**Data Exploration and Analysis by SIEMENS**

**Project (a) Description**

**Title**: Multiple Y-Axis Plotter

**Introduction**:

The "Multiple Y-Axis Plotter" is a Python script designed for data visualization tasks, particularly when dealing with CSV datasets. It allows users to select and plot multiple datasets on the same graph with a single X-axis and multiple Y-axes. This tool is beneficial for analyzing data with varying scales and units, providing a clear and comprehensive view of the relationships between different variables.

**Features**:

Interactive Column Selection: Users can interactively select columns from a CSV dataset for both the X-axis and multiple Y-axes by providing search strings to match column names. This feature enhances ease of use, especially when working with large datasets with numerous columns.

Customizable Colors: The script provides a range of predefined colors for each plotted dataset, ensuring visual distinction between different variables. Users can customize the color scheme for better visualization.

Dynamic Legends: Legends are generated dynamically for all plotted datasets, making it easy to identify which dataset corresponds to each Y-axis. Legends are placed at the upper center of the plot for convenience.

Adjustable X-Axis: Users can specify the desired range for the X-axis, providing flexibility in focusing on specific time periods or data points of interest. The script calculates the middle partition of the X-axis for improved data exploration.

**How to Use:**

X-Axis Selection: Users enter a search string to filter and select the desired X-axis column from the CSV dataset.

Y-Axis Selection: Users specify the number of Y-axes they want to plot. For each Y-axis, they provide a search string to filter and select the corresponding column from the dataset.

Customization: Users can customize the color scheme for each plotted dataset and set the X-axis range to focus on specific data points.

Interactive Plot: The script generates an interactive plot with multiple Y-axes, each representing a selected dataset. Legends provide a clear association between Y-axes and datasets.

Visualization: Users can explore and analyze the data relationships within the plot, including identifying trends, correlations, and patterns.

**Benefits:**

Data Exploration: The Multiple Y-Axis Plotter simplifies the process of exploring complex datasets with multiple variables, aiding in data-driven decision-making.

Visualization: The script provides an effective visualization tool for comparing and contrasting multiple datasets on a single graph.

Interactivity: Users have the flexibility to interactively select columns and customize the plot, making it adaptable to various data analysis tasks.

Clarity: Different colors and legends ensure that users can easily distinguish between different datasets, improving the clarity of the plot.

Efficiency: The script streamlines the process of creating multiple Y-axis plots, saving time and effort in data visualization tasks.

**Conclusion:**

The "Multiple Y-Axis Plotter" script is a valuable tool for data analysts and scientists working with CSV datasets. It simplifies the process of visualizing and comparing multiple datasets on a single graph, enhancing data exploration and analysis capabilities. Its interactivity and customization options make it a versatile choice for a wide range of data visualization tasks, contributing to informed decision-making and insights extraction from complex datasets.

**CODE­­­­­­­**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

file = pd.read\_csv(r'C:\Users\HP G5\Desktop\NEW PROJECTT.csv')

search\_str = input('Enter the string to search for x axis: ')

matched = []

serial\_num = 1

for col in file.columns:

if search\_str in col:

matched.append((serial\_num, col))

serial\_num += 1

for serial, column\_name in matched:

print(f'{serial}.{column\_name}')

if not matched:

print('The string does not match')

else:

num = int(input('Enter the number to select: '))

if 1 <= num <= len(matched):

selected\_column = matched[num - 1][1]

print('Selected column: ', selected\_column)

x = selected\_column

y\_count = int(input('Enter the number of Y-columns: '))

y\_names = []

colors = ['blue', 'green', 'red', 'purple', 'orange', 'brown', 'pink']

for i in range(y\_count):

search\_str2 = input('Enter the string to search for y axis: ')

matched2 = []

serial\_num2 = 1

for col in file.columns:

if search\_str2 in col:

matched2.append((serial\_num2, col))

serial\_num2 += 1

for serial2, column\_name2 in matched2:

print(f'{serial2}.{column\_name2}')

if not matched2:

print('The string does not match')

else:

num2 = int(input('Enter the number to select: '))

if 1 <= num2 <= len(matched2):

selected\_column2 = matched2[num2 - 1][1]

print('Selected column: ', selected\_column2)

y\_names.append(selected\_column2)

fig, ax = plt.subplots()

x\_data = file[x]

y1\_data = file[y\_names[0]]

line1, = ax.plot(x\_data, y1\_data, label=y\_names[0])

ax.set\_xlabel(x,color ='r')

y1\_label = ax.set\_ylabel(y\_names[0], color=colors[0])

ax.legend(loc='upper center')

x\_min = min(x\_data)

x\_max = max(x\_data)

interval = (x\_max - x\_min) / 2

plt.xticks(np.arange(x\_min, x\_max + 1, interval))

legends = [line1]

for i in range(1, len(y\_names)):

ax\_new = ax.twinx()

y\_data = file[y\_names[i]]

line, = ax\_new.plot(x\_data, y\_data, label=y\_names[i], color=colors[i])

ax\_new.plot(x\_data, y\_data, label=y\_names[i], color=colors[i])

ax\_new.set\_ylabel(y\_names[i], color=colors[i])

ax\_new.spines['right'].set\_position(('outward', i \* 50))

#ax\_new.legend(loc='upper right')

legends.append(line)

ax.legend(legends, [line.get\_label() for line in legends], loc='upper center')

plt.title('Multiple Y-Columns Plot')

plt.show()

Enter the string to search for x axis: AB

1. AB01

2. AB02

3. AB03

4. AB04

Enter the number to select: 1

Selected column: AB01

Enter the number of Y-columns: 3

Enter the string to search for y axis: AB

1. AB01

2. AB02

3. AB03

4. AB04

Enter the number to select: 2

Selected column: AB02

Enter the string to search for y axis: AB

1. AB01

2. AB02

3. AB03

4. AB04

Enter the number to select: 3

Selected column: AB03

Enter the string to search for y axis: AB

1. AB01

2. AB02

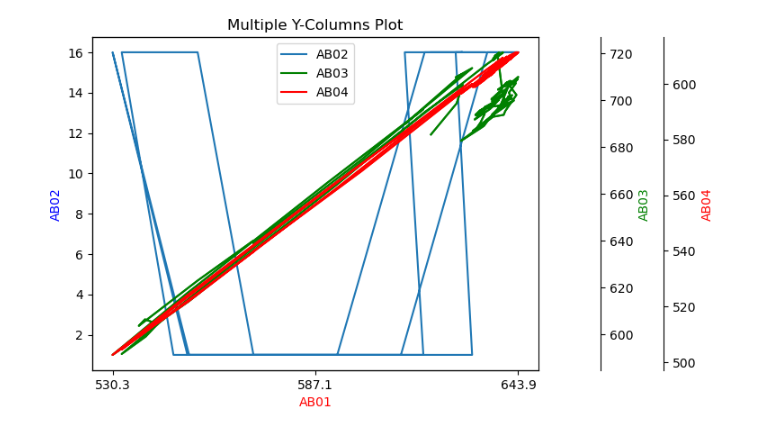
3. AB03

4. AB04

Enter the number to select: 4

Selected column: AB04

**OUTPUT**



**Project (b) Description**

**Title**: Multi-Y-Axis Time Series Data Plotter

**Introduction**:

The "Multi-Y-Axis Time Series Data Plotter" is a Python script designed to visualize and analyze time series data from CSV files. This tool allows users to interactively select columns for the X-axis and multiple Y-axes, making it easy to explore relationships between different variables over time. It is particularly useful for tasks that involve the analysis of time-stamped data with varying scales and units.

**Features**:

Interactive Column Selection: Users can interactively choose columns for both the X-axis (time) and multiple Y-axes by specifying search strings to match column names. This feature simplifies the selection process, especially when dealing with extensive datasets.

Customizable Colors: The script provides a range of predefined colors for each plotted dataset, ensuring clear visual differentiation between different variables. Users can customize the color scheme to enhance visualization.

Dynamic Legends: Legends are generated dynamically for all plotted datasets, enabling users to identify the relationship between Y-axes and datasets. Legends are conveniently placed at the upper center of the plot.

Adjustable X-Axis: Users can specify the desired time range (in the format dd-mm-yyyy HH:MM) for the X-axis, allowing them to focus on specific time periods or data points of interest. The script calculates the middle partition of the X-axis for improved data exploration.

Efficient Data Filtering: The script filters the dataset based on the selected X-axis range, ensuring that only relevant data points are visualized, which is especially useful for large datasets.

**How to Use:**

X-Axis Selection: Users provide a search string to filter and select the desired X-axis column, representing time, from the CSV dataset.

Y-Axis Selection: Users specify the number of Y-axes they want to plot and provide search strings for each Y-axis. The script matches these search strings to columns in the dataset.

Customization: Users can customize the color scheme for each plotted dataset and set the X-axis range to focus on specific time periods.

Interactive Plot: The script generates an interactive plot with multiple Y-axes, each representing a selected dataset. Legends provide a clear association between Y-axes and datasets.

Data Exploration: Users can explore and analyze the data relationships within the plot, including identifying trends, correlations, and patterns over time.

**Benefits**:

Time Series Analysis: The Multi-Y-Axis Time Series Data Plotter simplifies the analysis of time series data by providing an efficient and interactive visualization tool.

Visualization: Users can visualize multiple variables over time on a single graph, making it easier to identify patterns and relationships.

Interactivity: The script offers interactive column selection and customization options, making it adaptable to various time-based data analysis tasks.

Clarity: Different colors and legends ensure that users can easily distinguish between different datasets, improving the clarity of the plot.

Efficiency: The script efficiently filters data based on the selected time range, enhancing the speed and relevance of the visualization.

**Conclusion**:

The "Multi-Y-Axis Time Series Data Plotter" script is a valuable tool for data analysts and scientists working with time series data in CSV format. It simplifies the process of visualizing and analyzing multiple variables over time, providing insights into complex datasets. Its interactivity and customization options make it versatile for a wide range of time-based data analysis tasks, contributing to informed decision-making and insights extraction from time-stamped datasets.

**CODE**

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

# Read data from the imported file (update the file path accordingly)

df = pd.read\_csv(r"C:\Users\HP G5\Desktop\IDS292B\_20221007073000\_tdy285.csv")

# Prompt the user to enter the search string for X-axis

search\_string\_x = input("Enter the search string for X-axis: ")

matching\_columns\_x = [col for col in df.columns if search\_string\_x in col]

if matching\_columns\_x:

for i, col in enumerate(matching\_columns\_x, 1):

print(f"{i}. {col}")

if len(matching\_columns\_x) == 1:

selected\_column\_x = matching\_columns\_x[0]

print(f"Selected column for X-axis: {selected\_column\_x}")

else:

while True:

try:

selection\_x = int(input("Enter the serial number of the column for X-axis: "))

if selection\_x >= 1 and selection\_x <= len(matching\_columns\_x):

selected\_column\_x = matching\_columns\_x[selection\_x - 1]

print(f"Selected column for X-axis: {selected\_column\_x}")

break

else:

print("Invalid selection. Please enter a valid serial number.")

except ValueError:

print("Invalid input. Please enter a valid serial number.")

else:

print("No matching columns found for X-axis.")

# Prompt the user to enter the search string and serial number for each Y-axis

y\_columns = []

y\_labels = []

colors = ['blue', 'green', 'red', 'purple', 'orange']

num\_columns = int(input("Enter the number of Y-axis columns: "))

for i in range(num\_columns):

search\_string\_y = input(f"Enter the search string for Y-axis column {i+1}: ")

matching\_columns\_y = [col for col in df.columns if search\_string\_y in col]

if matching\_columns\_y:

for j, col in enumerate(matching\_columns\_y, 1):

print(f"{j}. {col}")

if len(matching\_columns\_y) == 1:

selected\_column\_y = matching\_columns\_y[0]

print(f"Selected column for Y-axis column {i+1}: {selected\_column\_y}")

else:

while True:

try:

selection\_y = int(input(f"Enter the serial number of the column for Y-axis column {i+1}: "))

if selection\_y >= 1 and selection\_y <= len(matching\_columns\_y):

selected\_column\_y = matching\_columns\_y[selection\_y - 1]

print(f"Selected column for Y-axis column {i+1}: {selected\_column\_y}")

break

else:

print("Invalid selection. Please enter a valid serial number.")

except ValueError:

print("Invalid input. Please enter a valid serial number.")

else:

print(f"No matching columns found for Y-axis column {i+1}.")

y\_columns.append(selected\_column\_y)

y\_labels.append(f"Y-axis label {i+1}")

# Prompt the user to enter the range for the x-axis ["dd-mm-yyyy HH:MM"]

x\_min = input("Enter the minimum value for the x-axis (dd-mm-yyyy HH:MM): ")

x\_max = input("Enter the maximum value for the x-axis (dd-mm-yyyy HH:MM): ")

# Convert x-axis values to datetime objects with custom format

date\_format = "%d-%m-%Y %H:%M"

df[selected\_column\_x] = pd.to\_datetime(df[selected\_column\_x], format=date\_format)

# Filter the dataframe based on the selected range of x-axis values

filtered\_df = df[(df[selected\_column\_x] >= datetime.strptime(x\_min, date\_format)) &

(df[selected\_column\_x] <= datetime.strptime(x\_max, date\_format))]

# Create the figure and axes

fig, ax1 = plt.subplots()

# Plot y values with the corresponding y-axis

lines = []

line\_labels = []

for i, y\_column in enumerate(y\_columns):

if i == 0:

line, = ax1.plot(filtered\_df[selected\_column\_x], filtered\_df[y\_column], color=colors[i])

line\_labels.append(y\_labels[i])

lines.append(line)

ax1.set\_ylabel(y\_labels[i], color=colors[i])

ax1.tick\_params(axis='y', labelcolor=colors[i])

else:

ax = ax1.twinx()

line, = ax.plot(filtered\_df[selected\_column\_x], filtered\_df[y\_column], color=colors[i])

line\_labels.append(y\_labels[i])

lines.append(line)

ax.spines['right'].set\_position(('outward', 60 + (i - 1) \* 40)) # Adjust position of additional y-axes

ax.set\_ylabel(y\_labels[i], color=colors[i])

# Set the limits of the x-axis

ax1.set\_xlim(datetime.strptime(x\_min, date\_format), datetime.strptime(x\_max, date\_format))

# Set the x-ticks to the four equal partitions

x\_ticks = pd.date\_range(start=datetime.strptime(x\_min, date\_format), end=datetime.strptime(x\_max, date\_format), periods=5)

ax1.set\_xticks(x\_ticks)

# Rotate the x-axis tick labels by 90 degrees

ax1.set\_xticklabels(x\_ticks.strftime(date\_format).tolist(), rotation=90)

# Adjust the spacing between subplots

fig.tight\_layout()

# Set the y-axis labels for all subplots

if len(y\_labels) > 0:

plt.legend(lines, line\_labels, loc='upper center', bbox\_to\_anchor=(0.5, 1.15), ncol=len(y\_labels))

# Set the title of the plot

plt.xlabel(selected\_column\_x)

plt.show()

**OUTPUT**

