**Some of the Most Used DAX:**  
  
-- Core KPIs

Total Sales := SUM(FactSales[SalesAmount])

Total Cost := SUM(FactSales[CostAmount])

Total Profit := [Total Sales] - [Total Cost]

Margin % := DIVIDE([Total Profit], [Total Sales], 0)

-- Counts

Orders := DISTINCTCOUNT(FactSales[OrderID])

Customers := DISTINCTCOUNT(FactSales[CustomerID])

**COUNT, COUNTROWS**

-- Time

Sales MTD := TOTALMTD([Total Sales], DimDate[Date])

Sales YTD := TOTALYTD([Total Sales], DimDate[Date])

Sales LY YTD := CALCULATE([Total Sales], SAMEPERIODLASTYEAR(DimDate[Date]))

Sales LY := CALCULATE([Total Sales], DATEADD(DimDate[Date], -1, YEAR))

YoY % := DIVIDE([Total Sales] - [Sales LY], [Sales LY])

-- Percent of visible total

Sales % of Visible := DIVIDE([Total Sales],

CALCULATE([Total Sales], ALLSELECTED(DimProduct)))

-- Top N label

Top 5 Products Label :=

CONCATENATEX(

TOPN(5, VALUES(DimProduct[ProductName]), [Total Sales], DESC),

DimProduct[ProductName],

", "

)

-- Rank

Product Rank := RANKX(ALL(DimProduct[ProductName]), [Total Sales], , DESC, Dense)

-- Rolling 12 months

Sales Rolling 12M :=

CALCULATE([Total Sales], DATESINPERIOD(DimDate[Date], MAX(DimDate[Date]), -12, MONTH))

– Active Customers in the last 90 Days

Active Customers (90d) :=

CALCULATE(

DISTINCTCOUNT(FactSales[CustomerID]),

DATESINPERIOD(DimDate[Date], MAX(DimDate[Date]), -90, DAY)

)

– Running/Cumulative Total (by slicers respected):

Cumulative Sales :=

VAR LastDate = MAX(DimDate[Date])

RETURN

CALCULATE([Total Sales], FILTER(ALLSELECTED(DimDate[Date]), DimDate[Date] <= LastDate))

– IF, SWITCH(TRUE()), AND/OR, IN

Order Size Band :=

SWITCH(

TRUE(),

[Total Sales] >= 50000, "Large",

[Total Sales] >= 10000, "Medium",

"Small"

)

**DAX Core Concepts (foundation):**

| **Concept** | **Explanation** |
| --- | --- |
| **Row Context** | DAX evaluates each row individually (e.g., in a calculated column or iterator like SUMX). |
| **Filter Context** | The current filters applied by visuals, slicers, or CALCULATE. |
| **Context Transition** | When CALCULATE converts a row context into a filter context. |
| **VAR … RETURN** | Use variables for readability and performance. |
| **DIVIDE(n, d, 0)** | Safer than n/d — handles divide-by-zero gracefully. |

**Aggregation & Iterators:**

| **Function** | **Explanation** | **Example** | **Use Case** |
| --- | --- | --- | --- |
| **SUMX / AVERAGEX / MINX / MAXX** | Iterate over a table, evaluating an expression for each row, then aggregate. | SUMX(FactSales, FactSales[Qty]\*(FactSales[UnitPrice]-FactSales[UnitCost])) | Custom row-level math like profit margin. |
| **DISTINCTCOUNT** | Counts unique values, ignoring blanks. | DISTINCTCOUNT(FactSales[CustomerID]) | Unique customers, products, etc. |

**Filter Context Shaping (advanced DAX muscle):**

### **🧩 CALCULATE()**

**What it does:** changes the current filter context — think of it as “re-evaluating a measure under new conditions.”

Sales LY :=

CALCULATE([Total Sales], DATEADD(DimDate[Date], -1, YEAR))

CALCULATE replaces or adds filters before computing the measure.

It’s the only function that *changes* filter context.

Core of all advanced DAX.

### **🧩 FILTER()**

**What it does:** builds a new *virtual table* that contains only rows matching a condition.

HighValueSales :=

CALCULATE([Total Sales], FILTER(FactSales, FactSales[SalesAmount] > 500))

* Returns a table, not a value.
* Must be wrapped in CALCULATE to affect measure results.

### **🧩 KEEPFILTERS()**

**What it does:** narrows filters instead of replacing them.  
 Useful when combining slicer filters + function filters.

CALCULATE(

[Total Sales],

KEEPFILTERS( FILTER(FactSales, FactSales[Region] = "East") )

)

If you omit KEEPFILTERS, your new filter would override existing slicers.

🧩 ALL / ALLEXCEPT / ALLSELECTED / REMOVEFILTERS

| **Function** | **Meaning** | **Common Use** |
| --- | --- | --- |
| **ALL(Table/Column)** | Removes all filters from the given scope. | % of Total ignoring slicers |
| **ALLEXCEPT(Table, col1, col2)** | Keeps only specified columns filtered. | % within category |
| **ALLSELECTED(Table)** | Keeps filters from visuals but ignores inner ones. | % of *visible* total |
| **REMOVEFILTERS(Table)** | Cleaner version of ALL in new syntax. | General unfiltering |

Example:

Sales % of Visible :=

DIVIDE([Total Sales], CALCULATE([Total Sales], ALLSELECTED(DimProduct)))

## **Relationship & Lookup Logic:**

### **🧩 RELATED() / RELATEDTABLE()**

* **RELATED**: from *many* to *one* (lookup).
* **RELATEDTABLE**: from *one* to *many* (fetch all children).

Order Country := RELATED(DimCustomer[Country])

DAX walks model relationships automatically — no need for joins.

### **🧩 USERELATIONSHIP()**

When two tables have multiple relationships (e.g., OrderDate vs ShipDate), only one is active.  
 USERELATIONSHIP() temporarily activates another.

Sales by Ship Date :=

CALCULATE([Total Sales], USERELATIONSHIP(FactSales[ShipDate], DimDate[Date]))

**Virtual Table Builders (power-user tools):**

| **Function** | **What It Does** | **Example / Insight** |
| --- | --- | --- |
| **VALUES() / DISTINCT()** | Returns unique values of a column (with/without blanks). | Great for slicer or rank bases. |
| **SUMMARIZE()** | Groups data and adds calculated columns (like SQL GROUP BY). | SUMMARIZE(FactSales, DimProduct[Category], "Sales", [Total Sales]) |
| **SUMMARIZECOLUMNS()** | Safer, measure-aware grouping for visuals. | Preferred in modern DAX. |
| **ADDCOLUMNS()** | Adds calculated columns to an existing table expression. | Used in ranking or filtering logic. |
| **SELECTCOLUMNS()** | Build a slimmed-down table with chosen columns. |  |
| **GENERATESERIES()** | Creates a sequential list (dates, numbers). | Great for scaffolding bins. |
| **UNION / INTERSECT / EXCEPT** | Set operations between tables. |  |
| **TOPN()** | Returns the top N rows based on an expression. | Used with CALCULATE to find Top N filters. |

Example:

Top Category Sales :=

CALCULATE(

[Total Sales],

KEEPFILTERS(TOPN(1, VALUES(DimProduct[Category]), [Total Sales], DESC))

)

Here, TOPN returns a table of the top 1 category, and CALCULATE re-evaluates [Total Sales] for that category only.

**Ranking & Segmentation:**

### **🧩 RANKX()**

Ranks each row in a table by a measure or expression.

Product Rank :=

RANKX(ALL(DimProduct[ProductName]), [Total Sales], , DESC, Dense)

* ALL removes filters to rank globally.
* Fifth parameter (Dense) means 1,2,3,... with no gaps for ties.

### **🧩 TREATAS()**

Applies filters from one table to another (without relationship).

CALCULATE([Total Sales], TREATAS(VALUES(Scenario[Category]), DimProduct[Category]))

Mimics a relationship at runtime.

Crucial for “disconnected slicers” (e.g., scenario tables, input what-if).

**Time Intelligence (calendar required!):**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| **DATEADD** | Shifts dates by given intervals (respects missing dates). | DATEADD(DimDate[Date], -1, YEAR) |
| **SAMEPERIODLASTYEAR** | Returns same range last year. | CALCULATE([Total Sales], SAMEPERIODLASTYEAR(DimDate[Date])) |
| **TOTALYTD / TOTALMTD / TOTALQTD** | Cumulative period totals. | TOTALYTD([Total Sales], DimDate[Date]) |
| **DATESINPERIOD** | Dynamic rolling window around a date. | Rolling 12 M total. |
| **DATESBETWEEN** | Specify explicit start and end dates. |  |

Example:

Rolling 12M Sales :=

CALCULATE(

[Total Sales],

DATESINPERIOD(DimDate[Date], MAX(DimDate[Date]), -12, MONTH)

)

DATESINPERIOD builds a 12-month virtual date table up to the current date.

**Statistical / Segmentation Functions:**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| **MEDIANX / PERCENTILEX.INC** | Return median or percentile value across a table. | Useful in pricing or RFM segmentation. |
| **NORM.DIST / NORM.S.DIST** | Normal distribution calculations. | For statistical modelling. |

### **1. PERCENTILEX.INC / PERCENTILEX.EXC**

**Purpose:** Calculate a percentile value (like 80th or 90th percentile) across a measure or column — just like Excel’s PERCENTILE.INC.

**Example: Identify top 10% customers by revenue**

Top10% Revenue Threshold :=

PERCENTILEX.INC(

ALL(DimCustomer),

[Total Sales],

0.9

)

**Explanation:**

* Returns the sales amount that separates the top 10% of customers.
* You can then compare each customer’s [Total Sales] with this value to mark them as “Top 10%.”

**Use Case:** Create “VIP Customer” or “High-Value Customer” flags dynamically for segmentation visuals.

### **🧭 2. MEDIANX**

**Purpose:** Finds the *median* (middle value) of an expression across a table.  
 More robust than averages for skewed data (like order values or delivery times).

**Example: Median Order Value**

Median Order Value :=

MEDIANX(

VALUES(FactSales[OrderID]),

[Total Sales per Order]

)

**Explanation:**

* Iterates each order and takes the median of sales amounts.
* Less sensitive to outliers (e.g., few huge orders).

**Use Case:** Useful when visualizing “typical order size” in CLTV or sales performance dashboards.

### **🎯 3. NORM.DIST / NORM.S.DIST**

**Purpose:** Returns the probability (or density) of a value in a *normal distribution*.  
 Useful for z-score–based standardization or statistical grading.

**Example: Customer Spending Z-Score**

AvgSalesAll := AVERAGEX(ALL(DimCustomer), [Total Sales])

StdSalesAll := STDEVX.P(ALL(DimCustomer), [Total Sales])

Customer Z-Score :=

VAR Mean = [AvgSalesAll]

VAR SD = [StdSalesAll]

RETURN

DIVIDE([Total Sales] - Mean, SD)

**Explanation:**

* Computes how far a customer’s spending deviates from the mean in standard deviations.
* You can later convert this into a probability using NORM.S.DIST([Z-Score], TRUE).

**Use Case:** Identify “above-average” or “below-average” spenders; highlight anomalies in performance dashboards.

💡 Bonus Pattern: RFM-Style Binning using Quantiles

Monetary\_Score :=

SWITCH(

TRUE(),

[Total Sales] >= PERCENTILEX.INC(ALL(DimCustomer), [Total Sales], 0.8), 5,

[Total Sales] >= PERCENTILEX.INC(ALL(DimCustomer), [Total Sales], 0.6), 4,

[Total Sales] >= PERCENTILEX.INC(ALL(DimCustomer), [Total Sales], 0.4), 3,

[Total Sales] >= PERCENTILEX.INC(ALL(DimCustomer), [Total Sales], 0.2), 2,

1

)

**Explanation:**

* Dynamically segments customers into 5 bins based on monetary percentile.
* Works even as new data arrives — fully automated segmentation.

**Use Case:** Directly applicable in **RFM Analysis**, **CLTV Models**, or **Customer Tiering Dashboards**.

| **Function** | **Realistic Business Use** | **Example Purpose** |
| --- | --- | --- |
| PERCENTILEX.INC | VIP segmentation | Top 10% revenue threshold |
| MEDIANX | Central tendency | Median order value |
| NORM.S.DIST + STDEVX.P | Z-score normalization | Customer spending deviation |
| SWITCH + PERCENTILEX.INC | RFM scoring | 1-5 customer banding |

**Conditional Logic:**

| **Function** | **Explanation** | **Example** |
| --- | --- | --- |
| **SWITCH(TRUE())** | Cleaner version of multiple nested IFs. | SWITCH(TRUE(), [Sales]>=50K, "A", [Sales]>=10K, "B", "C") |
| **IN** | Checks membership in a list. | [Region] IN {"East","West"} |

**Analytical Patterns:**

| **Pattern** | **Explanation** | **Example** |
| --- | --- | --- |
| **Running Total** | Adds all previous rows up to current date. | FILTER(ALLSELECTED(DimDate), DimDate[Date] <= MAX(DimDate[Date])) |
| **% of Parent** | Keeps parent filter only. | ALLEXCEPT(DimProduct, DimProduct[Category]) |
| **Active Customer (N days)** | Uses DATESINPERIOD to count recent activity. |  |
| **Pareto 80/20** | Combine RANKX, SUMX, and virtual tables to find top 20% contributors. |  |

**User Awareness & Security:**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| **USERPRINCIPALNAME / USERNAME** | Returns current logged-in user email or domain name. | IF(USERPRINCIPALNAME() = "alex@company.com", [Total Sales]) |
| **LOOKUPVALUE** | Similar to Excel VLOOKUP; fetches value from another table without relationship. | LOOKUPVALUE(DimRegion[Country], DimRegion[Region], "East") |

**Date/Time & Duration Logic:**

| **Function** | **Explanation** | **Example** |
| --- | --- | --- |
| **DATEDIFF()** | Calculates difference between two dates in chosen unit. | DATEDIFF([StartDate],[EndDate],MONTH) |
| **EDATE() / EOMONTH()** | Offset by N months or return month end date. | EOMONTH(TODAY(),-1) for previous month end. |