Data Matrix using 200 users only

Generate data matrix between users and movies and value is rating

```
[154]
user_movie_rating = movie_data.pivot_table(index='userId', columns='title', values='rating')[:200]
print(user_movie_rating.shape)
user_movie_rating.head(3)
```

```
(200, 9719)
```

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)	anohana: The Flower We Saw That Day - The Movie (2013)	eXistenZ (1999)	xXx (2002)	xXx: State of the Union (2005)	¡Three Amigos! (1986)	1 (F f
userId																					
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

3 rows x 9719 columns

I replaced NAN Values with zero

```
[99]
user_movie_rating.replace(np.nan, 0,inplace=True)
user_movie_rating.head(2)
```

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)	anohana: The Flower We Saw That Day - The Movie (2013)	eXistenZ (1999)	xXx (2002)	xXx: State of the Union (2005)	¡Three Amigos! (1986)	1 (F f
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	0.0	0.0	0.0	0.0	4.0	
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	0.0	0.0	0.0	0.0	0.0	

2 rows x 9719 columns

Cosin similarity matrix implementation

Cosin similarit Function

```
[900]
def Cosin_Similarity(data_Matrix,film_name):
    similarty_df = pd.DataFrame(columns=['title', 'Cosin_similarty'])
    film=data_Matrix[film_name]
    for i in data_Matrix:
        dot_product = np.dot(film, data_Matrix[i])
        norm_a = np.linalg.norm(film)
        norm_b = np.linalg.norm(data_Matrix[i])
        similarity=dot_product / (norm_a * norm_b)
        similarty_df=similarty_df.append({'title': i, 'Cosin_similarty': similarity},ignore_index=True)
    return similarty_df
```

```
{}
```

Toy sory ratings by all users

Use Data matrix to match Toy Story (1995)

908]

```
Toy_story_ratings = user_movie_rating['Toy Story (1995)']
Toy_story_ratings
```

```
userId
1      4.0
2      0.0
3      0.0
4      0.0
5      4.0
      ...
606    2.5
607    4.0
608    2.5
609    3.0
610    5.0
Name: Toy Story (1995), Length: 610, dtype: float64
```

902]

I obtained Movies like toy story using cosin similarity

```
movies_like_Toy_story=Cosin_Similarity(user_movie_rating,'Toy Story (1995)').sort_values('Cosin_similarty', ascending=False)
movies_like_Toy_story.dropna(inplace=True)
movies_like_Toy_story.sort_values('Cosin_similarty', ascending=False).head(10)
```

	title	Cosin_similarty
8871	Toy Story (1995)	1.000000
8872	Toy Story 2 (1999)	0.572601
4662	Jurassic Park (1993)	0.565637
4337	Independence Day (a.k.a. ID4) (1996)	0.564262
8001	Star Wars: Episode IV - A New Hope (1977)	0.557388
3158	Forrest Gump (1994)	0.547096
5103	Lion King, The (1994)	0.541145
8003	Star Wars: Episode VI - Return of the Jedi (1983)	0.541089
5711	Mission: Impossible (1996)	0.538913
3656	Groundhog Day (1993)	0.534169

> **I droped all Nan values which is user didnt give rate to this movie**
>
> **The first top 10 movies similar to toy story**

I joined rating count too to obtaind on movies similar to toy story and also rated by more than 100 users

[903]
```
cos_Toy_Story = movies_like_Toy_story.join(ratings_mean_count['rating_counts'],on='title')
cos_Toy_Story.head()
```

	title	Cosin_similarty	rating_counts
8871	Toy Story (1995)	1.000000	215
8872	Toy Story 2 (1999)	0.572601	97
4662	Jurassic Park (1993)	0.565637	238
4337	Independence Day (a.k.a. ID4) (1996)	0.564262	202
8001	Star Wars: Episode IV - A New Hope (1977)	0.557388	251

[904]

[904]
```
cos_Toy_Story[cos_Toy_Story ['rating_counts']>100].sort_values('Cosin_similarty', ascending=False).head(10)
```

	title	Cosin_similarty	rating_counts
8871	Toy Story (1995)	1.000000	215
4662	Jurassic Park (1993)	0.565637	238
4337	Independence Day (a.k.a. ID4) (1996)	0.564262	202
8001	Star Wars: Episode IV - A New Hope (1977)	0.557388	251
3158	Forrest Gump (1994)	0.547096	329
5103	Lion King, The (1994)	0.541145	172
8003	Star Wars: Episode VI - Return of the Jedi (1983)	0.541089	196
5711	Mission: Impossible (1996)	0.538913	162
3656	Groundhog Day (1993)	0.534169	143
744	Back to the Future (1985)	0.530381	171

I did the same steps to "Waiting to Exhale (1995)"

[907]
```
cos_Waiting_to_exhale[cos_Waiting_to_exhale ['rating_counts']>100].sort_values('Cosin_similarty', ascending=False).head(10)
```

	title	Cosin_similarty	rating_counts
3406	Ghost (1990)	0.189254	115
2139	Dances with Wolves (1990)	0.184297	164
7755	Sleepless in Seattle (1993)	0.182025	106
565	Apollo 13 (1995)	0.174043	201
3048	Firm, The (1993)	0.149059	101
5862	Mrs. Doubtfire (1993)	0.148164	144
7421	Schindler's List (1993)	0.147778	220
6394	Outbreak (1995)	0.144176	101
3158	Forrest Gump (1994)	0.144128	329
836	Batman (1989)	0.143980	189

> **and these the first top movies similar to waiting to exhale and rated by more than 100 users**

Part three

```
[11]
Movies_User_Ratings=user_movie_rating.T # Transpose Movie user table
Movies_User_Ratings.head(10)
```

userId	1	2	3	4	5	6	7	8	9	10	...	191	192	193	194	195	196	197	198	199	200
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	0.0
'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
'Tis the Season for Love (2015)	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
'burbs, The (1989)	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
'night Mother (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
(500) Days of Summer (2009)	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
*batteries not included (1987)	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

10 rows × 200 columns

> **I transposed Data Matrix. So each column is certain user with all rating that he gave to all movies in original data.**
>
> **as i mentioned before i filled non rated film with zero to handle some calculations that i do to recommend similarties**

```
[212]
User_200_Ratings=Movies_User_Ratings[200]
User_200_Ratings.head(10)
```

```
title
'71 (2014)                                0.0
'Hellboy': The Seeds of Creation (2004)   0.0
'Round Midnight (1986)                    0.0
'Salem's Lot (2004)                       0.0
'Til There Was You (1997)                 0.0
'Tis the Season for Love (2015)           0.0
'burbs, The (1989)                        0.0
'night Mother (1986)                      0.0
(500) Days of Summer (2009)               0.0
*batteries not included (1987)            0.0
Name: 200, dtype: float64
```

> **Here all rated that user 200 gave to movies**

Users like USer200

[202]

```
Users_like_userID200=Cosin_Similarity(Movies_User_Ratings,200).sort_values('Cosin_similarty', ascending=False)
Users_like_userID200.dropna(inplace=True)
Top_ten_similar_users=Users_like_userID200.sort_values('Cosin_similarty', ascending=False).head(10)
Top_ten_similar_users
```

	userId	Cosin_similarty
199	200.0	1.000000
67	68.0	0.460507
140	141.0	0.403566
131	132.0	0.399698
44	45.0	0.384170
165	166.0	0.361068
63	64.0	0.356268
176	177.0	0.355917
128	129.0	0.353020
168	169.0	0.337422

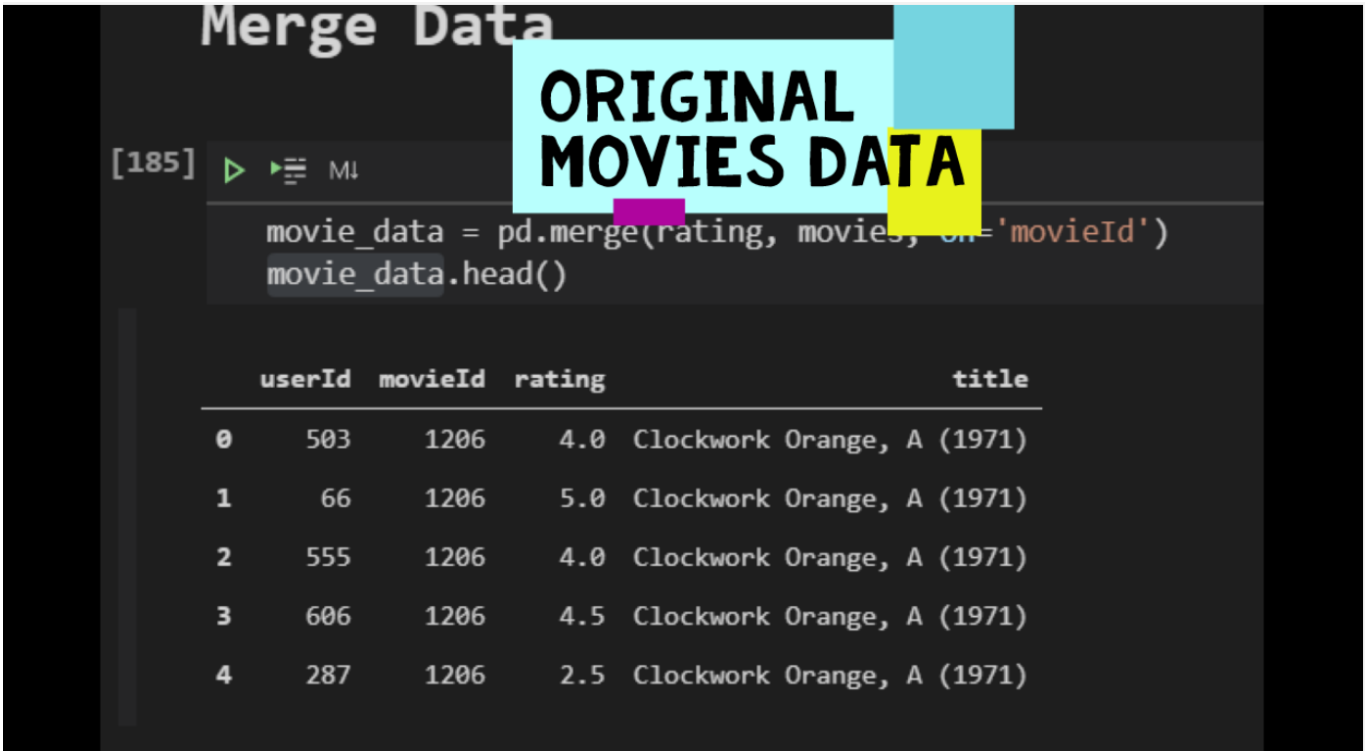
> **The Top ten similar users to users of ID 200**
>
> **I did the same process that i did in item based algorithm. by getting cosin similarity between users and sorted values of similarity asending to get the top ones**

Intersection between original merging data with Top ten similar users on userId

[210]

```
intersection_df = pd.merge(movie_data, Top_ten_similar_users, how ='inner', on ='userId')
#generate new dataFrame groupby title on mean of ratings for ten similar users
ratings_mean_for_ten_users = pd.DataFrame(intersection_df.groupby('title')['rating'].mean())
# dropes Movies that isn't rated by ten similar users
ratings_mean_for_ten_users.dropna(inplace=True)
#sort mean of rating asending to recommend the higher three movies to user of id 200
ratings_mean_for_ten_users.sort_values("rating",ascending=False).head(3)
```

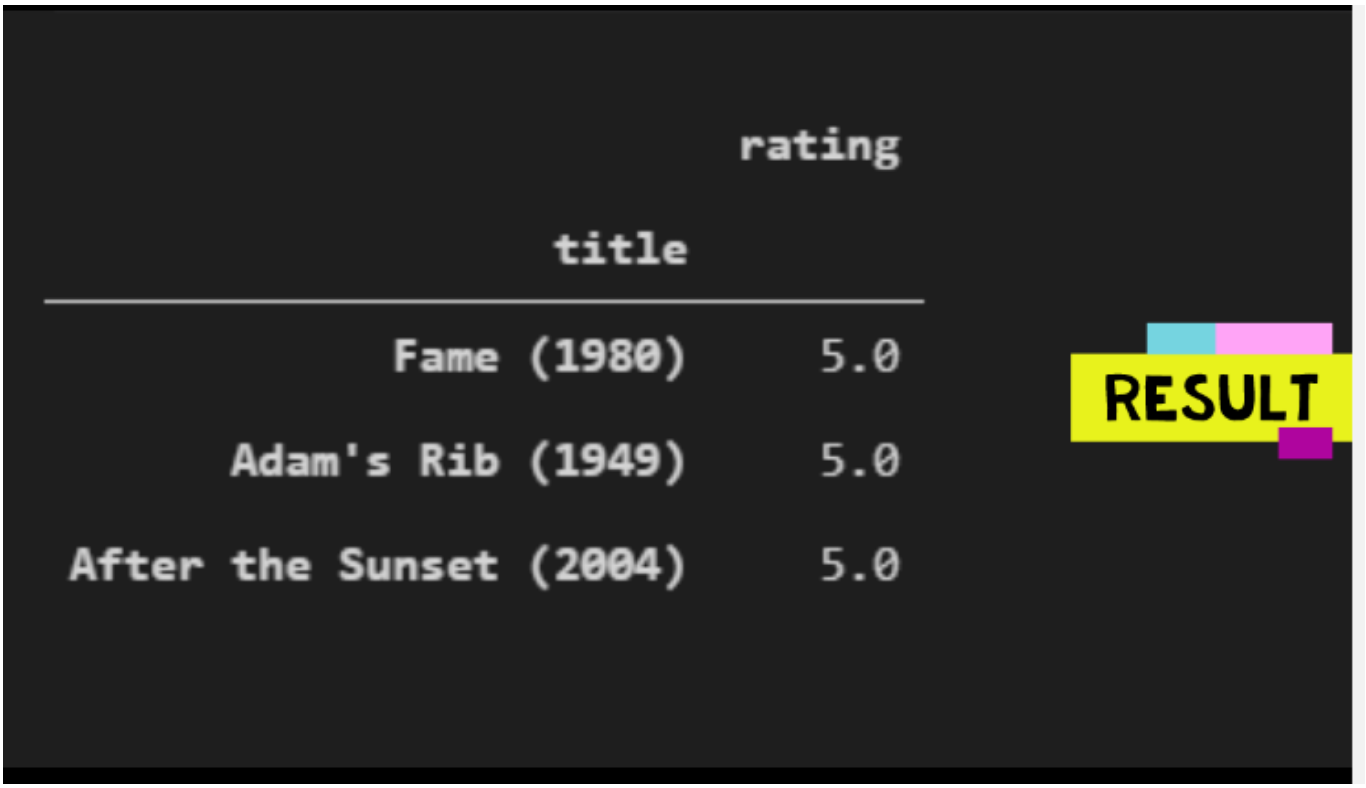
> **I did intersection between original movies data and dataFrame of top ten similar users to get all rating and data for top ten similar useres only**



	userId	movieId	rating	title
0	503	1206	4.0	Clockwork Orange, A (1971)
1	66	1206	5.0	Clockwork Orange, A (1971)
2	555	1206	4.0	Clockwork Orange, A (1971)
3	606	1206	4.5	Clockwork Orange, A (1971)
4	287	1206	2.5	Clockwork Orange, A (1971)

	userId	Cosin_similarty
199	200.0	1.000000
67	68.0	0.460507
140	141.0	0.403566
131	132.0	0.399698
44	45.0	0.384170
165	166.0	0.361068
63	64.0	0.356268
176	177.0	0.355917
128	129.0	0.353020
168	169.0	0.337422

Result



> **These Three top movies that got higher rate by 10 similar users so recommeded them to user of id 200**