```
In [4]: import pandas as pd
```

Read the Dataset

```
In [9]: movies = pd.read_csv(r"C:\Users\Lenovo\Downloads\archive\movie.csv")
        print(type(movies))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
In [10]: print(movies)
```

```
       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)
...        ...                                 ...
27273   131254         Kein Bund für's Leben (2007)
27274   131256        Feuer, Eis & Dosenbier (2002)
27275   131258                   The Pirates (2014)
27276   131260                  Rentun Ruusu (2001)
27277   131262                    Innocence (2014)

                                            genres
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

[27278 rows x 3 columns]
```

```
In [11]: movies.head(20)
```

Out[11]:

	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
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Adventure|Animation|Children
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Crime|Drama
16	17	Sense and Sensibility (1995)	Drama|Romance
17	18	Four Rooms (1995)	Comedy
18	19	Ace Ventura: When Nature Calls (1995)	Comedy
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller

In [12]:

```
tags = pd.read_csv(r"C:\Users\Lenovo\Downloads\archive\tag.csv")
tags
```

Out[12]:

	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
...	...	...	...	...
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

465564 rows × 4 columns

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

Out[13]:

	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 [14]:
```
ratings = pd.read_csv(r"C:\Users\Lenovo\Downloads\archive\rating.csv")
ratings
```

Out[14]:

	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
...	...	...	...	...
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

20000263 rows × 4 columns

In [15]:
```
ratings.head()
```

Out[15]:

	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 [16]:
```
del ratings['timestamp']
del tags['timestamp']
```

In [17]:
```
ratings
```

Out[17]:

	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 [18]:
```
tags
```

Out[18]:

	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

Data Structures:

In [19]:
```
row_0 = tags.iloc[0]          # .iloc[] is used to access rows by index number (i
row_0
```

Out[19]:
```
userId               18
movieId            4141
tag         Mark Waters
Name: 0, dtype: object
```

In [20]:
```
type[row_0]
```

Out[20]:
```
type[userId               18
movieId            4141
tag         Mark Waters
Name: 0, dtype: object]
```

In [21]:
```
row_0.index     # .index gives you the list of all labels (i.e., column names) i
```

Out[21]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [22]:
```
row_0['userId']
```

Out[22]:
```
18
```

In [23]:
```
'rating' in row_0
```

Out[23]:
```
False
```

In [24]:
```
row_0.name
```

Out[24]:
```
0
```

In [25]:
```
row_0 = row_0.rename('firstRow')
row_0
```

Out[25]:
```
userId               18
movieId            4141
tag         Mark Waters
Name: firstRow, dtype: object
```

In [26]:
```
row_0.name
```

Out[26]:
```
'firstRow'
```

DataFrames

In [27]:
```
tags.head()
```

Out[27]:

	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 [28]:
```
# A Pandas DataFrame (2D table — rows and columns)
# The row labels (indexes) of the DataFrame.
tags.index
```

Out[28]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [29]:
```
tags.columns
```

Out[29]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [31]:
```
tags.iloc[[0,11,500]]
```

Out[31]:

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

📈 📉 Descriptive Statistics

In [33]:
```
ratings
```

Out[33]:

	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 [32]:
```
ratings['rating'].describe()
```

Out[32]:
```
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 [34]:
```
ratings.describe()
```

Out[34]:

	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 [35]:
```
ratings['rating'].mean()
```

Out[35]:
```
3.5255285642993797
```

In [36]:
```
ratings['rating'].min()
```

Out[36]:
```
0.5
```

In [37]:
```
ratings['rating'].std()
```

Out[37]:
```
1.051988919275684
```

In [38]:
```
ratings['rating'].mode()
```

Out[38]:
```
0    4.0
Name: rating, dtype: float64
```

In [39]:
```
ratings.corr()         # correlation tells the relation between two columns
```

Out[39]:

	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 [43]:
```
filter1 = ratings['rating']>10
filter1
```

```
Out[43]: 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 [44]: filter1.any()
```

```
Out[44]: False
```

```
In [45]: filter2 = ratings['rating']>0
         filter2
```

```
Out[45]: 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
```

```
In [46]: filter2.all()
```

```
Out[46]: True
```

🔧 Data Cleaning: Handling Missing Data

```
In [47]: movies
```

Out[47]:

	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 [48]: movies.shape        # (number_of_rows, number_of_columns)

Out[48]: (27278, 3)

In [49]: movies.isnull().any().any()

Out[49]: False

In [50]: ratings.shape

Out[50]: (20000263, 3)

In [52]: ratings.isnull().any().any()

Out[52]: False

In [53]: tags.shape

Out[53]: (465564, 3)

In [54]: tags.isnull().any().any()

Out[54]: True

In [55]: tags = tags.dropna()

In [56]: tags.isnull().any().any()

Out[56]: False
```

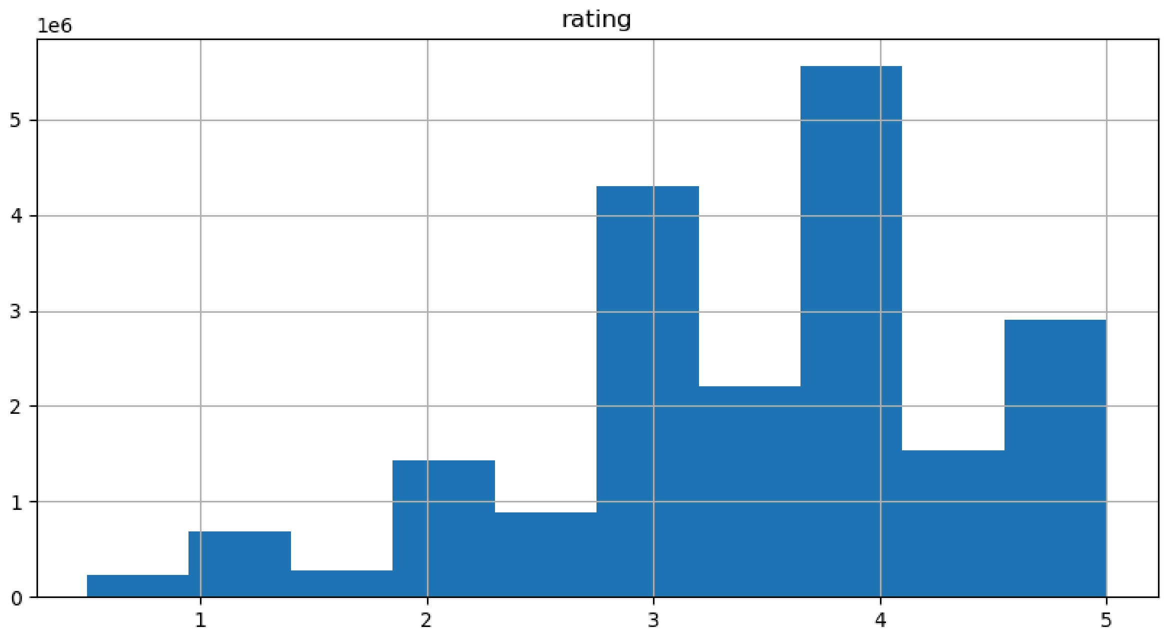
```
In [57]: tags.shape

Out[57]: (465548, 3)
```

📊 Data Visualization

```
In [66]: import matplotlib.pyplot as plt
         %matplotlib inline
```

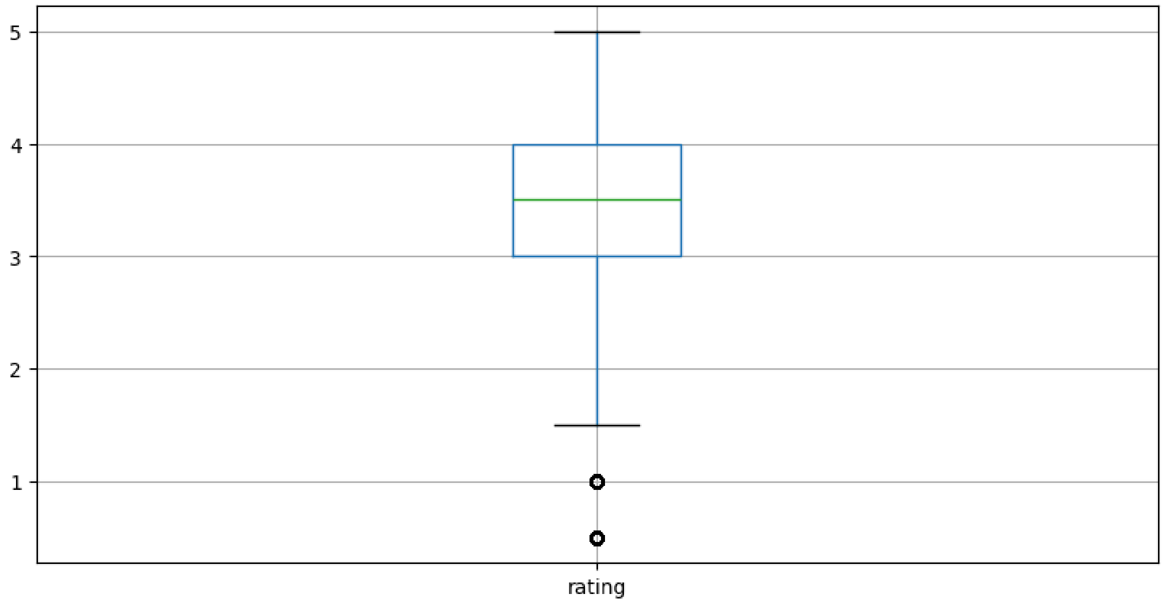
```
In [68]: ratings.hist(column='rating', figsize=(10,5))
         plt.show()
```



```
In [69]: ratings.boxplot(column='rating', figsize=(10,5))

Out[69]: <Axes: >
```

```
In [70]: plt.show()
```



📤 Slicing Out Columns

In [71]:
```
tags['tag'].head()
```

Out[71]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [74]:
```
movies[['title','genres']].head()
```

Out[74]:

	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 [75]:
```
ratings[-10:]
```

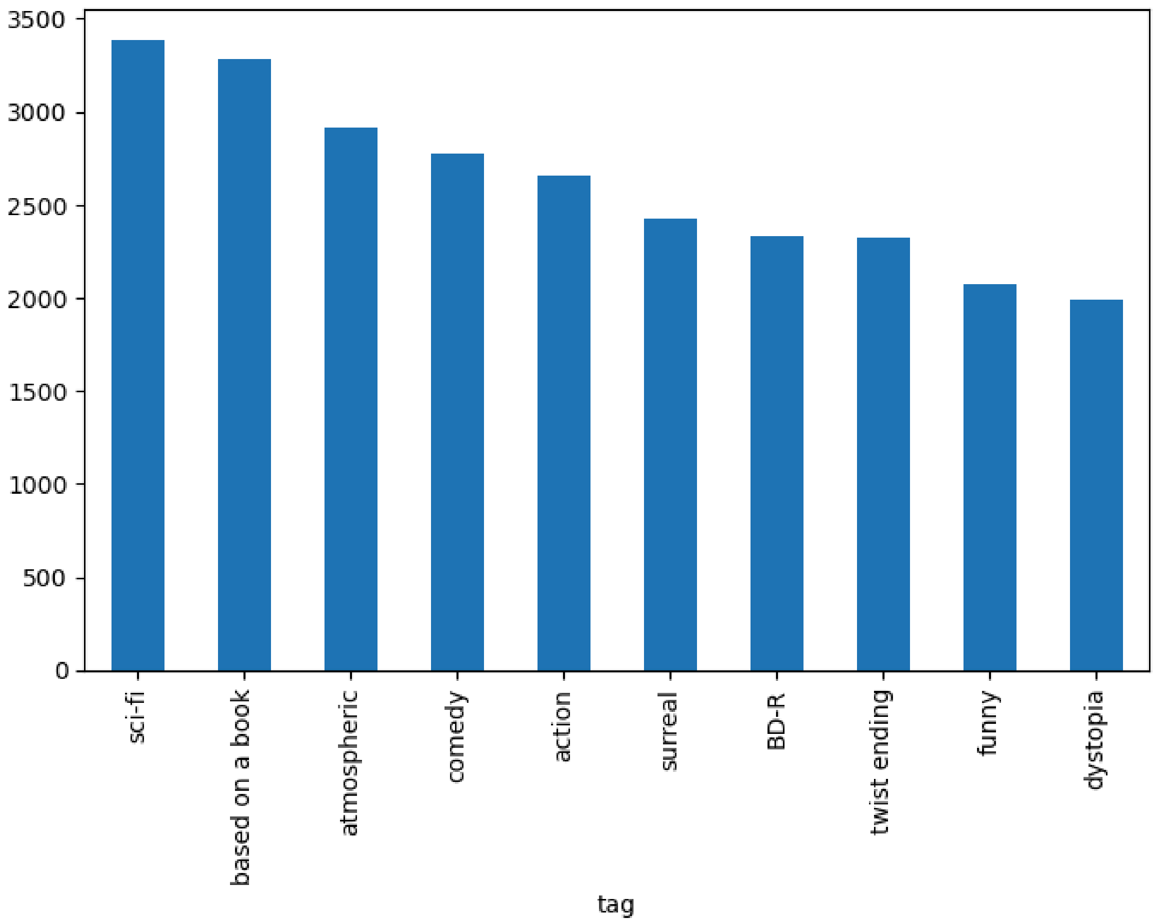
Out[75]:

	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 [80]:
```
tag_count = tags['tag'].value_counts()   #Counts how many times each unique tag
tag_count[-10:]                          # Shows the 10 rarest tags
```

Out[80]:
```
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 [82]:
```
tag_count[:10].plot(kind='bar', figsize=(8,5))    # top 10 most frequent tags us
plt.show()
```



In [ ]: