ML LAB 12

Explore Holt’s Linear Exponential Smoothing, Nonlinear Trend Regression, and Seasonality for the Time Series Analysis in a given business environment.

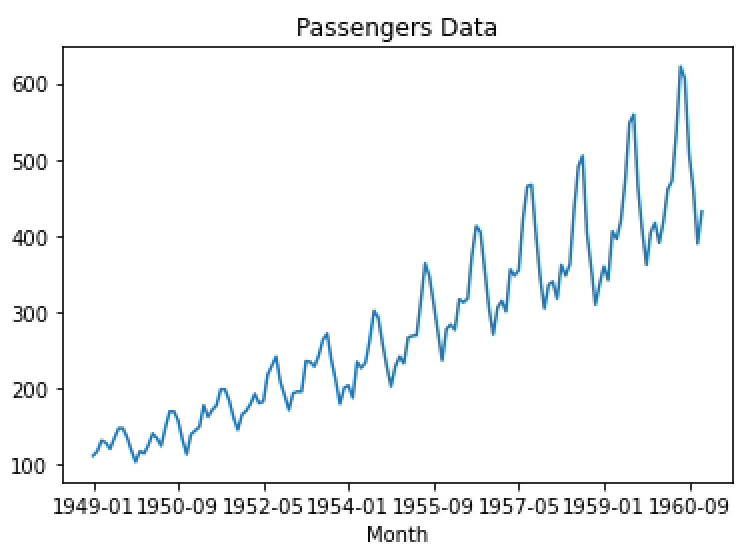
Importing the libraries

In [1]:
```
# dataframe opertations - pandas
import pandas as pd
# plotting data - matplotlib
from matplotlib import pyplot as plt
# time series - statsmodels
# Seasonality decomposition
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.seasonal import seasonal_decompose
# holt winters
# single exponential smoothing
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
# double and triple exponential smoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [2]:
```
airline = pd.read_csv('C:/Users/user/Downloads/archive (3)/international-airline-
airline = pd.read_csv('C:/Users/user/Downloads/archive (3)/international-airline-
# finding shape of the dataframe
print(airline.shape)
# having a look at the data
print(airline.head())

# plotting the original data
airline['International airline passengers: monthly totals in thousands. Jan 49 ? I
```

```
(145, 1)
         International airline passengers: monthly totals in thousands. Jan 49
? Dec 60
Month
1949-01                                              112.0
1949-02                                              118.0
1949-03                                              132.0
1949-04                                              129.0
1949-05                                              121.0
```

Out[2]: <AxesSubplot:title={'center':'Passengers Data'}, xlabel='Month'>



Fitting the Data with Holt-Winters Exponential Smoothing

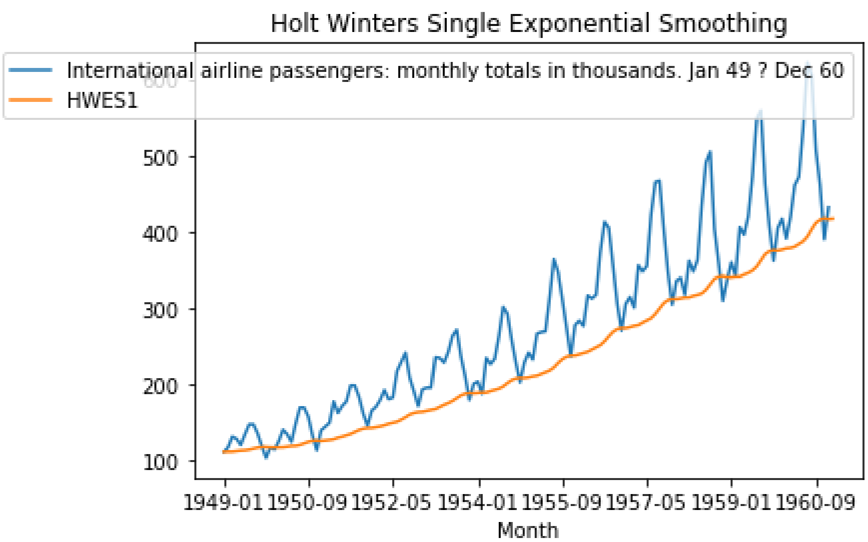
In [3]:
```
# Set the frequency of the date time index as Monthly start as indicated by the da
airline.index.freq = 'MS'
# Set the value of Alpha and define m (Time Period)
m = 12
alpha = 1/(2*m)
```

Single HWES

Now, we will fit the data on the Single Exponential Smoothing,

In [4]:
```
airline['HWES1'] = SimpleExpSmoothing(airline['International airline passengers: 
airline[['International airline passengers: monthly totals in thousands. Jan 49 ?
```

```
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:57
8: ValueWarning: An unsupported index was provided and will be ignored when e.
g. forecasting.
  warnings.warn('An unsupported index was provided and will be'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\holtwinters\model.py:
427: FutureWarning: After 0.13 initialization must be handled at model creation
  warnings.warn(
```

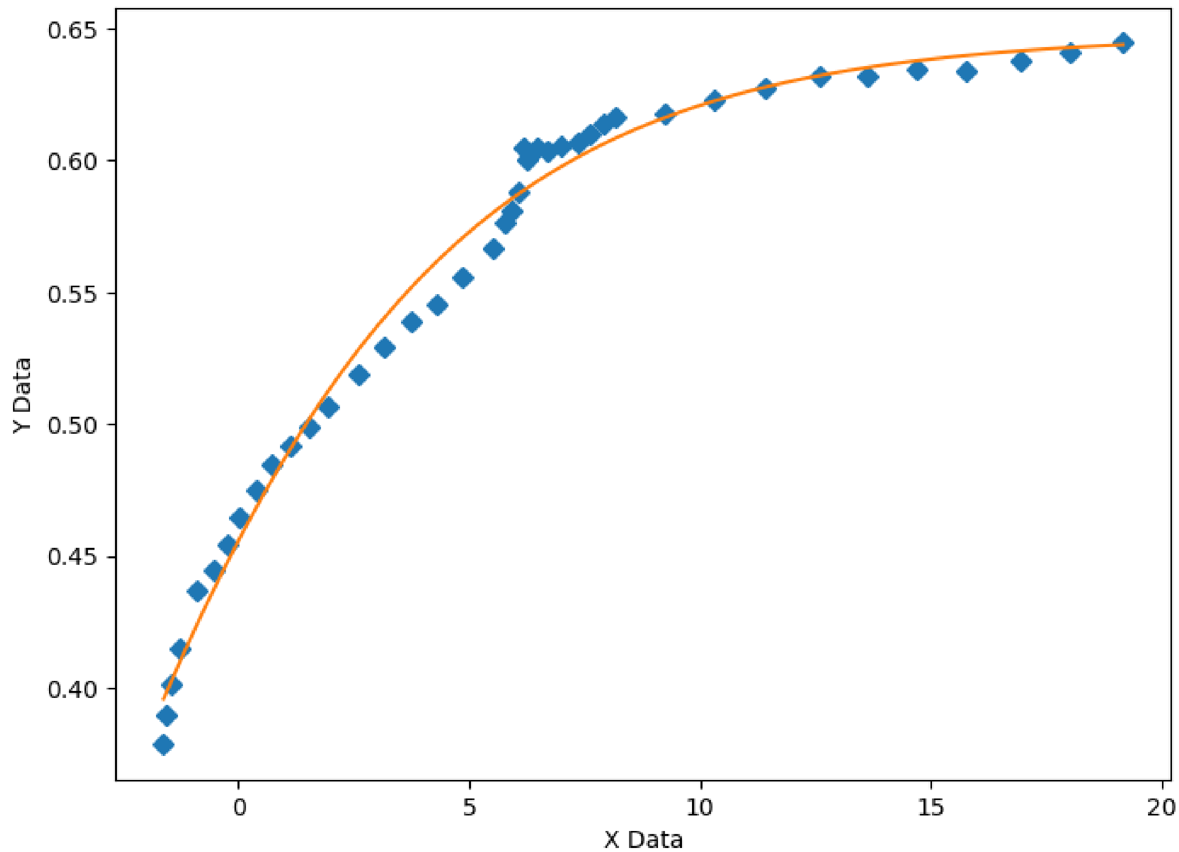


Non Linear Trend Regression

In [5]:
```
import numpy, scipy, matplotlib
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.optimize import differential_evolution
import warnings

xData = numpy.array([19.1647, 18.0189, 16.9550, 15.7683, 14.7044, 13.6269, 12.604
yData = numpy.array([0.644557, 0.641059, 0.637555, 0.634059, 0.634135, 0.631825, 


def func(x, a, b, Offset): # Sigmoid A With Offset from zunzun.com
    return  1.0 / (1.0 + numpy.exp(-a * (x-b))) + Offset


# function for genetic algorithm to minimize (sum of squared error)
def sumOfSquaredError(parameterTuple):
    warnings.filterwarnings("ignore") # do not print warnings by genetic algorith
    val = func(xData, *parameterTuple)
    return numpy.sum((yData - val) ** 2.0)


def generate_Initial_Parameters():
    # min and max used for bounds
    maxX = max(xData)
    minX = min(xData)
    maxY = max(yData)
    minY = min(yData)

    parameterBounds = []
    parameterBounds.append([minX, maxX]) # search bounds for a
    parameterBounds.append([minX, maxX]) # search bounds for b
    parameterBounds.append([0.0, maxY]) # search bounds for Offset

    # "seed" the numpy random number generator for repeatable results
    result = differential_evolution(sumOfSquaredError, parameterBounds, seed=3)
    return result.x

# generate initial parameter values
geneticParameters = generate_Initial_Parameters()

# curve fit the test data
fittedParameters, pcov = curve_fit(func, xData, yData, geneticParameters)

print('Parameters', fittedParameters)

modelPredictions = func(xData, *fittedParameters)

absError = modelPredictions - yData

SE = numpy.square(absError) # squared errors
MSE = numpy.mean(SE) # mean squared errors
RMSE = numpy.sqrt(MSE) # Root Mean Squared Error, RMSE
Rsquared = 1.0 - (numpy.var(absError) / numpy.var(yData))
print('RMSE:', RMSE)
print('R-squared:', Rsquared)
```

```
##########################################################
# graphics output section
def ModelAndScatterPlot(graphWidth, graphHeight):
    f = plt.figure(figsize=(graphWidth/100.0, graphHeight/100.0), dpi=100)
    axes = f.add_subplot(111)

    # first the raw data as a scatter plot
    axes.plot(xData, yData,  'D')

    # create data for the fitted equation plot
    xModel = numpy.linspace(min(xData), max(xData))
    yModel = func(xModel, *fittedParameters)

    # now the model as a line plot
    axes.plot(xModel, yModel)

    axes.set_xlabel('X Data') # X axis data label
    axes.set_ylabel('Y Data') # Y axis data label

    plt.show()
    plt.close('all') # clean up after using pyplot

graphWidth = 800
graphHeight = 600
ModelAndScatterPlot(graphWidth, graphHeight)
```

```
Parameters [ 0.21540306 -6.67449153 -0.35241296]
RMSE: 0.008428738373451258
R-squared: 0.9886222631484034
```



Seasonality for the Time Series Analysis

In [6]:

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

# reading the dataset using read_csv
df = pd.read_csv("C:/Users/user/Downloads/Dataset-main/Dataset-main/stock_data.cs
                 parse_dates=True,
                 index_col="Date")

# displaying the first five rows of dataset
df.head()
```

Out[6]:

Date	Unnamed: 0	Open	High	Low	Close	Volume	Name
2006-01-03	NaN	39.69	41.22	38.79	40.91	24232729	AABA
2006-01-04	NaN	41.22	41.90	40.77	40.97	20553479	AABA
2006-01-05	NaN	40.93	41.73	40.85	41.53	12829610	AABA
2006-01-06	NaN	42.88	43.57	42.80	43.21	29422828	AABA
2006-01-09	NaN	43.10	43.66	42.82	43.42	16268338	AABA

In [7]:
```
# deleting column
df.drop(columns='Unnamed: 0')
```

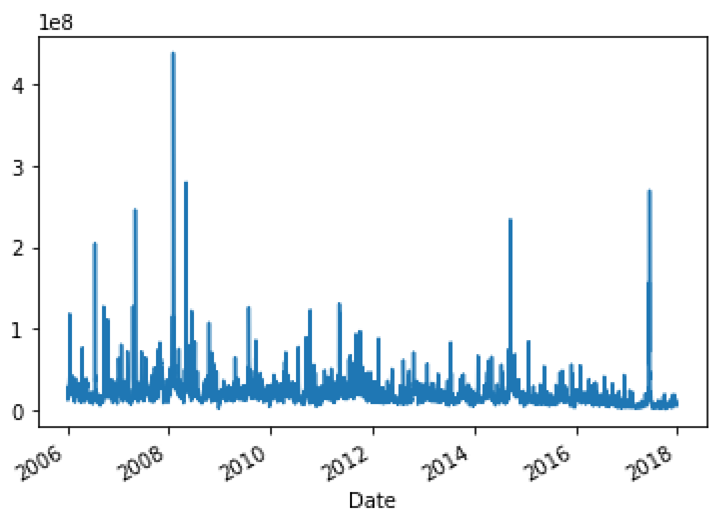
Out[7]:

	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
...	...	...	...	...	...	...
2017-12-22	71.42	71.87	71.22	71.58	10979165	AABA
2017-12-26	70.94	71.39	69.63	69.86	8542802	AABA
2017-12-27	69.77	70.49	69.69	70.06	6345124	AABA
2017-12-28	70.12	70.32	69.51	69.82	7556877	AABA
2017-12-29	69.79	70.13	69.43	69.85	6613070	AABA

3019 rows × 6 columns

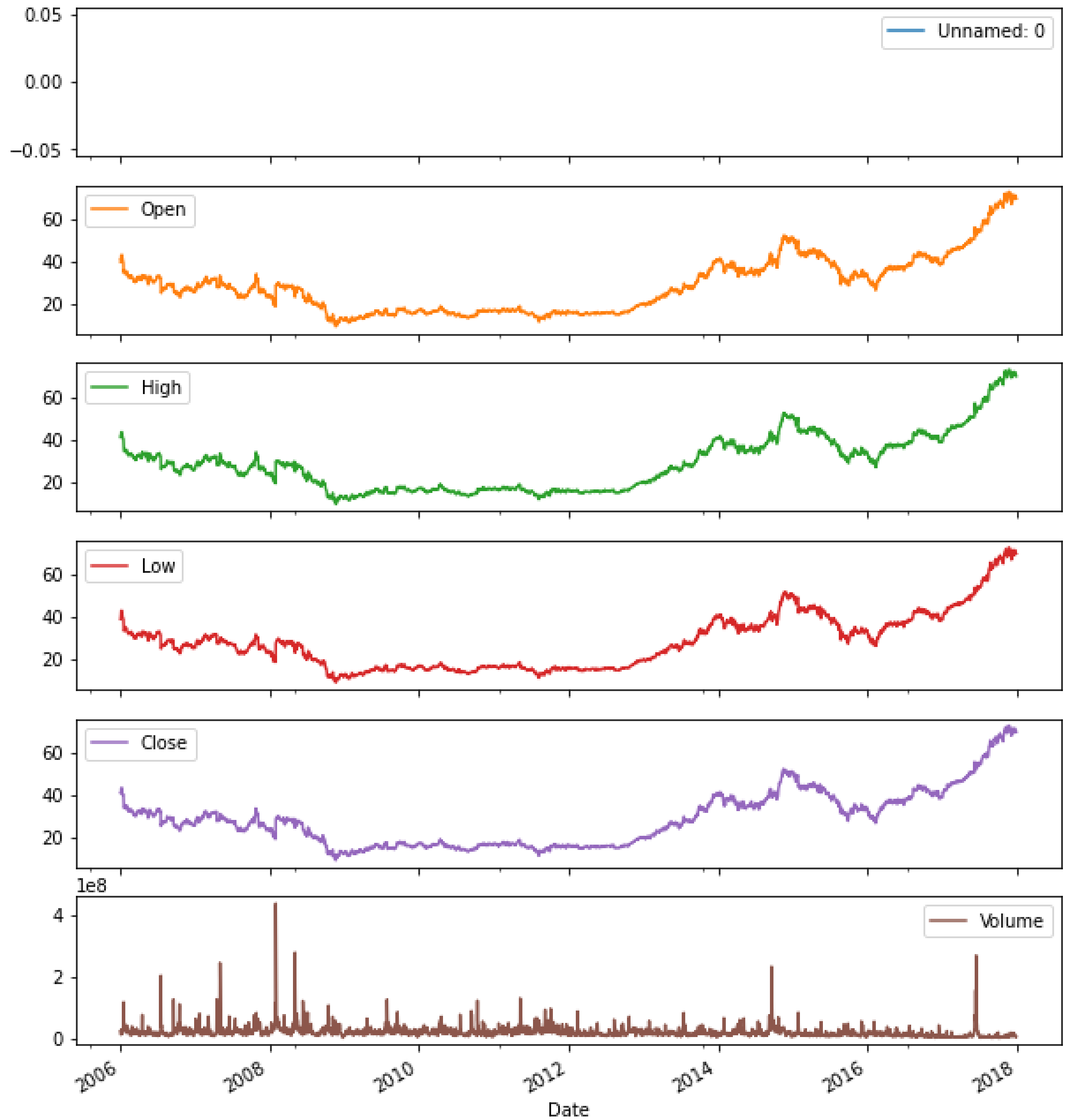
In [8]:
```
df['Volume'].plot()
```

Out[8]: <AxesSubplot:xlabel='Date'>



In [9]:
```
df.plot(subplots=True, figsize=(10, 12))
```

Out[9]:
```
array([<AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>,
       <AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>,
       <AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>],
      dtype=object)
```



The line plots used above are good for showing seasonality.

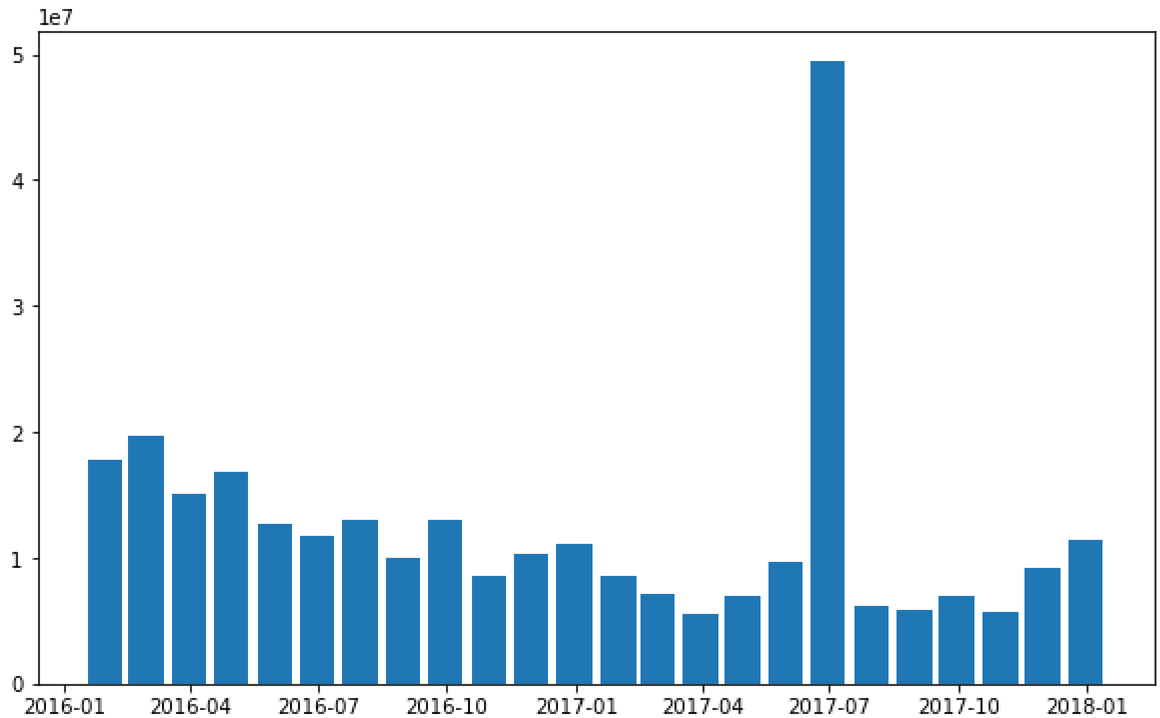
Seasonality:

In time-series data, seasonality is the presence of variations that occur at specific regular time intervals less than a year, such as weekly, monthly, or quarterly.

