**Stored Procedures and Functions in MySQL**

**Stored procedures** and **functions** in MySQL are precompiled SQL code blocks that perform specific tasks. They help in reusing code, reducing redundancy, and improving query efficiency.

**1. Stored Procedures**

A stored procedure is a collection of SQL statements that can be executed with a single call.

**Features**

* Can perform multiple queries and logic operations.
* Supports input, output, and input-output parameters.
* Does not return a value directly but can use output parameters to pass results.

**Syntax**

sql

Copy code

CREATE PROCEDURE procedure\_name ([IN|OUT|INOUT] parameter\_name data\_type, ...)

BEGIN

-- SQL statements

END;

**Example: Simple Stored Procedure**

Create a procedure to fetch employees with a salary above a given threshold:

sql

Copy code

DELIMITER //

CREATE PROCEDURE GetHighSalaryEmployees(IN salary\_threshold DECIMAL(10, 2))

BEGIN

SELECT \*

FROM employees

WHERE salary > salary\_threshold;

END //

DELIMITER ;

**Explanation:**

* IN salary\_threshold: Input parameter to specify the salary threshold.
* The SELECT query fetches rows matching the condition.

**Calling the Procedure:**

sql

Copy code

CALL GetHighSalaryEmployees(5000);

**Example: Procedure with Output Parameter**

Create a procedure to calculate the total salary in a department:

sql

Copy code

DELIMITER //

CREATE PROCEDURE GetTotalSalary(IN dept\_id INT, OUT total\_salary DECIMAL(10, 2))

BEGIN

SELECT SUM(salary)

INTO total\_salary

FROM employees

WHERE department\_id = dept\_id;

END //

DELIMITER ;

**Calling the Procedure:**

sql

Copy code

CALL GetTotalSalary(1, @total\_salary);

SELECT @total\_salary;

**2. Functions**

A function is similar to a stored procedure but is designed to return a single value.

**Features**

* Always returns a value using the RETURN statement.
* Cannot modify data in the database (no INSERT, UPDATE, or DELETE).
* Used in SQL expressions.

**Syntax**

sql

Copy code

CREATE FUNCTION function\_name(parameter\_name data\_type, ...)

RETURNS return\_data\_type

BEGIN

-- Logic and calculations

RETURN value;

END;

**Example: Simple Function**

Create a function to calculate the tax for a given salary:

sql

Copy code

DELIMITER //

CREATE FUNCTION CalculateTax(salary DECIMAL(10, 2))

RETURNS DECIMAL(10, 2)

BEGIN

RETURN salary \* 0.1; -- Assume a 10% tax rate

END //

DELIMITER ;

**Using the Function:**

sql

Copy code

SELECT CalculateTax(5000) AS tax;

**Example: Function for Full Name**

Create a function to concatenate first and last names:

sql

Copy code

DELIMITER //

CREATE FUNCTION GetFullName(first\_name VARCHAR(50), last\_name VARCHAR(50))

RETURNS VARCHAR(100)

BEGIN

RETURN CONCAT(first\_name, ' ', last\_name);

END //

DELIMITER ;

**Using the Function:**

sql

Copy code

SELECT GetFullName('John', 'Doe') AS full\_name;

**3. Differences Between Stored Procedures and Functions**

| **Feature** | **Stored Procedure** | **Function** |
| --- | --- | --- |
| **Purpose** | Executes a series of SQL statements. | Returns a single value. |
| **Return** | Uses OUT parameters to return data. | Returns a value with RETURN. |
| **Usage in Queries** | Cannot be used in SQL queries. | Can be used in queries. |
| **Allowed Operations** | Can modify database data. | Cannot modify database data. |

**4. Managing Procedures and Functions**

**View Stored Procedures or Functions**

sql

Copy code

SHOW PROCEDURE STATUS WHERE Db = 'database\_name';

SHOW FUNCTION STATUS WHERE Db = 'database\_name';

**View Procedure or Function Definition**

sql

Copy code

SHOW CREATE PROCEDURE procedure\_name;

SHOW CREATE FUNCTION function\_name;

**Drop a Procedure or Function**

sql

Copy code

DROP PROCEDURE procedure\_name;

DROP FUNCTION function\_name;

**5. Best Practices**

1. **Use Descriptive Names:**
   * Name procedures and functions clearly to indicate their purpose.
2. **Keep Logic Simple:**
   * Avoid overly complex procedures and functions; keep them focused on specific tasks.
3. **Use Transactions:**
   * Use transactions (START TRANSACTION, COMMIT, ROLLBACK) within procedures for data integrity.
4. **Error Handling:**
   * Use DECLARE and HANDLER for handling errors in procedures.

sql

Copy code

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

END;

1. **Test Extensively:**
   * Test all edge cases and performance under load