Employee Management System SQL Exercises

Database Schema

The following schema defines the structure for an Employee Management System:

Departments Table

```
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(100)
);
```

Employees Table

```
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

The following sample data can be used for testing:

Departments Sample Data

```
INSERT INTO Departments (DepartmentID, DepartmentName) VALUES
(1, 'HR'),
(2, 'Finance'),
(3, 'IT'),
(4, 'Marketing');
```

Employees Sample Data

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID, Salary,
JoinDate) 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');

Exercises

Exercise 1: Create a Stored Procedure

Goal: Create a stored procedure to retrieve employee details by department.

Steps:

1. Define the stored procedure with a parameter for DepartmentID.

2. Write the SQL query to select employee details based on the DepartmentID.

3. Create a stored procedure named `sp_InsertEmployee` with the following code:

```
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;
```

Exercise 2: Modify a Stored Procedure

Goal: Modify the stored procedure to include employee salary in the result.

Steps:

1. Open the existing stored procedure.

2. Add the Salary column to the SELECT statement.

3. Save the changes by executing the Stored procedure content.

Exercise 3: Delete a Stored Procedure

Goal: Delete the stored procedure created in Exercise 1.

Steps:

1. Write the SQL command to delete the stored procedure.

2. Execute the command.

Exercise 4: Execute a Stored Procedure

Goal: Execute the stored procedure to retrieve employee details for a specific department.

Steps:

1. Write the SQL command to execute the stored procedure with a DepartmentID parameter.

2. Execute the command and review the results.

Exercise 5: Return Data from a Stored Procedure

Goal: Create a stored procedure that returns the total number of employees in a department.

Steps:

1. Define the stored procedure with a parameter for DepartmentID.

2. Write the SQL query to count the number of employees in the specified department.

3. Save the stored procedure by executing the Stored procedure content.

Exercise 6: Use Output Parameters in a Stored Procedure

Goal: Create a stored procedure that returns the total salary of employees in a department using an output parameter.

Steps:

1. Define the stored procedure with a parameter for DepartmentID and an output parameter for total salary.

2. Write the SQL query to calculate the total salary of employees in the specified department.

3. Save the stored procedure by executing the Stored procedure content.

Exercise 7: Create a Stored Procedure with Multiple Parameters

Goal: Create a stored procedure to update employee salary.

Steps:
1. Open SQL Server Management Studio (SSMS).
2. Connect to your database.
3. Create a stored procedure named `sp_UpdateEmployeeSalary` to update employee salary
4. Execute the stored procedure with the following code:

EXEC sp_UpdateEmployeeSalary 1, 5500.00;

Exercise 8: Create a Stored Procedure with Conditional Logic

Goal: Create a stored procedure to give a bonus to employees based on their department.

Steps:
1. Open SQL Server Management Studio (SSMS).
2. Connect to your database.
3. Create a stored procedure named `sp_GiveBonus` to give a bonus to employees based on their department.
4. Execute the stored procedure with the following code:

EXEC sp_GiveBonus 1, 500.00;

Exercise 9: Use Transactions in a Stored Procedure

Goal: Create a stored procedure that updates employee salaries and uses a transaction to ensure data integrity.

Steps:

1. Define the stored procedure with parameters for EmployeeID and new Salary.

2. Write the SQL query to update the employee salary.

3. Use a transaction to ensure data integrity.

4. Save the stored procedure by executing the Stored procedure content.

Exercise 10: Use Dynamic SQL in a Stored Procedure

Goal: Create a stored procedure that uses dynamic SQL to retrieve employee details based on a flexible filter.

Steps:

1. Define the stored procedure with parameters for filter column and filter value.

2. Write the dynamic SQL query to retrieve employee details based on the filter.

3. Save the stored procedure by executing the Stored procedure content.

Exercise 11: Handle Errors in a Stored Procedure

Goal: Create a stored procedure that handles errors and returns a custom error message.

Steps:

1. Define the stored procedure with parameters for EmployeeID and new Salary.

2. Write the SQL query to update the employee salary.

3. Use TRY...CATCH to handle errors and return a custom error message.

4. Save the stored procedure by executing the Stored procedure content.