Kaggle IMDB Dataset Practice - Tag, Movie and Rating Csv

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

In [5]:
```
'''
movies.csv : movieId, title, genres

ratings.csv : userId,movieId,rating, timestamp

tags.csv : userId,movieId, tag, timestamp

'''
```

Out[5]:
```
'\nmovies.csv : movieId, title, genres\n\nratings.csv : userId,movieId,rating, timest
amp\n\ntags.csv : userId,movieId, tag, timestamp\n\n'
```

In [13]:
```
movies = pd.read_csv(r'D:\00 - Naresh IT\1 - Materials\Jypter\10 - Kaggle IMDB Dataset
movies.head()
```

Out[13]:

	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 [20]:
```
movies.shape
```

Out[20]:
```
(27278, 3)
```

In [4]:
```
movies
```

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
...	...	...	...
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 [15]:
```
ratings = pd.read_csv(r'D:\00 - Naresh IT\1 - Materials\Jypter\10 - Kaggle IMDB Datase
ratings.head()
```

Out[15]:

	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 [22]:
```
ratings.shape
```

Out[22]:
```
(20000263, 4)
```

In [19]:
```
tags = pd.read_csv(r'D:\00 - Naresh IT\1 - Materials\Jypter\10 - Kaggle IMDB Dataset\I
                   parse_dates=['timestamp'])
tags.head()
```

Out[19]:

	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 [8]:
```
tags
```

Out[8]:

	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 [21]:
```
tags.shape
```

Out[21]:
```
(465564, 4)
```

In [9]:
```
movies.head(20)
```

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
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 [10]:
```
print(type(movies))
```

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

In [11]:
```
tags.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]:
```
ratings.head()
```

Out[12]:

	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 [23]:
```
del ratings['timestamp']
del tags['timestamp']
```

In [26]:
```
ratings.head(2)
```

Out[26]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5

In [25]:
```
tags.head(2)
```

Out[25]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero

In [27]:
```
movies.head(2)
```

Out[27]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy

In [28]:
```
movies.columns
```

Out[28]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [29]:
```
ratings.columns
```

Out[29]:
```
Index(['userId', 'movieId', 'rating'], dtype='object')
```

In [30]:
```
tags.columns
```

Out[30]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

loc and iloc properties in Pandas Dataframe - Ex:

In [ ]:
```
# Pandas DataFrame loc Property - dataframe.loc[row, column)

'''
loc:
----

The loc property gets, or sets, the value(s) of the specified labels.

Specify both row and column with a label.

'''
```

In [42]:
```
import pandas as pd

data = [[50,True],[40,False],[30,False]]
label_rows = ["Goud","Goud P","Chetan"]
label_cols = ["age","qualified"]

df = pd.DataFrame(data, label_rows,label_cols)

# One row and column
print(df.loc["Goud P","age"])
print("----------------------")

# To access more than one row, use double brackets and specify the labels, separated b
print(df.loc[["Goud", "Chetan"], ["age", "qualified"]])
print("----------------------")

# slice of the DataFrame with from and to labels, separated by a colon:
print(df.loc["Goud":"Goud P"])
print("----------------------")
```

```
40
----------------------
        age  qualified
Goud     50       True
Chetan   30      False
----------------------
        age  qualified
Goud     50       True
Goud P   40      False
----------------------
```

In [ ]:
```
# Pandas DataFrame iloc Property - dataframe.iloc[row, column)

'''
iloc:
----

The iloc property gets, or sets, the value(s) of the specified indexes.

Specify both row and column with an index.

'''
```

In [50]:
```
import pandas as pd

data = [[50, True], [40, False], [30, False]]
```

```
df = pd.DataFrame(data)

print(df.iloc[1,0])
print('-----------------------')

print(df.iloc[[0, 2]])
print('-----------------------')


print(df.iloc[[0, 2], [0, 1]])
print('-----------------------')

print(df.iloc[0:2])
print('-----------------------')
```

```
40
-----------------------
    0      1
0  50   True
2  30  False
-----------------------
    0      1
0  50   True
2  30  False
-----------------------
    0      1
0  50   True
1  40  False
-----------------------
```

In [ ]:
```
# Data Structures - Series # (A Pandas Series is like a column in a table)
```

In [53]:
```
import pandas as pd

a = [1,5,2]
myvar = pd.Series(a)
print(myvar)
print('-----------')
print(myvar[0])
```

```
0    1
1    5
2    2
dtype: int64
-----------
1
```

In [ ]:
```
'''
Labels
If nothing else is specified, the values are labeled with their index number. First va

This label can be used to access a specified value
'''
```

In [57]:
```
import pandas as pd

a = [1, 5, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])
```

```
print(myvar)
print('-----------')
print(myvar['y'])
```

```
x    1
y    5
z    2
dtype: int64
-----------
5
```

In [58]:
```
tags
```

Out[58]:

	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 [59]:
```
tags.head()
```

Out[59]:

	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 [60]:
```
tags.iloc[0] # It will give the information of a row
```

Out[60]:
```
userId             18
movieId          4141
tag        Mark Waters
Name: 0, dtype: object
```

In [61]:
```
tags.iloc[2]
```

Out[61]:
```
userId            65
movieId          353
tag         dark hero
Name: 2, dtype: object
```

In [62]:
```
tags.iloc[2:10]
```

Out[62]:

	userId	movieId	tag
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero
5	65	668	bollywood
6	65	898	screwball comedy
7	65	1248	noir thriller
8	65	1391	mars
9	65	1617	neo-noir

In [67]:
```
tags.index
```

Out[67]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [68]:
```
tags.iloc[0].index
```

Out[68]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [63]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[63]:
```
pandas.core.series.Series
```

In [64]:
```
print(row_0)
```

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

In [65]:
```
row_0.index
```

Out[65]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [66]:
```
row_0['userId']
```

Out[66]:
```
18
```

In [70]:
```
'rating' in row_0
```

Out[70]:
```
False
```

In [71]:
```
row_0.name
```

Out[71]:
```
0
```

In [72]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[72]:
```
'firstRow'
```

In [ ]:
```
# Data Frames
'''
A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a
'''
```

In [73]:
```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

In [74]:
```
print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
```

In [76]:
```
tags.head()
```

Out[76]:

	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 [77]:
```
tags.index
```

Out[77]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [78]:
```
tags.columns
```

Out[78]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [83]:
```
tags.iloc[[0,11,500]]
```

Out[83]:

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

In [ ]:
```
# Descriptive Statistics
```

In [84]:
```
ratings['rating'].describe()
```

Out[84]:
```
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 [85]:
```
ratings.describe()
```

Out[85]:

	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 [87]:
```
ratings['rating'].mean()
```

Out[87]:
```
3.5255285642993797
```

In [89]:
```
ratings.mean()
```

Out[89]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [91]:
```
ratings['rating'].min()
```

Out[91]:
```
0.5
```

In [92]:
```
ratings['rating'].max()
```

Out[92]:
```
5.0
```

In [93]:
```
ratings['rating'].std()
```

Out[93]:
```
1.051988919275684
```

In [94]:
```
ratings['rating'].mode()
```

Out[94]:
```
0    4.0
Name: rating, dtype: float64
```

In [96]:
```
ratings.corr()
```

Out[96]:

	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 [99]:
```
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[99]:
```
False
```

In [100…
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[100]:
```
True
```

In [ ]:
```
# Data Cleaning: Handling Missing Data
```

In [101…
```
movies.shape
```

Out[101]:
```
(27278, 3)
```

In [102…
```
movies.isnull()
```

Out[102]:

	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 [104…
```
movies.isnull().any()
```

Out[104]:
```
movieId    False
title      False
genres     False
dtype: bool
```

In [105…
```
movies.isnull().any().any()
```

Out[105]:
```
False
```

In [106…
```
ratings.shape
```

Out[106]:
```
(20000263, 3)
```

In [109…
```
ratings.isnull().any()
```

Out[109]:
```
userId     False
movieId    False
rating     False
dtype: bool
```

In [110…
```
ratings.isnull().any().any()
```

Out[110]:
```
False
```

In [111…
```
tags.shape
```

Out[111]:
```
(465564, 3)
```

In [112…
```
tags.isnull().any()
```

Out[112]:
```
userId     False
movieId    False
tag         True
dtype: bool
```

In [113…
```
tags.isnull().any().any()
```

Out[113]:
```
True
```

In [114…
```
tags = tags.dropna()
```

In [115…
```
tags.isnull().any().any()
```

Out[115]:
```
False
```

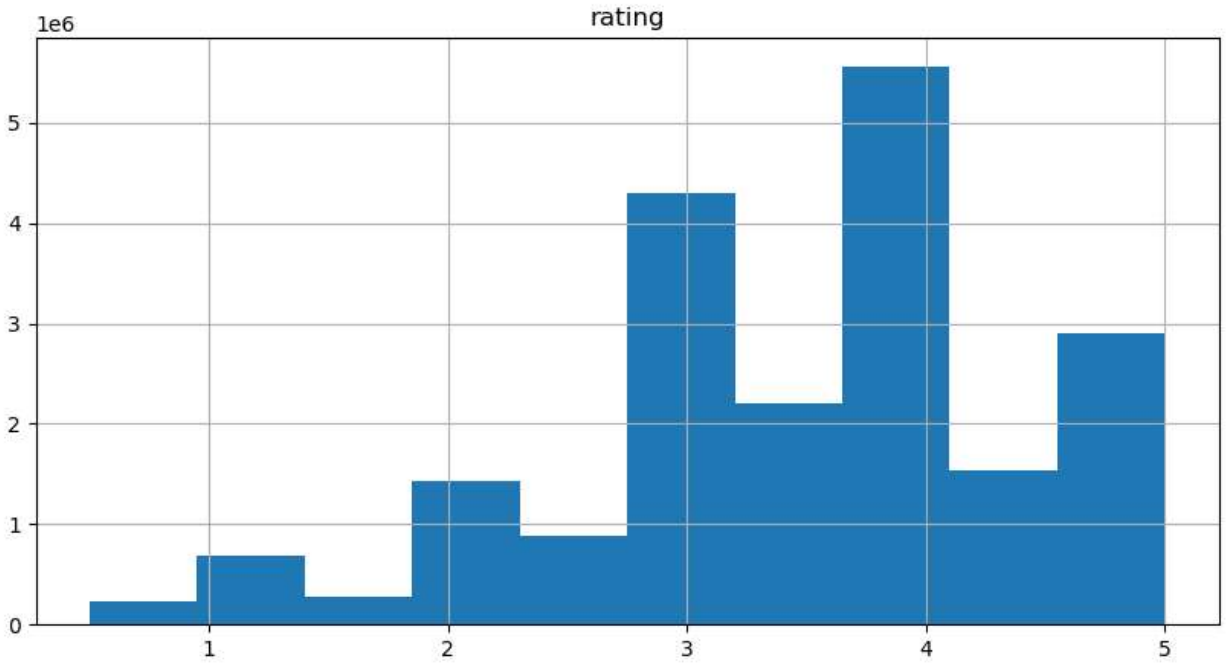
In [116…
```
tags.shape # reduced almost 16 lines after using tags.dropna()
```

Out[116]:
```
(465548, 3)
```

In [ ]:
```
# Data Visualization
```

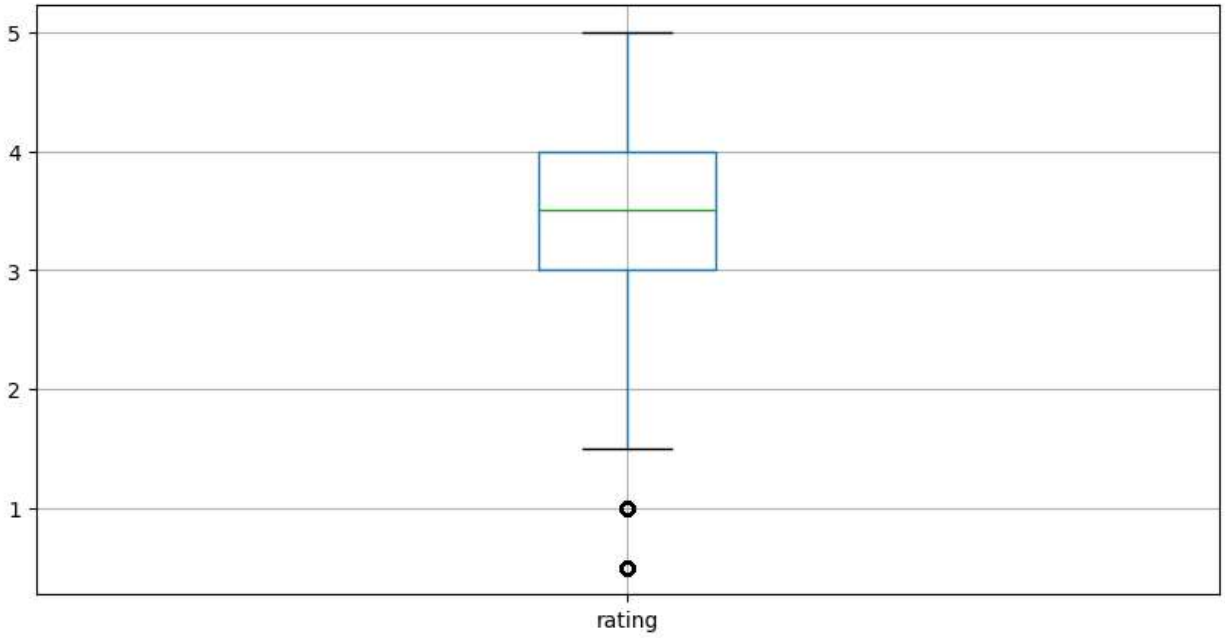
In [117…
```
import pandas as pd
import matplotlib as plt
%matplotlib inline

ratings.hist(column='rating',figsize = (10,5)) # Make a histogram of the DataFrame's c
```

Out[117]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



In [118…
```
ratings.boxplot(column='rating',figsize = (10,5))
```

Out[118]:
```
<Axes: >
```



In [ ]:
```
# Slicing Out Columns
```

In [119…
```
tags['tag'].head()
```

Out[119]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [120…
```
movies[['title','genres']].head()
```

Out[120]:

	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 [123…
```
ratings.shape
```

Out[123]:
```
(20000263, 3)
```

In [126…
```
ratings[-10:]
```

Out[126]:

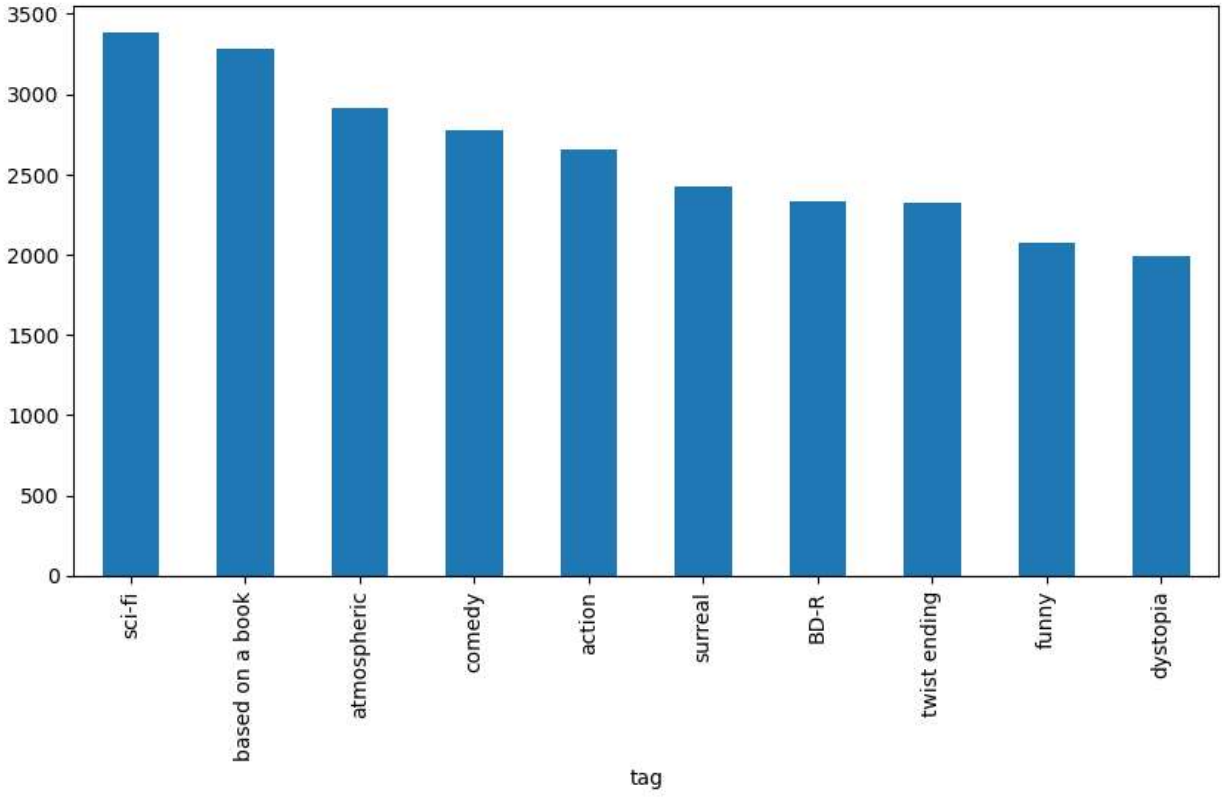
	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [125…
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[125]:
```
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 [129…
```
tag_counts[:10].plot(kind='bar',figsize = (10,5))
```

Out[129]:
```
<Axes: xlabel='tag'>
```



In [ ]:
```
# Filters for Selecting Rows
```

In [131…
```
is_highly_rated = ratings['rating'] >= 5.0
ratings[is_highly_rated][30:50]
```

Out[131]:

	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 [132…

```
is_action = movies['genres'].str.contains('Action')
movies[is_action][5:15]
```

Out[132]:

	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 [133…

```
movies[is_action].head(15)
```

Out[133]:

	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 [ ]:

```
# Group By and Aggregate
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: