```
%%html
<marquee style='width: 60%;color: red;'><b>To get latest stock price  using python </b></ma
```

To get latest stock price using python

```
pip install yfinance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting yfinance
  Downloading yfinance-0.1.77-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.7/dist-p
Collecting requests>=2.26
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB 1.4 MB/s
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: lxml>=4.5.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/d
Installing collected packages: requests, yfinance
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
Successfully installed requests-2.28.1 yfinance-0.1.77
```

```
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
today = date.today()
```

To obatail current price of TCS : we took its symbol ***TCS.NS*** means NSE data of TCS

```
d1 = today.strftime("%Y-%m-%d")
end_date = d1
d2 = date.today() - timedelta(days=100)
d2 = d2.strftime("%Y-%m-%d")
start_date = d2
```

✓ 0s completed at 2:37 PM

```
                      end=end_date,
                      progress=False)
print(manoj.tail(50))
```

```
                                  Open         High          Low        Close  \
Date
2022-08-03 00:00:00+05:30  3280.000000  3349.000000  3277.050049  3339.500000
2022-08-04 00:00:00+05:30  3350.000000  3387.899902  3307.350098  3354.949951
2022-08-05 00:00:00+05:30  3361.199951  3377.000000  3355.000000  3365.050049
2022-08-08 00:00:00+05:30  3365.000000  3378.899902  3336.000000  3374.449951
2022-08-10 00:00:00+05:30  3385.000000  3385.000000  3335.000000  3354.250000
2022-08-11 00:00:00+05:30  3400.000000  3428.699951  3375.050049  3422.500000
2022-08-12 00:00:00+05:30  3419.000000  3419.000000  3381.250000  3401.550049
2022-08-16 00:00:00+05:30  3411.000000  3414.850098  3387.500000  3392.699951
2022-08-17 00:00:00+05:30  3385.100098  3417.949951  3371.100098  3401.100098
2022-08-18 00:00:00+05:30  3390.000000  3392.000000  3362.000000  3381.250000
2022-08-19 00:00:00+05:30  3387.000000  3421.500000  3371.250000  3385.750000
2022-08-22 00:00:00+05:30  3365.000000  3384.100098  3347.399902  3354.550049
2022-08-23 00:00:00+05:30  3319.949951  3341.899902  3270.000000  3284.600098
2022-08-24 00:00:00+05:30  3292.000000  3308.000000  3250.199951  3255.350098
2022-08-25 00:00:00+05:30  3276.000000  3278.149902  3214.750000  3218.199951
2022-08-26 00:00:00+05:30  3234.300049  3257.000000  3216.800049  3222.199951
2022-08-29 00:00:00+05:30  3125.000000  3142.699951  3081.000000  3132.550049
2022-08-30 00:00:00+05:30  3155.000000  3226.500000  3142.100098  3211.149902
2022-09-01 00:00:00+05:30  3190.000000  3190.000000  3121.000000  3131.699951
2022-09-02 00:00:00+05:30  3163.000000  3163.000000  3120.300049  3130.399902
2022-09-05 00:00:00+05:30  3123.649902  3147.949951  3112.250000  3133.399902
2022-09-06 00:00:00+05:30  3135.500000  3140.850098  3106.350098  3127.050049
2022-09-07 00:00:00+05:30  3102.000000  3161.899902  3102.000000  3149.600098
2022-09-08 00:00:00+05:30  3170.000000  3183.500000  3160.100098  3169.649902
2022-09-09 00:00:00+05:30  3195.000000  3233.500000  3168.500000  3217.649902
2022-09-12 00:00:00+05:30  3239.899902  3269.800049  3225.000000  3242.949951
2022-09-13 00:00:00+05:30  3263.449951  3263.449951  3225.000000  3229.350098
2022-09-14 00:00:00+05:30  3135.000000  3141.399902  3113.800049  3120.399902
2022-09-15 00:00:00+05:30  3130.000000  3137.750000  3100.000000  3104.350098
2022-09-16 00:00:00+05:30  3076.000000  3094.350098  3000.000000  3008.699951
2022-09-19 00:00:00+05:30  3036.000000  3042.000000  2987.800049  3028.800049
2022-09-20 00:00:00+05:30  3050.000000  3079.949951  3030.000000  3040.300049
2022-09-21 00:00:00+05:30  3028.000000  3041.399902  2998.149902  3001.199951
2022-09-22 00:00:00+05:30  2990.000000  3029.949951  2979.300049  3007.399902
2022-09-23 00:00:00+05:30  3004.000000  3022.500000  2979.000000  2982.050049
2022-09-26 00:00:00+05:30  2959.850098  3025.850098  2926.100098  2994.399902
2022-09-27 00:00:00+05:30  3009.399902  3025.000000  2976.000000  3017.449951
2022-09-28 00:00:00+05:30  2980.000000  3049.949951  2980.000000  3035.649902
2022-09-29 00:00:00+05:30  3054.000000  3055.850098  2990.000000  2997.300049
2022-09-30 00:00:00+05:30  2990.850098  3019.699951  2950.100098  3004.550049
2022-10-03 00:00:00+05:30  2995.000000  3020.699951  2974.000000  2984.949951
2022-10-04 00:00:00+05:30  3029.949951  3098.000000  3023.000000  3091.149902
2022-10-06 00:00:00+05:30  3111.000000  3124.000000  3092.449951  3101.949951
2022-10-07 00:00:00+05:30  3097.399902  3105.000000  3058.100098  3064.899902
2022-10-10 00:00:00+05:30  3010.000000  3127.000000  3005.000000  3118.550049
2022-10-11 00:00:00+05:30  3100.000000  3145.000000  3053.350098  3069.550049
2022-10-12 00:00:00+05:30  3084.899902  3109.899902  3062.050049  3100.750000
```

```
2022-10-13 00:00:00+05:30  3100.750000  3110.000000  3052.350098  3103.300049
2022-10-14 00:00:00+05:30  3145.000000  3150.000000  3071.000000  3099.149902
2022-10-17 00:00:00+05:30  3072.649902  3128.399902  3071.449951  3111.750000

                                Adj Close   Volume
Date
2022-08-03 00:00:00+05:30  3330.879639  2895824
2022-08-04 00:00:00+05:30  3346.289551  2150567
2022-08-05 00:00:00+05:30  3356.363770  1106933
```

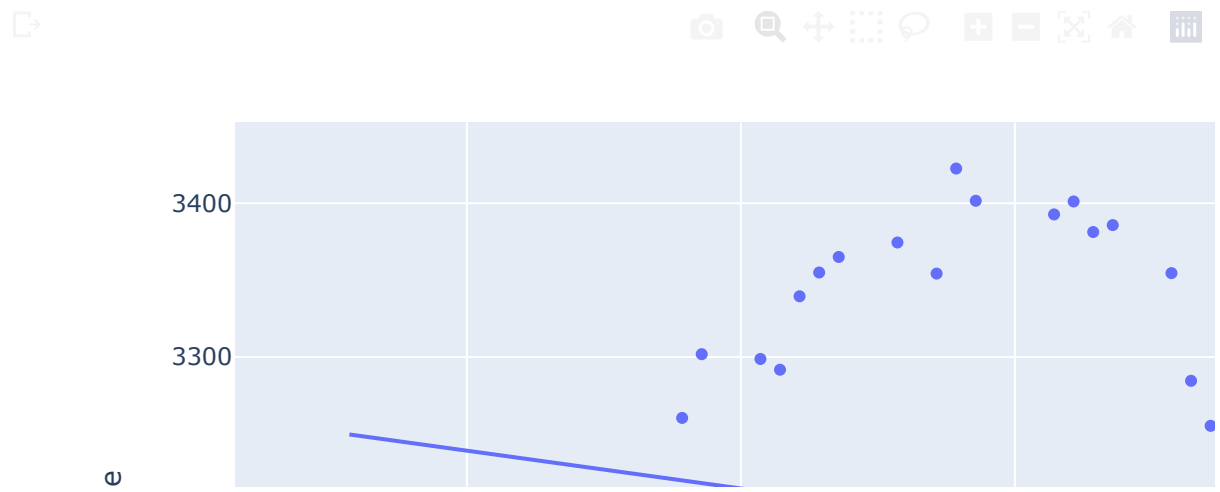
```
manoj["Date"] = manoj.index
manoj = manoj[["Date", "Open", "High",
            "Low", "Close", "Adj Close", "Volume"]]
manoj.reset_index(drop=True, inplace=True)
print(manoj.head())
```

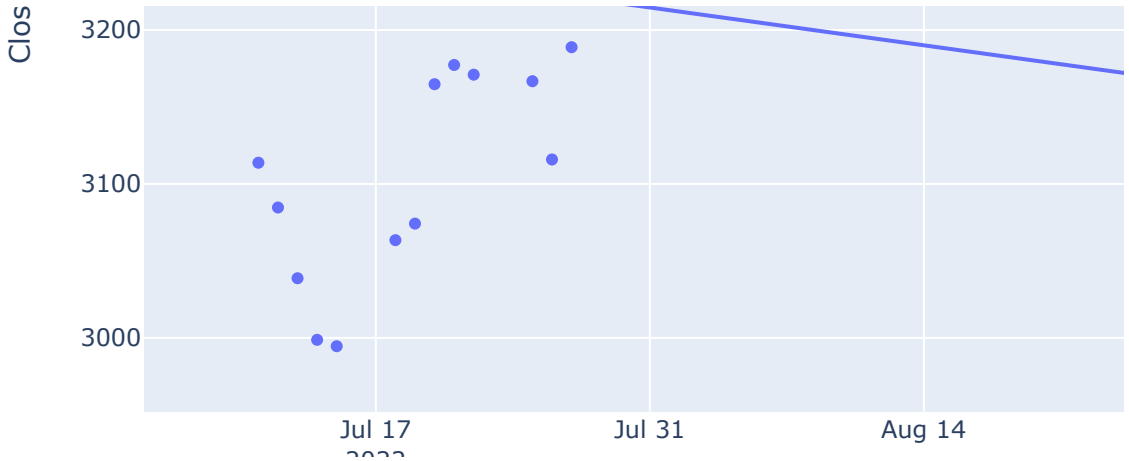
```
                       Date         Open         High      Low        Close  \
0 2022-07-11 00:00:00+05:30  3206.149902  3225.000000  3106.00  3113.800049
1 2022-07-12 00:00:00+05:30  3114.899902  3136.199951  3080.25  3084.699951
2 2022-07-13 00:00:00+05:30  3104.000000  3110.000000  3035.00  3038.750000
3 2022-07-14 00:00:00+05:30  3056.000000  3057.000000  2967.00  2998.750000
4 2022-07-15 00:00:00+05:30  3018.550049  3028.899902  2953.00  2994.600098

     Adj Close   Volume
0  3097.585938  6974600
1  3068.637207  3734815
2  3022.926514  3863530
3  2991.009033  4764908
4  2986.869873  4574806
```

plot of Date Vs Closing price of stock

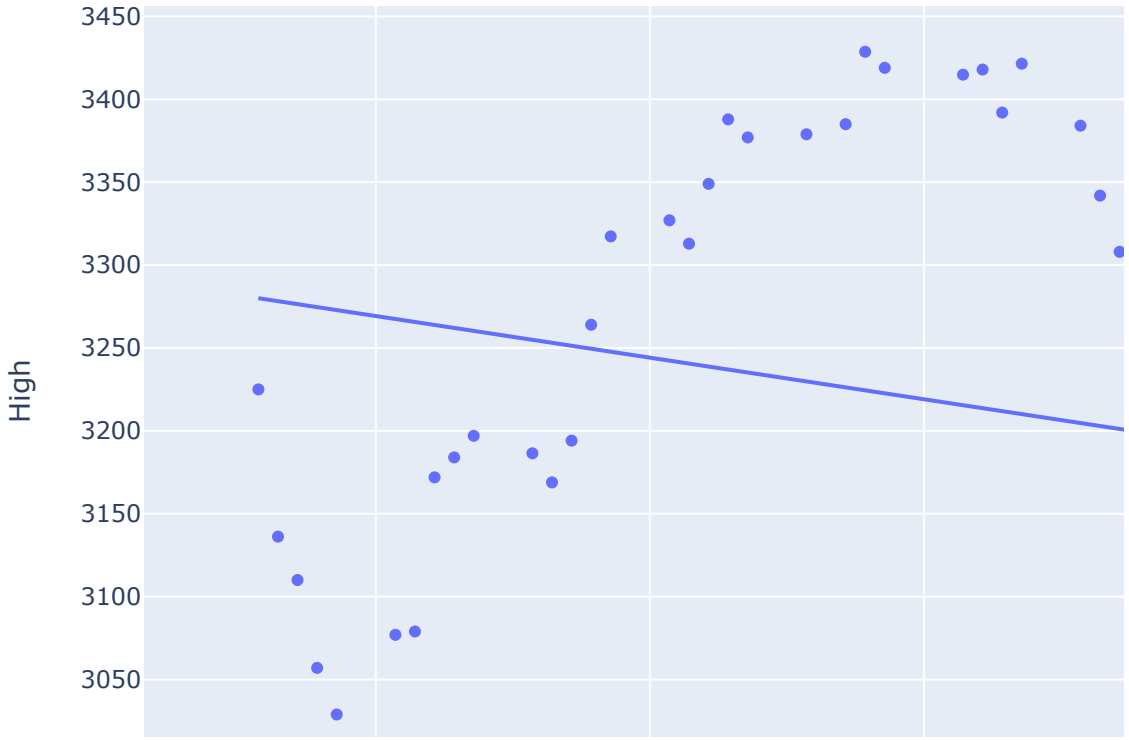
```
import plotly.express as px
import plotly.graph_objects as go
figure = px.scatter(data_frame = manoj, x="Date",
                    y="Close" ,trendline="ols")
figure.show()
```





***plot of date Vs hight price of stock ***

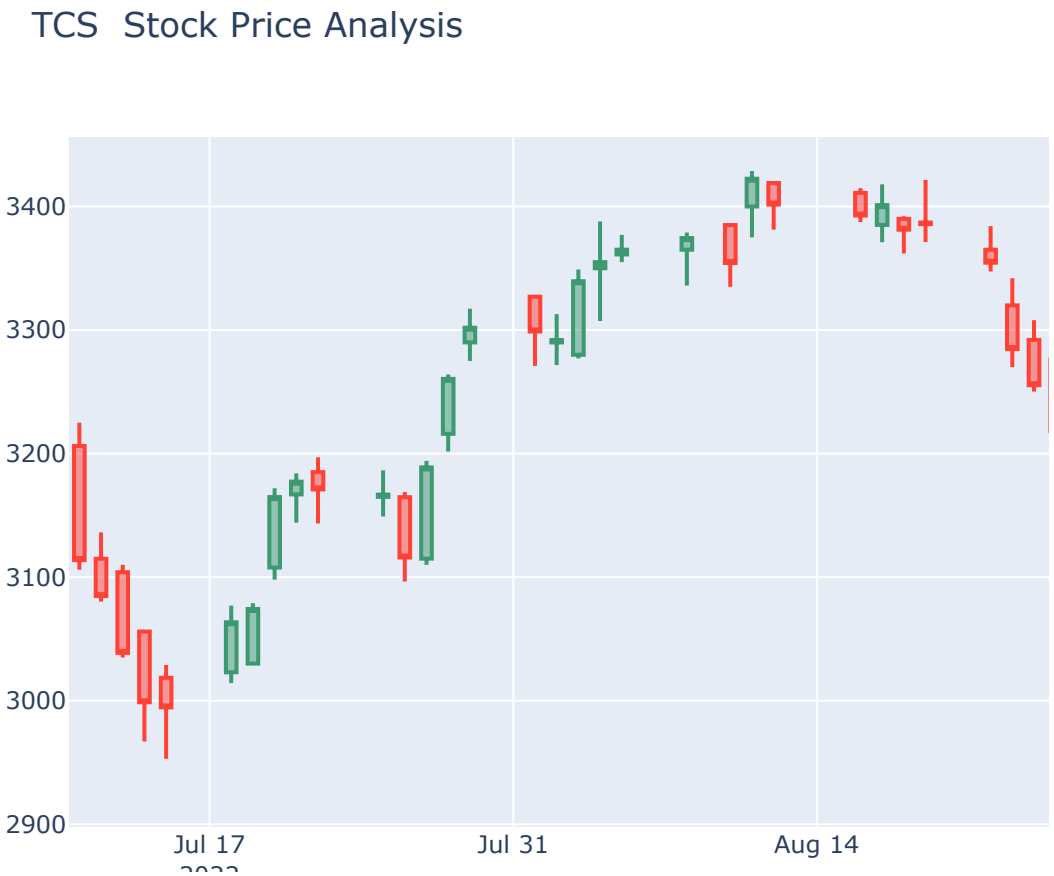
```
import plotly.express as px
import plotly.graph_objects as go
figure = px.scatter(data_frame = manoj, x="Date",
                    y="High" ,trendline="ols")
figure.show()
```



here is the link to get symbols of some important stocks of national stock exchange from yahooo finance website just repale the stock symbol like here in this program i have taken TCS stock https://finance.yahoo.com/quote/%5ENSEI/components/

***Lets see the candlestick chart of TCS stock ***

```
import plotly.graph_objects as go
figure = go.Figure(data=[go.Candlestick(x=manoj["Date"],
                                        open=manoj["Open"],
                                        high=manoj["High"],
                                        low=manoj["Low"],
                                        close=manoj["Close"])])
figure.update_layout(title = "TCS  Stock Price Analysis",
                     xaxis_rangeslider_visible=False)
figure.show()
```



```
correlation = manoj.corr()
print(correlation["Close"].sort_values(ascending=False))
```

```
Close        1.000000
Adj Close    0.999885
High         0.985750
Low          0.984472
Open         0.957477
Volume      -0.469572
Name: Close, dtype: float64
```

Lets try to predict the closing price of TCS by LSTM(long short-term memory)

```
x = manoj[["Open", "High", "Low", "Volume"]]
y = manoj["Close"]
x = x.to_numpy()
y = y.to_numpy()
y = y.reshape(-1, 1)
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

neural network architecture for LSTM

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

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 4, 128)            66560

 lstm_5 (LSTM)               (None, 64)                49408

 dense_4 (Dense)             (None, 25)                1625
```

```
 dense_5 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=30)
```

```
Epoch 1/30
53/53 [==============================] - 3s 5ms/step - loss: 9944778.0000
Epoch 2/30
53/53 [==============================] - 0s 5ms/step - loss: 9660500.0000
Epoch 3/30
53/53 [==============================] - 0s 5ms/step - loss: 9277011.0000
Epoch 4/30
53/53 [==============================] - 0s 5ms/step - loss: 8784380.0000
Epoch 5/30
53/53 [==============================] - 0s 5ms/step - loss: 8193021.5000
Epoch 6/30
53/53 [==============================] - 0s 5ms/step - loss: 7517844.0000
Epoch 7/30
53/53 [==============================] - 0s 5ms/step - loss: 6780827.0000
Epoch 8/30
53/53 [==============================] - 0s 5ms/step - loss: 6005355.5000
Epoch 9/30
53/53 [==============================] - 0s 5ms/step - loss: 5217704.0000
Epoch 10/30
53/53 [==============================] - 0s 5ms/step - loss: 4440636.5000
Epoch 11/30
53/53 [==============================] - 0s 5ms/step - loss: 3698359.0000
Epoch 12/30
53/53 [==============================] - 0s 5ms/step - loss: 3008869.7500
Epoch 13/30
53/53 [==============================] - 0s 5ms/step - loss: 2389087.0000
Epoch 14/30
53/53 [==============================] - 0s 5ms/step - loss: 1846652.5000
Epoch 15/30
53/53 [==============================] - 0s 5ms/step - loss: 1388870.3750
Epoch 16/30
53/53 [==============================] - 0s 5ms/step - loss: 1015413.5625
Epoch 17/30
53/53 [==============================] - 0s 5ms/step - loss: 720181.8125
Epoch 18/30
53/53 [==============================] - 0s 5ms/step - loss: 496857.0312
Epoch 19/30
53/53 [==============================] - 0s 5ms/step - loss: 332845.9688
Epoch 20/30
53/53 [==============================] - 0s 5ms/step - loss: 218545.4844
Epoch 21/30
53/53 [==============================] - 0s 5ms/step - loss: 141040.7031
Epoch 22/30
```

```
Epoch 22/30
53/53 [==============================] - 0s 5ms/step - loss: 91015.5391
Epoch 23/30
53/53 [==============================] - 0s 5ms/step - loss: 60040.2344
Epoch 24/30
53/53 [==============================] - 0s 5ms/step - loss: 41737.5000
Epoch 25/30
53/53 [==============================] - 0s 5ms/step - loss: 30941.4551
Epoch 26/30
53/53 [==============================] - 0s 5ms/step - loss: 24713.1738
Epoch 27/30
53/53 [==============================] - 0s 6ms/step - loss: 21422.4004
Epoch 28/30
53/53 [==============================] - 0s 5ms/step - loss: 19589.3320
Epoch 29/30
53/53 [==============================] - 0s 5ms/step - loss: 18653.4453
```

```
import numpy as np
#features = [Open, High, Low, Adj Close, Volume]
features = np.array([[3100.0896, 3190.4198, 3095.0707,3120.04578, 3834845]])
model.predict(features)
```

```
1/1 [==============================] - 0s 25ms/step
array([[3151.6848]], dtype=float32)
```

The ***predicted closing price of TCs is 3151.6848*** for a opening price of 3100.0896, day high of 3190.4198,day low of 3095.0707, previous day close of 3120.04578 and traded vloume of 3834845 shares

Colab paid products - Cancel contracts here