

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

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

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

```
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
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40

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

```
In [9]: tags.head()
```

```
Out[9]:
```

	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 [11]: movie = pd.read_csv(r'C:\Users\LENOVO\Downloads\archive\movie.csv')
```

```
In [13]: movie.head()
```

```
Out[13]:
```

	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 [15]: del ratings['timestamp']
del tags['timestamp']
```

# Data Structures:

## \*Series\*

```
In [19]: row_0 = tags.iloc[0]  
         type(row_0)
```

```
Out[19]: pandas.core.series.Series
```

```
In [21]: print(row_0)  
  
userId          18  
movieId        4141  
tag            Mark Waters  
Name: 0, dtype: object
```

```
In [23]: row_0.index
```

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

```
In [25]: row_0['userId']
```

```
Out[25]: 18
```

```
In [27]: 'rating' in row_0
```

```
Out[27]: False
```

```
In [29]: row_0.name
```

```
Out[29]: 0
```

```
In [31]: row_0 = row_0.rename('firstRow')  
         row_0.name
```

```
Out[31]: 'firstRow'
```

# DataFrames

```
In [33]: tags.head()
```

```
Out[33]:
```

	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 [35]: tags.index
```

```
Out[35]: RangeIndex(start=0, stop=465564, step=1)
```

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

```
Out[37]: 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

## Descriptive Statistics

```
In [44]: ratings['rating'].describe()
```

```
Out[44]: 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 [46]: ratings.describe()
```

Out[46]:

	userId	movieId	rating
<b>count</b>	2.000026e+07	2.000026e+07	2.000026e+07
<b>mean</b>	6.904587e+04	9.041567e+03	3.525529e+00
<b>std</b>	4.003863e+04	1.978948e+04	1.051989e+00
<b>min</b>	1.000000e+00	1.000000e+00	5.000000e-01
<b>25%</b>	3.439500e+04	9.020000e+02	3.000000e+00
<b>50%</b>	6.914100e+04	2.167000e+03	3.500000e+00
<b>75%</b>	1.036370e+05	4.770000e+03	4.000000e+00
<b>max</b>	1.384930e+05	1.312620e+05	5.000000e+00

In [48]: ratings['rating'].mean()

Out[48]: 3.5255285642993797

In [50]: ratings.mean()

Out[50]:

userId	69045.872583
movieId	9041.567330
rating	3.525529
dtype:	float64

In [52]: ratings['rating'].min()

Out[52]: 0.5

In [56]: ratings['rating'].max()

Out[56]: 5.0

In [58]: ratings['rating'].std()

Out[58]: 1.051988919275684

In [60]: ratings['rating'].mode()

Out[60]:

0	4.0
Name:	rating, dtype: float64

In [62]: ratings.corr()

Out[62]:

	userId	movieId	rating
<b>userId</b>	1.000000	-0.000850	0.001175
<b>movieId</b>	-0.000850	1.000000	0.002606
<b>rating</b>	0.001175	0.002606	1.000000

```
In [68]: 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[68]: False

```
In [70]: filter2 = ratings['rating']>0
filter2.all()
```

Out[70]: True

## Data Cleaning: Handling Missing Data

```
In [75]: movie.shape
```

Out[75]: (27278, 3)

```
In [79]: movie.isnull().any().any()
```

Out[79]: False

```
In [81]: ratings.shape
```

Out[81]: (20000263, 3)

```
In [83]: tags.isnull().any().any()
```

Out[83]: True

```
In [87]: tags=tags.dropna()
```

```
In [89]: tags.isnull().any().any()
```

```
Out[89]: False
```

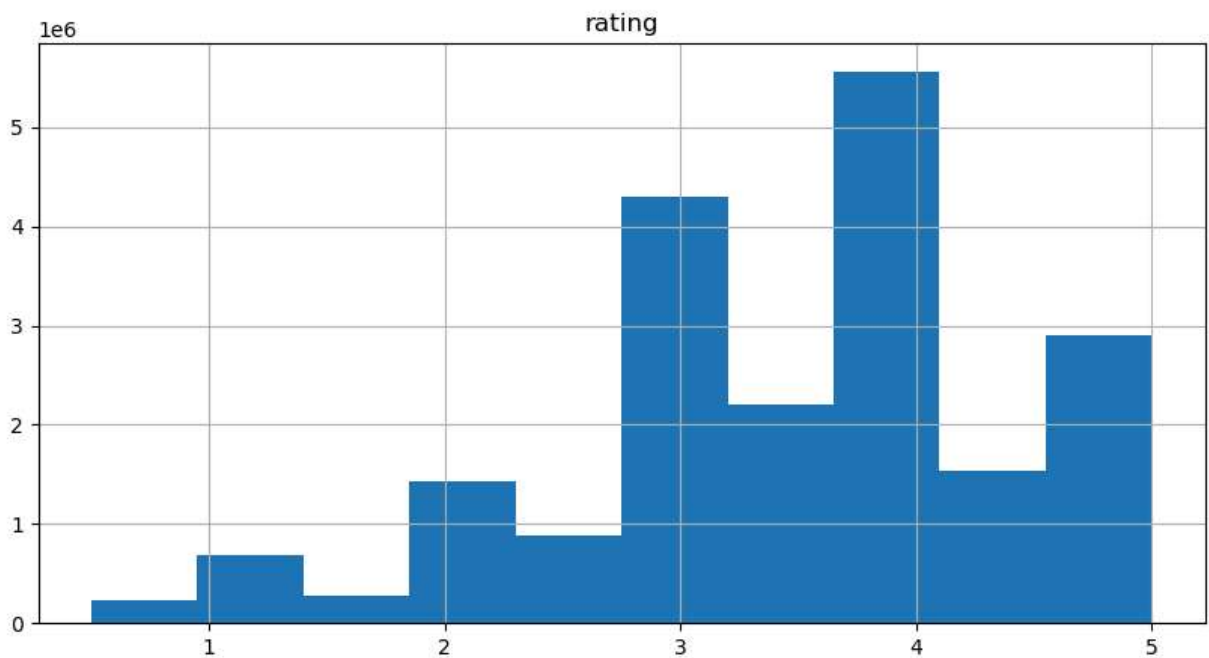
```
In [91]: tags.shape
```

```
Out[91]: (465548, 3)
```

## Data Visualization

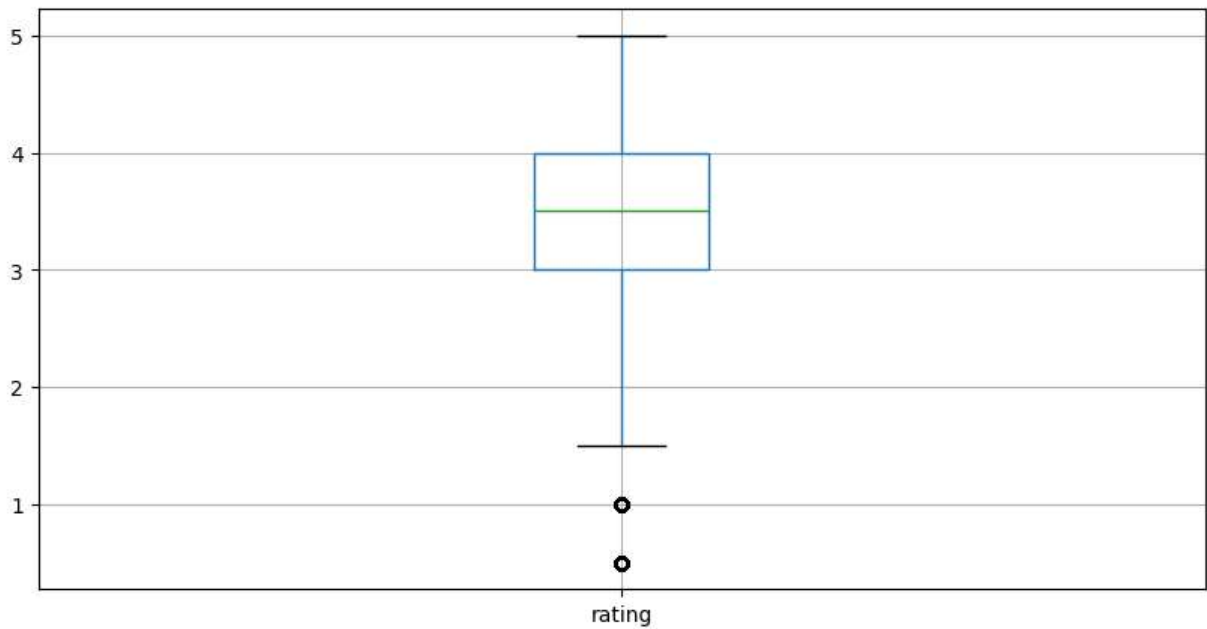
```
In [96]: %matplotlib inline
ratings.hist(column='rating',figsize=(10,5))
```

```
Out[96]: array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



```
In [98]: ratings.boxplot(column='rating',figsize=(10,5))
```

```
Out[98]: <Axes: >
```



## Slicing Out Columns

In [101... `tags['tag'].head()`

Out[101...   
 0 Mark Waters  
 1 dark hero  
 2 dark hero  
 3 noir thriller  
 4 dark hero  
 Name: tag, dtype: object

In [105... `movie[['title', 'genres']].head()`

Out[105... 

	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 [107... `ratings[-10:]`

Out[107...

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

In [113...

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

Out[113...

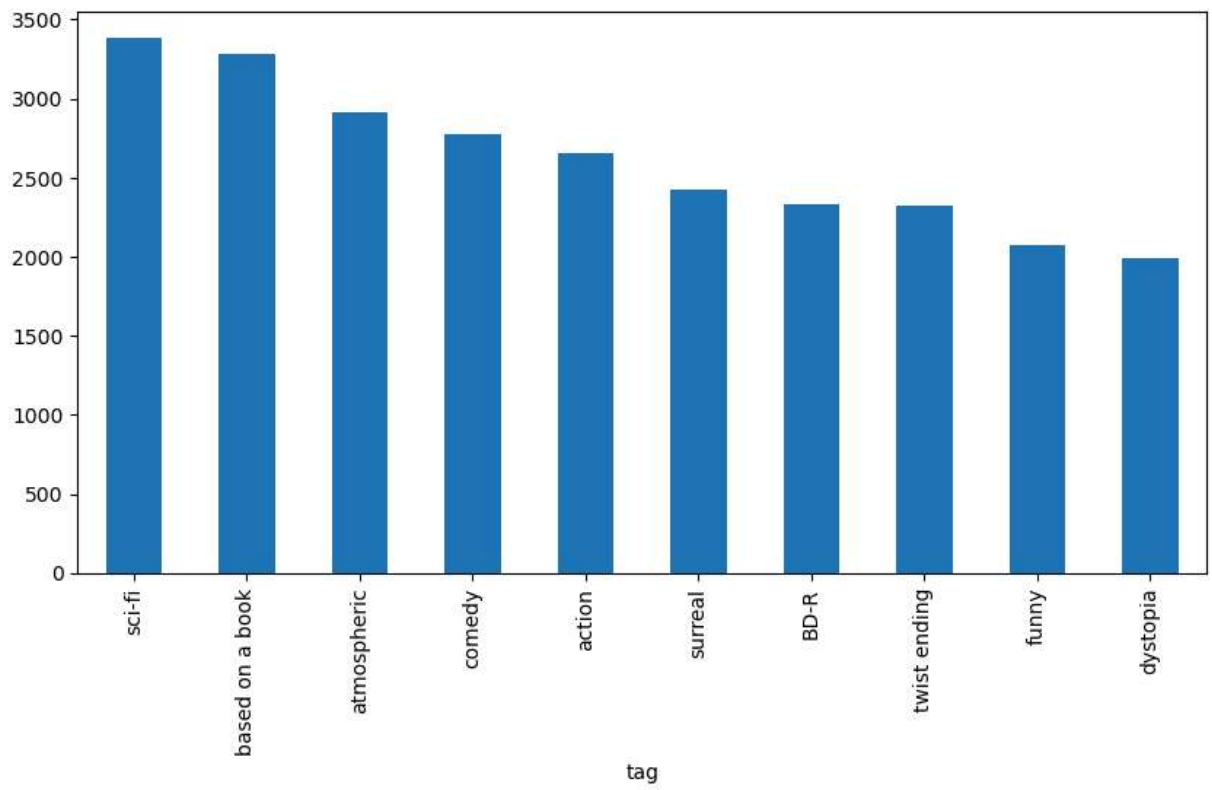
```
tag
missing child          1
Ron Moore              1
Citizen Kane          1
mullet                1
biker gang            1
Paul Adelstein        1
the wig               1
killer fish           1
genetically modified monsters  1
topless scene         1
Name: count, dtype: int64
```

In [123...

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

Out[123...

```
<Axes: xlabel='tag'>
```



In [ ]: