```
import pandas as pd
ratings = pd.read_csv('/content/ratings.csv')
movies = pd.read_csv('/content/movies.csv')
print(ratings.head())
print(movies.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
   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
```

```
data = pd.merge(ratings, movies, on='movieId')
```

```
user_movie_matrix = data.pivot_table(index='userId', columns='title', values='rating')
user_movie_matrix = user_movie_matrix.fillna(0)
```

```
from sklearn.feature_extraction.text import CountVectorizer

# Create a CountVectorizer instance
count_vectorizer = CountVectorizer(tokenizer=lambda x: x.split('|'))
genre_matrix = count_vectorizer.fit_transform(movies['genres'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarn
  warnings.warn(
```

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_sim = cosine_similarity(genre_matrix, genre_matrix)
```

```
def recommend_movies_content_based(movie_title, cosine_sim=cosine_sim):
    if movie_title not in movies['title'].values:
        return "Movie not found in the database."

    idx = movies[movies['title'] == movie_title].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]
    movie_indices = [i[0] for i in sim_scores]
    return movies['title'].iloc[movie_indices]

# Taking input for content-based recommendation
movie_title = input("Enter a movie title: ")
recommended_movies = recommend_movies_content_based(movie_title)
print("Recommended movies based on content-based filtering:")
print(recommended_movies)
```

```
Enter a movie title: Jumanji
Recommended movies based on content-based filtering:
Movie not found in the database.
```

```
from sklearn.metrics.pairwise import cosine_similarity

# Compute item-item similarity
movie_similarity = cosine_similarity(user_movie_matrix.T)
```

```
import numpy as np
def recommend_movies_collaborative(user_id, user_movie_matrix=user_movie_matrix, mc
    if user_id not in user_movie_matrix.index:
        return "User ID not found in the database."

    user_ratings = user_movie_matrix.loc[user_id]
    similar_scores = movie_similarity.dot(user_ratings)
    similar_scores = similar_scores / np.sum(user_movie_matrix != 0, axis=0)

    # Convert to a Pandas Series and sort
    similar_scores_series = pd.Series(similar_scores, index=user_movie_matrix.colum
    similar_scores_series = similar_scores_series.sort_values(ascending=False)

    # Return top 10 recommended movies
    return similar_scores_series.index[:10]

# Taking input for collaborative filtering recommendation
user_id = int(input("Enter your user ID: "))
recommended_movies = recommend_movies_collaborative(user_id)
print("Recommended movies based on collaborative filtering:")
print(recommended_movies)
```

```
Enter your user ID: 2
Recommended movies based on collaborative filtering:
     Index(['The Jinx: The Life and Deaths of Robert Durst (2015)',
            'Visit, The (2015)', 'Adventures of Mowgli: The Kidnapping (1968)',
            'What Men Still Talk About (2011)',
            'Heart of a Dog (Sobachye serdtse) (1988)',
            'Priklyucheniya Kapitana Vrungelya (1979)',
            'Andrei Rublev (Andrey Rublyov) (1969)',
            'Ernest & Célestine (Ernest et Célestine) (2012)',
            'Bobik Visiting Barbos (1977)',
            'From Up on Poppy Hill (Kokuriko-zaka kara) (2011)'],
           dtype='object', name='title')
```

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```