CPE311 Computational Thinking with Python

submitted by: Calamba, Liam Francis

performed on: 03/20/2024

performed on: 03/20/2024

submitted to: Engr. Roman M. Richard

7.1 SUPPLEMENTARY ACTIVITY

using the datasets provided, perform the following exercises:

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 FAANG data as faang for the rest of the exercises:

1. Read each file in

```
import pandas as p
import numpy as n

apl = '/content/aapl.csv'
amzn = '/content/amzn.csv'
fb = '/content/fb.csv'
goog = '/content/goog.csv'
nflx = '/content/nflx.csv'

FB = p.read_csv(fb)
AAPL = p.read_csv(apl)
AMZN = p.read_csv(amzn)
NFLX = p.read_csv(nflx)
GOOG = p.read_csv(goog)
```

2. Add a column to each dataframe, called ticker, indicating the ticker symbol it is for (Apple's is APPL, 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.

```
AAPL.loc[:, 'ticker'] = 'AAPL'
AAPL
```

	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: View recommended plots

```
AMZN.loc[:, 'ticker'] = 'AMZN'
AMZN
```

	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: View recommended plots

```
FB.loc[:, 'ticker'] = 'FB'
FB
```

	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: View recommended plots

```
GOOG.loc[:, 'ticker'] = 'GOOG'
GOOG
```

	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: View recommended plots

```
NFLX.loc[:, 'ticker'] = 'NFLX'
NFLX
```

	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: View recommended plots

3. Append them together into a single dataframe.

```
faang = p.concat([FB,AAPL,AMZN,NFLX,GOOG])
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 × 7 columns

Next steps: View recommended plots

4. Save the result in a CSV file called faang.csv

```
faang.to_csv('/content/faang.csv', index=False)
```

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 copmpletely 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.

```
faang['date'] = p.to_datetime(faang['date'])# change date datatype to datetime
faang['volume'].astype(int) # change volume datatype to integer
faang.dtypes # verify
```

```
date      datetime64[ns]
open             float64
high             float64
low              float64
close            float64
volume             int64
ticker            object
dtype: object
```

```
faang.sort_values(by=['date', 'ticker'], inplace=True) # sort faang by date and ticker
faang # verify
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
0	2018-01-02	1172.0000	1190.0000	1170.5100	1189.0100	2694494	AMZN
0	2018-01-02	177.6800	181.5800	177.5500	181.4200	18151903	FB
0	2018-01-02	1048.3400	1066.9400	1045.2300	1065.0000	1237564	GOOG
0	2018-01-02	196.1000	201.6500	195.4200	201.0700	10966889	NFLX
...	...	...	...	...	...	...	...
250	2018-12-31	157.8529	158.6794	155.8117	157.0663	35003466	AAPL
250	2018-12-31	1510.8000	1520.7600	1487.0000	1501.9700	6954507	AMZN
250	2018-12-31	134.4500	134.6400	129.9500	131.0900	24625308	FB
250	2018-12-31	1050.9600	1052.7000	1023.5900	1035.6100	1493722	GOOG
250	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

```
faang
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
0	2018-01-02	1172.0000	1190.0000	1170.5100	1189.0100	2694494	AMZN
0	2018-01-02	177.6800	181.5800	177.5500	181.4200	18151903	FB
0	2018-01-02	1048.3400	1066.9400	1045.2300	1065.0000	1237564	GOOG
0	2018-01-02	196.1000	201.6500	195.4200	201.0700	10966889	NFLX
...	...	...	...	...	...	...	...
250	2018-12-31	157.8529	158.6794	155.8117	157.0663	35003466	AAPL
250	2018-12-31	1510.8000	1520.7600	1487.0000	1501.9700	6954507	AMZN
250	2018-12-31	134.4500	134.6400	129.9500	131.0900	24625308	FB
250	2018-12-31	1050.9600	1052.7000	1023.5900	1035.6100	1493722	GOOG
250	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

```
faang.nlargest(7,['volume']) # 7 rows with highest value for volume
```

	date	open	high	low	close	volume	ticker
142	2018-07-26	174.8900	180.1300	173.7500	176.2600	169803668	FB
53	2018-03-20	167.4700	170.2000	161.9500	168.1500	129851768	FB
57	2018-03-26	160.8200	161.1000	149.0200	160.0600	126116634	FB
54	2018-03-21	164.8000	173.4000	163.3000	169.3900	106598834	FB
182	2018-09-21	219.0727	219.6482	215.6097	215.9768	96246748	AAPL
245	2018-12-21	156.1901	157.4845	148.9909	150.0862	95744384	AAPL
212	2018-11-02	207.9295	211.9978	203.8414	205.8755	91328654	AAPL

```
# melted

faang.melt(id_vars = ['date','ticker'],
           value_vars = ['open','high','low','close', 'volume']
)
```

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

6275 rows × 4 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, hospital.csv.
- Using the generated hospitals.csv, convert the csv file into pandas dataframe. Prepare the data using the necessary preprocessing techniques.

```
'''di ko po gets yung data scraping :('''


# import requests
# from bs4 import BeautifulSoup
# import pandas as pd

# # URL of the website you want to scrape
# url = 'https://en.wikipedia.org/wiki/List_of_hospitals_in_Metro_Manila'

# # Send a GET request to the website
# response = requests.get(url)

# # Check if the request was successful
# if response.status_code == 200:
#     # Parse the content of the response using BeautifulSoup
#     soup = BeautifulSoup(response.content, 'html.parser')

#     # Extract the name, address, and contact information of each hospital
#     hospitals = []
#     for hospital in soup.select('.hospital'):
#         name = hospital.select_one('.name').text.strip()
#         address = hospital.select_one('.address').text.strip()
#         contact = hospital.select_one('.contact').text.strip()
#         hospitals.append([name, address, contact])

#     # Save the data in a CSV file
#     df = pd.DataFrame(hospitals, columns=['Name', 'Address', 'Contact'])
#     df.to_csv('hospital.csv', index=False)

# else:
#     print(f'Failed to retrieve data from {url}. Status code: {response.status_code}')



# import csv
# import requests
# from bs4 import BeautifulSoup

# url = 'https://en.wikipedia.org/wiki/List_of_hospitals_in_Metro_Manila'
# response = requests.get(url)
# soup = BeautifulSoup(response.text, 'html.parser')

# with open('hospital.csv', mode='w', newline='', encoding='utf-8') as f:
#     writer = csv.writer(f)
#     writer.writerow(['Hospital Name', 'Address', 'Contact Information'])

#     for row in soup.find_all('tr'):
#         columns = row.find_all('td')
#         if len(columns) > 0:
#             hospital_name = columns[0].get_text(strip=True)
#             address = columns[1].get_text(strip=True)
#             contact_info = columns[2].get_text(strip=True)

#             writer.writerow([hospital_name, address, contact_info])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-210-4ca9a8385a0a> in <cell line: 39>()
     45         if len(columns) > 0:
     46             hospital_name = columns[0].get_text(strip=True)
---> 47             address = columns[1].get_text(strip=True)
     48             contact_info = columns[2].get_text(strip=True)
     49

IndexError: list index out of range
```

```
# hos = p.read_csv('/content/hospital.csv')
# hos
```

Name	Address	Contact

CONCLUSION

What I learned from this Hands On Activity is how to make gathered data more readable and understandable by, combining separate datasets (concatinating part), sorting it(sortby part) and summarizing it (melting part). Having the data is not enough, what will you do with all of that data if you can't really understand what it is and what they are used for.