Title: Anime Recommendation Dataset Analysis

In [ ]:
```
Name: K. Swapna
Roll No: 2211CS010302
Section: S3
```

Anime Dataset Overview

- **Total Records:** 1,864
- **Columns:** `anime_id`, `name`, `genre`, `type`, `episodes`, `rating`, `members`
- **Column Types:**
  - `anime_id` → float
  - `name` → object
  - `genre` → object
  - `type` → object
  - `episodes` → object
  - `rating` → float
  - `members` → float

Key Points:

- Each record represents one anime.
- `episodes` has some unknown/missing values.
- `genre` may contain multiple genres per anime.
- `rating` shows average user ratings.
- `members` indicates popularity.

Use: The dataset can be used to analyze anime popularity, ratings, trends by type or genre, and relationships between numeric fields like `episodes`, `rating`, and `members`.

In [1]:
```
sc
```

Out[1]: **SparkContext**

Spark UI

Version	`v3.5.6`
Master	`local[*]`
AppName	`PySparkShell`

In [4]:
```
# PySpark imports
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Create Spark session
spark = SparkSession.builder.appName("AnimeAnalysis").getOrCreate()

# Load CSV as DataFrame
df = spark.read.option("header", True).option("inferSchema", True).csv("anime.csv")

# Show first 5 rows
df.show(5)
```

```
+--------+--------------------+--------------------+-----+--------+------+-------+
|anime_id|                name|               genre| type|episodes|rating|members|
+--------+--------------------+--------------------+-----+--------+------+-------+
|   32281|      Kimi no Na wa.|Drama, Romance, S...|Movie|       1|  9.37| 200630|
|    5114|Fullmetal Alchemi...|Action, Adventure...|   TV|      64|  9.26| 793665|
|   28977|            Gintama°|Action, Comedy, H...|   TV|      51|  9.25| 114262|
|    9253|         Steins;Gate|    Sci-Fi, Thriller|   TV|      24|  9.17| 673572|
|    9969|       Gintama&#039;|Action, Comedy, H...|   TV|      51|  9.16| 151266|
+--------+--------------------+--------------------+-----+--------+------+-------+
only showing top 5 rows
```

In [5]:
```
# Column names
print("=== Column Names ===")
```

```
print(df.columns)

# Data types
print("\n=== Data Types ===")
print(df.dtypes)
```

```
=== Column Names ===
['anime_id', 'name', 'genre', 'type', 'episodes', 'rating', 'members']

=== Data Types ===
[('anime_id', 'int'), ('name', 'string'), ('genre', 'string'), ('type', 'string'), ('episodes', 'string'), ('rating',
'double'), ('members', 'int')]
```

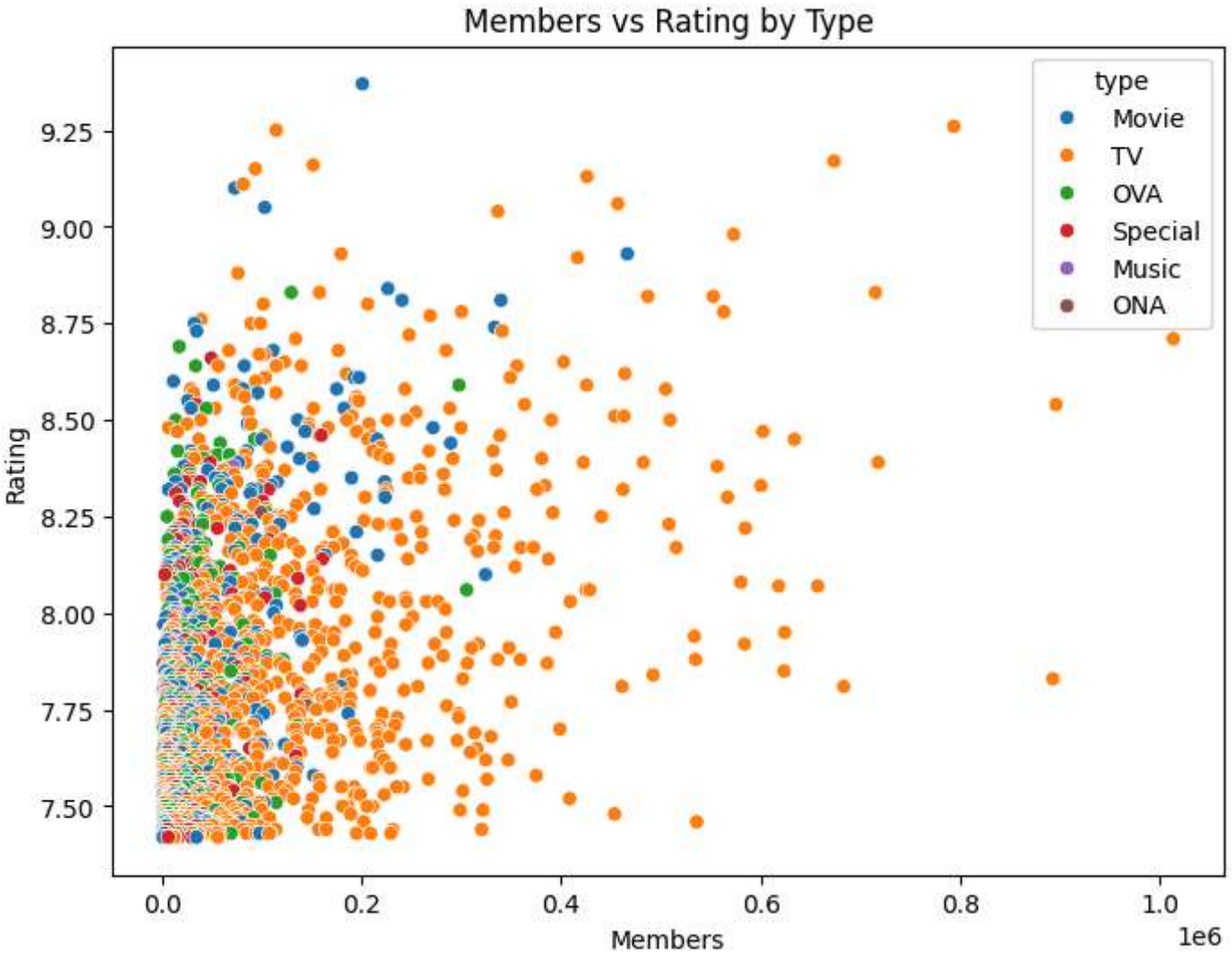
In [7]:
```
# Convert episodes to numeric (some may be 'Unknown')
df = df.withColumn("episodes", when(col("episodes").rlike("^[0-9]+$"), col("episodes").cast("int")).otherwise(None))

# Ensure numeric columns are floats
df = df.withColumn("anime_id", col("anime_id").cast("float")) \
       .withColumn("rating", col("rating").cast("float")) \
       .withColumn("members", col("members").cast("float"))
```

In [8]:
```
df.describe().show()
```

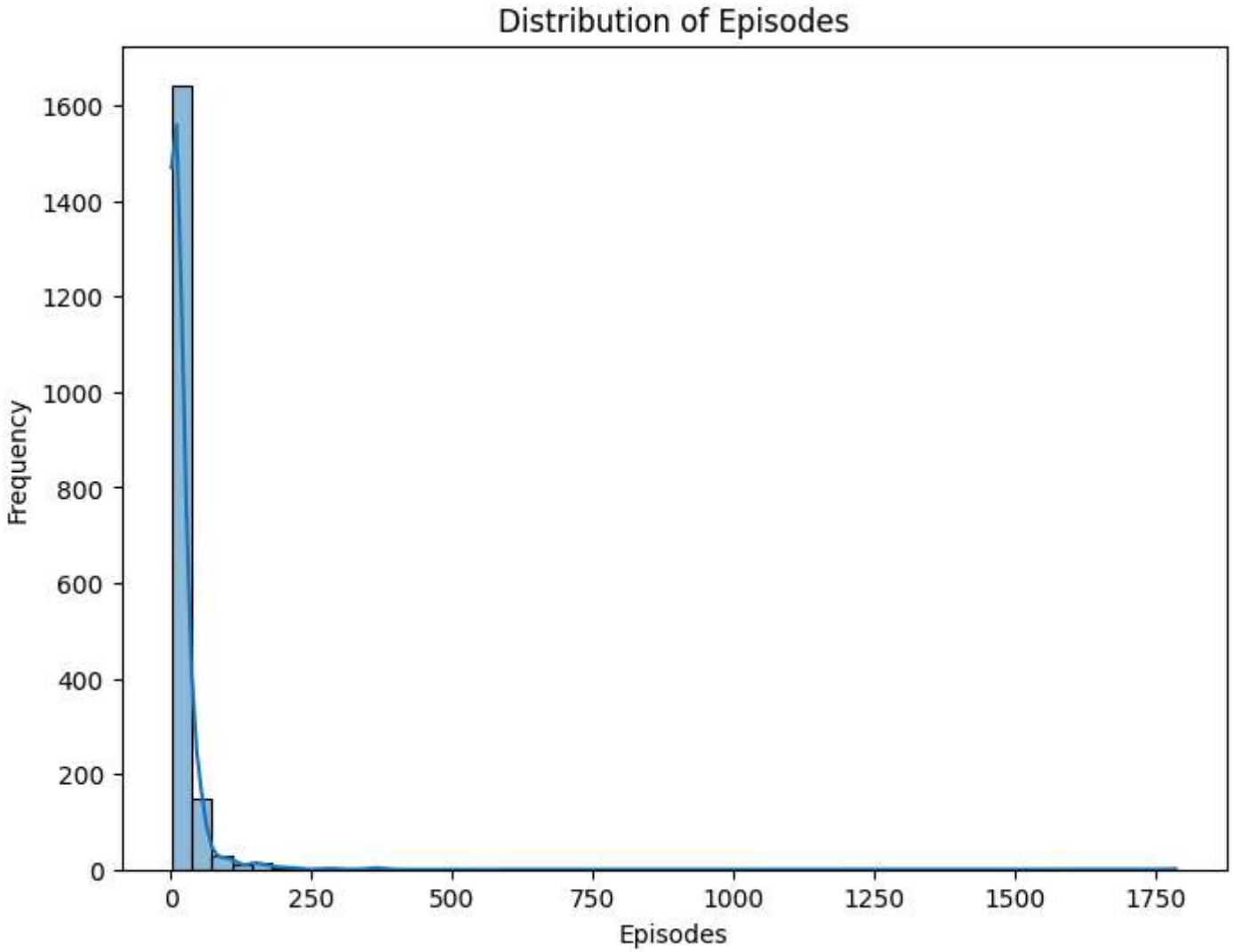
```
+-------+-----------------+------------------+------+-----+-----------------+------------------+----------------
-+
|summary|         anime_id|              name| genre| type|         episodes|            rating|          member
s|
+-------+-----------------+------------------+------+-----+-----------------+------------------+----------------
-+
|  count|             1863|              1863|  1863| 1863|             1856|              1863|             186
3|
|   mean| 10485.70692431562|              NULL|  NULL| NULL|17.82112068965517|  7.844787962642717| 80396.6403650026
8|
| stddev|10409.690888451749|              NULL|  NULL| NULL|50.68857727678354|0.35065711927354315|113987.9365746915
6|
|    min|              1.0|&amp;quot;Bungaku Sho...|Action|Movie|                1|              7.42|             369.
0|
|    max|          34240.0|   xxxHOLiC Shunmuki|Sports|   TV|             1787|              9.37|         1013917.
0|
+-------+-----------------+------------------+------+-----+-----------------+------------------+----------------
-+
```

In [9]:
```
# Convert to Pandas
pdf = df.toPandas()
```

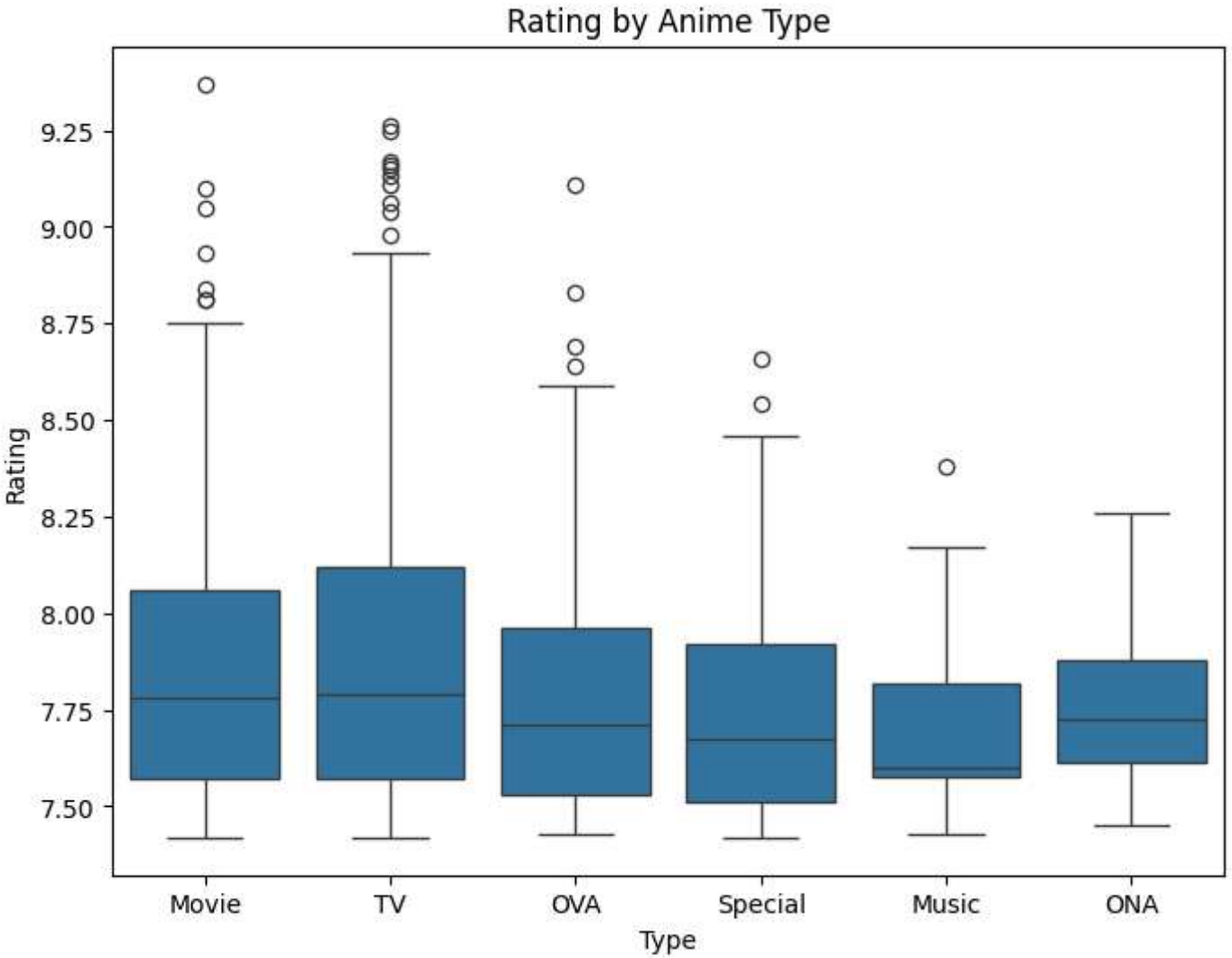
In [10]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=pdf, x="members", y="rating", hue="type")
plt.title("Members vs Rating by Type")
plt.xlabel("Members")
plt.ylabel("Rating")
plt.show()
```



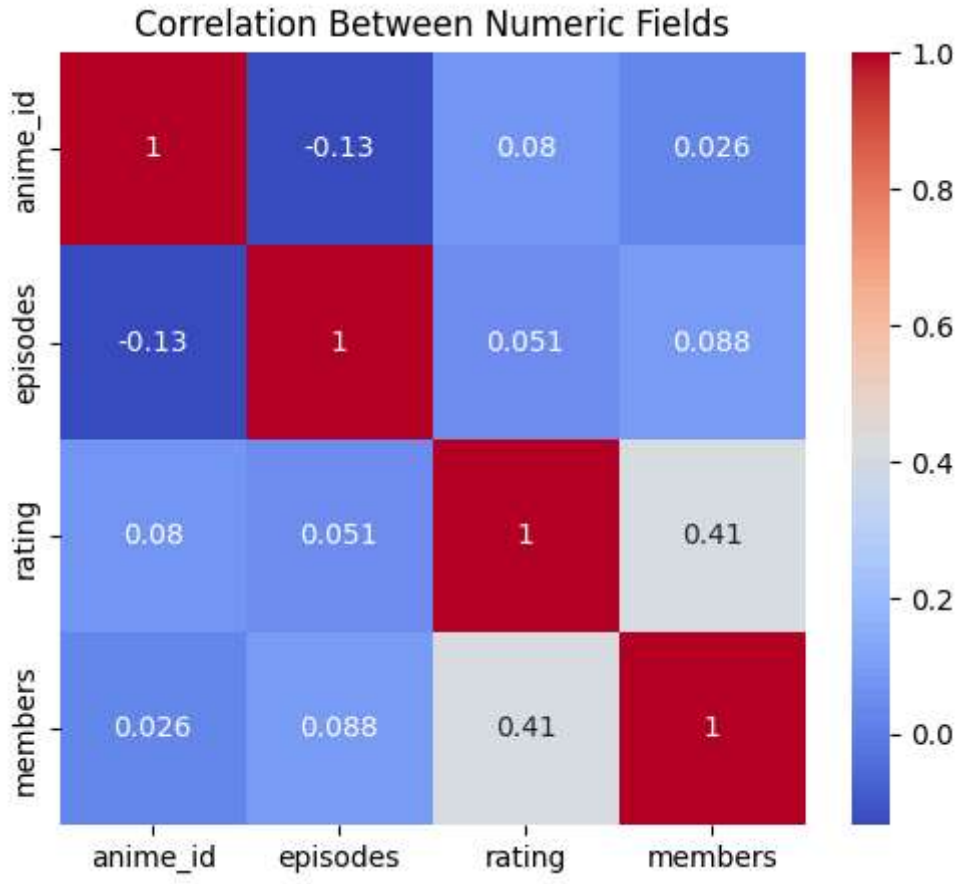
In [12]:
```
plt.figure(figsize=(8,6))
sns.histplot(data=pdf, x="episodes", bins=50, kde=True)
plt.title("Distribution of Episodes")
plt.xlabel("Episodes")
plt.ylabel("Frequency")
plt.show()
```



In [13]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=pdf, x="type", y="rating")
plt.title("Rating by Anime Type")
plt.xlabel("Type")
plt.ylabel("Rating")
plt.show()
```



In [14]:
```
plt.figure(figsize=(6,5))
sns.heatmap(pdf[["anime_id","episodes","rating","members"]].corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Between Numeric Fields")
plt.show()
```



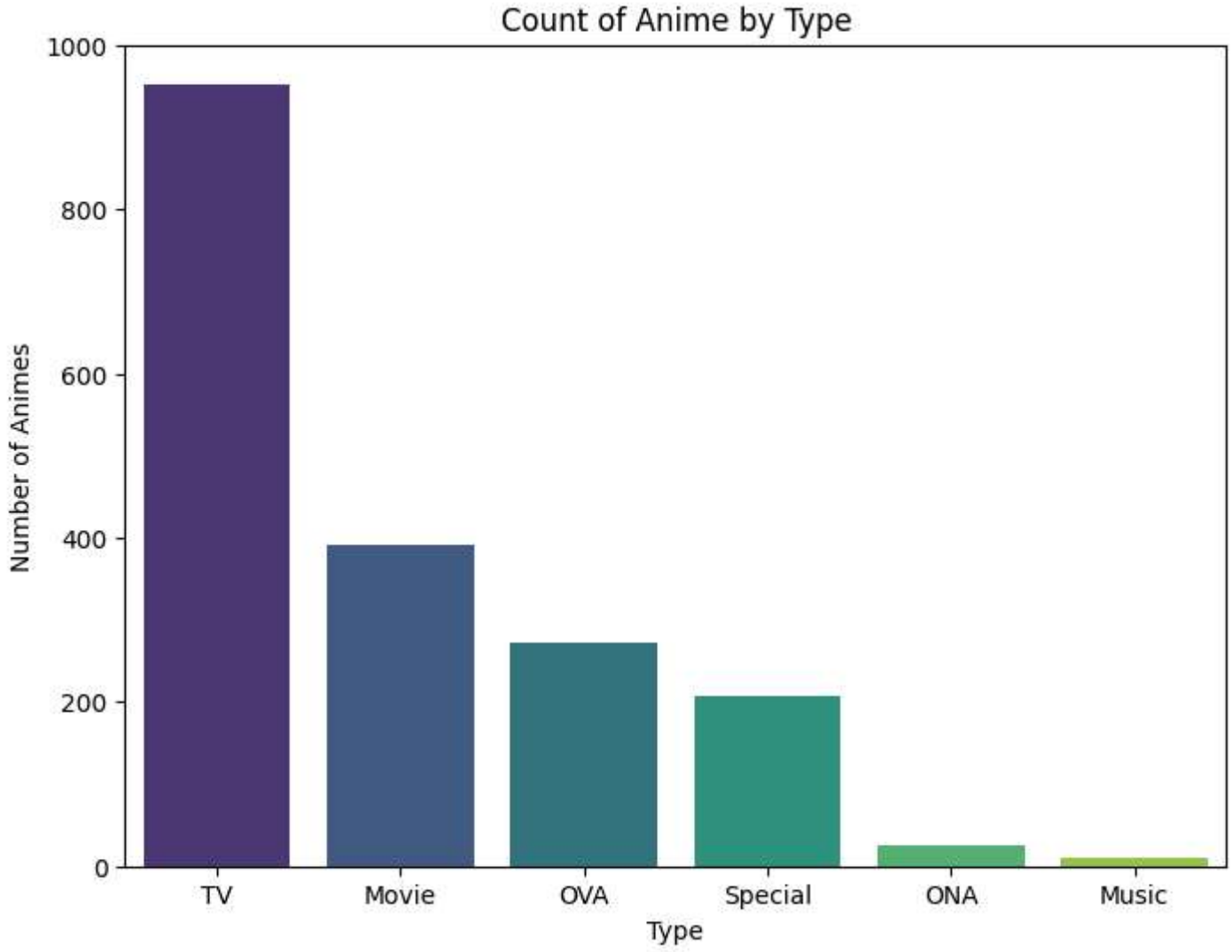
In [15]:
```python
# Average rating
avg_rating = pdf["rating"].mean()
print(f"Average Rating: {avg_rating:.2f}")

# Top genres by count
top_genres = pdf['genre'].value_counts().head(10)
print("\nTop 10 Genres:")
print(top_genres)

# Top members
top_members = pdf.sort_values(by="members", ascending=False)[["name", "members", "rating"]].head(10)
print("\nTop 10 Animes by Members:")
print(top_members)
```

```
Average Rating: 7.84

Top 10 Genres:
genre
Adventure, Comedy, Mystery, Police, Shounen      27
Comedy, School, Slice of Life                    25
Comedy, Slice of Life                            22
Comedy                                           15
Comedy, School, Shounen, Sports                  13
Comedy, Drama, Shounen, Sports                   13
Action, Drama, Mecha, Military, Sci-Fi, Space    11
Comedy, Historical, Parody                       11
Comedy, Seinen, Slice of Life                    10
Comedy, Parody, School                           10
Name: count, dtype: int64

Top 10 Animes by Members:
                                name    members  rating
40                        Death Note  1013917.0    8.71
86                 Shingeki no Kyojin   896229.0    8.54
804                  Sword Art Online   893100.0    7.83
1    Fullmetal Alchemist: Brotherhood   793665.0    9.26
159                      Angel Beats!   717796.0    8.39
19    Code Geass: Hangyaku no Lelouch   715151.0    8.83
841                           Naruto   683297.0    7.81
3                       Steins;Gate   673572.0    9.17
445                 Mirai Nikki (TV)   657190.0    8.07
131                        Toradora!   633817.0    8.45
```

