DATABASE MANAGEMENT SYSTEM - CSA0593

ASSIGNMENT 5

K.GAYATHRI

192311448

QUESTION:

Design and implement a comprehensive database management system for a large organization to efficiently manage employee data, departmental structure, project assignments, and compensationModel tables for employees, departments, projects, and salaries.

Write stored procedures to assign employees to projects and update salaries.

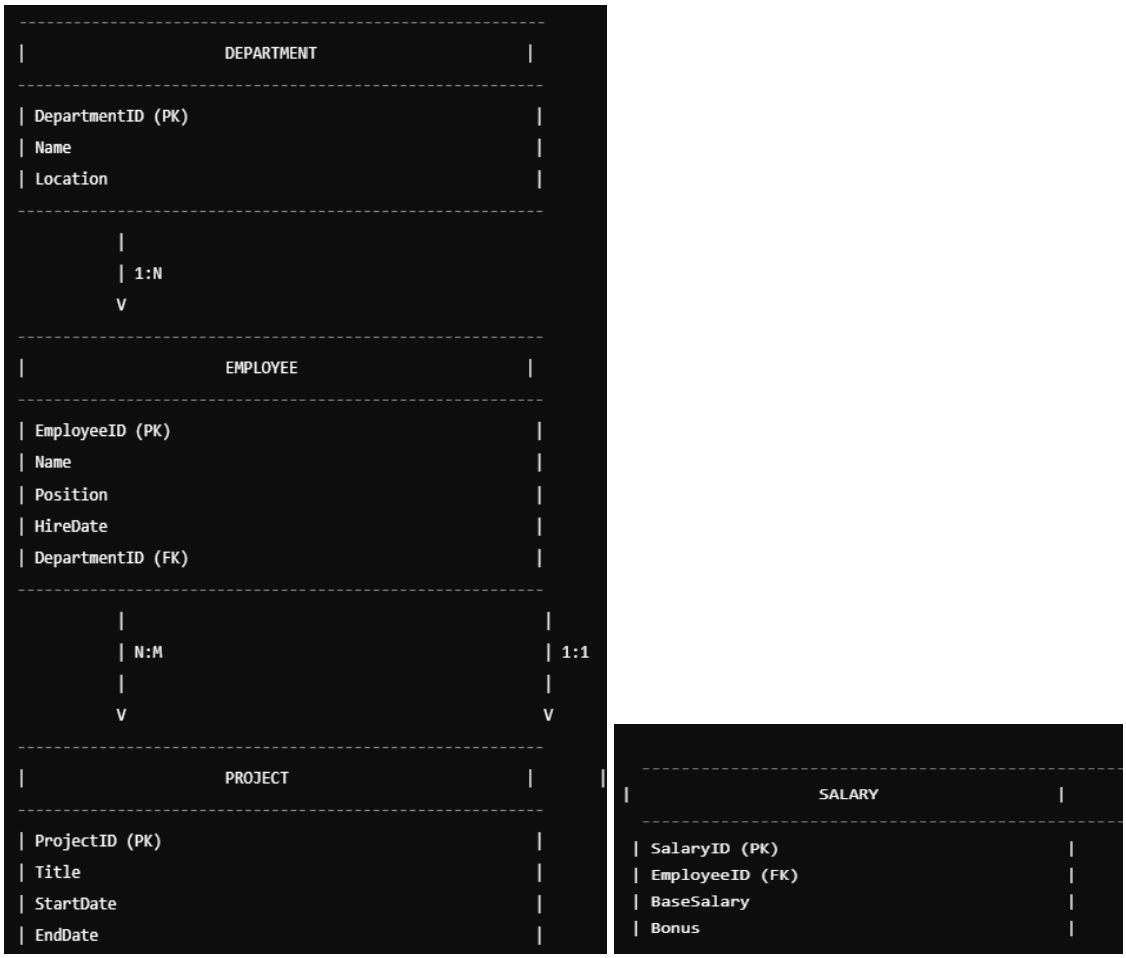
Implement triggers to track promotions and department transfers.

Write SQL queries to analyze employee performance and salary distribution."

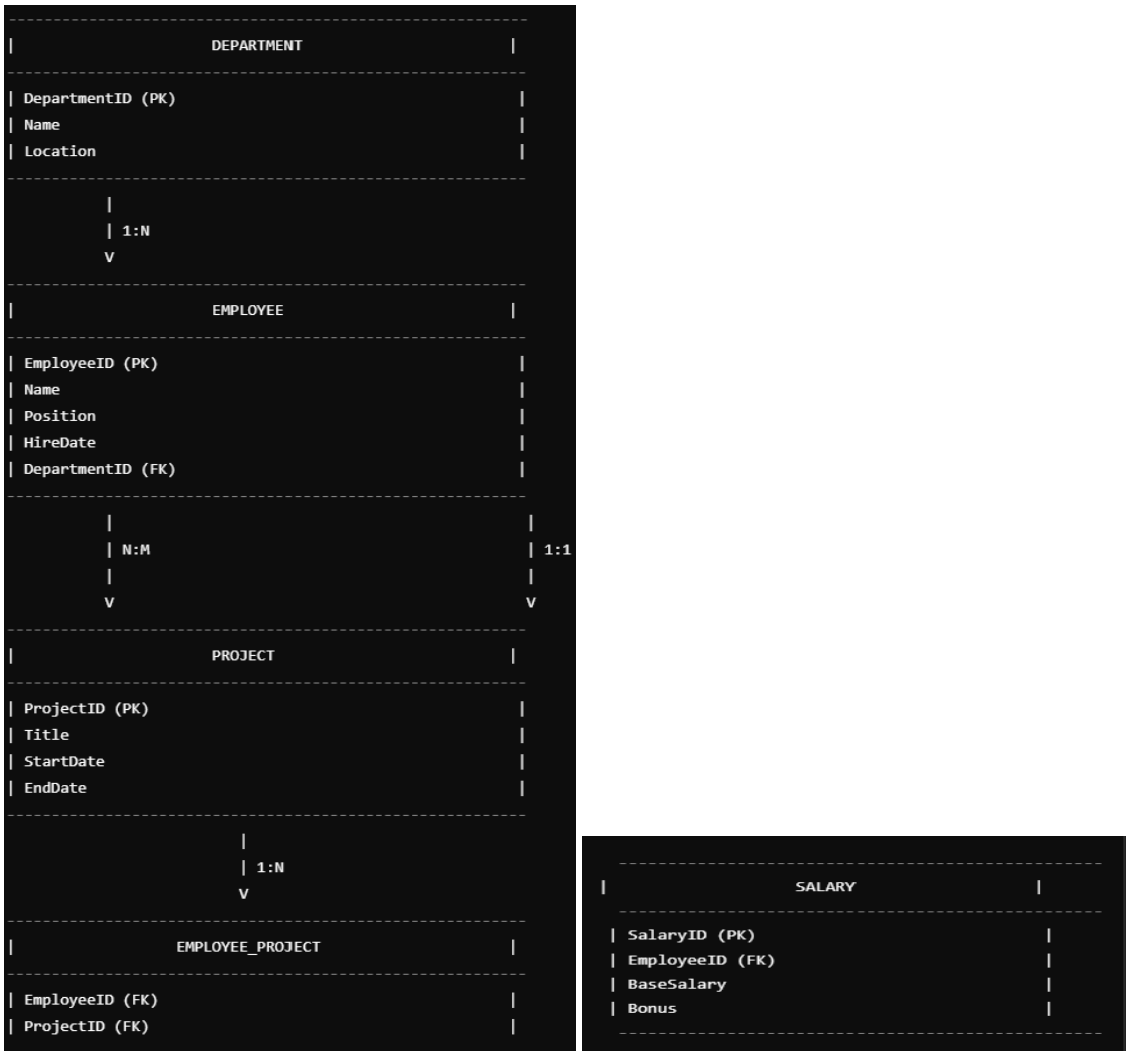
ANSWER:

CONCEPTUAL E.R.DIAGRAM:

LOGICAL E.R.DIAGRAM:



PHYSICAL E.R.DIAGRAM:



MYSQL STATEMENTS:

Database Design

CREATE DATABASE org_management;

```
USE org_management;

CREATE TABLE departments (
  department_id INT PRIMARY KEY,
  department_name VARCHAR(255)
);

CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  name VARCHAR(255),
  email VARCHAR(255),
  hire_date DATE,
  job_title VARCHAR(255),
  department_id INT,
  salary DECIMAL(10, 2),
  FOREIGN KEY (department_id) REFERENCES departments(department_id)
);

CREATE TABLE projects (
  project_id INT PRIMARY KEY,
  project_name VARCHAR(255),
  start_date DATE,
  end_date DATE
);
```

```
CREATE TABLE project_assignments (
  assignment_id INT PRIMARY KEY,
  project_id INT,
  employee_id INT,
  assignment_date DATE,
  end_date DATE,
  FOREIGN KEY (project_id) REFERENCES projects(project_id),
  FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);

CREATE TABLE salaries (
  salary_id INT PRIMARY KEY,
  employee_id INT,
  salary DECIMAL(10, 2),
  effective_date DATE,
  FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);

CREATE TABLE promotions (
  promotion_id INT PRIMARY KEY,
  employee_id INT,
  promotion_date DATE,
  new_job_title VARCHAR(255),
  FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
);
```

```
CREATE TABLE department_transfers (
  transfer_id INT PRIMARY KEY,
  employee_id INT,
  transfer_date DATE,
  new_department_id INT,
  FOREIGN KEY (employee_id) REFERENCES employees(employee_id),
  FOREIGN KEY (new_department_id) REFERENCES
departments(department_id)
);
```

Stored Procedures

```
DELIMITER //

CREATE PROCEDURE assign_employee_to_project(
  IN project_id INT,
  IN employee_id INT,
  IN assignment_date DATE,
  IN end_date DATE
)
BEGIN
  INSERT INTO project_assignments (project_id, employee_id,
assignment_date, end_date)
  VALUES (project_id, employee_id, assignment_date, end_date);
END //
```

```
CREATE PROCEDURE update_salary(
  IN employee_id INT,
  IN new_salary DECIMAL(10, 2),
  IN effective_date DATE
)
BEGIN
  INSERT INTO salaries (employee_id, salary, effective_date)
  VALUES (employee_id, new_salary, effective_date);

  UPDATE employees
  SET salary = new_salary
  WHERE employee_id = employee_id;
END //
```

Triggers

```
DELIMITER //

CREATE TRIGGER track_promotions
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
  IF NEW.job_title != OLD.job_title THEN
```

```
    INSERT INTO promotions (employee_id, promotion_date, new_job_title)
    VALUES (NEW.employee_id, NOW(), NEW.job_title);
  END IF;
END //

CREATE TRIGGER track_department_transfers
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
  IF NEW.department_id != OLD.department_id THEN
    INSERT INTO department_transfers (employee_id, transfer_date,
new_department_id)
    VALUES (NEW.employee_id, NOW(), NEW.department_id);
  END IF;
END //
```

SQL Queries

```
-- Analyze employee performance
SELECT
  employees.name,
  COUNT(project_assignments.assignment_id) AS number_of_projects,
  SUM(DATEDIFF(project_assignments.end_date,
project_assignments.assignment_date)) /
COUNT(project_assignments.assignment_id) AS average_project_duration
```

```
FROM
  employees
  JOIN project_assignments ON employees.employee_id =
project_assignments.employee_id
GROUP BY
  employees.name;

-- Salary distribution
SELECT
  departments.department_name,
  AVG(employees.salary) AS average_salary,
  MIN(employees.salary) AS minimum_salary,
  MAX(employees.salary) AS maximum_salary
FROM
  employees
  JOIN departments ON employees.department_id =
departments.department_id
GROUP BY
  departments.department_name;
```

Conclusion:

Designing a comprehensive database management system for a large organization requires careful consideration of various factors.

Key benefits of this system include:

1. Efficient employee data management.

2. Automated tracking of project assignments and salary updates.

3. Data-driven insights into employee performance and salary distribution.

By implementing this database management system, organizations can improve operational efficiency, enhance employee experiences, and make data-driven decisions.