IMDb movie Rating Analysis

In [2]:
```
import pandas as pd
```

In [3]:
```
movies = pd.read_csv(r"C:\Users\Affan\OneDrive\Desktop\FSDS Course NIT\Prakash S
movies.head()
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:
```
print(type(movies))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [5]:
```
ratings = pd.read_csv(r"C:\Users\Affan\OneDrive\Desktop\FSDS Course NIT\Prakash
```

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

Out[6]:
```
(20000263, 4)
```

In [7]:
```
tags = pd.read_csv(r"C:\Users\Affan\OneDrive\Desktop\FSDS Course NIT\Prakash Sir
```

In [8]:
```
tags.shape
```

Out[8]:
```
(465564, 4)
```

In [9]:
```
tags.columns
```

Out[9]:
```
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [10]:
```
ratings.columns
```

Out[10]:
```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

In [11]:
```
del tags['timestamp']
del ratings['timestamp']
```

In [12]:
```
tags.columns
```

Out[12]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [13]:
```
ratings.columns
```

Out[13]:
```
Index(['userId', 'movieId', 'rating'], dtype='object')
```

Series

movie.csv and tag.csv has same col name movieid ---relational table both are in realtion with primary and foreign key

In [14]:
```
tags.head()
```

Out[14]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero

In [15]:
```
tags.iloc[0] #first row info
```

Out[15]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [16]:
```
tags.iloc[2] #it gives info of row
```

Out[16]:
```
userId            65
movieId          353
tag        dark hero
Name: 2, dtype: object
```

In [17]:
```
row_0 = tags.iloc[0]
row_0
```

Out[17]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [18]:
```
row_0.index
```

Out[18]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [19]:
```
'rating' in row_0
```

Out[19]:
```
False
```

In [20]:
```
'timestamp' in row_0
```

Out[20]:
```
False
```

In [21]:
```
'userId' in row_0
```

Out[21]:
```
True
```

In [22]:
```
row_0.name
```

Out[22]:
```
0
```

In [23]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[23]:
```
'firstRow'
```

In [24]:
```
row_0.index
```

Out[24]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

Dataframes

In [25]:
```
tags.head()
```

Out[25]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero

In [26]:
```
print(tags.index)
print(tags.columns)
```

```
RangeIndex(start=0, stop=465564, step=1)
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [27]:
```
tags.iloc[0] #row info
```

Out[27]:
```
userId             18
movieId          4141
tag        Mark Waters
Name: 0, dtype: object
```

In [28]:
```
tags.iloc[[0,11,500]] #multiple rows at index can be printed
```

Out[28]:

	userId	movieId	tag
0	18	4141	Mark Waters
11	65	1783	noir thriller
500	342	55908	entirely dialogue

In [29]:
```
tags.iloc[[0,69,96]]
```

Out[29]:

	userId	movieId	tag
0	18	4141	Mark Waters
69	121	5283	R
96	121	36529	Nudity (Topless)

In [30]:
```
'tag' in tags
```

Out[30]:
```
True
```

In [31]:
```
tags.iloc[465563]
```

Out[31]:
```
userId            138472
movieId              923
tag        rise to power
Name: 465563, dtype: object
```

Descriptive Analysis

lets see how the ratings are distributed

In [32]:
```
ratings.head()
```

Out[32]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5

In [33]:
```
ratings['rating'].describe() #of particular row 'rating' in the dataset
```

Out[33]:
```
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
```

In [34]:
```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000263 entries, 0 to 20000262
Data columns (total 3 columns):
 #   Column   Dtype  
---  ------   -----  
 0   userId   int64  
 1   movieId  int64  
 2   rating   float64
dtypes: float64(1), int64(2)
memory usage: 457.8 MB
```

In [35]:
```
print(len(ratings))
print(id(ratings))
```

```
20000263
1495743085264
```

In [36]:
```
ratings.describe() #of whole dataset
```

Out[36]:

	userId	movieId	rating
count	2.000026e+07	2.000026e+07	2.000026e+07
mean	6.904587e+04	9.041567e+03	3.525529e+00
std	4.003863e+04	1.978948e+04	1.051989e+00
min	1.000000e+00	1.000000e+00	5.000000e-01
25%	3.439500e+04	9.020000e+02	3.000000e+00
50%	6.914100e+04	2.167000e+03	3.500000e+00
75%	1.036370e+05	4.770000e+03	4.000000e+00
max	1.384930e+05	1.312620e+05	5.000000e+00

In [37]:
```
ratings.describe().head()
```

Out[37]:

	userId	movieId	rating
count	2.000026e+07	2.000026e+07	2.000026e+07
mean	6.904587e+04	9.041567e+03	3.525529e+00
std	4.003863e+04	1.978948e+04	1.051989e+00
min	1.000000e+00	1.000000e+00	5.000000e-01
25%	3.439500e+04	9.020000e+02	3.000000e+00

In [38]:
```
ratings.head().describe()
```

Out[38]:

	userId	movieId	rating
count	5.0	5.000000	5.0
mean	1.0	32.000000	3.5
std	0.0	19.091883	0.0
min	1.0	2.000000	3.5
25%	1.0	29.000000	3.5
50%	1.0	32.000000	3.5
75%	1.0	47.000000	3.5
max	1.0	50.000000	3.5

#REMEMBER --> the sequence of putting multiple function in same line also matters on how you want the output to careful!!

In [39]:
```
ratings['rating'].mean() #of particular row in dataset
```

Out[39]:
```
3.5255285642993797
```

In [40]:
```
ratings.mean() #of whole dataset(all rows)
```

Out[40]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [41]:
```
ratings['rating'].min()
```

Out[41]:
```
0.5
```

In [42]:
```
ratings['rating'].max()
```

Out[42]:
```
5.0
```

In [43]:
```
ratings['rating'].std()
```

Out[43]:
```
1.051988919275684
```

In [44]:
```
ratings['rating'].mode().head()
```

Out[44]:
```
0    4.0
Name: rating, dtype: float64
```

In [45]:
```
ratings['rating'].cumsum().head()
```

Out[45]:
```
0     3.5
1     7.0
2    10.5
3    14.0
4    17.5
Name: rating, dtype: float64
```

In [46]:
```
ratings['rating'].cumprod().head()
```

```
C:\Users\Affan\anaconda3\Lib\site-packages\numpy\core\fromnumeric.py:59: RuntimeW
arning: overflow encountered in accumulate
  return bound(*args, **kwds)
```

Out[46]:
```
0      3.50000
1     12.25000
2     42.87500
3    150.06250
4    525.21875
Name: rating, dtype: float64
```

In [47]:
```
ratings.corr()
```

Out[47]:

	userId	movieId	rating
userId	1.000000	-0.000850	0.001175
movieId	-0.000850	1.000000	0.002606
rating	0.001175	0.002606	1.000000

In [48]:
```
ratings.tail()
```

Out[48]:

	userId	movieId	rating
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [49]:
```
filter1 = ratings['rating']>10
print(filter1)
filter1.any()
```

```
0           False
1           False
2           False
3           False
4           False
            ...
20000258    False
20000259    False
20000260    False
20000261    False
20000262    False
Name: rating, Length: 20000263, dtype: bool
```

Out[49]:
```
False
```

In [50]:
```
filter1.tail()
```

Out[50]:
```
20000258    False
20000259    False
20000260    False
20000261    False
20000262    False
Name: rating, dtype: bool
```

In [51]:
```
filter1.any() #because there is no movie rated moree than 10 thats why we get fa
```

Out[51]:
```
False
```

In [ ]:
```

```

In [52]:
```
filter2 = ratings['rating']>0
print(filter2)
```

```
0           True
1           True
2           True
3           True
4           True
            ...
20000258    True
20000259    True
20000260    True
20000261    True
20000262    True
Name: rating, Length: 20000263, dtype: bool
```

In [53]:
```
filter2.all()
```

Out[53]:
```
True
```

In [ ]:
```

```

In [54]:
```
filter3 = ratings['rating']>5
print(filter3)
```

```
0           False
1           False
2           False
3           False
4           False
            ...
20000258    False
20000259    False
20000260    False
20000261    False
20000262    False
Name: rating, Length: 20000263, dtype: bool
```

In [55]:
```
filter4 = ratings['rating']>3.5
print(filter4)
```

```
0           False
1           False
2           False
3           False
4           False
            ...
20000258     True
20000259     True
20000260    False
20000261     True
20000262    False
Name: rating, Length: 20000263, dtype: bool
```

In [56]:
```
print(len(ratings['rating']))
```

```
20000263
```

In [57]:
```
ratings['rating']
```

Out[57]:
```
0           3.5
1           3.5
2           3.5
3           3.5
4           3.5
           ...
20000258    4.5
20000259    4.5
20000260    3.0
20000261    5.0
20000262    2.5
Name: rating, Length: 20000263, dtype: float64
```

Data Cleaning: handling missing data

In [58]:
```
movies.shape
```

Out[58]:
```
(27278, 3)
```

In [59]:
```
movies.isna().tail()
```

Out[59]:

	movieId	title	genres
27273	False	False	False
27274	False	False	False
27275	False	False	False
27276	False	False	False
27277	False	False	False

In [60]:
```
movies.columns
```

Out[60]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [61]:
```
movies.index
```

Out[61]:
```
RangeIndex(start=0, stop=27278, step=1)
```

In [62]:
```
movies.any()
```

Out[62]:
```
movieId    True
title      True
genres     True
dtype: bool
```

In [63]:
```
movies.all()
```

Out[63]:
```
movieId    True
title      True
genres     True
dtype: bool
```

In [64]:
```
movies.isnull().any().any()
```

Out[64]:
```
False
```

In [65]:
```
movies.shape
```

Out[65]:
```
(27278, 3)
```

In [66]:
```
#no null values
```

In [67]:
```
ratings.shape
```

Out[67]:
```
(20000263, 3)
```

In [68]:
```
ratings.isnull().any().any()
```

Out[68]:
```
False
```

In [69]:
```
tags.shape
```

Out[69]:
```
(465564, 3)
```

In [70]:
```
tags.isnull().any().any()
```

Out[70]:
```
True
```

In [71]:
```
tags.columns
```

Out[71]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [72]:
```
tags.index
```

Out[72]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [73]:
```
tags = tags.dropna()
```

In [74]:
```
tags.isnull().any().any()
```

Out[74]:
```
False
```

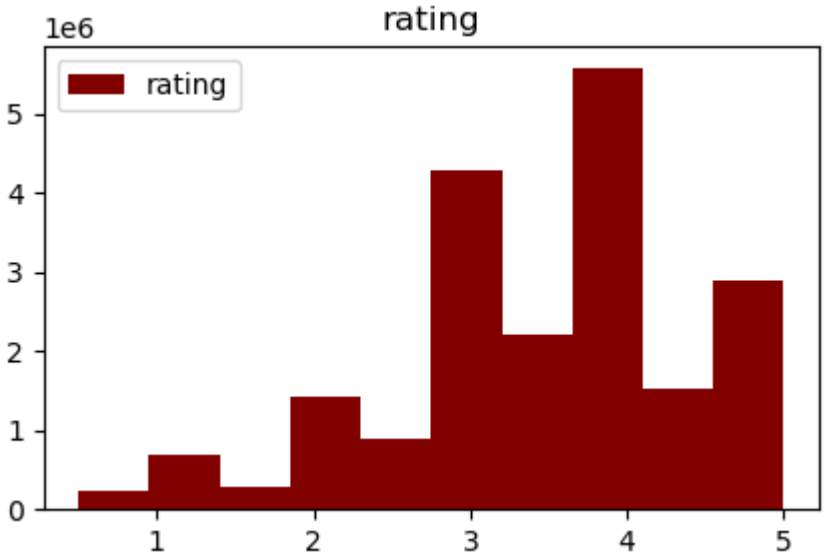
In [75]:
```
tags.shape #notice the no. of lines changes..meaning null values have been remov
```

Out[75]: (465548, 3)

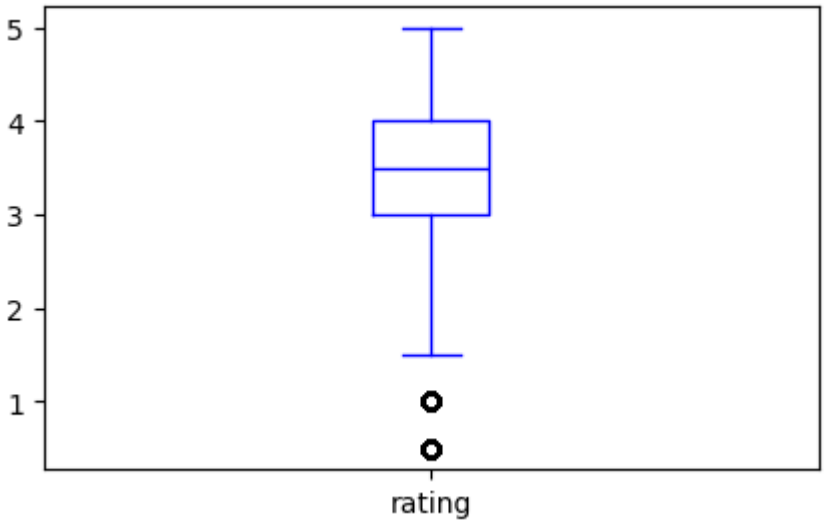
Data visualization

In [76]:
```
import matplotlib.pyplot as plt
%matplotlib inline

ratings.hist(column='rating',legend=True,bins=10,figsize=(5,3),color='maroon')
plt.grid(False)
plt.show()
```



In [77]:
```
ratings.boxplot(column='rating',figsize=(5,3),color='blue')
plt.grid(False)
plt.show()
```



slicing out columns

In [78]:
```
tags['tag'].head()
```

Out[78]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [79]:
```
tags.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 465548 entries, 0 to 465563
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype 
---  ------   --------------   ----- 
 0   userId   465548 non-null  int64 
 1   movieId  465548 non-null  int64 
 2   tag      465548 non-null  object
dtypes: int64(2), object(1)
memory usage: 14.2+ MB
```

In [80]:
```
movies.columns
```

Out[80]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [81]:
```
movies[['title','genres']].head()
```

Out[81]:

	title	genres
0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	Jumanji (1995)	Adventure|Children|Fantasy
2	Grumpier Old Men (1995)	Comedy|Romance
3	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	Father of the Bride Part II (1995)	Comedy

In [82]:
```
movies[['title','genres']].describe()
```

Out[82]:

	title	genres
count	27278	27278
unique	27262	1342
top	Aladdin (1992)	Drama
freq	2	4520

In [83]:
```
movies[['title','genres']].head().max()
```

Out[83]:
```
title     Waiting to Exhale (1995)
genres              Comedy|Romance
dtype: object
```

In [84]:
```
ratings[-10:]
```

Out[84]:

	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [85]:
```
ratings[::-1].tail()
```

Out[85]:

	userId	movieId	rating
4	1	50	3.5
3	1	47	3.5
2	1	32	3.5
1	1	29	3.5
0	1	2	3.5

In [86]:
```
ratings[::-1].head()
```

Out[86]:

	userId	movieId	rating
20000262	138493	71619	2.5
20000261	138493	70286	5.0
20000260	138493	69644	3.0
20000259	138493	69526	4.5
20000258	138493	68954	4.5

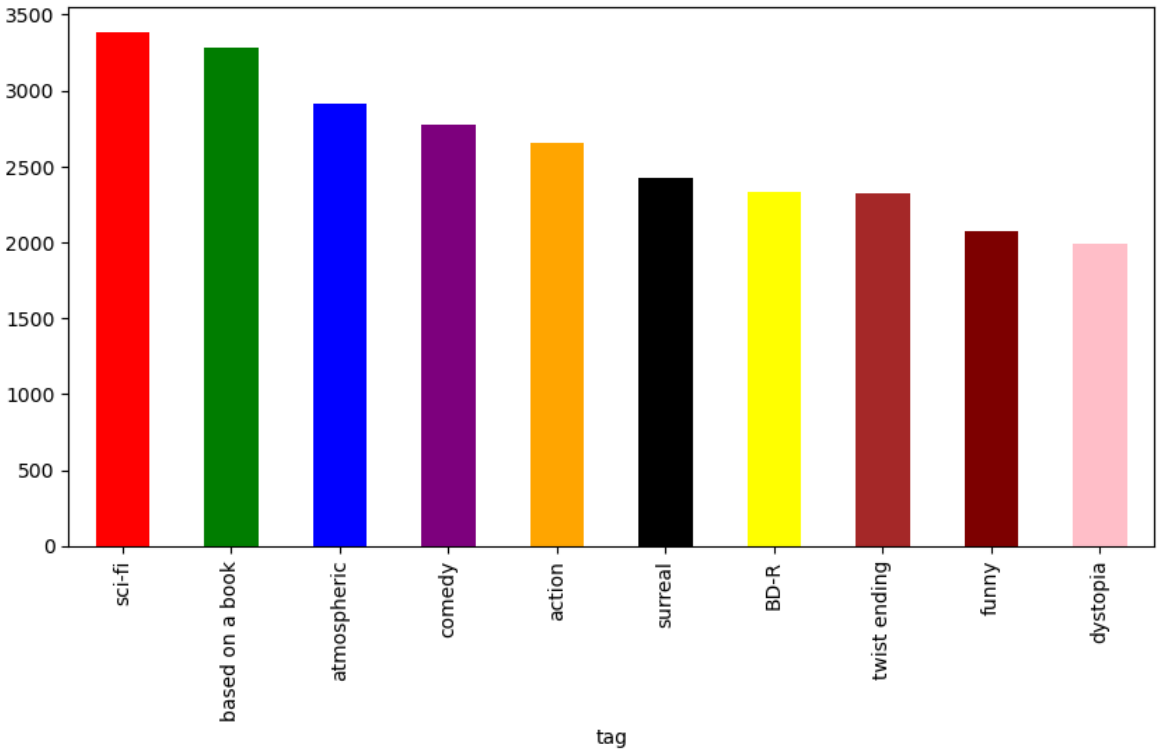
In [87]:
```
tag_count = tags['tag'].value_counts() #it return the count of all values or tag
tag_count
```

Out[87]:
```
tag
sci-fi                            3384
based on a book                   3281
atmospheric                       2917
comedy                            2779
action                            2657
                                  ...
Paul Adelstein                       1
the wig                              1
killer fish                          1
genetically modified monsters        1
topless scene                        1
Name: count, Length: 38643, dtype: int64
```

In [88]:
```
tag_count[-10:]
```

Out[88]:
```
tag
missing child                     1
Ron Moore                         1
Citizen Kane                      1
mullet                            1
biker gang                        1
Paul Adelstein                    1
the wig                           1
killer fish                       1
genetically modified monsters     1
topless scene                     1
Name: count, dtype: int64
```

In [89]:
```
bar_colors = ['red', 'green', 'blue', 'purple', 'orange','black','yellow','brown
tag_count[:10].plot(kind='bar',figsize=(10,5),color=bar_colors) #bar plot of fir
plt.show()
```



Filters for selecting rows

In [90]:
```
is_high_rated=ratings['rating']>=5.0
is_high_rated
```

Out[90]:
```
0           False
1           False
2           False
3           False
4           False
            ...
20000258    False
20000259    False
20000260    False
20000261     True
20000262    False
Name: rating, Length: 20000263, dtype: bool
```

In [91]:
```
ratings[is_high_rated][30:50]
```

Out[91]:

	userId	movieId	rating
239	3	50	5.0
242	3	175	5.0
244	3	223	5.0
245	3	260	5.0
246	3	316	5.0
247	3	318	5.0
248	3	329	5.0
252	3	457	5.0
253	3	480	5.0
254	3	490	5.0
256	3	541	5.0
258	3	593	5.0
263	3	858	5.0
264	3	904	5.0
267	3	924	5.0
268	3	953	5.0
271	3	1060	5.0
272	3	1073	5.0
275	3	1084	5.0
276	3	1089	5.0

In [92]:
```
ratings[is_high_rated][:10]
```

Out[92]:

	userId	movieId	rating
131	1	4993	5.0
142	1	5952	5.0
158	1	7153	5.0
170	1	8507	5.0
176	2	62	5.0
177	2	70	5.0
180	2	260	5.0
181	2	266	5.0
183	2	480	5.0
184	2	541	5.0

In [93]:
```
is_action=movies['genres'].str.contains('Action')
movies[is_action][5:15]
```

Out[93]:

	movieId	title	genres
22	23	Assassins (1995)	Action|Crime|Thriller
41	42	Dead Presidents (1995)	Action|Crime|Drama
43	44	Mortal Kombat (1995)	Action|Adventure|Fantasy
50	51	Guardian Angel (1994)	Action|Drama|Thriller
65	66	Lawnmower Man 2: Beyond Cyberspace (1996)	Action|Sci-Fi|Thriller
69	70	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
70	71	Fair Game (1995)	Action
75	76	Screamers (1995)	Action|Sci-Fi|Thriller
77	78	Crossing Guard, The (1995)	Action|Crime|Drama|Thriller
85	86	White Squall (1996)	Action|Adventure|Drama

In [94]:
```
is_action1=movies['genres'] == 'Action'
movies[is_action1][:10]
```

Out[94]:

	movieId	title	genres
8	9	Sudden Death (1995)	Action
70	71	Fair Game (1995)	Action
202	204	Under Siege 2: Dark Territory (1995)	Action
248	251	Hunted, The (1995)	Action
659	667	Bloodsport 2 (a.k.a. Bloodsport II: The Next K...	Action
962	980	Yes, Madam (a.k.a. Police Assassins) (a.k.a. I...	Action
1080	1102	American Strays (1996)	Action
1087	1110	Bird of Prey (1996)	Action
1147	1170	Best of the Best 3: No Turning Back (1995)	Action
1390	1424	Inside (1996)	Action

In [95]:
```
movies[is_action1].head()
```

Out[95]:

	movieId	title	genres
8	9	Sudden Death (1995)	Action
70	71	Fair Game (1995)	Action
202	204	Under Siege 2: Dark Territory (1995)	Action
248	251	Hunted, The (1995)	Action
659	667	Bloodsport 2 (a.k.a. Bloodsport II: The Next K...	Action

Group by and Agg

In [96]:
```
ratings.columns
```

Out[96]:
```
Index(['userId', 'movieId', 'rating'], dtype='object')
```

In [97]:
```
rat_count=ratings[['rating','movieId']].groupby('rating').count() #grouped movie
rat_count
```

Out[97]:

rating	movieId
0.5	239125
1.0	680732
1.5	279252
2.0	1430997
2.5	883398
3.0	4291193
3.5	2200156
4.0	5561926
4.5	1534824
5.0	2898660

In [98]:
```
len(rat_count)
```

Out[98]: 10

In [99]:
```
len(ratings)
```

Out[99]: 20000263

In [100…
```
ratings[['userId','rating']].groupby('userId').count()
```

Out[100…

userId	rating
1	175
2	61
3	187
4	28
5	66
…	…
138489	38
138490	151
138491	22
138492	82
138493	373

138493 rows × 1 columns

In [101…
```
avg_rat=ratings[['rating','movieId']].groupby('movieId').mean() #grouped movie i
avg_rat.head()
```

Out[101…

movieId	rating
1	3.921240
2	3.211977
3	3.151040
4	2.861393
5	3.064592

In [102…
```
movies.columns
```

Out[102…
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [103…
```
movie_count=ratings[['movieId','rating']].groupby('movieId').count()
movie_count
```

Out[103…

movieId	rating
1	49695
2	22243
3	12735
4	2756
5	12161
…	…
131254	1
131256	1
131258	1
131260	1
131262	1

26744 rows × 1 columns

In [104…
```
genre_based=movies[['movieId','title']].groupby('movieId')
genre_based.head()
```

Out[104…

	movieId	title
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)
...	...	...
27273	131254	Kein Bund für's Leben (2007)
27274	131256	Feuer, Eis & Dosenbier (2002)
27275	131258	The Pirates (2014)
27276	131260	Rentun Ruusu (2001)
27277	131262	Innocence (2014)

27278 rows × 2 columns

Merge Databases

In [105…
```
tags.head()
```

Out[105…

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero

In [106…
```
movies.head()
```

Out[106…

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [107…
```
ratings.head()
```

Out[107…

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5

In [108…

```
t = movies.merge(tags,on='movieId',how='inner')
t.head()
```

Out[108…

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

combine aggregations, mergigns and filters to get useful insights

In [110…

```
avg_ratings = ratings.groupby('movieId',as_index=False).mean()
del avg_ratings['userId']
avg_ratings.head()
```

Out[110…

	movieId	rating
0	1	3.921240
1	2	3.211977
2	3	3.151040
3	4	2.861393
4	5	3.064592

In [115…
```
box_office = movies.merge(avg_ratings, on='movieId', how='inner')
box_office.tail() #merging movies and avg_rating table based on movieId
```

Out[115…

	movieId	title	genres	rating
26739	131254	Kein Bund für's Leben (2007)	Comedy	4.0
26740	131256	Feuer, Eis & Dosenbier (2002)	Comedy	4.0
26741	131258	The Pirates (2014)	Adventure	2.5
26742	131260	Rentun Ruusu (2001)	(no genres listed)	3.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

In [116…
```
is_highly_rated = box_office['rating'] >= 4.0
box_office[is_highly_rated][-5:] #printing rating 4.0 rows from box_office table
```

Out[116…

	movieId	title	genres	rating
26737	131250	No More School (2000)	Comedy	4.0
26738	131252	Forklift Driver Klaus: The First Day on the Jo...	Comedy|Horror	4.0
26739	131254	Kein Bund für's Leben (2007)	Comedy	4.0
26740	131256	Feuer, Eis & Dosenbier (2002)	Comedy	4.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

In [117…
```
is_adventure = box_office['genres'].str.contains('Adventure')
box_office[is_adventure][:5] #printing rows from box_office having 'adventure' g
```

Out[117…

	movieId	title	genres	rating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3.921240
1	2	Jumanji (1995)	Adventure|Children|Fantasy	3.211977
7	8	Tom and Huck (1995)	Adventure|Children	3.142049
9	10	GoldenEye (1995)	Action|Adventure|Thriller	3.430029
12	13	Balto (1995)	Adventure|Animation|Children	3.272416

In [119…
```
box_office[is_adventure & is_highly_rated][-5:]
```

Out[119…

	movieId	title	genres	rating
26611	130586	Itinerary of a Spoiled Child (1988)	Adventure|Drama	4.5
26655	130996	The Beautiful Story (1992)	Adventure|Drama|Fantasy	5.0
26667	131050	Stargate SG-1 Children of the Gods - Final Cut...	Adventure|Sci-Fi|Thriller	5.0
26736	131248	Brother Bear 2 (2006)	Adventure|Animation|Children|Comedy|Fantasy	4.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

In [120…
```
box_office[-5:]
```

Out[120…

	movieId	title	genres	rating
26739	131254	Kein Bund für's Leben (2007)	Comedy	4.0
26740	131256	Feuer, Eis & Dosenbier (2002)	Comedy	4.0
26741	131258	The Pirates (2014)	Adventure	2.5
26742	131260	Rentun Ruusu (2001)	(no genres listed)	3.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

vectorized string opr

In [121…
```
movies.head()
```

Out[121…

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [122…
```
movies.tail()
```

Out[122...

	movieId	title	genres
27273	131254	Kein Bund für's Leben (2007)	Comedy
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy
27275	131258	The Pirates (2014)	Adventure
27276	131260	Rentun Ruusu (2001)	(no genres listed)
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror

Split 'genres' into mul cols

In [126...

```
movies_genres = movies['genres'].str.split('|',expand=True)
movies_genres[:10]
```

Out[126...

	0	1	2	3	4	5	6	7	8	9
0	Adventure	Animation	Children	Comedy	Fantasy	None	None	None	None	None
1	Adventure	Children	Fantasy	None	None	None	None	None	None	None
2	Comedy	Romance	None	None	None	None	None	None	None	None
3	Comedy	Drama	Romance	None	None	None	None	None	None	None
4	Comedy	None	None	None	None	None	None	None	None	None
5	Action	Crime	Thriller	None	None	None	None	None	None	None
6	Comedy	Romance	None	None	None	None	None	None	None	None
7	Adventure	Children	None	None	None	None	None	None	None	None
8	Action	None	None	None	None	None	None	None	None	None
9	Action	Adventure	Thriller	None	None	None	None	None	None	None

add new col for comedy genre flag

In [131...

```
movies_genres['isComedy']=movies['genres'].str.contains('Comedy')
movies_genres[0:10] #in here, a new col is created which gives TRUE or FALSE if
```

Out[131…

	0	1	2	3	4	5	6	7	8	9
0	Adventure	Animation	Children	Comedy	Fantasy	None	None	None	None	None
1	Adventure	Children	Fantasy	None	None	None	None	None	None	None
2	Comedy	Romance	None	None	None	None	None	None	None	None
3	Comedy	Drama	Romance	None	None	None	None	None	None	None
4	Comedy	None	None	None	None	None	None	None	None	None
5	Action	Crime	Thriller	None	None	None	None	None	None	None
6	Comedy	Romance	None	None	None	None	None	None	None	None
7	Adventure	Children	None	None	None	None	None	None	None	None
8	Action	None	None	None	None	None	None	None	None	None
9	Action	Adventure	Thriller	None	None	None	None	None	None	None

Extract year from title e.g. (2007)

In [136…

```
movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand=True)
movies.tail() #taking out the year from title and putting them in a new colum 'y
```

```
<>:1: SyntaxWarning: invalid escape sequence '\('
<>:1: SyntaxWarning: invalid escape sequence '\('
C:\Users\Affan\AppData\Local\Temp\ipykernel_18088\2413664988.py:1: SyntaxWarning:
invalid escape sequence '\('
  movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand=True)
```

Out[136…

	movieId	title	genres	year
27273	131254	Kein Bund für's Leben (2007)	Comedy	2007
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy	2002
27275	131258	The Pirates (2014)	Adventure	2014
27276	131260	Rentun Ruusu (2001)	(no genres listed)	2001
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror	2014

In [139…

```
tags.dtypes
```

Out[139…

```
userId      int64
movieId     int64
tag        object
dtype: object
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: