GOVERNMENT COLLEGE OF TECHNOLOGY

PROJECT NAME: CSV QUERY BOT SYSTEM

PROJECT MEMBERS:

- SIVARANJANI S(71772117140)
- PRIYANKA S (71772117133)

DEPT: BE CSE

COURSE: ARTIFICIAL INTELLIGENCE

COURSE CODE:18SPC702

CSV Query Bot System

Project Overview:

Talk with CSV is an innovative tool designed to simplify data interaction and analysis by enabling users to query CSV data using natural language inputs. The solution aims to bridge the gap between non-technical users and complex data analysis, transforming how data is interpreted, analyzed, and presented.

Objectives:

The primary objectives of this project include:

- Enabling users to query CSV data using natural language.
- Providing meaningful responses, including textual data, tables, and visualizations.
- Ensuring a user-friendly interface that makes data analysis intuitive and accessible.

Technical Stack:

The project leverages a robust technical stack to deliver its functionalities:

- Streamlit: For building an interactive and user-friendly web interface.
- Python (Pandas): For efficient data manipulation and analysis.
- Google Generative AI (Gemini API): For processing and responding to natural language queries.
- dotenv & json Libraries: Used for environment configuration and data serialization.

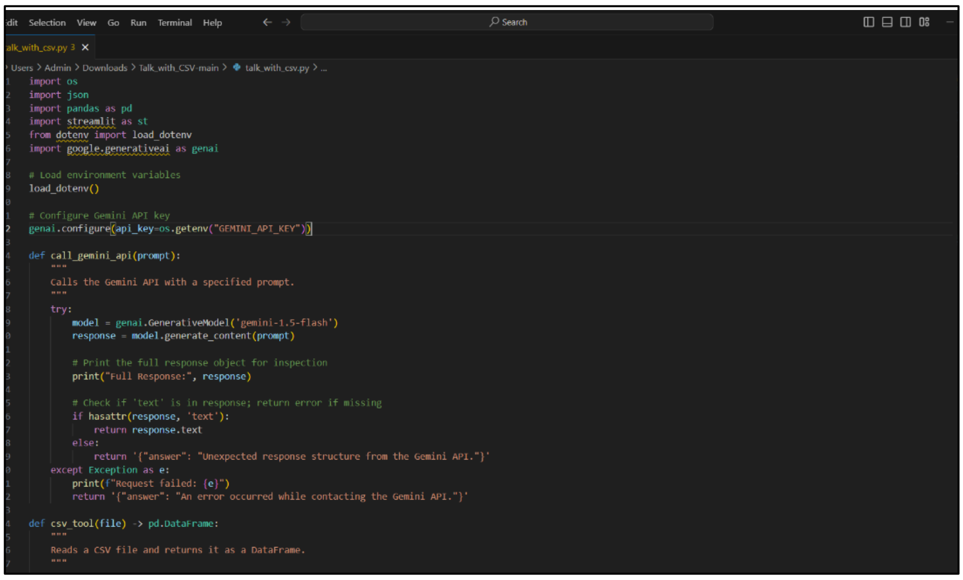
Project Architecture and Workflow:

- **CSV Upload**: Users begin by uploading their CSV file through the Streamlit web interface.
- **Natural Language Query**:Users can input a natural language query related to their data.
- **Query Processing**: The system constructs a prompt and sends it to the Gemini API for natural language processing.
- **Response Handling**: The API returns a response, which is then processed to generate the desired output (e.g., text, tables, visualizations).
- **Result Display**: The output is displayed on the Streamlit interface, making data interaction easy and intuitive for users.

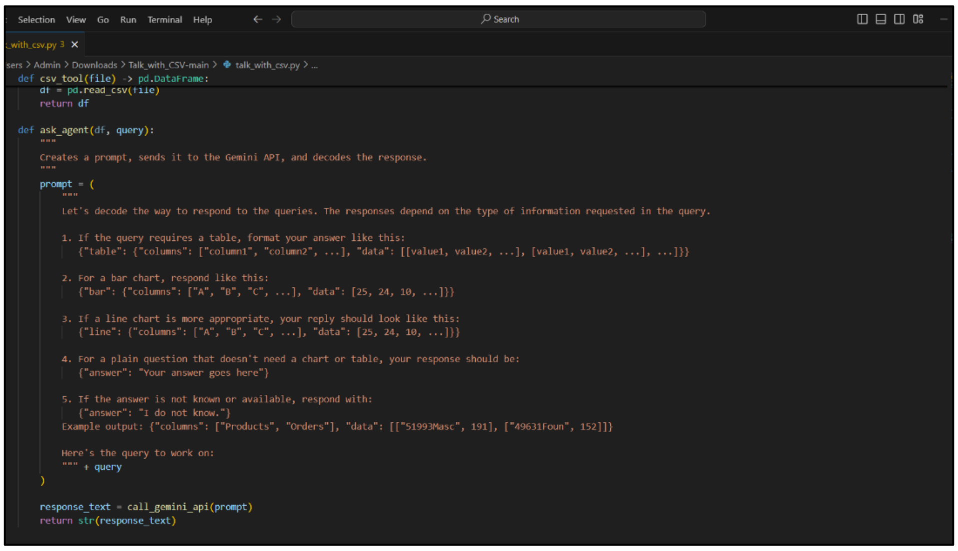
Key Features:

- **Natural Language Querying**: Users can interact with data using everyday language without the need for complex code or SQL queries.
- **User-Friendly Interface**: Built with Streamlit, the interface is designed to make data interaction seamless and accessible.

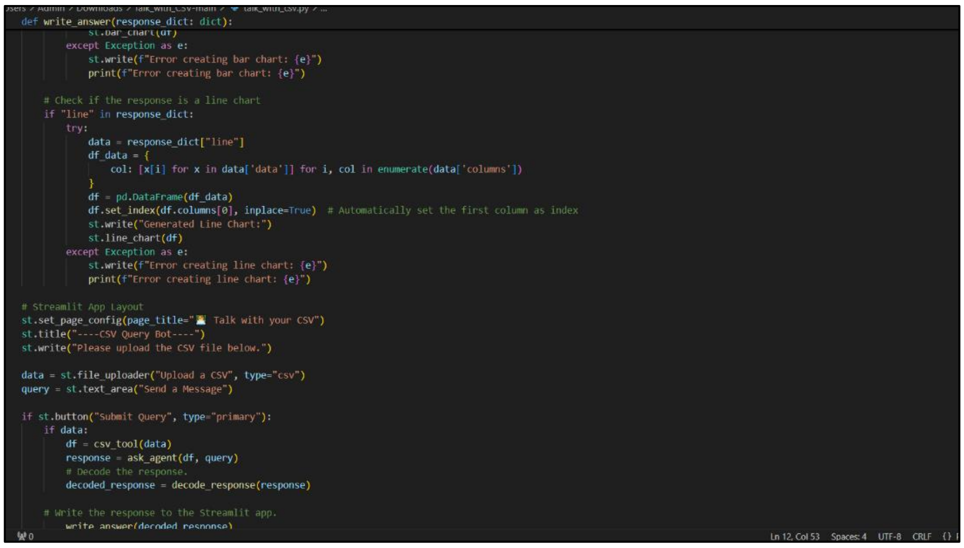
Code Implementation:



```python
import os
import json
import pandas as pd
import streamlit as st
from dotenv import load_dotenv
import google.generativeai as genai

# Load environment variables
load_dotenv()

# Configure Gemini API key
genai.configure(api_key=os.getenv("GEMINI_API_KEY"))

def call_gemini_api(prompt):
    """
    Calls the Gemini API with a specified prompt.
    """
    try:
        model = genai.GenerativeModel('gemini-1.5-flash')
        response = model.generate_content(prompt)

        # Print the full response object for inspection
        print("Full Response:", response)

        # Check if 'text' is in response; return error if missing
        if hasattr(response, 'text'):
            return response.text
        else:
            return '{"answer": "Unexpected response structure from the Gemini API."}'
    except Exception as e:
        print(f"Request failed: {e}")
        return '{"answer": "An error occurred while contacting the Gemini API."}'

def csv_tool(file) -> pd.DataFrame:
    """
    Reads a CSV file and returns it as a DataFrame.
    """
```



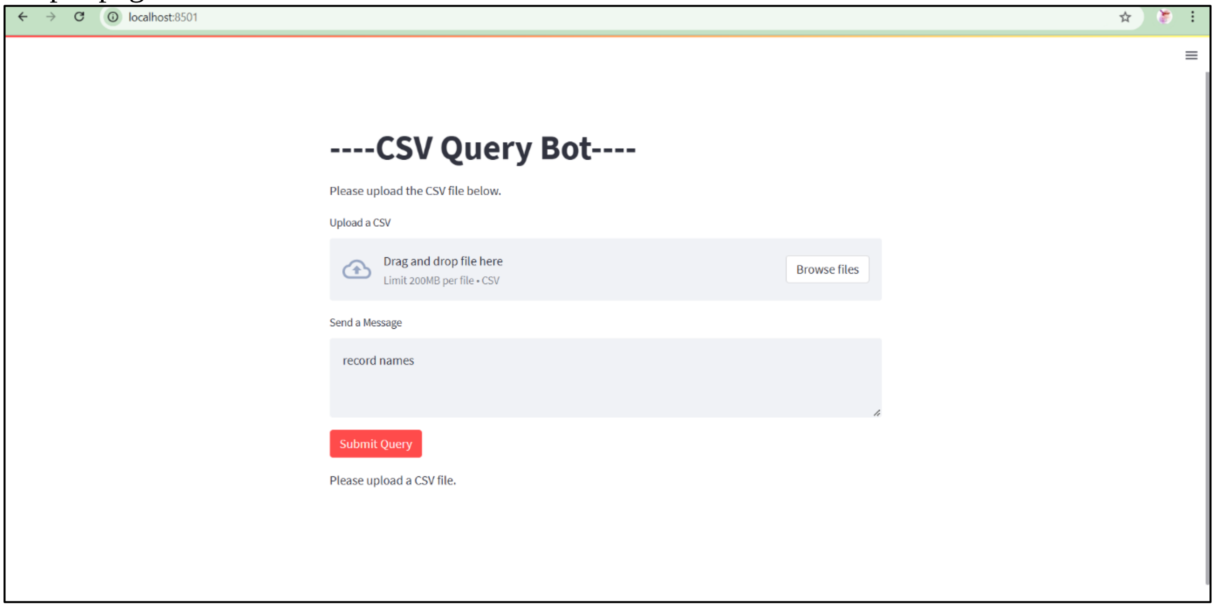
```python
def csv_tool(file) -> pd.DataFrame:
    df = pd.read_csv(file)
    return df

def ask_agent(df, query):
    """
    Creates a prompt, sends it to the Gemini API, and decodes the response.
    """
    prompt = (
        """
        Let's decode the way to respond to the queries. The responses depend on the type of information requested in the query.

        1. If the query requires a table, format your answer like this:
           {"table": {"columns": ["column1", "column2", ...], "data": [[value1, value2, ...], [value1, value2, ...], ...]}}

        2. For a bar chart, respond like this:
           {"bar": {"columns": ["A", "B", "C", ...], "data": [25, 24, 10, ...]}}

        3. If a line chart is more appropriate, your reply should look like this:
           {"line": {"columns": ["A", "B", "C", ...], "data": [25, 24, 10, ...]}}

        4. For a plain question that doesn't need a chart or table, your response should be:
           {"answer": "Your answer goes here"}

        5. If the answer is not known or available, respond with:
           {"answer": "I do not know."}
        Example output: {"columns": ["Products", "Orders"], "data": [["51993Masc", 191], ["49631Foun", 152]]}

        Here's the query to work on:
        """ + query
    )

    response_text = call_gemini_api(prompt)
    return str(response_text)
```



```python
def write_answer(response_dict: dict):
                st.bar_chart(df)
            except Exception as e:
                st.write(f"Error creating bar chart: {e}")
                print(f"Error creating bar chart: {e}")

    # Check if the response is a line chart
    if "line" in response_dict:
        try:
            data = response_dict["line"]
            df_data = {
                col: [x[i] for x in data['data']] for i, col in enumerate(data['columns'])
            }
            df = pd.DataFrame(df_data)
            df.set_index(df.columns[0], inplace=True)  # Automatically set the first column as index
            st.write("Generated Line Chart:")
            st.line_chart(df)
        except Exception as e:
            st.write(f"Error creating line chart: {e}")
            print(f"Error creating line chart: {e}")

# Streamlit App Layout
st.set_page_config(page_title="👨‍💻 Talk with your CSV")
st.title("----CSV Query Bot----")
st.write("Please upload the CSV file below.")

data = st.file_uploader("Upload a CSV", type="csv")
query = st.text_area("Send a Message")

if st.button("Submit Query", type="primary"):
    if data:
        df = csv_tool(data)
        response = ask_agent(df, query)
        # Decode the response.
        decoded_response = decode_response(response)

    # Write the response to the Streamlit app.
        write_answer(decoded_response)
```

Output Snapshots:

Output page:



----CSV Query Bot----

Please upload the CSV file below.

Upload a CSV

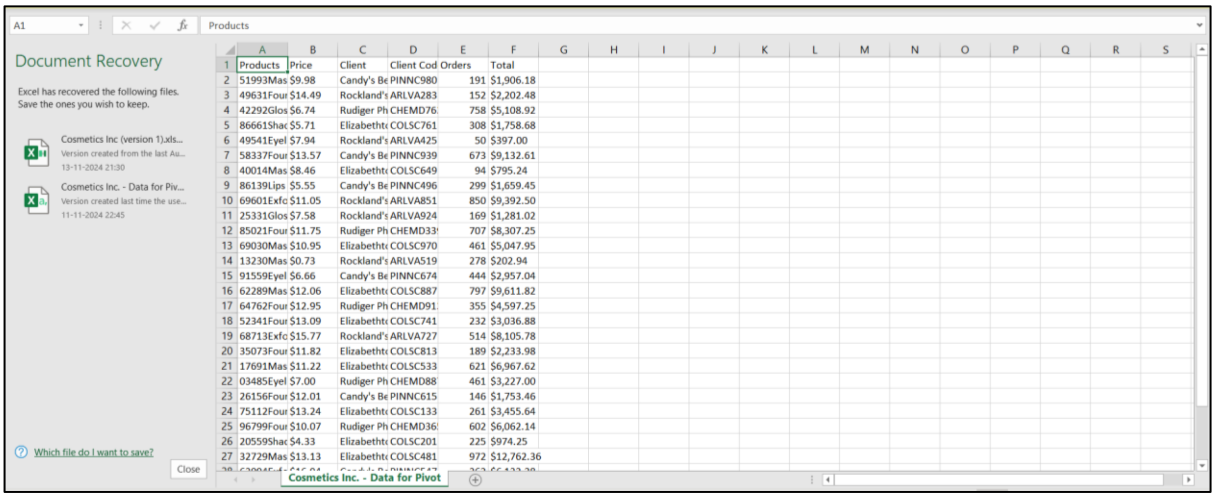
Drag and drop file here
Limit 200MB per file • CSV

Send a Message

record names

Please upload a CSV file.

CSV file:



Document Recovery

Excel has recovered the following files.
Save the ones you wish to keep.

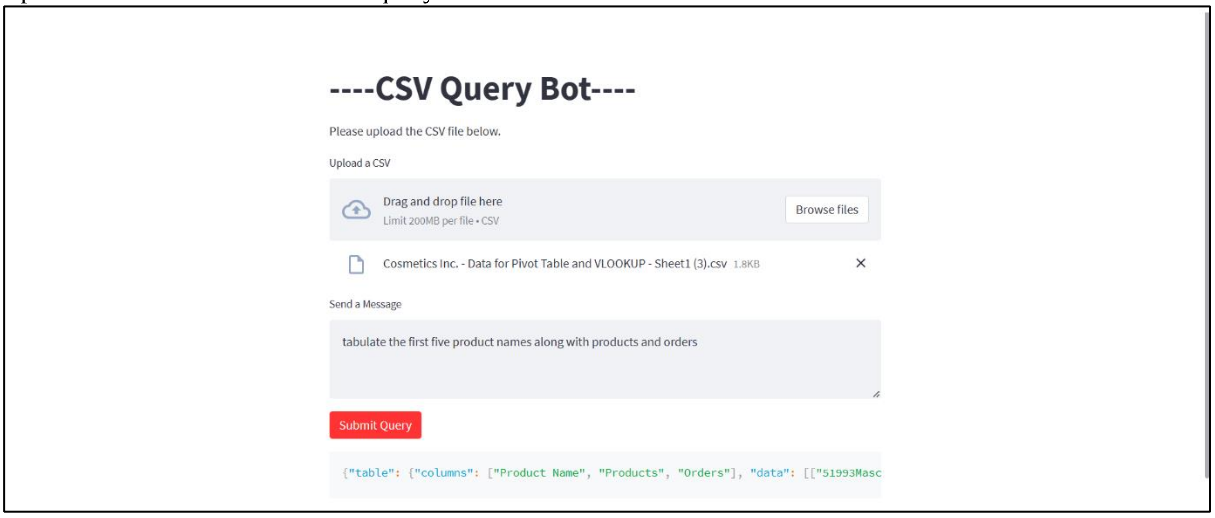
Cosmetics Inc (version 1).xls...
Version created from the last Au...
13-11-2024 21:30

Cosmetics Inc. - Data for Piv...
Version created last time the use...
11-11-2024 22:45

Which file do I want to save?
Close
Cosmetics Inc. - Data for Pivot

	A	B	C	D	E	F
1	Products	Price	Client	Client Cod	Orders	Total
2	51993Mas	$9.98	Candy's Be	PINNC980	191	$1,906.18
3	49631Four	$14.49	Rockland's	ARLVA283	152	$2,202.48
4	42292Glos	$6.74	Rudiger Ph	CHEMD76	758	$5,108.92
5	86661Shac	$5.71	Elizabetht	COLSC761	308	$1,758.68
6	49541Eyel	$7.94	Rockland's	ARLVA425	50	$397.00
7	58337Four	$13.57	Candy's Be	PINNC939	673	$9,132.61
8	40014Mas	$8.46	Elizabetht	COLSC649	94	$795.24
9	86139Lips	$5.55	Candy's Be	PINNC496	299	$1,659.45
10	69601Exfo	$11.05	Rockland's	ARLVA851	850	$9,392.50
11	25331Glos	$7.58	Rockland's	ARLVA924	169	$1,281.02
12	85021Four	$11.75	Rudiger Ph	CHEMD33	707	$8,307.25
13	69030Mas	$10.95	Elizabetht	COLSC970	461	$5,047.95
14	13230Mas	$0.73	Rockland's	ARLVA519	278	$202.94
15	91559Eyel	$6.66	Candy's Be	PINNC674	444	$2,957.04
16	62289Mas	$12.06	Elizabetht	COLSC887	797	$9,611.82
17	64762Four	$12.95	Rudiger Ph	CHEMD91	355	$4,597.25
18	52341Four	$13.09	Elizabetht	COLSC741	232	$3,036.88
19	68713Exfo	$15.77	Rockland's	ARLVA727	514	$8,105.78
20	35073Four	$11.82	Elizabetht	COLSC813	189	$2,233.98
21	17691Mas	$11.22	Elizabetht	COLSC533	621	$6,967.62
22	03485Eyel	$7.00	Rudiger Ph	CHEMD88	461	$3,227.00
23	26156Four	$12.01	Candy's Be	PINNC615	146	$1,753.46
24	75112Four	$13.24	Elizabetht	COLSC133	261	$3,455.64
25	96799Four	$10.07	Rudiger Ph	CHEMD36	602	$6,062.14
26	20559Shac	$4.33	Elizabetht	COLSC201	225	$974.25
27	32729Mas	$13.13	Elizabetht	COLSC481	972	$12,762.36

Uploaded CSV File and submitted query:



----CSV Query Bot----

Please upload the CSV file below.

Upload a CSV

Drag and drop file here
Limit 200MB per file • CSV

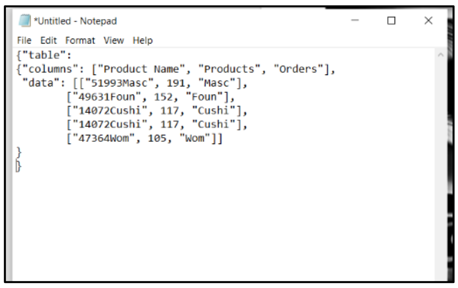
Cosmetics Inc. - Data for Pivot Table and VLOOKUP - Sheet1 (3).csv 1.8KB

Send a Message

tabulate the first five product names along with products and orders

```
{"table": {"columns": ["Product Name", "Products", "Orders"], "data": [["51993Masc
```

Desired output:



```
{"table":
{"columns": ["Product Name", "Products", "Orders"],
 "data": [["51993Masc", 191, "Masc"],
        ["49631Foun", 152, "Foun"],
        ["14072Cushi", 117, "Cushi"],
        ["14072Cushi", 117, "Cushi"],
        ["47364Wom", 105, "Wom"]]
}
}
```

Challenges Faced:

During the development of **Talk with CSV**, we encountered several challenges:

- Handling Complex Queries: Ensuring accurate responses for diverse and complex queries posed a significant challenge. This required fine-tuning of prompts sent to the Gemini API and error handling mechanisms.
- Data Processing Efficiency: Processing large CSV files efficiently while maintaining a smooth user experience demanded optimization efforts.
- User Interface Design: Balancing a simple, user-friendly interface with the provision of comprehensive functionality required iterative design improvements.

Solutions and Optimizations:

To overcome these challenges, we implemented the following solutions:

- Query Optimization: We improved the accuracy of responses by enhancing the prompt construction and integrating validation mechanisms.
- Efficient Data Handling:Using Python's Pandas library enabled us to process large datasets efficiently, reducing response times.
- User-Centric Design: Continuous feedback and usability testing allowed us to enhance the interface for improved user experience.

Real-World Applications:

The versatility of **Talk with CSV** makes it applicable in various scenarios, including:

- **Education**: Students can analyze and interpret data without extensive technical knowledge.
- **Business Analysis**: Professionals can gain insights from sales data, customer data, and more, using simple queries.
- **Data Exploration**: Researchers and data analysts can quickly explore and visualize datasets.

Future Enhancements:

We plan to further enhance **Talk with CSV** by:

- Adding Support for Additional Data Formats: Enabling interaction with other data formats beyond CSV files.
- Improving Natural Language Understanding: Refining query processing capabilities to handle more complex queries accurately.
- Expanding Visualization Options: Introducing more chart types and customizable data visualizations.
- Integrating Additional AI Capabilities: Leveraging other AI models to improve response generation and user engagement.