[redacted]

November 7, 2018

[redacted]

##Question 1

###1. Please write codes to read the data file TrainingData.csv. [redacted]

```
In [1]: #1. Open the CSV file
        import os
        os.chdir("[redacted]")
        TrainingData = open('[redacted]').readlines()

        #2. Set the first row as the header (Variable names) and clean the data
        VariableNames = TrainingData[0].replace(' ','').replace('\n','').split(';')[:]

        #3 Set the subsequent rows as the record in the dataset and clean the data
        TrainingRecords = [x.replace('_','').replace(' ','').replace('\n','').split(';')[:] fo

        #4. Define a new list, let the first row is the header (variable names) and data are s
        TrainingData = [VariableNames] + TrainingRecords

        print(TrainingData[:4])
```

```
[['ID', 'NumComplains', 'hour_id', 'LCID', 'RACH_Setup_Completion_Success_Rate_AVG', 'Total_RR
```

###2. Determine the number of variables and the number of records in this dataset.

```
In [2]: #1. Calculate the number the number of variables
        NumberofVaribles = len(TrainingData[0])
        print("1. the number of variables is : %a" %NumberofVaribles)

        #2. Calculate the number of records in this dataset
        NumberofRecords = len(TrainingData[1:])
        print("2. the number of records in the dataset is: %a" %NumberofRecords)
```

```
1. the number of variables is : 22
2. the number of records in the dataset is: 13949
```

###3. Store the variable names in a list.

```
In [3]: #Store the variable names in a list
        print("1. As we can find in solution to question 1, all the variable names have been sa
        print('2. The list of variable names is as followed: ','\n', VariableNames)
```

```
1. As we can find in solution to question 1, all the variable names have been saved in a list r

2. The list of variable names is as followed:
 ['ID', 'NumComplains', 'hour_id', 'LCID', 'RACH_Setup_Completion_Success_Rate_AVG', 'Total_RRC
```

###4. Determine if there is any missing values in the data set. If yes, please report the total number of missing values.

```
In [4]: #1. Count if there is any missing values in the data set
        count = 0
        for i in TrainingData:
            count += i.count('')

        #2. Report the total number of missing values
        print('1. Yes, there are missing values in the data set.')
        print("2. The total number of missing values is: %a" %count)
```

```
1. Yes, there are missing values in the data set.
2. The total number of missing values is: 1031
```

###5. Find the number of distinct LCID in the data set.
####Method One

```
In [5]: #1. Create a list of LCID records
        LCID_list = [i[3] for i in TrainingData]; del LCID_list[0]

        #2. Find the number of distinct LCID in the list above
        DistinctLCID = len(set(map(tuple,LCID_list)))
        print("The number of distinct LCID in the dataset is:", DistinctLCID)
```

```
The number of distinct LCID in the dataset is: 11745
```

####Method Two

```
In [6]: #Rearrange the training records, make a dictionary of the record inside a list
        TrainingRecordsList = [dict(zip(VariableNames,TrainingRecords[i])) for i in range(len(T
```

```
In [7]: #1. Divide the entire data set(TraningRecordsList) by distinct value of LCID
        from collections import defaultdict
        DividedData = defaultdict(list)
        for records in TrainingRecordsList:
            DividedData[records['LCID']].append(records)
        DividedData = list(DividedData.values())

        #2. Count the length of the dataset
        print("The number of distinct LCID in the dataset is:", len(DividedData))
```

```
The number of distinct LCID in the dataset is: 11745
```

###6. Find the variable with the most missing values.

```
In [8]: # Define a new stucture
        TrainingRecordsList2 = []
        for item in VariableNames:
            TrainingRecordsList2.append([i[item] for i in TrainingRecordsList])
        NewSrtucture = {x: TrainingRecordsList2[VariableNames.index(x)] for x in VariableNames
```

```
In [9]: # Find the variable with the most missing values
        MissingValues = {}
        for item in VariableNames:
            MissingValues[item] = NewSrtucture[item].count('')
        MissingValues = sorted(MissingValues.items(), key= lambda x:x[1], reverse= True)
        print('The variable with the most missing values is:',MissingValues[0][0],'\n','The mis
```

```
The variable with the most missing values is: Total_RRC_Connection_Re_establishment_Success_Ra
 The missing values are: 411
```

###7. Convert the variable hour_id to datetime format.

```
In [10]: from datetime import datetime
         for i in TrainingRecordsList:
             i['hour_id'] = datetime.strptime(i['hour_id'],'%m/%d/%Y%H:%M')
         print("An example for the converted result:",'\n',TrainingRecordsList[1]['hour_id'])
```

```
An example for the converted result:
 2015-11-20 00:27:00
```

###8. What is the time duration of the entire data set?

```
In [11]: TrainingRecordsList.sort(key= lambda x:x['hour_id'])
         Duration = TrainingRecordsList[-1]['hour_id'] - TrainingRecordsList[0]['hour_id']
         print("The time duration of the entire data set is:", Duration)
```

```
The time duration of the entire data set is: 13 days, 15:00:00
```

###9. Determine the number of records per day.

```
In [12]: from copy import deepcopy
         from collections import Counter
         import datetime
         TrainingRecordsList1 = deepcopy(TrainingRecordsList)
         for i in TrainingRecordsList1:
             i['hour_id'] = datetime.date(i['hour_id'].year, i['hour_id'].month, i['hour_id'].
         CountTimeList = list(Counter(i['hour_id'] for i in TrainingRecordsList1).items())
         print('---Date----Number--')
         for i in CountTimeList:
             print(i[0],':  ',i[1])
```

```
---Date----Number--
2015-11-13 :   982
2015-11-14 :   993
2015-11-15 :   990
2015-11-16 :   946
2015-11-17 :   851
2015-11-18 :   824
2015-11-19 :   761
2015-11-20 :   937
2015-11-21 :   985
2015-11-22 :   963
2015-11-23 :   1054
2015-11-24 :   1095
2015-11-25 :   1317
2015-11-26 :   1251
```

###10. Use the median method in the statistics package (from statistics import median) or else, do the followings:

###(a) Divide the entire data set by distinct value of LCID.

###(b) For each distinct LCID value, determine the median of each variables in the divided data set.

###(c) Package the result in (b) in a dictionary.

```
In [13]: #(a) Divide the entire data set by distinct value of LCID
         from collections import defaultdict
         DividedData = defaultdict(list)
         for records in TrainingRecordsList:
             DividedData[records['LCID']].append(records)
         DividedData = list(DividedData.items())

         #Print an example
         print('LCID:',DividedData[10][0],'\n',DividedData[100][1])
```

```
LCID: 235Q9113
 [{'ID': '1660', 'NumComplains': '0', 'hour_id': datetime.datetime(2015, 11, 13, 9, 0), 'LCID'
```

```
In [14]: #(b) For each distinct LCID value, determine the median of each variables in the divi
         from copy import deepcopy
         from statistics import median
         import datetime
         #1.1 Determine a MedianList collecting all the data for each variable by distinct LCI
         DividedData1 = deepcopy(DividedData); MedianList = []
         for records in DividedData1:
             listx = []
             for key in VariableNames:
                 listx.append([item[key] for item in records[1] if item[key] != '' if key in i
             MedianList.append(listx)

         #1.2 Convert data to float and determine the median value for each variable in distin
         def Tofloat(x):
             try:
                 return float(x)
             except:
                 return x
         for x in MedianList:
             # Clean the data:(1) Transfrom the datetime to timestamp; (2) Transfrom other dat
             x[2] = [item.timestamp() for item in x[2]]
             for y in x:
                 for item in range(len(y)):
                     #Delete missing values
                     if y[item] == '':
                         del y[item]
                     y[item] = Tofloat(y[item])
             # Delete duplicate LCID value
             x[3] = list(set(x[3])); x[3][0] = str(x[3][0])
             # Determine the median in float list of each records
             for y in range(len(x)):
                 try:
                     x[y] = median(x[y])
                 except:
                     x[y] = x[y]
             x[2] = datetime.datetime.fromtimestamp(x[2])
         #2 Make the data more readable
         MedianList = [dict(zip(VariableNames,MedianList[i])) for i in range(len(MedianList))]
```

```
In [15]: #(c) Package the result in (b) in a dictionary
         ResultInDict = {}
         for lcid in range(len(MedianList)):
             ResultInDict[MedianList[lcid]['LCID']] = MedianList[lcid]

         #Print the median result for the example in (a) part
         ResultInDict['235Q9113']
```

```
Out[15]: {'ID': 1570.0,
          'NumComplains': 0.0,
          'hour_id': datetime.datetime(2015, 11, 13, 8, 0),
          'LCID': '235Q9113',
          'RACH_Setup_Completion_Success_Rate_AVG': 0.98017,
          'Total_RRC_Connection_Setup_Success_Ratio_AVG': 0.99854,
          'Total_RRC_Connection_Re_establishment_Success_Ratio_AVG': 0.5625,
          'Radio_Bearer_Success_Ratio_AVG': 1.0,
          'ERAB_SetupSR_AVG': 1.0,
          'HO_Success_Ratio_intra_eNB_AVG': 0.98374,
          'Average_CQI_AVG': 7.77209,
          'Radio_Bearer_Drop_Ratio_AVG': [],
          'Average_Physical_Resource_Block_Usage_UL_AVG': 0.125,
          'Average_Physical_Resource_Block_Usage_DL_AVG': 3.85,
          'Complete_RACH_Setup_Success_Rate_AVG': 0.90299,
          'AGG1_blocked_distribution_rate_AVG': 0.0,
          'AGG2_blocked_distribution_rate_AVG': 0.33333,
          'AGG4_blocked_distribution_rate_AVG': 0.16667,
          'AGG8_blocked_distribution_rate_AVG': 0.5,
          'ECM_failure_ratio_due_to_Rejection_by_RRM_RAC_AVG': 0.0,
          'AGG_level_block_Rate_AVG': 1e-05,
          'ERAB_Drop_Ratio_with_UE_Lost_cause_initiated_by_eNB_AVG': 0.0}
```

###11.Determine the number of Complaint cases and Non-complaint cases in the entire data set.

```
In [16]: CountOfComplaints = 0
         for i in TrainingRecordsList:
             if int(i['NumComplains']) != 0:
                 CountOfComplaints += 1
         print('1. The number of complaint cases is:', CountOfComplaints)
         print('2. The number of non-complaint cases is:',len(TrainingRecordsList)-CountOfCompl
```

```
1. The number of complaint cases is: 1559
2. The number of non-complaint cases is: 12390
```

###12. Determine the top 10 LCIDs with the most complaint cases.

```
In [17]: #1. Create a list containing distinct LCID and number of compliants
         LCIDComplaints_list = {}
         for records in DividedData:
             CountOfComplaints = 0
             for item in records[1]:
                 if int(item['NumComplains']) != 0:
                     CountOfComplaints += int(item['NumComplains'])
                 LCIDComplaints_list[item['LCID']] = CountOfComplaints
         LCIDComplaints_list = [[x, LCIDComplaints_list[x]] for x in LCIDComplaints_list]

         #2. Sort the result in the reverse order
         LCIDComplaints_list.sort(key= lambda x:x[0])
         LCIDComplaints_list.sort(key= lambda x:x[1], reverse=True)

         #3 Combine the variable names with the corresponding records and print the result
         VariableNames1 = [VariableNames[3],VariableNames[1]]
         LCIDComplaints_list = [dict(zip(VariableNames1,LCIDComplaints_list[i])) for i in range
         print("The top 10 LCIDs with the most complaint cases are:")
         for i in range(10):
             print(i+1,":",LCIDComplaints_list[i])
```

```
The top 10 LCIDs with the most complaint cases are:
1 : {'LCID': '24880112', 'NumComplains': 28}
2 : {'LCID': '221D7131', 'NumComplains': 10}
3 : {'LCID': '24879131', 'NumComplains': 7}
4 : {'LCID': '22156112', 'NumComplains': 6}
5 : {'LCID': '80256111', 'NumComplains': 6}
6 : {'LCID': '11039231', 'NumComplains': 5}
7 : {'LCID': '221L0111', 'NumComplains': 5}
8 : {'LCID': '2361A232', 'NumComplains': 5}
9 : {'LCID': '110V4233', 'NumComplains': 4}
10 : {'LCID': '23783112', 'NumComplains': 4}
```

###13. Calculate the median value per day per each variable in the entire data set.

```
In [18]: #Divide the entire data set by distinct value of date
         from collections import defaultdict
         DividedDate = defaultdict(list)
         for records in TrainingRecordsList1:
             DividedDate[records['hour_id']].append(records)
         DividedDate = list(DividedDate.items())
```

```
In [19]: #(b) For each distinct LCID value, determine the median of each variables in the divi
         from copy import deepcopy
         from statistics import median
         ##Considering that LCID consists of number and alphabet, delete the LCID data in the
         VariableNames1 = deepcopy(VariableNames)
         del VariableNames1[3]

         #1.1 Determine a MedianList collecting all the data for each variable by distinct LCI
         MedianList_for_Date = []
         for records in DividedDate:
             listx = []
             for key in VariableNames1:
                 listx.append([item[key] for item in records[1] if item[key] != '' if key in it
             MedianList_for_Date.append(listx)

         #1.2 Convert data to float and determine the median value for each variable in distin
         for x in MedianList_for_Date:
             # Delete duplicate date and LCID value
```

```
    x[2] = list(set(x[2]))
    for y in range(len(x)):
        for item in range(len(x[y])):
            x[y][item] = Tofloat(x[y][item])
        # Determine the median in float list of each records
        x[y] = median(x[y])

#2 Make the data more readable
MedianList_for_Date = [dict(zip(VariableNames1,MedianList_for_Date[i])) for i in range
```

In [20]:

```
#Print an example
print('The median value per each variable except LCID for 2015-11-13 is:')
MedianList_for_Date[0]
```

```
The median value per each variable except LCID for 2015-11-13 is:
```

Out[20]:

```
{'ID': 2050.5,
 'NumComplains': 0.0,
 'hour_id': datetime.date(2015, 11, 13),
 'RACH_Setup_Completion_Success_Rate_AVG': 0.9980508,
 'Total_RRC_Connection_Setup_Success_Ratio_AVG': 0.9991566000000001,
 'Total_RRC_Connection_Re_establishment_Success_Ratio_AVG': 0.43286959999999997,
 'Radio_Bearer_Success_Ratio_AVG': 0.9995161,
 'ERAB_SetupSR_AVG': 0.99990125,
 'HO_Success_Ratio_intra_eNB_AVG': 0.99878215,
 'Average_CQI_AVG': 9.726086,
 'Radio_Bearer_Drop_Ratio_AVG': 0.0013826665,
 'Average_Physical_Resource_Block_Usage_UL_AVG': 0.721875,
 'Average_Physical_Resource_Block_Usage_DL_AVG': 5.7171875,
 'Complete_RACH_Setup_Success_Rate_AVG': 0.867407212,
 'AGG1_blocked_distribution_rate_AVG': 0.17026616049999999,
 'AGG2_blocked_distribution_rate_AVG': 0.193446128,
 'AGG4_blocked_distribution_rate_AVG': 0.155385,
 'AGG8_blocked_distribution_rate_AVG': 0.25,
 'ECM_failure_ratio_due_to_Rejection_by_RRM_RAC_AVG': 0.0,
 'AGG_level_block_Rate_AVG': 4.85e-05,
 'ERAB_Drop_Ratio_with_UE_Lost_cause_initiated_by_eNB_AVG': 0.0}
```

###14. Use the first 5 digits of the LCID values to define a new variable Region.

In [21]:

```
#Add NewLCID and its corresponding data in TrainingRecordsList
for i in TrainingRecordsList:
    i["NewLCID"] = i["LCID"][:5]

#Divide the entire data set by distinct value of LCID
from collections import defaultdict
NewData = defaultdict(list)
for records in TrainingRecordsList:
    NewData[records['NewLCID']].append(records)
NewData = list(NewData.items())
```

###15. Determine the region with the most complaint cases found in the data set.

```
In [22]: #Add the count number of complaints in each disctinct data in 'NewData' in question 1
         countlist = []
         for records in NewData:
             records = list(records)
             CountOfComplaints = 0
             for item in records[1]:
                 if int(item['NumComplains']) != 0:
                     CountOfComplaints += int(item['NumComplains'])
             records[1].append({'CountOfComplaints':CountOfComplaints})
             countlist.append(CountOfComplaints)

         #3. Determine the region with the most complaint cases found in the data set.
         MaxComplaintsIndex = countlist.index(max(countlist))
         MaxComplaintsRegion = NewData[MaxComplaintsIndex]
```

```
In [23]: #Print some detial about the example
         print('The maximum complaints are:',max(countlist))
         print('The NewLCID of the region is:',MaxComplaintsRegion[0])
```

```
The maximum complaints are: 28
The NewLCID of the region is: 24880
```

