In [75]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [76]:
```
#loading rating dataset
ratings = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial
print(ratings.head())
```

```
   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
```

In [77]:
```
# loading movie dataset
movies = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/
print(movies.head())
```

```
   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
```

In [78]:
```
n_ratings = len(ratings)
n_movies = len(ratings['movieId'].unique())
n_users = len(ratings['userId'].unique())

print(f"Number of ratings: {n_ratings}")
print(f"Number of unique movieId's: {n_movies}")
print(f"Number of unique users: {n_users}")
print(f"Average ratings per user: {round(n_ratings/n_users, 2)}")
print(f"Average ratings per movie: {round(n_ratings/n_movies, 2)}")
```

```
Number of ratings: 100836
Number of unique movieId's: 9724
Number of unique users: 610
Average ratings per user: 165.3
Average ratings per movie: 10.37
```

In [79]:
```
user_freq = ratings[['userId', 'movieId']].groupby(
    'userId').count().reset_index()
user_freq.columns = ['userId', 'n_ratings']
print(user_freq.head())
```

```
   userId  n_ratings
0       1        232
1       2         29
2       3         39
3       4        216
4       5         44
```

In [86]:
```
# Calculating mean ratings for each movie
mean_rating = ratings.groupby('movieId')[['rating']].mean()

# Finding the movie with the lowest mean rating
lowest_rated = mean_rating['rating'].idxmin()
lowest_rated_movie = movies.loc[movies['movieId'] == lowest_rated]

# Finding the movie with the highest mean rating
highest_rated = mean_rating['rating'].idxmax()
highest_rated_movie = movies.loc[movies['movieId'] == highest_rated]

# Displaying number of people who rated the highest and lowest rated movies
num_people_highest_rated = len(ratings[ratings['movieId'] == highest_rated])
num_people_lowest_rated = len(ratings[ratings['movieId'] == lowest_rated])

print(f"Lowest Rated Movie:\n{lowest_rated_movie}")
print(f"\nHighest Rated Movie:\n{highest_rated_movie}")
print(f"\nNumber of people who rated the highest rated movie: {num_people_high
print(f"Number of people who rated the lowest rated movie: {num_people_lowest_
```

```
Lowest Rated Movie:
      movieId         title   genres
2689     3604  Gypsy (1962)  Musical

Highest Rated Movie:
    movieId            title           genres
48       53  Lamerica (1994)  Adventure|Drama

Number of people who rated the highest rated movie: 2
Number of people who rated the lowest rated movie: 1
```

In [87]:
```
# Now, we create user-item matrix using scipy csr matrix
from scipy.sparse import csr_matrix

def create_matrix(df):

    N = len(df['userId'].unique())
    M = len(df['movieId'].unique())

    # Map Ids to indices
    user_mapper = dict(zip(np.unique(df["userId"]), list(range(N))))
    movie_mapper = dict(zip(np.unique(df["movieId"]), list(range(M))))

    # Map indices to IDs
    user_inv_mapper = dict(zip(list(range(N)), np.unique(df["userId"])))
    movie_inv_mapper = dict(zip(list(range(M)), np.unique(df["movieId"])))

    user_index = [user_mapper[i] for i in df['userId']]
    movie_index = [movie_mapper[i] for i in df['movieId']]

    X = csr_matrix((df["rating"], (movie_index, user_index)), shape=(M, N))

    return X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper

X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper = create_matri
```

In [88]:
```
"""
Find similar movies using KNN
"""
def find_similar_movies(movie_id, X, k, metric='cosine', show_distance=False):

    neighbour_ids = []

    movie_ind = movie_mapper[movie_id]
    movie_vec = X[movie_ind]
    k+=1
    kNN = NearestNeighbors(n_neighbors=k, algorithm="brute", metric=metric)
    kNN.fit(X)
    movie_vec = movie_vec.reshape(1,-1)
    neighbour = kNN.kneighbors(movie_vec, return_distance=show_distance)
    for i in range(0,k):
        n = neighbour.item(i)
        neighbour_ids.append(movie_inv_mapper[n])
    neighbour_ids.pop(0)
    return neighbour_ids


movie_titles = dict(zip(movies['movieId'], movies['title']))

movie_id = 3

similar_ids = find_similar_movies(movie_id, X, k=10)
movie_title = movie_titles[movie_id]

print(f"Since you watched {movie_title}")
for i in similar_ids:
    print(movie_titles[i])
```

```
Since you watched Grumpier Old Men (1995)
Grumpy Old Men (1993)
Striptease (1996)
Nutty Professor, The (1996)
Twister (1996)
Father of the Bride Part II (1995)
Broken Arrow (1996)
Bio-Dome (1996)
Truth About Cats & Dogs, The (1996)
Sabrina (1995)
Birdcage, The (1996)
```