10

DAX

functions that I use in (almost) every

Power BI project/model/report

1

Aggregators & iterators, such as

SUM / SUMX

SUM(<column>)

SUMX(<table>, <expression>)

Aggregators like **SUM** are straightforward and operate on entire columns, such as summing up sales.

Iterators like **SUMX** calculate row by row, applying expressions before aggregation. These are critical for more complex scenarios where calculations depend on other columns or involve custom logic.

Examples

SUM(Sales[Amount]) - Returns the sum of sales amount

SUMX (Sales, Sales[Quantity] * Sales[Net Price]) - Calculates total revenue by multiplying Quantity and Net Price for each row in the Sales table and summing the results

2

DIVIDE

DIVIDE (<Numerator>, <Denominator> [, <AlternateResult>])

A safer alternative to standard division that handles division by zero. Instead of producing an error, **DIVIDE** returns a specified alternative value, such as zero or blank.

Example

DIVIDE(SUM(Sales[Profit]), SUM(Sales[Revenue]), 0) - Calculates profit margin by dividing total profit by total revenue, using DIVIDE to avoid errors if revenue is zero, returning 0 instead.

3

COUNT / DISTINCTCOUNT

COUNT (<ColumnName>)
DISTINCTCOUNT (<ColumnName>)

Count functions are crucial for summarizing data.

COUNT counts the rows in a column, useful for metrics like total transactions.

DISTINCTCOUNT counts unique values, often used for unique customer counts or product diversity.

Examples

COUNT(Sales[Order ID]) - Counts the total number of orders in the Sales table.

DISTINCTCOUNT(Sales[Customer ID]) - Counts the unique number of customers in the Sales table by identifying each unique Customer ID. This helps track the number of unique buyers.

4

CALCULATE

CALCULATE (<Expression> [, <Filter> [, <Filter> [, …]]])

One of the most **important** functions in DAX, used to change or add filters dynamically.

It allows you to adjust the filter context of your measures to calculate values for specific scenarios, such as: isolating sales for a particular product category, calculating year-to-date totals, or applying region-specific filters in a single measure.

Example

CALCULATE(SUM(Sales[Amount]), Products[Category] = "Electronics") - Calculates the total sales amount specifically for products in the "Electronics" category.

5

IF / SWITCH

IF (<LogicalTest>, <ResultIfTrue> [, <ResultIfFalse>])

SWITCH (<Expression>, <Value>, <Result> [, <Value>, <Result> [, …]] [, <Else>])

These logical functions let you implement business logic directly in your measures, enabling conditional calculations or formatting, data labeling, or complex grouping.

IF handles simple conditions, while **SWITCH** is a more flexible alternative for handling multiple conditions.

Examples

IF(Sales[Amount] > 1000, "High Value", "Standard") - Classifies orders based on their amount

SWITCH(TRUE(), Sales[Amount] > 10000, "Excellent", Sales[Amount] > 5000, "Good", Sales[Amount] > 1000, "Average", "Below Average") - Assigns performance levels based on sales

6

AND (&&) / OR(||)

AND (<Logical1>, <Logical2>) / <Logical1> && <Logical2>...

OR (<Logical1>, <Logical2>) / <Logical1> || <Logical2> ...

Allows you to combine multiple logical conditions for flexibility.

AND (&&) ensures all conditions are true, while **OR** (||) requires at least one to be true.

You can use the AND or OR functions, but this may require nested calls. Alternatively, you can directly use the && / || operators to simplify your expressions.

Examples

Using AND function: IF(AND(Products[Stock] < 20, Products[Demand] > 500), "Top Priority", "Normal")

Using && operator: IF(Products[Stock] < 20 && Products[Demand] > 500, "Top Priority", "Normal")

- Flags products as "Top Priority" if stock is low and demand is high

7

DATEADD / SAMEPERIODLASTYEAR

DATEADD (<Dates>, <NumberOfIntervals>, <Interval>)

SAMEPERIODLASTYEAR (<Dates>)

Essential for time intelligence calculations.

DATEADD shifts a date range by a specific interval (e.g., months, years).

SAMEPERIODLASTYEAR compares the current period with the same period in the previous year.

Examples

CALCULATE(SUM(Sales[Amount]), DATEADD(Calendar[Date], -1, MONTH)) -
Calculates sales from the previous month

CALCULATE(SUM(Sales[Amount]), SAMEPERIODLASTYEAR(Calendar[Date])) -
Calculates sales for the same period last year

8

KEEPFILTERS

KEEPFILTERS (<Expression>)

Function used within CALCULATE to add a new filter without overriding existing filters on the same column. This is particularly useful when you want to refine the filter context without losing previous filters applied in the report or matrix.

Example:

CALCULATE([Sales Amount], KEEPFILTERS(Products[Color] = “Blue")) - In this example, KEEPFILTERS ensures that the “Blue" filter is added without replacing any other active filters on Products[Color]. This maintains the existing color context in the report while still filtering for blue products.

9

SELECTEDVALUE

SELECTEDVALUE (<ColumnName> [, <AlternateResult>])

Returns the value of a column in a slicer or single selection. Useful to dynamically adapt outputs based on user selections.

Example

SELECTEDVALUE(Regions[Region], "All Regions") - Displays the selected region dynamically

10

USERELATIONSHIP

USERELATIONSHIP (<ColumnName1>, <ColumnName2>)

Activates an inactive relationship between tables in the scope of a calculation. This is especially useful when you have multiple relationships between tables (e.g., Order Date and Ship Date) and need to switch contexts for specific calculations.

Example

CALCULATE(SUM(Sales[Amount]), USERELATIONSHIP(Sales[Ship Date], Calendar[Date]))
- USERELATIONSHIP activates the relationship between Sales[Ship Date] and Calendar[Date] to calculate sales by shipment date. This relationship is usually inactive because the active relationship is set to Order Date.

BONUS

CALENDAR

CALENDAR (<StartDate>, <EndDate>)

If your data model doesn't include a built-in date table, you can create one using the CALENDAR function in DAX. A date table is essential for time intelligence calculations and ensures consistent date filtering across your report.

CALENDAR generates a table with a continuous range of dates (static or dynamic) that you define, making it easy to create a comprehensive date table.

Example

CALENDAR(DATE(2020, 1, 1), DATE(2023, 12, 31)) - *Creates a date table that spans from January 1, 2020, to December 31, 2023*

What about you?

What DAX function do you use the most?

All icons coming from https://www.flaticon.com/