Hands-on Activity 8.1 Aggregating Pandas DataFrames

8.1.5 Supplementary Activity

John Rome A. Belocora

```
import pandas as pd

eq = pd.read_csv('data/earthquakes.csv')
eq
```

	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
...	...	...	...	...	...	...
9327	0.62	md	1537230228060	9km ENE of Mammoth Lakes, CA	0	California
9328	1.00	ml	1537230135130	3km W of Julian, CA	0	California
9329	2.40	md	1537229908180	35km NNE of Hatillo, Puerto Rico	0	Puerto Rico
9330	1.10	ml	1537229545350	9km NE of Aguanga, CA	0	California
9331	0.66	ml	1537228864470	9km NE of Aguanga, CA	0	California

9332 rows × 6 columns

Next steps: View recommended plots

```
df = pd.DataFrame(eq) #Dataframe of earthquakes.csv
df.dtypes #checking the datatypes
```

```
mag             float64
magType          object
time              int64
place            object
tsunami           int64
parsed_place     object
dtype: object
```

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 .

```
# Selecting all the earthquakes in japan with a magType of mb and a magnitude of 4. 9 or greater
# using query()
with_mb = eq.query('magType == "mb" and mag >= 4.9 and parsed_place == "Japan"')

with_mb
```

	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

Next steps: View recommended plots

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 magT ype of ml and count how many are in each bin

```
import pandas as pd

# Sorting the Data frame of earthquakes with the magtype of "ml"
ml_earthquakes = pd.DataFrame(eq.query('magType == "ml"').sort_values('mag',ascending = False))

# Setting the values of bins from -2 to 6
bins = [-2,-0.1,1,1.99,2.99,3.99,4.99,6]

# Labels consists the values that will be using for the bins column
labels = ['>0','0-1','1-2','2-3','3-4','4-5','5-6']

# Creating another column "bins" using pd.cut to determine the values
ml_earthquakes['bins'] = pd.cut(ml_earthquakes.mag, bins = bins,
                                labels = labels)
ml_earthquakes
```

	mag	magType	time	place	tsunami	parsed_place	bins
9133	5.10	ml	1537274456960	64km SSW of Kaktovik, Alaska	1	Alaska	5-6
1015	5.00	ml	1539152878406	61km SSW of Chignik Lake, Alaska	1	Alaska	5-6
4101	4.20	ml	1538355504955	131km NNW of Arctic Village, Alaska	0	Alaska	4-5
1273	4.00	ml	1539069081499	71km SW of Kaktovik, Alaska	1	Alaska	4-5
2752	4.00	ml	1538658776412	67km SSW of Kaktovik, Alaska	1	Alaska	4-5
...	...	...	...	...	...	...	...
2428	-1.12	ml	1538741950500	42km ENE of Adak, Alaska	0	Alaska	>0
2405	-1.22	ml	1538747692790	43km ENE of	0	Alaska	>0

Next steps: View recommended plots

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

```
faang = pd.read_csv('data/faang.csv')
faang
```

	ticker	date	open	high	low	close	volume
0	FB	2018-01-02	177.68	181.58	177.5500	181.42	18151903
1	FB	2018-01-03	181.88	184.78	181.3300	184.67	16886563
2	FB	2018-01-04	184.90	186.21	184.0996	184.33	13880896
3	FB	2018-01-05	185.59	186.90	184.9300	186.85	13574535
4	FB	2018-01-08	187.20	188.90	186.3300	188.28	17994726
...	...	...	...	...	...	...	...
1250	GOOG	2018-12-24	973.90	1003.54	970.1100	976.22	1590328
1251	GOOG	2018-12-26	989.01	1040.00	983.0000	1039.46	2373270
1252	GOOG	2018-12-27	1017.15	1043.89	997.0000	1043.88	2109777
1253	GOOG	2018-12-28	1049.62	1055.56	1033.1000	1037.08	1413772
1254	GOOG	2018-12-31	1050.96	1052.70	1023.5900	1035.61	1493722

