In [6]:
```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
movies=pd.read_csv(r"C:\Users\Rachitha\Desktop\Movie-Recommender-System\Dataset\movies.csv")
ratings=pd.read_csv(r"C:\Users\Rachitha\Desktop\Movie-Recommender-System\Dataset\ratings.csv")
print(movies.head())
print(ratings.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
   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 [8]:
```
print(ratings.columns)
```

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype
='object')
```

In [10]:
```
user_movie_matrix = ratings.pivot_table(index='userId' , columns='movieId' , values = 'rating')
user_movie_matrix.fillna(0,inplace=True)
print(user_movie_matrix.head())
```

```
movieId  1       2       3       4       5       6       7
8       \
userId
1           4.0     0.0     4.0     0.0     0.0     4.0
0.0     0.0
2           0.0     0.0     0.0     0.0     0.0     0.0
0.0     0.0
3           0.0     0.0     0.0     0.0     0.0     0.0
0.0     0.0
4           0.0     0.0     0.0     0.0     0.0     0.0
0.0     0.0
5           4.0     0.0     0.0     0.0     0.0     0.0
0.0     0.0

movieId  9       10      ...  193565  193567  193571  19357
3  193579  193581  \
userId                   ...
1           0.0     0.0  ...     0.0     0.0     0.0     0.
0     0.0     0.0
2           0.0     0.0  ...     0.0     0.0     0.0     0.
0     0.0     0.0
3           0.0     0.0  ...     0.0     0.0     0.0     0.
0     0.0     0.0
4           0.0     0.0  ...     0.0     0.0     0.0     0.
0     0.0     0.0
5           0.0     0.0  ...     0.0     0.0     0.0     0.
0     0.0     0.0
```

movieId	193583	193585	193587	193609
userId				
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0

[5 rows x 9724 columns]

In [11]:
```
user_similarity = cosine_similarity(user_movie_matrix)
user_similarity_df = pd.DataFrame(user_similarity,index=user_movie_matrix.index,columns=user_movie_matrix.inde
user_similarity_df.head()
```

Out[11]:

userId	1	2	3	4	5	6	7	8	9	10	...	601
userId												
1	1.000000	0.027283	0.059720	0.194395	0.129080	0.128152	0.158744	0.136968	0.064263	0.016875	...	0.080554
2	0.027283	1.000000	0.000000	0.003726	0.016614	0.025333	0.027585	0.027257	0.000000	0.067445	...	0.202671
3	0.059720	0.000000	1.000000	0.002251	0.005020	0.003936	0.000000	0.004941	0.000000	0.000000	...	0.005048
4	0.194395	0.003726	0.002251	1.000000	0.128659	0.088491	0.115120	0.062969	0.011361	0.031163	...	0.085938
5	0.129080	0.016614	0.005020	0.128659	1.000000	0.300349	0.108342	0.429075	0.000000	0.030611	...	0.068048

5 rows × 610 columns

In [34]:
```
user_id=1
user_ratings = user_movie_matrix.loc[user_id]
unrated_movies = user_ratings[user_ratings == 0].index
print(len(unrated_movies))
```

9492

In [37]:
```
def get_user_recommendations(user_id, num_recommendations=5):
    similar_users = user_similarity_df[user_id].sort_values(ascending=False)
    similar_users = similar_users[similar_users.index != user_id]

    user_ratings = user_movie_matrix.loc[user_id]
    unrated_movies = user_ratings[user_ratings == 0].index

    if len(unrated_movies) == 0:
        print("No unrated movies found for this user.")
        return None

    weighted_scores = pd.Series(0, index=unrated_movies)

    for sim_user_id, similarity_score in similar_users.items():
        sim_user_ratings = user_movie_matrix.loc[sim_user_id, unrated_movies]
        weighted_scores += sim_user_ratings * similarity_score

    # Get top recommended movie IDs
    top_scores = weighted_scores.sort_values(ascending=False).head(num_recommendations)
    recommended_movie_ids = top_scores.index.astype(int)

    # Match with movie titles
    recommended_movies = movies[movies['movieId'].isin(recommended_movie_ids)].copy()

    # Merge predicted scores with movie titles
    recommended_movies['Predicted_Score'] = recommended_movies['movieId'].map(top_scores)

    # Sort by predicted score
    recommended_movies = recommended_movies.sort_values(by='Predicted_Score', ascending=False)

    return recommended_movies[['title', 'Predicted_Score']]
```

In [40]:
```
display(get_user_recommendations(user_id=1,num_recommendations=5))
```

	title	Predicted_Score
277	Shawshank Redemption, The (1994)	215.449703
507	Terminator 2: Judgment Day (1991)	169.401182
659	Godfather, The (1972)	150.915327
2078	Sixth Sense, The (1999)	135.009864
3638	Lord of the Rings: The Fellowship of the Ring,...	131.865541

In [ ]: