Import the Dataset

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

Read the dataset

In [3]:
```
movies = pd.read_csv(r'C:\Users\Swapnil Rajbhar\Desktop\archive\movie.csv')
rating = pd.read_csv(r'C:\Users\Swapnil Rajbhar\Desktop\archive\rating.csv')
tag = pd.read_csv(r'C:\Users\Swapnil Rajbhar\Desktop\archive\tag.csv')
```

In [4]:
```
print(movies.shape)
print(rating.shape)
print(tag.shape)
```

```
(27278, 3)
(20000263, 4)
(465564, 4)
```

In [5]:
```
movies.head(1)
```

Out[5]:

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

In [139…
```
parse_dates = ['timestamp']
rating.head(1)
```

Out[139…

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47

In [7]:
```
rating.tail(1)
```

Out[7]:

	userId	movieId	rating	timestamp
20000262	138493	71619	2.5	2009-10-17 20:25:36

In [143…
```
del rating['timestamp']
del tag['timestamp']
```

In [8]:
```
print(type(movies))
movies.head(20)
```

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

Out[8]:

	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 [9]:
```
tag.head()
```

Out[9]:

	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 [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]:
```
row_0 = tag.iloc[0]
type(row_0)
```

Out[11]:
```
pandas.core.series.Series
```

In [13]:
```
print(row_0)
```

```
userId                        18
movieId                     4141
tag                 Mark Waters
timestamp    2009-04-24 18:19:40
Name: 0, dtype: object
```

In [14]:
```
row_0.index
```

Out[14]:
```
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [15]:
```
row_0['userId']
```

Out[15]:
```
18
```

In [16]:
```
'rating' in row_0
```

Out[16]:
```
False
```

In [17]:
```
row_0.name
```

Out[17]:
```
0
```

In [18]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[18]:
```
'firstRow'
```

Data Frames

In [20]:
```
tag.head()
```

Out[20]:

	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 [21]:
```
tag.index
```

Out[21]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [22]:
```
tag.columns
```

Out[22]:
```
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [23]:
```
tag.iloc[ [0,11,500] ]
```

Out[23]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
11	65	1783	noir thriller	2013-05-10 01:39:43
500	342	55908	entirely dialogue	2012-01-31 18:41:16

📈 📉 Descriptive Statistics

Let's look how the ratings are distributed!

In [25]:
```
rating['rating'].describe()
```

Out[25]:
```
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 [26]:
```
rating.describe()
```

Out[26]:

	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 [27]:
```
rating['rating'].mean()
```

Out[27]:
```
3.5255285642993797
```

In [31]:
```
rating['rating'].min()
```

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

In [33]:
```
rating['rating'].max()
```

Out[33]:
```
5.0
```

In [35]:
```
rating['rating'].std
```

Out[35]:
```
<bound method Series.std of 0           3.5
1           3.5
2           3.5
3           3.5
4           3.5
           ...
20000258    4.5
20000259    4.5
20000260    3.0
20000261    5.0
20000262    2.5
Name: rating, Length: 20000263, dtype: float64>
```

In [39]:
```
rating['rating'].mode()
```

Out[39]:
```
0    4.0
Name: rating, dtype: float64
```

In [145…
```
rating.corr()
```

Out[145…

	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 [45]:
```
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[45]: False

In [51]:
```
filter2 = rating['rating'] > 0
filter2.all()
```

Out[51]: True

🔧 Data Cleaning: Handling Missing Data

In [57]:
```
movies.shape
```

Out[57]: (27278, 3)

In [61]:
```
movies.isnull().any().any()
```

Out[61]: False

In [63]:
```
rating.shape
```

Out[63]: (20000263, 4)

In [65]:
```
rating.isnull().any().any()
```

Out[65]: False

In [69]:
```
tag.shape
```

Out[69]: (465564, 4)

In [71]:
```
tag.isnull().any().any()
```

Out[71]: True

In [73]:
```
tag = tag.dropna()
```

In [75]:
```
tag.isnull().any().any()
```

Out[75]: False

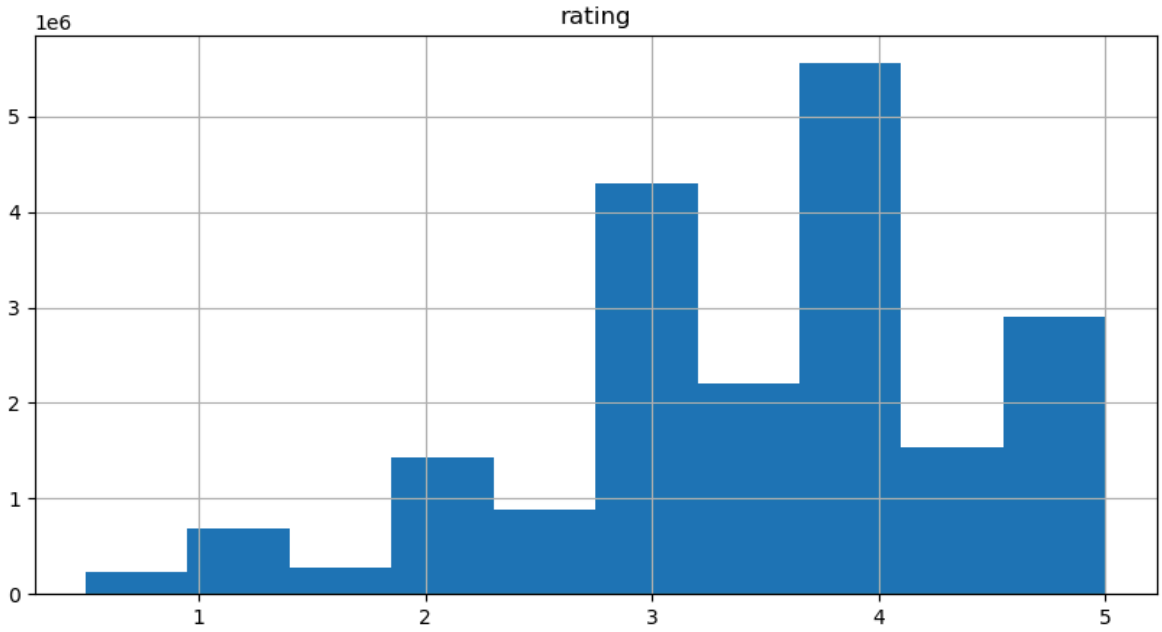
In [77]:
```
tag.shape
```

Out[77]:
```
(465548, 4)
```

📊 Data Visualization

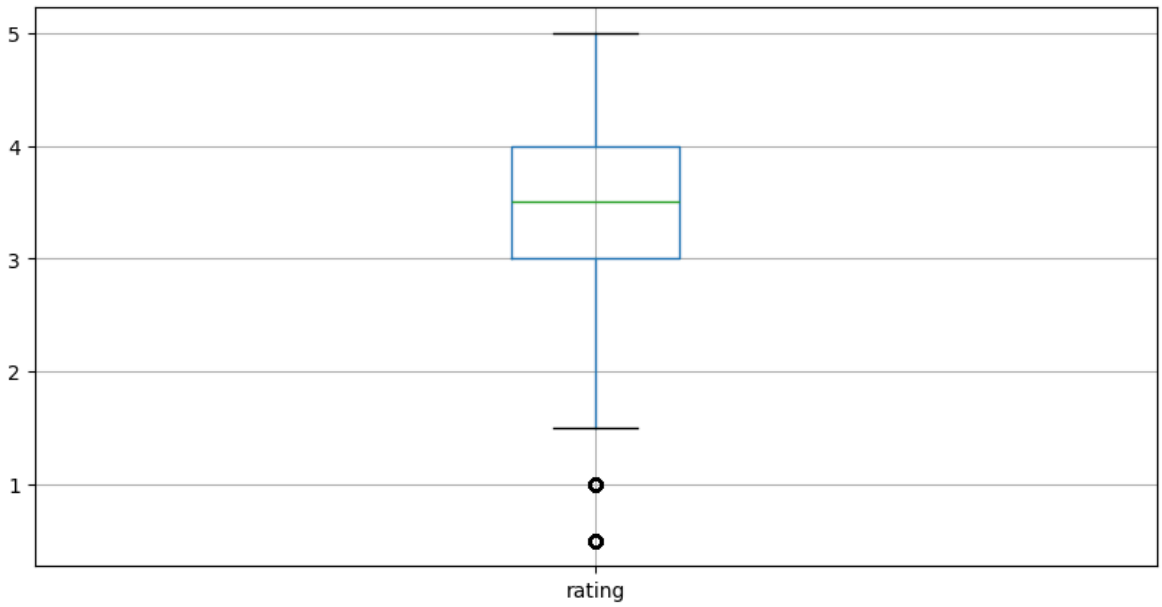
In [84]:
```
%matplotlib inline
rating.hist(column = 'rating', figsize =(10,5))
```

Out[84]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



In [86]:
```
rating.boxplot(column='rating', figsize=(10,5))
```

Out[86]:
```
<Axes: >
```



📤 Slicing Out Columns

In [91]:
```
tag['tag'].head()
```

Out[91]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [93]:
```
movies[['title','genres']].head()
```

Out[93]:

	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 [97]:
```
rating[-10:]
```

Out[97]:

	userId	movieId	rating	timestamp
20000253	138493	60816	4.5	2009-12-03 18:32:43
20000254	138493	61160	4.0	2009-11-16 16:55:37
20000255	138493	65682	4.5	2009-10-17 21:52:53
20000256	138493	66762	4.5	2009-10-17 18:50:08
20000257	138493	68319	4.5	2009-12-07 18:15:20
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

In [101…
```
tag_counts = tag['tag'].value_counts()
tag_counts[-10:]
```

Out[101…

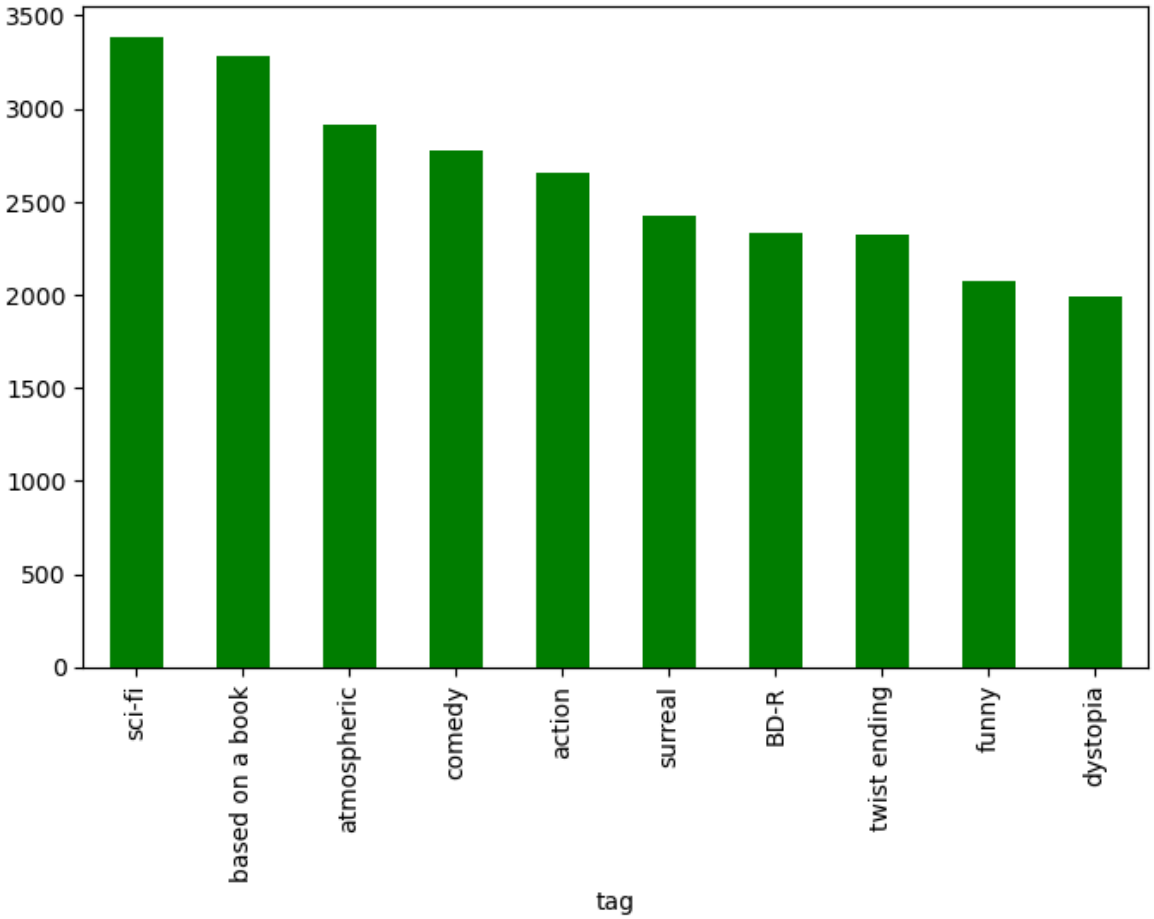
```
tag
missing child                    1
Ron Moore                        1
Citizen Kane                     1
mullet                           1
biker gang                       1
Paul Adelstein                   1
the wig                          1
killer fish                      1
genetically modified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

In [131…

```
tag_counts[:10].plot(kind = 'bar',figsize = (8,5),color = 'green')
```

Out[131…

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



In [ ]: