IMPORTING LIBRARIES

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import datetime
from pylab import rcParams
import matplotlib.pyplot as plt
import warnings
import itertools
import statsmodels.api as sm
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
import seaborn as sns
sns.set_context("paper", font_scale=1.3)
sns.set_style('white')
import math
from sklearn.preprocessing import MinMaxScaler
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

IMPORTING DATA

CRUDE_OIL_PRICE_PREDICTION

dataplatform.cloud.ibm.com/analytics/notebooks/v2/b5204dce-8855-4f18-9253-39228b9db75a/view?projectid=9ff555e5-89ad-4660-a966-2b56770310a9&co...

IBM Watson Studio | Search in your workspaces | Buy | GOKUL RAJ K's Account | Dallas | GK

Projects / Models / CRUDE_OIL_PRICE_PREDICTION

IMPORTING DATA

In [2]:
```
dateparse = lambda x: pd.datetime.strptime(x, '%b %d, %Y')
#Read csv file
from google.colab import files
uploaded = files.upload()
```

Choose Files | No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Crude Oil Prices Daily.xlsx to Crude Oil Prices Daily.xlsx
```

In [8]:
```
import io
# df = pd.read_excel(io.BytesIO(uploaded['Crude Oil Prices Daily.xlsx']))
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='erek5dnok6ft-bwrlp26amjb-h4sii4agmuue8ham_sp',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'models-donotdelete-pr-6ymjuwbxnhbwf3'
object_key = 'Crude Oil Prices Daily.xlsx'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']

df_data_0 = pd.read_excel(body.read())
df_data_0.head()
df=df_data_0
df[:10]
```

Search | ENG IN | 23:41 19-11-2022

```
df=df_data_0
df[:10]
```

Out[8]:

	Date	Closing Value
0	1986-01-02	25.56
1	1986-01-03	26.00
2	1986-01-06	26.53
3	1986-01-07	25.85
4	1986-01-08	25.87
5	1986-01-09	26.03
6	1986-01-10	25.65
7	1986-01-13	25.08
8	1986-01-14	24.97
9	1986-01-15	25.18

In [9]:

```
#Sort dataset by column Date
df = df.sort_values('Date')
df = df.groupby('Date')['Closing Value'].sum().reset_index()
df.set_index('Date', inplace=True)
df=df.loc[datetime.date(year=2000,month=1,day=1):]
```

In [10]:

```
df.head()
```

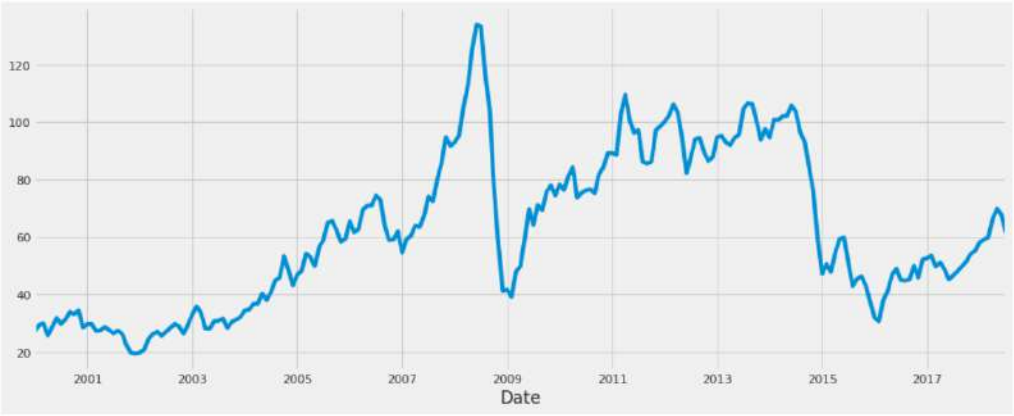
Out[10]:

Date	Closing Value
2000-01-04	25.56
2000-01-05	24.65

```
               '2018-06-26', '2018-06-27', '2018-06-28', '2018-06-29',
               '2018-07-02', '2018-07-03', '2018-07-04', '2018-07-05',
               '2018-07-06', '2018-07-09'],
              dtype='datetime64[ns]', name='Date', length=4673, freq=None)
```

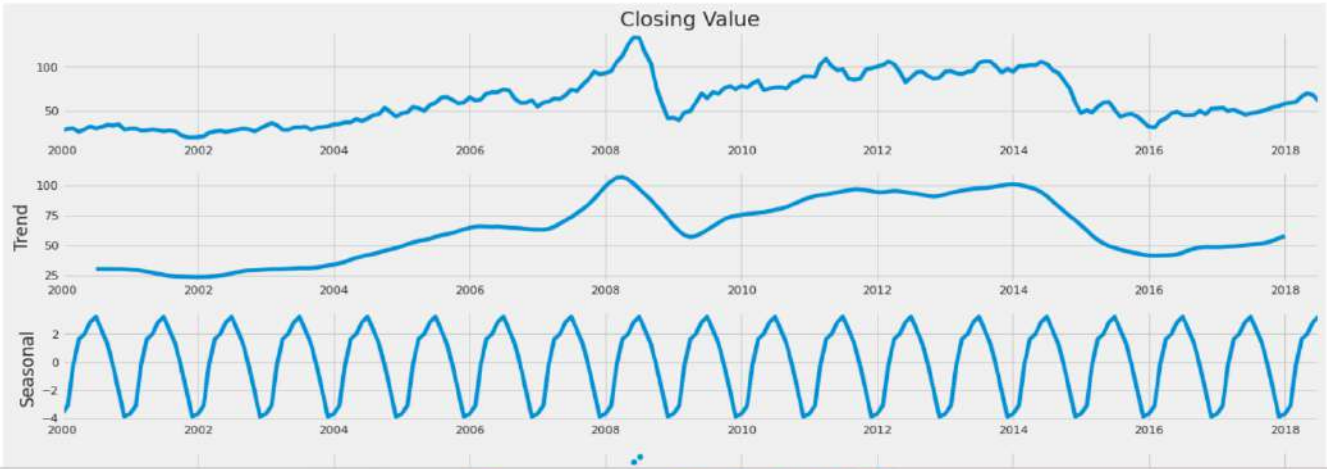
In [14]:
```
y = df['Closing Value'].resample('MS').mean()
```

In [15]:
```
y.plot(figsize=(15, 6))
plt.show()
```



In [16]:
```
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
```

In [16]:
```
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```



CRUDE_OIL_PRICE_PREDICTION

dataplatform.cloud.ibm.com/analytics/notebooks/v2/b5204dce-8855-4f18-9253-39228b9db75a/view?projectid=9ff555e5-89ad-4660-a966-2b56770310a9&co...

IBM Watson Studio | Search in your workspaces | Buy | GOKUL RAJ K's Account | Dallas

Projects / Models / CRUDE_OIL_PRICE_PREDICTION

```
X_train,Y_train=create_data_set(train,look_back)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test,Y_test=create_data_set(test,look_back)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

LSTM LAYER

In [21]:

```
regressor = Sequential()

regressor.add(LSTM(units = 60, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 60, return_sequences = True))
regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 60))
regressor.add(Dropout(0.1))

regressor.add(Dense(units = 1))


regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
reduce_lr = ReduceLROnPlateau(monitor='val_loss',patience=5)
history =regressor.fit(X_train, Y_train, epochs = 20, batch_size = 15,validation_data=(X_test, Y_test), callbacks=[reduce_lr],shuffle=False)
```

```
Epoch 1/20
212/212 [==============================] - 23s 88ms/step - loss: 0.0047 - val_loss: 0.0251 - lr: 0.0010
Epoch 2/20
212/212 [==============================] - 17s 82ms/step - loss: 0.0122 - val_loss: 0.0478 - lr: 0.0010
Epoch 3/20
212/212 [==============================] - 17s 82ms/step - loss: 0.0115 - val_loss: 0.0505 - lr: 0.0010
Epoch 4/20
212/212 [==============================] - 17s 81ms/step - loss: 0.0163 - val_loss: 0.0461 - lr: 0.0010
Epoch 5/20
212/212 [==============================] - 19s 91ms/step - loss: 0.0193 - val_loss: 0.0461 - lr: 0.0010
Epoch 6/20
```

Search | ENG IN | 23:41 19-11-2022

epochs

In [25]:
```
aa=[x for x in range(180)]
plt.figure(figsize=(8,4))
plt.plot(aa, Y_test[0][:180], marker='.', label="actual")
plt.plot(aa, test_predict[:,0][:180], 'r', label="prediction")
plt.tight_layout()
sns.despine(top=True)
plt.subplots_adjust(left=0.07)
plt.ylabel('Price', size=15)
plt.xlabel('Time step', size=15)
plt.legend(fontsize=15)
plt.show();
```