Hands-on Activity 8.1: Aggregating Data with Pandas

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 Githublink for 8.1
- 8.2 Querying and Merging Githublink for 8.2
- 8.3 Dataframe Operations Githublink for 8.3
- 8.4 Aggregations Githublink for 8.4
- 8.5 Time Series Githublink for 8.5

8.1.4 Data Analysis

Provide some comments here about the results of the procedures.

- Storing Data in SQLite:

I learned how to store a Pandas DataFrame in an SQLite database. It felt really empowering because now I know how to save data in a way that's easy to retrieve and manage later.

- Data Manipulation Language (DML):

I practiced using DML to update and organize data within the database. It was a little tricky at first, but I got the hang of it. It's definitely going to help when I need to adjust or clean up data for analysis.

- Aggregating Data:

I learned was how to use Pandas' aggregate functions to calculate things like the mean, minimum, and maximum values all at once.

- Grouping Data:

I also got to work with the groupby function, which groups data by a category and lets me apply different calculations to each group. For example, if I have weather data from different stations, I can group the data by station and easily get the average temperature or total rainfall.

- Handling Time-Based Data:

The real challenge came when I had to work with data that included dates and times (not just dates). I used functions like between_time and at_time to pull out data for specific hours or time ranges. This was a big step up because it let me work with hourly weather data instead of just daily data.

- Storing, Manipulating, and Analyzing Data:

Overall, I feel so much more confident in my ability to store, manipulate, and analyze data now. Learning how to work with time-based data, in particular, has been a game-changer.

- Challenges with Newer Pandas Versions:

But, of course, it wasn't all smooth sailing. Some of the code I was using before didn't work with the latest version of Pandas. I got a warning about deprecated functions like last() and first(), which meant I had to figure out how to replace those with newer methods. It was a bit frustrating, but I eventually found the alternatives and it was a good reminder to stay updated with library changes!

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 value

In [2]:
```
import pandas as pd

eq = pd.read_csv('earthquakes.csv')
faang = pd.read_csv('faang.csv')
```

In [3]:
```
eq.head(3)
```

Out[3]:

	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

In [4]:
```
eq.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9332 entries, 0 to 9331
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mag           9331 non-null   float64
 1   magType       9331 non-null   object 
 2   time          9332 non-null   int64  
 3   place         9332 non-null   object 
 4   tsunami       9332 non-null   int64  
 5   parsed_place  9332 non-null   object 
dtypes: float64(1), int64(2), object(3)
memory usage: 437.6+ KB
```

In [5]:
```
#I changed the names of the columns for a more appropriate analyis
eq.rename(columns={
    'mag': 'Magnitude',
    'parsed_place': 'Location',
    'magType': 'MagnitudeType',
    'time' :'Time',
    'tsunami' : 'Tsunami' ,
    'place': 'PlaceDescription'
}, inplace=True)
```

In [6]:
```
eq.head(3)
```

Out[6]:

	Magnitude	MagnitudeType	Time	PlaceDescription	Tsunami	Location
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

In [7]:
```
# 1. With the earthquakes.csv file, select all the earthquakes in Japan with a magT
filtered_eq = eq.query("Location == 'Japan' and MagnitudeType == 'mb' and Magnitude
filtered_eq
```

Out[7]:

	Magnitude	MagnitudeType	Time	PlaceDescription	Tsunami	Location
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 [25]:
```
# Create bins for each full number of magnitude (for example, the first bin is 0-1,
ml = eq[eq['MagnitudeType'] == 'ml'] # I filter the magnitude type first then
bins = [i for i in range(0, int(ml['Magnitude'].max()) + 4)] # I made the bin
counts = pd.cut(ml['Magnitude'], bins=bins, right=False).value_counts().sort_index(

ml = pd.DataFrame({'Magnitude (ml)':bins[:-1], 'occurrences': counts})
ml
```

Out[25]:

Magnitude	Magnitude (ml)	occurrences
[0, 1)	0	2072
[1, 2)	1	3126
[2, 3)	2	985
[3, 4)	3	153
[4, 5)	4	6
[5, 6)	5	2
[6, 7)	6	0
[7, 8)	7	0

In [9]:
```
faang.head()
```

Out[9]:

	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

In [10]:
```
# 3. Using the faang.csv file, group by the ticker and resample to monthly frequenc
# 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 [11]:
```
faang.info() #I first check the date if it is in the right datatypes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1255 entries, 0 to 1254
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ticker  1255 non-null   object 
 1   date    1255 non-null   object 
 2   open    1255 non-null   float64
 3   high    1255 non-null   float64
 4   low     1255 non-null   float64
 5   close   1255 non-null   float64
 6   volume  1255 non-null   int64  
dtypes: float64(4), int64(1), object(2)
memory usage: 68.8+ KB
```

In [12]:
```
faang['date'] = pd.to_datetime(faang['date']) #convert into datetime
faang.set_index('date',inplace = True) #then set it as index
```

In [13]:
```
agg = faang.groupby('ticker').resample('ME').agg({
    'open' : 'mean',
    'high' : 'max' ,
    'low'  : 'min',
    'close' : 'mean',
    'volume' : 'sum'
})
agg
```

Out[13]:

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 [14]:
```
# 4. Build a crosstab with the earthquake data between the tsunami column and the m
# magnitude that was observed for each combination. Put the magType along the colum
eq['Tsunami'].value_counts()
```

Out[14]:
```
Tsunami
0    9271
1      61
Name: count, dtype: int64
```

In [15]:
```
crosstab_max_mag = pd.crosstab(eq['Tsunami'], eq['MagnitudeType'],values=eq['Magnit
crosstab_max_mag
```

Out[15]:

MagnitudeType	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 [16]:
```
# 5. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG
agg = faang.groupby('ticker').rolling('60D').agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})
agg
```

Out[16]:

ticker	date	open	high	low	close	volume
AAPL	2018-01-02	166.927100	169.0264	166.0442	168.987200	25555934.0
	2018-01-03	168.089600	171.2337	166.0442	168.972500	55073833.0
	2018-01-04	168.480367	171.2337	166.0442	169.229200	77508430.0
	2018-01-05	168.896475	172.0381	166.0442	169.840675	101168448.0
	2018-01-08	169.324680	172.2736	166.0442	170.080040	121736214.0
...	...	...	...	...	...	...
NFLX	2018-12-24	283.509250	332.0499	233.6800	281.931750	525657894.0
	2018-12-26	281.844500	332.0499	231.2300	280.777750	520444588.0
	2018-12-27	281.070488	332.0499	231.2300	280.162805	532679805.0
	2018-12-28	279.916341	332.0499	231.2300	279.461341	521968250.0
	2018-12-31	278.430769	332.0499	231.2300	277.451410	476309676.0

1255 rows × 5 columns

In [17]:
```
# 6. Create a pivot table of the FAANG data that compares the stocks. Put the ticke
table = pd.pivot_table(faang, index='ticker', aggfunc='mean')
table
```

Out[17]:

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 [18]:
```
# 7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NF
netflix = faang[faang['ticker'] == 'NFLX']
def z_score(column):
    return (column - column.mean()) / column.std()
z_scores = netflix.select_dtypes(include='number').apply(z_score)

z_scores
```

Out[18]:

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 [19]:
```
# 8. Add event descriptions
events_data = 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']
})

events_data['date'] = pd.to_datetime(events_data['date'])

# Merge with FAANG data using outer join
data = pd.merge(faang, events_data, on=['date', 'ticker'], how='outer')


data
```

Out[19]:

	date	ticker	open	high	low	close	volume	event
0	2018-01-02	AAPL	166.9271	169.0264	166.0442	168.9872	25555934	NaN
1	2018-01-02	AMZN	1172.0000	1190.0000	1170.5100	1189.0100	2694494	NaN
2	2018-01-02	FB	177.6800	181.5800	177.5500	181.4200	18151903	NaN
3	2018-01-02	GOOG	1048.3400	1066.9400	1045.2300	1065.0000	1237564	NaN
4	2018-01-02	NFLX	196.1000	201.6500	195.4200	201.0700	10966889	NaN
...	...	...	...	...	...	...	...	...
1250	2018-12-31	AAPL	157.8529	158.6794	155.8117	157.0663	35003466	NaN
1251	2018-12-31	AMZN	1510.8000	1520.7600	1487.0000	1501.9700	6954507	NaN
1252	2018-12-31	FB	134.4500	134.6400	129.9500	131.0900	24625308	NaN
1253	2018-12-31	GOOG	1050.9600	1052.7000	1023.5900	1035.6100	1493722	NaN
1254	2018-12-31	NFLX	260.1600	270.1001	260.0000	267.6600	13508920	NaN

1255 rows × 8 columns

In [178…
```
datas = data[data['event'] == 'FTC investigation']
datas
```

Out[178…

	date	ticker	open	high	low	close	volume	event
267	2018-03-20	FB	167.47	170.2	161.95	168.15	129851768	FTC investigation

In [179…
```
datas = data[data['event'] == 'Cambridge Analytica story']
datas
```

Out[179…

	date	ticker	open	high	low	close	volume	event
262	2018-03-19	FB	177.01	177.17	170.06	172.56	88140060	Cambridge Analytica story

In [180…
```
datas = data[data['event'] == 'Disappointing user growth announced after close.']
datas
```

Out[180…

	date	ticker	open	high	low	close	volume	event
707	2018-07-25	FB	215.715	218.62	214.27	217.5	64592585	Disappointing user growth announced after close.

In [181…
```
# 9. Use the transform() method on the FAANG data to represent all the values in te
datb = faang.groupby('ticker').transform(lambda x: x / x.iloc[0])

datb
```

Out[181...

date	open	high	low	close	volume
2018-01-02	1.000000	1.000000	1.000000	1.000000	1.000000
2018-01-03	1.023638	1.017623	1.021290	1.017914	0.930292
2018-01-04	1.040635	1.025498	1.036889	1.016040	0.764707
2018-01-05	1.044518	1.029298	1.041566	1.029931	0.747830
2018-01-08	1.053579	1.040313	1.049451	1.037813	0.991341
...	...	...	...	...	...
2018-12-24	0.928993	0.940578	0.928131	0.916638	1.285047
2018-12-26	0.943406	0.974750	0.940463	0.976019	1.917695
2018-12-27	0.970248	0.978396	0.953857	0.980169	1.704782
2018-12-28	1.001221	0.989334	0.988395	0.973784	1.142383
2018-12-31	1.002499	0.986653	0.979296	0.972404	1.206986

1255 rows × 5 columns