CS 839 – Data Science

Stage 4: Integrating and performing analysis

Group Members: Pradeep Kumar, Pulkit Kapoor, Sonu Agarwal

Description:

In this project stage, our team performed merging two tables containing Movie data as entities.

Both the tables have the same schema as below:

Source1: http://www.imdb.com/ - The table from Source1 contains **3013 tuples**. This table includes the following columns (attributes):

MID,Title,Certificate,Genre,Rating,Running Time,Directors,Stars Cast,Country,Release Date,Production Company,Release Year,Release Month.

Source2: https://www.allmovie.com/ - The table from Source2 contains **3300 tuples**. This table includes the following columns (attributes):

MID,Title,Certificate,Genre,Rating,Running Time,Directors,Stars Cast,Country,Release Date,Production Company,Release Year,Release Month.

Therefore, merged table also has the same schema as above.

We used the matches created in the previous stage to merge these two tables. We decided to keep only matched tuples in the newly created tuple E. We did not use any other table to create table E.

Merging Functions:

- MergeTitle (Title a, Title b): return larger of two strings.
- MergeCertificate (Certificate a, Certificate b): return a if a is not null; else return b.
- MergeGenre (Genre a, Genre b): return superset of a and b
- MergeRating (Rating a, Rating b): return (a+b)/2
- MergeRunningTime (RunningTime a, RunningTime b): return superset of a and b
- MergeDirectors (Directors a, Directors b): return superset of a and b
- MergeStarsCast (StarsCast a, StarsCast b): return superset of a and b
- MergeCountry (Country a, Country b): return a if a is not null; else return b.
- MergeReleaseDate (ReleaseDate a, ReleaseDate b): return a if a is not null; else return b.

- MergeProductionCompany (ProductionCompany a, ProductionCompany b): return larger of two strings
- MergeReleaseYear (ReleaseYear a, ReleaseYear b): return a if a is not null; else return b.
- MergeReleaseMonth (ReleaseMonth a, ReleaseMonth b): return a if a is not null; else return b.

Statistics of Table E

Schema of Tuple E:

[MID,Title,Certificate,Genre,Rating,Running Time,Directors,Stars Cast,Country,Release Date,Production Company,Release Year,Release Month]

Number of Tuples

There are total 631 tuples in table E (equal to number of matches between table A and B in stage 3).

Sample Tuples

[36,American Hustle,R,Drama|Crime,7.65,138 min,David O. Russell,Jeremy Renner|Amy Adams|Christian Bale|Bradley Cooper,USA,20-Dec-13,Columbia Pictures|Annapurna Pictures|Atlas Entertainment,2013,December]

[93,Captain America: The First Avenger,PG13,Action|Adventure|Sci-Fi,7.45,124 min,Joe Johnston,Hugo Weaving|Hayley Atwell|Tommy Lee Jones|Samuel L. Jackson|Sebastian Stan|Chris Evans,USA,22-Jul-11,Paramount Pictures|Marvel Entertainment|Marvel Studios,2011,July]

[134,Dirty Dancing,PG13,Drama|Romance|Music|Musical,7.45,100 min,Emile Ardolino,Jennifer Grey|Patrick Swayze|Jerry Orbach|Cynthia Rhodes,USA,21-Aug-87,Great American Films Limited Partnership|Vestron Pictures,1987,August]

[201,Guardians of the Galaxy Vol. 2,PG13,Action|Science Fiction|Adventure|Sci-Fi,7.35,136 min,James Gunn,Vin Diesel|Dave Bautista|Bradley Cooper|Chris Pratt|Zoe Saldana,USA|USA|USA,5-May-17,Marvel Studios|Walt Disney Pictures,2017,May]

Data Analysis

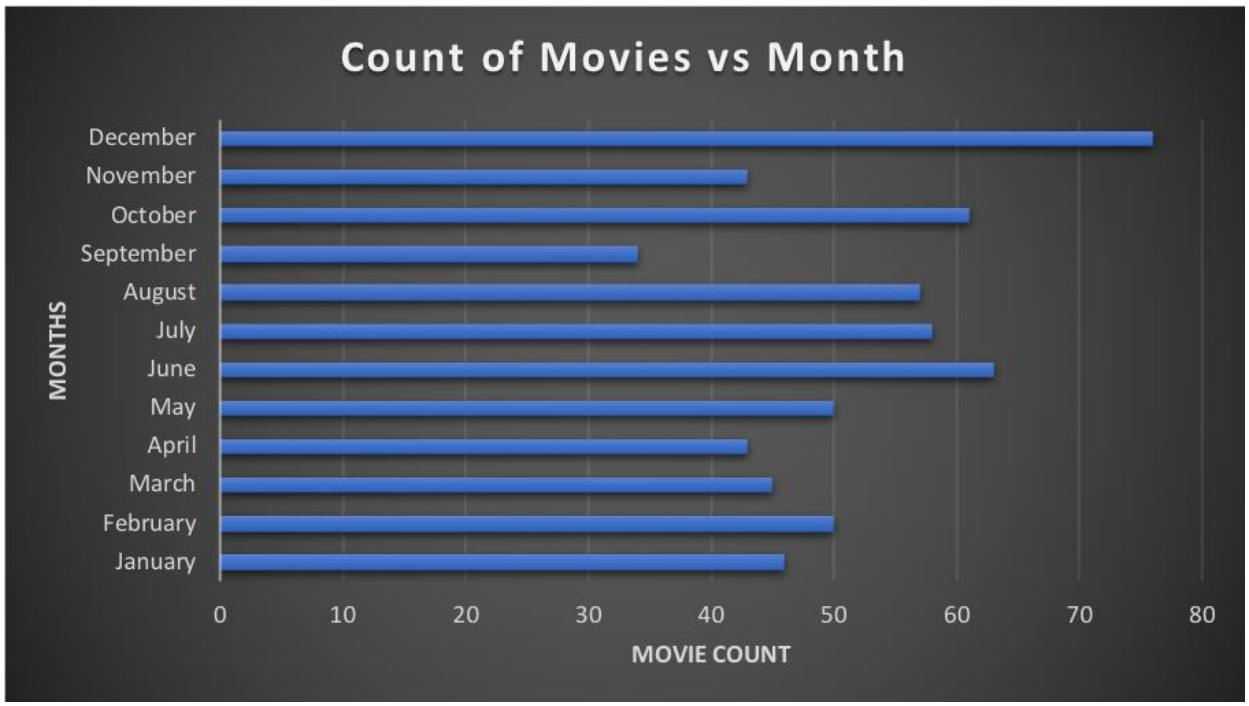
We use some of the concepts of OLAP for our analysis. We mostly used roll up and drill down operations to find the aggregated results.

- We analyze the frequency of movies across different months of the year.

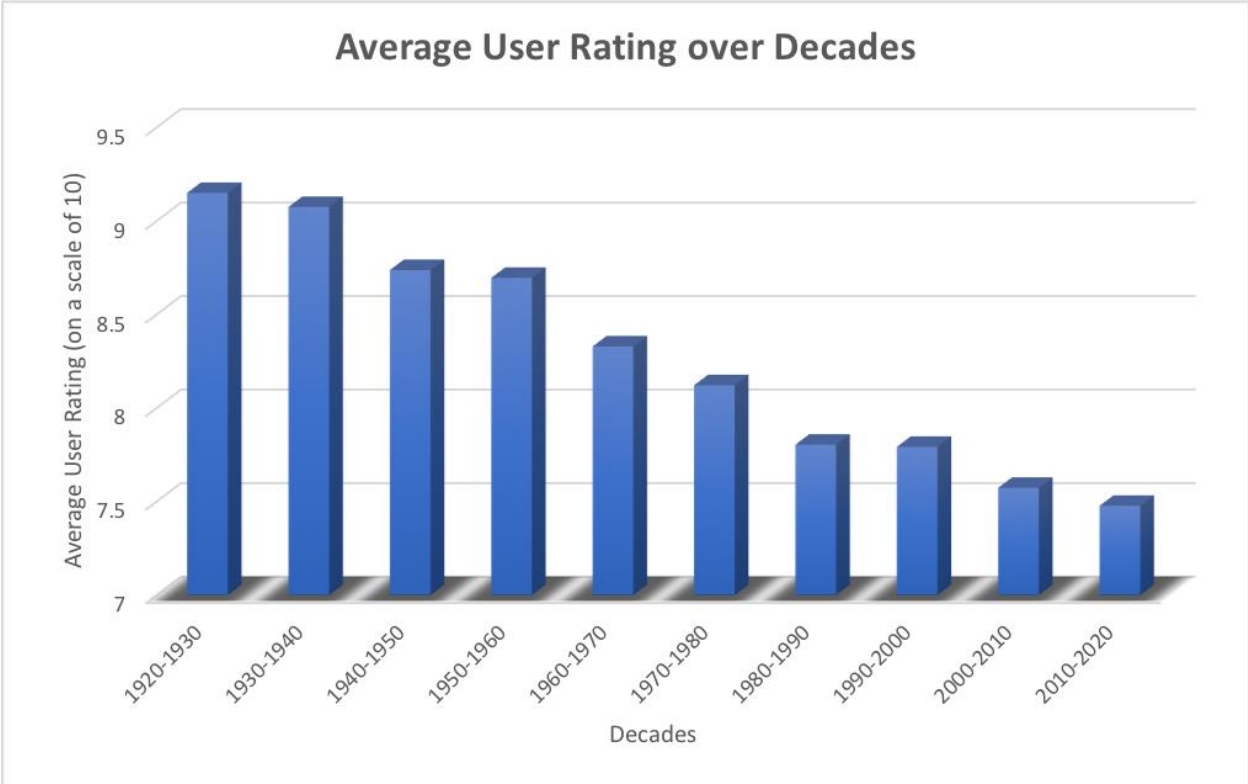
- We analyze if there is any pattern in user “rating” and “run time” over the decades and over different certificates of the movie.
- We analyze the frequency of movies over the decades and over different certificates of the movie.
- We analyze the frequency of movies over different rating buckets.
- We analyze if there is any correlation between “rating” and “run time”.

Analysis results and interpretations:

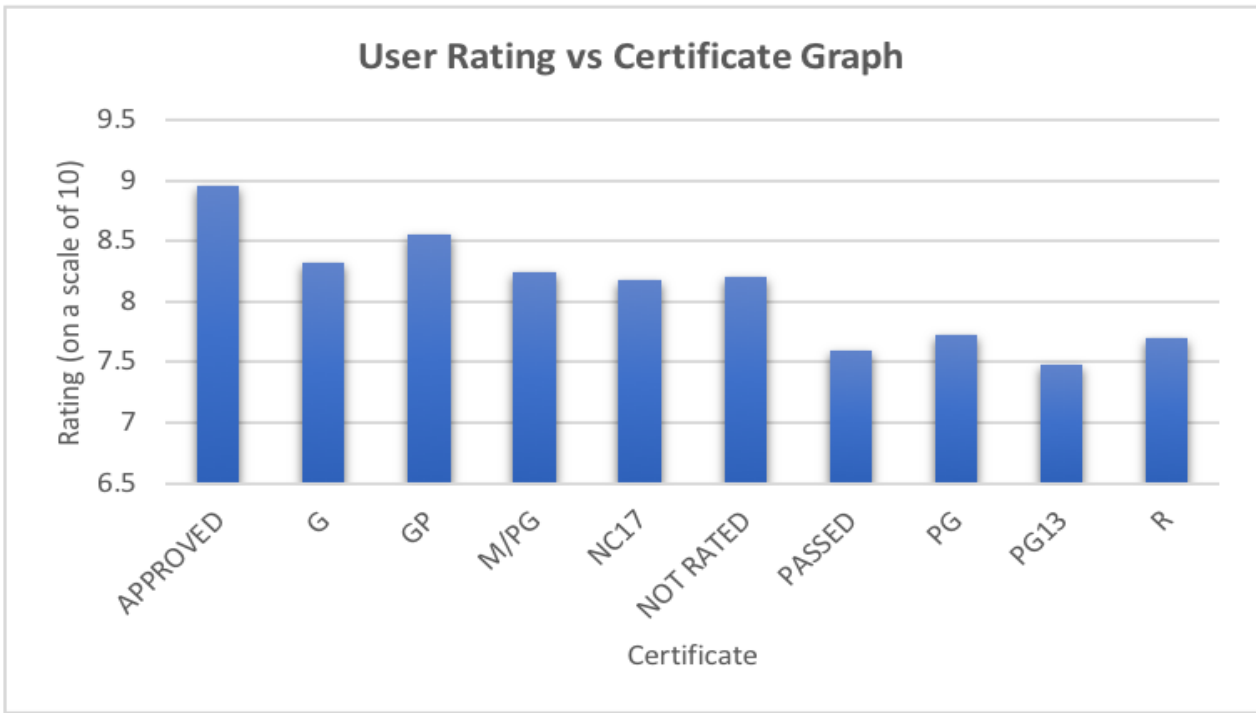
a) Below graph indicates an interesting pattern in the number of movies released across different months. We can see that maximum number of movies are released in June, October and December, which probably corresponds to summer, Thanksgiving and Christmas seasons. We performed groupBy operation on months attribute and then found the aggregate count of each group.



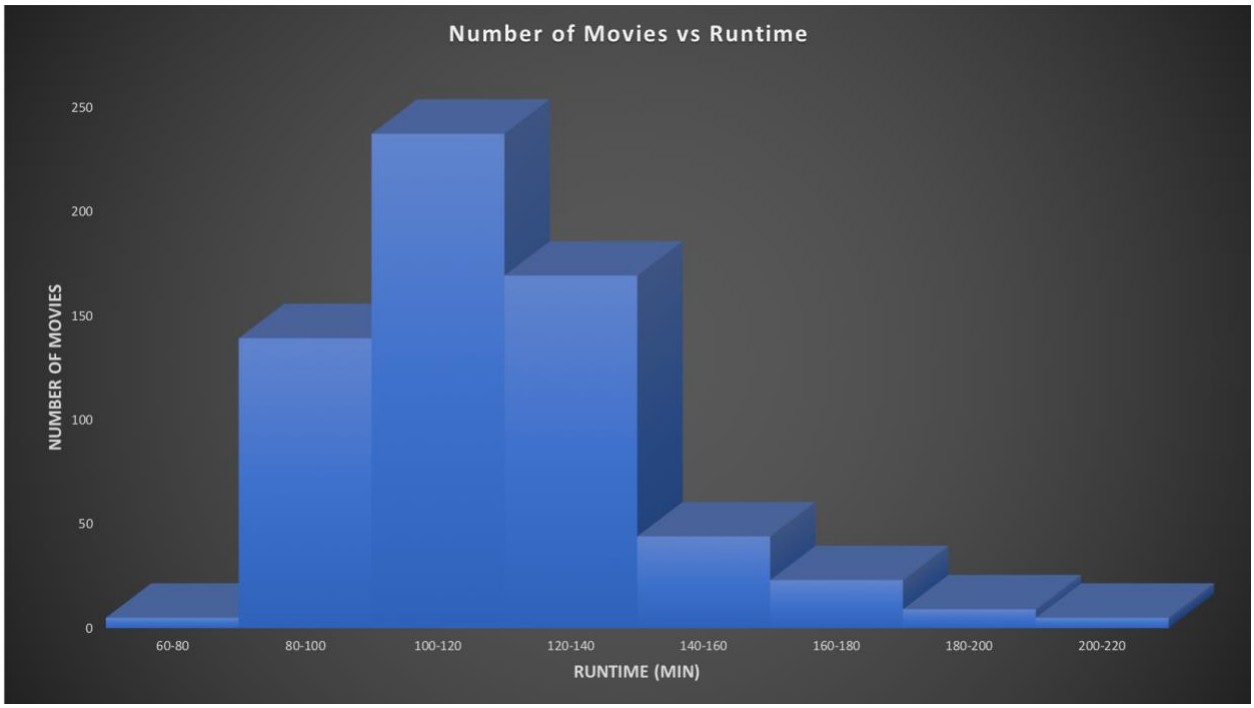
b) Below graph indicates that average user ratings have been decreasing over the decades. This could be attributed to people’s stricter criticism of more recent movies. However, there may be more reasons for which we need to do more data analysis. We performed groupBy operation on release year attribute and then aggregate over user rating. After this, we bucketize the year attribute to generate decades.



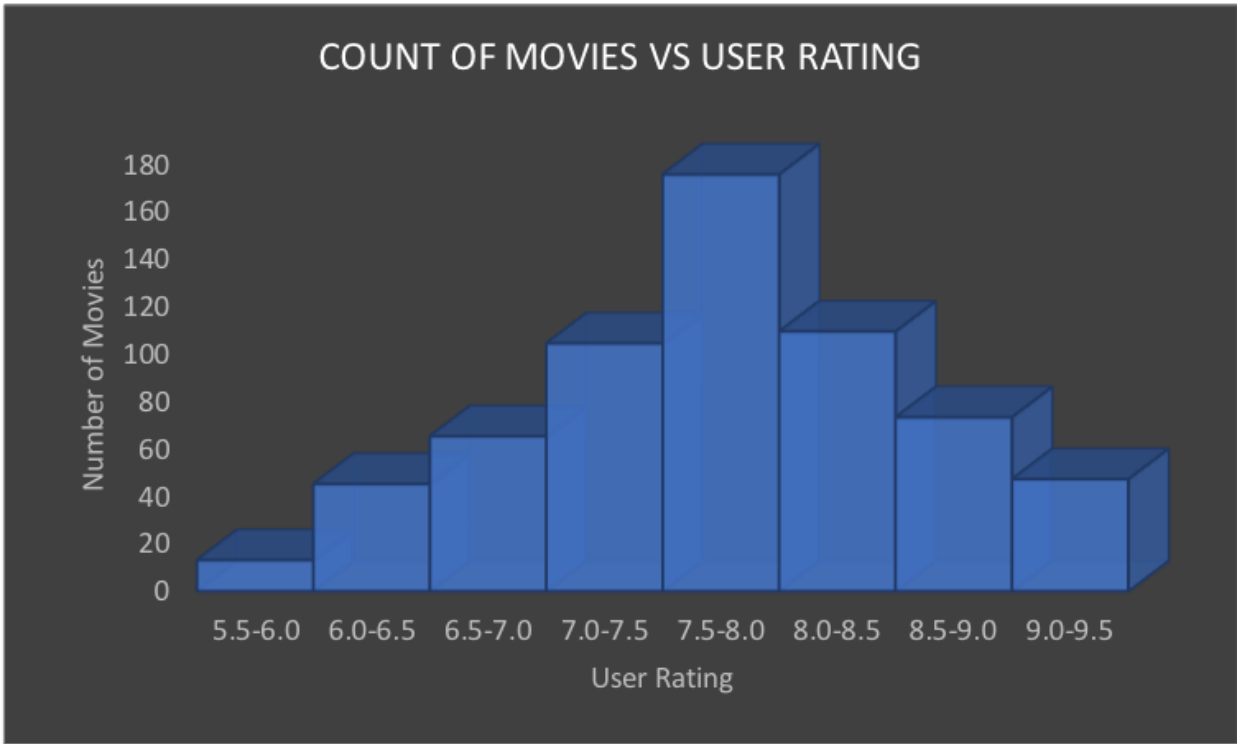
c) Below graph indicates that average rating is lowest for movies with PG and PG13 certificates while highest for APPROVED certificate. We performed groupBy operation on certificate attribute and then aggregate over user rating.



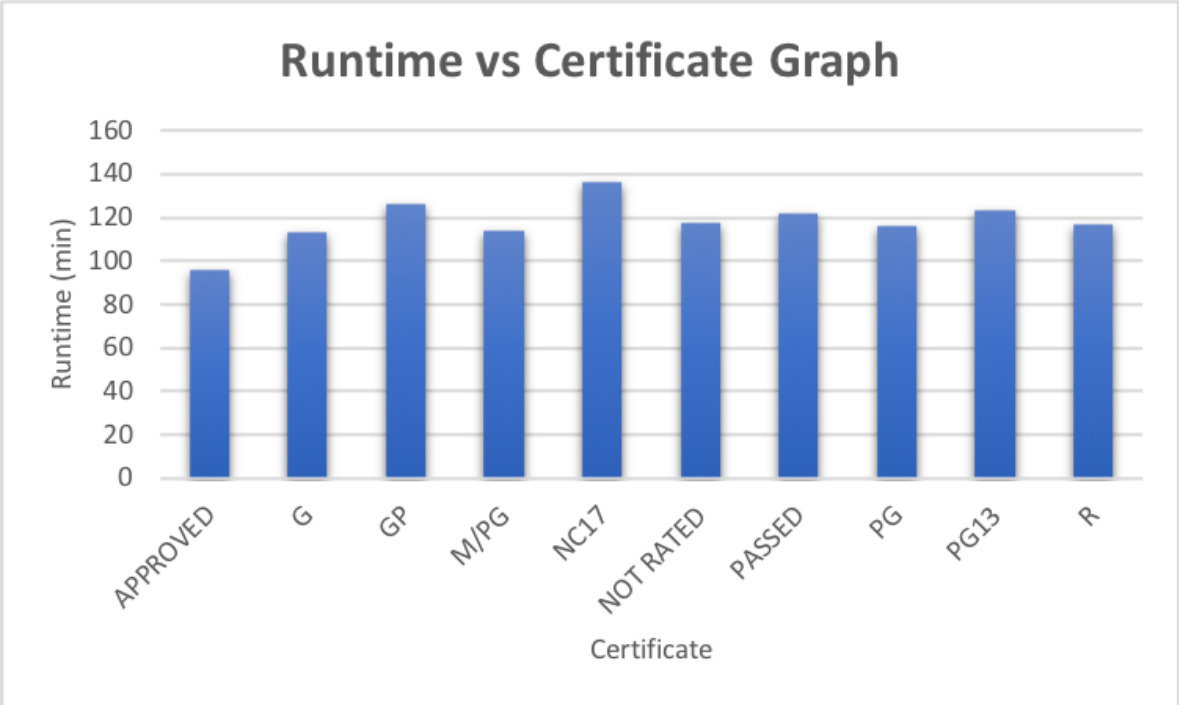
d) Below graph indicates that most of the movies have runtime in the range of 100-120 minutes, which is the average runtime for any movie especially hollywood. There are very few movies with runtime less than 80 minutes and more than 180 minutes. We bucketize the runtime to create a histogram and then did an aggregate on the movie count.



e) Below graph indicates that the maximum number of movies falls in the rating bucket of 7.5 to 8. We bucketize the user rating to create a histogram and then did an aggregate on the movie count.



f) Below graph doesn't indicate any significant pattern in runtime across different certificates. This shows that runtime does not have any dependence on Certificates. We performed groupBy operation on certificate attribute and then aggregate over runtime.



g) Correlations: We expected that there could be some correlation between run time and user ratings. We calculated Pearson product-moment correlation coefficient between these attributes, which is equal to 0.26. This value is relatively lower and indicate very week correlation between these two attributes, contrary to our expectation.

Problem faced during analysis process

- We faced some problems because of some null values in the rating and run time attributes. We filled these values with the average rating and runtime to continue with the analysis.
- Certificates were not normalized, i.e., there were multiple variations of the same certificate for some of the certificates. This led to more than correct number of certificates in our group by operation (roll up) operation. We normalized this certificate values to do the analysis efficiently.

If we had more time, we could do following analysis:

- We could add some more attributes to the table and try to predict the gross income of the movie. We could use attributes like ratings of the actors in the movie, ratings of the directors, average income of movies of corresponding actors and directors, release month etc. to predict the gross income of the movie.

- We could do clustering of the movies to divide the movies into different genres. But again, we would need some information in the form of attributes to facilitate this analysis. For example, we might need the brief synopsis of the movie.

Table Merging Code:

```
# Import py_entitymatching package and other required packages
import py_entitymatching as em
import os
import pandas as pd
import math

pd.set_option('display.max_columns',30)
pd.set_option('display.max_rows',1000)

# Get the datasets directory
datasets_dir = '../data'

# Get the paths of the input tables
```

```
path_A = datasets_dir + os.sep + 'Table_IMDB_with_ID.csv'
path_B = datasets_dir + os.sep + 'Table_Allmovie_with_ID.csv'

# Read the CSV files and set 'MID' as the key attribute
A = em.read_csv_metadata(path_A, key='MID')
B = em.read_csv_metadata(path_B, key='MID')

# Matched_Output.csv contains the matched output between A and B Tables.
path_M = datasets_dir + os.sep + 'Matched_Output.csv'
M = em.read_csv_metadata(path_M, key='_id')

def getTitle(titleA, titleB):
   return titleA if len(titleA) > len(titleB) else titleB

def getGenre(genreA, genreB):
   genreSetA = set()
   genreSetB = set()
   if(genreA!='NULL'):
      genreSetA = set(genreA.split('|'))
   if(genreB!='NULL'):
      genreSetB = set(genreB.split('|'))
   genreSet = genreSetA.union(genreSetB);
   return '|'.join(list(genreSet))

def getRating(ratingA, ratingB):
   return (ratingA + ratingB)/2

def getRunningTime(runTimeA, runTimeB):
   if runTimeA!='NULL':
      return runTimeA
   else:
      return runTimeB

def getDir(dirA, dirB):
   dirSetA = set()
   dirSetB = set()
   if(dirA!='NULL'):
      dirSetA = set(dirA.split('|'))
   if(dirB!='NULL'):
      dirSetB = set(dirB.split('|'))
   dirSet = dirSetA.union(dirSetB);
   return '|'.join(list(dirSet))

def getCast(castA, castB):
```

```
    castSetA = set()
    castSetB = set()
    if(castA!='NULL'):
        castSetA = set(castA.split('|'))
    if(castB!='NULL'):
        castSetB = set(castB.split('|'))
    castSet = castSetA.union(castSetB);
    return '|'.join(list(castSet))

def getReleaseDate(relA, relB):
    if relA!='NULL':
        return relA
    else:
        return relB

def getProductionCompany(relA, relB):
    if relA!='NULL':
        return relA
    else:
        return relB

def getReleaseYear(relA, relB):
    if relA!='NULL':
        return relA
    else:
        return relB

def getReleaseMonth(relA, relB):
    if relA!='NULL':
        return relA
    else:
        return relB

index = 0
for indexA, rowA in A.iterrows():
    for indexB, rowB in B.iterrows():
        aRow = M['l_MID'] == rowA['MID']
        bRow = M['r_MID'] == rowB['MID']
        cRow = M['predicted'] == 1
        if len(M.loc[aRow & bRow & cRow]) > 0:
            print rowA['MID'], rowB['MID']
            resultdf.loc[index, 'Title'] = getTitle(rowA['Title'], rowB['Title'])
            resultdf.loc[index, 'Certificate'] = rowA['Certificate']
            resultdf.loc[index, 'Genre'] = getGenre(rowA['Genre'], rowB['Genre'])
```

```
        resultdf.loc[index, 'Rating'] = getRating(rowA['Rating'], rowB['Rating'])
        resultdf.loc[index, 'Running Time'] = getRunningTime(rowA['Running Time'], rowB['Running Time'])
        resultdf.loc[index, 'Directors'] = getDir(rowA['Directors'], rowB['Directors'])
        resultdf.loc[index, 'Stars Cast'] = getCast(rowA['Stars Cast'], rowB['Stars Cast'])
        resultdf.loc[index, 'Country'] = rowA['Country']
        resultdf.loc[index, 'Release Date'] = getReleaseDate(rowA['Release Date'], rowB['Release Date'])
        resultdf.loc[index, 'Production Company'] = getProductionCompany(rowA['Production Company'], rowB['Production Company'])
        resultdf.loc[index, 'Release Year'] = getReleaseYear(rowA['Release Year'], rowB['Release Year'])
        resultdf.loc[index, 'Release Month'] = getReleaseMonth(rowA['Release Month'], rowB['Release Month'])
        index += 1
```