Assignment 8

August 24, 2023

1 Assignment 8

1.0.1 Your Name Here

1.0.2 Date

1.0.3 The libraries you will use are already loaded for you below

```
[ ]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     from itertools import chain
```

1.1 Question 1

Read in the two Netflix CSV files from /Data/Netflix as pandas dataframes. Print the number of unique genres. This is not as simple as it sounds. You cannot simply find the length of `titles['genres'].unique()`. You must convert the output of that code to a list, iterate over that list and replace the following characters: `[]',`. Once you have them replace you can split the individual strings to list items and flatten the list. I have already imported the `chain()` function for you to flatten the list. Look up the documentation to see its usage. There are 19 unique genres, but I want you to write the code to find them.

```
[1]: # your code here

     net_titles = pd.read_csv('Week_8/Data/Netflix/titles.csv')
     net_credits = pd.read_csv('Week_8/Data/Netflix/credits.csv')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 3
      1 # your code here
----> 3 net_titles = pd.read_csv('Week_8/Data/Netflix/titles.csv')
      4 net_credits = pd.read_csv('Week_8/Data/Netflix/credits.csv')

NameError: name 'pd' is not defined
```

[8]:
```
genre_list = net_titles['genres'].tolist()
genre_list = str(genre_list)
genre_list = genre_list.replace('[', "")
genre_list = genre_list.replace(']', "")
genre_list = genre_list.replace("'", "")
genre_list = genre_list.replace('"', "")
genre_list = genre_list.replace(' ', "")
genre_list = genre_list.split(',') #I know this should be a loop but I couldn't␣
↪wrap my head around it
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 genre_list = net_titles['genres'].tolist()
      2 genre_list = str(genre_list)
      3 genre_list = genre_list.replace('[', "")

NameError: name 'net_titles' is not defined
```

[3]:
```
unique_genre = []
list_of_unique_genres = set(genre_list)
for genre in list_of_unique_genres:
    unique_genre.append(genre)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 2
      1 unique_genre = []
----> 2 list_of_unique_genres = set(genre_list)
      3 for genre in list_of_unique_genres:
      4     unique_genre.append(genre)

NameError: name 'genre_list' is not defined
```

[4]:
```
unique_genre.remove('')
list(chain(unique_genre))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[4], line 1
----> 1 unique_genre.remove('')
      2 unique_genre

ValueError: list.remove(x): x not in list
```

1.2 Question 2

Print the release year and the imdb score of the highest average score of all movies by year. This is trickier than it sounds. To do this you will need to aggregate the means by year. If you use the simple method you will get a pandas series. The series will need to be converted to a dataframe and the index will need to be set as a column (release year). Once you have done that you can find the numerical index with the highest average imdb score.

[5]:
```
# your code here

movies_by_year = net_titles[net_titles.type == 'MOVIE']
movies_by_year = movies_by_year.groupby(by = 'release_year').mean()
movie_highest_score = movies_by_year.sort_values('imdb_score', ascending=False)
movie_highest_score = movie_highest_score.drop(['runtime', 'seasons',␣
↪'imdb_votes', 'tmdb_popularity', 'tmdb_score'], axis=1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 3
      1 # your code here
----> 3 movies_by_year = net_titles[net_titles.type == 'MOVIE']
      4 movies_by_year = movies_by_year.groupby(by = 'release_year').mean()
      5 movie_highest_score = movies_by_year.sort_values('imdb_score',␣
↪ascending=False)

NameError: name 'net_titles' is not defined
```

1.3 Question 3

There were 208 actors in the movie with the most credited actors. What is the title of that movie? Nulls and NaN values do not count.

[6]:
```
# your code here

both_net = pd.merge(net_titles, net_credits, how= 'left', on = 'id')
count_net = both_net['title'].value_counts()
count_net.loc[count_net['title'] == 208, 'Index'].iloc[0]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 3
      1 # your code here
----> 3 both_net = pd.merge(net_titles, net_credits, how= 'left', on = 'id')
      4 count_net = both_net['title'].value_counts()
      5 count_net.loc[count_net['title'] == 208, 'Index'].iloc[0]
```

```
NameError: name 'pd' is not defined
```

1.4 Question 4

Which movie has the highest IMDB score for the actor Robert De Niro? What year was it made? Create a kdeplot (kernel density estimation to show the distribution of his IMDB movie scores.

[7]:
```
# your code here

de_nero_scores = both_net.loc[both_net['name'] == 'Robert De Niro']
print(de_nero_scores['imdb_score'].max())
de_nero_scores.query("imdb_score== 8.3")['release_year']
sns.kdeplot(x = 'imdb_score', data= de_nero_scores)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 3
      1 # your code here
----> 3 de_nero_scores = both_net.loc[both_net['name'] == 'Robert De Niro']
      4 print(de_nero_scores['imdb_score'].max())
      5 de_nero_scores.query("imdb_score== 8.3")['release_year']

NameError: name 'both_net' is not defined
```

1.5 Question 5

Create two new boolean columns in the titles dataframe that are true when the description contains war or gangster. Call these columns `war_movies` and `gangster_movies`. How many movies are there in both categories? Which category has a higher average IMDB score? Show the IMDB score kernel density estimations of both categories.

[ ]:
```
# your code here

movie = 'war'
for i in range(len(net_titles['description'])):
    string= net_titles['description'][i]
    string_lower = string.lower().replace(' ', '')
    if movie in string_lower:
        net_titles.at[i, 'war_movie'] = 'True'
    else:
        net_titles.at[i,'war_movie'] = 'False'
```

[ ]:
```
movie = 'gangster'
for i in range(len(net_titles['description'])):
    string= net_titles['description'][i]
    string_lower = string.lower().replace(' ', '')
    if movie in string_lower:
        net_titles.at[i, 'gangster_movie'] = 'True'
    else:
        net_titles.at[i,'gangster_movie'] = 'False'
```

```
[ ]: net_titles['war_movie'].value_counts()
     net_titles['gangster_movie'].value_counts()
```

```
[ ]: net_titles.groupby('war_movie')['imdb_score'].agg(np.mean)
     net_titles.groupby('gangster_movie')['imdb_score'].agg(np.mean)
```

```
[ ]: sns.set(style="darkgrid")
     df = sns.load_dataset('iris')
     fig = sns.kdeplot(net_titles.groupby('war_movie')['imdb_score'].agg(np.mean),␣
      ↪shade=True, color="r", label = 'war')
     fig = sns.kdeplot(net_titles.groupby('gangster_movie')['imdb_score'].agg(np.
      ↪mean), shade=True, color="b", label = 'gangster')
```