Chapter: Subqueries for Smarter Data Access

Topic: Subqueries in SQL

Section 1: Learn

1.1 What Is a Subquery?

A **subquery** is a query that is written inside another SQL query.

It is also called a **nested query** or **inner query**.

Subqueries help break down complex problems and can return:

- A single value
- A single row
- Multiple rows
- An entire result set

They are used for filtering, comparison, transformation, and embedding calculations.

1.2 Why Use Subqueries?

- Break large problems into smaller, readable steps
- Dynamically filter or transform data
- Embed logic without writing multiple queries
- Perform row-by-row calculations using correlated subqueries

---

1.3 Common Types of Subqueries

1. **Scalar Subquery** – Returns a single value
2. **Multiple Row Subquery** – Returns many values (e.g., used with IN)

3. **Table Subquery** – Used in FROM clause
4. **Correlated Subquery** – References column from outer query and executes for each row

---

1.4 Subquery in WHERE Clause (Single Value)

Get employees earning more than the average salary:

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

1.5 Subquery in WHERE Clause (Multiple Values)

Get names of students who took tests:

```
SELECT name
FROM students
WHERE id IN (SELECT student_id FROM test_scores);
```

1.6 Subquery in FROM Clause (Inline View)

Calculate average salary per department, and filter those with high averages:

```
SELECT department, avg_salary
FROM (
  SELECT department, AVG(salary) AS avg_salary
  FROM employees
  GROUP BY department
) AS dept_avg
```

```
WHERE avg_salary > 60000;
```

1.7 Correlated Subquery

Get employees earning more than the average of their department:

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

1.8 Subquery in SELECT Clause

Show total orders placed by each customer:

```
SELECT name,
       (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.id) AS
total_orders
FROM customers c;
```

1.9 Subquery Limitations

- Slower than joins for large data sets
- May be disallowed in some places depending on SQL engine
- Correlated subqueries can be resource-intensive

Section 2: Practise

Exercise 1: Employees above company average salary

```
SELECT name, salary

FROM employees

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

Exercise 2: Customers who placed at least one order

```
SELECT name

FROM customers

WHERE id IN (SELECT customer_id FROM orders);
```

Exercise 3: Cities with average order value greater than ₹1000

```
SELECT city

FROM (

  SELECT city, AVG(order_value) AS avg_order

  FROM orders

  GROUP BY city

) AS city_avg

WHERE avg_order > 1000;
```

Exercise 4: Students with highest marks per subject

```
SELECT name, subject, marks
FROM results r
WHERE marks = (
  SELECT MAX(marks)
  FROM results
  WHERE subject = r.subject
);
```

Exercise 5: Count of products in each category

```
SELECT category_name,
       (SELECT COUNT(*) FROM products p WHERE p.category_id = c.id) AS
product_count
FROM categories c;
```

Section 3: FAQ – Know More

Q1. What's the difference between subqueries and joins?

- Subqueries are useful for filtering and logic separation.
- Joins are preferred for combining rows and generally perform better.

Q2. When to use subqueries instead of joins?

Use subqueries when:

- You need to use an aggregate result for filtering

- You want to embed logic or reuse a partial result
- You need dynamic filtering per row (correlated subqueries)

Q3. Can I use subqueries inside INSERT, UPDATE, DELETE?

Yes. Example:

```
DELETE FROM orders
WHERE customer_id IN (SELECT id FROM customers WHERE active = 0);
```

Q4. Can subqueries be nested more than one level deep?

Yes. You can nest subqueries inside other subqueries. Use carefully as it may affect performance.

Q5. Do all databases support subqueries?

Most modern RDBMS like MySQL, PostgreSQL, SQL Server, and Oracle support subqueries, though syntax might slightly differ.

End of Notes for Chapter: Subqueries for Smarter Data Access