```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas_datareader as data
```

```
import yfinance as yf

start = '2010-01-01'
end = '2024-12-09'

# Download stock data for Apple (AAPL)
df = yf.download('AAPL', start=start, end=end)

# Display the first few rows of data
df
```

```
[*********************100%***********************]  1 of 1 completed
Price       Adj Close   Close       High        Low         Open        Volume
Ticker      AAPL        AAPL        AAPL        AAPL        AAPL        AAPL
      Date
2010-01-04    6.447413    7.643214    7.660714    7.585000    7.622500  493729600
2010-01-05    6.458558    7.656429    7.699643    7.616071    7.664286  601904800
2010-01-06    6.355827    7.534643    7.686786    7.526786    7.656429  552160000
2010-01-07    6.344078    7.520714    7.571429    7.466071    7.562500  477131200
2010-01-08    6.386254    7.570714    7.571429    7.466429    7.510714  447610800
    ...            ...         ...         ...         ...         ...        ...
2024-12-02  239.589996  239.589996  240.789993  237.160004  237.270004   48137100
2024-12-03  242.649994  242.649994  242.759995  238.899994  239.809998   38861000
2024-12-04  243.009995  243.009995  244.110001  241.250000  242.869995   44383900
2024-12-05  243.039993  243.039993  244.539993  242.130005  243.990005   40033900
2024-12-06  242.839996  242.839996  244.630005  242.080002  242.910004   36852100
3758 rows × 6 columns
```

Next steps: Generate code with df   View recommended plots   New interactive sheet

```
df.tail()
```

```
Price       Adj Close   Close       High        Low         Open        Volume
Ticker      AAPL        AAPL        AAPL        AAPL        AAPL        AAPL
      Date
2024-12-02  239.589996  239.589996  240.789993  237.160004  237.270004  48137100
2024-12-03  242.649994  242.649994  242.759995  238.899994  239.809998  38861000
2024-12-04  243.009995  243.009995  244.110001  241.250000  242.869995  44383900
2024-12-05  243.039993  243.039993  244.539993  242.130005  243.990005  40033900
2024-12-06  242.839996  242.839996  244.630005  242.080002  242.910004  36852100
```

```
df = df.reset_index()
df.head()
```

```
Price   Date        Adj Close  Close     High      Low       Open      Volume
Ticker              AAPL       AAPL      AAPL      AAPL      AAPL      AAPL
  0     2010-01-04  6.447413   7.643214  7.660714  7.585000  7.622500  493729600
  1     2010-01-05  6.458558   7.656429  7.699643  7.616071  7.664286  601904800
  2     2010-01-06  6.355827   7.534643  7.686786  7.526786  7.656429  552160000
  3     2010-01-07  6.344078   7.520714  7.571429  7.466071  7.562500  477131200
  4     2010-01-08  6.386254   7.570714  7.571429  7.466429  7.510714  447610800
```

Next steps: Generate code with df View recommended plots New interactive sheet

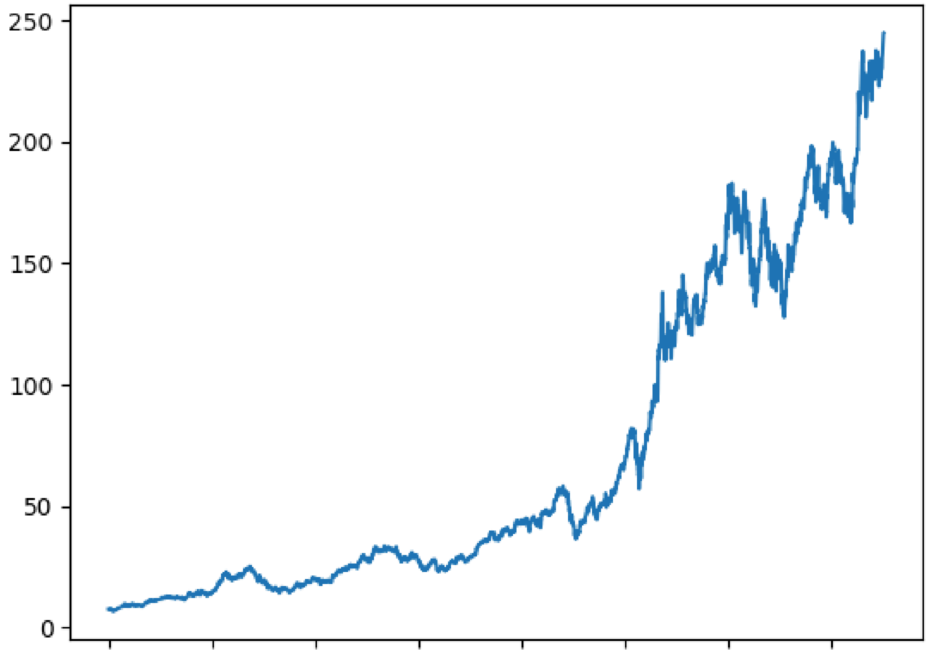
```
df=df.drop(['Date','Adj Close'],axis=1)
df.head()
```

```
<ipython-input-8-42a5f720bdfa>:1: PerformanceWarning: dropping on a non-lexsorted multi-index withou
  df=df.drop(['Date','Adj Close'],axis=1)
Price   Close     High      Low       Open      Volume
Ticker  AAPL      AAPL      AAPL      AAPL      AAPL
  0     7.643214  7.660714  7.585000  7.622500  493729600
  1     7.656429  7.699643  7.616071  7.664286  601904800
  2     7.534643  7.686786  7.526786  7.656429  552160000
  3     7.520714  7.571429  7.466071  7.562500  477131200
  4     7.570714  7.571429  7.466429  7.510714  447610800
```

Next steps: Generate code with df View recommended plots New interactive sheet

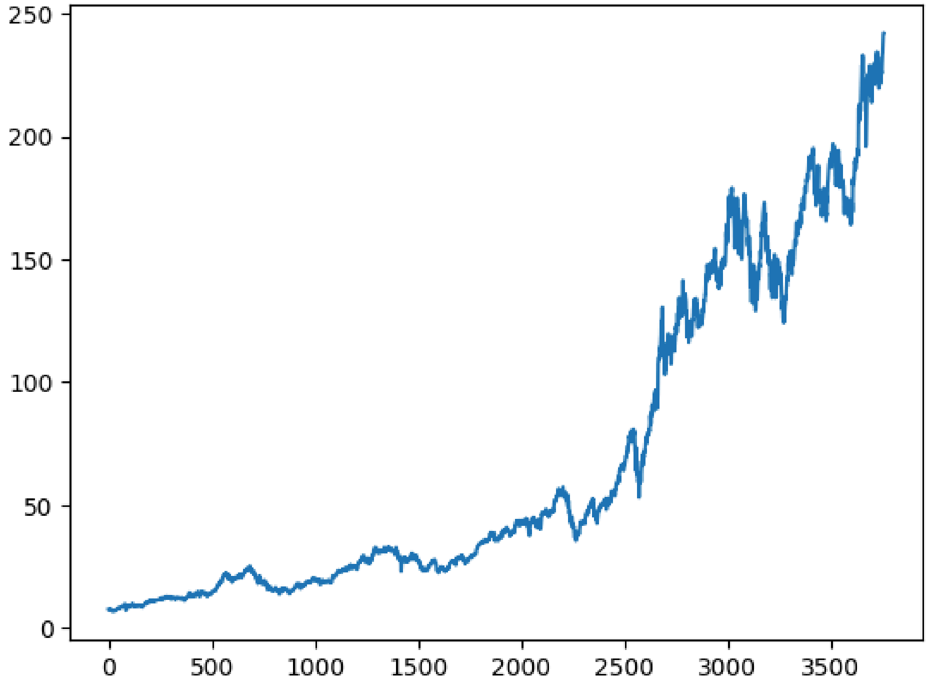
```
plt.plot(df.High)
```

```
[<matplotlib.lines.Line2D at 0x78588a4693c0>]
```



```
plt.plot(df.Low)
```

```
[<matplotlib.lines.Line2D at 0x78588a31bc40>]
```



```
#moving average
ma100 = df.Close.rolling(100).mean()
ma100
```

```
Ticker        AAPL
     0         NaN
     1         NaN
     2         NaN
     3         NaN
     4         NaN
   ...         ...
  3753    225.7750
  3754    225.8961
  3755    225.9822
  3756    226.0644
  3757    226.2040

3758 rows × 1 columns
```

Next steps: Generate code with `ma100` View recommended plots New interactive sheet

```
ma200 = df.Close.rolling(200).mean()
ma200
```

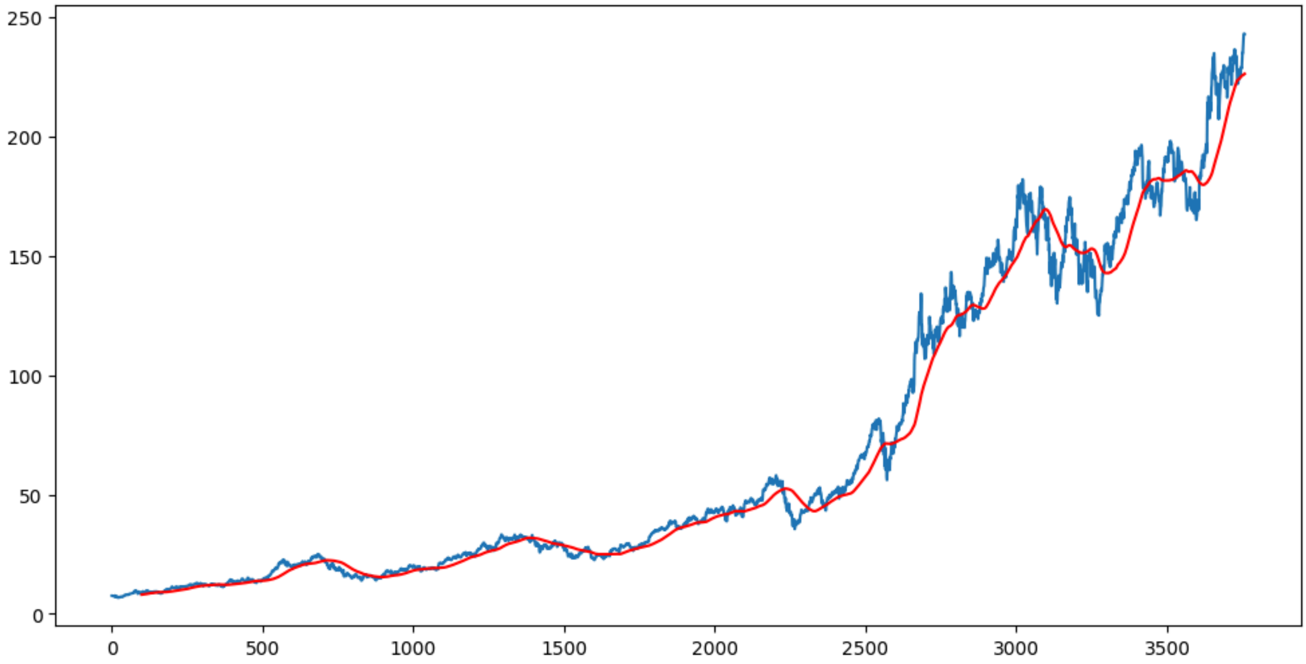
```
Ticker         AAPL
     0          NaN
     1          NaN
     2          NaN
     3          NaN
     4          NaN
   ...          ...
  3753    206.10510
  3754    206.40680
  3755    206.71405
  3756    207.01765
  3757    207.31000

3758 rows × 1 columns
```

Next steps: Generate code with `ma200` View recommended plots New interactive sheet

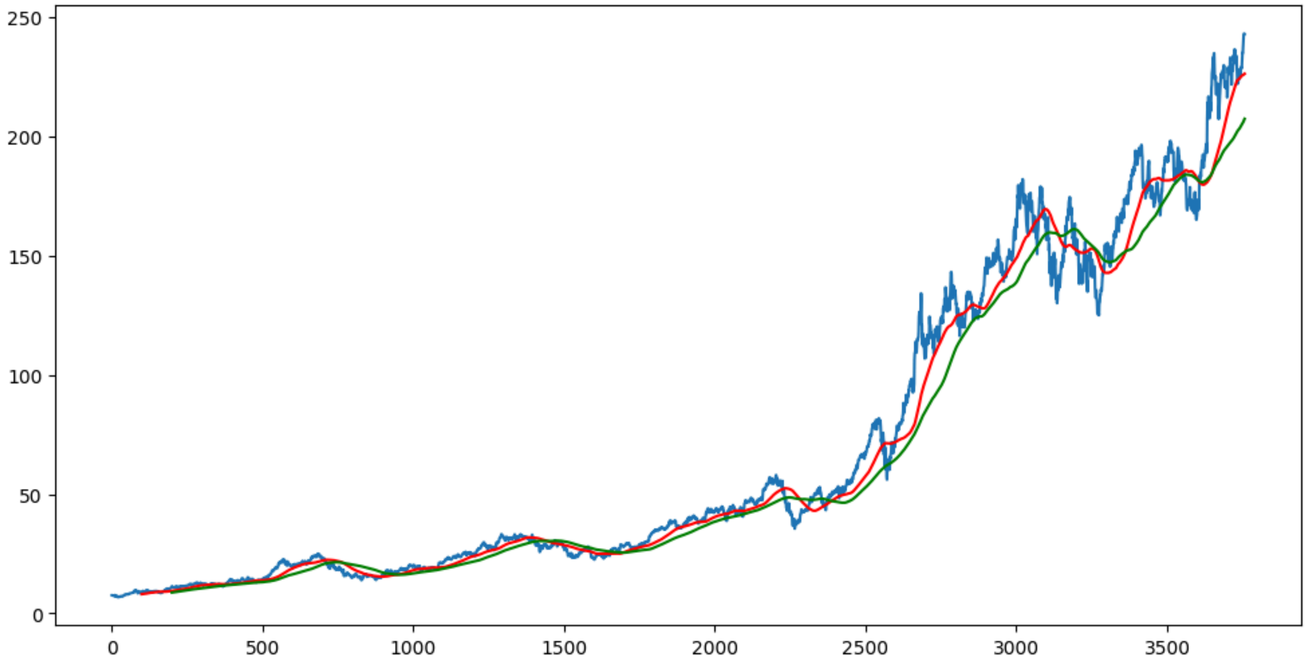
```
plt.figure(figsize=(12,6))
plt.plot(df.Close)
plt.plot(ma100,'r')
```

```
[<matplotlib.lines.Line2D at 0x7858f7104430>]
```



```
plt.figure(figsize=(12,6))
plt.plot(df.Close)
plt.plot(ma100,'r')
plt.plot(ma200,'g')
```

```
[<matplotlib.lines.Line2D at 0x7858f771e4d0>]
```



```
df.shape
```

```
(3758, 5)
```

```
#spliting data into training and testing , we are making predictions based on closing price
data_training = pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
data_testing = pd.DataFrame(df['Close'][int(len(df)*0.70):int(len(df))])
```

```
print(data_training.shape)
print(data_testing.shape)
```

```
(2630, 1)
(1128, 1)
```

```
data_training.head()
data_testing.head()
```

Ticker	AAPL
2630	88.019997
2631	87.897499
2632	87.932503
2633	87.430000
2634	89.717499

Next steps: Generate code with `data_testing` View recommended plots New interactive sheet

```
#scaling data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
```

```
data_training_array = scaler.fit_transform(data_training)
data_training_array
```

```
array([[0.00964075],
       [0.00980319],
       [0.00830615],
       ...,
       [0.94794167],
       [0.95685365],
       [0.96972994]])
```

```
#example is say we train on 10 days and we want to predict for 11th day
x_train = []
y_train = []

for i in range(100,data_training_array.shape[0]):
    x_train.append(data_training_array[i-100:i])
    y_train.append(data_training_array[i,0])

x_train,y_train = np.array(x_train),np.array(y_train)
```

```
#ml model
from keras.layers import Dense,Dropout,LSTM
from keras.models import Sequential
```

```
model = Sequential()
model.add(LSTM(units=50,activation='relu',return_sequences=True,input_shape=(x_train.shape[1],1)))
model.add(Dropout(0.2))

model.add(LSTM(units=60,activation='relu',return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(units=80,activation='relu',return_sequences=True))
model.add(Dropout(0.4))

model.add(LSTM(units=120,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(units=1))


model.summary
```

```
keras.src.models.model.Model.summary
def summary(line_length=None, positions=None, print_fn=None, expand_nested=False,
show_trainable=False, layer_range=None)

/usr/local/lib/python3.10/dist-packages/keras/src/models/model.py
Prints a string summary of the network.

Args:
    line_length: Total length of printed lines
        (e.g. set this to adapt the display to different
```

Generated code may be subject to a license | QuantMallah/ProjectCode

```
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train,epochs=50)
```

```
80/80 ━━━━━━━━━━━━━━━━━━━━ 43s 311ms/step - loss: 0.0016
Epoch 44/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 23s 287ms/step - loss: 0.0015
Epoch 45/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 25s 313ms/step - loss: 0.0013
Epoch 46/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 41s 314ms/step - loss: 0.0014
Epoch 47/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 40s 306ms/step - loss: 0.0018
Epoch 48/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 39s 287ms/step - loss: 0.0016
Epoch 49/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 25s 310ms/step - loss: 0.0014
Epoch 50/50
80/80 ━━━━━━━━━━━━━━━━━━━━ 41s 313ms/step - loss: 0.0013
<keras.src.callbacks.history.History at 0x78587f935f90>
```

```
model.save('ml_project.h5')
```

```
ive Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.ke
```

```
data_testing.head()
```

Ticker	AAPL
2630	88.019997
2631	87.897499
2632	87.932503
2633	87.430000
2634	89.717499

Next steps: Generate code with data_testing View recommended plots New interactive sheet

```
#to predict the next value we need the value of the previous 100 days
past_100_days = data_training.tail(100)
```

```
final_df = pd.concat([past_100_days, data_testing], ignore_index=True)
```

```
input_data = scaler.fit_transform(final_df)
input_data
```

```
array([[0.12685382],
       [0.12562351],
       [0.11310665],
       ...,
       [0.99983953],
       [1.        ],
       [0.9989302 ]])
```

```
input_data.shape
```

```
(1228, 1)
```

```
x_test = []
y_test = []

for i in range(100,input_data.shape[0]):
    x_test.append(input_data[i-100:i])
    y_test.append(input_data[i,0])


x_test,y_test = np.array(x_test),np.array(y_test)
print(x_test.shape)
print(y_test.shape)
```

```
(1128, 100, 1)
(1128,)
```

```
#making predictions
y_predicted = model.predict(x_test)
```

```
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 95ms/step
```

```
y_predicted.shape
```

```
(1128, 1)
```

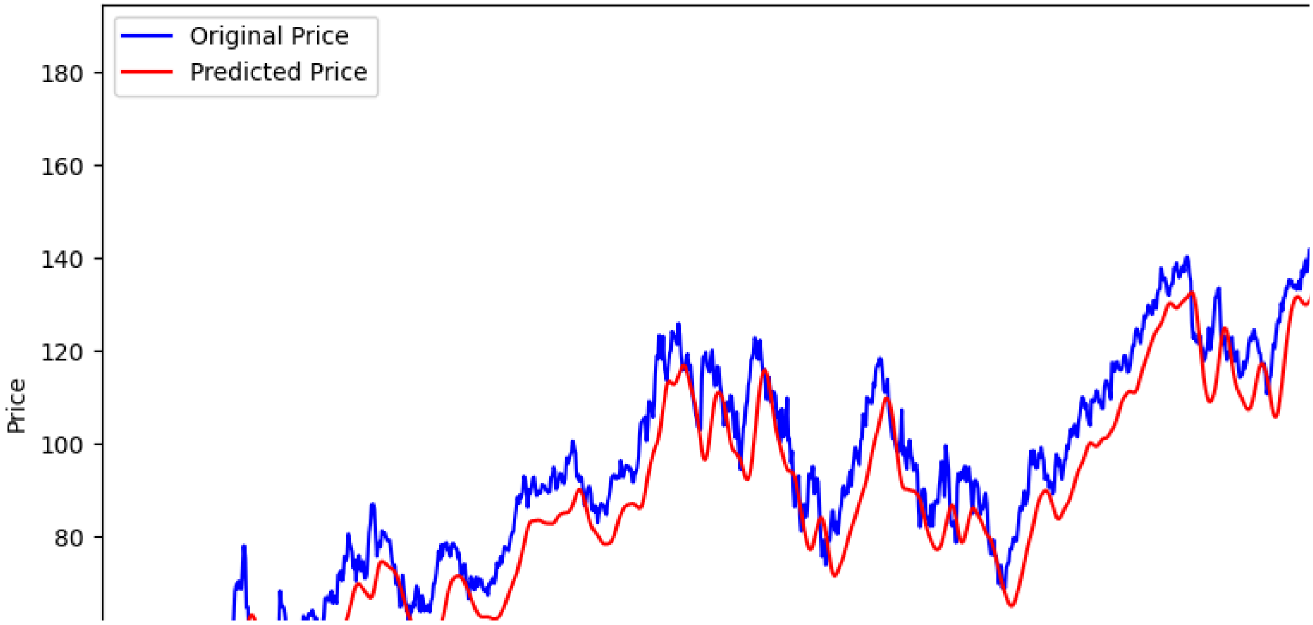
```
scaler.scale_
```

```
array([0.0053491])
```

```
scale_factor = 1/0.0053491
y_predicted = y_predicted*scale_factor
y_test = y_test*scale_factor
```

```
plt.figure(figsize=(12,6))
plt.plot(y_test,'b',label='Original Price')
plt.plot(y_predicted,'r',label='Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py'>
```


Original Price
Predicted Price
Price