```
!pip install gradio google-auth google-auth-oauthlib google-auth-httplib2 google-api-python-client openai pandas
```

```
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.2.2)
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-api
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0.dev0,>=3.19.5 i
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*,!
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*,!
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.10/dist-packages (from h
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (1.0.6)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (3.16.1)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-au
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (2.23.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->goog
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.0,>=0.12
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.0,>=0.
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->type
Downloading gradio-5.5.0-py3-none-any.whl (56.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.7/56.7 MB 7.4 MB/s eta 0:00:00
Downloading gradio_client-1.4.2-py3-none-any.whl (319 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 319.8/319.8 kB 22.4 MB/s eta 0:00:00
Downloading python_multipart-0.0.12-py3-none-any.whl (23 kB)
Downloading tomlkit-0.12.0-py3-none-any.whl (37 kB)
Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.5-py3-none-any.whl (94 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.9/94.9 kB 6.4 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Downloading ruff-0.7.3-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.0/11.0 MB 51.1 MB/s eta 0:00:00
Downloading safehttpx-0.1.1-py3-none-any.whl (8.4 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.41.2-py3-none-any.whl (73 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.3/73.3 kB 6.5 MB/s eta 0:00:00
Downloading uvicorn-0.32.0-py3-none-any.whl (63 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.7/63.7 kB 4.1 MB/s eta 0:00:00
Downloading ffmpy-0.4.0-py3-none-any.whl (5.8 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (130 kB
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.2/130.2 kB 8.8 MB/s eta 0:00:00
Installing collected packages: pydub, websockets, uvicorn, tomlkit, semantic-version, ruff, python-multipart, markupsafe, ffmpy, aiof
  Attempting uninstall: markupsafe
    Found existing installation: MarkupSafe 3.0.2
    Uninstalling MarkupSafe-3.0.2:
      Successfully uninstalled MarkupSafe-3.0.2
Successfully installed aiofiles-23.2.1 fastapi-0.115.5 ffmpy-0.4.0 gradio-5.5.0 gradio-client-1.4.2 markupsafe-2.1.5 pydub-0.25.1 pyt
```

```python
import gradio as gr
import pandas as pd
import matplotlib.pyplot as plt
from io import BytesIO

def load_csv(file):
    # Ensure file is uploaded before attempting to read it
    if file is None:
        return None, gr.Dropdown.update(choices=[])

    try:
        # Read the CSV file
        df = pd.read_csv(file.name)
        columns = df.columns.tolist()

        # Display the first few rows for preview (convert to JSON for Gradio compatibility)
        data_preview = df.head(10).to_dict()  # Show first 10 rows for preview
        # Update the data preview and column dropdown
        return data_preview, gr.Dropdown.update(choices=columns)
    except Exception as e:
        # If there's an error reading the CSV, display the error message
        return str(e), gr.Dropdown.update(choices=[])

def parse_column_with_llm(df_json, selected_column):
    try:
        # Convert JSON back to a dataframe
        df = pd.DataFrame(df_json)

        if selected_column not in df.columns:
            return "No column selected.", None, None

        # Extract summary information (placeholder for LLM processing if available)
        column_data = df[selected_column]
        summary = column_data.describe().to_frame().to_string()

        # Plot the selected column's data (histogram and boxplot for numeric data)
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))
        column_data.plot(kind='hist', ax=ax1, title=f"{selected_column} Histogram", color='skyblue')
        column_data.plot(kind='box', ax=ax2, title=f"{selected_column} Boxplot")

        # Save the plot as an image in memory
        output_file = BytesIO()
        fig.savefig(output_file, format="png")
        output_file.seek(0)

        # Save the summary to a file for download
        summary_file_path = "/mnt/data/parsed_column_info.txt"
        with open(summary_file_path, "w") as f:
            f.write(f"Summary of '{selected_column}':\n\n{summary}")

        plt.close(fig)  # Close the figure to free memory
        return summary, output_file, summary_file_path
    except Exception as e:
        return str(e), None, None

# Set up Gradio interface
with gr.Blocks() as dashboard:
    gr.Markdown("### AI-powered Data Analysis and Extraction Dashboard")

    # Step 1: File Upload and Column Selection
    file_input = gr.File(label="Upload CSV File")
    data_preview = gr.Dataframe(label="Data Preview", interactive=False)
    column_select = gr.Dropdown(label="Select the Column to Analyze", choices=[])

    # Step 2: Process Selected Column
    summary_output = gr.Textbox(label="Summary Information")
    graph_output = gr.Image(label="Data Visualization")
    download_button = gr.File(label="Download Summary File")

    # Actions to load data and update column selection dropdown
    file_input.change(load_csv, inputs=file_input, outputs=[data_preview, column_select])
    column_select.change(parse_column_with_llm, inputs=[data_preview, column_select], outputs=[summary_output, graph_output, download_button

dashboard.launch()
```

```
Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://8a19dc1b14d0815d4e.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working c
```

gradio

No interface is running right now

```python
import pandas as pd
import matplotlib.pyplot as plt
from io import BytesIO

# Step 1: Load CSV File
def load_csv(file_path):
    """Load the CSV file and preview the first few rows."""
