University of Management and Technology,
Lahore Campus

Semeter Project: Spring 2023

Project Title : Salary Management System

Database Systems (W6)

Submitted to: Waqar Ashiq

Submitted by

Ayesha Raza Toor (F2021065290)

Hassan Tahir (F2021065204)

Session

2021 – 2025

C-II Johar Town Lahore Pakistan

University of Management and Technology

Table of Contents

Chapter 1: Introduction to the Problem

1.1 Introduction

The salary management system is a software application or a set of processes designed to streamline and automate the management of employee salaries within an organization. It provides a centralized platform for handling various aspects of salary administration, including calculating salaries, generating payrolls, and maintaining salary-related data.

1.2 Purpose

The purpose of a salary management system is to efficiently handle the complex task of managing employee salaries, ensuring accuracy, timeliness, and compliance with relevant regulations. It replaces manual and paper-based processes with a digital solution, offering several benefits such as increased efficiency, reduced errors, improved data security, and enhanced reporting capabilities.

1.3 Objective

The primary objective of a salary management system is to simplify and optimize the overall salary management process within an organization. The key objectives include:

1. Automation: The system aims to automate repetitive and time-consuming tasks associated with salary management, such as calculating employee salaries, deductions, and taxes. This automation reduces human error and ensures accuracy in salary calculations.
2. Efficiency: The system strives to improve the efficiency of salary-related processes by providing a centralized platform where HR personnel can easily access and manage employee salary data. It eliminates the need for manual data entry, allowing HR staff to focus on more strategic activities.
3. Compliance: The system ensures compliance with legal and regulatory requirements related to salary management, including tax regulations, labor laws, and employee benefits. It helps HR departments stay updated with

changes in regulations and ensures that salaries and deductions are processed correctly.

4. Data Management: A salary management system helps in organizing and maintaining a vast amount of employee data, such as personal information, employment history, salary records, and tax details. It facilitates secure storage, retrieval, and analysis of salary-related information.
5. Reporting and Analysis: The system provides comprehensive reporting and analytical features, enabling HR managers to generate various salary-related reports, including payroll summaries, tax reports, and salary trends. These reports assist in decision-making and strategic planning.

   Overall, the salary management system aims to streamline salary administration, improve efficiency, enhance compliance, and provide accurate and timely salary-related information for both HR personnel and employees within an organization.

Chapter 2: Logical Database Design

2.1 Entities

- Employee
- Department
- Salary
- Deduction
- Allowance
- Tax

2.2 Attributes

1. Employee Table:
   - Employee ID (Primary Key)
   - First Name
   - Last Name
   - Date of Birth

- Gender
- Department ID (Foreign Key)
- Position
- Employment Start Date
- Employment End Date (if applicable)

2. Department Table:
    - Department ID (Primary Key)
    - Department Name

3. Salary Table:
    - Salary ID (Primary Key)
    - Employee ID (Foreign Key)
    - Base Salary
    - Effective Date (start date of the salary)

4. Deductions Table:
    - Deduction ID (Primary Key)
    - Employee ID (Foreign Key)
    - Deduction Type
    - Amount
    - Effective Date

5. Allowances Table:
    - Allowance ID (Primary Key)
    - Employee ID (Foreign Key)
    - Allowance Type
    - Amount
    - Effective Date

6. Tax Table:
    - Tax ID (Primary Key)
    - Employee ID (Foreign Key)
    - Tax Type
    - Amount
    - Effective Date

2.3 Relationships

1. Employee Table:
    - Department Table

      Relationship: Many-to-One (Many employees can belong to One Department)

      Cardinality: Mandary Many to Mandatory One

    - Salary Table

      Relationship: One-to-One (One employee can have One salary)

      Cardinality: Mandary One to Mandatory One

    - Deductions Table

      Relationship: One-to-Many (One employee can have multiple deductions)

      Cardinality: Mandatory One to Optional Many

    - Allowances Table

      Relationship: One-to-Many (One employee can have multiple allowances)

      Cardinality: Mandatory One to Optional Many

- Tax Table

  Relationship: One-to-Many (One employee can have multiple tax records)

  Cardinality: Mandatory One to Optional Many

2.4 Functional Dependenices and Normalization

2.4.1 Mentioned Functional Dependencies of attributes

1. **Employee table:**

   EmployeeID -> FirstName, LastName, DateOfBirth, Gender, Position, EmploymentStartDate, EmploymentEndDate, DepartmentID

2. **Department table:**

   DepartmentID -> DepartmentName

3. **Salary table:**

   SalaryID -> EmployeeID, BaseSalary, EffectiveDate, EndDate

4. **Deductions table:**

   DeductionID -> EmployeeID, DeductionType, Amount, EffectiveDate

5. **Allowances table:**

   AllowanceID -> EmployeeID, AllowanceType, Amount, EffectiveDate

6. **Tax table:**

   TaxID -> EmployeeID, TaxType, Amount, EffectiveDate

2.4.2 Normalization

1st NF

Employee Table:

EmployeeID, FirstName, LastName, DateOfBirth, Gender, Position, EmploymentStartDate, EmploymentEndDate, DepartmentID, DepartmentName, SalaryID, BaseSalary, EffectiveDate, EndDate, DeductionID, DeductionType, DAmount, DEffectiveDate, AllowanceID, AllowanceType, AAmount, AEffectiveDate, TaxID, TaxType, TAmount, TEffectiveDate

2nd NF

Employee Table:

EmployeeID, FirstName, LastName, DateOfBirth, Gender, Position, EmploymentStartDate, EmploymentEndDate, DepartmentID, DepartmentName, DeductionID, DeductionType, DAmount, DEffectiveDate, AllowanceID, AllowanceType, AAmount, AEffectiveDate, TaxID, TaxType, TAmount, TEffectiveDate

Salary Table:

SalaryID, BaseSalary, EffectiveDate, EndDate

3rd NF

Employee Table:

<u>EmployeeID</u>, FirstName, LastName, DateOfBirth, Gender, Position, EmploymentStartDate, EmploymentEndDate, DepartmentID

Department Table:

<u>DepartmentID</u>, DepartmentName

Salary Table:

SalaryID, EmployeeID, BaseSalary, EffectiveDate, EndDate

Deductions Table:

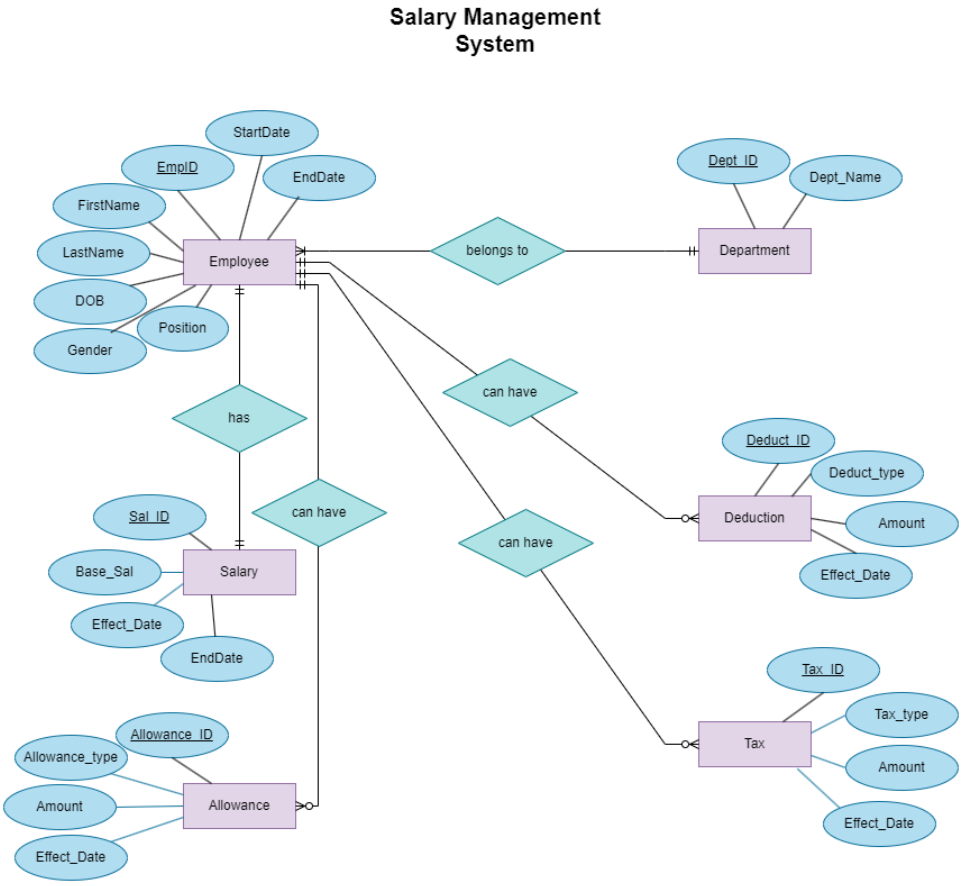
DeductionID, EmployeeID, DeductionType, DAmount, DEffectiveDate

Allowances Table:

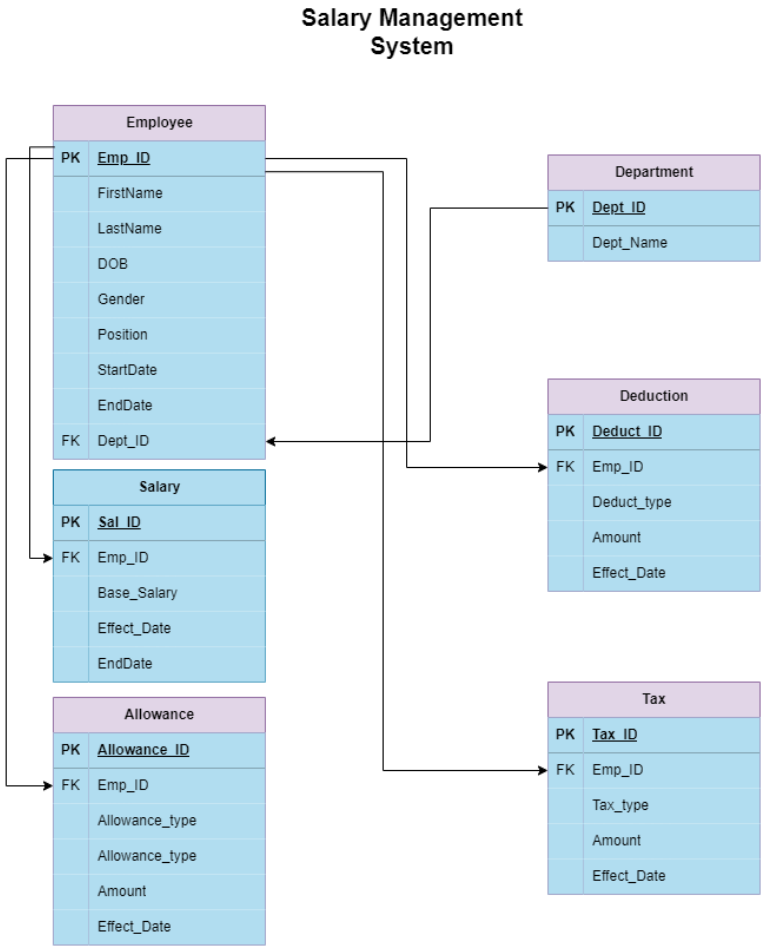
AllowanceID, EmployeeID, AllowanceType, AAmount, AEffectiveDate

Tax Table:
TaxID, EmployeeID, TaxType, TAmount, TEffectiveDate

2.5 Complete Enhanced Entity Relationship Diagram

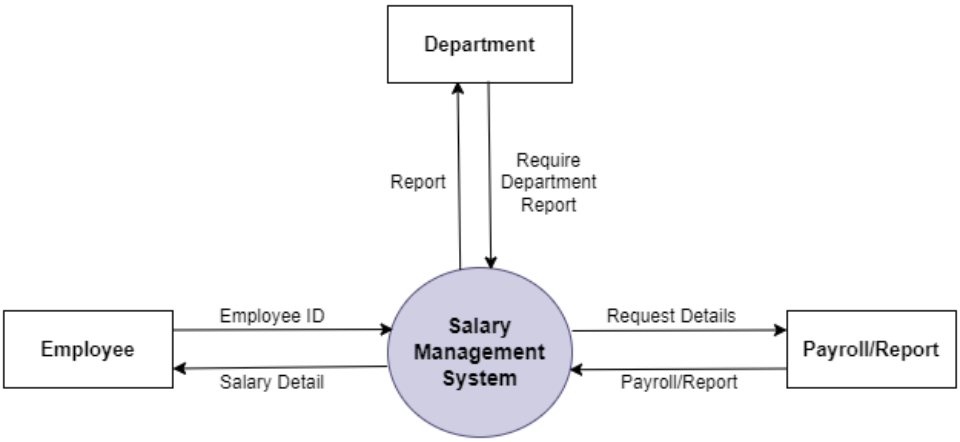


2.6 Complete Relational Model

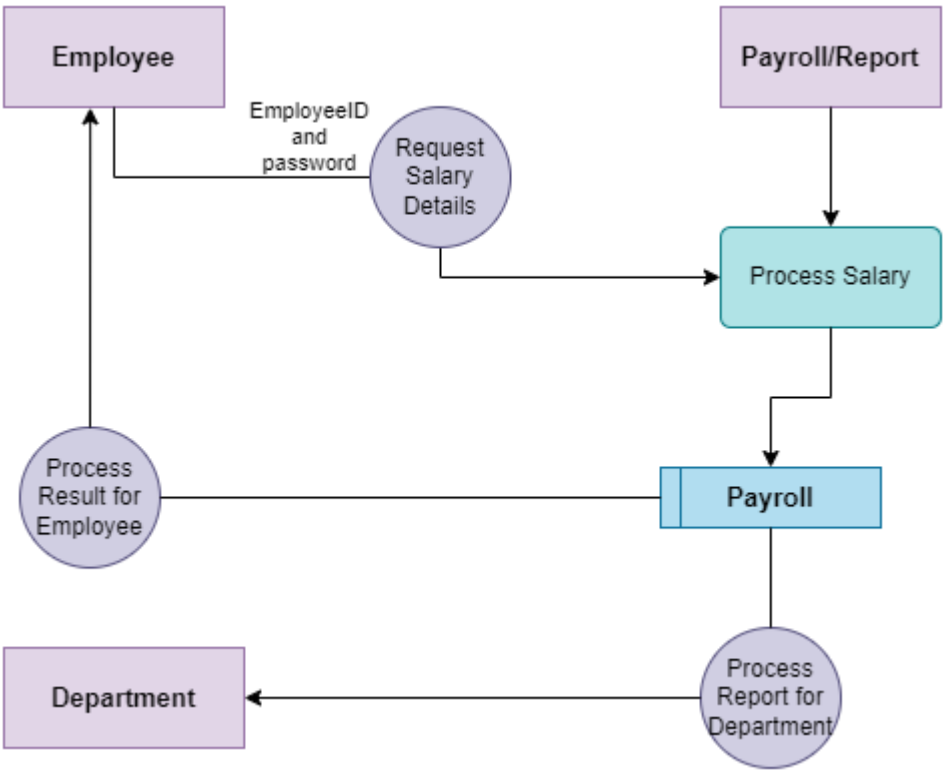


2.7 Data Flow Diagram (DFD)

2.7.1 Level 0 DFD

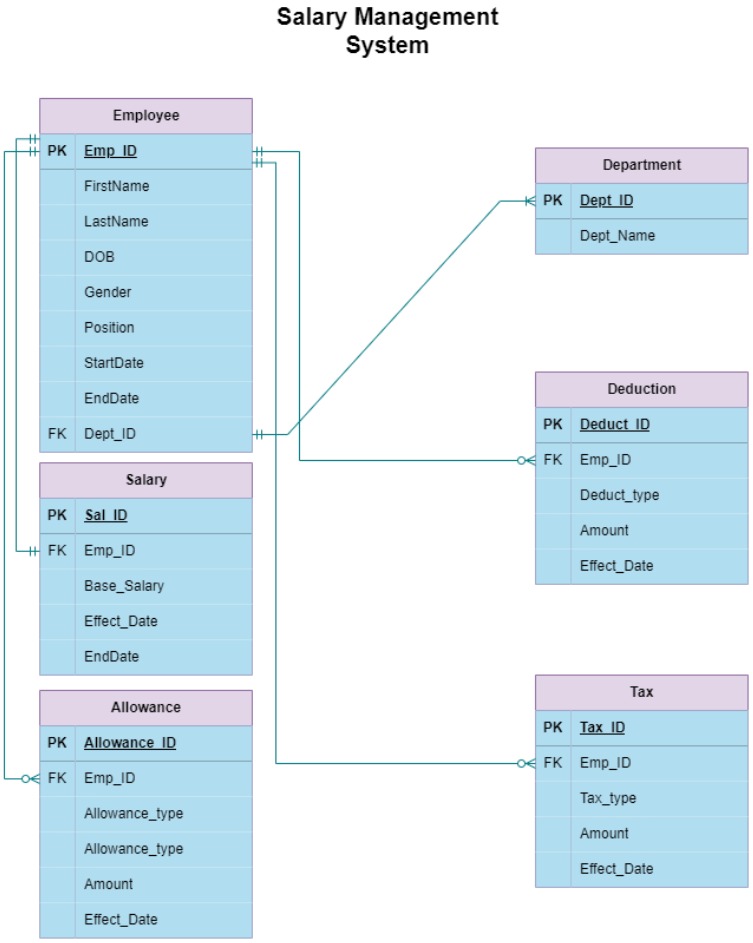


2.7.2 Level 1 DFD



Chapter 3: Physical Database Design

3.1 Physical Data Model



3.2 Physical Data Model Implementation (MySQL Workbench)

- Create Query

```
-- Create the Database Salary

create Database Salary;

Use Salary;

-- Drop Database Salary;


-- Create the Employee table

CREATE TABLE Employee (

  EmployeeID INT auto_increment  KEY,

  FirstName VARCHAR(50),

  LastName VARCHAR(50),

  DateOfBirth DATE,

  Gender VARCHAR(10),

  Position VARCHAR(50),

  EmploymentStartDate DATE,

  EmploymentEndDate DATE

);

-- Create the Department table

CREATE TABLE Department (

  DepartmentID INT auto_increment PRIMARY KEY,
```

```
  DepartmentName VARCHAR(50)
);
-- Create the Salary table
CREATE TABLE Salary (
  SalaryID INT auto_increment PRIMARY KEY,
  EmployeeID INT,
  BaseSalary DECIMAL(10,2),
  EffectiveDate DATE,
  EndDate DATE,
  Currency VARCHAR(10),
  );
-- Create the Deductions table
CREATE TABLE Deductions (
  DeductionID INT auto_increment PRIMARY KEY,
  EmployeeID INT,
  DeductionType VARCHAR(50),
  Amount DECIMAL(10,2),
  EffectiveDate DATE,
  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
-- Create the Allowances table
```

```
CREATE TABLE Allowances (
  AllowanceID INT auto_increment PRIMARY KEY,
  EmployeeID INT,
  AllowanceType VARCHAR(50),
  Amount DECIMAL(10,2),
  EffectiveDate DATE,
  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
-- Create the Tax table
CREATE TABLE Tax (
  TaxID INT auto_increment PRIMARY KEY,
  EmployeeID INT,
  TaxType VARCHAR(50),
  Amount DECIMAL(10,2),
  EffectiveDate DATE,
  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
```

- Insert Query

```
-- Inserting data into the Employee table
INSERT INTO Employee (FirstName, LastName, DateOfBirth, Gender,
Position, EmploymentStartDate, EmploymentEndDate)
```

```
VALUES
('John', 'Doe', '1990-05-15', 'Male', 'Manager', '2020-01-01', NULL),
('Jack', 'Taylor', '1995-10-5', 'Male', 'Assistant', '2023-01-31', NULL),
('Sid', 'Alice', '1993-03-01', 'Female', 'Clerk', '2019-04-17', '2022-06-12'),
('Alina', 'Kristen', '1997-12-04', 'Female', 'Database Engineer', '2019-01-31', NULL),
('Steve', 'Ryan', '1995-03-07', 'Male', 'Assistant', '2018-04-09', '2020-03-16');

-- Inserting data into the Department table
INSERT INTO Department (DepartmentID, DepartmentName)
VALUES
(1, 'Sales'),
(2, 'HR'),
(3, 'HR'),
(4, 'Marketing'),
(5, 'Sales');

-- Inserting data into the Salary table
INSERT INTO Salary (SalaryID, EmployeeID, BaseSalary, EffectiveDate, EndDate)
VALUES
(1, 1, 75000, '2021-01-01', NULL),
```

```
(2, 2, 50000, '2021-03-04', NULL),

(3, 3, 15000, '2020-01-15', '2022-06-12'),

(4, 4, 35000, '2019-11-20', NULL),

(5, 5, 20000, '2014-01-01', '2020-03-16');


-- Inserting data into the Deductions table

INSERT INTO Deductions (DeductionID, EmployeeID, DeductionType,
Amount, EffectiveDate)

VALUES

(1, 1, 'Health Insurance', 2000, '2022-09-01'),

(2, 1, 'Home Insurance', 250000, '2023-11-30'),

(3, 2, 'Life Insurance',70000, '2023-05-14'),

(4, 4, 'Life Insurance', 900000, '2021-01-05'),

(5, 2, 'Vehicle Insurance', 200000, '2023-02-01');


-- Inserting data into the Allowances table

INSERT INTO Allowances (AllowanceID, EmployeeID, AllowanceType,
Amount, EffectiveDate)

VALUES

(1, 1, 'Bonus', 5000, '2021-02-01'),

(2, 5, 'Overtime', 2000, '2018-11-18'),

(3, 4, 'Special', 2500, '2021-01-01'),
```

```
(4, 5, 'Medical', 2000, '2019-12-20'),

(5, 3, 'Bonus', 2100, '2020-03-26');



-- Inserting data into the Tax table

INSERT INTO Tax (TaxID, EmployeeID, TaxType, Amount, EffectiveDate)

VALUES

(1, 1, 'Income Tax', 10000, '2021-02-01'),

(2, 2, 'Excise', 1000, '2020-04-21'),

(3, 5, 'Property Tax', 15000, '2019-10-17'),

(4, 5, 'Corporate Tax', 2000, '2020-01-01'),

(5, 3, 'Income Tax', 5000, '2021-10-20');
```

- Alter Query: altering a few things in project tables
- Select Query: Retrievong data from single table or multiple tables
- Queries for orderby, having, like, limit etc…
- Applying Referential Integrity Constraint
- Applying Joins between tables (ALL TYPES)
- Applying Views, stored procedures, triggers

```
ALTER TABLE Employee
ADD COLUMN Email VARCHAR(100);

ALTER TABLE Salary
DROP COLUMN EndDate;
```

```
ALTER TABLE Salary
ADD CONSTRAINT FK_Salary_Employee
FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID);

ALTER TABLE Department
RENAME COLUMN DepartmentName TO DeptName;

ALTER TABLE Employee
ADD CONSTRAINT Check_DATE Check (EmploymentEndDate  >
EmploymentStartDate);

Rename Table Employee to Employees;

SELECT * FROM Employees;

SELECT * FROM Department WHERE DepartmentID = 1;

SELECT * FROM Employees WHERE EmployeeID between 1 and 4 and Position
NOT IN ('Clerk', 'Assistant');

SELECT * FROM Employees
order by FirstName ASC
limit 3;

Select FirstName FROM Employees
where FirstName like 'j%';

Select LastName FROM Employees
where LastName like '_r%';

Select Taxtype FROM Tax
group by TaxType;

Select AllowanceType, sum(Amount) FROM Allowances
GROUP BY AllowanceType
```

```
HAVING sum(amount) > 250.0;

SELECT EmployeeID, SUM(Amount) AS TotalDeductions
FROM Deductions
GROUP BY EmployeeID;

SELECT e.*, d.DeptName
FROM Employees e
JOIN Department d ON e.DepartmentID = d.DepartmentID;
-- Join = Inner Join

SELECT e.*, t.TaxType, t.Amount
FROM Employees e
LEFT JOIN Tax t ON e.EmployeeID = t.EmployeeID;

SELECT e.*, a.AllowanceType, a.Amount
FROM Employees e
Right JOIN Allowances a ON e.EmployeeID = a.EmployeeID;

SELECT e.*, d.deductionType, a.Amount
FROM Employee e;
-- Full Outer JOIN Deduction d ON e.EmployeeID = d.EmployeeID;

CREATE VIEW EmployeeDetails AS
SELECT e.*, d.DeptName
FROM Employees e
JOIN Department d ON e.DepartmentID = d.DepartmentID;

Select * FROM EmployeeDetails;
Select * FROM EmployeeDetails where EmployeeID = 1;

CREATE VIEW TotalDeductions AS
SELECT EmployeeID, SUM(Amount) AS TotalDeductions
FROM Deductions
GROUP BY EmployeeID;
```

```
Select * FROM TotalDeductions;

CREATE VIEW TotalAllowances AS
SELECT EmployeeID, SUM(Amount) AS TotalAllowances
FROM Allowances
GROUP BY EmployeeID;

SELECT * FROM TotalAllowances;

CREATE VIEW EmployeeSalary AS
SELECT e.EmployeeID, e.FirstName, e.LastName, s.BaseSalary, s.EffectiveDate
FROM Employees e
JOIN Salary s ON e.EmployeeID = s.EmployeeID;

SELECT * FROM EmployeeSalary where EmployeeID = 5;

CREATE VIEW EmployeeTaxes AS
SELECT e.EmployeeID, e.FirstName, e.LastName, t.TaxType, t.Amount
FROM Employees e
LEFT JOIN Tax t ON e.EmployeeID = t.EmployeeID;

SELECT * FROM EmployeeTaxes;
SELECT * FROM EmployeeTaxes where EmployeeID = 2;

DELIMITER //
CREATE PROCEDURE GetEmployeeDetails()
BEGIN
  SELECT e.*, d.DeptName
  FROM Employees e
  JOIN Department d ON e.DepartmentID = d.DepartmentID;
END //

CALL GetEmployeeDetails();
```

```
DELIMITER //
CREATE PROCEDURE GetEmployeeDetailsbyID(IN p_EmployeeID INT)
BEGIN
  SELECT e.*, d.DeptName
  FROM Employees e
  JOIN Department d ON e.DepartmentID = d.DepartmentID
  where e.employeeID = p_employeeID;
END //

CALL GetEmployeeDetailsbyID(1);

DELIMITER //
CREATE PROCEDURE DeleteEmployee(IN p_EmployeeID INT)
BEGIN
  DELETE FROM Employees
  WHERE EmployeeID = p_EmployeeID;
  Select 'Employee Record Deleted';
END;

CALL DeleteEmployee(6);

DELIMITER //
CREATE PROCEDURE CalculateTotalDeductions(OUT TotalDeductions INT)
BEGIN
  SELECT SUM(Amount) INTO TotalDeductions
  FROM Deductions;
END;

CALL CalculateTotalDeductions(@TotalDeductions);
SELECT @TotalDeductions AS TotalDeductions;

DROP PROCEDURE CalculateTotalDeductions;

DELIMITER //
CREATE TRIGGER InsertBaseSalary
```

```
BEFORE INSERT ON Salary
FOR EACH ROW
BEGIN
IF New.BaseSalary < 0 Then SET New.BaseSalary = 0;
END IF;
END //

INSERT EMPLOYEES(FirstName, LastName, DateOfBirth, Gender, Position,
EmploymentStartDate, EmploymentEndDate, DepartmentID, email)
VALUES ('Kamran', 'Arshad', '1996-04-14', 'Male', 'Software Engineer', '2020-09-
01', NULL, 2, 'kamranarshad@umt.edu.pk');
INSERT INTO Salary(EmployeeID, BaseSalary, EffectiveDate)
Values (6, -100000, '2023-05-30');

DELIMITER //
CREATE TRIGGER BeforeDeleteAllowance
BEFORE DELETE ON Allowances
FOR EACH ROW
BEGIN
IF old.amount > 500 Then
SIGNAL SQLSTATE '25000' SET MESSAGE_TEXT = 'Cannot Delete Record';
END IF;
END //

DELETE FROM Allowances where AllowanceID = 1;

drop trigger BeforeDeleteAllowance;

SELECT * FROM EMPLOYEES;

Desc Table Employees;
Show TABLES;
DROP TABLE Deductions;
```