Data Wrangling with Pandas

CPE 311 Computational Thinking with Python

Submitted by: Cu, Angelo Luis C.

Performed on : 03/20/2024

Sumbitted on : 03/20/2024

Exercise 1

```
import pandas as pd
import numpy as np
# reads each file
aapl = pd.read_csv('/content/aapl.csv')
amzn = pd.read_csv('/content/amzn.csv')
fb = pd.read_csv('/content/fb.csv')
goog = pd.read_csv('/content/goog.csv')
nflx = pd.read_csv('/content/nflx.csv')
```

```
# adds a ticker column to each dataframe
aapl['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['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['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['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['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

```
# appends each dataframe to a single dataframe called faang
faang = pd.concat([aapl, amzn, fb, goog, nflx])
faang
```

	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

```
# saves the result to a csv file
faang.to_csv('/content/faang.csv', index=False)
```

Exercise 2

```
# converts the date column to datetime
faang['date'] = pd.to_datetime(faang['date'])
faang.dtypes
```

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

```
# converts the volume column to integer
faang['volume'] = faang['volume'].astype(int)
faang.dtypes
```

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

```
# sorts by date
sorted_by_date = faang.sort_values(by='date')
sorted_by_date
```

	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	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	1172.0000	1190.0000	1170.5100	1189.0100	2694494	AMZN
0	2018-01-02	196.1000	201.6500	195.4200	201.0700	10966889	NFLX
...	...	...	...	...	...	...	...
250	2018-12-31	134.4500	134.6400	129.9500	131.0900	24625308	FB
250	2018-12-31	157.8529	158.6794	155.8117	157.0663	35003466	AAPL
250	2018-12-31	1050.9600	1052.7000	1023.5900	1035.6100	1493722	GOOG
250	2018-12-31	1510.8000	1520.7600	1487.0000	1501.9700	6954507	AMZN
250	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

```
# sorts by ticker
sorted_by_ticker = faang.sort_values(by='ticker')
sorted_by_ticker
```

	date	open	high	low	close	volume	ticker
0	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	AAPL
160	2018-08-21	215.1235	215.5104	212.3699	213.3771	26159755	AAPL
161	2018-08-22	212.4443	214.6869	212.1863	213.3870	19018131	AAPL
162	2018-08-23	212.9901	215.3715	212.9405	213.8236	18883224	AAPL
163	2018-08-24	214.9250	215.2227	213.4465	214.4884	18476356	AAPL
...	...	...	...	...	...	...	...
88	2018-05-09	328.7900	331.9500	327.5100	330.3000	5633444	NFLX
89	2018-05-10	331.5000	332.0550	327.3438	329.6000	5302254	NFLX
90	2018-05-11	329.6500	331.2600	324.8700	326.4600	4589731	NFLX
77	2018-04-24	319.2168	320.2490	302.3100	307.0200	13893217	NFLX
250	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NFLX

1255 rows × 7 columns

Next steps: View recommended plots

```
# gets the 7 highest rows by volume
faang.sort_values(by='volume', ascending=False).head(7)
```

	date	open	high	low	close	volume	ticker

```
# melts the data
melted_faang = faang.melt(
    id_vars = ['date', 'ticker'],
    value_vars = ['open', 'high', 'low', 'close', 'volume'],
    var_name = 'measurement',
    value_name = 'data'
)
melted_faang
```

	date	ticker	measurement	data
0	2018-01-02	AAPL	open	1.669271e+02
1	2018-01-03	AAPL	open	1.692521e+02
2	2018-01-04	AAPL	open	1.692619e+02
3	2018-01-05	AAPL	open	1.701448e+02
4	2018-01-08	AAPL	open	1.710375e+02
...	...	...	...	...
6270	2018-12-24	NFLX	volume	9.547616e+06
6271	2018-12-26	NFLX	volume	1.440274e+07
6272	2018-12-27	NFLX	volume	1.223522e+07
6273	2018-12-28	NFLX	volume	1.098729e+07
6274	2018-12-31	NFLX	volume	1.350892e+07

6275 rows × 4 columns

Next steps: View recommended plots

Exercise 3

I attempted to get data from this website: https://sulit.ph/list-of-hospitals-in-metro-manila-with-contact-details-website-and-social-media-accounts/

However, the function

```
soup.find_all("table")
```

returns empty, so I decided to get from wikipedia, although the contact information is missing

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

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

tables = soup.find_all("table") # gets all table elements

temp = [] # store the scraped rows to a temp list

for table in tables: # for each table,
  headers = [th.get_text().strip() for th in table.find("tr").find_all("th")] # get column name

  if "Name of Hospital" in headers and "Location" in headers: # and check if name of hospital and location are the column names
      selected_columns = ["Name of Hospital", "Location"]
      rows = []
      for tr in table.find_all("tr")[1:]:
          data = [td.get_text().strip() for td in tr.find_all("td")] # get the row
          filtered_data = [data[headers.index(col)] for col in selected_columns]
          rows.append(filtered_data)

      df = pd.DataFrame(rows, columns=selected_columns) # create a dataframe to store rows
      temp.append(df)

hospitals = pd.concat(temp, ignore_index=True) # join all rows together

hospitals.to_csv('/content/hospitals.csv', index=False) # saves to a csv file

hospitals # saved as a dataframe
```

	Name of Hospital	Location
0	Caloocan City Medical Center	450 A. Mabini St., Caloocan City
1	Ospital ng Malabon	F. Sevilla Boulevard, Tañong, Malabon City
2	San Lorenzo Ruiz General Hospital	O. Reyes St., Rosita Subdivision, Santulan, Ma...
3	Gat Andres Bonifacio Memorial Medical Center	8001 Delpan St., Tondo, Manila
4	Ospital ng Tondo	Jose Abad Santos Avenue, Tondo, Manila
...	...	...
813	Salaam Hospital Foundation Inc.	Brgy. Papandayan, Marawi City
814	Cotabato Medical Specialist Hospital	Quezon Avenue, Rosary Heights, Cotabato City
815	Cotabato Puericulture Center and General Hospi...	Alonzo St., Poblacion 6, Cotabato City
816	Eros Medical Clinic and Hospital	Lawaan St., Brgy. Poblacion, Datu Paglas, Magu...
817	United Doctors Hospital of Cotabato City, Inc.	ND Avenue, Immaculada Concepcion Rosary Height...

818 rows × 2 columns

Next steps: View recommended plots

```
hospitals.dtypes
```

```
    Name of Hospital    object
    Location            object
    dtype: object
```

```
contain_nulls = hospitals[ # stores data with null values to contain_nulls
  hospitals['Location'].isnull() | hospitals['Location'].isna()\
  | pd.isnull(hospitals['Name of Hospital']) | pd.isna(hospitals['Name of Hospital'])\
  | hospitals['Name of Hospital'].isna()
]
contain_nulls.shape[0] # no null data
```

```
    0
```

```
hospitals[hospitals['Location'].isin([-np.inf, np.inf])].shape[0] # checks if there are data which has inf or -inf
```

```
    0
```

```
hospitals[hospitals['Name of Hospital'].isin([-np.inf, np.inf])].shape[0] # checks if there are data which has inf or -inf
```

```
    0
```

```
hospitals[hospitals.duplicated()].shape[0] # checks for duplicate values
```

```
    0
```

7.2 Conclusion

I have learned that the data that we could get from outside sources could be fragmented and not in the right format. By preprocessing the data we are able to turn it to something more useful.

The resulting csv files are in the github: https://github.com/a-cuc/CPE311/tree/main/Midterm/Module%207/Hands%20on%20Activity%207.1