1️⃣1️⃣ RELATED vs RELATEDTABLE - RELATIONSHIP NAVIGATION

```
SYNTAX:
  RELATED(<column>)
  RELATEDTABLE(<table>)

CONCEPT:
  RELATED: Get single value from related table
  RELATEDTABLE: Get entire related table

REQUIREMENTS:
  ✓ Relationship must exist between tables
  ✓ Must be in correct direction
  ✓ One-to-many relationship needed

EXAMPLE 1: RELATED() - GET CUSTOMER NAME (Calculated Column)

  Customer Name = RELATED(CustomerMaster[CustomerName])

  How it works:
    1. SalesData.CustomerID = 2001
    2. Find matching CustomerMaster.CustomerID = 2001
    3. Return CustomerMaster.CustomerName = "Rajesh Kumar"

  Result: Each order shows customer name
    Order 10001 → "Rajesh Kumar"
    Order 10002 → "Priya Singh"

EXAMPLE 2: RELATED() - GET CUSTOMER EMAIL (Calculated Column)

  Customer Email = RELATED(CustomerMaster[Email])

  Result: "rajesh.kumar@gmail.com"

EXAMPLE 3: RELATED() - GET PRODUCT CATEGORY (Calculated Column)

  Product Category = RELATED(ProductMaster[Category])

  Result:
    Laptop → "Electronics"
    T-Shirt → "Clothing"

EXAMPLE 4: RELATEDTABLE() - GET ALL RELATED ORDERS (Advanced)

  Customer Order Count = COUNTROWS(RELATEDTABLE(SalesData))

  How it works:
    1. From CustomerMaster table
    2. Find all SalesData rows where CustomerID matches
    3. Count them

  Result: In CustomerMaster, shows how many orders each customer made

EXAMPLE 5: RELATEDTABLE() - SUM OF CUSTOMER ORDERS (Advanced)

  Customer Total Spending = SUMX(RELATEDTABLE(SalesData),
    SalesData[FinalAmount])

  Result: In CustomerMaster, shows total amount spent by each customer
    Rajesh Kumar → ₹125,000
    Priya Singh → ₹98,500
```

```
RELATIONSHIP DIRECTION:

One-to-Many Example:
  CustomerMaster (One) ⟷ SalesData (Many)

  From SalesData perspective:
    ✓ RELATED(CustomerMaster[Name]) = Works!

  From CustomerMaster perspective:
    ✓ RELATEDTABLE(SalesData) = Works!

WHEN TO USE:

RELATED():
  ✓ Get single value from related table
  ✓ Common for lookups
  ✓ Add dimension attributes
  ✓ Simple value retrieval

RELATEDTABLE():
  ✓ Need entire related table
  ✓ Count related records
  ✓ Aggregate related data
  ✓ Advanced calculations

_____________________________________________________________________________

1️⃣2️⃣ DATESBETWEEN vs DATEINPERIOD - DATE RANGE FUNCTIONS
=============================================================================

SYNTAX:
  DATESBETWEEN(<dates>, <start_date>, <end_date>)
  DATEINPERIOD(<dates>, <date>, <number>, <interval>)

CONCEPT:
  Both return filtered date ranges
  Use in CALCULATE() to filter by date range
  Return table of dates matching criteria

EXAMPLE 1: DATESBETWEEN() - SPECIFIC DATE RANGE

Sales Last 3 Months (Calculated Measure):
  Last 3 Months Sales = CALCULATE(SUM(SalesData[FinalAmount]),
    DATESBETWEEN(DateTable[Date], DATE(2025, 9, 7), DATE(2025, 12, 7)))

  Result: Sum of Sep 7 to Dec 7, 2025 sales = ₹12,500,000

Sales in Q4 2025 (Calculated Measure):
  Q4 Sales = CALCULATE(SUM(SalesData[FinalAmount]),
    DATESBETWEEN(DateTable[Date], DATE(2025, 10, 1), DATE(2025, 12, 31)))

  Result: Oct-Dec sales = ₹8,900,000

EXAMPLE 2: DATEINPERIOD() - DYNAMIC RANGE

Last 30 Days Sales (Calculated Measure):
  Last 30 Days = CALCULATE(SUM(SalesData[FinalAmount]),
    DATEINPERIOD(DateTable[Date], TODAY(), -30, DAY))

  Result: Sales from 30 days ago to today

Last 3 Months (Calculated Measure):
  Last 3 Months = CALCULATE(SUM(SalesData[FinalAmount]),
    DATEINPERIOD(DateTable[Date], TODAY(), -3, MONTH))
```

```
  Result: Sales from 3 months ago to today

Last Year (Calculated Measure):
  Last Year = CALCULATE(SUM(SalesData[FinalAmount]),
    DATEINPERIOD(DateTable[Date], TODAY(), -1, YEAR))

COMPARISON:
```

Aspect	DATESBETWEEN()	DATEINPERIOD()
Range	Fixed dates	Dynamic offset
Flexibility	Static boundaries	Relative to TODAY
Use	Known date ranges	Rolling periods
Example	"Oct 1 to Dec 31"	"Last 90 days"
Automation	Manual update needed	Auto updates daily

```
WHEN TO USE:

DATESBETWEEN():
  ✓ Fixed date ranges
  ✓ Specific periods (Q4, FY2025)
  ✓ Historical comparisons
  ✓ Known start/end dates

DATEINPERIOD():
  ✓ Rolling periods
  ✓ "Last N days/months"
  ✓ Dynamic calculations
  ✓ Self-updating reports
```

1️⃣3️⃣ DATESMTD vs DATESQTD vs DATESYTD - PERIOD TO DATE

```
SYNTAX:
  DATESMTD(<dates>, [month_end])
  DATESQTD(<dates>, [quarter_end])
  DATESYTD(<dates>, [year_end])

CONCEPT:
  Return dates from period start to specified date (or TODAY)
  Used in CALCULATE() for cumulative calculations
  Perfect for dashboards tracking progress through period

EXAMPLE 1: MONTH-TO-DATE (MTD) SALES
  MTD Sales = CALCULATE(SUM(SalesData[FinalAmount]),
    DATESMTD(DateTable[Date]))

  As of Dec 7, 2025:
    Result: ₹2,500,000 (Dec 1-7 sales)

  If Dec 15:
    Result: ₹3,500,000 (Dec 1-15 sales)

  Usage: Track "sales so far this month"

EXAMPLE 2: QUARTER-TO-DATE (QTD) SALES
  QTD Sales = CALCULATE(SUM(SalesData[FinalAmount]),
    DATESQTD(DateTable[Date]))
```

```
  As of Dec 7, 2025 (Q4):
    Result: ₹8,500,000 (Oct-Dec sales)

  If in January (Q1):
    Result: ₹2,900,000 (Jan sales only)

  Usage: Track "sales so far this quarter"

EXAMPLE 3: YEAR-TO-DATE (YTD) SALES
  YTD Sales = CALCULATE(SUM(SalesData[FinalAmount]),
    DATESYTD(DateTable[Date]))

  As of Dec 7, 2025:
    Result: ₹35,000,000 (Jan-Dec 7 sales)

  If in June:
    Result: ₹17,500,000 (Jan-Jun sales)

  Usage: Track "sales so far this year"

COMPARISON:

┌──────────────────┬──────────────────────┬──────────────────────┬──────────────────┐
│ Period           │ DATESMTD()           │ DATESQTD()           │ DATESYTD()       │
├──────────────────┼──────────────────────┼──────────────────────┼──────────────────┤
│ Start            │ 1st of month         │ 1st of quarter       │ 1st of year      │
│ End              │ TODAY()              │ TODAY()              │ TODAY()          │
│ Recalcs          │ Monthly              │ Quarterly            │ Annually         │
│ Usage            │ Month progress       │ Quarter progress     │ Year progress    │
│ Period           │ 1-31 days            │ 90 days (approx)     │ 365 days         │
│ Resets           │ Monthly              │ Quarterly            │ Annually         │
└──────────────────┴──────────────────────┴──────────────────────┴──────────────────┘

REAL DASHBOARD USAGE:

KPI Card 1: YTD Progress
  Label: "₹35,000,000"
  Subtitle: "2025 Year-to-Date"
  Formula: DATESYTD(DateTable[Date])

KPI Card 2: MTD Progress
  Label: "₹2,500,000"
  Subtitle: "December-to-Date"
  Formula: DATESMTD(DateTable[Date])

KPI Card 3: vs Target
  YTD Target: ₹40,000,000
  YTD Actual: ₹35,000,000
  % Achieved: DIVIDE(YTD Sales, ₹40,000,000) = 87.5%

WHEN TO USE:
  ✓ Track period progress
  ✓ Executive dashboards
  ✓ Compare to targets
```

