TIME SERIES ANALYSIS PROJECTS (SEASONAL AND NON-SEASONAL)

-SHIVANI MOGILI
(CWID-10473465)

Seasonal Data – Predicting the Furniture Sales for the brand Superstore

1. Data Set Description – The Superstore data set has 1000 rows and 22 columns. The Rows contain- Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer Name, Segment, Country, City, State, Postal Code, Region, Product ID, Category, Sub-Category, Product Name, Sales, Quantity, Discount, Profit. Considering only furniture sales data.

In [1]:
```python
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```

```python
df = pd.read_excel("Superstore.xls")
df.head()
```

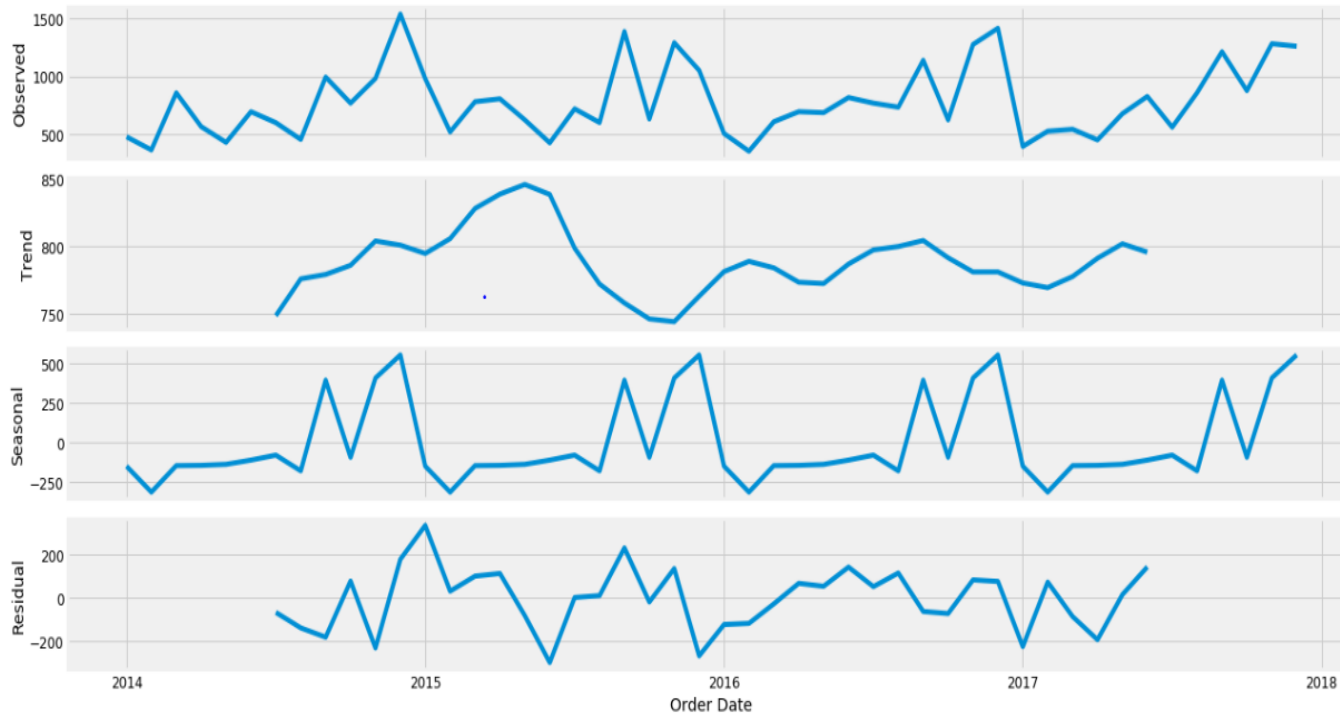
	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2

5 rows × 21 columns

```
y.plot(figsize=(15, 6))
plt.show()
```

2. Data Pre-processing

```
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```



In Data -preprocessing removing the unnecessary columns removed missing values. Also used Indexing as it is important in sales or any kind of data with a date to index the Date Column. Make sure the date format is in (yyyy-mm-dd). Checking if the data is seasonal. The plot clearly shows the seasonality.

3. BOX-JENKIN MODELS (ARIMA, SARIMAX)

```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
results = mod.fit()
print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

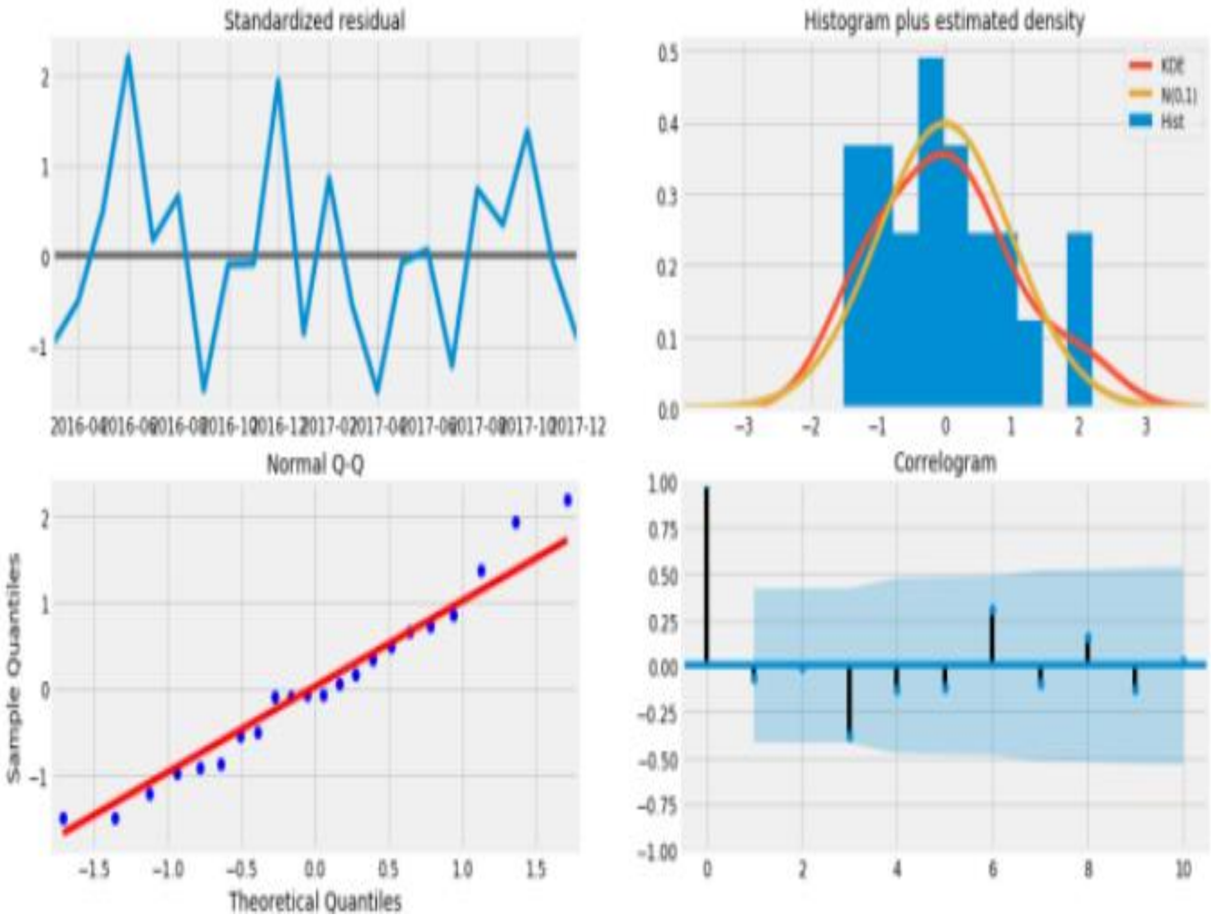
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

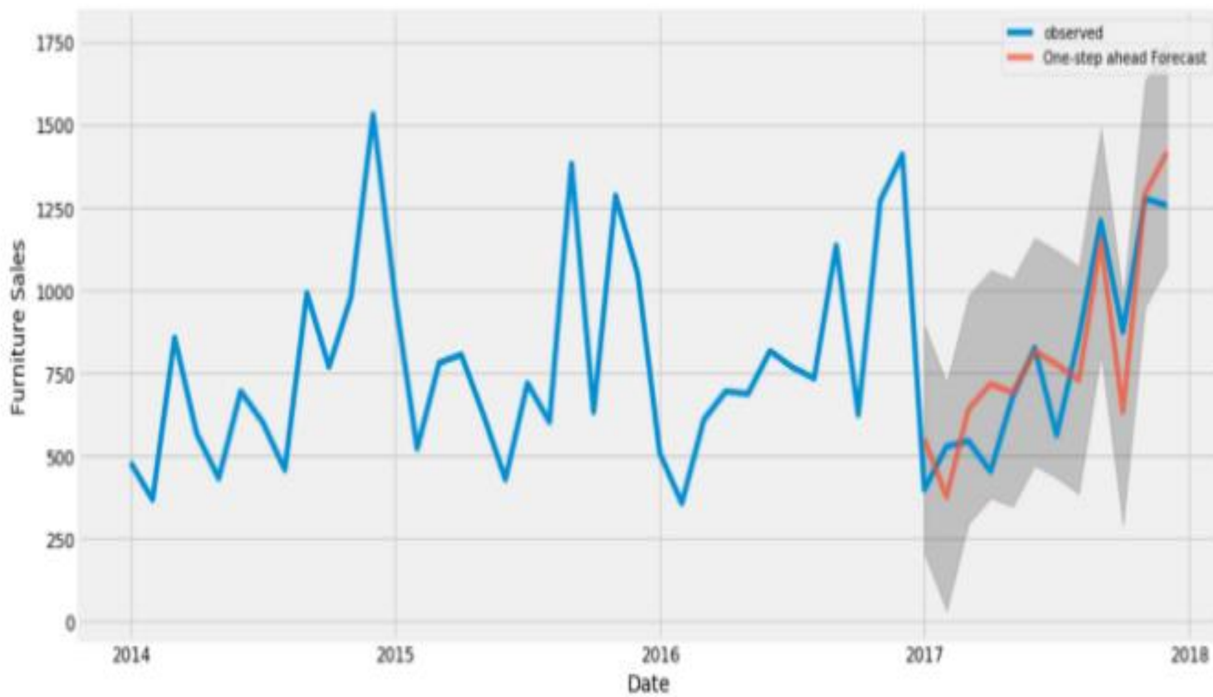
```python
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
results = mod.fit()
print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

