Name: Xander Sam E. Galapia

Section: CPE22S3

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 - 8.1

8.2 Querying and Merging - 8.2

8.3 Dataframe Operations - 8.3

8.4 Aggregations - 8.4

8.5 Time Series - 8.5

8.1.4 Data Analysis

Provide some comments here about the results of the procedures.

8.1.5 Supplementary Activity

Using the CSV files provided and what we have learned so far in this module complete the following exercises:

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:Statisticalconcept-Indexandbaseyear). When data is in this format, we can easily see growth over time. Hint: transform() can take a function name.

```
import pandas as pd
import numpy as np
eq = pd.read_csv('data/earthquakes.csv')
faang = pd.read_csv('data/faang.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

```
eq.rename(columns = {'parsed_place': 'Country'}, inplace = True) ##Changing parsed_place to Country
eq
```

	mag	magType	time	place	tsunami	Country
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

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.

```
jap = eq.query('Country == "Japan" and mag >= 4.9 and magType == "mb"') #Specifying for Japan, mag of greater than 4.9 that belongs to mb
jap
```

	mag	magType	time	place	tsunami	Country
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

```
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

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.

```
ml_eq = pd.DataFrame(eq.query('magType == "ml"').sort_values('mag',ascending = False)) #Getting descending datas that belong to ml in magType
ml_eq
```

	mag	magType	time	place	tsunami	Country
9133	5.10	ml	1537274456960	64km SSW of Kaktovik, Alaska	1	Alaska
1015	5.00	ml	1539152878406	61km SSW of Chignik Lake, Alaska	1	Alaska
4101	4.20	ml	1538355504955	131km NNW of Arctic Village, Alaska	0	Alaska
1273	4.00	ml	1539069081499	71km SW of Kaktovik, Alaska	1	Alaska
2752	4.00	ml	1538658776412	67km SSW of Kaktovik, Alaska	1	Alaska
...	...	...	...	...	...	...
2428	-1.12	ml	1538741950500	42km ENE of Adak, Alaska	0	Alaska
2405	-1.22	ml	1538747692790	43km ENE of Adak, Alaska	0	Alaska
6244	-1.24	ml	1537934601100	42km ENE of Adak, Alaska	0	Alaska
2409	-1.26	ml	1538746911930	41km ENE of Adak, Alaska	0	Alaska
6767	-1.26	ml	1537846638890	17km W of Akutan, Alaska	0	Alaska

6803 rows × 6 columns

Next steps: View recommended plots

```
bins = [-2,-0.1,1,1.99,2.99,3.99,4.99,6]
labels = ['>0','0-1','1-2','2-3','3-4','4-5','5-6']
ml_eq['bins'] = pd.cut(ml_eq.mag, bins = bins,
                       labels = labels)
ml_eq
```

	mag	magType	time	place	tsunami	Country	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 Adak, Alaska	0	Alaska	>0
6244	-1.24	ml	1537934601100	42km ENE of Adak, Alaska	0	Alaska	>0
2409	-1.26	ml	1538746911930	41km ENE of Adak, Alaska	0	Alaska	>0
6767	-1.26	ml	1537846638890	17km W of Akutan, Alaska	0	Alaska	>0

6803 rows × 7 columns

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['date']=pd.to_datetime(faang['date'])
faang.set_index('date',inplace=True)
```

```
#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 trade

#Grouping the faang data first by it's ticker and then monthly
MMfaang = faang.groupby('ticker').resample('M')

#Aggregating the arranged datas from MMFaang
faang_agg = MMfaang.agg({
'open': 'mean',
'high': 'max',
'low': 'min',
'close': 'mean',
'volume' : 'sum'

})
faang_agg
```

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	1038.413238	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

Next

View recommended plots

2018-12-31	266.309474	298.7200	231.2300	265.302368	234304628

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.

```
ct = pd.DataFrame(pd.crosstab(eq['tsunami'],eq['magType']).max()).T
ct
```

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.

```
r60d = faang.groupby([pd.Grouper(freq='60D'),'ticker'])
```

```
r60d.agg({'open':'mean',
          'high':'max',
          'low':'min',
          'close':'mean',
          'volume':'sum'})
```

date	ticker	open	high	low	close	volume
2018-01-02	AAPL	168.042169	177.9059	147.9865	168.193874	1674829833
	AMZN	1376.351429	1528.7000	1170.5100	1378.281667	247578104
	FB	182.325476	195.3200	167.1800	182.426190	1055505258
	GOOG	1107.526905	1186.8900	992.5600	1108.822619	75910051
	NFLX	251.920119	301.1800	195.4200	253.331905	448240716
2018-03-03	AAPL	169.577651	180.7477	158.2207	169.344005	1346401050
	AMZN	1511.184634	1638.1000	1352.8800	1506.558780	251495171
	FB	168.526212	186.1000	149.0200	168.404878	1730161261
	GOOG	1065.309756	1177.0500	980.6400	1062.433902	83843720
	NFLX	311.991466	338.8200	271.2239	310.620488	506273183
2018-05-02	AAPL	185.066460	192.0247	171.1932	185.254314	1095031195
	AMZN	1645.430476	1763.1000	1546.0200	1647.164762	152984738
	FB	188.678252	203.5500	172.1200	189.314762	762384016
	GOOG	1101.407857	1186.2900	1006.2900	1104.212381	62524981
	NFLX	357.631052	423.2056	305.7300	358.269286	380046476
2018-07-01	AAPL	198.659193	221.7617	181.3655	199.204829	1002028760
	AMZN	1833.881667	1998.6900	1678.0600	1836.026667	182723861
	FB	188.516410	218.6200	166.5600	188.767976	1159498357
	GOOG	1204.170238	1273.8900	1093.8000	1206.489048	57625900
	NFLX	362.121702	419.7700	310.9280	362.787857	499707058
2018-08-30	AAPL	220.584073	231.6645	210.6781	220.406912	1439899623
	AMZN	1904.764146	2050.5000	1603.0000	1894.892683	253439506
	FB	160.783537	179.7901	143.8000	160.555122	1023230525
	GOOG	1155.098780	1253.6300	1034.0900	1151.615854	70884083
	NFLX	355.630561	386.7999	292.3000	352.967561	487601647
2018-10-29	AAPL	181.979212	220.6405	145.9639	181.212129	1903898507
	AMZN	1602.105366	1784.0000	1307.0000	1596.387805	314035429
	FB	140.578659	156.4000	123.0200	140.395610	1171650100

	GOOG	1049.130244	1124.6500	970.1100	1047.579756	83707685
	NFLX	281.070488	332.0499	231.2300	280.162805	532679805
2018-12-28	AAPL	157.340100	158.6794	153.8899	156.314500	77294890
	AMZN	1492.075000	1520.7600	1449.0000	1489.995000	15783457
	FB	134.895000	135.9200	129.9500	132.145000	47252877
	GOOG	1050.290000	1055.5600	1023.5900	1036.345000	2907494
	NFLX	259.050000	270.1001	249.8000	261.870000	24496206

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]
faangpiv = pd.pivot_table(faang,index = 'ticker', values = columns, aggfunc = 'mean')
faangpiv
```

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

```
fagg_nflx = faang_agg.query('ticker == "NFLX"')
fagg_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

```
def calculate_zscores(data):
  return(data - data.mean()) / data.std()
ZNF = fagg_nflx.apply(calculate_zscores)
ZNF
```

ticker	date	open	high	low	close	volume