HW3

December 19, 2021

1 HW3

[1]:
```
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from functools import reduce
import math

%matplotlib inline
```

[2]:
```
#Read csv data into pandas DataFrame
market_caps = pd.read_csv('data/hw3.csv')
market_caps.ts = pd.to_datetime(market_caps.ts)
market_caps = market_caps.set_index('ts')
```

[3]:
```
#Check time series
market_caps.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2667 entries, 2021-12-02 14:00:00 to 2021-12-06 07:00:00
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   open       2667 non-null   float64
 1   high       2163 non-null   float64
 2   low        2378 non-null   float64
 3   close      2531 non-null   float64
 4   volume     2667 non-null   float64
 5   volumeUSD  0 non-null      float64
 6   token      2667 non-null   object
 7   chain      2667 non-null   object
dtypes: float64(6), object(2)
memory usage: 187.5+ KB
```

We can see that we have 2667 total entries, but high, low and close have a bunch of nulls.

[4]:
```
market_caps['token'].value_counts()
```

[4]:
```
BTC                                    323
COMP                                   322
CRV                                    318
SOL                                    318
USDT                                   314
UNI                                    312
AAVE                                   312
ETH                                    298
<span name="tokenName">UNI</span>       30
<span name="tokenName">ETH</span>       28
<span name="tokenName">USDT</span>      19
<span name="tokenName">CRV</span>       17
<span name="tokenName">SOL</span>       16
<span name="tokenName">AAVE</span>      16
<span name="tokenName">BTC</span>       14
<span name="tokenName">COMP</span>      10
Name: token, dtype: int64
```

We also have some badly formatted token names.

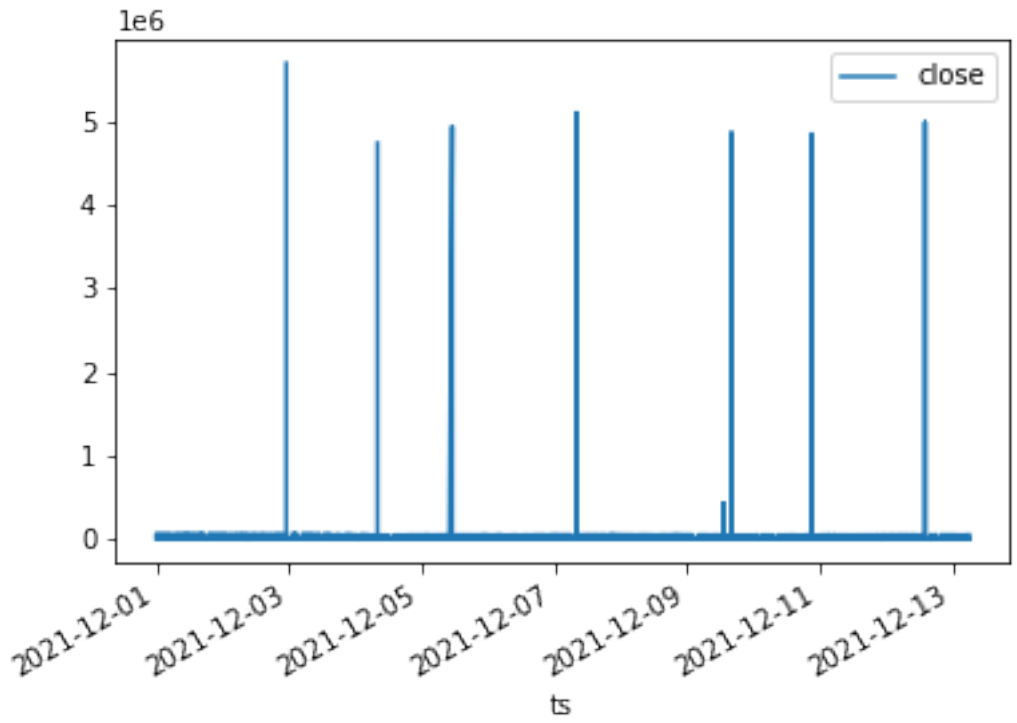
[5]:
```
market_caps.duplicated().sum()
```

[5]:
```
307
```

We also have some duplicated rows.

[6]:
```
market_caps[['token', 'close']].plot()
```

[6]:
```
<AxesSubplot:xlabel='ts'>
```



[7]:
```
#Plot by token
tokens_in_market_caps = market_caps.token.unique()

fig, axes = plt.subplots(nrows=math.ceil(tokens_in_market_caps.size / 2),␣
 ↪ncols=2, figsize=(15, 5 * math.ceil(tokens_in_market_caps.size / 2)))

idx = 0
for label, market_caps_token in market_caps[['token', 'close']].
 ↪groupby('token'):
    market_caps_token['close'].plot(ax=axes[idx // 2, idx % 2], label=label)
    axes[idx // 2, idx % 2].legend()

    idx += 1
```


AAVE
BTC
COMP
CRV
ETH
SOL
UNI
USDT
AAVE
BTC
COMP
CRV
ETH
SOL
UNI
USDT
1e6
ts
ts

[8]:
```
#Remove the duplicated rows
market_caps = market_caps.drop_duplicates()
```

[9]:
```
#Format the token names (Tried string.extract yet didn't work T_T)
market_caps['token'] = market_caps['token'].replace({'<span␣
 ↪name="tokenName">UNI</span>': 'UNI',
                                                      '<span␣
 ↪name="tokenName">ETH</span>': 'ETH',
                                                      '<span␣
 ↪name="tokenName">USDT</span>': 'USDT',
                                                      '<span␣
 ↪name="tokenName">CRV</span>': 'CRV',
                                                      '<span␣
 ↪name="tokenName">AAVE</span>': 'AAVE',
                                                      '<span␣
 ↪name="tokenName">SOL</span>': 'SOL',
                                                      '<span␣
 ↪name="tokenName">BTC</span>': 'BTC',
                                                      '<span␣
 ↪name="tokenName">COMP</span>': 'COMP'})
```

[10]:
```
#Filling missing high and low with open since all open values are available
market_caps.loc[market_caps['high'].isnull(), 'high'] = market_caps.
 ↪loc[market_caps['high'].isnull(), 'open']
market_caps.loc[market_caps['low'].isnull(), 'low'] = market_caps.
 ↪loc[market_caps['low'].isnull(), 'open']
```

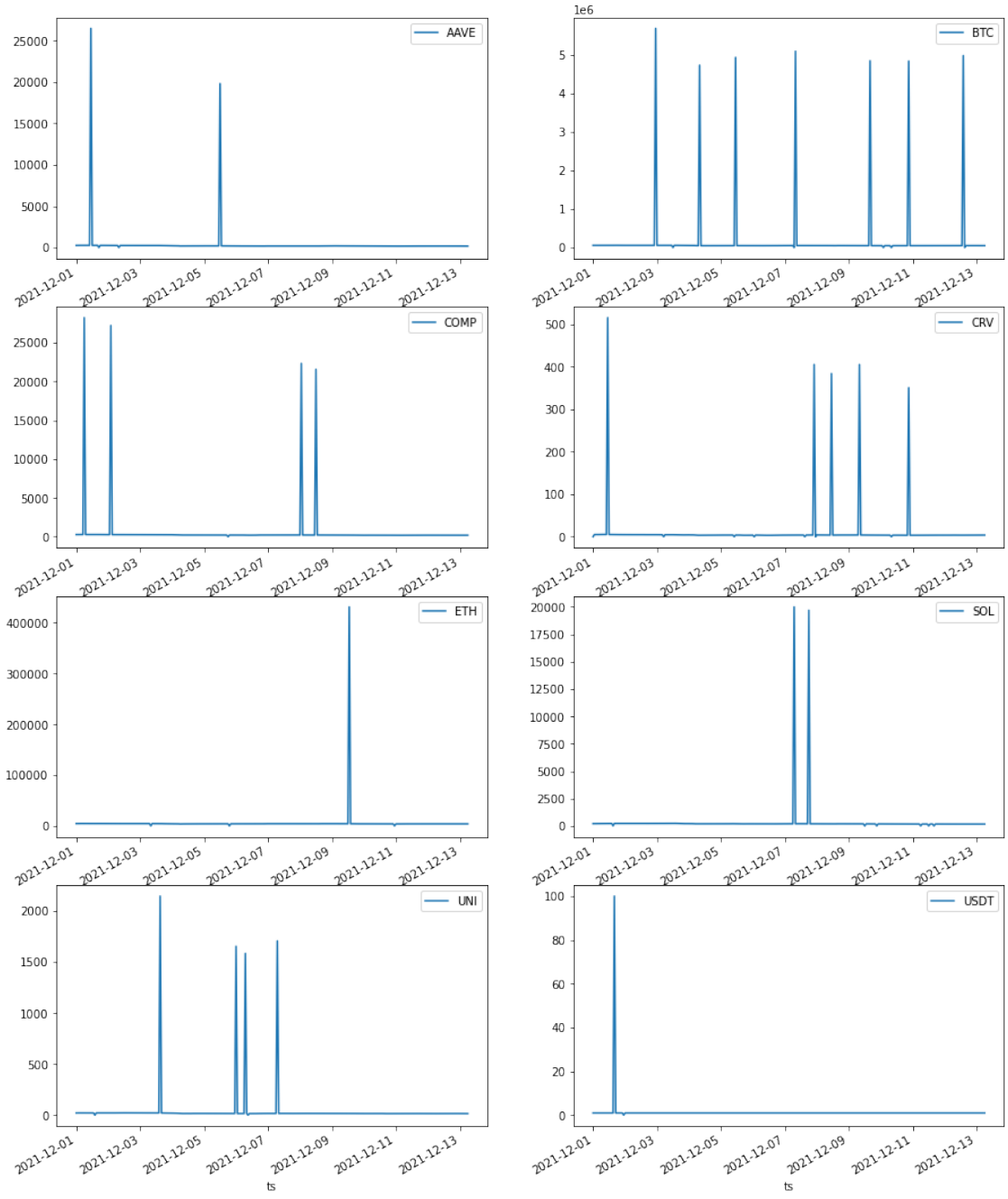
[11]:
```
#Use a heuristic for missing close price as average of high + low of the day
market_caps.loc[market_caps['close'].isnull(), 'close'] = 0.5 * (market_caps.
 ↪loc[market_caps['close'].isnull(), 'high'] + market_caps.
 ↪loc[market_caps['close'].isnull(), 'low'])
```

[12]:
```
#Plot by token
tokens_in_market_caps = market_caps.token.unique()

fig, axes = plt.subplots(nrows=math.ceil(tokens_in_market_caps.size / 2),␣
 ↪ncols=2, figsize=(15, 5 * math.ceil(tokens_in_market_caps.size / 2)))

idx = 0
for label, market_caps_token in market_caps[['token', 'close']].
 ↪groupby('token'):
    market_caps_token['close'].plot(ax=axes[idx // 2, idx % 2], label=label)
    axes[idx // 2, idx % 2].legend()
```

```
    idx += 1
```



Now we have good names, no duplicates and no missing values; However, we still have a lot massive outliers.

[13]:
```
#Come up with a heuristic to define an outlier
market_caps.loc[market_caps.close / market_caps.open >= 2]
```

[13]:

ts	open	high	low	close	volume	volumeUSD	token	chain
2021-12-10 21:00:00	3.4600	3.5100	3.4600	351.00	1.080245e+05	NaN	CRV	ETH
2021-12-12 14:00:00	49542.3900	49542.3900	49534.7100	4983812.00	3.426704e+02	NaN	BTC	BTC
2021-12-07 08:00:00	51224.9900	51398.2600	50883.3400	5097338.00	4.433800e+02	NaN	BTC	BTC
2021-12-07 18:00:00	197.0930	197.8000	194.3450	19698.90	6.369084e+04	NaN	SOL	SOL
2021-12-08 01:00:00	223.8500	224.9800	220.3900	22333.00	1.477532e+03	NaN	COMP	ETH
2021-12-08 12:00:00	214.8900	217.1000	212.3800	21568.00	2.372441e+03	NaN	COMP	ETH
2021-12-02 23:00:00	56904.6500	56952.8100	56639.4900	5688883.00	3.821792e+02	NaN	BTC	BTC
2021-12-02 02:00:00	269.7800	272.1200	268.2900	27212.00	4.564180e+02	NaN	COMP	ETH
2021-12-06 07:00:00	16.0391	16.2298	15.8308	1584.74	7.441640e+04	NaN	UNI	ETH
2021-12-01 11:00:00	266.9360	267.9630	264.2370	26501.50	1.018631e+03	NaN	AAVE	ETH
2021-12-03 15:00:00	21.7247	21.7884	21.4125	2145.88	2.317697e+04	NaN	UNI	ETH
2021-12-04 08:00:00	47711.3400	47869.7300	46750.0000	4734782.00	1.661132e+03	NaN	BTC	BTC
2021-12-09 16:00:00	48600.0100	48806.9000	48100.0000	4850319.00	1.509217e+03	NaN	BTC	BTC
2021-12-05 11:00:00	49113.5500	49113.5500	49113.5500	4936795.00	6.211598e+02	NaN	BTC	BTC
2021-12-05 12:00:00	198.8770	199.5750	196.7450	19832.40	2.262924e+03	NaN	AAVE	ETH
2021-12-09 13:00:00	4307.4400	4307.4400	4268.8800	431046.00	6.367062e+03	NaN	ETH	ETH
2021-12-01 16:00:00	1.0009	1.0009	1.0009	100.10				
2021-12-10 21:00:00	48123.2900	48479.7800	48123.2900	4842749.00				
2021-12-07 07:00:00	16.9835	17.1448	16.9514	1706.66				
2021-12-09 08:00:00	4.0880	4.1152	3.9977	405.70				
2021-12-01 06:00:00	282.6500	283.7300	282.0000	28222.00				
2021-12-07 07:00:00	197.2230	200.2290	197.2230	20010.00				
2021-12-07 22:00:00	3.9266	4.1266	3.9146	405.74				
2021-12-08 11:00:00	3.9238	3.9745	3.8301	384.45				
2021-12-06 00:00:00	16.5095	16.6871	16.3847	1654.04				
2021-12-01 11:00:00	5.2728	5.2882	5.1510	516.10				

```
2021-12-01 16:00:00  5.186434e+06        NaN  USDT  USDT
2021-12-10 21:00:00  7.488489e+02        NaN   BTC   BTC
2021-12-07 07:00:00  4.928282e+04        NaN   UNI   ETH
2021-12-09 08:00:00  1.787248e+05        NaN   CRV   ETH
2021-12-01 06:00:00  3.701940e+02        NaN  COMP   ETH
2021-12-07 07:00:00  8.352123e+04        NaN   SOL   SOL
2021-12-07 22:00:00  7.591058e+05        NaN   CRV   ETH
2021-12-08 11:00:00  2.061670e+05        NaN   CRV   ETH
2021-12-06 00:00:00  3.706263e+04        NaN   UNI   ETH
2021-12-01 11:00:00  2.459183e+05        NaN   CRV   ETH
```

This heuristic looks okay. Use the missing value heuristic to fill them in:

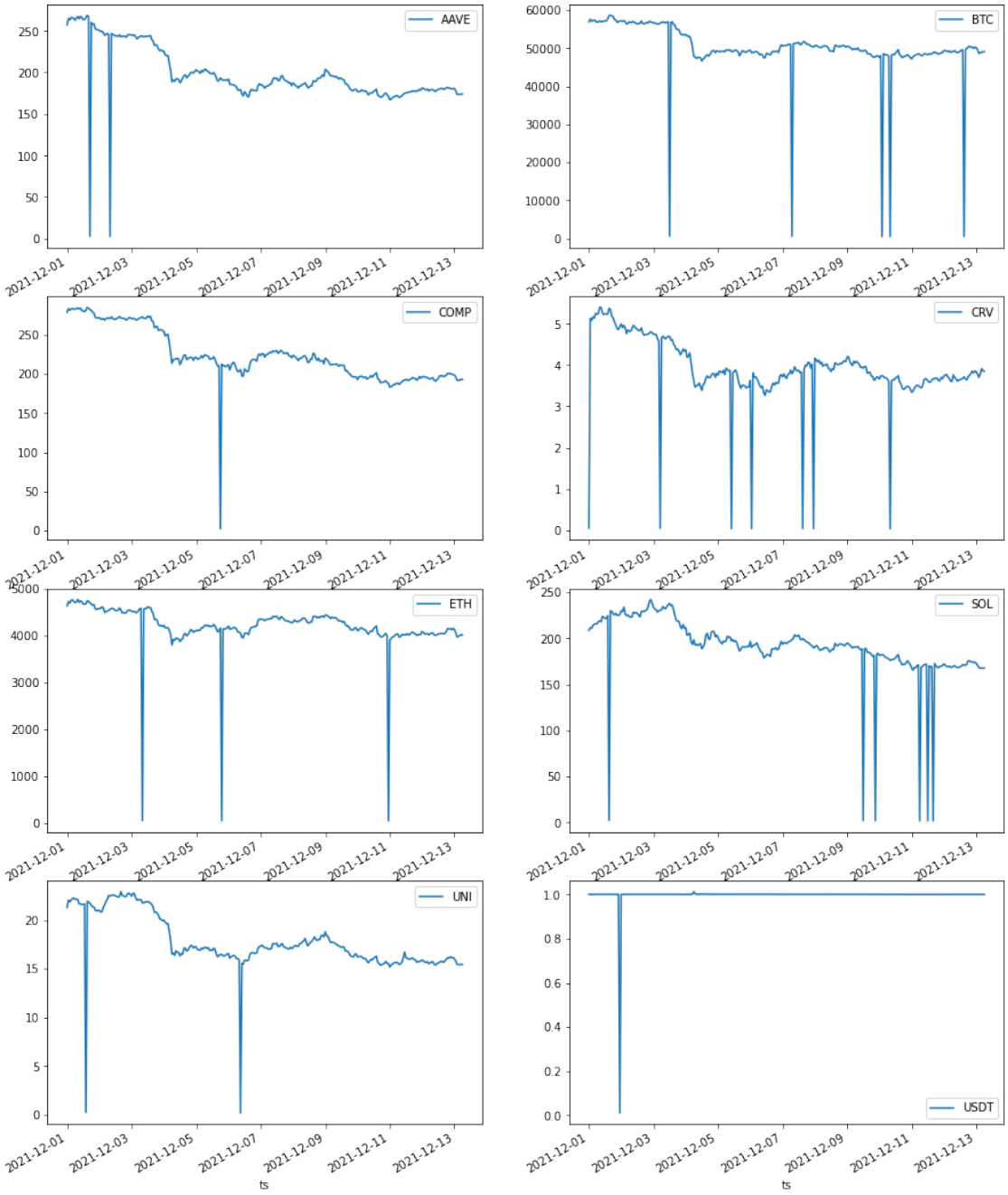
[14]:
```
#Replace outliers with average of high and low values
market_caps.loc[market_caps.close / market_caps.open >= 2, 'close'] = 0.5 *␣
↪(market_caps.loc[market_caps.close / market_caps.open >= 2, 'high'] +␣
↪market_caps.loc[market_caps.close / market_caps.open >= 2, 'low'])
```

[15]:
```
#Plot by token
tokens_in_market_caps = market_caps.token.unique()

fig, axes = plt.subplots(nrows=math.ceil(tokens_in_market_caps.size / 2),␣
↪ncols=2, figsize=(15, 5 * math.ceil(tokens_in_market_caps.size / 2)))

idx = 0
for label, market_caps_token in market_caps[['token', 'close']].
↪groupby('token'):
    market_caps_token['close'].plot(ax=axes[idx // 2, idx % 2], label=label)
    axes[idx // 2, idx % 2].legend()

    idx += 1
```



Now we see outliers on the other side needs to be removed as well.

[16]:
```
#Check outliers on the other side
market_caps.loc[market_caps.open / market_caps.close >= 2]
```

[16]:

ts	open	high	low	close	\
2021-12-03 12:00:00	56891.7000	57209.7000	56858.0200	570.337400	

2021-12-07 07:00:00	50941.5000	51272.8300	50936.0500	512.250000
2021-12-06 01:00:00	3.6302	3.6365	3.4638	0.035758
2021-12-03 08:00:00	4584.3400	4588.0300	4548.0600	45.722700
