Sales Q&A Chatbot

Objective:

The primary objective of this chatbot is to **convert natural language queries into SQL queries** and retrieve corresponding data from a SQLite database. This is designed to help users, particularly those without extensive SQL knowledge, interact with a database via simple English questions. The chatbot will use **Google Gemini** to generate the SQL queries based on the user's input, which can then be executed on the SQLite database.

Overview

The chatbot integrates **Google Gemini**, **Streamlit**, and **SQLite** to convert user questions into SQL queries and fetch the results from a database. This system is built to help non-technical users retrieve information from a database by simply typing questions in natural language.

Core Technologies:

- **Google Gemini API**: Provides natural language processing (NLP) capabilities to interpret and convert English text into SQL queries.
- **Streamlit**: A Python-based tool for creating interactive web applications, used to create the user interface for this chatbot.
- **SQLite**: A lightweight, serverless database to store and retrieve data.

Problem Statement

The challenge is to enable non-technical users to access and query a database (in this case, an **E-commerce dataset** stored in an SQLite database) without needing to know SQL. Users often need insights from data stored in databases but may lack the skills to write SQL queries themselves.

Key Challenges:

- **Natural Language to SQL Conversion**: Translating a variety of user inputs into syntactically correct SQL queries.

Workflow

1. **User Input**:
    - The user inputs a question in natural language, e.g., "How many records are in the customers table?"
2. **Query Generation**:
    - The chatbot sends the user's question along with a **prompt** to the Gemini API. Gemini uses the prompt to understand the context of the database and generates an appropriate SQL query.
3. **SQL Query Execution**:

        - The generated SQL query is cleaned and executed against the **SQLite database** (E-commerce dataset.sqlite), using the sqlite3 Python library.
4. **Display Results**:
    - The results of the query are fetched and displayed to the user via the Streamlit interface.

User Personas

Persona 1: Business Analyst

- **Name**: Jack
- **Occupation**: Business Analyst
- **Skill Level**: Intermediate with data analysis tools but not proficient in SQL.
- **Goals**:
    - Quickly retrieve insights from large datasets without needing to write complex SQL queries.
    - Focus on analyzing business metrics rather than learning database query languages.
- **Pain Points**:
    - Struggles with SQL syntax and complex query generation.

Persona 2: E-commerce Manager

- **Name**: David
- **Age**: 40
- **Occupation**: E-commerce Manager
- **Skill Level**: Beginner with data analysis tools and SQL.
- **Goals**:
    - Want to retrieve real-time data about sales, customer behaviour, and product performance.
    - Needs a simple way to query data without SQL knowledge.
- **Pain Points**:
    - Difficulty in accessing specific data from the database.

Code overview:

The application is built in Python and uses several libraries to connect the various components:

Libraries Used:

- **Streamlit**: For building the user interface.
- **google.generativeai**: For integrating the Gemini API to generate SQL queries from natural language.
- **sqlite3**: To interact with the SQLite database.
- **re**: For cleaning up the SQL queries before execution.

Errors Encountered and Resolved

1. **Error 1: google.api_core.exceptions.InvalidArgument: 400**
   - **Cause**: The API call was using an incorrect or outdated model name (e.g., 'Gemini API'), or the generateContent method was not available for the specified model.
   - **Resolution**: After checking the available models in the Gemini API, the model name was updated to gemini-1.5-flash, which resolved the error. I also checked the documentation to ensure the correct API endpoint and method were used.
2. **Error 2: SQL Query Formatting Issues**
   - **Cause**: The generated SQL queries from Gemini included Markdown formatting (````sql`) that caused errors when executed in SQLite.
   - **Resolution**: I implemented a clean_sql_query() function that strips away the unwanted formatting and ensures that only the raw SQL query is passed to the SQLite database.
3. **Error 3: google.api_core.exceptions.NotFound: 404**
   - **Cause**: The API call used an incorrect model version or API endpoint.
   - **Resolution**: This issue was fixed by correctly setting up the Gemini API version and specifying the correct model in the API call.
4. **Error 4: TypeError: Could not create Blob**
   - **Cause**: This error occurred when trying to render the SQL result as a visual output, likely due to passing a list or invalid object to a visualization component.
   - **Resolution**: I ensured that the query result was in a proper format (a list of tuples) before trying to display it in Streamlit, allowing proper rendering.

Conclusion:

This chatbot provides a seamless interface for users to interact with a database using natural language. By integrating the **Google Gemini API** with **Streamlit** and **SQLite**, the system can dynamically convert questions into SQL queries and fetch data from the database, making it accessible to users with little to no SQL knowledge.

With this chatbot, users can now effortlessly extract meaningful insights from the database by simply asking questions, significantly enhancing productivity for business analysts, e-commerce managers, and other non-technical stakeholders.