HW2a Pandas (Total Points - 10)

You have to submit two files for this part of the HW

1. FirstNameLastName_Hw2a.ipynb (colab notebook)
2. FirstNameLastName_Hw2a.pdf pdf file

HW: Recommendation system using Pandas Only

- Our agenda is to build simple movie recommedation system using Pandas dataframe only.
- In real-world more complex algorithms are used to build any recommendation system.
- Algorithms like content-based filtering and collaborative filtering are mostly used to build these kinds of recommendation system.
- We will not cover these topics as it is out of the scope for this course.
- To complete this assignment follow the below instructions.

Importing Libraries

```
# Import the required packages
import numpy as np
import pandas as pd
from pathlib import Path
import zipfile
```

Mount Google Drive

We will mount Google deive and specify Path to dowanload the data set

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
```

```
# Make sure you change the Path to where you want to save data
# In the code below - datasets is the folder name in my google drive
# you can change this to appropriate folder for your drive
# for example you may want to save data to BUAN6341/HW1/Data
# in this case the below code should be modified to : '/content/drive/MyDrive/BUAN6341

base_path = '/content/drive/MyDrive/Applied_ML/Class_2/Data'
```

Double-click (or enter) to edit

```
# create a POSIX path for data folder
# we can use this to navigate file system
base_folder = Path(base_path)
print(base_path)
```

```
/content/drive/MyDrive/Applied_ML/Class_2/Data
```

```
# I usually keep the compressed files in archive folder and unzip these files in data
# You can skip this step if you do not want to follow this folder structure

archive_folder = base_folder/'archive'
data_folder = base_folder/'data'
```

```
# I usually keep teh compressed files in archive folder and unzip these files in data
# You can skip this step if you do not want to follow this folder structure

# The / can join several paths or a mix of paths and strings given, atleast one of the
# paths should be an instance of class `Path` from `pathlib` library (as shown below).

archive_folder = base_folder/'archive'
data_folder = base_folder/'data'
print(archive_folder)
```

```
/content/drive/MyDrive/Applied_ML/Class_2/Data/archive
```

Data set

We will download the movie lens data set from the following URL:
https://grouplens.org/datasets/movielens/latest/

Summary about the data files:

- This dataset describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742

movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

- Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.
- The data are contained in the files `links.csv`, `movies.csv`, `ratings.csv` and `tags.csv`.
- As part of this task, we will focus only on 2 files i.e. ratings.csv and movies.csv

use wget command to get data from the url

Syntax !wget {url} -P {path_to_save_file}

```
url = 'https://files.grouplens.org/datasets/movielens/ml-latest-small.zip'
!wget {url} -P {archive_folder}
```

```
--2023-09-08 03:56:58--  https://files.grouplens.org/datasets/movielens/ml-latest
Resolving files.grouplens.org (files.grouplens.org)... 128.101.65.152
Connecting to files.grouplens.org (files.grouplens.org)|128.101.65.152|:443... co
HTTP request sent, awaiting response... 200 OK
Length: 978202 (955K) [application/zip]
Saving to: '/content/drive/MyDrive/Applied_ML/Class_2/Data/archive/ml-latest-smal

ml-latest-small.zip 100%[===================>] 955.28K  4.64MB/s    in 0.2s

2023-09-08 03:56:59 (4.64 MB/s) - '/content/drive/MyDrive/Applied_ML/Class_2/Data
```

As we can see the downloaded file is a zip file.
We will open and read the zip file using

```
with zipfile.ZipFile(file, mode)
```

here `file` is the file to open and we can specify the mode (read, write etc,) In below command we have used `'r'` to specify that we can open the file in reading mode. Finally we use namelist() method to list the content of zipped folder.

```
# this is a zipped folder
# let us first look at the content of the zipped files
# without unziping it
zipped_file = archive_folder / 'ml-latest-small.zip'
with zipfile.ZipFile(zipped_file, 'r') as f:
  print(f.namelist())
```

```
['ml-latest-small/', 'ml-latest-small/links.csv', 'ml-latest-small/tags.csv', 'ml
```

We can see that the zip file has folder : ml-latest-small. Within the folder we have the files: links.csv, movies.csv, ratings.csv and tags.csv.

We will open the file using

```
with zipfile.ZipFile(file, mode)
```

Finally we will use file.extractall to extarct all the files from zipped folder

```
file.extractall( path)
```

In the above command path is where we want to save the extracted files.

```
with zipfile.ZipFile(zipped_file, 'r') as f:
  f.extractall(path=data_folder)
```

Task1 : Create data frames using (1) movies.csv file and (2) ratings.csv file - 1 Point

```
# our file is in the folder ml-latest-small
# We can construct a path to the file by joining the parts using the special operator
#The / can join several paths or a mix of paths and strings given, atleast one of thos
# paths should be an instance of class `Path` from `pathlib` library (as shown below).

path_ratings = data_folder / 'ml-latest-small' / 'ratings.csv'
path_movies = data_folder / 'ml-latest-small' / 'movies.csv'


# create pandas dataframe using ratings.csv file
user_movie_ratings = pd.read_csv('/content/drive/MyDrive/Applied_ML/Class_2/Data/data/
user_movie_ratings
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091

```
# DO NOT WRITE ANYTHING HERE
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091
100833	610	168250	5.0	1494273047
100834	610	168252	5.0	1493846352
100835	610	170875	3.0	1493846415

100836 rows × 4 columns

```
# create pandas dataframe using movies.csv file
movie_info = pd.read_csv('/content/drive/MyDrive/Applied_ML/Class_2/Data/data/ml-lates
movie_info
```

	movieId	title	genre
0	1	Toy Story (1995)	AdventureIAnimationIChildrenIComedyIFantas
1	2	Jumanji (1995)	AdventureIChildrenIFantas
2	3	Grumpier Old Men (1995)	ComedyIRomanc
3	4	Waiting to Exhale (1995)	ComedyIDramaIRomanc
4	5	Father of the Bride Part II (1995)	Comed
...	...	...	.
9737	193581	Black Butler: Book of the Atlantic (2017)	ActionIAnimationIComedyIFantas
9738	193583	No Game No Life: Zero (2017)	AnimationIComedyIFantas

```
# DO NOT WRITE ANYTHING HERE
```

	movieId	title	genre
0	1	Toy Story (1995)	AdventureIAnimationIChildrenIComedyIFantas
1	2	Jumanji (1995)	AdventureIChildrenIFantas
2	3	Grumpier Old Men (1995)	ComedyIRomanc
3	4	Waiting to Exhale (1995)	ComedyIDramaIRomanc
4	5	Father of the Bride Part II (1995)	Comed
...	...	...	.
9737	193581	Black Butler: Book of the Atlantic (2017)	ActionIAnimationIComedyIFantas
9738	193583	No Game No Life: Zero (2017)	AnimationIComedyIFantas
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	ActionIAnimatio
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comed

9742 rows × 3 columns

Task2 : Identify top movies based on average rating and number of ratings received - 4 Points

Step 1: Get movie review stats in a new dataframe

Step 1 : Use groupby on user_movie_ratings to get count and mean of ratings for each movie. Store this information in a new dataframe : movie_ratings_stats

Double-click (or enter) to edit

```
movie_rating_stats = user_movie_ratings.groupby('movieId')['rating'].agg(['mean','cour
movie_rating_stats.head(10)
```

movieId	mean	count
1	3.920930	215
2	3.431818	110
3	3.259615	52
4	2.357143	7
5	3.071429	49
6	3.946078	102
7	3.185185	54
8	2.875000	8
9	3.125000	16
10	3.496212	132

```
# DO NOT WRITE ANYTHING HERE
```

movieId	mean	count

Step 2 : Merge new dataframe with movie_info dataframe

Merge the new dataset (movie_rating_stats) with movie_info dataset

```
movie_info = pd.merge(left=movie_info,right=movie_rating_stats, on='movieId',how="left
movie_info
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantas
1	2	Jumanji (1995)	Adventure|Children|Fantas
2	3	Grumpier Old Men (1995)	Comedy|Romanc
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romanc
4	5	Father of the Bride Part II (1995)	Comed
...	...	...	.
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantas
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantas
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animatio
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comed

9742 rows × 5 columns

Step3: rename new columns in movie_info

count should be renamed to num_ratings and mean should be renamed to avg_ratings)

Double-click (or enter) to edit

```
movie_info = movie_info.rename(columns={'count':'num_ratings','mean':'avg_ratings'})
```

Step 4: check if any column in movie_info has null values

```
# check any column in movie_info has null values
movie_info = movie_info.dropna()
print(movie_info.isnull().sum())
```

```
movieId        0
title          0
genres         0
avg_ratings    0
num_ratings    0
dtype: int64
```

```
# DO NOT WRITE ANYTHING HERE
```

```
movieId        0
title          0
genres         0
avg_ratings    0
num_ratings    0
dtype: int64
```

Step 5: Display top 10 movies based on mean ratings.

Hint : Sort by avg_ratings in descending order

```
movie_info.sort_values(by='avg_ratings',ascending=False).head(10)
```

	movieId	title	genres	avg_ratings
7656	88448	Paper Birds (Pájaros de papel) (2010)	Comedy|Drama	5.0
8107	100556	Act of Killing, The (2012)	Documentary	5.0
9083	143031	Jump In! (2007)	Comedy|Drama|Romance	5.0
9094	143511	Human (2015)	Documentary	5.0
9096	143559	L.A. Slasher (2015)	Comedy|Crime|Fantasy	5.0
4251	6201	Lady Jane (1986)	Drama|Romance	5.0
8154	102217	Bill Hicks: Revelations (1993)	Comedy	5.0
8148	102084	Justice League: Doom (2012)	Action|Animation|Fantasy	5.0
4246	6192	Open Hearts (Elsker dig for evigt) (2002)	Romance	5.0
9122	145994	Formula of Love (1984)	Comedy	5.0

```
# DO NOT WRITE ANYTHING HERE
```

	movieId	title	genres	avg_ratings
7638	88448	Paper Birds (Pájaros de papel) (2010)	Comedy|Drama	5.0
8089	100556	Act of Killing, The (2012)	Documentary	5.0
9065	143031	Jump In! (2007)	Comedy|Drama|Romance	5.0
9076	143511	Human (2015)	Documentary	5.0
9078	143559	L.A. Slasher (2015)	Comedy|Crime|Fantasy	5.0
4245	6201	Lady Jane (1986)	Drama|Romance	5.0
8136	102217	Bill Hicks: Revelations (1993)	Comedy	5.0
8130	102084	Justice League: Doom (2012)	Action|Animation|Fantasy	5.0
4240	6192	Open Hearts (Elsker dig for evigt) (2002)	Romance	5.0
9104	145994	Formula of Love (1984)	Comedy	5.0

It seems that this does not give us a good set of top movies. Most of the movies has got only one or two ratings. We cannot recommend these movies. Let us impose condition that movies should have atleast 100 ratings and then sort by mean of ratings.

Step 6: Display top 10 movies based on mean ratings with additional constraint.

Constraint: The movies should have at least 100 ratings i.e num_ratings >100 Hint: select only those movies that has more than 100 ratings and then sort by avg_ratings in descending order.

```
movie_info[movie_info['num_ratings']>100].sort_values(by='avg_ratings',ascending= Fals
```

	movieId	title	genr
277	318	Shawshank Redemption, The (1994)	CrimeIDrar
659	858	Godfather, The (1972)	CrimeIDrar
2226	2959	Fight Club (1999)	ActionICrimeIDramaIThril

```
# DO NOT WRITE ANYTHING HERE
```

	movieId	title	genr
277	318	Shawshank Redemption, The (1994)	CrimeIDrar
659	858	Godfather, The (1972)	CrimeIDrar
2224	2959	Fight Club (1999)	ActionICrimeIDramaIThril
921	1221	Godfather: Part II, The (1974)	CrimeIDrar
6298	48516	Departed, The (2006)	CrimeIDramaIThril
913	1213	Goodfellas (1990)	CrimeIDrar
6693	58559	Dark Knight, The (2008)	ActionICrimeIDramaIIM
46	50	Usual Suspects, The (1995)	CrimeIMysteryIThril
898	1197	Princess Bride, The (1987)	ActionIAdventureIComedyIFantasyIRoman
224	260	Star Wars: Episode IV - A New Hope (1977)	ActionIAdventureISci

- We have fetched top movies to be recommended based on their average rating and rated by more than 100 users.
- But these recommendations are based only on average ratings.
- However, users might have watched some movies and they may like other similar movies.
- So, we'll try to find the relations between movies and recommend those movies that are highly related with movies users have allready watched.
- To do so, we'll calculate correlation for each movie with other movies.
- Correlation tell us about the direction of the relationship, and the degree (strength) of the relationship between two variables. High correlation value indicates variables are highly related to each other.
- To find correlation between each movies, first we will create pivot table. In this pivot table each column will be a movie (since we want to find correlation between movies) and row will be a user. The values will be rating given by a user to a movie.

Task3: Find top ten similar movies to a given movie - 5 points

Step 1: Create a Pivot Table

1. Create a matrix that has the user ids on one axis (rows) and the movie ids on another axis (columns).
2. Each cell will then consist of the rating the user gave to that movie.

(Note there will be a lot of NaN values, because users have not rated all the movies)

```
# Create a pivot table (based on user_movie_ratings datafarne)
# rows should have user_id
# columns should have movie id
# values should be ratings
```

```
movie_pivot = user_movie_ratings.pivot_table(index='userId',columns='movieId',values='
movie_pivot.head()
```

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	1
userId														
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

5 rows × 9724 columns

```
# DO NOT WRITE ANYTHING HERE
```

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	1
userId														

```
movie_pivot.shape
```

```
(610, 9724)
```

3 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ... NaN NaN

Step 2: Get movie id of a particular movie

Get the movie_id of movie 'Shawshank Redemption, The (1994)' from movie_info table

```
# given a movie name, get movieId from movie_info table
# Hint : use loc method, you will have to use condition for the row (get row where tit
# for column use column name
# then use .values attribute to get the numpy array
# finally use .item() to convert numpy array to a python number'
movie_id = movie_info.loc[movie_info['title'] == 'Shawshank Redemption, The (1994)','r
movie_id = movie_id.item()
movie_id
```

```
318
```

```
# DO NOT WRITE ANYTHING HERE
```

```
318
```

Step 3: Get the column from pivot table corresponding to the focal movie

Hint use loc method - we have extracted columns name (movie_id) above and we need all the rows for this column from pivot table created in step 1.

```
movie_ratings = movie_pivot.loc[:, movie_id]
movie_ratings
```

```
userId
1      NaN
2      3.0
3      NaN
4      NaN
5      3.0
      ...
606    3.5
```

```
607    5.0
608    4.5
609    4.0
610    3.0
Name: 318, Length: 610, dtype: float64
```

```
# DO NOT WRITE ANYTHING HERE
```

```
userId
1      NaN
2      3.0
3      NaN
4      NaN
5      3.0
      ...
606    3.5
607    5.0
608    4.5
609    4.0
610    3.0
Name: 318, Length: 610, dtype: float64
```

Step 4: Get correlation of the selected movie with all movies

Hint :Use corrwith() method to get correlation movie with all movies. You will need to ue of pivot table (created in step 1) and movie column created in step 3 in corrwith() method.

Correlation tell us about the direction of the relationship, and the degree (strength) of the relationship between two variables. High correlation value indicates variables are highly related to each other.

In our case, the correlation between two movies will be higher if they have received similar ratings from multiple users.

```
# Use corrwith() method to get correlation of the slected movie with the given movie

movie_corr = movie_pivot.corrwith(movie_ratings)
movie_corr
```

```
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2845: RuntimeW
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2704: RuntimeW
  c *= np.true_divide(1, fact)
movieId
1         0.174984
2         0.097461
3         0.466380
4         0.644380
5         0.138314
```

```
              ...
193581         NaN
193583         NaN
193585         NaN
193587         NaN
193609         NaN
Length: 9724, dtype: float64
```

```
# DO NOT WRITE ANYTHING HERE
```

```
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2683: RuntimeW
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2542: RuntimeW
  c *= np.true_divide(1, fact)
movieId
1         0.174984
2         0.097461
3         0.466380
4         0.644380
5         0.138314
              ...
193581         NaN
193583         NaN
193585         NaN
193587         NaN
193609         NaN
Length: 9724, dtype: float64
```

Step 5: Create Data frame of correlations and clean datfarme

Create Dataframe from movie correlations created in previous step. Remove Null values from data frame.

```
# create a DataFrame having the correlation values
movie_corr_data = pd.DataFrame({ 'movieId' : movie_corr.index, 'Correlation' : movie_c
# Drop the NA values, make sure to use inplace = True
movie_corr_data.dropna(inplace = True)
# Display top 5 values using head() method
movie_corr_data.head()
```

	Correlation
movieId	

```
# DO NOT WRITE ANYTHING HERE
```

	Correlation
movieId	
1	0.174984
2	0.097461
3	0.466380
4	0.644380
5	0.138314

```
## Merge Movie_corr data with movie_info data using movieID column
movie_corr_with_title = pd.merge(left=movie_corr_data,right=movie_info,on='movieId',ho

movie_corr_with_title
```

	movieId	Correlation	title	
0	1	0.174984	Toy Story (1995)	AdventureIAnimationIChildrenICome
1	2	0.097461	Jumanji (1995)	AdventureIChildr
2	3	0.466380	Grumpier Old Men (1995)	Comed
3	4	0.644380	Waiting to Exhale (1995)	ComedyIDram
4	5	0.138314	Father of the Bride Part II (1995)	
...	...	...	...	
4780	185029	-0.991241	A Quiet Place (2018)	DramaIHo
4781	185135	-1.000000	Sherlock - A Study in Pink (2010)	
4782	187593	-0.004544	Deadpool 2 (2018)	ActionICor
4783	187595	0.207514	Solo: A Star Wars Story (2018)	ActionIAdventureIChi
4784	188301	-1.000000	Ant-Man and the Wasp (2018)	ActionIAdventureIComedyIFa

4785 rows × 6 columns

```
# DO NOT WRITE ANYTHING HERE
```

	movieId	Correlation	title	
0	1	0.174984	Toy Story (1995)	AdventureIAnimationIChildrenICome
1	2	0.097461	Jumanji (1995)	AdventureIChildr
2	3	0.466380	Grumpier Old Men (1995)	Comed
3	4	0.644380	Waiting to Exhale (1995)	ComedyIDram
4	5	0.138314	Father of the Bride Part II (1995)	
...	...	...	...	
4780	185029	-0.991241	A Quiet Place (2018)	DramaIHc
4781	185135	-1.000000	Sherlock - A Study in Pink (2010)	
4782	187593	-0.004544	Deadpool 2 (2018)	ActionICor
4783	187595	0.207514	Solo: A Star Wars Story (2018)	ActionIAdventureIChi
4784	188301	-1.000000	Ant-Man and the Wasp (2018)	ActionIAdventureIComedyIFar

4785 rows × 6 columns

Step 6: Sort the above dataframe in descending order

Sort the above data frame in descending order based on values in the correlation column. Display top ten results.

```
# Sort values in descending order
# Mention the column name to sort
# display top 10 rows
movie_corr_with_title.sort_values(by = 'Correlation', ascending= False).head(10)
```

	movieId	Correlation	title	genres
3760	55080	1.0	Brave One, The (2007)	Crime|Drama|Thriller

```
# DO NOT WRITE ANYTHING HERE
```

	movieId	Correlation	title	genres
3760	55080	1.0	Brave One, The (2007)	Crime|Drama|Thriller
3155	8656	1.0	Short Film About Killing, A (Krótki film o zab...	Crime|Drama
4145	80166	1.0	Switch, The (2010)	Comedy|Romance
2335	4833	1.0	Changeling, The (1980)	Horror|Mystery|Thriller
3178	8835	1.0	Paparazzi (2004)	Drama|Thriller
4153	80846	1.0	Devil (2010)	Horror|Mystery|Thriller
2672	6013	1.0	Kangaroo Jack (2003)	Action|Comedy
2336	4835	1.0	Coal Miner's Daughter (1980)	Drama
4169	81819	1.0	Biutiful (2010)	Drama
2675	6022	1.0	American Me (1992)	Drama

The top ten movies dos not seem to be related to focal movie "Shawshank Redemption, The (1994)". The movie has highest correlation with movies that has very few ratings. The correlations based on movies that has very few ratings are not very reliable. For Example, If a movie Z is rated by only one user and the same user has also rated the focal movie "Shawshank Redemption, The (1994)". Let us assume that user has given a rating of 5 to both the movies. The correlation between these two movies will be very high. However the correlation is based on preference of only one user and hence might not be very reliable. To overcome this, we will use only those movies that have atleast a minimum number of ratings.

```
# sleect only those movies from movie_corr_with_title dataframe that has more than 10(
movie_corr_with_title = movie_corr_with_title[movie_corr_with_title['num_ratings']> 1(
```

```
# Sort the above dataframe using Correlation column and get top ten rows using head()
# the output should be stored in a new datafrme : top_ten_recommendations
top_ten_recommendations = movie_corr_with_title.sort_values(by = 'Correlation', ascenc
```

```
top_ten_recommendations
```

	movieId	Correlation	title	
206	318	1.000000	Shawshank Redemption, The (1994)	
234	357	0.446212	Four Weddings and a Funeral (1994)	
341	527	0.402202	Schindler's List (1993)	
42	50	0.394294	Usual Suspects, The (1995)	C
2382	4963	0.391546	Ocean's Eleven (2001)	
1649	3147	0.382818	Green Mile, The (1999)	
4128	79132	0.377839	Inception (2010)	Action|Crime|Drama|Mystery
2660	5989	0.356612	Catch Me If You Can (2002)	
643	1193	0.354215	One Flew Over the Cuckoo's Nest (1975)	
669	1221	0.349872	Godfather: Part II, The (1974)	

```
# DO NOT WRITE ANYTHING HERE
```

	movieId	Correlation	title	genres	avg_ratings	num
206	318	1.000000	Shawshank Redemption, The (1994)	Crime|Drama	4.429022	
234	357	0.446212	Four Weddings and a Funeral (1994)	Comedy|Romance	3.519417	
341	527	0.402202	Schindler's List (1993)	Drama|War	4.225000	
42	50	0.394294	Usual Suspects, The (1995)	Crime|Mystery|Thriller	4.237745	
2382	4963	0.391546	Ocean's Eleven	Crime|Thriller	3.844538