**Lab 1: Creating a Simple Stored Procedure**

**Task:**

Create a stored procedure named GetAllEmployees that retrieves all records from the Employees table.

**Solution:**

CREATE PROCEDURE GetAllEmployees

AS

BEGIN

SELECT \* FROM Employees

END

**Execution:**

EXEC GetAllEmployees

**Lab 2: Stored Procedure with Input Parameters**

**Task:**

Create a stored procedure named GetEmployeeByID that accepts an employee ID and retrieves employee details.

**Solution:**

CREATE PROCEDURE GetEmployeeByID

@EmpID INT

AS

BEGIN

SELECT \* FROM Employees WHERE EmployeeID = @EmpID

END

**Execution:**

EXEC GetEmployeeByID @EmpID = 1

**Lab 3: Stored Procedure with Output Parameters**

**Task:**

Create a stored procedure named GetEmployeeSalary that returns an employee's salary as an output parameter.

**Solution:**

CREATE PROCEDURE GetEmployeeSalary

@EmpID INT,

@Salary DECIMAL(10,2) OUTPUT

AS

BEGIN

SELECT @Salary = Salary FROM Employees WHERE EmployeeID = @EmpID

END

**Execution:**

DECLARE @EmpSalary DECIMAL(10,2)

EXEC GetEmployeeSalary @EmpID = 1, @Salary = @EmpSalary OUTPUT

PRINT @EmpSalary

**Lab 4: Stored Procedure with Multiple Parameters**

**Task:**

Create a stored procedure GetEmployeesByDepartment that retrieves employees based on department ID and job title.

**Solution:**

CREATE PROCEDURE GetEmployeesByDepartment

@DeptID INT,

@JobTitle NVARCHAR(50)

AS

BEGIN

SELECT \* FROM Employees WHERE DepartmentID = @DeptID AND JobTitle = @JobTitle

END

**Execution:**

EXEC GetEmployeesByDepartment @DeptID = 2, @JobTitle = 'Manager'

**Lab 5: Stored Procedure with IF...ELSE**

**Task:**

Create a stored procedure CheckEmployeeStatus that checks if an employee exists. If they do, return "Exists", otherwise "Not Found".

**Solution:**

CREATE PROCEDURE CheckEmployeeStatus

@EmpID INT,

@Status NVARCHAR(20) OUTPUT

AS

BEGIN

IF EXISTS (SELECT 1 FROM Employees WHERE EmployeeID = @EmpID)

SET @Status = 'Exists'

ELSE

SET @Status = 'Not Found'

END

**Execution:**

DECLARE @Status NVARCHAR(20)

EXEC CheckEmployeeStatus @EmpID = 1, @Status = @Status OUTPUT

PRINT @Status

**Lab 6: Stored Procedure with Transactions**

**Task:**

Create a stored procedure TransferSalary to transfer salary from one employee to another, using a transaction.

**Solution:**

CREATE PROCEDURE TransferSalary

@FromEmpID INT,

@ToEmpID INT,

@Amount DECIMAL(10,2)

AS

BEGIN

BEGIN TRANSACTION

UPDATE Employees SET Salary = Salary - @Amount WHERE EmployeeID = @FromEmpID

UPDATE Employees SET Salary = Salary + @Amount WHERE EmployeeID = @ToEmpID

IF @@ERROR <> 0

BEGIN

ROLLBACK TRANSACTION

END

ELSE

BEGIN

COMMIT TRANSACTION

END

END

**Execution:**

EXEC TransferSalary @FromEmpID = 1, @ToEmpID = 2, @Amount = 500

**Lab 7: Stored Procedure with TRY...CATCH**

**Task:**

Create a stored procedure DivideNumbers that divides two numbers and handles errors using TRY...CATCH.

**Solution:**

CREATE PROCEDURE DivideNumbers

@Num1 INT,

@Num2 INT,

@Result DECIMAL(10,2) OUTPUT

AS

BEGIN

BEGIN TRY

SET @Result = @Num1 / @Num2

END TRY

BEGIN CATCH

PRINT 'Error: Division by zero is not allowed'

END CATCH

END

**Execution:**

DECLARE @Res DECIMAL(10,2)

EXEC DivideNumbers @Num1 = 10, @Num2 = 0, @Result = @Res OUTPUT

PRINT @Res

**Lab 8: Stored Procedure Returning a Table**

**Task:**

Create a stored procedure GetHighSalaryEmployees that returns employees with salary greater than a given amount.

**Solution:**

CREATE PROCEDURE GetHighSalaryEmployees

@MinSalary DECIMAL(10,2)

AS

BEGIN

SELECT \* FROM Employees WHERE Salary > @MinSalary

END

**Execution:**

EXEC GetHighSalaryEmployees @MinSalary = 50000

**Lab 9: Stored Procedure for Dynamic SQL**

**Task:**

Create a stored procedure SearchEmployees that dynamically filters employees based on provided conditions.

**Solution:**

CREATE PROCEDURE SearchEmployees

@ColumnName NVARCHAR(50),

@SearchValue NVARCHAR(100)

AS

BEGIN

DECLARE @SQLQuery NVARCHAR(MAX)

SET @SQLQuery = 'SELECT \* FROM Employees WHERE ' + @ColumnName + ' = ''' + @SearchValue + ''''

EXEC sp\_executesql @SQLQuery

END

**Execution:**

EXEC SearchEmployees @ColumnName = 'JobTitle', @SearchValue = 'Developer'

**Lab 10: Stored Procedure with Cursor**

**Task:**

Create a stored procedure ListEmployees that uses a cursor to print employee names one by one.

**Solution:**

CREATE PROCEDURE ListEmployees

AS

BEGIN

DECLARE @EmpName NVARCHAR(100)

DECLARE EmployeeCursor CURSOR FOR

SELECT Name FROM Employees

OPEN EmployeeCursor

FETCH NEXT FROM EmployeeCursor INTO @EmpName

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT @EmpName

FETCH NEXT FROM EmployeeCursor INTO @EmpName

END

CLOSE EmployeeCursor

DEALLOCATE EmployeeCursor

END

**Execution:**

EXEC ListEmployees