**REQUIREMENTS:**

Libraries and Packages Used:

1. pandas, pandasai: For handling data manipulation and visualization.
2. os: For environment management and file handling.
3. langchain: For working with language models (e.g., ChatGoogleGenerativeAI).
4. streamlit: For creating the web application interface.
5. sqlite3: For connecting to and querying SQLite databases.
6. dotenv: For loading environment variables (e.g., API keys).
7. PIL: For image handling and visualization.
8. pyngrok: For exposing the app to the internet via an external URL (for deployment).

Ensure Dependencies are Installed:

* The following packages should be installed:
* streamlit
* sqlite3
* langchain
* pandas
* pandasai
* pyngrok
* PIL
* **Set Up API Keys:** You will need a valid GOOGLE\_API\_KEY and PANDASAI\_API\_KEY. Make sure to replace the placeholder keys in the code with valid ones.
* **Prepare SQLite Database:** The code expects a SQLite database file (.db or .sqlite). Ensure your database has the appropriate tables and data for querying.
* **Running Streamlit Locally**: Run the app by executing the following command: streamlit run QueryData\_AI.py
* Must have ngrok Authtoken
* Ensure you have ngrok installed for exposing the app via an external URL
* **Avoid Re-running on Execution**: Streamlit may rerun the app on every interaction (button press). To prevent data loss or errors, use Streamlit's session state properly to maintain the connection and results across interactions.

**WORKING OF THE CODE.**

The code creates a web application using Streamlit that allows users to upload a database and generate SQL queries automatically based on user input. At its core, the app connects to an SQLite database, extracts schema information dynamically, generates SQL queries using a large language model (LLM), executes those queries on the database, and visualizes the results in a user-friendly format. The combination of tools like LangChain, Google Generative AI, and PandasAI helps streamline this process, making it possible to automate SQL query generation and result interpretation with minimal user input.

The script begins by importing necessary libraries, such as Streamlit for building the web interface, SQLite for managing the database, and various components from LangChain and PandasAI for natural language processing and data visualization. It also sets up API keys and environment variables for accessing services like Google Generative AI and PandasAI, which are essential for generating queries and visualizations. After the imports and setup, the app defines several prompt templates that structure how SQL queries and responses are generated. These templates ensure that the generated queries are accurate, concise, and tailored to the user’s question while aligning with the database schema.

The core functionality of the app lies in its ability to dynamically extract schema information from an uploaded SQLite database. When a user uploads a database, the app reads the database structure, retrieves table and column information, and displays the available tables to the user. This schema extraction enables the app to generate accurate SQL queries by understanding the structure of the database. After connecting the database, users can input natural language questions about the data. The app processes these questions using the LLM, which generates SQL queries based on the schema and user input. These queries are displayed to the user and can be executed against the database with the click of a button.

Once the SQL query is executed, the app retrieves and displays the results in a table format using pandas. Additionally, it employs PandasAI’s Agent to analyze the data further and generate visualizations if necessary. For example, the user may receive a graph or chart depending on the nature of the data. The app also provides a natural language summary of the SQL query results, offering a concise, human-readable explanation of the findings. This combination of automation and visualization allows users to interact with their database in a highly intuitive manner, without requiring deep technical knowledge of SQL.

Finally, the app is designed to offer a seamless experience with features like reconnecting to a database, exiting the session, and generating visual data representations. A key part of the script is the ngrok integration, which allows the app to be exposed publicly, making it accessible to users via a shared link. This is particularly useful for quick prototyping or demonstrations, as it eliminates the need for complex server setups. Overall, the code effectively combines SQL query generation, database interaction, and data visualization into a single web interface, leveraging modern AI tools to simplify these processes.