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

In [9]:
```
movies= pd.read_csv(r"C:\Users\Appala nithin\OneDrive\Pictures\Documents\NARESH-IT\
```

In [27]:
```
ratings= pd.read_csv(r"C:\Users\Appala nithin\OneDrive\Pictures\Documents\NARESH-IT
```

In [34]:
```
tags= pd.read_csv(r"C:\Users\Appala nithin\OneDrive\Pictures\Documents\NARESH-IT\ne
```

In [19]:
```
print(type(movies))
movies.head(20)
```

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

Out[19]:

	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 [36]:

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

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

In [38]:

```
tags.head()
```

Out[38]:

	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 [29]:

```
ratings.head()
```

Out[29]:

	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 [52]:

```
if 'timestamp' in ratings.columns:
    del ratings['timestamp']

if 'timestamp' in tags.columns:
    del tags['timestamp']
```

DATA STRUCTURES

In [55]:

```
# SERIES
row_0=tags.iloc[0]
type(row_0)
```

Out[55]:

```
pandas.core.series.Series
```

In [57]:

```
print(row_0)
```

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

In [59]:

```
row_0.index
```

Out[59]:

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

In [61]:

```
row_0['userId']
```

Out[61]:

```
18
```

In [65]:

```
'rating' in row_0
```

Out[65]:

```
False
```

In [67]:

```
row_0.name
```

Out[67]:

```
0
```

In [71]:

```
row_0=row_0.rename('firstRow')
row_0.name
```

Out[71]:

```
'firstRow'
```

DATA FRAMES

In [75]:

```
tags.head()
```

Out[75]:

	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 [77]:

```
tags.index
```

Out[77]:

```
RangeIndex(start=0, stop=465564, step=1)
```

In [79]:

```
tags.columns
```

Out[79]:

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

In [89]:

```
tags.iloc[[0,11,500]]
```

Out[89]:

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

DESCRIPTIVE STATISTICS

In [92]:

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

Out[92]:

```
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 [94]:

```
ratings.describe()
```

Out[94]:

	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 [96]:

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

Out[96]:

```
3.5255285642993797
```

In [98]:

```
ratings.mean()
```

Out[98]:

```
userId      69045.872583
movieId      9041.567330
rating          3.525529
dtype: float64
```

In [100…

```
ratings['rating'].min()
```

Out[100…

```
0.5
```

In [102…

```
ratings['rating'].max()
```

Out[102…

```
5.0
```

In [104…

```
ratings['rating'].std()
```

Out[104…

```
1.051988919275684
```

In [110…

```
ratings['rating'].mode()
```

Out[110…

```
0    4.0
Name: rating, dtype: float64
```

In [114…

```
ratings.corr()
```

Out[114…

	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 [116…

```
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[116…

```
False
```

In [118…

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

Out[118…

```
True
```

DATA-CLEANING : HANDLING MISSING DATA

In [121…

```
movies.shape
```

Out[121…

```
(27278, 3)
```

In [123…

```
movies.isnull().any().any()
```

Out[123…

```
False
```

In [125…

```
ratings.shape
```

Out[125…

```
(20000263, 3)
```

In [127…

```
ratings.isnull().any().any()
```

Out[127…

```
False
```

In [129…

```
tags.shape
```

Out[129…

```
(465564, 3)
```

In [131…

```
tags.isnull().any().any()
```

Out[131…

```
True
```

In [133…

```
tags= tags.dropna()
```

In [135…

```
tags.isnull().any().any()
```

Out[135…

```
False
```

In [137…

```
tags.shape
```

Out[137…

```
(465548, 3)
```

DATA--VISUALIZATION

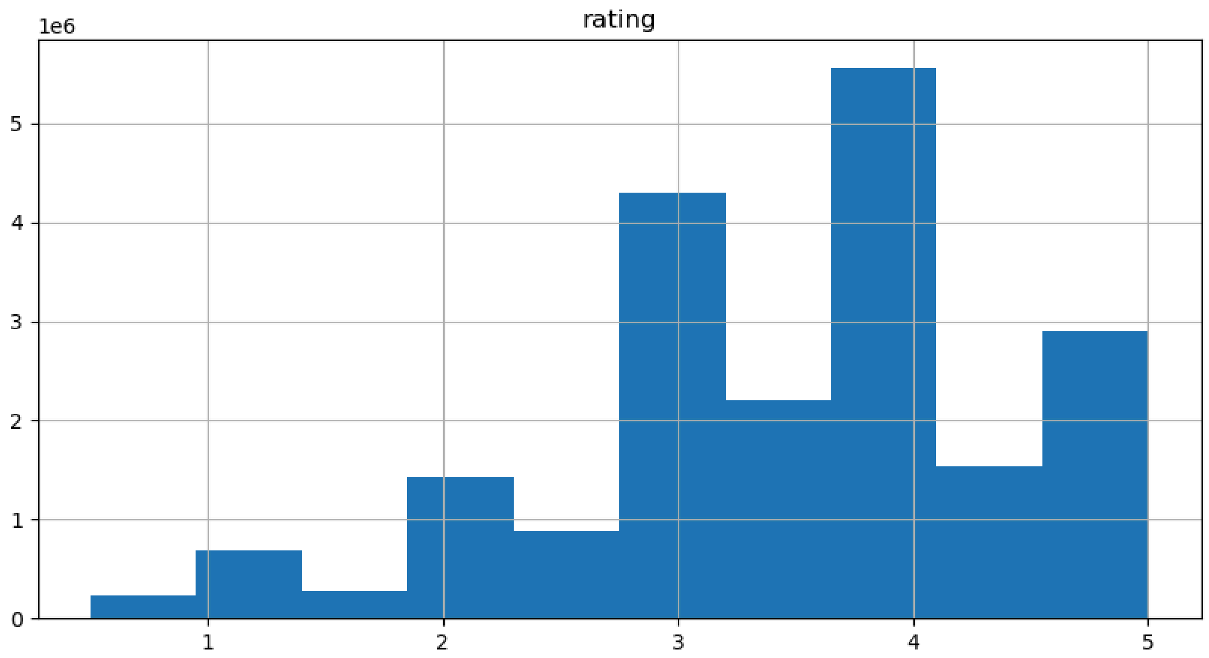
In [146…

```
%matplotlib inline

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

Out[146…

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

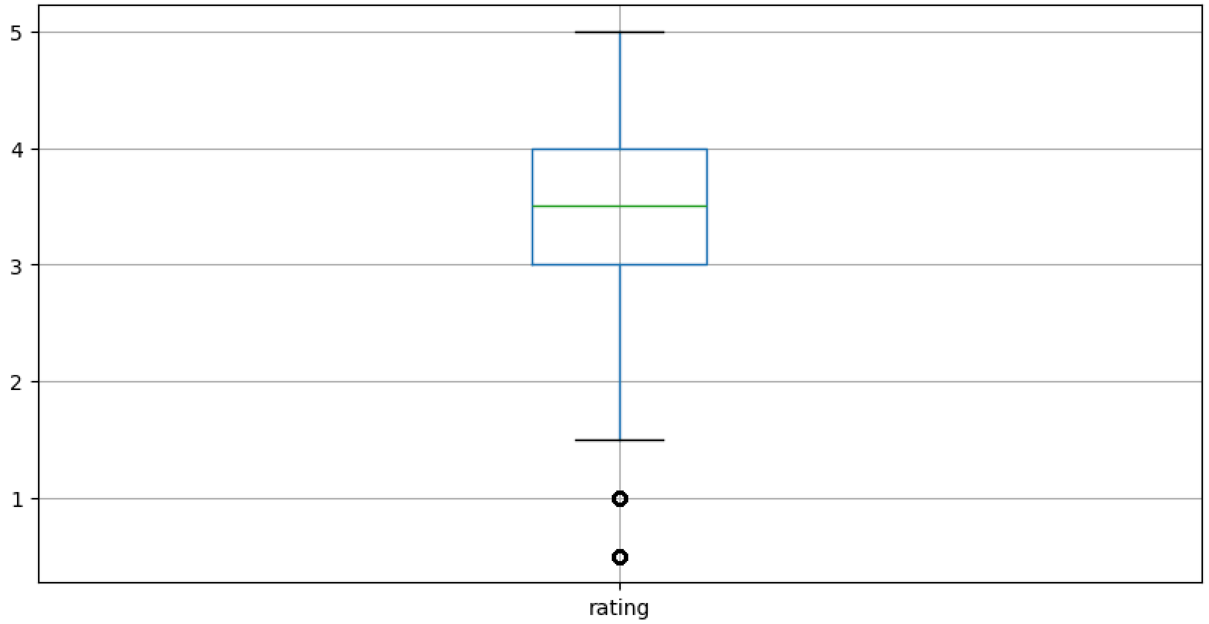


In [151…

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

Out[151…

```
<Axes: >
```



SLICING OUT COLUMNS

In [154…

```
tags['tag'].head()
```

Out[154…

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

In [156…

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

Out[156…

	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 [158…

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

Out[158…

	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 [164…

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

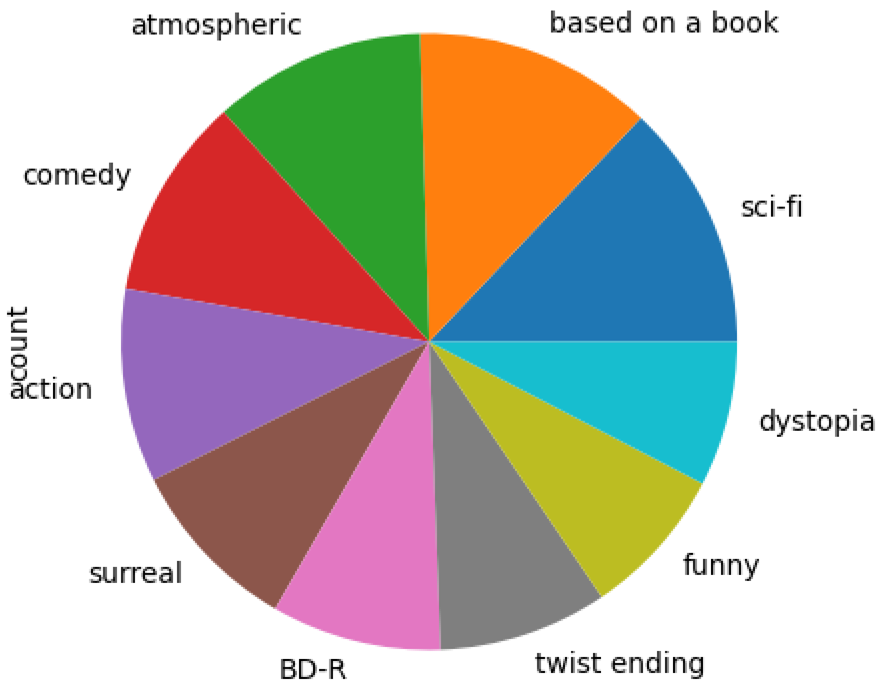
```
Out[164…  tag
          missing child                  1
          Ron Moore                      1
          Citizen Kane                   1
          mullet                         1
          biker gang                     1
          Paul Adelstein                 1
          the wig                        1
          killer fish                    1
          genetically modified monsters  1
          topless scene                  1
          Name: count, dtype: int64
```

```
In [166…  tag_counts = tags['tag'].value_counts()
          tag_counts[-10:]
```

```
Out[166…  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
```

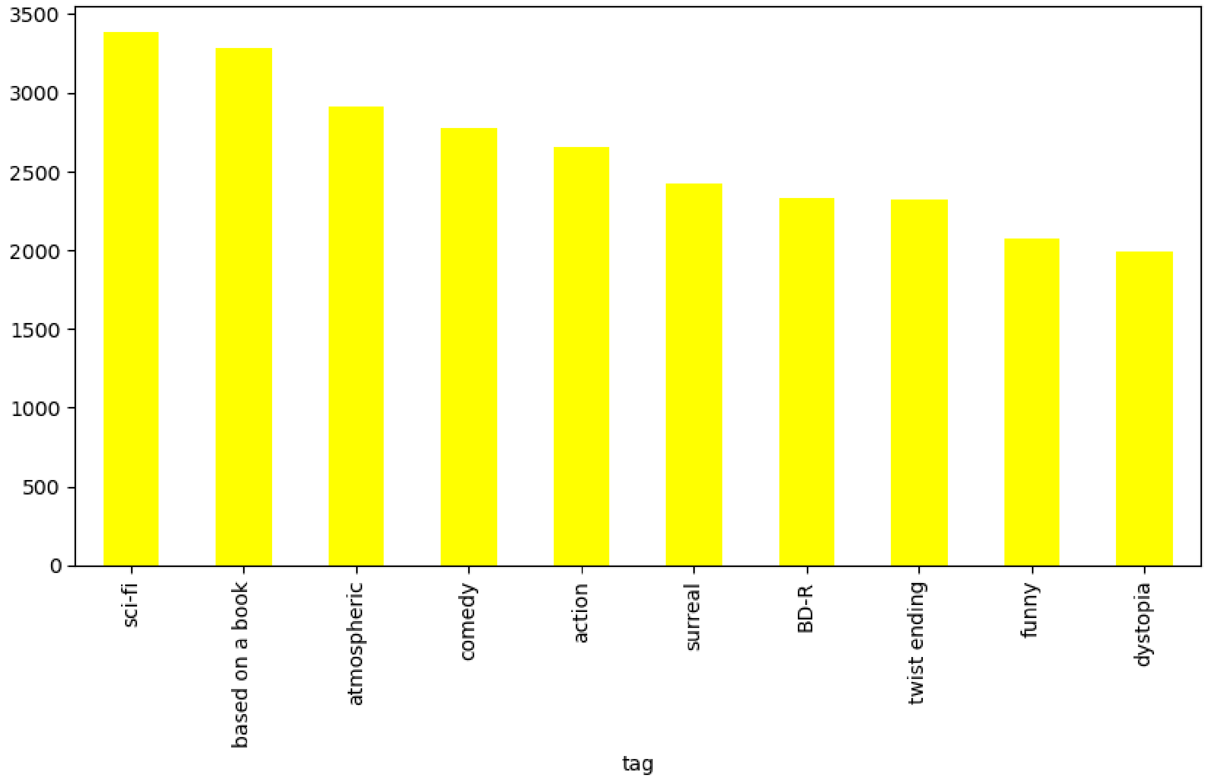
```
In [168…  tag_counts[:10].plot(kind='pie',figsize=(10,5))
```

```
Out[168…  <Axes: ylabel='count'>
```



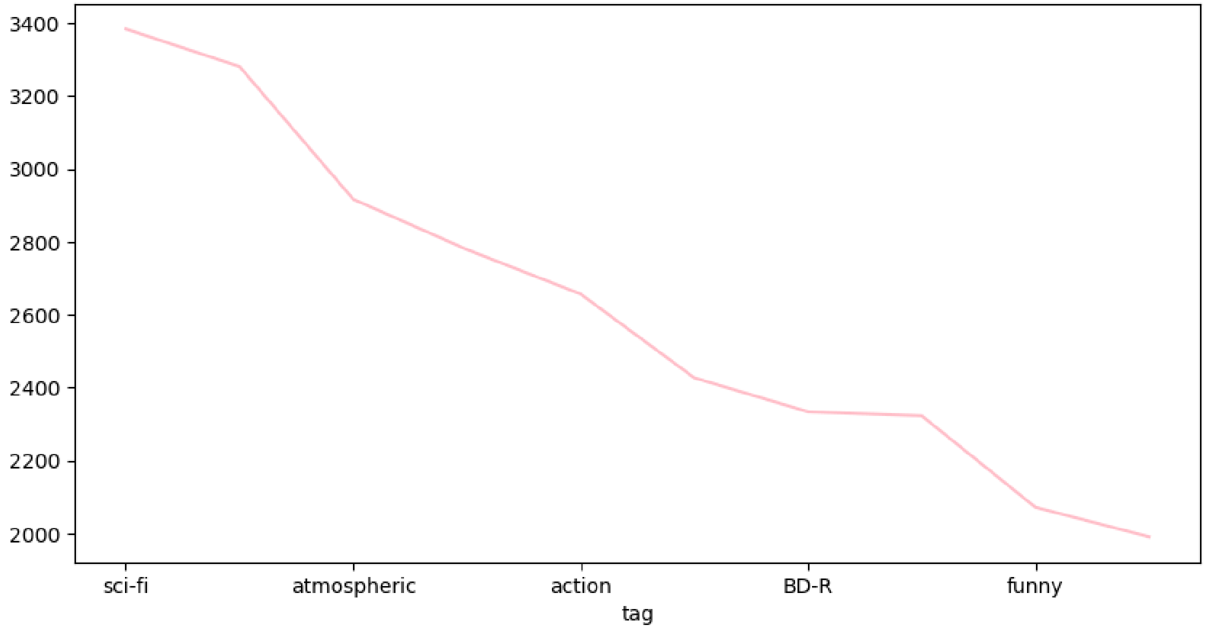
```
In [230… tag_counts[:10].plot(kind='bar', figsize=(10,5),color='yellow')
```

```
Out[230… <Axes: xlabel='tag'>
```



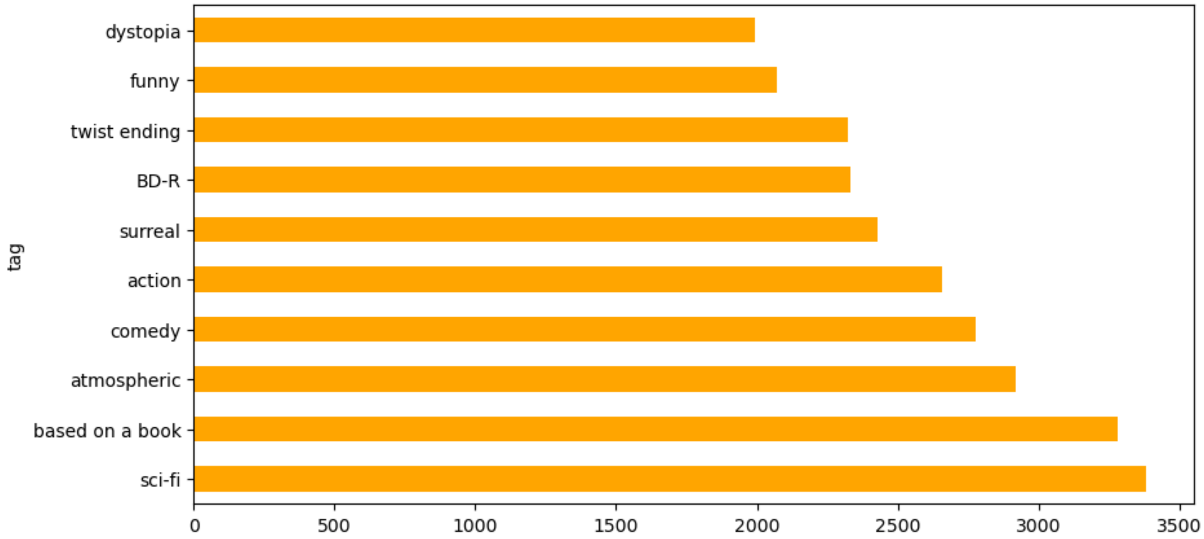
```
In [216… tag_counts[:10].plot(kind='line', figsize=(10,5),color='pink')
```

```
Out[216… <Axes: xlabel='tag'>
```



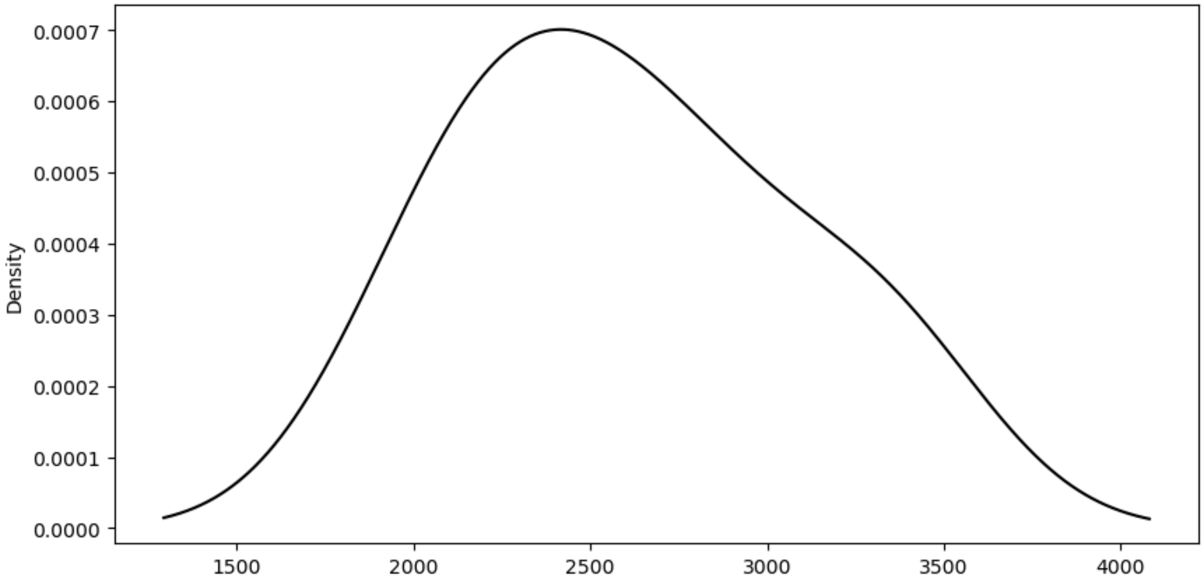
```
In [218… tag_counts[:10].plot(kind='barh', figsize=(10,5),color='orange')
```

```
Out[218… <Axes: ylabel='tag'>
```



```
In [220… tag_counts[:10].plot(kind='density', figsize=(10,5),color='black')
```

```
Out[220… <Axes: ylabel='Density'>
```

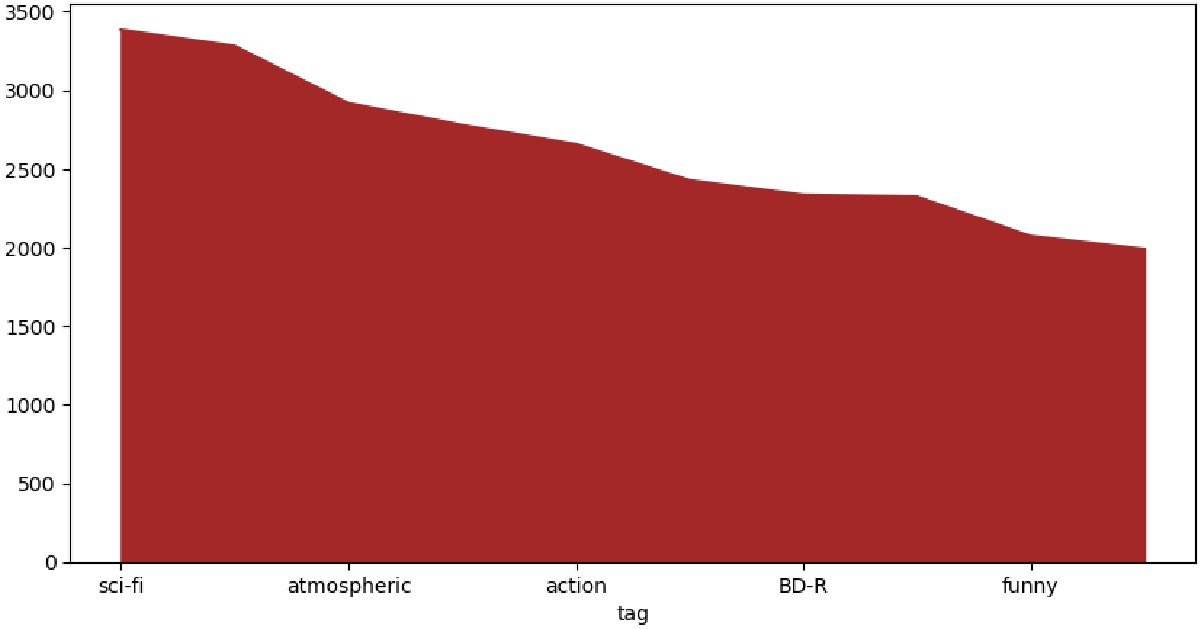


In [224…

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

Out[224…

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

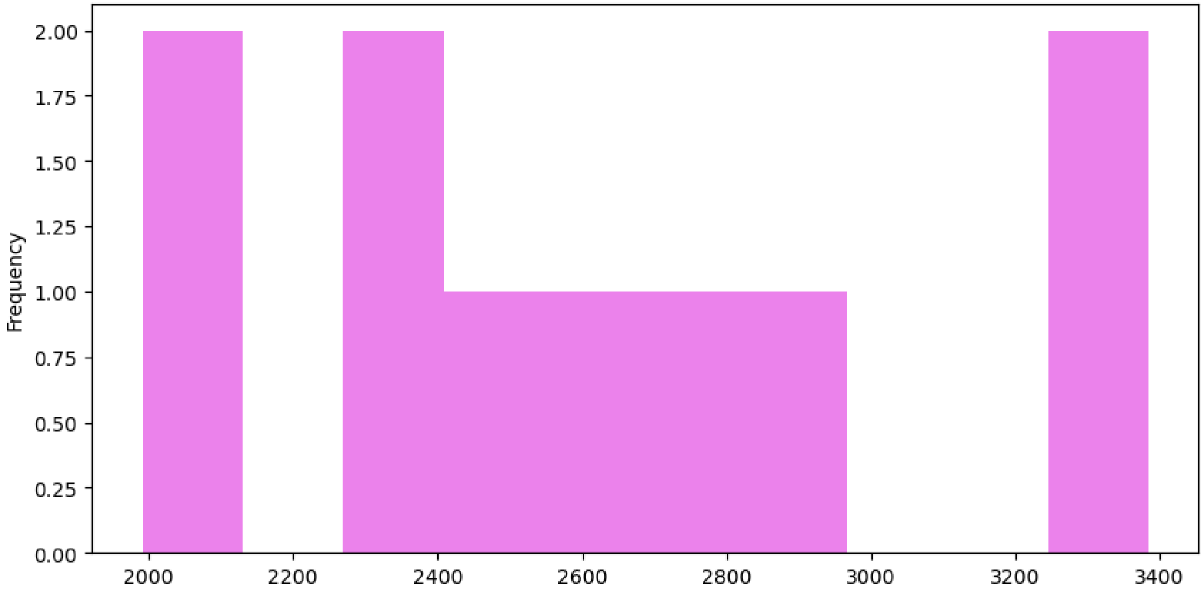


In [226…

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

Out[226…

```
<Axes: ylabel='Frequency'>
```


Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2000
2200
2400
2600
2800
3000
3200
3400

```
In [ ]:
```