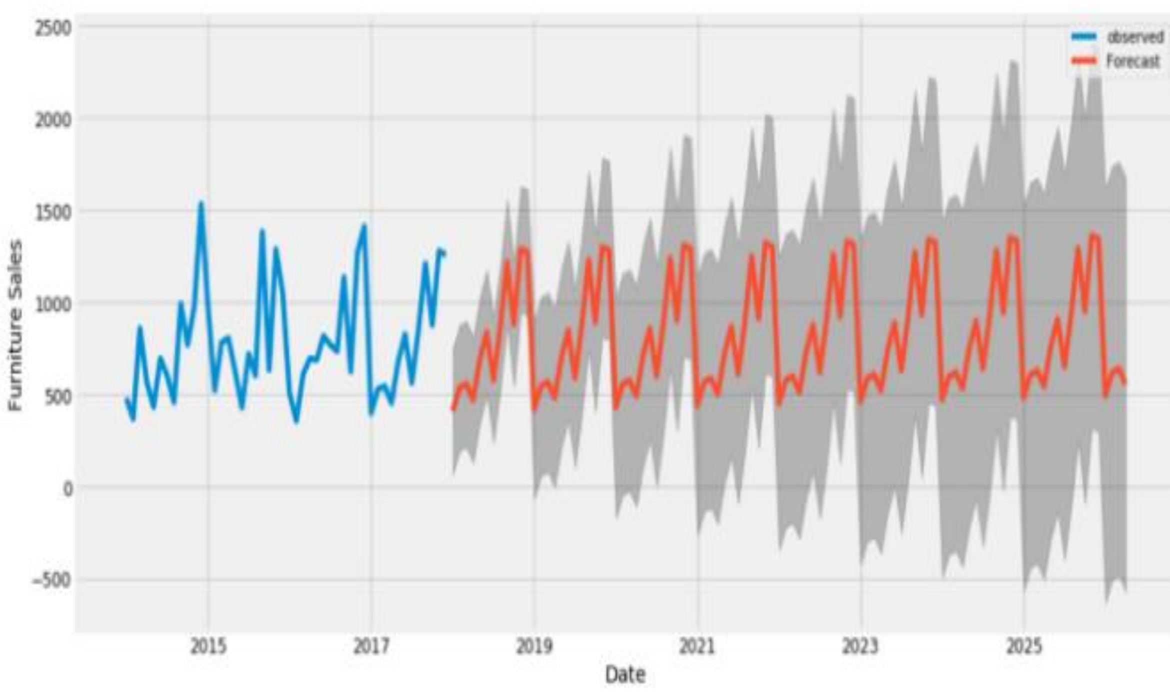
```
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:1131.2657078645939
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:497.23144334183365
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:1001.3915524374769
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:318.0047199116341
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:720.9252270758095
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:2876.7174897071977
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:466.56074298091255
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:499.54290594685824
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:2461.517421827548
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:319.98848769468657
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:1287.5697512865586
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:497.78896630044073
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:1388.8924232046936
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:319.7714068109211
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:649.9056176816999
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:3307.7208814993064
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:458.8705548482932
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:486.18329774427826
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:2625.602326434297
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:310.75743684172994
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:692.1645522067712
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:1399.3709974017943
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:479.46321478521355
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:480.92593679352177
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:1431.0752736869172
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:304.4664675084554
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:665.779444218685
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:246116.34689777798
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:468.3685195814987
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:482.5763323876739
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:3365796.8535189056
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:306.0156002122138
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:671.2513547541902
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:1393.2157168383435
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:479.2003422281134
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:475.34036587848493
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:2102.468501404909
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:300.6270901345443
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:649.0318019835024
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:2603.9208285600357
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:460.4762687610111
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:469.52503546608614
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:2586.7750340396897
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:297.7875439553055
```

```python
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0146      0.342      0.043      0.966      -0.655       0.684
ma.L1         -1.0000      0.360     -2.781      0.005      -1.705      -0.295
ar.S.L12      -0.0253      0.042     -0.609      0.543      -0.107       0.056
sigma2      2.958e+04   1.22e-05   2.43e+09      0.000    2.96e+04    2.96e+04
==============================================================================
```


Standardized residual
2016-04
2016-06
2016-08
2016-10
2016-12
2017-02
2017-04
2017-06
2017-08
2017-10
2017-12
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram


observed
One-step ahead Forecast
Furniture Sales
Date
1750
1500
1250
1000
750
500
250
0
2014
2015
2016
2017
2018


observed
Forecast
Furniture Sales
Date
2500
2000
1500
1000
500
0
-500
2015
2017
2019
2021
2023
2025

Hence using the forecasting model ARIMA model to check the seasonality trend. The output SARIMAX(1,1,1)* (1,1,0,12) yields the lowest and the AIC value is 297.78 . The SARIMAX Model fits the best to this data set. The observed data and forecast data are almost similar. Hence this model can be used to forecast the future.

4. STATISTICAL OBSERVATIONS-

If we observe the Standardized residual and Q-Q plot it helps us to understand the theoretical quantiles and sample quantiles by observing the correlation between the attributes.

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.0146	0.342	0.043	0.966	-0.655	0.684
ma.L1	-1.0000	0.360	-2.781	0.005	-1.705	-0.295
ar.S.L12	-0.0253	0.042	-0.609	0.543	-0.107	0.056
sigma2	2.958e+04	1.22e-05	2.43e+09	0.000	2.96e+04	2.96e+04

5. CONCLUSION-

Predictions show that the time series is expected to continue to grow steadily. The more naturally we lose confidence in our values. This is reflected in the confidence intervals generated by the model. Confidence intervals will increase as you progress into the future. If given more time I could have tried to forecast for different attributes and see the seasonality between them. Also, I could have changed the date of dynamic forecasts to see how this affects the overall quality of your forecasts.

NON-SEASONAL DATA SET –

1. DATA-SET

LINK- https://finance.yahoo.com/quote/AAPL?p=AAPL&.tsrc=fin-srch . Used Data Reader to extract APPLE stock data from the yahoo finance website. The Data set contains 4 Columns i.e., High, Low, Close, Volume, and Adj Close. Dropped the Adj close column. [Date indicates the date of trading, Open indicates the price at which security first trades, High indicates the highest price of the trading day, Low indicates the lowest price of the trading day, Close indicates the last price the stock traded during the trading day. Adj Close indicates the price that adjusts cooperate actions on the closing price.] . It is important to understand every column in stock data as it helps us to understand and select the feature columns in better way and understand the correlation between the attributes.

Also need to check the date and high for which the correlation and forecasting we are calculating should be in the same dimension and same data type. Storing the file as a .csv file through python and reading the data again to put it in a data frame for ease of calculations. ACF and PACF are plotted. Took the data from 2013 to 2019 to a new data frame and plotted Q-Q Plot.

In [1]:
```
!pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in c:\users\shiva\anaconda3\lib\site-packages (0.10.0)
Requirement already satisfied: requests>=2.19.0 in c:\users\shiva\anaconda3\lib\site-packages (from pandas-datareader) (2.2
4.0)
Requirement already satisfied: pandas>=0.23 in c:\users\shiva\anaconda3\lib\site-packages (from pandas-datareader) (1.1.3)
Requirement already satisfied: lxml in c:\users\shiva\anaconda3\lib\site-packages (from pandas-datareader) (4.6.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\shiva\anaconda3\lib\site-packages (from pandas>=0.23->pandas-datare
ader) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\shiva\anaconda3\lib\site-packages (from pandas>=0.23->pan
das-datareader) (2.8.1)
Requirement already satisfied: numpy>=1.15.4 in c:\users\shiva\anaconda3\lib\site-packages (from pandas>=0.23->pandas-datar
eader) (1.19.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shiva\anaconda3\lib\site-packages (from requests>=2.19.0->pan
das-datareader) (2020.6.20)
Requirement already satisfied: idna<3,>=2.5 in c:\users\shiva\anaconda3\lib\site-packages (from requests>=2.19.0->pandas-da
tareader) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\shiva\anaconda3\lib\site-packages (from
requests>=2.19.0->pandas-datareader) (1.25.11)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\shiva\anaconda3\lib\site-packages (from requests>=2.19.0->pand
as-datareader) (3.0.4)
Requirement already satisfied: six>=1.5 in c:\users\shiva\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas>
=0.23->pandas-datareader) (1.15.0)
```

```
WARNING: You are using pip version 21.3.1; however, version 22.1 is available.
You should consider upgrading via the 'c:\users\shiva\anaconda3\python.exe -m pip install --upgrade pip' command.
```

In [98]:
```
import pandas_datareader as pdr
import pandas as pd
from datetime import datetime
```

In [150]:
```
df_apple=pdr.get_data_yahoo('AAPL')
```

In [151]:
```
df_apple.tail()
```

Out[151]:

High	Low	Open	Close	Volume	Adj Close

Plotting the time series for High column vs the Date column.

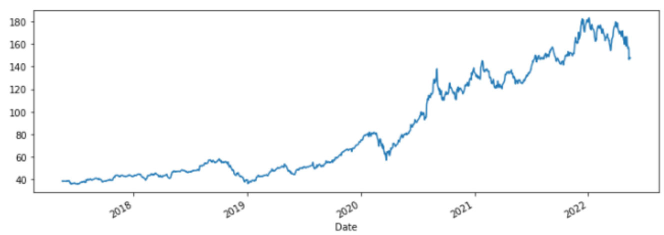
```
In [151]:  df_apple.tail()
```

Out[151]:

Date	High	Low	Open	Close	Volume	Adj Close
2022-05-10	156.740005	152.929993	155.520004	154.509995	115366700.0	154.509995
2022-05-11	155.449997	145.809998	153.500000	146.500000	142689800.0	146.500000
2022-05-12	146.199997	138.800003	142.770004	142.559998	182602000.0	142.559998
2022-05-13	148.100006	143.110001	144.589996	147.110001	113787000.0	147.110001
2022-05-16	147.519897	144.190002	145.550003	145.539993	86452429.0	145.539993

```
In [152]:  df_apple['High'].plot(figsize=(12,4))
```

```
Out[152]: <AxesSubplot:xlabel='Date'>
```



2. DATA PREPROCESSING –

```
In [153]: df_apple.index

Out[153]: DatetimeIndex(['2017-05-17', '2017-05-18', '2017-05-19', '2017-05-22',
                         '2017-05-23', '2017-05-24', '2017-05-25', '2017-05-26',
                         '2017-05-30', '2017-05-31',
                         ...
                         '2022-05-03', '2022-05-04', '2022-05-05', '2022-05-06',
                         '2022-05-09', '2022-05-10', '2022-05-11', '2022-05-12',
                         '2022-05-13', '2022-05-16'],
                        dtype='datetime64[ns]', name='Date', length=1259, freq=None)
```

```
In [154]: index=df_apple.loc['2021-01-01':'2022-01-01'].index
          share_open=df_apple.loc['2021-01-01':'2022-01-01']['Open']
```

```
In [155]: share_open

Out[155]: Date
          2021-01-04    133.520004
          2021-01-05    128.889999
          2021-01-06    127.720001
          2021-01-07    128.360001
          2021-01-08    132.429993
                           ...
          2021-12-27    177.089996
          2021-12-28    180.160004
          2021-12-29    179.330002
          2021-12-30    179.470001
          2021-12-31    178.089996
          Name: Open, Length: 252, dtype: float64
```

Used Date time Index which can be boxed to Timestamp objects.

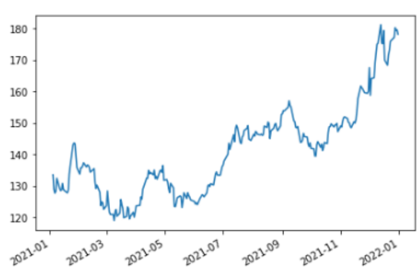
```
In [156]: index

Out[156]: DatetimeIndex(['2021-01-04', '2021-01-05', '2021-01-06', '2021-01-07',
                         '2021-01-08', '2021-01-11', '2021-01-12', '2021-01-13',
                         '2021-01-14', '2021-01-15',
                         ...
                         '2021-12-17', '2021-12-20', '2021-12-21', '2021-12-22',
                         '2021-12-23', '2021-12-27', '2021-12-28', '2021-12-29',
                         '2021-12-30', '2021-12-31'],
                        dtype='datetime64[ns]', name='Date', length=252, freq=None)
```

```
In [157]: import matplotlib.pyplot as plt
          %matplotlib inline
```

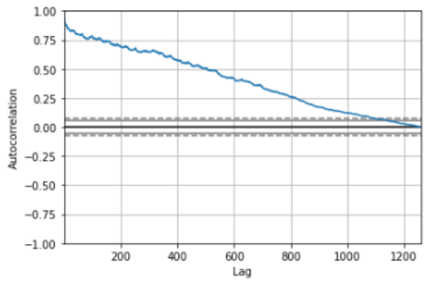
```
In [158]: figure,axis=plt.subplots()
          plt.tight_layout()
          ## Preventing overlapping
          figure.autofmt_xdate()
          axis.plot(index,share_open)

Out[158]: [<matplotlib.lines.Line2D at 0x1e5599e73d0>]
```



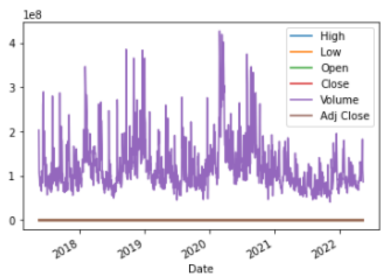
In [164]:

```
from pandas import read_csv
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(df_apple)
pyplot.show()
```



In [165]:

```
df_apple.plot()
pyplot.show()
```

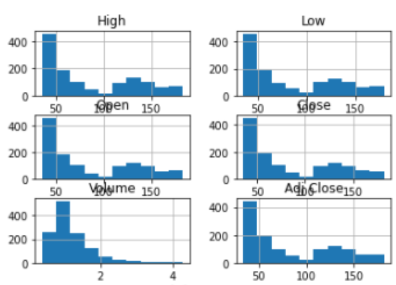


In [166]:

```
df_apple.hist()
```

Out[166]:

```
array([[<AxesSubplot:title={'center':'High'}>,
        <AxesSubplot:title={'center':'Low'}>],
       [<AxesSubplot:title={'center':'Open'}>,
        <AxesSubplot:title={'center':'Close'}>],
       [<AxesSubplot:title={'center':'Volume'}>,
        <AxesSubplot:title={'center':'Adj Close'}>]], dtype=object)
```



In [167]:

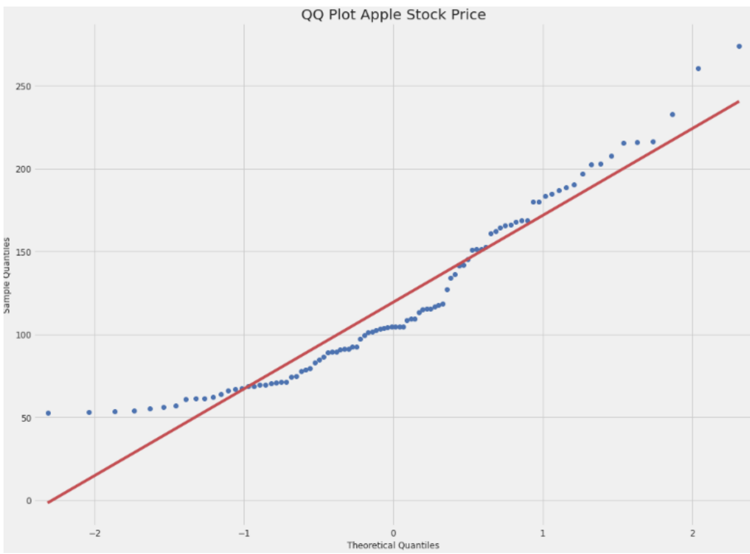
```
High = df_apple['High']
High.head(20)
```

Out[167]:

```
Date
2017-05-17    38.642502
2017-05-18    38.334999
2017-05-19    38.494999
2017-05-22    38.645000
2017-05-23    38.724998
2017-05-24    38.542500
2017-05-25    38.587502
2017-05-26    38.560001
2017-05-30    38.607498
2017-05-31    38.542500
2017-06-01    38.332500
```

```
ADF Statistic: -6.501865
p-value: 0.000000
Critical Values:
        1%: -3.502
        5%: -2.893
        10%: -2.583
Reject Null Hypothesis(Ho)-Time Series is Stationary
```

I checked the stationary of the Time- series using Augmented Dickey-Fuller (ADF) test. we know that for the stationary we must have constant mean, constant variance, and trend.



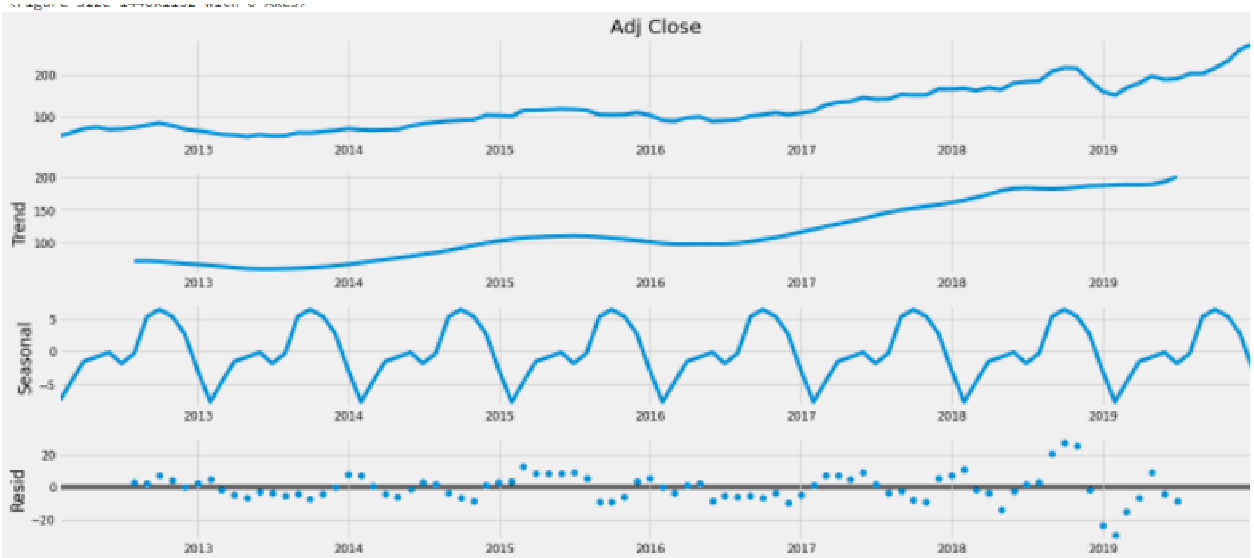
QQ plot Inference:-

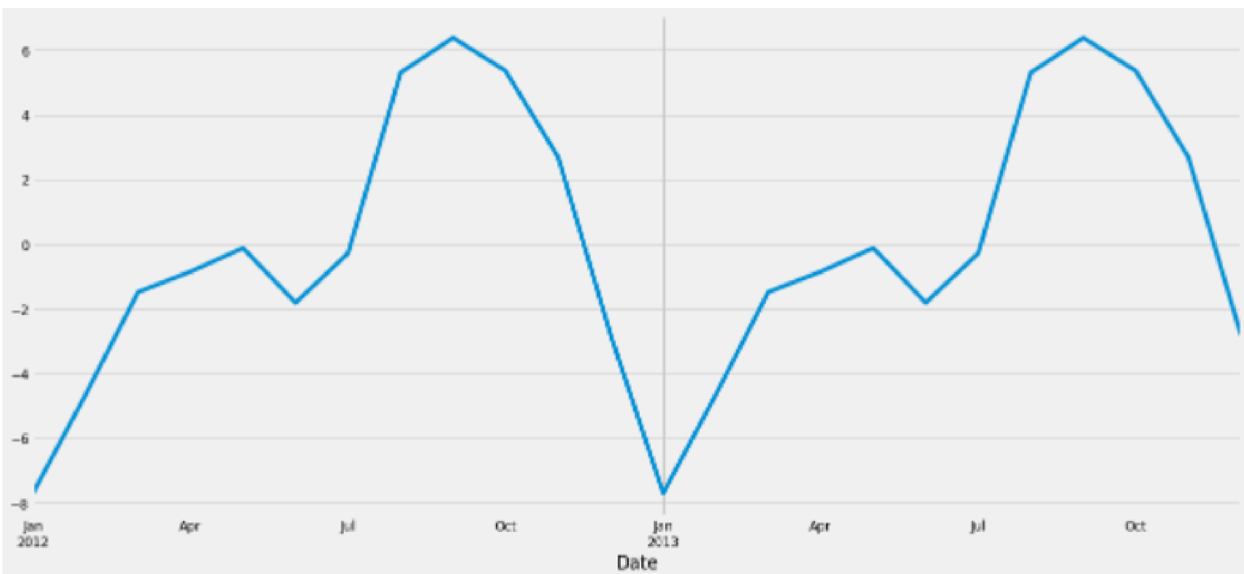
Heavy-Tailed Distribution-Curve at Extremities. Shows the extent of both right and left skews.

Shows Distribution is Not following Gaussian Normal Distribution.

```
rcParams['figure.figsize'] = 18, 8
plt.figure(figsize=(20,16))
decomposed_series = sd(monthly_data['Adj Close'],model='additive',freq=12)
decomposed_series.plot()
plt.show()
```

<Figure size 1440x1152 with 0 Axes>





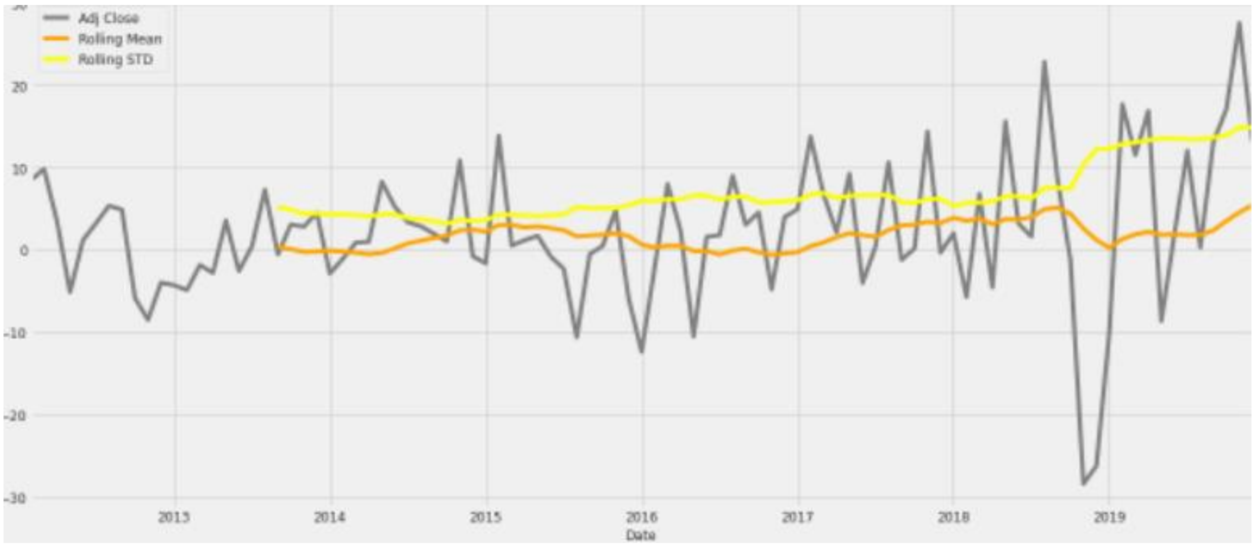
```python
##ADF Test-Statsmodels Library

def ad_fuller_func(X):
  result_ad_fuller = adfuller(X)
  print('ADF Statistic: %f' % result_ad_fuller[0])
  print('p-value: %f' %result_ad_fuller[1])
  print('Critical Values:')
  for key, value in result_ad_fuller[4].items():
          print('\t%s: %.3f' % (key, value))

  if result_ad_fuller[0] < result_ad_fuller[4]['5%']:
    print('Reject Null Hypothesis(Ho)-Time Series is Stationary')
  else:
    print('Failed to Reject Ho-Time Series is Non-Stationary')
```

```python
ad_fuller_func(monthly_data['Adj Close'])
```

```
ADF Statistic: 1.339253
p-value: 0.996820
Critical Values:
	1%: -3.504
	5%: -2.894
	10%: -2.584
```



ARIMA MODEL

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time-series data to either better understand the data set or to predict future trends. ACF and PACF that series has kind of stationery we conducted ADF Test and carry out Grid Search. Parameter for Series indicate non-seasonality part order (1,1,1) . ARIMA in which Auto-Regressive (1) and Moving Average (1) are derived by ACF Plot which is differencing (1) derived by difference and observing stationarity. By observing the lowest AIC and seasonality order (2,2,0) and non-seasonal component (1,1,1) as shown by correlograms.

```python
list_param = []
list_param_seasonal=[]
list_results_aic=[]

for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            model = sm.tsa.statespace.SARIMAX(train,
                                              order=param,
                                              seasonal_order=param_seasonal,
                                              enforce_stationarity=False,
                                              enforce_invertibility=False)

            results = model.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))

            list_param.append(param)
            list_param_seasonal.append(param_seasonal)
            list_results_aic.append(results.aic)
        except:
            continue
```

ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:350.75081385350666

ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:332.11071968501557

ARIMA(1, 1, 1)x(1, 2, 0, 12)12 - AIC:300.4957600928522

ARIMA(1, 1, 1)x(1, 2, 1, 12)12 - AIC:286.2126039361744

ARIMA(1, 1, 1)x(2, 0, 0, 12)12 - AIC:331.740255110838

ARIMA(1, 1, 1)x(2, 0, 1, 12)12 - AIC:333.46473592208514

ARIMA(1, 1, 1)x(2, 0, 2, 12)12 - AIC:324.7832626860535

ARIMA(1, 1, 1)x(2, 1, 0, 12)12 - AIC:262.4409992969335

ARIMA(1, 1, 1)x(2, 1, 1, 12)12 - AIC:256.71390487682834

ARIMA(1, 1, 1)x(2, 2, 0, 12)12 - AIC:206.26186908985358

ARIMA(1, 1, 1)x(2, 2, 1, 12)12 - AIC:206.79066847021136

ARIMA(1, 1, 2)x(0, 0, 0, 12)12 - AIC:459.6835652708871

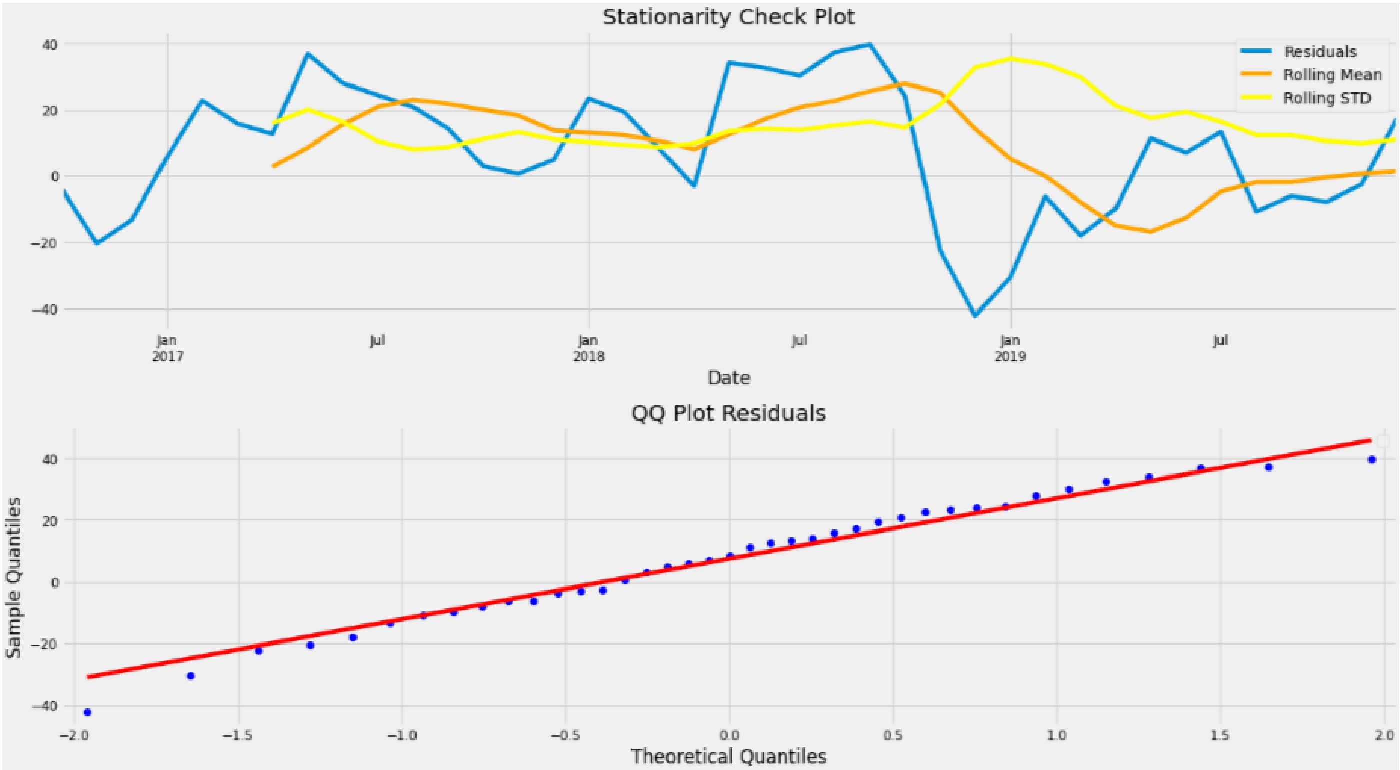
ARIMA(1, 1, 2)x(0, 0, 1, 12)12 - AIC:386.9565978957946

ARIMA(1, 1, 2)x(0, 0, 2, 12)12 - AIC:3937.920519627023

By Observing the Lowest AIC, we come to the Seasonality Order of (2,2,0)12 and the non-seasonal component is (1,1,1) as derived earlier by correlograms. Seasonal Arima is used as we have a seasonality component present. In November, the stock seems to rally on the news of product launches and product releases in that cycle of the year.

```
                                     Statespace Model Results
==========================================================================================
Dep. Variable:                            Adj Close   No. Observations:                 57
Model:             SARIMAX(1, 1, 1)x(2, 2, 0, 12)   Log Likelihood                -120.570
Date:                            Fri, 21 Aug 2020   AIC                            251.139
Time:                                    18:17:51   BIC                            258.468
Sample:                                01-31-2012   HQIC                           253.569
                                     - 09-30-2016
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7677      0.439      1.748      0.081      -0.093       1.629
ma.L1         -0.5014      0.592     -0.848      0.397      -1.661       0.658
ar.S.L12      -0.5420      0.265     -2.045      0.041      -1.062      -0.023
ar.S.L24      -0.3440      0.399     -0.862      0.389      -1.126       0.438
sigma2        92.9283     34.127      2.723      0.006      26.041     159.815
===================================================================================
Ljung-Box (Q):                      nan   Jarque-Bera (JB):                  1.24
Prob(Q):                            nan   Prob(JB):                          0.54
Heteroskedasticity (H):            2.30   Skew:                              0.46
Prob(H) (two-sided):               0.18   Kurtosis:                          2.73
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```



CONCLUSION

The Residual of the model shows the ACF plot which shows the randomness. There is little bias in this modeling. The model indicates the random residuals, and we can say that the series is showing up in a better way. Hence, we can say that ARIMA is better at checking seasonality, and it is also robust in nature.