RECOMMENDATION SYSTEM CODE EXPLAIN

Dataset

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
...	...	...	...	...
99995	880	476	3	880175444
99996	716	204	5	879795543
99997	276	1090	1	874795795
99998	13	225	2	882399156
99999	12	203	3	879959583

100000 rows × 4 columns

Pivot table

Pivot Table

942 x 942

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
user_id																					
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	4.0	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	0.0	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	0.0	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	0.0	0.0	0.0
5	4.0	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
939	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	0.0	0.0	0.0
940	0.0	0.0	0.0	2.0	0.0	0.0	4.0	5.0	3.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
941	5.0	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	0.0	0.0	0.0	0.0	0.0
942	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

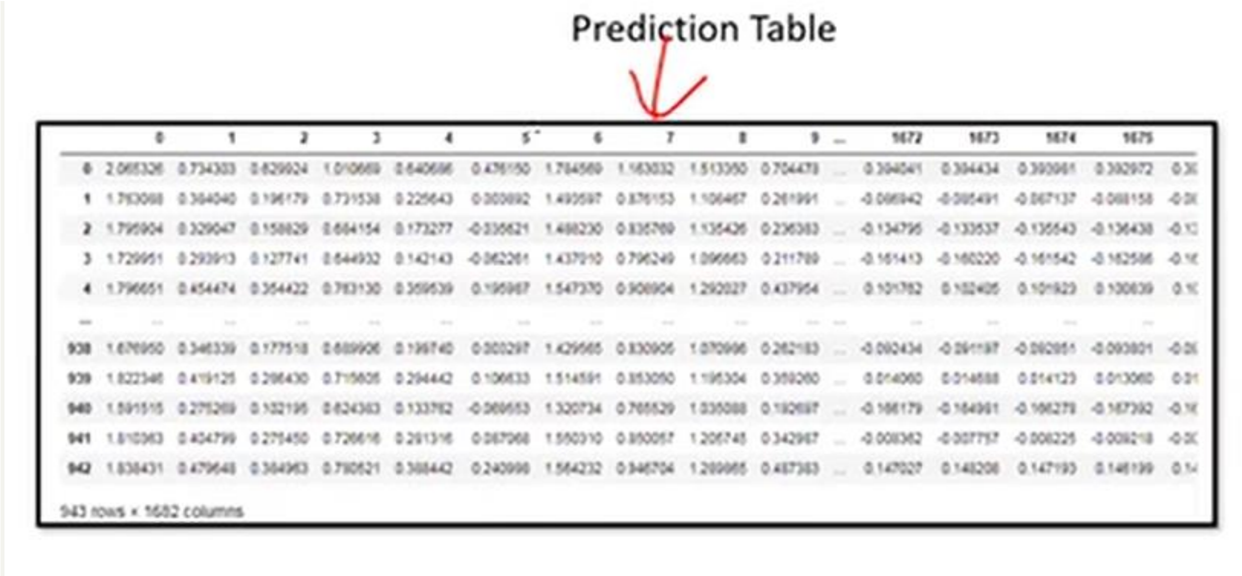
User similarity table (cosine formula)

Similarity Table(Cosine)

	0	1	2	3	4	5	6	7	8	9	...	933	934	935	936	
0	2.220446e-15	0.833069	0.952540	0.935642	6.215248e-01	0.569761	0.559633	0.680928	0.921862	0.623456	...	0.630473	0.880518	0.725124	0.810295	0.80
1	8.330690e-01	0.000000	0.889409	0.821879	9.270210e-01	0.754157	0.892672	0.896656	0.838952	0.840138	...	0.843014	0.692058	0.641211	0.575954	0.68
2	9.525405e-01	0.889409	0.000000	0.655849	9.787555e-01	0.927585	0.933863	0.916940	0.938960	0.934849	...	0.968125	0.957247	0.836171	0.930962	0.87
3	9.356422e-01	0.821879	0.655849	0.000000	9.681958e-01	0.931956	0.908770	0.811940	0.898716	0.939141	...	0.947893	0.963216	0.866885	0.806529	0.85
4	6.215248e-01	0.927021	0.978755	0.968196	1.110223e-16	0.762714	0.626400	0.751070	0.943153	0.798573	...	0.661206	0.919420	0.905076	0.920221	0.85
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
938	8.819047e-01	0.771417	0.973729	0.969862	9.285415e-01	0.888148	0.892973	0.904102	0.960148	0.928540	...	0.933961	0.568846	0.741979	0.773551	0.56
939	6.859280e-01	0.773210	0.838110	0.803142	7.600453e-01	0.647551	0.670075	0.753117	0.879505	0.657039	...	0.672847	0.892976	0.812464	0.818683	0.82
940	8.513831e-01	0.838515	0.898757	0.847959	8.604049e-01	0.855554	0.940007	0.853855	0.856755	0.909695	...	0.953048	0.796699	0.711682	0.765789	0.68
941	8.204921e-01	0.827732	0.866584	0.829914	8.475026e-01	0.682672	0.717997	0.824678	0.907503	0.787670	...	0.773560	0.926487	0.910412	0.870446	0.90
942	6.018253e-01	0.894202	0.973444	0.941248	6.860592e-01	0.723958	0.605636	0.700191	0.924383	0.778140	...	0.736209	0.789237	0.856747	0.922207	0.79

