Manish Jaiswal

SQL Interview Question

MASTERING

Correlated vs Non-Correlated Subqueries in SQL

swipe

01

Introduction to Subqueries

A **subquery** is a query nested within another SQL query. It's powerful for performing operations that require referencing data from multiple tables or results.

swipe

Understanding Correlated Subquery

- A correlated subquery is a subquery that depends on the values from the outer query to execute.
- For each row processed by the outer query, the subquery is re-evaluated with the specific values from the current row of the outer query.

03 Example

Suppose we want to find all employees whose salary is greater than the average salary of their respective departments.

```
SELECT emp_id, emp_name, department, salary
FROM employees e1
WHERE salary > (
    SELECT AVG(salary)
    FROM employees e2
    WHERE e1.department = e2.department
);
```

Subquery (SELECT AVG(salary) FROM employees e2 WHERE e1.department = e2.department) is correlated to the outer query by the department column. For each row processed by the outer query (e1), the subquery is re-evaluated with the specific department value from the current row of the outer query.

swipe

04

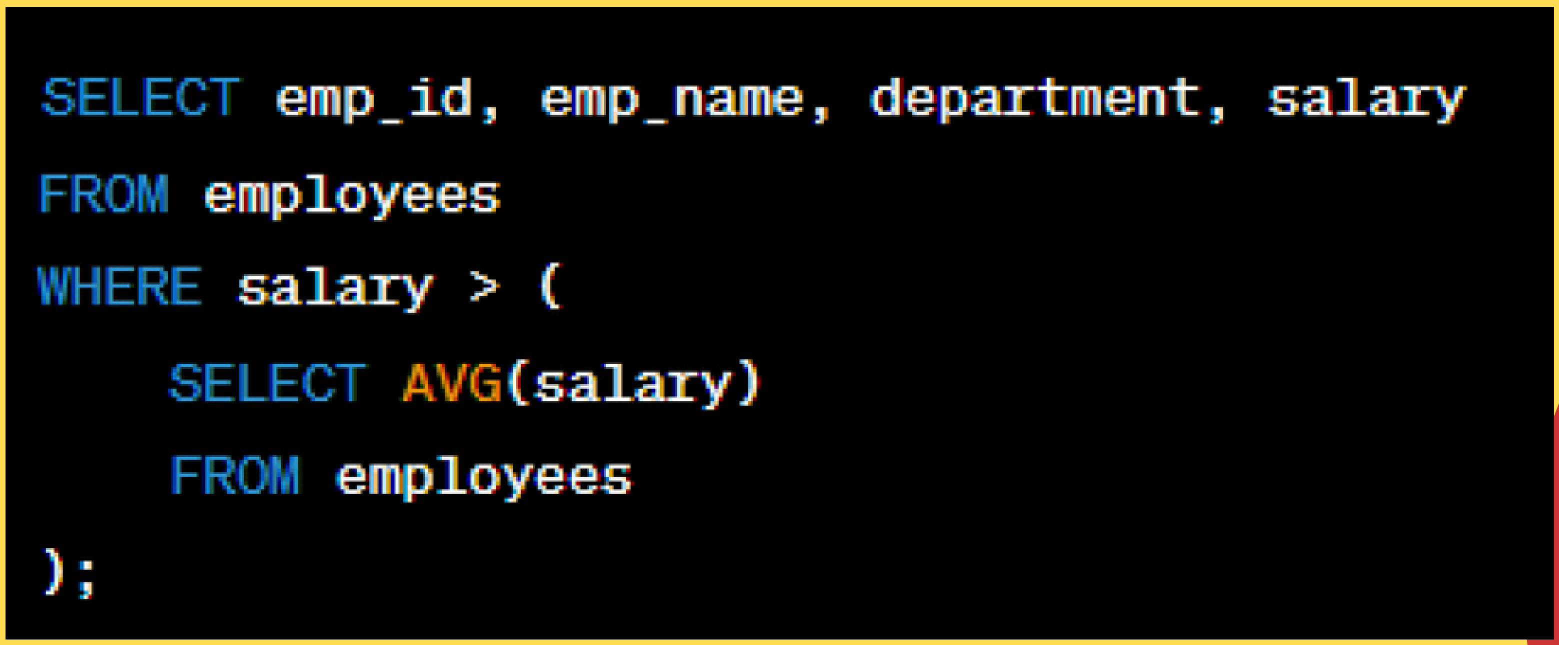
Understanding Non-Correlated Subquery

- A non-correlated subquery is an **independent** query that can be executed on its own without reference to the outer query.
- The subquery is evaluated first, and its result is then used in the outer query to filter or perform other operations.

swipe

Example

Suppose we want to find all employees whose salary is greater than the average salary of all employees.



```sql
SELECT emp_id, emp_name, department, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
);
```

The subquery (**SELECT AVG**(**salary**) **FROM employees**) is evaluated only once and provides the average salary value. The outer query then uses this value to filter the employees whose salary is greater than the average.

swipe

Performance Impact🚀

Correlated Subquery:

- Runs for each row in the outer query, which can impact performance, especially with large datasets. It's best for small data sets or situations where precise, row-by-row logic is necessary.

Non-Correlated Subquery:

- Runs once, making it faster and more efficient for larger data sets. Prefer this type whenever possible to improve performance.

swipe

07

When to Use Each Type

Correlated Subquery:

- Ideal when you need to check each row in the outer query against a condition in the inner query.
    - **Example**: Finding employees with a salary higher than the department average.

Non-Correlated Subquery:

- Best when the inner query's result is static and applies universally.
    - **Example**: Retrieving items from one table that match a value set from another.

swipe

Manish Jaiswal

SQL Interview Question

08

Interview Questions

1. "What is the difference between correlated and non-correlated subqueries?"
2. "Explain when to use a correlated subquery instead of a join."
3. "Describe how you would optimize a correlated subquery."
4. "Provide examples of both types of subqueries in real-world scenarios."

swipe