Task-8

Objective:

Learn how to write **reusable blocks of SQL** using:

- **Stored Procedures** (for performing tasks with logic)
- **Stored Functions** (for returning a value from logic)

Tools:

- ☑ **MySQL Workbench** (best suited for this task)
- ❌ SQLite does **not** support stored procedures or functions natively

---

Deliverables:

1. One **stored procedure**
2. One **stored function**
3. Usage examples for each (e.g., `CALL procedure_name(...)`, `SELECT function_name(...)`)

Mini Guide:

☑ 1. Stored Procedure:

A stored procedure performs one or more SQL statements — often used for operations like inserting or updating data.

Example:

```
DELIMITER //

CREATE PROCEDURE GetEmployeesByDept(IN dept_name VARCHAR(50))
BEGIN
    SELECT first_name, last_name
    FROM employees
    WHERE department = dept_name;
END //

DELIMITER;

-- Call the procedure
CALL GetEmployeesByDept('Sales');
```

☑ 2. Stored Function:

A stored function returns a single value — useful in SELECT queries or WHERE clauses.

Example:

```
DELIMITER //

CREATE FUNCTION GetTotalSalary (emp_id INT)
RETURNS DECIMAL10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL10,2);
    SELECT salary + IFNULL (bonus, 0) INTO total FROM employees WHERE id =
emp_id;
    RETURN total;
END //

DELIMITER;

-- Use the function
SELECT GetTotalSalary (101);
```

📌 Key Concepts

- Use **`DELIMITER //`** when defining procedures/functions in MySQL.
- Use **IN / OUT / INOUT parameters** in procedures.
- **Functions must return a value** (using `RETURN`).
- Add **conditional logic** using `IF`, `CASE`, `WHILE`, etc.

☑ Expected Outcome

You will understand how to:

- Write modular, reusable SQL logic
- Simplify complex operations
- Reduce code duplication in applications

We attach the file of sql coding and we can open this in **MySQL Workbench**, run it step by step, and test both the procedure and function with the sample data.

Step-by-Step Instructions to Run the Code:

🔹 Step 1: Setup Sample Table (if you don't have one)

```
DROP TABLE IF EXISTS employees;

CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10,2),
    bonus DECIMAL(10,2)
);

-- Insert sample data
INSERT INTO employees (first_name, last_name, department, salary, bonus)
VALUES
('John', 'Doe', 'Sales', 50000, 5000),
('Jane', 'Smith', 'IT', 60000, NULL),
('Alice', 'Brown', 'Sales', 55000, 2000),
('Bob', 'White', 'HR', 45000, 3000);
```

🔹 Step 2: Create Stored Procedure

```
DELIMITER //

CREATE PROCEDURE GetEmployeesByDept(IN dept_name VARCHAR(50))
BEGIN
    SELECT first_name, last_name
    FROM employees
    WHERE department = dept_name;
END //

DELIMITER ;
```

🔹 Step 3: Call the Stored Procedure

```
CALL GetEmployeesByDept('Sales');
```

🔹 Step 4: Create Stored Function

```
DELIMITER //

CREATE FUNCTION GetTotalSalary(emp_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);
    SELECT salary + IFNULL(bonus, 0) INTO total FROM employees WHERE id =
emp_id;
    RETURN total;
END //
```

```
DELIMITER ;
```

🔹 Step 5: Use the Stored Function

```
SELECT GetTotalSalary(1) AS total_salary;
```

☑ Output We Should See:

- `CALL GetEmployeesByDept('Sales')` → shows rows with "John Doe" and "Alice Brown"
- `SELECT GetTotalSalary(1)` → returns `55000.00`
