# **Employee Management System**

# **1. Project Overview**

The **Employee Management System** is a simple project built using **Python** and **MySQL**. It allows users to:

* Add new employees
* View all employees
* Update employee details (salary)
* Delete employees

This project runs entirely on the **command-line interface (CLI)** and does not have a graphical user interface (GUI). It is ideal for beginners to learn **Python-MySQL integration**.

## **2. Project Objective**

* Understand **CRUD operations** (Create, Read, Update, Delete) in Python with MySQL.
* Learn to **connect Python programs to a MySQL database**.
* Practice **writing SQL queries** from Python.
* Develop a **menu-driven CLI application**.

## **3. Requirements**

* **Python 3.x**
* **MySQL Server**

Python library: mysql-connector-python

## **4. Database Design**

We need a single table employee with the following fields:

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | INT AUTO\_INCREMENT | Primary key, unique ID |
| name | VARCHAR(100) | Employee full name |
| age | INT | Employee age |
| salary | DECIMAL(10,2) | Employee salary |

**SQL Code to Create Database and Table:**

**CREATE DATABASE empdb;**

**USE empdb;**

**CREATE TABLE employee (**

**id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100),**

**age INT,**

**salary DECIMAL(10,2)**

**);**

## **5. Python Code**

Save the following code as simple.py:

**import mysql.connector**

# Connect to MySQL

**conn = mysql.connector.connect(**

**host="localhost",**

**user="root", # replace with your MySQL username**

**password="", # replace with your MySQL password**

**database="empdb"**

**)**

**cur = conn.cursor()**

# Add a new employee

**def add\_employee():**

**name = input("Enter name: ")**

**age = int(input("Enter age: "))**

**salary = float(input("Enter salary: "))**

**cur.execute("INSERT INTO employee (name, age, salary) VALUES (%s, %s, %s)", (name, age, salary))**

**conn.commit()**

**print("Employee added successfully!")**

# View all employees

**deff view\_employees():**

**cur.execute("SELECT \* FROM employee")**

**rows = cur.fetchall()**

**for row in rows:**

**print(row)**

# Update employee salary

**def update\_employee():**

**emp\_id = int(input("Enter employee id to update: "))**

**new\_salary = float(input("Enter new salary: "))**

**cur.execute("UPDATE employee SET salary=%s WHERE id=%s", (new\_salary, emp\_id))**

**conn.commit()**

**print("Employee updated successfully!")**

# Delete an employee

**def delete\_employee():**

**emp\_id = int(input("Enter employee id to delete: "))**

**cur.execute("DELETE FROM employee WHERE id=%s", (emp\_id,))**

**conn.commit()**

**print("Employee deleted successfully!")**

# Menu

**def menu():**

**while True:**

**print("\n--- Employee Management ---")**

**print("1. Add Employee")**

**print("2. View Employees")**

**print("3. Update Employee")**

**print("4. Delete Employee")**

**print("5. Exit")**

**choice = input("Enter choice: ")**

**if choice == '1':**

**add\_employee()**

**elif choice == '2':**

**view\_employees()**

**elif choice == '3':**

**update\_employee()**

**elif choice == '4':**

**delete\_employee()**

**elif choice == '5':**

**break**

**else:**

**print("Invalid choice!")**

**menu()**

**cur.close()**

**conn.close()**

## **6. How the Code Works**

**Connecting to MySQL** conn = mysql.connector.connect(...)

cur = conn.cursor()

* + conn is the connection to MySQL.
  + cur is the cursor used to execute SQL queries.

**Adding Employee** cur.execute("INSERT INTO employee ...")

conn.commit()

* + Inserts a new record into the employee table.
  + conn.commit() saves the changes in the database.

**Viewing Employees** cur.execute("SELECT \* FROM employee")

rows = cur.fetchall()

* + Fetches all records from the table.
  + cur.fetchall() returns all rows as a list of tuples.

**Updating Employee** cur.execute("UPDATE employee SET salary=%s WHERE id=%s", (new\_salary, emp\_id))

* + Updates the salary of a specific employee.

**Deleting Employee** cur.execute("DELETE FROM employee WHERE id=%s", (emp\_id,))

* + Removes the employee record from the database.

1. **Menu Loop**
   * Provides a simple CLI interface to choose operations.
   * Loops until user selects Exit.

## **7. Sample Output**

--- Employee Management ---

1. Add Employee

2. View Employees

3. Update Employee

4. Delete Employee

5. Exit

Enter choice: 1

Enter name: John Doe

Enter age: 28

Enter salary: 50000

Employee added successfully!

Enter choice: 2

(1, 'John Doe', 28, 50000.0)

Enter choice: 3

Enter employee id to update: 1

Enter new salary: 55000

Employee updated successfully!

Enter choice: 4

Enter employee id to delete: 1

Employee deleted successfully!

### **✨ Extra Features**

1. **More Data Fields** – department, role, salary, date of joining.
2. **Search & Filter** – by ID, name, department, or joining date.
3. **Advanced Updates** – update specific fields (like salary only).
4. **Reports** – employee count by department, highest/lowest salary, average salary.
5. **Validation** – no duplicate IDs, positive salary, valid departments.
6. **Backup/Export** – save employee data to CSV and restore it later.

## **🔹 15 Project Ideas for Students**

1. **Employee Management System**
   * Add / View / Update / Delete employees
   * Search by ID, department
2. **Library Management System**
   * Manage books, issue & return
   * Track borrowed books
3. **Student Result Management**
   * Store marks, calculate total & grade
   * Find toppers, subject-wise highest
4. **Inventory Management System**
   * Manage stock of products
   * Generate low-stock reports
5. **Hospital Patient Record System**
   * Manage patients, doctors
   * Assign patients to doctors
6. **Attendance Management System**
   * Mark daily attendance
   * Calculate attendance percentage
7. **Banking System**
   * Create accounts
   * Deposit, withdraw, balance check
8. **Hotel Booking / Room Management**
   * Manage rooms & customers
   * Checkout & bill generation
9. **Course Enrollment System**
   * Add courses & students
   * Enroll students in courses
10. **Transport/Bus Booking System**

* Book bus seats
* Cancel ticket, check availability

1. **Exam Seating Arrangement System**

* Assign students to rooms/seats
* Generate seating reports

1. **Gym/Fitness Membership Management**

* Add members, track subscription
* Renewal reminders

1. **Restaurant Order Management**

* Take customer orders
* Generate final bill

1. **Complaint Management System**

* Register complaints (IT, Hostel, Maintenance)
* Track complaint status (Pending/Resolved)

1. **Event Management System**

* Add events & participants
* Track registrations and winners