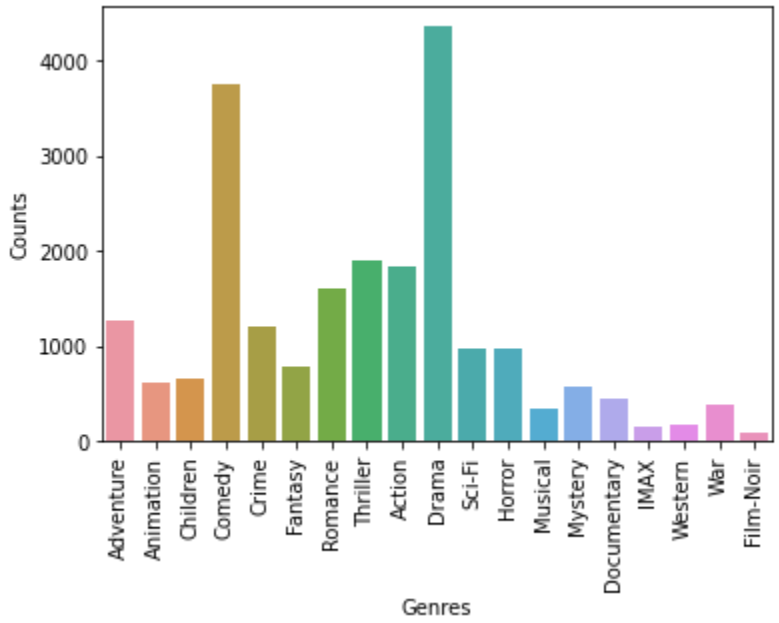
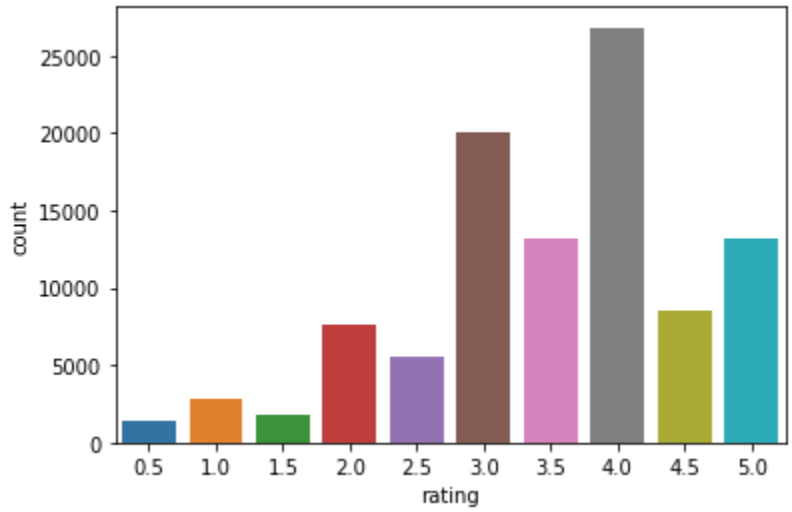
Q1]
Frequency of categorical features: movies.csv:
Categorical Attribute:Genres
Different genres count:19



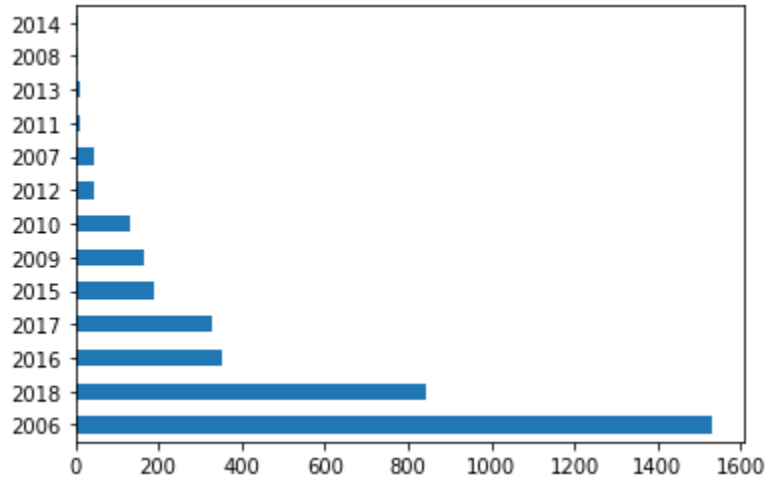
Frequently occuring genre is Drama with 4361 count value.
Mostly released movies are of type Drama least released movie is of type Film-Noir.

Genre had (no listed genre) in few rows so we deleted those many rows considering them Nan
Frequency of rows removed:34

Categorical Atribute in ratings.csv is rating.
Below is graph of rating and its count.



Frequency of giving 4.0 rating is maximum.
Categorical feature in tag.csv is timestamp.



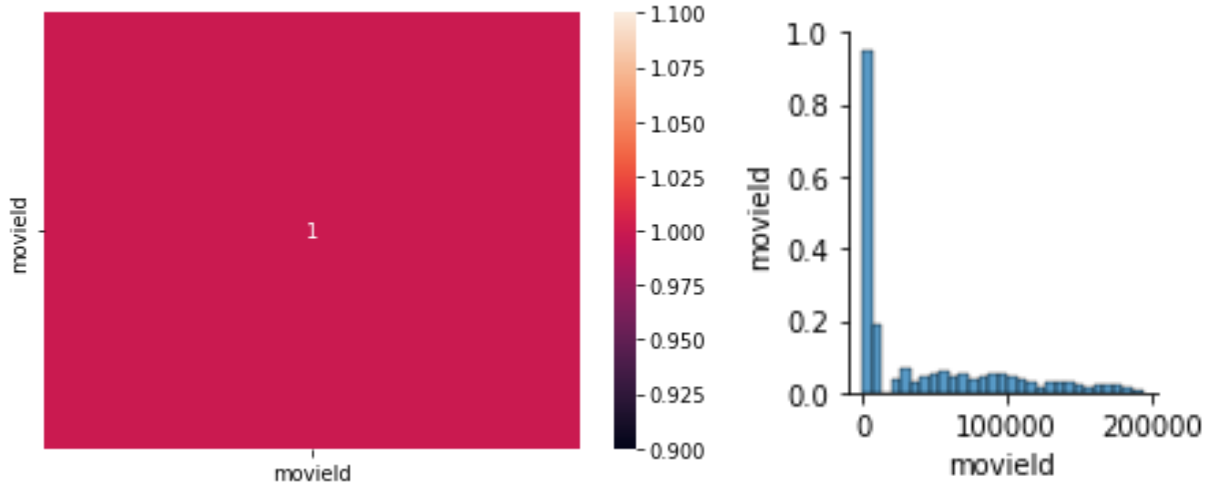
Maximum movies are tagged in 2006 around 1500 tags in 2006.

There is no Categorical attribute in links.csv.

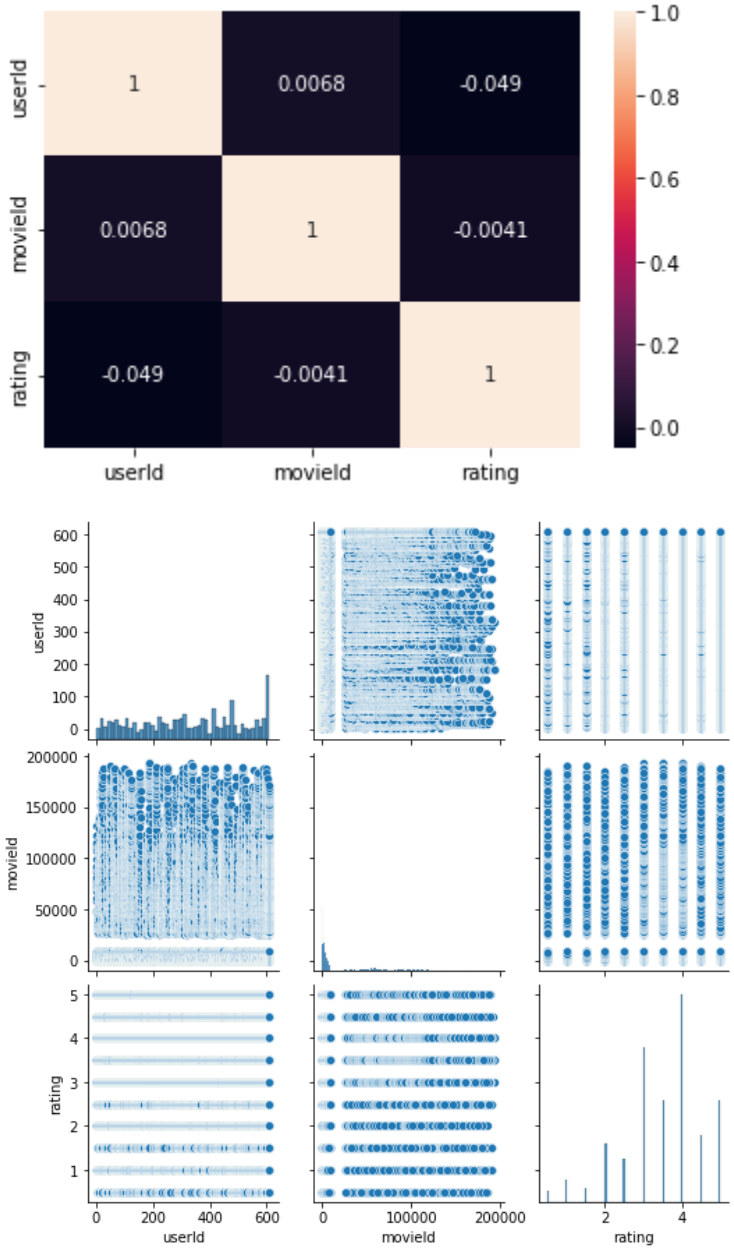
NaN values for every column in movies.csv dataset
movieId : 0 title : 0 genres : 0

Nan values from rating.csv dataset userId : 0
movieId : 0 rating : 0 timestamp : 0
Nan values from tags.csv dataset userId : 0
movieId : 0 tag : 0 timestamp : 0

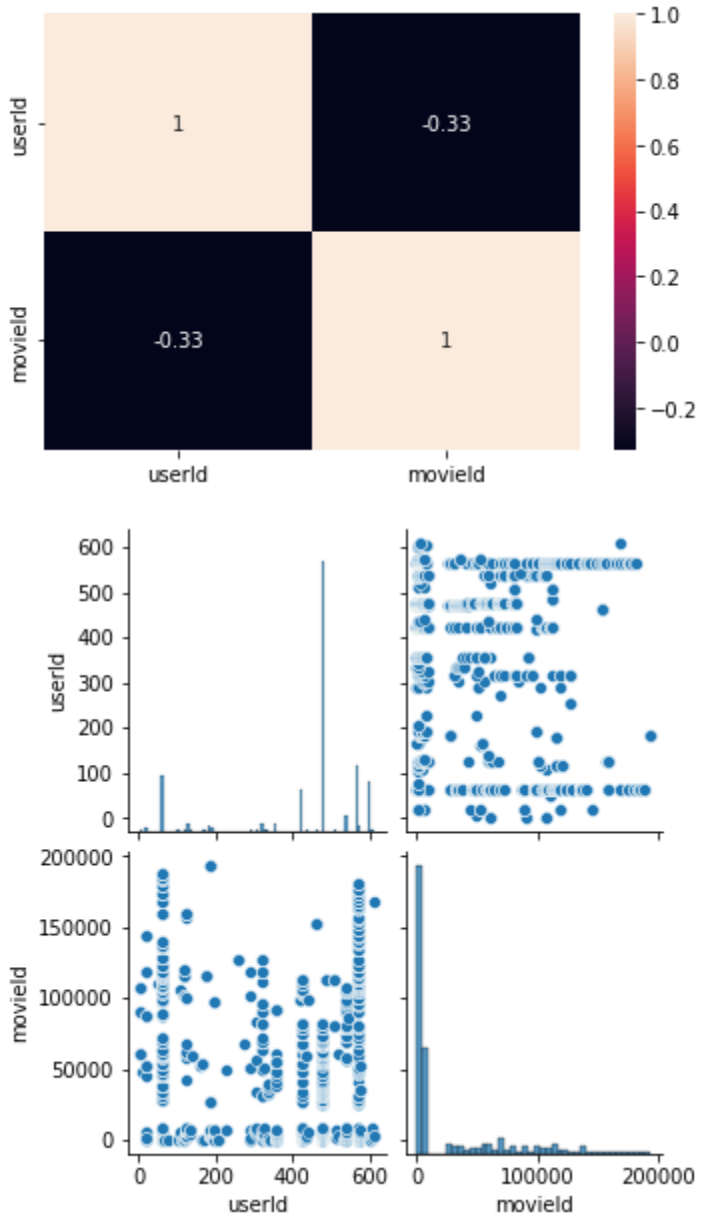
Nan values from links.csv dataset movieId : 0
imdbId : 0 tmdbId : 8 Correlation between
features:
Movie dataset



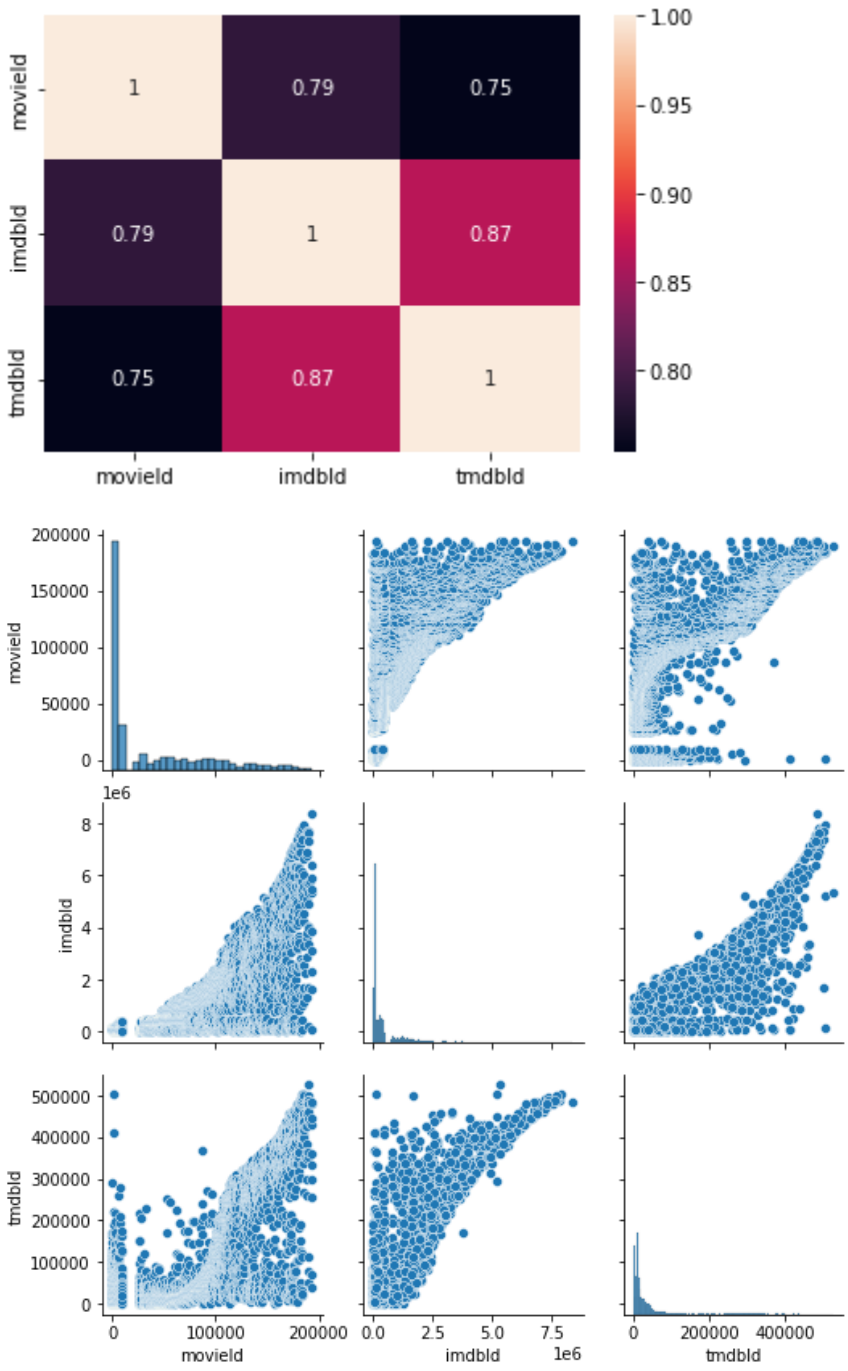
Rating dataset:



Tag dataset:



Links dataset:



Insights:

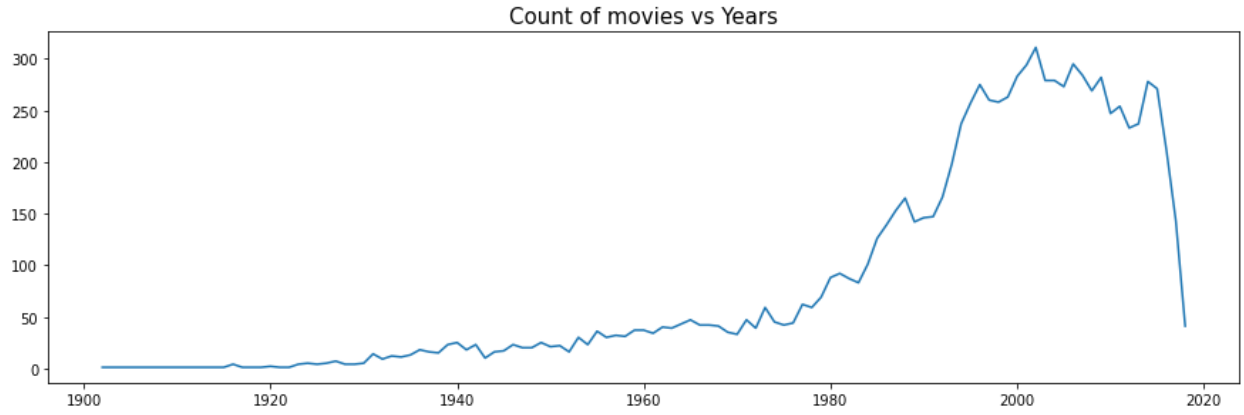
Graph 1: 1]Maximum movies are watched near the 2000 year as we can see a peak in below graph Graph 2: 2]Average rating is 3.5 and from graph, we can say the average rating for all year is nearby 3.5 3]Maximum Average rating went to 3.9 in the year 2013.

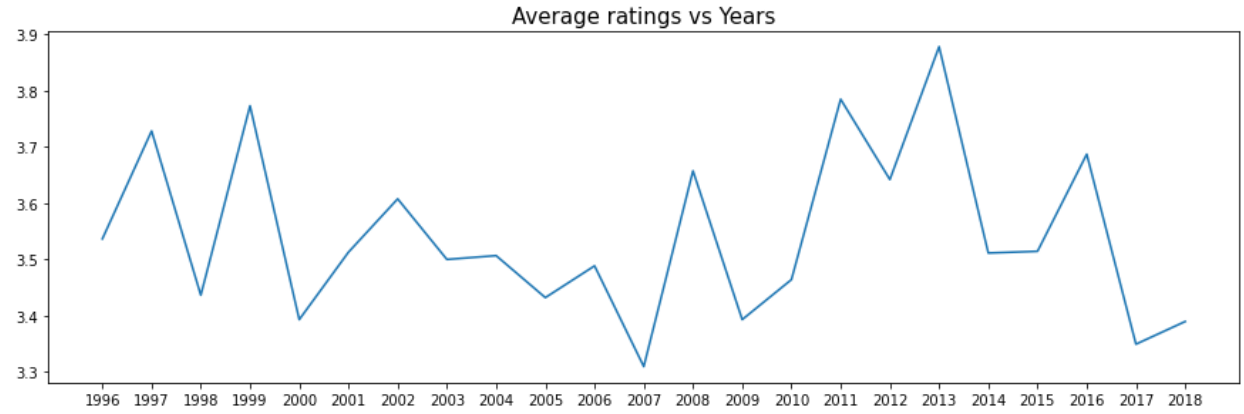
4] Minimum Average rating went to 3.3 in the year 2007.

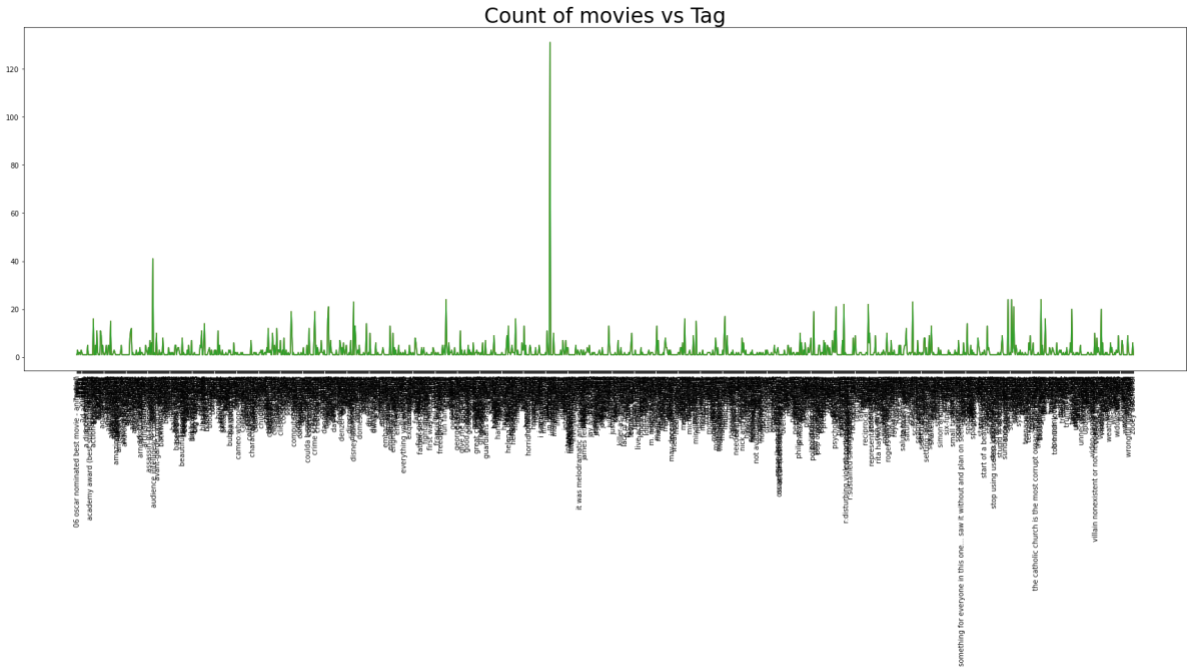
Graph 3:

5] 131 movies were tagged as in netflix queue.

6] 41 movies were tagged as atmospheric.







From the graph we get two highest count of movies for these two tag:
in netflix queue 131 atmospheric 41

Q2:

Assumption: No handling of upper and lower case difference for this recommendation system. This can be done easily but for now It has been assumed that the case sensitive data is expected as given in the input file

Approach:

1. Transactions are made by grouping the movies seen by a user using userId
2. These transactions are converted to pandas data frame and then apriori algorithm is applied to get the frequent itemsets with minimum support 0.08
3. After that 'rules' are retrieve from association_rules with minimum threshold of 0.5
4. We call recommend function to get the recommended movies for the given input. This function then calls top4recommendation function
5. Top4recommendation function first gets movies ids corresponding to the movie. If for the given set of movies rules are present then the top4 recommended movies are suggested on the behalf of highest confidence score. If no rules present then the combination of different size for the given sets are made and then rules are mined to get the top4 movies. If the movies are not present in data or support is so low that the apriori algorithm itself has ignored them , then movies with top 4 support we will be recommended by this system

   Example of test case is as follows:
   Sample test.tsv:

   movies
   Jumanji (1995)
   Toy Story (1995)
   "Dark Knight, The (2008)
   Dark Knight Rises, The (2012)"
   Inception (2010)
   Titanic (1997)
   "Dark Knight, The (2008)"

   Output:

movies	recommendation
Toy Story (1995)	Star Wars: Episode IV - A New Hope (1977) Pulp Fiction (1994) Shawshank Redemption, The (1994) Forrest Gump (1994)
Titanic (1997)	Shawshank Redemption, The (1994) Forrest Gump (1994) Silence of the Lambs, The (1991) Matrix, The (1999)
Jumanji (1995)	Toy Story (1995) Forrest Gump (1994) Lion King, The (1994) Jurassic Park (1993)
Inception (2010)	Shawshank Redemption, The (1994) Forrest Gump (1994) Matrix, The (1999) Dark Knight, The (2008)
Dark Knight, The (2008) Dark Knight Rises, The (2012)	Matrix, The (1999) Lord of the Rings: The Fellowship of the Ring, The (2001) Lord of the Rings: The Return of the King, The (2003) Inception (2010)
Dark Knight, The (2008)	Shawshank Redemption, The (1994) Forrest Gump (1994) Matrix, The (1999) Lord of the Rings: The Return of the King, The (2003)

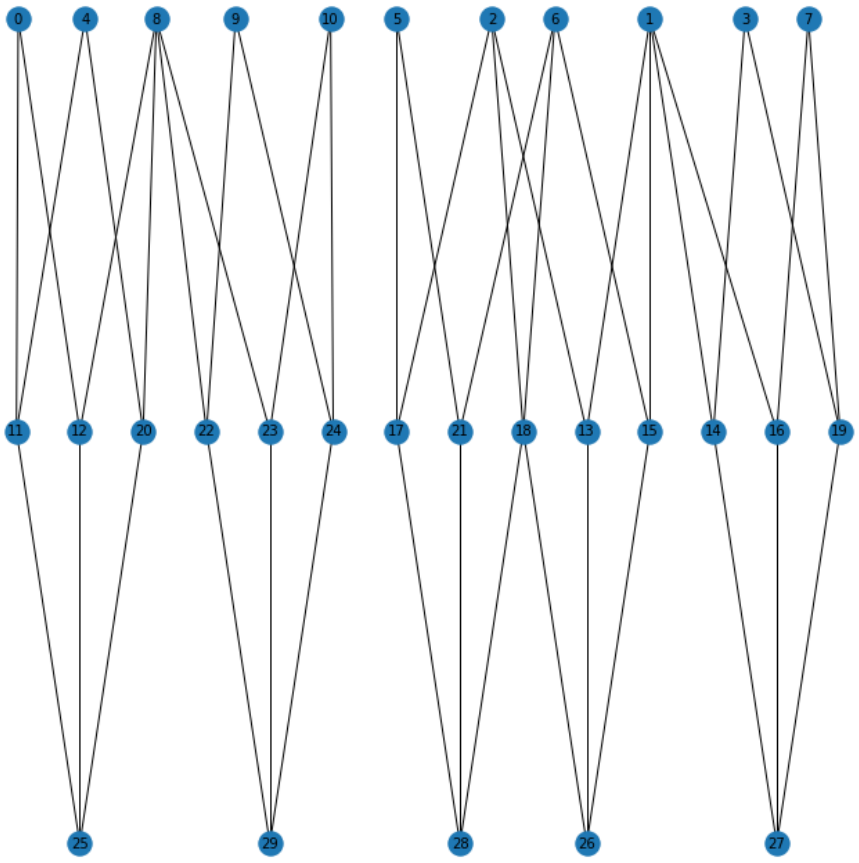
Q3:

Approach:

1. Sample set of 5000 with rating greater than 4 is being considered.
2. Apriori algorithm is applied with minimum support of 0.005 to get the frequent item sets in the sample
3. Then maximal frequent itemset is retrieve that is any item set which is frequent and none of its superset is frequent. Last 5 maximal frequent item set are displayed as follows:

   [frozenset({260, 1210, 2959}),
   frozenset({318, 858, 1221}),
   frozenset({318, 1198, 2571}),
   frozenset({858, 1213, 1221}),
   frozenset({2959, 5952, 8368})]

4. Visualisation of maximal frequent item set is as follows:
   Nodes 25 to 29 are maximal frequency itemset



5. Above label has names as shown below:

{0: ['Star Wars: Episode IV - A New Hope (1977)'],
1: ['Shawshank Redemption, The (1994)'],
2: ['Godfather, The (1972)'],
3: ['Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) '
'(1981)'],
4: ['Star Wars: Episode VI - Return of the Jedi (1983)'],
5: ['Goodfellas (1990)'],
6: ['Godfather: Part II, The (1974)'],
7: ['Matrix, The (1999)'],
8: ['Fight Club (1999)'],
9: ['Lord of the Rings: The Two Towers, The (2002)'],
10: ['Harry Potter and the Prisoner of Azkaban (2004)'],
11: ['Star Wars: Episode VI - Return of the Jedi (1983)',
'Star Wars: Episode IV - A New Hope (1977)'],
12: ['Star Wars: Episode IV - A New Hope (1977)', 'Fight Club (1999)'],
13: ['Godfather, The (1972)', 'Shawshank Redemption, The (1994)'],
14: ['Shawshank Redemption, The (1994)',
'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) '
'(1981)'],
15: ['Godfather: Part II, The (1974)', 'Shawshank Redemption, The (1994)'],
16: ['Matrix, The (1999)', 'Shawshank Redemption, The (1994)'],
17: ['Godfather, The (1972)', 'Goodfellas (1990)'],
18: ['Godfather, The (1972)', 'Godfather: Part II, The (1974)'],
19: ['Matrix, The (1999)',
'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) '
'(1981)'],
20: ['Star Wars: Episode VI - Return of the Jedi (1983)', 'Fight Club (1999)'],
21: ['Godfather: Part II, The (1974)', 'Goodfellas (1990)'],
22: ['Lord of the Rings: The Two Towers, The (2002)', 'Fight Club (1999)'],
23: ['Harry Potter and the Prisoner of Azkaban (2004)', 'Fight Club (1999)'],
24: ['Harry Potter and the Prisoner of Azkaban (2004)',
'Lord of the Rings: The Two Towers, The (2002)'],
25: ['Star Wars: Episode VI - Return of the Jedi (1983)',
'Star Wars: Episode IV - A New Hope (1977)',
'Fight Club (1999)'],
26: ['Godfather, The (1972)',
'Godfather: Part II, The (1974)',
'Shawshank Redemption, The (1994)'],
27: ['Shawshank Redemption, The (1994)',
'Matrix, The (1999)','Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) '
'(1981)'],
28: ['Godfather, The (1972)',
'Godfather: Part II, The (1974)',
'Goodfellas (1990)'],
29: ['Harry Potter and the Prisoner of Azkaban (2004)',
'Lord of the Rings: The Two Towers, The (2002)',
'Fight Club (1999)']

6.
Support values for the nodes are as follows:
0: 0.05185185185185185,
1: 0.07407407407407407,
2: 0.03888888888888889,
3: 0.03888888888888889,

4: 0.037037037037037035,
5: 0.02962962962963,
6: 0.03148148148148148,
7: 0.0592592592592926,
8: 0.05555555555555555,
9: 0.02962962962963,
10: 0.014814814814814815,
11: 0.007407407407407408,
12: 0.009259259259259259,
13: 0.007407407407407408,
14: 0.007407407407407408,
15: 0.007407407407407408,
16: 0.009259259259259259,
17: 0.012962962962963,
18: 0.009259259259259259,
19: 0.007407407407407408,
20: 0.005555555555555556,
21: 0.005555555555555556,
22: 0.007407407407407408,
23: 0.007407407407407408,
24: 0.005555555555555556,
25: 0.005555555555555556,
26: 0.005555555555555556,
27: 0.005555555555555556,
28: 0.005555555555555556,
29: 0.005555555555555556

Note:
Libraries used: pandas, numpy, mlxtend,networkx,pydot,matplotlib,itertools