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

In [5]:
```
movies = pd.read_csv(r"D:\NIT Project\movie.csv")
```

In [7]:
```
print(type(movies))
movies.head
```

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

Out[7]:
```
<bound method NDFrame.head of        movieId                               titl
e  \
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 [9]:
```
tags = pd.read_csv(r"D:\NIT Project\tag.csv")
```

In [11]:
```
tags.head()
```

Out[11]:

	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]:
```
ratings = pd.read_csv(r"D:\NIT Project\rating.csv")
```

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']
```

Data Structures

Series

In [22]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[22]: pandas.core.series.Series

In [24]:
```
print(row_0)
```

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

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

Out[26]: Index(['userId', 'movieId', 'tag'], dtype='object')

In [28]:
```
row_0['userId']
```

Out[28]: 18

In [30]:
```
'rating' in row_0
```

Out[30]: False

In [32]:
```
row_0.name
```

Out[32]: 0

In [34]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[34]: 'firstRow'

Data Frames

In [37]:
```
tags.head()
```

Out[37]:

	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 [39]:
```
tags.index
```

Out[39]:
```
RangeIndex(start=0, stop=465564, step=1)
```

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

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

In [43]:
```
tags.iloc[[0,11,500]]
```

Out[43]:

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

Descritive Statistics

In [46]:
```
ratings['rating'].describe()
```

Out[46]:
```
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 [48]:
```
ratings.describe()
```

Out[48]:

	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 [50]:
```
ratings['rating'].mean()
```

Out[50]:
```
3.5255285642993797
```

In [52]:
```
ratings.mean()
```

Out[52]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [54]:
```
ratings['rating'].min()
```

Out[54]:
```
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
userId	1.000000	-0.000850	0.001175
movieId	-0.000850	1.000000	0.002606
rating	0.001175	0.002606	1.000000

In [64]:
```
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[64]: False

In [66]:
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[66]: True

Data Cleaning: Handling Missing Data

In [69]:
```
movies.shape
```

Out[69]: (27278, 3)

In [71]:
```
movies.isnull().any().any()
```

Out[71]: False

In [73]:
```
ratings.shape
```

Out[73]: (20000263, 3)

In [75]:
```
ratings.isnull().any().any()
```

Out[75]: False

In [77]:
```
tags.shape
```

Out[77]: (465564, 3)

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

Out[79]: True

In [81]:
```
tags=tags.dropna()
```

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

Out[83]: False

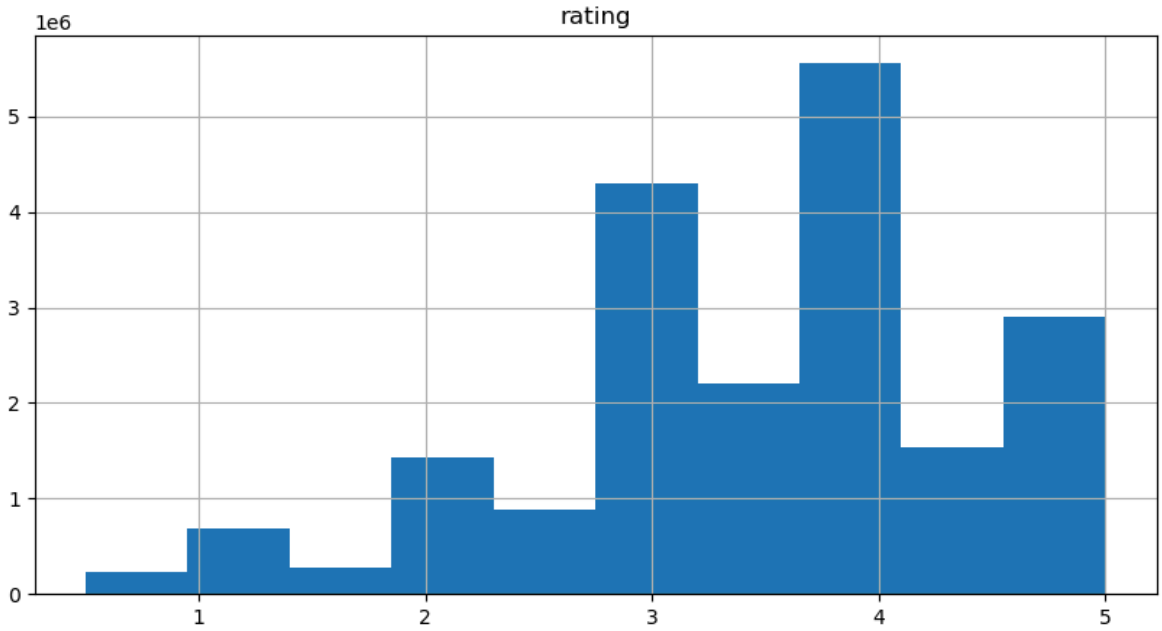
In [85]:
```
tags.shape
```

Out[85]: (465548, 3)

Data Visualization

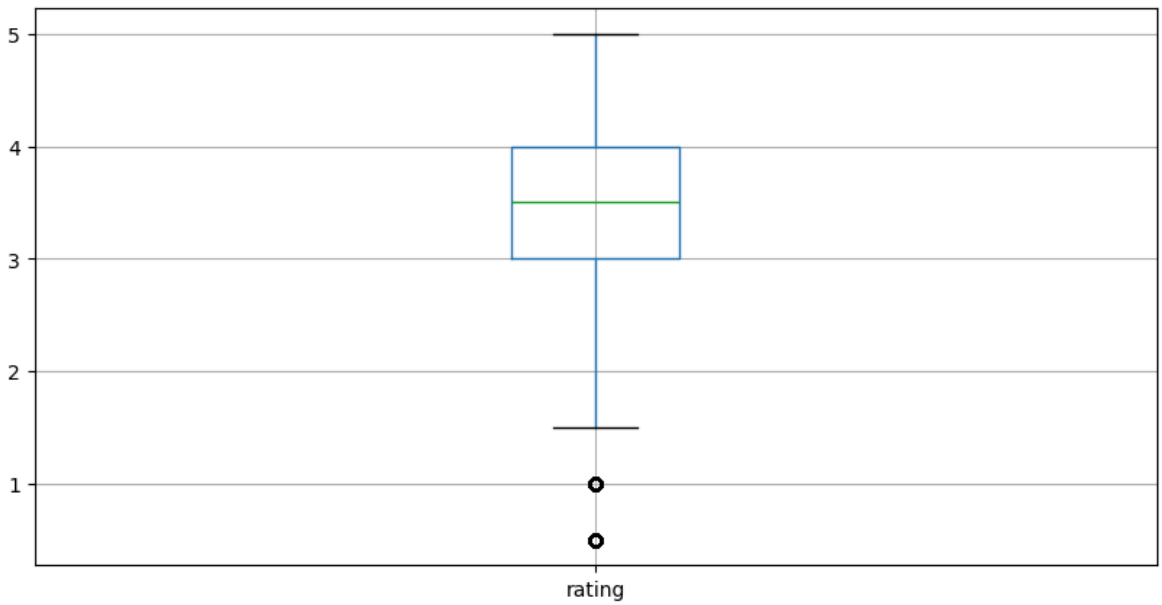
In [88]:
```
%matplotlib inline

