Develop an user based collaborative filtering model on the movie dataset.

In [1]:

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
movie = pd.read_csv("movies.csv")
movie.head()
```

Out[2]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [3]:

```
rating = pd.read_csv("ratings.csv")
rating.head()
```

Out[3]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

In [4]:

```
movie_final = pd.merge(movie,rating,on="movieId")
movie_final.head()
```

Out[4]:

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	7	3.0	851866703
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	9	4.0	938629179
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	13	5.0	1331380058
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	15	2.0	997938310
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	19	3.0	855190091

In [5]:

```
movie_final.drop(["genres","timestamp"],axis=1,inplace=True)
movie_final.head()
```

Out[5]:

	movieId	title	userId	rating
0	1	Toy Story (1995)	7	3.0
1	1	Toy Story (1995)	9	4.0
2	1	Toy Story (1995)	13	5.0
3	1	Toy Story (1995)	15	2.0
4	1	Toy Story (1995)	19	3.0

In [6]:

```
movie_final.shape
```

Out[6]:

```
(100004, 4)
```

In [7]:

```
# user-item matrix
movie_user_matrix = movie_final.pivot_table(index="userId",columns="title",values="rating")
```

In [8]:

```
movie_user_matrix.head()
```

Out[8]:

title	"Great Performances" Cats (1998)	$9.99 (2008)	'Hellboy': The Seeds of Creation (2004)	'Neath the Arizona Skies (1934)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'burbs, The (1989)	'night Mother (1986)
userId									
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 9064 columns

In [9]:

```
movie_user_matrix.fillna(0,inplace=True)
```

In [10]:

```
movie_user_matrix.head()
```

Out[10]:

title	"Great Performances" Cats (1998)	$9.99 (2008)	'Hellboy': The Seeds of Creation (2004)	'Neath the Arizona Skies (1934)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'burbs, The (1989)	'night Mother (1986)
userId									
1	0.0	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	0.0
3	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	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	0.0

5 rows × 9064 columns

In [11]:

```
from sklearn.metrics.pairwise import cosine_similarity
import operator
def similar_users(user_id, matrix, k=3):
    # create a df of just the current user
    user = matrix[matrix.index == user_id]

    # and a df of all other users
    other_users = matrix[matrix.index != user_id]

    # calc cosine similarity between user and each other user
    similarities = cosine_similarity(user,other_users)[0].tolist()

    # create list of indices of these users
    indices = other_users.index.tolist()

    # create key/values pairs of user index and their similarity
    index_similarity = dict(zip(indices, similarities))

    # sort by similarity
    index_similarity_sorted = sorted(index_similarity.items(), key=operator.itemgetter(1))
    index_similarity_sorted.reverse()

    # grab k users off the top
    top_users_similarities = index_similarity_sorted[:k]
    users = [u[0] for u in top_users_similarities]

    return users
```

In [12]:

```
current_user = 300
similar_user_indices = similar_users(current_user, movie_user_matrix)
print(similar_user_indices)
```

```
[282, 243, 523]
```

In [ ]: