Specification

Our problem

Non-programmers often face barriers when trying to interact with databases due to the technical nature of programming languages. Users lose significant time of their working day due to menial tasks that could easily be automated. The aim here is to bridge this gap by providing a tool that translates natural language queries into executable code and provides a visual representation of the result, to help non-technical people make decisions from their data.

Functionality

The main role of the project is to answer a user's natural language query precisely and accurately, providing clear visualisations when appropriate. The user should be able to write in a query in natural English and (usually) receive a graph or chart generated with data from their database which helps to answer their question. The graph and code will also be saved such that the user can recreate the query at a later date, allowing them to easily gain insight on more up to date data.

We could extend this project to add in suggestions to the user to make their query more clear. We could also extend this to let our application have a conversation with the user, to better understand the user's request, allowing users to get a better output from the application.

Major components

Actioner (GPT 3.5 turbo)

- Feed in necessary data such as the database schema as well as a description of each table.
- Feed in an instruction to get the desired output e.g. "You are an expert in SQL. Given the database schema below and the user's request, come up with an actionable command for data retrieval from a database with the given database schema."
- Choose the desirable chart/graph type from a given list
- Refactor the instruction to make it into one or multiple subqueries

Inputs: User's question (in natural language)
Outputs: Type of graph to be generated, Database fields to be queried, Specific prompts to generate graph

SQL Generator (GPT 4)

- Feed in necessary data such as the database schema and any context generated in the Actioner pipeline step.

- Feed in instruction to get the desired query e.g. “You are an expert in SQL. Given the database schema below and the user’s action command, generate SQL code to query the database.”
- Specify the expected format in which the SQL code should output data.
- Debugging and correcting Hallucination by using SQL query validation libraries.
- Then use the SQL query to retrieve data

Inputs: Database fields to be queried, Specific prompt to generate graph
Outputs: SQL query used to retrieve data from database relevant to prompt

Description/Explanation and Visualisation

- Takes in SQL code from the previous component and queries the database
- Generates a description of the graph which will be given to the user
- Takes in the SQL output and visualises the data

Inputs: Type of graph (actioner), SQL query (SQL generator)
Outputs: Graph answering initial question, SQL query, Code used to generate the graph

User Interface

How the user will interact with our application. This will allow the user to enter a prompt and see the output graph.

- Prompt to allow user to input data
- Query suggestions and user help
- Provide option to save query, result (graph or text result) and result explanation
- Interface to view previous queries and corresponding results

Acceptance criteria

Functionality

- The application should determine whether a user query is serviceable (the necessary data exists) and raise an error accordingly
- The application must return an output to the user - an answer or an error message
- The application should decide the most relevant visualisation of the data, where necessary - pie chart or bar chart
- The application could have more visualisation options: e.g. line chart, histogram etc.
- The SQL code created by the project must be valid and run on the user’s databases
- The code that creates the visualisation should be as simple as possible and possible for the user to understand
- The application should comment on and explain the code to help the user understand how it works
- An answer, if given, must be entirely accurate
- The visualisations should be clearly labelled and understandable to the user
- The application should use the data provided to create a visualisation that most closely answers the users request
- The application must allow the user to upload SQLite3 databases to query
- The application could allow the user to upload other types of databases

User Interface

- The user interface must be simple to understand and use for a first time, non technical user
- The user interface must have a text box to let the user enter a query and a button to create the result
- The user interface should have a button to allow the user to keep or delete the output from the system
- The user interface must have a way to let the user upload their databases
- The user interface should allow the user to see previous queries and their respective answers that they previously chose to keep
- The user interface should allow the user to see the code and SQL query used to create the visualisation
- The user interface could allow the user to update a previous visualisation with more up to date data from their database

Management strategies

Team meetings (Twice a week)

- Task updates and open discussion

Asana for team management (ticketing)

- Individual tasks
- Deadlines
- Dependencies

Github for code management

- Code status
- Management via branching
- Regression testing (Base functionality)

Technical contributions to be made by each member

Actioner Ram Vinjamuri Isaac Lam Leo Takashige
SQL Management Mmesoma Okoro Samuel Jie
Visualistions & UI Izzi Millar Leo Takashige

Project Plan

Actioner + Pre-processing

- Create a function to generate the schema - 10 hours
- Create a qualitative description of the database - 11 hours
- Generate a list of database specific words - 8 hours
    - Requires CRM database test examples
- Refactor the instruction
    - Create prompt for generating subqueries - 5 hours
    - Design the query with set examples - 5 hours
    - Generate examples for GPT - 7 hours
- Determine if a subset of database (schema) can be used - 10 hours

SQL Management

- Find an appropriate CRM database for testing - 2 hours
    - Required for most other components
- Use context and schema to ask AI for suitable SQL query/queries - 13 hours
    - Schema dependent on the subset given by Actioner
    - Use Query with Param to prevent SQL injection
- Create debugger for SQL query to validate SQL and identify errors - 11 hours
    - Required for validating SQL query
- Error Handling Types - 5 hours
- Create procedure for retrying prompt or outputting error message - 7 hours
- Use SQL query to retrieve data from database - 3 hours

Visualisations & UI

- Simple UI for user to interact with and enter questions - 5 hours
- Dashboard to switch between previous requests or add new ones, etc. - 10 hours
- Code view and explanation for user - 15 hours
    - Requires code for graphs to be written
- SQL view and explanation for user - 8 hours
    - Requires SQL to be generated
- OOP structure for graphs (pie chart and bar graph or data point initially) - 15 hours
    - Needed for actioner and SQL management to pass data in a useable format
- Implementation for each type of graph - 10 hours
    - Requires OOP structure