In [361…

```
#Anime_Rating_&_Data_Analysis
```

In [362…

```
import pandas as pd
import numpy as np
import pyodbc
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import matplotlib.pyplot as plt
```

In [337…

```
import pandas as pd
import numpy as np
import pyodbc
from sqlalchemy import create_engine

# Database connection
server = 'SATYEN78'
database = 'AnimeDB'
username = 'sa'
password = '123'
driver = 'ODBC Driver 17 for SQL Server'

# Create connection using SQLAlchemy
engine = create_engine(f'mssql+pyodbc://{username}:{password}@{server}/{database}?d

# Read data from Anime table
query = "SELECT * FROM Anime"
df_anime = pd.read_sql(query, con=engine)
pd.set_option('display.width', 200)  # Increase width
# Read user ratings
query = "SELECT * FROM UserRatings"
df_ratings = pd.read_sql(query, con=engine)

# Display data
print("data from Anime table")
print(df_anime.head())
print("data from UserRating table \n")
print(df_ratings.head())
```

```
data from Anime table
   AnimeID           Title                           Genre
Synopsis  Rating
0      101  Attack on Titan           Action, Drama, Fantasy            Humans fight
against Titans to survive.     9.1
1      102       Death Note  Mystery, Thriller, Supernatural  A student discovers a
notebook that grants him...     9.0
2      103        One Piece      Adventure, Fantasy, Comedy  A pirate sets sail to
find the legendary One P...     8.8
3      104           Naruto      Action, Adventure, Fantasy        A young ninja st
rives to become the Hokage.     8.5
4      105     Demon Slayer      Action, Fantasy, Adventure            A boy fights
demons to save his sister.     8.7
data from UserRating table

   UserID  AnimeID  Rating
0       1      101     9.0
1       1      102     8.5
2       1      103     7.5
3       1      121     9.5
4       2      101     9.5
```

In [338…

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Convert anime synopsis into a TF-IDF matrix
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df_anime['Synopsis'])

# Compute cosine similarity
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Create a mapping of anime titles to indices
anime_indices = pd.Series(df_anime.index, index=df_anime['Title']).drop_duplicates(
```

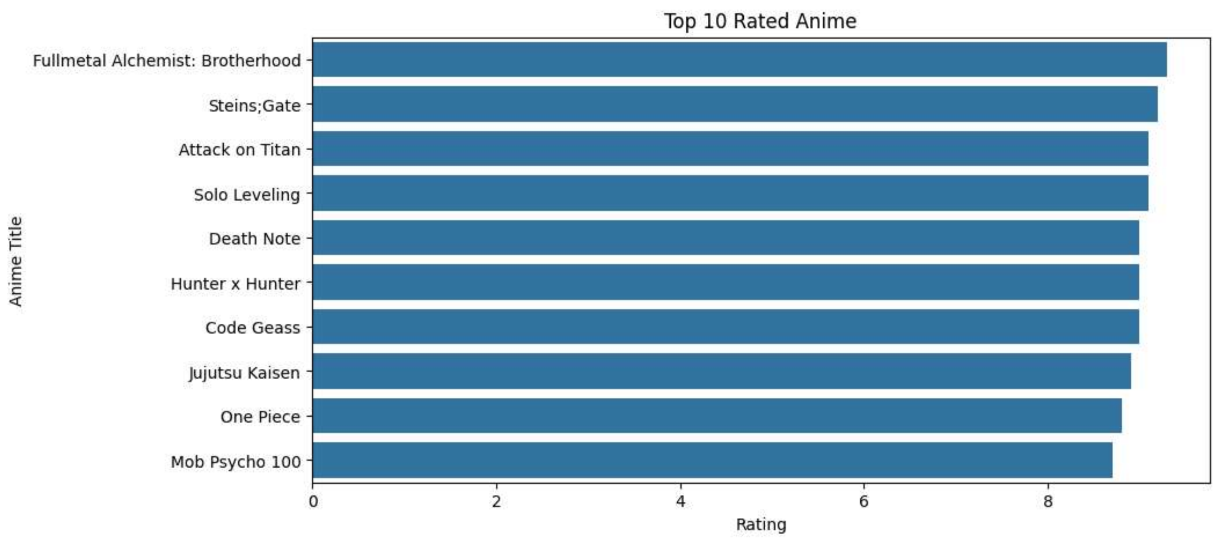
In [339…

```python
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the anime
    idx = anime_indices[title]

    # Get similarity scores
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort anime by similarity score
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get top 5 similar animes
    sim_scores = sim_scores[1:6]

    # Get anime indices
    anime_indices_list = [i[0] for i in sim_scores]

    return df_anime['Title'].iloc[anime_indices_list]

# Test the recommendation function
print(get_recommendations('Attack on Titan'))
```

```
18    Dragon Ball Z
1        Death Note
2         One Piece
3            Naruto
4     Demon Slayer
Name: Title, dtype: object
```

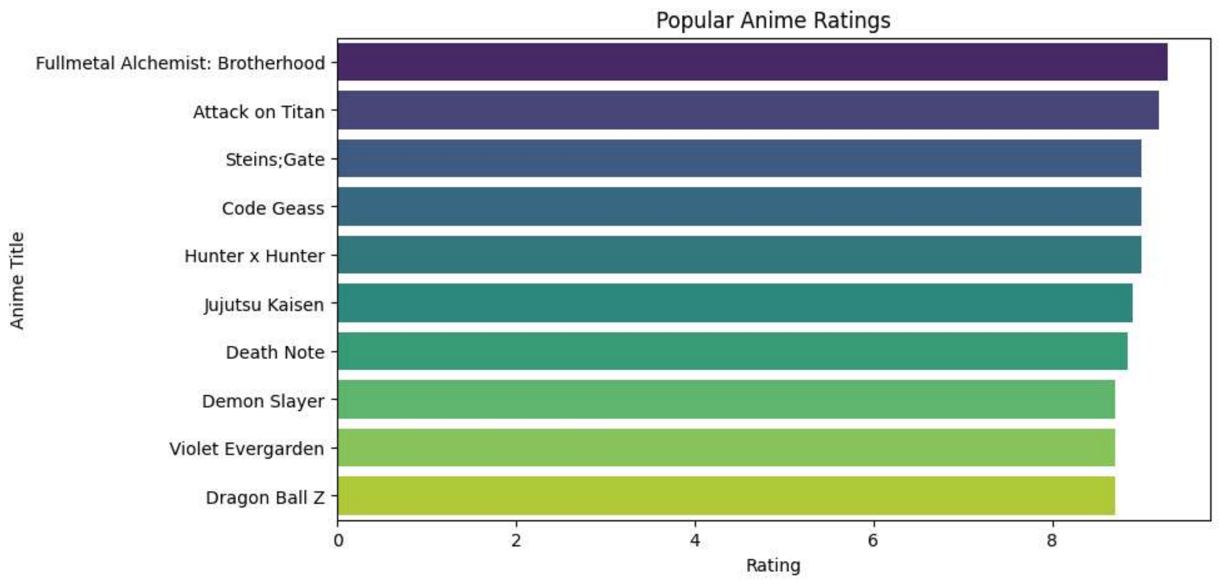
In [340…

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,5))
top_anime = df_anime.sort_values('Rating', ascending=False).head(10)
sns.barplot(x=top_anime['Rating'], y=top_anime['Title'])
plt.xlabel('Rating')
plt.ylabel('Anime Title')
plt.title('Top 10 Rated Anime')
plt.show()
```



In [341…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df_popular contains 'Rating' and 'Title' columns
plt.figure(figsize=(9,5))
sns.barplot(x=df_popular['Rating'], y=df_popular['Title'], hue=df_popular['Title'],

plt.xlabel("Rating")
plt.ylabel("Anime Title")
plt.title("Popular Anime Ratings")
plt.show()
```



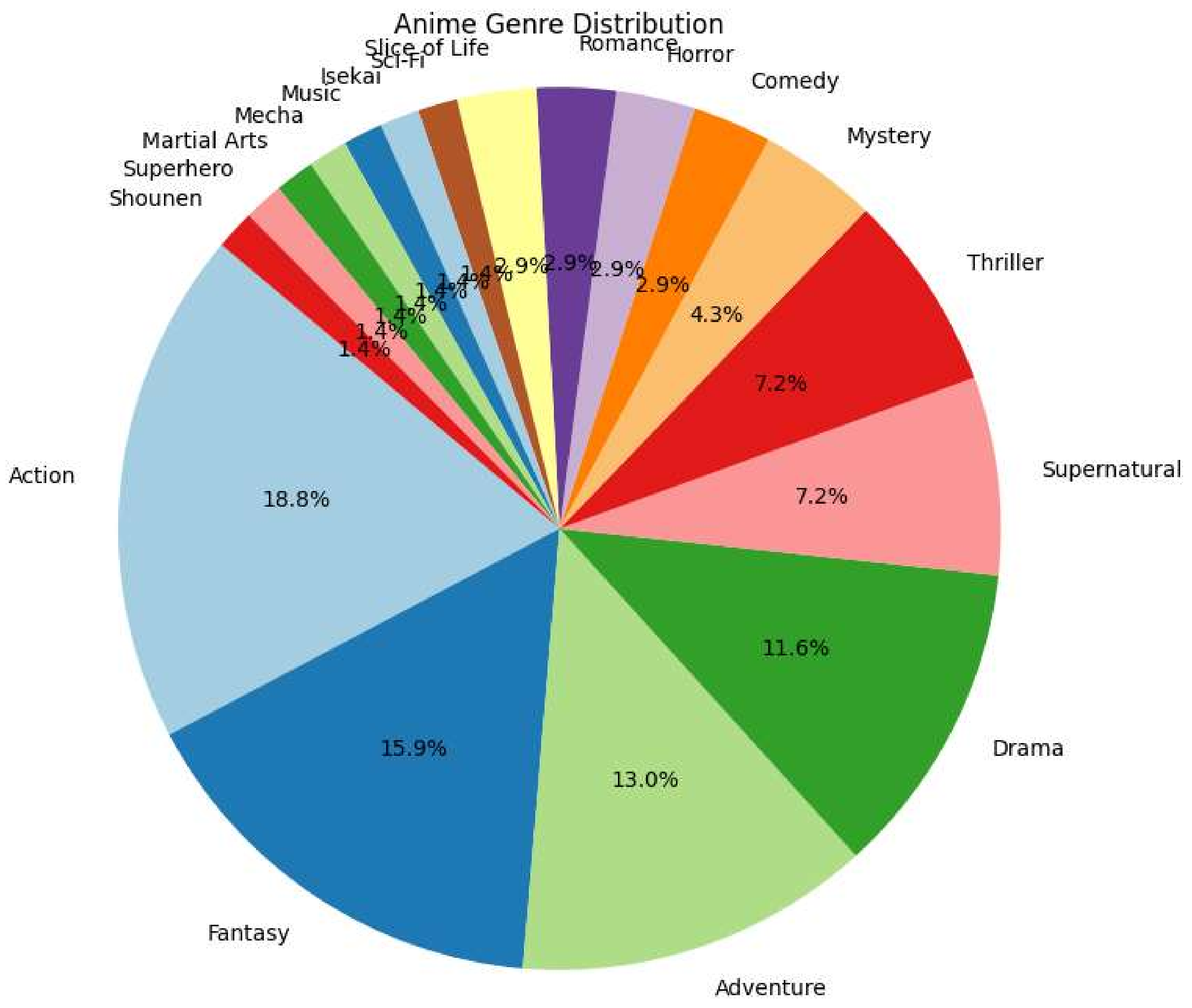
In [342…

```
def recommend_for_user(user_id):
    # Get user's highest-rated anime
    top_anime = df_ratings[df_ratings['UserID'] == user_id].sort_values('Rating', a

    if top_anime.empty:
        return "User has not rated any anime yet."

    anime_title = df_anime[df_anime['AnimeID'] == top_anime['AnimeID'].values[0]]['

    print(f"Since you liked '{anime_title}', you might also enjoy:")
    return get_recommendations(anime_title)

# Test recommendation for User 1
print(recommend_for_user(1))
```

```
Since you liked 'Solo Leveling', you might also enjoy:
11         Code Geass
0     Attack on Titan
1          Death Note
2           One Piece
3              Naruto
Name: Title, dtype: object
```

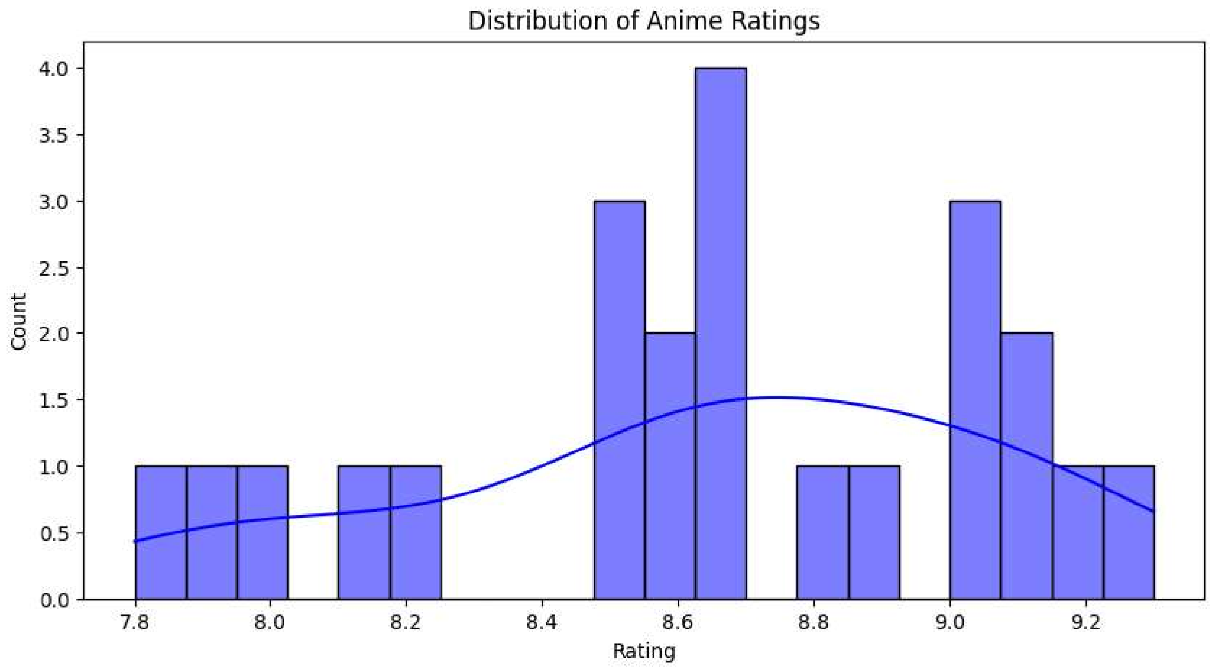
In [343…

```
import matplotlib.pyplot as plt

# Split genres into a list and count occurrences
from collections import Counter

genre_list = df_anime['Genre'].str.split(', ').explode()
genre_counts = Counter(genre_list)

# Convert to DataFrame
df_genres = pd.DataFrame(genre_counts.items(), columns=['Genre', 'Count']).sort_val

# Plot Pie Chart
plt.figure(figsize=(8,8))
plt.pie(df_genres['Count'], labels=df_genres['Genre'], autopct='%1.1f%%', colors=pl
plt.title('Anime Genre Distribution')
plt.axis('equal')  # Equal aspect ratio ensures the pie is a circle
plt.show()
```



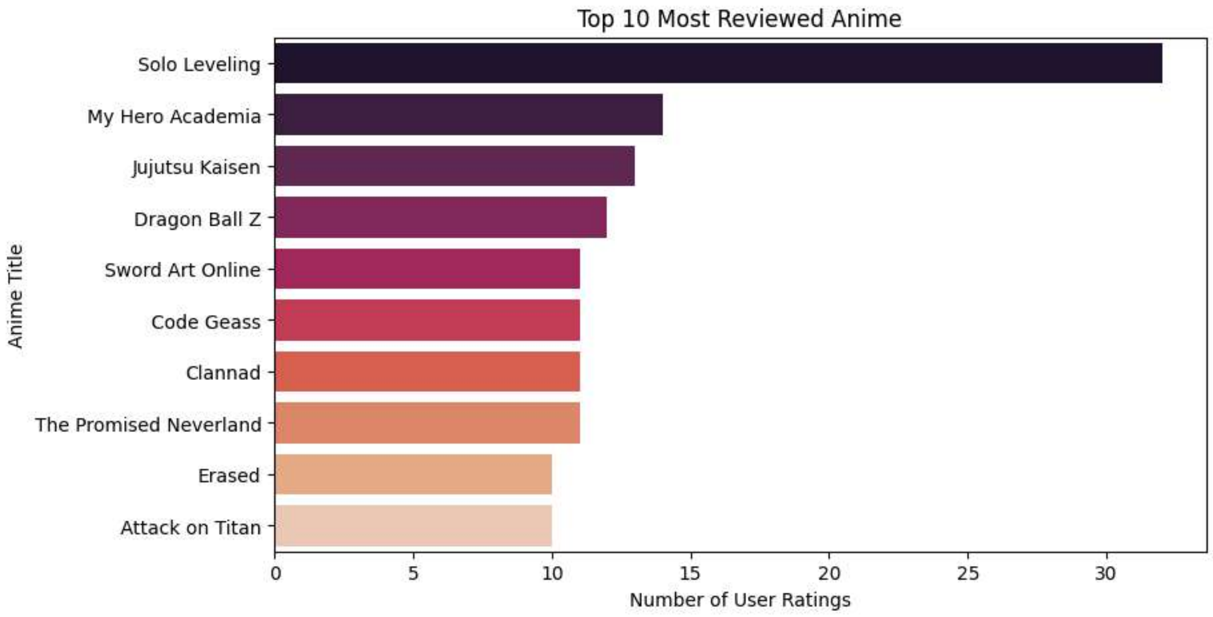
In [344…

```
plt.figure(figsize=(10,5))
sns.histplot(df_anime['Rating'], bins=20, kde=True, color='blue')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.title('Distribution of Anime Ratings')
plt.show()
```



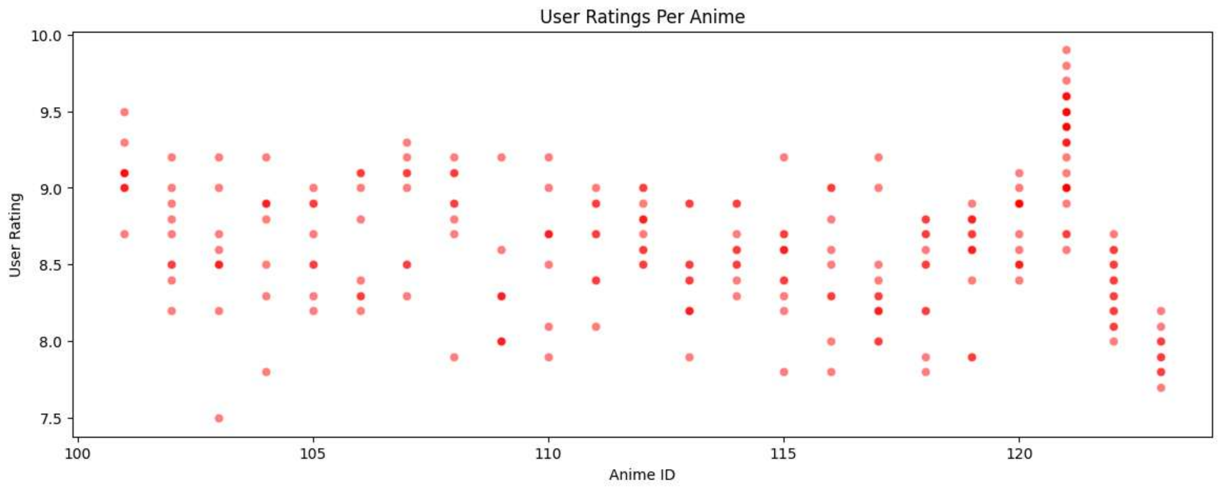
In [345…

```
df_review_count = df_ratings.groupby('AnimeID')['UserID'].count().reset_index()
df_review_count = df_review_count.merge(df_anime[['AnimeID', 'Title']], on='AnimeID
df_review_count = df_review_count.sort_values('UserID', ascending=False).head(10)

plt.figure(figsize=(9,5))

sns.barplot(x=df_review_count['UserID'], y=df_review_count['Title'],hue=df_review_c
plt.xlabel('Number of User Ratings')
plt.ylabel('Anime Title')
plt.title('Top 10 Most Reviewed Anime')
plt.show()
```



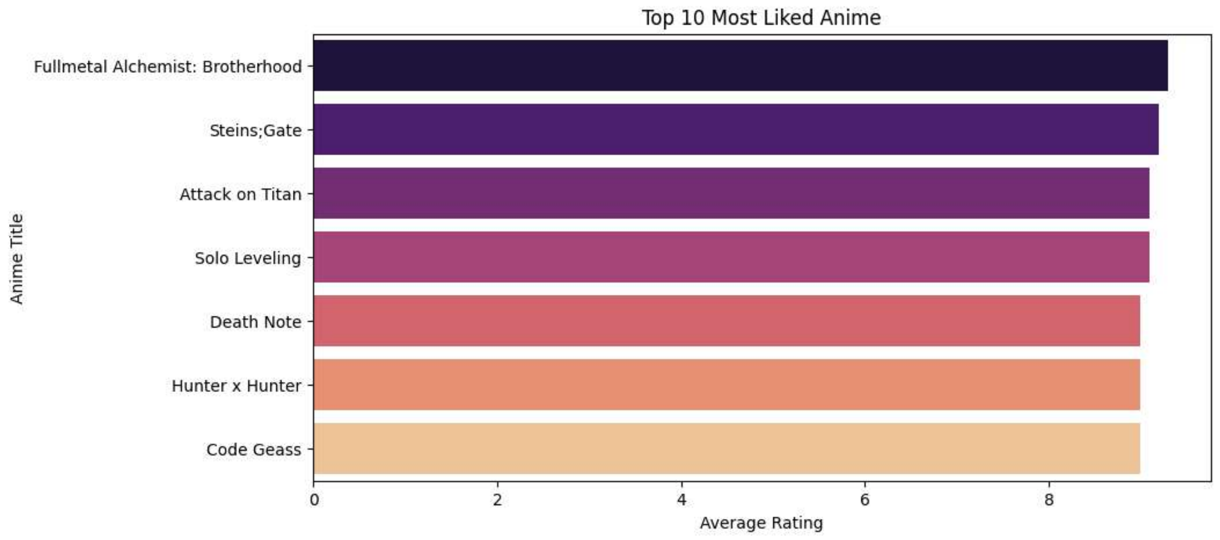
In [346…

```
plt.figure(figsize=(14,5))
sns.scatterplot(x=df_ratings['AnimeID'], y=df_ratings['Rating'], alpha=0.5, color='
plt.xlabel('Anime ID')
plt.ylabel('User Rating')
plt.title('User Ratings Per Anime')
plt.show()
```



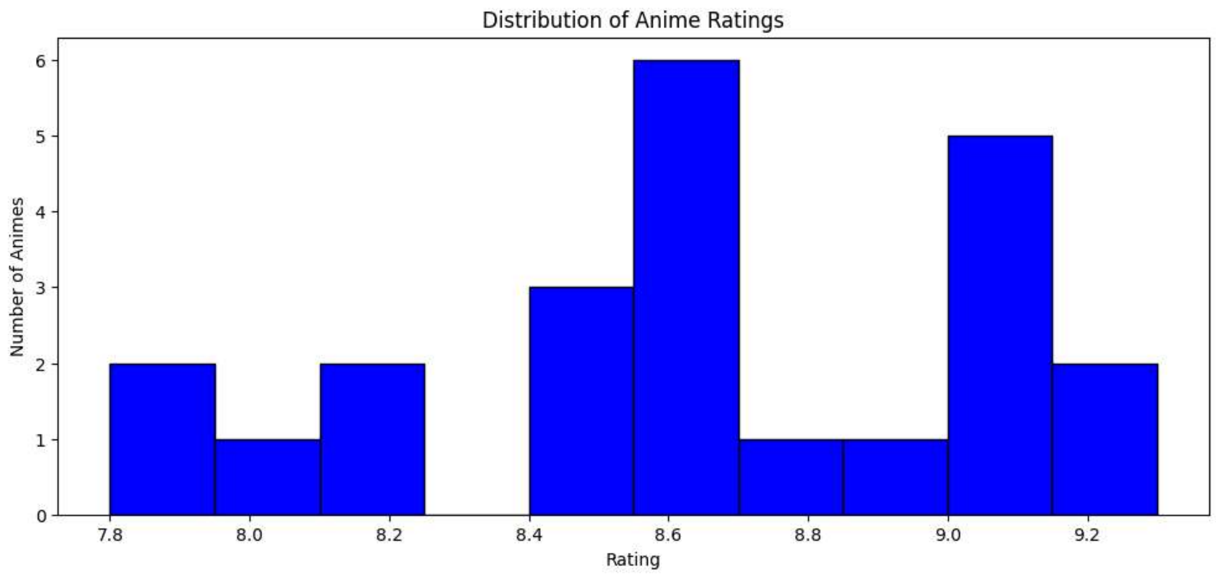
In [347…

```
df_top_rated = df_anime[df_anime['Rating'] >= 9].sort_values('Rating', ascending=Fa

plt.figure(figsize=(10,5))
sns.barplot(x=df_top_rated['Rating'], y=df_top_rated['Title'],hue=df_top_rated['Tit
plt.xlabel('Average Rating')
plt.ylabel('Anime Title')
plt.title('Top 10 Most Liked Anime')
plt.show()
```



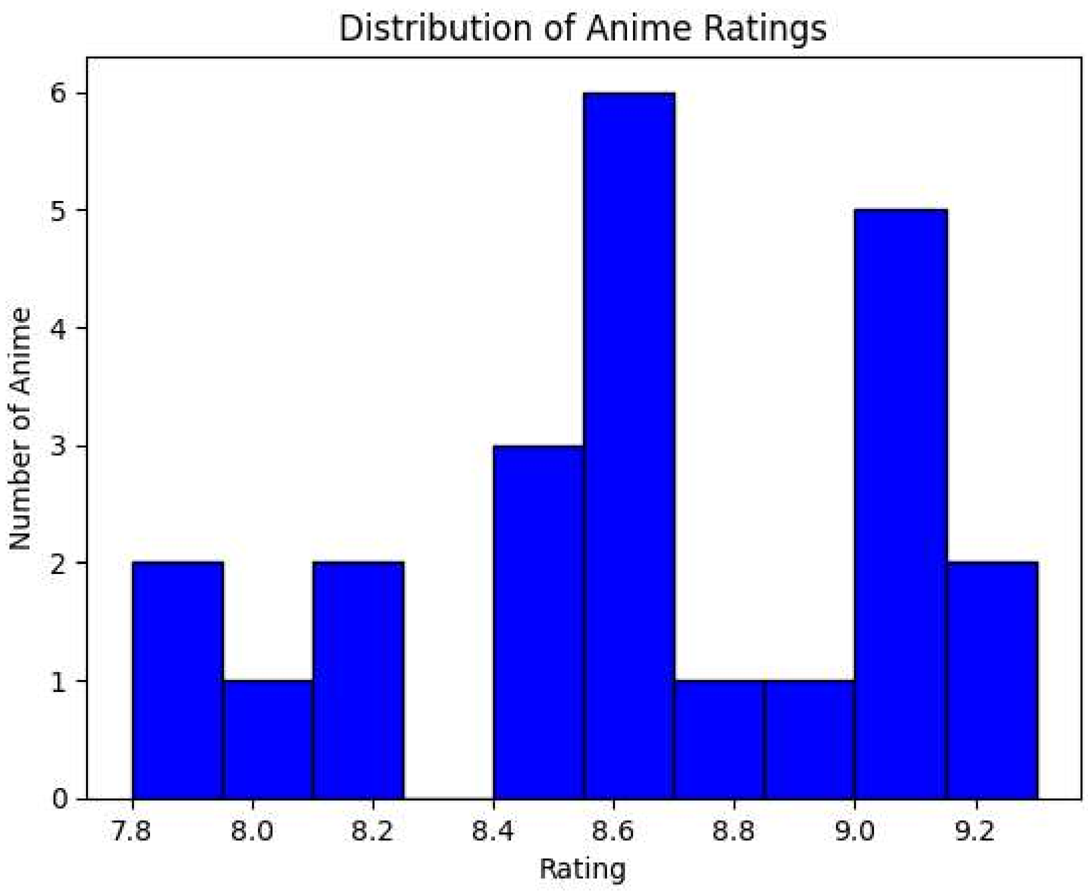
In [348…

```
plt.figure(figsize=(12,5))
plt.hist(df_anime['Rating'], bins=10, color='blue', edgecolor='black')
plt.xlabel('Rating')
plt.ylabel('Number of Animes')
plt.title('Distribution of Anime Ratings')
plt.show()
```



In [349…

```
plt.hist(df_anime['Rating'], bins=10, color='blue', edgecolor='black')
plt.xlabel('Rating')
plt.ylabel('Number of Anime')
plt.title('Distribution of Anime Ratings')
plt.show()
```



In [350…

```
import pyodbc
import sqlalchemy
import pandas as pd
```

```
# Database connection
conn = pyodbc.connect("DRIVER={SQL Server};SERVER=SATYEN78;DATABASE=AnimeDB;Trusted

# Query for Solo Leveling
query = """
SELECT u.UserID, u.Rating, a.Title, a.Genre, a.Rating AS AvgAnimeRating
FROM UserRatings u
JOIN Anime a ON u.AnimeID = a.AnimeID
WHERE a.Title = 'Solo Leveling'
"""
df = pd.read_sql(query, conn)

# Close connection
conn.close()

# Display first few rows
df.head()
```

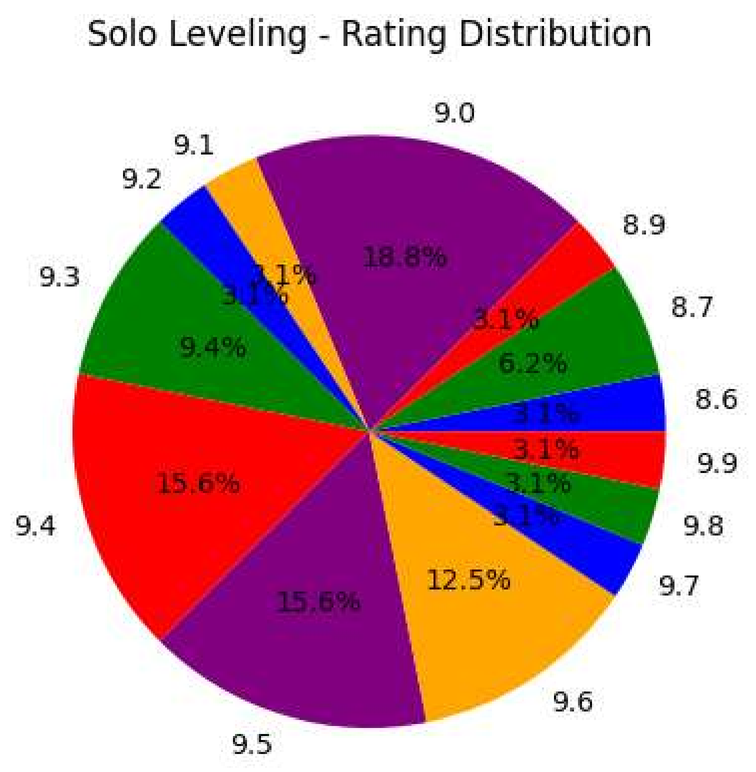
```
C:\Users\satye\AppData\Local\Temp\ipykernel_18188\1049313264.py:15: UserWarning: pan
das only supports SQLAlchemy connectable (engine/connection) or database string URI
or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  df = pd.read_sql(query, conn)
```

Out[350…

	UserID	Rating	Title	Genre	AvgAnimeRating
0	1	9.5	Solo Leveling	Action, Fantasy, Adventure	9.1
1	2	9.3	Solo Leveling	Action, Fantasy, Adventure	9.1
2	3	9.7	Solo Leveling	Action, Fantasy, Adventure	9.1
3	4	9.6	Solo Leveling	Action, Fantasy, Adventure	9.1
4	5	9.8	Solo Leveling	Action, Fantasy, Adventure	9.1

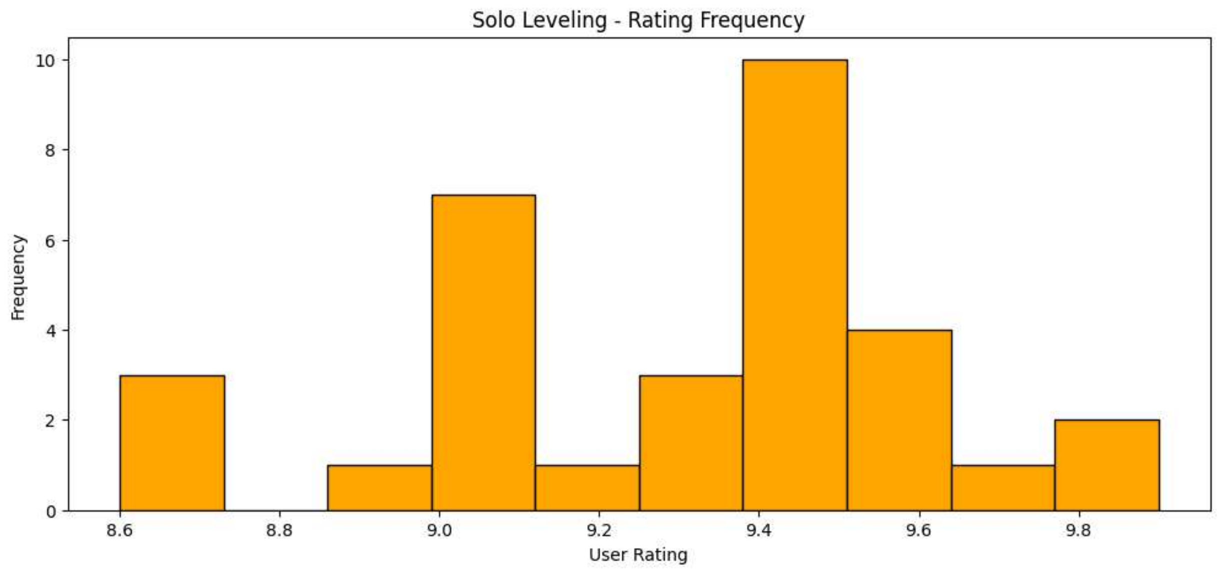
In [351…

```
plt.pie(rating_counts, labels=rating_counts.index, autopct='%1.1f%%', colors=['blue
plt.title('Solo Leveling - Rating Distribution')
plt.show()
```



In [352…

```
plt.figure(figsize=(12,5))
plt.hist(df['Rating'], bins=10, color='orange', edgecolor='black')
plt.xlabel('User Rating')
plt.ylabel('Frequency')
plt.title('Solo Leveling - Rating Frequency')
plt.show()
```



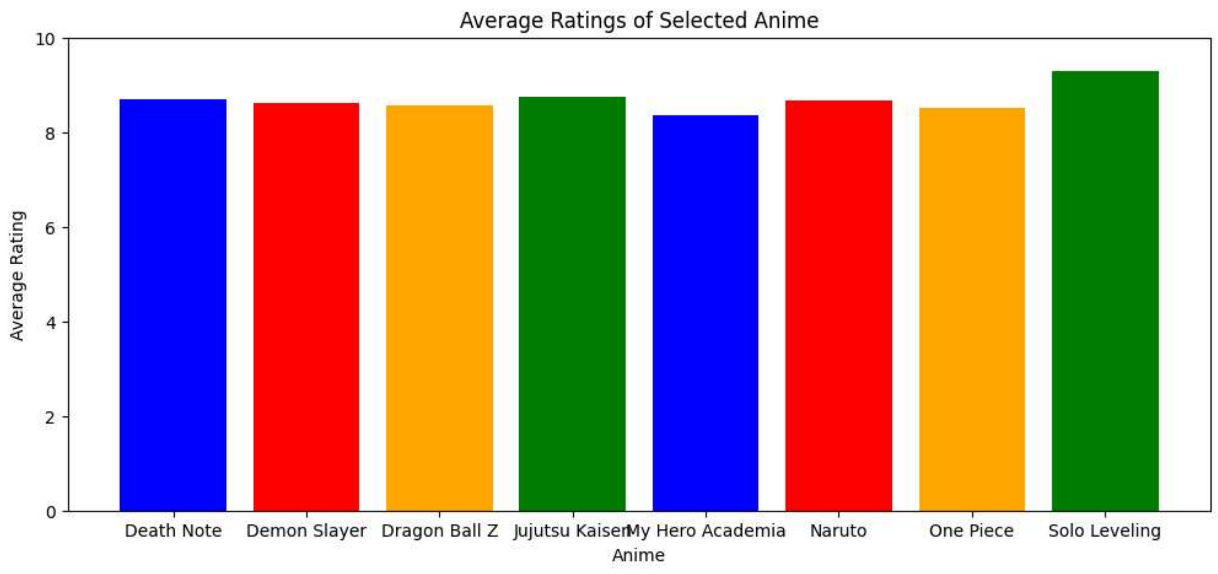
In [353…

```
import pyodbc
import pandas as pd

# Connect to MSSQL
conn = pyodbc.connect('DRIVER={SQL Server};SERVER=SATYEN78;DATABASE=AnimeDB;Trusted
# SQL Query
query = """
SELECT A.Title, UR.Rating
FROM UserRatings UR
JOIN Anime A ON UR.AnimeID = A.AnimeID
WHERE A.Title IN ('Solo Leveling', 'Dragon Ball Z', 'Naruto', 'One Piece', 'Demon S
"""

# Load into DataFrame
df = pd.read_sql(query, conn)
conn.close()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_18188\1518458631.py:16: UserWarning: pan
das only supports SQLAlchemy connectable (engine/connection) or database string URI
or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  df = pd.read_sql(query, conn)
```

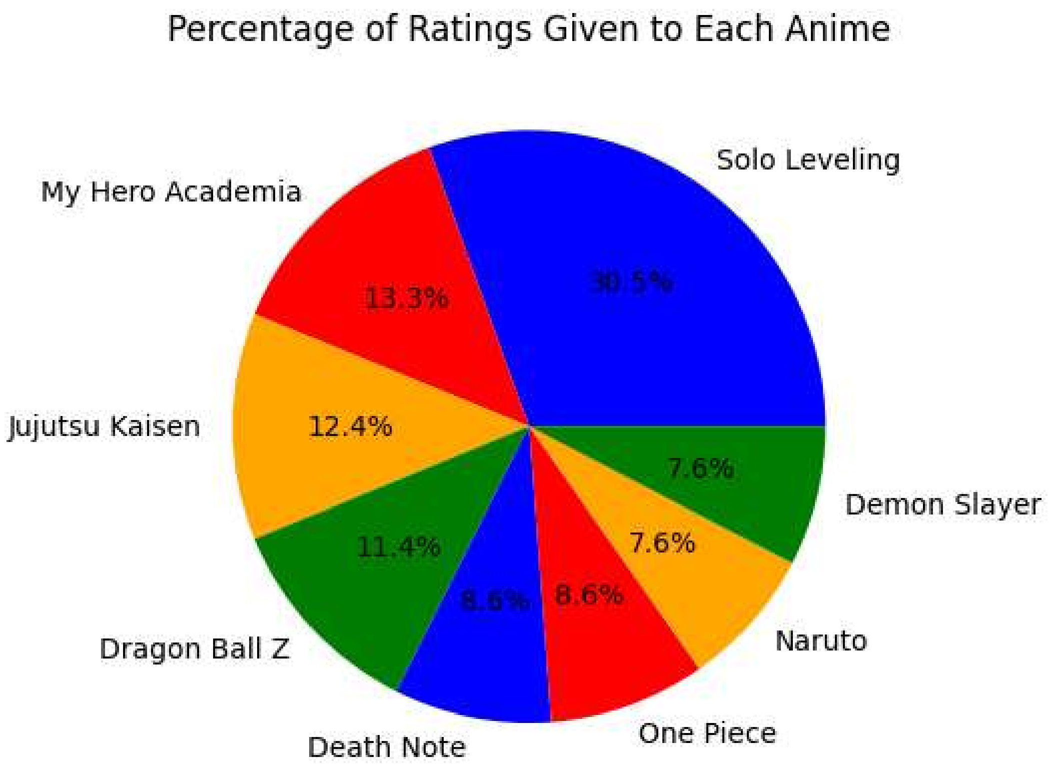
In [354…

```
import matplotlib.pyplot as plt

# Group by Title and calculate average rating
df_avg = df.groupby('Title')['Rating'].mean().reset_index()

# Plot Bar Chart
plt.figure(figsize=(12,5))
plt.bar(df_avg['Title'], df_avg['Rating'], color=['blue', 'red', 'orange', 'green']
plt.xlabel('Anime')
plt.ylabel('Average Rating')
plt.title('Average Ratings of Selected Anime')
plt.ylim(0, 10)
plt.show()
```



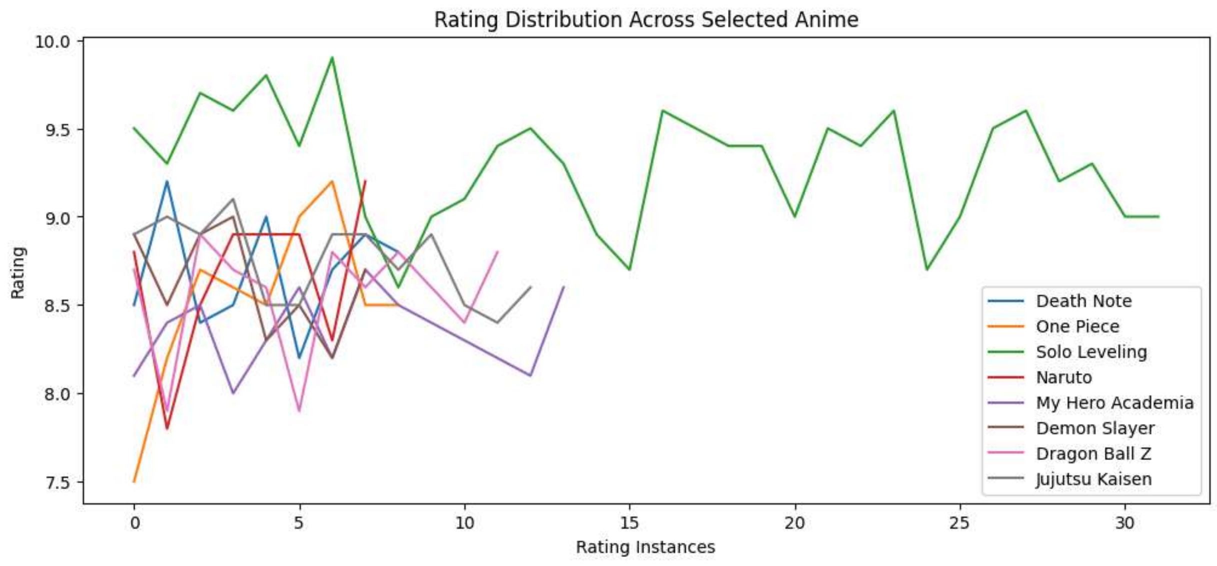
In [355…

```
df_counts = df['Title'].value_counts()
plt.pie(df_counts, labels=df_counts.index, autopct='%1.1f%%', colors=['blue', 'red'
plt.title('Percentage of Ratings Given to Each Anime')
plt.show()
```



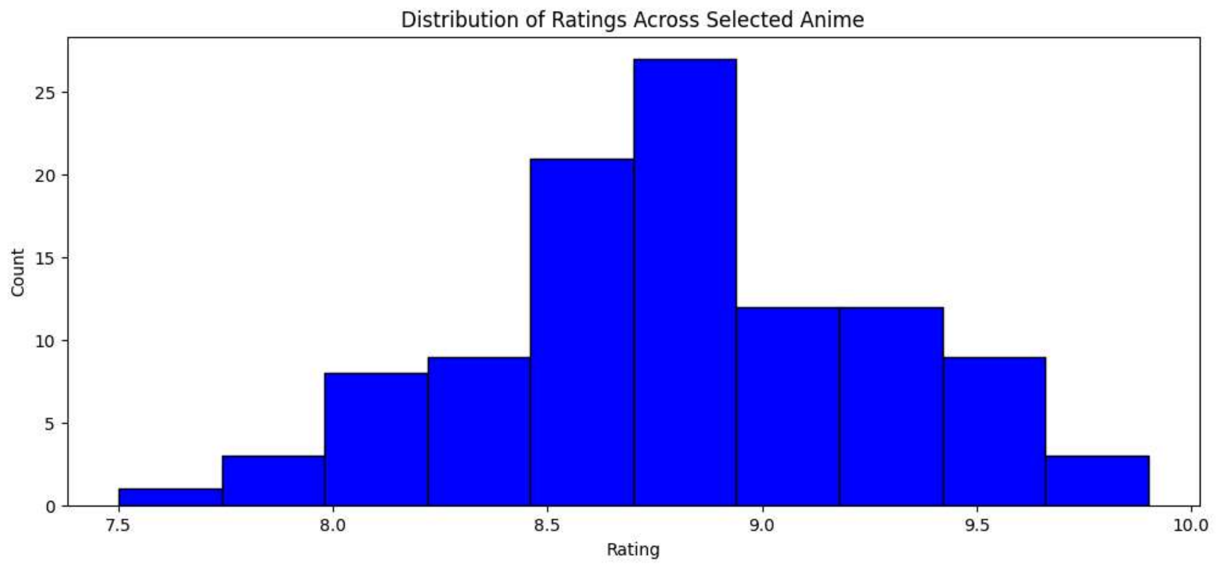
In [356…

```
import numpy as np
plt.figure(figsize=(12,5))
df_sorted = df.sort_values(by='Rating')
for anime in df['Title'].unique():
    plt.plot(np.arange(len(df[df['Title'] == anime])), df[df['Title'] == anime]['Ra

plt.xlabel('Rating Instances')
plt.ylabel('Rating')
plt.title('Rating Distribution Across Selected Anime')
plt.legend()
plt.show()
```



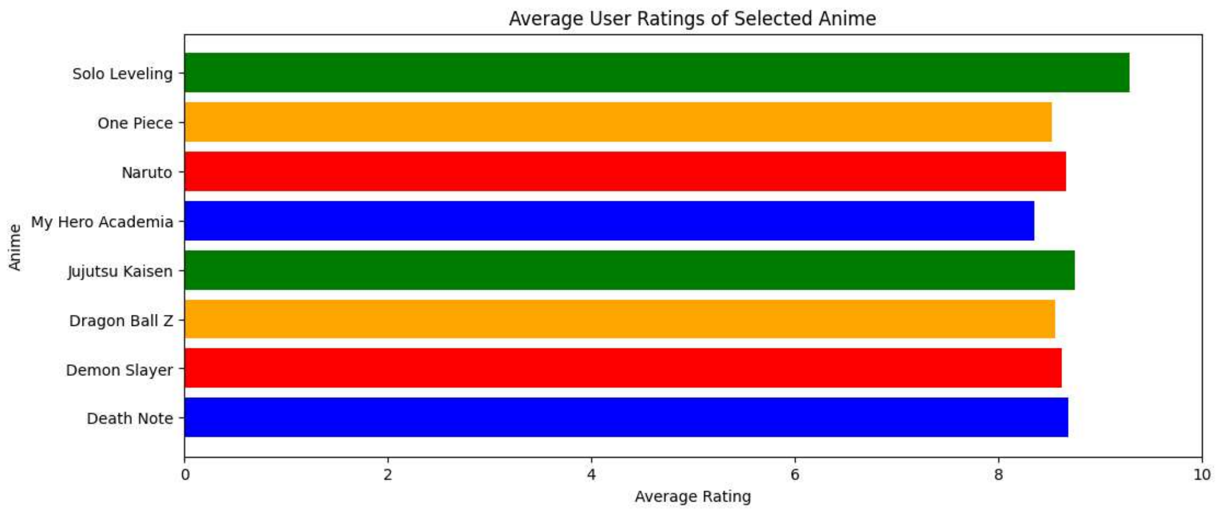
In [357…

```
plt.figure(figsize=(12,5))
plt.hist(df['Rating'], bins=10, color='blue', edgecolor='black')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.title('Distribution of Ratings Across Selected Anime')
plt.show()
```

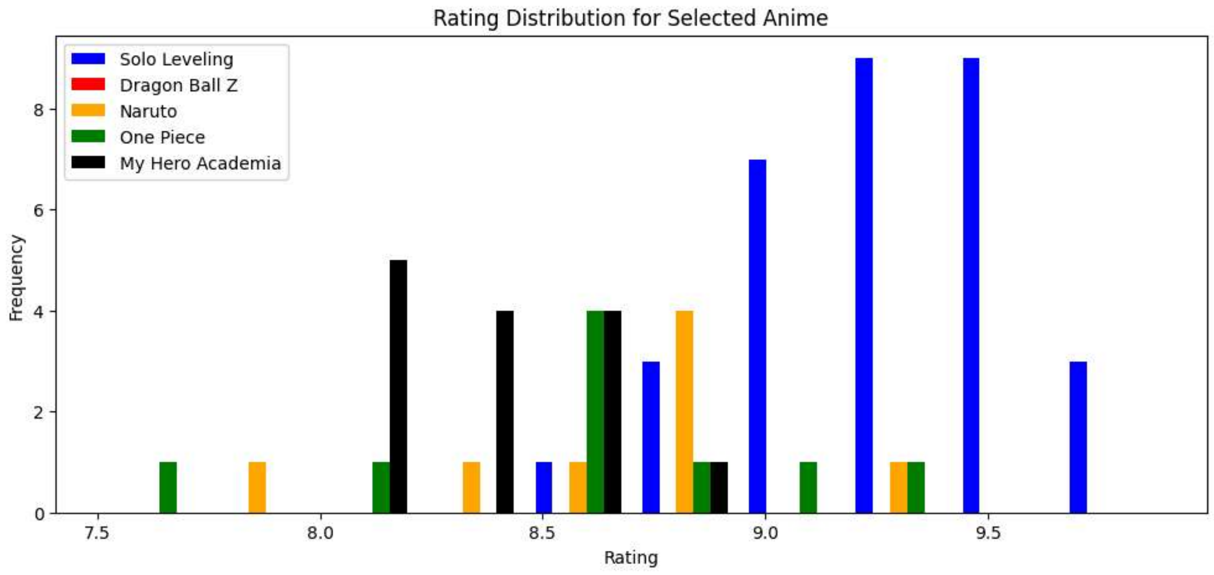


In [358…

```
import matplotlib.pyplot as plt

# Calculate average rating per anime
df_avg = df.groupby('Title')['Rating'].mean().reset_index()

# Plot
plt.figure(figsize=(12,5))
plt.barh(df_avg['Title'], df_avg['Rating'], color=['blue', 'red', 'orange', 'green'
plt.xlabel('Average Rating')
plt.ylabel('Anime')
plt.title('Average User Ratings of Selected Anime')
plt.xlim(0, 10)
plt.show()
```



In [359…
```
plt.figure(figsize=(12,5))
plt.hist([df[df['Title'] == 'Solo Leveling']['Rating'],
          df[df['Title'] == 'Dragon Ball']['Rating'],
          df[df['Title'] == 'Naruto']['Rating'],
          df[df['Title'] == 'One Piece']['Rating'],
          df[df['Title'] == 'My Hero Academia']['Rating']],
         bins=10, color=['blue', 'red', 'orange', 'green','black'], label=['Solo Le

plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Rating Distribution for Selected Anime')
plt.legend()
plt.show()
```



In [ ]: