A time series is the series of data points listed in time order.

A time series is a sequence of successive equal interval points in time.

A time-series analysis consists of methods for analyzing time series data in order to extract meaningful insights and other useful characteristics of data.

For performing time series analysis download stock_data.csv

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# reading the dataset using read_csv
df = pd.read_csv("/content/stock_data.csv",
                 parse_dates=True,
                 index_col="Date")
  # displaying the first five rows of dataset
df.head()
```

	Open	High	Low	Close	Volume	Name
Date						
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

We have used the ‘parse_dates’ parameter in the read_csv function to convert the ‘Date’ column to the DatetimeIndex format.

By default, Dates are stored in string format which is not the right format for time series data analysis.

Now, removing the unwanted columns from dataframe i.e. ‘Unnamed: 0’.

```
# deleting column
df.drop(columns='Close')
```

	Open	High	Low	Volume
Date				
2006-01-03	39.69	41.22	38.79	24232729
2006-01-04	41.22	41.90	40.77	20553479
2006-01-05	40.93	41.73	40.85	12829610
2006-01-06	42.88	43.57	42.80	29422828
2006-01-09	43.10	43.66	42.82	16268338
...	...	...	...	...
2014-12-23	51.46	51.46	49.93	15514036
2014-12-24	50.19	50.92	50.19	5962870
2014-12-26	50.65	51.06	50.61	5170048
2014-12-29	50.67	51.01	50.51	6624489
2014-12-30	50.35	51.27	50.35	10703455

2263 rows × 4 columns

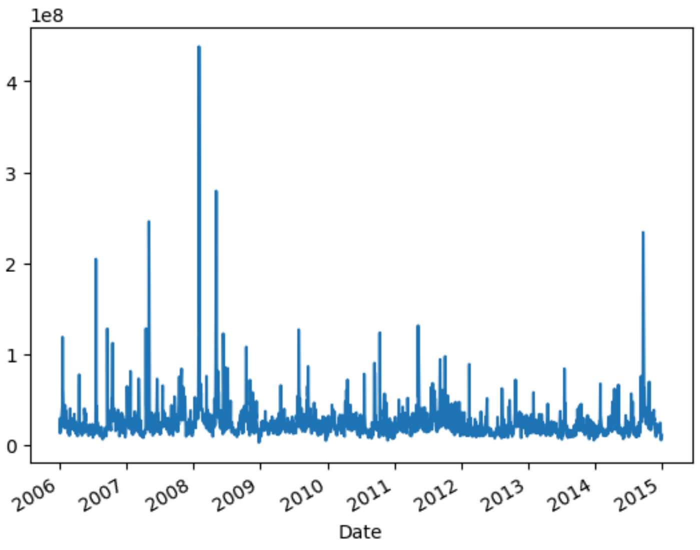
```
print(df.columns)
```

```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Name', 'Change'], dtype='object')
```

Example 1: Plotting a simple line plot for time series data.

