# Hands-on Activity 8.1: Aggregating Data with Pandas

---

**Name**: Cuadra, Audrick Zander G.

**Section**: CPE22S3

**Date**: March 29, 2024

**Submitted to**: Engr. Roman Richard

---

**About the data**

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**

The modules demonstrates methods that make use of several Python packages for data analysis. This module includes a number of libraries, including data cleaning, modification, and visualization. Additionally, by utilizing real-world data, it illustrates the significance of data analysis from the real world. Overall, the modules offer learners more than enough opportunities to learn about various data manipulation and analysis approaches.

**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.

```
# Used for the few first column as well as the structure of the dataframe
import pandas as p

earthquakes = p.read_csv('/content/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: View recommended plots

```
# filters out the column name and column value type as well as mag limiter
earthquakes_data = earthquakes.query('magType == "mb" and mag >= 4.9')
earthquakes_data
```

| | 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 |

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.

```
# filters out the the rows that contains the magType ml
earthquakes_ml = earthquakes.query('magType == "ml"')
earthquakes_ml
```

| | 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 |
| 6 | 1.70 | ml | 1539473176017 | 105km W of Talkeetna, Alaska | 0 | Alaska |
| ... | ... | ... | ... | ... | ... | ... |
| 9325 | 0.51 | ml | 1537230344890 | 4km WNW of Julian, CA | 0 | California |
| 9326 | 1.82 | ml | 1537230230260 | 4km W of Julian, CA | 0 | California |
| 9328 | 1.00 | ml | 1537230135130 | 3km W of Julian, CA | 0 | California |

Next steps: View recommended plots

```
# checks for the maximum value of mag to know how many bins should be made
max(earthquakes_ml.mag)
```

```
5.1
```

```
# creation of bins based on the specified conditions and displays value count
magnitude_binned = p.cut(
    earthquakes_ml.mag, bins=6, labels=['0-1', '1-2', '2-3', '3-4', '4-5', '5-6']
    )
magnitude_binned.value_counts()
```

```
2-3    3436
1-2    1889
3-4    1027
0-1     288
4-5     160
5-6       3
Name: mag, dtype: int64
```

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

```
# reads the csv file and checks the structure of the dataframe
faang = p.read_csv('/content/faang.csv', index_col= 'date', parse_dates=True)
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

```
# displayed value based on the what was asked
import numpy as ny

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

```
open      6.871481e+02
high      2.050500e+03
low       1.230200e+02
close     6.865478e+02
volume    2.022356e+10
dtype: float64
```

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.

```
# made a list of values of each columns
# displayed the values of max magnitude for each combination
earthquakes_crosstab = p.crosstab(
    index=earthquakes['tsunami'],
    columns=earthquakes['magType'],
    values=earthquakes['mag'],
    aggfunc=ny.max
)
earthquakes_crosstab
```

| magType / tsunami | mb | mb_lg | md | mh | ml | ms_20 | mw | mwb | mwr | mww |
|---|---|---|---|---|---|---|---|---|---|---|
| **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 |

Next steps: View recommended plots

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

```
# checks the structure of the dataframe
faang
```

| | ticker | open | high | low | close | volume |
|---|---|---|---|---|---|---|
| **date** | | | | | | |
| **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 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2018-12-24** | GOOG | 973.90 | 1003.54 | 970.1100 | 976.22 | 1590328 |
| **2018-12-26** | GOOG | 989.01 | 1040.00 | 983.0000 | 1039.46 | 2373270 |
| **2018-12-27** | GOOG | 1017.15 | 1043.89 | 997.0000 | 1043.88 | 2109777 |
| **2018-12-28** | GOOG | 1049.62 | 1055.56 | 1033.1000 | 1037.08 | 1413772 |
| **2018-12-31** | GOOG | 1050.96 | 1052.70 | 1023.5900 | 1035.61 | 1493722 |

1255 rows × 6 columns

Next steps: View recommended plots

```
# groups the OHLC aggregation by its ticker
faang.groupby('ticker').rolling('60D').agg({
    'open': ny.mean,
    'high': ny.max,
    'low': ny.min,
    'close': ny.mean,
    'volume': ny.sum
})
```

| | | 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

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.

```
faang_pivot = p.pivot(faang,
                      index=['ticker', 'open', 'high', 'low', 'close', 'volume'],
                      columns=[],
                      values=['open', 'high', 'low', 'close'],
                      )
ny.mean(faang_pivot)
```

```
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3502: FutureWarning: In a future version, [
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
open     687.148081
high     695.272838
low      677.693621
close    686.547753
dtype: float64
```

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

```
# filters out all of the values with the ticker "NFLX"
faang_data = faang.query('ticker == "NFLX"')
faang_data
```

| date | ticker | open | high | low | close | volume |
|---|---|---|---|---|---|---|
| **2018-01-02** | NFLX | 196.10 | 201.6500 | 195.4200 | 201.070 | 10966889 |
| **2018-01-03** | NFLX | 202.05 | 206.2100 | 201.5000 | 205.050 | 8591369 |
| **2018-01-04** | NFLX | 206.20 | 207.0500 | 204.0006 | 205.630 | 6029616 |
| **2018-01-05** | NFLX | 207.25 | 210.0200 | 205.5900 | 209.990 | 7033240 |
| **2018-01-08** | NFLX | 210.02 | 212.5000 | 208.4400 | 212.050 | 5580178 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2018-12-24** | NFLX | 242.00 | 250.6500 | 233.6800 | 233.880 | 9547616 |
| **2018-12-26** | NFLX | 233.92 | 254.5000 | 231.2300 | 253.670 | 14402735 |
| **2018-12-27** | NFLX | 250.11 | 255.5900 | 240.1000 | 255.565 | 12235217 |
| **2018-12-28** | NFLX | 257.94 | 261.9144 | 249.8000 | 256.080 | 10987286 |
| **2018-12-31** | NFLX | 260.16 | 270.1001 | 260.0000 | 267.660 | 13508920 |

251 rows × 6 columns

Next steps: View recommended plots

```
# checks the columns that have numerical datatype
faang.dtypes
```

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

```
# calls upon all of the numerical columns and performs z score operation to all of them
faang_data_z_score = faang_data.loc[
    '2018', ['open', 'high', 'low', 'close', 'volume']
].apply(
    lambda x: x.sub(x.mean()).div(x.std())
    )
faang_data_z_score
```

| | open | high | low | close | volume |
|---|---|---|---|---|---|
| **date** | | | | | |
| **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

Next steps: View recommended plots

8. Add event descriptions:

a.) 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']

b.) Set the index to ['date', 'ticker']

c.) Merge this data with the FAANG data using an outer join

```
# creates a new dataframe for faang.csv
faang2 = p.read_csv(
    '/content/faang.csv'
    )
faang2
```

| | 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

```
# filters out the ticker "FB"
faang_fb = faang2.query('ticker == "FB"')
faang_fb
```

| | 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 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 246 | FB | 2018-12-24 | 123.10 | 129.74 | 123.0200 | 124.06 | 22066002 |
| 247 | FB | 2018-12-26 | 126.00 | 134.24 | 125.8900 | 134.18 | 39723370 |
| 248 | FB | 2018-12-27 | 132.44 | 134.99 | 129.6700 | 134.52 | 31202509 |
| 249 | FB | 2018-12-28 | 135.34 | 135.92 | 132.2000 | 133.20 | 22627569 |
| 250 | FB | 2018-12-31 | 134.45 | 134.64 | 129.9500 | 131.09 | 24625308 |

251 rows × 7 columns

Next steps: View recommended plots

```
# creates a new dataframe with the extracted date and ticker column
faang_new = faang_fb.filter(['date', 'ticker'])
faang_new
```

| | date | ticker |
|---|---|---|
| 0 | 2018-01-02 | FB |
| 1 | 2018-01-03 | FB |
| 2 | 2018-01-04 | FB |
| 3 | 2018-01-05 | FB |
| 4 | 2018-01-08 | FB |
| ... | ... | ... |
| 246 | 2018-12-24 | FB |
| 247 | 2018-12-26 | FB |
| 248 | 2018-12-27 | FB |
| 249 | 2018-12-28 | FB |
| 250 | 2018-12-31 | FB |

251 rows × 2 columns

Next steps: View recommended plots

```
# writes a condition wherein the events would appear with a specific date
# otherwise the date's event would be "NaN"
date_new = faang_new['date'].isin(['2018-07-25', '2018-03-19', '2018-03-20'])
events = ['Disappointing user growth announced after close.',
          'Cambridge Analytica story',
          'FTC investigation']

faang_new['events'] = 'NaN'
faang_new.loc[date_new, 'events'] = ', '.join(events)

faang_new.query('date == ["2018-07-25", "2018-03-19", "2018-03-20"]')
```

| | date | ticker | events |
|---|---|---|---|
| 52 | 2018-03-19 | FB | Disappointing user growth announced after clos... |
| 53 | 2018-03-20 | FB | Disappointing user growth announced after clos... |
| 141 | 2018-07-25 | FB | Disappointing user growth announced after clos... |

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

```
faang_new_index = faang.groupby('ticker').transform(
    lambda x: x.iloc[0]
)
faang_new_index
```

| | open | high | low | close | volume |
|---|---|---|---|---|---|
| date | | | | | |
| **2018-01-02** | 177.68 | 181.58 | 177.55 | 181.42 | 18151903.0 |
| **2018-01-03** | 177.68 | 181.58 | 177.55 | 181.42 | 18151903.0 |
| **2018-01-04** | 177.68 | 181.58 | 177.55 | 181.42 | 18151903.0 |
| **2018-01-05** | 177.68 | 181.58 | 177.55 | 181.42 | 18151903.0 |
| **2018-01-08** | 177.68 | 181.58 | 177.55 | 181.42 | 18151903.0 |
| **...** | ... | ... | ... | ... | ... |
| **2018-12-24** | 1048.34 | 1066.94 | 1045.23 | 1065.00 | 1237564.0 |
| **2018-12-26** | 1048.34 | 1066.94 | 1045.23 | 1065.00 | 1237564.0 |

Next steps: View recommended plots