    df = pd.read_csv(file_path)
    print("Dataset Preview:")
    print(df.head())
    return df

# Step 2: Select Column and Analyze
def parse_column_with_llm(df, selected_column):
    """Analyze the selected column and return a summary and plots."""
    if selected_column not in df.columns:
        return "Error: Column not found in dataset.", None

    # Get summary statistics (placeholder for LLM if needed)
    column_data = df[selected_column]
    summary = column_data.describe().to_frame().to_string()
    print(f"\nSummary of '{selected_column}':\n{summary}")

    # Check if the column is numeric before plotting
    if pd.api.types.is_numeric_dtype(column_data):
        # Plot histogram and boxplot for numeric data
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))
        column_data.plot(kind='hist', ax=ax1, title=f"{selected_column} Histogram", color='skyblue')
        column_data.plot(kind='box', ax=ax2, title=f"{selected_column} Boxplot")

        # Save the figure
        plot_file = "column_plots.png"  # Saves in the current directory
        fig.savefig(plot_file)
        plt.close(fig)  # Free up memory
    else:
        # Skip plotting if data is non-numeric
        plot_file = None
        print(f"\nColumn '{selected_column}' is not numeric. Skipping plots.")

    # Save summary to a text file
    summary_file = "column_summary.txt"  # Saves in the current directory
    with open(summary_file, "w") as f:
        f.write(f"Summary of '{selected_column}':\n\n{summary}")

    print("\nSummary and visualization saved successfully.")
    return summary, plot_file, summary_file

# Example Usage
if __name__ == "__main__":
    # File path for the CSV file
    file_path = "/content/apple_quality.csv"

    # Load dataset and preview
    df = load_csv(file_path)

    # Choose a column to analyze (replace 'ColumnName' with the actual column name)
    selected_column = input("Enter the column to analyze: ")

    # Analyze and save results
    summary, plot_file, summary_file = parse_column_with_llm(df, selected_column)

    print(f"\nFiles generated:\nSummary: {summary_file}")
    if plot_file:
        print(f"Plot: {plot_file}")
    else:
        print("No plot generated.")
```

```
Dataset Preview:
   A_id      Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0   0.0 -3.970049 -2.512336   5.346330    -1.012009   1.844900  0.329840
1   1.0 -1.195217 -2.839257   3.664059     1.588232   0.853286  0.867530
2   2.0 -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -0.038033
3   3.0 -0.657196 -2.271627   1.324874    -0.097875   3.637970 -3.413761
4   4.0  1.364217 -1.296612  -0.384658    -0.553006   3.030874 -1.303849

        Acidity Quality
0  -0.491590483    good
1  -0.722809367    good
2   2.621636473     bad
3   0.790723217    good
4   0.501984036    good
Enter the column to analyze: Quality

Summary of 'Quality':
       Quality
count     4000
unique       2
top       good
freq      2004

Column 'Quality' is not numeric. Skipping plots.

