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

8.1.5 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.

In [167…
```
import pandas as pd
```

In [240…
```
earthquake = pd.read_csv('./data/earthquakes.csv')
```

In [241…
```
earthquake
```

Out[241…

	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

In [170…

```
select_earthquake = ((earthquake['magType'] == 'mb') & (earthquake['mag'] >= 4.9) )
```

In [171…

```
filtered_df = earthquake[select_earthquake]
filtered_df
```

Out[171…

	mag	magType	time	place	tsunami	parsed_place
227	5.2	mb	1539389603790	15km WSW of Pisco, Peru	0	Peru
229	4.9	mb	1539389546300	193km N of Qulansiyah, Yemen	0	Yemen
248	4.9	mb	1539382925190	151km S of Severo-Kuril'sk, Russia	0	Russia
258	5.1	mb	1539380306940	236km NNW of Kuril'sk, Russia	0	Russia
391	5.1	mb	1539337221080	Pacific-Antarctic Ridge	0	Pacific-Antarctic Ridge
…	…	…	…	…	…	…
9154	4.9	mb	1537268270010	Southwest Indian Ridge	0	Southwest Indian Ridge
9175	5.2	mb	1537262729590	126km N of Dili, East Timor	1	East Timor
9176	5.2	mb	1537262656830	90km S of Raoul Island, New Zealand	0	New Zealand
9213	5.1	mb	1537255481060	South of Tonga	0	Tonga
9304	5.1	mb	1537236235470	34km NW of Finschhafen, Papua New Guinea	1	Papua New Guinea

122 rows × 6 columns

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.

In [ ]:
```
ml_quakes = earthquake[earthquake['magType'] == 'ml'] # select the column where magType value
```

In [172…
```
# to store the count
bin_counts = {
    '0-1': 0,
    '1-2': 0,
    '2-3': 0,
    '3-4': 0,
    '4-5': 0,
    '5-6': 0
}

# loop to add the count on each dictionary when the condition on mag is met
for mag in ml_quakes['mag']:
    if mag >= 0 and mag < 1:
        bin_counts['0-1'] += 1
    elif mag >= 1 and mag < 2:
        bin_counts['1-2'] += 1
    elif mag >= 2 and mag < 3:
        bin_counts['2-3'] += 1
    elif mag >= 3 and mag < 4:
        bin_counts['3-4'] += 1
    elif mag >= 4 and mag < 5:
        bin_counts['4-5'] += 1
    elif mag >= 5 and mag < 6:
        bin_counts['5-6'] += 1

for bin_range, count in bin_counts.items():
    print(f"{bin_range}: {count}")
```

```
0-1: 2072
1-2: 3126
2-3: 985
3-4: 153
4-5: 6
5-6: 2
```

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

In [185…
```
import pandas as pd
```

In [186…
```
faang = pd.read_csv('./data/faang.csv')
```

In [187…
```
faang
```

Out[187…

	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

In [188…
```
faang['date'] = pd.to_datetime(faang['date']) # to convert date column into datetime
```

In [189…
```
# Set 'date' as index
faang.set_index('date', inplace=True) # to use the .resample(), it reuqires to set teh date as
```

In [190…
```
# Group by 'ticker' and resample monthly
faang_groupby = faang.groupby('ticker').resample('ME').agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})
```

In [192…
```
faang_groupby= faang_groupby.reset_index()
```

In [193…
```
faang_groupby
```

Out[193...

	ticker	date	open	high	low	close	volume
0	AAPL	2018-01-31	170.714690	176.6782	161.5708	170.699271	659679440
1	AAPL	2018-02-28	164.562753	177.9059	147.9865	164.921884	927894473
2	AAPL	2018-03-31	172.421381	180.7477	162.4660	171.878919	713727447
3	AAPL	2018-04-30	167.332895	176.2526	158.2207	167.286924	666360147
4	AAPL	2018-05-31	182.635582	187.9311	162.7911	183.207418	620976206
5	AAPL	2018-06-30	186.605843	192.0247	178.7056	186.508652	527624365
6	AAPL	2018-07-31	188.065786	193.7650	181.3655	188.179724	393843881
7	AAPL	2018-08-31	210.460287	227.1001	195.0999	211.477743	700318837
8	AAPL	2018-09-30	220.611742	227.8939	213.6351	220.356353	678972040
9	AAPL	2018-10-31	219.489426	231.6645	204.4963	219.137822	789748068
10	AAPL	2018-11-30	190.828681	220.6405	169.5328	190.246652	961321947
11	AAPL	2018-12-31	164.537405	184.1501	145.9639	163.564732	898917007
12	AMZN	2018-01-31	1301.377143	1472.5800	1170.5100	1309.010952	96371290
13	AMZN	2018-02-28	1447.112632	1528.7000	1265.9300	1442.363158	137784020
14	AMZN	2018-03-31	1542.160476	1617.5400	1365.2000	1540.367619	130400151
15	AMZN	2018-04-30	1475.841905	1638.1000	1352.8800	1468.220476	129945743
16	AMZN	2018-05-31	1590.474545	1635.0000	1546.0200	1594.903636	71615299
17	AMZN	2018-06-30	1699.088571	1763.1000	1635.0900	1698.823810	85941510
18	AMZN	2018-07-31	1786.305714	1880.0500	1678.0600	1784.649048	97629820
19	AMZN	2018-08-31	1891.957826	2025.5700	1776.0200	1897.851304	96575676
20	AMZN	2018-09-30	1969.239474	2050.5000	1865.0000	1966.077895	94445693
21	AMZN	2018-10-31	1799.630870	2033.1900	1476.3600	1782.058261	183228552
22	AMZN	2018-11-30	1622.323810	1784.0000	1420.0000	1625.483810	139290208
23	AMZN	2018-12-31	1572.922105	1778.3400	1307.0000	1559.443158	154812304
24	FB	2018-01-31	184.364762	190.6600	175.8000	184.962857	495655736
25	FB	2018-02-28	180.721579	195.3200	167.1800	180.269474	516621991
26	FB	2018-03-31	173.449524	186.1000	149.0200	173.489524	996232472
27	FB	2018-04-30	164.163557	177.1000	150.5100	163.810476	751130388
28	FB	2018-05-31	181.910509	192.7200	170.2300	182.930000	401144183
29	FB	2018-06-30	194.974067	203.5500	186.4300	195.267619	387265765
30	FB	2018-07-31	199.332143	218.6200	166.5600	199.967143	652763259
31	FB	2018-08-31	177.598443	188.3000	170.2700	177.491957	549016789
32	FB	2018-09-30	164.232895	173.8900	158.8656	164.377368	500468912
33	FB	2018-10-31	154.873261	165.8800	139.0300	154.187826	622446235

	ticker	date	open	high	low	close	volume
34	FB	2018-11-30	141.762857	154.1300	126.8500	141.635714	518150415
35	FB	2018-12-31	137.529474	147.1900	123.0200	137.161053	558786249
36	GOOG	2018-01-31	1127.200952	1186.8900	1045.2300	1130.770476	28738485
37	GOOG	2018-02-28	1088.629474	1174.0000	992.5600	1088.206842	42384105
38	GOOG	2018-03-31	1096.108095	1177.0500	980.6400	1091.490476	45430049
39	GOOG	2018-04-30	1038.415238	1094.1600	990.3700	1035.696190	41773275
40	GOOG	2018-05-31	1064.021364	1110.7500	1006.2900	1069.275909	31849196
41	GOOG	2018-06-30	1136.396190	1186.2900	1096.0100	1137.626667	32103642
42	GOOG	2018-07-31	1183.464286	1273.8900	1093.8000	1187.590476	31953386
43	GOOG	2018-08-31	1226.156957	1256.5000	1188.2400	1225.671739	28820379
44	GOOG	2018-09-30	1176.878421	1212.9900	1146.9100	1175.808947	28863199
45	GOOG	2018-10-31	1116.082174	1209.9600	995.8300	1110.940435	48496167
46	GOOG	2018-11-30	1054.971429	1095.5700	996.0200	1056.162381	36735570
47	GOOG	2018-12-31	1042.620000	1124.6500	970.1100	1037.420526	40256461
48	NFLX	2018-01-31	231.269286	286.8100	195.4200	232.908095	238377533
49	NFLX	2018-02-28	270.873158	297.3600	236.1100	271.443684	184585819
50	NFLX	2018-03-31	312.712857	333.9800	275.9000	312.228095	263449491
51	NFLX	2018-04-30	309.129529	338.8200	271.2239	307.466190	262064417
52	NFLX	2018-05-31	329.779759	356.1000	305.7300	331.536818	142051114
53	NFLX	2018-06-30	384.557595	423.2056	352.8200	384.133333	244032001
54	NFLX	2018-07-31	380.969090	419.7700	328.0000	381.515238	305487432
55	NFLX	2018-08-31	345.409591	376.8085	310.9280	346.257826	213144082
56	NFLX	2018-09-30	363.326842	383.2000	335.8300	362.641579	170832156
57	NFLX	2018-10-31	340.025348	386.7999	271.2093	335.445652	363589920
58	NFLX	2018-11-30	290.643333	332.0499	250.0000	290.344762	257126498
59	NFLX	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

In [194…

```
earthquake.head(5) # to check dataframe only
```

Out[194...

	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 [195...

```
earthquake_crosstab = pd.crosstab(
    index=earthquake['tsunami'],
    columns=earthquake['magType'],
    values=earthquake['mag'],
    aggfunc='max'
)
earthquake_crosstab
```

Out[195...

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

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

In [196...

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

In [197...

```
faang_rolling = faang.sort_values(['ticker', 'date'])
faang_rolling.set_index('date', inplace=True)
```

In [198...

```
# rolling
rolling_60d = faang_rolling.groupby('ticker').rolling(window=60).agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})
```

In [199...

```
faang
```

Out[199…

	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

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.

In [122…

```
# pivoting the table then get the mean of it by using the aggFunction
faang_pivot = pd.pivot_table(
    faang,
    index='ticker',
    values=['open', 'high', 'low', 'close', 'volume'],
    aggfunc='mean'
)
```

In [123…

```
faang_pivot
```

Out[123…

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

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

In [200…

```
nflx = faang[faang['ticker'] == 'NFLX'] # select column with nflx ticker
```

In [201…

```
numeric_column = ['open', 'high', 'low', 'close', 'volume'] # to just select the header
```

In [202…

```
nflx_zscores = nflx[numeric_column].apply(lambda x: (x - x.mean()) / x.std())
```

In [203…

```
nflx_zscores
```

Out[203…

	open	high	low	close	volume
753	-2.500753	-2.516023	-2.410226	-2.416644	-0.088760
754	-2.380291	-2.423180	-2.285793	-2.335286	-0.507606
755	-2.296272	-2.406077	-2.234616	-2.323429	-0.959287
756	-2.275014	-2.345607	-2.202087	-2.234303	-0.782331
757	-2.218934	-2.295113	-2.143759	-2.192192	-1.038531
…	…	…	…	…	…
999	-1.571478	-1.518366	-1.627197	-1.745946	-0.339003
1000	-1.735063	-1.439978	-1.677339	-1.341402	0.517040
1001	-1.407286	-1.417785	-1.495805	-1.302664	0.134868
1002	-1.248762	-1.289018	-1.297285	-1.292137	-0.085164
1003	-1.203817	-1.122354	-1.088531	-1.055420	0.359444

251 rows × 5 columns

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

In [220…
```
faang = pd.read_csv('./data/faang.csv')
```

In [221…
```
# to create a data frame
event_df = pd.DataFrame({
    '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'
    ]
})
```

In [222…
```
event_df['date'] = pd.to_datetime(event_df['date']) # to convert date column to datetime
```

In [223…
```
event_df.set_index(['date', 'ticker'], inplace=True) # to set index as date and ticker
```

In [224…
```
faang['date'] = pd.to_datetime(faang['date'])  # to convert date column of faang to datetime
```

In [225…
```
faang.set_index(['date', 'ticker'], inplace=True) #  set index to this and ensure that it's ma
```

In [226…
```
merged_df = faang.merge(event_df, how='outer', left_index=True, right_index=True)
```

In [229…
```
merged_df
```

Out[229…

		open	high	low	close	volume	event
date	**ticker**						
2018-01-02	**AAPL**	166.9271	169.0264	166.0442	168.9872	25555934	NaN
	AMZN	1172.0000	1190.0000	1170.5100	1189.0100	2694494	NaN
	FB	177.6800	181.5800	177.5500	181.4200	18151903	NaN
	GOOG	1048.3400	1066.9400	1045.2300	1065.0000	1237564	NaN
	NFLX	196.1000	201.6500	195.4200	201.0700	10966889	NaN
...	**...**	...	...	...	...	...	...
2018-12-31	**AAPL**	157.8529	158.6794	155.8117	157.0663	35003466	NaN
	AMZN	1510.8000	1520.7600	1487.0000	1501.9700	6954507	NaN
	FB	134.4500	134.6400	129.9500	131.0900	24625308	NaN
	GOOG	1050.9600	1052.7000	1023.5900	1035.6100	1493722	NaN
	NFLX	260.1600	270.1001	260.0000	267.6600	13508920	NaN

1255 rows × 6 columns

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 [235…
```
faang = faang.sort_values(['ticker', 'date']) # Make sure the data is sorted by ticker and dat
```

In [236…
```
numeric_cols = ['open', 'high', 'low', 'close', 'volume'] # Select the numeric columns
```

In [237…
```
# to transform each value to the first one per ticker
faang_relative = faang.copy()
faang_relative[numeric_cols] = faang.groupby('ticker')[numeric_cols].transform(
    lambda x: x / x.iloc[0]
)
```

In [238…
```
faang_relative
```

Out[238...

		open	high	low	close	volume
date	**ticker**					
2018-01-02	**AAPL**	1.000000	1.000000	1.000000	1.000000	1.000000
2018-01-03	**AAPL**	1.013928	1.013059	1.015952	0.999826	1.155031
2018-01-04	**AAPL**	1.013987	1.006791	1.016661	1.004470	0.877863
2018-01-05	**AAPL**	1.019276	1.017818	1.022392	1.015906	0.925813
2018-01-08	**AAPL**	1.024624	1.019211	1.027591	1.012133	0.804814
...	**...**	...	...	...	...	...
2018-12-24	**NFLX**	1.234064	1.242995	1.195783	1.163177	0.870586
2018-12-26	**NFLX**	1.192861	1.262088	1.183246	1.261600	1.313293
2018-12-27	**NFLX**	1.275421	1.267493	1.228636	1.271025	1.115651
2018-12-28	**NFLX**	1.315349	1.298856	1.278272	1.273586	1.001860
2018-12-31	**NFLX**	1.326670	1.339450	1.330468	1.331178	1.231791

1255 rows × 5 columns

Conclusion

This activity is a bit long but it made me practice the syntax of pandas. It also forces me to use function where I usually doesn't use because of its compexity like the transform and pivot of table. I have also learned another way to aggregate data with aggfunc, it makes the aggregating dat easier as it can be use in more than one varaiable. Before, I aggregate data, column by column. Overall, this activity is challenging in a good way.