RASHI, 11910010

Big DATA Management Assignment

Submitted to:

Peeyush Taori

1. By making use of Spark Core (i.e. without using Spark SQL) find out:

a) Count of unique locations where each product is sold.
b) Find out products bought by each user.
c) Total spending done by each user on each product.

```
import pyspark
import pandas as pd
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

conf = pyspark.SparkConf().setAppName('appName').setMaster('local')
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession(sc)

#Reading "/Users/rashi/Downloads/users.csv and transactions.csv"

file1 =sc.textFile("file:///Users/rashi/Desktop/transactions.csv")
file2 =sc.textFile("file:///Users/rashi/Desktop/users.csv")
file1.collect()
```

```
In [3]:
file1.collect()

Out[3]: ['1,1004,19,129,whatchamacallit',
 '2,1001,10,99,thingamajig',
 '3,1004,17,129,whatchamacallit',
 '4,1001,9,99,thingamajig',
 '5,1003,3,89,gadget',
 '6,1002,19,149,gizmo',
 '7,1002,30,149,gizmo',
 '8,1002,26,149,gizmo',
 '9,1001,22,99,thingamajig',
 '10,1003,6,89,gadget',
 '11,1004,1,129,whatchamacallit',
 '12,1004,2,129,whatchamacallit',
 '13,1005,5,199,doohickey',
 '14,1004,7,129,whatchamacallit',
 '15,1002,16,149,gizmo']
```

a) Count of unique locations where each product is sold

```
#Collecting productname and it's UserID from transactions.csv
data = file1.map(lambda x: (x.split(',')[4], x.split(',')[2]) ).collect()
print(data)
```

```
In [4]:
#Collecting productname and it's UserID from transactions.csv
data = file1.map(lambda x: (x.split(',')[4], x.split(',')[2]) ).collect()
print(data)

[('whatchamacallit', '19'), ('thingamajig', '10'), ('whatchamacallit', '17'), ('thingamajig',
'9'), ('gadget', '3'), ('gizmo', '19'), ('gizmo', '30'), ('gizmo', '26'), ('thingamajig', '2
2'), ('gadget', '6'), ('whatchamacallit', '1'), ('whatchamacallit', '2'), ('doohickey', '5'),
('whatchamacallit', '7'), ('gizmo', '16')]
```

```
print(file2.collect())
data2 = file2.map(lambda x: (x.split(',')[0],x.split(',')[3])).collect()
print(data2)
```

In [5]:

```
print(file2.collect())
data2 = file2.map(lambda x: (x.split(',')[0],x.split(',')[3])).collect()
print(data2)
```

```
['1,user1@company.com,ES,MX', '2,user4@domain.com,EN,US', '3,user5@company.com,FR,FR', '4,use
r9@site.org,HI,IN', '5,user12@service.io,EN,CA', '6,user17@website.net,FR,FR', '7,user21@comp
any.com,FR,FR', '8,user25@company.com,FR,FR', '9,user27@school.edu,ES,MX', '10,user31@websit
e.net,EN,CA', '11,user36@website.net,FR,FR', '12,user39@domain.com,FR,FR', '13,user41@compan
y.com,ES,MX', '14,user45@domain.com,HI,IN', '15,user48@site.org,ES,MX', '16,user53@school.ed
u,EN,US', '17,user57@school.edu,ES,MX', '18,user59@website.net,HI,IN', '19,user64@school.edu,
EN,US', '20,user67@domain.com,HI,IN', '21,user68@site.org,EN,US', '22,user71@domain.com,ES,M
X', '23,user74@service.io,EN,US', '24,user79@website.net,ES,MX', '25,user81@site.org,EN,US',
'26,user85@service.io,HI,IN', '27,user89@service.io,EN,CA', '28,user91@company.com,EN,CA', '2
9,user96@site.org,ES,MX', '30,user99@website.net,EN,US']
[('1', 'MX'), ('2', 'US'), ('3', 'FR'), ('4', 'IN'), ('5', 'CA'), ('6', 'FR'), ('7', 'FR'),
('8', 'FR'), ('9', 'MX'), ('10', 'CA'), ('11', 'FR'), ('12', 'FR'), ('13', 'MX'), ('14', 'I
N'), ('15', 'MX'), ('16', 'US'), ('17', 'MX'), ('18', 'IN'), ('19', 'US'), ('20', 'IN'), ('2
1', 'US'), ('22', 'MX'), ('23', 'US'), ('24', 'MX'), ('25', 'US'), ('26', 'IN'), ('27', 'C
A'), ('28', 'CA'), ('29', 'MX'), ('30', 'US')]
```

```
my_dict = {}
for i in data:
    for j in data2:
        if(i[1] == j[0]):
            if(i[0] in my_dict.keys()):
                my_dict[i[0]].add(j[1])
            else:
                temp = set()
                temp.add(j[1])
                my_dict[i[0]]=temp

print(my_dict)
#took a set inside a dictionary. Product name is the key for the dictionary and it's value is set.

print('Product         Unique location Count')
for i in my_dict.keys():
    print(i,'         ',len(my_dict[i]))
```

OUTPUT on NEXT PAGE:

```
{'whatchamacallit': {'MX', 'US', 'FR'}, 'thingamajig': {'MX', 'CA'}, 'gadget': {'FR'}, 'gizm
o': {'US', 'IN'}, 'doohickey': {'CA'}}
```

a) Count of unique locations where each product is sold

```
print('Product          Unique location Count')
for i in my_dict.keys():
    print(i,'          ',len(my_dict[i]))
```

```
Product          Unique location Count
whatchamacallit            3
thingamajig             2
gadget           1
gizmo            2
doohickey            1
```

b) Find out products bought by each user.

```
user_dict = {}
for i in data:
    if(i[1] in user_dict.keys()):
        user_dict[i[1]].append(i[0])
    else:
        temp = []
        temp.append(i[0])
        user_dict[i[1]]=temp

print('CustomerID    Products')
for i in user_dict.keys():
    print(i,'        ',user_dict[i])
```

OUTPUT:

```
CustomerID    Products
19           ['whatchamacallit', 'gizmo']
10           ['thingamajig']
17           ['whatchamacallit']
9           ['thingamajig']
3           ['gadget']
30           ['gizmo']
26           ['gizmo']
22           ['thingamajig']
6           ['gadget']
1           ['whatchamacallit']
2           ['whatchamacallit']
5           ['doohickey']
7           ['whatchamacallit']
16           ['gizmo']
```

c) Total spending done by each user on each product.

```
data3 = file1.map(lambda x: (x.split(',')[2], x.split(',')[3]) ).collect()
print(data3)

pr_dict = {}
for i in data3:
    if(i[0] in pr_dict.keys()):
        pr_dict[i[0]] += int(i[1])
    else:
        pr_dict[i[0]] = int(i[1])

print('CustomerID   Total Cost')
for i in pr_dict.keys():
    print(i,'        ',pr_dict[i])
```

OUTPUT

```
CustomerID   Total Cost
19             278
10             99
17             129
9             99
3             89
30             149
26             149
22             99
6             89
1             129
2             129
5             199
7             129
16             149
```

Q2. Consider the dataset file Olympics.csv. This file contains information about the Olympic games, players participating in the games, and details of medals won by them. Using Spark core and the data file, compute the following:

1- Total medals that each country won in a particular sport (such as Gymnastics).

2- In each Olympic games, how many medals has India won?

3- Compute top 3 countries in terms of total medals by each Olympic games year.

```
file3 =sc.textFile("file:///Users/rashi/Desktop/olympics.csv")
file3.collect()
```

```
5]: ['Michael Phelps\t23\tUnited States\t2008\t08-24-08\tSwimming\t8\t0\t0\t8',
     'Michael Phelps\t19\tUnited States\t2004\t08-29-04\tSwimming\t6\t0\t2\t8',
     'Michael Phelps\t27\tUnited States\t2012\t08-12-12\tSwimming\t4\t2\t0\t6',
     'Natalie Coughlin\t25\tUnited States\t2008\t08-24-08\tSwimming\t1\t2\t3\t6',
     'Aleksey Nemov\t24\tRussia\t2000\t10-01-00\tGymnastics\t2\t1\t3\t6',
     'Alicia Coutts\t24\tAustralia\t2012\t08-12-12\tSwimming\t1\t3\t1\t5',
     'Missy Franklin\t17\tUnited States\t2012\t08-12-12\tSwimming\t4\t0\t1\t5',
     'Ryan Lochte\t27\tUnited States\t2012\t08-12-12\tSwimming\t2\t2\t1\t5',
     'Allison Schmitt\t22\tUnited States\t2012\t08-12-12\tSwimming\t3\t1\t1\t5',
     'Natalie Coughlin\t21\tUnited States\t2004\t08-29-04\tSwimming\t2\t2\t1\t5',
     'Ian Thorpe\t17\tAustralia\t2000\t10-01-00\tSwimming\t3\t2\t0\t5',
     'Dara Torres\t33\tUnited States\t2000\t10-01-00\tSwimming\t2\t0\t3\t5',
     'Cindy Klassen\t26\tCanada\t2006\t02-26-06\tSpeed Skating\t1\t2\t2\t5',
     'Nastia Liukin\t18\tUnited States\t2008\t08-24-08\tGymnastics\t1\t3\t1\t5',
     'Marit Bjørgen\t29\tNorway\t2010\t02-28-10\tCross Country Skiing\t3\t1\t1\t5',
     'Sun Yang\t20\tChina\t2012\t08-12-12\tSwimming\t2\t1\t1\t4',
     'Kirsty Coventry\t24\tZimbabwe\t2008\t08-24-08\tSwimming\t1\t3\t0\t4',
     'Libby Lenton-Trickett\t23\tAustralia\t2008\t08-24-08\tSwimming\t2\t1\t1\t4',
     'Ryan Lochte\t24\tUnited States\t2008\t08-24-08\tSwimming\t2\t0\t2\t4',
```

1- Total medals that each country won in a particular sport (such as Gymnastics).

```
data4 = file3.map(lambda x: (x.split('\t')[2], x.split('\t')[5], x.split('\t')[9])).collect()
print(data4)
```

```
[('United States', 'Swimming', '8'), ('United States', 'Swimming', '8'), ('United States', 'S
wimming', '6'), ('United States', 'Swimming', '6'), ('Russia', 'Gymnastics', '6'), ('Australi
a', 'Swimming', '5'), ('United States', 'Swimming', '5'), ('United States', 'Swimming', '5'),
('United States', 'Swimming', '5'), ('United States', 'Swimming', '5'), ('Australia', 'Swimmi
ng', '5'), ('United States', 'Swimming', '5'), ('Canada', 'Speed Skating', '5'), ('United Sta
tes', 'Gymnastics', '5'), ('Norway', 'Cross Country Skiing', '5'), ('China', 'Swimming',
'4'), ('Zimbabwe', 'Swimming', '4'), ('Australia', 'Swimming', '4'), ('United States', 'Swimm
ing', '4'), ('Netherlands', 'Swimming', '4'), ('Australia', 'Swimming', '4'), ('Australia',
'Swimming', '4'), ('Netherlands', 'Swimming', '4'), ('United States', 'Swimming', '4'), ('Aus
tralia', 'Swimming', '4'), ('Australia', 'Swimming', '4'), ('United States', 'Swimming',
'4'), ('Netherlands', 'Swimming', '4'), ('South Korea', 'Short-Track Speed Skating', '4'),
('Russia', 'Gymnastics', '4'), ('United States', 'Gymnastics', '4'), ('Russia', 'Diving',
'4'), ('Netherlands', 'Cycling', '4'), ('Norway', 'Cross Country Skiing', '4'), ('Norway', 'B
iathlon', '4'), ('Croatia', 'Alpine Skiing', '4'), ('United States', 'Swimming', '3'), ('Fran
ce', 'Swimming', '3'), ('Australia', 'Swimming', '3'), ('United States', 'Swimming', '3'),
('Japan', 'Swimming', '3'), ('United States', 'Swimming', '3'), ('Netherlands', 'Swimming',
'3'), ('France', 'Swimming', '3'), ('Australia', 'Swimming', '3'), ('Australia', 'Swimming',
'3'), ('United States', 'Swimming', '3'), ('Japan', 'Swimming', '3'), ('United States', 'Swim
ming', '3'), ('France', 'Swimming', '3'), ('Hungary', 'Swimming', '3'), ('United States', 'Sw
```

```
medal_dict = {}
for i in data4:
    if(i[0] in medal_dict.keys()):
        if(i[1] in medal_dict[i[0]]):
            medal_dict[i[0]][i[1]] += int(i[2])
        else:
            medal_dict[i[0]][i[1]] = int(i[2])
    else:
        sport = {}
        sport[i[1]] = int(i[2])
        medal_dict[i[0]]= sport

print('Country          Sports          Total Medals')
for i in medal_dict.keys():
    for j in medal_dict[i].keys():
        print(i,'    ',j,'          ',medal_dict[i][j])
```

OUTPUT ON NEXT PAGE:

Country	Sports	Total Medals
United States	Swimming	267
United States	Gymnastics	55
United States	Short-Track Speed Skating	23
United States	Speed Skating	22
United States	Nordic Combined	7
United States	Athletics	147
United States	Alpine Skiing	12
United States	Tennis	16
United States	Synchronized Swimming	11
United States	Fencing	19
United States	Equestrian	30
United States	Diving	8
United States	Cycling	18
United States	Archery	7
United States	Wrestling	20
United States	Weightlifting	2
United States	Waterpolo	63
United States	Volleyball	36
United States	Triathlon	1
United States	Taekwondo	8
United States	Softball	45
United States	Snowboarding	17
United States	Skeleton	3
United States	Shooting	16
United States	Sailing	14
United States	Rowing	60
United States	Modern Pentathlon	1
United States	Luge	4
United States	Judo	4
United States	Ice Hockey	106
United States	Football	64
Russia	Gymnastics	47
Russia	Diving	25
Russia	Cross Country Skiing	21
Russia	Biathlon	27
Russia	Synchronized Swimming	43
Russia	Swimming	20
Russia	Speed Skating	9
Russia	Shooting	21
Russia	Cycling	15
Russia	Canoeing	16
Russia	Athletics	98
Russia	Wrestling	41
Russia	Weightlifting	24
Russia	Waterpolo	38
Russia	Volleyball	70
Russia	Trampoline	4
Russia	Taekwondo	3
Russia	Tennis	8
Russia	Snowboarding	1
Russia	Skeleton	1
Russia	Rowing	8
Russia	Rhythmic Gymnastics	31
Russia	Modern Pentathlon	3
Russia	Luge	1
Russia	Judo	13
Russia	Ice Hockey	21
Russia	Handball	43
Russia	Figure Skating	16
Russia	Freestyle Skiing	1
Russia	Fencing	32
Russia	Boxing	22
Russia	Bobsleigh	6
Russia	Basketball	36

```
Russia      Archery              1
Australia      Swimming            163
Australia      Rowing             61
Australia      Equestrian             10
Australia      Diving             17
Australia      Cycling             36
Australia      Canoeing             19
Australia      Athletics              16
Australia      Waterpolo              39
Australia      Triathlon              5
Australia      Trampoline              1
Australia      Taekwondo              2
Australia      Tennis             3
Australia      Short-Track Speed Skating              1
Australia      Softball              45
Australia      Snowboarding               1
Australia      Shooting              6
Australia      Sailing             21
Australia      Judo             1
Australia      Hockey             81
Australia      Freestyle Skiing                5
Australia      Baseball             24
Australia      Beach Volleyball                2
Australia      Basketball               48
Australia      Archery              2
Canada      Speed Skating               26
Canada      Short-Track Speed Skating             38
Canada      Equestrian              5
Canada      Diving            12
Canada      Canoeing             10
Canada      Wrestling              6
Canada      Weightlifting               1
```

Please run the python file to see the whole output

2- In each Olympic games, how many medals has India won?

```
data5 = file3.map(lambda x: (x.split('\t')[3], x.split('\t')[2], x.split('\t')[9])).collect()

in_med_dict = {}
for i in data5:
    if(i[1]=='India'):
        if(i[0] in in_med_dict.keys()):
            in_med_dict[i[0]] += int(i[2])
        else:
            in_med_dict[i[0]] = int(i[2])

print('Year    India Medals')
for i in in_med_dict.keys():
    print(i,'        ',in_med_dict[i])
```

```
Year     India Medals
2012             6
2008             3
2000             1
2004             1
```

3- Compute top 3 countries in terms of total medals by each Olympic games year.

```
from heapq import nlargest

top3_dict = {}
for i in data5:
    if(i[0] in top3_dict.keys()):
        if(i[1] in top3_dict[i[0]]):
            top3_dict[i[0]][i[1]] += int(i[2])
        else:
            top3_dict[i[0]][i[1]] = int(i[2])
    else:
        country = {}
        country[i[1]] = int(i[2])
        top3_dict[i[0]]= country

for i in top3_dict.keys():
    #nlargest is the function of heapq, used for finding the 3 largest skillsets required by company
    threeHighest = nlargest(3, top3_dict[i], key = top3_dict[i].get)
    print("Year:", i)
    for val in threeHighest:
        print('\t',val, ":", top3_dict[i].get(val))

sc.stop()
```

```
Year: 2008
         United States : 317
         China : 184
         Australia : 149
Year: 2004
         United States : 265
         Russia : 191
         Australia : 156
Year: 2012
         United States : 254
         Russia : 140
         Great Britain : 126
Year: 2000
         United States : 243
         Russia : 187
         Australia : 183
Year: 2006
         Canada : 69
         Sweden : 64
         Germany : 54
Year: 2010
         United States : 97
         Canada : 90
         Germany : 54
Year: 2002
         United States : 84
         Canada : 74
         Germany : 61
```

Q3. Consider the Movie Recommendation code and problem that was discussed during the class (Session 5). Please provide a brief write-up on the problem, steps needed to arrive at the solution (recommendation system), and how exactly those steps are implemented in the code. You can make use of the PPT file that discusses the broad solution. While you are doing so, please also mention what each line of code does (It is not sufficient to mention what each block of code does, you would have to provide explanation for each line)

Brief write-up:

In the movie recommedation system, we want to recommend movies to user on the basis of what he rated in the past. We will look into user's history and see, how much rating he gave to movies.
We will recommend him movies of same ratings(No matter good or bad)

Steps to solve our problem:

- Read the data from users file and map it like users, (movies, ratings)
  In the user database, we will see the movies rated by particular user and extract it.

- Then, combine the movies with their ratings for the particular user.
  Ex: if a user rated, movie1 with rating1 and movie2 with rating2. So, we will form a tupple for each user, like—— [user1,(movie1 , rating1)]

- Remove duplicate ratings if present.

- Now, join them like this (movie1,movie2),(rating1,rating2)) for same user.

- Now, Group it like, What ratings given to these two movies by N users.
  Ex: -((m1,m2),((r1,r2),(r1,r2),…,(r1,r2)))

- Calculate Statistical similarity between the above ratings.We can use correlation but In the code, we are using cosineSimilarity.

- Filter and sort top 10 movies requested by user.

DESCRIPTION BY CODE:

```
import sys
from pyspark import SparkConf, SparkContext
from math import sqrt


def loadMovieNames():  #function to load the movies names
    movieNames = {}    #Empty dictioary of movieNames
    with open("/home/cloudera/moviedata/itemfile.txt") as f:
        #open the itemfile.txt as f
        for line in f:  #going in each line of file f
            fields = line.split('|')
             #splitting the fields on the basis of '|'
             movieNames[int(fields[0])] = fields[1].decode('ascii',
'ignore')
#Storing  the  value  of  fields[1]  in  the  key(fields[0])  of
dictionary movieNames
    return movieNames #returning movieNames dictionary


#-------------------------------------------------------------------
-----------------------------------------


def makePairs((user, ratings)):
    (movie1, rating1) = ratings[0]
#ratings is the dictionary, storing the key as movie1 and value as
rating1
     (movie2, rating2) = ratings[1] #storing the key as movie2 and
value as rating2
    return ((movie1, movie2), (rating1, rating2))
#Now, making pair of movie1 and movie2 as key and rating1, rating2
as it's value


#-------------------------------------------------------------------
-----------------------------------------


# You might have the same two movies joined in both orders, or a
movie joined to itself, for example:
# (user, ((100,5),(200,4)))
# (user, ((200,4),(100,5)))
# Remove duplicates by filtering for the condition where movie1 <
movie2
def filterDuplicates( (userID, ratings) ):
    (movie1, rating1) = ratings[0] #movie 1 with it's rating
    (movie2, rating2) = ratings[1] #movie 2 with it's rating
    return movie1 < movie2
    #Filtering after comparison of movie1<movie2
```

```
#-------------------------------------------------------------------
----------------------------------------


# Cosine similarity is a metric used to measure how similar the
documents are irrespective of their size.
# Mathematically, it measures the cosine of the angle between two
vectors projected in a multi-dimensional space
def computeCosineSimilarity(ratingPairs):
    numPairs = 0
    sum_xx = sum_yy = sum_xy = 0
    for ratingX, ratingY in ratingPairs:
#In the rating Pairs, we have pairs of ratings. Here we are
traversing those ratings in the pair as Rating x and rating y.
        sum_xx += ratingX * ratingX
#adding the multiplication of ratingX every time in sum_xx
        sum_yy += ratingY * ratingY
 #adding the multiplication of ratingY every time in sum_yy
         sum_xy += ratingX * ratingY    #adding the multiplication
of ratingX and ratingY every time in sum_xy
        numPairs += 1                #increment the count of Pairs

    numerator = sum_xy        #taking value of sum_xx as numerator
    denominator = sqrt(sum_xx) * sqrt(sum_yy)
#(squareroot of sum_xx)*(squareroot of sum_yy) as denominator

    score = 0   #initially score is 0
    if (denominator):
#when denominator is not 0, it will go inside if
        score = (numerator / (float(denominator)))
#Now calculating score by dividing numerator by denominatr

    return (score, numPairs)
#returning the score with the count of pairs


#-------------------------------------------------------------------
----------------------------------------


conf                                                              =
SparkConf().setMaster("local[*]").setAppName("MovieSimilarities")
#To run a Spark application on the local/cluster, we need to set a
few configurations and parameters, this is what SparkConf helps
with. It provides configurations to run a Spark application.
#Initially, we will create a SparkConf object with SparkConf(),
which will load the values from spark.* Java system properties as
well. Now we can set different parameters using the SparkConf
object and their parameters will take priority over the system
properties.
```

```
#setMaster(value) - To set the master URL.
#setAppName(value) - To set an application name.

sc = SparkContext(conf = conf)

print "\nLoading movie names..."
nameDict = loadMovieNames()  #calling loadMovieNames function that
will return a dictionary of id's and MovieNames

data        =        sc.textFile("file:///home/cloudera/moviedata/
datafile2.txt") #Reading the textfile datafile2

ratings = data.map(lambda l: l.split()).map(lambda l: (int(l[0]),
(int(l[1]), float(l[2]))))
# There will be two integer and 1 float in a row of file
datafile2.txt. For extracting this data.map(lambda l: l.split()),
we are using the map function to split the line into words, This
will return a list of words and
#then map(lambda l: (int(l[0]), (int(l[1]), float(l[2]), again
applying to read the words and convert them to respective formats.
This will return userID, MovieID and Ratings

joinedRatings = ratings.join(ratings)
#Join the ratings of movies done by same user

uniqueJoinedRatings    =    joinedRatings.filter(filterDuplicates)
#filter the duplicates

moviePairs = uniqueJoinedRatings.map(makePairs) #making pair of
movies and ratings done by same user.

moviePairRatings = moviePairs.groupByKey() #grouping by pair of
movies

moviePairSimilarities                                          =
moviePairRatings.mapValues(computeCosineSimilarity).cache()
#finding the cosine similarity for the movie pairs based on the
ratings given by multiple users

if (len(sys.argv) > 1):

    scoreThreshold = 0.10 #assigning scoreThreshold = 0.10
    coOccurenceThreshold = 2 #assigning coOccurenceThreshold = 2

    movieID = int(sys.argv[1])  #typcasting sys.argv[1] in int and
then storing it in movieID

                                        filteredResults          =
moviePairSimilarities.filter(lambda((pair,sim)): \ #filtering the
movie on the basis
                (pair[0]  ==  movieID  or  pair[1]  ==  movieID)  \
#of similarities.
```

```
                    and sim[0] > scoreThreshold and sim[1] >
coOccurenceThreshold)       ## If similarity is greater than
score threshold than only we will recommend that movie

        results = filteredResults.map(lambda((pair,sim)): (sim,
pair)).sortByKey(ascending = False).take(10)
    #sorting the similarity with movie pairs in descending order

       print "Top 10 similar movies for " + nameDict[movieID]
#printing top10 movies using loop
    for result in results:
        (sim, pair) = result
        similarMovieID = pair[0]
        if (similarMovieID == movieID):
            similarMovieID = pair[1]
        print nameDict[similarMovieID] + "\tscore: " + str(sim[0])
+ "\tstrength: " + str(sim[1])
```