CODE BASICS

15 Essential DAX Functions

Every Power BI Developer Must Master!

AGGREGATION FUNCTIONS:

Aggregation functions involve summarizing or combining numerical data to provide insights such as totals, averages, counts, minima, and maxima.

1 SUM:

Returns the sum of all the numbers in a column.

Syntax: *SUM(ColumnName)*

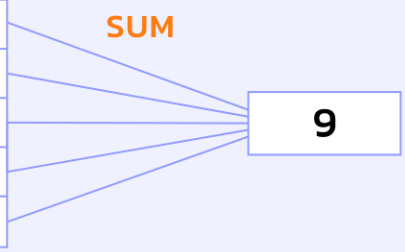
Example: Calculate the total quantity sold.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Quantity
2
1
3
1
2



Qty Sold = SUM(Sales[Quantity])

Qty Sold = SUM(2+1+3+1+2)

Qty Sold = 9

2 AVERAGE:

Returns the average (arithmetic mean) of all the numbers in a column.

Syntax: *AVERAGE(ColumnName)*

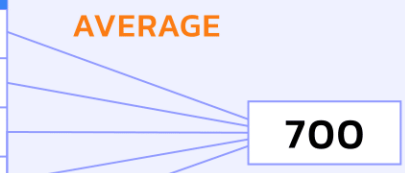
Example: Calculate the average price of transactions.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Price
500
800
450
900
850



AVG Price=
AVERAGE(Sales[Price])

AVG Price=
(500+800+450+900+850)/5

AVG Price= 700

3 MIN:

Returns the smallest value in a column, or between two scalar expressions.

Syntax: *MIN(ColumnName)* or *MIN(Expression1, Expression2)*

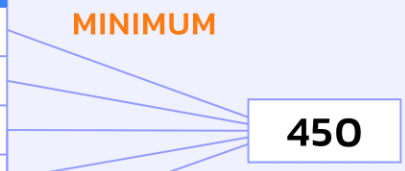
Example: Find the smallest price.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Price
500
800
450
900
850



MIN Price=
MIN(Sales[Price])

MIN Price=
MIN(500,800,450,900,850)

MIN Price= 450

4 MAX:

Returns the largest value in a column, or between two scalar expressions.

Syntax: *MAX(ColumnName)* or *MAX(Expression1, Expression2)*

Example: Find the largest price.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Price
500
800
450
900
850

MAXIMUM → 900

MAX Price=
MAX(Sales[Price])

MAX Price=
MAX(500,800,450,900,850)

MIN Price= 900

"X" FUNCTIONS FOR AGGREGATION:

The major drawback of basic aggregate functions is that they cannot perform filtering/row-by-row evaluation while aggregating values. 'X' functions help in overcoming this drawback.

The "X" functions perform two main steps:

Iteration: They iterate over each row in the specified table or a set of rows.

Aggregation: After applying the given expression to each row, they aggregate these individual results into a single value.

SUMX:

Returns the sum of an expression evaluated for each row in a table.

Syntax: ***SUMX(Table, Expression)***

Example: Calculate the total sales value (Quantity * Price for each transaction).

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Quantity	Price	Applying condition Quantity* Price
2	500	1000
1	800	800
3	450	1350
1	900	900
2	850	1700

SUM → 5750

Total Price=
SUMX(Sales,Sales[Quantity]*Sales[Price])

Total Price=
2*500+1*800+3*450+1*900+850

Total Price=
100+800+1350+900+1700

Total Price=5750

The other 'X' functions in DAX, such as AVGX, MINX, MAXX, and others, work similarly to SUMX by allowing for row-by-row evaluation of an expression across a table or table expression, and then performing the respective aggregation operation based on the results of that evaluation.

FILTER FUNCTIONS:

Filtering functions are a crucial part of DAX, providing the capability to manipulate data context, which is fundamental for creating dynamic and context-sensitive calculations.

5 CALCULATE:

Modifies the filter context for a given expression.

Syntax: *CALCULATE(Expression, [Filter1, Filter2,...])*

Example: Calculate the total quantity sold for the 'North' region.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
3	A	3	400	North

Quantity
2
3

SUM → 5

North QTY=
CALCULATE(SUM(Sales[Quantity]), Sales[Region]="North")

North QTY= 2+3

North QTY= 5

6 FILTER:

Returns a table that includes only the rows that meet a certain condition.

Syntax: *FILTER(Expression,Filter)*

Example: Filters transactions over North region.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

North QTY=
FILTER(Sales,Sales[Region]="North")

Explanation:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
3	A	3	400	North

7 ALL:

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

Syntax: *ALL(TableName or ColumnName, [Column1,...])*

Example: Calculate the total quantity ignoring the Region filter.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

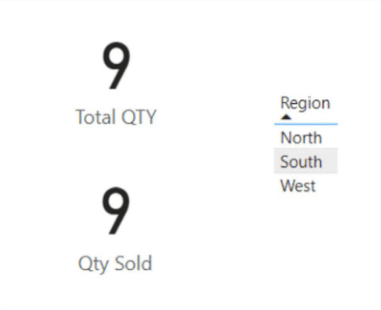
Explanation:

Total QTY=
CALCULATE(SUM(Sales[Quantity]), ALL(Region)

Total QTY=2+1+3+1+2

Total QTY=9

Without Filters:

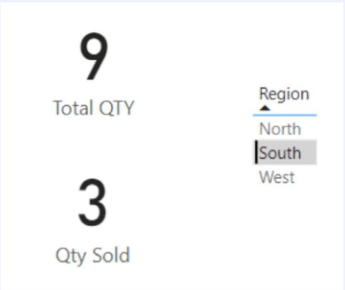


Qty Sold=SUM(Sales[Quantity])

Qty Sold=SUM(2+1+3+1+2)

Qty Sold=9

With Filters:



8 ALLEXCEPT:

Removes all context filters in the table except filters that have been applied to the specified columns.

Syntax: *ALLEXCEPT(TableName, Column1,[Column2,...])*

Example: Calculate the total quantity ignoring all filters except the Region filter.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

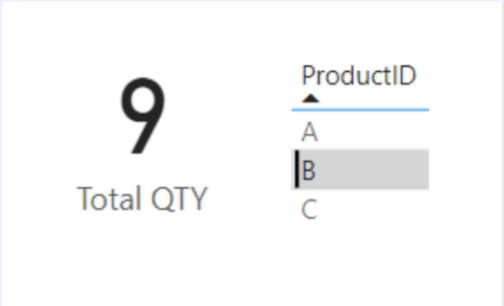
Explanation:

Total QTY=
CALCULATE(SUM(Sales[Quantity]), ALLEXCEPT(Region)

Total QTY=2+1+3+1+2

Total QTY=9

With Other Filters:



With Region Filter:

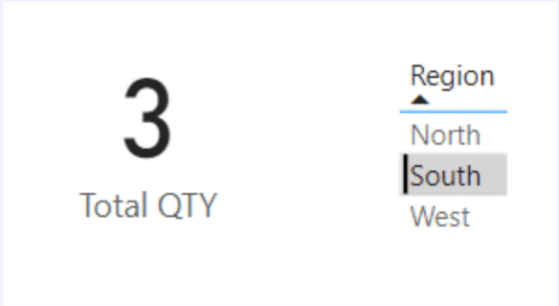


TABLE MANIPULATION FUNCTIONS:

These functions return a table or manipulate existing tables.

9 DISTINCT:

Returns a table containing only distinct rows.

Syntax: *DISTINCT(TableName)*

Returns a column of unique values.

Syntax: *DISTINCT(ColumnName)*

Example: List unique product IDs sold.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Products=DISTINCT(Sales[ProductID])

Product ID
A
B
C

MATH AND TRIG FUNCTIONS:

These are functions in DAX that allow for the execution of mathematical and trigonometric operations on data.

10 ABS:

Returns the absolute value of a number.

Syntax: *ABS(Number)*

Example: ABS(10-15)
5

11 DIVIDE:

Performs division and returns an alternate result or BLANK on division by 0.

Syntax: *DIVIDE(Numerator, Denominator, [AlternateResult])*

Example: = DIVIDE(8,2,0)
= 4

Example: DIVIDE(3,0,0)
= 0

LOGICAL FUNCTIONS:

Logical functions act upon an expression to return information about the values or sets in the expression.

12 IF:

Checks a condition, and returns one value if True, and another value if False.

Syntax: *IF(LogicTest, ResultIfTrue, [ResultIfFalse])*

Example: Categorize transactions as 'High' or 'Low' based on price.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

Category = IF(Sales[Price] >= 800, "High", "Low")

Price	Applying condition	Category
500	→	Low
800	→	High
450	→	Low
900	→	High
850	→	High

13 SWITCH:

Evaluates an expression against a list of values and returns the result corresponding to the first matching value.

Syntax: ***SWITCH(Expression, Value1, Result1, Value2, Result2, ..., [DefaultResult])***

Example: Categorize transactions as 'High Price' or 'Medium Price' or 'Low Price' based on price.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

```
Price Category = SWITCH(TRUE(),
    Sales[Price] >= 800, "High Price",
    Sales[Price] >= 500 && Sales[Price] < 800, "Medium Price",
    Sales[Price] < 500, "Low Price",
    "Undefined" // Default case if no other conditions are met
)
```

Price	Applying condition	Price Category
500	→	Medium Price
800	→	High Price
450	→	Low Price
900	→	High Price
850	→	High Price

RELATIONSHIP FUNCTIONS:

Relationship functions facilitate data flow between tables when there is an established relationship between them.

15 RELATED:

Returns a related value from another table.

Syntax: *RELATED(ColumnName)*

Example: Retrieve related product details for sales transactions.

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

'Products' Table:

Product ID	Product Name	Quantity
A	Widget	Electronics
B	Gadget	Home
C	Other	Electronics

Explanation:

Product Details = RELATED(Products[ProductName])

Transaction ID	Product ID	Quantity	Price	Region	Category	Price Category	Product Category
1	A	2	500	North	Low	Medium Price	Widget
2	B	1	800	South	High	High Price	Gadget
3	A	3	450	North	Low	Low Price	Widget
4	C	1	900	West	High	High Price	Other
5	B	2	850	South	High	High Price	Gadget

15 RELATEDTABLE:

Retrieves a table of rows related to the current row context based on an existing relationship.

Syntax: ***RELATEDTABLE(TableName)***

Example: Retrieve related average price for each category.

'Products' Table:

Product ID	Product Name	Quantity
A	Widget	Electronics
B	Gadget	Home
C	Other	Electronics

'Sales' Table:

Transaction ID	Product ID	Quantity	Price	Region
1	A	2	500	North
2	B	1	800	South
3	A	3	450	North
4	C	1	900	West
5	B	2	850	South

Explanation:

AVG Price=
CALCULATE(AVERAGE(Sales[Price]),RELATEDTABLE(sales_data))

Product ID	Product ID	Category	AVG Price
A	Widget	Electronics	475
B	Gadget	Home	825
C	Other	Electronics	900