-- SQL Stored Procedures Exercise File

-- Based on Employee Management System schema

-- Table Setup

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

-- Sample Data

INSERT INTO Departments VALUES

(1, 'HR'), (2, 'Finance'), (3, 'IT'), (4, 'Marketing');

INSERT INTO Employees VALUES

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

-- Exercise 1: Insert Procedure

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

GO

-- Output: Adds a new employee record

EXEC sp\_InsertEmployee 'Alice', 'Brown', 3, 6200.00, '2022-05-01';

-- Exercise 2: Modify to Include Salary in Output

ALTER PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

SELECT \* FROM Employees WHERE DepartmentID = @DepartmentID;

END;

GO

-- Exercise 3: Drop the Procedure

DROP PROCEDURE IF EXISTS sp\_InsertEmployee;

-- Output: Procedure dropped

-- Exercise 4: Execute to See Results

EXEC sp\_InsertEmployee 'David', 'White', 3, 6300.00, '2023-01-10';

-- Output: Employees in DepartmentID 3 listed

-- Exercise 5: Count Employees in Department

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

-- Output: TotalEmployees for a given department

EXEC sp\_CountEmployeesByDepartment 3;

-- Exercise 6: Total Salary with Output Parameter

CREATE PROCEDURE sp\_TotalSalaryByDepartment

@DepartmentID INT,

@TotalSalary DECIMAL(10,2) OUTPUT

AS

BEGIN

SELECT @TotalSalary = SUM(Salary)

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

-- Output:

DECLARE @SalaryTotal DECIMAL(10,2);

EXEC sp\_TotalSalaryByDepartment 3, @SalaryTotal OUTPUT;

SELECT @SalaryTotal AS TotalSalary;

-- Exercise 7: Update Salary

CREATE PROCEDURE sp\_UpdateEmployeeSalary

@EmployeeID INT,

@NewSalary DECIMAL(10,2)

AS

BEGIN

UPDATE Employees

SET Salary = @NewSalary

WHERE EmployeeID = @EmployeeID;

END;

GO

-- Output: Updates salary of given EmployeeID

EXEC sp\_UpdateEmployeeSalary 1, 5500.00;

-- Exercise 8: Bonus for Department

CREATE PROCEDURE sp\_GiveBonus

@DepartmentID INT,

@BonusAmount DECIMAL(10,2)

AS

BEGIN

UPDATE Employees

SET Salary = Salary + @BonusAmount

WHERE DepartmentID = @DepartmentID;

END;

GO

-- Output: All employees in department get a salary increase

EXEC sp\_GiveBonus 2, 300.00;

-- Exercise 9: Transactional Update

CREATE PROCEDURE sp\_UpdateSalaryWithTransaction

@EmployeeID INT,

@NewSalary DECIMAL(10,2)

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

UPDATE Employees

SET Salary = @NewSalary

WHERE EmployeeID = @EmployeeID;

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

SELECT ERROR\_MESSAGE() AS ErrorMessage;

END CATCH

END;

GO

-- Output: Salary updated, or rollback if error

EXEC sp\_UpdateSalaryWithTransaction 4, 5800.00;

-- Exercise 10: Dynamic SQL

CREATE PROCEDURE sp\_GetEmployeesByFilter

@ColumnName NVARCHAR(50),

@Value NVARCHAR(100)

AS

BEGIN

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'SELECT \* FROM Employees WHERE ' + QUOTENAME(@ColumnName) + ' = @Param';

EXEC sp\_executesql @SQL, N'@Param NVARCHAR(100)', @Param = @Value;

END;

GO

-- Output: Employees matching filter

EXEC sp\_GetEmployeesByFilter 'DepartmentID', '3';

-- Exercise 11: Error Handling

CREATE PROCEDURE sp\_UpdateSalaryWithErrorHandling

@EmployeeID INT,

@NewSalary DECIMAL(10,2)

AS

BEGIN

BEGIN TRY

UPDATE Employees

SET Salary = @NewSalary

WHERE EmployeeID = @EmployeeID;

END TRY

BEGIN CATCH

SELECT ERROR\_MESSAGE() AS ErrorMessage;

END CATCH

END;

GO

-- Output: Error message if update fails

EXEC sp\_UpdateSalaryWithErrorHandling 999, 6500.00;