

DAX

Intermediate

Alish Bista - 10/10/2025

Variables

Vary..Not very much

- Avoid repeating same expression
- Improve code readability
- Helps in code optimization (computed once)
- Has **Lazy** evaluation (not evaluated unless used)
- Definition requires a **RETURN** statement

```
VAR TotalSales = SUM ( Sales[SalesAmount] )  
VAR TotalCosts = SUM ( Sales[TotalProductCost] )  
VAR GrossMargin = TotalSales - TotalCosts  
RETURN  
    GrossMargin / TotalSales
```

Aggregators and Iterators

Aggregators

Summarize One At A Time

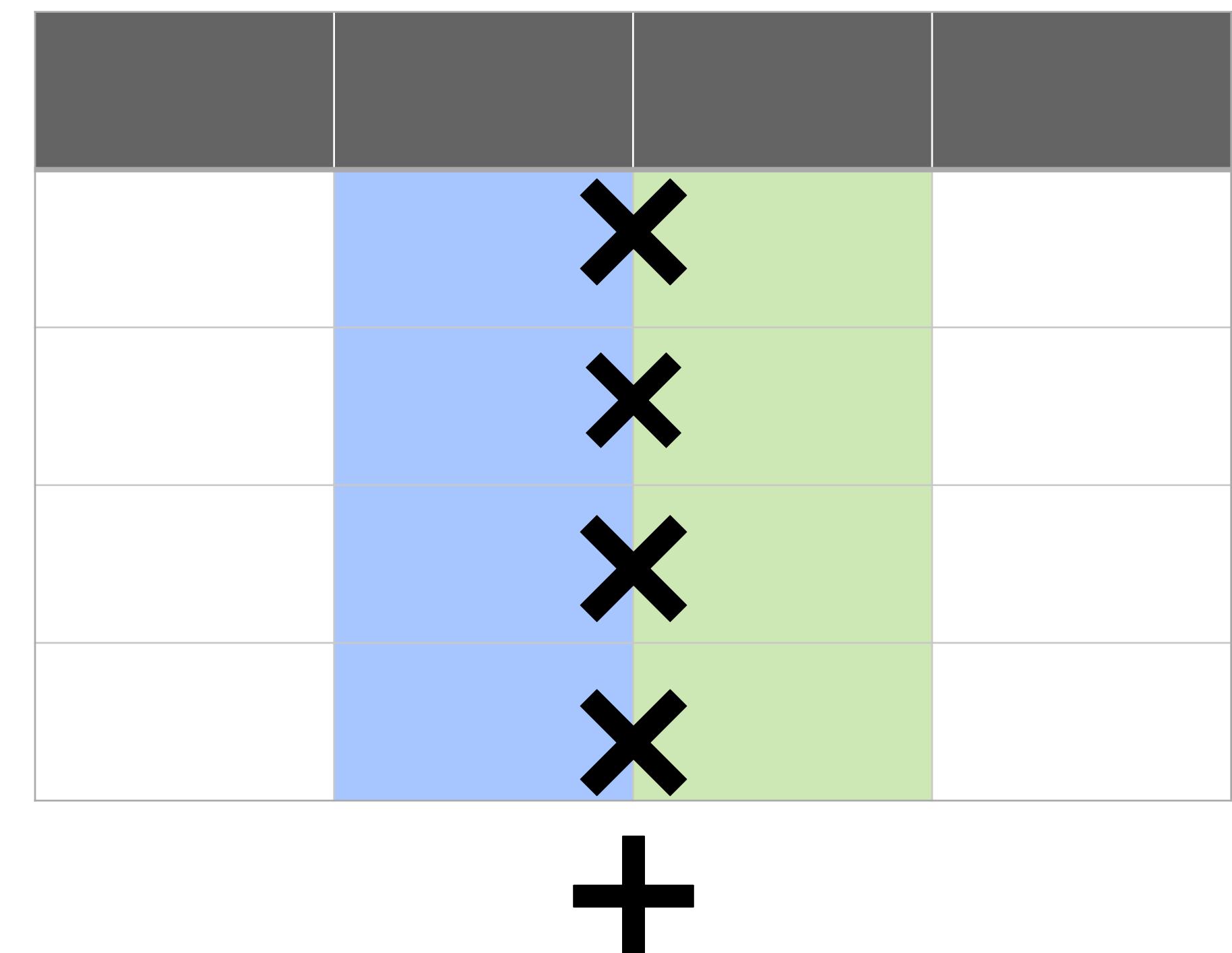
- Functions that work on entire columns at once to give you a single result
- Basically column summarizers that crunch all the data and give you one number
- Common examples: *SUM, AVERAGE, COUNT, MIN, MAX*


```
TotalSales = SUM ( Sales[SalesAmount] )
```

Iterators

Go Row By Row

- Functions that go through the table row-by-row, doing calculations on each row
- Then they aggregate the results into a final answer
- Usually end with an “X”: SUMX, AVERAGEX, COUNTX, MINX, MAXX



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

Common Functions

Aggregation Functions

Handle Aggregations

- Functions like MIN, MAX, SUM, COUNT and their families.
 - **COUNT** operates on any data type, apart from Boolean.
 - **COUNTA** operates on any type of column.
 - **COUNTBLANK** returns the number of empty cells (blanks or empty strings) in a column.
 - **COUNTROWS** returns the number of rows in a table.
 - **DISTINCTCOUNT** returns the number of distinct values of a column, blank value included if present.
 - **DISTINCTCOUNTNOBLANK** returns the number of distinct values of a column, no blank value included.

Logical Functions

This or That

- Allows implementing different calculations depending on the value of a column
- Helps to intercept an error condition
- Common Functions:
 - AND, OR, NOT
 - IF, SWITCH
 - IFERROR
 - TRUE, FALSE

Information Functions

Brief Me

- Useful in analyzing type of an expression
- Returns a *boolean* value
- Common Functions:
 - ISBLANK
 - ISERROR
 - ISLOGICAL
 - ISNONTEXT
 - ISNUMBER
 - ISTEXT

Mathematical Functions

Do The Maths

- **ROUND** - Rounds a number to a specified number of digits - $ROUND(3.14159, 2) = 3.14$, $ROUND([Sales], 0)$
- **ABS** - Absolute value - $ABS(-25) = 25$, $ABS([Profit])$
- **MOD** - Remainder after division - $MOD(10, 3) = 1$, $MOD([OrderID], 2)$
- **POWER** - Number raised to a power - $POWER(2, 3) = 8$, $POWER([Sales], 2)$
- **SQRT** - Square root - $SQRT(16) = 4$, $SQRT([Area])$
- **QUOTIENT** - Integer division (no remainder) - $QUOTIENT(10, 3) = 3$, $QUOTIENT([TotalMinutes], 60)$
- **TRUNC** - Truncates a number to an integer (removes decimals) - $TRUNC(3.14159) = 3$, $TRUNC([Price])$
- **SIGN** - Sign of a number (1, -1, or 0) - $SIGN(-15) = -1$, $SIGN([NetIncome])$
- **RANDBETWEEN** - Random integer between two numbers - $RANDBETWEEN(1, 100)$, $RANDBETWEEN(1, 6)$
- **LOG10** - Base-10 logarithm - $LOG10(1000) = 3$, $LOG10([Value])$
- **Honorable Mentions:** $RAND$, PI , $EVEN$, ODD , EXP , LN , LOG , $FACT$, GCD , LCM

Text Functions

Texting Around

- **FORMAT** - Formats a value as text using a format string - *FORMAT([Date], "MMM YYYY") = "Jan 2025", FORMAT([Amount], "\$#,##0.00")*
- **CONCATENATE / CONCATENATEX** - Joins text strings; CONCATENATEX iterates through tables - *CONCATENATE([FirstName], " ", [LastName]), CONCATENATEX(Products, [ProductName], ", ")*
- **LEFT / RIGHT / MID** - Extracts characters from start/end/middle of text - *LEFT([Code], 2), RIGHT([Account], 4), MID([Phone], 4, 3)*
- **LEN** - Returns the number of characters in text - *LEN("Hello") = 5, LEN([Description])*
- **UPPER / LOWER** - Converts text to uppercase/lowercase - *UPPER("hello") = "HELLO", LOWER("HELLO") = "hello"*
- **TRIM** - Removes extra spaces from text - *TRIM(" Hello ") = "Hello", TRIM([CustomerName])*
- **SUBSTITUTE / REPLACE** - Replaces text; SUBSTITUTE finds and replaces, REPLACE by position - *SUBSTITUTE([Text], " ", ""), REPLACE([Code], 1, 2, "XX")*
- **VALUE** - Converts text to a number - *VALUE("123") = 123, VALUE([StringAmount])*
- **FIND / SEARCH** - Returns position of text; FIND is case-sensitive, SEARCH is not - *FIND("@", [Email]), SEARCH("error", [Log])*

Conversion Functions

Transform!

- **CURRENCY** - Converts an expression to Currency type - `CURRENCY([Amount]) = currency format`
- **INT** - Converts an expression to Integer - `INT(3.7) = 3, INT([Decimal])`
- **VALUE** - Converts text to numeric format - `VALUE("123.45") = 123.45, VALUE([StringNumber])`
- **FORMAT** - Converts numeric/date values to text with specified format - `FORMAT(DATE(2019, 01, 12), "yyyy mmm dd") = "2019 Jan 12", FORMAT([Sales], "$#,##0")`
- **DATE** - Creates a DateTime from year, month, day parameters - `DATE(2019, 1, 12) = January 12, 2019`
- **TIME** - Creates a time from hour, minute, second parameters - `TIME(14, 30, 0) = 2:30 PM`
- **DATEVALUE** - Converts text string to DateTime value - `DATEVALUE("28/02/2018") = Feb 28, 2018, DATEVALUE("2018-02-28")`
 - **DATEVALUE** Special Behavior:
 - Tries European format (dd/mm/yy) first, then American format (mm/dd/yy) if that fails
 - Supports unambiguous ISO format (yyyy-mm-dd)
 - `DATEVALUE("28/02/2018") = Feb 28 (European)`
 - `DATEVALUE("02/28/2018") = Feb 28 (American)`
 - `DATEVALUE("2018-02-28") = Feb 28 (unambiguous)`
 - Automatically switches month/day to avoid errors when possible

Date and Time Functions

Dictating Conditions

- DATE - Creates a DateTime from year, month, day parameters - DATE(2019, 1, 12) = January 12, 2019, DATE([Year], [Month], [Day])
- DATEVALUE - Converts text string to DateTime value - DATEVALUE("28/02/2018") = Feb 28, 2018, DATEVALUE("2018-02-28")
- DAY - Returns the day of the month (1-31) - DAY(DATE(2019, 1, 12)) = 12, DAY([OrderDate])
- EDATE - Returns date that is specified months before/after a date - EDATE(DATE(2019, 1, 15), 3) = April 15, 2019, EDATE([StartDate], 6)
- EOMONTH - Returns the last day of the month, specified months away - EOMONTH(DATE(2019, 1, 15), 0) = Jan 31, 2019, EOMONTH([InvoiceDate], 1)
- HOUR - Returns the hour (0-23) - HOUR(TIME(14, 30, 0)) = 14, HOUR([DateTime])
- MINUTE - Returns the minute (0-59) - MINUTE(TIME(14, 30, 0)) = 30, MINUTE([DateTime])
- MONTH - Returns the month (1-12) - MONTH(DATE(2019, 1, 12)) = 1, MONTH([OrderDate])
- NOW - Returns the current date and time - NOW() = 2025-11-06 10:45:23
- TODAY - Returns the current date (no time) - TODAY() = 2025-11-06
- WEEKDAY - Returns day of the week (1-7, Sunday=1) - WEEKDAY(DATE(2019, 1, 12)) = 7, WEEKDAY([OrderDate])
- WEEKNUM - Returns the week number of the year (1-53) - WEEKNUM(DATE(2019, 1, 12)) = 2, WEEKNUM([Date])
- YEAR - Returns the year - YEAR(DATE(2019, 1, 12)) = 2019, YEAR([OrderDate])

Q&A