Pandas With Data SCience.AI

- P - 5

1 import Pandas

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

read CSV

In [6]:
```
ratings = pd.read_csv(r"E:\One_Drive(Microsoft)\OneDrive\Data_Science_cource\Module_1_Python_29_July\D20_28Aug_work_s
tags = pd.read_csv(r"E:\One_Drive(Microsoft)\OneDrive\Data_Science_cource\Module_1_Python_29_July\D20_28Aug_work_shop
movies = pd.read_csv(r"E:\One_Drive(Microsoft)\OneDrive\Data_Science_cource\Module_1_Python_29_July\D20_28Aug_work_sh

# printing shapes
print(ratings.shape)
print(tags.shape)
print(movies.shape)
```

```
(20000263, 4)
(465564, 4)
(27278, 3)
```

In [10]:
```
# printing types
print(" ratings type = ",type(ratings))
print(" movies type = ",type(movies))
print(" ratings type = ",type(tags))
```

```
 ratings type =  <class 'pandas.core.frame.DataFrame'>
 movies type =  <class 'pandas.core.frame.DataFrame'>
 ratings type =  <class 'pandas.core.frame.DataFrame'>
```

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

ratings.head(10)
```

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
5	1	112	3.5	2004-09-10 03:09:00
6	1	151	4.0	2004-09-10 03:08:54
7	1	223	4.0	2005-04-02 23:46:13
8	1	253	4.0	2005-04-02 23:35:40
9	1	260	4.0	2005-04-02 23:33:46

In [14]:
```
movies.head(7)
```

Out[14]:

	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

In [16]:
```
tags.head(5)
```

Out[16]:

	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

del

In [20]:
```
print(ratings.columns)
print(tags.columns)

# del
del ratings['timestamp']
del tags['timestamp']

# after del

print(ratings.columns)
print(tags.columns)
```

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='object')
Index(['userId', 'movieId', 'rating'], dtype='object')
Index(['userId', 'movieId', 'tag'], dtype='object')
```

Data structre

In [25]:
```
# series
# accessing 0th row from tags data frame
row_0 = tags.iloc[0]
print(type(row_0)) # row_0 type
```

```
<class 'pandas.core.series.Series'>
```

```
In [27]: #  priting row_0 values

         print(row_0)
```

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

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

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

```
In [34]: row_0['tag']
```

```
Out[34]: 'Mark Waters'
```

```
In [38]: row_0['userId']
```

```
Out[38]: 18
```

```
In [46]: row_0['movieId']
```

```
Out[46]: 4141
```

```
In [40]: 'rating' in row_0
```

```
Out[40]: False
```

```
In [52]: print("current name of row_0  = " ,row_0.name)
```

```
current name of row_0  =  0
```

```
In [62]: # renameing to first row
         row_0 = row_0.rename('Firstrow')
         row_0.index
         print("new name row_0 = " ,row_0.name)
```

```
new name row_0 =  Firstrow
```

```
In [56]: print(row_0)
```

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

Data Frames

In [65]:
```
tags.index
```

Out[65]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [69]:
```
tags.head()
```

Out[69]:

	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 [71]:
```
tags.head
```

Out[71]:
```
<bound method NDFrame.head of         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 x 3 columns]>
```

In [73]:
```python
tags.columns
```

Out[73]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [75]:
```python
tags.index
```

Out[75]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [82]:
```python
tags.iloc[[0,11,500]] # iloc => known as integer location
# selcetive [[ r1,r2,r3]]rows
# Here we select only three rows from the tags data frame
# Here in this code, how we can access/select multiple row index
```

Out[82]:

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

In [88]:
```python
tags.iloc[[5,55,400,300,2000,60000]]
```

Out[88]:

	userId	movieId	tag
5	65	668	bollywood
55	121	1288	Christopher Guest
400	342	4848	pretentious
300	316	45186	Ethan Hunt Should Stop Hogging The Screen!
2000	910	68554	conspiracy theory
60000	12792	7669	England

