What is row context? Give an example in a calculated column.

Row context means Power BI is evaluating a formula **one row at a time** in a table.

When using a **calculated column**, Power BI automatically knows **which row** it is currently calc ulating and can use other column values *from the same row*.

Key Point

- **Row context = row-by-row calculation**
- It exists automatically in **calculated columns**, but **not** in measures.

Write a measure that finds total sales

```
Total sales = SUMX(Sales, Sales[Quantity]*Sales[UnitPrice])
```

Use RELATED to fetch the Name from the Customers table into the Sales table.

```
CustomerName = RELATED(Customers[Name])
```

What does CALCULATE(SUM(Sales[Quantity]), Sales[Category] = "Electronics") ret urn?

It returns sum of only all electronics

Explain the difference between VAR and RETURN in DAX.

VAR

- VAR is used to **store a value or calculation one time**.
- It makes your DAX **faster**, **cleaner**, and **easier to read**.
- You can store:
    - Numbers
    - Measures
    - Tables
    - Calculations

RETURN

- RETURN tells Power BI **which result should be output** at the end of the measure.
- It uses the values defined in the VAR section.

Create a calculated column in Sales called TotalPrice using row context (Quantity * UnitPrice).

Done

Write a measure Electronics Sales using CALCULATE to sum sales only for the "Elec tronics" category.

```
Electronics sales = CALCULATE(SUM(Sales[TotalPrice]), Sales[Category]="Electronics")
```

Use ALL(Sales[Category]) in a measure to show total sales ignoring category filters

.

ALL removes filters from the specified column or table. When used inside CALCULATE, it ove rrides the current filter context, allowing us to return results that ignore slicers or visual filters.

```
Total Sales (ignore category) = CALCULATE(SUM(Sales[TotalPrice]), All(Sales[Category]))
```

Fix this error: A calculated column in Sales uses RELATED(Customers[Region]) but returns blanks.

Why the Error Occurs

RELATED(Customers[Region]) works **only when**:

1. **Sales** is on the **many** side of a **one-to-many** relationship.
2. **Customers** is on the **one** side.
3. The relationship is **active** and **Single direction** (from Customers → Sales).

If:

- There's **no relationship**
- Or the relationship is **inactive**
- Or the direction is **from Sales → Customers** (wrong direction)

→ RELATED() **returns BLANK**.

Check the Relationship

Go to **Model View** →

Ensure you have:

Customers[CustomerID] 1 ---> * Sales[CustomerID]

Direction: *Single*

Status: *Active*

If not:

- Create or edit the relationship.

Enable Cross Filtering Direction

If you cannot change the model structure:

Set cross filter direction to **Both**:

Model View → Click Relationship →

Cross filter direction: Both

. If Relationship Cannot Be Activated

Use LOOKUPVALUE() instead of RELATED():

Region =

LOOKUPVALUE(

Customers[Region], -- Column to return

Customers[CustomerID], -- Search Column

Sales[CustomerID] -- Value to match

)

This works **even when filter direction isn't correct**.

Why does CALCULATE override existing filters?

CALCULATE() can **override existing filters** because it **changes the filter context**. It takes the *current* context and then **adds, replaces, or removes filters** based on the filter arguments you pa ss into it.

How CALCULATE Overrides Filters

CALCULATE() has two jobs:

1. **Evaluate an expression** (like SUM of Sales).
2. **Modify the filter context** using the filters you pass in.

```
Total Sales All Years =

CALCULATE(

    SUM(Sales[Amount]),

    ALL(Calendar[Year])     -- Removes the year filter

)
```

Write a measure that returns average unitprice of products

```
Avg UNitPrice = CALCULATE(AVERAGE(Sales[UnitPrice]),all(Sales[ProductID]))
```

Use VAR to store a temporary table of high-quantity sales (Quantity > 2), then count rows.

```
High Quantity Sales Count =

VAR HighQtyTable =

    FILTER(
```

```
        Sales,
        Sales[Quantity] > 2
    )
RETURN
    COUNTROWS(HighQtyTable)
```

Write a measure `% of Category Sales` that shows each sale's contribution to its category total.

```
% of Category Sales =
Var TotalSales = Sum(Sales[TotalPrice])
 VAR EachSales = CALCULATE(sum(Sales[TotalPrice]), all(Sales[Category]))
 RETURN DIVIDE(EachSales, TotalSales)
```

Simulate a "remove filters" button using ALL in a measure.

```
Sales (Ignore Filters) =
CALCULATE(
    SUM(Sales[TotalPrice]),
    ALL(Sales)
)
```

Troubleshoot: A CALCULATE measure ignores a slicer. What's the likely cause?

The most likely cause is that the measure is using a **function that removes filters**, such as **ALL**, **ALLEXCEPT**, or **REMOVEFILTERS**, inside **CALCULATE**.

Why this happens

- **CALCULATE()** changes the filter context.
- If your measure contains something like:

Total Sales (Ignoring Filter) =

CALCULATE(

SUM(Sales[SalesAmount]),

ALL(Sales) -- ← This removes slicer filters

)

The slicer tries to filter the data → but **ALL(Sales)** clears those filters → so the slicer has no effect.

How to Fix

Remove or narrow the **ALL()** function.

Or if you must use ALL (for percent calculations), restrict it: