In [81]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import collections
import seaborn as sns
%matplotlib inline
```

In [82]:

```
movies_df = pd.read_csv("movies.csv")
movies_df.head(3)
```

Out[82]:

	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

In [83]:

```
# inspecting various genres
genres = movies_df['genres']
genres.head()
```

Out[83]:

```
0    Adventure|Animation|Children|Comedy|Fantasy
1                    Adventure|Children|Fantasy
2                                Comedy|Romance
3                          Comedy|Drama|Romance
4                                        Comedy
Name: genres, dtype: object
```

In [84]:

```
genre_list = ""
for index,row in movies_df.iterrows():
        genre_list += row.genres + "|"
#split the string into a list of values
genre_list_split = genre_list.split('|')
#de-duplicate values
new_list = list(set(genre_list_split))
#remove the value that is blank
new_list.remove('')
#inspect list of genres
new_list
```

Out[84]:

```
['Drama',
 'Action',
 'Adventure',
 'Western',
 'War',
 'Romance',
 'Fantasy',
 'Film-Noir',
 'Animation',
 'Comedy',
 'Mystery',
 '(no genres listed)',
 'Thriller',
 'Sci-Fi',
 'Horror',
 'Documentary',
 'Children',
 'Musical',
 'IMAX',
 'Crime']
```

In [85]:

```
#Enriching the movies dataset by adding the various genres columns.
movies_with_genres = movies_df.copy()

for genre in new_list :
    movies_with_genres[genre] = movies_with_genres.apply(lambda _:int(genre in _.genres), axis = 1)
```

In [86]:

```
movies_with_genres.head()
```

Out[86]:

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

5 rows × 23 columns

In [87]:

```
genre_df = pd.DataFrame(movies_df['genres'].str.split('|').tolist(), index = movies_df['movieId']).
stack()
genre_df = genre_df.reset_index([0, 'movieId'])
genre_df.columns = ['movieId', 'genres']
genre_df.head(5)
```

Out[87]:

	movieId	genres
0	1	Adventure
1	1	Animation
2	1	Children
3	1	Comedy
4	1	Fantasy

In [88]:

```
ratings_data = pd.read_csv('ratings.csv')
ratings_data.head(3)
```

Out[88]:

userId	movieId	rating	timestamp

	userId	movieId	rating	timestamp
~~0~~	~~1~~	~~2~~	~~3.5~~	~~1112486027~~
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819

In [89]:

```
ratings_df = ratings_data.iloc[:1000000,:]
```

In [90]:

```
R_df = ratings_df.pivot(index = 'userId', columns ='movieId', values = 'rating').fillna(0)
R_df.head()
```

Out[90]:

movieId	1	2	3	4	5	6	7	8	9	10	...	129350	129354	129428	129707	130052	130073	130219	130462	130490
userId																				
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
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
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
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
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

5 rows × 13950 columns

In [91]:

```
R = R_df.as_matrix()

user_ratings_mean = np.mean(R, axis = 1)

R_demeaned = R - user_ratings_mean.reshape(-1, 1)
print(R_demeaned)
```

```
C:\Users\Mahesh\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: Method
.as_matrix will be removed in a future version. Use .values instead.
  """Entry point for launching an IPython kernel.
```

```
[[-0.04695341  3.45304659 -0.04695341 ... -0.04695341 -0.04695341
  -0.04695341]
 [-0.01749104 -0.01749104  3.98250896 ... -0.01749104 -0.01749104
  -0.01749104]
 [ 3.94473118 -0.05526882 -0.05526882 ... -0.05526882 -0.05526882
  -0.05526882]
 ...
 [ 3.98089606 -0.01910394 -0.01910394 ... -0.01910394 -0.01910394
  -0.01910394]
 [-0.00637993 -0.00637993 -0.00637993 ... -0.00637993 -0.00637993
  -0.00637993]
 [ 3.95189964  2.95189964  2.95189964 ... -0.04810036 -0.04810036
  -0.04810036]]
```

In [92]:

```
from scipy.sparse.linalg import svds
U, sigma, Vt = svds(R_demeaned, k = 40)
```

In [93]:

```
sigma = np.diag(sigma)
```

In [94]:

```
a= np.dot(np.dot(U, sigma), Vt)
```

In [95]:

```
a
```

Out[95]:

```
array([[ 2.41509819e-01,  5.78733093e-01, -1.09100617e-01, ...,
        -4.36291925e-02, -3.55178729e-02, -4.45711236e-02],
       [ 1.02445223e+00,  1.61417266e-01,  2.56404718e-01, ...,
        -1.57224318e-02, -1.15934980e-02, -2.03722657e-02],
       [ 1.56739099e+00,  6.46318865e-01, -2.54342028e-01, ...,
        -4.77301009e-02, -4.30731942e-02, -5.61806399e-02],
       ...,
       [ 2.47055295e+00,  1.67294205e-01, -8.62565327e-03, ...,
        -1.87575383e-02, -2.58865345e-02, -1.60058753e-02],
       [ 5.13571138e-01, -8.58154449e-02, -4.10802131e-01, ...,
        -1.19769189e-03, -1.08907895e-02, -3.90409010e-04],
       [ 4.01643834e+00,  2.35048783e+00,  1.85342661e+00, ...,
        -5.39784336e-02, -4.48139004e-02, -5.94621263e-02]])
```

In [96]:

```
 user_ratings_mean.reshape(-1, 1)
```

Out[96]:

```
array([[0.04695341],
       [0.01749104],
       [0.05526882],
       ...,
       [0.01910394],
       [0.00637993],
       [0.04810036]])
```

In [97]:

```
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.reshape(-1, 1)
print(all_user_predicted_ratings)
```

```
[[ 2.88463224e-01  6.25686498e-01 -6.21472122e-02 ...  3.32421252e-03
   1.14355321e-02  2.38228146e-03]
 [ 1.04194327e+00  1.78908305e-01  2.73895757e-01 ...  1.76860765e-03
   5.89754143e-03 -2.88122632e-03]
 [ 1.62265981e+00  7.01587683e-01 -1.99073211e-01 ...  7.53871632e-03
   1.21956231e-02 -9.11822661e-04]
 ...
 [ 2.48965689e+00  1.86398148e-01  1.04782894e-02 ...  3.46404396e-04
  -6.78259188e-03  3.09806734e-03]
 [ 5.19951067e-01 -7.94355166e-02 -4.04422203e-01 ...  5.18223642e-03
  -4.51086121e-03  5.98951931e-03]
 [ 4.06453870e+00  2.39858819e+00  1.90152697e+00 ... -5.87807521e-03
   3.28645805e-03 -1.13617678e-02]]
```

In [98]:

```
preds_df = pd.DataFrame(all_user_predicted_ratings, columns = R_df.columns)
preds_df.head()
```

Out[98]:

movieId	1	2	3	4	5	6	7	8	9	10	...	129350	129354
0	0.288463	0.625686	-0.062147	-0.100269	-0.293077	0.529618	-0.335913	-0.046073	-0.133599	0.009705	...	0.011139	0.000930
1	1.041943	0.178908	0.273896	0.081557	0.193555	0.152205	0.397419	0.006470	0.062602	-0.065872	...	-0.001868	-0.007790
2	1.622660	0.701588	-0.199073	-0.034604	-0.151640	0.498565	-0.091586	0.029022	-0.105952	0.028441	...	0.003473	0.019103

movieId	1	2	3	4	5	6	7	8	9	10	...	129350	129354	
3	0.499641	0.626433	0.394443	0.028003	0.288118	0.670830	0.061892	0.086604	0.181397	0.856338	...	0.001720	0.061785	0
4	2.378414	1.211986	1.289963	0.099752	1.211267	0.823061	1.442084	0.136348	0.224241	1.585773	...	-0.000469	0.003078	0

5 rows × 13950 columns

In [99]:

```
def recommend_movies(predictions_df, userId, movies_df, original_ratings_df, num_recommendations=5)
:

    # Get and sort the user's predictions
    user_row_number = userId - 1  # UserID starts at 1, not 0
    sorted_user_predictions = predictions_df.iloc[user_row_number].sort_values(ascending=False) # U
serID starts at 1

    # Get the user's data and merge in the movie information.
    user_data = original_ratings_df[original_ratings_df.userId == (userId)]
    user_full = (user_data.merge(movies_df, how = 'left', left_on = 'movieId', right_on = 'movieId'
).
                     sort_values(['rating'], ascending=False)
                 )

    print('User {0} has already rated {1} movies.'.format(userId, user_full.shape[0]))
    print('Recommending highest {0} predicted ratings movies not already
rated.'.format(num_recommendations))
    # Recommend the highest predicted rating movies that the user hasn't seen yet.
    recommendations = (movies_df[~movies_df['movieId'].isin(user_full['movieId'])].
         merge(pd.DataFrame(sorted_user_predictions).reset_index(), how = 'left',
               left_on = 'movieId',
               right_on = 'movieId').
         rename(columns = {user_row_number: 'Predictions'}).
         sort_values('Predictions', ascending = False).
                      iloc[:num_recommendations, :-1]
                     )

    return user_full, recommendations
```

In [100]:

```
already_rated, predictions = recommend_movies(preds_df, 2, movies_df, ratings_df, 10)
```

```
User 2 has already rated 61 movies.
Recommending highest 10 predicted ratings movies not already rated.
```

In [101]:

```
already_rated.head(10)
```

Out[101]:

	userId	movieId	rating	timestamp	title	genres
60	2	3959	5.0	974820659	Time Machine, The (1960)	Action|Adventure|Sci-Fi
15	2	1196	5.0	974821014	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi
33	2	1974	5.0	974820598	Friday the 13th (1980)	Horror|Mystery|Thriller
45	2	3450	5.0	974820846	Grumpy Old Men (1993)	Comedy
1	2	62	5.0	974820598	Mr. Holland's Opus (1995)	Drama
46	2	3513	5.0	974820659	Rules of Engagement (2000)	Drama|Thriller
26	2	1748	5.0	974821014	Dark City (1998)	Adventure|Film-Noir|Sci-Fi|Thriller
23	2	1544	5.0	974820943	Lost World: Jurassic Park, The (1997)	Action|Adventure|Sci-Fi|Thriller
22	2	1356	5.0	974820598	Star Trek: First Contact (1996)	Action|Adventure|Sci-Fi|Thriller
21	2	1327	5.0	974820846	Amityville Horror, The (1979)	Drama|Horror|Mystery|Thriller

In [102]:

```
predictions
```

Out[102]:

	movieId	title	genres
1159	1200	Aliens (1986)	Action|Adventure|Horror|Sci-Fi
1194	1240	Terminator, The (1984)	Action|Sci-Fi|Thriller
2448	2571	Matrix, The (1999)	Action|Sci-Fi|Thriller
30	32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
1909	2028	Saving Private Ryan (1998)	Action|Drama|War
2505	2628	Star Wars: Episode I - The Phantom Menace (1999)	Action|Adventure|Sci-Fi
1157	1198	Raiders of the Lost Ark (Indiana Jones and the...	Action|Adventure
345	356	Forrest Gump (1994)	Comedy|Drama|Romance|War
1061	1097	E.T. the Extra-Terrestrial (1982)	Children|Drama|Sci-Fi
756	780	Independence Day (a.k.a. ID4) (1996)	Action|Adventure|Sci-Fi|Thriller

In [103]:

```
#Just taking the required columns
ratings = ratings_data[['userId', 'movieId','rating']]
```

In [104]:

```
ratings.shape
```

Out[104]:

```
(20000263, 3)
```

In [105]:

```
ratings = ratings.iloc[:1000000,:]
```

In [106]:

```
#get ordered list of movieIds
item_indices = pd.DataFrame(sorted(list(set(ratings['movieId']))),columns=['movieId'])
#add in data frame index value to data frame
item_indices['movie_index']=item_indices.index
#inspect data frame
item_indices.head()
```

Out[106]:

	movieId	movie_index
0	1	0
1	2	1
2	3	2
3	4	3
4	5	4

In [107]:

```
#get ordered list of movieIds
user_indices = pd.DataFrame(sorted(list(set(ratings['userId']))),columns=['userId'])
#add in data frame index value to data frame
user_indices['user_index']=user_indices.index
#inspect data frame
user_indices.head()
user_indices.head()
```

Out[107]:

	userId	user_index
0	1	0
1	2	1
2	3	2
3	4	3
4	5	4

In [108]:

```
#join the movie indices
df_with_index = pd.merge(ratings,item_indices,on='movieId')
#join the user indices
df_with_index=pd.merge(df_with_index,user_indices,on='userId')
#inspec the data frame
df_with_index.head()
```

Out[108]:

	userId	movieId	rating	movie_index	user_index
0	1	2	3.5	1	0
1	1	29	3.5	28	0
2	1	32	3.5	31	0
3	1	47	3.5	46	0
4	1	50	3.5	49	0

In [109]:

```
#import train_test_split module
from sklearn.model_selection import train_test_split
#take 80% as the training set and 20% as the test set
df_train, df_test= train_test_split(df_with_index,test_size=0.2)
print(len(df_train))
print(len(df_test))
```

```
800000
200000
```

In [110]:

```
df_train.head()
```

Out[110]:

	userId	movieId	rating	movie_index	user_index
84976	1568	7022	3.5	6571	1567
801399	5133	8228	3.5	7267	5132
400699	1667	2936	4.0	2774	1666
575717	3821	316	3.5	310	3820
55808	982	6858	3.5	6425	981

In [111]:

```
df_test.head()
```

Out[111]:

	userId	movieId	rating	movie_index	user_index
177220	3251	38038	3.0	9039	3250
361392	6366	3105	4.0	2935	6365
639998	5726	3683	5.0	3467	5725
847599	4484	1284	4.0	1227	4483
548242	2918	1939	4.0	1797	2917

In [112]:

```
n_users = ratings.userId.unique().shape[0]
n_items = ratings.movieId.unique().shape[0]
print(n_users)
print(n_items)
```

```
6743
13950
```

In [113]:

```
#Create two user-item matrices, one for training and another for testing
train_data_matrix = np.zeros((n_users, n_items))
    #for every line in the data
for line in df_train.itertuples():
    #set the value in the column and row to
    #line[1] is userId, line[2] is movieId and line[3] is rating, line[4] is movie_index and
line[5] is user_index
    train_data_matrix[line[5], line[4]] = line[3]
train_data_matrix.shape
```

Out[113]:

```
(6743, 13950)
```

In [114]:

```
#Create two user-item matrices, one for training and another for testing
test_data_matrix = np.zeros((n_users, n_items))
    #for every line in the data
for line in df_test[:1].itertuples():
    #set the value in the column and row to
    #line[1] is userId, line[2] is movieId and line[3] is rating, line[4] is movie_index and
line[5] is user_index
    #print(line[2])
    test_data_matrix[line[5], line[4]] = line[3]
    #train_data_matrix[line['movieId'], line['userId']] = line['rating']
test_data_matrix.shape
```

Out[114]:

```
(6743, 13950)
```

In [115]:

```
pd.DataFrame(train_data_matrix).head()
```

Out[115]:

	0	1	2	3	4	5	6	7	8	9	...	13940	13941	13942	13943	13944	13945	13946	13947	13948	13949
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	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
2	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
3	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
4	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 × 13950 columns

5 rows × 13950 columns

In [116]:

```
df_train['rating'].max()
```

Out[116]:

```
5.0
```

In [117]:

```
from sklearn.metrics import mean_squared_error
from math import sqrt
def rmse(prediction, ground_truth):
    #select prediction values that are non-zero and flatten into 1 array
    prediction = prediction[ground_truth.nonzero()].flatten()
    #select test values that are non-zero and flatten into 1 array
    ground_truth = ground_truth[ground_truth.nonzero()].flatten()
    #return RMSE between values
    return sqrt(mean_squared_error(prediction, ground_truth))
```

In [118]:

```
#Calculate the rmse sscore of SVD using different values of k (latent features)
rmse_list = []
for i in [1,2,5,20,40,60,100,200]:
    #apply svd to the test data
    u,s,vt = svds(train_data_matrix,k=i)
    #get diagonal matrix
    s_diag_matrix=np.diag(s)
    #predict x with dot product of u s_diag and vt
    X_pred = np.dot(np.dot(u,s_diag_matrix),vt)
    #calculate rmse score of matrix factorisation predictions
    rmse_score = rmse(X_pred,test_data_matrix)

    rmse_list.append(rmse_score)
    print("Matrix Factorisation with " + str(i) +" latent features has a RMSE of " + str(rmse_score
))
```

```
Matrix Factorisation with 1 latent features has a RMSE of 2.4970794033324313
Matrix Factorisation with 2 latent features has a RMSE of 2.1618495099787776
Matrix Factorisation with 5 latent features has a RMSE of 1.8733869001830206
Matrix Factorisation with 20 latent features has a RMSE of 1.7853130663941765
Matrix Factorisation with 40 latent features has a RMSE of 2.6168758523072304
Matrix Factorisation with 60 latent features has a RMSE of 2.66013353235069
Matrix Factorisation with 100 latent features has a RMSE of 1.8426839656094136
Matrix Factorisation with 200 latent features has a RMSE of 1.918122414192567
```

In [119]:

```
#Convert predictions to a DataFrame
mf_pred = pd.DataFrame(X_pred)
mf_pred.head()
```

Out[119]:

	0	1	2	3	4	5	6	7	8	9	...	13940	13941	13942	
0	0.218255	1.313368	-0.246001	-0.023728	-0.155889	0.852599	-0.054993	-0.074503	0.020425	0.421483	...	0.0	0.0	-0.003133	0.0
1	0.449365	0.110443	0.883769	0.013483	0.309498	0.459796	0.258331	0.022881	0.112002	0.391264	...	0.0	0.0	0.027892	0.0
2	3.343079	0.131079	0.031574	-0.265863	0.368169	0.339018	0.326889	0.034546	-0.034879	-0.074379	...	0.0	0.0	-0.007011	0.0
3	-0.212770	0.607811	0.298388	0.008983	0.127687	1.495635	0.111081	0.082198	0.072506	2.220253	...	0.0	0.0	-0.010224	0.0
4	0.320289	1.860684	0.958116	-0.006943	1.120503	0.707049	1.060215	0.071907	0.276401	0.078626	...	0.0	0.0	0.009203	0.0

5 rows × 13950 columns

In [120]:

```
df_names = pd.merge(ratings,movies_df,on='movieId')
df_names.head()
```

Out[120]:

	userId	movieId	rating	title	genres
0	1	2	3.5	Jumanji (1995)	Adventure|Children|Fantasy
1	5	2	3.0	Jumanji (1995)	Adventure|Children|Fantasy
2	13	2	3.0	Jumanji (1995)	Adventure|Children|Fantasy
3	29	2	3.0	Jumanji (1995)	Adventure|Children|Fantasy
4	34	2	3.0	Jumanji (1995)	Adventure|Children|Fantasy

In [121]:

```
#choose a user ID
user_id = 2
#get movies rated by this user id
users_movies = df_names.loc[df_names["userId"]==user_id]
#print how many ratings user has made
print("User ID : " + str(user_id) + " has already rated " + str(len(users_movies)) + " movies")
#list movies that have been rated
users_movies
```

```
User ID : 2 has already rated 61 movies
```

Out[121]:

	userId	movieId	rating	title	genres
12071	2	260	5.0	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi
25092	2	541	5.0	Blade Runner (1982)	Action|Sci-Fi|Thriller
26619	2	589	5.0	Terminator 2: Judgment Day (1991)	Action|Sci-Fi
34185	2	924	5.0	2001: A Space Odyssey (1968)	Adventure|Drama|Sci-Fi
46020	2	1196	5.0	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi
53763	2	1214	5.0	Alien (1979)	Horror|Sci-Fi
61214	2	1249	5.0	Femme Nikita, La (Nikita) (1990)	Action|Crime|Romance|Thriller
62860	2	1259	5.0	Stand by Me (1986)	Adventure|Drama
81161	2	2291	2.0	Edward Scissorhands (1990)	Drama|Fantasy|Romance
125843	2	3	4.0	Grumpier Old Men (1995)	Comedy|Romance
126494	2	62	5.0	Mr. Holland's Opus (1995)	Drama
127485	2	70	5.0	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
128142	2	110	4.0	Braveheart (1995)	Action|Drama|War
130809	2	242	3.0	Farinelli: il castrato (1994)	Drama|Musical
130871	2	266	5.0	Legends of the Fall (1994)	Drama|Romance|War|Western
131640	2	469	3.0	House of the Spirits, The (1993)	Drama|Romance
131722	2	480	5.0	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller
134701	2	891	2.0	Halloween: The Curse of Michael Myers (Hallowe...	Horror|Thriller
134796	2	908	4.0	North by Northwest (1959)	Action|Adventure|Mystery|Romance|Thriller
135582	2	1121	3.0	Glory Daze (1995)	Drama
135591	2	1210	5.0	Star Wars: Episode VI - Return of the Jedi (1983)	Action|Adventure|Sci-Fi
137945	2	1270	5.0	Back to the Future (1985)	Adventure|Comedy|Sci-Fi
140030	2	1327	5.0	Amityville Horror, The (1979)	Drama|Horror|Mystery|Thriller
140180	2	1356	5.0	Star Trek: First Contact (1996)	Action|Adventure|Sci-Fi|Thriller

	userId	movieId	rating	title	genres
141153	2	1544	5.0	Lost World: Jurassic Park, The (1997)	Action|Adventure|Sci-Fi|Thriller
141890	2	1580	4.0	Men in Black (a.k.a. MIB) (1997)	Action|Comedy|Sci-Fi
143714	2	1673	4.0	Boogie Nights (1997)	Drama
144410	2	1748	5.0	Dark City (1998)	Adventure|Film-Noir|Sci-Fi|Thriller
144986	2	1965	3.0	Repo Man (1984)	Comedy|Sci-Fi
145317	2	1969	2.0	Nightmare on Elm Street 2: Freddy's Revenge, A...	Horror
...	...	...	...	...	...
145550	2	1971	2.0	Nightmare on Elm Street 4: The Dream Master, A...	Horror|Thriller
145652	2	1972	2.0	Nightmare on Elm Street 5: The Dream Child, A ...	Horror
145734	2	1973	3.0	Freddy's Dead: The Final Nightmare (Nightmare ...	Horror
145810	2	1974	5.0	Friday the 13th (1980)	Horror|Mystery|Thriller
146002	2	1986	2.0	Halloween 5: The Revenge of Michael Myers (1989)	Horror
146052	2	2454	4.0	Fly, The (1958)	Horror|Mystery|Sci-Fi
146219	2	2455	4.0	Fly, The (1986)	Drama|Horror|Sci-Fi|Thriller
146724	2	2791	2.0	Airplane! (1980)	Comedy
147627	2	2858	3.0	American Beauty (1999)	Comedy|Drama
149861	2	2948	5.0	From Russia with Love (1963)	Action|Adventure|Thriller
150209	2	2951	4.0	Fistful of Dollars, A (Per un pugno di dollari...	Action|Western
150570	2	3150	4.0	War Zone, The (1999)	Drama|Thriller
150589	2	3159	3.0	Fantasia 2000 (1999)	Animation|Children|Musical|IMAX
150809	2	3173	4.0	Any Given Sunday (1999)	Drama
151012	2	3450	5.0	Grumpy Old Men (1993)	Comedy
151300	2	3513	5.0	Rules of Engagement (2000)	Drama|Thriller
151416	2	3534	3.0	28 Days (2000)	Drama
151651	2	3555	4.0	U-571 (2000)	Action|Thriller|War
151993	2	3565	3.0	Where the Heart Is (2000)	Comedy|Drama
152087	2	3703	4.0	Road Warrior, The (Mad Max 2) (1981)	Action|Adventure|Sci-Fi
152502	2	3753	4.0	Patriot, The (2000)	Action|Drama|War
153137	2	3917	4.0	Hellraiser (1987)	Horror
153247	2	3918	3.0	Hellbound: Hellraiser II (1988)	Horror
153302	2	3923	4.0	Return of the Fly (1959)	Horror|Sci-Fi
153324	2	3926	4.0	Voyage to the Bottom of the Sea (1961)	Adventure|Sci-Fi
153353	2	3927	5.0	Fantastic Voyage (1966)	Adventure|Sci-Fi
153432	2	3928	5.0	Abbott and Costello Meet Frankenstein (1948)	Comedy|Horror
153479	2	3930	5.0	Creature from the Black Lagoon, The (1954)	Adventure|Horror|Sci-Fi
153529	2	3937	4.0	Runaway (1984)	Sci-Fi|Thriller
153556	2	3959	5.0	Time Machine, The (1960)	Action|Adventure|Sci-Fi

61 rows × 5 columns

In [122]:

```
user_index = df_train.loc[df_train["userId"]==user_id]['user_index'][:1].values[0]
#get movie ratings predicted for this user and sort by highest rating prediction
sorted_user_predictions = pd.DataFrame(mf_pred.iloc[user_index].sort_values(ascending=False))
#rename the columns
sorted_user_predictions.columns=['ratings']
#save the index values as movie id
sorted_user_predictions['movieId']=sorted_user_predictions.index
print("Top 10 predictions for User " + str(user_id))
#display the top 10 predictions for this user
pd.merge(sorted_user_predictions,movies_df, on = 'movieId')[:10]
```

```
Top 10 predictions for User 2
```

Out[122]:

	ratings	movieId	title	genres
0	5.117545	254	Jefferson in Paris (1995)	Drama
1	5.108069	470	House Party 3 (1994)	Comedy
2	4.726569	1155	Invitation, The (Zaproszenie) (1986)	Drama
3	4.713203	1142	Get Over It (1996)	Drama
4	4.687417	577	Andre (1994)	Adventure|Children|Drama
5	4.156220	531	Secret Garden, The (1993)	Children|Drama
6	4.120356	1159	Love in Bloom (1935)	Romance
7	3.901330	107	Muppet Treasure Island (1996)	Adventure|Children|Comedy|Musical
8	3.609280	1489	Cats Don't Dance (1997)	Animation|Children|Musical
9	2.787896	2697	My Son the Fanatic (1997)	Comedy|Drama|Romance

In [ ]: