Kaggle Task - MovieLens 20M Dataset:

- https://www.kaggle.com/code/harunshimanto/pandas-with-data-science-ai

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

Read the Dataset

In [3]:
```
ratings = pd.read_csv (r'C:\Users\Vemuri.Meeravali\Downloads\archive\rating.csv'
```

In [4]:
```
ratings.shape
```

Out[4]: (20000263, 4)

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

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

Out[7]: (465564, 4)

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

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

Out[9]: (27278, 3)

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

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

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

```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                         Comedy|Drama|Romance
4                                       Comedy
   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
   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 [13]:
```
del ratings ['timestamp']
del tags ['timestamp']
```

In [14]:
```
print (movies.columns)
print (tags.columns)
print (ratings.columns)
```

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

Data Structure - Series

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

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

In [16]:
```
type (row_0)
```

Out[16]:
```
pandas.core.series.Series
```

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]:
```
row_0.name
```

Out[19]:
```
0
```

In [20]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[20]:
```
'firstRow'
```

Data Frames

In [21]:
```
tags.head()
```

Out[21]:

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

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

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

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

Descriptive Statistics

In [24]:
```
ratings['rating'].describe()
```

Out[24]:
```
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 [25]:
```
ratings.describe()
```

Out[25]:

	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 [26]:
```
ratings['rating'].mean()
```

Out[26]:
```
np.float64(3.5255285642993797)
```

In [27]:
```
ratings.mean()
```

Out[27]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [28]:
```
ratings['rating'].min()
```

Out[28]:
```
0.5
```

In [29]:
```
ratings['rating'].max()
```

Out[29]:
```
5.0
```

In [30]:
```
ratings['rating'].std()
```

Out[30]:
```
1.051988919275684
```

In [31]:
```
ratings['rating'].mode()
```

Out[31]:
```
0    4.0
Name: rating, dtype: float64
```

In [32]:
```
filter1 = ratings['rating']>10
print(filter1)
```

```
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 [33]:
```
filter1.any()
```

Out[33]:
```
np.False_
```

In [34]:
```
filter1.all()
```

Out[34]:
```
np.False_
```

Data Cleaning:

- used to Handling Missing Data

In [35]:
```
movies.shape
```

Out[35]:
```
(27278, 3)
```

In [38]:
```
movies.isnull().any().any()
```

Out[38]:
```
np.False_
```

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

Out[39]:
```
(20000263, 3)
```

In [40]:
```
ratings.isnull().any().any()
```

Out[40]:
```
np.False_
```

In [41]:
```
tags.shape
```

Out[41]:
```
(465564, 3)
```

In [42]:
```
tags.isnull().any()
```

Out[42]:
```
userId     False
movieId    False
tag         True
dtype: bool
```

Data Visulization

In [43]:
```
ratings
```

Out[43]:

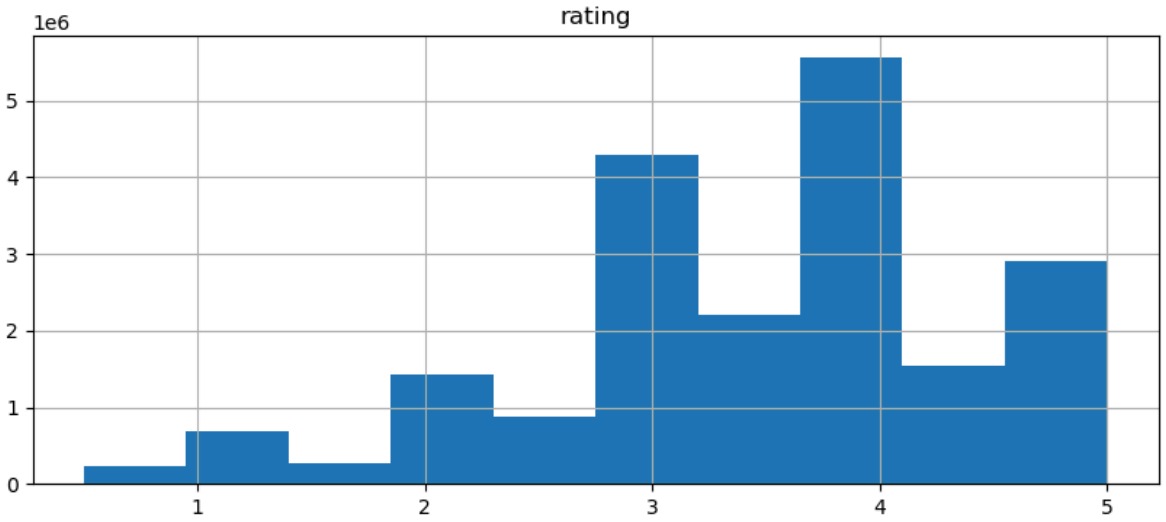
	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5
...	...	...	...
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

20000263 rows × 3 columns

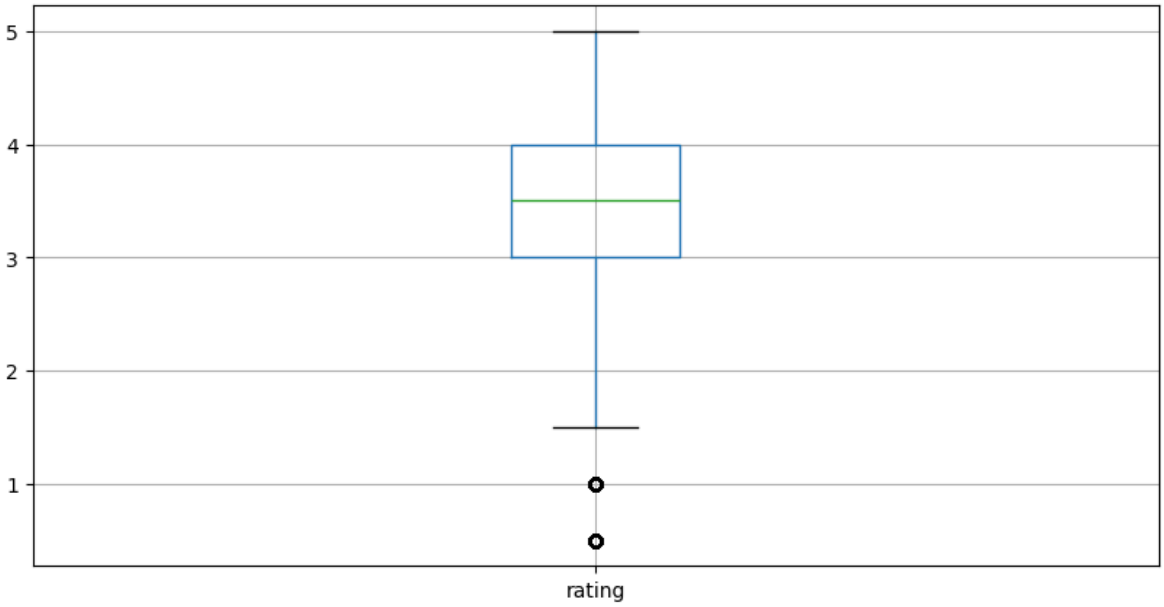
In [44]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [46]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [47]:
```
ratings.hist(column='rating', figsize=(10,4))
plt.show()
```



In [51]:
```
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 [ ]:
```
#
```