Module 7: Data Wrangling with Pandas

CPE311 Computational Thinking with Python3

Submitted by: Montojo, Lance

Performed on: 03/20/2024

Submitted 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 Facebook, Apple, Amazon, Netflix, and Google (FAANG) stocks, but we were given each as a seperate 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.

```
# Reading each file
import pandas as pd

facebook_df = pd.read_csv('/content/fb.csv')
apple_df = pd.read_csv('/content/aapl.csv')
amazon_df = pd.read_csv('/content/amzn.csv')
netflix_df = pd.read_csv('/content/nflx.csv')
google_df = pd.read_csv('/content/goog.csv')
```

```
# Checking
google_df
```

	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
...	...	...	...	...	...	...
246	2018-12-24	973.90	1003.54	970.11	976.22	1590328
247	2018-12-26	989.01	1040.00	983.00	1039.46	2373270
248	2018-12-27	1017.15	1043.89	997.00	1043.88	2109777
249	2018-12-28	1049.62	1055.56	1033.10	1037.08	1413772
250	2018-12-31	1050.96	1052.70	1023.59	1035.61	1493722

251 rows × 6 columns

```
# 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.

# Adding new dataframe called ticker to each dataframe

facebook_df = facebook_df.assign(ticker = lambda x:'FCBK'.upper())
facebook_df
```

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

251 rows × 7 columns

```
apple_df = apple_df.assign(ticker = lambda x:'APPL'.upper())
apple_df
```

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

251 rows × 7 columns

```
amazon_df = amazon_df.assign(ticker = lambda x:'AMZN'.upper())
amazon_df
```

	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

```
netflix_df = netflix_df.assign(ticker = lambda x:'NTFX'.upper())
netflix_df
```

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

251 rows × 7 columns

```
google_df = google_df.assign(ticker = lambda x:'GOGL'.upper())
google_df
```

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

251 rows × 7 columns

```
# Append them together into a single dataframe.
FAANG = [facebook_df, apple_df, amazon_df, netflix_df, google_df]
faang = pd.concat(FAANG, ignore_index = True)
faang
```

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

1255 rows × 7 columns

```
# Save the result in a CSV file called faang.csv.
faang.to_csv('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 somewehere 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 seperate columns for open, high, low, close, and volume.

```
# 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.

# First check the current data types
faang.dtypes
```

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

```
# With faang, use type conversion to change the date column into a datetime and the volume column into integers.
# Converting data types
faang = faang.assign(
    date = pd.to_datetime(faang.date),
    volume = faang['volume'].astype('int') # Volume is already in integer but still would execute code
)

# Check if it got converted
faang.dtypes
```

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

```
# Sorting by date and ticker
sorted_faang = faang.sort_values(by = ['date', 'ticker'])
# sorted_faang = faang = faang.sort_values(by = ['date', 'ticker'], ascending = True) if you want to ascending order
# sorted_faang = faang = faang.sort_values(by = ['date', 'ticker'], ascending = False) if you want descending order

sorted_faang
```

	date	open	high	low	close	volume	ticker
502	2018-01-02	1172.0000	1190.0000	1170.5100	1189.0100	2694494	AMZN
251	2018-01-02	166.9271	169.0264	166.0442	168.9872	25555934	APPL
0	2018-01-02	177.6800	181.5800	177.5500	181.4200	18151903	FCBK
1004	2018-01-02	1048.3400	1066.9400	1045.2300	1065.0000	1237564	GOGL
753	2018-01-02	196.1000	201.6500	195.4200	201.0700	10966889	NTFX
...	...	...	...	...	...	...	...
752	2018-12-31	1510.8000	1520.7600	1487.0000	1501.9700	6954507	AMZN
501	2018-12-31	157.8529	158.6794	155.8117	157.0663	35003466	APPL
250	2018-12-31	134.4500	134.6400	129.9500	131.0900	24625308	FCBK
1254	2018-12-31	1050.9600	1052.7000	1023.5900	1035.6100	1493722	GOGL
1003	2018-12-31	260.1600	270.1001	260.0000	267.6600	13508920	NTFX

1255 rows × 7 columns

```
# Find the seven rows with the highest value for volume.
faang.nlargest(n = 7, columns = 'volume')
```

	date	open	high	low	close	volume	ticker
142	2018-07-26	174.8900	180.1300	173.7500	176.2600	169803668	FCBK
53	2018-03-20	167.4700	170.2000	161.9500	168.1500	129851768	FCBK
57	2018-03-26	160.8200	161.1000	149.0200	160.0600	126116634	FCBK
54	2018-03-21	164.8000	173.4000	163.3000	169.3900	106598834	FCBK
433	2018-09-21	219.0727	219.6482	215.6097	215.9768	96246748	APPL
496	2018-12-21	156.1901	157.4845	148.9909	150.0862	95744384	APPL
463	2018-11-02	207.9295	211.9978	203.8414	205.8755	91328654	APPL

```
# Right now, the data is somewehere 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 seperate columns for open, high, low, close, and volume.

melt_faang = faang.melt(
    id_vars = ['date', 'ticker']
)

melt_faang
```

	date	ticker	variable	value
0	2018-01-02	FCBK	open	177.68
1	2018-01-03	FCBK	open	181.88
2	2018-01-04	FCBK	open	184.90
3	2018-01-05	FCBK	open	185.59
4	2018-01-08	FCBK	open	187.20
...	...	...	...	...
6270	2018-12-24	GOGL	volume	1590328.00
6271	2018-12-26	GOGL	volume	2373270.00
6272	2018-12-27	GOGL	volume	2109777.00
6273	2018-12-28	GOGL	volume	1413772.00
6274	2018-12-31	GOGL	volume	1493722.00

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.

