LLM Agents For SQL and Pandas Query Generation (QueryGenAI)

Avinash Reddy Vasipalli, Akshay Kumar Sureddy, Abhishek Sureddy

Muhammad Yusuf Hassan, Dhanus Kanth Anand

December 10, 2024

GitHub Link - `https://github.com/Avi-2362/LLM-agents-for-SQL-Pandas-query-generation`

1 Introduction

The increasing complexity of data analysis tasks and the growing volume of data necessitate the development of tools that empower both technical and non-technical users to efficiently derive insights. The *LLM Agents for SQL/Pandas Query Generation* project aims to address this requirement by leveraging the capabilities of Large Language Models (LLMs) to simplify data querying and processing.

This project introduces a web-based platform designed to bridge the gap between user intent and technical execution. By enabling users to input queries in natural language, the system generates optimized SQL or Pandas queries, processes data from CSV files. The primary objectives of this project include:

- **Empowering Non-technical Users:** Facilitate data analysis for individuals without coding expertise, enabling them to perform complex queries tasks in text.
- **Time-Saving for Businesses and Researchers:** Reduce the time and effort required for data querying.

The technological stack for this project incorporates advanced tools and frameworks, including Angular for the user interface, Flask for backend integration. Additional features include secure user spaces, and support for exporting results in CSV, ensuring practicality and user-friendliness.

By providing an intuitive and efficient interface for data interaction, the *LLM Agents for SQL/Pandas Query Generation* project presents a transformative approach to modern data analysis, fostering both accessibility and productivity.

2 Features

2.1 Natural Language Interface

Users can input queries and commands using natural language, eliminating the need for technical knowledge. This makes the tool accessible to non-technical users for performing complex data Query tasks easily.

2.2 Query Generation

The system automatically generates SQL or Pandas queries based on the user's natural language input. This feature ensures accurate query creation tailored to the uploaded data and the user's intent.

2.3 Data Upload and Processing

Users can upload CSV files for analysis. The system validates the uploaded file format, processes the data, and ensures it is ready for further querying.

2.4 File Management with Dashboard

The platform includes a dedicated dashboard for file management(User Space):

- Users can upload, view, and delete files.
- A dedicated page allows easy addition and organization of new files.
- This feature ensures all uploaded files are readily accessible and manageable.

2.5 Query Execution

The system validates and safely executes the generated queries on the uploaded data. A built-in security check ensures no vulnerabilities are exploited during query execution.

2.6 Results Display

Processed data is displayed in a clean, user-friendly format, making it easy to review and understand the results of the queries.

2.7 Export Capabilities

Users can export processed data in CSV. This allows for seamless integration of analyzed data into other tools and workflows.

3 Functional Requirements (Use Cases)

1. **Secure Login and Access Control:**
   As a user, I want to securely log in and sign up using hashed passwords so that my account and data are protected from unauthorized access.
2. **File Upload and Validation:**
   As a user, I want to upload CSV files and have the system validate them, ensuring only properly formatted files are accepted for further processing.
3. **Natural Language Query Input:**
   As a user, I want to input my queries in natural language so that I can easily request data analysis without needing technical knowledge of SQL or Pandas.
4. **Query Generation and Execution:**
   As a user, I want the system to generate and safely execute SQL or Pandas queries based on my natural language input, ensuring that the queries are optimized and secure.
5. **File Management with Dashboard:**
   As a user, I want a dedicated dashboard to manage my uploaded files, where I can view, upload and delete files efficiently.
6. **Results Display:**
   As a user, I want to view the processed data in a user-friendly tabular format so that I can easily interpret the results.
7. **Export Processed Data:**
   As a user, I want the ability to export processed data in my CSV to integrate it into other tools or reports.
8. **Real-Time Error Feedback:**
   As a user, I want to receive real-time error messages and suggestions for invalid inputs or failed queries so that I can quickly correct them and continue working.
9. **User Preferences and History:**
   As a user, I want my preferences and query history to be securely saved so that I can access them easily on future sessions without reconfiguring.

4 Installation

Installation for our project is quite straightforward. First, we install the frontend dependencies, followed by the backend setup, then configure the environment variables.

4.1 Frontend Setup

If not already present, please install *Node.js* from the official website `https://nodejs.org/`. Then, run the following commands:

```
npm install -g @angular/cli
cd csv-query-app
npm install crypto-js
npm install prismjs
npm install --save-dev @types/prismjs
```

4.2 Backend Setup

In order to install backend dependecies, run the following from inside the `520_Project/` directory.

```
conda create -n querygenai python==3.10
conda activate querygenai
pip install -r requirements.txt
```

4.3 AWS Setup

This section provides a detailed explanation for creating and configuring the S3 bucket and DynamoDB tables required for the project.

Prerequisites

Ensure you have the following set up:

1. **AWS CLI** installed (Installation Guide).
2. An **AWS Account** with sufficient permissions to create S3 buckets and DynamoDB tables.
3. **IAM User/Role** with the following policies:
    - `AmazonS3FullAccess`
    - `AmazonDynamoDBFullAccess`

AWS Resources

1. Create S3 Bucket

The S3 bucket is used to store user-uploaded files.

S3 Bucket Details

- **Bucket Name**: `llm-query-generator`
- **Region**: `<your-region>` [e.g., `us-east-1`]

Steps to Create the S3 Bucket

1. Open the AWS Management Console and navigate to the **S3 Service**.
2. Click **Create Bucket**.
3. Enter `llm-query-generator` as the **Bucket Name**.
4. Select your preferred **Region**.
5. Enable or disable additional settings such as versioning or encryption based on your requirements.
6. Click **Create Bucket**.

Configure Bucket Policies To ensure proper access to the bucket:

1. Go to the **Permissions** tab of your S3 bucket.
2. Add the following bucket policy (modify `<AWS-Account-ID>` as needed):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutGet",
            "Effect": "Allow",
            "Principal": "*",
            "Action": ["s3:GetObject", "s3:PutObject"],
            "Resource": "arn:aws:s3:::llm-query-generator/*"
        }
    ]
}
```

2. Create DynamoDB Tables

The DynamoDB tables store metadata related to users and their files.

Table Details

Table Name	Purpose	Primary Key
`llm-user-files`	Tracks files uploaded by users	`user_id` (String)
`llm-user-table`	Stores user details	`user_id` (String)

Steps to Create Each Table

1. Open the AWS Management Console and navigate to the **DynamoDB Service**.
2. Click **Create Table**.
3. Enter the **Table Name** and **Primary Key** based on the table details above.
4. Leave **Sort Key** empty for `llm-user-table`.
5. Configure the **Read/Write capacity** (choose **On-Demand** or **Provisioned** as needed).
6. Click **Create Table**.

CLI Commands (Optional) To create the tables via CLI:

```
# Create llm-user-files table
aws dynamodb create-table \
    --table-name llm-user-files \
    --attribute-definitions AttributeName=user_id,AttributeType=S AttributeName=file_id,AttributeType=S
    --key-schema AttributeName=user_id,KeyType=HASH AttributeName=file_id,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST

# Create llm-user-table
aws dynamodb create-table \
    --table-name llm-user-table \
    --attribute-definitions AttributeName=user_id,AttributeType=S \
    --key-schema AttributeName=user_id,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST
```

3. Permissions

Ensure the application has IAM permissions to access these resources. Below is an example IAM policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::llm-query-generator",
                "arn:aws:s3:::llm-query-generator/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query"
            ],
            "Resource": [
                "arn:aws:dynamodb:<your-region>:<AWS-Account-ID>:table/llm-user-files",
                "arn:aws:dynamodb:<your-region>:<AWS-Account-ID>:table/llm-user-table"
            ]
        }
    ]
}
```

4.4 Environment Variables

Since we are using LLMs via the OpenAI API, the key needs to be set in the environment variables. For this, create a `.env` file inside of `backend/app/Api` with the following configuration:

```
OPENAI_API_KEY="YOUR_KEY_HERE"
```

5 Usage

The website can be accessed from `http://localhost:4200/` once the following commands are used to get the frontend and backend running.

5.1 Frontend

To run the frontend, open a terminal and run the following:

```
cd csv-query-app
ng serve --configuration production
```

5.2 Backend

For the backend, open another terminal and run the following:

```
cd backend
python run.py
```

6 Unit Testing

We write unit tests for most of the functionalities in the project. All unit tests are present in the `backend/tests/` directory. In order to run all tests at once, one can run `pytest *.py` from inside the `backend/tests/` directory. Alternatively, one can also run tests separately for each set of functionalities using the following:

```
pytest test_api_models.py    # tests the User and UserFile models used in the backend
pytest test_aws_conn.py      # tests AWS connection
pytest test_pandas_agent.py  # tests API for the Pandas agent
pytest test_sql_agent.py     # tests API for the SQL agent
pytest test_validation.py    # tests query validation to filter dangerous queries
```

6.1 Testing API Models

These tests are written in the `test_api_models.py` file to check the functionality of the `User` and `UserFiles` models. First, we create objects of the type `User` or `UserFiles` using dummy attributes like `user_id`, `name`, etc. The object is then inserted into the corresponding table using the `put()` method. Then, the `get()` method is used to verify if all the data can be correctly retrieved from the table.

6.2 Testing the LLM Agents

These tests are written in the `test_pandas_agent.py` and the `test_sql_agent.py` files to check the functionality of both the LLM agents. We run checks for a range of queries on a sample CSV file and match outputs to the ground truth answers. Below are a few queries we test for using the Titanic dataset[1]:

```
Filter the rows where age is 21
How many people are there with age = 21?
What was the maximum fare?
Which cabin had the fare of 76.2917?
How many people were named John?
What was the name of passenger in cabin D56?
How many male people survived?
```

We also have testcases to check that the inputs to the agent function calls are valid, i.e. are a valid dataframe and query string.

[1] https://raw.githubusercontent.com/pandas-dev/pandas/main/doc/data/titanic.csv

6.3 Testing Query Validation

These tests are written in the `test_validation.py` file to check the functionality of the query validator function. The query validator functions make sure to filter out any keywords that if executed, could cause harm to the system like delete or alter data, execute other commands, etc.

7 Project Flow

7.1 SignUp Page

Figure 1: SignUp Page

Figure 1 shows Signup page has the fields as shown above. And also we do have real time error feedback to help user about an error if he encounted. The Figure 1 shows that the user exists and asks user to login.

Figure 2: SignUp Successful

Figure 2 shows successful sign of user. This is the page user is redirected when he successfully signed up. And the user is given the real time notification to login for the user.

7.2 Login Page

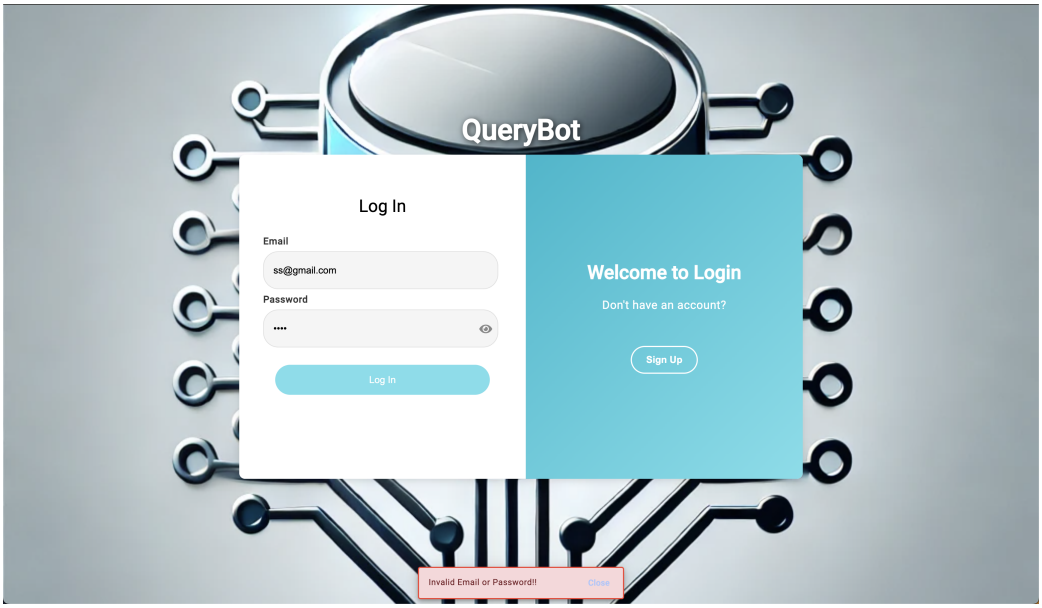

Figure 3: Login

Figure 3 shows the Login Page of the website and the the error message shows that the email and password is invalid giving the user the real time error message to retry the login details.

7.3 Dashboard Page

7.3.1 Dashboard

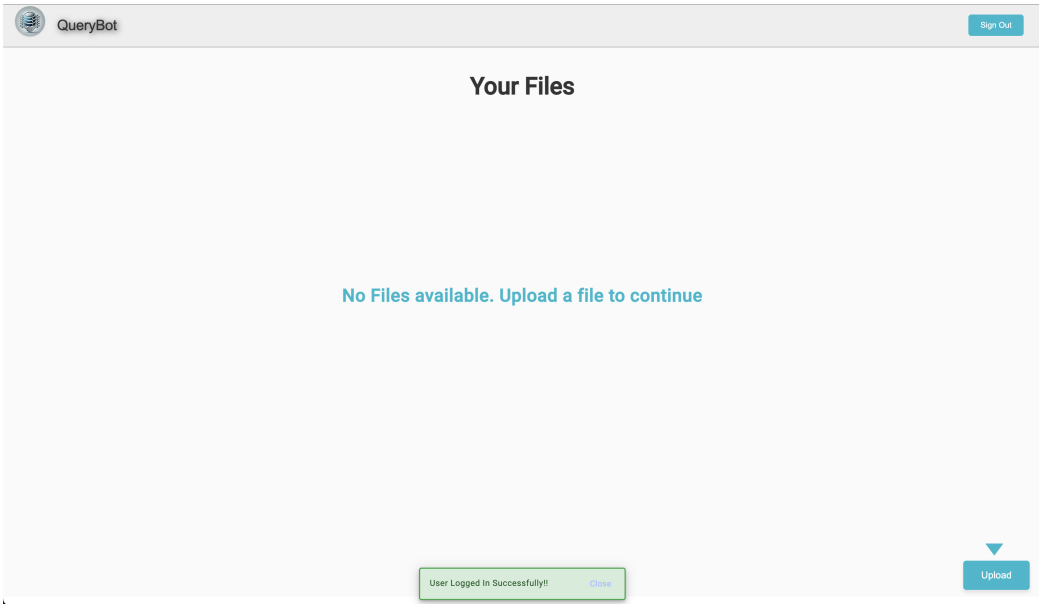

Figure 4: Dashboard

Figure 4 shows the Dashboard of the user when he is logged in successful and the image shows the user the next steps he needs to take and what is expected by the user to proceed to next step.

7.3.2 Upload File

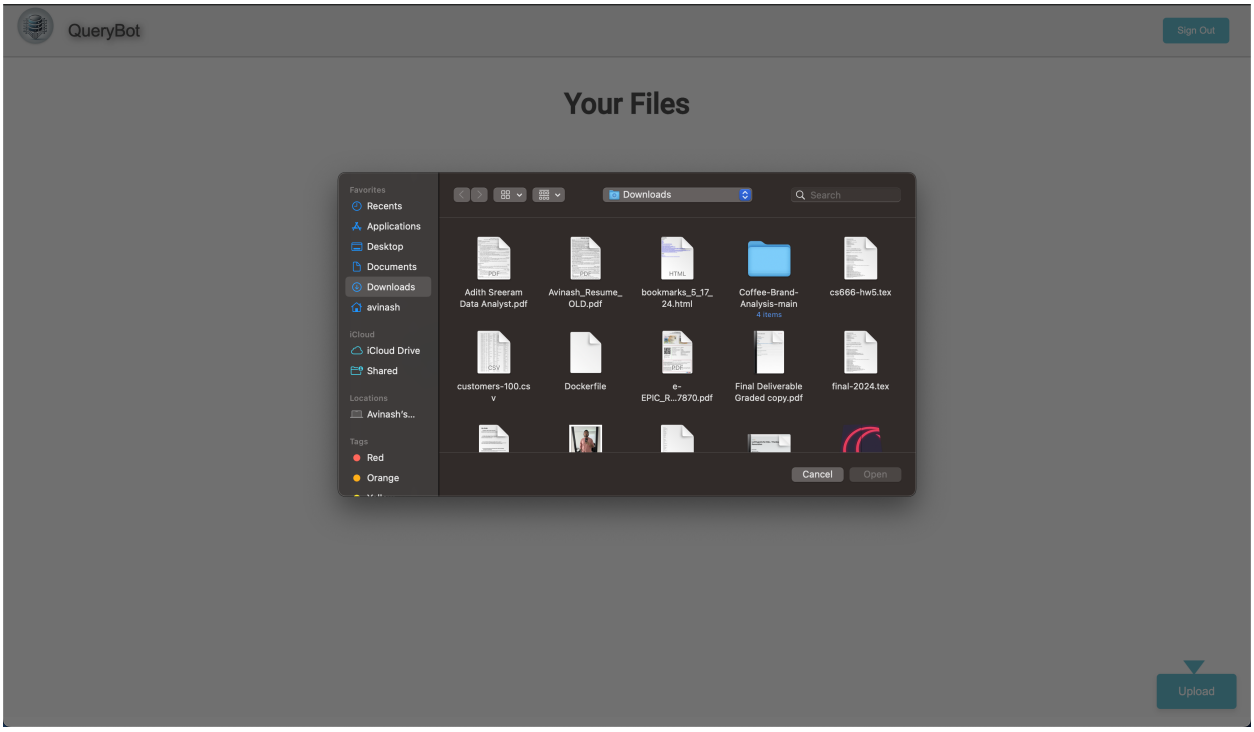

Figure 5: Upload Page

Figure 5 shows the upload function that allows user to upload a file from his local computer.

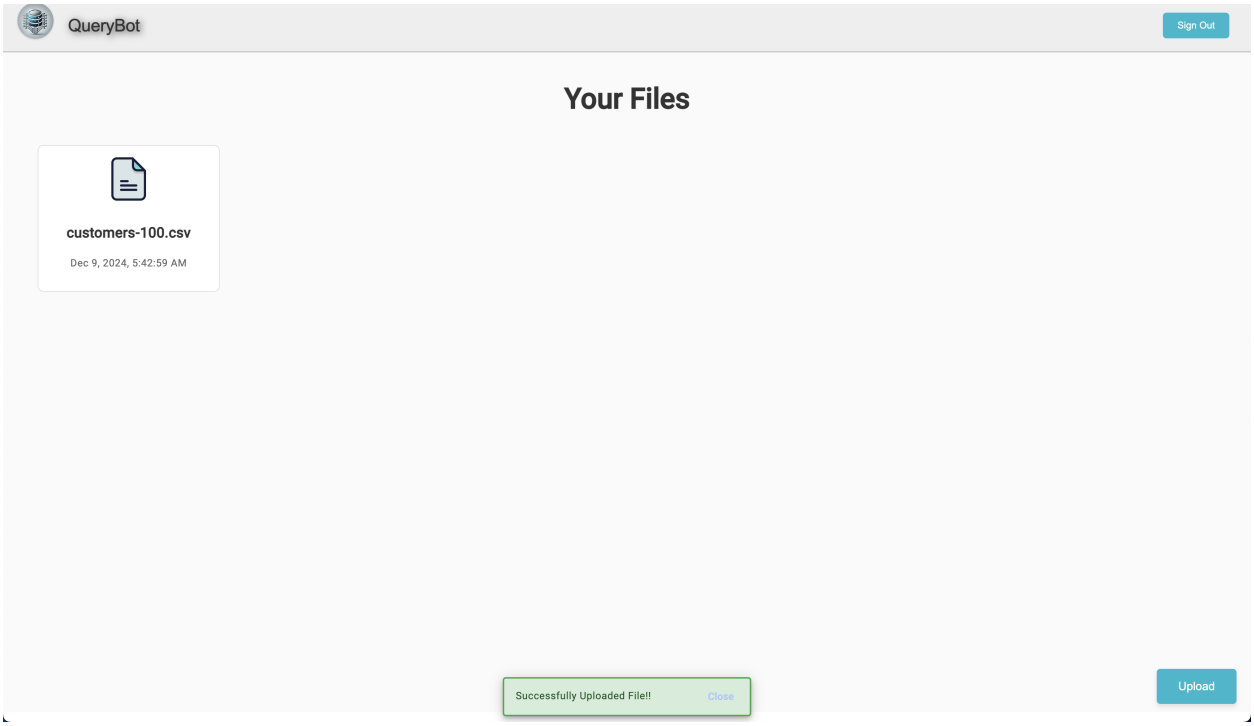

Figure 6: Upload Successful

In Figure 6 if the upload is successful then the user will be notified as above in real time. And the uploaded CSV will be sent to S3 by back-end and the file is shown in the dashboard with the time stamp.

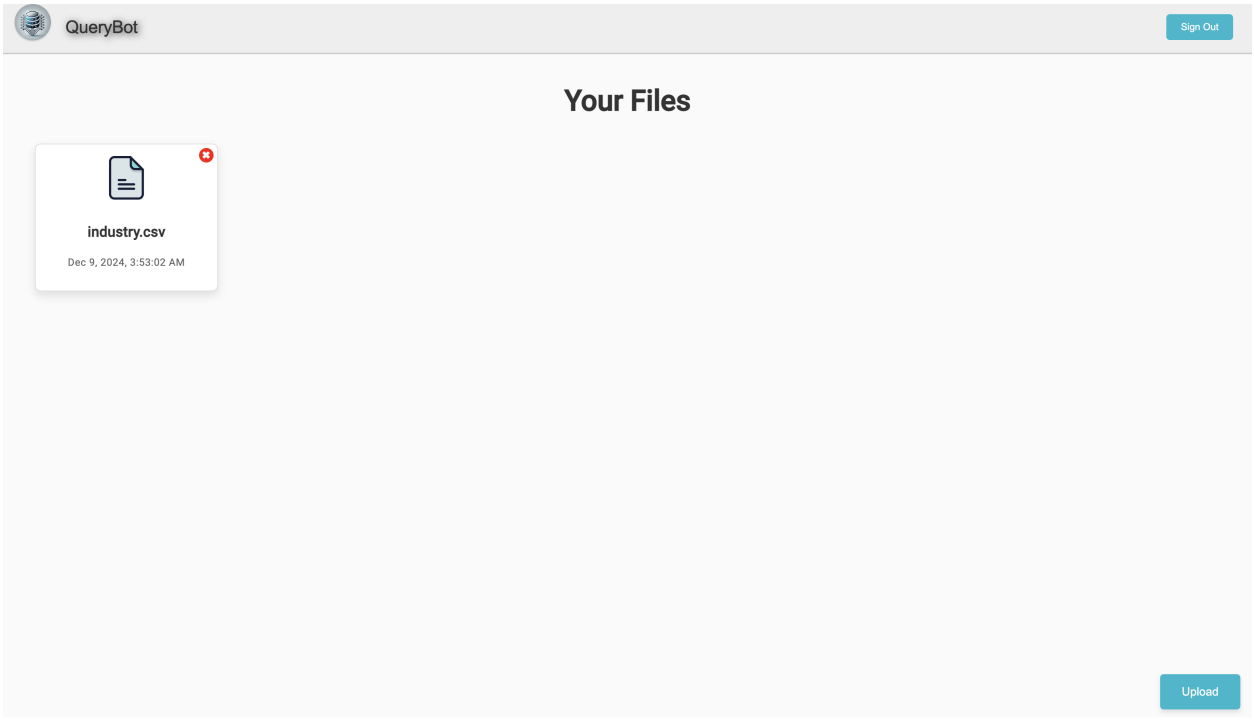

Figure 7: Upload Tile

Figure 7 shows the uploaded file as a tile and when the cursor is on the tile the hover effect shows the deletion mark which helps the user to delete the file.

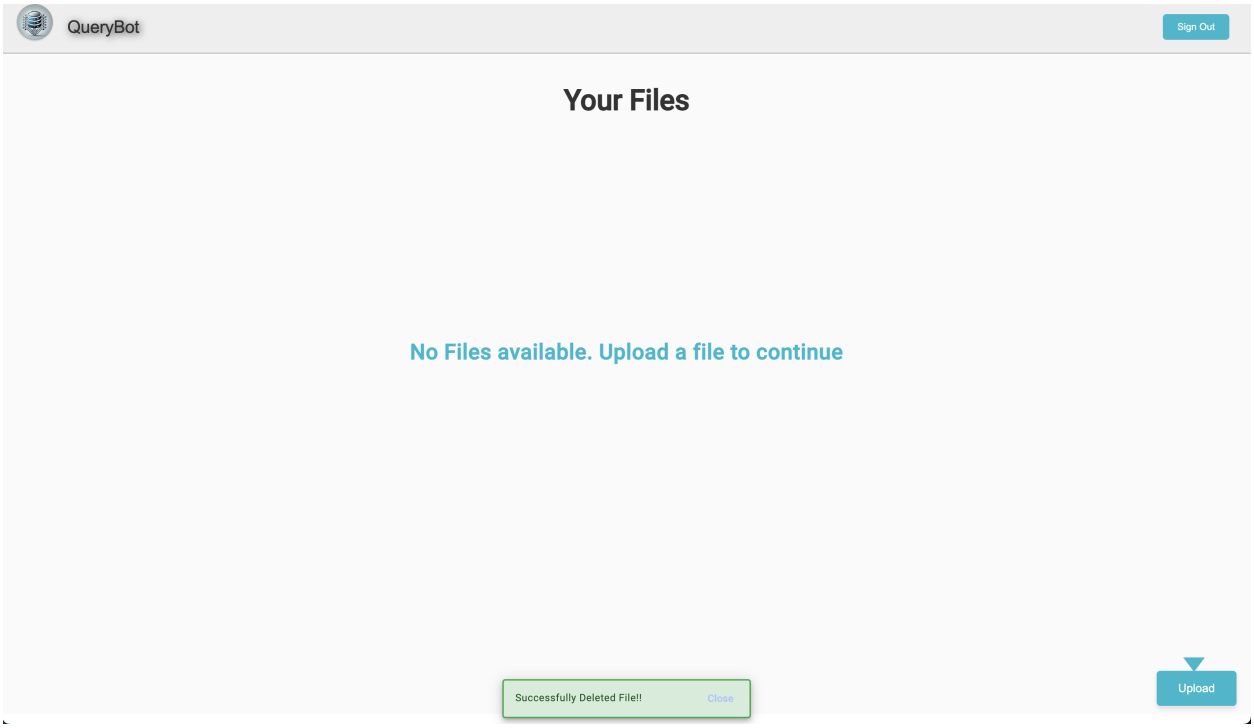

Figure 8: File Deletion

Figure 8 shows the deletion mark when the cursor is put on the tile and when we click the deletion mark the user file is deleted and the user is given real time notification below as shown in the Figure.

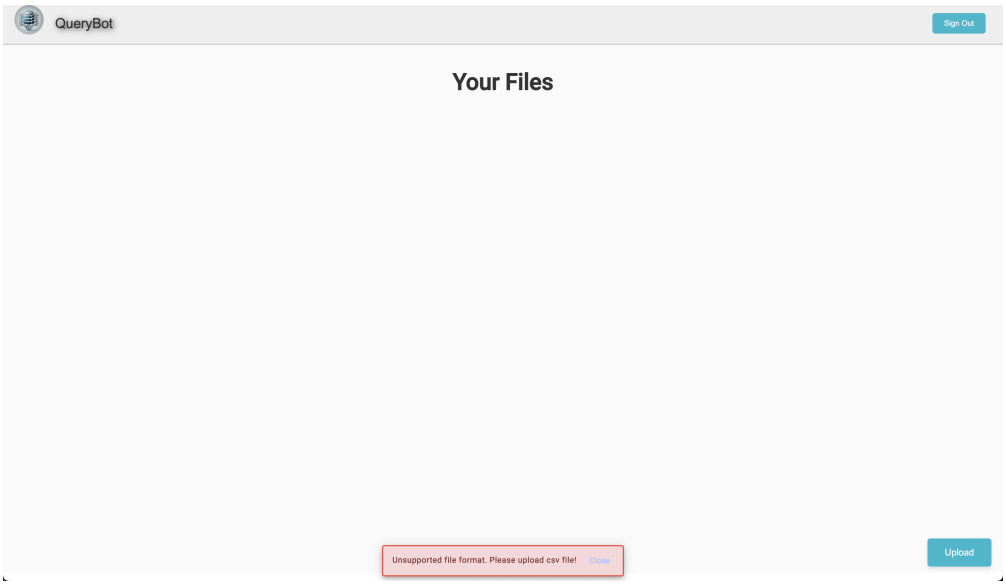

Figure 9: Unsuccessful Upload

In Figure 9if the user uploads the file other than CSV then the above image is shown where the user is notified in real time above the format of the file being unsupported and lets the use know to upload CSV only.

7.4 File Upload Page

Figure 10 shows File Upload landing Page which has 2 sections left and right section. Left section has the Parsed CSV file and displays the first 50 rows and columns of the file. And below that is a box to type in text to send which can be imputed by typing or by using voice option as well which will be sent to the LLm agents when clicked.
Right section has the query options to select from SQL or Pandas and below that is the result the LLM agent generates for the option we selected. We also have features like copy the text returned by the LLM.

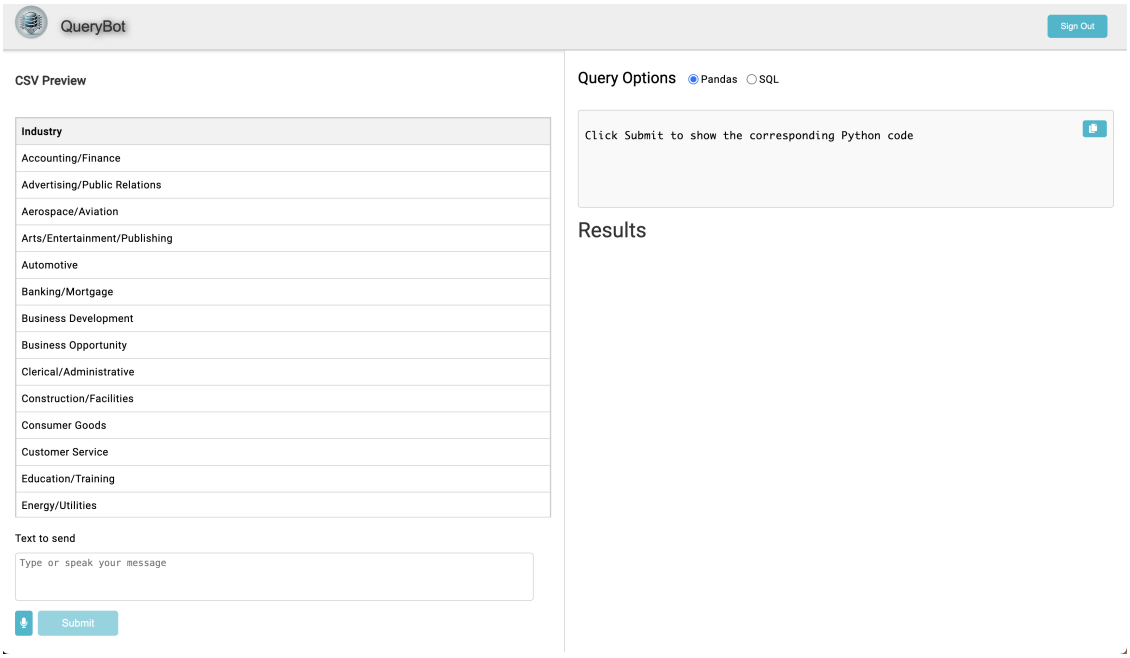

Figure 10: File Upload Page

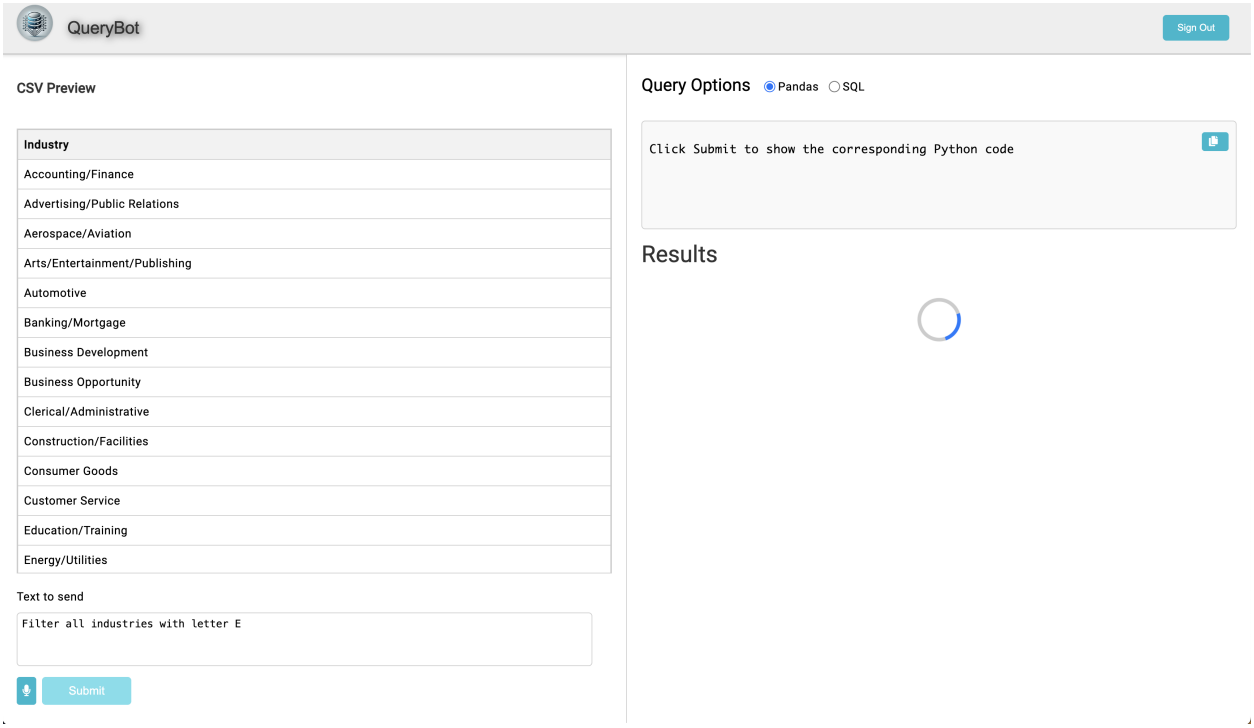

Figure 11: LLM Calling

Figure 11 shows the text the user inputted and when submitted the right section shows a reloading animation giving the user the understand that the LLM agent is working on the request.

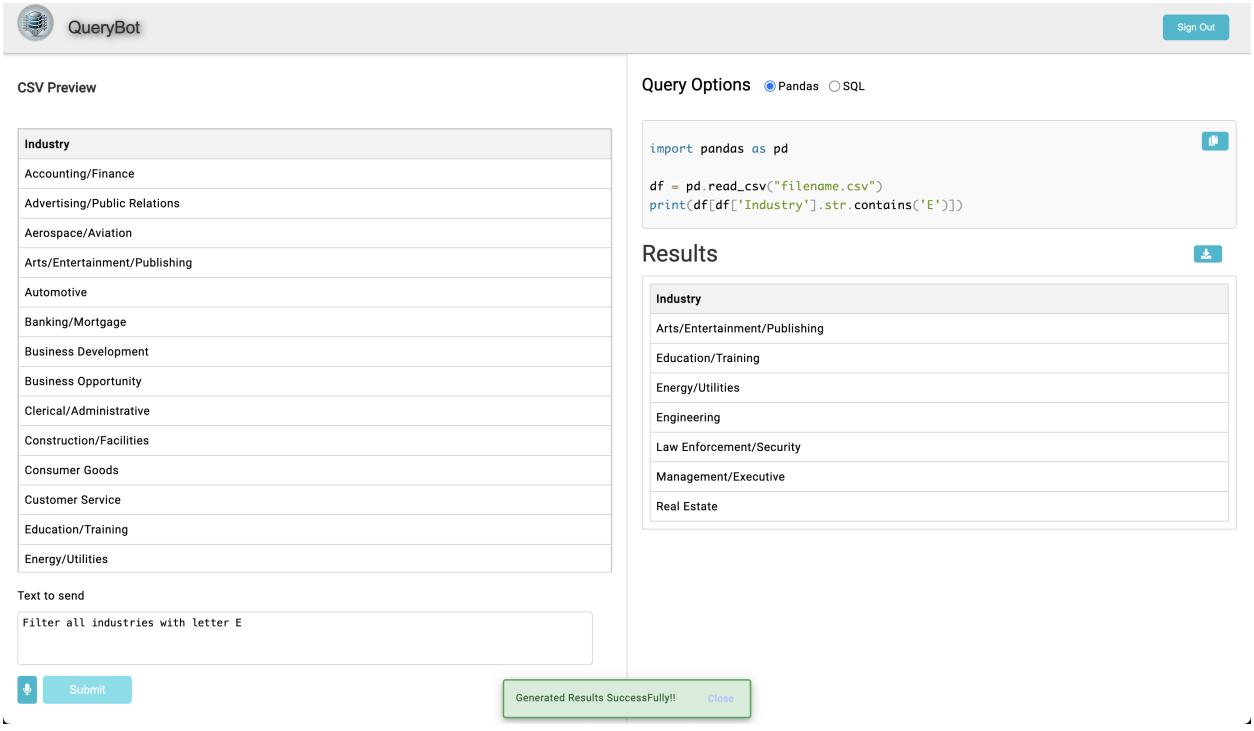

Figure 12: Successful Result

Figure 12 shows the successful result of the user request and real time notification that appears when the result is generated successful. Also the LLM agent generates the code for pandas and generates the code for the text the user requested.

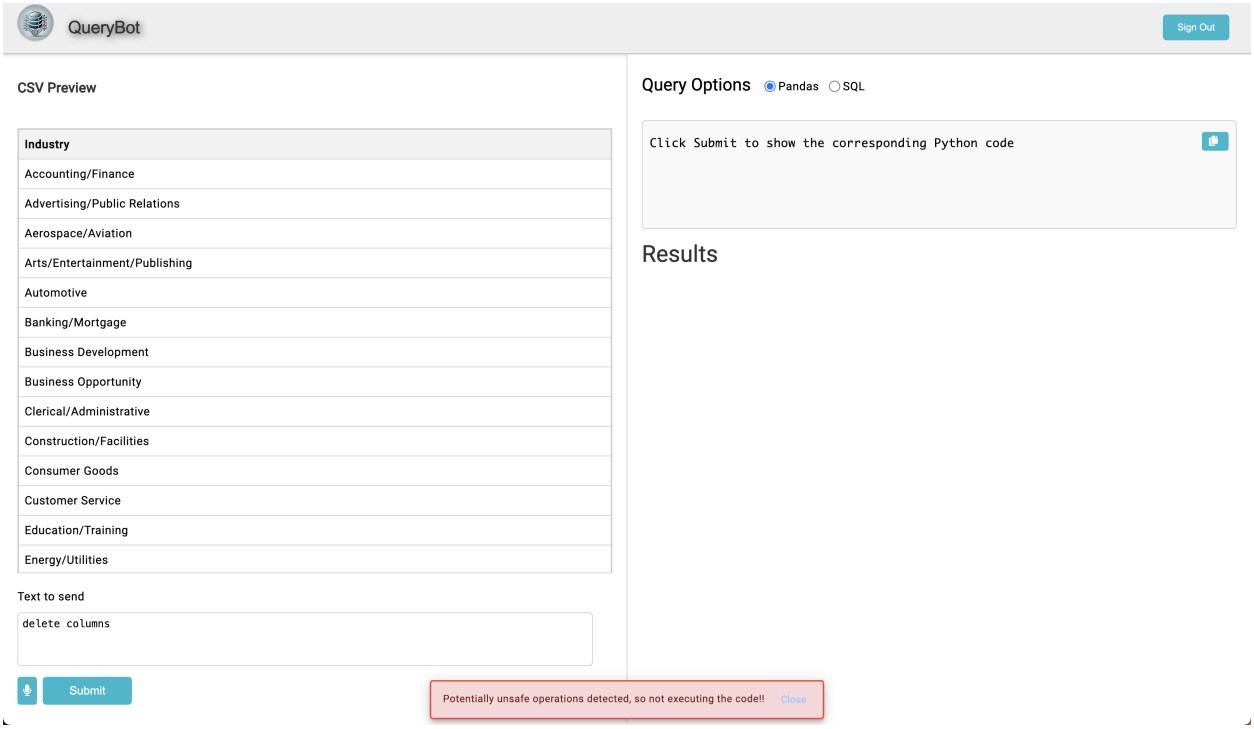

Figure 13: Unsafe Operation

Figure 13 shows the user, system generated notification error as shown in the figure. This gives the user a notification that user should not perform. This kind of real time notification helps user know what kid of operations are not allowed.

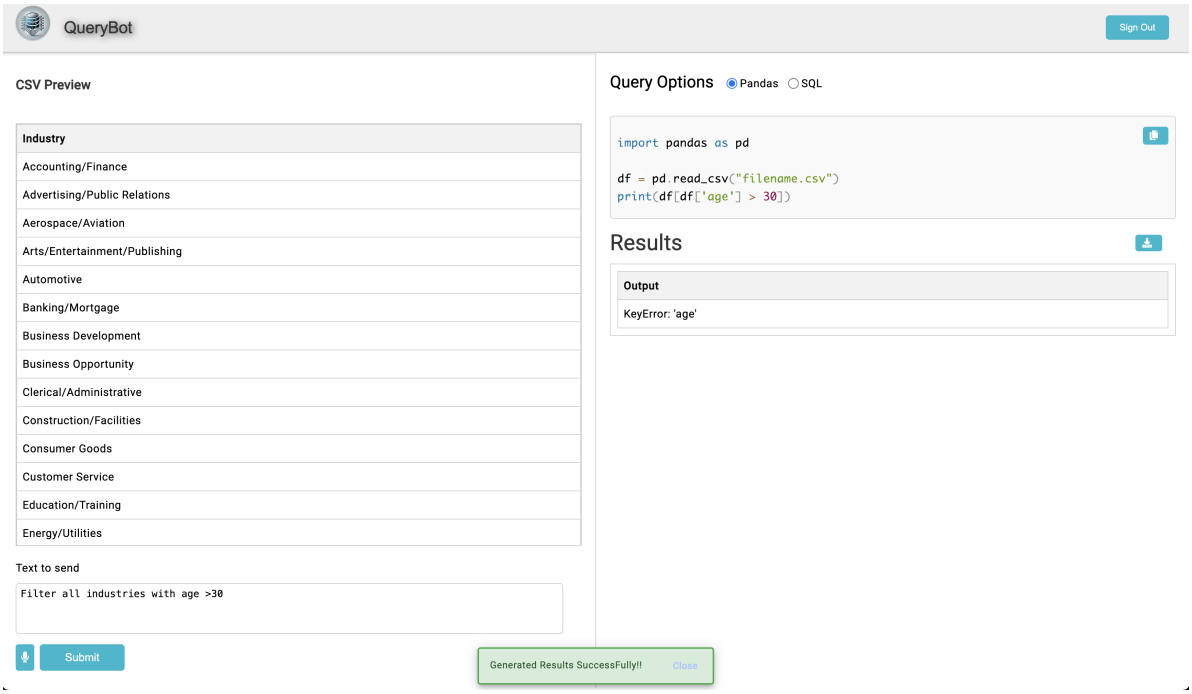

Figure 14: KeyError

Figure 14 shows the error. When the user requested for filter age¿30 the result given is KeyError"age" which shows that the table dosen't have the age column or row to analysis.

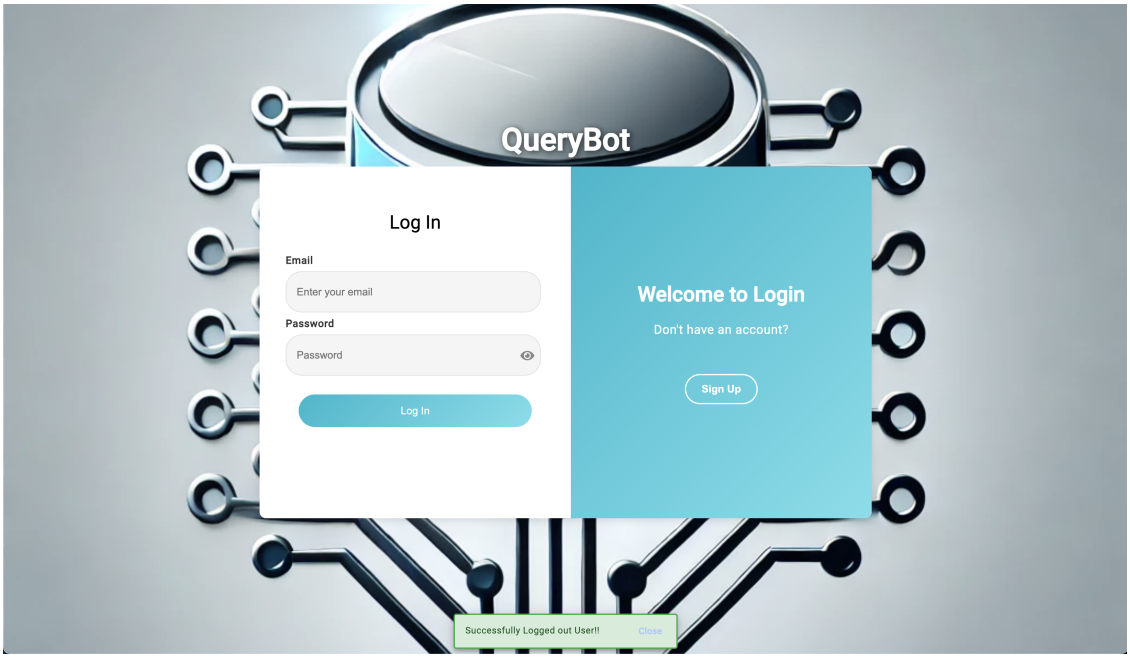

Figure 15: Signout

Figure 15 appears when the user clicks signout button. This redirects the website to the login page as shown and a real time notification is given saying the signout is successful.

8 Logger File

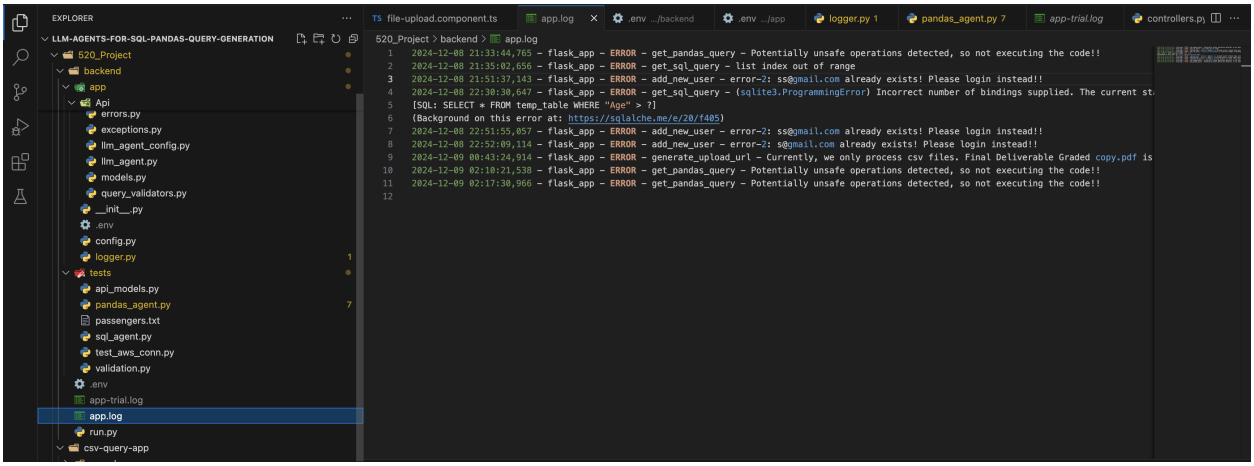

Figure 16: Logger File

Figure 16 shows the logger file on the admin side. This file logs every error that occurred in the website and shows in detail the type of error. This side of admin helps him understand and make required changes for any bugs or to track user path.

9 Demo Video

`https://drive.google.com/file/d/1WT693CTNA-ejxAjdLeu0GIlnYCjIxB4Y/view?usp=sharing`