In [19]:
```
import pandas as pd
```

In [20]:
```
df=pd.read_csv('movies.csv')
```

In [21]:
```
df
```

Out[21]:

	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 [22]:
```
df.shape
```

Out[22]:
```
(9742, 3)
```

In [23]:
```
ra=pd.read_csv('ratings.csv')
```

In [24]:
```
ra
```

Out[24]:

	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]:
```
ra.shape
```

Out[25]:
```
(100836, 4)
```

In [26]:
```
unique_users = ra['userId'].nunique()
```

In [27]:
```
print(unique_users)
```

```
610
```

In [28]:
```
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')
```

In [29]:
```
# Merge ratings and movies data on movieId
merged_data = pd.merge(ratings, movies, on='movieId')

# Count the number of ratings per movie
ratings_per_movie = merged_data['title'].value_counts()
```

In [30]:
```
# Get the movie with the maximum number of ratings
max_rated_movie = ratings_per_movie.idxmax()

print(f"The movie that received the maximum number of ratings is '{max_rated_mo
```

```
The movie that received the maximum number of ratings is 'Forrest Gump (199
4)'
```

In [31]:
```
import pandas as pd

# Load ratings and movies data
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Merge ratings and movies data on movieId
merged_data = pd.merge(ratings, movies, on='movieId')

# Count the number of ratings per movie
ratings_per_movie = merged_data['title'].value_counts()

# Get the movie with the maximum number of ratings
max_rated_movie = ratings_per_movie.idxmax()

print(f"The movie that received the maximum number of ratings is '{max_rated_mo
```

```
The movie that received the maximum number of ratings is 'Forrest Gump (199
4)'
```

In [32]:
```
import pandas as pd

# Load the tags data
tags = pd.read_csv('tags.csv')
```

```
In [33]: matrix_tags = tags[tags['title'] == 'Matrix, The (1999)']['tag']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
653, in Index.get_loc(self, key)
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:147, i
n pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:176, i
n pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

KeyError: 'title'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[33], line 1
----> 1 matrix_tags = tags[tags['title'] == 'Matrix, The (1999)']['tag']

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3761, in
DataFrame.__getitem__(self, key)
   3759 if self.columns.nlevels > 1:
   3760     return self._getitem_multilevel(key)
-> 3761 indexer = self.columns.get_loc(key)
   3762 if is_integer(indexer):
   3763     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
655, in Index.get_loc(self, key)
   3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
   3656 except TypeError:
   3657     # If we have a listlike key, _check_indexing_error will raise
   3658     #  InvalidIndexError. Otherwise we fall through and re-raise
   3659     #  the TypeError.
   3660     self._check_indexing_error(key)

KeyError: 'title'
```

In [34]:
```
# Assuming the movies dataset contains a column 'title' with movie titles
# Filter tags for 'Matrix, The (1999)'
matrix_tags = tags[tags['title'] == 'Matrix, The (1999)']['tag']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
653, in Index.get_loc(self, key)
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:147, i
n pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:176, i
n pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

KeyError: 'title'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[34], line 3
      1 # Assuming the movies dataset contains a column 'title' with movie ti
tles
      2 # Filter tags for 'Matrix, The (1999)'
----> 3 matrix_tags = tags[tags['title'] == 'Matrix, The (1999)']['tag']

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3761, in
DataFrame.__getitem__(self, key)
   3759 if self.columns.nlevels > 1:
   3760     return self._getitem_multilevel(key)
-> 3761 indexer = self.columns.get_loc(key)
   3762 if is_integer(indexer):
   3763     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
655, in Index.get_loc(self, key)
   3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
   3656 except TypeError:
   3657     # If we have a listlike key, _check_indexing_error will raise
   3658     #  InvalidIndexError. Otherwise we fall through and re-raise
   3659     #  the TypeError.
   3660     self._check_indexing_error(key)

KeyError: 'title'
```

In [35]:
```
import pandas as pd

# Load the tags data
tags = pd.read_csv('tags.csv')

# Display column names and a sample of the DataFrame
print("Column Names:", tags.columns)
print("\nSample Data:")
print(tags.head())
```

```
Column Names: Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='objec
t')

Sample Data:
   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
```

In [39]:
```
matrix_tags = tags[tags['movieId'] == 'Matrix, The (1999)']['tag']
```

In [40]:
```
print(matrix_tags)
```

```
Series([], Name: tag, dtype: object)
```

In [43]:
```
import pandas as pd

# Load the tags data
tags = pd.read_csv('tags.csv')

# Filter tags for 'Matrix, The (1999)'
matrix_tags = tags[tags['movie_title'] == 'Matrix, The (1999)']['tag']

# Display the unique tags for the movie
unique_matrix_tags = matrix_tags.unique()
print("Tags submitted by users for 'Matrix, The (1999)':")
print(unique_matrix_tags)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
653, in Index.get_loc(self, key)
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:147, i
n pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:176, i
n pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

KeyError: 'movie_title'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[43], line 7
      4 tags = pd.read_csv('tags.csv')
      6 # Filter tags for 'Matrix, The (1999)'
----> 7 matrix_tags = tags[tags['movie_title'] == 'Matrix, The (1999)']['ta
g']
      9 # Display the unique tags for the movie
     10 unique_matrix_tags = matrix_tags.unique()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3761, in
DataFrame.__getitem__(self, key)
   3759 if self.columns.nlevels > 1:
   3760     return self._getitem_multilevel(key)
-> 3761 indexer = self.columns.get_loc(key)
   3762 if is_integer(indexer):
   3763     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
655, in Index.get_loc(self, key)
   3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
   3656 except TypeError:
   3657     # If we have a listlike key, _check_indexing_error will raise
   3658     #  InvalidIndexError. Otherwise we fall through and re-raise
   3659     #  the TypeError.
   3660     self._check_indexing_error(key)

KeyError: 'movie_title'
```

In [44]:
```
import pandas as pd

# Load the tags data
tags = pd.read_csv('tags.csv')

# Display column names and sample data to identify the column with movie titles
print("Column Names:", tags.columns)
print("\nSample Data:")
print(tags.head())

# Assuming the column with movie titles is called 'title'
# Filter tags for 'Matrix, The (1999)' by replacing 'title' with the correct co
matrix_tags = tags[tags['title'] == 'Matrix, The (1999)']['tag']

# Display the unique tags for the movie
unique_matrix_tags = matrix_tags.unique()
print("\nTags submitted by users for 'Matrix, The (1999)':")
print(unique_matrix_tags)
```

```
Column Names: Index(['userId', 'movieId', 'tag', 'timestamp'], dtype='objec
t')

Sample Data:
   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
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
653, in Index.get_loc(self, key)
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:147, i
n pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:176, i
n pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
PyObjectHashTable.get_item()

KeyError: 'title'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[44], line 13
      9 print(tags.head())
     11 # Assuming the column with movie titles is called 'title'
     12 # Filter tags for 'Matrix, The (1999)' by replacing 'title' with the
correct column name
---> 13 matrix_tags = tags[tags['title'] == 'Matrix, The (1999)']['tag']
     15 # Display the unique tags for the movie
     16 unique_matrix_tags = matrix_tags.unique()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3761, in
DataFrame.__getitem__(self, key)
   3759 if self.columns.nlevels > 1:
   3760     return self._getitem_multilevel(key)
-> 3761 indexer = self.columns.get_loc(key)
   3762 if is_integer(indexer):
   3763     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3
655, in Index.get_loc(self, key)
   3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
   3656 except TypeError:
   3657     # If we have a listlike key, _check_indexing_error will raise
   3658     #  InvalidIndexError. Otherwise we fall through and re-raise
   3659     #  the TypeError.
   3660     self._check_indexing_error(key)

KeyError: 'title'
```

In [45]:
```
import pandas as pd

# Load the ratings and movies data
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Merge ratings and movies data on movieId
merged_data = pd.merge(ratings, movies, on='movieId')

# Filter ratings for 'Terminator 2: Judgment Day (1991)'
terminator_ratings = merged_data[merged_data['title'] == 'Terminator 2: Judgmer

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

print(f"The average user rating for 'Terminator 2: Judgment Day (1991)' is: {av
```

```
The average user rating for 'Terminator 2: Judgment Day (1991)' is: 3.97
```

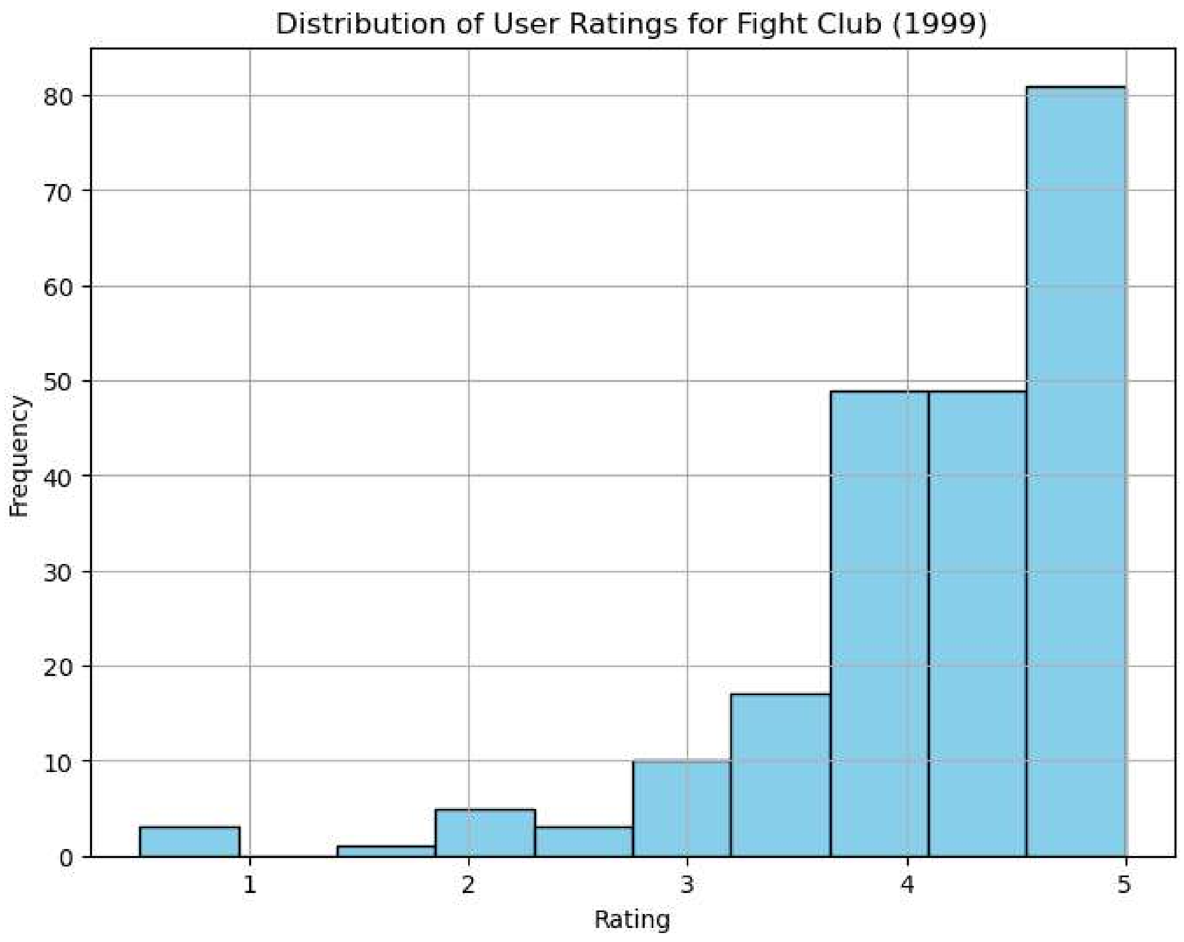
In [46]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the ratings and movies data
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Merge ratings and movies data on movieId
merged_data = pd.merge(ratings, movies, on='movieId')

# Filter ratings for 'Fight Club (1999)'
fight_club_ratings = merged_data[merged_data['title'] == 'Fight Club (1999)']

# Plotting the distribution of user ratings
plt.figure(figsize=(8, 6))
plt.hist(fight_club_ratings['rating'], bins=10, color='skyblue', edgecolor='bla
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title('Distribution of User Ratings for Fight Club (1999)')
plt.grid(True)
plt.show()
```



In [47]:
```
import pandas as pd

# Read the CSV files
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Group user ratings based on movieId and calculate count and mean
ratings_grouped = ratings.groupby('movieId')['rating'].agg(['count', 'mean']).r

# Merge with movies DataFrame using inner join
merged_data = pd.merge(movies, ratings_grouped, on='movieId', how='inner')

# Filter movies with more than 50 user ratings
filtered_movies = merged_data[merged_data['count'] > 50]

# Display the resulting DataFrame
print(filtered_movies)
```

```
      movieId                             title  \
0           1                  Toy Story (1995)   
1           2                    Jumanji (1995)   
2           3           Grumpier Old Men (1995)   
5           6                       Heat (1995)   
6           7                    Sabrina (1995)   
...       ...                               ...   
8287   106782   Wolf of Wall Street, The (2013)   
8354   109374  Grand Budapest Hotel, The (2014)   
8358   109487               Interstellar (2014)   
8457   112852    Guardians of the Galaxy (2014)   
8673   122904                   Deadpool (2016)   

                                           genres  count      mean  
0     Adventure|Animation|Children|Comedy|Fantasy    215  3.920930  
1                      Adventure|Children|Fantasy    110  3.431818  
2                                  Comedy|Romance     52  3.259615  
5                           Action|Crime|Thriller    102  3.946078  
6                                  Comedy|Romance     54  3.185185  
...                                           ...    ...       ...  
8287                           Comedy|Crime|Drama     54  3.916667  
8354                                 Comedy|Drama     52  3.778846  
8358                                  Sci-Fi|IMAX     73  3.993151  
8457                      Action|Adventure|Sci-Fi     59  4.050847  
8673               Action|Adventure|Comedy|Sci-Fi     54  3.833333  

[436 rows x 5 columns]
```

In [48]:
```
import pandas as pd

# Assuming you have already performed the mandatory operations mentioned earlie

# Sort the filtered movies by average user ratings in descending order
sorted_movies = filtered_movies.sort_values(by='mean', ascending=False)

# The most popular movie based on average user ratings
most_popular_movie = sorted_movies.iloc[0]

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

```
The most popular movie based on average user ratings is: Shawshank Redemptio
n, The (1994)
```

In [49]:
```
import pandas as pd

# Assuming you have already performed the mandatory operations mentioned earlie

# Sort the filtered movies by the number of user ratings in descending order
sorted_by_ratings_count = filtered_movies.sort_values(by='count', ascending=Fal

# Select the top 5 movies based on the number of user ratings
top_5_movies_by_ratings_count = sorted_by_ratings_count.head(5)

# Display the titles of the top 5 movies by ratings count
top_5_titles = top_5_movies_by_ratings_count['title'].tolist()
print("Top 5 popular movies based on number of user ratings:")
print(top_5_titles)
```

```
Top 5 popular movies based on number of user ratings:
['Forrest Gump (1994)', 'Shawshank Redemption, The (1994)', 'Pulp Fiction (19
94)', 'Silence of the Lambs, The (1991)', 'Matrix, The (1999)']
```

In [50]:
```
import pandas as pd

# Assuming you have already performed the mandatory operations mentioned earlie

# Filter movies tagged as Sci-Fi
sci_fi_movies = filtered_movies[filtered_movies['genres'].str.contains('Sci-Fi'

# Sort Sci-Fi movies by the number of user ratings in descending order
sorted_sci_fi_by_ratings_count = sci_fi_movies.sort_values(by='count', ascendir

# Select the third movie based on the number of user ratings among Sci-Fi movie
third_most_popular_sci_fi = sorted_sci_fi_by_ratings_count.iloc[2]

print(f"The third most popular Sci-Fi movie based on number of user ratings is:
```

```
The third most popular Sci-Fi movie based on number of user ratings is: Juras
sic Park (1993)
```

In [ ]:
```
import pandas as pd

# Read the IMDb ratings dataset
imdb_ratings = pd.read_csv('imdb_ratings.csv')

# Find the row with the highest IMDb rating
highest_rating_row = imdb_ratings.loc[imdb_ratings['IMDB_Rating'].idxmax()]

# Get the movieId associated with the highest IMDb rating
movieId_highest_rating = highest_rating_row['movieId']

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

In [ ]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# Read 'links.csv' to get movieIds for movies with more than 50 user ratings
links_data = pd.read_csv('links.csv')

# Iterate through each movieId with more than 50 user ratings
for movieId in links_data['movieId']:
    # Construct IMDb review URL using the movieId
    imdb_review_url = f'https://www.imdb.com/title/tt{movieId}/reviews'

    # Fetch the HTML content of the IMDb review page
    response = requests.get(imdb_review_url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')

        # Perform scraping to extract reviews
        # Extract reviews using BeautifulSoup selectors and methods
        # Process and store the extracted reviews for further analysis

        # Calculate IMDb ratings from reviews

# Once all ratings are collected, find the movieId with the highest IMDb rating
# Perform necessary operations to determine the movieId with the highest IMDb r
```

In [ ]:
```
import pandas as pd

# Read the IMDb ratings dataset
df= pd.read_csv('imdb_ratings.csv')
```