Hands-on Activity 7.1 Data Collection and Wrangling

Intended Learning Outcomes:

1. Demonstrate how to gather sensor data, image data and voice data
2. Demonstrate how to gather data (text and images) from web
3. Demonstrate how to prepare data using different data preprocessing techniques

Resources:

- Personal Computer
- Jupyter Notebook
- Internet Connection

Instruction:

1. Download the following datasets: aapl.csv, amzn.csv, fb.csv, goog.csv, nflx.csv
2. Accomplish the notebook for this activity and submit as a pdf file: Data Wrangling HOA.pdf

Exercise 1

We want to look at data for the Facebook, Apple, Amazon, Netflix, and Google (FAANG) stocks, but we were given each as a separate CSV file. Combine them into a single file and store the dataframe of the FAANG data as faang for the rest of the exercises:

1. Read each file in.
2. Add a column to each dataframe, called ticker, indicating the ticker symbol it is for (Apple's is AAPL, for example). This is how you look up a stock. Each file's name is also the ticker symbol, so be sure to capitalize it.
3. Append them together into a single dataframe.
4. Save the result in a CSV file called faang.csv.

```
# read each file in

import pandas as pd

filepathFB= pd.read_csv("content/fb.csv")
filepathAmz= pd.read_csv("content/amzn.csv")
filepathApp= pd.read_csv("content/aapl.csv")
filepathNet= pd.read_csv("content/nflx.csv")
filepathGoog= pd.read_csv("content/goog.csv")


# add a column to each dataframe called ticker

filepathFB['ticker'] = 'FB'
filepathFB
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	177.68	181.58	177.5500	181.42	18151903	FB
1	2018-01-03	181.88	184.78	181.3300	184.67	16886563	FB
2	2018-01-04	184.90	186.21	184.0996	184.33	13880896	FB
3	2018-01-05	185.59	186.90	184.9300	186.85	13574535	FB
4	2018-01-08	187.20	188.90	186.3300	188.28	17994726	FB
...	...	...	...	...	...	...	...
246	2018-12-24	123.10	129.74	123.0200	124.06	22066002	FB
247	2018-12-26	126.00	134.24	125.8900	134.18	39723370	FB
248	2018-12-27	132.44	134.99	129.6700	134.52	31202509	FB
249	2018-12-28	135.34	135.92	132.2000	133.20	22627569	FB
250	2018-12-31	134.45	134.64	129.9500	131.09	24625308	FB

251 rows × 7 columns

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

```
filepathAmz['ticker'] = 'AMZN'
filepathAmz
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	1172.00	1190.00	1170.51	1189.01	2694494	AMZN
1	2018-01-03	1188.30	1205.49	1188.30	1204.20	3108793	AMZN
2	2018-01-04	1205.00	1215.87	1204.66	1209.59	3022089	AMZN
3	2018-01-05	1217.51	1229.14	1210.00	1229.14	3544743	AMZN
4	2018-01-08	1236.00	1253.08	1232.03	1246.87	4279475	AMZN
...	...	...	...	...	...	...	...
246	2018-12-24	1346.00	1396.03	1307.00	1343.96	7219996	AMZN
247	2018-12-26	1368.89	1473.16	1363.01	1470.90	10411801	AMZN
248	2018-12-27	1454.20	1469.00	1390.31	1461.64	9722034	AMZN
249	2018-12-28	1473.35	1513.47	1449.00	1478.02	8828950	AMZN
250	2018-12-31	1510.80	1520.76	1487.00	1501.97	6954507	AMZN

251 rows × 7 columns

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

```
filepathApp['ticker'] = 'AAPL'
filepathApp
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
1	2018-01-03	169.2521	171.2337	168.6929	168.9578	29517899	AAPL
2	2018-01-04	169.2619	170.1742	168.8106	169.7426	22434597	AAPL
3	2018-01-05	170.1448	172.0381	169.7622	171.6751	23660018	AAPL
4	2018-01-08	171.0375	172.2736	170.6255	171.0375	20567766	AAPL
...	...	...	...	...	...	...	...
246	2018-12-24	147.5173	150.9027	145.9639	146.2029	37169232	AAPL
247	2018-12-26	147.6666	156.5585	146.0934	156.4987	58582544	AAPL
248	2018-12-27	155.1744	156.1004	149.4291	155.4831	53117065	AAPL
249	2018-12-28	156.8273	157.8430	153.8899	155.5627	42291424	AAPL
250	2018-12-31	157.8529	158.6794	155.8117	157.0663	35003466	AAPL

251 rows × 7 columns

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

```
filepathNet['ticker'] = 'NFLX'
filepathNet
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	196.10	201.6500	195.4200	201.070	10966889	NFLX
1	2018-01-03	202.05	206.2100	201.5000	205.050	8591369	NFLX
2	2018-01-04	206.20	207.0500	204.0006	205.630	6029616	NFLX
3	2018-01-05	207.25	210.0200	205.5900	209.990	7033240	NFLX
4	2018-01-08	210.02	212.5000	208.4400	212.050	5580178	NFLX
...	...	...	...	...	...	...	...
246	2018-12-24	242.00	250.6500	233.6800	233.880	9547616	NFLX
247	2018-12-26	233.92	254.5000	231.2300	253.670	14402735	NFLX
248	2018-12-27	250.11	255.5900	240.1000	255.565	12235217	NFLX
249	2018-12-28	257.94	261.9144	249.8000	256.080	10987286	NFLX
250	2018-12-31	260.16	270.1001	260.0000	267.660	13508920	NFLX

251 rows × 7 columns

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

```
filepathGoog['ticker'] = 'GOOG'
filepathGoog
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	1048.34	1066.94	1045.23	1065.00	1237564	GOOG
1	2018-01-03	1064.31	1086.29	1063.21	1082.48	1430170	GOOG
2	2018-01-04	1088.00	1093.57	1084.00	1086.40	1004605	GOOG
3	2018-01-05	1094.00	1104.25	1092.00	1102.23	1279123	GOOG
4	2018-01-08	1102.23	1111.27	1101.62	1106.94	1047603	GOOG
...	...	...	...	...	...	...	...
246	2018-12-24	973.90	1003.54	970.11	976.22	1590328	GOOG
247	2018-12-26	989.01	1040.00	983.00	1039.46	2373270	GOOG
248	2018-12-27	1017.15	1043.89	997.00	1043.88	2109777	GOOG
249	2018-12-28	1049.62	1055.56	1033.10	1037.08	1413772	GOOG
250	2018-12-31	1050.96	1052.70	1023.59	1035.61	1493722	GOOG

251 rows × 7 columns

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

```
# append together in single dataframe

faang = pd.concat([filepathFB, filepathAmz, filepathApp, filepathNet, filepathGoog])
print(faang)
```

```
           date     open     high        low    close    volume ticker
0    2018-01-02   177.68   181.58   177.5500   181.42  18151903     FB
1    2018-01-03   181.88   184.78   181.3300   184.67  16886563     FB
2    2018-01-04   184.90   186.21   184.0996   184.33  13880896     FB
3    2018-01-05   185.59   186.90   184.9300   186.85  13574535     FB
4    2018-01-08   187.20   188.90   186.3300   188.28  17994726     FB
..          ...      ...      ...        ...      ...       ...    ...
246  2018-12-24   973.90  1003.54   970.1100   976.22   1590328   GOOG
247  2018-12-26   989.01  1040.00   983.0000  1039.46   2373270   GOOG
248  2018-12-27  1017.15  1043.89   997.0000  1043.88   2109777   GOOG
249  2018-12-28  1049.62  1055.56  1033.1000  1037.08   1413772   GOOG
250  2018-12-31  1050.96  1052.70  1023.5900  1035.61   1493722   GOOG

[1255 rows x 7 columns]
```

```
# save result in a new csv file

faang.to_csv("content/faang.csv")
filepathFaang=pd.read_csv("content/faang.csv")
print("New Dataframe 'faang.csv' ")
print(filepathFaang)
```

```
New Dataframe 'faang.csv'
      Unnamed: 0        date     open     high        low    close    volume  \
0              0  2018-01-02   177.68   181.58   177.5500   181.42  18151903
1              1  2018-01-03   181.88   184.78   181.3300   184.67  16886563
2              2  2018-01-04   184.90   186.21   184.0996   184.33  13880896
3              3  2018-01-05   185.59   186.90   184.9300   186.85  13574535
4              4  2018-01-08   187.20   188.90   186.3300   188.28  17994726
...          ...         ...      ...      ...        ...      ...       ...
1250         246  2018-12-24   973.90  1003.54   970.1100   976.22   1590328
1251         247  2018-12-26   989.01  1040.00   983.0000  1039.46   2373270
1252         248  2018-12-27  1017.15  1043.89   997.0000  1043.88   2109777
1253         249  2018-12-28  1049.62  1055.56  1033.1000  1037.08   1413772
1254         250  2018-12-31  1050.96  1052.70  1023.5900  1035.61   1493722

     ticker
0        FB
1        FB
2        FB
3        FB
4        FB
...     ...
1250   GOOG
1251   GOOG
1252   GOOG
1253   GOOG
1254   GOOG

[1255 rows x 8 columns]
```

Exercise 2

- With faang, use type conversion to change the date column into a datetime and the volume column into integers. Then, sort by date and ticker.
- Find the seven rows with the highest value for volume.
- Right now, the data is somewhere between long and wide format. Use melt() to make it completely long format. Hint: date and ticker are our ID variables (they uniquely identify each row). We need to melt the rest so that we don't have separate columns for open, high, low, close, and volume.