```
df['Volume'].plot()
```

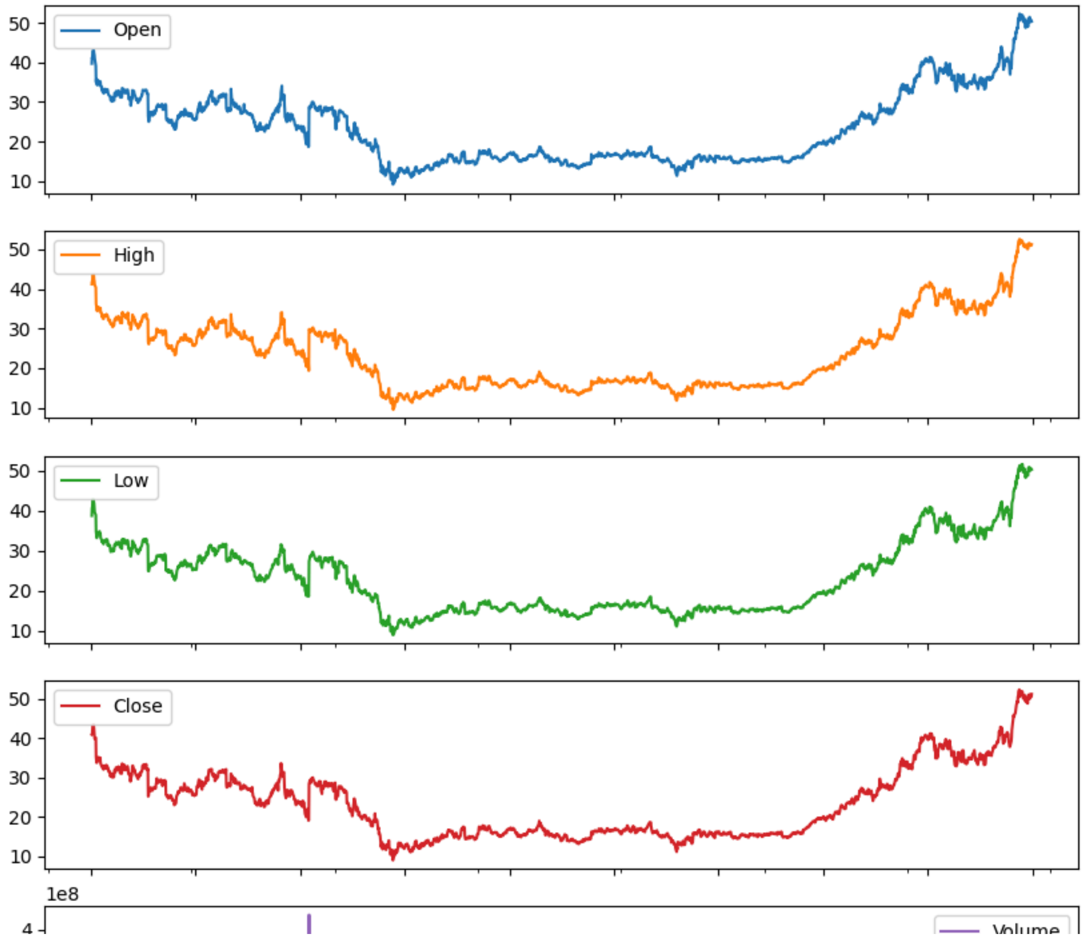
```
<Axes: xlabel='Date'>
```



Example 2: Now let’s plot all other columns using subplot.

```
df.plot(subplots=True, figsize=(10, 12))
```

```
array([<Axes: xlabel='Date'>, <Axes: xlabel='Date'>,
       <Axes: xlabel='Date'>, <Axes: xlabel='Date'>,
       <Axes: xlabel='Date'>], dtype=object)
```



```
df["Volume"].head
```

```
pandas.core.generic.NDFrame.head
def head(n: int=5) -> NDFrameT

Return the first `n` rows.

This function returns the first `n` rows for the object based
on position. It is useful for quickly testing if your object
has the right type of data in it.
```

Resampling: Resampling is a methodology of economically using a data sample to improve the accuracy and quantify the uncertainty of a population parameter. Resampling for months or weeks and making bar plots is another very simple and widely used method of finding seasonality. Here we are going to make a bar plot of month data for 2016 and 2017.

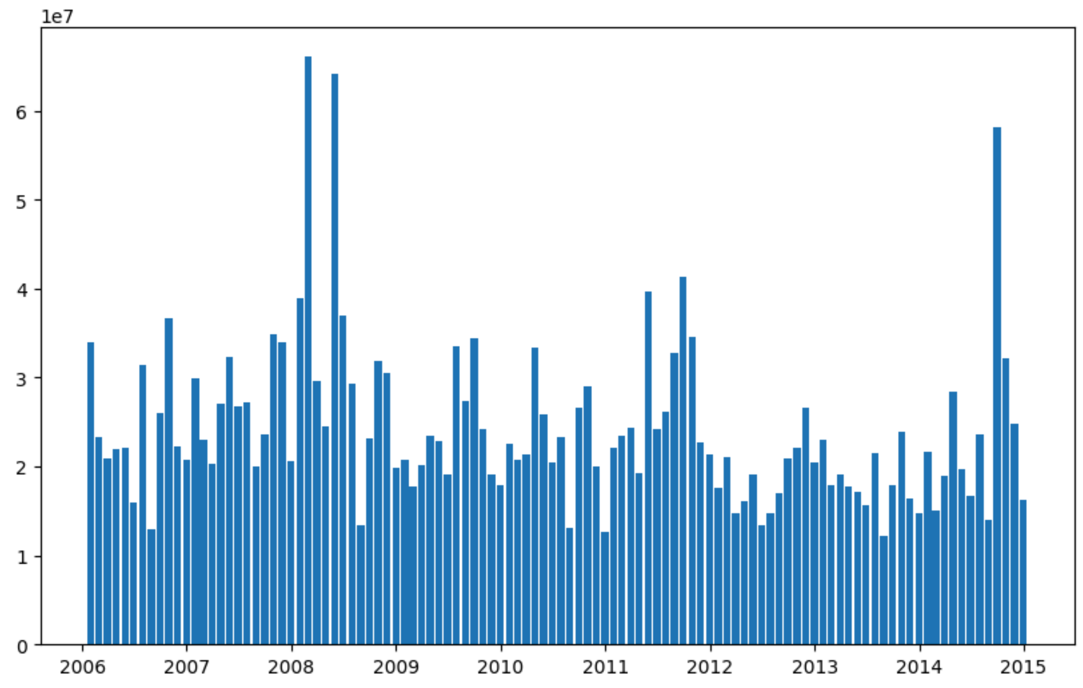
Example 3:

```
df_month = df.resample("M").mean()

# using subplot
fig, ax = plt.subplots(figsize=(10, 6))

# plotting bar graph
ax.bar(df_month['2006':].index,
       df_month.loc['2006':, "Volume"],
       width=25, align='center')
```

```
<BarContainer object of 108 artists>
```



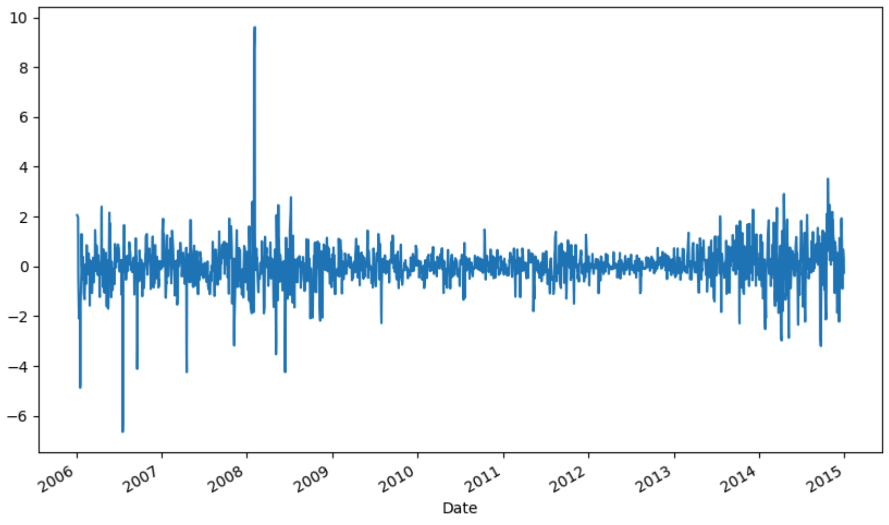
```
df["Volume"] = pd.to_numeric(df["Volume"], errors="coerce")
```

```
df["Volume"] = pd.to_numeric(df["Volume"], errors="coerce")
```

Differencing: Differencing is used to make the difference in values of a specified interval. By default, it’s one, we can specify different values for plots. It is the most popular method to remove trends in the data.

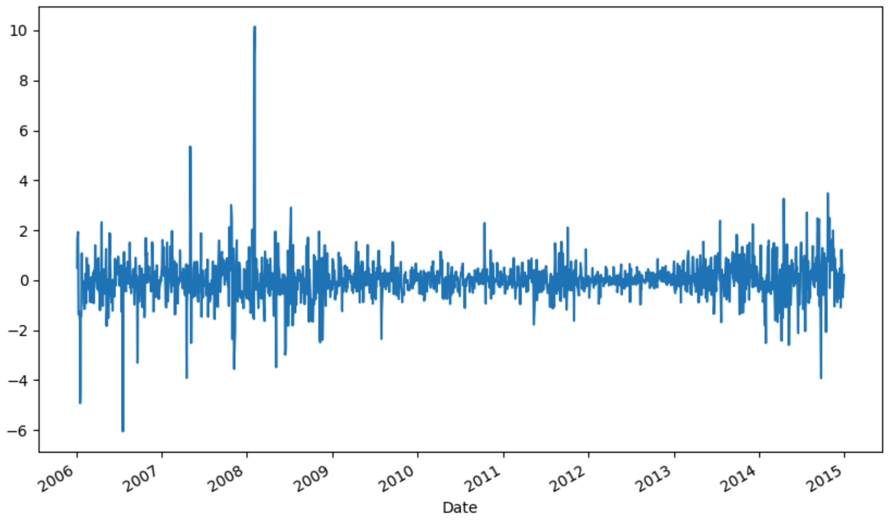
```
df.Low.diff(2).plot(figsize=(10, 6))
```

```
<Axes: xlabel='Date'>
```



```
df.High.diff(2).plot(figsize=(10, 6))
```

```
<Axes: xlabel='Date'>
```



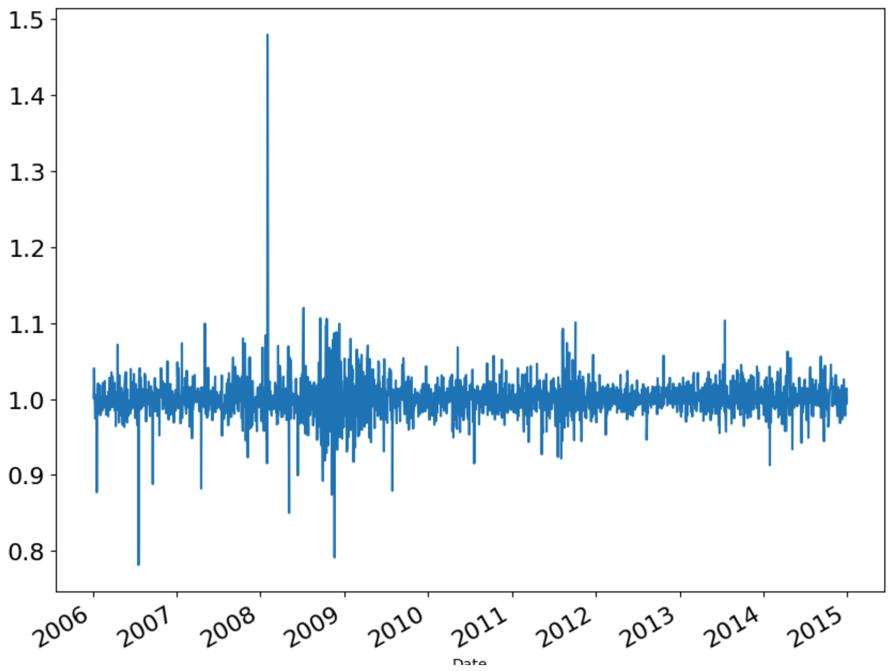
Plotting the Changes in Data

We can also plot the changes that occurred in data over time. There are a few ways to plot changes in data.

Shift: The shift function can be used to shift the data before or after the specified time interval. We can specify the time, and it will shift the data by one day by default. That means we will get the previous day's data. It is helpful to see previous day data and today's data simultaneously side by side.

```
df['Change'] = df.Close.div(df.Close.shift())
df['Change'].plot(figsize=(10, 8), fontsize=16)
```

```
<Axes: xlabel='Date'>
```



.div() function helps to fill up the missing data values.

Actually, div() means division.

If we take df. div(6) it will divide each element in df by 6.

We do this to avoid the null or missing values that are created by the 'shift()' operation.

```
sns.boxplot(data=df.drop(columns=['Volume', 'Change'], axis=1), orient='h')
plt.title("Boxplot")
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-59-b51b3d02c41b> in <cell line: 1>()
----> 1 sns.boxplot(data=df.drop(columns=['Volume', 'Change'], axis=1), orient='h')
      2 plt.title("Boxplot")
      3 plt.show()
```