



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

Power BI-ს გადმოწერა და პარამეტრების დაყენება

Power BI-ს ინტერფეისი და სამუშაო სივრცე

მონაცემთა შეერთების ტიპები

Query Editor

Power BI Desktop ვერსიის ინსტალაცია

<https://www.microsoft.com/en-us/download/details.aspx?id=58494>

Power BI Desktop

Microsoft Power BI Desktop is built for the analyst. It combines state-of-the-art interactive visualizations, with industry-leading data query and modeling built-in. Create and publish your reports to Power BI. Power BI Desktop helps you empower others with timely critical insights, anytime, anywhere.

Important! Selecting a language below will dynamically change the complete page content to that language.

Select language

English

Download

Choose the download you want

File Name

PBIDesktopSetup.exe

Size

405.4
MB

PBIDesktopSetup_x64.exe

446.2
MB

[Download](#)

Total size: 446.2 MB

Power BI Desktop

 Get data

 Recent sources

 AdventureWorks_Dashboard_WIP_Se...

C:\Users\Chris\Documents\SecondLens...

 AdventureWorks_Dashboard_WIP_Se...

C:\Users\Chris\Documents\SecondLens...

 SDA_Call_Appointment_Report_4-4-2...

C:\Users\Chris\Documents\SecondLens...

 SDA_Call_Appointment_Report_3-27-...

C:\Users\Chris\Documents\SecondLens...

 Open other reports



Sign in to collaborate and share content

Power BI Pro enables you to collaborate across departments and distribute content to your organization. Sign in or sign up for a free 60-day trial.

Try free

Sign in

WHAT'S NEW

Take a look at what's new and improved in Power BI in this month's update.

POWER BI BLOG

Keep up to date with the latest news, resources, and updates from the Power BI team.

FORUMS

Visit the Power BI Forum to ask questions or interact with other users in the Power BI community.

TUTORIALS

Ready to learn more about Power BI?

მნიშვნელოვანია:

- Power BI Desktop-ზე წვდომისთვის არ გჭირდებათ შესვლა ან რეგისტრაცია Power BI Pro ანგარიშზე.
- შესვლა საჭიროა მხოლოდ Power BI სერვისის საშუალებით [რეპორტის](http://app.powerbi.com) გაზიარებისთვის

რა არის Power BI?

- Power BI არის Microsoft-ის Business Intelligence-ის პროდუქტი, რომელიც მოიცავს როგორც დესკტოპის, ისე ვებ-ზე დაფუძნებულ სერვისის აპლიკაციას, მონაცემთა ჩატვირთვის, მოდელირებისა და ვიზუალიზაციისთვის.

რატომ Power BI?

Power BI-ის
საშუალებით
შესაძლებელია :

მონაცემთა
მილიონობი
თ მწკრივის
დაკავშირება,
გარდაქმნა
და
გააანალიზებ
ა

რელაციური
მოდელების
შექმნა
რამდენიმე
წყაროდან
მონაცემების
დაკავშირები
სთვის

რთული
გამოთვლები
DAX-ის
გამოყენებით

მონაცემთა
ვიზუალიზა
ცია
ინტერაქტიუ
ლი
რეპორტები
თ და
Dashboard-
ებით

THE POWER BI ინტერფეისი

Three Core Views:

Report



Data



Relationships
(aka Model)

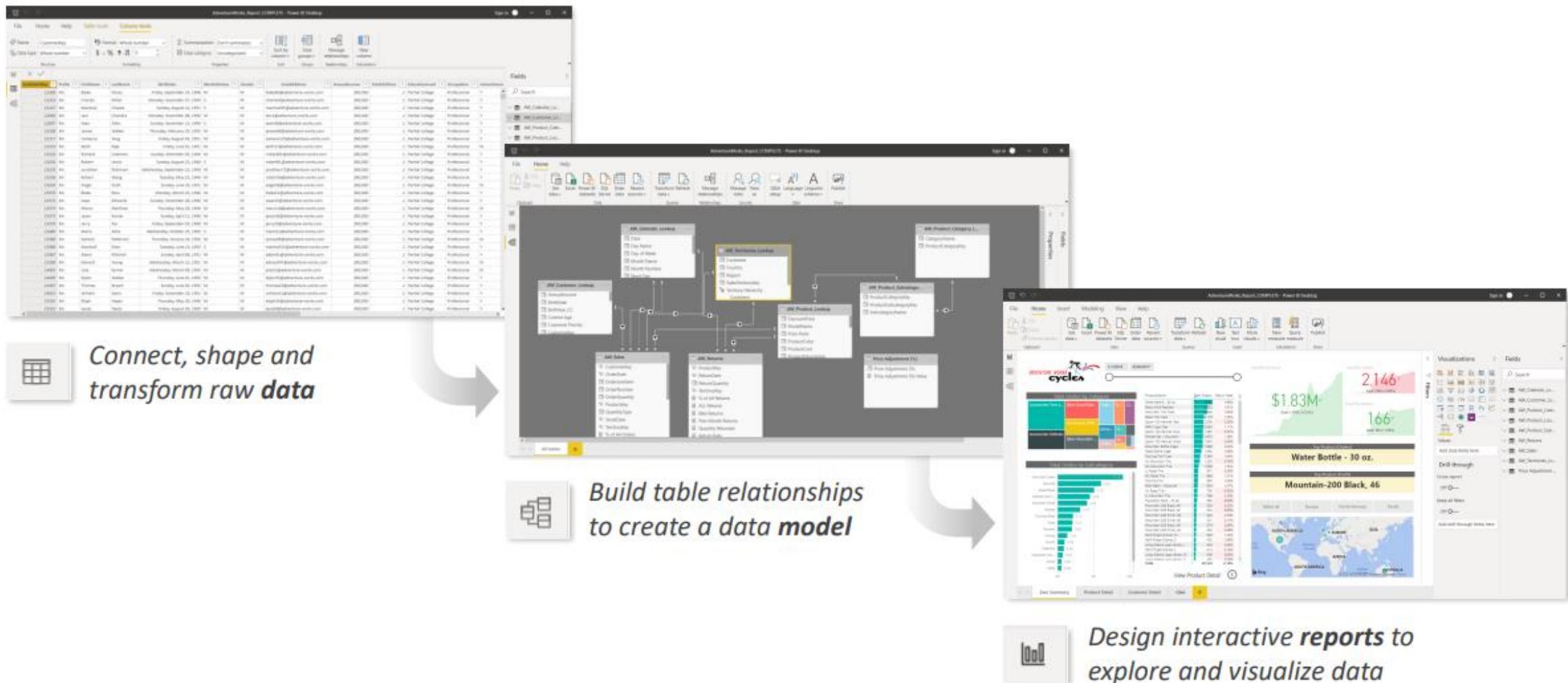


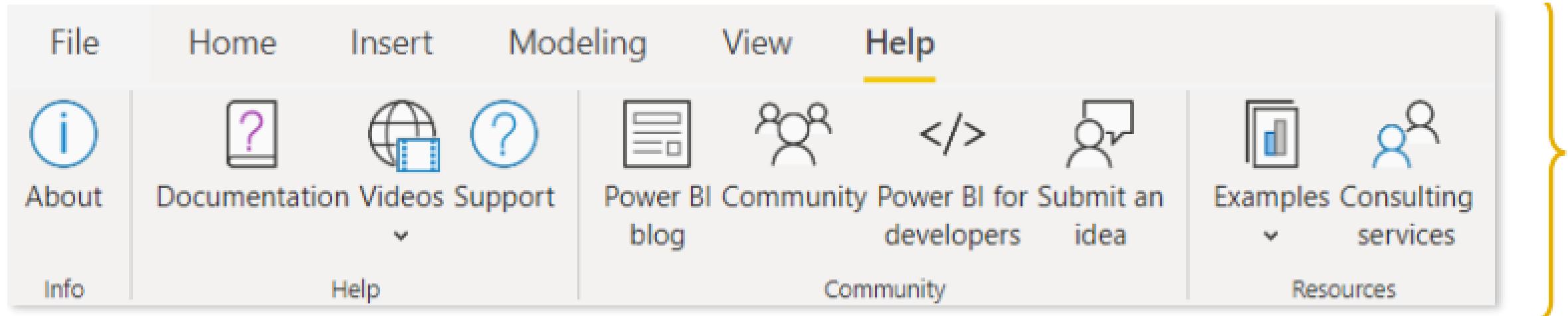
The screenshot displays the Microsoft Power BI desktop application interface. On the left, a vertical navigation bar lists the 'Three Core Views': Report, Data, and Relationships (aka Model). The 'Report' view is currently selected, indicated by a yellow box around its icon. The main workspace shows a dashboard for 'ADVENTURE WORKS cycles' with the following components:

- Report View:** Displays a 'Total Orders by Category' treemap and a 'Total Orders by Subcategory' bar chart.
- Data View:** Shows a 'Monthly Revenue' table with columns for ProductName, Total Orders, and Return Rate, and a 'Monthly Returns' table with columns for ProductName, Total Returns, and Return Rate.
- Relationships View:** A world map showing regional distribution with labels for North America, Europe, Asia, South America, Africa, and Australia.

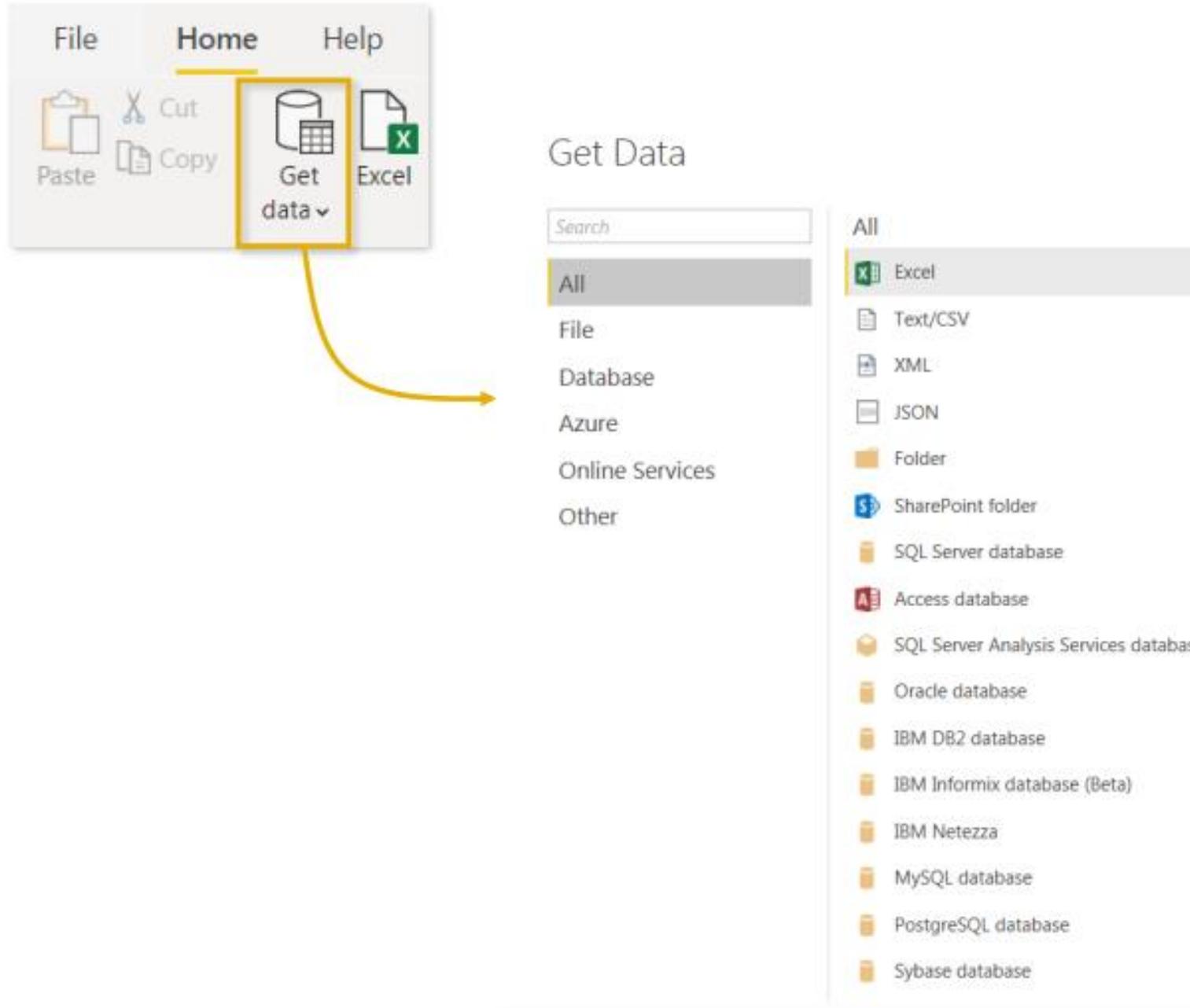
The ribbon at the top includes tabs for Clipboard, Data, Queries, Insert, Calculations, Share, and a date range selector from 1/1/2016 to 6/30/2017. The bottom of the interface shows the 'Visual Details' ribbon and the 'Bing' logo.

POWER BI-OB WORKFLOW





“Help” ტაბი მოიცავს დოკუმენტაციას, ტრენინგების ვიდეოებს, ნიმუშის ფაილებს და ლინკებს რომელიც დაგეხმარებათ გაიგოს ყველაფერი Power BI Desktop ვერსიის შესახებ



მონაცემთა შეერთების ტიპები

- Power BI-ს შეუძლია დაუკავშირდეს თითქმის ნებისმიერი ტიპის მონაცემთა წყაროს, მათ შორის :
- მონაცემთა ბაზებს (SQL, Access, Oracle, IBM, Azure და ა.შ.)
- ონლაინ სერვისებს (SharePoint, GitHub, Dynamics 365, Google Analytics, Salesforce და ა.შ.)
- ფაილებს და ფოლდერებს (csv, ტექსტი, xls და ა.შ.)
- სხვა (ვებ არხები, R სკრიპტები, Spark, Hadoop და ა.შ.)



Formula Bar
(this is "M" code)

**Query
Pane**

Query Editing Tools (Table transformations, calculated columns, etc)

Table Name & Properties

Applied Steps (like a macro)

| CustomerKey | Prefix | FirstName | LastName | BirthDate |
|-------------|--------|-----------|----------|------------|
| 11000 | Mr. | Jon | Yang | 4/8/1966 |
| 11001 | Mr. | Eugene | Huang | 5/14/1965 |
| 11002 | Mr. | Ruben | Torres | 8/12/1965 |
| 11003 | Ms. | Christy | Zhu | 2/15/1968 |
| 11004 | Mrs. | Elizabeth | Johnson | 8/8/1968 |
| 11005 | Mr. | Julio | Ruiz | 8/5/1965 |
| 11007 | Mr. | Marco | Mehta | 5/9/1964 |
| 11008 | Mrs. | Robin | Verhoff | 7/7/1964 |
| 11009 | Mr. | Shannon | Carlson | 4/1/1964 |
| 11010 | Ms. | Jacquelyn | Suarez | 2/6/1964 |
| 11011 | Mr. | Curtis | Lu | 11/4/1963 |
| 11012 | Mrs. | Lauren | Walker | 1/18/1968 |
| 11013 | Mr. | Ian | Jenkins | 8/6/1968 |
| 11014 | Mrs. | Sydney | Bennett | 5/9/1968 |
| 11015 | Ms. | Chloe | Young | 2/27/1979 |
| 11016 | Mr. | Wyatt | Hill | 4/28/1979 |
| 11017 | Mrs. | Shannon | Wang | 6/26/1944 |
| 11018 | Mr. | Clarence | Rai | 10/9/1944 |
| 11019 | Mr. | Luke | Lai | 3/7/1978 |
| 11020 | Mr. | Jordan | King | 9/20/1978 |
| 11021 | Ms. | Destiny | Wilson | 9/3/1978 |
| 11022 | Mr. | Ethan | Zhang | 10/12/1978 |

Query Editor

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

მონაცემთა დაკავშირება და ფორმირება

ცხრილის ძირითადი გარდაქმნები

ტექსტურ და რიცხვით მონაცემებთან მუშაობა, თარილის მონაცემებთან მუშაობა

მოძრავი კალენდრის შექმნა, ინდექსისა და პირობითი სვეტის შექმნა, მონაცემთა დაჯგუფება და ჯამური მონაცემების კალკულაციები

Pivot და Unpivot ბრძანება, Merge Queries ბრძანება



Query Editing Tools (Table transformations, calculated columns, etc)

Formula Bar
(this is "M" code)

**Query
Pane**

The screenshot shows the Power Query Editor interface. The main area displays a table of customer data with columns: CustomerKey, Prefix, FirstName, LastName, and BirthDate. The 'Query Settings' pane on the right shows the table is named 'AW_Customer_Lookup' and lists the applied steps, which include removing columns and filtering rows.

| CustomerKey | Prefix | FirstName | LastName | BirthDate |
|-------------|--------|-----------|----------|------------|
| 11000 | Mr. | Jon | Yang | 4/8/1966 |
| 11001 | Mr. | Eugene | Huang | 5/14/1965 |
| 11002 | Mr. | Ruben | Torres | 8/12/1965 |
| 11003 | Ms. | Christy | Zhu | 2/15/1968 |
| 11004 | Mrs. | Elizabeth | Johnson | 8/8/1968 |
| 11005 | Mr. | Julio | Ruiz | 8/5/1965 |
| 11007 | Mr. | Marco | Mehta | 5/9/1964 |
| 11008 | Mrs. | Robin | Verhoff | 7/7/1964 |
| 11009 | Mr. | Shannon | Carlson | 4/1/1964 |
| 11010 | Ms. | Jacquelyn | Suarez | 2/6/1964 |
| 11011 | Mr. | Curtis | Lu | 11/4/1963 |
| 11012 | Mrs. | Lauren | Walker | 1/18/1968 |
| 11013 | Mr. | Ian | Jenkins | 8/6/1968 |
| 11014 | Mrs. | Sydney | Bennett | 5/9/1968 |
| 11015 | Ms. | Chloe | Young | 2/27/1979 |
| 11016 | Mr. | Wyatt | Hill | 4/28/1979 |
| 11017 | Mrs. | Shannon | Wang | 6/26/1944 |
| 11018 | Mr. | Clarence | Rai | 10/9/1944 |
| 11019 | Mr. | Luke | Lai | 3/7/1978 |
| 11020 | Mr. | Jordan | King | 9/20/1978 |
| 11021 | Ms. | Destiny | Wilson | 9/3/1978 |
| 11022 | Mr. | Ethan | Zhang | 10/12/1978 |

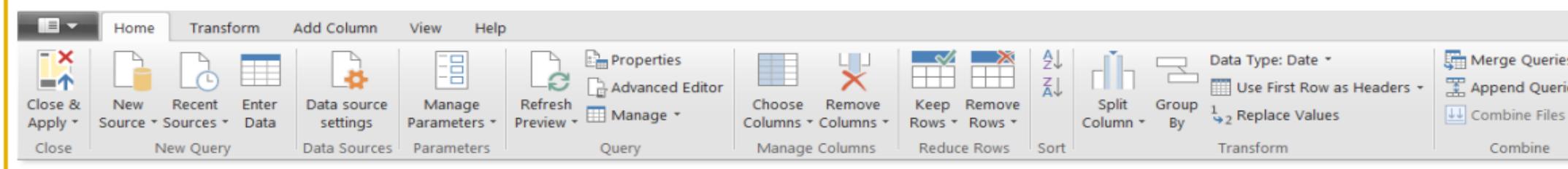
**Table Name
& Properties**

**Applied Steps
(like a macro)**

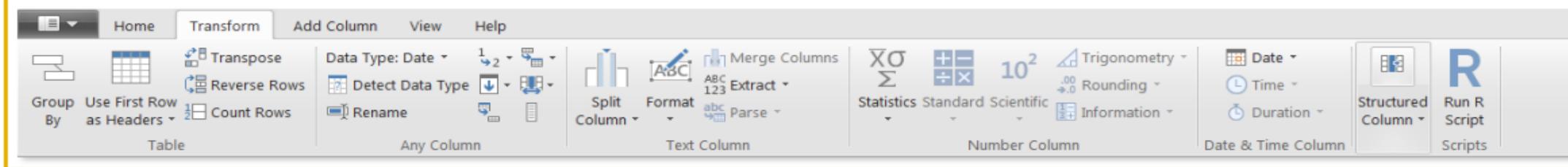
Query Editor

Query Editing Tools

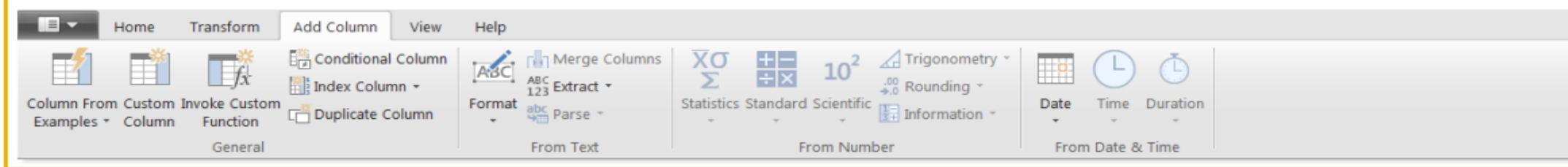
*The **HOME** tab includes **general settings** and **common table transformation tools***



*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)*



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

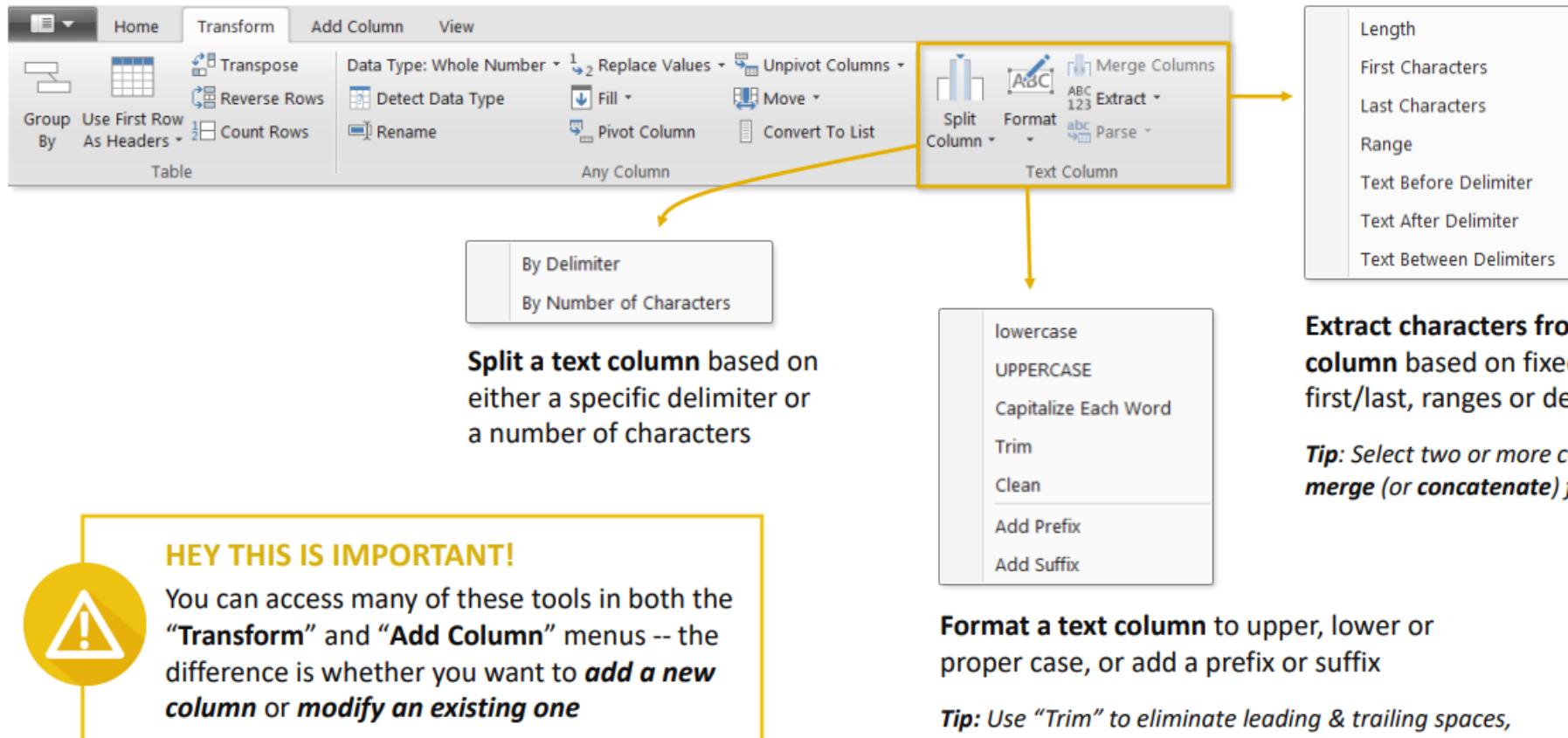
Tip: use the “Remove Duplicates” option to create a new lookup table from scratch

Duplicate, move & rename columns

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

- ძირითადი ტრანსფორმაციები

TEXT-SPECIFIC TOOLS



The screenshot shows the Power BI ribbon with the 'Transform' tab selected. A context menu is open over a 'Text Column', with several options highlighted in yellow: 'Split Column', 'Format', and 'Text Column'. A callout box points to 'Split Column' with the sub-options 'By Delimiter' and 'By Number of Characters'. Another callout box points to 'Format' with the sub-options 'lowercase', 'UPPERCASE', 'Capitalize Each Word', 'Trim', 'Clean', 'Add Prefix', and 'Add Suffix'. A third callout box points to the 'Text Column' option with a list of text manipulation functions: 'Length', 'First Characters', 'Last Characters', 'Range', 'Text Before Delimiter', 'Text After Delimiter', and 'Text Between Delimiters'. A yellow box with a warning icon contains the text: '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**'.

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

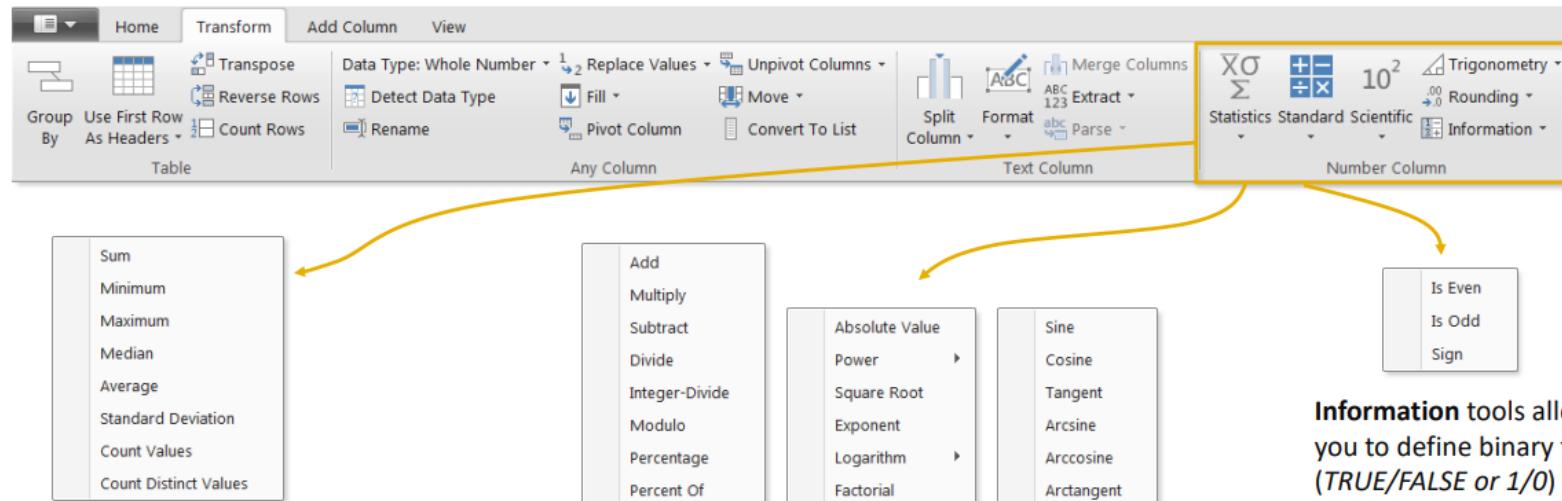
Format a text column to upper, lower or proper case, or add a prefix or suffix

Extract characters from column based on fixed first/last, ranges or del

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

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

NUMBER-SPECIFIC TOOLS



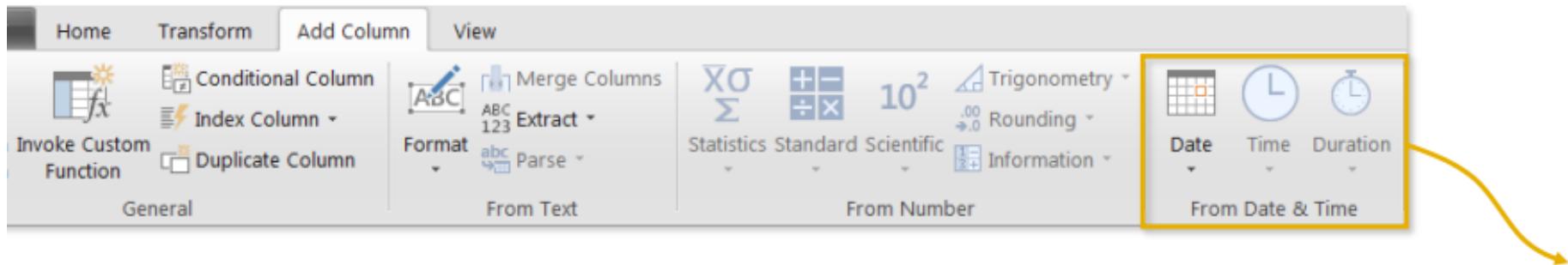
Statistics functions allow you to evaluate basic stats for the selected column (sum, min/max, average, count, countdistinct, etc)

Note: These tools return a **SINGLE** value,

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

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

DATE-SPECIFIC TOOLS

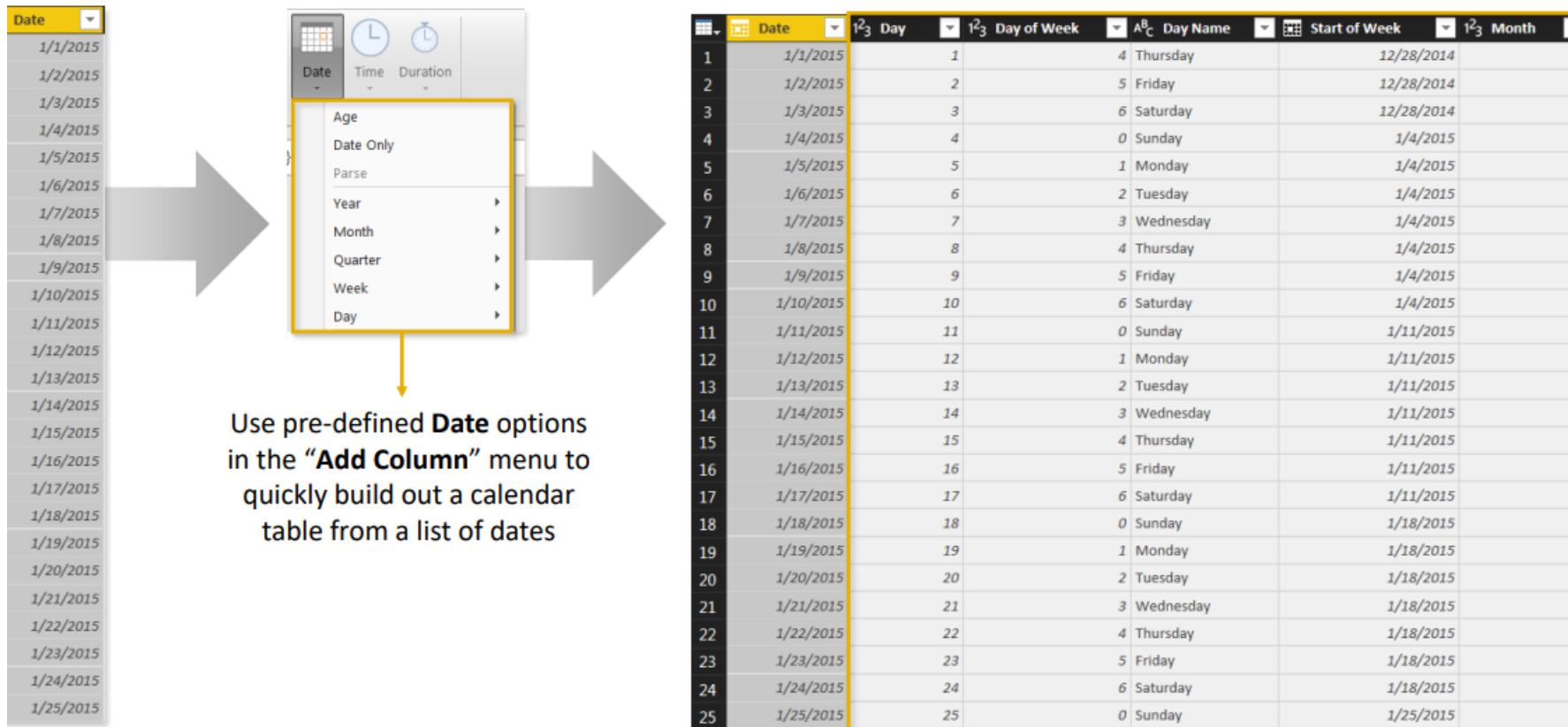


Date & Time tools are relatively straight-forward, and include the following options:

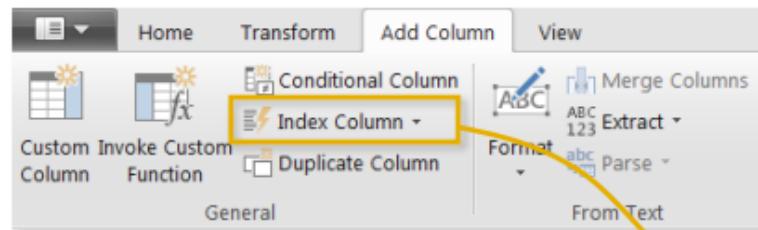
- **Age:** Difference between the current time and the date in each row
- **Date Only:** Removes the time component of a date/time field
- **Year/Month/Quarter/Week/Day:** Extracts individual components from a date field
(Time-specific options include Hour, Minute, Second, etc.)
- **Earliest/Latest:**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

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

კალენდრის შექმნა



ADDING INDEX COLUMNS

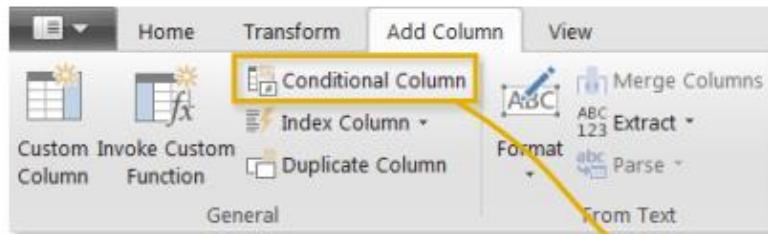


Index Columns contain a list of sequential values that can be used to identify each unique row in a table (typically starting from 0 or 1)

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 |

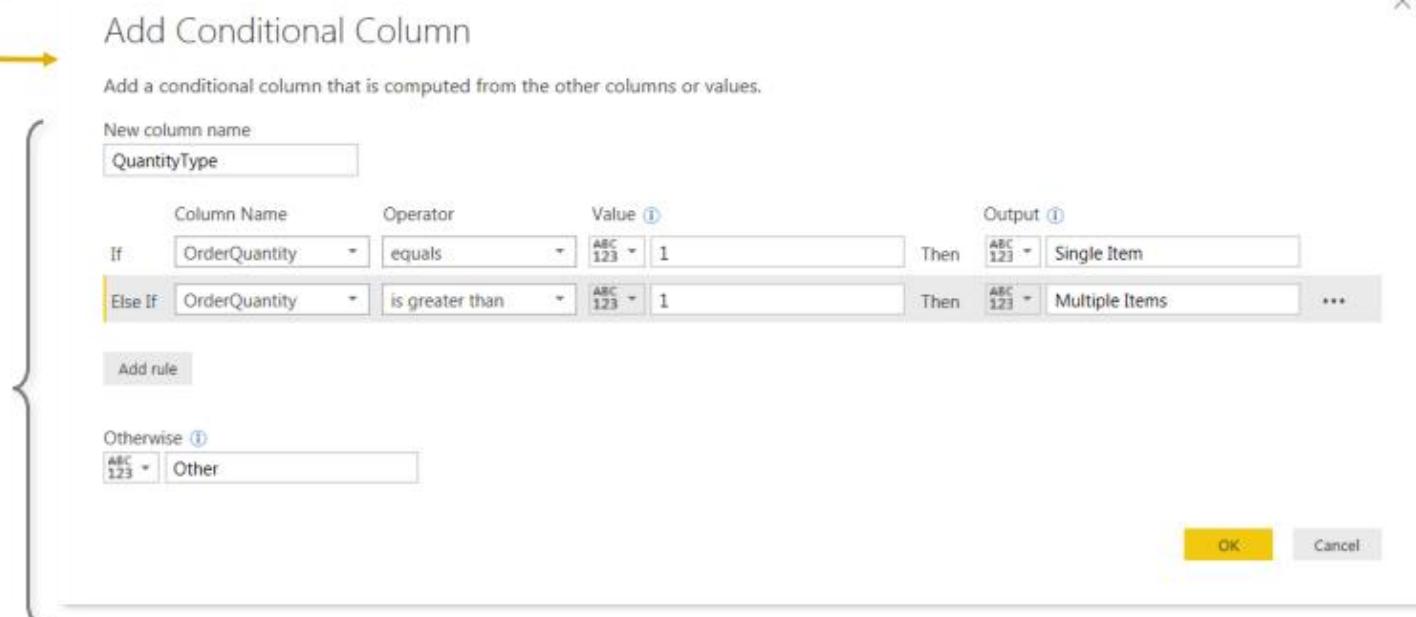
ADDING CONDITIONAL COLUMNS



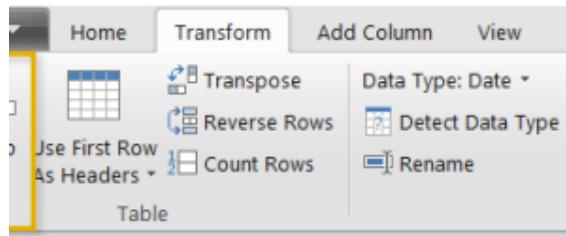
Conditional Columns allow you to define new fields based on logical rules and conditions (*IF/THEN statements*)

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"*

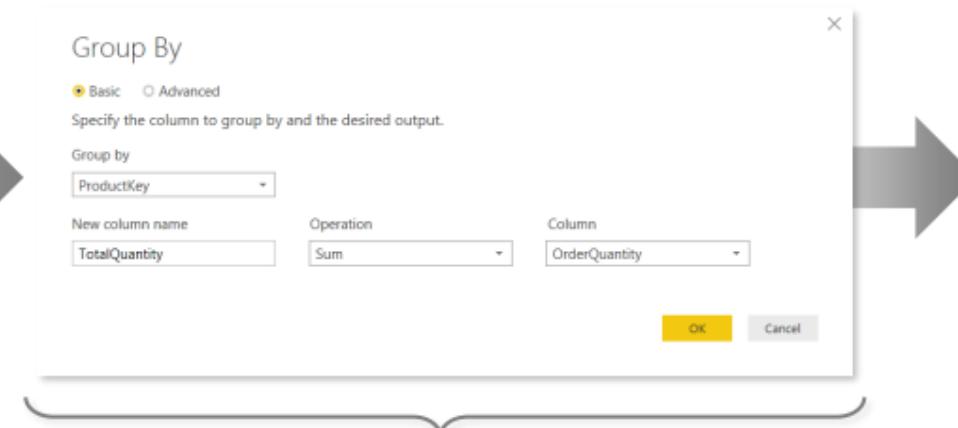


GROUPING & AGGREGATING DATA



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 |
|------------|------------|-------------|---------------|
| 6/25/2017 | 214 | 14719 | 1 |
| 7/16/2016 | 214 | 11243 | 1 |
| 12/31/2016 | 214 | 21452 | 1 |
| 6/29/2017 | 214 | 22748 | 1 |
| 10/6/2016 | 214 | 25025 | 1 |
| 10/7/2016 | 214 | 16504 | 1 |
| 10/13/2016 | 214 | 13043 | 1 |
| 1/19/2017 | 214 | 23101 | 1 |
| 9/7/2016 | 214 | 24900 | 1 |
| 1/19/2017 | 214 | 24196 | 1 |
| 6/29/2017 | 214 | 12963 | 1 |
| 11/6/2016 | 214 | 14570 | 1 |
| 11/13/2016 | 214 | 16999 | 1 |
| 7/31/2016 | 214 | 12281 | 1 |
| 10/9/2016 | 214 | 15685 | 1 |
| 8/1/2016 | 214 | 16982 | 1 |
| 12/4/2016 | 214 | 12835 | 1 |

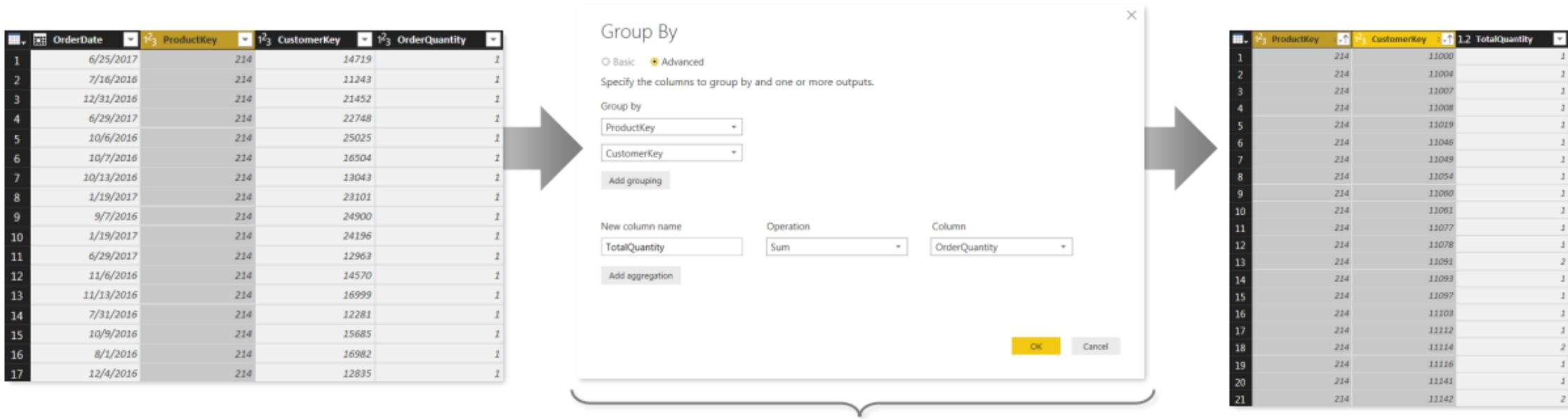


| | ProductKey | TotalQuantity |
|----|------------|---------------|
| 1 | 214 | 2099 |
| 2 | 217 | 1940 |
| 3 | 222 | 1995 |
| 4 | 225 | 4151 |
| 5 | 228 | 392 |
| 6 | 231 | 408 |
| 7 | 234 | 424 |
| 8 | 237 | 381 |
| 9 | 310 | 169 |
| 10 | 311 | 139 |
| 11 | 312 | 179 |
| 12 | 313 | 168 |
| 13 | 314 | 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*”)

PIVOT

UNPIVOT

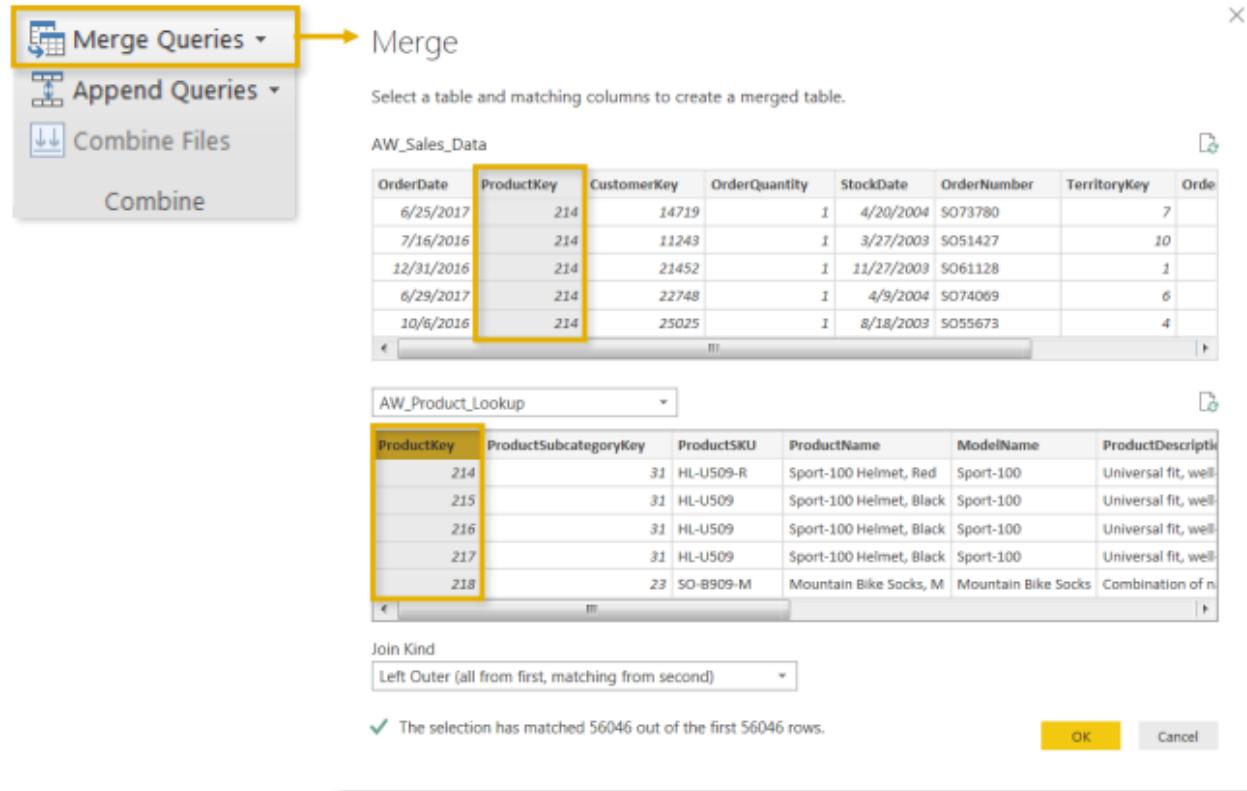
| | 1.2 Unit Sales |
|---|----------------|
| 1 | 286322 |
| 2 | 253787 |
| 3 | 155483 |
| 4 | 246491 |
| 5 | 130602 |

| | 1.2 1994 | 1.2 1995 | 1.2 1996 | 1.2 1997 | 1.2 1998 |
|---|----------|----------|----------|----------|----------|
| 1 | 286322 | 253787 | 155483 | 246491 | 130602 |

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



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!*)

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

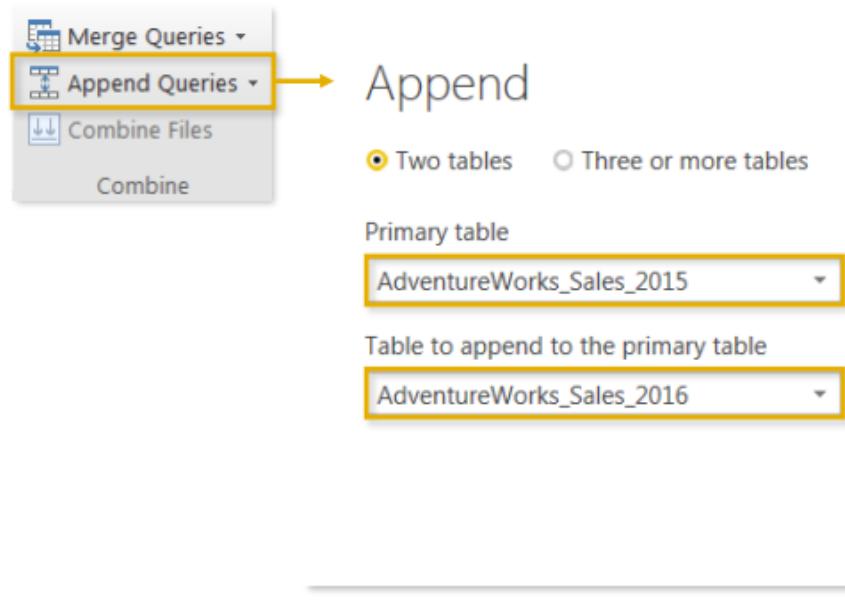
Append Queries ბრძანება, Append ბრძანების ფაილზე მიბმა;

Query-ს განახლების პარამეტრები; დამატებითი მონაცემთა ტიპები და კატეგორიები;

იერარქიის განსაზღვრა; ექსელის ფაილის იმპორტირება

მონაცემთა კავშირის საუკეთესო პრაქტიკა

APPENDING QUERIES



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

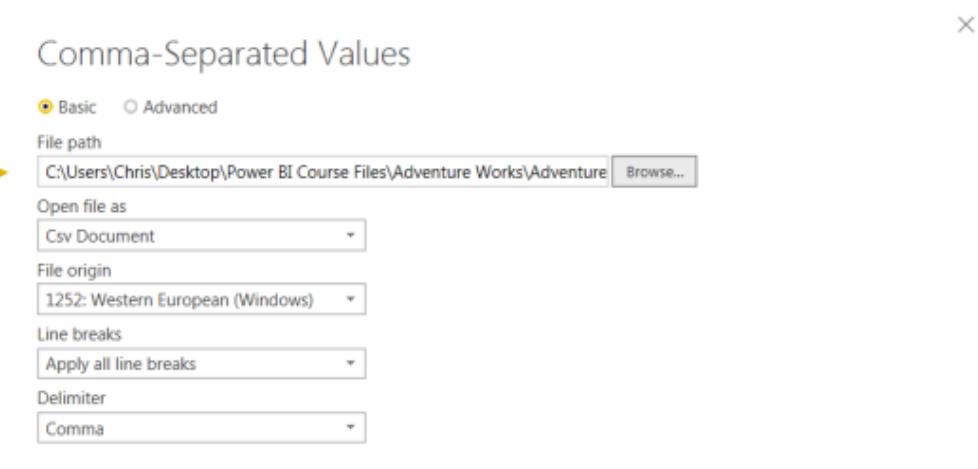
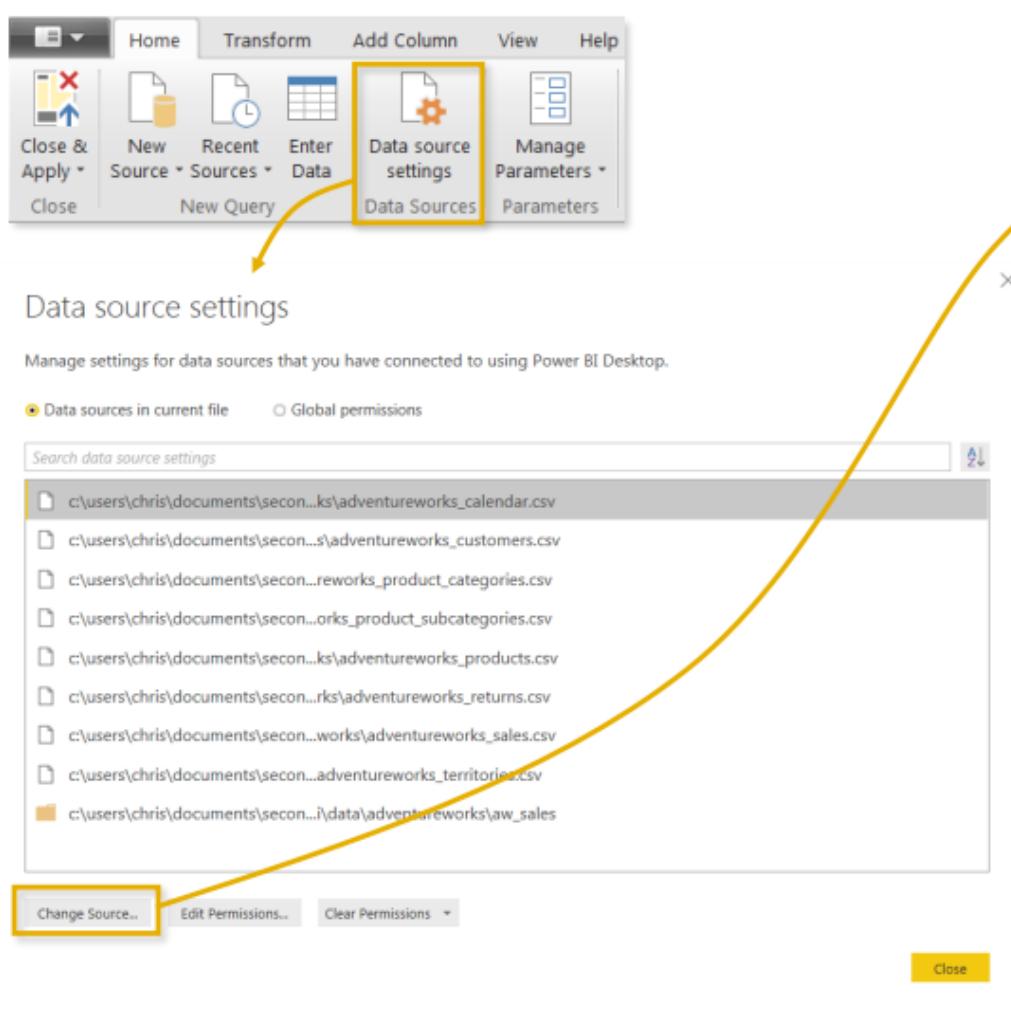
NOTE: Appending *adds rows* to an existing table



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



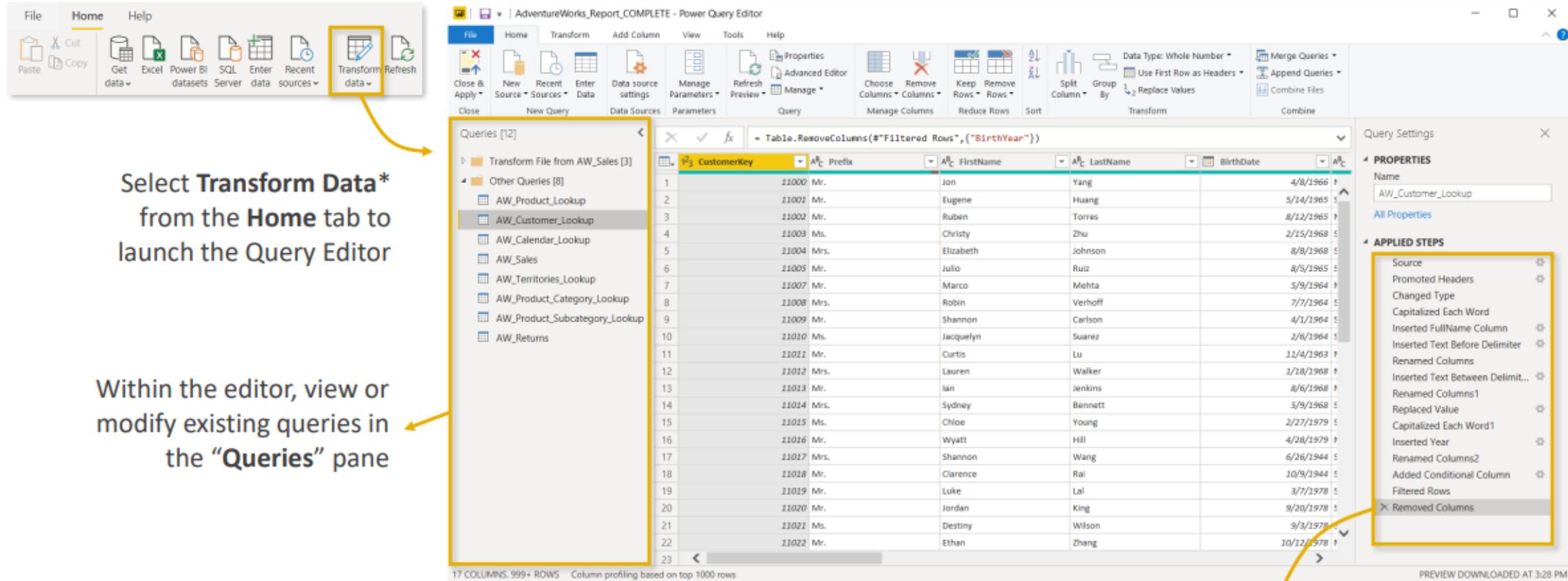
The **Data Source Settings** in the Query Editor allow you to manage data connections and permissions



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 **Transform Data***
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

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.



PRO TIP:

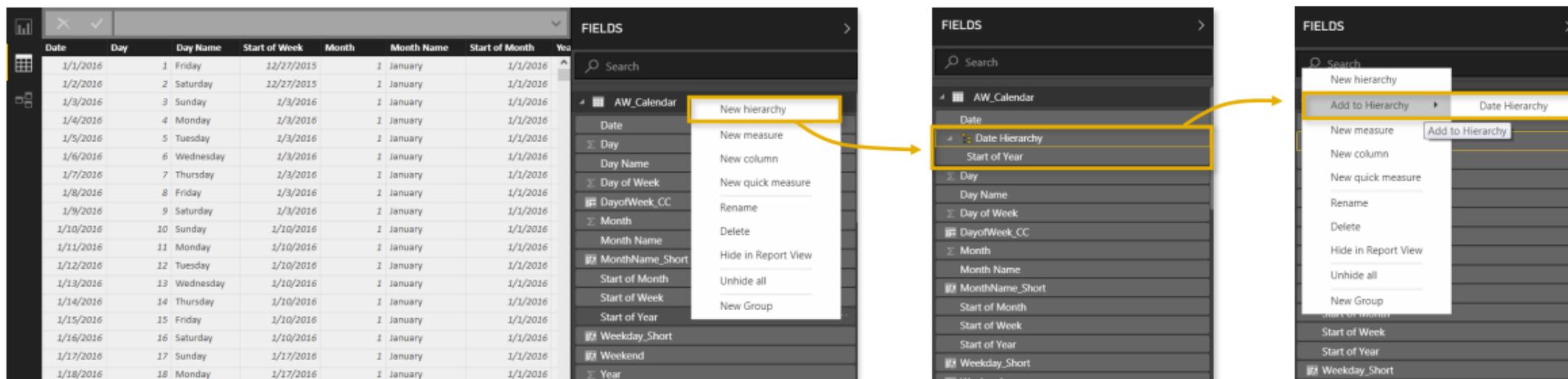
Exclude queries that don't change often, like lookups or static data tables

The screenshot shows the Power BI desktop interface. The ribbon at the top includes File, Home, Transform, Add Column, View, Tools, and Help. Below the ribbon, there are buttons for Close & Apply, New Source, Recent Sources, Enter Data, Data source settings, Manage Parameters, Refresh Preview, Properties, Advanced Editor, and Manage. The 'Queries' pane on the left shows 12 queries, with 'Transform File from AW_Sales' and 'Other Queries' expanded. A context menu is open for the query 'AW_C', listing options like Copy, Paste, Delete, Rename, Enable load (with a checked checkbox), Include in report refresh (highlighted with a yellow box and a yellow arrow), Duplicate, Reference, Move To Group, Move Up, Move Down, Create Function, Convert To Parameter, Advanced Editor, and Properties.

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"

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*

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

რა არის მონაცემთა მოდელი;

მონაცემთა ბაზის ნორმალიზაციის პრინციპები;

განსხვავება მონაცემთა ცხრილებს და საძიებო ცხრილებს შორის;

განსხვავება Table Relationship და Merged Queries შორის; Table Relationship-ს შექმნა;

"Snowflake" სქემები; Relationship-ის მართვა და რედაქტირება;

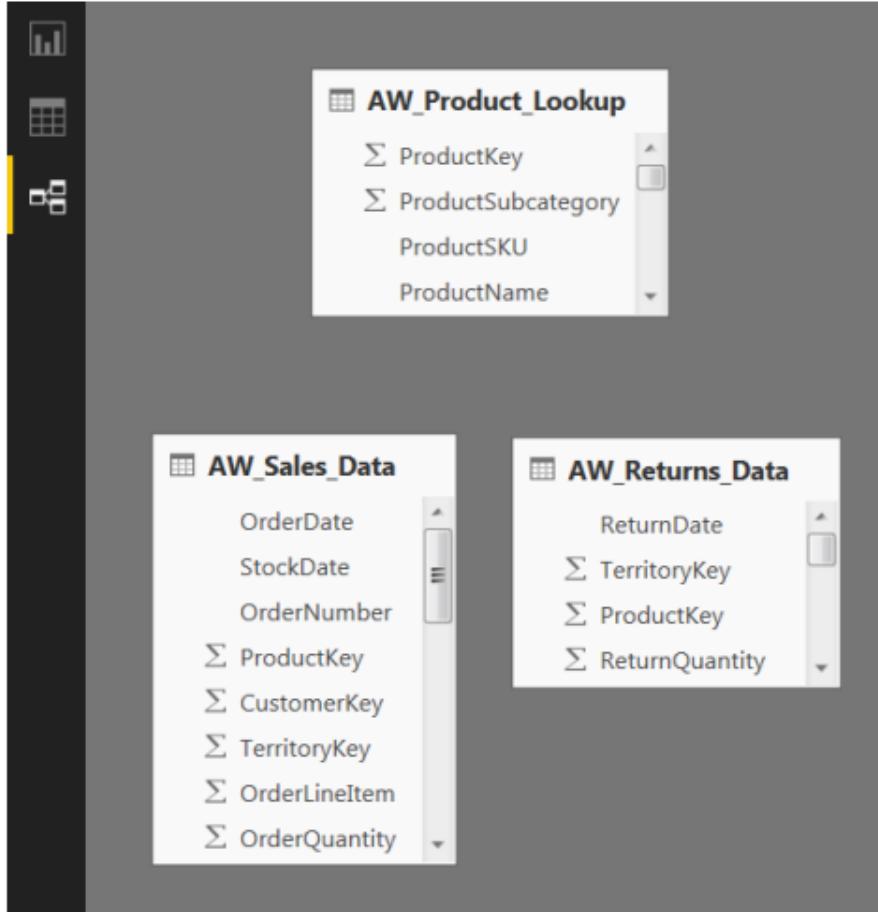
აქტიური და არააქტიური Relationship-ის მართვა;

Relationship-ის კარდინალურობა;

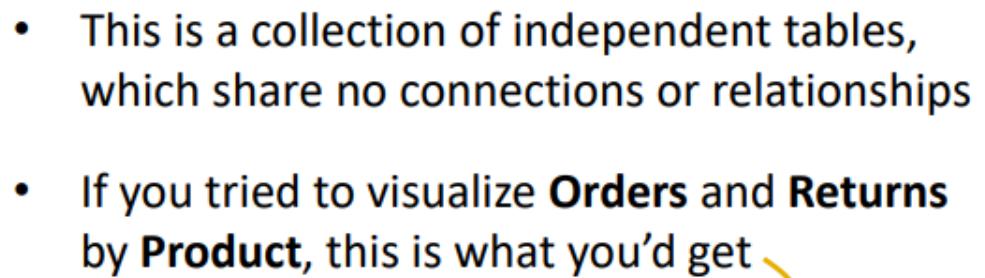
მრავალი მონაცემთა ცხრილის დაკავშირება; ფილტრის ნაკადი; ორმხრივი ფილტრები;

ველების დამალვა Report View-დან

WHAT'S A "DATA MODEL"?

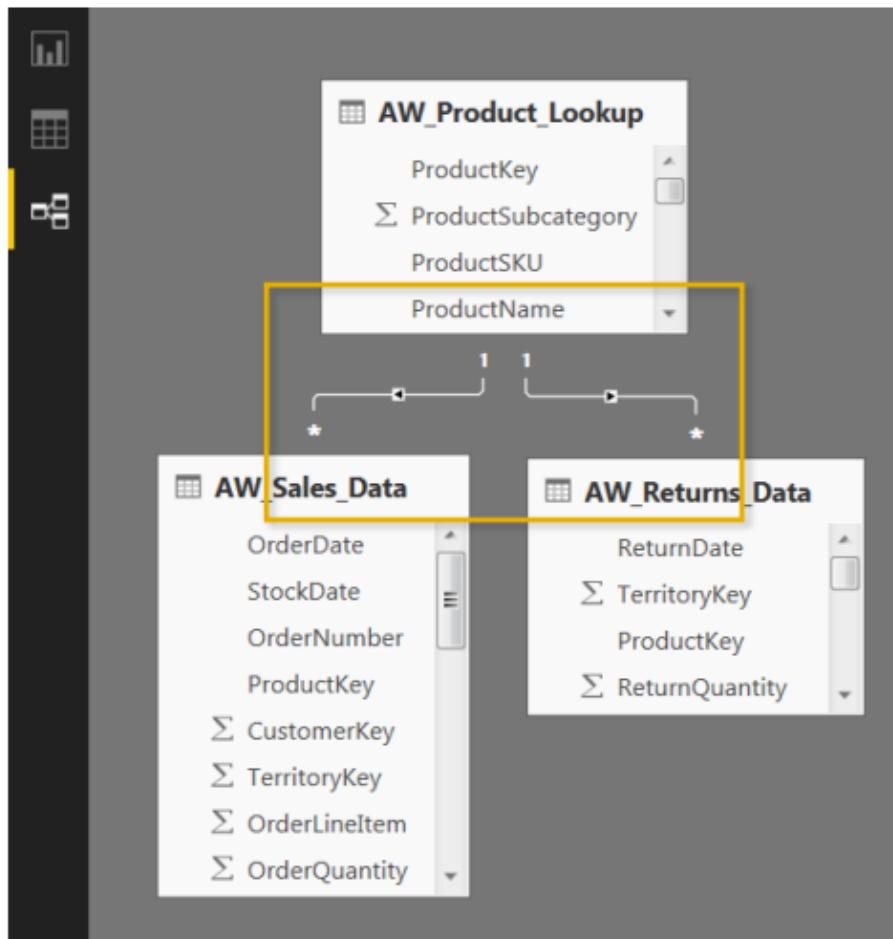


This **IS NOT** a data model 



| 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 | 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 |

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

| 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 |

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 |

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

| 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 **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

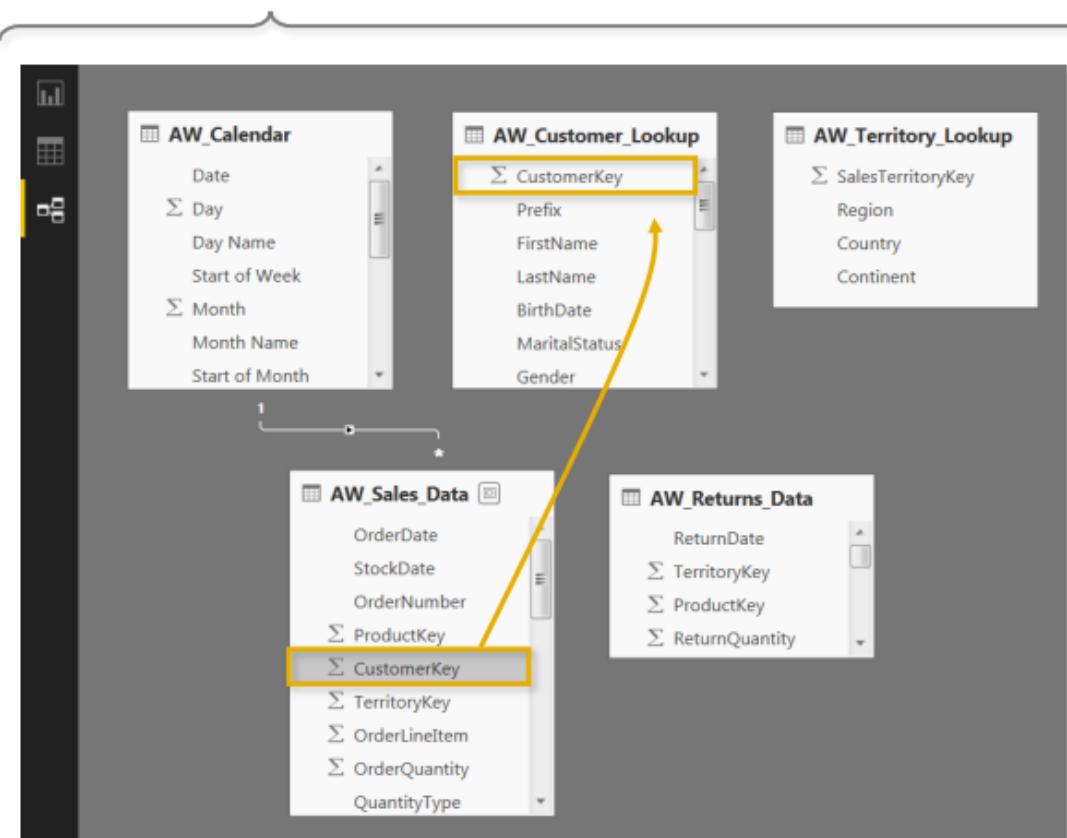
| date | product_id | quantity | day_of_month | month | year | weekday | month_name | quarter | product_brand | product_name | product_sku | product_weight |
|----------|------------|----------|--------------|-------|------|-----------|------------|---------|---------------|------------------------------|-------------|----------------|
| 1/1/1997 | 869 | 5 | 1 | 1 | 1997 | Wednesday | January | Q1 | Nationaleel | Nationaleel Grape Fruit Roll | 52382137179 | 17 |
| 1/7/1997 | 869 | 2 | 7 | 1 | 1997 | Tuesday | January | Q1 | Nationaleel | Nationaleel 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!*

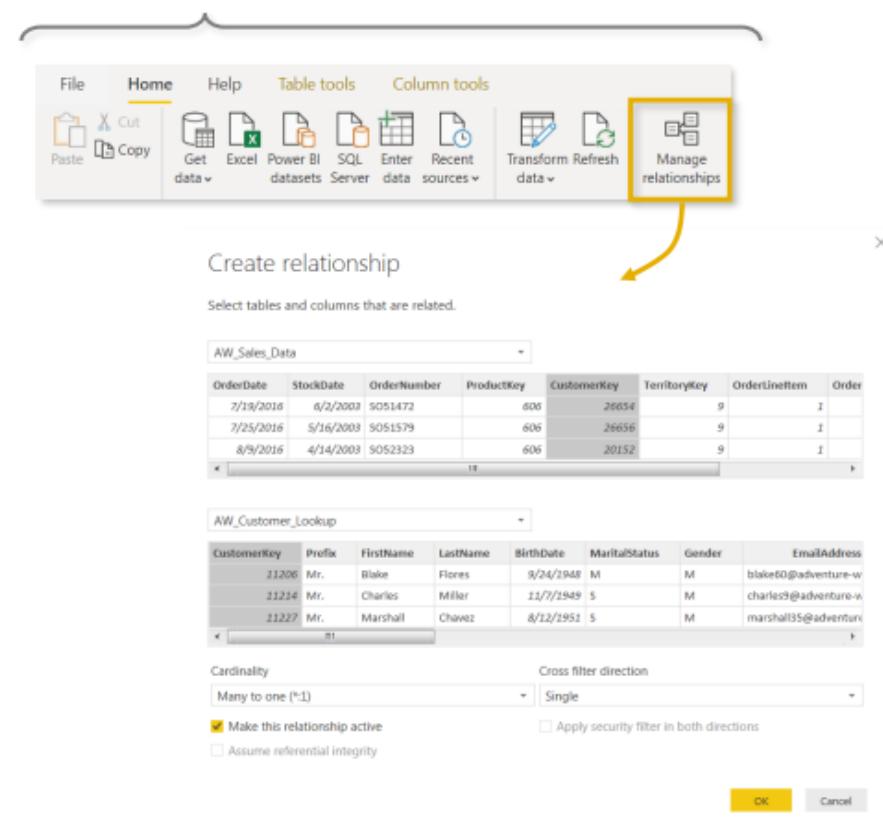
- Merging data in this way creates **redundant data** and utilizes **significantly more memory and processing power** than creating relationships between multiple small tables

CREATING TABLE RELATIONSHIPS

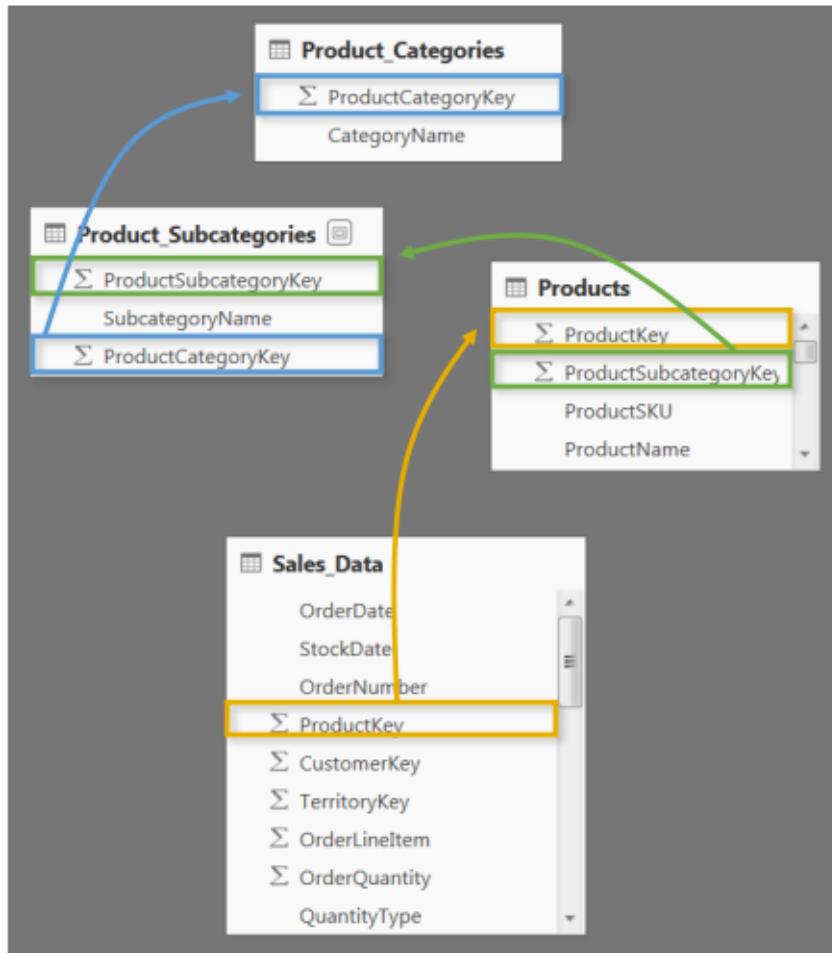
Option 1: Click and drag to connect primary and foreign keys within the **Relationships** pane



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

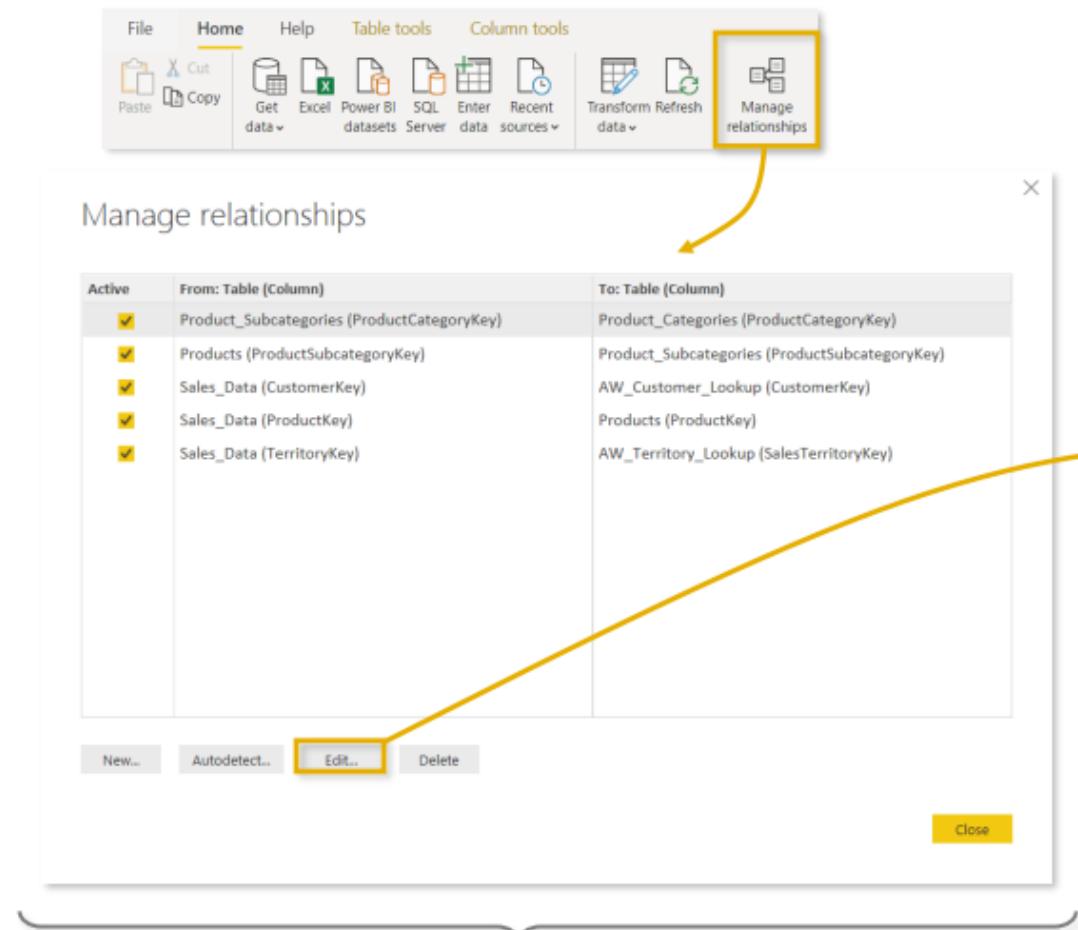
By creating relationships from **Products** to **Subcategories** (using **ProductSubcategoryKey**) and **Subcategories** to **Categories** (using **ProductCategoryKey**), we have essentially connected **Sales_Data** to each lookup table; filter context will now flow all the way down the chain



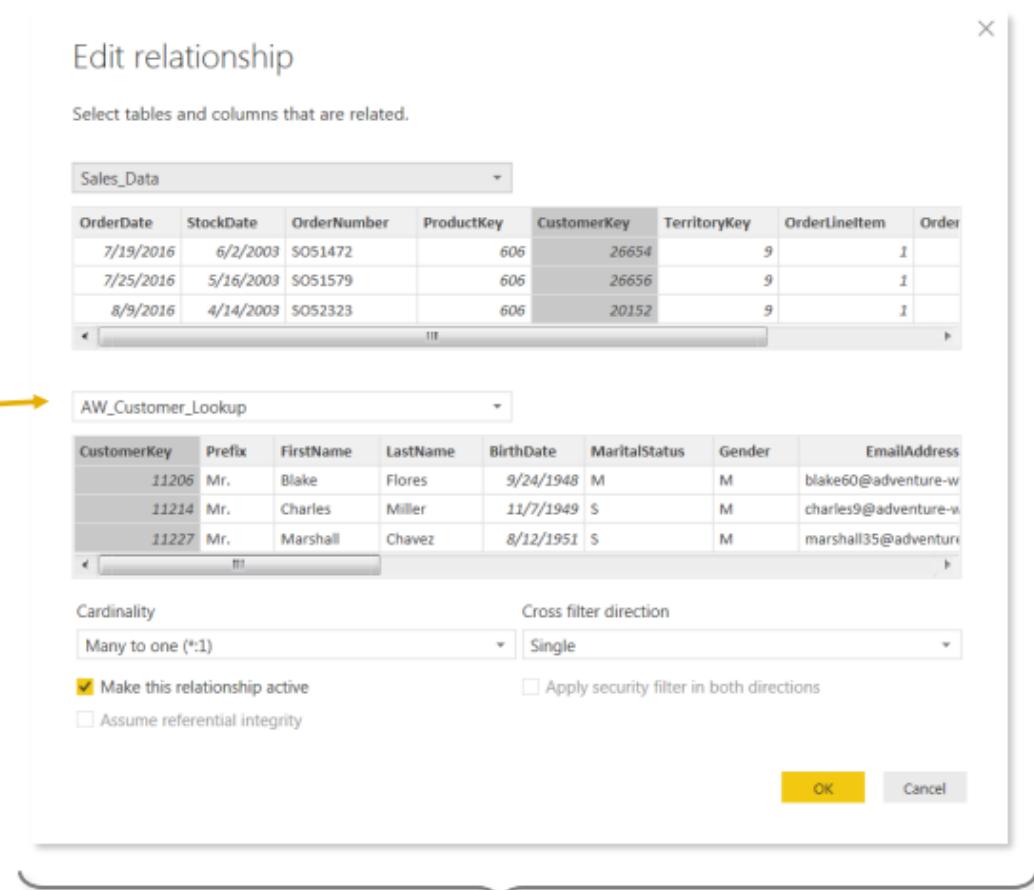
PRO TIP:

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

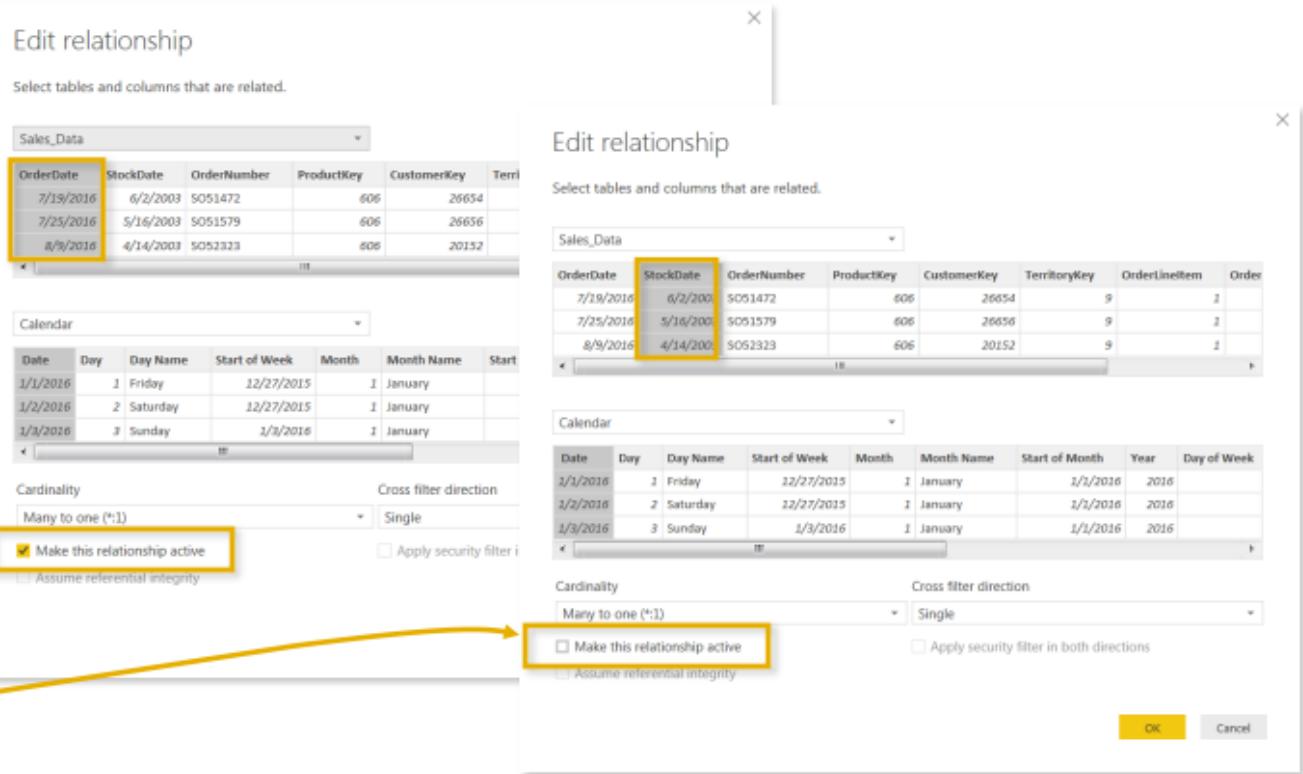
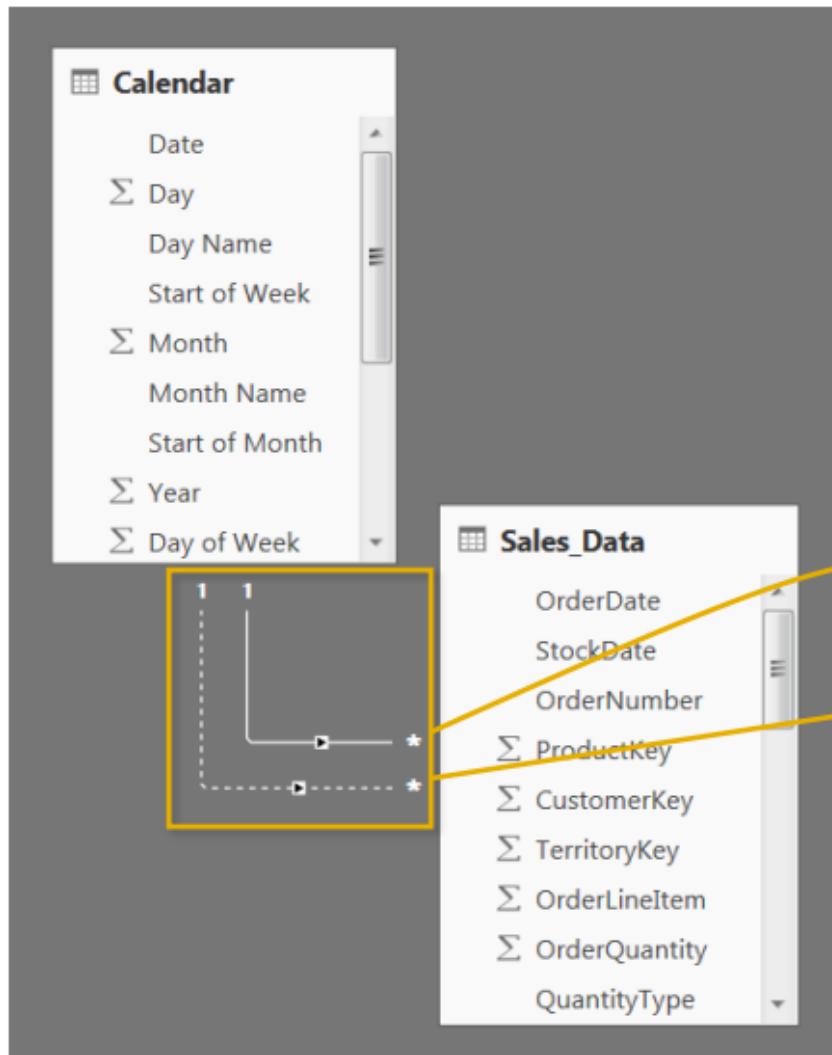


The “**Manage Relationships**” dialog box allows you to **add, edit, or delete** table relationships



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

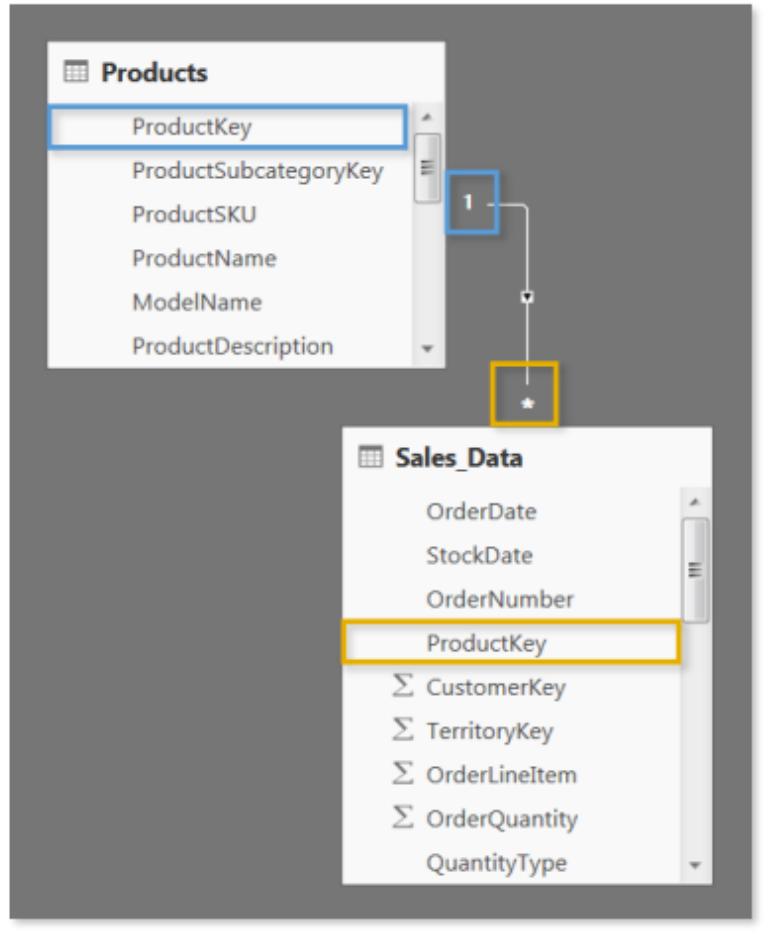
ACTIVE VS. INACTIVE RELATIONSHIPS



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*

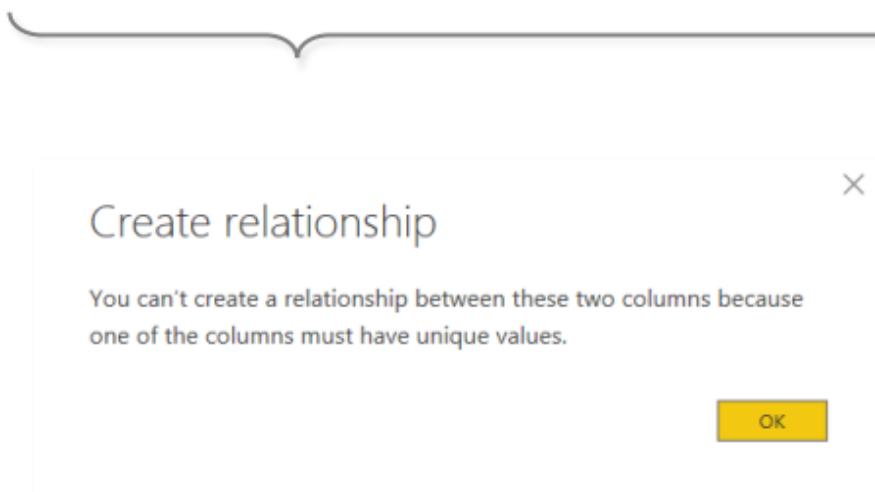
In this case, there is only **ONE instance of each ProductKey** in the Products table (noted by the “1”), since each row contains **attributes of a single product** (Name, SKU, Description, Retail Price, etc)

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 |



- 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

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

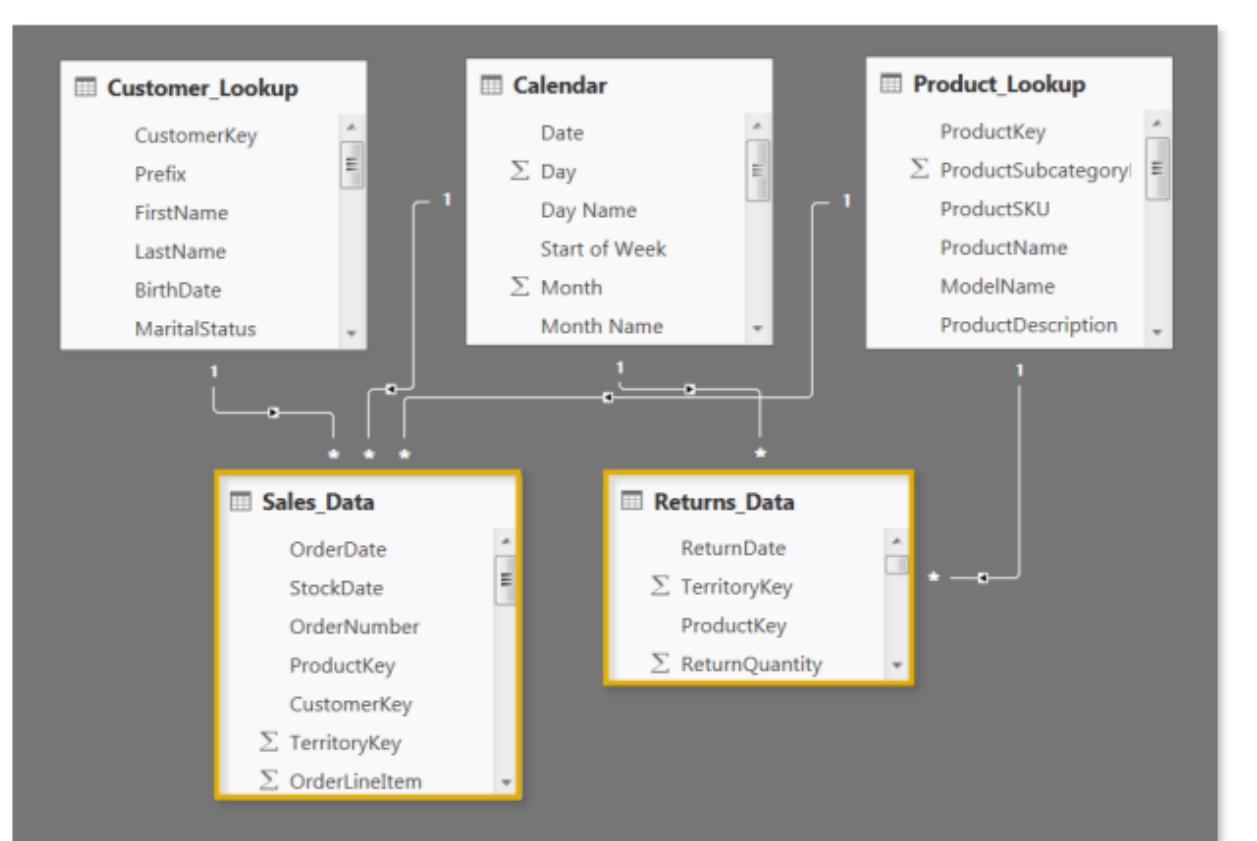
- Connecting the two tables above using the `product_id` field creates a ***one-to-one relationship***, since each ID only appears once in each table
 - Unlike many-to-many, there is nothing *illegal* about this relationship; it's just **inefficient**

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

| 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.)



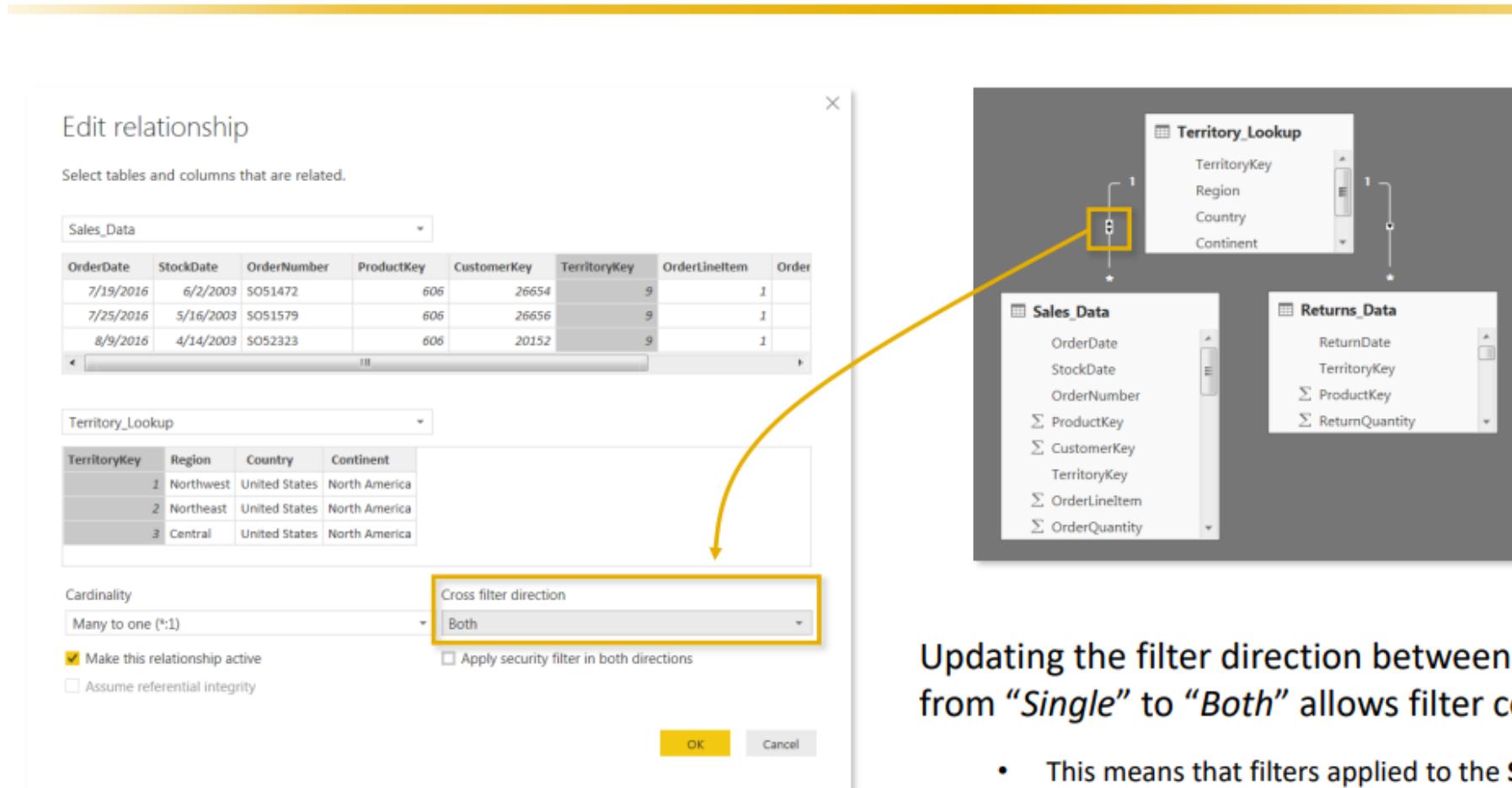
1) Filtering using `TerritoryKey` from the `Territory_Lookup` table

2) Filtering using *TerritoryKey* from the *Sales_Data* table

3) Filtering using `TerritoryKey` from the `Returns_Data` table

- 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)

TWO-WAY FILTERS

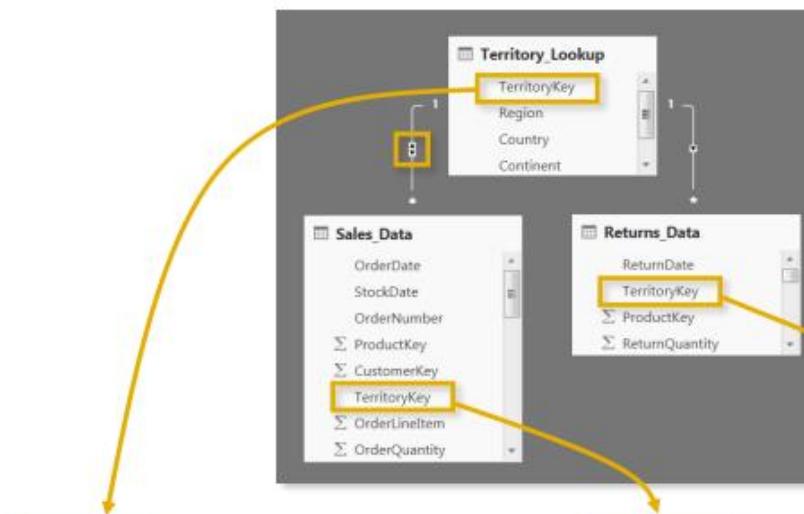


Updating the filter direction between **Sales** and **Territory** from “*Single*” to “*Both*” allows filter context to flow both ways

- 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

| TerritoryKey | OrderQuantity | ReturnQuantity |
|--------------|---------------|----------------|
| 1 | 12,513 | 270 |
| 2 | 40 | |
| 3 | 30 | |
| 4 | 17,191 | 362 |
| 5 | 49 | 1 |
| 6 | 894 | 238 |
| 7 | 7,862 | 186 |
| 8 | 7,950 | 163 |
| 9 | 17,951 | 404 |
| 10 | 9,694 | 204 |
| Total | 84,174 | 1,828 |

1) Filtering using `TerritoryKey` from the **Territory Lookup** table

| TerritoryKey | OrderQuantity | ReturnQuantity |
|--------------|---------------|----------------|
| 1 | 12,513 | 270 |
| 2 | 40 | |
| 3 | 30 | |
| 4 | 17,191 | 362 |
| 5 | 49 | 1 |
| 6 | 694 | 238 |
| 7 | 7,862 | 186 |
| 8 | 7,950 | 163 |
| 9 | 17,951 | 404 |
| 10 | 9,694 | 204 |
| Total | 84,174 | 1,828 |

2) Filtering using `TerritoryKey` from the `Sales Data` table

| TerritoryKey | OrderQuantity | ReturnQuantity |
|--------------|---------------|----------------|
| 1 | 84,174 | 270 |
| 4 | 84,174 | 362 |
| 5 | 84,174 | 1 |
| 6 | 84,174 | 238 |
| 7 | 84,174 | 186 |
| 8 | 84,174 | 163 |
| 9 | 84,174 | 404 |
| 10 | 84,174 | 204 |
| Total | 84,174 | 1,828 |

3) Filtering using TerritoryKey from the **Returns Data** 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*)

| TerritoryKey | OrderQuantity | ReturnQuantity |
|--------------|---------------|----------------|
| 1 | 12,513 | 270 |
| 2 | 40 | |
| 3 | 30 | |
| 4 | 17,191 | 362 |
| 5 | 49 | 1 |
| 6 | 894 | 238 |
| 7 | 7,862 | 186 |
| 8 | 7,950 | 163 |
| 9 | 17,951 | 404 |
| 10 | 9,694 | 204 |
| Total | 84,174 | 1,828 |

1) Filtering using `TerritoryKey` from the `Territory_Lookup` table

| TerritoryKey | OrderQuantity | ReturnQuantity |
|--------------|---------------|----------------|
| 1 | 12,513 | 1,828 |
| 2 | 40 | 1,828 |
| 3 | 30 | 1,828 |
| 4 | 17,191 | 1,828 |
| 5 | 49 | 1,828 |
| 6 | 1,894 | 1,828 |
| 7 | 7,862 | 1,828 |
| 8 | 7,950 | 1,828 |
| 9 | 17,951 | 1,828 |
| 10 | 9,694 | 1,828 |
| Total | 84,174 | 1,828 |

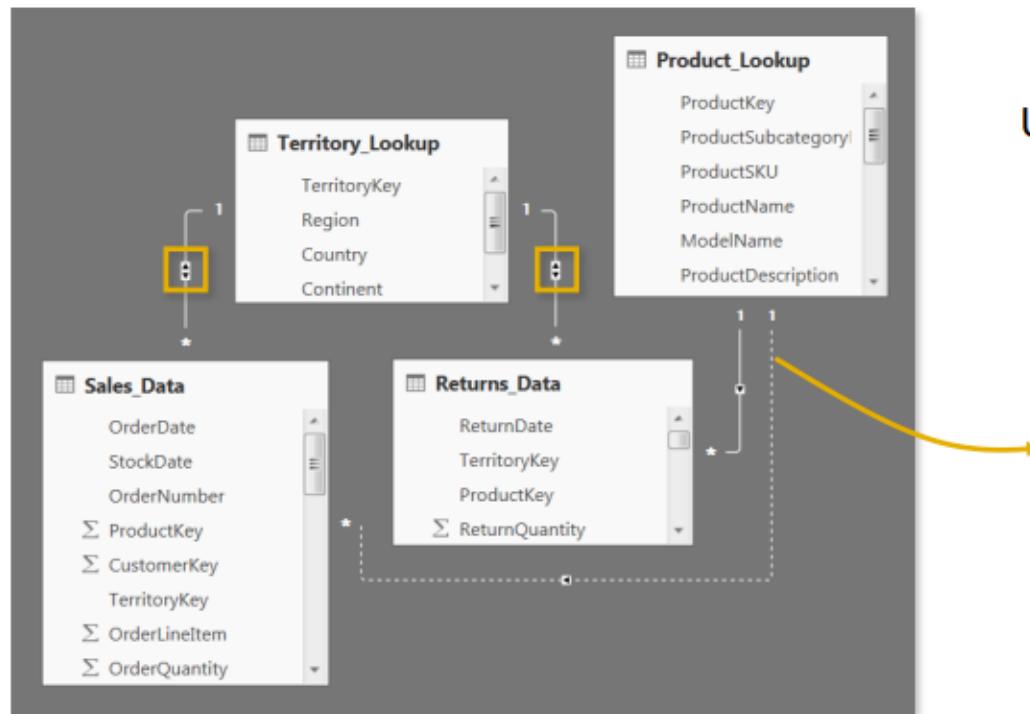
2) Filtering using TerritoryKey from the Sales_Data table

| TerritoryKey | OrderQuantity | ReturnQuantity |
|--------------|---------------|----------------|
| 1 | 12,513 | 270 |
| 4 | 17,191 | 362 |
| 5 | 49 | 1 |
| 6 | 10,894 | 238 |
| 7 | 7,862 | 86 |
| 8 | 7,950 | 163 |
| 9 | 17,951 | 404 |
| 10 | 9,694 | 204 |
| Total | 84,174 | 1,828 |

3) Filtering using TerritoryKey from the **Returns_Data** table

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.

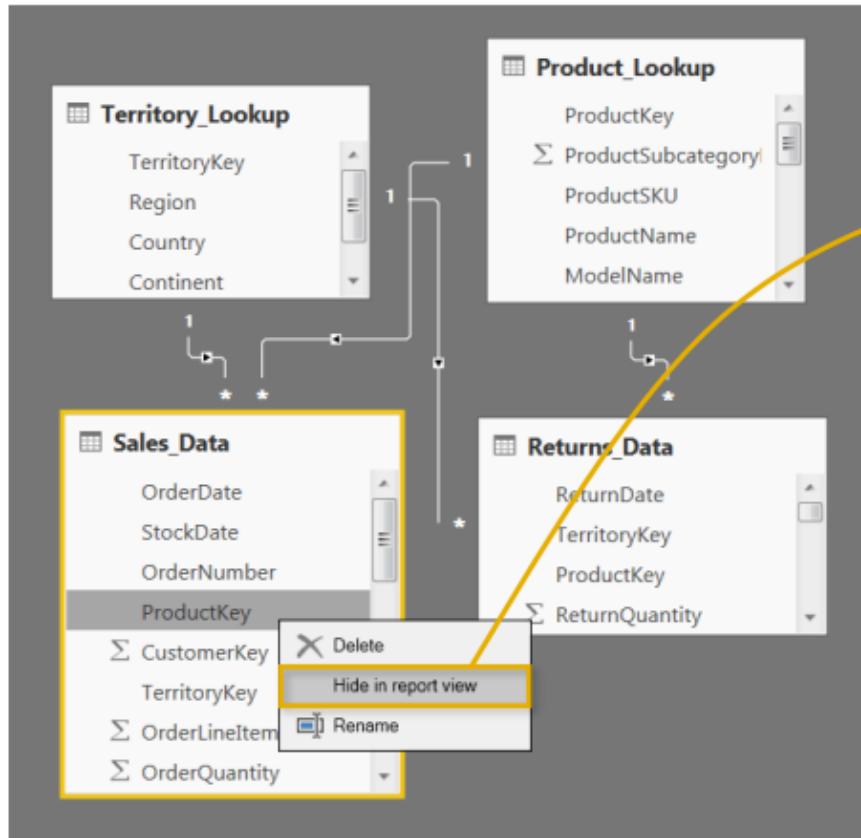
If we were able to activate the relationship between **Product_Lookup** and **Sales_Data** as well, filters could pass from the **Product_Lookup** table through EITHER the Sales or Returns table to reach the **Territory_Lookup**, which could yield conflicting filter context



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



PRO TIP:

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

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

მონაცემთა ანალიზი DAXის გამოყენებით;
ამოთვლილი სვეტები VS DAX კალკულაციები

Implicit vs. Explicit DAX კალკულაციები;

DAX სინტაქსი და ოპერატორები;
DAX ფუნქციების საერთო კატეგორიები;
ძირითადი თარიღი და დროის ფუნქციები;

პირობითი და ლოგიკური ფუნქციები (IF / AND / OR); ტექსტის საერთო ფუნქციები

მონაცემთა შეერთება (RELATED); ძირითადი მათემატიკისა და სტატისტიკის ფუნქციები

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*

| Parent = IF(Customer_Lookup[TotalChildren]>0, "Yes", "No") | | | | | | | Fields | |
|--|--------------|-----------|-----------------------|------------|-----------------|-------------|--------|--|
| Education level | Occupation | HomeOwner | Full Name | User Name | Domain | IncomeLevel | Parent | Search |
| Partial College | Professional | Y | Mr. Blake Flores | blake60 | Adventure Works | Average | Yes | <input type="checkbox"/> AW_Product_Categ... |
| Partial College | Professional | Y | Mr. Charles Miller | charles9 | Adventure Works | Average | Yes | <input type="checkbox"/> AW_Product_Subcat... |
| Partial College | Professional | Y | Mr. Marshall Chavez | marshall95 | Adventure Works | Average | Yes | <input type="checkbox"/> AW_Sales_Data_Full... |
| Partial College | Professional | Y | Mr. Levi Chandra | levi1 | Adventure Works | Average | Yes | <input type="checkbox"/> Calendar |
| Partial College | Professional | Y | Mr. Sean Allen | sean49 | Adventure Works | Average | Yes | <input type="checkbox"/> Customer_Lookup |
| Partial College | Professional | Y | Mr. James Walker | james96 | Adventure Works | Average | Yes | <input type="checkbox"/> AnnualIncome |
| Partial College | Professional | Y | Mr. Cameron Yang | cameron23 | Adventure Works | Average | Yes | <input type="checkbox"/> Average Age |
| Partial College | Professional | N | Mr. Keith Iraje | keith17 | Adventure Works | Average | Yes | <input type="checkbox"/> BirthDate |
| Partial College | Professional | Y | Mr. Richard Coleman | richard61 | Adventure Works | Average | Yes | <input type="checkbox"/> BirthYear_CTC |
| Partial College | Professional | Y | Mr. Robert Lewis | robert81 | Adventure Works | Average | Yes | <input type="checkbox"/> CurrentAge |
| Partial College | Professional | Y | Mr. Jonathan Robinson | jonathan72 | Adventure Works | Average | Yes | |
| Partial College | Professional | Y | Mr. Robert Wang | robert36 | Adventure Works | Average | Yes | |
| Partial College | Professional | N | Mr. Angel Scott | angel68 | Adventure Works | Average | Yes | |

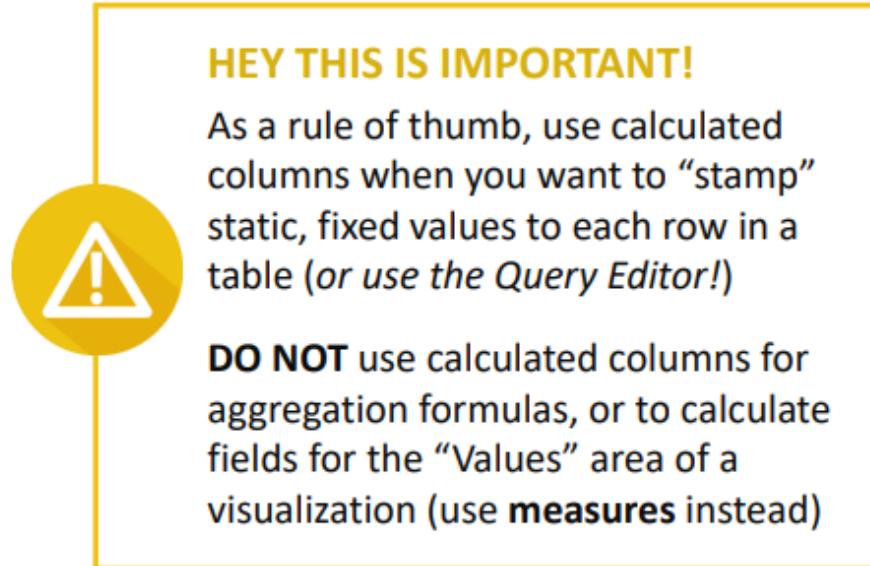
2) *Measures*

| | | |
|---------------|--------------|---|
| 1948 Standard | | >Returns_Data |
| 1948 Standard | | Sales_Data |
| 1951 Standard | New measure | All Orders |
| 1948 Standard | New column | My Rolling Rev... |
| 1948 Standard | New quick me | Total Orders = DISTINCTCOUNT(Sales_Data[OrderNumber]) |
| 1948 Standard | Refresh data | |
| 1948 Standard | Edit Query | Total Revenue = SUMX(Sales_Data, Sales_Data[OrderQuantity] * RELATED(Product_Lookup[ProductPrice])) |
| 1948 Standard | | Quantity Ordered = SUM(Sales_Data[OrderQuantity]) |

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*)



PRO TIP:

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

CALCULATED COLUMNS (EXAMPLES)

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

| 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.

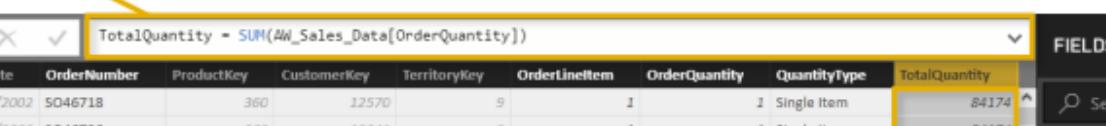


Table showing Sales Data with a calculated column:

```
let
    SalesData = SUM(AW_Sales_Data[OrderQuantity])

```

Fields pane:

- AW_Calendar
- AW_Customer_Loo...
- AW_Product_Categ...
- AW_Product_Lookup
- AW_Product_Subca...
- AW_Returns_Data
- AW_Sales_Data
- % of All Orders
- 10-Day Rolling Reve...
- Adjusted Revenue
- All Orders

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

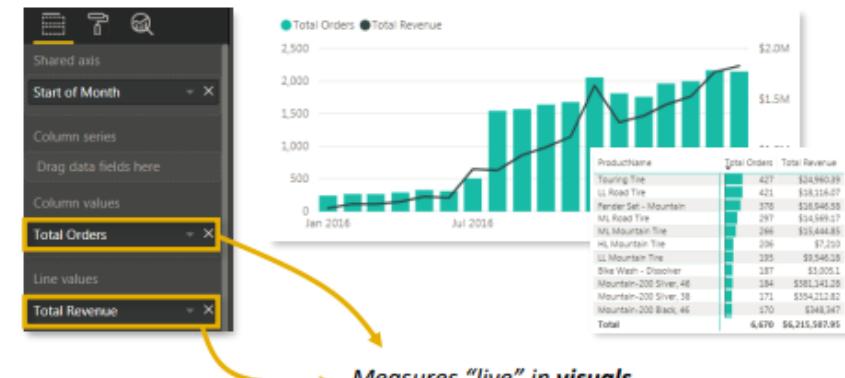
- Values are calculated based on information from each row of a table (**has row context**)
 - Appends static values to each row in a table and stores them in the model (*which increases file size*)
 - Recalculate on data source refresh or when changes are made to component columns
 - Primarily used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

| Parent = IF([Customer_Lookup[TotalChildren]>0, "Yes", "No") | | | | | | | |
|---|--------------|-----------|-----------------------|------------|-----------------|-------------|--------|
| Level | Occupation | HomeOwner | Full Name | User Name | Domain | IncomeLevel | Parent |
| Beginner | Professional | Y | Mr. Blake Flores | blake60 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Charles Miller | charles9 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Marshall Chavez | marshall35 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Levi Chandra | levi1 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Sean Allen | sean49 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. James Walker | james96 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Cameron Yang | cameron23 | Adventure Works | Average | Yes |
| Beginner | Professional | N | Mr. Keith Raji | keith17 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Richard Coleman | richard61 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Robert Lewis | robert81 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Jonathan Robinson | jonathan72 | Adventure Works | Average | Yes |
| Beginner | Professional | Y | Mr. Robert Wang | robert36 | Adventure Works | Average | Yes |

Calculated columns “live” in tables

MEASURES

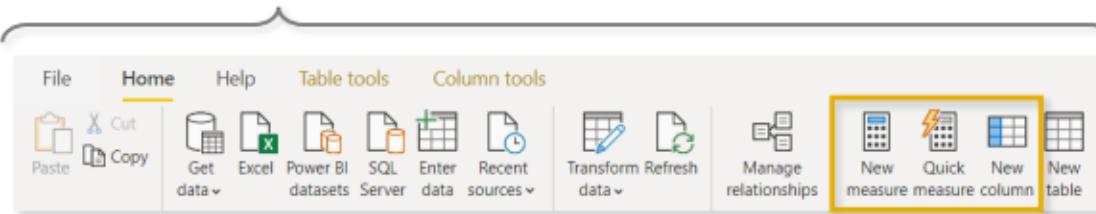
- Values are calculated based on information from any filters in the report (has **filter context**)
 - Does not create new data in the tables themselves (*doesn't increase file size*)
 - Recalculate in response to any change to filters within the report
 - Almost *always* used within the **values**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



- Measures "live" in visuals

ADDING COLUMNS & MEASURES

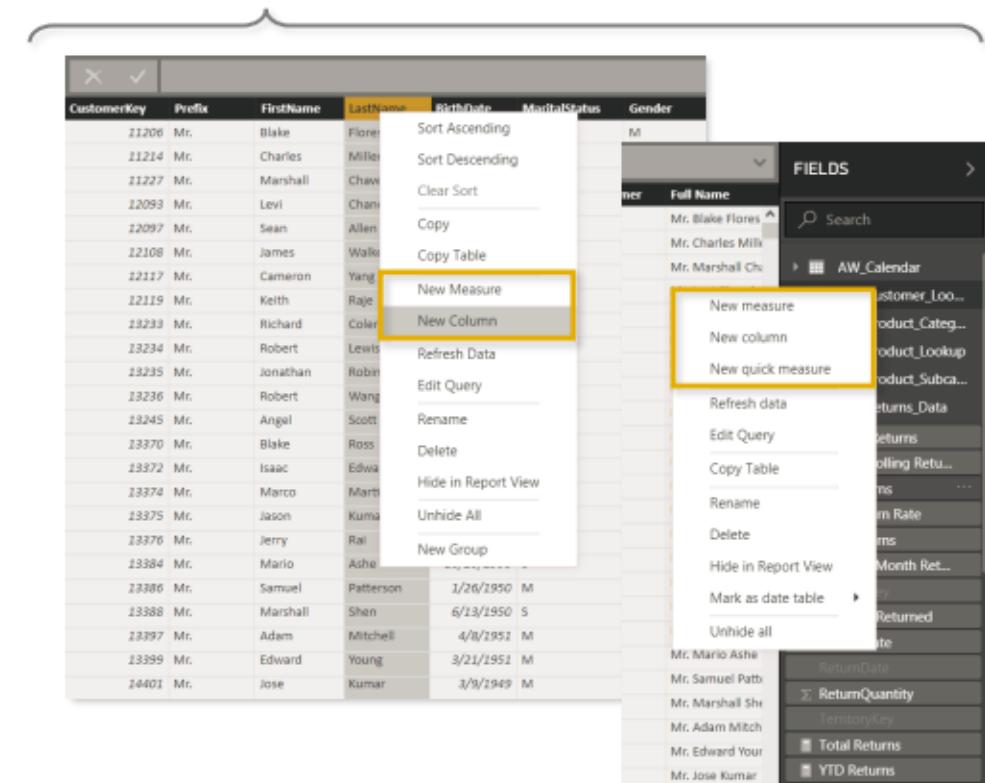
Option 1: Select “New Measure” or “New Column” from the Home tab*



When you insert Columns or Measures using the **Home** 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 “Structure” options in the contextual **Measure Tools** tab), but 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

New measure

New column

New quick measure

Refresh data

Edit Query

Rename

Delete

Hide

Mark as date table ▾

View hidden

Unhide all

Collapse all

Expand all

Properties

Quick measures

Calculation

Average per category

Select a calculation

Aggregate per category

Average per category

Variance per category

Max per category

Min per category

Weighted average per category

Filters

Filtered value

Difference from filtered value

Percentage difference from filtered value

Sales from new customers

Time intelligence

Year-to-date total

Quarter-to-date total

Month-to-date total

Year-over-year change

Quarter-over-quarter change

Month-over-month change

Rolling average

Fields

Search

AW_Calendar

AW_Customer_Lookup

AW_Product_Category_Lookup

AW_Product_Lookup

AW_Product_Subcategory_Lookup

AW_Returns_Data

AW_Sales_Data

AW_Territory_Lookup

Parameter

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

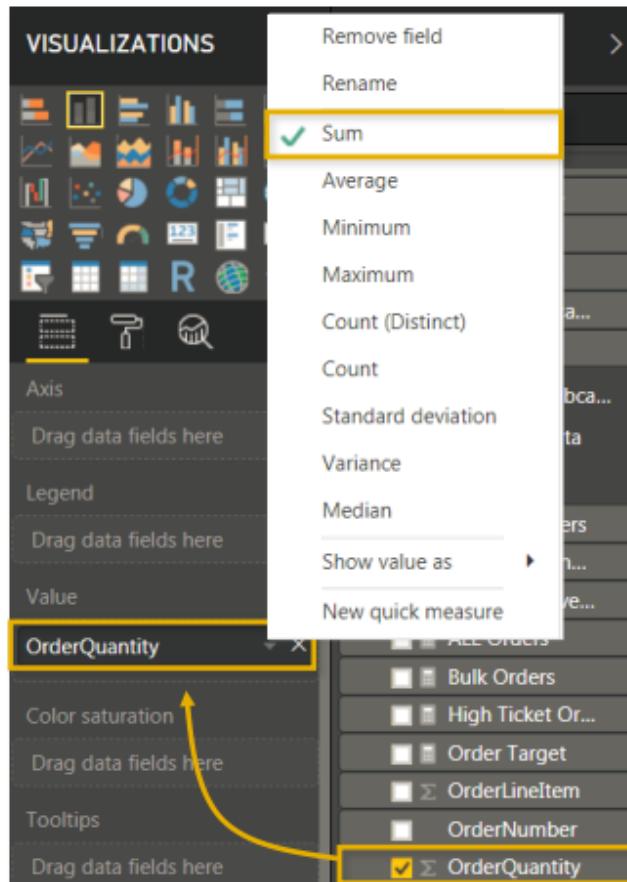
While these tools can be helpful for defining more complex measures (*like weighted averages or time intelligence formulas*), they encourage laziness and don't help you understand the fundamentals of DAX



PRO TIP:

*Just say “NO” to quick measures
(you’re better than that)*

IMPLICIT VS. EXPLICIT MEASURES



Example of an implicit measure

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

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



HEY THIS IS IMPORTANT!

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

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

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 | 1.63 % |
| 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 % |
| 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 | 155 | 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 it's 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 | \$12,015.38 |
| Mrs. Lisa C. | 7 | \$11,330.44 |
| Mrs. Lacey Zheng | 7 | \$11,085.74 |
| Mr. Jordan Turner | 7 | \$11,022.38 |
| Mr. Larry Munoz | 3 | \$10,852.04 |
| Mrs. Ariana Gray | 6 | \$10,628.94 |
| Mr. Marco Lopez | 6 | \$10,200.63 |
| Mr. Franklin Xu | 5 | \$10,164.34 |
| Mrs. Margaret He | 4 | \$9,266.74 |
| Mrs. Kaitlyn Henderson | 4 | \$9,228.82 |
| Mrs. Nichole Nara | 4 | \$9,234.66 |
| Mr. Randall Dominguez | 4 | \$9,210.36 |
| Mrs. Rosa Hu | 4 | \$9,201.21 |
| Adriana Gonzalez | 4 | \$9,155.69 |
| Mrs. Dominique Prasad | 6 | \$9,180.93 |
| Mrs. Brandi Gill | 4 | \$9,166.18 |
| Mr. Brad She | 4 | \$9,161.03 |
| 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. J. Suarez | 4 | \$7,652.61 |
| Mr. Preston Raman | 4 | \$7,599.49 |
| Mr. Willie Xu | 4 | \$7,535.55 |
| Mrs. Abby Subram | 4 | \$7,308.38 |
| Mr. Lance Blanco | 4 | \$7,207.07 |
| Mrs. Audrey Blanco | 4 | \$7,139.15 |
| Mrs. Ricky Navarro | 3 | \$7,119.63 |
| Mr. Eddie Dominguez | 3 | \$7,044.38 |
| Mrs. Molly Madan | 3 | \$7,043.23 |
| Mr. Jarrod Mehta | 3 | \$7,038.37 |
| Ms. Susan Zhou | 3 | \$7,027.98 |
| Mr. Brent Zhang | 3 | \$7,018.95 |
| Ms. Alyssa Bradley | 3 | \$7,018.64 |
| Total | 22,534 | \$18,509,633.22 |

MEASURE: Total Orders

FILTER CONTEXT:

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



Mr. Maurice Shan
Top Customer

23K

\$18.51M

MEASURE: Total Orders

FILTER CONTEXT:

- *Calendar[Year] = 2016 or 2017*

MEASURE: Total Orders

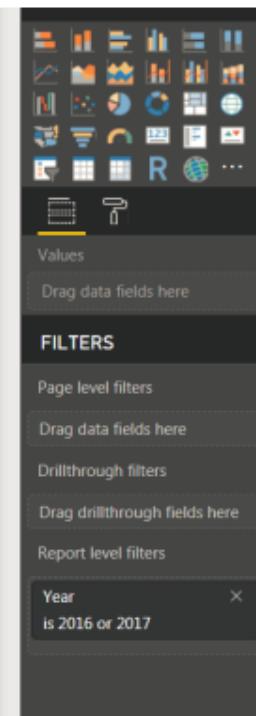
FILTER CONTEXT:

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

MEASURE: Total Orders

FILTER CONTEXT:

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



This is a page-level filter, which impact ALL visuals on the report page

STEP-BY-STEP MEASURE CALCULATION

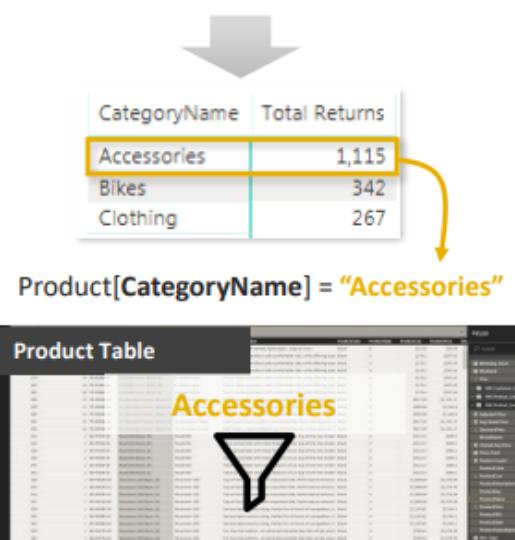
| CategoryName | Total Returns |
|--------------|---------------|
| Accessories | 1,115 |
| Bikes | 342 |
| Clothing | 267 |

• How exactly is this measure calculated?

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

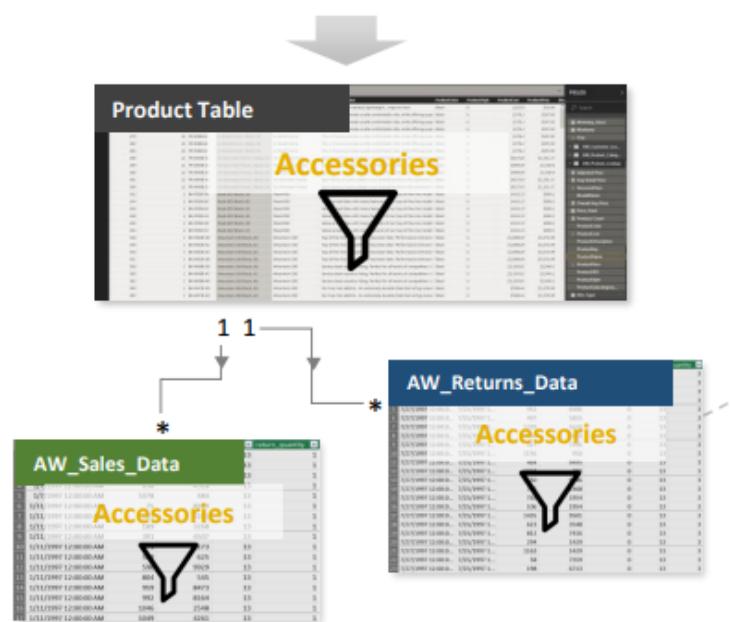
STEP 1

Filter context is detected & applied



STEP 2

Filters flow “downstream” to all related tables



STEP 3

Measure formula evaluates against the filtered table



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

= 1,115

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*)

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 |
| >= | Greater than or equal to | [Unit_Price] >= 2.5 |
| <= | Less than or equal to | [Unit_Price] <= 2.5 |
| <> | Not equal to | [Country] <> "Mexico" |

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

COMMON FUNCTION CATEGORIES

MATH & STATS

Functions

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

Common Examples:

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

Iterator Functions:

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

LOGICAL Functions

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

Common Examples:

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

TEXT Functions

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

Common Examples:

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

FILTER Functions

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

Common Examples:

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

DATE & TIME Functions

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

Common Examples:

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

Time Intelligence Functions:

- DATESYTD
 - DATESQTD
 - DATESMTD
 - DATEADD
 - DATESINPERIOD

BASIC DATE & TIME FUNCTIONS

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

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)

BASIC MATH & STATS FUNCTIONS

SUM()

Evaluates the sum of a column

=SUM(Column Name)

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])

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(ColumnName)

↓

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

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

COUNT ფუნქცია (COUNTA, DISTINCTCOUNT, COUNTROWS)

CALCULATE ბრძანება; CALCULATE და ALL შეკვეთა

CALCULATE და FILTER ბრძანება;

Iterator ფუნქციები (SUMX, RANKX); Time intelligence ფორმულები; DAX-ის
საუკეთესო პრაქტიკა

BASIC MATH & STATS FUNCTIONS

SUM()

Evaluates the sum of a column

=SUM(Column Name)

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(Column Name) 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(Column Name)

COUNTA()

Counts the number of non-empty cells in a column (numerical and non-numerical)

=COUNTA(Column Name)

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)

  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?



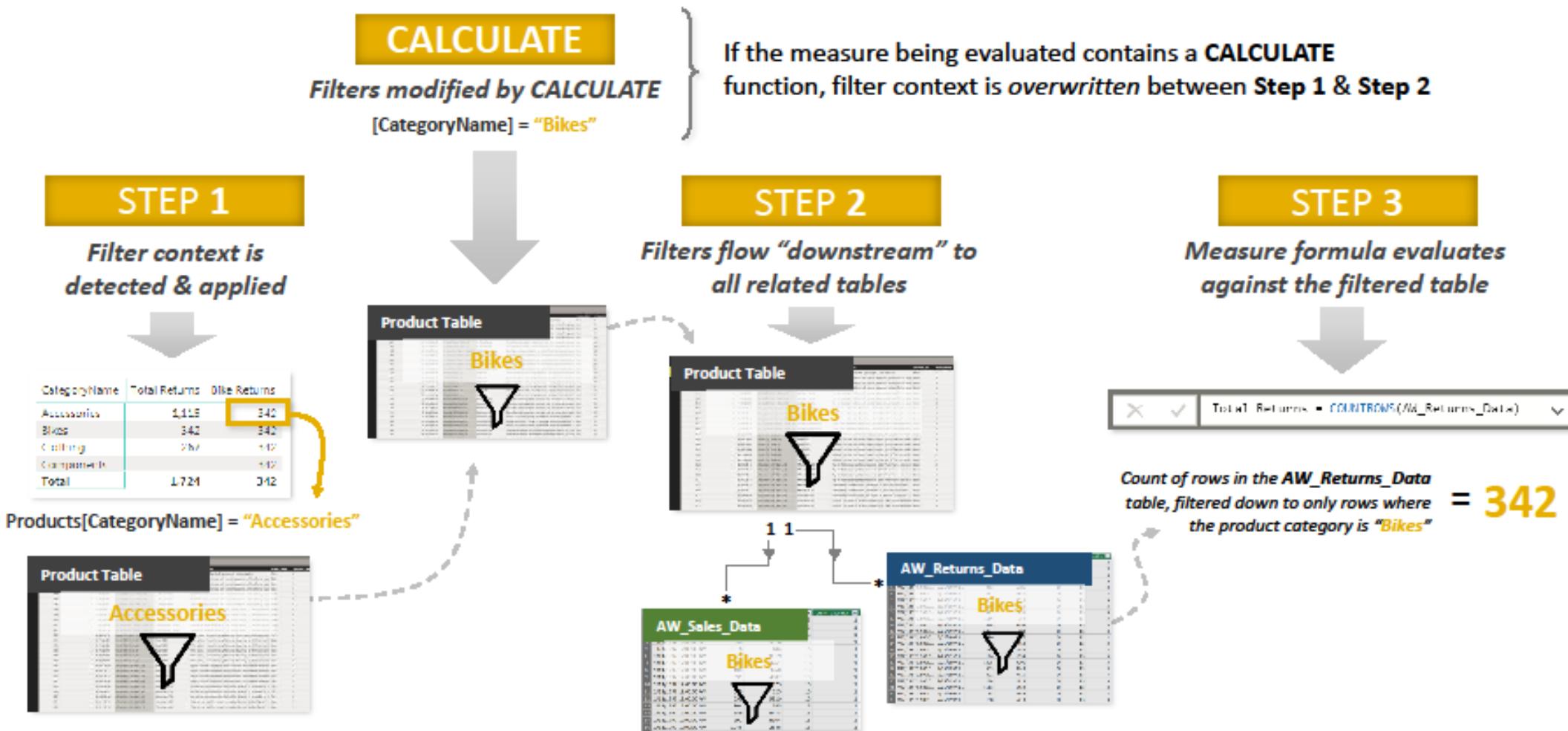
HEY THIS IS IMPORTANT!

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

*Copyright 2018, Excel Maven & Maven Analytics, LLC

TIME INTELLIGENCE FORMULAS

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

Performance To-Date =CALCULATE(Measure, DATESYTD(Calendar[Date]))

Use **DATESQTD** for Quarters or **DATESMTD** for Months

Previous Period =CALCULATE(Measure, DATEADD(Calendar[Date], -1, MONTH))

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

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



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

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

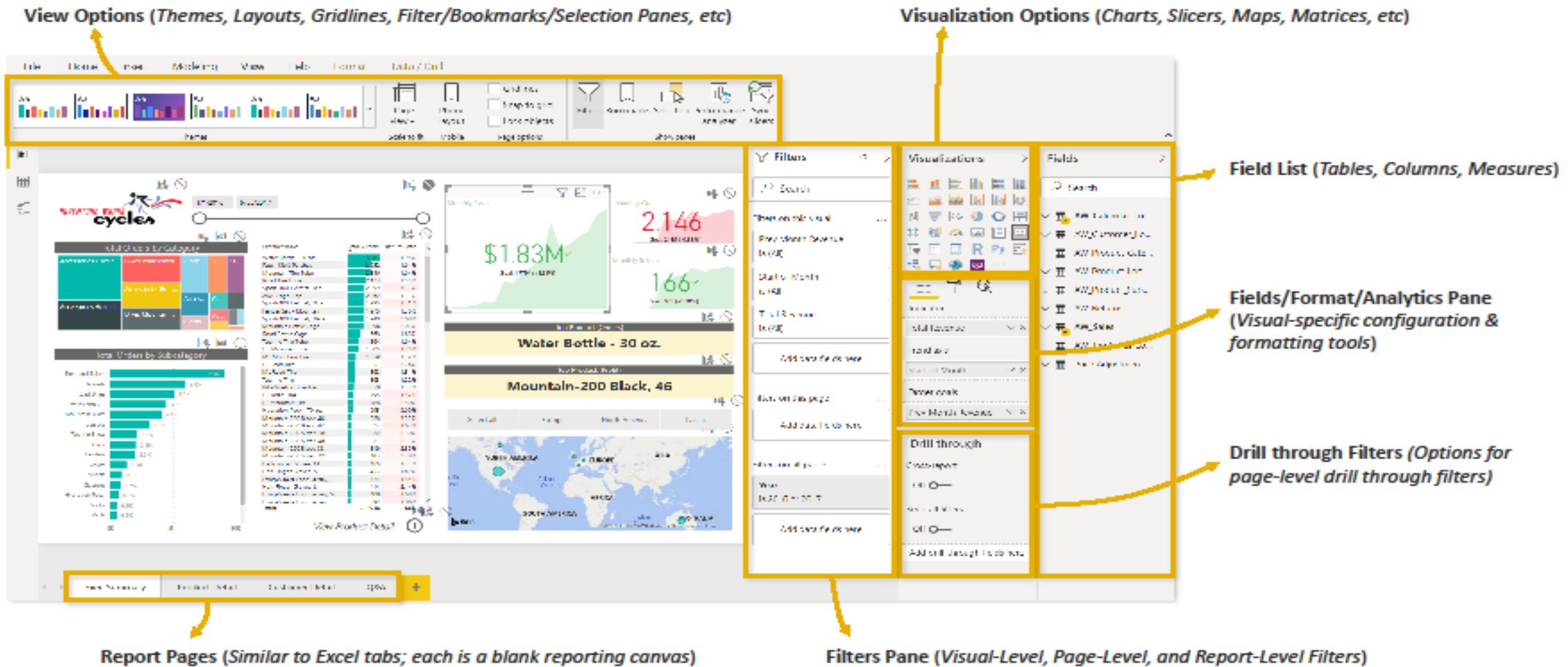
მონაცემთა ვიზუალიზაცია; მოვლე მიმოხილვა; "Report" ჩანართი;

მარტივი ობიექტების დამატება (Report Canvas); ძირითადი გრაფიკების და დიაგრამების დამატება;

პირობითი ფორმატირება; რეპორტის ფორმატირების პარამეტრები;

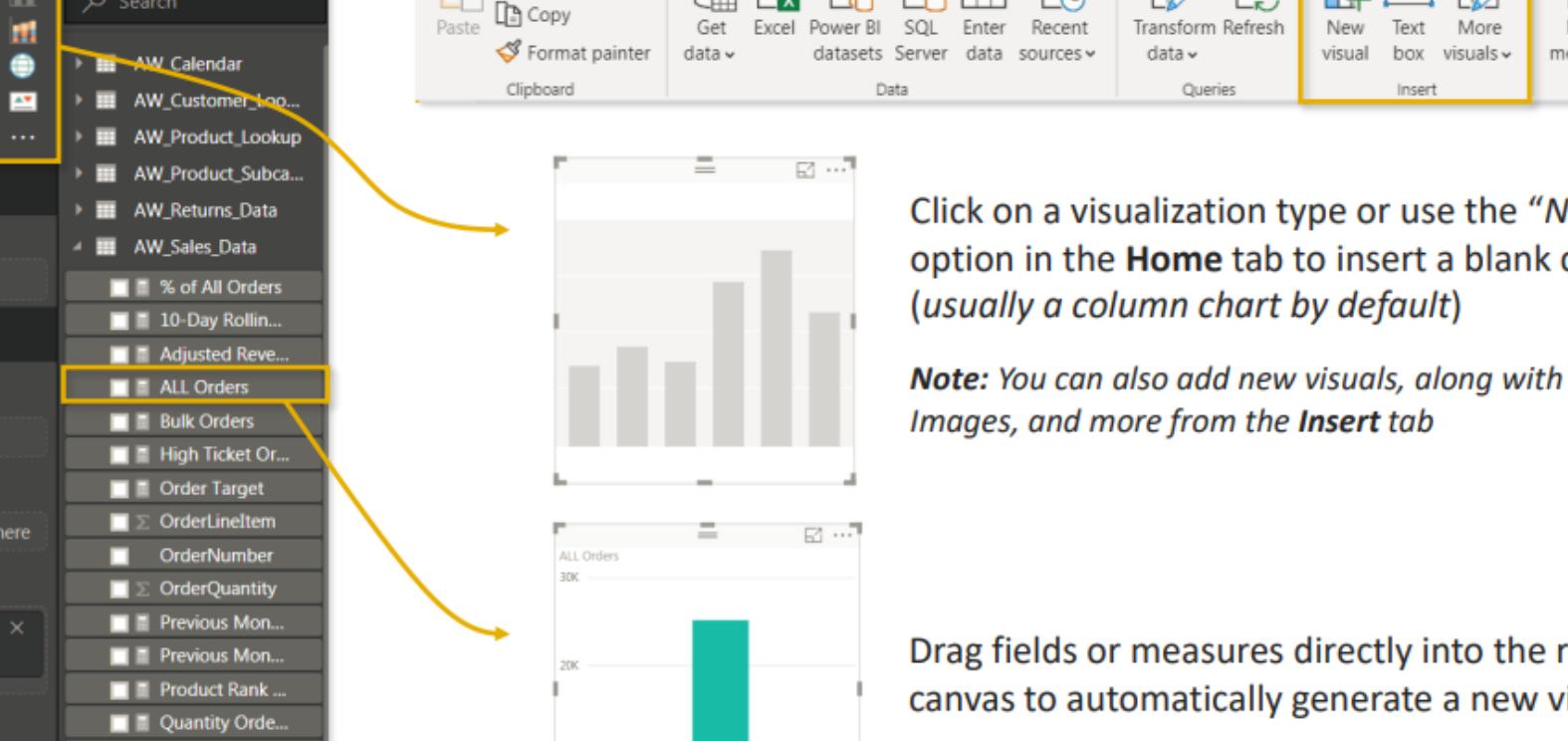
რეპორტის ფილტრაციის პარამეტრები;

THE POWER BI REPORT VIEW



©Copyright 2018, Dual Moon & Moon Analytics, LLC

INSERTING OBJECTS & BASIC CHARTS



The screenshot shows the Power BI desktop interface. The top navigation bar includes File, Home, Insert, Modeling, View, Help, Format, and Data / Drill. The Home tab is selected. The ribbon below the navigation bar includes sections for Paste, Cut, Copy, Format painter, Get data, Excel, Power BI datasets, SQL Server, Enter data, Recent sources, Transform, Refresh data, New visual, Text box, More visuals, New measure, Quick measure, Calculations, and Share. The Fields pane on the left lists various data sources and fields, with 'AW_Sales_Data' expanded to show 'ALL Orders' and other measures. The main canvas area displays two visualizations: a bar chart showing sales data and a single large teal bar chart for 'ALL Orders'.

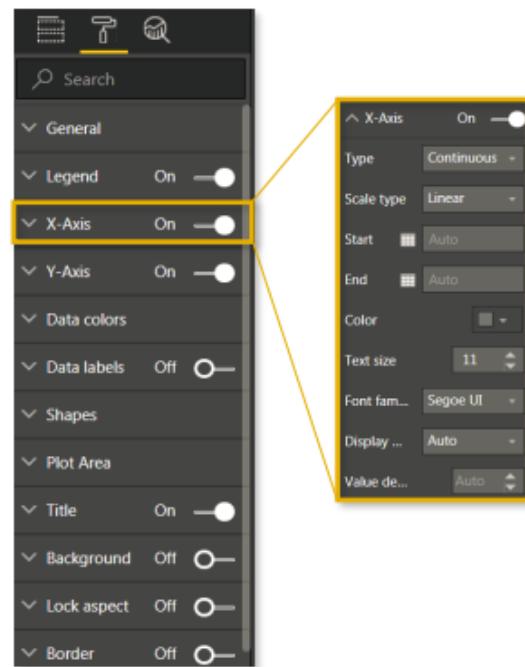
Click on a visualization type or use the “New Visual” option in the **Home tab to insert a blank chart template (usually a column chart by default)**

Note: You can also add new visuals, along with Pages, Buttons, Images, and more from the **Insert** tab

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

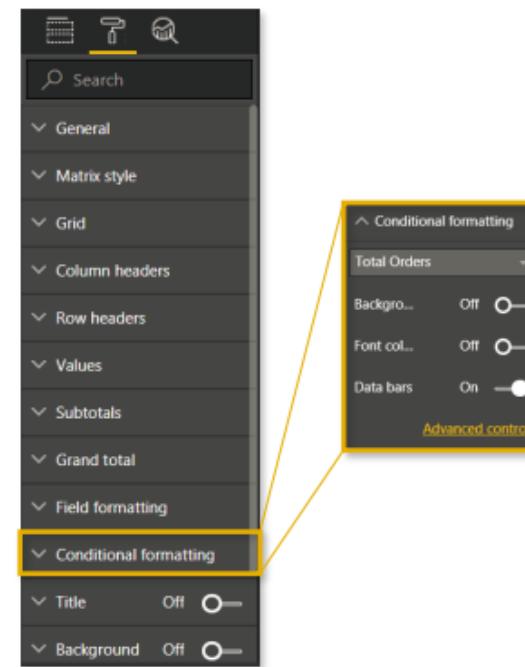
FORMATTING OPTIONS

Example: Line & Column Chart

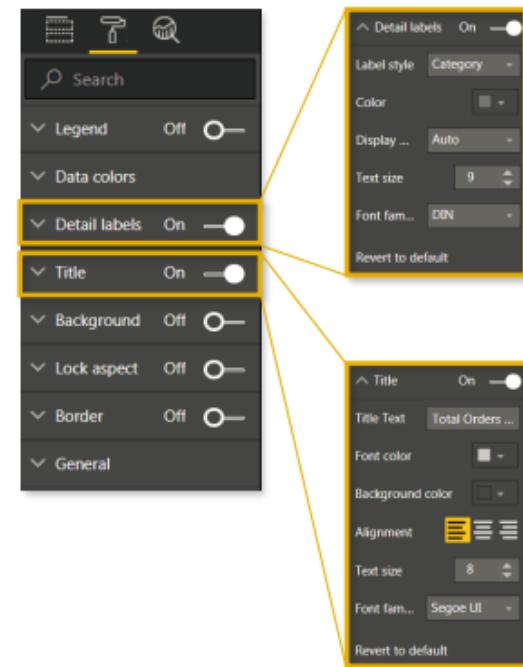
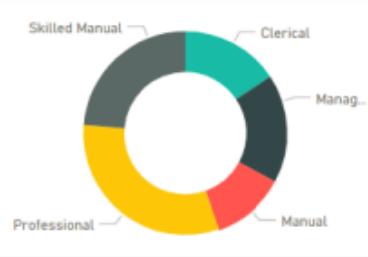


Example: Matrix

| ProductName | Total Orders | Return Rate |
|-------------------------|--------------|-------------|
| Water Bottle - 30 oz. | 1164 | 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**

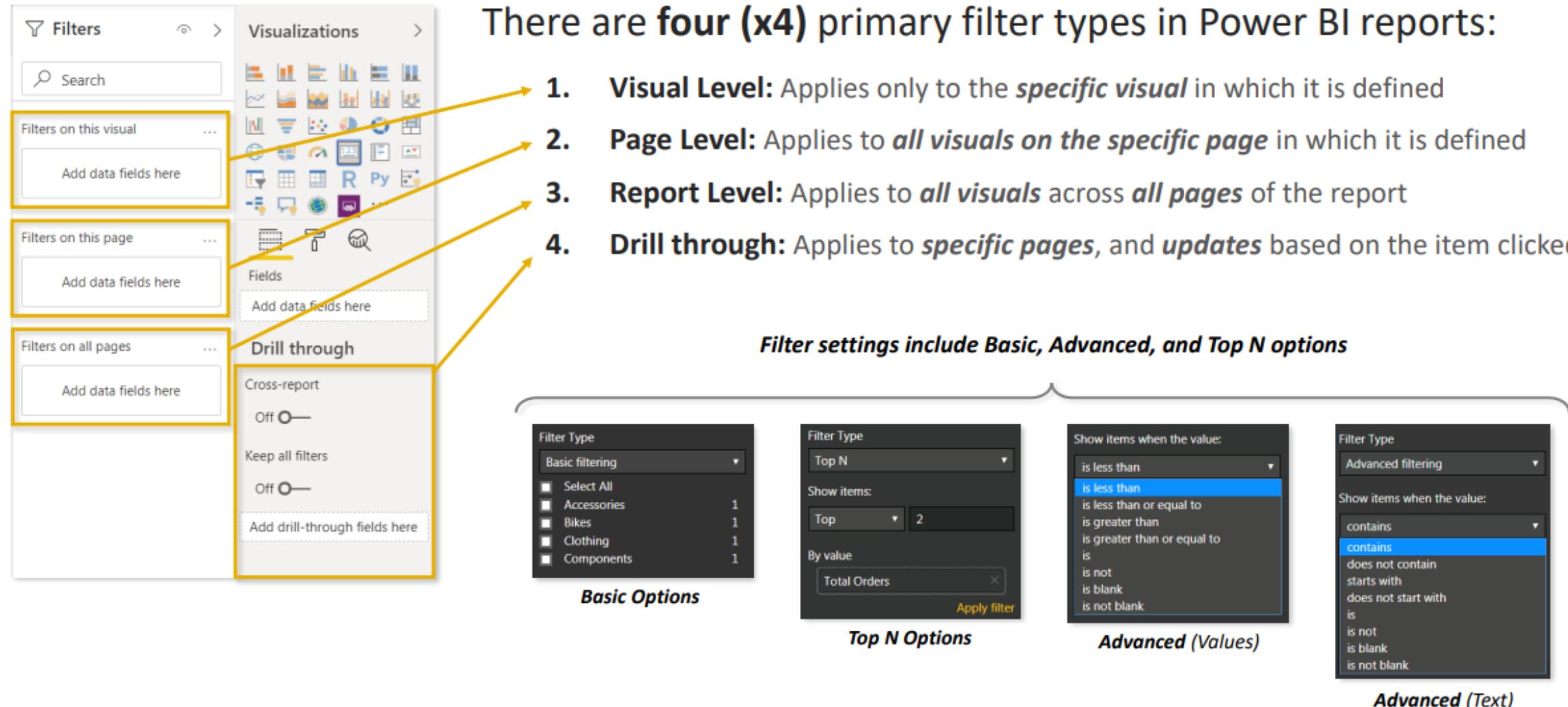


FILTERING OPTIONS

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. **Drill through:** Applies to *specific pages*, and *updates* based on the item clicked

Filter settings include Basic, Advanced, and Top N options



მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

მონაცემთა ვიზუალიზაცია; მონაცემების გაფილტვრა Slicer-ის გამოყენებით

ძირითადი მეთოდების ჩვენება Cards და KPI ვიზუალებით;

Geospatial მონაცემების ვიზუალიზაცია რუკების გამოყენებით;

მონაცემების ვიზუალიზაცია Treemaps-ის გამოყენებით;

ტენდენციების ჩვენება ხაზოვანი და რეგიონალური დიაგრამებით

მონაცემების გაფილტვრა Slicer-ის გამოყენებით

The screenshot shows the Microsoft Power BI desktop interface. At the top, the ribbon menu is visible with tabs: File, Home, Insert, Modeling, View, Optimize, and Help. The Home tab is selected, indicated by a green underline. Below the ribbon, the 'Clipboard' section is highlighted with an orange arrow, showing a 'Get workbook' option. The 'Data' section contains icons for OneLake data hub, SQL Server, and Enter data, along with a 'Recent sources' dropdown. The 'Queries' section includes 'Transform data' and 'Refresh data' with dropdowns, and 'New visual' (with 'Text box' and 'More visuals' options). The 'Insert' section has 'New measure' and 'Quick measure' buttons, along with 'Calculations' and 'Sensitivity' dropdowns. The 'Sensitivity' section includes 'Publish' and 'Share' buttons. The main workspace displays six visualizations: a 2x2 grid for 'Year' (2010, 2011, 2012, 2013), a 2x2 grid for 'Month' (January, February, April, May, March, June), a large text visual '14.43M' for 'Totalsales', a card visual for 'Sum of SalesAmount and Target by Date' showing '41.27' and 'Goal: 13.30 (+210.3%)', a bar chart for 'Totalsales by Gender' comparing Female (7.29M) and Male (7.14M), and a bar chart for 'Totalsales by Year and Month' showing sales volume by month from December 2011 to February 2013. To the right, the 'Visualizations' pane is open, showing a list of available visual types like 'Build visual', 'Slicer', and various chart and table icons. The 'Filters' section is also visible. At the bottom, the 'Values' section is open with options for 'Add data fields here', 'Drill through' (with 'Off' selected), 'Cross-report', 'Keep all filters' (with 'On' selected), and 'Add drill-through fields here'.

The screenshot shows the 'Visualizations' pane in Power BI. At the top, the title 'Visualizations' is followed by a 'Build visual' button and a 'More' button (indicated by three dots and a double arrow). Below this is a toolbar with three icons: a grid icon, a brush icon, and a magnifying glass icon. The main area contains a grid of 30 icons representing different types of visualizations, including bar charts, line graphs, maps, and other data representation tools. A large orange triangle points downwards from the top of the visualization grid towards the 'Field' section below. At the bottom, there is a search bar with the placeholder 'Date' and 'Year', and a set of small navigation icons.

Fields-ს ჩანართში
ვაგდებთ იმ სვეტს,
რომელიც გვინდა Slicer-
ზე გავაკტიუროთ

ძირითადი მეთოდების ჩვენება Cards ვიზუალებით

The screenshot shows the Power BI desktop application with the following interface elements:

- Top Bar:** Home, Insert, Modeling, View, Optimize, Help.
- Clipboard:** Paste, Cut, Copy, Format painter, Clipboard.
- Data:** Get data, Excel workbook, OneLake data hub, SQL Server, Enter data, Dataverse, Recent sources.
- Insert:** New visual, Text box, More visual, Refresh data, Insert.
- Calculations:** New measure, Quick measure, Sensitivity.
- Share:** Publish, Sensitivity.
- Visualizations:** A large area displaying six visualizations:
 - Year: A 2x2 grid showing 2010, 2012, 2011, 2013.
 - Month: A 3x2 grid showing January, April, February, May, March, June.
 - A large text visualization showing "14.43M" with the subtitle "Totalsales".
 - A card visualization showing "41.27" with the subtitle "Goal: 13.30 (+210.3%)".
 - A bar chart titled "Totalsales by Gender" comparing Female (7.29M) and Male (7.14M) sales.
 - A bar chart titled "Totalsales by Year and Month" showing monthly sales from December 2010 to February 2013.
- Visualizations pane:** Build visual, Filters, Values, Drill through, Cross-report, Keep all filters, Add drill-through fields here.

The screenshot shows the 'Visualizations' pane in Power BI. At the top, the title 'Visualizations' is followed by a 'Build visual' button and a 'More' button (indicated by three dots). Below this is a large icon representing a matrix or grid. A horizontal line with a downward arrow is positioned to the left of a grid icon. The main area contains a 5x5 grid of smaller icons representing different types of charts and data visualizations, including bar charts, line graphs, pie charts, and more complex data grid and map icons. At the bottom of the pane, there is a section labeled 'Fields' and a search bar containing the text 'Totalsales'.

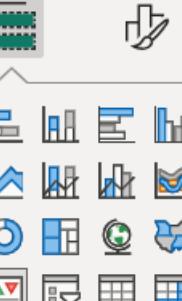
Fields-ს ჩანართში
ვაგდებთ იმ Measure-
ს, რომელიც გვინდა
Cards-ში
გამოვიტანოთ.

ძირითადი მეთოდების ჩვენება KPI ვიზუალებით;

The screenshot shows the Microsoft Power BI desktop interface. The ribbon menu includes File, Home, Insert, Modeling, View, Optimize, and Help. The Home tab is selected, displaying various data sources (Clipboard, Get data, OneLake data hub, SQL Server, Enter data, Dataverse, Recent sources) and tools (Transform, Refresh, New visual, Text box, More visuals, Insert, New, Quick, Calculations, Sensitivity, Publish, Share). The main workspace contains six visualizations: a 2x2 grid for years (2010-2013), a 2x2 grid for months (January-June), a large text visualization '14.43M' with 'Totalsales' below it, a KPI visualization '41.27' with 'Goal: 13.30 (+210.3%)', a bar chart for 'Totalsales by Gender' (F: 7.29M, M: 7.14M), and a bar chart for 'Totalsales by Year and Month' showing monthly sales from December 2010 to February 2013. The Visualizations pane on the right lists 'Build visual', 'Filters', 'KPI' (selected), 'R', 'Py', and '...' options. It also includes sections for 'Values' (Add data fields here), 'Drill through' (Off), 'Cross-report', 'Keep all filters' (On), and 'Add drill-through fields here'.

Visualizations >

Build visual



Value

Sum of SalesAmount ▾ X

Trend axis

Date ▾ X

Target

Target ▾ X

Trend axis -აჩვენებს ტენდენციის დერძს, როგორც KPI ვიზუალის ფონს, ჩვენს შემთხვევაში გამოყენებულია თარიღი.

Value - ფაქტიურად დამდგარი შედეგი
Target - სამიზნე მაჩვენებელი

მონაცემების ვიზუალიზაცია Treemap-ის გამოყენებით;

The screenshot shows a Microsoft Power BI interface with a ribbon at the top. The ribbon tabs are: File, Home (selected), Insert, Modeling, View, Optimize, Help, Format, and Data / Drill. The Home tab has several icons: Paste, Cut, Copy, Format painter, Get data, Excel workbook, OneLake data hub, SQL Server, Enter data, Dataverse, Recent sources, Transform data, Refresh data, New visual, Text box, More visuals, New measure, Quick measure, Sensitivity, Publish, and Share. Below the ribbon is a 'Visualizations' pane on the right. It includes a 'Build visual' section with a 'Treemap' icon highlighted by a red arrow, and sections for 'Filters', 'Category' (set to 'Gender'), 'Details' (set to 'EnglishEducation'), 'Values' (set to 'Totalsales'), and 'Toolips' (with a placeholder 'Add data fields here'). The main area displays a dashboard with four cards and two visualizations. The cards are: 'Year' (2010, 2011, 2012, 2013), 'Month' (January, February, April, May, March, June), 'Totalsales' (14.43M), and 'Sum of SalesAmount and Target by Date' (41.27, Goal: 13.30 (+210.3%)). The visualizations are: 'Totalsales by Gender and EnglishEducation' (Treemap chart) and 'Totalsales by Year and Month' (Bar chart).

ტენდენციების ჩვენება ხაზოვანი და რეგიონალური დიაგრამებით

The screenshot shows the Microsoft Power BI desktop interface with the following elements:

- File, Home, Insert, Modeling, View, Optimize, Help, Format, Data / Drill** tabs are visible in the top navigation bar.
- Clipboard** and **Share** buttons are located on the far right of the top bar.
- Data** section of the ribbon includes buttons for **Get data** (with sub-options for **OneLake data hub**, **SQL Server**, **Enter data**, and **Recent sources**), **Transform** (with sub-options for **Refresh data** and **Queries**), **Insert** (with sub-options for **New visual**, **Text box**, **More visuals**), **Calculations** (with sub-options for **New measure** and **Quick measure**), and **Sensitivity** (with sub-options for **Sensitivity** and **Share**).
- Visualizations** pane on the right side contains:
 - Build visual** button.
 - Filters** section with a dropdown menu showing **Line chart** selected.
 - Visuals** section showing a grid of visualization icons.
 - X-axis** and **Y-axis** sections for the line chart, with **Gender** selected for the legend.
 - Legend** section showing **Gender**.
- Visuals** on the main canvas:
 - A **Matrix** visual titled **Year** showing a 2x2 grid of years (2010, 2011, 2012, 2013).
 - A **Matrix** visual titled **Month** showing a 3x2 grid of months (January, February, March, April, May, June).
 - A **Text** visual titled **14.43M** with the subtitle **Totalsales**.
 - A **Card** visual titled **Sum of SalesAmount and Target by Date** showing the value **41.27** and the goal **13.30 (+210.3%)**.
 - A **Matrix** visual titled **Totalsales by Gender and EnglishEducation** showing sales by gender (F, M) and education level (Bachelor's, Graduate Degree, Partial College, High School, Partial High School).
 - A **Line** visual titled **Totalsales by Year, Month and Gender** showing sales over time (Jan 2011 to Jan 2013) for gender F (blue line) and M (light blue line).

Geospatial მონაცემების ვიზუალიზაცია რუკების გამოყენებით;

The screenshot shows a Microsoft Power BI desktop interface. At the top, the ribbon menu is visible with tabs: File, Home, Insert, Modeling, View, Optimize, Help, Format, and Data / Drill. The Home tab is selected. The main area displays a dashboard with four visualizations:

- A card showing a year-over-year comparison with four boxes: 2010, 2011, 2012, and 2013. An orange arrow points to the 2011 box.
- A large card displaying the total sales value: **14.43M** (Totalsales).
- A card showing the **Sum of SalesAmount and Target by Date** with a value of **41.27** and a goal of **13.30 (+210.3%)**.
- A map visualization titled "Sum of Sales by Country" showing sales distribution across continents.
- A line chart titled "Totalsales by Year, Month and Gender" showing sales trends from Jan 2011 to Jan 2013, categorized by gender (F and M).

On the right side, the **Visualizations** pane is open, showing a list of visualization types with icons and a search bar. An orange arrow points to the "Map" icon. The pane also includes sections for **Filters**, **Location** (set to "Country"), **Legend** (set to "Add data fields here"), **Latitude** (set to "Add data fields here"), **Longitude** (set to "Add data fields here"), **Bubble size** (set to "Sum of Sales"), and **Toolips**.

მადლობა ყურადღებისთვის



ანალიტიკური ინსტრუმენტი – Microsoft Power BI Desktop

გასავლელი საკითხები

მონაცემთა ვიზუალიზაცია; ტენდენციური ხაზების და პროგნოზების დამატება; Gauge Chart;

ურთიერთქმედების რედაქტირება; Drillthrough ფილტრების დამატება; სანიშნების გამოყენება;

სცენარების ტესტირება "What-If" პარამეტრებით; როლების მართვა და დათვალიერება;

პროექტირება ტელეფონისა და დესკტოპის მნახველებისთვის; გამოქვეყნება Power BI Server-ში; Power BI მონაცემთა ვიზუალიზაციის საუკეთესო პრაქტიკა;

EDITING REPORT INTERACTIONS

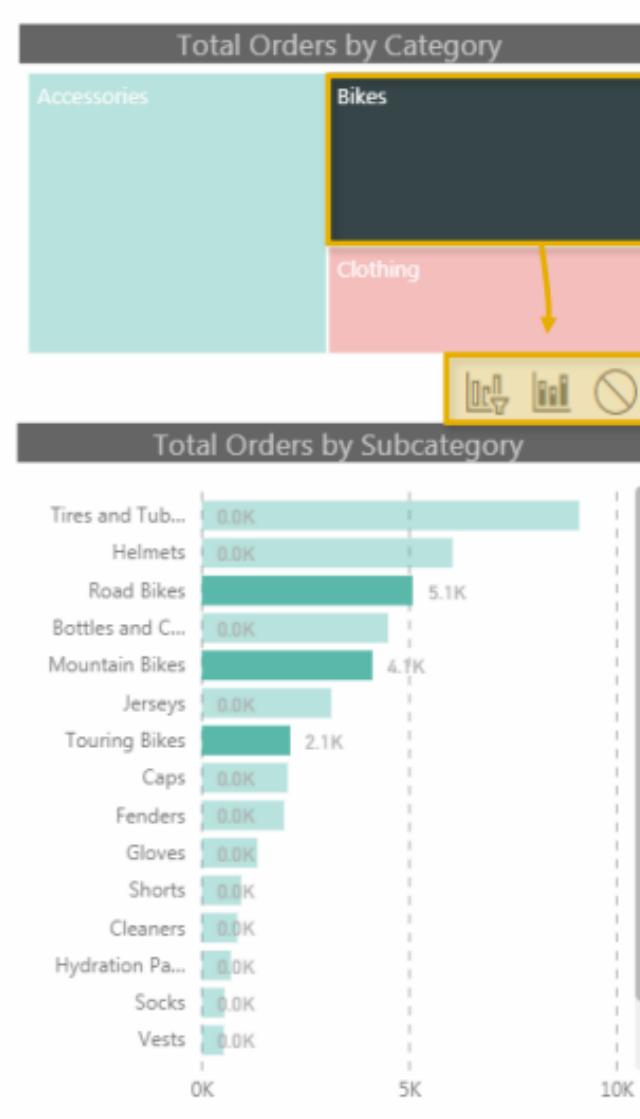
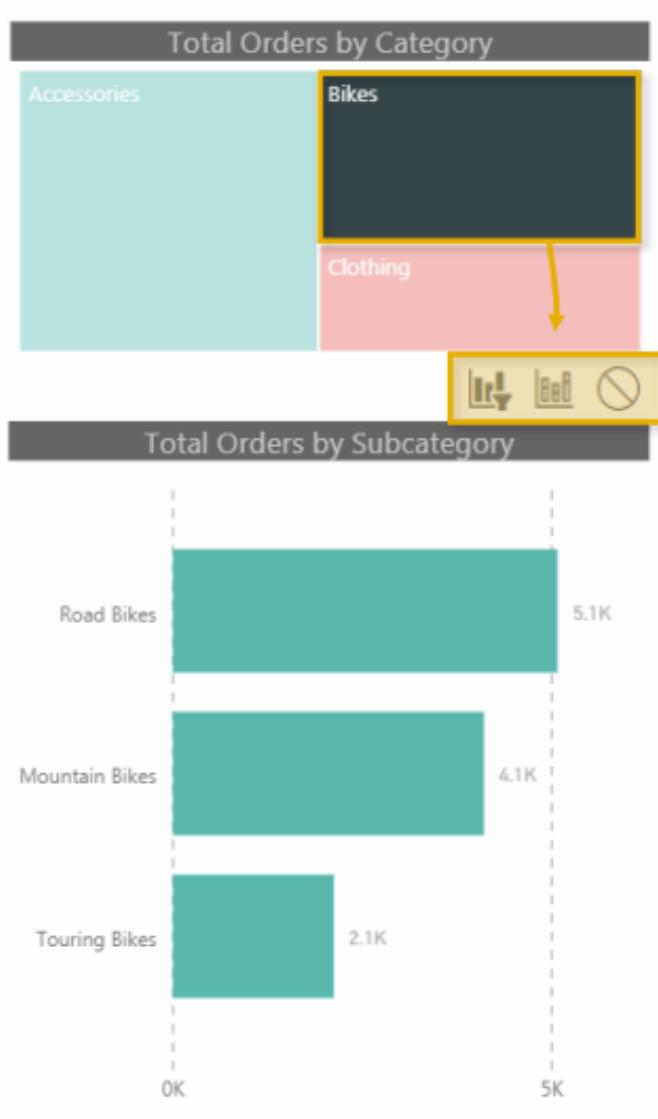
The screenshot illustrates a Power BI report with various visualizations and their interactions. At the top, a ribbon menu is shown with the 'Format' tab selected. A yellow box highlights the 'Edit interactions' button in the 'Format' tab's ribbon. A callout arrow points from this button to a 'Timeline' visual, which displays two date ranges: '1/1/1998' and '12/31/1998'. A yellow box highlights the 'Format' tab for this visual. Another callout arrow points from the 'Edit interactions' button to a 'Product matrix' table below. The table has columns for 'Product Brand', 'Quantity', 'Retail Price', 'Product Cost', and 'Profit'. A yellow box highlights the 'Format' tab for this table. A third callout arrow points from the 'Edit interactions' button to a 'Country slicer' with options for 'CANADA', 'MEXICO', and 'USA'. A yellow box highlights the 'Format' tab for this slicer. A fourth callout arrow points from the 'Edit interactions' button to a 'Map' visualization showing the Pacific Northwest region of North America, with cities like Vancouver, Victoria, and Bellingham marked. A yellow box highlights the 'Format' tab for this map. The report also features three summary cards: 'SELECT COUNTRY:' with values '\$5,798', '\$17K', and '\$64K' (MTD Profit, QTD Profit, and YTD Profit respectively), and a 'Bing' map of the same region.

| 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 |
| Nationeel | 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 |
| Compartant | 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* ()

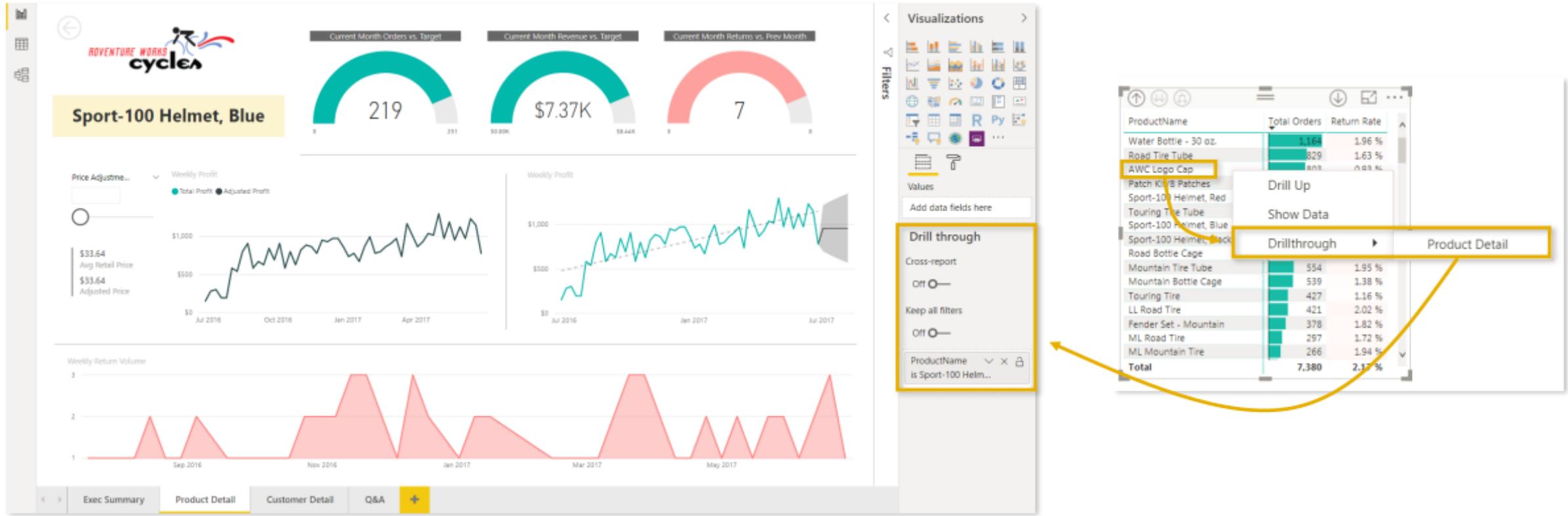
EDITING REPORT INTERACTIONS (CONT.)



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

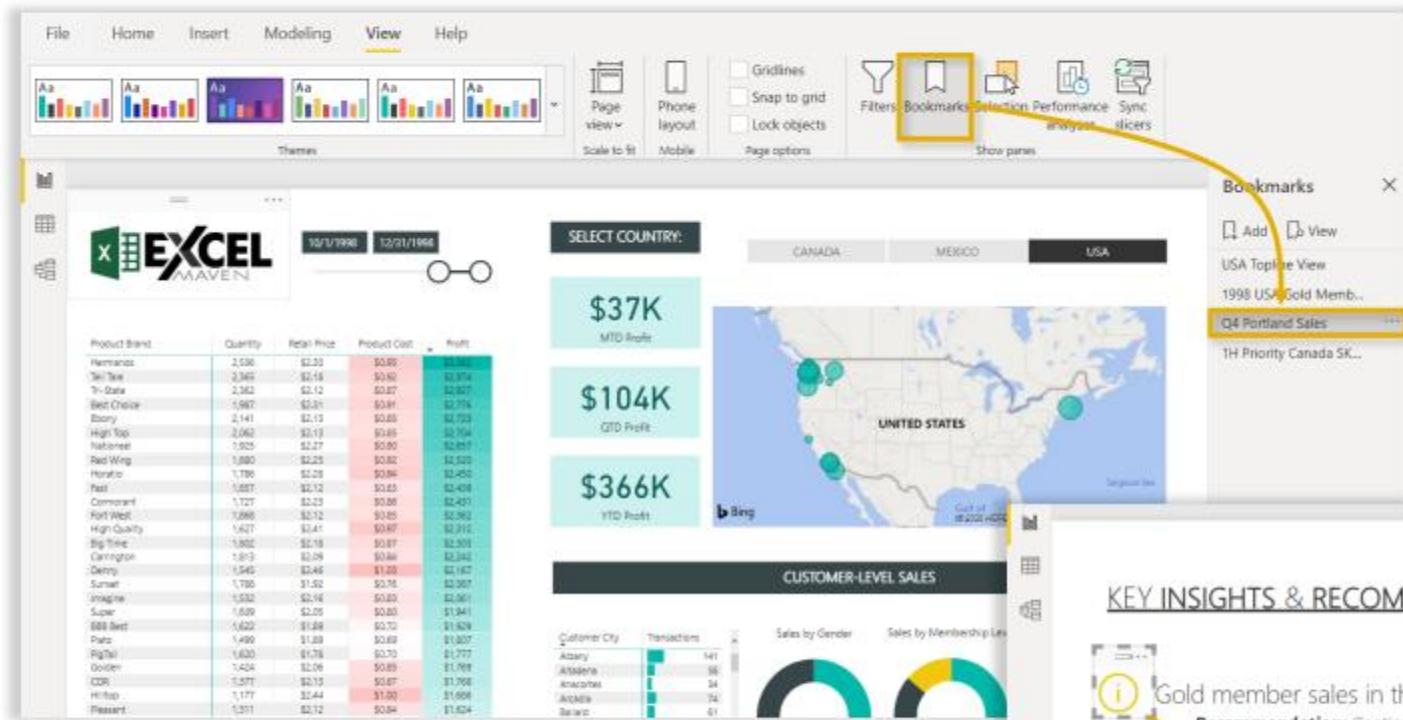
DRILLTHROUGH FILTERS



Drill through 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 ("Sport-100 Helmet, Blue" shown in the example above)

ADDING & LINKING BOOKMARKS



File Home Insert Modeling View Help

Themes

Page view Scale to fit Phone layout Page options

Gridlines Snap to grid Lock objects

Filters Bookmarks Selection Performance Sync Slicers

Show pane

Bookmarks

Add View

USA Topline View
1998 USA Gold Memb.
Q4 Portland Sales
1H Priority Canada SK...

SELECT COUNTRY: CANADA MEXICO USA

\$37K MTD Profit

\$104K GTD Profit

\$366K YTD Profit

UNITED STATES

CUSTOMER-LEVEL SALES

Customer City Transactions

| Customer City | Transactions |
|---------------|--------------|
| Albany | 141 |
| Atlanta | 36 |
| Acacore | 24 |
| Alaska | 74 |
| Balance | 61 |

Sales by Gender

Sales by Membership Level

Product Brand

| Product Brand | Quantity | Retail Price | Product Cost | Profit |
|---------------|----------|--------------|--------------|--------|
| Permanence | 2,036 | \$2.10 | \$0.85 | \$1.25 |
| Tri-State | 2,365 | \$2.18 | \$0.82 | \$1.36 |
| Red Wing | 2,362 | \$2.12 | \$0.87 | \$1.25 |
| Bed Custer | 1,987 | \$2.21 | \$0.81 | \$1.27 |
| Ebony | 2,141 | \$2.13 | \$0.85 | \$1.28 |
| High Top | 2,062 | \$2.10 | \$0.85 | \$1.25 |
| Nationell | 1,925 | \$2.27 | \$0.80 | \$1.47 |
| Red Wing | 1,880 | \$2.25 | \$0.82 | \$1.43 |
| Horizon | 1,798 | \$2.28 | \$0.84 | \$1.44 |
| Red | 1,687 | \$2.12 | \$0.83 | \$1.29 |
| Commerce | 1,727 | \$2.23 | \$0.88 | \$1.35 |
| Fort West | 1,686 | \$2.12 | \$0.85 | \$1.27 |
| High Quality | 1,627 | \$2.41 | \$0.87 | \$1.54 |
| Big Three | 1,882 | \$2.18 | \$0.87 | \$1.31 |
| Carrington | 1,813 | \$2.09 | \$0.86 | \$1.23 |
| Denny | 1,545 | \$2.48 | \$1.08 | \$1.40 |
| Sunrise | 1,789 | \$2.02 | \$0.87 | \$1.15 |
| Imagine | 1,532 | \$2.18 | \$0.89 | \$1.29 |
| Super | 1,689 | \$2.05 | \$0.83 | \$1.22 |
| BBB Bed | 1,622 | \$1.88 | \$0.70 | \$1.18 |
| Patio | 1,499 | \$1.89 | \$0.69 | \$1.20 |
| Right | 1,620 | \$1.78 | \$0.70 | \$1.07 |
| Golden | 1,424 | \$2.06 | \$0.89 | \$1.17 |
| CDR | 1,577 | \$2.15 | \$0.87 | \$1.28 |
| High | 1,177 | \$2.44 | \$1.80 | \$1.64 |
| Present | 1,311 | \$2.12 | \$0.84 | \$1.28 |

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!

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**”

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

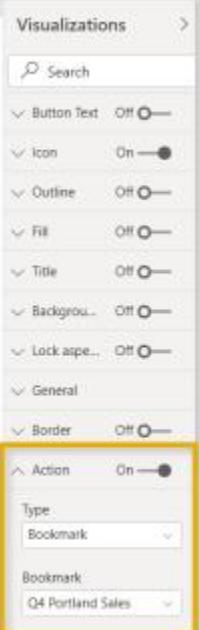
Portland sales were strong in Q4, led by High Top, Ebony, and Red Wing brands

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

Back to Topline View



Visualizations

Search

Button Text: Off

Icon: On

Outline: Off

Fill: Off

Title: Off

Background: Off

Lock aspect: Off

General

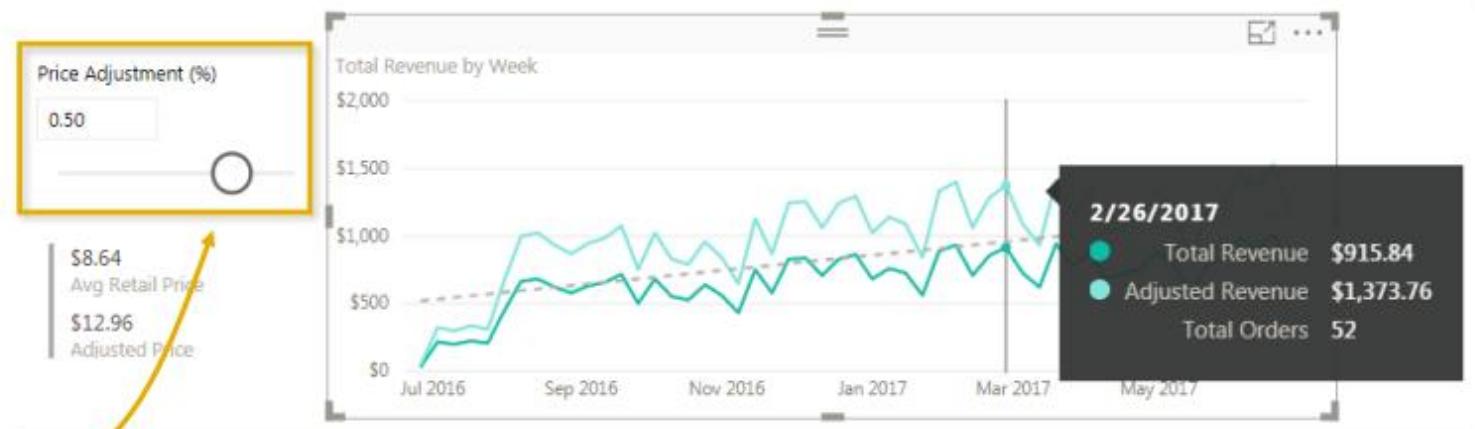
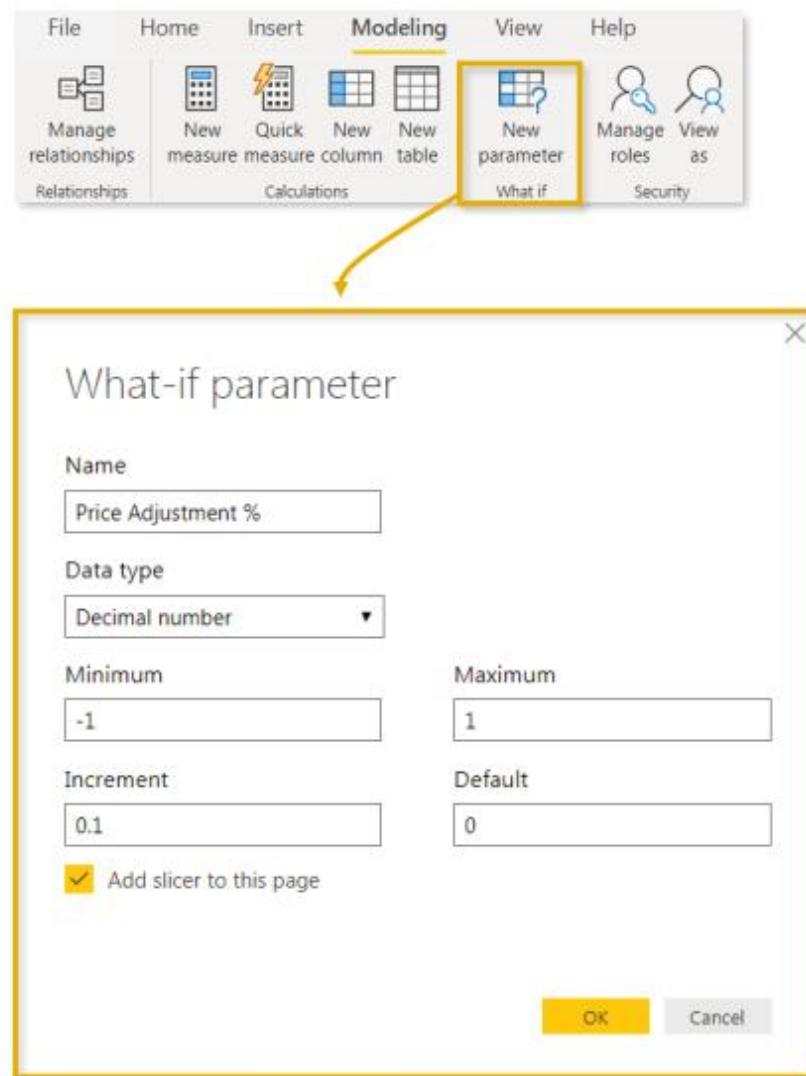
Border: Off

Action: On

Type: Bookmark

Bookmark: Q4 Portland Sales

“WHAT-IF” PARAMETERS



“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 Microsoft Power BI. It shows the Power BI ribbon with the 'Modeling' tab selected, and the 'Manage roles' option highlighted. A callout box points to the 'Manage roles' dialog, which lists roles (Europe, North America, Pacific) and their corresponding tables and DAX filters. Another callout box points to the 'View as roles' dialog, showing checkboxes for Europe, North America, and Pacific, with Europe selected. The main area shows a report for 'Adventure Works Cycles' with a 'View as roles' header indicating the report is being viewed as Europe. The report includes a 'Orders by Category' treemap, a 'Product Items' table, and various charts and maps.

Manage roles

Roles

| Role | Tables |
|---------------|--|
| Europe | AW_Calendar, AW_Customer_Lookup, AW_Territory_Lookup |
| North America | AW_Product_Lookup, AW_Product_Subcategory_Lookup, AW_Returns_Data, AW_Sales_Data, AW_Sales_Data_Folder |
| Pacific | AW_Product_Lookup, AW_Product_Subcategory_Lookup, AW_Returns_Data, AW_Sales_Data, AW_Sales_Data_Folder |

Table filter DAX expression

[Continent] = "Europe"

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

None

Other user

Europe

North America

Pacific

Now viewing report as: Europe

Stop viewing

ADVENTURE WORKS cycles

1/1/2015 6/1/2017

Current Month Revenue

\$627.3K

Goal: \$550.7K (+13.9%)

Current Month Orders

644

Goal: 625 (+3.04%)

Most Ordered Product: Water Bottle - 30 oz.

Top Revenue Product: Mountain-200 Silver, 46

Product Items

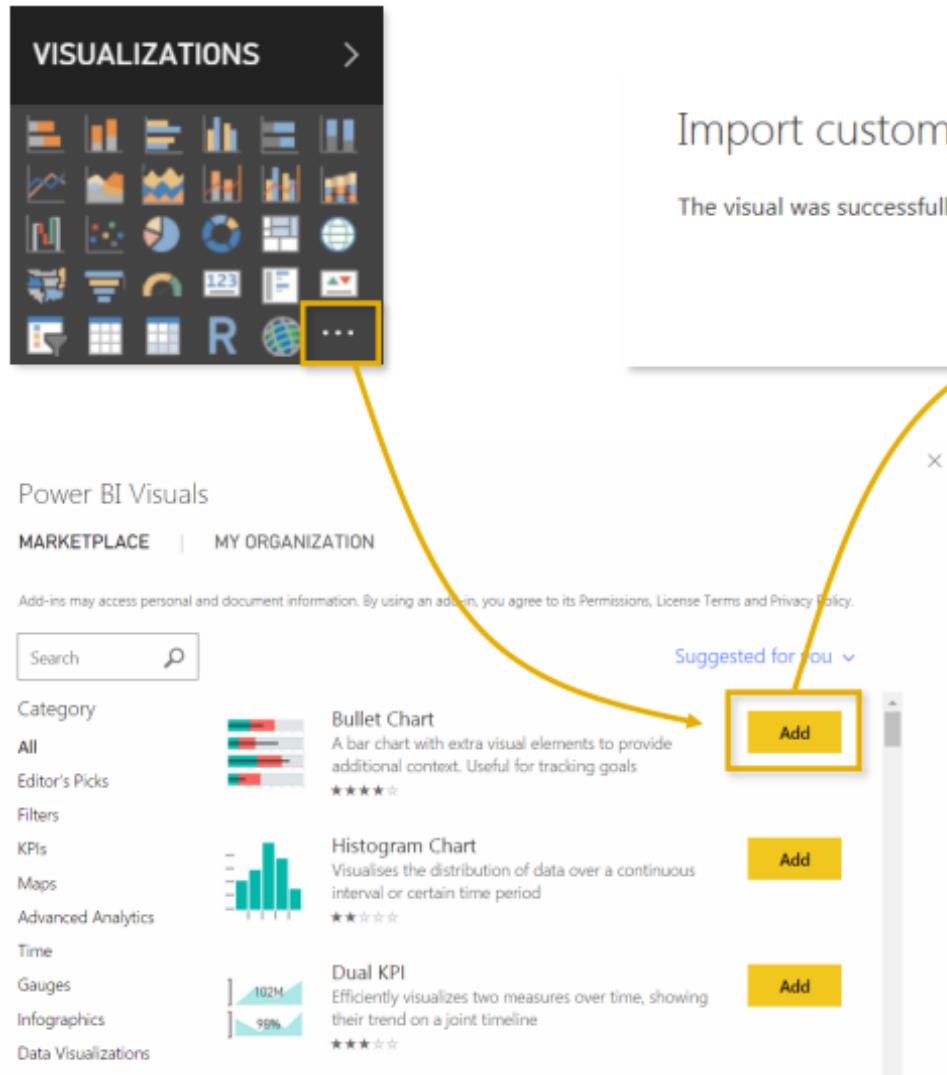
| Product Item | Total Orders | Return Rate |
|----------------------------|--------------|-------------|
| Water Bottle - 30 oz. | 120 | 1.66% |
| Road Tire - 700 | 125 | 1.63% |
| AWC Logo Cap | 103 | 0.93% |
| Rock 'n' Roll Patch | 788 | 1.07% |
| Scout-120 Helmet, Red | 753 | 2.78% |
| Touring Tire - 700 | 702 | 1.23% |
| Scout-120 Helmet, Blue | 688 | 3.23% |
| Scout-120 Helmet, Black | 626 | 1.67% |
| Road Bike Cage | 562 | 1.89% |
| Mountain Bike Cage | 554 | 1.81% |
| Mountain Bike Cage | 518 | 1.18% |
| Touring Tire | 427 | 1.18% |
| 12 Road Tire | 421 | 2.02% |
| Fender Set - Mountain | 378 | 1.82% |
| MT Road Tire | 297 | 1.72% |
| MT Mountain Tire | 288 | 1.94% |
| MT Mountain Tire | 204 | 1.48% |
| Mountain-200 Silver, 46 | 188 | 1.31% |
| Mountain-200 Black, 46 | 188 | 1.06% |
| 12 Mountain Tire | 159 | 2.09% |
| Mountain-200 Silver, 38 | 129 | 2.85% |
| Bike Watch - Dissolve | 107 | 2.33% |
| Mountain-200 Black, 42 | 102 | 3.85% |
| Mountain-200 Silver, 38 | 100 | 3.33% |
| Long-Sleeve Logo Jersey, M | 101 | 4.33% |
| Hi-Rise Tie | 100 | 3.06% |
| Mountain-200 Silver, 42 | 98 | 1.23% |
| Hydration Pack - 70 oz. | 147 | 4.08% |
| Long-Sleeve Logo Jersey, L | 147 | 2.71% |
| Long-Sleeve Logo Jersey, S | 102 | 2.03% |
| Total | 7,389 | 2.17% |

View product Details

Map of Europe showing locations like London, Berlin, Paris, and Rome.

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

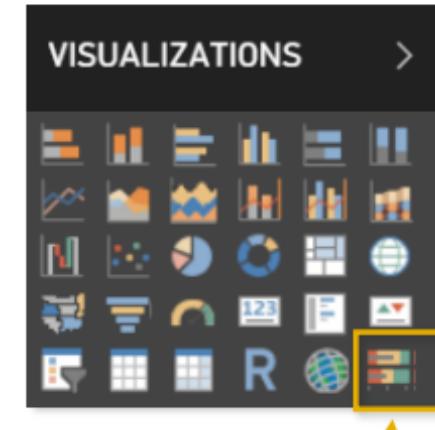
IMPORTING CUSTOM VISUALS



Import custom visual

The visual was successfully imported into this report.

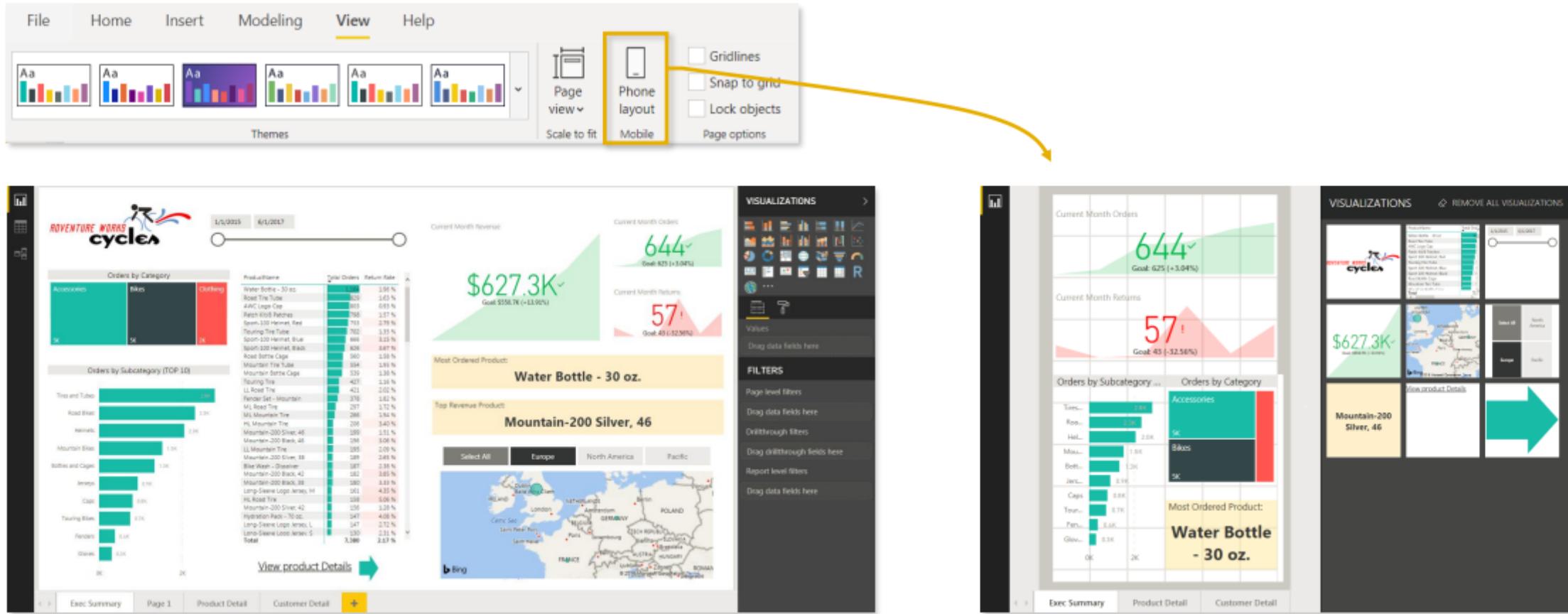
OK



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



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)*

მადლობა ყურადღებისთვის