# Working with Subqueries and Nested Subqueries in SQL

In SQL, subqueries (also called inner queries or nested queries) are queries within another query. They allow us to break down complex problems into smaller, manageable parts and can be used in SELECT, INSERT, UPDATE, and DELETE statements.

## 1. Creating a Sample Table

We will first create a table named Employees to demonstrate subqueries and nested subqueries.

SQL Query:  
CREATE TABLE Employees (  
 EmpID INT PRIMARY KEY,  
 Name VARCHAR(50),  
 Department VARCHAR(50),  
 Salary DECIMAL(10,2)  
);

Explanation: The table Employees has columns EmpID, Name, Department, and Salary. EmpID is the primary key.

## 2. Inserting Sample Records

We insert a few sample records for demonstration.

INSERT INTO Employees (EmpID, Name, Department, Salary) VALUES  
(1, 'Aarav Mehta', 'IT', 60000),  
(2, 'Sanya Kapoor', 'HR', 45000),  
(3, 'Rohan Sharma', 'Finance', 55000),  
(4, 'Ananya Gupta', 'IT', 70000),  
(5, 'Kabir Verma', 'Finance', 50000);

Explanation: These records will be used to illustrate subqueries.

## 3. Simple Subquery

Find employees whose salary is above the average salary.

SQL Query:  
SELECT Name, Salary  
FROM Employees  
WHERE Salary > (SELECT AVG(Salary) FROM Employees);

Explanation: The inner query (SELECT AVG(Salary) FROM Employees) calculates the average salary. The outer query then selects employees with salary greater than this value.

## 4. Nested Subquery

Find employees who earn the maximum salary in the IT department.

SQL Query:  
SELECT Name, Salary  
FROM Employees  
WHERE Salary = (  
 SELECT MAX(Salary)  
 FROM Employees  
 WHERE Department = 'IT'  
);

Explanation: The inner query finds the maximum salary in the IT department. The outer query retrieves employee details with that salary.

## 5. Subquery with IN

Find employees who work in departments where the average salary is more than 55,000.

SQL Query:  
SELECT Name, Department, Salary  
FROM Employees  
WHERE Department IN (  
 SELECT Department  
 FROM Employees  
 GROUP BY Department  
 HAVING AVG(Salary) > 55000  
);

Explanation: The inner query selects departments with an average salary greater than 55,000. The outer query retrieves all employees belonging to those departments.

## 6. Using Triggers with Subqueries

Let’s create a trigger that prevents inserting an employee with a salary lower than the minimum salary in the same department.

SQL Query:  
DELIMITER //

CREATE TRIGGER trg\_MinSalary

BEFORE INSERT ON Employees

FOR EACH ROW

BEGIN

IF NEW.Salary < (

SELECT MIN(Salary)

FROM Employees

WHERE Department = NEW.Department

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Salary is below minimum for this department' ;

END IF ;

END//

DELIMITER ;

Explanation: Before inserting a new record, this trigger checks whether the new salary is below the minimum salary in that department. If yes, it raises an error and prevents insertion.

**Audit Log Trigger**

👉 Keep track of every new employee added.

-- Audit table

CREATE TABLE Employee\_Audit (

AuditID INT AUTO\_INCREMENT PRIMARY KEY,

EmpID INT,

Action VARCHAR(20),

ActionTime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

DELIMITER //

CREATE TRIGGER trg\_AuditInsert

AFTER INSERT ON Employees

FOR EACH ROW

BEGIN

INSERT INTO Employee\_Audit (EmpID, Action)

VALUES (NEW.EmpID, 'INSERT');

END//

DELIMITER ;

**Explanation**: Each time a new employee is added, their ID and the action are recorded in the audit log.

**2. Prevent Negative Salary**

👉 Stop inserting or updating employees with salary ≤ 0.

DELIMITER //

CREATE TRIGGER trg\_PreventNegativeSalary

BEFORE INSERT ON Employees

FOR EACH ROW

BEGIN

IF NEW.Salary <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Salary must be greater than zero';

END IF;

END//

DELIMITER ;

And for updates:

DELIMITER //

CREATE TRIGGER trg\_PreventNegativeSalaryUpdate

BEFORE UPDATE ON Employees

FOR EACH ROW

BEGIN

IF NEW.Salary <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Salary must be greater than zero';

END IF;

END//

DELIMITER ;

**3. Automatic Timestamp Update**

👉 Update a LastModified column whenever a record changes.

ALTER TABLE Employees ADD COLUMN LastModified TIMESTAMP;

DELIMITER //

CREATE TRIGGER trg\_UpdateTimestamp

BEFORE UPDATE ON Employees

FOR EACH ROW

BEGIN

SET NEW.LastModified = NOW();

END//

DELIMITER ;

**Explanation**: Keeps a live track of when employee data was last updated.

**4. Cascading Delete**

👉 Automatically delete audit records when an employee is deleted.

DELIMITER //

CREATE TRIGGER trg\_CascadeDelete

AFTER DELETE ON Employees

FOR EACH ROW

BEGIN

DELETE FROM Employee\_Audit WHERE EmpID = OLD.EmpID;

END//

DELIMITER ;

**5. Prevent Duplicate Names in Same Department**

👉 Enforce uniqueness (similar to a constraint).

DELIMITER //

CREATE TRIGGER trg\_PreventDuplicateName

BEFORE INSERT ON Employees

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1 FROM Employees

WHERE Name = NEW.Name AND Department = NEW.Department

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Duplicate employee in same department not allowed' ;

END IF ;

END//

Loops

In **standard SQL**, there’s no generic FOR or WHILE loop like in programming languages.  
But in **procedural SQL** (PL/SQL for Oracle, T-SQL for SQL Server, PL/pgSQL for PostgreSQL, stored procedures in MySQL), we *can* use loops.

And yes — there are also ways to **insert multiple records with a single command** without looping.

**Insert Multiple Records in One Command (No Loop Needed)**

INSERT INTO Employees (EmpID, Name, Department, Salary) VALUES

(6, 'Ritika Jain', 'IT', 62000),

(7, 'Manish Agarwal', 'HR', 48000),

(8, 'Divya Singh', 'Finance', 53000);

**Using Loops in MySQL Stored Procedure**

MySQL allows loops inside procedures with LOOP, WHILE, or REPEAT.

Create a procedure to add employess

DELIMITER //

CREATE PROCEDURE AddEmployee(

IN p\_EmpID INT,

IN p\_Name VARCHAR(50),

IN p\_Department VARCHAR(50),

IN p\_Salary DECIMAL(10,2)

)

BEGIN

INSERT INTO Employees (EmpID, Name, Department, Salary)

VALUES (p\_EmpID, p\_Name, p\_Department, p\_Salary);

END//

DELIMITER ;