```
# 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 web scraping, search for the list of the hospitals, their address and contact information.
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Send a HTTP request to the website like asking for token in 7.2
website_address = "https://sulit.ph/list-of-hospitals-in-metro-manila-with-contact-details-website-and-social-media-accounts/"
response = requests.get(website_address)

# Checker if we can access the webpage
if response.status_code == 200:
  print("Success! We can access the website (status code: 200)")
else:
  print("Failed! We can not access the website (status code:", response.status_code, ")")
```

```
# Parse the HTML content of the website
soup = BeautifulSoup(response.content, "html.parser")

# Since the data is in the table, we need to find the table that contain the hospital information
table = soup.find("table")

# Let's initialize an empty list
hospital_list = []

# We then need to extract the data we seek from the table, we should also start at the third row because there is unnecessary data
# in the first to the third row
for i, row in enumerate(table.find_all("tr")[3:]): # Start from 4th index
  columns = row.find_all("td")
  # Check if row has enough columns
  if len(columns) >= 5:
        hospital_name = columns[1].text.strip()
        hospital_address = columns[0].text.strip()
        hospital_contact = columns[2].text.strip()
        hospital_email = columns[3].text.strip()
        hospital_facebook = columns[4].text.strip()
        hospital_list.append({"Name": hospital_name, "Address": hospital_address, "Contact": hospital_contact,
                              "Email": hospital_email, "Facebook": hospital_facebook})
```

```
    Success! We can access the website (status code: 200)
```

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

# Converting csv file into pandas dataframe
hospital_dataframe = pd.read_csv('/content/hospitals.csv')

# Checking the dataframe
print("Dimension of the hospital data:",  "\nRows of the data", hospital_dataframe.shape[0],  "\nColumns of the data"
      , hospital_dataframe.shape[1])
print("\nColumn names:", list(hospital_dataframe.columns))
print("\nData types:\n", hospital_dataframe.dtypes)
print("\nChecking for missing values(NaN)\n", hospital_dataframe.isnull()) # If there is missing value it would output True, otherwise False

# Checking for duplicate rows
duplicate_rows = hospital_dataframe.duplicated()
if duplicate_rows.any():
    print("\nDuplicate rows found!")
    duplicate_count = hospital_dataframe[duplicate_rows].shape[0]
    print("Number of duplicate rows:", duplicate_count)
    # Dropping duplicate rows
    hospital_dataframe.drop_duplicates(inplace=True)
    print("Duplicate rows dropped.")
else:
    print("\nNo duplicate rows found.")

print("\nSample rows of the data")
hospital_dataframe.head()
```

```
Dimension of the hospital data:
Rows of the data 96
Columns of the data 5

Column names: ['Name', 'Address', 'Contact', 'Email', 'Facebook']

Data types:
 Name        object
Address     object
Contact     object
Email       object
Facebook    object
dtype: object

Checking for missing values(NaN)
      Name  Address  Contact  Email  Facebook
0   False    False    False   True     False
1   False    False    False  False     False
2   False    False    False  False      True
3   False    False    False  False     False
4   False    False    False   True     False
..    ...      ...      ...    ...       ...
91  False    False    False   True     False
92  False    False    False   True     False
93  False    False    False   True      True
94  False    False    False  False     False
95  False    False    False  False     False

[96 rows x 5 columns]

No duplicate rows found.

Sample rows of the data
```

	Name	Address	Contact	Email	
0	Caloocan City Medical Center	Caloocan	South 5310 7925, North 8282 3397, 0943 216 6963	NaN	https://www.faceboo
1	Dr. Jose N. Rodriguez Memorial Hospital and Sa...	Caloocan	0966 549 2697, 8294 2571 to 73	http://djnrmh.doh.gov.ph/	https://www.facebook.c

Next steps: View recommended plots

```
# Creating dataframe from the extracted information
hospital_df = pd.DataFrame(hospital_list)

# Since there are missing values, we would fill them instead of leaving it blank. We would put N/A in there since it is not numeric.
hospital_df.replace('', 'N/A', inplace = True)

hospital_df
```

	Name	Address	Contact	Email	
0	Caloocan City Medical Center	Caloocan	South 5310 7925, North 8282 3397, 0943 216 6963	N/A	https://www.face
1	Dr. Jose N. Rodriguez Memorial	Caloocan	0966 549 2697, 8294	http://djnrmh.doh.gov.ph/	https://www.facebo

Next steps: View recommended plots

```
# Save the list in a new csv file, hospitals.csv.
hospital_df.to_csv("hospitals.csv", index = False)
hospital_df
```

	Name	Address	Contact	Email	
0	Caloocan City Medical Center	Caloocan	South 5310 7925, North 8282 3397, 0943 216 6963	N/A	https://www.face
1	Dr. Jose N. Rodriguez Memorial Hospital and Sa...	Caloocan	0966 549 2697, 8294 2571 to 73	http://djnrmh.doh.gov.ph/	https://www.facebo
2	MCU – FDT Medical Foundations Hospital	Caloocan	8367 2031	https://www.mcuhospital.org/	
3	Metro Balayan Medical Center	Caloocan	(043) 740 1350	http://www.metrobalayanmc.com.ph/	https://www.fac
4	Alabang Medical Center	Las Pinas	8807 8189, 8850	N/A	https://www.facebook.c

Next steps: View recommended plots

7.2 Conclusion

The imports of 'requests' and 'BeautifulSoup' libraries is useful to scrape data from web pages. After that, employing 'pandas' library to manipulate and analyze the scraped data we took from a website. The process includes sending HTTP request to website, parsing its HTML content using BeautifulSoup, and converting the extracted data into pandas DataFrame for further analysis and data cleaning. Their combination that has a process of collecting, parsing, and organizing data from web sources, provides an effective approach in data gathering tasks.