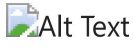


In [40]:
```
import pandas as pd # Reading the csv data here
df = pd.read_csv(r'D:\Information Technology\Goldman sachs\Goldman Sachs stock data.csv')
df.head() # printing it to see it down there
```

Out[40]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1999-05-04	76.0000	77.250	70.0000	70.3750	53.576797	22320900
1	1999-05-05	69.8750	69.875	66.2500	69.1250	52.625153	7565700
2	1999-05-06	68.0000	69.375	67.0625	67.9375	51.721100	2905700
3	1999-05-07	67.9375	74.875	66.7500	74.1250	56.431648	4862300
4	1999-05-10	73.3750	73.500	70.2500	70.6875	53.814709	2589400

In [42]:
```
df.info() # checking the information about the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5762 entries, 0 to 5761
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       5762 non-null   object 
 1   Open       5762 non-null   float64
 2   High       5762 non-null   float64
 3   Low        5762 non-null   float64
 4   Close      5762 non-null   float64
 5   Adj Close  5762 non-null   float64
 6   Volume     5762 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 315.2+ KB
```

In [44]:
```
df.isnull().sum() # Checking the null values of the data set
```

Out[44]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [46]:
```
df.describe() # describing it in term of mean, stander deviation etc.
```

Out[46]:

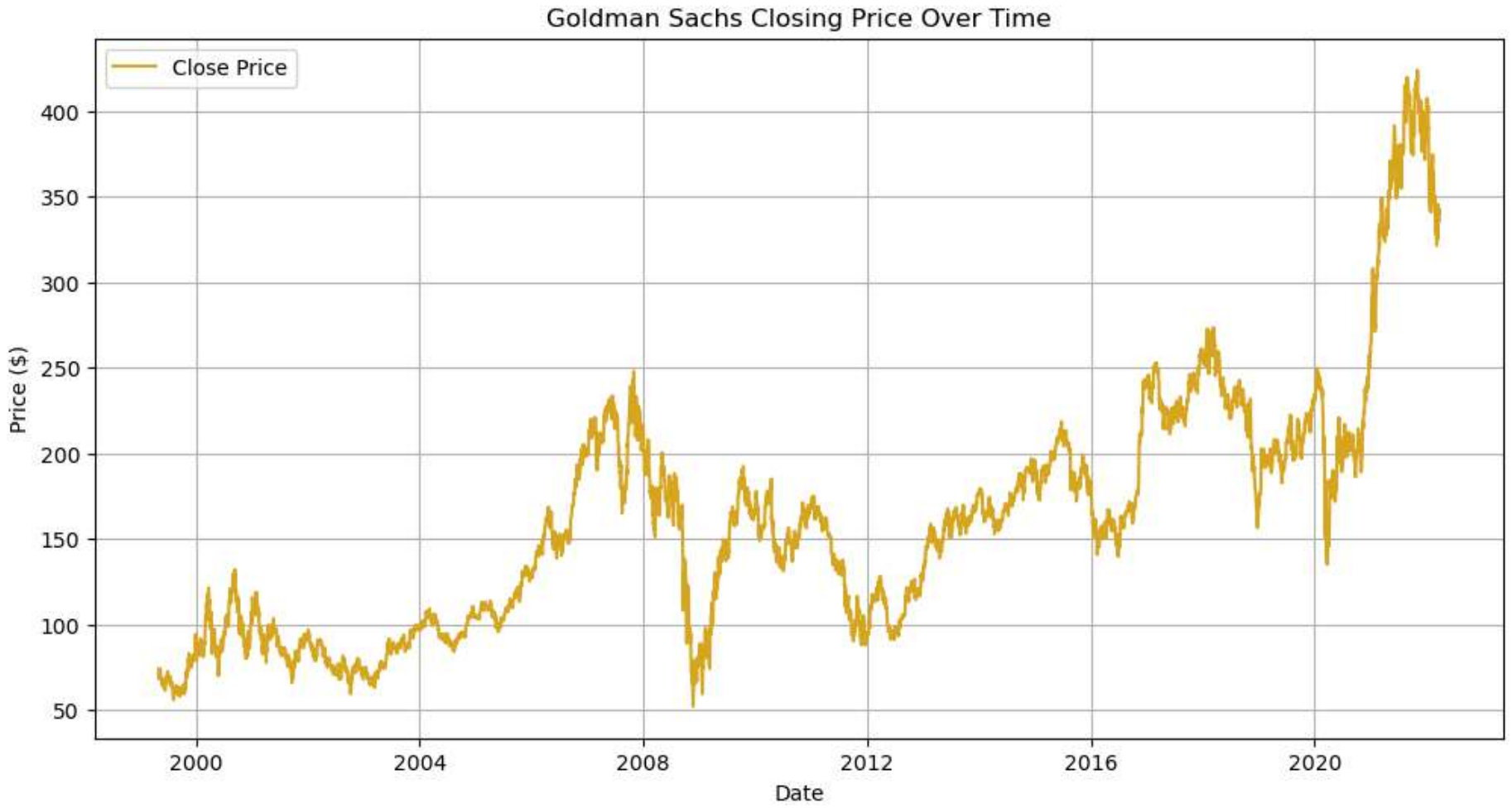
	Open	High	Low	Close	Adj Close	Volume
count	5762.000000	5762.000000	5762.000000	5762.000000	5762.000000	5.762000e+03
mean	160.717986	162.655643	158.761075	160.732162	140.676221	5.264446e+06
std	71.225751	71.755243	70.613907	71.177093	73.187668	6.077306e+06
min	54.000000	54.540001	47.410000	52.000000	42.490295	1.076000e+05
25%	100.552502	102.277499	99.162501	100.705002	81.146532	2.487625e+06
50%	157.605004	159.699997	156.004998	157.750000	134.576713	3.553000e+06
75%	199.507496	201.847496	197.419995	199.502499	177.186802	5.578525e+06
max	423.119995	426.160004	413.760010	423.850006	419.154236	1.145907e+08

In [50]:
```
# conveting the date column datatype
df['Date'] = pd.to_datetime(df['Date']) # done converting it
```

Time series analysis

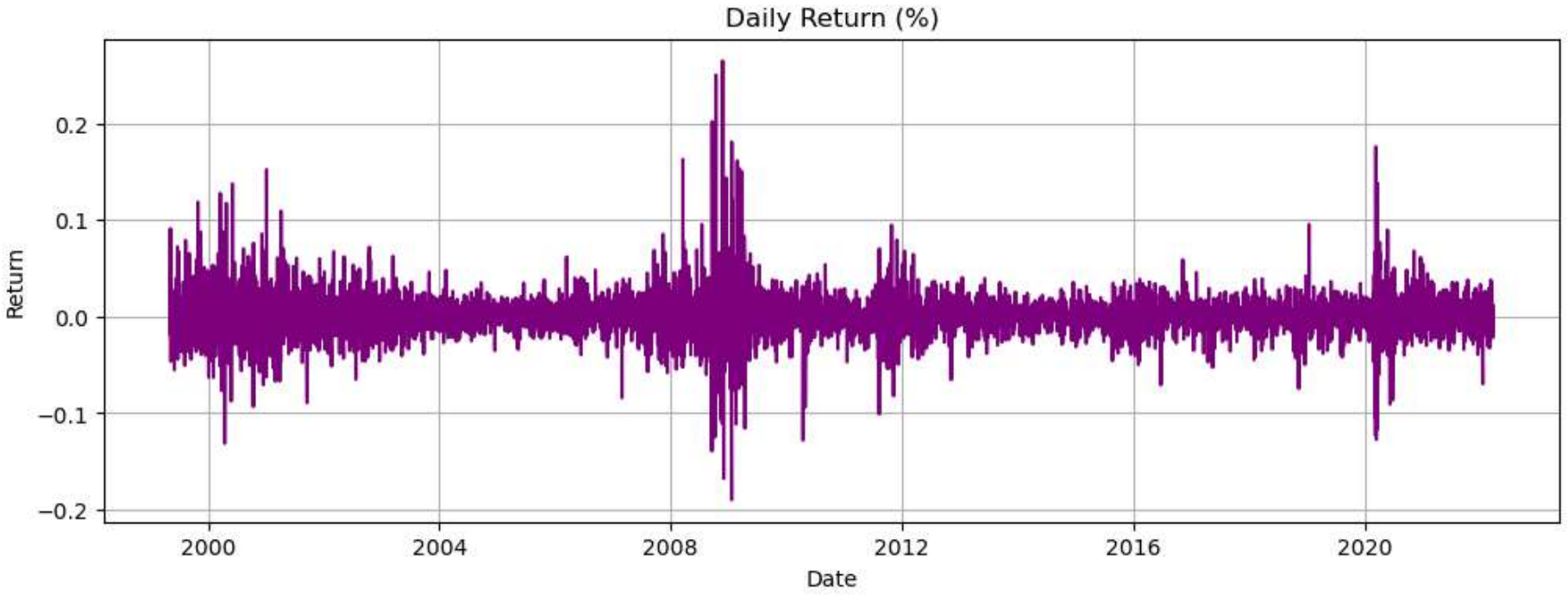
In [56]:
```
import matplotlib.pyplot as plt # importing 'matplotlib'
df = df.sort_values('Date') # sort datas
df.set_index('Date', inplace=True) # set date as index
```

In [58]:
```
# Plot closing price over time

plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Close Price', color='goldenrod')
plt.title('Goldman Sachs Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()
plt.grid(True)
plt.show()
```



In [60]:
```
# Calculate daily returns

df['Daily Return'] = df['Adj Close'].pct_change()

# Plot Daily Returns
plt.figure(figsize=(12, 4))
plt.plot(df['Daily Return'], color='purple')
plt.title('Daily Return (%)')
plt.xlabel('Date')
plt.ylabel('Return')
```

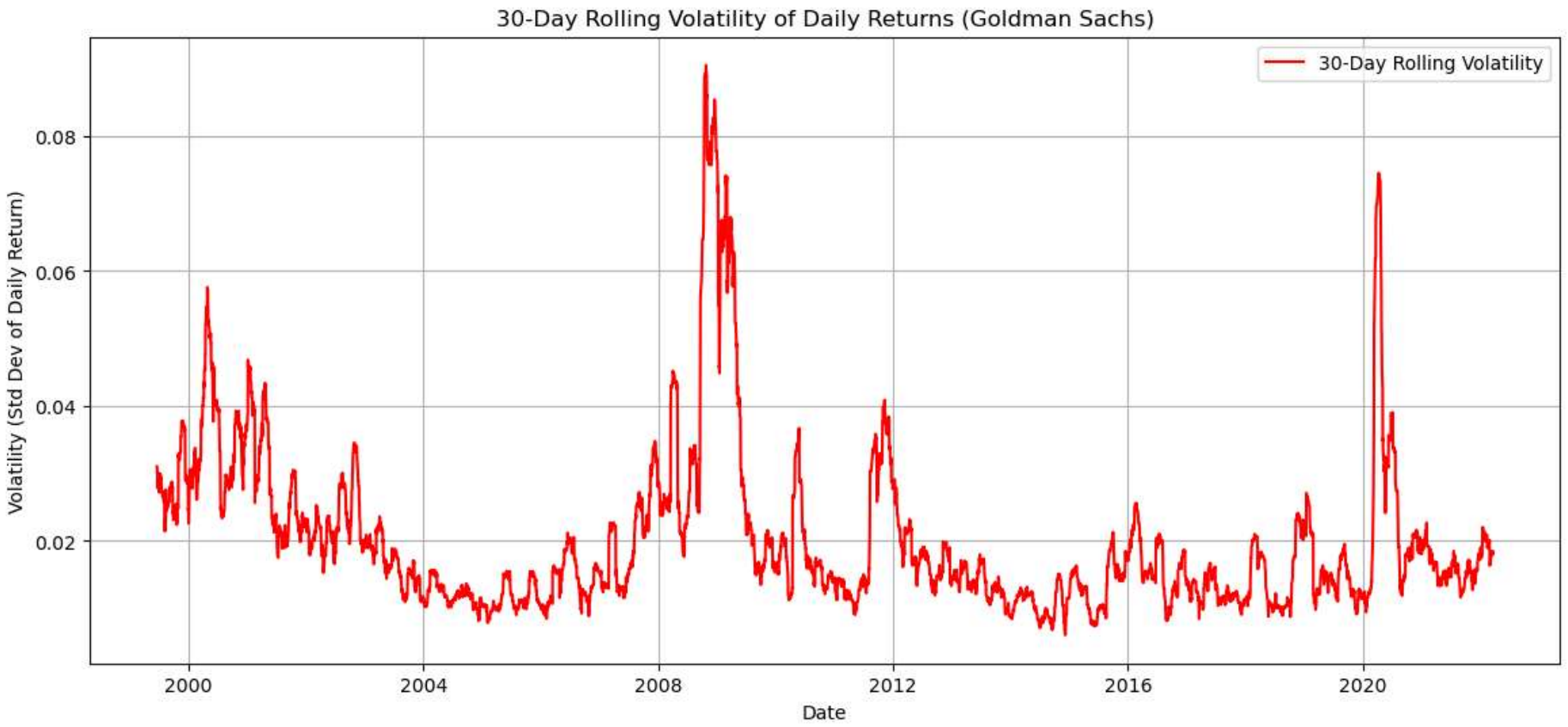
```
plt.grid(True)
plt.show()
```



Volatility Analysis

In [67]:
```
# 30-day rolling volatility
df['30D Volatility'] = df['Daily Return'].rolling(window=30).std() # std dev of 30days rolling

import matplotlib.pyplot as plt # ploting it here as a line plot

plt.figure(figsize=(14, 6))
plt.plot(df['30D Volatility'], color='red', label='30-Day Rolling Volatility')
plt.title('30-Day Rolling Volatility of Daily Returns (Goldman Sachs)')
plt.xlabel('Date')
plt.ylabel('Volatility (Std Dev of Daily Return)')
plt.legend()
```

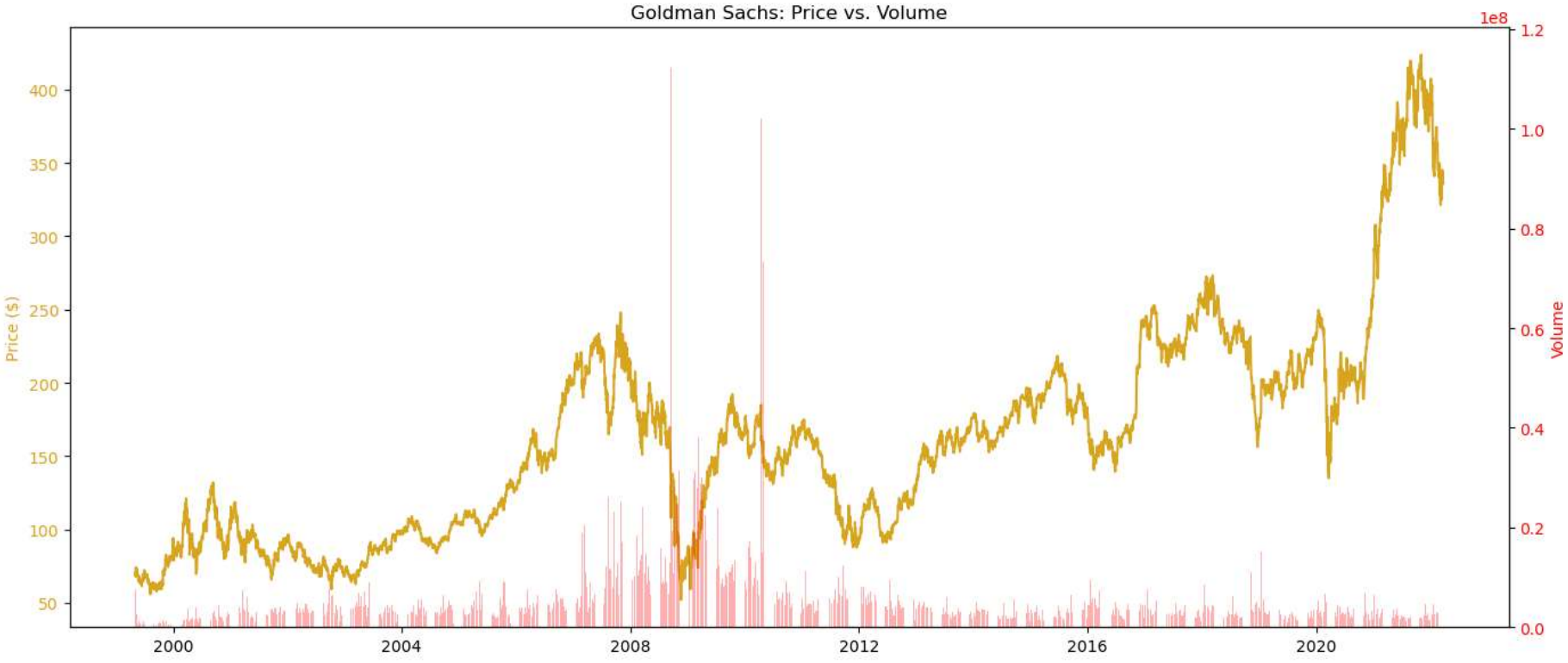
```
plt.grid(True)
plt.show()
```



Volume analysis - stocks traded on that perticular day

In [74]:
```
import matplotlib.pyplot as plt

fig, ax1 = plt.subplots(figsize=(14, 6))

# Plot Closing Price
ax1.plot(df.index, df['Close'], color='goldenrod', label='Closing Price')
ax1.set_ylabel('Price ($)', color='goldenrod')
```

```
ax1.tick_params(axis='y', labelcolor='goldenrod')
ax1.set_title('Goldman Sachs: Price vs. Volume')

# Create a second y-axis for volume
ax2 = ax1.twinx()
ax2.bar(df.index, df['Volume'], alpha=0.3, color='red', label='Volume')
ax2.set_ylabel('Volume', color='red')
ax2.tick_params(axis='y', labelcolor='red')

fig.tight_layout()
plt.show()
```



Candlestick Chart Using plotly

In [81]:
```
import plotly.graph_objects as go

# Reset index in case Date is the index
df_reset = df.reset_index()

fig = go.Figure(data=[go.Candlestick(
    x=df_reset['Date'],
    open=df_reset['Open'],
    high=df_reset['High'],
    low=df_reset['Low'],
    close=df_reset['Close'],
    increasing_line_color='green',
    decreasing_line_color='red'
)])

fig.update_layout(
    title='Goldman Sachs Candlestick Chart',
    xaxis_title='Date',
    yaxis_title='Price ($)',
    xaxis_rangeslider_visible=False,
    template='plotly_dark'
)

fig.show()
```

In [83]:
```
pip install mplfinance
```

```
Collecting mplfinance
  Downloading mplfinance-0.12.10b0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: matplotlib in c:\users\dell\anaconda3\lib\site-packages (from mplfinance) (3.8.4)
Requirement already satisfied: pandas in c:\users\dell\anaconda3\lib\site-packages (from mplfinance) (2.2.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (1.
2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (4.
51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (1.
4.4)
Requirement already satisfied: numpy>=1.21 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (23.
2)
Requirement already satisfied: pillow>=8 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance) (3.
0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib->mplfinance)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\site-packages (from pandas->mplfinance) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\dell\anaconda3\lib\site-packages (from pandas->mplfinance) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->mp
lfinance) (1.16.0)
Downloading mplfinance-0.12.10b0-py3-none-any.whl (75 kB)
   ---------------------------------------- 0.0/75.0 kB ? eta -:--:--
   --------------- ---------------------- 30.7/75.0 kB 1.3 MB/s eta 0:00:01
   -------------------- ------------------ 41.0/75.0 kB 388.9 kB/s eta 0:00:01
   ---------------------------------------- 75.0/75.0 kB 588.9 kB/s eta 0:00:00
Installing collected packages: mplfinance
Successfully installed mplfinance-0.12.10b0
Note: you may need to restart the kernel to use updated packages.
```

In [97]:
```
# Filter last 100 days
df_recent = df.tail(200)

mpf.plot(
    df_recent,
    type='candle',
    volume=True,
    style='yahoo',
    title='Goldman Sachs - Last 100 Days',
```

```
    mav=(20, 50),
    figsize=(14, 6)
)
```

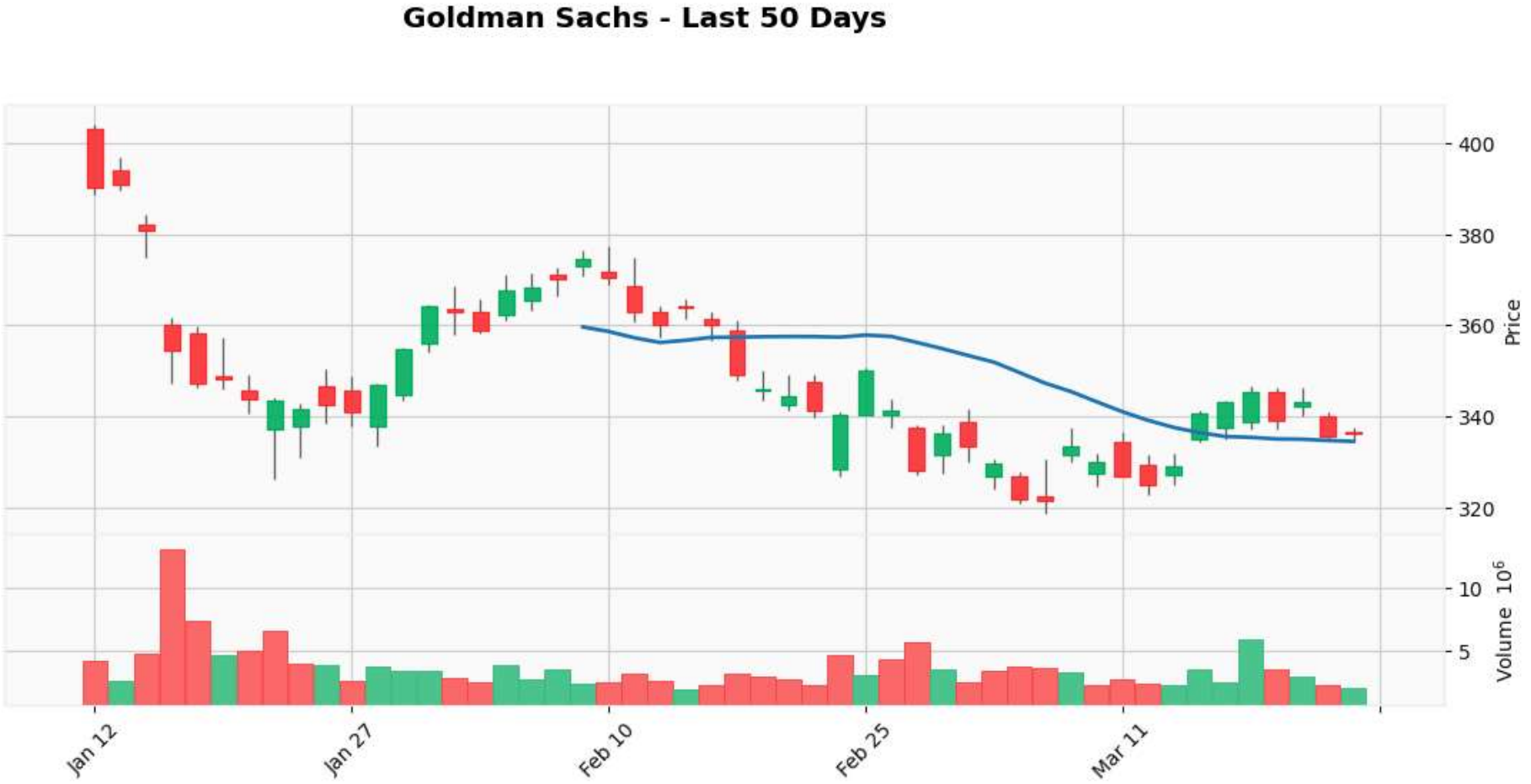
In [103…

```
# Filter last 100 days
df_recent = df.tail(100)

mpf.plot(
    df_recent,
    type='candle',
```

```
    volume=True,
    style='yahoo',
    title='Goldman Sachs - Last 100 Days',
    mav=(20, 50),
    figsize=(14, 6)
)
```

In [99]:
```
# Filter last 50 days
df_recent = df.tail(50)
```

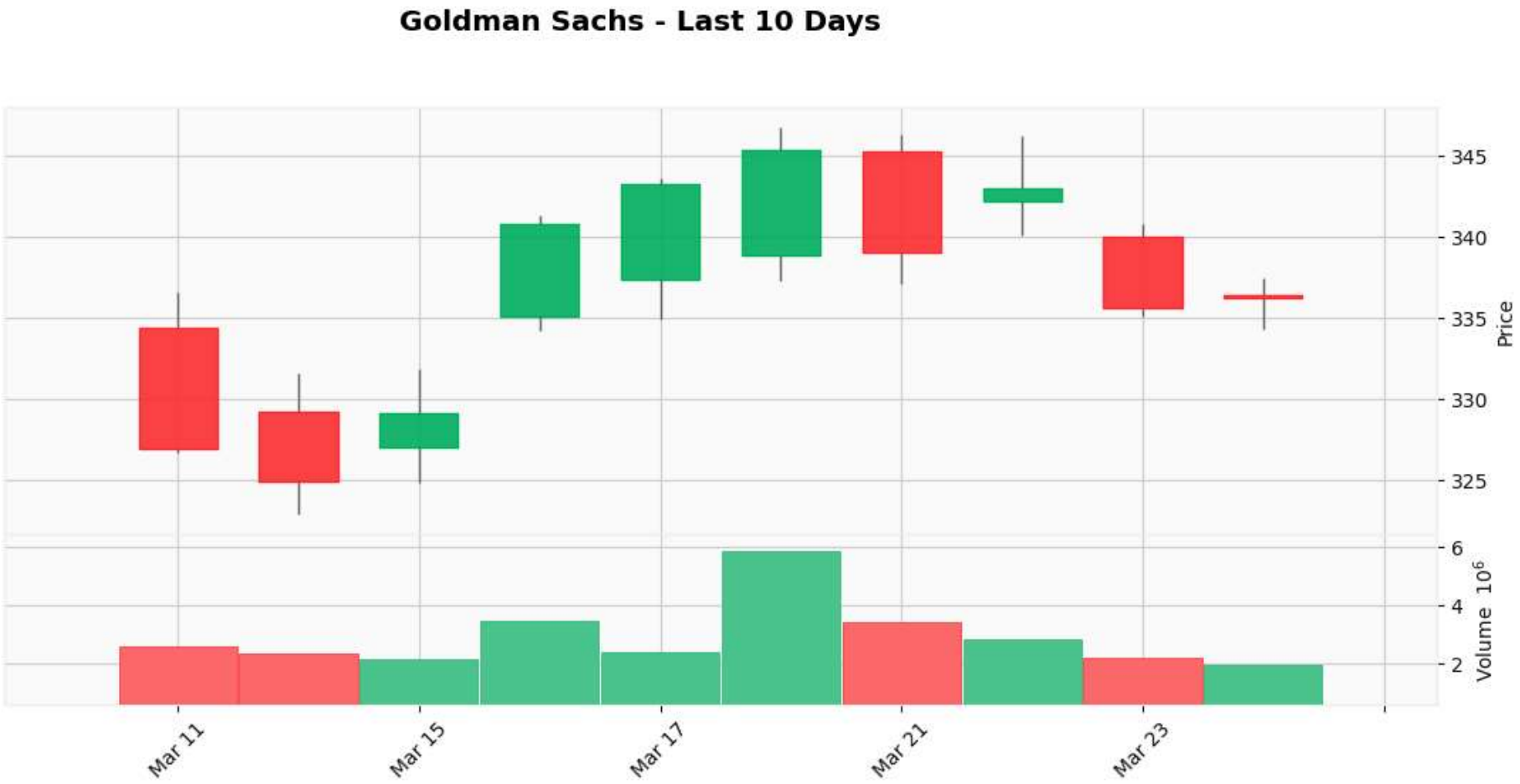
```
mpf.plot(
    df_recent,
    type='candle',
    volume=True,
    style='yahoo',
    title='Goldman Sachs - Last 50 Days',
    mav=(20, 50),
    figsize=(14, 6)
)
```



In [101…

```
# Filter last 10 days
df_recent = df.tail(10)
```

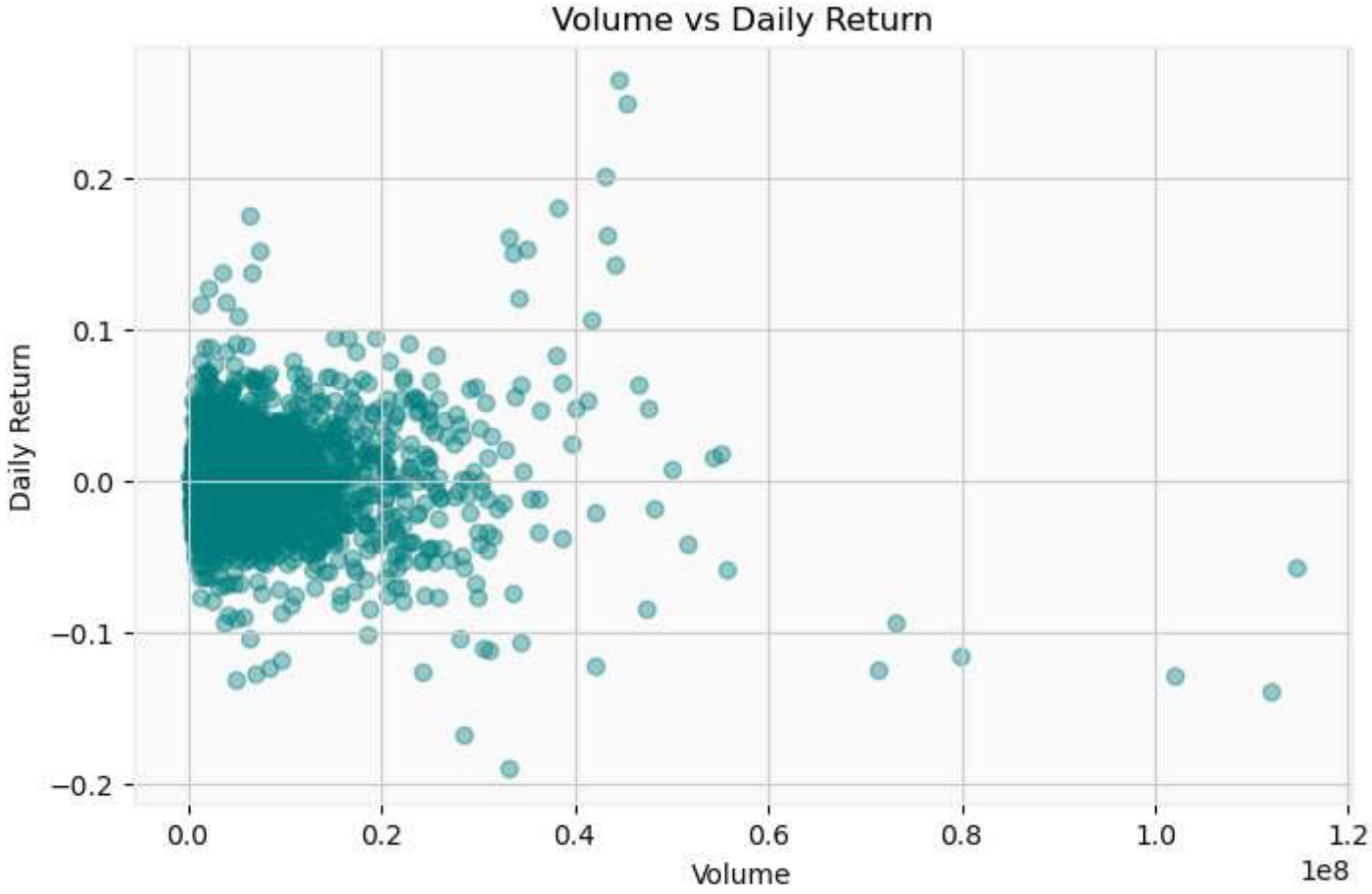
```
mpf.plot(
    df_recent,
    type='candle',
    volume=True,
    style='yahoo',
    title='Goldman Sachs - Last 10 Days',
    mav=(20, 50),
    figsize=(14, 6)
)
```



In [107…
```
corr_df = df[['Daily Return', 'Volume']].dropna()
correlation = corr_df['Daily Return'].corr(corr_df['Volume'])
print(f"Correlation between Daily Return and Volume: {correlation:.4f}")
```

```
Correlation between Daily Return and Volume: -0.0257
```

In [109…
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.scatter(corr_df['Volume'], corr_df['Daily Return'], alpha=0.4, color='teal')
plt.title('Volume vs Daily Return')
plt.xlabel('Volume')
plt.ylabel('Daily Return')
plt.grid(True)
plt.show()
```



Above scatter plot is looking like mostly no relation , because we can not identify the clear and spacific pattern here

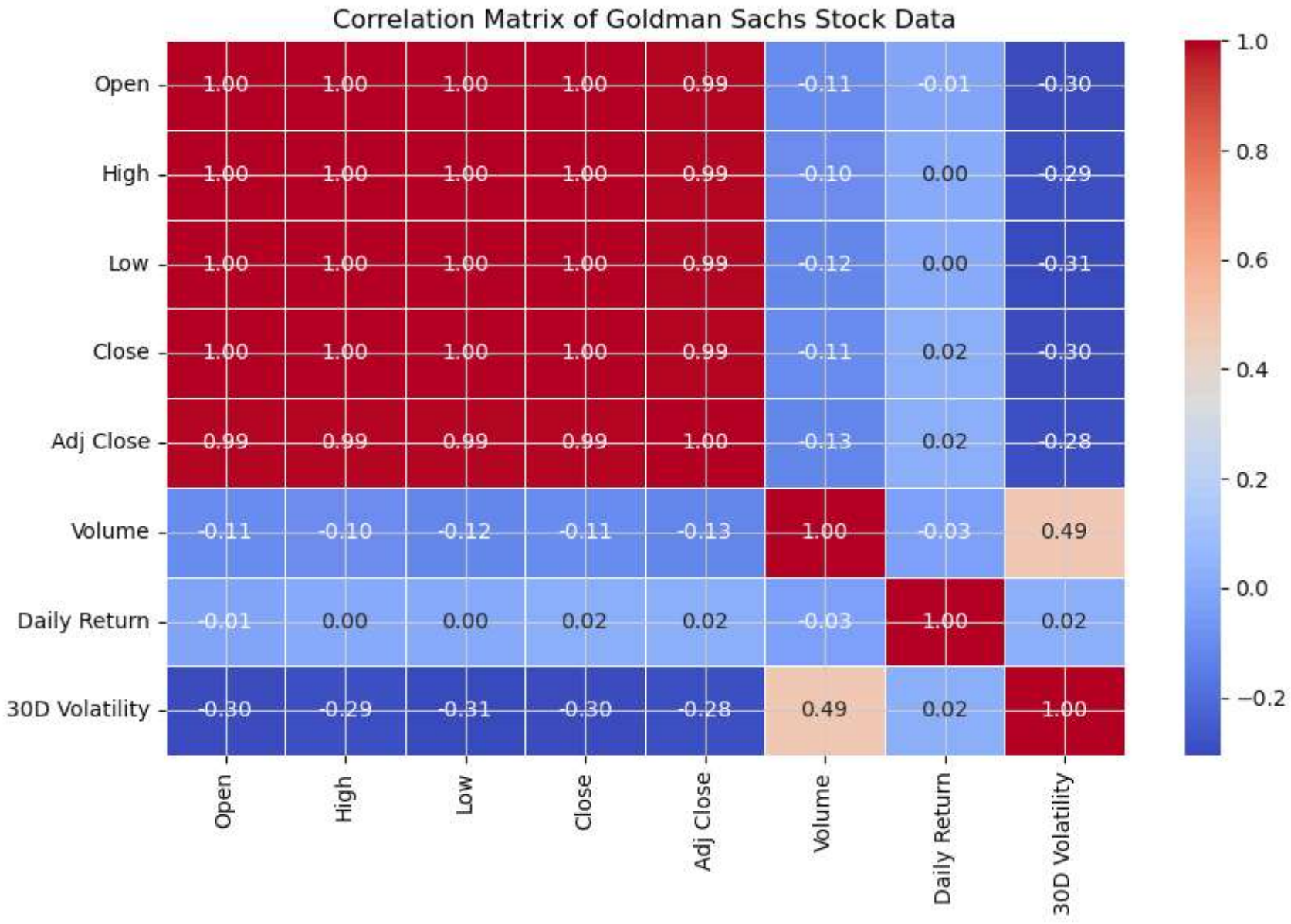
In [111…

```
correlation_matrix = df.corr(numeric_only=True) # creating the co-relation matrix
print(correlation_matrix)
```

```
                    Open      High       Low     Close  Adj Close    Volume  \
Open            1.000000  0.999635  0.999551  0.999222   0.993664 -0.105193
High            0.999635  1.000000  0.999368  0.999611   0.994014 -0.096869
Low             0.999551  0.999368  1.000000  0.999600   0.993984 -0.117488
Close           0.999222  0.999611  0.999600  1.000000   0.994347 -0.106611
Adj Close       0.993664  0.994014  0.993984  0.994347   1.000000 -0.126727
Volume         -0.105193 -0.096869 -0.117488 -0.106611  -0.126727  1.000000
Daily Return   -0.011109  0.002739  0.003664  0.018523   0.015756 -0.025743
30D Volatility -0.296851 -0.287582 -0.306435 -0.296408  -0.281359  0.486114

                Daily Return  30D Volatility
Open               -0.011109       -0.296851
High                0.002739       -0.287582
Low                 0.003664       -0.306435
Close               0.018523       -0.296408
Adj Close           0.015756       -0.281359
Volume             -0.025743        0.486114
Daily Return        1.000000        0.021284
30D Volatility      0.021284        1.000000
```

In [117…

```python
import seaborn as sns # Checking the coorelation here
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title('Correlation Matrix of Goldman Sachs Stock Data')
plt.show()
```



Technical indicators like RSI, MACD, and Bollinger Bands are powerful tools for identifying market trends, momentum, and

overbought/oversold conditions.

To calculate these indicators easily, we can use the Python library ta (Technical Analysis Library in Python).

In [123…
```
pip install ta
```

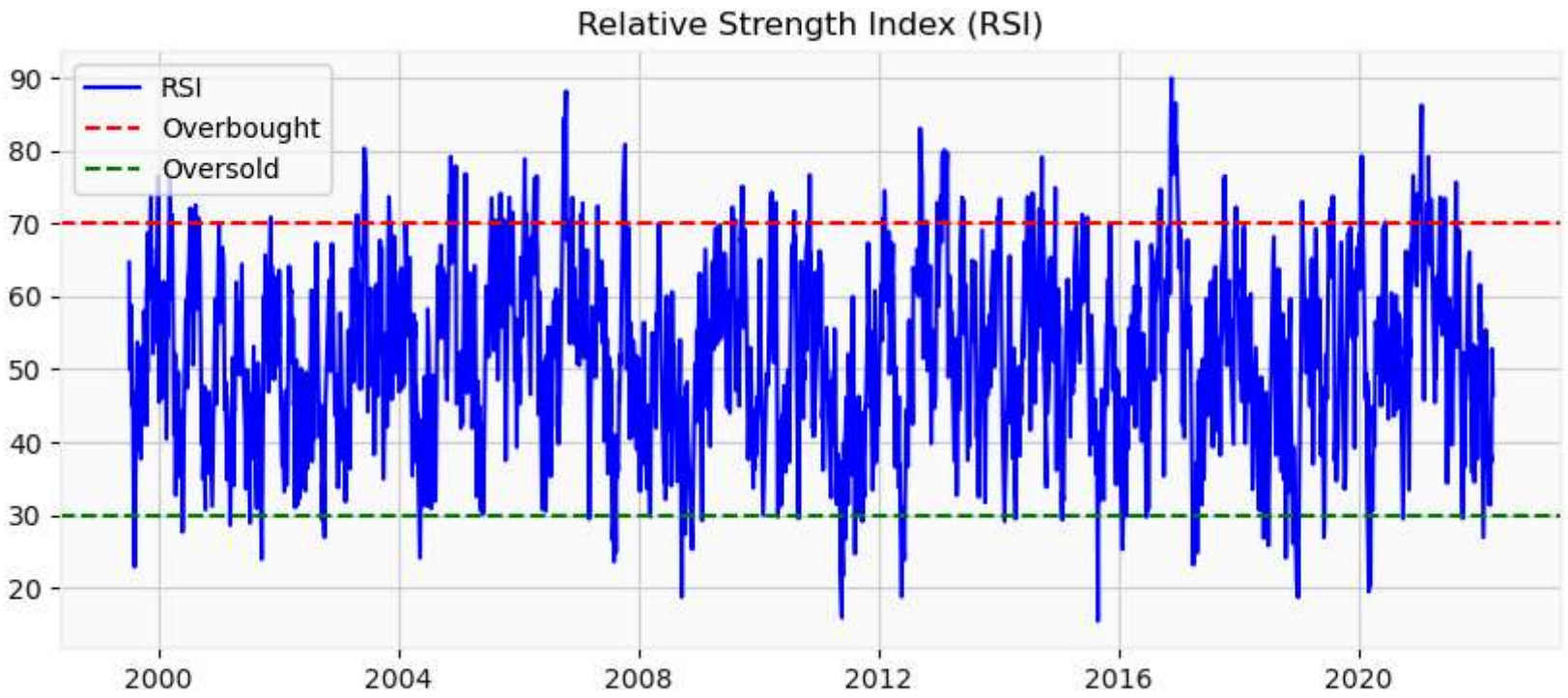
```
Collecting taNote: you may need to restart the kernel to use updated packages.

  Downloading ta-0.11.0.tar.gz (25 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: numpy in c:\users\dell\anaconda3\lib\site-packages (from ta) (1.26.4)
Requirement already satisfied: pandas in c:\users\dell\anaconda3\lib\site-packages (from ta) (2.2.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\dell\anaconda3\lib\site-packages (from pandas->ta) (2.9.0.pos
t0)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\site-packages (from pandas->ta) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\dell\anaconda3\lib\site-packages (from pandas->ta) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas->ta)
(1.16.0)
Building wheels for collected packages: ta
  Building wheel for ta (setup.py): started
  Building wheel for ta (setup.py): finished with status 'done'
  Created wheel for ta: filename=ta-0.11.0-py3-none-any.whl size=29421 sha256=74a448b13e3ec8425c08dd95e792adcba9a9d7f90b13d2a4a
3fedc13841dcd97
  Stored in directory: c:\users\dell\appdata\local\pip\cache\wheels\5c\a1\5f\c6b85a7d9452057be4ce68a8e45d77ba34234a6d46581777c6
Successfully built ta
Installing collected packages: ta
Successfully installed ta-0.11.0
```

In [125…
```
# Import and apply indecator

from ta.trend import MACD
from ta.momentum import RSIIndicator
from ta.volatility import BollingerBands

# Drop NaN values
```

```
df = df.dropna()

# RSI
rsi = RSIIndicator(close=df['Close'], window=14)
df['RSI'] = rsi.rsi()

# MACD
macd = MACD(close=df['Close'])
df['MACD'] = macd.macd()
df['MACD_Signal'] = macd.macd_signal()

# Bollinger Bands
bb = BollingerBands(close=df['Close'], window=20, window_dev=2)
df['BB_Upper'] = bb.bollinger_hband()
df['BB_Lower'] = bb.bollinger_lband()
df['BB_Middle'] = bb.bollinger_mavg()
```

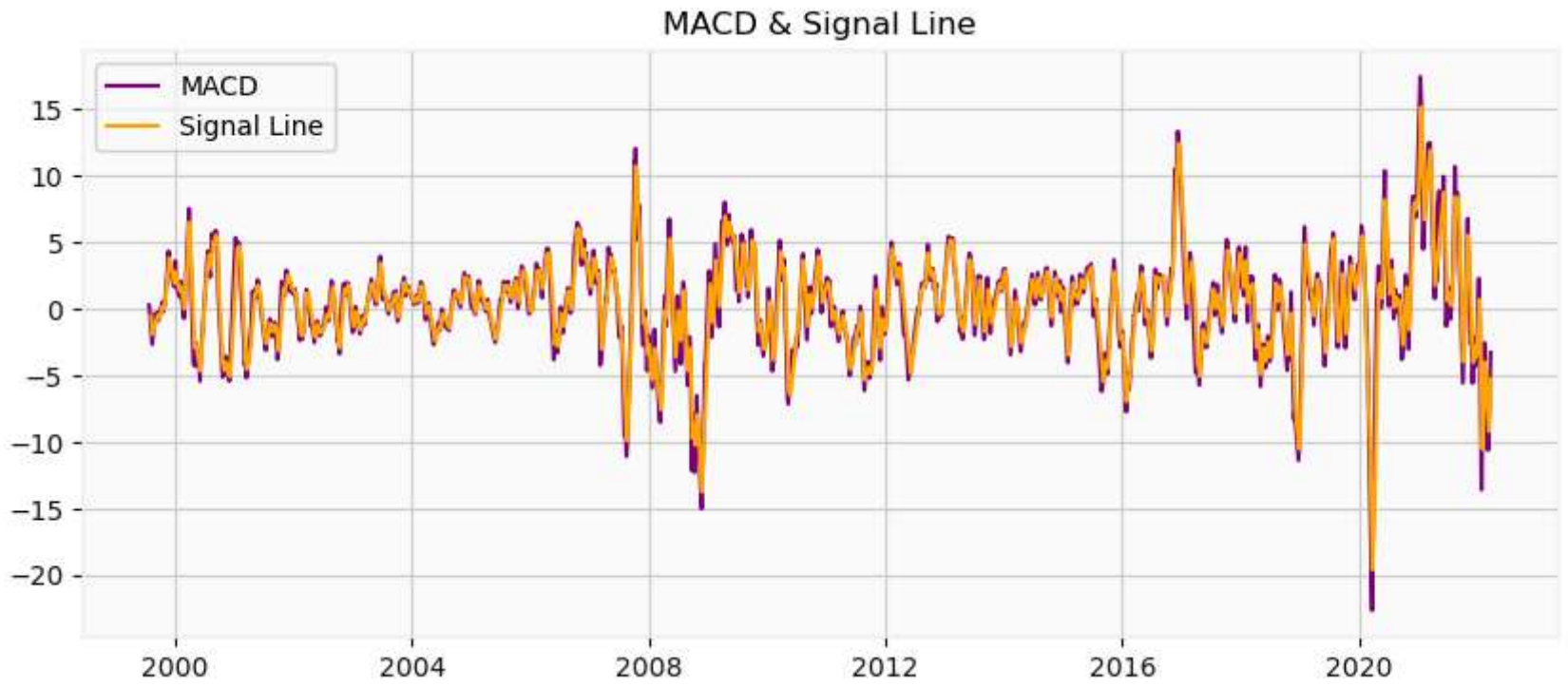
In [127…

```
# Visulize each indecator

plt.figure(figsize=(10, 4))
plt.plot(df['RSI'], label='RSI', color='blue')
plt.axhline(70, color='red', linestyle='--', label='Overbought')
plt.axhline(30, color='green', linestyle='--', label='Oversold')
plt.title('Relative Strength Index (RSI)')
plt.legend()
plt.grid(True)
plt.show()
```



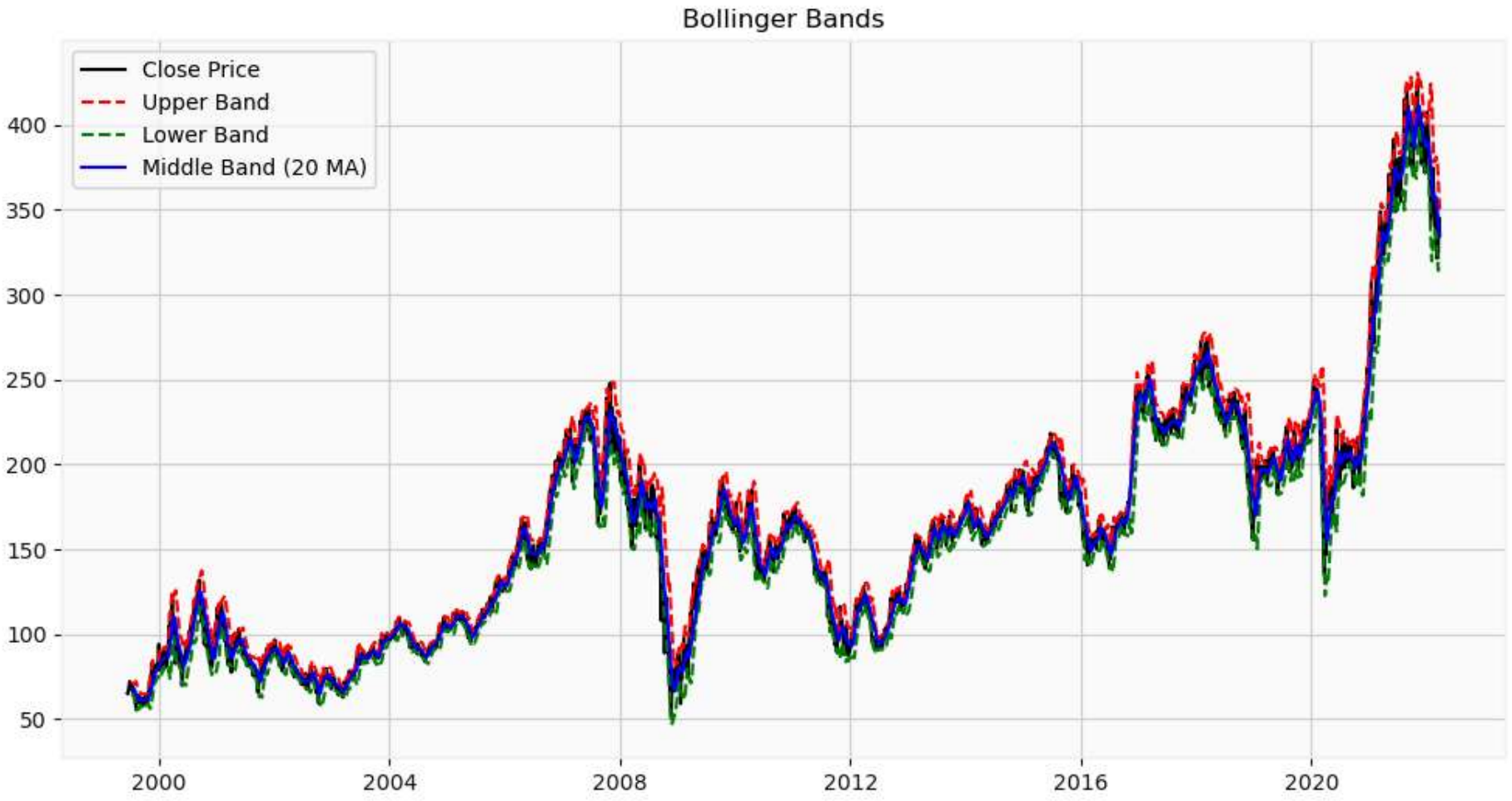
In [129…

```
# MACD Plot

plt.figure(figsize=(10, 4))
plt.plot(df['MACD'], label='MACD', color='purple')
plt.plot(df['MACD_Signal'], label='Signal Line', color='orange')
plt.title('MACD & Signal Line')
plt.legend()
plt.grid(True)
plt.show()
```



In [131…

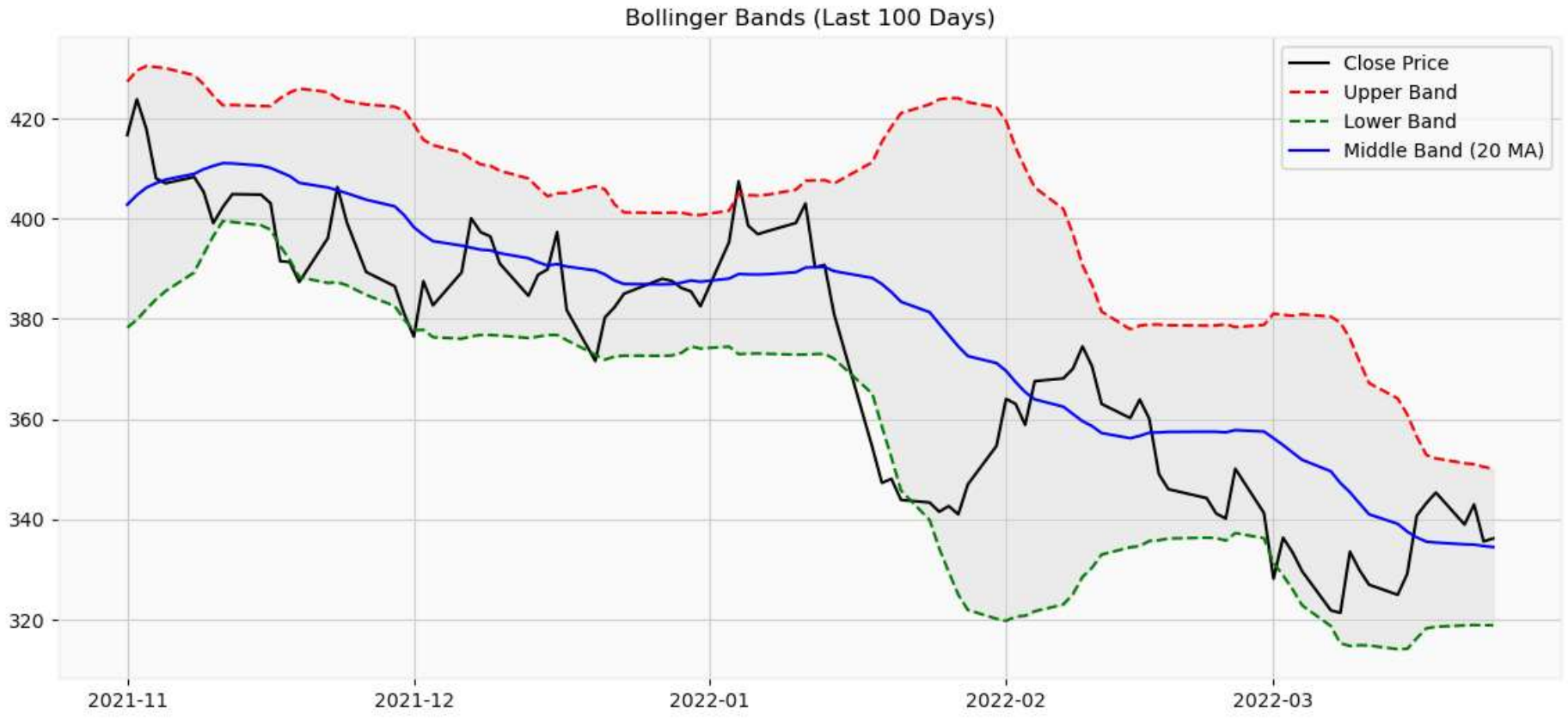
```
# Bollinger Bands with Price

plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Close Price', color='black')
plt.plot(df['BB_Upper'], label='Upper Band', color='red', linestyle='--')
plt.plot(df['BB_Lower'], label='Lower Band', color='green', linestyle='--')
plt.plot(df['BB_Middle'], label='Middle Band (20 MA)', color='blue')
plt.fill_between(df.index, df['BB_Lower'], df['BB_Upper'], color='lightgray', alpha=0.3)
plt.title('Bollinger Bands')
plt.legend()
plt.grid(True)
plt.show()
```



In [133…

```
# Last 100 days Bollinger bands ploting

df_bb = df.tail(100)

plt.figure(figsize=(14, 6))
plt.plot(df_bb['Close'], label='Close Price', color='black')
plt.plot(df_bb['BB_Upper'], label='Upper Band', color='red', linestyle='--')
plt.plot(df_bb['BB_Lower'], label='Lower Band', color='green', linestyle='--')
plt.plot(df_bb['BB_Middle'], label='Middle Band (20 MA)', color='blue')
plt.fill_between(df_bb.index, df_bb['BB_Lower'], df_bb['BB_Upper'], color='lightgray', alpha=0.3)
```

```
plt.title('Bollinger Bands (Last 100 Days)')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]: