Prepared by Mohammad Umer

Power BI Tutorial

Content

1. Core Principles of Power BI Development ---------- 2

2. DAX ---------- 3

   2.1. Basic Functions ---------- 4
      2.1.1. CALENDAR() ---------- 4
      2.1.2. MONTH() ---------- 6
      2.1.3. WEEKDAY() & WEEKNUM() ---------- 7
      2.1.4. FORMAT() ---------- 8
      2.1.5. CONCATENATE() ---------- 9
      2.1.6. LEN() ---------- 10
      2.1.7. RIGHT(), LEFT(), MID() ---------- 10

   2.2. Aggregate Functions ---------- 11
      2.2.1. SUM() ---------- 13
      2.2.2. SUMX() ---------- 14

   2.3. DAX Filter Functions ---------- 17
      2.3.1. CALCULATE(), ALL(), ALLSELECTED(), ALLEXCEPT() ---------- 17

   2.4. Logical & Date Function ---------- 23
      2.4.1. AND() ---------- 25
      2.4.2. OR() ---------- 26
      2.4.3. NOT() ---------- 26
      2.4.4. IF() ---------- 27
      2.4.5. DATEDIFF() ---------- 27

2.5. Cumulative Functions ---------- 28
2.5.1. MTD ---------- 30
2.5.2. QTD ---------- 30
2.5.3. YTD ---------- 31

2.6. SAMEPERIODLASTYEAR() in Power BI ---------- 32

1. Core Principles of Power BI Development

There are two most important things we should keep in mind:

- When creating reports that provide accurate insights and meaningful information.
- When focusing on performance tuning and optimization.

The two important things are:

1. Data Preparation

This is where the **Power Query Editor** comes into the picture. Here, we can:

- Clean the data.
- Prepare and transform the data.
- Merge queries.
- Append queries.
- Remove null values.
- Replace values.
- Change data types.

By performing these operations, we obtain clean and well-structured data that can be effectively used for reporting purposes.

2. Data Modeling

After preparing the data, the next step is **Data Modeling**, where we organize the data in a meaningful way for analysis. In this stage, we:

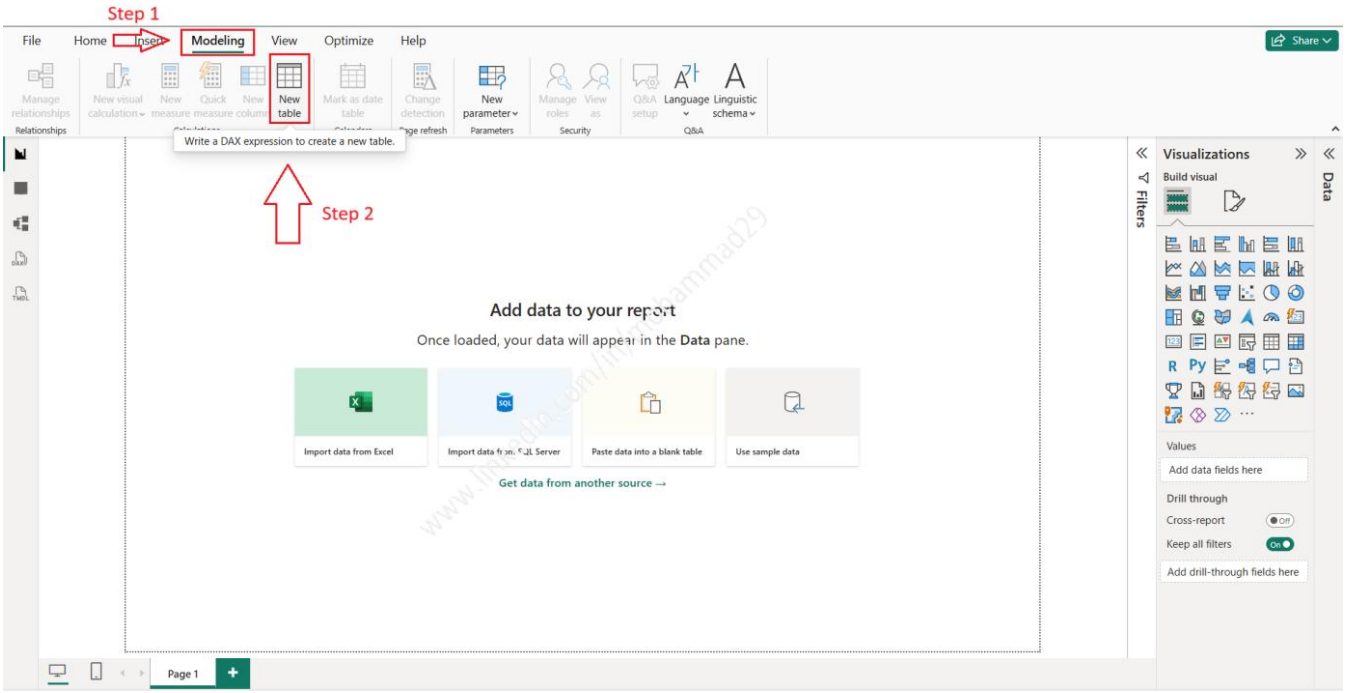
- Create relationships between different tables.
- Define primary and foreign keys.
- Apply normalization or star schema design.
- **Create calculated columns and measures using DAX**.
- Optimize the model for better performance and faster query execution.

A well-designed data model helps Power BI generate accurate visuals, improve performance, and make reports easier to maintain and understand.

- DAX

DAX (**Data Analysis Expressions**) is a formula language used in Power BI to create calculations and data within the model. It allows us to create **calculated columns, measures, and tables** directly in **Power BI Desktop**, without importing them from external sources.

We use DAX when we want to generate data inside Power BI itself, perform calculations, and derive new values from existing data for analysis and reporting purposes.

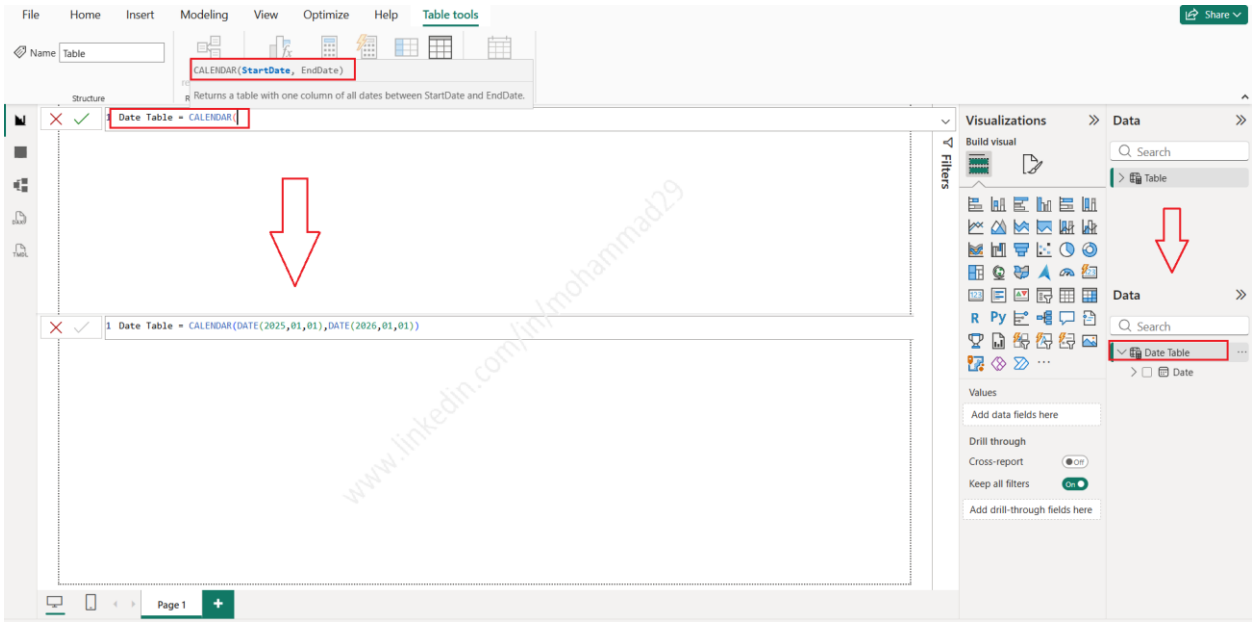


- Basic functions

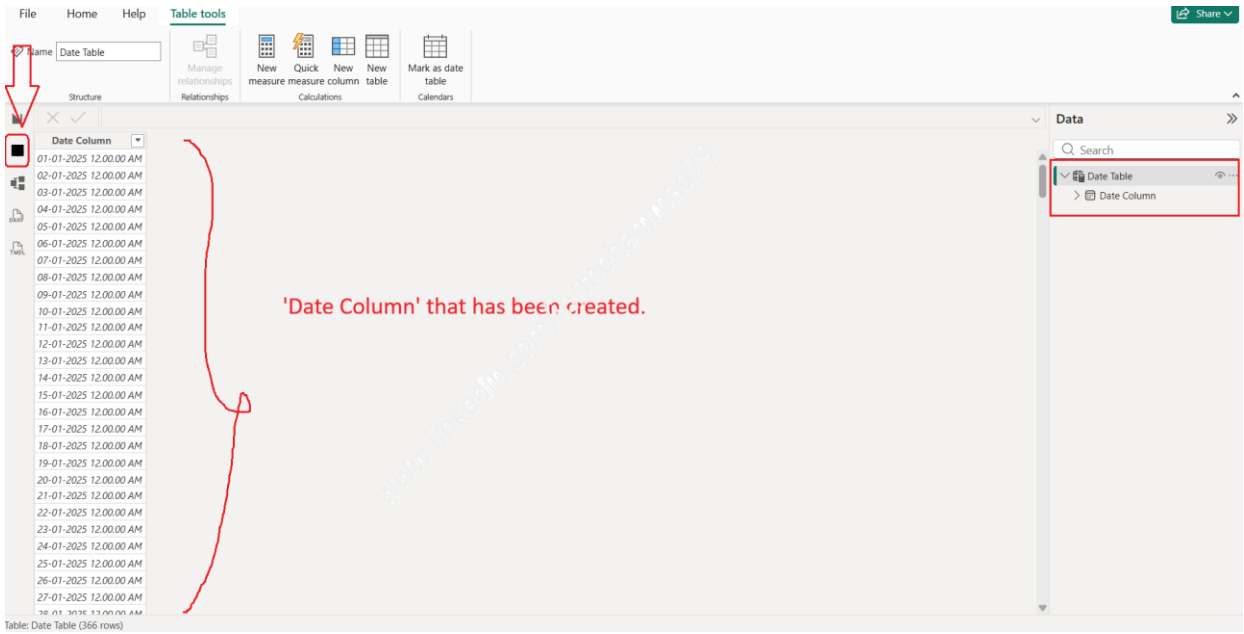
1. CALENDAR() function

Sometimes, it is required to create a **Date table** in Power BI. In this table, we create a **Date column**, which can be achieved using the DAX CALENDAR() function.

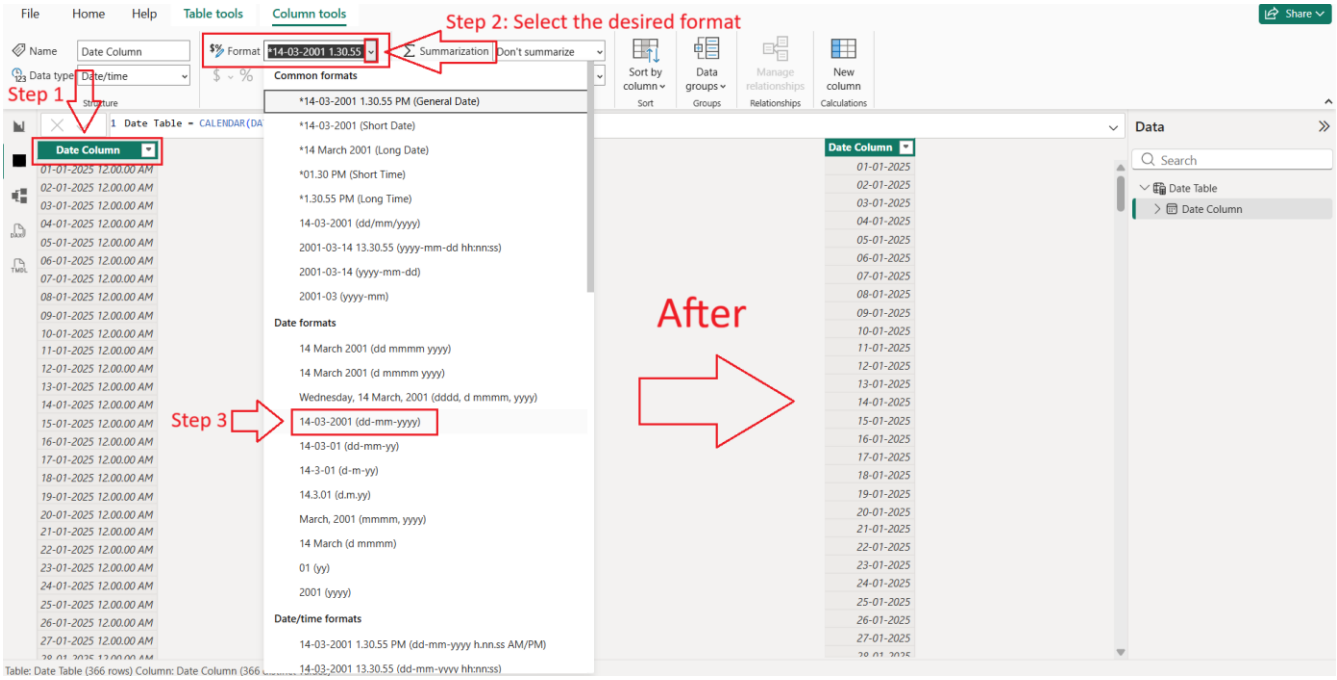
- **CALENDAR():**Returns a table with a single column containing all dates between a specified StartDate and EndDate.



We can also view the column that has been created.



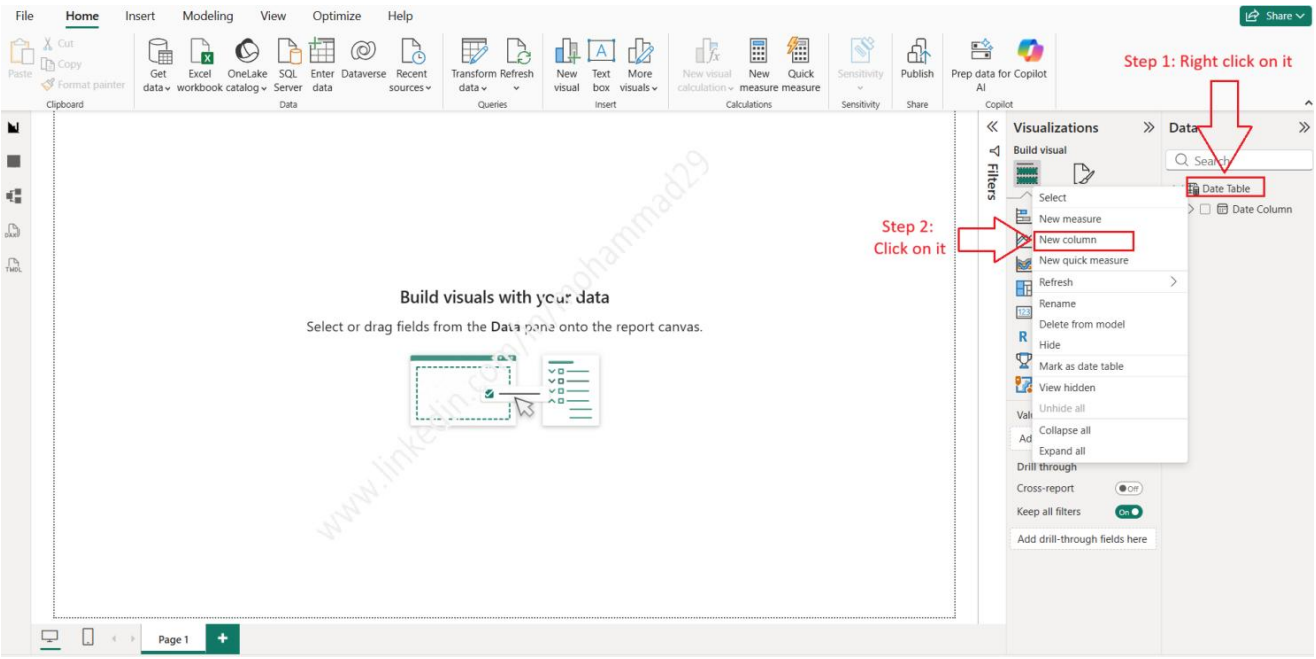
We can also change the format.

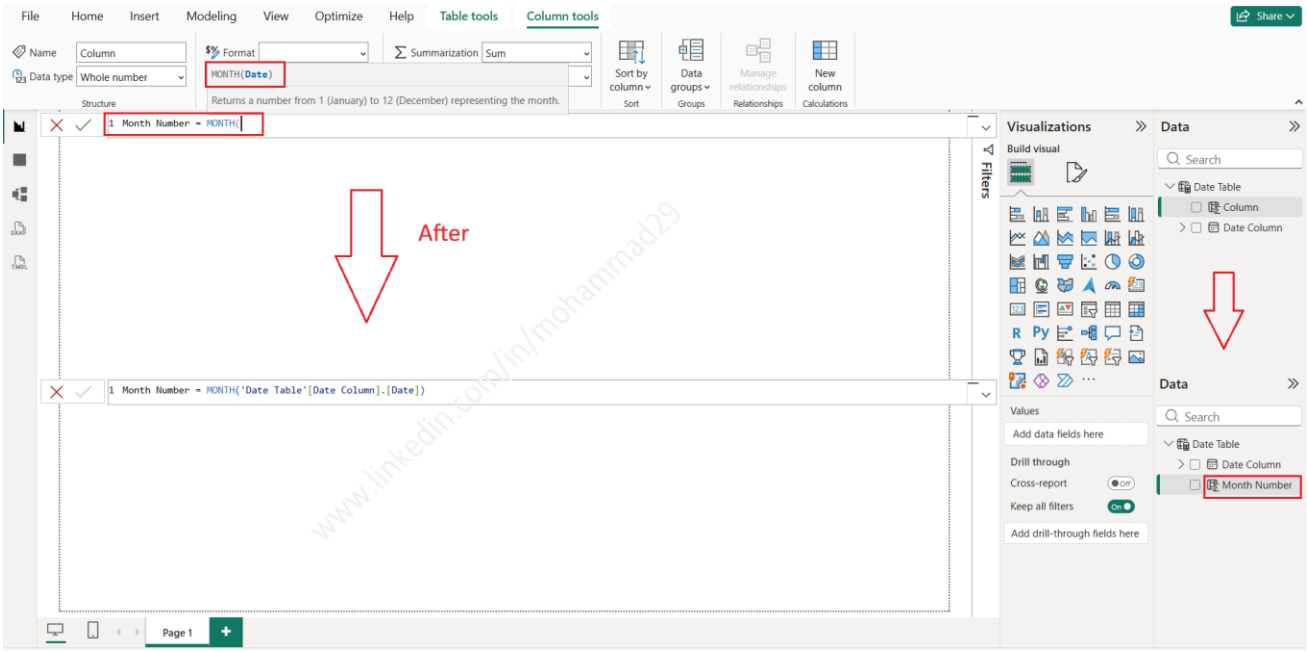


2. MONTH() function

Let's say we are interested in representing the **month number** for each date in a separate column in a table. We can use the **MONTH()** function to achieve this.

- **MONTH():** Returns a number from 1 (January) to 12 (December) representing the month of a given date.

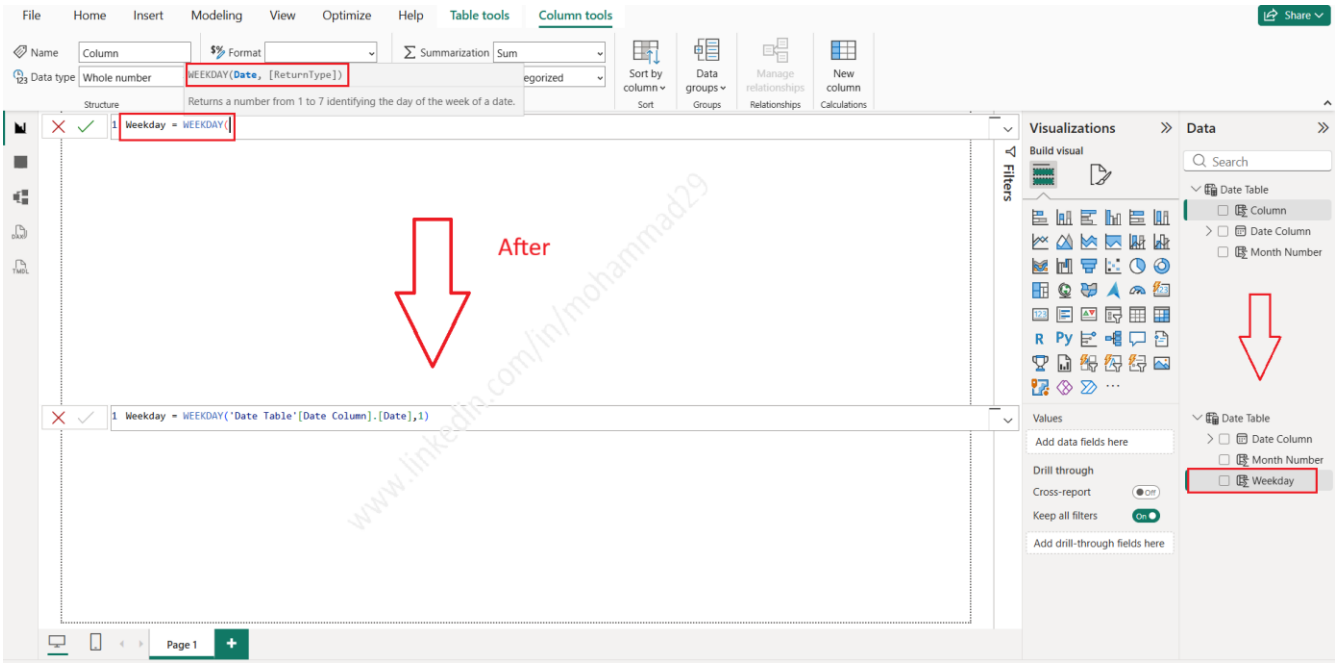




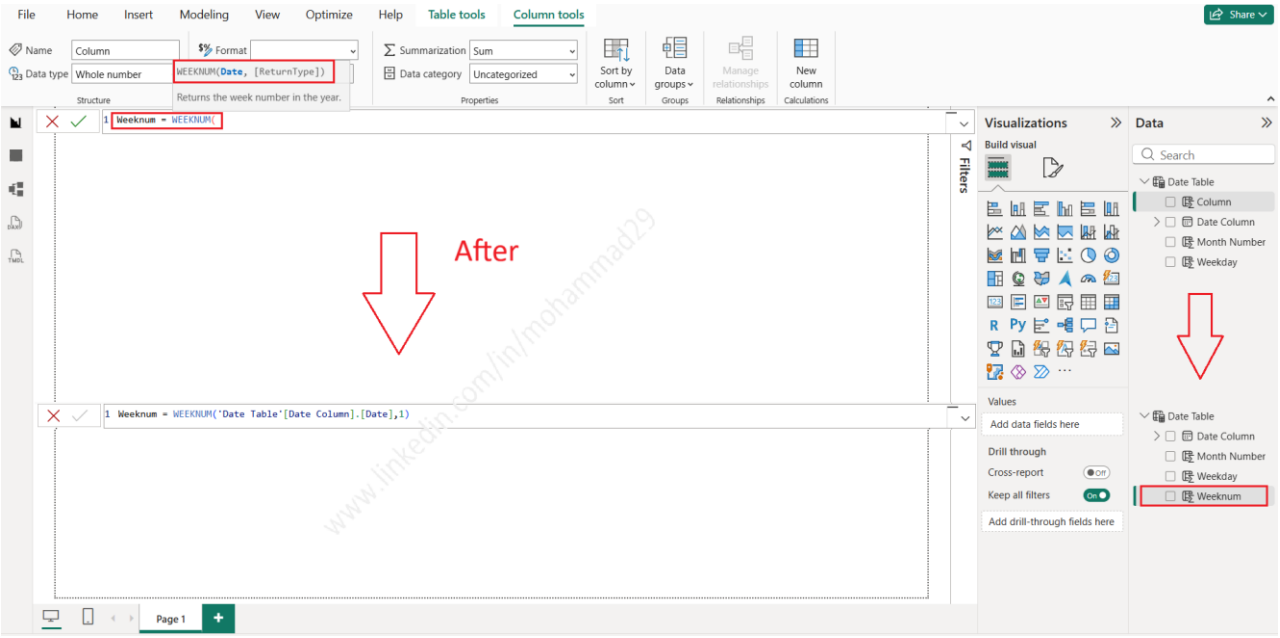
3. WEEKDAY() & WEEKNUM() functions

Now, let's say we are interested in knowing the **day of the week** and the **week number** for a given date. For this purpose, we can use the following DAX functions:

- **WEEKDAY():** Returns a number from **1 to 7** identifying the day of the week for a given date.



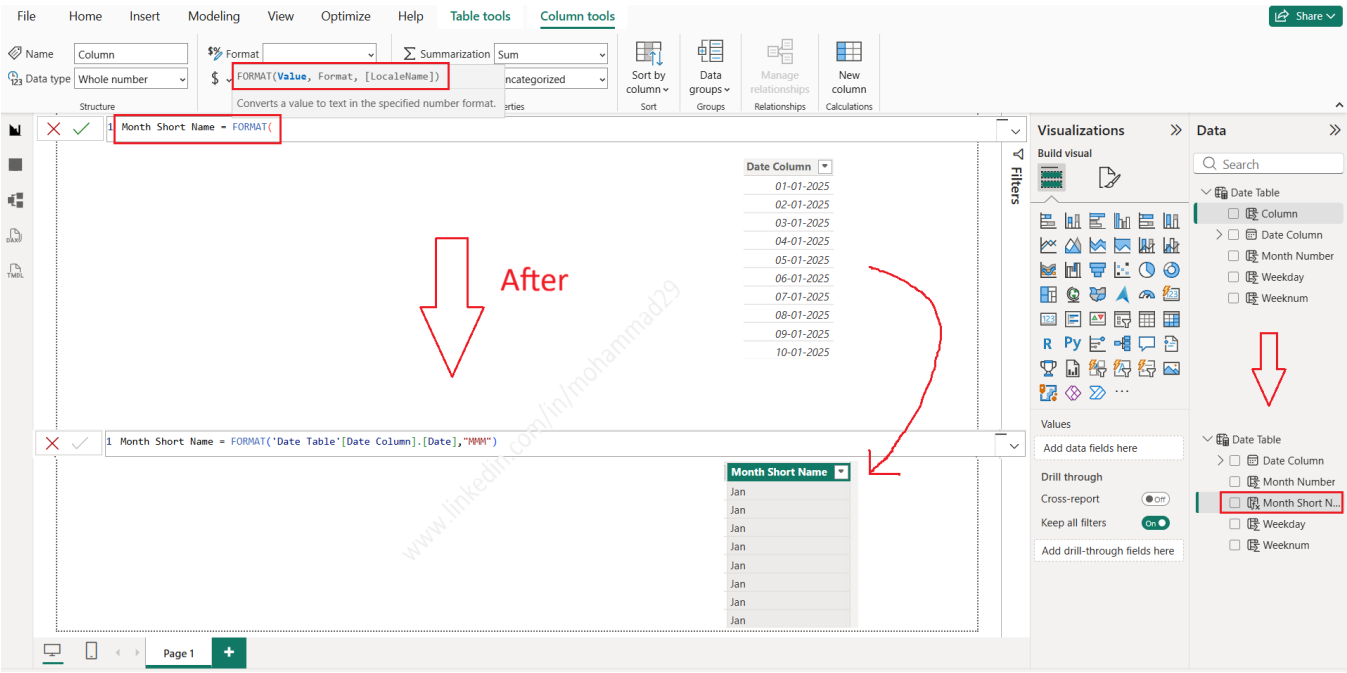
- **WEEKNUM():** Returns the **week number of the year** for a given date.

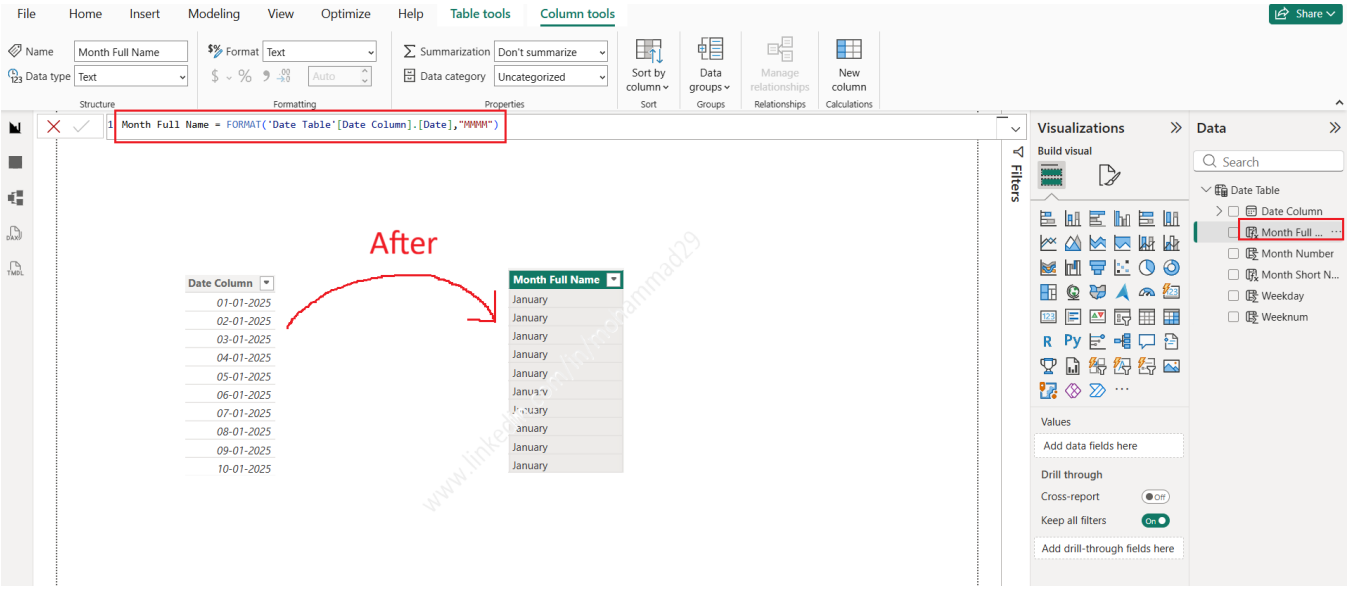


3. FORMAT() function

Now, let's say we are interested in representing months using **short names** and **full names**, such as a **"Month Short Name"** column and a **"Month Full Name"** column. For this purpose, we use:

- **FORMAT():** Converts a value to text using a specified format.

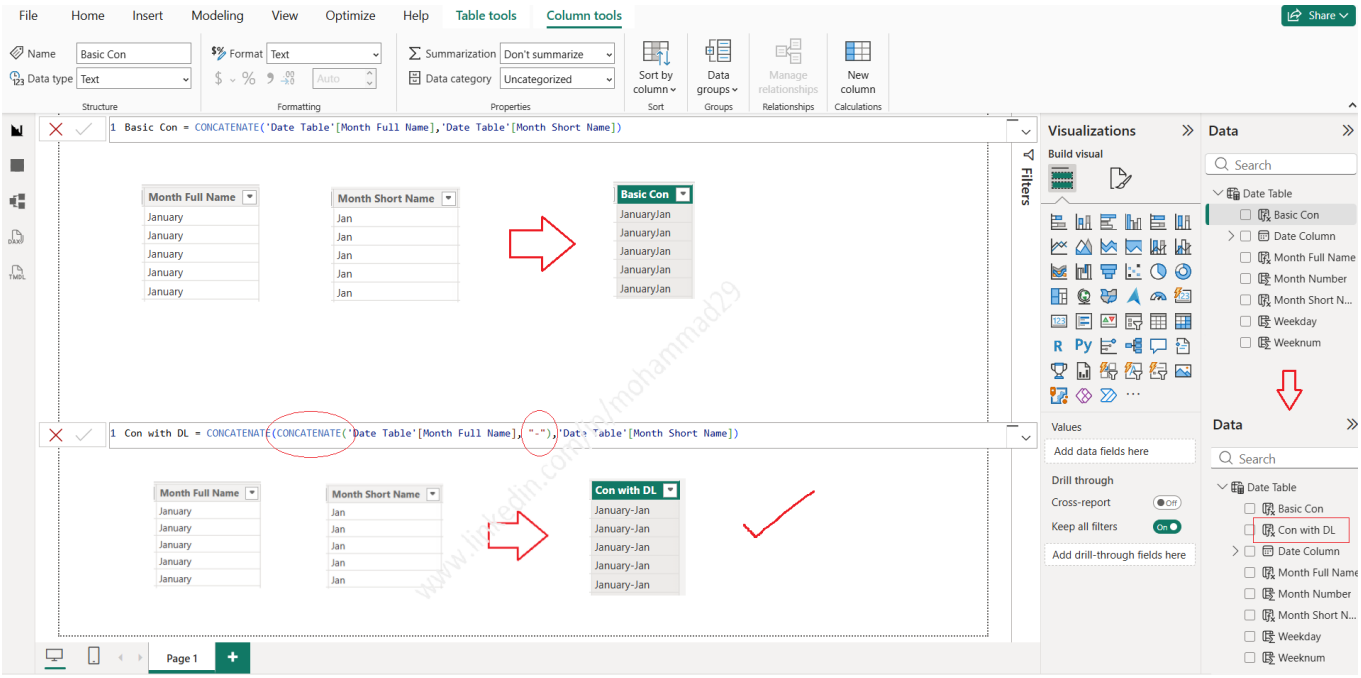




4. CONCATENATE() function

Sometimes, we may want to combine values from different columns into a single column. In such cases, we use the CONCATENATE() function.

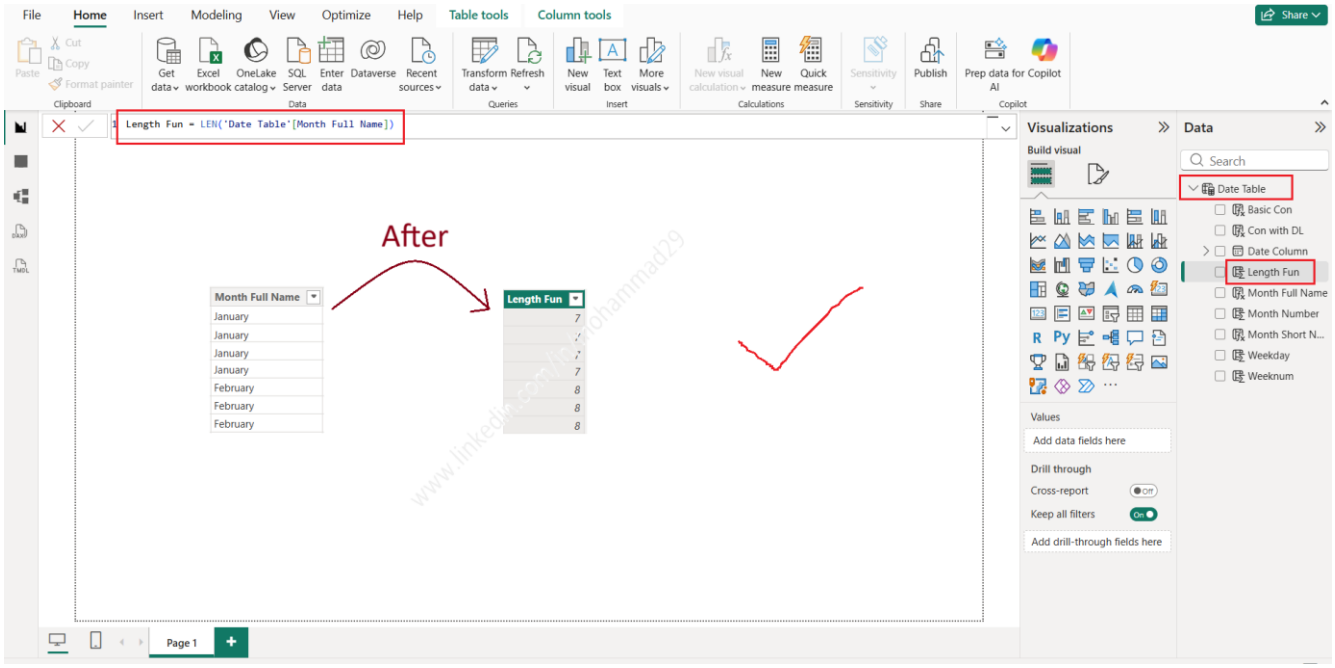
- **CONCATENATE()**: Joins two text strings into one text string.



5. LEN() function

let's say we want to extract or we want to know about the length of the characters of the values in the 'month full name' column. In such cases, we use the LEN() function.

- **LEN():** Returns the number of characters in a text string.



6. Other Functions

Apart from the above functions, DAX also provides several useful text functions, such as:

- **RIGHT():** Extracts a specified number of characters from the right side of a text string.
- **LEFT():** Extracts a specified number of characters from the left side of a text string.
- **MID():** Extracts a specific number of characters from the middle of a text string, starting at a given position.

- Aggregate functions

Aggregate functions are used to perform calculations across a set of values and return a single result. Common aggregate functions include:

- Works on Column

1. **SUM():** Returns the total of all values in a column.
2. **MIN():** Returns the smallest value in a column.
3. **MAX():** Returns the largest value in a column.
4. **AVERAGE():** Returns the average (mean) of a column.
5. **COUNT():** Returns the number of non-blank values in a column.
6. **PRODUCT:** Returns the product of all numbers in a column.

- Works on Table + Expression

Functions with X in DAX are **iterator functions** that evaluate an expression for each row in a table and then aggregate the results, whereas functions without X directly aggregate a single column.

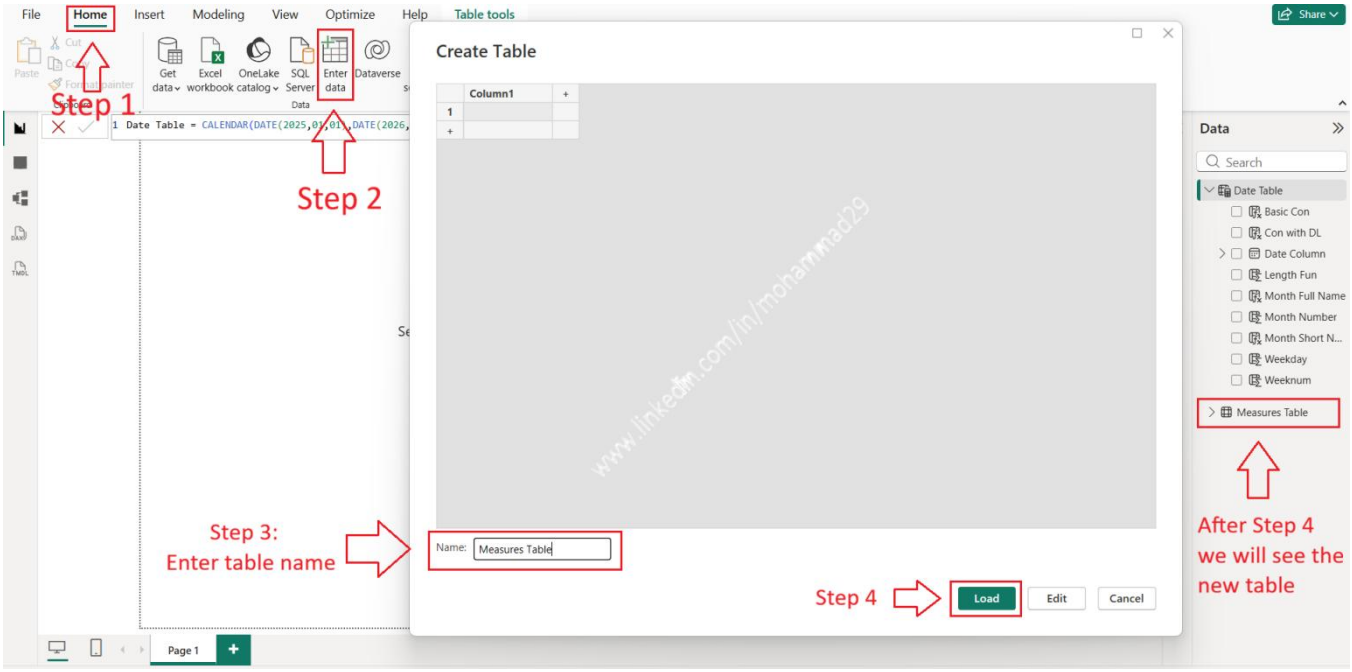
1. **SUMX():** Evaluates an expression for each row in a table and then sums the results.
2. **MINX():** Evaluates an expression for each row in a table and returns the smallest result.
3. **MAXX():** Evaluates an expression for each row in a table and returns the largest result.
4. **AVERAGEX():** Evaluates an expression for each row in a table and returns the average of the results.
5. **PRODUCTX():** Evaluates an expression for each row in a table & then returns the product of the results.

- Works on Table + Condition

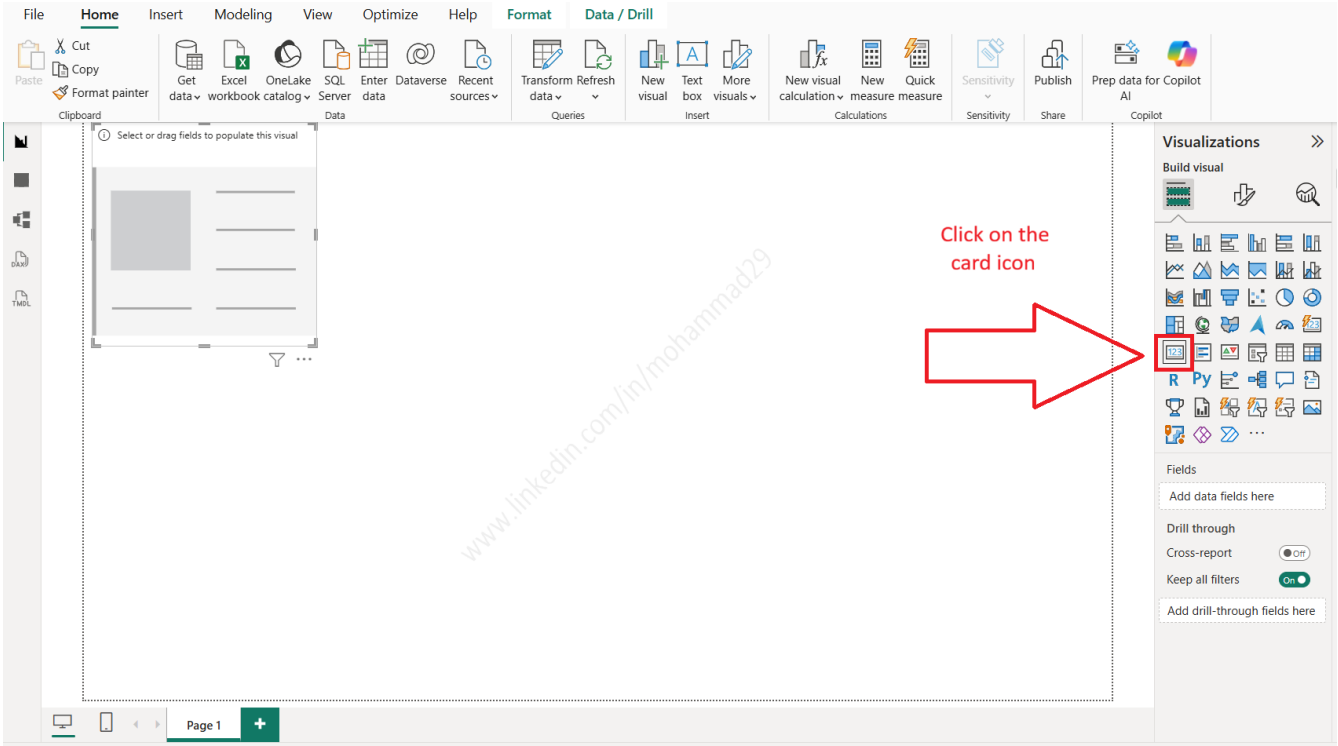
1. **FILTER():** Returns a table that has been filtered based on a given condition.

Function Type	Works On	How It Operates	Examples	Use Case
`SUM`, `MIN`, `MAX`, `AVERAGE`, `PRODUCT`, `COUNT`	Column	Directly aggregates values from a single column	`SUM(Sales[Amount])`	When the required data already exists in one column
`SUMX`, `MINX`, `MAXX`, `AVERAGEX`, `PRODUCTX`	Table + Expression	Iterates row by row, evaluates an expression, then aggregates results	`SUMX(Sales, Sales[Price]*Sales[Qty])`	When calculations must be performed per row before aggregation
`FILTER`	Table + Condition	Returns a filtered table based on a logical condition	`FILTER(Sales, Sales[Amount] > 1000)`	When you need to apply row-level conditions before using aggregations

Creating a new tabe i.e ‘Measures table’

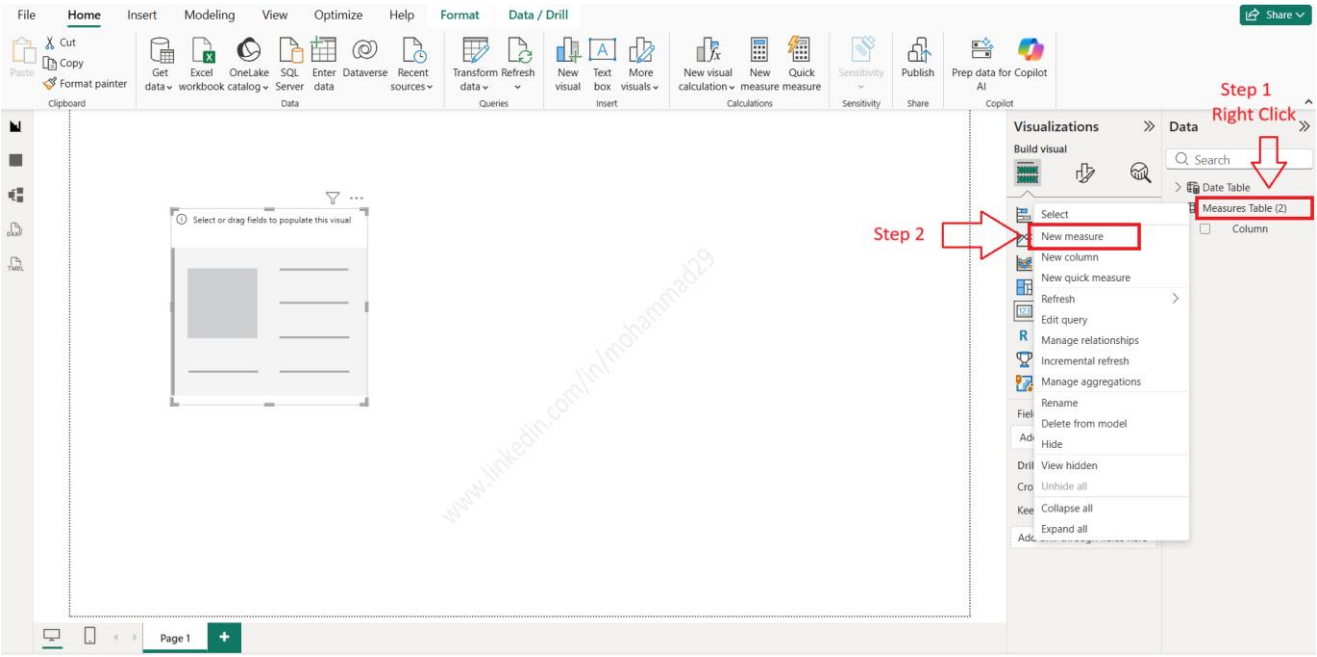


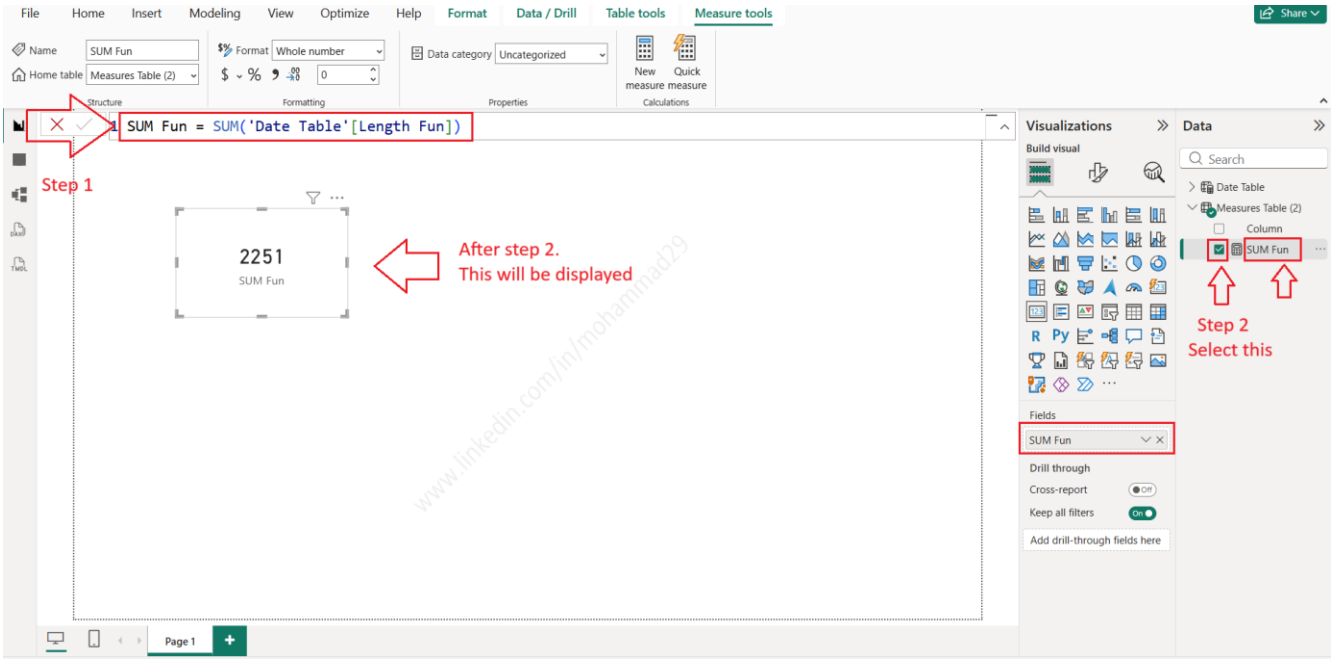
Select the card icon, as it will be used for all the Aggregate functions.



1. SUM(): Returns the total of all values in a column.

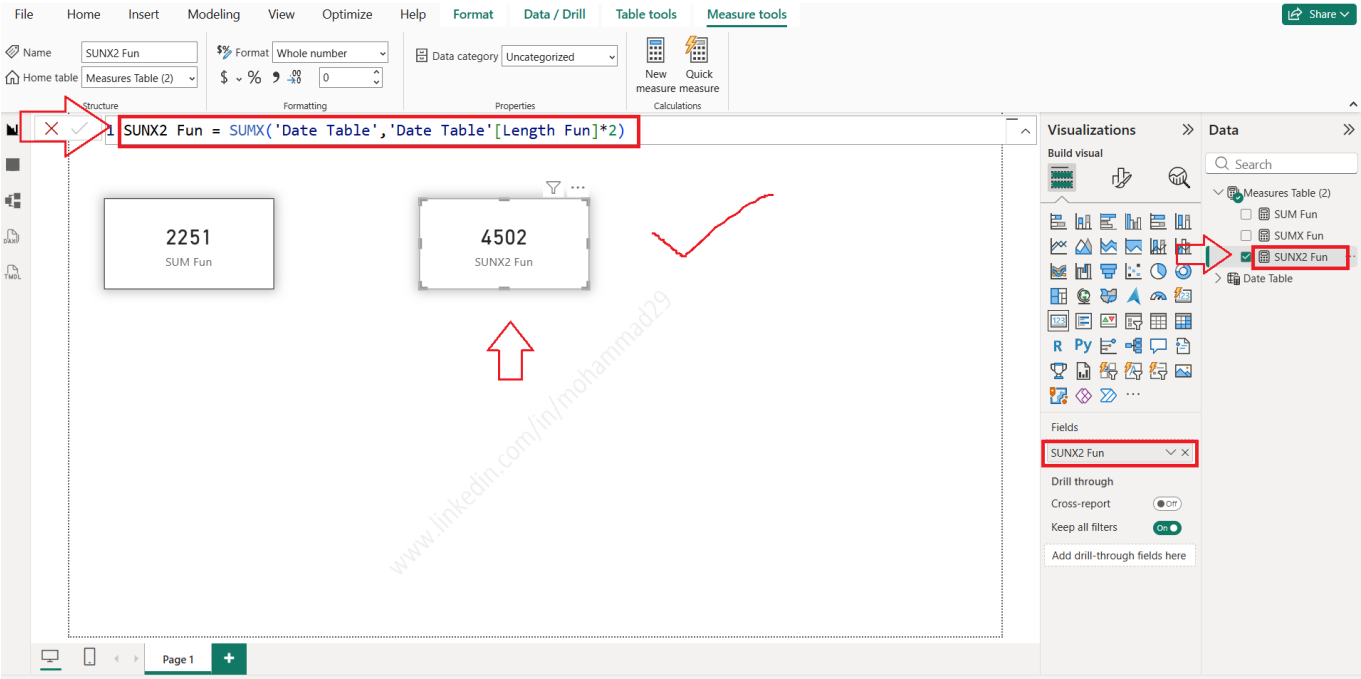
Q. let's say we want to find the sum of the values in the length function.



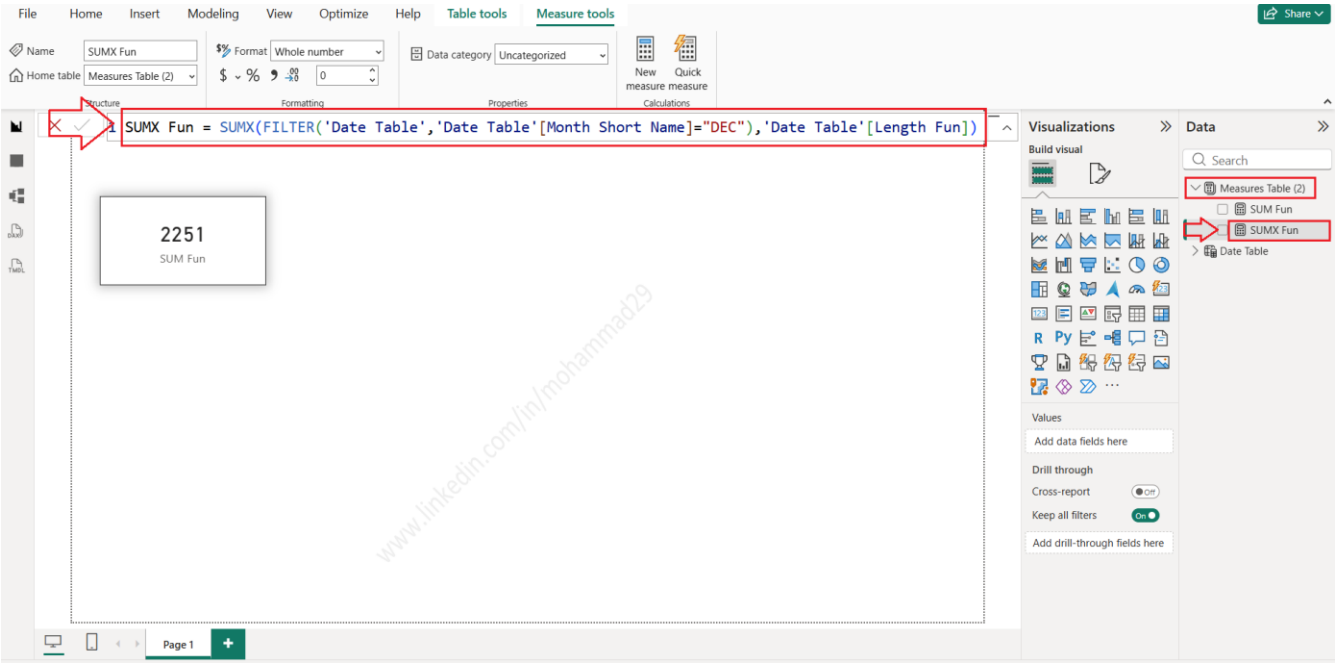


2. **SUMX():** Evaluates an expression for each row in a table and then sums the results.

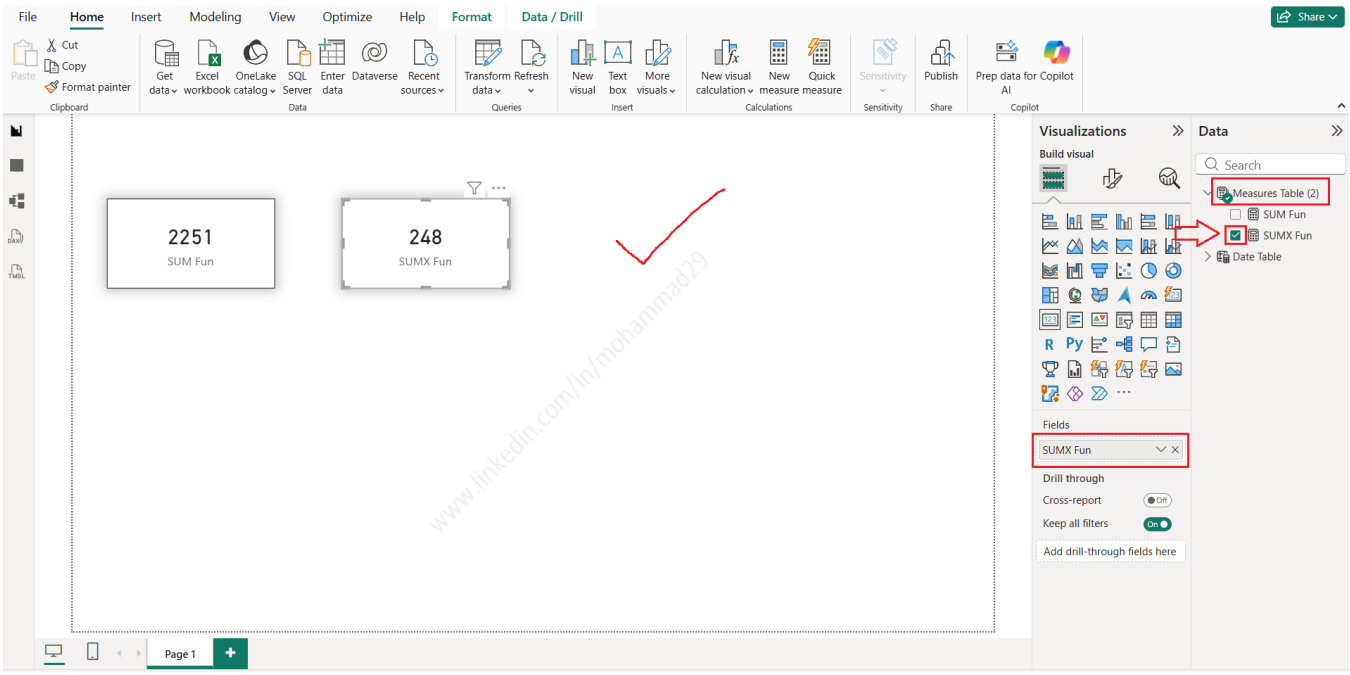
Q. let's say we want to find the sum of the values in the length function.



Q. let's say there's a condition that we want to find the sum of the values in the length function, but we want to do that only and only for December month.



Now lets visualize it using 'card visual'



✓ Note 1: Measures and Calculated Columns

Calculated columns occupy physical space in the data model, which increases the model size and can impact report performance.

Measures are dynamic calculations evaluated at runtime, so they use negligible storage space.

Therefore, whenever possible, prefer **measures over calculated columns** to keep the model efficient and improve report performance.

✓ Note 2: Two Main Contexts in DAX

DAX works using two main types of contexts:

1. Row context

Row context refers to the **current row** being evaluated in a table.

- It works row by row.
- It is mainly used in **calculated columns** and iterator functions (**SUMX**, **FILTER**, etc.).
- Each row is evaluated independently using column values from that row.

2. Filter Context

Filter context refers to the **set of filters** applied to the data model.

- It is created by **slicers**, **filters**, **CALCULATE**
- It affects which rows are included in a calculation.
- It is mainly used in **measures**.

- Important DAX Filter Functions

These functions are mainly used to **control & modify** the filter context in Power BI.

1. CALCULATE()

CALCULATE() is one of the most important functions in DAX.
It **modifies the filter context** for an expression and evaluates it under the specified filters.
It is used to **apply filters dynamically while calculating measures**.

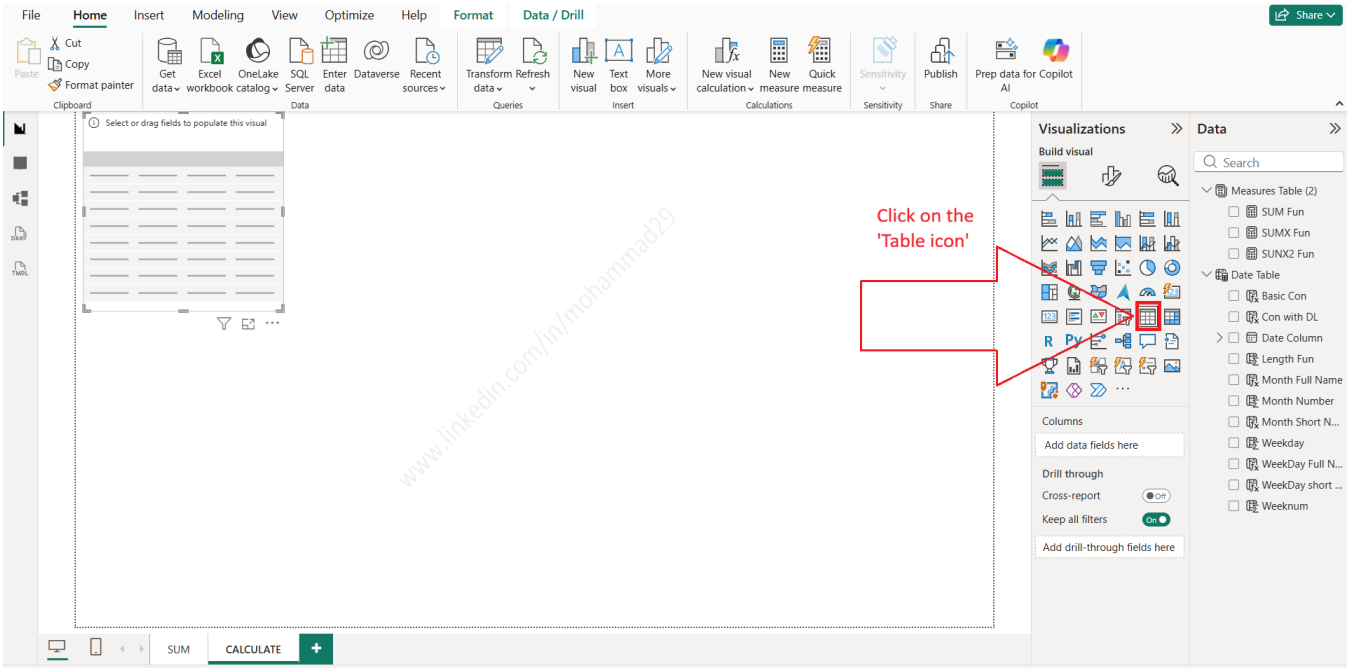
Syntax: CALCULATE(<expression>, <filter1>, <filter2>, ...)

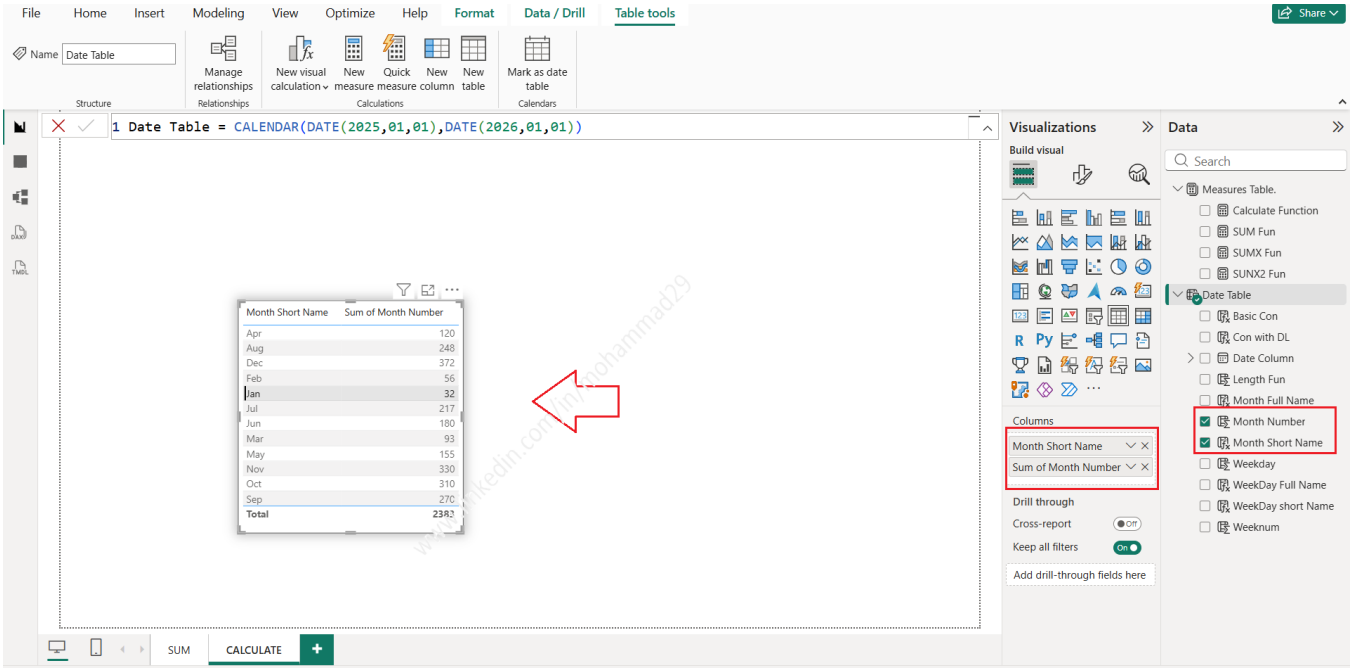
- Other functions used with CALCULATE()

Functions such as **ALL**, **ALLSELECTED**, and **ALLEXCEPT** are commonly used with CALCULATE() to control which filters are removed or preserved during calculation.

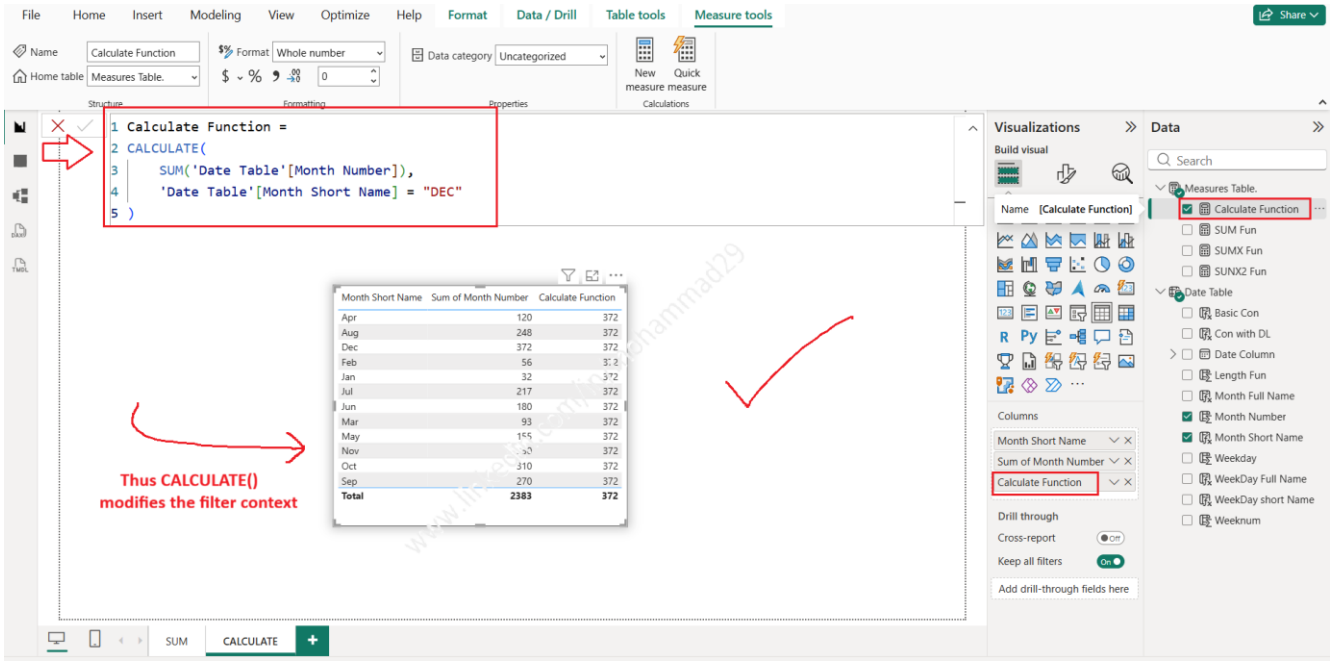
i. **ALL():** It removes all filters from a table or column and returns all rows, ignoring existing filters.
ii. **ALLSELECTED():** It removes filters except those applied by the user selection
iii. **ALLEXCEPT():** It removes all filters from a table except the specified columns.

- Practical for CALCULATE() in Power BI

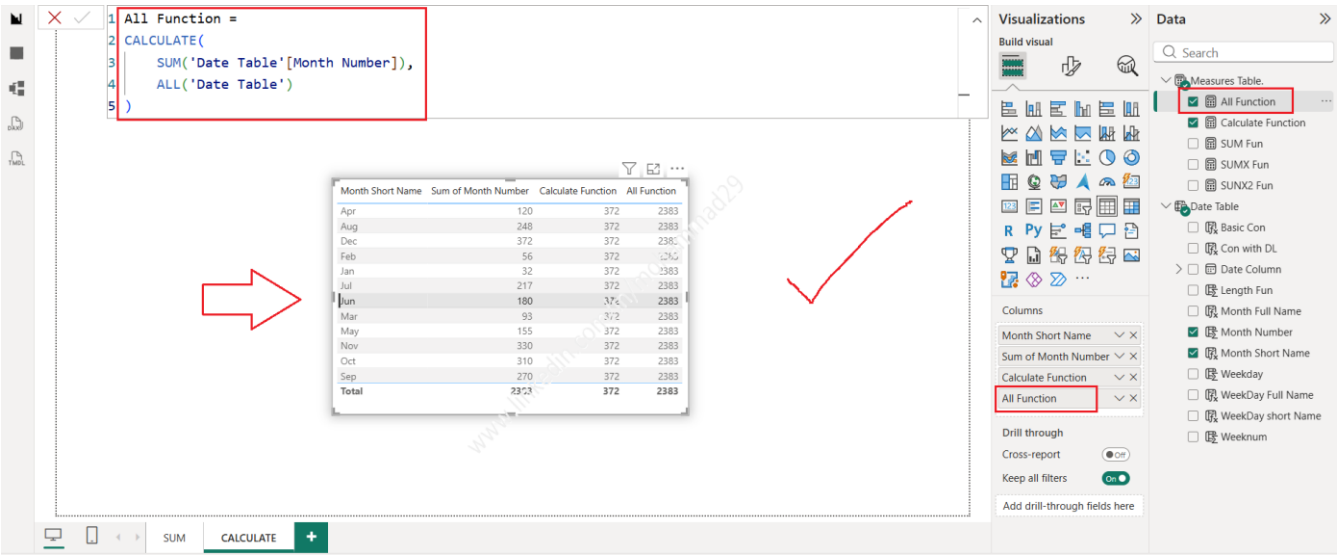




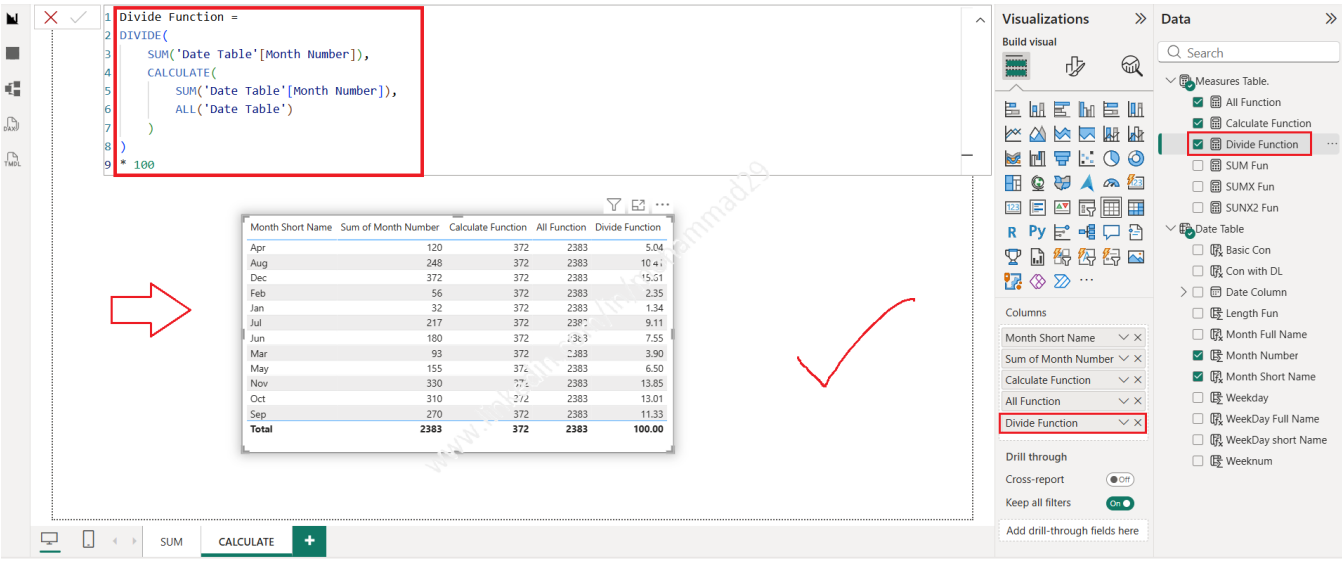
Q. Suppose we want to calculate the sum of values from the Month Number column for December only.



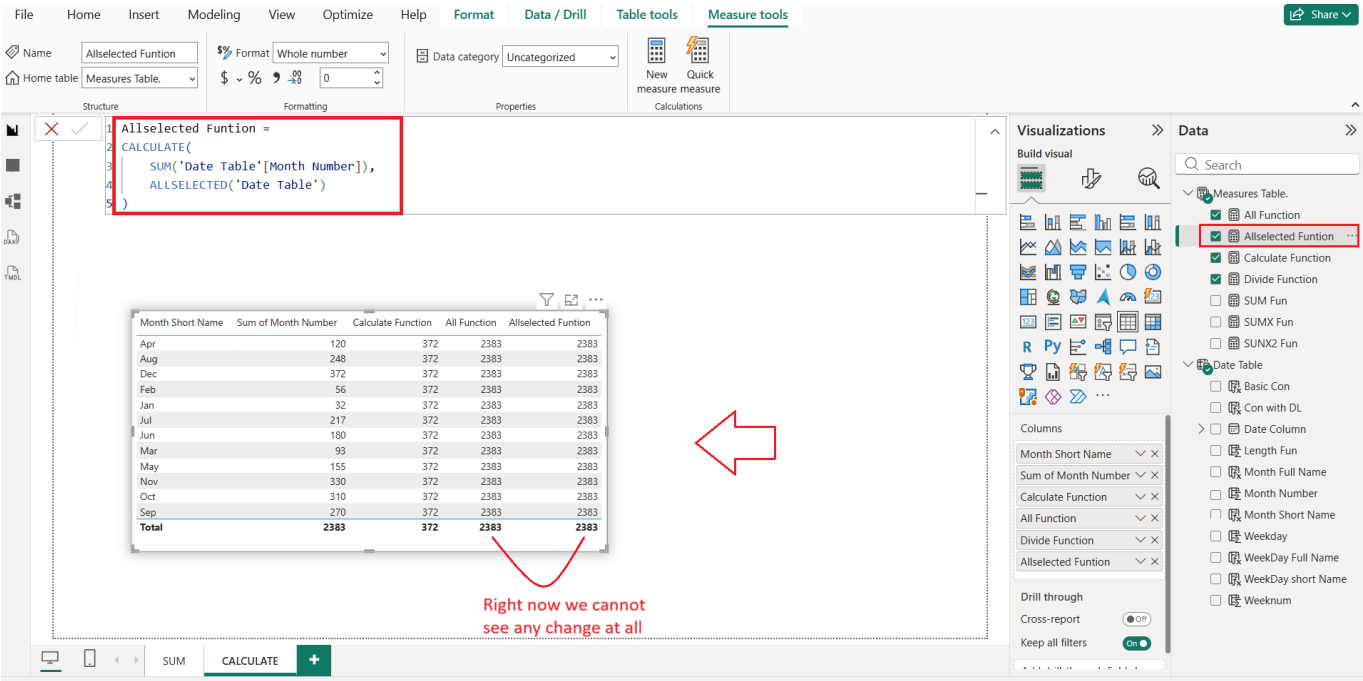
Q. Suppose we want to calculate the sum of all values from the Month Number column by removing all filters from the Date Table.



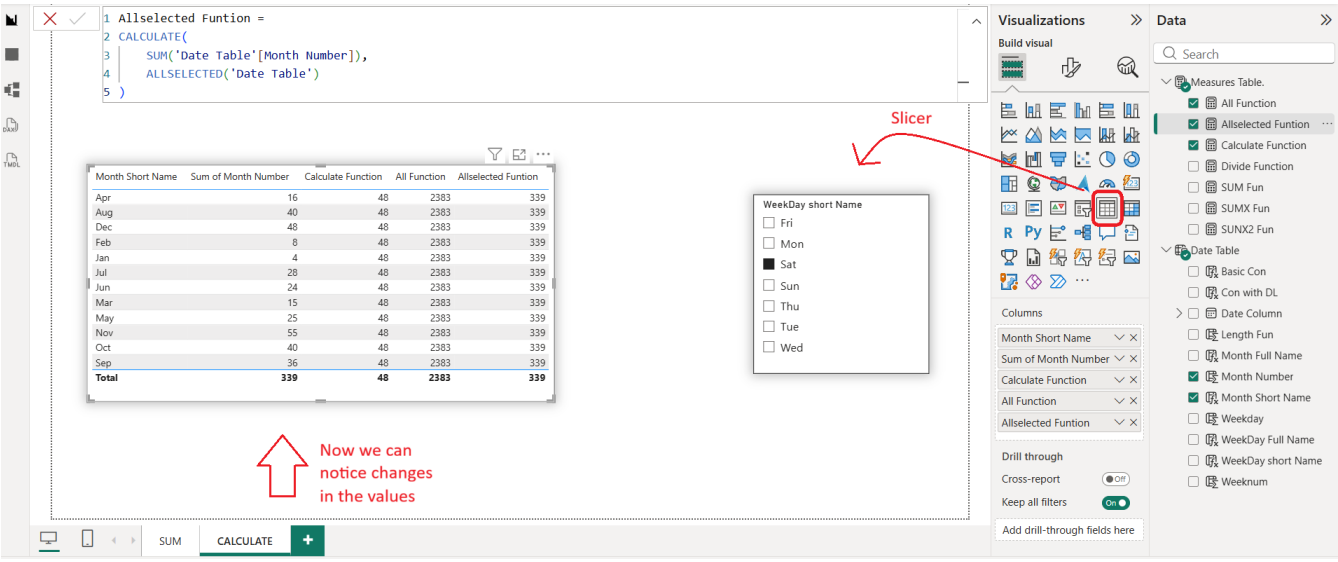
Q. Suppose we want to calculate the percentage contribution of the current filter context's Month Number sum w.r.t the total Month Number sum of the entire Date Table (ignoring all filters).



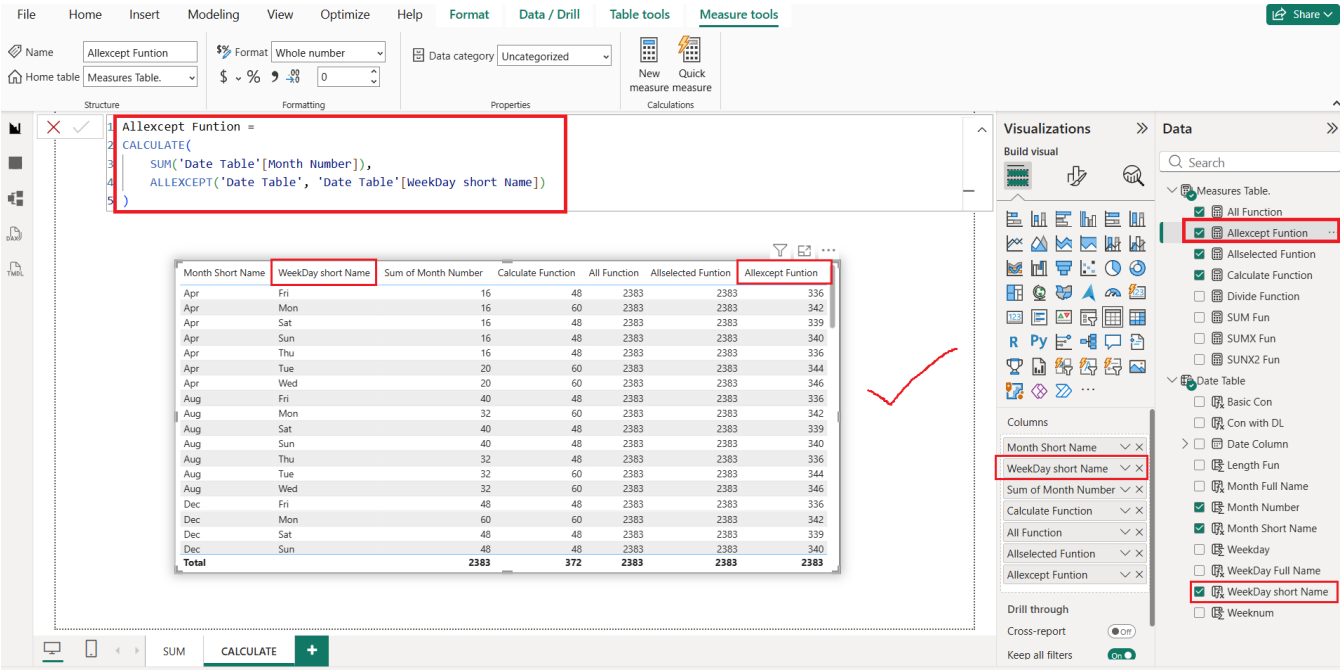
Q. Suppose we want to calculate the sum of the Month Number column while respecting only the filters and selections made by the user (using slicers or visuals) on the Date Table.



We will add 'slicer'



Q. Suppose we want to calculate the sum of the Month Number column while removing all filters from the Date Table except the filter on the WeekDay Short Name column.



- Logical & Date functions

Logical and date functions are used to evaluate conditions, control decision-making, and perform operations on date values. Common logical and date functions include:

- Works on Conditions & Dates

1. **AND()**: Checks whether all arguments are TRUE and returns TRUE only if all conditions are satisfied.

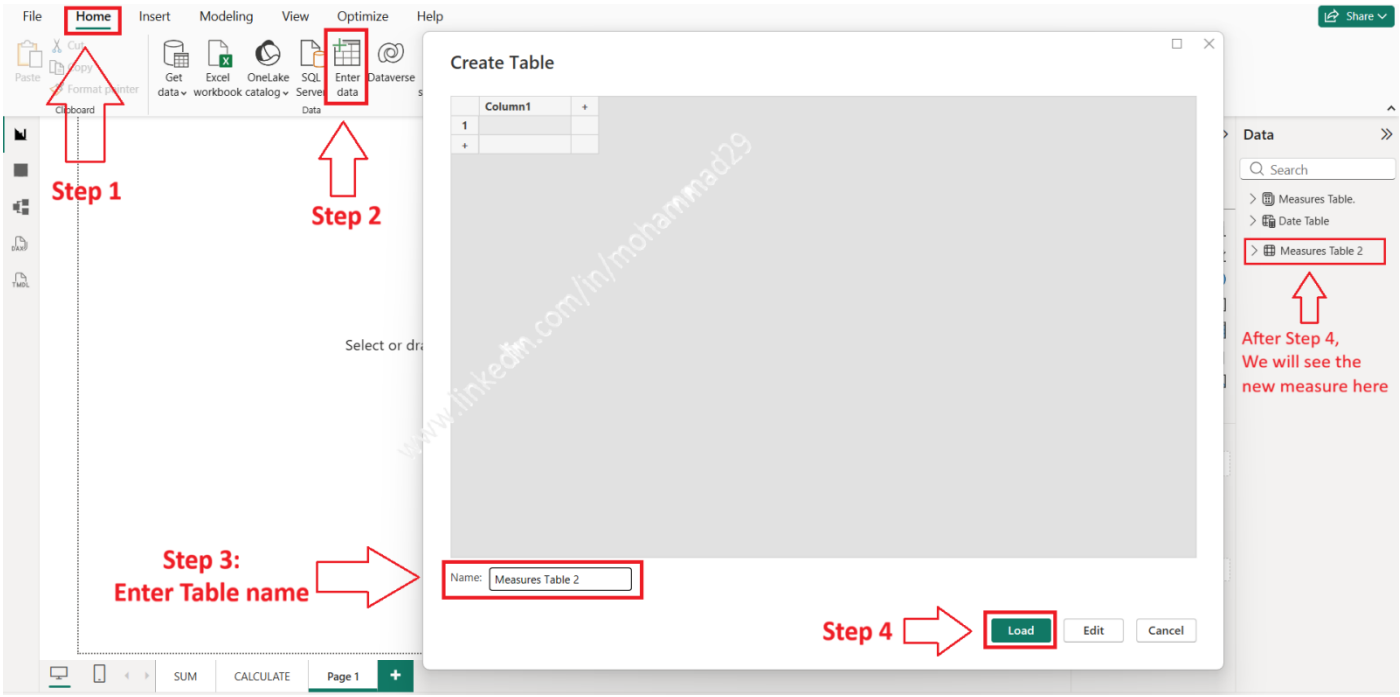
2. **OR()**: Returns TRUE if any of the arguments are TRUE, and returns FALSE only if all arguments are FALSE.

3. **NOT()**: Returns the opposite (negated) result of a given condition.

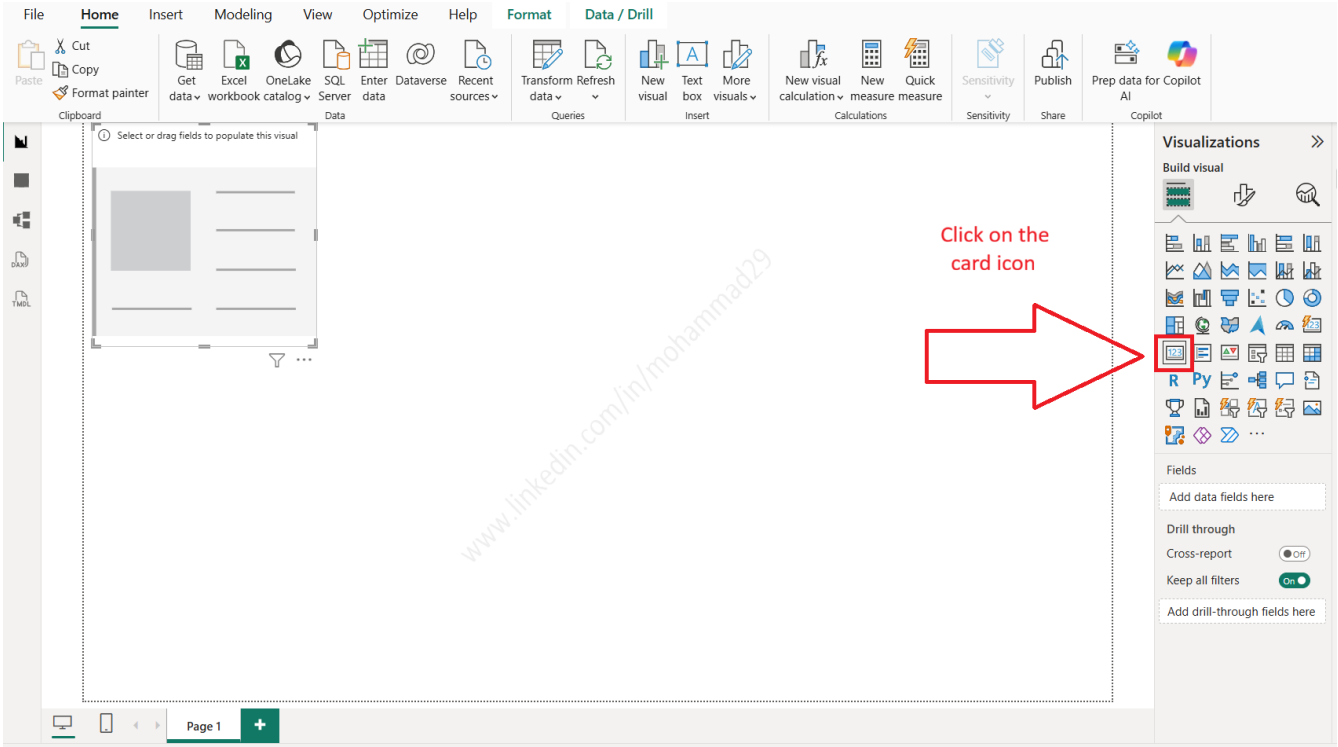
4. **IF()**: Evaluates a condition and returns one value if the condition is TRUE, and another value if the condition is FALSE.

5. **DATEDIFF()**: Returns the number of specified time units (interval) between two input dates.

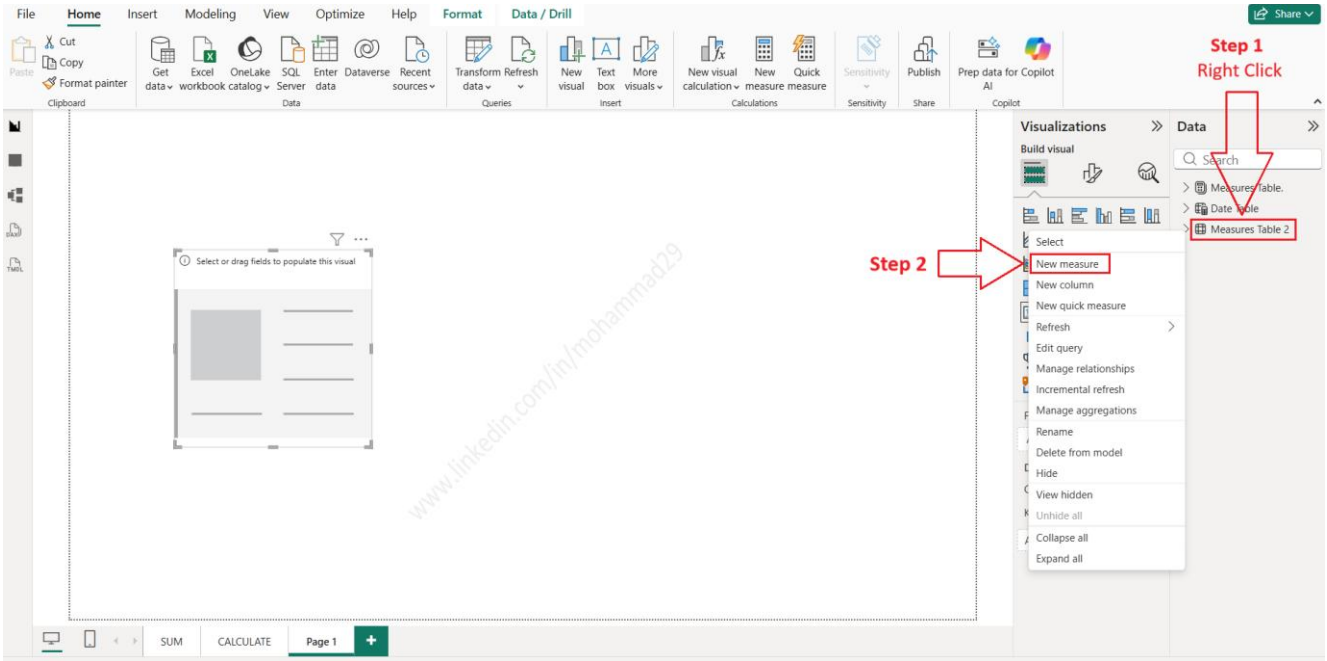
Creating a new tabe i.e ‘Measures Table 2’

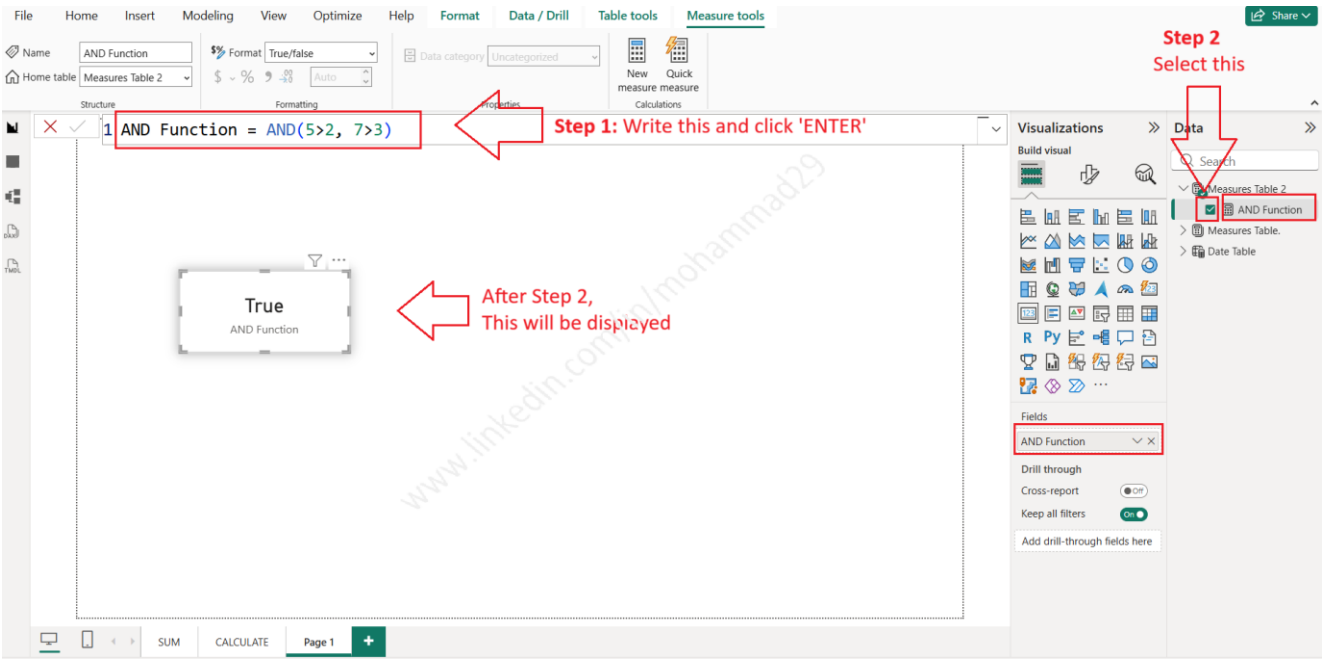


Select the card icon, as it will be used for all the Aggregate functions.

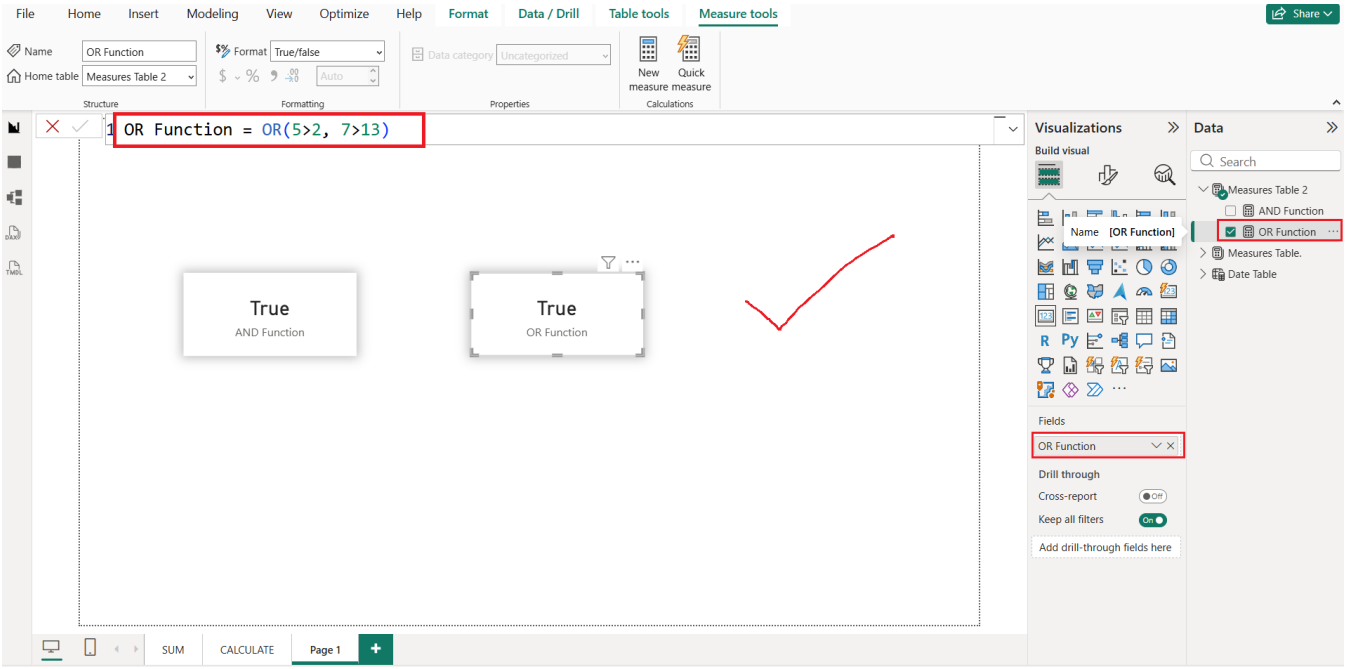


1. AND(): Checks whether all arguments are TRUE, and returns TRUE if all arguments are TRUE.

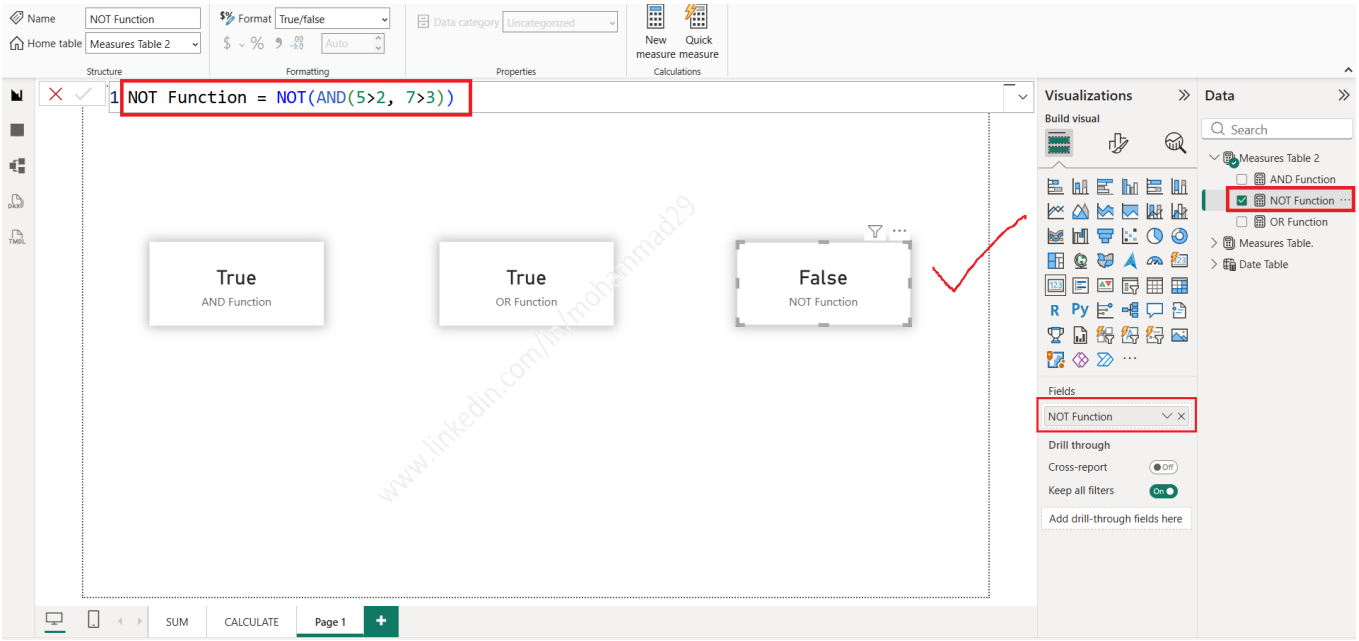




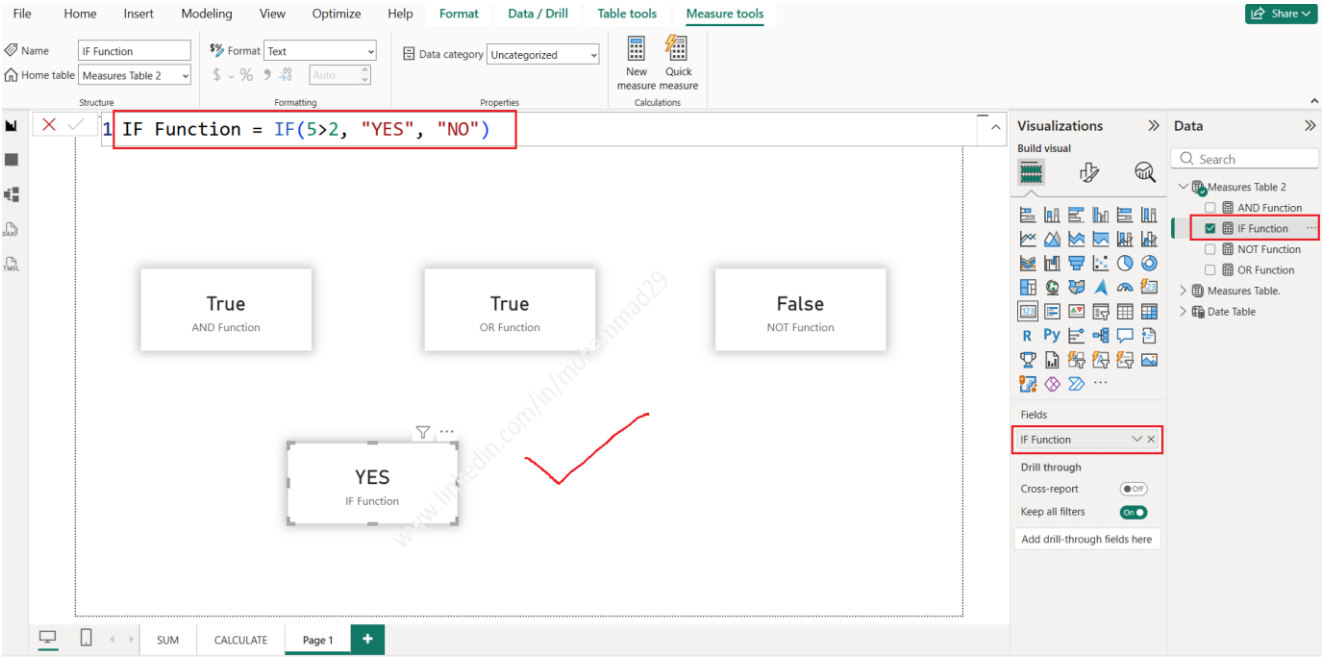
2. OR(): Returns TRUE if any of the arguments are TRUE, and returns FALSE if all arguments are FALSE.



3. NOT(): Returns the opposite (negated) result of a given condition.

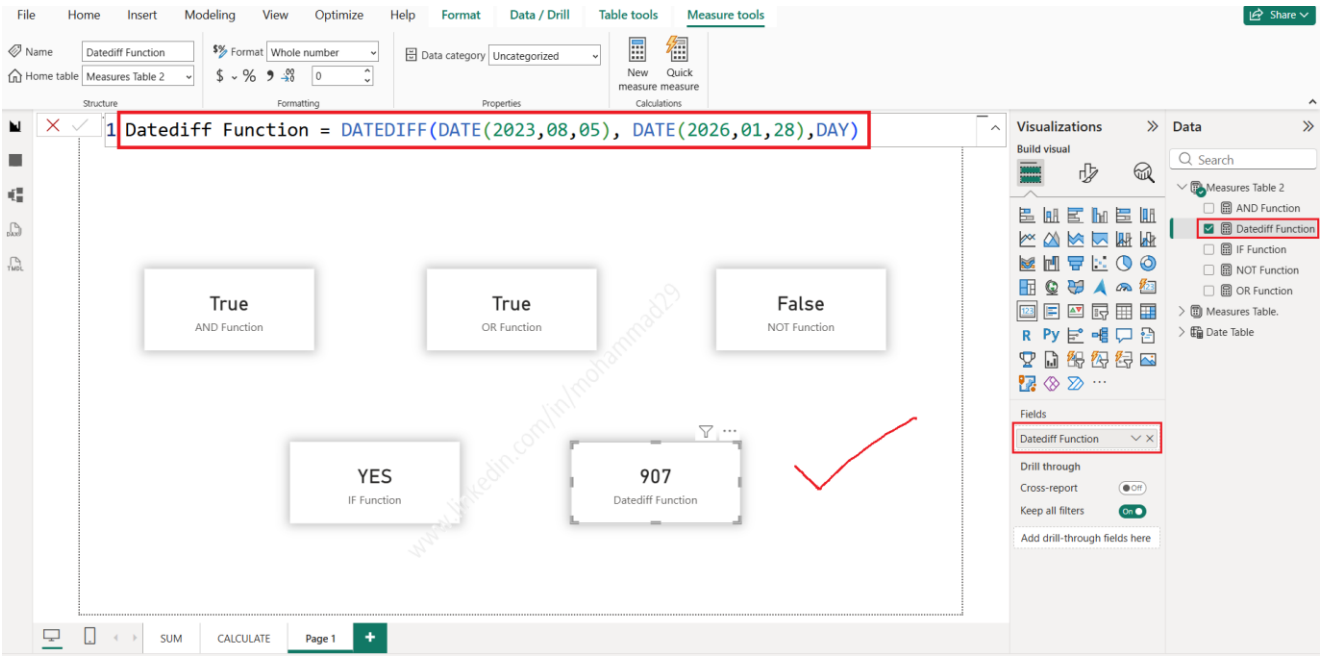


4. IF(): Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE.



5. DATEDIFF(): Returns the number of units (unit specified in Interval) between the input two dates.

There in ter interval we can put: YEAR, DAY, MONTH.



- Cumulative Functions – MTD, QTD & YTD

Cumulative functions are used to calculate **running totals over a period of time**. They summarize data progressively from the beginning of a specific time period up to the current date or record. These functions are very useful in **business intelligence, reporting, and analytics** for tracking performance over time.

They help answer questions like:

- How much sales have we made **this month so far**?
- What is the total revenue **this quarter to date**?
- What is the performance **from the start of the year till today**?

- Types of Cumulative Functions

1. **MTD (Month-To-Date):** Calculates the cumulative total from the first day of the current month up to the current date.

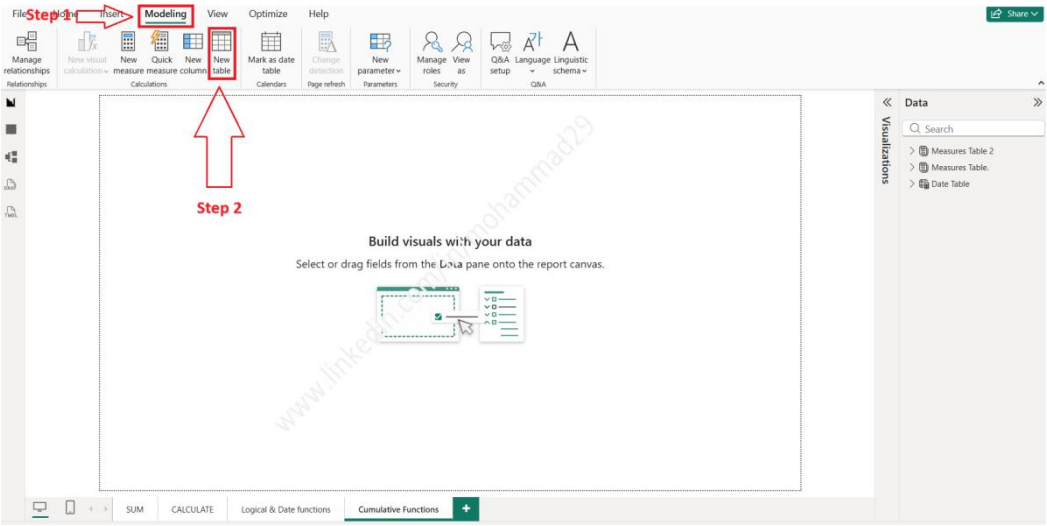
2. **QTD (Quarter-To-Date):** Calculates the cumulative total from the beginning of the current quarter to the current date.

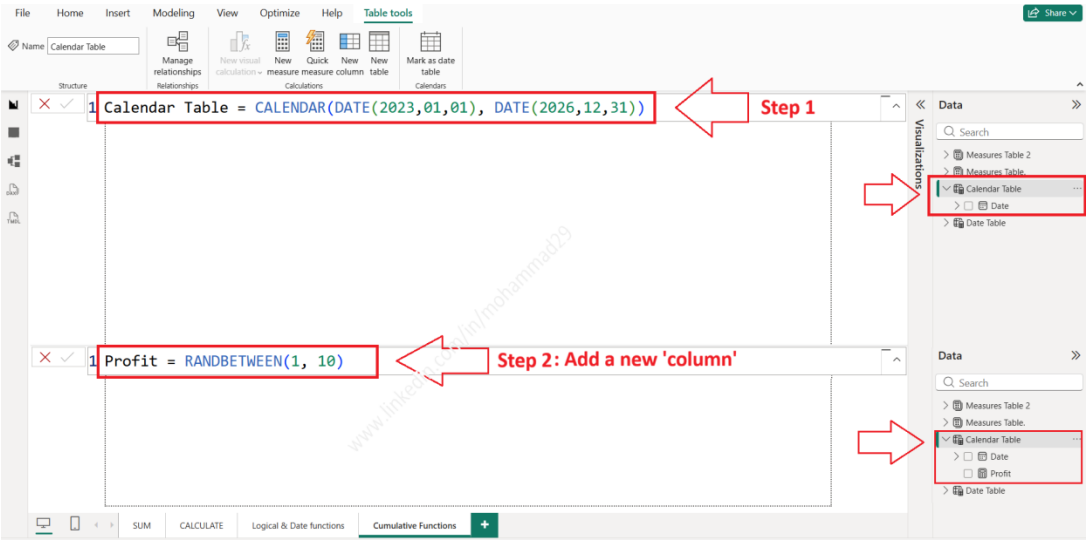
3. **YTD (Year-To-Date):** Calculates the cumulative total from the first day of the year up to the current date.

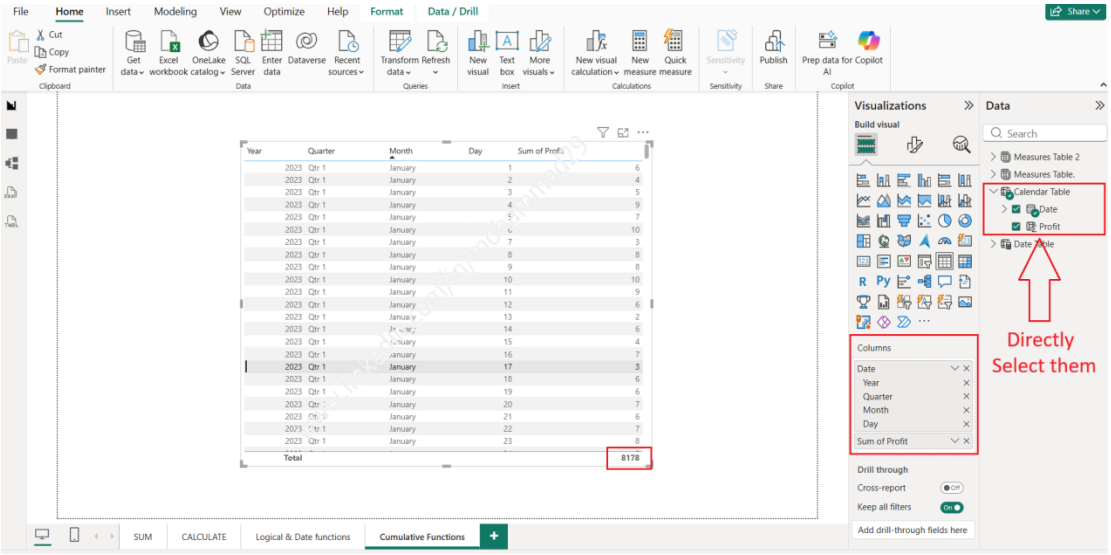
- Benefits of Cumulative Functions

- Helps in **trend analysis**.
- Supports **real-time performance tracking**.
- Useful for **business decisions**.
- Makes reporting more **dynamic and meaningful**.
- Reduces manual calculation effort.

- For Practicals lets create a new table i.e,. Calender Table

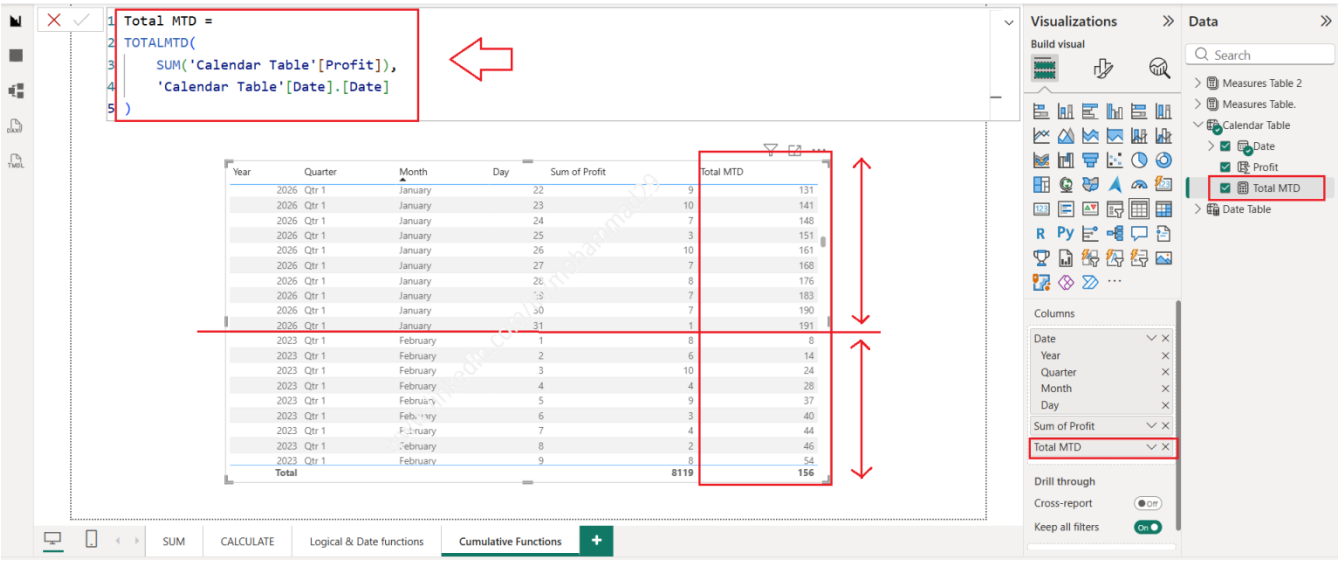






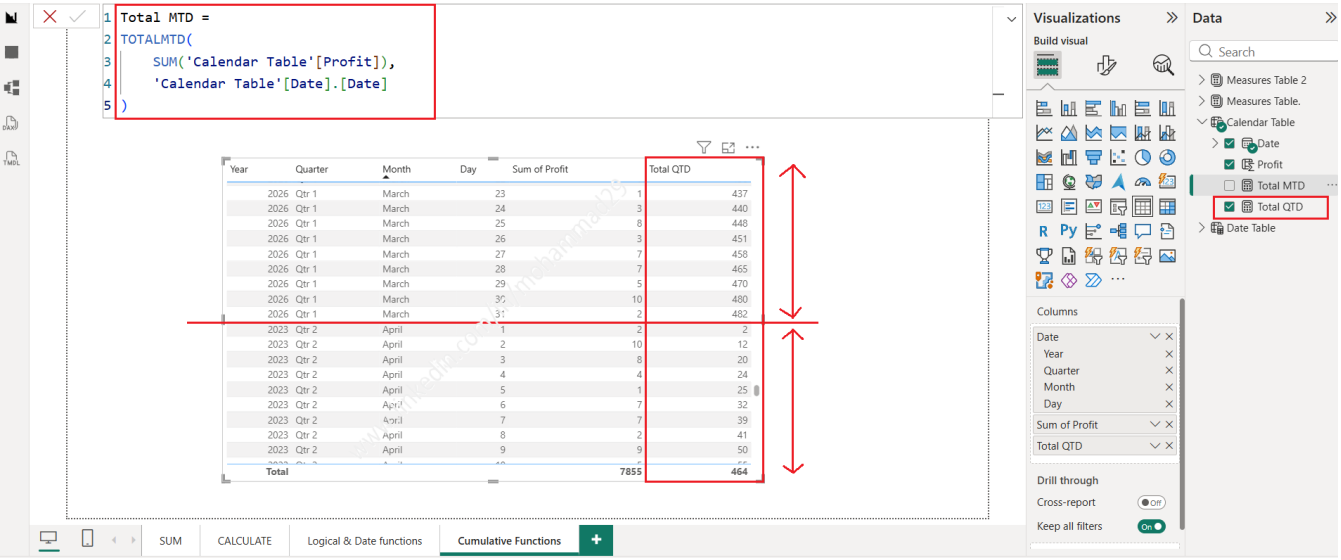
1. MTD (Month-To-Date): Calculates the cumulative total from the first day of the current month up to the current date.

*Q. How much sales have we made **this month so far**?*



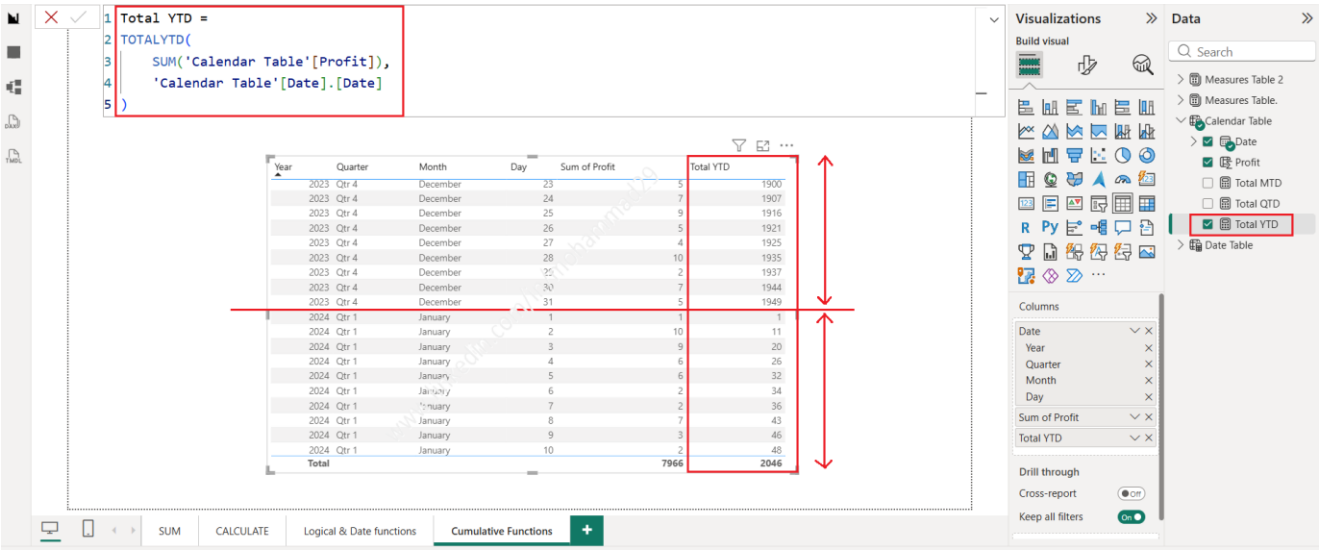
2. QTD (Quarter-To-Date): Calculates the cumulative total from the beginning of the current quarter to the current date.

*Q. What is the total revenue **this quarter to date**?*



3. YTD (Year-To-Date): Calculates the cumulative total from the first day of the year up to the current date.

*Q. What is the performance **from the start of the year till today**?*



Year	Quarter	Month	Day	Sum of Profit	Total YTD
2023	Qtr 4	December	23	5	1900
2023	Qtr 4	December	24	7	1907
2023	Qtr 4	December	25	9	1916
2023	Qtr 4	December	26	5	1921
2023	Qtr 4	December	27	4	1925
2023	Qtr 4	December	28	10	1935
2023	Qtr 4	December	29	2	1937
2023	Qtr 4	December	30	7	1944
2023	Qtr 4	December	31	5	1949
2024	Qtr 1	January	1	1	1
2024	Qtr 1	January	2	10	11
2024	Qtr 1	January	3	9	20
2024	Qtr 1	January	4	6	26
2024	Qtr 1	January	5	6	32
2024	Qtr 1	January	6	2	34
2024	Qtr 1	January	7	2	36
2024	Qtr 1	January	8	7	43
2024	Qtr 1	January	9	3	46
2024	Qtr 1	January	10	2	48
Total				7966	2046

- SAMEPERIODLASTYEAR() in Power BI (DAX)

SAMEPERIODLASTYEAR() is a **time-intelligence function** in DAX used to compare current period data with the **same period from the previous year**.

It is usually used inside CALCULATE() to modify the filter context.

- Purpose of SAMEPERIODLASTYEAR()

The main goal of SAMEPERIODLASTYEAR() is to:

- Compare current performance with last year.
- Track business growth or decline.
- Support trend and variance analysis.
- Enable historical benchmarking.

It always works based on a **Date column from a proper Calendar/Date table**.

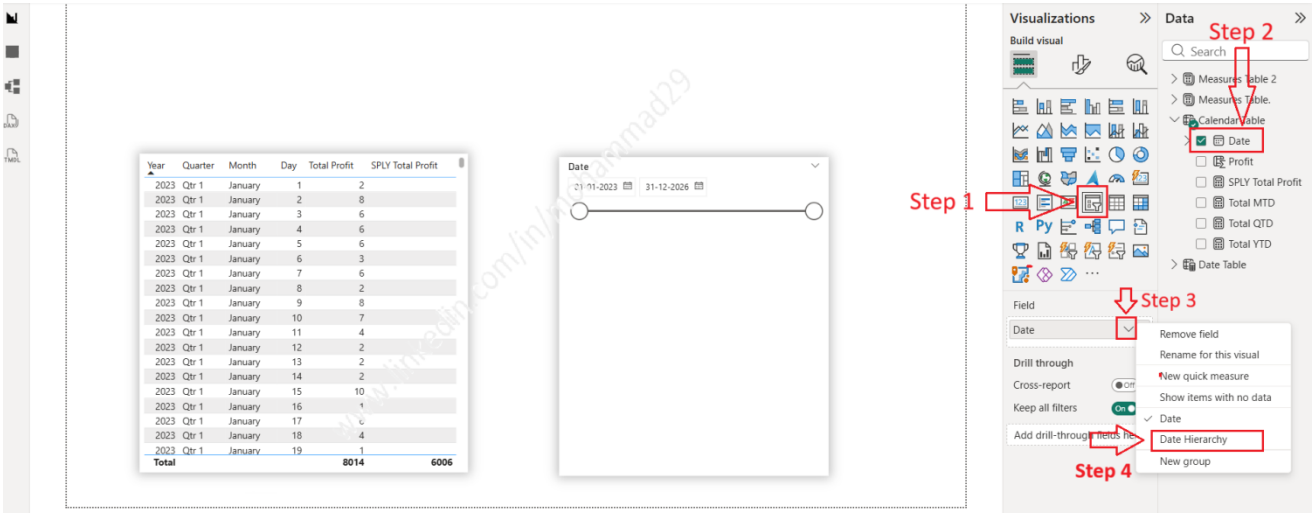
- Types of Analysis Using SAMEPERIODLASTYEAR()

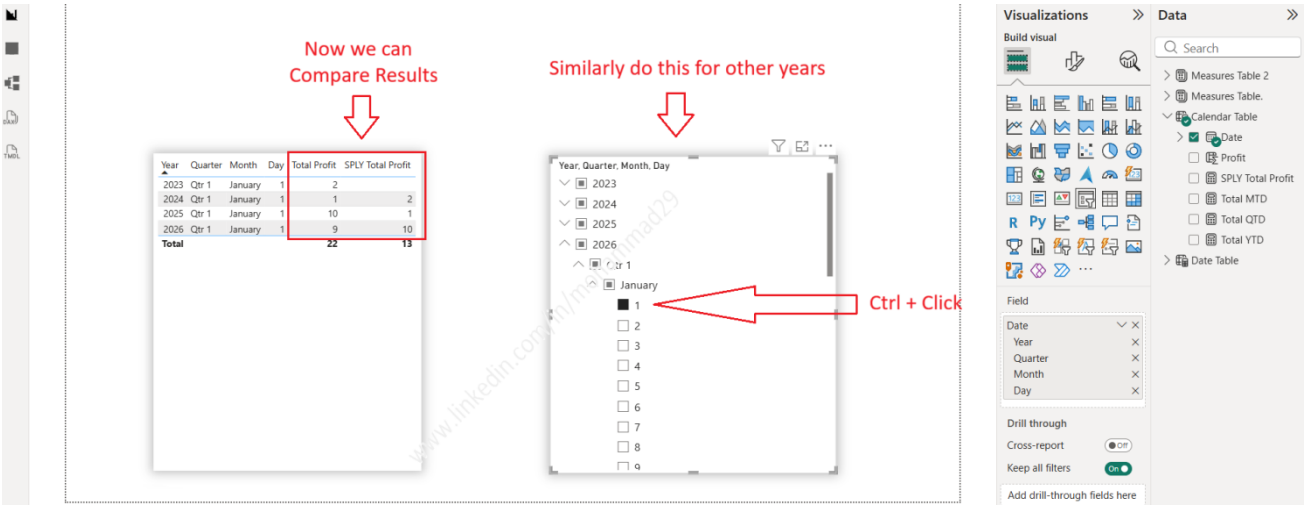
You can combine SAMEPERIODLASTYEAR() with:

1. **Total Measures** – Compare totals with last year.
2. **MTD Measures** – Compare month-to-date with last year.
3. **QTD Measures** – Compare quarter-to-date with last year.
4. **YTD Measures** – Compare year-to-date with last year.

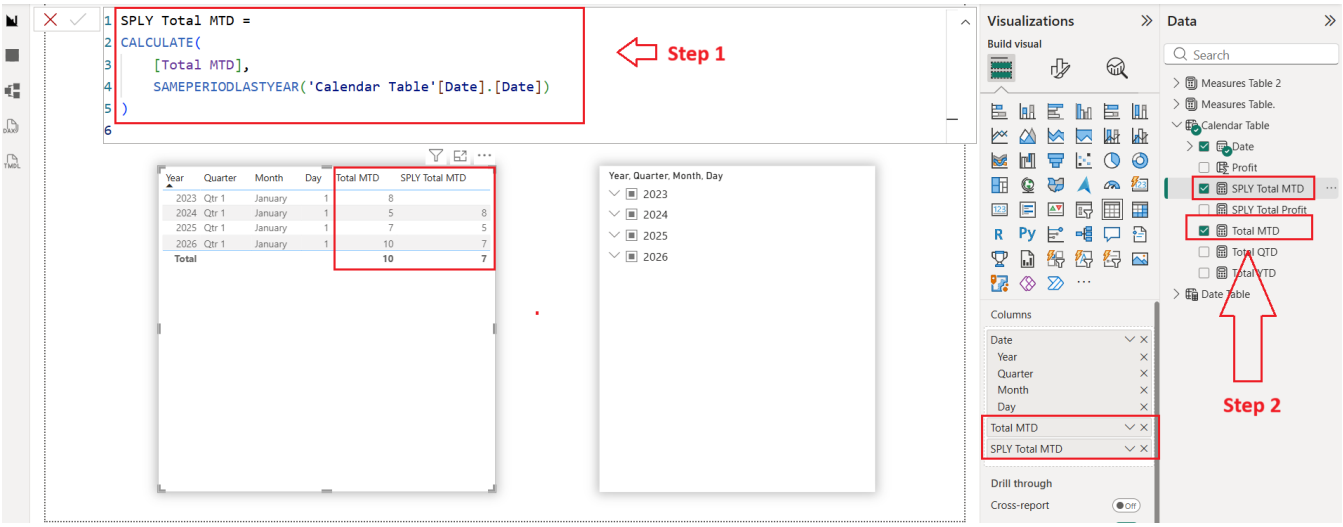
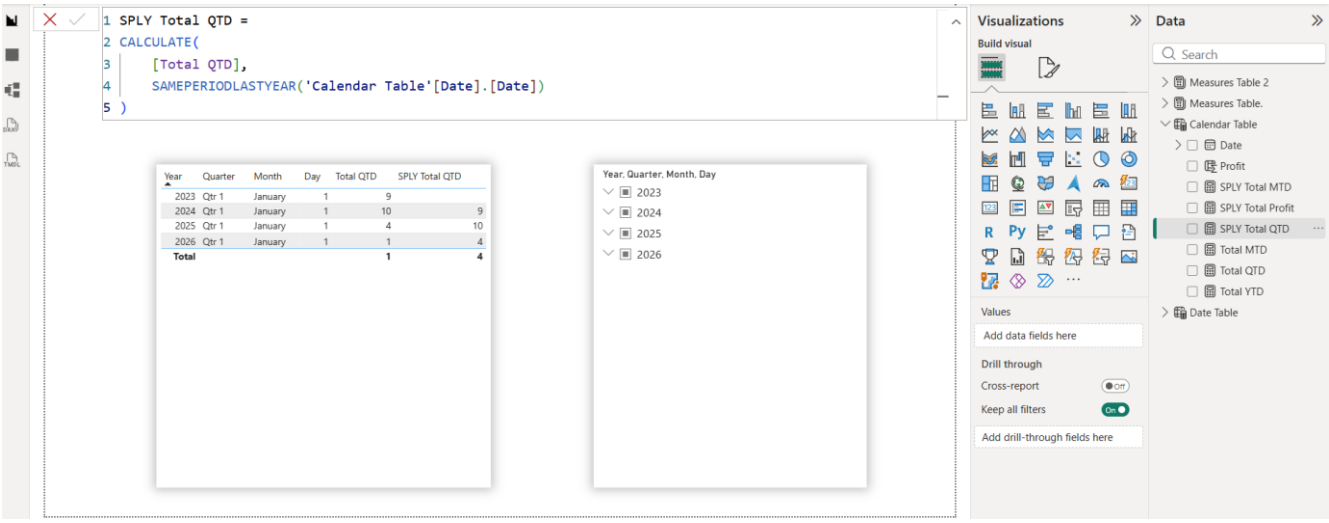
This makes reports more powerful and business-oriented.

Q: *Write a DAX measure to calculate the* ***Total Profit for the same period last year (SPLY Total Profit)*** *using CALCULATE() and SAMEPERIODLASTYEAR().*





Q: *Write a DAX measure to calculate the* ***Month-To-Date (MTD) value for the same period last year (SPLY MTD)*** *using CALCULATE() and SAMEPERIODLASTYEAR().*



Q: *Write a DAX measure to calculate the* ***Quarter-To-Date (QTD) value for the same period last year (SPLY QTD)*** *using CALCULATE() and SAMEPERIODLASTYEAR().*