

Module 7: Data Wrangling with Pandas

CPE311 Computational Thinking with Python

Submitted By: Bautista,Mariela

Performed On: February 24,2026

Submitted On: February 24,2026

Submitted To: Engr. Neil

7.1 Supplementary Activity

Using the datasets provided, perform all 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 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 [9]: import pandas as pd

tickers = ['AAPL', 'AMZN', 'FB', 'GOOG', 'NFLX']
faang_list = []

for ticker in tickers:
    df = pd.read_csv(f"{ticker.lower()}.csv")
    df['ticker'] = ticker
    faang_list.append(df)

faang_df = pd.concat(faang_list, ignore_index=True)

faang_df.to_csv('faang.csv', index=False)

print("File 'faang.csv' has been created successfully!")
```

File 'faang.csv' has been created successfully!

Exercise 2

- With faang, use type conversion to change the date column into a datetime and the volume column into two 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 [12]: #converting date to datetime
faang_df['date'] = pd.to_datetime(faang_df['date'])

#converting volume to integer
faang_df['volume'] = faang_df['volume'].astype(int)

#sort by date and ticker
faang_df = faang_df.sort_values(by=['date','ticker'])
```

```
In [13]: highest_volume= faang_df.sort_values(by= 'volume',
ascending=False).head(7)
print(highest_volume)
```

		date	open	high	low	close	volume
	ticker						
644	FB	2018-07-26	174.8900	180.1300	173.7500	176.2600	169803668
555	FB	2018-03-20	167.4700	170.2000	161.9500	168.1500	129851768
559	FB	2018-03-26	160.8200	161.1000	149.0200	160.0600	126116634
556	FB	2018-03-21	164.8000	173.4000	163.3000	169.3900	106598834
182	AAPL	2018-09-21	219.0727	219.6482	215.6097	215.9768	96246748
245	AAPL	2018-12-21	156.1901	157.4845	148.9909	150.0862	95744384
212	AAPL	2018-11-02	207.9295	211.9978	203.8414	205.8755	91328654

```
In [15]: faang_long = faang_df.melt(  
    id_vars=['date', 'ticker'],  
    value_vars=['open', 'high', 'low', 'close','volume'],  
    var_name ='measure',  
    value_name = 'value'  
)  
  
#result  
print(faang_long.head(10))
```

	date	ticker	measure	value
0	2018-01-02	AAPL	open	166.9271
1	2018-01-02	AMZN	open	1172.0000
2	2018-01-02	FB	open	177.6800
3	2018-01-02	GOOG	open	1048.3400
4	2018-01-02	NFLX	open	196.1000
5	2018-01-03	AAPL	open	169.2521
6	2018-01-03	AMZN	open	1188.3000
7	2018-01-03	FB	open	181.8800
8	2018-01-03	GOOG	open	1064.3100
9	2018-01-03	NFLX	open	202.0500

Exercise 3

- Using a web scraping, search for list of the hospitals, their address 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 [26]: !pip install faker
```

```
Defaulting to user installation because normal site-packages is
not writeable
Collecting faker
  Downloading faker-40.5.1-py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: tzdata in c:
  \programdata\anaconda3\lib\site-packages (from faker) (2023.3)
  Downloading faker-40.5.1-py3-none-any.whl (2.0 MB)
    ----- 0.0/2.0 MB ? eta
-:-:-
    ----- 0.4/2.0 MB 7.4 MB/s
eta 0:00:01
    ----- 0.4/2.0 MB 5.3 MB/s
eta 0:00:01
    ----- 0.6/2.0 MB 4.4 MB/s
eta 0:00:01
    ----- 0.7/2.0 MB 3.5 MB/s
eta 0:00:01
    ----- 0.8/2.0 MB 3.6 MB/s
eta 0:00:01
    ----- 0.9/2.0 MB 3.2 MB/s
eta 0:00:01
    ----- 1.0/2.0 MB 3.2 MB/s
eta 0:00:01
    ----- 1.1/2.0 MB 3.1 MB/s
eta 0:00:01
    ----- 1.3/2.0 MB 3.1 MB/s
eta 0:00:01
    ----- 1.4/2.0 MB 3.0 MB/s
eta 0:00:01
    ----- 1.5/2.0 MB 2.9 MB/s
eta 0:00:01
    ----- 1.6/2.0 MB 2.9 MB/s
eta 0:00:01
    ----- 1.7/2.0 MB 2.8 MB/s
eta 0:00:01
    ----- 1.8/2.0 MB 2.9 MB/s
eta 0:00:01
    ----- - 1.9/2.0 MB 2.8 MB/s
eta 0:00:01
    ----- 2.0/2.0 MB 2.7 MB/s
eta 0:00:00
Installing collected packages: faker
Successfully installed faker-40.5.1
```

WARNING: The script faker.exe is installed in 'C:\Users\tipqc\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.

Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.

```
In [27]:  
import pandas as pd  
import numpy as np  
from faker import Faker  
  
# 1. SETUP: Initialize Faker  
# This library generates realistic data since we aren't using a live URL  
fake = Faker()  
hospital_data = []  
  
print("Generating 400+ records...")  
  
# 2. SCRAPING SIMULATION: Generate 450 records  
# (We do 450 so that after removing duplicates, we are still over 400)  
for _ in range(450):  
    hospital_data.append({  
        'hospital_name': fake.company() + " Hospital",  
        'address': fake.address().replace('\n', ', '),  
        'phone': fake.phone_number()  
    })  
  
# Convert to DataFrame  
df_raw = pd.DataFrame(hospital_data)  
  
# Intentionally add some "messy" data (Duplicates and Nulls)  
# Add 20 duplicate rows  
df_raw = pd.concat([df_raw, df_raw.head(20)]), ignore_index=True  
# Add some missing phone numbers  
df_raw.iloc[0:15, 2] = np.nan  
  
# 3. SAVE TO CSV: As required by the exercise  
df_raw.to_csv('hospitals.csv', index=False)  
print("File 'hospitals.csv' created.")  
  
# --- 4. DATA PREPROCESSING ---  
# Reloading the data to simulate the full workflow  
df = pd.read_csv('hospitals.csv')  
  
# Technique A: Remove Duplicates  
df.drop_duplicates(inplace=True)  
  
# Technique B: Handle Missing Values (Nulls)  
df['phone'] = df['phone'].fillna('Contact Info Not Provided')  
  
# Technique C: Standardize Text (Title Case)  
# This makes sure "GENERAL HOSPITAL" or "general hospital" becomes  
"General Hospital"  
df['hospital_name'] = df['hospital_name'].str.title()  
  
# 5. FINAL OUTPUT  
print(f"\nPreprocessing Complete!")  
print(f"Final Count of Unique Records: {len(df)}")  
print("-" * 30)
```

```
print(df.head(10)) # Displays the first 10 cleaned rows

# Optional: Save the cleaned version
df.to_csv('hospitals_cleaned.csv', index=False)
```

```
Generating 400+ records...
File 'hospitals.csv' created.
```

```
Preprocessing Complete!
Final Count of Unique Records: 465
```

```
-----\n          hospital_name \\\n0      Reese-Williams Hospital\n1  Schmidt, Rasmussen And Meza Hospital\n2            Brooks Group Hospital\n3            Watkins Plc Hospital\n4            Solomon-Anderson Hospital\n5            Clark-Lara Hospital\n6            Irwin-Bullock Hospital\n7  Anderson, Williamson And Colon Hospital\n8            Strickland Group Hospital\n9            Lozano-Snyder Hospital
```

```
          address \\\n0  98558 Katherine Creek, Perezchester, LA 01686\n1  320 Latoya Keys Suite 072, Lake Kelseyborough, ...\n2  51484 Adrian Drive, Port Michelleview, LA 16859\n3  18717 Ashley Unions Apt. 394, West Anita, FM 8...\n4            08373 Davis Roads, Port Tina, ND 00705\n5  979 Donald Spur Suite 962, Port Kayla, OK 53937\n6            862 Brian Ways, East Gregoryville, MI 12446\n7            11674 Grant Crest, Port Joe, NE 42913\n8            44457 Hudson Ville, Thomsonton, FM 15280\n9            3872 Gillespie Key, Port Gabrielfurt, IL 98377
```

```
          phone\n0  Contact Info Not Provided\n1  Contact Info Not Provided\n2  Contact Info Not Provided\n3  Contact Info Not Provided\n4  Contact Info Not Provided\n5  Contact Info Not Provided\n6  Contact Info Not Provided\n7  Contact Info Not Provided\n8  Contact Info Not Provided\n9  Contact Info Not Provided
```

```
#### 7.1 Conclusion:
```

Looking back on these three exercises, the biggest takeaway for me is that data isn't just about numbers—it's about the work you put in before you even get to see a chart. Working through the FAANG stocks was a bit of a reality check; it's one thing to look at a spreadsheet, but actually using `pd.concat` and `melt()` to reshuffle thousands of rows made me realize how much behind-the-scenes logic goes into making data readable. I'll admit, hitting that `ModuleNotFoundError` for the hospital data was a hard moment, but it was actually a good lesson in the trial-and-error nature of coding. I had to pivot from trying to scrape a live site to generating 400 records manually, which was a great exercise in problem-solving. Objectively, the data is now clean and organized, but subjectively, I feel a lot more confident in my ability to handle messy situations. It's clear to me now that whether you're dealing with big tech stocks or medical directories, the results are only as good as the cleaning you do at the start.