In [5]:

```
! pip install pmdarima
```

```
Requirement already satisfied: pmdarima in c:\users\saisr\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\saisr\anaconda3\lib\site-packages (from pmdarima)
(1.2.1)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in c:\users\saisr\anaconda3\lib\site-packages (
from pmdarima) (0.29.36)
Requirement already satisfied: statsmodels>=0.13.2 in c:\users\saisr\anaconda3\lib\site-packages (from pmdarima
) (0.13.5)
Requirement already satisfied: pandas>=0.19 in c:\users\saisr\anaconda3\lib\site-packages (from pmdarima) (1.5.
3)
Requirement already satisfied: urllib3 in c:\users\saisr\anaconda3\lib\site-packages (from pmdarima) (1.26.14)
Requirement already satisfied: scipy>=1.3.2 in c:\users\saisr\anaconda3\lib\site-packages (from pmdarima) (1.10
.0)
Requirement already satisfied: numpy>=1.21.2 in c:\users\saisr\appdata\roaming\python\python310\site-packages (
from pmdarima) (1.25.1)
Requirement already satisfied: joblib>=0.11 in c:\users\saisr\anaconda3\lib\site-packages (from pmdarima) (1.1.
1)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\saisr\anaconda3\lib\site-packages (from
pmdarima) (65.6.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\saisr\anaconda3\lib\site-packages (from pandas>=0.19->p
mdarima) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\saisr\anaconda3\lib\site-packages (from panda
s>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\saisr\anaconda3\lib\site-packages (from scikit-
learn>=0.22->pmdarima) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\saisr\anaconda3\lib\site-packages (from statsmodels>=0.
13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\saisr\anaconda3\lib\site-packages (from statsmodels>
=0.13.2->pmdarima) (22.0)
Requirement already satisfied: six in c:\users\saisr\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodel
s>=0.13.2->pmdarima) (1.16.0)
```

In [14]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
plt.style.use('fivethirtyeight')
from keras.optimizers import Adam
import statistics
```

In [6]:

```
! pip install sequential
```

```
Collecting sequential
  Using cached sequential-1.0.0.tar.gz (2.3 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: sequential
  Building wheel for sequential (setup.py): started
  Building wheel for sequential (setup.py): finished with status 'done'
  Created wheel for sequential: filename=sequential-1.0.0-py3-none-any.whl size=2870 sha256=23a847e755d8fa05e00
c6831dd5481396eb9ddc4f12de794ac508151d49f5431
  Stored in directory: c:\users\saisr\appdata\local\pip\cache\wheels\06\e8\4e\0aca131055d08a09e226d94a886c115ff
71dc602e61fbec422
Successfully built sequential
Installing collected packages: sequential
Successfully installed sequential-1.0.0
```

In [8]:

```
path=r'C:\Users\saisr\Downloads\yahoo_stock.csv'
df=pd.read_csv(path)
df=df.dropna()
print('shape od data',df.shape)
df.head()
```

```
shape od data (1825, 7)
```

Out[8]:

	Date	High	Low	Open	Close	Volume	Adj Close
0	2015-11-23	2095.610107	2081.389893	2089.409912	2086.590088	3.587980e+09	2086.590088
1	2015-11-24	2094.120117	2070.290039	2084.419922	2089.139893	3.884930e+09	2089.139893
2	2015-11-25	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
3	2015-11-26	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
4	2015-11-27	2093.290039	2084.129883	2088.820068	2090.110107	1.466840e+09	2090.110107

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1825 entries, 0 to 1824
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1825 non-null   object 
 1   High       1825 non-null   float64
 2   Low        1825 non-null   float64
 3   Open       1825 non-null   float64
 4   Close      1825 non-null   float64
 5   Volume     1825 non-null   float64
 6   Adj Close  1825 non-null   float64
dtypes: float64(6), object(1)
memory usage: 99.9+ KB
```

In [12]:
```
len(df['Date'].unique())
```

Out[12]:
```
1825
```

In [13]:
```
df.dtypes
```

Out[13]:
```
Date          object
High         float64
Low          float64
Open         float64
Close        float64
Volume       float64
Adj Close    float64
dtype: object
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Date         0
High         0
Low          0
Open         0
Close        0
Volume       0
Adj Close    0
dtype: int64
```

In [15]:
```
df.Date=pd.to_datetime(df['Date'])
```

In [16]:
```
df.Date.min(),df.Date.max()
```

Out[16]:
```
(Timestamp('2015-11-23 00:00:00'), Timestamp('2020-11-20 00:00:00'))
```

In [17]:
```
df.Date.max()-df.Date.min()
```

Out[17]:
```
Timedelta('1824 days 00:00:00')
```

In [18]:
```
df.set_index('Date',inplace=True)
```

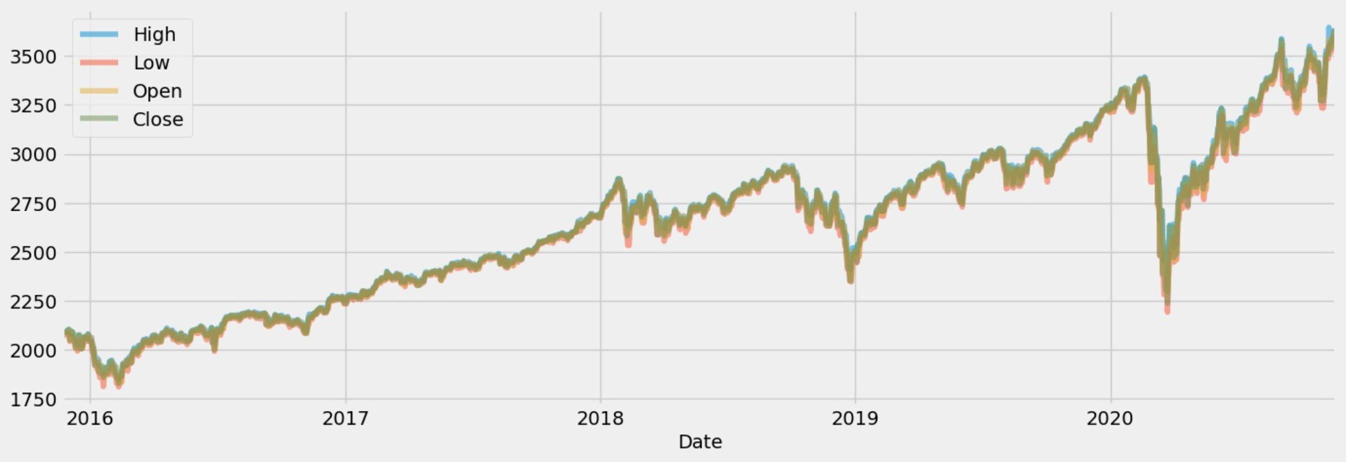
In [19]:
```
df.head()
```

Out[19]:

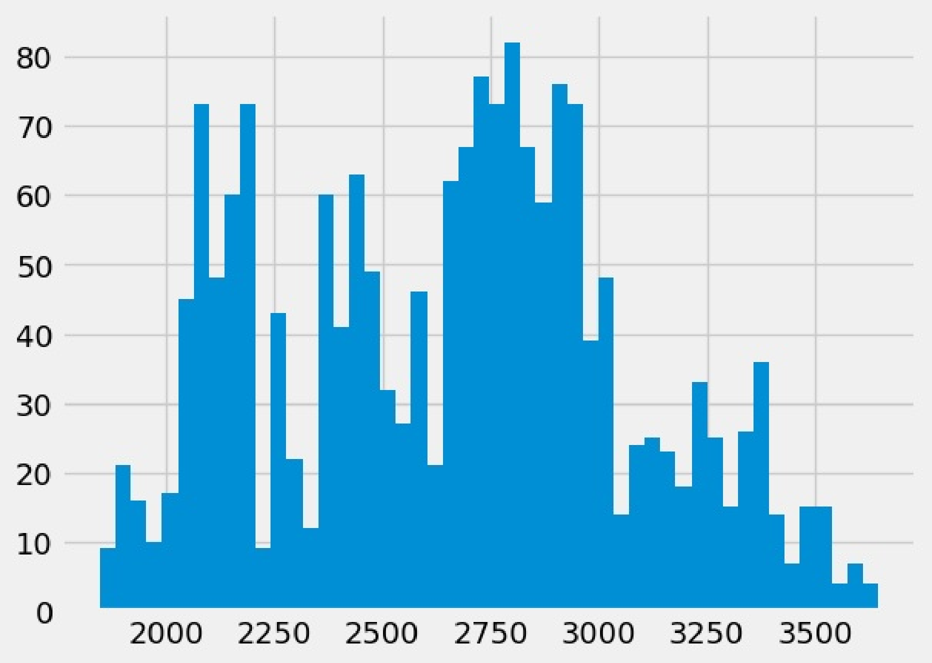
	High	Low	Open	Close	Volume	Adj Close
Date						
2015-11-23	2095.610107	2081.389893	2089.409912	2086.590088	3.587980e+09	2086.590088
2015-11-24	2094.120117	2070.290039	2084.419922	2089.139893	3.884930e+09	2089.139893
2015-11-25	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
2015-11-26	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
2015-11-27	2093.290039	2084.129883	2088.820068	2090.110107	1.466840e+09	2090.110107

In [20]:
```
df[['High','Low','Open','Close']].plot(figsize = (15, 5), alpha = 0.5)
```

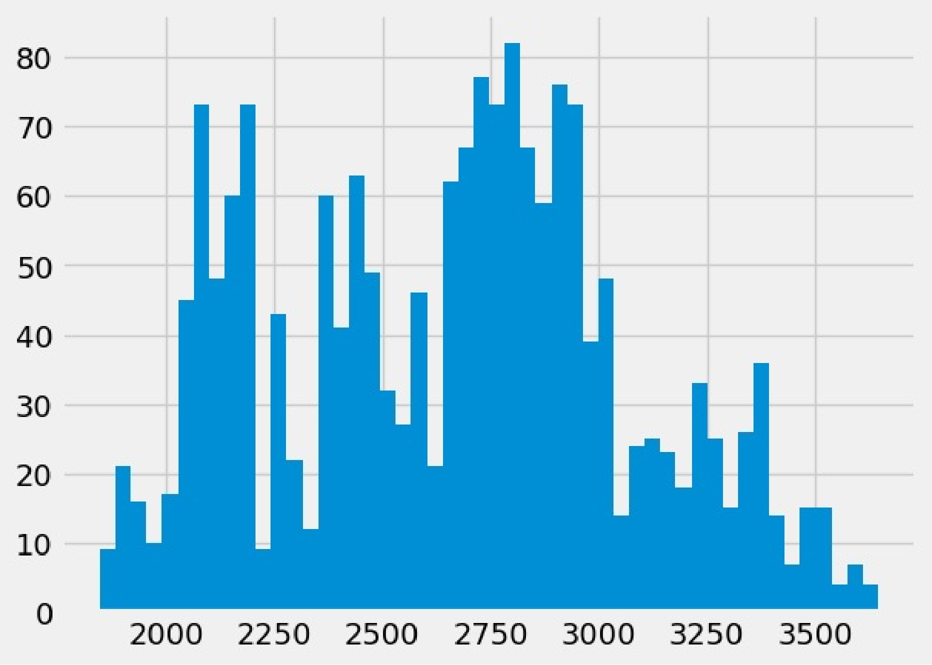
Out[20]:
```
<Axes: xlabel='Date'>
```



```
In [21]: df.High.hist(bins=50)
Out[21]: <Axes: >
```



```
In [22]: df.High.hist(bins=50)
Out[22]: <Axes: >
```



In [23]:

```
np.round(df.shape[0]/10,0)
```

Out[23]:

```
182.0
```

In [24]:

```
from statistics import stdev
mean=[]
std=[]

for i in range(0,10):
    mean.append(df['High'].iloc[(i*182):(i*182)+182].mean())
    std.append(stdev(df['High'].iloc[(i*182):(i*182)+182]))
```

In [25]:

```
pd.concat([pd.DataFrame(mean,columns=['mean']),pd.DataFrame(std,columns=['std'])], axis=1)
```

Out[25]:

	mean	std
0	2019.354300	71.041024
1	2147.740282	36.287599
2	2322.969570	60.228422
3	2489.706581	55.878102
4	2711.253743	63.125935
5	2816.088946	68.351685
6	2754.165647	131.792411
7	2965.153137	74.059325
8	3045.669328	267.695412
9	3309.076588	165.733813

In [27]:

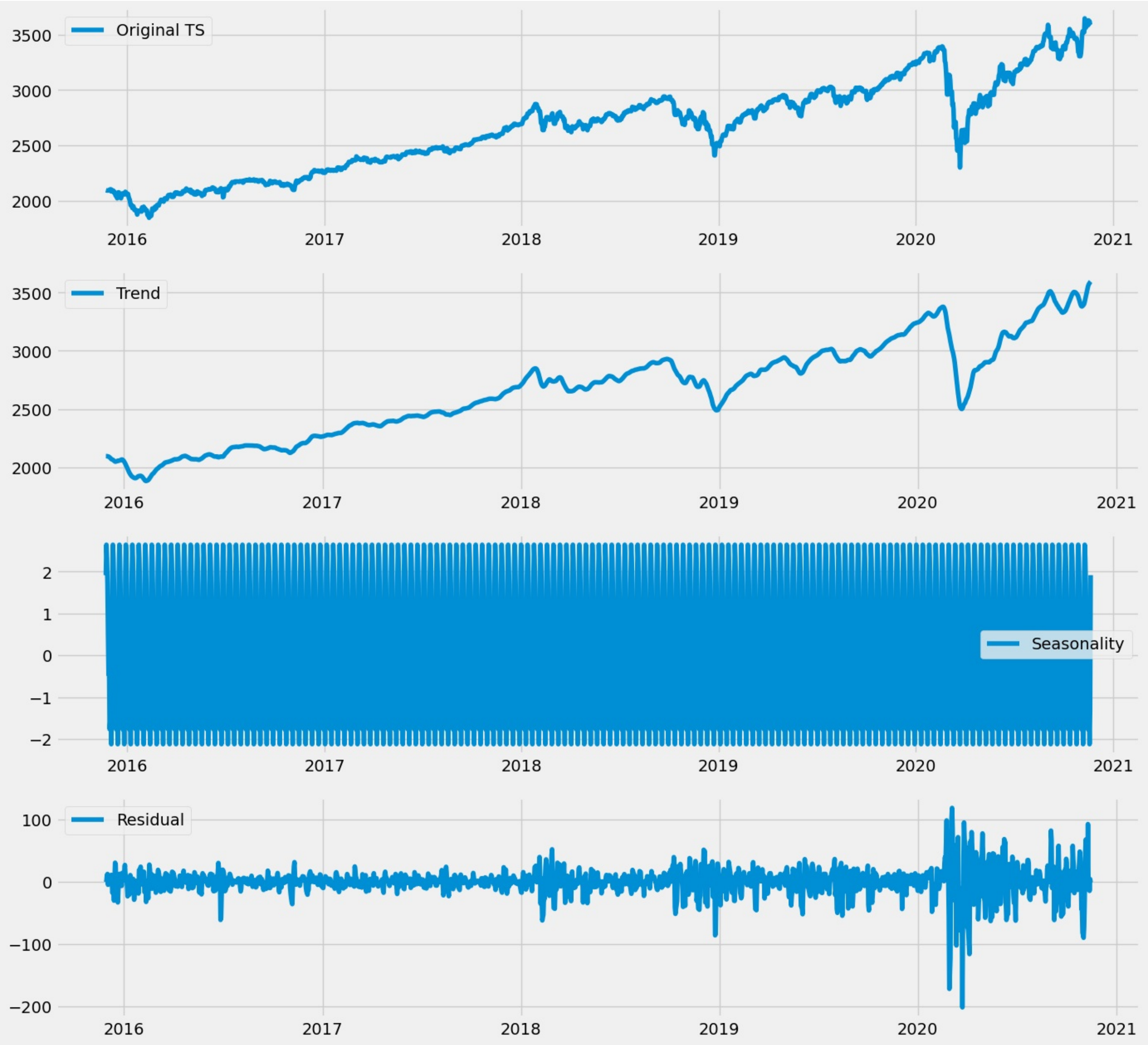
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [28]:

```
decompose_add=seasonal_decompose(df['High'], model='additive', period=12)
plt.figure(figsize=(15,15))
plt.subplot(411)
plt.plot(df['High'], label='Original TS')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(decompose_add.trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(decompose_add.seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(decompose_add.resid, label='Residual')
plt.legend(loc='best')
```

Out[28]:

```
<matplotlib.legend.Legend at 0x2751fc81900>
```

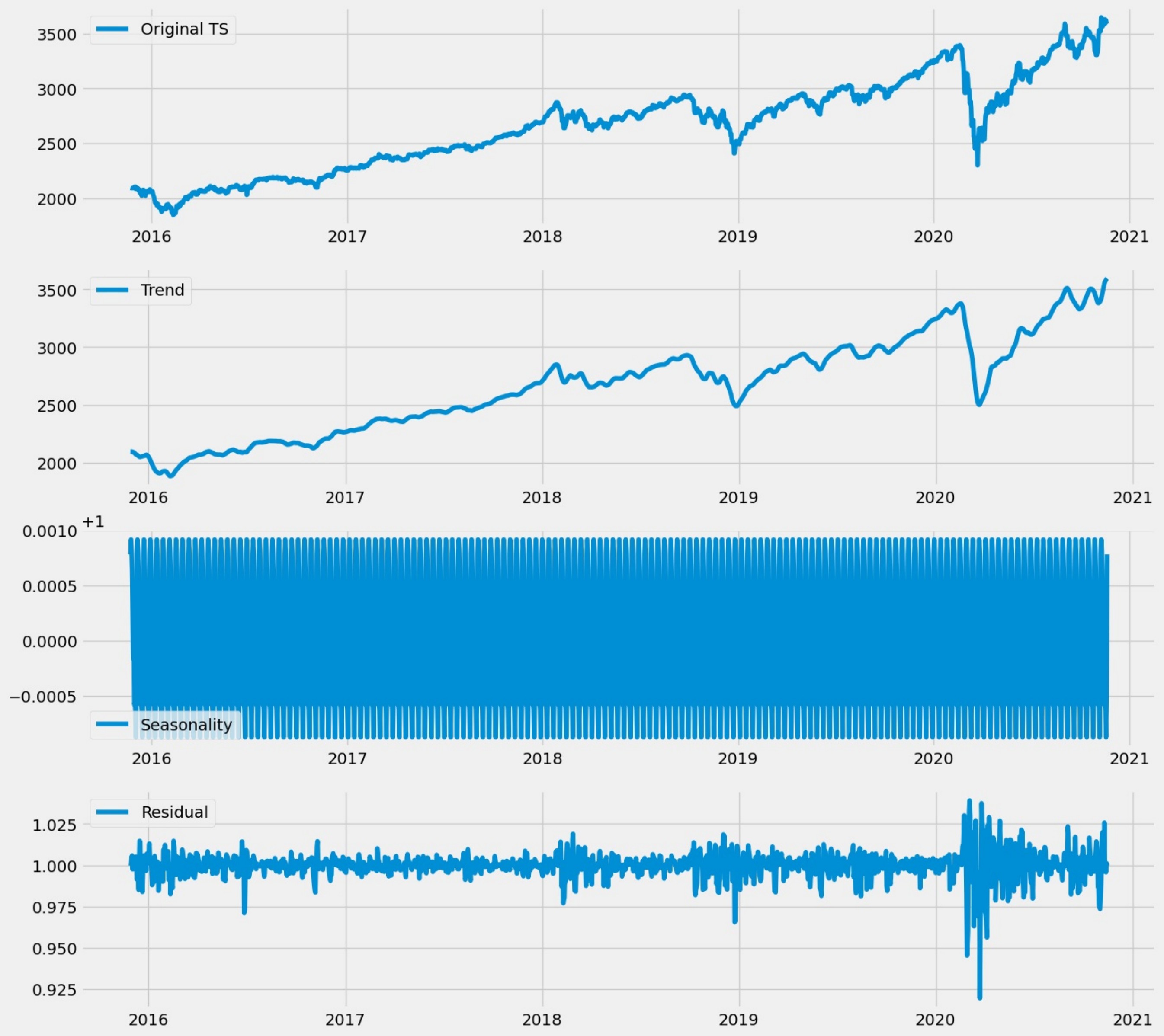


In [29]:

```
decompose_mul=seasonal_decompose(df['High'], model='multiplicative', period=12)
plt.figure(figsize=(15,15))
plt.subplot(411)
plt.plot(df['High'], label='Original TS')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(decompose_mul.trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(decompose_mul.seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(decompose_mul.resid, label='Residual')
plt.legend(loc='best')
```

Out[29]:

```
<matplotlib.legend.Legend at 0x275207057e0>
```

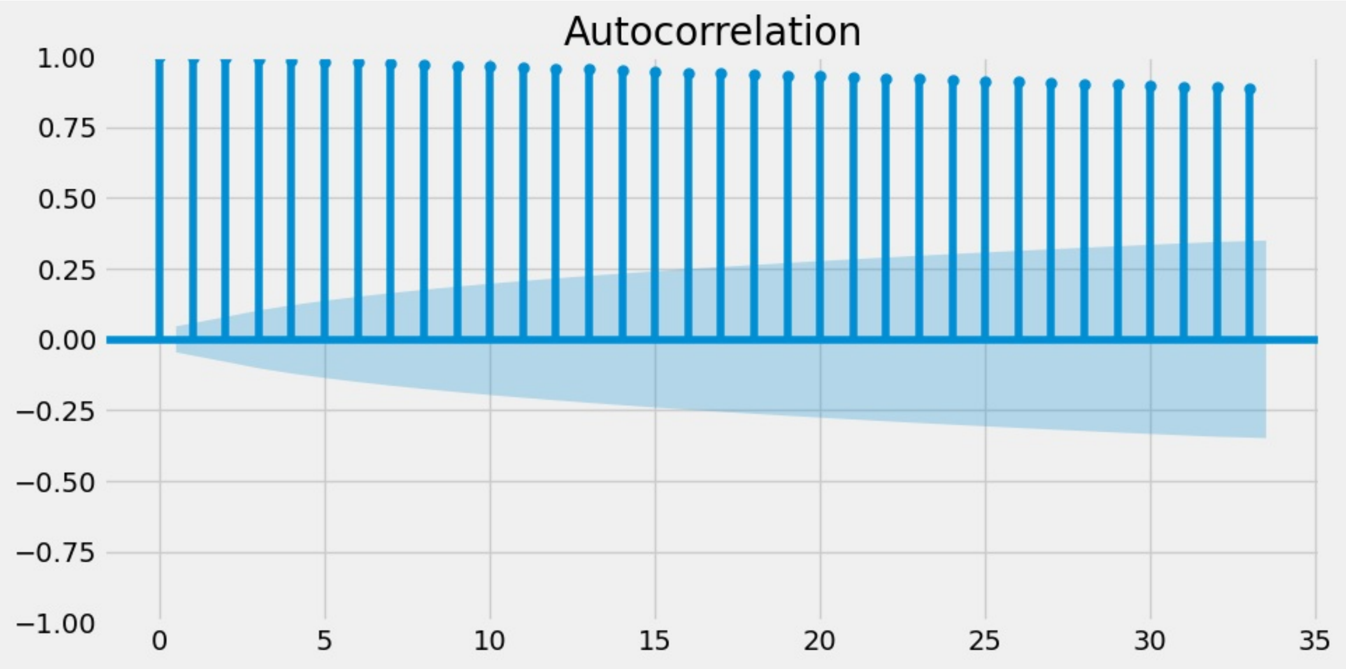


In [31]:

```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
```

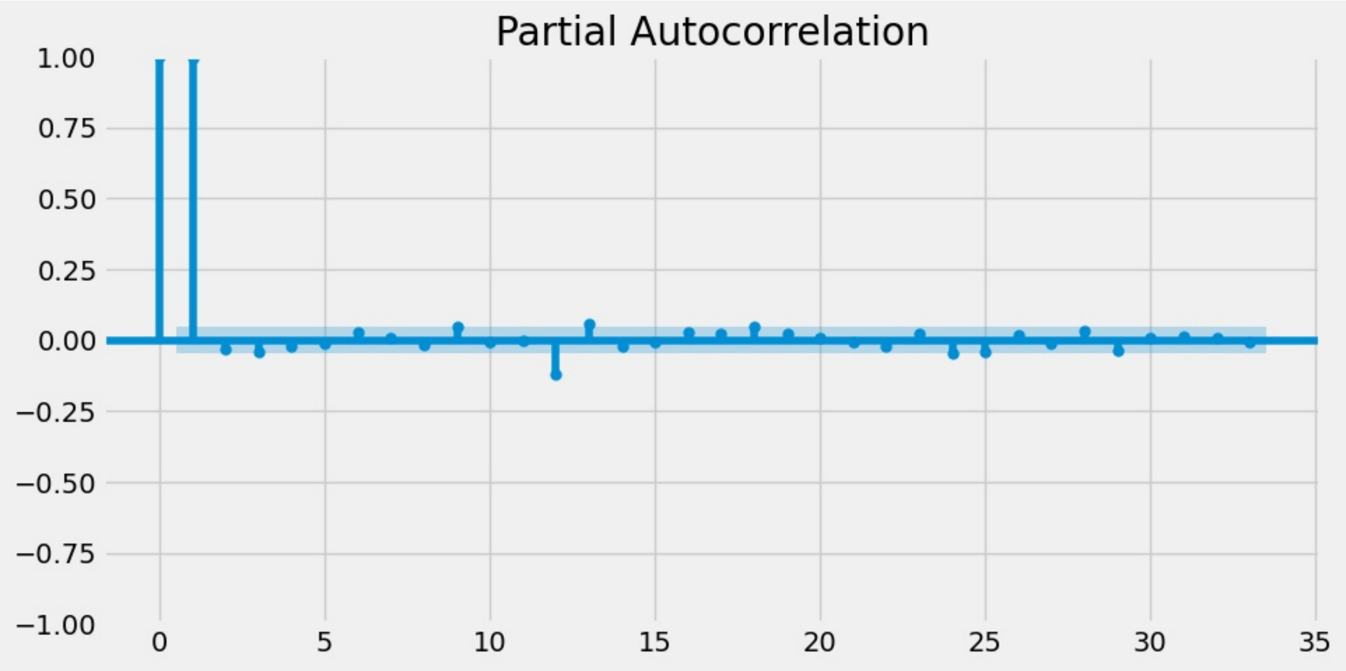
In [32]:

```
plt.rc("figure", figsize=(10,5))
plot_acf(df['High'])
print()
```



```
In [33]: plt.rc("figure", figsize=(10,5))
         plot_pacf(df['High'])
         print()
```

```
C:\Users\saisr\anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default met
hod 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default will change tounadjust
ed Yule-Walker ('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```



```
In [35]: from statsmodels.tsa.stattools import adfuller
```

In [37]:

```
result = adfuller(df['High'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -0.713598
p-value: 0.843196
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [38]:

```
from numpy import log

result = adfuller(log(df['High']))
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -0.920468
p-value: 0.781157
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

In [39]:

```
new_df=df['High'].iloc[:-4]
```

In [40]:

```
train_len = math.ceil(len(new_df)*0.8)
train_len
```

Out[40]:

```
1457
```

In [41]:

```
window=10
```

In [42]:

```
train_data = new_df[0:train_len]

X_train=[]
Y_train=[]

for i in range(window, len(train_data)):
    X_train.append(train_data[i-window:i])
    Y_train.append(train_data[i])
```

In [43]:

```
X_train, Y_train= np.array(X_train), np.array(Y_train)
```

In [44]:

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape
```

Out[44]:

```
(1447, 10, 1)
```

In [45]:

```
X_train
```

Out[45]:

```
array([[[2095.61010742],
        [2094.12011719],
        [2093.        ],
        ...,
        [2093.81005859],
        [2103.37011719],
        [2104.27001953]],

       [[2094.12011719],
        [2093.        ],
        [2093.        ],
        ...,
        [2103.37011719],
        [2104.27001953],
        [2085.        ]],

       [[2093.        ],
        [2093.        ],
        [2093.29003906],
        ...,
        [2104.27001953],
        [2085.        ],
        [2093.84008789]],

       ...,

       [[3078.34008789],
        [3097.77001953],
        [3093.09008789],
        ...,
        [3098.06005859],
        [3098.19995117],
        [3120.45996094]],

       [[3097.77001953],
        [3093.09008789],
        [3093.09008789],
        ...,
        [3098.19995117],
        [3120.45996094],
        [3120.45996094]],

       [[3093.09008789],
        [3093.09008789],
        [3093.09008789],
        ...,
        [3120.45996094],
        [3120.45996094],
        [3120.45996094]]])
```

In [46]:

```python
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
```

In [47]:

```python
model=Sequential()
model.add(LSTM(50, activation='relu', input_shape=(X_train.shape[1],1)))
model.add(Dense(25))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
model.fit(X_train, Y_train, epochs=10, batch_size=10, verbose=0)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50)                10400

 dense (Dense)               (None, 25)                1275

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 11,701
Trainable params: 11,701
Non-trainable params: 0
_________________________________________________________________
```

Out[47]:

```
<keras.callbacks.History at 0x275212f3790>
```

In [48]:

```python
test_data = new_df[train_len-window:]

X_val=[]
Y_val=[]

for i in range(window, len(test_data)):
    X_val.append(test_data[i-window:i])
    Y_val.append(test_data[i])
```

In [49]:

```python
X_val, Y_val = np.array(X_val), np.array(Y_val)
```

```
X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1],1))
```

In [50]:
```
X_val, Y_val = np.array(X_val), np.array(Y_val)
X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1],1))
```

In [51]:
```
X_val.shape, Y_val.shape
```

Out[51]:
```
((364, 10, 1), (364,))
```

In [52]:
```
prediction = model.predict(X_val)
```

```
12/12 [==============================] - 1s 4ms/step
```

In [53]:
```
len(prediction), len(Y_val)
```

Out[53]:
```
(364, 364)
```

In [54]:
```
from sklearn.metrics import mean_squared_error

lstm_train_pred = model.predict(X_train)
lstm_valid_pred = model.predict(X_val)
print('Train rmse:', np.sqrt(mean_squared_error(Y_train, lstm_train_pred)))
print('Validation rmse:', np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)))
```

```
46/46 [==============================] - 0s 5ms/step
12/12 [==============================] - 0s 4ms/step
Train rmse: 20.630865979189004
Validation rmse: 139.53608995892196
```

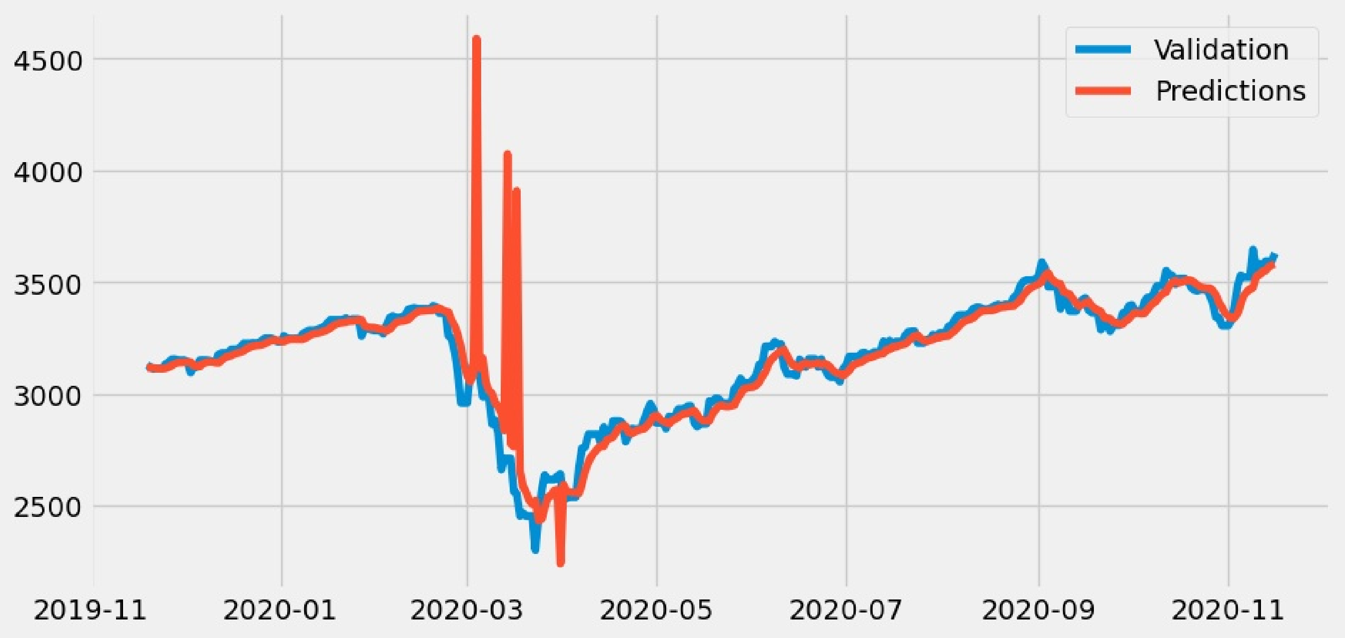
In [55]:
```
valid = pd.DataFrame(new_df[train_len:])
valid['Predictions']=lstm_valid_pred
valid
```

Out[55]:

	High	Predictions
Date		
2019-11-19	3127.639893	3112.767090
2019-11-20	3118.969971	3115.913330
2019-11-21	3110.110107	3115.288086
2019-11-22	3112.870117	3112.446289
2019-11-23	3112.870117	3111.477051
...	...	...
2020-11-12	3569.020020	3546.993652
2020-11-13	3593.659912	3550.545166
2020-11-14	3593.659912	3567.268311
2020-11-15	3593.659912	3574.935791
2020-11-16	3628.510010	3579.819092

364 rows × 2 columns

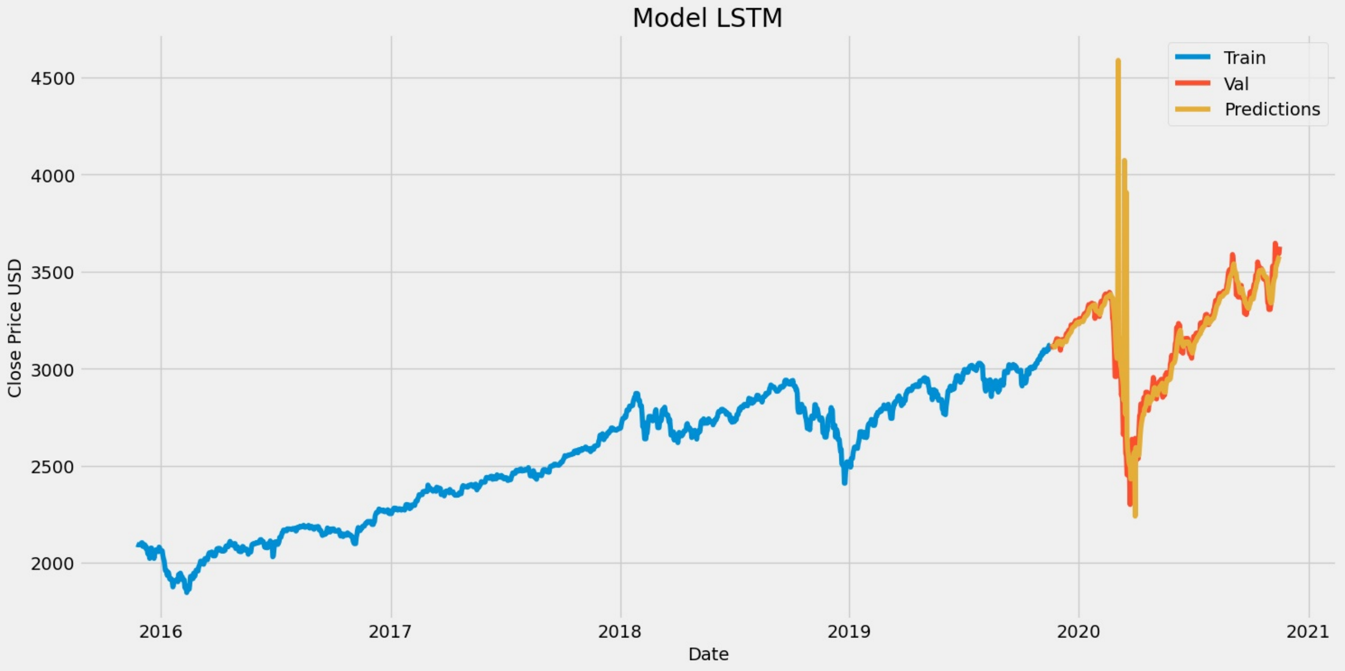
In [56]:
```
plt.plot(valid[['High','Predictions']])
plt.legend(['Validation','Predictions'])
plt.show()
```



In [57]:

```
train = new_df[:train_len]
valid = pd.DataFrame(new_df[train_len:])
valid['Predictions']=lstm_valid_pred

plt.figure(figsize=(16,8))
plt.title('Model LSTM')
plt.xlabel('Date')
plt.ylabel('Close Price USD')
plt.plot(train)
plt.plot(valid[['High','Predictions']])
plt.legend(['Train','Val','Predictions'])
plt.show()
```



In [58]:

```
train_error=[]
val_error=[]

window_number=[5,8,10,15,20,30,40]
for i in window_number:
    #_______________________________________________________________________
    train_data = new_df[0:train_len]

    X_train=[]
    Y_train=[]

    for i in range(window, len(train_data)):
        X_train.append(train_data[i-window:i])
        Y_train.append(train_data[i])

    X_train, Y_train= np.array(X_train), np.array(Y_train)
    X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
    #________________________________________________________________________
    test_data = new_df[train_len-window:]
    X_val=[]
    Y_val=[]

    for i in range(window, len(test_data)):
        X_val.append(test_data[i-window:i])
        Y_val.append(test_data[i])

    X_val, Y_val = np.array(X_val), np.array(Y_val)
    X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1],1))
    #______________________________________________________________________
    model=Sequential()
    model.add(LSTM(50, activation='relu', input_shape=(X_train.shape[1],1)))
    model.add(Dense(25))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.fit(X_train, Y_train, epochs=10, batch_size=10, verbose=0)
    #______________________________________________________________________
    lstm_train_pred = model.predict(X_train)
    lstm_valid_pred = model.predict(X_val)
    train_error.append(np.sqrt(mean_squared_error(Y_train, lstm_train_pred)))
    val_error.append(np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)))
```

```
46/46 [==============================] - 1s 5ms/step
12/12 [==============================] - 0s 5ms/step
46/46 [==============================] - 1s 5ms/step
12/12 [==============================] - 0s 6ms/step
46/46 [==============================] - 1s 5ms/step
12/12 [==============================] - 0s 5ms/step
46/46 [==============================] - 1s 5ms/step
12/12 [==============================] - 0s 5ms/step
46/46 [==============================] - 1s 5ms/step
12/12 [==============================] - 0s 5ms/step
46/46 [==============================] - 1s 6ms/step
12/12 [==============================] - 0s 5ms/step
46/46 [==============================] - 1s 6ms/step
12/12 [==============================] - 0s 7ms/step
```

In [59]:

```
train_error
```

Out[59]:

```
[31.939593196995773,
 15.9205709079055,
 22.189286413160833,
 31.545365553339785,
 35.06977502283139,
 29.62113714349387,
 30.75894659664298]
```

In [60]:

```
val_error
```

Out[60]:

```
[31.939593196995773,
 15.9205709079055,
 22.189286413160833,
 31.545365553339785,
 35.06977502283139,
 29.62113714349387,
 30.75894659664298]
```

In [ ]:

```
pd.concat([pd.DataFrame(train_error,columns=['train_error']),
           pd.DataFrame(val_error,columns=['val_error']),
           pd.DataFrame([5,8,10,15,20,30,40],columns=['window'])], axis=1).set_index('window')
```

In [61]:

```
window=10

train_data = new_df[0:train_len]
X_train=[]
Y_train=[]
for i in range(window, len(train_data)):
    X_train.append(train_data[i-window:i])
    Y_train.append(train_data[i])

X_train, Y_train= np.array(X_train), np.array(Y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
#____________________________________________________________________
test_data = new_df[train_len-window:]
X_val=[]
Y_val=[]
for i in range(window, len(test_data)):
    X_val.append(test_data[i-window:i])
    Y_val.append(test_data[i])

X_val, Y_val = np.array(X_val), np.array(Y_val)
X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1],1))
```

In [7]:

```
model=Sequential()
model.add(LSTM(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
model.add(LSTM(50,return_sequences=False,activation='relu'))
model.add(Dense(100))
model.add(Dense(25))
model.add(Dense(1))
```

```
opt1=Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999)
model.compile(loss='mean_squared_error', optimizer=opt1)
model.summary()
model.fit(X_train, Y_train, epochs=100, batch_size=10, verbose=0)
Model: "sequential_8"
_________________________________________________________________
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 model=Sequential()
      2 model.add(LSTM(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
      3 model.add(LSTM(50,return_sequences=False,activation='relu'))

NameError: name 'Sequential' is not defined
```

In [8]:
```
lstm_train_pred = model.predict(X_train)
lstm_valid_pred = model.predict(X_val)
print('Train rmse:', np.sqrt(mean_squared_error(Y_train, lstm_train_pred)))
print('Validation rmse:', np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 lstm_train_pred = model.predict(X_train)
      2 lstm_valid_pred = model.predict(X_val)
      3 print('Train rmse:', np.sqrt(mean_squared_error(Y_train, lstm_train_pred)))

NameError: name 'model' is not defined
```

In [ ]:
```
valid = pd.DataFrame(new_df[train_len:])
valid['Predictions']=lstm_valid_pred
```

In [ ]:
```
plt.plot(valid[['High','Predictions']])
plt.legend(['Validation','Predictions'])
plt.show()
```

In [ ]:
```
r1=[]
r2=[]

for i in range(0,10):
    model=Sequential()
    model.add(LSTM(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
    model.add(LSTM(50,return_sequences=False,activation='relu'))
    model.add(Dense(100))
    model.add(Dense(25))
    model.add(Dense(1))
    opt1=Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999)
    model.compile(loss='mean_squared_error', optimizer=opt1)
    model.fit(X_train, Y_train, epochs=100, batch_size=10,verbose=0)

    lstm_train_pred = model.predict(X_train)
    lstm_valid_pred = model.predict(X_val)
    r1.append(np.round(np.sqrt(mean_squared_error(Y_train, lstm_train_pred)),2))
    r2.append(np.round(np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)),2))
```

In [ ]:
```
r1, statistics.mean(r1), statistics.stdev(r1)
```

In [ ]:
```
r2, statistics.mean(r2), statistics.stdev(r2)
```

In [ ]:
```
r1=[]
r2=[]

for i in range(0,10):
    model=Sequential()
    model.add(LSTM(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1),recurrent_drop
    model.add(LSTM(50,return_sequences=False,activation='relu'))
    model.add(Dense(100))
    model.add(Dropout(0.2))
    model.add(Dense(25))
    model.add(Dense(1))
    opt1=Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999)
    model.compile(loss='mean_squared_error', optimizer=opt1)
    model.fit(X_train, Y_train, epochs=100, batch_size=10,verbose=0)

    lstm_train_pred = model.predict(X_train)
    lstm_valid_pred = model.predict(X_val)
    r1.append(np.round(np.sqrt(mean_squared_error(Y_train, lstm_train_pred)),2))
    r2.append(np.round(np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)),2))
```

In [ ]:
```
r1, statistics.mean(r1), statistics.stdev(r1)
```

In [ ]:
```
r2, statistics.mean(r2), statistics.stdev(r2)
```

In [ ]:
```
from keras.layers import SimpleRNN
```

In [ ]:

```python
r1=[]
r2=[]

for i in range(0,10):
    model=Sequential()
    model.add(SimpleRNN(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
    model.add(SimpleRNN(50,return_sequences=False,activation='relu'))
    model.add(Dense(100))
    model.add(Dense(25))
    model.add(Dense(1))
    opt1=Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999)
    model.compile(loss='mean_squared_error', optimizer=opt1)
    model.fit(X_train, Y_train, epochs=100, batch_size=10,verbose=0)

    lstm_train_pred = model.predict(X_train)
    lstm_valid_pred = model.predict(X_val)
    r1.append(np.round(np.sqrt(mean_squared_error(Y_train, lstm_train_pred)),2))
    r2.append(np.round(np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)),2))
```

In [64]:

```python
r1,statistics.mean(r1), statistics.stdev(r1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[64], line 1
----> 1 r1, statistics.mean(r1), statistics.stdev(r1)

NameError: name 'r1' is not defined
```

In [65]:

```python
r2, statistics.mean(r2), statistics.stdev(r2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[65], line 1
----> 1 r2, statistics.mean(r2), statistics.stdev(r2)

NameError: name 'r2' is not defined
```

In [66]:

```python
valid = pd.DataFrame(new_df[train_len:])
valid['Predictions']=lstm_valid_pred
valid
```

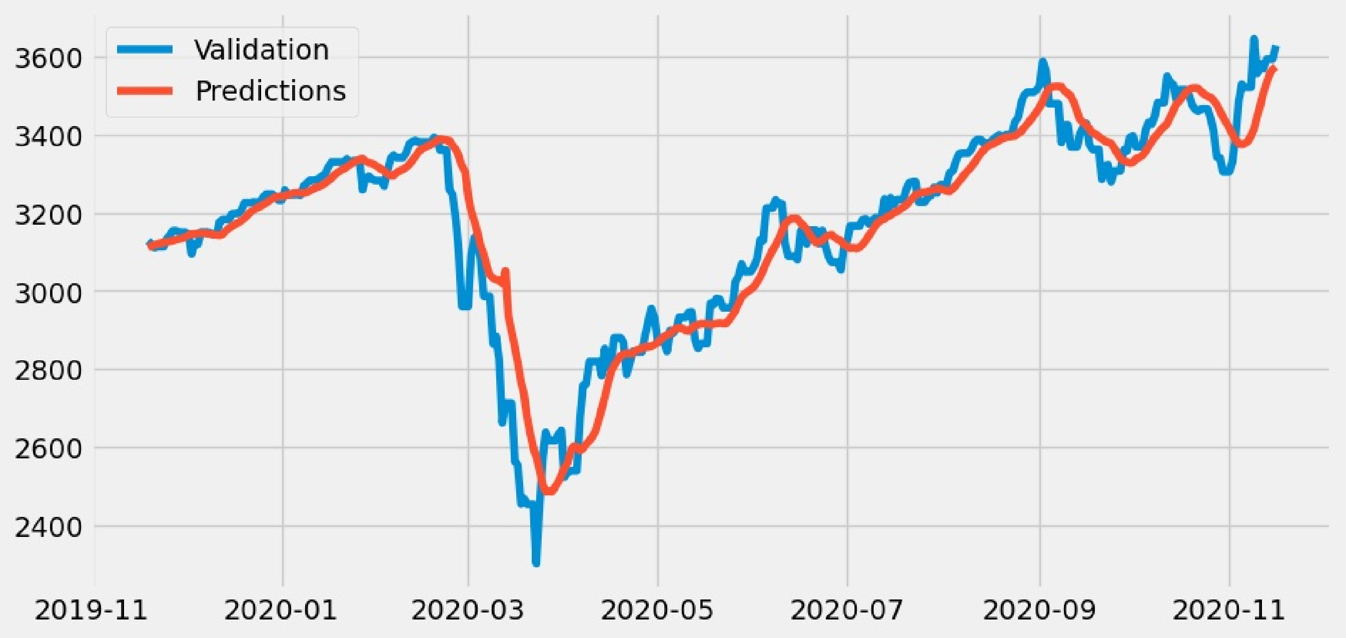
Out[66]:

	High	Predictions
Date		
2019-11-19	3127.639893	3110.744629
2019-11-20	3118.969971	3114.309814
2019-11-21	3110.110107	3117.012939
2019-11-22	3112.870117	3119.484863
2019-11-23	3112.870117	3121.323730
...	...	...
2020-11-12	3569.020020	3506.516113
2020-11-13	3593.659912	3533.486328
2020-11-14	3593.659912	3556.412354
2020-11-15	3593.659912	3567.464355
2020-11-16	3628.510010	3574.162842

364 rows × 2 columns

In [67]:

```python
plt.plot(valid[['High','Predictions']])
plt.legend(['Validation','Predictions'])
plt.show()
```



In [68]:

```
import tensorflow

r1=[]
r2=[]

model=Sequential()
model.add(SimpleRNN(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
model.add(SimpleRNN(50,return_sequences=False,activation='relu'))
model.add(Dense(100))
model.add(Dense(25))
model.add(Dense(1))
lr_schedule = tensorflow.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-5 * 10**(epoch / 85))
opt1=Adam(learning_rate=1e-5,beta_1=0.9,beta_2=0.7)
model.compile(loss='mean_squared_error', optimizer=opt1)
history=model.fit(X_train, Y_train, epochs=100, batch_size=10,verbose=2, callbacks=[lr_schedule])

lstm_train_pred = model.predict(X_train)
lstm_valid_pred = model.predict(X_val)
r_train_new=np.round(np.sqrt(mean_squared_error(Y_train, lstm_train_pred)),2)
r_val_new=np.round(np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)),2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[68], line 7
      4 r2=[]
      6 model=Sequential()
----> 7 model.add(SimpleRNN(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
      8 model.add(SimpleRNN(50,return_sequences=False,activation='relu'))
      9 model.add(Dense(100))

NameError: name 'SimpleRNN' is not defined
```

In [69]:

```
plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-5, 5e-4, 0, 1000])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[69], line 1
----> 1 plt.semilogx(history.history["lr"], history.history["loss"])
      2 plt.axis([1e-5, 5e-4, 0, 1000])

NameError: name 'history' is not defined
```

In [70]:

```
r1=[]
r2=[]

for i in range(0,10):
    model=Sequential()
    model.add(SimpleRNN(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
    model.add(SimpleRNN(50,return_sequences=False,activation='relu'))
    model.add(Dense(100))
    model.add(Dense(25))
    model.add(Dense(1))
    opt1=Adam(learning_rate=1e-4,beta_1=0.9,beta_2=0.7)
    model.compile(loss='mean_squared_error', optimizer=opt1)
    model.fit(X_train, Y_train, epochs=100, batch_size=10,verbose=0)

    lstm_train_pred = model.predict(X_train)
    lstm_valid_pred = model.predict(X_val)
    r1.append(np.round(np.sqrt(mean_squared_error(Y_train, lstm_train_pred)),2))
    r2.append(np.round(np.sqrt(mean_squared_error(Y_val, lstm_valid_pred)),2))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[70], line 6
      4 for i in range(0,10):
      5     model=Sequential()
----> 6     model.add(SimpleRNN(50,return_sequences=True, activation='relu', input_shape=(X_train.shape[1],1)))
      7     model.add(SimpleRNN(50,return_sequences=False,activation='relu'))
      8     model.add(Dense(100))

NameError: name 'SimpleRNN' is not defined
```

In [71]:

```
r1, statistics.mean(r1), statistics.stdev(r1)
```

```
---------------------------------------------------------------------------
StatisticsError                           Traceback (most recent call last)
Cell In[71], line 1
----> 1 r1, statistics.mean(r1), statistics.stdev(r1)

File ~\anaconda3\lib\statistics.py:328, in mean(data)
    326 n = len(data)
    327 if n < 1:
--> 328     raise StatisticsError('mean requires at least one data point')
    329 T, total, count = _sum(data)
    330 assert count == n

StatisticsError: mean requires at least one data point
```

In [ ]:

```
r2, statistics.mean(r2), statistics.stdev(r2)
```

In [72]:

```
valid = pd.DataFrame(new_df[train_len:])
valid['Predictions']=lstm_valid_pred
valid
```

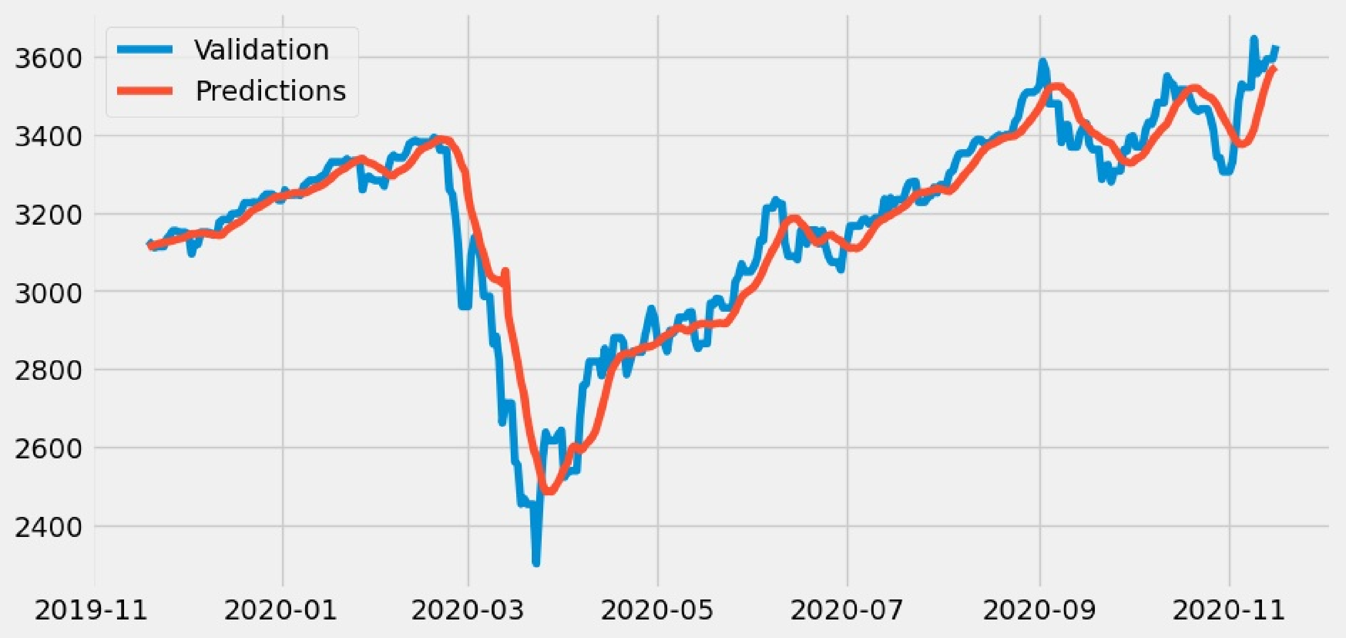
Out[72]:

	High	Predictions
Date		
2019-11-19	3127.639893	3110.744629
2019-11-20	3118.969971	3114.309814
2019-11-21	3110.110107	3117.012939
2019-11-22	3112.870117	3119.484863
2019-11-23	3112.870117	3121.323730
...	...	...
2020-11-12	3569.020020	3506.516113
2020-11-13	3593.659912	3533.486328
2020-11-14	3593.659912	3556.412354
2020-11-15	3593.659912	3567.464355
2020-11-16	3628.510010	3574.162842

364 rows × 2 columns

In [73]:

```
plt.plot(valid[['High','Predictions']])
plt.legend(['Validation','Predictions'])
plt.show()
```



In [74]:

```
last_10_days=new_df[-10:].values
X_test=[]
X_test.append(last_10_days)
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
pred_price=model.predict(X_test)
print(pred_price)
```

```
1/1 [==============================] - 0s 126ms/step
[[[3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]]]
```

In [75]:

```
df['High'].iloc[-4]
```

Out[75]:

```
3623.110107421875
```

In [76]:

```
df['High'].iloc[-4]-pred_price
array([[-17.979248]], dtype=float32)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[76], line 2
      1 df['High'].iloc[-4]-pred_price
----> 2 array([[-17.979248]], dtype=float32)

NameError: name 'array' is not defined
```

In [77]:

```
df.High.tail(14)
```

Out[77]:

```
Date
2020-11-07    3521.580078
2020-11-08    3521.580078
2020-11-09    3645.989990
2020-11-10    3557.219971
2020-11-11    3581.159912
2020-11-12    3569.020020
2020-11-13    3593.659912
2020-11-14    3593.659912
2020-11-15    3593.659912
2020-11-16    3628.510010
2020-11-17    3623.110107
2020-11-18    3619.090088
2020-11-19    3585.219971
2020-11-20    3581.229980
Name: High, dtype: float64
```

In [78]:

```
last_9_days=new_df[-9:].values
X_test=[]
X_test=np.append(last_9_days,pred_price)
X_test=np.array(X_test)
X_test
X_test=np.reshape(X_test,(1,X_test.shape[0],1))
pred_price2=model.predict(X_test)
```

```
print(pred_price2)
```

```
1/1 [==============================] - 0s 82ms/step
[[[3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]]]
```

In [79]:

```
df['High'].iloc[-3]
```

Out[79]:

```
3619.090087890625
```

In [80]:

```
df['High'].iloc[-3]-pred_price2
```

Out[80]:

```
array([[[ 97.51001 ],
        [-26.899902],
        [ 61.870117],
        [ 37.930176],
        [ 50.07007 ],
        [ 25.430176],
        [ 25.430176],
        [ 25.430176],
        [ -9.419922],
        [ 97.51001 ],
        [ 97.51001 ],
        [-26.899902],
        [ 61.870117],
        [ 37.930176],
        [ 50.07007 ],
        [ 25.430176],
        [ 25.430176],
        [ 25.430176],
        [ -9.419922]]], dtype=float32)
```

In [81]:

```
last_8_days=new_df[-8:].values
X_test=[]
X_test=np.append(last_8_days,pred_price)
X_test=np.append(X_test,pred_price2)
X_test=np.array(X_test)
X_test
X_test=np.reshape(X_test,(1,X_test.shape[0],1))
pred_price3=model.predict(X_test)
print(pred_price3)
```

```
1/1 [==============================] - 0s 63ms/step
[[[3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]]]
```

In [82]:

```
df['High'].iloc[-2]
```

Out[82]:

```
3585.219970703125
```

In [83]:

```
df['High'].iloc[-2]-pred_price3
```

Out[83]:

```
array([[[-60.77002  ],
        [ 28.       ],
        [  4.0600586],
        [ 16.199951 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [-43.29004  ],
        [ 63.639893 ],
        [ 63.639893 ],
        [-60.77002  ],
        [ 28.       ],
        [  4.0600586],
        [ 16.199951 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [-43.29004  ],
        [ 63.639893 ],
        [-60.77002  ],
        [ 28.       ],
        [  4.0600586],
        [ 16.199951 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [-43.29004  ],
        [ 63.639893 ],
        [ 63.639893 ],
        [-60.77002  ],
        [ 28.       ],
        [  4.0600586],
        [ 16.199951 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [ -8.439941 ],
        [-43.29004  ]]], dtype=float32)
```

In [84]:

```
last_7_days=new_df[-7:].values
X_test=[]
X_test=np.append(last_7_days, pred_price)
X_test=np.append(X_test, pred_price2)
X_test=np.append(X_test, pred_price3)
```

```
X_test=np.array(X_test)
X_test
X_test=np.reshape(X_test,(1,X_test.shape[0],1))
pred_price4=model.predict(X_test)
print(pred_price4)
```

```
1/1 [==============================] - 0s 51ms/step
[[[3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]
  [3521.58]
  [3521.58]
  [3645.99]
  [3557.22]
  [3581.16]
  [3569.02]
  [3593.66]
  [3593.66]
  [3593.66]
  [3628.51]]]
```

In [85]:

```
df['High'].iloc[-1]
```

Out[85]:

```
3581.22998046875
```

In [86]:

```
df['High'].iloc[-1]-pred_price4
```

Out[86]:

```
array([[[ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [ 59.649902  ],
        [ 59.649902  ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [ 59.649902  ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [ 59.649902  ],
        [ 59.649902  ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [ 59.649902  ],
        [ 59.649902  ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [ 59.649902  ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ],
        [ 59.649902  ],
        [ 59.649902  ],
        [-64.76001   ],
        [ 24.01001   ],
        [  0.07006836],
        [ 12.209961  ],
        [-12.429932  ],
        [-12.429932  ],
        [-12.429932  ],
        [-47.28003   ]]], dtype=float32)
```

In [87]:

```
df.High.iloc[-4], df.High.iloc[-3], df.High.iloc[-2], df.High.iloc[-1]
```

Out[87]:

```
(3623.110107421875, 3619.090087890625, 3585.219970703125, 3581.22998046875)
```

In [88]:

```
pred_price, pred_price2, pred_price3, pred_price4
```

Out[88]:

```
(array([[[3521.58],
         [3521.58],
         [3645.99],
         [3557.22],
         [3581.16],
         [3569.02],
         [3593.66],
         [3593.66],
                 [3593.66],
                 [3628.51]]], dtype=float32),
        array([[[3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51]]], dtype=float32),
        array([[[3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51]]], dtype=float32),
        array([[[3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                 [3569.02],
                 [3593.66],
                 [3593.66],
                 [3593.66],
                 [3628.51],
                 [3521.58],
                 [3521.58],
                 [3645.99],
                 [3557.22],
                 [3581.16],
                  [3569.02],
                  [3593.66],
                  [3593.66],
                  [3593.66],
                  [3628.51],
                  [3645.99],
                  [3557.22],
                  [3581.16],
                  [3569.02],
                  [3593.66],
                  [3593.66],
                  [3593.66],
                  [3628.51],
                  [3521.58],
                  [3521.58],
                  [3645.99],
                  [3557.22],
                  [3581.16],
                  [3569.02],
                  [3593.66],
                  [3593.66],
                  [3593.66],
                  [3628.51],
                  [3521.58],
                  [3645.99],
                  [3557.22],
                  [3581.16],
                  [3569.02],
                  [3593.66],
                  [3593.66],
                  [3593.66],
                  [3628.51],
                  [3521.58],
                  [3521.58],
                  [3645.99],
                  [3557.22],
                  [3581.16],
                  [3569.02],
                  [3593.66],
                  [3593.66],
                  [3593.66],
                  [3628.51]]], dtype=float32))
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js