1. **Database Schema**

* **Departments Table**

CREATE TABLE Departments ( DepartmentID INT PRIMARY KEY IDENTITY(1,1), DepartmentName NVARCHAR(100) NOT NULL );

* **Employees Table**

CREATE TABLE Employees ( EmployeeID INT PRIMARY KEY IDENTITY(1,1), Name NVARCHAR(100) NOT NULL, DepartmentID INT NOT NULL, HireDate DATE NOT NULL, FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID) );

* **Salaries Table**

CREATE TABLE Salaries ( EmployeeID INT PRIMARY KEY, BaseSalary DECIMAL(10, 2) NOT NULL, Bonus DECIMAL(10, 2) NOT NULL, Deductions DECIMAL(10, 2) NOT NULL, FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID) );

* **SalaryHistory Table (For Bonus Task)**

-- Create SalaryHistory Table (Bonus Task) CREATE TABLE SalaryHistory ( HistoryID INT PRIMARY KEY IDENTITY(1,1), EmployeeID INT NOT NULL, OldBaseSalary DECIMAL(10, 2), OldBonus DECIMAL(10, 2), OldDeductions DECIMAL(10, 2), NewBaseSalary DECIMAL(10, 2), NewBonus DECIMAL(10, 2), NewDeductions DECIMAL(10, 2), UpdatedAt DATETIME DEFAULT GETDATE(), FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID) );

1. **SQL Queries**

* **List all employees with their department names**

SELECT e.EmployeeID, e.Name, d.DepartmentName, e.HireDate FROM Employees e JOIN Departments d ON e.DepartmentID = d.DepartmentID;

* **Calculate Net Salary for Each Employee**

SELECT e.EmployeeID, e.Name, (s.BaseSalary + s.Bonus - s.Deductions) AS NetSalary FROM Employees e JOIN Salaries s ON e.EmployeeID = s.EmployeeID;

* **Identify Department with the Highest Average Salary**

SELECT TOP 1 d.DepartmentName, AVG(s.BaseSalary + s.Bonus - s.Deductions) AS AvgSalary FROM Salaries s JOIN Employees e ON s.EmployeeID = e.EmployeeID JOIN Departments d ON e.DepartmentID = d.DepartmentID GROUP BY d.DepartmentName ORDER BY AvgSalary DESC;

1. **Stored Procedure**

* **Add Employee**

CREATE PROCEDURE AddEmployee @empName NVARCHAR(100), @deptID INT, @hireDate DATE AS BEGIN IF NOT EXISTS (SELECT 1 FROM Departments WHERE DepartmentID = @deptID) BEGIN RAISERROR ('Invalid DepartmentID', 16, 1); RETURN; END INSERT INTO Employees (Name, DepartmentID, HireDate) VALUES (@empName, @deptID, @hireDate); END;

* **Update Salary**

CREATE PROCEDURE UpdateSalary @empID INT, @newBaseSalary DECIMAL(10, 2), @newBonus DECIMAL(10, 2), @newDeductions DECIMAL(10, 2) AS BEGIN IF NOT EXISTS (SELECT 1 FROM Employees WHERE EmployeeID = @empID) BEGIN RAISERROR ('Invalid EmployeeID', 16, 1); RETURN; END DECLARE @oldBaseSalary DECIMAL(10, 2), @oldBonus DECIMAL(10, 2), @oldDeductions DECIMAL(10, 2); SELECT @oldBaseSalary = BaseSalary, @oldBonus = Bonus, @oldDeductions = Deductions FROM Salaries WHERE EmployeeID = @empID; INSERT INTO SalaryHistory (EmployeeID, OldBaseSalary, OldBonus, OldDeductions, NewBaseSalary, NewBonus, NewDeductions) VALUES (@empID, @oldBaseSalary, @oldBonus, @oldDeductions, @newBaseSalary, @newBonus, @newDeductions); UPDATE Salaries SET BaseSalary = @newBaseSalary, Bonus = @newBonus, Deductions = @newDeductions WHERE EmployeeID = @empID; END;

* **Calculate Payroll**

CREATE PROCEDURE CalculatePayroll @deptID INT = NULL AS BEGIN IF @deptID IS NULL BEGIN SELECT SUM(BaseSalary + Bonus - Deductions) AS TotalPayroll FROM Salaries; END ELSE BEGIN SELECT SUM(BaseSalary + Bonus - Deductions) AS TotalPayroll FROM Salaries s JOIN Employees e ON s.EmployeeID = e.EmployeeID WHERE e.DepartmentID = @deptID; END END;

1. **Views**

* **Employee Salary View**

CREATE VIEW EmployeeSalaryView AS SELECT e.EmployeeID, e.Name, d.DepartmentName, s.BaseSalary, s.Bonus, s.Deductions, (s.BaseSalary + s.Bonus - s.Deductions) AS NetSalary FROM Employees e JOIN Departments d ON e.DepartmentID = d.DepartmentID JOIN Salaries s ON e.EmployeeID = s.EmployeeID;

* **High Earner View**

CREATE VIEW HighEarnerView AS SELECT e.EmployeeID, e.Name, d.DepartmentName, (s.BaseSalary + s.Bonus - s.Deductions) AS NetSalary FROM Employees e JOIN Salaries s ON e.EmployeeID = s.EmployeeID JOIN Departments d ON e.DepartmentID = d.DepartmentID WHERE (s.BaseSalary + s.Bonus - s.Deductions) > 50000;

1. **Bonus Tasks**

CREATE TRIGGER trg\_SalaryUpdate ON Salaries AFTER UPDATE AS BEGIN INSERT INTO SalaryHistory (EmployeeID, OldBaseSalary, OldBonus, OldDeductions, NewBaseSalary, NewBonus, NewDeductions) SELECT i.EmployeeID, d.BaseSalary AS OldBaseSalary, d.Bonus AS OldBonus, d.Deductions AS OldDeductions, i.BaseSalary AS NewBaseSalary, i.Bonus AS NewBonus, i.Deductions AS NewDeductions FROM inserted i JOIN deleted d ON i.EmployeeID = d.EmployeeID; END;

1. **Example Data Population**

-- Insert Departments

INSERT INTO Departments (DepartmentName) VALUES ('HR'), ('Finance'), ('IT');

Insert Employees

INSERT INTO Employees (Name, DepartmentID, HireDate) VALUES ('John Doe', 1, '2023-01-15'), ('Jane Smith', 2, '2022-11-20');

Insert Salaries

INSERT INTO Salaries (EmployeeID, BaseSalary, Bonus, Deductions) VALUES (1, 50000, 5000, 2000), (2, 60000, 7000, 3000);

1. **Optimization and Indexing (Add Indexes)**

CREATE INDEX idx\_Employees\_DepartmentID ON Employees(DepartmentID);

CREATE INDEX idx\_Salaries\_EmployeeID ON Salaries(EmployeeID);