In [17]:

```python
# Count of anime by type
type_count = pdf['type'].value_counts()

plt.figure(figsize=(8,6))
sns.barplot(x=type_count.index, y=type_count.values, palette="viridis")
plt.title("Count of Anime by Type")
plt.xlabel("Type")
plt.ylabel("Number of Animes")
plt.show()

print("=== Count of Anime by Type ===")
print(type_count)
```

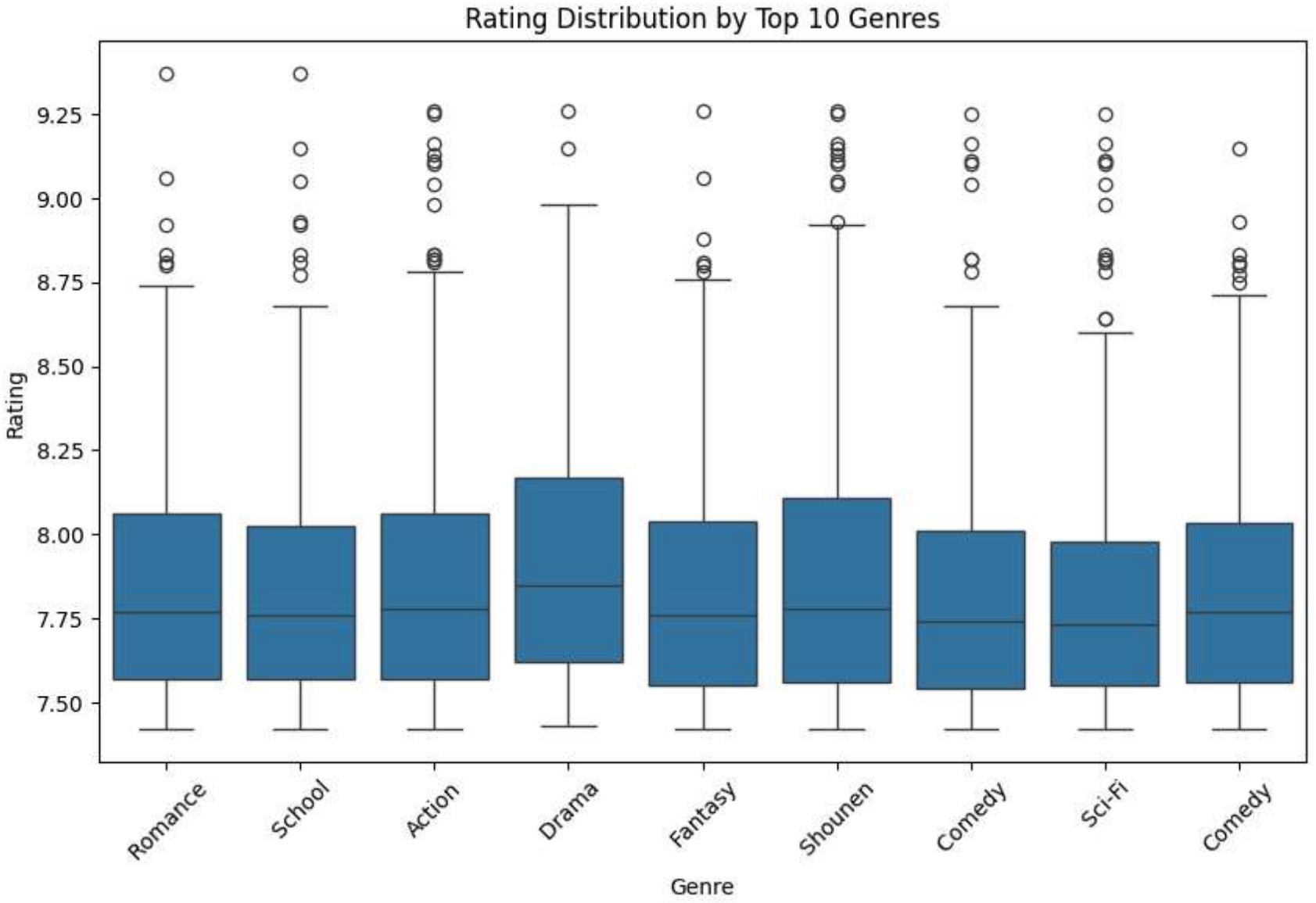
```
C:\Users\Harini\AppData\Local\Temp\ipykernel_3040\3396051903.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `h
ue` and set `legend=False` for the same effect.

  sns.barplot(x=type_count.index, y=type_count.values, palette="viridis")
```



```
=== Count of Anime by Type ===
type
TV         953
Movie      392
OVA        273
Special    208
ONA         26
Music       11
Name: count, dtype: int64
```

