In [1]:

```
"""
Load the data
"""


import pandas as pd

ydf=pd.read_csv('/home/harshit/Downloads/YESBANK.csv')
ydf
```

Out[1]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-01-16	334.000000	338.500000	328.000000	333.899994	319.873657	470267.0
1	2018-01-17	336.000000	343.750000	331.250000	342.500000	328.112457	653618.0
2	2018-01-18	350.000000	356.500000	333.100006	340.250000	325.956970	2419109.0
3	2018-01-19	348.000000	352.000000	339.250000	348.299988	333.668793	1659646.0
4	2018-01-22	349.000000	358.000000	349.000000	355.250000	340.326874	663569.0
...	...	...	...	...	...	...	...
484	2020-01-09	47.150002	48.450001	46.299999	47.299999	47.299999	6835915.0
485	2020-01-10	47.599998	48.349998	43.900002	44.799999	44.799999	15918973.0
486	2020-01-13	43.400002	44.000000	41.200001	42.099998	42.099998	10763969.0
487	2020-01-14	41.750000	41.750000	36.549999	38.549999	38.549999	18250917.0
488	2020-01-15	38.549999	41.099998	36.650002	39.799999	39.799999	19876620.0

489 rows × 7 columns

In [2]:

```
#import LinearRegression Class
from sklearn.linear_model import LinearRegression
```

In [3]:

```
#Algorithm---------------->step by step solution to a problem
#training

#model-------------->system of prediction which will give you your final output/answer

#feature!!!
```

In [4]:

```
#step 1------------> select your target Attribute
```

In [5]:

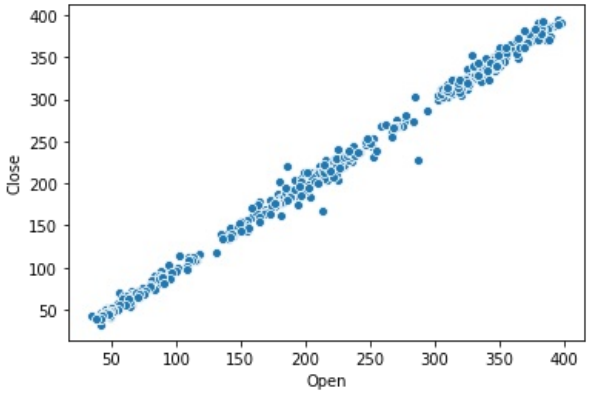
```
import seaborn as sns
```

In [6]:

```
sns.scatterplot(x='Open',y='Close',data=ydf)
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f819efc3430>
```

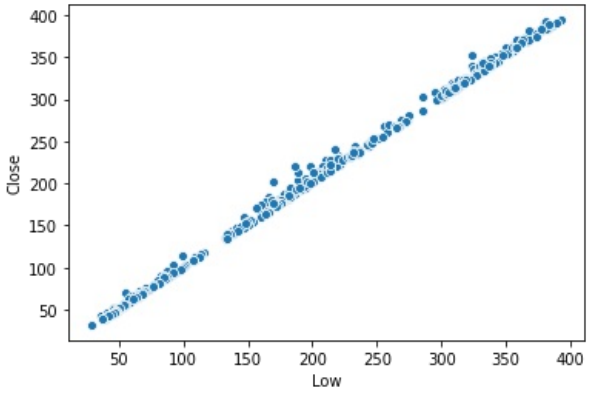


In [7]:

```
sns.scatterplot(x='Low',y='Close',data=ydf)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f819c8b19a0>
```



In [8]:

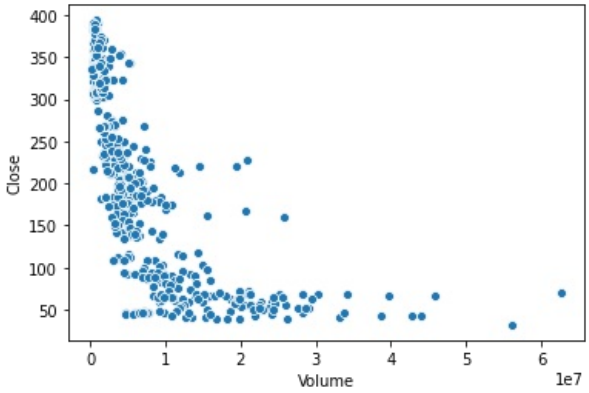
```
# Univariate Linear Regression
```

In [9]:

```
sns.scatterplot(x='Volume',y='Close',data=ydf)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f819c8c2160>
```



In [10]:

```
ydf.isna().sum() #check for missing values
```

Out[10]:

```
Date         0
Open         1
High         1
Low          1
Close        1
Adj Close    1
Volume       1
dtype: int64
```

In [11]:

```
ydf.dropna(axis=0,how='any',inplace=True) #drop missing values
```

In [12]:

```
ydf.isna().sum()
```

Out[12]:

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [13]:

```
#separate the column which is target
target=ydf.pop('Close')
ydf
```

Out[13]:

	Date	Open	High	Low	Adj Close	Volume
0	2018-01-16	334.000000	338.500000	328.000000	319.873657	470267.0
1	2018-01-17	336.000000	343.750000	331.250000	328.112457	653618.0
2	2018-01-18	350.000000	356.500000	333.100006	325.956970	2419109.0
3	2018-01-19	348.000000	352.000000	339.250000	333.668793	1659646.0
4	2018-01-22	349.000000	358.000000	349.000000	340.326874	663569.0
...	...	...	...	...	...	...
484	2020-01-09	47.150002	48.450001	46.299999	47.299999	6835915.0
485	2020-01-10	47.599998	48.349998	43.900002	44.799999	15918973.0
486	2020-01-13	43.400002	44.000000	41.200001	42.099998	10763969.0
487	2020-01-14	41.750000	41.750000	36.549999	38.549999	18250917.0
488	2020-01-15	38.549999	41.099998	36.650002	39.799999	19876620.0

488 rows × 6 columns

In [14]:

```
target
```

Out[14]:

```
0      333.899994
1      342.500000
2      340.250000
3      348.299988
4      355.250000
          ...
484     47.299999
485     44.799999
486     42.099998
487     38.549999
488     39.799999
Name: Close, Length: 488, dtype: float64
```

In [15]:

```
feature=ydf[['Open','High','Low']] #feature to be used on x-axis
feature
```

Out[15]:

	Open	High	Low
0	334.000000	338.500000	328.000000
1	336.000000	343.750000	331.250000
2	350.000000	356.500000	333.100006
3	348.000000	352.000000	339.250000
4	349.000000	358.000000	349.000000
...	...	...	...
484	47.150002	48.450001	46.299999
485	47.599998	48.349998	43.900002
486	43.400002	44.000000	41.200001
487	41.750000	41.750000	36.549999
488	38.549999	41.099998	36.650002

488 rows × 3 columns

In [16]:

```
from sklearn.model_selection import train_test_split

#split data into training and testing set
xtrain,xtest,ytraining,ytest=train_test_split(feature ,target,test_size=0.2)
```

In [17]:

```
xtrain
```

Out[17]:

	Open	High	Low
259	185.600006	185.600006	175.449997
27	328.000000	328.000000	319.600006
427	41.900002	48.200001	40.650002
94	343.000000	343.000000	334.100006
54	315.149994	319.899994	314.500000
...	...	...	...
280	236.000000	236.000000	228.800003
385	84.000000	84.349998	72.849998
257	202.000000	203.500000	190.350006
240	188.000000	193.199997	185.000000
443	69.900002	73.550003	69.199997

390 rows × 3 columns

In [18]:

```
ytraining
```

Out[18]:

```
259    179.899994
27     325.399994
427     47.400002
94     336.700012
54     316.100006
          ...
280    231.800003
385     73.599998
257    194.300003
240    192.300003
443     73.000000
Name: Close, Length: 390, dtype: float64
```

In [19]:

```
#predictor model


model=LinearRegression()
```

In [20]:

```
# import numpy as np


# #reshape the data in 1 column layout
# xtrain=np.reshape(np.array(xtrain),(-1,1))
# ytraining=np.reshape(np.array(ytraining),(-1,1))
```

In [21]:

```
#Training Model


model.fit(xtrain,ytraining) #training process!!!!!
```

Out[21]:

```
LinearRegression()
```

In [22]:

```
xtest
```

Out[22]:

	Open	High	Low
272	214.899994	223.500000	213.399994
232	183.000000	183.550003	177.300003
28	326.600006	333.700012	325.250000
218	169.449997	172.399994	167.500000
271	219.000000	219.100006	214.199997
...	...	...	...
23	312.700012	316.000000	307.799988
125	382.700012	387.750000	378.049988
407	67.050003	68.849998	64.699997
39	318.899994	320.799988	311.200012
228	179.000000	187.550003	178.000000

98 rows × 3 columns

In [23]:

```
ytest
```

Out[23]:

```
272    221.949997
232    178.550003
28     327.049988
218    168.300003
271    215.000000
          ...
23     309.299988
125    383.799988
407     65.150002
39     312.399994
228    186.649994
Name: Close, Length: 98, dtype: float64
```

In [24]:

```
#reshapte testing x values as well
# xtest=np.reshape(np.array(xtest),(-1,1))
values=model.predict(xtest)
```

In [25]:

```
from sklearn.metrics import mean_squared_error
mean_squared_error(ytest,values)
```

Out[25]:

```
4.375502349871036
```

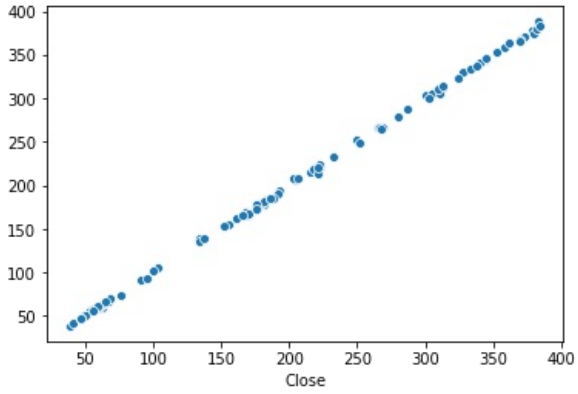
In [26]:

```
import seaborn as sns

sns.scatterplot(ytest,values) #plot for
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f819abe20d0>
```



In [ ]:

```
-IN case of continuous numbers

- To estimate / predict the value of a traget column
```

In [ ]:

```
Gender Pclass Embarked Survived
M       2         'C'        0
F       2          'S'       1
M       3         'F'       0
```

In [ ]:

```
                     Gender
```

In [ ]:

```
Gender Pclass Embarked                                                  Survived
M       2          'C'                                                     0
F       2          'S'                                                     1
M       3          'C'                                                     1
```

In [ ]:

```
F      3     'C'


                                    GENDER

                               FEMALE              MALE

                          [SURVIVED]             PCLASS

                                              2                          3

                                        [NOT SURVIVED]             [SURVIVED]
```