МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №8

з дисципліни

«Спеціалізовані мови приграмування»

на тему

«Візуалізація та обробка даних за допомогою спеціалізованих бібліотек Python»

Виконав:

ст. гр. РІ-32

Сергій БИХНЮК

Прийняв:

Сергій Щербак

Львів – 2024

**Мета роботи:**

Розробка додатка для візуалізації CSV-наборів даних за допомогою Matplotlib та базових принципів ООП (наслідування, інкапсуляція, поліморфізм).

**Завдання:**

**Завдання 1**: Вибір CSV-набору даних

Оберіть CSV-набір даних, який ви хочете візуалізувати. Переконайтеся, що він містить відповідні дані для створення змістовних візуалізацій.

**Завдання 2:** Завантаження даних з CSV

Напишіть код для завантаження даних з CSV-файлу в ваш додаток Python. Використовуйте бібліотеки, такі як Pandas, для спрощення обробки даних.

**Завдання 3:** Дослідження даних

Визначте екстремальні значення по стовцям

**Завдання 4:** Вибір типів візуалізацій

Визначте, які типи візуалізацій підходять для представлення вибраних наборів даних. Зазвичай це може бути лінійні графіки, стовпчикові діаграми, діаграми розсіювання, гістограми та секторні діаграми.

**Завдання 5:** Підготовка даних

Попередньо обробіть набір даних за необхідністю для візуалізації. Це може включати виправлення даних, фільтрацію, агрегацію або трансформацію.

**Завдання 6:** Базова візуалізація

Створіть базову візуалізацію набору даних, щоб переконатися, що ви можете відображати дані правильно за допомогою Matplotlib. Розпочніть з простої діаграми для візуалізації однієї змінної.

**Завдання 7:** Розширені візуалізації

Реалізуйте більш складні візуалізації, виходячи з характеристик набору. Поекспериментуйте з різними функціями Matplotlib та налаштуваннями.

**Завдання 8:** Декілька піддіаграм

Навчіться створювати кілька піддіаграм в межах одного малюнка для відображення декількох візуалізацій поруч для кращого порівняння.

**Завдання 9:** Експорт і обмін

Реалізуйте функціональність для експорту візуалізацій як зображень (наприклад, PNG, SVG) або інтерактивних веб-додатків (наприклад, HTML)

**Код програми:**

# Classes/DataProcessor.py

import pandas as pd

class DataProcessor:

def \_\_init\_\_(self, file\_path):

self.data = pd.read\_csv(file\_path)

def preprocess\_data(self):

# Basic data cleaning and handling of NaN values

self.data.dropna(inplace=True)

def identify\_extremes(self, column):

max\_value = self.data[column].max()

min\_value = self.data[column].min()

return {'max': max\_value, 'min': min\_value}

def get\_data(self):

return self.data

# Classes/Visualizer.py

import plotly.express as px

class Visualizer:

def \_\_init\_\_(self, data):

self.data = data

def plot\_basic\_chart(self, column):

fig = px.line(self.data, y=column, title=f'{column} Over Time')

fig.show()

def save\_interactive\_chart(self, column, file\_name):

fig = px.line(self.data, y=column, title=f'{column} Over Time')

# Save as HTML if PNG export fails

try:

fig.write\_image(file\_name)

except ValueError:

print("PNG export failed, saving as HTML instead.")

html\_file\_name = file\_name.replace('.png', '.html')

fig.write\_html(html\_file\_name)

# Constants/config.py

CSV\_FILE\_PATH = '/home/vladusald/PycharmProjects/СМП Л1/Data/Lab8/Assets/your\_file.csv'

IMAGE\_FILE\_PATH = '/home/vladusald/PycharmProjects/СМП Л1/Data/Lab8/Assets/sales\_amount\_interactive.png'

# Functions/data\_loading.py

import pandas as pd

def load\_data(file\_path):

return pd.read\_csv(file\_path)

# Functions/data\_preprocessing.py

def clean\_data(df):

# Example: Remove rows with any NaN values

df.dropna(inplace=True)

return df

# Functions/visualization.py

import matplotlib.pyplot as plt

import plotly.express as px

def plot\_histogram(data, column):

plt.hist(data[column], bins=10)

plt.title(f'{column} Distribution')

plt.xlabel(column)

plt.ylabel('Frequency')

plt.show()

def plot\_interactive\_line(data, column, file\_name):

fig = px.line(data, y=column, title=f'{column} Over Time')

fig.write\_image(file\_name)

# Interfaces/IVisualizer.py

from abc import ABC, abstractmethod

class IVisualizer(ABC):

@abstractmethod

def plot\_basic\_chart(self, column):

pass

@abstractmethod

def save\_interactive\_chart(self, column, file\_name):

pass

# main.py

from lab8.Classes.DataProcessor import DataProcessor

from lab8.Classes.Visualizer import Visualizer

from lab8.Constants.config import CSV\_FILE\_PATH, IMAGE\_FILE\_PATH

def main8():

# Load and preprocess data

data\_processor = DataProcessor(CSV\_FILE\_PATH)

data\_processor.preprocess\_data()

# Identify extremes for a column

extremes = data\_processor.identify\_extremes('Sales\_Amount')

print(f"Extreme values for Sales\_Amount: {extremes}")

# Get processed data

data = data\_processor.get\_data()

# Create visualizations

visualizer = Visualizer(data)

visualizer.plot\_basic\_chart('Sales\_Amount')

visualizer.save\_interactive\_chart('Sales\_Amount', IMAGE\_FILE\_PATH)

import pandas as pd

import numpy as np

# Generate a sample dataset

data = pd.DataFrame({

'Date': pd.date\_range(start='2023-01-01', periods=10),

'Sales\_Amount': np.random.randint(200, 500, 10),

'Advertising\_Spend': np.random.randint(100, 300, 10),

'Product': np.random.choice(['Product A', 'Product B', 'Product C'], 10)

})

# Save as CSV

data.to\_csv('your\_file.csv', index=False)

print("Sample dataset saved as your\_file.csv")

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from dash import Dash, dcc, html

# 1. Load Data from a CSV File

# Replace 'your\_file.csv' with the path to your CSV file.

file\_path = '../Assets/your\_file.csv'

data = pd.read\_csv(file\_path)

# Assuming the CSV file has columns 'Date', 'Sales\_Amount', 'Advertising\_Spend', and 'Product'.

data['Date'] = pd.to\_datetime(data['Date'])

# 2. Identify Extreme Values by Column

print("Extreme Values by Column:")

for column in data.select\_dtypes(include=['float64', 'int64']).columns:

print(f"{column} - Min: {data[column].min()}, Max: {data[column].max()}")

# 3. Preprocessing the Data

# Filtering data for the example, e.g., removing rows with null values.

data.dropna(inplace=True)

# Aggregate sales by product category for plotting purposes.

product\_sales = data.groupby('Product')['Sales\_Amount'].sum().sort\_values()

# 4. Create Basic Visualization for a Single Variable

plt.figure(figsize=(10, 6))

plt.plot(data['Date'], data['Sales\_Amount'], marker='o', color='skyblue')

plt.title('Sales Amount Over Time')

plt.xlabel('Date')

plt.ylabel('Sales Amount')

plt.xticks(rotation=45)

plt.grid()

plt.savefig("sales\_amount\_basic.png", format='png')

plt.show()

# 5. Multiple Sub-Charts in a Single Figure

fig, axs = plt.subplots(2, 2, figsize=(14, 10))

fig.suptitle("Multiple Data Visualizations")

# Subplot 1: Line Plot

axs[0, 0].plot(data['Date'], data['Sales\_Amount'], color='blue', marker='o')

axs[0, 0].set\_title("Sales Amount Over Time")

axs[0, 0].set\_xlabel("Date")

axs[0, 0].set\_ylabel("Sales Amount")

axs[0, 0].tick\_params(axis='x', rotation=45)

# Subplot 2: Bar Plot

axs[0, 1].bar(product\_sales.index, product\_sales.values, color='green')

axs[0, 1].set\_title("Total Sales by Product")

axs[0, 1].set\_xlabel("Product")

axs[0, 1].set\_ylabel("Sales Amount")

axs[0, 1].tick\_params(axis='x', rotation=45)

# Subplot 3: Histogram

axs[1, 0].hist(data['Sales\_Amount'], bins=20, color='purple', edgecolor='black')

axs[1, 0].set\_title("Distribution of Sales Amount")

axs[1, 0].set\_xlabel("Sales Amount")

axs[1, 0].set\_ylabel("Frequency")

# Subplot 4: Scatter Plot with Trend Line

sns.regplot(x='Advertising\_Spend', y='Sales\_Amount', data=data, ax=axs[1, 1], scatter\_kws={'alpha':0.6}, line\_kws={"color":"red"})

axs[1, 1].set\_title("Sales Amount vs. Advertising Spend")

axs[1, 1].set\_xlabel("Advertising Spend")

axs[1, 1].set\_ylabel("Sales Amount")

# Save the multiple subplots figure

plt.tight\_layout(rect=[0, 0, 1, 0.95])

plt.savefig("multiple\_data\_visualizations.png", format='png')

plt.show()

# 6. Interactive HTML Visualization with Plotly

fig = px.line(data, x='Date', y='Sales\_Amount', title='Sales Amount Over Time')

fig.write\_html("sales\_amount\_interactive.html")

fig.show()

# Export as an interactive image (optional, requires Kaleido)

fig.write\_image("sales\_amount\_interactive.png")

# 7. Building an Interactive Web App with Dash

app = Dash(\_\_name\_\_)

app.layout = html.Div([

html.H1("Sales Amount Over Time"),

dcc.Graph(

id="line-chart",

figure=px.line(data, x='Date', y='Sales\_Amount', title='Sales Amount Over Time')

),

dcc.Graph(

id="scatter-chart",

figure=px.scatter(data, x='Advertising\_Spend', y='Sales\_Amount', trendline='ols', title='Sales vs. Advertising Spend')

)

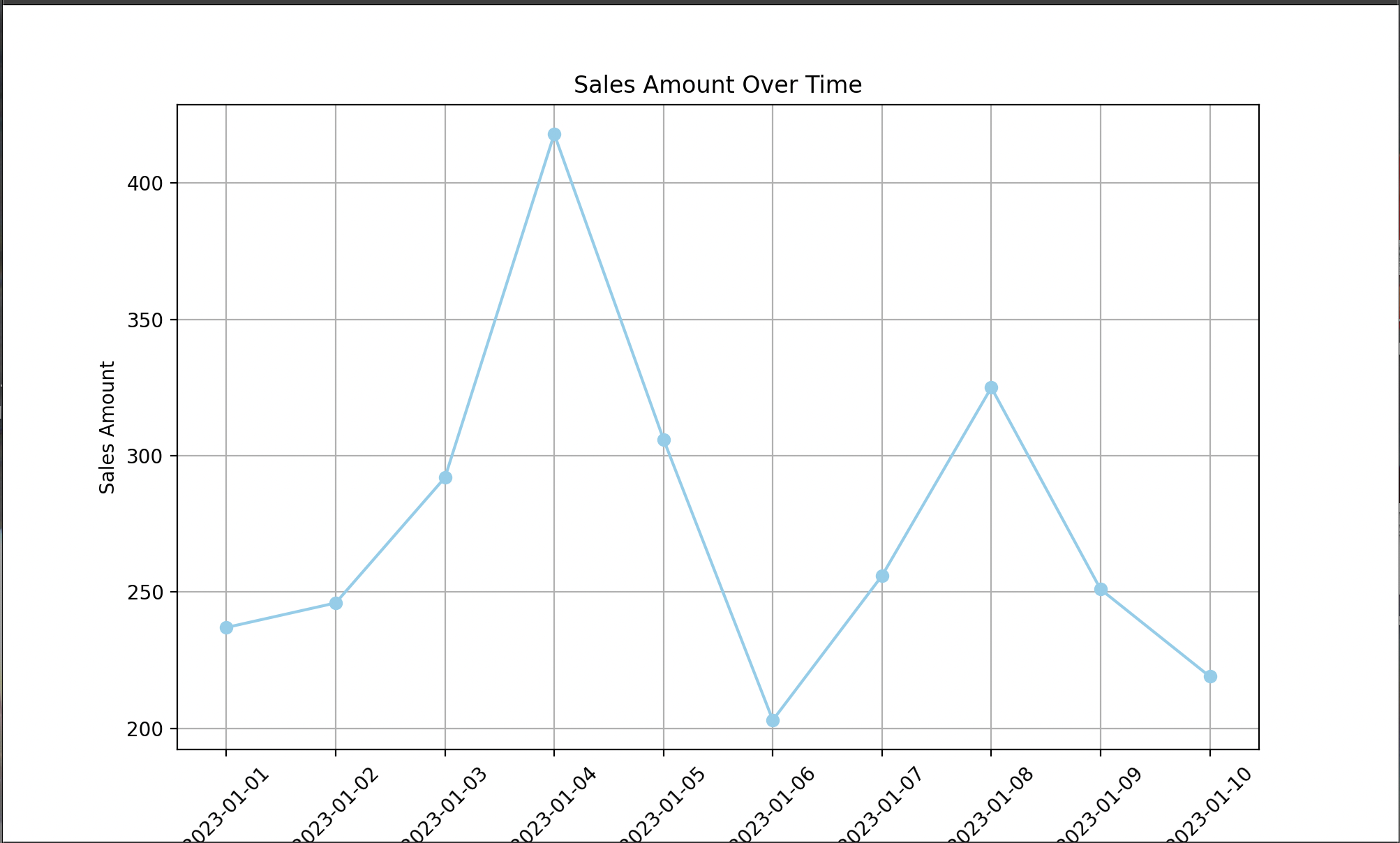
])

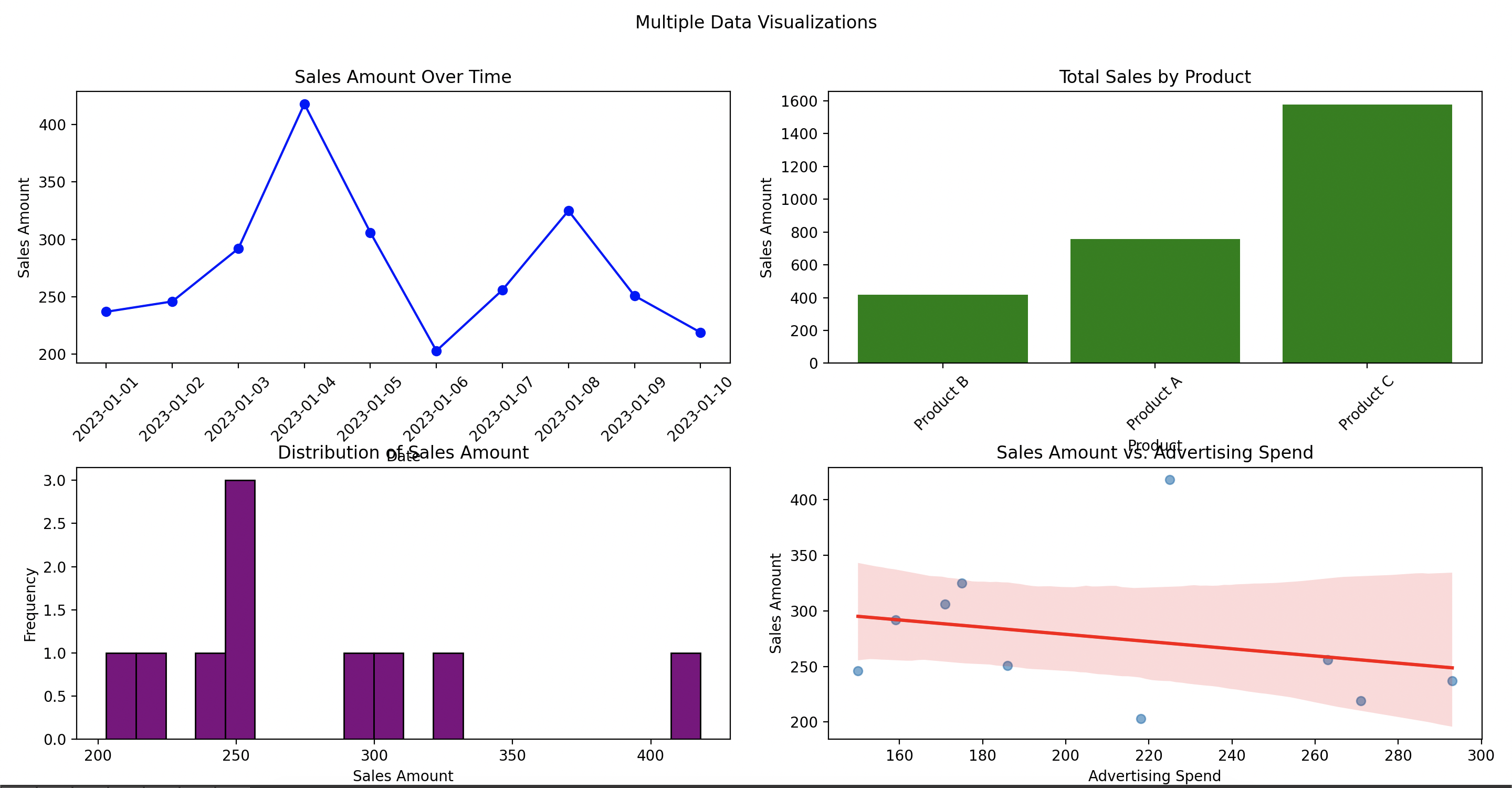
#Run the Dash app (for web app use; comment out if not testing the app)

# if \_\_name\_\_ == '\_\_main\_\_':

# app.run\_server(debug=True)

**Виконання:**







**Висновок:**

Виконавши ці завдання, я створив багатофункціональний додаток для візуалізації CSV-наборів даних за допомогою Matplotlib. Цей проект покращив мої навички візуалізації даних, дозволяючи досліджувати результати з різноманітними наборами даних.