In [21]:
```python
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# ------------------------------
# Load Dataset
# ------------------------------
ratings = pd.read_csv(r"C:\Users\LENOVO\Downloads\ml-32m\ml-32m\ratings.csv")
movies = pd.read_csv(r"C:\Users\LENOVO\Downloads\ml-32m\ml-32m\movies.csv")

# Merge ratings and movies
data = pd.merge(ratings, movies, on='movieId')
data = data[['userId', 'title', 'rating']]

# ------------------------------
# Use subset for testing
# ------------------------------
data_subset = data[data['userId'] <= 500]   # first 500 users
data_subset = data_subset[data_subset['rating'] >= 3]  # optional: only positive ratings

# Handle duplicate ratings by taking the mean
data_subset = data_subset.groupby(['userId', 'title']).rating.mean().reset_index()

# ------------------------------
# Create User-Movie Matrix
# ------------------------------
user_movie_matrix = data_subset.pivot(index='userId', columns='title', values='rating').fillna(0)

# ------------------------------
# Compute User Similarity
# ------------------------------
user_similarity = cosine_similarity(user_movie_matrix)
user_similarity_df = pd.DataFrame(user_similarity, index=user_movie_matrix.index, columns=user_movie_matrix.index)

# ------------------------------
# Recommendation Function
# ------------------------------
def recommend_movies(user_id, user_movie_matrix, user_similarity_df, top_n=5):
    # Get similar users, excluding self
    similar_users = user_similarity_df[user_id].sort_values(ascending=False)[1:]

    recommendations = pd.Series(dtype=float)

    for other_user, similarity in similar_users.items():
        # Ratings of the other user
        other_user_ratings = user_movie_matrix.loc[other_user]
        # Movies not seen by the target user
        unseen_movies = other_user_ratings[user_movie_matrix.loc[user_id] == 0]
        # Weighted ratings
        weighted_ratings = unseen_movies * similarity
        # Add to recommendations
        recommendations = recommendations.add(weighted_ratings, fill_value=0)

    # Sort and return top N
    recommendations = recommendations.sort_values(ascending=False)
    return recommendations.head(top_n)

# ------------------------------
# Example: Recommend for user 1
# ------------------------------
recommended = recommend_movies(1, user_movie_matrix, user_similarity_df)
print("Recommended movies for User 1:\n", recommended)
```

```
Recommended movies for User 1:
 title
Shawshank Redemption, The (1994)    88.768157
Pulp Fiction (1994)                 83.240113
Matrix, The (1999)                  80.997769
Godfather, The (1972)               77.542325
American Beauty (1999)              69.749771
dtype: float64
```

In [22]:
```python
print("Movies rated by User 1:")
print(user_movie_matrix.loc[1][user_movie_matrix.loc[1] > 0])
```

```
Movies rated by User 1:
title
12 Angry Men (1957)                            5.0
Airport (1970)                                 5.0
Aliens (1986)                                  5.0
All About Eve (1950)                           5.0
Amadeus (1984)                                 4.0
                                              ...
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)      5.0
Welcome to the Dollhouse (1995)                5.0
White Balloon, The (Badkonake sefid) (1995)    5.0
Women, The (1939)                              5.0
Yojimbo (1961)                                 4.0
Name: 1, Length: 102, dtype: float64
```

In [23]:
```python
print("Top similar users to User 1:")
print(user_similarity_df[1].sort_values(ascending=False)[1:11])
```

```
Top similar users to User 1:
userId
40     0.306704
372    0.288085
59     0.241980
86     0.236558
248    0.222990
278    0.222077
378    0.221548
380    0.215528
78     0.209876
158    0.209694
Name: 1, dtype: float64
```

In [24]:
```
user_1_unseen = user_movie_matrix.loc[1] == 0
print("Number of movies User 1 hasn't rated:", user_1_unseen.sum())
```

```
Number of movies User 1 hasn't rated: 8221
```

In [ ]: