**Exp no: 2** Implement programs for visualizing time series data.

**Date: 24/1/25**

**Aim:**

To implement program for visualizing time series data

### ****Steps for Time Series Sales Data Preprocessing****:

### ****1. Reading the Dataset****

This step involves reading the CSV file containing stock data and parsing the 'Date' column as a datetime index.

**python**

**Copy**

**# Reading the dataset using read\_csv**

**df = pd.read\_csv("stock\_data.csv",**

**parse\_dates=True,**

**index\_col="Date")**

**# Displaying the first five rows of dataset**

**df.head()**

**# Printing the column names**

**print(df.columns)**

### ****2. Plotting the High Price Over Time****

In this step, we plot the 'High' price over time using seaborn's lineplot for visualization.

**python**

**Copy**

**# Assuming df is your DataFrame**

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

**# Plotting the 'High' price with seaborn**

**plt.figure(figsize=(12, 6)) # Setting the figure size**

**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()**

### ****3. Autocorrelation Function (ACF) Plot****

In this step, we plot the autocorrelation function (ACF) for the 'Volume' column to identify any correlations in the data at different lags.

**python**

**Copy**

**# 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()**

### ****4. Differencing****

To make the time series stationary, we apply differencing on the 'High' column and plot the original and differenced series for comparison.

**python**

**Copy**

**# 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()**

### ****5. Moving Average****

We apply a moving average with a window size of 120 to smooth the 'High' price data and plot the results to visualize trends.

**python**

**Copy**

**# 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})', linestyle='--', color='orange')**

**plt.xlabel('Date')**

**plt.ylabel('High')**

**plt.title('Original vs Moving Average')**

**plt.legend()**

**plt.show()**

### ****6. Monthly Resampling****

This step involves resampling the time series data to a monthly frequency and plotting the resampled data. The mean of the 'High' price is calculated for each month.

**python**

**Copy**

**# Assuming df is your DataFrame with a datetime index**

**df\_resampled = df.resample('M').mean() # Resampling to monthly frequency, using mean as an aggregation function**

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

**# 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 Average High Price', color='blue')**

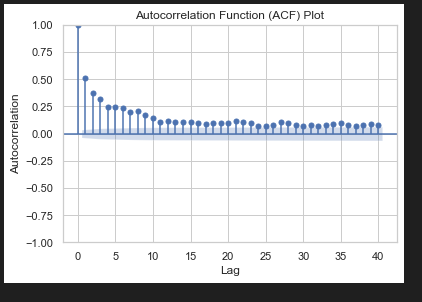
**# Adding labels and title**

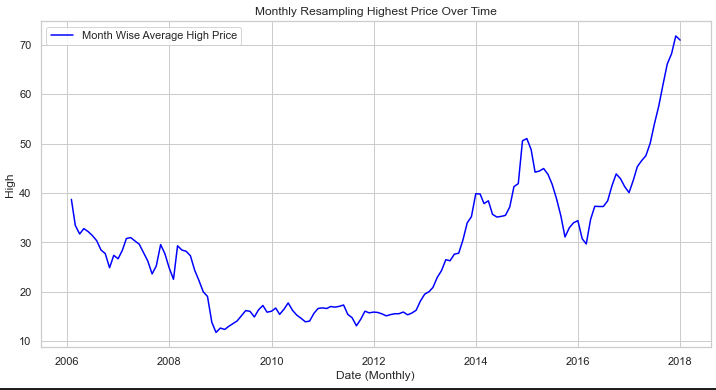
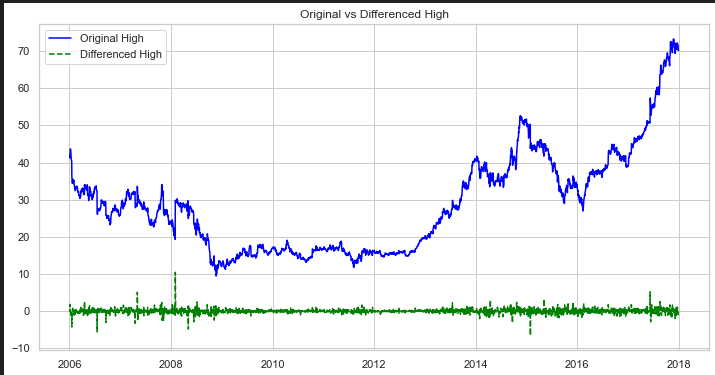
**plt.xlabel('Date (Monthly)')**

**plt.ylabel('High')**

**plt.title('Monthly Resampling Highest Price Over Time')**

output :





Result:

Thus the program is implemented.

221501024