Comparing SQL Techniques: Subqueries, CTEs, and Window Functions

Recently, I explored **different ways to solve the same SQL problem** — and it was a great reminder that there's always more than one way to think in SQL 💡

The Question: Find stores whose total sales were higher than the average sales across all stores.

Here's how I solved it using **four different SQL techniques**, each with its own use case 👇

1️⃣ Using HAVING + Subquery

```
SELECT store_id, store_name, SUM(price) AS TotalSale
FROM sales
GROUP BY store_id, store_name
HAVING SUM(price) > (
  SELECT AVG(TotalSale) FROM (
    SELECT store_id, store_name, SUM(price) AS TotalSale
    FROM sales
    GROUP BY store_id, store_name
  ) X
);
```

- **Simple and effective**

  Quick aggregation-based filters

- **Best for**

  Straightforward comparisons

2️⃣ Using JOIN

```
SELECT Sales.*
FROM (
  SELECT store_id, store_name, SUM(price) AS TotalSales
  FROM sales
  GROUP BY store_id, store_name
) AS Sales
JOIN (
  SELECT AVG(TotalSale) AS AVG_Sale
  FROM (...) X
) AvgSale
ON Sales.TotalSales > AvgSale.AVG_Sale;
```

Good when you want to **compare group-level results** side by side.
Think of it as joining detailed results with an aggregated summary.

3️⃣ Using a CTE (Common Table Expression)

```
WITH CTE_Sales AS (
  SELECT store_id, store_name, SUM(price) AS TotalSale
  FROM sales
  GROUP BY store_id, store_name
)
SELECT * FROM CTE_Sales
WHERE TotalSale > (SELECT AVG(TotalSale) FROM CTE_Sales);
```

Cleaner & Reusable

Much better than nested subqueries. CTEs improve readability, especially when logic expands.

4️⃣ Using Multiple CTEs

```
WITH CTE_Sales AS (
  SELECT store_id, store_name, SUM(price) AS TotalSale
  FROM sales
  GROUP BY store_id, store_name
),
CTE_Avg AS (
  SELECT AVG(TotalSale) AS AvgSale FROM CTE_Sales
)
SELECT *, TotalSale - AvgSale AS DifferInAvg
FROM CTE_Sales
JOIN CTE_Avg ON CTE_Sales.TotalSale > CTE_Avg.AvgSale;
```

Great for adding **analytical insights** like difference from average. Shows calculated metrics clearly.

5️⃣ Using Window Functions

```
WITH StoreSales AS (
  SELECT store_id, store_name, SUM(price) AS TotalSale
  FROM sales
  GROUP BY store_id, store_name
)
SELECT *
FROM (
  SELECT *, AVG(TotalSale) OVER() AS AvgSale,
  TotalSale - AVG(TotalSale) OVER() AS AvgDiff
  FROM StoreSales
) AS X
WHERE TotalSale > AvgSale;
```

Perfect for Analytics

Shows aggregates beside each row

Important Note

Can't use window functions in WHERE directly — wrap in CTE or subquery

🧠 Key Takeaways

Subqueries

Quick and compact

JOINs

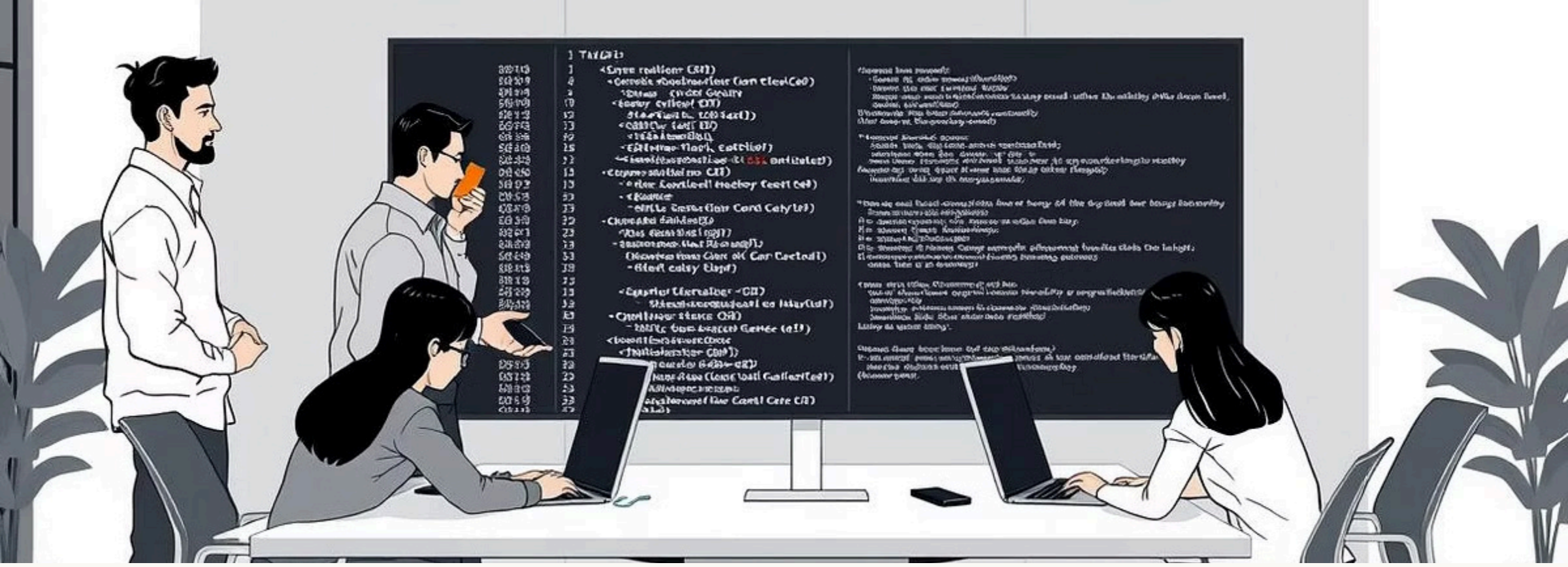
Compare multiple aggregates

CTEs

Improve readability

Window Functions

Analytics and dashboards



What's Your Favorite Approach?

💬 Do you prefer readability or performance first?

Share this carousel with your SQL community!

#SQL #DataAnalytics #DataEngineering #CTE #WindowFunctions #Subqueries #DataCommunity