IMDB ANALYSIS

Import Library

In [2]:

```
import pandas as pd
import numpy as np
```

Reading Datasets

In [3]:

```
movie = pd.read_csv(r'C:\Users\LENOVO\Desktop\Monty Datascien\movie.csv')
```

In [4]:

```
rating = pd.read_csv(r'C:\Users\LENOVO\Desktop\Monty Datascien\rating.csv')
```

In [5]:

```
tag = pd.read_csv(r'C:\Users\LENOVO\Desktop\Monty Datascien\tag.csv')
```

Understanding each DataFrame

In [6]:

```
movie.shape
```

Out[6]:

```
(27278, 3)
```

In [7]:

```
rating.shape
```

Out[7]:

```
(20000263, 4)
```

In [8]:

```
tag.shape
```

Out[8]:

```
(465564, 4)
```

In [9]:

```
movie.head()
```

Out[9]:

	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 [10]:

```
rating.head()
```

Out[10]:

	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 [11]:

```
tag.head()
```

Out[11]:

	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 [12]:

```
type(movie)
```

Out[12]:

```
pandas.core.frame.DataFrame
```

In [13]:

```
type(rating)
```

Out[13]:

```
pandas.core.frame.DataFrame
```

In [14]:

```
type(tag)
```

Out[14]:

```
pandas.core.frame.DataFrame
```

In [17]:

```
movie.columns
```

Out[17]:

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

In [18]:

```
rating.columns
```

Out[18]:

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

In [19]:

```
tag.columns
```

Out[19]:

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

In [77]:

```
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27278 entries, 0 to 27277
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   movieId  27278 non-null  int64 
 1   title    27278 non-null  object
 2   genres   27278 non-null  object
dtypes: int64(1), object(2)
memory usage: 639.5+ KB
```

In [78]:

```
rating.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 [79]:

```
tag.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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
```

Removing Timestamp

```
del rating['timestamp']
```

In [21]:

```
del tag['timestamp']
```

In [23]:

```
rating.columns
```

Out[23]:

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

In [24]:

```
tag.columns
```

Out[24]:

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

Data Structure

In [30]:

```
tag.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

In [25]:

```
r0 = tag.iloc[0]
```

In [27]:

```
r0
```

Out[27]:

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

In [29]:

```
r0['userId']
```

Out[29]:

```
18
```

In [31]:

```
'rating' in r0 # Membership operator
```

Out[31]:

```
False
```

In [32]:

```
tag.iloc[[0,11,150]]
```

Out[32]:

	userId	movieId	tag
0	18	4141	Mark Waters
11	65	1783	noir thriller
150	129	3556	visually appealing

In [34]:

```
tag.loc[5:9]
```

Out[34]:

	userId	movieId	tag
5	65	668	bollywood
6	65	898	screwball comedy
7	65	1248	noir thriller
8	65	1391	mars
9	65	1617	neo-noir

In [35]:

```
tag.iloc[5:9]
```

Out[35]:

	userId	movieId	tag
5	65	668	bollywood
6	65	898	screwball comedy
7	65	1248	noir thriller
8	65	1391	mars

In [37]:

```
tag.loc[[5,15,20]]
```

Out[37]:

	userId	movieId	tag
5	65	668	bollywood
15	65	2662	mars
20	65	6539	treasure

In [82]:

```
tag.loc[tag['tag']=='bollywood']
```

Out[82]:

	userId	movieId	tag
5	65	668	bollywood
19	65	5135	bollywood
30	65	51884	bollywood
1970	910	6683	bollywood
7858	1741	6683	bollywood
...	...	...	...
459306	135155	63082	bollywood
459984	135834	6696	bollywood
460219	136015	5135	bollywood
460277	136015	63082	bollywood

In [83]:

```
tag.iloc[tag['tag']=='bollywood']tag.loc[tag['tag']=='bollywood']
```

```
  Input In [83]
    tag.iloc[tag['tag']=='bollywood']tag.loc[tag['tag']=='bollywood']
                                      ^
SyntaxError: invalid syntax
```

In [84]:

```
#iloc is used for integer indexing.
#Loc we can pass name of the row or column which we want to select.
```

Descriptive Statistics

In [38]:

```
movie.describe()
```

Out[38]:

	movieId
count	27278.000000
mean	59855.480570
std	44429.314697
min	1.000000
25%	6931.250000
50%	68068.000000
75%	100293.250000
max	131262.000000

In [39]:

```
rating.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

In [40]:

```
tag.describe()
```

Out[40]:

	userId	movieId
count	465564.000000	465564.000000
mean	68712.354263	32627.762920
std	41877.674053	36080.241157
min	18.000000	1.000000
25%	28780.000000	2571.000000
50%	70201.000000	7373.000000
75%	107322.000000	62235.000000
max	138472.000000	131258.000000

In [41]:

```
rating['rating'].mean()
```

Out[41]:

```
3.5255285642993797
```

In [42]:

```
rating['rating'].min()
```

Out[42]:

```
0.5
```

In [43]:

```
rating['rating'].max()
```

Out[43]:

```
5.0
```

In [44]:

```
rating['rating'].median()
```

Out[44]:

```
3.5
```

In [45]:

```
rating['rating'].mode()
```

Out[45]:

```
0    4.0
Name: rating, dtype: float64
```

In [46]:

```
rating['rating'].std()
```

Out[46]:

```
1.051988919275684
```

In [48]:

```
rating.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 [49]:

```
f = rating['rating'] > 4
```

In [50]:

```
f
```

Out[50]:

```
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
```

Missing Values

In [51]:

```
movie.isnull().sum()
```

Out[51]:

```
movieId    0
title      0
genres     0
dtype: int64
```

In [52]:

```
rating.isnull().sum()
```

Out[52]:

```
userId     0
movieId    0
rating     0
dtype: int64
```

In [53]:

```
tag.isnull().sum() # We got 16 Missing Values in tag
```

Out[53]:

```
userId      0
movieId     0
tag        16
dtype: int64
```

In [54]:

```
tag = tag.dropna() # We have removed the missing values
```

In [55]:

```
tag.isnull().sum()
```

Out[55]:

```
userId     0
movieId    0
tag        0
dtype: int64
```

Data Visualization

Importing Libraries

In [56]:

```
import matplotlib.pyplot as plt # used for normal visualization
import seaborn as sns           # used for advance visualization

%matplotlib inline
plt.rcParams['figure.figsize'] = 10,5

import warnings
warnings.filterwarnings('ignore')   # use for not getting os error
```

In [58]:

```
rating.head()
```

Out[58]:

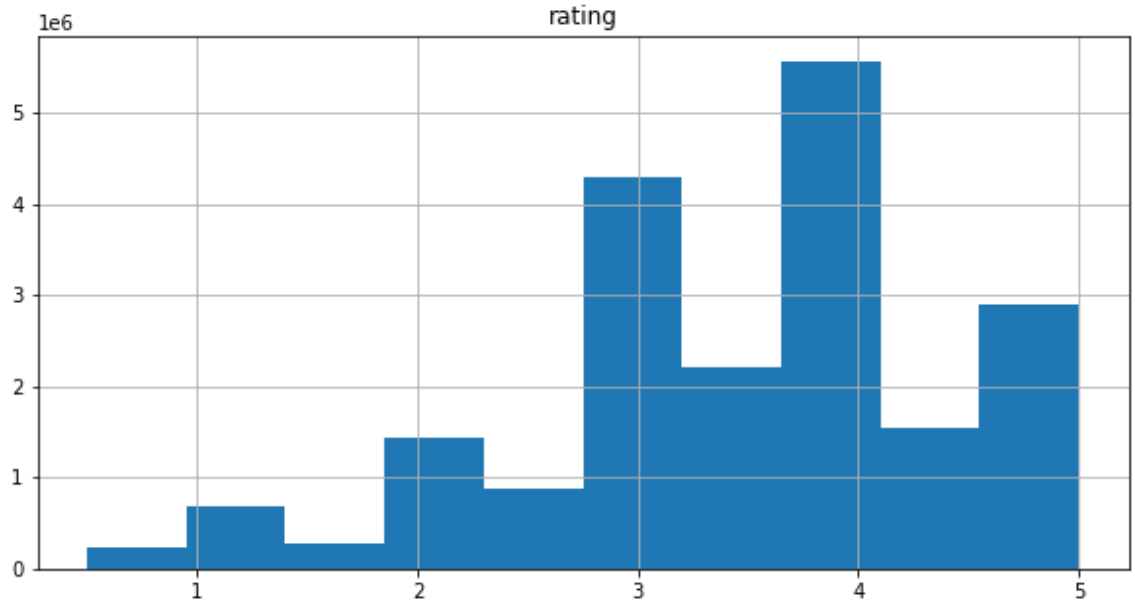
	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5

In [61]:

```
rating.hist(column='rating')
```

Out[61]:

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



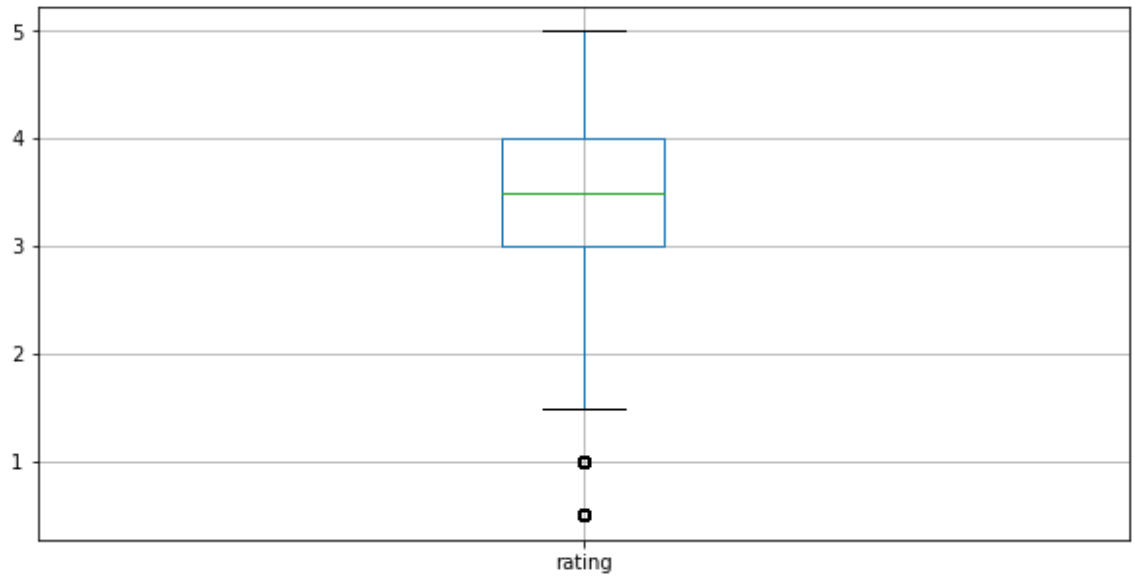
From this graph we understood that most of movies are rated between 2.8-4.1

In [62]:

```
rating.boxplot(column='rating')
```

Out[62]:

<AxesSubplot:>



Average rating is 3.5 and most of the movie ratings are below the average value

In [67]:

```
tag.head()
```

Out[67]:

	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 [63]:

```
tag_count = tag['tag'].value_counts()
```

In [64]:

```
tag_count[:10]
```

Out[64]:

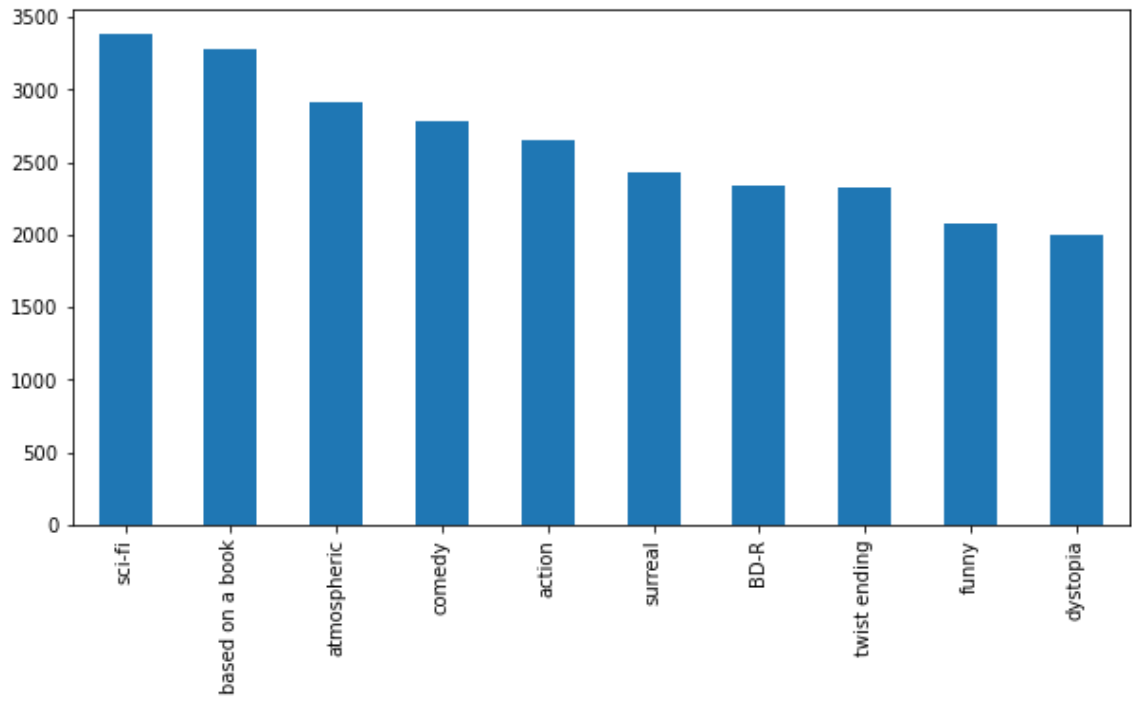
```
sci-fi             3384
based on a book    3281
atmospheric        2917
comedy             2779
action             2657
surreal            2427
BD-R               2334
twist ending       2323
funny              2072
dystopia           1991
Name: tag, dtype: int64
```

In [76]:

```
tag_count[:10].plot(kind='bar')
```

Out[76]:

```
<AxesSubplot:>
```



In [ ]: