In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
mydata=pd.read_csv("SBI_Historical_Data.csv")
```

In [3]:
```
mydata
```

Out[3]:

	Date	Price	Open	High	Low	Vol.	Change %
0	Aug 07, 2020	190.65	191.45	192.10	189.55	44.82M	-0.16%
1	Aug 06, 2020	190.95	192.30	194.50	190.25	59.74M	-0.26%
2	Aug 05, 2020	191.45	192.75	196.85	191.00	68.89M	-0.08%
3	Aug 04, 2020	191.60	193.35	193.80	190.50	43.87M	-0.34%
4	Aug 03, 2020	192.25	192.00	194.60	190.05	72.46M	0.42%
...	...	...	...	...	...	...	...
1380	Jan 07, 2015	300.15	300.00	302.55	295.15	15.05M	0.08%
1381	Jan 06, 2015	299.90	310.00	311.10	298.70	15.33M	-4.11%
1382	Jan 05, 2015	312.75	316.25	316.80	312.10	9.14M	-0.79%
1383	Jan 02, 2015	315.25	314.35	318.30	314.35	9.94M	0.40%
1384	Jan 01, 2015	314.00	312.45	315.00	310.70	6.14M	0.69%

1385 rows × 7 columns

In [4]:
```
mydata['Date']= mydata['Date'].apply(lambda x: pd.to_datetime(x))
```

In [5]:
```
1 mydata
```

Out[5]:

	Date	Price	Open	High	Low	Vol.	Change %
0	2020-08-07	190.65	191.45	192.10	189.55	44.82M	-0.16%
1	2020-08-06	190.95	192.30	194.50	190.25	59.74M	-0.26%
2	2020-08-05	191.45	192.75	196.85	191.00	68.89M	-0.08%
3	2020-08-04	191.60	193.35	193.80	190.50	43.87M	-0.34%
4	2020-08-03	192.25	192.00	194.60	190.05	72.46M	0.42%
...	...	...	...	...	...	...	...
1380	2015-01-07	300.15	300.00	302.55	295.15	15.05M	0.08%
1381	2015-01-06	299.90	310.00	311.10	298.70	15.33M	-4.11%
1382	2015-01-05	312.75	316.25	316.80	312.10	9.14M	-0.79%
1383	2015-01-02	315.25	314.35	318.30	314.35	9.94M	0.40%
1384	2015-01-01	314.00	312.45	315.00	310.70	6.14M	0.69%

1385 rows × 7 columns

In [6]:
```
1 # checking basic info
```

In [7]:
```
1 mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1385 entries, 0 to 1384
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      1385 non-null   datetime64[ns]
 1   Price     1385 non-null   float64
 2   Open      1385 non-null   float64
 3   High      1385 non-null   float64
 4   Low       1385 non-null   float64
 5   Vol.      1385 non-null   object
 6   Change %  1385 non-null   object
dtypes: datetime64[ns](1), float64(4), object(2)
memory usage: 75.9+ KB
```

In [8]:
```
1 mydata.describe()
```

Out[8]:

	Price	Open	High	Low
count	1385.000000	1385.000000	1385.000000	1385.000000
mean	266.274404	266.903213	270.343682	262.757906
std	45.555277	45.590664	45.829745	45.248334
min	150.850000	151.950000	153.200000	148.250000
25%	245.650000	245.900000	248.800000	242.550000
50%	270.800000	271.150000	274.600000	267.400000
75%	296.150000	296.500000	300.750000	292.200000
max	372.400000	371.950000	373.800000	366.200000

In [9]:
```
1 mydata.shape
```

Out[9]: (1385, 7)

In [10]:
```
1 mydata.isnull().sum()
```

Out[10]:
```
Date        0
Price       0
Open        0
High        0
Low         0
Vol.        0
Change %    0
dtype: int64
```

In [11]:
```
1 mydata = mydata.sort_values(by=['Date'], ignore_index=True)
```

In [12]:
```
1 mydata
```

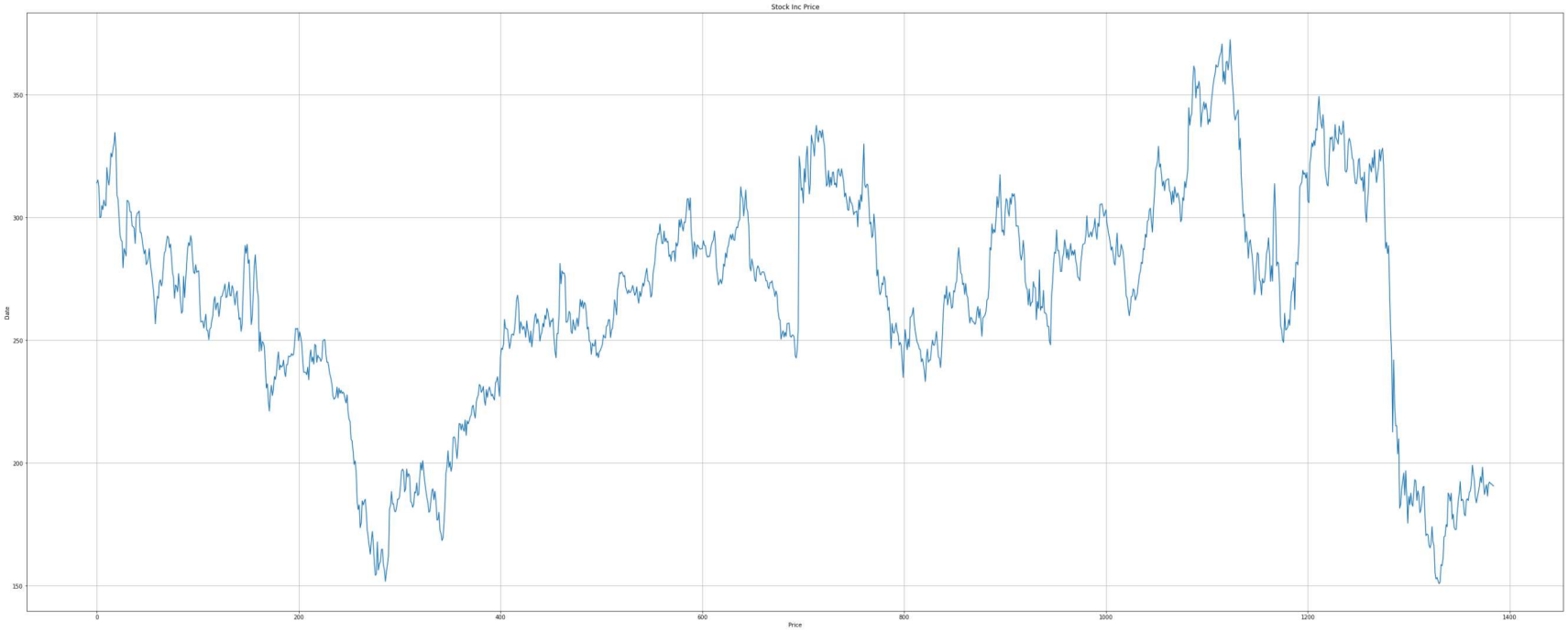
Out[12]:

	Date	Price	Open	High	Low	Vol.	Change %
0	2015-01-01	314.00	312.45	315.00	310.70	6.14M	0.69%
1	2015-01-02	315.25	314.35	318.30	314.35	9.94M	0.40%
2	2015-01-05	312.75	316.25	316.80	312.10	9.14M	-0.79%
3	2015-01-06	299.90	310.00	311.10	298.70	15.33M	-4.11%
4	2015-01-07	300.15	300.00	302.55	295.15	15.05M	0.08%
...	...	...	...	...	...	...	...
1380	2020-08-03	192.25	192.00	194.60	190.05	72.46M	0.42%
1381	2020-08-04	191.60	193.35	193.80	190.50	43.87M	-0.34%
1382	2020-08-05	191.45	192.75	196.85	191.00	68.89M	-0.08%
1383	2020-08-06	190.95	192.30	194.50	190.25	59.74M	-0.26%
1384	2020-08-07	190.65	191.45	192.10	189.55	44.82M	-0.16%

1385 rows × 7 columns

In [13]:
```
1 #  visualize time series data
```

In [14]:
```
plt.figure(figsize=(50,20))
plt.grid(True)
plt.xlabel("Price")
plt.ylabel("Date")
plt.plot(mydata['Price'])
plt.title("Stock Inc Price")
plt.show()
```



In [15]:
```
mydata["s1"]=mydata['Price']-mydata['Price'].shift(1)
```

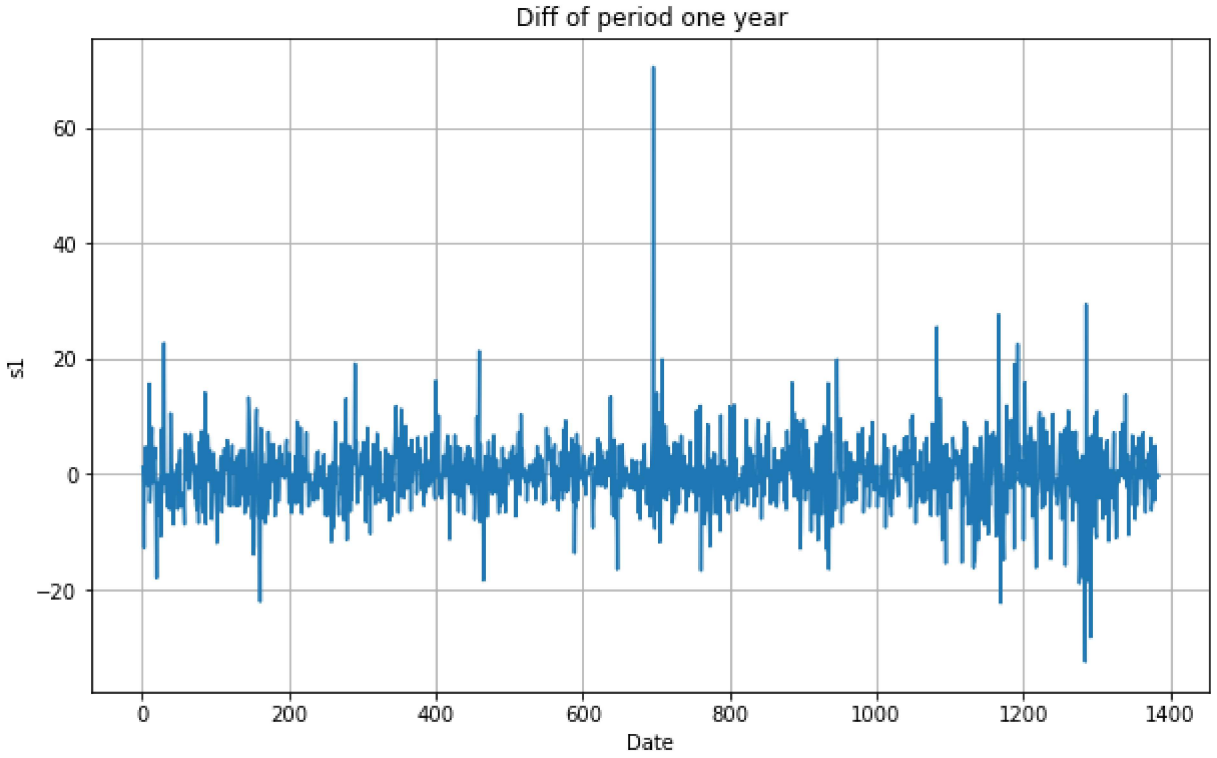
In [16]:
```
1 mydata
```

Out[16]:

	Date	Price	Open	High	Low	Vol.	Change %	s1
0	2015-01-01	314.00	312.45	315.00	310.70	6.14M	0.69%	NaN
1	2015-01-02	315.25	314.35	318.30	314.35	9.94M	0.40%	1.25
2	2015-01-05	312.75	316.25	316.80	312.10	9.14M	-0.79%	-2.50
3	2015-01-06	299.90	310.00	311.10	298.70	15.33M	-4.11%	-12.85
4	2015-01-07	300.15	300.00	302.55	295.15	15.05M	0.08%	0.25
...	...	...	...	...	...	...	...	...
1380	2020-08-03	192.25	192.00	194.60	190.05	72.46M	0.42%	0.80
1381	2020-08-04	191.60	193.35	193.80	190.50	43.87M	-0.34%	-0.65
1382	2020-08-05	191.45	192.75	196.85	191.00	68.89M	-0.08%	-0.15
1383	2020-08-06	190.95	192.30	194.50	190.25	59.74M	-0.26%	-0.50
1384	2020-08-07	190.65	191.45	192.10	189.55	44.82M	-0.16%	-0.30

1385 rows × 8 columns

In [17]:
```
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel("Date")
plt.ylabel("s1")
plt.plot(mydata['s1'])
plt.title("Diff of period one year")
plt.show()
```



In [18]:
```
1 mydata.dropna(inplace=True)
```

In [19]:
```
1 mydata.head()
```

Out[19]:

	Date	Price	Open	High	Low	Vol.	Change %	s1
1	2015-01-02	315.25	314.35	318.30	314.35	9.94M	0.40%	1.25
2	2015-01-05	312.75	316.25	316.80	312.10	9.14M	-0.79%	-2.50
3	2015-01-06	299.90	310.00	311.10	298.70	15.33M	-4.11%	-12.85
4	2015-01-07	300.15	300.00	302.55	295.15	15.05M	0.08%	0.25
5	2015-01-08	304.85	305.00	306.50	302.35	8.94M	1.57%	4.70

In [20]:
```
1 # AD Fuller
```

In [21]:
```
1 from statsmodels.tsa.stattools import adfuller
```

In [22]:
```
1 ADF_test=adfuller(mydata['s1'],autolag='AIC')
```

In [23]:
```
1 print(ADF_test)
```

```
(-13.798233855531922, 8.668623870320859e-26, 6, 1377, {'1%': -3.4351078301822016, '5%': -2.8636412316027577,
'10%': -2.5678886927682663}, 8706.274782330333)
```

In [24]:
```
1 output=pd.DataFrame(ADF_test[0:4],index=["Test Statistics",'p-Value','Lag','number of observation'])
```

In [25]:
```
1 output
```

Out[25]:

	0
Test Statistics	-1.379823e+01
p-Value	8.668624e-26
Lag	6.000000e+00
number of observation	1.377000e+03

In [26]:
```
1 # KPSS test
```

In [27]:
```
1 mydata['s2']=mydata['Price']-mydata['Price'].shift(2)
```

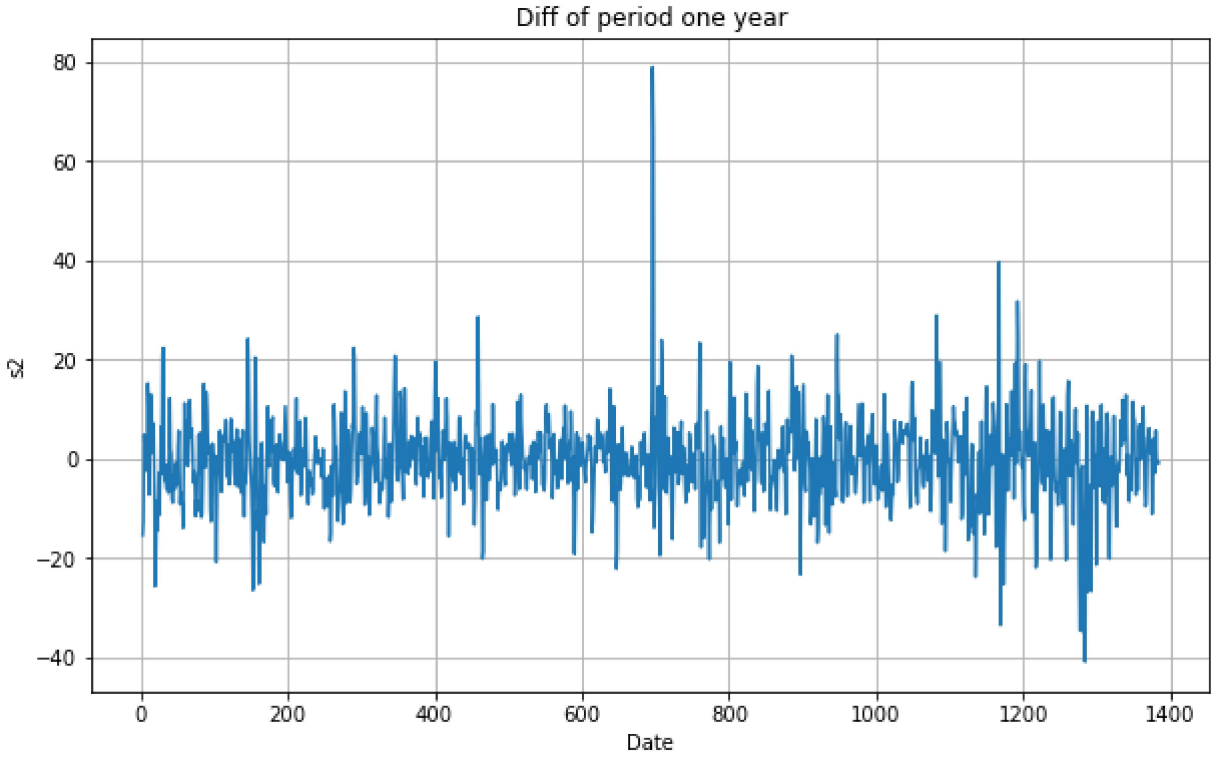
In [28]:
```
1 mydata
```

Out[28]:

	Date	Price	Open	High	Low	Vol.	Change %	s1	s2
1	2015-01-02	315.25	314.35	318.30	314.35	9.94M	0.40%	1.25	NaN
2	2015-01-05	312.75	316.25	316.80	312.10	9.14M	-0.79%	-2.50	NaN
3	2015-01-06	299.90	310.00	311.10	298.70	15.33M	-4.11%	-12.85	-15.35
4	2015-01-07	300.15	300.00	302.55	295.15	15.05M	0.08%	0.25	-12.60
5	2015-01-08	304.85	305.00	306.50	302.35	8.94M	1.57%	4.70	4.95
...	...	...	...	...	...	...	...	...	...
1380	2020-08-03	192.25	192.00	194.60	190.05	72.46M	0.42%	0.80	5.70
1381	2020-08-04	191.60	193.35	193.80	190.50	43.87M	-0.34%	-0.65	0.15
1382	2020-08-05	191.45	192.75	196.85	191.00	68.89M	-0.08%	-0.15	-0.80
1383	2020-08-06	190.95	192.30	194.50	190.25	59.74M	-0.26%	-0.50	-0.65
1384	2020-08-07	190.65	191.45	192.10	189.55	44.82M	-0.16%	-0.30	-0.80

1384 rows × 9 columns

In [29]:
```
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel("Date")
plt.ylabel("s2")
plt.plot(mydata['s2'])
plt.title("Diff of period one year")
plt.show()
```



In [30]:
```
1 mydata.dropna(inplace=True)
```

In [31]:
```
1 mydata.head()
```

Out[31]:

	Date	Price	Open	High	Low	Vol.	Change %	s1	s2
3	2015-01-06	299.90	310.00	311.10	298.70	15.33M	-4.11%	-12.85	-15.35
4	2015-01-07	300.15	300.00	302.55	295.15	15.05M	0.08%	0.25	-12.60
5	2015-01-08	304.85	305.00	306.50	302.35	8.94M	1.57%	4.70	4.95
6	2015-01-09	303.20	306.70	307.85	302.00	11.95M	-0.54%	-1.65	3.05
7	2015-01-12	307.10	304.15	307.80	301.10	8.54M	1.29%	3.90	2.25

In [32]:
```
1 # KPSS test
2 from statsmodels.tsa.stattools import kpss
```

In [33]:
```
1 kpss_test=kpss(mydata['s2'],nlags='auto')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:1910: InterpolationWarning: The test s
tatistic is outside of the range of p-values available in the
look-up table. The actual p-value is greater than the p-value returned.

  warnings.warn(
```

In [34]:
```
1 print(kpss_test)
```

```
(0.08890690301715795, 0.1, 7, {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

In [35]:
```
1 output=pd.DataFrame(kpss_test[0:4],index=['Test Statistics','p-Value','Lag','number of observations'])
```

In [36]:
```
1 output
```

Out[36]:

	0
Test Statistics	0.088907
p-Value	0.1
Lag	7
number of observations	{'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%...

In [37]:
```
1 # perform decompose
```

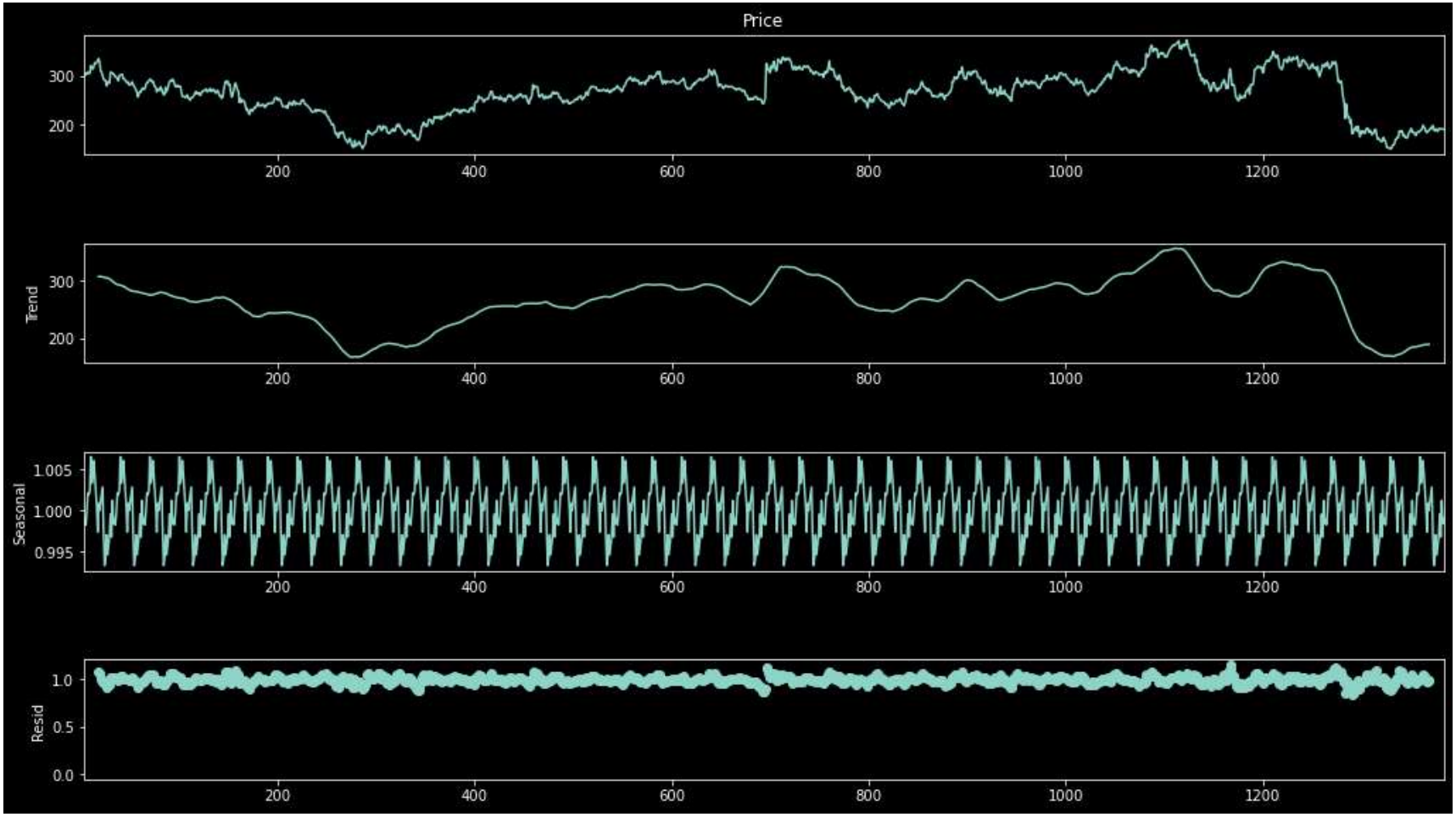
In [38]:
```
1 from statsmodels.tsa.seasonal import seasonal_decompose
```

In [39]:
```
1 plt.style.use('dark_background')
```

In [40]:
```
result = seasonal_decompose(mydata['Price'],model='multiplicative',freq = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16,9)
```

```
<ipython-input-40-4ca968517b92>:1: FutureWarning: the 'freq'' keyword is deprecated, use 'period' instead
  result = seasonal_decompose(mydata['Price'],model='multiplicative',freq = 30)

<Figure size 432x288 with 0 Axes>
```



In [41]:
```
# arima= autoregressive integrated moving average
```

In [42]:
```
mydata.shape
```

Out[42]: (1382, 9)

In [43]:
```
train=mydata['Price'].loc[:1105]
```

In [44]:
```
test=mydata['Price'].loc[1106:]
```

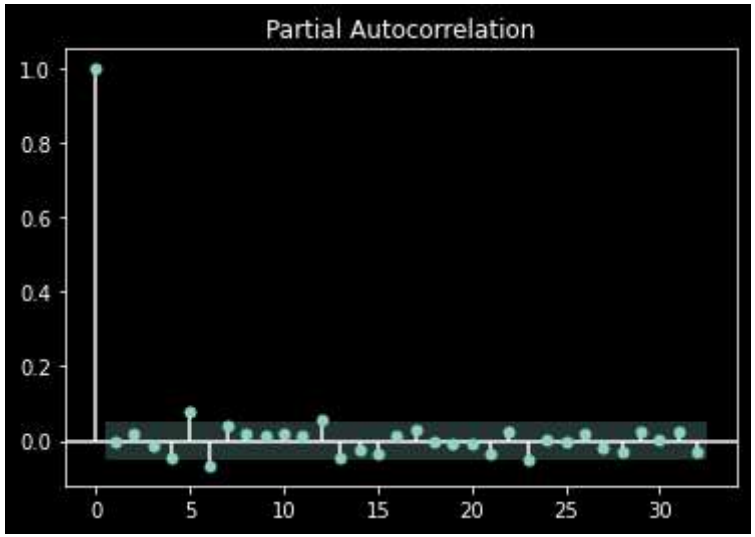
In [45]:
```
test.shape
```

Out[45]: (279,)

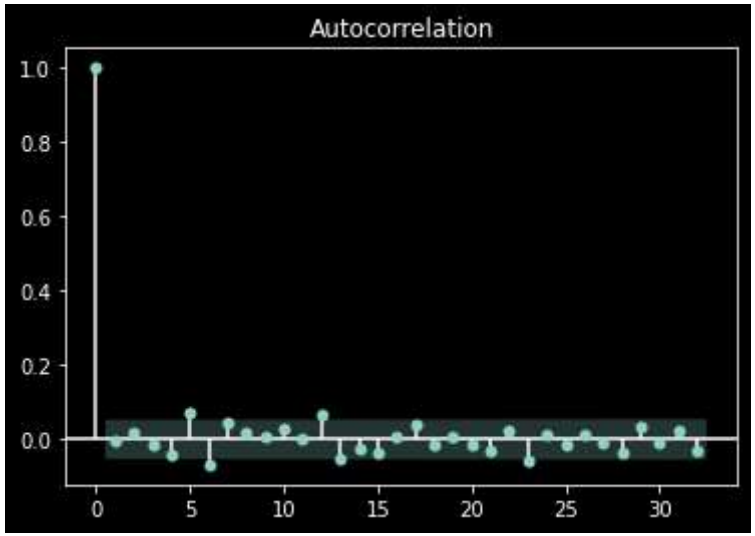
In [46]:
```
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

In [47]:
```
# for identifying p value we have plot pacf
```

In [48]:
```
1 plot_pacf(mydata['s1']);   # p=0 autoregressive
```



In [49]:
```
1 plot_acf(mydata[['s1']]);    # q=0
```



In [50]:
```
model_arima=ARIMA(train,order=(0,1,0))
model_arima_fit=model_arima.fit()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:578: 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:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:578: ValueWarning: An unsupported
index was provided and will be ignored when e.g. forecasting.
  warnings.warn('An unsupported index was provided and will be'
```

In [51]:
```
# validate forecast
```

In [52]:
```
y_pred=model_arima_fit.forecast(steps=279)[2]
```

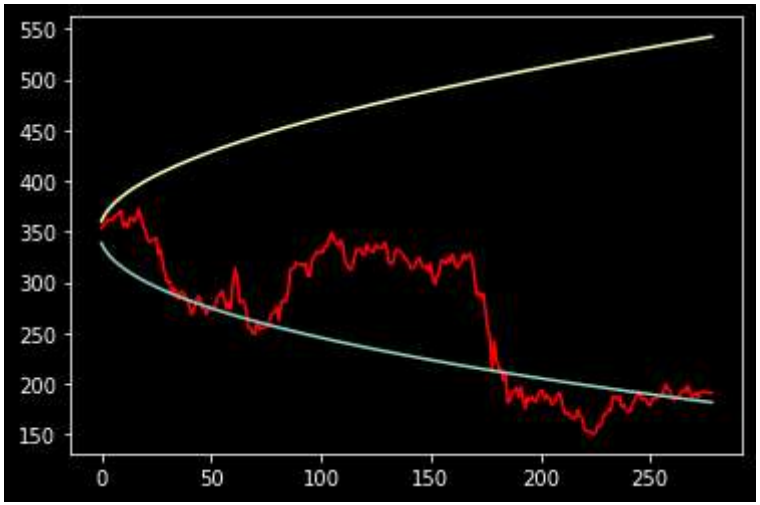
In [53]:
```
1 y_pred
```

Out[53]:
```
array([[338.64606854, 360.24376812],
       [334.21795683, 364.7617165 ],
       [330.83059849, 368.2389115 ],
       [327.98197374, 371.17737291],
       [325.47762944, 373.77155387],
       [323.21783818, 376.12118178],
       [321.14335731, 378.28549931],
       [319.21558697, 380.30310631],
       [317.40771559, 382.20081435],
       [315.70022185, 383.99814476],
       [314.0783687 , 385.70983456],
       [312.53070695, 387.34733297],
       [311.04813165, 388.91974494],
       [309.62326053, 390.43445272],
       [308.25000955, 391.89754036],
       [306.92329411, 393.31409246],
       [305.63881328, 394.68840995],
       [304.39289044, 396.02416945],
       [303.18235332, 397.32454323],
```

In [54]:
```
1 # visualize forecast
```

In [55]:
```
plt.plot(np.array(test),color='red')
plt.plot(np.array(y_pred))
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x286097e6df0>,
 <matplotlib.lines.Line2D at 0x286097e6cd0>]
```



In [56]:
```
from pmdarima.arima import auto_arima
```

In [57]:
```
model=auto_arima(train,start_p=1,start_q=1,test='adf',max_p=3,max_q=3,m=1,d=None,seasonal=False,start_P=0,
          D=0,trace=True,error_action='ignore',suppress_warnings=True,stepwise=True)
```

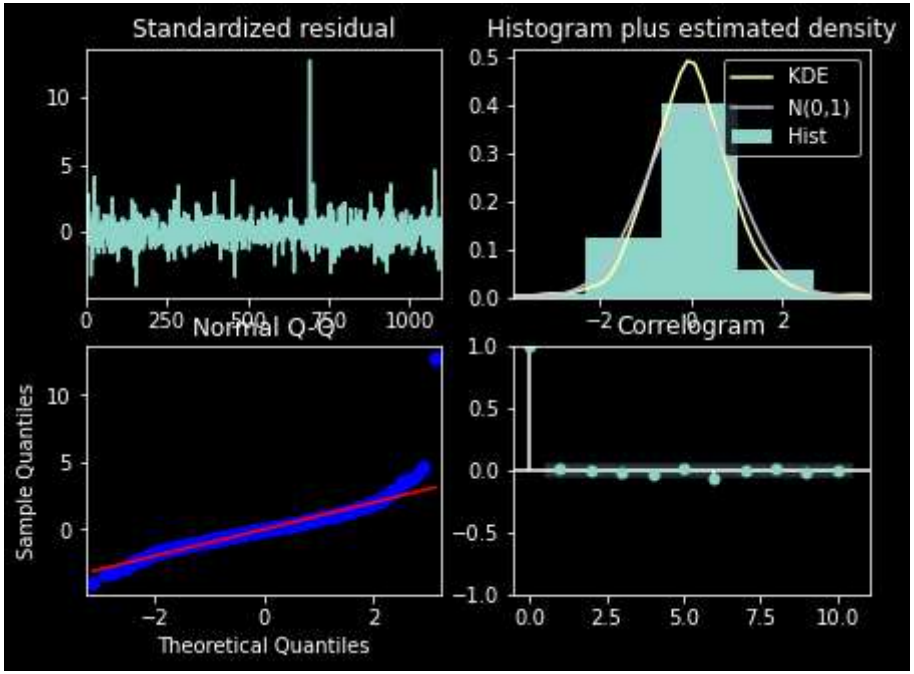
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=6896.429, Time=0.29 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=6892.496, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=6894.391, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=6894.392, Time=0.10 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=6890.570, Time=0.05 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 0.624 seconds
```

In [58]:
```
1 !pip install pmdarima
```

```
Requirement already satisfied: pmdarima in c:\programdata\anaconda3\lib\site-packages (1.8.3)
Requirement already satisfied: pandas>=0.19 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.
2.4)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.
6.2)
Requirement already satisfied: scikit-learn>=0.22 in c:\programdata\anaconda3\lib\site-packages (from pmdarim
a) (0.24.1)
Requirement already satisfied: urllib3 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.26.4)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.
0.1)
Requirement already satisfied: Cython!=0.29.18,>=0.29 in c:\programdata\anaconda3\lib\site-packages (from pmda
rima) (0.29.23)
Requirement already satisfied: numpy>=1.19.3 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.
20.1)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\programdata\anaconda3\lib\site-packages (from
pmdarima) (52.0.0.post20210125)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in c:\programdata\anaconda3\lib\site-packages (from
pmdarima) (0.12.2)
Requirement already satisfied: pytz>=2017.3 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.19->
pmdarima) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anaconda3\lib\site-packages (from pand
as>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=
2.7.3->pandas>=0.19->pmdarima) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit
-learn>=0.22->pmdarima) (2.1.0)
Requirement already satisfied: patsy>=0.5 in c:\programdata\anaconda3\lib\site-packages (from statsmodels!=0.1
2.0,>=0.11->pmdarima) (0.5.1)
```

In [59]:
```
model.plot_diagnostics(figsize=(7,5))
plt.show()
```



In [60]:
```
from sklearn.metrics import mean_squared_error
```

In [61]:
```
1 mse= mean_squared_error(test,y_pred)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-61-0c90f1b31800> in <module>
----> 1 mse= mean_squared_error(test,y_pred)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in mean_squared_error(y_true, y_pre
d, sample_weight, multioutput, squared)
    333     0.825...
    334     """
--> 335     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    336         y_true, y_pred, multioutput)
    337     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in _check_reg_targets(y_true, y_pre
d, multioutput, dtype)
     97
     98     if y_true.shape[1] != y_pred.shape[1]:
---> 99         raise ValueError("y_true and y_pred have different number of output "
    100                          "({0}!={1})".format(y_true.shape[1], y_pred.shape[1]))
    101

ValueError: y_true and y_pred have different number of output (1!=2)
```

In [ ]:
```
1
```