IMDb Movies Rating Data Analysis

In [ ]:
```
 #importing libraries

import numpy as np
import pandas as pd
```

In [2]:
```
# Reading the excel sheets

Rating = pd.read_csv(r"C:\Users\ratho\.ipynb_checkpoints\DATA\rating.csv")
Tags = pd.read_csv(r"C:\Users\ratho\.ipynb_checkpoints\DATA\tag.csv")
Movies = pd.read_csv(r"C:\Users\ratho\.ipynb_checkpoints\DATA\movie.csv")
```

In [3]:
```
# fetching data

Rating
```

Out[3]:

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40
...	...	...	...	...
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

20000263 rows × 4 columns

In [4]:
```
Tags
```

Out[4]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
1	65	208	dark hero	2013-05-10 01:41:18
2	65	353	dark hero	2013-05-10 01:41:19
3	65	521	noir thriller	2013-05-10 01:39:43
4	65	592	dark hero	2013-05-10 01:41:18
...	...	...	...	...
465559	138446	55999	dragged	2013-01-23 23:29:32
465560	138446	55999	Jason Bateman	2013-01-23 23:29:38
465561	138446	55999	quirky	2013-01-23 23:29:38
465562	138446	55999	sad	2013-01-23 23:29:32
465563	138472	923	rise to power	2007-11-02 21:12:47

465564 rows × 4 columns

In [5]:
```
Movies
```

Out[5]:

	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
...	...	...	...
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

27278 rows × 3 columns

In [6]:
```
Tags.shape                # Shape of tag data
```

Out[6]:
```
(465564, 4)
```

In [7]:
```
Rating.shape              # Shape rating data
```

Out[7]:
```
(20000263, 4)
```

In [8]:
```
Movies.shape                # shape of movies data
```

Out[8]:
```
(27278, 3)
```

In [9]:
```
Rating.head()               # Top 5 rows of Rating data
```

Out[9]:

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40

In [10]:
```
Tags.head()                    # Top 5 rows of Tags data
```

Out[10]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
1	65	208	dark hero	2013-05-10 01:41:18
2	65	353	dark hero	2013-05-10 01:41:19
3	65	521	noir thriller	2013-05-10 01:39:43
4	65	592	dark hero	2013-05-10 01:41:18

In [11]:
```
Movies.head()                  # Top 5 rows of Movies data
```

Out[11]:

	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 [12]:
```
del Rating['timestamp']              # deleting 'timestamp' column form
del Tags['timestamp']                # both the data of rating and tags
```

In [13]:
```
Rating
```

Out[13]:

	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
...	...	...	...
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

20000263 rows × 3 columns

In [14]:
```
Tags
```

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
...	...	...	...
465559	138446	55999	dragged
465560	138446	55999	Jason Bateman
465561	138446	55999	quirky
465562	138446	55999	sad
465563	138472	923	rise to power

465564 rows × 3 columns

In [15]:
```
Movies
```

Out[15]:

	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
...	...	...	...
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

27278 rows × 3 columns

In [16]:
```
Row = Tags.iloc[0]          # fetching 1st row from row data
Row
```

Out[16]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [17]:
```
type(Row)                   # Type of row
```

Out[17]:
```
pandas.core.series.Series
```

In [18]:
```
Row.index                   # index
```

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

In [19]:
```
Row['userId']               # fetching user it of row
```

Out[19]:
```
18
```

In [20]:
```
'rating' in Row             # is rating column is there in row
```

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

In [21]:
```
Row.name
```

Out[21]:
```
0
```

In [22]:
```
Row = Row.rename('firstRow')        # rename row
Row.name
```

Out[22]:
```
'firstRow'
```

Dataframes

In [23]:
```
Tags.head()                              # top 5 rows of Tags data
```

Out[23]:

	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 [24]:
```
Tags.index                               # index in Tags data
```

Out[24]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [25]:
```
Tags.columns                             # columns in Tags data
```

Out[25]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [26]:
```
Tags.iloc[[0,11,500]]                    # fetching rows of given indices
```

Out[26]:

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

Descriptive stas

In [27]:
```
Rating.head()                            # Top 5 rows of Rating data
```

Out[27]:

	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 [28]:
```
Rating['rating'].describe()              # describe 'rating' column from Rating data
```

Out[28]:
```
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 [29]:
```
Rating.describe()                    # Description of Rating columnn
```

Out[29]:

	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 [30]:
```
Rating['rating'].mean()                # mean of 'rating' column of Rating data
```

Out[30]:
```
3.5255285642993797
```

In [31]:
```
Rating.mean()                         # mean of each column of Rating data
```

Out[31]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [32]:
```
Rating['rating'].min()                 # minimum value of 'rating' column of Rating d
```

Out[32]:
```
0.5
```

In [33]:
```
Rating['rating'].max()                 # maximum value of 'rating' column of Rating d
```

Out[33]:
```
5.0
```

In [34]:
```
Rating['rating'].std()                  # standard deviation of 'rating' column of Ra
```

Out[34]:
```
1.051988919275684
```

In [35]:
```
Rating['rating'].mode()                 # mode value of 'rating' column of Rating dat
```

Out[35]:
```
0    4.0
Name: rating, dtype: float64
```

In [36]:
```
Rating.corr()                          # Correlation of each  column of Rating data
```

Out[36]:

	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 [37]:
```
Filter1 = Rating['rating']>10              # filtering/ checking, is any of the 'rating
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[37]:
```
False
```

In [38]:
```
Filter2 = Rating['rating']>0            # filtering /checking , is all the 'rating' fo
Filter2.all()
```

Out[38]:
```
True
```

Data cleaning and handeling missing data

In [39]:
```
Movies.head()            # Top 5 rows of Movies
```

Out[39]:

	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 [40]:
```
Movies.shape               # sape of movies
```

Out[40]:
```
(27278, 3)
```

In [41]:
```
Movies.isnull()            # checking is there any null values in Moveis data or not
```

Out[41]:

	movieId	title	genres
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
27273	False	False	False
27274	False	False	False
27275	False	False	False
27276	False	False	False
27277	False	False	False

27278 rows × 3 columns

In [42]:
```
Movies.isnull().any()              # checking is there null value in any columns of the
```

Out[42]:
```
movieId    False
title      False
genres     False
dtype: bool
```

In [43]:
```
Rating.shape                       # Shape of ratings
```

Out[43]:
```
(20000263, 3)
```

In [44]:
```
Rating.isnull().any()               # checking is there null value in any columns of th
```

Out[44]:
```
userId     False
movieId    False
rating     False
dtype: bool
```

In [45]:
```
Tags.shape                          # Shape of Tags
```

Out[45]:
```
(465564, 3)
```

In [46]:
```
Tags.isnull().any()                 # checking is there null value in any columns of t
```

Out[46]:
```
userId     False
movieId    False
tag         True
dtype: bool
```

In [47]:
```
Tags = Tags.dropna()                # dropping the null values form Tasgs
Tags
```

Out[47]:

	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
...	...	...	...
465559	138446	55999	dragged
465560	138446	55999	Jason Bateman
465561	138446	55999	quirky
465562	138446	55999	sad
465563	138472	923	rise to power

465548 rows × 3 columns

In [48]:
```
Tags.shape                    # check shape after droping null values
```

Out[48]:
```
(465548, 3)
```

In [49]:
```
Tags.isnull().any()           # checking is there null value in any columns of the mp
```

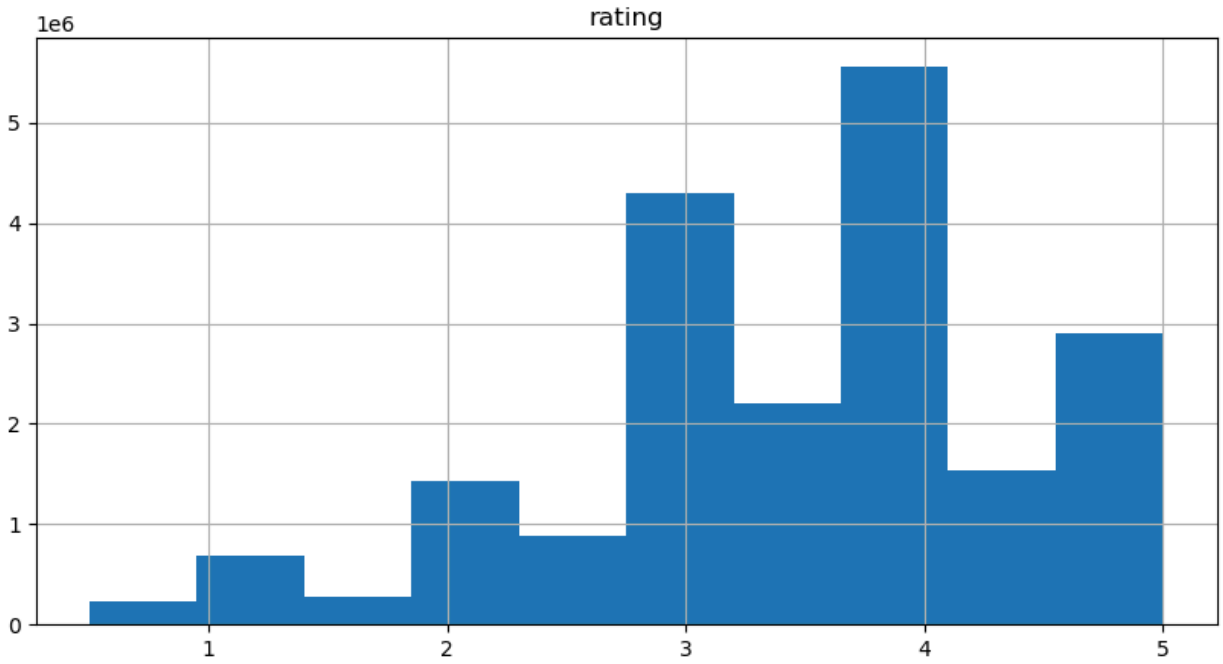
Out[49]:
```
userId     False
movieId    False
tag        False
dtype: bool
```

Data visualization

In [55]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns# matplotlib
```

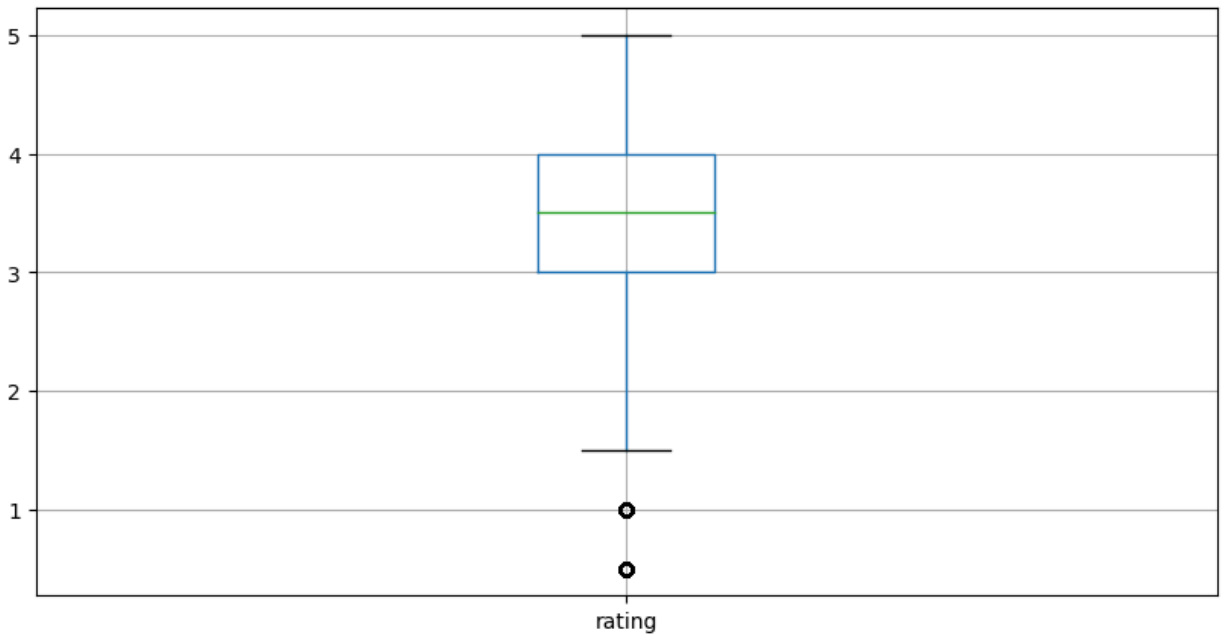
In [56]:
```
Viz1 = Rating.hist(column= 'rating',figsize = (10,5))
Viz1

# ploting visualization 1 of histogram of rating column from Rating dataseet with figu
```

Out[56]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



In [57]:
```
Viz2 = Rating.boxplot(column = 'rating', figsize =(10,5))
Viz2

# ploting visualization 2 of boxplot of rating column from Rating dataseet with figure
```

Out[57]:
```
<Axes: >
```



Slicing columns

In [58]:
```
Tags['tag'].head()          # Top 5 rows of 'tags' columns form Tags dataset
```

Out[58]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [59]:
```
Movies[['title','genres']].head()          # top 5 rows from 'title' and 'genres' col
```

Out[59]:

	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 [60]:
```
Rating[-10:]                    # # fetching rows from -10 th indices till last
```

Out[60]:

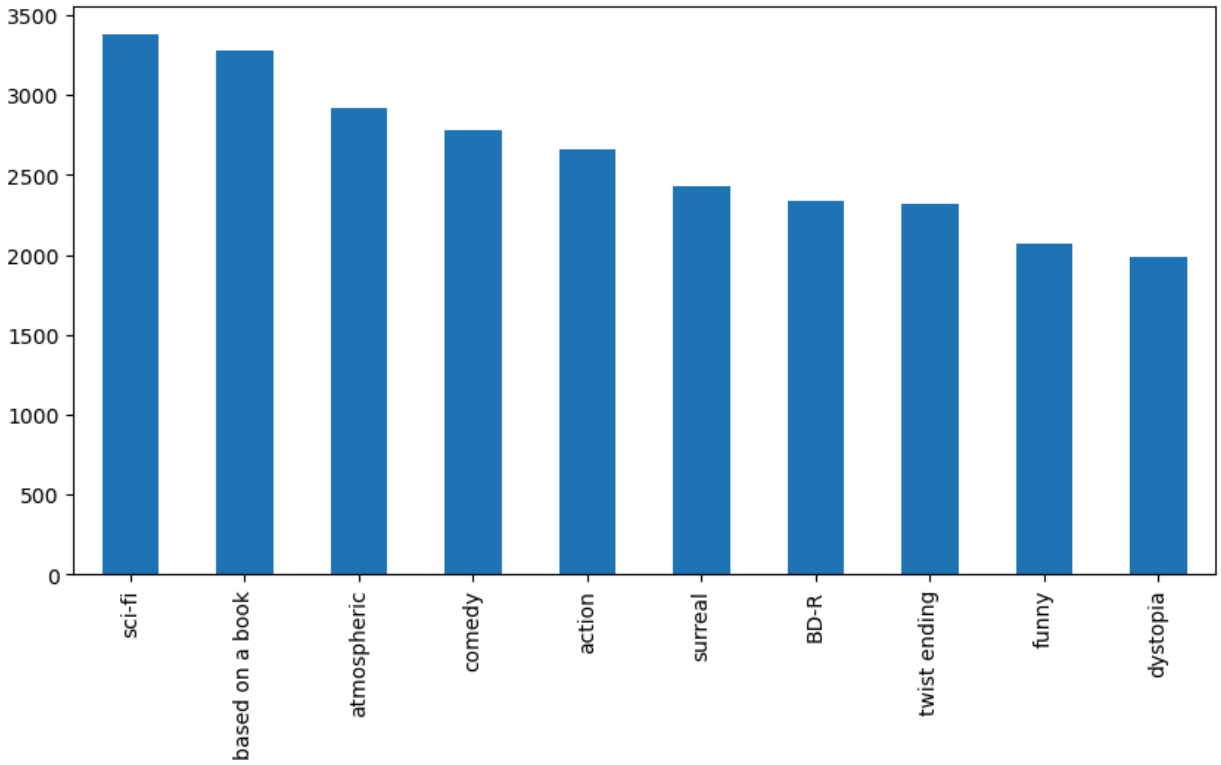
	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [61]:
```
Tag_counts = Tags['tag'].value_counts()
Tag_counts[-10:]

# Total tag count
```

Out[61]:
```
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: tag, dtype: int64
```

In [62]:
```
Tag_counts[:10].plot(kind='bar', figsize = (10,5))

# ploting graph of tag_counts
```

Out[62]:
```
<Axes: >
```



In [67]:
```
# fetching highly rated movies
highly_rated = Rating['rating'] >= 5.0
Rating[highly_rated][30:50]
```

Out[67]:

	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 [68]:

```
# fetching Action movies
act = Movies['genres'].str.contains('Action')
Movies[act][5:15]
```

Out[68]:

	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 [69]:
```
# first 15 action movies
Movies[act].head(15)
```

Out[69]:

	movieId	title	genres
5	6	Heat (1995)	Action|Crime|Thriller
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
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 [76]:
```
# using  group by
# give rating counts of movies wrt to rating
rating_count = Rating[['movieId','rating']].groupby('rating').count()
rating_count
```

Out[76]:

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 [79]:
```
# give average rating of movies group by id

avg_rating = Rating[['movieId','rating']].groupby('movieId').mean()
avg_rating.head()
```

Out[79]:

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

In [86]:
```
# movie count by movie id
movie_count = Rating[['movieId','rating']].groupby('movieId').count()
movie_count.head()
```

Out[86]:

movieId	rating
1	49695
2	22243
3	12735
4	2756
5	12161

In [88]:
```
# MErging tables
Tags.head()
```

Out[88]:

	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 [89]:
```
Movies.head()
```

Out[89]:

	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 [91]:
```
# mering the both tag and movies table
m1 = Movies.merge(Tags,on ='movieId', how ='inner')
m1.head()
```

Out[91]:

	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

In [93]:
```
# Give avg ratings of movies grouped by movied id without user id

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

Out[93]:

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

In [97]:
```
# make new data as box offic mergig avg_ratinng and movies

box_office= Movies.merge(avg_rating, on= 'movieId',how ='inner')
box_office
```

Out[97]:

	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
2	3	Grumpier Old Men (1995)	Comedy|Romance	3.151040
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2.861393
4	5	Father of the Bride Part II (1995)	Comedy	3.064592
...	...	...	...	...
26739	131254	Kein Bund für's Leben (2007)	Comedy	4.000000
26740	131256	Feuer, Eis & Dosenbier (2002)	Comedy	4.000000
26741	131258	The Pirates (2014)	Adventure	2.500000
26742	131260	Rentun Ruusu (2001)	(no genres listed)	3.000000
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.000000

26744 rows × 4 columns

In [98]:
```
# give highly rated box offic movies
HR = box_office['rating'] >= 4.0
box_office[HR][-5:]
```

Out[98]:

	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 [100…
```
# fetch adventure movies
Adv = box_office['genres'].str.contains('Adventure')
```

```
box_office[Adv][:5]
```

Out[100]:

	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 [103…

```
# which are both adventures and highly rated movies
box_office[Adv&HR][-5:]
```

Out[103]:

	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 [105…

```
# Average movie ratings
average_rating = Rating[['movieId','rating']].groupby('movieId', as_index=False).mean(
average_rating.tail()
```

Out[105]:

	movieId	rating
26739	131254	4.0
26740	131256	4.0
26741	131258	2.5
26742	131260	3.0
26743	131262	4.0

In [ ]:

```
```