```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from scipy import stats
import seaborn as sns
!pip install mplfinance --quiet
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.0/75.0 kB 2.3 MB/s eta 0:00:00
```

```
df=pd.read_csv('/content/Tesla.csv - Tesla.csv.csv')
df.head()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1692 entries, 0 to 1691
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1692 non-null   object
 1   Open       1692 non-null   float64
 2   High       1692 non-null   float64
 3   Low        1692 non-null   float64
 4   Close      1692 non-null   float64
 5   Volume     1692 non-null   int64
 6   Adj Close  1692 non-null   float64
dtypes: float64(5), int64(1), object(1)
memory usage: 92.7+ KB
```

```
df.shape
```

```
(1692, 9)
```

```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
```

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
```

```
plt.figure(figsize=(12,6))
plt.plot(df['Close'])
plt.title('Tesla Stock Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```


```
train_size = int(len(df) * 0.8)
train, test = df[:train_size], df[train_size:]
```

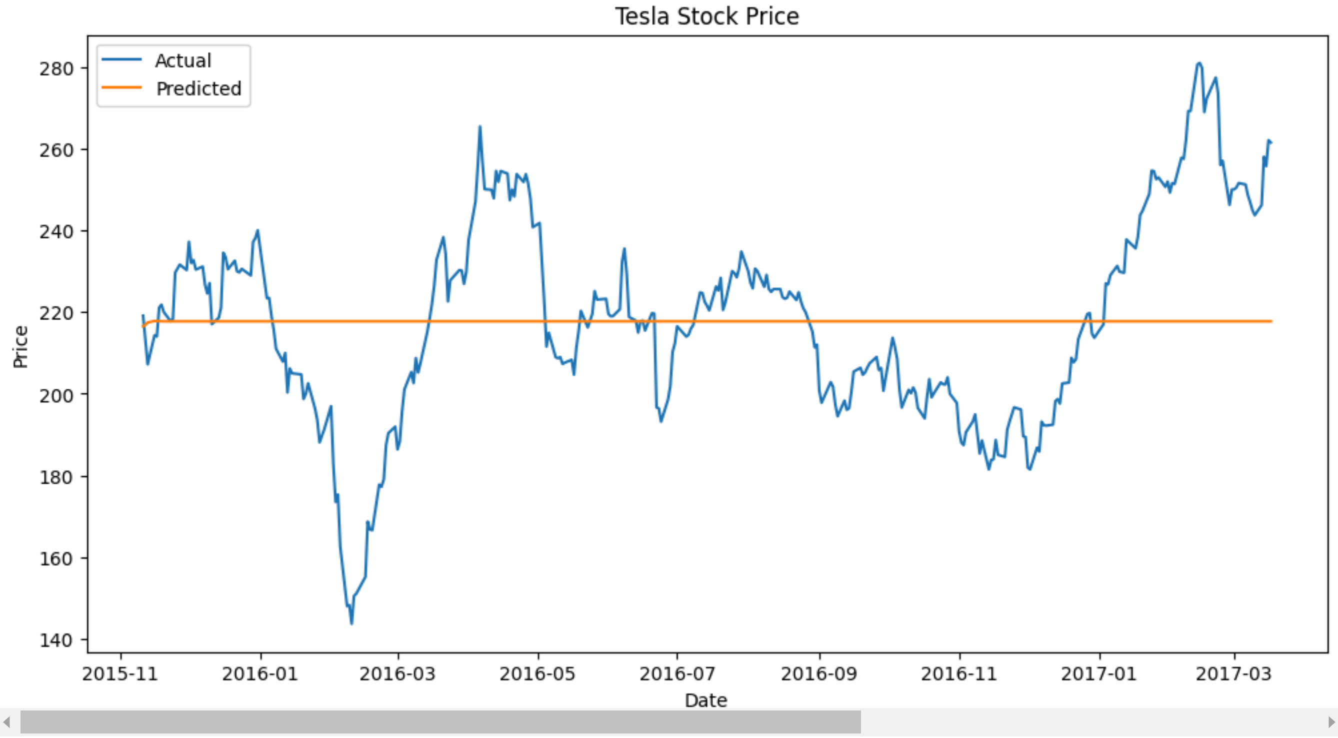
```
model = ARIMA(train['Close'], order=(5,1,0))
model_fit = model.fit()
predictions = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Predic
  return get_prediction_index(
```

```
mse = mean_squared_error(test['Close'], predictions)
rmse = np.sqrt(mse)
print('RMSE: %.2f' % rmse)
```

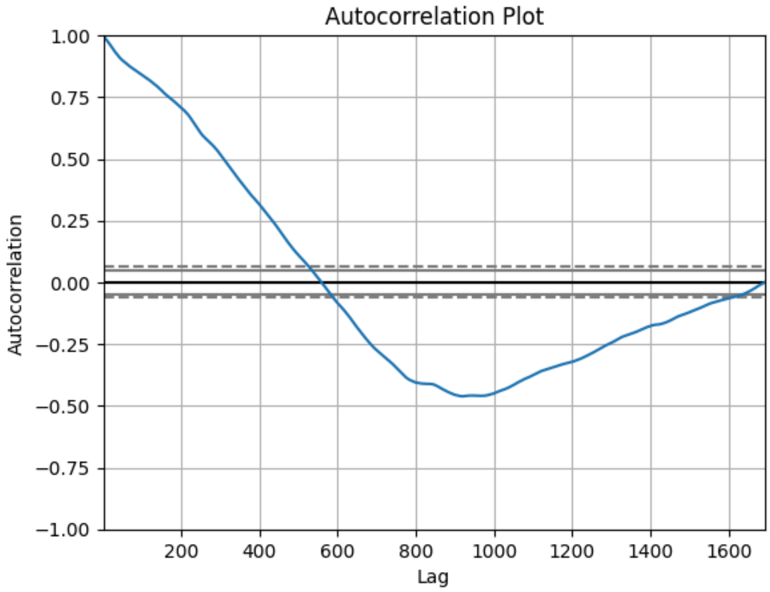
```
RMSE: 24.61
```

```
plt.figure(figsize=(12,6))
plt.plot(test.index, test['Close'], label='Actual')
plt.plot(test.index, predictions, label='Predicted')
plt.title('Tesla Stock Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

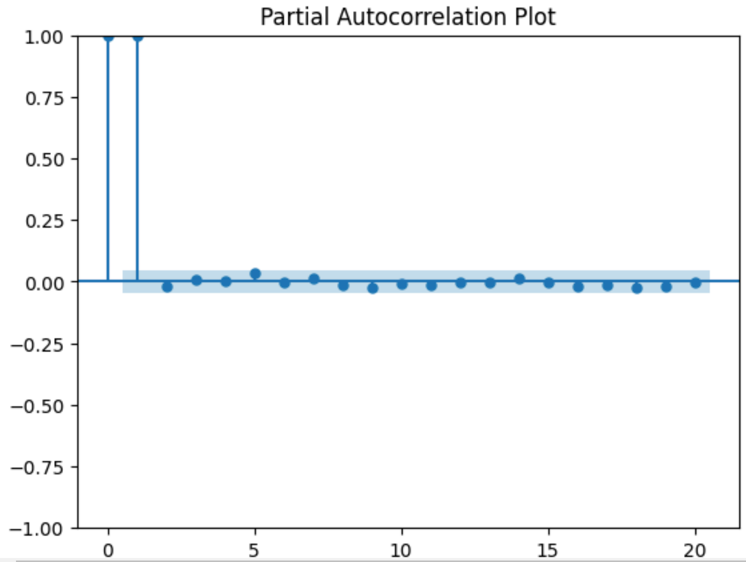



```
from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_pacf
```

```
plt.figure()
autocorrelation_plot(df['Close'])
plt.title('Autocorrelation Plot')
plt.show()

plt.figure()
plot_pacf(df['Close'], lags=20)
plt.title('Partial Autocorrelation Plot')
plt.show()
```

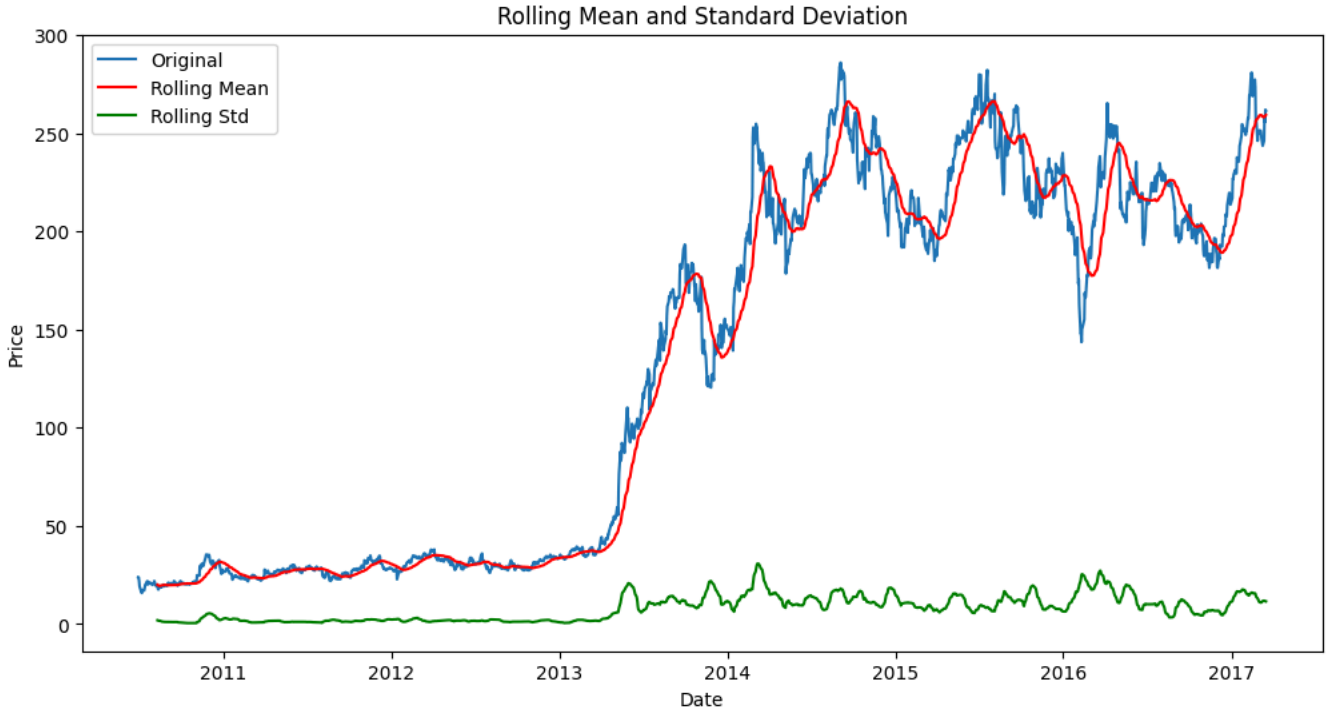


<Figure size 640x480 with 0 Axes>

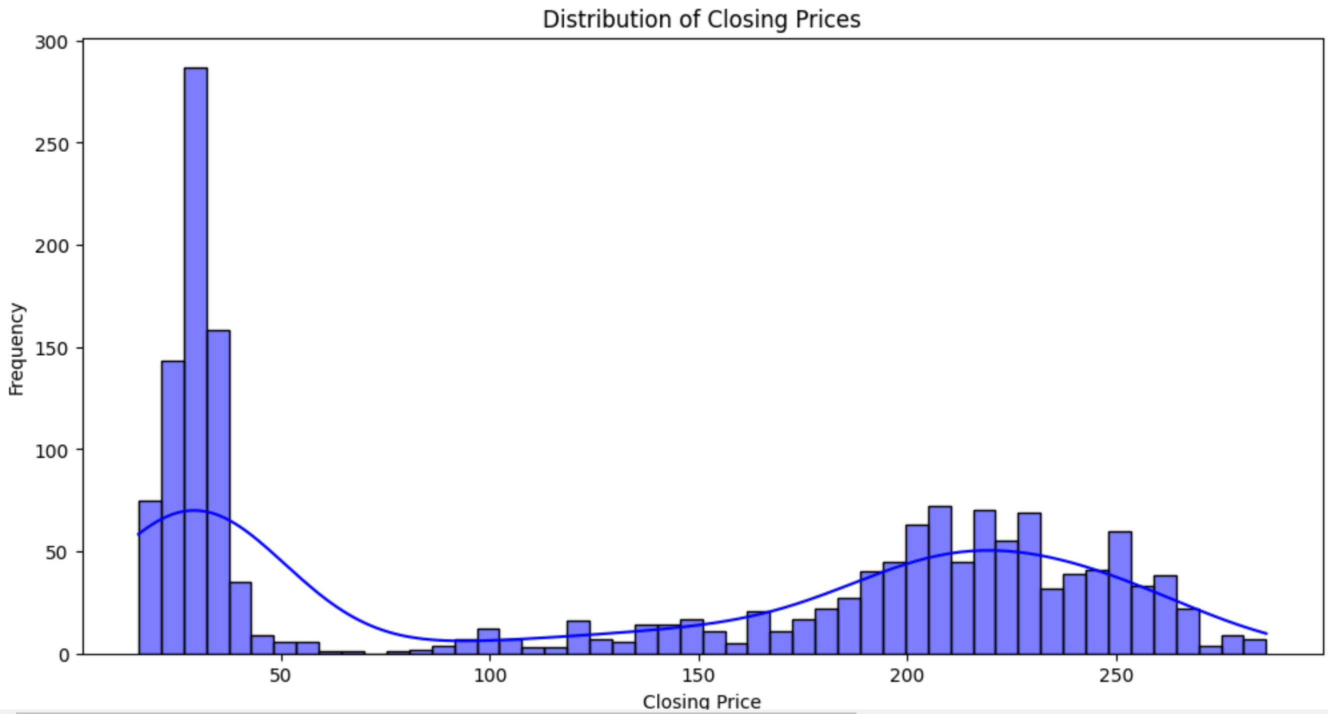



```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df['Close'], model='multiplicative', period=30)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.figure(figsize=(12, 12))

plt.subplot(411)
plt.plot(df['Close'], label='Original')
plt.legend(loc='best')
plt.title('Original')

plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.title('Trend')

plt.subplot(413)
plt.plot(seasonal, label='Seasonality')
plt.legend(loc='best')
plt.title('Seasonality')

plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.title('Residuals')

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

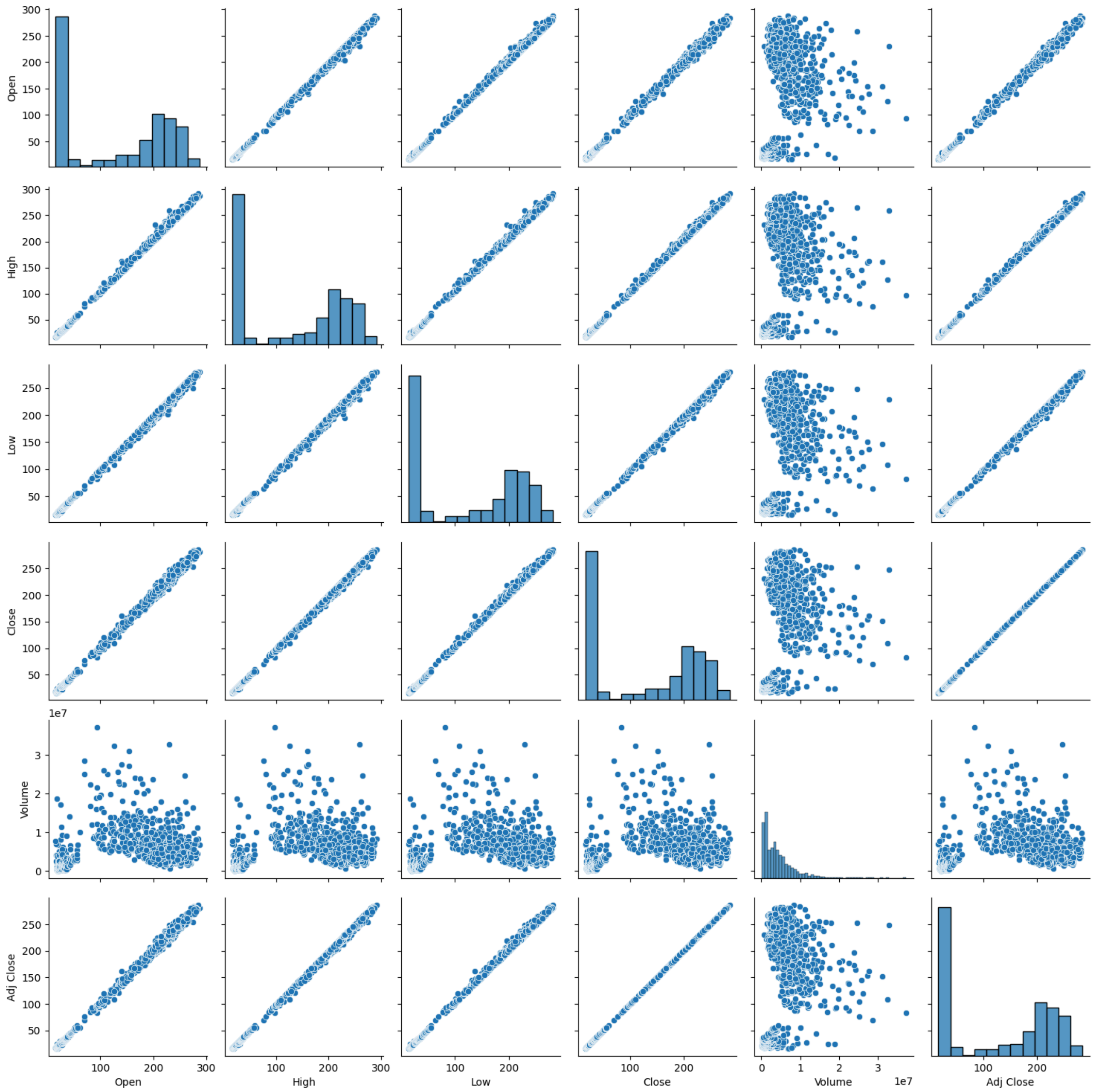

```
rolling_mean = df['Close'].rolling(window=30).mean()
rolling_std = df['Close'].rolling(window=30).std()

plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Original')
plt.plot(rolling_mean, label='Rolling Mean', color='r')
plt.plot(rolling_std, label='Rolling Std', color='g')
plt.legend(loc='best')
plt.title('Rolling Mean and Standard Deviation')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

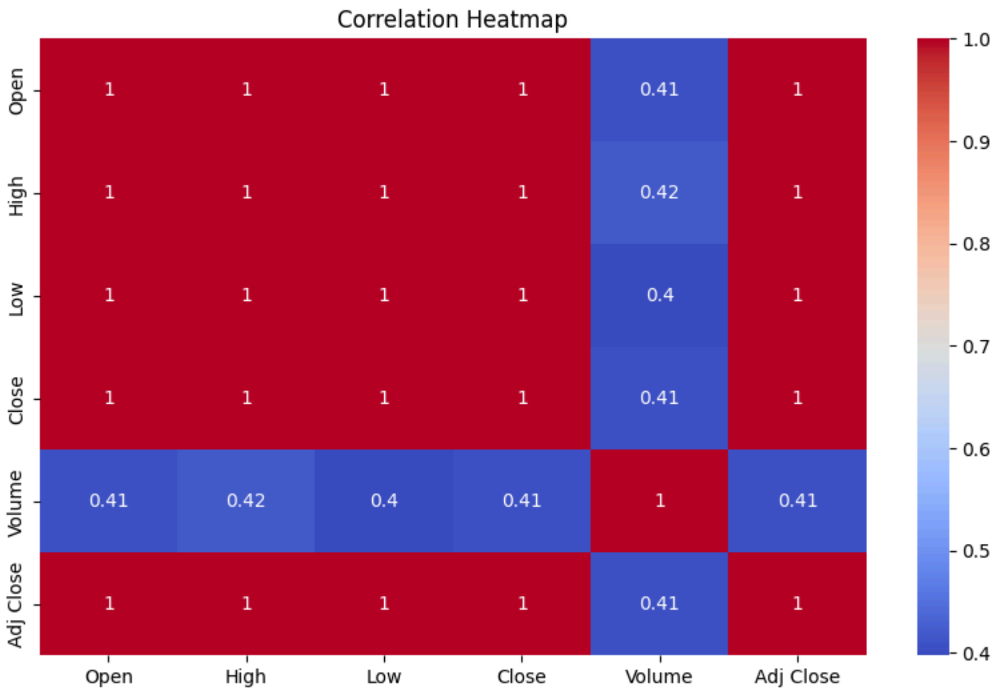



```
plt.figure(figsize=(12, 6))
sns.histplot(df['Close'], kde=True, bins=50, color='blue')
plt.title('Distribution of Closing Prices')
plt.xlabel('Closing Price')
plt.ylabel('Frequency')
plt.show()
```

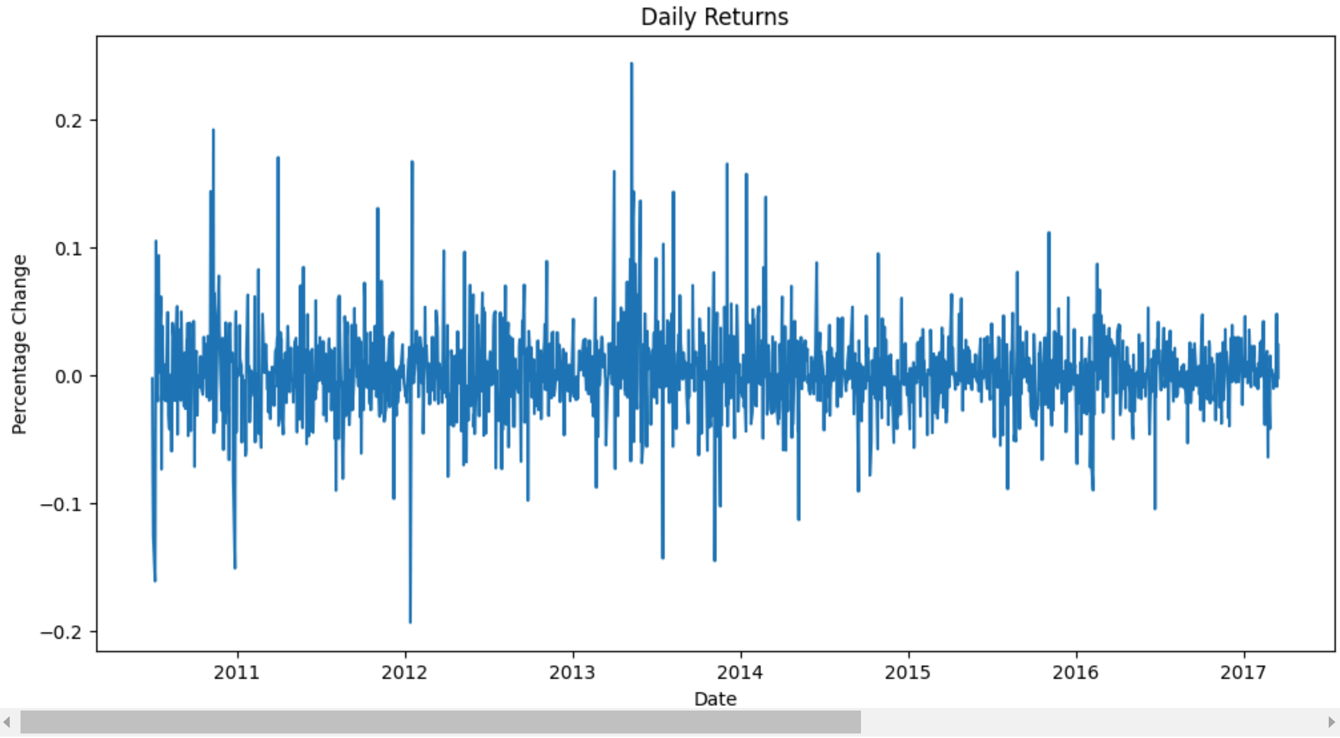



```
sns.pairplot(df[['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']])
plt.show()

plt.figure(figsize=(10, 6))
sns.heatmap(df[['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

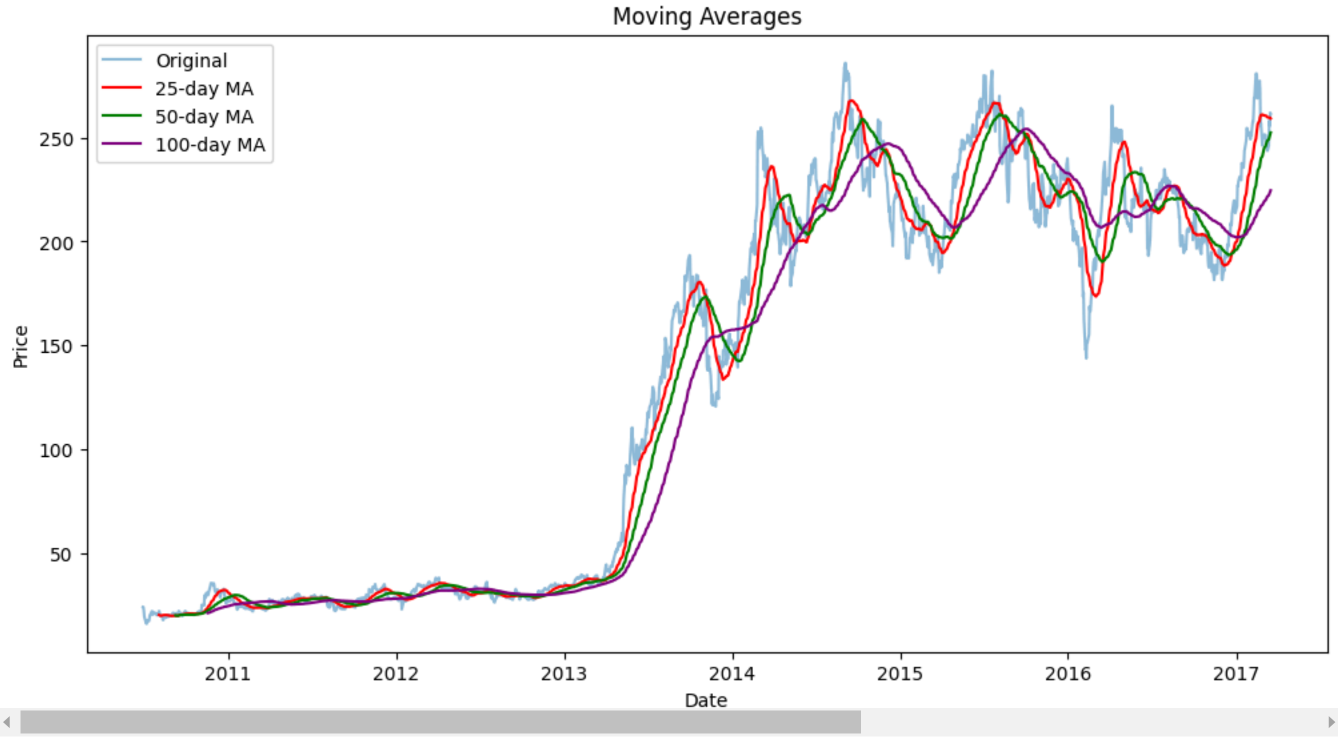

Open
High
Low
Close
Volume
Adj Close
1e7


Correlation Heatmap
Open
High
Low
Close
Volume
Adj Close

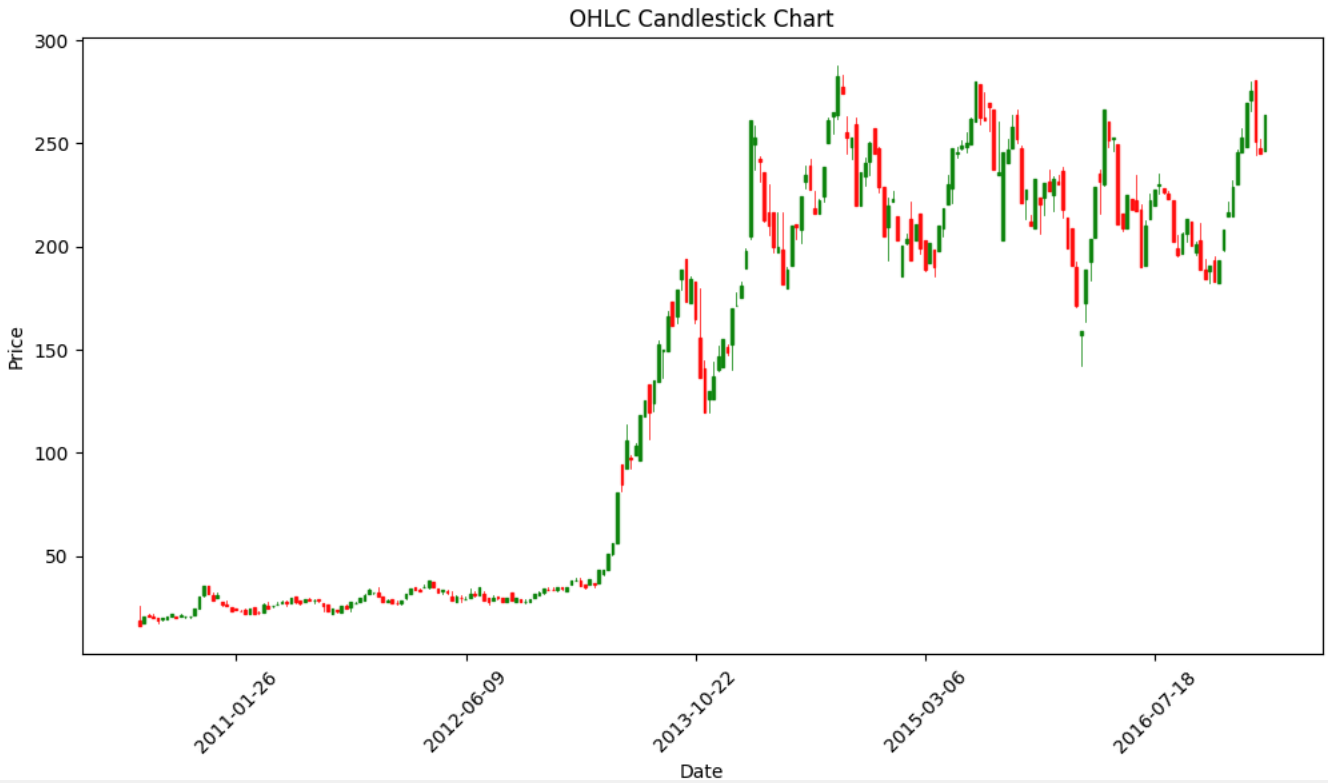
```python
daily_returns = df['Close'].pct_change().dropna()

plt.figure(figsize=(12, 6))
plt.plot(daily_returns)
plt.title('Daily Returns')
plt.xlabel('Date')
plt.ylabel('Percentage Change')
plt.show()
```




```python
from mplfinance.original_flavor import candlestick_ohlc
import matplotlib.dates as mdates

df['25_day_MA'] = df['Close'].rolling(window=25).mean()
df['50_day_MA'] = df['Close'].rolling(window=50).mean()
df['100_day_MA'] = df['Close'].rolling(window=100).mean()

plt.figure(figsize=(12, 6))
plt.plot(df['Close'], label='Original', alpha=0.5)
plt.plot(df['25_day_MA'], label='25-day MA', color='r')
plt.plot(df['50_day_MA'], label='50-day MA', color='g')
plt.plot(df['100_day_MA'], label='100-day MA', color='purple')
plt.legend(loc='best')
plt.title('Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

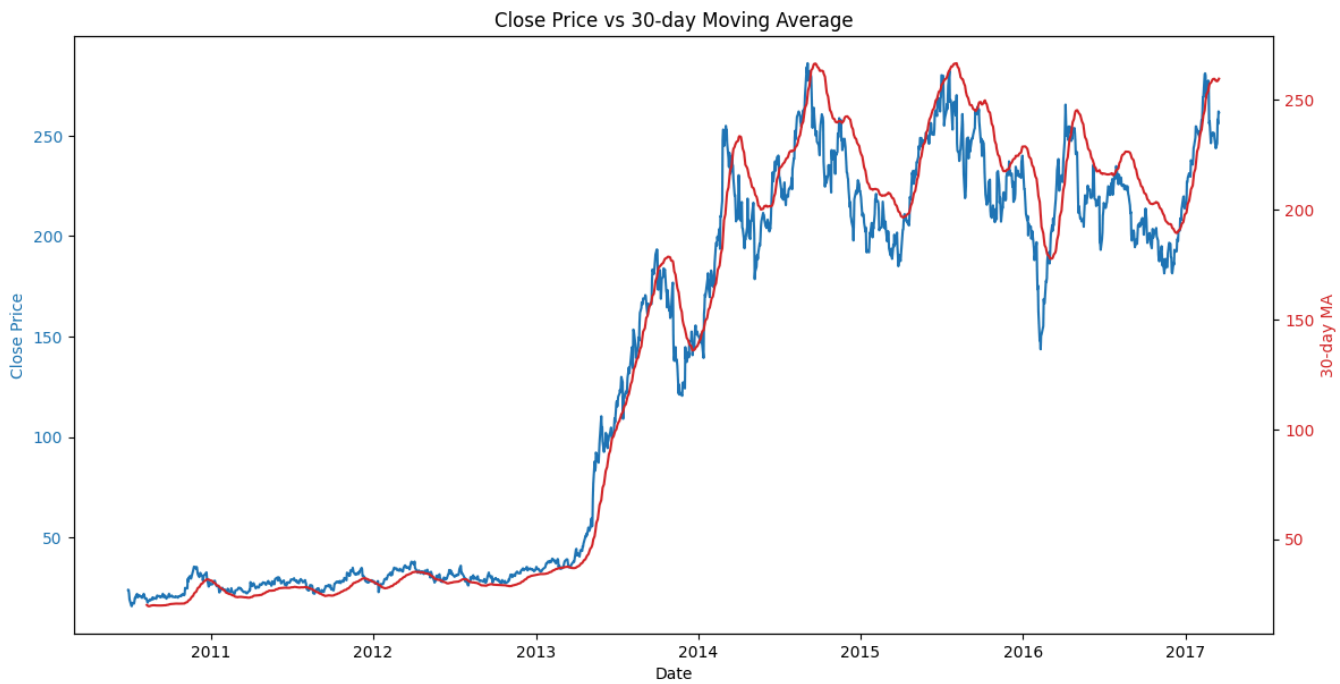



```
ohlc_data = df[['Open', 'High', 'Low', 'Close', 'Volume']].resample('10D').ohlc().reset_index()
ohlc_data['Date'] = ohlc_data['Date'].map(mdates.date2num)

fig, ax = plt.subplots(figsize=(12, 6))
candlestick_ohlc(ax, ohlc_data.values, width=5, colorup='g', colordown='r', alpha=0.8)
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.xticks(rotation=45)
plt.title('OHLC Candlestick Chart')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```




```
fig, ax1 = plt.subplots(figsize=(12, 6))

color = 'tab:blue'
ax1.set_xlabel('Date')
ax1.set_ylabel('Close Price', color=color)
ax1.plot(df.index, df['Close'], label='Close', color=color)
```

```
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()

color = 'tab:red'
ax2.set_ylabel('30-day MA', color=color)
ax2.plot(df.index, df['30_day_MA'], label='30-day MA', color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.title('Close Price vs 30-day Moving Average')
plt.show()
```




```
import itertools

p = range(0, 6)
d = range(0, 3)
q = range(0, 6)

pdq_combinations = list(itertools.product(p, d, q))

def find_best_arima(train_data, pdq_values):
    best_aic = float('inf')
    best_order = None

    for order in pdq_values:
        try:
            model = ARIMA(train_data, order=order)
            model_fit = model.fit()
            aic = model_fit.aic

            if aic < best_aic:
                best_aic = aic
                best_order = order

        except:
            continue

    return best_order, best_aic

best_order, best_aic = find_best_arima(train['Close'], pdq_combinations)
print(f"Best ARIMA order: {best_order} with AIC: {best_aic}")

model = ARIMA(train['Close'], order=best_order)
model_fit = model.fit()

predictions = model_fit.predict(start=len(train), end=len(train) + len(test) - 1, dynamic=False

```