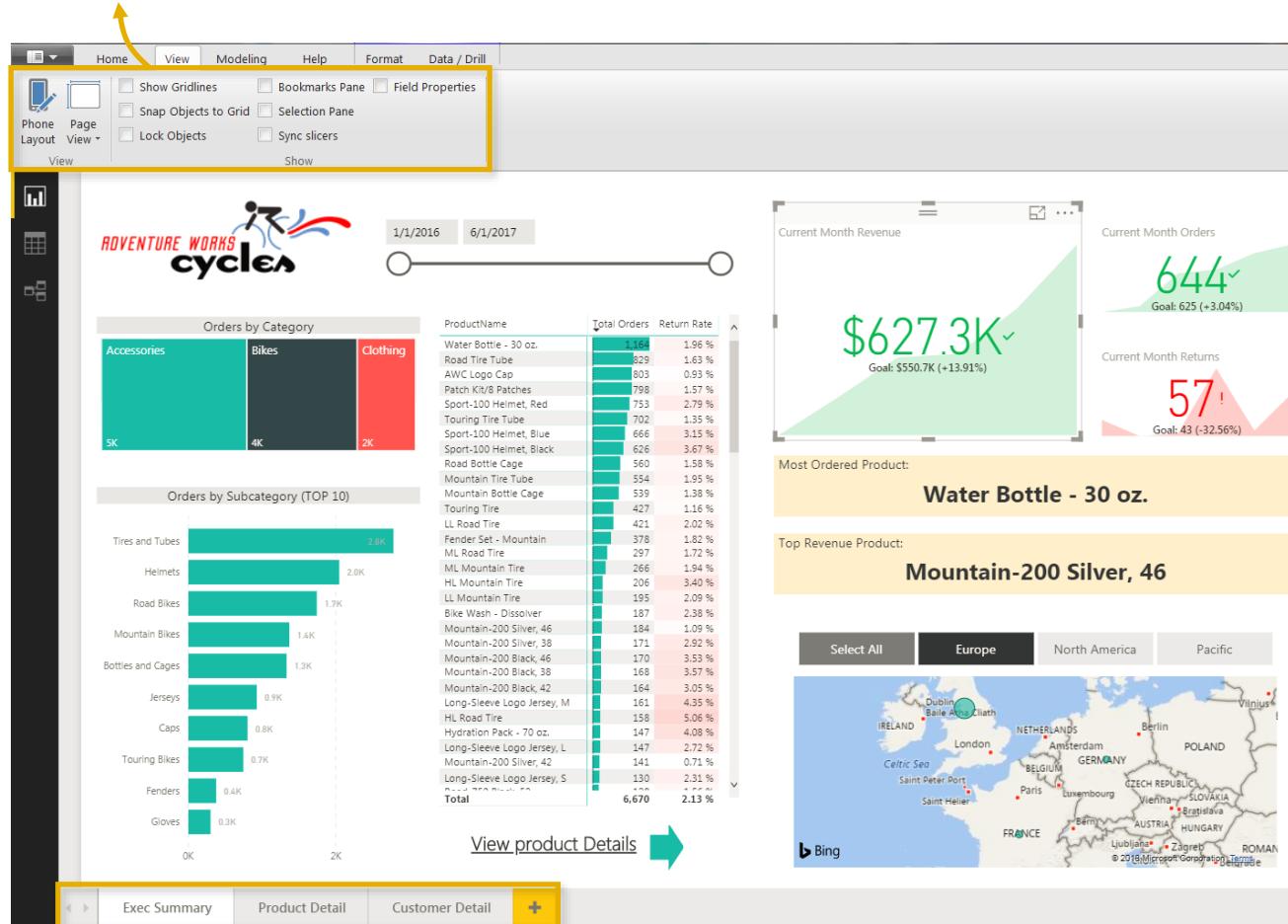
Minimize iterator functions (FILTER, SUMX, etc.)

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

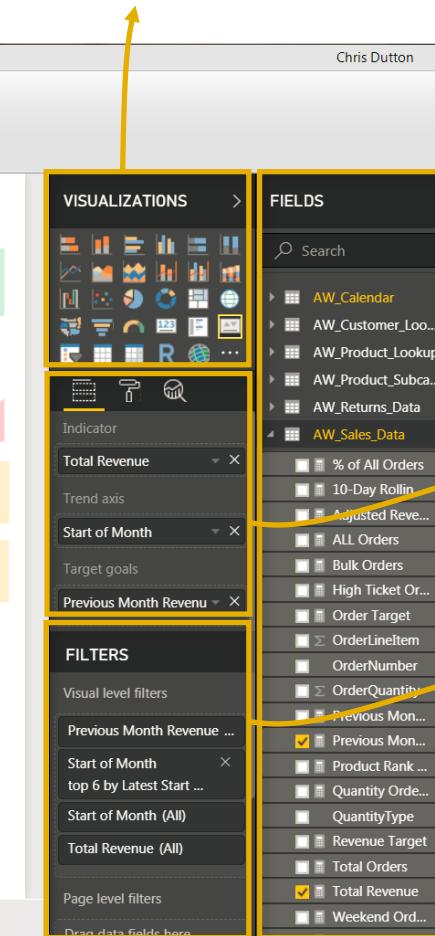
BUILDING REPORTS

THE POWER BI REPORT VIEW

"View" Options (Layout, Gridlines, Snap to Grid, Bookmarks/Selection Pane, etc)



Visualization Options (Charts, Slicers, Maps, Matrices, etc)



Field List (Tables, columns & measures)

Fields/Format/Analytics Pane (Fields pane shown here)

Filters Pane (Visual-Level, Page-Level, Report-Level, and Drillthrough Filters)

Report Pages (Similar to Excel tabs; each is a blank reporting canvas)

INSERTING OBJECTS & BASIC CHARTS

The screenshot shows the Power BI desktop interface. On the left, the **Visualizations** pane is open, displaying various chart icons. The **FIELDS** pane is also open, showing a list of fields including **AW_Calendar**, **AW_Customer_Loo...**, **AW_Product_Lookup**, **AW_Product_Subca...**, **AW_Returns_Data**, and **AW_Sales_Data**. A filter for **Year** is applied, showing options like **is 2016 or 2017**. In the center, a bar chart is displayed with the title **ALL Orders** and a value of **30K**. On the right, the **Home** tab of the ribbon is selected, with the **Insert** tab highlighted. The **Insert** tab contains options for **New Page**, **New Visual**, **Buttons**, **Text box**, **Image**, and **Shapes**. Arrows from the text descriptions point to the corresponding sections in the interface.

Click on a visualization type or use the “New Visual” option in the **Home tab to insert a blank chart template (usually a column chart by default)**

Note: You can also add **New Pages**, **Buttons**, **Text Boxes**, **Images** and **Shapes** from this menu

Drag fields or measures directly into the report canvas to automatically generate a new visual

FORMATTING OPTIONS

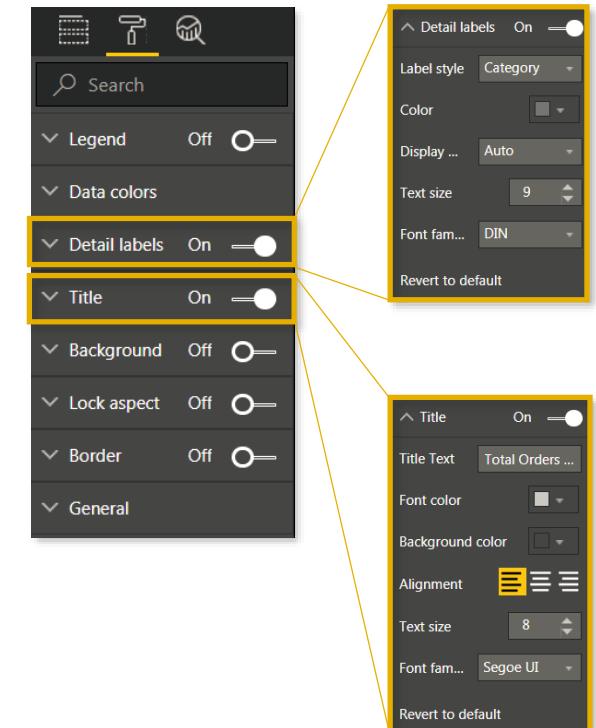
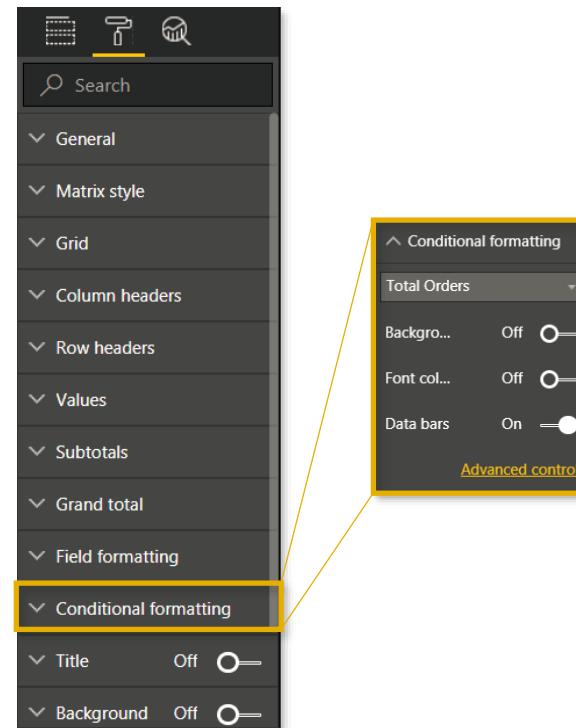
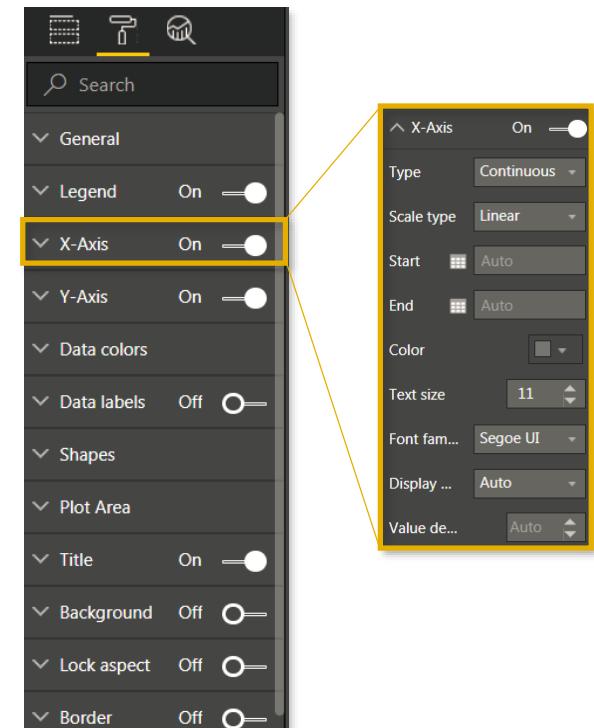
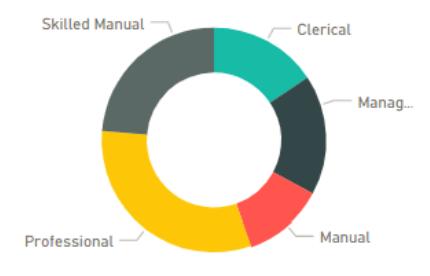
Example: Line & Column Chart



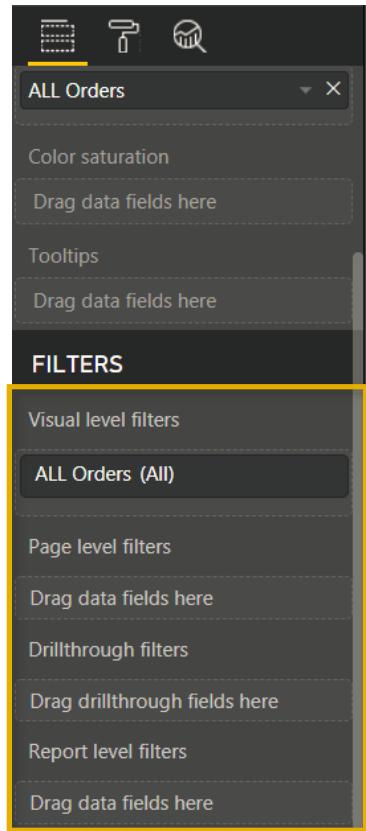
Example: Matrix

ProductName	Total Orders	Return Rate
Water Bottle - 30 oz.	1,164	1.96 %
Road Tire Tube	829	1.63 %
AWC Logo Cap	803	0.93 %
Patch Kit/8 Patches	798	1.57 %
Sport-100 Helmet, Red	753	2.79 %
Touring Tire Tube	702	1.35 %
Sport-100 Helmet, Blue	666	3.15 %
Sport-100 Helmet, Black	626	3.67 %
Road Bottle Cage	560	1.58 %
Mountain Tire Tube	554	1.95 %
Mountain Bottle Cage	539	1.38 %

Example: Donut Chart



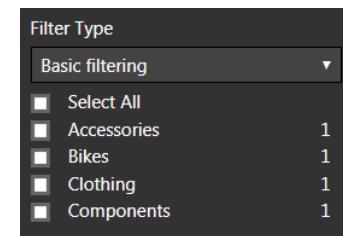
FILTERING OPTIONS



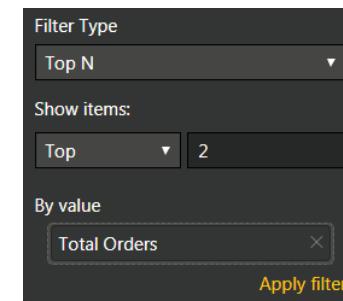
There are **four (x4)** primary filter types in Power BI reports:

1. **Visual Level:** Applies only to the *specific visual* in which it is defined
2. **Page Level:** Applies to *all visuals on the specific page* in which it is defined
3. **Report Level:** Applies to *all visuals* across *all pages* of the report
4. **Drillthrough:** Applies to *specific pages*, and *dynamically changes* based on user paths

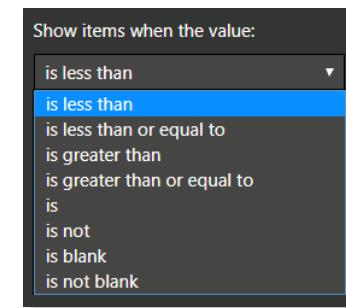
Filter settings include Basic, Advanced, and Top N options



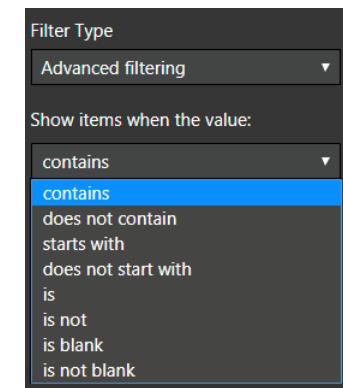
Basic Options



Top N Options



Advanced (Values)



Advanced (Text)

EDITING REPORT INTERACTIONS

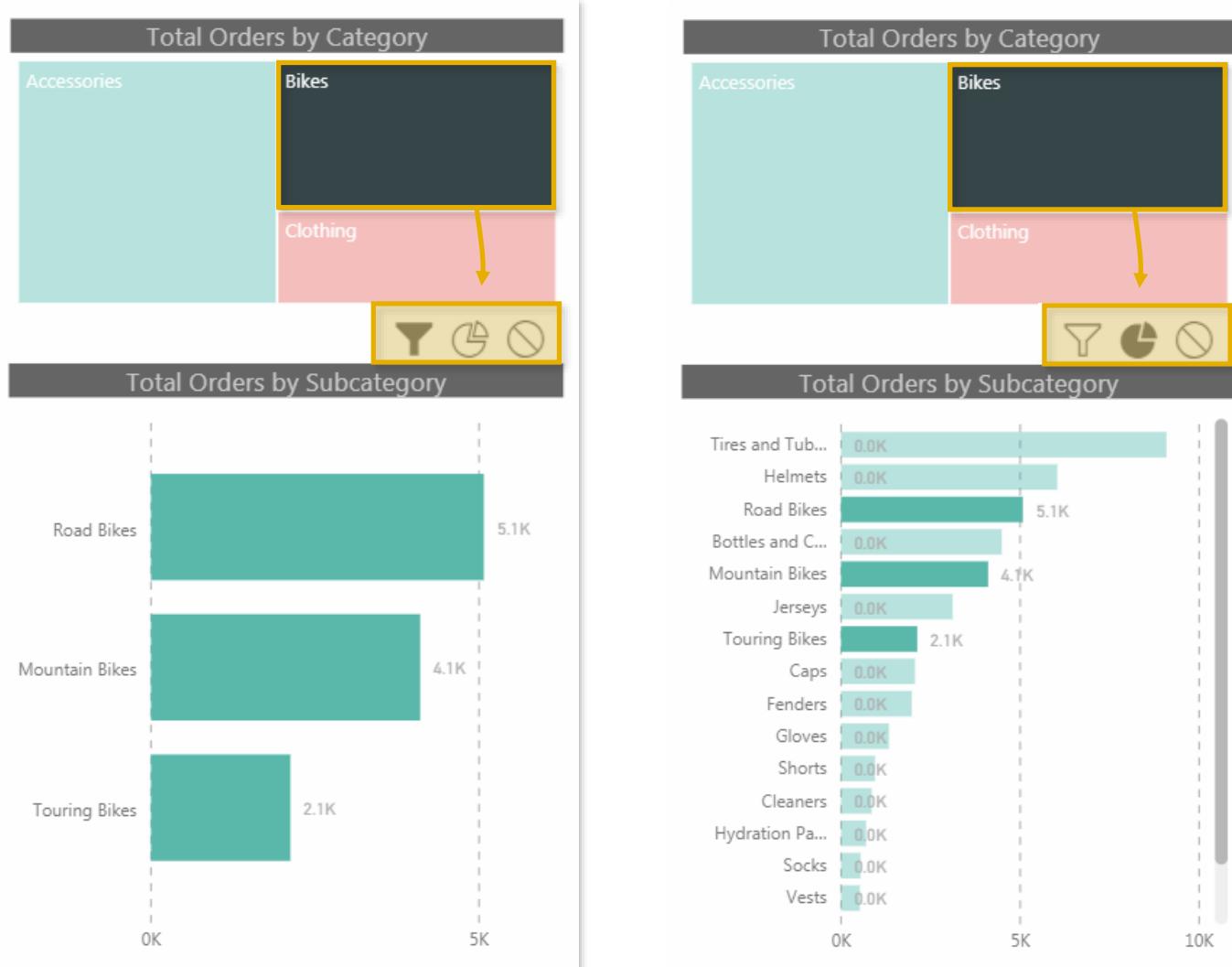
The screenshot illustrates a Power BI report interface with various visualizations and interaction controls. At the top, a ribbon menu includes 'Format' and 'Data / Drill' tabs. A yellow box highlights the 'Edit interactions' button under the 'Format' tab. Below the ribbon, a 'Drilling filters other visuals' section shows a hierarchy of visual elements. A yellow arrow points from the 'Edit interactions' button to a Timeline visual showing dates '1/1/1998' and '12/31/1998'. Another yellow arrow points from the Timeline to a Product matrix table. The matrix has columns for 'Product Brand' (Hermanos, High Top, Tri-State, Tell Tale, National, Horatio, Best Choice, Big Time, High Quality, Denny, Red Wing, Cormorant) and rows for 'Quantity', 'Retail Price', 'Product Cost', and 'Profit'. Total values at the bottom are 31,670, \$2.13, \$0.85, and \$40,634 respectively. To the right of the matrix is a 'SELECT COUNTRY:' dropdown with options for 'CANADA', 'MEXICO', and 'USA', each with a filter icon. Below the matrix are three teal-colored cards displaying '\$5,798 MTD Profit', '\$17K OTD Profit', and '\$64K YTD Profit', each also with a filter icon. On the far right is a map of the Pacific Northwest showing locations like Vancouver, Victoria, Bellingham, and Mount Vernon. The map includes labels for 'Vancouver Island', 'Salish Sea', 'North Cascades N.P.', and 'Okean N.P.'. A Bing logo is visible in the bottom left corner of the map area.

Product Brand	Quantity	Retail Price	Product Cost	Profit
Hermanos	1,385	\$2.30	\$0.95	\$1,898
High Top	1,415	\$2.13	\$0.85	\$1,811
Tri-State	1,428	\$2.12	\$0.87	\$1,760
Tell Tale	1,374	\$2.18	\$0.92	\$1,727
National	1,195	\$2.27	\$0.90	\$1,674
Horatio	1,173	\$2.28	\$0.94	\$1,601
Best Choice	1,130	\$2.31	\$0.91	\$1,601
Big Time	1,139	\$2.18	\$0.87	\$1,523
High Quality	1,048	\$2.41	\$0.97	\$1,485
Denny	998	\$2.46	\$1.03	\$1,458
Red Wing	1,101	\$2.25	\$0.92	\$1,436
Cormorant	1,065	\$2.23	\$0.86	\$1,427
Total	31,670	\$2.13	\$0.85	\$40,634

Report interactions allow you to determine how filters applied to *one* visual impact the *others*

- For example, by selecting the Timeline visual and enabling “Edit interactions” from the **Format** tab, we can manually determine which visuals should “react” when the date range changes
- In this case the **Product matrix**, **Country slicer** and **Map** will filter in response to timeline changes (), but the **MTD**, **QTD**, and **YTD Profit** cards will not ()

EDITING REPORT INTERACTIONS (CONT.)



For certain types of visuals, a third option allows you to “highlight” sub-segments of the data, rather than simply filtering vs. not filtering

- When the interaction mode is set to “filter”, selecting the “**Bikes**” category in the treemap produces a filtered list of subcategories in the chart below
- When the interaction mode is set to “highlight”, selecting the “**Bikes**” category in the treemap highlights the relevant subsegment of data in the chart below

DRILLTHROUGH FILTERS

The screenshot displays a Power BI report interface. On the left, there's a dashboard with three donut charts: 'Current Month Orders vs. Target' (232), 'Current Month Revenue vs. Target' (\$4.03K), and 'Current Month Returns vs. Previous Month' (6). Below these are two line charts: 'Total Revenue by Week' (green line) and 'Total Returns by Week' (red line), both spanning from July 2016 to May 2017. A central sidebar contains 'VISUALIZATIONS' and 'FILTERS' sections. The 'FILTERS' section shows a 'Drillthrough filter' for 'ProductName' set to 'is AWC Logo Cap'. On the right, a detailed table titled 'Product Detail' lists various products with their total orders and return rates. The row for 'AWC Logo Cap' is highlighted, and a context menu is open over it, showing options like 'Drill Up', 'Show Data', and 'Drillthrough'. A yellow arrow points from the 'Drillthrough' option in the context menu to the 'Drillthrough filter' in the sidebar.

ProductName	Total Orders	Return Rate
Water Bottle - 30 oz.	1,164	1.96 %
Road Tire Tube	829	1.63 %
AWC Logo Cap	803	0.03 %
Patch Kit/8 Patches	554	1.95 %
Sport-100 Helmet, Red	539	1.38 %
Touring Tire Tube	427	1.16 %
Sport-100 Helmet, Blue	421	2.02 %
Road Bottle Cage	378	1.82 %
Mountain Tire Tube	297	1.72 %
Mountain Bottle Cage	266	1.94 %
Touring Tire	7,380	2.17 %
LL Road Tire		
Fender Set - Mountain		
ML Road Tire		
ML Mountain Tire		
Total		

Drillthrough filters allow users to jump to different report pages (*like bookmarks*), while simultaneously filtering based on the *specific item selected*

- Here we've built a report page ("Product Detail") featuring *product-level* performance, and added a Drillthrough filter for **ProductName**; users can now right-click any report visual containing product names, and jump straight to a pre-filtered version of this page ("AWC Logo Cap" shown in the example above)

ADDING & LINKING BOOKMARKS

The screenshot shows a Power BI desktop interface. At the top, the ribbon has 'Bookmarks Pane' checked. Below the ribbon is a 'Phone Layout' button. The main area contains a dashboard with a map of the Pacific Northwest, a table of product sales, and three summary cards for profit: MTD Profit (\$168), QTD Profit (\$432), and YTD Profit (\$1,460). To the right is a 'BOOKMARKS' pane with a list of bookmarks: USA Topline View, 1998 USA Gold Member..., Q4 Portland Sales (which is highlighted with a yellow box), and 1H Priority Canada SKUs.

On a new page, we present our key insights, insert buttons, and link them to bookmarks using the object “**Action**” properties

Now we’re able to create a *narrative* from the data, and really bring our insights to life!

KEY INSIGHTS & RECOMMENDATIONS

- Gold member sales in the US reached \$46,000 this year, up 17% Y-o-Y**
Recommendation: Continue to promote key membership benefits and broaden test markets in Q2
- Portland sales were strong in Q4, led by High Top, Ebony, and Red Wing brands**
Recommendation: Shift product stock away from Tri-State and Carrington to support high-margin products
- In the first half of the year, Canada sales to priority customers drove profits of only \$1,726**
Recommendation: Launch paid media support across Canada markets targeted to high-priority lookalike audiences

In this example, we notice that Q4 sales were particularly strong in the Portland market, so we add a new **bookmark** (*View > Bookmarks Pane > Add*) and name it “**Q4 Portland Sales**”

VISUALIZATIONS

BOOKMARKS

Action: On

Type: Bookmark

Bookmark: Q4 Portland Sales

“WHAT-IF” PARAMETERS

The screenshot shows the Power BI desktop interface. On the left, a 'What-if parameter' dialog box is open, containing fields for Name (Price Adjustment %), Data type (Decimal number), Minimum (-1), Maximum (1), Increment (0.1), and Default (0). A checkbox for 'Add slicer to this page' is checked. At the bottom are 'OK' and 'Cancel' buttons. A yellow box highlights the 'What If' button in the ribbon, and another yellow box highlights the 'Price Adjustment (%)' input field in the dialog. An arrow points from the 'What If' button to the 'Price Adjustment (%)' field. To the right, a chart titled 'Total Revenue by Week' displays weekly revenue from July 2016 to May 2017. A vertical line marks the date '2/26/2017'. A callout box at the bottom right of the chart provides specific data for that date: Total Revenue \$915.84, Adjusted Revenue \$1,373.76, and Total Orders 52.

What-if parameter

Name
Price Adjustment %

Data type
Decimal number

Minimum
-1

Maximum
1

Increment
0.1

Default
0

Add slicer to this page

OK Cancel

Price Adjustment (%)
0.50

\$8.64 Avg Retail Price
\$12.96 Adjusted Price

Total Revenue by Week

2/26/2017

Total Revenue \$915.84
Adjusted Revenue \$1,373.76
Total Orders 52

“What If” Parameters are essentially pre-set measures that produce values within a given range, based on user-inputs (*data type, min/max, increment, and default*)

These can be great tools for forecasting or scenario testing; here we’ve created a **“Price Adjustment %”** parameter in order to compare **Total Revenue** (based on the *actual price*) against **Adjusted Revenue** (based on the *parameter-adjusted* price)

NOTE: When you create a parameter, a new table is automatically added with DAX calculations for “Parameter” and “Parameter Value”, which look something like this:

Parameter = `GENERATESERIES(-1, 1, 0.1)`

Parameter Value = `SELECTEDVALUE(Parameter[Parameter], 0)`

MANAGING & VIEWING AS ROLES

The screenshot illustrates the process of managing and viewing roles in Power BI. It shows the Power BI ribbon at the top, the 'Manage roles' dialog, a 'View as roles' dialog, and a sample report.

Manage roles dialog: Shows a list of roles (Europe, North America, Pacific) and tables. A DAX filter expression is applied to the 'Europe' role: `[Continent] = "Europe"`. A note below says: "Filter the data that this role can see by entering a DAX filter expression that returns a True/False value. For example: [Entity ID] = "Value"'.

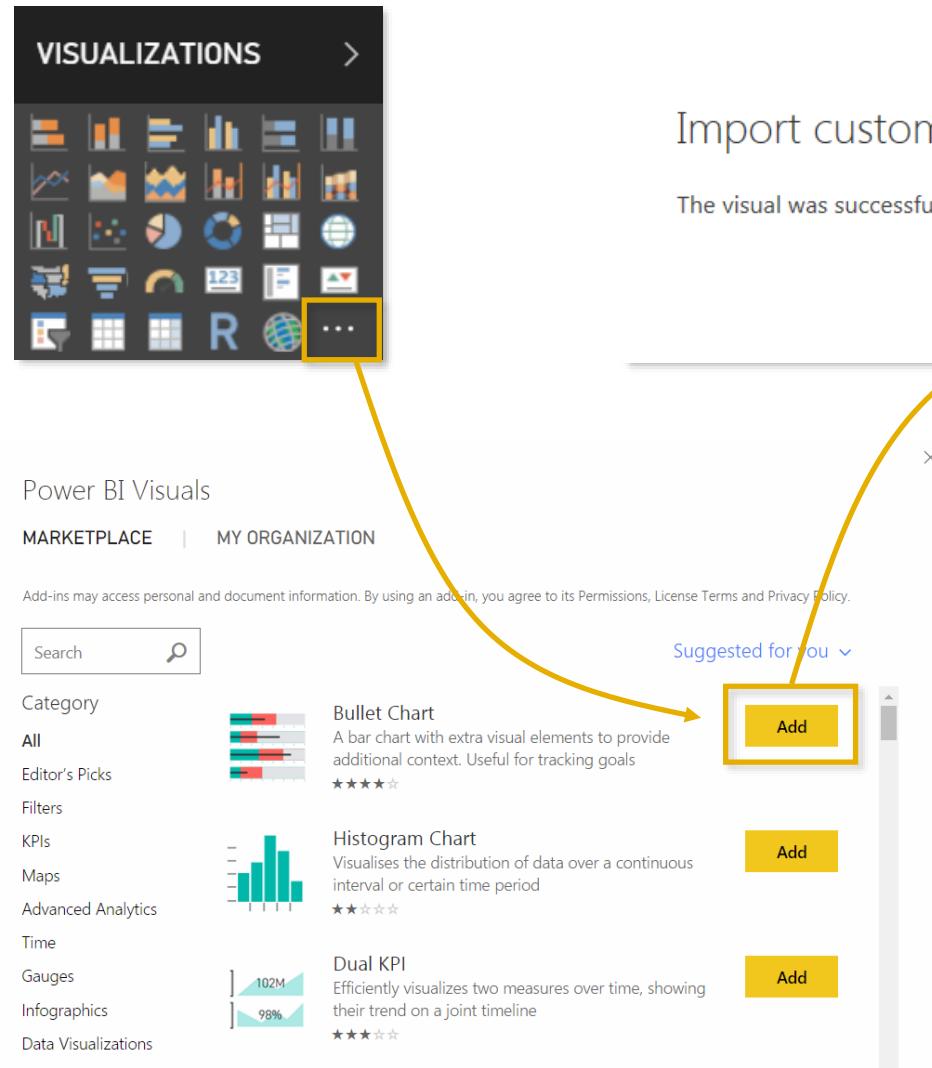
View as roles dialog: Shows checkboxes for viewing as Europe (checked), North America, and Pacific.

Report View: Shows a report for Adventure Works Cycles. The status bar indicates "Now viewing report as: Europe". The report includes a chart of Orders by Category (Accessories, Bikes, Clothing), a table of ProductName, Total Orders, and Return Rate, and various summary metrics like Current Month Revenue (\$627.3K), Current Month Orders (644), and Current Month Returns (57). A map at the bottom highlights the 'Europe' region.

Text: Roles allow you to define filtered views that can be tailored to specific audiences

- In this example we've created views for each territory manager (**Europe, North America, and Pacific**) based on simple DAX filter statements

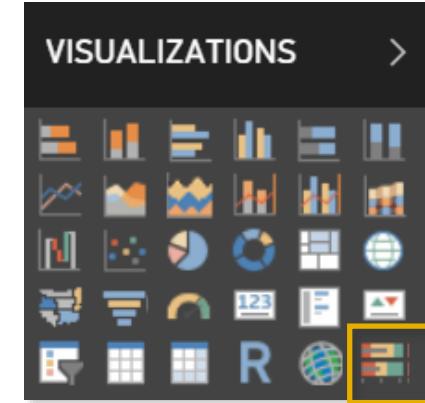
IMPORTING CUSTOM VISUALS



Import custom visual

The visual was successfully imported into this report.

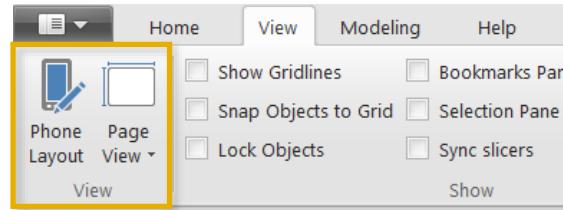
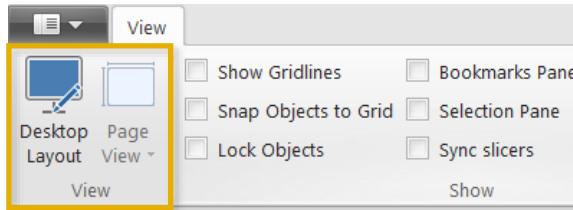
OK



Click the ellipsis in the visuals pane to import **custom visuals** from files or from the Power BI Marketplace, directly into the report (*no installation or restart necessary*)

In this case we've added a **Bullet Chart** from the marketplace

DESKTOP VS. PHONE LAYOUT



The image shows two side-by-side screenshots of the Power BI interface. Both screenshots feature a 'View' ribbon at the top with several options: Show Gridlines, Bookmarks Pane, Snap Objects to Grid, Selection Pane, Lock Objects, Sync slicers, and a 'Show' button. The left screenshot, labeled 'Desktop Layout', has the 'Desktop Layout' icon highlighted with a yellow box. The right screenshot, labeled 'Phone Layout', has the 'Phone Layout' icon highlighted with a yellow box. Below the ribbon, both screens display a complex dashboard for 'Adventure Works Cycles'. The desktop version shows a map of Europe, while the phone version shows a simplified version of the same map. Both versions include a large green bar chart for 'Current Month Revenue' (\$627.3K), a red bar chart for 'Current Month Returns' (57), and a stacked bar chart for 'Orders by Category' (Accessories, Bikes, Clothing). A sidebar on the right contains sections for 'VISUALIZATIONS', 'FILTERS', and 'VALUES'.

Phone Layout view allows you to design on a canvas size optimized for mobile viewing (vs. desktop)

- **NOTE:** You can't actually build content within the Phone Layout view; recommend building in **Desktop Layout**, and assembling select visuals for mobile if you plan to share content via the Power BI app

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)*

WRAPPING UP

Week Starting
11/2/2014
11/9/2014
11/14/2014
11/21/2014
11/28/2014
weekly A
14/20
12/01

\$10,000 100,000 400 0.40%
\$12,000 125,000 600 0.48%
\$9,000 112,000 440 0.37%
\$11,000 135,000 360 0.30%
\$8,000 105,000 320 0.30%
-27.3% -22.2% -11.1% +4.3%
1000 1000 1000 1000
1000 1000 1000 1000

1.12 1.44 1.16
1.35 0.98 1.14
1.08 1.19 1.08
1.14 1.02 1.05
1.05 1.05 1.00
0.91 1.08 0.97
0.99 0.95 0.91
0.96 0.86 0.82
0.75 0.88 0.77
0.66 1.09 0.75
0.58 0.58 0.75
1.10

RESOURCES & NEXT STEPS



Looking to become an absolute **Excel ROCK STAR?** Try the full stack:

- *Microsoft Excel – Advanced Excel Formulas & Functions*
- *Microsoft Excel – Data Analysis with Excel PivotTables*
- *Microsoft Excel – Data Viz with Excel Charts & Graphs*
- *Microsoft Excel – Intro to Power Query, Power Pivot & DAX*



Remember to check out these **helpful resources** for additional support:

- docs.microsoft.com/en-us/power-bi/sample-datasets for free sample datasets
- powerbi.microsoft.com/blog for great blog posts and tutorials from Microsoft
- pbiusergroup.com for helpful forums and local meetup groups
- **Microsoft Power BI** and **Guy in a Cube** YouTube channels for demos and advanced tutorials



Ratings and reviews mean the world to me, so I'd love your feedback

- *Please reach out if there's anything I can do to improve your experience along the way – I'm here to help!*



THANK YOU