```
# Convert 'date' column - datetime
faang['date'] = pd.to_datetime(faang['date'])

# Convert 'volume' column - integers
faang['volume'] = faang['volume'].astype(int)

# Sort 'date' and 'ticker'
faang = faang.sort_values(by=['date', 'ticker'])

print(filepathFaang)
```

```
      Unnamed: 0        date     open     high        low    close    volume  \
0              0  2018-01-02   177.68   181.58   177.5500   181.42  18151903
1              1  2018-01-03   181.88   184.78   181.3300   184.67  16886563
2              2  2018-01-04   184.90   186.21   184.0996   184.33  13880896
3              3  2018-01-05   185.59   186.90   184.9300   186.85  13574535
4              4  2018-01-08   187.20   188.90   186.3300   188.28  17994726
...          ...         ...      ...      ...        ...      ...       ...
1250         246  2018-12-24   973.90  1003.54   970.1100   976.22   1590328
1251         247  2018-12-26   989.01  1040.00   983.0000  1039.46   2373270
1252         248  2018-12-27  1017.15  1043.89   997.0000  1043.88   2109777
1253         249  2018-12-28  1049.62  1055.56  1033.1000  1037.08   1413772
1254         250  2018-12-31  1050.96  1052.70  1023.5900  1035.61   1493722

     ticker
0        FB
1        FB
2        FB
3        FB
4        FB
...     ...
1250   GOOG
1251   GOOG
1252   GOOG
1253   GOOG
1254   GOOG

[1255 rows x 8 columns]
```

```
# find seven rows with highest value for volume

top_seven_volume = faang.nlargest(7, 'volume')


# using pd.melt()
faangLong = pd.melt(faang, id_vars=['date', 'ticker'], var_name='variable', value_name='valu

# saving to a new csv file
faangLong.to_csv("content/faang_long_format.csv")

filepathLong= pd.read_csv("content/faang_long_format.csv")
print(filepathLong)
```

```
      Unnamed: 0        date ticker variable         value
0              0  2018-01-02   AAPL     open  1.669271e+02
1              1  2018-01-02   AMZN     open  1.172000e+03
2              2  2018-01-02     FB     open  1.776800e+02
3              3  2018-01-02   GOOG     open  1.048340e+03
4              4  2018-01-02   NFLX     open  1.961000e+02
...          ...         ...    ...      ...           ...
6270        6270  2018-12-31   AAPL   volume  3.500347e+07
6271        6271  2018-12-31   AMZN   volume  6.954507e+06
6272        6272  2018-12-31     FB   volume  2.462531e+07
6273        6273  2018-12-31   GOOG   volume  1.493722e+06
6274        6274  2018-12-31   NFLX   volume  1.350892e+07

[6275 rows x 5 columns]
```

Exercise 3

- Using web scraping, search for the list of the hospitals, their address and contact information. Save the list in a new csv file, hospitals.csv.
- Using the generated hospitals.csv, convert the csv file into pandas dataframe. Prepare the data using the necessary preprocessing techniques.

```
# creating hospitals.csv

import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np

url = 'https://en.wikipedia.org/wiki/List_of_hospitals_in_South_Korea' # list of hospitals i

# send a get requests
response = requests.get(url)
```

```
# convert csv file into pandas dataframe and prepare the data for pre-processing

import pandas as pd

# Read the CSV file into a Pandas DataFrame
hospitals_df = pd.read_csv('content/hospitals.csv')

# Check for missing values
missing_values = hospitals_df.isnull().sum()

# Drop rows with missing values
hospitals_df = hospitals_df.dropna()

# Reset the index
hospitals_df = hospitals_df.reset_index(drop=True)
            name = cols[0].text.strip()
```

```
# show table hospitals

print(hospitals_df.to_string())
```

```
Empty DataFrame
Columns: [Name, Location, Contact]
Index: []
```

Conclusion

In conclusion, I applied the previous learnings in other modules in this activity. The pandas and numpy, as well as the new library which is BeautifulSoup and requests can be helpful in future topics. Using these libraries will be important for data collection and wrangling. Additionally, I have learned web scraping is also useful since it helps you scrape data from the internet and web pages like the one I did in Exercise 3. Lastly, it is important we need to learn different data collection techniques because these will help us in the near future for Data Science.