##Question 2
###1. Import the two data files with an appropriate separator. Do the followings:
###(a) Set the timestamp variable to its datetime format using datetime.fromtimestamp() method.
###(b) Add leading zeros to the movieid and userid with zfill(4) method, e.g. "0023".

```
In [9]: #1.1 Open the fisrt file---u.data
        import os
        os.chdir("[redacted]")
        file1 = open('[redacted]').readlines()
        udata = [x.replace('\n','').split('\t')[:] for x in file1[:]]

        #1.2 Rearrange the dataset, make a dictionary pairing variable name with the data list
        VariableNames1 = ['userid', 'movieid', 'rating', 'timestamp']
        udata = [dict(zip(VariableNames1,udata[i])) for i in range(len(udata))]

        #2.(a)+(b) Set the timestamp variable to datetime format and add leading zeros to the
        from datetime import datetime
        for i in udata:
            i['timestamp'] = datetime.fromtimestamp(int(i['timestamp']))
            i['userid'] = i['userid'].zfill(4)
            i['movieid'] = i['movieid'].zfill(4)
```

```
In [10]: #Print an example
         udata[:3]
```

```
Out[10]: [{'userid': '0196',
           'movieid': '0242',
           'rating': '3',
           'timestamp': datetime.datetime(1997, 12, 4, 23, 55, 49)},
          {'userid': '0186',
           'movieid': '0302',
           'rating': '3',
           'timestamp': datetime.datetime(1998, 4, 5, 3, 22, 22)},
          {'userid': '0022',
           'movieid': '0377',
           'rating': '1',
           'timestamp': datetime.datetime(1997, 11, 7, 15, 18, 36)}]
```

```
In [11]: #1.2 Open the Second file---u.item
         import os
         os.chdir("[redacted]")
         file2 = open([redacted], encoding='ISO-8859-1').readlines()
         uitem = [x.replace('\n','').split('|')[:] for x in file2[:]]
         for i in uitem:
             del i[3]

         #1.2 Rearrange the dataset, make a dictionary pairing variable name with the data lis
         VariableNames2 = ['movieid', 'title', 'release', 'url', 'unknown', 'Action', 'Adventu
                           'Children', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'F
                           'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Weste
         uitem = [dict(zip(VariableNames2,uitem[i])) for i in range(len(uitem))]

         #2.(b) Add leading zeros to the movieid
         for i in uitem:
             i['movieid'] = i['movieid'].zfill(4)
```

```
In [12]: #Print an example
         uitem[0]
```

```
Out[12]: {'movieid': '0001',
          'title': 'Toy Story (1995)',
          'release': '01-Jan-1995',
          'url': 'http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)',
          'unknown': '0',
          'Action': '0',
          'Adventure': '0',
          'Animation': '1',
          'Children': '1',
          'Comedy': '1',
          'Crime': '0',
          'Documentary': '0',
          'Drama': '0',
          'Fantasy': '0',
          'Film-Noir': '0',
          'Horror': '0',
          'Musical': '0',
          'Mystery': '0',
          'Romance': '0',
          'Sci-Fi': '0',
          'Thriller': '0',
          'War': '0',
          'Western': '0'}
```

###2. Remove movies with title = ‘unknown’.

```
In [13]: deletelist = []
         for i in uitem:
             if i['title'] == 'unknown':
                 deletelist.append(i)
                 uitem.remove(i)
         for m in udata:
             for q in deletelist:
                 if q['movieid'] == m['movieid']:
                     udata.remove(m)
```

###3.Find the average ratings and the number of reviews for all movies in u.item.

```
In [14]: #1. Divide the entire u.data dataset by distinct value of movieid
         from collections import defaultdict
         DividedList = defaultdict(list)
         for records in udata:
             DividedList[records['movieid']].append(records)
         DividedList = list(DividedList.items())
```

```
In [18]: #2. Find the average ratings and the number of reviews for all movies in u.item
         for records in DividedList:
             count = 0
             for item in records[1]:
                 count += int(item['rating'])
             NumberofReviews = len(records[1])
             AvgOfReviews = count/NumberofReviews
             for x in uitem:
                 if records[0] == x['movieid']:
                     x['AverageRating'] = AvgOfReviews
                     x['NumberofReviews'] = NumberofReviews

         #3. Print an example:
         uitem[0]
```

```
Out[18]: {'movieid': '0001',
          'title': 'Toy Story (1995)',
          'release': '01-Jan-1995',
          'url': 'http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)',
          'unknown': '0',
          'Action': '0',
          'Adventure': '0',
          'Animation': '1',
          'Children': '1',
          'Comedy': '1',
          'Crime': '0',
          'Documentary': '0',
          'Drama': '0',
          'Fantasy': '0',
          'Film-Noir': '0',
          'Horror': '0',
          'Musical': '0',
          'Mystery': '0',
          'Romance': '0',
          'Sci-Fi': '0',
          'Thriller': '0',
          'War': '0',
          'Western': '0',
          'AverageRating': 3.8783185840707963,
          'NumberofReviews': 452}
```

###4. Write a function to list the top n (e.g. 10) rated movies, title names and their number of reviews.

```
In [19]: def TopMovies(n ,uitem = uitem, DividedList=DividedList):
             from copy import deepcopy
             uitem1 = deepcopy(uitem)
             for records in DividedList:
                 count = 0
                 for item in records[1]:
                     count += int(item['rating'])
                 NumberofReviews = len(records[1])
                 AvgOfReviews = count/NumberofReviews
                 for x in uitem1:
                     if records[0] == x['movieid']:
                         x['AverageRating'] = AvgOfReviews
                         x['NumberofReviews'] = NumberofReviews
             uitem1.sort(key= lambda x:x['AverageRating'], reverse= True)
             print('The result is as followed:')
             for i in range(n):
                 print(i+1,': movieid:',uitem1[i]['movieid'],'   title names:',uitem1[i]['titl
                       '   Average Rating:',uitem1[i]['AverageRating'],'   Number of reviews::

         TopMovies(10, uitem, DividedList)
```

```
The result is as followed:
```

```
1 : movieid: 0814    title names: Great Day in Harlem, A (1994)    Average Rating: 5.0    Numbe
2 : movieid: 1122    title names: They Made Me a Criminal (1939)    Average Rating: 5.0    Numb
3 : movieid: 1189    title names: Prefontaine (1997)    Average Rating: 5.0    Number of review
4 : movieid: 1201    title names: Marlene Dietrich: Shadow and Light (1996)    Average Rating
5 : movieid: 1293    title names: Star Kid (1997)    Average Rating: 5.0    Number of reviews:
6 : movieid: 1467    title names: Saint of Fort Washington, The (1993)    Average Rating: 5.0
7 : movieid: 1500    title names: Santa with Muscles (1996)    Average Rating: 5.0    Number of
8 : movieid: 1536    title names: Aiqing wansui (1994)    Average Rating: 5.0    Number of revi
9 : movieid: 1599    title names: Someone Else's America (1995)    Average Rating: 5.0    Numbe
10 : movieid: 1653    title names: Entertaining Angels: The Dorothy Day Story (1996)    Averag
```

###5. Considering that a movie with a higher number of reviews should have given a higher weight, we adjust the average rating formula by incorporating c hypothetical users. These users rate each movie with rating m. Use c = 59 and m = 3, write a function to list the top n rated movies, title names and their number of reviews using the adjusted average formula. Compare the listing with that found in question 4. Which one is more reasonable?

```
In [20]: def TopMovies_Adjusted(n, c, m, uitem = uitem, DividedList=DividedList):
             from copy import deepcopy
             uitem1 = deepcopy(uitem)
             for records in DividedList:
                 count = 0
                 for item in records[1]:
                     count += int(item['rating'])
                 NumberofReviews = len(records[1]) + c
                 AvgOfReviews = (count+c*m)/NumberofReviews
                 for x in uitem1:
                     if records[0] == x['movieid']:
                         x['AverageRating'] = AvgOfReviews
                         x['NumberofReviews'] = NumberofReviews
             uitem1.sort(key= lambda x:x['AverageRating'], reverse= True)
             print('The result is as followed:')
             for i in range(n):
                 print(i+1,': movieid:',uitem1[i]['movieid'],'   title names:',uitem1[i]['title
                       '      Average Rating:',uitem1[i]['AverageRating'],'     Real Number of
                 #id = uitem1[i]['movieid']; title = uitem1[i]['title']; Avg = uitem1[i]['Avera
                 #print(f'{i+1:2}  movieid:{id:6}    title names:{title:50}      Average Ratin

         TopMovies_Adjusted(10, 59, 3, uitem, DividedList)
```

```
The result is as followed:
1 : movieid: 0050    title names: Star Wars (1977)       Average Rating: 4.233644859813084
2 : movieid: 0318    title names: Schindler's List (1993)       Average Rating: 4.22408963585
3 : movieid: 0064    title names: Shawshank Redemption, The (1994)       Average Rating: 4.195
4 : movieid: 0483    title names: Casablanca (1942)       Average Rating: 4.172185430463577
5 : movieid: 0012    title names: Usual Suspects, The (1995)       Average Rating: 4.13496932
6 : movieid: 0127    title names: Godfather, The (1972)       Average Rating: 4.12288135593220
```

```
7 : movieid: 0098    title names: Silence of the Lambs, The (1991)       Average Rating: 4.120
8 : movieid: 0174    title names: Raiders of the Lost Ark (1981)       Average Rating: 4.09812
9 : movieid: 0603    title names: Rear Window (1954)       Average Rating: 4.082089552238806
10 : movieid: 0313    title names: Titanic (1997)       Average Rating: 4.06601466992665
```

####Answer: Comparing to the result of question 4, the adjusted result of question 5 is more reasonable and more fair.

###6. For two distinct users A and B, find the set of movies common to both users, that is the set of movies both users have given ratings. Apply the Euclidean distance formula on the two sets of ratings to determine a "distance" between user A and user B. Write a distance function with userid of A and userid of B as input. The output of the function is 1/(1+d(A,B)), where d(A,B) is the distance between user A and user B.

```
In [23]: # Divide the entire u.data dataset by distinct value of userid
         from collections import defaultdict
         DividedUser = defaultdict(list)
         for records in udata:
             DividedUser[records['userid']].append(records)
         DividedUser = list(DividedUser.items())
```

```
In [24]: # Write a distance function with userid of A and userid of B as input
         def DistanceA_and_B (A, B, DividedUser = DividedUser):
             UserList = [A,B]
             #Define a dict containing all the movieid and corresponding rating made by user a
             from collections import defaultdict
             CommomMovie = defaultdict(list)
             for records in DividedUser:
                 for user in UserList:
                     for item in records[1]:
                         if records[0] == user:
                             CommomMovie[item['movieid']].append(int(item['rating']))
             #Delete the movie contaning only one rating
             for key in list(CommomMovie.keys()):
                 if len(CommomMovie[key]) == 1:
                     del CommomMovie[key]
             #Calculate the Euclidean distance and the output(function defined in question 6)
             summation = 0
             for key in list(CommomMovie.keys()):
                 CommomMovie[key].append(pow((CommomMovie[key][0]-CommomMovie[key][1]),2))
                 summation += CommomMovie[key][2]
             distance = 1/(1+ pow(summation,1/2))
             return distance
```

```
In [25]: # Print an example
         DistanceA_and_B('0717','0427',DividedUser)
```

```
Out[25]: 0.2240092377397959
```

###7. Given a user, write a function to determine and output a list of distances between the given user and others. Mark the distances with their users.

```
In [26]: #Write a distance function with userid of userA as input and the distance list as out
         def DistanceA_and_Others (userA, DividedUser = DividedUser):
             #Define a dict containing all the common movieid and corresponding rating made by
             DistanceList = {}
             idlist = []
             for x in DividedUser:
                 if x[0] != userA:
                     idlist.append(x[0])
             for userB in idlist:
                 UserList = [userA,userB]
                 #Define a dict containing all the movieid and corresponding rating made by us
                 from collections import defaultdict
                 CommomMovie = defaultdict(list)
                 for records in DividedUser:
                     for user in UserList:
                         for item in records[1]:
                             if records[0] == user:
                                 CommomMovie[item['movieid']].append(int(item['rating']))
                 #Delete the movie contaning only one rating
                 for key in list(CommomMovie.keys()):
                     if len(CommomMovie[key]) == 1:
                         del CommomMovie[key]
                 #Calculate the Euclidean distance and the output(function defined in question
                 summation = 0
                 for key in list(CommomMovie.keys()):
                     CommomMovie[key].append(pow((CommomMovie[key][0]-CommomMovie[key][1]),2))
                     summation += CommomMovie[key][2]
                 distance = 1/(1+ pow(summation,1/2))
                 #Define the Distancelist. Key: other distinct userid as key Value: distance b
                 DistanceList[userB] = distance
             return DistanceList
```

```
In [27]: #Print an example
         ExampleList = DistanceA_and_Others('0001', DividedUser)
         ExampleList = dict(sorted(ExampleList.items()))
         KeyList = list(ExampleList.keys())
         print('The first 10 rows of distance list of is:')
         for i in range(10):
             print(i+1, 'userid:', KeyList[i], 'Distance:', ExampleList[KeyList[i]])
```

```
The first 10 rows of distance list of is:
1 userid: 0002 Distance: 0.16139047779640892
2 userid: 0003 Distance: 0.1639607805437114
3 userid: 0004 Distance: 0.18660549686337075
4 userid: 0005 Distance: 0.07595595279317306
```

```
5 userid: 0006 Distance: 0.07284423640594942
6 userid: 0007 Distance: 0.05564656009922349
7 userid: 0008 Distance: 0.1566130288262323
8 userid: 0009 Distance: 0.3090169943749474
9 userid: 0010 Distance: 0.09535767393825997
10 userid: 0011 Distance: 0.08462632608958592
```

