In [94]:
```python
import pandas as pd
```

We load the anime dataset

In [95]:
```python
anime_df = pd.read_csv('/content/drive/MyDrive/DIC-Anime-Recommendation/Data
```

In [96]:
```python
anime_df.head()
```

Out[96]:

	anime_id	Name	English name	Other name	Score	Genres	Synopsis	Type
0	1	Cowboy Bebop	Cowboy Bebop	カウボーイビバップ	8.75	Action, Award Winning, Sci-Fi	Crime is timeless. By the year 2071, humanity ...	TV
1	5	Cowboy Bebop: Tengoku no Tobira	Cowboy Bebop: The Movie	カウボーイビバップ天国の扉	8.38	Action, Sci-Fi	Another day, another bounty—such is the life o...	Movie
2	6	Trigun	Trigun	トライガン	8.22	Action, Adventure, Sci-Fi	Vash the Stampede is the man with a $$60,000,0...	TV
3	7	Witch Hunter Robin	Witch Hunter Robin	Witch Hunter ROBIN (ウイッチハンターロビン)	7.25	Action, Drama, Mystery, Supernatural	Robin Sena is a powerful craft user drafted in...	TV
4	8	Bouken Ou Beet	Beet the Vandel Buster	冒険王ビィト	6.94	Adventure, Fantasy, Supernatural	It is the dark century and the people are suff...	TV

5 rows × 24 columns

The **Aired** attribute is very important for us. Our target is to extract information like how long an anime runs, is it still ongoing, how many episodes does it have

Preprocessing step 1: The aim is to extract start date and end date of an anime and add those 2 as new columns to the dataframe

I split the date using the word **to**

In [97]:
```
aired = anime_df['Aired'].str.split('to', expand=True)
```

Then strip whitespaces

In [98]:
```
aired[0] = aired[0].str.strip()
aired[1] = aired[1].str.strip()
```

In [99]:
```
aired
```

Out[99]:

	0	1
0	Apr 3, 1998	Apr 24, 1999
1	Sep 1, 2001	None
2	Apr 1, 1998	Sep 30, 1998
3	Jul 3, 2002	Dec 25, 2002
4	Sep 30, 2004	Sep 29, 2005
...	...	...
24900	Jul 4, 2023	?
24901	Jul 27, 2023	?
24902	Jul 19, 2023	?
24903	Apr 23, 2022	None
24904	Sep 5, 2022	None

24905 rows × 2 columns

Finally convert both Start date and end date to datetime objects

In [100…
```
aired[0] = pd.to_datetime(aired[0], format='%b %d, %Y', errors='coerce')
aired[1] = pd.to_datetime(aired[1], format='%b %d, %Y', errors='coerce')
```

In [101…
```
aired
```

Out[101…

	0	1
0	1998-04-03	1999-04-24
1	2001-09-01	NaT
2	1998-04-01	1998-09-30
3	2002-07-03	2002-12-25
4	2004-09-30	2005-09-29
...	...	...
24900	2023-07-04	NaT
24901	2023-07-27	NaT
24902	2023-07-19	NaT
24903	2022-04-23	NaT
24904	2022-09-05	NaT

24905 rows × 2 columns

Rename the clomns

In [102…

```
aired.rename(columns={0: 'Start Date', 1: 'End Date'}, inplace=True)
```

In [103…

```
aired
```

Out[103…

	Start Date	End Date
0	1998-04-03	1999-04-24
1	2001-09-01	NaT
2	1998-04-01	1998-09-30
3	2002-07-03	2002-12-25
4	2004-09-30	2005-09-29
...	...	...
24900	2023-07-04	NaT
24901	2023-07-27	NaT
24902	2023-07-19	NaT
24903	2022-04-23	NaT
24904	2022-09-05	NaT

24905 rows × 2 columns

Inserted the new columns to the original dataframe

In [104…
```
anime_df.insert(10, 'Start Date', aired['Start Date'])
anime_df.insert(11, 'End Date', aired['End Date'])
```

In [105…
```
anime_df.head()
```

Out[105…

	anime_id	Name	English name	Other name	Score	Genres	Synopsis	Type
0	1	Cowboy Bebop	Cowboy Bebop	カウボーイビバップ	8.75	Action, Award Winning, Sci-Fi	Crime is timeless. By the year 2071, humanity ...	TV
1	5	Cowboy Bebop: Tengoku no Tobira	Cowboy Bebop: The Movie	カウボーイビバップ天国の扉	8.38	Action, Sci-Fi	Another day, another bounty—such is the life o...	Movie
2	6	Trigun	Trigun	トライガン	8.22	Action, Adventure, Sci-Fi	Vash the Stampede is the man with a $$60,000,0...	TV
3	7	Witch Hunter Robin	Witch Hunter Robin	Witch Hunter ROBIN (ウイッチハンターロビン)	7.25	Action, Drama, Mystery, Supernatural	Robin Sena is a powerful craft user drafted in...	TV
4	8	Bouken Ou Beet	Beet the Vandel Buster	冒険王ビィト	6.94	Adventure, Fantasy, Supernatural	It is the dark century and the people are suff...	TV

5 rows × 26 columns

In [106…
```
anime_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24905 entries, 0 to 24904
Data columns (total 26 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   anime_id      24905 non-null  int64
 1   Name          24905 non-null  object
 2   English name  24905 non-null  object
 3   Other name    24905 non-null  object
 4   Score         24905 non-null  object
 5   Genres        24905 non-null  object
 6   Synopsis      24905 non-null  object
 7   Type          24905 non-null  object
 8   Episodes      24905 non-null  object
 9   Aired         24905 non-null  object
 10  Start Date    20090 non-null  datetime64[ns]
 11  End Date      9337 non-null   datetime64[ns]
 12  Premiered     24905 non-null  object
 13  Status        24905 non-null  object
 14  Producers     24905 non-null  object
 15  Licensors     24905 non-null  object
 16  Studios       24905 non-null  object
 17  Source        24905 non-null  object
 18  Duration      24905 non-null  object
 19  Rating        24905 non-null  object
 20  Rank          24905 non-null  object
 21  Popularity    24905 non-null  int64
 22  Favorites     24905 non-null  int64
 23  Scored By     24905 non-null  object
 24  Members       24905 non-null  int64
 25  Image URL     24905 non-null  object
dtypes: datetime64[ns](2), int64(4), object(20)
memory usage: 4.9+ MB
```

Preprocessing step 2: The aim is to addd a new cloumn named **Ongoing**. The way I do this is the aired column has format from start date to end date. The end date has ?. Hence the rows having ? are tagged as ongoing animes

In [107…
```
import numpy as np
```

In [108…
```
def check(value):
    return 1 if '?' in value else 0
```

In [109…
```
anime_df['Ongoing'] = anime_df['Aired'].apply(check)
```

In [110…
```
anime_df.loc[11,'Ongoing']
```

Out[110…
```
1
```

In [111…
```
anime_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24905 entries, 0 to 24904
Data columns (total 27 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   anime_id      24905 non-null  int64
 1   Name          24905 non-null  object
 2   English name  24905 non-null  object
 3   Other name    24905 non-null  object
 4   Score         24905 non-null  object
 5   Genres        24905 non-null  object
 6   Synopsis      24905 non-null  object
 7   Type          24905 non-null  object
 8   Episodes      24905 non-null  object
 9   Aired         24905 non-null  object
 10  Start Date    20090 non-null  datetime64[ns]
 11  End Date      9337 non-null   datetime64[ns]
 12  Premiered     24905 non-null  object
 13  Status        24905 non-null  object
 14  Producers     24905 non-null  object
 15  Licensors     24905 non-null  object
 16  Studios       24905 non-null  object
 17  Source        24905 non-null  object
 18  Duration      24905 non-null  object
 19  Rating        24905 non-null  object
 20  Rank          24905 non-null  object
 21  Popularity    24905 non-null  int64
 22  Favorites     24905 non-null  int64
 23  Scored By     24905 non-null  object
 24  Members       24905 non-null  int64
 25  Image URL     24905 non-null  object
 26  Ongoing       24905 non-null  int64
dtypes: datetime64[ns](2), int64(5), object(20)
memory usage: 5.1+ MB
```

Preprocessing step 3: The episodes field is also very important for us. We can infer whether people like short animes or long animes based on number of episodes. However some records of our dataset have "UNKNOWN" in the episodes field, this is because the anime is currently running. Just for analysis purpose, we assume all animes end on jan 01 2024 to get the episode count till that date, since each episode is released once in a week

In [112…

```
for index, row in anime_df.iterrows():
    if row['Episodes'] == 'UNKNOWN':
        anime_df.loc[index, 'Episodes'] = ((pd.to_datetime('Jan 01, 2024', f
```

In [113…

```
anime_df[anime_df['Episodes'] == 'UNKNOWN']
```

Out[113…

anime_id	Name	English name	Other name	Score	Genres	Synopsis	Type	Episodes

0 rows × 27 columns

In [114…
```
anime_df.loc[11]
```

Out[114…

	11
anime_id	21
Name	One Piece
English name	One Piece
Other name	ONE PIECE
Score	8.69
Genres	Action, Adventure, Fantasy
Synopsis	Gol D. Roger was known as the "Pirate King," t...
Type	TV
Episodes	1262.714286
Aired	Oct 20, 1999 to ?
Start Date	1999-10-20 00:00:00
End Date	NaT
Premiered	fall 1999
Status	Currently Airing
Producers	Fuji TV, TAP, Shueisha
Licensors	Funimation, 4Kids Entertainment
Studios	Toei Animation
Source	Manga
Duration	24 min
Rating	PG-13 - Teens 13 or older
Rank	55.0
Popularity	20
Favorites	198986
Scored By	1226493.0
Members	2168904
Image URL	https://cdn.myanimelist.net/images/anime/6/732...
Ongoing	1

dtype: object

Preprocessing Step 4: finally we normalize episodes field so that we can bring it to a common scale for comparing between different animes

We use MinMax normalization which shrinks the scale between 0 to 1. X_norm = (X−min(X))/(max(X)−min(X))

In [115…
```
anime_df['Episodes'] = anime_df['Episodes'].astype(float)
```

In [116…
```
anime_df['Episodes_Normalized'] = (anime_df['Episodes'] - anime_df['Episodes
```

In [117…
```
anime_df['Episodes_Normalized']
```

Out[117…

	Episodes_Normalized
0	0.008181
1	0.000000
2	0.008181
3	0.008181
4	0.016688
...	...
24900	0.004581
24901	0.005563
24902	0.004908
24903	0.000000
24904	0.000000

24905 rows × 1 columns

dtype: float64

Question 1: Do people like long animes or short animes?

This question is critical for an Anime Recommendation system. People's preference may differ based on interest

We frame the **hypothesis** as 'Long running animes have higher scores'

We will use 'Episode' column and 'Score' column. 'Episode' column had some 'UNKNOWN' values, which we have already preprocessed.

EDA 1

We plan to plot a scatter plot to check for trends and biases. We do necessary cleaning and feature selection

In [118…
```
target_df = anime_df[['Episodes', 'Score']]
```

In [119…

```
target_df.head()
```

Out[119…

	Episodes	Score
0	26.0	8.75
1	1.0	8.38
2	26.0	8.22
3	26.0	7.25
4	52.0	6.94

In [120…

```
target_df['Episodes'].unique()
```

```
Out[120… array([2.60000000e+01, 1.00000000e+00, 5.20000000e+01, 1.45000000e+02,
       2.40000000e+01, 7.40000000e+01, 2.20000000e+02, 1.26271429e+03,
       1.78000000e+02, 1.20000000e+01, 2.20000000e+01, 6.90000000e+01,
       2.50000000e+01, 4.00000000e+00, 9.40000000e+01, 5.00000000e+00,
       3.00000000e+00, 1.30000000e+01, 2.30000000e+01, 4.30000000e+01,
       6.00000000e+00, 5.00000000e+01, 4.70000000e+01, 5.10000000e+01,
       4.90000000e+01, 3.90000000e+01, 8.00000000e+00, 7.00000000e+00,
       7.50000000e+01, 6.20000000e+01, 1.40000000e+01, 4.40000000e+01,
       4.50000000e+01, 6.40000000e+01, 1.01000000e+02, 2.70000000e+01,
       1.61000000e+02, 2.00000000e+00, 1.53000000e+02, 7.00000000e+01,
       7.80000000e+01, 1.46000000e+03, 4.20000000e+01, 1.10000000e+01,
       1.67000000e+02, 1.50000000e+02, 3.66000000e+02, 9.00000000e+00,
       1.60000000e+01, 3.80000000e+01, 4.80000000e+01, 1.00000000e+01,
       7.60000000e+01, 4.00000000e+01, 2.00000000e+01, 3.70000000e+01,
       4.10000000e+01, 1.12000000e+02, 2.24000000e+02, 1.80000000e+02,
       2.96000000e+02, 3.58000000e+02, 6.30000000e+01, 2.76000000e+02,
       4.60000000e+01, 5.40000000e+01, 1.50000000e+01, 2.10000000e+01,
       3.50000000e+01, 1.24000000e+02, 8.60000000e+01, 1.02000000e+02,
       3.60000000e+01, 6.70000000e+01, 2.91000000e+02, 1.10000000e+02,
       2.90000000e+01, 5.50000000e+01, 2.01000000e+02, 1.42000000e+02,
       1.65500000e+03, 1.09000000e+02, 3.40000000e+01, 1.36000000e+02,
       3.20000000e+01, 1.60328571e+03, 7.30000000e+01, 1.14000000e+02,
       1.90000000e+01, 1.95000000e+02, 5.80000000e+01, 1.55000000e+02,
       9.60000000e+01, 1.03000000e+02, 1.13000000e+02, 1.04000000e+02,
       1.92000000e+02, 1.91000000e+02, 2.03000000e+02, 5.60000000e+01,
       5.00000000e+02, 8.00000000e+01, 1.72000000e+02, 6.50000000e+01,
       1.17000000e+02, 2.80000000e+01, 1.83900000e+03, 6.10000000e+01,
       3.00000000e+01, 1.48000000e+02, 1.28000000e+02, 1.00000000e+02,
       1.70000000e+01, 2.43000000e+02, 9.20000000e+01, 1.05000000e+02,
       7.90000000e+01, 2.83014286e+03, 3.10000000e+01, 1.78700000e+03,
       5.30000000e+01, 3.30000000e+01, 1.30000000e+02, 1.80000000e+01,
       9.70000000e+01, 1.93000000e+02, 1.15000000e+02, 1.70000000e+02,
       6.60000000e+01, 3.30000000e+02, 1.08000000e+02, 6.80000000e+01,
       1.19000000e+02, 9.50000000e+01, 1.37000000e+02, 6.00000000e+01,
       7.70000000e+01, 7.20000000e+01, 1.27000000e+02, 9.90000000e+01,
       3.73000000e+02, 3.00000000e+02, 1.63000000e+02, 9.10000000e+01,
       8.80000000e+01, 1.54000000e+02, 1.31700000e+03, 1.56000000e+02,
       6.94000000e+02, 8.70000000e+01, 2.25000000e+02, 1.64000000e+02,
       2.15000000e+02, 5.90000000e+01, 1.82000000e+02, 3.05000000e+02,
       3.65000000e+02, 1.51214286e+03, 1.47100000e+03, 3.31000000e+02,
       1.75000000e+02, 1.43000000e+02, 1.49942857e+03, 2.00000000e+02,
       5.10000000e+02, 1.51000000e+02, 1.42800000e+03, 7.18000000e+02,
       9.75428571e+02, 8.40000000e+01, 7.26000000e+02, 1.40000000e+02,
       8.30000000e+01, 7.83000000e+02, 3.05700000e+03, 1.47000000e+02,
       1.00600000e+03, 6.66285714e+02, 4.25000000e+02, 8.50000000e+01,
       2.60000000e+02, 5.26000000e+02, 1.81800000e+03, 2.58000000e+02,
       3.98000000e+02, 4.31000000e+02, 3.12000000e+02, 2.63000000e+02,
       2.27000000e+02, 6.13000000e+02, 2.14000000e+02, 9.30000000e+01,
       1.32000000e+02, 1.62000000e+02, 9.00000000e+01, 2.40000000e+02,
       2.83000000e+02, 7.73000000e+02, 1.99000000e+02, 1.27400000e+03,
       6.62000000e+02, 1.56500000e+03, 5.09000000e+02, 5.07714286e+02,
       7.10000000e+01, 5.33428571e+02, 4.69285714e+02, 4.92285714e+02,
       1.49985714e+03, 5.67428571e+02, 7.18428571e+02, 3.65428571e+02,
       5.09428571e+02, 5.74857143e+02,            nan, 5.80428571e+02,
       5.46000000e+02, 1.25000000e+02, 1.31000000e+02, 1.39000000e+02,
       5.30000000e+02, 4.75000000e+02, 5.40571429e+02, 4.04285714e+02,
```

```
1.30600000e+03, 8.90000000e+01, 4.11571429e+02, 6.64714286e+02,
2.30000000e+02, 7.44000000e+02, 1.20000000e+02, 4.03714286e+02,
1.46000000e+02, 3.80285714e+02, 3.77285714e+02, 5.29428571e+02,
2.93000000e+02, 5.11571429e+02, 4.18714286e+02, 3.54000000e+02,
8.20000000e+01, 3.78428571e+02, 7.38571429e+02, 5.20285714e+02,
3.52000000e+02, 3.47428571e+02, 9.76428571e+02, 5.16571429e+02,
4.36000000e+02, 4.62428571e+02, 4.21714286e+02, 1.22000000e+02,
4.70285714e+02, 3.25142857e+02, 5.42142857e+02, 4.28428571e+02,
4.17857143e+02, 4.58571429e+02, 3.30142857e+02, 1.35000000e+02,
3.06571429e+02, 2.99714286e+02, 2.98000000e+02, 2.37000000e+02,
3.12714286e+02, 2.87142857e+02, 2.84571429e+02, 2.82714286e+02,
6.66000000e+02, 2.74000000e+02, 2.82428571e+02, 2.68142857e+02,
5.17285714e+02, 2.67428571e+02, 4.92000000e+02, 5.88571429e+01,
2.04428571e+02, 2.39428571e+02, 2.75571429e+02, 1.88428571e+02,
3.84000000e+02, 1.94857143e+02, 1.96571429e+02, 1.61800000e+03,
1.72857143e+02, 2.14857143e+02, 3.80000000e+02, 1.65428571e+02,
1.67142857e+02, 1.74000000e+02, 1.42714286e+02, 3.94857143e+02,
1.66571429e+02, 3.40000000e+02, 1.66857143e+02, 2.12000000e+02,
2.43142857e+02, 2.10000000e+02, 1.23714286e+02, 4.18000000e+02,
3.09000000e+02, 2.34000000e+02, 2.08000000e+02, 6.24000000e+02,
3.20000000e+02, 4.32000000e+02, 1.18000000e+02, 9.80000000e+01,
2.50000000e+02, 8.00000000e+02, 4.20000000e+02, 1.60000000e+02,
1.16000000e+02, 3.67571429e+02, 5.70000000e+01, 4.99000000e+02,
3.64000000e+02, 3.56000000e+02, 2.94000000e+02, 1.52000000e+02,
3.67000000e+02, 1.44000000e+02, 2.29000000e+02, 1.21000000e+02,
1.38000000e+02, 1.66400000e+03, 1.00000000e+03, 2.47000000e+02,
3.25000000e+02, 2.99000000e+02, 2.67000000e+02, 3.60000000e+02,
2.84000000e+02, 1.77000000e+02, 8.10000000e+01, 2.17000000e+02,
1.96000000e+02, 4.00000000e+02, 1.53428571e+02, 3.02571429e+02,
3.90000000e+02, 1.91428571e+02, 2.30714286e+02, 3.02000000e+02,
5.11428571e+01, 1.45714286e+02, 2.52857143e+01, 1.39428571e+02,
2.45000000e+02, 1.31285714e+02, 2.44285714e+01, 1.28571429e+02,
3.04285714e+01, 1.17428571e+02, 2.51428571e+01, 1.06428571e+02,
1.05428571e+02, 3.91428571e+01, 9.44285714e+01, 1.26000000e+02,
1.10857143e+02, 1.03285714e+02, 1.53571429e+02, 9.64285714e+01,
1.03714286e+02, 9.11428571e+01, 2.54285714e+01, 1.69000000e+02,
7.44285714e+01, 9.54285714e+01, 2.55714286e+01, 2.47142857e+01,
2.58571429e+01, 1.32857143e+01, 2.27142857e+01, 9.32857143e+01,
2.62857143e+01, 7.94285714e+01, 2.57142857e+01, 8.84285714e+01,
1.03428571e+02, 2.56428571e+02, 7.01428571e+01, 8.37142857e+01,
1.13571429e+02, 1.87428571e+02, 7.34285714e+01, 6.94285714e+01,
2.71428571e+01, 1.31571429e+02, 6.54285714e+01, 6.47142857e+01,
6.44285714e+01, 6.24285714e+01, 8.87142857e+01, 1.14428571e+02,
7.25714286e+01, 1.47714286e+02, 2.06571429e+02, 1.79714286e+02,
7.85714286e+01, 4.54285714e+01, 2.61428571e+01, 2.53571429e+02,
2.48571429e+01, 2.45714286e+01, 5.28571429e+01, 5.34285714e+01,
6.58571429e+01, 5.62857143e+01, 4.84285714e+01, 6.07142857e+01,
2.74285714e+01, 4.74285714e+01, 4.71428571e+01, 4.44285714e+01,
4.34285714e+01, 5.94285714e+01, 3.74285714e+01, 3.94285714e+01,
4.72857143e+01, 2.01285714e+02, 4.97142857e+01, 4.80000000e+02,
5.01428571e+01, 3.83000000e+02, 4.27142857e+01, 3.82857143e+01,
3.54285714e+01, 2.56285714e+02, 4.64285714e+01, 4.52857143e+01,
3.14285714e+01, 3.92857143e+01, 4.38571429e+01, 4.07142857e+01,
1.85142857e+02, 2.34857143e+02, 3.81428571e+01, 2.65714286e+01,
3.78571429e+01, 3.75714286e+01, 7.04285714e+01, 3.61428571e+01,
1.57000000e+02, 3.64285714e+01, 3.65714286e+01, 2.24285714e+01,
```

```
       3.27142857e+01, 6.87142857e+01, 1.93428571e+02, 1.84285714e+01,
       2.84285714e+01, 3.01428571e+01, 2.44000000e+02, 4.58571429e+01,
       2.77142857e+01, 4.15714286e+01, 3.07142857e+01])
```

In [121…

```
target_df['Score'].unique()
```

```
Out[121… array(['8.75', '8.38', '8.22', '7.25', '6.94', '7.92', '8.0', '7.55',
       '8.16', '8.87', '7.99', '8.69', '7.86', '6.39', '7.89', '7.38',
       '7.76', '7.29', '7.91', '7.48', '8.35', '7.46', '8.55', '8.56',
       '8.27', '8.71', '8.29', '6.95', '7.32', '6.27', '7.26', '7.11',
       '7.06', '6.51', '5.86', '7.03', '7.4', '7.62', '7.17', '6.72',
       '6.56', '7.41', '7.33', '6.31', '7.61', '8.03', '7.93', '7.64',
       '7.97', '8.01', '7.96', '6.76', '7.24', '7.9', '6.68', '7.66',
       '6.62', '6.77', '7.69', '7.74', '7.34', '7.75', '7.18', '7.57',
       '7.79', '7.42', '7.28', '7.16', '7.44', '7.49', '6.4', '7.19',
       '6.79', '7.58', '6.97', '4.82', '7.37', '6.93', '6.78', '6.52',
       '8.11', '7.94', '6.82', '6.44', '6.86', '6.92', '6.74', '6.15',
       '7.39', '8.08', '8.41', '8.31', '8.18', '8.25', '6.1', '6.81',
       '7.07', '7.59', '6.71', '7.1', '7.27', '8.02', '6.57', '6.28',
       '7.3', '6.55', '8.67', '7.36', '8.54', '6.32', '7.2', '6.88',
       '6.59', '6.09', '7.52', '7.67', '8.33', '6.03', '7.21', '7.31',
       '7.51', '8.78', '7.43', '7.8', '6.08', '8.51', '7.13', '7.63',
       '7.53', '6.17', '6.85', '5.94', '6.07', '6.19', '5.81', '6.73',
       '5.62', '6.65', '6.48', '6.89', '6.63', '8.17', '8.06', '8.14',
       '6.38', '6.83', '7.22', '7.0', '7.54', '7.81', '7.12', '6.6',
       '6.61', '8.76', '8.26', '8.23', '7.83', '8.04', '5.85', '7.15',
       '7.14', '5.73', '7.77', '7.47', '6.58', '6.8', '6.0', '6.41',
       '6.69', '8.1', '5.28', '7.23', '6.02', '6.13', '5.99', '6.34',
       '6.14', '7.72', '5.96', '6.54', '5.97', '7.85', '5.98', '7.6',
       '6.43', '7.01', '6.67', '6.16', '5.65', '7.45', '5.78', '6.5',
       '5.93', '8.21', '6.66', '6.99', '8.46', '7.65', '7.84', '6.3',
       '7.73', '2.22', '7.95', '6.45', '6.9', '6.36', '6.11', '8.66',
       '5.51', '6.46', '8.2', '6.53', '7.56', '6.96', '6.24', '7.09',
       '6.87', '6.7', '8.42', '7.71', '7.05', '5.33', '7.35', '5.77',
       '6.12', '6.01', '6.64', '5.27', '5.91', '6.98', '6.75', '6.05',
       '8.19', '7.08', '7.78', '8.36', '5.61', '6.2', '5.89', '4.86',
       '6.42', '6.26', '5.64', '5.63', '7.02', '6.84', '5.42', '7.04',
       '5.76', '5.02', '5.67', '6.33', '6.04', '7.5', '5.58', '5.66',
       '4.64', '6.49', '6.35', '8.13', '5.16', '4.89', '5.04', '5.83',
       '5.88', '4.84', '5.59', '5.82', '7.88', '4.31', '8.52', '5.5',
       '6.21', '9.02', '6.37', '6.47', '8.05', '5.84', '7.98', '6.25',
       '8.94', '7.87', '6.18', '5.48', '5.4', '6.29', '5.79', '5.68',
       '5.9', '5.54', '4.7', '4.78', '5.23', '4.93', '5.69', '5.18',
       '4.79', '5.21', '5.74', '6.22', '5.95', '4.42', '5.44', '5.6',
       '5.56', '6.91', '5.45', '7.7', '5.7', '5.75', '5.92', '5.49',
       '5.8', '5.26', '5.71', '5.47', '6.23', '5.32', '5.01', '5.34',
       '5.29', '3.08', '5.57', '4.83', '8.12', '8.3', '8.07', '5.37',
       '5.87', '8.15', '5.72', '5.19', '6.06', '5.39', '4.57', '8.62',
       '4.96', '8.7', '3.55', '5.31', '4.72', '5.15', '5.36', '4.23',
       '4.49', '4.76', '4.1', '5.06', '5.55', '5.43', '5.24', '5.22',
       'UNKNOWN', '5.1', '5.25', '5.53', '3.33', '5.09', '4.13', '3.83',
       '4.59', '8.63', '4.85', '4.81', '5.38', '4.3', '5.13', '4.54',
       '7.68', '4.66', '5.3', '5.46', '4.92', '4.95', '4.51', '8.43',
       '5.41', '4.11', '8.37', '4.8', '5.05', '3.75', '5.35', '5.07',
       '4.29', '5.2', '8.32', '4.77', '5.11', '3.07', '3.32', '3.56',
       '3.21', '8.91', '4.5', '4.75', '5.14', '4.2', '5.17', '4.03',
       '5.12', '4.74', '1.85', '8.48', '4.68', '5.52', '4.61', '4.63',
       '4.88', '4.87', '4.44', '7.82', '8.61', '5.0', '3.69', '4.28',
       '4.73', '8.09', '4.45', '4.97', '8.93', '4.69', '4.62', '8.53',
       '3.48', '5.03', '5.08', '3.22', '8.57', '3.79', '4.48', '4.56',
       '4.98', '4.33', '4.16', '4.65', '8.4', '4.99', '9.1', '8.39',
       '4.35', '4.47', '4.91', '4.41', '4.52', '4.6', '4.58', '4.55',
```

```
'4.43', '3.9', '4.67', '4.46', '4.25', '2.53', '4.32', '4.94',
'4.19', '8.28', '3.93', '4.38', '4.26', '4.37', '3.6', '2.74',
'3.86', '4.53', '3.73', '3.91', '3.81', '2.91', '4.39', '3.13',
'4.9', '4.18', '9.07', '2.98', '4.71', '4.08', '4.17', '9.04',
'4.22', '3.98', '8.45', '8.64', '3.62', '4.21', '8.24', '3.19',
'3.25', '3.36', '8.47', '8.58', '8.5', '4.4', '1.98', '8.34',
'3.63', '9.03', '3.35', '3.5', '3.47', '2.76', '3.58', '8.77',
'4.36', '3.29', '8.59', '8.44', '4.12', '3.61', '3.14', '3.05',
'3.4', '2.37', '3.46', '4.15', '3.97', '3.87', '8.65', '4.09',
'4.07', '8.73', '3.71', '3.52', '3.28', '9.06', '2.95', '3.54',
'4.14', '3.76', '3.99', '4.34', '4.27', '8.49', '4.0', '3.49',
'3.26', '2.99', '2.3', '3.1', '8.79', '8.85', '2.63', '4.05',
'2.59', '8.68', '4.01', '3.31', '4.06', '3.51', '8.98', '3.3',
'8.88', '3.57', '3.88', '8.81', '3.06', '8.8', '8.74', '8.9',
'8.6', '9.05', '2.9', '3.03', '3.65', '9.0', '8.72'], dtype=object)
```

In [122…
```
target_df = target_df.drop(target_df[target_df['Score'] == 'UNKNOWN'].index)
```

In [123…
```
target_df['Score'] = target_df['Score'].astype(float)
```

In [124…
```
target_df.dropna()
```

Out[124…

	Episodes	Score
0	26.0	8.75
1	1.0	8.38
2	26.0	8.22
3	26.0	7.25
4	52.0	6.94
...	...	...
24590	1.0	5.99
24635	1.0	6.45
24729	1.0	6.07
24831	1.0	6.29
24856	1.0	6.45

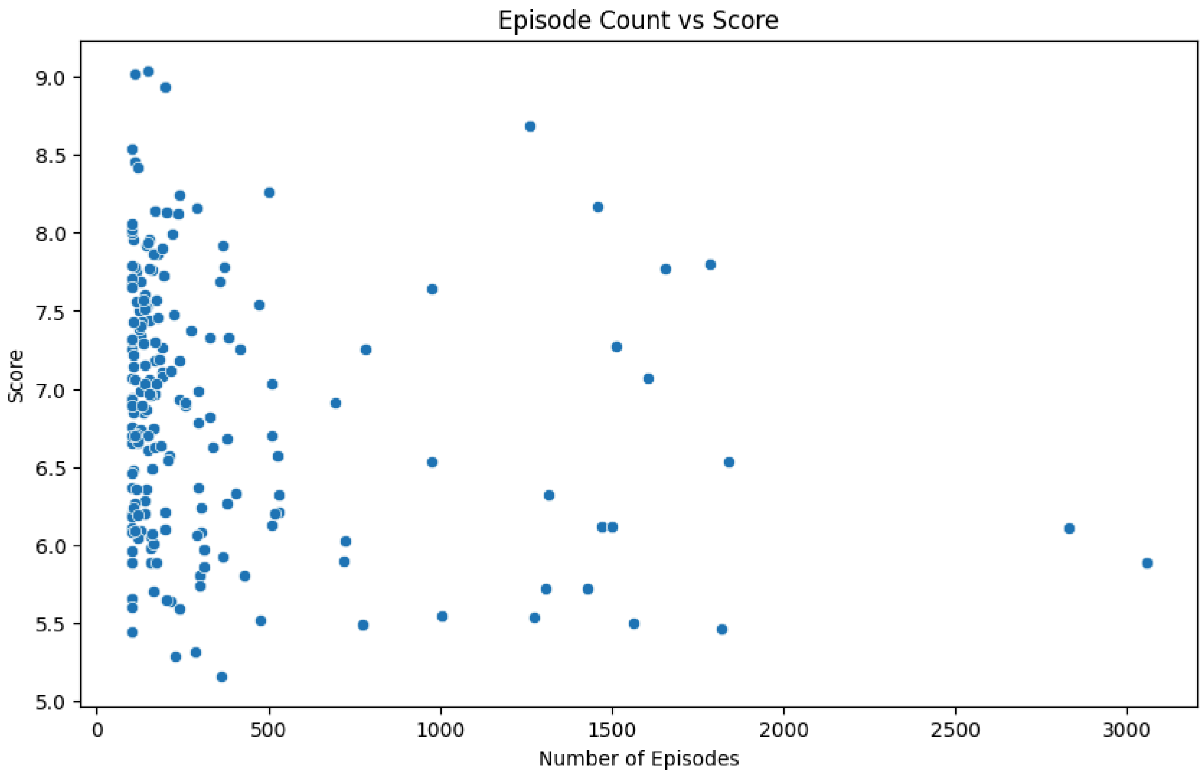
15692 rows × 2 columns

We filter animes with episode count minimum 100. We have some movies and really short series which include bias to our dataset

In [125…
```
target_df = target_df[target_df['Episodes'] >100]
```

In [126…
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [127…
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Episodes', y='Score', data=target_df)
plt.title('Episode Count vs Score')
plt.xlabel('Number of Episodes')
plt.ylabel('Score')
plt.show()
```



There seems to be a cluster for animes having less than 500 episodes, however their ratings are scattered from 5.5 to 8. This plot basically suggests that there is no strong correlation between anime score and number of episodes, and hence score depends on other factors too.

We also have some outliers indicating series with either a high number of episodes and a low score, or vice versa.

The presence of outliers also suggest potential bias present in the dataset since animes with around 3000 episodes are rated lower

Lets calculate correlation between the episodes and score

In [128…
```
correlation = target_df['Episodes'].corr(target_df['Score'])
```

In [129…
```
correlation
```

Out[129…
```
-0.1820350120174825
```

Correlation close to 0 suggest episodes and score have weak correlation

EDA 2

We plan to distribute the score into bins to get an idea which range of episodes have the highest score

In [130…

```
bins = [1, 5, 7, 9, 10]
labels = ['1-5', '5-7', '7-9', '9-10']
target_df['score_category'] = pd.cut(target_df['Score'], bins=bins, labels=l
```

In [131…

```
target_df
```

Out[131…

	Episodes	Score	score_category
5	145.000000	7.92	7-9
10	220.000000	7.99	7-9
11	1262.714286	8.69	7-9
12	178.000000	7.86	7-9
148	101.000000	8.54	7-9
...	...	...	...
21738	104.000000	7.32	7-9
21809	103.285714	5.44	5-7
22099	120.000000	6.19	5-7
22468	103.428571	6.70	5-7
23023	112.000000	6.70	5-7

199 rows × 3 columns

In [132…

```
score_group = target_df.groupby('score_category')['Episodes'].mean()
```

```
<ipython-input-132-9704a731daad>:1: FutureWarning: The default of observed=F
alse is deprecated and will be changed to True in a future version of panda
s. Pass observed=False to retain current behavior or observed=True to adopt
the future default and silence this warning.
  score_group = target_df.groupby('score_category')['Episodes'].mean()
```

In [133…

```
score_group
```

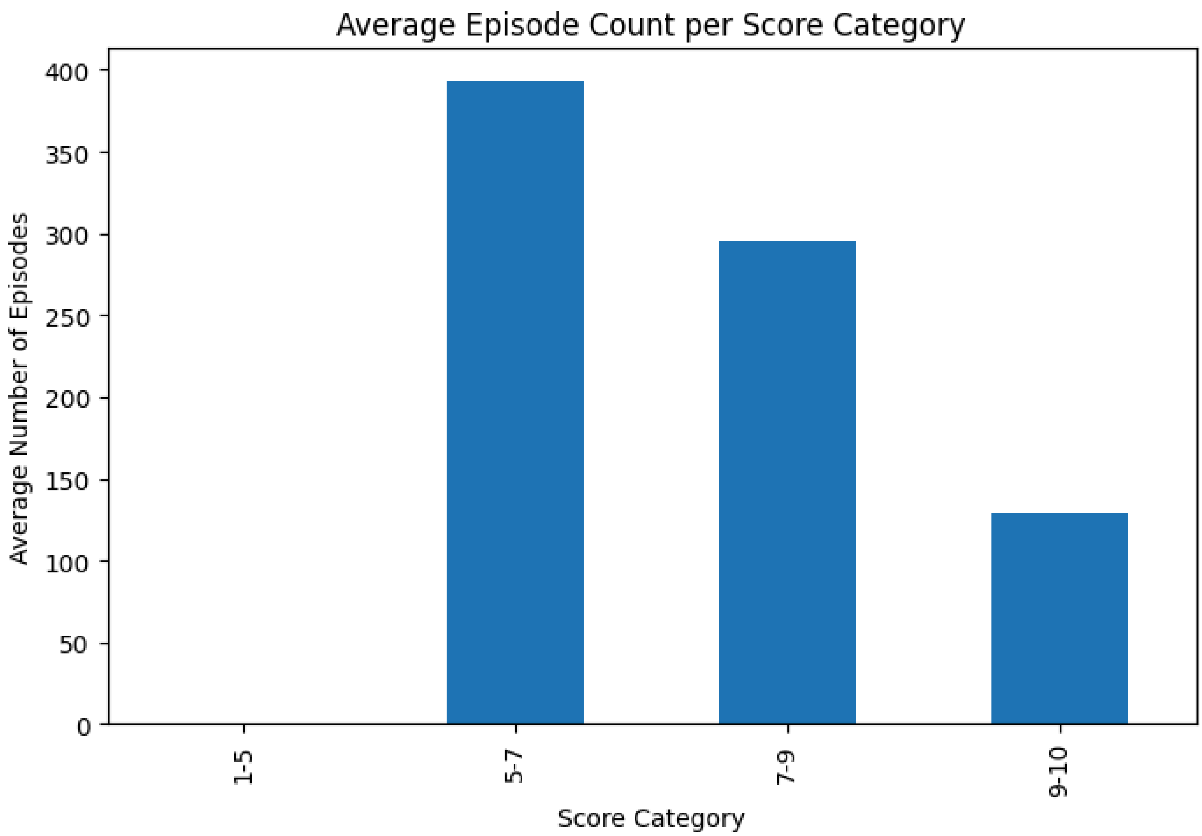
Out[133…

score_category	Episodes
1-5	NaN
5-7	393.352041
7-9	295.588235
9-10	129.000000

dtype: float64

In [134…

```
plt.figure(figsize=(8, 5))
score_group.plot(kind='bar')
plt.title('Average Episode Count per Score Category')
plt.xlabel('Score Category')
plt.ylabel('Average Number of Episodes')
plt.show()
```



We get similar results from this graph too. The general trend that this plot shows is that score increases as number of episodes decrease. Also we have no anime in our dataset having score in the range 1 to 5.

This further adds to the bias since we do not know what number of episodes those lower scored animes had.

The sweet spot acoording to our plot is for the highest rated animes having average number of episoded around 150

In [135…
```
target_df[(target_df['Score']>1) & (target_df['Score']<5)]
```

Out[135…

Episodes	Score	score_category

Here we confirm that we have 0 animes having score in the range 1 to 5 in our data frame after processing

Finally, to answer our hypotheses, our plots suggest that score and number of episodes are not strongly correlated and the score depends on other factors too.

However,our bar graph does suggest highest rated animes having average number of episoded around 150

Question 2: which Genres are the most popular?

This question will help us figure out the popular genres so as to recommend animes having those genres to new user

We frame the hypothesis as 'Popular Genre animes usually have higher score'

We will use 'Genres' column and 'Score' column. Both columns have 'UNKNOWN' values which need to be dropped

EDA 1

We plan to plot a box plot to analyse the score's min, max, mean and quartiles. We do necessary cleaning and feature selection

In [136…
```
target_df = anime_df[['Genres', 'Score']]
```

In [137…
```
target_df
```

Out[137…

	Genres	Score
0	Action, Award Winning, Sci-Fi	8.75
1	Action, Sci-Fi	8.38
2	Action, Adventure, Sci-Fi	8.22
3	Action, Drama, Mystery, Supernatural	7.25
4	Adventure, Fantasy, Supernatural	6.94
...	...	...
24900	Comedy, Fantasy, Slice of Life	UNKNOWN
24901	Action, Adventure, Fantasy	UNKNOWN
24902	Action, Adventure, Fantasy, Sci-Fi	UNKNOWN
24903	UNKNOWN	UNKNOWN
24904	UNKNOWN	UNKNOWN

24905 rows × 2 columns

In [138…
```
target_df = target_df.drop(target_df[target_df['Score'] == 'UNKNOWN'].index)
```

In [139…
```
target_df = target_df.drop(target_df[target_df['Genres'] == 'UNKNOWN'].index
```

In [140…
```
target_df['Score'] = target_df['Score'].astype(float)
```

In [141…
```
target_df['Genres_Split'] = target_df['Genres'].str.split(', ')
```

In [142…
```
target_df
```

Out[142…

	Genres	Score	Genres_Split
0	Action, Award Winning, Sci-Fi	8.75	[Action, Award Winning, Sci-Fi]
1	Action, Sci-Fi	8.38	[Action, Sci-Fi]
2	Action, Adventure, Sci-Fi	8.22	[Action, Adventure, Sci-Fi]
3	Action, Drama, Mystery, Supernatural	7.25	[Action, Drama, Mystery, Supernatural]
4	Adventure, Fantasy, Supernatural	6.94	[Adventure, Fantasy, Supernatural]
...	...	...	...
24539	Action, Adventure, Comedy, Fantasy	6.47	[Action, Adventure, Comedy, Fantasy]
24557	Fantasy	7.78	[Fantasy]
24579	Action, Comedy, Fantasy	5.84	[Action, Comedy, Fantasy]
24590	Action, Comedy, Mystery	5.99	[Action, Comedy, Mystery]
24831	Action, Adventure, Comedy, Fantasy	6.29	[Action, Adventure, Comedy, Fantasy]

13939 rows × 3 columns

In [143…

```
target_df = target_df.explode('Genres_Split')
```

In [144…

```
target_df
```

Out[144…

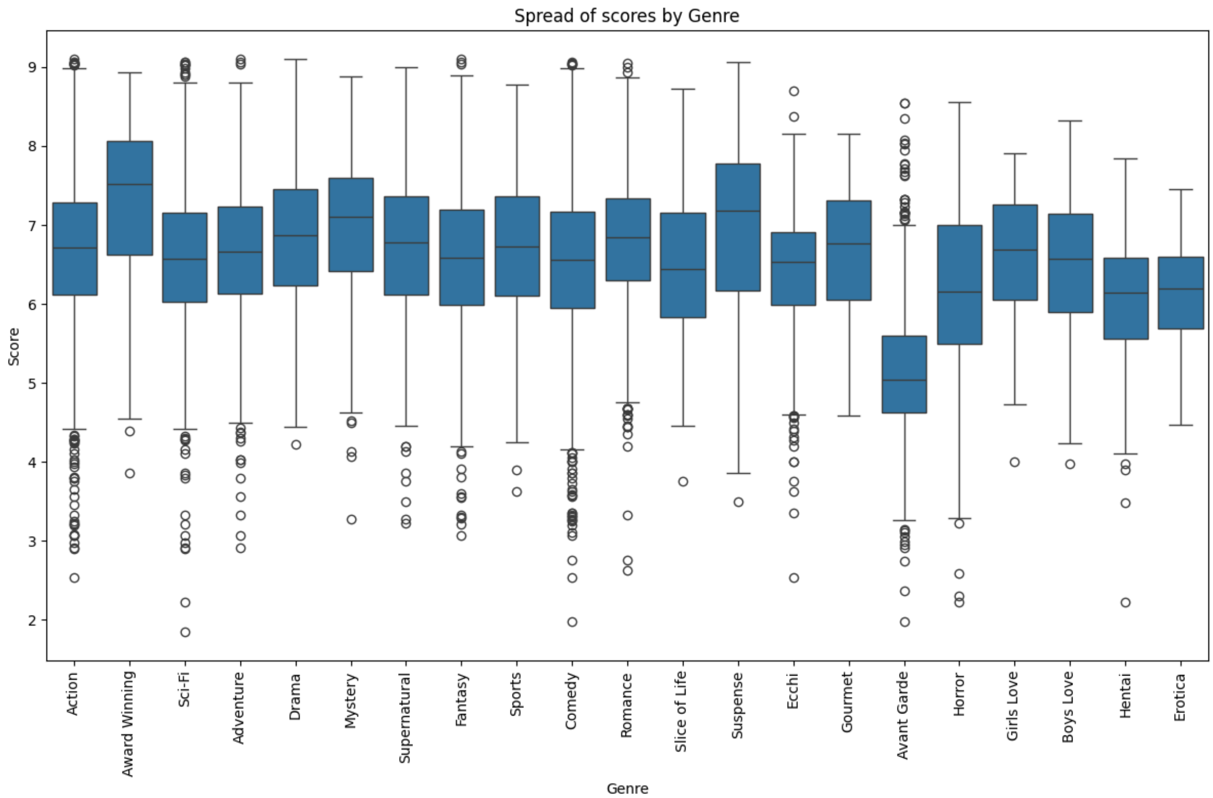
	Genres	Score	Genres_Split
0	Action, Award Winning, Sci-Fi	8.75	Action
0	Action, Award Winning, Sci-Fi	8.75	Award Winning
0	Action, Award Winning, Sci-Fi	8.75	Sci-Fi
1	Action, Sci-Fi	8.38	Action
1	Action, Sci-Fi	8.38	Sci-Fi
...	...	...	...
24590	Action, Comedy, Mystery	5.99	Mystery
24831	Action, Adventure, Comedy, Fantasy	6.29	Action
24831	Action, Adventure, Comedy, Fantasy	6.29	Adventure
24831	Action, Adventure, Comedy, Fantasy	6.29	Comedy
24831	Action, Adventure, Comedy, Fantasy	6.29	Fantasy

29207 rows × 3 columns

In [145…

```
plt.figure(figsize=(12, 8))
sns.boxplot(x='Genres_Split', y='Score', data=target_df)
plt.title('Spread of scores by Genre')
plt.xlabel('Genre')
plt.ylabel('Score')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWa
rning: SeriesGroupBy.grouper is deprecated and will be removed in a future v
ersion of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



Genres like Award-Winning and Suspense have the highest median scores whereas Avant Garde has the lowest median score

We have many outliers suggesting some genres recieved much lower or higher scoress in their genre respectively which suggests potential bias present, maybe for animes like Action, Comedy and Avant Garde

EDA 2

We plan to plot a Bar chart of average score by genre We do necessary cleaning and feature selection

In [146…

```
target_df = target_df.groupby('Genres_Split')['Score'].mean().sort_values(as
```

In [147…

```
target_df
```

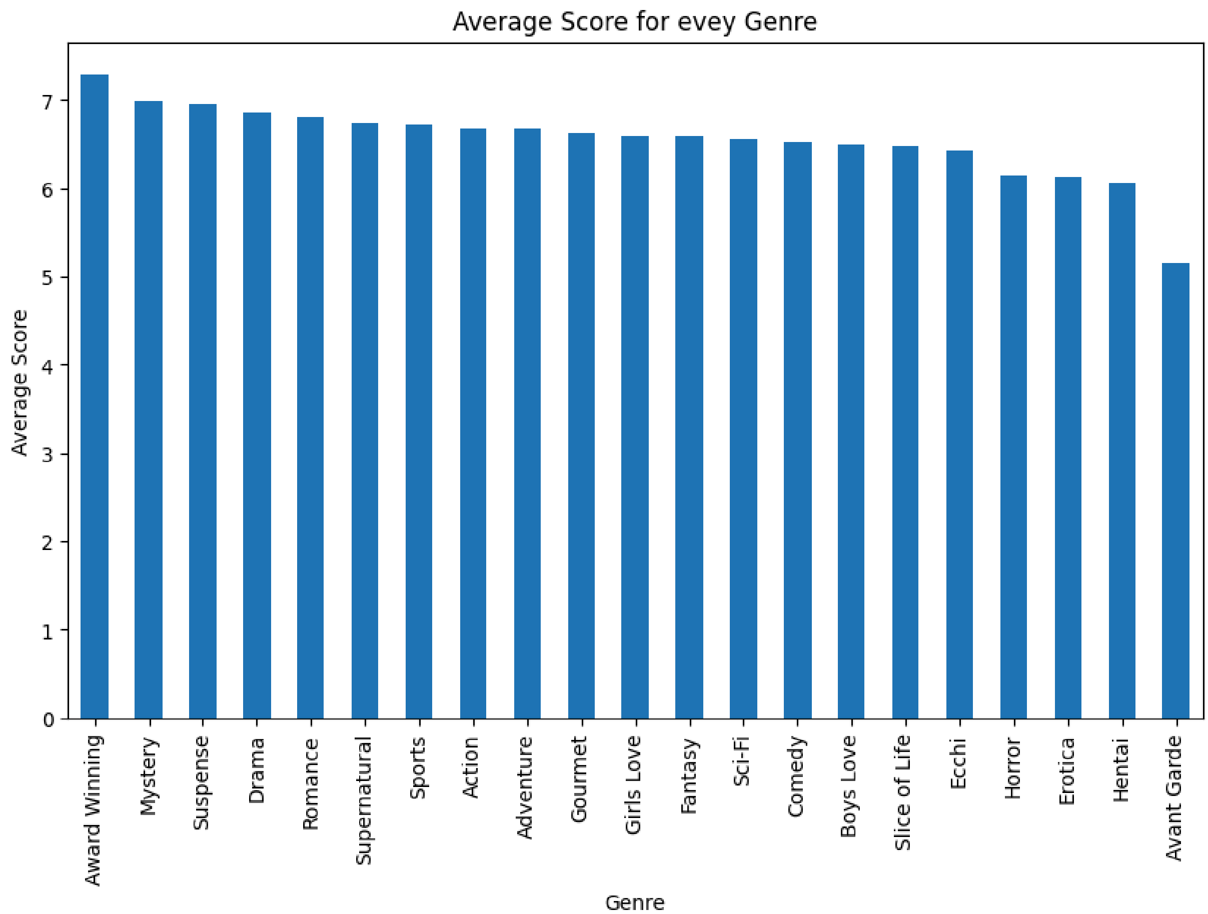
Out[147…

Genres_Split	Score
Award Winning	7.296308
Mystery	6.995093
Suspense	6.962963
Drama	6.850645
Romance	6.804509
Supernatural	6.744600
Sports	6.722046
Action	6.674112
Adventure	6.673997
Gourmet	6.627664
Girls Love	6.591553
Fantasy	6.591243
Sci-Fi	6.563554
Comedy	6.522961
Boys Love	6.500533
Slice of Life	6.475667
Ecchi	6.431904
Horror	6.148137
Erotica	6.124419
Hentai	6.065379
Avant Garde	5.143932

dtype: float64

In [148…

```
plt.figure(figsize=(10, 6))
target_df.plot(kind='bar')
plt.title('Average Score for evey Genre')
plt.xlabel('Genre')
plt.ylabel('Average Score')
plt.xticks(rotation=90)
plt.show()
```



Award Winning animes are the only animes managed to cross 7 average score suggesting high preference among viewers.

Genres like avant garde have lowest average score suggesting they may be a niche or have a more specialized appealing

Finally, to answer our hypotheses, we have identified popular anime genres like Award Wining, Mystery, Suspense

```
In [ ]:
```