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

In [2]:
```
import numpy as np
```

In [3]:
```
movies = pd.read_csv(r'C:\Users\dell\Downloads\archive/movie.csv')
```

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

Out[4]:
```
(27278, 3)
```

In [5]:
```
print(type(movies))
```

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

In [6]:
```
movies.head(20)
```

Out[6]:

	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 [7]:
```
tags = pd.read_csv(r'C:\Users\dell\Downloads\archive/tag.csv')
```

In [8]:
```
tags.shape
```

Out[8]:
```
(465564, 4)
```

In [9]:
```
tags.head(20)
```

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
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
10	65	1694	jesus	2013-05-10 01:38:45
11	65	1783	noir thriller	2013-05-10 01:39:43
12	65	2022	jesus	2013-05-10 01:38:45
13	65	2193	dragon	2013-05-10 02:01:54
14	65	2353	conspiracy theory	2013-05-10 02:01:06
15	65	2662	mars	2013-05-10 01:40:55
16	65	2726	noir thriller	2013-05-10 01:39:43
17	65	2840	jesus	2013-05-10 01:38:45
18	65	3052	jesus	2013-05-10 01:38:46
19	65	5135	bollywood	2013-05-10 01:37:56

In [10]:
```
ratings = pd.read_csv(r'C:\Users\dell\Downloads\archive/rating.csv')
```

In [12]:
```
ratings.head()
```

Out[12]:

	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 [13]:
```
ratings.shape
```

Out[13]:
```
(20000263, 4)
```

In [17]:
```
del ratings['timestamp']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3802,
 in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File C:\ProgramData\anaconda3\lib\site-packages\pandas\_libs\index.pyx:138, in pan
das._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\lib\site-packages\pandas\_libs\index.pyx:165, in pan
das._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'timestamp'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[17], line 1
----> 1 del ratings['timestamp']

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\generic.py:4243, in ND
Frame.__delitem__(self, key)
   4238             deleted = True
   4239 if not deleted:
   4240     # If the above loop ran and didn't delete anything because
   4241     # there was no match, this call should raise the appropriate
   4242     # exception:
-> 4243     loc = self.axes[-1].get_loc(key)
   4244     self._mgr = self._mgr.idelete(loc)
   4246 # delete from the caches

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3804,
 in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'timestamp'
```

In [16]:
```
del movies['timestamp']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3802,
 in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File C:\ProgramData\anaconda3\lib\site-packages\pandas\_libs\index.pyx:138, in pan
das._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\lib\site-packages\pandas\_libs\index.pyx:165, in pan
das._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'timestamp'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[16], line 1
----> 1 del movies['timestamp']

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\generic.py:4243, in ND
Frame.__delitem__(self, key)
   4238             deleted = True
   4239 if not deleted:
   4240     # If the above loop ran and didn't delete anything because
   4241     # there was no match, this call should raise the appropriate
   4242     # exception:
-> 4243     loc = self.axes[-1].get_loc(key)
   4244     self._mgr = self._mgr.idelete(loc)
   4246 # delete from the caches

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3804,
 in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'timestamp'
```

In [19]:
```
ratings.head()
```

Out[19]:

	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

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

Out[20]:

	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 [21]:
```
del tags['timestamp']
```

In [22]:
```
tags.head()
```

Out[22]:

	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 [23]:
```
row_0 = tags.iloc[0]
type(row_0)
```

Out[23]:
```
pandas.core.series.Series
```

In [24]:
```
row_0
```

Out[24]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [25]:
```
row_0.index
```

Out[25]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [26]:
```
row_0['userId']
```

Out[26]:
```
18
```

In [29]:
```
row_0 in 'ratings'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 row_0 in 'ratings'

TypeError: 'in <string>' requires string as left operand, not Series
```

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

Out[30]:
```
False
```

In [31]:
```
row_0.name
```

Out[31]:
```
0
```

In [32]:
```
row_0 = row_0.rename('firstRow')
row_0.name
```

Out[32]:
```
'firstRow'
```

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 [34]:
```
tags.index
```

Out[34]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [36]:
```
tags.shape
```

Out[36]:
```
(465564, 3)
```

In [39]:
```
tags.columns
```

Out[39]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [45]:
```
tags.iloc[ [0,11,500] ]
```

Out[45]:

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

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 [47]:
```
ratings.describe()
```

Out[47]:

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

Out[48]:
```
3.5255285642993797
```

In [49]:
```
ratings.mean()
```

Out[49]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [50]:
```
ratings['rating'].min()
```

Out[50]:
```
0.5
```

In [51]:
```
ratings['rating'].max()
```

Out[51]:
```
5.0
```

In [52]:
```
ratings['rating'].std()
```

Out[52]:
```
1.051988919275684
```

In [53]:
```
ratings['rating'].mode()
```

Out[53]:
```
0    4.0
Name: rating, dtype: float64
```

In [54]:
```
ratings.corr()
```

Out[54]:

	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 [61]:
```
filter1 = ratings['rating'] > 100
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[61]:
```
False
```

In [62]:
```
filter2 = ratings['rating'] > 0
filter2.all()
```

Out[62]:
```
True
```

In [63]:
```
movies.shape
```

Out[63]:
```
(27278, 3)
```

In [64]:
```
movies.isnull().any().any()
```

Out[64]:
```
False
```

In [65]:
```
ratings.shape
```

Out[65]:
```
(20000263, 3)
```

In [66]:
```
ratings.isnull().any().any()
```

Out[66]:
```
False
```

In [67]:
```
tags.shape
```

Out[67]:
```
(465564, 3)
```

In [68]:
```
tags.isnull().any().any()
```

Out[68]:
```
True
```

In [69]:
```
tags=tags.dropna()
```

In [70]:
```
tags.isnull().any().any()
```

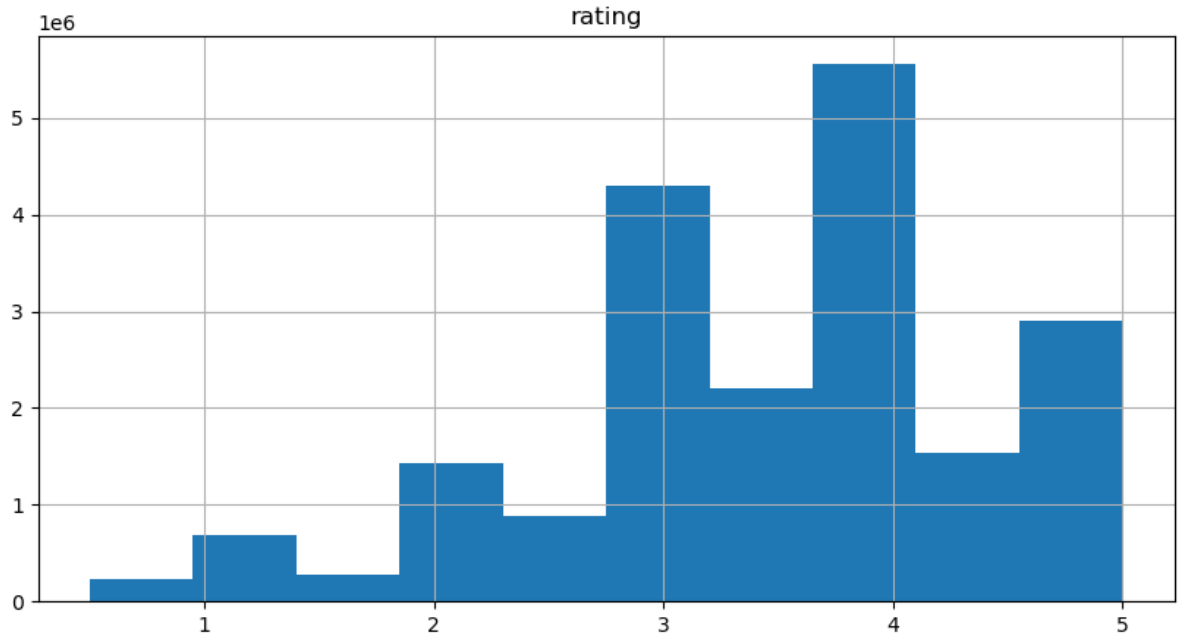
Out[70]:
```
False
```

In [71]:
```
tags.shape
```

Out[71]:
```
(465548, 3)
```

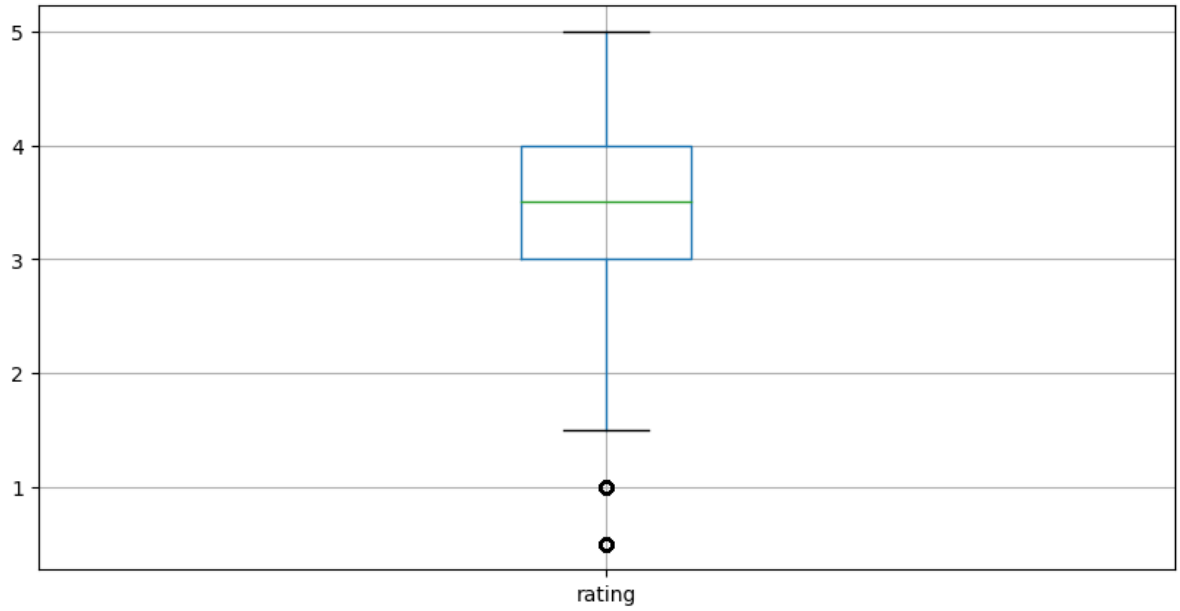
In [72]:
```
%matplotlib inline

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

Out[72]:
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



In [73]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[73]:
```
<Axes: >
```



In [74]:
```
tags['tag'].head()
```

Out[74]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [75]:
```
movies[['title','genres']].head()
```

Out[75]:

	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 [77]:
```
ratings['rating'][-10:]
```

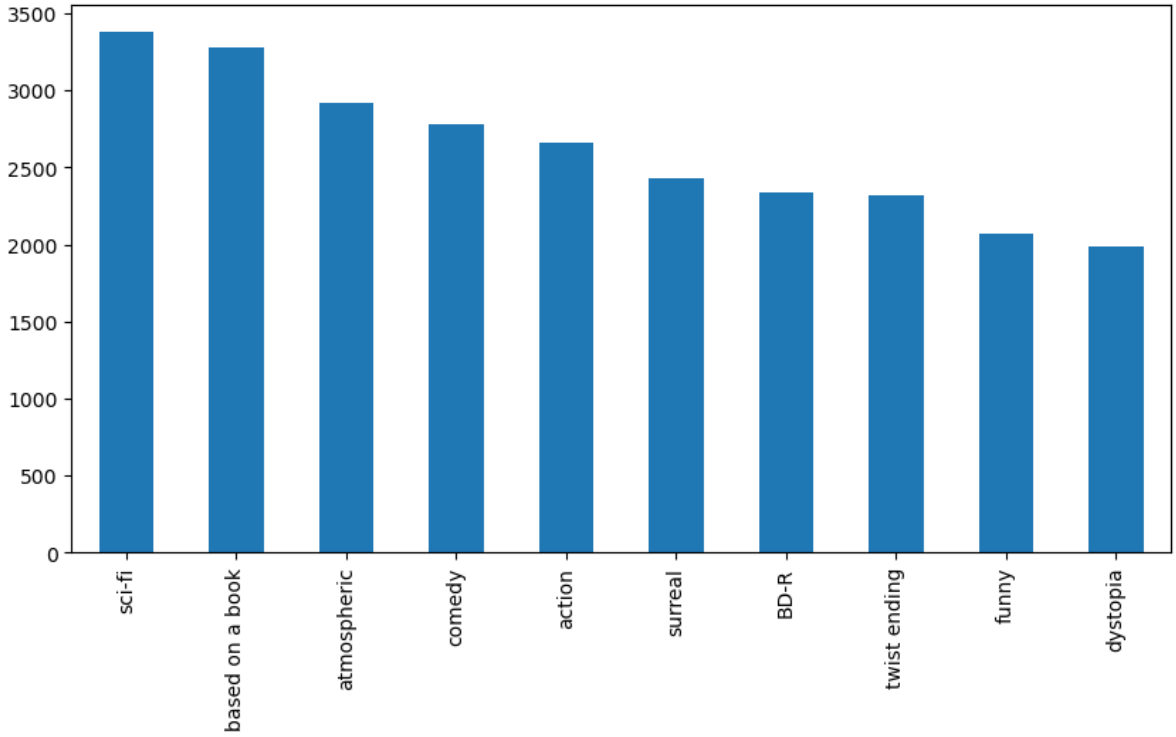
Out[77]:
```
20000253    4.5
20000254    4.0
20000255    4.5
20000256    4.5
20000257    4.5
20000258    4.5
20000259    4.5
20000260    3.0
20000261    5.0
20000262    2.5
Name: rating, dtype: float64
```

In [78]:
```
tag_counts = tags['tag'].value_counts()
tag_counts[-10:]
```

Out[78]:
```
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: tag, dtype: int64
```

In [79]:
```
tag_counts[:10].plot(kind='bar', figsize=(10,5))
```

Out[79]:
```
<Axes: >
```


3500
3000
2500
2000
1500
1000
500
0
sci-fi
based on a book
atmospheric
comedy
action
surreal
BD-R
twist ending
funny
dystopia

In [ ]: