Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_ / .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.0.1

/\_/

Using Python version 2.7.6 (default, Mar 22 2014 22:59:56)

SparkSession available as 'spark'.

>>> sc

<pyspark.context.SparkContext object at 0x7fe245a5fcd0>

>>> rdd=sc.textFile("file:////home/shalbigdata/Documents/spark/Assignments/assignment4/input/ratings.csv")

**# Remove the header row**

>>> header = rdd.first()

>>> data = rdd.filter(lambda x: x != header)

>>> data.count()

|  |
| --- |
| **24404096** |

**# Count of unique users**

>>> users = data.map(lambda x:x.split(',')[0])

>>> output2 = users.distinct().count()

>>> output2

|  |
| --- |
| **259137** |

**# Count of unique movies**

>>> movies = data.map(lambda x:x.split(',')[1])

>>> output3 = movies.distinct().count()

>>> output3

|  |
| --- |
| **39443** |

**# Top-3 average-rating by all users**

>>> userandrating = data.map(lambda x: (x.split(',')[0],x.split(',')[2]))

>>> userandratinggrouped = userandrating.groupByKey().mapValues(lambda x:list(x))

**# Define a function to find average**

>>> def avrg(x):

... sum = 0

... count = 0

... for item in x:

... sum += float(item)

... count += 1

... return sum/count

...

>>> avguserrating = userandratinggrouped.mapValues(lambda x:avrg(x))

**# Take ordered to get the top 3 values**

>>> output4 = avguserrating.takeOrdered(3, key = lambda x: -x[1])

>>> output4

|  |
| --- |
| **[(u'177797', 5.0), (u'205933', 5.0), (u'107703', 5.0)]** |

**# number of users who have rated movies in a given range**

>>> input = rdd.map(lambda x: (x.split(',')[0],x.split(',')[1]))

>>> nummovies = input.distinct().groupByKey().mapValues(lambda x:list(x)).mapValues(lambda x: len(x))

**# function to define the user range**

>>> def userrange(x):

... num = x[1]

... tup = ()

... if(num > 0 and num <= 10):

... tup = ('users rated (minimum, maximum) of (1,10) movies',1)

... elif(num > 10 and num <= 20):

... tup = ('users rated (minimum, maximum) of (11,20) movies',1)

... elif(num > 20 and num <= 30):

... tup = ('users rated (minimum, maximum) of (21,30) movies',1)

... elif(num > 30 and num <= 40):

... tup = ('users rated (minimum, maximum) of (31,40) movies',1)

... elif(num > 40 and num <= 50):

... tup = ('users rated (minimum, maximum) of (41,50) movies',1)

... elif(num > 50 and num <= 200):

... tup = ('users rated (minimum, maximum) of (51,200) movies',1)

... return tup

...

>>> usermovirange = nummovies.map(lambda x:userrange(x))

>>> filtered = usermovirange.filter(lambda x: x!= ())

>>> op5 = filtered.reduceByKey(lambda x,y: x+y)

>>> op5.collect()

|  |
| --- |
| **[('users rated (minimum, maximum) of (11,20) movies', 64252),**  **('users rated (minimum, maximum) of (21,30) movies', 26447),**  **('users rated (minimum, maximum) of (41,50) movies', 13014),**  **('users rated (minimum, maximum) of (31,40) movies', 17307),**  **('users rated (minimum, maximum) of (51,200) movies', 66092),**  **('users rated (minimum, maximum) of (1,10) movies', 41271)]** |

**# users who have rated the same identical movies of 5 or more**

>>> usermovilist = input.distinct().groupByKey().mapValues(lambda x:list(x))

>>> changekey = usermovilist.map(lambda (x,y) : (y,x)).map(lambda x : (sorted(x[0],key=int),x[1]))

>>> changekeytotuple = changekey.map(lambda (x,y) : (tuple(x),y))

>>> op6 = changekeytotuple.groupByKey().mapValues(lambda x:list(x))

>>> output6final = op6.filter(lambda (x,y): len(x)>= 5 and len(y)>1)

**# function to format the output**

>>> def disp(x):

... st = ''

... st += 'Users: '+str(x[1])+' Movies: '+str(x[0])

... return st

...

>>> dispop6 = output6final.map(lambda x:disp(x))

>>> dispop6.take(5)

|  |
| --- |
| ["Users: [u'117516', u'235406'] Movies: (u'150', u'153', u'231', u'296', u'316', u'318', u'344', u'349', u'380', u'588', u'590', u'592', u'595')",  "Users: [u'222754', u'20920'] Movies: (u'153', u'165', u'231', u'296', u'318', u'344', u'349', u'380', u'588', u'590', u'592', u'595')",  "Users: [u'135668', u'107130'] Movies: (u'10', u'19', u'21', u'34', u'47', u'110', u'150', u'153', u'161', u'165', u'185', u'208', u'225', u'231', u'253', u'288', u'292', u'296', u'300', u'316', u'318', u'329', u'339', u'344', u'349', u'356', u'364', u'367', u'377', u'380', u'410', u'434', u'454', u'457', u'480', u'500', u'586', u'587', u'588', u'589', u'590', u'592', u'593', u'595', u'597')",  "Users: [u'191530', u'90536'] Movies: (u'150', u'153', u'165', u'231', u'292', u'296', u'316', u'318', u'329', u'344', u'349', u'356', u'380', u'434', u'457', u'588', u'590', u'592', u'593')",  "Users: [u'244354', u'252483', u'204208'] Movies: (u'260', u'318', u'356', u'527', u'1196', u'1198', u'2571')"] |