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

Out[88]: array([[<Axes: title={'center': 'rating'}>]], dtype=object)



In [89]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[89]: <Axes: >



Slicing Out Columns

In [94]:
```
tags['tag'].head()
```

Out[94]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [96]:
```
movies[['title','genres']].head()
```

Out[96]:

	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 [98]:
```
ratings[-10:]
```

Out[98]:

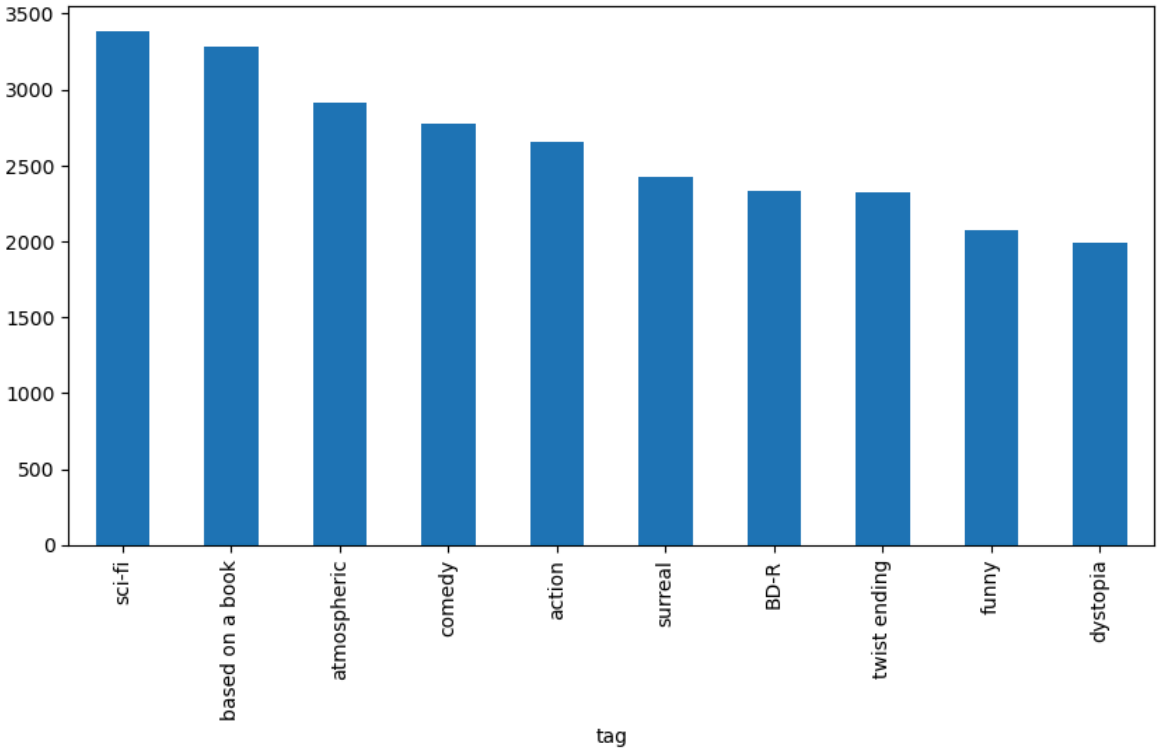
	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

In [100…
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[100…
```
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 [102…
```
tag_counts[:10].plot(kind='bar', figsize=(10,5))
```

Out[102… <Axes: xlabel='tag'>



In [ ]: