```
CREATE DATABASE storedprocedure;

USE storedprocedure;

--Best Practices for SQL Stored Procedures:
--1. Design and Structure
--Modular Design:

CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   employee_name NVARCHAR(255),
   salary DECIMAL(10, 2)
);

INSERT INTO employees (employee_id, employee_name, salary)
VALUES
(1, 'Alice Johnson', 50000.00),
(2, 'Bob Smith', 60000.00),
(3, 'Charlie Brown', 55000.00);

-- Step 3: Create the stored procedure
        CREATE PROCEDURE UpdateEmployeeSalary
   @EmployeeID INT,
   @NewSalary DECIMAL(10, 2)
        AS
        BEGIN
   -- Update the salary of the specified employee
   UPDATE employees
   SET salary = @NewSalary
   WHERE employee_id = @EmployeeID;
        END;


--2. Parameters and Input Handling
CREATE PROCEDURE GetEmployeeDetails(@EmployeeID INT)
AS
BEGIN
SELECT * FROM employees WHERE employee_id = @EmployeeID;
END;


--Validate Input Parameters:
CREATE PROCEDURE GetEmployeeDetail(@EmployeeID INT)
AS
```

```
BEGIN
IF @EmployeeID IS NULL
BEGIN
RAISERROR('EmployeeID cannot be NULL', 16, 1);
RETURN;
END
SELECT * FROM employees WHERE employee_id = @EmployeeID;
END;



--3. Error Handling
-- Step 1: Create the Employees table
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,        -- Employee unique identifier
    EmployeeName NVARCHAR(255),        -- Employee's name
    Salary DECIMAL(10, 2)              -- Employee's salary
);

-- Step 2: Insert sample data into the Employees table
INSERT INTO Employee (EmployeeID, EmployeeName, Salary)
VALUES
(1, 'Alice Johnson', 50000.00),
(2, 'Bob Smith', 60000.00),
(3, 'Charlie Brown', 55000.00);

-- Step 3: Create the stored procedure with error handling
CREATE PROCEDURE UpdateEmployeeSalaries
    @EmployeeID INT,            -- Employee ID whose salary needs to be updated
    @NewSalary DECIMAL(10, 2)  -- New salary value for the employee
AS
BEGIN
    BEGIN TRY
        -- Update the salary of the specified employee
        UPDATE Employee
        SET Salary = @NewSalary
        WHERE EmployeeID = @EmployeeID;
    END TRY
    BEGIN CATCH
        -- Declare a variable to capture the error message
        DECLARE @ErrorMessage NVARCHAR(4000);

        -- Capture the error message
        SET @ErrorMessage = ERROR_MESSAGE();
        -- Raise an error with severity level 16 (user-defined error)
        RAISERROR(@ErrorMessage, 16, 1);
    END CATCH;
END;


--Log Errors:
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,       -- Unique identifier for the employee
    EmployeeName NVARCHAR(255),       -- Name of the employee
    Salary DECIMAL(10, 2)             -- Salary of the employee
);


INSERT INTO Employee (EmployeeID, EmployeeName, Salary)
VALUES
(1, 'Alice Johnson', 50000.00),
(2, 'Bob Smith', 60000.00),
(3, 'Charlie Brown', 55000.00);

-- Step 3: Create the ErrorLog table to capture errors
CREATE TABLE ErrorLog (
    ErrorID INT IDENTITY(1,1) PRIMARY KEY,
    ErrorMessage NVARCHAR(4000),
    ErrorDate DATETIME
);


CREATE PROCEDURE UpdateEmployeesSalary
    @EmployeeID INT,           -- Employee ID whose salary needs to be updated
    @NewSalary DECIMAL(10, 2)  -- New salary value for the employee
AS
BEGIN
    BEGIN TRY
        -- Attempt to update the salary of the specified employee
        UPDATE Employee
        SET Salary = @NewSalary
        WHERE EmployeeID = @EmployeeID;
    END TRY
    BEGIN CATCH
        -- Capture the error message and log it into the ErrorLog table
        INSERT INTO ErrorLog (ErrorMessage, ErrorDate)
        VALUES (ERROR_MESSAGE(), GETDATE());
        -- Raise the error to notify the caller
        RAISERROR(ERROR_MESSAGE(), 16, 1);
    END CATCH;
END;



--4. Performance Optimization
CREATE PROCEDURE UpdateEmployeesSalary(@EmployeeID INT, @NewSalary DECIMAL)
AS
BEGIN
SET NOCOUNT ON;
UPDATE employees
SET salary = @NewSalary
WHERE employee_id = @EmployeeID;
END;


--5. Security Practices
--Encrypt Stored Procedures:

CREATE TABLE Employ (
    EmployeeID INT PRIMARY KEY,       -- Unique identifier for each employee
    EmployeeName NVARCHAR(255),       -- Name of the employee
    Salary DECIMAL(10, 2),            -- Salary of the employee
    DepartmentID INT                  -- ID of the department the employee belongs to
);


INSERT INTO Employ (EmployeeID, EmployeeName, Salary, DepartmentID)
VALUES
(1, 'Alice Johnson', 50000.00, 101),
(2, 'Bob Smith', 60000.00, 102),
(3, 'Charlie Brown', 55000.00, 101),
(4, 'Diana Prince', 65000.00, 103),
(5, 'Eve Adams', 70000.00, 101);


CREATE PROCEDURE UpdateEmployeeSalaries
WITH ENCRYPTION
AS
BEGIN
UPDATE employ
```

```
SET salary = salary * 1.1
WHERE department_id = @DepartmentID;
END;
```