Chat Assistant for SQLite Database

Note: Please record your intro using this link if you have not already.

Objective

To evaluate your ability to design, implement, and deploy a chat assistant that interacts with an SQLite database to answer user queries.

Scope and Requirements

Database

Create an SQLite database file containing two tables with the following schema:

You are expected to use this database as the foundation for the assignment. Feel free to add more data and columns.

Table 1: Employees

ID	Name	Department	Salary	Hire_Date
1	Alice	Sales	50000	2021-01-15
2	Bob	Engineering	70000	2020-06-10
3	Charlie	Marketing	60000	2022-03-20

Table 2: Departments

ID	Name	Manager
1	Sales	Alice
2	Engineering	Bob

3	Marketing	Charlie

Functionality

Build a Python-based chat assistant that:

- Accepts natural language queries.
- Converts queries into SQL to fetch data from the provided SQLite database.
- Responds to the user with clear, formatted answers.

Supported Queries:

The chat assistant must support the following types of queries:

- “Show me all employees in the [department] department.”
- “Who is the manager of the [department] department?”
- “List all employees hired after [date].”
- “What is the total salary expense for the [department] department?”
- & more…

Error Handling:

Ensure that the assistant can:

- Gracefully handle invalid queries.
- Return meaningful messages when no results are found.
- Handle incorrect department names or invalid input formats.

Deliverables

- Hosted Link: A public URL to access and test your deployed chat assistant.
- Code **Repository:** A GitHub (or equivalent) repository containing:
  - The source code for the assistant.
  - The database file
  - A README.md file with:
    - An explanation of how the assistant works.
    - Steps to run the project locally.
    - Known limitations or suggestions for improvement.

Evaluation Criteria

- **Correctness:** Does the assistant correctly answer the supported queries?

- **Deployment:** Is the application hosted properly and accessible without issues?
- **Code Quality:** Is the code modular, readable, and well-documented?
- **Error Handling:** Does the assistant gracefully handle edge cases and invalid inputs?
- **User Experience:** Is the hosted application intuitive and user-friendly?

Notes for Candidates

- Use Python and any framework or libraries of your choice (e.g., Flask, FastAPI).
- Ensure that your deployed version is functional and provides a seamless user experience.
- Focus on simplicity, clarity, and robustness in your solution.
- This solution needs to work without any third-party API. You may download open-source models and use them.