SQL OPTIMIZATION **PART 2**

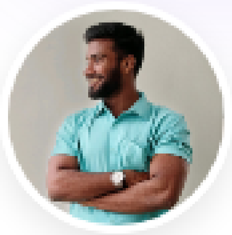
The N+1 Query Trap

When Correlated Subqueries kill your DB performance.

slow_query.sql

```sql
SELECT
  o.order_id,
  o.total_amount,
  -- ⚠️ Subquery runs for EVERY row
  (SELECT customer_name
     FROM customers c
     WHERE c.id = o.customer_id),

  (SELECT COUNT(*)
     FROM order_items oi
     WHERE oi.order_id = o.order_id)
FROM orders o
WHERE o.order_date > '2024-01-01';
```



Anandnarayanan S
Follow for more SQL skills

PATTERN

Row-by-Row vs Set-Based

Why your database is struggling.

🛑

The Trap

3,001

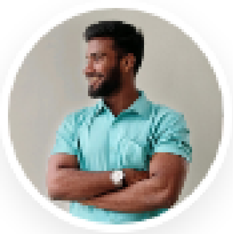
Total Queries (For 1k rows)
1 Main Query +
3 Subqueries × 1,000 Rows

🚀

The Fix

1

Total Query (JOIN Pattern)
Process all data in a single pass using Set Operations



Anandnarayanan S
Follow for more SQL skills

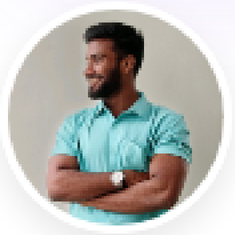
SOLUTION

The JOIN Pattern

Aggregate First, Then Join.

optimized.sql

```sql
SELECT
  o.order_id, c.customer_name,
  COALESCE(oi.item_count, 0) AS count
FROM orders o
LEFT JOIN customers c ON c.id = o.customer_id
-- ✅ Pre-aggregate the "Many" side
LEFT JOIN (
  SELECT
    order_id, COUNT(*) AS item_count
  FROM order_items
  GROUP BY order_id
) oi ON oi.order_id = o.order_id
WHERE o.order_date > '2024-01-01';
```



Anandnarayanan S

Follow for more SQL skills

PART 2 • CORRELATED SUBQUERIES

SUMMARY

Optimization Checklist

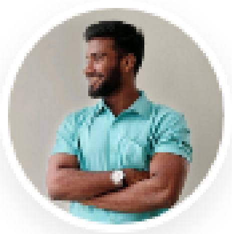
✓ **Audit SELECT Clauses**
Ensure no subqueries inside column list.

✓ **Aggregate First**
Group data in derived tables before joining.

✓ **Use LEFT JOIN**
Process all data in one efficient pass.

Found this helpful?

Repost if you liked this!



Anandnarayanan S
Follow for more SQL skills