**BDM Homework Assignment**

**Answer 1**

Loading the files,

Launching commands,

PySpark

usersRDD=sc.textFile(“file//:cloudera/Desktop/Users.csv”)

transRDD=sc.textFile(“file//:cloudera/Desktop/transactions.csv”)

Converting them to paired RDD’s to perform join

pairedUsers=usersRDD.map(lambda x:(x.split(“,”)[0],x))

pairedUsers.collect()

**Output:**

[(u'1', u'1,user1@company.com,ES,MX'), (u'2', u'2,user4@domain.com,EN,US'), (u'3', u'3,user5@company.com,FR,FR'), (u'4', u'4,user9@site.org,HI,IN'), (u'5', u'5,user12@service.io,EN,CA'), (u'6', u'6,user17@website.net,FR,FR'), (u'7', u'7,user21@company.com,FR,FR'), (u'8', u'8,user25@company.com,FR,FR'), (u'9', u'9,user27@school.edu,ES,MX'), (u'10', u'10,user31@website.net,EN,CA'), (u'11', u'11,user36@website.net,FR,FR'), (u'12', u'12,user39@domain.com,FR,FR'), (u'13', u'13,user41@company.com,ES,MX'), (u'14', u'14,user45@domain.com,HI,IN'), (u'15', u'15,user48@site.org,ES,MX'), (u'16', u'16,user53@school.edu,EN,US'), (u'17', u'17,user57@school.edu,ES,MX'), (u'18', u'18,user59@website.net,HI,IN'), (u'19', u'19,user64@school.edu,EN,US'), (u'20', u'20,user67@domain.com,HI,IN'), (u'21', u'21,user68@site.org,EN,US'), (u'22', u'22,user71@domain.com,ES,MX'), (u'23', u'23,user74@service.io,EN,US'), (u'24', u'24,user79@website.net,ES,MX'), (u'25', u'25,user81@site.org,EN,US'), (u'26', u'26,user85@service.io,HI,IN'), (u'27', u'27,user89@service.io,EN,CA'), (u'28', u'28,user91@company.com,EN,CA'), (u'29', u'29,user96@site.org,ES,MX'), (u'30', u'30,user99@website.net,EN,US')]

pairedTrans=transRDD.map(lambda x:(x.split(“,”)[2],x))

pairedTrans.collect()

**Output:**

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

Performing the join in order to solve the given queries,

resultsRDD=pairedUsers.join(pairedTrans)

**Output:**

[(u'19', (u'19,user64@school.edu,EN,US', u'1,1004,19,129,whatchamacallit')), (u'19', (u'19,user64@school.edu,EN,US', u'6,1002,19,149,gizmo')), (u'1', (u'1,user1@company.com,ES,MX', u'11,1004,1,129,whatchamacallit')), (u'5', (u'5,user12@service.io,EN,CA', u'13,1005,5,199,doohickey')), (u'9', (u'9,user27@school.edu,ES,MX', u'4,1001,9,99,thingamajig')), (u'10', (u'10,user31@website.net,EN,CA', u'2,1001,10,99,thingamajig')), (u'17', (u'17,user57@school.edu,ES,MX', u'3,1004,17,129,whatchamacallit')), (u'22', (u'22,user71@domain.com,ES,MX', u'9,1001,22,99,thingamajig')), (u'26', (u'26,user85@service.io,HI,IN', u'8,1002,26,149,gizmo')), (u'3', (u'3,user5@company.com,FR,FR', u'5,1003,3,89,gadget')), (u'7', (u'7,user21@company.com,FR,FR', u'14,1004,7,129,whatchamacallit')), (u'30', (u'30,user99@website.net,EN,US', u'7,1002,30,149,gizmo')), (u'16', (u'16,user53@school.edu,EN,US', u'15,1002,16,149,gizmo')), (u'2', (u'2,user4@domain.com,EN,US', u'12,1004,2,129,whatchamacallit')), (u'6', (u'6,user17@website.net,FR,FR', u'10,1003,6,89,gadget'))]

a)

uniqueCountries=resultsRDD.map(lambda (t,(u,v)):u.split(“,”)[3]).distinct()

uniqueCountries.collect()

**Output:**

[u'MX', u'CA', u'FR', u'US', u'IN']

uniqueCountries=resultsRDD.map(lambda (t,(u,v)):u.split(“,”)[3]).distinct().count()

print(uniqueCountries)

**Output:**

5

b)

productsByUsers= resultsRDD.map(lambda (t,(u,v)):(t,v.split(“,”)[4]))

**Output:**

[(u'19', u'whatchamacallit'), (u'19', u'gizmo'), (u'1', u'whatchamacallit'), (u'5', u'doohickey'), (u'9', u'thingamajig'), (u'10', u'thingamajig'), (u'17', u'whatchamacallit'), (u'22', u'thingamajig'), (u'26', u'gizmo'), (u'3', u'gadget'), (u'7', u'whatchamacallit'), (u'30', u'gizmo'), (u'16', u'gizmo'), (u'2', u'whatchamacallit'), (u'6', u'gadget')]

c)

Spend=resultsRDD.map(lambda (t,(u,v)):(t,v.split(“,”)[3],v.split(“,”)[4]))

**Output:**

[(u'19', u'129', u'whatchamacallit'), (u'19', u'149', u'gizmo'), (u'1', u'129', u'whatchamacallit'), (u'5', u'199', u'doohickey'), (u'9', u'99', u'thingamajig'), (u'10', u'99', u'thingamajig'), (u'17', u'129', u'whatchamacallit'), (u'22', u'99', u'thingamajig'), (u'26', u'149', u'gizmo'), (u'3', u'89', u'gadget'), (u'7', u'129', u'whatchamacallit'), (u'30', u'149', u'gizmo'), (u'16', u'149', u'gizmo'), (u'2', u'129', u'whatchamacallit'), (u'6', u'89', u'gadget')]

**Answer 2**

a)

tweets=sqlContext.read.json(<file:///home/cloudera/Desktop/tweets.json>)

tweets.registerTempTable(“tweets”)

tweets.printSchema()

b)

sqlContext.sql(“select \* from tweets where user=’Chase’”).show()

**Output:**

18/05/22 14:00:19 INFO scheduler.DAGScheduler: Job 43 finished: showString at NativeMethodAccessorImpl.java:-2, took 0.809608 s

+-------------+------------------+--------------+--------------------+-----+

| country| id| place| text| user|

+-------------+------------------+--------------+--------------------+-----+

|United States|572616165786185728| Wahpeton|Chicago takin ove...|Chase|

|United States|572577843428737026| Wahpeton|Music never annoy...|Chase|

|United States|572570237901537280|West Lafayette|@Bordenkircher13 ...|Chase|

+-------------+------------------+--------------+--------------------+-----+

c)

sqlContext.sql(“select user , count(id) as count from tweets group by user”).show()

**Output:**

18/05/22 14:18:54 INFO scheduler.DAGScheduler: Job 51 finished: showString at NativeMethodAccessorImpl.java:-2, took 0.111357 s

+-------------------+-----+

| user|count|

+-------------------+-----+

| вяεиdα| 1|

| Natty Dread| 1|

| Cat Woman| 1|

| goggle chroome| 1|

| s a v a n n a| 1|

|milena visnieski ☯ | 1|

| Virgie Riddick| 1|

| BIG COUNTRY J| 1|

| Ray Heard| 1|

| Raven Givens| 1|

| Tito Chase| 1|

| Filip| 1|

| Matthew Pham| 1|

| ilyse| 1|

| Jonah Kesoyan| 1|

| Benjamin Jr| 1|

| Myrna Cortés| 1|

| Last Poet| 1|

| CEB Jobs| 1|

| Cleveland\_bastard | 1|

+-------------------+-----+

only showing top 20 rows

d)

mentions=sqlContext.sql(“select text from tweets where text like ‘%@%’”)

import re

regex= re.compile(“(?:(?<=\s)|)@(\w\*[A-Za-z\_]+\w\*)")

mention\_list=mentions.rdd.map(lambda x :regex.findall(str(x))).collect()

print(mention\_list)

**Output:**

[['always\_nidhi', 'YouTube'], ['OnlyDancers'], ['IcelandNatural'], ['AdeRais', 'SMTOWNGLOBAL', 'skehehdanfdldi', 'llama\_ajol'], ['BeezyDH\_'], ['blakeshelton'], ['DjGregStreet'], ['traceyb65'], ['hokkazonn'], ['daunugh'], ['3Will1'], ['4fucksakesmag'], ['XAmbassadors'], ['Ashton5SOS'], ['jamiisonkent'], ['Parisblass'], ['\_\_akgrown'], [], ['Jazzfeezy'], ['NBCTheVoice'], ['justinbieber', 'diplo', 'Skrillex'], ['cort\_guitars'], [], ['luubreeze'], ['lopestelma24'], ['AaliyahSymone', 'MR\_JAYJONES'], ['iambeckyg'], ['vodrecordings', 'SpotonLI', 'IndieMusicBus', 'indieshuffle', 'IndieMusicBizz'], ['ARMIrocks', 'LNsobolewski'], ['Gurmeetramrahim'], ['TheDurttyBoyz', 'MR\_JAYJONES'], ['JordinSparks'], ['curlsgonewild26'], ['ShawnMendes'], ['itsfatell'], ['BilkoTweet', 'RadioZincSC'], ['SonyMusicSouth'], ['albertposis'], ['TheDJ33'], ['WorshipTeamPod'], ['Tastebuds'], ['OwehMuhammad24'], ['LORIROSELL'], [], ['BenAdamsAuthor'], ['jhornain'], ['Giu\_Capovilla'], ['mcflymusic', 'DougiePoynter'], ['ShawnMendes'], ['OldMillSE18', 'steulalia', 'saminus', 'KarenWilson41'], ['Chica0823'], ['prettydrea7'], ['ShawnMendes'], ['MogorDAmore'], ['ANGCapone', 'Denyque', 'ANGCapone', 'DjRockyUg', 'WickedMusicEnt'], [], [' ---truncating the ouptut due to size

mentioned\_people=list()

For i in mention\_list:

For j in i:

mentioned\_people.append(j)

print(mentioned\_people)

**Output:**

Tube', 'OwehMuhammad24', 'tameimpala', 'gns\_mir', 'henrydog900', 'FLAGALine', 'DJHoppa', 'Hopsin', 'SwizZzleFish', 'soPanel', 'infogarut', 'mazaad87', 'TheVoidMiami', 'HIITMANonDECK', 'samsungaragebar', 'MindshareParis', 'FreeLabelNet', 'ODTDANK', 'waveyboyjam', ' ---truncating the outut because of size

e)

for i in mentioned\_people:

print i ,mentioned\_people.count(i)

**Output:**

thedreamchazer 3

CHlLDHOODRUINER 4

BlackPplVines 4

grimm\_missa 1

MintTunes 1

yokee\_\_ 1

ShawnMendes 190

allithun 2

HIITMANonDECK 100

officialdjjuice 59---truncating the outut because of size

f)

#calculating the mentions

>>> for i in mention\_list:

... for j in i:

... mentioned\_people.append(j)

...

>>> mention\_ordr\_ppl={}

>>> for i in mentioned\_people:

... mention\_ordr\_ppl[i]=mentioned\_people.count(i)

print(mention\_ordr\_ppl)

**Output:**

{'lysganime': 2, 'zaratekarate28': 2, 'NaYsurreal': 2, 'Lil\_Ball': 2, 'OohHeSoSmart': 2, 'netanamaguire': 2, 'emmarosemurray': 2, 'GinaxxBina': 2, 'LBergreen': 2, 'ANGCapone': 4, 'ThatsSarcasm': 2, 'ClarkeeeCom': 2, 'ChrisAmoah\_PF': 2, 'Olympics': 2, 'USAGOV': 2, 'SongLyricMemory': 2, 'ilpaulus': 2, 'mspennypuppy': 2, 'patrickowtf': 2, 'agibsonccc': 2, 'CrankyPappy': 2, 'Spinrilla': 24, 'VilliSpeaks': 2, 'Childs\_Playyy': 2, 'GossipGirliee': 2, 'donaldglover': 2, 'Jsullhart': 2, 'xtoastadox': 2, 'icarusaccount': 2, 'MikeMinttt': 2, 'SayWeCanFly': 2, 'MxshtonNiall': 2, 'theweeknd': 2, 'Pompiers\_Paris': 2, 'NigeriainfoFM': 2, '\_haela': 2, 'OnlineViolin': 2, 'MillardPS': 2, 'paulwesley': 2, 'RealLagugga': 2,

'starsbarsnpbrs': 2, 'ARMIrocks': 2, 'lobosolitario1': 2, 'CamiIIe\_\_': 2, 'JRX\_SID': 2, 'livinlike\_LEX': 2, 'Marlinsoccer20': 2, 'MarkRonson': 2, 'NMannas': 2, 'MiCasaMusic': 2, 'Lydia\_Cupcake': 2, '\_fuckgio': 2, 'BubbaCel': 2, 'dancemom1313': 2, 'FreakingTrue': 4, 'troykoshal': 2, 'jacobmrozinski': 4, 'SamuraiPrimz': 2, 'LaurenEMusic': 2, 'KILLTONYpod': 2, 'larriii7': 2, 'CousinMike\_': 2, 'ParisZigZag': 2, 'TedStryker': 4, 'DjKnightatl': 2, 'WorIdOfDancing': 2, 'cher': 6, 'kristin\_toppel': 2, 'SW\_A9': 2, 'kfcindonesia': 2, 'elkay14': 2, 'allison\_swider': 2, 'Infopadang': 2, 'syssimananga': 2, 'Disciple777777': 2, 'alyssabeetlerxo': 2, '420evilangel': 2, 'tonygallopin': 2, 'LouReed': 2, 'amoebamusic': 2, 'BET': 2, 'tiaraoktaliaa': 2, 'suleymanyer': 2, 'pengenkepoooo': 2, 'PNASNews': 2, 'JBALVIN': 2, 'Fergusonec': 2, 'TheKooriWoman': 4, 'ChelseyWilsonn': 2, 'travelingmoms': 2, 'EurostarUK': 2, 'JayJamesBeats': 2, 'eja090': 2, 'JheneAiko': 2, 'starstunning': 2, 'Cpiepz': 2, 'TurnToTech': 4, '

#putting them into a list

>>> highest\_mentions=sorted(mention\_ordr\_ppl,key=lambda x:x[1],reverse=True)

>>> highest\_mentions

**Output:**

['ezgisilAlpmen', 'mzveegh', 'LzzDaBadGuy', 'EzeaOluchi', 'rzwoodward', 'kzyoung', 'IzzyxPinkprint', 'izzy\_mar0720', 'ozrodriguez1', 'lysganime', 'Lydia\_Cupcake', 'syssimananga', 'kylieminogue', 'Mykeulkeul30', 'TymScorpus', 'mykectown', 'ayeitzsherry', 'dymond\_Tyche', 'Tyrese', 'IyannaOsborn', 'KyStateU', 'zynga', 'nycmusicians', 'PyzzaGod', 'CyberEveryword', 'LyricalSon', 'CyndiDickey', 'cynrobson', 'mymixtapez', 'tylernthetribe', 'TyBaynton', 'xyougetsoalonex', 'Hyjro', 'FyM\_MoTiVe', 'bymaddyness', 'dyeaugust', 'tyturnerband', 'PyxeeStyx', 'HyattTweets', 'GymSavage\_', 'RyaaanMulveeey', 'MylessRaw', 'tylerrjoseph', 'tyleroakley', 'Tyeleez', 'RyanMillaAC', 'Ayoo\_Yeezy', 'Tyler\_Johnson\_5', 'Lyracyst82', 'dyemarcos', 'wyssdaniel', 'ayu\_19980408', 'tyeleez', 'dyeaustins', 'lynnxb', 'NyQuilAndVagina', 'tydollasign', 'RyanSeacrest', 'GypsyBaiIey', 'KygoMusic', 'rydds', 'AyeeeNaaasty\_', 'My\_Little\_Paris', 'dyejoyce', 'nynexttopmodel', 'ayooo\_veee', 'myniakal', 'ty\_army', 'ryanwaniata', 'CyclePronto', 'SylvanEsso', 'PyriteSidewalk', 'ryylor', 'mychonny', 'tylersantee', 'Tyler\_\_Myatt', 'MxshtonNiall', 'ExchangeVocal', 'Expedia', 'kxngsh1t', 'cxstiel', 'expIores', 'oxycodonexo', 'xxblondienxx', '\_xoxochas', 'ExoticRobot', 'ExpectAmiracle9', 'owenstarr', 'Tweetnesian', 'TwitIsenk', 'Awakxning', 'swade750', 'mwhit1999', 'hwmngs', 'swole\_studios', 'kwawkese', 'twentyonepilots', 'Twice\_Bali', 'twaimz', 'awaws22', 'pw052064', 'SwooshyCueb', 'awghaadnicole', 'bwilliamsmusic', '4waysproduction', 'OwehMuhammad24', 'SwizZzleFish', 'SwissLips', 'KwekuAmusah', 'Kwadir\_w', 'twitter', 'gwenstefani', 'AwreadyTV', 'eviemarie\_', 'tvrxb', 'lvst\_\_\_queen', 'dvs\_lkck', 'mvgvlutt', 'xvnexx', 'Eve\_Rags', 'cviddytv', 'Avicii', 'overweightcdub', 'Everyday\_Maree', 'EvaristoVillela', 'pvjbandung', 'avery\_joyce', 'evan\_burnett', 'avachristy3', 'fvckedvps0ciety', 'Qveen\_Bvii', 'Yvonneeee1', 'Kvelertak', 'BubbaCel', 'suleymanyer', 'EurostarUK', 'TurnToTech', 'numanofficial', 'surfregui', 'music', 'authornicoleh', 'Juppel55', 'Rudderbutt',

#getting a sorted list of top 50

>>> top50Mentions = list()

>>> i=0

>>> for k in highest\_mentions:

... if i<=50:

... top50Mentions.append(k)

... i=i+1

...

>>> top50Mentions

**Output:**

['ezgisilAlpmen', 'mzveegh', 'LzzDaBadGuy', 'EzeaOluchi', 'rzwoodward', 'kzyoung', 'IzzyxPinkprint', 'izzy\_mar0720', 'ozrodriguez1', 'lysganime', 'Lydia\_Cupcake', 'syssimananga', 'kylieminogue', 'Mykeulkeul30', 'TymScorpus', 'mykectown', 'ayeitzsherry', 'dymond\_Tyche', 'Tyrese', 'IyannaOsborn', 'KyStateU', 'zynga', 'nycmusicians', 'PyzzaGod', 'CyberEveryword', 'LyricalSon', 'CyndiDickey', 'cynrobson', 'mymixtapez', 'tylernthetribe', 'TyBaynton', 'xyougetsoalonex', 'Hyjro', 'FyM\_MoTiVe', 'bymaddyness', 'dyeaugust', 'tyturnerband', 'PyxeeStyx', 'HyattTweets', 'GymSavage\_', 'RyaaanMulveeey', 'MylessRaw', 'tylerrjoseph', 'tyleroakley', 'Tyeleez', 'RyanMillaAC', 'Ayoo\_Yeezy', 'Tyler\_Johnson\_5', 'Lyracyst82', 'dyemarcos', 'wyssdaniel']

g)

>>> hashtags=sqlContext.sql("select text from tweets where text like '%#%'")

>>> regex=re.compile("(?:(?<=\s)|)#(\w\*[A-Za-z\_]+\w\*)")

>>> only\_hashtags=hashtags.rdd.map(lambda s:regex.findall(str(s))).collect()

>>> hashtags\_list=list()

>>> for i in only\_hashtags:

... for j in i:

... hashtags\_list.append(j)

...

>>> hashtags\_list

**Output:**

['shutUpAndDANCE', 'SHINee', 'AMBER', 'JR50', 'flipagram', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'nowplaying', 'iloveyou', 'newmusic', 'CurrentlyVibingTo', 'MusicMonday', 'theatre', 'dream', 'life', 'music', 'rixton', 'onedirection', 'bonjovi', 'art', 'artstudents', 'cmj', 'IndonesiaTulen', 'StuntTeam', 'IGGL', 'rocknroll', 'music', 'AAofficial', 'MSG3SuccessfulWeeks', 'StuntTeam', 'IGGL', 'Studio', 'Music', 'Production', 'Showtime', 'Jesus', 'GraceXperienceShow', 'EDC2015', 'Music', 'M', 'MexicanBoy', 'DeFiestaEnFiesta', 'DeLunesADomingo', 'dance', 'music', 'eldesmadritodeminovia', 'laamounchingo', 'lamasguapa', 'love', 'Okkanmani', 'ManiRatnam', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Paris', 'VVIPCulture', 'SMS', 'NP', 'next', 'Radio', 'Music', 'Asian', 'MONDAYNIGHTTEASE', 'PULLUP', 'humanity', 'music', 'mahool', 'love', 'life', 'art', 'karma', 'vwbus', 'motorcycle', 'tattoo', 'Paris', 'pfw', 'feminasverige', 'transylvania', 'music', 'music', 'fun', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'ephyre', 'paris', 'leather', 'DisneyParks', 'Paris', 'ONPlay', 'RadarHOTNews', 'GOAT', 'instashot', 'Waves', 'MrProbz', 'Summer', 'Beach', 'Snapback', 'Deep', 'Sea', 'Music', 'EDM', 'Relax', 'SW15', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'uploadforfun', 'uploadforfunaja', 'uffminimal', 'Music', 'MaejorMeAndYou', 'StuntTeam', 'IGGL', 'StuntTeam', 'IGGL', 'Disneyland', 'Paris', 'Lambo', 'Paris', 'BCN', 'HomeSharing', 'MWC15', 'france', 'paris', 'hoteldeville', 'hoteldevilledeparis', 'iceskating', 'BFWmorocco', 'SoundHound', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Quora', 'Hadoop', 'BigData', 'kids', 'Passing', 'cafe', 'Paris', 'notaclosetrapper', 'MBAMBADU', 'AATM', 'JR', 'video', 'performance', 'famous', 'nyc', 'newyork', 'singer', 'live', 'music', 'streetscienceEnt', 'Trippythursdaymia', 'Moretocome', 'VoteDanielPH', 'KCA', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'StuntTeam', 'IGGL', 'cotwoldsvillageinn', 'mind', 'body', 'soul', 'music', 'hiphop', 'danceclasses', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Cookie', 'lastfm', 'grill', 'steak', 'Lobster', 'cocktails', 'music', 'TurnOffTheLights', 'Q971', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Music', 'Listen', 'Download', 'spinrilla', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'NextTime', 'Arianators', 'BestFanArmy', 'iHeartAwards', 'ROADBOYZ', 'QuissyUpSoon', 'jango', 'MusicMonday', 'thankyou', 'myxmusicawards', 'UsAgainstTheWorld', 'PARIS', 'SMS', 'NP', 'Radio', 'Music', 'Asian', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'YFM', 'RNT', 'TAMUFollowTrain', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Paris', 'sambaarte', 'passagemdesom', 'i9', 'therewillbehaters', 'RumBullions', 'IslandReggae', 'mybraintech', 'proudparents', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'CaribbeanVsAmerican2', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'noticeme', 'songwriting', 'music', 'gig', 'filming', 'recording', 'upcoming', 'OurGreatLifeboat', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'jealous', 'misheardlyrics', 'angrymusician', 'music', 'music', 'music', 'Music', 'KPRS', 'music', 'lovesongmedley', 'CandraLagiLesBro', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Trippythursdaymia', 'Respect', 'internasionalfieldstudy', 'paris', 'Music', 'Acoustic', 'Indie', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'KnicksTape', 'IoT', 'BigData', 'ShakedownRadio', 'Podcast', 'Sydney', 'Australia', 'Radio', 'Music', 'WakeUp', 'BelleJourn', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Jbr', 'Dubai', 'TheBeach', 'Paris', 'PureLove', 'Paris', 'jpost', 'skeelo', 'inspired', 'Paris', 'Paris', 'radar', 'autoroute', 'arnaque', 'tunnel', 'A86', 'Fallapart', 'concert', 'sergei', 'music', 'singing', 'family', 'potd', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'madonna', 'RebelHeartTour', 'JohnnyApple1D', 'ArtSchool', 'Saxon', 'Paris', 'Thailandmiss', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Art', 'Job', 'Frankfurt', 'Jobs', 'TweetMyJobs', 'takemetoRio', 'irishweather', 'piano', 'vintagepiano', 'kawai', 'ce10N', 'beautiful', 'instrument', 'music', 'recordplantstudio', 'edcmx', 'edcmexico', 'edcmexico2015', 'kineticcathedral', 'electricdaisycarnival', 'lights', 'stage', 'music', 'SME', 'TurnYaSneakUp', 'rebelheart', 'FifthHarmony', 'NoIDontHaveAProblem', 'lamb', 'beef', 'shawarma', 'radishsalad', 'love', 'art', 'new', 'travel', 'music', 'money', 'edm', 'style', 'guitar', 'design', 'Rock', 'fit', 'Fun', 'hot', 'herrera', 'music', 'superstark', 'song', 'korea', 'roykim', 'Par', 'soir', 'France', 'incroyable', 'Trippythursdaymia', 'love', 'art', 'new', 'travel', 'music', 'money', 'edm', 'style', 'guitar', 'design', 'Rock', 'fit', 'Fun', 'hot', 'herrera', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'Trippythursdaymia', 'IGetPulledOverAlot', 'HMD', 'amen', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'FINALLYFRIDAY', 'oldestoneshere', 'gamh', 'MBAMBADU', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'sistahood', 'teamgoldii', 'music', 'LifeFails', 'MoneyGxng', 'ProjectMgmt', 'Job', 'Wien', 'Jobs', 'TweetMyJobs', 'cantwaitforthis', 'podscast', 'motivation', 'workout', 'bcn', 'hardwell', 'house', 'cinematopeia', 'fitness', 'top', 'muybien', 'music', 'live', 'NickiMinaj', 'viroshop', 'viroaddict', 'onlineshop', 'virolovers', 'viromania', 'DME', 'ROADBOYZ', 'QuissyUpSoon', 'np', 'SoundCloud', 'TheVoice', 'ComingSoon', 'FPD15', 'StuntTeam', 'IGGL', 'Klezmer', 'Klezmer', 'PFW', 'ParisFashionWeek', 'FW15', 'FW2015', 'SHINee', 'AMBER', 'vibetime', 'love', 'paris', 'yyc', 'yycarts', 'cbc', 'music', 'genre', 'song', 'songs', 'TagsForLikes', 'melody', 'hiphop', 'rnb', 'Trippythursdaymia', 'Trippythursdaymia',

h)

for i in hashtags\_list:

print i, hashtags\_list.count(i)

**Output:**

Winner 1

AlexandreDesplat 1

Best 1

Music 97

score 1

TheGrandBudapestHotel 1

fans 1

customized 1

shoes 2

kicks 2

saga 1

bandswag 2

music 258

bands 3

SME 21

TurnYaSneakUp 21

DME 253

ROADBOYZ 252

QuissyUpSoon 255

hadoop 1

i)

>>> placedf=sqlContext.sql("select \* from tweets where place='Manhattan'")

>>>

>>> placedf.show()

**Output:**

+-------------+------------------+---------+--------------------+--------------------+

| country| id| place| text| user|

+-------------+------------------+---------+--------------------+--------------------+

|United States|572575240615796737|Manhattan|@OnlyDancers Bell...| TagineDiningGlobal|

|United States|572571014653214720|Manhattan|Rachel Potter CD ...|Jillian / Horribella|

|United States|572573152234106880|Manhattan|@FollowNYCNews Be...| TagineDiningGlobal|

|United States|572632707588616193|Manhattan|if you're a white...| childlike empress|

|United States|572585418534744064|Manhattan|#Visually stimula...| Jasmine R. Castillo|

|United States|572595037596471296|Manhattan|@YahooMusic Real ...| constantino pires|

|United States|572626103740862464|Manhattan|Friday happy hour...| Chris Alker|

|United States|572568898127728640|Manhattan|@therachelpotter ...| Brett Africk|

|United States|572573935059013633|Manhattan|@BellyDanceNYC Be...| TagineDiningGlobal|

|United States|572601694040694784|Manhattan|"Maybe music isn'...| thank you sorry|

|United States|572576649209901056|Manhattan|I have got to sta...| Jen Wu|

|United States|572599091433238530|Manhattan|“@RapFavorites: N...| AME|

|United States|572570291790069760|Manhattan|@BellyDanceNYC Be...| TagineDiningGlobal|

|United States|572564344589955072|Manhattan|The music selecti...| SSGG|

|United States|572632192968491008|Manhattan|at least being a ...| childlike empress|

|United States|572595516887998464|Manhattan|Music and Movemen...| Steve Scher|

|United States|572556614689017857|Manhattan|@ShaneTsunami in ...| Sahar|

|United States|572571658155896832|Manhattan|@foodiegeektv Bel...| TagineDiningGlobal|

|United States|572573393524027393|Manhattan|@dancehallflex1 ...| TagineDiningGlobal|

|United States|572568951911260160|Manhattan| Loft music👍| Luis De La Cruz|

+-------------+------------------+---------+--------------------+--------------------+

only showing top 20 rows

j)

>>> order\_countrydf=sqlContext.sql("select country, count(user) as users from tweets group by country order by users desc")

>>> order\_countrydf.show()

**Output:**

+--------------------+-----+

| country|users|

+--------------------+-----+

| United States| 4841|

| France| 737|

| Indonesia| 370|

| United Kingdom| 365|

| Brasil| 256|

| Canada| 172|

|Republika ng Pili...| 151|

| Argentina| 104|

| South Africa| 92|

| Australia| 90|

| India| 66|

| México| 59|

| 日本| 57|

| España| 53|

| Malaysia| 50|

| Kenya| 44|

| Türkiye| 42|

| Ghana| 38|

| Colombia| 33|

| Nigeria| 33|

+--------------------+-----+

>>> tweets\_countrydf=sqlContext.sql("select country, count(text) as no\_of\_tweets from tweets group by country order by no\_of\_tweets desc")

>>> tweets\_countrydf.show()

**Output:**

+--------------------+------------+

| country|no\_of\_tweets|

+--------------------+------------+

| United States| 4841|

| France| 737|

| Indonesia| 370|

| United Kingdom| 365|

| Brasil| 256|

| Canada| 172|

|Republika ng Pili...| 151|

| Argentina| 104|

| South Africa| 92|

| Australia| 90|

| India| 66|

| México| 59|

| 日本| 57|

| España| 53|

| Malaysia| 50|

| Kenya| 44|

| Türkiye| 42|

| Ghana| 38|

| Nigeria| 33|

| Colombia| 33|

Note: Unites states is leading in both number of tweets and number of users.

k)

>>> df=sqlContext.sql("select user, count(text) as no\_of\_tweets from tweets where country='France' and LOWER(text) like '%paris%' group by user")

>>> df.show()

**Output:**

+--------------------+------------+

| user|no\_of\_tweets|

+--------------------+------------+

| Filip| 1|

| CEB Jobs| 1|

| Mél | I MET HIM| 1|

| 이 진연 asiatophile| 1|

| Traveling Precils| 1|

| EMILIE| 1|

| amidala.| 1|

| SH.| 1|

| deschadul| 1|

| Ana Clara Garmendia| 1|

| ART aujourd'hui gal| 1|

| Antha Kincy ORTIES| 1|

| SAMUUUUH| 1|

| TheTony (•—•)| 1|

| リラックス| 1|

| Dimitri Kazantzaki| 2|

|La Cuerda Floja -200| 1|

| Marie-Charlotte| 2|

| Yann Gillet| 1|

| Filippo Fior| 1|

+--------------------+------------+

**Answer 3**

**a)**

from pyspark.sql import SQLContext

sqlContext= SQLContext(sc)

stationdf=sqlContext.read.format('com.databricks.spark.csv').options(header='true',inferSchema='true').load('file:///home/cloudera/Desktop/station.csv')

tripdf=sqlContext.read.format('com.databricks.spark.csv').options(header='true',inferSchema='true').load('file:///home/cloudera/Desktop/trip.csv')

stationdf.show()

**Output:**

+----------+--------------------+---------+-----------+---------+------------+------------+

|station\_id| name| lat| long|dockcount| landmark|installation|

+----------+--------------------+---------+-----------+---------+------------+------------+

| 2|San Jose Diridon ...|37.329732|-121.901782| 27| San Jose| 8/6/2013|

| 3|San Jose Civic Ce...|37.330698|-121.888979| 15| San Jose| 8/5/2013|

| 4|Santa Clara at Al...|37.333988|-121.894902| 11| San Jose| 8/6/2013|

| 5| Adobe on Almaden|37.331415| -121.8932| 19| San Jose| 8/5/2013|

| 6| San Pedro Square|37.336721|-121.894074| 15| San Jose| 8/7/2013|

| 7|Paseo de San Antonio|37.333798|-121.886943| 15| San Jose| 8/7/2013|

| 8| San Salvador at 1st|37.330165|-121.885831| 15| San Jose| 8/5/2013|

| 9| Japantown|37.348742|-121.894715| 15| San Jose| 8/5/2013|

| 10| San Jose City Hall|37.337391|-121.886995| 15| San Jose| 8/6/2013|

| 11| MLK Library|37.335885| -121.88566| 19| San Jose| 8/6/2013|

| 12|SJSU 4th at San C...|37.332808|-121.883891| 19| San Jose| 8/7/2013|

| 13| St James Park|37.339301|-121.889937| 15| San Jose| 8/6/2013|

| 14|Arena Green / SAP...|37.332692|-121.900084| 19| San Jose| 8/5/2013|

| 16|SJSU - San Salvad...|37.333955|-121.877349| 15| San Jose| 8/7/2013|

| 21| Franklin at Maple|37.481758|-122.226904| 15|Redwood City| 8/12/2013|

| 22|Redwood City Calt...|37.486078|-122.232089| 25|Redwood City| 8/15/2013|

| 23|San Mateo County ...|37.487616|-122.229951| 15|Redwood City| 8/15/2013|

| 24|Redwood City Publ...|37.484219|-122.227424| 15|Redwood City| 8/12/2013|

| 25|Stanford in Redwo...| 37.48537|-122.203288| 15|Redwood City| 8/12/2013|

| 26|Redwood City Medi...|37.487682|-122.223492| 15|Redwood City| 8/12/2013|

+----------+--------------------+---------+-----------+---------+------------+------------+

only showing top 20 rows

tripdf.show()

+-------+--------+---------------+--------------------+--------------+---------------+--------------------+------------+------+---------------+--------+

|Trip ID|Duration| Start Date| Start Station|Start Terminal| End Date| End Station|End Terminal|Bike #|Subscriber Type|Zip Code|

+-------+--------+---------------+--------------------+--------------+---------------+--------------------+------------+------+---------------+--------+

| 913460| 765|8/31/2015 23:26|Harry Bridges Pla...| 50|8/31/2015 23:39|San Francisco Cal...| 70| 288| Subscriber| 2139|

| 913459| 1036|8/31/2015 23:11|San Antonio Shopp...| 31|8/31/2015 23:28|Mountain View Cit...| 27| 35| Subscriber| 95032|

| 913455| 307|8/31/2015 23:13| Post at Kearny| 47|8/31/2015 23:18| 2nd at South Park| 64| 468| Subscriber| 94107|

| 913454| 409|8/31/2015 23:10| San Jose City Hall| 10|8/31/2015 23:17| San Salvador at 1st| 8| 68| Subscriber| 95113|

| 913453| 789|8/31/2015 23:09|Embarcadero at Fo...| 51|8/31/2015 23:22|Embarcadero at Sa...| 60| 487| Customer| 9069|

| 913452| 293|8/31/2015 23:07|Yerba Buena Cente...| 68|8/31/2015 23:12|San Francisco Cal...| 70| 538| Subscriber| 94118|

| 913451| 896|8/31/2015 23:07|Embarcadero at Fo...| 51|8/31/2015 23:22|Embarcadero at Sa...| 60| 363| Customer| 92562|

| 913450| 255|8/31/2015 22:16|Embarcadero at Sa...| 60|8/31/2015 22:20| Steuart at Market| 74| 470| Subscriber| 94111|

| 913449| 126|8/31/2015 22:12| Beale at Market| 56|8/31/2015 22:15|Temporary Transba...| 55| 439| Subscriber| 94130|

| 913448| 932|8/31/2015 21:57| Post at Kearny| 47|8/31/2015 22:12|South Van Ness at...| 66| 472| Subscriber| 94702|

| 913443| 691|8/31/2015 21:49|Embarcadero at Sa...| 60|8/31/2015 22:01| Market at Sansome| 77| 434| Subscriber| 94109|

| 913442| 633|8/31/2015 21:44| Market at 10th| 67|8/31/2015 21:54|San Francisco Cal...| 70| 531| Subscriber| 94107|

| 913441| 387|8/31/2015 21:39| Market at 4th| 76|8/31/2015 21:46|Grant Avenue at C...| 73| 383| Subscriber| 94104|

| 913440| 281|8/31/2015 21:31| Market at Sansome| 77|8/31/2015 21:36|Broadway St at Ba...| 82| 621| Subscriber| 94107|

| 913435| 424|8/31/2015 21:25|Temporary Transba...| 55|8/31/2015 21:33|San Francisco Cal...| 69| 602| Subscriber| 94401|

| 913434| 283|8/31/2015 21:19|San Francisco Cal...| 69|8/31/2015 21:24| Townsend at 7th| 65| 521| Subscriber| 94107|

| 913433| 145|8/31/2015 21:17|University and Em...| 35|8/31/2015 21:20|Cowper at University| 37| 75| Customer| 6907|

| 913432| 703|8/31/2015 21:16| Spear at Folsom| 49|8/31/2015 21:28|San Francisco Cal...| 69| 426| Subscriber| 95032|

| 913431| 605|8/31/2015 21:11|Temporary Transba...| 55|8/31/2015 21:21|Grant Avenue at C...| 73| 572| Subscriber| 94133|

| 913429| 902|8/31/2015 21:07|San Francisco Cal...| 70|8/31/2015 21:22|Broadway St at Ba...| 82| 501| Subscriber| 94133|

+-------+--------+---------------+--------------------+--------------+---------------+--------------------+------------+------+---------------+--------+

only showing top 20 rows

vertices=stationdf['name','station\_id']

vertices

vertices.show()

**Output:**

+--------------------+----------+

| name|station\_id|

+--------------------+----------+

|San Jose Diridon ...| 2|

|San Jose Civic Ce...| 3|

|Santa Clara at Al...| 4|

| Adobe on Almaden| 5|

| San Pedro Square| 6|

|Paseo de San Antonio| 7|

| San Salvador at 1st| 8|

| Japantown| 9|

| San Jose City Hall| 10|

| MLK Library| 11|

|SJSU 4th at San C...| 12|

| St James Park| 13|

|Arena Green / SAP...| 14|

|SJSU - San Salvad...| 16|

| Franklin at Maple| 21|

|Redwood City Calt...| 22|

|San Mateo County ...| 23|

|Redwood City Publ...| 24|

|Stanford in Redwo...| 25|

|Redwood City Medi...| 26|

+--------------------+----------+

only showing top 20 rows

edges=tripdf['Start Station','End Station']

from pyspark.sql.functions import \*

edges=edges.select(col("Start Station").alias("src"), col("End Station").alias("dst"))

vertices=vertices.select(col("name").alias("id"),col("station\_id").alias("s\_id"))

from graphframes import \*

g = GraphFrame(vertices, edges)

b)

g.inDegrees.sort(desc("inDegree")).limit(10).show()

**Output:**

+--------------------+--------+

| id|inDegree|

+--------------------+--------+

|San Francisco Cal...| 34810|

|San Francisco Cal...| 22523|

|Harry Bridges Pla...| 17810|

| 2nd at Townsend| 15463|

| Townsend at 7th| 15422|

|Embarcadero at Sa...| 15065|

| Market at Sansome| 13916|

| Steuart at Market| 13617|

|Temporary Transba...| 12966|

| Powell Street BART| 10239|

+--------------------+--------+

g.outDegrees.sort(desc("outDegree")).limit(10).show()

**Output:**

+--------------------+---------+

| id|outDegree|

+--------------------+---------+

|San Francisco Cal...| 26304|

|San Francisco Cal...| 21758|

|Harry Bridges Pla...| 17255|

|Temporary Transba...| 14436|

|Embarcadero at Sa...| 14158|

| 2nd at Townsend| 14026|

| Townsend at 7th| 13752|

| Steuart at Market| 13687|

| Market at 10th| 11885|

| Market at Sansome| 11431|

| Market at 4th| 9894|

| Powell Street BART| 9695|

| 2nd at South Park| 9469|

| Beale at Market| 8359|

|Grant Avenue at C...| 8337|

| 2nd at Folsom| 7999|

|Civic Center BART...| 7760|

| 5th at Howard| 7708|

|Broadway St at Ba...| 7676|

|Embarcadero at Fo...| 7596|

+--------------------+---------+

g.vertices.show(10)

**Output:**

+--------------------+----+

| id|s\_id|

+--------------------+----+

|San Jose Diridon ...| 2|

|San Jose Civic Ce...| 3|

|Santa Clara at Al...| 4|

| Adobe on Almaden| 5|

| San Pedro Square| 6|

|Paseo de San Antonio| 7|

| San Salvador at 1st| 8|

| Japantown| 9|

| San Jose City Hall| 10|

| MLK Library| 11|

+--------------------+----+

only showing top 10 rows

c)

paths = g.find("()-[e]->()")

paths.show(10)

print(paths)

**Output:**

+--------------------+

| e|

+--------------------+

|[Harry Bridges Pl...|

|[San Antonio Shop...|

|[Post at Kearny,2...|

|[San Jose City Ha...|

|[Embarcadero at F...|

|[Yerba Buena Cent...|

|[Embarcadero at F...|

|[Embarcadero at S...|

|[Beale at Market,...|

|[Post at Kearny,S...|

+--------------------+

only showing top 10 rows

d)

inDeg=g.inDegrees

outDeg=g.outDegrees

degreeRatio = inDeg.join(outDeg, inDeg.id==outDeg.id).drop(outDeg.id).selectExpr("id","inDegree/outDegree as degreeRatio").cache()

degreeRatio.sort(desc("degreeRatio")).show(10)

**Output:**

+--------------------+------------------+

| id| degreeRatio|

+--------------------+------------------+

|Redwood City Medi...|1.5333333333333334|

|San Mateo County ...|1.4724409448818898|

|SJSU 4th at San C...|1.3621052631578947|

|San Francisco Cal...|1.3233728710462287|

|Washington at Kearny|1.3086466165413533|

|Paseo de San Antonio|1.2535046728971964|

|California Ave Ca...| 1.24|

| Franklin at Maple|1.2345679012345678|

|Embarcadero at Va...|1.2201707365495336|

| Market at Sansome|1.2173913043478262|

+--------------------+------------------+

only showing top 10 rows

e)

paths = g.find("(a)-[e]->(b); (b)-[e]->(c); !(c)-[e]->(a)")

paths.show(10)

**Output:**

Had to interrupt as it was taking too much time to process.

f)

ranks=g.pageRank(resetProbability=0.15,maxIter=5)

ranks.vertices.orderBy(ranks.vertices.pagerank.desc()).limit(10).show()

**Output:**

+--------------------+----+------------------+

| id|s\_id| pagerank|

+--------------------+----+------------------+

|San Jose Diridon ...| 2|2.2815258224238137|

|San Francisco Cal...| 70|1.9076925128811206|

|Redwood City Calt...| 22|1.4225549210068156|

|Mountain View Cal...| 28|1.3956850803608403|

|San Francisco Cal...| 69|1.2880093979599325|

|Harry Bridges Pla...| 50|1.0394789374393372|

| 2nd at Townsend| 61|0.9139488938809491|

| Townsend at 7th| 65|0.9105184744605765|

|Embarcadero at Sa...| 60|0.8966730699783413|

|Santa Clara at Al...| 4|0.8896981662193627|

+--------------------+----+------------------+

**Answer 4**

**Problem Statement** : to replicate the functionality of movie recommendations from IMDB

**Given Data**: Moviesinfo

UserInfo

RatingsInfo

Steps to arrive at a solution or the multiple solutions we have:

1)Looking at past user data

2)Looking at the population as a whole, e.g: Google Trends ,Sample based

3)Recommendations by Collabrative filtering based on like minded people. But how to select like mindedness. We see some data points (users) who like a certain movie and then find the correlation. So that we can get the movies that are liked by certain users who like a certain movie on an average. Correlation gives us the overall mood of how people have been rating rather than average of somebody. Hence this might be the best method to go ahead.

Step wise approach discussed

1.Read the data and map it to (user, (movie,rating))

2.Self-join for each user to get all combinations of movie ratings by the user:

(user, ((movie,rating),(movie,rating))

­Every user rated at least 20 movies, so this blows up the data to huge scale

3.Remove duplicates from the self-join

4.Key by every combination of movies that were rated together:

((movie1,movie2),(rating1,rating2))

5.Group pairs of ratings by pairs of movies: ((m1,m2),((r1,r2),(r1,r2),…,(r1,r2)))

6.Calculate statistical similarity for each pair: ((m1,m2),(score,numPairs))

7.Filter and sort the top ten similarities to a movie requested by the user.

**Code Explaining**

Note:Kindly look for the comments in blue

import sys

from pyspark import SparkConf, SparkContext

from math import sqrt

def loadMovieNames(): \*\*defining LoadMovieNames

movieNames = {}

\*\*creating an empty array

with open("/home/cloudera/moviedata/itemfile.txt") as f:

\*\*opening itemfile

for line in f:

\*\*traversing lines in file

fields = line.split('|')

\*\* splitting text by separator | and storing in fields

movieNames[int(fields[0])] = fields[1].decode('ascii', 'ignore')

\*\*now fields array is converted from Unicode to asci with ignoring the errors and storing it in movieNames

return movieNames

\*\*making pairs of movies and their ratings from (users, ratings)

\*\* this will create a movie rating combination but since we are getting this from users.If a user who has not watched a certain movie for which we are finding will not come here.

def makePairs((user, ratings)):

\*\*assigning value to different variables from ratings array and returning the pairs

(movie1, rating1) = ratings[0]

(movie2, rating2) = ratings[1]

return ((movie1, movie2), (rating1, rating2))

\*\*function to filter out the output we have after join

def filterDuplicates( (userID, ratings) ):

\*\*the value we receive in the ratings array could be assigned to variables and then can be compared

(movie1, rating1) = ratings[0]

(movie2, rating2) = ratings[1]

return movie1 < movie2

\*\*function that gives the correlation score of all movie pairs

def computeCosineSimilarity(ratingPairs):

numPairs = 0

sum\_xx = sum\_yy = sum\_xy = 0

for ratingX, ratingY in ratingPairs:

sum\_xx += ratingX \* ratingX

sum\_yy += ratingY \* ratingY

sum\_xy += ratingX \* ratingY

numPairs += 1

numerator = sum\_xy

denominator = sqrt(sum\_xx) \* sqrt(sum\_yy)

score = 0

if (denominator):

score = (numerator / (float(denominator)))

return (score, numPairs)

\*\*this function returns the pair score and normal score which will be our parameters to set thresholds

conf = SparkConf().setMaster("local[\*]").setAppName("MovieSimilarities")

\*\*setting app value

sc = SparkContext(conf = conf)

\*\* setting up spark context

print "\nLoading movie names..."

nameDict = loadMovieNames()

\*\*calling loadMovieNames function in order to have all the names in nameDict

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

\*\*loading the file into data

# Map ratings to key / value pairs: user ID => movie ID, rating

ratings = data.map(lambda l: l.split()).map(lambda l: (int(l[0]), (int(l[1]), float(l[2]))))

\*\*using the map function we are mapping the data in form of (userId, (movieId,rating))

# Emit every movie rated together by the same user.

# Self-join to find every combination.

joinedRatings = ratings.join(ratings)

\*\*joining these rating to get every possible combination using Cartesian join

# At this point our RDD consists of userID => ((movieID, rating), (movieID, rating))

# Filter out duplicate pairs

uniqueJoinedRatings = joinedRatings.filter(filterDuplicates)

\*\*passing filterDuplicates to filter function in order filterDuplicates function to be applied on joinedRatings

# Now key by (movie1, movie2) pairs.

moviePairs = uniqueJoinedRatings.map(makePairs)

\*\*passing the makePairs function into map in order to make (movie1,movie2) pairs

# We now have (movie1, movie2) => (rating1, rating2)

# Now collect all ratings for each movie pair and compute similarity

moviePairRatings = moviePairs.groupByKey()

\*\*now by groupByKey() we can find all the possible ratings given by all users that are given to a set of 2 movies and then compare the performance of those movies by correlating them

# We now have (movie1, movie2) = > (rating1, rating2), (rating1, rating2) ...

# Can now compute similarities.

moviePairSimilarities = moviePairRatings.mapValues(computeCosineSimilarity).cache()

\*\*passing the CosineSimilarity function in these ratings in order to analyse them.

# Save the results if desired

#moviePairSimilarities.sortByKey()

#moviePairSimilarities.saveAsTextFile("movie-sims")

\*\*we can sort the results by movies having highest similarity and then maybe save it in a text file

# Extract similarities for the movie we care about that are "good".

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

scoreThreshold = 0.10

coOccurenceThreshold = 2

\*\*setting up some thresholds for scores so that we can categorize them into highly similar movies. We can alter these values in order to improve code. So any value which is below these scoreThreshold we can remove it from the code. This step is to avoid giving bad recommendations. CoOcurrence score is actually giving us the value that at least two people must have watched both movie 1 and movie 2

\*\*Now we can take input from user in order to give him recommendations

movieID = int(sys.argv[1])

\*\*Now we want to get our result by giving the movieId to a filter according to the thresholds above

# Filter for movies with this sim that are "good" as defined by

# our quality thresholds above

filteredResults = moviePairSimilarities.filter(lambda((pair,sim)): \

(pair[0] == movieID or pair[1] == movieID) \

and sim[0] > scoreThreshold and sim[1] > coOccurenceThreshold)

\*\*we can now sort our filteredResults in order to get top 10 recommendations to the movieID we provided

# Sort by quality score.

results = filteredResults.map(lambda((pair,sim)): (sim, pair)).sortByKey(ascending = False).take(10)

print "Top 10 similar movies for " + nameDict[movieID]

for result in results:

(sim, pair) = result

\*\*Now we can print all those highly recommended movies to a person who has watched a movie with given moviedId

# Display the similarity result that isn't the movie we're looking at

similarMovieID = pair[0]

if (similarMovieID == movieID):

similarMovieID = pair[1]

print nameDict[similarMovieID] + "\tscore: " + str(sim[0]) + "\tstrength: " + str(sim[1])