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

Importing the required libraries and the files

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving links.csv to links.csv
Saving movies.csv to movies.csv
Saving ratings.csv to ratings.csv
Saving tags.csv to tags.csv
```

Understanding the dataset and its shape

```
import pandas as pd

# Loading the files
links = pd.read_csv('links.csv')
movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
tags = pd.read_csv('tags.csv')
# Display the first few rows of the DataFrame
print ("First few rows of links dataframe\n")
print(links.head(),"\n",links.shape)
print ("First few rows of movies dataframe\n")
print(movies.head(),"\n",movies.shape)
print ("First few rows of ratings dataframe\n")
print(ratings.head(),"\n",ratings.shape)
print ("First few rows of tags dataframe\n")
print(tags.head(),"\n",tags.shape)
```

```
First few rows of links dataframe

   movieId  imdbId   tmdbId
0        1  114709    862.0
1        2  113497   8844.0
2        3  113228  15602.0
3        4  114885  31357.0
4        5  113041  11862.0
 (9742, 3)
First few rows of movies dataframe

   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
 (9742, 3)
First few rows of ratings dataframe

   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
 (100836, 4)
First few rows of tags dataframe

   userId  movieId              tag   timestamp
0       2    60756            funny  1445714994
1       2    60756  Highly quotable  1445714996
2       2    60756     will ferrell  1445714992
3       2    89774     Boxing story  1445715207
4       2    89774              MMA  1445715200
 (3683, 4)
```

Recommender system

The recommender system used here is based out of collaborative filtering and it is item based.
As the asked question, "Recommender system that allows users to input a movie they like (in the data set) and recommends ten other movies for them to watch." requires the input of liked movie (item) by the user, we need to get the similarity based on the items.
Cosine similarity is used to find the similarity between the movies (items) and the value lies between 0 and 1, meaning they are not related to highly related. If the user watches and likes the movie, then he is more likely to give a higher rating. But if he doesnt like the movie, then lesser rating is more likely.
The features that we are using here are the user ids, movie ratings given by them and the movie id/name.

The dataframes ratings and movies are merged to get the movie names and the userid/ratings columns.

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

```
# Merge ratings and movies on 'movieId' to get a complete dataset
movie_ratings = pd.merge(ratings, movies, on='movieId')
movie_ratings.head()
```

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller

Pivot table has been created based on the rating values.

```
# Create a user-item matrix
user_movie_matrix = movie_ratings.pivot_table(index='userId', columns='title', values='rating')
user_movie_matrix.fillna(0, inplace=True)
user_movie_matrix.head()
```

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	[REC]³ 3 Génesis (2012)	anohan Th Flow We S That D - Th Mov (201
userId																
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	C
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	C
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	C
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	C

As seen above, the dataframe has the userid as index. However, we need the movies as index, as cosine similarity is found for the movies (not for the user).
The matrix is transposed appropriately, so that movie names come up in the index.

```
user_movie_matrix_transpose = user_movie_matrix.T
user_movie_matrix_transpose
```

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
title																					
'71 (2014)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0
'Hellboy': The Seeds of Creation (2004)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
'Round Midnight (1986)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
'Salem's Lot (2004)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
'Til There Was You (1997)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
eXistenZ (1999)	0.0	0.0	0.0	0.0	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	4.5	0.0	0.0
xXx (2002)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.5	0.0	2.0
xXx: State of the Union (2005)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5
¡Three Amigos! (1986)	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
À nous la liberté (Freedom for Us) (1931)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

9719 rows × 610 columns

A dataframe is constructed for ease of retrieval based on the cosine similarity.

```
from sklearn.metrics.pairwise import cosine_similarity


movie_similarity = cosine_similarity(user_movie_matrix_transpose)
movie_similarity_df = pd.DataFrame(movie_similarity, index=user_movie_matrix.columns, columns=user_movie_matrix.columns)
movie_similarity_df
```

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	(
title															
'71 (2014)	1.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.141653	0.000000	...	0.000000	0.342055	0.543305	0
'Hellboy': The Seeds of Creation (2004)	0.000000	1.000000	0.707107	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0
'Round Midnight (1986)	0.000000	0.707107	1.000000	0.000000	0.000000	0.0	0.176777	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0
'Salem's Lot (2004)	0.000000	0.000000	0.000000	1.000000	0.857493	0.0	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0
'Til There Was You (1997)	0.000000	0.000000	0.000000	0.857493	1.000000	0.0	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
eXistenZ (1999)	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.211467	0.216295	0.097935	0.132489	...	0.000000	0.000000	0.000000	0
xXx (2002)	0.139431	0.000000	0.000000	0.000000	0.000000	0.0	0.089634	0.000000	0.276512	0.019862	...	0.069716	0.305535	0.173151	0
xXx: State of the Union (2005)	0.327327	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.156764	0.000000	...	0.000000	0.382543	0.177838	0
¡Three Amigos! (1986)	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.372876	0.180009	0.169385	0.249586	...	0.180009	0.000000	0.000000	0
À nous la															

A basic function has been written, which takes in a movie name and recommends 10 other movies based on the cosine similarity.

```
def recommend_movies(movie_title, movie_similarity_df, n_recommendations=10):
    if movie_title not in movie_similarity_df.index:
        return f"'{movie_title}' not found in the dataset."

    # Get the similarity scores for the input movie
    similar_movies = movie_similarity_df[movie_title].sort_values(ascending=False)

    # Recommend the top N movies (excluding the input movie itself)
    recommended_movies = similar_movies.index[1:n_recommendations+1]
    return recommended_movies
```

title	'71 (2014)
'71 (2014)	1.0
City of Lost Souls, The (Hyôryuu-gai) (2000)	1.0
Clown (2014)	1.0
Strange Circus (Kimyô na sâkasu) (2005)	1.0
Ginger Snaps: Unleashed (2004)	1.0
...	...
Girl on the Bridge, The (Fille sur le pont, La) (1999)	0.0
Girl Who Played with Fire, The (Flickan som lekte med elden) (2009)	0.0
Girl Who Kicked the Hornet's Nest, The (Luftslottet som sprängdes) (2009)	0.0
Girl Walks Into a Bar (2011)	0.0
À nous la liberté (Freedom for Us) (1931)	0.0

9719 rows × 1 columns

dtype: float64

The function is tested by giving a movie name and the cosine similarity dataframe that has been created previously.
Ten movies has beens selected based on the descending order of scores, with 1 being highest and 0 the lowest.

```
print (recommend_movies ("'71 (2014)",movie_similarity_df))
```

```
Index(['City of Lost Souls, The (Hyôryuu-gai) (2000)', 'Clown (2014)',
       'Strange Circus (Kimyô na sâkasu) (2005)',
       'Ginger Snaps: Unleashed (2004)',
       'Ginger Snaps Back: The Beginning (2004)', 'Get on the Bus (1996)',
       'Collector, The (2009)', 'Prince of Darkness (1987)',
       'Gen-X Cops (1999)', 'Stingray Sam (2009)'],
      dtype='object', name='title')
```

Conclusion

The required libraries and dataset has been imported to dataframes.
The used features are the user ids, movie ratings given by them and the movie id/name.
The used recommender system is based out of collaborative filtering and item based.
Cosine similarity is used to find the similarity between the movies (items) and a dataframe is created.
Basic python function is written and tested to get the movie name as input and provide 10 similar movie recommendations.

References

"Building a Content-Based Movie Recommender System with Cosine Similarity". Retrieved, https://medium.com/@imeshadilshani212/building-a-content-based-movie-recommender-system-with-cosine-similarity-557d3c78ca8f

"The cosine similarity and its use in recommendation systems". Retrieved, https://naomy-gomes.medium.com/the-cosine-similarity-and-its-use-in-recommendation-systems-cb2ebd811ce1