Loading Libraries

In [1]:
```
import pandas as pd
import difflib
from sklearn.metrics.pairwise import cosine_similarity
import ipywidgets as widgets
import matplotlib.pyplot as plt
import seaborn as sns
```

Exploring Dataset

In [2]:
```
#Loading movies dataset
movies = pd.read_csv('movies_info.csv')
```

In [3]:
```
movies.head()
```

Out[3]:

	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

In [4]:
```
#Loading rating dataset
ratings = pd.read_csv('ratings.csv')
```

In [5]:
```
ratings.head()
```

Out[5]:

	userId	movieId	rating	timestamp
0	1	16	4.0	1217897793
1	1	24	1.5	1217895807
2	1	32	4.0	1217896246
3	1	47	4.0	1217896556
4	1	50	4.0	1217896523

In [6]:
```
#merging two datasets into one dataset with inner join in movieId column
movies_dataset = pd.merge(movies,ratings,on='movieId',how='inner')
```

In [7]:
```
movies_dataset.head()
```

Out[7]:

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	2	5.0	859046895
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	1303501039
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	8	5.0	858610933
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	11	4.0	850815810
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	14	4.0	851766286

In [8]:
```
#checking null values
movies_dataset.isnull().sum()
```

Out[8]:
```
movieId      0
title        0
genres       0
userId       0
rating       0
timestamp    0
dtype: int64
```

In [9]:
```
#checking datatype of values in each column
movies_dataset.dtypes
```

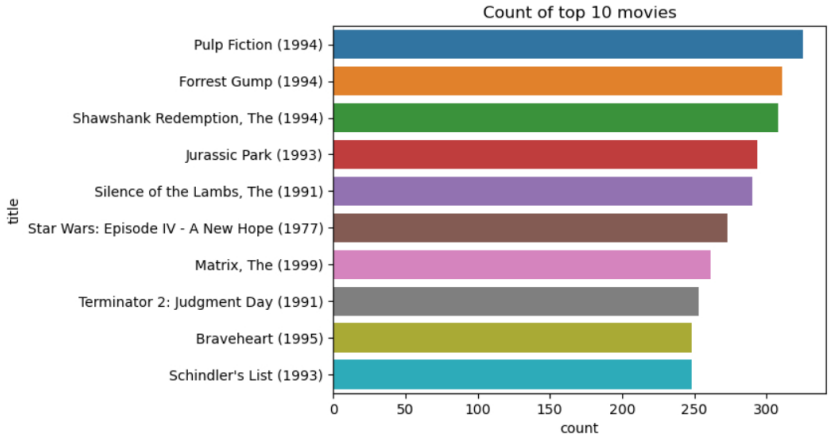
Out[9]:
```
movieId       int64
title        object
genres       object
userId        int64
rating      float64
timestamp     int64
dtype: object
```

In [ ]:
```
#Exploratory Data Analysis
```

In [10]:
```
#count of top movies
g1 = movies_dataset.groupby('title')['title'].count().sort_values(ascending=False)
g1
```

Out[10]:
```
title
Pulp Fiction (1994)                        325
Forrest Gump (1994)                        311
Shawshank Redemption, The (1994)           308
Jurassic Park (1993)                       294
Silence of the Lambs, The (1991)           290
                                          ...
Kind Lady (1935)                             1
Killing of Sister George, The (1968)         1
Killing Season (2013)                        1
Killer Klowns from Outer Space (1988)        1
À nous la liberté (Freedom for Us) (1931)    1
Name: title, Length: 10323, dtype: int64
```

In [11]:
```
#count of top 10 movies
sns.barplot(y=g1.index[:10],x=g1.values[:10])
plt.title('Count of top 10 movies')
plt.xlabel('count')
plt.show()
```

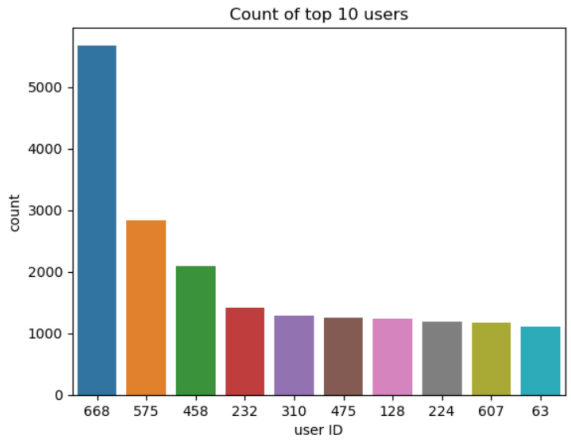


In [12]:
```
#count of users
g2 = movies_dataset.groupby('userId')['userId'].count().sort_values(ascending=False)
g2
```

Out[12]:
```
userId
668    5678
575    2837
458    2086
232    1421
310    1287
       ...
58       20
51       20
288      20
388      20
257      20
Name: userId, Length: 668, dtype: int64
```

In [13]:
```
indices = []
for i in g2.index[:10]:
    indices.append(str(i))
```

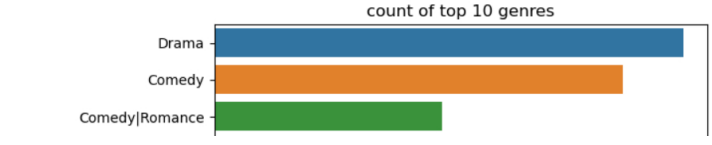
In [14]:
```
#count of top 10 users
sns.barplot(x=indices,y=g2.values[:10])
plt.title('Count of top 10 users')
plt.xlabel('user ID')
plt.ylabel('count')
plt.show()
```

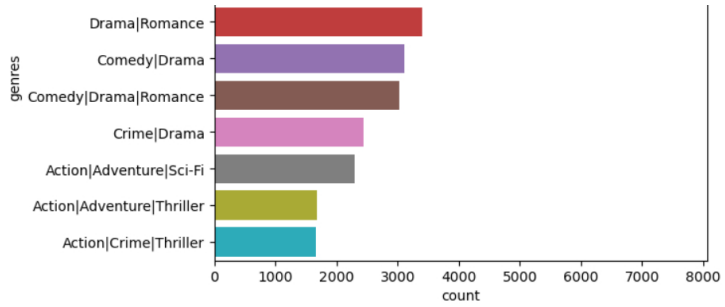


In [15]:
```
#count of genres
g3 = movies_dataset.groupby('genres')['genres'].count().sort_values(ascending=False)
g3
```

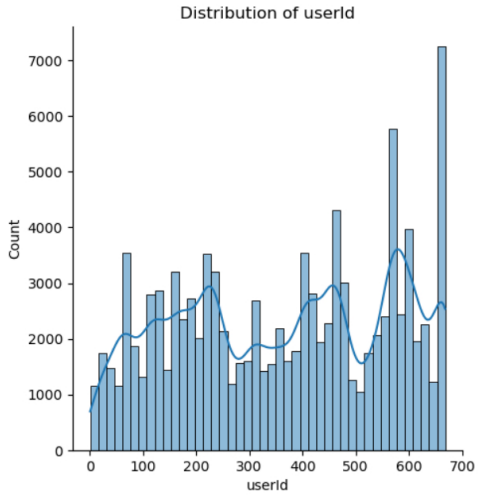
Out[15]:
```
genres
Drama                                                 7678
Comedy                                                6676
Comedy|Romance                                        3733
Drama|Romance                                         3407
Comedy|Drama                                          3101
                                                      ...
Documentary|Sci-Fi                                       1
Adventure|Animation                                      1
Action|Romance|War                                       1
Action|IMAX                                              1
Adventure|Fantasy|Horror|Romance|Sci-Fi|Thriller         1
Name: genres, Length: 938, dtype: int64
```

In [16]:
```
#count of top 10 genres
sns.barplot(y=g3.index[:10],x=g3.values[:10])
plt.title('count of top 10 genres')
plt.xlabel('count')
plt.show()
```

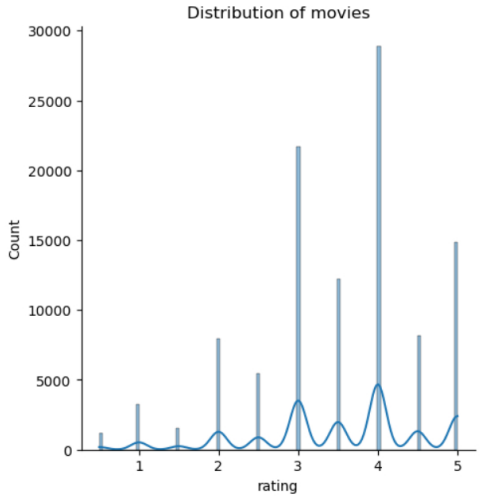




In [17]:
```
#distribution of user ID
sns.displot(movies_dataset['userId'],kde=True)
plt.title('Distribution of userId')
plt.show()
```



In [18]:
```
#distribution of movies
sns.displot(movies_dataset['rating'],kde=True)
plt.title('Distribution of movies')
plt.show()
```



In [19]:
```
#average ratings and total movies count as per genre level
g4 = movies_dataset.groupby('genres').agg({'rating':'mean','title':'count'})
g4.columns = ['Average Ratings','Total Movies']
g4 = g4.sort_values(by=['Average Ratings','Total Movies'],ascending=False)
g4
```

Out[19]:

genres	Average Ratings	Total Movies
Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi|Thriller	5.0	2
Animation|Fantasy|Sci-Fi|Thriller	5.0	2
Action|Adventure|Comedy|Crime|Romance|Thriller	5.0	1
Action|Animation|Crime|Sci-Fi	5.0	1
Action|Comedy|Drama|Romance	5.0	1
...	...	...
Children|Comedy|Drama|Musical|Romance	0.5	2
Adventure|Drama|Romance|Sci-Fi|Thriller	0.5	1
Children|Comedy|Crime	0.5	1
Children|Comedy|Drama|Musical	0.5	1
Comedy|Documentary|Romance	0.5	1

938 rows × 2 columns

Algorithms used for movie recommendation system

```
Algorithm for problem statement 1
def algo_1(genre,number_of_recommendations,minimum_reviews):
    try:
        closest_match = difflib.get_close_matches(genre,movies_dataset.genres.unique())[0]
        closest_genre_movies_dataset = movies_dataset[movies_dataset['genres']==closest_match]
        review_groupby = closest_genre_movies_dataset.groupby('title').agg({'rating':['mean','count']})
        review_groupby.columns = ['avg_ratings','count_ratings']
        minimun_reviews_groupby =  review_groupby[review_groupby['count_ratings']>=int(minimum_reviews)]
        sorted_review_groupby = minimun_reviews_groupby.sort_values(by='avg_ratings',ascending=False)
        top_5_recommendations = sorted_review_groupby.head(number_of_recommendations)
        return top_5_recommendations
    except Exception as e:
        print(e)
```

```
Algorithm for problem statement 2
def algo_2(movie,number_of_recommendations):
        try:
            movie = movie+' '+'()'
            closest_match_movie = difflib.get_close_matches(movie,movies_dataset.title.unique())[0]
            genre = movies_dataset[movies_dataset['title']==closest_match_movie]['genres']
            closest_genre_movies_dataset = movies_dataset[movies_dataset['genres']==genre.values[0]]
['title'].reset_index(drop=True).unique()
            top_n_recommendations = closest_genre_movies_dataset[:number_of_recommendations]
            sr_no = []
            m=1
            for l in range(len(top_n_recommendations)):
                sr_no.append(m)
                m+=1
            return pd.DataFrame({'Sr no.':sr_no,'movies':top_n_recommendations})
        except Exception as e:
            print(e)
```

```
Algorithm for problem statement 3
def algo_3(user_id,num_recommendations,minimum_users):
    try:
        pivot_table = movies_dataset.pivot(columns='movieId',index='userId',values='rating').fillna(0)
        similarity_scores = cosine_similarity(pivot_table)
        target_scores = similarity_scores[user_id-1]
        limited_scores = target_scores[:minimum_users]
        listed_values = []
        for i in enumerate(limited_scores):
            listed_values.append(i)
        list_indices = []
        for j in sorted(listed_values,key=lambda x:x[1],reverse=True)[:num_recommendations]:
            list_indices.append(j[0])
        list_of_movies = []
        for k in list_indices:
            list_of_movies.append(movies_dataset[movies_dataset['userId'] == k+1]['title'].head(1).values[0])
        sr_no = []
        m=1
        for l in range(len(list_of_movies)):
            sr_no.append(m)
            m+=1
        return pd.DataFrame({'Sr no.':sr_no,'movies':list_of_movies})
    except Exception as e:
        print(e)
```

Movie Recommendation System with user interface for problem statement 1

In [20]:
```
#building user interface
genre = widgets.Text(description='Genre:')
min_reviews = widgets.IntText(description='Min Reviews:')
num_recommendations_1 = widgets.IntText(description='No of Movies:')
inputs_1 = widgets.VBox([genre,min_reviews,num_recommendations_1])
button_1 = widgets.Button(description='get info!')
interface_1 = widgets.HBox([inputs_1,button_1])
```

In [21]:
```
#algorithm and main function
def algo_mod_1(response):
    try:
        closest_match = difflib.get_close_matches(response[0],movies_dataset.genres.unique())[0]
        closest_genre_movies_dataset = movies_dataset[movies_dataset['genres']==closest_match]
        review_groupby = closest_genre_movies_dataset.groupby('title').agg({'rating':['mean','count']})
        review_groupby.columns = ['avg_ratings','count_ratings']
        minimun_reviews_groupby =  review_groupby[review_groupby['count_ratings']>=response[1]]
        sorted_review_groupby = minimun_reviews_groupby.sort_values(by='avg_ratings',ascending=False)
        top_n_recommendations = sorted_review_groupby.head(response[2])
        return top_n_recommendations
    except Exception as e:
        return e
```

In [22]:
```
#function for button
def button_1_click(b):
    response = [genre.value,min_reviews.value,num_recommendations_1.value]
    print(algo_mod_1(response))
```

In [23]:
```
#adding function to button
b = button_1.on_click(button_1_click)
```

In [40]:
```
#final system
interface_1
```

Out[40]:
```
HBox(children=(VBox(children=(Text(value='Comedy', description='Genre:'), IntText(value=100, description='Min …

                                    avg_ratings  count_ratings
title
Clerks (1994)                          3.980198            101
Ferris Bueller's Day Off (1986)        3.960938            128
Monty Python's Life of Brian (1979)    3.810680            103
Birdcage, The (1996)                   3.551887            106
Ace Ventura: Pet Detective (1994)      2.849711            173
```

Movie Recommendation System with user interface for problem statement 2

In [25]:
```
#algorithm and main function
def algo_mod_2(response):
        try:
            response[0] = response[0]+' '+'()'
            closest_match_movie = difflib.get_close_matches(response[0],movies_dataset.title.unique())[0]
            genre = movies_dataset[movies_dataset['title']==closest_match_movie]['genres']
            closest_genre_movies_dataset = movies_dataset[movies_dataset['genres']==genre.values[0]]['title'].reset_index(drop=Tr
            top_n_recommendations = closest_genre_movies_dataset[:response[1]]
            sr_no = []
            m=1
            for l in range(len(top_n_recommendations)):
                sr_no.append(m)
                m+=1
            return pd.DataFrame({'Sr no.':sr_no,'movies':top_n_recommendations})
        except Exception as e:
            return e
```

In [26]:
```
#building user interface
movie = widgets.Text(description='Movie:')
num_recommendations_2 = widgets.IntText(description='No of Movies:')
inputs_2 = widgets.VBox([movie,num_recommendations_2])
button_2 = widgets.Button(description='get info!')
interface_2 = widgets.HBox([inputs_2,button_2])
```

In [27]:
```
#function for button
def button_2_click(b):
    response = [movie.value,num_recommendations_2.value]
    print(algo_mod_2(response))
```

In [28]:
```
#adding function to button
b = button_2.on_click(button_2_click)
```

In [38]:
```
#final system
interface_2
```

Out[38]:
```
HBox(children=(VBox(children=(Text(value='Toy Story', description='Movie:'), IntText(value=5, description='No …
```

	Sr no.	movies
0	1	Toy Story (1995)
1	2	Antz (1998)
2	3	Toy Story 2 (1999)
3	4	Adventures of Rocky and Bullwinkle, The (2000)
4	5	Emperor's New Groove, The (2000)

Movie Recommendation System with user interface for problem statement 3

In [30]:
```
#algorithm and main function
def algo_mod_3(response):
    try:
        pivot_table = movies_dataset.pivot(columns='movieId',index='userId',values='rating').fillna(0)
        similarity_scores = cosine_similarity(pivot_table)
        target_scores = similarity_scores[response[0]-1]
        limited_scores = target_scores[:response[2]]
        listed_values = []
        for i in enumerate(limited_scores):
            listed_values.append(i)
        list_indices = []
        for j in sorted(listed_values,key=lambda x:x[1],reverse=True)[:response[1]]:
            list_indices.append(j[0])
        list_of_movies = []
        for k in list_indices:
            list_of_movies.append(movies_dataset[movies_dataset['userId'] == k+1]['title'].head(1).values[0])
        sr_no = []
        m=1
        for l in range(len(list_of_movies)):
            sr_no.append(m)
            m+=1
        return pd.DataFrame({'Sr no.':sr_no,'movies':list_of_movies})
    except Exception as e:
        return e
```

In [31]:
```
#building user interface
user_id = widgets.IntText(description='User ID:')
num_recommendations_3 = widgets.IntText(description='No of Movies:')
min_users = widgets.IntText(description='Min Users:')
inputs_3 = widgets.VBox([user_id,num_recommendations_3,min_users])
button_3 = widgets.Button(description='get info!')
interface_3 = widgets.HBox([inputs_3,button_3])
```

In [32]:
```
#function for button
def button_3_click(b):
    response = [user_id.value,num_recommendations_3.value,min_users.value]
    print(algo_mod_3(response))
```

In [33]:
```
#adding function to button
b = button_3.on_click(button_3_click)
```

In [39]:
```
#final system
interface_3
```

Out[39]:
```
HBox(children=(VBox(children=(IntText(value=1, description='User ID:'), IntText(value=5, description='No of Mo…
```

	Sr no.	movies
0	1	Casino (1995)
1	2	Toy Story (1995)
2	3	Heat (1995)
3	4	Toy Story (1995)
4	5	GoldenEye (1995)

In [ ]: