```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
movies = pd.read_csv("/content/drive/MyDrive/movies dataset/movies.csv")
ratings = pd.read_csv("/content/drive/MyDrive/movies dataset/ratings.csv")
tags = pd.read_csv("/content/drive/MyDrive/movies dataset/tags.csv")
links = pd.read_csv("/content/drive/MyDrive/movies dataset/links.csv")
```

```
# Preview each dataset
print("Movies Data:\n", movies.head(), "\n")
print("Ratings Data:\n", ratings.head(), "\n")
print("Tags Data:\n", tags.head(), "\n")
print("Links Data:\n", links.head(), "\n")
```

```
Movies Data:
    movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                         Comedy|Drama|Romance
4                                       Comedy

Ratings Data:
    userId  movieId  rating  timestamp
0       1       17     4.0  944249077
1       1       25     1.0  944250228
2       1       29     2.0  943230976
3       1       30     5.0  944249077
4       1       32     5.0  943228858

Tags Data:
    userId  movieId          tag   timestamp
0      22    26479  Kevin Kline  1583038886
1      22    79592     misogyny  1581476297
2      22   247150   acrophobia  1622483469
3      34     2174        music  1249808064
4      34     2174        weird  1249808102

Links Data:
    movieId  imdbId   tmdbId
0        1  114709    862.0
1        2  113497   8844.0
2        3  113228  15602.0
3        4  114885  31357.0
4        5  113041  11862.0
```

```
# Check basic info
print("Movies Info:")
print(movies.info())
print("\nRatings Info:")
print(ratings.info())
print("\nTags Info:")
print(tags.info())
print("\nLinks Info:")
print(links.info())
```

```
Movies Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87585 entries, 0 to 87584
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  87585 non-null  int64
 1   title    87585 non-null  object
 2   genres   87585 non-null  object
dtypes: int64(1), object(2)
memory usage: 2.0+ MB
None
```

```
Ratings Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9516780 entries, 0 to 9516779
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   userId     int64
 1   movieId    int64
 2   rating     float64
 3   timestamp  int64
dtypes: float64(1), int64(3)
memory usage: 290.4 MB
None

Tags Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000072 entries, 0 to 2000071
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   userId     int64
 1   movieId    int64
 2   tag        object
 3   timestamp  int64
dtypes: int64(3), object(1)
memory usage: 61.0+ MB
None

Links Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87585 entries, 0 to 87584
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  87585 non-null  int64
 1   imdbId   87585 non-null  int64
 2   tmdbId   87461 non-null  float64
dtypes: float64(1), int64(2)
memory usage: 2.0 MB
None
```

```
# Check for null values
print("Missing Values:\n")
print("Movies:\n", movies.isnull().sum())
print("\nRatings:\n", ratings.isnull().sum())
print("\nTags:x\n", tags.isnull().sum())
print("\nLinks:\n", links.isnull().sum())
```

```
Missing Values:

Movies:
 movieId    0
title      0
genres     0
dtype: int64

Ratings:
 userId       0
movieId      0
rating       0
timestamp    0
dtype: int64

Tags:x
 userId        0
movieId       0
tag          17
timestamp     0
dtype: int64

Links:
 movieId      0
imdbId       0
tmdbId     124
dtype: int64
```

```
# Convert timestamp to datetime
ratings['timestamp'] = pd.to_datetime(ratings['timestamp'], unit='s')
tags['timestamp'] = pd.to_datetime(tags['timestamp'], unit='s')
```

```
# Merge movies and ratings for analysis
merged_df = pd.merge(ratings, movies, on='movieId')
print("\nMerged Data Sample:\n", merged_df.head())
```

```

Merged Data Sample:
```

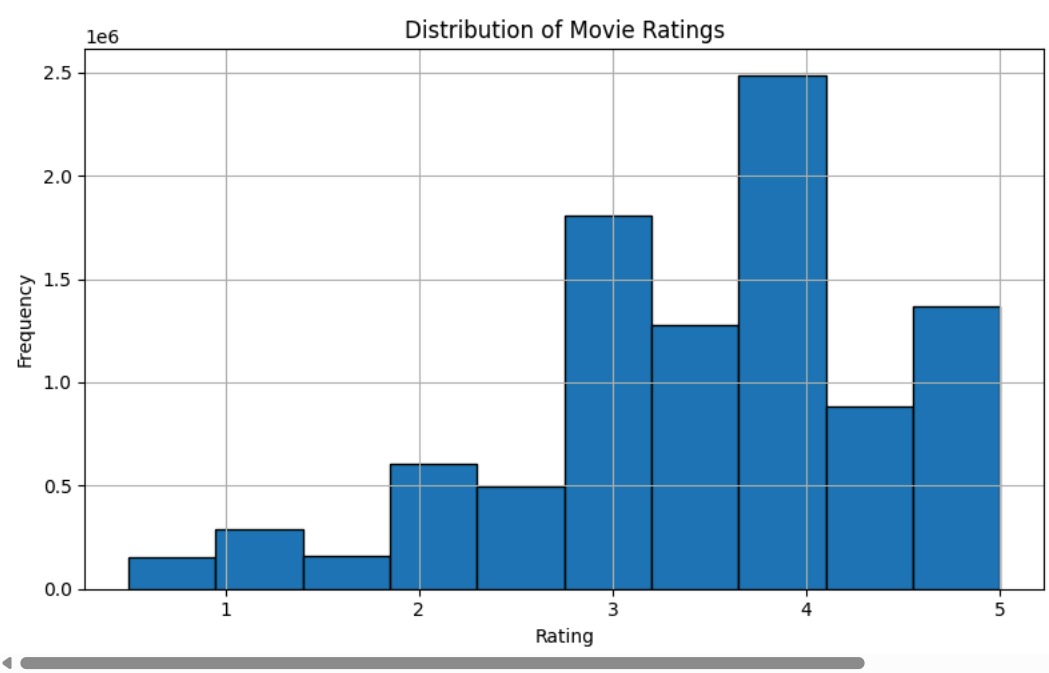
```
   userId  movieId  rating           timestamp  \
0       1       17     4.0 1999-12-03 19:24:37
1       1       25     1.0 1999-12-03 19:43:48
2       1       29     2.0 1999-11-22 00:36:16
3       1       30     5.0 1999-12-03 19:24:37
4       1       32     5.0 1999-11-22 00:00:58

                                          title  \
0                     Sense and Sensibility (1995)
1                          Leaving Las Vegas (1995)
2  City of Lost Children, The (Cité des enfants p...
3  Shanghai Triad (Yao a yao yao dao waipo qiao) ...
4           Twelve Monkeys (a.k.a. 12 Monkeys) (1995)

                                genres
0                        Drama|Romance
1                        Drama|Romance
2  Adventure|Drama|Fantasy|Mystery|Sci-Fi
3                          Crime|Drama
4                 Mystery|Sci-Fi|Thriller
```

```
# Save merged data (optional)
merged_df.to_csv("merged_movies_ratings.csv", index=False)
```

```
# Basic statistics on ratings
print("\nRating Stats:\n", merged_df['rating'].describe())
```

```
Rating Stats:
 count    9.516780e+06
mean     3.539277e+00
std      1.059430e+00
min      5.000000e-01
25%      3.000000e+00
50%      3.500000e+00
75%      4.000000e+00
max      5.000000e+00
Name: rating, dtype: float64
```

```
# Plot histogram of ratings
plt.figure(figsize=(8, 5))
plt.hist(merged_df['rating'], bins=10, edgecolor='black')
plt.title("Distribution of Movie Ratings")
plt.xlabel("Rating")
plt.ylabel("Frequency")
plt.grid(True)
plt.tight_layout()
plt.show()
```

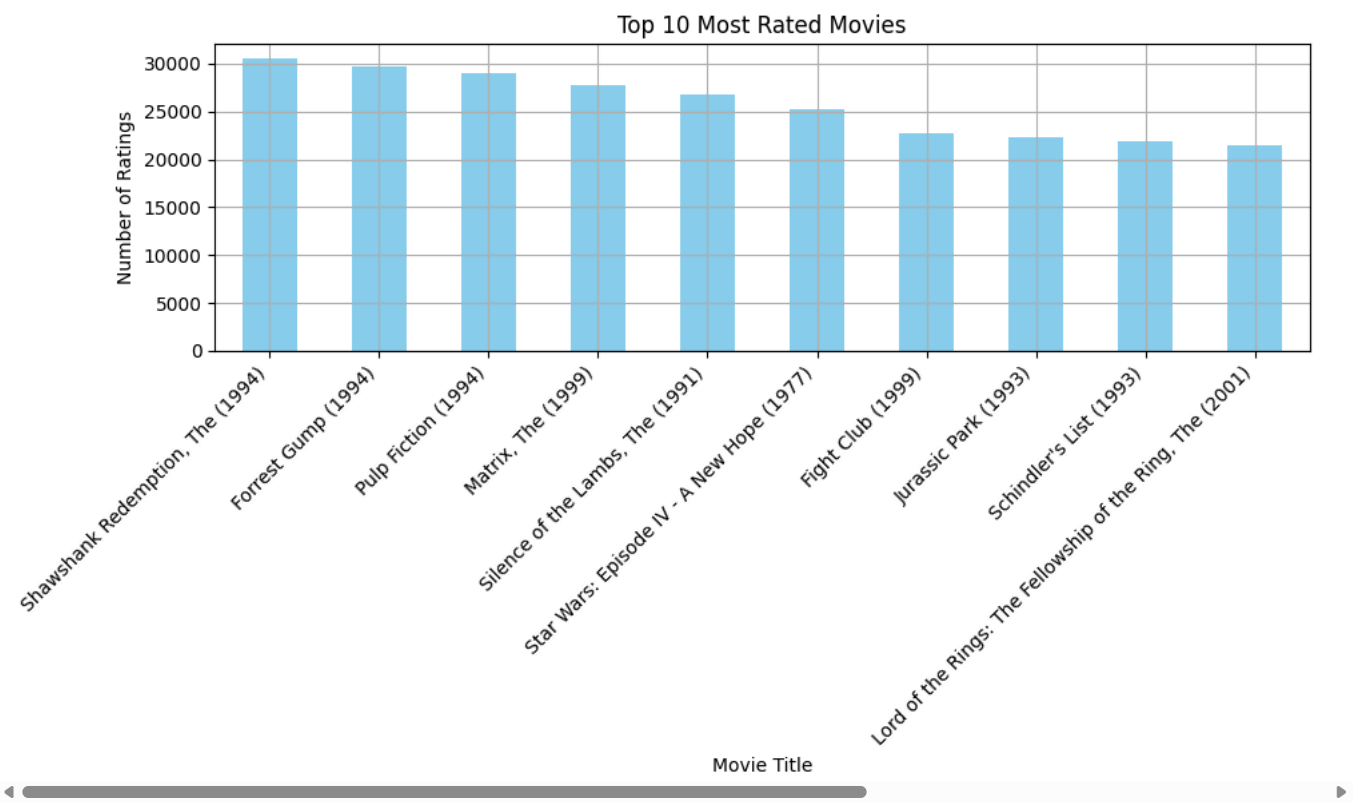



```
# Load merged dataset if not already loaded
merged_df = pd.read_csv("merged_movies_ratings.csv")
```

```
# 1. Most Rated Movies
most_rated = merged_df.groupby('title').size().sort_values(ascending=False).head(10)
print("\nTop 10 Most Rated Movies:\n", most_rated)
```

```
Top 10 Most Rated Movies:
 title
Shawshank Redemption, The (1994)                            30506
Forrest Gump (1994)                                         29769
Pulp Fiction (1994)                                         28995
Matrix, The (1999)                                          27695
Silence of the Lambs, The (1991)                            26746
Star Wars: Episode IV - A New Hope (1977)                   25296
Fight Club (1999)                                           22757
Jurassic Park (1993)                                        22374
Schindler's List (1993)                                     21831
Lord of the Rings: The Fellowship of the Ring, The (2001)    21525
dtype: int64
```

```
# Bar chart: Most Rated Movies
plt.figure(figsize=(10,6))
most_rated.plot(kind='bar', color='skyblue')
plt.title("Top 10 Most Rated Movies")
plt.xlabel("Movie Title")
plt.ylabel("Number of Ratings")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```

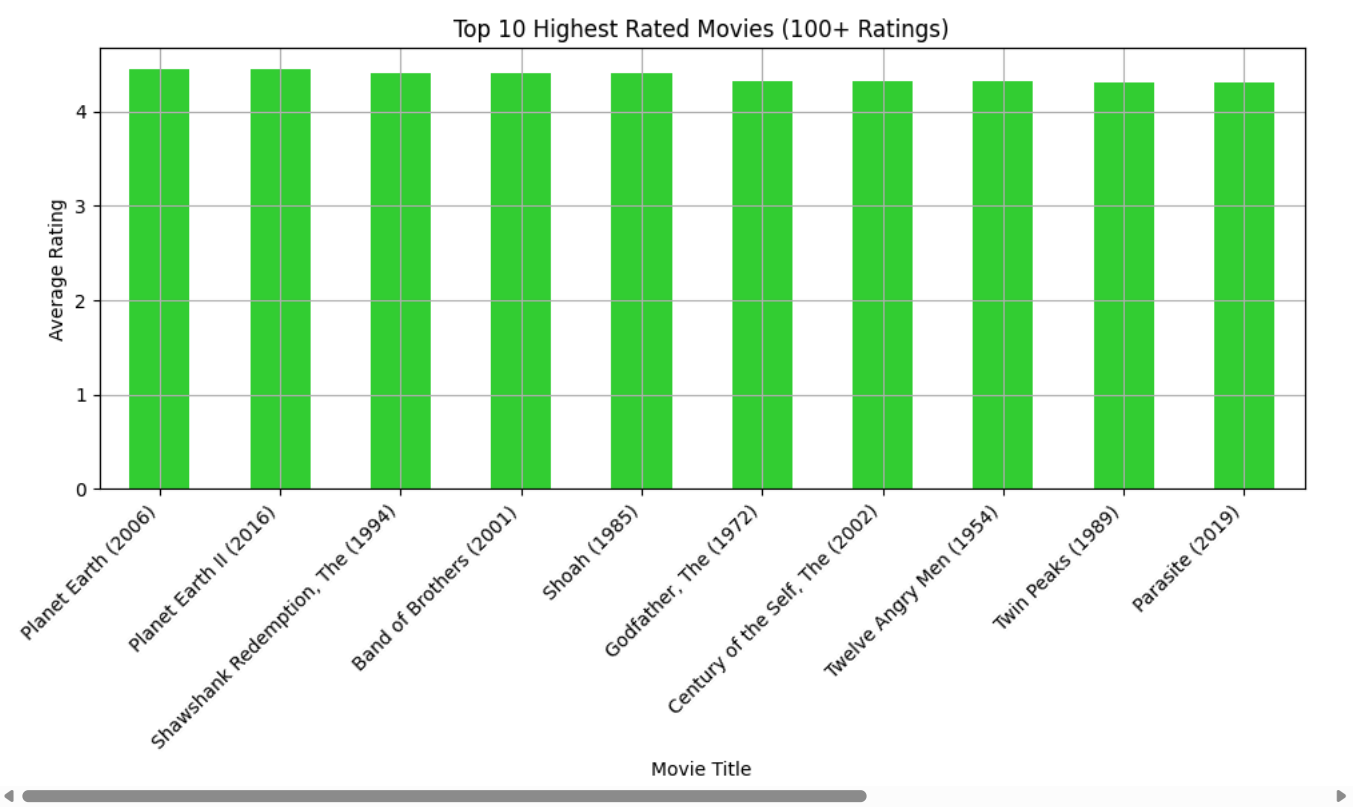



```
#top 10 highest rated movies
ratings_count = merged_df.groupby('title')['rating'].count()
average_ratings = merged_df.groupby('title')['rating'].mean()
popular_movies = ratings_count[ratings_count >= 100]
top_rated_movies = average_ratings[popular_movies.index]
top_rated_movies = top_rated_movies.sort_values(ascending=False).head(10)
print("\nTop 10 Highest Rated Movies (100+ ratings):\n", top_rated_movies)
```

```
Top 10 Highest Rated Movies (100+ ratings):
 title
Planet Earth (2006)                 4.445898
Planet Earth II (2016)              4.442755
Shawshank Redemption, The (1994)    4.407133
Band of Brothers (2001)             4.404732
Shoah (1985)                        4.403846
Godfather, The (1972)               4.325191
Century of the Self, The (2002)     4.322034
Twelve Angry Men (1954)             4.318898
Twin Peaks (1989)                   4.309451
```

```
Parasite (2019)                    4.304131
Name: rating, dtype: float64
```

```
# Bar chart: Highest Rated Movies
plt.figure(figsize=(10,6))
top_rated_movies.plot(kind='bar', color='limegreen')
plt.title("Top 10 Highest Rated Movies (100+ Ratings)")
plt.xlabel("Movie Title")
plt.ylabel("Average Rating")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```

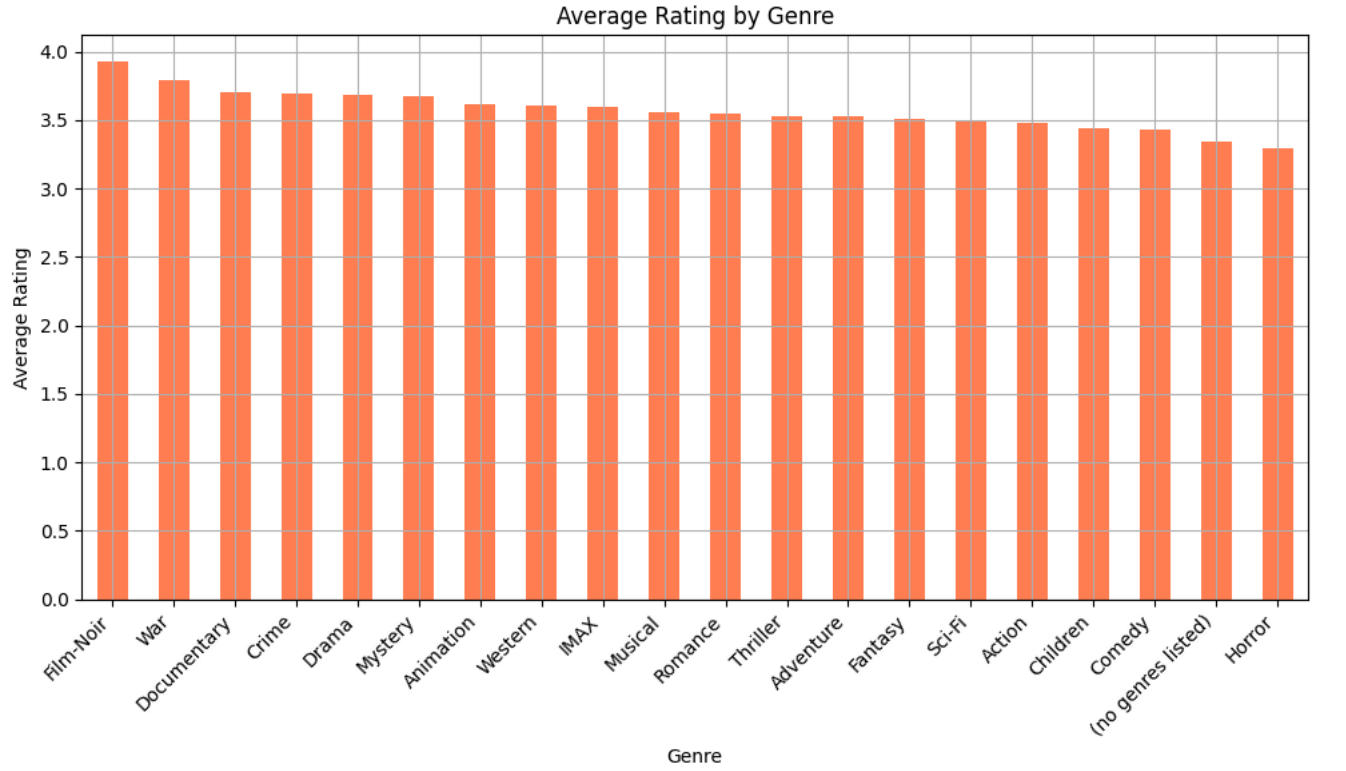



```
# 3. Top Genres by Average Rating
# Expand genre column (movies have multiple genres separated by "|")
genre_df = merged_df.copy()
genre_df['genres'] = genre_df['genres'].str.split('|')
genre_df = genre_df.explode('genres')
```

```
# Group by genre and calculate average rating
genre_ratings = genre_df.groupby('genres')['rating'].mean().sort_values(ascending=False)
print("\nAverage Ratings by Genre:\n", genre_ratings)
```

```
Average Ratings by Genre:
 genres
Film-Noir              3.923882
War                    3.796043
Documentary            3.699971
Crime                  3.690724
Drama                  3.682076
Mystery                3.672023
Animation              3.618760
Western                3.602119
IMAX                   3.596603
Musical                3.556245
Romance                3.543914
Thriller               3.530282
Adventure              3.525175
Fantasy                3.512448
Sci-Fi                 3.492283
Action                 3.476816
Children               3.438379
Comedy                 3.429893
(no genres listed)     3.339110
Horror                 3.298118
Name: rating, dtype: float64
```

```
# Bar chart: Top Genres
plt.figure(figsize=(10,6))
genre_ratings.plot(kind='bar', color='coral')
plt.title("Average Rating by Genre")
plt.xlabel("Genre")
plt.ylabel("Average Rating")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```




```
#1. Number of ratings per user
ratings_per_user = merged_df['userId'].value_counts()
print("\nTop 10 Most Active Users:\n", ratings_per_user.head(10))
```

```
Top 10 Most Active Users:
 userId
17035    9577
55653    9178
10202    7748
49305    7488
22744    7372
7858     7322
14674    6407
53192    6265
57304    6061
43703    5784
Name: count, dtype: int64
```

```
# 2. Average rating per user
avg_rating_per_user = merged_df.groupby('userId')['rating'].mean()
print("\nAverage Rating by Top Users:\n", avg_rating_per_user.sort_values(ascending=False).head(10))
```

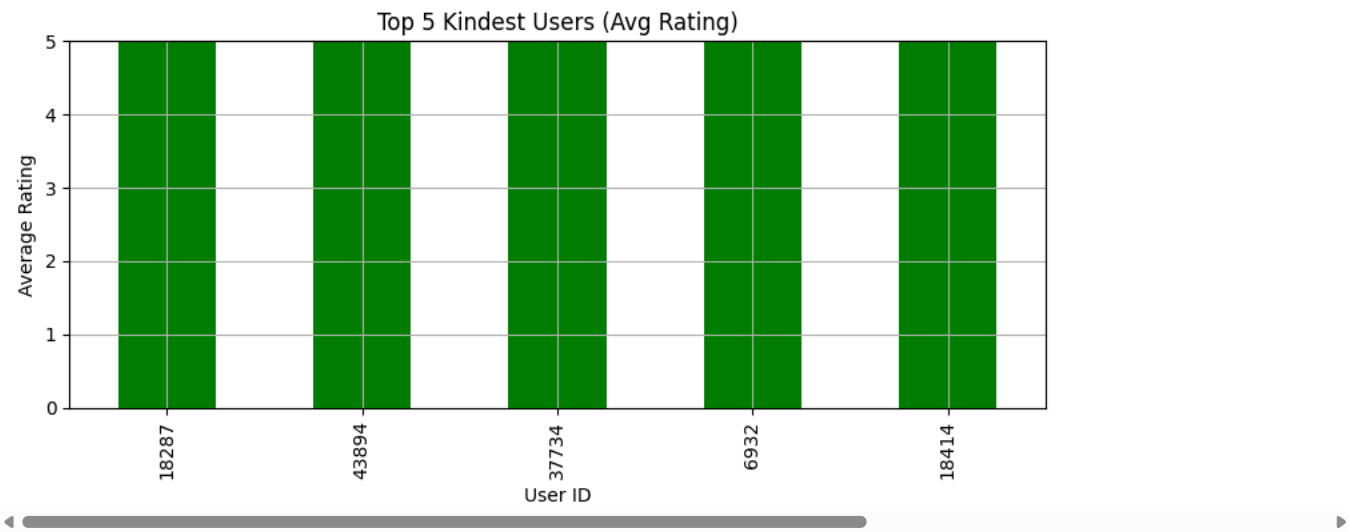
```
Average Rating by Top Users:
 userId
18287    5.0
43894    5.0
37734    5.0
6932     5.0
18414    5.0
10811    5.0
44003    5.0
16972    5.0
37712    5.0
10483    5.0
Name: rating, dtype: float64
```

```
# 3. Harshest and Kindest Users
kindest_users = avg_rating_per_user.sort_values(ascending=False).head(5)
```

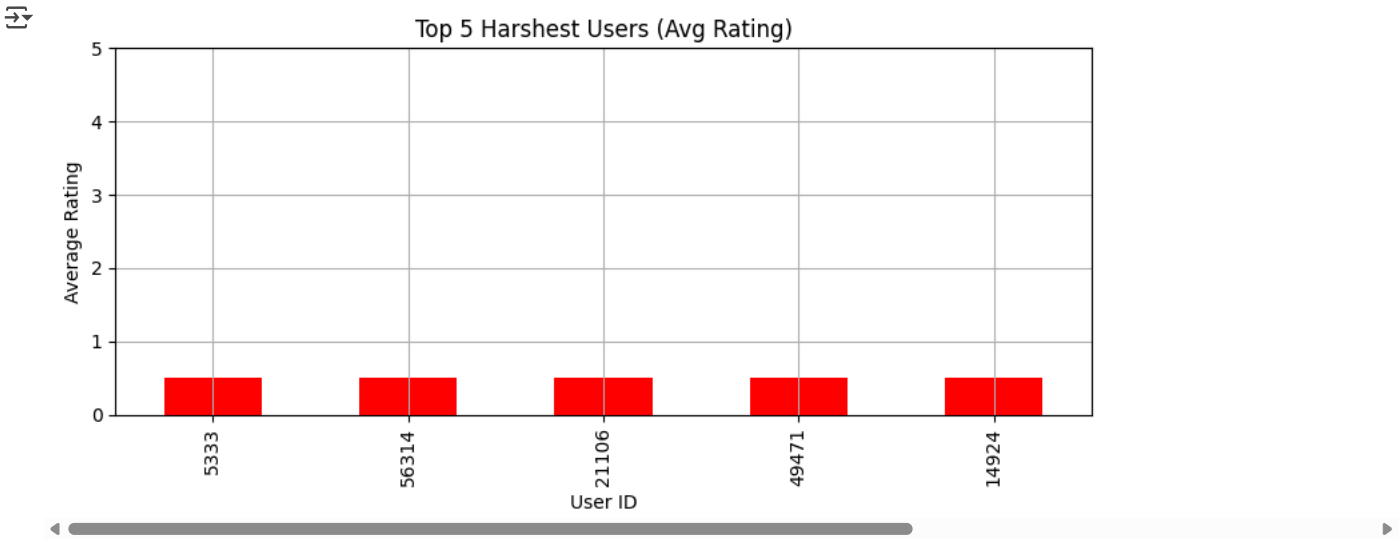
```
harshest_users = avg_rating_per_user.sort_values(ascending=True).head(5)
print("\nKindest Users (Highest Avg Rating):\n", kindest_users)
print("\nHarshest Users (Lowest Avg Rating):\n", harshest_users)
```

```
Kindest Users (Highest Avg Rating):
 userId
18287    5.0
43894    5.0
37734    5.0
6932     5.0
18414    5.0
Name: rating, dtype: float64

Harshest Users (Lowest Avg Rating):
 userId
5333     0.5
56314    0.5
21106    0.5
49471    0.5
14924    0.5
Name: rating, dtype: float64
```

```
# Bar plot for kindest & harshest
plt.figure(figsize=(8, 4))
kindest_users.plot(kind='bar', color='green')
plt.title("Top 5 Kindest Users (Avg Rating)")
plt.xlabel("User ID")
plt.ylabel("Average Rating")
plt.ylim(0, 5)
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(8, 4))
harshest_users.plot(kind='bar', color='red')
plt.title("Top 5 Harshest Users (Avg Rating)")
plt.xlabel("User ID")
plt.ylabel("Average Rating")
plt.ylim(0, 5)
plt.grid(True)
plt.tight_layout()
plt.show()
```



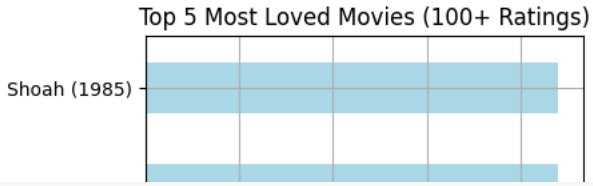

```
# Load cleaned data
merged_df = pd.read_csv("merged_movies_ratings.csv")
tags_df = pd.read_csv("/content/drive/MyDrive/movies dataset/tags.csv")
```

```
# 4. Tagging behavior (Optional)
# Most active taggers
taggers = tags_df['userId'].value_counts().head(10)
print("\nTop 10 Taggers:\n", taggers)
```

```
Top 10 Taggers:
 userId
78213     723473
119227     20369
68821      20317
147560     18849
159300     16843
34874      13838
17035      12829
34458      12735
102040     12681
123480     11551
Name: count, dtype: int64
```

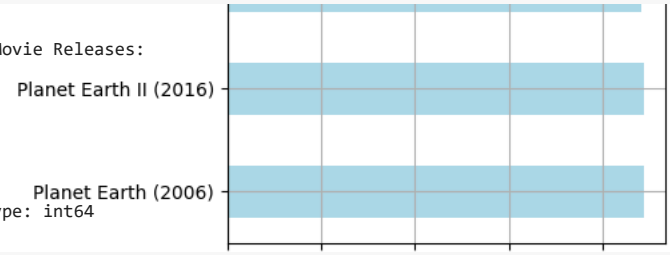
```
# Top 5 most rated and highest average rated movies (min 100 votes)
popular_movies = merged_df.groupby('title').agg({'rating': ['mean', 'count']})
popular_movies.columns = ['avg_rating', 'num_ratings']
popular_movies = popular_movies[popular_movies['num_ratings'] >= 100]
top_combined = popular_movies.sort_values(by=['avg_rating', 'num_ratings'], ascending=[False, False]).head(5)

print("\nTop 5 Most Loved Movies:\n", top_combined)

# Bar chart
top_combined['avg_rating'].plot(kind='barh', color='lightblue')
plt.title("Top 5 Most Loved Movies (100+ Ratings)")
plt.xlabel("Average Rating")
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Top 5 Most Loved Movies:
                                   avg_rating  num_ratings
title
Planet Earth (2006)                 4.445898          841
Planet Earth II (2016)              4.442755          559
Shawshank Redemption, The (1994)    4.407133        30506
Band of Brothers (2001)             4.404732          803
Shoah (1985)                        4.403846          104
```




```
#Year with Most Movies Released
movies['year'] = movies['title'].str.extract(r'\((\d{4})\)')
year_counts = movies['year'].value_counts().sort_values(ascending=False)
print("\nYear with Most Movie Releases:\n", year_counts.head(5))
```

```
Year with Most Movie Releases:
 year
2017    3269
2018    3180
2016    3167
2019    3084
2015    3080
Name: count, dtype: int64
```




```
#most common genre
genre_counts = movies['genres'].str.split('|').explode().value_counts()
print("\nMost Common Genre:\n", genre_counts.head(1))
```

```
Most Common Genre:
 genres
Drama    34175
Name: count, dtype: int64
```

```
#avg rating for each movie by different users
avg_user_ratings = merged_df.groupby(['title', 'userId'])['rating'].mean()
print("\nAverage Ratings per Movie by Each User:\n", avg_user_ratings.head())
```

```
Average Ratings per Movie by Each User:
 title   userId
(2019)  9101      4.0
        15489     2.5
        17035     3.0
        26769     3.0
        31899     2.5
Name: rating, dtype: float64
```

```
#movie with most unique user ratings
user_counts_per_movie = merged_df.groupby('title')['userId'].nunique().sort_values(ascending=False)
print("\nMovie with Most Unique User Ratings:\n", user_counts_per_movie.head(1))
```

```
Movie with Most Unique User Ratings:
 title
Shawshank Redemption, The (1994)    30506
Name: userId, dtype: int64
```

```
#user who rated the most movies and their avg rating
user_rating_counts = merged_df.groupby('userId')['rating'].agg(['count', 'mean'])
most_active_user = user_rating_counts.sort_values(by='count', ascending=False).head(1)
print("\nMost Active User (Number of Ratings and Avg Rating):\n", most_active_user)
```

```
Most Active User (Number of Ratings and Avg Rating):
         count      mean
userId
17035    9577  2.567819
```

Double-click (or enter) to edit