1255 rows × 7 columns

Next steps: View recommended plots

```
# Changing the datatype of date to datetime using "to_datetime"
faang['date'] = pd.to_datetime(faang['date'])

faang.set_index('date', inplace = True)
```

```
faang_group = faang.groupby('ticker').resample('M')
```

```
<pandas.core.resample.DatetimeIndexResamplerGroupby object at 0x79f047148100>
```

```
# Getting the mean, max, min, sum using the .agg()
faang_group = faang_group.agg({'open': 'mean',
 'high' : 'max',
 'low' : 'min',
 'close' : 'mean',
 'volume' : 'sum'})

faang_group
```

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
	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
	2018-11-30	290.643333	332.0499	250.0000	290.344762	257126498
	2018-12-31	266.309474	298.7200	231.2300	265.302368	234304628

Next steps: View recommended plots

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.

```
faang.head()
```

date	ticker	open	high	low	close	volume
2018-01-02	FB	177.68	181.58	177.5500	181.42	18151903
2018-01-03	FB	181.88	184.78	181.3300	184.67	16886563
2018-01-04	FB	184.90	186.21	184.0996	184.33	13880896
2018-01-05	FB	185.59	186.90	184.9300	186.85	13574535
2018-01-08	FB	187.20	188.90	186.3300	188.28	17994726

Next steps: View recommended plots

```
# Crosstab Data frame of "tsunami" and "magType" ang getting the maximum magnitude
# In the combination
pd.DataFrame(pd.crosstab(eq['tsunami'],eq['magType']).max()).T
```

magType	mb	mb_lg	md	mh	ml	ms_20	mw	mwb	mwr	mww
0	574	30	1796	12	6798	1	2	2	14	42

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

```
faang_group.dtypes #Datatype
```

```
open      float64
high      float64
low       float64
close     float64
volume      int64
dtype: object
```

```
# 60 day aggregation of the faang data by using freq='2M'
# 2M (2 Months)
faang60 = faang.groupby([pd.Grouper(freq='2M'),'ticker'])

# Using the same aggregations in exercise no. 3
faang60 = faang60.agg({
 'open' : 'mean',
 'high' : 'max',
 'low' : 'min',
 'close' : 'mean',
 'volume' : 'sum'
})

faang60
```

date	ticker	open	high	low	close	volume
2018-01-31	AAPL	170.714690	176.6782	161.5708	170.699271	659679440
	AMZN	1301.377143	1472.5800	1170.5100	1309.010952	96371290
	FB	184.364762	190.6600	175.8000	184.962857	495655736
	GOOG	1127.200952	1186.8900	1045.2300	1130.770476	28738485
	NFLX	231.269286	286.8100	195.4200	232.908095	238377533
2018-03-31	AAPL	168.688533	180.7477	147.9865	168.574328	1641621920
	AMZN	1497.012750	1617.5400	1265.9300	1493.815500	268184171
	FB	176.903750	195.3200	149.0200	176.710000	1512854463
	GOOG	1092.555750	1177.0500	980.6400	1089.930750	87814154
	NFLX	292.839000	333.9800	236.1100	292.855500	448035310
2018-05-31	AAPL	175.162177	187.9311	158.2207	175.432293	1287336353
	AMZN	1534.491163	1638.1000	1352.8800	1533.035116	201561042
	FB	173.243393	192.7200	150.5100	173.592558	1152274571
	GOOG	1051.516047	1110.7500	990.3700	1052.876512	73622471
	NFLX	319.694763	356.1000	271.2239	319.781395	404115531
2018-07-31	AAPL	187.335814	193.7650	178.7056	187.344188	921468246
	AMZN	1742.697143	1880.0500	1635.0900	1741.736429	183571330
	FB	197.153105	218.6200	166.5600	197.617381	1040029024
	GOOG	1159.930238	1273.8900	1093.8000	1162.608571	64057028
	NFLX	382.763343	423.2056	328.0000	382.824286	549519433
2018-09-30	AAPL	215.052612	227.8939	195.0999	215.494257	1379290877
	AMZN	1926.918571	2050.5000	1776.0200	1928.715714	191021369
	FB	171.552124	188.3000	158.8656	171.559167	1049485701
	GOOG	1203.864286	1256.5000	1146.9100	1203.114762	57683578
	NFLX	353.515014	383.2000	310.9280	353.669524	383976238
2018-11-30	AAPL	205.810434	231.6645	169.5328	205.348855	1751070015
	AMZN	1715.007045	2033.1900	1420.0000	1707.329545	322518760
	FB	148.616023	165.8800	126.8500	148.197045	1140596650
	GOOG	1086.915682	1209.9600	995.8300	1084.796364	85231737
	NFLX	316.456659	386.7999	250.0000	313.920227	620716418
2019-01-31	AAPL	164.537405	184.1501	145.9639	163.564732	898917007
	AMZN	1572.922105	1778.3400	1307.0000	1559.443158	154812304
	FB	137.529474	147.1900	123.0200	137.161053	558786249
	GOOG	1042.620000	1124.6500	970.1100	1037.420526	40256461
	NFLX	266.309474	298.7200	231.2300	265.302368	234304628

