

# 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

```
1 # To read the following data on csv file, we need to import pandas.
2 import pandas as pd
3
4 #Reading the aapl.csv file
5 df1 = pd.read_csv('/content/drive/MyDrive/data/aapl.csv')
6
7 #Display the first 5 of aapl.csv
8 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](#)

```
1 #Reading the amzn.csv file
2 df2 =pd.read_csv('/content/drive/MyDrive/data/amzn.csv')
3
4 #Displaying te first 5 of amzn.csv
5 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](#)

```
1 #Reading the fb.csv file
2 df3 =pd.read_csv('/content/drive/MyDrive/data/fb.csv')
3
4 #Displaying the first 5 of fb.csv
5 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](#)

```
1 #Reading the goog.csv file
2 df4 =pd.read_csv('/content/drive/MyDrive/data/goog.csv')
3
4 #Displaying the first 5 of goog.csv
5 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](#)



```
1 #Reading the nflx.csv file
2 df5 =pd.read_csv('/content/drive/MyDrive/data/nflx.csv')
3
4 #Displaying the first 5 of nflx.csv
5 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.

```
1 new_df1 = df1.assign(ticker = 'AAPL') #adding a column called ticker for Apple's stock
2 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](#)

```
1 new_df2= df2.assign(ticker = 'AMZN') # adding a column called ticker for Amazon's stock
2 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](#)

```
1 new_df3 = df3.assign(ticker = 'FB') #adding a column called ticker for Facebook's stock
2 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](#)

```
1 new_df4 = df4.assign(ticker = 'GOOG') #adding a column called ticker for Google's stock
2 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](#)

```
1 new_df5 = df5.assign(ticker = 'NFLX') #adding a column called ticker for Netflix's stock
2 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.

```
1 big_df = pd.concat([new_df1,new_df2,new_df3,new_df4,new_df5], axis =0 )
2 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.

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

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

PEH002finals

SLF 2023-2024

Video tech

WINNER FORM (File re...

data

- aapl.csv
- amzn.csv
- faang.csv
- fb.csv
- goog.csv
- nflx.csv

1 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.

```
1 # First we need to setup the csv file to get started.
2
3 df = pd.read_csv('/content/drive/MyDrive/data/faang.csv')
4 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.

```
1 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()`

```
1 df.loc[:, 'date'] = pd.to_datetime(df.date)
2 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.

```
1 df = df.assign(volume = df.volume.astype('int'))
2 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.

```
1 sorted_df = df.sort_values(by = ['date','ticker'])
2 #sort the dataframe by date and and ticker from oldest to new
3 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.

```
1 df.nlargest(n=7,columns = 'volume')
2 # 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.

```
1 # Before we perform melt(), we need to setup our data.
2 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

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

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

```
1 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](#)

```
1 melted_df = sorted_df.melt(
2     id_vars = ['date', 'ticker'],
3     value_vars=['open', 'high', 'low', 'close', 'volume']
4 )
5 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.

```
1 import requests
2 from bs4 import BeautifulSoup
3 url = "https://www.listsclub.com/hospitals-in-japan/"
4 soup = BeautifulSoup(requests.get(url).text, 'html')
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

### ✓ 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.



