Impoting libraries

In [2]:
```
import pandas as pd
```

Reading the Dataset

In [4]:
```
movies = pd.read_csv(r"C:\Users\mohap\Downloads\IMDB RATING ANALYSIS\movie.csv")
```

In [8]:
```
movies.shape #rows,attributes
```

Out[8]:
```
(27278, 3)
```

In [11]:
```
tags= pd.read_csv(r"C:\Users\mohap\Downloads\IMDB RATING ANALYSIS\tag.csv")
```

In [13]:
```
tags.shape
```

Out[13]:
```
(465564, 4)
```

In [15]:
```
ratings = pd.read_csv(r"C:\Users\mohap\Downloads\IMDB RATING ANALYSIS\rating.csv
```

In [19]:
```
ratings.shape
```

Out[19]:
```
(20000263, 4)
```

In [21]:
```
movies.head() #by defeault top 5 rows
```

Out[21]:

	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 [23]:
```
movies.head(10)
```

Out[23]:

	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

In [25]:
```
tags.head()
```

Out[25]:

	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.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]:
```
ratings.head()
```

Out[29]:

	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

In [31]:
```
ratings.columns
```

Out[31]:
```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

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

Out[35]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [37]:
```
tags.columns
```

Out[37]:
```
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [39]:
```
del ratings['timestamp']
del tags['timestamp']   #deleting
```

In [41]:
```
ratings.columns
```

Out[41]:
```
Index(['userId', 'movieId', 'rating'], dtype='object')
```

In [43]:
```
tags.columns
```

Out[43]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

Data Structure

In [46]:
```
row_0 = tags.iloc[0] #Location based indexing for selection by position.
```

In [48]:
```
type(row_0)
```

Out[48]:
```
pandas.core.series.Series
```

In [50]:
```
print(row_0)
```

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

In [52]:
```
tags
```

Out[52]:

	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 [54]:
```
row_0.index
```

Out[54]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [56]:
```
row_0['userId']
```

Out[56]:
```
18
```

In [58]:
```
'rating' in row_0
```

Out[58]:
```
False
```

In [60]:
```
'movieId' in row_0
```

Out[60]:
```
True
```

In [62]:
```
row_0.name
```

Out[62]:
```
0
```

In [64]:
```
row_0 = row_0.rename('firstrow')
row_0.name #renaming the row
```

Out[64]:
```
'firstrow'
```

In [66]:
```
row_0
```

Out[66]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: firstrow, dtype: object
```

Data Frame

In [69]:
```
tags.head()
```

Out[69]:

	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 [71]:
```
tags.index
```

Out[71]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [73]:
```
tags.columns
```

Out[73]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [77]:
```
tags
```

Out[77]:

	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 [81]:
```
tags.iloc[[0,67,456]]
```

Out[81]:

	userId	movieId	tag
0	18	4141	Mark Waters
67	121	5283	College Humor
456	342	47044	Miami

Slicing

In [86]:
```
ratings.head(23)
```

Out[86]:

	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
5	1	112	3.5
6	1	151	4.0
7	1	223	4.0
8	1	253	4.0
9	1	260	4.0
10	1	293	4.0
11	1	296	4.0
12	1	318	4.0
13	1	337	3.5
14	1	367	3.5
15	1	541	4.0
16	1	589	3.5
17	1	593	3.5
18	1	653	3.0
19	1	919	3.5
20	1	924	3.5
21	1	1009	3.5
22	1	1036	4.0

In [88]:
```
ratings[0:233]
```

Out[88]:

	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
...	...	...	...
228	2	3918	3.0
229	2	3923	4.0
230	2	3926	4.0
231	2	3927	5.0
232	2	3928	5.0

233 rows × 3 columns

In [90]:
```
ratings[:657]
```

Out[90]:

	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
...	...	...	...
652	7	2081	3.0
653	7	2108	3.0
654	7	2112	3.0
655	7	2125	3.0
656	7	2146	3.0

657 rows × 3 columns

In [94]:
```
ratings[0:25:6]
```

Out[94]:

	userId	movieId	rating
0	1	2	3.5
6	1	151	4.0
12	1	318	4.0
18	1	653	3.0
24	1	1080	3.5

In [96]:
```
ratings[::-1]
```

Out[96]:

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

20000263 rows × 3 columns

In [100…
```
ratings[999:1222:56]
```

Out[100…

	userId	movieId	rating
999	11	519	1.0
1055	11	1255	4.0
1111	11	2572	5.0
1167	11	3991	3.0

Descriptive Statistics

In [103…
```
ratings.describe()
```

Out[103…

	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 [105…

```
ratings.describe().T #Transpose:rows to column
```

Out[105…

	count	mean	std	min	25%	50%	75%	
userId	20000263.0	69045.872583	40038.626653	1.0	34395.0	69141.0	103637.0	1384
movieId	20000263.0	9041.567330	19789.477445	1.0	902.0	2167.0	4770.0	1312
rating	20000263.0	3.525529	1.051989	0.5	3.0	3.5	4.0	

In [107…

```
ratings.mean()
```

Out[107…

```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [109…

```
ratings
```

Out[109…

	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 [111…
```
ratings['rating'].describe()
```

Out[111…
```
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 [113…
```
ratings['rating'].min()
```

Out[113…
```
0.5
```

In [115…
```
ratings['rating'].max()
```

Out[115…
```
5.0
```

In [117…
```
ratings['rating'].std() #Return sample standard deviation over requested axis
```

Out[117…
```
1.051988919275684
```

In [119…
```
ratings['rating'].mode() #the highest population distribution,robust measure of
```

Out[119…
```
0    4.0
Name: rating, dtype: float64
```

In [124…
```
ratings.corr()#The corr() method finds the correlation of each column in a DataF
```

Out[124…

	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 [130…
```
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 [132…
```
filter1.any()
```

Out[132… False

In [134…
```
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[134… False

In [144…
```
filter2 = ratings['rating']>0
```

In [146…
```
filter2.all()
```

Out[146… True

Data Cleaning: Handling missing Values

In [152…
```
movies.shape
```

Out[152… (27278, 3)

In [156…
```
movies.isnull()
```

Out[156…

	movieId	title	genres
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
…	…	…	…
27273	False	False	False
27274	False	False	False
27275	False	False	False
27276	False	False	False
27277	False	False	False

27278 rows × 3 columns

In [158…
```
movies.isnull().any().any() #No null values
```

Out[158… False

In [160…
```
ratings.shape
```

Out[160… (20000263, 3)

In [162…
```
ratings.isnull().any().any()
```

Out[162… False

In [166…
```
tags.shape
```

Out[166… (465564, 3)

In [168…
```
tags.isnull().any().any() #HERE IS NULL VALUES
```

Out[168… True

so we have null values in tags

In [171…
```
tags = tags.dropna()#dropna function in Python is a method used with pandas Data
```

In [173…
```
tags.isnull().any().any()
```

Out[173… False

In [175…
```
tags.shape
```

Out[175… (465548, 3)

Now we have NO Null values

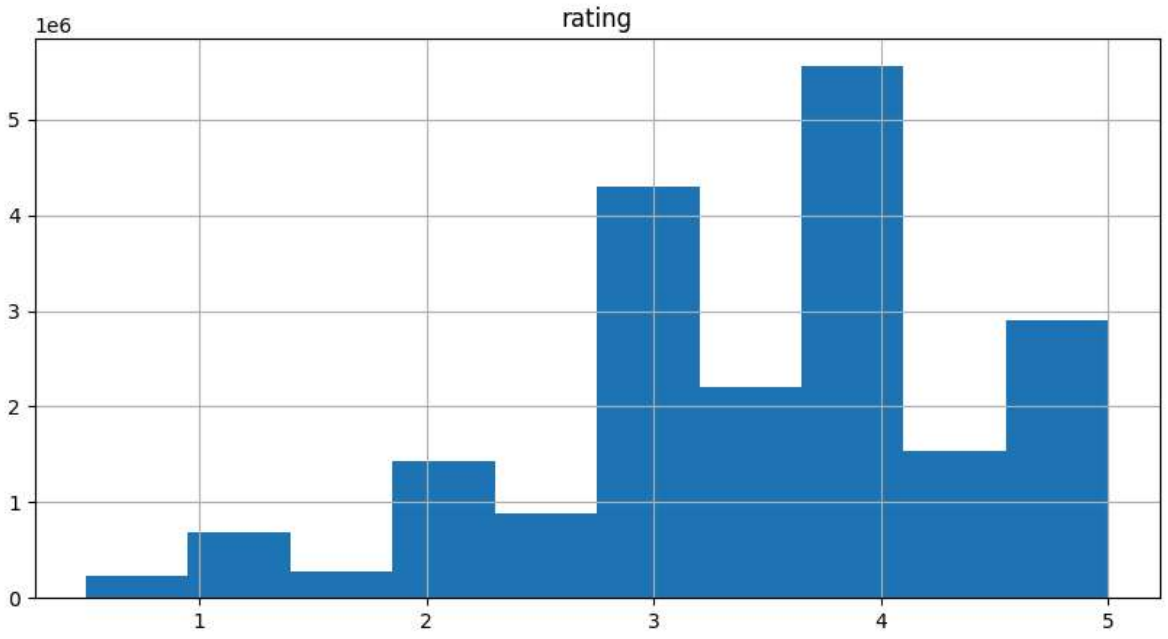
dropna() function in Python is a method used with pandas DataFrames and Series to remove missing values (NaNs)

Data Visualization

In [195…
```
%matplotlib inline

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

Out[195… array([[<Axes: title={'center': 'rating'}>]], dtype=object)



%matplotlib inline ensures that the plot appears

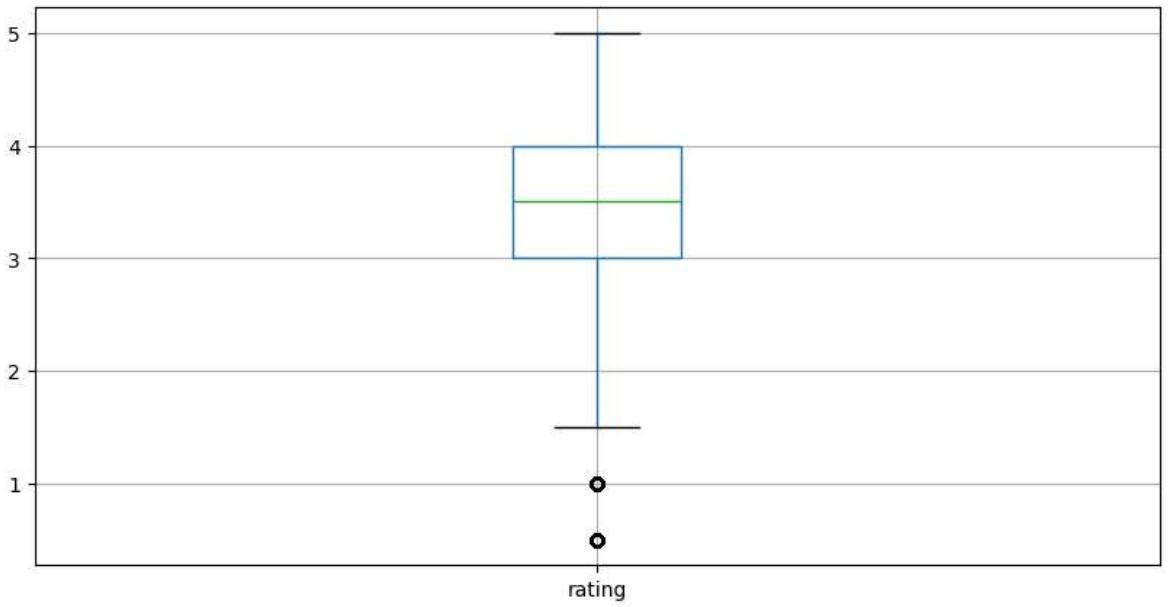
directly in the notebook.

10-width ,5-height

figsize is an important parameter that allows us to set the size of the figure

In [200…
```
ratings.boxplot(column='rating',figsize=(10,5))
```

Out[200…
```
<Axes: >
```



Slicing Out Columns

In [213…
```
tags['tag'].head()
```

Out[213…
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [215…
```
movies
```

Out[215…

	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 [217…

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

Out[217…

	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 [219…

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

Out[219…

	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 [221…

```
tags_counts = tags['tag'].value_counts() #counts occurence of unique values in s
tags_counts[-10:]
```

Out[221…

```
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 [223…

```
tags
```

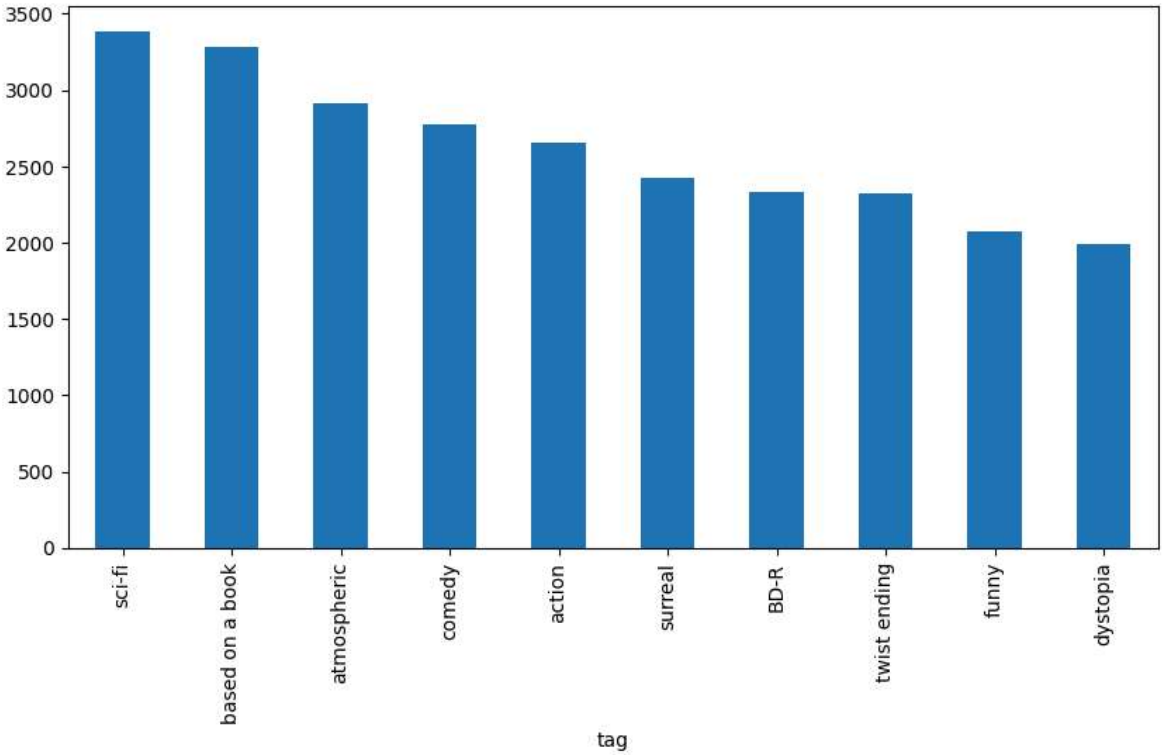
Out[223...

	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 [231...
```
tags_counts[:10].plot(kind='bar', figsize=(10,5)) #bar graph
```

Out[231...
```
<Axes: xlabel='tag'>
```



Fliters for selecting rows

In [248...
```
ratings['rating']
```

Out[248…

```
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 [242…

```
high_rated = ratings['rating'] >=5.0
ratings[high_rated][30:50]
```

Out[242…

	userId	movieId	rating
239	3	50	5.0
242	3	175	5.0
244	3	223	5.0
245	3	260	5.0
246	3	316	5.0
247	3	318	5.0
248	3	329	5.0
252	3	457	5.0
253	3	480	5.0
254	3	490	5.0
256	3	541	5.0
258	3	593	5.0
263	3	858	5.0
264	3	904	5.0
267	3	924	5.0
268	3	953	5.0
271	3	1060	5.0
272	3	1073	5.0
275	3	1084	5.0
276	3	1089	5.0

In [252…

```
movies['genres']
```

Out[252...

```
0        Adventure|Animation|Children|Comedy|Fantasy
1                          Adventure|Children|Fantasy
2                                       Comedy|Romance
3                                Comedy|Drama|Romance
4                                              Comedy
                              ...
27273                                          Comedy
27274                                          Comedy
27275                                       Adventure
27276                             (no genres listed)
27277                          Adventure|Fantasy|Horror
Name: genres, Length: 27278, dtype: object
```

In [254...

```
action = movies['genres'].str.contains('Action')
movies[action][5:15]
```

Out[254...

	movieId	title	genres
22	23	Assassins (1995)	Action|Crime|Thriller
41	42	Dead Presidents (1995)	Action|Crime|Drama
43	44	Mortal Kombat (1995)	Action|Adventure|Fantasy
50	51	Guardian Angel (1994)	Action|Drama|Thriller
65	66	Lawnmower Man 2: Beyond Cyberspace (1996)	Action|Sci-Fi|Thriller
69	70	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
70	71	Fair Game (1995)	Action
75	76	Screamers (1995)	Action|Sci-Fi|Thriller
77	78	Crossing Guard, The (1995)	Action|Crime|Drama|Thriller
85	86	White Squall (1996)	Action|Adventure|Drama

In [256...

```
movies[action].head(15)
```

Out[256...

	movieId	title	genres
5	6	Heat (1995)	Action|Crime|Thriller
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
22	23	Assassins (1995)	Action|Crime|Thriller
41	42	Dead Presidents (1995)	Action|Crime|Drama
43	44	Mortal Kombat (1995)	Action|Adventure|Fantasy
50	51	Guardian Angel (1994)	Action|Drama|Thriller
65	66	Lawnmower Man 2: Beyond Cyberspace (1996)	Action|Sci-Fi|Thriller
69	70	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
70	71	Fair Game (1995)	Action
75	76	Screamers (1995)	Action|Sci-Fi|Thriller
77	78	Crossing Guard, The (1995)	Action|Crime|Drama|Thriller
85	86	White Squall (1996)	Action|Adventure|Drama

Group By Aggregate

In [258...

```
# Group by aggregate
ratings_count = ratings[['movieId','rating']].groupby('rating').count()
ratings_count
```

Out[258...

rating	movieId
0.5	239125
1.0	680732
1.5	279252
2.0	1430997
2.5	883398
3.0	4291193
3.5	2200156
4.0	5561926
4.5	1534824
5.0	2898660

In [267...

```
average_rating = ratings[['movieId','rating']].groupby('movieId').mean()
average_rating.head()
```

Out[267...

movieId	rating
1	3.921240
2	3.211977
3	3.151040
4	2.861393
5	3.064592

In [269...

```
average_rating = ratings[['movieId','rating']].groupby('movieId').mean()
average_rating.tail()
```

Out[269...

movieId	rating
131254	4.0
131256	4.0
131258	2.5
131260	3.0
131262	4.0

Merge Dataframe

In [272…
```
tags.head()
```

Out[272…

	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 [274…
```
movies.head()
```

Out[274…

	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 [278…
```
x = movies.merge(tags, on = 'movieId', how = 'inner')
x.head()
```

Out[278…

	movieId	title	genres	userId	tag
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1644	Watched
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	computer animation
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	Disney animated feature
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	Pixar animation
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1741	TÃ©a Leoni does not star in this movie

In [ ]:
```
## Certainly! Let's break down the line of code you provided, which is using the

### Code Breakdown
```

```
# 1. **`x =`**:
   # - This part of the code is assigning the result of the merge operation to a

# 2. **`movies.merge(...)`**:
##   - Here, we are calling the `merge` method on the `movies` DataFrame. The `m

# 3. **`tag`**:
 #  - This is the second DataFrame that we are merging with `movies`. It is assu

# 4. **`on = 'movieId'`**:
  # - This parameter specifies the column on which the two DataFrames will be me

# 5. **`how = 'inner'`**:
 ##  - This parameter specifies the type of merge to be performed. An "inner" me

# 6. **`x.head()`**:
 ##  - This part of the code is not part of the merge operation itself but is a

### Summary

## In summary, the line of code merges the `movies`
## DataFrame with the `tag` DataFrame on the `movieId` column using an inner joi
## meaning only the rows with matching `movieId` values in both DataFrames will
##The merged DataFrame is then stored in the variable `x`, and `x.head()` is cal
```

In [286…

```
avg_ratings = ratings.groupby('movieId',as_index=False).mean()
del avg_ratings['userId']
avg_ratings.head()
```

Out[286…

	movieId	rating
0	1	3.921240
1	2	3.211977
2	3	3.151040
3	4	2.861393
4	5	3.064592

In [292…

```
box_office = movies.merge(avg_ratings, on='movieId',how='inner')
box_office.tail()
```

Out[292…

	movieId	title	genres	rating
26739	131254	Kein Bund für's Leben (2007)	Comedy	4.0
26740	131256	Feuer, Eis & Dosenbier (2002)	Comedy	4.0
26741	131258	The Pirates (2014)	Adventure	2.5
26742	131260	Rentun Ruusu (2001)	(no genres listed)	3.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

In [296…

```
high_rated = box_office['rating'] >= 4.0
box_office[high_rated][-5:]
```

Out[296...

	movieId	title	genres	rating
26737	131250	No More School (2000)	Comedy	4.0
26738	131252	Forklift Driver Klaus: The First Day on the Jo...	Comedy|Horror	4.0
26739	131254	Kein Bund für's Leben (2007)	Comedy	4.0
26740	131256	Feuer, Eis & Dosenbier (2002)	Comedy	4.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

In [298...

```
Adventure = box_office['genres'].str.contains('Adventure')
box_office[Adventure][:8]
```

Out[298...

	movieId	title	genres	rating
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3.921240
1	2	Jumanji (1995)	Adventure|Children|Fantasy	3.211977
7	8	Tom and Huck (1995)	Adventure|Children	3.142049
9	10	GoldenEye (1995)	Action|Adventure|Thriller	3.430029
12	13	Balto (1995)	Adventure|Animation|Children	3.272416
14	15	Cutthroat Island (1995)	Action|Adventure|Romance	2.721993
28	29	City of Lost Children, The (Cité des enfants p...	Adventure|Drama|Fantasy|Mystery|Sci-Fi	3.952230
32	33	Wings of Courage (1995)	Adventure|Romance|IMAX	3.007692

In [302...

```
box_office[Adventure & high_rated][-10:]
```

Out[302…

	movieId	title	genres	rating
25940	126953	The Fabulous Baron Munchausen (1962)	Adventure|Animation|Fantasy	4.0
26045	127230	The World Forgotten (2011)	Adventure|Documentary	4.0
26059	127262	The Count of Monte Cristo (1954)	Adventure|Drama|Romance	4.0
26186	128508	Valley Of Flowers (2006)	Adventure|Drama|Romance	4.5
26601	130518	The Amazing Screw-On Head (2006)	Action|Adventure|Animation|Comedy|Sci-Fi	4.0
26611	130586	Itinerary of a Spoiled Child (1988)	Adventure|Drama	4.5
26655	130996	The Beautiful Story (1992)	Adventure|Drama|Fantasy	5.0
26667	131050	Stargate SG-1 Children of the Gods - Final Cut...	Adventure|Sci-Fi|Thriller	5.0
26736	131248	Brother Bear 2 (2006)	Adventure|Animation|Children|Comedy|Fantasy	4.0
26743	131262	Innocence (2014)	Adventure|Fantasy|Horror	4.0

Vectorize String operation

In [307…

```
movies.head()
```

Out[307…

	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 [309…

```
# split 'generes ' into multiple columns
```

In [311…
```
genres = movies['genres'].str.split('|',expand=True)
```

In [313…
```
genres[:20]
```

Out[313…

	0	1	2	3	4	5	6	7	8	9
0	Adventure	Animation	Children	Comedy	Fantasy	None	None	None	None	None
1	Adventure	Children	Fantasy	None	None	None	None	None	None	None
2	Comedy	Romance	None	None	None	None	None	None	None	None
3	Comedy	Drama	Romance	None	None	None	None	None	None	None
4	Comedy	None	None	None	None	None	None	None	None	None
5	Action	Crime	Thriller	None	None	None	None	None	None	None
6	Comedy	Romance	None	None	None	None	None	None	None	None
7	Adventure	Children	None	None	None	None	None	None	None	None
8	Action	None	None	None	None	None	None	None	None	None
9	Action	Adventure	Thriller	None	None	None	None	None	None	None
10	Comedy	Drama	Romance	None	None	None	None	None	None	None
11	Comedy	Horror	None	None	None	None	None	None	None	None
12	Adventure	Animation	Children	None	None	None	None	None	None	None
13	Drama	None	None	None	None	None	None	None	None	None
14	Action	Adventure	Romance	None	None	None	None	None	None	None
15	Crime	Drama	None	None	None	None	None	None	None	None
16	Drama	Romance	None	None	None	None	None	None	None	None
17	Comedy	None	None	None	None	None	None	None	None	None
18	Comedy	None	None	None	None	None	None	None	None	None
19	Action	Comedy	Crime	Drama	Thriller	None	None	None	None	None

In [315…
```
# Add new comedy genres
```

In [317…
```
genres['isComedy'] = movies['genres'].str.contains('Comedy')
```

In [319…
```
genres[:10]
```

Out[319...

	0	1	2	3	4	5	6	7	8	9
0	Adventure	Animation	Children	Comedy	Fantasy	None	None	None	None	None
1	Adventure	Children	Fantasy	None	None	None	None	None	None	None
2	Comedy	Romance	None	None	None	None	None	None	None	None
3	Comedy	Drama	Romance	None	None	None	None	None	None	None
4	Comedy	None	None	None	None	None	None	None	None	None
5	Action	Crime	Thriller	None	None	None	None	None	None	None
6	Comedy	Romance	None	None	None	None	None	None	None	None
7	Adventure	Children	None	None	None	None	None	None	None	None
8	Action	None	None	None	None	None	None	None	None	None
9	Action	Adventure	Thriller	None	None	None	None	None	None	None

In [333...
```
# Parsing Timestamp
```

In [337...
```
tags = pd.read_csv(r"C:\Users\mohap\Downloads\IMDB RATING ANALYSIS\tag.csv", sep
```

In [339...
```
movies.tail()
```

Out[339...

	movieId	title	genres	year
27273	131254	Kein Bund für's Leben (2007)	Comedy	2007
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy	2002
27275	131258	The Pirates (2014)	Adventure	2014
27276	131260	Rentun Ruusu (2001)	(no genres listed)	2001
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror	2014

In [341...
```
tags.dtypes
```

Out[341...
```
userId        int64
movieId       int64
tag          object
timestamp    object
dtype: object
```

In [345...
```
tags.head(10)
```

Out[345...

	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
5	65	668	bollywood	2013-05-10 01:37:56
6	65	898	screwball comedy	2013-05-10 01:42:40
7	65	1248	noir thriller	2013-05-10 01:39:43
8	65	1391	mars	2013-05-10 01:40:55
9	65	1617	neo-noir	2013-05-10 01:43:37

In [354...

```
# Average Movie Ratings over time
```

In [356...

```
ratings = ratings[['movieId','rating']].groupby('movieId', as_index=False).mean(
ratings.tail()
```

Out[356...

	movieId	rating
26739	131254	4.0
26740	131256	4.0
26741	131258	2.5
26742	131260	3.0
26743	131262	4.0

In [362...

```
joined = movies.merge(average_rating, on='movieId', how='inner')
joined.head()
joined.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[362], line 3
      1 joined = movies.merge(ratings, on='movieId', how='inner')
      2 joined.head()
----> 3 joined.corr()

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:11049,
in DataFrame.corr(self, method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:1993,
in DataFrame.to_numpy(self, dtype, copy, na_value)
   1991 if dtype is not None:
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1994 if result.dtype is not dtype:
   1995     result = np.asarray(result, dtype=dtype)

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\manag
ers.py:1694, in BlockManager.as_array(self, dtype, copy, na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\manag
ers.py:1753, in BlockManager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

ValueError: could not convert string to float: 'Toy Story (1995)'
```

In [ ]: