import the dataset

In [16]:

```
import pandas as pd
```

read the data set

In [28]:

```
movies=pd.read_csv(r"C:\Users\user\Documents\movie.csv",sep=',')
print(movies.shape)
movies.head(20)
```

```
(27278, 3)
```

Out[28]:

	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
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Adventure|Animation|Children
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Crime|Drama
16	17	Sense and Sensibility (1995)	Drama|Romance
17	18	Four Rooms (1995)	Comedy
18	19	Ace Ventura: When Nature Calls (1995)	Comedy
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller

In [42]:

```
tag=pd.read_csv(r"C:\Users\user\Documents\tag.csv",sep=',')
tag.head()
```

Out[42]:

	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 [40]:

```
rating=pd.read_csv(r"C:\Users\user\Documents\rating.csv",sep=',')
rating.head()
```

Out[40]:

	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

for current analysis, we will remove timestamp

In [46]:

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

In [50]:

```
rating.head()
```

Out[50]:

	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 [52]:

```
tag.head()
```

Out[52]:

	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

series

In [68]:

```python
row_0=tag.iloc[0] # extract 0th row
print(type(row_0))
print(row_0)
```

```
<class 'pandas.core.series.Series'>
userId                18
movieId             4141
tag          Mark Waters
Name: 0, dtype: object
```

In [60]:

```python
row_0=tag.iloc[0,1] # extract 0th row 1st col value
row_0
```

Out[60]:

```
4141
```

In [82]:

```python
row_0.index
```

Out[82]:

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

In [84]:

```python
row_0['userId'] # gives 1st value in userid column
```

Out[84]:

```
18
```

In [86]:

```python
'rating' in row_0 # since row_0 having tag df values rating col is not present s
```

Out[86]:

```
True
```

In [92]:

```python
row_0.name
```

Out[92]:

```
0
```

In [100…

```python
row_0=row_0.rename('firstrow')
row_0.name
```

Out[100…

```
'firstrow'
```

data frames

In [102…

```python
tag.head()
```

Out[102…

	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 [108…

```
tag.index # gives rows columns size
```

Out[108…

```
RangeIndex(start=0, stop=465564, step=1)
```

In [110…

```
tag.columns # gives columns names
```

Out[110…

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

In [114…

```
tag.iloc[[0,11,500]] # to select specific rows
```

Out[114…

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

descriptive statistics

In [116…

```
rating['rating'].describe() # describe talks 8 things as below and we mention on
```

Out[116…

```
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 [118…

```
rating.describe() # describes about all columns
```

Out[118…

	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 [122…

```
rating['rating'].mean() # find mean as mentioned 'rating' so find mean of rating
```

Out[122…

```
3.5255285642993797
```

In [124…

```
rating.mean() # find mean of all columns
```

Out[124…

```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [126…

```
rating['rating'].min() # min value in rating col
```

Out[126…

```
0.5
```

In [130…

```
rating['rating'].max() # max value in rating col
```

Out[130…

```
5.0
```

In [132…

```
rating['rating'].std() # taking the square root of the sum of the squared differ
```

Out[132…

```
1.051988919275684
```

In [ ]:

```
2
```

In [134…

```
rating['rating'].mode() # Get the mode(s) of each element along the selected axi
                        # The mode of a set of values is the value that appears
```

Out[134…

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

In [136…

```
rating.corr() #Compute pairwise correlation of columns, excluding NA/null values
```

Out[136...

	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 [138...

```
filter1=rating['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[138...

```
False
```

In [140...

```
filter2=rating['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[140...

```
True
```

Data Cleaning:handling missing data

In [144...

```
movies.shape
```

Out[144...

```
(27278, 3)
```

In [148...

```
movies.isnull().any()
```

Out[148...

```
movieId    False
title      False
genres     False
dtype: bool
```

In [150…

```
movies.isnull().any().any() # no null values in dataframe
```

Out[150…

```
False
```

In [152…

```
rating.shape
```

Out[152…

```
(20000263, 3)
```

In [154…

```
rating.isnull().any()
```

Out[154…

```
userId     False
movieId    False
rating     False
dtype: bool
```

In [156…

```
rating.isnull().any().any() # no null values
```

Out[156…

```
False
```

In [158…

```
tag.shape
```

Out[158…

```
(465564, 3)
```

In [160…

```
tag.isnull().any()
```

Out[160…

```
userId     False
movieId    False
tag         True
dtype: bool
```

In [162…

```
tag.isnull().any().any() # True menas here indicates We have some tags which are
```

Out[162…

```
True
```

In [164…

```
tag=tag.dropna() # Remove missing values.
```

In [166…

```
tag.isnull().any()
```

Out[166…

```
userId     False
movieId    False
tag        False
dtype: bool
```

In [168…

```
tag.isnull().any().any() # so more null values
```

Out[168…

```
False
```

In [170…

```
tag.shape # we can observe after removing null values rows size got decresed
```

Out[170…

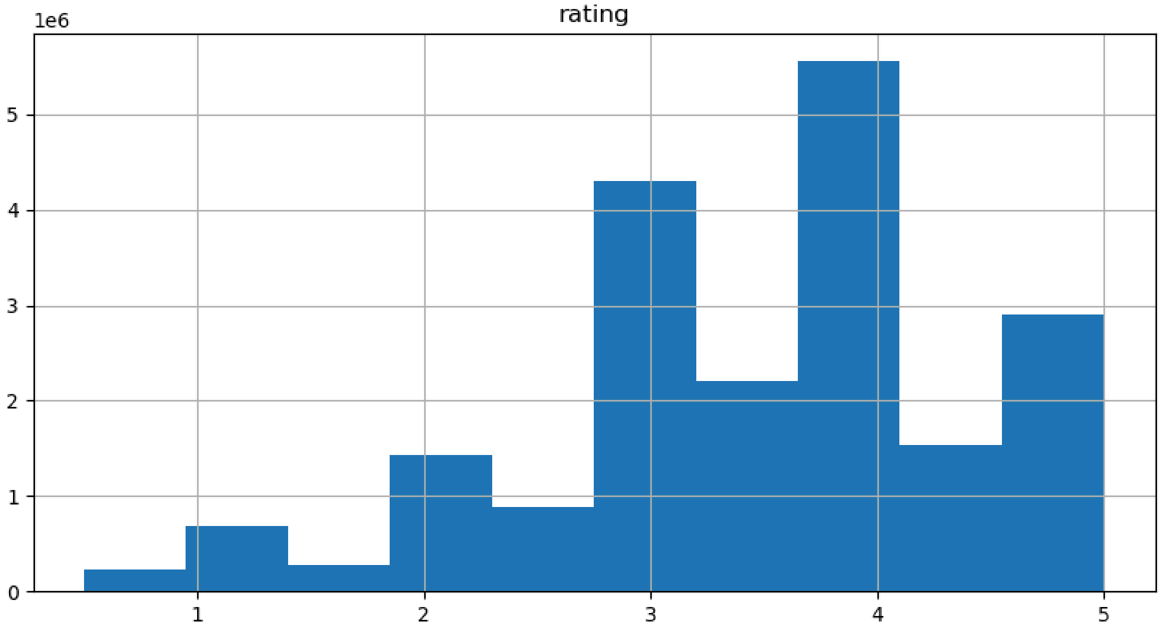
```
(465548, 3)
```

Data Visualization

In [180…

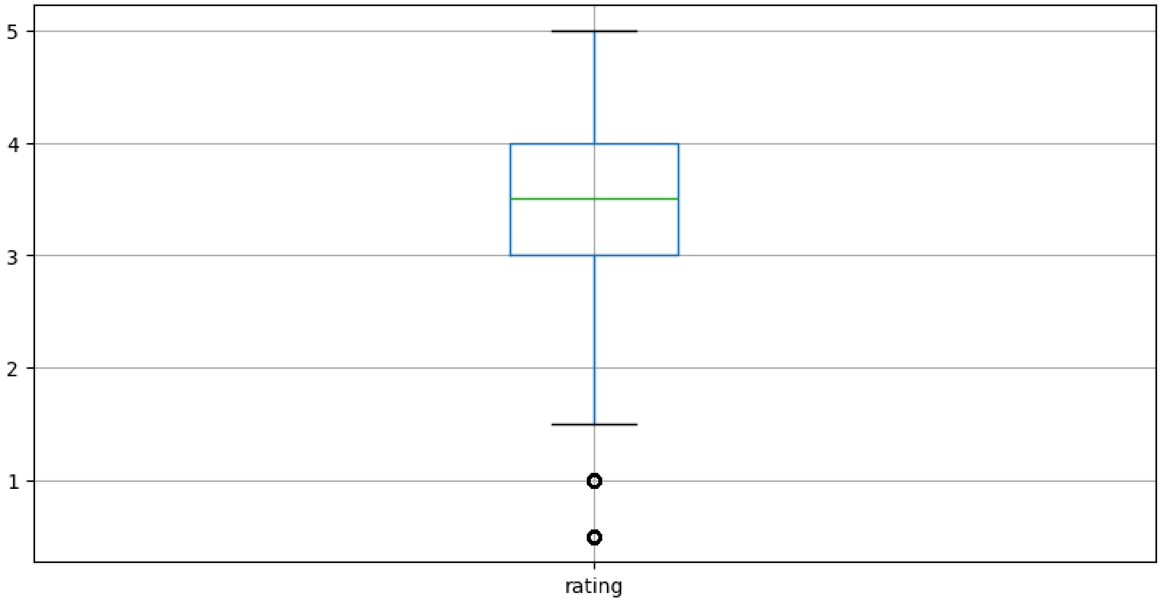
```
%matplotlib inline
rating.hist(column='rating',figsize=(10,5))
```

```
Out[180… array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



```
In [182… rating.boxplot(column='rating',figsize=(10,5))
```

```
Out[182… <Axes: >
```



slicing out columns

```
In [184… tag['tag'].head()
```

```
Out[184… 0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

```
In [188… movies[['title','genres']].head()
```

Out[188…

	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 [192…

```
rating[-10:] # prints last 10 rows
```

Out[192…

	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 [204…

```
tag_counts=tag['tag'].value_counts() # Return a Series containing the frequency
tag_counts[-10:]
```

Out[204…

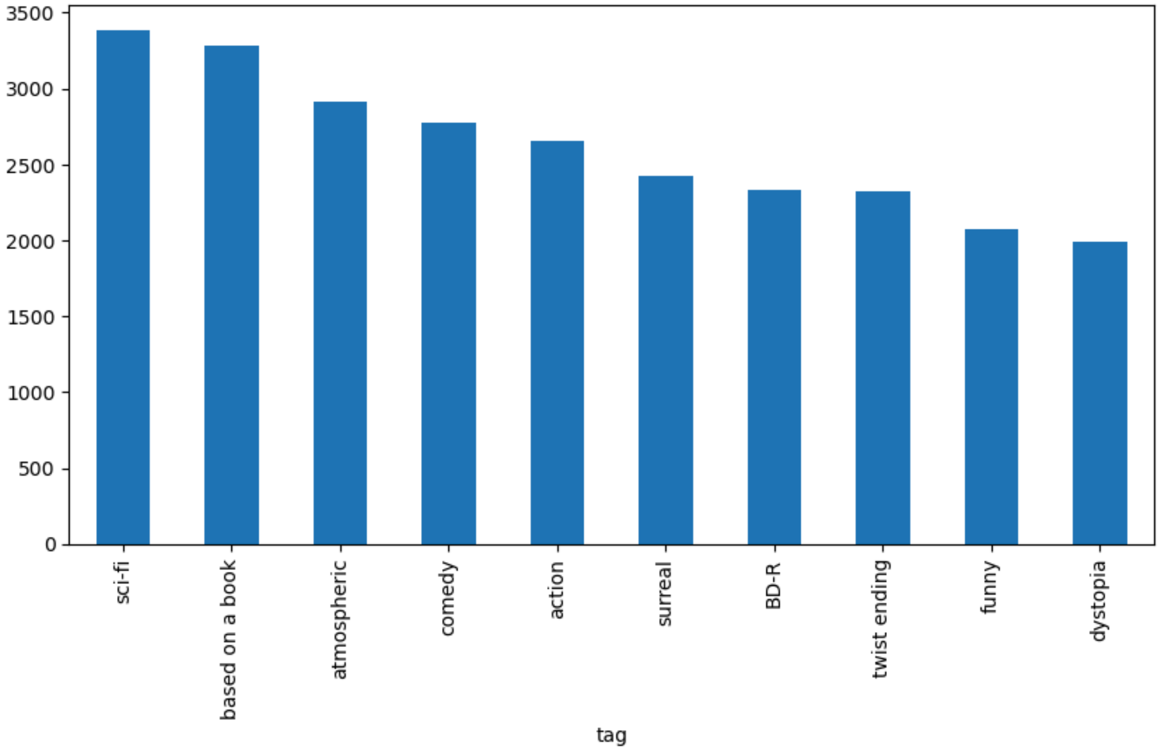
```
tag
missing child                  1
Ron Moore                      1
Citizen Kane                   1
mullet                         1
biker gang                     1
Paul Adelstein                 1
the wig                        1
killer fish                    1
genetically modified monsters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

In [201…

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

Out[201…

```
<Axes: xlabel='tag'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: