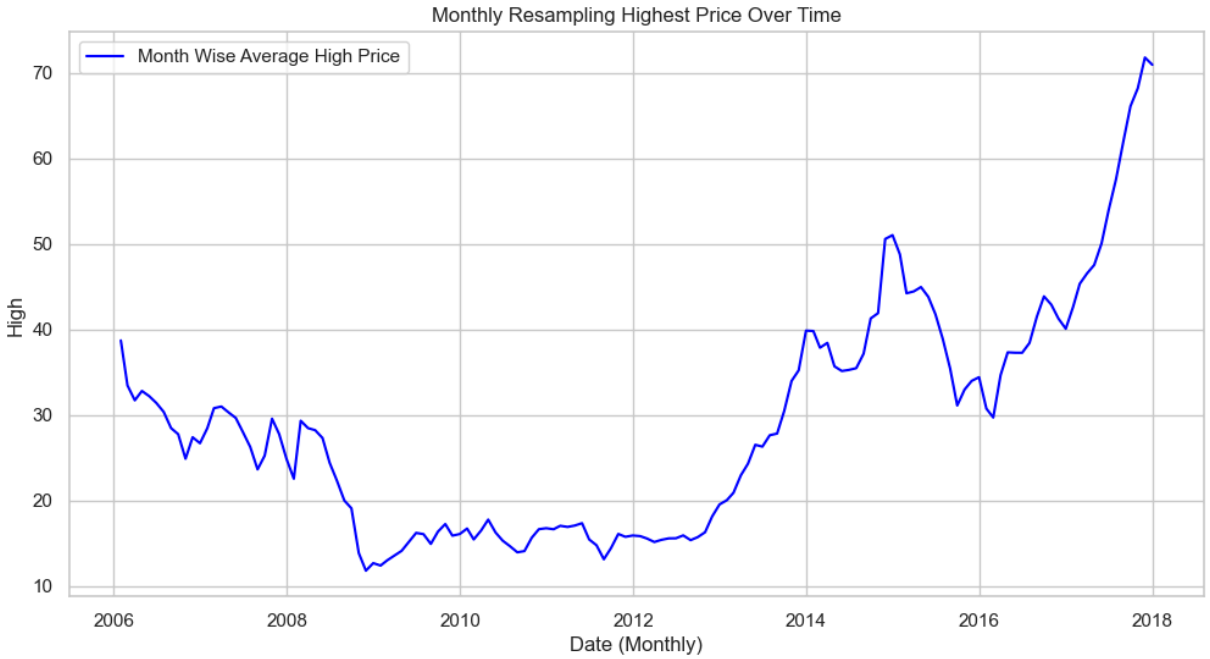
In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import adfuller

df = pd.read_csv('stock_data.csv', parse_dates=True, index_col='Date')
df.drop('Unnamed: 0', axis=1, inplace= True)
df.head(10)
```

Out[ ]:

Date	Open	High	Low	Close	Volume	Name
2006-01-03	39.69	41.22	38.79	40.91	24232729	AABA
2006-01-04	41.22	41.90	40.77	40.97	20553479	AABA
2006-01-05	40.93	41.73	40.85	41.53	12829610	AABA
2006-01-06	42.88	43.57	42.80	43.21	29422828	AABA
2006-01-09	43.10	43.66	42.82	43.42	16268338	AABA
2006-01-10	42.96	43.34	42.34	42.98	16288580	AABA
2006-01-11	42.19	42.31	41.72	41.87	26192772	AABA
2006-01-12	41.92	41.99	40.76	40.89	18921686	AABA
2006-01-13	41.00	41.08	39.62	39.90	30966185	AABA
2006-01-17	39.09	40.39	38.96	40.11	42429911	AABA

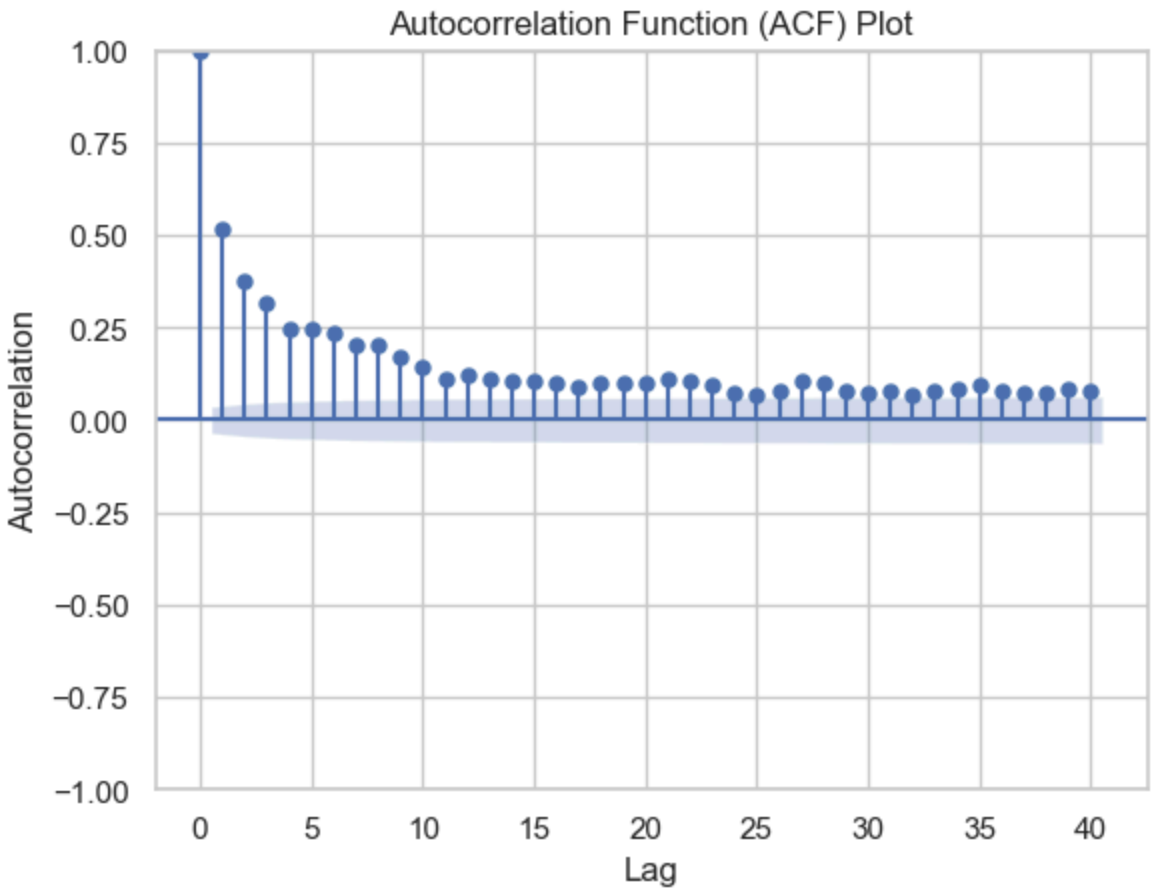
In [ ]:
```
# Setting the style to whitegrid for a clean background
sns.set(style="whitegrid")

# Setting the figure size
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='Date', y='High', label='High Price', color='blue')

# Adding labels and title
plt.xlabel('Date')
plt.ylabel('High')
plt.title('Share Highest Price Over Time')

plt.show()
```

In [ ]:
```
# Select only numeric columns for resampling
numeric_df = df.select_dtypes(include=[float, int])

# Resampling to monthly frequency, using mean as an aggregation function
df_resampled = numeric_df.resample('ME').mean()

# Setting the style to whitegrid for a clean background
sns.set(style="whitegrid")

# Plotting the 'high' column with seaborn, setting x as the resampled 'Date'
plt.figure(figsize=(12, 6)) # Setting the figure size
sns.lineplot(data=df_resampled, x=df_resampled.index, y='High', label='Month Wise A

# Adding labels and title
plt.xlabel('Date (Monthly)')
plt.ylabel('High')
plt.title('Monthly Resampling Highest Price Over Time')

plt.show()
```



In [ ]:
```
# Detecting Seasonality Using Auto Correlation

# Plot the ACF
plt.figure(figsize=(12, 6))
plot_acf(df['Volume'], lags=40) # You can adjust the number of lags as needed
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.title('Autocorrelation Function (ACF) Plot')
plt.show()
```

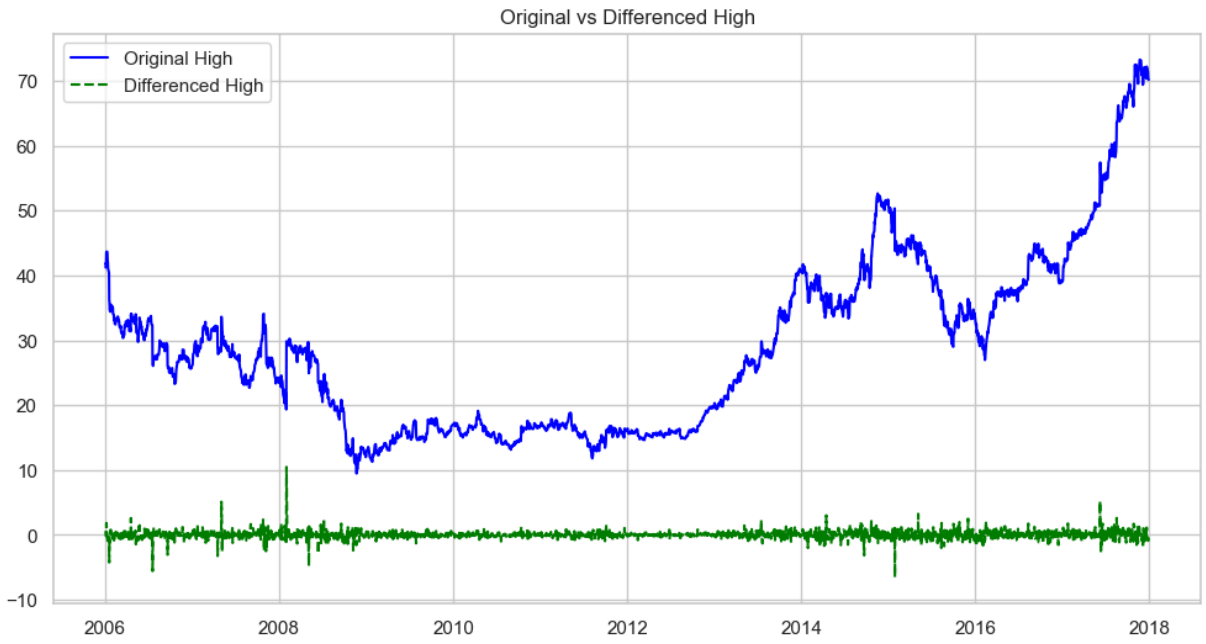
```
<Figure size 1200x600 with 0 Axes>
```



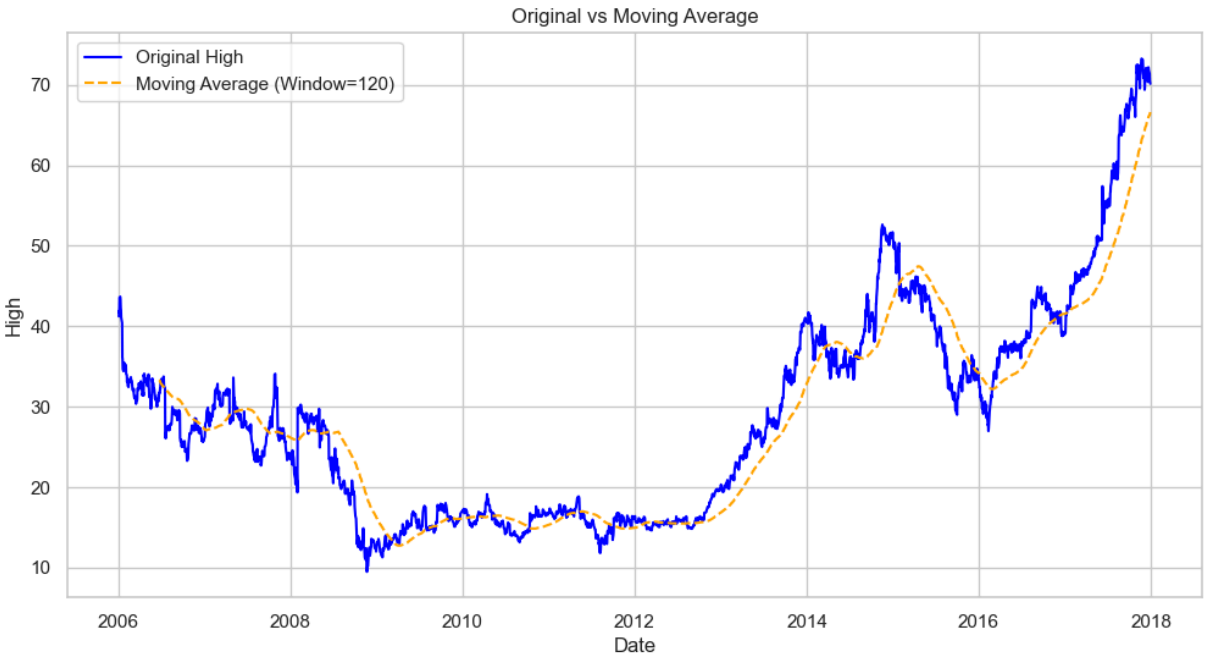
In [ ]:
```
# Detecting Stationarity

result = adfuller(df['High'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])
print('Critical Values:', result[4])
```

```
ADF Statistic: 0.7671404880535945
p-value: 0.9910868050318213
Critical Values: {'1%': np.float64(-3.4325316347197403), '5%': np.float64(-2.8625039
05260741), '10%': np.float64(-2.5672831121111113)}
```

In [ ]:
```
# Smoothening the data using Differencing and Moving Average

# Differencing
df['high_diff'] = df['High'].diff()

# Plotting
plt.figure(figsize=(12, 6))
plt.plot(df['High'], label='Original High', color='blue')
plt.plot(df['high_diff'], label='Differenced High', linestyle='--', color='green')
plt.legend()
plt.title('Original vs Differenced High')
plt.show()
```



In [ ]:
```
# Moving Average
window_size = 120
df['high_smoothed'] = df['High'].rolling(window=window_size).mean()

# Plotting
plt.figure(figsize=(12, 6))

plt.plot(df['High'], label='Original High', color='blue')
plt.plot(df['high_smoothed'], label=f'Moving Average (Window={window_size})', lines

plt.xlabel('Date')
plt.ylabel('High')
plt.title('Original vs Moving Average')
plt.legend()
plt.show()
```



In [ ]:
```
# Original Data Vs Differenced Data

# Create a DataFrame with 'high' and 'high_diff' columns side by side
df_combined = pd.concat([df['High'], df['high_diff']], axis=1)

# Display the combined DataFrame
print(df_combined.head())
```

```
             High  high_diff
Date
2006-01-03  41.22        NaN
2006-01-04  41.90       0.68
2006-01-05  41.73      -0.17
2006-01-06  43.57       1.84
2006-01-09  43.66       0.09
```

In [ ]:
```
# Remove rows with missing values
df.dropna(subset=['high_diff'], inplace=True)
df['high_diff'].head()
```

Out[ ]:
```
Date
2006-01-04    0.68
2006-01-05   -0.17
2006-01-06    1.84
2006-01-09    0.09
2006-01-10   -0.32
Name: high_diff, dtype: float64
```

In [ ]:
```
# Conduct the ADF test

result = adfuller(df['high_diff'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])
print('Critical Values:', result[4])
```

```
ADF Statistic: -12.14836747834325
p-value: 1.5912766134148351e-22
Critical Values: {'1%': np.float64(-3.4325316347197403), '5%': np.float64(-2.8625039
05260741), '10%': np.float64(-2.5672831121111113)}
```