TABLEAU TUESDAYS

Aggregation in Tableau vs SQL

One concept. Two tools. One pipeline.

THE FOUNDATION

What is an aggregation?

Aggregation answers one question: **At what level of detail do you want to see your data?**

SUM, COUNT, AVG, MAX, MIN are just different ways of answering it.
Many rows collapse into one summary

RAW DATA (ROW-LEVEL)	AFTER AGGREGATION
3 transactions Nairobi · KSh 5,000 Mombasa · KSh 3,200 Kisumu · KSh 7,500	**1 summary row** Total revenue **KSh 15,700**

- **Aggregation is fundamentally about granularity i.e., what level of detail do you want to see?**

HOW SQL AGGREGATES: GROUP BY

In SQL, every aggregation is explicit.

```
SELECT
    region,
    SUM (sales) AS total_sales,
    COUNT (DISTINCT customer_id) AS customers,
    AVG (sales) AS avg_sales
FROM customer_details
WHERE sale_date >= '2026-01-01'
GROUP BY 1
ORDER BY 2 DESC
```

region	total_sales	customers	avg_sale
Nairobi	182,400	47	3,881
Mombasa	94,200	28	3,364
Kisumu	61,750	19	3,250

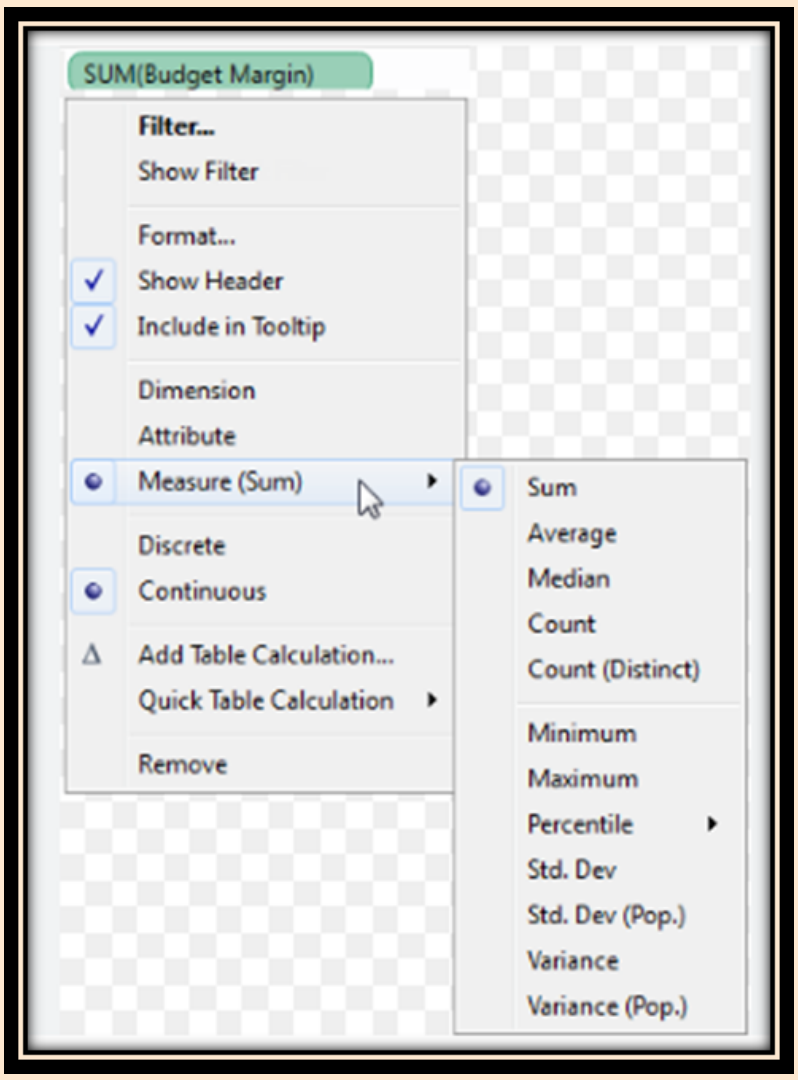
HOW TABLEAU AGGREGATES: AUTOMATICALLY

- **Tableau aggregates implicitly.**

The canvas is the GROUP BY.
Whatever dimensions live on your shelves define the aggregation. Change the view, change the grain.

By default, Tableau assumes measures should be totaled.

When you drag a measure to a shelf, it instantly becomes SUM(Sales).
You can change it to AVG, COUNT, etc. by right clicking the pill.



COMMON ERRORS

Aggregate vs non-aggregate errors

a. Tableau: "cannot mix aggregate and non-aggregate"

SUM([Sales]) / [Sales] mixes a summary value with a row-level value.

Tableau doesn't know which grain to use.

Fix: aggregate both sides, or neither.

b. SQL: filtering on aggregates with "WHERE"

WHERE SUM (sales) > 10000 → **error.**

- Use HAVING SUM (sales) > 10000 instead

Tableau's superpower: LOD expressions

What if you need aggregation at a *different* grain than the view? That's what Level of Detail (LOD) expressions solves.

SQL EQUIVALENT **Subquery / CTE**	TABLEAU LOD **FIXED expression**
`WITH cust_totals AS ( SELECT customer_id, SUM(sales) AS ltv FROM orders GROUP BY customer_id ) SELECT region, AVG(ltv) FROM cust_totals ...`	`{ FIXED [Customer ID] : SUM([Sales]) // Then AVG() that in // the view – done. // INCLUDE: adds dims // EXCLUDE: removes dims`

LOD = Nested aggregation

FIXED *— total sales per customer, regardless of what region filter is active*

INCLUDE *— sales per customer per region, even if region isn't on the view*

EXCLUDE *— sales per region, ignoring the date dimension on the view*

FILTER ORDER MATTERS

How Filters change Aggregation

The #1 source of "why is my number wrong?" in Tableau is filter order.

SQL FILTER ORDER	**TABLEAU FILTER ORDER**
1. FROM (load data)	1. Extract filters
2. WHERE (row filter)	2. Data source filters
3. GROUP BY (aggregate)	3. Context filters
4. HAVING (agg filter)	4. FIXED LOD
5. SELECT / ORDER BY	5. Dimension filters
	6. INCLUDE / EXCLUDE
	7. Measure filters
WHERE runs BEFORE aggregation	***FIXED runs before dimension filters!***

PERFORMANCE

SQL and Tableau don't compete — they stack

The question isn't which tool to use, but rather, which tool should carry which burden — each doing what it does best.

SQL - Heavy Lifting

SQL works where the data lives.
Millions of rows, one engine, zero in-memory strain.

Layer 1
Database

Pre-compute once. Query frequently.

A materialised view or Tableau extract stores the pre-aggregated result — so Tableau never touches the raw table again.

Layer 1.2
Database

Tableau aggregates again — interactively

One just needs to drag measure to rows — and Tableau handles GROUP BY for you instantly, at whatever grain makes sense.

Layer 2
Tableau

We've established it's not SQL *or* Tableau — it's SQL *then* Tableau.

In your current workflow, where does SQL stop and Tableau begin?