08 Subqueries

Subqueries in SQL

Subqueries, also known as nested queries or inner queries, are SQL queries embedded within another query. They are used to retrieve data from one or more tables based on a specified condition. Subqueries can be used in SELECT, INSERT, UPDATE, and DELETE statements.

Types of Subqueries:

1. **Single Row Subquery:** Returns only one row as the result.
2. **Multiple Row Subquery:** Returns multiple rows as the result.
3. **Multiple Column Subquery:** Returns multiple columns as the result.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name OPERATOR (SELECT column_name FROM table_name WHERE condition);
```

Example Queries:

1. Single Row Subquery:

```
-- Find programmers whose salary is greater than the average salary
SELECT Programmer_Name, Salary
FROM programmers
WHERE Salary > (SELECT AVG(Salary) FROM programmers);
```

2. Multiple Row Subquery:

```
-- Find programmers who have developed software applications
SELECT Programmer_Name
FROM programmers
WHERE Programmer_Name IN (SELECT Programmer_Name FROM softwareApp);
```

3. Multiple Column Subquery:

```
-- Find programmers who have studied at MIT and Stanford
SELECT Programmer_Name
FROM studies
WHERE (Institute, Course) IN (SELECT Institute, Course FROM studies WHERE Institute IN
('MIT', 'Stanford'));


-- Method 2
SELECT programmer_name, Institute, Course
FROM (SELECT programmer_name, Institute, Course FROM studies WHERE Institute IN ('MIT',
'Stanford')) as t ;
```

Use Cases of Subqueries:

1. **Filtering Data:** Subqueries are useful for filtering data based on dynamic conditions that cannot be hardcoded.
2. **Calculating Aggregate Values:** Subqueries can be used to calculate aggregate functions like SUM, AVG, COUNT, etc., dynamically based on certain criteria.
3. **Data Comparison:** Subqueries are handy for comparing data between different tables or within the same table.
4. **Finding Extremes:** Subqueries are helpful for finding maximum, minimum, or other extreme values from a dataset.
5. **Nested Conditions:** Subqueries can be used within conditional clauses like WHERE, HAVING, and FROM, allowing for complex logical conditions.

Questions Using Subqueries:

1. **Second Highest Paid Programmer:**

```
SELECT Programmer_Name, Salary
FROM programmers
WHERE Salary = (SELECT DISTINCT Salary FROM programmers ORDER BY Salary DESC LIMIT 1 OFFSET
1);
```

2. **Programmers Studying Most Expensive Course:**

```
SELECT Programmer_Name
FROM studies
WHERE Course_Fee = (SELECT MAX(Course_Fee) FROM studies);
```

3. **Programmers Who Developed Costliest Software:**

```
SELECT Programmer_Name
FROM softwareApp
WHERE Software_Cost = (SELECT MAX(Software_Cost) FROM softwareApp);
```

Conclusion:

Subqueries provide a powerful mechanism for performing complex data retrieval and manipulation operations in SQL. They offer flexibility and enable developers to write more dynamic and expressive queries. By understanding how to use subqueries effectively, you can write more efficient and concise SQL code to meet various business requirements.

Questions:

-- 1. Find programmers who earn more than the average salary of programmers with the same primary language.

```
SELECT Programmer_Name, Salary, Primary_Language
FROM programmers p1
WHERE Salary > (
    SELECT AVG(Salary)
    FROM programmers p2
    WHERE p2.Primary_Language = p1.Primary_Language
);
```

- Correlated subqurey - Related with outer query

-- 2. For each programmer, show their name, salary, and the number of software they've developed

```
SELECT
    p.Programmer_Name,
    p.Salary,
    (SELECT COUNT(*) FROM software s WHERE s.Programmer_Name = p.Programmer_Name) AS
Software_Count
FROM programmers p;
```

-- 3. Find programmers who earn more than all programmers with 'Python' as their primary language.

```
SELECT Programmer_Name, Salary
FROM programmers
WHERE Salary > ALL (
    SELECT Salary
    FROM programmers
    WHERE Primary_Language = 'Python'
);
```