Requirements Document for CSV Data Analysis Application

Overview

This document outlines the requirements for a **CSV data analysis application implemented in Python**. The application consists of a Python script for **command-line interaction** and a **Streamlit application** for **web-based interaction**. Both components provide functionality for analyzing and visualizing data from a CSV file named Data.csv.

Requirements

Functional Requirements

1. **CSV File Handling**
   - The application must be able to read data from a **CSV file** named **Data.csv.**
   - The CSV file is expected to contain **various columns of data**.
2. **Display Column Names**
   - The application must list all column names present in the CSV file.
3. **Display Index Values**
   - The application must display index values for each column in the CSV file.
4. **View Row by Index**
   - The application must allow the user to view data for a specific row by providing an index value.
   - The user must be able to input the index value and receive the corresponding row data.
5. **Display Column Statistics**
   - The application must provide statistical summaries for selected columns.
     - **Numeric Columns**: **Minimum, Maximum, Mean, Mode,** and **Median values**.
     - **Columns with '$' values**: Must handle values formatted with '**$**' and **commas**, converting them to numeric values for statistical analysis.
6. **Plot Columns**
   - The application must allow the user to plot data from one column against another column or the index.
   - The plot should be a scatter plot showing the relationship between the selected columns.
7. **Interactive User Experience**
   - The application must provide a user-friendly interface for interacting with the data.
   - Users should be able to choose columns, input index values, and view results or plots through both the command-line interface and the Streamlit web application.
8. **Continue or Exit**
   - The application must give users the option to continue using the program or exit after performing operations.

Non-Functional Requirements

1. **Usability**
   - The application must be easy to use and understand for users with basic knowledge of data analysis.
   - Error messages and instructions should be clear and helpful.
2. **Performance**
   - The application should perform efficiently with reasonably sized CSV files.
   - Loading and processing times should be acceptable for typical use cases.
3. **Compatibility**
   - The application must be compatible with Python 3.x.
   - Dependencies must include Pandas, Matplotlib, Colorama, and Streamlit.
4. **Error Handling**
   - The application must handle errors gracefully, such as invalid inputs or missing data.
   - The user should receive informative error messages in case of issues.
5. **Documentation**
   - The application must include clear documentation for installation, usage, and functionality.
   - Both the command-line script and the Streamlit application should be documented with examples.

Technical Requirements

1. **Python Version**
   - The application should be developed and tested using Python 3.6 or later.
2. **Libraries and Dependencies**
   - **Pandas**: For data manipulation and analysis.
   - **Matplotlib**: For plotting data.
   - **Colorama**: For colored terminal text.
   - **Streamlit**: For creating a web-based interface.
3. **File Format**
   - The CSV file must be in a standard CSV format with a header row and data rows.
   - The file must be named Data.csv and located in the same directory as the application.
4. **Code Quality**
   - The code must adhere to standard coding practices and be well-organized.
   - The application should include comments and documentation within the code for clarity.

Example Use Cases

1. **Display Column Names**
   - **Input**: User runs the script or accesses the Streamlit app.
   - **Output**: List of column names displayed.
2. **Display Index Values**
   - **Input**: User requests index values for columns.
   - **Output**: Index values for each column are displayed.

3. **View Row by Index**
    - **Input**: User provides an index value.
    - **Output**: Data for the specified row is displayed.
4. **Display Column Statistics**
    - **Input**: User selects a column for statistics.
    - **Output**: Statistical values for the selected column are shown.
5. **Plot Columns**
    - **Input**: User selects two columns for plotting.
    - **Output**: A scatter plot of the two columns is generated.
6. **Continue or Exit**
    - **Input**: User decides to continue or exit the application.
    - **Output**: Application either continues with additional operations or exits.

2. Python Script Implementation

Overview

The Python script is designed to read data from a CSV file (Data.csv) and perform various data analysis tasks. It uses pandas for data manipulation, matplotlib for plotting, and colorama for colored terminal text.

Key Features

1. **Read CSV File**
   - The script reads the CSV file into a DataFrame using pandas.
   - df = pd.read_csv("Data.csv")
2. **Display Column Names**
   - It prints out all column names from the DataFrame.
   - Uses print(Fore.YELLOW + "Requirement 1 - Column Names:", Fore.GREEN) and a for loop to iterate over df.columns.
3. **Display Index Values**
   - Displays index values for each column.
   - Uses a for loop to print index values: print(Fore.YELLOW + "Index Values for Column:", Fore.GREEN + "", column, Fore.YELLOW + " - ", Fore.GREEN + "", df[column].index)
4. **View Row by Index**
   - Prompts the user to enter an index value to view a specific row.
   - Checks if the input is a valid number and if it is within the DataFrame's length.
   - Prints the row data or an error message.
5. **Display Column Statistics**
   - Provides statistics for a selected column, including minimum, maximum, mean, median, and mode values.
   - Handles numeric columns and columns with monetary values formatted with $ and commas.
   - Uses conditional checks to apply appropriate statistical functions based on the column type.
6. **Plot Columns**
   - Allows the user to plot data from one column against another column.
   - Uses matplotlib to create scatter plots.
   - df.plot(x=column1, y=column2, kind="scatter")
7. **Continue or Exit**
   - Prompts the user to continue with the program or exit.
   - Repeats the loop or exits based on user input.

Code Structure

- **Imports**: Imports necessary libraries (pandas, matplotlib, colorama).
- **Initialization**: Initializes colorama for colored output.
- **Main Code Execution**: Includes logic for displaying column names, index values, row data, statistics, and plots.

- **User Interaction**: Handles user input for various actions (viewing rows, displaying statistics, plotting).

2. Streamlit Application Implementation

Overview

The Streamlit application provides a web-based interface for similar functionalities as the command-line script. It allows users to interact with the CSV data through a web app.

Key Features

1. **Load CSV File**
   - Reads the CSV file into a DataFrame using `pandas`.
   - `df = load_data()`
2. **Display Column Names**
   - Displays all column names in the web app using Streamlit's `st.write()`.
3. **View Row by Index**
   - Provides a dropdown to select a column and an input field to enter an index value.

- Displays the selected row based on the input index using st.write().

4. **Display Column Statistics**
   - Allows users to select a column and displays statistical summaries for that column.
   - Handles numeric columns and columns with monetary values.
   - Uses st.write() to display statistics and st.error() or st.warning() for error messages.
5. **Plot Columns**
   - Provides dropdowns to select columns for the X-axis and Y-axis.
   - Uses matplotlib to create scatter plots and st.pyplot() to display the plot in the web app.
6. **Exit Option**
   - Provides an "Exit" button to stop the application.

Code Structure

- **Imports**: Imports libraries (pandas, matplotlib, streamlit).
- **Functions**: Defines functions for loading data, displaying column names, viewing rows, showing statistics, and plotting data.
- **Main Function**: Runs the Streamlit app and calls the functions to provide the user interface.

Example Usage

- **Command-Line Script**:
  - Run the script using python script_name.py.
  - Follow prompts to display column names, view rows, show statistics, and plot data.
- **Streamlit App**:
  - Run the app using streamlit run app.py.
  - Interact with the web interface to load data, view columns, display statistics, and plot data.