Stock Prediction:

In [ ]:

In [1]:
```python
#importing
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```python
df = pd.read_csv(r"C:\Users\HP\Downloads\stock_market.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

In [4]:
```python
df.tail()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
1004	2022-01-31	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	2022-02-01	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	2022-02-02	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	2022-02-03	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	2022-02-04	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

In [5]:
```python
df.shape
```

Out[5]:
```
(1009, 7)
```

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1009 non-null   object
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

In [7]:
```python
df.describe()
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1.009000e+03
mean	419.059673	425.320703	412.374044	419.000733	419.000733	7.570685e+06
std	108.537532	109.262960	107.555867	108.289999	108.289999	5.465535e+06
min	233.919998	250.649994	231.229996	233.880005	233.880005	1.144000e+06
25%	331.489990	336.299988	326.000000	331.619995	331.619995	4.091900e+06
50%	377.769989	383.010010	370.880005	378.670013	378.670013	5.934500e+06
75%	509.130005	515.630005	502.529999	509.079987	509.079987	9.322400e+06
max	692.349976	700.989990	686.090027	691.690002	691.690002	5.890430e+07

In [8]:
```python
df.corr()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_18840\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defau
lt to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
Open	1.000000	0.998605	0.998508	0.996812	0.996812	-0.415838
High	0.998605	1.000000	0.998203	0.998551	0.998551	-0.400699
Low	0.998508	0.998203	1.000000	0.998544	0.998544	-0.432116
Close	0.996812	0.998551	0.998544	1.000000	1.000000	-0.413362
Adj Close	0.996812	0.998551	0.998544	1.000000	1.000000	-0.413362
Volume	-0.415838	-0.400699	-0.432116	-0.413362	-0.413362	1.000000

In [9]:
```python
df.nunique()
```

Out[9]:
```
Date         1009
Open          976
High          983
Low           989
Close         988
Adj Close     988
Volume       1005
dtype: int64
```

In [10]:
```python
df.isnull().sum()
```

Out[10]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [11]:
```python
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```python
df.columns
```

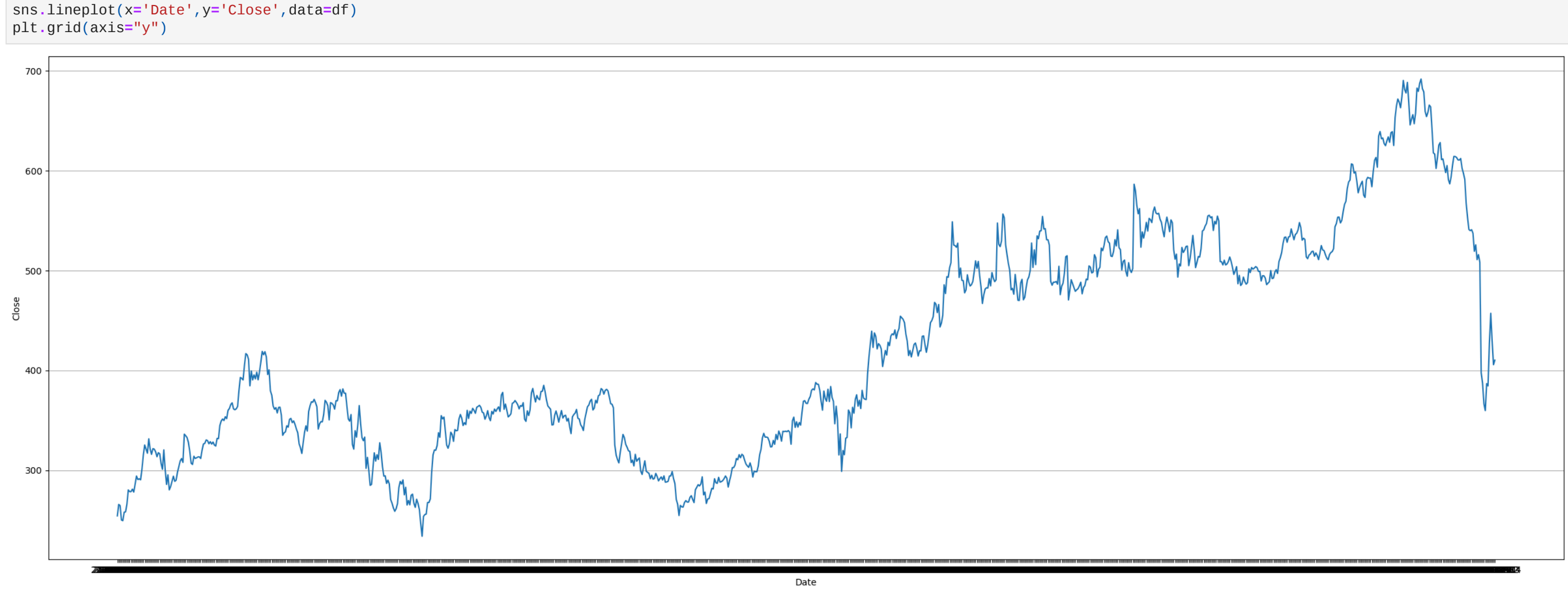
Out[12]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [ ]:

In [ ]:

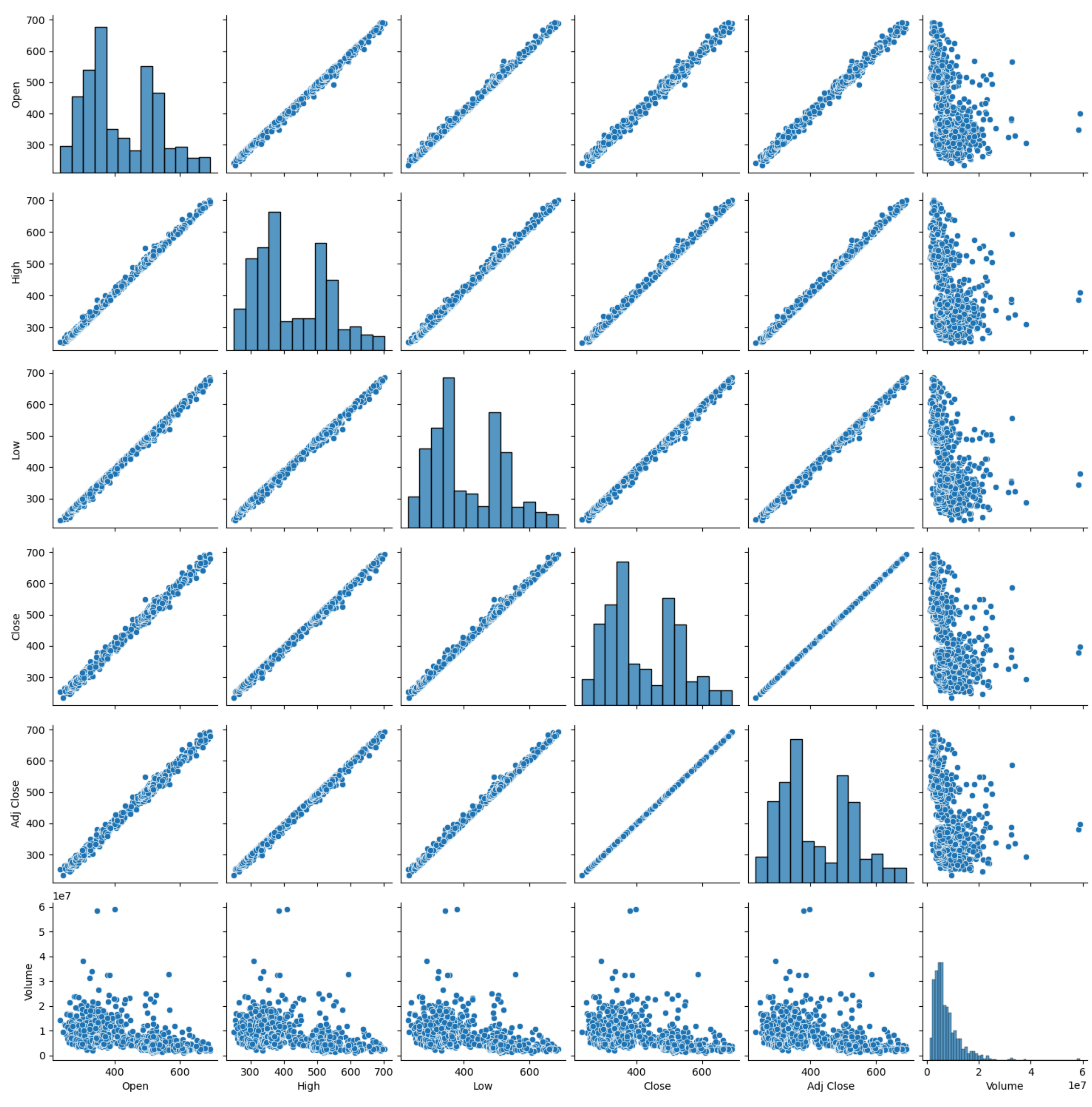
Data Visualization:

In [13]:
```python
plt.figure(figsize=(30,10))
sns.lineplot(x='Date',y='Close',data=df)
plt.grid(axis="y")
```



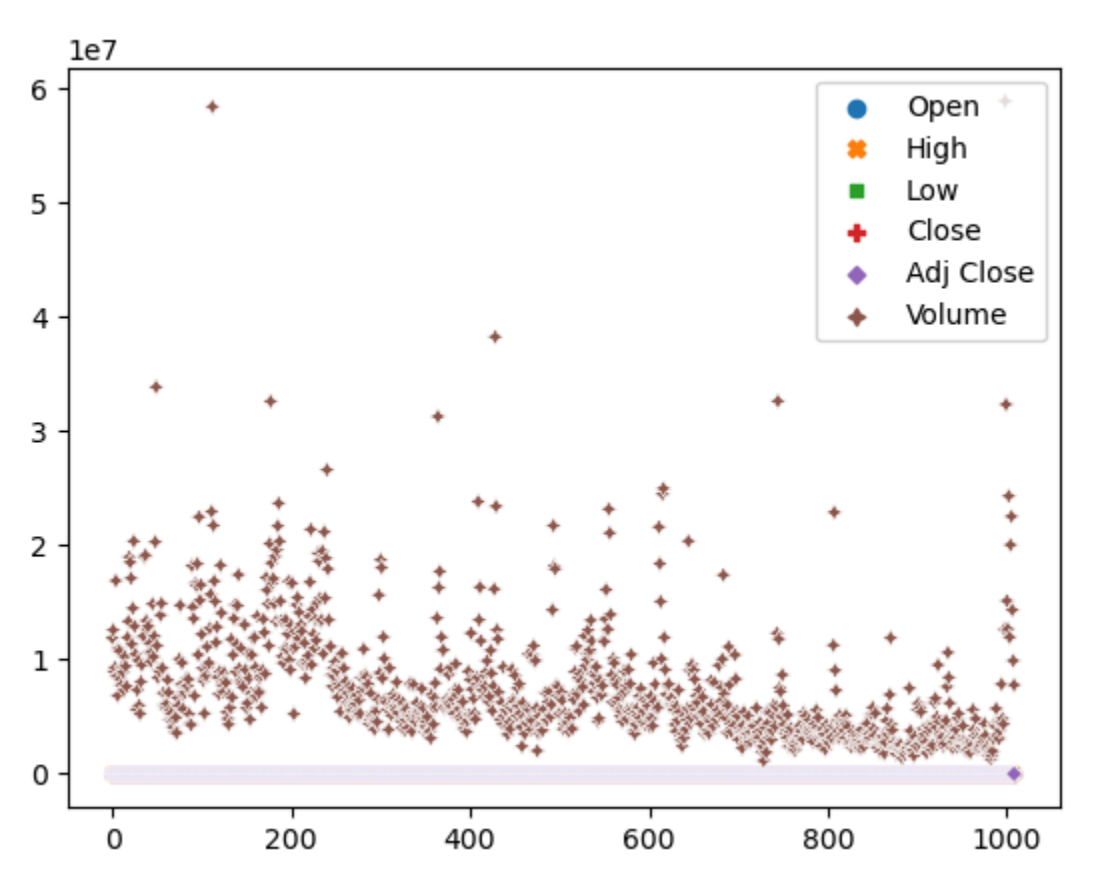
In [14]:
```python
sns.pairplot(df)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1220b546cd0>
```



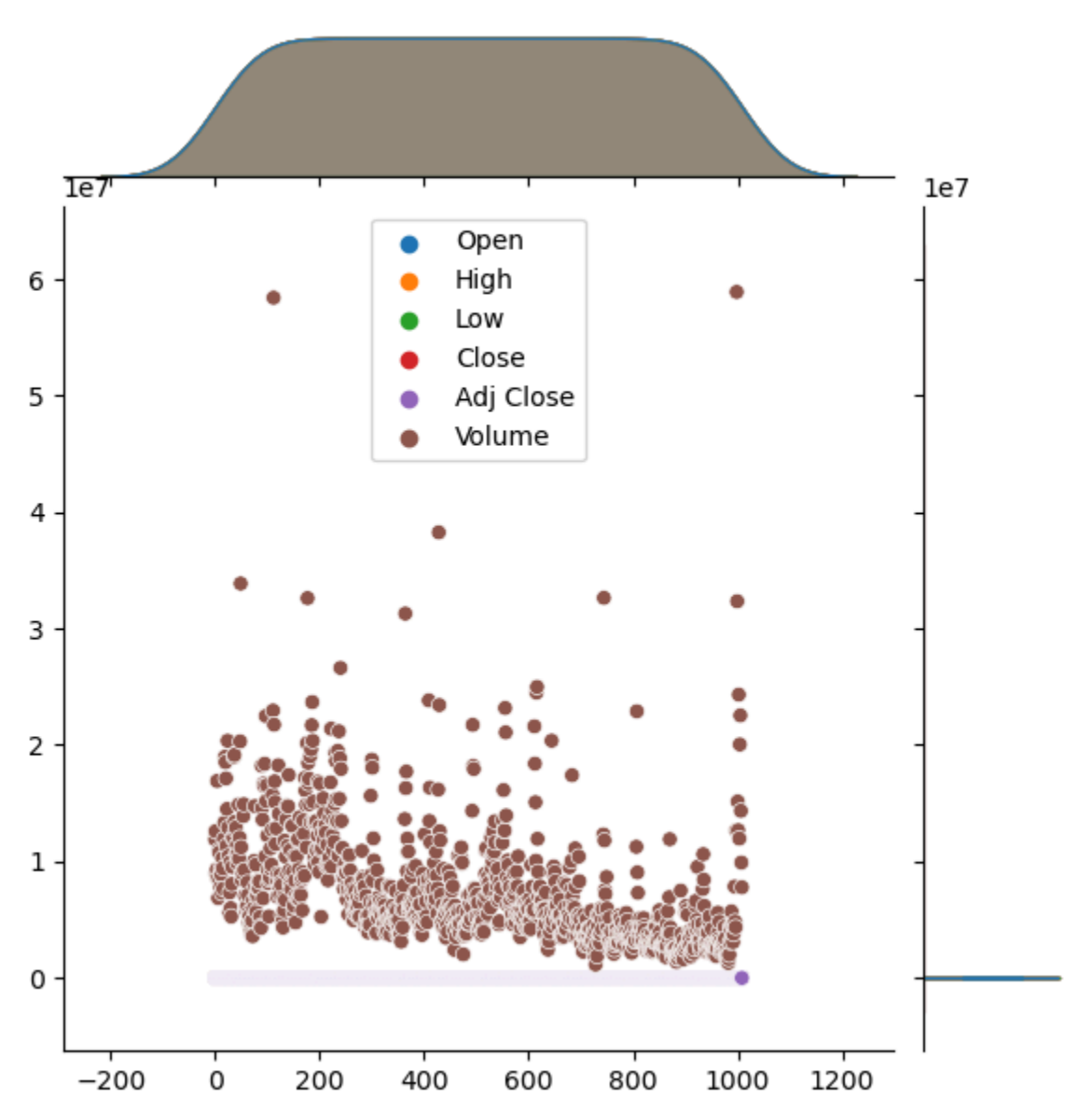
In [15]:
```python
sns.scatterplot(df)
```

Out[15]:
```
<Axes: >
```



In [16]:
```python
sns.jointplot(df)
```

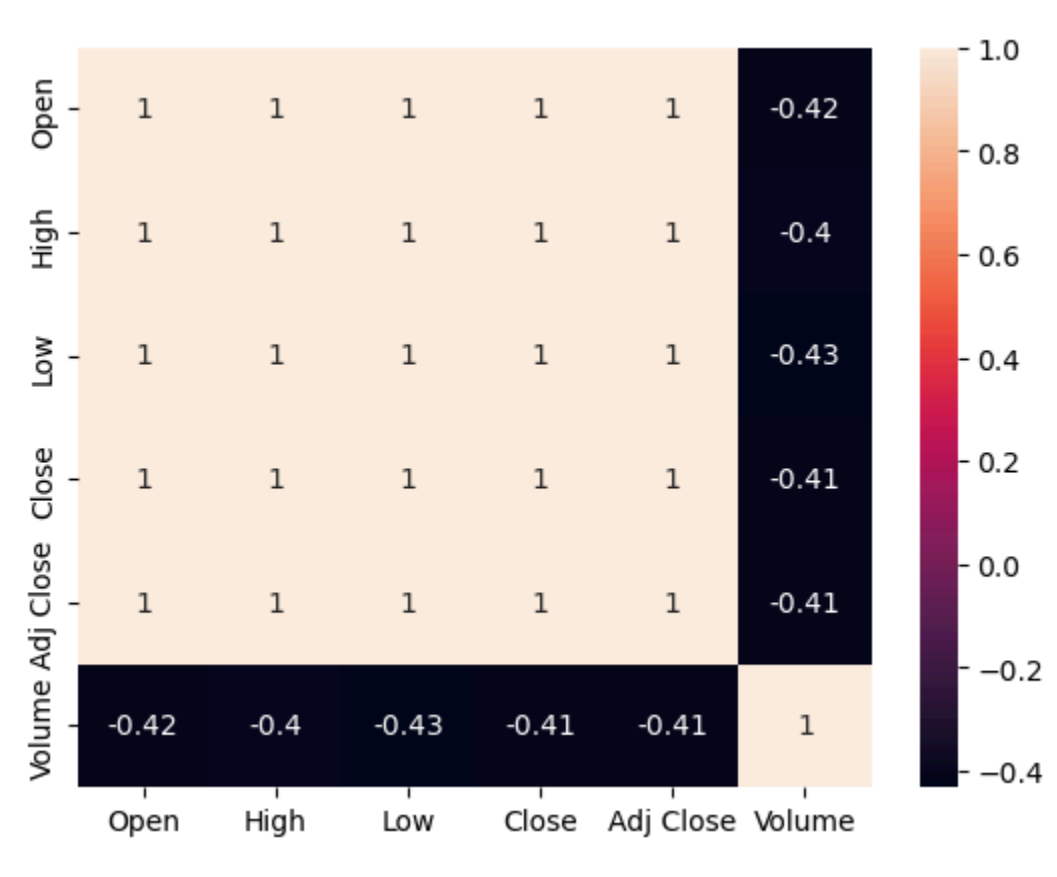
Out[16]:
```
<seaborn.axisgrid.JointGrid at 0x12209c737d0>
```



In [17]:
```python
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_18840\4277794465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defau
lt to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[17]:
```
<Axes: >
```



In [ ]:

In [ ]:

Data Modification:

In [18]:
```python
df
```

Out[18]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
...	...	...	...	...	...	...	...
1004	2022-01-31	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	2022-02-01	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	2022-02-02	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	2022-02-03	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	2022-02-04	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

1009 rows × 7 columns

In [19]:
```python
df.columns
```

Out[19]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [20]:
```python
x = df.drop(['Date','High', 'Low', 'Close', 'Adj Close', 'Volume'],axis=1)
x
```

Out[20]:

	Open
0	262.000000
1	247.699997
2	266.579987
3	267.079987
4	253.850006
...	...
1004	401.970001
1005	432.959991
1006	448.250000
1007	421.440002
1008	407.309998

1009 rows × 1 columns

In [21]:
```python
y = df['Close']
y
```

Out[21]:
```
0       254.259995
1       265.720001
2       264.559998
3       250.100006
4       249.470001
           ...
1004    427.140015
1005    457.130005
1006    429.480011
1007    405.600006
1008    410.170013
Name: Close, Length: 1009, dtype: float64
```

In [ ]:

In [ ]:

Machine Learning:

In [22]:
```python
from sklearn import linear_model
from sklearn.model_selection import train_test_split
```

In [23]:
```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [24]:
```python
x_train
```

Out[24]:

	Open
643	488.190002
294	365.109985
928	633.020020
131	339.890015
234	314.570007
...	...
185	305.260010
74	334.049988
1006	448.250000
907	598.570007
281	358.910004

706 rows × 1 columns

In [25]:
```python
x_test
```

Out[25]:

	Open
741	507.350006
624	492.250000
455	308.829987
866	553.969971
422	270.019989
...	...
280	366.399994
658	499.989990
889	515.469971
513	386.559998
909	578.169983

303 rows × 1 columns

In [26]:
```python
y_train
```

Out[26]:
```
643     490.579987
294     361.410004
928     624.940002
131     341.309998
234     324.660004
           ...
185     284.839996
74      331.619995
1006    429.480011
907     589.289978
281     375.220001
Name: Close, Length: 706, dtype: float64
```

In [27]:
```python
y_test
```

Out[27]:
```
741    500.859985
624    484.480011
455    315.549988
866    542.950012
422    267.529999
          ...
280    358.779999
658    483.859985
889    518.909973
513    386.000000
909    582.869995
Name: Close, Length: 303, dtype: float64
```

In [28]:
```python
print(df.shape)
print(x_train.shape)
print(x_test.shape)
```

```
(1009, 7)
(706, 1)
(303, 1)
```

In [29]:
```python
print(df.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1009, 7)
(706,)
(303,)
```

In [30]:
```python
model = linear_model.LinearRegression()
```

In [34]:
```python
model.fit(x_train,y_train)
```

Out[34]:
```
▾ LinearRegression
LinearRegression()
```

In [35]:
```python
model.score(x_test,y_test)
```

Out[35]:
```
0.9920768195057009
```

In [ ]:

Prediction:

In [36]:
```python
a = model.predict([[262.000]]) # input is Open and our predicion is close
print("Stock close:",a)
```

```
Stock close: [262.84197979]
C:\Users\HP\.conda\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [ ]:

In [ ]: