```
import pandas as pd
df=pd.read_csv("/content/GOOG.csv")
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2022-08-08	119.120003	120.860001	117.830002	118.139999	118.139999	17061100
1	2022-08-09	117.989998	118.199997	116.559998	117.500000	117.500000	15424300
2	2022-08-10	119.589996	121.779999	119.360001	120.650002	120.650002	20497000
3	2022-08-11	122.080002	122.339996	119.550003	119.820000	119.820000	16671600
4	2022-08-12	121.160004	122.650002	120.400002	122.650002	122.650002	16121100
...	...	...	...	...	...	...	...
245	2023-07-31	133.009995	133.830002	132.130005	133.110001	133.110001	18381900
246	2023-08-01	130.854996	132.919998	130.750000	131.889999	131.889999	22154300
247	2023-08-02	129.839996	130.419998	127.849998	128.639999	128.639999	22705800
248	2023-08-03	128.369995	129.770004	127.775002	128.770004	128.770004	15018100
249	2023-08-04	129.600006	131.929993	128.315002	128.539993	128.539993	20509500

250 rows × 7 columns

```
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2022-08-08	119.120003	120.860001	117.830002	118.139999	118.139999	17061100
1	2022-08-09	117.989998	118.199997	116.559998	117.500000	117.500000	15424300
2	2022-08-10	119.589996	121.779999	119.360001	120.650002	120.650002	20497000
3	2022-08-11	122.080002	122.339996	119.550003	119.820000	119.820000	16671600
4	2022-08-12	121.160004	122.650002	120.400002	122.650002	122.650002	16121100

```
df.tail()
```

	Date	Open	High	Low	Close	Adj Close	Volume
245	2023-07-31	133.009995	133.830002	132.130005	133.110001	133.110001	18381900
246	2023-08-01	130.854996	132.919998	130.750000	131.889999	131.889999	22154300
247	2023-08-02	129.839996	130.419998	127.849998	128.639999	128.639999	22705800
248	2023-08-03	128.369995	129.770004	127.775002	128.770004	128.770004	15018100
249	2023-08-04	129.600006	131.929993	128.315002	128.539993	128.539993	20509500

```
df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       250 non-null    object 
 1   Open       250 non-null    float64
 2   High       250 non-null    float64
 3   Low        250 non-null    float64
 4   Close      250 non-null    float64
 5   Adj Close  250 non-null    float64
 6   Volume     250 non-null    int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 13.8+ KB
```

```
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	250.000000	250.000000	250.000000	250.000000	250.000000	2.500000e+02
mean	105.696616	107.172900	104.519432	105.829780	105.829780	2.707343e+07
std	12.367202	12.436379	12.389783	12.388743	12.388743	1.111356e+07
min	85.510002	86.550003	83.449997	83.489998	83.489998	8.567800e+06
25%	95.749998	97.344997	94.440003	95.835001	95.835001	2.063240e+07
50%	102.799999	104.205002	101.857502	103.549999	103.549999	2.423745e+07
75%	117.929998	119.688748	116.782501	118.070002	118.070002	3.022488e+07

```
df.isna().sum()
```

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

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

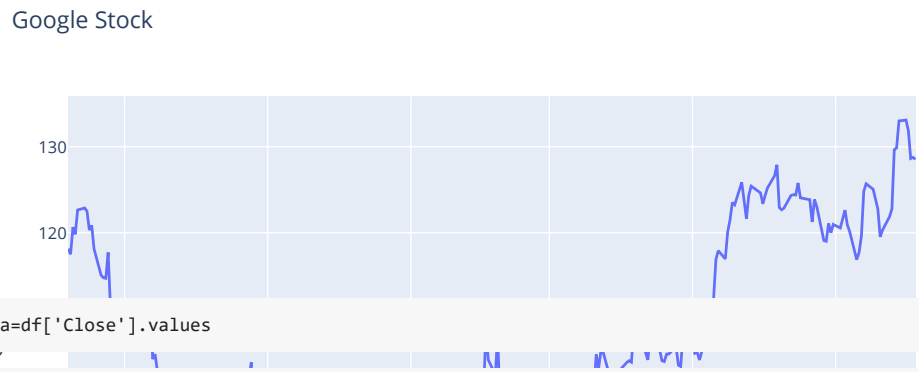
```
df.set_index(df['Date'],inplace=True)
df.drop(['Open','High','Low','Volume'],axis=1,inplace=True)
df
```

	Date	Close	Adj Close
Date			
2022-08-08	2022-08-08	118.139999	118.139999
2022-08-09	2022-08-09	117.500000	117.500000
2022-08-10	2022-08-10	120.650002	120.650002
2022-08-11	2022-08-11	119.820000	119.820000
2022-08-12	2022-08-12	122.650002	122.650002
...	...	...	...
2023-07-31	2023-07-31	133.110001	133.110001
2023-08-01	2023-08-01	131.889999	131.889999
2023-08-02	2023-08-02	128.639999	128.639999
2023-08-03	2023-08-03	128.770004	128.770004
2023-08-04	2023-08-04	128.539993	128.539993

250 rows × 3 columns

```
import plotly.graph_objects as go
```

```
tracel=go.Scatter(x=df['Date'],y=df['Close'],mode='lines',name='Data')
layout=go.Layout(title='Google Stock',xaxis={'title':"Date"},yaxis={'title':'Close'})
fig=go.Figure(data=[tracel],layout=layout)
fig.show()
```




```
close_Data=df['Close'].values
```

```
close_Data
```

```
array([118.139999, 117.5     , 120.650002, 119.82    , 122.650002,
       122.879997, 122.510002, 120.32    , 120.860001, 118.120003,
       115.07    , 114.769997, 114.699997, 117.699997, 111.300003,
       110.339996, 109.910004, 109.150002, 110.550003, 108.68    ,
       107.480003, 110.480003, 109.419998, 111.779999, 111.870003,
       105.309998, 105.870003, 103.900002, 103.629997, 103.849998,
       101.830002, 100.010002, 100.57    ,  99.169998,  98.809998,
        98.089996, 100.739998,  98.089996,  96.150002,  99.300003,
       102.410004, 102.220001, 102.239998,  99.57    ,  98.709999,
        98.050003,  98.300003,  99.709999,  97.18    , 100.779999,
       101.389999, 100.290001, 100.529999, 101.480003, 102.970001,
       104.93    ,  94.82    ,  92.599998,  96.580002,  94.660004,
        90.5     ,  87.07    ,  83.489998,  86.699997,  88.650002,
        88.910004,  87.400002,  94.169998,  96.730003,  96.029999,
        98.720001,  98.989998,  98.5     ,  97.800003,  95.830002,
        97.330002,  98.82    ,  97.599998,  96.25    ,  95.440002,
       101.449997, 101.279999, 100.830002,  99.870003,  97.309998,
        95.150002,  93.949997,  93.07    ,  93.559998,  95.849998,
        95.309998,  91.199997,  90.860001,  89.150002,  89.629997,
        90.25    ,  88.260002,  89.809998,  87.93    ,  86.459999,
        88.949997,  88.730003,  89.699997,  88.709999,  86.769997,
        88.160004,  88.800003,  89.239998,  92.260002,  91.910004,
        92.800003,  92.160004,  91.779999,  93.910004,  99.279999,
       101.209999,  99.209999,  96.730003,  99.160004, 100.709999,
        97.949997,  99.870003, 101.43    , 108.800003, 105.220001,
       103.470001, 108.040001, 100.      ,  95.459999,  94.860001,
        95.      ,  94.949997,  97.099998,  95.779999,  94.589996,
        92.050003,  91.800003,  91.07    ,  89.349998,  90.099998,
        90.300003,  90.510002,  92.309998,  94.019997,  95.580002,
        94.169998,  94.650002,  92.660004,  91.010002,  91.660004,
        94.25    ,  96.550003, 101.07    , 102.459999, 101.93    ,
       105.839996, 104.220001, 106.260002, 106.059998, 103.059998,
       101.360001, 101.900002, 101.32    , 104.      , 104.910004,
       105.120003, 104.949997, 108.900002, 106.949997, 106.120003,
       105.220001, 108.190002, 109.459999, 106.419998, 105.120003,
       105.019997, 105.900002, 105.910004, 106.779999, 104.610001,
       104.449997, 108.370003, 108.220001, 107.709999, 105.980003,
       106.120003, 105.209999, 106.214996, 108.239998, 107.940002,
       112.279999, 116.900002, 117.919998, 116.959999, 120.089996,
       121.480003, 123.519997, 123.25    , 125.870003, 123.290001,
       121.639999, 124.349998, 125.43    , 124.639999, 123.370003,
       124.370003, 125.230003, 126.629997, 127.910004, 122.940002,
       122.669998, 122.870003, 124.349998, 124.43    , 124.379997,
       125.790001, 124.059998, 123.849998, 121.260002, 123.870003,
       123.019997, 119.089996, 119.010002, 121.080002, 120.010002,
       120.970001, 120.559998, 122.629997, 120.93    , 120.139999,
       116.870003, 117.709999, 119.620003, 124.830002, 125.699997,
       125.059998, 124.080002, 122.779999, 119.529999, 120.309998,
       121.879997, 122.790001, 129.660004, 129.869995, 133.009995,
       133.110001, 131.889999, 128.639999, 128.770004, 128.539993])
```

```
close_Data=close_Data.reshape(-1,1) #-1 means we dont know row counts and 1 means only one column will be output
close_Data
```

```
            [ 94.82    ],
            [ 92.599998],
            [ 96.580002],
            [ 94.660004],
            [ 90.5     ],
            [ 87.07    ],
            [ 83.489998],
            [ 86.699997],
            [ 88.650002],
            [ 88.910004],
            [ 87.400002],
            [ 94.169998],
            [ 96.730003],
            [ 96.029999],
            [ 98.720001],
            [ 98.989998],
            [ 98.5     ],
            [ 97.800003],
            [ 95.830002],
            [ 97.330002],
            [ 98.82    ],
            [ 97.599998],
            [ 96.25    ],
            [ 95.440002],
            [101.449997],
            [101.279999],
            [100.830002],
            [ 99.870003],
            [ 97.309998],
            [ 95.150002],
            [ 93.949997],
            [ 93.07    ],
            [ 93.559998],
            [ 95.849998],
            [ 95.309998],
            [ 91.199997],
            [ 90.860001],
            [ 89.150002],
            [ 89.629997],
            [ 90.25    ],
            [ 88.260002],
            [ 89.809998],
            [ 87.93    ],
            [ 86.459999],
            [ 88.949997],
            [ 88.730003],
```

```
split_percentage=0.80 #used for spliing the data set into training and testing
split=int(split_percentage*len(close_Data))
```

```
split
```

```
200
```

```
close_train=close_Data[:split]
close_test=close_Data[split:]
```

```
date_train=df['Date'][:split]
date_test=df["Date"][split:]
```

```
print(len(close_train))
```

```
200
```

```
print(len(close_test))
```

```
50
```

```
from keras.preprocessing.sequence import TimeseriesGenerator
```

```
look_back=15
train_genarator=TimeseriesGenerator(close_train,close_train,length=look_back,batch_size=20) #close_train is both input and output
test_genarator=TimeseriesGenerator(close_test,close_test,length=look_back,batch_size=1)
```

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

```
model=Sequential()
model.add(
    LSTM(10,activation='relu',input_shape=(look_back,1))
)
model.add(Dense(1)
)
```

```
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel
```

```
model.compile(optimizer='adam',loss='mse')
```

```
model.fit_generator(train_genarator,epochs=100,)
```

```
<ipython-input-87-8f2881aeba82>:1: UserWarning:

`Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generato

Epoch 1/100
10/10 [==============================] - 2s 58ms/step - loss: 9367.5420
Epoch 2/100
10/10 [==============================] - 0s 29ms/step - loss: 9195.4951
Epoch 3/100
10/10 [==============================] - 0s 28ms/step - loss: 8952.8262
Epoch 4/100
10/10 [==============================] - 0s 29ms/step - loss: 4510.4404
Epoch 5/100
10/10 [==============================] - 0s 30ms/step - loss: 140.5448
Epoch 6/100
10/10 [==============================] - 0s 32ms/step - loss: 122.8043
Epoch 7/100
10/10 [==============================] - 0s 29ms/step - loss: 87.4980
Epoch 8/100
10/10 [==============================] - 0s 36ms/step - loss: 28.8179
Epoch 9/100
10/10 [==============================] - 0s 32ms/step - loss: 19.6674
Epoch 10/100
10/10 [==============================] - 0s 30ms/step - loss: 19.7965
Epoch 11/100
10/10 [==============================] - 0s 28ms/step - loss: 13.8567
Epoch 12/100
10/10 [==============================] - 0s 26ms/step - loss: 11.1368
Epoch 13/100
10/10 [==============================] - 0s 28ms/step - loss: 9.9028
Epoch 14/100
10/10 [==============================] - 0s 28ms/step - loss: 9.9213
Epoch 15/100
10/10 [==============================] - 0s 29ms/step - loss: 10.4986
Epoch 16/100
10/10 [==============================] - 0s 28ms/step - loss: 8.2055
Epoch 17/100
10/10 [==============================] - 0s 32ms/step - loss: 6.9885
Epoch 18/100
10/10 [==============================] - 0s 30ms/step - loss: 6.6960
Epoch 19/100
10/10 [==============================] - 0s 28ms/step - loss: 7.2236
Epoch 20/100
10/10 [==============================] - 0s 28ms/step - loss: 6.4092
Epoch 21/100
10/10 [==============================] - 0s 28ms/step - loss: 6.2445
Epoch 22/100
10/10 [==============================] - 0s 32ms/step - loss: 6.3545
Epoch 23/100
10/10 [==============================] - 0s 31ms/step - loss: 6.2501
Epoch 24/100
10/10 [==============================] - 0s 30ms/step - loss: 6.1379
Epoch 25/100
10/10 [==============================] - 0s 28ms/step - loss: 6.4521
Epoch 26/100
10/10 [==============================] - 1s 53ms/step - loss: 6.5180
Epoch 27/100
```

```
prediction=model.predict_generator(test_genarator)
prediction
```

```
<ipython-input-88-8b5c33933389>:1: UserWarning:

`Model.predict_generator` is deprecated and will be removed in a future version. Please use `Model.predict`, which supports ger

array([[125.135376],
       [127.04791 ],
       [124.332985],
```

```
       [124.28649 ],
       [120.73858 ],
       [124.88652 ],
       [123.31645 ],
       [118.06904 ],
       [118.70412 ],
       [121.71025 ],
       [119.91447 ],
       [121.44635 ],
       [120.74133 ],
       [123.74471 ],
       [121.1273  ],
       [120.257965],
       [116.12085 ],
       [117.85892 ],
       [120.46043 ],
       [126.93349 ],
       [127.66151 ],
       [126.82065 ],
       [125.499084],
       [123.667694],
       [119.25298 ],
       [120.809685],
       [122.927376],
       [123.911095],
       [131.96411 ],
       [132.16415 ],
       [135.19061 ],
       [135.49931 ],
       [134.30556 ],
       [129.90195 ],
       [130.21184 ]], dtype=float32)
```

```
model.save('LSTM_TIME_SERIES') #saving the mdoel
```

```
close_train=close_train.reshape((-1))
close_test=close_test.reshape((-1))
prediction=prediction.reshape((-1))
```

```
prediction
```

```
array([125.135376, 127.04791 , 124.332985, 124.28649 , 120.73858 ,
       124.88652 , 123.31645 , 118.06904 , 118.70412 , 121.71025 ,
       119.91447 , 121.44635 , 120.74133 , 123.74471 , 121.1273  ,
       120.257965, 116.12085 , 117.85892 , 120.46043 , 126.93349 ,
       127.66151 , 126.82065 , 125.499084, 123.667694, 119.25298 ,
       120.809685, 122.927376, 123.911095, 131.96411 , 132.16415 ,
       135.19061 , 135.49931 , 134.30556 , 129.90195 , 130.21184 ],
      dtype=float32)
```

```
trace1=go.Scatter(x=date_train,y=close_train,mode='lines',name='Data')
trace2=go.Scatter(x=date_test,y=close_test,mode='lines',name='test')
trace3=go.Scatter(x=date_test,y=prediction,mode='lines',name='prediction')
layout=go.Layout(title='google stock prediction',xaxis={'title':'Date'},yaxis={'title':'close'})
fig=go.Figure(data=[trace1,trace2,trace3],layout=layout)
fig.show()
```