DAX for Power BI Professionals: From Basic Formulas to Complex Analytics

DAX for Power BI Professionals: From Basic Formulas to Complex Analytics

©2024 – AMZ Consulting Pty Ltd

All rights reserved. No part of this publication titled "DAX for Power BI Professionals: From Basic Formulas to Complex Analytics" may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of the authors, except for the use of brief excerpts in a book review or scholarly journal.

The content presented herein is based on the version of Power BI current as of July 2023. While considerable effort has been made to ensure the accuracy and currency of the information contained in this book, the author and publisher make no express or implied representations or warranties regarding the completeness, accuracy, or fitness for any particular purpose of the content. The author and publisher shall not be held liable for any errors, omissions, or any consequences from the application of the information contained in this book and assume no responsibility for any loss or damage caused as a result of any statement contained herein. This book is provided 'as is', and readers are advised that the material contained herein should be used in conjunction with professional advice.

Author: Ali Noorani, Muneeba Sirshar

Marketing Design: Queper Pty Ltd

Website support: Sadia Batool

Published by: self-publishing by AMZ Consulting Pty Ltd

ISBN: 9798333409768

DAX for Power BI Professionals:

From Basic Formulas to Complex Analytics

By

Ali Noorani & Muneeba Sirshar

"Data is the new oil. It’s valuable, but if unrefined it cannot really be used"

– Clive Humby

Contents

Framework for Understanding

Power BI's superiority lies in its agile approach to data visualization and business intelligence. It empowers users to move beyond the constraints of traditional spreadsheets and databases, facilitating a dynamic environment where data becomes interactive, and insights are readily accessible. The tool's robust feature set is designed to cater to both the novice user and the seasoned data analyst, ensuring that sophisticated analyses are not the exclusive domain of experts. With Power BI, the democratization of data analysis is not a promise; it is a present reality.

As organizations navigate through the torrents of Big Data, Power BI offers a lighthouse of clarity. It is not just about presenting data; it is about presenting it in a way that is immediately understood and transformed into a strategic advantage. In this narrative, Power BI is not merely a tool; it is the craftsman’s instrument, essential in sculpting raw data into the foundations of success.

Raise Your Analytics Game:

Begin your journey with Raise Your Analytics Game to build a solid foundation in data analytics. This essential first step ensures you grasp the basics and are well-prepared for the next level. Once you've mastered the fundamentals, dive into Designing Powerful Reports in Power BI. Here, you'll learn to create reports that are not only functional but also visually stunning and impactful. This book will transform your ability to communicate data insights effectively. For those who are ready to elevate their skills even further, Mastering Power BI awaits. This advanced guide offers deeper techniques and insights to take your proficiency to new heights.

The Power of Active Learning

Embarking on the journey to master Power BI DAX is akin to learning a new language; it is through practice and application that proficiency is achieved. This book is more than just a guide; it is a workbook that comes alive when you interact with it. The text is designed not just to be read, but to be used. We have meticulously prepared exercise files for each chapter, providing a structured path to apply your newfound knowledge. As you progress through the pages, these exercises will challenge you to solve problems, create visualizations, and discover insights on your own. It is through this process of active learning that the concepts of Power BI will move from abstract ideas to concrete skills.

From Learning to Mastery

By engaging directly with the exercises, you solidify your understanding and develop the kind of muscle memory that only comes from hands-on experience. These exercises are not just add-ons; they are integral to the learning experience, crafted to reinforce the lessons and ensure retention of the material. As you work through the various scenarios and datasets, you'll gain a level of familiarity with Power BI that reading alone cannot provide. This book encourages you to not just passively absorb information but to become an active participant in your educational

journey. In doing so, you will be able to capture the full value of the book and emerge not just with knowledge, but with competence and confidence in using Power BI to its full extent.

Your Comprehensive Power BI Toolkit

Unlock a complete, interactive learning suite with "Mastering Power BI":

- **Rich Datasets**: Dive into a diverse array of real-world datasets, ready for you to dissect and explore.
- **Exercise Files**: Tackle hands-on exercises, each crafted to reinforce chapter concepts with practical application.
- **Interactive Quiz**: Challenge your comprehension with a quiz that provides instant feedback.
- **Certification of Mastery**: Earn a shareable certificate upon completion, validating your expertise and dedication.

With each element, you'll build not just knowledge, but also confidence and credibility in the field of data analytics.

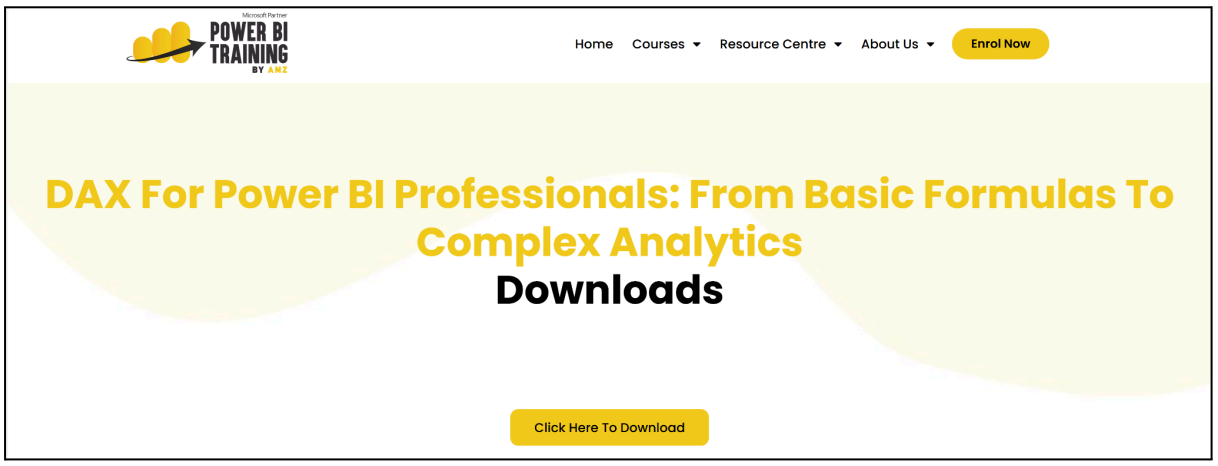
Need Assistance? We're Here to Help!

For any questions or support, simply email us at info@amzconsulting.com.au or reach out on LinkedIn. Our team is ready to assist you on your path to mastery.

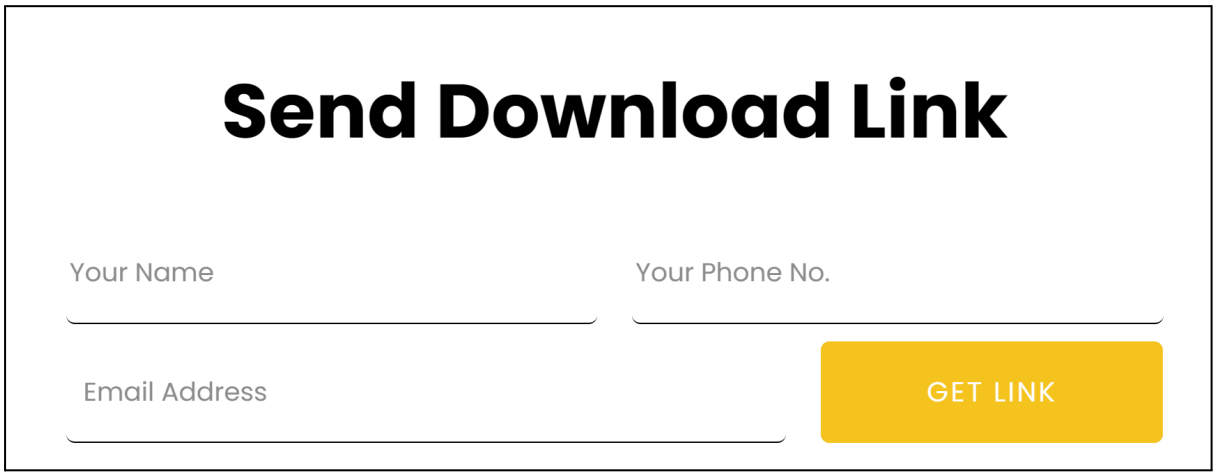
Getting Started with "DAX for Power BI Professionals"

Let's get you set up with the materials you need for this book:

1. Navigate to the page: https://powerbitraining.com.au/dax-for-power-bi-professionals-from-basic-formulas-to-complex-analytics/

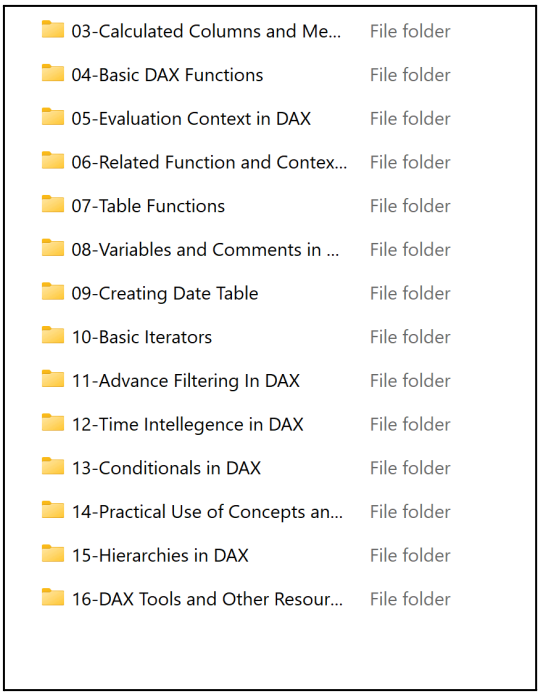


2. Look for the '**Click Here to Download**' button and click on it. A small window will pop up on the screen.
3. In the window, you'll need to type in your details, like your name and email address and click on **GET LINK.**



4. Look in your email inbox for a message from us. This email will have a link for you to download your resources. If you don't see it right away, it might be in your spam or junk mail folder.
5. Click the link in our email to download a zip file that contains all your resources.

6. Find the file you just downloaded and extract the contents.
7. Inside this main folder, you'll find smaller folders named after the chapters in this book.



Chapter 1: Getting to Know Data Analysis Expression (DAX) language

What will you learn in this chapter?

- ✔ What is DAX?
- ✔ Where is DAX used?
- ✔ What are the Data types in DAX
- ✔ DAX type handling/operator overloading
- ✔ DAX as Formula Language
- ✔ Vertipaq engine in DAX
- ✔ Difference between DAX and EXCEL functions
- ✔ Difference between DAX and M
- ✔ Best Practices in DAX
- ✔ Dos, Don'ts and Best Practices in DAX

What is DAX?

DAX Stands for **Data Analysis Expressions**. It is a formula language developed by Microsoft to interact with data in your model. DAX is a formula language and NOT a programming language.

DAX helps you to enrich your data by creating more information from the data that you already have. DAX is a collection of functions, operators, and constants that can be used in a formula, or expression, to calculate and return one or more values.

Here are some examples where simply importing data into Power BI will not work and DAX code will be the only solution:

- ✔ Analyze growth percentage across product categories.
- ✔ Finding profit margin percentage for products and product categories.
- ✔ Calculate year-over-year growth.
- ✔ Compare your brand's growth rate with that of competitors.
- ✔ Analyze your product's performance over different date ranges.

DAX formulas provide this capability and many other important capabilities as well.

Perquisites:

You should already be familiar with using Power BI desktop to import data and add fields to a report.

Where is DAX used?

DAX is used and applied in many Microsoft tools and platforms such as:

- ✔ Power BI
- ✔ Microsoft Power Pivot for Excel
- ✔ SSAS Tabular

All these business intelligence tools use the same underlying engine known as vertipaq.

DAX was developed by the SQL Server Analysis Services team at Microsoft as part of Project Gemini and released in 2009. It was developed with the first version of the PowerPivot for Excel 2010 as an add-in. Future versions of SSAS (both multidimensional & tabular models) will support DAX natively.

Here a list of resources to help you learn DAX:

- ✔ Microsoft Documentation: https://docs.microsoft.com/en-us/dax/
- ✔ Power BI DAX blogs: https://www.powerbitraining.com.au/category/dax/

What are the Data types in DAX

The data types in DAX can be divided into three categories: numerical, non-numerical and variant:

Numerical data types:

- ✔ Integer: The integer data type stores 64-bit value.
- ✔ Decimal: The decimal data type is stored as double-precision floating point value.
- ✔ Currency: The currency data type is internally stored as 64-bit integer value divided by 10,000. It represents four decimal points. Summing or subtracting Currency data types always ignores decimals beyond the fourth decimal point. The default format of the Currency data type includes the currency symbol.
- ✔ Date Time: This data type uses a floating-point number internally, wherein the integer corresponds to the number of days since December 30, 1899, and the decimal part identifies the fraction of the day. Hours, minutes, and seconds are converted to decimal fractions of a day.
- ✔ Boolean: This data time is used to express logical conditions such as TRUE() and FALSE().

Non-numerical data types:

- ✔ Binary objects: This data type is used to store non-structural information such as the images.
- ✔ Strings: The string data type in Power BI is case-insensitive and accent-sensitive. Every string in DAX is stored as a Unicode string, where each character is stored in 16 bits.

Variant Data Type:

The Variant data type is used for expressions that might return different data types, depending on the conditions. As an example consider the following expression.

```
IF ( [measure] > 0, 1, "N/A" )
```

A DAX measure and in general a DAX expression can have a variant data type. However, any column in a regular table cannot have a variant data type.

It is important to consider that formatting strings does not change the data type.

DAX type handling/operator overloading

Operators are not strongly typed in DAX. The results depend on the input operator and conversion happens for the input operands. Let's understand this using two examples:

Addition of two "string" inputs:

Let's create a simple measure described in the image below.

```
Addition Operator =
"5" + "4"
```

The results are visualized in a card visual.

Despite the two input operands being present in double quotes (which represents strings), they were added because of the addition operator between the operands.

Concatenation of two "numbers":

Let's create a simple measure described in the image below.

```
Concatenation Operator =
5 & 9
```

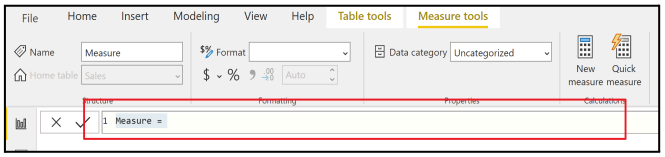
The results are visualized in a card visual.

Despite the two input operands being simple integers, they were concatenated because of the operator between the operands.

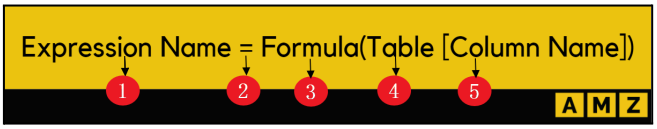
Pay attention to the undesired conversions

DAX as Formula Language

DAX is a formula language which means there is one formula call with many parameters. This function call can also contain other function call as parameters. For Power BI desktop, all the DAX code is typed in the formula bar shown below. The measure/column tools tab additionally provides all the information related to the measure or column respectively.

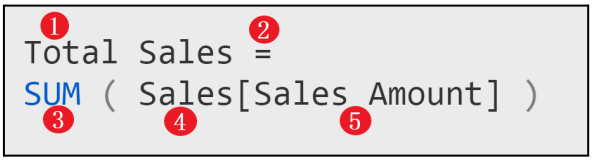


A DAX expression consists of a formula followed by a measure or a column reference. Let's look at the structure of a typical DAX expression.



1. This is the name of the expression that you are going to create.
2. Equal sign represents the beginning of DAX expression.
3. Formula represents the calculation that you want to perform.
4. Name of Table.
5. Name of Column.

Now let's map this to a simple example that calculates the sum of Sales Amount from the sales table using the SUM function.



Vertipaq engine in DAX

Vertipaq is the compression engine of Microsoft Power BI. As soon as the data is loaded in Power BI, Vertipaq engine performs a series of algorithms on the data to compress it as much as possible. This optimizes the memory footprint and DAX query time in Power BI.

There are three algorithms used by vertipaq engine to compress the data:

Value Encoding:

Value encoding discovers the mathematical relationship between different values in a column and uses this relationship for compression of data. The data is encoded in the first stage and then stored. In the second stage, the data is decoded and original values are returned. Value encoding is performed only for integers and not for strings and floating point values.

Dictionary or Hash Encoding:

Dictionary encoding builds a dictionary of the distinct values of a column and then it replaces the column values with indexes of the dictionary. Dictionary encoding therefore converts column values into integers only. This takes lesser memory and the workflow is optimized.

Run Length Encoding (RLE):

Run length encoding optimizes the size of the data set by avoiding repetitive values. RLE creates a relatively complex code structure containing the value only once and the number of contiguous rows containing that value.

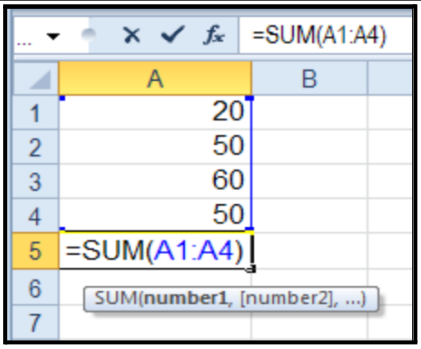
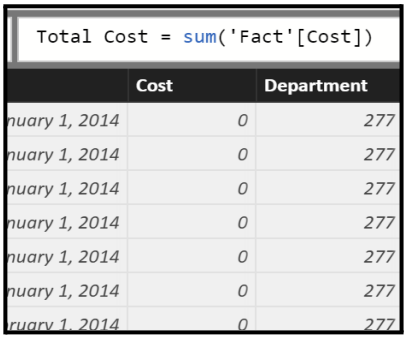
The model compression depends on various factors.

Factors Affecting Model Compression:

- ✔ Cardinality of a column. It is the number of distinct values that define the number of bits used for storage.
- ✔ The number of rows in the table.
- ✔ The data type of column. It is related to the dictionary size.
- ✔ The number of repetitions. This directly determines the distribution of the column.

Difference between DAX and EXCEL functions

Many DAX functions have the same name and general behavior as EXCEL functions but they work in completely different ways. Let's look at some significant differences between these two languages.

	Excel	**DAX**
Referencing	Cell references are used to define calculation	Column references are used to define calculations
Evaluation context	No evaluation context exists	Row and filter contexts exist
Variant data type	Supports Variant data type in a column	Does not support variant data type in a column
Date and time functions	Date and time functions return an integer	DAX date and time functions return date time value.
Example		

Difference between DAX and M

M and DAX, both are used in Power BI to transform, filter and analyze data. Although both of these languages look similar, they operate independently of each other. Here's a list of ways in which they differ.

	M Language	**DAX**
Functionality	Query language to Shape and Transform Data	Functional language to build calculations and model data
Usage	Used for Query-Time Transformations	Used for In-Memory Transformations
Scope	M is used to preprocess data inside the query editor.	Enriches data once the data is loaded into the data model.
Application	M is a mashup language that works on multiple data sources therefore uses commands to shape and transform the data.	DAX is a formula language therefore it uses formulas to develop new data columns and measures.
Suitability	M is best suited for ETL.	DAX is best suited for analytics.

Best Practices in DAX

DAX code can be error free and easier to read when the following practices are followed:

- ✔ Use descriptive names for variables, measures and columns.
- ✔ Follow a naming convention.
- ✔ Format your code.
- ✔ Use the correct data type for columns and measures.
- ✔ Never use table names for measure reference.
- ✔ Always use table names for column reference.
- ✔ Split your calculations in small blocks using variables.
- ✔ Write once and re-use measures.
- ✔ Remove unnecessary prefixes before importing the tables.
- ✔ Check the measures using matrix visuals.
- ✔ Improve code readability by using variables.
- ✔ Sort the alphabetic column with a suitable numerical column.

Dos, and Don'ts and Best Practices in DAX

Now that we have looked at some best practices, let's look at some technical aspects of writing DAX:

- ✔ Either create measures in the most appropriate table or have separate measures-only tables (we'll do this in the course).
- ✔ Don't use bi-directional filters until it's necessary.
- ✔ Avoid context transition for large iterators.
- ✔ Reduce nested calls.
- ✔ Use comments to document your code.
- ✔ Do not filter the whole table, filter selected columns.
- ✔ Break the measure code into interim parts.
- ✔ Do not break the column code into interim parts.
- ✔ Do not hide errors unless you are sure how to deal with the aftermath.
- ✔ Avoid double evaluation of the same expression by using variables.
- ✔ Avoid using spaces in table names, otherwise you will have to add single quotes to reference that table.
- ✔ Do use spaces in column names, it increases readability.
- ✔ Do use spaces in measure names, it increases readability.
- ✔ Calculated Columns can be expensive (they update when the data is refreshed).
- ✔ Measures contain an implicit CALCULATE().
- ✔ Avoid using the FILTER() function unnecessarily and avoid using it on your fact table, as it can perform poorly on large tables. (You will learn about this later)
- ✔ A standard filter expression in a CALCULATE() function performs better. (You will learn about this later)

Chapter 2: Data Modeling Recap

What will you learn in this chapter?

✔ What is Data Model

✔ Tables in a Data Model

✔ Fact vs Dimensions

✔ Schemas in Data Model

✔ Why star Schema is Preferred

✔ Filter Propagation

✔ Interpreting Mechanism for Filtering

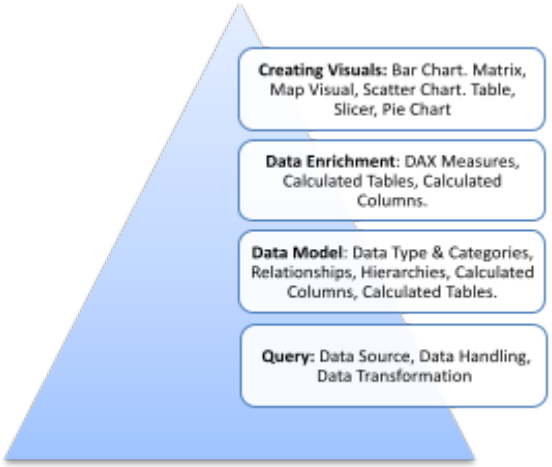
✔ Using Multiple Filters together

✔ Data Denormalization

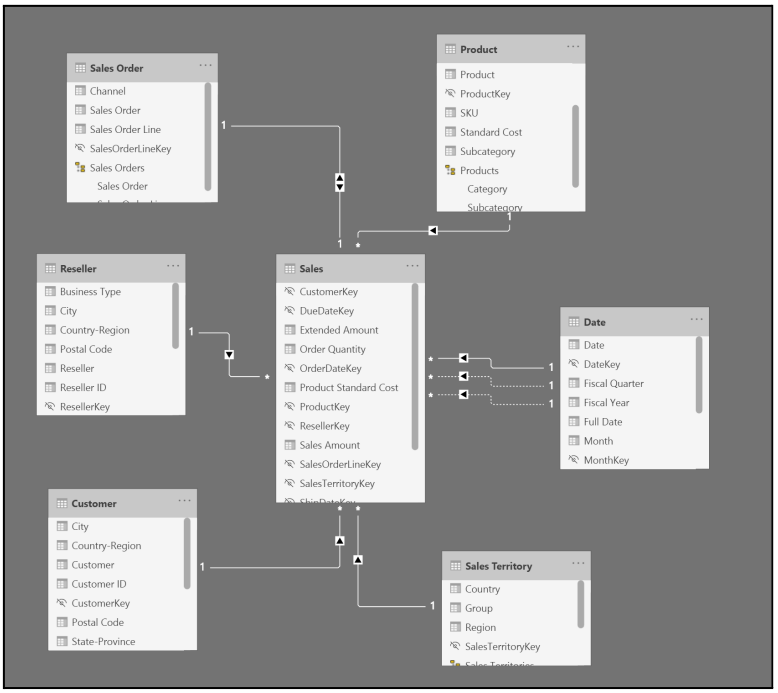
✔ Relationship Characteristics

What is Data Model

Data model refers to the ***abstract model*** that demonstrates the ***logical structure*** of data and the ***relationships*** that exist in the data. In Microsoft Power BI, data model refers to everything that is loaded from query. In terms of data, a good data model consists of the right number of tables with the right relationship between them. Let's look at some basics before digging into the data modeling concept.



A data model consists of ***Two or more tables related*** to each other. For example, let's consider the data model in the image. The sales table in the center is related to each of the other tables, i.e., Product, Sales Order, Reseller, Customer, Sales Territory and Date.



Tables in a Data Model

A typical data model consists of:

Fact Table / Data Table

If we **record a value** of an event or something that happened, it's called a fact and the complete record is known as **fact table/data table** in the BI world. Fact tables record **measurements or metrics** for a specific event. They are usually number heavy, thin (less no. of columns) and long (large no. of rows). They also contain foreign keys to relate to dimensional data sets.

Following are some examples of facts:

- ✔ Sale of a product.
- ✔ Purchase of an item with a specific quantity and price.
- ✔ Order received from a customer.

Dimension Table / Lookup Table:

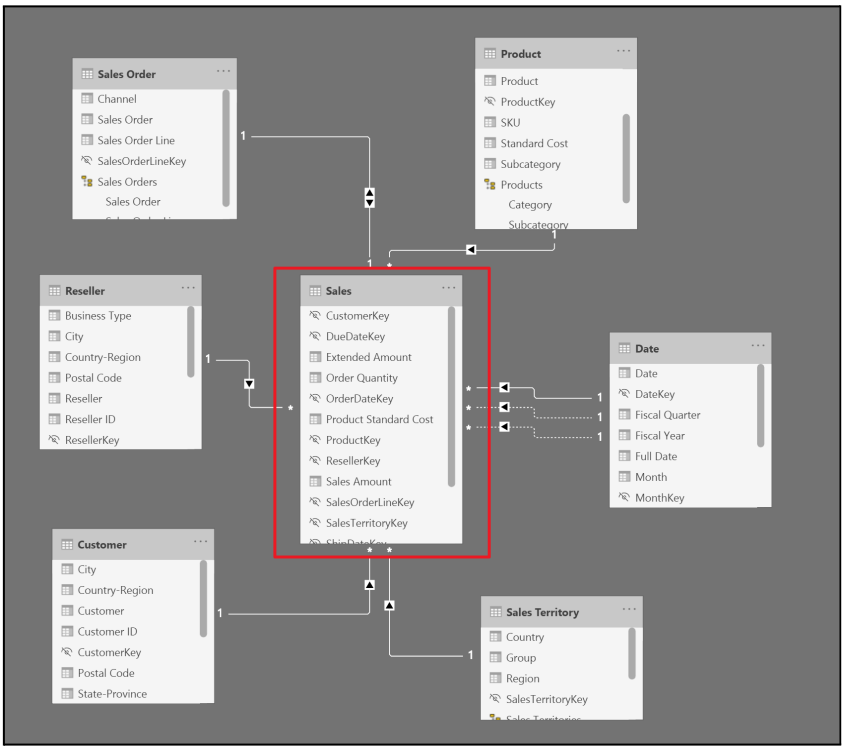
Dimensions or lookup tables have additional **information** that describe a fact. Dimension tables are text heavy. They usually have a relatively small number of records compared to fact tables, but each record may have a very large number of **attributes to describe** the fact data. Dimension tables have unique identifiers called primary keys.

Following are some examples:

- ✔ Product category, band and color.
- ✔ Vendor's origin and lead time
- ✔ Customer's name, gender, location and annual income

Fact vs Dimensions

In a Data model, a fact and dimension table are linked by **many to one relationship**. **Dimension table** exist on the **1 side** of relationship and the **fact tables** exist on **many side** of the relationship. Both of these tables have a common column that constitutes their relationship. In the dimension table, this column is the primary key, therefore, this column holds uniquely identified values. Whereas, in the fact table, this column acts a foreign key and holds duplicates. The fact table is usually present at the center, which is connected to all other tables. For the data model discussed in the last topic, Sales table (in the red box) is the fact table.



The table below provides a comparison between fact table and a dimension table.

Comparison	Fact Table	Dimension Table
Data	Contains large number of rows with consistent level of details.	Contains relatively small number of rows.
Shape	Tall and Skinny	Short & Squat
Business purpose	Stores Observations or Events	Stores Business Entities
Query purpose	Summarization	Filtering, Grouping

Table structure	Contains Dimension Keys (Foreign Keys) and Numeric Measure Columns	Contains unique identifier (Primary Key) and Attribute Columns
Examples	Sales Order, Stock Balance	Date, Products, People
Common data	Numbers	Text

Schemas in Data Model

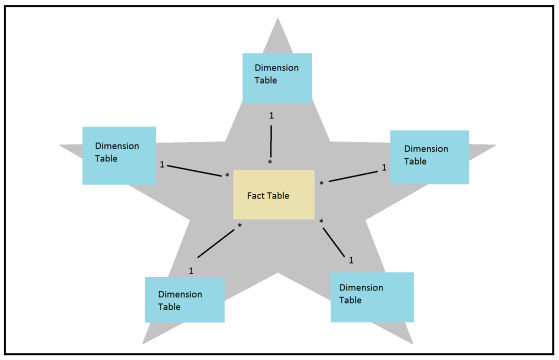
The **logical structure** of the tables that exist in a data model is known as the schema. A good schema follows the following rules:

- ✔ No looping exists between three or more tables of a data model.
- ✔ Two tables are connected via one active relationship only.

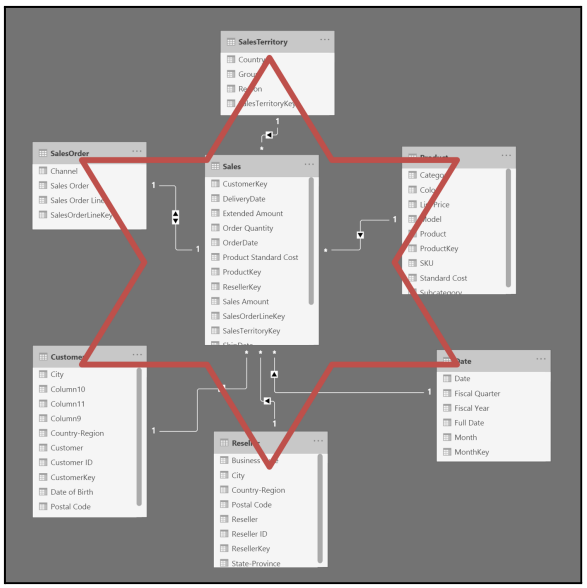
Let's consider the two types of schemas:

Star Schema

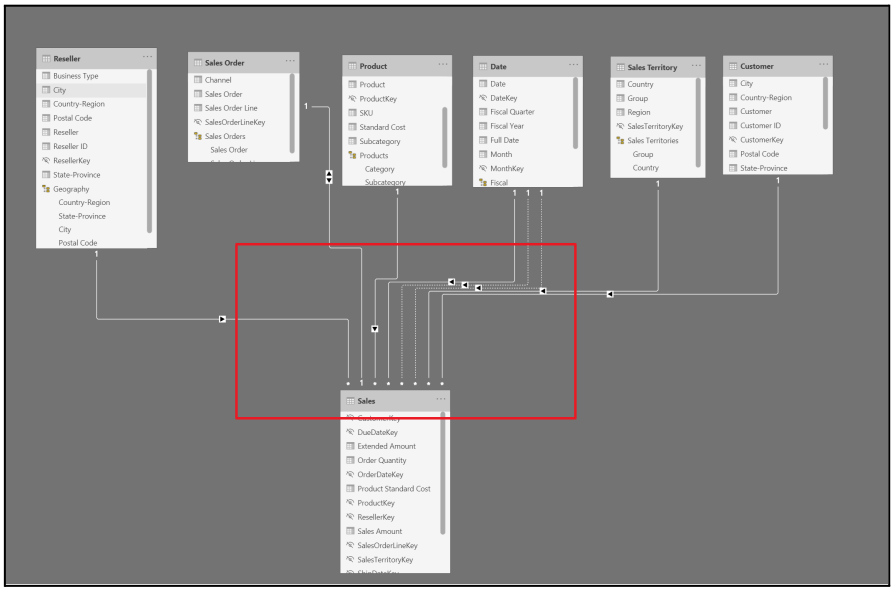
In a star schema, a **fact table is surrounded by multiple dimension tables.** Power BI engine works best with star schema.



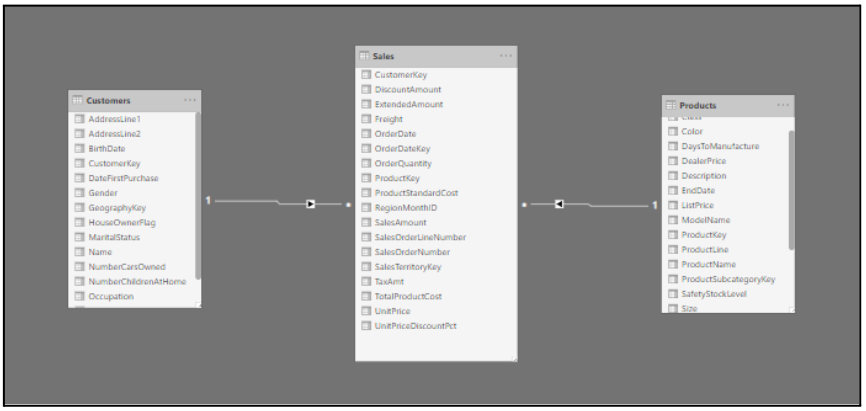
For example, the sales table exists on the many side of the relationship and all the dimension tables exist on the 1 side of the relationship as shown below.



The star schema does not necessarily have to be in the shape of a star. Below is also a star schema and is a preferred arrangement for many experts as in this arrangement is easier to visualize filter propagation from dimension to fact tables.

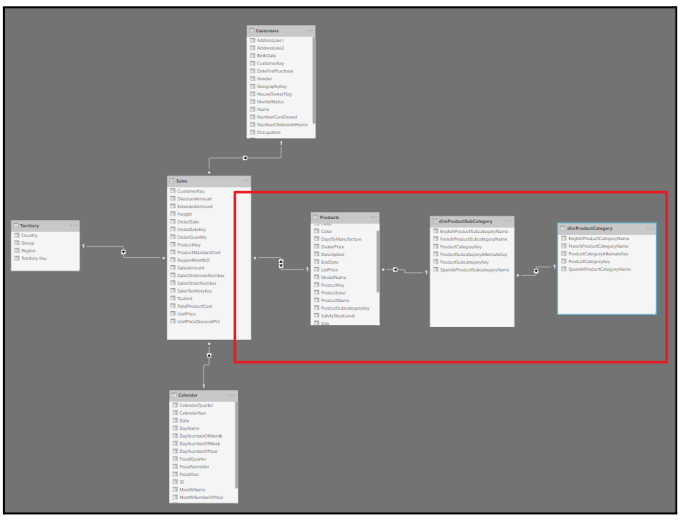


The star schema does not necessarily need 5-dimension tables to complete the star. Let's consider the example below of a star schema with two dimensional tables.



Snowflake Schema

Snowflake is a **variant of star schema** with more **things hanging off the end**. For example, the dimension product is further divided into category and subcategory which can be seen attached to the product dimension table.



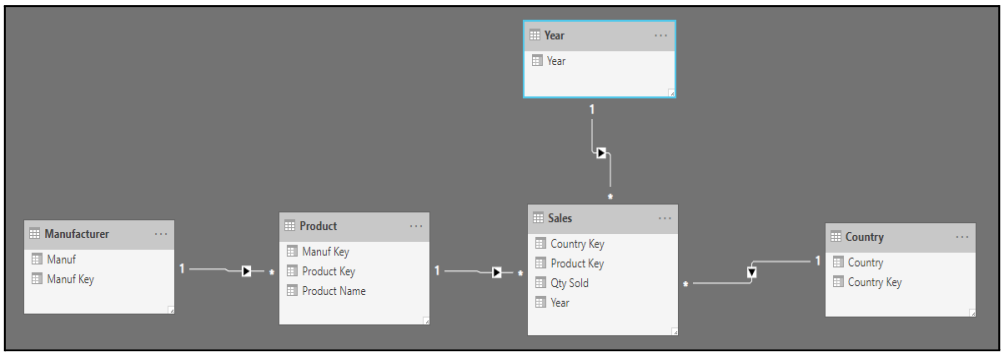
Why star Schema is Preferred

Snow flake schema structure is common in traditional transactional databases as it is the most efficient way to store the data in those systems. However, this is not the best way to structure data in Power BI. Power BI's engine performs best with the star schema. Here are few reasons why snow flake schema is not the best option for Power BI:

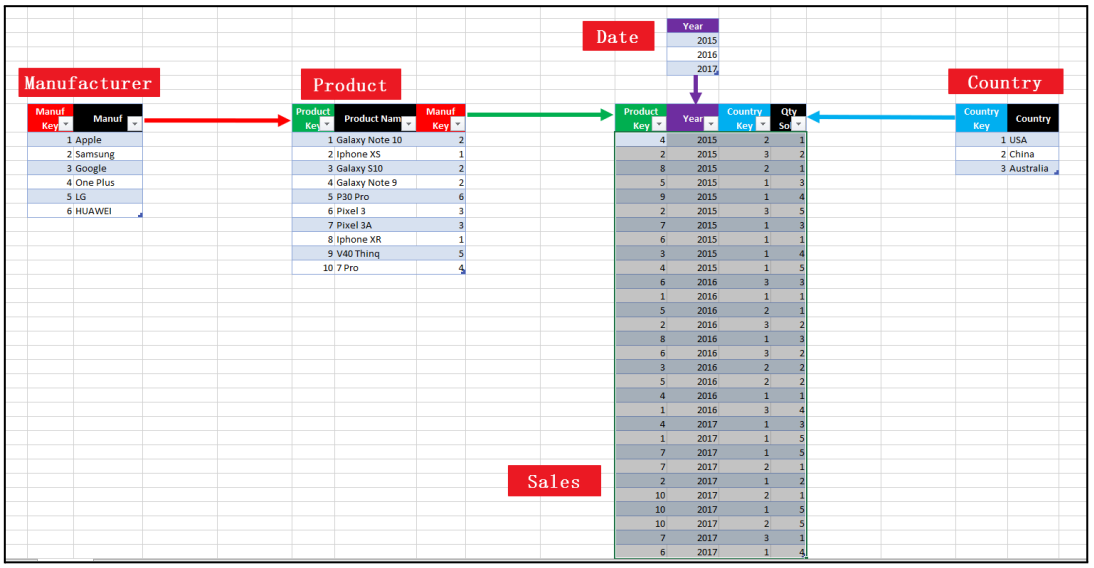
- ✔ Star schemas are easier to understand as dimensions can be used to slice and dice data and facts.
- ✔ Star schema has lesser joins and shorter paths, which means a better performance is guaranteed.
- ✔ Star schema is more scalable. New dimensions can easily be added.
- ✔ Every relationship comes at a cost. The extra relationships will potentially have negative performance impacts on the performance.
- ✔ The users can have access to the entire database.
- ✔ Power BI's vertipaq engine stores repetitive data very efficiently, particularly in the smaller lookup tables so there is no need to create hanging tables.

Filter Propagation

Let's walk along the **path** a filter takes in a data model. The Sales table in this Data set is a fact table surrounded by multiple dimensional tables.



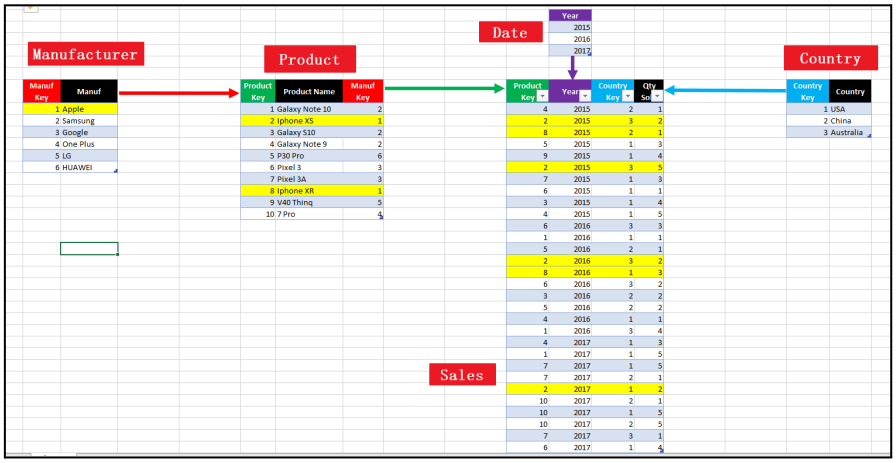
The Filters can propagate from the 1 to many side of a relationship. The screenshot below shows the direction in which the data is filtered. The filters start propagating from the dimension table and end at the fact table.



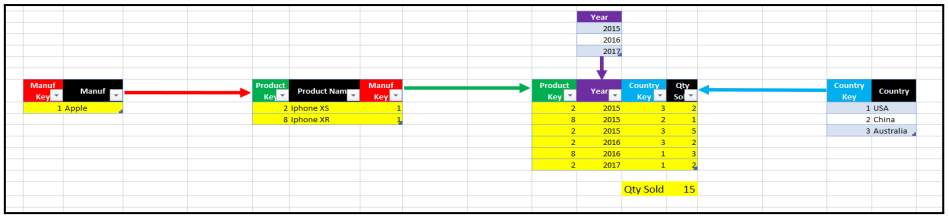
Interpreting Mechanism for Filtering

Let's put the theory of data modeling to a business use. Let's use filtering to answer different business questions.

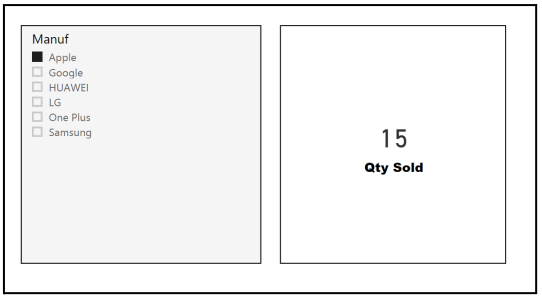
Q: What is the total Sale Qty for Apple?



To answer this question, the filters start propagation form apple in the Manufacturer table and end up by listing rows containing 1 as the primary key (Manuf Key) in the Products Table. At this point, the product key is fetched using the Manuf key. The filter then propagates from the Product table to the Sales table and lists down the rows containing 2 and 8 as the Product key. Finally, a summation DAX function is performed to find the quantity of Apple Products sold.



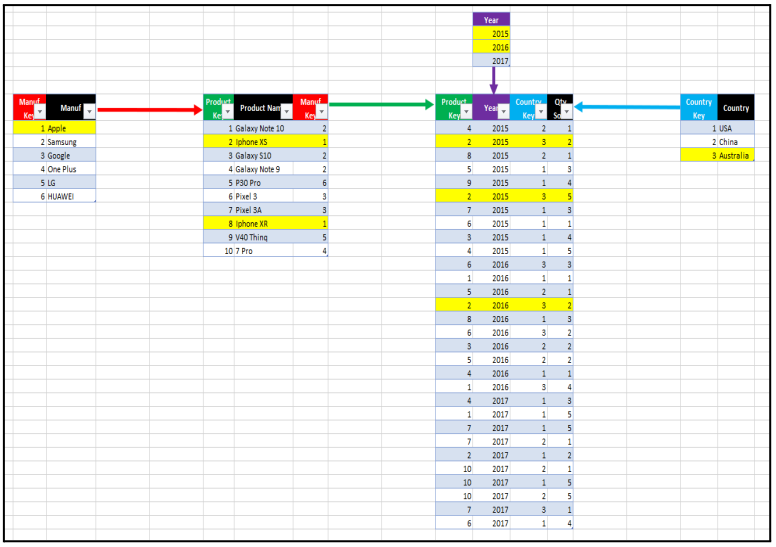
To answer this question in Microsoft Power BI, a card visual is created from the visualization pane with quantity sold in the fields. A slicer from Manuf column of Manufacturer table is created to slice the data for Apple.



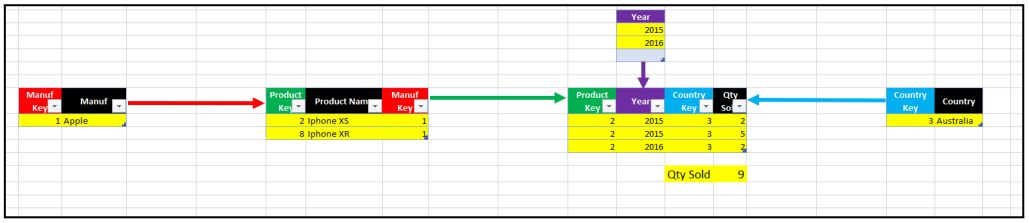
Using Multiple Filters together

Now you will use **Multiple Filters** to answer relatively complex questions.

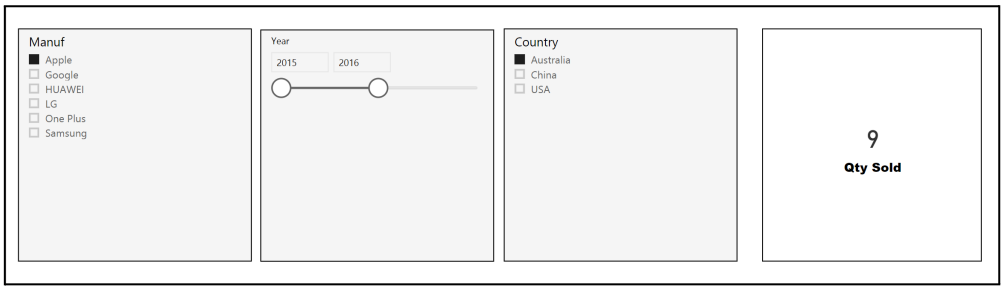
Q: What is the total Sale Qty for Apple in Australia in the Year 2015 and 2016?



Along with the filters filtering **Apple** now you have filter propagation starting from the country table, i.e., **Australia** and another filter propagation starting from the year table (**2015 and 2016**). The result only contains values from the intersection of all three filters.

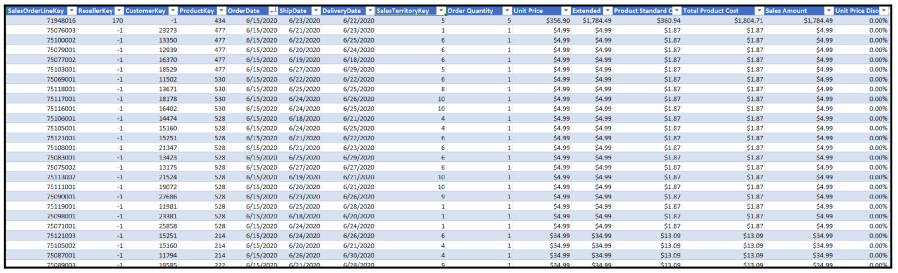


To answer this Question in Power BI, a card visual is created from the visualization pane with quantity sold in the fields. three slicers are additionally created to slice the data for Manuf, year and country.



Data Denormalization

Data denormalization refers to combining of multiple data tables into one so that the data can be queued quickly. Data denormalization introduces redundancy in the data, thereby wasting the storage. Let's consider a Sales table shown in the picture. This table has many columns and rows.



SalesOrderLineKey	ResellerKey	CustomerKey	ProductKey	OrderDate	ShipDate	DeliveryDate	SalesTerritoryKey	Order Quantity	Unit Price	Extended	Product Standard C	Total Product Cost	Sales Amount	Unit Price Disc
71948016	170	-1	434	6/15/2020	6/23/2020	6/22/2020	5	5	$356.90	$1,784.49	$360.54	$1,804.71	$1,784.49	0.00%
75076003	-1	23273	477	6/15/2020	6/21/2020	6/23/2020	1	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75100002	-1	13390	477	6/15/2020	6/22/2020	6/25/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75079001	-1	12939	477	6/15/2020	6/20/2020	6/24/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75077002	-1	16370	477	6/15/2020	6/19/2020	6/18/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75103001	-1	18529	477	6/15/2020	6/27/2020	6/29/2020	5	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75069001	-1	11502	530	6/15/2020	6/22/2020	6/22/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75118001	-1	13671	530	6/15/2020	6/25/2020	6/25/2020	8	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75117001	-1	18178	530	6/15/2020	6/24/2020	6/26/2020	10	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75116001	-1	16402	530	6/15/2020	6/24/2020	6/25/2020	10	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75106001	-1	14474	528	6/15/2020	6/18/2020	6/21/2020	4	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75105001	-1	15160	528	6/15/2020	6/24/2020	6/25/2020	4	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75121001	-1	15251	528	6/15/2020	6/21/2020	6/22/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75108001	-1	21347	528	6/15/2020	6/21/2020	6/23/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75083001	-1	13423	528	6/15/2020	6/25/2020	6/29/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75075002	-1	13375	528	6/15/2020	6/27/2020	6/27/2020	6	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75113002	-1	21526	528	6/15/2020	6/19/2020	6/21/2020	10	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75111001	-1	19072	528	6/15/2020	6/20/2020	6/21/2020	10	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75090001	-1	21686	528	6/15/2020	6/21/2020	6/26/2020	9	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75119001	-1	11981	528	6/15/2020	6/23/2020	6/28/2020	1	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75098001	-1	23381	528	6/15/2020	6/18/2020	6/20/2020	1	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75071001	-1	25858	528	6/15/2020	6/24/2020	6/24/2020	1	1	$4.99	$4.99	$1.87	$1.87	$4.99	0.00%
75121003	-1	15251	214	6/15/2020	6/24/2020	6/26/2020	6	1	$34.99	$34.99	$13.09	$13.09	$34.99	0.00%
75105002	-1	15160	214	6/15/2020	6/20/2020	6/21/2020	4	1	$34.99	$34.99	$13.09	$13.09	$34.99	0.00%
75087001	-1	11794	214	6/15/2020	6/26/2020	6/30/2020	4	1	$34.99	$34.99	$13.09	$13.09	$34.99	0.00%
75088001	-1	18585	222	6/15/2020	6/21/2020	6/28/2020	5	1	$34.99	$34.99	$13.09	$13.09	$34.99	0.00%

For some business analytics, you created a pivot table to show the total sales against each product key.

Row Labels	Sum of Sales Amount
212	11385.34
213	29031.62
214	117355.95
215	12098.28
216	31867.07
217	116904.72
218	6060.43
219	513
220	13331.69
221	33795.53
222	118279.77
223	5081.19
224	11699.19
225	34449.72

Now let's suppose you want to get some more information related to the product from the products table such as Product name, model, category and subcategory.

Row Labels	Sum of Sales Amount	Product	Model	Subcategor	Category
212	11385.34	Sport-100 Helmet, Red	Sport-100	Helmets	Accessories
213	29031.62	Sport-100 Helmet, Red	Sport-100	Helmets	Accessories
214	117355.95	Sport-100 Helmet, Red	Sport-100	Helmets	Accessories
215	12098.28	Sport-100 Helmet, Black	Sport-100	Helmets	Accessories
216	31867.07	Sport-100 Helmet, Black	Sport-100	Helmets	Accessories
217	116904.72	Sport-100 Helmet, Black	Sport-100	Helmets	Accessories
218	6060.43	Mountain Bike Socks, M	Mountain Bike Socks	Socks	Clothing
219	513	Mountain Bike Socks, L	Mountain Bike Socks	Socks	Clothing
220	13331.69	Sport-100 Helmet, Blue	Sport-100	Helmets	Accessories
221	33795.53	Sport-100 Helmet, Blue	Sport-100	Helmets	Accessories
222	118279.77	Sport-100 Helmet, Blue	Sport-100	Helmets	Accessories
223	5081.19	AWC Logo Cap	Cycling Cap	Caps	Clothing
224	11699.19	AWC Logo Cap	Cycling Cap	Caps	Clothing
225	34449.72	AWC Logo Cap	Cycling Cap	Caps	Clothing
228	21445.71	Long-Sleeve Logo Jersey, S	Long-Sleeve Logo Jersey	Jerseys	Clothing
229	12603.08	Long-Sleeve Logo Jersey, M	Long-Sleeve Logo Jersey	Jerseys	Clothing
230	35500.46	Long-Sleeve Logo Jersey, M	Long-Sleeve Logo Jersey	Jerseys	Clothing
231	67145.08	Long-Sleeve Logo Jersey, M	Long-Sleeve Logo Jersey	Jerseys	Clothing
232	28399.13	Long-Sleeve Logo Jersey, L	Long-Sleeve Logo Jersey	Jerseys	Clothing
233	66165.47	Long-Sleeve Logo Jersey, L	Long-Sleeve Logo Jersey	Jerseys	Clothing

Using VLOOKUP(), the required columns can be fetched. However, this adds duplicating rows. This may not seem as a problem with fewer rows but when the data contains many rows, it could be a problem as the data is stored in tables. Power BI avoids this using a columnar database engine, i.e., vertipaq.

Relationship Characteristics

Relationships provide the **connection** or **link** between two tables. Relationships are defined in Power BI based on the following rules:

- ✔ They take place between 1 column to another.
- ✔ No multi-column relationships exist.
- ✔ No self-referencing relationships exist.
- ✔ Tables cannot have 2 active relationships. Only 1 active path can exist between two tables.

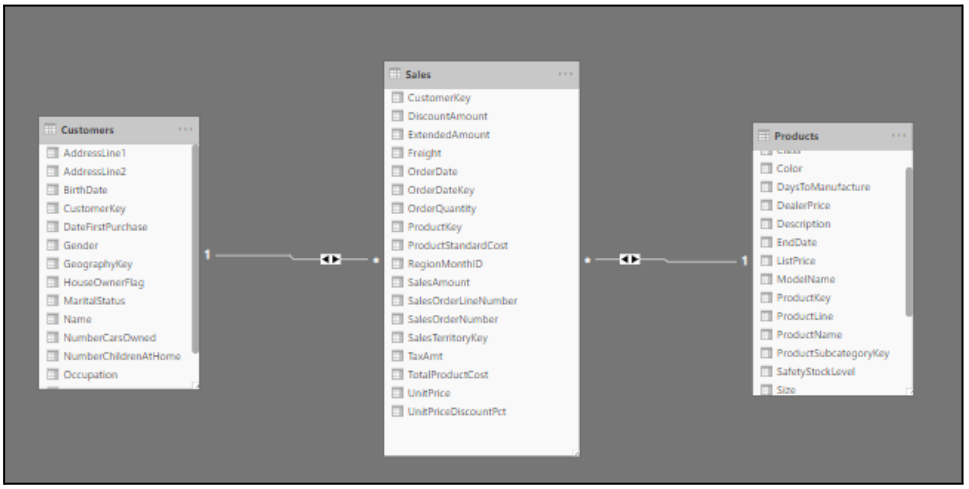
Each relationship has a few **properties** which define the relationship.

Cardinality

Cardinality determines if it's a fact or dimension table. Cardinality of a relationship can be one-to-one, one-to-many or many-to-many.

Filter Direction

Filter direction can take on two values either **singe or both**. Filter Direction if set to single value allows the flow of filter in one direction. The filter flows from the dimension to the fact table. It cannot filter across two dimensions and therefore, provides optimal performance. On the other hand, setting the filter direction to both makes it as a one big table and also allows to filter data between two dimension tables. This can however have performance issues if data model is huge and it also can create loops if multiple fact tables exist.



If the filtering is turned on to both directions, Power BI treats both sides of the filter as a single query. For example, the above data set consists of three tables however it is considered as a single query.

Apply Security Filter in Both Directions

This option is only relevant if you have **row level security** in your model. Row Level Security filtering uses single direction filtering regardless of whether relationships are set to single or both. You can manually enable bi-directional cross-filter with row-level security by selecting the relationship and checking the **Apply security filter in both directions** checkbox.

Active

Only one active relationship can exist between two tables. It is necessary to ensure that filtering is done correctly.

Assume Referential Integrity

This option is only enabled when connecting to data using the Direct Query method. For referential integrity to work smoothly, the date column must not contain null or blank values and for each “from” column there must be a “to” column. In this context, the form column is the Many in a One-to-Many relationship.

Chapter 3: Calculated Columns and Measures

What will you learn in this chapter?

- ✔ Calculated Column and its use case
- ✔ When do Calculated Columns fail
- ✔ Calculated Measure and its use case
- ✔ When to use Calculated Column or Calculated Measure

Getting Started

To get started, you will have to load the data set into Power BI desktop. For this course, we will be using a modified version of Adventure Works 2020 data set. The original Adventure Works 2020 dataset developed by Microsoft for DAX learning is available at: https://docs.microsoft.com/en-us/power-bi/guidance/dax-sample-model

This data set has been slightly modified to meet the learning objectives of this course. To get started:

1. Open a new **Power BI desktop file**.
2. Click on **Home** tab in the **Ribbon** and
3. Click on **Excel** in **Data section**.
4. Select "Adventure Works – Data Revised" file.
5. Click on Open.
6. Select all the sheets.
7. Click on **Load**.

A few errors have been intentionally introduced for learning.

8. Click on **Model** Tab in the **Left Ribbon**.
9. Click on OrderDate in the Sales table and drag the cursor to Date in the Date Table.

This step is required because Power BI does not automatically detect relationship between Sales and Date table due to multiple date fields in the Sales Table. Here's a brief summary of the dataset.

Table	Description
Customer	Describes customers and their geographic location. Customers purchase products online (Internet sales).
Date	There are three relationships between the **Date** and **Sales** tables, for order date, ship date, and due date. The order date relationship is active. The company's reports sales using a fiscal year that commences on July 1 of each year. The table is marked as a date table using the **Date** column.
Product	Stores finished products only.
Reseller	Describes resellers and their geographic location. Reseller on sell products to their customers.
Sales	Stores rows at sales order line grain. All financial values are in US dollars (USD). The earliest order date is July 1, 2017, and the latest order date is June 15, 2020.
Sales Order	Describes sales order and order line numbers, and also the sales channel, which is either **Reseller** or **Internet**. This table has a one-to-one relationship with the **Sales** table.
Sales Territory	Sales territories are organized into groups (North America, Europe, and Pacific), countries, and regions. Only the United States sells products at the region level.

Source: Microsoft | Docs – DAX Sample Model Structure

Please note that the Adventure Works data set has been slightly modified and the Sales table may not contain all the columns listed above.

Calculated Column and its use case

Calculated columns and measures are created to add additional information in a data set. They add value to the dataset by bridging the gap between data model and good data visualization.

What is a Calculated Column:

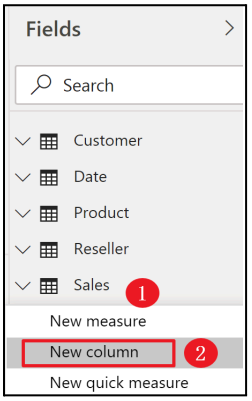
Calculated Columns are a method for adding additional columns to the tables. Calculated Column is evaluated for each row of the table, creating a row context for each row. A naked column can be used while creating calculated columns. A Calculated Column is typically used in the axis, legend, filter or group area of a visualization.

Calculated Column is more intuitive for excel/business users. However, calculated column takes more disk space as its stored in data set. Therefore, creating a calculated column must be avoided if its use is un-certain.

Continue using the previous file for the exercise or open the file Creating_Calculated_Columns_and_Measures_1.pbix..

To create a calculated column:

1. Right click on the **Sales** table in the **Fields** Pan.
2. Click on **New Column**.



Column tools menu opens, containing the formula bar to type the DAX code.

3. Type the following formula in the formula bar and press Enter.

```
Profit (CC) =
Sales[Sales Amount] - Sales[Total Product Cost]
```

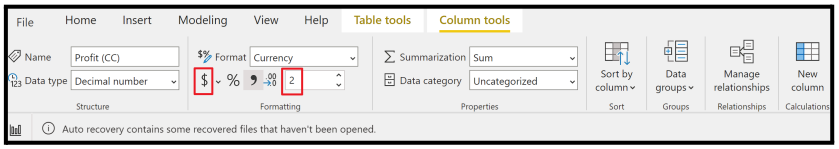
In this Expression:

- *Profit is calculated by subtracting the Total Product Cost from Sales Amount for each transaction.*

4. Click on Profit (CC) in the Sales Table.

   Column tools menu opens,

5. Click on the "$" and type 2 in the box.



6. Click on Sales column in the Sales Table.

   Column tools menu opens,

7. Click on the "$" and type 2 in the box.
8. Click on the **matrix** visual in the **visualization pane**.
9. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
10. Drag and drop **Sales Amount** and **Profit (CC)** from Sales to the Values.
11. Expand Accessories category and you will have the following visual

Category	Sales Amount	Profit (CC)
Accessories	**$1,272,059.71**	**$634,391.75**
Bike Racks	$237,096.21	$95,006.13
Bike Stands	$39,591.00	$24,782.97
Bottles and Cages	$64,274.95	$38,214.28
Cleaners	$18,407.03	$8,541.62
Fenders	$46,619.58	$29,184.96
Helmets	$484,049.97	$226,383.23
Hydration Packs	$105,826.62	$49,040.07
Locks	$16,225.22	$5,025.51
Pumps	$13,514.71	$4,196.75
Tires and Tubes	$246,454.42	$154,016.23
Bikes	**$94,620,526.59**	**$10,515,090.51**
Clothing	**$2,117,613.52**	**$368,846.60**
Components	**$11,799,074.18**	**$1,032,957.07**
Total	**$109,809,274.00**	**$12,551,285.93**

All the DAX code snippets where calculated columns have been created are colour coded blue for this manual.

When do Calculated Columns fail

Let's create a few more calculated columns to understand when do calculated columns fail.

Continue using the previous file for the exercise or open the file Creating_Calculated_Columns_and_Measures_2.pbix..

To create a calculated column:

1. Right click on the **Sales** table in the **Fields** Pan.
2. Click on **New Column**.

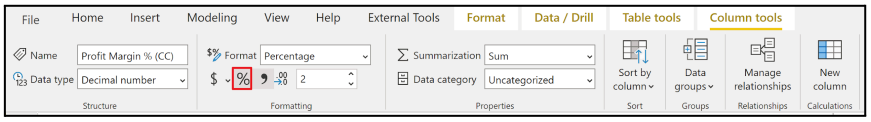
   Column tools open.
3. Type the following formula in the formula bar and press Enter

```
Profit Margin % (CC) =
Sales[Profit (CC)] / Sales[Sales Amount]
```

In this Expression:

- *The Profit Margin % is found by dividing the profit by Sales Amount for each transaction.*

4. Click on % in the **formatting section**.

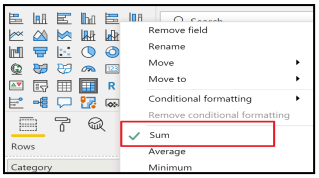


5. Click on the **matrix** visual in the **visualization pane**.
6. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
7. Drag and drop **Sales Amount**, **Profit (CC)** and **Profit Margin % (CC)** from Sales table to the Values.

Category	Sales Amount	Profit (CC)	Profit Margin % (CC)
Accessories	**$1,272,059.71**	**$634,391.75**	**24,348.03**
Bike Racks	$237,096.21	$95,006.13	377.78
Bike Stands	$39,591.00	$24,782.97	155.87
Bottles and Cages	$64,274.95	$38,214.28	5,156.53
Cleaners	$18,407.03	$8,541.62	723.68
Fenders	$46,619.58	$29,184.96	1,327.80
Helmets	$484,049.97	$226,383.23	4,953.62
Hydration Packs	$105,826.62	$49,040.07	585.20
Locks	$16,225.22	$5,025.51	80.66
Pumps	$13,514.71	$4,196.75	83.24
Tires and Tubes	$246,454.42	$154,016.23	10,903.66
Bikes	**$94,620,526.59**	**$10,515,090.51**	**4,635.61**
Clothing	**$2,117,613.52**	**$368,846.60**	**5,027.39**
Components	**$11,799,074.18**	**$1,032,957.07**	**2,723.41**
Total	**$109,809,274.00**	**$12,551,285.93**	**36,734.44**

Take a minute to analyze the numbers in the Profit Margin (CC) column.

8. Right click on the Profit Margin % (CC) in the values field and check the aggregation.



Notice that none of the aggregation available suits the requirement of percentage calculation because $\sum(\frac{Profit}{Sales\ Amount}) \neq \frac{\sum Profit}{\sum Sales\ Amount}$.

Calculated Measure and its use case

Calculated measures are memory-friendly alternatives for calculated columns.

What is a Calculated Measure:

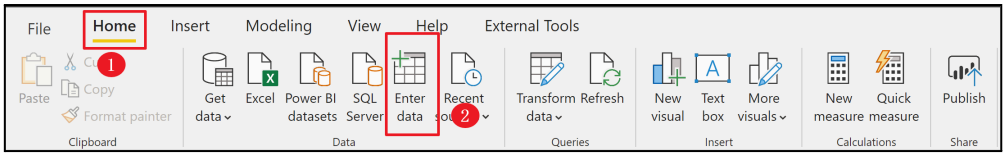
Calculated measures are used to perform aggregations. They do not add a value for every row in the table. This makes measures memory efficient as they do not appear in your data set. They are lightweight alternatives to **calculated columns**. You cannot however use naked columns while writing regular DAX expressions for creating a measure.

Measure is used in the values area of a visualisation. Therefore, they are only created when a measure is used in a visual. Following the best practise, we will be creating a separate table for measures.

Continue using the previous file for the exercise or open the file Creating_Calculated_Columns_and_Measures_3.pbix..

To create a table for measures:

1. Click on **Home** tab in the **Ribbon**.
2. Click on **Enter data** in the **Data** section.



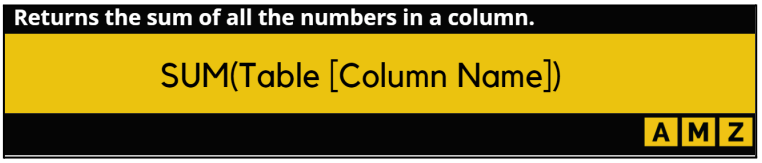
A pop-up menu appears.

3. Name the new table to **All Measures**.

4. Click load.

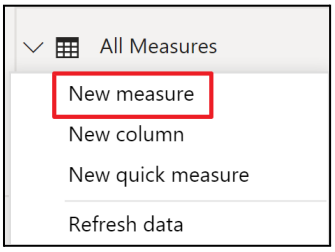
Now that a table for measures is created. Let's start creating measures.

Before we create our first measure, lets quickly get affiliated with the simplest function in DAX.



To create measure:

5. Right click on **All Measures** table in the **Fields** Pan.
6. Click on **New measure**.



7. Type the follow DAX code in the formula bar in the measure tools and press Enter.

```
Profit (CM) =
SUM ( Sales[Sales Amount] ) - SUM ( Sales[Total Product Cost] )
```

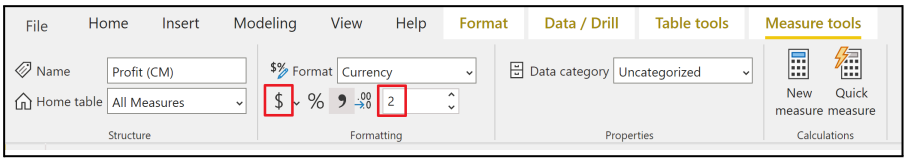
In this Expression:

- *The aggregated value of Total Product Cost for the entire column is subtracted from the aggregated value of Sales amount for the entire column.*

8. Click on the Profit (CM) in the All Measures table.

   Measure tools menu opens,

9. Click on the "$" and type 2 in the box.



10. Create another measure by following the steps 5-6.
11. Type the following DAX code in the formula bar in the measure tools and press Enter.

```
Profit Margin % (CM) =
[Profit (CM)] / SUM ( Sales[Sales Amount] )
```

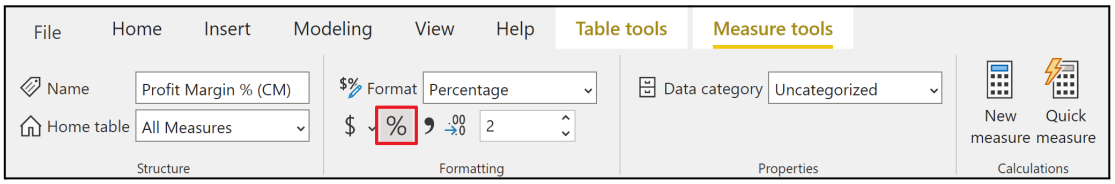
In this Expression:

- *The profit margin % is found by dividing the profit calculated in the previous measure by the total sales amount.*

12. Click on Profit Margin % (CM) in All Measures.

Measure tools open

13. Click on the % sign in the formatting section.



14. Click on the **matrix** visual in the **visualization pane**.
15. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
16. Drag and drop **Sales Amount** from Sales table to the Values.
17. Drag and drop **Profit (CM)** and **Profit Margin % (CM)** from All Measures to the Values.

Category	Sales Amount	Profit (CM)	Profit Margin % (CM)
Accessories	**$1,272,059.71**	**$634,391.75**	**49.87 %**
Bike Racks	$237,096.21	$95,006.13	40.07 %
Bike Stands	$39,591.00	$24,782.97	62.60 %
Bottles and Cages	$64,274.95	$38,214.28	59.45 %
Cleaners	$18,407.03	$8,541.62	46.40 %
Fenders	$46,619.58	$29,184.96	62.60 %
Helmets	$484,049.97	$226,383.23	46.77 %
Hydration Packs	$105,826.62	$49,040.07	46.34 %
Locks	$16,225.22	$5,025.51	30.97 %
Pumps	$13,514.71	$4,196.75	31.05 %
Tires and Tubes	$246,454.42	$154,016.23	62.49 %
Bikes	**$94,620,526.59**	**$10,515,090.51**	**11.11 %**
Clothing	**$2,117,613.52**	**$368,846.60**	**17.42 %**
Components	**$11,799,074.18**	**$1,032,957.07**	**8.75 %**
Total	**$109,809,274.00**	**$12,551,285.93**	**11.43 %**

Notice the change in percentage values.

All the DAX code snippets where calculated measures have been created are color coded green for this manual.

When to use Calculated Column or Calculated Measure

Now that we have a situation where calculated columns fail, lets consider a situation when calculated columns are the best solution.

Continue using the previous file for the exercise or open the file Creating_Calculated_Columns_and_Measures_4.pbix..

To create a calculated column:

1. Right click on the **Sales** table.
2. Click on **New Column**.

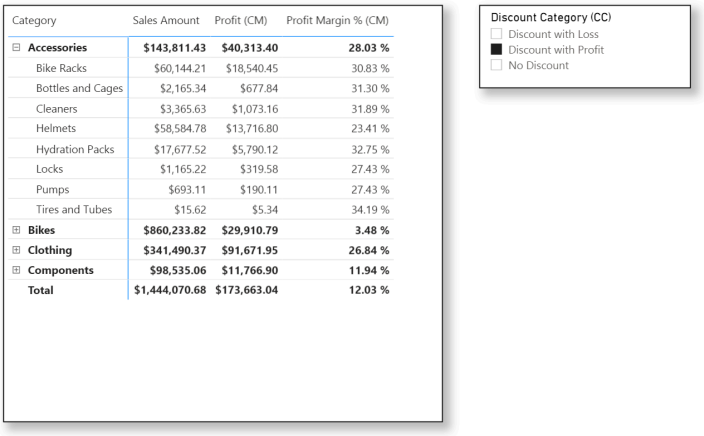
   Column tools open.
3. Type the following code in the formula bar.

```
Discount Category (CC) =
IF (
    Sales[Unit Price Discount Pct] = 0,
    "No Discount",
    IF ( Sales[Profit (CC)] > 0, "Discount with Profit", "Discount with Loss" )
)
```

In this Expression:

- *In the first part of the IF() function, a logical test for the Unit Price Discount Pct is performed.*
- *If the test returns TRUE this means that the discount is 0 so the cell in the new column is assigned a string value of "No Discount".*
- *When the IF() returns FALSE, then another logical test on the Profit column is performed and Discount is categorized based on the profit.*

4. Click on the **matrix** visual in the **visualization pane**.
5. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
6. Drag and drop **Sales Amount** from Sales table to the Values.
7. Drag and drop **Profit (CM)** and **Profit Margin % (CM)** from All Measures to the Values.
8. Click on the **slicer** in the visualization pane.
9. Drag and drop Discount Category (CC) from the Sales table to the Field area.
10. Click on Discount with Profit.



Category	Sales Amount	Profit (CM)	Profit Margin % (CM)
Accessories	**$143,811.43**	**$40,313.40**	**28.03 %**
Bike Racks	$60,144.21	$18,540.45	30.83 %
Bottles and Cages	$2,165.34	$677.84	31.30 %
Cleaners	$3,365.63	$1,073.16	31.89 %
Helmets	$58,584.78	$13,716.80	23.41 %
Hydration Packs	$17,677.52	$5,790.12	32.75 %
Locks	$1,165.22	$319.58	27.43 %
Pumps	$693.11	$190.11	27.43 %
Tires and Tubes	$15.62	$5.34	34.19 %
Bikes	**$860,233.82**	**$29,910.79**	**3.48 %**
Clothing	**$341,490.37**	**$91,671.95**	**26.84 %**
Components	**$98,535.06**	**$11,766.90**	**11.94 %**
Total	**$1,444,070.68**	**$173,663.04**	**12.03 %**

Notice that the slicer values have been used to slice the data.

Now that you're familiar with the calculated columns and calculated measures, let's look at situations when they should be preferred over the other.

Calculated Column vs. Calculated Measures

Both of them have their own use cases and the choice depends on the business requirement. There are few points to consider when you're choosing between a calculated column and calculated measure.

	Calculated Column	Calculated Measure
Slice or filter values	✔	
Calculate percentages		✔
Calculate Ratios		✔
Categorize texts/numbers	✔	
Complex aggregations		✔
Expressions that are strictly bound to current row	✔	
Consume Memory	✔	
Consume CPU		✔

Calculated on the fly		✔
Calculated at refresh time	✔	
Data can be seen in the Data tab	✔	

Chapter 4: Basic DAX Functions

What will you learn in this chapter?

- ✔ Two broad categories of functions in DAX
- ✔ Use of DIVIDE() and COUNT()
- ✔ Counting Rows
- ✔ Use of DISTINCTCOUNT ()
- ✔ Finding Min, Max and Average
- ✔ Counting Blank Values

Two broad categories of functions in DAX

DAX is used to bring about some meaningful information hidden inside raw data. This is done using functions in DAX. Functions in DAX perform data manipulation.

There are two broad categories of functions in DAX.

Regular Functions

Regular functions simply calculate the values based on the filter selection. They are used to perform calculations for a column.

Examples of Regular Functions in DAX are:

- ✔ SUM()
- ✔ COUNT()
- ✔ AVERAGE()
- ✔ MIN()
- ✔ MAX()

Iterator Functions

Iterator functions are used to perform calculations for expressions. The X at the end of the functions signifies their use.

The iterator functions go through every single record of data and calculate the result of expression for that record. It then stores the results in a temporary memory. After the function has parsed the complete table, this temporary memory is released and the aggregated results are shown.

Examples of Iterator Functions in DAX are:

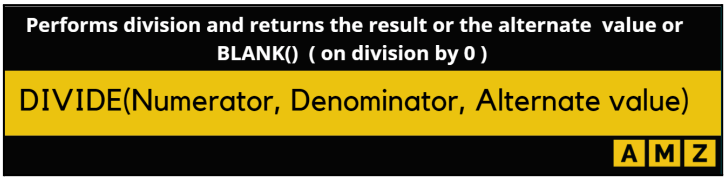
- ✔ SUMX()
- ✔ COUNTX()
- ✔ AVERAGEX()
- ✔ MINX()
- ✔ MAXX()

Use of DIVIDE() and COUNT()

Now that you're familiar with the SUM() function, lets introduce you to some more basic functions in DAX.

Continue using the previous file for the exercise or open the file Basic_DAX_Functions_1.pbix..

DIVIDE():



To use DIVIDE() function:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Total Cost (CM) =
SUM ( Sales[Total Product Cost] )
```

In this Expression:

- *Sum of all values in the Total Product Cost column from the Sales table is calculated.*

4. Right click on **All Measures** in the **Fields**.
5. Click on **New measure.**

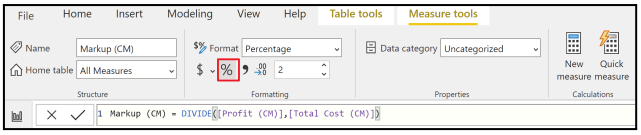
   Measure tools open.
6. Type the following formula in the formula bar and press Enter.

```
Markup (CM) =
DIVIDE ( [Profit (CM)], [Total Cost (CM)], 0 )
```

In this Expression:

- *The profit is divided by the Total cost.*
- *If the division results in a mathematical error, 0 is returned.*

7. Click on % in the formatting section.

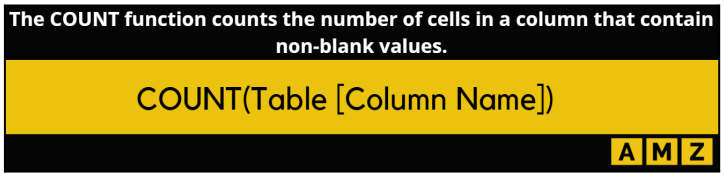


8. Click on the **matrix** visual in the **visualization pane**.
9. Drag and drop **Category** and **Subcategory** from the Product table to the rows.
10. Drag and drop **Sales Amount** from Sales to the Values.
11. Drag and drop **Profit (CM)** and **Markup (CM)** from All Measures to the Values.

Category	Sales Amount	Profit (CM)	Markup (CM)
Accessories	**$1,272,059.71**	**$634,391.75**	**99.49 %**
Bike Racks	$237,096.21	$95,006.13	66.86 %
Bike Stands	$39,591.00	$24,782.97	167.36 %
Bottles and Cages	$64,274.95	$38,214.28	146.64 %
Cleaners	$18,407.03	$8,541.62	86.58 %
Fenders	$46,619.58	$29,184.96	167.40 %
Helmets	$484,049.97	$226,383.23	87.86 %
Hydration Packs	$105,826.62	$49,040.07	86.36 %
Locks	$16,225.22	$5,025.51	44.87 %
Pumps	$13,514.71	$4,196.75	45.04 %
Tires and Tubes	$246,454.42	$154,016.23	166.62 %
Bikes	**$94,620,526.59**	**$10,515,090.51**	**12.50 %**
Clothing	**$2,117,613.52**	**$368,846.60**	**21.09 %**
Components	**$11,799,074.18**	**$1,032,957.07**	**9.59 %**
Total	**$109,809,274.00**	**$12,551,285.93**	**12.91 %**

Notice the different markup values for different categories.

COUNT():



To use COUNT() function:

12. Right click on **All Measures** in the **Fields**.
13. Click on **New measure.**

    Measure tools open.
14. Type the following formula in the formula bar and press Enter.

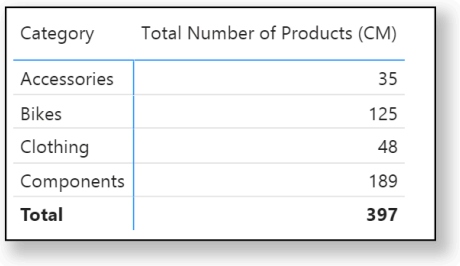
```
Total Number of Products (CM) =
COUNT ( 'Product'[ProductKey] )
```

In this Expression:

- *The non-blank values in the Product Key column from in Product table are counted.*

15. Click on the **matrix** visual in the **visualization pane**.
16. Drag and drop **Category** from the Products table to the rows.

17. Drag and drop **Total Number of Products (CM)** from All Measures tableto the Values.



Category	Total Number of Products (CM)
Accessories	35
Bikes	125
Clothing	48
Components	189
Total	**397**

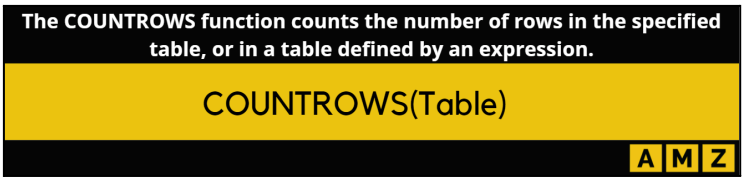
Notice that we have a total of 397 products with the highest number of products in the Components category.

Counting Rows

In this exercise, you will learn how to count rows in any table using DAX.

Continue using the previous file for the exercise or open the file Basic_DAX_Functions_2.pbix..

COUNTROWS():



To use COUNTROWS() function:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Number of Sales (CM) =
COUNTROWS ( Sales )
```

In this Expression:

- *The number of rows in the Sales Table are counted. This also corresponds to the total number of transactions.*

4. Click on the **matrix** visual in the **visualization pane**.
5. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
6. Drag and drop **Total Number of Products (CM), Number of Sales (CM)** from All Measures table to the Values.

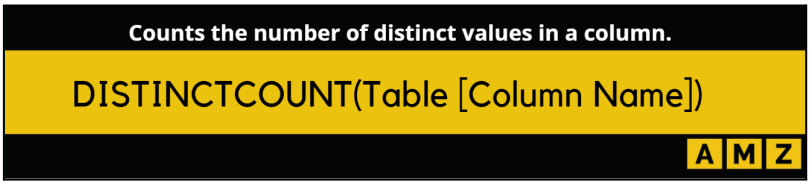
Category	Total Number of Products (CM)	Number of Sales (CM)
⊞ **Accessories**	**35**	**41193**
⊟ **Bikes**	**125**	**40005**
Mountain Bikes	38	12457
Road Bikes	65	20918
Touring Bikes	22	6630
⊞ **Clothing**	**48**	**21368**
⊞ **Components**	**189**	**18687**
Total	**397**	**121253**

Notice that although the category of Accessories has 35 products only, the greatest number of sales are recorded for this category.

Use of DISTINCTCOUNT ()

In this exercise, you will learn how to find the distinct products sold using DAX functions.

Continue using the previous file for the exercise or open the file Basic_DAX_Functions_3.pbix..



To use DISTINCTCOUNT() function:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Total Number of Products Sold (CM) =
DISTINCTCOUNT ( Sales[ProductKey] )
```

In this Expression:

- *The number of unique product keys in the Sales table are counted.*

4. Click on the **card** visual in the **Visualization Pane.**
5. Drag and drop Total number of Products Sold (CM) from All Measures to the Fields.

Notice that 350 out of 397 distinct products have been sold.

6. Click on the **matrix** visual in the **visualization pane**.
7. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
8. Drag and drop **Total Number of Products (CM), Number of Sales (CM) and Total Number of Products Sold (CM)** from All Measures table to the Values.

Category	Total Number of Products (CM)	Total Number of Products Sold (CM)	Number of Sales (CM)
⊞ Accessories	35	30	41193
⊞ Bikes	125	125	40005
⊞ Clothing	48	45	21368
⊞ Components	189	150	18687
Total	397	350	121253

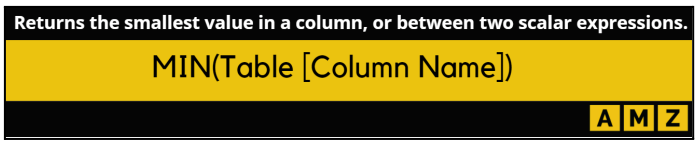
Notice the breakdown of values due to the incoming filter context. Notice that all the products from the category of bikes have been sold. Whereas, only 150 products out of 189 total products in the category of components have been sold.

Finding Min, Max and Average

Now let's find the minimum, maximum and average unit price per category.

Continue using the previous file for the exercise or open the file Basic_DAX_Functions_4.pbix..

MIN()



To use MIN() function:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

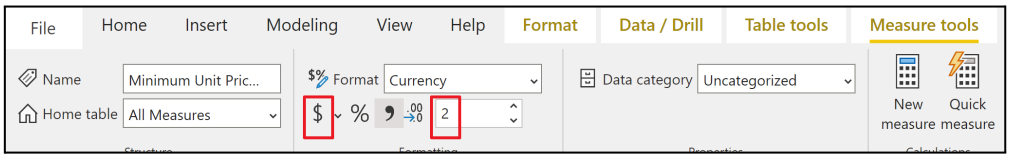
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Minimum Unit Price (CM) =
MIN ( Sales[Unit Price] )
```

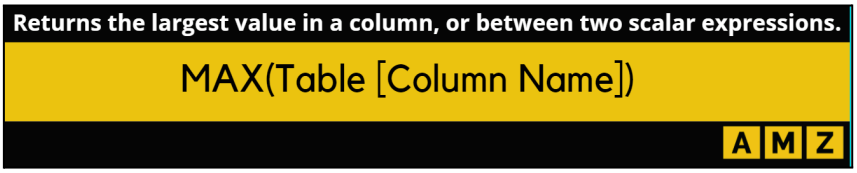
In this Expression:

- *The minimum value of unit price in the Unit Price column from the Sales Table is calculated.*

4. Click on "$" symbol and type "2" in the text box.



MAX():



To use MAX() function:

5. Right click on **All Measures** in the **Fields**.
6. Click on **New measure.**

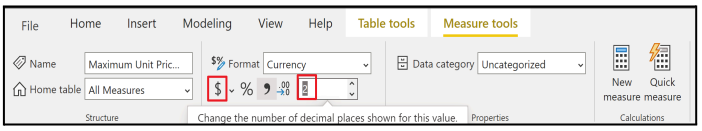
   Measure tools open.
7. Type the following formula in the formula bar and press Enter.

```
Maximum Unit Price (CM) =
MAX ( Sales[Unit Price] )
```

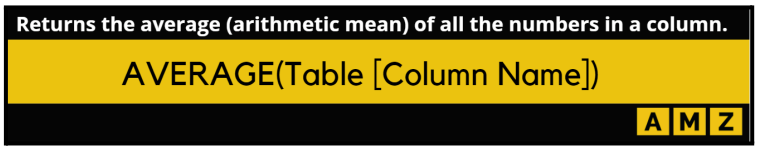
In this Expression:

- *The maximum value of unit price in the Unit Price column from the Sales Table is calculated.*

8. Click on "$" symbol and type "2" in the text box.



AVERAGER():



To use AVERAGE() function:

9. Right click on **All Measures** in the **Fields**.
10. Click on **New measure.**

    Measure tools open.

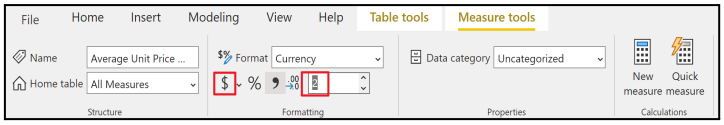
11. Type the following formula in the formula bar and press Enter.

```
Average Unit Price (CM) =
AVERAGE ( Sales[Unit Price] )
```

In this Expression:

- *The average value of unit price in the Unit Price column from the Sales Table is calculated.*

12. Click on "$" symbol and type "2" in the text box.



13. Click on the **matrix** visual in the **visualization pane**.
14. Drag and drop **Category** and **Subcategory** from the Product table to the rows.
15. Drag and drop **Minimum Unit Price (CM), Maximum Unit Price (CM) and Average Unit Price (CM)** from All Measures to the Values.
16. Click on "+" next to Bikes to expand.

Category	Minimum Unit Price (CM)	Maximum Unit Price (CM)	Average Unit Price (CM)
⊞ **Accessories**	**$1.33**	**$159.00**	**$19.69**
⊟ **Bikes**	**$113.00**	**$3,578.27**	**$1,255.08**
Mountain Bikes	$113.00	$3,399.99	$1,451.21
Road Bikes	$234.90	$3,578.27	$1,172.79
Touring Bikes	$334.06	$2,384.07	$1,146.20
⊞ **Clothing**	**$4.32**	**$69.99**	**$32.09**
⊞ **Components**	**$11.74**	**$858.90**	**$251.40**
Total	**$1.33**	**$3,578.27**	**$465.18**

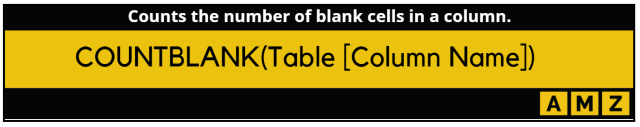
Notice the maximum, minimum and average unit price per category.

Counting Blank Values

Finally lets count the blank values in the column.

Continue using the previous file for the exercise or open the file Basic_DAX_Functions_5.pbix..

COUNTBLANK():



To use COUNTBLANK() function:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Total Resellers (CM) =
COUNT ( Reseller[ResellerKey] )
```

In this Expression:

- *The number of resellers with a non-blank Reseller key are counted.*

You are already familiar with the COUNT() function.

4. Click on the **Card visual** in the **Visualization Pane**.
5. Drag and drop the Total Resellers (CM) to the Field.

Note that there are total 702 resellers in our data. Now lets look at the resellers without a postal code.

6. Right click on **All Measures** in the **Fields**.
7. Click on **New measure.**

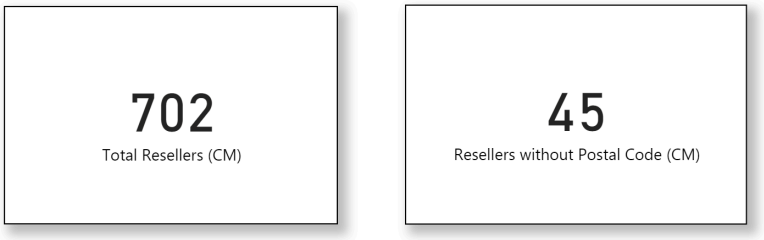
   Measure tools open.
8. Type the following formula in the formula bar and press Enter.

```
Resellers without Postal Code (CM) =
COUNTBLANK ( Reseller[Postal Code] )
```

In this Expression:

- *The number of resellers with a blank postal code are counted.*

9. Click on the **Card visual** in the **Visualization Pane**.
10. Drag and drop the Resellers without Postal Code (CM) to the Field.



Notice that there are 45 resellers from a total of 702 resellers who do not have their postal code value in the current database.

If you're getting a value of 152 resellers without postal code. Focus on the comment below.

It is possible that you do not obtain the same result for the last measure as shown in the image. You can get this by changing the data type of Postal Code column to text in the query editor. This happens because DAX does not allow variant data types in columns of table.

Chapter 5: Evaluation Contexts in DAX

What will you learn in this chapter?

✔ What are the two evaluation contexts in DAX?

✔ What is initial or incoming filter context

✔ What is Row Context

✔ Filter Context vs Row Context

✔ Evaluation Context in Formulas

✔ Rules of Evaluation

What are the two evaluation contexts in DAX?

Any DAX function is evaluated under a context. Evaluation context is an "environment" in which the formula is calculated.

There are two types of evaluation context which are completely different from each other.

Row Context:

When you write an expression in a calculated column, the expression is evaluated for each row of the table, creating a row context for each row. In a similar manner, when you use an iterator like FILTER(), SUMX(), AVERAGEX(), ADDCOLUMNS(), or any one of the DAX functions that iterates over a table or a table expression, it creates a row context.

A row context is evaluated whenever an expression iterates over a table. Row context does not follow relationships and does not create a filter context. DAX functions are used to covert a row context into a filter context.

A row context exists in either a calculated column or an iterating function such as SUMX(). It does not exist in a regular measure. This is because regular measures cannot be created using naked columns. Regular measures only operate on columns and tables so the row filtering does not come into play using measures unless the iterator function is used.

Filter Context:

Filter context is evaluated based on filtering applied on the data model in DAX. The Initial/Incoming filter context comes from

- ✔ Slicers
- ✔ Data Fields of a visual (e.g. Rows and Columns in Matrix)
- ✔ Page or Report Filters
- ✔ From other visuals

Filter context can be modified or replaced by using CALCULATE() function. Filters otherwise automatically follow relationships from 1 to many sides.

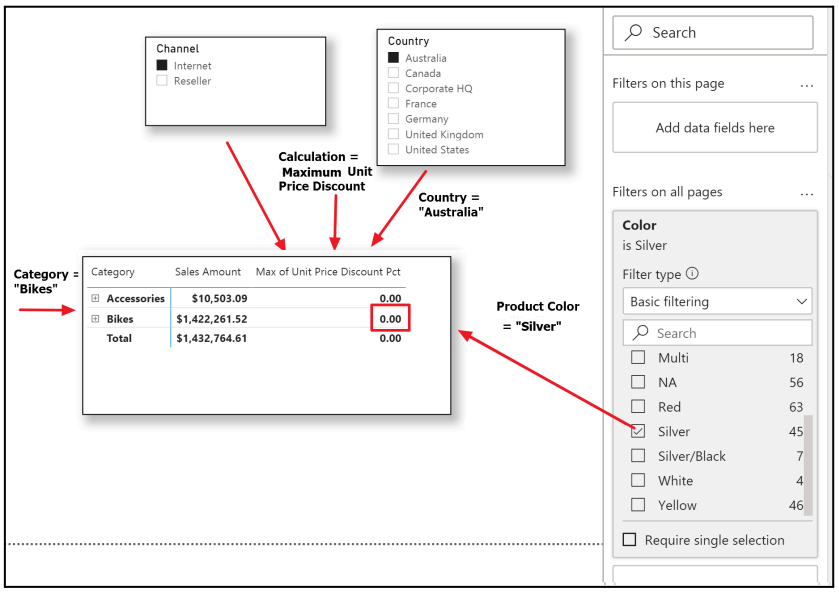
What is initial or incoming filter context

The initial filters applied on the report in the form of slicers, page and report filters. Visual filters act as the first-hand filtering before evaluation context is evaluated.

Continue using the previous file for the exercise or open the file Evaluation_Context_In_DAX_1.pbix..

To see the effect of initial or incoming filter context.

1. Click on the **matrix** in the **visualization pane**.
2. Add the **Category** from the **Product** to the rows.
3. Add the **Sales Amount** and **Unit Price Discount Pct** from the **Sales** table to the values
4. Right click on the **Unit price Discount Pct** in values and change the Calculation to **Maximum.**
5. Drag and drop the **Color** column from the Product table to the **Filters pane**.
6. Select the checkbox next to silver in the filter.
7. Click on **Slicer** in the **Visualization Pane.**
8. Drag and drop **Channel** from SalesOrder table into the **Fields**.
9. Click on **Slicer** in the **Visualization Pane.**
10. Drag and drop **Country** from SalesTerritory table into the **Fields**.
11. Click on **Internet** in the Channel slicer.
12. Click on **Australia** in the Country slicer.



Notice that the value in the red box is dependent on:

✔ Filter on the Rows of matrix.

✔ Filter on the Column of matrix (which specifies the calculations).

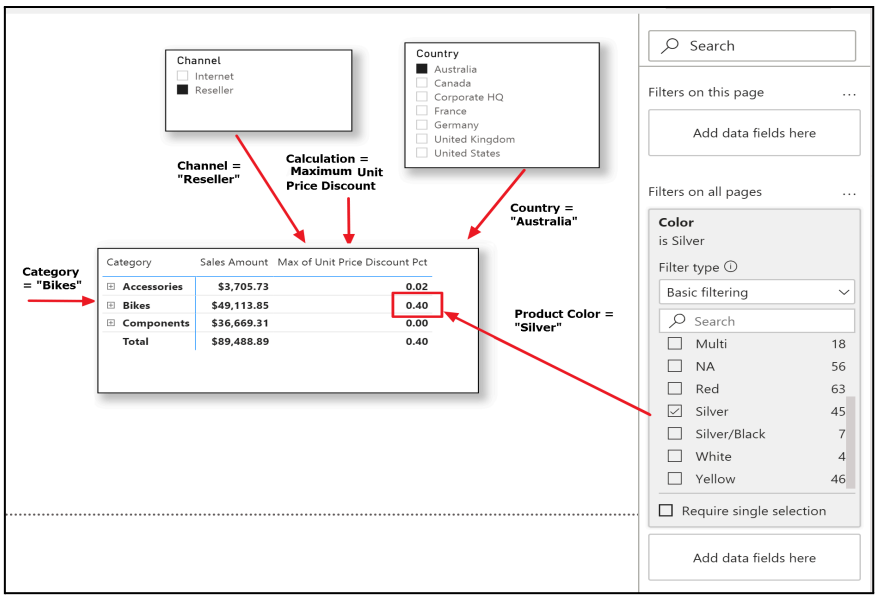
✔ The page level filter for product color.

✔ The filter by slicer by slicing data for channels.

✔ The filter by slicer by slicing data for one country only.

Notice that no discount has been provided using the selected filter values.

13. Click on **Reseller** in the Channel slicer.



Notice the change in values. This means that the Unit price discount was applicable for purchases through resellers only.

What is Row Context

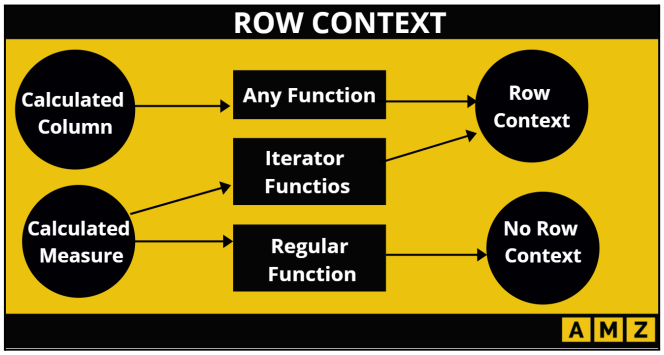
A row context is created when any DAX expression is evaluated for each row. Row context can be created by:

Row context using Calculated Column:

When a calculated column is created, the expression for the values in the new column are evaluated for each row value. This creates a row context. Therefore, any function used for creating a calculated column in Power BI creates a row context.

Row context using Calculated Measure:

Measures may or may not create a row context. If the measure is created using an iterator function, then the expression is evaluated for each row and therefore a row context is created. Whereas, if the measures use regular functions then no row context is created.



Filter Context vs Row Context

There are always two contexts in DAX and these are known as the pillars of DAX. Let's look at some basic differences between these two:

	Filter Context	**Row Context**
Context creation	Provided by coordinates of the visual	Exists in Calculated Column or iterative DAX functions
Role of CALCULATE()	CALCULATE() function can be used to alter a filter context.	CALCULATE() can convert a row context to a filter context.
Relationships	Automatically follow the relationships from the 1 to many side.	Does not follow relationships or create a filter context.

Evaluation Context in Functions

Let's look at some example of the evaluation contexts in functions.

- ✔ RELATED(): This function expands the row context of the current row and allows you to perform VLOOKUP to include values in a column.
- ✔ FILTER(): This function allows you to specify a subset of data to include in the current context.
- ✔ ALL(): This function sets a new context by ignoring the existing filter context.
- ✔ ALLEXCEPT(): This function lets you remove all the filters except the specified one. Thereby, changing the existing filter context.
- ✔ EARLIER() or EARLIEST(): These functions let you perform calculations in a loop by referencing a value in the inner loop.

Rules of Evaluation

When a DAX expression is evaluated there are a series of rules that are executed in order. They form the foundation of how you will use DAX functions.

The Rules for evaluation of DAX Expression are:

- ✔ Evaluation of the Initial/Incoming filter context from Slicers.
- ✔ Data Fields of a visual (e.g. Rows and Columns in Matrix).
- ✔ Page or Report Filters.
- ✔ From other visuals.
- ✔ If CALCULATE () is used, Check for Filter modification.
- ✔ Check for applied filter to underlying table(s)
- ✔ Interpret the filter propagation per relationship(s).

Evaluate the result after catering the effect of the above mentioned filters.

This will be easier to understand once we create some expressions to demonstrate how the evaluation evolves.

There are many exceptions and variations to these rules but will intentionally overlook these nuances to make things simple and more generic. This will allow you to understand the most important fundamentals without getting bogged down by complex details. All those exceptions to the rules are important but I believe you should only go once you've spent enough time practicing these fundamentals and have deep understanding of it.

Chapter 6: Related Functions and Context Transition

What will you learn in this chapter?

✔ Using RELATED()

✔ Using RELATEDTABLE()

✔ RELATED() vs. RELATEDTABLE()

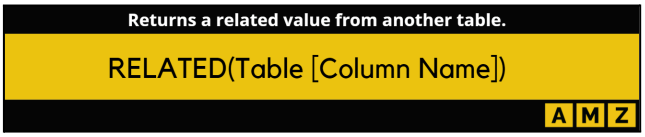
✔ Introduction to Context Transition

✔ Row to Filter Context transition

Using RELATED()

Now that you're familiar with the basic DAX functions, let's look at some functions which can help you fetch data from other tables.

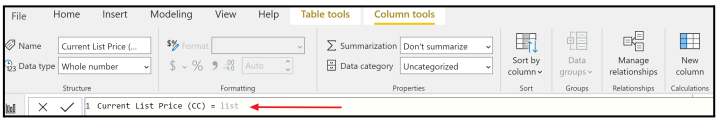
Continue using the previous file for the exercise or open the file Related_Functions_and_Context_Transition_1.pbix..



RELATED() takes value from the one side of the one-to-many relationship and populates it to the many side of the relationship. To use RELATED() for bringing the list price value from the Product table into the Sales table:

1. Right click on the **Sales** table in the **Fields** Pan.
2. Click on **New Column**.

   Column tools open.
3. Type the following formula in the formula bar.



Notice that the intellesence detects that currently List Price does not exist in Sales Table. So intellisense does not give option to select a column from the Product table. We will use RELATED() function to activate the relationship.

4. Type **RELATED** in the formula bar after =.
5. Select the List Price from the Product table.

Your DAX expression should look similar to the expression below.

```
Current List Price (CC) =
RELATED ( 'Product'[List Price] )
```

In this Expression:

- *A calculated column with each row containing the respective List Price from the Product table has been created.*

6. Click on **Data** tab in the left **Ribbon**.
7. Click on **Sales** table in the **Fields**.

rder Quantity	Unit Price	Extended Amount	Product Standard Cost	Total Product Cost	Sales Amount	Unit Price Discount Pct	Profit (CC)	Profit Margin % (CC)	Discount Category (CC)	Current List Price (CC)
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982
1	419.46	419.46	413.15	413.15	$419.46	0	$6.31	0.0150431507175893	No Discount	699.0982

Notice the current list price column at the end.

Using RELATEDTABLE()

RELATEDTABLE() takes values from the many side of the one-to-many relationship and populates the one side of the relationship.

Continue using the previous file for the exercise or open the file Related_Functions_and_Context_Transition_2.pbix..

Since there are multiple values for RALTEDTABLE() takes the table name as the input argument.

To use the RELATEDTABLE() function for calculating the number of sales in Customer Table:

1. Right click on the **Customer** table in the **Fields** pan.
2. Click on **New column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Number of Sales (CC) =
COUNTROWS ( Sales )
```

In this Expression:

- *The rows in the Sales table have been counted.*

4. Click on **Data** tab in the left **Ribbon**.
5. Click on **Customer** table in the **Fields** pan.

CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)
16768	AW00016768	Eduardo Barnes	Concord	California	United States	94519	Wednesday, January 8, 1947	121253
16874	AW00016874	Leslie Munoz	Concord	California	United States	94519	Saturday, September 11, 1954	121253
17525	AW00017525	Adam Wright	Concord	California	United States	94519	Wednesday, June 3, 1959	121253
18552	AW00018552	Kelsey Goel	Concord	California	United States	94519	Sunday, December 29, 2002	121253
28345	AW00028345	Diana Serrano	Concord	California	United States	94519	Wednesday, March 1, 1944	121253
14562	AW00014562	Bryce Stewart	Concord	California	United States	94519	Saturday, July 17, 1954	121253
14919	AW00014919	Erik Torres	Concord	California	United States	94519	Friday, November 30, 1951	121253

As you can see, row context does not follow relationship. RELATEDTABLE() allows us to traverse the chain of relationship.

6. Click on the **Number of Sales (CC)** column.

   Column tool box appears containing the formula bar.
7. Add RELATEDTABLE() before the sales in the formula.

   Your DAX expression should look similar to the expression below.

```
Number of Sales (CC) =
COUNTROWS ( RELATEDTABLE ( Sales ) )
```

In this Expression:

- *The related number of rows from the sales table are counted based on the incoming filter context.*

Notice the change of values in the Number of Sales (CC) column.

CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)
16768	AW00016768	Eduardo Barnes	Concord	California	United States	94519	Wednesday, January 8, 1947	1
16874	AW00016874	Leslie Munoz	Concord	California	United States	94519	Saturday, September 11, 1954	1
17525	AW00017525	Adam Wright	Concord	California	United States	94519	Wednesday, June 3, 1959	1
18552	AW00018552	Kelsey Goel	Concord	California	United States	94519	Sunday, December 29, 2002	1
28345	AW00028345	Diana Serrano	Concord	California	United States	94519	Wednesday, March 1, 1944	1
14562	AW00014562	Bryce Stewart	Concord	California	United States	94519	Saturday, July 17, 1954	2
14919	AW00014919	Erik Torres	Concord	California	United States	94519	Friday, November 30, 1951	2

The Number of Sales (CC) gives the number of Sales transactions by each customer.

RELATED() vs. RELATEDTABLE()

RELATED() and RELATEDTABLE() are the two functions used to traverse one-to-many relationship in DAX. The few key differences between the two functions are:

	RELATED()	**RELATEDTABLE()**
Input Argument	Column name	Table name
Output	Single value from the current row.	A table of values
Takes values from	one side of relationship	many side of the relationship
Populates	many side of the relationship	one side of relationship

Introduction to Context Transition

Now let's create another calculated column which shows the number of distinct products purchased by a customer. When you create this calculated column, you will face the same problem but this time you cannot use RELATEDTABLE() function as it only accepts a table argument. So for this, you will put DISTINCTCOUNT() on Sales[ProductKey].

Continue using the previous file for the exercise or open the file Related_Functions_and_Context_Transition_3.pbix..

To find number of distinct products bought by a customer:

1. Right click on the **Customer** table in the **Fields** pan.
2. Click on **New column**.
3. Type the following formula in the formula bar and press Enter.

```
Number of Products Purchased (CC) =
DISTINCTCOUNT ( Sales[ProductKey] )
```

In this Expression:

- *The unique product keys from the Sales table are counted.*

4. Click on **Data** tab in the left **Ribbon**.
5. Click on **Customer** table in the **Fields** pan.
6. Click on the drop box at the top on **Number of Products Purchased (CC).**
7. Click on **Sort Descending**.

CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)	Number of Products Purchased (CC)
18457	AW00018457	Olivia White	Wollongong	New South Wales	Australia	2500	Thursday, February 21, 1974	6	350
11052	AW00011052	Heidi Lopez	Matraville	New South Wales	Australia	2036	Saturday, October 2, 1982	7	350
11006	AW00011006	Janet Alvarez	Matraville	New South Wales	Australia	2036	Friday, September 10, 1993	5	350
11462	AW00011462	Laura Lin	Matraville	New South Wales	Australia	2036	Friday, December 20, 1946	5	350
11105	AW00011105	Candace Fernandez	Matraville	New South Wales	Australia	2036	Friday, May 11, 1945	5	350
13631	AW00013631	Joe Subram	Matraville	New South Wales	Australia	2036	Friday, November 23, 1990	8	350
11031	AW00011031	Theresa Ramos	Matraville	New South Wales	Australia	2036	Friday, April 2, 1993	6	350
19602	AW00019602	Stanley Madan	Matraville	New South Wales	Australia	2036	Tuesday, August 11, 1998	5	350
20615	AW00020615	Laura Wu	Matraville	New South Wales	Australia	2036	Wednesday, November 10, 1971	5	350

The result is not correct because all of the customers cannot buy the same number of products. However, CALCULATE() allows us to get the result. Calculate converts Row context into filter context. This is called **context transition**. CALCULATE() will be discussed in more detail later in the course.

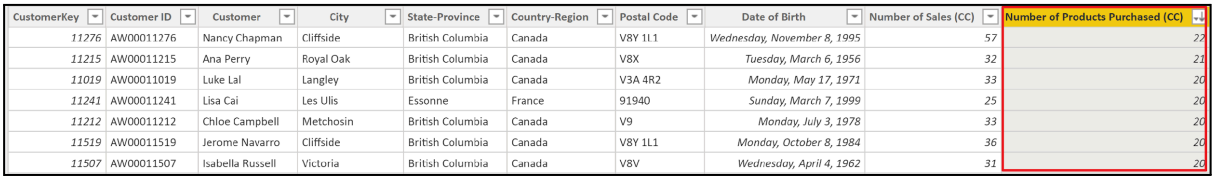
8. Click on the **Number of Products Purchased (CC)** column.
   Column tool box appears containing the formula bar.
9. Add CALCULATE() before the DISTINTCOUMT() in the formula and press Enter.

```
Number of Products Purchased (CC) =
CALCULATE ( DISTINCTCOUNT ( Sales[ProductKey] ) )
```

In this Expression:

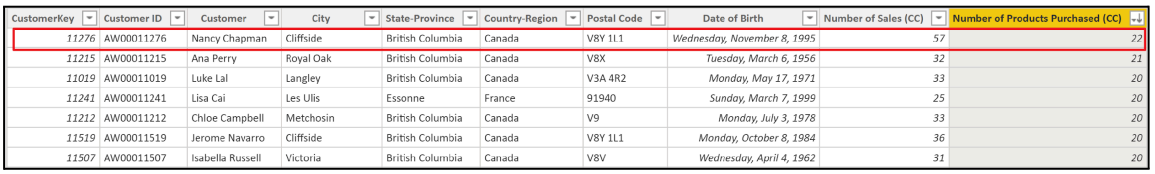
- *The filter context is modified by CALCULATE().*

10. Click on the drop box at the top on **Number of Products Purchased (CC).** Click on **Sort Descending.**



CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)	Number of Products Purchased (CC)
11276	AW00011276	Nancy Chapman	Cliffside	British Columbia	Canada	V8Y 1L1	Wednesday, November 8, 1995	57	22
11215	AW00011215	Ana Perry	Royal Oak	British Columbia	Canada	V8X	Tuesday, March 6, 1956	32	21
11019	AW00011019	Luke Lal	Langley	British Columbia	Canada	V3A 4R2	Monday, May 17, 1971	33	20
11241	AW00011241	Lisa Cai	Les Ulis	Essonne	France	91940	Sunday, March 7, 1999	25	20
11212	AW00011212	Chloe Campbell	Metchosin	British Columbia	Canada	V9	Monday, July 3, 1978	33	20
11519	AW00011519	Jerome Navarro	Cliffside	British Columbia	Canada	V8Y 1L1	Monday, October 8, 1984	36	20
11507	AW00011507	Isabella Russell	Victoria	British Columbia	Canada	V8V	Wednesday, April 4, 1962	31	20

In the example below, you can see Nancy Chapman had 57 sales but only bought 22 different products.



CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)	Number of Products Purchased (CC)
11276	AW00011276	Nancy Chapman	Cliffside	British Columbia	Canada	V8Y 1L1	Wednesday, November 8, 1995	57	22
11215	AW00011215	Ana Perry	Royal Oak	British Columbia	Canada	V8X	Tuesday, March 6, 1956	32	21
11019	AW00011019	Luke Lal	Langley	British Columbia	Canada	V3A 4R2	Monday, May 17, 1971	33	20
11241	AW00011241	Lisa Cai	Les Ulis	Essonne	France	91940	Sunday, March 7, 1999	25	20
11212	AW00011212	Chloe Campbell	Metchosin	British Columbia	Canada	V9	Monday, July 3, 1978	33	20
11519	AW00011519	Jerome Navarro	Cliffside	British Columbia	Canada	V8Y 1L1	Monday, October 8, 1984	36	20
11507	AW00011507	Isabella Russell	Victoria	British Columbia	Canada	V8V	Wednesday, April 4, 1962	31	20

Last Order Date of a Customer:

To find last order date by a customer using context transition:

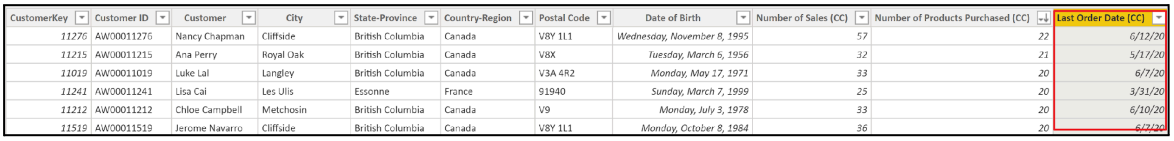
11. Right click on the **Customer** table in the **Fields** pan. Click on **New column** and then Type the following formula in the formula bar and press Enter.

```
Last Order Date (CC) =
CALCULATE ( MAX ( Sales[OrderDate] ) )
```

In this Expression:

- *The MAX() function finds the latest order date in the Sales Table.*
- *CALCULATE() modifies the filter context so that the last order date for each customer can be filtered.*

12. Format the Last Order Date (CC) column to (m/d/yy).
13. Click on **Data** tab in the left **Ribbon**. then
14. Click on **Customer** table in the **Fields** pan.



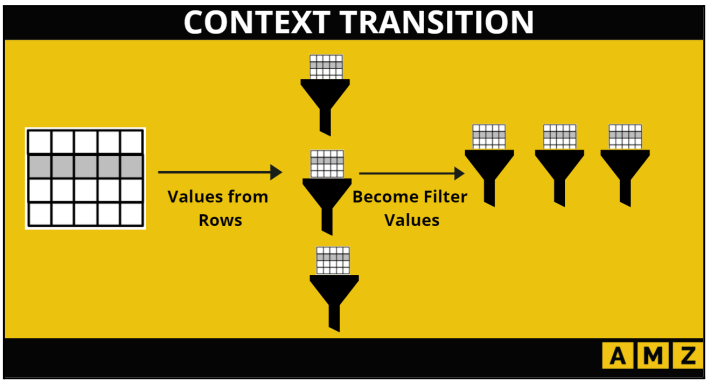
CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)	Number of Products Purchased (CC)	Last Order Date (CC)
11276	AW00011276	Nancy Chapman	Cliffside	British Columbia	Canada	V8Y 1L1	Wednesday, November 8, 1995	57	22	6/12/20
11215	AW00011215	Ana Perry	Royal Oak	British Columbia	Canada	V8X	Tuesday, March 6, 1956	32	21	5/17/20
11019	AW00011019	Luke Lal	Langley	British Columbia	Canada	V3A 4R2	Monday, May 17, 1971	33	20	6/7/20
11241	AW00011241	Lisa Cai	Les Ulis	Essonne	France	91940	Sunday, March 7, 1999	25	20	3/31/20
11212	AW00011212	Chloe Campbell	Metchosin	British Columbia	Canada	V9	Monday, July 3, 1978	33	20	6/10/20
11519	AW00011519	Jerome Navarro	Cliffside	British Columbia	Canada	V8Y 1L1	Monday, October 8, 1984	36	20	6/2/20

Notice that Nancy last ordered on 12th of June, 2020.

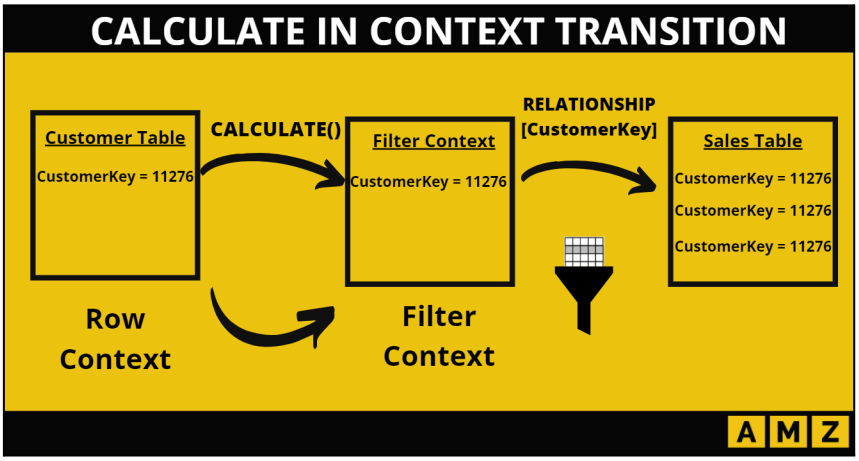
It is important to format the newly created columns to the format used in your organization. For this course we have followed the m/d/yy format for the date.

Row to Filter Context transition

As you have seen in the last exercise, CALCULATE() is used to perform row to filter context transition. The image below depicts what row to filter context transition looks like.



Row context does not follow relationships but filter context does follow relationships. So CALCULATE() take the current row, transform it into a filter context and after CALCULATE transitions to the new filter context (before the expression in CALCULATE is evaluated) the CustomerKey value will filter records in the Sales table.



Chapter 7: Table Functions

What will you learn in this chapter?

- ✔ Creating a Calculated Table
- ✔ Using VALUES()
- ✔ DISTICT vs. VALUES()
- ✔ Using ALL() to ignore Filters.
- ✔ Creating summary table for the requested totals using SUMMARIZE()
- ✔ Using ALLEXCEPT()
- ✔ Using FILTER() as a Table function
- ✔ Virtual table lineage
- ✔ Mixing table functions
- ✔ The Cross Joint Measures
- ✔ Understanding the diff
- ✔ Exploring Table Joins
- ✔ Testing table function for VALUES(), DISTINCT(), ALL() and ALLNOBLANKROW()

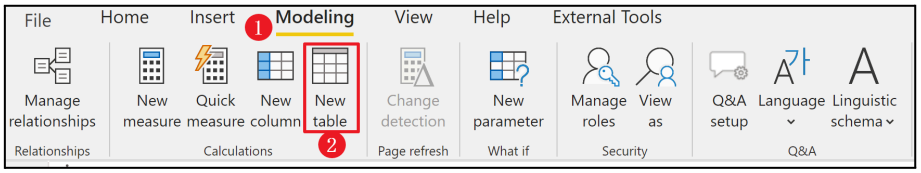
Creating a Calculated Table

Using DAX expressions, you can calculate tables containing values of your choice. This is as easy as creating calculated columns and calculated measures. Calculated tables have all the properties of regular tables in Power BI desktop. Some of the common functions used for creating a calculated table are:

- ✔ DISTINCT()
- ✔ VALUES()
- ✔ CROSSJOIN()
- ✔ ALL()
- ✔ ALLEXCEPT()
- ✔ FILTER()
- ✔ SUMMARIZE()
- ✔ CALENDAR()
- ✔ CALENDERAUTO()
- ✔ UNION()

To create a calculated table:

1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New table**.

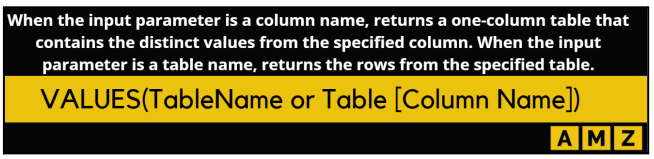


All the DAX code snippets where calculated tables have been created are color coded yellow for this manual.

Using VALUES()

The first table function that you will be using to calculate tables is VALUES().

Continue using the previous file for the exercise or open the file Table_Functions_1.pbix..



The result of VALUES() depends on the input argument:

- ✔ When the input argument is a table, all the rows of the table are returned. Duplicate rows are preserved.
- ✔ When the input argument is a column, a single-column table is returned containing distinct values. Duplicates are removed in this case and only unique values are returned.

Values using table name as argument:

To see how VALUES() is used to create tables:

1. Click on the **Modeling** tab in the **Ribbon**. Click on **New Table**.
2. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions using VALUES (CT) =
VALUES ( 'Product' )
```

In this Expression:

- *The rows from Product table are stored in a new calculated table.*

3. Click on the **Data** tab in the left **Ribbon**.
4. Click on Testing Table Functions using VALUES (CT) in the Fields pan.

Category	ProductKey	Product	Standard Cost	Color	List Price	Model	Subcategory	SKU
Bikes	376	Road-250 Black, 48	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	BK-R89B-48
Bikes	378	Road-250 Black, 52	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	BK-R89B-52
Bikes	380	Road-250 Black, 58	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	BK-R89B-58
Bikes	374	Road-250 Black, 44	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	BK-R89B-44
Bikes	606	Road-750 Black, 52	$343.65	Black	$539.99	Road-750	Road Bikes	BK-R19B-52
Bikes	604	Road-750 Black, 44	$343.65	Black	$539.99	Road-750	Road Bikes	BK-R19B-44
Bikes	605	Road-750 Black, 48	$343.65	Black	$539.99	Road-750	Road Bikes	BK-R19B-48
Bikes	584	Road-750 Black, 58	$343.65	Black	$539.99	Road-750	Road Bikes	BK-R19B-58
Bikes	337	Road-650 Black, 62	$486.71	Black	$782.99	Road-650	Road Bikes	BK-R50B-62
Bikes	333	Road-650 Black, 58	$486.71	Black	$782.99	Road-650	Road Bikes	BK-R50B-58
Bikes	335	Road-650 Black, 60	$486.71	Black	$782.99	Road-650	Road Bikes	BK-R50B-60
Bikes	339	Road-650 Black, 44	$486.71	Black	$782.99	Road-650	Road Bikes	BK-R50B-44
Bikes	341	Road-650 Black, 48	$486.71	Black	$782.99	Road-650	Road Bikes	BK-R50B-48
Bikes	343	Road-650 Black, 52	$486.71	Black	$782.99	Road-650	Road Bikes	BK-R50B-52
Components	255	LL Road Frame - Black, 58	$204.63	Black	$337.22	LL Road Frame	Road Frames	FR-R38B-58

Values using column name as argument

Notice that all the rows from the Product table are returned. Now we will what happens when a column is used as an argument in VALUES() function.

5. Click on Testing Table Functions using VALUES (CT) in the Fields pan.

   Table tools menu box opens.

6. Replace the Product table with Product Category column and press Enter.

```
Testing Table Functions using VALUES (CT) =
VALUES ( 'Product'[Category] )
```

In this Expression:

- *The unique product categories from the Product table are found because the argument to VALUES() is s table column.*

7. Click on the **Data** tab in the left **Ribbon**.
8. Click on Testing Table Functions using VALUES (CT) in the Fields pan.

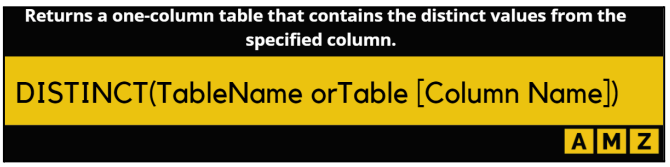
Category
Clothing
Bikes
Components
Accessories

Notice that only distinct values without any duplicates have been returned in the form of a single column table.

DISTINCT vs. VALUES()

DISTINCT() may look similar to VALUES() but they are slightly different.

Continue using the previous file for the exercise or open the file Table_Functions_2.pbix..



To see how DISTINCT() is used to create tables:

1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions using DISTINCT (CT) =
DISTINCT ( 'Product'[Category] )
```

In this Expression:

- *The unique product categories from the Product table are found because the argument to DISTINCT() is s table column*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Testing Table Functions using DISTINCT (CT) in the Fields pan.

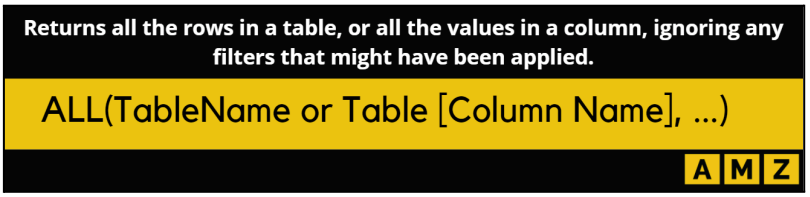
Category
Clothing
Bikes
Components
Accessories

Although both of these functions return non-duplicating distinct values when a column is passed as an argument, DISTINCT() does not count BLANK() as a value whereas, VALUES() returns BLANK(). This concept will be demonstrated more clearly in the last exercise of this chapter.

Using ALL() to ignore Filters

ALL() gives same result as DISTINCT() and VALUES() with a column name argument but allows you to select more than 1 column and it returns a unique combination of both columns.

Continue using the previous file for the exercise or open the file Table_Functions_3.pbix..



ALL() has a special feature which makes it different from all other functions. ALL() ignores the filter context. The result of ALL() depends on the input argument:

- ✔ When the table is passed as an argument, all the rows of a table returned (including duplicate rows) are returned ignoring any filter context. These values can be used by an X-function or iterators.
- ✔ When the column is passed as an argument, all the unique values are returned.

ALL() using single column:

To see how ALL() is used to create tables:

1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions using ALL (CT) =
ALL ( 'Product'[Category] )
```

In this Expression:

- *The unique product categories from the Product table are found because the argument to ALL() is s table column*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Testing Table Functions using ALL (CT) in the Fields pan.

Category
Clothing
Bikes
Components
Accessories

ALL() using multiple columns:

To understand the behavior of ALL() using multiple columns:

6. Click on the **Modeling** tab in the **Ribbon**.
7. Click on **New Table**.
8. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions ALL - two col (CT) =
ALL ( 'Product'[Category], 'Product'[Subcategory] )
```

In this Expression:

- *Two columns from the Product table are used as arguments in the ALL() function.*

9. Click on the **Data** tab in the left **Ribbon**.
10. Click on Testing Table Functions using ALL – two col (CT) in the Fields pan.

Subcategory	Category
Jerseys	Clothing
Caps	Clothing
Gloves	Clothing
Shorts	Clothing
Tights	Clothing
Bib-Shorts	Clothing
Socks	Clothing
Vests	Clothing

Notice that distinct combination of category and sub category are returned. Let's see what happens when two columns from different data sets are used as arguments in ALL().

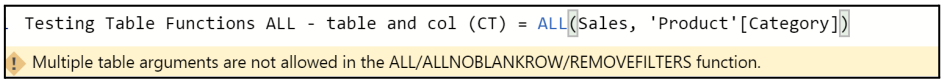
11. Click on the **Modeling** tab in the **Ribbon**.
12. Click on **New Table**.
13. Type the following expression in the formula bar.

```
1  Testing Table Functions using ALL (CT) = ALL('Product'[Category], SalesTerritory[Country])
```

All column arguments of the ALL/ALLNOBLANKROW/ALLSELECTED/REMOVEFILTERS function must be from the same table.

You can't select two columns from different tables. Now let's see what happens when a table and column (of a different table) are used as arguments in ALL().

14. Click on the **Modeling** tab in the **Ribbon**.
15. Click on **New Table**.
16. Type the following expression in the formula bar.



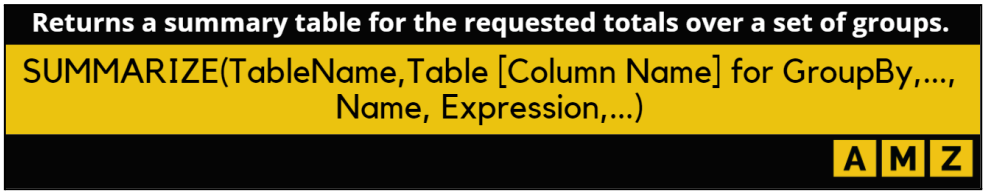
The warning shows that this type of syntax combination is not allowed in ALL().

This can be done using SUMMARIZE(). Which will be explained in the next topic.

Creating summary table for the requested totals using SUMMARIZE()

SUMMARIZE() allows us to create a summary of the input table grouped by the specified columns.

Continue using the previous file for the exercise or open the file Table_Functions_4.pbix..



Totals over a Group-by column:

Let's create a summary table where the input table is Sales while the category is coming from the Product Table. To use SUMMARIZE() for this:

1. Click on the Modeling tab in the Ribbon.
2. Click on New Table.
3. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions using SUMMARIZE (CT) =
SUMMARIZE ( Sales, 'Product'[Category] )
```

In this Expression:

- *The Sales table is grouped by the product Category.*

4. Click on the Data tab in the left Ribbon.
5. Click on Testing 'Table Functions using SUMMARIZE (CT)' in the Fields pan.

Category
Clothing
Bikes
Components
Accessories

Totals over a multiple Group-by columns:

6. Click on the Modeling tab in the Ribbon.
7. Click on New Table.
8. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions SUMMARIZE - 2 col (CT) =
SUMMARIZE ( Sales, 'Product'[Category], 'Product'[Subcategory] )
```

In this Expression:

- *The Sales table is grouped by the Product Category and Subcategory.*

9. Click on the Data tab in the left Ribbon.
10. Click on Testing 'Table Functions using SUMMARIZE – 2 col (CT)' in the Fields pan.

Category	Subcategory
Clothing	Jerseys
Clothing	Caps
Clothing	Gloves
Clothing	Shorts
Clothing	Tights
Clothing	Bib-Shorts
Clothing	Socks
Clothing	Vests
Bikes	Road Bikes
Bikes	Touring Bikes
Bikes	Mountain Bikes

11. Scroll down to the end of table.

	Accessories	Helmets
TABLE: Testing Table Functions SUMMARIZE - 2 col (CT) (35 rows) COLUMN: Category (4 distinct values)		

The table statistics show that 35 rows exist in this table. however, we have 37 Sub-categories. This means 2 Subcategories for which we have no sales does not appear here.

This happens because input table of the function is Sales and it only brings those categories and subcategories where we have a sale (at least 1 row in sales table)

SUMMARIZE() using measure or expression for GroupBy:

12. Click on the Modeling tab in the Ribbon.
13. Click on New Table.
14. Type the following expression in the formula bar and press Enter.

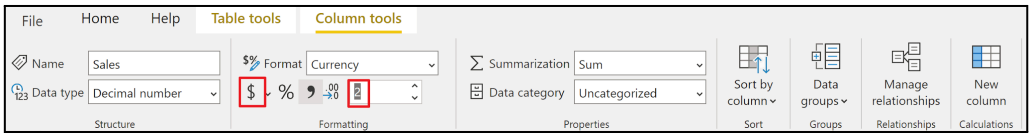
```
Testing Table Functions SUMMARIZE - Expression or Measure (CT) =
SUMMARIZE (
    Sales,
    'Product'[Category],
    'Product'[Subcategory],
    "Sales", SUM ( Sales[Sales Amount] ),
    "Total Cost", SUM ( Sales[Total Product Cost] )
)
```

In this Expression:

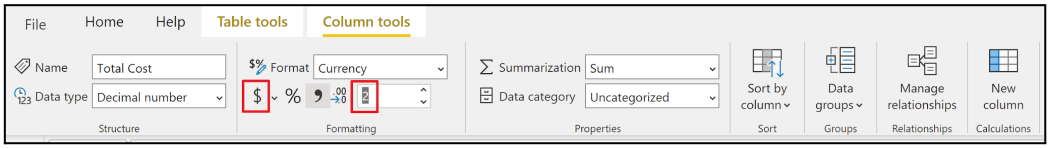
- *The Sales table is grouped by the product Category, Subcategory, Sales Amount and Total Product Cost.*

- *The "Sales" column stores the aggregated value of Sales Amount for each row of the new table.*
- *The "Total Cost" column stores the aggregated value of Total Product Cost for each row of the new table.*

15. Click on the Sales Column in Testing Table Functions SUMMARIZE – Expression or Measure (CT).
    Column tools open.
16. Click on the $ symbol and write 2 in the text box to set formatting.



17. Click on the Total Cost Column in Testing Table Functions SUMMARIZE – Expression or Measure (CT).
    Column tools open.
18. Click on the $ symbol and write 2 in the text box to set formatting.



19. Click on the Data tab in the left Ribbon.
20. Click on Testing Table Functions using SUMMARIZE – Expression or Measure (CT) in the Fields pan.

Category	Subcategory	Sales	Total Cost
Clothing	Jerseys	$752,257.82	$845,169.46
Clothing	Caps	$51,230.10	$52,428.59
Clothing	Gloves	$242,796.58	$159,384.76
Clothing	Shorts	$413,522.68	$257,581.51
Clothing	Tights	$201,832.98	$141,117.89
Clothing	Bib-Shorts	$166,739.64	$115,482.06
Clothing	Socks	$29,745.21	$17,580.22
Clothing	Vests	$259,488.51	$160,022.43
Bikes	Road Bikes	$43,878,791.78	$39,513,893.82
Bikes	Touring Bikes	$14,296,290.29	$14,079,009.59
Bikes	Mountain Bikes	$36,445,444.52	$30,512,532.67
Components	Road Frames	$3,849,853.72	$3,710,985.95
Components	Touring Frames	$1,642,327.49	$1,647,899.57

Adding columns from other data tables:

You can also add columns from other tables but this is only possible when the input table is on the many side of the relationship. Let's see how this can be done.

21. Click on the Modeling tab in the Ribbon.
22. Click on New Table.
23. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions SUMMARIZE - Expression or Measure (CT) =
SUMMARIZE (
    Sales,
    'Product'[Category],
    'Product'[Subcategory],
    'SalesTerritory'[Country],
    "Sales", SUM ( Sales[Sales Amount] ),
    "Total Cost", SUM ( Sales[Total Product Cost] )
)
```

In this Expression:

- *The Sales table is grouped by the product Category, Subcategory, Country, Sales Amount and Total Product Cost.*
- *The "Sales" column stores the aggregated value of Sales Amount for each row of the new table.*
- *The "Total Cost" column stores the aggregated value of Total Product Cost for each row of the new table.*

24. Click on the Data tab in the left Ribbon.
25. Click on Testing Table Functions using SUMMARIZE – Expression or Measure (CT) in the Fields pan.

Category	Subcategory	Country	Sales	Total Cost
Components	Road Frames	Germany	$36,061.61	$36,470.23
Components	Touring Frames	Germany	$208,071.46	$207,863.25
Components	Mountain Frames	Germany	$57,716.03	$52,675.53
Components	Chains	Germany	$522.20	$386.43
Components	Saddles	Germany	$3,256.86	$2,410.07
Components	Handlebars	Germany	$4,449.59	$3,292.73
Components	Pedals	Germany	$4,316.62	$3,194.41
Components	Bottom Brackets	Germany	$2,591.71	$1,917.84
Components	Derailleurs	Germany	$3,344.75	$2,475.09
Components	Cranksets	Germany	$10,999.43	$8,139.64
Components	Brakes	Germany	$2,939.40	$2,175.17
Components	Road Frames	United Kingdom	$189,891.58	$184,719.98
Components	Touring Frames	United Kingdom	$135,337.74	$136,490.71
Components	Mountain Frames	United Kingdom	$283,875.97	$256,617.28
Components	Chains	United Kingdom	$660.23	$494.27

Please note that you might get a different order of columns when you try to add the Country column from Sales Territory in an existing table. You can fix this by deleting the original table and recreating it with same expression.

Using SUMMARIZE() to get data for all categories:

If you use Product table as an input table and get category and SubCategory columns over the requested totals, you will get all Subcategories. Let's see how this works:

26. Click on the 'Testing Table Functions using SUMMARIZE(CT)' in Fields.

27. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions using SUMMARIZE (CT) =
SUMMARIZE ( 'Product', 'Product'[Category], 'Product'[Subcategory] )
```

In this expression:

- *you have replaced the first argument from 'Sales' to 'Product'.*
- *Added a column for Product Subcategroy as the last argument of SUMMARIZE().*

28. Click on the Data tab in the left Ribbon.
29. Click on Testing Table Functions using SUMMARIZE (CT) in the Fields pan.

Category	Subcategory
Clothing	Jerseys
Clothing	Caps
Clothing	Gloves
Clothing	Shorts
Clothing	Tights
Clothing	Bib-Shorts
Clothing	Socks
Clothing	Vests

30. Scroll to the end to this table.

TABLE: Testing Table Functions using SUMMARIZE (CT) (37 rows)

Notice that the table contains 37 rows, which means all the sub categories are present. Now let's add totals from the Sales column.

Now lets try adding measures.

31. Right click on All Measures in the Fields. Click on New measure.

    Measure tools open.

32. Type the following formula in the formula bar and press Enter.

```
Total Sales Amount (CM) =
SUM ( Sales[Sales Amount] )
```

In this expression:

- *The total sales amount is calculated by summation of Sales Amount values in the Sales table.*

33. Click on the Testing Table Functions using SUMMARIZE (CT) in the Fields pan.
34. Add "Sales", [Total Sales Amount (CM)] as the last argument of SUMMARIZE().

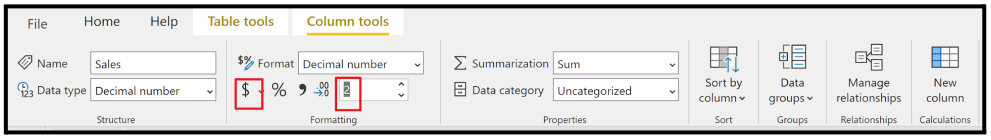
Your expression should look similar to the one below.

```
Testing Table Functions using SUMMARIZE (CT) =
SUMMARIZE (
    'Product',
    'Product'[Category],
    'Product'[Subcategory],
    "Sales", [Total Sales Amount (CM)]
)
```

In this Expression:

- *The Product table is grouped by the product Category, Subcategory and Sales Amount.*
- *The "Sales" column stores the result of the measure for each row of the new table.*

35. Click on Sales column in the Testing Table Functions using SUMMARIZE (CT) in the Fields pan.
36. Click on the $ symbol and write 2 in the text box to set formatting.



37. Click on the Data tab in the left Ribbon. Click on Testing Table Functions using SUMMARIZE (CT) in the Fields pan.
38. Click on the drop down on the Sales Column. Click on Sort Ascending.

Category	Subcategory	Sales
Accessories	Lights	
Accessories	Panniers	
Components	Chains	$9,377.61
Accessories	Pumps	$13,514.71
Accessories	Locks	$16,225.22
Accessories	Cleaners	$18,407.03
Clothing	Socks	$29,745.21
Accessories	Bike Stands	$39,591.00
Accessories	Fenders	$46,619.58
Clothing	Caps	$51,230.10

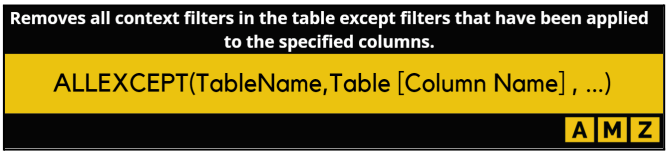
Notice two subcategories where there is no sale in the Sales Table. This shows that if the input table is on one side of the relationship, you cannot fetch Columns from other datasets.

SUMMARIZE() function (unlike ALL) does not ignore the filter context.

Using ALLEXCEPT()

ALLEXCEPT() returns all the columns apart from the ones mentioned in the argument. You can specify more than 1 column in the argument.

Continue using the previous file for the exercise or open the file Table_Functions_5.pbix..



The first argument to the ALLEXCEPT() function must be a table and all subsequent arguments must be references to columns. The function may not be so useful to use at it is. It is rather used as a virtual table for further calculations.

Let's create a table where the input table is Product but the resultant table does not include the Product Key. To use ALLEXCEPT() for this:

1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
Testing Table Functions using ALLEXCEPT (CT) =
ALLEXCEPT ( 'Product', 'Product'[ProductKey] )
```

In this Expression:

- *A new table containing all the columns from the Product table except the Product Key are calculated while ignoring the incoming filters.*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Testing Table Functions using ALLEXCEPT (CT) in the Fields pan.

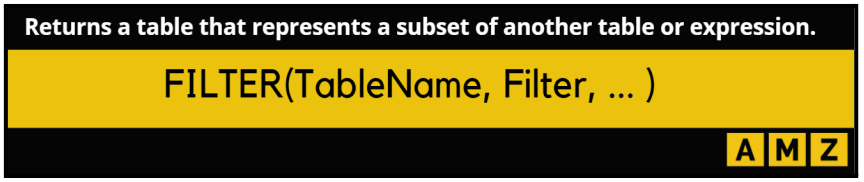
Product	Standard Cost	Color	List Price	Model	Subcategory	Category	SKU
Road-250 Black, 48	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-48
Road-250 Black, 52	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-52
Road-250 Black, 58	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-58
Road-250 Black, 44	$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-44
Road-750 Black, 52	$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-52
Road-750 Black, 44	$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-44
Road-750 Black, 48	$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-48
Road-750 Black, 58	$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-58
Road-650 Black, 62	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-62
Road-650 Black, 58	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-58
Road-650 Black, 60	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-60

Notice that all the columns from the Product table have been returned except the Product Key.

Using FILTER() as a Table function

The FILTER() function is used to slice the data based on certain conditions. The resulting table is a subset of the original table containing only the rows that fulfill the given condition. The resulting table can also be a virtual table that can be used for iteration by other functions.

Continue using the previous file for the exercise or open the file Table_Functions_6.pbix..



Let's create a table where only the products that fall in the category of "Accessories". To use FILTER() for this:

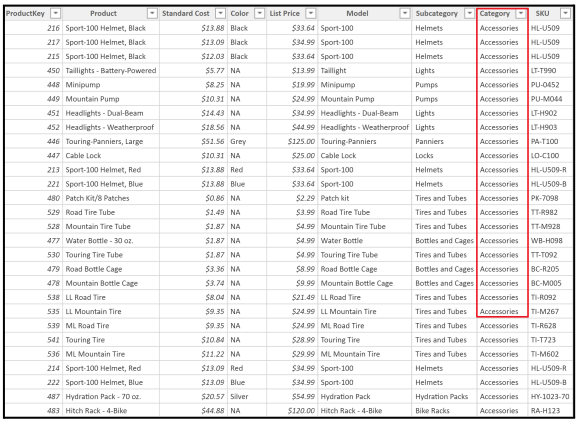
1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
Testing table functions using FILTER (CT) =
FILTER ( 'Product', 'Product'[Category] = "Accessories" )
```

In this Expression:

- *Only the products from the category of "Accessories" are filtered.*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Testing Table Functions using FILTER (CT) in the Fields pan.



ProductKey	Product	Standard Cost	Color	List Price	Model	Subcategory	Category	SKU
216	Sport-100 Helmet, Black	$13.88	Black	$33.64	Sport-100	Helmets	Accessories	HL-U509
217	Sport-100 Helmet, Black	$13.09	Black	$34.99	Sport-100	Helmets	Accessories	HL-U509
215	Sport-100 Helmet, Black	$12.03	Black	$33.64	Sport-100	Helmets	Accessories	HL-U509
450	Taillights - Battery-Powered	$5.77	NA	$13.99	Taillight	Lights	Accessories	LT-T990
448	Minipump	$8.25	NA	$19.99	Minipump	Pumps	Accessories	PU-0452
449	Mountain Pump	$10.31	NA	$24.99	Mountain Pump	Pumps	Accessories	PU-M044
451	Headlights - Dual-Beam	$14.43	NA	$34.99	Headlights - Dual-Beam	Lights	Accessories	LT-H902
452	Headlights - Weatherproof	$18.56	NA	$44.99	Headlights - Weatherproof	Lights	Accessories	LT-H903
446	Touring-Panniers, Large	$51.56	Grey	$125.00	Touring-Panniers	Panniers	Accessories	PA-T100
447	Cable Lock	$10.31	NA	$25.00	Cable Lock	Locks	Accessories	LO-C100
213	Sport-100 Helmet, Red	$13.88	Red	$33.64	Sport-100	Helmets	Accessories	HL-U509-R
221	Sport-100 Helmet, Blue	$13.88	Blue	$33.64	Sport-100	Helmets	Accessories	HL-U509-B
480	Patch Kit/8 Patches	$0.86	NA	$2.29	Patch kit	Tires and Tubes	Accessories	PK-7098
529	Road Tire Tube	$1.49	NA	$3.99	Road Tire Tube	Tires and Tubes	Accessories	TT-R982
528	Mountain Tire Tube	$1.87	NA	$4.99	Mountain Tire Tube	Tires and Tubes	Accessories	TT-M928
477	Water Bottle - 30 oz.	$1.87	NA	$4.99	Water Bottle	Bottles and Cages	Accessories	WB-H098
530	Touring Tire Tube	$1.87	NA	$4.99	Touring Tire Tube	Tires and Tubes	Accessories	TT-T092
479	Road Bottle Cage	$3.36	NA	$8.99	Road Bottle Cage	Bottles and Cages	Accessories	BC-R205
478	Mountain Bottle Cage	$3.74	NA	$9.99	Mountain Bottle Cage	Bottles and Cages	Accessories	BC-M005
538	LL Road Tire	$8.04	NA	$21.49	LL Road Tire	Tires and Tubes	Accessories	TI-R092
535	LL Mountain Tire	$9.35	NA	$24.99	LL Mountain Tire	Tires and Tubes	Accessories	TI-M267
539	ML Road Tire	$9.35	NA	$24.99	ML Road Tire	Tires and Tubes	Accessories	TI-R628
541	Touring Tire	$10.84	NA	$28.99	Touring Tire	Tires and Tubes	Accessories	TI-T723
536	ML Mountain Tire	$11.22	NA	$29.99	ML Mountain Tire	Tires and Tubes	Accessories	TI-M602
214	Sport-100 Helmet, Red	$13.09	Red	$34.99	Sport-100	Helmets	Accessories	HL-U509-R
222	Sport-100 Helmet, Blue	$13.09	Blue	$34.99	Sport-100	Helmets	Accessories	HL-U509-B
487	Hydration Pack - 70 oz.	$20.57	Silver	$54.99	Hydration Pack	Hydration Packs	Accessories	HY-1023-70
483	Hitch Rack - 4-Bike	$44.88	NA	$120.00	Hitch Rack - 4-Bike	Bike Racks	Accessories	RA-H123

Notice that the new table contains only the product details for accessories. Please note that you can format any column (such as Standard Cost and List Price) of your choice using the formatting section.

Filter Using Multiple Conditions:

Now let's create a table containing only those accessories that have the list price greater than 30.

6. Click on the **Modeling** tab in the **Ribbon**.
7. Click on **New Table**.
8. Type the following expression in the formula bar and press Enter.

```
Testing table functions using FILTER with multiple conditions (CT) =
FILTER (
    'Product',
    'Product'[Category] = "Accessories"
        && 'Product'[List Price] > 30
)
```

In this Expression:

- *Only the products from the category of "Accessories" that have a list price greater than 30 are filtered.*

9. Click on the **Data** tab in the left **Ribbon**.
10. Click on Testing Table Functions using FILTER (CT) in the Fields pan.

ProductKey	Product	Standard Cost	Color	List Price	Model	Subcategory	Category	SKU
216	Sport-100 Helmet, Black	$13.88	Black	$33.64	Sport-100	Helmets	Accessories	HL-U509
217	Sport-100 Helmet, Black	$13.09	Black	$34.99	Sport-100	Helmets	Accessories	HL-U509
215	Sport-100 Helmet, Black	$12.03	Black	$33.64	Sport-100	Helmets	Accessories	HL-U509
451	Headlights - Dual-Beam	$14.43	NA	$34.99	Headlights - Dual-Beam	Lights	Accessories	LT-H902
452	Headlights - Weatherproof	$18.56	NA	$44.99	Headlights - Weatherproof	Lights	Accessories	LT-H903
446	Touring-Panniers, Large	$51.56	Grey	$125.00	Touring-Panniers	Panniers	Accessories	PA-T100
213	Sport-100 Helmet, Red	$13.88	Red	$33.64	Sport-100	Helmets	Accessories	HL-U509-R
221	Sport-100 Helmet, Blue	$13.88	Blue	$33.64	Sport-100	Helmets	Accessories	HL-U509-B
214	Sport-100 Helmet, Red	$13.09	Red	$34.99	Sport-100	Helmets	Accessories	HL-U509-R
222	Sport-100 Helmet, Blue	$13.09	Blue	$34.99	Sport-100	Helmets	Accessories	HL-U509-B
487	Hydration Pack - 70 oz.	$20.57	Silver	$54.99	Hydration Pack	Hydration Packs	Accessories	HY-1023-70
483	Hitch Rack - 4-Bike	$44.88	NA	$120.00	Hitch Rack - 4-Bike	Bike Racks	Accessories	RA-H123
486	All-Purpose Bike Stand	$59.47	NA	$159.00	All-Purpose Bike Stand	Bike Stands	Accessories	ST-1401
537	HL Mountain Tire	$13.09	NA	$35.00	HL Mountain Tire	Tires and Tubes	Accessories	TI-M823
540	HL Road Tire	$12.19	NA	$32.60	HL Road Tire	Tires and Tubes	Accessories	TI-R982
212	Sport-100 Helmet, Red	$12.03	Red	$33.64	Sport-100	Helmets	Accessories	HL-U509-R
220	Sport-100 Helmet, Blue	$12.03	Blue	$33.64	Sport-100	Helmets	Accessories	HL-U509-B

Notice that the resulting table contains only those accessories that have the list price greater than 30.

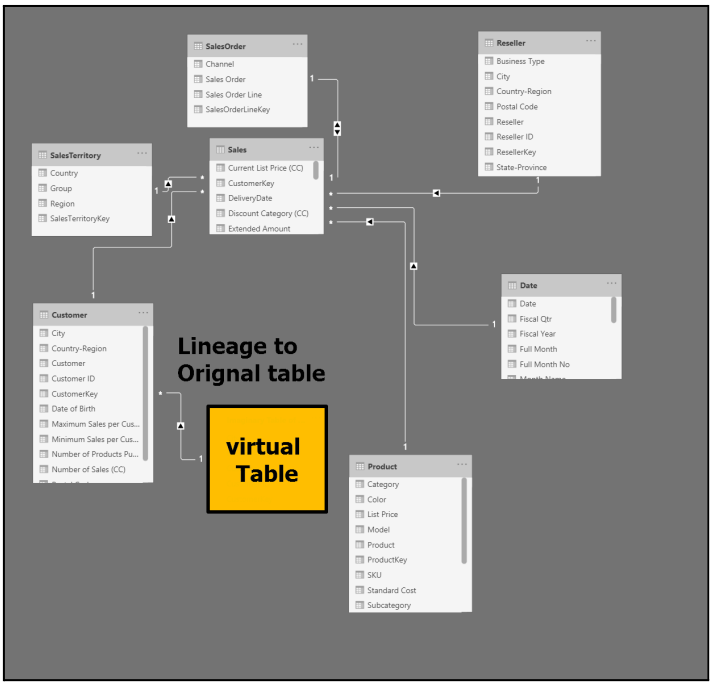
Virtual table lineage

Virtual table lineage is the link retained by any virtual table to the original table. For example, consider the DAX code shown below:

```
Imaginary Table of Customers with Sales Greater Than $5,000 =
CALCULATE (
    COUNTROWS ( Customer ),
    FILTER ( Customer, [Total Sales Amount (CM)] >= 5000 )
)
```

In the formula above, it may look like FILTER() pulls the data from the Customers table and functions in a standalone manner. This is however not the case, all table functions maintain a link to the original table and this is called lineage. This means that as soon as the Customer table is updated, the table maintaining lineage with it is also updated.

The imaginary table is considered a part of data model but it is never materialized or physically created.



Mixing table functions

Now that you have used the table functions individually, lets mix them together to see the right usefulness of DAX functions.

Continue using the previous file for the exercise or open the file Table_Functions_7.pbix..

ALL and Filter

To use ALL() and FILTER() together:

1. Click on the Modeling tab in the Ribbon.
2. Click on New Table.
3. Type the following expression in the formula bar and press Enter.

```
Mixing table functions - FILTER AND ALL (CT) =
FILTER (
    ALL ( 'Product'[Category], 'Product'[Product], 'Product'[List Price] ),
    'Product'[Category] = "Accessories"
        && 'Product'[List Price] > 30
)
```

In this Expression:

- *A virtual table ignoring any incoming filters from Category, Product and List Price is calculated.*
- *FILTER() filters this virtual table and only the products from the category of "Accessories" that have the list price greater than 30 are filtered.*

4. Click on the Data tab in the left Ribbon.
5. Click on Mixing table functions – FILTER AND ALL (CT) in the Fields pan.
6. Click on the List Price Column.

   Column tools open.
7. Click on "$" in the formatting section.

Product	List Price	Category
Sport-100 Helmet, Black	$33.6442	Accessories
Sport-100 Helmet, Black	$34.99	Accessories
Headlights - Dual-Beam	$34.99	Accessories
Headlights - Weatherproof	$44.99	Accessories
Touring-Panniers, Large	$125	Accessories
Sport-100 Helmet, Red	$33.6442	Accessories
Sport-100 Helmet, Blue	$33.6442	Accessories
Sport-100 Helmet, Red	$34.99	Accessories
Sport-100 Helmet, Blue	$34.99	Accessories
Hydration Pack - 70 oz.	$54.99	Accessories
Hitch Rack - 4-Bike	$120	Accessories
All-Purpose Bike Stand	$159	Accessories
HL Mountain Tire	$35	Accessories
HL Road Tire	$32.6	Accessories

Notice that the resulting table contains 3 columns only because ALL() finds the combination of column values and FILTER() finds the subset of rows that belong to the category of Accessories and have list price greater than 30.

Summarize and Filter

Now we will use FILTER() and SUMMARIZE() to create a summary table containing products from the "Clothing "category" that have sales greater than $250,000. To use FILTER() and SUMMARIZE() together for this:

8. Click on the Modeling tab in the Ribbon.
9. Click on New Table.
10. Type the following expression in the formula bar and press Enter.

```
Mixing table functions - FILTER AND SUMMARIZE (CT) =
FILTER (
    SUMMARIZE (
        'Product',
        'Product'[Category],
        'Product'[Subcategory],
        "Sales", [Total Sales Amount (CM)]
    ),
    'Product'[Category] = "Clothing"
        && [Total Sales Amount (CM)] > 250000
)
```

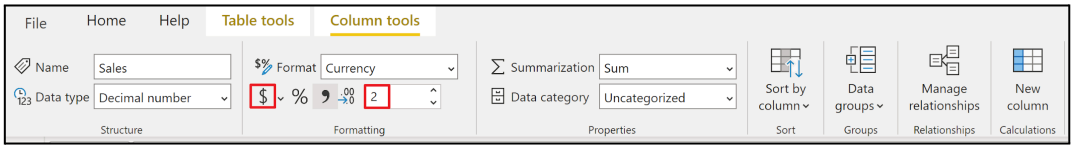
In this Expression:

- *The Product table is grouped by the product Category, Subcategory and Sales Amount in a virtual table.*
- *From this table, only the Products from the clothing category that have total sales greater than $250,000 are filtered*

11. Click on Sales column in the Mixing table functions – FILTER AND SUMMARIZE (CT) in the Fields pan.

Column tools box opens.

12. Click on the $ symbol and write 2 in the text box to set formatting.



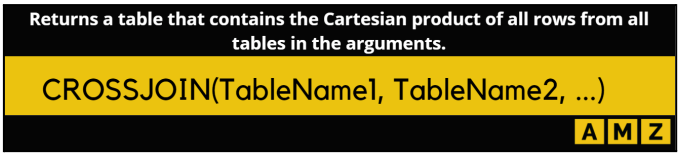
13. Click on the Data tab in the left Ribbon.
14. Click on Mixing table functions – FILTER AND SUMMARIZE (CT) in the Fields pan
15. Click on Sales Column.
16. Click on "$' symbol and type "2" in the text box in formatting section.

Category	Subcategory	Sales
Clothing	Jerseys	*$752,257.82*
Clothing	Shorts	*$413,522.68*
Clothing	Vests	*$259,488.51*

Notice that a summary table of products from clothing that contain the sales greater than $250,000 has been created.

The Cross Join

CROSSJOIN() allows you to build Cartesian product between two tables and brings all possible combinations from both tables.



Continue using the previous file for the exercise or open the file Table_Functions_8.pbix..

To see how CROSSJOIN() works:

1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
Testing Cross Join (CT) =
CROSSJOIN ( Customer, Product )
```

In this Expression:

- *A new table containing the Cartesian product of Customer and Product table is calculated.*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Testing Cross Join (CT) in the Fields pan.

CustomerKey	Customer ID	Customer	City	State-Province	Country-Region	Postal Code	Date of Birth	Number of Sales (CC)	Number of Products Purchases (CC)	Last Order Date (CC)	ProductKey	P
14562	AW00014562	Bryce Stewart	Concord	California	United States	94519	Saturday, July 17, 1954	2	2	Sunday, October 20, 2019	255	LL Road
14919	AW00014919	Erik Torres	Concord	California	United States	94519	Friday, November 30, 1951	2	2	Tuesday, October 1, 2019	255	LL Road
15177	AW00015177	Jorge Lu	Concord	California	United States	94519	Tuesday, September 16, 1969	2	2	Monday, April 27, 2020	255	LL Road
15319	AW00015319	Jade Sanchez	Concord	California	United States	94519	Tuesday, October 12, 1948	2	2	Tuesday, July 2, 2019	255	LL Road
15858	AW00015858	Jonathan Parker	Concord	California	United States	94519	Thursday, January 2, 1958	2	2	Monday, December 23, 2019	255	LL Road
17455	AW00017455	Dylan Hayes	Concord	California	United States	94519	Wednesday, April 6, 1994	2	2	Thursday, October 3, 2019	255	LL Road
18328	AW00018328	Alvin Rai	Concord	California	United States	94519	Wednesday, February 19, 1964	2	2	Wednesday, March 11, 2020	255	LL Road
17145	AW00017145	Sarah Bennett	Concord	California	United States	94519	Monday, November 8, 1982	2	2	Wednesday, April 15, 2020	255	LL Road
17447	AW00017447	Sydney Lewis	Concord	California	United States	94519	Wednesday, May 19, 1976	2	2	Monday, July 8, 2019	255	LL Road
17562	AW00017562	Zachary Sherma	Concord	California	United States	94519	Sunday, February 26, 1984	2	2	Friday, November 22, 2019	255	LL Road
21341	AW00021341	Louis Liu	Concord	California	United States	94519	Thursday, April 12, 2001	2	2	Friday, June 5, 2020	255	LL Road
21449	AW00021449	Devon Raheem	Concord	California	United States	94519	Sunday, November 27, 1977	2	2	Sunday, October 20, 2019	255	LL Road
21698	AW00021698	Tracy She	Concord	California	United States	94519	Thursday, August 26, 1948	2	2	Sunday, August 18, 2019	255	LL Road
21724	AW00021724	Mandy Yang	Concord	California	United States	94519	Tuesday, April 17, 1962	2	2	Friday, December 13, 2019	255	LL Road
21779	AW00021779	Maurice Black	Concord	California	United States	94519	Friday, April 5, 1974	2	2	Thursday, May 21, 2020	255	LL Road
13365	AW00013365	Luke Shan	Concord	California	United States	94519	Tuesday, May 20, 1969	2	2	Wednesday, February 12, 2020	255	LL Road
26171	AW00026171	Randall Ruiz	Concord	California	United States	94519	Sunday, September 21, 1997	2	2	Wednesday, November 6, 2019	255	LL Road
26237	AW00026237	Grace Martin	Concord	California	United States	94519	Sunday, January 28, 2001	2	2	Friday, July 26, 2019	255	LL Road
26502	AW00026502	Taylor Brown	Concord	California	United States	94519	Friday, October 7, 1988	2	2	Wednesday, February 5, 2020	255	LL Road

6. Scroll down to the end of table.

TABLE: Testing Cross Join (CT) (7,338,545 rows)

CROSSJOIN() with simple tables returns 7.4M rows and becomes an expensive operation to perform. We can use Values fn with Cross join to get all possible combinations of customer city and product subcategory.

CROSSJOIN() with VALUES()

To use VALUES() with CROSSJOIN() to get all possible combination of customer city and product subcategory:

7. Click on the **Testing Cross Join (CT)** tab in the **Fields**.

   Table tools open.

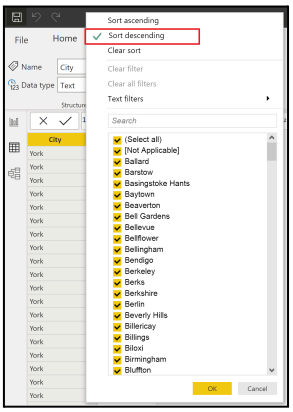
8. Type the following expression in the formula bar and press Enter.

```
Testing Cross Join (CT) =
CROSSJOIN ( VALUES ( Customer[City] ), VALUES ( 'Product'[Subcategory] ) )
```

In this Expression:

- *A new table containing the Cartesian product of distinct Customer cities and distinct Product Subcategories is calculated.*

9. Click on the **Data** tab in the left **Ribbon**.
10. Click on Testing Cross Join (CT) in the Fields pan.
11. Click on the drop down next to **City**.
12. Click on **Sort Descending**.



City	Subcategory
York	Cleaners
York	Bike Racks
York	Hydration Packs
York	Socks
York	Bottles and Cage
York	Tires and Tubes
York	Helmets
York	Locks
York	Panniers
York	Bib-Shorts

13. Scroll down to the end of table.

TABLE: Testing Cross Join (CT) (9,990 rows)

Notice the reduction in the number of rows.

Exploring Table Joins

Lets further combine this snippet with SUMMARIZE() to get something meaninful. It show us the combination of City and SubCategory where there is no sales.

Continue using the previous file for the exercise or open the file Table_Functions_9.pbix..

To see how to use SUMMARIZE() with CROSSJOIN() and VALUES():

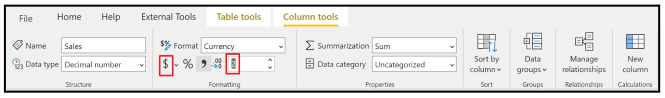
1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
City and Category with No Sales (CT) =
SUMMARIZE (
    CROSSJOIN ( VALUES ( Customer[City] ), VALUES ( 'Product'[Subcategory] ) ),
    Customer[City],
    'Product'[Subcategory],
    "Sales", [Total Sales Amount (CM)]
)
```

In this Expression:

- *A new virtual table containing the Cartesian product of distinct Customer cities and distinct Product Subcategories is calculated.*
- *This table is grouped by the product Subcategory and Sales Amount in a new virtual table.*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on City and Category with No Sales (CT) in the Fields pan.
6. Click on the **Sales** Column.
7. Click on "$" and type "2" in the formatting section.



8. Click on the drop down in City column.
9. Click on **sort descending**.

City	Subcategory	Sales
York	Helmets	$1,644.53
York	Mountain Bikes	$77,940.55
York	Jerseys	$1,185.77
York	Touring Bikes	$36,753.12
York	Caps	$206.77
York	Fenders	$43.96
York	Gloves	$244.9
York	Cleaners	$39.75
York	Hydration Packs	$274.95
York	Shorts	$209.97
York	Vests	$190.5
York	Socks	$35.96
York	Bike Racks	
York	Bike Stands	
York	Locks	
York	Bib-Shorts	
York	Tights	
York	Bottles and Cages	$490.3
York	Road Bikes	$105,757.82
York	Tires and Tubes	$1,377.15

Notice that no sales for Socks, Bike Racks, Locks, Bib-Shorts and Tights occur in York,

Testing table functions in measures

So far, we have tested the table functions in calculated tables. This is not the only way in which these functions can be used. In this exercise, you will learn how to use table functions in measures.

Continue using the previous file for the exercise or open the file Table_Functions_10.pbix..

To test the table functions in measures:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

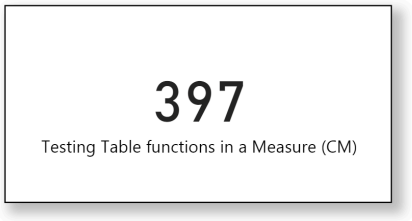
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Testing Table functions in a Measure (CM) =
COUNTROWS ( VALUES ( 'Product' ) )
```

In this Expression:

- *You have used VALUES() on the Product table which returns all the rows.*
- *VALUES() is encapsulated in COUNTROWS() function to return the number of rows.*

4. Click on the **card** visual in the **visualization pane**.
5. Drag and drop **Testing Table functions in a Measure (CM)** from the All Measures table to the Fields.



6. Click on the **matrix** visual in the **visualization pane**.
7. Drag and drop **Category** from the Products table to the rows.
8. Drag and drop **Testing Table functions in a Measure (CM)** from the All Measures table to the Values.

Category	Testing Table functions in a Measure (CM)
Accessories	35
Bikes	125
Clothing	48
Components	189
Total	**397**

Notice that as the filter context from the rows propagates, the values for the Testing Table functions in a Measure (CM) change. The total of the card visual and matrix is same.

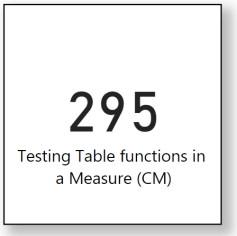
Let's now slightly alter the DAX expression and add column to the VALUES() parameter.

9. Click on **Testing Table functions in a Measure (CM)** in All Measures table. *Measure tools open.*
10. Replace Product table with Product Column so that your expression looks like the image below.

```
Testing Table functions in a Measure (CM) =
COUNTROWS ( VALUES ( 'Product'[Product] ) )
```

In this expression:

- *The VALUES() function finds the distinct values of products.*
- *The VALUES() forms a virtual table and COUNTROWS() counts the number of rows in the virtual table.*



Category	Testing Table functions in a Measure (CM)
Accessories	29
Bikes	97
Clothing	35
Components	134
Total	**295**

11. Click on **Testing Table functions in a Measure (CM)** in All Measures table. *Measure tools open.*
12. Replace with Subcategory column so that your expression looks like the image below.

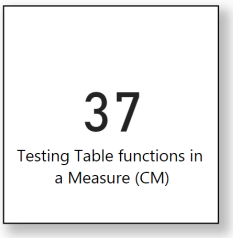
```
Testing Table functions in a Measure (CM) =
COUNTROWS ( VALUES ( 'Product'[Subcategory] ) )
```

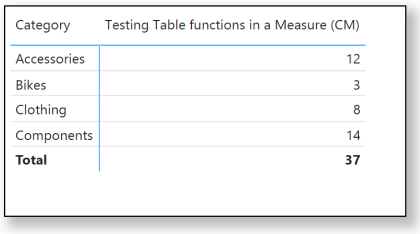
In this expression:

- *The VALUES() function finds the distinct values of Product Subcategory.*

- *The VALUES() forms a virtual table and COUNTROWS() counts the number of rows in the virtual table.*

Notice that only 37 categories exist. As the filter context comes into play in the Matrix, it is seen that Accessories have the greatest number of subcategories and Bikes have the least.





Category	Testing Table functions in a Measure (CM)
Accessories	12
Bikes	3
Clothing	8
Components	14
Total	**37**

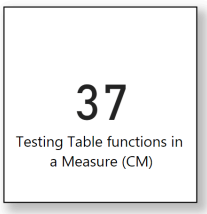
13. Click on **Testing Table functions in a Measure (CM)** in All Measures table. *Measure tools open.*
14. Replace the VALUES() function with ALL() so that your expression looks like the image below.

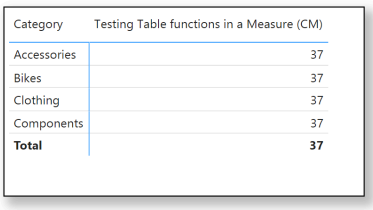
```
Testing Table functions in a Measure (CM) =
COUNTROWS ( ALL ( 'Product'[Subcategory] ) )
```

In this expression:

- *The ALL() function finds the distinct values of Products Subcategory.*
- *The ALL() forms a virtual table and COUNTROWS() counts the number of rows in the virtual table.*

Notice the change in values. ALL() ignores the filter context so the same value is shown for all categories and on the card. This is the total number of subcategories in the data.





Category	Testing Table functions in a Measure (CM)
Accessories	37
Bikes	37
Clothing	37
Components	37
Total	**37**

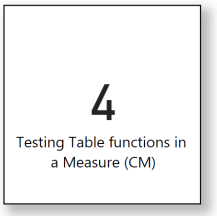
15. Click on **Testing Table functions in a Measure (CM)** in All Measures table. *Measure tools open.*
16. Replace the Subcategory column with the Category column so that your expression looks like the image below.

```
Testing Table functions in a Measure (CM) =
COUNTROWS ( ALL ( 'Product'[Category] ) )
```

In this expression:

- *The ALL() function finds the distinct values of Products Category.*
- *The ALL() forms a virtual table and COUNTROWS() counts the number of rows in the virtual table.*

Notice that only 4 categories exist. Filter context is ignored by ALL().

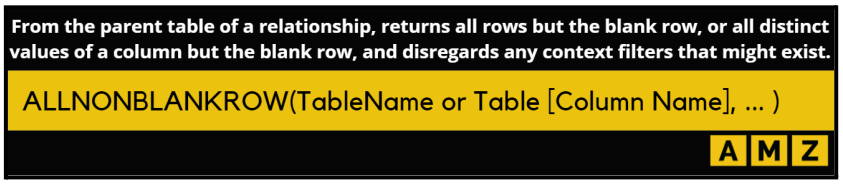


Category	Testing Table functions in a Measure (CM)
Accessories	4
Bikes	4
Clothing	4
Components	4
Total	**4**

Understanding the difference between VALUES(), DISTINCT(), ALL() and ALLNOBLANKROW()

So far, we have used various DAX functions that have a similar behavior. It is hard to understand the difference and usability of the DAX functions when you're practicing on large data sets like Adventure Works. So, we have developed a relatively simple data set with a very few rows.

Before we analyze the different DAX functions, let's see one last one.



For this exercise, open the file Table_Functions_11.pbix..

The data set consists of two table, connected by the Product ID column. The Product table contains 4 products (cellphones).

Product ID	Product Name
1	IPhone XS
2	Galaxy Note 10
3	P 30 Pro
4	7 Pro

The Sales table contains the quantity of different products sold. Notice that some Product IDs (5 and 6) exist in the Sales Quantity but these products do not exist in the Product table.

Product ID	Sales Quantity
1	2
1	4
2	3
3	7
4	1
5	4
5	6
6	1

To understand the behavior of different DAX functions for the same column in the data set:

1. Right click on the **Products** table.

   Since the data set is very small and this exercise is purely focused on understanding the DAX functions, we will not be creating a separate table for measures.
2. Click on **New Measure**.

Measure tools open.

3. Type the following DAX expression in the Formula bar and press Enter.

```
Count Rows using DISTINCT (CM) =
COUNTROWS ( DISTINCT ( 'Product' ) )
```

In this expression:

- *The DISTINCT() function returns the distinct products in a virtual table.*
- *COUNTROWS() counts the rows in the virtual table.*

4. Right click on the **Products** table.
5. Click on **New Measure**.

   Measure tools open.

6. Type the following DAX expression in the Formula bar and press Enter.

```
Count Rows using VALUES (CM) =
COUNTROWS ( VALUES ( 'Product' ) )
```

In this expression:

- *The VALUES() function returns the distinct products in a virtual table.*
- *COUNTROWS() counts the rows in the virtual table.*

7. Right click on the **Products** table.
8. Click on **New Measure**.

   Measure tools open.

9. Type the following DAX expression in the Formula bar and press Enter.

```
Count Rows Using ALL (CM) =
COUNTROWS ( ALL ( 'Product' ) )
```

In this expression:

- *The ALL() function returns the distinct products in a virtual table.*
- *COUNTROWS() counts the rows in the virtual table.*

10. Right click on the **Products** table.
11. Click on **New Measure**.

    Measure tools open.

12. Type the following DAX expression in the Formula bar and press Enter.

```
Count Rows using ALLNONBLANK (CM) =
COUNTROWS ( ALLNOBLANKROW ( 'Product' ) )
```

In this expression:

- *The ALLNONBLANK() function returns the distinct products in a virtual table.*
- *COUNTROWS() counts the rows in the virtual table.*

13. Click on the **matrix** visual in the **visualization pane**.
14. Drag and drop **Product** from the Products table to the rows.
15. Drag and drop **Count Rows using DISTINCT (CM)** from the Products table to the Values.
16. Drag and drop **Count Rows using VALUES (CM)** from the Products table to the Values.
17. Drag and drop **Count Rows using ALL (CM)** from the Products table to the Values.
18. Drag and drop **Count Rows using ALLNONBLANK** (CM) from the Products table to the Values.

Product Name	Count Rows using DISTINCT (CM)	Count Rows using VALUES (CM)	Count Rows Using ALL (CM)	Count Rows using ALLNONBLANK (CM)
P 30 Pro	1	1	5	4
IPhone XS	1	1	5	4
Galaxy Note 10	1	1	5	4
7 Pro	1	1	5	4
		1	5	4
Total	**4**	**5**	**5**	**4**

Now that you have used all the different functions in separate measures, lets analyze the results.

- ✔ The blank row in the Product Name indicates the invalid relationship created by the Product ID 5 and 6.
- ✔ **Count Rows using DISTINCT (CM)** counts the Distinct products and shows the results based on the incoming filter context.
- ✔ Another important consideration at this point is that DISTINCT() does not count the blank value and the total distinct products is 4.
- ✔ **Count Rows using VALUES (CM)** counts the Distinct products and shows the results based on the incoming filter context.
- ✔ VALUES() counts the blank value and the total distinct products is 5.
- ✔ **Count Rows using ALL (CM)** counts the Distinct products and shows the results ignoring the incoming filter context.

- ✔ ALL() counts the blank value and the total distinct products is 5 (Which is same for all rows and totals).
- ✔ **Count Rows using ALLNONBLANKROW (CM)** counts the Distinct products and shows the results ignoring the incoming filter context.
- ✔ ALLNONBLANK() does not count the blank value and the total distinct products is 4 (Which is same for all rows and totals).

Chapter 8: Variables and Comments in DAX

What will you learn in this chapter?

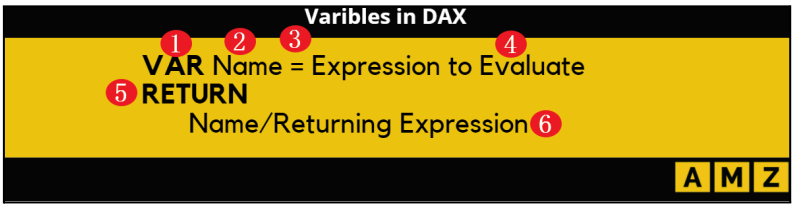
- ✔ Creating Variables
- ✔ Features of a Variable
- ✔ Conditional Computations using Variables
- ✔ Location of Variable Evaluation
- ✔ Increasing Code Readability
- ✔ Writing single and multi-line comments

Creating Variables

Variables are used in DAX to optimize the DAX code. The DAX engine evaluates the variables only once, making the code faster and more readable. Using variables in DAX helps you:

- ✔ Improve code readability.
- ✔ Improves code performance.
- ✔ Reduces complexity.
- ✔ Simplifies code.

Let's look at the structure of the code.



1. VAR key word is used to define the variable.
2. Name refers to the name of the variable.

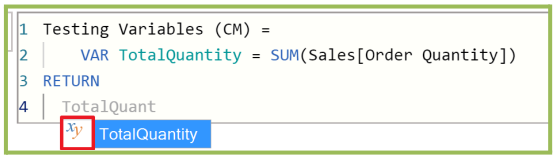
   Please note that you can't use space in Variable names.
3. The equal sign links variable name to the value.
4. The expression that is evaluated and the resultant value is stored in variable name.
5. RETURN is the key word for returning the value.
6. The returned value of variable or results of evaluated expression.

Let's see how a variable works:

Open the file Variables_and_Comments_In_DAX_1.pbix..

To see how variables work:

1. Right click on the All Measures table.
2. Click on New Measure.
3. Type the following expression in the formula bar.



Notice that the intellesence detects the variables and identifies them by the 'xy' symbol.

Features of a Variable

Let's look at some of the features of a variable:

- ✔ Variables always have 'VAR' and 'RETURN' keyword.
- ✔ Variables can be used for scaler values or for tables.
- ✔ Variables can be defined by other variables.
- ✔ Variables can be defined anywhere in the DAX expression.
- ✔ They are evaluated only once.
- ✔ The variables are executed in the initial filter and row contexts.
- ✔ Once the variable's value is evaluated their value does not change. This means they act like constants in the regular programming language.

Conditional Computations using Variables

Conditional computations can be faster using the variables because the variables are evaluated only once. Variables are very useful to avoid repeating subexpressions in the code.

Continue using the previous file for the exercise or open the file Variables_and_Comments_In_DAX_2.pbix..

To understand how variables are used to avoid repeating subexpressions:

1. Right click on the All Measures table.
2. Click on New Measure.

Measure tools open.

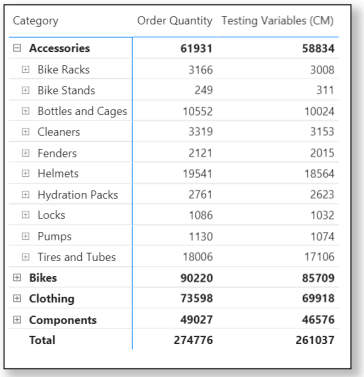
3. Type the following expression in the formula bar and press Enter.

```
Testing Variables (CM) =
VAR TotalQuantity =
    SUM ( Sales[Order Quantity] )
RETURN
    IF ( TotalQuantity > 1000, TotalQuantity * 0.95, TotalQuantity * 1.25 )
```

In this Expression:

- *The TotalQuantity variable stores the sum of all the values in the order quantity column.*
- *The value of the variable is evaluated and if it is greater than 1000, this value is multiplied by 0.95 and returned otherwise the value in the variable is multiplied by 1.25 and returned.*

4. Click on the **matrix** visual in the **visualization pane**. Drag and drop **Category, Subcategory** and **Model** from the Products table to the rows.
5. Drag and drop **Order Quantity** from the Sales table to the Values. Drag and drop **Testing Variables (CM)** from the All Measures table to the Values.



Category	Order Quantity	Testing Variables (CM)
Accessories	**61931**	**58834**
Bike Racks	3166	3008
Bike Stands	249	311
Bottles and Cages	10552	10024
Cleaners	3319	3153
Fenders	2121	2015
Helmets	19541	18564
Hydration Packs	2761	2623
Locks	1086	1032
Pumps	1130	1074
Tires and Tubes	18006	17106
Bikes	**90220**	**85709**
Clothing	**73598**	**69918**
Components	**49027**	**46576**
Total	**274776**	**261037**

Notice that the value for order quantity is evaluated only once and that value is used for further calculations in IF(). If the variables were not used here, the same expression needed to be evaluated multiple times.

Location of Variable Evaluation

Variables are computed in the evaluation where they are defined and not where they are used. CALCULATE() cannot be used to modify the value of a variable.

Continue using the previous file for the exercise or open the file Variables_and_Comments_In_DAX_3.pbix..

To see how important, it is to understand the location of variable evaluation:

1. Right click on the All Measures table.
2. Click on New Measure.

Measure tools open.

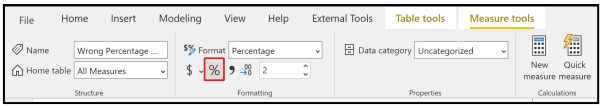
3. Type the following expression in the formula bar and press Enter.

```
Wrong Percentage of Sales (CM) =
VAR TotalSales =
    SUM ( Sales[Sales Amount] )
RETURN
    DIVIDE ( TotalSales, CALCULATE ( TotalSales, ALL ( Sales ) ) )
```

In this expression:

- *The total Sales Amount is stored in the variable TotalSales.*
- *This value is then divided by itself ignoring any incoming filter context from Sales.*

4. Click on "%" in the formatting section.



5. Click on the matrix visual in the visualization pane.
6. Drag and drop Category from the Products table to the rows.
7. Drag and drop Sales Amount from the Sales table to the Values.
8. Drag and drop Wrong Percentage of Sales (CM) from the All Measures table to the Values.

Category	Sales Amount	Wrong Percentage of Sales (CM)
Accessories	$1,272,059.71	100.00 %
Bikes	$94,620,526.59	100.00 %
Clothing	$2,117,613.52	100.00 %
Components	$11,799,074.18	100.00 %
Total	**$109,809,274.00**	**100.00 %**

Notice the resultant value. This value is same and 100% because the sales amount value is evaluated once and divided by itself to find the percentage.

The Sales Amount Column from the Sales Table has been formatted using the Formatting options in Power BI.

Increasing Code Readability

In this exercise, you will learn how to use variables to increase code readability and debugging.

Continue using the previous file for the exercise or open the file Variables_and_Comments_In_DAX_4.pbix..

Improving code Readability:

To see how variables can be used to increase readability:

1. Right click on the All Measures table.
2. Click on New Measure.

Measure tools open.

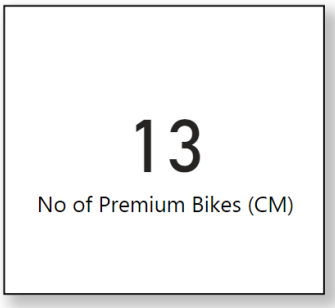
3. Type the following expression in the formula bar and press Enter.

```
No of Premium Bikes (CM) =
COUNTROWS (
    FILTER (
        'Product',
        'Product'[Category] = "Bikes"
            && 'Product'[List Price] > 2500
    )
)
```

In this expression:

- *Only the Products in the Product table that belong to the category of "Bikes" and have a List Price greater than 2500 are filtered.*

4. Click on the Card visual in the Visualization Pane.
5. Drag and drop No of Premium Bikes (CM) from All Measures table to the Fields.



Notice that we have 13 premium bikes in our database. This result is simple but the code consists of nested DAX functions which may not be that easy to understand and comprehend..

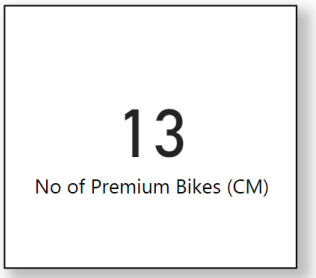
Let's get the same results by breaking down the code in smaller steps and increasing code readability by assigning variable names at different steps of the calculation.

6. Right click on No of Premium Bikes (CM) in the All Measures table.
   Measure tools open.
7. Erase the existing code.
8. Type the following code in the formula bar and press Enter.

```
No of Premium Bikes (CM) =
VAR PremiumBikeListPrice = 3000
VAR BikesCategoryProducts =
    FILTER ( 'Product', 'Product'[Category] = "Bikes" )
VAR PremiumBikeCategoryProducts =
    FILTER ( BikesCategoryProducts, 'Product'[List Price] > PremiumBikeListPrice )
RETURN
    COUNTROWS ( PremiumBikeCategoryProducts )
```

In the Expression:

- *First of all, the List Price for the Premium bikes is defined and the value is stored in a variable PremiumBikesListPrice.*
- *After this, a virtual table of products is created where the product category is "Bikes". This variable is stored in a variable name BikesCategoryProducts.*
- *As the third step, a table of the products from the BikesCategoryProducts that have the List Price greater than PremiumBikesListPrice is evaluated and stored in PremiumBikeCategoryProducts.*
- *Finally, the total rows in the PremiumBikeCategoryProducts are evaluated.*



Notice the value in the card visual is still the same. However, the logic behind the calculation is much easier to understand.

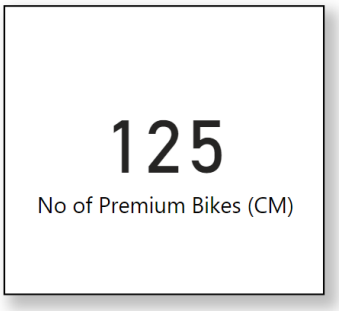
Variables for debugging:

You can also change the variables after returning. This helps to debug code.

9. Right click on No of Premium Bikes (CM) in the All Measures table.
   Measure tools open.
10. Replace the PremiumBikeCategoryProducts in the argument of COUNTROWS() with BikesCategoryProducts.

```
No of Premium Bikes (CM) =
VAR PremiumBikeListPrice = 3000
VAR BikesCategoryProducts =
    FILTER ( 'Product', 'Product'[Category] = "Bikes" )
VAR PremiumBikeCategoryProducts =
    FILTER ( BikesCategoryProducts, 'Product'[List Price] > PremiumBikeListPrice )
RETURN
    COUNTROWS ( BikesCategoryProducts )
```

Notice the change in the value of card visual.



Writing single and multi-line comments

Comments are used in DAX to document the code. They can also be used for debugging. Comments are not evaluated by DAX engine. Therefore, anything written in single and multi-line comments do not generate errors.

Continue using the previous file for the exercise or open the file Variables_and_Comments_In_DAX_5.pbix..

Single line comments:

Single line comments can be written on a single line only. They are suitable for short description and can be written anywhere in the code. Single line comments begin with "//" and the end of line is an automatic indication for the ending of the comment. We can also use double dash "--" to add single line comments. To see how we can write comments:

1. Click on the Wrong Percentage of Sales (CM) in the All Measures table.

   Measure tools open.

2. Type the comment at the end.

```
Wrong Percentage of Sales (CM) =
VAR TotalSales =
    SUM ( Sales[Sales Amount] )
RETURN
    DIVIDE ( TotalSales, CALCULATE ( TotalSales, ALL ( Sales ) ) )
// This measure is intentially created to emphasize the importance of location variable evaluation.
```

Notice that adding comment did not affect the evaluation and no errors are generated.

Multi line comments

Multi line comments can be written on multiple lines. These comments start with "/*" and end with "*/". Such comments can often be used for creating a playground to write complex nested codes without the issue of generating errors as shown in the image below.

```
Complex Logic (CM) =
/* SUMX(Customers,
                COUNTROWS (
                    FILTER (Sales,
                    Sales[CustomerKey]<152 && Sales[ProductKey]="352"*/
```

Chapter 9: Creating Date Table

What will you learn in this chapter?

✔ Creating & Optimizing Calendar Table using DAX functions

✔ Adding Date Columns

✔ Marking as Date table

✔ Working with Fiscal Years

✔ Finding Weekday

✔ Creating Relationship

✔ Set sorting options

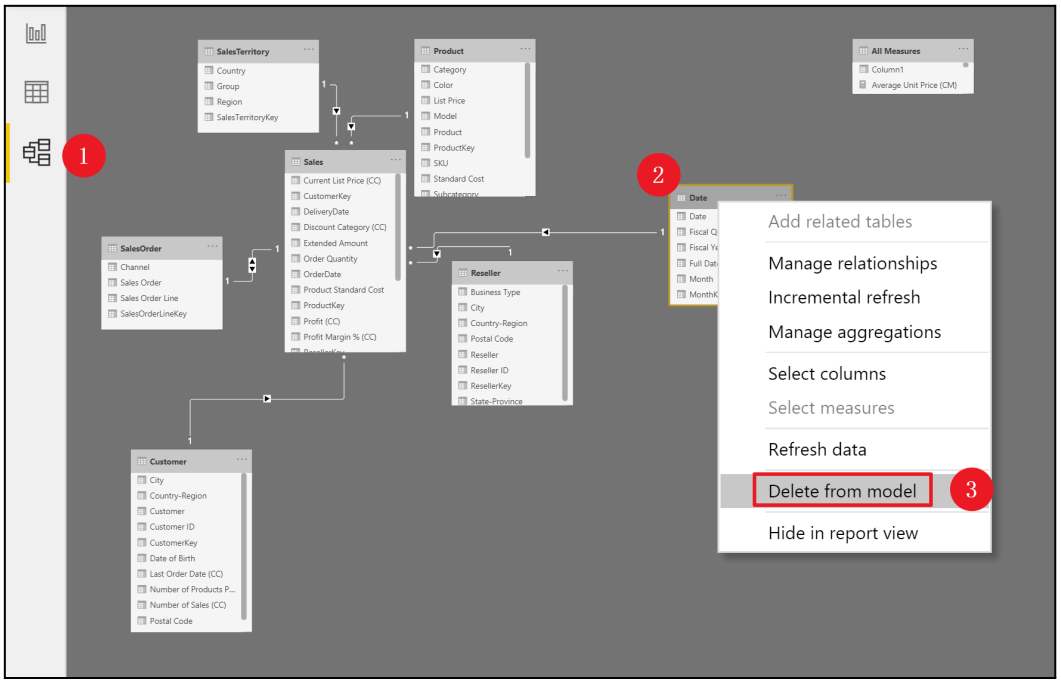
Creating & Optimizing Calendar Table using DAX functions

Date is an important dimension table in any data model. Date table can be generated in Power BI using either the **CALENDER()** function or the **CALENDERAUTO()** function. Both of these functions generate a single date column table.

Continue using the previous file for the exercise or open the file Creating_Date_Table_1.pbix..

Before you start building the calendar table, lets delete the existing date table and create a new one from scratch. To delete date table:

1. Click on the **Model** tab in the left ribbon.
2. Right click on the **Date** table.
3. Click on **Delete from Model**.



A pop-up menu appears.

4. Click **Delete.**

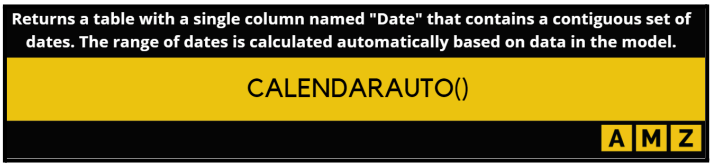
Now the existing date table has been removed from the data model. Lets create a new date table using the DAX functions.

Please note that for this chapter the naming convention for calculated columns, measures and tables, i.e., CC,CM and CT have been ignored because Date table is an essential dimension table reconstructed.

Using CALENDERAUTO():

CALENDERAUTO() automatically detects the first date and the last date in any date field in the model. CALNDERAUTO() also extends the first date to the 1st of January and the

last date to 31st December for that year. It has an optional parameter for fiscal year end month.



To use CALENDERAUTO() for calendar table:

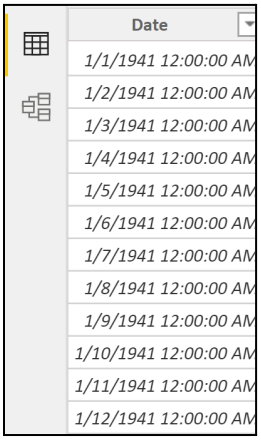
5. Click on the **Modeling** tab in the **Ribbon**.
6. Click on **New Table**.
7. Type the following expression in the formula bar and press Enter.

```
Date =
CALENDARAUTO ()
```

In this expression:

- *A Date table using the CALENDARAUTO() function is created.*

8. Click on the **Data** tab in the left **Ribbon**.
9. Click on Date table in Fields pan.

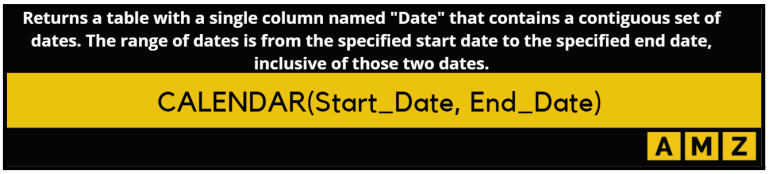


Notice that the dates present in the Date table start from 1941. This happens because a customer in the Customer table has the birth date in 1941 and CALENDERAUTO() scans the entire data model to find the first date and last date. This however is not required and generates unnecessary rows in date table.

10. Repeat the steps 1-3 to delete the Date table from the model.

Using CALENDER():

Using CALENDER() function, you can set the starting and the ending dates of the date table in the data model.



11. Click on the **Modeling** tab in the **Ribbon**.
12. Click on **New Table**.
13. Type the following expression in the formula bar and press Enter.

```
Date =
CALENDAR ( DATE ( 2017, 1, 1 ), DATE ( 2020, 12, 31 ) )
```

In the above DAX expression:

- *1/1/2017 has been provided as the starting date and 2020/12/31 has been provided as the ending date for the calendar.*

14. Click on the **Data** tab in the left **Ribbon**.
15. Click on Date in Fields pan.

Date
1/1/2017 12:00:00 AM
1/2/2017 12:00:00 AM
1/3/2017 12:00:00 AM
1/4/2017 12:00:00 AM
1/5/2017 12:00:00 AM
1/6/2017 12:00:00 AM
1/7/2017 12:00:00 AM

You have used the CALENDER() option and hard coded the starting and ending date but by using this, but you will have to manage this table on on-going basis. So let's create a better self-updating Calendar table using the variables.

16. Click on the Date table in the Fields.

    Table tools open.

17. Type the following DAX expression.

```
Date =
VAR FirstSalesDate =
    MIN ( Sales[OrderDate] )
VAR LastSalesDate =
    MAX ( Sales[OrderDate] )
RETURN
    CALENDAR ( FirstSalesDate, LastSalesDate )
```

In this expression,

- *FirstSalesDate variable stores the value of minimum order date in the Sales table.*
- *Similarly, maximum order date is stored in the LastSalesDate variable.*
- *Finally, these two variables have been used as arguments to in the CALENDER() function to set the range.*

18. Click on the **Data** tab in the left **Ribbon**.
19. Click on Date in Fields pan.

Date
7/1/2017 12:00:00 AM
7/2/2017 12:00:00 AM
7/3/2017 12:00:00 AM
7/4/2017 12:00:00 AM
7/5/2017 12:00:00 AM
7/6/2017 12:00:00 AM

Notice that this trick can be used to create a table but it may not provide full calendar years.

20. Click on the Date table in the Fields.

    Table tools open.

21. Type the following DAX expression.

```
Date =
VAR FirstSalesDate =
    MIN ( Sales[OrderDate] )
VAR FirstSalesYear =
    YEAR ( FirstSalesDate )
VAR LastSalesDate =
    MAX ( Sales[OrderDate] )
VAR LastSalesYear =
    YEAR ( LastSalesDate )
RETURN
    CALENDAR ( DATE ( FirstSalesYear, 1, 1 ), DATE ( LastSalesYear, 12, 31 ) )
```

In the above expression,

- *The year of the FirstSalesDate and the LastSalesDate has been calculated.*
- *The years are passed as arguments to the DATE() function.*
- *The CALENDER() table then uses these results to set the first date of Calendar to the 1st January of the year of first sales and the last date as 31st December of the year of the last sales.*

22. Click on the **Data** tab in the left **Ribbon**.
23. Click on Date in Fields pan.

Date
1/1/2017 12:00:00 AM
1/2/2017 12:00:00 AM
1/3/2017 12:00:00 AM
1/4/2017 12:00:00 AM
1/5/2017 12:00:00 AM
1/6/2017 12:00:00 AM
1/7/2017 12:00:00 AM
1/8/2017 12:00:00 AM

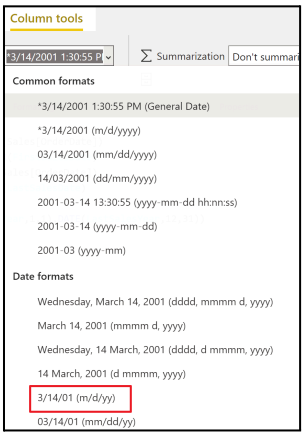
Although you have found your way to a self-updating automatic date table, you will still need to format the date to make it more readable.

24. Click on the **Date** column in the Date Table.

    Column tools open.

25. Click on drop-down next to **Format** menu.

26. Click on m/d/yy.



Marking as Date Table

Power BI is a tabular model and like other tabular models it requires marking the date table as date table to get appropriate results with time intelligence calculations.

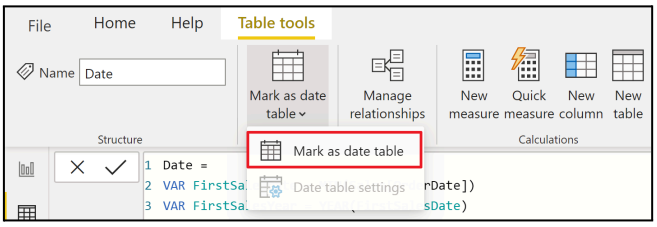
Continue using the previous file for the exercise or open the file Creating_Date_Table_2.pbix..

To mark as date table:

1. Click on **Date** table in the **Fields.**

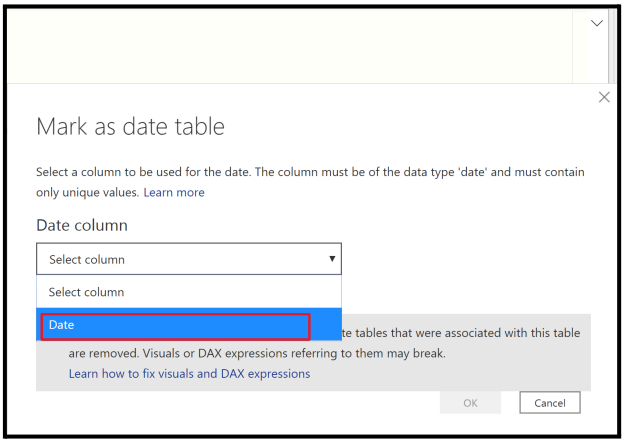
   Table tools open.

2. Click on Mark as date table > Mark as date table.



A pop up menu appears.

3. Click on the drop down menu next to **Date** column.
4. Click on **Date**.
5. Click ok.



Adding Date Columns

Now that you have created a data table, lets add columns to the table using DAX functions.

Continue using the previous file for the exercise or open the file Creating_Date_Table_3.pbix..

Creating YEAR() Column:

The YEAR() function is used to extract the value of year from the Date field.

To create a Year column:

1. Right click on the **Date** table in the Fields.
2. Click on **New Column**.

   Column tools open.
3. Type the following DAX expression in the Formula bar and press Enter.

```
Year =
YEAR ( 'Date'[Date] )
```

In this expression:

- *YEAR() is used to fetch the Year value from the Date column.*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Date in Fields pan and you will find Year column ..

Date	Year
1/1/17	2017
1/2/17	2017
1/3/17	2017
1/4/17	2017
1/5/17	2017
1/6/17	2017

Creating MONTH() Column:

The MONTH() function is used to extract the value of month from the Date field.

To create a Month column:

6. Right click on the **Date** table in the Fields.
7. Click on **New Column**.

Column tools open.

8. Type the following DAX expression in the Formula bar and press Enter.

```
MonthNo =
MONTH ( 'Date'[Date] )
```

In this expression:

- *MONTH() is used to fetch the Month value from the Date column.*

9. Click on the **Data** tab in the left **Ribbon**.
10. Click on Date in Fields pan and you will find MonthNo column.

Date	Year	MonthNo
1/1/17	2017	1
1/2/17	2017	1
1/3/17	2017	1
1/4/17	2017	1
1/5/17	2017	1
1/6/17	2017	1

Creating QUARTER() Column:

The QUARTER() function is used to extract the value of Quarter from the Date field.

To create a Quarter Number column:

11. Right click on the **Date** table in the Fields.
12. Click on **New Column**.

Column tools open.

13. Type the following DAX expression in the Formula bar and press Enter.

```
QtrNo =
QUARTER ( 'Date'[Date] )
```

In this expression:

- *QUARTER() is used to fetch the Quarter value from the Date column.*

14. Click on the **Data** tab in the left **Ribbon**.
15. Click on Date in Fields pan. and you will also find QtrNo column

Date	Year	MonthNo	QtrNo
1/1/17	2017	1	1
1/2/17	2017	1	1
1/3/17	2017	1	1
1/4/17	2017	1	1
1/5/17	2017	1	1
1/6/17	2017	1	1

Now let's create another column to store the quarter number with a meaningful name.

To create a Qtr column:

16. Right click on the **Date** table in the Fields.
17. Click on **New Column**.

    Column tools open.

18. Type the following DAX expression in the Formula bar and press Enter.

```
Qtr =
"Q" & QUARTER ( 'Date'[Date] )
```

In this expression:

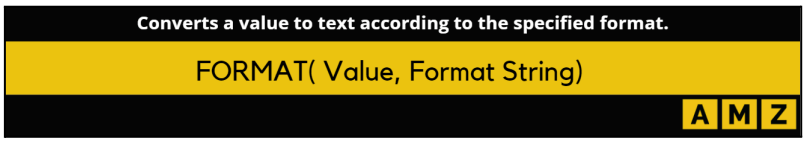
- *"Q" is concatenated with the value for the Quarter.*

19. Click on the **Data** tab in the left **Ribbon**.
20. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr
1/1/17	2017	1	1	Q1
1/2/17	2017	1	1	Q1
1/3/17	2017	1	1	Q1
1/4/17	2017	1	1	Q1
1/5/17	2017	1	1	Q1
1/6/17	2017	1	1	Q1

Formatting the Columns:

Now let's use the FORMAT() function to format the date field to a more readable form such as Month name, Full Month and Full Month No.



To create a Month Name column:

21. Right click on the **Date** table in the Fields.
22. Click on **New Column**.

    Column tools open.

23. Type the following DAX expression in the Formula bar and press Enter.

```
Month Name =
FORMAT ( 'Date'[Date], "mmm" )
```

In this expression:

- *The Date column is formatted to return the month name only.*

24. Click on the **Data** tab in the left **Ribbon**.
25. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name
1/1/17	2017	1	1	Q1	Jan
1/2/17	2017	1	1	Q1	Jan
1/3/17	2017	1	1	Q1	Jan
1/4/17	2017	1	1	Q1	Jan
1/5/17	2017	1	1	Q1	Jan
1/6/17	2017	1	1	Q1	Jan

To create a Full Month column in 'Year Month' format:

26. Right click on the **Date** table in the Fields.
27. Click on **New Column**.

    Column tools open.

28. Type the following DAX expression in the Formula bar and press Enter.

```
Full Month =
FORMAT ( 'Date'[Date], "yyyy mmm" )
```

In this expression:

- *The Date column is formatted to return the year and month name only.*

29. Click on the **Data** tab in the left **Ribbon**.
30. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month
1/1/17	2017	1	1	Q1	Jan	2017 Jan
1/2/17	2017	1	1	Q1	Jan	2017 Jan
1/3/17	2017	1	1	Q1	Jan	2017 Jan
1/4/17	2017	1	1	Q1	Jan	2017 Jan
1/5/17	2017	1	1	Q1	Jan	2017 Jan
1/6/17	2017	1	1	Q1	Jan	2017 Jan

To create a Full Month No column:

31. Right click on the **Date** table in the Fields.
32. Click on **New Column**.

    Column tools open.

33. Type the following DAX expression in the Formula bar and press Enter.

```
Full Month No =
FORMAT ( 'Date'[Date], "yyyymm" )
```

In this expression:

- *The Date column is formatted to return the year and month number only.*

34. Click on the **Data** tab in the left **Ribbon**.
35. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month	Full Month No
1/1/17	2017	1	1	Q1	Jan	2017 Jan	201701
1/2/17	2017	1	1	Q1	Jan	2017 Jan	201701
1/3/17	2017	1	1	Q1	Jan	2017 Jan	201701
1/4/17	2017	1	1	Q1	Jan	2017 Jan	201701
1/5/17	2017	1	1	Q1	Jan	2017 Jan	201701
1/6/17	2017	1	1	Q1	Jan	2017 Jan	201701

Working with Fiscal Years

A fiscal year or a financial year is a one-year period chosen by the company for its financial reporting. Until now, the date table does not have fiscal year and fiscal quarter. In this exercise, you will learn how to create fiscal year and quarters column in the same date table.

Continue using the previous file for the exercise or open the file Creating_Date_Table_4.pbix..

To create a Fiscal Year No column if fiscal year ends on June:

1. Right click on the **Date** table in the Fields.
2. Click on **New Column**.

   Column tools open.
3. Type the following DAX expression in the Formula bar.

```
Fiscal Year =
VAR CurrentYear = 'Date'[Year]
VAR CurrentMonth = 'Date'[MonthNo]
RETURN
    IF ( CurrentMonth < 7, "FY" & CurrentYear, "FY" & CurrentYear + 1 )
```

In the above expression:

- *The CurrentYear variable stores the value of year.*
- *The CurrentMonth variable stores the month number.*
- *If the value of CurrentMonth is less than 6 then this "FY" is concatenated with this value CurrentYear otherwise 1 is added to the value of CurrentYear before concatenation with "FY".*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month	Full Month No	Fiscal Year
1/1/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017
1/2/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017
1/3/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017
1/4/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017
1/5/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017
1/6/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017

Now let's create a Fiscal Quarter column.

To create a Fiscal Qtr column:

6. Right click on the **Date** table in the Fields.
7. Click on **New Column**.

   Column tools open.
8. Type the following DAX expression in the Formula bar and press Enter.

```
Fiscal Qtr =
VAR CurrentYear = 'Date'[Year]
VAR CurrentMonth = 'Date'[MonthNo]
RETURN
    IF (
        CurrentMonth <= 3,
        "FYQ" & 3,
        IF (
            CurrentMonth <= 6,
            "FYQ" & 4,
            IF ( CurrentMonth <= 9, "FYQ" & 1, IF ( CurrentMonth <= 12, "FYQ" & 2 ) )
        )
    )
```

In the above expression:

- *The CurrentYear variable stores the value of year.*
- *The CurrentMonth variable stores the month number.*
- *If the value of CurrentMonth is less than 3 then this "FYQ" is concatenated with this value otherwise CurrentMonth is tested to have a value less then 6.*
- *The same logic is repeated for all the months of the year to define Fiscal Quarters.*

9. Click on the **Data** tab in the left **Ribbon**.
10. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month	Full Month No	Fiscal Year	Fiscal Qtr
1/1/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3
1/2/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3
1/3/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3
1/4/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3
1/5/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3
1/6/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3

Notice that the fiscal year runs parallel to the calendar year.

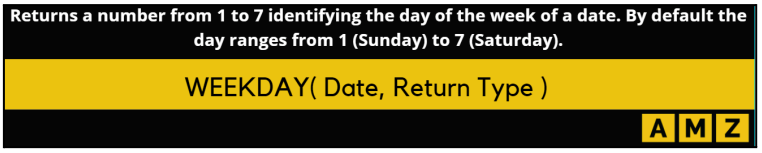
Finding Weekday

Finally, lets create columns to identify the weekday.

Continue using the previous file for the exercise or open the file Creating_Date_Table_5.pbix..

Calculating WEEDAY():

WEEKDAY() is a DAX function used to identify day of the week.



To create a WeekDayNo column:

1. Right click on the **Date** table in the Fields.
2. Click on **New Column**.

Column tools open.

3. Type the following DAX expression in the Formula bar and press Enter.

```
WeekDayNo =
WEEKDAY ( 'Date'[Date] )
```

In this expression:

- *WEEKDAY() is used to fetch the Week day value from the Date column.*

4. Click on the **Data** tab in the left **Ribbon**.
5. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month	Full Month No	Fiscal Year	Fiscal Qtr	WeekDayNo
1/1/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	1
1/2/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	2
1/3/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	3
1/4/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	4
1/5/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	5
1/6/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	6

Now that we have week day, lets create a column to find the day of week.

To create a WeekDay column:

6. Right click on the **Date** table in the Fields.
7. Click on **New Column**.

Column tools open.

8. Type the following DAX expression in the Formula bar and press Enter.

```
WeekDay =
FORMAT ( 'Date'[Date], "ddd" )
```

In this expression:

- *The Date column is formatted to return the day name only.*

9. Click on the **Data** tab in the left **Ribbon**.
10. Click on Date in Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month	Full Month No	Fiscal Year	Fiscal Qtr	WeekDayNo	WeekDay
1/1/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	1	Sun
1/2/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	2	Mon
1/3/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	3	Tue
1/4/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	4	Wed
1/5/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	5	Thu
1/6/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	6	Fri

Now that WeekDay is found, lets create a column to identify working day and weekends.

To create a WorkingDays column:

11. Right click on the **Date** table in the Fields.
12. Click on **New Column**.

Column tools open.

13. Type the following DAX expression in the Formula bar and press Enter.

```
WorkingDays =
IF ( 'Date'[WeekDayNo] IN { 1, 7 }, "Weekend", "Weekday" )
```

In this expression:

- *If the WeekDayNo is 1 or 7 then the current day is categorized as Weekend otherwise it is a Weekday.*

14. Click on the **Data** tab in the left **Ribbon**.
15. Click on Date in the Fields pan.

Date	Year	MonthNo	QtrNo	Qtr	Month Name	Full Month	Full Month No	Fiscal Year	Fiscal Qtr	WeekDayNo	WeekDay	WorkingDays
1/1/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	1	Sun	Weekend
1/2/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	2	Mon	Weekday
1/3/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	3	Tue	Weekday
1/4/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	4	Wed	Weekday
1/5/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	5	Thu	Weekday
1/6/17	2017	1	1	Q1	Jan	2017 Jan	201701	FY2017	FYQ3	6	Fri	Weekday

DAX Date Template:

For further learning, DAX Date Template is created by Sqlbi. This template saves the time for writing extensive codes.

To download DAX Date Template: https://www.sqlbi.com/tools/dax-date-template/

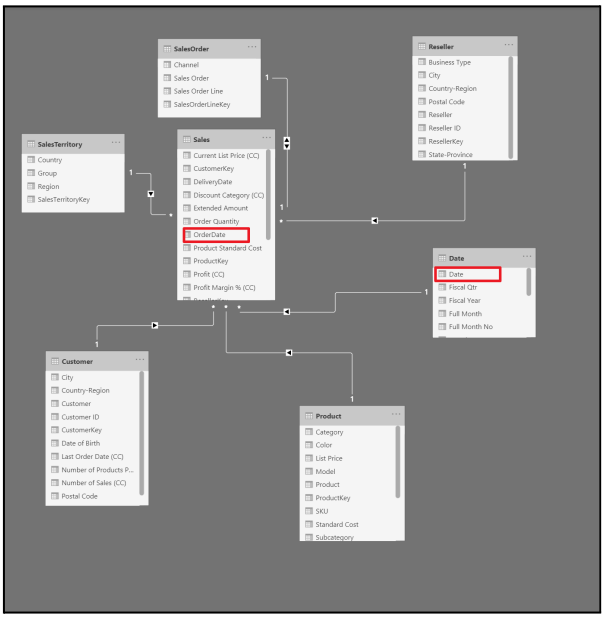
Creating Relationship

Relationships link the date table and are used to define the filter propagation. Without the relationship, the calculated date table cannot be used for any analysis.

Continue using the previous file for the exercise or open the file Creating_Date_Table_6.pbix..

To create relationship:

1. Click on the **Data** tab in the left ribbon.
2. Click on OrderDate in the Sales table.
3. Drag the cursor to Date column in the Date table.



When the relationship is created, a line connecting two tables is created.

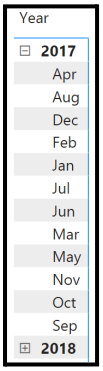
Set sorting options

Sorting of date columns by one another is important, otherwise the Power BI's engine sorts the months alphabetically.

Continue using the previous file for the exercise or open the file Creating_Date_Table_7.pbix..

To see and understand sorting.

1. Click on the **Matrix** visual in the **Visualization Pane**.
2. Drag and drop the **Year Column** from the Date table to the Rows.
3. Drag and drop the **Month Name Column** from the Date table to the Rows.

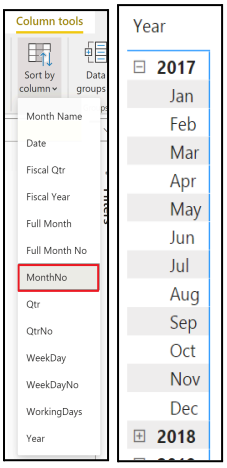


Notice that April occurs before January.

To sort the column:

4. Click on Month Name in the Date table.

   Column tools open.
5. Click on Sort by Column.
6. Click on MonthNo.



Notice the change in the Month names sorting in the Matrix.

Chapter 10: Basic Iterators

What will you learn in this chapter?

- ✔ Iterators vs. Aggregator Functions
- ✔ Using SUMX()
- ✔ Using AVERAGEX()
- ✔ Using MINX() and MAXX()
- ✔ Implicit CALCULATE() and Context Transition

Iterators vs. Aggregator Functions

Iterators are a class of functions identified by the “X” at the end. These functions are powerful because they evaluate the expression at every row of the table. Aggregators are sometimes known as syntax-sugared versions of iterators. Aggregators have shorter syntax but iterators can perform advanced computations, which are not possible with simple syntax.

Feature	**Iterators**	**Aggregator Functions**
Definition	Functions that perform operations row-by-row	Functions that perform operations on a set of values
Usage	Used for row context operations with functions like SUMX, AVERAGEX	Used to aggregate a column or table, e.g., SUM, AVERAGE
Syntax Example	SUMX(Table, Expression)	SUM(Column)
Performance	Can be less efficient for large datasets	Generally more efficient for simple aggregations
Context	Requires explicit row context	Operates in a filter context
Flexibility	Allows for complex calculations on each row	Limited to basic summarizations
Common Scenarios	Creating calculated columns, custom measures	Basic summarization in visuals
Result Type	Can return complex results or tables	Typically returns a single value

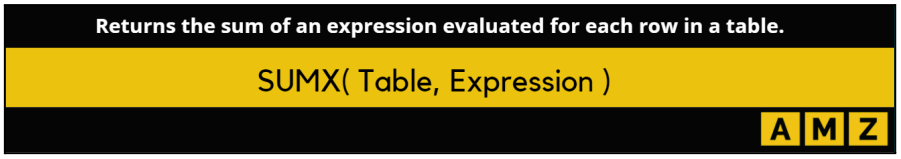
Using SUMX()

SUMX() iterates through every row of the table to evaluate an expression and then returns the aggregated result of all the rows.

Continue using the previous file for the exercise or open the file Basic_Iterators_1.pbix..

SUMX() has the following features:

- ✔ The table used in SUMX() can be either virtual or physical.
- ✔ The expression used for evaluation must always return a number.
- ✔ SUMX() ignores blank, logical and text values in a column.



Let's calculate the discount amount for each transaction in the Sales table.

To use SUMX() for calculating the discount amount:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

Measure tools open.

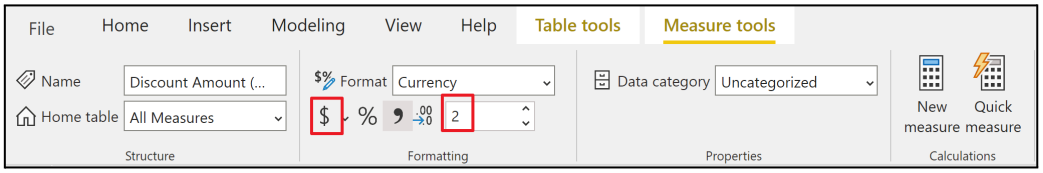
3. Type the following formula in the formula bar and press Enter.

```
Discount Amount (CM) =
SUMX ( Sales, Sales[Unit Price Discount Pct] * Sales[Extended Amount] )
```

In the above expression:

- *For each row in the Sales table, Unit Price Discount is multiplied by Extended Amount and the value is stored in a virtual table.*
- *All the values in the virtual table are aggregated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



Now let's see what happens if we try to calculate the discount amount using aggregators instead of iterators.

6. Right click on **All Measures** in the **Fields**.
7. Click on **New measure.**

   Measure tools open.
8. Type the following formula in the formula bar and press Enter.

```
Discount Amount Wrong (CM) =
SUM ( Sales[Unit Price Discount Pct] ) * SUM ( Sales[Extended Amount] )
```

In the above expression:

- *The arrogated value of "Unit Price Discount Pct" from the Sales table is multiplied by the aggregated value of "Extended Amount" from the Sales Table.*

9. Click on the "$" in the formatting section.
10. Type "2" in the text box.

11. Click on the **matrix** visual in the **visualization pane**.
12. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
13. Drag and drop **Discount Amount (CM)** and **Discount Amount Wrong(CM)** from the All Measures table to the Values

Category	Discount Amount (CM)	Discount Amount Wrong (CM)
⊞ Accessories	$6,688.03	$42,940,331.31
⊞ Bikes	$494,641.03	$25,102,795,191.34
⊞ Clothing	$20,964.54	$92,728,708.26
⊞ Components	$5,214.74	$23,490,534.87
Total	$527,508.34	$37,828,965,540.70

Notice the difference in values of both measures. Discount Amount (CM) evaluates the Discount Amount for each row and then sums the values for each category. Whereas, the Discount Amount Wrong (CM) sums the Unit Price Discount Pct and Extended Amount columns and then multiplies them.

Now lets use variables with SUMX() to calculate the discount amount for large sales.

14. Right click on **All Measures** in the **Fields**.
15. Click on **New measure.**

    Measure tools open.

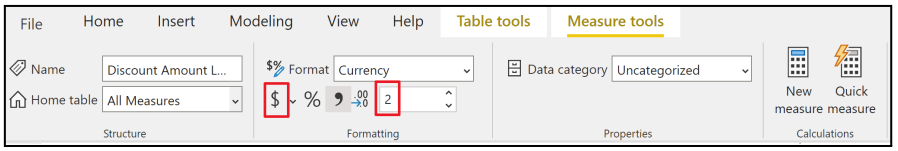
16. Type the following formula in the formula bar and press Enter.

```
Discount Amount Large Sales (CM) =
VAR LargeSalesTable =
    FILTER ( Sales, Sales[Order Quantity] > 10 )
RETURN
    SUMX (
        LargeSalesTable,
        Sales[Unit Price Discount Pct] * Sales[Extended Amount]
    )
```

In the above expression:

- *The variable LargeSalesTable stores a subset of rows from the Sales table where the order quantity is greater then 10.*
- *SUMX() then performs iteration over the LargeSalesTable.*

17. Click on the "$" in the formatting section.
18. Type "2" in the text box.



19. Drag and drop **Discount Amount Large Sales (CM)** from the All Measures table to the Values

Category	Discount Amount (CM)	Discount Amount Wrong (CM)	Discount Amount Large Sales (CM)
⊞ Accessories	$6,688.03	$42,940,331.31	$5,314.82
⊞ Bikes	$494,641.03	$25,102,795,191.34	$148,798.45
⊞ Clothing	$20,964.54	$92,728,708.26	$20,964.54
⊞ Components	$5,214.74	$23,490,534.87	$5,214.74
Total	$527,508.34	$37,828,965,540.70	$180,292.56

Notice the difference in values of discounted amounts using iterators and aggregators.

Using AVERAGEX()

AVERAGEX() works on the same principle as SUMX() and other iterators do. AVERAGEX() evaluates the expression at each row and then evaluates the athematic mean of the resulting set of values. In all other aspects, AVERAGEX() follows the same rules as AVERAGE(), i.e., you cannot use non-numeric or null cells.

Continue using the previous file for the exercise or open the file Basic_Iterators_2.pbix..

Let's use AVERAGEX() to find the average delivery days.

To use AVERAFEX():

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Average Delivery Days (CM) =
AVERAGEX ( Sales, Sales[DeliveryDate] - Sales[OrderDate] )
```

In the above expression:

- *For each row in the Sales table the OrderDate is subtracted from the DeliveryDate and the value is stored in a virtual table.*
- *The average over all values in this virtual table is calculated.*

4. Click on the **matrix** visual in the **visualization pane**.
5. Drag and drop **Year** from the Date table to the rows.
6. Drag and drop **Average Delivery Days (CM)** from the All Measures table to the Values

Year	Average Delivery Days (CM)
2017	10.05
2018	9.28
2019	8.21
2020	8.26
Total	**8.47**

Notice that the average delivery days reduced from almost 10 days in 2017 to 8 days in 2020. Let's use the same measure in a different filter context.

7. Click on the **matrix** visual in the **visualization pane**.
8. Drag and drop **Country** from the SalesTerritory table to the rows.
9. Drag and drop **Average Delivery Days (CM)** from the All Measures table to the Values.

Country	Average Delivery Days (CM)
Australia	10.03
Canada	8.27
France	8.17
Germany	8.02
United Kingdom	8.13
United States	8.31
Total	**8.47**

Notice that the Average Delivery Days are slightly higher for Australia.

Let's finally calculate the Bonus Amounts associated with the Sales taking place on the Weekday.

10. Right click on **All Measures** in the **Fields**.
11. Click on **New measure.**

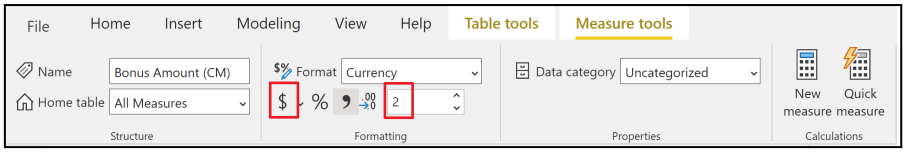
    Measure tools open.
12. Type the following formula in the formula bar and press Enter.

```
Bonus Amount (CM) =
SUMX (
    Sales,
    IF (
        RELATED ( 'Date'[WorkingDays] ) = "WeekDay",
        Sales[Sales Amount] * 0.01,
        Sales[Sales Amount] * 0.02
    )
)
```

In the above expression:

- *The day of the sales is logically tested.*
- *If the sales occur on weekday than the sales amount is multiplied by 1% and stored in a virtual table.*
- *If the sales occur on weekend than the sales amount is multiplied by 2% and stored in a virtual table.*
- *All the values in the virtual table are aggregated using SUMX().*

13. Click on the "$" in the formatting section.
14. Type "2" in the text box.



15. Click on the **matrix** visual in the **visualization pane**.
16. Drag and drop **Year** from the Date table to the rows.
17. Drag and drop **Sales Amount** from the Sales table to the Values.
18. Drag and drop **Average Delivery Days (CM)** and **Bonus Amount (CM)** from the All Measures table to the Values.

Year	Sales Amount	Average Delivery Days (CM)	Bonus Amount (CM)
2017	$11,928,556.40	10.05	$151,740.13
2018	$30,516,895.19	9.28	$393,031.80
2019	$42,895,108.14	8.21	$547,183.03
2020	$24,468,714.27	8.26	$313,361.18
Total	**$109,809,274.00**	**8.47**	**$1,405,316.13**

Notice that as the average delivery days decreased from 2017 to 2020, the Bonus Amount relatively increased.

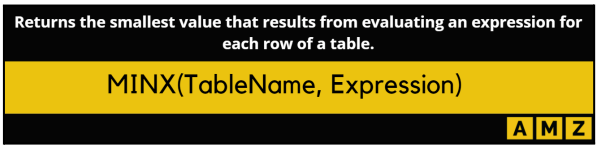
Using MINX() and MAXX()

MINX() returns the smallest value that results after the evaluation of the expression. Whereas, MAXX() returns the largest value that results after the evaluation of the expression. For both of these functions the blank values are skipped and TRUE/FALSE is not supported.

Continue using the previous file for the exercise or open the file Basic_Iterators_3.pbix..

Using MINX():

Let's use MINX() to find the minimum sales amount for each customer.



To use MINX() for this:

1. Right click on the **Customer** table in the **Fields**.
2. Click on **New Column**.

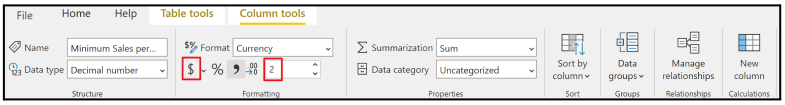
   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Minimum Sales per Customer (CC) =
MINX ( RELATEDTABLE ( Sales ), Sales[Sales Amount] )
```

In the above expression:

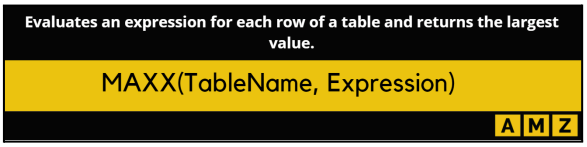
- *The RELATEDTABLE() stores the subset of rows in a virtual table.*
- *MINX() iterates over this table and finds the minimum value.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



Using MAXX():

Let's use MAXX() to find the maximum sales amount for each customer.



To use MAXX() for this:

6. Right click on the Customer table in the **Fields**.

7. Click on **New Column**.

   Column tools open.

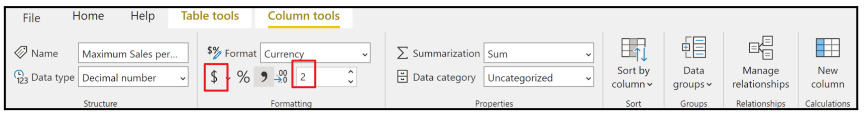
8. Type the following formula in the formula bar and press Enter.

```
Maximum Sales per Customer (CC) =
MAXX ( RELATEDTABLE ( Sales ), Sales[Sales Amount] )
```

In the above expression:

- *The RELATEDTABLE() stores the subset of rows in a virtual table.*
- *MANX() iterates over this table and finds the maximum value.*

9. Click on the "$" in the formatting section.
10. Type "2" in the text box.

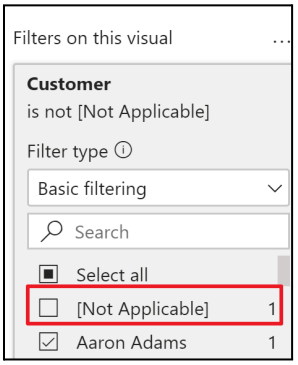


11. Click on the **matrix** visual in the **visualization pane**.
12. Drag and drop **Customer** from the Customer table to the rows.
13. Drag and drop **Minimum Sales per Customer (CC)** and **Maximum Sales per Customer (CC)** from the Customer table to the Values.

Customer	Minimum Sales per Customer (CC)	Maximum Sales per Customer (CC)
[Not Applicable]	$1.37	$27,893.62
Aaron Adams	$3.99	$53.99
Aaron Alexander	$69.99	$69.99
Aaron Allen	$3,399.99	$3,399.99
Aaron Baker	$49.99	$1,700.99
Aaron Bryant	$4.99	$49.99
Aaron Butler	$4.99	$9.99
Aaron Campbell	$34.99	$1,120.49
Aaron Carter	$4.99	$34.99
Aaron Chen	$4.99	$34.99

Notice the results have a [Not Applicable] column. This shows the values for the resellers.

14. In the Filters Pane, un check the box next to [Not Applicable].



Notice the difference between the minimum and maximum sales amount for the customer.

Customer	Minimum Sales per Customer (CC)	Maximum Sales per Customer (CC)
Aaron Adams	$3.99	$53.99
Aaron Alexander	$69.99	$69.99
Aaron Allen	$3,399.99	$3,399.99
Aaron Baker	$49.99	$1,700.99
Aaron Bryant	$4.99	$49.99
Aaron Butler	$4.99	$9.99
Aaron Campbell	$34.99	$1,120.49
Aaron Carter	$4.99	$34.99
Aaron Chen	$4.99	$34.99
Aaron Coleman	$4.99	$34.99
Aaron Collins	$34.99	$3,578.27

Implicit CALCULATE() and Context Transition

There is always an implicit CALCULATE() within measures and if a measure is used with an iterator, then Calculate performs context transition i.e., converting row context into filter context.

Continue using the previous file for the exercise or open the file Basic_Iterators_4.pbix..

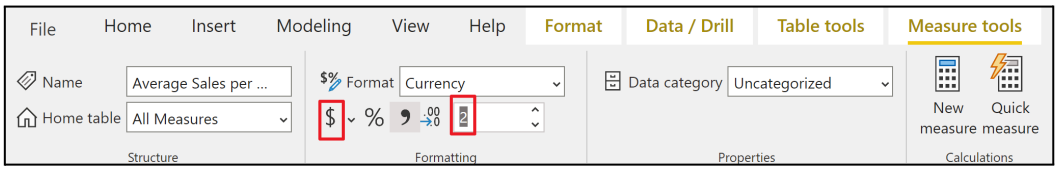
To see how implicit CALCULATE() within a measure works:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Average Sales per Product (CM) =
AVERAGEX ( 'Product', [Total Sales Amount (CM)] )
```

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Click on the **matrix** visual in the **visualization pane**.
7. Drag and drop **Year** from the Date table to the rows.
8. Drag and drop **Total Sales Amount (CM)** and **Average Sales Per Product (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Average Sales per Product (CM)
2017	$11,928,556.40	$198,809.27
2018	$30,516,895.19	$183,836.72
2019	$42,895,108.14	$152,110.31
2020	$24,468,714.27	$147,401.89
Total	**$109,809,274.00**	**$313,740.78**

Average Sales Per Product (CM) iterates over Product Table and because there is a measure within an iterator it performs context transition (due to implicit CALCULATE()) and gives us Average Sales Per Product.

9. Click on the **Total Sales Amount (CM)** in the All Measures table.

```
Total Sales Amount (CM) =
SUM ( Sales[Sales Amount] )
```

Notice that although [Total Sales Amount CM] measure is equals to SUM(Sales[Sales Amount]), if you replace the calculated measure with its underlying formula within Average Sales Per Product, context transition will not take place.

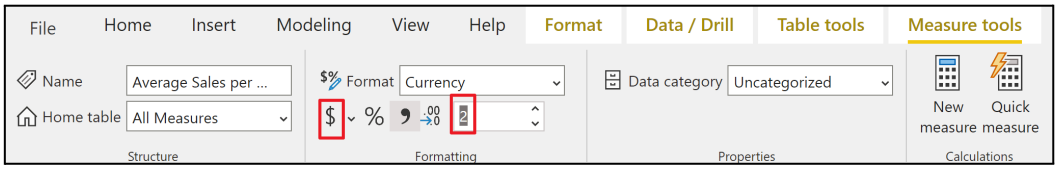
10. Right click on **All Measures** in the **Fields**.
11. Click on **New measure.**

    Measure tools open.

12. Type the following formula in the formula bar and press Enter.

```
Average Sales per Product without Context Transition (CM) =
AVERAGEX ( 'Product', SUM ( Sales[Sales Amount] ) )
```

13. Click on the "$" in the formatting section.
14. Type "2" in the text box.



15. Drag and drop **Average Sales Per Product without Context Transition (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Average Sales per Product (CM)	Average Sales per Product without Context Transition (CM)
2017	$11,928,556.40	$198,809.27	$11,928,556.40
2018	$30,516,895.19	$183,836.72	$30,516,895.19
2019	$42,895,108.14	$152,110.31	$42,895,108.14
2020	$24,468,714.27	$147,401.89	$24,468,714.27
Total	**$109,809,274.00**	**$313,740.78**	**$109,809,274.00**

As you can see without using measure, context transition does not take place and Filter context is comprised of only Date[Year].

What happens behind the scene?

To explain what happens, let us consider Year 2017.

Year	Total Sales Amount (CM)	Average Sales per Product (CM)	Average Sales per Product without Context Transition (CM)
2017	$11,928,556.40	$198,809.27	$11,928,556.40
2018	$30,516,895.19	$183,836.72	$30,516,895.19
2019	$42,895,108.14	$152,110.31	$42,895,108.14
2020	$24,468,714.27	$147,401.89	$24,468,714.27
Total	**$109,809,274.00**	**$313,740.78**	**$109,809,274.00**

Total Sales Amount with Context Transition:

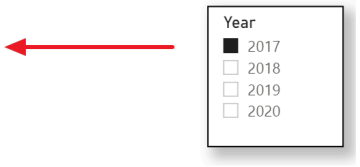
The following virtual table is generated in the DAX engine. Filter context is coming from Date[Year] = "2017" and current row of the Product Table.

ProductKey	First Product	Year	Total Sales Amount (CM)	Average Sales per Product (CM)
215	Sport-100 Helmet, Black	2017	$6,681.84	$6,681.84
218	Mountain Bike Socks, M	2017	$3,057.72	$3,057.72
219	Mountain Bike Socks, L	2017	$376.20	$376.20
220	Sport-100 Helmet, Blue	2017	$7,114.21	$7,114.21
223	AWC Logo Cap	2017	$2,686.92	$2,686.92
229	Long-Sleeve Logo Jersey, M	2017	$6,315.96	$6,315.96
232	Long-Sleeve Logo Jersey, L	2017	$14,758.08	$14,758.08
235	Long-Sleeve Logo Jersey, XL	2017	$7,181.16	$7,181.16
238	HL Road Frame - Red, 62	2017	$9,855.00	$9,855.00
241	HL Road Frame - Red, 44	2017	$8,338.85	$8,338.85
253	LL Road Frame - Black, 58	2017	$19,286.64	$19,286.64
256	LL Road Frame - Black, 60	2017	$178.58	$178.58
262	LL Road Frame - Red, 44	2017	$31,821.40	$31,821.40
264	LL Road Frame - Red, 48	2017	$20,049.32	$20,049.32
266	LL Road Frame - Red, 52	2017	$919.70	$919.70
270	LL Road Frame - Red, 60	2017	$31,453.49	$31,453.49

Total Sales Amount without Context Transition:

Filter context is only coming from Date[Year] = "2017".

ProductKey	Product	Average Sales per Product without Context Transition (CM)
486	All-Purpose Bike Stand	$11,928,556.40
223	AWC Logo Cap	$11,928,556.40
224	AWC Logo Cap	$11,928,556.40
225	AWC Logo Cap	$11,928,556.40
484	Bike Wash - Dissolver	$11,928,556.40
447	Cable Lock	$11,928,556.40
559	Chain	$11,928,556.40
473	Classic Vest, L	$11,928,556.40
472	Classic Vest, M	$11,928,556.40
471	Classic Vest, S	$11,928,556.40
485	Fender Set - Mountain	$11,928,556.40
555	Front Brakes	$11,928,556.40
552	Front Derailleur	$11,928,556.40



16. Click on Total Number of Products Sold (CM) in the All Measures table.

```
Total Number of Products Sold (CM) =
DISTINCTCOUNT ( Sales[ProductKey] )
```

Notice that this measure calculates the total distinct products sold.

17. Drag and drop **Total Number of Products Sold (CM)** from the All Measures table to the Values of matrix.

Year	Total Sales Amount (CM)	Average Sales per Product (CM)	Average Sales per Product without Context Transition (CM)	Total Number of Products Sold (CM)
2017	$11,928,556.40	$198,809.27	$11,928,556.40	60
2018	$30,516,895.19	$183,836.72	$30,516,895.19	166
2019	$42,895,108.14	$152,110.31	$42,895,108.14	282
2020	$24,468,714.27	$147,401.89	$24,468,714.27	166
Total	**$109,809,274.00**	**$313,740.78**	**$109,809,274.00**	**350**

Average Sales per Product:

Notice that there were 60 distinct products sold in 2017. Now lets methimatically verify the calculation of our measure Average Sales per Product (CM):

$$Avg\ Sales\ Amount\ per\ Product\ \frac{Total\ Sales\ in\ 2017}{Total\ No\ of\ Products\ Sold\ in\ 2017}$$

$$= \frac{11{,}928{,}556.40}{60} = 198,809.27$$

Average Sales Amount:

To see how to calculate Average Sales Amount:

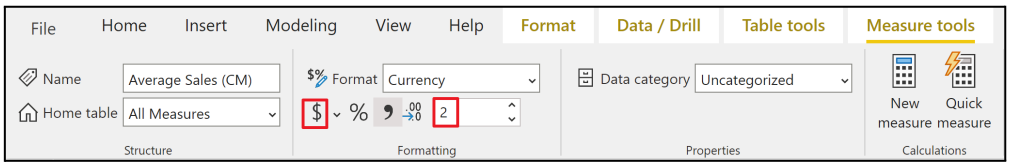
18. Right click on **All Measures** in the **Fields**.
19. Click on **New measure.**

    Measure tools open.

20. Type the following formula in the formula bar and press Enter.

```
Average Sales (CM) =
AVERAGE ( Sales[Sales Amount] )
```

21. Click on the "$" in the formatting section.
22. Type "2" in the text box.



23. Click on the Number of Sales (CM) in the All Measures table.

```
Number of Sales (CM) =
COUNTROWS ( Sales )
```

24. Drag and drop **Average Sales (CM)** and **Number of Sales (CM)** from the All Measures table to the values of the matrix.

Year	Total Sales Amount (CM)	Average Sales per Product (CM)	Average Sales per Product without Context Transition (CM)	Total Number of Products Sold (CM)	Average Sales (CM)	Number of Sales (CM)
2017	$11,928,556.40	$198,809.27	$11,928,556.40	60	$2,234.23	5339
2018	$30,516,895.19	$183,836.72	$30,516,895.19	166	$1,572.31	19409
2019	$42,895,108.14	$152,110.31	$42,895,108.14	282	$772.30	55542
2020	$24,468,714.27	$147,401.89	$24,468,714.27	166	$597.34	40963
Total	**$109,809,274.00**	**$313,740.78**	**$109,809,274.00**	**350**	**$905.62**	**121253**

Lets verify this value mathematically for the year 2017:

$$Avg\ Sales\ Amount = \frac{Total\ Sales\ in\ 2017s}{Total\ No\ of\ Sales\ in\ 2017} = \frac{11,928,556.40}{5339} = 2234.23$$

Notice that 60 distinct products were Sold in 2017 but total no of sales transactions were 5339 with Average Sales Amount of $2,234.23.

Be mindful that context transition is safe on dimension tables with unique identifiers (no duplicates). If performed on fact table (where there is a possibility of duplicates) it can return incorrect results.

Chapter 11: Advance Filtering in DAX

What will you learn in this chapter?

✔ Modification of Filter context using CALCULATE()

✔ Modify how filters are applied using KEEPFILTERS()

✔ Modification of multiple filters using CALCULATE()

✔ Using FILTER() with CALCULATE()

✔ Return all the rows in a table using All()

✔ Using Filter modifiers

✔ Using ALLSELECTED()

Modification of Filter context using CALCULATE()

CALCULATE() is the most important function in DAX. It is the only function that modifies the filter context. CALCULATE() can also additionally perform context transition.

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_1.pbix..

It is important to keep in mind that the CALCULATE() function works in the following order:

- ✔ A virtual table containing filtered rows is prepared. The condition for the filtering is defined by the second and proceeding arguments.
- ✔ The expression is evaluated on the filtered data.

To see how CALCULATE() modifies the existing filter context.

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

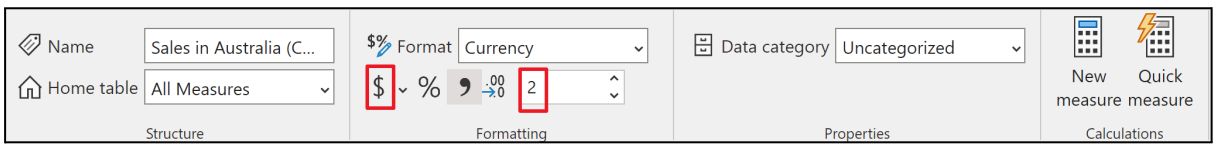
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Sales in Australia (CM) =
CALCULATE ( [Total Sales Amount (CM)], SalesTerritory[Country] = "Australia" )
```

In the above expression:

- *A virtual table containing the sales from Australia only is created.*
- *The Total Sales Amount is calculated for this virtual table containing filtered data.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Click on the **matrix** visual in the **visualization pane**.
7. Drag and drop **Category** and **Subcategory** from the Product table to the rows.

8. Drag and drop **Total Sales Amount (CM)** and **Sales in Australia (CM)** from the All Measures table to the Values.

Category	Total Sales Amount (CM)	Sales in Australia (CM)
⊞ **Accessories**	**$1,272,059.71**	**$162,638.18**
⊞ **Bikes**	**$94,620,526.59**	**$10,175,870.58**
⊞ **Clothing**	**$2,117,613.52**	**$113,175.73**
⊞ **Components**	**$11,799,074.18**	**$203,651.08**
Total	**$109,809,274.00**	**$10,655,335.57**

Notice that the initial filter context is comprised of Product [Category] and you have introducing another filter i.e. SalesTerritory[Country]="Australia" into the context.

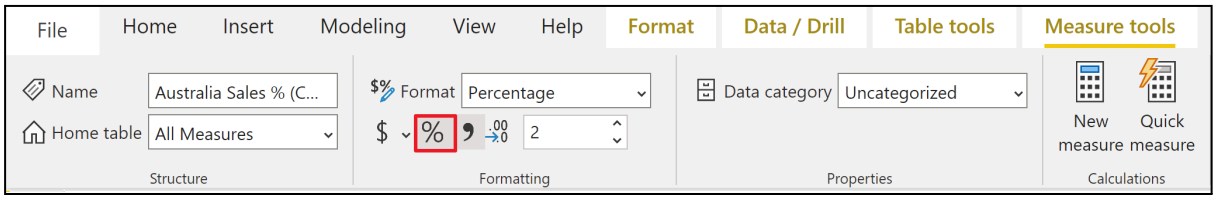
Now lets use this value to perform comparison.

9. Right click on **All Measures** in the **Fields**.
10. Click on **New measure.**

    Measure tools open.
11. Type the following formula in the formula bar and press Enter.

```
Australia Sales % (CM) =
DIVIDE ( [Sales in Australia (CM)], [Total Sales Amount (CM)], 0 )
```

12. Click on the "%" in the formatting section.



13. Drag and drop **Australia Sales % (CM)** from the All Measures table to the Values.

Category	Total Sales Amount (CM)	Sales in Australia (CM)	Australia Sales % (CM)
⊞ **Accessories**	**$1,272,059.71**	**$162,638.18**	**12.79 %**
⊟ **Bikes**	**$94,620,526.59**	**$10,175,870.58**	**10.75 %**
Mountain Bikes	$36,445,444.52	$2,972,755.45	8.16 %
Road Bikes	$43,878,791.78	$5,007,480.08	11.41 %
Touring Bikes	$14,296,290.29	$2,195,635.05	15.36 %
⊞ **Clothing**	**$2,117,613.52**	**$113,175.73**	**5.34 %**
⊞ **Components**	**$11,799,074.18**	**$203,651.08**	**1.73 %**
Total	**$109,809,274.00**	**$10,655,335.57**	**9.70 %**

Notice that the sales in Australia are only 9.70 % of the total sales.

Modify how filters are applied using KEEPFILTERS()

If the initial filter context and the one introduced by CALCULATE() are in conflict. CALCULATE() will override the initial filter context. .

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_2.pbix..

To see what this means:

1. Remove the **Category** from the rows of the matrix.
2. Remove **Australia Sales % (CM)** from the values.
3. Drag and drop **Country** from **SalesTerritoy** table to the Values.

Country	Total Sales Amount (CM)	Sales in Australia (CM)
Australia	$10,655,335.57	$10,655,335.57
Canada	$16,355,770.60	$10,655,335.57
Corporate HQ		$10,655,335.57
France	$7,251,555.66	$10,655,335.57
Germany	$4,878,300.11	$10,655,335.57
United Kingdom	$7,670,721.13	$10,655,335.57
United States	$62,997,590.93	$10,655,335.57
Total	**$109,809,274.00**	**$10,655,335.57**

Notice that the same value is shown for all the countries and this value bescially is the sales amount for Australia only. For all countries other then Australia, the initial filter context is in conflict with that from CALCULATE() and CALCULATE() overrides the incoming filter context. The reason for this behavior is that internally Sales in Australia (CM) is converted into below for execution.

```
Sales in Australia (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    FILTER (
        ALL ( SalesTerritory[Country] ),
        SalesTerritory[Country] = "Australia"
    )
)
```

In the above expression:

- *ALL() ignores the filter coming from Country Column.*
- *FILTER() filters the Sales from "Australia" only and stores in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

Intersect:

You can stop CALCULATE() from overriding the initial filter context by using KEEPFILTERS().

To see how KEEPFILTERS() work:

4. Right click on **All Measures** in the **Fields**.
5. Click on **New measure.**

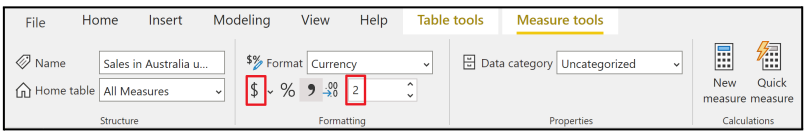
   Measure tools open.
6. Type the following formula in the formula bar and press Enter.

```
Sales in Australia using KEEPFILTERS (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    KEEPFILTERS ( SalesTerritory[Country] = "Australia" )
)
```

In the above expression:

- *KEEPFILTER() modifies and filters the Sales from "Australia" only and stores results in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

7. Click on the "$" in the formatting section.
8. Type "2" in the text box.



9. Drag and drop **Sales in Australia using KEEPFILTERS (CM)** from the All Measures table to the Values.

Country	Total Sales Amount (CM)	Sales in Australia (CM)	Sales in Australia using KEEPFILTERS (CM)
Australia	$10,655,335.57	$10,655,335.57	$10,655,335.57
Canada	$16,355,770.60	$10,655,335.57	
Corporate HQ		$10,655,335.57	
France	$7,251,555.66	$10,655,335.57	
Germany	$4,878,300.11	$10,655,335.57	
United Kingdom	$7,670,721.13	$10,655,335.57	
United States	$62,997,590.93	$10,655,335.57	
Total	**$109,809,274.00**	**$10,655,335.57**	**$10,655,335.57**

Notice the change in behaviour of CALCULATE(). Sales in Australia using KEEPFILTERS (CM) shows values only for cells where the incoming filter context is not in conflict with that of CALCULATE().

Modification of multiple filters with CALCULATE()

Now you will learn how to introduce a filter with multiple values in a same column using || operator.

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_3.pbix..

To see how to use || for a single column:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

Measure tools open.

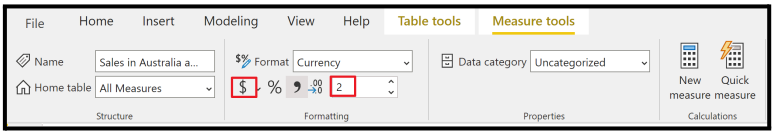
3. Type the following formula in the formula bar and press Enter.

```
Sales in Australia and Canada (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
     ( SalesTerritory[Country] = "Australia"
        || SalesTerritory[Country] = "Canada" )
)
```

In the above expression:

- *A subset of rows containing sales for Australia or Canada is calculated and stored in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Drag and drop **Sales in Australia and Canada (CM)** from the All Measures table to the Values.

Country	Total Sales Amount (CM)	Sales in Australia (CM)	Sales in Australia using KEEPFILTERS (CM)	Sales in Australia and Canada (CM)
Australia	$10,655,335.57	$10,655,335.57	$10,655,335.57	$27,011,106.17
Canada	$16,355,770.60	$10,655,335.57		$27,011,106.17
Corporate HQ		$10,655,335.57		$27,011,106.17
France	$7,251,555.66	$10,655,335.57		$27,011,106.17
Germany	$4,878,300.11	$10,655,335.57		$27,011,106.17
United Kingdom	$7,670,721.13	$10,655,335.57		$27,011,106.17
United States	$62,997,590.93	$10,655,335.57		$27,011,106.17
Total	**$109,809,274.00**	**$10,655,335.57**	**$10,655,335.57**	**$27,011,106.17**

Notice that the measure Sales in Australia and Canada (CM) shows one single value which is the summation of sales in Australia and Canada. This measure does not use KEEPFILTERS() therefore, CALCULATE() overrides the initial filter context.

7. Right click on **All Measures** in the **Fields**.
8. Click on **New measure.**

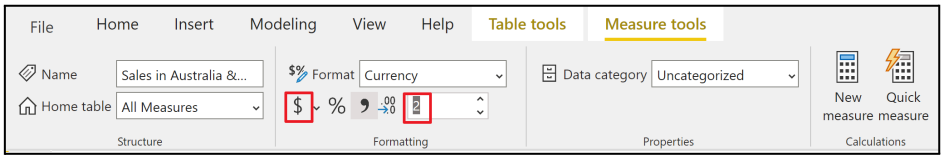
   Measure tools open.
9. Type the following formula in the formula bar and press Enter.

```
Sales in Australia & Canada Using KEEPFILTERS (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    KEEPFILTERS ( SalesTerritory[Country] = "Australia"
        || SalesTerritory[Country] = "Canada" )
)
```

In the above expression:

- *A modified and filtered subset of rows containing sales for Australia or Canada is calculated and stored in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

10. Click on the "$" in the formatting section.
11. Type "2" in the text box.



12. Drag and drop **Sales in Australia and Canada using KEEPFILTER (CM)** from the All Measures table to the Values.

Country	Total Sales Amount (CM)	Sales in Australia (CM)	Sales in Australia using KEEPFILTERS (CM)	Sales in Australia and Canada (CM)	Sales in Australia & Canada Using KEEPFILTERS (CM)
Australia	$10,655,335.57	$10,655,335.57	$10,655,335.57	$27,011,106.17	$10,655,335.57
Canada	$16,355,770.60	$10,655,335.57		$27,011,106.17	$16,355,770.60
Corporate HQ		$10,655,335.57		$27,011,106.17	
France	$7,251,555.66	$10,655,335.57		$27,011,106.17	
Germany	$4,878,300.11	$10,655,335.57		$27,011,106.17	
United Kingdom	$7,670,721.13	$10,655,335.57		$27,011,106.17	
United States	$62,997,590.93	$10,655,335.57		$27,011,106.17	
Total	**$109,809,274.00**	**$10,655,335.57**	**$10,655,335.57**	**$27,011,106.17**	**$27,011,106.17**

Notice that the Sales in Australia and Canada using KEEPFILTER (CM) shows the results for Canada and Australia only.

Now let's change the incoming filter context.

13. Remove the **Country** from the rows of the matrix.
14. Drag and drop **Category** from Product table to the Rows.

Category	Total Sales Amount (CM)	Sales in Australia (CM)	Sales in Australia using KEEPFILTERS (CM)	Sales in Australia and Canada (CM)	Sales in Australia & Canada Using KEEPFILTERS (CM)
Accessories	$1,272,059.71	$162,638.18	$162,638.18	$384,143.76	$384,143.76
Bikes	$94,620,526.59	$10,175,870.58	$10,175,870.58	$23,633,553.67	$23,633,553.67
Clothing	$2,117,613.52	$113,175.73	$113,175.73	$545,288.08	$545,288.08
Components	$11,799,074.18	$203,651.08	$203,651.08	$2,448,120.66	$2,448,120.66
Total	**$109,809,274.00**	**$10,655,335.57**	**$10,655,335.57**	**$27,011,106.17**	**$27,011,106.17**

Notice that you get same values in the cells with or without KEEPFILTERS() because CALCULATE() is no more in conflict with the incoming filter context.

Now let's use IN operator to select multiple values. We will use curly braces {} to enclose multiple values.

15. Right click on **All Measures** in the **Fields**.
16. Click on **New measure.**

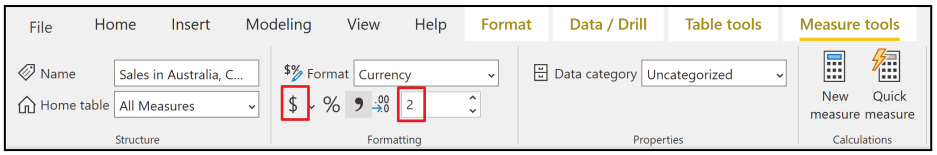
    Measure tools open.
17. Type the following formula in the formula bar and press Enter.

```
Sales in Australia, Canada & UK (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    SalesTerritory[Country] IN { "Australia", "Canada", "United Kingdom" }
)
```

In the above expression:

- *A subset of rows containing sales for Australia or Canada or United Kingdom is calculated and stored in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

18. Click on the "$" in the formatting section.
19. Type "2" in the text box.



20. Drag and drop Sales in Australia, Canada & UK (CM) from the All Measures table to the Values.

Category	Total Sales Amount (CM)	Sales in Australia (CM)	Sales in Australia using KEEPFILTERS (CM)	Sales in Australia and Canada (CM)	Sales in Australia & Canada Using KEEPFILTERS (CM)	Sales in Australia, Canada & UK (CM)
Accessories	$1,272,059.71	$162,638.18	$162,638.18	$384,143.76	$384,143.76	$503,366.93
Bikes	$94,620,526.59	$10,175,870.58	$10,175,870.58	$23,633,553.67	$23,633,553.67	$30,322,143.39
Clothing	$2,117,613.52	$113,175.73	$113,175.73	$545,288.08	$545,288.08	$696,356.46
Components	$11,799,074.18	$203,651.08	$203,651.08	$2,448,120.66	$2,448,120.66	$3,159,960.52
Total	**$109,809,274.00**	**$10,655,335.57**	**$10,655,335.57**	**$27,011,106.17**	**$27,011,106.17**	**$34,681,827.30**

Notice that using the IN operator, two or more then two values can be selected.

Using FILTER() with CALCULATE()

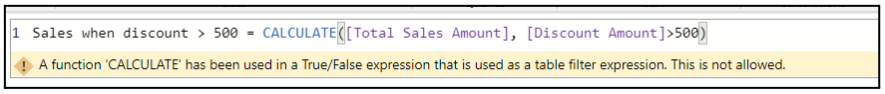
In this exercise you will learn some limitations of CALCULATE(). One limitation of CALCULATE() is that you cannot use measure or an expression as a filter parameter of CALCULATE().

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_4.pbix..

To see how to solve this issue:

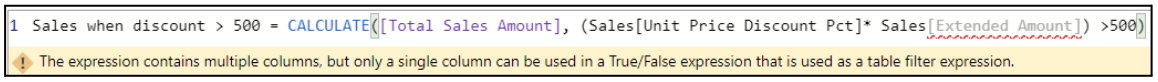
1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

   Measure tools open.
3. Type the following formula in the formula bar.



Notice the warning generated on using the measure as a filter parameter in CALCULATE().

4. Replace the measure with an expression to filter.



Notice that a different warning is generated this time. However it is still not an allowed syntax.

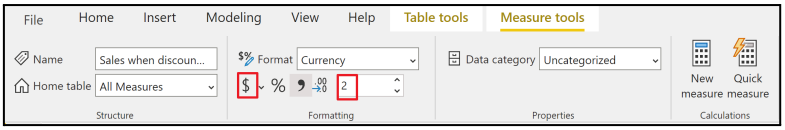
5. Replace the incorrect syntax with FILTER():

```
Sales when discount > 500 (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    FILTER ( Sales, [Discount Amount (CM)] > 500 )
)
```

In the above expression:

- *A subset of rows containing sales where the Discount is greater than 500 is calculated and stored in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

6. Click on the "$" in the formatting section.
7. Type "2" in the text box.



8. Click on the **matrix** visual in the **visualization pane**.
9. Drag and drop **Country** from the SalesTerritory table to the rows.
10. Drag and drop **Total Sales Amount (CM)** and **Sales when discount > 500 (CM)** from the All Measures table to the Values.

Country	Total Sales Amount (CM)	Sales when discount > 500 (CM)
Australia	$10,655,335.57	$50,351.56
Canada	$16,355,770.60	$187,644.71
France	$7,251,555.66	$172,440.31
Germany	$4,878,300.11	$80,381.39
United Kingdom	$7,670,721.13	$106,627.54
United States	$62,997,590.93	$853,275.27
Total	**$109,809,274.00**	**$1,450,720.78**

Notice how FILTER() has fixed the issue of using measure or expression as filters in CALCULATE().

Return all the rows in a table using All()

In this exercise, you will learn how to use ALL() to ignore all filters.

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_5.pbix..

To use ALL():

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

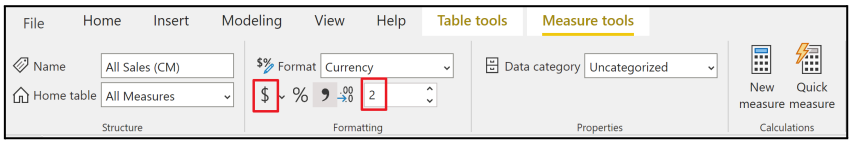
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
All Sales (CM) =
CALCULATE ( [Total Sales Amount (CM)], ALL ( SalesTerritory ) )
```

In the above expression:

- *A subset of rows containing sales (ignoring the filter coming from SalesTerritory) table is calculated and stored in a virtual table.*
- *The total sales amount for the rows in the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Click on the **matrix** visual in the **visualization pane**.
7. Drag and drop **Country** from the SalesTerritory table to the rows.
8. Drag and drop **Total Sales Amount (CM)** and **All Sales (CM)** from the All Measures table to the Values.

Country	Total Sales Amount (CM)	All Sales (CM)
Australia	$10,655,335.57	$109,809,274.00
Canada	$16,355,770.60	$109,809,274.00
Corporate HQ		$109,809,274.00
France	$7,251,555.66	$109,809,274.00
Germany	$4,878,300.11	$109,809,274.00
United Kingdom	$7,670,721.13	$109,809,274.00
United States	$62,997,590.93	$109,809,274.00
Total	**$109,809,274.00**	**$109,809,274.00**

Notice that the incomming filter context is ignored by All Sales (CM) and the value of the measure is the total sales amount as it includes all the countries. The incoming filter context is also in conflict with CALCULATE().

Using Filter modifiers

So far you have used ALL() to remove any incoming filters.

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_6.pbix..

Let's use a filter modifier with a more intuitive name.

REMOVEFILTERS() can only be used to clear filters but not to return a table.

To use REMOVEFILTERS():

1. Right click on *All Sales (CM)* in the **All Measures**.

   Measure tools open.

2. Type the following formula in the formula bar and press Enter.

```
All Sales (CM) =
CALCULATE ( [Total Sales Amount (CM)], REMOVEFILTERS ( SalesTerritory ) )
```

Notice that REMOVEFILTERS() performs the same function as ALL() but the name of the function is easier to comprehend.

Country	Total Sales Amount (CM)	All Sales (CM)
Australia	$10,655,335.57	$109,809,274.00
Canada	$16,355,770.60	$109,809,274.00
Corporate HQ		$109,809,274.00
France	$7,251,555.66	$109,809,274.00
Germany	$4,878,300.11	$109,809,274.00
United Kingdom	$7,670,721.13	$109,809,274.00
United States	$62,997,590.93	$109,809,274.00
Total	**$109,809,274.00**	**$109,809,274.00**

Let's use this value to calculate percentage of total sales.

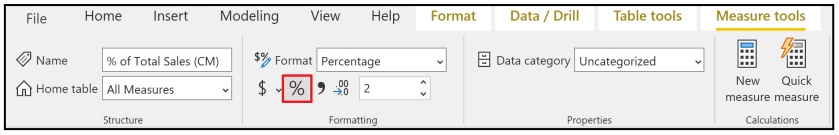
3. Right click on **All Measures** in the **Fields**.
4. Click on **New measure.**

   Measure tools open.

5. Type the following formula in the formula bar and press Enter.

```
% of Total Sales (CM) =
DIVIDE ( [Total Sales Amount (CM)], [All Sales (CM)], 0 )
```

6. Click on the "%" in the formatting section.



7. Remove the **All Sales (CM)** from the Values of the matrix.

   ALL Sales (CM) Measure is not required in the visual. It was only created as an intermediary measure.

8. Drag and drop **% of Total Sales (CM)** from All Measures table to the Values.

Country	Total Sales Amount (CM)	% of Total Sales (CM)
Australia	$10,655,335.57	9.70 %
Canada	$16,355,770.60	14.89 %
France	$7,251,555.66	6.60 %
Germany	$4,878,300.11	4.44 %
United Kingdom	$7,670,721.13	6.99 %
United States	$62,997,590.93	57.37 %
Total	**$109,809,274.00**	**100.00 %**

Notice the effect of incoming filter context.

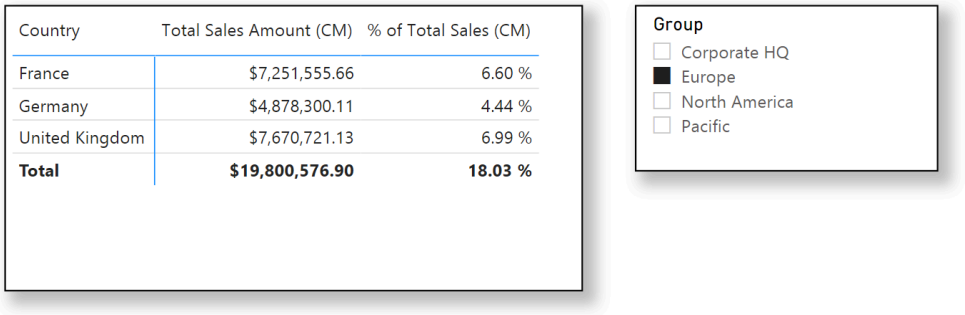
Using ALLSELECTED()

In this exercise, you will learn when do functions like ALL() and REMOVEFILTERS() fail and how to create robust measures.

Continue using the previous file for the exercise or open the file Advance_Filtering_In_DAX_7.pbix..

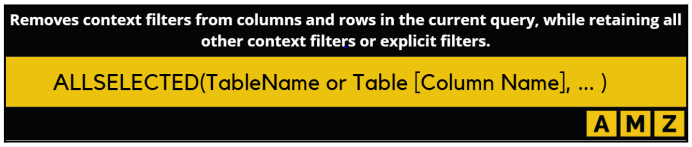
To understand this concept:

1. Click on **Slicer** in the **Visualization Pane**.
2. Drag and drop **Group** from SalesTerritoty to the Field.
3. Click on Europe.



Country	Total Sales Amount (CM)	% of Total Sales (CM)
France	$7,251,555.66	6.60 %
Germany	$4,878,300.11	4.44 %
United Kingdom	$7,670,721.13	6.99 %
Total	**$19,800,576.90**	**18.03 %**

Notice that the % of Total Sales shows the value of 18.03% as total. This means that the measure has no effect of the selection in the slicer and continues to shows results based on initial calculation.



To see how ALLSELECTED() can be used to solve this:

4. Right click on **All Measures** in the **Fields**.
5. Click on **New measure.**

   Measure tools open.

6. Type the following formula in the formula bar and press Enter.

```
ALL Selected Country Sales (CM) =
CALCULATE ( [Total Sales Amount (CM)], ALLSELECTED ( SalesTerritory ) )
```

You do not need to format this measure because it is in intermediate measure.

Lets use this measure to find the percentage of sales.

7. Right click on **All Measures** in the **Fields**.

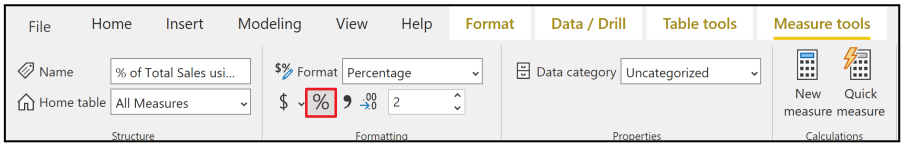
8. Click on **New measure.**

   Measure tools open.

9. Type the following formula in the formula bar and press Enter.

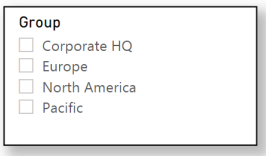
```
% of Total Sales using ALLSELECTED (CM) =
DIVIDE ( [Total Sales Amount (CM)], [ALL Selected Country Sales (CM)], 0 )
```

10. Click on the "%" in the formatting section.



11. Drag and drop **ALL Selected Country Sales (CM)** and **% of Total Sales using ALLSELECTED (CM)** from the All Measures table to the Values.

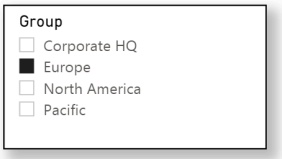
Country	Total Sales Amount (CM)	% of Total Sales (CM)	ALL Selected Country Sales (CM)	% of Total Sales using ALLSELECTED (CM)
Australia	$10,655,335.57	9.70 %	109,809,274.00	9.70 %
Canada	$16,355,770.60	14.89 %	109,809,274.00	14.89 %
Corporate HQ			109,809,274.00	
France	$7,251,555.66	6.60 %	109,809,274.00	6.60 %
Germany	$4,878,300.11	4.44 %	109,809,274.00	4.44 %
United Kingdom	$7,670,721.13	6.99 %	109,809,274.00	6.99 %
United States	$62,997,590.93	57.37 %	109,809,274.00	57.37 %
Total	**$109,809,274.00**	**100.00 %**	**109,809,274.00**	**100.00 %**



Notice that the % of Total Sales using ALLSELECTED (CM) has a value of 100% and nothing is selected in the slicer.

12. Click on Europe.

Country	Total Sales Amount (CM)	% of Total Sales (CM)	ALL Selected Country Sales (CM)	% of Total Sales using ALLSELECTED (CM)
France	$7,251,555.66	6.60 %	19,800,576.90	36.62 %
Germany	$4,878,300.11	4.44 %	19,800,576.90	24.64 %
United Kingdom	$7,670,721.13	6.99 %	19,800,576.90	38.74 %
Total	**$19,800,576.90**	**18.03 %**	**19,800,576.90**	**100.00 %**



Notice that after selecting Europe, you still have 100% as the total value and contribution from each country in the group has been recalculated.

Chapter 12: Time Intelligence in DAX

What will you learn in this chapter?

✔ What is Time Intelligence

✔ Aggregations over time

✔ Total Year till Date, Month Till Date and Quarter till Date

✔ Same Period Last Year

✔ Handling Fiscal Year

✔ Finding YOY%

✔ Using PARALLELPERIOD()

✔ Finding Moving Total

✔ Calculate Running Total

What is Time Intelligence

Time intelligence allows you to manipulate data using ***fixed time intervals*** such as days, months, quarters and years. Time intelligence functions in DAX require a date table inside the data model to slice and dice the data.

Date table properties for Time Intelligence

For smooth working of time intelligence functions in Power BI:

- ✔ The calendar date table must be marked as the date table.
- ✔ The date column should have only one date per row.
- ✔ All dates should be present with no gaps.
- ✔ The dates should be complete from 1st of January to 31st of December.
- ✔ If your date column also contains time, remove or separate the time component
- ✔ If the report only references fiscal years, then the date table must include all the dates from the first to the last day of a fiscal year.

Aggregations over time

In this topic, you will learn how to aggregate values over time starting from 1st of January and end 31st of December.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_1.pbix..

Using DATE():

Initially, let's try to create a custom code with fixed date range using DATE().

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

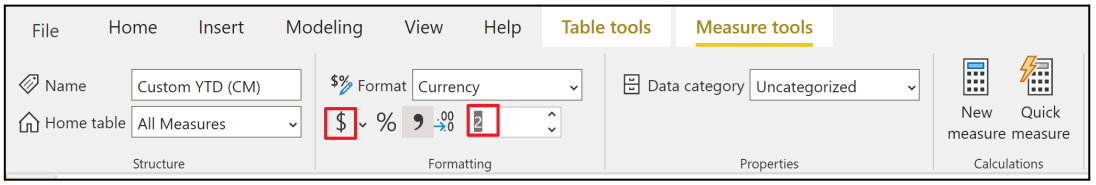
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Custom YTD (CM) =
CALCULATE (
    SUM ( Sales[Sales Amount] ),
    'Date'[Date] >= DATE ( 2018, 1, 1 )
        && 'Date'[Date] <= DATE ( 2018, 5, 15 )
)
```

In the above expression:

- *A virtual date table containing the dates from 1st of Jan 2018 to 15th of April is calculated.*
- *The total sales amount for the dates in the rows of the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Click on the **matrix** visual in the **visualization pane**.
7. Drag and drop **Year, Month Name** and **Date** column from the Date table to the rows.

Notice that automatic date hierarchy is created.

8. Drag and drop **Total Sales Amount (CM)** and **Custom YTD (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Custom YTD (CM)
2017	**$11,928,556.40**	**$8,801,394.26**
Jan		$8,801,394.26
Feb		$8,801,394.26
Mar		$8,801,394.26
Apr		$8,801,394.26
May		$8,801,394.26
Jun		$8,801,394.26
Jul	$1,423,357.36	$8,801,394.26
Aug	$2,057,902.54	$8,801,394.26
Sep	$2,523,947.76	$8,801,394.26
Oct	$561,681.51	$8,801,394.26
Nov	$4,764,920.63	$8,801,394.26
Dec	$596,746.60	$8,801,394.26
2018	**$30,516,895.19**	**$8,801,394.26**
2019	**$42,895,108.14**	**$8,801,394.26**
2020	**$24,468,714.27**	**$8,801,394.26**
Total	**$109,809,274.00**	**$8,801,394.26**

Notice that Custom YTD (CM) shows the fixed value of sales from 1st of January 2018 to 15th of April 2018. The incoming filter from rows is modified by CALCULATE().

Using DATEBETWEEN():

Now let's use DATEBETWEEN() to create a cleaner version of the above code.

9. Click on Custom YTD (CM) from **All Measures** in the **Fields**.

   Measure tools open.

10. Type the following formula in the formula bar and press Enter.

```
Custom YTD (CM) =
CALCULATE (
    SUM ( Sales[Sales Amount] ),
    DATESBETWEEN ( 'Date'[Date], DATE ( 2018, 1, 1 ), DATE ( 2018, 5, 15 ) )
)
```

Notice that DATESBETWEEN() has increased the code readability, the code is still not dynamic and shows the same result as above.

Let's try to a create a dynamic version of aggregation over time using variables.

11. Right click on **All Measures** in the **Fields**.
12. Click on **New measure.**

    Measure tools open.

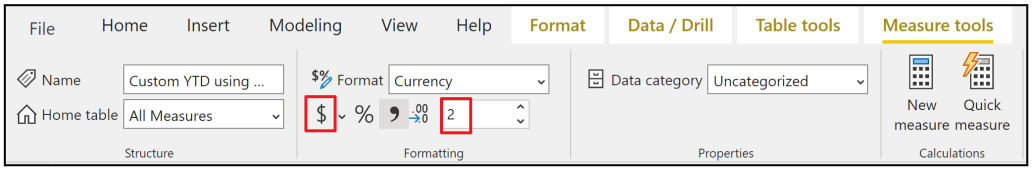
13. Type the following formula in the formula bar and press Enter.

```
Custom YTD using Variables (CM) =
VAR LastDateofSales =
    MAX ( 'Date'[Date] )
RETURN
    CALCULATE (
        SUM ( Sales[Sales Amount] ),
        DATESBETWEEN (
            'Date'[Date],
            DATE ( YEAR ( LastDateofSales ), 1, 1 ),
            LastDateofSales
        )
    )
```

In this expression:

- *For the starting point of YTD, the year of the last selection is calculated.*
- *The last date in the current filter context is calculated using MAX().*
- *This is passed as the last argument to YEAR() which is then passed as an argument to DATESBETWEEN().*

14. Click on the "$" in the formatting section.
15. Type "2" in the text box.



16. Drag and drop **Custom YTD using Variables (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)
2017	**$11,928,556.40**	**$8,801,394.26**	**$11,928,556.40**
Jan		$8,801,394.26	
Feb		$8,801,394.26	
Mar		$8,801,394.26	
Apr		$8,801,394.26	
May		$8,801,394.26	
Jun		$8,801,394.26	
Jul	$1,423,357.36	$8,801,394.26	$1,423,357.36
Aug	$2,057,902.54	$8,801,394.26	$3,481,259.90
Sep	$2,523,947.76	$8,801,394.26	$6,005,207.66
Oct	$561,681.51	$8,801,394.26	$6,566,889.17
Nov	$4,764,920.63	$8,801,394.26	$11,331,809.80
Dec	$596,746.60	$8,801,394.26	$11,928,556.40
2018	**$30,516,895.19**	**$8,801,394.26**	**$30,516,895.19**
2019	**$42,895,108.14**	**$8,801,394.26**	**$42,895,108.14**
2020	**$24,468,714.27**	**$8,801,394.26**	**$24,468,714.27**
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**

Notice that the first transaction was recorded in July 2017 and the Custom YTD using Variables (CM) starts calculating the aggregation from July onwards. Due to the incoming filter context of months on the rows, the value for each month is calculated and displayed.

Total Year till Date, Month till Date and Quarter till Date

Up till now, you have learnt how to find aggregations over time using fixed time range and using variables. In this exercise, you will learn how to use Time intelligence functions in DAX to get the same results in a simpler and optimized way. This function is used within calculate to compute an expression.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_2.pbix..

Using DATESYTD():

To understand how DATESYTD() is used with CALCULATE():

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

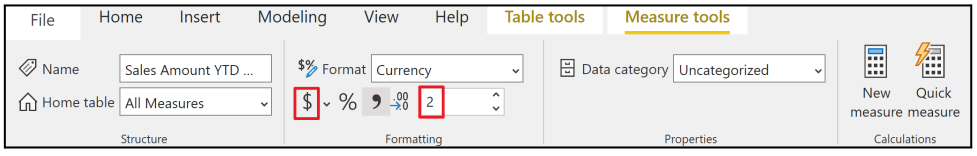
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Sales Amount YTD using DATESYTD (CM) =
CALCULATE ( [Total Sales Amount (CM)], DATESYTD ( 'Date'[Date] ) )
```

In the above expression:

- *A virtual date table containing the dates is calculated.*
- *The total sales amount for the dates in the rows of the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Drag and drop **Sales Amount YTD using DATESYTD (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)	Sales Amount YTD using DATESYTD (CM)
2017	$11,928,556.40	$8,801,394.26	$11,928,556.40	$11,928,556.40
2018	$30,516,895.19	$8,801,394.26	$30,516,895.19	$30,516,895.19
2019	$42,895,108.14	$8,801,394.26	$42,895,108.14	$42,895,108.14
2020	$24,468,714.27	$8,801,394.26	$24,468,714.27	$24,468,714.27
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**

Notice that the same results are produced.

Now let's look at another function which calculates the total value till date.

Using TOTALYTD():

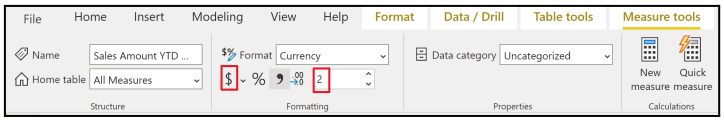
To use TOTALYTD():

7. Right click on **All Measures** in the **Fields**.
8. Click on **New measure.**

   Measure tools open.
9. Type the following formula in the formula bar and press Enter.

```
Sales Amount YTD using TOTALYTD (CM) =
TOTALYTD ( [Total Sales Amount (CM)], 'Date'[Date] )
```

10. Click on the "$" in the formatting section.
11. Type "2" in the text box.

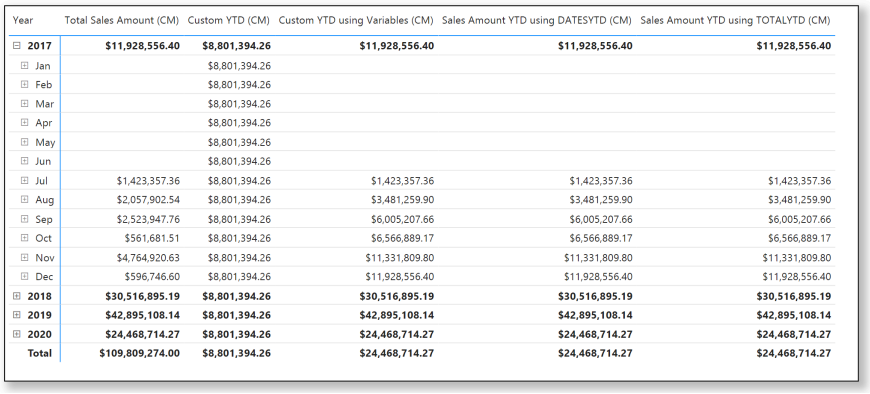


12. Drag and drop **Sales Amount YTD using TOTALYTD (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)	Sales Amount YTD using DATESYTD (CM)	Sales Amount YTD using TOTALYTD (CM)
2017	$11,928,556.40	$8,801,394.26	$11,928,556.40	$11,928,556.40	$11,928,556.40
2018	$30,516,895.19	$8,801,394.26	$30,516,895.19	$30,516,895.19	$30,516,895.19
2019	$42,895,108.14	$8,801,394.26	$42,895,108.14	$42,895,108.14	$42,895,108.14
2020	$24,468,714.27	$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**

Notice that all the three measures produce the same results.

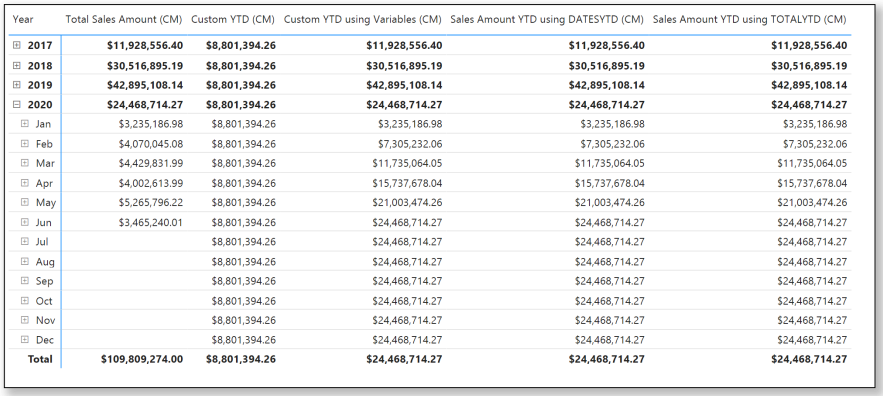
13. Click on “+” next to 2017 in the Year.



Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)	Sales Amount YTD using DATESYTD (CM)	Sales Amount YTD using TOTALYTD (CM)
2017	**$11,928,556.40**	**$8,801,394.26**	**$11,928,556.40**	**$11,928,556.40**	**$11,928,556.40**
Jan		$8,801,394.26			
Feb		$8,801,394.26			
Mar		$8,801,394.26			
Apr		$8,801,394.26			
May		$8,801,394.26			
Jun		$8,801,394.26			
Jul	$1,423,357.36	$8,801,394.26	$1,423,357.36	$1,423,357.36	$1,423,357.36
Aug	$2,057,902.54	$8,801,394.26	$3,481,259.90	$3,481,259.90	$3,481,259.90
Sep	$2,523,947.76	$8,801,394.26	$6,005,207.66	$6,005,207.66	$6,005,207.66
Oct	$561,681.51	$8,801,394.26	$6,566,889.17	$6,566,889.17	$6,566,889.17
Nov	$4,764,920.63	$8,801,394.26	$11,331,809.80	$11,331,809.80	$11,331,809.80
Dec	$596,746.60	$8,801,394.26	$11,928,556.40	$11,928,556.40	$11,928,556.40
2018	**$30,516,895.19**	**$8,801,394.26**	**$30,516,895.19**	**$30,516,895.19**	**$30,516,895.19**
2019	**$42,895,108.14**	**$8,801,394.26**	**$42,895,108.14**	**$42,895,108.14**	**$42,895,108.14**
2020	**$24,468,714.27**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**

Notice that the first transaction was observed in July 2017, so no totals exist for dates before July, 2020.

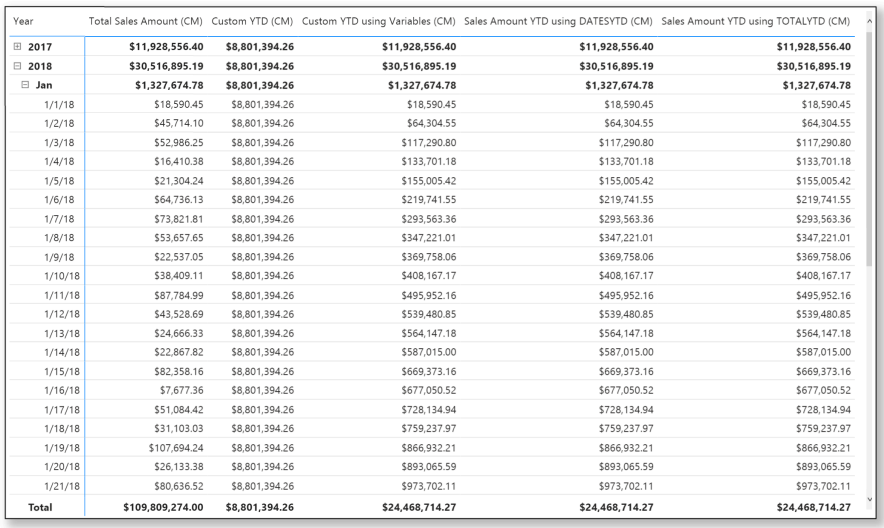
14. Click on “+” next to 2020 in the Year.



Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)	Sales Amount YTD using DATESYTD (CM)	Sales Amount YTD using TOTALYTD (CM)
2017	**$11,928,556.40**	**$8,801,394.26**	**$11,928,556.40**	**$11,928,556.40**	**$11,928,556.40**
2018	**$30,516,895.19**	**$8,801,394.26**	**$30,516,895.19**	**$30,516,895.19**	**$30,516,895.19**
2019	**$42,895,108.14**	**$8,801,394.26**	**$42,895,108.14**	**$42,895,108.14**	**$42,895,108.14**
2020	**$24,468,714.27**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**
Jan	$3,235,186.98	$8,801,394.26	$3,235,186.98	$3,235,186.98	$3,235,186.98
Feb	$4,070,045.08	$8,801,394.26	$7,305,232.06	$7,305,232.06	$7,305,232.06
Mar	$4,429,831.99	$8,801,394.26	$11,735,064.05	$11,735,064.05	$11,735,064.05
Apr	$4,002,613.99	$8,801,394.26	$15,737,678.04	$15,737,678.04	$15,737,678.04
May	$5,265,796.22	$8,801,394.26	$21,003,474.26	$21,003,474.26	$21,003,474.26
Jun	$3,465,240.01	$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Jul		$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Aug		$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Sep		$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Oct		$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Nov		$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Dec		$8,801,394.26	$24,468,714.27	$24,468,714.27	$24,468,714.27
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**

Notice that no transaction was recorded after Jun 2020 so the same value is shown for all the months after Jun 2020.

15. Click on “+” next to 2018 in the Year. Click on “+” next to Jan in the Year.



Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)	Sales Amount YTD using DATESYTD (CM)	Sales Amount YTD using TOTALYTD (CM)
2017	**$11,928,556.40**	**$8,801,394.26**	**$11,928,556.40**	**$11,928,556.40**	**$11,928,556.40**
2018	**$30,516,895.19**	**$8,801,394.26**	**$30,516,895.19**	**$30,516,895.19**	**$30,516,895.19**
Jan	**$1,327,674.78**	**$8,801,394.26**	**$1,327,674.78**	**$1,327,674.78**	**$1,327,674.78**
1/1/18	$18,590.45	$8,801,394.26	$18,590.45	$18,590.45	$18,590.45
1/2/18	$45,714.10	$8,801,394.26	$64,304.55	$64,304.55	$64,304.55
1/3/18	$52,986.25	$8,801,394.26	$117,290.80	$117,290.80	$117,290.80
1/4/18	$16,410.38	$8,801,394.26	$133,701.18	$133,701.18	$133,701.18
1/5/18	$21,304.24	$8,801,394.26	$155,005.42	$155,005.42	$155,005.42
1/6/18	$64,736.13	$8,801,394.26	$219,741.55	$219,741.55	$219,741.55
1/7/18	$73,821.81	$8,801,394.26	$293,563.36	$293,563.36	$293,563.36
1/8/18	$53,657.65	$8,801,394.26	$347,221.01	$347,221.01	$347,221.01
1/9/18	$22,537.05	$8,801,394.26	$369,758.06	$369,758.06	$369,758.06
1/10/18	$38,409.11	$8,801,394.26	$408,167.17	$408,167.17	$408,167.17
1/11/18	$87,784.99	$8,801,394.26	$495,952.16	$495,952.16	$495,952.16
1/12/18	$43,528.69	$8,801,394.26	$539,480.85	$539,480.85	$539,480.85
1/13/18	$24,666.33	$8,801,394.26	$564,147.18	$564,147.18	$564,147.18
1/14/18	$22,867.82	$8,801,394.26	$587,015.00	$587,015.00	$587,015.00
1/15/18	$82,358.16	$8,801,394.26	$669,373.16	$669,373.16	$669,373.16
1/16/18	$7,677.36	$8,801,394.26	$677,050.52	$677,050.52	$677,050.52
1/17/18	$51,084.42	$8,801,394.26	$728,134.94	$728,134.94	$728,134.94
1/18/18	$31,103.03	$8,801,394.26	$759,237.97	$759,237.97	$759,237.97
1/19/18	$107,694.24	$8,801,394.26	$866,932.21	$866,932.21	$866,932.21
1/20/18	$26,133.38	$8,801,394.26	$893,065.59	$893,065.59	$893,065.59
1/21/18	$80,636.52	$8,801,394.26	$973,702.11	$973,702.11	$973,702.11
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**

Notice that all the measures show value to the day level granularity.

Handling Fiscal Year

Fiscal year calculations are widely used in financial reporting. For fiscal year you will just add the fiscal year argument in our standard time intelligence functions.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_3.pbix..

To see how to modify time intelligence functions in DAX for fiscal year reporting:

1. Right click on **All Measures** in the **Fields**.
2. Click on **New measure.**

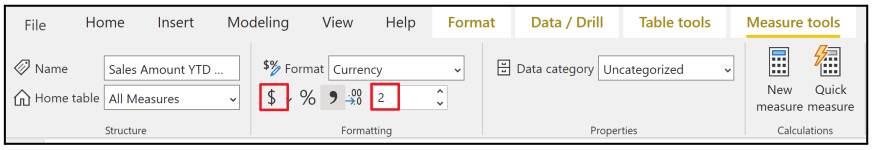
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Sales Amount YTD for Fiscal Year (CM) =
TOTALYTD ( [Total Sales Amount (CM)], 'Date'[Date], "06-30" )
```

In this expression:

- *The third argument is used as the ending date for fiscal year.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Drag and drop **Sales Amount YTD for Fiscal Year (CM)** from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Custom YTD (CM)	Custom YTD using Variables (CM)	Sales Amount YTD using DATESYTD (CM)	Sales Amount YTD using TOTALYTD (CM)	Sales Amount YTD for Fiscal Year (CM)
2017	**$11,928,556.40**	**$8,801,394.26**	**$11,928,556.40**	**$11,928,556.40**	**$11,928,556.40**	**$11,928,556.40**
2018	**$30,516,895.19**	**$8,801,394.26**	**$30,516,895.19**	**$30,516,895.19**	**$30,516,895.19**	**$18,584,558.46**
Jan	$1,327,674.78	$8,801,394.26	$1,327,674.78	$1,327,674.78	$1,327,674.78	$13,256,231.18
Feb	$3,936,463.67	$8,801,394.26	$5,264,138.45	$5,264,138.45	$5,264,138.45	$17,192,694.85
Mar	$700,873.23	$8,801,394.26	$5,965,011.68	$5,965,011.68	$5,965,011.68	$17,893,568.08
Apr	$1,519,275.52	$8,801,394.26	$7,484,287.20	$7,484,287.20	$7,484,287.20	$19,412,843.60
May	$2,960,378.17	$8,801,394.26	$10,444,665.37	$10,444,665.37	$10,444,665.37	$22,373,221.77
Jun	$1,487,671.36	$8,801,394.26	$11,932,336.73	$11,932,336.73	$11,932,336.73	$23,860,893.13
Jul	$2,939,691.10	$8,801,394.26	$14,872,027.83	$14,872,027.83	$14,872,027.83	$2,939,691.10
Aug	$3,964,801.97	$8,801,394.26	$18,836,829.80	$18,836,829.80	$18,836,829.80	$6,904,493.07
Sep	$3,287,606.21	$8,801,394.26	$22,124,436.01	$22,124,436.01	$22,124,436.01	$10,192,099.28
Oct	$2,157,287.80	$8,801,394.26	$24,281,723.81	$24,281,723.81	$24,281,723.81	$12,349,387.08
Nov	$3,611,092.66	$8,801,394.26	$27,892,816.47	$27,892,816.47	$27,892,816.47	$15,960,479.74
Dec	$2,624,078.72	$8,801,394.26	$30,516,895.19	$30,516,895.19	$30,516,895.19	$18,584,558.46
2019	**$42,895,108.14**	**$8,801,394.26**	**$42,895,108.14**	**$42,895,108.14**	**$42,895,108.14**	**$27,409,553.93**
2020	**$24,468,714.27**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**	
Total	**$109,809,274.00**	**$8,801,394.26**	**$24,468,714.27**	**$24,468,714.27**	**$24,468,714.27**	

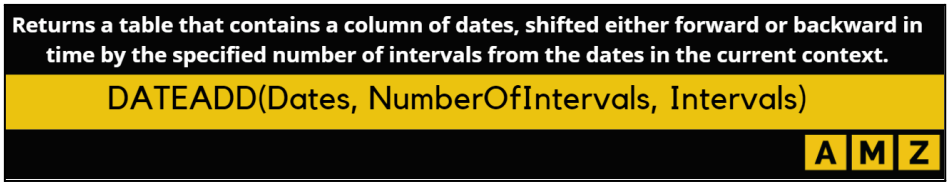
Notice that from July, the aggregation calculation is initialized.

Same Period Last Year

In this exercise, you will learn how to compare sales totals for two consecutive years.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_4.pbix..

Using DATEADD():



DATEADD() returns a table containing the dates. This function is used within calculate to compute an expression, such as total sales, within a given time period.

To use DATEADD():

1. Right click on All Measures in the Fields.
2. Click on New measure.

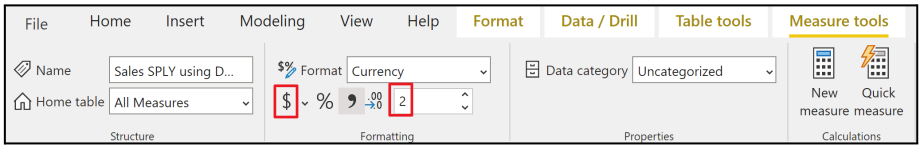
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Sales SPLY using DATEADD (CM) =
CALCULATE ( [Total Sales Amount (CM)], DATEADD ( 'Date'[Date], -1, YEAR ) )
```

In the above expression:

- *A virtual date table containing the dates of the last one year is calculated.*
- *The total sales amount for the dates in rows of the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box



6. Click on the matrix visual in the visualization pane.
7. Drag and drop Year, Month Name and Date column from the Date table to the rows.

   Notice that automatic date hierarchy is created.
8. Drag and drop Total Sales Amount (CM) and Sales SPLY using DATEADD (CM) from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Sales SPLY using DATEADD (CM)
⊟ 2017	$11,928,556.40	
⊞ Jul	$1,423,357.36	
⊞ Aug	$2,057,902.54	
⊞ Sep	$2,523,947.76	
⊞ Oct	$561,681.51	
⊞ Nov	$4,764,920.63	
⊞ Dec	$596,746.60	
⊞ 2018	$30,516,895.19	$11,928,556.40
⊞ 2019	$42,895,108.14	$30,516,895.19
⊞ 2020	$24,468,714.27	$42,895,108.14
Total	$109,809,274.00	$85,340,559.73

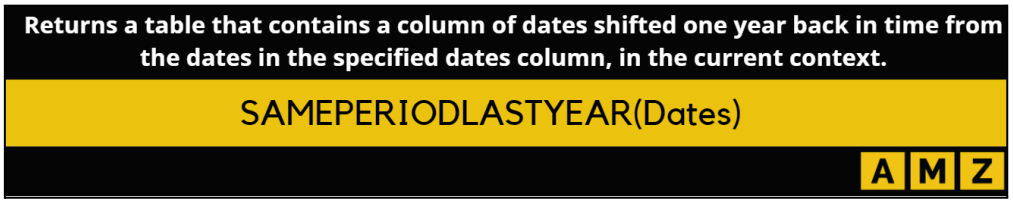
9. Click on "+" next to 2017 in the Year.
10. Click on "+" next to 2018 in the Year.

Year	Total Sales Amount (CM)	Sales SPLY using DATEADD (CM)
⊟ 2017	$11,928,556.40	
⊞ Jul	$1,423,357.36	
⊞ Aug	$2,057,902.54	
⊞ Sep	$2,523,947.76	
⊞ Oct	$561,681.51	
⊞ Nov	$4,764,920.63	
⊞ Dec	$596,746.60	
⊟ 2018	$30,516,895.19	$11,928,556.40
⊞ Jan	$1,327,674.78	
⊞ Feb	$3,936,463.67	
⊞ Mar	$700,873.23	
⊞ Apr	$1,519,275.52	
⊞ May	$2,960,378.17	
⊞ Jun	$1,487,671.36	
⊞ Jul	$2,939,691.10	$1,423,357.36
⊞ Aug	$3,964,801.97	$2,057,902.54
⊞ Sep	$3,287,606.21	$2,523,947.76
⊞ Oct	$2,157,287.80	$561,681.51
⊞ Nov	$3,611,092.66	$4,764,920.63
⊞ Dec	$2,624,078.72	$596,746.60
⊞ 2019	$42,895,108.14	$30,516,895.19
⊞ 2020	$24,468,714.27	$42,895,108.14
Total	$109,809,274.00	$85,340,559.73

Notice that the total sales for July 2017 is reflected as the last year sales in July 2018.

Now let's look at another function which calculates the total value for the same period in the last year. This function is more suitable than DATEADD() and is recommended for comparisons in similar periods.

Using SAMEPERIODLASTYEAR():



To use SAMEPERIODLASTYEAR():

11. Right click on All Measures in the Fields.
12. Click on New measure.

    Measure tools open.

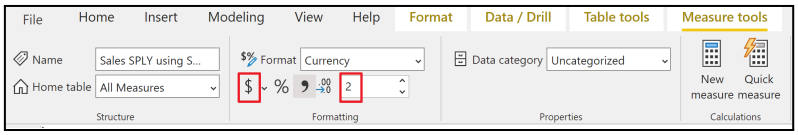
13. Type the following formula in the formula bar and press Enter.

```
Sales SPLY using SAMEPERODLASTYEAR (CM) =
CALCULATE ( [Total Sales Amount (CM)], SAMEPERIODLASTYEAR ( 'Date'[Date] ) )
```

In the above expression:

- *A virtual date table containing the dates from the last one year is calculated.*
- *The total sales amount for the dates in the rows of the virtual table is calculated.*

14. Click on the "$" in the formatting section.
15. Type "2" in the text box.

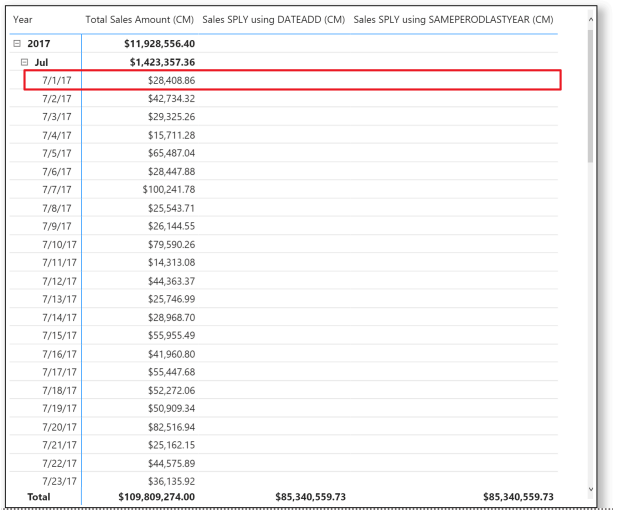


16. Drag and drop Sales SPLY using SAMEPERIODLASTYEAR (CM) from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Sales SPLY using DATEADD (CM)	Sales SPLY using SAMEPERODLASTYEAR (CM)
2017	**$11,928,556.40**		
2018	**$30,516,895.19**	**$11,928,556.40**	**$11,928,556.40**
Jan	$1,327,674.78		
Feb	$3,936,463.67		
Mar	$700,873.23		
Apr	$1,519,275.52		
May	$2,960,378.17		
Jun	$1,487,671.36		
Jul	$2,939,691.10	$1,423,357.36	$1,423,357.36
Aug	$3,964,801.97	$2,057,902.54	$2,057,902.54
Sep	$3,287,606.21	$2,523,947.76	$2,523,947.76
Oct	$2,157,287.80	$561,681.51	$561,681.51
Nov	$3,611,092.66	$4,764,920.63	$4,764,920.63
Dec	$2,624,078.72	$596,746.60	$596,746.60
2019	**$42,895,108.14**	**$30,516,895.19**	**$30,516,895.19**
2020	**$24,468,714.27**	**$42,895,108.14**	**$42,895,108.14**
Total	**$109,809,274.00**	**$85,340,559.73**	**$85,340,559.73**

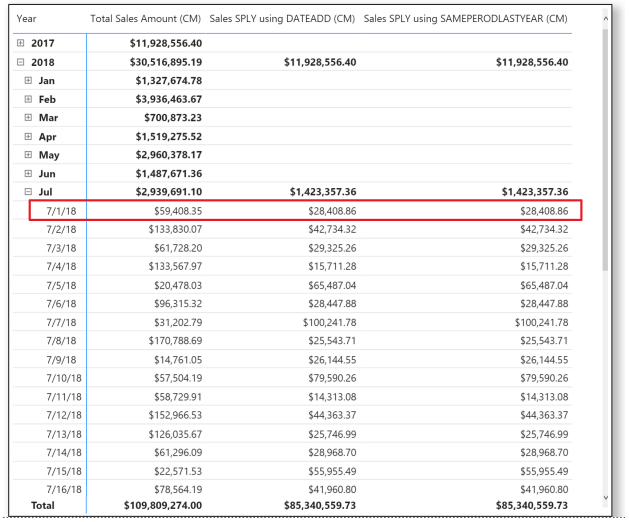
Notice that both of the measures produce same results.

17. Click on “+” next to 2017 in the Year.
18. Click on “+” next to Jul.



Year	Total Sales Amount (CM)	Sales SPLY using DATEADD (CM)	Sales SPLY using SAMEPERODLASTYEAR (CM)
⊟ 2017	$11,928,556.40		
⊟ Jul	$1,423,357.36		
7/1/17	$28,408.86		
7/2/17	$42,734.32		
7/3/17	$29,325.26		
7/4/17	$15,711.28		
7/5/17	$65,487.04		
7/6/17	$28,447.88		
7/7/17	$100,241.78		
7/8/17	$25,543.71		
7/9/17	$26,144.55		
7/10/17	$79,590.26		
7/11/17	$14,313.08		
7/12/17	$44,363.37		
7/13/17	$25,746.99		
7/14/17	$28,968.70		
7/15/17	$55,955.49		
7/16/17	$41,960.80		
7/17/17	$55,447.68		
7/18/17	$52,272.06		
7/19/17	$50,909.34		
7/20/17	$82,516.94		
7/21/17	$25,162.15		
7/22/17	$44,575.89		
7/23/17	$36,135.92		
Total	$109,809,274.00	$85,340,559.73	$85,340,559.73

19. Click on “+” next to 2018 in the Year.
20. Click on “+” next to Jul.



Year	Total Sales Amount (CM)	Sales SPLY using DATEADD (CM)	Sales SPLY using SAMEPERODLASTYEAR (CM)
⊞ 2017	$11,928,556.40		
⊟ 2018	$30,516,895.19	$11,928,556.40	$11,928,556.40
⊞ Jan	$1,327,674.78		
⊞ Feb	$3,936,463.67		
⊞ Mar	$700,873.23		
⊞ Apr	$1,519,275.52		
⊞ May	$2,960,378.17		
⊞ Jun	$1,487,671.36		
⊟ Jul	$2,939,691.10	$1,423,357.36	$1,423,357.36
7/1/18	$59,408.35	$28,408.86	$28,408.86
7/2/18	$133,830.07	$42,734.32	$42,734.32
7/3/18	$61,728.20	$29,325.26	$29,325.26
7/4/18	$133,567.97	$15,711.28	$15,711.28
7/5/18	$20,478.03	$65,487.04	$65,487.04
7/6/18	$96,315.32	$28,447.88	$28,447.88
7/7/18	$31,202.79	$100,241.78	$100,241.78
7/8/18	$170,788.69	$25,543.71	$25,543.71
7/9/18	$14,761.05	$26,144.55	$26,144.55
7/10/18	$57,504.19	$79,590.26	$79,590.26
7/11/18	$58,729.91	$14,313.08	$14,313.08
7/12/18	$152,966.53	$44,363.37	$44,363.37
7/13/18	$126,035.67	$25,746.99	$25,746.99
7/14/18	$61,296.09	$28,968.70	$28,968.70
7/15/18	$22,571.53	$55,955.49	$55,955.49
7/16/18	$78,564.19	$41,960.80	$41,960.80
Total	$109,809,274.00	$85,340,559.73	$85,340,559.73

Notice that all the measures show value to the day level granularity.

Finding YOY %

In this exercise, you will learn how to find the year over year change in Sales value.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_5.pbix..

To find YOY%:

1. Right click on All Measures in the Fields.
2. Click on New measure.

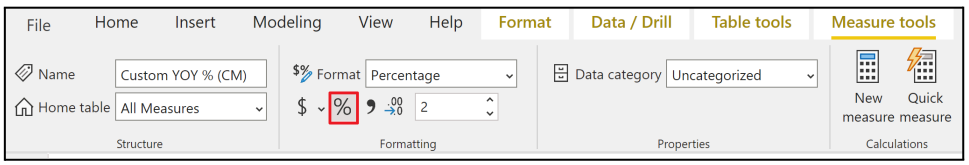
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Custom YOY % (CM) =
VAR CurrentSales = [Total Sales Amount (CM)]
VAR PreviousSales = [Sales SPLY using SAMEPERODLASTYEAR (CM)]
VAR DeltaSales = CurrentSales - PreviousSales
VAR Result =
    IF ( NOT ISBLANK ( CurrentSales ), DIVIDE ( DeltaSales, PreviousSales ) )
RETURN
    Result
```

In this expression:

- *The sum of the Sales Amount current filter context is found and stored in CurrentSales variable.*
- *The sales for the last year are stored in PreviousSales.*
- *The variable DeltaSales is used to find the difference between the values of the first two variables.*
- *Finally, at the end the DeltaSales is divided by the PreviousSales to get the change.*

4. Click on the "%" in the formatting section.



5. Click on the matrix visual in the visualization pane.
6. Drag and drop Year, Month Name and Date column from the Date table to the rows.

   Notice that automatic date hierarchy is created.
7. Drag and drop Total Sales Amount (CM) and Custom YOY % (CM) from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Custom YOY % (CM)
⊞ 2017	$11,928,556.40	
⊞ 2018	$30,516,895.19	155.83 %
⊞ 2019	$42,895,108.14	40.56 %
⊞ 2020	$24,468,714.27	-42.96 %
Total	$109,809,274.00	28.67 %

8. Click on "+" next to 2018 in the Year.
9. Click on "+" next to 2019 in the Year.

Year	Total Sales Amount (CM)	Custom YOY % (CM)
⊞ 2017	$11,928,556.40	
⊟ 2018	$30,516,895.19	155.83 %
⊞ Jan	$1,327,674.78	
⊞ Feb	$3,936,463.67	
⊞ Mar	$700,873.23	
⊞ Apr	$1,519,275.52	
⊞ May	$2,960,378.17	
⊞ Jun	$1,487,671.36	
⊞ Jul	$2,939,691.10	106.53 %
⊞ Aug	$3,964,801.97	92.66 %
⊞ Sep	$3,287,606.21	30.26 %
⊞ Oct	$2,157,287.80	284.08 %
⊞ Nov	$3,611,092.66	-24.22 %
⊞ Dec	$2,624,078.72	339.73 %
⊟ 2019	$42,895,108.14	40.56 %
⊞ Jan	$1,847,692.08	39.17 %
⊞ Feb	$2,829,361.72	-28.12 %
⊞ Mar	$2,092,434.59	198.55 %
⊞ Apr	$2,405,971.41	58.36 %

Let's verify the value:

$$YOY\% = \frac{1847692.08 - 1327674.78}{1327674.78} = \frac{520017.30}{1327674.78} \times 100 = 39.17\%$$

Quick Measures Option:

Power BI provides you with the Quick Measures option in which Power BI automatically generates the code for you.

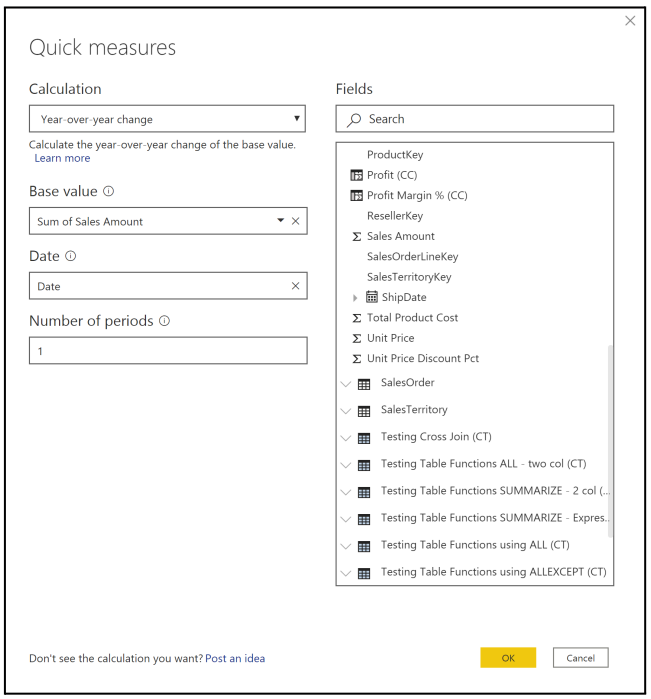
10. Right click on All Measures in the Fields.
11. Click on New quick measure.

    Pop-up menu appears.

12. From the Calculation menu, select year over year change.
13. Drag and drop Sales Amount from the Sales table to Base value.
14. Drag and drop Date from the Date table to Date selection box.

15. Type 1 in the Number of Periods.

Please note that "1" will be default option as well.



16. Click Ok.

17. Drag and drop Sales Amount YoY% to the Values.

Year	Total Sales Amount (CM)	Custom YOY % (CM)	Sales Amount YoY%
2017	$11,928,556.40		
2018	$30,516,895.19	155.83 %	155.83 %
2019	$42,895,108.14	40.56 %	40.56 %
2020	$24,468,714.27	-42.96 %	-42.96 %
Total	$109,809,274.00	28.67 %	28.67 %

Notice that similar results are generated by both measures.

18. Click on Sales Amount YoY % in the All Measures table.

```
Sales Amount YoY% =
VAR __PREV_YEAR = CALCULATE(SUM('Sales'[Sales Amount]), DATEADD('Date'[Date], -1, YEAR))
RETURN
    DIVIDE(SUM('Sales'[Sales Amount]) - __PREV_YEAR, __PREV_YEAR)
```

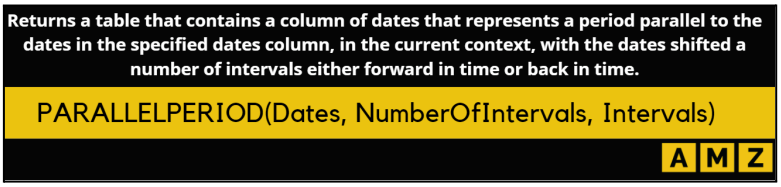
The code for YoY% internally calls DATEADD().

Notice how descriptive variable names can make understanding the code much easier.

Using PARALLELPERIOD()

PARALLELPERIOD() is used to generate a table of dates over which further calculations are performed.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_6.pbix..



To use PARALLELPERIOD() to find Sales compared to last year:

1. Right click on All Measures in the Fields.
2. Click on New measure.

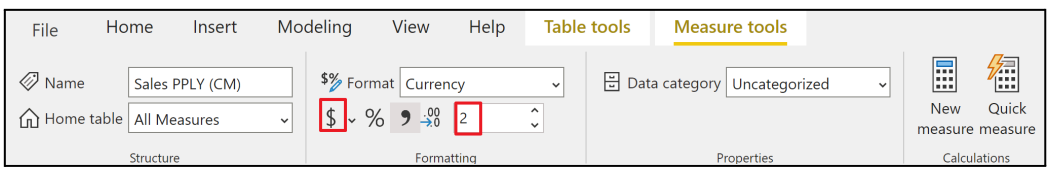
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Sales PPLY (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    PARALLELPERIOD ( 'Date'[Date], -1, YEAR )
)
```

In the above expression:

- *A virtual date table containing the dates for one year is calculated.*
- *The total sales amount for the dates in the rows of the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Click on the matrix visual in the visualization pane.
7. Drag and drop Year, Month Name and Date column from the Date table to the rows.

   Notice that automatic date hierarchy is created.

8. Drag and drop Total Sales Amount (CM) and Sales PPLY (CM) from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Sales PPLY (CM)
⊞ 2017	$11,928,556.40	
⊞ 2018	$30,516,895.19	$11,928,556.40
⊞ 2019	$42,895,108.14	$30,516,895.19
⊞ 2020	$24,468,714.27	$42,895,108.14
Total	$109,809,274.00	$85,340,559.73

Notice the Total Sales Amount for 2017 is reflected in Sales PPLY (CM) for 2018.

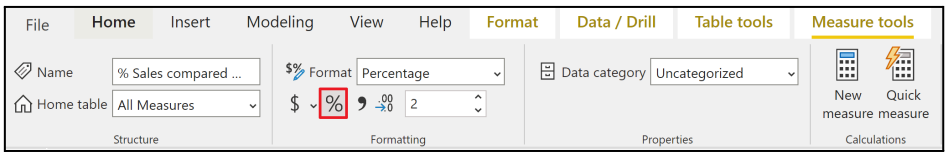
Let's use this value for sales comparison.

9. Right click on All Measures in the Fields.
10. Click on New measure.

    Measure tools open.
11. Type the following formula in the formula bar and press Enter.

```
% Sales compared to last year (CM) =
DIVIDE ( [Sales Amount YTD using TOTALYTD (CM)], [Sales PPLY (CM)] )
```

12. Click on the "%" in the formatting section.



13. Drag and drop % Sales compared to last year (CM) from the All Measures table to the Values.
14. Click on "+" next to 2019 in the Year.

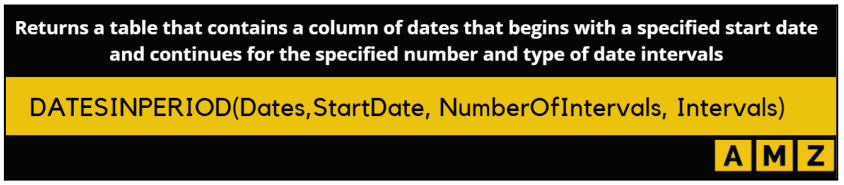
Year	Total Sales Amount (CM)	Sales PPLY (CM)	% Sales compared to last year (CM)
⊞ 2017	$11,928,556.40		
⊞ 2018	$30,516,895.19	$11,928,556.40	255.83 %
⊟ 2019	$42,895,108.14	$30,516,895.19	140.56 %
⊞ Jan	$1,847,692.08	$30,516,895.19	6.05 %
⊞ Feb	$2,829,361.72	$30,516,895.19	15.33 %
⊞ Mar	$2,092,434.59	$30,516,895.19	22.18 %
⊞ Apr	$2,405,971.41	$30,516,895.19	30.07 %
⊞ May	$3,459,444.49	$30,516,895.19	41.40 %
⊞ Jun	$2,850,649.92	$30,516,895.19	50.74 %
⊞ Jul	$3,513,063.74	$30,516,895.19	62.26 %
⊞ Aug	$5,247,164.93	$30,516,895.19	79.45 %
⊞ Sep	$5,104,087.65	$30,516,895.19	96.18 %
⊞ Oct	$3,542,149.22	$30,516,895.19	107.78 %
⊞ Nov	$5,151,895.52	$30,516,895.19	124.67 %
⊞ Dec	$4,851,192.87	$30,516,895.19	140.56 %
⊞ 2020	$24,468,714.27	$42,895,108.14	57.04 %
Total	$109,809,274.00	$85,340,559.73	28.67 %

Notice a constant value throughout all months of the year. DATEADD() can work on the granularity of day as well as month, quarter and year. Whereas, PARALLELPERIOD() cannot work on the granularity of day, instead it works on month, quarter and year only according to the argument provided in the function call.

Finding Moving Total

In this exercise, you will learn how to calculate the moving total for the last 1 year. It is the total sales for the last 1 year starting from an incoming filter context.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_7.pbix..



To use DATESINPERIOD() to find Sales for the last year:

1. Right click on All Measures in the Fields.
2. Click on New measure.

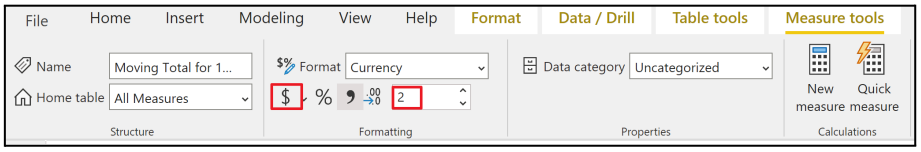
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Moving Total for 12 Months (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    DATESINPERIOD ( 'Date'[Date], MAX ( 'Date'[Date] ), -1, YEAR )
)
```

In the above expression:

- *The MAX() function finds the maximum date from the incoming filter context.*
- *DATESINPERIOD() creates a virtual table containing dates for 1 year from the maximum date.*
- *The total sales for all the dates in the virtual table is calculated.*

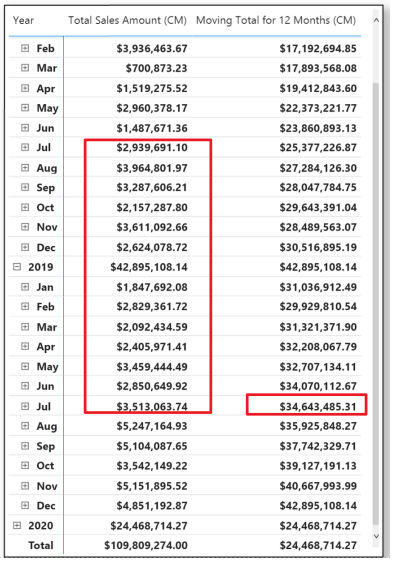
4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Click on the matrix visual in the visualization pane.
7. Drag and drop Year, Month Name and Date column from the Date table to the rows.

   Notice that automatic date hierarchy is created.

8. Drag and drop Total Sales Amount (CM) and Moving Total for 12 Months (CM) from the All Measures table to the Values.



Year	Total Sales Amount (CM)	Moving Total for 12 Months (CM)
⊞ Feb	$3,936,463.67	$17,192,694.85
⊞ Mar	$700,873.23	$17,893,568.08
⊞ Apr	$1,519,275.52	$19,412,843.60
⊞ May	$2,960,378.17	$22,373,221.77
⊞ Jun	$1,487,671.36	$23,860,893.13
⊞ Jul	$2,939,691.10	$25,377,226.87
⊞ Aug	$3,964,801.97	$27,284,126.30
⊞ Sep	$3,287,606.21	$28,047,784.75
⊞ Oct	$2,157,287.80	$29,643,391.04
⊞ Nov	$3,611,092.66	$28,489,563.07
⊞ Dec	$2,624,078.72	$30,516,895.19
⊟ 2019	$42,895,108.14	$42,895,108.14
⊞ Jan	$1,847,692.08	$31,036,912.49
⊞ Feb	$2,829,361.72	$29,929,810.54
⊞ Mar	$2,092,434.59	$31,321,371.90
⊞ Apr	$2,405,971.41	$32,208,067.79
⊞ May	$3,459,444.49	$32,707,134.11
⊞ Jun	$2,850,649.92	$34,070,112.67
⊞ Jul	$3,513,063.74	$34,643,485.31
⊞ Aug	$5,247,164.93	$35,925,848.27
⊞ Sep	$5,104,087.65	$37,742,329.71
⊞ Oct	$3,542,149.22	$39,127,191.13
⊞ Nov	$5,151,895.52	$40,667,993.99
⊞ Dec	$4,851,192.87	$42,895,108.14
⊞ 2020	$24,468,714.27	$24,468,714.27
Total	$109,809,274.00	$24,468,714.27

Notice that the summation of all values from July 2018 to July 2019 is $34,643,485.31

DATESBETWEEN() and DATESINPERIOD() both generate a table of dates. However, DATESBETWEEN() generates a tables between a starting date and an ending date. DATESINPERIOD() gives dates in a particular staring from a particular date.

Calculate Running Total

Running total calculates the summation of Sales amount to date starting from the date of first transaction recorded in the system.

Continue using the previous file for the exercise or open the file Time_Intellegence_In_DAX_8.pbix..

To use Running Total:

1. Right click on All Measures in the Fields.
2. Click on New measure.

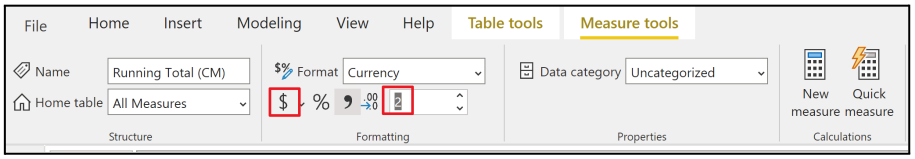
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Running Total (CM) =
VAR LastVisibleDate =
    MAX ( 'Date'[Date] )
RETURN
    CALCULATE (
        [Total Sales Amount (CM)],
        FILTER ( ALL ( 'Date' ), 'Date'[Date] <= LastVisibleDate )
    )
```

In the above expression:

- *The LastVisibleDate variable stores the maximum date from the incoming filter context.*
- *The FILTER() creates a virtual table where all the dates less than the LastVisibleDate are stored.*
- *The total sales amount over the dates in the virtual table is calculated.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Drag and drop Running Total (CM) from the All Measures table to the Values.

Year	Total Sales Amount (CM)	Moving Total for 12 Months (CM)	Running Total (CM)
2017	$11,928,556.40	$11,928,556.40	$11,928,556.40
Jul	$1,423,357.36	$1,423,357.36	$1,423,357.36
Aug	$2,057,902.54	$3,481,259.90	$3,481,259.90
Sep	$2,523,947.76	$6,005,207.66	$6,005,207.66
Oct	$561,681.51	$6,566,889.17	$6,566,889.17
Nov	$4,764,920.63	$11,331,809.80	$11,331,809.80
Dec	$596,746.60	$11,928,556.40	$11,928,556.40
2018	$30,516,895.19	$30,516,895.19	$42,445,451.59
2019	$42,895,108.14	$42,895,108.14	$85,340,559.73
2020	$24,468,714.27	$24,468,714.27	$109,809,274.00
Total	$109,809,274.00	$24,468,714.27	$109,809,274.00

Notice that the value is continuously increasing.

Chapter 13: Conditionals in DAX

What will you learn in this chapter?

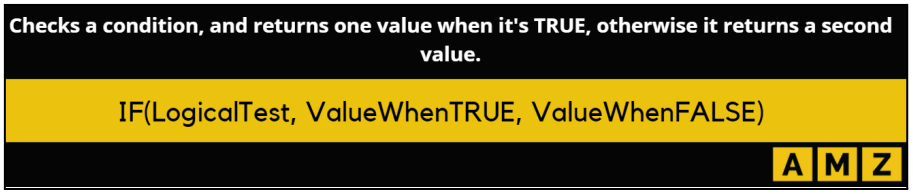
- ✔ Use of IF() statements
- ✔ Using SWITCH
- ✔ Switching between cases using SWITCH()
- ✔ Finding your text

Use of IF() statements

IF() is the most commonly used function to perform conditional calculations in DAX.

Continue using the previous file for the exercise or open the file Conditionals_In_DAX_1.pbix..

IF() performs a Logical Test and the result is either TRUE or FALSE. If the result of logical evaluation is TRUE, then the second argument of IF() is returned otherwise the third argument is returned.



In this exercise, you will learn how to use IF() to calculate the Profit Group. For this you will need to enrich your data with some calculated columns.

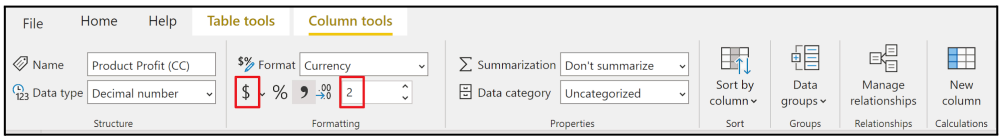
Let's first find Product Profit:

1. Right click on the **Product** table in the **Fields**.
2. Click on **New Column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Product Profit (CC) =
'Product'[List Price] - 'Product'[Standard Cost]
```

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



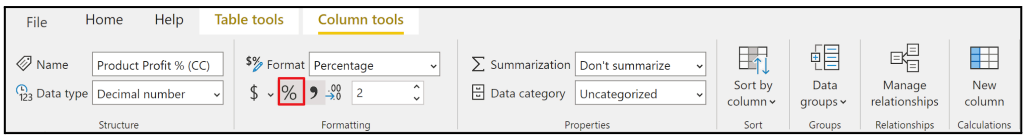
Now let's find Product Profit % (CC):

6. Right click on the **Product** table in the **Fields**.
7. Click on **New Column**.

   Column tools open.
8. Type the following formula in the formula bar and press Enter.

```
Product Profit % (CC) =
'Product'[Product Profit (CC)] / 'Product'[List Price]
```

9. Click on the "%" in the formatting section.



To use IF() for calculating profit group:

10. Right click on the **Product** table in the **Fields**.
11. Click on **New Column**.

    Column tools open.

12. Type the following formula in the formula bar and press Enter.

```
Profit group (CC) =
IF (
    'Product'[Product Profit % (CC)] > 0.5,
    "High",
    IF ( 'Product'[Product Profit % (CC)] > 0.3, "Medium", "Low" )
)
```

In this DAX expression:

- *The value for the Product Profit % (CC) is tested for each row.*
- *If the value is greater than 0.5 then the profit group column in that row is assigned a value of "High".*
- *Otherwise, the inner IF() checks if the Product Profit % (CC) is greater than 0.3, the profit group is categorized as "Medium" else its "Low"*

13. Click on **Data** tab in the left **Ribbon**.
14. Click on **Product** table in the **Fields**.

ProductKey	Product	Standard Cost	Color	List Price	Model	Subcategory	Category	SKU	Product Profit (CC)	Product Profit % (CC)	Profit group (CC)
335	Road-650 Black, 60	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-60	$296.28	37.84 %	Medium
339	Road-650 Black, 44	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-44	$296.28	37.84 %	Medium
341	Road-650 Black, 48	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-48	$296.28	37.84 %	Medium
343	Road-650 Black, 52	$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-52	$296.28	37.84 %	Medium
255	LL Road Frame - Black, 58	$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-58	$132.59	39.32 %	Medium
258	LL Road Frame - Black, 60	$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-60	$132.59	39.32 %	Medium
261	LL Road Frame - Black, 62	$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-62	$132.59	39.32 %	Medium
281	LL Road Frame - Black, 44	$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-44	$132.59	39.32 %	Medium
284	LL Road Frame - Black, 48	$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-48	$132.59	39.32 %	Medium
287	LL Road Frame - Black, 52	$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-52	$132.59	39.32 %	Medium
440	HL Road Frame - Black, 44	$868.63	Black	$1,431.5	HL Road Frame	Road Frames	Components	FR-R92B-44	$562.87	39.32 %	Medium
210	HL Road Frame - Black, 58	$868.63	Black	$1,431.5	HL Road Frame	Road Frames	Components	FR-R92B-58	$562.87	39.32 %	Medium
438	HL Road Frame - Black, 62	$868.63	Black	$1,431.5	HL Road Frame	Road Frames	Components	FR-R92B-62	$562.87	39.32 %	Medium

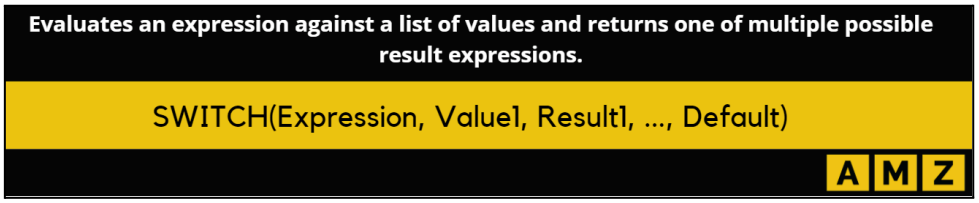
A calculated column with each row containing the respective Profit Group has been created.

Using SWITCH()

SWITCH() is used for conditional calculations when the result is evaluated against multiple values or expressions.

Continue using the previous file for the exercise or open the file Conditionals_In_DAX_2.pbix..

Lets use SWITCH() to categorize products into four categories:



To use SWITCH () for calculating profit group:

1. Right click on the **Product** table in the **Fields**.
2. Click on **New Column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

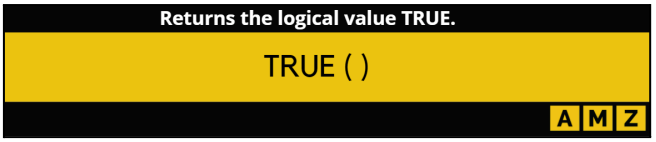
```
Profit group - Switch (CC) =
SWITCH (
    TRUE (),
    'Product'[Product Profit % (CC)] > 0.6, "V.High",
    'Product'[Product Profit % (CC)] > 0.5, "High",
    'Product'[Product Profit % (CC)] > 0.4, "Medium",
    'Product'[Product Profit % (CC)] > 0.3, "Low",
    "V.Low"
)
```

In the expression above:

- *In the second argument, Product Profit % (CC) is tested, if it is greater than 0.6 than the profit group is categorized as "V.High".*
- *If the condition in the case above is not satisfied then the next case is tested, i.e., Product Profit % (CC) greater than 0.5. If this returns TRUE then profit group is categorized as "High".*
- *The same logic is repeated for all cases.*
- *If none of the condition is satisfied, the default result is "V.Low".*

Since you have created a column containing text only, you will need a numerical column to sort this alphabetically.

Notice the use of TRUE():



4. Right click on the **Product** table in the **Fields**.
5. Click on **New Column**.

   Column tools open.
6. Type the following formula in the formula bar and press Enter.

```
Profit group-Sort Order (CC) =
SWITCH (
    TRUE (),
    'Product'[Product Profit % (CC)] > 0.6, 1,
    'Product'[Product Profit % (CC)] > 0.5, 2,
    'Product'[Product Profit % (CC)] > 0.4, 3,
    'Product'[Product Profit % (CC)] > 0.3, 4,
    5
)
```

In the expression above:

- *The first argument in SWITCH() is TRUE() it always returns TRUE.*
- *In the second argument, Product Profit % (CC) is tested, if it is greater than 0.6 than the profit group sorting order is categorized as "1".*
- *If the condition in the case above is not satisfied then the next case is tested, i.e., Product Profit % (CC) greater than 0.5. If this returns TRUE then profit group sorting order is categorized as "2".*
- *The same logic is repeated for all cases.*
- *If none of the condition is satisfied, the default result is "5".*

7. Click on **Data** tab in the left **Ribbon**.
8. Click on **Product** table in the **Fields**.

Standard Cost	Color	List Price	Model	Subcategory	Category	SKU	Product Profit (CC)	Product Profit % (CC)	Profit group (CC)	Profit group - Switch (CC)	Profit group-Sort Order (CC)
$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-48	$888.40	36.36 %	Medium	Low	4
$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-52	$888.40	36.36 %	Medium	Low	4
$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-58	$888.40	36.36 %	Medium	Low	4
$1,554.95	Black	$2,443.35	Road-250	Road Bikes	Bikes	BK-R89B-44	$888.40	36.36 %	Medium	Low	4
$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-52	$196.34	36.36 %	Medium	Low	4
$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-44	$196.34	36.36 %	Medium	Low	4
$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-48	$196.34	36.36 %	Medium	Low	4
$343.65	Black	$539.99	Road-750	Road Bikes	Bikes	BK-R19B-58	$196.34	36.36 %	Medium	Low	4
$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-62	$296.28	37.84 %	Medium	Low	4
$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-58	$296.28	37.84 %	Medium	Low	4
$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-60	$296.28	37.84 %	Medium	Low	4
$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-44	$296.28	37.84 %	Medium	Low	4
$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-48	$296.28	37.84 %	Medium	Low	4
$486.71	Black	$782.99	Road-650	Road Bikes	Bikes	BK-R50B-52	$296.28	37.84 %	Medium	Low	4
$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-58	$132.59	39.32 %	Medium	Low	4
$204.63	Black	$337.22	LL Road Frame	Road Frames	Components	FR-R38B-60	$132.59	39.32 %	Medium	Low	4

Now that you have created all the required columns, lets sort them.

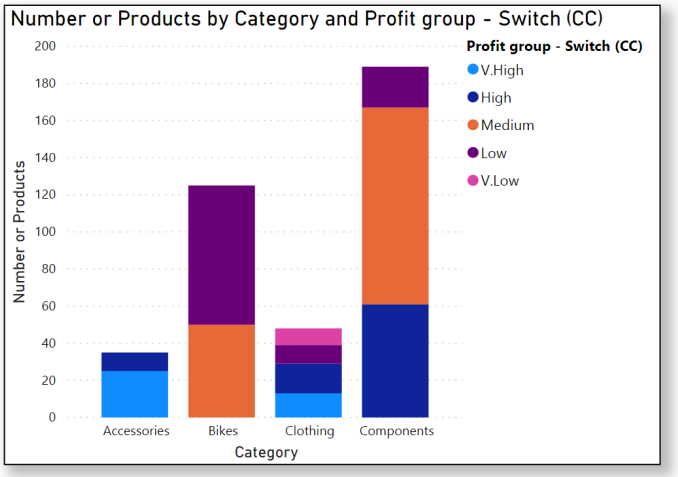
9. Click on Profit group-SWITCH (CC) in the Product table.

   Column tools open.

10. Click on Sort by column.
11. Click on Profit group Sort Order (CC).

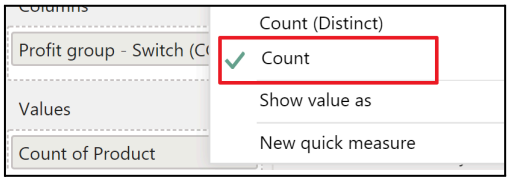
Now let's visualize the results.

12. Click on **Stacked Column chart** in the **Visualization Pane.**
13. Drag and drop **Category** from the Product table to the Axis.
14. Drag and drop **Product** from the Product table to the Values.
15. Rename this to "Number of Products"
16. Drag and drop **Profit Group-SWITCH (CC)** from the Product table to the Legend.
17. Click on the three dots on the top.



Notice that the Accessories contain the highest proportion of "V.High" products and only Clothing contains "V.Low" products. Let's see the numerical stats for this analysis.

18. Click on the **matrix** visual in the **visualization pane**.
19. Drag and drop **Category** from the Products table to the rows.
20. Drag and drop **Product** from Product table to the Values.
21. Drag and drop **Profit Group-SWITCH (CC)** from the Product table to the Column.
22. Right click on **Product** and select Count.



Notice that each cell of the matrix has an incoming filter context from rows as well as the columns.

Category	V.High	High	Medium	Low	V.Low	**Total**
Accessories	25	10				**35**
Clothing	13	16		10	9	**48**
Bikes			50	75		**125**
Components		61	106	22		**189**
Total	**38**	**87**	**156**	**107**	**9**	**397**

Switching between cases using SWITCH()

SWITCH() can also be used to calculate different cases.

Continue using the previous file for the exercise or open the file Conditionals_In_DAX_3.pbix..

To understand how to use SWITCH() for switching between cases:

1. Right click on the **Product** table in the **Fields**.
2. Click on **New Column**.

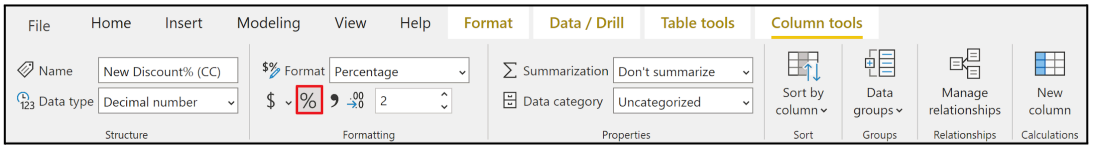
   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
New Discount% (CC) =
SWITCH (
    TRUE (),
    'Product'[Color] = "Red", 0.50,
    AND ( 'Product'[Profit group (CC)] = "Low", 'Product'[Color] = "Yellow" ), 0.40,
    0
)
```

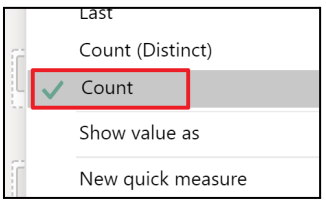
In the expression above:

- *In the first case, product color is tested and if it is red then a value of 0.5 is assigned.*
- *In the second case, the existence of two conditions simultaneously is evaluated using AND() and if both of these are satisfied then a value of 0.40 is assigned.*
- *Otherwise the value is 0.*

4. Click on the "%" in the formatting section.



5. Click on the **matrix** visual in the **visualization pane**.
6. Drag and drop **Category** from the Products table to the rows.
7. Drag and drop **Product** from Product table to the Values.
8. Drag and drop **New Discount % (CC)** from the Product table to the Column.
9. Right click on **Product** and select Count.



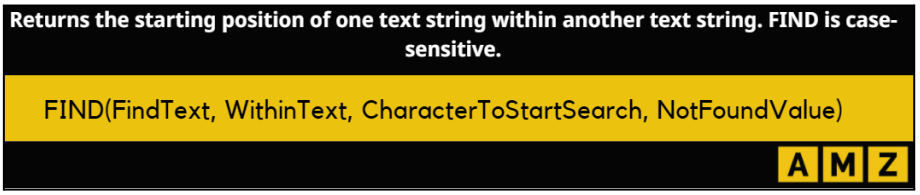
Notice that the New Discount % campaign is focused on products from the category of Bikes and Components and no products from Clothing qualifies the set criteria.

Category	0.00 %	40.00 %	50.00 %	**Total**
Accessories	32		3	**35**
Bikes	98		27	**125**
Clothing	44	4		**48**
Components	156		33	**189**
Total	**330**	**4**	**63**	**397**

Finding your text

FIND() checks the presence of a string inside another string column and returns its starting position. FIND is case-sensitive

Continue using the previous file for the exercise or open the file Conditionals_In_DAX_4.pbix..



To see how FIND() works:

1. Right click on the **Product** table in the **Fields**.
2. Click on **New Column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Finding Road Products (CC) =
FIND ( "Road", 'Product'[Model], 1, 0 )
```

In the expression above:

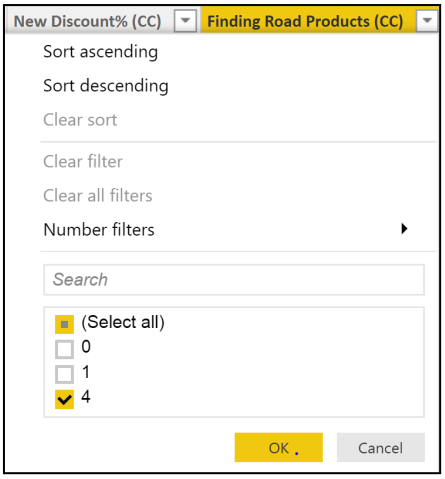
- *The word "Road" is searched in the Model Column from the Product table.*

4. Click on **Data** tab in the left **Ribbon**.
5. Click on **Product** table in the **Fields**.

A calculated column with each row containing the position of the text 'Road' in Model column of the Product table has been created.

6. Click on the drop down at the Finding Road Products (CC) column.

   Notice that multiple values exist in this column.
7. Click on 4.



Notice that 4 indicates that the Road is present at the 4th position in the text string.

st	Color	List Price	Model	Subcategory	Category	SKU	Product Profit (CC)	Product Profit % (CC)	Profit group (CC)	Profit group - Switch (CC)	Profit group-S	New Discou	Finding Road
$12.19	NA	$32.6	HL Road Tire	Tires and Tubes	Accessories	TI-R982	$20.41	62.60 %	High	V.High	1	0.00 %	4
$9.35	NA	$24.99	ML Road Tire	Tires and Tubes	Accessories	TI-R628	$15.64	62.60 %	High	V.High	1	0.00 %	4
$8.04	NA	$21.49	LL Road Tire	Tires and Tubes	Accessories	TI-R092	$13.45	62.60 %	High	V.High	1	0.00 %	4
$48.55	NA	$109.3364	HL Road Handlebars	Handlebars	Components	HB-R956	$60.79	55.60 %	High	High	2	0.00 %	4
$53.40	NA	$120.27	HL Road Handlebars	Handlebars	Components	HB-R956	$66.87	55.60 %	High	High	2	0.00 %	4
$27.49	NA	$61.92	ML Road Handlebars	Handlebars	Components	HB-R720	$34.43	55.60 %	High	High	2	0.00 %	4
$35.96	Silver/Blac	$80.99	HL Road Pedal	Pedals	Components	PD-R853	$45.03	55.60 %	High	High	2	0.00 %	4
$27.57	Silver/Blac	$62.09	ML Road Pedal	Pedals	Components	PD-R563	$34.52	55.60 %	High	High	2	0.00 %	4
$24.99	NA	$56.2909	ML Road Handlebars	Handlebars	Components	HB-R720	$31.30	55.60 %	High	High	2	0.00 %	4
$12.04	NA	$27.12	LL Road Seat/Saddle	Saddles	Components	SE-R581	$15.08	55.60 %	High	High	2	0.00 %	4
$23.37	NA	$52.64	HL Road Seat/Saddle	Saddles	Components	SE-R995	$29.27	55.60 %	High	High	2	0.00 %	4
$19.78	NA	$44.54	LL Road Handlebars	Handlebars	Components	HB-R504	$24.76	55.60 %	High	High	2	0.00 %	4
$17.98	Silver/Blac	$40.49	LL Road Pedal	Pedals	Components	PD-R347	$22.51	55.60 %	High	High	2	0.00 %	4
$17.98	NA	$40.4909	LL Road Handlebars	Handlebars	Components	HB-R504	$22.51	55.60 %	High	High	2	0.00 %	4
$17.38	NA	$39.14	ML Road Seat/Saddl	Saddles	Components	SE-R908	$21.76	55.60 %	High	High	2	0.00 %	4

As mentioned before, FIND() is case sensitive so if you change "Road" to either "road" or "ROAD", nothing will be found and the resulting column will show 0

Chapter 14: Practical Use of Concepts and Other Functions

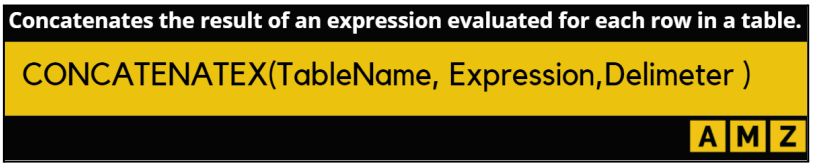
What will you learn in this chapter?

- ✔ Using CONCATENATEX()
- ✔ CONCATENATEX() with HASONEVALUE()
- ✔ Using ISINSCOPE()
- ✔ Parameter table and SELECTEDVALUE()
- ✔ Using RANKX()
- ✔ ISINSCOPE() with RANKX()
- ✔ Counting spikes

Using CONCATENATEX()

In this exercise, you will learn how to find the maximum daily sales. Each day can have multiple sales and there can be many days with the same maximum sales.

Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_1.pbix..



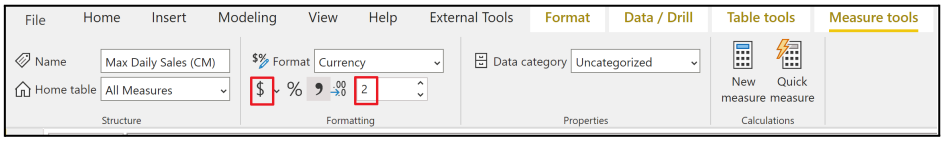
To find the dates with the maximum daily sales:

1. Right click on the **All Measures** table in the **Fields** pan.
2. Click on **New measure**.

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Max Daily Sales (CM) =
MAXX ( 'Date', [Total Sales Amount (CM)] )
```

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.



6. Right click on the **All Measures** table in the **Fields** pan.
7. Click on **New measure**.

   Measure tools open.
8. Type the following formula in the formula bar and press Enter.

```
Date of Max Sales (CM) =
VAR MaxDailySales = [Max Daily Sales (CM)]
VAR DatesWithMaxSales =
    FILTER ( 'Date', [Total Sales Amount (CM)] = MaxDailySales )
VAR Result =
    CONCATENATEX ( DatesWithMaxSales, FORMAT ( 'Date'[Date], "mm/dd/yy" ), ", " )
RETURN
    IF ( NOT ISBLANK ( MaxDailySales ), Result, BLANK () )
```

In the expression above:

- *MaxDialySales finds the maximum daily sales using the measure.*
- *DatesWithMaxSales finds the date when the maximum sales were recorded.*

- *CONCATENATEX() has been used to concatenate the list of dates containing highest sales in the current filter context after evaluation.*

9. Click on the **matrix** visual in the **visualization pane**.
10. Drag and drop **Year, Qtr, Full Month** and **Date** from the Date table to the rows.
    A hierarchy will be detected by Power BI.
11. Drag and drop **Total Sales Amount (CM), Max Daily Sales (CM)** and **Date of Max Sales (CM)** from All Measures table to the Values.

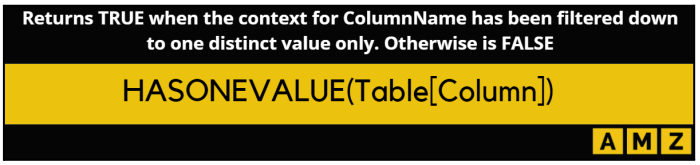
Year	Total Sales Amount (CM)	Max Daily Sales (CM)	Date of Max Sales (CM)
2017	**$11,928,556.40**	**$276,783.36**	**11/26/17**
Q3	**$6,005,207.66**	**$220,081.82**	**09/12/17**
2017 Jul	$1,423,357.36	$116,318.97	07/29/17
2017 Aug	$2,057,902.54	$181,994.51	08/13/17
2017 Sep	$2,523,947.76	$220,081.82	09/12/17
Q4	**$5,923,348.74**	**$276,783.36**	**11/26/17**
2017 Oct	$561,681.51	$28,740.42	10/14/17
2017 Nov	$4,764,920.63	$276,783.36	11/26/17
2017 Dec	$596,746.60	$31,669.59	12/14/17
2018	**$30,516,895.19**	**$374,697.42**	**09/14/18**
2019	**$42,895,108.14**	**$456,533.50**	**07/18/19**
2020	**$24,468,714.27**	**$569,749.73**	**06/06/20**
Total	**$109,809,274.00**	**$569,749.73**	**06/06/20**

CONCATENATEX() with HASONEVALUE()

In this exercise, you will solve the business problem of listing the duplicate keys of a product along with the list price using CONCATENATEX() with HASONEVALUE().

Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_2.pbix..

HASONEVALUE() checks if context of the column name contains a single value.



To find the duplicate keys of products and duplicate list price:

1. Right click on the **All Measures** table in the **Fields** pan.
2. Click on **New measure**.

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
No of Products (CM) =
COUNT ( 'Product'[ProductKey] )
```

4. Right click on the **All Measures** table in the **Fields** pan.
5. Click on **New measure**.

   Measure tools open.
6. Type the following formula in the formula bar and press Enter.

```
Product Keys Of Duplicate Products (CM) =
IF (
    HASONEVALUE ( 'Product'[Product] ),
    CONCATENATEX ( 'Product', 'Product'[ProductKey], "," )
)
```

The expression is evaluated for each product and then the Product Key for each product is listed.

7. Right click on the **All Measures** table in the **Fields** pan.
8. Click on **New measure**.

   Measure tools open.
9. Type the following formula in the formula bar and press Enter.

```
Product List Price of Duplicate Products (CM) =
IF (
    HASONEVALUE ( 'Product'[Product] ),
    CONCATENATEX ( 'Product', ROUND ( 'Product'[List Price], 1 ), ", " )
)
```

The expression is evaluated for each product and then the Product List Price for each product is listed.

10. Click on the **table** visual in the **visualization pane**.
11. Drag and drop **Product** and **SKU** from Product table to the Values.
12. Drag and drop **No of Products (CM), Product Keys of Duplicate Products (CM)** and **Product List Price of Duplicate Products (CM)** from All Measures table to the Values.
13. Click on **No of Products (CM)** to sort the visual.

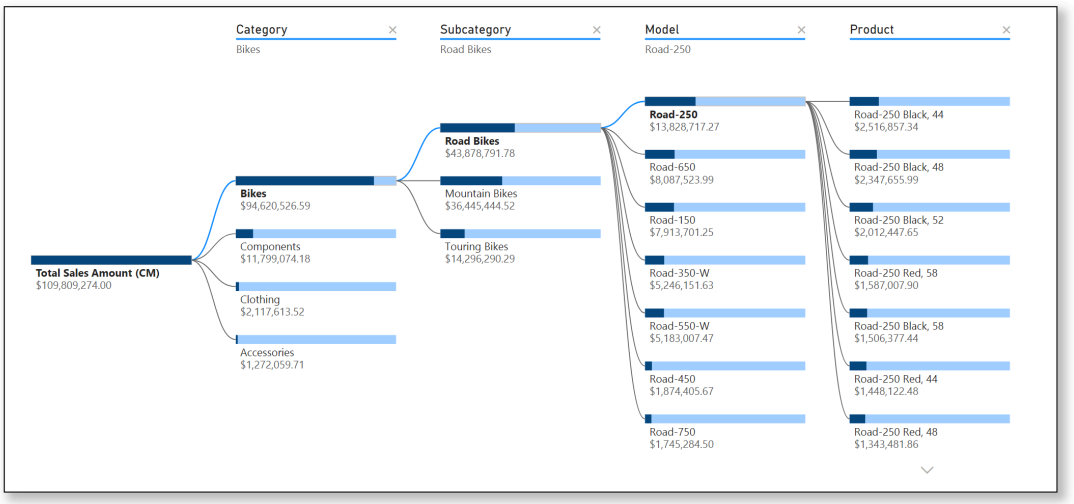
Product	SKU	No of Products (CM)	Product Keys Of Duplicate Products (CM)	Product List Price of Duplicate Products (CM)
AWC Logo Cap	CA-1098	3	225,223,224	9, 8.6, 8.6
HL Mountain Frame - Black, 38	FR-M94B-38	3	306,305,304	1349.6, 1226.9, 1191.2
HL Mountain Frame - Black, 42	FR-M94B-42	3	298,297,296	1349.6, 1226.9, 1191.2
HL Mountain Frame - Black, 46	FR-M94B-46	3	303,302,301	1349.6, 1226.9, 1191.2
HL Mountain Frame - Silver, 38	FR-M94S-38	3	309,308,307	1364.5, 1240.5, 1204.3
HL Mountain Frame - Silver, 42	FR-M94S-42	3	290,289,288	1364.5, 1240.5, 1204.3
HL Mountain Frame - Silver, 46	FR-M94S-46	3	295,294,293	1364.5, 1240.5, 1204.3
HL Road Frame - Red, 44	FR-R92R-44	3	243,241,242	1431.5, 1301.4, 1263.5
HL Road Frame - Red, 48	FR-R92R-48	3	246,244,245	1431.5, 1301.4, 1263.5
HL Road Frame - Red, 52	FR-R92R-52	3	249,247,248	1431.5, 1301.4, 1263.5
HL Road Frame - Red, 56	FR-R92R-56	3	252,250,251	1431.5, 1301.4, 1263.5
HL Road Frame - Red, 62	FR-R92R-62	3	240,238,239	1431.5, 1301.4, 1263.5
LL Road Frame - Black, 44	FR-R38B-44	3	281,279,280	337.2, 297.6, 306.6
LL Road Frame - Black, 48	FR-R38B-48	3	284,282,283	337.2, 297.6, 306.6
LL Road Frame - Black, 52	FR-R38B-52	3	287,285,286	337.2, 297.6, 306.6
LL Road Frame - Black, 58	FR-R38B-58	3	255,253,254	337.2, 297.6, 306.6
LL Road Frame - Black, 60	FR-R38B-60	3	258,256,257	337.2, 297.6, 306.6

Notice how different product keys have different list prices.

Using ISINSCOPE()

In this exercise, you will learn how to find subtotals where each subcategory shows itself as a percentage of total. The data set consists of four levels of hierarchy Category SubCategory Model Product.

The business problem under consideration is to show the percentage shown by the dark blue lines at each node of the following decomposition tree.



Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_3.pbix..

To understand the problem lets first create a few plain measures:

1. Right click on the **All Measures** table in the **Fields** pan.
2. Click on **New measure**.

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
All Product Sales (CM) =
CALCULATE ( [Total Sales Amount (CM)], REMOVEFILTERS ( 'Product' ) )
```

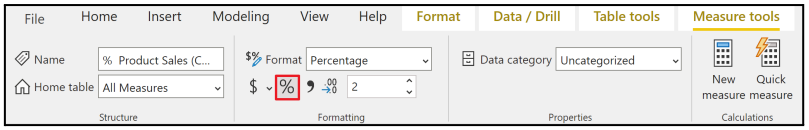
Notice that All Product Sales (CM) is not formatted because it is an intermediate

4. Right click on the **All Measures** table in the **Fields** pan.
5. Click on **New measure**.

   Measure tools open.
6. Type the following formula in the formula bar and press Enter.

```
%  Product Sales (CM) =
DIVIDE ( [Total Sales Amount (CM)], [All Product Sales (CM)], 0 )
```

7. Click on the "%" in the formatting section.



8. Click on the **matrix** visual in the **visualization pane**.
9. Drag and drop **Category** and **SubCategory** from the Product table to the rows.
   A hierarchy will be detected by Power BI.
10. Drag and drop **Total Sales Amount (CM), All Product Sales (CM)** and **% Product Sales (CM)** from All Measures table to the Values.

Category	Total Sales Amount (CM)	All Product Sales (CM)	% Product Sales (CM)
Accessories	**$1,272,059.71**	**109,809,274.00**	**1.16 %**
Bike Racks	$237,096.21	109,809,274.00	0.22 %
Bike Stands	$39,591.00	109,809,274.00	0.04 %
Bottles and Cages	$64,274.95	109,809,274.00	0.06 %
Cleaners	$18,407.03	109,809,274.00	0.02 %
Fenders	$46,619.58	109,809,274.00	0.04 %
Helmets	$484,049.97	109,809,274.00	0.44 %
Hydration Packs	$105,826.62	109,809,274.00	0.10 %
Lights		109,809,274.00	
Locks	$16,225.22	109,809,274.00	0.01 %
Panniers		109,809,274.00	
Pumps	$13,514.71	109,809,274.00	0.01 %
Tires and Tubes	$246,454.42	109,809,274.00	0.22 %
Bikes	**$94,620,526.59**	**109,809,274.00**	**86.17 %**
Clothing	**$2,117,613.52**	**109,809,274.00**	**1.93 %**
Components	**$11,799,074.18**	**109,809,274.00**	**10.75 %**
Total	**$109,809,274.00**	**109,809,274.00**	**100.00 %**

Notice that the Bike Racks have a value of 0.22 %. This does not provide any significant information as it is the contribution in the total Sales.

Expected Result:

A more logical visualization where each SubCategory shows its percentage contribution to the total is shown below.

Category	Total Sales Amount (CM)	% Product Sales (CM)	% ProductSales Revised (CM)
Accessories	**$1,272,059.71**	**1.16 %**	**1.16 %**
Bike Racks	$237,096.21	0.22 %	18.64 %
Bike Stands	$39,591.00	0.04 %	3.11 %
Bottles and Cages	$64,274.95	0.06 %	5.05 %
Cleaners	$18,407.03	0.02 %	1.45 %
Fenders	$46,619.58	0.04 %	3.66 %
Helmets	$484,049.97	0.44 %	38.05 %
Hydration Packs	$105,826.62	0.10 %	8.32 %
Locks	$16,225.22	0.01 %	1.28 %
Pumps	$13,514.71	0.01 %	1.06 %
Tires and Tubes	$246,454.42	0.22 %	19.37 %
Bikes	**$94,620,526.59**	**86.17 %**	**86.17 %**
Clothing	**$2,117,613.52**	**1.93 %**	**1.93 %**
Components	**$11,799,074.18**	**10.75 %**	**10.75 %**
Total	**$109,809,274.00**	**100.00 %**	**100.00 %**

Using ISINSCOPE():

ISINSCOPE() can be used to restrict the scope of evaluation.

11. Right click on the **All Measures** table in the **Fields** pan.
12. Click on **New measure**.

    Measure tools open.
13. Type the following formula in the formula bar and press Enter.

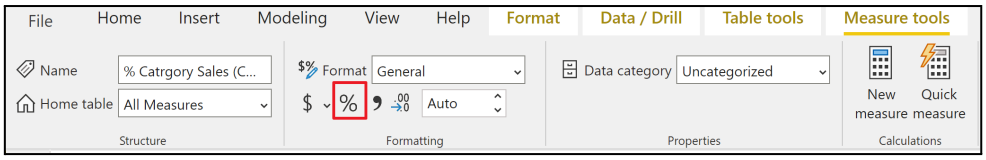
```
All Category Sales (CM) =
CALCULATE ( [Total Sales Amount (CM)], REMOVEFILTERS ( 'Product'[Category] ) )
```

14. Right click on the **All Measures** table in the **Fields** pan.
15. Click on **New measure**.

    Measure tools open.
16. Type the following formula in the formula bar and press Enter.

```
% Catrgory Sales (CM) =
DIVIDE ( [Total Sales Amount (CM)], [All Category Sales (CM)], 0 )
```

17. Click on the "%" in the formatting section.



So far you have learnt how to calculate the percentage sales of a category to the total sales amount. The logic will be replicated for all other hierarchical levels. When we drill down, SubCategorySales% is calculated by dividing SubCategorySales to AllProductSales.

18. Right click on the **All Measures** table in the **Fields** pan.
19. Click on **New measure**.

    Measure tools open.
20. Type the following formula in the formula bar and press Enter.

```
All SubCategory Sales (CM) =
CALCULATE (
    [Total Sales Amount (CM)],
    REMOVEFILTERS ( 'Product'[Subcategory] )
)
```

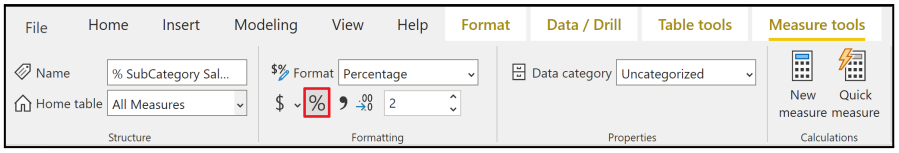
21. Right click on the **All Measures** table in the **Fields** pan.
22. Click on **New measure**.

Measure tools open.

23. Type the following formula in the formula bar and press Enter.

```
% SubCategory Sales (CM) =
DIVIDE ( [Total Sales Amount (CM)], [All SubCategory Sales (CM)], 0 )
```

24. Click on the "%" in the formatting section.



25. Right click on the **All Measures** table in the **Fields** pan.
26. Click on **New measure**.

Measure tools open.

27. Type the following formula in the formula bar and press Enter.

```
Product Category ISINSCOPE (CM) =
ISINSCOPE ( 'Product'[Category] )
```

It is here that we have restricted the scope.

28. Right click on the **All Measures** table in the **Fields** pan.
29. Click on **New measure**.

Measure tools open.

30. Type the following formula in the formula bar and press Enter.

```
Product SubCategory ISINSCOPE (CM) =
ISINSCOPE ( 'Product'[Subcategory] )
```

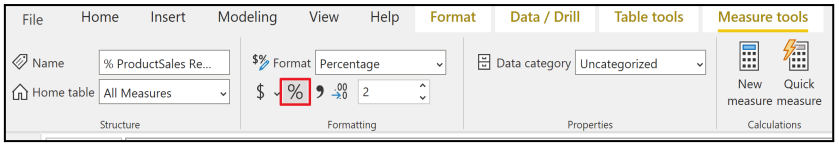
31. Right click on the **All Measures** table in the **Fields** pan.
32. Click on **New measure**.

Measure tools open.

33. Type the following formula in the formula bar and press Enter.

```
% ProductSales Revised (CM) =
IF (
    [Product SubCategory ISINSCOPE (CM)],
    [% SubCategory Sales (CM)],
    [% Catrgory Sales (CM)]
)
```

34. Click on the "%" in the formatting section.



35. Remove **All Product Sales (CM)** and **% Product Sales (CM)** from Values of the matrix.
36. Drag and drop **All Category Sales (CM), All SubCategory Sales (CM), % SubCategory Sales (CM), Product Category ISINSCOPE (CM), Product SubCategory ISINSCOPE (CM)** and **% ProductSales Revised (CM)** from All Measures table to the Values.

Category	Total Sales Amount (CM)	All Category Sales (CM)	All SubCategory Sales (CM)	% SubCategory Sales (CM)	Product Category ISINSCOPE (CM)	Product SubCategory ISINSCOPE (CM)	% ProductSales Revised (CM)
Accessories	**$1,272,059.71**	**109,809,274.00**	**1,272,059.71**	**100.00 %**	**True**	**False**	**1.16 %**
Bike Racks	$237,096.21	237,096.21	1,272,059.71	18.64 %	True	True	18.64 %
Bike Stands	$39,591.00	39,591.00	1,272,059.71	3.11 %	True	True	3.11 %
Bottles and Cages	$64,274.95	64,274.95	1,272,059.71	5.05 %	True	True	5.05 %
Cleaners	$18,407.03	18,407.03	1,272,059.71	1.45 %	True	True	1.45 %
Fenders	$46,619.58	46,619.58	1,272,059.71	3.66 %	True	True	3.66 %
Helmets	$484,049.97	484,049.97	1,272,059.71	38.05 %	True	True	38.05 %
Hydration Packs	$105,826.62	105,826.62	1,272,059.71	8.32 %	True	True	8.32 %
Lights			1,272,059.71		True	True	
Locks	$16,225.22	16,225.22	1,272,059.71	1.28 %	True	True	1.28 %
Panniers			1,272,059.71		True	True	
Pumps	$13,514.71	13,514.71	1,272,059.71	1.06 %	True	True	1.06 %
Tires and Tubes	$246,454.42	246,454.42	1,272,059.71	19.37 %	True	True	19.37 %
Bikes	**$94,620,526.59**	**109,809,274.00**	**94,620,526.59**	**100.00 %**	**True**	**False**	**86.17 %**
Clothing	**$2,117,613.52**	**109,809,274.00**	**2,117,613.52**	**100.00 %**	**True**	**False**	**1.93 %**
Components	**$11,799,074.18**	**109,809,274.00**	**11,799,074.18**	**100.00 %**	**True**	**False**	**10.75 %**
Total	**$109,809,274.00**	**109,809,274.00**	**109,809,274.00**	**100.00 %**	**False**	**False**	**100.00 %**

Now that you have all the building blocks, you can use ISINSCOPE() to create a measure which determines the browsing level of the matrix and then calculates the percentage contribution.

37. Right click on the **All Measures** table in the **Fields** pan.
38. Click on **New measure**.

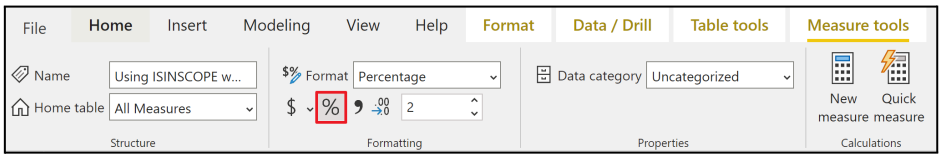
    Measure tools open.
39. Type the following formula in the formula bar and press Enter.

```
Using ISINSCOPE with variables (CM) =
VAR AllCategorySales =
    CALCULATE ( [Total Sales Amount (CM)], REMOVEFILTERS ( 'Product'[Category] ) )
VAR AllSubCategorySales =
    CALCULATE (
        [Total Sales Amount (CM)],
        REMOVEFILTERS ( 'Product'[Subcategory] )
    )
VAR PctCatSales =
    DIVIDE ( [Total Sales Amount (CM)], AllCategorySales, 0 )
VAR PctSubCatSales =
    DIVIDE ( [Total Sales Amount (CM)], AllSubCategorySales, 0 )
RETURN
    IF ( ISINSCOPE ( 'Product'[Subcategory] ), PctSubCatSales, PctCatSales )
```

In the expression above:

- *AllCategorySales calculates the total sales removing the filter from the category.*
- *AllSubCategorySales calculates the total sales removing the filter from the Subcategory.*
- *PctCatSales calculates the percentage of the category to the total sales of all categories.*
- *PctSubCatSales calculates the percentage of the Subcategory to the total sales of all Subcategories.*
- *In the last line, the current filter context is checked and the result is returned based on the scope of the current filter context.*

40. Click on the "%" in the formatting section.



41. Click on the **matrix** visual in the **visualization pane**.
42. Drag and drop **Category** and **SubCategory** from the Product table to the rows.

    A hierarchy will be detected by Power BI.
43. Drag and drop **Total Sales Amount (CM)** and **Using ISINSCOPE with variables (CM)** from All Measures table to the Values.

Category	Total Sales Amount (CM)	Using ISINSCOPE with variables (CM)
Accessories	**$1,272,059.71**	**1.16 %**
Bike Racks	$237,096.21	18.64 %
Bike Stands	$39,591.00	3.11 %
Bottles and Cages	$64,274.95	5.05 %
Cleaners	$18,407.03	1.45 %
Fenders	$46,619.58	3.66 %
Helmets	$484,049.97	38.05 %
Hydration Packs	$105,826.62	8.32 %
Locks	$16,225.22	1.28 %
Pumps	$13,514.71	1.06 %
Tires and Tubes	$246,454.42	19.37 %
Bikes	**$94,620,526.59**	**86.17 %**
Clothing	**$2,117,613.52**	**1.93 %**
Components	**$11,799,074.18**	**10.75 %**
Total	**$109,809,274.00**	**100.00 %**

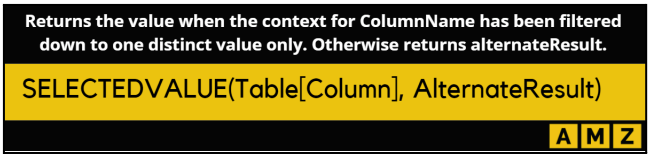
Notice that if the product is in scope, the model and subcategory will also be in scope. However, the opposite is not correct. If the category is in scope, the subcategory might not be in scope.

Parameter table and SELECTEDVALUE()

In this exercise, you will learn how to create a parameter table and find the number of customers buying multiple products in the current filter context.

Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_4.pbix..

SELECTEDVALUE() is used to provide values from a slicer where the user can select a number.



To create a parameter table:

1. Click on the **Modeling** tab in the **Ribbon**.
2. Click on **New Table**.
3. Type the following expression in the formula bar and press Enter.

```
Number of Products (CT) =
DATATABLE (
    "Num Products", INTEGER,
    {
        { 2 },
        { 3 },
        { 4 },
        { 5 },
        { 10 },
        { 15 },
        { 20 }
    }
)
```

The parameter table has been created.

Now let's create a DAX expression to get the value of parameter from the table and use that value to filter the customers.

4. Right click on the **All Measures** table in the **Fields** pan.
5. Click on **New measure**.

   Measure tools open.

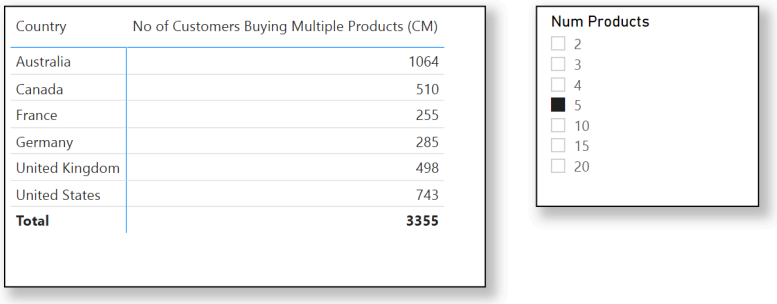
6. Type the following formula in the formula bar and press Enter.

```
No of Customers Buying Multiple Products (CM) =
VAR NumberOfProductsSelected =
    SELECTEDVALUE ( 'Number of Products (CT)'[Num Products], 2 )
VAR CustomersWithSelectedProducts =
    FILTER (
        Customer,
        AND (
            Customer[CustomerKey] <> -1,
            CALCULATE ( DISTINCTCOUNT ( Sales[ProductKey] ) >= NumberOfProductsSelected )
        )
    )
VAR Result =
    COUNTROWS ( CustomersWithSelectedProducts )
RETURN
    Result
```

In this expression:

- *The variable NumberOfProductsSelected stores the value selected by the customer in the slicer.*
- *The variable CustomersWithSelectedProducts stores a table containing the list of customers buying the selected number of products by filtering the Customer table.*
- *Finally, in the result, the rows of virtual table CustomersWithSelectedProducts are counted.*

7. Click on the **matrix** visual in the **visualization pane**.
8. Drag and drop **Country** from the SalesTerritory table to the rows.
9. Drag and drop **Number of Customers buying Multiple Products (CM)** from All Measures table to the Values.
10. Click on the **slicer** visual in the **visualization pane**.
11. Drag and drop Num Products from Number of Products (CT) to the Fields.
12. From the drop down at the top of slicer visual select List.
13. Click on 5 in the slicer.

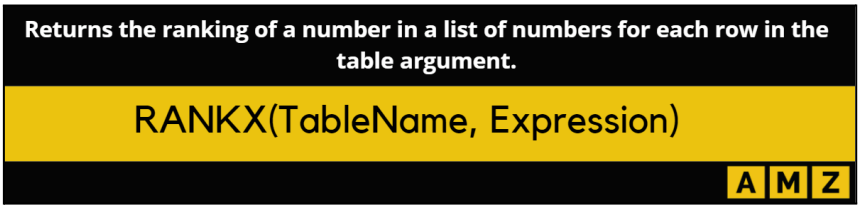


Country	No of Customers Buying Multiple Products (CM)
Australia	1064
Canada	510
France	255
Germany	285
United Kingdom	498
United States	743
Total	**3355**

Using RANKX()

RANKX() is used to rank entities over one another based on a measure value. In this exercise, you will learn how to use RANKX() to create static as well as dynamic ranking.

Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_5.pbix..



Static Ranking:

To create a static ranking of customers by total sales amount:

1. Right click on the **Customer** table in the **Fields**.
2. Click on **New Column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Customer Ranking (CC) =
RANKX ( Customer, [Total Sales Amount (CM)] )
```

In the expression above:

- *The rank of the customer based on the Sales amount is calculated. The expression is evaluated for each row.*

4. Click on the **matrix** visual in the **visualization pane**.
5. Drag and drop **Customer** from the Customer table to the rows.
6. Drag and drop **Total Sales Amount (CM)** and **Customer Ranking (CC)** from All Measures table to the Values.
7. Click on Customer Ranking (CC) to sort in ascending order.

Customer	Total Sales Amount (CM)	Customer Ranking (CC)
[Not Applicable]	$80,450,596.11	1
Nichole Nara	$13,295.38	2
Kaitlyn Henderson	$13,294.27	3
Margaret He	$13,269.27	4
Randall Dominguez	$13,265.99	5
Adriana Gonzalez	$13,242.70	6
Rosa Hu	$13,215.65	7
Brandi Gill	$13,195.64	8
Brad She	$13,173.19	9

Notice that the invalid relationships for the customer table have the highest ranking.

It is important to note here that these refer to the Sales from resellers and not the customers. Therefore, considering them in the ranking will be incorrect.

8. Click on **Customer Ranking (CC)** in the Customer table.

   Column tools open.

9. Type the following formula in the formula bar and press Enter.

```
Customer Ranking (CC) =
IF (
    Customer[CustomerKey] = -1,
    BLANK (),
    RANKX ( Customer, [Total Sales Amount (CM)] )
)
```

In the above expression:

- *The customer key is checked, if it is "-1" then the blank value is returned otherwise the Rank is returned.*

Notice the change in values of the ranking in the matrix visual. The ranking now starts from two because the iterator iterates over the customer table and considers CustomerKey of -1 in ranking.

Customer	Total Sales Amount (CM)	Customer Ranking (CC)
[Not Applicable]	$80,450,596.11	
Nichole Nara	$13,295.38	2
Kaitlyn Henderson	$13,294.27	3
Margaret He	$13,269.27	4
Randall Dominguez	$13,265.99	5
Adriana Gonzalez	$13,242.70	6
Rosa Hu	$13,215.65	7
Brandi Gill	$13,195.64	8
Brad She	$13,173.19	9
Francisco Sara	$13,164.64	10

10. Click on **Customer Ranking (CC)** in the Customer table.

    Column tools open.

11. Type the following formula in the formula bar and press Enter.

```
Customer Ranking (CC) =
IF (
    Customer[CustomerKey] = -1,
    BLANK (),
    RANKX ( Customer, [Total Sales Amount (CM)] ) - 1
)
```

In the above expression:

- *1 was subtracted from the Rank because RANKX() is evaluated row by row and the largest sales still occur where CustomerKey = -1.*

A little hack fixed the results.

Customer	Total Sales Amount (CM)	Customer Ranking (CC)
[Not Applicable]	$80,450,596.11	
Nichole Nara	$13,295.38	1
Kaitlyn Henderson	$13,294.27	2
Margaret He	$13,269.27	3
Randall Dominguez	$13,265.99	4
Adriana Gonzalez	$13,242.70	5
Rosa Hu	$13,215.65	6
Brandi Gill	$13,195.64	7
Brad She	$13,173.19	8
Francisco Sara	$13,164.64	9
Maurice Shan	$12,909.67	10
Janet Munoz	$12,489.17	11
Lisa Cai	$11,469.19	12
Lacey Zheng	$11,248.46	13
Total	**$109,809,274.00**	**170230312**

12. Click on Filters Pan.
13. Uncheck the box next to [Not Applicable] in the Customers filter.

Customer	Total Sales Amount (CM)	Customer Ranking (CC)
Nichole Nara	$13,295.38	1
Kaitlyn Henderson	$13,294.27	2
Margaret He	$13,269.27	3
Randall Dominguez	$13,265.99	4
Adriana Gonzalez	$13,242.70	5
Rosa Hu	$13,215.65	6
Brandi Gill	$13,195.64	7
Brad She	$13,173.19	8
Francisco Sara	$13,164.64	9
Maurice Shan	$12,909.67	10

Dynamic Ranking:

For dynamic ranking, let's try to rank the customers based on the country selection in the slicer.

14. Right click on the **Customer** table in the **Fields**.
15. Click on **New Column**.

    Column tools open.

16. Type the following formula in the formula bar and press Enter.

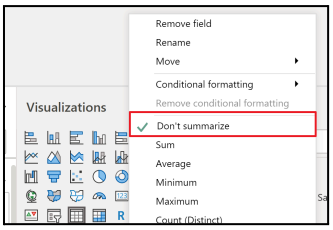
```
Country Wise Customer Ranking (CC ) =
RANKX (
    ALLEXCEPT ( Customer, Customer[Country-Region] ),
    [Total Sales Amount (CM)]
)
```

In the above expression:

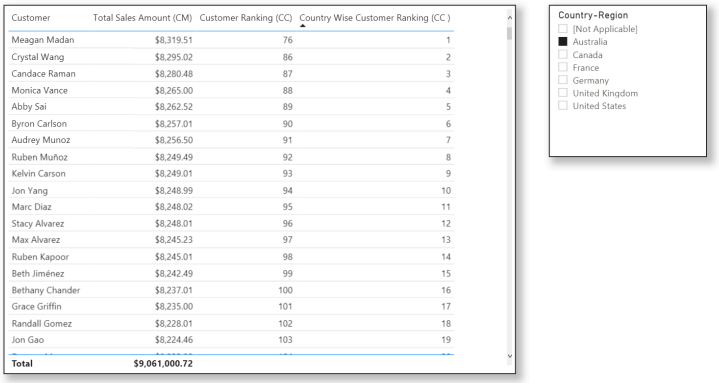
- *ALLEXCEPT() is used to leverage the power of RANKX()*

17. Click on the **table** visual in the **visualization pane**.
18. Drag and drop **Customer** from the Customer table to the values.

19. Drag and drop **Total Sales Amount (CM)** from All Measures table to the Values.
20. Drag and drop **Customer Ranking (CC)** and **Country Wise Customer Ranking (CC)** from Customer table to the Values.
21. Right click on **Country Wise Customer Ranking (CC)** in Values.
22. Click on Don't Summarize.



23. Click on Slicer visual in the Visualization Pane.
24. Drag and drop **Country-Region** from Customer to the Field.
25. Click on **Australia**.



Customer	Total Sales Amount (CM)	Customer Ranking (CC)	Country Wise Customer Ranking (CC)
Meagan Madan	$8,319.51	76	1
Crystal Wang	$8,295.02	86	2
Candace Raman	$8,280.48	87	3
Monica Vance	$8,265.00	88	4
Abby Sai	$8,262.52	89	5
Byron Carlson	$8,257.01	90	6
Audrey Munoz	$8,256.50	91	7
Ruben Muñoz	$8,249.49	92	8
Kelvin Carson	$8,249.01	93	9
Jon Yang	$8,248.99	94	10
Marc Diaz	$8,248.02	95	11
Stacy Alvarez	$8,248.01	96	12
Max Alvarez	$8,245.23	97	13
Ruben Kapoor	$8,245.01	98	14
Beth Jiménez	$8,242.49	99	15
Bethany Chander	$8,237.01	100	16
Grace Griffin	$8,235.00	101	17
Randall Gomez	$8,228.01	102	18
Jon Gao	$8,224.46	103	19
Total	**$9,061,000.72**		

Notice that the dynamic ranking starts from 1 but the static ranking maintains its state despite the selection.

ISINSCOPE() with RANKX()

In this exercise, you will learn how to create a ranking such that the sub levels are ranked against the parent level.

Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_6.pbix..

To create s geographical ranking with State-Province at the sub level and Country-Region at a higher level:

1. Right click on the **All Measures** table in the **Fields** pan.
2. Click on **New measure**.

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Geographical Rank (CM) =
IF (
    ISINSCOPE ( Customer[State-Province] ),
    RANKX ( ALLSELECTED ( Customer[State-Province] ), [Total Sales Amount (CM)] ),
    IF (
        ISINSCOPE ( Customer[Country-Region] ),
        RANKX ( ALLSELECTED ( Customer[Country-Region] ), [Total Sales Amount (CM)] )
    )
)
```

In the expression above:

- *ISINSCOPE() is used to test the scope of the current filter context.*
- *RANKX() than evaluates the Rank for ALLSELECTED[Country-Region] based on the total sales amount.*

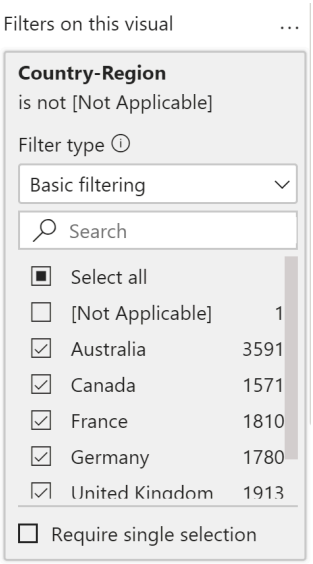
4. Click on the **matrix** visual in the **visualization pane**.
5. Drag and drop **Country-Region** and **State-Province** from the Customer table to the rows.

   A hierarchy will be detected by Power BI.
6. Drag and drop **Geographical Rank (CM)** from All Measures table to the Values.

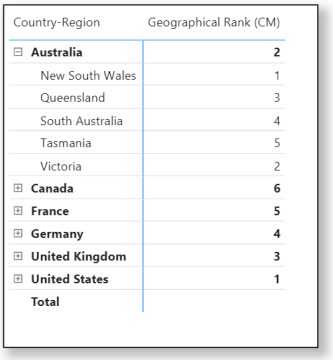
Country-Region	Geographical Rank (CM)
[Not Applicable]	**1**
Australia	**3**
New South Wales	1
Queensland	3
South Australia	4
Tasmania	5
Victoria	2
Canada	**7**
France	**6**
Germany	**5**
United Kingdom	**4**
United States	**2**
Total	

Notice that the [Not Applicable] still appears in the ranking.

7. Uncheck the box at the left on Not Applicable in the Filters on the visual in the Filters Pane.



Notice the change in Ranking.



Country-Region	Geographical Rank (CM)
Australia	**2**
New South Wales	1
Queensland	3
South Australia	4
Tasmania	5
Victoria	2
Canada	**6**
France	**5**
Germany	**4**
United Kingdom	**3**
United States	**1**
Total	

Australia itself ranks at the 2nd position. When you analyze Australia, New South Whales has the highest rank.

Counting spikes

In this exercise, you will learn how to analyze the monthly sales by visualizing:

- ✔ **Threshold:** line representing a value of 75% of the largest monthly value of Sales Amount.
- ✔ **High Months:** Number of times that the monthly value has been greater than a threshold.
- ✔ **Spikes:** Number of times the sales amount has crossed the threshold value.

Continue using the previous file for the exercise or open the file Practical_Use_Of_Concepts_And_Other_Functions_7.pbix..

First of all, lets calculate the threshold.

Setting the threshold:

1. Right click on the **All Measures** table in the **Fields** pan.
2. Click on **New measure**.

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Threshold (CM) =
CALCULATE (
    VAR MinSales =
        MINX ( ALL ( 'Date'[Full Month] ), [Total Sales Amount (CM)] )
    VAR MaxSales =
        MAXX ( ALL ( 'Date'[Full Month] ), [Total Sales Amount (CM)] )
    VAR Boundary = MinSales + ( MaxSales - MinSales ) * 0.75
    RETURN
        Boundary,
    ALL ( 'Date' )
)
```

Finding the High Months:

4. Right click on the **All Measures** table in the **Fields** pan.
5. Click on **New measure**.

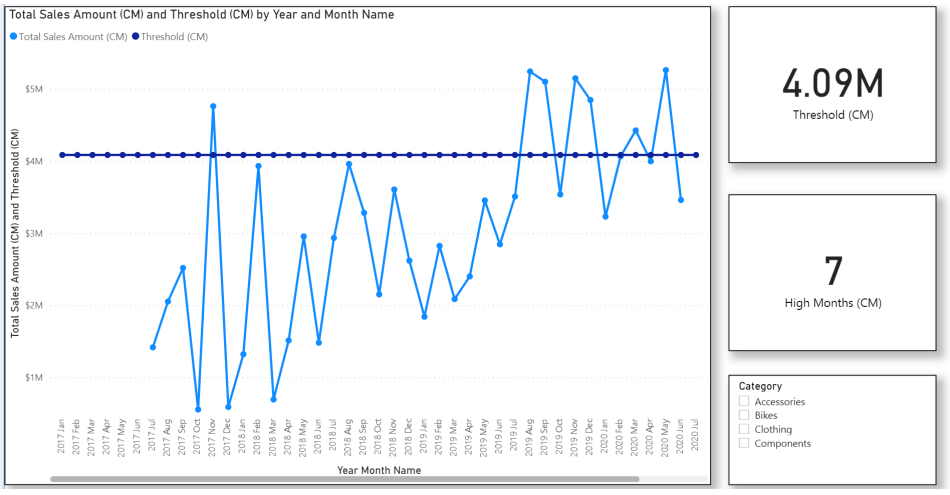
   Measure tools open.
6. Type the following formula in the formula bar and press Enter.

```
High Months (CM) =
VAR Threshold = [Threshold (CM)]
VAR MonthsOverThreshold =
    FILTER ( ALL ( 'Date'[Full Month] ), [Total Sales Amount (CM)] > Threshold )
VAR Result =
    COUNTROWS ( MonthsOverThreshold )
RETURN
    Result
```

In the expression above:

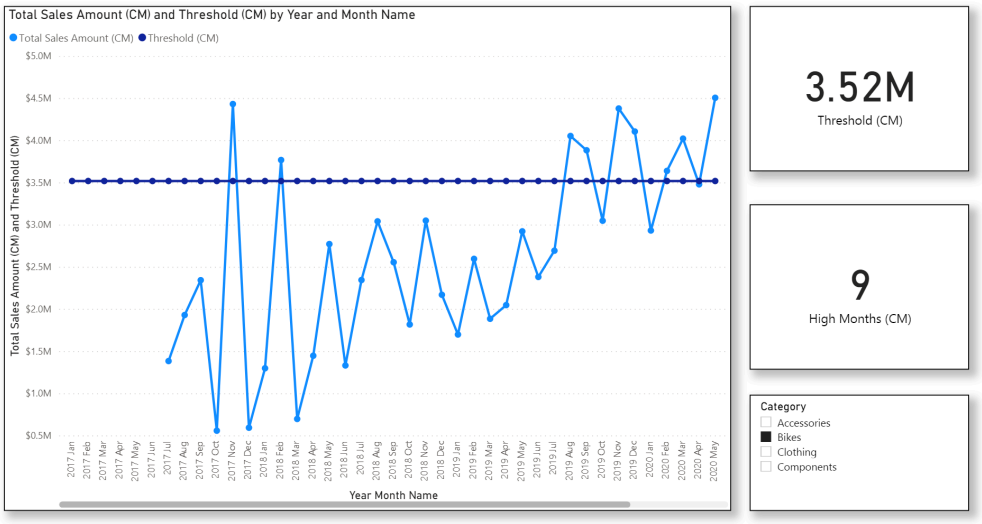
- *The High Month Sales are found by finding the number of months when the sales were greater than the Threshold.*
- *The Threshold variables stores the value evaluated from the measure.*
- *MonthsOverThreshold filters all the months where the total sales amount is greater than Threshold value and stores these month dates in a virtual table.*
- *At the end, the rows in the MonthsOverThreshold table are counted.*

7. Click on the **card visual** in the **visualization pane**.
8. Drag and drop **Threshold (CM)** from All Measures to the Fields.
9. Click on the **card visual** in the **visualization pane**.
10. Drag and drop **High Months (CM)** from All Measures to the Fields.
11. Click on the **Slicer** in the **visualization pane**.
12. Drag and drop **Category** from Products to the Fields.
13. Click on the **line chart visual** in the **visualization pane**.
14. Drag and drop **Year, Month Name and Date** from the Date table to the Axis.
15. Drag and drop **Total Sales Amount (CM)** and **Threshold (CM)** from All Measures to the values.



Notice that the Threshold is 4.09M and there were 7 months when the sales amount was greater then 4.09M.

16. Click on Bikes in the slicer.



Notice the change in the values of the visuals.

Using PREVIOUSMONTH():

Returns a table that contains a column of all dates from the previous month, based on the first date in the dates column, in the current context.

PREVIOUSMONTH(Date[Date])

AMZ

Finding the Spikes:

17. Right click on the **All Measures** table in the **Fields** pan.
18. Click on **New measure**.

    Measure tools open.
19. Type the following formula in the formula bar and press Enter.

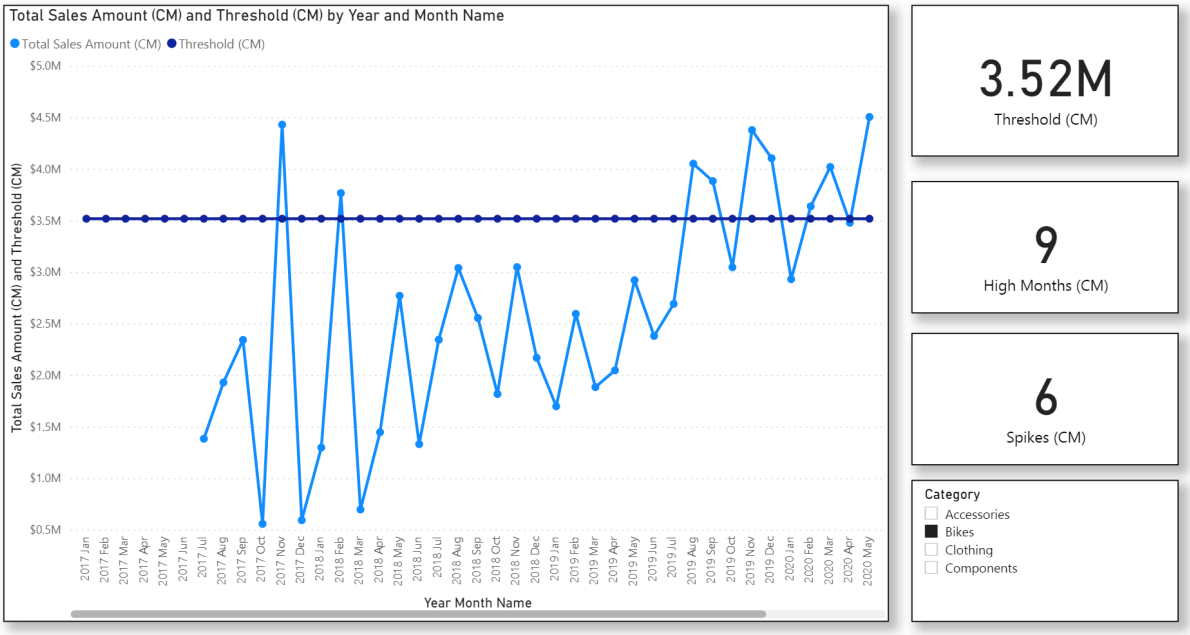
```
Spikes (CM) =
VAR Threshold = [Threshold (CM)]
VAR MaxSpikes =
    FILTER (
        VALUES ( 'Date'[Full Month] ),
        VAR SalesCurrentMonth = [Total Sales Amount (CM)]
        VAR SalesPreviousMonth =
            CALCULATE ( [Total Sales Amount (CM)], PREVIOUSMONTH ( 'Date'[Date] ) )
        VAR EnteringThreshold = SalesCurrentMonth >= Threshold
            && SalesPreviousMonth < Threshold
        RETURN
            EnteringThreshold
    )
VAR Result =
    COUNTROWS ( MaxSpikes )
RETURN
    Result
```

In the expression above:

- *Spikes specify the number of times that the sales trend crossed the threshold line.*
- *The Threshold variable calculates the threshold based on the incoming filter context.*

- *In the MaxSpikes variable, a virtual table containing the months of the Spikes is stored.*
- *Inside the MaxSpikes variable, VALUES() generate the unique month table and SalesCurrentMonth stores the sales for the month in the current filter context.*
- *SalesPreviousMonth calculates the sales for the last month using PREVIOUSMONTH() function.*
- *The EnteringThreshold variable checks if the sales for the current month is greater than threshold and the sales for the previous month is less than threshold, i.e., there was a transition in the trend line.*
- *Finally, the number of rows in the MaxSpkies is counted.*

20. Click on the **card visual** in the **visualization pane**.
21. Drag and drop **Spikes (CM)** from All Measures to the Fields.



Notice that there were 6 spikes. However, the high months were recorded to be 9.

Chapter 15: Hierarchies In DAX

What will you learn in this chapter?

- ✔ What are hierarchies
- ✔ Using Path functions to traverse hierarchy
- ✔ Creating Levels
- ✔ Finding Maximum Depths
- ✔ Determining Browsing Levels
- ✔ Analyzing Sales in Hierarchies

What are hierarchies

Hierarchy in Power BI is a system where the elements are ranked over one another. They are used to define the exploration paths. A few examples of hierarchy include:

✔ Year □Month□Day.

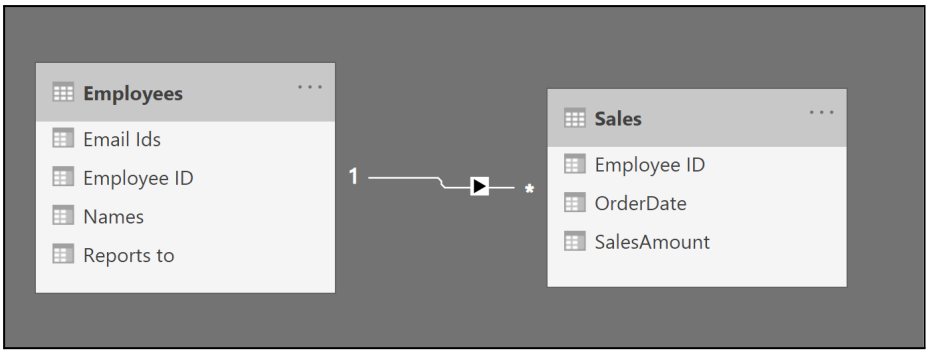
✔ Category □ Subcategory □ Product.

This makes browsing easier.

Open the file Hierarchies_In_DAX_1.pbix..

For this exercise, the data set contains 2 tables:

1. **Employees Table:** Employees record for AMZ Consulting Pty Ltd.
2. **Sales**: Sales Amount associated with each employee.



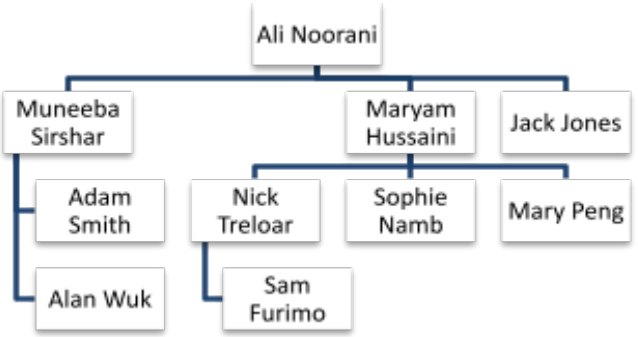
Parent-Child Hierarchy:

For a Parent child hierarchy, a 1:M relationship exists between the parent and the children. All the columns in the hierarchy belong to the same table therefore self-referencing relationships cannot be created. Some characteristics of a parent-child hierarchy are:

✔ They are unbalanced or ragged. This means that each parent can have one, many or no children.

✔ Each node can have a variable depth.

✔ Data is associated with each node whether it is parent node or child.



Final Expected Result:

Power BI to date cannot typically handle Parent-Child hierarchies. Here's a final result that you will want to display.

Level 1 (CC)	Sales at Level (CM)
⊟ Ali Noorani	$361,077.95
⊞ Jack Jones	$21,463.76
⊟ Maryam Hussaini	$151,125.68
⊞ Mary Peng	$42,114.20
⊟ Nick Treloar	$83,489.46
Sam Furimo	$15,484.00
⊞ Sophie Namb	$8,841.24
⊟ Muneeba Sirshar	$148,266.83
⊞ Adam Smith	$44,888.61
⊞ Alan Wuk	$34,907.65
Total	$361,077.95

You will want to browse 'Ali Noorani' and the total sales will be the sum of Ali Norani's sales and the Employees reporting to Ali Noorani.

It is also worth noticing that Jack Jones cannot be expanded any further because there are no employees reporting to Jack. However, Maryam Hussaini can be expanded because there are 3 employees reporting to her directly and 1 indirectly. The same concept will be applied at all hierarchical levels.

Sam Furimo is reporting to Nick Treloar who is then reporting to Maryam Hussaini so you can expand 4 levels here as compared to 2 levels in Jack Jones.

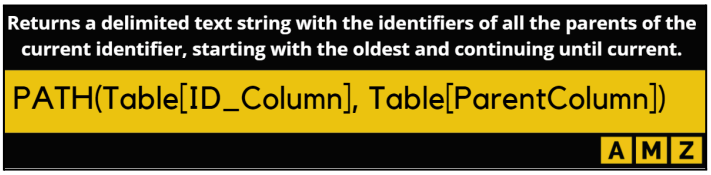
In a typical company's data base, the hierarchy has been set up by adding a 'Report to' column with each 'Employee ID'. You cannot create a relationship because both the 'Employee ID' and 'Reports to' belong to the same table. In this chapter, you will learn how to use DAX to calculate expressions that resemble to self-join.

Employee ID	Email Ids	Names	Reports to
1	anoorani@amzconsulting.com.au	Ali Noorani	
2	m.sirshar@amzconsulting.com.au	Muneeba Sirshar	1
3	m.hussaini@amzconsulting.com.au	Maryam Hussaini	1
4	jjones@amzconsulting.com.au	Jack Jones	1
5	asmith@amzconsulting.com.au	Adam Smith	2
6	awuk@amzconsulting.com.au	Alan Wuk	2
7	mpeng@amzconsulting.com.au	Mary Peng	3
8	ntreloar@amzconsulting.com.au	Nick Treloar	3
9	snamb@amzconsulting.com.au	Sophie Namb	3
10	sfurimo@amzconsulting.com.au	Sam Furimo	8

Using Path functions to traverse hierarchy

PATH() is a function in Power BI that is specifically created to handle Parent-Child hierarchies. It returns the path that needs to be traversed from the child to the parent.

Continue using the previous file for the exercise or open the file Hierarchies_In_DAX_2.pbix..



To create the path column:

1. Right click on the **Employee** table in the **Fields** pan.
2. Click on **New column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Full Path (CC) =
PATH ( Employees[Employee ID], Employees[Reports to] )
```

4. Click on **Data** tab in the left **Ribbon**.
5. Click on **Employee** table in the **Fields** pan.

Employee ID	Email Ids	Names	Reports to	Full Path (CC)
1	anoorani@amzconsulting.com.au	Ali Noorani		1
2	m.sirshar@amzconsulting.com.au	Muneeba Sirshar	1	1|2
3	m.hussaini@amzconsulting.com.au	Maryam Hussaini	1	1|3
4	jjones@amzconsulting.com.au	Jack Jones	1	1|4
5	asmith@amzconsulting.com.au	Adam Smith	2	1|2|5
6	awuk@amzconsulting.com.au	Alan Wuk	2	1|2|6
7	mpeng@amzconsulting.com.au	Mary Peng	3	1|3|7
8	ntreloar@amzconsulting.com.au	Nick Treloar	3	1|3|8
9	snamb@amzconsulting.com.au	Sophie Namb	3	1|3|9
10	sfurimo@amzconsulting.com.au	Sam Furimo	8	1|3|8|10

Notice the Path for Sam Furimo, it shows that you need to traverse from Sam Furimo □ Nick Treloar □ Maryam Hussaini □ Ali Noorani.

The column for path itself is not useful, it is a hidden column use to create levels in a parent-child hierarchy.

Creating Levels

In order to fix Parent-Child hierarchy in DAX, you will need to fix the maximum number of levels allowed.

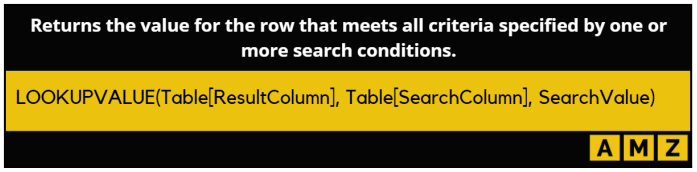
Continue using the previous file for the exercise or open the file Hierarchies_In_DAX_3.pbix..

Using PATHITEM()

PATHITEM() searches the Path column and returns the key present at the Nth position of the path.

Using LOOKUPVALUE()

The function is used to the return the name of the Employee whose Employee ID matches with the searched value by PATHITEM()



To create the Level 1 column:

1. Right click on the **Employee** table in the **Fields** pan.
2. Click on **New column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Level 1 (CC) =
LOOKUPVALUE (
    Employees[Names],
    Employees[Employee ID], PATHITEM ( Employees[Full Path (CC)], 1, INTEGER )
)
```

In the expression above:

- *The PATHITEM() is use to return the Employee ID which is at the 1st position of the Path.*
- *The LOOKUPVALUE() searches for the Employee ID in the Employee ID column and when the results match with that of PATHITEM(), it returns the corresponding Employee Name from the Employee[Names] column.*

Now that you have created the column for Level 1, you will do similar calculations for all other columns. However, you will additionally need to check it the Path length is greater than 2 using the PATHLENGTH() function.

To create the remaining levels:

4. Right click on the **Employee** table in the **Fields** pan.
5. Click on **New column**.

   Column tools open.
6. Type the following formula in the formula bar and press Enter.

```
Level 2 (CC) =
IF (
    PATHLENGTH ( Employees[Full Path (CC)] ) >= 2,
    LOOKUPVALUE (
        Employees[Names],
        Employees[Employee ID], PATHITEM ( Employees[Full Path (CC)], 2, INTEGER )
    ),
    Employees[Level 1 (CC)]
)
```

In the expression above:

- *First of all the PATHLENGTH() is evaluated.*
- *If the PATHLENGTH() is greater than 2 then it means that the current filtering is at second level of hierarchy and LOOKUPVALUE() is performed.*
- *Otherwise the Employee name at last level is returned.*

7. Right click on the **Employee** table in the **Fields** pan.
8. Click on **New column**.

   Column tools open.
9. Type the following formula in the formula bar and press Enter.

```
Level 3 (CC) =
IF (
    PATHLENGTH ( Employees[Full Path (CC)] ) >= 3,
    LOOKUPVALUE (
        Employees[Names],
        Employees[Employee ID], PATHITEM ( Employees[Full Path (CC)], 3, INTEGER )
    ),
    Employees[Level 2 (CC)]
)
```

10. Right click on the **Employee** table in the **Fields** pan.
11. Click on **New column**.

    Column tools open.
12. Type the following formula in the formula bar and press Enter.

```
Level 4 (CC) =
IF (
    PATHLENGTH ( Employees[Full Path (CC)] ) >= 4,
    LOOKUPVALUE (
        Employees[Names],
        Employees[Employee ID], PATHITEM ( Employees[Full Path (CC)], 4, INTEGER )
    ),
    Employees[Level 3 (CC)]
)
```

13. Click on **Data** tab in the left **Ribbon**.
14. Click on **Employee** table in the **Fields** pan.

Employee ID	Email Ids	Names	Reports to	Full Path (CC)	Level 1 (CC)	Level 2 (CC)	Level 3 (CC)	Level 4 (CC)
1	anoorani@amzconsulting.com.au	Ali Noorani		1	Ali Noorani	Ali Noorani	Ali Noorani	Ali Noorani
4	jjones@amzconsulting.com.au	Jack Jones	1	1|4	Ali Noorani	Jack Jones	Jack Jones	Jack Jones
7	mpeng@amzconsulting.com.au	Mary Peng	3	1|3|7	Ali Noorani	Maryam Hussaini	Mary Peng	Mary Peng
8	ntreloar@amzconsulting.com.au	Nick Treloar	3	1|3|8	Ali Noorani	Maryam Hussaini	Nick Treloar	Nick Treloar
10	sfurimo@amzconsulting.com.au	Sam Furimo	8	1|3|8|10	Ali Noorani	Maryam Hussaini	Nick Treloar	Sam Furimo
9	snamb@amzconsulting.com.au	Sophie Namb	3	1|3|9	Ali Noorani	Maryam Hussaini	Sophie Namb	Sophie Namb
3	m.hussaini@amzconsulting.com.au	Maryam Hussaini	1	1|3	Ali Noorani	Maryam Hussaini	Maryam Hussaini	Maryam Hussaini
2	m.sirshar@amzconsulting.com.au	Muneeba Sirshar	1	1|2	Ali Noorani	Muneeba Sirshar	Muneeba Sirshar	Muneeba Sirshar
5	asmith@amzconsulting.com.au	Adam Smith	2	1|2|5	Ali Noorani	Muneeba Sirshar	Adam Smith	Adam Smith
6	awuk@amzconsulting.com.au	Alan Wuk	2	1|2|6	Ali Noorani	Muneeba Sirshar	Alan Wuk	Alan Wuk

15. Click on the **matrix** visual in the **visualization pane**.
16. Drag and drop **Level 1, Level 2, Level 3** and **Level 4** columns from the Employee table to the rows.

    A hierarchy will be detected by Power BI.
17. Drag and drop **SalesAmount** from Sales table to the Values.

Level 1 (CC)	SalesAmount
⊟ **Ali Noorani**	**$361,077.95**
⊞ Ali Noorani	$40,221.68
⊞ Jack Jones	$21,463.76
⊞ Maryam Hussaini	$151,125.68
⊞ Muneeba Sirshar	$148,266.83
Total	**$361,077.95**

Notice that Ali Noorani is shown as the child for himself, this is not what you want.

Finding Maximum Depths

In this exercise, you will learn how to hide the parent member from the hierarchy. This involves 2 logical steps:

✔ Understand the hierarchal level that you are browsing on.

✔ Understand which member you want to hide.

So you will have to hide the node, if the browsing of the hierarchy exceeds the depth of the node.

Continue using the previous file for the exercise or open the file Hierarchies_In_DAX_4.pbix..

To compute the depth of node:

1. Right click on the **Employee** table in the **Fields** pan.
2. Click on **New column**.

   Column tools open.
3. Type the following formula in the formula bar and press Enter.

```
Node Depth (CC) =
PATHLENGTH ( Employees[Full Path (CC)] )
```

4. Click on **Data** tab in the left **Ribbon**.
5. Click on **Employee** table in the **Fields** pan

Employee ID	Email Ids	Names	Reports to	Full Path (CC)	Level 1 (CC)	Level 2 (CC)	Level 3 (CC)	Level 4 (CC)	Node Depth (CC)
1	anoorani@amzconsulting.com.au	Ali Noorani		1	Ali Noorani	Ali Noorani	Ali Noorani	Ali Noorani	1
4	jjones@amzconsulting.com.au	Jack Jones	1	1|4	Ali Noorani	Jack Jones	Jack Jones	Jack Jones	2
7	mpeng@amzconsulting.com.au	Mary Peng	3	1|3|7	Ali Noorani	Maryam Hussaini	Mary Peng	Mary Peng	3
8	ntreloar@amzconsulting.com.au	Nick Treloar	3	1|3|8	Ali Noorani	Maryam Hussaini	Nick Treloar	Nick Treloar	3
10	sfurimo@amzconsulting.com.au	Sam Furimo	8	1|3|8|10	Ali Noorani	Maryam Hussaini	Nick Treloar	Sam Furimo	4
9	snamb@amzconsulting.com.au	Sophie Namb	3	1|3|9	Ali Noorani	Maryam Hussaini	Sophie Namb	Sophie Namb	3
3	m.hussaini@amzconsulting.com.au	Maryam Hussaini	1	1|3	Ali Noorani	Maryam Hussaini	Maryam Hussaini	Maryam Hussaini	2
2	m.sirshar@amzconsulting.com.au	Muneeba Sirshar	1	1|2	Ali Noorani	Muneeba Sirshar	Muneeba Sirshar	Muneeba Sirshar	2
5	asmith@amzconsulting.com.au	Adam Smith	2	1|2|5	Ali Noorani	Muneeba Sirshar	Adam Smith	Adam Smith	3
6	awuk@amzconsulting.com.au	Alan Wuk	2	1|2|6	Ali Noorani	Muneeba Sirshar	Alan Wuk	Alan Wuk	3

Notice that Ali Noorani has a node depth of 1. Whereas, Jack Jones has a node depth of 2 because he is reporting to Ali.

Determining Browsing Levels

So far you have learnt how to find the depth of the node. However, you have still not found a way to learn the level that you are browsing on.

Continue using the previous file for the exercise or open the file Hierarchies_In_DAX_5.pbix..

To compute the depth of node:

1. Right click on the **Employee** table in the **Fields** pan.
2. Click on **New measure**.

   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Maximum Node Depth (CM) =
MAX ( Employees[Node Depth (CC)] )
```

4. Drag and drop **Maximum Node Depth (CM)** from Employee table to the Values.

Level 1 (CC)	SalesAmount	Maximum Node Depth (CM)
Ali Noorani	**$361,077.95**	**4**
Ali Noorani	$40,221.68	1
Jack Jones	$21,463.76	2
Maryam Hussaini	$151,125.68	4
Muneeba Sirshar	$148,266.83	3
Total	**$361,077.95**	**4**

Notice that at the top most level Ali Noorani has four employees reporting to him. However, at the lower level, the maximum depth for Ali Noorani is 1 because there are no further reporting levels.

Let's browse further to find the browsing depth.

5. Right click on the **Employee** table in the **Fields** pan.
6. Click on **New measure**.

   Measure tools open.
7. Type the following formula in the formula bar and press Enter.

```
Browzing Depth (CM) =
ISINSCOPE ( Employees[Level 1 (CC)] ) + ISINSCOPE ( Employees[Level 2 (CC)] )
    + ISINSCOPE ( Employees[Level 3 (CC)] )
    + ISINSCOPE ( Employees[Level 4 (CC)] )
```

In the expression above:

- *The ISINSCOPE() checks the level of current filter context.*
- *In the result, all the TRUE values are counted.*

8. Drag and drop **Browsing Depth (CM)** from Employee table to the Values.

Level 1 (CC)	SalesAmount	Maximum Node Depth (CM)	Browzing Depth (CM)
Ali Noorani	**$361,077.95**	**4**	**1**
Ali Noorani	$40,221.68	1	2
Jack Jones	$21,463.76	2	2
Maryam Hussaini	$151,125.68	4	2
Muneeba Sirshar	$148,266.83	3	2
Total	**$361,077.95**	**4**	**0**

Notice that now you can see the level at which you are browsing using the DAX function ISINSCOPE().

Analyzing Sales in Hierarchies

In this exercise, you will learn how to hide the node if the browsing of the hierarchy exceeded the depth of node.

Continue using the previous file for the exercise or open the file Hierarchies_In_DAX_6.pbix..

To find the sales for each node:

1. Right click on the **Employee** table in the **Fields** pan.
2. Click on **New measure**.

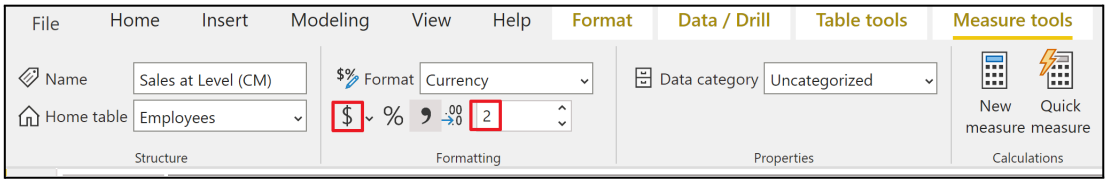
   Measure tools open.
3. Type the following formula in the formula bar and press Enter.

```
Sales at Level (CM) =
IF (
    MAX ( Employees[Node Depth (CC)] ) < [Browzing Depth (CM)],
    BLANK (),
    SUM ( Sales[SalesAmount] )
)
```

In the expression above:

- *If the maximum Employee Node Depth (CC) is less than the browsing depth then BLANK() is returned.*
- *Otherwise the sum of the SalesAmount is returned.*

4. Click on the "$" in the formatting section.
5. Type "2" in the text box.

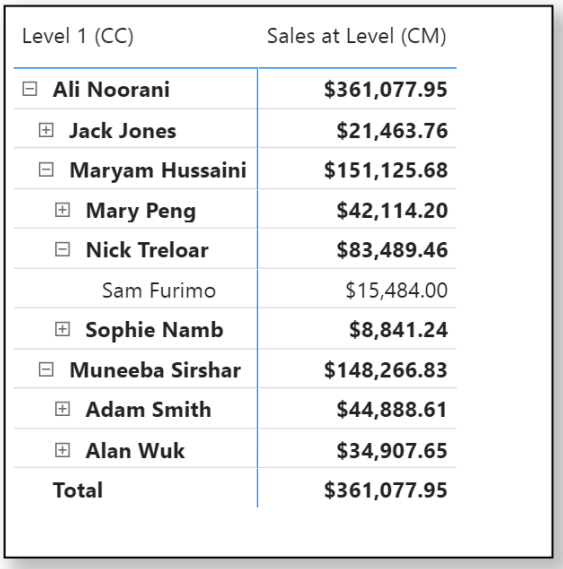


6. Drag and drop **Sales at level (CM)** from Employee table to the Values

Level 1 (CC)	SalesAmount	Maximum Node Depth (CM)	Browzing Depth (CM)	Sales at Level (CM)
Ali Noorani	**$361,077.95**	**4**	**1**	**361,077.95**
Ali Noorani	$40,221.68	1	2	
Jack Jones	$21,463.76	2	2	21,463.76
Maryam Hussaini	$151,125.68	4	2	151,125.68
Muneeba Sirshar	$148,266.83	3	2	148,266.83
Total	**$361,077.95**	**4**	**0**	**361,077.95**

Now lets do the final trick to get the desired result.

7. Remove **Maximum Node Depth (CM)** and **Browsing Depth (CM)** from the Values of the matrix



Level 1 (CC)	Sales at Level (CM)
⊟ **Ali Noorani**	**$361,077.95**
⊞ **Jack Jones**	**$21,463.76**
⊟ **Maryam Hussaini**	**$151,125.68**
⊞ **Mary Peng**	**$42,114.20**
⊟ **Nick Treloar**	**$83,489.46**
Sam Furimo	$15,484.00
⊞ **Sophie Namb**	**$8,841.24**
⊟ **Muneeba Sirshar**	**$148,266.83**
⊞ **Adam Smith**	**$44,888.61**
⊞ **Alan Wuk**	**$34,907.65**
Total	**$361,077.95**

Chapter 16: DAX Tools and Other Resources

What will you learn in this chapter?

✔ Common errors, causes and debugging

✔ What is DAX Formatter

✔ What is DAX Studio

✔ Downloading and Installing DAX Studio

✔ Connecting DAX studio with Power BI

✔ Getting Started with DAX Studio

✔ Writing Queries in DAX Studio

✔ Top 10 Product's contribution to Sales Revenue

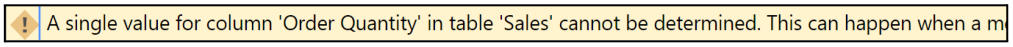
✔ Categorizing Low, Medium and High-value Sales

✔ Top three products in each product subcategory

✔ Using CALCULATETABLE()

Common errors, causes and debugging

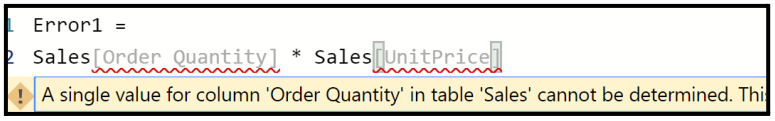
Syntax errors in DAX are displayed with a yellow warning symbol appearing below the formula bar. These errors effect the report execution and are therefore automatically highlighted by Power BI's engine during development.



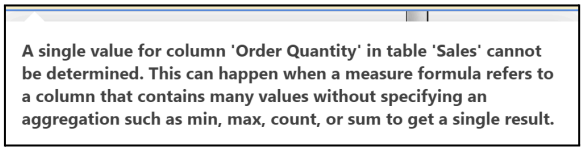
Although, the waring message tells the author how to fix the issue but it does not describe the under lying issue.

Missing Row Context:

Let's consider the following expression to create a measure.



The error message states that:



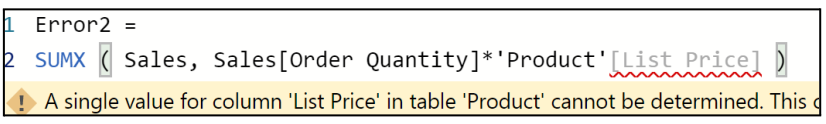
If you analyze this error closely, its saying that the Order Quantity column from the Sales table does not have a row context.

This means that you have tried to use measure for a row by row evaluation but the measures do not create a row context. The row context can only be created if you had created a calculated column or used an iterator when writing a measure expression.

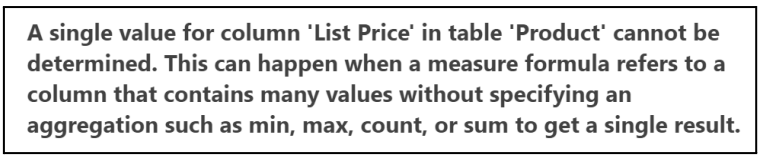
The error message is misleading because:

1. The first part of the error message states that a single value cannot be determined. However, even if the Order Quantity had one row, the same error message will be generated.
2. The second part of the error message suggests the use of SUM(). However, this will not perform row by row evaluation.

Let's consider another scenario where missing row context is an issue:



Notice that this measure generates a similar message.



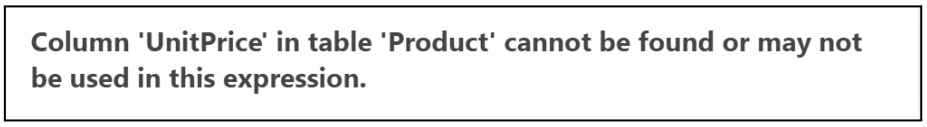
The error message is same as the previous example despite the fact that the row context exists but in different tables. In this case the error message is misleading because no aggregation is suitable for this. Infect a RELATED() function will solve the issue.

Wrong Column Reference:

Consider the following expression:

```
Error3 =
SUMX(Sales, Sales[Order Quantity]*Product[UnitPrice])
```

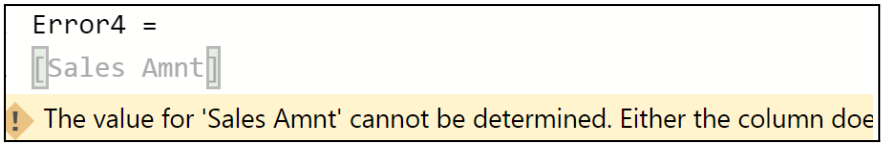
The error message in this case will be:



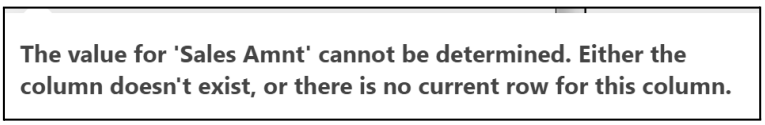
Notice that the error message generated in this case is different because the column Unit Price does not exist in the Product table.

Wrong Measure Reference:

Consider the following expression:



The error message in this case will be:



The problem here is that the measure is not correctly referenced. However, it is very difficult to understand the issue from the warning.

What is DAX Formatter

DAX Formatter is a tool developed by SQLBI that formats DAX code into clean and readable code.

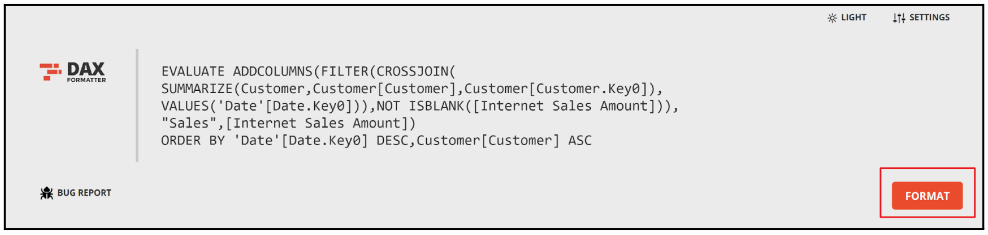
The syntax used for formatting codes can be found at: https://www.sqlbi.com/articles/rules-for-dax-code-formatting/

To learn how to use **DAX Formatter**:

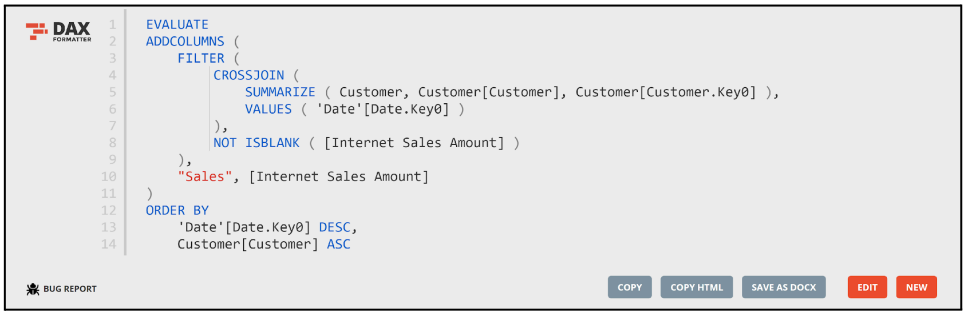
1. Open https://www.daxformatter.com/.
2. Copy the following code from DAX_Tools_And_Other_Resources_1.txt file.

```
EVALUATE ADDCOLUMNS(FILTER(CROSSJOIN(
SUMMARIZE(Customer,Customer[Customer],Customer[Customer.Key0]),
VALUES('Date'[Date.Key0])),NOT ISBLANK([Internet Sales Amount])),
"Sales",[Internet Sales Amount])
ORDER BY 'Date'[Date.Key0] DESC,Customer[Customer] ASC
```

3. Paste the code in the text area.
4. Click **Format**.



Notice how the readability has been improved using colors and indentation.



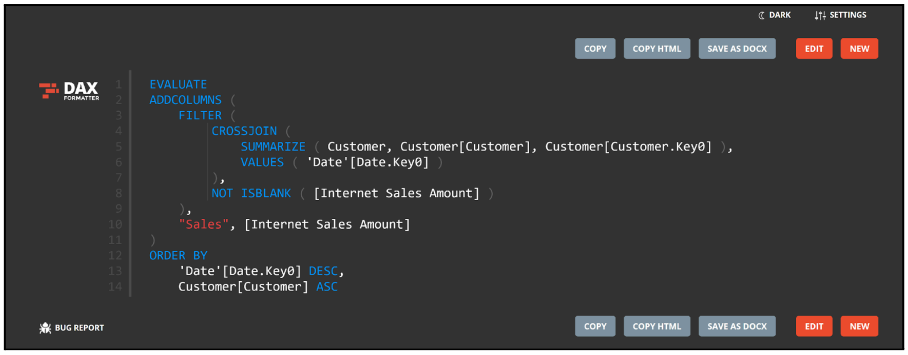
Please note that this is the standard formatting. Power BI also has the option for short line formatting.

```
EVALUATE
ADDCOLUMNS (
    FILTER (
        CROSSJOIN (
            SUMMARIZE (
                Customer,
                Customer[Customer],
                Customer[Customer.Key0]
            ),
            VALUES ( 'Date'[Date.Key0] )
        ),
        NOT ISBLANK ( [Internet Sales Amount] )
    ),
    "Sales", [Internet Sales Amount]
)
ORDER BY
    'Date'[Date.Key0] DESC,
    Customer[Customer] ASC
```

Setting the mode:

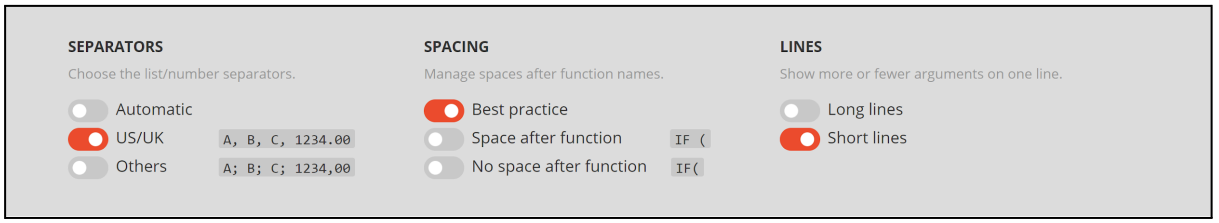
Notice that up till now, you have been working with Light mode.

5. Click on Light at the top to toggle to dark mode.



Notice the change in display colors as the mode is toggled.

6. Click on Setting on the top.



What is DAX Studio

DAX Studio is an incredible free, open-source tool that allows you to directly query your Power BI / Power Pivot data models. It not only helps you to author DAX queries and analyze performance but also to learn the DAX language. DAX studio includes an editor that allows you to write and execute a query. Queries run inside the DAX studio and the results are returned as a table.

Darren Gosbell, an Australian MVP, created the first version of DAX studio, and then he opened it to the public domain. Since then, several developers have added features to it.

DAX Studio can be connected to:

- ✔ Power BI Desktop
- ✔ Power Pivot for Excel
- ✔ Tabular Servers
- ✔ Visual Studio

Useful Features of DAX Studio:

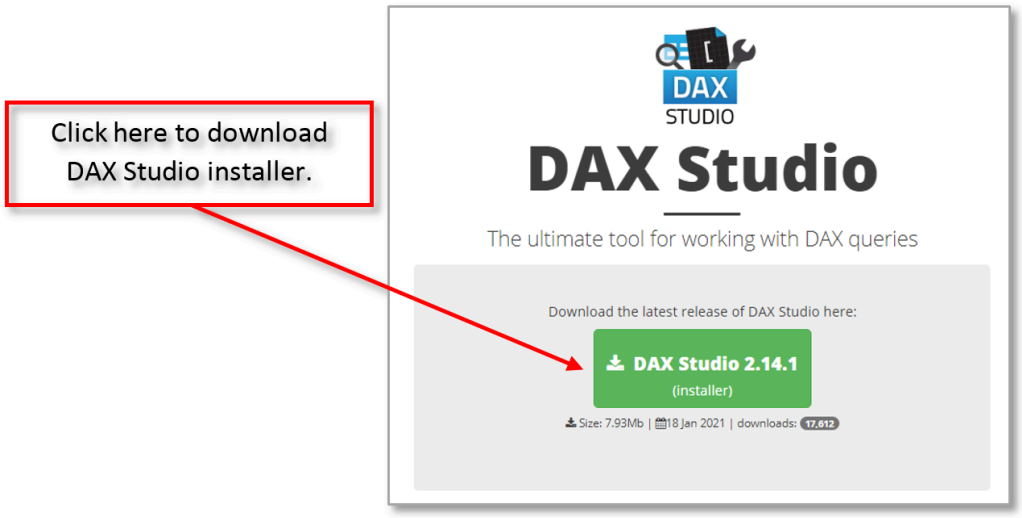
- ✔ Optimize your model performance with the VertiPaq Analyzer integrated with DAX Studio.
- ✔ Visualize the result of DAX formulas that contain 'tables' including 'table expressions'. You can't see the result of such measures while working it in Power BI Desktop, whereas in DAX studio you can explore what are the results being produced.
- ✔ Authoring DAX measures
- ✔ A support for learning DAX language.
- ✔ Query Building.

An important thing to note is that query run in DAX Studio ALWAYS returns a table, unlike a measure created in Power BI, which returns a scalar value.

Downloading and Installing DAX Studio

The latest version of DAX studio is available at: https://daxstudio.org/

It's a free tool and you can trust its source.



Operating system requirements

Following are the operating system requirements for running DAX Studio.

- ✔ Windows 10 (Recommended)/Windows 8 / Windows 8.1/Windows 7)
- ✔ 32 bit or 64 bit operating system
- ✔ Operating system supports the .Net Framework 4.7.1 or later.

Installation Options:

	All Users	**Current User**
Requires Admin rights	Yes	No
Available for all users on the current machine	Yes	No
Excel Add-in available	Yes	Yes
Power BI External Tools integration	Yes	No

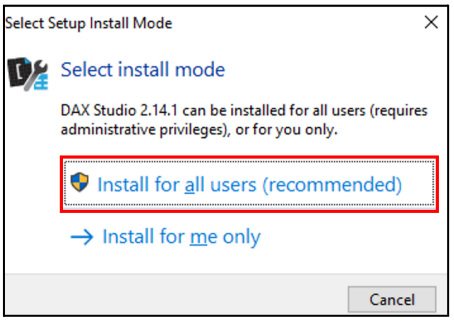
To install DAX Studio:

1. Open https://daxstudio.org/
2. Click on DAX Studio (Installer)

3. Run DaxStudio_Setup.exe

   A pop-up menu appears.

4. Click on Install for all users (Recommended).

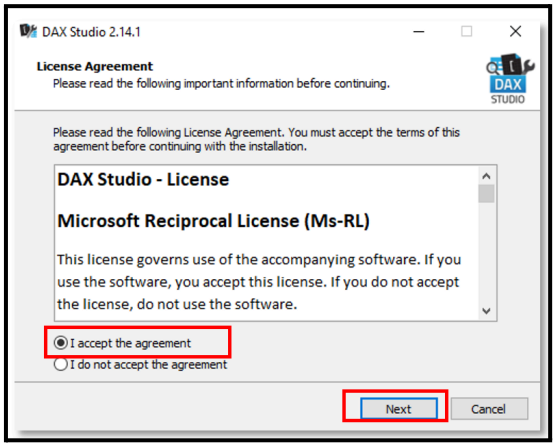


A pop-up verification box appears.

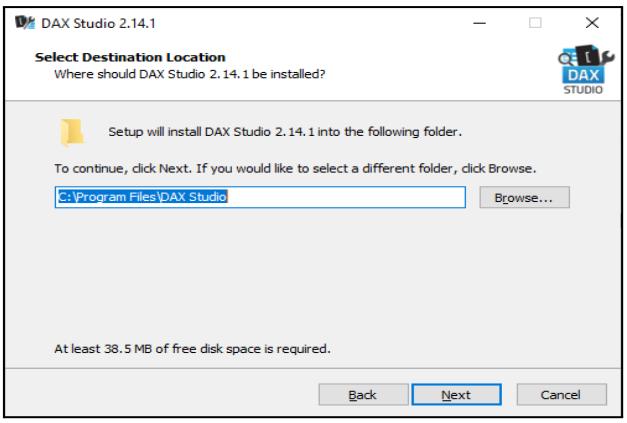
5. Click on Allow access.

Please note to review the agreement.

6. Select “I accept the agreement”
7. Click Ok.

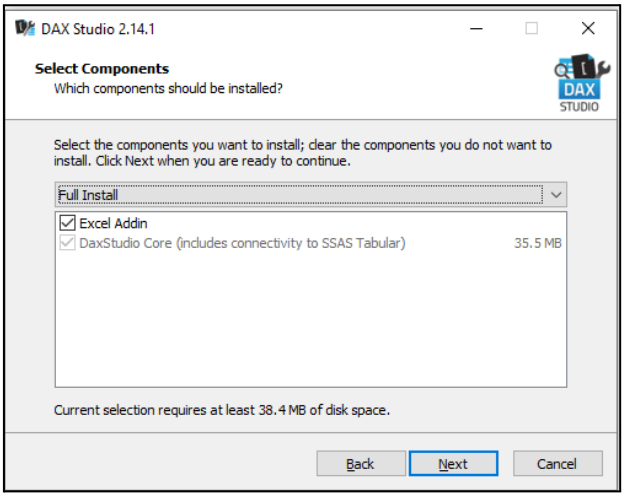


8. Select Destination location.
9. Click Next.

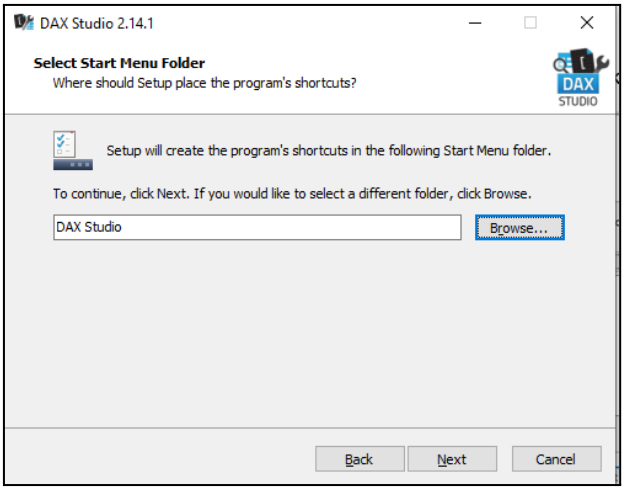


10. Select Excel Add on.

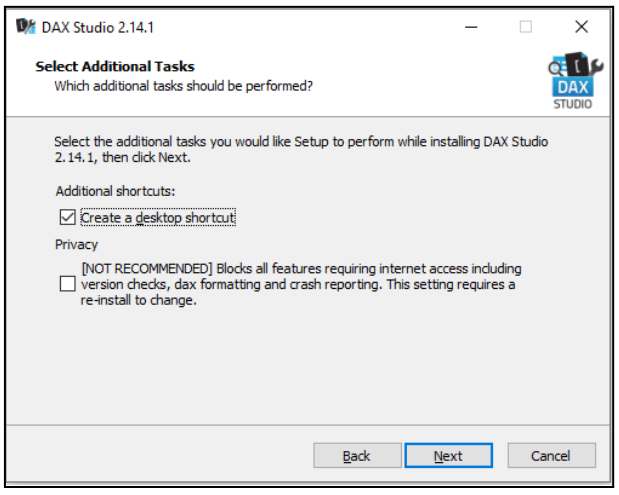
11. Click Next.



12. Click on Next.



13. Select “Create a desktop shortcut”.
14. Click Next.



15. Click Finish.

DAX studio has now been installed on your local system.

Connecting DAX studio with Power BI

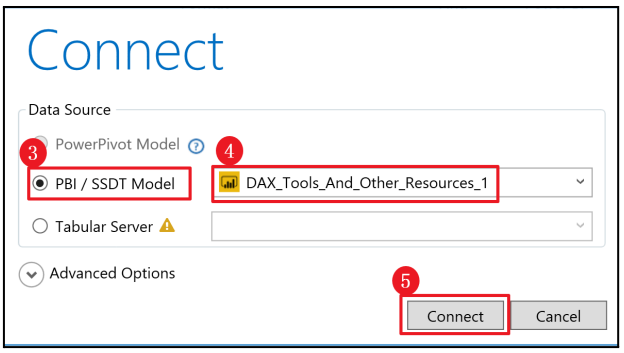
DAX Studio can be connected with Power BI using 2 methods:

Launching it directly from your system

To launch DAX Studio directly from your system:

1. Open DAX_Tools_And_Other_Resources_1.pbix..
2. Click on **DAX Studio** shortcut on desktop.

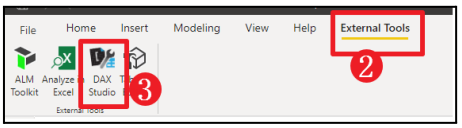
   A pop-up menu appears.
3. Select the check box at the left of PBI/SSDT Model.
4. Select the Power BI file from drop down.
5. Click **Connect.**



Launching from Power BI Desktop

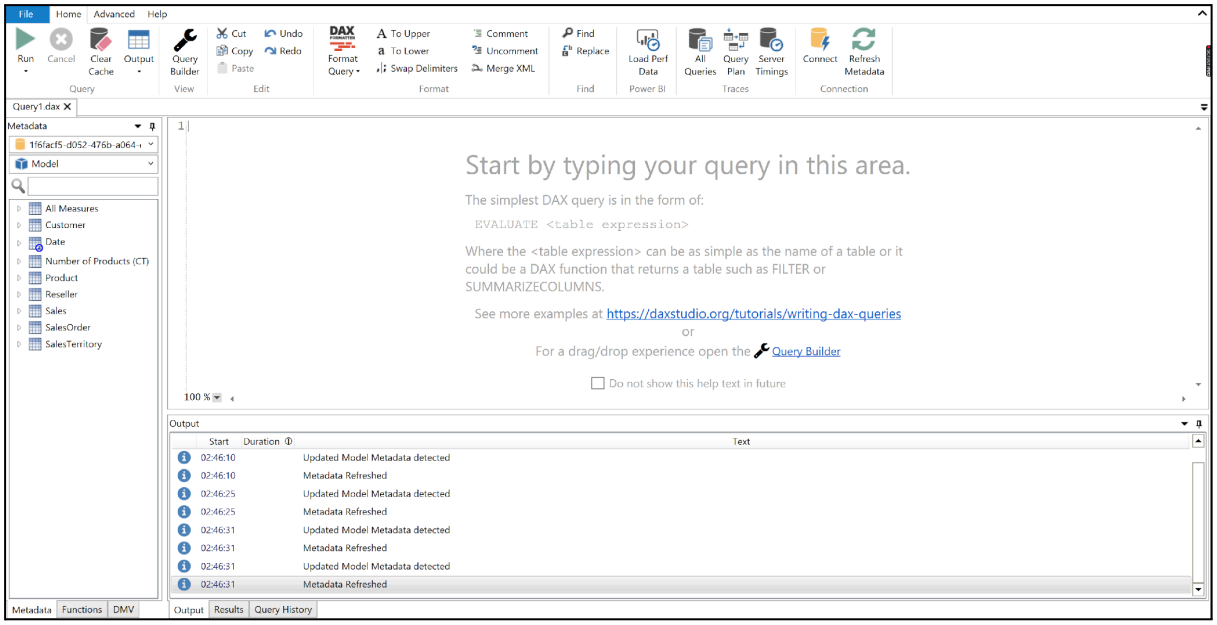
To Launch DAX Studio from Power BI desktop:

1. Open DAX_Tools_And_Other_Resources_1.pbix..
2. Click on **External tools** in the top ribbon.
3. Click on **DAX Studio.**



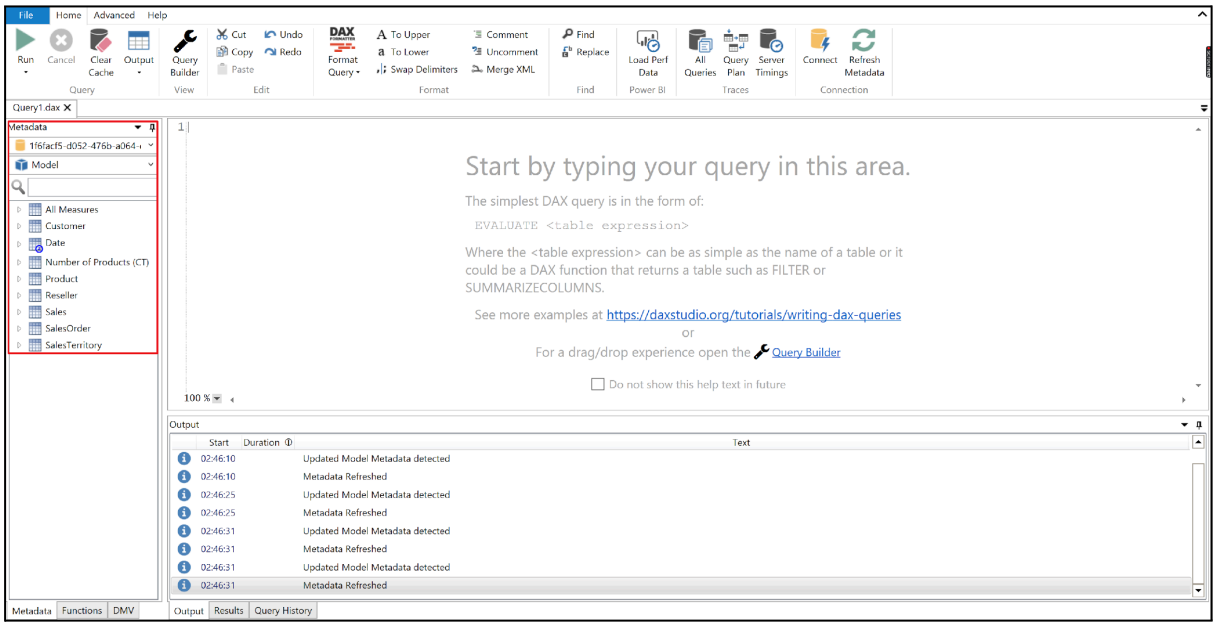
Getting Started with DAX Studio

Whatever method you use; you can start working with the database using its features accessible from the user interface once you open DAX Studio.



Let's walk through each of the panel:

Metadata Panel

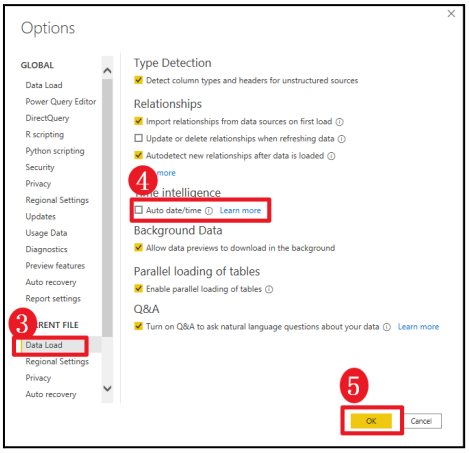


In the Metadata panel, you can find all the tables, columns and DAX measures in your data model. If you have marked a table as **Date Table**, a specific clock icon appears on it. It's always a good idea to disable the auto Date/time option in **Power BI File.**

To Disable Date/time option:

1. Click on File**.**

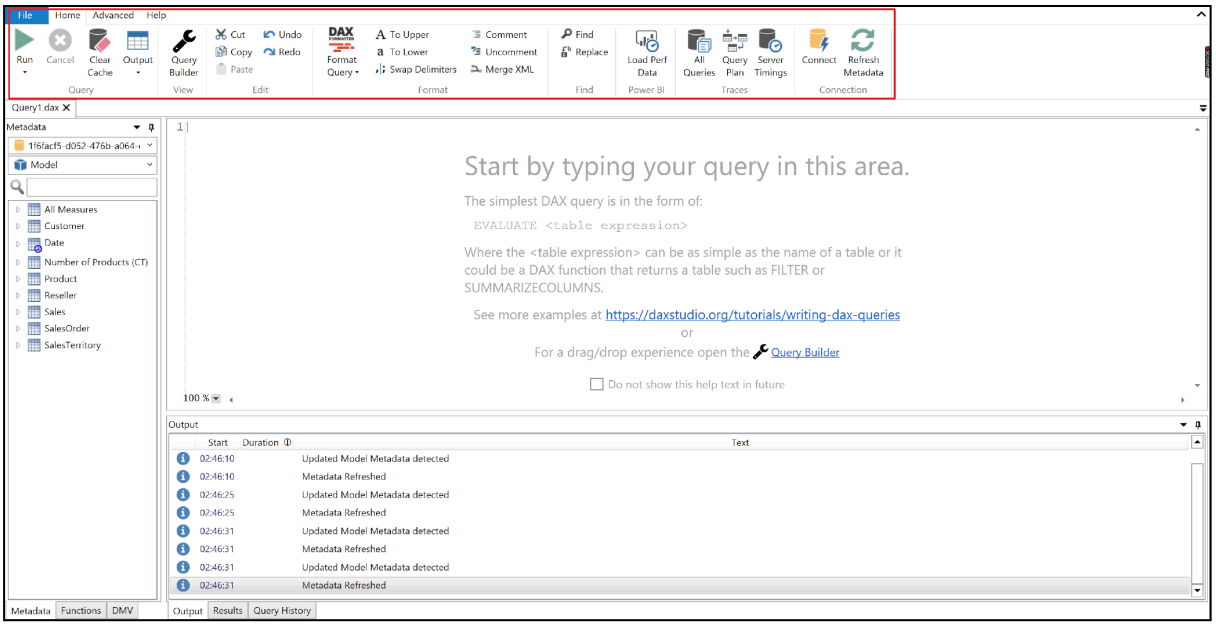
2. Click on **Options and Settings > Options.**
3. Click on Data Load.
4. Disable the check box next to Auto Date/Time.
5. Click ok.



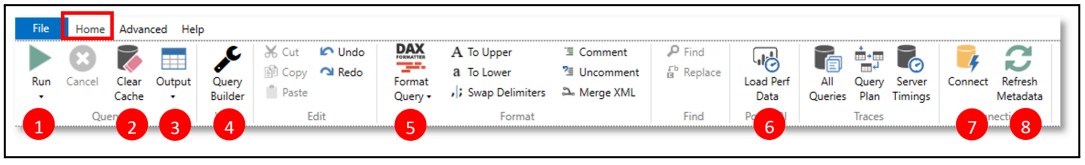
If Date/time is not disabled, DAX Studio will also show all the local date tables created automatically by Power BI.

The Ribbon

You can access all the features of DAX Studio through the ribbon. Here is the preview of its most important options.



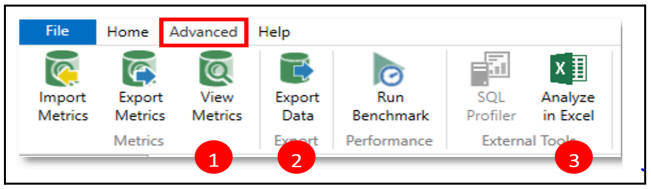
Home Tab:



The functions for each of the buttons are:

1. Executes the query.
2. Clears the cache for the current database.
3. Output target to specify where you want to send the query results.
4. Displays a drag and drop query builder.
5. Uses daxformatter.com to provide a nicely formatted query.
6. Imports performance data from Power BI Performance Analyzer.
7. Shows the connection to the Power BI Desktop file.
8. Use to refresh the metadata manually.

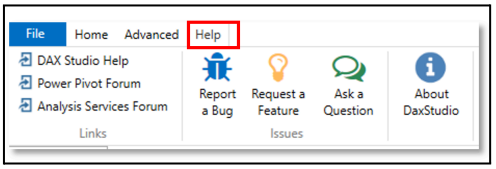
Advanced Tab:



The functions for each of the buttons are:

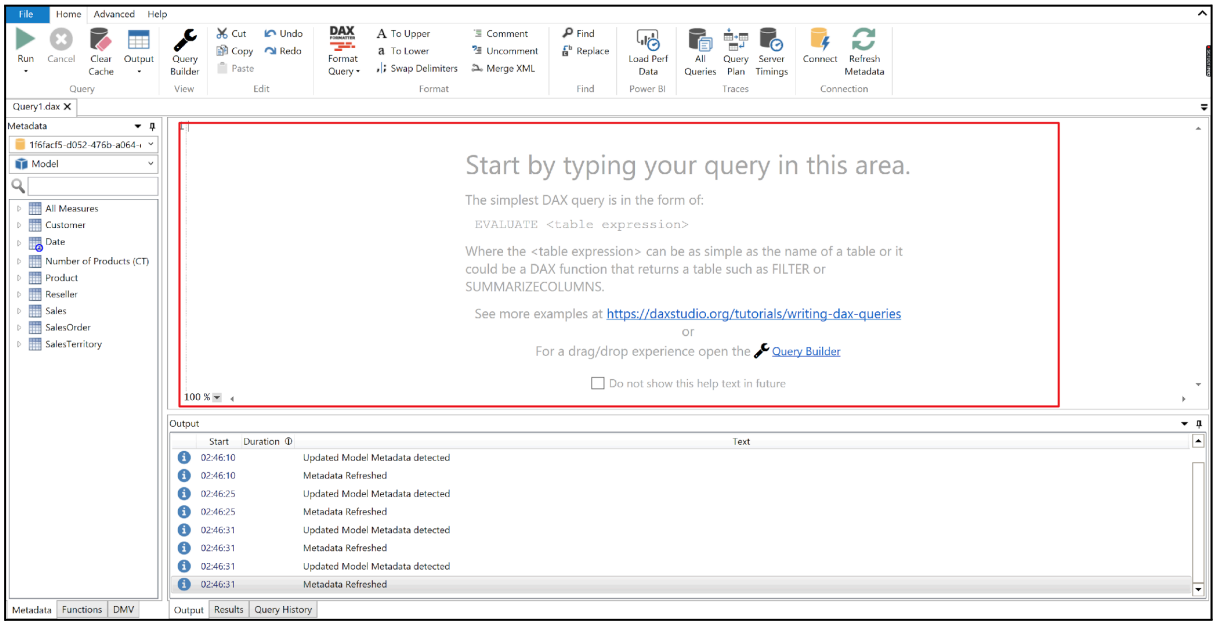
1. Display information and matrices about your model.
2. Exports data to CSV files or SQL Server tables.
3. Launch an Excel file connected to the current data source.

Help Tab:



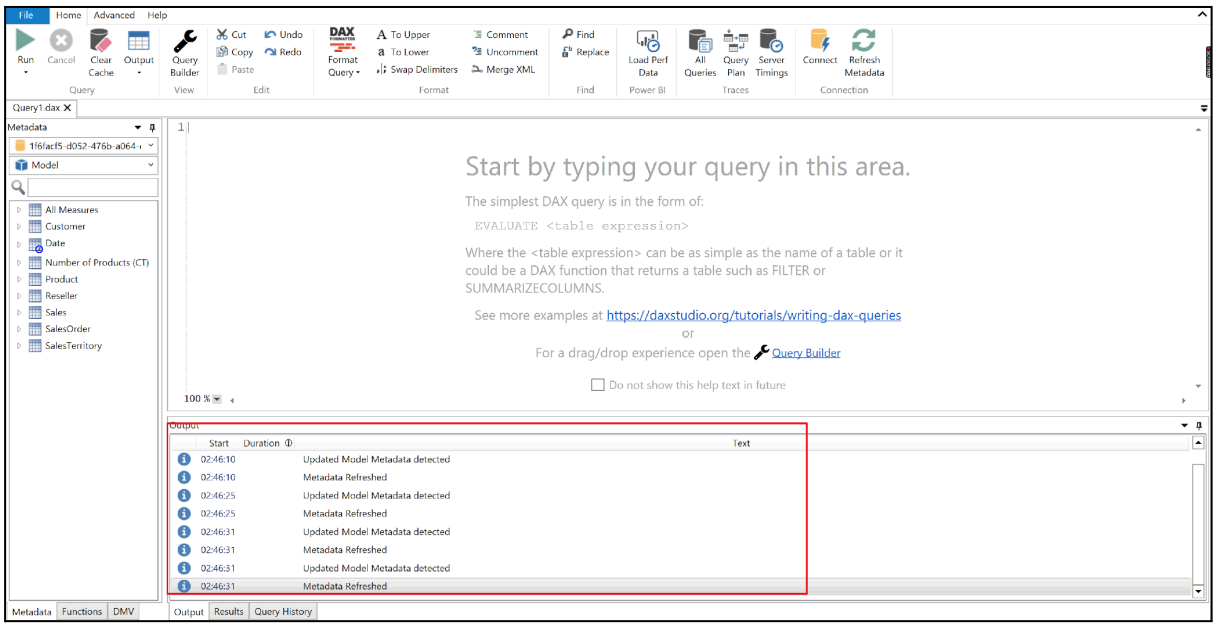
This tab is mainly about general info and help.

The Query Pane:



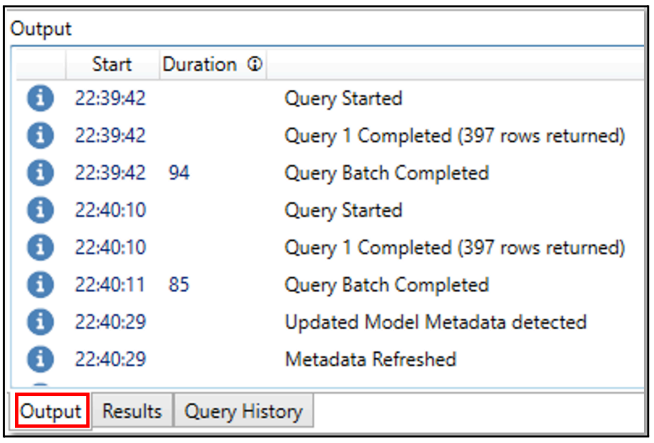
This is the query pane where you write, edit, format and view your queries.

Output, results and history pane



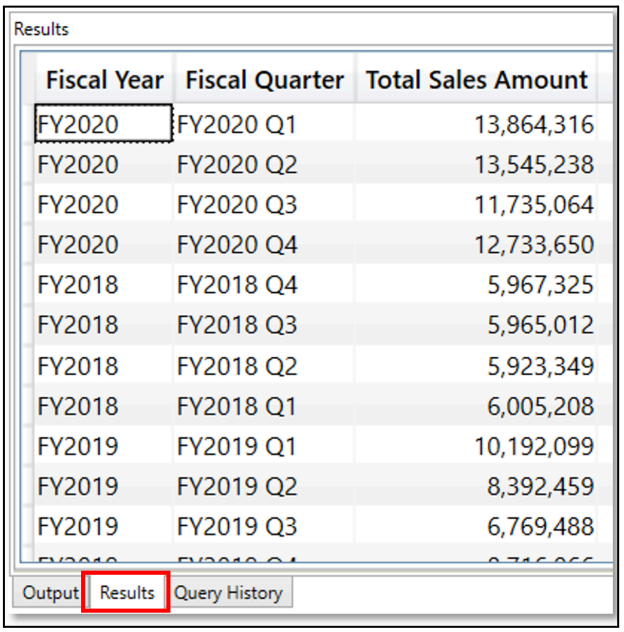
This pane has three tabs:

Output:



This tab shows the general information on query run time.

Results:



Results tab gives the output of the query.

Query History:

The Query History tab keeps a log of currently executed queries.

Writing Queries in DAX Studio

The most straightforward DAX query is in the form of **EVALUATE.**

The table expression can be as simple as the name of a table like in our case, it can be **EVALUATE (Customers)** or it could be a DAX function that returns a table such as **FILTER** or **SUMMARIZECOLUMNS.**

EVALUATE statement:

EVALUATE is a DAX statement containing a table expression and is needed to execute a query. A query can also have multiple EVALUATE statements, but the key point here is that it requires a keyword.

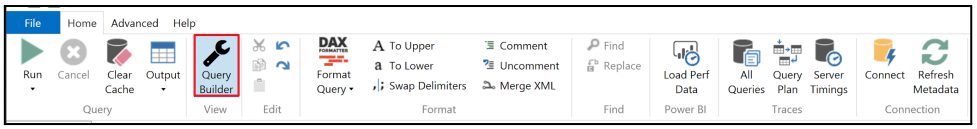
Continue using the previous file for the exercise or open the file DAX_Tools_And_Other_Resources_2.pbix..

Query Builder:

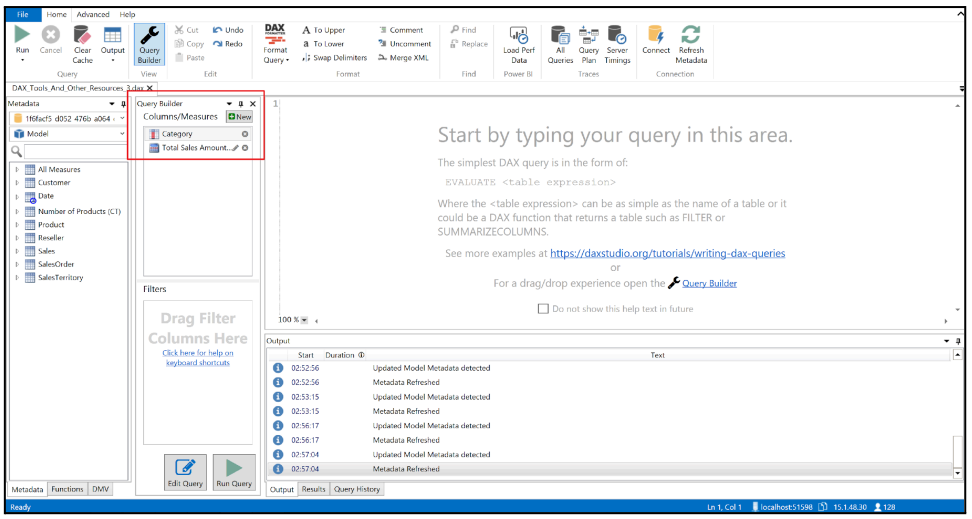
If you are learning DAX, or if you want to see how a query is created, **Query Builder** is the best option in DAX Studio. It provides a graphical user interface that makes it easier to build queries. Let's use query builder to get total sales amount by product category.

To use DAX Studio for this:

1. Click on the **Home Tab** in the Ribbon.
2. Click on **Query Builder** in the View Section.

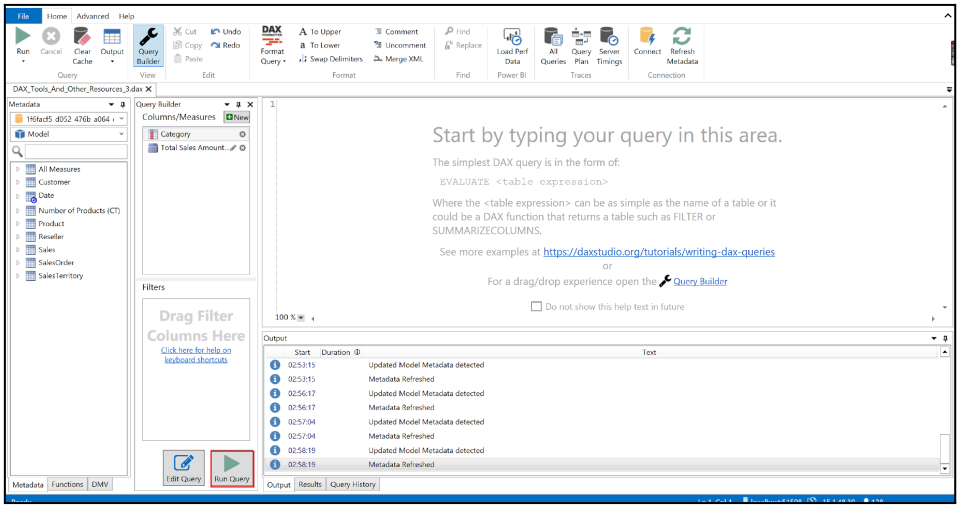


3. Drag and drop **Category** Column from Products to Columns/Measures area
4. Drag and drop **Total Sales Amount (CM)** from All Measures to Columns/Measures area

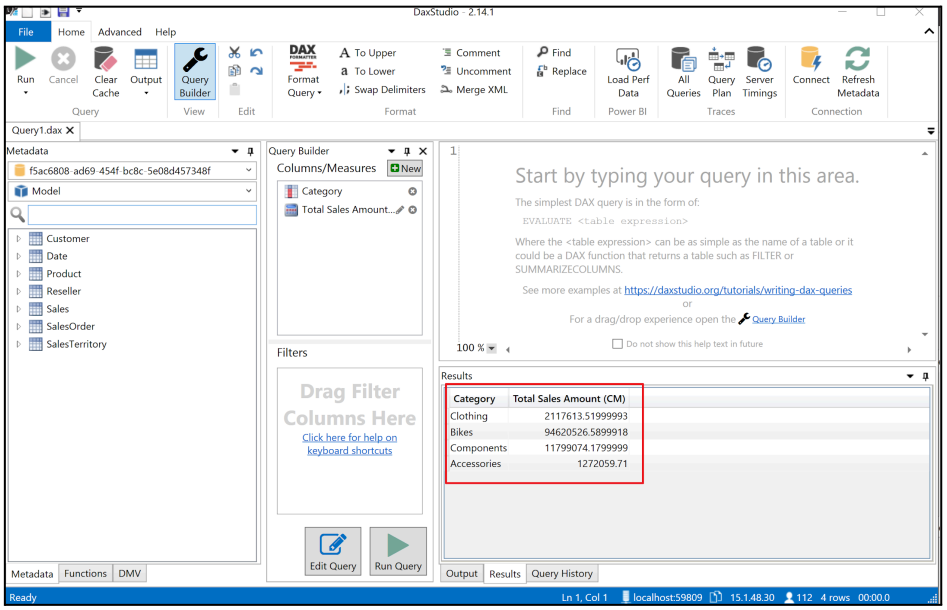


If you want to add any filter you can do so in filter area.

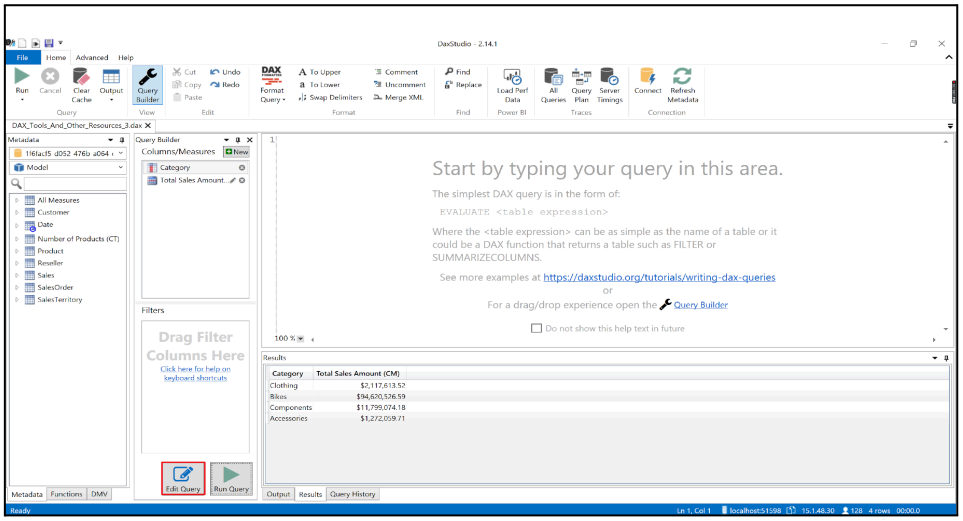
5. Press **Run Query** button.



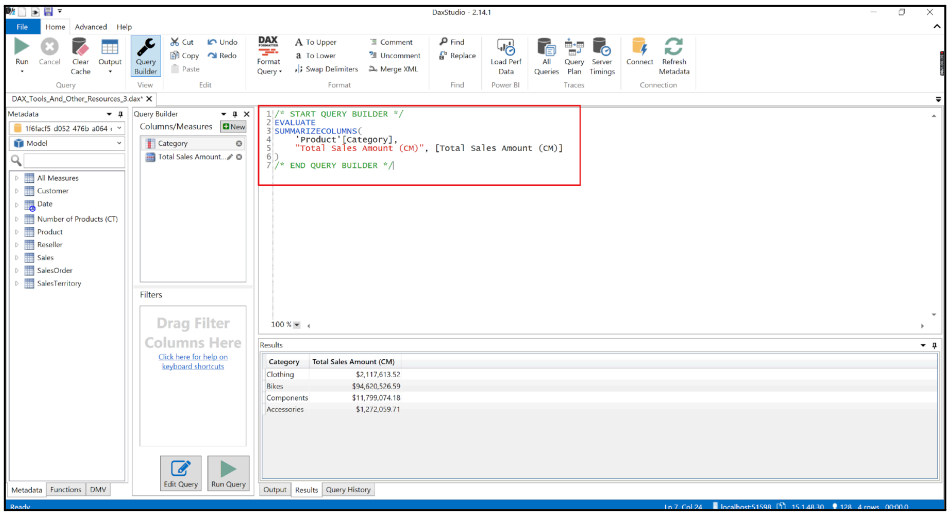
*Notice the output in **Results** area.*



6. Click on **Edit Query** button.

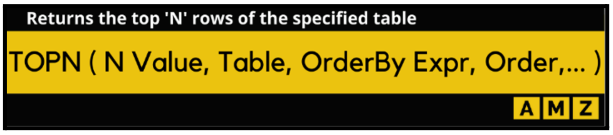


Notice the automatic code generated in Query Pane.



Top 10 Product's contribution to Sales Revenue

In this exercise, you will learn how to use TOPN() to find the top 10 products by sales revenue in DAX Studio.



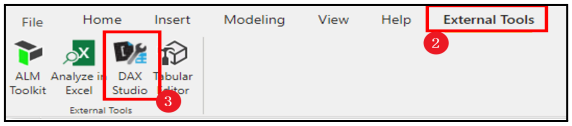
Continue using the previous file for the exercise or open the file DAX_Tools_And_Other_Resources_3.dax..

You will need to connect with DAX_Tools_And_Other_Resources_3.pbix

TOPN() needs a table (or a table expression) to iterate row by row and calculate the top rows. We will use SUMMARIZE() to create a table expression for product wise sales amount. The DAX function SUMMARIZE() aggregates and groups data by leveraging existing relationships in your data model.

To find Top 10 Product's contribution to Sales Revenue:

1. Open DAX_Tools_And_Other_Resources_3.pbix
2. Click on **External Tools** tab in the **Ribbon**.
3. Click on **DAX Studio**.



Please close the query builder if it is already open.

4. Erase the existing code in the Query Pan.
5. In the Query Pan, type the following code.

```
EVALUATE
TOPN(
    10,
    SUMMARIZE(
        Sales,
        PRODUCT[ProductKey],
        "Total Sales Amount", [Total Sales Amount (CM)]
    ),
    [Total Sales Amount (CM)]
)
ORDER BY [Total Sales Amount (CM)] DESC
```

6. Click on **Run** in the File Tab.

The Results are generated in the Results Pan.

ProductKey	Total Sales Amount
359	2589363.72
361	2265485.46999999
353	2160981.64
363	1957528.18
355	1914547.85
357	1906248.54
314	1847818.36
358	1811229.74000001
583	1774883.49

The Query generated in DAX Studio is always a table.

To create a calculated table in Power BI:

7. Copy the Code from line 2-10 in **DAX Studio**.
8. Open **Power B Desktop**.
9. Click on the **Modeling tab** in the Ribbon.
10. Click on **New Table**.
11. Paste the code in the formula bar and press Enter

```
Testing DAX Studio Table (CT) =
TOPN(
    10,
    SUMMARIZE(
        Sales,
        PRODUCT[ProductKey],
        "Total Sales Amount", [Total Sales Amount (CM)]
    ),
    [Total Sales Amount (CM)]
)
```

Let's create a calculated measure using an iterator so that data can be further sliced and diced.

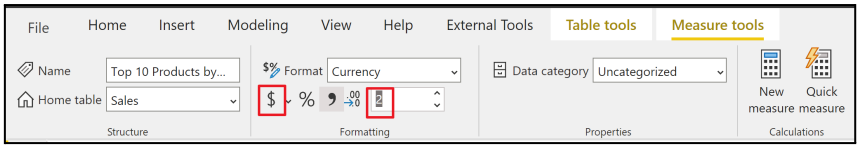
12. Right click on **Testing DAX Studio (CT)** Table in the Fields.
13. Click on **New measure.**

Measure tools open.

14. Type the following formula in the formula bar and press Enter.

```
Top 10 Products by Sales Amount (CM) =
SUMX (
    TOPN (
        10,
        SUMMARIZE (
            Sales,
            PRODUCT[ProductKey],
            "Total Sales Amount", [Total Sales Amount (CM)]
        ),
        [Total Sales Amount (CM)]
    ),
    [Total Sales Amount (CM)]
)
```

15. Click on the "$" and type 2 in the box.



16. Click on the **matrix** visual in the visualization pane.
17. Drag and drop **Category** and **Subcategory** from the Products table to the rows.
18. Drag and drop **Total Sales Amount (CM)** from All Measures to the Values
19. Drag and drop **Top 10 Products by Sales Amount (CM)** from **Testing DAX Studio (CT)** to the Values.
20. Expand **Bikes** Category.

Category	Total Sales Amount (CM)	Top 10 Products by Sales Amount (CM)
⊞ **Accessories**	**$1,272,059.71**	**$899,147.77**
⊟ **Bikes**	**$94,620,526.59**	**$19,997,183.50**
Mountain Bikes	$36,445,444.52	$19,410,100.16
Road Bikes	$43,878,791.78	$15,397,667.00
Touring Bikes	$14,296,290.29	$11,123,826.15
⊞ **Clothing**	**$2,117,613.52**	**$1,122,081.94**
⊞ **Components**	**$11,799,074.18**	**$3,162,480.56**
Total	**$109,809,274.00**	**$19,997,183.50**

Notice that contribution of the top 10 products in each category and subcategory is being shown.

To find the percentage contribution:

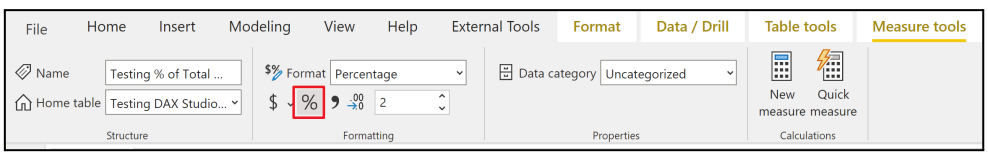
21. Right click on **Testing DAX Studio (CT)** Table in the Fields.
22. Click on **New measure**.

    Measure tools open.

23. Type the following formula in the formula bar and press Enter.

```
Testing % of Total Sales (CM) =
DIVIDE ( [Top 10 Products by Sales Amount (CM)], [Total Sales Amount (CM)], 0 )
```

24. Click on % in the formatting section.



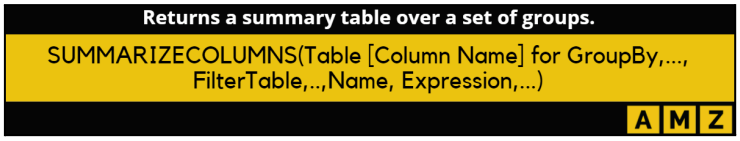
25. Drag and drop **Testing % of Total Sales (CM)** from **Testing DAX Studio (CT)** to the Values.

Category	Total Sales Amount (CM)	Top 10 Products by Sales Amount (CM)	Testing % of Total Sales (CM)
⊟ **Accessories**	**$1,272,059.71**	**$899,147.77**	**70.68 %**
Bike Racks	$237,096.21	$237,096.21	100.00 %
Bike Stands	$39,591.00	$39,591.00	100.00 %
Bottles and Cages	$64,274.95	$64,274.95	100.00 %
Cleaners	$18,407.03	$18,407.03	100.00 %
Fenders	$46,619.58	$46,619.58	100.00 %
Helmets	$484,049.97	$484,049.97	100.00 %
Hydration Packs	$105,826.62	$105,826.62	100.00 %
Locks	$16,225.22	$16,225.22	100.00 %
Pumps	$13,514.71	$13,514.71	100.00 %
Tires and Tubes	$246,454.42	$239,029.30	96.99 %
⊞ **Bikes**	**$94,620,526.59**	**$19,997,183.50**	**21.13 %**
⊞ **Clothing**	**$2,117,613.52**	**$1,122,081.94**	**52.99 %**
⊞ **Components**	**$11,799,074.18**	**$3,162,480.56**	**26.80 %**
Total	**$109,809,274.00**	**$19,997,183.50**	**18.21 %**

Notice that the percent contribution of to 10 products sales is being shown in the visual.

Categorizing Low, Medium and High-value Sales

In this exercise, you will learn how to categorize monthly sales for a product as low, medium, high and very high using SUMMARIZECOLUMNS() function.



This function provides more capabilities and was introduced by Microsoft as a replacement to SUMMARIZE().

Continue using the previous file for the exercise or open the file DAX_Tools_And_Other_Resources_4.dax..

You will need to connect with DAX_Tools_And_Other_Resources_4.pbix

To categorize the products:

1. Open DAX_Tools_And_Other_Resources_4.pbix..
2. Right click on the **Sales** table in the Fields Pan.
3. Click on **New Column**.

   Column tools open.
4. Type the following formula in the formula bar and press Enter

```
End of Month (CC) =
EOMONTH ( Sales[OrderDate], 0 )
```

5. Click on **External Tools** in the Ribbon.
6. Click on **DAX Studio.**
7. Erase the existing code in the Query Pan.
8. In the Query Pan, type the following code.

```
EVALUATE

SUMMARIZECOLUMNS(
        PRODUCT[ProductKey],
        sales[End of Month (CC)],

    FILTER(
    PRODUCT,
    PRODUCT[Category]<>"Accessories"
    ),
    "Total Sales", [Total Sales Amount (CM)]
)
```

9. Click on **Run** in the File Tab.

The Results are generated in the Results Pan.

ProductKey	End of Month (CC)	Total Sales
218	6/30/2018 12:00:00 AM	552.9
218	5/31/2018 12:00:00 AM	756.25
218	4/30/2018 12:00:00 AM	427.75
218	2/28/2018 12:00:00 AM	1009.31
218	1/31/2018 12:00:00 AM	256.5
218	11/30/2017 12:00:00 AM	1234.81
218	9/30/2017 12:00:00 AM	895.02
218	8/31/2017 12:00:00 AM	711.29
218	7/31/2017 12:00:00 AM	216.6

10. Copy the Code from line 3-12 in DAX Studio.
11. Open **Power B Desktop**.
12. Click on the **Modeling** tab in the Ribbon.
13. Click on **New Table**.
14. Paste the code in the formula bar and press Enter

```
High Value Sales (CT) =
SUMMARIZECOLUMNS (
    Sales[ProductKey],
    sales[End of Month (CC)],
    FILTER ( 'Product', 'Product'[Category] <> "Accessories" ),
    "Total Sales", [Total Sales Amount (CM)]
)
```

Notice that we are filtering the product table and not including Accessories category.

Now let's create a calculated column in "High value Sales (CT)" to classify the sales.

14. Right click on the **High value Sales (CT)** table in the Fields Pan.
15. Click on **New Column**.

Column tools open.

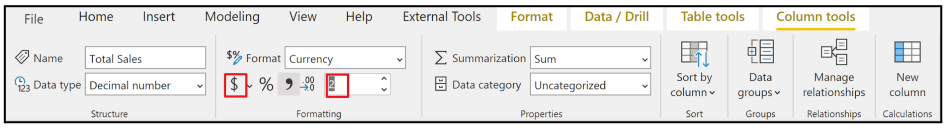
16. Type the following formula in the formula bar and press Enter.

```
Sales Value Classification (CC) =
SWITCH (
    TRUE (),
    'High Value Sales (CT)'[Total Sales] < 10000, "1) Low",
    'High Value Sales (CT)'[Total Sales] < 25000, "2) Medium",
    'High Value Sales (CT)'[Total Sales] < 100000, "3)High",
    "4) V. High"
)
```

17. Click on **Total Sales** in **High value Sales (CT)** table in the Fields Pan.

    Column Tools Open.

18. Click on "$" and type "2" in the text box in formatting section.

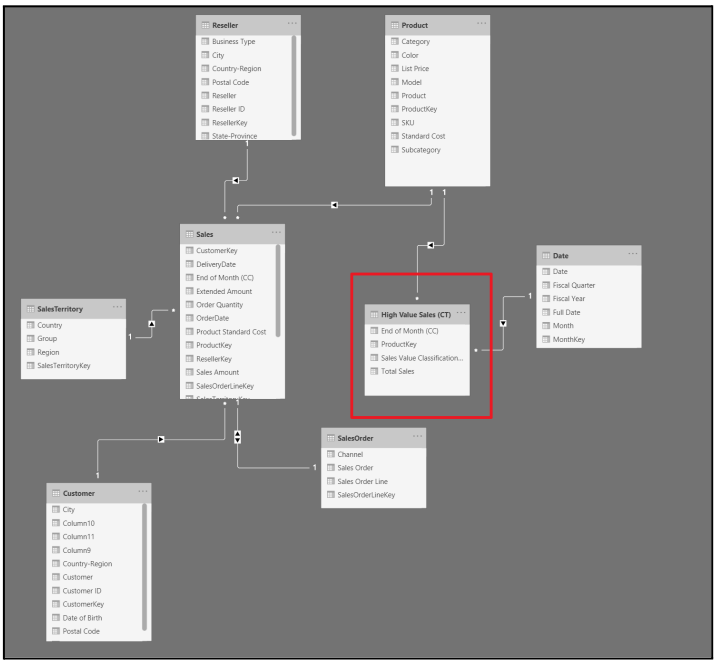


19. Click on the **Model** tab in the left ribbon.
20. Drag the **End of Month (CC)** column from High Value Sales (CT) and drop to the **Date** in the Date table.

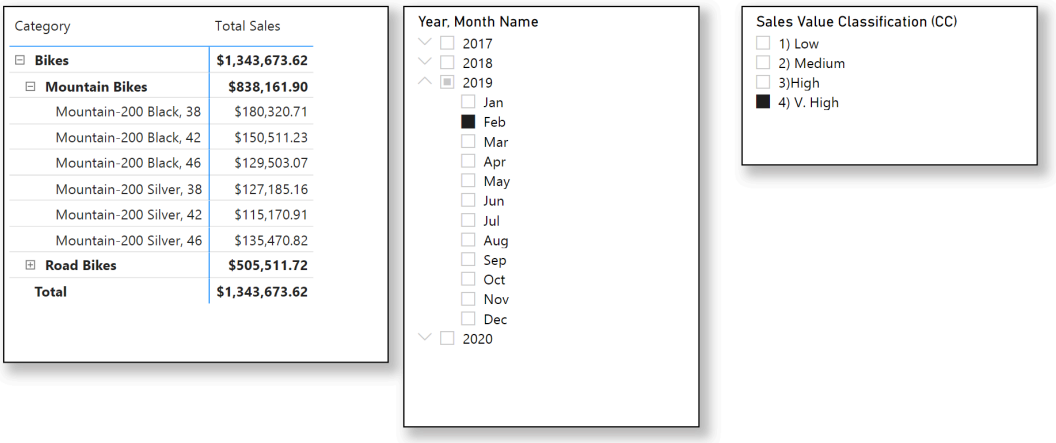
Please note that a many to one relationship between High Value Sales (CT) and Date table is created.

21. Drag the **ProductKey** column from High Value Sales (CT) and drop to the **ProductKey** in the Product table.

    Please note that a many to one relationship between High Value Sales (CT) and Date table is created.



22. Click on **Report** View on the left Ribbon.
23. Click on the **matrix** visual in the visualization pane.
24. Drag and drop **Category** and **Subcategory** from the **Product** table to the rows.
25. Drag and drop **Total Sales** from **High Value Sales (CT)** in Values
26. Click on **Slicer** visual in the Visualization Pane.
27. Drag and drop **Sales Value Classification (CC)** from **High Value Sales (CT)** to the Fields.
28. Click on **Slicer** visual in the Visualization Pane.
29. Drag and drop **Year** and **Month Name** from **Date** to the Fields.
30. Click on the Arrow at the top and select List.
31. In the **Year, Month Name** Slicer, Select 2017>Oct,Nov,Dec.
32. In the **Sales Value Classification (CC)** slicer, Select 2) Medium.

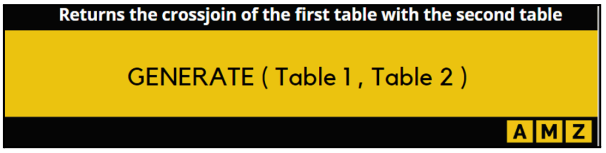


SUMMARIZE() vs. SUMMARIZECOLUMNS():

	SUMMARIZE()	SUMMARIZECOLUMNS()
Returns a summary table with implicit cross join from any Table[Column] in the data model	No	Yes
Replacement of ADDCOLUMNS()/SUMMARIZE() pattern and SUMMARIZE() function	No	Yes
Multiple filter tables (expressions) can be added	No	Yes
Recommended for data models with multiple fact tables	No	Yes

Top three products in each product subcategory

In this exercise, you will learn how to extract top three products in each subcategory using GENERATE() and TOPN() functions.



Continue using the previous file for the exercise or open the file DAX_Tools_And_Other_Resources_5.dax..

You will need to connect with DAX_Tools_And_Other_Resources_5.pbix

To extract top three products in each subcategory using GENERATE() and TOPN() functions:

1. Click on **External Tools** in the Ribbon in Power BI Desktop.
2. Click on **DAX Studio.**
3. Erase the existing Code in the Query Pan.
4. In the Query Pan, type the following code.

```
1  EVALUATE
2
3  FILTER(
4      GENERATE(
5          VALUES(PRODUCT[Subcategory]),
6
7          TOPN(
8              3,
9              ADDCOLUMNS(
10                 VALUES(Product[Product]),
11                 "Sales Amount", [Total Sales Amount (CM)]
12                 ),
13                 [Total Sales Amount (CM)]
14             )
15     ),
16     [Total Sales Amount (CM)]> 0
17 )
```

5. Click on **Run** in the File Tab.

The Results are generated in the Results Pan.

Subcategory	Product	Sales Amount
Road Bikes	Road-250 Black, 48	2347655.99000002
Road Bikes	Road-250 Black, 44	2516857.34000002
Road Bikes	Road-250 Black, 52	2012447.65000002
Touring Bikes	Touring-1000 Blue, 60	1721242.31000001
Touring Bikes	Touring-1000 Yellow, 60	1518132.79000001
Touring Bikes	Touring-1000 Blue, 46	1586953.36000001
Jerseys	Long-Sleeve Logo Jersey, M	115248.620000001
Jerseys	Long-Sleeve Logo Jersey, L	198754.009999997
Road Frames	ML Road Frame-W - Yellow, 44	485700.79

6. Copy the Code from line 3-17 in DAX Studio.

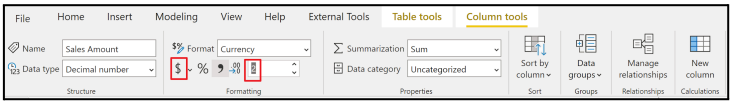
7. Open **Power B Desktop**.
8. Click on the **Modeling** tab in the Ribbon.
9. Click on **New Table**.
10. Paste the code in the formula bar and press Enter

```
Top 3 Products in each SubCategory (CT) =
FILTER (
    GENERATE (
        VALUES ( PRODUCT[Subcategory] ),
        TOPN (
            3,
            ADDCOLUMNS (
                VALUES ( Product[Product] ),
                "Sales Amount", [Total Sales Amount (CM)]
            ),
            [Total Sales Amount (CM)]
        )
    ),
    [Total Sales Amount (CM)] > 0
)
```

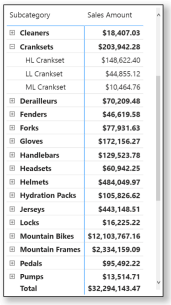
11. Click on **Sales Amount** in **Top 3 Products in each SubCategory (CT)** table in the Fields Pan.

    Column Tools Open.

12. Click on "$" and type "2" in the text box in formatting section.



13. Click on the **matrix** visual in the visualization pane.
14. Drag and drop **Subcategory** and **Product** from the **Top 3 Products in each SubCategory (CT)** table to the rows.
15. Drag and drop **Sales Amount** from the **Top 3 Products in each SubCategory (CT)** table to **Values.**
16. Expand 'Cranksets'.

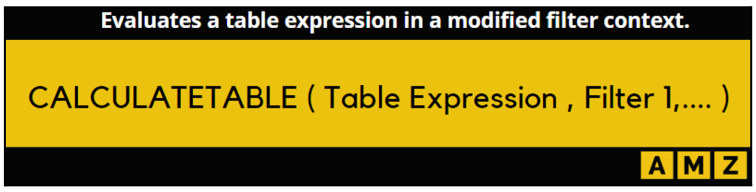


Subcategory	Sales Amount
Cleaners	$18,407.03
Cranksets	$203,942.28
HL Crankset	$148,622.40
LL Crankset	$44,855.12
ML Crankset	$10,464.76
Derailleurs	$70,209.48
Fenders	$46,619.58
Forks	$77,931.63
Gloves	$172,156.27
Handlebars	$129,523.78
Headsets	$60,942.25
Helmets	$484,049.97
Hydration Packs	$105,826.62
Jerseys	$443,148.51
Locks	$16,225.22
Mountain Bikes	$12,103,767.16
Mountain Frames	$2,334,159.09
Pedals	$95,492.22
Pumps	$13,514.71
Total	$32,294,143.47

Please note that for the categories which have less than three products, the top products are shown.

Using CALCULATETABLE()

CALCULATETABLE() is performs the same functionality as CALCULATE(). The only difference between these two is that CALCULATETABLE() returns a table. Whereas, CALCULATE() returns scaler values.



Continue using the previous file for the exercise or open the file DAX_Tools_And_Other_Resources_6.dax..

You will need to connect with DAX_Tools_And_Other_Resources_6.pbix

In this exercise, you will learn how to find the sales of all categories where color of the product is Black.

To use CALCULATETABLE() for sales of "Black colored" products:

1. Click on **External Tools** in the Ribbon in Power BI Desktop.
2. Click on **DAX Studio**.
3. Erase the existing code in Query Pane.
4. In the Query Pan, type the following code.

```
EVALUATE
CALCULATETABLE(
    'Product',
    PRODUCT[Color]="Black"
    )

```

5. Click on Run in the File Tab.

The Results are generated in the Results Pan.

ProductKey	Product	Standard Cost	Color	List Price	Model	Subcategory	Category	SKU
376	Road-250 Black, 48	1554.9479	Black	2443.35	Road-250	Road Bikes	Bikes	BK-R89B-48
378	Road-250 Black, 52	1554.9479	Black	2443.35	Road-250	Road Bikes	Bikes	BK-R89B-52
380	Road-250 Black, 58	1554.9479	Black	2443.35	Road-250	Road Bikes	Bikes	BK-R89B-58
374	Road-250 Black, 44	1554.9479	Black	2443.35	Road-250	Road Bikes	Bikes	BK-R89B-44
606	Road-750 Black, 52	343.6496	Black	539.99	Road-750	Road Bikes	Bikes	BK-R19B-52
604	Road-750 Black, 44	343.6496	Black	539.99	Road-750	Road Bikes	Bikes	BK-R19B-44
605	Road-750 Black, 48	343.6496	Black	539.99	Road-750	Road Bikes	Bikes	BK-R19B-48
584	Road-750 Black, 58	343.6496	Black	539.99	Road-750	Road Bikes	Bikes	BK-R19B-58
337	Road-650 Black, 62	486.7066	Black	782.99	Road-650	Road Bikes	Bikes	BK-R50B-62

5. Open **Power BI Desktop**.

6. Right click on **All Measures** in the Fields.

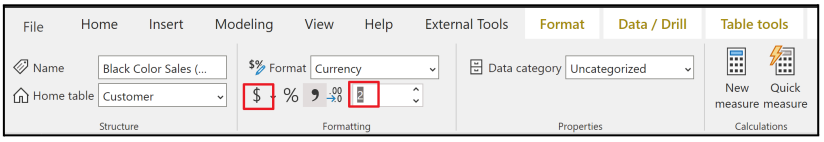
7. Click on **New measure**.

Measure tools open.

8. Type the following formula in the formula bar and press Enter.

```
Black Color Sales (CM) =
SUMX (
    CALCULATETABLE ( 'Product', PRODUCT[Color] = "Black" ),
    [Total Sales Amount (CM)]
)
```

9. Click on "$" and Type "2" in the text box in Formatting Section.



Click on the **matrix** visual in the visualization pane.

10. Drag and drop **Color** from the **Product** table to the rows.

11. Drag and drop **Total Sales Amount (CM)** and **Black Color Sales (CM)** from the **All Measures** table to **Values.**

Color	Total Sales Amount (CM)	Black Color Sales (CM)
Black	$38,236,128.00	$38,236,128.00
Blue	$9,602,851.12	$38,236,128.00
Grey		$38,236,128.00
Multi	$649,028.73	$38,236,128.00
NA	$1,099,301.70	$38,236,128.00
Red	$21,597,890.99	$38,236,128.00
Silver	$19,777,338.95	$38,236,128.00
Silver/Black	$147,482.77	$38,236,128.00
White	$29,745.21	$38,236,128.00
Yellow	$18,669,506.53	$38,236,128.00
Total	**$109,809,274.00**	**$38,236,128.00**

Notice that the total sales of Black color is being repeated in each row.

Final Quiz

Test Your Knowledge and Claim Your Certificate

After mastering the concepts in this book, it's time to demonstrate your understanding by taking the final quiz. Follow these steps to complete the quiz and earn your certificate of completion:

1. Navigate to: https://powerbitraining.com.au/dax-for-power-bi-professionals-from-basic-formulas-to-complex-analytics-quiz/

   Here you'll find a series of multiple-choice questions based on the material covered.

2. Take your time to go through each question, choosing the option that best answers the question. These questions test the comprehensive knowledge you've gained.
3. Continue answering all questions until you reach the end, marked by a flag icon.
4. Once you reach the flag, click on it to submit your quiz. This will automatically send your answers for evaluation.
5. If you pass, a certificate of completion will be emailed to you. Please check your email inbox (and the spam folder, just in case) to find and download your certificate.
6. Well done on completing the book and the quiz! Share your certificate of completion with others and take pride in your accomplishment.

By following these steps, you not only validate your proficiency but also enhance your credentials in the field of Data Analytics. Good luck!

Next Steps in Your Journey

Enhance your Power BI skills and engage with our community. Here are the actions you can take to continue your learning and connect with like-minded professionals:

- **Book a training session:** Deepen your understanding of Power BI with our specialized training sessions. Sessions for all levels are available through our main website. Book your session today at https://www.powerbitraining.com.au/
- **Set up a discovery session**: Customize your learning journey with a free discovery session, tailored to meet your specific goals. Schedule your discovery session at https://amzconsulting.com.au/
- **Enroll in our learning platform:** Access a wealth of knowledge through our online learning platform. Join now on https://lms.amzconsulting.com.au/classes
- **Follow our blogs**: Keep up with the latest in Power BI through our informative blogs. Explore our blogs on https://powerbitraining.com.au/blogs
- **Get Hands-On Resources:** Find essential learning materials, including cheat sheets and datasets, to practice and refine your skills. To Access resources navigate to https://powerbitraining.com.au/power-bi-resources
- **Connect on Socials**: Engage with us on social media for the latest updates, insights, and more:
    - LinkedIn: https://www.linkedin.com/company/amz-consulting-pty-ltd
    - Facebook: https://www.facebook.com/amzconsulting
    - Instagram: https://www.instagram.com/amzconsulting
- **Subscribe to Our Emails**: Receive the latest updates, promotions, and articles directly in your inbox.Subscribe at https://powerbitraining.com.au/subscribe

Leverage these resources and opportunities to advance your expertise in Power BI and build connections within the community.

Author Profiles

Ali Noorani:

In today's digital age, data is more than just a buzzword; it's a fundamental asset driving businesses forward. Ali Noorani understands the importance of data analytics and is the driving force behind AMZ Consulting. With an interesting journey from mechanical engineering to becoming a Microsoft Certified Trainer and an esteemed figure in the field of data analytics, Ali Noorani's expertise is one of the best in the field.

At AMZ Consulting, Ali brings over 13 years of experience in data analytics and training to the forefront. His tenure as the Business Intelligence Lead at Oil Search (now Santos), a prominent oil and gas company in Australia, has honed his skills and positioned him as a trailblazer in the industry. However, it's not just Ali's professional background that sets him apart; it's his passion for sharing knowledge and empowering others that truly defines his work.

Central to AMZ Consulting's offerings are Ali's meticulously crafted courses on Power BI, a leading business analytics tool. These courses, which cater to varying skill levels, are designed to equip professionals with the practical skills needed to thrive in today's data-driven landscape. Moreover, Ali's approach to teaching is lauded for its clarity and engagement, making complex concepts accessible to all.

Beyond training, AMZ Consulting offers consulting services tailored to the specific needs of clients, ranging from private corporations to government organizations. By prioritizing education and empowering clients to become self-sufficient, Ali ensures that the impact of his work extends far beyond the duration of a training session or consulting engagement.

Ali's influence extends beyond the confines of AMZ Consulting, as evidenced by his recognition as a Top Analytics Skills Voice on LinkedIn. His commitment to sharing insights, engaging with the community, and fostering a culture of continuous learning exemplifies his dedication to advancing the field of data analytics.

Given that data is the new currency, Ali Noorani and AMZ Consulting are focused on empowering individuals and organizations alike to harness the power of data for

success. Whether it's through expert training, insightful consulting, or thought leadership, Ali's impact is set to shape the future of data analytics.

To learn more about Ali Noorani and AMZ Consulting, visit the website or connect with them on LinkedIn, Instagram, and Facebook.

Muneeba Sirshar:

Muneeba Sirshar is a data science professional and a founding member of AMZ Consulting Pty Ltd, where she serves as the Operations Manager. Muneeba holds a Master's degree in Electrical Engineering, which provides her with a robust foundation for her technical and analytical endeavors.

With around seven years of experience in data science and data analytics, Muneeba has contributed significantly to both academic and professional spheres. Her research involving big data analytics using autoencoders was published in a Springer journal, part of the Research and Innovation Forum 2019.

Muneeba's continuous contributions at AMZ Consulting demonstrate her commitment to advancing the application of data analytics in business through education and operational leadership.

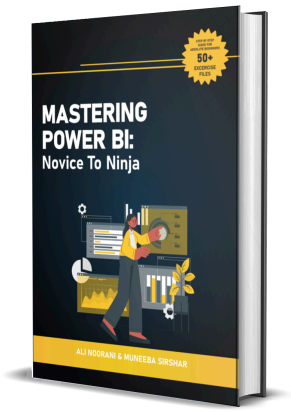
For more about her professional journey and courses, you can connect with Muneeba Sirshar on her LinkedIn profile.

Other Books by the Same Authors

Raise Your Analytics Game: From Excel to Power BI

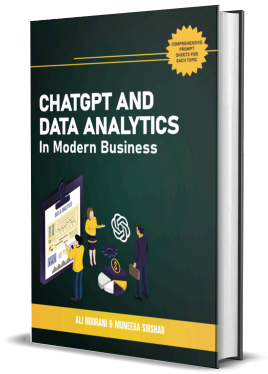
"Raise Your Analytics Game: From Excel to Power BI" provides an in-depth journey through the complete business intelligence cycle, with a strong focus on data transformation. This book is ideal for both novices and experienced professionals, guiding readers through advanced ETL (Extract, Transform, Load) processes, M language intricacies in Power Query, and best practices for shaping and cleansing data. You will learn to build dynamic, interactive dashboards and visualizations that enhance decision-making and data comprehension using DAX (Data Analysis Expressions) and advanced visualization techniques in Power BI. This guide equips you with the skills to refine your analytical capabilities, advance your career, and open new opportunities in data analytics.

Mastering Power BI: Novice to Ninja



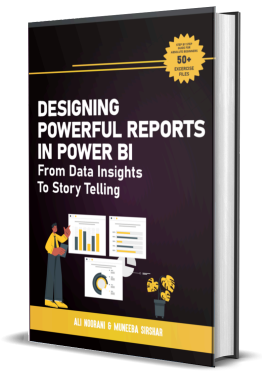
"Mastering Power BI: Novice to Ninja" is an essential resource focused on the critical aspect of data modeling within Power BI. This book covers key concepts such as building efficient data models, relationships, and hierarchies, and optimizing model performance. Readers will explore advanced DAX formulas for calculated columns and measures, enabling complex business logic and analytical solutions. Authored by Microsoft-certified trainers, the guide emphasizes practical, real-world applications, teaching you how to manage large datasets, implement Row-Level Security (RLS), and integrate with other data sources. Whether you're a beginner or looking to enhance your Power BI skills, this book provides a clear path to mastery in data modeling.

ChatGPT and Data Analytics in Modern Business



"ChatGPT and Data Analytics in Modern Business" explores the integration of artificial intelligence and business analytics. This comprehensive guide provides real-world case studies with tailored datasets, demonstrating how to use ChatGPT for advanced data analysis and automation. Readers will learn to build interactive AI-driven dashboards, develop conversational agents for data queries, and utilize exported sessions for hands-on AI analytics experience. The book also covers techniques for natural language processing, sentiment analysis, and predictive modeling, ensuring you acquire practical and applicable skills in modern data analytics. The custom chatbot feature offers personalized assistance, enhancing your ability to extract meaningful insights and drive data-driven decision-making.

Designing Powerful Reports in Power BI: From Data Insights to Storytelling



"Designing Powerful Reports in Power BI" is dedicated to the art and science of creating visually compelling and highly functional Power BI reports. This book introduces advanced techniques for data visualization, including the use of custom visuals, themes, and color schemes to enhance user engagement and comprehension. It covers storytelling with data, guiding you through the process of designing reports that highlight key insights and drive business decisions. You will learn to utilize Power BI's interactive features, such as bookmarks, tooltips, and drill-throughs, to create a seamless and intuitive user experience. This guide is essential for those aiming to achieve professional-level mastery in Power BI reporting.

Help & Support

We are here to help you on your journey in data analytics. If you encounter any issues or have questions, our dedicated support team is ready to assist you.

Contact Us:

- **Email:** For quick assistance, reach out to us at info@amzconsulting.com.au. We aim to respond to all inquiries within two business days.
- **Phone:** If you prefer to speak directly to our support team, call us at 1300 194 753. Our lines are open from 9 AM to 5 PM AEST, Monday through Friday.
- **Online Resources:** Visit our website at www.powerbitraining.com.au for additional resources and FAQs.
- **Support Forum:** For community support and to share your experiences, visit our support forum at https://amzconsulting.com.au/.

Your feedback is invaluable to us, and we encourage you to reach out with any suggestions or improvements for future editions of the book.