import the dataset

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

Read the dataset

In [2]:
```
movies=pd.read_csv(r'C:\Users\arati\Downloads\archive\movie.csv', sep=',')
```

In [3]:
```
print(type(movies))
```

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

In [4]:
```
movies.head(20)
```

Out[4]:

	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 [5]:
```
ratings=pd.read_csv(r'C:\Users\arati\Downloads\archive\rating.csv')
```

In [6]:
```
type(ratings)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
ratings.head()
```

Out[7]:

	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 [8]:
```
tags=pd.read_csv(r'C:\Users\arati\Downloads\archive\tag.csv')
```

In [9]:
```
print(type(tags))
```

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

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

For Current analysis, we will remove timestamp

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

In [12]:
```
ratings.columns #see the column names
```

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

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

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

Data Structures:

Series

In [14]:
```
row_0=tags.iloc[0] #iloc -- for index location
```

In [15]:
```
type(row_0) #see the type
```

Out[15]:
```
pandas.core.series.Series
```

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]:
```
row_0['userId']
```

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

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

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

In [20]:
```
row_0.name
```

Out[20]:
```
0
```

In [21]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[21]:
```
'firstRow'
```

DataFrames

In [22]:
```
tags.head()
```

Out[22]:

	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 [23]:
```
tags.index #showes the indexes
```

Out[23]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [24]:
```
tags.columns #shows the column names
```

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

In [25]:
```
tags.iloc[[0,11,500]] #shows the index position
```

Out[25]:

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

Descriptive Statistics

In [26]:
```
# Let's look how the ratinngs are distributed
```

In [27]:
```
ratings['rating'].describe()
```

Out[27]:
```
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 [28]:
```
ratings.describe()
```

Out[28]:

	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 [29]:
```
ratings['rating'].mean() #evaluate the avg
```

Out[29]:
```
3.5255285642993797
```

In [30]:
```
ratings.mean()
```

Out[30]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [31]:
```
ratings['rating'].min() #returns the lowest number
```

Out[31]:
```
0.5
```

In [32]:
```
ratings['rating'].max() #returns the highest number
```

Out[32]:
```
5.0
```

In [33]:
```
ratings['rating'].std() #compute the standard deviation
```

Out[33]:
```
1.051988919275684
```

In [34]:
```
ratings['rating'].mode() #value that repeatedly occuring in a given set
```

Out[34]:
```
0    4.0
Name: rating, dtype: float64
```

In [35]:
```
ratings.corr() #The pairwise correlation of all collumns
```

Out[35]:

	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 [36]:
```
filter1 = ratings['rating'] > 10 #filtering the ratings which is greater than 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[36]:
```
False
```

In [37]:
```
filter2 = ratings['rating']
filter2.all()
```

Out[37]:
```
True
```

Data Cleaning: Handling Missing data

In [38]:
```
movies.shape #shows the number of rows and columns
```

Out[38]:
```
(27278, 3)
```

In [39]:
```
movies.isnull() #showes true if there is any missing values
```

Out[39]:

	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 [40]:
```
movies.isnull().any().any() #There is no Null values
```

Out[40]:
```
False
```

In [41]:
```
ratings.shape#shows the number of rows and columns
```

Out[41]:
```
(20000263, 3)
```

In [42]:
```
ratings.isnull().any().any() #There is no null values
```

Out[42]:
```
False
```

In [43]:
```
tags.shape  #shows the number of rows and columnstags.shape
```

Out[43]:
```
(465564, 3)
```

In [44]:
```
tags.isnull().any().any() #we have some tage which are NULL
```

Out[44]:
```
True
```

In [45]:
```
tags = tags.dropna() #drop the missing values
```

In [46]:
```
tags
```

Out[46]:

	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 [47]:
```
tags.isnull().any().any() # Now there is no Null values
```

Out[47]:
```
False
```

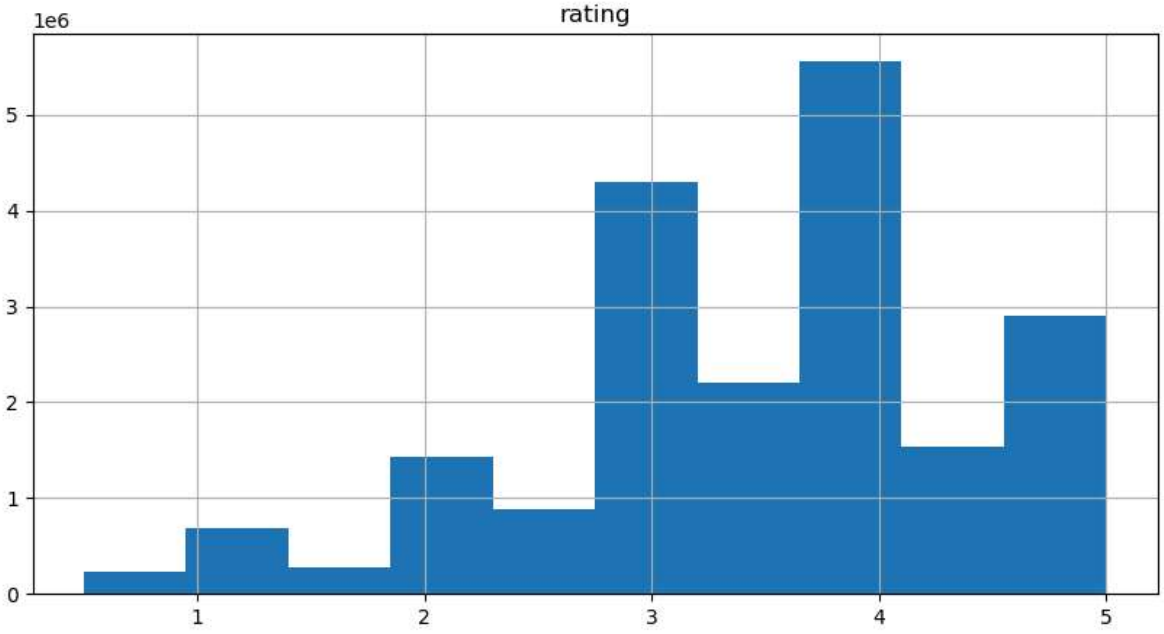
In [48]:
```
tags.shape #shows the number of rows and columns
# No NULL values..Number of lines have reduced
```

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

Data Visualization

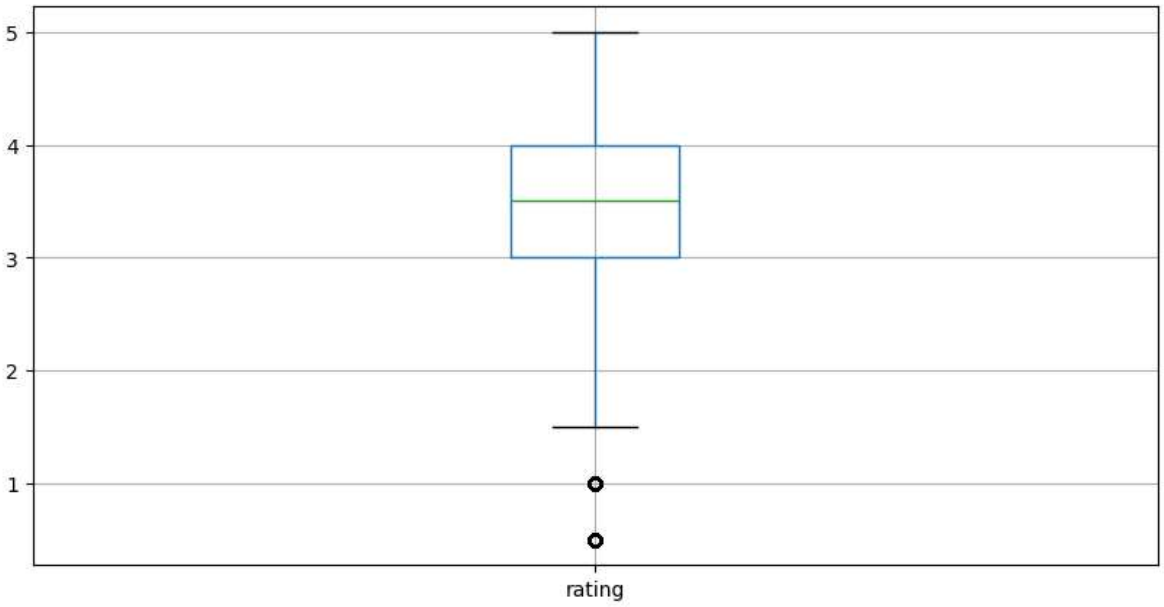
In [49]:
```
%matplotlib inline

ratings.hist(column='rating', figsize=(10,5))
```

Out[49]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



In [50]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[50]: <Axes: >



Slicing Out Columns

In [51]:
```
tags['tag'].head()
```

Out[51]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [52]:
```
movies[['title','genres']].head()
```

Out[52]:

	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 [54]:
```
ratings
```

Out[54]:

	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 [53]:
```
ratings[-10:]
```

Out[53]:

	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

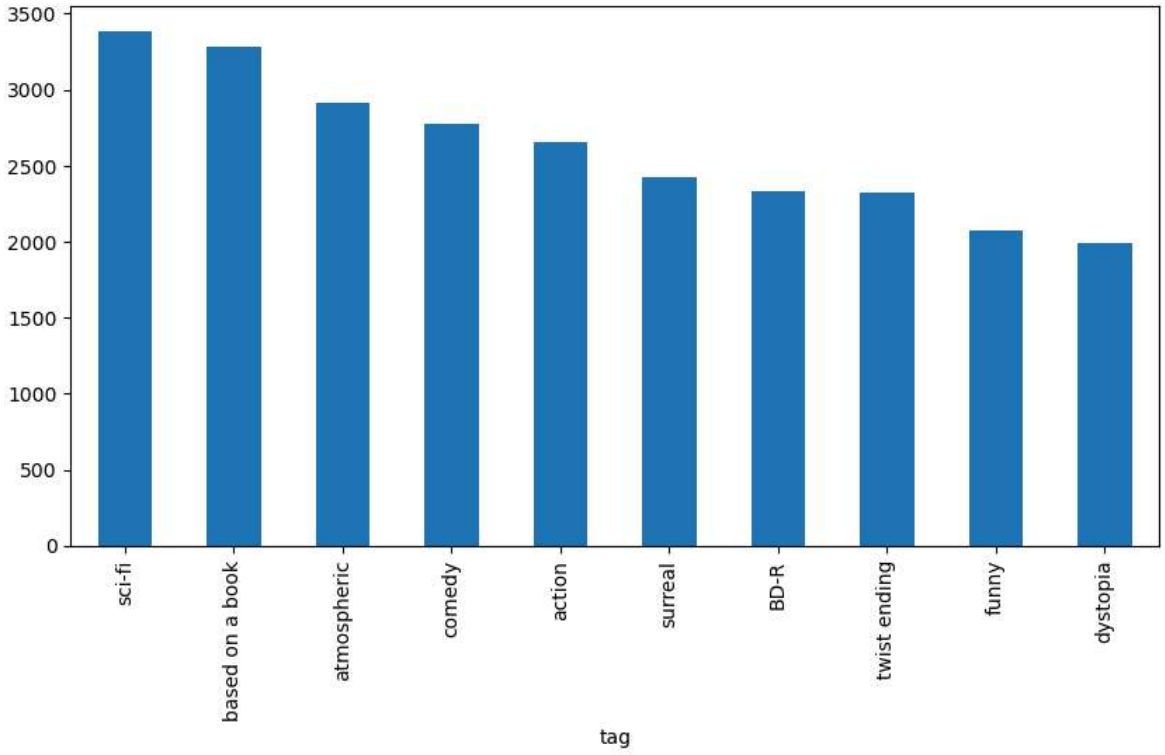
In [59]:
```
tag_counts = tags['tag'].value_counts() #returns Series containing counts of uni
#The resulting object will be in descending order so that the first element is t
#Excludes NA values by default.
```

In [61]:
```
tag_counts[-10:]
```

Out[61]:
```
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 [67]:
```
tag_counts[:10].plot(kind='bar',figsize=(10,5))
```

Out[67]:
```
<Axes: xlabel='tag'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: