Get Data from the following link: [http://files.grouplens.org/datasets/movielens/ml-20m.zip (http://files.grouplens.org/datasets/movielens/ml-20m.zip)](http://files.grouplens.org/datasets/movielens/ml-20m.zip)

We will be using the following files for this exercise:

```
ratings.csv : userId,movieId,rating, timestamp
tags.csv : userId,movieId, tag, timestamp
movies.csv : movieId, title, genres
```

1. Read the dataset using pandas.

In [1]:
```
import pandas as pd
import os

path="C:\\Users\\manoj\\Documents\\Acadgild DSB\\S6 - Pandas 2\\Assignments\\Addi
os.chdir(path)

ratings=pd.read_csv("ratings.csv")
tags=pd.read_csv("tags.csv")
movies=pd.read_csv("movies.csv")
```

2. Extract the first row from tags and print its type.

In [2]:
```
tagsFirstRow=tags.iloc[0]
print(type(tagsFirstRow))
```

```
<class 'pandas.core.series.Series'>
```

3. Extract row 0, 11, 2000 from tags DataFrame.

In [3]:
```
tags.iloc[[0,11,2000]]
```

Out[3]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	1240597180
11	65	1783	noir thriller	1368149983
2000	910	68554	conspiracy theory	1368043943

4. Print index, columns of the DataFrame.

In [4]:
```
print(ratings.index.values)
print(ratings.columns.values)
print(tags.index.values)
print(tags.columns.values)
print(movies.index.values)
print(movies.columns.values)
```

```
[       0        1        2 ... 20000260 20000261 20000262]
['userId' 'movieId' 'rating' 'timestamp']
[     0      1      2 ... 465561 465562 465563]
['userId' 'movieId' 'tag' 'timestamp']
[    0     1     2 ... 27275 27276 27277]
['movieId' 'title' 'genres']
```

5. Calculate descriptive statistics for the 'ratings' column of the ratings DataFrame. Verify using describe().

In [5]:
```
print("count\t"+str(ratings.rating.count()))
print("mean\t"+str(ratings.rating.mean()))
print("std\t"+str(ratings.rating.std()))
print("min\t"+str(ratings.rating.min()))
print("25%\t"+str(ratings.rating.quantile(0.25)))
print("50%\t"+str(ratings.rating.quantile(0.5)))
print("75%\t"+str(ratings.rating.quantile(0.75)))
print("max\t"+str(ratings.rating.max()))
print(ratings.rating.describe())
```

```
count   20000263
mean    3.5255285642993797
std     1.051988919275684
min     0.5
25%     3.0
50%     3.5
75%     4.0
max     5.0
count    2.000026e+07
mean     3.525529e+00
std      1.051989e+00
min      5.000000e-01
25%      3.000000e+00
50%      3.500000e+00
75%      4.000000e+00
max      5.000000e+00
Name: rating, dtype: float64
```

6. Filter out ratings with rating > 4

In [7]:
```
ratings[ratings.rating>4].head()
```

Out[7]:

	userId	movieId	rating	timestamp
30	1	1196	4.5	1112484742
31	1	1198	4.5	1112484624
131	1	4993	5.0	1112484682
142	1	5952	5.0	1112484619
158	1	7153	5.0	1112484633

7. Find how many null values, missing values are present. Deal with them. Print out how many rows have been modified.

In [8]:
```
    #Checking each dataframe
print("ratings has null:- "+str(ratings.isnull().any().any()))
print("tags has null:- " +str(tags.isnull().any().any()))
print("movies has null:- " +str(movies.isnull().any().any()))
    #Removing null rows
oldCount=tags.movieId.count()
noOfNullRows = tags[tags.tag.isnull()].movieId.count()
tags.dropna(inplace=True)
newCount=tags.movieId.count()
    #Check if the correct count has been removed
print("Old Count:- " + str(oldCount))
print("New Count:- " + str(newCount))
print("Null Count:- " + str(noOfNullRows))
print("Number of Null count matches with old count - new count? :- " + str(noOfNu
```

```
ratings has null:- False
tags has null:- True
movies has null:- False
Old Count:- 465564
New Count:- 465548
Null Count:- 16
Number of Null count matches with old count - new count? :- True
```

8. Filter out movies from the movies DataFrame that are of type 'Animation'.

In [10]:
```
movies[movies['genres']=="Animation"].head()
```

Out[10]:

	movieId	title	genres
2503	2588	Cloudland (1998)	Animation
4906	5002	Fritz the Cat (1972)	Animation
4907	5003	Nine Lives of Fritz the Cat, The (1974)	Animation
9455	27738	Cathedral, The (Katedra) (2002)	Animation
9989	32840	Vincent (1982)	Animation

9. Find the average rating of movies.

In [11]:
```python
import numpy as np
print(ratings.groupby("movieId")["rating"].agg(np.mean))
```

```
movieId
1         3.921240
2         3.211977
3         3.151040
4         2.861393
5         3.064592
6         3.834930
7         3.366484
8         3.142049
9         3.004924
10        3.430029
11        3.667713
12        2.619766
13        3.272416
14        3.432082
15        2.721993
16        3.787455
17        3.968573
18        3.373631
19        2.607412
20        2.880754
21        3.581689
22        3.319400
23        3.148235
24        3.199849
25        3.689510
26        3.628857
27        3.413520
28        4.057546
29        3.952230
30        3.633880
            ...
131146    4.000000
131148    4.000000
131150    4.000000
131152    0.500000
131154    3.500000
131156    4.000000
131158    4.000000
131160    4.000000
131162    2.000000
131164    4.000000
131166    4.000000
131168    3.500000
131170    3.500000
131172    1.000000
131174    3.500000
131176    4.500000
131180    2.500000
131231    3.500000
131237    3.000000
131239    4.000000
131241    4.000000
131243    4.000000
131248    4.000000
131250    4.000000
131252    4.000000
131254    4.000000
131256    4.000000
```

```
131258    2.500000
131260    3.000000
131262    4.000000
Name: rating, Length: 26744, dtype: float64
```

10. Perform an inner join of movies and tags based on movieId.

In [13]:
```
pd.merge(movies,tags,on="movieId").head()
```

Out[13]:

	movieId	title	genres	userId	tag	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1644	Watched	1417736680
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	computer animation	1183903155
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	Disney animated feature	1183933307
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	Pixar animation	1183934770
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	TÃ©a Leoni does not star in this movie	1245093573

11. Print out the 5 movies that belong to the Comedy genre and have rating greater than 4.

In [15]:
```
movies_ratings=pd.merge(movies,ratings,on="movieId")
comedy_4=movies_ratings[(movies_ratings["genres"].str.contains("Comedy")) & (movi
comedy_4.head()
```

Out[15]:

	movieId	title	genres	userId	rating	timesta
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	6	5.0	858275
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	11	4.5	1230858
7	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	14	4.5	1225311
9	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	19	5.0	855176
14	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	34	5.0	846509
71941	3	Grumpier Old Men (1995)	Comedy|Romance	8	5.0	833981
71952	3	Grumpier Old Men (1995)	Comedy|Romance	117	5.0	861553
71959	3	Grumpier Old Men (1995)	Comedy|Romance	251	5.0	965281
71976	3	Grumpier Old Men (1995)	Comedy|Romance	465	5.0	845587
71990	3	Grumpier Old Men (1995)	Comedy|Romance	593	5.0	861600
84683	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	518	5.0	839943
84684	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	601	5.0	848681
84699	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1638	5.0	836265
84703	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1931	5.0	848772
84707	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2174	5.0	851331
87435	5	Father of the Bride Part II (1995)	Comedy	117	5.0	861553
87437	5	Father of the Bride Part II (1995)	Comedy	127	5.0	847127
87455	5	Father of the Bride Part II (1995)	Comedy	350	5.0	1360209

	movieId	title	genres	userId	rating	timesta
87460	5	Father of the Bride Part II (1995)	Comedy	390	5.0	836139
87462	5	Father of the Bride Part II (1995)	Comedy	401	5.0	847049
123489	7	Sabrina (1995)	Comedy|Romance	6	5.0	858275
123493	7	Sabrina (1995)	Comedy|Romance	19	5.0	855176
123496	7	Sabrina (1995)	Comedy|Romance	38	5.0	835885
123501	7	Sabrina (1995)	Comedy|Romance	113	5.0	858515
123511	7	Sabrina (1995)	Comedy|Romance	251	5.0	965280
170830	11	American President, The (1995)	Comedy|Drama|Romance	5	5.0	851527
170840	11	American President, The (1995)	Comedy|Drama|Romance	54	5.0	974840
170842	11	American President, The (1995)	Comedy|Drama|Romance	58	4.5	1144058
170853	11	American President, The (1995)	Comedy|Drama|Romance	156	5.0	1039204
170855	11	American President, The (1995)	Comedy|Drama|Romance	160	5.0	833408
...	...	...	...	...	...	
19999624	129348	A Short History of Decay (2014)	Comedy	68026	4.5	1426052
19999629	129354	Focus (2015)	Comedy|Crime|Drama|Romance	21210	4.5	1426671
19999642	129354	Focus (2015)	Comedy|Crime|Drama|Romance	51995	5.0	1425664
19999648	129354	Focus (2015)	Comedy|Crime|Drama|Romance	81396	4.5	1425894
19999650	129354	Focus (2015)	Comedy|Crime|Drama|Romance	98886	5.0	1425999
19999653	129354	Focus (2015)	Comedy|Crime|Drama|Romance	130746	5.0	1427667
19999665	129370	SpongeBob Movie: Sponge Out of Water, The (2015)	Adventure|Animation|Children|Comedy	79570	4.5	1427651

	movieId	title	genres	userId	rating	timesta
19999683	129401	Kevin Smith: Sold Out - A Threevening with Kev...	Comedy|Documentary	74142	5.0	1425516
19999704	129428	The Second Best Exotic Marigold Hotel (2015)	Comedy|Drama	17308	5.0	1426180
19999708	129428	The Second Best Exotic Marigold Hotel (2015)	Comedy|Drama	36725	4.5	1425889
19999714	129428	The Second Best Exotic Marigold Hotel (2015)	Comedy|Drama	94583	5.0	1425832
19999715	129428	The Second Best Exotic Marigold Hotel (2015)	Comedy|Drama	95692	5.0	1426912
19999719	129428	The Second Best Exotic Marigold Hotel (2015)	Comedy|Drama	137055	5.0	1426126
19999731	129451	Ingenious (2009)	Comedy|Drama|Romance	51558	4.5	1425259
19999744	129514	George Carlin: It's Bad for Ya! (2008)	Comedy	59417	5.0	1425800
19999745	129514	George Carlin: It's Bad for Ya! (2008)	Comedy	61923	4.5	1425682
19999746	129514	George Carlin: It's Bad for Ya! (2008)	Comedy	69470	4.5	1425418
19999747	129514	George Carlin: It's Bad for Ya! (2008)	Comedy	85554	5.0	1426064
19999750	129516	Poison (1951)	Comedy	52697	5.0	1425418
19999751	129518	Forgive Me (2006)	Comedy	52697	4.5	1425418
19999753	129520	Papa (2005)	Comedy|Drama	115147	4.5	1425419
19999756	129526	The Color of Milk (2004)	Comedy|Drama	52697	5.0	1425421
19999800	129719	That's Life (1998)	Comedy	13965	4.5	1425815
19999816	129773	Soulless (2012)	Comedy|Drama	98886	4.5	1425999
19999866	129905	The Floating Castle (2012)	Comedy|Drama	134701	5.0	1426268

	movieId	title	genres	userId	rating	timesta
19999995	130347	Bill Hicks: Sane Man (1989)	Comedy	122319	5.0	1426542
20000157	130970	George Carlin: Life Is Worth Losing (2005)	Comedy	69470	4.5	1427401
20000158	130970	George Carlin: Life Is Worth Losing (2005)	Comedy	86211	5.0	1427490
20000162	130978	Love and Pigeons (1985)	Comedy|Romance	26497	5.0	1427532
20000182	131027	But Forever in My Mind (1999)	Comedy|Drama	67380	4.5	1427612

22300 rows × 6 columns

12. Split 'genres' into multiple columns.

In [16]:

```
m=pd.concat([movies.drop(["genres"],axis=1),pd.DataFrame(movies["genres"].str.spl
m.head()
```

Out[16]:

	movieId	title	genre_0	genre_1	genre_2	genre_3	genre_4	genre_5	genre_6	genre_
0	1	Toy Story (1995)	Adventure	Animation	Children	Comedy	Fantasy	None	None	Non
1	2	Jumanji (1995)	Adventure	Children	Fantasy	None	None	None	None	Non
2	3	Grumpier Old Men (1995)	Comedy	Romance	None	None	None	None	None	Non
3	4	Waiting to Exhale (1995)	Comedy	Drama	Romance	None	None	None	None	Non
4	5	Father of the Bride Part II (1995)	Comedy	None	None	None	None	None	None	Non

13. Extract year from title e.g. (1995).

In [18]:
```
movies['title'].str.extract('(\(\d{4}\))',expand=True).head()
```
Out[18]:

	0
0	(1995)
1	(1995)
2	(1995)
3	(1995)
4	(1995)

In [19]:
```
movies['title'].str.extract('(\d{4})',expand=True).head()
```
Out[19]:

	0
0	1995
1	1995
2	1995
3	1995
4	1995

14. Select rows based on timestamps later than 2015-02-01.

In [20]:
```
t=pd.concat([tags.drop(["timestamp"],axis=1),pd.to_datetime(tags['timestamp'], un
t[t['timestamp']>'2015-02-01'].head()
```
Out[20]:

	userId	movieId	tag	timestamp
301	318	260	1970s	2015-02-20 22:42:49
302	318	260	fantasy	2015-02-20 22:42:49
303	318	260	sci-fi	2015-02-20 22:42:49
304	318	115149	Action	2015-02-21 15:58:30
305	318	115149	Revenge	2015-02-21 15:58:03

15. Sort the tags DataFrame based on timestamp.

In [21]:
```
tags.sort_values("timestamp").head()
```
Out[21]:

	userId	movieId	tag	timestamp
333932	100371	2788	monty python	1135429210
333927	100371	1732	coen brothers	1135429236
333924	100371	1206	stanley kubrick	1135429248
333923	100371	1193	jack nicholson	1135429371
333939	100371	5004	peter sellers	1135429399

In [ ]: