2. Implement programs for visualizing time series data.

EX.N0 : 2	**Implement programs for visualization time series data.**
DATE : 25/01/2025	

AIM:

Implement programs for visualizing time series data.

PROGRAM:

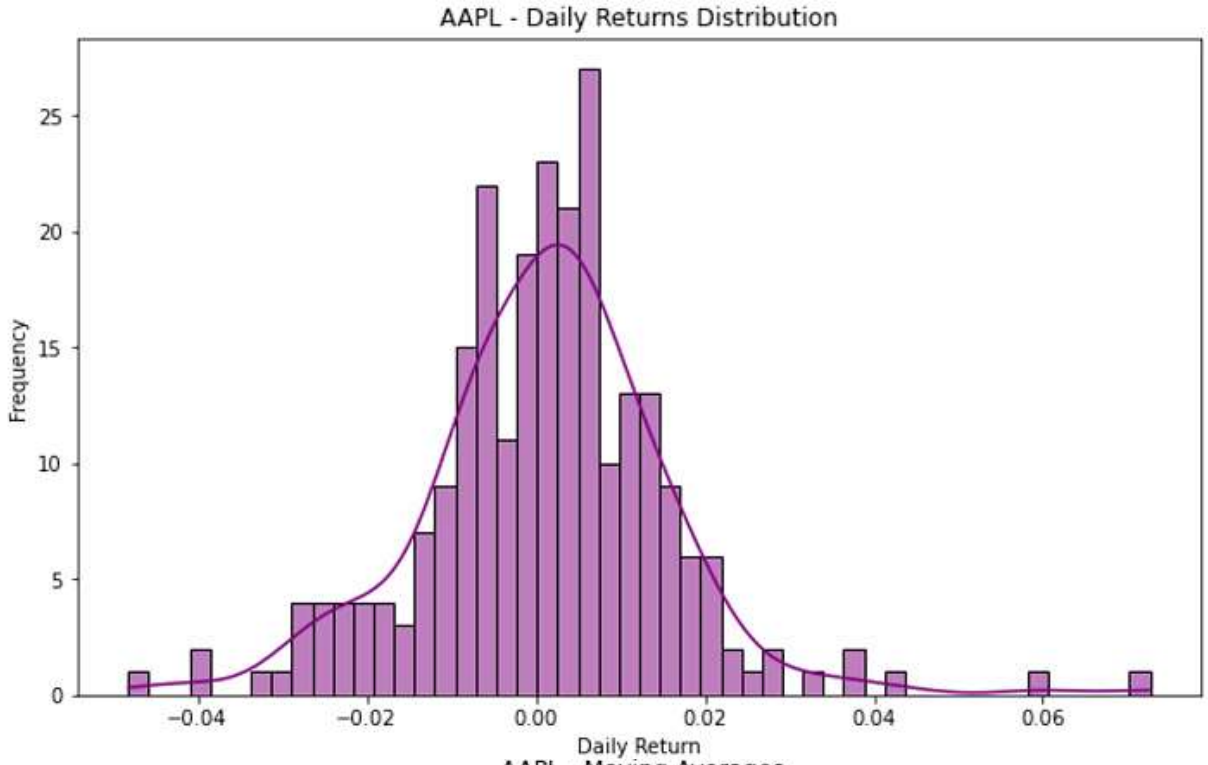
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

def load_data(filepath):

    try:

        data = pd.read_csv(filepath, parse_dates=True, index_col='Date')

        print("Dataset loaded successfully.")

        return data

    except Exception as e:

        print(f"Error loading dataset: {e}")

        return None

def clean_data(data):

    print("Initial dataset shape:", data.shape)

    data = data.drop_duplicates()

    data = data.fillna(method='ffill')  # Forward fill missing values

    data = data.fillna(method='bfill')  # Backward fill for remaining missing values

    data = data.dropna()
```

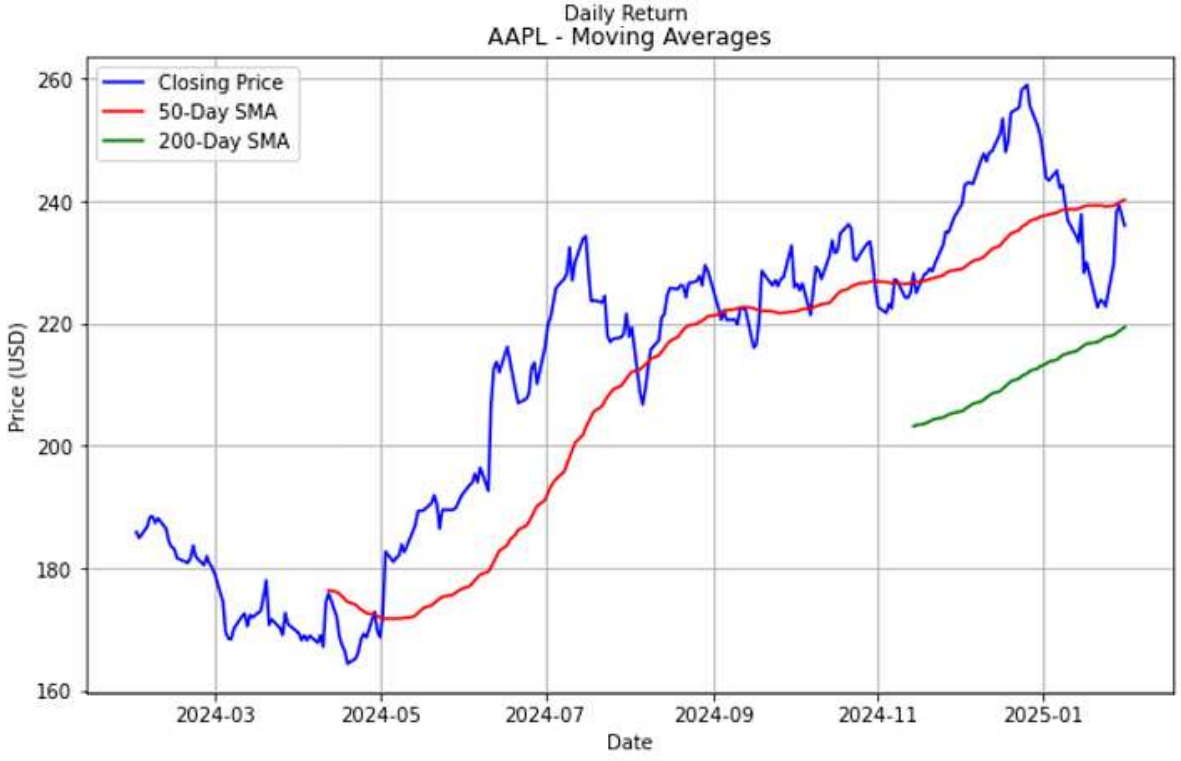
```
    print("Dataset shape after cleaning:", data.shape)
    return data
def preprocess_time_series(data):
    print("Index type:", type(data.index))
    if not isinstance(data.index, pd.DatetimeIndex):
        data.index = pd.to_datetime(data.index)
    data = data.sort_index()
    return data
def feature_engineering(data):
    data['SMA_7'] = data['Close'].rolling(window=7).mean()  # 7-day Simple Moving Average
    data['SMA_30'] = data['Close'].rolling(window=30).mean()  # 30-day Simple Moving Average
    data['Lag_1'] = data['Close'].shift(1)  # Previous day's price
    data['Lag_7'] = data['Close'].shift(7)  # Price a week ago
    data = data.dropna()
    return data
def visualize_data(data):
    plt.figure(figsize=(12, 6))
    plt.plot(data['Close'], label='Gold Price')
    plt.plot(data['SMA_7'], label='7-Day SMA', linestyle='--')
    plt.plot(data['SMA_30'], label='30-Day SMA', linestyle='--')
    plt.title('Gold Price and Moving Averages')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
```

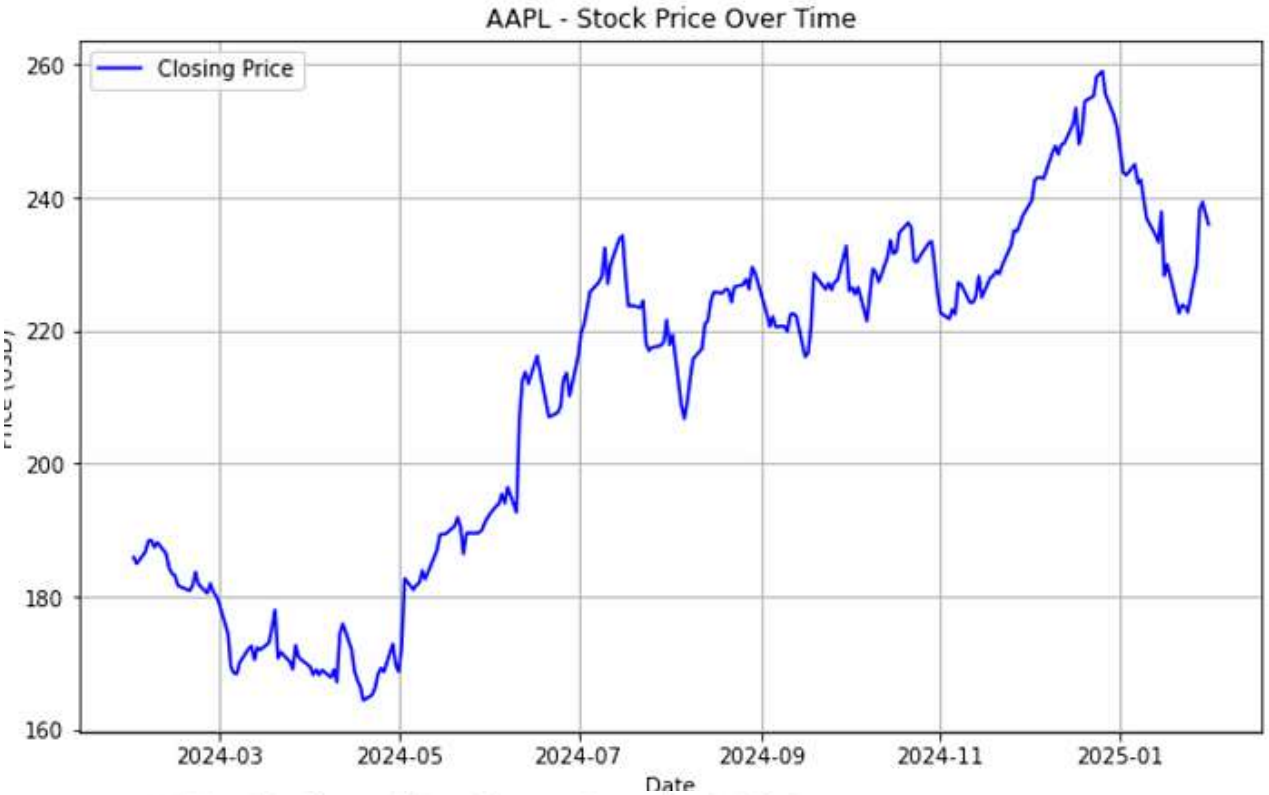
```
    plt.grid()
    plt.show()
def visualize_time_series(data):
    plt.figure(figsize=(14, 7))
    plt.subplot(2, 1, 1)
    plt.plot(data['Close'], label='Gold Price', color='blue')
    plt.title('Gold Price Over Time')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
    plt.grid()
    plt.subplot(2, 1, 2)
    plt.hist(data['Close'], bins=30, color='gold', edgecolor='black')
    plt.title('Distribution of Gold Prices')
    plt.xlabel('Price')
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
def main():
    filepath = "C:\\Users\\jaya karthick\\Downloads\\archive (1) (1)\\FINAL_USO.csv"
     data = load_data(filepath)
    if data is None:
        return
    data = clean_data(data)
    data = preprocess_time_series(data)
```

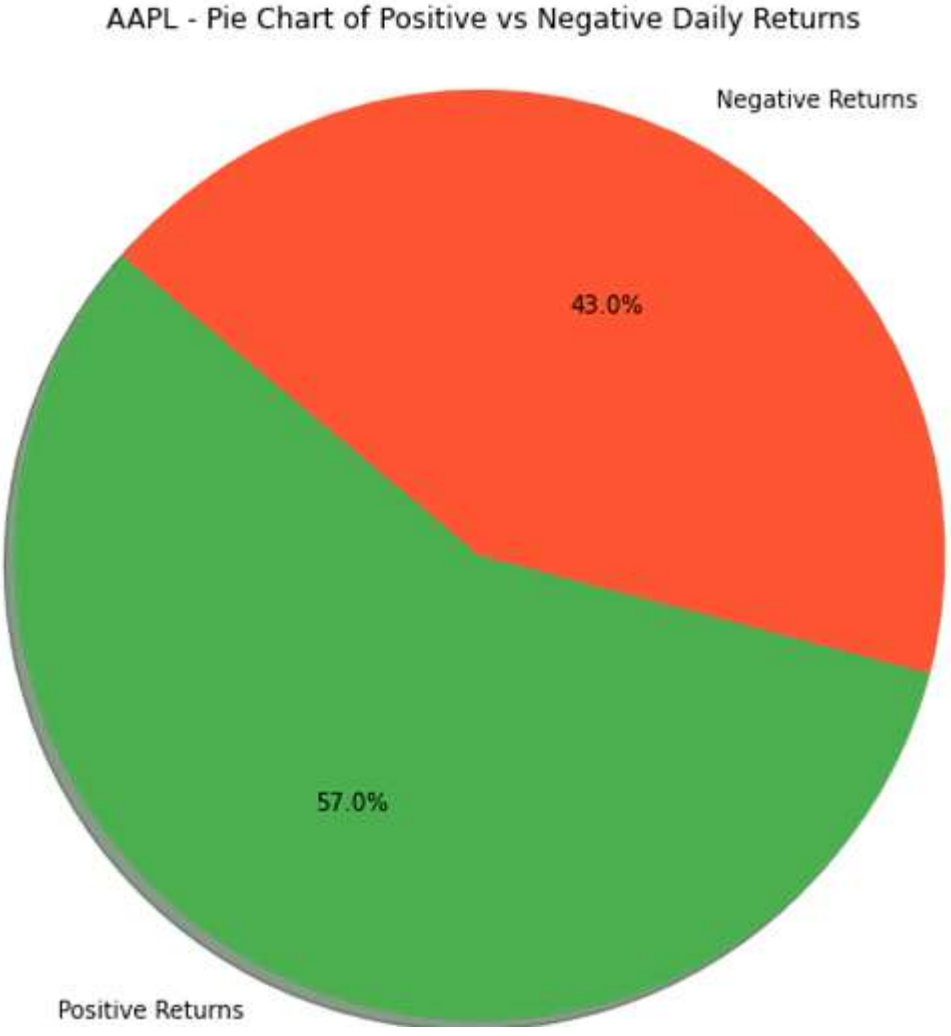
```
    data = feature_engineering(data)
    visualize_data(data)
    visualize_time_series(data)
    print("Processed dataset preview:\n", data.head())
if __name__ == "__main__":
    main()
```

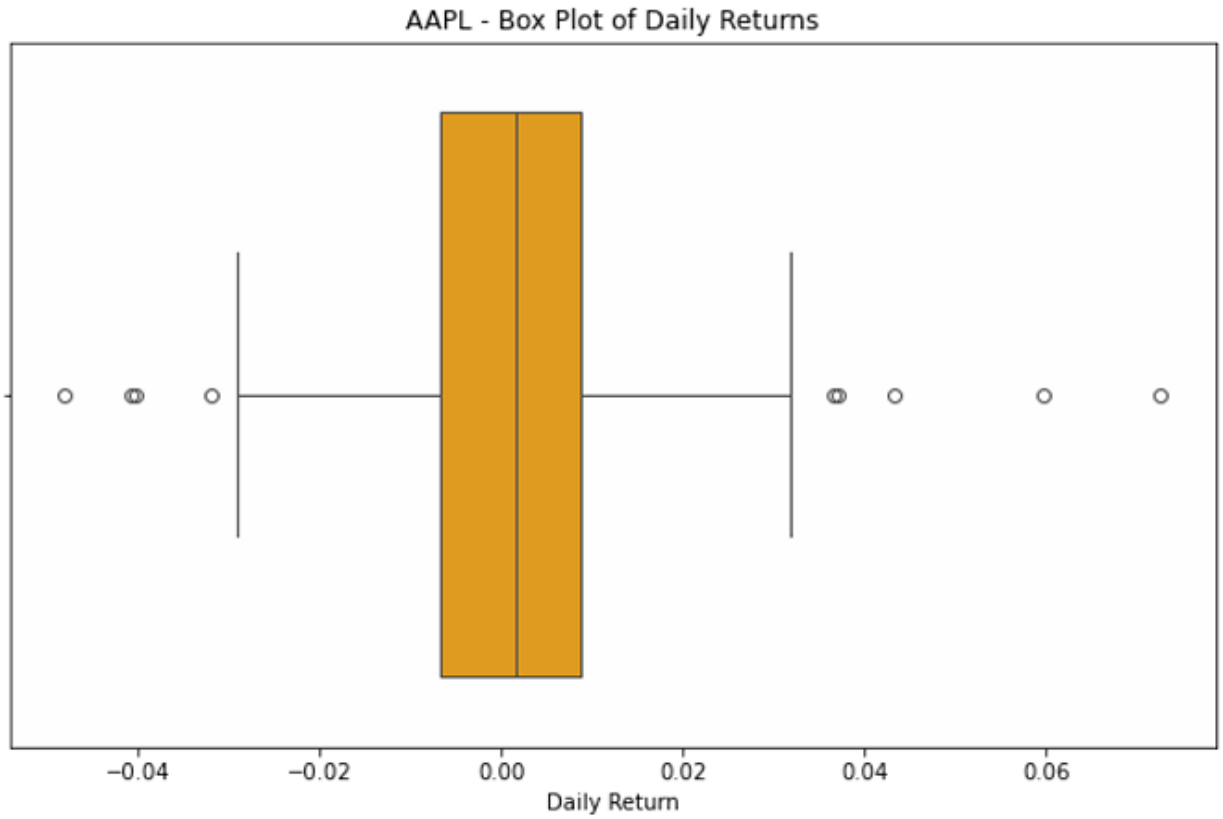
OUTPUT:


AAPL - Daily Returns Distribution
Frequency
Daily Return
25
20
15
10
5
0
−0.04
−0.02
0.00
0.02
0.04
0.06


AAPL - Moving Averages
Closing Price
50-Day SMA
200-Day SMA
Price (USD)
Date
260
240
220
200
180
160
2024-03
2024-05
2024-07
2024-09
2024-11
2025-01


AAPL - Stock Price Over Time
Closing Price
260
240
220
200
180
160
2024-03
2024-05
2024-07
2024-09
2024-11
2025-01
Date


AAPL - Pie Chart of Positive vs Negative Daily Returns
Negative Returns
43.0%
57.0%
Positive Returns



RESULT:

Thus, the program for Implement programs for visualizing time series data is executed successfully.