In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
```

Q1.How many ".csv" files are available in the dataset?

In [2]:
```
4
```

Out[2]: 4

Q2.What is the shape of "movies.csv"?

In [3]:
```
df_movies = pd.read_csv('movies.csv')
```

In [4]:
```
df_movies.shape
```

Out[4]: (9742, 3)

Q3.What is the shape of "ratings.csv"?

In [5]:
```
df_ratings = pd.read_csv("ratings.csv")
```

In [6]:
```
df_ratings.shape
```

Out[6]: (100836, 4)

Q4.How many unique "userId" are available in "ratings.csv"?

In [7]:
```
df_ratings['userId'].nunique()
```

Out[7]: 610

Q5.Which movie has recieved maximum number of user ratings?

In [8]:
```
merged_df = pd.merge(df_ratings, df_movies, on='movieId')

# Group by 'title' and count the number of ratings for each movie
ratings_count = merged_df.groupby('title')['rating'].count()

# Find the movie with the maximum number of ratings
max_rated_movie = ratings_count.idxmax()

print(f"The movie which has received the maximum number of user rati
```

```
The movie which has received the maximum number of user ratings i
s: Forrest Gump (1994)
```

Q6.Select all the correct tags submitted by users to "Matrix, The (1999)" movie?

In [9]:
```
df_tags = pd.read_csv("tags.csv")

# Find the movieId for "Matrix, The (1999)"
matrix_movie_id = df_movies[df_movies['title'] == 'Matrix, The (1999

# Filter tags for "Matrix, The (1999)"
matrix_tags = df_tags[df_tags['movieId'] == matrix_movie_id]

# Display the tags for the specified movie
correct_tags = matrix_tags['tag'].unique()
print(f"All the correct tags submitted by users to 'Matrix, The (199
```

```
All the correct tags submitted by users to 'Matrix, The (1999)' ar
e: martial arts, sci-fi, alternate universe, philosophy, post apoc
alyptic
```

Q7.What is the average user rating for movie named "Terminator 2: Judgment Day (1991)"?

In [10]:
```
# Find the movieId for "Terminator 2: Judgment Day (1991)"
terminator2_movie_id = df_movies[df_movies['title'] == 'Terminator 2

# Filter ratings for "Terminator 2: Judgment Day (1991)"
terminator2_ratings = df_ratings[df_ratings['movieId'] == terminator

# Calculate the average user rating
average_rating = terminator2_ratings['rating'].mean()

print(f"The average user rating for movie named 'Terminator 2: Judgm
```

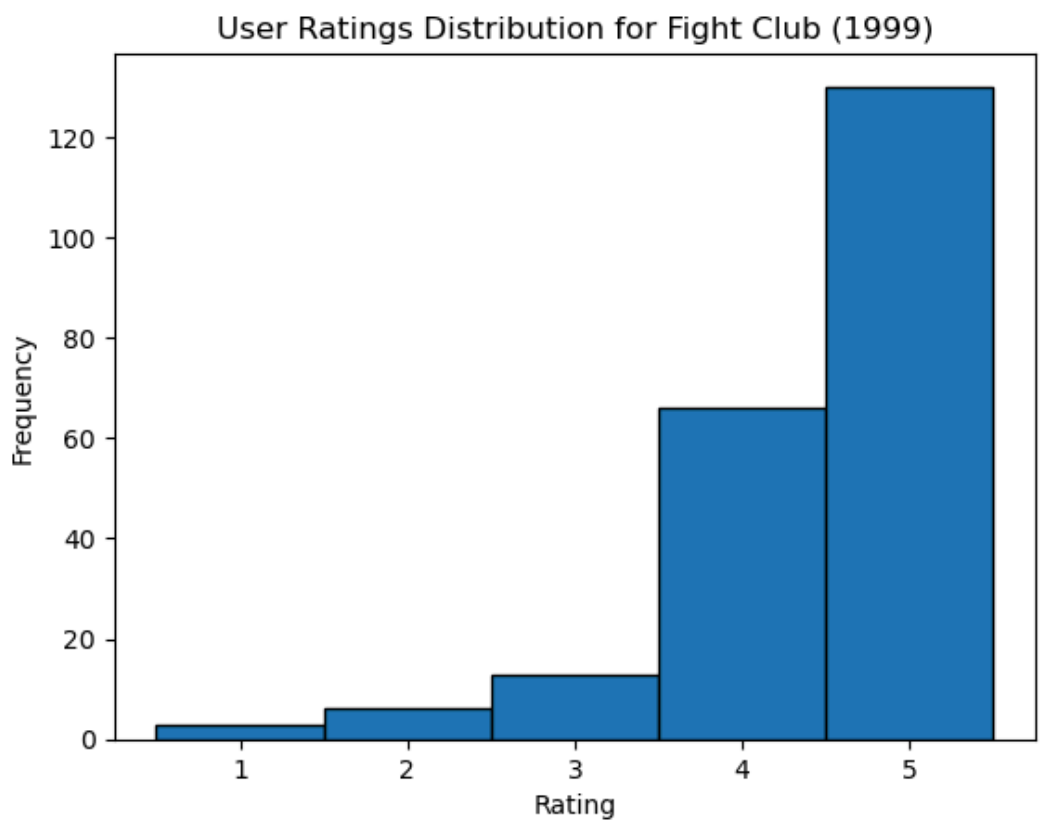
```
The average user rating for movie named 'Terminator 2: Judgment Da
y (1991)' is: 3.97
```

Q8.How does the data distribution of user ratings for "Fight Club (1999)" movie looks like?

In [11]:
```
# Find the movieId for "Fight Club (1999)"
fight_club_movie_id = df_movies[df_movies['title'] == 'Fight Club (1

# Filter ratings for "Fight Club (1999)"
fight_club_ratings = df_ratings[df_ratings['movieId'] == fight_club_

# Plot a histogram to visualize the data distribution
plt.hist(fight_club_ratings['rating'], bins=[0.5, 1.5, 2.5, 3.5, 4.5
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('User Ratings Distribution for Fight Club (1999)')
plt.show()
print(f"The data distribution of user ratings for 'Fight Club(1999)'
```




```
The data distribution of user ratings for 'Fight Club(1999)'movie
looks like - Left Skewed Distribution
```

Q9.Which movie is the most popular based on average user ratings?

In [12]:
```
# Group user ratings based on movieId and apply aggregation operatic
grouped_ratings = df_ratings.groupby('movieId').agg({'rating': ['cou

# Rename the columns for clarity
grouped_ratings.columns = ['rating_count', 'rating_mean']

# Inner join with movies dataframe
merged_df = pd.merge(df_movies, grouped_ratings, on='movieId', how='

# Filter only those movies with more than 50 user ratings
filtered_df = merged_df[merged_df['rating_count'] > 50]

# Find the movie with the highest average user rating
most_popular_movie = filtered_df.loc[filtered_df['rating_mean'].idxm

print(f"The most popular movie based on average user ratings is: {mo
```

```
The most popular movie based on average user ratings is: Shawshank
Redemption, The (1994) with an average rating of 4.43
```

Q10.Select all the correct options which comes under top 5 popular movies based on number of user ratings

In [13]:
```
# Find the top 5 popular movies based on number of user ratings
top_movies = filtered_df.sort_values(by='rating_count', ascending=Fa

print("Top 5 popular movies based on number of user ratings:")
print("")
print(top_movies[['movieId', 'title', 'rating_count']])
```

```
Top 5 popular movies based on number of user ratings:

      movieId                             title  rating_count
314       356                Forrest Gump (1994)           329
277       318  Shawshank Redemption, The (1994)           317
257       296                Pulp Fiction (1994)           307
510       593  Silence of the Lambs, The (1991)           279
1938     2571                 Matrix, The (1999)           278
```

Q11.Which Sci-Fi movie is "third most popular" based on the number of user ratings?

In [14]:
```
# Filter Sci-Fi movies from the filtered dataframe
sci_fi_movies = filtered_df[filtered_df['genres'].str.contains('Sci-

# Find the third most popular Sci-Fi movie based on number of user r
third_most_popular_sci_fi = sci_fi_movies.nlargest(3, 'rating_count'

print(f"The third most popular Sci-Fi movie based on the number of u
```

```
The third most popular Sci-Fi movie based on the number of user ra
tings is: Jurassic Park (1993) with 238 user ratings.
```

Question : Web Scraping Part Solutions

In [15]:
```
filtered_df.head()
```

Out[15]:

	movieId	title	genres	rating_count	ratin
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	215	3
1	2	Jumanji (1995)	Adventure|Children|Fantasy	110	3
2	3	Grumpier Old Men (1995)	Comedy|Romance	52	3
5	6	Heat (1995)	Action|Crime|Thriller	102	3
6	7	Sabrina (1995)	Comedy|Romance	54	3

In [16]:
```
df_links = pd.read_csv("links.csv")
```

In [17]:
```
import requests
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import json



def scrapper(imdbId):
    id = str(int(imdbId))
    n_zeroes = 7 - len(id)
    new_id = "0" * n_zeroes + id
    URL = f"https://www.imdb.com/title/tt{new_id}/"
    request_header = {'Content-Type': 'text/html; charset=UTF-8',
                      'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; W
                      'Accept-Encoding': 'gzip, deflate, br'}
    response = requests.get(URL, headers=request_header)
    soup = BeautifulSoup(response.text, 'html.parser')


    # Find the JSON data within the script tag
    json_data = soup.find('script', type='application/ld+json').text

    # Load the JSON data
    data = json.loads(json_data)

    # Get the IMDb rating from the JSON data
    imdb_rating = data['aggregateRating']['ratingValue']


    return imdb_rating if imdb_rating else np.nan
```

In [18]:
```
filtered_df_merged = pd.merge(filtered_df, df_links[['movieId', 'imd
```

In [19]:
```
filtered_df_merged.head()
```

Out[19]:

	movieId	title	genres	rating_count	ratin
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	215	3
1	2	Jumanji (1995)	Adventure|Children|Fantasy	110	3
2	3	Grumpier Old Men (1995)	Comedy|Romance	52	3
3	6	Heat (1995)	Action|Crime|Thriller	102	3
4	7	Sabrina (1995)	Comedy|Romance	54	3

In [20]:
```
filtered_df_merged['imdb_rating'] = filtered_df_merged['imdbId'].app
```

In [21]:
```
filtered_df_merged.head()
```

Out[21]:

	movieId	title	genres	rating_count	ratin
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	215	3
1	2	Jumanji (1995)	Adventure|Children|Fantasy	110	3
2	3	Grumpier Old Men (1995)	Comedy|Romance	52	3
3	6	Heat (1995)	Action|Crime|Thriller	102	3
4	7	Sabrina (1995)	Comedy|Romance	54	3

Q12.Mention the movieId of the movie which has the highest IMDB rating

In [22]:
```
# Find the movieId with the highest IMDB rating
highest_rated_movie = filtered_df_merged.loc[filtered_df_merged['imd

print(f"The movieId of the movie with the highest IMDB rating is: {h
```

```
The movieId of the movie with the highest IMDB rating is: 318 with
an IMDB rating of 9.3
```

Q13.Mention the movieId of the "Sci-Fi" movie which has the highest IMDB rating

In [23]:
```
# Filter the subset for Sci-Fi genre
scifi_movies = filtered_df_merged[filtered_df_merged['genres'].str.c

# Find the movie with the highest IMDb rating
highest_rated_scifi_movie = scifi_movies.loc[scifi_movies['imdb_rati

print("MovieId of the Sci-Fi movie with the highest IMDb rating:", h
```

```
MovieId of the Sci-Fi movie with the highest IMDb rating: 79132
```