Lab53:

Stored Procedures:

Create a Store procedure:

Creating a stored procedure to insert a new employee record.

```
-- Creating a stored procedure to insert a new employee record
CREATE PROCEDURE InsertEmployee
    @EmpName NVARCHAR(100),
    @EmpSalary DECIMAL(18, 2),
    @EmpGender CHAR(1)
AS
BEGIN
    INSERT INTO dbo.Employee (EmpName, EmpSalary, EmpGender)
    VALUES (@EmpName, @EmpSalary, @EmpGender);
END;
```

procedure named I nsertEmployee. accepts three parameters: @EmpName, @EmpSalary, and @EmpGender. It inserts a new record into the Employee table using the provided parameter values.

Modify a Stored Procedure:

Modifying the InsertEmployee stored procedure to include default values for parameters.

```
-- Modifying the InsertEmployee stored procedure to include default val
ALTER PROCEDURE InsertEmployee
    @EmpName NVARCHAR(100),
    @EmpSalary DECIMAL(18, 2) = 0,
    @EmpGender CHAR(1) = 'U'
AS
BEGIN
    INSERT INTO dbo.Employee (EmpName, EmpSalary, EmpGender)
    VALUES (@EmpName, @EmpSalary, @EmpGender);
END;
```

This script alters the existing InsertEmployee stored procedure. It adds default values for @EmpSalary and @EmpGender parameters.

Delete a Stored Procedure:

Deleting the InsertEmployee stored procedure.

```sql
-- Deleting the InsertEmployee stored procedure
DROP PROCEDURE InsertEmployee;
```

This script deletes the InsertEmployee stored procedure from the database.

Execute a Stored Procedure:

Executing the InsertEmployee stored procedure.

```sql
-- Executing the InsertEmployee stored procedure
EXEC InsertEmployee @EmpName = 'Alice', @EmpSalary = 75000, @EmpGender = 'F';
```

This script executes the InsertEmployee stored procedure. It provides values for the @EmpName, @EmpSalary, and @EmpGender parameters.

Return Data from a Stored Procedure:

Creating a stored procedure to retrieve employee details by ID.

```sql
-- Creating a stored procedure to retrieve employee details by ID
CREATE PROCEDURE GetEmployeeByID
    @EmployeeID INT
AS
BEGIN
    SELECT EmployeeID, EmpName, EmpSalary, EmpGender
    FROM dbo.Employee
    WHERE EmployeeID = @EmployeeID;
END;
```

This script creates a stored procedure named GetEmployeeByID. It retrieves employee details based on the provided @EmployeeID parameter.

Results Messages

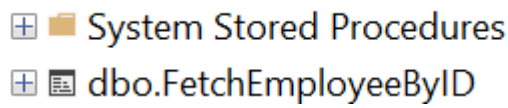
	EmployeeID	EmpName	EmpSalary	EmpGender
1	1	John Doe	50000.00	M

Rename a Stored Procedure:

Renaming the GetEmployeeByID stored procedure to FetchEmployeeByID.

```
-- Renaming the GetEmployeeByID stored procedure to FetchEmployeeByID
EXEC sp_rename 'GetEmployeeByID', 'FetchEmployeeByID';
```

This script renames the GetEmployeeByID stored procedure to FetchEmployeeByID.



Use Parameters in a Stored Procedure:

Creating a stored procedure with inpu.t and output parameters

```
CREATE PROCEDURE GetEmployeeName
    @EmployeeID INT,
    @EmpName NVARCHAR(100) OUTPUT
AS
BEGIN
    SELECT @EmpName = EmpName
    FROM dbo.Employee
    WHERE EmployeeID = @EmployeeID;
END;
```

This script creates a stored procedure named GetEmployeeName with an input parameter @EmployeeID and an output parameter @EmpName.

It assigns the employee name to the @EmpName output parameter based on the provided @EmployeeID.

Executing with parameter:

Executing the GetEmployeeName stored procedure with an input parameter.

following executes the GetEmployeeName stored procedure with an input parameter and captures the output value in the @EmpName variable.

```sql
-- Declaring a variable to hold the output value
DECLARE @EmpName NVARCHAR(100);

-- Executing the GetEmployeeName stored procedure with an input parameter
-- and capturing the output value in the @EmpName variable
EXEC GetEmployeeName @EmployeeID = 1, @EmpName = @EmpName OUTPUT;

-- Displaying the output value
SELECT @EmpName AS EmployeeName;
```

Results | Messages

	EmployeeName
1	John Doe