  ✓ Monitor KPIs
  ✓ Cumulative displays

1️⃣4️⃣ ENDOFMONTH, ENDOFQUARTER, ENDOFYEAR - PERIOD END DATES

```
SYNTAX:
  ENDOFMONTH(<dates>)
  ENDOFQUARTER(<dates>)
  ENDOFYEAR(<dates>)

CONCEPT:
  Return last date of period
  Used for comparisons and filtering
  Different from EOMONTH (simpler to use)

EXAMPLE 1: ENDOFMONTH() - LAST DAY OF MONTH

Get Last Day of Order Month (Calculated Column):
  Month End Date = ENDOFMONTH(SalesData[OrderDate])

  Result:
    "2023-01-15" → "2023-01-31"
    "2023-02-20" → "2023-02-28"
    "2025-12-07" → "2025-12-31"

Sales Up to Month End (Calculated Measure):
  MTD Complete = CALCULATE(SUM(SalesData[FinalAmount]),
    SalesData[OrderDate] <= ENDOFMONTH(DateTable[Date]))

EXAMPLE 2: ENDOFQUARTER() - LAST DAY OF QUARTER

Get Last Day of Quarter (Calculated Column):
  Quarter End Date = ENDOFQUARTER(SalesData[OrderDate])

  Result:
    "2023-01-15" (Q1) → "2023-03-31"
    "2023-05-20" (Q2) → "2023-06-30"
    "2023-10-07" (Q4) → "2023-12-31"

Sales Up to Quarter End (Calculated Measure):
  QTD Complete = CALCULATE(SUM(SalesData[FinalAmount]),
    SalesData[OrderDate] <= ENDOFQUARTER(DateTable[Date]))

EXAMPLE 3: ENDOFYEAR() - LAST DAY OF YEAR

Get Last Day of Year (Calculated Column):
  Year End Date = ENDOFYEAR(SalesData[OrderDate])

  Result:
    "2023-01-15" → "2023-12-31"
    "2024-06-20" → "2024-12-31"
    "2025-12-07" → "2025-12-31"

Sales Up to Year End (Calculated Measure):
  YTD Complete = CALCULATE(SUM(SalesData[FinalAmount]),
    SalesData[OrderDate] <= ENDOFYEAR(DateTable[Date]))

REAL USAGE:

Fiscal Year Closure:
  • Month End: Close monthly financials on last day
```

```
  • Quarter End: Q1, Q2, Q3, Q4 reports on these dates
  • Year End: Annual close on Dec 31

WHEN TO USE:
  ✓ Period boundaries
  ✓ Closure deadlines
  ✓ Period-end comparisons
  ✓ Date cutoffs

──────────────────────────────────────────────────────────────────────────────

1️⃣5️⃣ TOTALMTD, TOTALQTD, TOTALYTD - CUMULATIVE TOTALS
==============================================================================

SYNTAX:
  TOTALMTD(<measure>, <dates>, [year_end])
  TOTALQTD(<measure>, <dates>, [quarter_end])
  TOTALYTD(<measure>, <dates>, [year_end])

CONCEPT:
  Shortcuts for CALCULATE + period + measure
  Return cumulative total from period start to date
  Automatically includes all dates up to current

EXAMPLE 1: MONTH-TO-DATE TOTAL

Create Base Measure First:
  Daily Sales = SUM(SalesData[FinalAmount])

Then Create MTD Measure:
  MTD Sales = TOTALMTD(Daily Sales, DateTable[Date])

  Result:
    Dec 1: ₹500,000
    Dec 7: ₹2,500,000
    Dec 15: ₹4,200,000
    Dec 31: ₹9,500,000

EXAMPLE 2: QUARTER-TO-DATE TOTAL

QTD Sales = TOTALQTD(Daily Sales, DateTable[Date])

  Result (Q4 2025):
    Oct 1: ₹2,000,000
    Nov 15: ₹6,500,000
    Dec 7: ₹8,500,000
    Dec 31: ₹9,800,000

EXAMPLE 3: YEAR-TO-DATE TOTAL

YTD Sales = TOTALYTD(Daily Sales, DateTable[Date])

  Result:
    Jan 31: ₹1,500,000
    Jun 30: ₹17,500,000
    Dec 7: ₹35,000,000
    Dec 31: ₹35,860,529

SHORTCUT COMPARISON:

┌──────────────────────────────────────┬──────────────────────────────────────
┐
│ Traditional Method                   │ Shortcut Method
│
```

```
┌──────────────────────────────────────────┬──────────────────────────────────────────┐
│ MTD = CALCULATE(Daily Sales,             │ MTD = TOTALMTD(Daily Sales,              │
│       DATESMTD(DateTable[Date]))         │        DateTable[Date])                  │
│ (Longer, more explicit)                  │ (Shorter, simpler)                       │
│                                          │                                          │
│ QTD = CALCULATE(Daily Sales,             │ QTD = TOTALQTD(Daily Sales,              │
│       DATESQTD(DateTable[Date]))         │        DateTable[Date])                  │
│                                          │                                          │
│ YTD = CALCULATE(Daily Sales,             │ YTD = TOTALYTD(Daily Sales,              │
│       DATESYTD(DateTable[Date]))         │        DateTable[Date])                  │
└──────────────────────────────────────────┴──────────────────────────────────────────┘
```

WHEN TO USE:
- ✓ Simpler syntax than CALCULATE + DATES
- ✓ Running totals needed
- ✓ Progress tracking
- ✓ Cumulative charts

1️⃣6️⃣ UNION, INTERSECT, EXCEPT - TABLE SET OPERATIONS

SYNTAX:
- UNION(<table1>, <table2>, [<table3>], ...)
- INTERSECT(<table1>, <table2>)
- EXCEPT(<table1>, <table2>)

CONCEPT:
- UNION: Combine multiple tables (all rows from both)
- INTERSECT: Common rows in both tables
- EXCEPT: Rows in first table but not in second

REQUIREMENTS:
- ✓ Tables must have same column structure
- ✓ Column names must match
- ✓ Same data types

EXAMPLE 1: UNION() - COMBINE TABLES

Combine Online and Offline Orders:

```
All Orders = UNION(OnlineOrders, OfflineOrders)
```

Result: All rows from both tables combined

Combine Multiple Regions:

```
North South Orders = UNION(
  FILTER(SalesData, SalesData[Region] = "North"),
  FILTER(SalesData, SalesData[Region] = "South"))
```

Result: Only North and South region orders

EXAMPLE 2: INTERSECT() - COMMON ROWS

```
Customers with Both Online and Offline Orders:
  Multi Channel Customers = INTERSECT(
    DISTINCT(OnlineOrders[CustomerID]),
    DISTINCT(OfflineOrders[CustomerID]))

  Result: CustomerIDs who ordered both online and offline

High Value Customers in Q1 and Q2:
  Multi Quarter VIP = INTERSECT(
    FILTER(SalesData, QUARTER(SalesData[OrderDate]) = 1,
           SalesData[FinalAmount] > 50000),
    FILTER(SalesData, QUARTER(SalesData[OrderDate]) = 2,
           SalesData[FinalAmount] > 50000))

EXAMPLE 3: EXCEPT() - DIFFERENCE

Customers with Online Orders but Not Offline:
  Online Only = EXCEPT(
    DISTINCT(OnlineOrders[CustomerID]),
    DISTINCT(OfflineOrders[CustomerID]))

  Result: Customers who only ordered online

Products Sold in Q1 but Not Q2:
  Q1 Only Products = EXCEPT(
    FILTER(SalesData, QUARTER(SalesData[OrderDate]) = 1),
    FILTER(SalesData, QUARTER(SalesData[OrderDate]) = 2))

WHEN TO USE:

UNION():
  ✓ Combine multiple tables
  ✓ Add all rows from sources
  ✓ Multi-source datasets
  ✓ Concatenate tables

INTERSECT():
  ✓ Find common elements
  ✓ Overlap analysis
  ✓ Cross-channel analysis
  ✓ Multi-period customers

EXCEPT():
  ✓ Find differences
  ✓ Exclusive segments
  ✓ What's missing analysis
  ✓ Churn detection
```

---

17 COALESCE - RETURN FIRST NON-BLANK

```
SYNTAX:
  COALESCE(<expression1>, <expression2>, [<expression3>], ...)

CONCEPT:
  Returns first non-blank value from list
  Useful for handling null/empty values
  Short-circuits (stops at first non-blank)

EXAMPLE 1: CUSTOMER NAME FALLBACK
```

```
Show Name or ID if Name Missing (Calculated Column):
  Customer Display = COALESCE(CustomerMaster[CustomerName],
    "Customer_" & CustomerMaster[CustomerID])

  Result:
    If CustomerName exists: "Rajesh Kumar"
    If blank: "Customer_2001"

EXAMPLE 2: PRODUCT DESCRIPTION HIERARCHY

Show Description or Category or Default (Calculated Column):
  Product Display = COALESCE(SalesData[ProductDescription],
    SalesData[Category],
    "Uncategorized")

  Result:
    If description exists: "Premium Laptop"
    If blank, use category: "Electronics"
    If both blank: "Uncategorized"

EXAMPLE 3: ADDRESS FIELD PRIORITY

Show Street or City or Country (Calculated Column):
  Address = COALESCE(SalesData[StreetAddress],
    SalesData[City],
    SalesData[State],
    "No Address")

WHEN TO USE:
  ✓ Handle missing values
  ✓ Fallback values needed
  ✓ Priority-based defaults
  ✓ Data quality issues
```

1️⃣8️⃣ ALL RANK FUNCTIONS - RANKING & ORDERING

```
SYNTAX:
  RANK.EQ(<value>, <array>, [order])    -- Tie same rank
  RANK.AVG(<value>, <array>, [order])   -- Average rank for ties
  RANKX(<table>, <expression>, [value], [order], [ties])

CONCEPT:
  RANK.EQ/AVG: Built-in ranking
  RANKX: Advanced row-by-row ranking

EXAMPLE 1: RANK.EQ() - SIMPLE RANKING

Rank Orders by Value (Calculated Column):
  Order Rank = RANK.EQ(SalesData[FinalAmount],
    SalesData[FinalAmount])

  Result:
    ₹529,000 → Rank 1
    ₹510,000 → Rank 2
    ₹500,000 → Rank 3
    ₹118 → Rank 1000

EXAMPLE 2: RANKX() - ADVANCED RANKING

Rank Customers by Total Spending (Calculated Column in CustomerMaster):
  Customer Rank = RANKX(
```

```
    ALL(CustomerMaster),
    SUMX(RELATEDTABLE(SalesData), SalesData[FinalAmount]),
    SUMX(RELATEDTABLE(SalesData), SalesData[FinalAmount]),
    0)  -- 0 = descending (highest first)

  Result:
    Top spender (₹125,000) → Rank 1
    2nd spender (₹110,000) → Rank 2

EXAMPLE 3: TOP PRODUCTS BY SALES

Rank Products by Total Revenue (Calculated Column):
  Product Sales Rank = RANKX(ALL(SalesData[Product]),
    CALCULATE(SUM(SalesData[FinalAmount])),
    ,,0)  -- Descending

  Result:
    Laptop sales highest → Rank 1
    T-Shirt → Rank 2

WHEN TO USE:
  ✓ Find top/bottom items
  ✓ Competitive ranking
  ✓ Performance tiers
  ✓ Leaderboards
```

---

19 TOP N FUNCTIONS - SELECT TOP N ITEMS

```
SYNTAX:
  TOPN(<n>, <table>, <order_by_expression>, [<order>])

CONCEPT:
  Returns top N rows from table
  Ordered by expression
  0 = ascending, 1 = descending

EXAMPLE 1: TOP 10 ORDERS BY VALUE

Top 10 Orders (Calculated Measure):
  Top 10 Orders Table = TOPN(10, SalesData, SalesData[FinalAmount], 0)

  Result: 10 highest value orders

EXAMPLE 2: BOTTOM 5 PRODUCTS BY SALES

Bottom 5 Products (Calculated):
  Bottom 5 = TOPN(5, SalesData, SalesData[FinalAmount], 1)  -- 1 = ascending

  Result: 5 lowest selling products

EXAMPLE 3: TOP 3 CUSTOMERS BY SPENDING

Top 3 Customers:
  Top 3 = TOPN(3, CustomerMaster,
    SUMX(RELATEDTABLE(SalesData), SalesData[FinalAmount]), 0)

  Result:
    1. Rajesh Kumar - ₹125,000
    2. Priya Singh - ₹115,000
    3. Neha Gupta - ₹105,000
```

```
WHEN TO USE:
  ✓ Get top N items
  ✓ Pareto analysis
  ✓ Best sellers
  ✓ Top customers

─────────────────────────────────────────────────────────────────────────────

2️⃣0️⃣ EARLIER - REFERENCE PREVIOUS ROW CONTEXT
================================================================================

SYNTAX:
  EARLIER(<column>)

CONCEPT:
  Returns value from outer row context
  Used in nested calculations
  Complex but powerful

EXAMPLE 1: ROW-BY-ROW COMPARISON

Compare to Previous Customer Order (Calculated Column):
  Previous Customer Spend = LOOKUPVALUE(CustomerMaster[TotalSpend],
    CustomerMaster[CustomerID], EARLIER(SalesData[CustomerID]))

  Allows comparing current row to context

EXAMPLE 2: CUMULATIVE CALCULATION

Cumulative Customer Spending (Calculated Column):
  Cumulative Spend = SUMX(
    FILTER(SalesData,
      SalesData[CustomerID] = EARLIER(SalesData[CustomerID]) &&
      SalesData[OrderDate] <= EARLIER(SalesData[OrderDate])),
    SalesData[FinalAmount])

WHEN TO USE:
  ✓ Advanced row contexts
  ✓ Previous row references
  ✓ Complex nested calculations
  ✓ Rare, expert level

─────────────────────────────────────────────────────────────────────────────

2️⃣1️⃣ SELECTCOLUMNS vs ADDCOLUMNS - COLUMN SELECTION
================================================================================

SYNTAX:
  SELECTCOLUMNS(<table>, <name>, <expression>, ...)
  ADDCOLUMNS(<table>, <name>, <expression>, ...)

CONCEPT:
  SELECTCOLUMNS: Return only specified columns
  ADDCOLUMNS: Add new columns to existing table

EXAMPLE 1: SELECTCOLUMNS() - SUBSET OF COLUMNS

Get Only Order ID and Amount (Calculated Table):
  Order Summary = SELECTCOLUMNS(SalesData,
    "Order ID", SalesData[OrderID],
    "Customer ID", SalesData[CustomerID],
    "Amount", SalesData[FinalAmount])

  Result: Table with only 3 columns (OrderID, CustomerID, Amount)
```

EXAMPLE 2: ADDCOLUMNS() - ADD CALCULATED COLUMNS

Add Profit Margin to Sales (Calculated Table):

```
  Sales With Margin = ADDCOLUMNS(SalesData,
    "Profit Margin %", (SalesData[NetAmount] - SalesData[CostPrice]) /
SalesData[NetAmount] * 100,
    "Tax Amount", SalesData[NetAmount] * 0.18)
```

  Result: Original columns plus 2 new calculated columns

COMPARISON:

Aspect	SELECTCOLUMNS()	ADDCOLUMNS()
Input Columns	Select subset	Keep all
Output	Only selected columns	All + new columns
Use	Reduce data	Enhance data
Performance	Smaller result	Larger result
Example	Summary extract	Add calculations

WHEN TO USE:

SELECTCOLUMNS():
  ✓ Need only specific columns
  ✓ Reduce data size
  ✓ Create views
  ✓ Extract subsets

ADDCOLUMNS():
  ✓ Add calculated columns
  ✓ Enrich existing data
  ✓ Add derived metrics
  ✓ Extend table

2️⃣2️⃣ ALL FILTER DAX FUNCTIONS - COMPREHENSIVE LIST

FILTER FUNCTIONS SUMMARY:

1. FILTER(<table>, <condition>)
   Purpose: Get filtered table
   Usage: Filter rows matching condition

2. CALCULATE(<measure>, <filter>)
   Purpose: Measure with filter
   Usage: Change context of calculation

3. ALL(<table/column>)
   Purpose: Remove all filters
   Usage: Grand totals

4. ALLEXCEPT(<table>, <columns>)
   Purpose: Remove specific filters
   Usage: Keep some filters only

5. ALLSELECTED(<table/column>)
   Purpose: Clear but respect visual
   Usage: Visual-level totals

```
6. VALUES(<column>)
   Purpose: Get filtered distinct values
   Usage: See what values are selected

7. HASONEVALUE(<column>)
   Purpose: Check single filter
   Usage: Conditional display

8. ISINSCOPE(<column>)
   Purpose: Check if in visual
   Usage: Determine visual context

9. ISFILTERED(<column>)
   Purpose: Check if filtered
   Usage: See if filter applied

10. ISCROSSFILTERED(<column>)
    Purpose: Check cross-filter
    Usage: See if affected by other table

EXAMPLE: COMBINATION FILTER USAGE

Create Measures for Dashboard:

Total Sales = SUM(SalesData[FinalAmount])

Grand Total = CALCULATE(SUM(SalesData[FinalAmount]), ALL(SalesData))

Region Total = CALCULATE(SUM(SalesData[FinalAmount]),
  ALLEXCEPT(SalesData, SalesData[Region]))

% of Grand Total = DIVIDE(Total Sales, Grand Total)

% of Region = DIVIDE(Total Sales, Region Total)

Regional Distribution = VALUES(SalesData[Region])

Show Only If Single Region Selected = IF(HASONEVALUE(SalesData[Region]),
  "Selected: " & VALUES(SalesData[Region]), "Multiple Regions")
```

2️⃣3️⃣ DAX VARIABLES - VAR & RETURN

```
SYNTAX:
  VAR <variable_name> = <expression>
  RETURN <final_expression_using_variable>

CONCEPT:
  VAR: Define a variable to reuse
  RETURN: Use the variable in calculation
  Improves readability and performance

EXAMPLE 1: SIMPLE VARIABLE

Sales with Tax Multiplier (Calculated Measure):
  Sales Adjusted =
    VAR TaxRate = 0.18
    VAR BaseSales = SUM(SalesData[NetAmount])
    RETURN BaseSales * (1 + TaxRate)

  Result: ₹35,860,529 (with 18% tax applied)
```

```
EXAMPLE 2: FILTER VARIABLE

Sales for Premium Orders (Calculated Measure):
  Premium Sales =
    VAR PremiumThreshold = 50000
    VAR PremiumOrders = FILTER(SalesData, SalesData[FinalAmount] >
PremiumThreshold)
    RETURN SUMX(PremiumOrders, SalesData[FinalAmount])

  Result: ₹8,900,000 (sum of orders > ₹50,000)

EXAMPLE 3: MULTIPLE VARIABLES

Complex Profit Calculation:
  Profit Analysis =
    VAR BaseSales = SUM(SalesData[NetAmount])
    VAR TotalDiscount = SUM(SalesData[Amount]) - SUM(SalesData[NetAmount])
    VAR TotalTax = BaseSales * 0.18
    VAR Costs = BaseSales * 0.30  -- Assume 30% cost
    RETURN (BaseSales - Costs - TotalDiscount - TotalTax)

  Result: Net profit after all deductions

BENEFITS OF VARIABLES:
  ✓ Cleaner code (readable)
  ✓ Reuse calculations (avoid duplicating)
  ✓ Better performance (calculated once)
  ✓ Easier debugging (named variables)
  ✓ Maintainable (change in one place)

EXAMPLE 4: VARIABLE IN COMPLEX LOGIC

Customer Segment Classification (Calculated Column):
  Segment =
    VAR TotalSpend = SUMX(RELATEDTABLE(SalesData), SalesData[FinalAmount])
    VAR OrderCount = COUNTROWS(RELATEDTABLE(SalesData))
    VAR AvgOrder = DIVIDE(TotalSpend, OrderCount)
    RETURN IF(TotalSpend > 100000, "VIP",
      IF(TotalSpend > 50000, "Premium",
      IF(OrderCount > 10, "Loyal", "Regular")))

WHEN TO USE VARIABLES:
  ✓ Complex measures
  ✓ Multiple calculations
  ✓ Reusable values
  ✓ Improve readability
  ✓ Performance optimization
```

2️⃣4️⃣ ISFILTERED & ISCROSSFILTERED - FILTER DETECTION

```
SYNTAX:
  ISFILTERED(<column>)
  ISCROSSFILTERED(<column>)

CONCEPT:
  ISFILTERED: Check if column has direct filter
  ISCROSSFILTERED: Check if affected by filter in related table

EXAMPLE 1: ISFILTERED() - CHECK DIRECT FILTER

Show Message if Region Filtered (Calculated Measure):
```

```
  Region Filter Status = IF(ISFILTERED(SalesData[Region]),
    "Region Selected: " & VALUES(SalesData[Region]),
    "All Regions Shown")

  Behavior:
    • If Region filter applied: "Region Selected: North"
    • If no Region filter: "All Regions Shown"

EXAMPLE 2: ISCROSSFILTERED() - CHECK RELATED FILTER

Check if CustomerID Filter Applied (Calculated Measure):
  Customer Filter Status = IF(ISCROSSFILTERED(SalesData[CustomerID]),
    "Customer Filter Active",
    "No Customer Filter")

EXAMPLE 3: CONDITIONAL DISPLAY

Show Detail Only When Filtered (Calculated Measure):
  Detailed View = IF(ISFILTERED(SalesData[Region]),
    SUM(SalesData[FinalAmount]),
    BLANK())

  Result:
    • When Region selected: Shows sales
    • When no filter: Shows nothing (BLANK)

WHEN TO USE:
  ✓ Conditional measure logic
  ✓ Hide detail until filtered
  ✓ Validation messages
  ✓ Smart dashboards
```

---

25 SUMMARIZE, SUMMARIZECOLUMNS, SUMMARYTABLES - DATA AGGREGATION

```
SYNTAX:
  SUMMARIZE(<table>, <groupby_col>, "Label", <agg_expr>, ...)
  SUMMARIZECOLUMNS(<groupby_col>, <filter>, "Label", <agg_expr>, ...)
  (SUMMARYTABLES is deprecated, avoid using)

CONCEPT:
  Create summary tables with grouping and aggregation
  Custom table creation with multiple metrics
  Advanced grouping and filtering

EXAMPLE 1: SUMMARIZE() - GROUP BY REGION

Summary by Region (Calculated Table):
  Region Sales = SUMMARIZE(SalesData,
    SalesData[Region],
    "Total Sales", SUM(SalesData[FinalAmount]),
    "Order Count", COUNTA(SalesData[OrderID]),
    "Avg Order", AVERAGE(SalesData[FinalAmount]))

  Result:
    Region      | Total Sales  | Order Count | Avg Order
    North       | ₹5,973,000   | 167         | ₹35,750
    South       | ₹6,100,000   | 180         | ₹33,890
    East        | ₹5,200,000   | 150         | ₹34,670
    West        | ₹7,400,000   | 195         | ₹37,950
    Central     | ₹3,900,000   | 115         | ₹33,910
    Northeast   | ₹7,288,529   | 193         | ₹37,750
```

EXAMPLE 2: SUMMARIZE() - MULTI-LEVEL GROUPING

Summary by Region and Category (Calculated Table):

```
Region Category Sales = SUMMARIZE(SalesData,
  SalesData[Region],
  SalesData[Category],
  "Sales", SUM(SalesData[FinalAmount]),
  "Quantity", SUM(SalesData[Quantity]))
```

Result: 6 regions × 5 categories = 30 rows matrix

EXAMPLE 3: SUMMARIZECOLUMNS() - WITH FILTERS

Top Regions Only (Calculated Table):

```
Top Regions = SUMMARIZECOLUMNS(
  SalesData[Region],
  FILTER(ALL(SalesData), SalesData[Region] <> "Northeast"),
  "Total Sales", SUM(SalesData[FinalAmount]))
```

Result: Summary excluding Northeast region

WHEN TO USE:

SUMMARIZE():
- ✓ Group by one or more columns
- ✓ Create custom summary tables
- ✓ Multi-level aggregation
- ✓ Complex reports

SUMMARIZECOLUMNS():
- ✓ Need filtering in summary
- ✓ More advanced scenarios
- ✓ Cross-filtering needed

SUMMARY TABLE - QUICK REFERENCE

Advanced Function	Syntax Example	Main Use
LOOKUPVALUE()	LOOKUPVALUE(col,..	Lookup without rel
Nested IF	IF(A, B, IF(C, D))	Multiple conditions
SUM vs SUMX	SUM() vs SUMX(,expr)	Direct vs formula
Split Column	LEFT(), RIGHT(), MID	Text extraction
CALENDAR vs AUTO	CALENDAR() vs AUTO()	Date table creation
COUNTBLANK	COUNTBLANK(col)	Count empty cells
COUNTROWS	COUNTROWS(table)	Count all rows
COUNTAX	COUNTAX(t, expr)	Count with formula
ALL/ALLSELECTED	ALL(), ALLSELECTED()	Filter removal
MIN vs MINX	MIN() vs MINX()	Direct vs formula
RELATED	RELATED(col)	Lookup value
RELATEDTABLE	RELATEDTABLE(t)	Get related rows
DATESBETWEEN	DATESBETWEEN(d,s,e)	Fixed date range
DATEINPERIOD	DATEINPERIOD(d,d,-30)	Rolling period
DATESMTD/QTD/YTD	DATESMTD(dates)	Period to date
ENDOFMONTH/etc	ENDOFMONTH(dates)	Period end dates
TOTAL MTD/QTD/YTD	TOTALMTD(m, dates)	Cumulative totals
UNION/INTERSECT	UNION(t1, t2)	Set operations
COALESCE	COALESCE(a, b, c)	First non-blank
RANK Functions	RANKX(t, expr)	Ranking items
TOPN	TOPN(5, table, expr)	Get top N items
EARLIER	EARLIER(col)	Prev row context

SELECTCOLUMNS	SELECTCOLUMNS(t,..)	Select columns
ADDCOLUMNS	ADDCOLUMNS(t, ..)	Add columns
Filter Functions	FILTER(), CALCULATE	Filter rows/context
VAR & RETURN	VAR x = .. RETURN	Variables
ISFILTERED	ISFILTERED(col)	Check filter status
SUMMARIZE	SUMMARIZE(t, col, ..)	Group & aggregate

```
================================================================================
NEXT STEPS FOR LEARNING
================================================================================

Suggested Learning Order:

Beginner:
  1. LOOKUPVALUE & RELATED (Relationships)
  2. Nested IF (Conditional logic)
  3. SUM vs SUMX (Aggregation difference)
  4. Split Column (Text manipulation)
  5. CALENDAR (Date tables)

Intermediate:
  6. COUNTBLANK, COUNTROWS, COUNTAX (Counting)
  7. ALL/ALLSELECTED (Filter removal)
  8. DATESBETWEEN, DATEINPERIOD (Date ranges)
  9. DATESMTD/QTD/YTD (Period calculations)
  10. TOTALYTD (Cumulative)

Advanced:
  11. UNION/INTERSECT/EXCEPT (Set operations)
  12. RANKX, TOPN (Ranking)
  13. VAR & RETURN (Variables)
  14. SUMMARIZE (Data aggregation)
  15. ISFILTERED (Filter detection)

================================================================================
PRACTICE EXERCISES
================================================================================

Exercise 1: Lookup Exercise
  Create LOOKUPVALUE formula to get customer phone from SalesData

Exercise 2: Conditional Logic
  Create nested IF to classify orders into 5 tiers by amount

Exercise 3: Date Range Analysis
  Create measures for:
    • Sales in last 30 days (DATEINPERIOD)
    • Year-to-date sales (DATEYTD)
    • Q4 2025 sales (DATESBETWEEN)

Exercise 4: Ranking
  Create measure to show top 5 customers by total spending

Exercise 5: Summary Table
  Create SUMMARIZE table showing sales by Region and Status

Exercise 6: Complex Variable Formula
  Use VAR to calculate profit with multiple adjustments

================================================================================
KEY TAKEAWAYS
================================================================================
```

✓ LOOKUPVALUE: Powerful alternative to RELATED without relationships
✓ Nested IF: Multiple conditions (use SWITCH for cleaner code)
✓ SUM vs SUMX: Choose based on whether you need formula or direct column
✓ CALENDAR/CALENDARAUTO: Different ways to create date tables
✓ Filter Functions: ALL, ALLSELECTED, ALLEXCEPT for context manipulation
✓ Ranking & Sorting: RANKX, TOPN for analysis
✓ Variables (VAR): Makes code cleaner and faster
✓ SUMMARIZE: Create custom summary tables
✓ Set Operations: UNION, INTERSECT, EXCEPT for complex logic
✓ Filter Detection: ISFILTERED to check filter status

All examples use your real dataset - practice with actual numbers!

==========