



# Find employees earning more than the AVG of "Everyone Else"

Salary Benchmarking



INPUT: EMPLOYEE TABLE

ID	Salary	Department
101	\$60,000	Analytics
102	\$30,000	Analytics
103	\$50,000	Analytics
201	\$70,000	Engineering
202	\$40,000	Engineering
203	\$30,000	Engineering

**The Condition:**

Compare their salary to the Avg of ALL OTHER departments.  
(Exclude their own department from the average).

# The Exclusion Pattern

How to compare "One" vs "The Rest"



## Current Employee

Processing **ID 201** (Engineering).

We need to compare this person's \$70k salary...



## The Benchmark

...against the Average of **Non-Engineering** depts.

**Exclude: Engineering**

Include: Analytics, HR, Sales...

This requires a **Correlated Subquery** that re-calculates for every row.



Anandnarayanan S

+ FOLLOW FOR MORE SQL PATTERNS



## METHOD 1

# Correlated Subquery

The classic "Row-by-Row" approach

```
SELECT e.emp_id, e.salary, e.dept
FROM employee e
WHERE e.salary > (
    -- Calculate Avg for "Others"
    SELECT AVG(x.salary)
    FROM employee x
    -- The Exclusion Logic:
    WHERE x.dept ≠ e.dept
)
ORDER BY e.emp_id;
```

**How it works:** The subquery runs for every employee. If the outer employee is 'Analytics', the inner query averages everything *except* 'Analytics'.



**Anandnarayanan S**

+ FOLLOW FOR MORE SQL PATTERNS



## METHOD 2

# CTE Approach

More readable & modular

```
WITH DeptStats AS (  
    SELECT dept, AVG(salary) AS d_avg  
    FROM employee GROUP BY dept  
)  
OtherDeptAvg AS (  
    SELECT e.emp_id, e.salary,  
           AVG(ds.d_avg) AS other_avg  
    FROM employee e  
    CROSS JOIN DeptStats ds  
    WHERE ds.dept ≠ e.dept  
    GROUP BY e.emp_id, e.salary  
)  
SELECT * FROM OtherDeptAvg  
WHERE salary > other_avg;
```

cte\_optimized.sql



**Anandnarayanan S**

+ FOLLOW FOR MORE SQL PATTERNS



RESULT

# The Final Output

Who earns more than the "Other" departments?

THRESHOLD: ~\$46,667 (AVG OF OTHERS)

ID	Dept	Salary	Status
101	Analytics	\$60,000	✓ INCLUDED
102	Analytics	\$30,000	✗ EXCLUDED
103	Analytics	\$50,000	✓ INCLUDED
201	Engineer	\$70,000	✓ INCLUDED
202	Engineer	\$40,000	✗ EXCLUDED
203	Engineer	\$30,000	✗ EXCLUDED



Anandnarayanan S

+ FOLLOW FOR MORE SQL PATTERNS



## WRAP UP

# Key Takeaways

Master these patterns for your next interview.

01


Correlated Subqueries run **Row-by-Row**. Use them sparingly on large datasets.

02

Use **CTEs** to pre-calculate averages if you need to join multiple times.

03

**!= Operator** inside the subquery is the key to "Exclusion Logic".

Found this helpful?  
**REPOST** to your  
Friends too! 



Anandnarayanan S

+ FOLLOW FOR MORE SQL PATTERNS

