**1. Types of Subqueries**

Subqueries are classified based on how they are used in the main query:

**a) Single-Row Subquery**

* Returns only one row.
* Often used with comparison operators like =, <, >, >=, <=, etc.

**Example:**

sql

Copy code

SELECT name, salary

FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

*Explanation:* Fetches employees with a salary higher than the average salary.

**b) Multi-Row Subquery**

* Returns multiple rows.
* Used with operators like IN, ANY, ALL.

**Example with IN:**

sql

Copy code

SELECT name

FROM employees

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

*Explanation:* Fetches employees who work in departments located in New York.

**Example with ANY:**

sql

Copy code

SELECT name

FROM employees

WHERE salary > ANY (SELECT salary FROM employees WHERE department\_id = 5);

*Explanation:* Fetches employees earning more than the lowest salary in department 5.

**Example with ALL:**

sql

Copy code

SELECT name

FROM employees

WHERE salary > ALL (SELECT salary FROM employees WHERE department\_id = 5);

*Explanation:* Fetches employees earning more than the highest salary in department 5.

**c) Scalar Subquery**

* Returns exactly one value (single row and column).
* Often used in SELECT, WHERE, or HAVING clauses.

**Example:**

sql

Copy code

SELECT name, (SELECT MAX(salary) FROM employees) AS highest\_salary

FROM employees;

*Explanation:* Displays each employee's name along with the highest salary in the company.

**d) Correlated Subquery**

* A subquery that references columns from the outer query.
* Executes repeatedly for each row in the outer query.

**Example:**

sql

Copy code

SELECT name

FROM employees e1

WHERE salary > (SELECT AVG(salary) FROM employees e2 WHERE e1.department\_id = e2.department\_id);

*Explanation:* Fetches employees whose salary is above the average salary of their department.

**2. Uses of Subqueries**

* **Filtering Data:** Use subqueries in WHERE or HAVING clauses.
* **Derived Values:** Compute derived data using subqueries in SELECT statements.
* **Data Validation:** Check the existence of data using subqueries in EXISTS clauses.
* **Updating Data:** Use subqueries in UPDATE statements for conditional updates.

**3. Subquery Clauses**

Subqueries can be used in various SQL clauses, including:

**a) WHERE Clause**

sql

Copy code

SELECT name

FROM employees

WHERE department\_id = (SELECT id FROM departments WHERE name = 'HR');

**b) FROM Clause**

sql

Copy code

SELECT dept\_name, AVG(salary)

FROM (SELECT department\_id, salary FROM employees) AS dept\_data

GROUP BY dept\_name;

*Explanation:* Allows the result of the subquery to act as a temporary table.

**c) SELECT Clause**

sql

Copy code

SELECT name, (SELECT COUNT(\*) FROM projects WHERE manager\_id = employees.id) AS project\_count

FROM employees;

**d) HAVING Clause**

sql

Copy code

SELECT department\_id, AVG(salary)

FROM employees

GROUP BY department\_id

HAVING AVG(salary) > (SELECT AVG(salary) FROM employees);

**4. Key Points to Remember**

* A subquery must always be enclosed in parentheses.
* The subquery's result should match the requirement of the outer query (e.g., a single value for scalar subqueries).
* Use **correlated subqueries** carefully as they can significantly impact performance due to repeated execution.
* Subqueries can be replaced by **JOINs** in many scenarios, especially for better performance.

**5. Comparison Between Subqueries and Joins**

| **Aspect** | **Subquery** | **Join** |
| --- | --- | --- |
| **Performance** | Slower for large datasets due to nested execution. | Faster as it avoids nested queries. |
| **Complexity** | Easier to understand for simple queries. | Can be complex with multiple joins. |
| **Use Case** | Best for filtering or derived values. | Best for combining related tables. |

**6. Advanced Subquery Examples**

**a) Subquery with EXISTS**

sql

Copy code

SELECT name

FROM employees e

WHERE EXISTS (SELECT 1 FROM projects p WHERE p.manager\_id = e.id);

*Explanation:* Fetches employees who are managing at least one project.

**b) Subquery with NOT EXISTS**

sql

Copy code

SELECT name

FROM employees e

WHERE NOT EXISTS (SELECT 1 FROM projects p WHERE p.manager\_id = e.id);

*Explanation:* Fetches employees who are not managing any projects.

**c) Subquery with CASE**

sql

Copy code

SELECT name,

CASE

WHEN salary > (SELECT AVG(salary) FROM employees) THEN 'Above Average'

ELSE 'Below Average'

END AS salary\_category

FROM employees;

*Explanation:* Categorizes employees based on whether their salary is above or below the average salary.