In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
```

In [ ]:
```
bitcoin = pd.read_csv('D:\python\data\coin_Bitcoin.csv')
ethereum = pd.read_csv('D:\python\data\coin_Ethereum.csv')
ethereum = ethereum.sort_values('Date')
Litecoin = pd.read_csv('D:\python\data\coin_Litecoin.csv')
Monero = pd.read_csv('D:\python\data\coin_Monero.csv')
Ripple = pd.read_csv('D:\python\data\coin_Ripple.csv')
Ripple = Ripple.sort_values('Date')
Solana = pd.read_csv('D:\python\data\coin_Solana.csv')
Stellar = pd.read_csv('D:\python\data\coin_Stellar.csv')
Tether = pd.read_csv('D:\python\data\coin_Tether.csv')
Tron = pd.read_csv('D:\python\data\coin_Tron.csv')
```

In [ ]:
```
bitcoin['Date'] = pd.to_datetime(bitcoin['Date'])
ethereum['Date'] = pd.to_datetime(ethereum['Date'])
Litecoin['Date'] = pd.to_datetime(Litecoin['Date'])
Monero['Date'] = pd.to_datetime(Monero['Date'])
Ripple['Date'] = pd.to_datetime(Ripple['Date'])
Solana['Date'] = pd.to_datetime(Solana['Date'])
Stellar['Date'] = pd.to_datetime(Stellar['Date'])
Tether['Date'] = pd.to_datetime(Tether['Date'])
Tron['Date'] = pd.to_datetime(Tron['Date'])
```

In [ ]:
```
bitcoin.head()
```

Out[ ]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketca
0	1	Bitcoin	BTC	2013-04-29 23:59:59	147.488007	134.000000	134.444000	144.539993	0.0	1.603769e+0
1	2	Bitcoin	BTC	2013-04-30 23:59:59	146.929993	134.050003	144.000000	139.000000	0.0	1.542813e+0
2	3	Bitcoin	BTC	2013-05-01 23:59:59	139.889999	107.720001	139.000000	116.989998	0.0	1.298955e+0
3	4	Bitcoin	BTC	2013-05-02 23:59:59	125.599998	92.281898	116.379997	105.209999	0.0	1.168517e+0
4	5	Bitcoin	BTC	2013-05-03 23:59:59	108.127998	79.099998	106.250000	97.750000	0.0	1.085995e+0

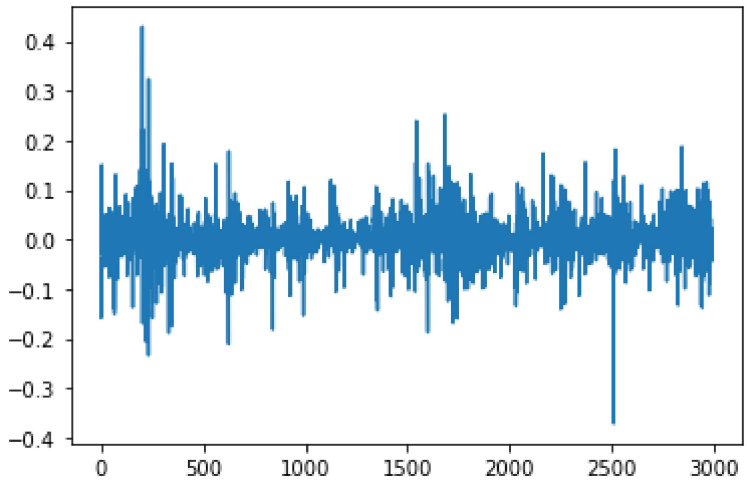
In [ ]:
```
bitcoin.describe()
```

Out[ ]:

	SNo	High	Low	Open	Close	Volume	Marketcap
count	2991.000000	2991.000000	2991.000000	2991.000000	2991.000000	2.991000e+03	2.991000e+0
mean	1496.000000	6893.326038	6486.009539	6700.146240	6711.290443	1.090633e+10	1.208761e+1
std	863.571653	11642.832456	10869.032130	11288.043736	11298.141921	1.888895e+10	2.109438e+1
min	1.000000	74.561096	65.526001	68.504997	68.431000	0.000000e+00	7.784112e+0
25%	748.500000	436.179001	422.879486	430.445496	430.569489	3.036725e+07	6.305579e+0
50%	1496.000000	2387.610107	2178.500000	2269.889893	2286.409912	9.460360e+08	3.741503e+1
75%	2243.500000	8733.926948	8289.800459	8569.656494	8576.238715	1.592015e+10	1.499957e+1
max	2991.000000	64863.098908	62208.964366	63523.754869	63503.457930	3.509679e+11	1.186364e+1

In [ ]:
```
bitcoin['return'] = bitcoin['Close'].pct_change()
bitcoin['return'].plot()
```

Out[ ]:
```
<AxesSubplot:>
```



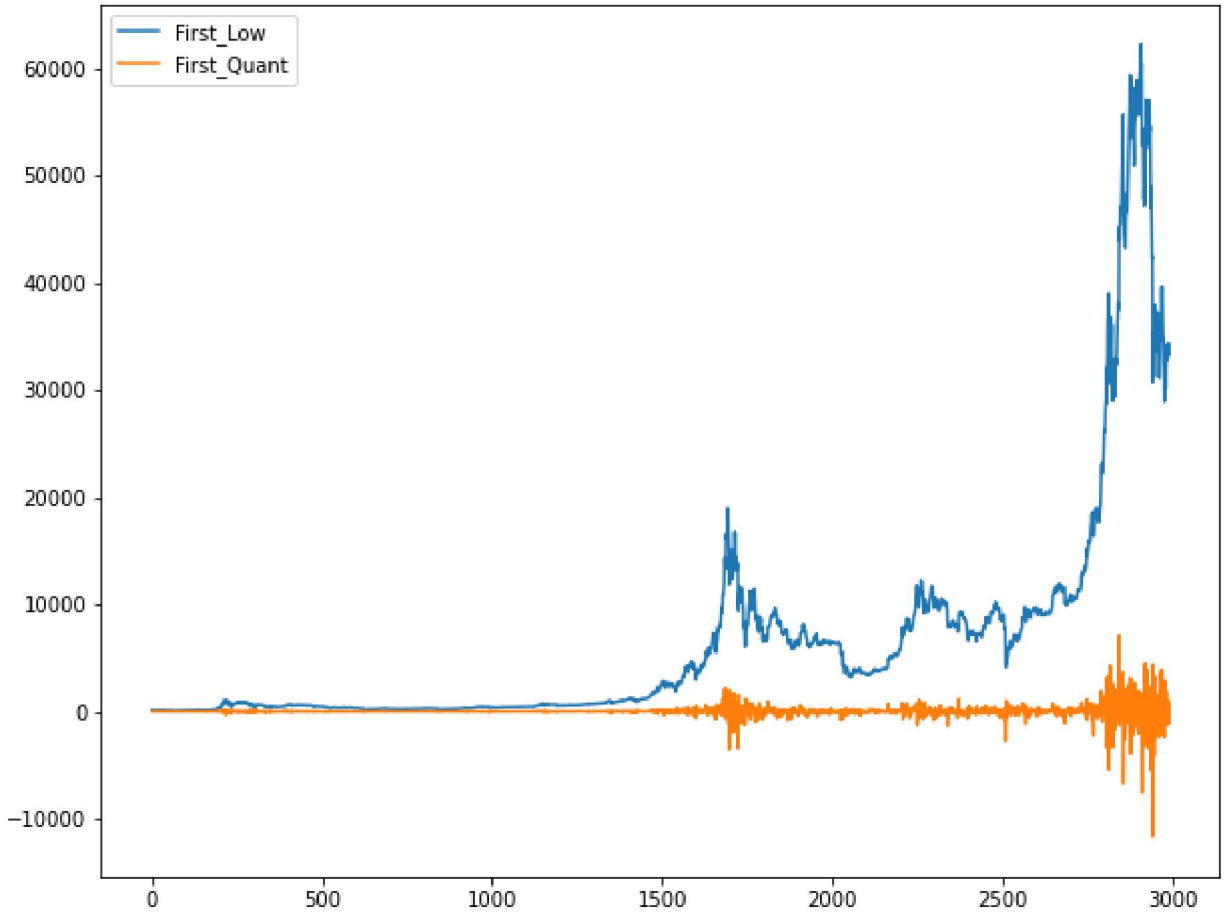
In [ ]:
```
(bitcoin['return']>0).value_counts()
```

Out[ ]:
```
True     1615
False    1376
Name: return, dtype: int64
```

In [ ]:
```
bitcoin['Quantum'] = bitcoin['Low'].diff()
a = bitcoin[['Low', 'Quantum']]
```

In [ ]:
```
ax = a.plot(figsize=(10,8))
ax.legend(['First_Low', 'First_Quant'])
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1fc00086940>
```



In [ ]:
```
bitcoin.head()
```

Out[ ]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketca
0	1	Bitcoin	BTC	2013-04-29 23:59:59	147.488007	134.000000	134.444000	144.539993	0.0	1.603769e+(
1	2	Bitcoin	BTC	2013-04-30 23:59:59	146.929993	134.050003	144.000000	139.000000	0.0	1.542813e+(
2	3	Bitcoin	BTC	2013-05-01 23:59:59	139.889999	107.720001	139.000000	116.989998	0.0	1.298955e+(
3	4	Bitcoin	BTC	2013-05-02 23:59:59	125.599998	92.281898	116.379997	105.209999	0.0	1.168517e+(
4	5	Bitcoin	BTC	2013-05-03 23:59:59	108.127998	79.099998	106.250000	97.750000	0.0	1.085995e+(

In [ ]:
```
Litecoin.head()
```

Out[ ]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
0	1	Litecoin	LTC	2013-04-29 23:59:59	4.57360	4.22564	4.36676	4.38390	0.0	7.538896e+07
1	2	Litecoin	LTC	2013-04-30 23:59:59	4.57238	4.16896	4.40352	4.29649	0.0	7.402092e+07
2	3	Litecoin	LTC	2013-05-01 23:59:59	4.35686	3.52029	4.28954	3.80101	0.0	6.560460e+07
3	4	Litecoin	LTC	2013-05-02 23:59:59	4.03930	3.00717	3.78002	3.37198	0.0	5.828798e+07
4	5	Litecoin	LTC	2013-05-03 23:59:59	3.45361	2.39594	3.39044	3.04491	0.0	5.269485e+07

In [ ]:
```
ethereum.head()
```

Out[ ]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	
146	147	Ethereum	ETH	2016-01-01 23:59:00	0.954822	0.931442	0.933712	0.948024	206062	7
512	513	Ethereum	ETH	2017-01-01 23:59:00	8.471230	7.982310	7.982310	8.172570	14731700	7
877	878	Ethereum	ETH	2018-01-01 23:59:00	782.530029	742.004028	755.757019	772.640991	2595760128	7
1242	1243	Ethereum	ETH	2019-01-01 23:59:00	141.397504	132.650705	133.418145	140.819413	2258709868	1
1607	1608	Ethereum	ETH	2020-01-01 23:59:00	132.835357	129.198285	129.630656	130.802008	7935230330	1

In [ ]:
```
Monero.head()
```

Out[ ]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
0	1	Monero	XMR	2014-05-22 23:59:59	2.18888	1.35909	1.58940	2.09558	132918.0	1.855608e+06
1	2	Monero	XMR	2014-05-23 23:59:59	3.43437	2.05104	2.05104	2.96302	266852.0	2.692180e+06
2	3	Monero	XMR	2014-05-24 23:59:59	4.01424	2.61778	2.92039	3.70245	248028.0	3.456658e+06
3	4	Monero	XMR	2014-05-25 23:59:59	4.04303	2.80330	4.04303	3.13856	283545.0	2.993339e+06

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
4	5	Monero	XMR	2014-05-26 23:59:59	3.76235	2.31202	3.21897	3.02186	220148.0	2.946133e+06

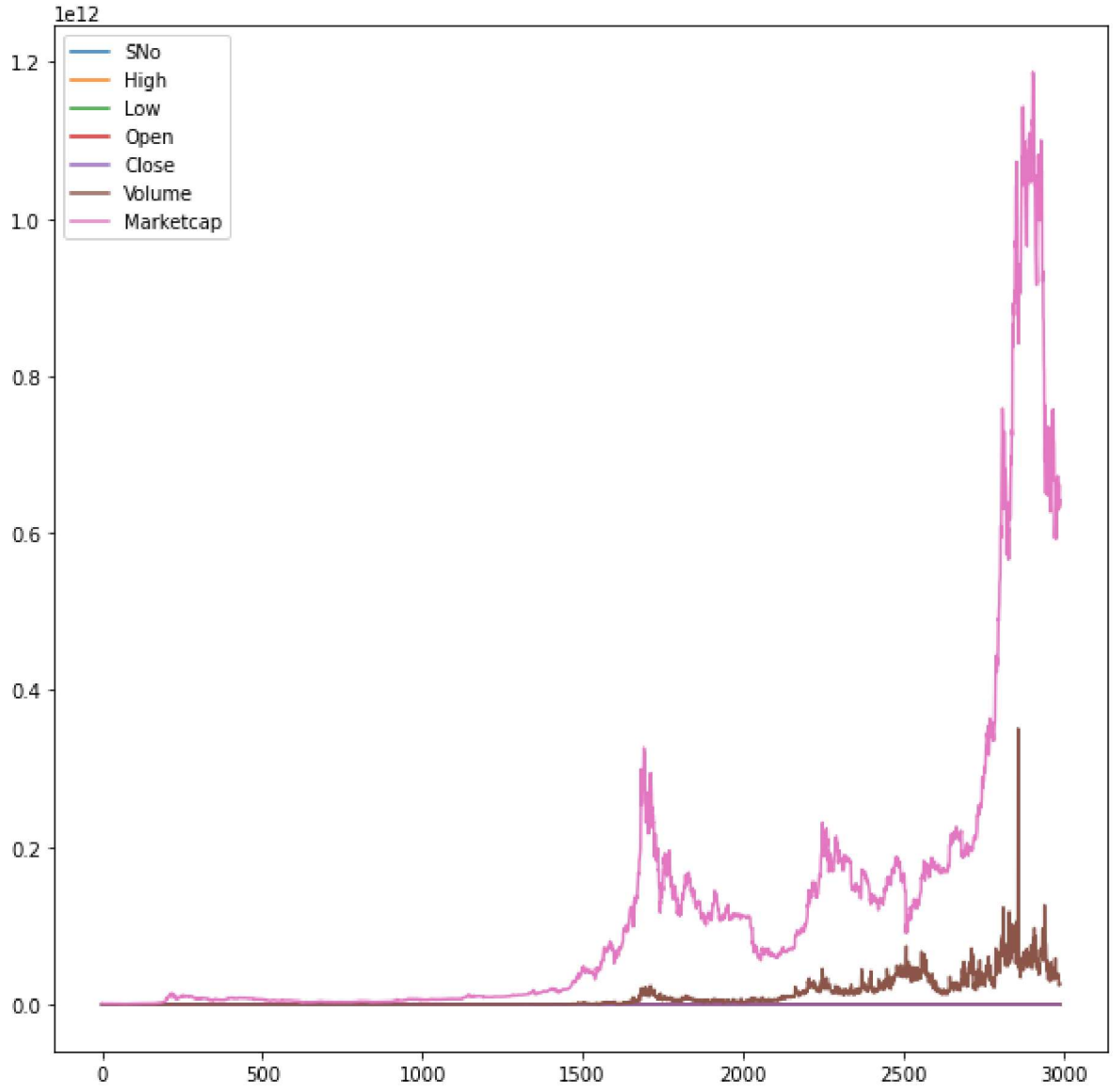
In [ ]:
```
ethereum['Date'] = pd.to_datetime(ethereum['Date'])
ethereum.head()
```

Out[ ]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	
146	147	Ethereum	ETH	2016-01-01 23:59:00	0.954822	0.931442	0.933712	0.948024	206062	7
512	513	Ethereum	ETH	2017-01-01 23:59:00	8.471230	7.982310	7.982310	8.172570	14731700	7
877	878	Ethereum	ETH	2018-01-01 23:59:00	782.530029	742.004028	755.757019	772.640991	2595760128	7
1242	1243	Ethereum	ETH	2019-01-01 23:59:00	141.397504	132.650705	133.418145	140.819413	2258709868	1
1607	1608	Ethereum	ETH	2020-01-01 23:59:00	132.835357	129.198285	129.630656	130.802008	7935230330	1

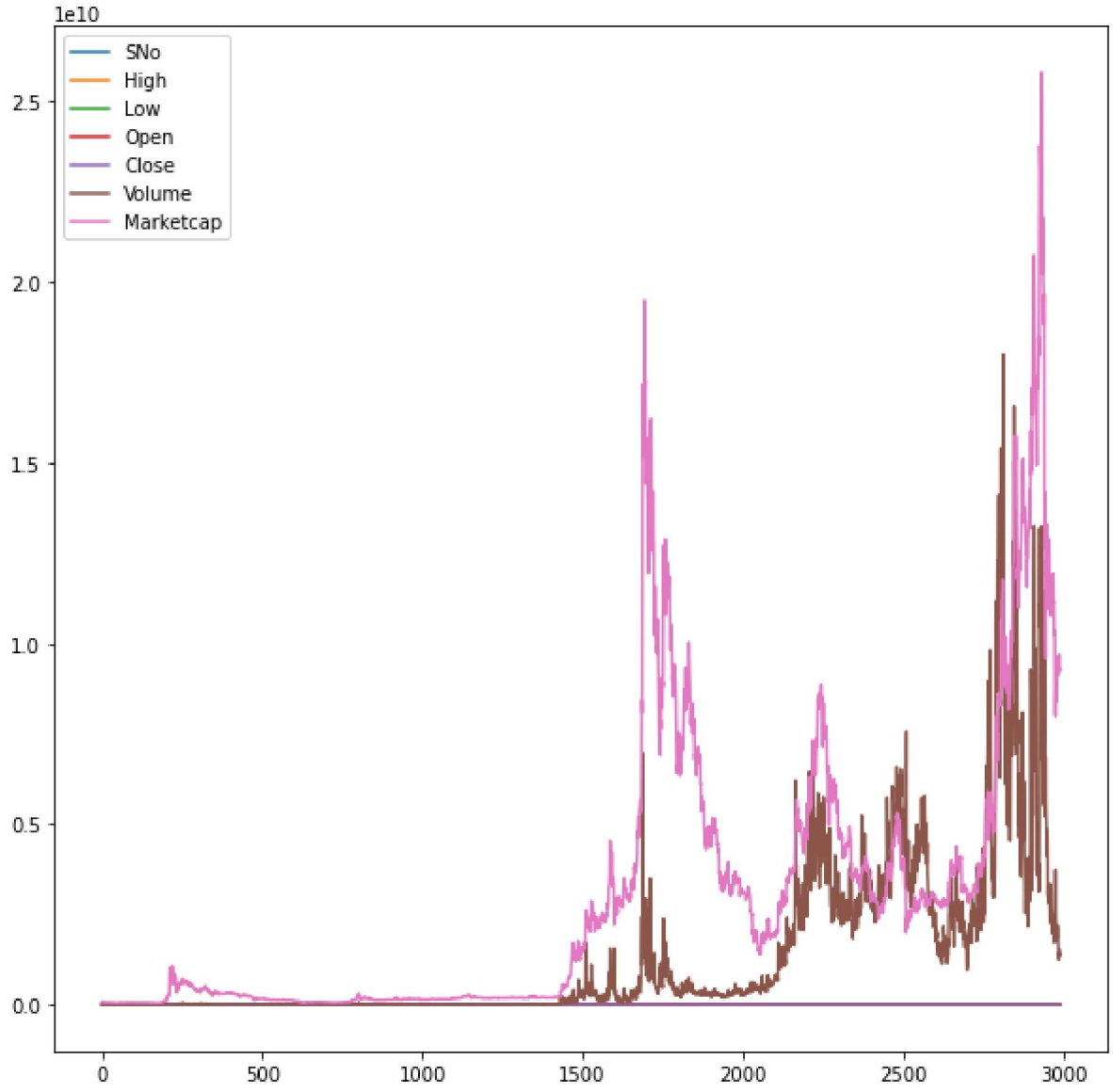
In [ ]:
```
bitcoin.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



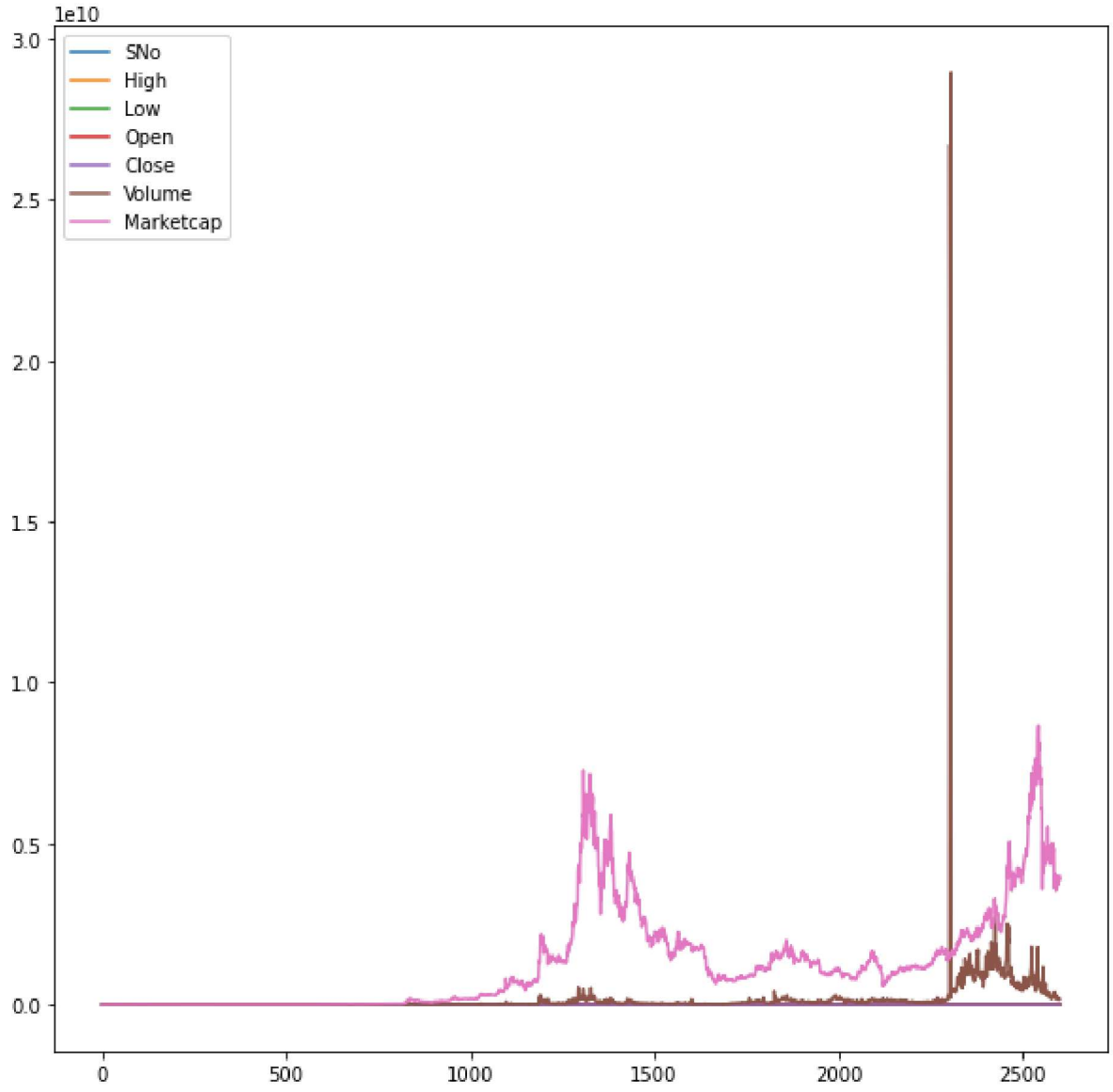
In [ ]:
```
Litecoin.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



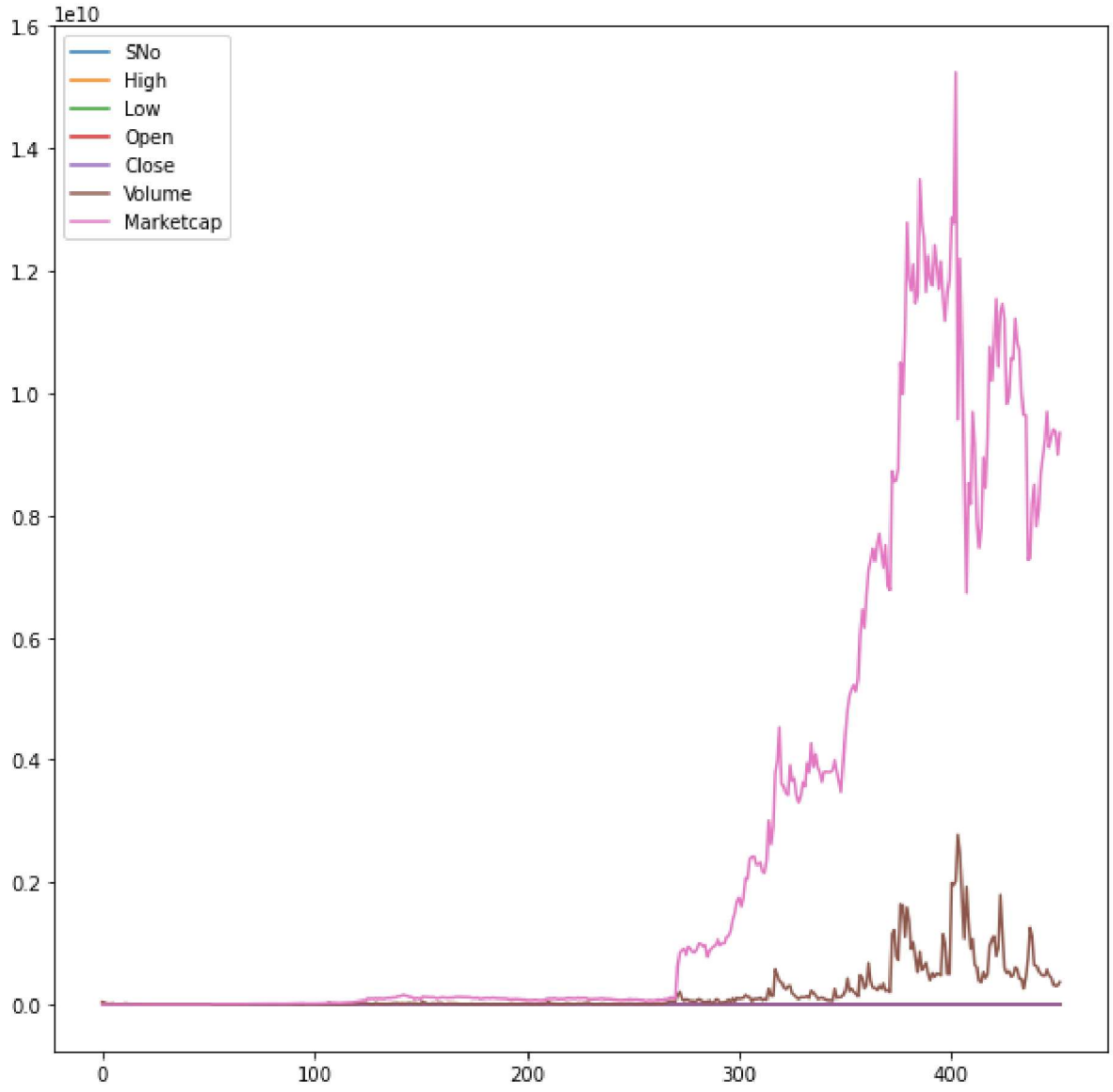
In [ ]:
```
Monero.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



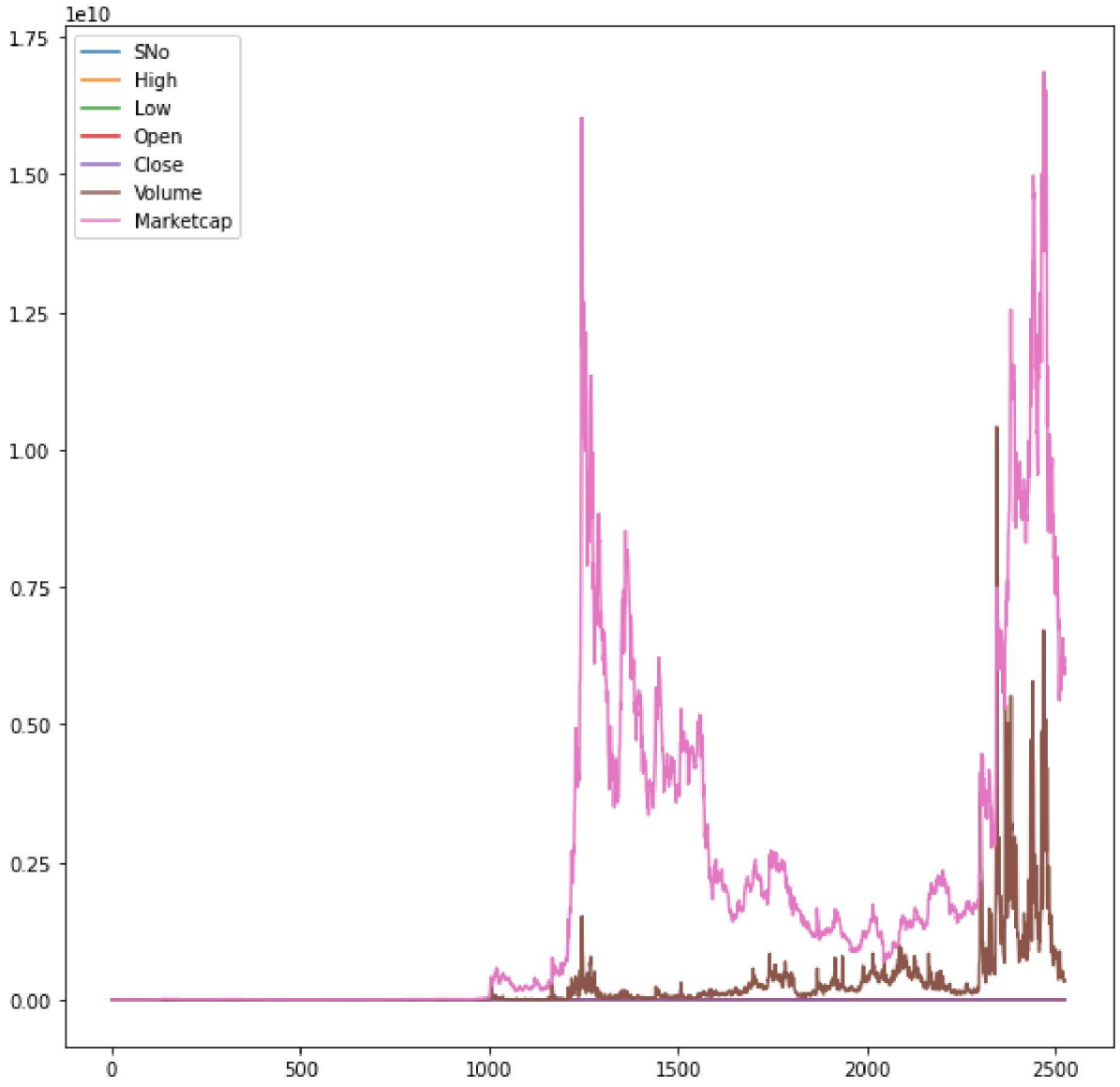
In [ ]:
```
Solana.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



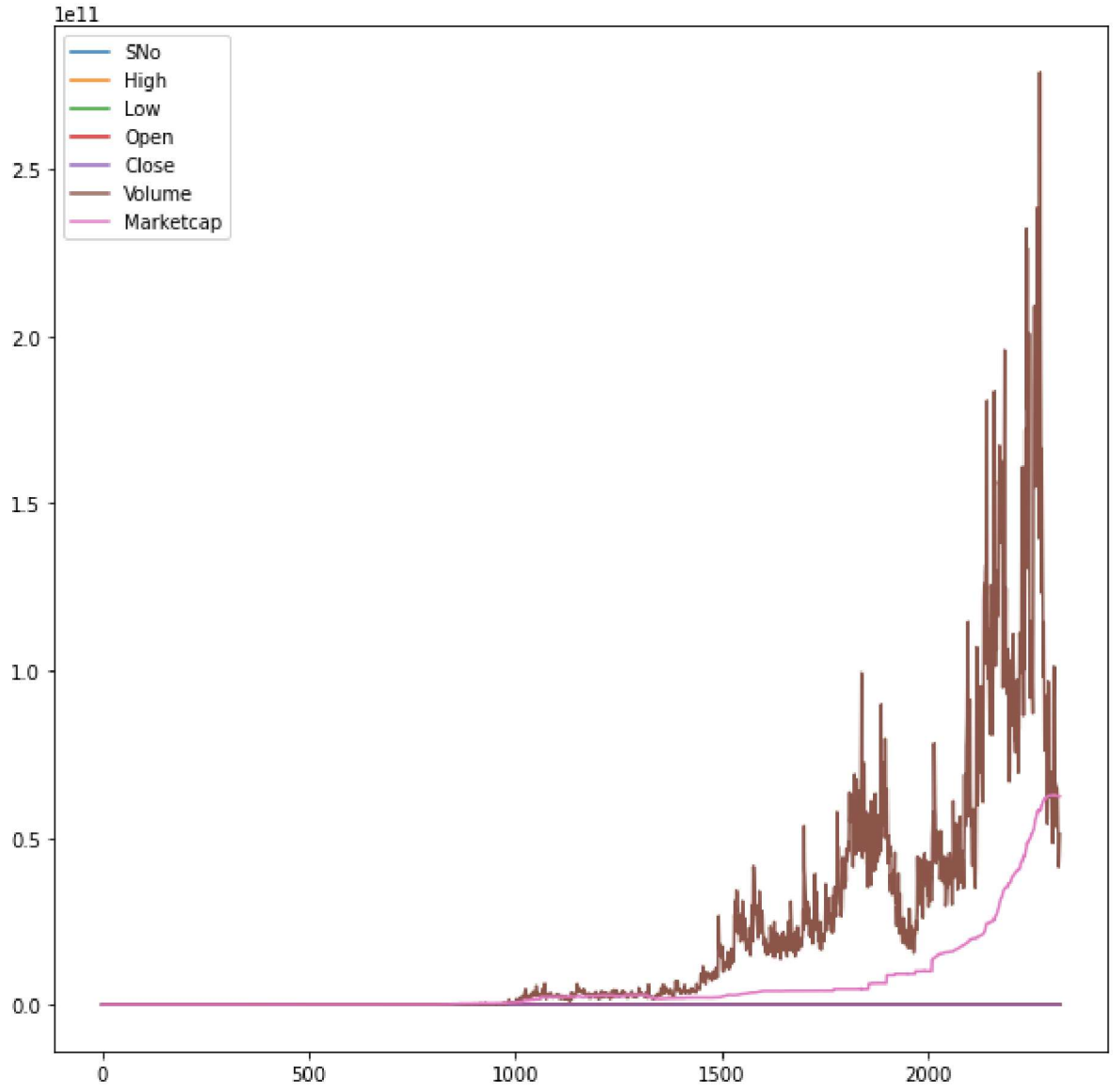
In [ ]:
```
Stellar.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



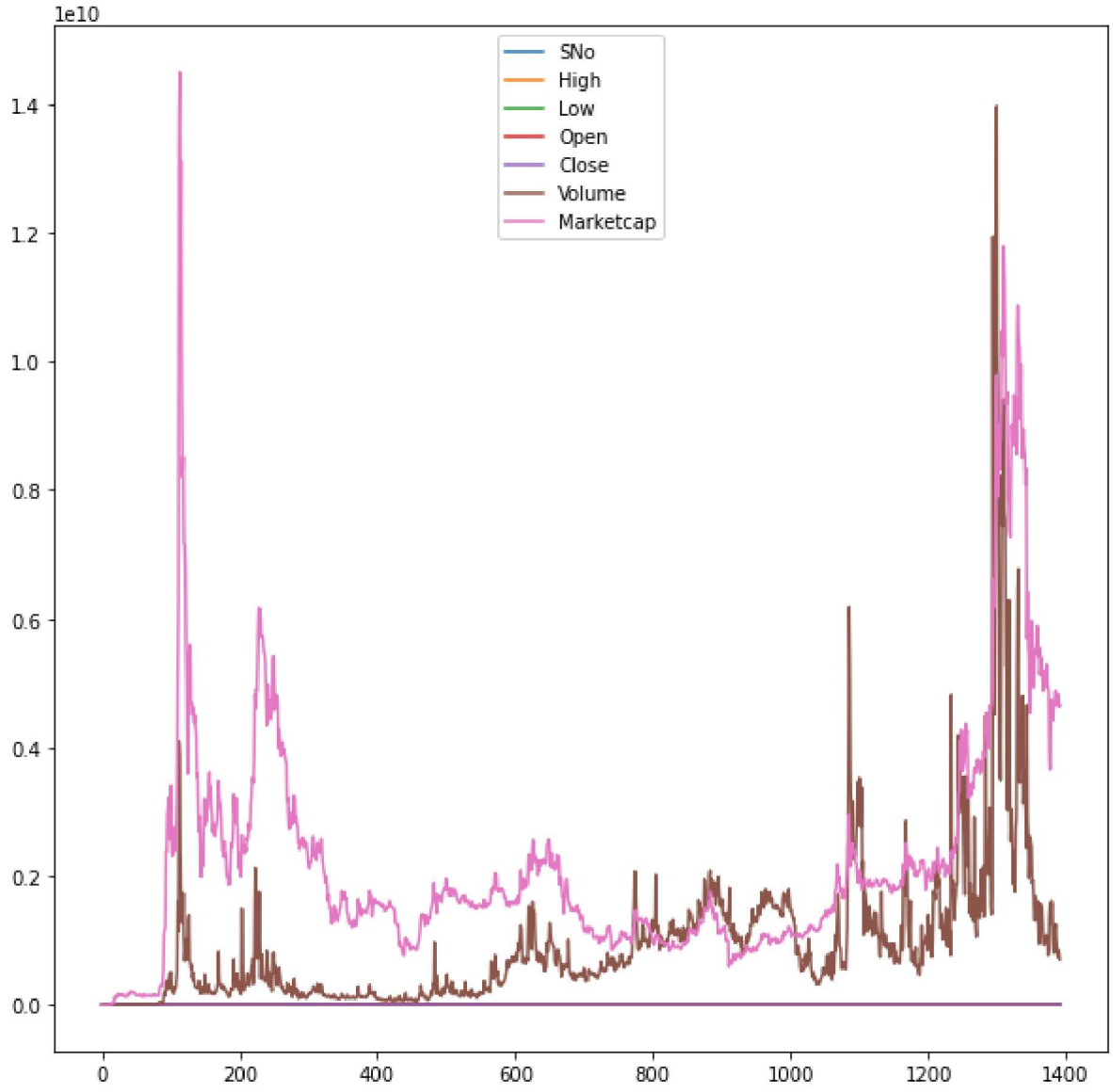
In [ ]:
```
Tether.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



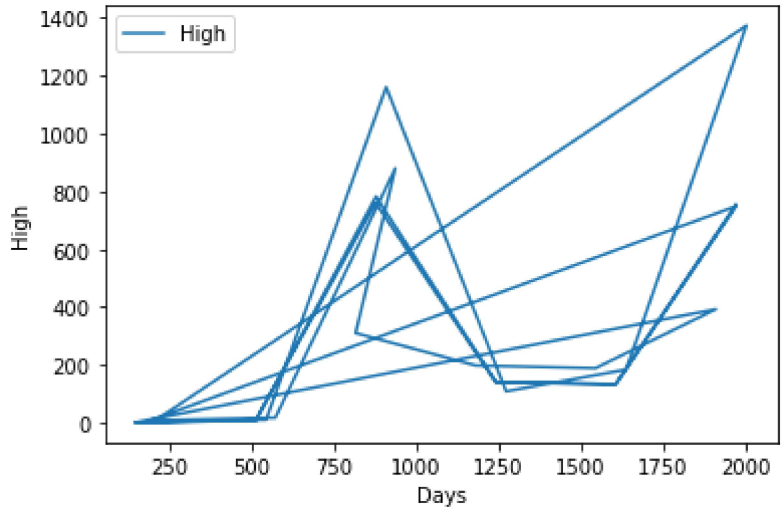
In [ ]:
```
Tron.plot(figsize=(10,10))
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
new = pd.concat([ethereum.head(15), ethereum.tail(10)]).plot(y='High')
plt.xlabel('Days')
plt.ylabel('High')
```

Out[ ]:
```
Text(0, 0.5, 'High')
```



In [ ]:
```
start_date = '2020-01-01'
end_date = '2020-01-31'
```

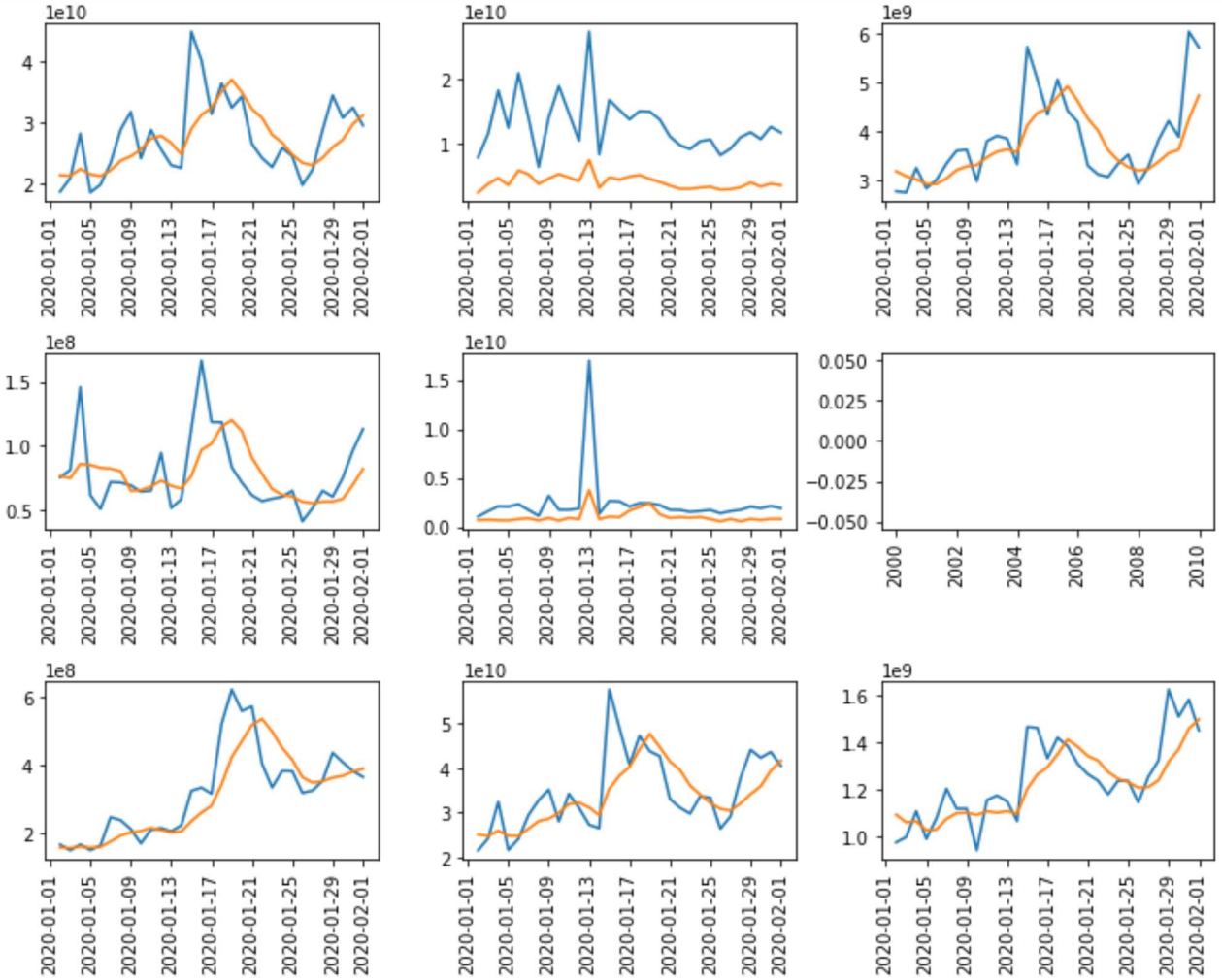
In [ ]:
```
bitcoin['Date'] = pd.to_datetime(bitcoin['Date'])
ethereum['Date'] = pd.to_datetime(ethereum['Date'])
Litecoin['Date'] = pd.to_datetime(Litecoin['Date'])
Monero['Date'] = pd.to_datetime(Monero['Date'])
Ripple['Date'] = pd.to_datetime(Ripple['Date'])
Solana['Date'] = pd.to_datetime(Solana['Date'])
Stellar['Date'] = pd.to_datetime(Stellar['Date'])
Tether['Date'] = pd.to_datetime(Tether['Date'])
Tron['Date'] = pd.to_datetime(Tron['Date'])
```

In [ ]:
```
bitcoin.set_index('Date', drop=True, inplace=True)
ethereum.set_index('Date', drop=True, inplace=True)
Litecoin.set_index('Date', drop=True, inplace=True)
Monero.set_index('Date', drop=True, inplace=True)
Ripple.set_index('Date', drop=True, inplace=True)
Solana.set_index('Date', drop=True, inplace=True)
Stellar.set_index('Date', drop=True, inplace=True)
Tether.set_index('Date', drop=True, inplace=True)
Tron.set_index('Date', drop=True, inplace=True)
```

In [ ]:
```
fig, ax = plt.subplots(nrows=3 , ncols=3 , figsize=(10,8))

bitcoin['Bit_moving'] = bitcoin['Volume'].rolling(window=5).mean()
a = bitcoin[start_date:end_date][['Volume', 'Bit_moving']]
ax[0][0].plot(a)

ethereum['Eth_moving'] = ethereum['Volume'].rolling(window=5).mean()
b = ethereum[start_date:end_date][['Volume', 'Eth_moving']]
ax[0][1].plot(b)

Litecoin['lite_moving'] = Litecoin['Volume'].rolling(window=5).mean()
c = Litecoin[start_date:end_date][['Volume', 'lite_moving']]
ax[0][2].plot(c)

Monero['Mon_moving'] = Monero['Volume'].rolling(window=5).mean()
d = Monero[start_date:end_date][['Volume', 'Mon_moving']]
ax[1][0].plot(d)

Ripple['Rpl_moving'] = Ripple['Volume'].rolling(window=5).mean()
e = Ripple[start_date:end_date][['Volume', 'Rpl_moving']]
ax[1][1].plot(e)

Solana['Sol'] = Solana['Volume'].rolling(window=5).mean()
f = Solana[start_date:end_date][['Volume', 'Sol']]
ax[1][2].plot(f)

Stellar['Stl'] = Stellar['Volume'].rolling(window=5).mean()
g = Stellar[start_date:end_date][['Volume', 'Stl']]
ax[2][0].plot(g)

Tether['tehr'] = Tether['Volume'].rolling(window=5).mean()
h = Tether[start_date:end_date][['Volume', 'tehr']]
```

```
ax[2][1].plot(h)

Tron['Tron'] = Tron['Volume'].rolling(window=5).mean()
i = Tron[start_date:end_date][['Volume', 'Tron']]
ax[2][2].plot(i)

for ax in fig.axes:
    plt.sca(ax)
    plt.xticks(rotation=90)

fig.tight_layout()
```

```
C:\Users\Goku\AppData\Local\Temp/ipykernel_18372/109765575.py:8: FutureWarning:

Value based partial slicing on non-monotonic DatetimeIndexes with non-existing keys is d
eprecated and will raise a KeyError in a future Version.

C:\Users\Goku\AppData\Local\Temp/ipykernel_18372/109765575.py:20: FutureWarning:

Value based partial slicing on non-monotonic DatetimeIndexes with non-existing keys is d
eprecated and will raise a KeyError in a future Version.
```

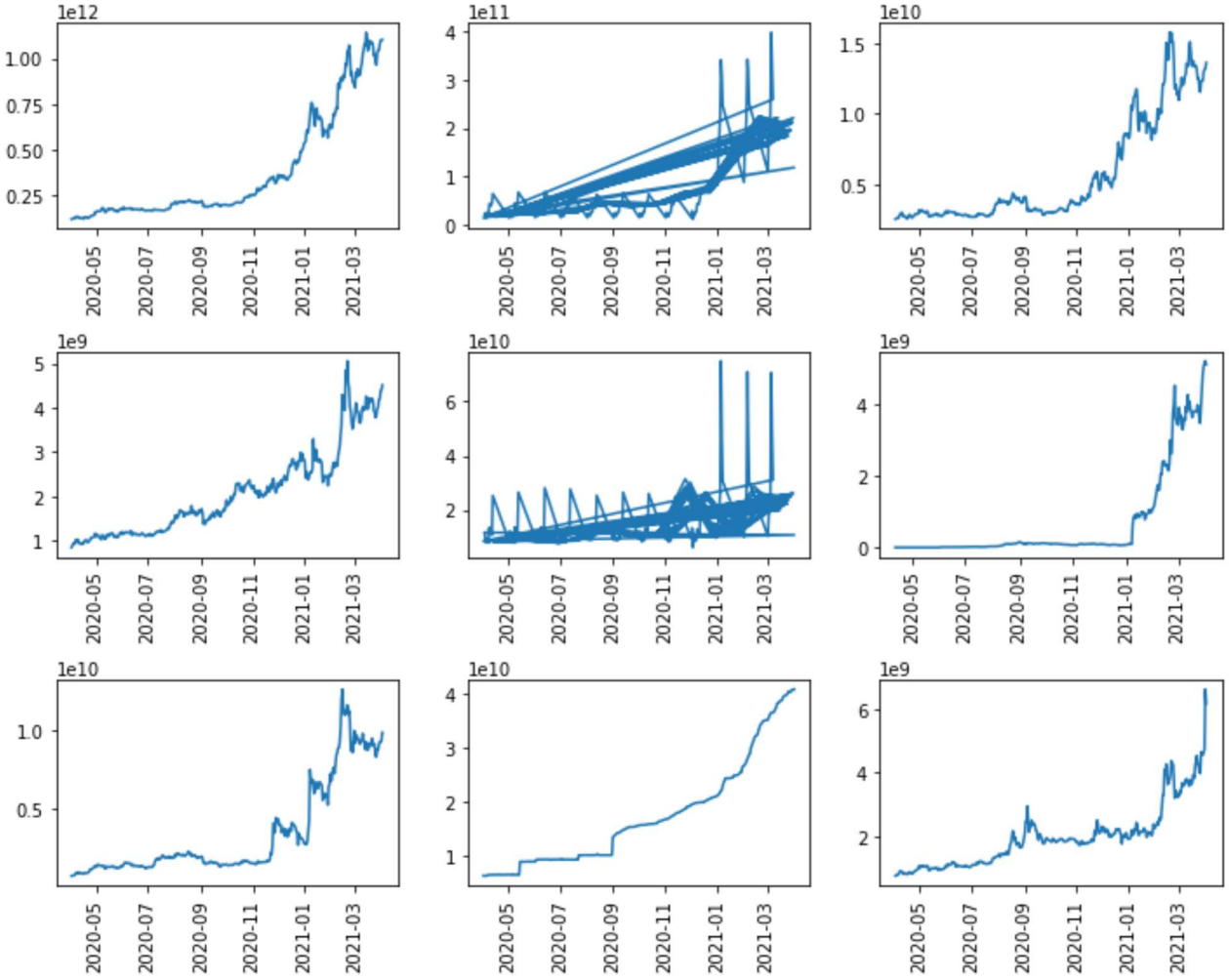


In [ ]:

```
start_date = '2020-04-01'
end_date = '2021-04-01'
```

In [ ]:
```
fig, ax = plt.subplots(nrows=3 , ncols=3 , figsize=(10,8))

a = bitcoin[start_date:end_date]['Marketcap']
ax[0][0].plot(a)

b = ethereum[start_date:end_date]['Marketcap']
ax[0][1].plot(b)

c = Litecoin[start_date:end_date]['Marketcap']
ax[0][2].plot(c)

d = Monero[start_date:end_date]['Marketcap']
ax[1][0].plot(d)

e = Ripple[start_date:end_date]['Marketcap']
ax[1][1].plot(e)

f = Solana[start_date:end_date]['Marketcap']
ax[1][2].plot(f)

g = Stellar[start_date:end_date]['Marketcap']
ax[2][0].plot(g)

h = Tether[start_date:end_date]['Marketcap']
ax[2][1].plot(h)

i = Tron[start_date:end_date]['Marketcap']
ax[2][2].plot(i)

for ax in fig.axes:
    plt.sca(ax)
    plt.xticks(rotation=90)

fig.tight_layout()
```

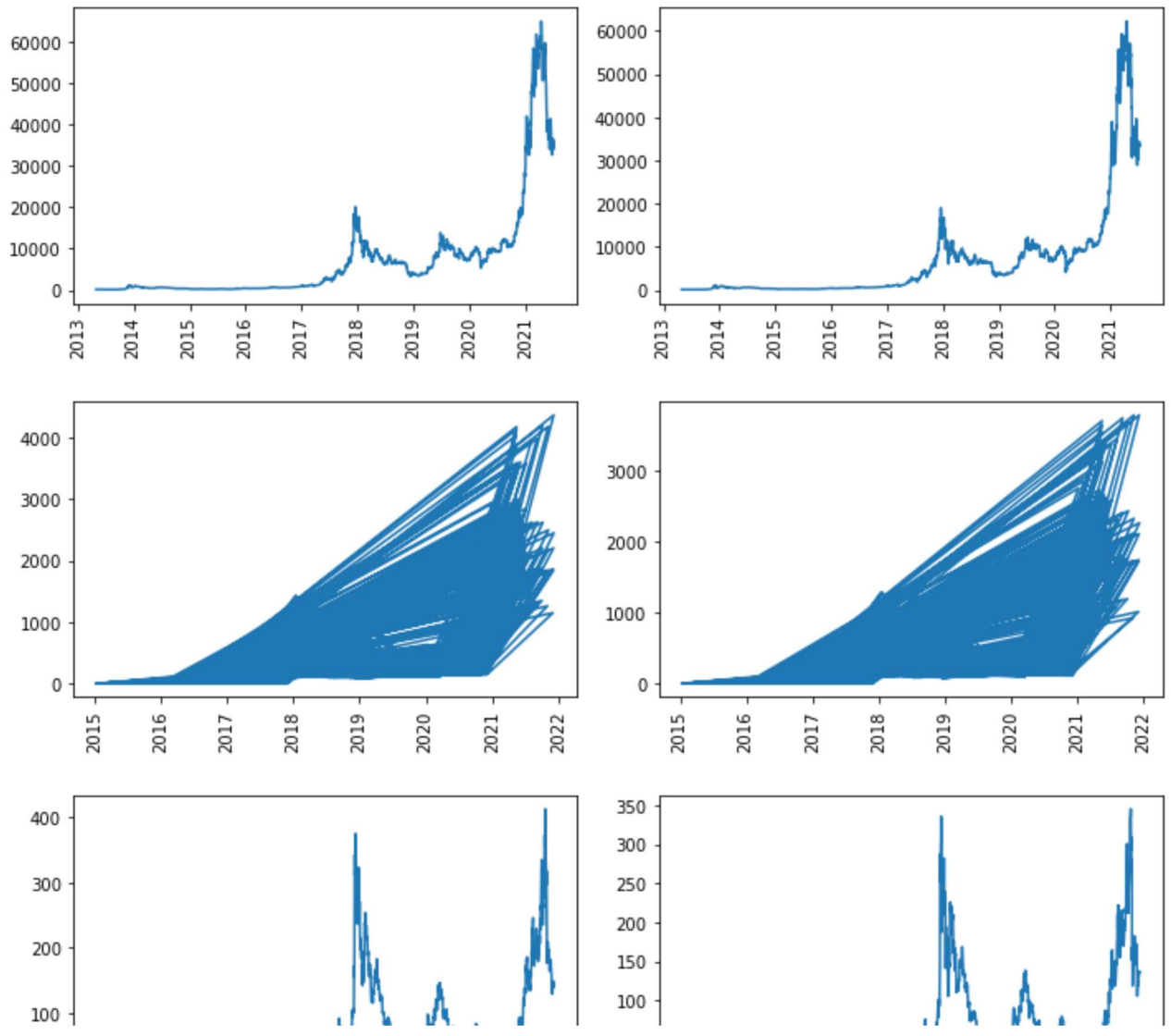
```
C:\Users\Goku\AppData\Local\Temp/ipykernel_18372/3498591309.py:6: FutureWarning:

Value based partial slicing on non-monotonic DatetimeIndexes with non-existing keys is d
eprecated and will raise a KeyError in a future Version.

C:\Users\Goku\AppData\Local\Temp/ipykernel_18372/3498591309.py:15: FutureWarning:

Value based partial slicing on non-monotonic DatetimeIndexes with non-existing keys is d
eprecated and will raise a KeyError in a future Version.
```

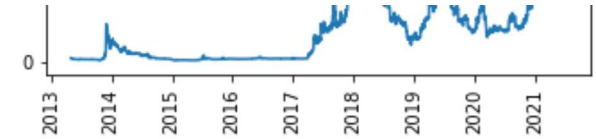


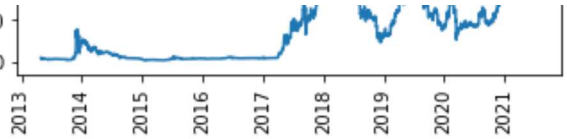
In [ ]:

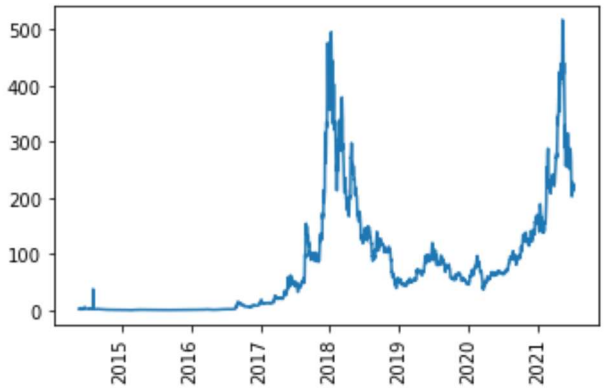
```
fig, ax = plt.subplots(nrows=9 , ncols=2 , figsize=(10,30))

a = bitcoin['High']
ax[0][0].plot(a)
a = bitcoin['Low']
ax[0][1].plot(a)

b = ethereum['High']
ax[1][0].plot(b)
b = ethereum['Low']
ax[1][1].plot(b)

c = Litecoin['High']
ax[2][0].plot(c)
c = Litecoin['Low']
ax[2][1].plot(c)

d = Monero['High']
ax[3][0].plot(d)
d = Monero['Low']
ax[3][1].plot(d)

e = Ripple['High']
ax[4][0].plot(e)
e = Ripple['Low']
ax[4][1].plot(e)
```

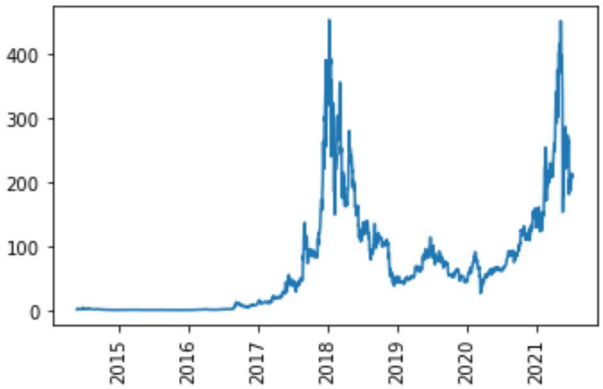
```
f = Solana['High']
ax[5][0].plot(f)
f = Solana['Low']
ax[5][1].plot(f)

g = Stellar['High']
ax[6][0].plot(g)
g = Stellar['Low']
ax[6][1].plot(g)

h = Tether['High']
ax[7][0].plot(h)
h = Tether['Low']
ax[7][1].plot(h)

i = Tron['High']
ax[8][0].plot(i)
i = Tron['Low']
ax[8][1].plot(i)

for ax in fig.axes:
    plt.sca(ax)
    plt.xticks(rotation=90)

fig.tight_layout()
```

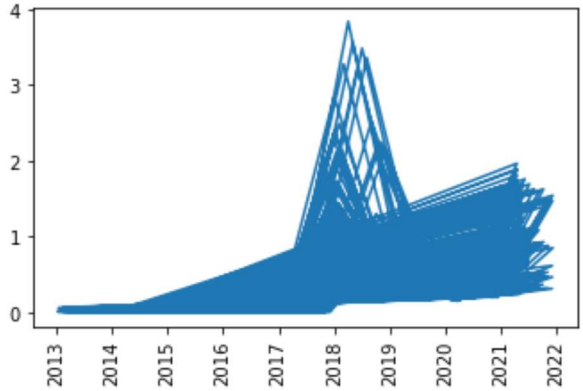


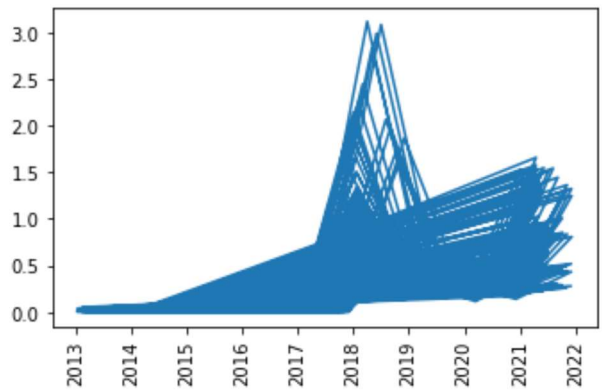

0
2013
2014
2015
2016
2017
2018
2019
2020
2021

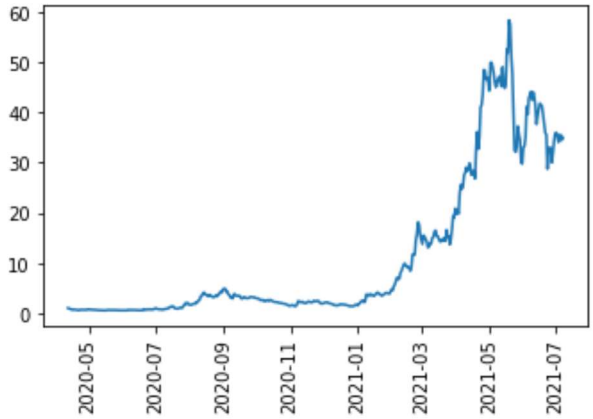

50
0
2013
2014
2015
2016
2017
2018
2019
2020
2021

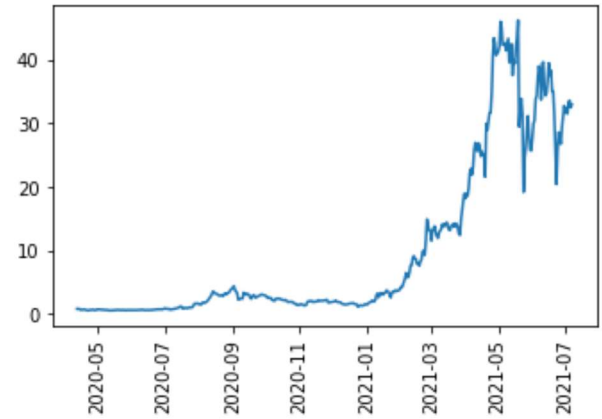

500
400
300
200
100
0
2015
2016
2017
2018
2019
2020
2021

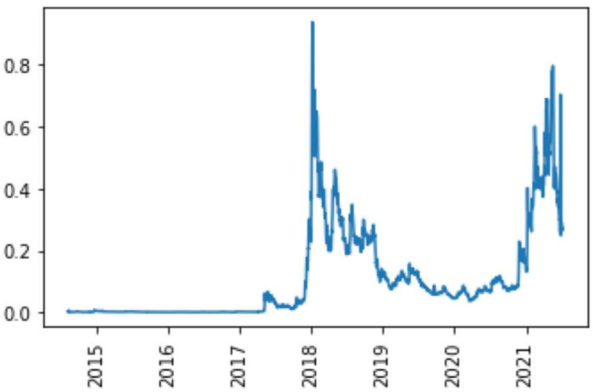

400
300
200
100
0
2015
2016
2017
2018
2019
2020
2021

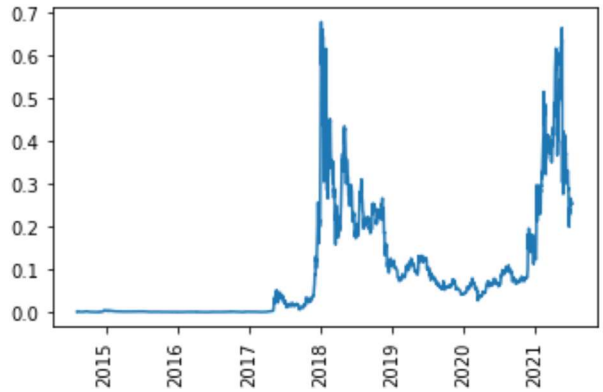

4
3
2
1
0
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022


3.0
2.5
2.0
1.5
1.0
0.5
0.0
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022


60
50
40
30
20
10
0
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07

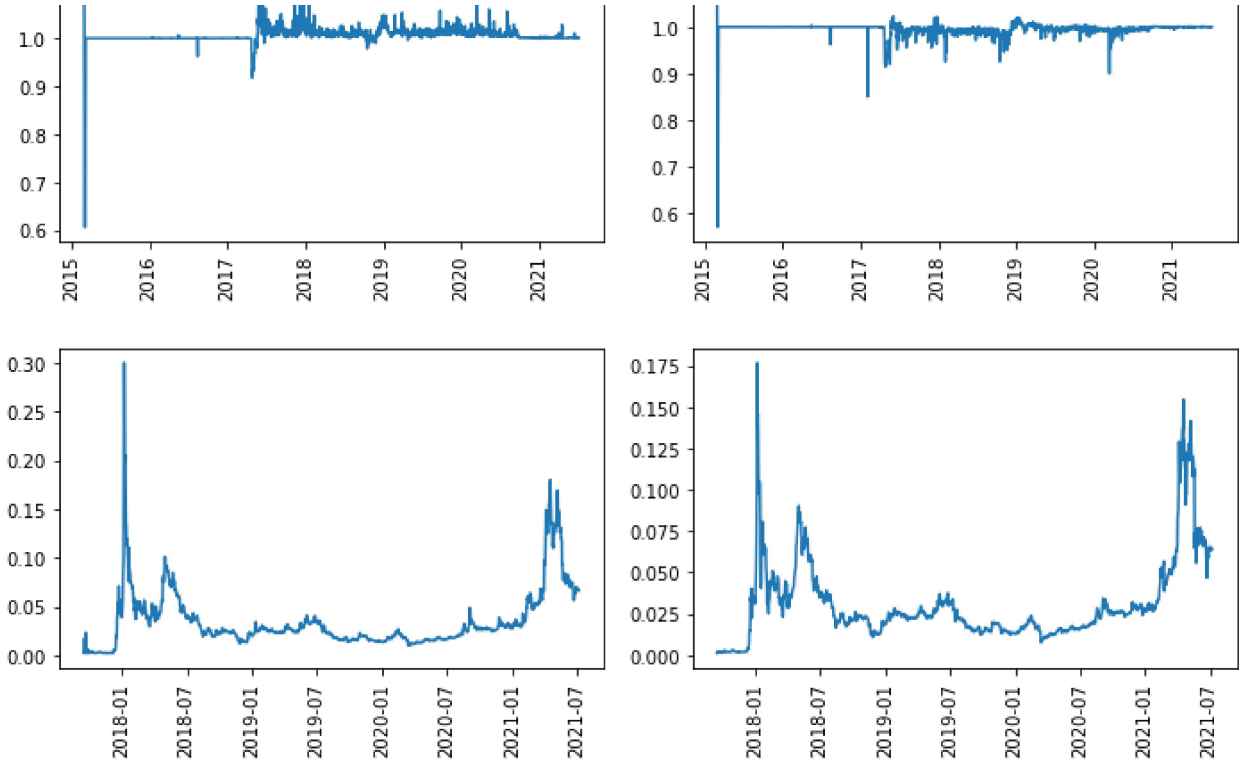

40
30
20
10
0
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07


0.8
0.6
0.4
0.2
0.0
2015
2016
2017
2018
2019
2020
2021


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2015
2016
2017
2018
2019
2020
2021


1.2
1.1

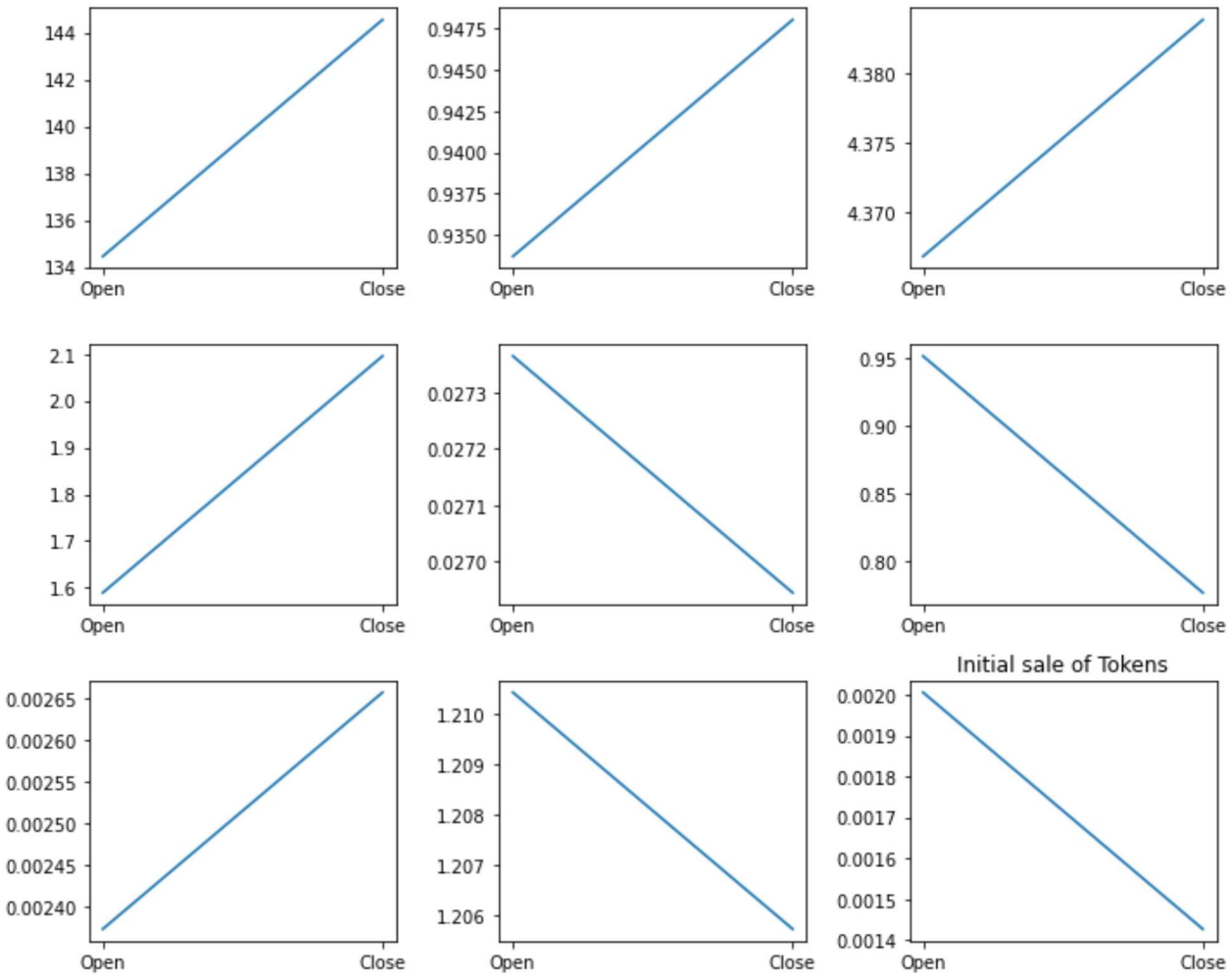

1.2
1.1



Ploting Open vs Close graphs

In [ ]:

```
fig, ax = plt.subplots(nrows=3 , ncols=3 , figsize=(10,8))
plt.title('Initial sale of Tokens')
a = bitcoin[['Open', 'Close']].iloc[0]
ax[0][0].plot(a)

b = ethereum[['Open', 'Close']].iloc[0]
ax[0][1].plot(b)

c = Litecoin[['Open', 'Close']].iloc[0]
ax[0][2].plot(c)

d = Monero[['Open', 'Close']].iloc[0]
ax[1][0].plot(d)

e = Ripple[['Open', 'Close']].iloc[0]
ax[1][1].plot(e)

f = Solana[['Open', 'Close']].iloc[0]
ax[1][2].plot(f)

g = Stellar[['Open', 'Close']].iloc[0]
ax[2][0].plot(g)

h = Tether[['Open', 'Close']].iloc[0]
ax[2][1].plot(h)

i = Tron[['Open', 'Close']].iloc[0]
ax[2][2].plot(i)


fig.tight_layout()
```



Performing Candlestick visualization for better understanding

In [ ]:
```
bitcoin = pd.read_csv('D:\python\data\coin_Bitcoin.csv')
ethereum = pd.read_csv('D:\python\data\coin_Ethereum.csv')
Litecoin = pd.read_csv('D:\python\data\coin_Litecoin.csv')
Monero = pd.read_csv('D:\python\data\coin_Monero.csv')
Ripple = pd.read_csv('D:\python\data\coin_Ripple.csv')
Solana = pd.read_csv('D:\python\data\coin_Solana.csv')
Stellar = pd.read_csv('D:\python\data\coin_Stellar.csv')
Tether = pd.read_csv('D:\python\data\coin_Tether.csv')
Tron = pd.read_csv('D:\python\data\coin_Tron.csv')
```

In [ ]:
```
from plotly.offline import plot
```

In [ ]:
```
fig = go.Figure(data=[go.Candlestick(x=bitcoin['Date'],
                                     open= bitcoin['Open'],
                                     high= bitcoin['High'],
                                     low=bitcoin['Low'],
                                     close = bitcoin['Low'])])
fig.update_layout(title = 'Bitcoin', yaxis_title = 'Price', xaxis_title='Duration',auto
fig.show()
```

In [ ]:

```
fig1 = go.Figure(data=[go.Candlestick(x=ethereum['Date'],
                                      open= ethereum['Open'],
                                      high= ethereum['High'],
                                      low=ethereum['Low'],
                                      close = ethereum['Low'])])
fig1.update_layout(title = 'ethereum', yaxis_title = 'Price', xaxis_title='Duration',au
fig1.show()
```

In [ ]:
```
fig2 = go.Figure(data=[go.Candlestick(x=Litecoin['Date'],
                                      open= Litecoin['Open'],
                                      high= Litecoin['High'],
                                      low=Litecoin['Low'],
                                      close = Litecoin['Low'])])
fig2.update_layout(title = 'Litecoin', yaxis_title = 'Price', xaxis_title='Duration',au
fig2.show()
```

In [ ]:
```
fig3 = go.Figure(data=[go.Candlestick(x=Monero['Date'],
                                      open= Monero['Open'],
                                      high= Monero['High'],
                                      low=Monero['Low'],
                                      close = Monero['Low'])])
fig3.update_layout(title = 'Monero', yaxis_title = 'Price', xaxis_title='Duration',auto
fig3.show()
```

In [ ]:
```
fig4 = go.Figure(data=[go.Candlestick(x=Ripple['Date'],
                                      open= Ripple['Open'],
                                      high= Ripple['High'],
                                      low=Ripple['Low'],
                                      close = Ripple['Low'])])
fig4.update_layout(title = 'Ripple', yaxis_title = 'Price', xaxis_title='Duration',auto
fig4.show()
```

In [ ]:
```
fig5 = go.Figure(data=[go.Candlestick(x=Solana['Date'],
                                      open= Solana['Open'],
                                      high= Solana['High'],
                                      low=Solana['Low'],
                                      close = Solana['Low'])])
fig5.update_layout(title = 'Solana', yaxis_title = 'Price', xaxis_title='Duration',auto
fig5.show()
```

In [ ]:
```
fig6 = go.Figure(data=[go.Candlestick(x=Stellar['Date'],
                                      open= Stellar['Open'],
                                      high= Stellar['High'],
                                      low=Stellar['Low'],
                                      close = Stellar['Low'])])
fig6.update_layout(title = 'Stellar', yaxis_title = 'Price', xaxis_title='Duration',aut
fig6.show()
```

In [ ]:
```
fig7 = go.Figure(data=[go.Candlestick(x=Tether['Date'],
                                      open= Tether['Open'],
                                      high= Tether['High'],
                                      low=Tether['Low'],
                                      close = Tether['Low'])])
fig7.update_layout(title = 'Tether', yaxis_title = 'Price', xaxis_title='Duration',auto
fig7.show()
```

In [ ]:
```
fig8 = go.Figure(data=[go.Candlestick(x=Tron['Date'],
                                      open= Tron['Open'],
                                      high= Tron['High'],
                                      low=Tron['Low'],
                                      close = Tron['Low'])])
fig8.update_layout(title = 'Tron', yaxis_title = 'Price', xaxis_title='Duration',autosi
fig8.show()
```

Using scatterplot to plot by **Year**

In [ ]:
```
bitcoin['Date'] = pd.to_datetime(bitcoin['Date'])
bitcoin['Year'] = pd.DatetimeIndex(bitcoin['Date']).year
ethereum['Date'] = pd.to_datetime(ethereum['Date'])
ethereum['Year'] = pd.DatetimeIndex(ethereum['Date']).year
Litecoin['Date'] = pd.to_datetime(Litecoin['Date'])
Litecoin['Year'] = pd.DatetimeIndex(Litecoin['Date']).year
Monero['Date'] = pd.to_datetime(Monero['Date'])
Monero['Year'] = pd.DatetimeIndex(Monero['Date']).year
Ripple['Date'] = pd.to_datetime(Ripple['Date'])
Ripple['Year'] = pd.DatetimeIndex(Ripple['Date']).year
Solana['Date'] = pd.to_datetime(Solana['Date'])
Solana['Year'] = pd.DatetimeIndex(Solana['Date']).year
Stellar['Date'] = pd.to_datetime(Stellar['Date'])
Stellar['Year'] = pd.DatetimeIndex(Stellar['Date']).year
Tether['Date'] = pd.to_datetime(Tether['Date'])
Tether['Year'] = pd.DatetimeIndex(Tether['Date']).year
Tron['Date'] = pd.to_datetime(Tron['Date'])
Tron['Year'] = pd.DatetimeIndex(Tron['Date']).year
```

In [ ]:
```
start_date = 2015
end_date = 2021
```

In [ ]:
```
BTH = (bitcoin['Year'] > start_date) & (bitcoin['Year'] <= end_date)
BTH = bitcoin.loc[BTH]

ETH = (ethereum['Year'] > start_date) & (ethereum['Year'] <= end_date)
ETH = ethereum.loc[ETH]

LTC = (Litecoin['Year'] > start_date) & (Litecoin['Year'] <= end_date)
LTC = Litecoin.loc[LTC]

XMR = (Monero['Year'] > start_date) & (Monero['Year'] <= end_date)
XMR = Monero.loc[XMR]
```

```
XRP = (Ripple['Year'] > start_date) & (Ripple['Year'] <= end_date)
XRP = Ripple.loc[XRP]

SOL = (Solana['Year'] > start_date) & (Solana['Year'] <= end_date)
SOL = Solana.loc[SOL]

XLM = (Stellar['Year'] > start_date) & (Stellar['Year'] <= end_date)
XLM = Stellar.loc[XLM]

USDT = (Tether['Year'] > start_date) & (Tether['Year'] <= end_date)
USDT = Tether.loc[USDT]

TRX = (Tron['Year'] > start_date) & (Tron['Year'] <= end_date)
TRX = Tron.loc[TRX]
```

In [ ]:

```
concat = pd.concat([BTH.assign(dataset='BTH'), ETH.assign(dataset='ETH'),LTC.assign(dat
                    SOL.assign(dataset='SOL'), XLM.assign(dataset='XLM'), USDT.assign(d
plt.figure(figsize=(20,10))
sns.scatterplot(data=concat, x='Year', y='Close', hue='dataset', style='dataset', palet
```

Out[ ]:

```
<AxesSubplot:xlabel='Year', ylabel='Close'>
```