MovieLens Case Study

The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. The data is widely used for collaborative filtering and other filtering solutions. However, we will be using this data to act as a means to demonstrate our skill in using Python to “play” with data.

Datasets Information:

- **ratings.csv:** It contains information on ratings given by the users to a particular movie. Columns: user id, movie id, rating, timestamp
- **movie.csv:** File contains information related to the movies and its genre. Columns: movie id, movie title, release date, unknown, Action, Adventure, Animation, Children’s, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western
- **user.csv:** It contains information of the users who have rated the movies. Columns: user id, age, gender, occupation, zip code

Objective:

`To extract insights from the dataset`

Learning Outcomes:

`Use of Pandas Functions - shape, describe, groupby, merge etc.`

Domain

`Internet and Entertainment`

Note that the case study will need you to apply the concepts of groupby and merging extensively.

1. Import the necessary packages

In [1]:
```
import pandas as pd
import numpy as np
```

2. Read all the three datasets

In [2]:
```
# Reading datasets by using read_csv from pandas package
ratings = pd.read_csv("ratings.csv")
movie = pd.read_csv("movie.csv")
user = pd.read_csv("user.csv")
```

3. View the first 5 rows of all the datasets.

`Note that you will need to do it for all the three datasets seperately`

In [3]:
```
ratings.head(5)
```

Out[3]:

	user id	movie id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

In [4]:
```
movie.head(5)
```

Out[4]:

	movie id	movie title	release date	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	...	Fantasy	Film-Noir	Horror	Musical	My
0	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	0	
1	2	GoldenEye	1-Jan-95	1	1	0	0	0	0	0	...	0	0	0	0	

2	3	Four Rooms	1-Jan-95	0	0	0	0	0	0	0	...	0	0	0	0
3	4	Get Shorty	1-Jan-95	1	0	0	0	1	0	0	...	0	0	0	0
4	5	Copycat	1-Jan-95	0	0	0	0	0	1	0	...	0	0	0	0

5 rows × 21 columns

In [5]:
```
user.head(5)
```

Out[5]:

	user id	age	gender	occupation	zip code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

4. Understand the shape of all the datasets.

`Note that you will need to do it for all the three datasets seperately`

In [6]:
```
# ratings
ratings.shape
```

Out[6]:
```
(100000, 4)
```

Observation: There are 100000 rows and 4 columns in the ratings dataset

In [7]:
```
# user
user.shape
```

Out[7]:
```
(943, 5)
```

Observation: There are 943 rows and 5 columns in the user dataset

In [8]:
```
# movie
movie.shape
```

Out[8]:
```
(1680, 21)
```

Observation: There are 1680 rows and 21 columns in the movie dataset

5. Check the data types of the columns for all the datasets.

`Note that you will need to do it for all the three datasets seperately`

In [9]:
```
# ratings
# We use dataframe.dtypes to get the data types of each column
ratings.dtypes
```

Out[9]:
```
user id      int64
movie id     int64
rating       int64
timestamp    int64
dtype: object
```

Observation: All columns have integer data type

In [10]:
```
# user
user.dtypes
```

Out[10]:
```
user id        int64
age            int64
gender        object
occupation    object
zip code      object
dtype: object
```

Observations:

1. user id and age columns are of integer data types
2. gender, occupation and zip code columns are of string data type

In [11]:
```
# movie
movie.dtypes
```

Out[11]:
```
movie id         int64
movie title     object
release date    object
Action           int64
Adventure        int64
Animation        int64
Childrens        int64
Comedy           int64
Crime            int64
Documentary      int64
Drama            int64
Fantasy          int64
Film-Noir        int64
Horror           int64
Musical          int64
Mystery          int64
Romance          int64
Sci-Fi           int64
Thriller         int64
War              int64
Western          int64
dtype: object
```

Observation:

1. movie title and release date are of string data type
2. movie id and all genres are of interger data type

6. Give a statistical summary for all the datasets.

`Note that you will need to do it for all the three datasets seperately`

In [12]:
```
# ratings
ratings.describe()
```

Out[12]:

	user id	movie id	rating	timestamp
count	100000.00000	100000.000000	100000.000000	1.000000e+05
mean	462.48475	425.530130	3.529860	8.835289e+08
std	266.61442	330.798356	1.125674	5.343856e+06
min	1.00000	1.000000	1.000000	8.747247e+08
25%	254.00000	175.000000	3.000000	8.794487e+08
50%	447.00000	322.000000	4.000000	8.828269e+08
75%	682.00000	631.000000	4.000000	8.882600e+08
max	943.00000	1682.000000	5.000000	8.932866e+08

In [13]:
```
ratings.median()
```

Out[13]:
```
user id            447.0
movie id           322.0
rating               4.0
timestamp    882826944.0
dtype: float64
```

Observation: Mean and median user ratings are 3.53 & 4.00 respectively

In [14]:
```
# user
user.describe()
```

Out[14]:

	user id	age
count	943.000000	943.000000
mean	472.000000	34.051962
std	272.364951	12.192740
min	1.000000	7.000000
25%	236.500000	25.000000
50%	472.000000	31.000000
75%	707.500000	43.000000
max	943.000000	73.000000

Observation: The average age of all the users is 34 years while the range lies between 7 to 73 years.

In [15]:
```
# movie
movie.describe()
```

Out[15]:

	movie id	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drama	Fantasy
count	1680.000000	1680.000000	1680.000000	1680.000000	1680.000000	1680.000000	1680.000000	1680.000000	1680.000000	1680.000000 16
mean	841.525595	0.149405	0.080357	0.025000	0.072619	0.300595	0.064881	0.029762	0.431548	0.013095
std	485.609591	0.356593	0.271926	0.156171	0.259587	0.458653	0.246389	0.169980	0.495440	0.113717
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	421.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	841.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1261.250000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000
max	1682.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Observation: The genres should be in categorical format and not in the numeric because it is of binary class

7. Find the number of movies per genre using the movie data

In [16]:
```
# Getting all the column names
movie.columns
```

Out[16]:
```
Index(['movie id', 'movie title', 'release date', 'Action', 'Adventure',
       'Animation', 'Childrens', 'Comedy', 'Crime', 'Documentary', 'Drama',
       'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance',
       'Sci-Fi', 'Thriller', 'War', 'Western'],
      dtype='object')
```

In [17]:
```
df_genres = movie.drop(['movie id', 'movie title', 'release date'], axis=1)
```

In [18]:
```
df_genres.sum()
```

Out[18]:
```
Action         251
Adventure      135
Animation       42
Childrens      122
```

```
Comedy         505
Crime          109
Documentary     50
Drama          725
Fantasy         22
Film-Noir       24
Horror          92
Musical         56
Mystery         61
Romance        247
Sci-Fi         101
Thriller       251
War             71
Western         27
dtype: int64
```

In [19]:
```python
df_genres["sum"] = df_genres.sum(axis=1)
```

In [20]:
```python
df_genres["sum"]
```

Out[20]:
```
0       3
1       3
2       1
3       3
4       3
       ..
1675    1
1676    2
1677    2
1678    1
1679    1
Name: sum, Length: 1680, dtype: int64
```

In [ ]:
```python

```

In [21]:
```python
# Taking all the genre columns and finding the sum for every column
movie[[ 'Action',
       'Adventure', 'Animation', 'Childrens', 'Comedy', 'Crime', 'Documentary',
       'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery',
       'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']].sum()
```

Out[21]:
```
Action         251
Adventure      135
Animation       42
Childrens      122
Comedy         505
Crime          109
Documentary     50
Drama          725
Fantasy         22
Film-Noir       24
Horror          92
Musical         56
Mystery         61
Romance        247
Sci-Fi         101
Thriller       251
War             71
Western         27
dtype: int64
```

In [22]:
```python
# Alternatively, we can also loc function
movie.loc[:,'Action':'Western'].sum()
```

Out[22]:
```
Action         251
Adventure      135
Animation       42
Childrens      122
Comedy         505
Crime          109
Documentary     50
Drama          725
```

```
Fantasy         22
Film-Noir       24
Horror          92
Musical         56
Mystery         61
Romance        247
Sci-Fi         101
Thriller       251
War             71
Western         27
dtype: int64
```

In [23]:

```
# Sorting the movies across genres
number = movie.loc[:,'Action':'Western'].sum()
number.sort_values(ascending = False)
```

Out[23]:

```
Drama          725
Comedy         505
Action         251
Thriller       251
Romance        247
Adventure      135
Childrens      122
Crime          109
Sci-Fi         101
Horror          92
War             71
Mystery         61
Musical         56
Documentary     50
Animation       42
Western         27
Film-Noir       24
Fantasy         22
dtype: int64
```

Observations:

1. Drama and Comedy are the most common movie genre.
2. Clearly, there are some movies that have more than one genre.

8. Find the movies that have more than one genre

`Hint: use sum on the axis = 1`

In [24]:

```
# Checking column names
movie.columns
```

Out[24]:

```
Index(['movie id', 'movie title', 'release date', 'Action', 'Adventure',
       'Animation', 'Childrens', 'Comedy', 'Crime', 'Documentary', 'Drama',
       'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance',
       'Sci-Fi', 'Thriller', 'War', 'Western'],
      dtype='object')
```

In [25]:

```
new_movie = movie[['movie id', 'movie title']].copy()
```

In [26]:

```
new_movie.loc[:, "Number of Genres"] = movie.loc[:, 'Action':'Western'].sum(axis=1)
```

In [27]:

```
# Filtering movies that have more than 1 genres
multi_genre_movies = new_movie[new_movie['Number of Genres'] > 1]
print(multi_genre_movies)
```

```
      movie id          movie title  Number of Genres
0            1            Toy Story                 3
1            2            GoldenEye                 3
3            4           Get Shorty                 3
4            5              Copycat                 3
6            7       Twelve Monkeys                 2
...        ...                  ...               ...
1666      1669   MURDER and murder                  3
```

```
1667       1670             Tainted                  2
1670       1673              Mirage                  2
1676       1679           B. Monkey                  2
1677       1680       Sliding Doors                  2

[849 rows x 3 columns]
```

Observation: 849 movies have more than one genre.

9. Find the top 25 movies according to average ratings such that each movie has number of ratings more than 100

Hint :

1. First find the movies that have more than 100 ratings(use groupby and count). Extract the movie id in a list.
2. Find the average rating of all the movies and sort them in the descending order.
3. Use isin(list obtained from 1) to filter out the movies which have more than 100 ratings.
4. You will have to use the .merge() function to get the movie titles.

Note: This question will need you to research about groupby and apply your findings. You can find more on groupby on https://realpython.com/pandas-groupby/.

In [28]:

```python
# Merging ratings dataset with movie dataset
df_merge = movie.merge(ratings, on = 'movie id', how = 'inner')
df_merge.head()
```

Out[28]:

	movie id	movie title	release date	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	...	Musical	Mystery	Romance	Sci-Fi	Thrille
0	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	0	
1	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	0	
2	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	0	
3	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	0	
4	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	0	

5 rows × 24 columns

In [29]:

```python
# Checking the dimensions of the merged dataframe
df_merge.shape
```

Out[29]: (99990, 24)

In [30]:

```python
# Finding the count of ratings for each movie using groupby() and count()
# reset_index() is used to shift movie title from being the dataframe’s (movie_count’s) index to
# being just a normal column
movie_count = df_merge.groupby(['movie title'])['rating'].count().reset_index()
movie_count.head()
```

Out[30]:

	movie title	rating
0	'Til There Was You	9
1	1-900	5
2	101 Dalmatians	109
3	12 Angry Men	125
4	187	41

In [31]:

```python
# Extracting the movie titles that have more than 100 ratings
movie_100 = movie_count[movie_count['rating']>100]['movie title']
movie_100.head()
```

```
Out[31]: 2            101 Dalmatians
3              12 Angry Men
7       2001: A Space Odyssey
15           Absolute Power
16                Abyss, The
Name: movie title, dtype: object
```

```
In [32]: len(movie_100)
```

```
Out[32]: 334
```

```
In [33]: # Finding average ratings for each movie and sorting them out in descending order,
# using groupby() and sort_values() on merged data frame
avg_rating = df_merge.groupby(['movie title'])['rating'].mean().sort_values(ascending=False).reset_index()
avg_rating
```

Out[33]:

	movie title	rating
0	Great Day in Harlem, A	5.0
1	Prefontaine	5.0
2	Someone Else's America	5.0
3	Entertaining Angels: The Dorothy Day Story	5.0
4	Marlene Dietrich: Shadow and Light (	5.0
...	...	...
1652	Babyfever	1.0
1653	Lashou shentan	1.0
1654	Shadows (Cienie)	1.0
1655	Shadow of Angels (Schatten der Engel)	1.0
1656	Power 98	1.0

1657 rows × 2 columns

```
In [ ]:
```

```
In [34]: # Extracting movie titles that have more than 100 ratings using movie titles in movie_100 and isin() function
# Displaying top 25 rows only
avg_rating[avg_rating['movie title'].isin(movie_100)].head(25)
```

Out[34]:

	movie title	rating
15	Close Shave, A	4.491071
16	Schindler's List	4.466443
17	Wrong Trousers, The	4.466102
18	Casablanca	4.456790
20	Shawshank Redemption, The	4.445230
21	Rear Window	4.387560
22	Usual Suspects, The	4.385768
23	Star Wars	4.358491
24	12 Angry Men	4.344000
28	Citizen Kane	4.292929
30	To Kill a Mockingbird	4.292237
31	One Flew Over the Cuckoo's Nest	4.291667
32	Silence of the Lambs, The	4.289744
33	North by Northwest	4.284916
34	Godfather, The	4.283293
35	Secrets & Lies	4.265432
36	Good Will Hunting	4.262626
37	Manchurian Candidate, The	4.259542
38	Dr. Strangelove or: How I Learned to Stop Worr...	4.252577

39	Raiders of the Lost Ark	4.252381
40	Vertigo	4.251397
44	Titanic	4.245714
45	Lawrence of Arabia	4.231214
47	Maltese Falcon, The	4.210145
48	Empire Strikes Back, The	4.204360

10. See gender distribution across different genres check for the validity of the below statements

- Men watch more drama than women
- Women watch more Sci-Fi than men
- Men watch more Romance than women

compare the percentages

1. There is no need to conduct statistical tests around this. Just **compare the percentages** and comment on the validity of the above statements.
2. you might want ot use the .sum(), .div() function here.
3. Use number of ratings to validate the numbers. For example, if out of 4000 ratings received by women, 3000 are for drama, we will assume that 75% of the women watch drama.

In [35]:
```
# Merging user dataset with movie and ratings(already merged : df_merge) dataset
df_merge_all = df_merge.merge(user, on = 'user id', how = 'inner')
```

In [36]:
```
df_merge_all
```

Out[36]:

	movie id	movie title	release date	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	...	Thriller	War	Western	user id
0	1	Toy Story	1-Jan-95	0	0	1	1	1	0	0	...	0	0	0	308
1	4	Get Shorty	1-Jan-95	1	0	0	0	1	0	0	...	0	0	0	308
2	5	Copycat	1-Jan-95	0	0	0	0	0	1	0	...	1	0	0	308
3	7	Twelve Monkeys	1-Jan-95	0	0	0	0	0	0	0	...	0	0	0	308
4	8	Babe	1-Jan-95	0	0	0	1	1	0	0	...	0	0	0	308
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99985	748	Saint, The	14-Mar-97	1	0	0	0	0	0	0	...	1	0	0	729
99986	751	Tomorrow Never Dies	1-Jan-97	1	0	0	0	0	0	0	...	1	0	0	729
99987	879	Peacemaker, The	1-Jan-97	1	0	0	0	0	0	0	...	1	1	0	729
99988	894	Home Alone 3	1-Jan-97	0	0	0	1	1	0	0	...	0	0	0	729
99989	901	Mr. Magoo	25-Dec-97	0	0	0	0	1	0	0	...	0	0	0	729

99990 rows × 28 columns

In [37]:
```
# Group by gender and aggregate with sum, selecting all the genre columns
Genre_by_gender = df_merge_all.groupby('gender').sum().loc[:,'Action':'Western']
```

In [38]:
```
Genre_by_gender
```

Out[38]:

	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	S
gender															

F	5442	3141	995	2232	8068	1794	187	11008	363	385	1197	1442	1314	5858	
M	20147	10612	2610	4950	21764	6261	571	28887	989	1348	4120	3512	3931	13603	1

In [39]:

```
# Add Row total of the dataframe, to get the total number of Males and Females who gave ratings
Genre_by_gender['total'] = df_merge_all['gender'].value_counts()
```

In [40]:

```
Genre_by_gender.T
```

Out[40]:

gender	F	M
Action	5442	20147
Adventure	3141	10612
Animation	995	2610
Childrens	2232	4950
Comedy	8068	21764
Crime	1794	6261
Documentary	187	571
Drama	11008	28887
Fantasy	363	989
Film-Noir	385	1348
Horror	1197	4120
Musical	1442	3512
Mystery	1314	3931
Romance	5858	13603
Sci-Fi	2629	10101
Thriller	5086	16786
War	2189	7209
Western	371	1483
total	25738	74252

In [41]:

```
# Divide each cell with row total and multiply by 100 to get the percentage
(Genre_by_gender.div(Genre_by_gender.total, axis= 0) * 100).T
```

Out[41]:

gender	F	M
Action	21.143834	27.133276
Adventure	12.203745	14.291871
Animation	3.865879	3.515057
Childrens	8.672002	6.666487
Comedy	31.346647	29.310995
Crime	6.970239	8.432096
Documentary	0.726552	0.769003
Drama	42.769446	38.904003
Fantasy	1.410366	1.331951
Film-Noir	1.495843	1.815439
Horror	4.650711	5.548672
Musical	5.602611	4.729839
Mystery	5.105292	5.294133
Romance	22.760121	18.320045
Sci-Fi	10.214469	13.603674
Thriller	19.760665	22.606798
War	8.504934	9.708829
Western	1.441448	1.997253
total	100.000000	100.000000

Here are the five top conclusions based on the revised and cleaned-up MovieLens case study:

Genre Distribution:

Drama and Comedy are the most common genres among the movies in the dataset, with Drama having 725 movies and Comedy having 505. These two genres dominate the dataset, indicating a higher production of movies in these categories. Movies with Multiple Genres:

A significant number of movies belong to multiple genres. Out of 1680 movies, 849 movies have more than one genre assigned to them. This highlights the versatility and blending of genres in the film industry. Top-Rated Movies:

The top-rated movies with more than 100 ratings include classics like "Close Shave, A", "Schindler's List", "Wrong Trousers, The", "Casablanca", and "Shawshank Redemption, The". These movies have high average ratings, demonstrating their popularity and critical acclaim among users. Gender Preferences in Genres:

Drama is more popular among women, with 43% of the ratings given by women for Drama movies, compared to 39% by men. Conversely, men prefer genres like Action (27% of ratings by men) and Sci-Fi (14% of ratings by men) more than women, who gave 21% and 10% of their ratings to these genres, respectively. Age Distribution of Users:

The average age of users in the dataset is 34 years, with a standard deviation of approximately 12 years. The age range spans from 7 to 73 years, showing a wide demographic of movie watchers. This diversity in age indicates that the MovieLens dataset captures a broad audience with varied movie preferences.