2021-12-02 08:00:00	245.8090	247.5180	245.7290	2.467160
2021-12-10 23:00:00	3996.1700	3996.1700	3953.7700	39.613200
2021-12-01 17:00:00	265.2620	265.2620	262.9060	2.632300
2021-12-11 12:00:00	171.8500	172.4100	169.9900	1.700800
2021-12-10 08:00:00	3.5900	3.6300	3.5400	0.035700
2021-12-09 21:00:00	181.3980	181.9630	178.7750	1.809360
2021-12-01 15:00:00	224.5410	229.2300	224.4140	2.290990
2021-12-05 10:00:00	3.8692	3.8692	3.7310	0.037671
2021-12-05 18:00:00	210.7000	214.7300	208.9300	2.145100
2021-12-10 02:00:00	48051.1000	48579.4100	47797.8800	484.367900
2021-12-11 16:00:00	169.2800	171.2300	169.1200	1.711400
2021-12-05 19:00:00	4161.7300	4186.0500	4141.3600	41.543200
2021-12-09 12:00:00	188.5920	188.9660	188.5920	1.877750
2021-12-10 08:00:00	47875.2900	48048.3400	47568.0600	476.863200
2021-12-03 05:00:00	4.5835	4.5835	4.5533	0.046530
2021-12-01 00:00:00	4.8830	5.0723	4.8264	0.050295
2021-12-11 06:00:00	171.0000	171.0000	169.6200	1.702800
2021-12-01 14:00:00	21.6462	21.8957	21.6436	0.216669
2021-12-01 23:00:00	1.0008	1.0009	1.0005	0.010005
2021-12-07 15:00:00	3.8254	3.8761	3.8000	0.038411
2021-12-12 15:00:00	49831.3000	49960.0000	49549.1900	499.086700
2021-12-06 09:00:00	15.8728	15.8728	15.6670	0.159170
2021-12-07 23:00:00	4.0575	4.1472	3.9908	0.040037

ts	volume	volumeUSD	token	chain
2021-12-03 12:00:00	1.897550e+02	NaN	BTC	BTC
2021-12-07 07:00:00	3.124719e+02	NaN	BTC	BTC
2021-12-06 01:00:00	4.227913e+05	NaN	CRV	ETH
2021-12-03 08:00:00	2.468821e+03	NaN	ETH	ETH
2021-12-02 08:00:00	3.166740e+02	NaN	AAVE	ETH
2021-12-10 23:00:00	8.379371e+03	NaN	ETH	ETH
2021-12-01 17:00:00	9.809190e+02	NaN	AAVE	ETH
2021-12-11 12:00:00	1.572705e+04	NaN	SOL	SOL
2021-12-10 08:00:00	1.034540e+05	NaN	CRV	ETH
2021-12-09 21:00:00	1.044998e+05	NaN	SOL	SOL
2021-12-01 15:00:00	1.160250e+05	NaN	SOL	SOL
2021-12-05 10:00:00	1.789825e+05	NaN	CRV	ETH
2021-12-05 18:00:00	1.282728e+03	NaN	COMP	ETH
2021-12-10 02:00:00	4.690459e+02	NaN	BTC	BTC
2021-12-11 16:00:00	4.090014e+04	NaN	SOL	SOL
2021-12-05 19:00:00	8.791289e+03	NaN	ETH	ETH
2021-12-09 12:00:00	2.331976e+04	NaN	SOL	SOL
2021-12-10 08:00:00	7.834112e+02	NaN	BTC	BTC

```
2021-12-03 05:00:00  8.807282e+04        NaN   CRV   ETH
2021-12-01 00:00:00  3.395810e+05        NaN   CRV   ETH
2021-12-11 06:00:00  3.492728e+04        NaN   SOL   SOL
2021-12-01 14:00:00  2.549481e+04        NaN   UNI   ETH
2021-12-01 23:00:00  2.967950e+06        NaN  USDT  USDT
2021-12-07 15:00:00  1.519071e+05        NaN   CRV   ETH
2021-12-12 15:00:00  3.623237e+02        NaN   BTC   BTC
2021-12-06 09:00:00  1.543971e+05        NaN   UNI   ETH
2021-12-07 23:00:00  6.614293e+05        NaN   CRV   ETH
```

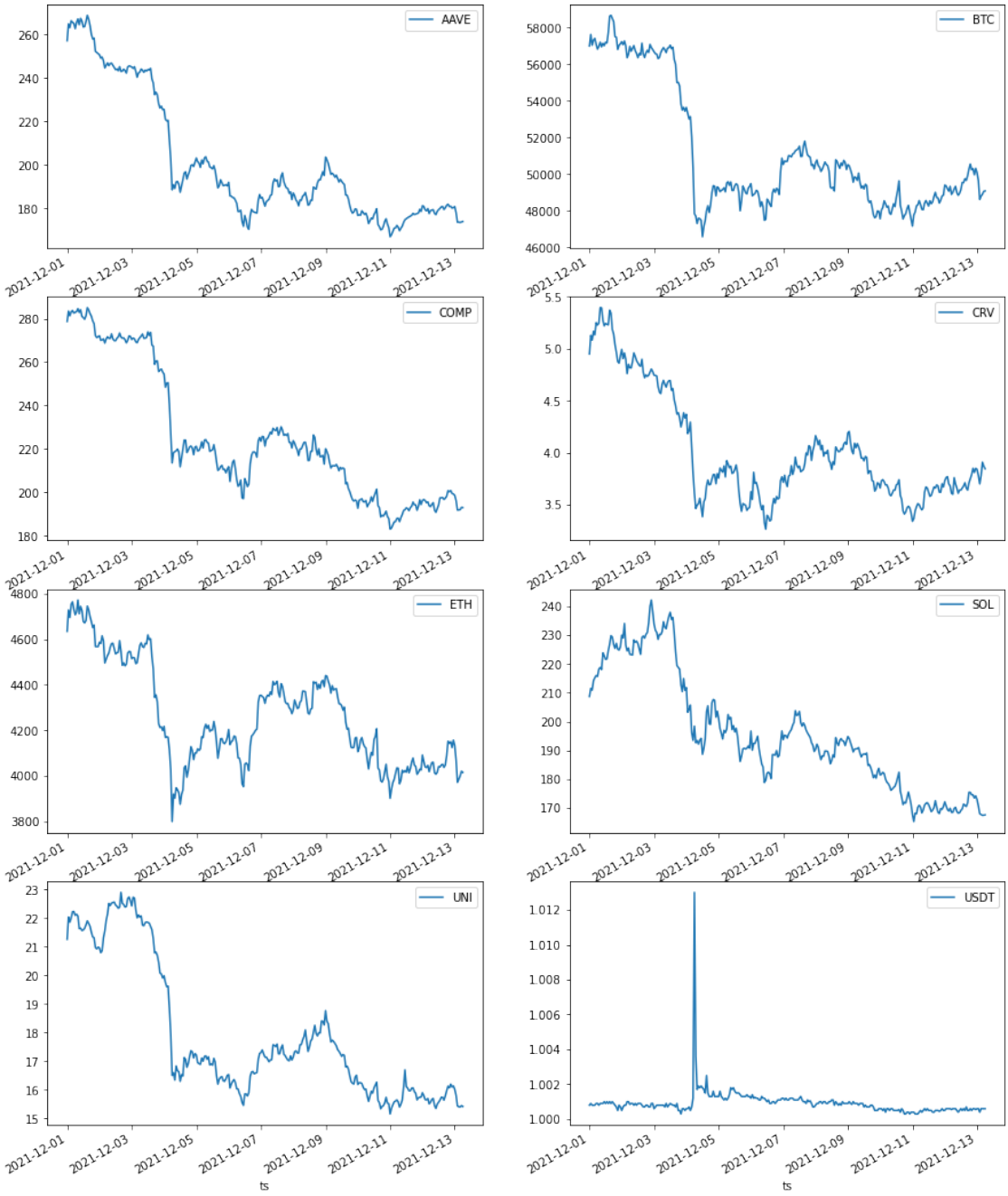
[17]:
```
#Replace outliers with average of high and low values
market_caps.loc[market_caps.open / market_caps.close >= 2, 'close'] = 0.5 *␣
 ↪(market_caps.loc[market_caps.open / market_caps.close >= 2, 'high'] +␣
 ↪market_caps.loc[market_caps.open / market_caps.close >= 2, 'low'])
```

[18]:
```
#Plot by token
tokens_in_market_caps = market_caps.token.unique()

fig, axes = plt.subplots(nrows=math.ceil(tokens_in_market_caps.size / 2),␣
 ↪ncols=2, figsize=(15, 5 * math.ceil(tokens_in_market_caps.size / 2)))

idx = 0
for label, market_caps_token in market_caps[['token', 'close']].
 ↪groupby('token'):
    market_caps_token['close'].plot(ax=axes[idx // 2, idx % 2], label=label)
    axes[idx // 2, idx % 2].legend()

    idx += 1
```



Now we have a much cleaner dataset. Calculate missing volueUSD = volume * close.

[19]:
```
#Calculate volumeUSD = volume * close
market_caps['volumeUSD'] = market_caps['volume'] * market_caps['close']
```

[20]:
```
#Calculate volumeUSD by chain
market_caps.groupby(['chain'])['volumeUSD'].sum().to_frame()
```

[20]:

chain	volumeUSD
BTC	1.095893e+10
ETH	1.364762e+10
SOL	4.075514e+09
USDT	1.008247e+09