What Are Stored Procedures in MySQL?

A **Stored Procedure** in **MySQL** is a **saved block of SQL code** stored inside the database that you can execute whenever needed.

Unlike a view:

- □ A view stores a **SELECT query**
- □ A stored procedure stores **multiple SQL statements + logic**

Think of it as a **database function that can perform operations**.

Why Use Stored Procedures?

Stored procedures are useful for:

1. □ Reusing complex logic
2. □ Improving performance (precompiled SQL)
3. □ Enhancing security
4. □ Reducing network traffic
5. □ Implementing business logic inside the database

Step-by-Step Guide with Detailed Examples

We'll reuse similar sample tables.

Step 1: Create Sample Tables

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    country VARCHAR(50)
);

CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
```

```
    amount DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

Insert data:

```
INSERT INTO customers VALUES
(1, 'John Doe', 'john@email.com', 'USA'),
(2, 'Alice Smith', 'alice@email.com', 'UK');

INSERT INTO orders VALUES
(101, 1, '2025-01-01', 500.00),
(102, 1, '2025-01-10', 200.00),
(103, 2, '2025-01-15', 300.00);
```

Step 2: Basic Stored Procedure (No Parameters)

Example: Get All Customers

```
DELIMITER //

CREATE PROCEDURE GetAllCustomers()
BEGIN
    SELECT * FROM customers;
END //

DELIMITER ;
```

Call It:

```
CALL GetAllCustomers();
```

✔ This runs the SELECT query inside the procedure.

Step 3: Stored Procedure with IN Parameter

Example: Get Orders for Specific Customer

```
DELIMITER //

CREATE PROCEDURE GetOrdersByCustomer(IN cust_id INT)
BEGIN
    SELECT *
    FROM orders
    WHERE customer_id = cust_id;
END //

DELIMITER ;
```

Call It:

```
CALL GetOrdersByCustomer(1);
```

✔ Returns only orders for customer 1.

Step 4: Stored Procedure with Multiple Parameters

Example: Get Orders Above Certain Amount

```
DELIMITER //

CREATE PROCEDURE GetOrdersAboveAmount(
    IN cust_id INT,
    IN min_amount DECIMAL(10,2)
)
BEGIN
    SELECT *
    FROM orders
    WHERE customer_id = cust_id
      AND amount >= min_amount;
END //

DELIMITER ;
```

Call:

```
CALL GetOrdersAboveAmount(1, 300);
```

✔ Returns orders ≥ 300 for customer 1.

Step 5: Stored Procedure with OUT Parameter

OUT parameters return values.

Example: Get Total Spending of Customer

```
DELIMITER //

CREATE PROCEDURE GetTotalSpending(
    IN cust_id INT,
    OUT total DECIMAL(10,2)
)
BEGIN
    SELECT SUM(amount)
    INTO total
    FROM orders
    WHERE customer_id = cust_id;
END //

DELIMITER ;
```

Call It:

```
CALL GetTotalSpending(1, @total_spent);
SELECT @total_spent;
```

✔ `@total_spent` stores the returned value.

Step 6: Using Variables Inside Procedure

```
DELIMITER //

CREATE PROCEDURE CustomerLevel(IN cust_id INT)
BEGIN
    DECLARE total DECIMAL(10,2);

    SELECT SUM(amount)
```

```
    INTO total
    FROM orders
    WHERE customer_id = cust_id;

    IF total > 500 THEN
        SELECT 'Premium Customer' AS Status;
    ELSE
        SELECT 'Regular Customer' AS Status;
    END IF;
END //

DELIMITER ;
```

Call:

```
CALL CustomerLevel(1);
```

✔ Demonstrates conditional logic.

Step 7: Procedure with INSERT (Data Modification)

Example: Add New Customer

```
DELIMITER //

CREATE PROCEDURE AddCustomer(
    IN p_id INT,
    IN p_name VARCHAR(100),
    IN p_email VARCHAR(100),
    IN p_country VARCHAR(50)
)
BEGIN
    INSERT INTO customers
    VALUES (p_id, p_name, p_email, p_country);
END //

DELIMITER ;
```

Call:

```
CALL AddCustomer(3, 'Mike', 'mike@email.com', 'Canada');
```

✔ Adds new row.

Step 8: Procedure with Transaction Handling

```
DELIMITER //

CREATE PROCEDURE TransferOrder(
    IN orderId INT,
    IN newCustomerId INT
)
BEGIN
    START TRANSACTION;

    UPDATE orders
    SET customer_id = newCustomerId
    WHERE order_id = orderId;

    COMMIT;
END //

DELIMITER ;
```

✔ Ensures safe update.

Step 9: Replace or Drop Procedure

Drop:

```
DROP PROCEDURE IF EXISTS GetAllCustomers;
```

View Existing Procedures:

```
SHOW PROCEDURE STATUS;
```

Parameter Types in Stored Procedures

Type	Description
IN	Input parameter
OUT	Output parameter
INOUT	Both input and output

Real-World Use Case Example

Business Rule: Place Order Procedure

Instead of writing multiple queries from application code:

```
DELIMITER //

CREATE PROCEDURE PlaceOrder(
    IN p_order_id INT,
    IN p_customer_id INT,
    IN p_amount DECIMAL(10,2)
)
BEGIN
    INSERT INTO orders(order_id, customer_id, order_date, amount)
    VALUES(p_order_id, p_customer_id, CURDATE(), p_amount);

    UPDATE customers
    SET country = country
    WHERE customer_id = p_customer_id;
END //

DELIMITER ;
```

✔ All business logic stays inside database.

Stored Procedure vs View (Quick Comparison)

Feature	View	Stored Procedure
Stores Data	☐	☐
Contains Logic	☐	☐
Parameters	☐	☐
DML (Insert/Update/Delete)	☐	☐
Conditional Logic	☐	☐

Advantages of Stored Procedures

✔ Reusable
✔ Better performance
✔ Secure (grant EXECUTE only)
✔ Reduce duplicate SQL
✔ Centralized business logic