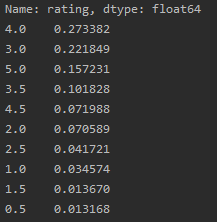
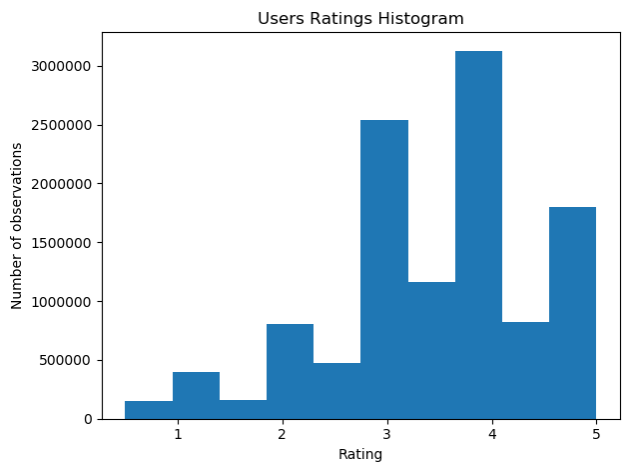
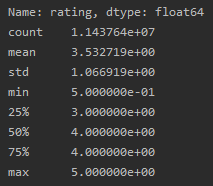
**Moran Sorka**

**Data Scientist – Home Test**

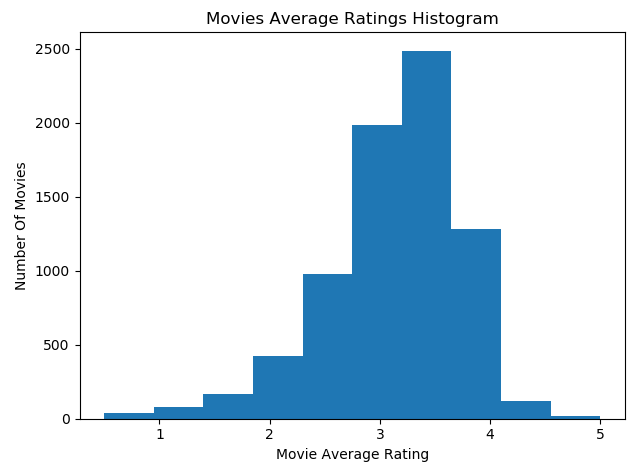
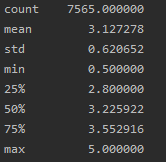
## Data Exploration:

The dataset contains 7,565 movies rated by 265,917 users.

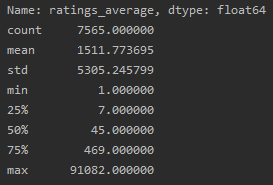
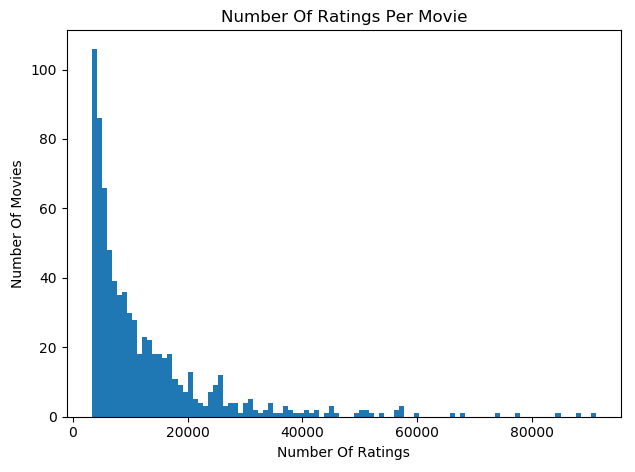
1. Histogram of users ratings:
   * Possible rating values: {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}
   * The average rating is 3.53, the mode is 4 (27.4%)
   * 75% of user’s ratings it integers (1, 2, 3, 4, 5)
   * 15% of user’s ratings is 5

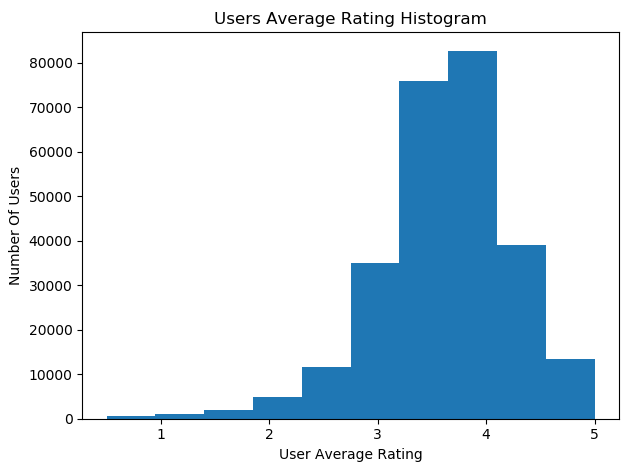
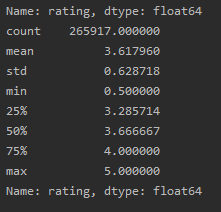
1. Histogram of movies average ratings:
   * The average rating per movie is 3.12 with std of 0.62
   * Only 5% of the movies have average rating greater than 4



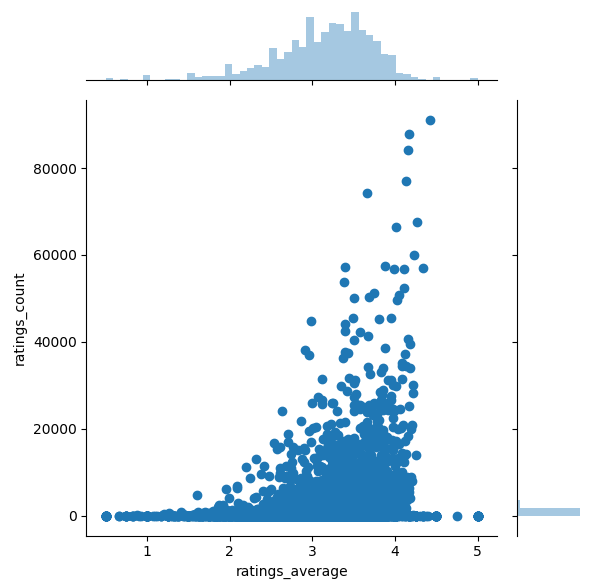
1. Histogram of number of ratings per movie:
   * Right skewed distribution
   * The median is 45 rating users per movie
   * 75% of the movies were ranked by less than 469 users

1. Plotting histogram of users average rating:
   * The average rating per user is: 3.6 with std of 0.63
   * Only 5% of the movies have average rating greater than 4

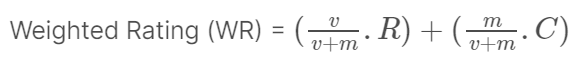


1. Relationship between ratings average and number of rating users:



## Q.1:

I choose to use IMDB’s weighted rating because this formula provides a Bayesian estimation which considers the number of ratings each movie has received, minimum ratings required and the mean ratings for all movies. Thus, this metric can use as an appropriate benchmark for this task.



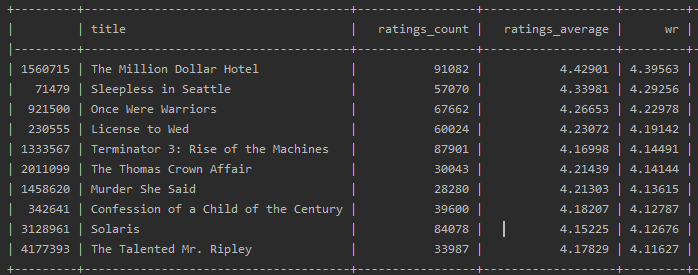
R = average rating for the movie

v = number of ratings for the movie

m = minimum ratings required to be listed in the Top Rated list

C = the mean ratings across the whole report

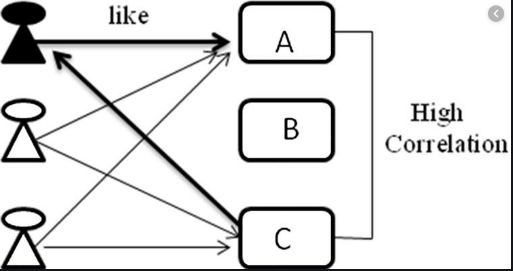
Top 10 movies:



## Q.2:

Splitting data into test and train sets

## Q.3:

I choose to implement Item-based collaborative filtering models (Model 1). Those models identify similar movies based on users’ previous ratings and predict the user’s rating for unrated movies.

I used grid search for tuning the models parameters.

* KNN-With-Means – clustering based algorithm
* SVD - Matrix Factorization based algorithm

.

* I compared the models using RMSE
* KNN is unsupervised learning model (pre-computed), we can limit the number of neighbors and make it more scalable. Many parameters to tune, sensitivity to noise.
* SVD is better dealing with scalability and sparsity data.
* In real life problems the recommendations must be “up to date” and response time must be very fast (trade-off). Evaluating the recommendation model prior to launch can be out-of-date.
* CTR, clicks for BPR, retention rate

