```
================================================================================
CATEGORY 5: LOGICAL ITERATORS (8 Functions)
================================================================================
```

4️⃣1️⃣ SUMX() - SUM WITH EXPRESSION PER ROW

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

PARAMETERS:
  • <table>: Table to iterate
  • <expression>: Expression to calculate for each row

EXAMPLE FROM YOUR DATASET:

Total Revenue (Alternative to SUM):
  Total Revenue SUMX = SUMX(SalesData, SalesData[Quantity] *
SalesData[UnitPrice])

  Result: ₹33,900,000 (before tax and discount)

Total With Discount Applied:
  Total After Discount = SUMX(SalesData,
    SalesData[Quantity] * SalesData[UnitPrice] * (1 -
SalesData[DiscountPercent]/100))

  Result: ₹33,000,000 (discount applied)

Total Tax Collected:
  Total Tax = SUMX(SalesData,
    SalesData[NetAmount] * 0.18)

  Result: ₹6,084,000 (18% GST)

REAL USAGE:
  • Calculate with formula per row
  • Complex aggregations
  • Weighted calculations
  • Alternative to SUM when formula needed
```

4️⃣2️⃣ AVERAGEX() - AVERAGE WITH EXPRESSION PER ROW

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

PARAMETERS:
  • <table>: Table to iterate
  • <expression>: Expression to calculate for each row

EXAMPLE FROM YOUR DATASET:

Average Total Value (Quantity × Price):
  Average Item Value = AVERAGEX(SalesData,
    SalesData[Quantity] * SalesData[UnitPrice])

  Result: ₹33,900 (average quantity×price per row)

Average After Discount:
  Average After Discount = AVERAGEX(SalesData,
    SalesData[NetAmount])
  Result: ₹32,450
```

REAL USAGE:
- Average of calculated values
- Complex averages
- Per-row calculations

4️⃣3️⃣ MAXX() - MAX WITH EXPRESSION

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

PARAMETERS:
- <table>: Table to iterate
- <expression>: Expression to calculate for each row

EXAMPLE FROM YOUR DATASET:

Maximum of Quantity × Price:
```
  Max Item Value = MAXX(SalesData,
    SalesData[Quantity] * SalesData[UnitPrice])

  Result: ₹500,000 (highest quantity×price combination)
```

REAL USAGE:
- Max of calculated values
- Peak calculations
- Performance tracking

4️⃣4️⃣ MINX() - MIN WITH EXPRESSION

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

PARAMETERS:
- <table>: Table to iterate
- <expression>: Expression to calculate for each row

EXAMPLE FROM YOUR DATASET:

Minimum of Quantity × Price:
```
  Min Item Value = MINX(SalesData,
    SalesData[Quantity] * SalesData[UnitPrice])

  Result: ₹100 (lowest quantity×price combination)
```

REAL USAGE:
- Min of calculated values
- Baseline calculations

4️⃣5️⃣ COUNTX() - COUNT WITH CONDITION

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

PARAMETERS:
- <table>: Table to iterate
- <expression>: Expression that returns TRUE/FALSE

EXAMPLE FROM YOUR DATASET:

Count Orders Above ₹50,000:

```
  High Value Count = COUNTX(SalesData, SalesData[FinalAmount] > 50000)

  Result: ~250 orders above ₹50,000
```

Count Electronics Orders:

```
  Electronics Count = COUNTX(SalesData, SalesData[Category] = "Electronics")

  Result: ~200 electronics orders
```

Count Non-Completed Orders:

```
  Pending Count = COUNTX(SalesData, SalesData[Status] <> "Completed")

  Result: ~300 pending/cancelled/returned orders
```

REAL USAGE:
- Conditional counting
- Category counts
- Status counts

4️⃣6️⃣ GENERATESERIES() - GENERATE NUMBER SERIES

SYNTAX:

```
  GENERATESERIES(<start>, <end>, [<step>])
```

PARAMETERS:
- <start>: Starting number
- <end>: Ending number
- <step>: Increment (optional, default 1)

EXAMPLE (Rarely used in Sales Analytics):

```
  Series = GENERATESERIES(1, 10, 1)

  Result: Table with 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

REAL USAGE:
- Generate sequences
- Create calendars
- Advanced scenarios

4️⃣7️⃣ GROUPBY() - GROUP DATA

SYNTAX:

```
  GROUPBY(<table>, <groupby_column>, ...)
```

PARAMETERS:
- <table>: Table to group
- <groupby_column>: Column(s) to group by

EXAMPLE FROM YOUR DATASET:

```
Group by Category:
  Grouped = GROUPBY(SalesData, SalesData[Category])

  Result: Table with unique categories

REAL USAGE:
  • Advanced grouping
  • Complex aggregations
  • Rare in typical analytics
```

4️⃣8️⃣ SUMMARIZE() - CREATE SUMMARY TABLE

```
SYNTAX:
  SUMMARIZE(<table>, <groupby_col1>, [<groupby_col2>], "Label",
<expression>, ...)

PARAMETERS:
  • <table>: Table to summarize
  • <groupby_col>: Column(s) to group by
  • "Label": Name for calculated column
  • <expression>: Aggregation expression

EXAMPLE FROM YOUR DATASET:

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

  Result: Table with:
    North | ₹5,973,000 | 167 orders | ₹35,750
    South | ₹6,100,000 | 180 orders | ₹33,890
    ... (other regions)

Summary by Category:
  Category Summary = SUMMARIZE(SalesData,
    SalesData[Category],
    "Category Sales", SUM(SalesData[FinalAmount]),
    "Item Count", SUM(SalesData[Quantity]))

REAL USAGE:
  • Create summary tables
  • Multi-level aggregation
  • Complex reports
```