Hands-on Activity 8.1: Aggregating Data with Pandas

CPE311 Computational Thinking with Python

Name: Bautista, Jhon Hendricks
Section: CPE22S3

Performed on: 04/09/2025
Submitted on: 04/011/2025
Submitted to: Engr. Roman M. Richard

8.1.1 Intended Learning Outcomes

After this activity, the student should be able to:

- Demonstrate querying and merging of dataframes
- Perform advanced calculations on dataframes
- Aggregate dataframes with pandas and numpy
- Work with time series data

8.1.2 Resources

- Computing Environment using Python 3.x - Attached Datasets (under Instructional Materials)

8.1.3 Procedures

The procedures can be found in the canvas module. Check the following under topics:
8.1 Weather Data Collection
8.2 Querying and Merging
8.3 Dataframe Operations
8.4 Aggregations
8.5 Time Series

8.1.4 Data Analysis

Provide some comments here about the results of the procedures

8.1 Weather Data Collection

In the Weather Data Collection, I observed that a request was made to the website which contain the data. Then the data was succesfully collect with proper accuracy through the use of extending the results from the response. The initial data contains the different stations but still need further data which is done by requesting another time and adding the station details to the initial data.

8.2 Querying and Merging

In this procedure I learn the different ways of merging dataframes. First is that query() works like a SQL query which is why if you compare the result of the code from python and Sqlite it will be the same. Then, I observe that before merging we must observe the number of rows we are merging. Then, the default merge is 'INNER" join which keeps the rows with matching values. Then the 'OUTER' is to keep all the rows but will result 'Nan' when there is no mathcing value.

8.3 Dataframe Operations

This procedure mostly is concerned about the different modifications and transformation you can do to your dataset. This can be observed in the binning of the data which divides your values into specific range of values. This has different paramaters that you can set to change your range and precision of binning. This can be used to categorize your data which can be useful when making analysis. The apply() method can run the operation on an entire columns with just a lambda function.

8.4 Aggregations

For this procedure it was mostly giving summary of columns and analysis through the aggregation methods available. It was able to perform single and multiple aggregations by using a dictionary of cloumns and their respective aggregation. We can observed this to be used in viewing calculations for the OHLC data. Then for a more specific data field of calcualtion I obersved that groupby() is used. An example was the analysis of only 'trading_volume'. There is also a modification of the dataframe which uses the pivot and crosstab methods. The pivot method is used for reshaping your dataset which is done by setting a index from your chosen columns. On the other hand, cross_tab is used to show relationship of entries by creating a cross tabulation. This can be observed when creating a frequency table of OHLC per month.

8.5 Time Series

In this procedure I observe that most of the data was using a datetime index. This was benificially for doing analysis in different instances with respect to time. An example of this is the usage of first() method and then passing what type of first in a time based argument. You can also resample dataset through changing the frequnecy of the time series data. This can be seen in the anaylsis of OHLC data in different date time functions.

8.1.5 Supplementary Activity

1. With the earthquakes.csv file, select all the earthquakes in Japan with a magType of mb and a magnitude of 4.9 or greater.

2. Create bins for each full number of magnitude (for example, the first bin is 0-1, the second is 1-2, and so on) with a magType of ml and count how many are in each bin.

3. Using the faang.csv file, group by the ticker and resample to monthly frequency. Make the following aggregations:

    - Mean of the opening price
    - Maximum of the high price
    - Minimum of the low price
    - Mean of the closing price
    - Sum of the volume traded

4. Build a crosstab with the earthquake data between the tsunami column and the magType column. Rather than showing the frequency count, show the maximum magnitude that was observed for each combination. Put the magType along the columns.

5. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG data. Use the same aggregations as exercise no. 3.

6. Create a pivot table of the FAANG data that compares the stocks. Put the ticker in the rows and show the averages of the OHLC and volume traded data.

7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NFLX) using apply().

8. Add event descriptions: - Create a dataframe with the following three columns:
   - ticker, date, and event. The columns should have the following values: ticker: 'FB'
   - date: ['2018-07-25', '2018-03-19', '2018-03-20']
   - event: ['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC investigation']
   - Set the index to ['date', 'ticker']
   - Merge this data with the FAANG data using an outer join

9. . Use the transform() method on the FAANG data to represent all the values in terms of the first date in the data. To do so, divide all the values for each ticker by the values

for the first date in the data for that ticker. This is referred to as an index, and the data for the first date is the base (https:// ec. europa. eu/ eurostat/ statistics- explained/ index. php/ Beginners:Statistical concept - Index and base year). When data is in this format, we can easily see growth over time. Hint: transform() can take a function name

In [ ]:

In [12]:
```
# 1. With the earthquakes.csv file, select all the earthquakes in Japan with a magT
import pandas as pd

earth = pd.read_csv('earthquakes.csv')

earth.head() #checking data
```

Out[12]:

	mag	magType	time	place	tsunami	parsed_place
0	1.35	ml	1539475168010	9km NE of Aguanga, CA	0	California
1	1.29	ml	1539475129610	9km NE of Aguanga, CA	0	California
2	3.42	ml	1539475062610	8km NE of Aguanga, CA	0	California
3	0.44	ml	1539474978070	9km NE of Aguanga, CA	0	California
4	2.16	md	1539474716050	10km NW of Avenal, CA	0	California

In [16]:
```
# 1. With the earthquakes.csv file, select all the earthquakes in Japan with a magT

earth[(earth['magType'] == 'mb') & (earth['parsed_place'] == 'Japan') & (earth['mag
```

Out[16]:

	mag	magType	time	place	tsunami	parsed_place
1563	4.9	mb	1538977532250	293km ESE of Iwo Jima, Japan	0	Japan
2576	5.4	mb	1538697528010	37km E of Tomakomai, Japan	0	Japan
3072	4.9	mb	1538579732490	15km ENE of Hasaki, Japan	0	Japan
3632	4.9	mb	1538450871260	53km ESE of Hitachi, Japan	0	Japan

In [22]:
```
# 2. Create bins for each full number of magnitude (for example, the first bin is 0
# with a magType of ml and count how many are in each bin.

data = earth[earth['magType'] == 'ml'].copy()
bins = range(0, int(data['mag'].max()) + 2)
data['mag_bin'] = pd.cut(data['mag'], bins=bins, right=False)
```

In [30]:
```
counts = data['mag_bin'].value_counts().sort_index()
print(counts)
```

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

In [28]:
```
data.head() # checking of values
```

Out[28]:

	mag	magType	time	place	tsunami	parsed_place	mag_bin
0	1.35	ml	1539475168010	9km NE of Aguanga, CA	0	California	[1, 2)
1	1.29	ml	1539475129610	9km NE of Aguanga, CA	0	California	[1, 2)
2	3.42	ml	1539475062610	8km NE of Aguanga, CA	0	California	[3, 4)
3	0.44	ml	1539474978070	9km NE of Aguanga, CA	0	California	[0, 1)
6	1.70	ml	1539473176017	105km W of Talkeetna, Alaska	0	Alaska	[1, 2)

In [96]:

```
'''3. Using the faang.csv file, group by the ticker and resample to monthly frequen

Mean of the opening price
Maximum of the high price
Minimum of the low price
Mean of the closing price
Sum of the volume traded'''

# Loading the data
faang = pd.read_csv('faang.csv')

# Resampling
faang['date'] = pd.to_datetime(faang['date'])  # Convert 'date' column to datetime
faang = faang.set_index('date')                # Set it as index
faang.groupby(['ticker']).resample('ME').agg({  # pass the dict of columns that nee
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})
```

Out[96]:

ticker	date	open	high	low	close	volume
AAPL	2018-01-31	170.714690	176.6782	161.5708	170.699271	659679440
	2018-02-28	164.562753	177.9059	147.9865	164.921884	927894473
	2018-03-31	172.421381	180.7477	162.4660	171.878919	713727447
	2018-04-30	167.332895	176.2526	158.2207	167.286924	666360147
	2018-05-31	182.635582	187.9311	162.7911	183.207418	620976206
	2018-06-30	186.605843	192.0247	178.7056	186.508652	527624365
	2018-07-31	188.065786	193.7650	181.3655	188.179724	393843881
	2018-08-31	210.460287	227.1001	195.0999	211.477743	700318837
	2018-09-30	220.611742	227.8939	213.6351	220.356353	678972040
	2018-10-31	219.489426	231.6645	204.4963	219.137822	789748068
	2018-11-30	190.828681	220.6405	169.5328	190.246652	961321947
	2018-12-31	164.537405	184.1501	145.9639	163.564732	898917007
AMZN	2018-01-31	1301.377143	1472.5800	1170.5100	1309.010952	96371290
	2018-02-28	1447.112632	1528.7000	1265.9300	1442.363158	137784020
	2018-03-31	1542.160476	1617.5400	1365.2000	1540.367619	130400151
	2018-04-30	1475.841905	1638.1000	1352.8800	1468.220476	129945743
	2018-05-31	1590.474545	1635.0000	1546.0200	1594.903636	71615299
	2018-06-30	1699.088571	1763.1000	1635.0900	1698.823810	85941510
	2018-07-31	1786.305714	1880.0500	1678.0600	1784.649048	97629820
	2018-08-31	1891.957826	2025.5700	1776.0200	1897.851304	96575676
	2018-09-30	1969.239474	2050.5000	1865.0000	1966.077895	94445693
	2018-10-31	1799.630870	2033.1900	1476.3600	1782.058261	183228552
	2018-11-30	1622.323810	1784.0000	1420.0000	1625.483810	139290208
	2018-12-31	1572.922105	1778.3400	1307.0000	1559.443158	154812304
FB	2018-01-31	184.364762	190.6600	175.8000	184.962857	495655736
	2018-02-28	180.721579	195.3200	167.1800	180.269474	516621991
	2018-03-31	173.449524	186.1000	149.0200	173.489524	996232472
	2018-04-30	164.163557	177.1000	150.5100	163.810476	751130388
	2018-05-31	181.910509	192.7200	170.2300	182.930000	401144183

ticker	date	open	high	low	close	volume
	2018-06-30	194.974067	203.5500	186.4300	195.267619	387265765
	2018-07-31	199.332143	218.6200	166.5600	199.967143	652763259
	2018-08-31	177.598443	188.3000	170.2700	177.491957	549016789
	2018-09-30	164.232895	173.8900	158.8656	164.377368	500468912
	2018-10-31	154.873261	165.8800	139.0300	154.187826	622446235
	2018-11-30	141.762857	154.1300	126.8500	141.635714	518150415
	2018-12-31	137.529474	147.1900	123.0200	137.161053	558786249
GOOG	**2018-01-31**	1127.200952	1186.8900	1045.2300	1130.770476	28738485
	2018-02-28	1088.629474	1174.0000	992.5600	1088.206842	42384105
	2018-03-31	1096.108095	1177.0500	980.6400	1091.490476	45430049
	2018-04-30	1038.415238	1094.1600	990.3700	1035.696190	41773275
	2018-05-31	1064.021364	1110.7500	1006.2900	1069.275909	31849196
	2018-06-30	1136.396190	1186.2900	1096.0100	1137.626667	32103642
	2018-07-31	1183.464286	1273.8900	1093.8000	1187.590476	31953386
	2018-08-31	1226.156957	1256.5000	1188.2400	1225.671739	28820379
	2018-09-30	1176.878421	1212.9900	1146.9100	1175.808947	28863199
	2018-10-31	1116.082174	1209.9600	995.8300	1110.940435	48496167
	2018-11-30	1054.971429	1095.5700	996.0200	1056.162381	36735570
	2018-12-31	1042.620000	1124.6500	970.1100	1037.420526	40256461
NFLX	**2018-01-31**	231.269286	286.8100	195.4200	232.908095	238377533
	2018-02-28	270.873158	297.3600	236.1100	271.443684	184585819
	2018-03-31	312.712857	333.9800	275.9000	312.228095	263449491
	2018-04-30	309.129529	338.8200	271.2239	307.466190	262064417
	2018-05-31	329.779759	356.1000	305.7300	331.536818	142051114
	2018-06-30	384.557595	423.2056	352.8200	384.133333	244032001
	2018-07-31	380.969090	419.7700	328.0000	381.515238	305487432
	2018-08-31	345.409591	376.8085	310.9280	346.257826	213144082
	2018-09-30	363.326842	383.2000	335.8300	362.641579	170832156
	2018-10-31	340.025348	386.7999	271.2093	335.445652	363589920

ticker	date	open	high	low	close	volume
	2018-11-30	290.643333	332.0499	250.0000	290.344762	257126498
	2018-12-31	266.309474	298.7200	231.2300	265.302368	234304628

In [98]:
```
# 4. Build a crosstab with the earthquake data between the tsunami column and the m
# show the maximum magnitude that was observed for each combination. Put the magTyp

earth.head() # checking data
```

Out[98]:

	mag	magType	time	place	tsunami	parsed_place
0	1.35	ml	1539475168010	9km NE of Aguanga, CA	0	California
1	1.29	ml	1539475129610	9km NE of Aguanga, CA	0	California
2	3.42	ml	1539475062610	8km NE of Aguanga, CA	0	California
3	0.44	ml	1539474978070	9km NE of Aguanga, CA	0	California
4	2.16	md	1539474716050	10km NW of Avenal, CA	0	California

In [112…
```
# Build a crosstab with the earthquake data between the tsunami column and the magT
# show the maximum magnitude that was observed for each combination. Put the magTyp

pd.crosstab(index = earth.tsunami, # set index used
            columns = earth.magType, # set column
            colnames = ['tsunami'], # set the column name
           values = earth.mag, # sets using the mag column values
           aggfunc = 'max')  # set type of aggregation
```

Out[112…

tsunami	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 [118…
```
# 5. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG
faang = pd.read_csv('faang.csv')

faang['date'] = pd.to_datetime(faang['date'])  # Convert 'date' column to datetime
faang = faang.set_index('date')                # Set it as index
faang.groupby(['ticker']).resample('60D').agg({ # using groupby to arrange data by
    'open': 'mean',           # pass the dict of columns that needs analysis and th
    'high': 'max',            # the dict allows us not to repat the aggreation meth
    'low': 'min',
    'close': 'mean',
})
```

Out[118...

ticker	date	open	high	low	close
AAPL	2018-01-02	168.042169	177.9059	147.9865	168.193874
	2018-03-03	169.577651	180.7477	158.2207	169.344005
	2018-05-02	185.066460	192.0247	171.1932	185.254314
	2018-07-01	198.659193	221.7617	181.3655	199.204829
	2018-08-30	220.584073	231.6645	210.6781	220.406912
	2018-10-29	181.979212	220.6405	145.9639	181.212129
	2018-12-28	157.340100	158.6794	153.8899	156.314500
AMZN	2018-01-02	1376.351429	1528.7000	1170.5100	1378.281667
	2018-03-03	1511.184634	1638.1000	1352.8800	1506.558780
	2018-05-02	1645.430476	1763.1000	1546.0200	1647.164762
	2018-07-01	1833.881667	1998.6900	1678.0600	1836.026667
	2018-08-30	1904.764146	2050.5000	1603.0000	1894.892683
	2018-10-29	1602.105366	1784.0000	1307.0000	1596.387805
	2018-12-28	1492.075000	1520.7600	1449.0000	1489.995000
FB	2018-01-02	182.325476	195.3200	167.1800	182.426190
	2018-03-03	168.526212	186.1000	149.0200	168.404878
	2018-05-02	188.678252	203.5500	172.1200	189.314762
	2018-07-01	188.516410	218.6200	166.5600	188.767976
	2018-08-30	160.783537	179.7901	143.8000	160.555122
	2018-10-29	140.578659	156.4000	123.0200	140.395610
	2018-12-28	134.895000	135.9200	129.9500	132.145000
GOOG	2018-01-02	1107.526905	1186.8900	992.5600	1108.822619
	2018-03-03	1065.309756	1177.0500	980.6400	1062.433902
	2018-05-02	1101.407857	1186.2900	1006.2900	1104.212381
	2018-07-01	1204.170238	1273.8900	1093.8000	1206.489048
	2018-08-30	1155.098780	1253.6300	1034.0900	1151.615854
	2018-10-29	1049.130244	1124.6500	970.1100	1047.579756
	2018-12-28	1050.290000	1055.5600	1023.5900	1036.345000
NFLX	2018-01-02	251.920119	301.1800	195.4200	253.331905

ticker	date	open	high	low	close
	2018-03-03	311.991466	338.8200	271.2239	310.620488
	2018-05-02	357.631052	423.2056	305.7300	358.269286
	2018-07-01	362.121702	419.7700	310.9280	362.787857
	2018-08-30	355.630561	386.7999	292.3000	352.967561
	2018-10-29	281.070488	332.0499	231.2300	280.162805
	2018-12-28	259.050000	270.1001	249.8000	261.870000

In [128…

```
# 6. Create a pivot table of the FAANG data that compares the stocks.
# Put the ticker in the rows and show the averages of the OHLC and volume traded da

pivotFAANG = pd.pivot_table(faang, values = ['open', 'high', 'low', 'close', 'volum
                            index = 'ticker', # setting the index
                            aggfunc = 'mean') # get the mean

pivotFAANG # show data
```

Out[128…

ticker	close	high	low	open	volume
AAPL	186.986218	188.906858	185.135729	187.038674	3.402145e+07
AMZN	1641.726175	1662.839801	1619.840398	1644.072669	5.649563e+06
FB	171.510936	173.615298	169.303110	171.454424	2.768798e+07
GOOG	1113.225139	1125.777649	1101.001594	1113.554104	1.742645e+06
NFLX	319.290299	325.224583	313.187273	319.620533	1.147030e+07

In [146…

```
# 7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NF

zscore = faang[faang['ticker'] == 'NFLX'][['open', 'high', 'low', 'close', 'volume'
'''This code first gets the rows with the ticker of 'NFLX'. Then dictate which colu
numerical columns in this dataset. using the apply and lambda function i used the z
each numeric columns'''

zcore # show data
```

Out[146…

date	open	high	low	close	volume
2018-01-02	-2.500753	-2.516023	-2.410226	-2.416644	-0.088760
2018-01-03	-2.380291	-2.423180	-2.285793	-2.335286	-0.507606
2018-01-04	-2.296272	-2.406077	-2.234616	-2.323429	-0.959287
2018-01-05	-2.275014	-2.345607	-2.202087	-2.234303	-0.782331
2018-01-08	-2.218934	-2.295113	-2.143759	-2.192192	-1.038531
…	…	…	…	…	…
2018-12-24	-1.571478	-1.518366	-1.627197	-1.745946	-0.339003
2018-12-26	-1.735063	-1.439978	-1.677339	-1.341402	0.517040
2018-12-27	-1.407286	-1.417785	-1.495805	-1.302664	0.134868
2018-12-28	-1.248762	-1.289018	-1.297285	-1.292137	-0.085164
2018-12-31	-1.203817	-1.122354	-1.088531	-1.055420	0.359444

251 rows × 5 columns

In [148…

```
# 8. Add event descriptions: - Create a dataframe with the following three columns:
'''- ticker, date, and event. The columns should have the following values: ticker:
- date: ['2018-07-25', '2018-03-19', '2018-03-20']
- event: ['Disappointing user growth announced after close.', 'Cambridge Analytica
- Set the index to ['date', 'ticker']
- Merge this data with the FAANG data using an outer join'''



datas = {
    'ticker': ['FB', 'FB', 'FB'],
    'date': ['2018-07-25', '2018-03-19', '2018-03-20'],
    'event': [
        'Disappointing user growth announced after close.',
        'Cambridge Analytica story',
        'FTC investigation'
    ]
}

# Create the dataframe
df = pd.DataFrame(datas)

# Set the index to ['date', 'ticker']
df['date'] = pd.to_datetime(df['date'])  # Ensure 'date' is in datetime format
df.set_index(['date', 'ticker'], inplace=True)
```

In [150…

```
faang2 = faang.copy()
```

In [152…
```
merged = pd.merge(faang2, df, on = 'ticker', how = 'outer')
```

In [154…
```
merged
```

Out[154…

	ticker	open	high	low	close	volume	event
0	AAPL	166.9271	169.0264	166.0442	168.9872	25555934	NaN
1	AAPL	169.2521	171.2337	168.6929	168.9578	29517899	NaN
2	AAPL	169.2619	170.1742	168.8106	169.7426	22434597	NaN
3	AAPL	170.1448	172.0381	169.7622	171.6751	23660018	NaN
4	AAPL	171.0375	172.2736	170.6255	171.0375	20567766	NaN
…	…	…	…	…	…	…	…
1752	NFLX	242.0000	250.6500	233.6800	233.8800	9547616	NaN
1753	NFLX	233.9200	254.5000	231.2300	253.6700	14402735	NaN
1754	NFLX	250.1100	255.5900	240.1000	255.5650	12235217	NaN
1755	NFLX	257.9400	261.9144	249.8000	256.0800	10987286	NaN
1756	NFLX	260.1600	270.1001	260.0000	267.6600	13508920	NaN

1757 rows × 7 columns

In [168…
```
'''9.  Use the transform() method on the FAANG data to represent all the values in
To do so, divide all the values for each ticker by the values for the first date in
This is referred to as an index, and the data for the first date is the base
(https:// ec. europa. eu/ eurostat/ statistics- explained/ index. php/ Beginners:St
When data is in this format, we can easily see growth over time. Hint: transform()

faang = pd.read_csv('faang.csv') # Load the data

faang = faang.sort_values(by = ['date', 'ticker'], ascending = True) # first sort t

faang.head() # checking of values
```

Out[168…

	ticker	date	open	high	low	close	volume
251	AAPL	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934
502	AMZN	2018-01-02	1172.0000	1190.0000	1170.5100	1189.0100	2694494
0	FB	2018-01-02	177.6800	181.5800	177.5500	181.4200	18151903
1004	GOOG	2018-01-02	1048.3400	1066.9400	1045.2300	1065.0000	1237564
753	NFLX	2018-01-02	196.1000	201.6500	195.4200	201.0700	10966889

In [182…
```
faang.tail() # checking of values
```

Out[182…

	ticker	date	open	high	low	close	volume
501	AAPL	2018-12-31	157.8529	158.6794	155.8117	157.0663	35003466
752	AMZN	2018-12-31	1510.8000	1520.7600	1487.0000	1501.9700	6954507
250	FB	2018-12-31	134.4500	134.6400	129.9500	131.0900	24625308
1254	GOOG	2018-12-31	1050.9600	1052.7000	1023.5900	1035.6100	1493722
1003	NFLX	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920

In [172…

```
getCols = ['open', 'high', 'low', 'close', 'volume'] # list all numeric columns to
transformFAANG = faang.copy() # create a copy

transformFAANG[getCols] = faang.groupby('ticker')[getCols].transform(lambda x: x /
'''In this portion first groupby ticker since it is the condtion we must work with
Then access the getCols and transform. the transform and lambda function allows us
then divide it by the first instance of the ticker we are accessing'''
```

In [176…

```
transformFAANG.head() # checking of values in the start
```

Out[176…

	ticker	date	open	high	low	close	volume
251	AAPL	2018-01-02	1.0	1.0	1.0	1.0	1.0
502	AMZN	2018-01-02	1.0	1.0	1.0	1.0	1.0
0	FB	2018-01-02	1.0	1.0	1.0	1.0	1.0
1004	GOOG	2018-01-02	1.0	1.0	1.0	1.0	1.0
753	NFLX	2018-01-02	1.0	1.0	1.0	1.0	1.0

In [180…

```
transformFAANG.tail() # checking of values in the end
# the values haved changed due to the division.
```

Out[180…

	ticker	date	open	high	low	close	volume
501	AAPL	2018-12-31	0.945640	0.938785	0.938375	0.929457	1.369681
752	AMZN	2018-12-31	1.289078	1.277950	1.270386	1.263211	2.581007
250	FB	2018-12-31	0.756697	0.741491	0.731907	0.722577	1.356624
1254	GOOG	2018-12-31	1.002499	0.986653	0.979296	0.972404	1.206986
1003	NFLX	2018-12-31	1.326670	1.339450	1.330468	1.331178	1.231791

In [ ]:

```
```

Conclusion

In doing this acctivity I was able to learn about the different ways you can aggregate the data in your dataset. I observe that getting the proper datatype is important because this can affect what aggregation you can do in the data. This is mostly applicable when you should be using a datetime data type for your index, since the different aggregation can only process this format of date. I also learned that bins were just ranges of values where you can categorize your entries to where they fall in the range. There are instances in this activity where the i do not understand clearly what to do or how to aggregate the data like number 6 and 9. I think I need work more on understanding the different functions and methods in pandas so that I would have an easier time identifying what to use for a given problem.

In [ ]: