📘 Advanced SQL – Lesson 5: Subqueries

🎯 What You'll Learn

In this lesson, you'll learn how to:

- Use **subqueries** (aka nested queries or inner queries) within `SELECT`, `FROM`, and `WHERE` clauses
- Understand the role of subqueries in filtering and transforming data
- Compare subqueries to window functions, CTEs, and temp tables

---

🧠 What is a Subquery?

A **subquery** is a SQL query **embedded inside another query**. It is used to:

- Return data to be used by the **outer query**
- Act as a dynamic filter, table, or value within the main query
- Be placed inside `SELECT`, `FROM`, `WHERE`, and even `INSERT`, `UPDATE`, or `DELETE` statements

---

🔍 Where Subqueries Can Be Used

1. In the SELECT Clause

Subqueries can be used to calculate values like averages or totals on the fly, often serving as a substitute for a window function.

> Example Use: Show each employee's salary alongside the overall average salary.

---

2. In the FROM Clause

A subquery in the `FROM` clause acts like a temporary table that you can query from.

> Tip: While useful, this can be less efficient than using **CTEs** or **temp tables** for reusable logic.

---

3. In the WHERE Clause

Subqueries in the `WHERE` clause help filter data based on dynamic conditions (e.g., values from another table).

> Example Use: Return employees whose IDs are in a list of those over a certain age from another table.

📂 Real-World Applications

- Filtering results based on related tables without joining
- Dynamically calculating thresholds for filtering or ranking
- Embedding complex logic without altering outer query structure
- Returning derived values inline without creating separate views or tables

---

✅ Best Practices

- Subqueries are powerful but can **impact performance** in large datasets
- When filtering by subquery, make sure it returns **only one column**
- For reusable logic, consider using a **CTE or temp table** instead
- Nesting subqueries too deeply can reduce clarity — keep it clean!

📌 Recap

✅ Subqueries let you run a query **inside another** to retrieve dynamic values or filter data
✅ Useful in `SELECT`, `FROM`, and `WHERE` clauses for powerful flexibility
✅ Best used for **quick, inline transformations or filters**
✅ Consider using **CTEs or temp tables** for more complex, repeatable logic

---

🏁 Final Note

This is the final lesson in the **Advanced SQL** series! You’ve now learned:

- Joins, Unions, and Case Statements
- Temp Tables, String Functions, and Stored Procedures
- Subqueries and more

🚀 Now you're ready to build advanced SQL workflows or dive into projects, data analysis, or dashboard development.