Imdb rating analysis

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

In [67]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Importing the dataset

In [2]:
```
movies = pd.read_csv('movie.csv')
```

In [3]:
```
ratings = pd.read_csv('rating.csv')
```

In [4]:
```
tags = pd.read_csv('tag.csv')
```

shape of data

In [5]:
```
movies.shape
```

Out[5]:
```
(27278, 3)
```

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

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

In [7]:
```
tags.shape
```

Out[7]:
```
(465564, 4)
```

Top 5 rows of data set

In [8]:
```
movies.head()
```

Out[8]:

	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 [9]:
```
ratings.head()
```

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()
```

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

Deleting the time stamp attributes from ratings and tag

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

In [12]:
```
ratings.head()
```

Out[12]:

	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 [13]:
```
tags.head()
```

Out[13]:

	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

Data structures

In [14]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[14]:
```
pandas.core.series.Series
```

In [15]:
```
tags.head()
```

Out[15]:

	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 [16]:
```
print(row_0)
```

```
userId                18
movieId             4141
tag          Mark Waters
Name: 0, dtype: object
```

In [17]:
```
row_0.index
```
Out[17]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [18]:
```
tags.loc[2] # it gives row and column values
```
Out[18]:
```
userId              65
movieId            353
tag          dark hero
Name: 2, dtype: object
```

In [24]:
```
row_0['userId']
```
Out[24]:
```
18
```

In [26]:
```
'ratings' in row_0
```
Out[26]:
```
False
```

In [27]:
```
row_0.name
```
Out[27]:
```
0
```

In [29]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```
Out[29]:
```
'firstRow'
```

DataFrame

In [30]:
```
tags.head()
```
Out[30]:

	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

These are the top 5 rows from the tags adataframe

In [31]:
```
tags.index
```
Out[31]:
```
RangeIndex(start=0, stop=465564, step=1)
```

tags dataframe starting is 0 and ending row number is 465564.

In [32]:
```
tags.columns
```

Out[32]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

tags dataframe have 'userId', 'movieId', 'tag' these attributes in it.

In [35]:
```
tags.iloc[[6,8,234,7777]]
```

Out[35]:

	userId	movieId	tag
6	65	898	screwball comedy
8	65	1391	mars
234	129	80549	Unreal reactions
7777	1741	6390	great dancing

Descriptive Statistics

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

Out[37]:

	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 [38]:
```
ratings['rating'].describe()
```

Out[38]:
```
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 [39]:
```
ratings.describe()
```

Out[39]:

	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

As we can see in the ratings dataset that userid and movieid attribute don't give any insights regarding the data.

```
for the movie rating min rating is 0.5 and mean rating is 3.5 and maximum rating is
5.
```

In [40]:
```
ratings['rating'].mean()
```

Out[40]:
```
3.5255285642993797
```

In [42]:
```
ratings.mean()
```

Out[42]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

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

Out[43]:
```
0.5
```

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

Out[44]:
```
5.0
```

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

Out[45]:
```
1.051988919275684
```

In [47]:
```
ratings['rating'].mode()
```

Out[47]:
```
0    4.0
Name: rating, dtype: float64
```

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

Out[48]:

	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 [52]:
```
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[52]:
```
False
```

In [54]:
```
filter2 = ratings['rating']>0

print(filter2)
filter2.all()
```

```
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
```

Out[54]:
```
True
```

Data Cleaning: Handling Missing Data

In [55]:
```
movies.shape
```

Out[55]:
```
(27278, 3)
```

```
In [57]: movies.isnull().any().any()
Out[57]: False
```

This means we don't have null values in the movies dataset.

```
In [58]: ratings.shape
Out[58]: (20000263, 3)
```

```
In [59]: ratings.isnull().any().any()
Out[59]: False
```

```
In [60]: tags.shape
Out[60]: (465564, 3)
```

```
In [61]: tags.isnull().any().any()
Out[61]: True
```

As for ratings we don't have null values but for tags we have some null values which we have to drop.

Null values treatment

```
In [62]: tags = tags.dropna()
```

```
In [63]: tags.isnull().any().any()
Out[63]: False
```

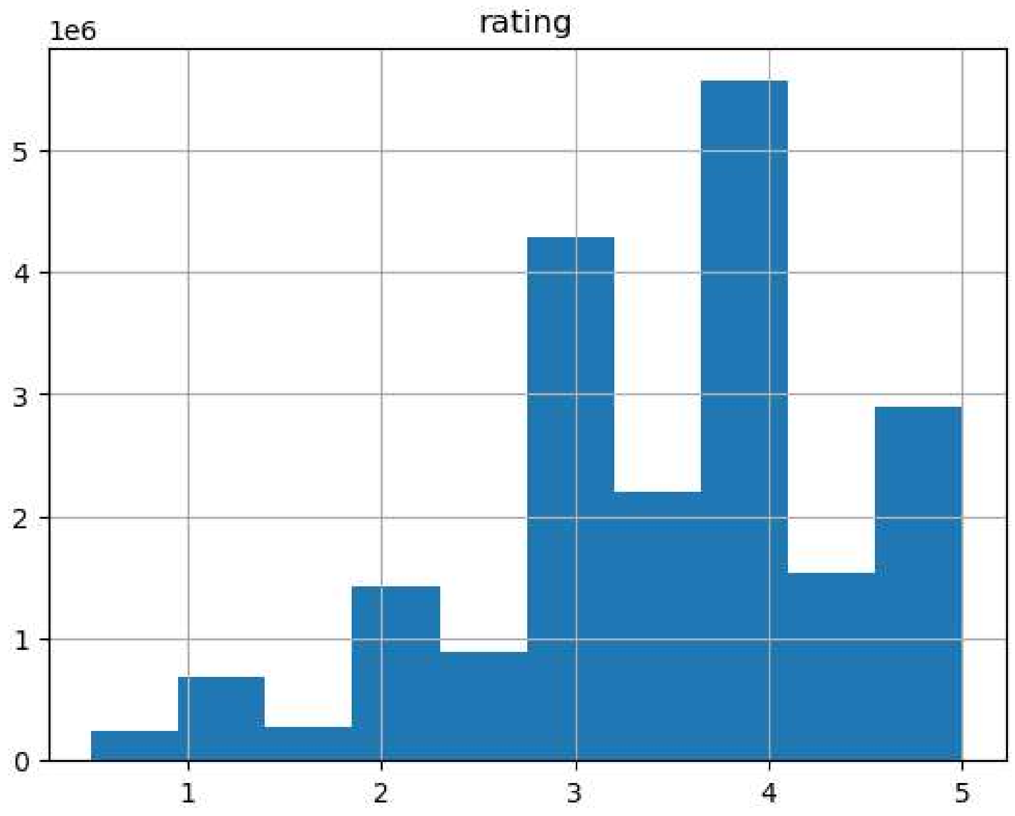
```
In [64]: tags.shape
Out[64]: (465548, 3)
```

Data Visualization

histogram

In [66]:
```
ratings.hist('rating')
```

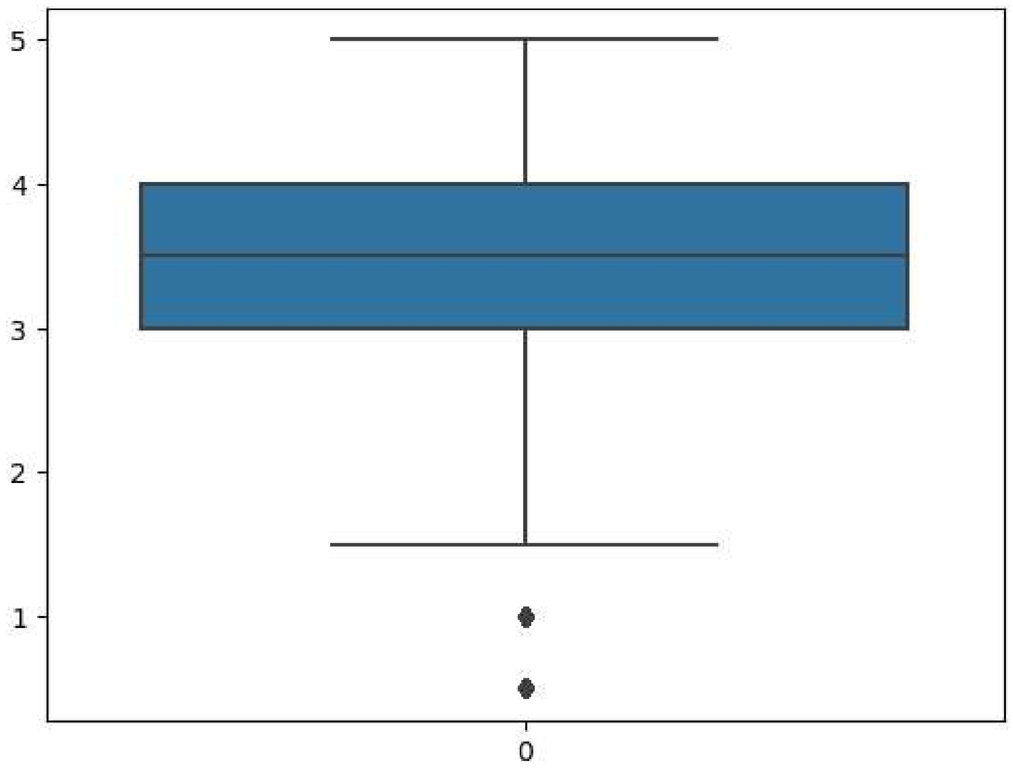
Out[66]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



Boxplot of rating attribute

In [68]:
```
sns.boxplot(ratings['rating'])
```

Out[68]:
```
<Axes: >
```



It shows that in rating column we have some outliers which we have to consider.

Slicing Out Columns

In [70]:
```
tags['tag'].head()
```

Out[70]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

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

Out[71]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [72]:
```
movies[['title', 'genres']].head()
```

Out[72]:

	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 [74]:
```
ratings[-8:]
```

Out[74]:

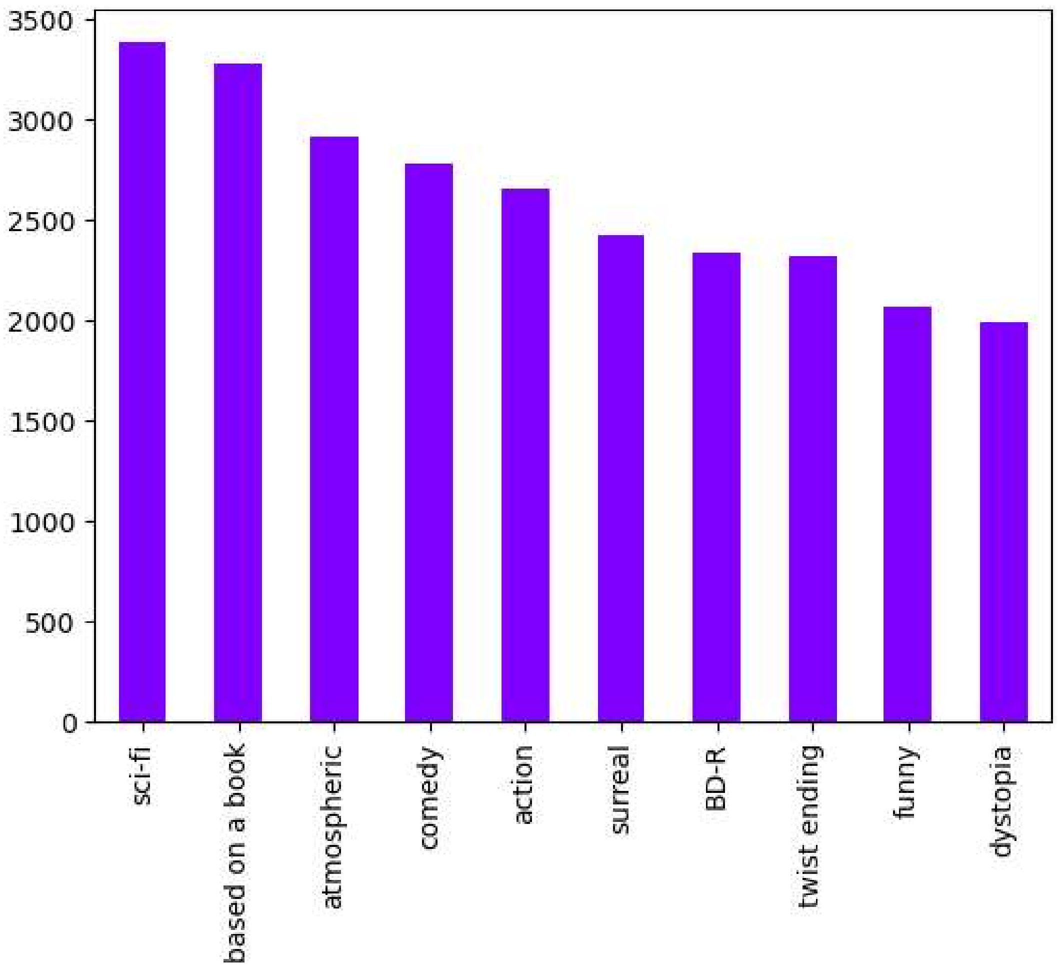
	userId	movieId	rating
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [78]:
```
tags_count = tags['tag'].value_counts()
tags_count[-10:]
```

Out[78]:
```
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 [84]: tags_count[:10].plot(kind='bar',colormap ='rainbow')

Out[84]: <Axes: >
```



Filters for Selecting Rows

In [89]:
```
above_average = ratings['rating']>=3.52

ratings[above_average][:10]
```

Out[89]:

	userId	movieId	rating
6	1	151	4.0
7	1	223	4.0
8	1	253	4.0
9	1	260	4.0
10	1	293	4.0
11	1	296	4.0
12	1	318	4.0
15	1	541	4.0
22	1	1036	4.0
23	1	1079	4.0

In [95]:
```
horror_movies = movies[movies['genres']=='Horror']
horror_movies[:10]
```

Out[95]:

	movieId	title	genres
175	177	Lord of Illusions (1995)	Horror
218	220	Castle Freak (1995)	Horror
393	397	Fear, The (1995)	Horror
723	735	Cemetery Man (Dellamorte Dellamore) (1994)	Horror
826	841	Eyes Without a Face (Yeux sans visage, Les) (1...	Horror
1083	1105	Children of the Corn IV: The Gathering (1996)	Horror
1105	1128	Fog, The (1980)	Horror
1230	1258	Shining, The (1980)	Horror
1293	1322	Amityville 1992: It's About Time (1992)	Horror
1294	1323	Amityville 3-D (1983)	Horror

These are the 10 Horror movies.

In [ ]: