import pandas as pd

import matplotlib.pyplot as plt

import streamlit as st

import google.generativeai as genai

# Initialize Gemini client (replace 'your\_api\_key' with your actual API key)

genai.configure(api\_key="your\_api\_key\_here")

# Function to load CSV file

def load\_csv():

    uploaded\_file = st.file\_uploader("Choose a CSV file", type="csv")

    if uploaded\_file is not None:

        df = pd.read\_csv(uploaded\_file)

        return df

    return None

# Function to perform basic statistical analysis

def basic\_statistics(df):

    stats = {

        'mean': df.mean(numeric\_only=True).to\_dict(),

        'median': df.median(numeric\_only=True).to\_dict(),

        'mode': df.mode(numeric\_only=True).iloc[0].to\_dict(),

        'std\_dev': df.std(numeric\_only=True).to\_dict(),

        'correlation': df.corr().to\_dict()

    }

    return stats

# Function to plot bar graph

def plot\_bar(df, column, top\_n=15):

    # Sort DataFrame by the specified column in descending order and select top N rows

    sorted\_df = df.sort\_values(by=column, ascending=False).head(top\_n)

    # Check if 'Book Title' and 'Rating' columns exist in sorted\_df

    if 'Book Title' not in sorted\_df.columns or 'Rating' not in sorted\_df.columns:

        st.write("Error: One or both of the columns 'Book Title' or 'Rating' not found in DataFrame.")

        return

    # Trim 'Book Title' to 15 characters

    sorted\_df['Book Title'] = sorted\_df['Book Title'].str.slice(0, 15)

    # Create the bar plot

    fig, ax = plt.subplots(figsize=(10, 6))

    ax.bar(sorted\_df['Book Title'], sorted\_df['Rating'], color='blue')

    ax.set\_xlabel('Book Title')

    ax.set\_ylabel('Rating')

    ax.set\_title(f'Top {top\_n} Books by Rating')

    ax.set\_xticklabels(sorted\_df['Book Title'], rotation=45, ha='right')

    plt.tight\_layout()

    st.pyplot(fig)

# Function to plot scatter graph

def plot\_scatter(df, x\_column, y\_column):

    # Check if the specified columns exist in the DataFrame

    if x\_column not in df.columns or y\_column not in df.columns:

        st.write(f"Error: One or both of the columns '{x\_column}' or '{y\_column}' not found in DataFrame.")

        return

    # Create the scatter plot

    fig, ax = plt.subplots(figsize=(10, 6))

    ax.scatter(df[x\_column], df[y\_column], color='blue', alpha=0.5)

    ax.set\_title(f'Scatter plot of {x\_column} vs {y\_column}')

    ax.set\_xlabel(x\_column)

    ax.set\_ylabel(y\_column)

    plt.tight\_layout()

    st.pyplot(fig)

# Function to plot line graph

def plot\_line(df, x\_column, y\_column):

    # Check if the specified columns exist in the DataFrame

    if x\_column not in df.columns or y\_column not in df.columns:

        st.write(f"Error: One or both of the columns '{x\_column}' or '{y\_column}' not found in DataFrame.")

        return

    # Trim 'x\_column' to 15 characters

    df[x\_column] = df[x\_column].astype(str).str.slice(0, 15)

    # Create the line plot

    fig, ax = plt.subplots(figsize=(10, 6))

    ax.plot(df[x\_column], df[y\_column], marker='o', linestyle='-', color='blue', label=y\_column)

    ax.set\_title(f'Line plot of {y\_column} over {x\_column}')

    ax.set\_xlabel(x\_column)

    ax.set\_ylabel(y\_column)

    ax.legend()

    plt.tight\_layout()

    st.pyplot(fig)

# Function to generate response using Gemini API

def generate\_llm\_response(df, question):

    try:

        prompt = f"Dataset: {df.head(5).to\_string()} \n\nQuestion: {question}\nAnswer:"

        response = genai.generate\_text(prompt=prompt)

        # Check if response is valid and has 'text' attribute

        if response and hasattr(response, 'text'):

            return response.text

        else:

            return "No valid response found. Please check your query or try again later."

    except Exception as e:

        return f"Error: {str(e)}"

# Streamlit app

def main():

    st.title("TensorGo Assignment")

    # Load CSV

    df = load\_csv()

    if df is not None:

        st.write("Dataframe Loaded Successfully:")

        st.write(df.head())

        # Display basic statistics

        st.subheader("Basic Statistics")

        stats = basic\_statistics(df)

        st.write(stats)

        # Plot options

        st.subheader("Plots")

        # Bar Plot

        column\_to\_plot = st.selectbox('Select column for Bar Plot', df.columns)

        if st.button('Plot Bar Graph'):

            plot\_bar(df, column\_to\_plot)

        # Scatter Plot

        x\_column\_scatter = st.selectbox('Select X column for Scatter Plot', df.columns)

        y\_column\_scatter = st.selectbox('Select Y column for Scatter Plot', df.columns)

        if st.button('Plot Scatter Graph'):

            plot\_scatter(df, x\_column\_scatter, y\_column\_scatter)

        # Line Plot

        x\_column\_line = st.selectbox('Select X column for Line Plot', df.columns)

        y\_column\_line = st.selectbox('Select Y column for Line Plot', df.columns)

        if st.button('Plot Line Graph'):

            plot\_line(df, x\_column\_line, y\_column\_line)

        # Generate LLM Response

        st.subheader("Ask a Question about the Data")

        question = st.text\_input("Enter your question:")

        if st.button('Get Answer'):

            answer = generate\_llm\_response(df, question)

            st.write("Answer:")

            st.write(answer)

if \_\_name\_\_ == "\_\_main\_\_":

    main()