Next steps: View recommended plots

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.

```
columns = faang.columns[:-1]

# Pivot table that compares every stocks in the faang
faang_pivot = pd.pivot_table(faang,index = 'ticker', values = columns, aggfunc = 'mean')
faang_pivot
```

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

Next steps: View recommended plots

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

```
# Query data of the data stock "NFLX"
faang_NFLX = faang_group.query('ticker == "NFLX"')
faang_NFLX
```

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

Next steps: View recommended plots

```
import scipy.stats as stats

# Getting the Zscore stats using the scipy.stats library
zscoresNFLX = faang_NFLX.apply(lambda column: stats.zscore(column), axis = 0)
zscoresNFLX
```

ticker	date	open	high	low	close	volume
NFLX	2018-01-31	-1.913251	-1.493069	-1.865932	-1.882154	-0.027024
	2018-02-28	-1.047098	-1.254375	-0.972140	-1.034121	-0.969211
	2018-03-31	-0.132046	-0.425848	-0.098117	-0.136599	0.412123
	2018-04-30	-0.210414	-0.316343	-0.200831	-0.241392	0.387862
	2018-05-31	0.241215	0.074617	0.557126	0.288318	-1.714227
	2018-06-30	1.439229	1.592882	1.591500	1.445783	0.072016
	2018-07-31	1.360747	1.515151	1.046307	1.388168	1.148437
	2018-08-31	0.583046	0.543147	0.671305	0.612276	-0.469000
	2018-09-30	0.974904	0.687755	1.218300	0.972825	-1.210113
	2018-10-31	0.465290	0.769203	-0.201152	0.374338	2.166129
	2018-11-30	-0.614715	-0.469516	-0.667033	-0.618174	0.301372
	2018-12-31	-1.146907	-1.223605	-1.079333	-1.169270	-0.098363

Next steps: View recommended plots

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

```
# Creating a data frame with the 3 values given
Faang_desc = {'ticker':'FB',
 'date':['2018-07-25', '2018-03-19', '2018-03-20'],
 'event':['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC investigation']}
value = pd.DataFrame(Faang_desc)
value
```

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

Next steps: View recommended plots

```
value.dtypes
```

```
ticker    object
date      object
event     object
dtype: object
```

```
# Changing the data type of date to "datetime"
value['date'] = pd.to_datetime(value['date'])
```

```
value.dtypes
```

```
ticker            object
date      datetime64[ns]
event             object
dtype: object
```

```
# Setting the ticker to index
value.set_index(['ticker','date'])
```

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

```
column = list(value.columns[:-1])
```

```
# Using the "pd.merge" we can combine the values
outer_Join = pd.merge(value, faang , on = column, how = 'outer')
outer_Join.head()
```

	ticker	date	event	open	high	low	close	volume
0	FB	2018-07-25	Disappointing user growth announced after close.	215.715	218.62	214.27	217.50	64592585
1	FB	2018-03-19	Cambridge Analytica story	177.010	177.17	170.06	172.56	88140060