SQL Subqueries

What is a Subquery?

A subquery, also known as an inner query or nested query, is a query within another SQL query. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used in SELECT, INSERT, UPDATE, and DELETE statements, as well as in the SET clause of an UPDATE statement.

Types of Subqueries

1. Single-row Subquery: Returns zero or one row.
2. Multiple-row Subquery: Returns one or more rows.
3. Multiple-column Subquery: Returns one or more columns.

Usage in SQL Statements

SELECT Statement

```
SELECT column_name
FROM table_name
WHERE column_name = (SELECT column_name FROM another_table WHERE condition);
```

INSERT Statement

```
INSERT INTO table_name (column1, column2, ...)
SELECT column1, column2, ...
FROM another_table
WHERE condition;
```

UPDATE Statement

```
UPDATE table_name
SET column1 = (SELECT column_name FROM another_table WHERE condition)
WHERE condition;
```

DELETE Statement

```
DELETE FROM table_name
WHERE column_name = (SELECT column_name FROM another_table WHERE condition);
```

Correlated vs. Non-Correlated Subqueries

Non-Correlated Subquery

The subquery is independent and can be executed on its own.

```
SELECT column_name
FROM table_name
WHERE column_name IN (SELECT column_name FROM another_table WHERE condition);
```

Correlated Subquery

The subquery depends on the outer query for its values.

```
SELECT column_name
FROM table_name AS t1
WHERE column_name IN (SELECT column_name FROM another_table AS t2 WHERE t1.id =
t2.id);
```

Examples

Single-row Subquery

```
SELECT first_name, last_name
FROM employees
WHERE department_id = (SELECT department_id FROM departments WHERE
department_name = 'Sales');
```

Multiple-row Subquery

```
SELECT first_name, last_name
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

Multiple-column Subquery

```
SELECT first_name, last_name
FROM employees
WHERE (department_id, salary) IN (SELECT department_id, MIN(salary) FROM employees
GROUP BY department_id);
```

Correlated Subquery

```
SELECT e1.first_name, e1.salary
FROM employees e1
WHERE e1.salary > (SELECT AVG(e2.salary) FROM employees e2 WHERE e1.department_id
= e2.department_id);
```

Subquery in FROM Clause

Subqueries can also be used in the FROM clause of a SELECT statement to define a temporary table.

```
SELECT avg_salary
FROM (SELECT AVG(salary) AS avg_salary FROM employees);
```

Tips for Writing Subqueries

Always ensure your subquery returns the correct data type for the outer query condition.
Use table aliases to improve readability, especially in correlated subqueries.
Test subqueries independently to ensure they return the expected results.

Best Practices

Performance: Subqueries can sometimes be less efficient than joins. Consider using joins if performance is an issue.
Readability: Keep subqueries simple and break complex queries into smaller parts for better readability.
Indexing: Ensure that the columns used in subqueries are indexed to improve query performance.