Take-Home Assignment: SQL Query Application

Objective:

Develop a system that allows users to query a SQL database using natural language queries. The system will process a dataset provided as a CSV file (`data.csv`), load it into a SQL database, and use a pre-trained text-to-SQL model to translate natural language queries into SQL. The application will execute the SQL queries on the database and return a human-readable answer via a user interface.

Tasks:

Database Service: load the provided CSV dataset into a SQL database and expose the database via an API endpoint for querying.

Text-to-SQL Model Service: utilize a pre-trained text-to-SQL model to translate natural language queries into SQL. You may use an open-source or pre-trained model such as SQLCoder-2–7b, T5-base, any Hugging Face models, or any other option you see fit. **This service should HOST the aforementioned model and output inference through it.** Expose an API endpoint to receive a natural language query and any other endpoint you believe necessary.

User Interface Service: create a user interface (web or API-based) where users can input a natural language query (e.g., "What is the most bought product on Fridays?"). Display the sql result after processing.

Bonus task [OPTIONAL] - LLM Model service: host a second model, an LLM that will translate the query results into human readable language making the system transparent to the end user. This means, the user asks in natural language “What is the most bought product on Fridays?" and receives after some seconds “Alfajor 70 cacao x un is the most bought product on Fridays” written in natural language.

Requirements:

- Code should be in **Python**.
- You may implement as many services as needed. Either a service per task, one service (monolith) for all the tasks or how many you deem fit.
  Simplicity is highly desirable and valuable. (KISS principle).
- Each service, job, or database should run within its own Docker container.
- Use Docker Compose to orchestrate and run all services locally.
- Ensure proper communication between services, using networking or shared volumes as needed.
- Provide clear documentation and setup instructions (README.md is a good place for this)

Deliverables:

- Source code for all services. Preferably a github link (other repos are also fine).
- A **docker-compose.yml** file in the repo to run all services locally.
- Instructions on how to build and run the application.
- You should be able to explain how you would scale out the architecture if there where bigger and more tables; and also high traffic to the front interface. You can add this as a section to the README.md

Recommendations:

- Focus on modularity and clear separation of concerns between services.
- Avoid using an external service API (OpenAi e.g.) for the model part, unless you want to include it as an extra functionality for benchmarking sake. To clarify, if you want to add an option to switch between the model you hosted and OpenAI's API (for example) so comparison becomes possible, you can do it. But prioritize hosting the model, since that is what I'm interested in.
- Ensure that services are resilient and handle errors gracefully.
- Write clean, maintainable code with appropriate documentation and comments.
- Start with the simple tasks such as downloading the creating and hosting the db, then move on to more complex ones like dealing with the text-to-sql model.
- It is expected to use chatGPT, Claude or any other LLM for help with this project.
- Feel free to write to me @ alex@nivii.ai or through whatsapp if you have any questions.