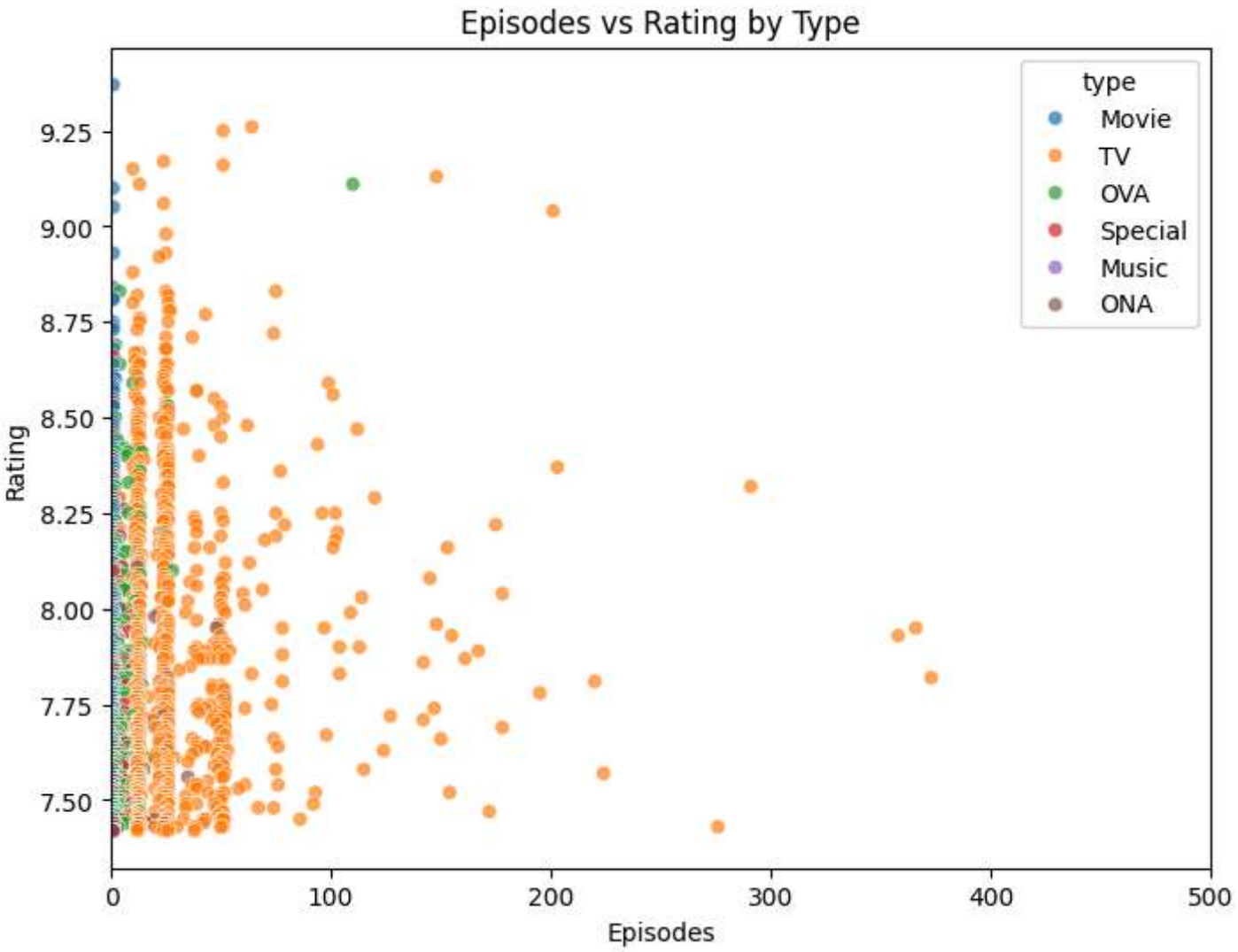
In [18]:

```python
# Explode genres (some rows have multiple genres separated by ',')
pdf['genre_list'] = pdf['genre'].fillna("").apply(lambda x: x.split(','))

# Flatten genres for plotting
import itertools
all_genres = list(itertools.chain(*pdf['genre_list']))
genre_df = pd.DataFrame(all_genres, columns=['genre'])

# Top 10 genres
top_10_genres = genre_df['genre'].value_counts().head(10).index.tolist()

# Filter only top 10 genres
pdf_top_genres = pdf[pdf['genre'].notna()]
pdf_top_genres = pdf_top_genres.explode('genre_list')
pdf_top_genres = pdf_top_genres[pdf_top_genres['genre_list'].isin(top_10_genres)]

plt.figure(figsize=(10,6))
sns.boxplot(data=pdf_top_genres, x='genre_list', y='rating')
plt.xticks(rotation=45)
plt.title("Rating Distribution by Top 10 Genres")
plt.xlabel("Genre")
plt.ylabel("Rating")
plt.show()
```



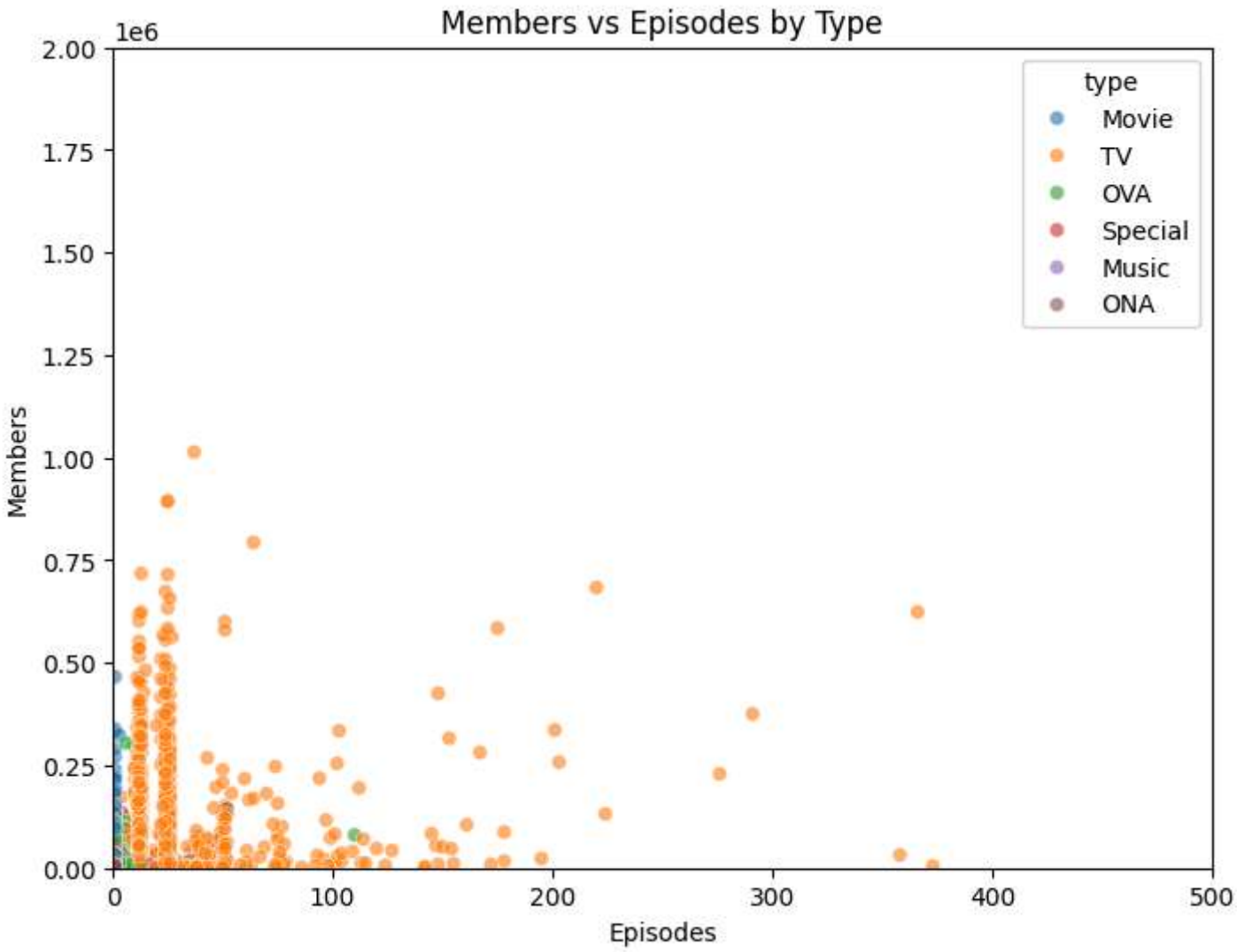
In [19]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=pdf, x='episodes', y='rating', hue='type', alpha=0.7)
plt.title("Episodes vs Rating by Type")
plt.xlabel("Episodes")
plt.ylabel("Rating")
```

```
plt.xlim(0, 500)  # Limit x-axis to ignore extreme outliers
plt.show()
```



In [20]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=pdf, x='episodes', y='members', hue='type', alpha=0.6)
plt.title("Members vs Episodes by Type")
plt.xlabel("Episodes")
plt.ylabel("Members")
plt.xlim(0, 500)
```

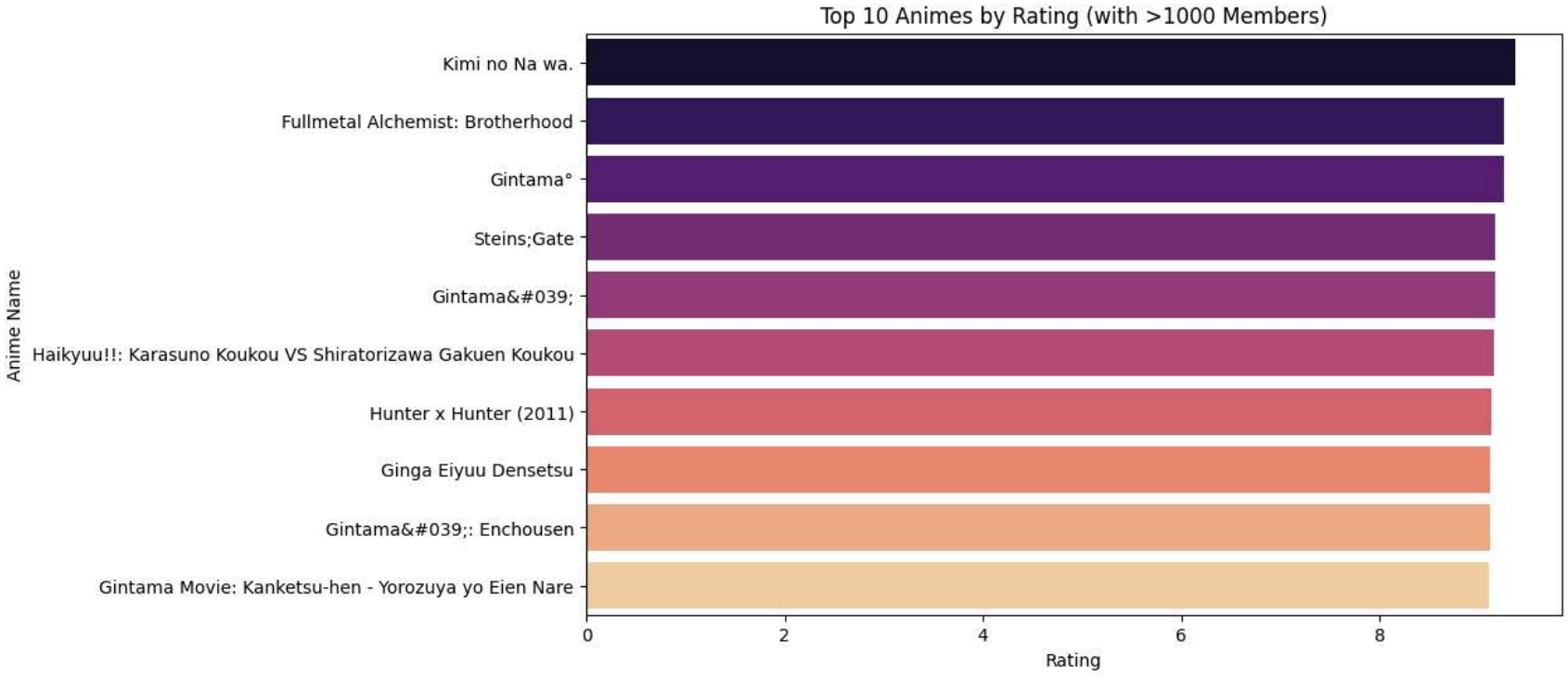
```
plt.ylim(0, 2000000)  # Limit y-axis to ignore extreme outliers
plt.show()
```



In [21]:
```
top_rated = pdf[pdf['members'] > 1000].sort_values(by='rating', ascending=False)[['name','rating','members']].head(10

plt.figure(figsize=(10,6))
sns.barplot(x='rating', y='name', data=top_rated, palette="magma")
plt.title("Top 10 Animes by Rating (with >1000 Members)")
plt.xlabel("Rating")
```

```
plt.ylabel("Anime Name")
plt.show()

print("=== Top 10 Animes by Rating (with >1000 Members) ===")
print(top_rated)
```

```
C:\Users\Harini\AppData\Local\Temp\ipykernel_3040\1038601983.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `h
ue` and set `legend=False` for the same effect.

  sns.barplot(x='rating', y='name', data=top_rated, palette="magma")
```



```
=== Top 10 Animes by Rating (with >1000 Members) ===
                                                name  rating   members
0                                    Kimi no Na wa.    9.37  200630.0
1                    Fullmetal Alchemist: Brotherhood    9.26  793665.0
2                                          Gintama°    9.25  114262.0
3                                       Steins;Gate    9.17  673572.0
4                                    Gintama&#039;    9.16  151266.0
5  Haikyuu!!: Karasuno Koukou VS Shiratorizawa Ga...    9.15   93351.0
6                              Hunter x Hunter (2011)    9.13  425855.0
7                                Ginga Eiyuu Densetsu    9.11   80679.0
9                             Gintama&#039;: Enchousen    9.11   81109.0
8  Gintama Movie: Kanketsu-hen - Yorozuya yo Eien...    9.10   72534.0
```

In [22]:
```python
print("=== Key Insights ===")

# Highest rated anime
highest_rated = pdf.loc[pdf['rating'].idxmax(), ['name','rating']]
print(f"Highest Rated Anime: {highest_rated['name']} with rating {highest_rated['rating']}")

# Most popular anime by members
most_popular = pdf.loc[pdf['members'].idxmax(), ['name','members']]
print(f"Most Popular Anime: {most_popular['name']} with {most_popular['members']} members")

# Average episodes
avg_episodes = pdf['episodes'].dropna().mean()
print(f"Average Number of Episodes: {avg_episodes:.2f}")

# Correlation between members and rating
corr = pdf[['members','rating']].corr().iloc[0,1]
print(f"Correlation between Members and Rating: {corr:.2f}")
```

```
=== Key Insights ===
Highest Rated Anime: Kimi no Na wa. with rating 9.369999885559082
Most Popular Anime: Death Note with 1013917.0 members
Average Number of Episodes: 17.82
Correlation between Members and Rating: 0.41
```

In [ ]:
```python

```