DAX Row Context vs Filter Context — Developer Cheat Sheet

🧱 PART 1 — ROW CONTEXT FUNCTIONS

Row context exists when DAX evaluates an expression **for each row** of a table.

These functions **create or work within row context.**

🧩 1. Calculated Columns

While not a function, calculated columns automatically create a **row context** for each row in the table.

Example:

Profit = Sales[SalesAmount] - Sales[Cost]

👉 Evaluated **row by row**, no filters involved.

🧩 2. Iterators (X Functions)

All "X" functions **create a row context** as they iterate through a table.

Function	**Description**	**Example**
SUMX	Iterates a table, evaluates expression per row, then sums results.	SUMX(Sales, Sales[Quantity] * Sales[Unit Price])
AVERAGEX	Evaluates expression per row, returns average.	AVERAGEX(Orders, Orders[Amount])
COUNTX	Counts rows where the expression is not blank.	COUNTX(Products, Products[Category])
MINX / MAXX	Find min/max value across rows of an expression.	MAXX(Orders, Orders[Profit])
RANKX	Ranks rows based on evaluated expression.	RANKX(ALL(Customer), [Total Sales])
PRODUCTX	Multiplies expression values across rows.	PRODUCTX(Sales, Sales[Quantity])
MEDIANX	Finds median of an expression over a table.	MEDIANX(Orders, Orders[DeliveryTime])
VARX.P / VARX.S, STDEVX.P / STDEVX.S	Calculate variance or standard deviation across row evaluations.	STDEVX.S(Sales, Sales[Amount])

💡 **Key insight:**

All these functions iterate row by row (row context) → evaluate an expression → aggregate results.

🧩 3. EARLIER / EARLIEST

Used to **access a previous row context** — typically inside nested row contexts.

Function	Purpose	Example
EARLIER(column)	Accesses the value of a column from an **outer** row context.	CALCULATE(SUM(Sales[Amount]), Customer[CustomerID] = EARLIER(Customer[CustomerID]))
EARLIEST(column)	Accesses the **first** (outermost) row context in nesting.	Used rarely, similar concept but for deeper nesting.

💬 Think: "look up the value from the row we were on before this inner loop."

🧩 4. ADDCOLUMNS / SELECTCOLUMNS

Both create **virtual row contexts** when building new tables.

Function	Description	Example
ADDCOLUMNS	Adds calculated columns to a table expression.	ADDCOLUMNS(Products, "Revenue", Sales[Quantity]*Sales[Price])
SELECTCOLUMNS	Creates a new table with selected columns and calculated expressions.	SELECTCOLUMNS(Sales, "Customer", Sales[Customer], "Profit", Sales[Profit])

🧩 5. GENERATE / GENERATEALL

These functions **nest row contexts** — evaluating one table for each row of another.

Function	Description	Example
GENERATE	Joins two tables, evaluating the second for each row of the first.	GENERATE(Customers, FILTER(Sales, Sales[CustomerID] = Customers[CustomerID]))
GENERATEALL	Similar, but doesn't apply relationship filters.	

🧩 6. RELATED / RELATEDTABLE

They rely on **existing row context** to navigate relationships.

Function	Description	Example
RELATED	Brings a value from a related table (one side → many side).	RELATED(Customer[Region])
RELATEDTABLE	Returns the related table (many side → one side).	COUNTROWS(RELATEDTABLE(Sales))

💬 These depend on the "current row" to know which related data to pull.

PART 2 — FILTER CONTEXT FUNCTIONS

Filter context defines **which rows are visible** when DAX calculates an expression.
These functions **create, modify, or remove filters.**

1. CALCULATE

The **most important function** in all of DAX.
It changes or adds filters → then re-evaluates the expression.

```
CALCULATE(
  SUM(Sales[SalesAmount]),
  Customer[Country] = "India"
)
```

- Converts **row context → filter context**
- Modifies filters
- Changes the evaluation context of a measure

2. CALCULATETABLE

Like CALCULATE, but returns a **table** instead of a scalar.

```
CALCULATETABLE(
  Sales,
  Customer[Country] = "India"
)
```

3. FILTER

Creates a new **filtered table** that changes the filter context when used inside CALCULATE.

```
CALCULATE(SUM(Sales[Amount]), FILTER(Sales, Sales[Quantity] > 10))
```

💬 Often used to define complex filter conditions.

4. ALL / ALLEXCEPT / ALLSELECTED / REMOVEFILTERS

Used to **remove or control filters**.

Function	What it does	Example
ALL(Table/Column)	Removes all filters on specified table/column.	CALCULATE(SUM(Sales[Amount]), ALL(Sales))
ALLEXCEPT(Table, Columns...)	Removes all filters **except** specified columns.	CALCULATE(SUM(Sales[Amount]), ALLEXCEPT(Sales, Sales[Region]))
ALLSELECTED(Table/Column)	Removes filters **but keeps user selections** (from visuals).	CALCULATE(SUM(Sales[Amount]), ALLSELECTED(Customer))
REMOVEFILTERS(Table/Column)	Modern, clearer way to remove filters.	CALCULATE(SUM(Sales[Amount]), REMOVEFILTERS(Sales))

5. KEEPFILTERS

Modifies how filters inside CALCULATE behave — instead of replacing filters, it **adds to them**.

CALCULATE(SUM(Sales[Amount]), KEEPFILTERS(Customer[Country] = "India"))

6. VALUES / DISTINCT / SELECTEDVALUE

Retrieve filtered values — based on current filter context.

Function	**Description**	**Example**
VALUES	Returns unique values under current filter context (can be blank).	VALUES(Customer[Region])
DISTINCT	Same as VALUES but never returns blank.	DISTINCT(Customer[Region])
SELECTEDVALUE	Returns the single visible value (or alternate if multiple).	SELECTEDVALUE(Customer[Country], "Multiple Countries")

7. HASONEVALUE / ISFILTERED / ISCROSSFILTERED

Used to check the state of the filter context.

Function	**Purpose**	**Example**
HASONEVALUE	True if exactly one value in context.	IF(HASONEVALUE(Customer[Country]), VALUES(Customer[Country]))
ISFILTERED	True if a column is directly filtered.	ISFILTERED(Customer[Country])
ISCROSSFILTERED	True if a column/table is indirectly filtered via relationships.	ISCROSSFILTERED(Customer[Country])

8. CROSSFILTER

Temporarily modifies **relationship direction or active/inactive** behavior.

```
CALCULATE(
  SUM(Sales[Amount]),
  CROSSFILTER(Sales[CustomerID], Customer[CustomerID], BOTH)
)
```

9. USERELATIONSHIP

Temporarily activates an **inactive relationship** in a measure.

```
CALCULATE(
  SUM(Sales[Amount]),
  USERELATIONSHIP(Sales[OrderDate], Calendar[Date])
)
```

10. TREATAS

Applies filters from one table/column to another — extremely powerful for **dynamic filtering**.

```
CALCULATE(
  SUM(Sales[Amount]),
  TREATAS(VALUES(Regions[Country]), Customer[Country])
```

)

🧩 11. ALLNOBLANKROW

Used to remove the special “blank” row in relationships when returning distinct values.

🧩 12. DAX Time Intelligence Functions (Built on Filter Context)

🧭 1 — Core Time Intelligence (Prebuilt)

These are built-in functions that *internally modify the filter context* to compare or aggregate over time.

Function	**Description**	**Example**
TOTALYTD	Extends the current date filter to include all dates from start of year to current date.	TOTALYTD(SUM(Sales[Amount]), 'Date'[Date])
TOTALMTD	Same as YTD but for the month.	TOTALMTD(SUM(Sales[Amount]), 'Date'[Date])
TOTALQTD	Same as YTD but for the quarter.	TOTALQTD(SUM(Sales[Amount]), 'Date'[Date])
DATESYTD	Returns a table of dates from the beginning of the year to the current date.	CALCULATE(SUM(Sales[Amount]), DATESYTD('Date'[Date]))
DATESMTD	Returns dates from start of month to current date.	CALCULATE(SUM(Sales[Amount]), DATESMTD('Date'[Date]))
DATESQTD	Returns dates from start of quarter to current date.	CALCULATE(SUM(Sales[Amount]), DATESQTD('Date'[Date]))

🧩 **These functions modify filter context** — they don’t create any row context.

When you write:

Sales YTD = TOTALYTD(SUM(Sales[Amount]), 'Date'[Date])

👉 The function internally expands the **filter context** from “Current Month” → “All Dates in the Year so far”.

🔁 2 — Time Shifting (Comparative)

These functions shift the **date filter context** to another time period (previous month, last year, etc.).

Function	**Description**	**Example**
SAMEPERIODLASTYEAR	Shifts current date filter by -1 year.	CALCULATE(SUM(Sales[Amount]), SAMEPERIODLASTYEAR('Date'[Date]))
PREVIOUSYEAR	Returns all dates in the previous year.	CALCULATE(SUM(Sales[Amount]), PREVIOUSYEAR('Date'[Date]))
NEXTYEAR	Returns all dates in the next year.	CALCULATE(SUM(Sales[Amount]), NEXTYEAR('Date'[Date]))
PREVIOUSMONTH	Returns all dates in the previous month.	CALCULATE(SUM(Sales[Amount]), PREVIOUSMONTH('Date'[Date]))
NEXTMONTH	Returns all dates in the next month.	CALCULATE(SUM(Sales[Amount]), NEXTMONTH('Date'[Date]))
PREVIOUSDAY	Returns the single previous day.	CALCULATE(SUM(Sales[Amount]), PREVIOUSDAY('Date'[Date]))

Function	Description	Example
NEXTDAY	Returns the next day.	CALCULATE(SUM(Sales[Amount]), NEXTDAY('Date'[Date]))
PARALLELPERIOD	Shifts the current filter by N intervals (days, months, quarters, years).	CALCULATE(SUM(Sales[Amount]), PARALLELPERIOD('Date'[Date], -1, MONTH))
DATEADD	Similar to PARALLELPERIOD, shifts date context by N intervals but allows negative/positive values.	CALCULATE(SUM(Sales[Amount]), DATEADD('Date'[Date], -1, YEAR))

🧩 **Filter context is modified**, not row context.
The functions produce a *new table of dates*, which Power BI uses as a new filter before recalculating your measure.

📆 *3 — Rolling Windows & Period-to-Date*

Rolling or moving calculations combine **iterators (row context)** + **time-based filter context**.

Function Type	Context Type	Example
Rolling 12 Months	Both (uses FILTER → Row Context, CALCULATE → Filter Context)	DAX Rolling 12M Sales = CALCULATE([Total Sales], DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -12, MONTH))
Moving Average	Both	DAX Moving Avg 3M = AVERAGEX(DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -3, MONTH), [Total Sales])
Year-over-Year	Filter Context	DAX YoY % = DIVIDE([Sales] - [Sales PY], [Sales PY])

💬 Here's how they mix:

- DATESINPERIOD() changes **filter context** (defines date range).
- AVERAGEX() or SUMX() creates **row context** over that date range.

🧩 *4 — Table-Returning Date Functions*

These produce a **table of dates** — used within CALCULATE to redefine the filter context.

Function	Context	Description	Example
DATESBETWEEN	Filter Context	Returns all dates between two given dates.	CALCULATE(SUM(Sales[Amount]), DATESBETWEEN('Date'[Date], DATE(2025,1,1), DATE(2025,6,30)))
DATESINPERIOD	Both	Returns all dates in an interval (useful for rolling periods).	DATESINPERIOD('Date'[Date], MAX('Date'[Date]), -90, DAY)

Function	Context	Description	Example
DATESYTD / DATESMTD / DATESQTD	Filter Context	Period-to-date ranges	DATESYTD('Date'[Date])

🧮 *5 — Advanced Custom Time Intelligence (Manual Context Manipulation)*

When built-in functions aren't flexible enough, you manually control filter context.

Example 1: Custom Fiscal YTD

```
Sales FYTD =
CALCULATE(
  [Total Sales],
  FILTER(
    ALL('Date'),
    'Date'[FiscalYear] = MAX('Date'[FiscalYear]) &&
    'Date'[FiscalMonthNumber] <= MAX('Date'[FiscalMonthNumber])
  )
)
```

👉 Here FILTER() creates **row context** while iterating over all dates,
and CALCULATE() applies that as a **filter context**.

Example 2: Custom Quarter Comparison

```
Sales vs Prev Quarter =
VAR CurrQ = MAX('Date'[Quarter])
RETURN
CALCULATE(
  [Total Sales],
  FILTER(
    ALL('Date'),
    'Date'[Quarter] = CurrQ - 1
  )
)
```

🧩 PART 3 — COMBINED USE

Some functions **bridge both contexts**:

Function	What It Does
CALCULATE	Converts row → filter context
FILTER	Creates new filters row by row
EARLIER	Reads row context inside a filter context
SUMX	Row context inside a filter context evaluation