In [1]:
```
import numpy as np
import pandas as pd
from scipy import spatial
import re
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
anime = pd.read_csv('anime.csv')
anime = anime.dropna()
anime = anime[anime['episodes'] != 'Unknown']
rating = pd.read_csv('rating.csv')
```

In [3]:
```
anime_full=pd.merge(anime,rating,on='anime_id',suffixes= ['', '_user'])
```

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

Out[4]:
```
(7813737, 3)
```

In [5]:
```
rating.head()
```

Out[5]:

	user_id	anime_id	rating
0	1	20	-1
1	1	24	-1
2	1	79	-1
3	1	226	-1
4	1	241	-1

In [6]:
```
ratings = rating[rating['rating'] != -1]
```

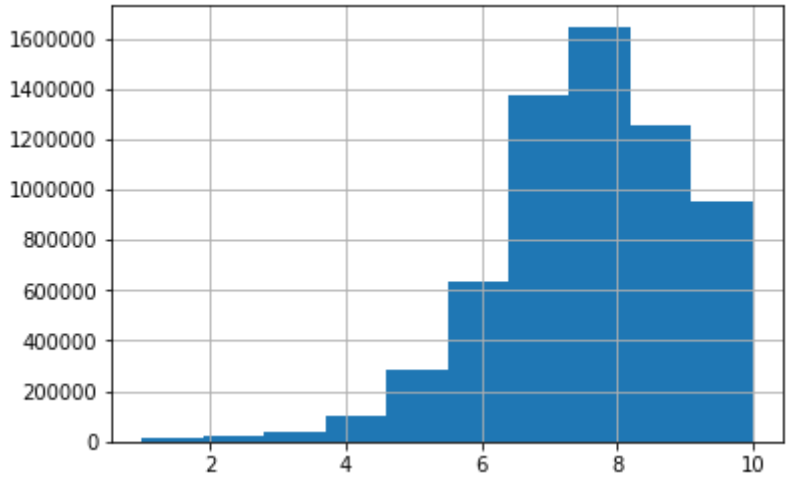
In [7]:
```
ratings.head()
```

Out[7]:

	user_id	anime_id	rating
47	1	8074	10
81	1	11617	10
83	1	11757	10
101	1	15451	10
153	2	11771	10

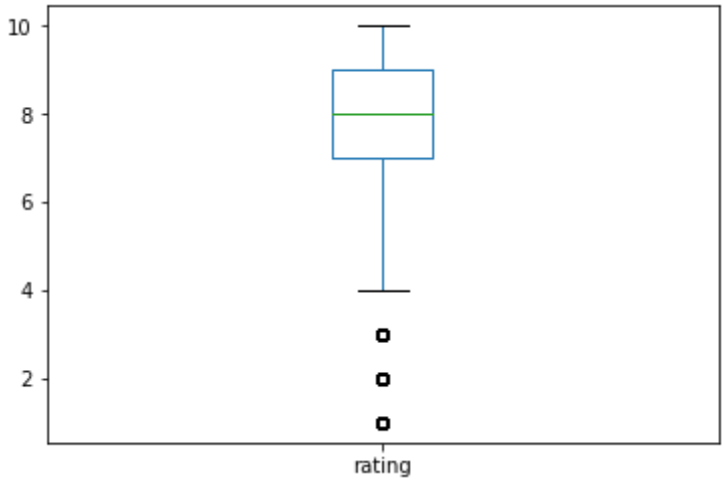
In [8]:
```
ratings['rating'].hist()
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9d01b7a240>
```



In [9]:
```
ratings['rating'].plot(kind = 'box', subplots = True)
```

Out[9]:
```
rating    AxesSubplot(0.125,0.125;0.775x0.755)
dtype: object
```



Anime ratings aggregated by user

In [10]:
```
uRatings = ratings.groupby(['user_id']).agg({'rating' : [np.size, np.mean]})
```

In [11]:
```
uRatings.reset_index(inplace = True)
```

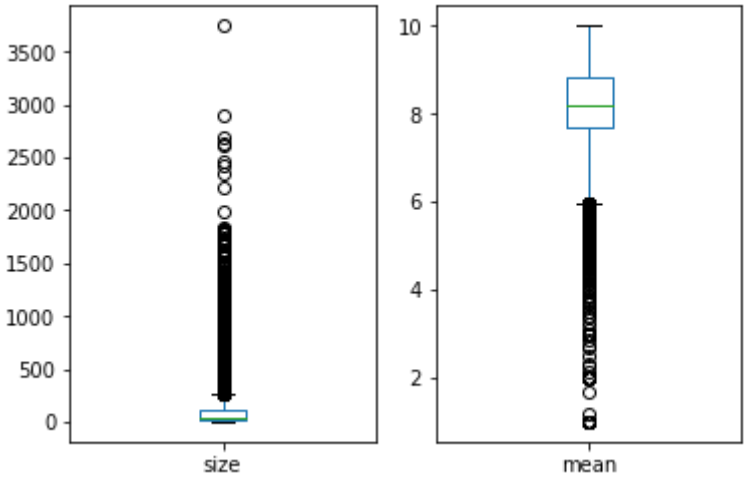
In [12]:
```
uRatings['rating'].describe()
```

Out[12]:

	size	mean
count	69600.000000	69600.000000
mean	91.052313	8.227761
std	135.764253	0.902856
min	1.000000	1.000000
25%	13.000000	7.666667
50%	45.000000	8.193548
75%	114.000000	8.815789
max	3747.000000	10.000000

In [13]:
```
uRatings['rating'].plot(kind = 'box', subplots = True)
```

Out[13]:
```
size       AxesSubplot(0.125,0.125;0.352273x0.755)
mean    AxesSubplot(0.547727,0.125;0.352273x0.755)
dtype: object
```



Anime ratings aggregated by anime

In [14]:
```
mRatings = ratings.groupby(['anime_id']).agg({'rating' : [np.size, np.mean]})
mRatings.reset_index(inplace = True)
```

In [15]:
```
mRatings['rating'].describe()
```

Out[15]:

	size	mean
count	9927.000000	9927.000000
mean	638.384305	6.637940
std	1795.865541	1.298863
min	1.000000	1.000000
25%	9.000000	6.066667
50%	57.000000	6.897959
75%	395.000000	7.491484
max	34226.000000	10.000000

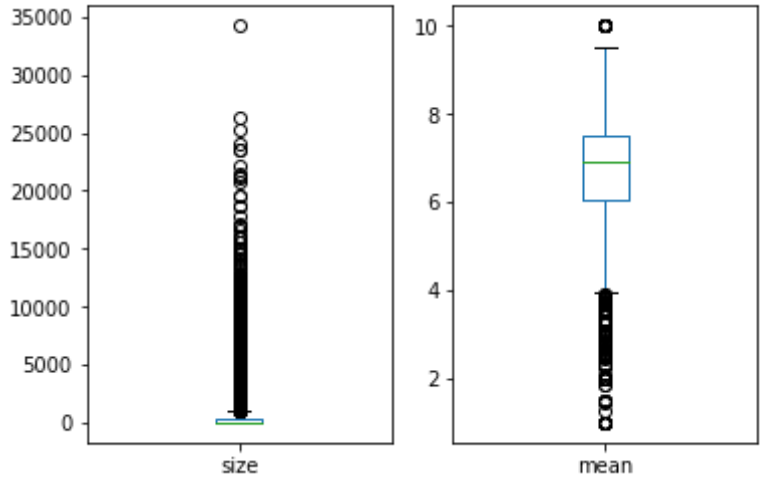
In [16]:
```
mRatings.head()
```

Out[16]:

	anime_id	rating	
		size	mean
0	1	13449	8.869433
1	5	5790	8.439724
2	6	9385	8.419393
3	7	2169	7.533426
4	8	308	7.198052

In [17]:
```
mRatings['rating'].plot(kind = 'box', subplots = True)
```

Out[17]:
```
size       AxesSubplot(0.125,0.125;0.352273x0.755)
mean    AxesSubplot(0.547727,0.125;0.352273x0.755)
dtype: object
```



Ger genre Matrix

In [18]:
```
animes = anime.merge(mRatings, on = 'anime_id', how = 'left')
animes.columns = ['anime_id', 'name', 'genre', 'type', 'episodes', 'rating', 'members', 'rating_coun
t', 'rating_avg']
```

```
/Users/boris/anaconda3/envs/python/lib/python3.7/site-packages/pandas/core/reshape/merge.py:522: Use
rWarning: merging between different levels can give an unintended result (1 levels on the left, 2 on
the right)
  warnings.warn(msg, UserWarning)
/Users/boris/anaconda3/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:3812: Performa
nceWarning: dropping on a non-lexsorted multi-index without a level parameter may impact performanc
e.
  new_axis = axis.drop(labels, errors=errors)
```

In [19]:
```
animes.fillna(0,inplace = True)
```

Top 10 Animes based on rating count

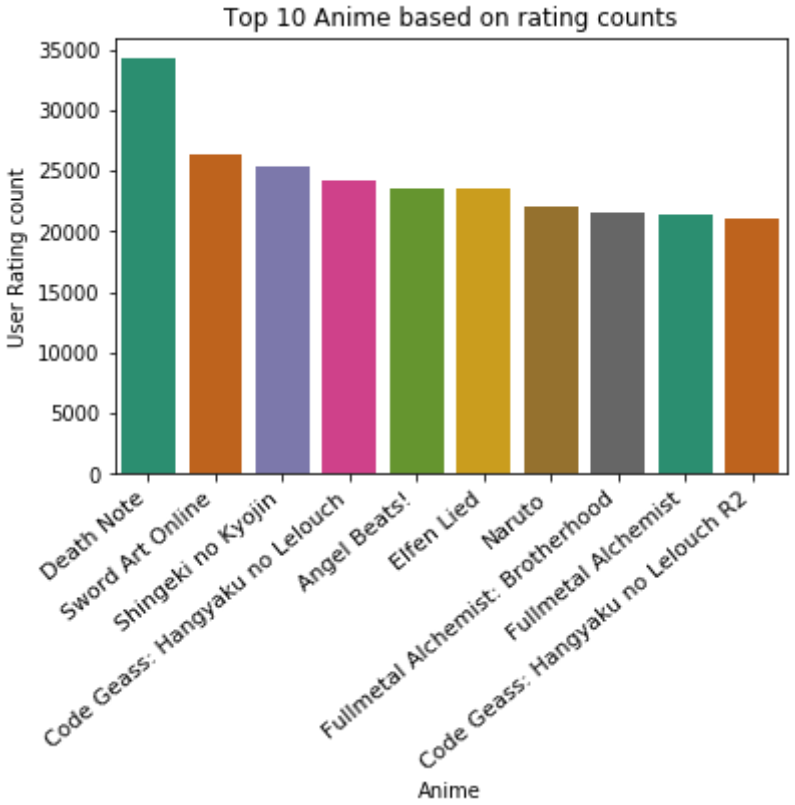
In [20]:
```
top10_animerating=animes[['name', 'rating_count']].sort_values(by = 'rating_count',ascending = False)
.head(10)
top10_animerating
```

Out[20]:

	name	rating_count
40	Death Note	34226.0
801	Sword Art Online	26310.0
85	Shingeki no Kyojin	25290.0
19	Code Geass: Hangyaku no Lelouch	24126.0
158	Angel Beats!	23565.0
757	Elfen Lied	23528.0
838	Naruto	22071.0
1	Fullmetal Alchemist: Brotherhood	21494.0
199	Fullmetal Alchemist	21332.0
13	Code Geass: Hangyaku no Lelouch R2	21124.0

In [21]:
```
ax=sns.barplot(x="name", y="rating_count", data=top10_animerating, palette="Dark2")
ax.set_xticklabels(ax.get_xticklabels(), fontsize=11, rotation=40, ha="right")
ax.set_title('Top 10 Anime based on rating counts',fontsize = 12)
ax.set_xlabel('Anime',fontsize = 10)
ax.set_ylabel('User Rating count', fontsize = 10)
```

Out[21]:
```
Text(0, 0.5, 'User Rating count')
```



Top 10 Animes based on Community size

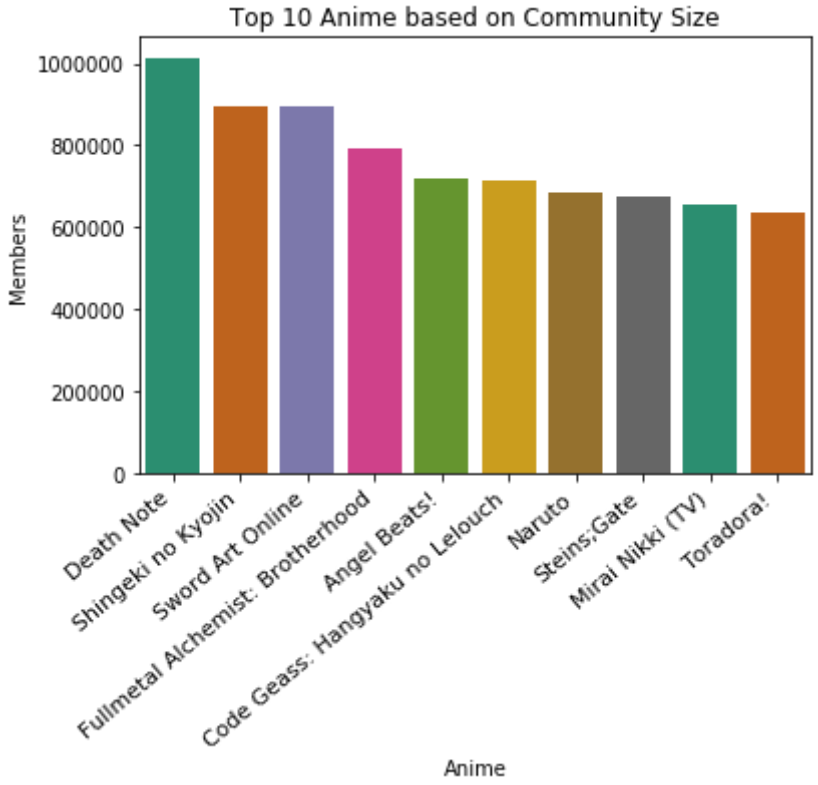
In [22]:
```
top10_animemember=animes[['name', 'members']].sort_values(by = 'members',ascending = False).head(10)
top10_animemember
```

Out[22]:

	name	members
40	Death Note	1013917
85	Shingeki no Kyojin	896229
801	Sword Art Online	893100
1	Fullmetal Alchemist: Brotherhood	793665
158	Angel Beats!	717796
19	Code Geass: Hangyaku no Lelouch	715151
838	Naruto	683297
3	Steins;Gate	673572
443	Mirai Nikki (TV)	657190
130	Toradora!	633817

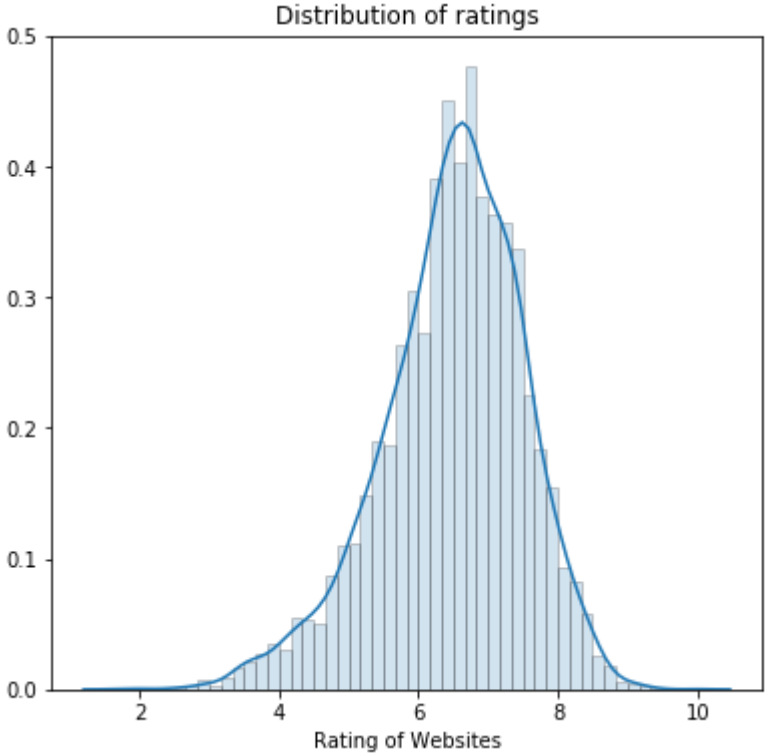
In [23]:
```
ax=sns.barplot(x="name", y="members", data=top10_animemember, palette="Dark2")
ax.set_xticklabels(ax.get_xticklabels(), fontsize=11, rotation=40, ha="right")
ax.set_title('Top 10 Anime based on Community Size',fontsize = 12)
ax.set_xlabel('Anime',fontsize = 10)
ax.set_ylabel('Members', fontsize = 10)
```

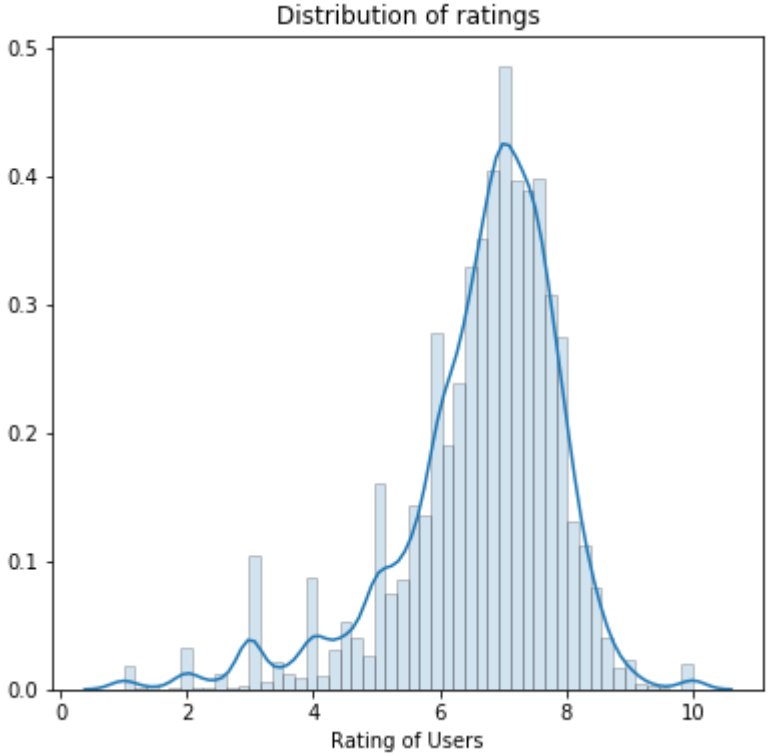
Out[23]: Text(0, 0.5, 'Members')



Distribution of ratings

In [24]:
```
hist_kws = {'histtype':'bar', 'edgecolor':'black', 'alpha':0.2}
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (14,6))
sns.distplot(animes['rating'], hist_kws=hist_kws, ax = ax[0])
sns.distplot(animes[animes['rating_avg'] != 0]['rating_avg'], hist_kws=hist_kws, ax = ax[1])
ax[0].set_title('Distribution of ratings')
ax[0].set_xlabel('Rating of Websites')
ax[1].set_title('Distribution of ratings')
ax[1].set_xlabel('Rating of Users')
plt.show()
```



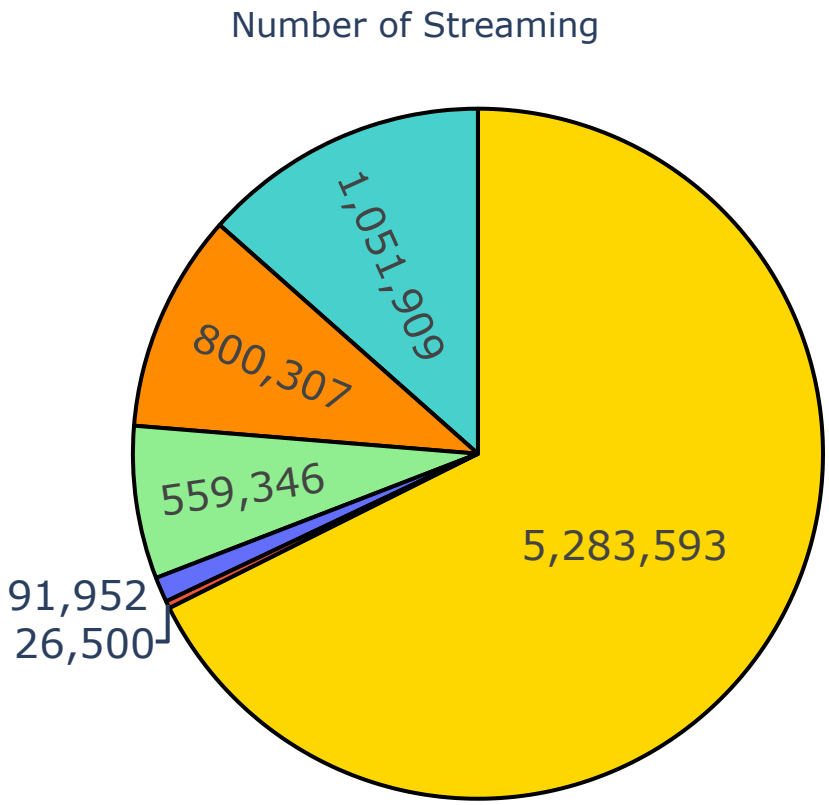


Streaming Type

In [25]:
```
import plotly.graph_objects as go
labels = anime_full['type'].value_counts().index
values = anime_full['type'].value_counts().values
colors = ['gold', 'mediumturquoise', 'darkorange', 'lightgreen']
fig = go.Figure(data=[go.Pie(labels=labels,
                             values=values)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=2)))

fig.update_layout(
    title={
        'text': "Number of Streaming",
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})

fig.show()
```


Number of Streaming
1,051,909
800,307
559,346
5,283,593
91,952
26,500

In [26]:
```
nonull_anime=anime_full.copy()
nonull_anime.dropna(inplace=True)
from collections import defaultdict

all_genres = defaultdict(int)

for genres in nonull_anime['genre']:
    for genre in genres.split(','):
        all_genres[genre.strip()] += 1

from wordcloud import WordCloud

genres_cloud = WordCloud(width=800, height=400, background_color='white', colormap='gnuplot').generat
e_from_frequencies(all_genres)
plt.imshow(genres_cloud, interpolation='bilinear')
plt.axis('off')
```

Out[26]:
```
(-0.5, 799.5, 399.5, -0.5)
```

Movie recommendation algorithm

In [27]:
```
#Preprocessing
anime_feature=anime_full.copy()
```

In [28]:
```
anime_feature["rating_user"].replace({-1: np.nan}, inplace=True)
anime_feature.head()
```

Out[28]:

	anime_id	name	genre	type	episodes	rating	members	user_id	rating_user
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	99	5.0
1	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	152	10.0
2	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	244	10.0
3	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	271	10.0
4	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630	278	NaN

In [29]:
```
anime_feature = anime_feature.dropna(axis = 0, how ='any')
anime_feature.isnull().sum()
```

Out[29]:
```
anime_id       0
name           0
genre          0
type           0
episodes       0
rating         0
members        0
user_id        0
rating_user    0
dtype: int64
```

In [30]:
```
counts = anime_feature['user_id'].value_counts()
```

In [31]:
```
anime_feature = anime_feature[anime_feature['user_id'].isin(counts[counts >= 200].index)]
```

In [32]:
```
#Pivot Table
anime_pivot=anime_feature.pivot_table(index='name',columns='user_id',values='rating_user').fillna(0)
anime_pivot.head()
```

Out[32]:

user_id	5	7	17	38	43	46	123	129	139	160	...	73406	73417	73422	73457	73460	73476	73499	73502	73503
name																				
"0"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0
"Bungaku Shoujo" Memoire	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
"Bungaku Shoujo" Movie	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0
"Eiji"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 8713 columns

Collaborative Filtering

In [33]:
```
from scipy.sparse import csr_matrix
anime_matrix = csr_matrix(anime_pivot.values)
from sklearn.neighbors import NearestNeighbors
model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(anime_matrix)
```

Out[33]:
```
NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

Clean Anime name

In [34]:
```
import re
def text_cleaning(text):
    text = re.sub(r'&quot;', '', text)
    text = re.sub(r'.hack//', '', text)
    text = re.sub(r'&#039;', '', text)
    text = re.sub(r'A&#039;s', '', text)
    text = re.sub(r'I&#039;', 'I\'', text)
    text = re.sub(r'&amp;', 'and', text)
    return text
```

In [35]:
```
anime['name'] = anime['name'].apply(text_cleaning)
```

Term Frequency (TF) and Inverse Document Frequency (IDF)

In [36]:
```
from sklearn.feature_extraction.text import TfidfVectorizer


tfv = TfidfVectorizer(min_df=3,  max_features=None,
            strip_accents='unicode', analyzer='word',token_pattern=r'\w{1,}',
            ngram_range=(1, 3),
            stop_words = 'english')

# Filling NaNs with empty string
anime['genre'] = anime['genre'].fillna('')
genres_str = anime['genre'].str.split(',').astype(str)
tfv_matrix = tfv.fit_transform(genres_str)
```

In [37]:
```
tfv_matrix.shape
```

Out[37]:
```
(11830, 1547)
```

In [38]:
```
from sklearn.metrics.pairwise import sigmoid_kernel

# Compute the sigmoid kernel
sig = sigmoid_kernel(tfv_matrix, tfv_matrix)
```

In [39]:
```
indices = pd.Series(anime.index, index=anime['name']).drop_duplicates()
```

Recommendation Function (content based)

In [40]:
```
def recommendation(title, sig=sig):
    idx = indices[title]
    sig_scores = list(enumerate(sig[idx]))
    sig_scores = sorted(sig_scores, key=lambda x: x[1], reverse=True)
    sig_scores = sig_scores[1:11]
    anime_indices = [i[0] for i in sig_scores]
    return pd.DataFrame({'Anime name': anime['name'].iloc[anime_indices].values,
                                 'Rating': anime['rating'].iloc[anime_indices].values})
```

In [41]:
```
recommendation('One Punch Man')
```

Out[41]:

	Anime name	Rating
0	One Punch Man Specials	7.86
1	One Punch Man: Road to Hero	7.85
2	Genji Tsuushin Agedama	6.58
3	Oh! Super Milk-chan	6.07
4	Super Milk-chan	5.88
5	Bobobo-bo Bo-bobo Recap	6.54
6	Gungrave	7.97
7	Darker than Black: Kuro no Keiyakusha Special	7.65
8	Haiyore! Nyaruko-san W	7.43
9	Haiyore! Nyaruko-san: Yasashii Teki no Shitome...	7.27

In [ ]:
```

```