In [1]:
```
#With the earthquakes.csv file, select all the earthquakes in Japan with amagnitude of
#earthquakes (1).csv

#load pd
import pandas as pd

#load csv
earthquakes = pd.read_csv('earthquakes (1).csv')

#filter
#parsed_place
#mag
#magType
japan_earthquakes = earthquakes[(earthquakes['parsed_place'] == 'Japan') & (earthquake
japan_earthquakes
```

Out[1]:

	mag	magType	time	place	tsunami	parsed_place
1563	4.9	mb	1.538980e+12	293km ESE of Iwo Jima, Japan	0	Japan
2576	5.4	mb	1.538700e+12	37km E of Tomakomai, Japan	0	Japan
3072	4.9	mb	1.538580e+12	15km ENE of Hasaki, Japan	0	Japan
3632	4.9	mb	1.538450e+12	53km ESE of Hitachi, Japan	0	Japan

In [2]:
```
#Create bins for each full number of earthquake magnitude (for instance, the first bin

#load pd
import pandas as pd

#earthquakes (1).csv
earthquakes = pd.read_csv('earthquakes (1).csv')

#filter by ml
#magType
ml_earthquakes = earthquakes[earthquakes['magType'] == 'ml']

#mag
max_magnitude = int(ml_earthquakes['mag'].max()) + 1
bins = range(0, max_magnitude + 1)

#make bins
#mag
bin_counts = pd.cut(ml_earthquakes['mag'], bins=bins, right=False, include_lowest=True
print(bin_counts)
```

```
[0, 1)    2072
[1, 2)    3126
[2, 3)     985
[3, 4)     153
[4, 5)       6
[5, 6)       2
Name: mag, dtype: int64
```

In [3]:
```
#Using the faang.csv file, group by the ticker and resample to monthly frequency.Make

#load pd
```

```
import pandas as pd

#faang.csv
faang = pd.read_csv('faang.csv')

#do date time format
faang['date'] = pd.to_datetime(faang['date'])

#set date index
faang.set_index('date', inplace=True)

#mean of open, max of high, min of low, mean of close, sum of volume
aggregated = (faang.groupby('ticker').resample('M').agg({'open': 'mean','high': 'max',
print(aggregated)
```

```
                        open        high         low       close      volume
ticker date
AAPL   2019-01-31   38.402143   42.250000   35.500000   38.541548  3312349600
       2019-02-28   42.848027   43.967499   41.482498   42.931973  1890162400
       2019-03-31   45.805953   49.422501   42.375000   45.823453  2603925600
       2019-04-30   49.966666   52.119999   47.095001   50.129048  2024470800
       2019-05-31   47.734659   53.827499   43.747501   47.818409  2957826400
...                       ...         ...         ...         ...         ...
NFLX   2020-08-31  496.385713  549.039978  466.549988  498.074762   116269200
       2020-09-30  496.910954  557.390015  458.600006  495.553333   118806100
       2020-10-31  517.601816  572.489990  472.209992  515.141814   154286700
       2020-11-30  486.917998  518.729981  463.410004  487.444000    91773400
       2020-12-31  514.014094  545.500000  491.290008  516.569090    77372300

[120 rows x 5 columns]
```

In [5]:

```
#Build a crosstab with the earthquake data between the tsunami column and themagType c

#Load pd
import pandas as pd

##earthquakes (1).csv
earthquakes = pd.read_csv('earthquakes (1).csv')

#make crosstab
#mag
#magType
#tsunami
crosstab_result = pd.crosstab(earthquakes['tsunami'], earthquakes['magType'],
values=earthquakes['mag'], aggfunc='max')
print(crosstab_result)
```

```
magType    mb  mb_lg    md   mh   ml  ms_20    mw  mwb  mwr  mww
tsunami
0         5.6    3.5  4.11  1.1  4.2    NaN  3.83  5.8  4.8  6.0
1         6.1    NaN   NaN  NaN  5.1    5.7  4.41  NaN  NaN  7.5
```

In [6]:

```
#Calculate the rolling 60-day aggregations of the OHLC data by ticker for theFAANG dat

#Load pd
import pandas as pd

#faang.csv
faang = pd.read_csv('faang.csv')
```

```
#set date time for index
faang['date'] = pd.to_datetime(faang['date'])
faang.set_index('date', inplace=True)

#group by ticker and aggregate
#mean of open, max of high, min of low, mean of close, sum of volume
rolling_aggregations = (faang.groupby('ticker').rolling(window='60D').agg({'open': 'me
print(rolling_aggregations)
```

```
                         open        high         low       close  \
ticker date
AAPL   2019-01-02   38.722500   39.712502   38.557499   39.480000
       2019-01-03   37.358749   39.712502   35.500000   37.513750
       2019-01-04   36.949999   39.712502   35.500000   37.364166
       2019-01-07   37.006249   39.712502   35.500000   37.268749
       2019-01-08   37.082999   39.712502   35.500000   37.352499
...                       ...         ...         ...         ...
NFLX   2020-12-24  497.874185  536.369995  463.410004  498.638371
       2020-12-28  499.327999  536.369995  463.410004  499.823750
       2020-12-29  499.775249  536.549988  463.410004  501.202000
       2020-12-30  500.515609  536.549988  463.410004  501.772439
       2020-12-31  501.111191  545.500000  463.410004  502.700000

                         volume
ticker date
AAPL   2019-01-02  1.481588e+08
       2019-01-03  5.134076e+08
       2019-01-04  7.478360e+08
       2019-01-07  9.669472e+08
       2019-01-08  1.131048e+09
...                         ...
NFLX   2020-12-24  1.897038e+08
       2020-12-28  1.656661e+08
       2020-12-29  1.618806e+08
       2020-12-30  1.637569e+08
       2020-12-31  1.691457e+08

[2524 rows x 5 columns]
```

In [7]:
```
#Create a pivot table of the FAANG data that compares the stocks. Put the ticker inthe

#load pd
import pandas as pd

#faang.csv
faang = pd.read_csv('faang.csv')

#pivot table
pivot_table = faang.pivot_table(index='ticker', values=['open', 'high', 'low', 'close'
print(pivot_table)
```

```
              close         high          low         open        volume
ticker
AAPL      73.748386    74.603614    72.782277    73.660342  1.348890e+08
AMZN    2235.904988  2260.671027  2208.550118  2235.758614  4.400448e+06
GOOG    1335.188544  1348.282903  1320.577058  1333.577882  1.653233e+06
META     207.994504   210.617381   205.128135   207.869425  1.933570e+07
NFLX     387.966593   393.779485   381.346595   387.505604  7.394716e+06
```

In [9]:
```
#Calculate the Z-scores for each numeric column of Amazon's data (ticker isAMZN) in Q4

#load pd
import pandas as pd

#faang.csv
faang = pd.read_csv('faang.csv')

#date time format
faang['date'] = pd.to_datetime(faang['date'])

#data for AMZN q4 2018
amazon_q4_2018 = faang[(faang['ticker'] == 'AMZN') & (faang['date'] >= '2018-10-01') &

#z scores
z_scores = amazon_q4_2018[['open', 'high', 'low', 'close', 'volume']].apply(
    lambda x: (x - x.mean()) / x.std())
print(z_scores)
```

```
Empty DataFrame
Columns: [open, high, low, close, volume]
Index: []
```

In [1]:
```
#Create a dataframe with the following three columns: ticker, date, andevent. The colu

#load pd
import pandas as pd

#dataframe
data = { 'ticker': 'FB', 'date': ['2018-07-25', '2018-03-19', '2018-03-20'], 'event':
df = pd.DataFrame(data)

#set index to date and ticker
df.set_index(['date', 'ticker'], inplace=True)
print(df)
```

```
                                                    event
date       ticker
2018-07-25 FB      Disappointing user growth announced after close.
2018-03-19 FB                           Cambridge Analytica story
2018-03-20 FB                                  FTC investigation
```

In [6]:
```
#Merge this data with the FAANG data using an outer join

#load pd
import pandas as pd

#new dataframe
data = {'ticker': ['FB', 'FB', 'FB'], 'date': ['2018-07-25', '2018-03-19', '2018-03-20
'event': ['Disappointing user growth announced after close.',  'Cambridge Analytica st
df_events = pd.DataFrame(data)
#change date to datetime
df_events['date'] = pd.to_datetime(df_events['date'])

#laod faang
faang = pd.read_csv('faang.csv')
#change date to datetime
faang['date'] = pd.to_datetime(faang['date'])

#merge (pd.merge)
```

```
merged_data = pd.merge(faang, df_events, on=['date', 'ticker'], how='outer')
print(merged_data)
```

```
           date         high          low         open        close  \
0    2019-01-02  1553.359985  1460.930054  1465.199951  1539.130005
1    2019-01-03  1538.000000  1497.109985  1520.010010  1500.280029
2    2019-01-04  1594.000000  1518.310059  1530.000000  1575.390015
3    2019-01-07  1634.560059  1589.189941  1602.310059  1629.510010
4    2019-01-08  1676.609985  1616.609985  1664.689941  1656.579956
...         ...          ...          ...          ...          ...
2522 2020-12-30   278.079987   271.709992   277.950012   271.869995
2523 2020-12-31   277.089996   269.809998   272.000000   273.160004
2524 2018-07-25          NaN          NaN          NaN          NaN
2525 2018-03-19          NaN          NaN          NaN          NaN
2526 2018-03-20          NaN          NaN          NaN          NaN

          volume    adj_close ticker  \
0      7983100.0  1539.130005   AMZN
1      6975600.0  1500.280029   AMZN
2      9182600.0  1575.390015   AMZN
3      7993200.0  1629.510010   AMZN
4      8881400.0  1656.579956   AMZN
...          ...          ...    ...
2522  11803800.0   271.869995   META
2523  12892900.0   273.160004   META
2524         NaN          NaN     FB
2525         NaN          NaN     FB
2526         NaN          NaN     FB

                                                event
0                                                 NaN
1                                                 NaN
2                                                 NaN
3                                                 NaN
4                                                 NaN
...                                               ...
2522                                              NaN
2523                                              NaN
2524  Disappointing user growth announced after close.
2525                          Cambridge Analytica story
2526                                  FTC investigation

[2527 rows x 9 columns]
```

In [7]:

```
#Use the transform() method on the FAANG data to represent all thevalues in terms of t

#load pd
import pandas as pd

#load faang
faang = pd.read_csv('faang.csv')
faang['date'] = pd.to_datetime(faang['date'])

#group by ticker and transform
indexed_faang = faang.groupby('ticker').transform(lambda x: x / x.iloc[0])

#non transformed columns
indexed_faang['ticker'] = faang['ticker']
indexed_faang['date'] = faang['date']
print(indexed_faang)
```

```
          high       low      open     close    volume  adj_close ticker  \
0     1.000000  1.000000  1.000000  1.000000  1.000000   1.000000   AMZN
1     0.990112  1.024765  1.037408  0.974758  0.873796   0.974758   AMZN
2     1.026163  1.039276  1.044226  1.023559  1.150255   1.023559   AMZN
3     1.052274  1.087793  1.093578  1.058721  1.001265   1.058721   AMZN
4     1.079344  1.106562  1.136152  1.076309  1.112525   1.076309   AMZN
...        ...       ...       ...       ...       ...        ...    ...
2519  1.966402  2.070629  2.084503  1.970814  0.238114   1.970814   META
2520  2.016581  2.066428  2.083417  2.041569  0.827149   2.041569   META
2521  2.039925  2.149036  2.147066  2.039947  0.582068   2.039947   META
2522  2.022253  2.113488  2.154818  2.003759  0.419375   2.003759   META
2523  2.015054  2.098709  2.108691  2.013267  0.458069   2.013267   META

           date
0    2019-01-02
1    2019-01-03
2    2019-01-04
3    2019-01-07
4    2019-01-08
...         ...
2519 2020-12-24
2520 2020-12-28
2521 2020-12-29
2522 2020-12-30
2523 2020-12-31

[2524 rows x 8 columns]
```

```
C:\Users\lexiw\AppData\Local\Temp\ipykernel_20868\1748592649.py:11: FutureWarning: Dr
opping invalid columns in DataFrameGroupBy.transform is deprecated. In a future versi
on, a TypeError will be raised. Before calling .transform, select only columns which
should be valid for the function.
  indexed_faang = faang.groupby('ticker').transform(lambda x: x / x.iloc[0])
```

In [ ]: