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

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

In [7]:
```
ratings.shape
```

Out[7]:
```
(20000263, 4)
```

In [9]:
```
ratings.head
```

Out[9]:
```
<bound method NDFrame.head of           userId  movieId  rating            time
stamp
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 x 4 columns]>
```

In [11]:
```
ratings.head(1)
```

Out[11]:

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

In [23]:
```
tags =  pd.read_csv(r'C:\Users\banot\Downloads\archive\tag.csv')
```

In [29]:
```
tags.shape
```

Out[29]:
```
(465564, 4)
```

In [31]:
```
tags.head(1)
```

Out[31]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40

In [37]:
```
movies =  pd.read_csv(r'C:\Users\banot\Downloads\archive\movie.csv')
```

In [39]:
```
movies.shape
```

Out[39]:
```
(27278, 3)
```

In [41]:
```
movies.head(1)
```

Out[41]:

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

In [43]:
```
del ratings ['timestamp']
del tags['timestamp']
```

In [45]:
```
ratings.columns
```

Out[45]:
```
Index(['userId', 'movieId', 'rating'], dtype='object')
```

In [47]:
```
tags.columns
```

Out[47]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [49]:
```
movies.columns
```

Out[49]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

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

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

Data structures

series

In [57]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[57]:
```
pandas.core.series.Series
```

In [63]:
```
print(row_0)
```

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

In [73]:
```
row_3 = tags.iloc[3]
type(row_3)
```

Out[73]:
```
pandas.core.series.Series
```

In [75]:
```
print(row_3)
```

```
userId                65
movieId              521
tag        noir thriller
Name: 3, dtype: object
```

In [77]:
```
row_0.index
```

Out[77]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [79]:
```
row_0['userId']
```

Out[79]:
```
65
```

In [81]:
```
row_0
```

Out[81]:
```
userId                65
movieId              521
tag        noir thriller
Name: 3, dtype: object
```

In [83]:
```
print(row_0)
```

```
userId                65
movieId              521
tag        noir thriller
Name: 3, dtype: object
```

In [85]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[85]:
```
pandas.core.series.Series
```

In [87]:
```
row_0
```

Out[87]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [89]:
```
row_0['userId']
```

Out[89]:
```
18
```

In [91]:
```
row_0.name
```

Out[91]:
```
0
```

In [93]:
```
row_0 = row_0.rename('firstrow')
row_0.name
```

Out[93]:
```
'firstrow'
```

DataFrames

In [96]:
```
tags.head()
```

Out[96]:

	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 [98]:
```
tags.iloc[[0,11,504]]
```

Out[98]:

	userId	movieId	tag
0	18	4141	Mark Waters
11	65	1783	noir thriller
504	342	55908	sci-fi

In [100…
```
tags.iloc[[7,67,35,245]]
```

Out[100…

	userId	movieId	tag
7	65	1248	noir thriller
67	121	5283	College Humor
35	96	106696	animation
245	129	81834	camping trip

Descriptive statistics

how the ratings are distributed!

In [104…
```
ratings['rating'].describe()
```

Out[104…
```
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 [106…
```
ratings.describe()
```

Out[106…

	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 [108…
```
ratings['rating'].mean()
```

Out[108…
```
3.5255285642993797
```

In [110…
```
ratings.mean()
```

Out[110…
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [112…
```
ratings['rating'].min
```

Out[112…
```
<bound method Series.min 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 [114…
```
ratings['rating'].max()
```

Out[114…
```
5.0
```

In [116…
```
ratings['rating'].std()
```

Out[116…
```
1.051988919275684
```

In [118…
```
ratings['rating'].mode()
```

Out[118…
```
0    4.0
Name: rating, dtype: float64
```

In [120…
```
ratings.corr()
```

Out[120…

	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 [122…
```
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[122…
```
False
```

In [124…
```
filter2 = ratings['rating'] > 10
filter2.all()
```

Out[124…
```
False
```

In [126…
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[126…
```
True
```

Data Cleaning: handling missing data

In [129…
```
movies
```

Out[129…

	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 [131…
```
movies.shape
```

Out[131…
```
(27278, 3)
```

In [133…
```
movies.isnull().any()
```

Out[133…
```
movieId    False
title      False
genres     False
dtype: bool
```

In [135…
```
movies.isnull().any().any()
```

Out[135…
```
False
```

No Null Values!

In [138…
```
ratings
```

Out[138…

	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 [140…
```
ratings.shape
```

Out[140…
```
(20000263, 3)
```

In [142…
```
ratings.isnull().any().any()
```

Out[142…
```
False
```

No Null Values!

In [145…
```
tags
```

Out[145...

	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 [147...
```
tags.shape
```

Out[147... (465564, 3)

In [149...
```
tags.isnull().any().any()
```

Out[149... True

Have some null values in tags

In [152...
```
tags = tags.dropna()
```

In [154...
```
tags.isnull().any().any()
```

Out[154... False

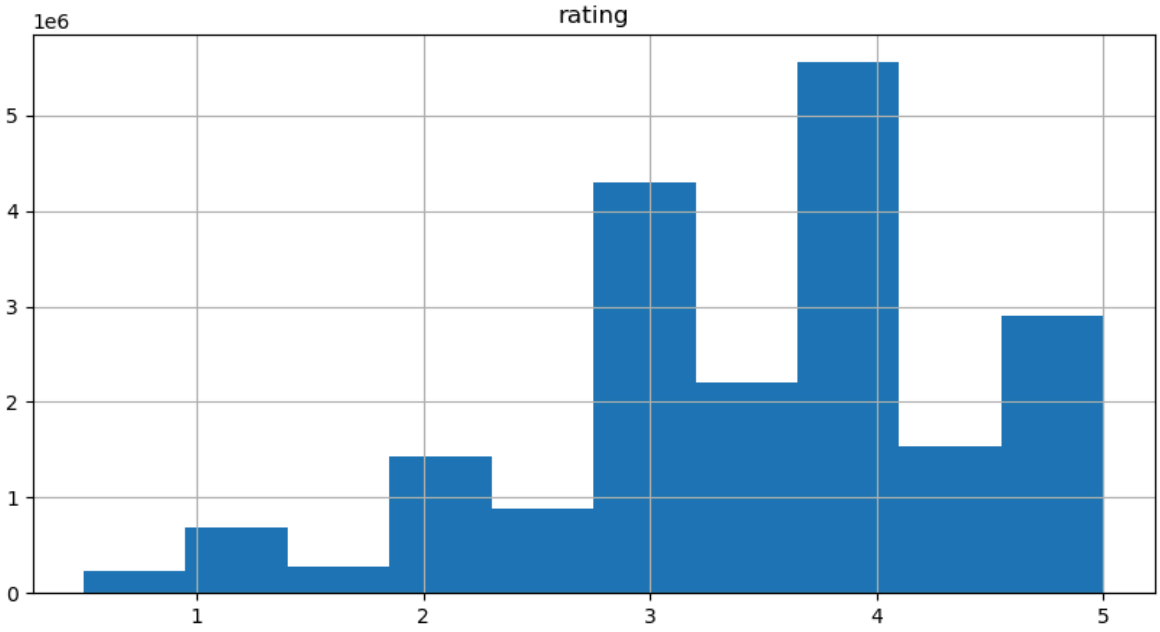
In [156...
```
tags.shape
```

Out[156... (465548, 3)

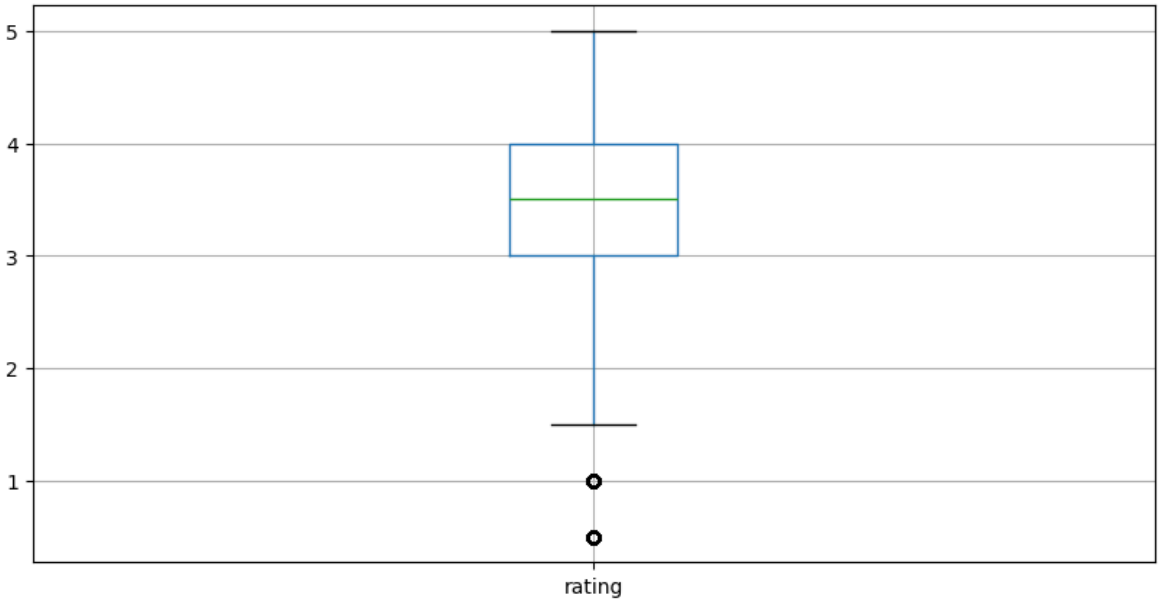
No Null values! Note the number of lines have reduced

Data Visulizatiuon

In [166...
```
%matplotlib inline
import matplotlib.pyplot as plt
ratings.hist(column = 'rating',figsize = (10,5))
plt.show()
```



In [171…
```
ratings.boxplot(column = 'rating',figsize = (10,5))
plt.show()
```



Slicing out columns

In [174…
```
tags
```

Out[174...

	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 [176...
```
tags['tag'].head()
```

Out[176...
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [178...
```
movies[['title','genres']].head()
```

Out[178...

	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 [180...
```
ratings[-10:]
```

Out[180…

	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 [182…

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

Out[182…

	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
...	...	...	...
20000248	138493	58879	4.5
20000249	138493	59315	4.0
20000250	138493	59725	3.0
20000251	138493	59784	5.0
20000252	138493	60069	4.0

20000253 rows × 3 columns

In [186…

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

Out[186…

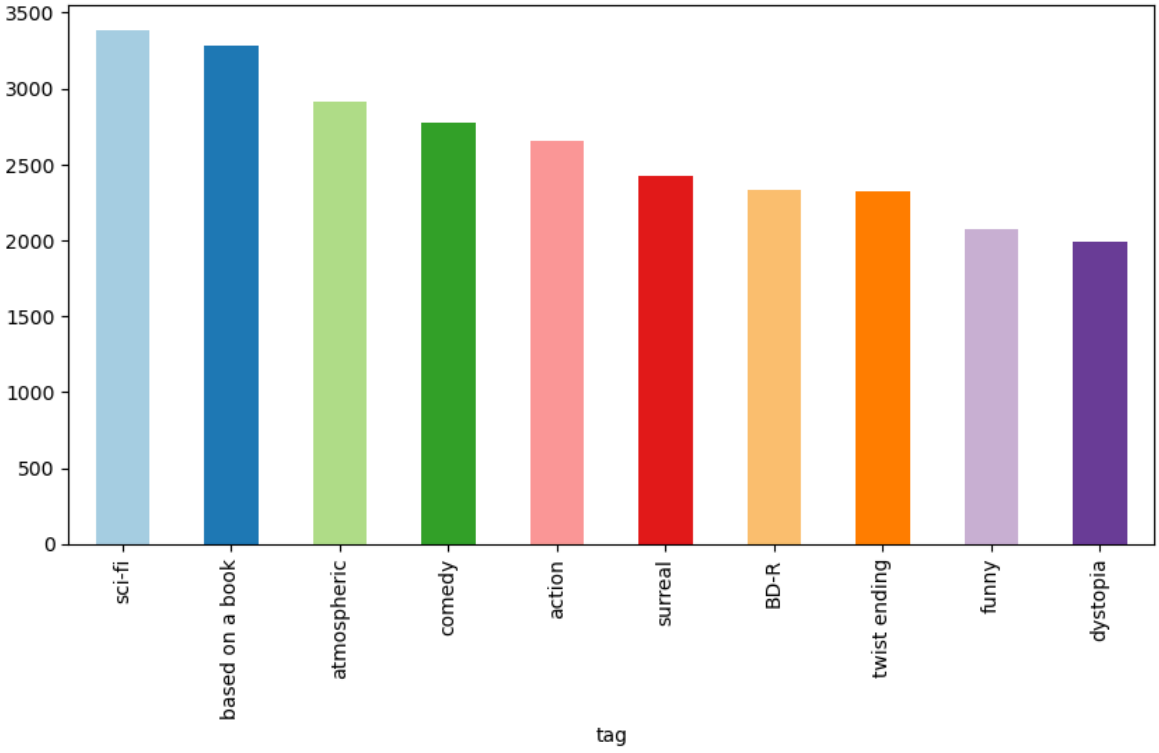
```
tag
missing child                  1
Ron Moore                      1
Citizen Kane                   1
mullet                         1
biker gang                     1
Paul Adelstein                 1
the wig                        1
killer fish                    1
genetically modified monsters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

In [190…

```
colors = plt.cm.Paired.colors
tag_counts[:10].plot(kind = 'bar',figsize = (10,5),color = colors)
```

Out[190…

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



In [ ]: