**SQL Subqueries - Detailed Notes**

**🔹 What is a Subquery?**

A **subquery** (or **inner query** or **nested query**) is a **query within another SQL query**. It is embedded inside SELECT, INSERT, UPDATE, or DELETE statements or inside another subquery.

Subqueries are typically used to:

* Perform intermediate filtering.
* Calculate values for use in the outer query.
* Dynamically respond to the database state.

**🔹 Basic Syntax**

sql

CopyEdit

SELECT column\_name

FROM table\_name

WHERE column\_name operator (SELECT column\_name FROM table\_name WHERE condition);

**🔹 Types of Subqueries**

1. **Single Row Subquery**
2. **Multiple Row Subquery**
3. **Multiple Column Subquery**
4. **Correlated Subquery**
5. **Nested Subquery**
6. **Scalar Subquery**

**🔹 1. Single Row Subquery**

Returns **one row**.

**Example:**

SELECT name, salary

FROM employees

WHERE salary > (SELECT salary FROM employees WHERE name = 'John');

**🔹 2. Multiple Row Subquery**

Returns **multiple rows**. Use operators like IN, ANY, ALL.

**Example:**

SELECT name

FROM employees

WHERE department\_id IN (SELECT department\_id FROM departments WHERE location = 'New York');

**🔹 3. Multiple Column Subquery**

Returns **multiple columns**, typically used with IN or in EXISTS.

**Example:**

SELECT name

FROM employees

WHERE (department\_id, job\_id) IN

(SELECT department\_id, job\_id FROM job\_openings WHERE status = 'Open');

**🔹 4. Correlated Subquery**

References a column from the outer query. Runs **once per row** of the outer query.

**Example:**

SELECT name

FROM employees e

WHERE salary > (

SELECT AVG(salary)

FROM employees

WHERE department\_id = e.department\_id

);

**🔹 5. Nested Subquery**

A subquery inside another subquery.

**Example:**

SELECT name

FROM employees

WHERE department\_id = (

SELECT department\_id

FROM departments

WHERE location\_id = (

SELECT location\_id

FROM locations

WHERE city = 'Chicago'

)

);

**🔹 6. Scalar Subquery**

Returns a **single value (one row, one column)**. Can be used in SELECT or WHERE clauses.

**Example:**

SELECT name,

(SELECT MAX(salary) FROM employees) AS max\_salary

FROM employees;

**🔹 Subqueries in Different Clauses**

**1. In SELECT Clause:**

SELECT name,

(SELECT COUNT(\*) FROM tasks WHERE tasks.emp\_id = employees.id) AS task\_count

FROM employees;

**2. In FROM Clause (Derived Table):**

SELECT dept\_id, AVG(salary) AS avg\_salary

FROM (

SELECT department\_id AS dept\_id, salary

FROM employees

) AS dept\_data

GROUP BY dept\_id;

**3. In WHERE Clause:**

SELECT name

FROM customers

WHERE id IN (SELECT customer\_id FROM orders WHERE order\_date > '2024-01-01');

**4. In HAVING Clause:**

SELECT department\_id, COUNT(\*)

FROM employees

GROUP BY department\_id

HAVING COUNT(\*) > (

SELECT AVG(emp\_count)

FROM (

SELECT department\_id, COUNT(\*) AS emp\_count

FROM employees

GROUP BY department\_id

) AS dept\_counts

);

**🔹 Real-Time Use Cases**

**✅ 1. Find Top Performer in Each Department**

SELECT name, department\_id

FROM employees e

WHERE salary = (

SELECT MAX(salary)

FROM employees

WHERE department\_id = e.department\_id

);

**✅ 2. Filter Customers with High-Value Orders**

SELECT name

FROM customers

WHERE id IN (

SELECT customer\_id

FROM orders

WHERE total\_amount > 1000

);

**✅ 3. Identify Products Never Ordered**

SELECT product\_name

FROM products

WHERE id NOT IN (

SELECT DISTINCT product\_id

FROM order\_items

);

**✅ 4. Detect Duplicate Emails**

SELECT email

FROM users

GROUP BY email

HAVING COUNT(\*) > 1;

**🔹 Performance Tips**

* Use **joins instead of subqueries** when performance is critical.
* **Correlated subqueries** can be slow — consider refactoring into joins or derived tables.
* Index columns involved in subqueries for faster lookups.

**🔹 Interview Tips**

* Be ready to convert subqueries into joins and vice versa.
* Understand when to use IN vs EXISTS vs JOIN.
* Practice writing **correlated subqueries** and understanding their execution flow.

Example for ALL & ANY

Table: employees

id name department salary

1 Alice Sales 5000

2 Bob Sales 6000

3 Charlie HR 7000

4 David HR 5500

5 Eve IT 8000

6 Frank IT 4000

🔸 Example 1: Using ALL

❓Question:

Find employees who earn more than all employees in the Sales department.

✅ SQL:

SELECT name, salary

FROM employees

WHERE salary > ALL (

SELECT salary

FROM employees

WHERE department = 'Sales'

);

🔍 Explanation:

Salaries in Sales: 5000, 6000

We want salaries greater than both of those (i.e., greater than the maximum of them → 6000).

✅ Result:

name salary

Charlie 7000

Eve 8000

🔸 Example 2: Using ANY

❓Question:

Find employees who earn more than any employee in the Sales department.

✅ SQL:

SELECT name, salary

FROM employees

WHERE salary > ANY (

SELECT salary

FROM employees

WHERE department = 'Sales'

);

🔍 Explanation:

We want salaries greater than at least one of 5000 or 6000 (so, greater than 5000).

✅ Result:

name salary

Bob 6000

Charlie 7000

David 5500

Eve 8000