Summary and visualization saved successfully.

Files generated:
Summary: column_summary.txt
No plot generated.
```

```
!pip install google-search-results
```

```
Collecting google-search-results
  Downloading google_search_results-2.4.2.tar.gz (18 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from google-search-results) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results) (2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results) (202
Building wheels for collected packages: google-search-results
  Building wheel for google-search-results (setup.py) ... done
  Created wheel for google-search-results: filename=google_search_results-2.4.2-py3-none-any.whl size=32009 sha256=c256c2ffba242af442857
  Stored in directory: /root/.cache/pip/wheels/d3/b2/c3/03302d12bb44a2cdff3c9371f31b72c0c4e84b8d2285eeac53
Successfully built google-search-results
Installing collected packages: google-search-results
```

```
!pip install google-search-results --upgrade # Ensure the correct package name and upgrade if already installed
```

```
Requirement already satisfied: google-search-results in /usr/local/lib/python3.10/dist-packages (2.4.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from google-search-results) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results) (3.10)
    Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results) (2.2
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->google-search-results) (202
```

```
from serpapi import GoogleSearch  # Change import statement

# Web Search Function
def web_search(query):
    """Perform a web search using SerpAPI."""
    search_params = {
        "q": query,
        "api_key": "3d2e268fe35d29b8eeddf69f6d2b79e11710d7d00e9ea0d97892bfeb7e4ecca8",  # Replace with your SerpAPI key
        "engine": "google"
    }
    search = GoogleSearch(search_params)
    results = search.get_dict()

    search_results = []
    for result in results.get("organic_results", []):
        search_results.append({
            "title": result.get("title"),
            "link": result.get("link"),
            "snippet": result.get("snippet")
        })

    return search_results
```

```
import pandas as pd

# Generate Queries Function
def generate_queries(df, column_name, template):
    """Generate search queries for each entity using the custom template."""
    queries = []

    # Check if the column exists in the DataFrame
    if column_name not in df.columns:
        print(f"Error: Column '{Juiciness}' not found in the DataFrame.")
        return queries

    # Iterate over the rows in the specified column
    for entity in df[column_name]:
        # Ensure the entity is a string and replace the placeholder in the template
        if isinstance(entity, str):  # Check if the entity is a string
            query = template.replace("{Juiciness}", entity)
        else:
            query = template.replace("{Juiciness}", str(entity))  # Convert to string if not already

        queries.append(query)

        # Debugging: Print the query for each entity
        print(f"Generated Query: {query}")

    return queries
```

```
import openai

# LLM Information Extraction Function
def extract_information_with_llm(search_results, prompt_template):
    """Send search results to GPT-3 (or another LLM) for information extraction."""
    openai.api_key = "sk-proj-vMSA3VSa99bQRfmW6w93SZaE7qgeolah7GKKFEeSMXfukDI4kGHrrO-b1G8MnZZX449HYXcP2DT3BlbkFJrR9FwT-ytssgue_pVUtx250EilHtt

    extracted_info = []
    for result in search_results:
        prompt = prompt_template.replace("{Juiciness}", result["title"]) + "\n\n" + result["snippet"]
        try:
            response = openai.Completion.create(
                engine="text-davinci-003",  # or another model
                prompt=prompt,
                max_tokens=100
            )
            extracted_info.append(response.choices[0].text.strip())
        except Exception as e:
            extracted_info.append(f"Error: {str(e)}")
    return extracted_info



# Display and Save Results Function
def display_extracted_data(df, extracted_info):
    """Display extracted information in a user-friendly format and provide download options."""
    df['Extracted Information'] = extracted_info
    print("Extracted Data Preview:")
    print(df[['Juiciness', 'Extracted Information']].head())

    # Save the updated DataFrame to a CSV file
    output_csv = "extracted_information.csv"
    df.to_csv(output_csv, index=False)

    print(f"\nData saved to {output_csv}")
    return output_csv
```

Start coding or generate with AI.