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.

In [10]:
```
# Resampling the time series data based on monthly 'M' frequency
df_month = df.resample("M").mean()

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

# plotting bar graph
ax.bar(df_month['2016':].index,
    df_month.loc['2016':, "Volume"],
    width=25, align='center')
```

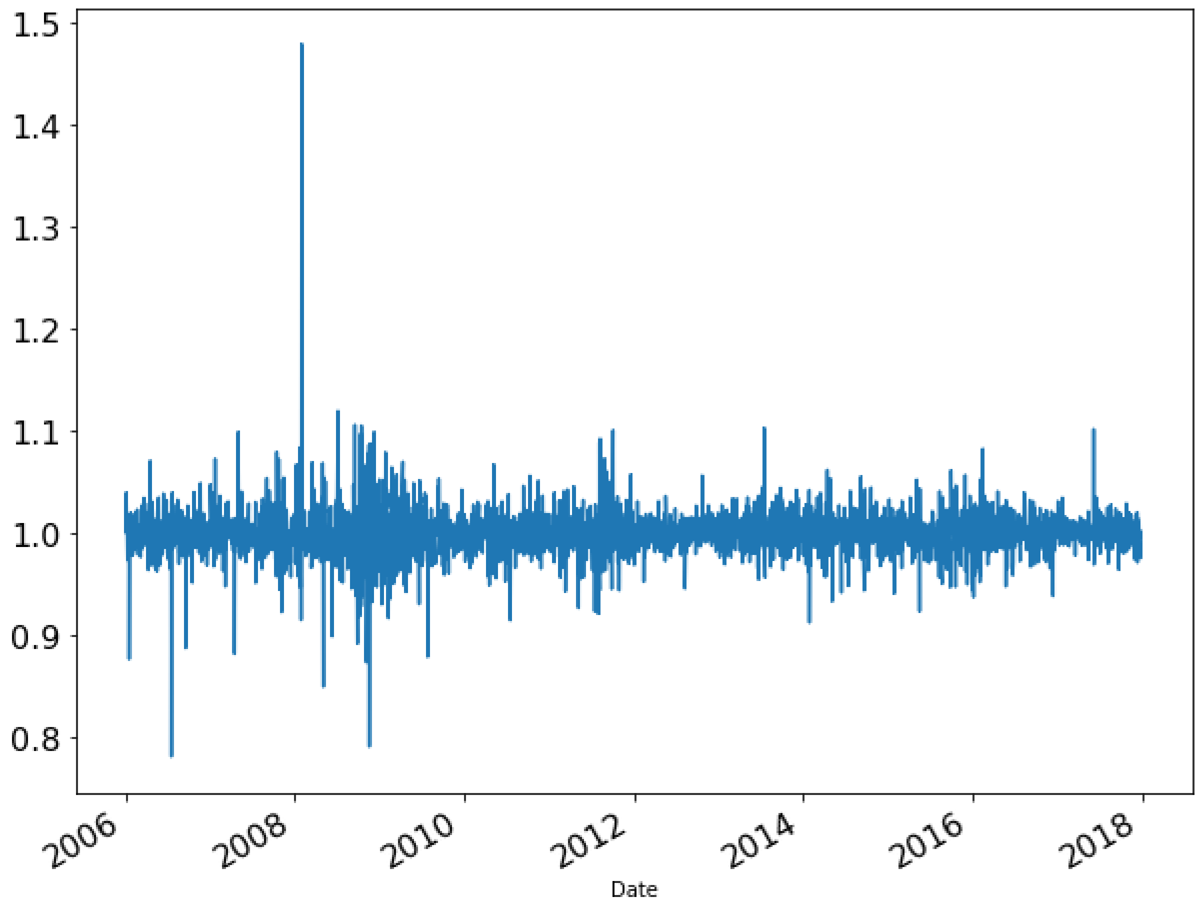
Out[10]: <BarContainer object of 24 artists>



In [12]:

```
#Plotting Chages in the data

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

Out[12]: <AxesSubplot:xlabel='Date'>



In [13]:
```
df['2017']['Change'].plot(figsize=(10, 6))
```

Out[13]: <AxesSubplot:xlabel='Date'>