# **Employee Management Application**

# **Overview**

The Employee Management Application is designed to facilitate the management of employee details and salary information using a SQL database. The application provides features to fetch, add, and filter employee data using a user-friendly interface with a data grid display.

# **Features**

#### **1. List Employee Details**

* Clicking the "List Employee Details" button retrieves all employee information along with salary details.
* The data is displayed on a data grid on the UI.
* The data is sourced from the Employee and EmployeeSalary tables from EmployeeRecords Database.

#### **2. Add Employee**

* Clicking the "Add Employee Details" button opens an employee registration window.
* The registration form enforces data validation, preventing submission of invalid or incomplete data.
* Upon successful validation, employee details are saved in the Employee and EmployeeSalary tables.

#### **3. Filtering Employee Data**

* The data grid allows filtering based on Employee Name and Employee Title.
* The data grid refreshes automatically when a new employee is added.

**4. View Titles and Salary Ranges**

* Clicking the "Titles" button displays a list of distinct employee titles.
* The display includes the minimum and maximum salary applicable for each title.
* The data grid refreshes automatically when a new title is added.

#### **Database Schema**

# **Employee Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| EmployeeID | INT | PRIMARY KEY, IDENTITY(1,1) |
| Name | VARCHAR(100) | NOT NULL |
| SSN | VARCHAR(11) | UNIQUE, NOT NULL |
| DOB | DATE | NOT NULL |
| Address | VARCHAR(255) | NOT NULL |
| City | VARCHAR(100) | NOT NULL |
| State | VARCHAR(50) | NOT NULL |
| Zip | VARCHAR(10) | NOT NULL |
| Phone | VARCHAR(15) | NOT NULL |
| JoinDate | DATE | NOT NULL |
| ExitDate | DATE | NULL |

# **EmployeeSalary Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| SalaryID | INT | PRIMARY KEY, IDENTITY(1,1) |
| EmployeeID | INT | FOREIGN KEY REFERENCES Employee(EmployeeID) |
| FromDate | DATE | NOT NULL |
| ToDate | DATE | NULL |
| Title | VARCHAR(100) | NOT NULL |
| Salary | DECIMAL(10,2) | NOT NULL |

#### **Initial Data Setup**

#### **Summary of the Database Script**

The provided database script performs the following actions:

1. **Database Creation**
   * Creates a new database named EmployeeRecords.
   * Switches to the EmployeeRecords database.
2. **Employee Table Creation**
   * Defines an Employee table with columns such as EmployeeID (Primary Key with Identity), Name, SSN (Unique), DOB, Address, City, State, Zip, Phone, JoinDate, and ExitDate.
3. **EmployeeSalary Table Creation**
   * Defines an EmployeeSalary table with columns such as SalaryID (Primary Key with Identity), EmployeeID (Foreign Key referencing EmployeeID in the Employee table), FromDate, ToDate, Title, and Salary.
4. **Identity Reseeding**
   * Resets the identity seed for both Employee and EmployeeSalary tables using DBCC CHECKIDENT to ensure sequential numbering from 1.
5. **Sample Data Insertion**
   * Uses a loop and temporary table to insert 100 sample employee records with random names and details into the Employee table.
   * The script likely includes salary records associated with each employee in the EmployeeSalary table.

#### **Purpose of the Script**

This script sets up the foundational structure for an Employee Management system, ensuring:

* Proper relational integrity between employees and their salaries.
* Unique identification for each employee.
* Sample data to test and validate the system functionalities.

#### **UI Design Considerations**

* **Data Grid**: Displays employee details and salary information.
* **Registration Form**: Includes validation for all required fields.
* **Filter Options**: Enables search by employee name or title.
* **Automatic Refresh**: Data grids update dynamically when new employees or titles are added.

#### **Summary**

This Employee Management Application streamlines employee data handling, ensuring proper storage and retrieval while maintaining data integrity through enforced constraints and validations.