EMPLOYEE MANAGEMENT SYSTEM

MYSQL + PYTHON

OVERVIEW

The Employee Management System is a command-line application designed to manage employee records using a MySQL database. This project allows users to perform basic CRUD (Create, Read, Update, Delete) operations on employee data. The system supports adding new employees, removing existing employees, promoting employees by increasing their salary, and displaying a list of all employees.

FEATURES

- **Add Employee**: Enter details including ID, name, position, and salary to add a new employee to the system.
- **Remove Employee**: Remove an employee from the system based on their ID.
- **Promote Employee**: Increase the salary of an existing employee.
- **Display Employees**: View details of all employees currently in the database.

TECHNOLOGIES USED

- **Python**: Programming language used to implement the application logic.
- **MySQL**: Database management system for storing employee records.
- **mysql-connector-python**: Python library used to interact with the MySQL database.

APPLICATIONS USED:

SQL: MYSQL Workbench

Python Coding: Jupiter Notebook

INSTALL DEPENDENCIES:

Python and MySQL Connector installation. Installating the MySQL Connector using pip:

!pip install mysql-connector-python

DATABASE CONFIGURATION:

- Ensuring MySQL Server is running.
- Create a database named **employee_management_system** in MySQL.
- Create a table employees with the following schema:

```
CREATE TABLE EMPLOYEES (
    ID VARCHAR(50) PRIMARY KEY,
    NAME VARCHAR(100),
    POSITION VARCHAR(100),
    SALARY DECIMAL(10, 2)
);
```

```sql
create database employee_management_system;
use employee_management_system;
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(50),
    salary DECIMAL(10, 2)
);
```

MySQL®

IMPLIMENTING PYTHON CODE :

1) DATABASE CONNECTION

THE SCRIPT CONNECTS TO A MYSQL DATABASE USING THE MYSQL-CONNECTOR-PYTHON LIBRARY

```
import mysql.connector

try:
    con = mysql.connector.connect(
        host="localhost",
        user="root",
        password="12345",
        database="employee_management_system"
    )
```

jupyter

2) ERROR HANDLING

THE SCRIPT HANDLES CONNECTION ERRORS AND ENSURES THAT DATABASE OPERATIONS ARE PERFORMED ONLY IF THE CONNECTION IS SUCCESSFUL:

```
except mysql.connector.Error as err:
    print(f"Error: Could not connect to database: {err}")
    exit(1)
```

FUNCTIONS

CHECK_EMPLOYEE(EMPLOYEE_ID)

CHECKS IF AN EMPLOYEE WITH THE GIVEN ID EXISTS IN THE DATABASE:

```
def check_employee(employee_id):
    sql = 'SELECT * FROM employees WHERE id=%s'
    cursor.execute(sql, (employee_id,))
    result = cursor.fetchall()
    return len(result) > 0
```

ADD_EMPLOYEE()

ADDS A NEW EMPLOYEE BY COLLECTING DETAILS FROM USER INPUT: PROMPTS FOR EMPLOYEE ID, NAME, POSITION, AND SALARY.
VALIDATES SALARY INPUT.
INSERTS THE EMPLOYEE RECORD INTO THE DATABASE.

```
def add_employee():
    Id = input("Enter Employee Id: ")
    if check_employee(Id):
        print("Employee already exists. Please try again.")
        return

    Name = input("Enter Employee Name: ")
    Post = input("Enter Employee Post: ")

    try:
        Salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Invalid Salary! Please enter a valid number.")
        return

    sql = 'INSERT INTO employees (id, name, position, salary) VALUES (%s, %s, %s, %s)'
    data = (Id, Name, Post, Salary)

    try:
        cursor.execute(sql, data)
        con.commit()
        print("Employee Added Successfully")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        con.rollback()
```

REMOVE_EMPLOYEE()

REMOVES AN EMPLOYEE BASED ON THE GIVEN ID:

- CHECKS IF THE EMPLOYEE EXISTS.
- DELETES THE RECORD FROM THE DATABASE.

```
def remove_employee():
    Id = input("Enter Employee Id: ")
    if not check_employee(Id):
        print("Employee does not exist. Please try again.")
        return

    sql = 'DELETE FROM employees WHERE id=%s'
    data = (Id,)

    try:
        cursor.execute(sql, data)
        con.commit()
        print("Employee Removed Successfully")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        con.rollback()
```

PROMOTE_EMPLOYEE()

INCREASES THE SALARY OF AN EXISTING EMPLOYEE:

- PROMPTS FOR THE ID AND THE AMOUNT TO INCREASE THE SALARY.
- UPDATES THE SALARY IN THE DATABASE.

```
def promote_employee():
    Id = input("Enter Employee's Id: ")
    if not check_employee(Id):
        print("Employee does not exist. Please try again.")
        return

    try:
        Amount = float(input("Enter increase in Salary: "))
        sql_select = 'SELECT salary FROM employees WHERE id=%s'
        cursor.execute(sql_select, (Id,))
        current_salary = cursor.fetchone()[0]
        new_salary = current_salary + Amount

        sql_update = 'UPDATE employees SET salary=%s WHERE id=%s'
        cursor.execute(sql_update, (new_salary, Id))
        con.commit()
        print("Employee Promoted Successfully")

    except (ValueError, mysql.connector.Error) as e:
        print(f"Error: {e}")
        con.rollback()
```

DISPLAY_EMPLOYEES()

DISPLAYS ALL EMPLOYEE RECORDS:

- FETCHES AND PRINTS DETAILS OF ALL EMPLOYEES FROM THE DATABASE.

```
def display_employees():
    try:
        sql = 'SELECT * FROM employees'
        cursor.execute(sql)
        employees = cursor.fetchall()
        for employee in employees:
            print("Employee Id : ", employee[0])
            print("Employee Name : ", employee[1])
            print("Employee Post : ", employee[2])
            print("Employee Salary : ", employee[3])
            print("-----------------------------------")

    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

MAIN MENU

THE MENU() FUNCTION PROVIDES A SIMPLE TEXT-BASED INTERFACE FOR THE USER TO CHOOSE OPERATIONS:

```
def menu():
    while True:
        print("\nWelcome to Employee Management Record")
        print("Press:")
        print("1 to Add Employee")
        print("2 to Remove Employee")
        print("3 to Promote Employee")
        print("4 to Display Employees")
        print("5 to Exit")

        ch = input("Enter your Choice: ")

        if ch == '1':
            add_employee()
        elif ch == '2':
            remove_employee()
        elif ch == '3':
            promote_employee()
        elif ch == '4':
            display_employees()
        elif ch == '5':
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid Choice! Please try again.")
```

CLEANUP

THE SCRIPT ENSURES THAT THE DATABASE CONNECTION AND CURSOR ARE CLOSED WHEN THE PROGRAM EXITS:

```
if __name__ == "__main__":
    try:
        menu()
    finally:
        cursor.close()
        con.close()
```

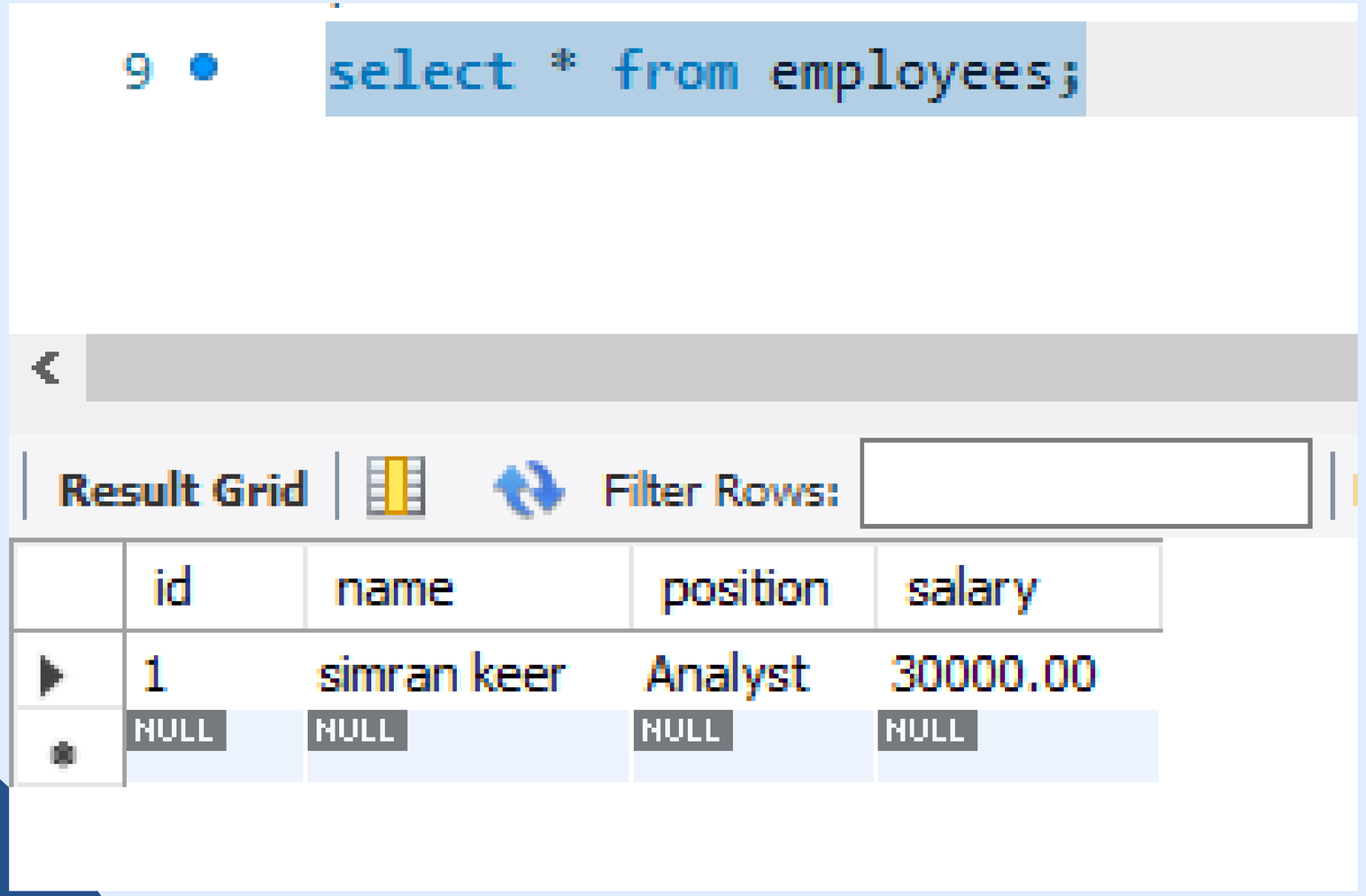
RESULT:

```
Welcome to Employee Management Record
Press:
1 to Add Employee
2 to Remove Employee
3 to Promote Employee
4 to Display Employees
5 to Exit
Enter your Choice:
```

LET’S TRY INPUT :

```
Welcome to Employee Management Record
Press:
1 to Add Employee
2 to Remove Employee
3 to Promote Employee
4 to Display Employees
5 to Exit
Enter your Choice:  1
Enter Employee Id:  1
Enter Employee Name:  simran keer
Enter Employee Post:  Analyst
Enter Employee Salary:  30000
Employee Added Successfully
```

RESULT SQL WORKBENCH:



THANK YOU

simran.keer7@gmail.com