1. Implement the "user-based similarity" method of Collaborative Filtering. Upload the "ratings.csv". Create a frame for this file.

```
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 rating (1).csv to rating (1) (1).csv
```

```
import pandas as pd
df=pd.read_csv("/content/rating (1).csv")
df.head(5)
```

	userId	movieId	rating	timestamp
0	1	2	3.5	02-04-2005 23:53
1	1	29	3.5	02-04-2005 23:31
2	1	32	3.5	02-04-2005 23:33
3	1	47	3.5	02-04-2005 23:32
4	1	50	3.5	02-04-2005 23:29

2. Drop the timestamp column and find the number of unique movies and unique users.

```
df.drop('timestamp', axis= 1,inplace=  True)
print("No. of unique users=",len(df['userId'].unique()))
print("No. of unique movies=",len(df['movieId'].unique()))
```

```
No. of unique users= 7120
No. of unique movies= 14026
```

3. Create a pivot table or matrix with users as rows and movies as columns. Matrix entries will represent the ratings given by the users. This will be a sparse matrix and those movies not watched will be NaN. Replace all such NaN values by zeros.

```
user_movies_df=df.pivot(index="userId",columns="movieId",values="rating").reset_index(drop=True)
user_movies_df.index=df.userId.unique()
user_movies_df.fillna(0,inplace=True)
user_movies_df.head(5)
```

movieId	1	2	3	4	5	6	7	8	9	10	...	129350	129354	129428	129707	130052	130073	130219	130462	130490	130642
1	0.0	3.5	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
2	0.0	0.0	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
3	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
4	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	4.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	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	0.0	0.0	0.0	0.0	0.0

5 rows × 14026 columns

4. Compute the cosine similarity matrix between all pairs of users.The diagonal values of this matrix will be 1. Print the matrix.

```
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine,correlation
user_sim=1-pairwise_distances(user_movies_df.values,metric="cosine")
user_sim_df=pd.DataFrame(user_sim)
user_sim_df.index=df.userId.unique()
user_sim_df.columns=df.userId.unique()
user_sim_df.iloc[0:5,0:5]
```

	1	2	3	4	5
1	1.000000	0.102916	0.261198	0.029091	0.139199
2	0.102916	1.000000	0.180124	0.064014	0.140042
3	0.261198	0.180124	1.000000	0.057323	0.156755
4	0.029091	0.064014	0.057323	1.000000	0.300828

5. Set the diagonal values as 0.0, since we need to find other users who are similar to a specific user.

```
import numpy as np
np.fill_diagonal(user_sim,0)
user_sim_df.iloc[0:5,0:5]
```

	1	2	3	4	5
1	0.000000	0.102916	0.261198	0.029091	0.139199
2	0.102916	0.000000	0.180124	0.064014	0.140042
3	0.261198	0.180124	0.000000	0.057323	0.156755
4	0.029091	0.064014	0.057323	0.000000	0.300828
5	0.139199	0.140042	0.156755	0.300828	0.000000

6. Filter similar users. To find most similar users, the maximum values of each column can be filtered. For example, the most similar user to first 5 users with userId from 1 to 5 can be printed as

```
user_sim_df.idxmax(axis=1)[0:5]
```

```
1    2595
2    5964
3     475
4     242
5    1367
dtype: int64
```

7. This result says that the most similar user to user1 is userId 2595 and so on.This also means that both have watched several movies in common and rated very similarly. This can be verified with the movies dataset. Load the movies dataset, drop the "genres" column.

```
uploaded=files.upload()
movies_df=pd.read_csv("/content/movie (1).csv")
movies_df.drop('genres',axis=1,inplace=True)
movies_df.iloc[0:5,0:5]
```

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 movie (1).csv to movie (1) (2).csv
```

	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)

```
movies_df=pd.read_csv("/content/movie (1).csv")
movies_df.drop('genres',axis=1,inplace=True)
movies_df.iloc[0:5,0:5]
```

	movieId	title
2	3	Grumpier Old Men (1995)

8. Find common movies of similar users.

+ Code + Text

```
def get_user_similar_movies(user1,user2):
  common=df[df.userId==user1].merge(df[df.userId==user2],on="movieId",how="Inner")
  return common.merge(movies_df,on="movieId")
```

9. Find the common movies of user 1 and user 2595.Print only those common movies with ratings above 4.

```
common_movies= get_user_similar_movies(1,2595)
common_movies[(common_movies.rating_x >=4.0) & ((common_movies.rating_y >=4.0))]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-39f86b421fb9> in <module>
----> 1 common_movies= get_user_similar_movies(1,2595)
      2 common_movies[(common_movies.rating_x >=4.0) & ((common_movies.rating_y >=4.0))]

NameError: name 'get_user_similar_movies' is not defined
```

SEARCH STACK OVERFLOW

10. Finding user similarity does not work for new users. We need to wait until the new user buys a few items and rates them. Only then user preferences can be founnd and recommendations can be made. This is called the "cold-start" problem in recommender systems. This is overcome by using item-based similarity methods.

Colab paid products - Cancel contracts here