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

In [2]:
```
movies = pd.read_csv(r'D:\Data Science with AI\Data Science With AI\14-july-arch
```

In [3]:
```
movies.head()
```

Out[3]:

	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 [4]:
```
movies.head(2)
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy

In [5]:
```
ratings=pd.read_csv(r'D:\Data Science with AI\Data Science With AI\14-july-archi
```

In [6]:
```
ratings
```

Out[6]:

	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 [7]:
```
ratings.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24696\2593860280.py in ?()
----> 1 ratings.show()

~\AppData\Roaming\Python\Python312\site-packages\pandas\core\generic.py in ?(self, name)
   6314             and name not in self._accessors
   6315             and self._info_axis._can_hold_identifiers_and_holds_name(name)
   6316         ):
   6317             return self[name]
-> 6318         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'show'
```

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

Out[8]: (20000263, 4)

In [9]:
```
movies.shape
```

Out[9]: (27278, 3)

In [10]:
```
print(type(movies))
print(type(ratings))
print(type(tags))
```

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 3
      1 print(type(movies))
      2 print(type(ratings))
----> 3 print(type(tags))

NameError: name 'tags' is not defined
```

In [11]:
```
tags=pd.read_csv(r"D:\Data Science with AI\Data Science With AI\14-july-archive\
```

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

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

In [13]:
```
tags
```

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
...	...	...	...	...
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 [14]:
```
print(type(tags))
```

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

In [15]:
```
tags.shape
```

Out[15]:
```
(465564, 4)
```

In [16]:
```
tags.columns
```

Out[16]:
```
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
```

In [17]:
```
ratings.columns
```

Out[17]:
```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

In [19]:
```
movies.columns
```

Out[19]:
```
Index(['movieId', 'title', 'genres'], dtype='object')
```

In [20]:
```
del ratings['timestamp']
del tags['timestamp']
```

In [21]:
```
ratings.columns
```

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

In [22]:
```
tags.columns
```

Out[22]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [23]:
```
tags.head()
```

Out[23]:

	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 [24]:
```
tags.iloc[0]
```

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

In [25]:
```
tags.iloc[2]
```

Out[25]:
```
userId            65
movieId          353
tag        dark hero
Name: 2, dtype: object
```

In [26]:
```
row_0=tags.iloc[0]
```

In [27]:
```
row_0
```

Out[27]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [28]:
```
print(row_0)
```

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

In [29]:
```
type(row_0)
```

Out[29]:
```
pandas.core.series.Series
```

In [30]:
```
row_0=index
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 row_0=index

NameError: name 'index' is not defined
```

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

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

In [32]:
```
row_0
```

Out[32]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

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.columns
```

Out[34]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [35]:
```
row_0.index
```

Out[35]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [36]:
```
row_0[1]
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_24696\1082734514.py:1: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future version,
integer keys will always be treated as labels (consistent with DataFrame behavio
r). To access a value by position, use `ser.iloc[pos]`
  row_0[1]
```

Out[36]:
```
4141
```

In [37]:
```
row_0.head()
```

Out[37]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [38]:
```
tags.head()
```

Out[38]:

	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]:
```
row_0[1]
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_24696\1082734514.py:1: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future version,
integer keys will always be treated as labels (consistent with DataFrame behavio
r). To access a value by position, use `ser.iloc[pos]`
  row_0[1]
```

Out[39]:
```
4141
```

In [40]:
```
row_0[2]
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_24696\1293913282.py:1: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future version,
integer keys will always be treated as labels (consistent with DataFrame behavio
r). To access a value by position, use `ser.iloc[pos]`
  row_0[2]
```

Out[40]:
```
'Mark Waters'
```

In [42]:
```
row_0['userId']
```

Out[42]:
```
18
```

In [43]:
```
'rating' in row_0
```

Out[43]:
```
False
```

In [44]:
```
row_0.name
```

Out[44]:
```
0
```

In [45]:
```
row_0=row_0.rename('firstRow')
row_0.name
```

Out[45]:
```
'firstRow'
```

In [46]:
```
tags.head()
```

Out[46]:

	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 [47]:
```
tags.head()
```

Out[47]:

	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 [48]:
```
tags.index
```

Out[48]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [49]:
```
tags.columns
```

Out[49]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [50]:
```
tags.iloc[[0,11,500]]
```

Out[50]:

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

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

Out[51]:
```
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 [52]:
```
ratings.describe()
```

Out[52]:

	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 [53]:
```
ratings.mean()
```

Out[53]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

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

Out[54]:
```
3.5255285642993797
```

In [55]:
```
ratings.mean()
```

Out[55]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [56]:
```
ratings['rating'].min()
```

Out[56]:
```
0.5
```

In [57]:
```
ratings.min()
```

Out[57]:
```
userId     1.0
movieId    1.0
rating     0.5
dtype: float64
```

In [58]:
```
ratings['rating'].max()
```

Out[58]:
```
5.0
```

In [59]:
```
ratings.max()
```

Out[59]:
```
userId     138493.0
movieId    131262.0
rating          5.0
dtype: float64
```

In [60]:
```
ratings['rating'].std()
```

Out[60]: 1.051988919275684

In [61]:
```
ratings.std()
```

Out[61]:
```
userId     40038.626653
movieId    19789.477445
rating         1.051989
dtype: float64
```

In [62]:
```
ratings['rating'].mode()
```

Out[62]:
```
0    4.0
Name: rating, dtype: float64
```

In [63]:
```
rating.mode()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[63], line 1
----> 1 rating.mode()

NameError: name 'rating' is not defined
```

In [64]:
```
ratings.mode()
```

Out[64]:

	userId	movieId	rating
0	118205	296	4.0

In [65]:
```
ratings.corr()
```

Out[65]:

	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 [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]:
```
filter8=ratings['rating']>1
print(filter8)
filter8.any()
```

```
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
```

Out[70]: True

In [72]:
```
filter2=ratings['rating']>0
filter2.all()
```

Out[72]: True

Data Cleaning : Handling Missing Data

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

Out[73]: (27278, 3)

In [74]:
```
movies.index
```

Out[74]: RangeIndex(start=0, stop=27278, step=1)

In [75]:
```
movies.columns
```

Out[75]: Index(['movieId', 'title', 'genres'], dtype='object')

In [76]:
```
movies[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3811 try:
-> 3812     return self._engine.get_loc(casted_key)
   3813 except KeyError as err:

File pandas/_libs/index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7096, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 0

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

KeyError                                  Traceback (most recent call last)
Cell In[76], line 1
----> 1 movies[0]

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:4107, in DataFrame.__getitem__(self, key)
   4105 if self.columns.nlevels > 1:
   4106     return self._getitem_multilevel(key)
-> 4107 indexer = self.columns.get_loc(key)
   4108 if is_integer(indexer):
   4109     indexer = [indexer]

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\indexes\base.py:3819, in Index.get_loc(self, key)
   3814     if isinstance(casted_key, slice) or (
   3815         isinstance(casted_key, abc.Iterable)
   3816         and any(isinstance(x, slice) for x in casted_key)
   3817     ):
   3818         raise InvalidIndexError(key)
-> 3819     raise KeyError(key) from err
   3820 except TypeError:
   3821     # If we have a listlike key, _check_indexing_error will raise
   3822     #  InvalidIndexError. Otherwise we fall through and re-raise
   3823     #  the TypeError.
   3824     self._check_indexing_error(key)

KeyError: 0
```

In [77]:
```
movies['movieId'][0]
```

Out[77]: 1

In [78]:
```
movies['title'][0]
```

Out[78]: 'Toy Story (1995)'

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

Out[79]: False

In [80]:
```
ratings.shape
```

Out[80]: (20000263, 3)

In [81]:
```
ratings.isnull().any()
```

Out[81]:
```
userId     False
movieId    False
rating     False
dtype: bool
```

In [82]:
```
ratings.isnull().any().any()
```

Out[82]: False

In [83]:
```
tags.shape
```

Out[83]: (465564, 3)

In [84]:
```
tags.isnull().any().any()
```

Out[84]: True

In [85]:
```
tags=tags.dropna()
```

In [86]:
```
tags.isnull().any().any()
```

Out[86]: False

In [87]:
```
tags.shape
```

Out[87]: (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]:
```
plt.show()
```

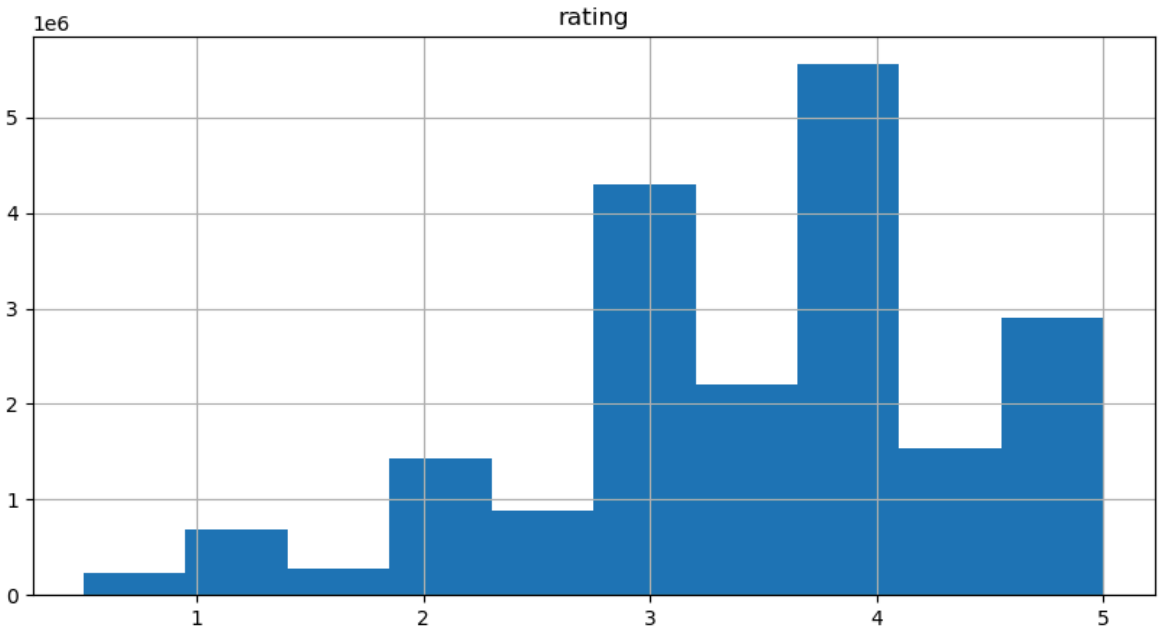
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[89], line 1
----> 1 plt.show()

NameError: name 'plt' is not defined
```

In [128…
```
import matplotlib.pyplot as plt
%matplotlib inline
ratings.hist(column='rating',figsize=(10,5))
```

Out[128… array([[<Axes: title={'center': 'rating'}>]], dtype=object)

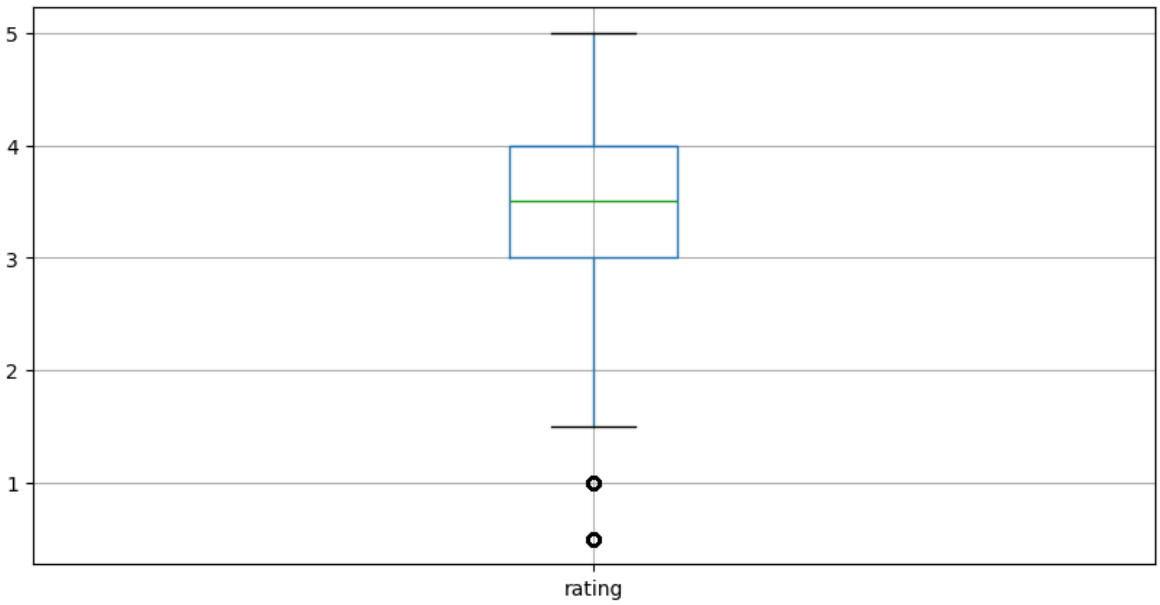
In [129…
```
plt.show()
```



In [130…
```
ratings.boxplot(column='rating',figsize=(10,5))
```

Out[130… <Axes: >

In [131…
```
plt.show()
```



Slicing Out Columns

In [126…
```
tags['tag'].head()
```

Out[126…
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [132…
```
movies[['title','genres']].head()
```

Out[132…

	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 [133…
```
ratings[-10:]
```

Out[133…

	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

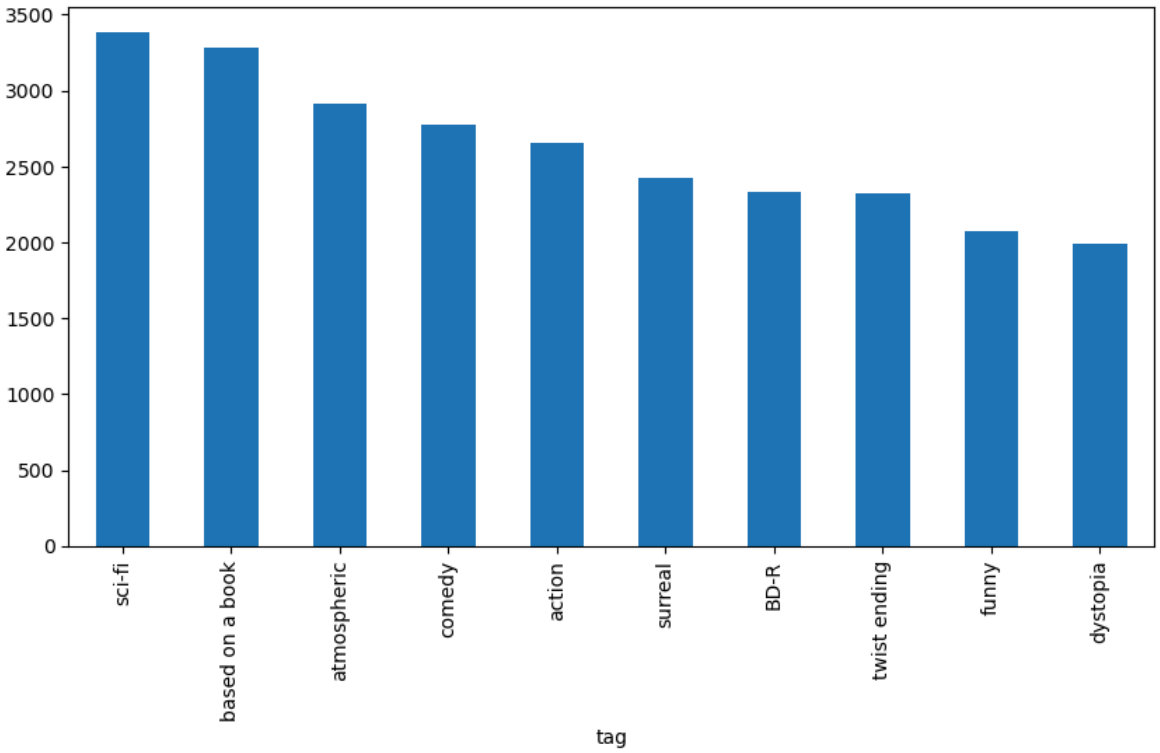
In [134…
```
tag_counts=tags['tag'].value_counts()
tag_counts[-10:]
```

Out[134…
```
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 [138…
```
tag_counts[:10].plot(kind='bar',figsize=(10,5))
```

Out[138…
```
<Axes: xlabel='tag'>
```

In [139…
```
plt.show()
```



In [ ]: