pandas with Data Science.AI

Location: https://www.kaggle.com/grouplens/moviel 20m-dataset

```
In [9]: import pandas as pd
```

```
In [10]: import matplotlib.pyplot as plt
```

```
In [68]: %matplotlib inline
```

```
In [11]: movies= pd.read_csv(r'C:\Users\ARUN KUMAR SAHU\Downloads\archive\movie.csv')
         print(type(movies))
         movies.head(20)
```

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

Out[11]:

	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 [14]:
```
tags = pd.read_csv(r'C:\Users\ARUN KUMAR SAHU\Downloads\archive\tag.csv')
tags.head()
```

Out[14]:

	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 [27]:
```
tags=pd.read_csv(r'C:\Users\ARUN KUMAR SAHU\Downloads\archive\tag.csv')
tags.tail()
```

Out[27]:

	userId	movieId	tag	timestamp
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

In [29]:
```
del tags['timestamp']
```

Data structure

.SERIES

In [22]:
```
row_0 =tags.iloc[0]
type(row_0)
```

Out[22]:
```
pandas.core.series.Series
```

In [23]:
```
print(row_0)
```

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

In [30]:
```
row_0.index
```

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

In [32]:
```
row_0['userId']
```

Out[32]:
```
np.int64(18)
```

In [33]:
```
'rating' in row_0
```

Out[33]:
```
False
```

In [34]:
```
row_0.name
```

Out[34]:
```
0
```

In [35]:
```
row_0=row_0.rename('firstRow')
row_0.name
```

Out[35]:
```
'firstRow'
```

Data Frames

```
In [36]: tags.head()
```

Out[36]:

	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 [37]: tags.index
```

```
Out[37]: RangeIndex(start=0, stop=465564, step=1)
```

```
In [38]: tags.columns
```

```
Out[38]: Index(['userId', 'movieId', 'tag'], dtype='object')
```

```
In [39]: tags.iloc[[0,11,500]]
```

Out[39]:

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

Descritive Statistics

```
In [40]: ratings['rating'].describe()
```

```
Out[40]: 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 [41]: ratings.describe()
```

Out[41]:

	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 [42]:
```
ratings['rating'].mean()
```
Out[42]:
```
np.float64(3.5255285642993797)
```

In [43]:
```
ratings.mean()
```
Out[43]:
```
userId      69045.872583
movieId      9041.567330
rating          3.525529
dtype: float64
```

In [44]:
```
ratings['rating'].min()
```
Out[44]:
```
np.float64(0.5)
```

In [45]:
```
ratings['rating'].min()
```
Out[45]:
```
np.float64(0.5)
```

In [46]:
```
ratings['rating'].std()
```
Out[46]:
```
np.float64(1.0519889192942418)
```

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

In [47]:
```
ratings.corr()
```
Out[47]:

	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 [48]:
```
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[48]:  np.False_
```

```
In [49]:  filter2=ratings['rating']>0
          filter2.all()
```

```
Out[49]:  np.True_
```

Data Cleaning:Handling Missing Data

```
In [50]:  movies.shape
```

```
Out[50]:  (27278, 3)
```

```
In [51]:  movies.isnull().any().any()   #no NULL value
```

```
Out[51]:  np.False_
```

```
In [52]:  ratings.shape
```

```
Out[52]:  (20000263, 3)
```

```
In [53]:  ratings.isnull().any().any()
```

```
Out[53]:  np.False_
```

```
In [54]:  tags.shape
```

```
Out[54]:  (465564, 3)
```

```
In [55]:  tags.isnull().any().any()
```

```
Out[55]:  np.True_
```

.we have some tags which are NULL

```
In [56]:  tags=tags.dropna()
```

```
In [57]:  tags.isnull().any().any()
```

```
Out[57]:  np.False_
```

```
In [58]:  tags.shape
```

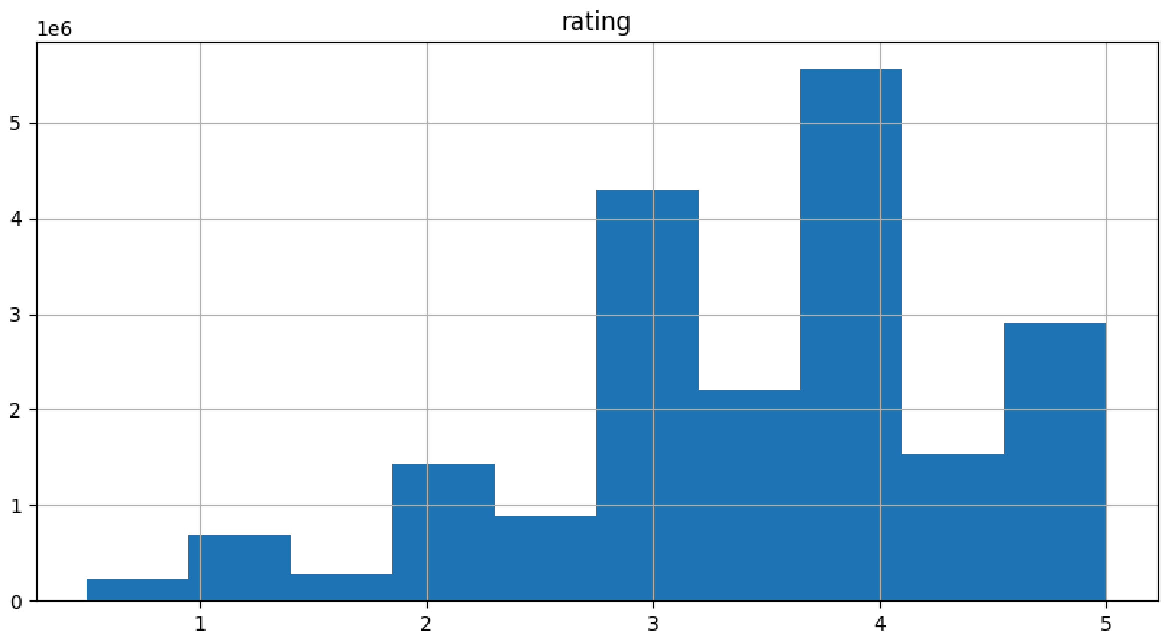
Out[58]:
```
(465548, 3)
```

thats nice!No NULL values! no of lines have reduced bcz of dropna() drop he null

Data Visualization

In [59]:
```
%matplotlib inline
ratings.hist(column='rating',figsize=(10,5))
```

Out[59]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



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

Slicing Out Columns

In [60]:
```
tags['tag'].head()
```

Out[60]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [61]:
```
movies[['title','genres']].head()
```

Out[61]:

	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 [62]:
```
ratings[-10:]
```

Out[62]:

	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

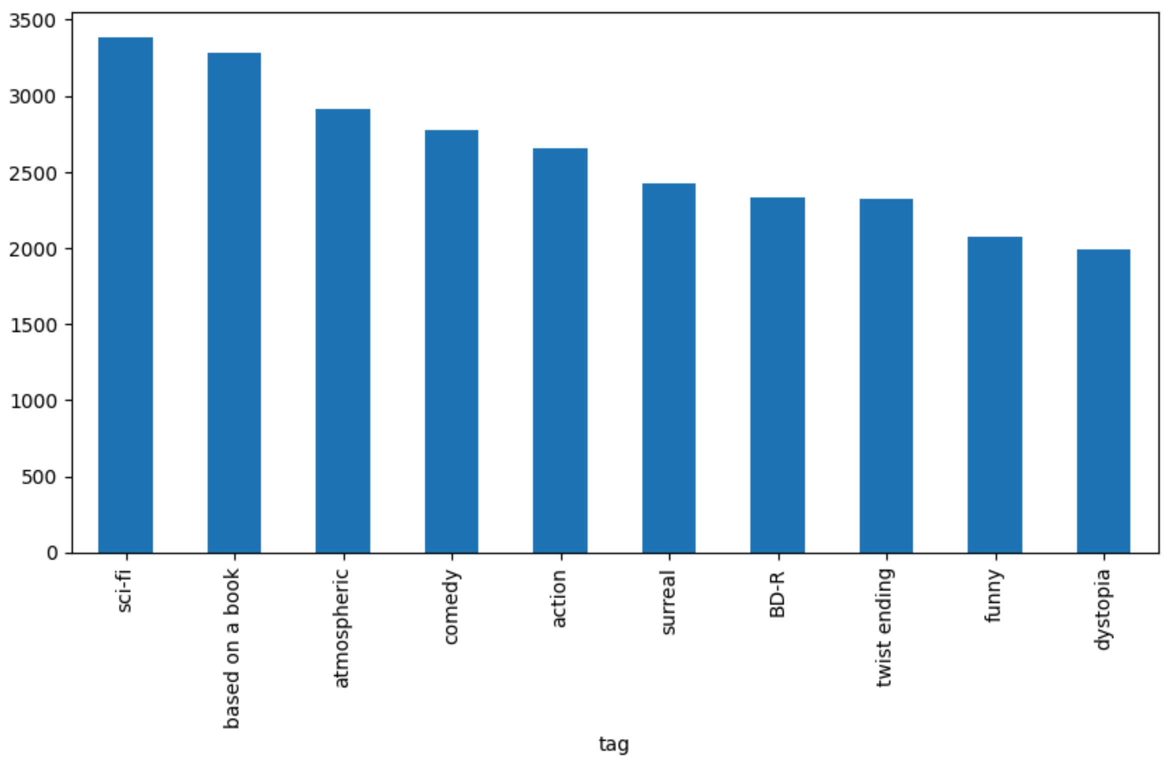
In [63]:
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
tag_counts[-10:]
```

Out[63]:
```
tag
Hell naw                      1
This is my happy face         1
I heel toe on Uday's house    1
Why?                          1
Bobo                          1
Diamond Dallas Page           1
I'm Devon Butler!             1
No arguement                  1
Really Bad                    1
Botox                         1
Name: count, dtype: int64
```

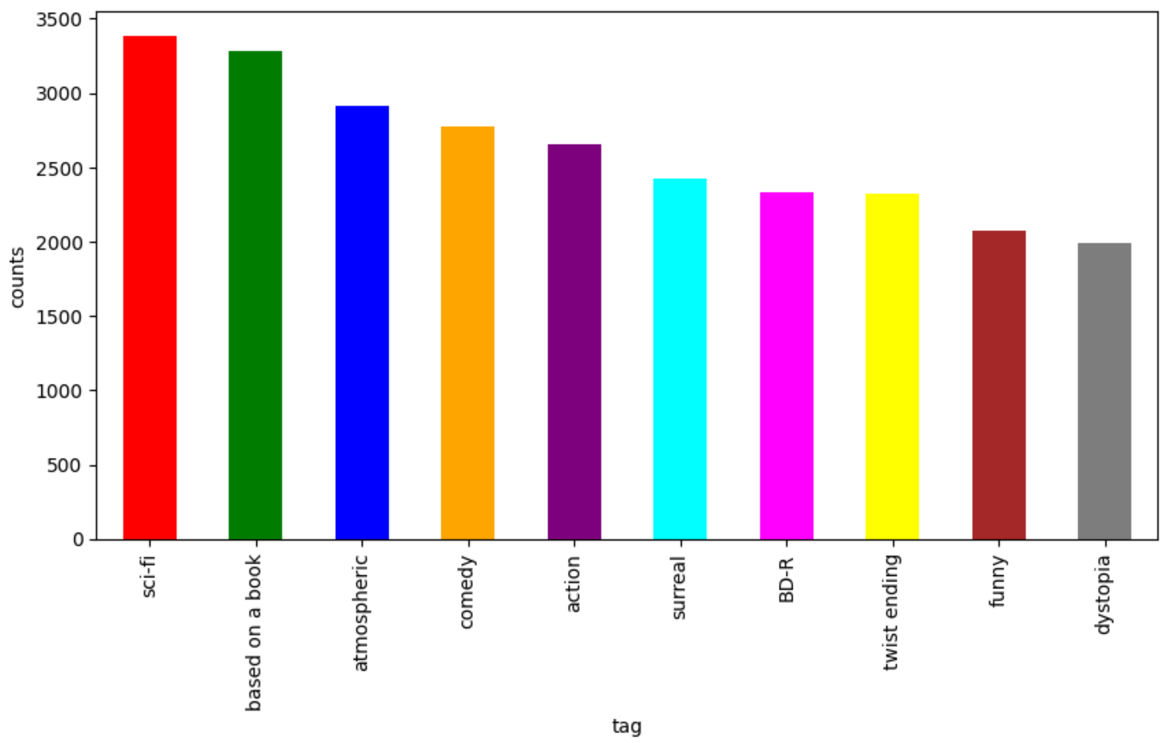
In [64]:
```
import matplotlib.pyplot as plt
```

In [65]:
```
tag_counts[:10].plot(kind='bar',figsize=(10,5))
```

Out[65]:
```
<Axes: xlabel='tag'>
```



```
In [69]: tag_counts[:10].plot(kind='bar',figsize=(10,5),
                              xlabel="tag",
                              ylabel="counts",
                              color= ['red', 'green', 'blue', 'orange', 'purple', 'cyan',
         plt.show()
```



```
In [ ]:
```