Question: 1

Let's create some stored procedures along with tables and then demonstrate how to call these procedures and observe the output.

First, let's create two tables: **Students** and **Courses**.

1. **Students Table**:
   CREATE TABLE Students (

   StudentID INT PRIMARY KEY AUTO_INCREMENT,

   Name VARCHAR (100),

   Age INT,

   CourseID INT,

   FOREIGN KEY (CourseID) REFERENCES Courses (CourseID)

   );

2. **Courses Table**:
   CREATE TABLE Courses (

   CourseID INT PRIMARY KEY AUTO_INCREMENT,

   CourseName VARCHAR (100)

   );

3. **Sample Data**:
   INSERT INTO Courses (CourseName) VALUES ('Math'), ('Physics'), ('History');

   INSERT INTO Students (Name, Age, CourseID) VALUES ('Alice', 20, 1), ('Bob', 22, 2), ('Charlie', 21, 3);

4. **Procedure to Get Students by Course**:
   DELIMITER //

   CREATE PROCEDURE GetStudentsByCourse (

   IN p_CourseName VARCHAR (100)

)

   BEGIN

```
    SELECT s.Name, s.Age, c.CourseName
    FROM Students s
    INNER JOIN Courses c ON s.CourseID = c.CourseID
    WHERE c.CourseName = p_CourseName;
END//
DELIMITER;
```

5. **Procedure to Insert Student**:

```
DELIMITER //

CREATE PROCEDURE InsertStudent (
    IN p_Name VARCHAR (100),
    IN p_Age INT,
    IN p_CourseID INT
)
BEGIN
    INSERT INTO Students (Name, Age, CourseID)
    VALUES (p_Name, p_Age, p_CourseID);
END//
DELIMITER;
```

6. **Calling Procedure to Get Students by Course**:

CALL GetStudentsByCourse('Physics');

Output:

```
+-------+-----+------------+
| Name  | Age | CourseName |
+-------+-----+------------+
| Bob   |  22 | Physics    |
+-------+-----+------------+
```

7. **Calling Procedure to Insert Student**:
   CALL InsertStudent('Dave', 23, 1);

   No output, but it inserts the new student into the **Students** table.

   These examples illustrate how stored procedures can be used to encapsulate logic and perform specific tasks within a database management system, allowing for better organization, security, and reusability of code.

Question: 2

Let's consider a simple scenario where we have a table named **Employees**.

1. **Employees Table**:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR (100),
   Salary DECIMAL (10, 2),
   DepartmentID INT
);
```

2. **Sample Data**:
   INSERT INTO Employees (Name, Salary, DepartmentID) VALUES ('John', 50000, 1), ('Alice', 60000, 2), ('Bob', 70000, 3);

3. **Procedure to Get Employee Details by ID**:

```
DELIMITER //

CREATE PROCEDURE GetEmployeeByID (
    IN p_EmployeeID INT
)
BEGIN
    SELECT * FROM Employees WHERE EmployeeID = p_EmployeeID;
END//

DELIMITER ;
```

Calling Procedure to Get Employee Details by ID:

CALL GetEmployeeByID(1);

Output:

```
+------------+-------+--------+-------------+
| EmployeeID | Name  | Salary | DepartmentID |
+------------+-------+--------+-------------+
```

| 1 | John | 50000.00 | 1 |

+----------+-------+--------+-------------+

4. **Procedure to Insert Employee**:
   DELIMITER //

   CREATE PROCEDURE InsertEmployee (

   IN p_Name VARCHAR(100),

   IN p_Salary DECIMAL(10, 2),

   IN p_DepartmentID INT

)

   BEGIN

   INSERT INTO Employees (Name, Salary, DepartmentID)

   VALUES (p_Name, p_Salary, p_DepartmentID);

   END//

   DELIMITER ;

   Calling Procedure to Insert Employee:

   CALL InsertEmployee('Sarah', 55000, 1);

   No output, but it inserts the new employee into the **Employees** table.

5. **Procedure to Update Employee Salary**:
   DELIMITER //

   CREATE PROCEDURE UpdateEmployeeSalary (

   IN p_EmployeeID INT,

   IN p_NewSalary DECIMAL(10, 2)

)

   BEGIN

   UPDATE Employees SET Salary = p_NewSalary WHERE EmployeeID = p_EmployeeID;

   END//

DELIMITER;

Calling Procedure to Update Employee Salary:

CALL UpdateEmployeeSalary(2, 65000);

No output, but it updates Alice's salary to $65,000.

6. **Procedure to Delete Employee**:
   DELIMITER //

   CREATE PROCEDURE DeleteEmployee (

   IN p_EmployeeID INT

)

   BEGIN

   DELETE FROM Employees WHERE EmployeeID = p_EmployeeID;

   END//

   DELIMITER ;

   Calling Procedure to Delete Employee:

   CALL DeleteEmployee(3);

   No output, but it deletes Bob's record from the **Employees** table.