943 rows × 943 columns

Now find the prediction – If userid 2 can see the movie_id 2 what rating can give? So compare with (similar user(table)) going to predict what ratings will give. Predict for all user.(now no null values)


Prediction Table

	0	1	2	3	4	5	6	7	8	9	...	1672	1673	1674	1675	
0	2.065326	0.734303	0.629924	1.010669	0.640696	0.476150	1.784569	1.163032	1.513350	0.704473	...	0.394041	0.394434	0.390961	0.392972	0.3
1	1.763068	0.384040	0.196179	0.731538	0.225643	0.300892	1.493597	0.376153	1.106467	0.261991	...	-0.096942	-0.095491	-0.097137	-0.098158	-0.0
2	1.795904	0.329047	0.158829	0.684154	0.173277	-0.336621	1.488230	0.335769	1.135426	0.236383	...	-0.134795	-0.133537	-0.135543	-0.136438	-0.1
3	1.729951	0.293913	0.127741	0.644932	0.142143	-0.362261	1.437010	0.796249	1.096663	0.211789	...	-0.161413	-0.160220	-0.161542	-0.162586	-0.1
4	1.796651	0.454474	0.354422	0.763130	0.359539	0.195987	1.547370	0.906904	1.292027	0.437964	...	0.101782	0.102405	0.101923	0.100639	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
938	1.676950	0.346339	0.177518	0.689906	0.196740	0.300297	1.429565	0.330905	1.070996	0.262183	...	-0.092434	-0.091197	-0.092951	-0.093801	-0.0
939	1.822346	0.419125	0.296430	0.719605	0.294442	0.106633	1.514591	0.353050	1.195304	0.359260	...	0.014060	0.014688	0.014123	0.013060	0.0
940	1.591515	0.275269	0.102195	0.624363	0.133762	-0.369653	1.320734	0.765529	1.035088	0.192697	...	-0.166179	-0.164961	-0.166279	-0.167392	-0.1
941	1.810363	0.404799	0.279450	0.726616	0.291316	0.067968	1.550310	0.850057	1.206745	0.342967	...	-0.008362	-0.007757	-0.008225	-0.009218	-0.0
942	1.838431	0.479648	0.384963	0.790521	0.398442	0.240998	1.564232	0.946704	1.289965	0.487383	...	0.147027	0.148208	0.147193	0.146199	0.1

943 rows × 1682 columns

```
def predict(ratings, similarity, type='user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        #We use np.newaxis so that mean_user_rating has same format as ratings
        mean_user_rating_array = np.array(mean_user_rating)  # Convert to numpy array
        ratings_diff = (ratings - mean_user_rating_array[:, np.newaxis])
        pred = mean_user_rating_array[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

```
mean_user_rating = ratings.mean(axis=1)
```

Calculate the Mean rating for each user in the (ratings matrix) is used to calculate the mean rating for each user in a user-item rating matrix.

.mean(axis=1): This applies the mean function along the rows of the ratings matrix (i.e., along the second axis). It calculates the mean rating for each user by averaging the ratings across all the items they have rated.

```
mean_user_rating_array = np.array(mean_user_rating)  # Convert to numpy array
```

Np.array->numpy array converted into a list

```
ratings_diff = (ratings - mean_user_rating_array[:, np.newaxis])
```

:→rows will take same

Np.newaxis→ mean user rating rows can take it as column(1Darray into 2Darray)

```
pred = mean_user_rating_array[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T
```

similarity.dot(ratings_diff)

similarity: This is a 2D NumPy array or pandas DataFrame that represents the similarity between users. Each row corresponds to a user, and each column corresponds to another user. The values in the matrix represent the similarity between each pair of users.

```
user_sim_table
```

	0	1	2	3	4	5	6	7	8	9	...	933	934	935	936
0	1.332268e-15	0.833069	0.952540	0.935642	0.621525	0.569761	0.559633	0.680928	0.921862	0.623456	...	0.630473	0.880518	0.725124	0.810295
1	8.330690e-01	0.000000	0.889409	0.821879	0.927021	0.754157	0.892672	0.896656	0.838952	0.840138	...	0.843014	0.692058	0.641211	0.575954
2	9.525405e-01	0.889409	0.000000	0.655849	0.978755	0.927585	0.933863	0.916940	0.938960	0.934849	...	0.968125	0.957247	0.836171	0.930962
3	9.356422e-01	0.821879	0.655849	0.000000	0.968196	0.931956	0.908770	0.811940	0.898716	0.939141	...	0.947893	0.963216	0.866885	0.806529
4	6.215248e-01	0.927021	0.978755	0.968196	0.000000	0.762714	0.626400	0.751070	0.943153	0.798573	...	0.661206	0.919420	0.905076	0.920221
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
938	8.819047e-01	0.771417	0.973729	0.969862	0.928541	0.888148	0.892973	0.904102	0.960148	0.928540	...	0.933961	0.568846	0.741979	0.773551
939	6.859280e-01	0.773210	0.838110	0.803142	0.760045	0.647551	0.670075	0.753117	0.879505	0.657039	...	0.672847	0.892976	0.812464	0.818683
940	8.513831e-01	0.838515	0.898757	0.847959	0.860405	0.855554	0.940007	0.853855	0.856755	0.909695	...	0.953048	0.796699	0.711682	0.765789

ratings_diff

ratings_diff: This is a 2D NumPy array or pandas DataFrame that represents the ratings differences between users. Each row corresponds to a user, and each column corresponds to an item (e.g., a movie, song, or product). The values in the matrix represent the difference between each user's ratings and the mean user rating.

```
ratings_diff
```

	user_id	movie_id	rating	unix_timestamp
0	-2.203127e+08	-2.203126e+08	-2.203128e+08	6.609381e+08
1	-2.229294e+08	-2.229293e+08	-2.229296e+08	6.687882e+08
2	-2.197219e+08	-2.197215e+08	-2.197219e+08	6.591652e+08
3	-2.201516e+08	-2.201518e+08	-2.201518e+08	6.604551e+08
4	-2.215994e+08	-2.215992e+08	-2.215995e+08	6.647981e+08
...	...	...	...	...
99995	-2.200433e+08	-2.200437e+08	-2.200442e+08	6.601312e+08
99996	-2.199484e+08	-2.199489e+08	-2.199491e+08	6.598464e+08
99997	-2.186990e+08	-2.186982e+08	-2.186993e+08	6.560965e+08
99998	-2.205998e+08	-2.205996e+08	-2.205998e+08	6.617993e+08
99999	-2.199899e+08	-2.199897e+08	-2.199899e+08	6.599696e+08

100000 rows × 4 columns

.dot(): This method performs a **matrix multiplication between the similarity matrix and the ratings_diff matrix**. It calculates the dot product of each row in the similarity matrix with the corresponding row in the ratings_diff matrix.

result: This is the output of the matrix multiplication, which is a new matrix that represents the **weighted sum of the ratings** differences for each user or item based on the similarity between them.

Now having user prediction table, given the ratings for not seen movies using of AI intelligence.

```
user_pred=pd.DataFrame(user_prediction)  #user prediction-->datamatrix X user similarity (table as dataframe)
```

```
user_pred   #943 users 1682 movie ku predict pana ratings (pakadha  ovie kum predict pana ratings)
```

	0	1	2	3	4	5	6	7	8	9	...	1672
0	2.065326	0.734303	0.629924	1.010669	0.640686	0.476150	1.784569	1.163032	1.513350	0.704478	...	0.394041
1	1.763088	0.384040	0.196179	0.731538	0.225643	0.003892	1.493597	0.876153	1.108467	0.261991	...	-0.086942
2	1.795904	0.329047	0.158829	0.684154	0.173277	-0.035621	1.488230	0.835769	1.135426	0.236383	...	-0.134795
3	1.729951	0.293913	0.127741	0.644932	0.142143	-0.062261	1.437010	0.796249	1.096663	0.211789	...	-0.161413
4	1.796651	0.454474	0.354422	0.763130	0.359539	0.195987	1.547370	0.908904	1.292027	0.437954	...	0.101762
...	...	...	...	...	...	...	...	...	...	...	...	...
938	1.676950	0.346339	0.177518	0.689906	0.199740	0.003297	1.429565	0.830905	1.070986	0.262183	...	-0.092434
939	1.822346	0.419125	0.286430	0.715605	0.294442	0.106633	1.514591	0.853050	1.195304	0.359260	...	0.014060
940	1.591515	0.275269	0.102195	0.624383	0.133762	-0.069553	1.320734	0.765529	1.035088	0.192697	...	-0.166179
941	1.810363	0.404799	0.275450	0.726616	0.281316	0.087068	1.550310	0.850057	1.205745	0.342987	...	-0.008362
942	1.838431	0.479648	0.384963	0.780521	0.388442	0.240998	1.564232	0.946704	1.289865	0.487383	...	0.147027

943 rows × 1682 columns

Next, Find from user_sim_table, take particular user(user=34) find the similar user and suggest or recommend the movies.