Module 7: Data Wrangling with Pandas

CPE 311 Computational Thinking with Python

Submitted by: **De los Reyes, Jann Moises Nyll B.**

Performed on: 20/03/2024

Submitted on: 20/03/2024

Submitted to: **Engr. Roman M. Richard**

7.1 Supplementary Activity

Using the dataset 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 separate CSV file. Combine them into a single file and store the dataframe of the FAANG data as a faang for the rest of 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.

Answer:

1. Reading each file in

```python
# To read the following data on csv file, we need to import pandas.
import pandas as pd

#Reading the aapl.csv file
df1 = pd.read_csv('/content/drive/MyDrive/data/aapl.csv')

#Display the first  5 of aapl.csv
df1.head()
```

	date	open	high	low	close	volume
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934
1	2018-01-03	169.2521	171.2337	168.6929	168.9578	29517899
2	2018-01-04	169.2619	170.1742	168.8106	169.7426	22434597
3	2018-01-05	170.1448	172.0381	169.7622	171.6751	23660018
4	2018-01-08	171.0375	172.2736	170.6255	171.0375	20567766

Next steps: View recommended plots

```python
#Reading the amzn.csv file
df2 =pd.read_csv('/content/drive/MyDrive/data/amzn.csv')

#Displaying  te first  5 of amzn.csv
df2.head()
```

	date	open	high	low	close	volume
0	2018-01-02	1172.00	1190.00	1170.51	1189.01	2694494
1	2018-01-03	1188.30	1205.49	1188.30	1204.20	3108793
2	2018-01-04	1205.00	1215.87	1204.66	1209.59	3022089
3	2018-01-05	1217.51	1229.14	1210.00	1229.14	3544743
4	2018-01-08	1236.00	1253.08	1232.03	1246.87	4279475

Next steps: View recommended plots

```
#Reading the fb.csv file
df3 =pd.read_csv('/content/drive/MyDrive/data/fb.csv')

#Displaying  te first  5 of fb.csv
df3.head()
```

	date	open	high	low	close	volume
0	2018-01-02	177.68	181.58	177.5500	181.42	18151903
1	2018-01-03	181.88	184.78	181.3300	184.67	16886563
2	2018-01-04	184.90	186.21	184.0996	184.33	13880896
3	2018-01-05	185.59	186.90	184.9300	186.85	13574535
4	2018-01-08	187.20	188.90	186.3300	188.28	17994726

Next steps: View recommended plots

```
#Reading the goog.csv file
df4 =pd.read_csv('/content/drive/MyDrive/data/goog.csv')

#Displaying  te first  5 of goog.csv
df4.head()
```

	date	open	high	low	close	volume
0	2018-01-02	1048.34	1066.94	1045.23	1065.00	1237564
1	2018-01-03	1064.31	1086.29	1063.21	1082.48	1430170
2	2018-01-04	1088.00	1093.57	1084.00	1086.40	1004605
3	2018-01-05	1094.00	1104.25	1092.00	1102.23	1279123
4	2018-01-08	1102.23	1111.27	1101.62	1106.94	1047603

Next steps: View recommended plots

```
#Reading the nflx.csv file
df5 =pd.read_csv('/content/drive/MyDrive/data/nflx.csv')

#Displaying  te first  5 of nflx.csv
df5.head()
```

	date	open	high	low	close	volume
0	2018-01-02	196.10	201.65	195.4200	201.07	10966889
1	2018-01-03	202.05	206.21	201.5000	205.05	8591369
2	2018-01-04	206.20	207.05	204.0006	205.63	6029616
3	2018-01-05	207.25	210.02	205.5900	209.99	7033240
4	2018-01-08	210.02	212.50	208.4400	212.05	5580178

Next steps: View recommended plots

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.

```
new_df1 = df1.assign(ticker ='AAPL') #adding a column called ticker for Apple's stock
new_df1.head() #display the first 5 rows
```

	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

Next steps: View recommended plots

```
new_df2= df2.assign(ticker = 'AMZN') # adding a column called ticker for Amazon's stock
new_df2.head() #display first 5 rows
```

	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

Next steps: View recommended plots

```
new_df3 = df3.assign(ticker = 'FB') #adding a column called ticker for Facebook's stock
new_df3.head() #display first 5 rows
```

	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

Next steps: View recommended plots

```
new_df4 = df4.assign(ticker = 'GOOG') #adding a column called ticker for Google's stock
new_df4.head() #display first 5 rows
```

	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

Next steps: View recommended plots

```
new_df5 = df5.assign(ticker = 'NFLX') #adding a column called ticker for Netflix's stock
new_df5.head() #display first 5 rows
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	196.10	201.65	195.4200	201.07	10966889	NFLX
1	2018-01-03	202.05	206.21	201.5000	205.05	8591369	NFLX
2	2018-01-04	206.20	207.05	204.0006	205.63	6029616	NFLX
3	2018-01-05	207.25	210.02	205.5900	209.99	7033240	NFLX
4	2018-01-08	210.02	212.50	208.4400	212.05	5580178	NFLX

Next steps: View recommended plots

3.Append them together into a single dataframe.

```
big_df = pd.concat([new_df1,new_df2,new_df3,new_df4,new_df5], axis =0 )
big_df
```

	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	242.0000	250.6500	233.6800	233.8800	9547616	NFLX
247	2018-12-26	233.9200	254.5000	231.2300	253.6700	14402735	NFLX
248	2018-12-27	250.1100	255.5900	240.1000	255.5650	12235217	NFLX
249	2018-12-28	257.9400	261.9144	249.8000	256.0800	10987286	NFLX
250	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

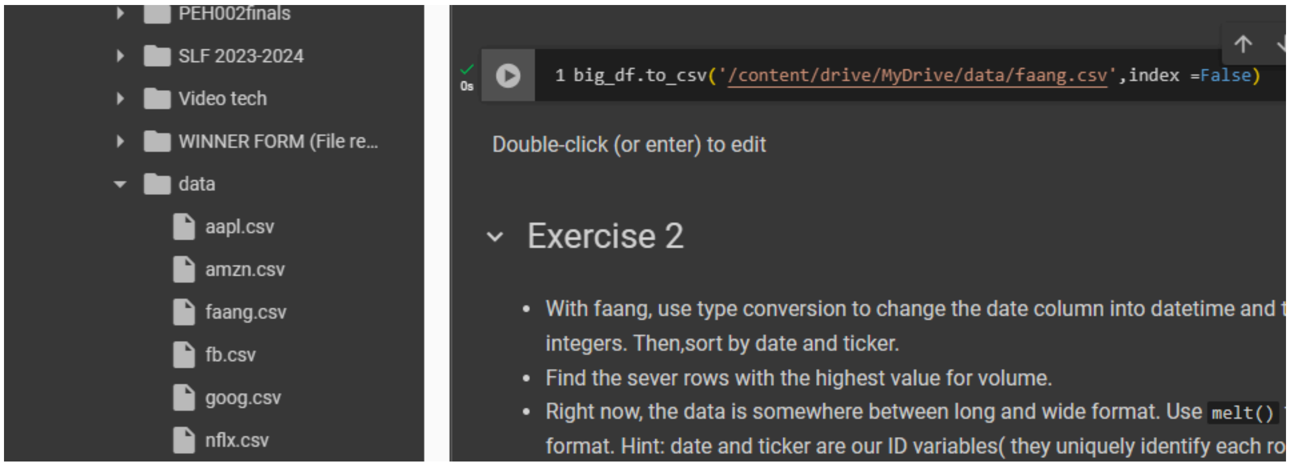
1255 rows × 7 columns

Next steps: View recommended plots

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

```
big_df.to_csv('/content/drive/MyDrive/data/faang.csv',index =False)
```

Screenshot of the faang.csv being saved in the drive.



```
big_df.to_csv('/content/drive/MyDrive/data/faang.csv',index =False)
```

Double-click (or enter) to edit

Exercise 2

- With faang, use type conversion to change the date column into datetime and t
  integers. Then,sort by date and ticker.
- Find the sever rows with the highest value for volume.
- Right now, the data is somewhere between long and wide format. Use `melt()`
  format. Hint: date and ticker are our ID variables(they uniquely identify each ro

Exercise 2

- With faang, use type conversion to change the date column into datetime and the volume column into integers. Then,sort by date and ticker.
- Find the sever 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 column for open,high,low,close and volume.

Answer:

- With faang, use type conversion to change the date column into datetime and the volume column into integers. Then,sort by date and ticker.

```
# First we need to setup the csv file to get started.

df = pd.read_csv('/content/drive/MyDrive/data/faang.csv')
df
```

	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
...	...	...	...	...	...	...	...
1250	2018-12-24	242.0000	250.6500	233.6800	233.8800	9547616	NFLX
1251	2018-12-26	233.9200	254.5000	231.2300	253.6700	14402735	NFLX
1252	2018-12-27	250.1100	255.5900	240.1000	255.5650	12235217	NFLX
1253	2018-12-28	257.9400	261.9144	249.8000	256.0800	10987286	NFLX
1254	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

We can now start the conversion, First we need to check the data type first.

```
df.dtypes
```

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

As we can see the `date` column was not currently stored as a `datetime`, we can perform the conversion using `pd.to_datetime()`

```
df.loc[:,'date'] =pd.to_datetime(df.date)
df.dtypes
```

```
<ipython-input-119-ad99edd1c048>:1: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values i
  df.loc[:,'date'] =pd.to_datetime(df.date)
date      datetime64[ns]
open             float64
high             float64
low              float64
close            float64
volume             int64
ticker            object
dtype: object
```

Lets now perform the conversion for volume, we can use astype to perform the conversion.

```
df = df.assign(volume = df.volume.astype('int'))
df.head()
```

	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

Next steps: View recommended plots

We can now perform sorting by date and ticker.

```
sorted_df = df.sort_values(by = ['date','ticker'])
#sort  the dataframe by date and and ticker from oldest to new
sorted_df
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
251	2018-01-02	1172.0000	1190.0000	1170.5100	1189.0100	2694494	AMZN
502	2018-01-02	177.6800	181.5800	177.5500	181.4200	18151903	FB
753	2018-01-02	1048.3400	1066.9400	1045.2300	1065.0000	1237564	GOOG
1004	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
501	2018-12-31	1510.8000	1520.7600	1487.0000	1501.9700	6954507	AMZN
752	2018-12-31	134.4500	134.6400	129.9500	131.0900	24625308	FB
1003	2018-12-31	1050.9600	1052.7000	1023.5900	1035.6100	1493722	GOOG
1254	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

- Find the seven rows with the highest value for volume.

```
df.nlargest(n=7,columns ='volume')
# We use nlargest() to find the highest volume on our dataframe
```

	date	open	high	low	close	volume	ticker
644	2018-07-26	174.8900	180.1300	173.7500	176.2600	169803668	FB
555	2018-03-20	167.4700	170.2000	161.9500	168.1500	129851768	FB
559	2018-03-26	160.8200	161.1000	149.0200	160.0600	126116634	FB
556	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

- 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 column for open,high,low,close and volume.

```
# Before we perform melt(). we need to setup our data.
sorted_df.dtypes
```

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

We can change the volume column type as float

```
df = df.assign(volume = df.volume.astype('float'))
df.dtypes
```

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

```
df
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934.0	AAPL
1	2018-01-03	169.2521	171.2337	168.6929	168.9578	29517899.0	AAPL
2	2018-01-04	169.2619	170.1742	168.8106	169.7426	22434597.0	AAPL
3	2018-01-05	170.1448	172.0381	169.7622	171.6751	23660018.0	AAPL
4	2018-01-08	171.0375	172.2736	170.6255	171.0375	20567766.0	AAPL
...	...	...	...	...	...	...	...
1250	2018-12-24	242.0000	250.6500	233.6800	233.8800	9547616.0	NFLX
1251	2018-12-26	233.9200	254.5000	231.2300	253.6700	14402735.0	NFLX
1252	2018-12-27	250.1100	255.5900	240.1000	255.5650	12235217.0	NFLX
1253	2018-12-28	257.9400	261.9144	249.8000	256.0800	10987286.0	NFLX
1254	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920.0	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

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

)
melted_df
```

	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

Next steps: View recommended plots

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 hospital.csv, convert the csv file into pandas dataframe. Prepare the data using the necessary preprocessing techniques.

Answer:

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.

```
import requests
from bs4 import BeautifulSoup
url = "https://www.listsclub.com/hospitals-in-japan/"
soup = BeautifulSoup(requests.get(url).text,'html')

```

```
p_tags = soup.find_all('p')
data_list = []
for p_tag in p_tags:
    data_list.append(p_tag.text)
```

```
df = pd.DataFrame(data_list, columns=['Text Content'])
```

```
df.to_excel('output.xlsx', index=False)
```

```
1. St. Luke’s International Hospital
```

Conclusion:

Using pandas for reshaping, cleaning, and sorting stock data, especially for big names like Facebook, Apple, Amazon, Netflix, and Google, is incredibly efficient. It simplifies the process and makes handling large datasets much more manageable. However, I understand that web scraping can be tricky when dealing with unstructured data. It's a common hurdle but one that can be overcome with practice and the right tools.

1

1