kaggle task for data science with pandas ai

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

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

Read The Data Set

In [2]:
```
ratings = pd.read_csv(r"C:\Users\shaik\Downloads\archive\rating.csv")
```

In [3]:
```
ratings.shape
```

Out[3]: (20000263, 4)

In [4]:
```
ratings.head(3)
```

Out[4]:

	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

In [5]:
```
tags = pd.read_csv(r"C:\Users\shaik\Downloads\archive\tag.csv")
```

In [6]:
```
tags.shape
```

Out[6]: (465564, 4)

In [7]:
```
tags.head(3)
```

Out[7]:

	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

In [8]:
```
movies = pd.read_csv(r"C:\Users\shaik\Downloads\archive\movie.csv")
```

In [9]:
```
movies.shape
```

Out[9]: (27278, 3)

In [10]:
```
movies.head()
```

Out[10]:

	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 [11]:
```
print(ratings.columns)
print(tags.columns)
print(movies.columns)
```

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [12]:
```
del ratings["timestamp"]
del tags["timestamp"]
```

In [13]:
```
print(ratings.columns)
print(tags.columns)
print(movies.columns)
```

```
Index(['userId', 'movieId', 'rating'], dtype='object')
Index(['userId', 'movieId', 'tag'], dtype='object')
Index(['movieId', 'title', 'genres'], dtype='object')
```

series

In [14]:
```
tags.head(3)
```

Out[14]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero

In [15]:
```
row_0 = tags.iloc[0]
row_0
```

Out[15]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [16]:
```
row_1 = tags.iloc[1]
row_1
```

Out[16]:
```
userId            65
movieId          208
tag        dark hero
Name: 1, dtype: object
```

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

Out[17]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [18]:
```
row_0["userId"]
```

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

In [19]:
```
'rating' in row_0
```

Out[19]:
```
False
```

In [20]:
```
row_0.name
```

Out[20]:
```
0
```

In [21]:
```
row_0 = row_0.rename("first row")
row_0.name
```

Out[21]:
```
'first row'
```

DataFrames

In [22]:
```
tags.index
```

Out[22]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [23]:
```
tags.columns
```

Out[23]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [24]:
```
tags.iloc[[0,11,500]]
```

Out[24]:

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

Descriptive Statistics

In [25]:
```
ratings["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]:
```
ratings.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 [28]:
```
ratings["rating"].mean()
```

Out[28]:
```
np.float64(3.5255285642993797)
```

In [29]:
```
ratings.head(3)
```

Out[29]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5

In [30]:
```
ratings.mean()
```

Out[30]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [33]:
```
ratings['rating'].min() #minimum of rating columns
```

Out[33]:
```
0.5
```

In [34]:
```
ratings['rating'].max() #maximum of rating columns
```

Out[34]:
```
5.0
```

In [35]:
```
ratings['rating'].std() #standerd deviation of rating columns
```

Out[35]:
```
1.051988919275684
```

In [36]:
```
ratings['rating'].mode() #returns highest frequency value in the columns
```

Out[36]:
```
0    4.0
Name: rating, dtype: float64
```

In [37]:
```
ratings.mode()
```

Out[37]:

	userId	movieId	rating
0	118205	296	4.0

In [38]:
```
ratings.corr()
```

Out[38]:

	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 [40]:
```
filter1= ratings["rating"]>10
filter1
```

Out[40]:
```
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
```

In [42]:
```
filter2= ratings["rating"]>0
filter2
```

Out[42]:
```
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
```

Data Cleaning: Handling missing data

In [43]:
```
movies.shape
```

Out[43]:
```
(27278, 3)
```

In [44]:
```
movies.isnull().sum()
```

Out[44]:
```
movieId    0
title      0
genres     0
dtype: int64
```

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

Out[45]:
```
(20000263, 3)
```

In [46]:
```
ratings.isnull().sum()
```

Out[46]:
```
userId     0
movieId    0
rating     0
dtype: int64
```

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

Out[47]:
```
(465564, 3)
```

In [48]:
```
tags.isnull().sum()
```

Out[48]:
```
userId      0
movieId     0
tag        16
dtype: int64
```

In [49]:
```
tags= tags.dropna()
```

In [50]:
```
tags.isnull().sum()
```

Out[50]:
```
userId     0
movieId    0
tag        0
dtype: int64
```

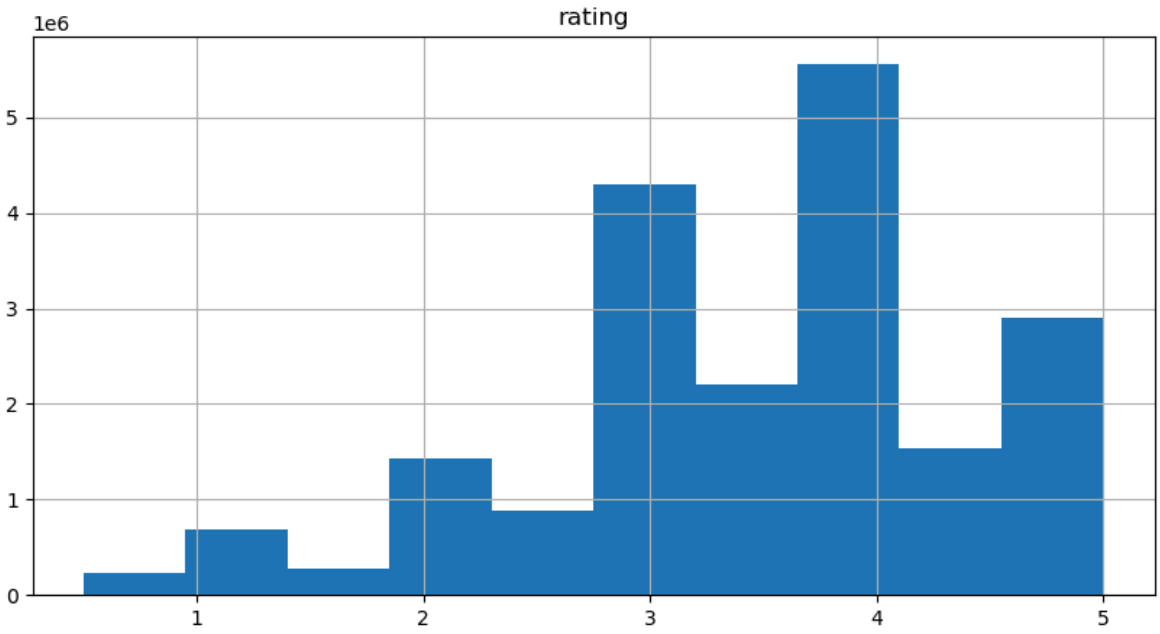
In [51]:
```
tags.shape
```

Out[51]:
```
(465548, 3)
```

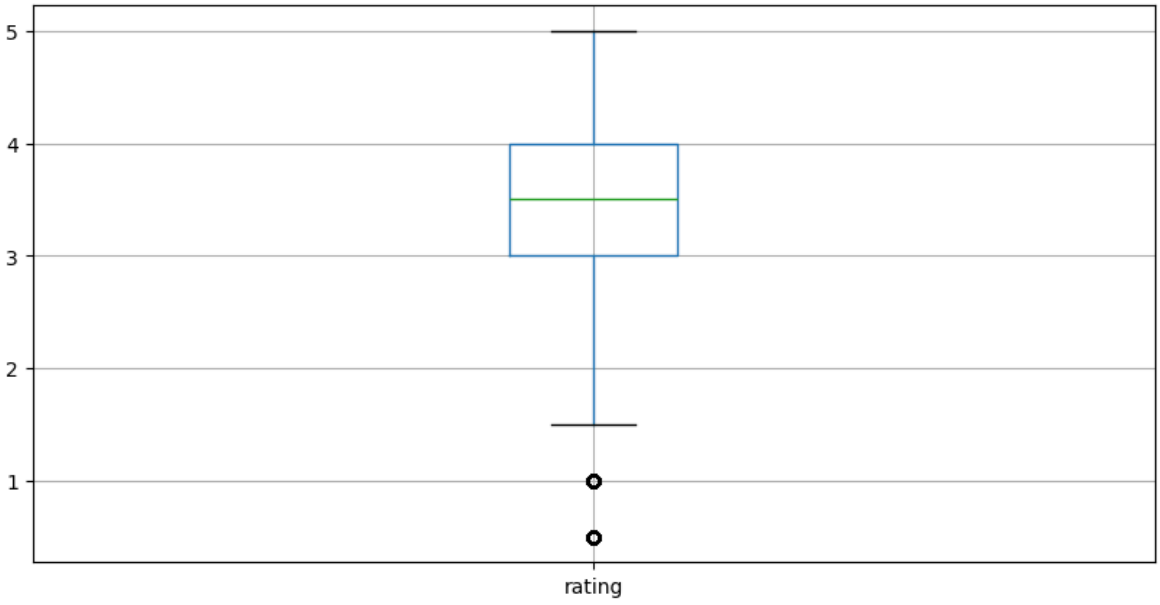
Data Visualization

In [53]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
ratings.hist(column='rating',figsize=(10,5))
plt.show()
```



In [54]:
```
ratings.boxplot(column='rating',figsize=(10,5))
plt.show()
```



Slicing out columns

In [55]:
```
tags['tag'].head()
```

Out[55]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [58]:
```
movies[['title','genres']].head()
```

Out[58]:

	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 [59]:
```
ratings[-10:]
```

Out[59]:

	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 [68]:
```
tag_counts= tags['tag'].value_counts()
tag_counts[-10:]
```

Out[68]:
```
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 [69]:
```
tag_counts.head(3)
```

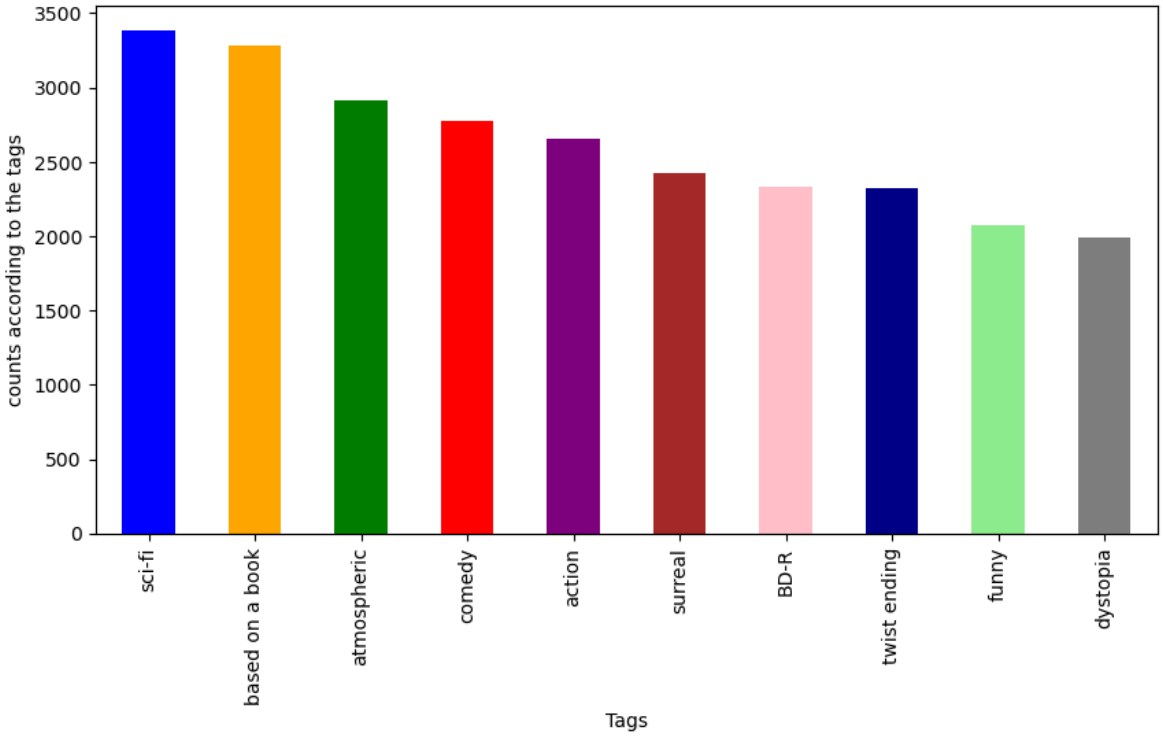
Out[69]:
```
tag
sci-fi            3384
based on a book   3281
atmospheric       2917
Name: count, dtype: int64
```

In [70]:
```
tag_counts.tail(3)
```

Out[70]:
```
tag
No arguement    1
Really Bad      1
Botox           1
Name: count, dtype: int64
```

In [77]:
```
tag_counts[:10].plot(kind="bar",figsize=(10,5),
                     xlabel = "Tags",
                     ylabel = "counts according to the tags",
                     color = ["blue","orange","green","red","purple","brown","pin
```

Out[77]:
```
<Axes: xlabel='Tags', ylabel='counts according to the tags'>
```



In [ ]: