```
================================================================================
      ADVANCED DAX FUNCTIONS - COMPLETE SYNTAX & EXAMPLES GUIDE
        25+ Advanced Functions with Real Dataset Examples
================================================================================

This guide covers ADVANCED DAX Functions that require deeper understanding
Perfect for intermediate to advanced Power BI users

Dataset Used: Your 1000-row Indian e-commerce dataset
Additional Reference Tables included in examples

================================================================================
ADVANCED DAX FUNCTIONS - COMPLETE GUIDE (25+ Functions)
================================================================================

📌 NOTE: All examples use your existing dataset:
  • SalesData: 1000 orders
  • CustomerMaster: 500 customers
  • ProductMaster: 25 products
  • DateTable: 1000+ dates

================================================================================
1️⃣ LOOKUPVALUE() - LOOKUP VALUE FROM ANOTHER TABLE
================================================================================

SYNTAX:
  LOOKUPVALUE(<result_column>, <search_column1>, <search_value1>,
              [<search_column2>, <search_value2>], ...)

PARAMETERS:
  • <result_column>: Column to return value from
  • <search_column>: Column to search in
  • <search_value>: Value to find
  • Multiple search criteria allowed

CONCEPT:
  Similar to VLOOKUP in Excel but works with DAX relationships
  Must specify exact match criteria
  Returns single value (first match)
  Does NOT require explicit relationships

EXAMPLE 1: GET CUSTOMER NAME FROM ID (Calculated Column)
  Customer Name = LOOKUPVALUE(CustomerMaster[CustomerName],
    CustomerMaster[CustomerID], SalesData[CustomerID])

  Result:
    Order 10001 (CustomerID 2001) → "Rajesh Kumar"
    Order 10002 (CustomerID 2002) → "Priya Singh"
    Order 10003 (CustomerID 2010) → "Sakshi Kalra"

EXAMPLE 2: GET CUSTOMER EMAIL (Calculated Column)
  Customer Email = LOOKUPVALUE(CustomerMaster[Email],
    CustomerMaster[CustomerID], SalesData[CustomerID])

  Result:
    CustomerID 2001 → "rajesh.kumar@gmail.com"
    CustomerID 2002 → "priya.singh@outlook.com"

EXAMPLE 3: GET PRODUCT CATEGORY (Calculated Column)
  Product Category = LOOKUPVALUE(ProductMaster[Category],
    ProductMaster[ProductName], SalesData[Product])

  Result:
    "Laptop" → "Electronics"
```

```
    "T-Shirt" → "Clothing"

DIFFERENCE FROM RELATED():
  LOOKUPVALUE:
    ✓ Does NOT require relationship
    ✓ Can search multiple columns
    ✓ Slower for large tables
    ✓ Explicit match required

  RELATED:
    ✓ Requires relationship
    ✓ Single column lookup
    ✓ Faster performance
    ✓ Automatic context

WHEN TO USE:
  ✓ No relationship exists
  ✓ Need to search multiple columns
  ✓ Conditional lookups
  ✓ Complex matching logic
```

2️⃣ NESTED IF STATEMENT - MULTIPLE CONDITIONS

```
SYNTAX:
  IF(condition1, result1,
    IF(condition2, result2,
      IF(condition3, result3, default_result)))

CONCEPT:
  Each IF is nested inside the false_value of previous IF
  Evaluates top-to-bottom (stops at first TRUE)
  Can be slow with many levels
  SWITCH() is cleaner alternative

EXAMPLE 1: ORDER VALUE CLASSIFICATION (Calculated Column)
  Order Class = IF(SalesData[FinalAmount] >= 75000, "Premium",
    IF(SalesData[FinalAmount] >= 50000, "Gold",
    IF(SalesData[FinalAmount] >= 25000, "Silver",
    IF(SalesData[FinalAmount] >= 10000, "Bronze", "Basic"))))

  Result:
    ₹80,000 → "Premium"
    ₹60,000 → "Gold"
    ₹40,000 → "Silver"
    ₹15,000 → "Bronze"
    ₹8,000 → "Basic"

EXAMPLE 2: DELIVERY STATUS (Calculated Column)
  Delivery Status = IF(SalesData[Status] = "Completed", "Delivered",
    IF(SalesData[Status] = "Pending", "In Transit",
    IF(SalesData[Status] = "Cancelled", "Cancelled", "Unknown")))

EXAMPLE 3: DISCOUNT TIER (Calculated Column)
  Discount Tier = IF(SalesData[DiscountPercent] > 15, "VIP",
    IF(SalesData[DiscountPercent] > 10, "Premium",
    IF(SalesData[DiscountPercent] > 5, "Standard", "No Discount")))

WHEN TO USE:
  ✓ Multiple sequential conditions
  ✓ Categories with priority order
  ✓ Value ranges/brackets
```

```
  ✓ Complex if-then-else logic

BETTER ALTERNATIVE - SWITCH():
  Order Class = SWITCH(TRUE,
    SalesData[FinalAmount] >= 75000, "Premium",
    SalesData[FinalAmount] >= 50000, "Gold",
    SalesData[FinalAmount] >= 25000, "Silver",
    SalesData[FinalAmount] >= 10000, "Bronze",
    "Basic")

  Result: Same as nested IF but cleaner code!
```

3️⃣ SUM vs SUMX - DIFFERENCE & WHEN TO USE

```
CONCEPT:
  SUM: Direct column aggregation (pre-calculated values)
  SUMX: Row-by-row calculation then sum (calculated values)

SYNTAX:
  SUM(<column>)
  SUMX(<table>, <expression>)

COMPARISON TABLE:
```

Aspect	SUM()	SUMX()
Usage	Sum existing column	Sum calculated values
Performance	Faster	Slower
Formula	Direct column	Expression per row
Complexity	Simple	Complex calculations
Example	Total Amount	Qty × Price per row
Best For	Simple totals	Derived metrics

```
EXAMPLE 1: SIMPLE SUM (Use SUM)
  Total Amount = SUM(SalesData[Amount])

  Result: ₹33,900,000 (direct column sum)
  Performance: Fast ⚡

EXAMPLE 2: CALCULATE TOTAL WITH FORMULA (Use SUMX)
  Total After Discount = SUMX(SalesData,
    SalesData[Amount] * (1 - SalesData[DiscountPercent] / 100))

  Result: ₹32,400,000 (after applying discount percentage)
  Performance: Slower 🐢

EXAMPLE 3: TAX CALCULATION (Use SUMX)
  Total Tax = SUMX(SalesData,
    SalesData[NetAmount] * 0.18)

  Result: ₹6,084,000 (18% GST on net amount)

EXAMPLE 4: MARGIN CALCULATION (Use SUMX)
  Total Margin = SUMX(SalesData,
    (SalesData[FinalAmount] - SalesData[CostPrice]) * (1 + Profit_Rate))

  Result: Profit with margin applied

WHEN TO USE SUM:
```

```
  ✓ Direct column sum needed
  ✓ Performance critical
  ✓ Simple aggregation
  ✓ Amount, Quantity columns

WHEN TO USE SUMX:
  ✓ Need row-by-row calculation
  ✓ Formula per each row
  ✓ Derived/calculated metrics
  ✓ Weighted calculations
  ✓ Complex formulas needed

_____________________________________________________________________________

4️⃣ SPLIT COLUMN USING DAX - EXTRACT TEXT PARTS
=============================================================================

CONCEPT:
  DAX doesn't have native SPLIT function like Excel
  Use LEFT(), RIGHT(), MID(), SEARCH() combination
  Or use SUBSTITUTE() for pattern replacement

EXAMPLE 1: SPLIT EMAIL INTO USERNAME AND DOMAIN (Calculated Columns)

Username = LEFT(CustomerMaster[Email], SEARCH("@", CustomerMaster[Email]) - 1)

  Formula breaks down:
    • SEARCH("@", Email) = Position of @ symbol
    • Subtract 1 to exclude @
    • LEFT(Email, position) = Get everything before @

  Result:
    "rajesh.kumar@gmail.com" → "rajesh.kumar"
    "priya.singh@outlook.com" → "priya.singh"

Email Domain = RIGHT(CustomerMaster[Email],
  LEN(CustomerMaster[Email]) - SEARCH("@", CustomerMaster[Email]))

  Formula:
    • SEARCH("@", Email) = Position of @
    • LEN(Email) - Position = Length of domain part
    • RIGHT(Email, length) = Get last N characters

  Result:
    "rajesh.kumar@gmail.com" → "gmail.com"
    "priya.singh@outlook.com" → "outlook.com"

EXAMPLE 2: SPLIT DATE INTO COMPONENTS (Calculated Columns)

Order Year = LEFT(SalesData[OrderDate], 4)
  Result: "2023-01-15" → "2023"

Order Month = MID(SalesData[OrderDate], 6, 2)
  Result: "2023-01-15" → "01"

Order Day = RIGHT(SalesData[OrderDate], 2)
  Result: "2023-01-15" → "15"

EXAMPLE 3: EXTRACT CATEGORY FROM PRODUCT NAME (Calculated Column)

Product Type = IF(SEARCH("Laptop", SalesData[Product], 1) > 0, "Computer",
  IF(SEARCH("T-Shirt", SalesData[Product], 1) > 0, "Apparel",
  IF(SEARCH("Microwave", SalesData[Product], 1) > 0, "Appliance", "Other")))
```

WHEN TO USE:
  ✓ Parse email addresses
  ✓ Extract text segments
  ✓ Split date components
  ✓ Parse product codes
  ✓ Extract domains

5️⃣ CALENDAR vs CALENDARAUTO - DATE TABLE FUNCTIONS

SYNTAX:
  CALENDAR(<start_date>, <end_date>)
  CALENDARAUTO([fiscal_year_end_month])

CONCEPT:
  Both create date tables automatically
  CALENDAR: Manual control over date range
  CALENDARAUTO: Automatic date range from data

CALENDAR() - MANUAL DATE TABLE

Purpose: Create date table with specific date range

Example:
  My Calendar = CALENDAR(DATE(2023, 1, 1), DATE(2025, 12, 31))

  HOW TO CREATE IN POWER BI:
    1. Go to Modeling → New Table
    2. Enter: DateTable = CALENDAR(DATE(2023,1,1), DATE(2025,12,31))
    3. Click checkmark

  Result: Table with 1,096 rows (Jan 1, 2023 to Dec 31, 2025)

CALENDARAUTO() - AUTOMATIC DATE TABLE

Purpose: Create date table automatically from min/max dates in model

Example:
  My Calendar = CALENDARAUTO()

  USES:
    • Finds earliest date in your data (2023-01-01)
    • Finds latest date in your data (2025-12-07)
    • Creates complete date range between them

  Result: Table with dates from data range automatically

With Fiscal Year (Indian fiscal year April-March):
  My Calendar = CALENDARAUTO(3) -- 3 = March

  Result:
    FY2023: Apr 2022 - Mar 2023
    FY2024: Apr 2023 - Mar 2024
    FY2025: Apr 2024 - Mar 2025

COMPARISON:

Aspect	CALENDAR()	CALENDARAUTO()
Date Range Control	Manual specified Full control	Automatic from data Limited control

Future Dates	Can include future	Only actual dates
Flexibility	Very flexible	Less flexible
Ease	More manual work	Simpler
Fiscal Year	Manual setup	CALENDARAUTO(3)

WHEN TO USE CALENDAR:
  ✓ Need specific date range
  ✓ Include future dates (forecasting)
  ✓ Custom fiscal calendars
  ✓ Fixed date ranges required

WHEN TO USE CALENDARAUTO:
  ✓ Quick date table needed
  ✓ Dynamic date range from data
  ✓ Standard calendar enough
  ✓ Less setup required

---

6 COUNTBLANK - COUNT EMPTY CELLS

SYNTAX:
  COUNTBLANK(<column>)

PARAMETERS:
  • <column>: Column to count blanks in

EXAMPLE 1: COUNT MISSING EMAILS
  Missing Emails = COUNTBLANK(CustomerMaster[Email])

  Result: 0 (all customers have emails in your data)

EXAMPLE 2: COUNT MISSING DELIVERY DATES (Calculated Measure)
  Missing Delivery Dates = COUNTBLANK(SalesData[DeliveryDate])

  Result: Shows orders without delivery date

EXAMPLE 3: DATA QUALITY CHECK (Calculated Measure)
  Data Quality % = (1 - COUNTBLANK(SalesData[OrderID]) /
COUNTA(SalesData[OrderID])) * 100

  Result: What % of rows have OrderID (completeness)

WHEN TO USE:
  ✓ Data quality checks
  ✓ Find missing values
  ✓ Identify incomplete records
  ✓ Monitor data gaps

---

7 COUNTROWS - COUNT TABLE ROWS

SYNTAX:
  COUNTROWS(<table>)

PARAMETERS:
  • <table>: Table to count rows in

EXAMPLE 1: COUNT TOTAL ORDERS
  Total Orders = COUNTROWS(SalesData)

```
  Result: 1000 rows

EXAMPLE 2: COUNT FILTERED ORDERS (Calculated Measure)
  Completed Orders = COUNTROWS(FILTER(SalesData, SalesData[Status] =
"Completed"))

  Result: ~700 completed orders

EXAMPLE 3: COUNT UNIQUE CUSTOMERS
  Active Customers = COUNTROWS(FILTER(CustomerMaster, CustomerMaster[Status] =
"Active"))

  Result: ~450 active customers

DIFFERENCE FROM COUNT:
  COUNT: Counts numeric values in column
  COUNTROWS: Counts all rows in table (regardless of values)

WHEN TO USE:
  ✓ Count all rows in table
  ✓ Count filtered tables
  ✓ Row-level counts
  ✓ Table size verification
```

---

8 COUNTAX - COUNT NON-BLANK WITH EXPRESSION

```
SYNTAX:
  COUNTAX(<table>, <expression>)

PARAMETERS:
  • <table>: Table to iterate
  • <expression>: Expression that returns non-blank for rows to count

EXAMPLE 1: COUNT ORDERS WITH DISCOUNT
  Orders With Discount = COUNTAX(SalesData,
    IF(SalesData[DiscountPercent] > 0, 1, BLANK()))

  Result: ~350 orders with discount > 0

EXAMPLE 2: COUNT HIGH VALUE ORDERS
  High Value Orders = COUNTAX(SalesData,
    IF(SalesData[FinalAmount] > 50000, 1, BLANK()))

  Result: ~250 orders above ₹50,000

EXAMPLE 3: COUNT COMPLETED WITH AMOUNT > THRESHOLD
  Completed High Value = COUNTAX(SalesData,
    IF(AND(SalesData[Status] = "Completed",
           SalesData[FinalAmount] > 75000), 1, BLANK()))

WHEN TO USE:
  ✓ Conditional counting
  ✓ Complex count logic
  ✓ Expression-based counting
  ✓ Advanced filtering counts
```

---

9 ALL vs ALLSELECTED vs ALLEXCEPT - FILTER MANIPULATION

CONCEPT:
  All three remove filters, but in different ways
  Used in CALCULATE() to change filter context

SYNTAX:
  ALL(<column> or <table>)
  ALLSELECTED(<column> or <table>)
  ALLEXCEPT(<table>, <column1>, <column2>, ...)

COMPARISON TABLE:

Scenario	ALL()	ALLSELECTED()	ALLEXCEPT()
Remove Filters	All filters	Keep visual	All except named
Use	Grand totals	Visual subtotals	Specific filters
Context	No context	Visual context	Partial context
Performance	Fast	Fast	Fast
Complexity	Simple	Medium	Complex

EXAMPLE 1: ALL() - REMOVE ALL FILTERS

Grand Total (Regardless of Filters):
  Grand Total Sales = CALCULATE(SUM(SalesData[FinalAmount]), ALL(SalesData))

  Behavior:
    • Filter Region = North: Still shows ₹35,860,529
    • Filter Category = Electronics: Still shows ₹35,860,529
    • Filter Customer = Rajesh: Still shows ₹35,860,529

  Result: ALWAYS shows total of all 1000 orders

Percentage of Grand Total:
  % of Grand Total = DIVIDE(
    SUM(SalesData[FinalAmount]),
    CALCULATE(SUM(SalesData[FinalAmount]), ALL(SalesData)),
    0)

  Result:
    • North region = ₹5,973,000 / ₹35,860,529 = 16.7%
    • Electronics = ₹8,900,000 / ₹35,860,529 = 24.8%

EXAMPLE 2: ALLSELECTED() - CLEAR FILTERS RESPECT VISUAL

Visual Level Grand Total:
  Visual Total = CALCULATE(SUM(SalesData[FinalAmount]), ALLSELECTED())

  Behavior in Table Visual:
    • Table filtered to Electronics only
    • Visual Total = ₹8,900,000 (total of visible Electronics)
    • NOT the grand total of all categories

Row Level Percentage:
  % of Visual = DIVIDE(
    SUM(SalesData[FinalAmount]),
    CALCULATE(SUM(SalesData[FinalAmount]), ALLSELECTED()),
    0)

  Result:
    • If showing Electronics: Each product % of Electronics total
    • If showing North: Each product % of North total
    • Dynamic to what's shown

EXAMPLE 3: ALLEXCEPT() - REMOVE SPECIFIC FILTERS ONLY

Total by Region (Ignore Category Filter):
  Region Total = CALCULATE(SUM(SalesData[FinalAmount]),
    ALLEXCEPT(SalesData, SalesData[Region]))

  Behavior:
    • If filtered: Region=North, Category=Electronics
    • Result = Total of all Electronics in North
    • Keeps Region filter, removes Category filter

Total by Category (Ignore Date Filter):
  Category Total = CALCULATE(SUM(SalesData[FinalAmount]),
    ALLEXCEPT(SalesData, SalesData[Category]))

  Result:
    • Shows total for selected category across all dates
    • Removes date filters, keeps category filter

REAL USAGE EXAMPLE:

Imagine Power BI report with slicers: Region, Category, Date

Measure 1: Grand Total
  Grand Total = CALCULATE(SUM(SalesData[FinalAmount]), ALL(SalesData))
  Result: ₹35,860,529 (ignores all slicers)

Measure 2: Category Total (Ignores Region & Date)
  Category Total = CALCULATE(SUM(SalesData[FinalAmount]),
    ALLEXCEPT(SalesData, SalesData[Category]))
  Result: Total for selected category across all regions & dates

Measure 3: Visual Subtotal
  Visual Total = CALCULATE(SUM(SalesData[FinalAmount]), ALLSELECTED())
  Result: Total of filtered data shown in visual

Measure 4: % of Grand Total
  % Grand = DIVIDE(SUM(SalesData[FinalAmount]), Grand Total)

Measure 5: % of Category Total
  % Category = DIVIDE(SUM(SalesData[FinalAmount]), Category Total)

WHEN TO USE:

ALL():
  ✓ Grand totals needed
  ✓ Percentage of overall calculation
  ✓ Remove all context
  ✓ Benchmarking against total

ALLSELECTED():
  ✓ Visual-level totals
  ✓ Row percentages within visual
  ✓ Keep visual context
  ✓ Subtotals

ALLEXCEPT():
  ✓ Selective filter removal
  ✓ Keep specific dimensions
  ✓ Complex multi-dimensional
  ✓ Partial context clearing

🔟 MIN vs MINA vs MINX - FINDING MINIMUM

```
SYNTAX:
  MIN(<column>)
  MINA(<column>)          -- Counts empty values as 0
  MINX(<table>, <expr>)   -- Min of expression
```

COMPARISON TABLE:

Aspect	MIN()	MINA()	MINX()
Input	Column	Column	Expression
Blanks	Ignored	Treated as 0	Custom handling
Text	Ignored	Error	Custom handling
Numbers	Min of column	Min (including 0)	Min of expression
Performance	Fast	Fast	Slower
Use	Most common	Rare	Calculated min

```
EXAMPLE 1: SIMPLE MIN() - MINIMUM ORDER VALUE
  Min Order Value = MIN(SalesData[FinalAmount])

  Result: ₹118 (smallest single order)

EXAMPLE 2: MIN() WITH FILTER (Calculated Measure)
  Min Electronics Price = CALCULATE(MIN(SalesData[UnitPrice]),
    SalesData[Category] = "Electronics")

  Result: ₹10,000 (cheapest electronics item)

EXAMPLE 3: MINA() - INCLUDING ZEROS (Calculated Column)
  Min With Zeros = MINA(SalesData[DiscountPercent])

  Result: 0 (many orders have no discount)

EXAMPLE 4: MINX() - MIN OF EXPRESSION
  Min Discounted Amount = MINX(SalesData,
    SalesData[NetAmount] * (1 - SalesData[DiscountPercent] / 100))

  Result: Minimum amount after applying discount

WHEN TO USE:

MIN():
  ✓ Find minimum in numeric column
  ✓ Lowest price, smallest order
  ✓ Most common choice

MINA():
  ✓ Need to include 0/empty values
  ✓ Rare scenarios
  ✓ Data quality checks

MINX():
  ✓ Min of calculated expression
  ✓ Derived metric minimum
  ✓ Complex calculations
```
