Module 7: Data Wrangling with Pandas

CPE311 Computational Thinking with Python

Submitted by: Valleser, Kenn Jie

Performed on: 07/04/2025

Submitted on: 07/04/2025

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 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.**

In [33]:

```
import pandas as pd

# Step 1: Read each CSV file
fb = pd.read_csv('fb.csv')
aapl = pd.read_csv('aapl.csv')
amzn = pd.read_csv('amzn.csv')
nflx = pd.read_csv('nflx.csv')
goog = pd.read_csv('goog.csv')

# Step 2: Add a "ticker" column to each dataframe
fb['ticker'] = 'FB'
aapl['ticker'] = 'AAPL'
amzn['ticker'] = 'AMZN'
nflx['ticker'] = 'NFLX'
goog['ticker'] = 'GOOG'

# Step 3: Combine them into one big dataframe
faang = pd.concat([fb, aapl, amzn, nflx, goog])

# Step 4: Save the combined dataframe to a CSV file
faang.to_csv('faang.csv', index=False)
faang
```

Out[33]:

	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

	date	open	high	low	close	volume	ticker
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

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.

In [34]:

```
faang['date'] = pd.to_datetime(faang['date'])
faang['volume'] =  pd.to_numeric(faang['volume'])
```

In [35]:

```
faang.dtypes
```

Out[35]:

	0
date	datetime64[ns]
open	float64
high	float64
low	float64
close	float64
volume	int64
ticker	object

dtype: object

In [36]:

```
faang = faang.sort_values(by=['date', 'ticker'], ascending=False)
```

In [39]:

```
faang = faang.sort_values(by='volume', ascending=False)
faang
```

Out[39]:

	date	open	high	low	close	volume	ticker
142	2018-07-26	174.8900	180.1300	173.7500	176.2600	169803668	FB

	date	open	high	low	close	volume	ticker
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
...	...	...	...	...	...	...	...
152	2018-08-09	1249.9000	1255.5400	1246.0100	1249.1000	848601	GOOG
130	2018-07-10	1156.9800	1159.5900	1149.5900	1152.8400	798412	GOOG
99	2018-05-24	1079.0000	1080.4700	1066.1500	1079.2400	766773	GOOG
226	2018-11-23	1030.0000	1037.5900	1022.4000	1023.8800	691462	GOOG
126	2018-07-03	1135.8200	1135.8200	1100.0200	1102.8900	679034	GOOG

1255 rows × 7 columns

In [40]:

```
melted = pd.melt(faang, id_vars=['date','ticker'], value_vars=['open', 'high','low','clo
se','volume'])
melted
```

Out[40]:

	date	ticker	variable	value
0	2018-07-26	FB	open	174.8900
1	2018-03-20	FB	open	167.4700
2	2018-03-26	FB	open	160.8200
3	2018-03-21	FB	open	164.8000
4	2018-09-21	AAPL	open	219.0727
...	...	...	...	...
6270	2018-08-09	GOOG	volume	848601.0000
6271	2018-07-10	GOOG	volume	798412.0000
6272	2018-05-24	GOOG	volume	766773.0000
6273	2018-11-23	GOOG	volume	691462.0000
6274	2018-07-03	GOOG	volume	679034.0000

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, hospitals.csv.
- Using the generated hospitals.csv, convert the csv file into pandas dataframe. Prepare the data using the necessary preprocessing techniques.

In [57]:

```
import requests

# New humdata JSON metadata URL
url = 'https://data.humdata.org/dataset/20e5069f-1eb8-465b-98c8-3442a62cd3f0/resource/9af
10e86-6425-4807-b6a4-333d38e25d80/download/philippines_hxl.csv'

# Make the request
response = requests.get(url)

# Check if it was successful
```

```
if response.ok:
    # Assuming the response is CSV, you might want to process the data as CSV
    with open('philippines_hxl.csv', 'wb') as file:
        file.write(response.content)
    print("File downloaded successfully.")
else:
    print(f'Request was not successful and returned code: {response.status_code}.')
```

```
File downloaded successfully.
```

In [58]:

```
df = pd.read_csv('philippines_hxl.csv')
df.drop_duplicates(inplace=True)
df
```

Out[58]:

	X	Y	osm_id	osm_type	completeness	#loc+amenity	#meta+healthcare	#loc+name	#meta+operator
0	125.620522	7.096439	11918986693	node	12.500	pharmacy	pharmacy	JFCK Enterprise	NaN
1	121.022381	14.605078	5868381611	node	18.750	pharmacy	pharmacy	TGP	NaN
2	121.022839	14.604971	4217156094	node	25.000	pharmacy	pharmacy	Mercury Drug	NaN
3	123.879073	10.298805	11843045944	node	12.500	dentist	dentist	Daclan Dental Clinic	NaN
4	123.884990	10.297961	413986571	node	12.500	pharmacy	pharmacy	NRMJ Quality Drug ,Inc.	NaN
...	...	...	...	...	...	...	...	...	...
14573	121.083454	14.572784	11914767575	node	15.625	pharmacy	pharmacy	Watsons	NaN
14574	122.583044	10.692036	5526119524	node	21.875	pharmacy	pharmacy	Watsons	NaN
14575	NaN	NaN	399035546	way	9.375	hospital	NaN	Living Hope Hospital	NaN
14576	120.602085	16.405736	5299050022	node	9.375	pharmacy	NaN	Conde Healthcare Pharmacy	NaN
14577	123.144435	9.860690	11743205069	node	12.500	pharmacy	pharmacy	Botica Gail	NaN

14546 rows × 35 columns

7.2 Conclusion:

I learned how to work with multiple datasets and clean them using pandas. First, I combined stock data for Facebook, Apple, Amazon, Netflix, and Google into one file, added a column for the ticker symbol, and saved it as a CSV file. Then, I converted the date and volume columns to the right formats and sorted the data. I also used the melt() function to make the data easier to analyze by turning it into a long format.

For the hospital data, I used web scraping to gather information like names, addresses, and contact details, and saved it in a CSV file. After loading the data into a pandas dataframe, I cleaned it by removing duplicates.