SQL Query Optimization

Transforming Complex Queries into Performant Code

```sql
SELECT
    segment,
    SUM(amount) AS revenue
FROM (
    SELECT
    t.amount,
    c.customer_segment AS segment,
    t.transaction_date,
    c.country,
    p.business_type
    FROM transactions t
    JOIN customers c ON c.customer_id = t.customer_id
    JOIN products p ON p.product_id = t.product_id
) raw
WHERE raw.transaction_date >= '2023-01-01'
    AND raw.country = 'US'
    AND raw.business_type = 'B2B'
GROUP BY segment;
```

Anandnarayanan S | 💾 Save for later | ➕ Follow for more

The Core Pattern

4-Step Intuition for Query Optimization

1

Early Filtering

Apply WHERE clauses as early as possible to reduce dataset size before joins.

2

Column Pruning

Select only necessary columns in subqueries to minimize memory overhead.

3

Strategic Ordering

Join smallest filtered tables first to reduce Cartesian product growth.

4

CTE Optimization

Use Common Table Expressions for better readability and potential materialization.

✨ **PATTERN APPLICABILITY**

This pattern works for ANY multi-table join query with filtering conditions, especially when dealing with large datasets.

Optimized Solution

Clean, Efficient Query with Early Filtering

```sql
/* Filter largest table FIRST using CTE */
WITH filtered_txn AS (
    SELECT
    t.transaction_id,
    t.amount,
    t.customer_id,
    t.product_id
    FROM transactions t
    WHERE t.transaction_date ≥ '2023-01-01'
)
SELECT
    c.customer_segment,
    SUM(f.amount) AS revenue
FROM filtered_txn f
JOIN customers c ON c.customer_id = f.customer_id
JOIN products p ON p.product_id = f.product_id
WHERE c.country = 'US'
    AND p.business_type = 'B2B'
GROUP BY c.customer_segment;
```