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

In [17]:
```
# READING THE MOVIES DATASET
```

In [3]:
```
movies = pd.read_csv(r"C:\Users\AKSHAY\OneDrive\Desktop\Code\Projects\Project Co
```

In [5]:
```
movies
```

Out[5]:

	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 [7]:
```
# CHECKING THE TYPE OF THE DATASET
```

In [9]:
```
type(movies)
```

Out[9]:
```
pandas.core.frame.DataFrame
```

In [11]:
```
# PRINTING VALUES FROM THE DATASET
```

In [13]:
```
movies.head(20)
```

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
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 [15]:
```
# READING THE TAGS DATASET
```

In [19]:
```
tags = pd.read_csv(r"C:\Users\AKSHAY\OneDrive\Desktop\Code\Projects\Project Code
```

In [21]:
```
tags
```

Out[21]:

	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 [23]:
```
# CHECKING THE TYPE OF THE DATASET
```

In [25]:
```
type(tags)
```

Out[25]:
```
pandas.core.frame.DataFrame
```

In [27]:
```
# PRINTING THE VALUES IN THE DATASET
```

In [29]:
```
tags.head(20)
```

Out[29]:

	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
5	65	668	bollywood	2013-05-10 01:37:56
6	65	898	screwball comedy	2013-05-10 01:42:40
7	65	1248	noir thriller	2013-05-10 01:39:43
8	65	1391	mars	2013-05-10 01:40:55
9	65	1617	neo-noir	2013-05-10 01:43:37
10	65	1694	jesus	2013-05-10 01:38:45
11	65	1783	noir thriller	2013-05-10 01:39:43
12	65	2022	jesus	2013-05-10 01:38:45
13	65	2193	dragon	2013-05-10 02:01:54
14	65	2353	conspiracy theory	2013-05-10 02:01:06
15	65	2662	mars	2013-05-10 01:40:55
16	65	2726	noir thriller	2013-05-10 01:39:43
17	65	2840	jesus	2013-05-10 01:38:45
18	65	3052	jesus	2013-05-10 01:38:46
19	65	5135	bollywood	2013-05-10 01:37:56

In [31]:
```
# READING THE RATINGS DATASET
```

In [33]:
```
ratings = pd.read_csv(r"C:\Users\AKSHAY\OneDrive\Desktop\Code\Projects\Project C
```

In [34]:
```
ratings
```

Out[34]:

	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
...	...	...	...	...
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

20000263 rows × 4 columns

In [37]:
```
# CHECKING THE TYPE OF THE DATASET
```

In [39]:
```
type(ratings)
```

Out[39]:
```
pandas.core.frame.DataFrame
```

In [41]:
```
# PRINTING VALUES FROM THE DATASET
```

In [43]:
```
ratings.head(20)
```

Out[43]:

	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
5	1	112	3.5	2004-09-10 03:09:00
6	1	151	4.0	2004-09-10 03:08:54
7	1	223	4.0	2005-04-02 23:46:13
8	1	253	4.0	2005-04-02 23:35:40
9	1	260	4.0	2005-04-02 23:33:46
10	1	293	4.0	2005-04-02 23:31:43
11	1	296	4.0	2005-04-02 23:32:47
12	1	318	4.0	2005-04-02 23:33:18
13	1	337	3.5	2004-09-10 03:08:29
14	1	367	3.5	2005-04-02 23:53:00
15	1	541	4.0	2005-04-02 23:30:03
16	1	589	3.5	2005-04-02 23:45:57
17	1	593	3.5	2005-04-02 23:31:01
18	1	653	3.0	2004-09-10 03:08:11
19	1	919	3.5	2004-09-10 03:07:01

In [45]:
```
# AS WE WONT BE USING THE 'TIMESTAMP' COLUMN, WE WILL DELETE IT FROM THE RATINGS
```

In [47]:
```
del ratings['timestamp']
del tags['timestamp']
```

In [49]:
```
ratings
```

Out[49]:

	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 [51]:
```
tags
```

Out[51]:

	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

DATA STRUCTURES

Series

In [57]:
```
# ACCESSING SPECIFIC VALUES USING THEIR INTEGER NUMBER
```

In [65]:
```
row_0 = tags.iloc[0] # is used to access values
print(type(row_0))
print(row_0)
print(row_0.index) # returns the attributes
print(row_0['userId']) # prints the value
```

```
<class 'pandas.core.series.Series'>
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
Index(['userId', 'movieId', 'tag'], dtype='object')
18
```

In [67]:
```
# MEMBERSHIP
```

In [69]:
```
'rating' in row_0
```

Out[69]: False

In [71]:
```
row_0.name # returns the name of the series or s.no
```

Out[71]: 0

In [79]:
```
# WE ARE CHANGING THE ROW NAME FROM 0 TO FirstRow
```

In [81]:
```
row_0 = row_0.rename('FirstRow')
print(row_0.name)
print(row_0)
```

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

Data Frames

In [84]:
```
tags.head() # diplay first 5 values by default
```

Out[84]:

	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 [88]:
```
tags.index # returns the total index
```

Out[88]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [90]:
```
# WE ARE PRITING VALUES USING THEIR INTEGER INDEX
```

In [92]:
```
tags.iloc[[0,11,500]]
```

Out[92]:

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

Descriptive Statistics

In [95]:
```
# WE ARE PERFORIMG SOME MATH FUNCTIONS
```

In [105…
```
ratings['rating'].describe() # for a particular column in the dataset
```

Out[105…
```
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 [107…
```
ratings.describe() # for the whole dataset
```

Out[107…

	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 [111…
```
ratings['rating'].mean() # average of all values in the column
```

Out[111…
```
3.5255285642993797
```

In [113…
```
ratings.mean() # average of all column separately
```

Out[113…
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [119…
```
ratings['rating'].min() # finds the minimum value in the column
```

Out[119…
```
0.5
```

In [121…
```
ratings['rating'].max() # finds the maximum value in the column
```

Out[121…
```
5.0
```

In [123…
```
ratings['rating'].std() # finds the standard deviation of the column
```

Out[123…
```
1.051988919275684
```

In [125…
```
ratings['rating'].mode() # finds the mode of the column
```

Out[125…
```
0    4.0
Name: rating, dtype: float64
```

In [127…
```
ratings.corr() # finds the correlation
```

Out[127…

	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 [129…
```
# WE ARE USING CASE STUDIES
```

In [131…
```
# WE ARE THE VALUES THAT HAVE RATING > 10 IN THE RATINGS DATASET AND RATING COLU
```

In [141…
```
filter1 = ratings['rating'] > 10
print(filter1)
filter1.any() # returns if any value satisfies the condition
```

```
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[141…
```
False
```

In [143…
```
# WE ARE THE VALUES THAT HAVE RATING > 0 IN THE RATINGS DATASET AND RATING COLUM
```

In [153…
```
filter2 = ratings['rating'] > 0
filter2
```

Out[153…
```
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
```

In [155…
```
ratings[filter2] # printing the records that satisifies the condition
```

Out[155…

	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

Data Cleaning: Handling Missing Data

In [171…
```
# CHECKING THE DIMESIONS OF THE MOVIES DATASET
```

In [165…
```
movies.shape # returns the rows and columns
```

Out[165…
```
(27278, 3)
```

In [205…
```
movies.isnull().any() # this gives True if there is any missing value
```

Out[205…
```
movieId    False
title      False
genres     False
dtype: bool
```

In [173…
```
# CHECKING THE DIMESIONS OF THE RATINGS DATASET
```

In [177…
```
ratings.shape # returns the rows and columns
```

Out[177…
```
(20000263, 3)
```

In [203…
```
ratings.isnull().any() # this gives True if there is any missing value
```

Out[203…
```
userId     False
movieId    False
rating     False
dtype: bool
```

In [183…
```
# CHECKING THE DIMESIONS OF THE TAGS DATASET
```

In [187…
```
tags.shape # returns the rows and columns
```

Out[187…
```
(465564, 3)
```

In [201…
```
tags.isnull().any() # this gives True if there is any missing value
```

Out[201…
```
userId     False
movieId    False
tag         True
dtype: bool
```

In [207…
```
# WE ARE GOING TO REMOVE THE NULL VALUES FROM THE TAG COLUMN IN THE TAGS DATASET
```

In [209…
```
tags = tags.dropna() # removes missing values
```

In [213…
```
tags.isnull().any() # this gives True if there is any missing value
```

Out[213…
```
userId     False
movieId    False
tag        False
dtype: bool
```

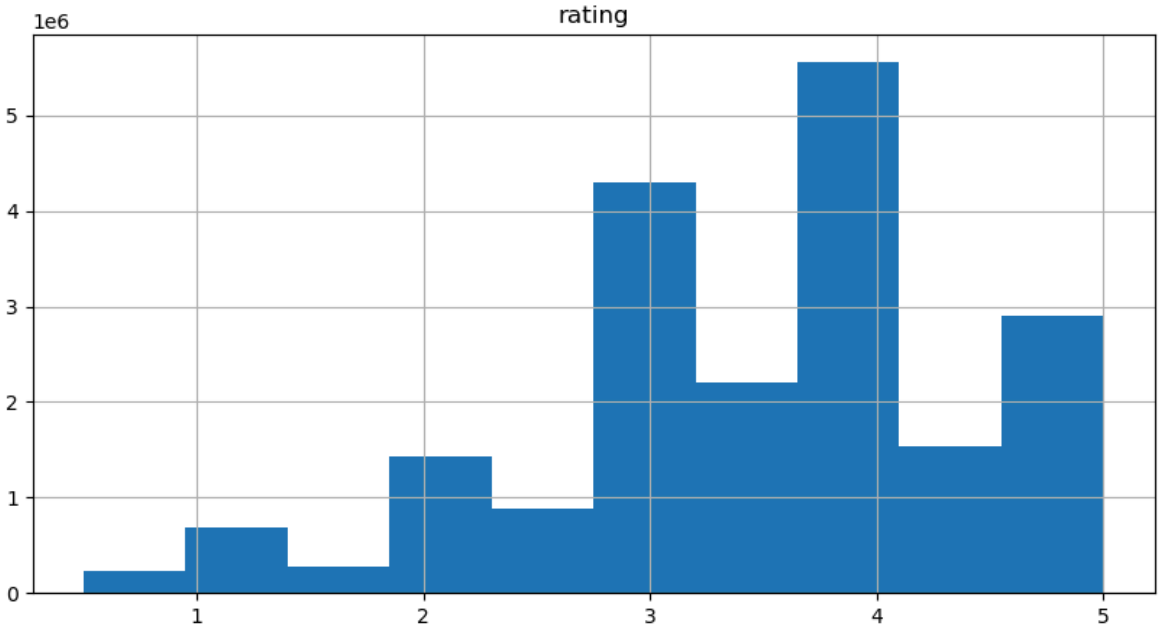
Data Visualization

In [222…
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [236…
```
# HISTOGRAM OF THE RATING COLUMN INSIDE THE RATINGS DATASET
```

In [234…
```
print(ratings.hist(column = 'rating', figsize=(10,5)))
plt.show()
```

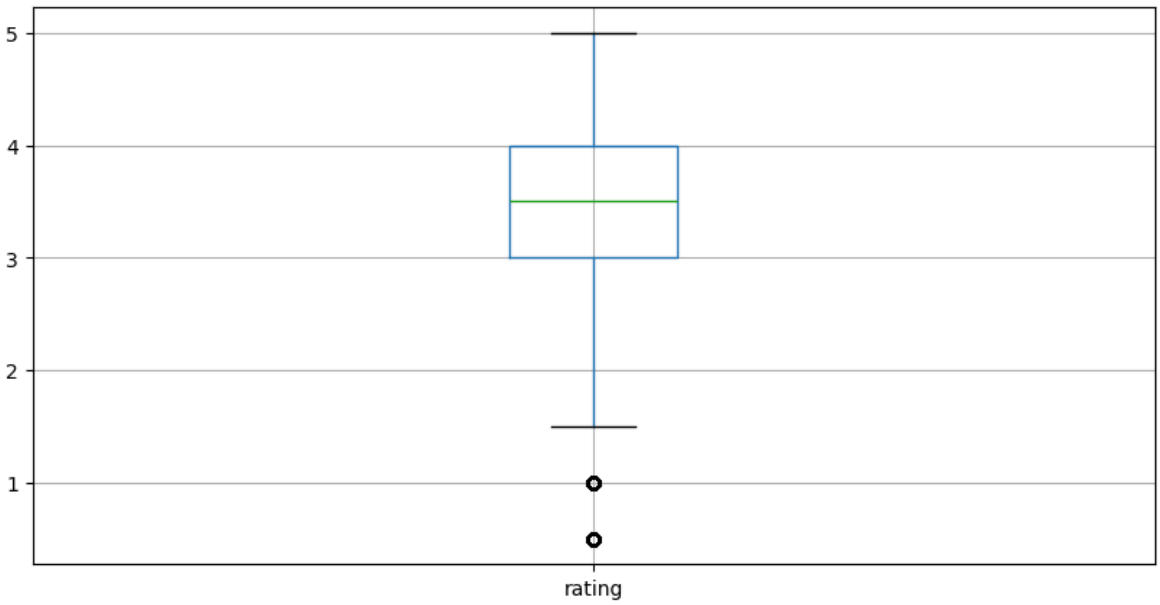
```
[[<Axes: title={'center': 'rating'}>]]
```



In [238…
```
# BOXPLOT OF THE RATING COLUMN INSIDE THE RATINGS DATASET
```

In [240…
```
print(ratings.boxplot(column = 'rating', figsize=(10,5)))
plt.show()
```

```
Axes(0.125,0.11;0.775x0.77)
```



Slicing Out Columns

In [247…
```
# PRINTING VALUES FROM THE TAG COLUMN IN THE TAGS DATASET
```

In [245…
```
tags['tag'].head()
```

Out[245…
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [249…
```
# PRINTING VALUES FROM THE TITILE, GENRE COLUMNS IN THE MOVIES DATASET
```

In [251…
```
movies[['title','genres']].head()
```

Out[251…

	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 [253…
```
# PRINTING THE BOTTOM 10 ROWS IN THE RATINGS DATASET
```

In [257…
```
ratings[-10:]
```

Out[257…

	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 [261…
```
tag_counts = tags['tag'].value_counts() # This function counts the occurrences o
tag_counts[-10:]
```

Out[261…
```
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 [295…
```
tag_counts # this is storing the genre occurences times
```

Out[295…
```
tag
sci-fi                          3384
based on a book                 3281
atmospheric                     2917
comedy                          2779
action                          2657
                                ...
Paul Adelstein                     1
the wig                            1
killer fish                        1
genetically modified monsters      1
topless scene                      1
Name: count, Length: 38643, dtype: int64
```

In [293…
```
# WE ARE PRINTING VALUES FROM 0 - 9 FROM THE TAGS_COUNTS VARIABLE THAT SOTES THE
```

In [ ]:
```
# WE ARE PLOTTING A BAR TYPE REPRESENTATING OF THOSE VALUES
```

In [299…
```
tag_counts[:10]
```

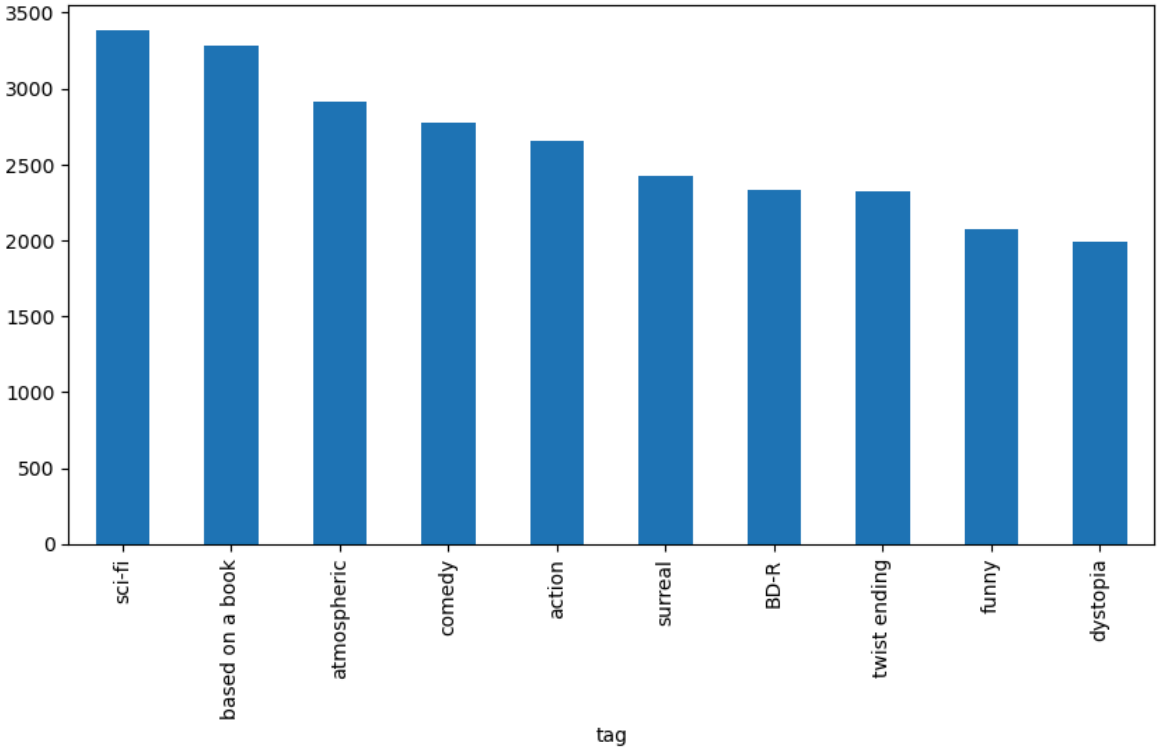
Out[299…
```
tag
sci-fi             3384
based on a book    3281
atmospheric        2917
comedy             2779
action             2657
surreal            2427
BD-R               2334
twist ending       2323
funny              2072
dystopia           1991
Name: count, dtype: int64
```

In [301…
```
# IN THE ABOVE OUTPUT WE CAN SEE THE TAG NAME, AND HOW MAN TIMES IT OCCURED
```

In [303…
```
# NOW WE ARE PLOTTING THAT COUNT IN THE BELOW BAR GRAPH
```

In [267…
```
print(tag_counts[:10].plot(kind = 'bar', figsize = (10,5)))
plt.show()
```

```
Axes(0.125,0.11;0.775x0.77)
```


3500
3000
2500
2000
1500
1000
500
0
sci-fi
based on a book
atmospheric
comedy
action
surreal
BD-R
twist ending
funny
dystopia
tag

In [ ]: