Predicting Stock Prices: Develop a time series prediction model to forecast stock prices.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import linear_model as lm
from sklearn.metrics import r2_score,mean_squared_error
import pickle
```

```
df=pd.read_csv('/content/drive/MyDrive/stock_price_timeseries_dataset.csv')
df
```

	Date	Stock	Open	High	Low	Close	Volume
0	2020-01-01	AAPL	173.80	174.75	173.68	174.43	8204212
1	2020-01-02	AAPL	176.01	177.59	174.07	177.55	2766891
2	2020-01-03	AAPL	177.57	178.37	176.62	176.71	5721339
3	2020-01-06	AAPL	176.01	177.58	171.33	171.73	9242680
4	2020-01-07	AAPL	171.85	172.20	170.69	170.82	4416664
...	...	...	...	...	...	...	...
6520	2024-12-25	TSLA	225.09	225.21	221.66	223.18	9701165
6521	2024-12-26	TSLA	222.07	222.87	220.31	220.32	6045143
6522	2024-12-27	TSLA	219.94	221.69	218.38	219.57	8512677
6523	2024-12-30	TSLA	219.54	220.33	216.70	217.33	9137479
6524	2024-12-31	TSLA	216.47	219.61	216.19	218.41	2097124

6525 rows × 7 columns

Next steps: Generate code with df | New interactive sheet

```
df_stock=df[df['Stock']=='AAPL']
df_stock.head()
```

	Date	Stock	Open	High	Low	Close	Volume
0	2020-01-01	AAPL	173.80	174.75	173.68	174.43	8204212
1	2020-01-02	AAPL	176.01	177.59	174.07	177.55	2766891
2	2020-01-03	AAPL	177.57	178.37	176.62	176.71	5721339
3	2020-01-06	AAPL	176.01	177.58	171.33	171.73	9242680
4	2020-01-07	AAPL	171.85	172.20	170.69	170.82	4416664

Next steps: Generate code with df_stock | New interactive sheet

```
df_stock['Date']=pd.to_datetime(df_stock['Date'])
df_stock.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1305 entries, 0 to 1304
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1305 non-null   datetime64[ns]
 1   Stock   1305 non-null   object
 2   Open    1305 non-null   float64
 3   High    1305 non-null   float64
 4   Low     1305 non-null   float64
 5   Close   1305 non-null   float64
 6   Volume  1305 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 113.9+ KB
```

```
/tmp/ipython-input-31740583.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  df_stock['Date']=pd.to_datetime(df_stock['Date'])
```

```
df_stock['Days']=(df_stock['Date']-df_stock['Date'].min())
df_stock
```

```
/tmp/ipython-input-1482232864.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  df_stock['Days']=(df_stock['Date']-df_stock['Date'].min())
```

	Date	Stock	Open	High	Low	Close	Volume	Days
0	2020-01-01	AAPL	173.80	174.75	173.68	174.43	8204212	0 days
1	2020-01-02	AAPL	176.01	177.59	174.07	177.55	2766891	1 days
2	2020-01-03	AAPL	177.57	178.37	176.62	176.71	5721339	2 days
3	2020-01-06	AAPL	176.01	177.58	171.33	171.73	9242680	5 days
4	2020-01-07	AAPL	171.85	172.20	170.69	170.82	4416664	6 days
...	...	...	...	...	...	...	...	...
1300	2024-12-25	AAPL	207.96	210.35	206.18	208.42	4510463	1820 days
1301	2024-12-26	AAPL	208.69	214.63	207.23	213.99	4335705	1821 days
1302	2024-12-27	AAPL	213.99	217.12	212.77	216.24	2601207	1822 days
1303	2024-12-30	AAPL	216.62	217.17	213.72	215.53	7203690	1825 days
1304	2024-12-31	AAPL	215.26	218.50	215.06	217.00	5083372	1826 days

1305 rows × 8 columns

Next steps: Generate code with df_stock New interactive sheet

```
x=df_stock['Days']
y=df_stock['Close']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,shuffle=False)
```

```
model=lm.LinearRegression()
model.fit(x_train.dt.days.values.reshape(-1,1), y_train)
with open('model_pickle','wb') as f:
  pickle.dump(model,f)
```

```
y_pred=model.predict(x_test.dt.days.values.reshape(-1,1))
y_pred
```

```
       150.49573228, 150.47262862, 150.40331765, 150.380214  ,
       150.35711034, 150.33400669, 150.31090303, 150.24159206,
       150.21848841, 150.19538475, 150.17228109, 150.14917744,
       150.07986647, 150.05676281, 150.03365916, 150.0105555 ,
       149.98745185, 149.91814088, 149.89503722, 149.87193357,
       149.84882991, 149.82572625, 149.75641529, 149.73331163,
       149.71020797, 149.68710432, 149.66400066, 149.59468969,
       149.57158604, 149.54848238, 149.52537873, 149.50227507,
       149.4329641 , 149.40986045, 149.38675679, 149.36365313,
       149.34054948, 149.27123851, 149.24813485, 149.2250312 ,
       149.20192754, 149.17882389, 149.10951292, 149.08640926,
       149.06330561, 149.04020195, 149.01709829, 148.94778733,
       148.92468367, 148.90158001, 148.87847636, 148.8553727 ,
       148.78606173, 148.76295808, 148.73985442, 148.71675077,
       148.69364711, 148.62433614, 148.60123249, 148.57812883,
       148.55502517, 148.53192152, 148.46261055, 148.43950689,
       148.41640324, 148.39329958, 148.37019593, 148.30088496,
       148.2777813 , 148.25467765, 148.23157399, 148.20847033,
       148.13915937, 148.11605571, 148.09295205, 148.0698484 ,
       148.04674474, 147.97743377, 147.95433012, 147.93122646,
```

```
       146.84535463, 146.82225097, 146.79914732, 146.77604366,
       146.75294001, 146.68362904, 146.66052538, 146.63742173,
       146.61431807, 146.59121441, 146.52190345, 146.49879979,
       146.47569613, 146.45259248, 146.42948882, 146.36017785,
       146.3370742 , 146.31397054, 146.29086689, 146.26776323,
       146.19845226, 146.17534861, 146.15224495, 146.12914129,
       146.10603764, 146.03672667, 146.01362301, 145.99051936,
       145.9674157 , 145.94431205, 145.87500108, 145.85189742,
       145.82879377, 145.80569011, 145.78258645, 145.71327549,
       145.69017183, 145.66706817, 145.64396452, 145.62086086,
       145.55154989, 145.52844624, 145.50534258, 145.48223893,
       145.45913527, 145.3898243 , 145.36672065, 145.34361699,
       145.32051333, 145.29740968, 145.22809871, 145.20499505,
       145.1818914 , 145.15878774, 145.13568409, 145.06637312,
       145.04326946, 145.02016581, 144.99706215, 144.97395849,
       144.90464753, 144.88154387, 144.85844021, 144.83533656,
       144.8122329 , 144.74292193, 144.71981828, 144.69671462,
       144.67361097, 144.65050731, 144.58119634, 144.55809269,
       144.53498903, 144.51188537, 144.48878172, 144.41947075,
       144.39636709, 144.37326344, 144.35015978, 144.32705613,
       144.25774516, 144.2346415 , 144.21153785, 144.18843419,
       144.16533053, 144.09601957, 144.07291591, 144.04981225,
       144.0267086 , 144.00360494, 143.93429397, 143.91119032,
       143.88808666, 143.86498301, 143.84187935, 143.77256838,
       143.74946473, 143.72636107, 143.70325741, 143.68015376,
       143.61084279, 143.58773913, 143.56463548, 143.54153182,
       143.51842817, 143.4491172 , 143.42601354, 143.40290989,
       143.37980623, 143.35670257, 143.28739161, 143.26428795,
       143.24118429, 143.21808064, 143.19497698, 143.12566601,
       143.10256236])
```

```
mse=mean_squared_error(y_test,y_pred)
mse
```

```
3072.378496877529
```

```
r2=r2_score(y_test,y_pred)
r2
```

```
-11.213999810715315
```

```
plt.scatter(x_test.dt.days,y_test)
plt.xlabel('Time')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```