Phonebook project

Project using pymysql
===================

INTRODUCTION:-
=============
Introduction to the Phonebook Project
In today's digital age, managing contact information efficiently is crucial for both personal and professional interactions. The Phonebook Project is a straightforward yet effective solution designed to handle basic contact management needs using a relational database system. This project aims to provide users with a reliable tool for storing, updating, deleting, and viewing contact information through a user-friendly command-line interface.

Objectives:-
=========
The primary objectives of the Phonebook Project are:

=>To Create a Structured Database: Develop a relational database schema that organizes contact information systematically, allowing for efficient storage and retrieval.

=>To Implement Core CRUD Operations: Provide functionalities to Create, Read, Update, and Delete (CRUD) contact records, ensuring comprehensive management of contact information.

=>To Offer a Simple User Interface: Design a command-line interface that facilitates user interactions with the phonebook, making it easy to perform essential operations.

Key Features:-
===========
=>Add New Contacts: Users can input and store new contact details, including name, phone number, and email address.

=>Update Existing Contacts: Users can modify the details of existing contacts using a unique contact ID.

=>Delete Contacts: Users can remove contacts from the database when they are no longer needed.

=>View Contacts: Users can view all stored contacts, which helps in managing and reviewing contact information easily.

CREATING TABLE:-
===============
The SQL commands is to create a new database named `phonebook` and set it as the active database. Within this database, a table called `contacts` is created. This table includes four columns: `id`, which is an auto-incrementing integer and it is the primary key; `name`, a variable character field with a maximum length of 100 characters; `phone_number`, an integer field; and `email`, another variable character field with a maximum length of 100 characters.

CONNECT THE DATABASE:-
======================
The connect function establishes a connection to a MySQL database named “PHONEBOOK” using the pymysql library. It connects to the database server running on localhost with the username “root” and an empty password. Once the connection is successfully established, it creates a cursor object c using the cursor() method of the connection object d. The function then returns both the cursor and the connection objects. The cursor is used to execute SQL queries, while the connection object is used to manage the database connection, including committing transactions or rolling back changes if necessary.

ADD THE CONTACT:-
================
The add_Contact function is designed to add a new contact to a database. It begins by establishing a connection to the database using the connect() function, which returns a cursor c and a connection object d. The SQL query for inserting a new contact’s details (name, phone number, and email) into the Contacts table is prepared, with placeholders for the values. The execute method of the cursor is then called with the SQL query and the provided contact details. If the insertion is successful, the changes are committed to the database using d.commit(), and a success message is printed. If an error occurs during the process, the except block is executed, which rolls back any changes made during the transaction using d.rollback(), ensuring the database remains in a consistent state.

UPDATE THE CONTACT:-
====================
The update_Contact function is designed to update a contact’s details in a database. It takes four parameters: Contact_id, name, phone_number, and email. Inside the function, it first establishes a connection to the database using the connect() function, which returns a cursor c and a connection object d. The SQL query for updating the contact’s information is prepared with placeholders for the new values. The execute method of the cursor is then called with the SQL query and the provided parameters to update the contact’s details where the id matches Contact_id. If the update is successful, the changes are committed to the database with d.commit(), and a success message is printed. If an error occurs, the changes are rolled back using d.rollback() to maintain database integrity

DELETE THE CONTACT:-
===================
The delete_Contact function is designed to remove a contact from the database based on the provided Contact_id. It establishes a connection to the database using the connect() function, which returns a cursor c and a connection object d. The SQL query for deleting the contact is prepared with a placeholder for the Contact_id. The execute method of the cursor runs this query to delete the contact where the id matches Contact_id. If the deletion is successful, the changes are committed to the database with d.commit(), and a success message is printed. If an error occurs, the changes are rolled back using d.rollback() to ensure database integrity.

VIEW THE CONTACT:-
=================
The view_Contacts function connects to a database and retrieves all records from the contacts table. It executes an SQL query to fetch all contact details, then checks if any contacts are found.

If contacts exist, it prints each contact’s ID, name, phone number, and email. If no contacts are found, it prints a message indicating this. Finally, the function ensures the database connection is closed, regardless of whether an exception occurred

The provided main function for a phonebook application is designed to manage contacts through a menu-driven interface. It begins by establishing a connection to the data storage using the connect() function. The menu offers five options: adding, updating, deleting, viewing contacts, and exiting the application. Each option prompts the user for necessary details, such as contact name, phone number, and email, and calls corresponding functions (add_Contact, update_Contact, delete_Contact, view_Contacts). The code includes basic error handling to ensure valid numeric inputs for phone numbers and contact IDs, enhancing user experience by preventing invalid entries. The loop continues until the user chooses to exit, ensuring continuous interaction with the phonebook.

Conclusion:-
==========
The Phonebook Project offers a practical and effective solution for managing contact information through a simple command-line interface and a well-structured relational database. By leveraging MySQL for data storage and Python with the pymysql library for database interactions, this project demonstrates fundamental principles of database management and application development.