Q1.

Rdd in use :

triplet_rdd : We have read "kaggle_visible_evaluation_triplets.txt" in it.
kaggle_songs_rdd: We have read "kaggle_songs.txt" file in it
rdd1 : In this rdd we made song name the key from triplet_rdd
rdd3 : Join of all songs and their id (kaggle_songs_rdd) with rdd1
rdd4: extracting userinfo, song id and count

Flow: We read both files, "kaggle_visible_evaluation_triplets.txt" and "kaggle_songs.txt". And Reordered them so that songname is the key. Then join both the rdd so that we have Songname as key and userinfo, count and song id as values. Then we extracting userinfo, song id and count.

Q2. Rdd in use :

triplet_map: Getting the main file "kaggle_visible_evaluation_triplets.txt" and making song name the key.
total_play_count : Rdd to capture the count of all songs
play_count_overall: Rdd to caluclate the count of each songs
play_count_overall_1 : Rdd to calculate each song rating

First we calculated the songs rating based on total play count of each song divided by total count of all songs played. Then we found the rating of each song to be very small. Then we apllied logic to get rating of each song by each user. And then calculating mean of the songs rating, to get each songs rating.

rdd5: it stores username, song id, int(count)
rdd6 : extract usernae, count from rdd5
rdd7 : extract username , sum(total count)
rdd8: to extract username , total count , song id , songid_count
rdd9 : to extract user info, song id, user's rating for the song
rdd10 : to get song id, rating
rdd11: to get the final rating per song

What we did in the second part of Q2: We first calulated the rating for each user by diving the song count by the user's total song count. Then we got each user's all songs rating. We grouped the data based on songs and summed up all the songs count. After that we divided each songs total overall count by total songs total count

Q3. Rdd in use :

user_liked_data: extract the particular user data
user_liked_songs: Extract the songs and their ratings, for songs that user liked
user_liked_songs_sorted: sort the above rdd in descending order, to get the most liked song of the user
fav_song: extract the most liked song
main_user_liked_songs : extract all the songs user liked
fav_song_rating : extract the favourite songs rating
similar_users: put a filter on rdd9 to get all the users for which the count of the "fav. song" is more than the count/rating that our user had given to the song.
recommended_users_full_data : get the full data of such users
recommended_users_full_data_sorted: sort the other user's data in descending order to get the songs that they liked/heard the most
recommended_users_new_songs : removed "our selected user's liked songs" from similar user's sorted data
recommendations : select top 5 songs from the generated data.

Picked a random user, extracted all the songs they liked, and used most liked/heard song to get other user who liked that song or heard it more than our selected user.
And then extracted the other songs and their rating from those user's data. And extracted top 5 making sure they are not in our selected user's list.

Q4.
1. Compute cosine similarity between all pairs of users.
2. Sort the similarity score and print the top-5 similar users.
3. If the top-5 user set has an user appearing more than once, ignore that pair and take the next best pair from the sorted list.
4. For a given user_id, identify the top-5 similar users and hence song recommendations from other user's list.

1.

```
## We have taken a reduced version of the data named in file : reduced_data
triplet_rdd_reduced = sc.textFile(r"reduced_data.txt")\
                    .map(lambda line: line.split("\t"))

#taking all the songs from the data
unique_interests = triplet_rdd_reduced.map(lambda x : x[1]).collect()

#print(unique_interests)
## Taking all the unique songs from the data
unique_interests_arr = np.array(unique_interests)
unique_interests_df_distict = np.unique(unique_interests_arr)

"""given a list of interests, produce a vector whose i-th element is 1
   if unique_interests[i] is in the list, 0 otherwise"""
```

We have added the comments to explain the code, in the Jupyter Notebook itself for Question 4