DSCI 551 - Project Guideline

Fall 2024

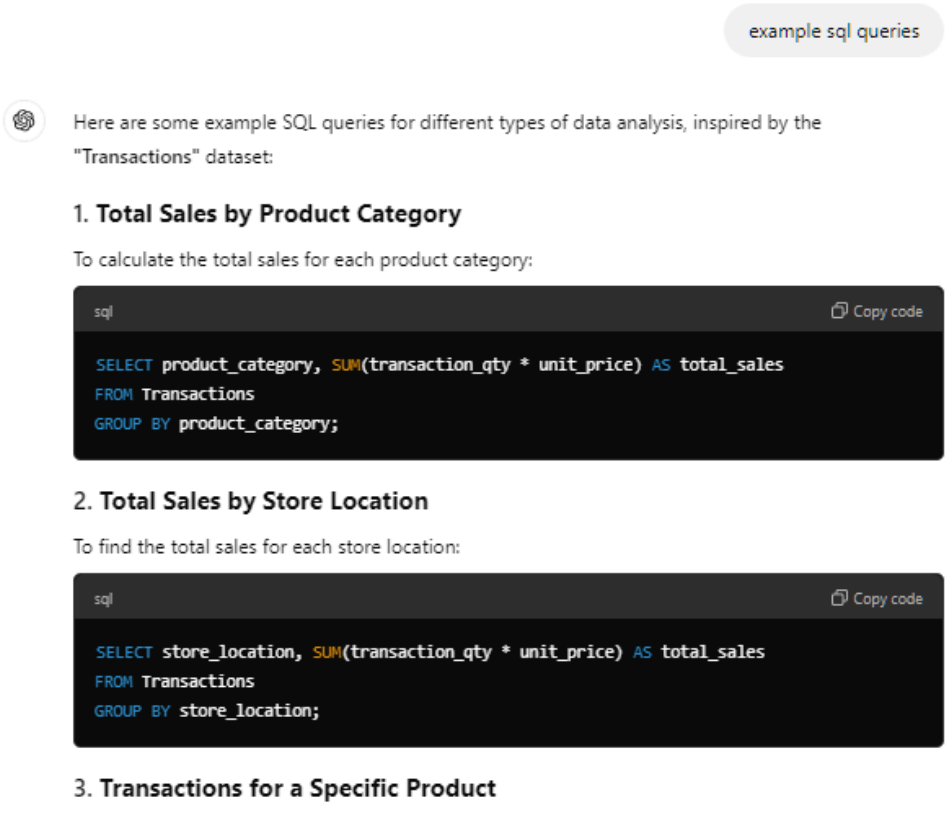
ChatDB – Learning to Query Database Systems Like ChatGPT

The goal of this project is to develop ChatDB, an interactive ChatGPT-like application that assists users in learning how to query data in database systems, including SQL and NoSQL databases. For example, it can suggest sample queries, including ones that include specific language constructs (e.g., group by in SQL), and understand queries in natural language (see examples below). But unlike ChatGPT, it can also execute the queries in the database systems and display the query results to the users.

Specifically, ChatDB should allow users to submit the following types of questions. Note the example uses the coffee shop data which can be downloaded here: https://www.kaggle.com/datasets/ahmedabbas757/coffee-sales?resource=download

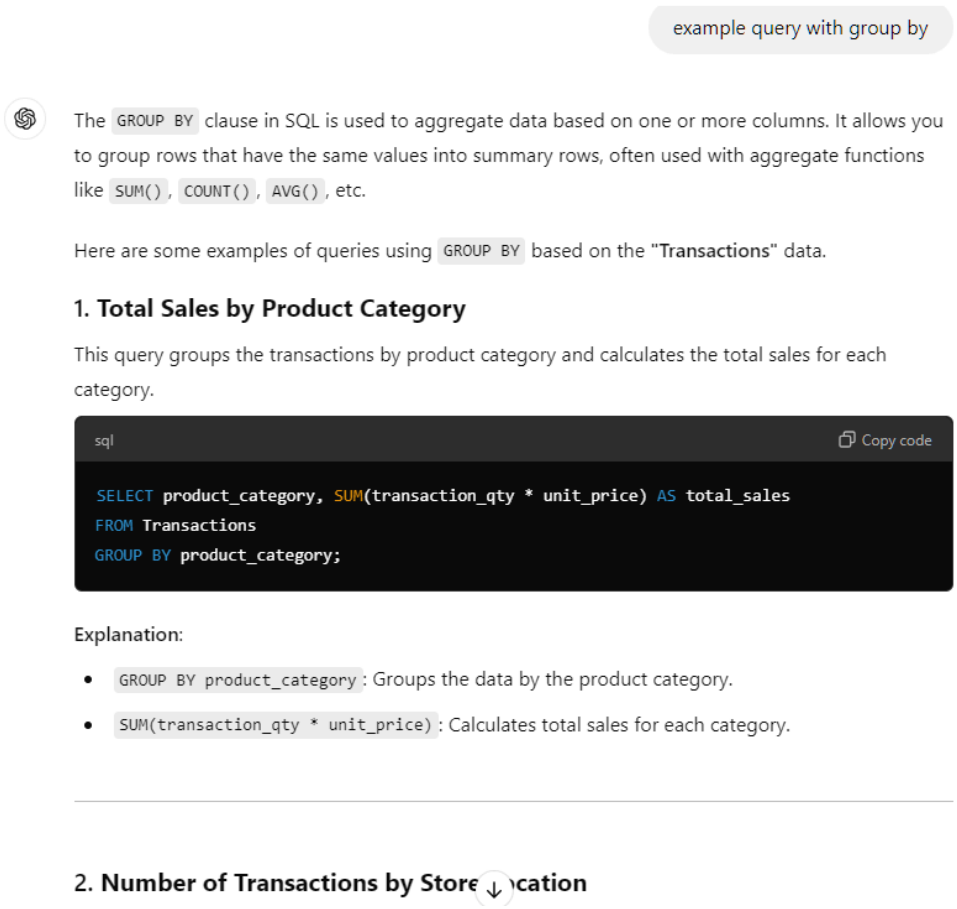
Note also that in ChatDB, the data set should be stored in the database system. For example, the data set may be stored as tables in MySQL or collections in MongoDB. In addition, ChatDB should provide a method that allows users to upload dataset into the database.

1. **Explore databases.** ChatDB should allow users to select which database to query and find out the table/collection and their attributes, along with some sample data.
2. **Obtain sample queries**. Once users select a database of interest, ChatDB should allow users to ask for example database queries. It should return a set of queries that uses a variety of



language constructs (e.g., ones with group by, having, order by, aggregation, where, join, etc.). Note users may ask sample queries multiple times, and ChatDB should return a different set of queries each time covering different constructs.

3. **Obtain sample queries with specific language constructs**, e.g., group by.



Note that for functions 2 and 3 above, ChatDB should have a strategy on how to systematically generate sample queries. For example, it might use a pattern or model to generate such queries.

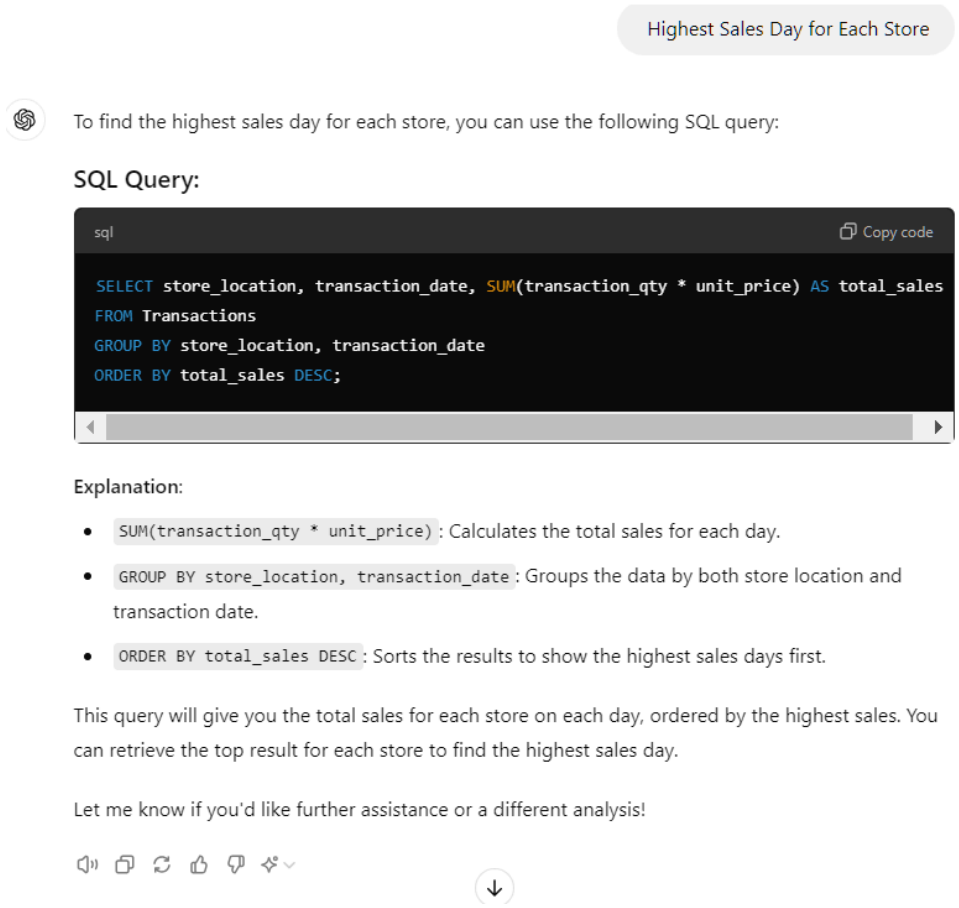
For example, a query pattern may be “total <A> by <B>” where <A> represents a quantitative attribute (e.g., sales amount) and <B> a categorical attribute (e.g., product category).

It is important to note that you should not hard code a set of queries, instead should use pattern/template, as just described, to generate the queries.

Note that for each query, it should return a natural language representation, e.g., “Total sales by product category”, and associated database query.

4. **Ask questions in natural language**. ChatDB should allow users to enter questions in natural language and detect questions that fit its query pattern/model. If a match is found, ChatDB

will return its corresponding database system, e.g., SQL or MongoDB functions. For example, if a user submits a query “find total sales amount broken down by product category”. ChatDB should recognize that it closely matches the query pattern “total <A> by <B>” described above.



Additional Requirements:

- ChatDB should be tested on at least three datasets for each type of database system.
- For queries suggested by ChatDB, it should allow users to execute the queries in the database and obtain the query results.
- You can form a group of up to 3 people for your project.
    - One-person group: Ok for ChatDB to support only one SQL/NoSQL database (e.g., MySQL, MongoDB). Note you should not use SQLite.
    - Two-person group: ChatDB should support one SQL and one NoSQL database.
    - Three-person group: ChatDB should support one SQL and one NoSQL database. You should also develop a Web browser based UI for users to interact with ChatDB.
- You should not use libraries or tools, including ChatGPT, that makes the implementation of ChatDB trivial.

Project deliverables:

- **Proposal** (due 9/23, Monday, 10 points): Detail your project plan, including database system, dataset, language, and how users interact with the system, how you plan to implement ChatDB. Also list group members and their roles. Note that your project is a collaborative effort and all members should contribute to the project. We will grade your project as a team effort and every member receives the same grade.
- **Midterm progress report** (due 10/18, Friday, 5 points): tell us your progress so far and the challenges you might have encountered.
- **Demo** (in-class, 11/25 for Monday section or 11/26 for Tuesday section, 5 points): Give a live demo of your project. All project members should be present during the demo.
- **Final report** (due on 12/13, Friday, 10 points): the final report should be comprehensive, detailing your design and implementation.
- **Implementation** (due on your demo time, 70 points): note your project should be fully implemented before the demo. You should include in your final report a link to Google drive where you will upload your project codebase and documentations. Make sure you give access to your project folder.