# CPE 311 - Computational Thinking with Python

**Name:** Gwyneth D. Esperat

**Section:** CPE22S3

**Date:** March 27, 2024

**Github Link:** [Module 8]

## 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

```
import pandas as pd

# Load the earthquakes data
earthquakes = pd.read_csv('/content/earthquakes.csv')

# Filter earthquakes in Japan with magType 'mb' and magnitude >= 4.9
earthquakes_japan = earthquakes[(earthquakes['place'].str.contains('Japan')) &
                                (earthquakes['magType'] == 'mb') &
                                (earthquakes['mag'] >= 4.9)]

earthquakes_japan
```

| | 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 |
| **3632** | 4.9 | mb | 1538450871260 | 53km ESE of Hitachi, Japan | 0 | Japan |

```
# Create bins for magnitudes
bins = pd.interval_range(start=0, end=int(earthquakes['mag'].max()) + 1, freq=1)
earthquakes['magnitude_bin'] = pd.cut(earthquakes[earthquakes['magType'] == 'ml']['mag'], bins=bins)

# Count earthquakes in each bin
magnitude_counts = earthquakes.groupby('magnitude_bin').size()

magnitude_counts
```

```
magnitude_bin
(0, 1]    2207
(1, 2]    3105
(2, 3]     862
(3, 4]     122
(4, 5]       2
(5, 6]       1
(6, 7]       0
(7, 8]       0
dtype: int64
```

```
# Load the FAANG data
faang = pd.read_csv('/content/faang.csv', parse_dates=['date'])
faang.set_index('date', inplace=True)

# Group by ticker and resample to monthly frequency with aggregations
faang_monthly = faang.groupby('ticker').resample('M').agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})

faang_monthly
```

| 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 |

```
# Create a crosstab
crosstab_max_magnitude = pd.crosstab(index=earthquakes['tsunami'], columns=earthquakes['magType'],
                                     values=earthquakes['mag'], aggfunc='max')

crosstab_max_magnitude
```

| 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 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 2018-03-31 | 1096.108095 | 1177.0500 | 980.6400 | 1091.490476 | 45430049 |

```
# Rolling 60-day aggregations by ticker
rolling_60d = faang.groupby('ticker').rolling(window='60D').agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})

rolling_60d
```

| 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

```
# Pivot table for FAANG data
faang_pivot = faang.pivot_table(index='ticker', values=['open', 'high', 'low', 'close', 'volume'], aggfunc='mean')

faang_pivot
```

| 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 |

```
# Calculate Z-scores for Netflix data
netflix_data = faang[faang['ticker'] == 'NFLX'].select_dtypes(include=['float64', 'int64'])

z_scores = netflix_data.apply(lambda x: (x - x.mean()) / x.std())

z_scores
```

| 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

```
# Create an event dataframe
events_df = pd.DataFrame({
    'ticker': 'FB',
    'date': pd.to_datetime(['2018-07-25', '2018-03-19', '2018-03-20']),
    'event': ['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC investigation']
}).set_index(['date', 'ticker'])

# Merge with FAANG data
faang_reset = faang.reset_index()
events_merged = pd.merge(faang_reset, events_df, on=['date', 'ticker'], how='outer').set_index(['date', 'ticker'])

events_merged
```

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

1255 rows × 6 columns

## 8.1.4 Data Analysis

Provide some comments here about the results of the procedures.

### 1. Earthquakes in Japan

- **Observation**: A few earthquakes in Japan with a magnitude of 4.9 or greater and `magType` of `mb` were identified.
- **Comments**: These specific earthquakes might be significant enough to be studied for patterns or potential impacts on the region, considering their relatively higher magnitude.

### 2. Magnitude Bins with `magType ml`

- **Observation**: The majority of earthquakes fall into lower magnitude bins, with a sharp decrease in frequency as magnitude increases.

- **Comments**: This distribution is expected and aligns with the general understanding that lower magnitude earthquakes are far more common than higher magnitude ones.

3. Monthly Aggregations for FAANG Data

- **Observation**: Monthly aggregated data shows the variability in trading volumes and price ranges for each FAANG company over time.
- **Comments**: These trends can help investors understand historical performance, identify seasonal patterns, and make informed decisions.

4. Crosstab between `tsunami` and `magType`

- **Observation**: The crosstab revealed the maximum magnitudes associated with tsunamis for different `magTypes`.
- **Comments**: This analysis could be critical for disaster preparedness and understanding the correlation between earthquake characteristics and tsunami generation.

5. Rolling 60-day Aggregations for FAANG Data

- **Observation**: Rolling averages smooth out short-term fluctuations and highlight longer-term trends in stock prices and trading volumes.
- **Comments**: Investors might use this information to gauge the momentum of a stock and identify potential buying or selling opportunities.

6. Pivot Table for FAANG Data

- **Observation**: The pivot table provided a comparative overview of the average open, high, low, close prices, and volumes for FAANG stocks.
- **Comments**: This comparative analysis is useful for portfolio diversification, showing how different stocks behave over time relative to each other.

7. Z-scores for Netflix's Data

- **Observation**: Calculating Z-scores for Netflix's data highlights how each trading day's prices and volumes deviate from the mean.
- **Comments**: This can help identify outliers or unusual trading days that may be driven by specific events or announcements.

8. FAANG Data with Event Descriptions Added

- **Observation**: Incorporating significant events into the FAANG dataset provides context for stock price movements.
- **Comments**: Understanding the impact of specific events on stock performance is crucial for fundamental analysis and can guide investment strategies.

## 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.

```
import pandas as pd

earthquakes = pd.read_csv('/content/earthquakes.csv')
earthquakes
```

| | 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

```
filtered_earthquakes = earthquakes[
    (earthquakes['place'].str.contains("Japan")) &
    (earthquakes['magType'] == 'mb') &
    (earthquakes['mag'] >= 4.9)
]

print(filtered_earthquakes)
```

```
      mag magType           time                        place  tsunami  \
1563  4.9      mb  1538977532250  293km ESE of Iwo Jima, Japan        0
2576  5.4      mb  1538697528010    37km E of Tomakomai, Japan        0
3072  4.9      mb  1538579732490     15km ENE of Hasaki, Japan        0
3632  4.9      mb  1538450871260    53km ESE of Hitachi, Japan        0

     parsed_place
1563        Japan
2576        Japan
3072        Japan
3632        Japan
```

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.

```
bins = range(int(earthquakes['mag'].min()), int(earthquakes['mag'].max()) + 2)
mag_ml_df = earthquakes[earthquakes['magType'] == 'ml']

mag_ml_df = mag_ml_df.copy()
mag_ml_df['magnitude_bin'] = pd.cut(mag_ml_df['mag'], bins=bins, include_lowest=True, right=False)

magnitude_counts = mag_ml_df['magnitude_bin'].value_counts().sort_index()

print(magnitude_counts)
```

```
[-1, 0)     446
[0, 1)     2072
[1, 2)     3126
[2, 3)      985
[3, 4)      153
[4, 5)        6
[5, 6)        2
[6, 7)        0
[7, 8)        0
Name: magnitude_bin, 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

```
import pandas as pd

# Corrected: Load the FAANG data
faang_df = pd.read_csv('/content/faang (1).csv', parse_dates=['date'])
faang_df.set_index('date', inplace=True)

# Group by ticker and resample to monthly frequency
faang_monthly = faang_df.groupby('ticker').resample('M').agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
})

print(faang_monthly.head())
```

```
                         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
```

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.

```
earthquakes = pd.read_csv('/content/earthquakes.csv')

# Build a crosstab showing the maximum magnitude for each combination of tsunami and magType
tsunami_magType_max_mag = pd.crosstab(index=earthquakes['tsunami'], columns=earthquakes['magType'],
                                      values=earthquakes['mag'], aggfunc='max')

tsunami_magType_max_mag
```

| 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.

```
# Ensure the FAANG data is loaded and indexed by date
faang_df = pd.read_csv('/content/faang (1).csv', parse_dates=['date'])
faang_df.set_index('date', inplace=True)

# Calculate the rolling 60-day aggregations of OHLC data by ticker
faang_rolling_60 = faang_df.groupby('ticker').rolling(window='60D').agg({
    'open': 'mean',
    'high': 'max',
    'low': 'min',
    'close': 'mean',
    'volume': 'sum'
}).dropna()

faang_rolling_60.head()
```

| | | 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 |

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.

```
# Create a pivot table for the FAANG data with ticker in the rows and averages of OHLC and volume
faang_pivot = faang_df.pivot_table(index='ticker', values=['open', 'high', 'low', 'close', 'volume'], aggfunc='mean')

faang_pivot
```

| | 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 |

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

```
# Filter the FAANG data for Netflix (NFLX) only
nflx_data = faang_df[faang_df['ticker'] == 'NFLX']

# Calculate Z-scores for each numeric column using apply()
nflx_z_scores = nflx_data.select_dtypes(include=['float64', 'int64']).apply(lambda x: (x - x.mean()) / x.std())

nflx_z_scores.head()
```

| | 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 |

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

```
# Create the events DataFrame
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']
})

# Convert 'date' to datetime format and set ['date', 'ticker'] as the index
events_data['date'] = pd.to_datetime(events_data['date'])
events_data.set_index(['date', 'ticker'], inplace=True)

# Ensure the FAANG data is in the correct format for merging
faang_df.reset_index(inplace=True)
faang_df.set_index(['date', 'ticker'], inplace=True)

# Merge the events data with the FAANG data using an outer join
faang_with_events = faang_df.join(events_data, on=['date', 'ticker'], how='outer').sort_index()

faang_with_events[faang_with_events['event'].notnull()]
```

| | | open | high | low | close | volume | event |
|---|---|---|---|---|---|---|---|
| **date** | **ticker** | | | | | | |
| **2018-03-19** | **FB** | 177.010 | 177.17 | 170.06 | 172.56 | 88140060 | Cambridge Analytica story |
| **2018-03-20** | **FB** | 167.470 | 170.20 | 161.95 | 168.15 | 129851768 | FTC investigation |
| **2018-07-25** | **FB** | 215.715 | 218.62 | 214.27 | 217.50 | 64592585 | Disappointing user growth announced after close. |

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.

```
# Reset index of the FAANG data to work with transform()
faang_df.reset_index(inplace=True)
faang_df.set_index('date', inplace=True)

# Function to calculate the index relative to the first row in the group
def index_relative_to_first(row):
    return row / row.iloc[0]

# Group by ticker and apply the transformation to numeric columns
faang_indexed = faang_df.groupby('ticker').transform(index_relative_to_first)

# Include the 'ticker' column back into the faang_indexed DataFrame for clarity
faang_indexed['ticker'] = faang_df['ticker']

faang_indexed.head()
```

| | open | high | low | close | volume | ticker |
|---|---|---|---|---|---|---|
| **date** | | | | | | |
| **2018-01-02** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | FB |
| **2018-01-03** | 1.023638 | 1.017623 | 1.021290 | 1.017914 | 0.930292 | FB |
| **2018-01-04** | 1.040635 | 1.025498 | 1.036889 | 1.016040 | 0.764707 | FB |
| **2018-01-05** | 1.044518 | 1.029298 | 1.041566 | 1.029931 | 0.747830 | FB |
| **2018-01-08** | 1.053579 | 1.040313 | 1.049451 | 1.037813 | 0.991341 | FB |

## Conclusion

In conclusion for the FAANG (Facebook, Apple, Amazon, Netflix, Google) and earthquake datasets yielded significant findings, showcasing the power of data manipulation and analysis. In examining earthquake data, we identified specific quakes in Japan based on magnitude and type, revealing the frequency distribution of these events and their potential to cause tsunamis. This analysis highlighted the commonality of lower magnitude earthquakes and provided insights into the relationship between earthquake characteristics and tsunamis. For the FAANG stocks, our investigation into monthly trading behaviors uncovered trends in prices and volumes, enhanced by a rolling 60-day aggregation for a deeper view of medium-term trends. A pivot table comparing the FAANG stocks illustrated variations in trading behavior, while the application of Z-scores to Netflix's data offered a statistical comparison of its performance. Additionally, merging specific event data with the FAANG dataset linked stock performance to major events, enriching the analysis. Transforming FAANG data to index values based on the first date facilitated