Database Visualization using Spreadsheet Interface

Team Members

- Chandrakant Pradhan (22B1023)
- Nimish Milind Manware (22B0944)
- Jayesh Vinod Jadhav (22B1056)
- Prasanna Ganesh Nage (22B0953)

Goals of Project

This project aims to provide an intuitive user interface for visualizing and interacting with databases. It eliminates the complexity of manually setting up PostgreSQL by using **PGLite**, an in-browser database. Finally, it includes features for visualizing data through various statistical plots.

User Perspective

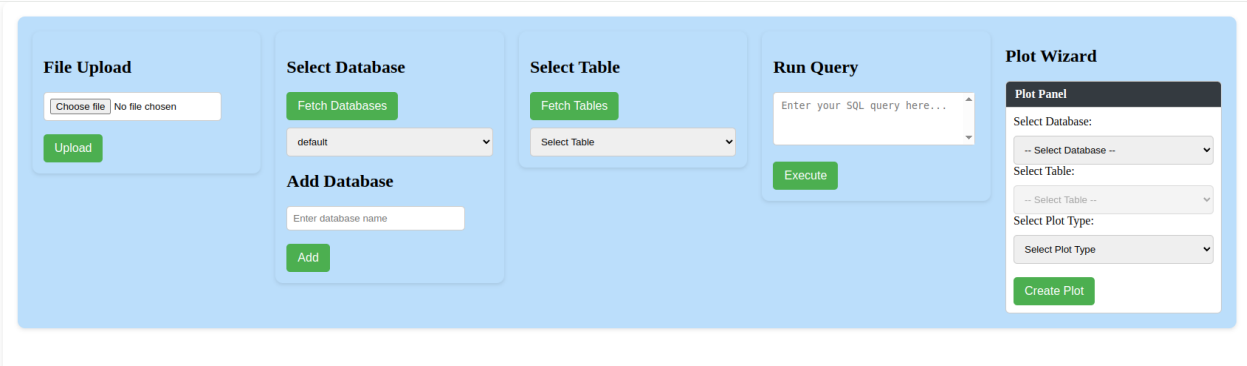

Figure 1: Features available to the user.

Here are the features implemented from a user perspective:

1. **File Upload**: Allows users to load DDL and DML files into the database.
2. **Add Database**: Enables users to create a new database.
3. **Select Database**: Allows users to choose an existing database.

4. **Select Table**: Enables users to select a table and view it in a spreadsheet format. An example is shown below:

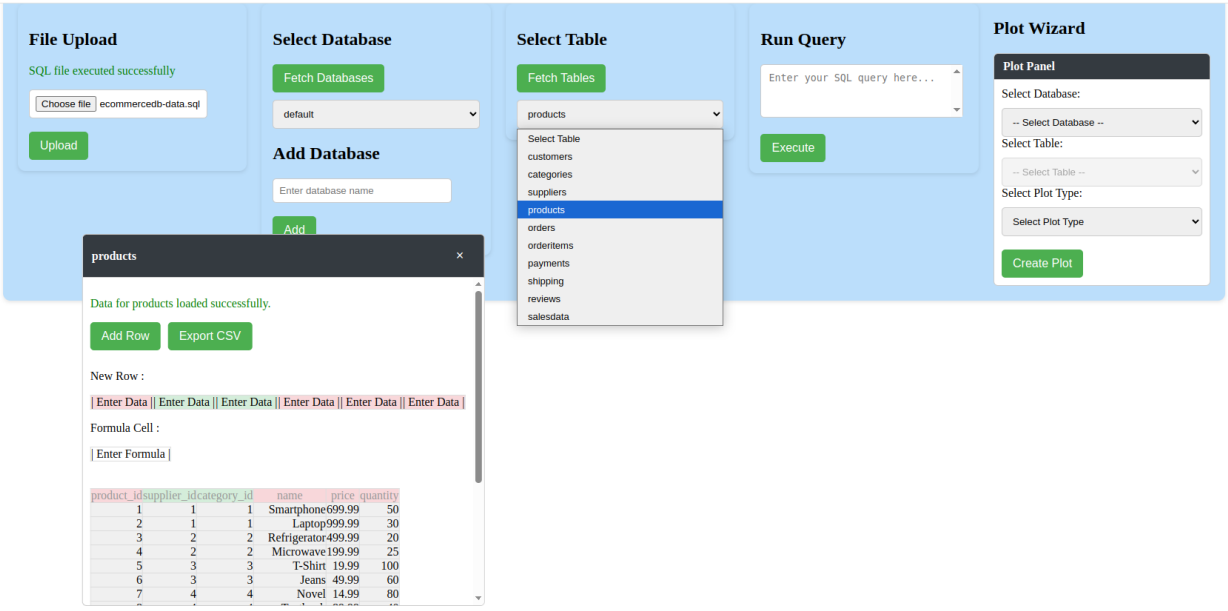

Figure 2: Spreadsheet Interface

5. **Add Row**: Enables users to add new row in the database.
6. **Export CSV**: Enables users to download the table in .csv format.
7. **Add Row**: Enables users to add new row in the database.
8. **Formula Cell** : Supports spreadsheet like formulas , when the content of the table changes , it is reflected in the formula cell. Here are the list of formulas supported :
   (a) **=SUM(range)**
   (b) **=MAX(range)**
   (c) **=MIN(range)**
   (d) **=AVG(range)**
   (e) **=COUNT(range)**
   (f) **=MEDIAN(range)**
   (g) **=MODE(range)**
   (h) **=PRODUCT(range)**
   (i) **=STDEV(range)**

9. **Run Query**: Enables users to run queries on the selected table and save result in temp table or alter table using sql queries.

10. **Plot Wizard**: Enables user to plot database data.Following are the supported plot types :

    (a) **Line Chart**:

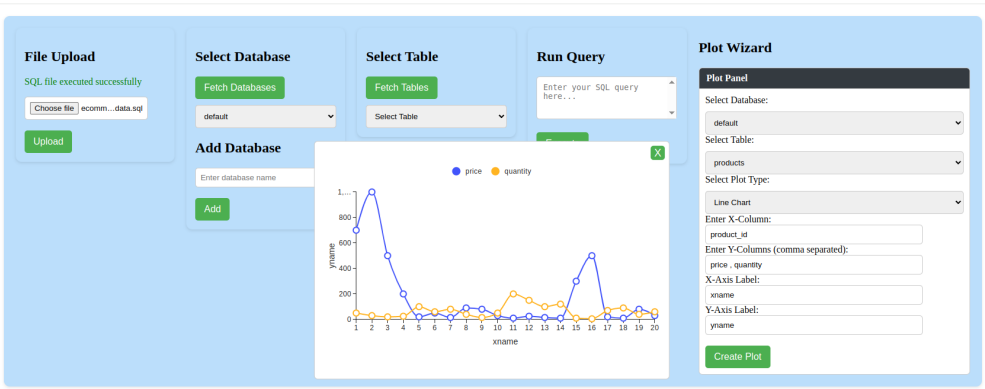

Figure 3: Line Chart Example

    (b) **Bar Chart**:

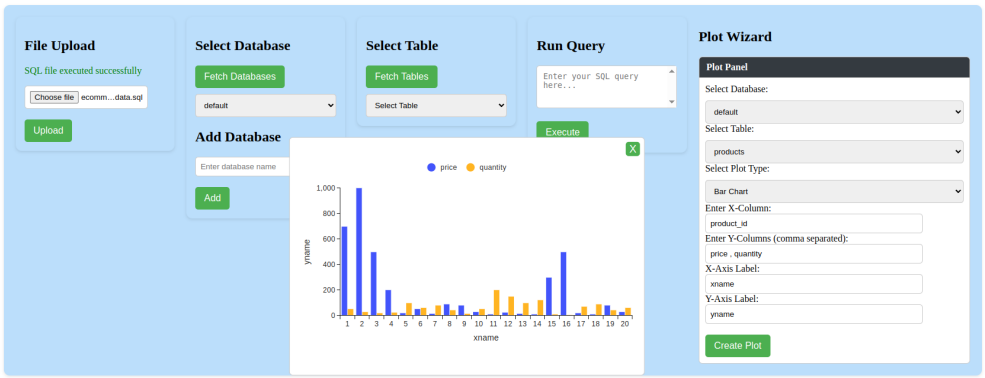

Figure 4: Bar Chart Example

    (c) **Pie Chart**:

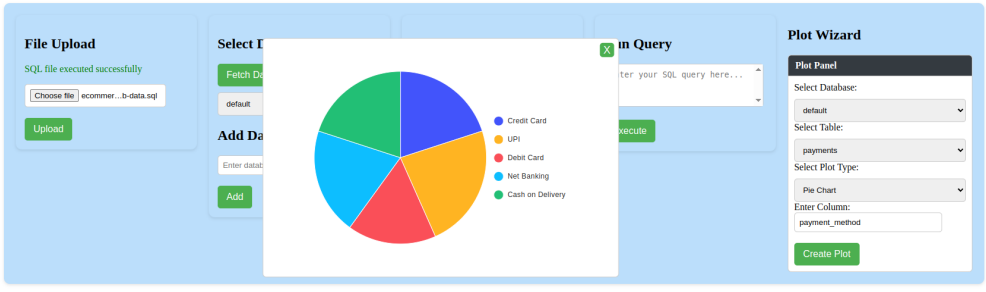

Figure 5: Pie Chart Example

(d) **Scatter Plot**:

Figure 6: Scatter Plot Example

(e) **Pivot table**:

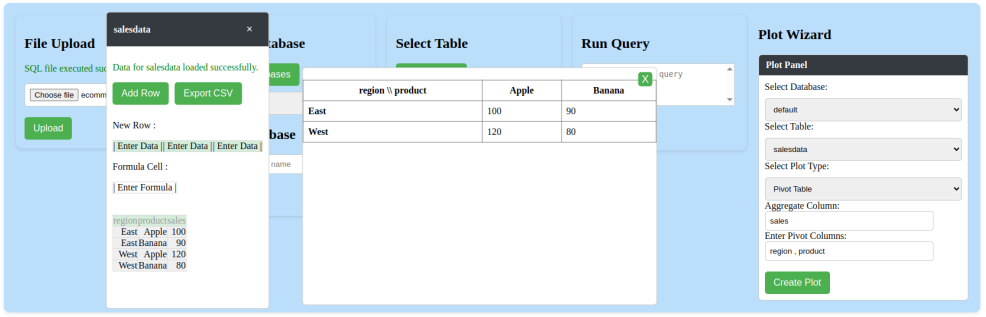

Figure 7: Pivot Table Example

Architecture Overview

Here is the overview of the architecture of the application:

Homepage

- Continuously fetches the list of available databases from the backend.
- Continuously fetches the list of tables corresponding to the currently selected database.
- Maintains the state of the currently selected database and table, which is propagated to other components that require this context.

File Upload Component

- Operates on the currently selected database.
- Parse DDL and DML commands and execute it in the backend for the corresponding database.

Database List Component

- Receives a list of available databases from the backend.
- Renders this list as a dropdown menu.

Database Add Component

- Allows users to create new databases by submitting a name.
- Upon submission, the name is added to the PGLite instance in the backend, which manages all local databases.

Table List Component

- Receives a list of available tables from the backend for the currently selected database.
- Displays the tables in a dropdown menu.

Run Query Component

- Provides a text input interface for users to write SQL queries.
- Executes the queries in the context of the selected database in the backend and returns results or error messages.

Add Row Component

- Enables users to input new row data in a temporary (dummy) row interface.
- When the "Add Row" button is clicked, the data is validated and inserted into the appropriate table in the backend.

Formula Cell Component

- Allows users to input formulas directly into spreadsheet cells.
- Extracts the parameters (e.g., cell ranges and operations) from the formula.
- Computes the value based on the formula and stores both the computed value and the formula itself.
- Automatically re-computes values when dependent cells are updated using the stored formula.

Database Data to Sheet Component

- Use external library for rendering spreadsheet.
- Populates the sheet with table data from the database.
- Includes computed primary key values, for the case when the value is changed.

Plot Wizard Component

- Use wizard to collect parameters such as axes, labels, filters, and chart types.
- Retrieves necessary data from the backend database based on user inputs.
- Uses a plotting library to render visualizations (e.g., bar charts, scatter plots, line graphs) directly from the queried data.

Frontend and Backend Functionalities

Backend Functionalities:

- **load-file** : handles file load
- **run-query** : handles queries
- **get-tables** : gets tables corresponding to a database
- **get-table-data** : gets all necessary data corresponding to a table
- **handle-cell-change** : handles stuff when a cell is changed in the sheet
- **add-row** : handles row addition
- **add-database** : handles database addition
- **get-databases** : gets all databases in the application
- **get-plot-data** : given the plot info , extracts data

Frontend Functionalities:

- **AddDatabase** : Component to add a new database
- **DatabasesView** : Component to provide a list of databases
- **FileUpload** : Component to handle file uploads
- **PlotComponent** : Component for displaying a single plot
- **PlotPanel** : Component to show the plot wizard
- **RunQuery** : Component that provides a space for writing queries
- **Sheet** : Component to display the database as a spreadsheet
- **TableDropDown** : Component to show the list of tables
- **Homepage** : Component aggregating all other components together

Implementation Details

Majority of the work was required in designing the flow of the program and the core logic like formula implementation etc and finding the compatible libraries for the project. For the syntax part online documentations and AI tools are used.

Conclusion and Future Work

The application is very user friendly for visualising databases but lacks some additional features like indexing , query optimization , support for more formulas and plot types.