Database Connectivity: Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

ANS=

Step 1: Install Required Packages

You need to install the mysql-connector-python package to connect to a MySQL database. You can install it using pip:

bash

Copy code

pip install mysql-connector-python

If you are connecting to an Oracle database, you can use the cx_Oracle library:

bash

Copy code

pip install cx_Oracle

Step 2: Set Up the Database

Before we create the program, ensure that you have a MySQL database set up with a table named Employees. Here's a simple table creation SQL:

sql

Copy code

```
CREATE DATABASE company;


USE company;


CREATE TABLE Employees (
  emp_id INT AUTO_INCREMENT PRIMARY KEY,
  emp_name VARCHAR(100) NOT NULL,
  emp_salary DECIMAL(10, 2) NOT NULL
);
```

Step 3: Create the Python Program

Below is the complete Python program that implements a simple GUI for adding, deleting, and editing employees in the Employees table.

python

Copy code

```
import mysql.connector
from mysql.connector import Error
import tkinter as tk
from tkinter import messagebox, simpledialog

class DatabaseApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Employee Database")

        # Database Connection
        self.connection = self.connect_to_database()

        # GUI Components
        self.create_widgets()

    def connect_to_database(self):
        try:
            connection = mysql.connector.connect(
                host='localhost',
                database='company',
                user='your_username',  # Replace with your MySQL username
                password='your_password'  # Replace with your MySQL password
            )
            if connection.is_connected():
                print("Connected to MySQL Database")
                return connection
        except Error as e:
            print(f"Error: {e}")
            return None
```

```
  def create_widgets(self):
    self.emp_id_label = tk.Label(self.root, text="Employee ID:")
    self.emp_id_label.grid(row=0, column=0)
    self.emp_id_entry = tk.Entry(self.root)
    self.emp_id_entry.grid(row=0, column=1)

    self.emp_name_label = tk.Label(self.root, text="Employee Name:")
    self.emp_name_label.grid(row=1, column=0)
    self.emp_name_entry = tk.Entry(self.root)
    self.emp_name_entry.grid(row=1, column=1)

    self.emp_salary_label = tk.Label(self.root, text="Employee Salary:")
    self.emp_salary_label.grid(row=2, column=0)
    self.emp_salary_entry = tk.Entry(self.root)
    self.emp_salary_entry.grid(row=2, column=1)

    self.add_button = tk.Button(self.root, text="Add", command=self.add_employee)
    self.add_button.grid(row=3, column=0)

    self.delete_button = tk.Button(self.root, text="Delete", command=self.delete_employee)
    self.delete_button.grid(row=3, column=1)

    self.edit_button = tk.Button(self.root, text="Edit", command=self.edit_employee)
    self.edit_button.grid(row=3, column=2)

    self.show_button = tk.Button(self.root, text="Show Employees",
command=self.show_employees)
    self.show_button.grid(row=4, column=0, columnspan=3)

  def add_employee(self):
    emp_name = self.emp_name_entry.get()
```

```
        emp_salary = self.emp_salary_entry.get()
        try:
            cursor = self.connection.cursor()
            cursor.execute("INSERT INTO Employees (emp_name, emp_salary) VALUES (%s, %s)", (emp_name, emp_salary))
            self.connection.commit()
            messagebox.showinfo("Success", "Employee added successfully")
            self.clear_entries()
        except Error as e:
            messagebox.showerror("Error", str(e))

    def delete_employee(self):
        emp_id = self.emp_id_entry.get()
        try:
            cursor = self.connection.cursor()
            cursor.execute("DELETE FROM Employees WHERE emp_id = %s", (emp_id,))
            self.connection.commit()
            messagebox.showinfo("Success", "Employee deleted successfully")
            self.clear_entries()
        except Error as e:
            messagebox.showerror("Error", str(e))

    def edit_employee(self):
        emp_id = self.emp_id_entry.get()
        emp_name = self.emp_name_entry.get()
        emp_salary = self.emp_salary_entry.get()
        try:
            cursor = self.connection.cursor()
            cursor.execute("UPDATE Employees SET emp_name = %s, emp_salary = %s WHERE emp_id = %s", (emp_name, emp_salary, emp_id))
            self.connection.commit()
            messagebox.showinfo("Success", "Employee updated successfully")
```

```
            self.clear_entries()
        except Error as e:
            messagebox.showerror("Error", str(e))

    def show_employees(self):
        try:
            cursor = self.connection.cursor()
            cursor.execute("SELECT * FROM Employees")
            records = cursor.fetchall()
            employee_list = ""
            for row in records:
                employee_list += f"ID: {row[0]}, Name: {row[1]}, Salary: {row[2]}\n"
            messagebox.showinfo("Employees", employee_list)
        except Error as e:
            messagebox.showerror("Error", str(e))

    def clear_entries(self):
        self.emp_id_entry.delete(0, tk.END)
        self.emp_name_entry.delete(0, tk.END)
        self.emp_salary_entry.delete(0, tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    app = DatabaseApp(root)
    root.mainloop()
```

Step 4: Run the Program

1. **Configure Database Connection:**
    - Replace your_username and your_password in the connection section of the code with your actual MySQL credentials.
2. **Run the Program:**
    - Execute the Python script. A GUI window will appear where you can add, delete, or edit employee records.

Summary

This program provides a simple GUI to interact with a MySQL database for managing employee records. You can perform the following operations:

- **Add Employee**: Enter the employee's name and salary, and click "Add."
- **Delete Employee**: Enter the employee ID and click "Delete."
- **Edit Employee**: Enter the employee ID, name, and salary, and click "Edit."
- **Show Employees**: Click "Show Employees" to see a list of all employees in the database.

Additional Considerations

- **Error Handling**: Ensure proper error handling for cases like duplicate entries or invalid inputs.
- **UI Enhancements**: You can further enhance the UI and features based on your requirements.
- **Oracle Database**: If you want to implement this for an Oracle database, you would need to adjust the connection and SQL statements accordingly.