In [13]:
```
import pandas as pd
```

In [17]:
```
movies = pd.read_csv("C:\\Users\\91630\\Desktop\\movies1.csv")
```

In [18]:
```
movies
```

Out[18]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

9742 rows × 3 columns

In [20]:
```
links = pd.read_csv("C:\\Users\\91630\\Desktop\\links1.csv")
links
```

Out[20]:

	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
...	...	...	...
9737	193581	5476944	432131.0
9738	193583	5914996	445030.0
9739	193585	6397426	479308.0
9740	193587	8391976	483455.0
9741	193609	101726	37891.0

9742 rows × 3 columns

In [21]:
```
tags = pd.read_csv("C:\\Users\\91630\\Desktop\\tags1.csv")
tags
```

Out[21]:

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200
...	...	...	...	...
3678	606	7382	for katie	1171234019
3679	606	7936	austere	1173392334
3680	610	3265	gun fu	1493843984
3681	610	3265	heroic bloodshed	1493843978
3682	610	168248	Heroic Bloodshed	1493844270

3683 rows × 4 columns

In [23]:
```
ratings = pd.read_csv("C:\\Users\\91630\\Desktop\\ratings1.csv")
ratings
```

Out[23]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091
100833	610	168250	5.0	1494273047
100834	610	168252	5.0	1493846352
100835	610	170875	3.0	1493846415

100836 rows × 4 columns

In [25]:
```
unique_user_ids = ratings['userId'].nunique()
unique_user_ids
```

Out[25]:
```
610
```

In [42]:
```
max_rated_movie = ratings['movieId'].value_counts().idxmax()
max_rated_movie_name = movies[movies['movieId'] == max_rated_movie]['title'].iloc[0]
max_rated_movie_name
```

Out[42]:
```
'Forrest Gump (1994)'
```

In [35]:
```
matrix_tags = tags[tags['movieId'] == 2571]
matrix_tags
```

Out[35]:

	userId	movieId	tag	timestamp
815	424	2571	martial arts	1457842912
816	424	2571	sci-fi	1457842899
1646	474	2571	alternate universe	1137204991
2794	537	2571	philosophy	1424141098
2795	537	2571	post apocalyptic	1424141101

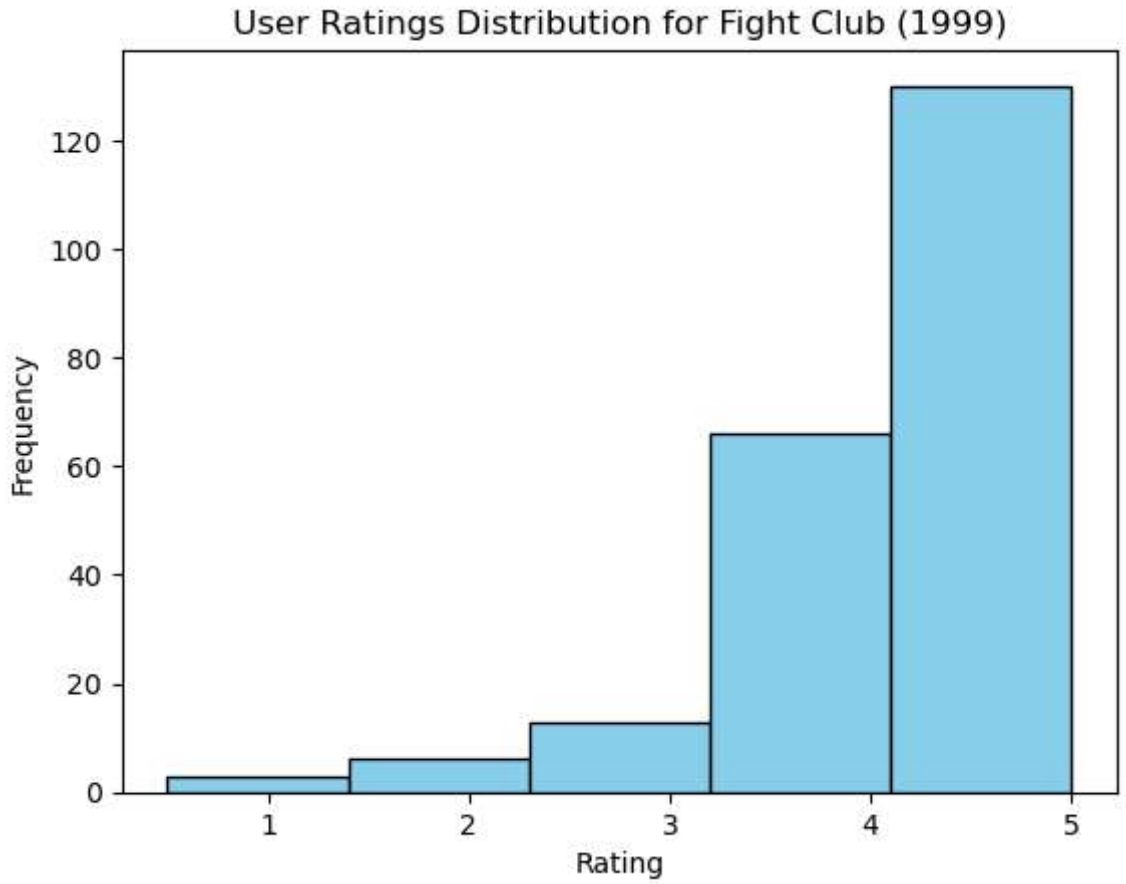
In [36]:
```
avg_rating_terminator = ratings[ratings['movieId'] == 589]['rating'].mean()
avg_rating_terminator
```

Out[36]:
```
3.970982142857143
```

In [37]:
```
import matplotlib.pyplot as plt

# Get the user ratings for the movie 'Fight Club (1999)'
fight_club_ratings = ratings[ratings['movieId'] == 2959]['rating']

# Plot the distribution of user ratings
plt.hist(fight_club_ratings, bins=5, color='skyblue', edgecolor='black')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('User Ratings Distribution for Fight Club (1999)')
plt.show()
```



In [47]:
```
# Group the user ratings based on movieId and apply aggregation operations like count and mean on ratings
grouped_ratings = ratings.groupby('movieId')['rating'].agg(['count', 'mean'])
grouped_ratings.head()
```

Out[47]:

movieId	count	mean
1	215	3.920930
2	110	3.431818
3	52	3.259615
4	7	2.357143
5	49	3.071429

In [48]:
```
# Apply inner join on dataframe created from movies.csv and the grouped df from step 1
merged_df = pd.merge(movies, grouped_ratings, on='movieId', how='inner')
merged_df.head()
```

Out[48]:

	movieId	title	genres	count	mean
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	215	3.920930
1	2	Jumanji (1995)	Adventure|Children|Fantasy	110	3.431818
2	3	Grumpier Old Men (1995)	Comedy|Romance	52	3.259615
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	7	2.357143
4	5	Father of the Bride Part II (1995)	Comedy	49	3.071429

In [49]:
```
# Filter only those movies which have more than 50 user ratings
popular_movies = merged_df[merged_df['count'] > 50]
most_popular_movie = popular_movies[popular_movies['mean'] == popular_movies['mean'].max()]
most_popular_movie
```

Out[49]:

	movieId	title	genres	count	mean
277	318	Shawshank Redemption, The (1994)	Crime|Drama	317	4.429022

In [50]:
```
# Find the top 5 popular movies based on number of user ratings
top_5_popular_movies = popular_movies.nlargest(5, 'count')
top_5_popular_movies[['title', 'count', 'mean']]
```

Out[50]:

	title	count	mean
314	Forrest Gump (1994)	329	4.164134
277	Shawshank Redemption, The (1994)	317	4.429022
257	Pulp Fiction (1994)	307	4.197068
510	Silence of the Lambs, The (1991)	279	4.161290
1938	Matrix, The (1999)	278	4.192446

In [51]:
```
# Find the third most popular Sci-Fi movie based on the number of user ratings
sci_fi_movies = popular_movies[popular_movies['genres'].str.contains('Sci-Fi', case=False)]
third_most_popular_sci_fi_movie = sci_fi_movies.nlargest(3, 'count').iloc[-1]
third_most_popular_sci_fi_movie['title']
```

Out[51]:
```
'Jurassic Park (1993)'
```

In [53]:
```
# Filter the movies with more than 50 user ratings
rated_movies = ratings['movieId'].value_counts()
highly_rated_movies = rated_movies[rated_movies > 50].index

# Merge the highly rated movies with their IMDB IDs
highly_rated_links = links[links['movieId'].isin(highly_rated_movies)]
highly_rated_movies_with_imdb = movies.merge(highly_rated_links, on='movieId', how='inner')
highly_rated_movies_with_imdb
```

Out[53]:

	movieId	title	genres	imdbId	tmdbId
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	114709	862.0
1	2	Jumanji (1995)	Adventure|Children|Fantasy	113497	8844.0
2	3	Grumpier Old Men (1995)	Comedy|Romance	113228	15602.0
3	6	Heat (1995)	Action|Crime|Thriller	113277	949.0
4	7	Sabrina (1995)	Comedy|Romance	114319	11860.0
...	...	...	...	...	...
431	106782	Wolf of Wall Street, The (2013)	Comedy|Crime|Drama	993846	106646.0
432	109374	Grand Budapest Hotel, The (2014)	Comedy|Drama	2278388	120467.0
433	109487	Interstellar (2014)	Sci-Fi|IMAX	816692	157336.0
434	112852	Guardians of the Galaxy (2014)	Action|Adventure|Sci-Fi	2015381	118340.0
435	122904	Deadpool (2016)	Action|Adventure|Comedy|Sci-Fi	1431045	293660.0

436 rows × 5 columns

In [56]:
```
# Save the scraped IMDB reviews to a file
highly_rated_movies_with_imdb.to_csv('highly_rated_movies_with_imdb.csv', index=False)
```

In [57]:
```
# Find the movie with the highest IMDB rating
highest_imdb_rating_movie = highly_rated_movies_with_imdb.loc[highly_rated_movies_with_imdb['imdbId'].idxmax()]
highest_imdb_rating_movie['movieId']
```

Out[57]:
```
109374
```

In [60]:
```
# Display the first few rows of the dataframe to inspect the data
highly_rated_movies_with_imdb.head()
```

Out[60]:

	movieId	title	genres	imdbId	tmdbId
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	114709	862.0
1	2	Jumanji (1995)	Adventure|Children|Fantasy	113497	8844.0
2	3	Grumpier Old Men (1995)	Comedy|Romance	113228	15602.0
3	6	Heat (1995)	Action|Crime|Thriller	113277	949.0
4	7	Sabrina (1995)	Comedy|Romance	114319	11860.0

In [61]:
```
# Find the movie with the highest IMDB rating in the 'Sci-Fi' genre
highest_imdb_rating_scifi_movie = highly_rated_movies_with_imdb[highly_rated_movies_with_imdb['genres'].str.contains('S
highest_imdb_rating_scifi_movie['movieId']
```

Out[61]:
```
434    112852
Name: movieId, dtype: int64
```

In [ ]: