Module 7: Data Wrangling with Pandas

CPE311 Computational Thinking with Python

Name: Dejoras, Dylan James N.

Section: CPE22S3

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 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.

```
# as usual, import pandas
import pandas as pd
# assign dataframe variables to each csv file
df1 = pd.read_csv('aapl.csv')
df2 = pd.read_csv('amzn.csv')
df3 = pd.read_csv('fb.csv')
df4 = pd.read_csv('goog.csv')
df5 = pd.read_csv('nflx.csv')
```

```
# use assign() to include ticker columns accordingly
df1 = df1.assign(
    ticker = 'AAPL'
)
df2 = df2.assign(
    ticker = 'AMZN'
)
df3 = df3.assign(
    ticker = 'FB'
)
df4 = df4.assign(
    ticker = 'GOOG'
)
df5 = df5.assign(
    ticker = 'NFLX'
)
```

```
# use .append to merge the five dataframes to a single dataframe
# use ignore_index to see the number of rows
faang = df1.append([df2, df3, df4, df5], ignore_index = True)
```

```
<ipython-input-64-7cfb7d9c1497>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  faang = df1.append([df2, df3, df4, df5], ignore_index = True)
```

```
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
...	...	...	...	...	...	...	...
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

```
# create faang.csv file
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 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.

```
# use assign() for converting date to datetime and volume to int
faang = faang.assign(
    date = lambda x: pd.to_datetime(x.date),
    volume = faang.volume.astype('int')
)
```

```
# check the datatypes if the date and volume are successfully converted
faang.dtypes
```

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

```
# use sort_values(by =, to sort it depending on the column(s))
sorted_faang = faang.sort_values(by= ['date','ticker'])
```

```
sorted_faang
```

	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

```
# a code for sorting the top 7 highest volumes
faang.sort_values(by=['volume'], ascending = False).head(7)
```

	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

```
# use melt() for getting the long format of the dataframe
# also to use certain columns as the unique identifiers
melted_faang = faang.melt(
    id_vars = ['date', 'ticker'],
)
```

```
melted_faang
```

	date	ticker	variable	value
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

- 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.

```
import pandas as pd
import requests
# source url
data_url = "https://www.communitybenefitinsight.org/api/get_hospitals.php"
# by importing requesting, we could use request.get(url)
# assign it to response variable for clarification
response = requests.get(data_url)
# 200 means OK so we use an if statement to check if the url could be used
if response.status_code == 200:
    # make it a json file
    hospital_data = response.json()

    # convert json to dataframe
    hospitals = pd.DataFrame(hospital_data)

    # save the dataframe as a csv file
    hospitals.to_csv('hospitals.csv', index=False)

    print("Data has been saved to 'hospitals.csv'")
else:
    print("Failed to retrieve data from the website.")
```

```
Data has been saved to 'hospitals.csv'
```

```
# this is the content for the hospitals of the url i used
# as could be seen, it does not have a contact no.
# to cope with this shortcoming, i selected city, state, and bed counts
hospitals
```

	hospital_id	hospital_org_id	ein	name	name_cr	street_address	city	state	zip_code	fips_
0	1	1	630307951	Mizell Memorial Hospital	Mizell Memorial Hospital	702 Main Street	Opp	AL	36462	
1	2	2	630578923	St Vincents East	St Vincents East	50 Medical Park Drive East	Birmingham	AL	35235	
2	3	3	630312913	Shelby Baptist Medical Center	Shelby Baptist Medical Center	1000 First Street North	Alabaster	AL	35007	
3	4	4	630459034	Callahan Eye Foundation Hosp	Callahan Eye Foundation Hosp	1720 University Boulevard	Birmingham	AL	35233	
4	5	5	581973570	Cherokee Medical Center	Cherokee Medical Center	400 Northwood Drive	Centre	AL	35960	
...	...	...	...	...	...	...	...	...	...	
3486	3487	2647	813040663	Bsw Medical Center - Austin	Bsw Medical Center - Austin	5245 W Us 290	Austin	TX	78735	
3487	3488	2304	741109643	Ascension Seton Bastrop	Ascension Seton Bastrop	630 Highway 71 W	Bastrop	TX	78602	
3488	3489	2648	831954982	Texas Health Hospital Frisco	Texas Health Hospital Frisco	12400 N Dallas Parkway	Frisco	TX	75033	
3489	3490	2302	750800661	Methodist Midlothian Medical Center	Methodist Midlothian Medical Center	1201 E Highway 287	Midlothian	TX	76065	
3490	3491	2649	831869297	Texas Health Hospital Mansfield	Texas Health Hospital Mansfield	2300 Lone Star Road	Mansfield	TX	76063	

3491 rows × 19 columns

Next steps: View recommended plots

```
# we tend to always check the datatypes
hospitals.dtypes
```

```
hospital_id                   object
hospital_org_id               object
ein                           object
name                          object
name_cr                       object
street_address                object
city                          object
state                         object
zip_code                      object
fips_state_and_county_code    object
hospital_bed_count            object
chrch_affl_f                  object
urban_location_f              object
children_hospital_f           object
memb_counc_teach_hosps_f      object
medicare_provider_number      object
county                        object
hospital_bed_size             object
updated_dt                    object
dtype: object
```

```
# first 5 values
hospitals.head()
```

	hospital_id	hospital_org_id	ein	name	name_cr	street_address	city	state	zip_code	fips_sta
0	1	1	630307951	Mizell Memorial Hospital	Mizell Memorial Hospital	702 Main Street	Opp	AL	36462	
1	2	2	630578923	St Vincents East	St Vincents East	50 Medical Park Drive East	Birmingham	AL	35235	
2	3	3	630312913	Shelby Baptist Medical Center	Shelby Baptist Medical Center	1000 First Street North	Alabaster	AL	35007	
3	4	4	630459034	Callahan Eye Foundation Hosp	Callahan Eye Foundation Hosp	1720 University Boulevard	Birmingham	AL	35233	
4	5	5	581973570	Cherokee Medical Center	Cherokee Medical Center	400 Northwood Drive	Centre	AL	35960	

Next steps: View recommended plots

```
# as the bed count is an object datatype, we need to convert it to an integer for sorting
hospitals = hospitals.assign(
    hospital_bed_count = hospitals.hospital_bed_count.astype('int')
)
```

```
hospitals.dtypes
```

```
     hospital_id                  object
     hospital_org_id              object
     ein                          object
     name                         object
     name_cr                      object
     street_address               object
     city                         object
     state                        object
     zip_code                     object
     fips_state_and_county_code   object
     hospital_bed_count            int64
     chrch_affl_f                 object
     urban_location_f             object
     children_hospital_f          object
     memb_counc_teach_hosps_f     object
     medicare_provider_number     object
     county                       object
     hospital_bed_size            object
     updated_dt                   object
     dtype: object
```

```
# create dataframe for the five chosen columns
hospital_data = hospitals[['name', 'street_address', 'city', 'state','hospital_bed_count']]
```

```
hospital_data
```

	name	street_address	city	state	hospital_bed_count
0	Mizell Memorial Hospital	702 Main Street	Opp	AL	99
1	St Vincents East	50 Medical Park Drive East	Birmingham	AL	362
2	Shelby Baptist Medical Center	1000 First Street North	Alabaster	AL	252
3	Callahan Eye Foundation Hosp	1720 University Boulevard	Birmingham	AL	106
4	Cherokee Medical Center	400 Northwood Drive	Centre	AL	60
...	...	...	...	...	...
3486	Bsw Medical Center - Austin	5245 W Us 290	Austin	TX	16
3487	Ascension Seton Bastrop	630 Highway 71 W	Bastrop	TX	7
3488	Texas Health Hospital Frisco	12400 N Dallas Parkway	Frisco	TX	63
3489	Methodist Midlothian Medical Center	1201 E Highway 287	Midlothian	TX	46
3490	Texas Health Hospital Mansfield	2300 Lone Star Road	Mansfield	TX	59

3491 rows × 5 columns

Next steps: View recommended plots

```
# we attempt to see the hospitals who has the highest bed counts
# from this info, we would learn the maximum capacity of a certain hospital
# to minimize the dataset, i've chosen the top 10 hospitals
hospital_data.sort_values(by=['hospital_bed_count'], ascending = False).head(10)
```

	name	street_address	city	state	hospital_bed_count
510	Adventhealth Orlando	601 E Rollins St	Orlando	FL	3060
2037	New York Presbyterian Hospital	525 East 68th Street	New York	NY	2262
2986	Methodist Hospital	7700 Floyd Curl Drive	San Antonio	TX	2071
1528	Mayo Clinic Hospital Rochester	1216 Second Street Sw	Rochester	MN	2059
509	Orlando Health	52 W Underwood St	Orlando	FL	1738
1680	Barnes-Jewish Hospital	One Barnes-Jewish Hospital Plaza	St Louis	MO	1737
911	Indiana University Health Methodist Hospital	1701 North Senate Blvd	Indianapolis	IN	1733
1136	Norton Hospitals Inc	200 E Chestnut St	Louisville	KY	1730
1554	University Of Minnesota Medical Ctr	2450 Riverside Avenue	Minneapolis	MN	1700
2866	Methodist H/C Memphis Hospt	1265 Union Avenue	Memphis	TN	1593

7.2 Conclusion:

As provided in this activity, we collect data from provided csv files to merge them as a single dataframe. This is due to them having the same category for financial data as provided in the volume column. I learned how to wrangle them by sorting, checking the highest value, or melting the data to check its value in a long format. I have also learned more on how to get data from a url and then converting it into csv. In doing this hands-on activity, I am more familiarized with data collection and wrangling.