###8. Write a function with a given user and a given movie. If the movie was rated by the user, output the rating provided. If the movie was not rated by the user, output the weighted average of the ratings of all other users weighted by their distances with the given user.

```
In [38]: #Write a function outputing ratings by given user and movieid
         def Rating(user, movie, DividedUser = DividedUser):
             Wei_AvgRating = []
             for records in DividedUser:
                 for item in records[1]:
                     if records[0] == user:
                         if item['movieid'] == movie:
                             Wei_AvgRating.append(int(item['rating']))
                             return sum(Wei_AvgRating)
                             break
                     if item['movieid'] == movie:
                         Wei_AvgRating.append(int(item['rating']) * DistanceA_and_B(user,item[
             return sum(Wei_AvgRating)#/len(Wei_AvgRating)
```

```
In [39]: #Print an Example
         print('The rating result is:', Rating('0717','0427',DividedUser))
```

```
The rating result is: 152.7834819947832
```

###9. Hence, given a user, write a function to suggest 10 movies.

Note: I want to clarify the answer in small question 8 and 9 of question 2 ####(1) At small question 8, I did not divide the rating for movie by the number of viewer, thus the number is quite large. Therefore, in small question 9, I did not delete the list of movie the given user had reviewed, but directly using the rating result from question 8. ####(2) That is, another type of answer for small question 8 is that using the final result to divide the number of reviews. Then, in small question 9, first deleting the list of movie the given user had reviewed, then suggesting 10 movies.

```
In [50]: #Write a function outputing suggest 10 movies by given user
         def SuggestedMovies(user, n, DividedUser = DividedUser):
             SuggestedMoviesList = {}
             for item in uitem:
                 rating = Rating(user,item['movieid'],DividedUser)
                 SuggestedMoviesList[item['movieid']] = rating
             SuggestedMoviesList = sorted(SuggestedMoviesList.items(), key= lambda x:x[1], reve
             print('The suggested movies are as followed:')
             for i in range(n):
                 print(i+1, ':  Movieid:',SuggestedMoviesList[i][0],'   Title:',uitem[int(Sugge
```

```
In [51]: SuggestedMovies('0001',10, DividedUser = DividedUser)
```

```
The suggested movies are as followed:
1 :  Movieid: 0286    Title: English Patient, The (1996)     Rating: 266.1008395038215
2 :  Movieid: 0300    Title: Air Force One (1997)     Rating: 235.11473864680806
3 :  Movieid: 0288    Title: Scream (1996)     Rating: 233.04701899660108
4 :  Movieid: 0313    Title: Titanic (1997)     Rating: 216.7316450557611
5 :  Movieid: 0294    Title: Liar Liar (1997)     Rating: 210.80016303667088
6 :  Movieid: 0302    Title: L.A. Confidential (1997)     Rating: 185.7539664152498
7 :  Movieid: 0328    Title: Conspiracy Theory (1997)     Rating: 149.39945140882068
8 :  Movieid: 0748    Title: Saint, The (1997)     Rating: 146.06076743769967
9 :  Movieid: 0333    Title: Game, The (1997)     Rating: 136.8012078216705
10 :  Movieid: 0276    Title: Leaving Las Vegas (1995)     Rating: 124.11467803904522
```