Descriptive Statistics¶

In [114…

```
ratings.columns
```

Out[114…

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

In [102…

```
ratings['rating'].describe()
print(ratings['rating'].describe())
```

```
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 [112…

```
ratings.columns
```

Out[112…

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

In [116…

```
ratings.describe()
```

Out[116…

	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 [118…
```
ratings['rating'].mean() # only for one column
```
Out[118…
```
3.5255285642993797
```
In [120…
```
ratings.mean() # for all data varables
```
Out[120…
```
userId      69045.872583
movieId      9041.567330
rating          3.525529
dtype: float64
```
In [122…
```
ratings['rating'].min() # minimum from rating value from rating variable
```
Out[122…
```
0.5
```
In [124…
```
ratings['rating'].max() # maximum from rating data frame of rating variacle
# operation for only one variable
```
Out[124…
```
5.0
```
In [126…
```
# operation for all variables
ratings.max()
```

Out[126…
```
userId     138493.0
movieId    131262.0
rating          5.0
dtype: float64
```

In [128…
```
ratings.min()
```

Out[128…
```
userId     1.0
movieId    1.0
rating     0.5
dtype: float64
```

In [130…
```
ratings['rating'].mode()
```

Out[130…
```
0    4.0
Name: rating, dtype: float64
```

In [136…
```
ratings['rating'].std()
```

Out[136…
```
1.051988919275684
```

In [138…
```
ratings.std()
```

Out[138…
```
userId     40038.626653
movieId    19789.477445
rating         1.051989
dtype: float64
```

In [140…
```
ratings.mode()
```

Out[140…

	userId	movieId	rating
0	118205	296	4.0

In [142…
```
ratings.corr()
```

Out[142…

	userId	movieId	rating
userId	1.000000	-0.000850	0.001175
movieId	-0.000850	1.000000	0.002606
rating	0.001175	0.002606	1.000000

filters

In [159…

```
filter1 = ratings['rating'] > 10
print(filter1)
print(filter1.any())
filter1.any() # any() methos atleast if one element is True or meet certine condition then  it returns True

# DataFrame Usage: Checks if any value in each column or row is True.
# Series Usage: Checks if any value in the Series is True.
```

```
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
False
```

Out[159…

```
False
```

In [169…

```
filter2 = ratings['rating'] > 0
print(filter2)
print("all() => ",filter2.all())
```

```
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
all() =>  True
```

Data Cleaning: Handling missing values/data

In [178…
```
print("shape of data frome movies => ",movies.shape)
```

```
shape of data frome movies =>  (27278, 3)
```

In [186…
```
# movies.isnull()
# movies.isna()
```

In [184…
```
movies.isnull().any().any()
# no missing values
```

Out[184…
```
False
```

In [208…
```
print("shape of rating data frame ")
ratings.shape
```

```
shape of rating data frame
```

Out[208…
```
(20000263, 3)
```

In [192…
```
# ratings.isnull()
# ratings.isna()
```

In [206…
```
# print("is any null values in rating data frme => ",ratings.isnull().any().any())
ratings.isnull().any().any()
```

Out[206…
```
False
```

In [210…
```
print("shape of tag data frame ")
tags.shape
```
```
shape of tag data frame
```
Out[210…
```
(465564, 3)
```

In [212…
```
# cheching is data is clean or row
tags.isnull().any().any()
tags.isna().any().any()

#  the the tag data is not clean
```
Out[212…
```
True
```

we have some tags which are null

In [216…
```
# to drop null tags of cells from the data set we have
# => dropna() method
tags = tags.dropna() # dropping and reassigning the data set
```

In [220…
```
# Now we are checking the tags data set
print("is any null values in tags data set ")
tags.isnull().any().any()
```
```
is any null values in tags data set
```
Out[220…
```
False
```

In [222…
```
tags.shape
# some rows are reduced
```
Out[222…
```
(465548, 3)
```

Data visualization

In [234…
```
ratings.columns
```

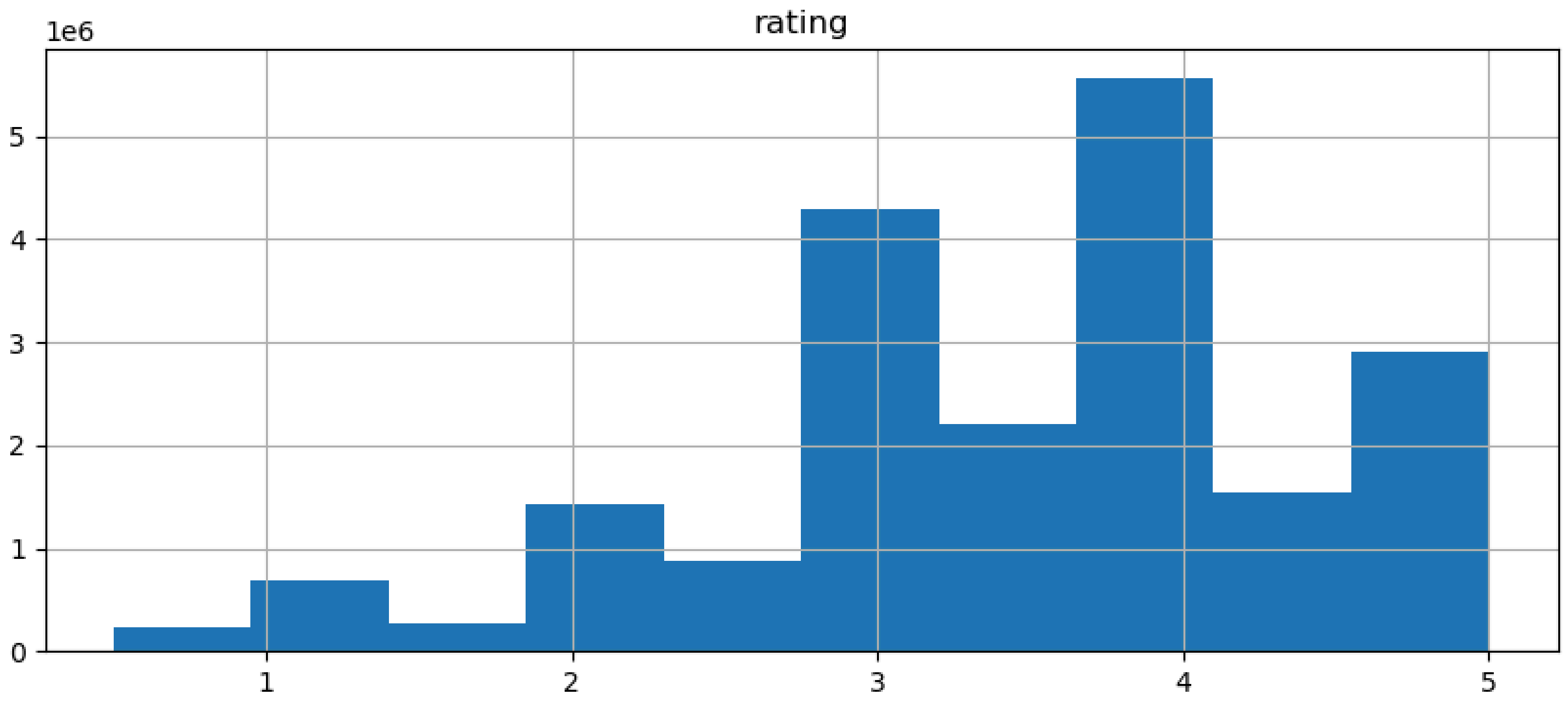
```
Out[234… Index(['userId', 'movieId', 'rating'], dtype='object')
```

```
In [256… # %matplotlib inline it is magical command
%matplotlib inline

ratings.hist(column = 'rating', figsize = (10,4),grid = True

# Syntax
# DataFrame.hist(column=None, by=None, grid=True, xlabelsize=None, xrot=None, ylabelsize=None, yrot=None, ax=None, sh
```

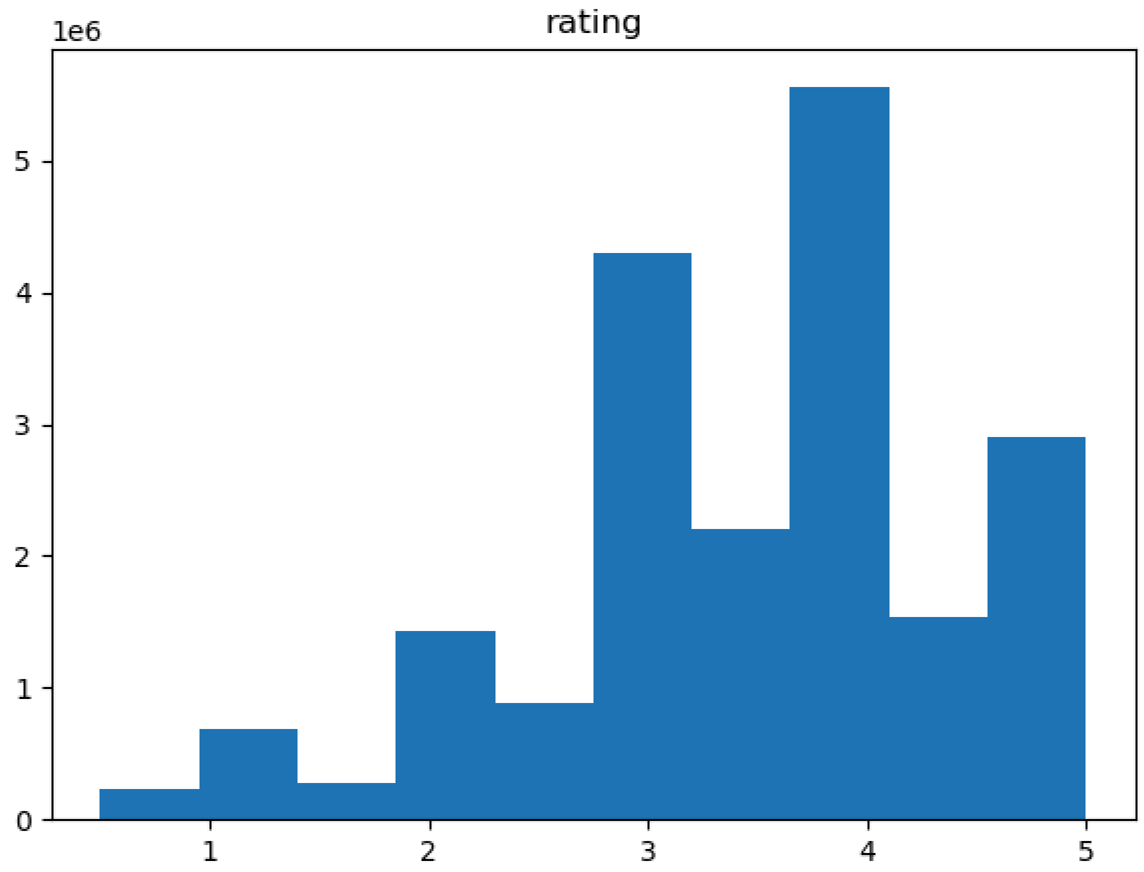
```
Out[256… array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



```
In [254… # import matplotlib.pyplot as plt
# plt.plot()
# plt.xlabel("X - axis")
# plt.ylabel("Y - axis")
# plt.title("The graph")
```

In [282…
```
# %matplotlib inline
ratings.hist(column = 'rating',grid = False, figsize = (7,5))
```

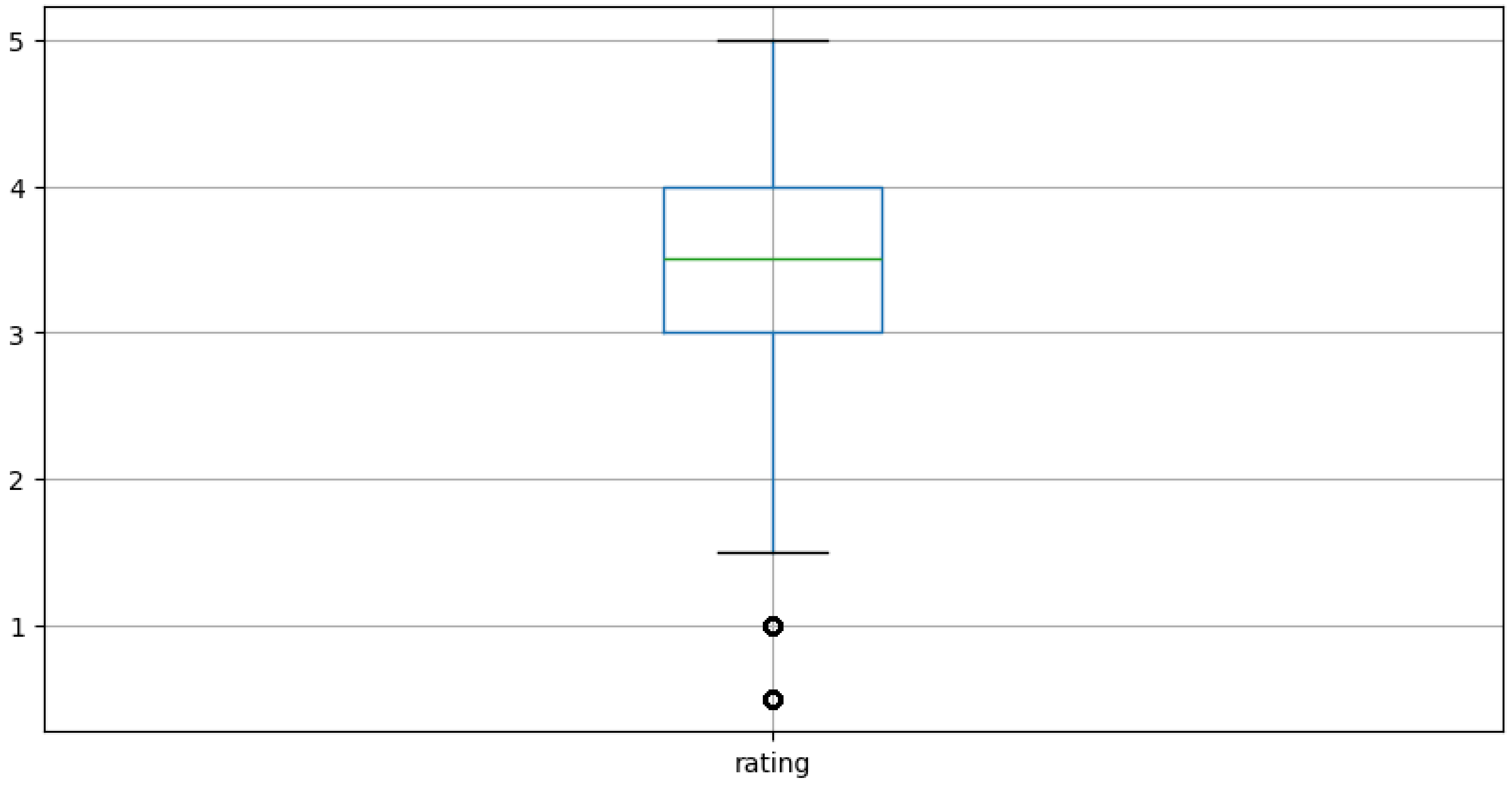
Out[282…
```
array([[<Axes: title={'center': 'rating'}>]], dtype=object)
```



In [289…
```
# ratings.boxplot(column = 'rating',figsize = (5,5))
# plt.show()
```

In [291…
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[291…
```
<Axes: >
```



Slicing Out Columns

In [295…
```
tags['tag'].head()
```

Out[295…
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [298…
```
movies[['title','genres']].head() # accessing two variables and first 5 rows
```

Out[298…

	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 [300…

```
ratings[-10:] # negative indexing
```

Out[300…

	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 [302…

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

Out[302…

```
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
```

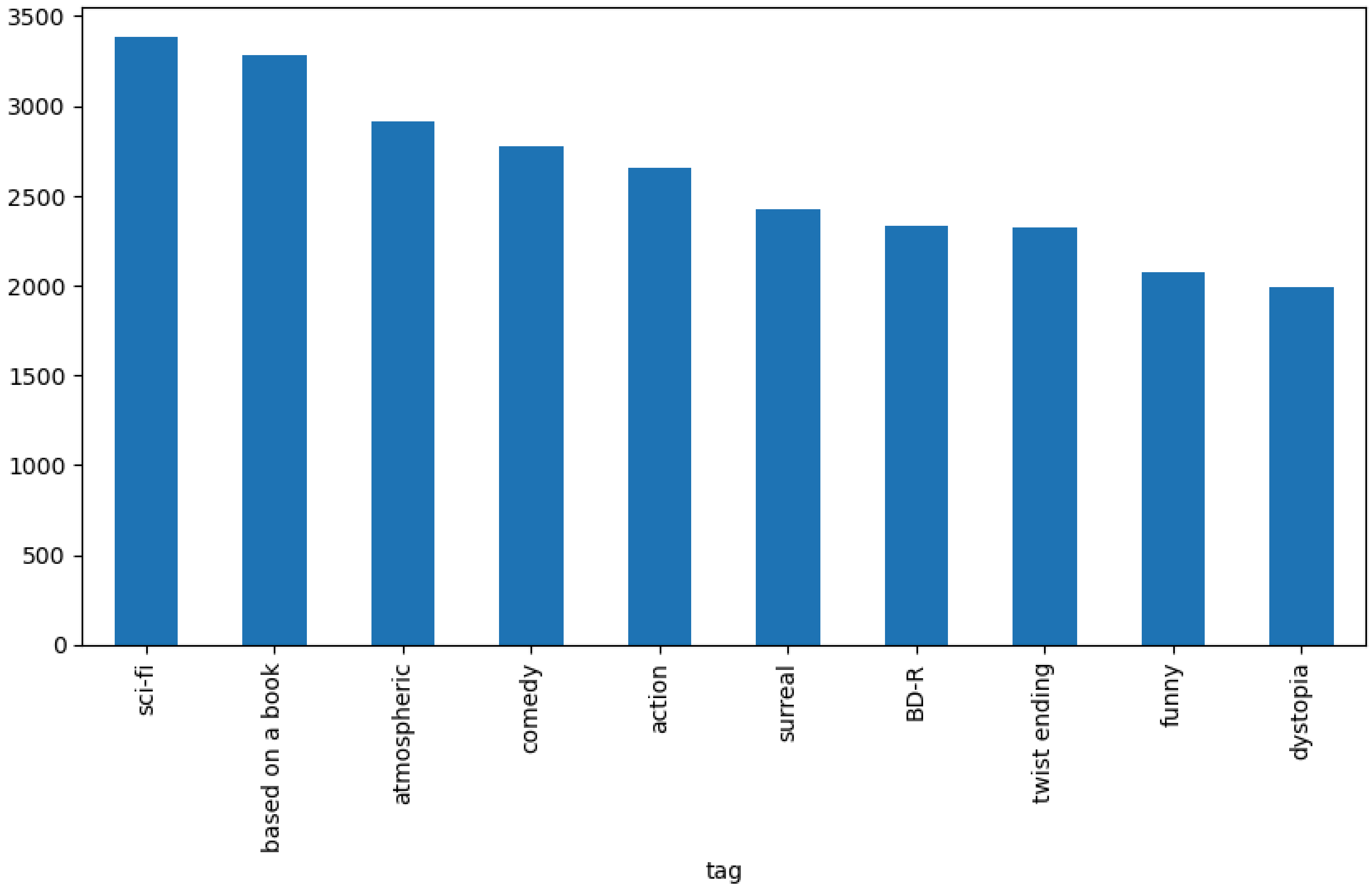
In [ ]:

In [304…

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

Out[304…

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



In [308…

```
#
```

In [ ]:

```
# https://www.kaggle.com/code/harunshimanto/pandas-with-data-science-ai
```