SQL Subqueries

A **subquery** is a query nested inside another query. It is used to retrieve data that will be processed by the outer query. Subqueries can be placed in various parts of an SQL query, including the `SELECT`, `FROM`, and `WHERE` clauses.

Types of Subqueries

1. Single-Row Subqueries

- Return one row as a result.
- Typically used with comparison operators like `=`, `<`, `>`, `<=`, `>=`, or `<>`.

Example: Find employees whose salary is greater than the average salary of all employees.

SQL

```sql
SELECT employee_id, name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

```

---

2. Multiple-Row Subqueries

- Return multiple rows as a result.
- Use operators like `IN`, `ANY`, `ALL`, or `EXISTS`.

Example: Find employees who work in departments located in New York.

SQL

```sql
SELECT employee_id, name
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM departments
    WHERE location = 'New York'
);

```

---

3. Correlated Subqueries

- A **correlated subquery** refers to a column in the outer query and is evaluated for each row processed by the outer query.
- They are slower than regular subqueries because they execute multiple times.

Example: Find employees who earn more than the average salary of their department.

SQL

```sql
SELECT employee_id, name, salary
FROM employees e
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id
);

```

---

4. Nested Subqueries

- Subqueries within subqueries.

Example: Find employees whose salary is higher than the average salary of employees in departments where the average salary exceeds 50,000.

SQL

```sql
SELECT employee_id, name, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id IN (
        SELECT department_id
        FROM departments
        WHERE AVG(salary) > 50000
    )
);

```

Placement of Subqueries

1. In the SELECT Clause

- A subquery can be used to compute derived values.

Example: Show each employee's salary and how it compares to the average salary in their department.

```sql
SELECT name,
       salary,
       (SELECT AVG(salary)
        FROM employees
        WHERE department_id = e.department_id) AS avg_department_salary
FROM employees e;
```

---

2. In the FROM Clause (Derived Tables)

- A subquery can act as a temporary table.

Example: Find the highest salary in each department.

```sql
SELECT department_id, MAX(salary) AS highest_salary
FROM (
    SELECT department_id, salary
    FROM employees
) AS subquery
GROUP BY department_id;
```

---

3. In the WHERE Clause

- Used to filter rows.

Example: Find employees who work in a department with more than 5 employees.

```sql
SELECT employee_id, name
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM employees
    GROUP BY department_id
    HAVING COUNT(*) > 5
);
```

```
9
```

---

4. In the HAVING Clause

- Used to filter groups.

Example: Find departments where the total salary exceeds the average total salary of all departments.

SQL

```sql
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id
HAVING SUM(salary) > (
    SELECT AVG(total_salary)
    FROM (
        SELECT SUM(salary) AS total_salary
        FROM employees
        GROUP BY department_id
    ) AS department_totals
);

```

Operators Commonly Used with Subqueries

1. IN

Example: Find customers who have placed orders.

SQL

```sql
SELECT customer_id, name
FROM customers
WHERE customer_id IN (
    SELECT customer_id
    FROM orders
);

```

2. EXISTS

Example: Find departments with employees.

SQL

```sql
SELECT department_id, name
FROM departments d
```

```sql
WHERE EXISTS (
    SELECT 1
    FROM employees e
    WHERE e.department_id = d.department_id
);
```

3. ANY

- Compare a value to any value in a subquery result.

Example: Find employees who earn more than any employee in department 5.

SQL

```sql
SELECT employee_id, name, salary
FROM employees
WHERE salary > ANY (
    SELECT salary
    FROM employees
    WHERE department_id = 5
);
```

4. ALL

- Compare a value to all values in a subquery result.

Example: Find employees who earn more than all employees in department 5.

SQL

```sql
SELECT employee_id, name, salary
FROM employees
WHERE salary > ALL (
    SELECT salary
    FROM employees
    WHERE department_id = 5
);
```

Best Practices for Subqueries

1. Use **aliases** for clarity, especially in correlated subqueries.
2. For better performance, replace subqueries with **JOINs** when possible.
3. Limit the use of correlated subqueries for large datasets due to performance overhead.

Subquery vs. JOIN

- Subqueries and joins can often achieve the same result, but joins are usually faster for large datasets as they are optimized by the database engine.

Subquery Example:

SQL

```sql
SELECT employee_id, name
FROM employees
WHERE department_id = (
    SELECT department_id
    FROM departments
    WHERE name = 'IT'
);

```

Equivalent JOIN Example:

SQL

```sql
SELECT e.employee_id, e.name
FROM employees e
INNER JOIN departments d
ON e.department_id = d.department_id
WHERE d.name = 'IT';

```

Would you like help practicing subqueries or exploring alternatives like `JOINs` for specific cases?