Homework 1

AUTHOR
Thomas Zwiller

Professional wrestling, while not everyone's cup of tea, is big business. What started as a carnival act has turned into a global entertainment industry. Netflix recently started showing Monday Night Raw, a program from the biggest North American wrestling company, WWE – this deal is reportedly worth $5 billion. Like any large entity, WWE is not without competition, drama, and scandal.

General Tips

This is very much a step-by-step process. Don't go crazy trying to get everything done with as few lines as possible. Read the documentation for the AlphaVantage api! Carefully explore the pages from cagematch. There isn't a need to get too fancy with anything here – just go with simple function and all should be good. Don't print comments, but use normal text for explanations.

Step 1

In the `calls` folder, you'll find 4 text files – these are transcripts from quarterly earnings calls. Read those files in (glob.glob will be very helpful here), with appropriate column names for ticker, quarter, and year columns; this should be done within a single function. Perform any data cleaning that you find necessary.

```
#importing needed packages
import glob as glob
import pandas as pd
import re

#pulling in the txt files using glob
quarterly_calls = glob.glob("/Users/TomTheIntern/Desktop/Mendoza/Mod 3/Unstructured/Homew

def report_cleaner(list_name: list, num: int):

  #using regex to pull in the the three alphabetical after the Calls/
  ticker = re.search(r"(?<=Calls/)[a-z]{3}",list_name[num])
  #then getting the next character and digit afterward as the Quarter
  quarter = re.search(r"(?<=Calls/[a-z]{3}_)..",list_name[num])
  #and finally pulling the next four characters, that should be numbers for the year
  year = re.search(r"(?<=Calls/[a-z]{3}_\w\d\_)....",list_name[num])

  #Originally this was returning the entire search, but thanks to a substack post I found

  #https://stackoverflow.com/questions/61636619/get-regex-in-python-to-return-the-matched
  ticker = ticker.group().upper()
  quarter = quarter.group().upper()
```

```
  year = int(year.group())

  #returning the ticker, quarter and year
  return ticker, quarter, year

#making a dataframe
cleaned_data = pd.DataFrame(index = range(4), columns= ['Ticker', 'Quarter', 'Year'])

#initializing i
i = 0

#using a for loop to iterate through the list of files and outputting the results to the
for i in range(len(quarterly_calls)):
  cleaned_data.loc[i, 'Ticker'], cleaned_data.loc[i, 'Quarter'], cleaned_data.loc[i, 'Yea

display(cleaned_data)
```

	Ticker	Quarter	Year
0	WWE	Q1	2023
1	TKO	Q3	2024
2	EDR	Q3	2023
3	WWE	Q2	2023

Step 2

Use the AlphaVantage api to get daily stock prices for WWE and related tickers for the last 5 years – pay attention to your data. You cannot use any AlphaVantage packages (i.e., you can only use requests to grab the data). Tell me about the general trend that you are seeing. I don't care which viz package you use, but plotly is solid and plotnine is good for ggplot2 users.

```
#importing packages
import requests
import datetime
import plotly.graph_objects as go
import plotly.io as plot

#establishing the API key
key = 'QWBNY4DDW4X220ZO'

#instead of making raw code for three seperate stocks I just made a function that does it

def Bullish (ticker: str):
  #reads in the ticker as a string
  av_link = "https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol={}&apike
  ticker, key)
```

```
  #pulling the data from the API
  av_request = requests.get(av_link)
  av_json = av_request.json()
  series_data = av_json['Time Series (Daily)']
  meta_data = av_json['Meta Data']

  #making the pandas dataframe
  av_data = pd.DataFrame.from_dict(series_data, orient='index')
  av_data['symbol'] = meta_data['2. Symbol']
  av_data.reset_index(inplace = True)

  #renaming the dataframe
  av_data = av_data.rename(columns = {'index':'date'})

  #https://www.geeksforgeeks.org/convert-the-column-type-from-string-to-datetime-format-i
  av_data['date'] = pd.to_datetime(av_data['date'])

  #https://www.geeksforgeeks.org/how-to-convert-strings-to-floats-in-pandas-dataframe/
  av_data['4. close'] = av_data['4. close'].astype(float)

  #filters it to include on the last 5 years
  av_data = av_data[(av_data['date']>datetime.datetime(year = 2020, day = 1,month = 2))]

  #returns the data frame
  return av_data

#calling the ticker function
WWE = Bullish('WWE')
TKO = Bullish('TKO')
EDR = Bullish('EDR')

#starting the plot
fig = go.Figure()

#WWE plot
fig.add_trace(go.Scatter(x = WWE['date'], y= WWE['4. close'], mode='lines', name='WWE', l

#TKO plot
fig.add_trace(go.Scatter(x= TKO['date'], y= TKO['4. close'], mode='lines', name='TKO', li

#EDR plot
fig.add_trace(go.Scatter(x= EDR['date'], y= EDR['4. close'], mode='lines', name='EDR', li

plot.show(fig)
```

Step 3

Just like every other nerdy hobby, professional wrestling draws dedicated fans. Wrestling fans often go to cagematch.net to leave reviews for matches, shows, and wrestlers. The following link contains the top 100 matches on cagematch: https://www.cagematch.net/?id=111&view=statistics

- What is the correlation between WON ratings and cagematch ratings?
- Which wrestler has the most matches in the top 100?
- Which promotion has the most matches in the top 100?
- What is each promotion's average WON rating?

```
#importing the functions I need
from bs4 import BeautifulSoup
import re
from collections import Counter
import numpy as np

#I started this chunk by defining the function that I used to count the stars for the WON
#I also worked with Nolan Santacroce and Logan Eades for this question
def One_Republic(Hollywood_Walk_of_Fame):
  '''
  First, the name: https://www.youtube.com/watch?v=hT_nvWreIhg
  Second, the input name is because it's a string of stars
```

```
  What it does: it reads in the string and adds to count for
  each star it encounters
  When it reaches the end of the stars it either
  figures out what the fraction is by indexing where the last parts of the list and adds
  Then returns count
  '''
  count = 0
  #if you didn't submit a list, you are submitting a list
  star_list = list(Hollywood_Walk_of_Fame)
  #iterates through the newly made list
  for i in star_list:
    if i == '*':
      #if it is indeed a *, add 1 to count
      count += 1
    else:
      #if it's not a star, the last and third to last index are converted to ints and div
      divisor = int(star_list[-1])
      dividend = int(star_list[-3])
      count += (dividend / divisor)
      break
  if count == 0:
    #if the count == 0 at the end of the process, NA is returned as it messed up the corr
    count = 'NA'
  return count

#link I need to scrape
link = 'https://www.cagematch.net/?id=111&view=statistics'

#pulling the html
wrastlin = requests.get(link)

#parsing it using the html.parser function
wrastlin_soup = BeautifulSoup(wrastlin.content, 'html.parser')

#found this reference that said to try tr to get all rows
#https://www.scraperapi.com/blog/python-loop-through-html-table/#:~:text=%3A%20Indicates%
all_rows = wrastlin_soup.find_all('tr')

#making an empty set list that I can append to later
fighter_rows = []

#this is where the magic happens
for fight in range(len(all_rows) - 1):
  #temporary dict
  fighter_row = {
    #pulling the fighters
    'Fighter 1': wrastlin_soup.select('.TCol a[href*="111"]')[fight].text,
    #the promoter by accessing the logomini title
    'Promoter': wrastlin_soup.select('.ImagePromotionLogoMini')[fight]['title'],
    #the rating as a float
    'Rating': float(wrastlin_soup.select('.Rating.Color9')[fight].text),
```

```
    #and the won through the starRating
    'Won': wrastlin_soup.select('.starRating')[fight].text}
    #then pushing it out to the fighter_rows list
  fighter_rows.append(fighter_row)

#making the rows into a dataframe
fighter_df = pd.DataFrame(fighter_rows)

#splitting the fighters along the vs
fighter_df[['Fighter 1', 'Fighter 2']] = fighter_df['Fighter 1'].str.split(' vs. ', n=1,

#using One_Republic to count the stars
fighter_df['Won'] = fighter_df['Won'].apply(One_Republic)

#and then replacing the NA's from One_Republic as an actual NA
fighter_df['Won'] = fighter_df['Won'].replace('NA', np.nan)

#https://stackoverflow.com/questions/12680754/split-explode-pandas-dataframe-string-entry
#got the idea to use .explode from here
fighter_list_clean = fighter_df[['Fighter 1', 'Fighter 2']].stack().explode().str.split(r

#https://stackoverflow.com/questions/2600191/how-do-i-count-the-occurrences-of-a-list-ite

#made them lower just in case there was a weird entry somewhere
#I saw WALTER in all caps and figured there was a non-zero chance
#I might encounter a weird error
fighter_list_clean = [name.lower() for name in fighter_list_clean]

#https://stackoverflow.com/questions/2600191/how-do-i-count-the-occurrences-of-a-list-ite
#and then I found a cool function that I imported that counted and returned everything as
fighter_count = Counter(fighter_list_clean)

fighter_count_df = pd.DataFrame(fighter_count.items(), columns = ['Fighters', 'Fights'])
fighter_count_df = fighter_count_df.sort_values(by = "Fights", ascending = False)
print(f'The number of fights each fighter had in the top-100:')
display(fighter_count_df)

#and if it worked once for the fighters...why not?
promoter_count = Counter(fighter_df['Promoter'])
promoter_count_df = pd.DataFrame(promoter_count.items(), columns = ['Promoter', 'Fights']
promoter_count_df = promoter_count_df.sort_values(by = 'Fights', ascending = False)
print(f'The number of fights in the top-100 for each promoter:')
display(promoter_count_df)

#and then all I had to do was use groupby to get the average of the promoter WON ratings
promoters = fighter_df.groupby('Promoter')['Won'].mean()
print(f'The average promoter WON ratings:')
display(promoters)

#and I should probably find the correlation between the rating and WON rating
```

```
ratings_corr = fighter_df['Rating'].corr(fighter_df['Won'])
print(f'The correlation for fights ratings and the WON Ratings is {round(ratings_corr, 3)
```

```
The number of fights each fighter had in the top-100:

/var/folders/dc/30zw0xgj67z2hzky0sd_pkv80000gp/T/ipykernel_59187/1303553222.py:80:
FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version.
To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in
to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
```

	Fighters	Fights
1	kenny omega	15
0	kazuchika okada	14
2	kenta kobashi	14
3	mitsuharu misawa	11
11	bryan danielson	11
...	...	...
60	scott dawson	1
61	tommaso ciampa	1
63	kaoru	1
64	kyoko inoue	1
97	juice robinson	1

98 rows × 2 columns

```
The number of fights in the top-100 for each promoter:
```

	Promoter	Fights
0	New Japan Pro Wrestling	34
4	World Wrestling Entertainment	14
3	All Elite Wrestling	13
2	All Japan Pro Wrestling	11
5	Ring Of Honor	7
1	Pro Wrestling NOAH	6
8	All Japan Women's Pro-Wrestling	4
7	World Wonder Ring Stardom	3
9	GAEA Japan	2
6	Total Nonstop Action Wrestling	1

	Promoter	Fights
10	Japanese Women Pro-Wrestling Project	1
11	Lucha Underground	1
12	DDT Pro Wrestling	1
13	Diamond Ring	1
14	World Championship Wrestling	1

```
The average promoter WON ratings:

Promoter
All Elite Wrestling                     5.519231
All Japan Pro Wrestling                 5.000000
All Japan Women's Pro-Wrestling         4.916667
DDT Pro Wrestling                            NaN
Diamond Ring                                 NaN
GAEA Japan                                   NaN
Japanese Women Pro-Wrestling Project    5.000000
Lucha Underground                            NaN
New Japan Pro Wrestling                 5.382353
Pro Wrestling NOAH                      4.791667
Ring Of Honor                           4.916667
Total Nonstop Action Wrestling          5.000000
World Championship Wrestling            5.000000
World Wonder Ring Stardom               5.250000
World Wrestling Entertainment           4.892857
Name: Won, dtype: float64

The correlation for fights ratings and the WON Ratings is 0.327.
```

- Select any single match and get the comments and ratings for that match into a data frame.

```
from IPython.display import display
#the link for the comments
ratings = 'https://www.cagematch.net/?id=111&nr=8034&page=99'

#the raw html for the ratings
ratins = requests.get(ratings)

#parsing the html
ratings_soup = BeautifulSoup(ratins.content, 'html.parser')

#getting the total comments so I can loop through it
ratings_rows = ratings_soup.find_all('div', class_='Comment')

#temporary empty list so I export to it
rating_frame = []

for rating in range(len(ratings_rows)):
  rating_row = {
```

```
    #grabbing the rating here
    'User': ratings_soup.select('.CommentHeader')[rating].text,
    #and then the raw comment
    'Comment': ratings_soup.select('.CommentContents')[rating].text}
  #and then I append the dict to the list
  rating_frame.append(rating_row)

#converting the list into a data frame
rating_df = pd.DataFrame(rating_frame)

#splitting up the rating from the comment so the frame has some extra data and looks clea
rating_df['Rating'] = rating_df['Comment'].str.extract(r'(\[\d+\.\d+\]|\[\d\.\d\])')

#then I'm pulling the date so the user column is cleaner
rating_df['Date'] = rating_df['User'].str.extract(r'(\d\d\.\d\d\.\d\d\d\d)')

#getting rid of the date from the suer column
rating_df['User'] = rating_df['User'].str.replace(r' wrote on \d\d\.\d\d\.\d\d\d\d\:' , '

#and then cleaning the the rating from the comment
rating_df['Comment'] = rating_df['Comment'].str.replace(r'(\[\d+\.\d+\]|\[\d\.\d\])', '',

display(rating_df.head(10))
```

	User	Comment	Rating	Date
0	JRMac	"The match that really got me back into pro-w...	[10.0]	09.05.2025
1	BJSunday	"Greatest Match Ever! Definitely prefer it to ...	NaN	08.05.2025
2	DaPrice1776	"It really is one of the greatest matches eve...	[10.0]	07.05.2025
3	Carsey0111	"The greatest match there has ever been, leve...	[10.0]	05.05.2025
4	gabad1234	"I am obviously not the first one to call thi...	[10.0]	04.05.2025
5	W	" "All time good match from start to finish. O...	[10.0]	02.05.2025
6	ColonelSany	"Like for many, this match has been my introd...	[10.0]	29.04.2025
7	KyonieClassics	"To me this is one of the greatest matches of...	[10.0]	17.04.2025
8	wrestlingreviewer	"The feeling of this match the first time was...	[10.0]	15.04.2025
9	Rassle Fan	"Maybe the greatest match I've ever seen. It'...	[10.0]	09.04.2025

Step 4

You can't have matches without wrestlers. The following link contains the top 100 wrestlers, according to cagematch: https://www.cagematch.net/?id=2&view=statistics

- Of the top 100, who has wrestled the most matches?
- Of the top 100, which wrestler has the best win/loss?

```
#the link that I need
link2 = 'https://www.cagematch.net/?id=2&view=statistics'

#pulling the html
best_wrastlers = requests.get(link2)

#then parsing it
wrastlers = BeautifulSoup(best_wrastlers.content, 'html.parser')

#selecting all the htmls
wrastlers_list = wrastlers.select(
  '.LayoutContent .TableContents .TCol.TColSeparator a[href*="gimmick"]')
#trimming down the links
link_list = [link['href'] for link in wrastlers_list]

#and then making them into a dataframe
wrestler_links = pd.DataFrame({'Links': link_list})

#what makes the links different? The nr=### portion of the url. If I harvcest those, I ca
wrestler_links['Code'] = wrestler_links['Links'].str.extract(r'(?<=nr=)(\d+)(?=&gimmick)'

#the link head so I can concatenate it with the tail and the unique code
link_start = 'https://www.cagematch.net/?id=2&nr='
link_end = '&page=22'

#temp list
link_list = []

#creating the links I'll need and sending the to link_list
for code in wrestler_links['Code']:
    link_code = code
    wrestler_link = link_start + link_code + link_end
    link_list.append(wrestler_link)

#temp list
top_fighters = []

#for each manufactured link, I req it, parse the html and pull the fighters name, the win
for link in link_list:
  stats_link = link
  stats_req = requests.get(stats_link)
  stats_soup = BeautifulSoup(stats_req.content, 'html.parser')
  top_fighter = {'Fighter Name': stats_soup.select('.LayoutBodyHeader .HeaderBox .TextHea
  'Win %': stats_soup.select('.InformationBoxContents')[1].text,
  'Total Fights': int(stats_soup.select('.InformationBoxContents')[0].text)}
  top_fighters.append(top_fighter)

#making the list into a dataframe
top_fighters_df = pd.DataFrame(top_fighters)
```

```
#stripping the win% so I can make it into a float
top_fighters_df['Win %'] = top_fighters_df['Win %'].str.extract(r'\((.*?)\%').astype(floa

#getting the most fights
most_fights = top_fighters_df.sort_values(by = "Total Fights", ascending  = False)

most_fights_fight = top_fighters_df.loc[top_fighters_df['Total Fights'].idxmax(), ['Total

print(f'The wrestler with the most fights in the top-100 is:')
display(most_fights.head(1))

#and the best win %
best_win_perc = top_fighters_df.sort_values(by = "Win %", ascending = False)

print(f'And the fighter in the top-100 with the best win % is')
display(best_win_perc.head(2))
```

```
The wrestler with the most fights in the top-100 is:
```

	Fighter Name	Win %	Total Fights
12	Jushin Thunder Liger	54.3	4401

```
And the fighter in the top-100 with the best win % is
```

	Fighter Name	Win %	Total Fights
29	Gene Okerlund	100.0	4
86	Antonio Inoki	79.4	3690

I did end up including two for the best win % as Okerlund was an announcer and only had four fights. Antonio Inoki was was the best wrestler who had over 10 fights.

Step 5

With all of this work out of the way, we can start getting down to strategy.

Step 5 Part 1

First, what talent should WWE pursue? Advise carefully.

To answer this question I decided to go and get the top 100 active wrestlers age, rating and promotion.

I learned a few interesting things:

The WWE's average age (in the top 100) is 48.125, which is older than I would have guessed, and older than AEW, but on the younger side of the active wrestlers.

Eight different organizations have a higher rating than the WWE's top active wrestlers, including AEW, by a slim margin.

There are a handful of freelancer wrestlers out there with high ratings who are younger than the WWE average age.

Here's the code.

```
#getting a link of the top 100 active wrestlers
active_link = 'https://www.cagematch.net/?id=2&view=statistics&page=18'

#pulling the html
active_wrastlers = requests.get(active_link)

#then parsing it
active_parsed = BeautifulSoup(active_wrastlers.content, "html.parser")

#selecting all the htmls
active_list = active_parsed.select(
  '.LayoutContent .TableContents .TCol.TColSeparator a[href*="gimmick"]')

#trimming down the links
active_link_list = [link['href'] for link in active_list]

#and then making them into a dataframe
active_links = pd.DataFrame({'Links': active_link_list})

active_links['Code'] = active_links['Links'].str.extract(r'(?<=nr=)(\d+)(?=&gimmick)')

link_start = 'https://www.cagematch.net/?id=2&nr='
link_end = '&gimmick='

#temp list
active_list = []

#creating the links I'll need and sending the to link_list
for code in active_links['Code']:
    link_code = code
    wrestler_link = link_start + link_code + link_end
    active_list.append(wrestler_link)

#temp list
active_fighters = []

#for each manufactured link, I req it, parse the html and pull the fighters name, the win
for link in active_list:
  stats_link = link
  stats_req = requests.get(stats_link)
  stats_soup = BeautifulSoup(stats_req.content, 'html.parser')
  active_fighter = {'Fighter Name': stats_soup.select('.LayoutBodyHeader .HeaderBox .Text
```

```
'Age': stats_soup.select('.InformationBoxContents')[1].text,
'Promotion': stats_soup.select('.InformationBoxTable .InformationBoxRow .InformationBoxCo
'Rating' : stats_soup.select('.LayoutRightPanelInner .RatingsBox .RatingsBoxAdjustedRatin
  active_fighters.append(active_fighter)

active_fighters_df = pd.DataFrame(active_fighters)

active_fighters_df.to_csv('active_fighters.csv', index=False)

active_fighters_df['Age'] = active_fighters_df['Age'].str.extract(r'(\d\d)')

active_hired = active_fighters_df.dropna()

active_hired['Age'] = active_hired['Age'].astype(int)
active_hired['Rating'] = active_hired['Rating'].astype(float)

averages = active_hired.groupby('Promotion')[['Age', 'Rating']].mean()

freelancers_df = active_hired[active_hired['Promotion'] == 'Freelancer']

freelancers_df = freelancers_df.sort_values(by = "Age", ascending = True)

display(freelancers_df.head(10))
```

```
/var/folders/dc/30zw0xgj67z2hzky0sd_pkv80000gp/T/ipykernel_59187/4069684975.py:56:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/dc/30zw0xgj67z2hzky0sd_pkv80000gp/T/ipykernel_59187/4069684975.py:57:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

	Fighter Name	Age	Promotion	Rating
52	VENY	26	Freelancer	9.36
70	Sean Legacy	29	Freelancer	9.29
73	Fuminori Abe	30	Freelancer	9.28

	Fighter Name	Age	Promotion	Rating
55	Takuya Nomura	31	Freelancer	9.34
46	Katsuhiko Nakajima	37	Freelancer	9.38
49	Masashi Takeda	39	Freelancer	9.36
65	Psycho Mike	39	Freelancer	9.31
29	Tetsuya Naito	42	Freelancer	9.46
19	Barb Sasaki	43	Freelancer	9.56
13	Christophe Agius	44	Freelancer	9.59

So the WWE should strategically try and pick out younger aged wrestlers with high ratings. So based on those criteria the list is:

VENY (age 26, rating of 9.35) Takuya Nomura (age 31, rating of 9.32) Psycho Mike (age 39, rating of 9.31) Christophe Agius (age 44, rating of 9.57) Katsuhiko Nakajima (age 36, rating of 9.37)

I understand that it's unlikely that the WWE can sign all of these wrestlers given that they are Freelancers, which means they likely enjoy the freedom that comes with a freelancer, but I think maybe a shortterm contract beyond a match or two could be highly beneficial. I also think signing wrestlers like Katsuhiko Nakajima might seem like a stretch since he is Japanese and unlikely to appeal to an American audience, but I do believe there is a chance he can pull in some of the Japanese audience since they should have access to Netflix.

Step 5 Part 2

Second, reconcile what you found in steps 3 and 4 with Netflix's relationship with WWE. Use the data from the following page to help make your case: https://wrestlenomics.com/tv-ratings/

I was curious about TKO's 2024 spike and what caused it and found out that WWE's last episode on USA was on Dec 30, 2024, and the inital spike occured in January of 2024, so it couldn't be the pivot to Netflix having gone well as WWE only started on Netflix in January of 2025. I also figured it was unlikely to be because of strong ratings because the ratings seem to be generally trending downward for RAW.

Instead, I think it was largely driven by the announcment of WWE's deal with Netflix, which happened on Jan 23 of 2024 and made WWE $5 billion. https://about.netflix.com/en/news/netflix-to-become-new-home-of-wwe-raw-beginning-2025

https://www.npr.org/2025/01/17/nx-s1-5230634/what-netflixs-wwe-deal-means-for-the-future-of-live-sports-tv#:~:text=WOODS%3A%20This%20is%20paret%20of,and%20the%20Women's%20World%20Cup.

Step 5 Part 3

Third, do you have any further recommendations for WWE?

I would say, purely as a consumer who I would imagine is in WWE's target demographic (20's, male, sports fan) they need to do a better job of marketing because I didn't know RAW was now on Netflix. That's doubly problematic because I have Netflix so a barrier to entry as a fan has been removed and I still had no idea the event happened.