ANIME RECOMMENDATION SYSTEM

1. BUSINESS UNDERSTANDING

Overview

Anime has become an increasingly popular genre of entertainment in recent years, brought about by nerd culture gaining popularity in the mainstream pop culture areas. Some people like it just for the subtitles or because they enjoy comics and animation. This project was inspired by nerd culture and the love for anime. The explosive growth in the amount of available digital information and the number of visitors to the Internet has created a big challenge where consumers have a wide variety of choices but yet very few choices at their disposal and producers have a difficult time figuring out their potential market. Information retrieval systems, such as Google, DevilFinder, and Altavista have partially solved this problem, but the personalization of this data to make relevant recommendations to consumers and producers was absent. This is where recommendation systems come in. Recommender systems are information filtering systems that deal with the problem of information overload by filtering vital information fragments out of large amounts of dynamically generated information according to the user's preferences, interests, or observed behavior about an item. A recommender system has the ability to predict whether a particular user would prefer an item or not based on the user's profile. Recommender systems are beneficial to both service providers and users. In an e-commerce setting, recommender systems enhance revenues, as they are an effective means of selling more products and they reduce the transaction costs of finding and selecting items. Recommendation systems have proved to improve the decision-making process and quality. In scientific libraries, recommender systems support

users by allowing them to move beyond catalog searches. Therefore, the need to use efficient and accurate recommendation techniques within a system that will provide relevant and dependable recommendations for users.

1.1 Problem Statement.

The Internet allows people to access abundant resources online. Streaming websites like Netflix, have an enormous collection of movies. With the amount of available content increasing, a new problem arose, that is, people have a hard time selecting the movies/shows they actually want to see from the large pool. Production companies also had a difficult time locating their target audience as the users are many and with very diverse tastes.

This is where our recommender system comes in. We need recommender systems in such that will assist people in finding content they are interested in and reduce the time and spare them some time searching.

1.2. Proposed Solution

The appropriate solution to deal with our problem is to come up with a system that would recommend movies to our consumers based on their preferences and tastes in order to maximize consumer utility and increase profits for producer companies.

The recommender system can be deployed on streaming websites to help producers market their products to the right consumers.

1.3. Specific Objectives

- To give anime recommendations to the user based on their tastes and preferences.
- To find out the most watched anime genre
- To determine the highest rated genre
- To determine the anime source with most members

1.4. Research Questions

- Which is the model that provides the best recommendations ?
- What are the most watched anime genre?
- What are the highest rated anime genre?
- Which anime sources have the most members

1.5. Success Criteria

Create a model that can recommend movies to users with an RMSE of 2 and below.

Loading Necessary Libraries

In [1]:
```
# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
```

Loading the Data

In this project, we will use two datasets;

- Anime dataset - Contains information on the anime titles such as their id, title, producer, studio etc.
- Ratings dataset - Contains information about the users such as their id, the animes they've watched and the ratings of the animes they have watched.

In [2]:
```
# load data
df_anime = pd.read_csv('Anime_data.csv')
df_rating = pd.read_csv('rating.csv')
```

2. DATA UNDERSTANDING

2.1 Anime Data

In the Data Understanding phase we shall carry out the following tasks:

- Preview the data
- Describe Data
- Explore Data
- Verify Data Quality

Preview the Data

In [3]:
```
# preview of first 5 rows
df_anime.head()
```

Out[3]:

	Anime_id	Title	Genre	Synopsis	Type	Producer	Studio	Rating	Score
0	1	Cowboy Bebop	['Action', 'Adventure', 'Comedy', 'Drama', 'Sc...	In the year 2071, humanity has colonized sever...	TV	['Bandai Visual']	['Sunrise']	8.81	3638
1	5	Cowboy Bebop: Tengoku no Tobira	['Action', 'Space', 'Drama', 'Mystery', 'Sci-Fi']	Another day, another bounty—such is the life o...	Movie	['Sunrise', 'Bandai Visual']	['Bones']	8.41	111
2	6	Trigun	['Action', 'Sci-Fi', 'Adventure', 'Comedy', 'D...	Vash the Stampede is the man with a $$60,000,0...	TV	['Victor Entertainment']	['Madhouse']	8.31	1974
3	7	Witch Hunter Robin	['Action', 'Magic', 'Police', 'Supernatural', ...	Witches are individuals with special powers li...	TV	['Bandai Visual']	['Sunrise']	7.34	318
4	8	Bouken Ou Beet	['Adventure', 'Fantasy', 'Shounen', 'Supernatu...	It is the dark century and the people are suff...	TV	NaN	['Toei Animation']	7.04	47

In [4]:
```
# checking last 15 rows
df_anime.tail(15)
```

Out[4]:

	Anime_id	Title	Genre	Synopsis	Type	Producer	Studio	Rating	ScoredBy	P
16987	12723	Loups=Garous Pilot	NaN	NaN	Special	NaN	NaN	5.87	NaN	
16988	32588	Meow no Hoshi	NaN	NaN	OVA	NaN	NaN	5.58	NaN	
16989	9056	Agitated Screams of Maggots	NaN	NaN	Music	NaN	NaN	4.45	NaN	
16990	33655	Alps no Shoujo Heidi? Chara Onji	NaN	NaN	TV	NaN	NaN	6.79	NaN	
16991	31385	Ginga Shounen Tai	NaN	NaN	TV	NaN	NaN	6.38	NaN	
16992	31605	Kana Kana Kazoku: Kakusan Mare Bo ! 1-Wa-5-wa ...	NaN	NaN	ONA	NaN	NaN	5.11	NaN	
16993	6366	Karuizawa Syndrome	NaN	NaN	OVA	NaN	NaN	6.27	NaN	
16994	13459	Ribbon-chan	NaN	NaN	TV	NaN	NaN	4.83	NaN	
16995	22391	Ring Ring Boy	NaN	NaN	Movie	NaN	NaN	4.40	NaN	
16996	22399	Saru Kani Gassen	NaN	NaN	OVA	NaN	NaN	5.23	NaN	
16997	27499	Sore Ike! Anpanman: Anpanman to Kaizoku Lobster	NaN	NaN	Special	NaN	NaN	5.50	NaN	
16998	12091	X Bomber	NaN	NaN	TV	NaN	NaN	5.94	NaN	
16999	29133	X Bomber Pilot	NaN	NaN	Special	NaN	NaN	4.28	NaN	
17000	34485	Ganko-chan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
17001	32713	Original C-V-P Momoko	NaN	NaN	OVA	NaN	NaN	4.00	NaN	

Description of Data

In [5]:
```
# shape of data
df_anime.shape
```

Out[5]:
```
(17002, 15)
```

In [6]:
```
# check info of data
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17002 entries, 0 to 17001
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    17002 non-null  int64  
 1   Title       17002 non-null  object 
 2   Genre       14990 non-null  object 
 3   Synopsis    15583 non-null  object 
 4   Type        16368 non-null  object 
 5   Producer    7635 non-null   object 
 6   Studio      7919 non-null   object 
 7   Rating      14425 non-null  float64
 8   ScoredBy    13227 non-null  float64
 9   Popularity  16368 non-null  float64
 10  Members     17002 non-null  float64
 11  Episodes    14085 non-null  float64
 12  Source      15075 non-null  object 
 13  Aired       16368 non-null  object 
 14  Link        16368 non-null  object 
dtypes: float64(5), int64(1), object(9)
memory usage: 1.9+ MB
```

The anime data consists of the following columns:

- `Anime_id` :anime Id (as per myanimelist.net)
- `Title` : Name of anime
- `Genre` :Main genres in the movie
- `Synopsis` :Brief Description
- `Type`
- `Producer` : Production company of the anime
- `Studio` : The studio that produced the anime
- `Rating` :Rating of anime as per myanimelist.net/(on a scale of 1-10)
- `ScoredBy` : Total no user scored given anime
- `Popularity` :Rank of anime based on popularity
- `Members` :No of members added given anime on their list
- `Episodes` : No. of episodes
- `Source`
- `Aired`
- `Link` : Link to the anime on myanimelist

The data seems to be quite comprehenssive with 12 columns having missing values. We shall deal with the in the Data Preparation section.

In [7]:
```
# summary statistics
df_anime.describe()
```

Out[7]:

	Anime_id	Rating	ScoredBy	Popularity	Members	Episodes
count	17002.000000	14425.000000	1.322700e+04	16368.000000	1.700200e+04	14085.000000
mean	20446.579638	6.287867	1.139084e+04	8131.919599	2.038130e+04	11.482712
std	14342.513259	1.141401	4.328434e+04	4714.683351	7.121404e+04	44.089040
min	1.000000	1.000000	1.000000e+00	1.000000	0.000000e+00	1.000000
25%	5581.500000	5.620000	4.300000e+01	4042.500000	1.450000e+02	1.000000
50%	21334.000000	6.410000	4.780000e+02	8115.000000	1.113000e+03	1.000000
75%	34789.250000	7.090000	3.831000e+03	12208.250000	7.855750e+03	12.000000
max	40960.000000	10.000000	1.006242e+06	16338.000000	1.451708e+06	1818.000000

Verifying Quality of Data

In [8]:
```
# check for the number of null values
df_anime.isna().sum()
```

Out[8]:
```
Anime_id         0
Title            0
Genre         2012
Synopsis      1419
Type           634
Producer      9367
Studio        9083
Rating        2577
ScoredBy      3775
Popularity     634
Members          0
Episodes      2917
Source        1927
Aired          634
Link           634
dtype: int64
```

In [9]:
```
# checking for duplicates
df_anime.duplicated().sum()
```

Out[9]:
```
63
```

DATA UNDERSTANDING SUMMARY

- The data has 17002 rows and 15 columns.
- The data has 5 columns of type `Float` , 1 column of type `int` and 9 columns of type `object` .
- The following columns have missing values;

- Genre
- Synopsis
- Type
- Producer
- Studio
- Rating
- ScoredBy
- Popularity
- Episodes
- Source
- Aired
- Link

- There are 63 duplicatesin our dataset.
- Data cleaning will be done in the Data cleaning phase.

2.2 Ratings data

In this section we shall carry out the following tasks:

- Preview the data
- Describe Data
- Explore Data
- Verify Data Quality

Preview the Data

In [10]:
```
# preview of first 5 rows
df_rating.head()
```

Out[10]:

	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 [11]:
```
# preview of the last 5 rows
df_rating.tail()
```

Out[11]:

	user_id	anime_id	rating
7813732	73515	16512	7
7813733	73515	17187	9
7813734	73515	22145	10
7813735	73516	790	9
7813736	73516	8074	9

Description of data

In [12]:
```
# shape of data
df_rating.shape
```

Out[12]:
```
(7813737, 3)
```

In [13]:
```
# info of data
df_rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7813737 entries, 0 to 7813736
Data columns (total 3 columns):
 #   Column    Dtype
---  ------    -----
 0   user_id   int64
 1   anime_id  int64
 2   rating    int64
dtypes: int64(3)
memory usage: 178.8 MB
```

The ratings data consists of the following columns:

- `user_id` :The unique identifier for each user
- `anime_id` : Anime id watched by the user
- `rating` : rating that the user gives the anime

In [14]:
```
# description of data
df_rating.describe()
```

Out[14]:

	user_id	anime_id	rating
count	7.813737e+06	7.813737e+06	7.813737e+06
mean	3.672796e+04	8.909072e+03	6.144030e+00
std	2.099795e+04	8.883950e+03	3.727800e+00
min	1.000000e+00	1.000000e+00	-1.000000e+00
25%	1.897400e+04	1.240000e+03	6.000000e+00
50%	3.679100e+04	6.213000e+03	7.000000e+00
75%	5.475700e+04	1.409300e+04	9.000000e+00
max	7.351600e+04	3.451900e+04	1.000000e+01

Verifying Quality of Data

In [15]:
```
# Checking for null values
df_rating.isna().sum()
```

Out[15]:
```
user_id     0
anime_id    0
rating      0
dtype: int64
```

In [16]:
```
# checking for duplicates
df_rating.duplicated().sum()
```

Out[16]:
```
1
```

DATA UNDERSTANDING SUMMARY

- The ratings data has 7813737 rows and 3 columns
- All the 3 columns are of type `int` .
- There are no missing values in the data.
- There is only one duplicate record in the data.
- Data cleaning will be done in the Data Cleaning phase

3. DATA PREPARATION

3.1 Anime Data

3.1.1 Completeness

In [17]:
```
# Function for checking percentage of missing values
def missing_values(data):
    """
    Identify the missing values and their percentages
    Drop values that have no missing values
    Return only data with missing values
    """
    miss_val = data.isna().sum().sort_values(ascending = False)
    percentage = (data.isna().sum() / len(data)*100).sort_values(ascending = Fals
    missing_values = pd.DataFrame({"Missing Values": miss_val, "In Percentage": p
    missing_values.drop(missing_values[missing_values["In Percentage"] == 0].inde
    return missing_values
```

In [18]:
```
#Checking for missing values
missing_values(df_anime)
```

Out[18]:

	Missing Values	In Percentage
Producer	9367	55.093518
Studio	9083	53.423127
ScoredBy	3775	22.203270
Episodes	2917	17.156805
Rating	2577	15.157040
Genre	2012	11.833902
Source	1927	11.333961
Synopsis	1419	8.346077
Link	634	3.728973
Aired	634	3.728973
Popularity	634	3.728973
Type	634	3.728973

Producer and Studio columns have above 50% missing values

In [19]:
```
#dropping rows with missing values below 15%
df_anime.dropna(axis=0, subset=['Link','Aired','Popularity','Type','Synopsis','Sc
```

In [20]:
```
missing_values(df_anime)
```

Out[20]:

	Missing Values	In Percentage
Producer	6350	46.205341
Studio	6093	44.335298
Rating	1219	8.869970
ScoredBy	1138	8.280579
Episodes	729	5.304519

In [21]:
```
# filling the missing values in the ScoredBy column with the median
df_anime['ScoredBy'].fillna(df_anime['ScoredBy'].median(), inplace= True)
```

In [22]:
```
missing_values(df_anime)
```

Out[22]:

	Missing Values	In Percentage
Producer	6350	46.205341
Studio	6093	44.335298
Rating	1219	8.869970
Episodes	729	5.304519

In [23]:
```
# dropping the missing values in the Episodes column
df_anime.dropna(subset=['Episodes'],inplace=True)
```

In [24]:
```
missing_values(df_anime)
```

Out[24]:

	Missing Values	In Percentage
Producer	5788	44.475181
Studio	5618	43.168895
Rating	765	5.878285

In [25]:
```
# filling missing values in the rating column with the median
df_anime['Rating'].fillna(df_anime['Rating'].median(),inplace=True)
```

In [26]:
```
missing_values(df_anime)
```

Out[26]:

	Missing Values	In Percentage
Producer	5788	44.475181
Studio	5618	43.168895

In [27]:
```
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13014 entries, 0 to 15064
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    13014 non-null  int64  
 1   Title       13014 non-null  object 
 2   Genre       13014 non-null  object 
 3   Synopsis    13014 non-null  object 
 4   Type        13014 non-null  object 
 5   Producer    7226 non-null   object 
 6   Studio      7396 non-null   object 
 7   Rating      13014 non-null  float64
 8   ScoredBy    13014 non-null  float64
 9   Popularity  13014 non-null  float64
 10  Members     13014 non-null  float64
 11  Episodes    13014 non-null  float64
 12  Source      13014 non-null  object 
 13  Aired       13014 non-null  object 
 14  Link        13014 non-null  object 
dtypes: float64(5), int64(1), object(9)
memory usage: 1.6+ MB
```

The `Producer` and `Studio` columns still have missing values. We will fill the columns with unknown since we do not know the prodecer and the studio.

In [28]:
```
# filling missing values in the producer and studio columns
df_anime[["Producer", "Studio"]] = df_anime[["Producer", "Studio"]].fillna("Unkno
```

In [29]:
```
# confirming there are no missing values
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13014 entries, 0 to 15064
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    13014 non-null  int64  
 1   Title       13014 non-null  object 
 2   Genre       13014 non-null  object 
 3   Synopsis    13014 non-null  object 
 4   Type        13014 non-null  object 
 5   Producer    13014 non-null  object 
 6   Studio      13014 non-null  object 
 7   Rating      13014 non-null  float64
 8   ScoredBy    13014 non-null  float64
 9   Popularity  13014 non-null  float64
 10  Members     13014 non-null  float64
 11  Episodes    13014 non-null  float64
 12  Source      13014 non-null  object 
 13  Aired       13014 non-null  object 
 14  Link        13014 non-null  object 
dtypes: float64(5), int64(1), object(9)
memory usage: 1.6+ MB
```

Our DataFrame is now complete with missing values taken care of.

In [30]:
```
def missing_values_col(df, col):
    """
    This function evaluates the total number of missing values in a column and
    the respective percentages
    """
    print(f"Missing Values: {df[col].isna().sum()}")
    print(f"Percentage: {round((df[col].isna().sum() / len(df)) * 100, 2)}%")
```

In [31]:
```
#Checking for the ScoredBy column missing values
missing_values_col(df_anime, 'ScoredBy')
```

```
Missing Values: 0
Percentage: 0.0%
```

3.1.2 Validity

Here we will be checking for:

- Duplicates
- Outliers

In [32]:
```
#Checking for duplicates
df_anime.duplicated().sum()
```

Out[32]: 35

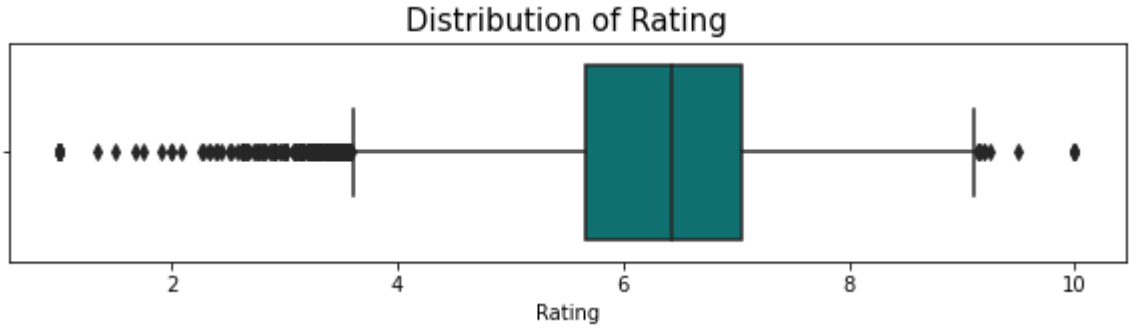
In [33]:
```
#Dropping rows that are duplicated
df_anime.drop_duplicates(inplace=True)
```

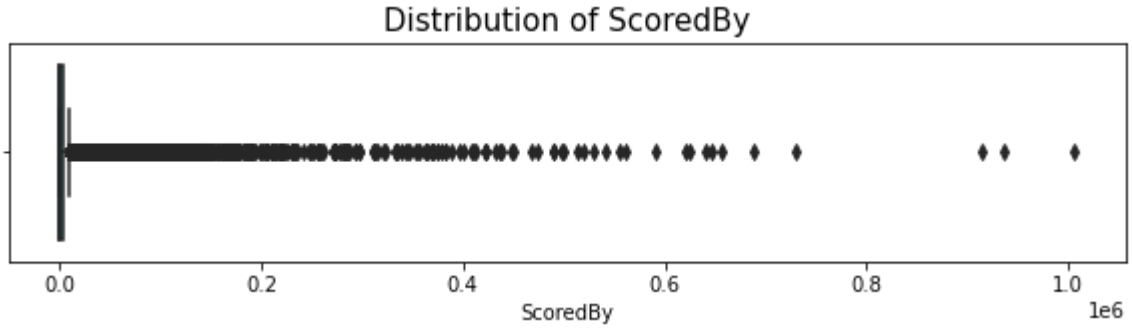
In [34]:
```
#Checking for duplicates
df_anime.duplicated().sum()
```

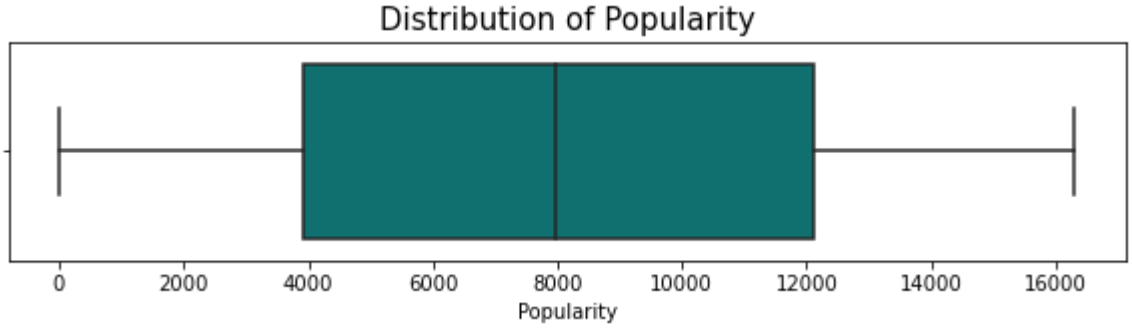
Out[34]: 0

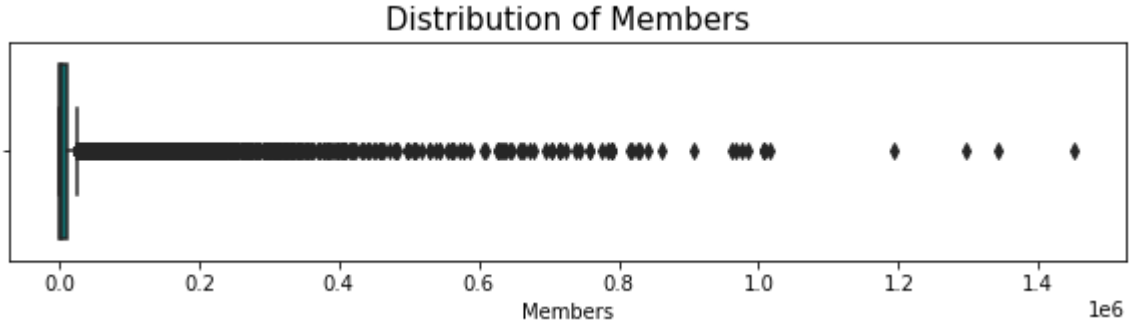
In [35]:
```
#Checking for outliers
def outliers(df, col):
    """
    This function visualises the outliers and outputs boxplots
    """
    plt.figure(figsize=(10,2))
    sns.boxplot(x=col, data=df, color= "#008080")
    plt.title(f'Distribution of {col}', fontsize=15);
```

In [36]:
```
num_list=['Rating','ScoredBy','Popularity','Members']
for col in num_list:
    outliers(df_anime,col)
```









- The `Rating` column is well distributed with ratings withing the range of 1-10
- The `ScoredBy` column has outliers but we will not remove them since they are genuine based on the structure of our data
- The `Popularity` column is distributed between 4000-12000 with no outliers
- The `Distribution of Members` column has outliers which are also genuine therefore we will not remove them

3.1.3 Uniformity

In this section, we will clean the `Title` column by removing the unnecessary characters in the anime titles.

In [37]:
```
# Creating a function to clean the title column
import re

def clean_title(title):
    title = re.sub("[^a-zA-Z0-9 ]", "", title)
    return title

df_anime["Title"] = df_anime["Title"].apply(clean_title)
df_anime
```

Out[37]:

	Anime_id	Title	Genre	Synopsis	Type	Producer	Studio	I
0	1	Cowboy Bebop	['Action', 'Adventure', 'Comedy', 'Drama', 'Sc...	In the year 2071, humanity has colonized sever...	TV	['Bandai Visual']	['Sunrise']	
1	5	Cowboy Bebop Tengoku no Tobira	['Action', 'Space', 'Drama', 'Mystery', 'Sci-Fi']	Another day, another bounty—such is the life o...	Movie	['Sunrise', 'Bandai Visual']	['Bones']	
2	6	Trigun	['Action', 'Sci-Fi', 'Adventure', 'Comedy', 'D...	Vash the Stampede is the man with a $$60,000,0...	TV	['Victor Entertainment']	['Madhouse']	
3	7	Witch Hunter Robin	['Action', 'Magic', 'Police', 'Supernatural', ...	Witches are individuals with special powers li...	TV	['Bandai Visual']	['Sunrise']	
4	8	Bouken Ou Beet	['Adventure', 'Fantasy', 'Shounen', 'Supernatu...	It is the dark century and the people are suff...	TV	Unknown	['Toei Animation']	
...	...	...	...	...	...	...	...	
15043	40033	Animagear	['Kids', 'Mecha']	A promotional video for the new Bandai candy-t...	ONA	Unknown	Unknown	
15049	40042	Magical Halloween MiracleQuartet	['Comedy', 'Ecchi', 'Fantasy', 'Game', 'Music']	Bundled with the "Magical Halloween: Miracle Q...	Music	Unknown	Unknown	
15061	40055	Orbital Era	['Action', 'Adventure', 'Fantasy', 'Sci-Fi', '...	Orbital Era is set in the near-future on a sp...	Movie	Unknown	Unknown	
15063	40057	Akai Hana Shiroi Hana	['Kids', 'Music']	Akai Hana Shiroi Hana was a song was original...	Music	Unknown	Unknown	

	Anime_id	Title	Genre	Synopsis	Type	Producer	Studio	I
15064	40058	Arui Tekoteko	['Kids', 'Music']	Of a pair of shoes, the left shoe is an animat...	Music	Unknown	Unknown	

12979 rows × 15 columns

3.1.4 Consistency

In [38]:
```
def get_value_counts(df, col):
    ''' Returns the value counts of a column in a dataframe '''
    counts = df[col].value_counts(dropna=False, ascending=False)
    return counts
```

In [39]:
```
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12979 entries, 0 to 15064
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    12979 non-null  int64  
 1   Title       12979 non-null  object 
 2   Genre       12979 non-null  object 
 3   Synopsis    12979 non-null  object 
 4   Type        12979 non-null  object 
 5   Producer    12979 non-null  object 
 6   Studio      12979 non-null  object 
 7   Rating      12979 non-null  float64
 8   ScoredBy    12979 non-null  float64
 9   Popularity  12979 non-null  float64
 10  Members     12979 non-null  float64
 11  Episodes    12979 non-null  float64
 12  Source      12979 non-null  object 
 13  Aired       12979 non-null  object 
 14  Link        12979 non-null  object 
dtypes: float64(5), int64(1), object(9)
memory usage: 1.6+ MB
```

In [40]:
```
# value counts for genre column
get_value_counts(df_anime,'Genre')
```

Out[40]:
```
['Hentai']
692
['Music']
625
['Comedy']
496
['Kids']
300
['Dementia']
175

...
['Action', 'Fantasy', 'Supernatural', 'Music', 'Seinen']
1
['Sci-Fi', 'Comedy', 'Parody', 'Romance']
1
['Dementia', 'Horror', 'Psychological', 'Drama']
1
['Comedy', 'Ecchi', 'Harem', 'Romance', 'School', 'Shounen', 'Slice of Life',
'Supernatural']      1
['Fantasy', 'Horror', 'Mystery']
1
Name: Genre, Length: 4199, dtype: int64
```

In [41]:
```
# value counts for synopsis column
get_value_counts(df_anime,'Synopsis')
```

Out[41]:
```
No synopsis has been added for this series yet. \r\n \r\n Click here  to update
this information.
31
Film by Takashi Ito.
13
Furukawa Taku film.
12
Short animation by Rapparu.
8
A short film by Okamoto Tadanari.
8


..
Murasame Sumika is popular in the high school for her excellence in the marks a
nd sports. However, she has a secret: she is in love with her classmate Kazama
Ushio. Ushio also has a liking to the love between girls, but she hasn't notice
d Sumika's feelings and has always been refused by other girls.
1
A music video for T.M.Revolution's song "Committed RED". \r\n \r\nThe concept o
f the video is "the song of a man who crosses through the ages and fights" and
was animated as a work about "a man who continues his journey of fighting even
though he hesitatingly hurts others, and even though society and a normal every
day life have tossed him away." \r\n \r\n(Source: ANN)
1
Mars  is the 1st remake of the franchise and restored the original manga's stor
yline of a young boy in suspended animation inside a South Sea volcano who awak
ens a century before his alien masters planned to use his powers to reduce huma
n civilization to a manageable low-tech level.  \r\n \r\n(Source: The Anime Enc
yclopedia)
1
U.A. High School's students of Class 1-A have made it to summer break. Izuku Mi
doriya accompanies his mentor All Might to a celebratory superhero festival on
I-Island, an isolated patch of land dedicated to researching Quirks and everyth
ing else associated with the hero business. Midoriya is granted the opportunity
to meet All Might's friend Dave and Dave's daughter Melissa, two talented hero
equipment engineers. He also encounters his classmates, most of whom have been
given the opportunity to spend part of their summer break at the festival. \r\n
\r\nHowever, a mysterious squad of villains infiltrates I-Island, and it is up
to Midoriya and his friends to confront them, using their developing Quirks to
fight off the new enemy and uncover a treacherous plot. \r\n \r\n[Written by MA
L Rewrite]     1
Two doctors traversing through the memories of a dying man to fulfill his last
wish.  \r\n \r\n(Source: Freebird Games)
1
Name: Synopsis, Length: 12732, dtype: int64
```

In [42]:
```
# value counts for the Type column
get_value_counts(df_anime,'Type')
```

Out[42]:
```
TV         3638
OVA        3176
Movie      2297
Special    1798
ONA        1050
Music      1019
Unknown       1
Name: Type, dtype: int64
```

In [43]:
```
# value_counts for source column
df_anime['Source'].value_counts()
```

Out[43]:
```
Unknown          3494
Original         3319
Manga            2750
Visual novel      750
Game              549
Light novel       487
Other             397
Music             344
Novel             321
4-koma manga      196
Web manga         137
Picture book       92
Book               85
Card game          41
Radio               9
Digital manga       8
Name: Source, dtype: int64
```

In [44]:
```
df_anime['Start_year'] = df_anime['Aired'].apply(lambda x : x.split('to')[0]).app
                         lambda x : x.split(',')).apply(
                         lambda x : x[1] if len(x) > 1 else x[0] ).apply(
                         lambda x : x.replace(' ', ''))
df_anime.head()
```

Out[44]:

	Anime_id	Title	Genre	Synopsis	Type	Producer	Studio	Rating	Score
0	1	Cowboy Bebop	['Action', 'Adventure', 'Comedy', 'Drama', 'Sc...	In the year 2071, humanity has colonized sever...	TV	['Bandai Visual']	['Sunrise']	8.81	3638
1	5	Cowboy Bebop Tengoku no Tobira	['Action', 'Space', 'Drama', 'Mystery', 'Sci-Fi']	Another day, another bounty—such is the life o...	Movie	['Sunrise', 'Bandai Visual']	['Bones']	8.41	111
2	6	Trigun	['Action', 'Sci-Fi', 'Adventure', 'Comedy', 'D...	Vash the Stampede is the man with a $$60,000,0...	TV	['Victor Entertainment']	['Madhouse']	8.31	1974
3	7	Witch Hunter Robin	['Action', 'Magic', 'Police', 'Supernatural', ...	Witches are individuals with special powers li...	TV	['Bandai Visual']	['Sunrise']	7.34	318
4	8	Bouken Ou Beet	['Adventure', 'Fantasy', 'Shounen', 'Supernatu...	It is the dark century and the people are suff...	TV	Unknown	['Toei Animation']	7.04	47

In [45]:
```
df_anime['End_year'] = df_anime['Aired'].apply(lambda x : x.split('to')[-1]).appl
                           lambda x : x.split(',')).apply(
                           lambda x : x[1] if len(x) > 1 else x[0] ).apply(
                           lambda x : x.replace(' ', ''))
df_anime.head()
```

Out[45]:

	Anime_id	Title	Genre	Synopsis	Type	Producer	Studio	Rating	Score
0	1	Cowboy Bebop	['Action', 'Adventure', 'Comedy', 'Drama', 'Sc...	In the year 2071, humanity has colonized sever...	TV	['Bandai Visual']	['Sunrise']	8.81	3638
1	5	Cowboy Bebop Tengoku no Tobira	['Action', 'Space', 'Drama', 'Mystery', 'Sci-Fi']	Another day, another bounty—such is the life o...	Movie	['Sunrise', 'Bandai Visual']	['Bones']	8.41	111
2	6	Trigun	['Action', 'Sci-Fi', 'Adventure', 'Comedy', 'D...	Vash the Stampede is the man with a $$60,000,0...	TV	['Victor Entertainment']	['Madhouse']	8.31	1974
3	7	Witch Hunter Robin	['Action', 'Magic', 'Police', 'Supernatural', ...	Witches are individuals with special powers li...	TV	['Bandai Visual']	['Sunrise']	7.34	318
4	8	Bouken Ou Beet	['Adventure', 'Fantasy', 'Shounen', 'Supernatu...	It is the dark century and the people are suff...	TV	Unknown	['Toei Animation']	7.04	47

In [46]:
```
df_anime['Start_year'].unique()
```

Out[46]:
```
array(['1998', '2001', '2002', '2004', '2005', '2003', '1995', '1997',
       '1999', '1996', '1988', '1993', '2000', '1979', '1989', '1991',
       '1985', '1986', '1994', '1992', '1990', '1978', '1973', '2006',
       '1987', '1984', '1982', '1983', '1980', '1976', '1968', '1977',
       '1981', '2007', '1971', '1967', '1975', '1962', '1965', '1969',
       '1974', '1964', '2008', '1972', '1970', '1966', '1963', '1945',
       '2009', '2020', '1933', '1929', '1943', '2010', '1931', '1934',
       '1960', '1958', '2011', '1959', '1930', '1928', '1947', '2012',
       '1932', '1936', '1917', '1935', '1938', '1939', '1941', '1942',
       '1948', '1950', '1957', '1961', '1918', '1924', '1925', '1926',
       '1927', '1940', '1944', '1946', '1952', '1953', '1954', '2016',
       '2013', '2019', '2014', '2015', '2017', 'Notavailable', '1949',
       '1955', '2018', '1937', '1951', '1956', '2021'], dtype=object)
```

In [47]:
```
df_anime['End_year'].unique()
```

Out[47]:
```
array(['1999', '2001', '1998', '2002', '2005', '2008', '2003', '2006',
       '2007', '2004', '1996', '1997', '1995', '1988', '1994', '2000',
       '1980', '1989', '1992', '1986', '1987', '1991', '2012', '1979',
       '1978', '1973', '1974', '1990', '1985', '1984', '2011', '1983',
       '1993', '2010', '1981', '1977', '1968', '1982', '1972', '1975',
       '1962', '2009', '1967', '1971', '1966', '1976', '2017', '1964',
       '1965', '1969', '1970', '1945', '2020', '1963', '1933', '1929',
       '1943', '1931', '1934', '1960', '1958', '1959', '2015', '1930',
       '1928', '2014', '1947', '1932', '1936', '1917', '2013', '1935',
       '1938', '1939', '1941', '1942', '1948', '1950', '1957', '1961',
       '1918', '1924', '1925', '1926', '1927', '1940', '1944', '1946',
       '1952', '1953', '1954', '2016', '2019', '2018', '?',
       'Notavailable', '1949', '1955', '1937', '1951', '1956', '2021'],
      dtype=object)
```

In [48]:
```
df_anime['Start_year'].replace({'Notavailable':0, " ": 0, '?': 0 }, inplace= True
df_anime['End_year'].replace({'Notavailable':0, " ": 0, '?': 0 }, inplace= True)
```

In [49]:
```
df_anime['Start_year'].unique()
```

Out[49]:
```
array(['1998', '2001', '2002', '2004', '2005', '2003', '1995', '1997',
       '1999', '1996', '1988', '1993', '2000', '1979', '1989', '1991',
       '1985', '1986', '1994', '1992', '1990', '1978', '1973', '2006',
       '1987', '1984', '1982', '1983', '1980', '1976', '1968', '1977',
       '1981', '2007', '1971', '1967', '1975', '1962', '1965', '1969',
       '1974', '1964', '2008', '1972', '1970', '1966', '1963', '1945',
       '2009', '2020', '1933', '1929', '1943', '2010', '1931', '1934',
       '1960', '1958', '2011', '1959', '1930', '1928', '1947', '2012',
       '1932', '1936', '1917', '1935', '1938', '1939', '1941', '1942',
       '1948', '1950', '1957', '1961', '1918', '1924', '1925', '1926',
       '1927', '1940', '1944', '1946', '1952', '1953', '1954', '2016',
       '2013', '2019', '2014', '2015', '2017', 0, '1949', '1955', '2018',
       '1937', '1951', '1956', '2021'], dtype=object)
```

In [50]:
```
df_anime['End_year'].unique()
```

Out[50]:
```
array(['1999', '2001', '1998', '2002', '2005', '2008', '2003', '2006',
       '2007', '2004', '1996', '1997', '1995', '1988', '1994', '2000',
       '1980', '1989', '1992', '1986', '1987', '1991', '2012', '1979',
       '1978', '1973', '1974', '1990', '1985', '1984', '2011', '1983',
       '1993', '2010', '1981', '1977', '1968', '1982', '1972', '1975',
       '1962', '2009', '1967', '1971', '1966', '1976', '2017', '1964',
       '1965', '1969', '1970', '1945', '2020', '1963', '1933', '1929',
       '1943', '1931', '1934', '1960', '1958', '1959', '2015', '1930',
       '1928', '2014', '1947', '1932', '1936', '1917', '2013', '1935',
       '1938', '1939', '1941', '1942', '1948', '1950', '1957', '1961',
       '1918', '1924', '1925', '1926', '1927', '1940', '1944', '1946',
       '1952', '1953', '1954', '2016', '2019', '2018', 0, '1949', '1955',
       '1937', '1951', '1956', '2021'], dtype=object)
```

- This is cleaner
- Now lets create a new column named `'YearsAired'` that is a difference of the two.

In [51]:
```
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12979 entries, 0 to 15064
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    12979 non-null  int64  
 1   Title       12979 non-null  object 
 2   Genre       12979 non-null  object 
 3   Synopsis    12979 non-null  object 
 4   Type        12979 non-null  object 
 5   Producer    12979 non-null  object 
 6   Studio      12979 non-null  object 
 7   Rating      12979 non-null  float64
 8   ScoredBy    12979 non-null  float64
 9   Popularity  12979 non-null  float64
 10  Members     12979 non-null  float64
 11  Episodes    12979 non-null  float64
 12  Source      12979 non-null  object 
 13  Aired       12979 non-null  object 
 14  Link        12979 non-null  object 
 15  Start_year  12979 non-null  object 
 16  End_year    12979 non-null  object 
dtypes: float64(5), int64(1), object(11)
memory usage: 1.8+ MB
```

In [52]:
```
#first lets convert the datatype to integer
df_anime['End_year']=df_anime['End_year'].values.astype(np.int64)
df_anime['Start_year']=df_anime['Start_year'].values.astype(np.int64)
df_anime['YearsAired']= df_anime['End_year']- df_anime['Start_year']
```

In [53]:
```
df_anime['YearsAired'].unique()
```

Out[53]:
```
array([    1,     0,     3,     2,     5,     4,     8,     7,     6,
          18,    10,    13,    26,     9,    12,    15,    19, -2015,
       -2012, -2006, -2013,    20, -2016, -1999, -2014, -2011, -2017,
       -1993, -2007, -2020, -2009, -2008, -2010, -2018, -2019],
      dtype=int64)
```

In [54]:
```
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12979 entries, 0 to 15064
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    12979 non-null  int64  
 1   Title       12979 non-null  object 
 2   Genre       12979 non-null  object 
 3   Synopsis    12979 non-null  object 
 4   Type        12979 non-null  object 
 5   Producer    12979 non-null  object 
 6   Studio      12979 non-null  object 
 7   Rating      12979 non-null  float64
 8   ScoredBy    12979 non-null  float64
 9   Popularity  12979 non-null  float64
 10  Members     12979 non-null  float64
 11  Episodes    12979 non-null  float64
 12  Source      12979 non-null  object 
 13  Aired       12979 non-null  object 
 14  Link        12979 non-null  object 
 15  Start_year  12979 non-null  int64  
 16  End_year    12979 non-null  int64  
 17  YearsAired  12979 non-null  int64  
dtypes: float64(5), int64(4), object(9)
memory usage: 1.9+ MB
```

In [55]:
```
# value_counts for Link column
df_anime['Link'].value_counts()
```

Out[55]:
```
https://myanimelist.net/anime/18055/Hakkenden__Touhou_Hakken_Ibun_2nd_Season (h
ttps://myanimelist.net/anime/18055/Hakkenden__Touhou_Hakken_Ibun_2nd_Season)
2
https://myanimelist.net/anime/324/Kidou_Keisatsu_Patlabor__On_Television (http
s://myanimelist.net/anime/324/Kidou_Keisatsu_Patlabor__On_Television)        2
https://myanimelist.net/anime/36005/Mainichi_JK_Kikaku_Episode_0 (https://myani
melist.net/anime/36005/Mainichi_JK_Kikaku_Episode_0)                 2
https://myanimelist.net/anime/1281/Gakkou_no_Kaidan (https://myanimelist.net/an
ime/1281/Gakkou_no_Kaidan)                                     2
https://myanimelist.net/anime/6903/Prima_Donna_Mai (https://myanimelist.net/ani
me/6903/Prima_Donna_Mai)                                       2

 ..
https://myanimelist.net/anime/309/Akahori_Gedou_Hour_Rabuge (https://myanimelis
t.net/anime/309/Akahori_Gedou_Hour_Rabuge)                         1
https://myanimelist.net/anime/4014/Choujin_Locke__Lord_Leon (https://myanimelis
t.net/anime/4014/Choujin_Locke__Lord_Leon)                         1
https://myanimelist.net/anime/39001/Light_Speed_Days (https://myanimelist.net/a
nime/39001/Light_Speed_Days)                                   1
https://myanimelist.net/anime/36904/Aggressive_Retsuko_ONA (https://myanimelis
t.net/anime/36904/Aggressive_Retsuko_ONA)                          1
https://myanimelist.net/anime/19125/Mogura_no_Adventure (https://myanimelist.ne
t/anime/19125/Mogura_no_Adventure)                               1
Name: Link, Length: 12943, dtype: int64
```

In [56]:
```
# previewing the columns
df_anime.columns
```

Out[56]:
```
Index(['Anime_id', 'Title', 'Genre', 'Synopsis', 'Type', 'Producer', 'Studio',
       'Rating', 'ScoredBy', 'Popularity', 'Members', 'Episodes', 'Source',
       'Aired', 'Link', 'Start_year', 'End_year', 'YearsAired'],
      dtype='object')
```

The `Aired` and `Link` columns will not be needed in the next parts of our project, therefore we will drop the columns.

In [57]:
```
# dropping columns
df_anime.drop(columns = ['Aired', 'Link', 'Start_year', 'End_year'], axis=1, inpl
```

In [58]:
```
# preview of data info again
df_anime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12979 entries, 0 to 15064
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Anime_id    12979 non-null  int64  
 1   Title       12979 non-null  object 
 2   Genre       12979 non-null  object 
 3   Synopsis    12979 non-null  object 
 4   Type        12979 non-null  object 
 5   Producer    12979 non-null  object 
 6   Studio      12979 non-null  object 
 7   Rating      12979 non-null  float64
 8   ScoredBy    12979 non-null  float64
 9   Popularity  12979 non-null  float64
 10  Members     12979 non-null  float64
 11  Episodes    12979 non-null  float64
 12  Source      12979 non-null  object 
 13  YearsAired  12979 non-null  int64  
dtypes: float64(5), int64(2), object(7)
memory usage: 1.5+ MB
```

In [59]:
```
df_anime.to_csv("cleaned_anime.csv")
```

Our Anime data is now clean and can be used in the next phases of the project.

We can now go ahead and clean the ratings data.

3.2 Ratings Data

3.2.1 Completeness

In [60]:
```
# missing values
missing_values(df_rating)
```

Out[60]:

	Missing Values	In Percentage

3.2.2 Validity

Here we will check for;

- Duplicates
- Outliers

In [61]:
```
# checking for duplicates
df_rating.duplicated().sum()
```

Out[61]: 1

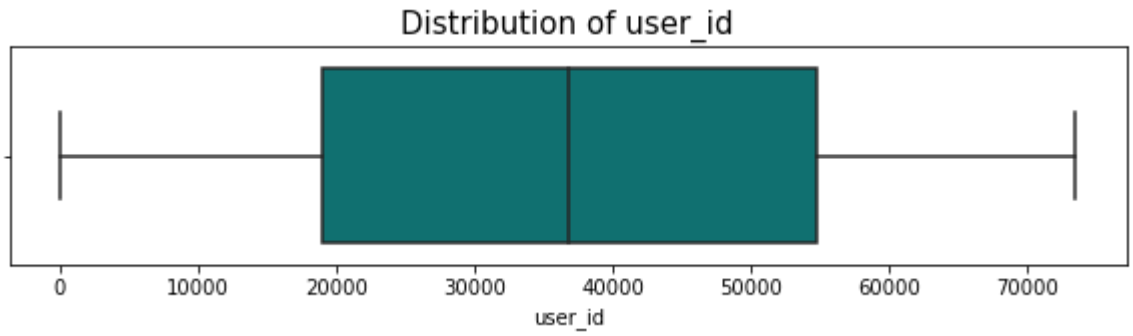
The rating data has only one duplicated record therefore we will just drop it.

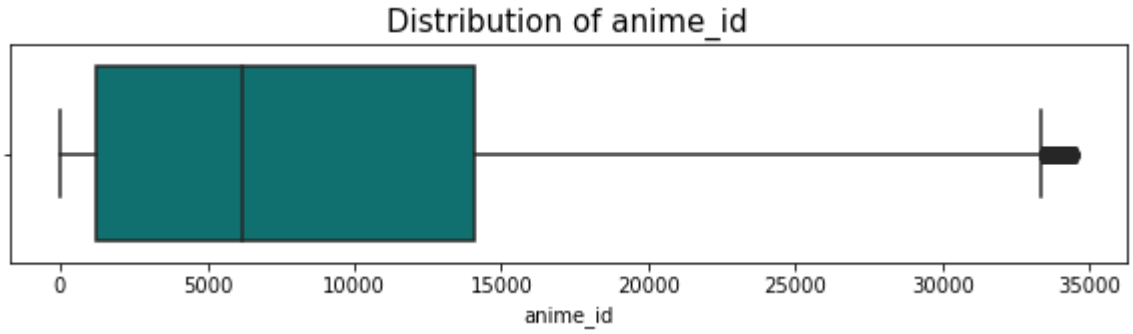
In [62]:
```
# dropping duplicates
df_rating.drop_duplicates(inplace = True)
```

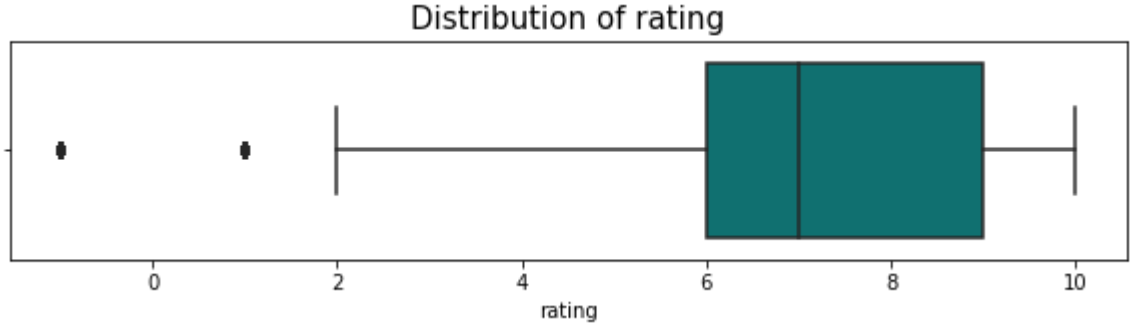
In [63]:
```
# confirming there are no duplicates
df_rating.duplicated().sum()
```

Out[63]: 0

In [64]:
```
# checking for outliers

column_list = ["user_id", "anime_id", "rating"]

for col in column_list:
    outliers(df_rating, col)
```







- There are outliers in the `anime_id` column. There is no need to remove them since the ids are just a unique identifier for an anime title.
- There are also outliers in the rating column. The boxplot shows ratings of -1. There is no need to remove them because a rating of -1 means that a user did not give the anime a rating therefore the outliers are genuine.

3.2.3 Consistency

In [65]:
```
# value counts for columns in the ratings data

for col in column_list:
    print(get_value_counts(df_rating, col))
```

```
48766    10227
42635     3747
53698     2905
57620     2702
59643     2633
         ...
60103        1
72442        1
15070        1
33749        1
5846         1
Name: user_id, Length: 73515, dtype: int64
1535     39340
11757    30583
16498    29583
1575     27718
226      27506
         ...
29716        1
20145        1
26229        1
21019        1
21509        1
Name: anime_id, Length: 11200, dtype: int64
 8     1646018
-1     1476496
 7     1375287
 9     1254096
 10     955715
 6      637775
 5      282806
 4      104291
 3       41453
 2       23150
 1       16649
Name: rating, dtype: int64
```

In [66]:
```
# preview of the data info again.
df_rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7813736 entries, 0 to 7813736
Data columns (total 3 columns):
 #   Column    Dtype
---  ------    -----
 0   user_id   int64
 1   anime_id  int64
 2   rating    int64
dtypes: int64(3)
memory usage: 238.5 MB
```

In [67]:
```
#saving cleaned dataset
df_rating.to_csv("cleaned_rating.csv")
```

The ratings data is now clean and can be used in the next phases of our project.

4. EXPLORATORY DATA ANALYSIS

In this section, we will be using the `Anime data` to do EDA.

In [68]:
```
df= df_anime
episode_per_season = df.Title.value_counts(sort=False)
episode_per_season
```

Out[68]:
```
Elmer no Bouken My Fathers Dragon        1
                                         1
Yukiguni no Oujisama                     1
Asarichan                                1
Toaru Majutsu no Index                   1
                                        ..
Tokimeki Memorial Forever With You       1
Loveless Specials                        1
Pokemon Movie 01 Mewtwo no Gyakushuu     1
Fatekaleid liner PrismaIllya Specials    1
Uchouten Kazoku 2                        1
Name: Title, Length: 12909, dtype: int64
```

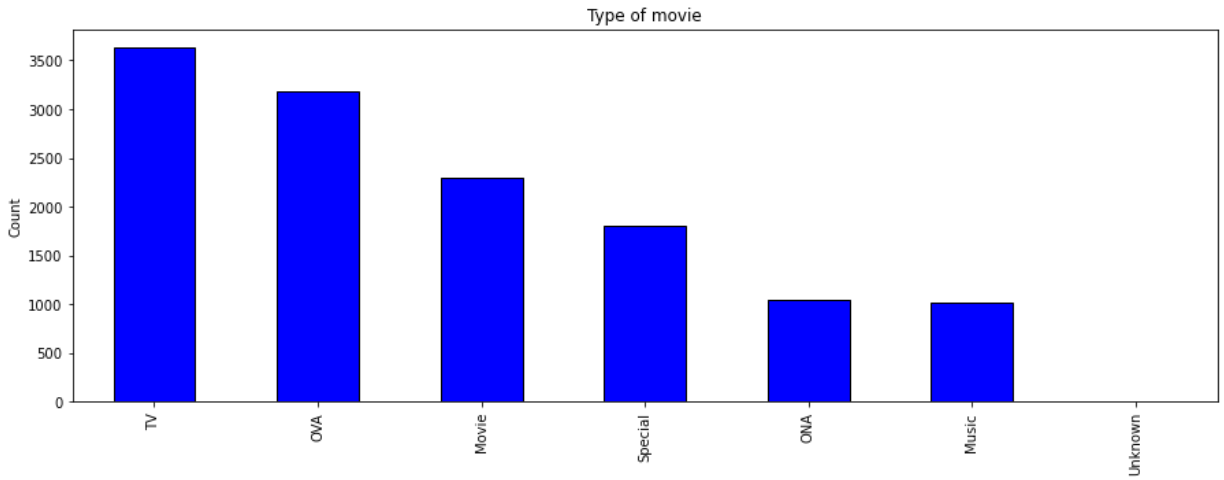
In [69]:
```
top_10_rated_episodes = df.sort_values("Rating", ascending = False).head(10)
top_10_rated_episodes.Title
```

Out[69]:
```
12401    Manichi ga Tsurai Kimochi Wakarimasu ka Yuruse...
11721    Kuunyan no Koutsuu Anzen Tadashii Jitensha no ...
11654                  Kyouiku Eigasai Jushou Anime Series
14510                                       Ling Feng Zhe
11734                                          Ittekimasu
13031                                       Xiao Hua Xian
11598                                        Hashire John
3634                      Fullmetal Alchemist Brotherhood
8283                                    Okaachan Gomen ne
10110                                       Kimi no Na wa
Name: Title, dtype: object
```

4.1 Univariate

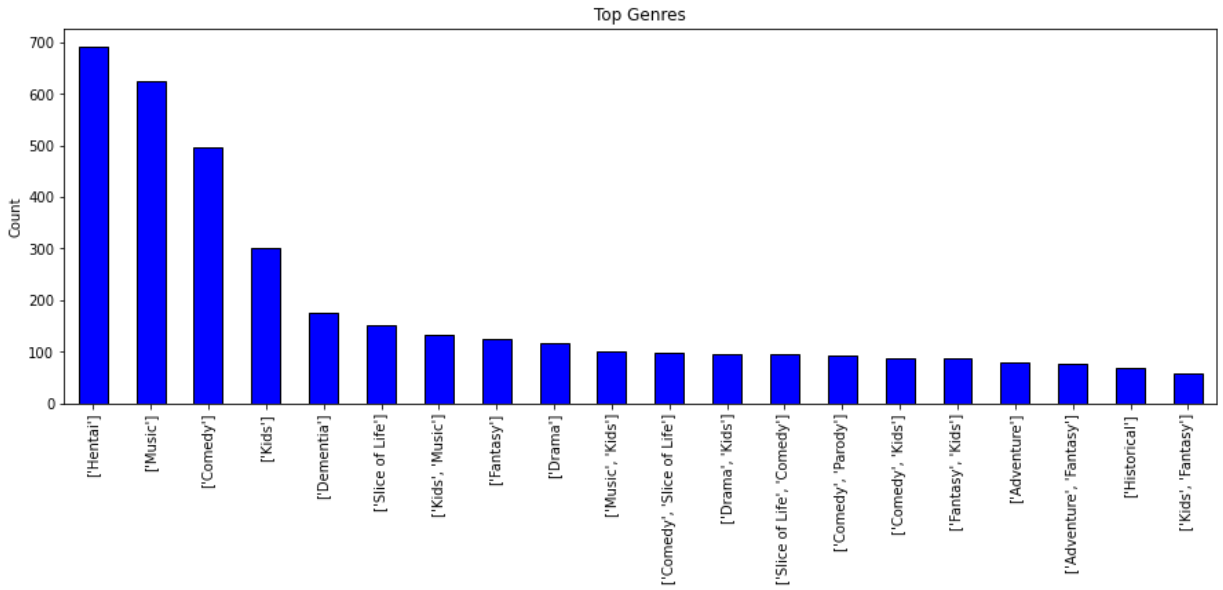
In [70]:
```
# Function to visualise the the the top 20 value counts in the columns
def plot_data(df, col, title):

    fig, ax = plt.subplots(figsize=(10, 5))

    ''' Plots the value counts of the top 20 categories of a column in a datafram

    get_value_counts(df, col).head(20).plot(kind='bar', figsize=(15, 5), color='#
    plt.title(title)
    plt.xticks(rotation=90)
    plt.ylabel('Count', fontsize=10)
```

In [71]:
```
plot_data(df,"Type",'Type of movie')
```



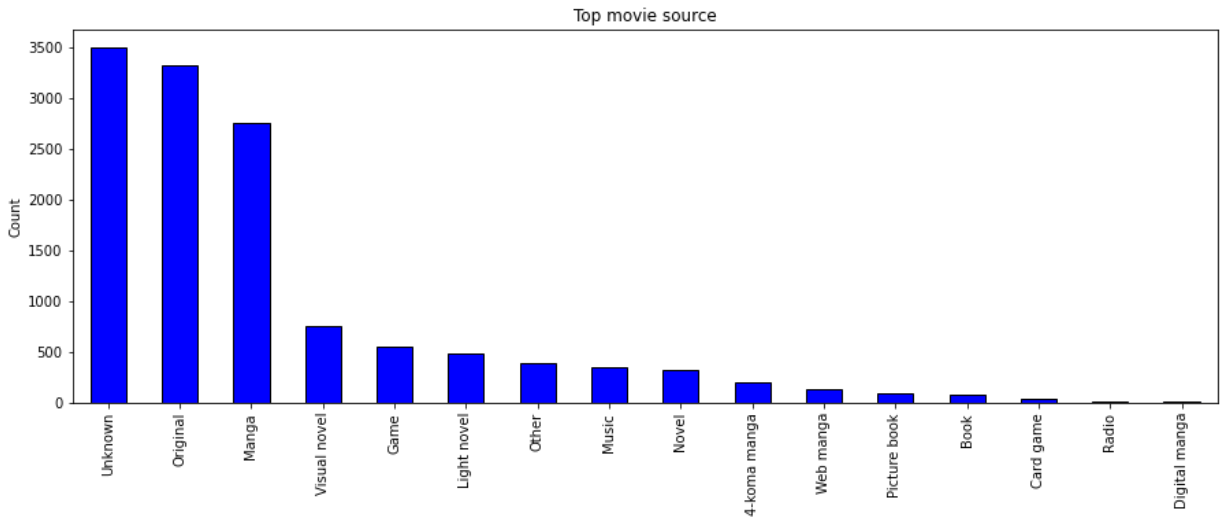
Animes aired on TV are the most watched

In [72]:
```
plot_data(df,"Genre",'Top Genres')
```



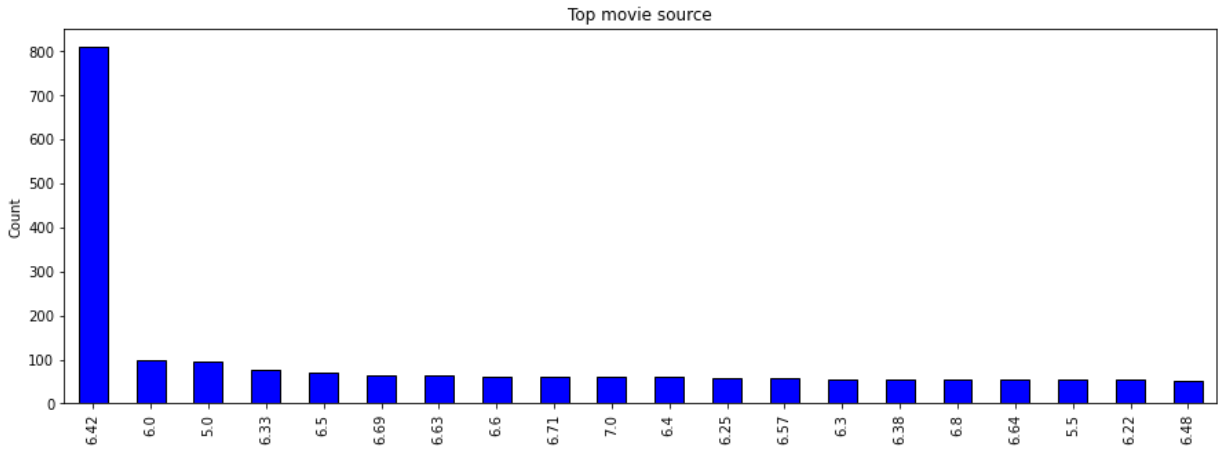
The top genre is Hental

In [73]:
```
plot_data(df,"Source",'Top movie source')
```



Most of the anime movies are adapted/ sourced from unknown sources, followed by original sources and manga.

In [74]:
```
plot_data(df,"Rating",'Top movie source')
```

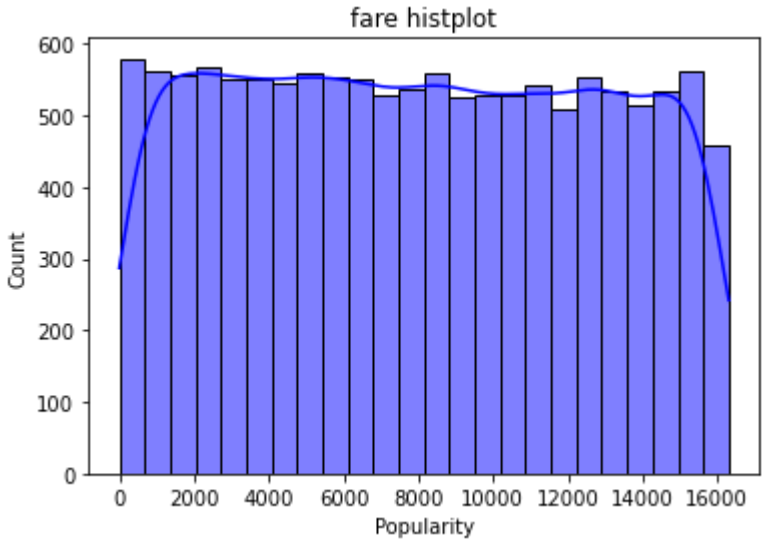


Most movies had a rating of 6.0

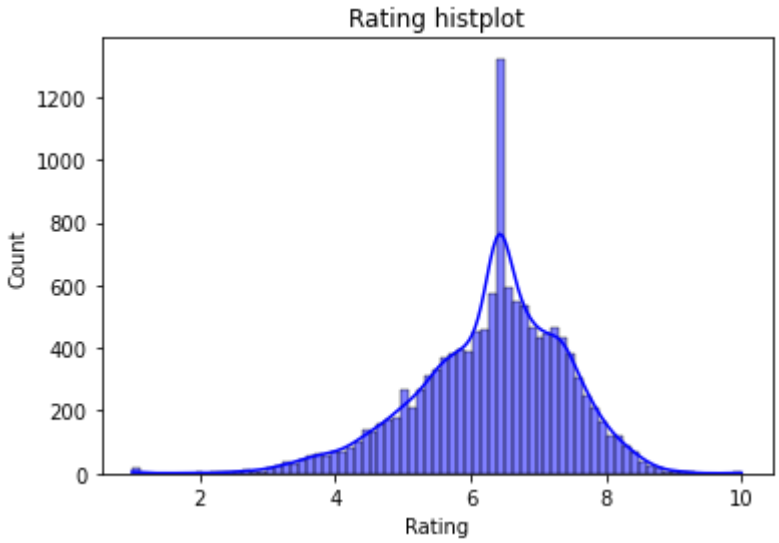
In [75]:
```
def value_counts(df, col):
    ''' Returns the value counts of a column in a dataframe '''
    counts = df[col].value_counts(dropna=False, ascending=False)
    return counts
```

In [76]:
```
def draw_data(df, col, title):

    fig, ax = plt.subplots(figsize=(10, 5))

    ''' Plots the value counts of the top 20 categories of a column in a datafram

    value_counts(df, col).head(20).plot(kind='hist', figsize=(15, 5), color='gree
    plt.title(title)
    plt.xticks(rotation=90)
    plt.ylabel('Count', fontsize=10)
```

In [77]:
```
plt.plot(figsize= (10,30))
sns.histplot(df, x = "Popularity", bins="auto", kde= True, color= "#0000FF")
plt.title("fare histplot")
plt.show()
```



In [78]:
```
plt.plot(figsize= (10,30))
sns.histplot(df, x = "Rating", bins="auto", kde= True, color= "#0000FF")
plt.title("Rating histplot")
plt.show()
```



4.2 Bivariate

Source vs Rating

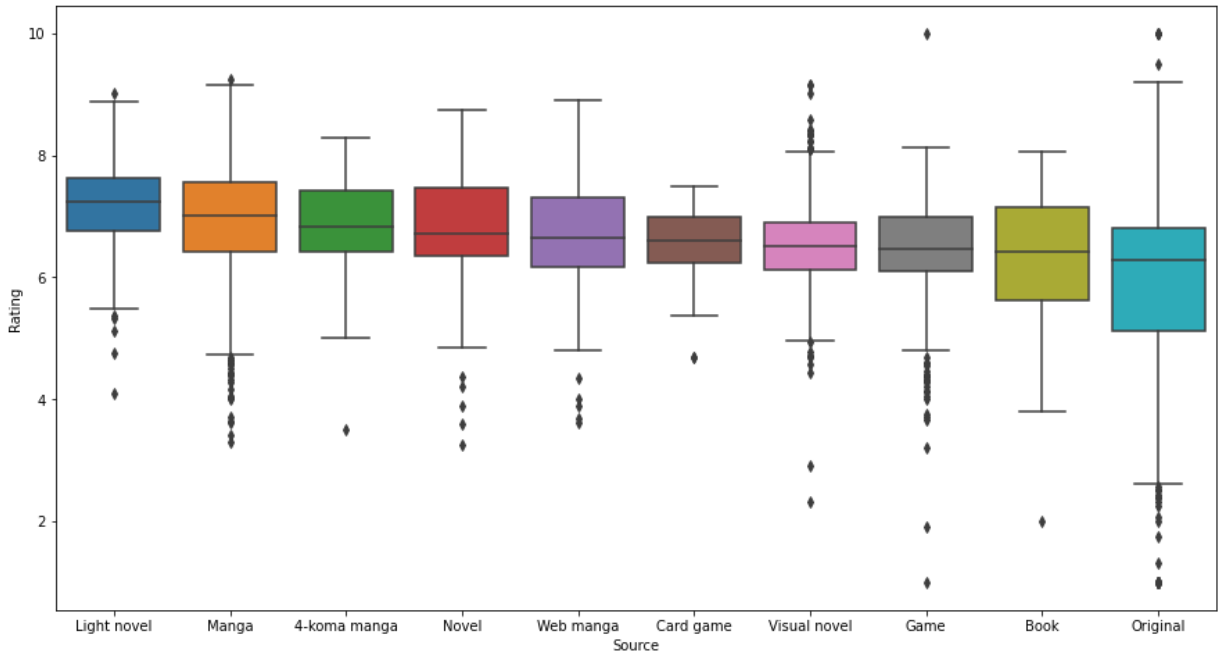
In [79]:
```
#TOP 10 OF SOURCES THAT ARE MOST HIGHLY VALUED
source_score__median_top10  = df_anime.groupby(['Source'] ,
                                    as_index=False).agg({'Rating' : 'median'}).sort
                                  ascending = False).head(10)

source_score__median_top10
```

Out[79]:

	Source	Rating
5	Light novel	7.230
6	Manga	7.000
0	4-koma manga	6.825
8	Novel	6.720
15	Web manga	6.640
2	Card game	6.610
14	Visual novel	6.520
4	Game	6.460
1	Book	6.420
9	Original	6.290

In [80]:
```
#BOXPLOT TOP10 SOURCE
plt.figure(figsize=(15,8))
sns.boxplot(x= 'Source' , y = 'Rating', data = df_anime , order= source_score__me
```

Out[80]: <AxesSubplot:xlabel='Source', ylabel='Rating'>



Producer vs Rating

In [81]:
```
# TOP 10 RATED Producers

Producer__median_top10  = df_anime.groupby(['Producer'] ,
                                    as_index=False).agg({'Rating' : 'median'}).sort
                                  ascending = False).head(10)

Producer__median_top10
```

Out[81]:

	Producer	Rating
258	['Aniplex', 'Square Enix', 'Mainichi Broadcast...	9.25
1236	['Kadokawa Shoten', 'Toho', 'Sound Team Don Ju...	9.19
2358	['TV Tokyo', 'Aniplex', 'Dentsu']	9.15
2357	['TV Tokyo', 'Aniplex', 'Dentsu', 'Trinity Sou...	9.11
2353	['TV Tokyo', 'Aniplex', 'Dentsu', 'Shueisha', ...	9.07
1940	['Shochiku', 'Pony Canyon', 'Kodansha', 'ABC A...	9.04
2356	['TV Tokyo', 'Aniplex', 'Dentsu', 'Trinity Sou...	9.01
1812	['Pony Canyon', 'TBS', 'Rakuonsha', 'Animation...	9.01
472	['Bandai Visual', 'Mainichi Broadcasting Syste...	8.95
2309	['TOHO animation', 'Shueisha']	8.93

In [82]:
```
Most_episodes__top10  = df_anime.groupby(['Title'] ,
                                    as_index=False).agg({'Episodes' : 'median'}).so
                                  ascending = False).head(10)
Most_episodes__top10
```

Out[82]:

	Title	Episodes
8604	Oyako Club	1818.0
2391	Doraemon 1979	1787.0
6901	Manga Nippon Mukashibanashi 1976	1471.0
4324	Hoka Hoka Kazoku	1428.0
7462	Monoshiri Daigaku Ashita no Calendar	1274.0
9827	Sekai Monoshiri Ryoko	1006.0
5961	Kotowaza House	773.0
10042	Shima Shima Tora no Shimajirou	726.0
7972	Ninja Hattorikun	694.0
8746	Perman 1983	526.0

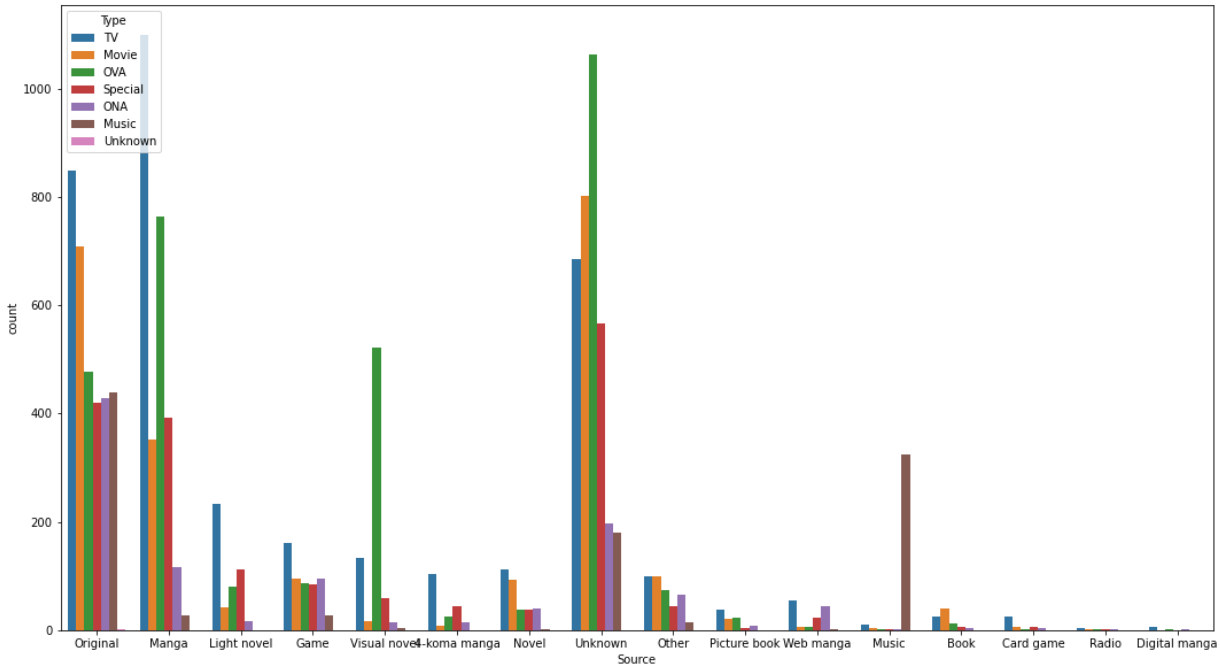
Title vs ScoredBy

In [83]:
```
# TOP Most scored animes

scored__top10  = df_anime.groupby(['Title'] ,
                                    as_index=False).agg({'ScoredBy' : 'median'}).sc
                                  ascending = False).head(10)
scored__top10
```

Out[83]:

	Title	ScoredBy
2143	Death Note	1006242.0
10132	Shingeki no Kyojin	936784.0
10844	Sword Art Online	913806.0
3000	Fullmetal Alchemist Brotherhood	730784.0
8336	One Punch Man	687965.0
11453	Tokyo Ghoul	656039.0
7747	Naruto	645672.0
429	Angel Beats	640177.0
1812	Code Geass Hangyaku no Lelouch	625466.0
8028	No Game No Life	620456.0

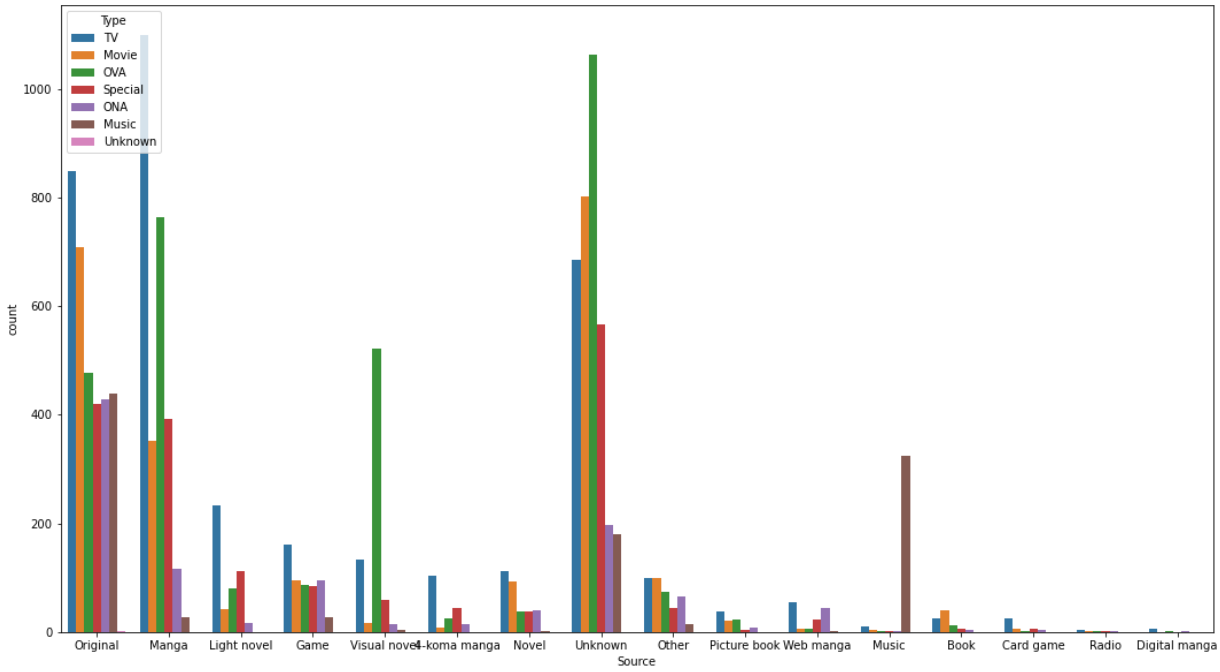
In [85]:
```
plt.figure(figsize=(18,10))
ax = sns.countplot(x=df_anime['Source'], hue=df_anime["Type"], data=scored__top10
```



From the above we can see the Original and Manga studios air most of their animes on TV

Type vs Rating

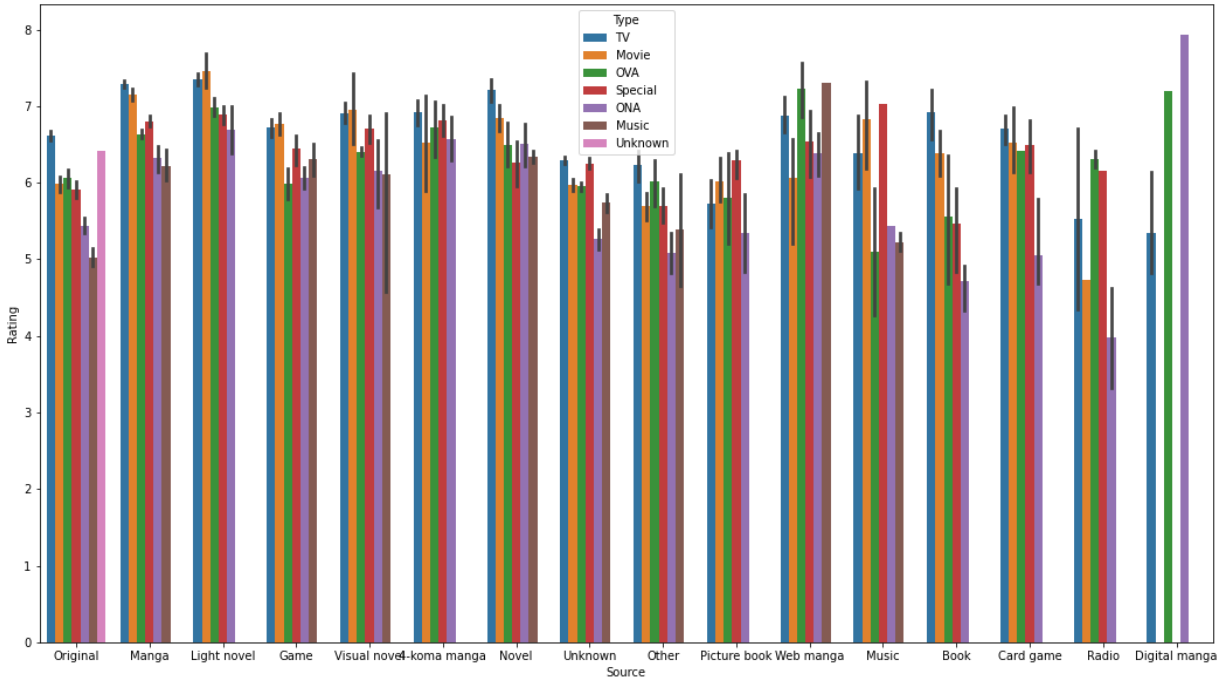
In [86]:
```
plt.figure(figsize=(18,10))
ax = sns.countplot(x='Source', hue="Type", data= df_anime)
```



4.3 Multivariate

In [87]:
```
plt.figure(figsize=(18,10))
sns.barplot(data=df_anime, x="Source", y="Rating", hue="Type")
```

Out[87]:
```
<AxesSubplot:xlabel='Source', ylabel='Rating'>
```



In [88]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True)
```

Out[88]: <AxesSubplot:>

- A negative valued correlation implies negative correlatation
- A posiitive valued correlation implies positive correlatation
- A correlation of 1 implies full similarity.
- The closer the the correlation is to 1 the stronger the relationship
- `ScoredBy` and `Members` columns are the most correlated

Modelling

For the modelling part, we will do it in the `Modelling` notebook.