# Employee Management System

## 1. Introduction:

The Employee Management System (EMS) is a software application designed to streamline and automate various HR-related tasks within an organisation. It serves as a central hub for managing employee information, tracking their activities, and facilitating HR operations.

Key Features of the Employee Management System:

- Employee Information Management
- Attendance Tracking
- Leave Management
- Salary and Compensation
- Performance Evaluation

The Employee Management System database will consist of the following tables:

1. Employee: Stores information about individual employees.
2. Job Department: Contains details about different departments within the organisation.
3. Users: This table likely stores information about system users, possibly including administrators or managers.
4. Salary: Tracks salary information for employees.
5. Qualification:This table seems to store data related to employee qualifications.
6. Payroll: This table appears to be related to payroll processing,

We will use MySQL Workbench as the DBMS to create the database and its related operations.

# 2. MySQL Workbench

## 2.1 MySQL

MySQL is an open-source Relational Database Management System (RDBMS) developed by Oracle Corporation. It uses Structured Query Language (SQL) for interacting with databases. MySQL is a popular choice for organizations due to its multiple storage engines, high performance, cost-effectiveness, and cross-platform compatibility.

## 2.2 MySQL Workbench

MySQL Workbench is a versatile, open-source tool that provides database design, SQL development, and administration features. It offers a graphical interface to interact with databases, create data models, reverse engineer existing databases, and forward engineer models to live databases. MySQL Workbench is a valuable tool for creating and maintaining the EMS database.

## 2.3 Download MySQL

The installation process is similar to other operating systems.

1. Open the MySQL website on a browser. Click on the following link: MySQL Downloads.
2. Select the Downloads option.
3. Select MySQL Installer for Windows.
4. Choose the desired installer and click on download.
5. After it downloads the installer, open it.
6. It will ask for permission; when it does, click Yes. The installer will then open. Now, it will ask to choose the setup type, here, select Custom.
7. Click on Next. With this, you will be able to install MySQL server, MySQL Workbench, and MySQL shell.

8. Open MySQL Servers, select the server you want to install, and move it to the Products/Features to be installed window section. Now, expand Applications, choose MySQL Workbench and MySQL shell. Move both of them to 'Products/Features to be installed'.

9. Click on the Next button. Now, click on the Execute button to download and install the MySQL server, MySQL Workbench, and the MySQL shell.

10. Once the product is ready to configure, click on Next. Under Type and Networking, go with the default settings and select Next.

11. For authentication, use the recommended strong password encryption.

12. Set your MySQL Root password and click on next.

13. Go for the default windows service settings and under apply configuration, click on execute. Once the configuration is complete, click on finish.

14. Complete the installation. This will now launch the MySQL Workbench and the MySQL Shell.

Once MySQL Workbench is installed, select the Local instance and enter the password.

Now, you can use the MySQL query tab to write your SQL queries.

**Identify Entities**

- Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data.
- Each entity should correspond to a table in your database.

**Define Attributes**

- For each entity, list the attributes (properties or fields) that describe it.
- These attributes will become columns in the corresponding database table.

**Identify Relationships**

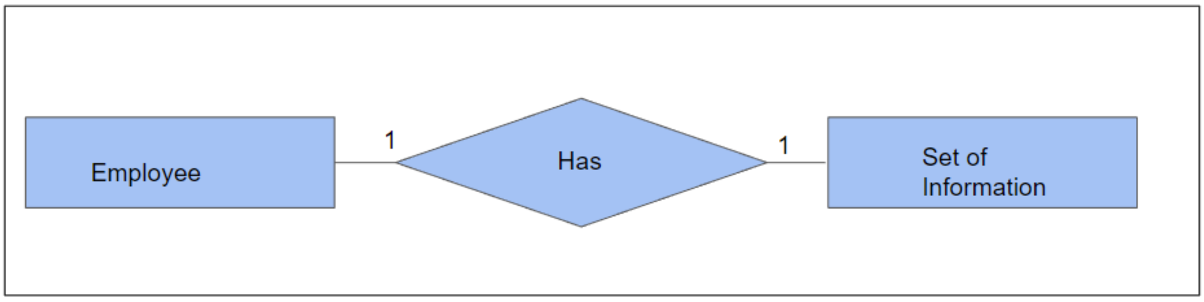
- Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
- Represent these relationships using lines connecting the entities.

Let’s see a relationships with example:

1. **One-to-One:** In a one-to-one relationship, each entity in each entity set can participate only once in the relationship.This means that for every entity in one set, there is a single corresponding entity in the other set.

   Example- Each employee can have only one set of personal information. Each set of personal information corresponds to a single employee.

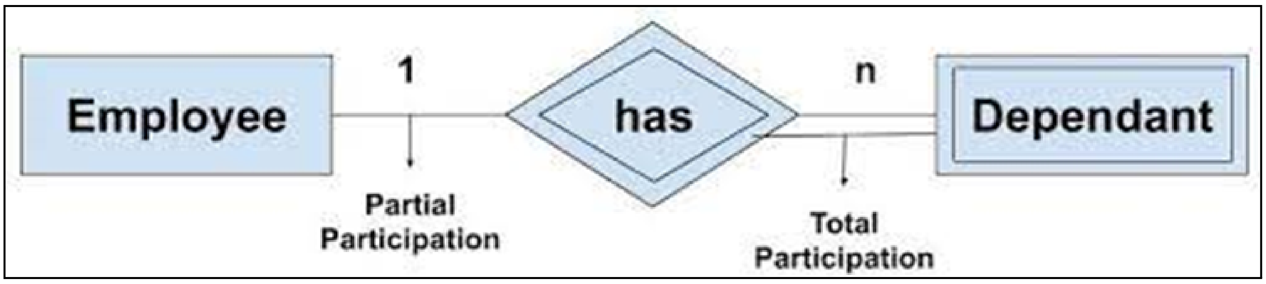
Diagram -



2. **Many-to-Many Relationship :** When more than one element of an entity is associated with more than one element of another entity, this is called a many-to-many relationship.

   Example- Each employee can have only one set of personal information. Each set of personal information corresponds to a single employee.

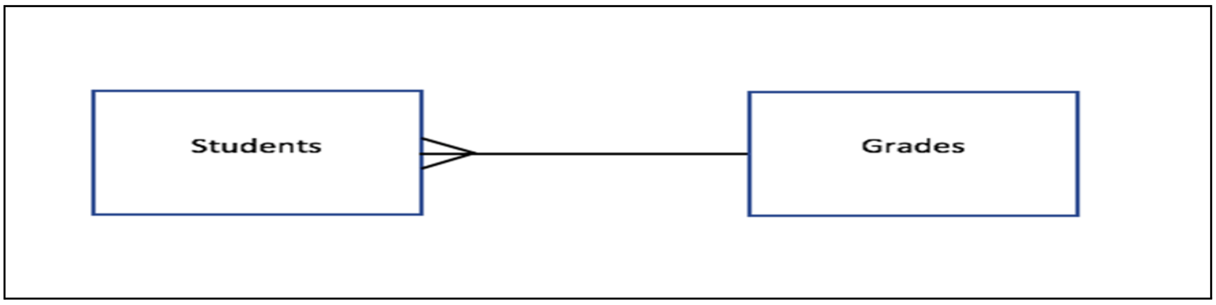
Diagram -



3. **Many-to-One:** When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one.

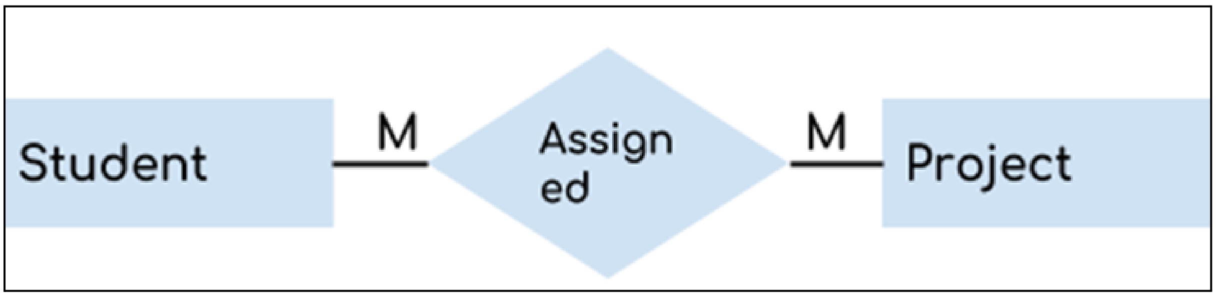
Example- Let us assume that a student can take only one course but one course can be taken by many students.

Diagram -



4. **Many-to-One:** When entities in all entity sets can take part more than once in the relationship cardinality is many to many.

Diagram -



**Cardinality Notation**

Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

- Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship.
- Cardinality describes how many instances of one entity are related to how many instances of another entity.

Common notations include:
- · One (1)
- · Zero or one (0..1)
- · Many (N)
- · Zero or many (0..N)

**Optional:**

**Add Attributes and Constraints**
- Include additional information in your ERD, such as primary keys, foreign keys, and constraints (e.g., unique constraints).

**Create the Diagram**
- Use specialized diagramming software or tools (e.g., Lucidchart, draw.io, or even pen and paper) to create your ERD.

**Refine and Review:**
- Review your ERD with stakeholders and team members to ensure it accurately represents the data model and relationships. Make any necessary refinements.

**Entities of Employee Management System :**

1. Employee
2. Users
3. Job Department
4. Salary/Bonous

5. Qualification
6. Payroll

➢ **Let's identify the attributes and relationships of each entity for the Employee Management System.**

1. **Employee Table**
   1.1. **emp_ID (PK):** This is the primary key, representing the unique identifier for each employee.
   1.2. **fname**: Employee's first name.
   1.3. **lname:** Employee's last name.
   1.4. **gender:** Employee's gender.
   1.5. **age:** Employee's age.
   1.6. **contact_add:** Employee's contact address.
   1.7. **emp_email:** Employee's email address.
   1.8. **emp_pass:** Employee's password.

2. **Users Table**
   2.1. **admin_ID (PK):** Primary key for admin or user identification.
   2.2. **fname:** First name of the admin or user.
   2.3. **lname:** Last name of the admin or user.
   2.4. **gender:** Gender of the admin or user.
   2.5. **age:** Age of the admin or user.
   2.6. **contact_add:** Contact address of the admin or user.
   2.7. **admin_email:** Email address of the admin or user.
   2.8. **admin_pass:** Password for the admin or user.

3. **Job Department Table**
   3.1. **job_ID (PK):** Unique identifier for each job department.

3.2. **job_dept:** Job department's name.
3.3. **name:** Job name.
3.4. **description:** Job description.
3.5. **salary_range:** Salary range for the job.

## 4. Salary or Bonus Table

4.1. **salary_ID (PK):** Unique identifier for each salary or bonus record.
4.2. **job_ID (FK):** Foreign key referencing the job department for which the salary or bonus is applicable.
4.3. **amount:** Salary or bonus amount.
4.4. **annual:** Date for annual salary.
4.5. **bonus:** Date for bonus.

## 5. Qualification Table

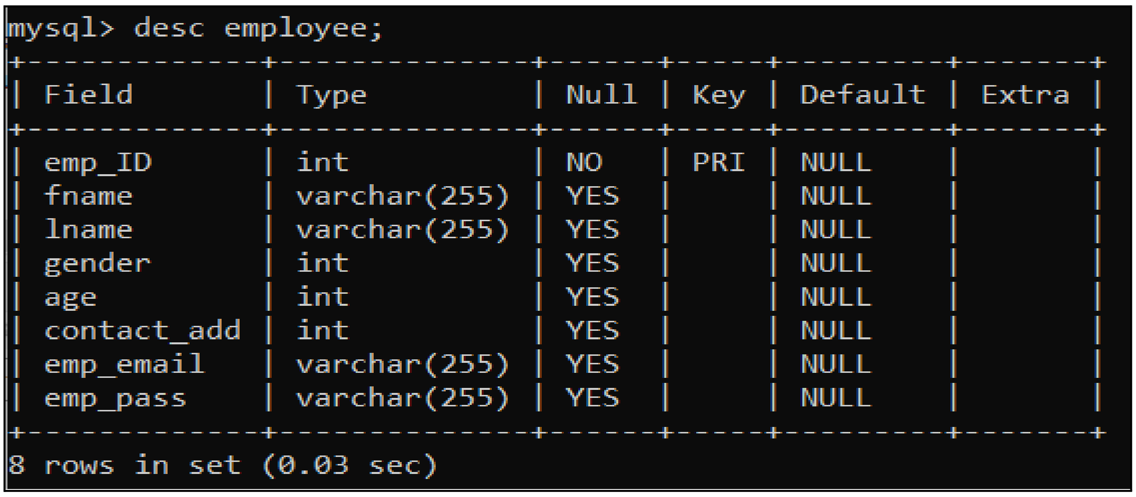
5.1. **qual_ID (PK):** Unique identifier for each qualification record.
5.2. **emp_ID (FK):** Foreign key referencing the employee to whom the qualification applies.
5.3. **position:** Position or job title for the qualification.
5.4. **requirements:** Qualification requirements.
5.5. **date_in:** Date when the qualification was added.

## 6. Payroll Table

6.1. **payroll_ID (PK):** Unique identifier for each payroll record.
6.2. **emp_ID (FK):** Foreign key referencing the employee for whom the payroll is generated.
6.3. **job_ID (FK):** Foreign key referencing the job department.
6.4. **salary_ID (FK):** Foreign key referencing the salary or bonus.
6.5. **leave_ID (FK):** Foreign key referencing leave records.
6.6. **date:** Date of the payroll report.
6.7. **report:** Payroll report in text format.
6.8. **total_amount:** Total amount in the payroll.

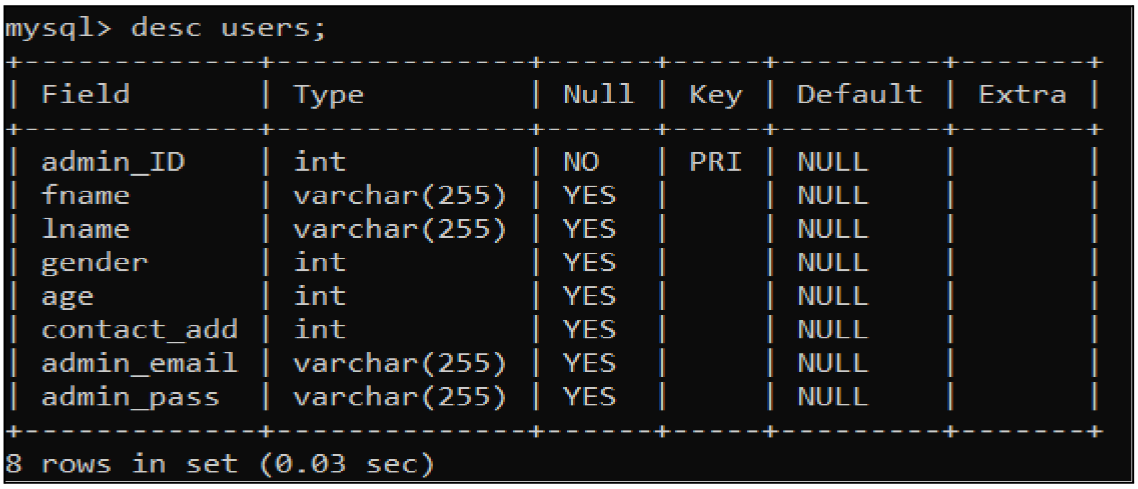
## 3. Table Structure

1. Employee



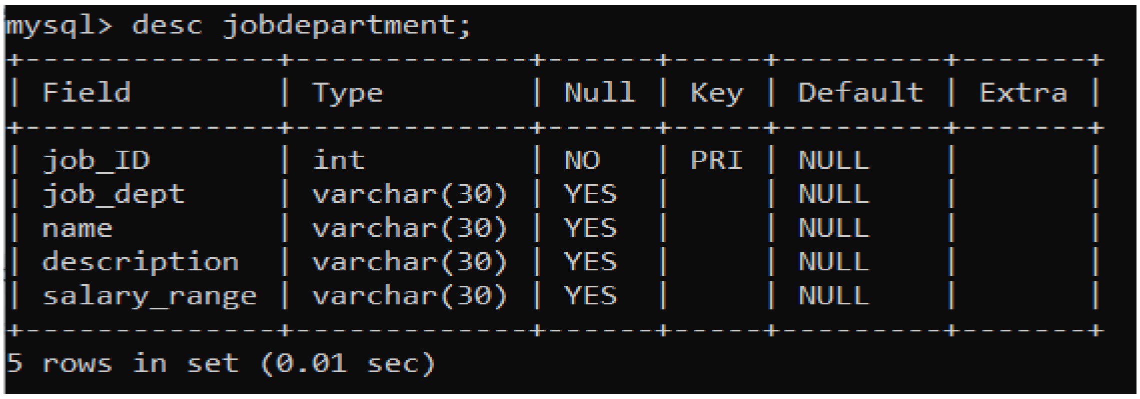
```
mysql> desc employee;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| emp_ID      | int          | NO   | PRI | NULL    |       |
| fname       | varchar(255) | YES  |     | NULL    |       |
| lname       | varchar(255) | YES  |     | NULL    |       |
| gender      | int          | YES  |     | NULL    |       |
| age         | int          | YES  |     | NULL    |       |
| contact_add | int          | YES  |     | NULL    |       |
| emp_email   | varchar(255) | YES  |     | NULL    |       |
| emp_pass    | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
8 rows in set (0.03 sec)
```

2. Users



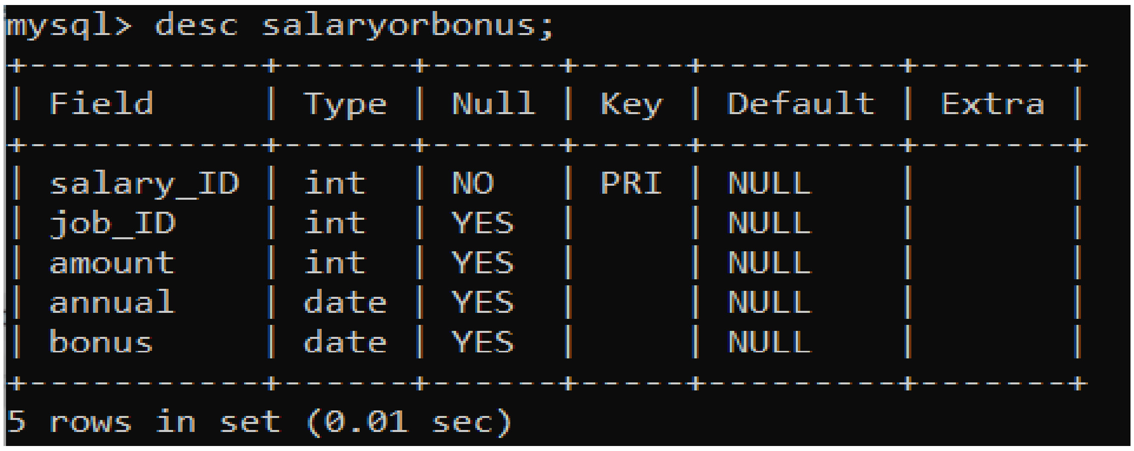
```
mysql> desc users;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| admin_ID    | int          | NO   | PRI | NULL    |       |
| fname       | varchar(255) | YES  |     | NULL    |       |
| lname       | varchar(255) | YES  |     | NULL    |       |
| gender      | int          | YES  |     | NULL    |       |
| age         | int          | YES  |     | NULL    |       |
| contact_add | int          | YES  |     | NULL    |       |
| admin_email | varchar(255) | YES  |     | NULL    |       |
| admin_pass  | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
8 rows in set (0.03 sec)
```

3. Job Department



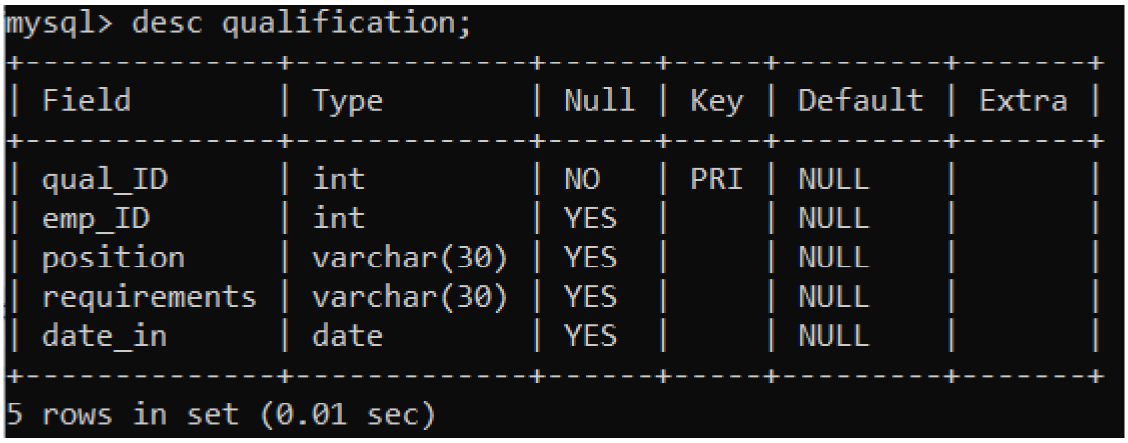
```
mysql> desc jobdepartment;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| job_ID       | int         | NO   | PRI | NULL    |       |
| job_dept     | varchar(30) | YES  |     | NULL    |       |
| name         | varchar(30) | YES  |     | NULL    |       |
| description  | varchar(30) | YES  |     | NULL    |       |
| salary_range | varchar(30) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

4. Salary/Bonous



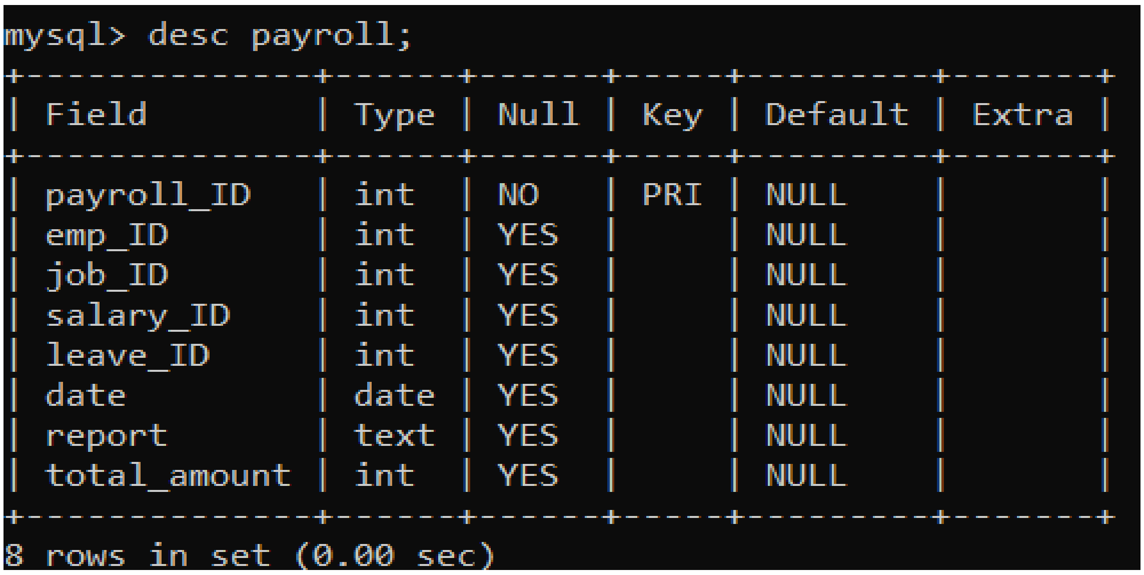
```
mysql> desc salaryorbonus;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| salary_ID | int  | NO   | PRI | NULL    |       |
| job_ID    | int  | YES  |     | NULL    |       |
| amount    | int  | YES  |     | NULL    |       |
| annual    | date | YES  |     | NULL    |       |
| bonus     | date | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

5. Qualification



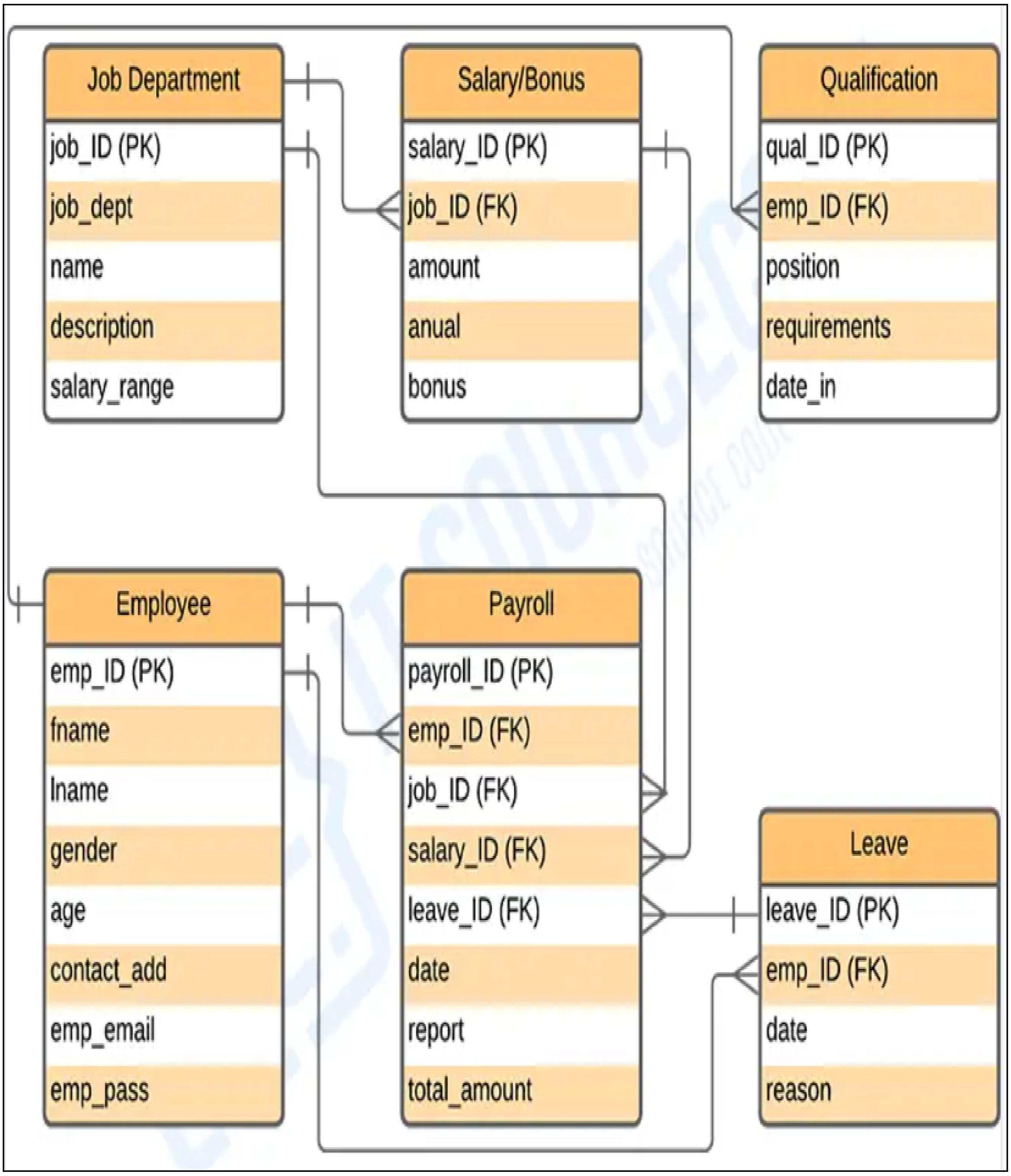
```
mysql> desc qualification;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| qual_ID      | int         | NO   | PRI | NULL    |       |
| emp_ID       | int         | YES  |     | NULL    |       |
| position     | varchar(30) | YES  |     | NULL    |       |
| requirements | varchar(30) | YES  |     | NULL    |       |
| date_in      | date        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

6. Payroll



```
mysql> desc payroll;
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| payroll_ID   | int  | NO   | PRI | NULL    |       |
| emp_ID       | int  | YES  |     | NULL    |       |
| job_ID       | int  | YES  |     | NULL    |       |
| salary_ID    | int  | YES  |     | NULL    |       |
| leave_ID     | int  | YES  |     | NULL    |       |
| date         | date | YES  |     | NULL    |       |
| report       | text | YES  |     | NULL    |       |
| total_amount | int  | YES  |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

## 4. ER Diagram For Employee Management System



Entity Relationship Diagram

**Explanation of the ERD:**

- **Employee Entity:** Represents information about employees. Each employee has a unique identifier (emp_ID) and attributes such as first name (fname), last name (lname), gender, age, contact address, email, and password.
- **Users Entity:** Stores data about administrators or users. Similar to the "Employee" entity, it includes attributes for admin_ID, first name, last name, gender, age, contact address, email, and password.
- **Job Department Entity:** Contains information about job departments. It has a primary key (job_ID) and attributes for job department name (job_dept), job name (name), description, and salary range.
- **Salary or Bonus Entity:** Records details related to salaries and bonuses. It includes attributes like salary_ID (primary key), job_ID (foreign key referencing "Job Department"), amount, annual date, and bonus date.
- **Qualification Entity:** Manages qualifications of employees. It has a unique identifier (qual_ID) and attributes for emp_ID (foreign key referencing "Employee"), position of application, requirements, and the date of qualification.
- **Payroll Entity:** Stores payroll information, such as payroll_ID (primary key), emp_ID (foreign key referencing "Employee"), job_ID (foreign key referencing "Job Department"), salary_ID (foreign key referencing "Salary or Bonus"), leave_ID (foreign key), date of the payroll report, a text-based report, and the total amount.

# 5. Database Setup

## 5.1 Creating the Database :

Using MySQL Workbench, the project database, named "employeedb," was created. The database will serve as the foundation for storing all EMS-related data.

Query:

- CREATE DATABASE employeedb;

## 5.2 Using the Database :

Query:

- USE DATABASE employeedb;

## 5.2 Creating Tables for  Each Entities :

1. **Employee**

Query:

```
CREATE TABLE Employee (
    emp_ID INT(11) PRIMARY KEY,
    fname VARCHAR(255),
    lname VARCHAR(255),
    gender INT(11),
    age INT(11),
    contact_add INT(11),
    emp_email VARCHAR(255),
    emp_pass VARCHAR(255)
);
```

2. **Users**

Query:

```
CREATE TABLE Users (
    admin_ID INT(11) PRIMARY KEY,
    fname VARCHAR(255),
    lname VARCHAR(255),
    gender INT(11),
    age INT(11),
    contact_add INT(11),
    admin_email VARCHAR(255),
    admin_pass VARCHAR(255)
);
```

### 3. Job Department

Query:

-
  ```
  CREATE TABLE JobDepartment (
      job_ID INT(11) PRIMARY KEY,
      job_dept VARCHAR(30),
      name VARCHAR(30),
      description VARCHAR(30),
      salary_range VARCHAR(30)
  );
  ```

### 4. Salary/Bonous

Query:

-
  ```
  CREATE TABLE SalaryOrBonus (
      salary_ID INT(11) PRIMARY KEY,
      job_ID INT(11),
      amount INT(11),
      annual DATE,
      bonus DATE
  );
  ```

### 5. Qualification

Query:

-
  ```
  CREATE TABLE Qualification (
      qual_ID INT(11) PRIMARY KEY,
      emp_ID INT(11),
      position VARCHAR(30),
      requirements VARCHAR(30),
      date_in DATE);
  ```

6. **Payroll**

Query:

-
  ```
  CREATE TABLE Payroll (
     payroll_ID INT(11) PRIMARY KEY,
     emp_ID INT(11),
     job_ID INT(11),
     salary_ID INT(11),
     leave_ID INT(11),
     date DATE,
     report TEXT,
     total_amount INT(11)
  );
  ```

# 6.Data Manipulation

Once the database and tables are created, data can be manipulated using SQL queries. Below are examples of data insertion, selection, and modification.

## 6.1 Data Insertion

Data records for employee, users, salary, job departments, qualification and payroll are inserted into the respective tables.

## 6.2 Inserting Data for Each Tables :

1. **Employee Table:**

   Query:

-
  ```
  INSERT INTO Employee (emp_ID, fname, lname, gender, age, contact_add, emp_email, emp_pass)
  VALUES
      (1, 'John', 'Doe', 1, 30, 123456789, 'john.doe@email.com', 'password1'),
      (2, 'Jane', 'Smith', 2, 28, 987654321, 'jane.smith@email.com', 'password2');
  ```

2. **Users Table :**

   Query:

-
  ```
  INSERT INTO Users (admin_ID, fname, lname, gender, age, contact_add, admin_email, admin_pass)
  VALUES
      (1, 'Admin', 'User', 1, 40, 987654321, 'admin@email.com', 'adminpass1'),
      (2, 'Manager', 'Smith', 2, 35, 123456789, 'manager@email.com', 'managerpass2');
  ```

3. **Job Department Table:**

   Query:

- INSERT INTO JobDepartment (job_ID, job_dept, name, description, salary_range)
  VALUES
  (1, 'HR', 'HR Manager', 'Human Resources Manager', '50,000 - 80,000'),
  (2, 'Finance', 'Finance Analyst', 'Financial Analysis', '60,000 - 90,000');

4. **Salary/Bonous Table:**

Query:

- INSERT INTO SalaryOrBonus (salary_ID, job_ID, amount, annual, bonus)
  VALUES
  (1, 1, 60000, '2023-01-01', '2023-12-31'),
  (2, 2, 75000, '2023-01-01', '2023-12-31');

5. **Qualification Table :**

Query:

- INSERT INTO Qualification (qual_ID, emp_ID, position, requirements, date_in)
  VALUES
  (1, 1, 'Software Developer', 'Bachelor in Computer Science', '2022-01-15'),
  (2, 2, 'Accountant', 'Bachelor in Accounting', '2022-02-20');

6. **Payroll Table :**

Query:

- INSERT INTO Payroll (payroll_ID, emp_ID, job_ID, salary_ID, leave_ID, date, report, total_amount)
  VALUES
  (1, 1, 1, 1, NULL, '2023-01-31', 'January Payroll Report', 60000),
  (2, 2, 2, 2, NULL, '2023-01-31', 'January Payroll Report', 75000);

## 6.3 Select record for Each Tables :

We can select records from each table using SQL queries

1. **Employee Table:**

   To select all records from the Employee table, we can use the following query:

   - SELECT * FROM Employee;

2. **Users Table :**

   To select all records from the Users table, we can use the following query:

   - SELECT * FROM Users;

3. **Job Department Table:**

   To select all records from the Job Department table,  can use the following query:

   - SELECT * FROM JobDepartment;

4. **Salary/Bonous Table:**

   To select all records from the SalaryOrBonus table,  can use the following query:

   - SELECT * FROM SalaryOrBonus;

5. **Qualification Table :**

   To select all records from the Qualification table,  can use the following query:

- SELECT * FROM Qualification;

6. **Payroll Table :**

   To select all records from the Payroll table,  can use the following query:

   - SELECT * FROM Payroll;

### 6.4 Update record for Each Tables :

1. **Employee Table:**

   Query:

   - UPDATE Employee
     SET emp_email = 'new.email@email.com', age = 31
     WHERE emp_ID = 1;

2. **Users Table :**

   Query:

   - UPDATE Users
     SET admin_email = 'new.admin@email.com', age = 41
     WHERE admin_ID = 1;

3. **Job Department Table:**

   Query:

   - UPDATE JobDepartment
     SET description = 'Updated Description'
     WHERE job_ID = 1;

4. **Salary/Bonous Table:**

Query:

- UPDATE SalaryOrBonus
  SET annual = '2023-02-01', bonus = '2023-02-15'
  WHERE salary_ID = 1;

5. **Qualification Table :**

Query:

- UPDATE Qualification
  SET position = 'Updated Position', requirements = 'Updated Requirements'
  WHERE qual_ID = 1;

6. **Payroll Table :**

Query:

- UPDATE Payroll
  SET report = 'Updated Payroll Report', total_amount = 65000
  WHERE payroll_ID = 1;

### 6.4 Update record for Each Tables :

I. delete the employee with EmployeeID 1

II. delete employees in a specific department

Query:

- DELETE FROM Employee

  WHERE DepartmentID = 2;

III. Delete employees with a salary less than 60000 or greater than 75000

Query:

- DELETE FROM Employee

  WHERE Salary < 60000 OR Salary > 75000;

## 6.5 Some other Queries:

· **JOIN**: Combines rows from two or more tables based on a related column.

Query:

```
SELECT Employee.First_name, JobDepartment.Name

FROM Employee

JOIN Department ON Employee.DepartmentID = Department.DepartmentID;
```

· **GROUP BY**: Groups rows that have the same values into summary rows.

Query:

```
SELECT JobDepartmentID, AVG(Salary) AS AvgSalary

FROM Employee

GROUP BY JobDepartmentID;
```

· **ORDER BY**: Sorts the result set in ascending or descending order.

Query:

```
SELECT * FROM Employee

ORDER BY Last_name ASC;
```

· **COUNT**: Counts the number of rows in a result set.

Query:

```
SELECT COUNT(*) FROM Employee;
```

· **SUM**: Calculates the sum of a numeric column.

Query:

```
SELECT SUM(Salary) FROM Employee;
```

· **AVG**: Calculates the average of a numeric column.

Query:

```
SELECT AVG(Salary) FROM Employee;
```

· **MAX** and **MIN**: Retrieves the maximum and minimum values from a column.

Query:

```
SELECT MAX(Salary) FROM Employee;

SELECT MIN(Salary) FROM Employee;
```

· **DISTINCT Operator**: retrieve a list of distinct values for the "gender" column from the "Employee" table

Query:

```
SELECT DISTINCT gender

FROM Employee;
```

· **LIKE Operator**: retrieve employees whose last names start with "S,"

Query

```
SELECT *

FROM Employee

WHERE Last_name LIKE 'S%';
```