Hands-on Activity 8.1: Aggregating Data with Pandas

CPE311 - Computational Thinking with Python

Name: Dela Cruz, Gabrielle

Section: CPE22S3

Performed on: 07/08/2024

Submitted on: 07/xx/2024

Submitted to: Engr. Roman M. Richard

Link to Colab: https://colab.research.google.com/drive/14YQh2gF-cAv2-DIx5P0s8WihFp9USBO-?usp=sharing

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

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

earthquakes =  pd.read_csv('/content/drive/MyDrive/Dataset Uploads/earthquakes.csv')
earthquakes.head()
```

	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

Next steps: Generate code with earthquakes | View recommended plots

```
earthquakeData = earthquakes.query('parsed_place == "Japan" & magType == "mb" & mag >= 4.9
earthquakeData
```

	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

Next steps: Generate code with earthquakeData | View recommended plots

```
# Creating bins for each full number of magnitude.
earthquakes.query('magType == "ml"').sort_values('mag',ascending=False)

bin = [-2,-1,0,1,2,3,4,5,6]
magnitudeBin = pd.cut(earthquakes.query('magType == "ml"').mag, bins = bin)
```

```
magnitudeBin = pd.cut(earthquakes.query('magType == "ml"').mag, bins = bin)
magnitudeBin.value_counts()
```

```
mag
(1, 2]      3105
(0, 1]      2207
(2, 3]       862
(-1, 0]      491
(3, 4]       122
(-2, -1]      13
(4, 5]         2
(5, 6]         1
Name: count, dtype: int64
```

```
# With faang.csv file, to group by ticker and resampling to frequency by month.
# Code setup with faang.csv
faang = pd.read_csv('/content/drive/MyDrive/Dataset Uploads/faang.csv')
faang.head()
```

	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

Next steps: Generate code with faang | View recommended plots

```
faang['date'] = pd.to_datetime(faang['date'])
faang = faang.set_index('date')
faang.groupby('ticker').resample('M').agg({'open':'mean', 'high':'max', 'low':'min', 'close
```

		open	high	low	close	volume
ticker	**date**					
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

```
# Building a crosstab with the columns of tsunami and magType, with maximum magnitude for
pd.crosstab(index=earthquakes.tsunami, columns=earthquakes.magType, colnames=['magnitude
```

magnitude type	**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

```
# Calculating the rolling 60-day aggregations of OHLC data by ticker.
faang.groupby('ticker').rolling('60D').agg({'open':'mean', 'high':'max', 'low':'min', 'cl
```

		open	high	low	close	volume
ticker	**date**					
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

```
# Creating a pivot table of faang comparing the stocks, putting the ticker by rows and th
faang.pivot_table(index='ticker', values=['open', 'high', 'low', 'close', 'volume'], aggf
```

	close	high	low	open	volume
ticker					
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

```
# Calculation of Netflix's Z-score for each numeric column of Netflix data.
nflx_zScore = faang.query('ticker == "NFLX"').loc[:,['close', 'high', 'low', 'open', 'vol
nflx_zScore
```

	close	high	low	open	volume

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

251 rows × 5 columns

Next steps: Generate code with `nflx_zScore` View recommended plots

```
# Addition of event descriptions.
dataframeNew = pd.DataFrame({'ticker':'FB', 'date':['2018-07-25', '2018-03-19', '2018-03-
dataframeNew
```

	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: Generate code with `dataframeNew` View recommended plots

```
dataframeNew.set_index(['date', 'ticker'])
dataframeNew['date'] = pd.to_datetime(dataframeNew['date'])
dataframeNew
```

	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: Generate code with dataframeNew View recommended plots

```
outerJoin = faang.merge(dataframeNew, on=['date', 'ticker'], how='outer')
outerJoin
```

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

1255 rows × 8 columns

Next steps: Generate code with outerJoin View recommended plots

```
outerJoin.query('date == "2018-07-25"')
```

	date	ticker	open	high	low	close	volume	ev
141	2018-07-25	FB	215.7150	218.6200	214.2700	217.5000	64592585	Disappoir user gr annour after cl
392	2018-07-25	AAPL	190.8977	192.6675	190.2746	192.6378	16826483	
643	2018-07-25	AMZN	1829.3000	1863.8400	1822.6400	1863.6100	3836333	
894	2018-07-25	NFLX	357.5700	363.2800	355.6500	362.8700	8516248	

```
# Using the transform() method on the faang data to represent values in the data.
```

```
def getIndex(x):
  return x/x.iloc[0]
faangIndex = faang.groupby('ticker').transform(getIndex)
faangIndex
```

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

Next steps: Generate code with faangIndex View recommended plots