Kaggle Workshop

Project 6 - IMDB Movie Data Analysis using Pandas

Step 1 - Import the libraries

In [3]:
```
import pandas as pd
```

Step 2 - Read the data set

In [5]:
```
movies = pd.read_csv(r'C:\Users\gadel\OneDrive\Desktop\Nareshit DataScience by Pra
```

In [7]:
```
ratings = pd.read_csv(r'C:\Users\gadel\OneDrive\Desktop\Nareshit DataScience by Pr
```

In [8]:
```
tags = pd.read_csv(r'C:\Users\gadel\OneDrive\Desktop\Nareshit DataScience by Praka
```

In [10]:
```
print(movies.shape)
print(ratings.shape)
print(tags.shape)
```

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

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

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

In [16]:
```
del ratings['timestamp']
del tags['timestamp']
# we are deleting the time stamp column in both the data sets
```

In [18]:
```
print(movies.columns)
print(ratings.columns)
print(tags.columns)
```

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

In [20]:
```
tags.head(2)
```

Out[20]:

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

📦 Data Structures:

🚦 Series

In [22]:
```
tags.iloc[0] # iloc --- index location which will be used in machine learning
```

Out[22]:
```
userId             18
movieId          4141
tag       Mark Waters
Name: 0, dtype: object
```

In [24]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[24]:
```
pandas.core.series.Series
```

In [26]:
```
print(row_0)
```

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

In [28]:
```
row_0.index
```

Out[28]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [30]:
```
row_0['userId']
```

Out[30]:
```
18
```

In [32]:
```
'rating' in row_0
```

Out[32]:
```
False
```

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

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

In [36]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[36]:
```
'firstRow'
```

▩ DataFrames

In [38]:
```
tags.head()
#it gives the top 5 rows
```

Out[38]:

	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 [40]:
```
tags.index
# it gives the range of the dataset
```

Out[40]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [42]:
```
tags.columns
```

Out[42]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [44]:
```
tags.iloc[ [0,11,500] ]
# iloc = index location it gives the rows for the specific inout that we give
```

Out[44]:

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

📈 📉 Descriptive Statistics¶

Let's look how the ratings are distributed!

In [55]:
```
ratings['rating'].describe()
# describe function give the description of the dataset or specific column by giv
```

Out[55]:
```
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 [59]:
```
ratings.describe()
```

Out[59]:

	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 [61]:
```
ratings['rating'].mean()
```
Out[61]:
```
3.5255285642993797
```

In [65]:
```
ratings.mean()
```
Out[65]:
```
userId      69045.872583
movieId      9041.567330
rating          3.525529
dtype: float64
```

In [67]:
```
ratings['rating'].min()
# gives the minimum or smallest value
```
Out[67]:
```
0.5
```

In [69]:
```
ratings['rating'].max()
# gives the maximum or largest value
```
Out[69]:
```
5.0
```

In [71]:
```
ratings['rating'].std()
```
Out[71]:
```
1.051988919275684
```

In [75]:
```
ratings['rating'].mode()
```
Out[75]:
```
0    4.0
Name: rating, dtype: float64
```

In [77]:
```
ratings.corr()
```
Out[77]:

	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 [79]:
```
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[79]:
```
False
```

In [81]:
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[81]:
```
True
```

🔧 Data Cleaning: Handling Missing Data

In [84]:
```
movies.shape
```

Out[84]:
```
(27278, 3)
```

In [86]:
```
movies.isnull().any().any()
```

Out[86]:
```
False
```

In [ ]:
```
-----------
Thats nice ! No NULL values !
-----------
```

In [88]:
```
ratings.shape
```

Out[88]:
```
(20000263, 3)
```

In [90]:
```
ratings.isnull().any().any()
```

Out[90]:
```
False
```

In [ ]:
```
-----------
Thats nice ! No NULL values !
-----------
```

In [92]:
```
tags.shape
```

Out[92]:
```
(465564, 3)
```

In [94]:
```
tags.isnull().any().any()
```

```
Out[94]:  True
```

```
In [ ]:  -----------
         We have some tags which are NULL.
         -----------
```

```
In [98]:  tags=tags.dropna()
```

```
In [100]:  tags.isnull().any().any()
```

```
Out[100]:  False
```

```
In [102]:  tags.shape
```

```
Out[102]:  (465548, 3)
```

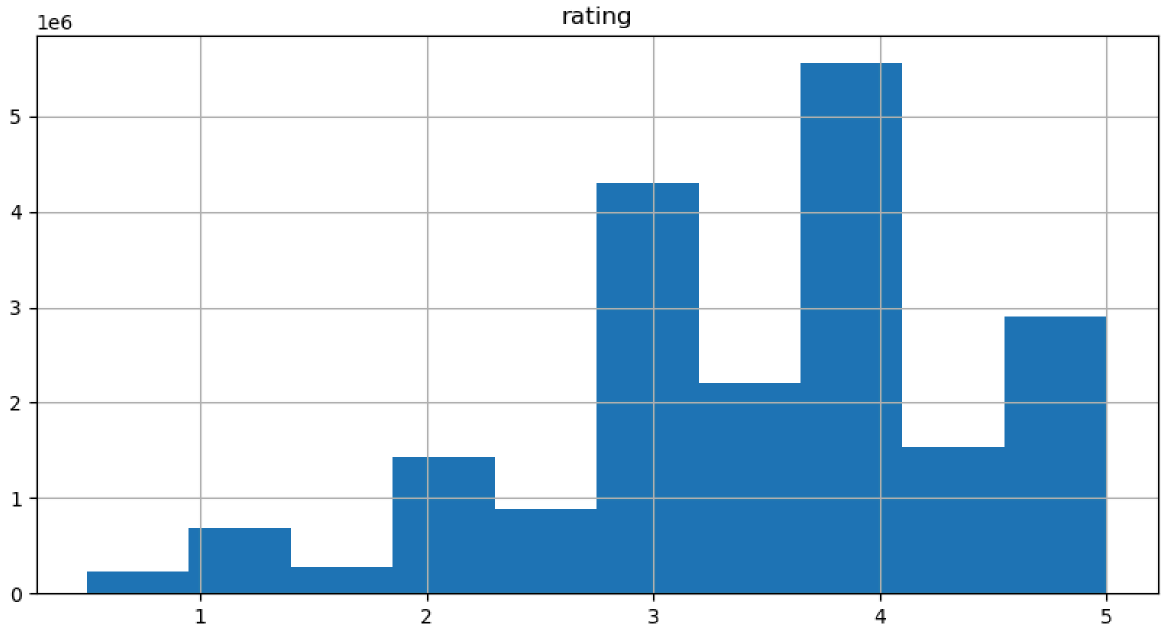
```
In [ ]:  -----------------
         Thats nice ! No NULL values ! Notice the number of lines have reduced.
         -----------------
```

📊 Data Visualization

```
In [107]:  import matplotlib.pyplot as plt
```

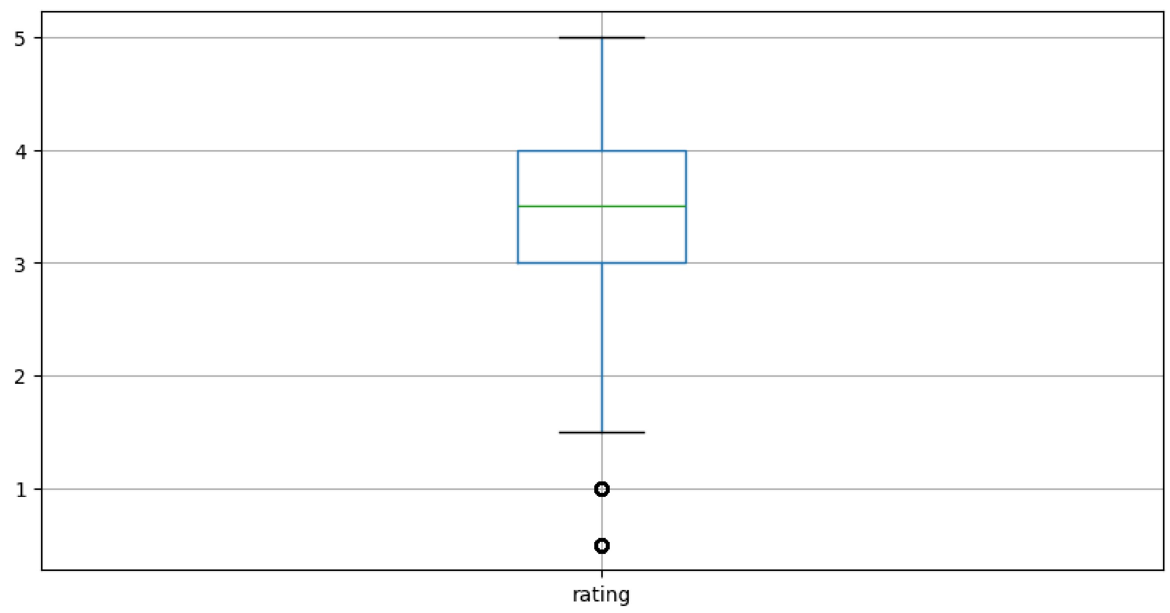
```
In [109]:  %matplotlib inline

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

```
Out[109]:  array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



```
In [112]:  ratings.boxplot(column='rating', figsize=(10,5))
```

```
Out[112]:  <Axes: >
```



📤 Slicing Out Columns

In [115]:
```
tags['tag'].head()
```

Out[115]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [117]:
```
movies[['title','genres']].head()
```

Out[117]:

	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 [119]:
```
ratings[-10:]
```

Out[119]:

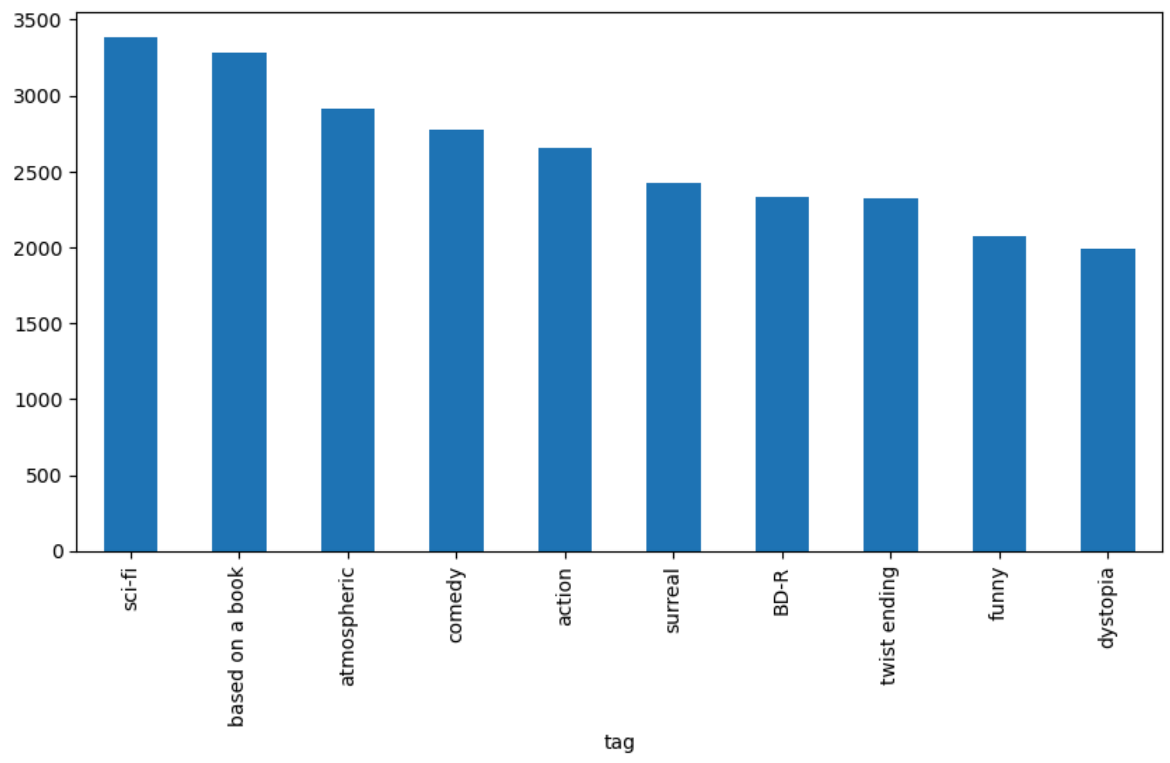
	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [121]:
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[121]:
```
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 [123]:
```
tag_counts[:10].plot(kind='bar', figsize=(10,5))
```

Out[123]:
```
<Axes: xlabel='tag'>
```


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 [ ]:
```