What is row context? Give an example in a calculated column.
Row context means “DAX is evaluating an expression for a single row and the column values in that row are implicitly available.” You get row context automatically when you create a calculated column: the formula is evaluated once for each row, and any reference like Sales[Quantity] or Sales[UnitPrice] inside that formula returns the value from the same row being evaluated. For example, if you add a calculated column called LineTotal:

LineTotal = Sales[Quantity] * Sales[UnitPrice]

DAX calculates LineTotal row-by-row: for row 1 it multiplies row1.Quantity × row1.UnitPrice, for row 2 it multiplies row2.Quantity × row2.UnitPrice, and so on. Iterators (like SUMX) also create a row context while they iterate a table. If you need a value from a related table inside a calculated column, you use RELATED() because row context does not automatically reach into related tables.

What does CALCULATE(SUM(Sales[Quantity]), Sales[Category] = "Electronics") return?
That expression returns the sum of Quantity but with the filter Category = "Electronics" applied. In plain terms: it ignores whatever category filter might already exist in the visual (or replaces it for that column) and computes the total quantity only for rows where Category equals "Electronics". Other filters (for example, on Date or Region) remain in effect unless you explicitly remove them. If no rows match "Electronics", the SUM returns blank (you can wrap with COALESCE(...,0) or IF(ISBLANK(...), 0, ...) to force zero).

Explain the difference between VAR and RETURN in DAX.
VAR declares a named variable inside a measure or calculated column; it stores a value (scalar or table) that you can reuse later in the expression. RETURN marks the end of the variable definitions and gives the final expression that produces the result. Variables improve readability and performance because you compute something once, store it, and reuse it instead of repeating expensive expressions.

Why does CALCULATE override existing filters?
CALCULATE is the DAX function that *modifies* the filter context for the expression it evaluates. When you pass filter arguments to CALCULATE (either boolean expressions like Table[Col] = "X" or filter functions like ALL()), CALCULATE builds a new filter context by applying those filters. If a filter argument targets the same column that’s already filtered, the CALCULATE filter replaces the existing filter on that column (replacement semantics). Filters passed into CALCULATE are combined logically (AND) with other filters unless you explicitly remove them with ALL() or use KEEPFILTERS() to force additive behavior. Because CALCULATE is explicitly changing the filter context, it can force a value regardless of what slicers or visual filters were present — that’s its purpose. If you want to add a filter without

replacing the existing filter on the same column, use KEEPFILTERS() or build the filter differently (e.g., CALCULATE([Measure], KEEPFILTERS(Table[Col] = "X"))).

Troubleshoot: A CALCULATE measure ignores a slicer. What’s the likely cause?
If a measure using CALCULATE is not responding to a slicer, likely causes and how to check/fix them are:

1. You removed filters inside the measure — the measure contains ALL(...), REMOVEFILTERS(...), or a filter that explicitly overrides the slicer. If CALCULATE([TotalSales], ALL(Sheet1[Region])) is used, it clears the Region filter so slicer changes won’t affect the result. Fix: remove ALL or use KEEPFILTERS if you want to add a filter without removing existing ones.
2. You hard-coded a filter inside CALCULATE — e.g. CALCULATE([TotalSales], Sheet1[Region] = "West") forces West and thus ignores slicer selections. Fix: remove the hard-coded equality or make it dynamic (use TREATAS or link to the slicer).
3. Slicer is from a disconnected table — the slicer’s field belongs to a table that has no relationship to your fact table, so selecting it has no effect. Fix: create a proper relationship or use TREATAS(VALUES(ConnectedSlicer[Field]), FactTable[Field]) in CALCULATE to map slicer values into the fact table.
4. Relationship is inactive or wrong direction — the slicer table and fact table may be related but the relationship is inactive or filter direction prevents propagation. Fix: enable relationship or use USERELATIONSHIP() inside CALCULATE or change cross-filter direction appropriately.
5. Slicer uses different column/hierarchy than measure uses — e.g., slicer based on Date[Year] but measure uses Sheet1[Date] column directly; mismatched fields may not filter each other. Fix: use a proper Date dimension table and make both visuals/measures reference the same Date table.
6. Visual interactions disabled — the visual may have interactions turned off (Format → Edit interactions) so slicer clicks don’t affect the visual. Fix: enable interaction.
7. Measure uses ALLSELECTED() or other time-intelligence quirks — ALLSELECTED can produce results that ignore page slicers depending on context. Understand which ALL* variant you used and change if needed