DS503/CS585 - Big Data Management
Project 3 Report

Xiaoting Cui
Brendan Foley
Jiaxing Zhang

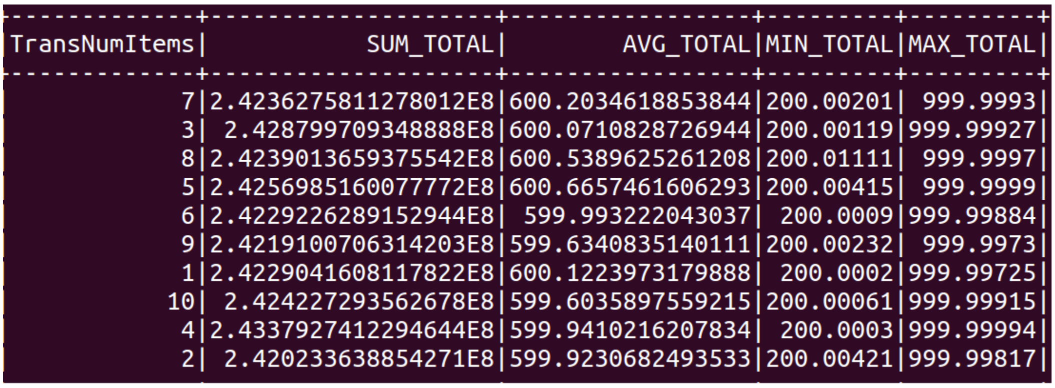
Problem1 SparkSQL

Step 1: Create Dataset Customers and Transactions.

The Transcations dataset has 5,000,000 records. And its size is 303 MB.

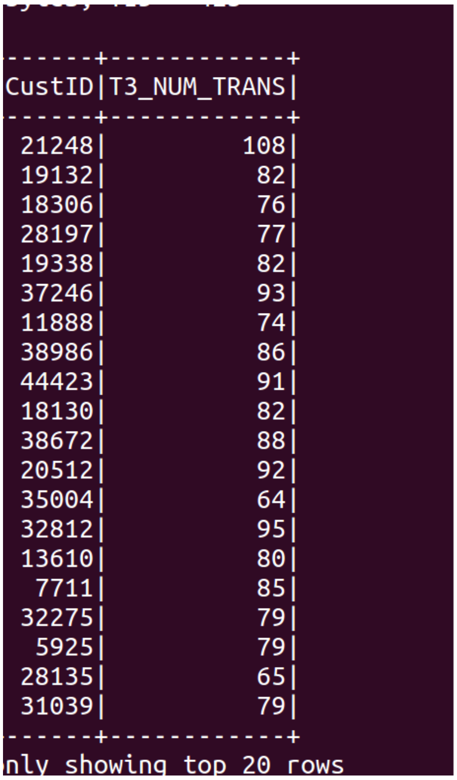
Step 2: SparkSQL query

Screenshor of T2:



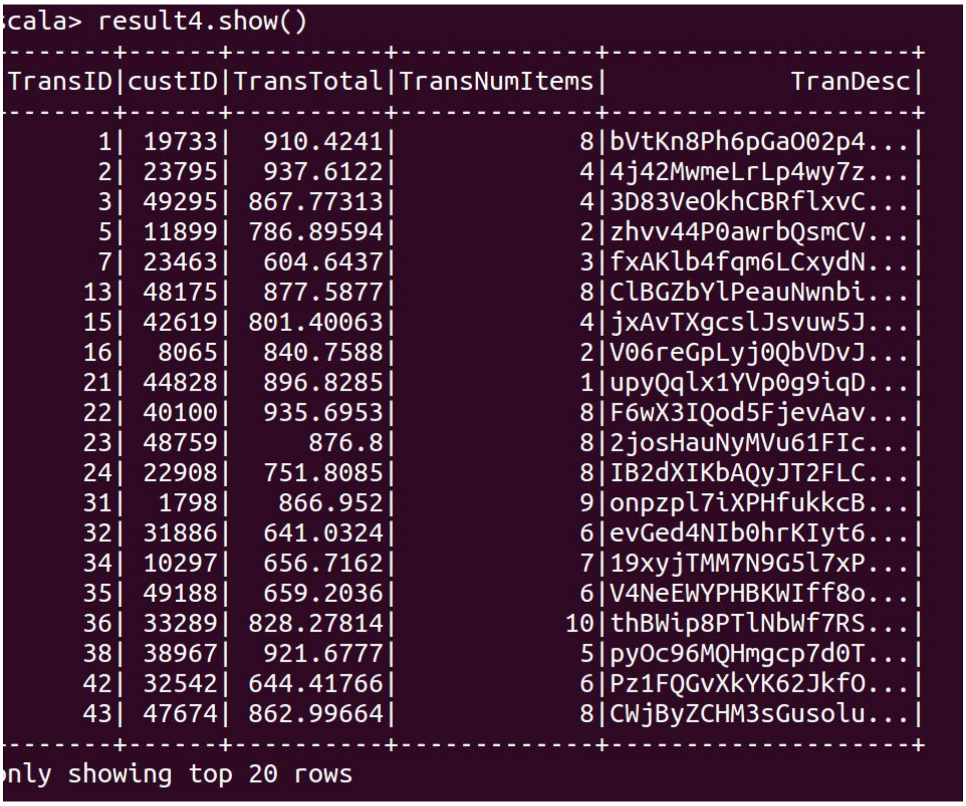
```
+-------------+--------------------+----------------+---------+---------+
|TransNumItems|           SUM_TOTAL|       AVG_TOTAL|MIN_TOTAL|MAX_TOTAL|
+-------------+--------------------+----------------+---------+---------+
|            7|2.4236275811278012E8|600.2034618853844|200.00201| 999.9993|
|            3| 2.428799709348888E8|600.0710828726944|200.00119|999.99927|
|            8|2.4239013659375542E8|600.5389625261208|200.01111| 999.9997|
|            5|2.4256985160077772E8|600.6657461606293|200.00415| 999.9999|
|            6|2.4229226289152944E8| 599.993222043037| 200.0009|999.99884|
|            9|2.4219100706314203E8|599.6340835140111|200.00232| 999.9973|
|            1|2.4229041608117822E8|600.1223973179888| 200.0002|999.99725|
|           10| 2.424227293562678E8|599.6035897559215|200.00061|999.99915|
|            4|2.4337927412294644E8|599.9410216207834| 200.0003|999.99994|
|            2| 2.420233638854271E8|599.9230682493533|200.00421|999.99817|
```

Screen of T3:



```
------+------------+
CustID|T3_NUM_TRANS|
------+------------+
 21248|         108|
 19132|          82|
 18306|          76|
 28197|          77|
 19338|          82|
 37246|          93|
 11888|          74|
 38986|          86|
 44423|          91|
 18130|          82|
 38672|          88|
 20512|          92|
 35004|          64|
 32812|          95|
 13610|          80|
  7711|          85|
 32275|          79|
  5925|          79|
 28135|          65|
 31039|          79|
------+------------+
nly showing top 20 rows
```

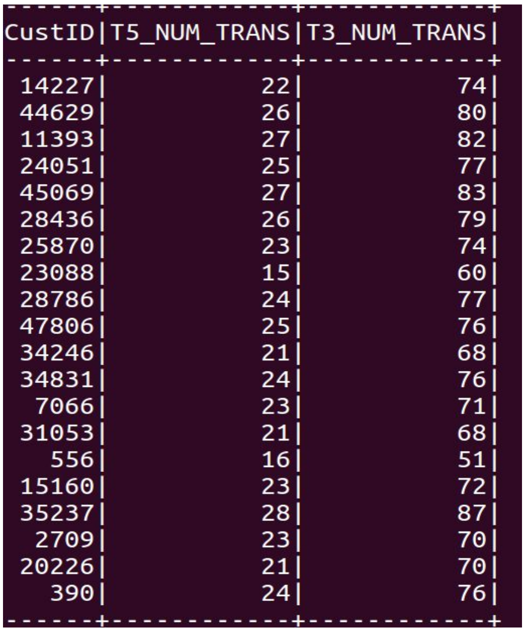
Screenshot of T5:



```
cala> result4.show()
-------+------+----------+-------------+--------------------+
TransID|custID|TransTotal|TransNumItems|            TranDesc|
-------+------+----------+-------------+--------------------+
      1| 19733|  910.4241|            8|bVtKn8Ph6pGaO02p4...|
      2| 23795|  937.6122|            4|4j42MwmeLrLp4wy7z...|
      3| 49295| 867.77313|            4|3D83VeOkhCBRflxvC...|
      5| 11899| 786.89594|            2|zhvv44P0awrbQsmCV...|
      7| 23463|  604.6437|            3|fxAKlb4fqm6LCxydN...|
     13| 48175|  877.5877|            8|ClBGZbYlPeauNwnbi...|
     15| 42619| 801.40063|            4|jxAvTXgcslJsvuw5J...|
     16|  8065|  840.7588|            2|V06reGpLyj0QbVDvJ...|
     21| 44828|  896.8285|            1|upyQqlx1YVp0g9iqD...|
     22| 40100|  935.6953|            8|F6wX3IQod5FjevAav...|
     23| 48759|     876.8|            8|2josHauNyMVu61FIc...|
     24| 22908|  751.8085|            8|IB2dXIKbAQyJT2FLC...|
     31|  1798|   866.952|            9|onpzpl7iXPHfukkcB...|
     32| 31886|  641.0324|            6|evGed4NIb0hrKIyt6...|
     34| 10297|  656.7162|            7|19xyjTMM7N9G5l7xP...|
     35| 49188|  659.2036|            6|V4NeEWYPHBKWIff8o...|
     36| 33289| 828.27814|           10|thBWip8PTlNbWf7RS...|
     38| 38967|  921.6777|            5|pyOc96MQHmgcp7d0T...|
     42| 32542| 644.41766|            6|Pz1FQGvXkYK62JkfO...|
     43| 47674| 862.99664|            8|CWjByZCHM3sGusolu...|
-------+------+----------+-------------+--------------------+
nly showing top 20 rows
```

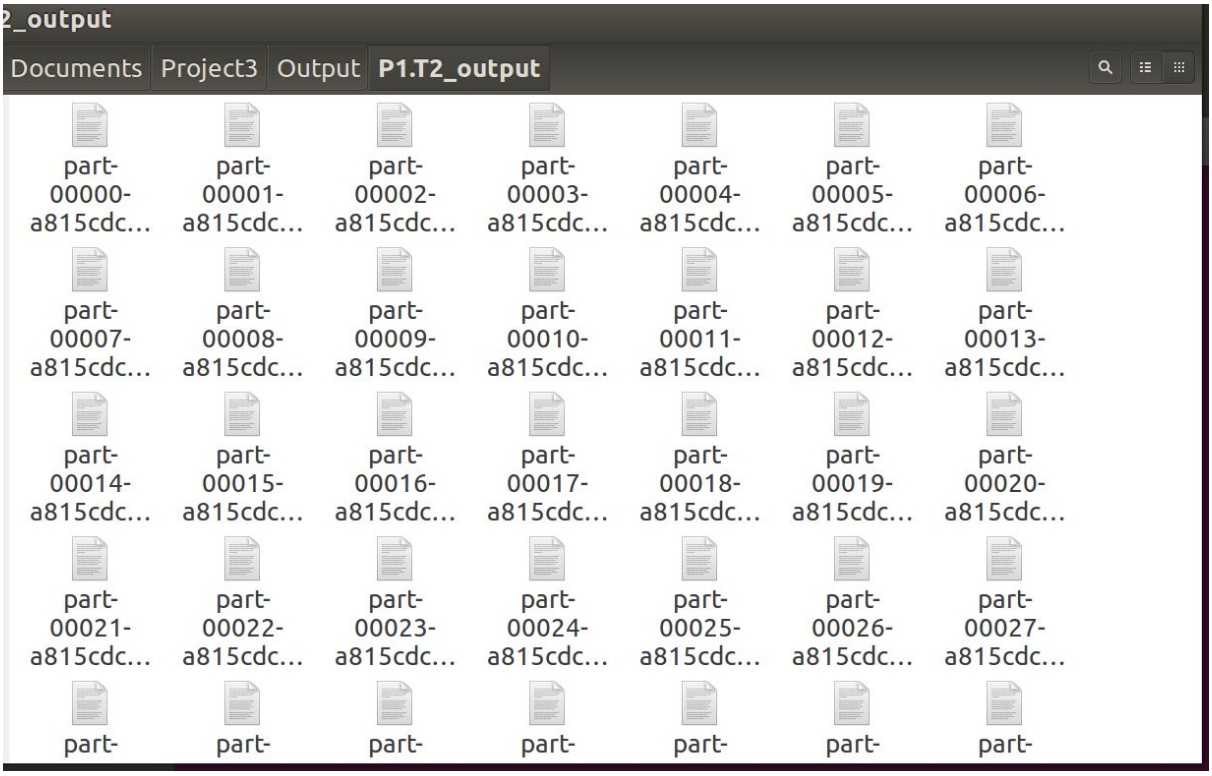
For the T6 we join the T5 and T3 with the key of CustID. Then conduct a simple SQL query base on the jointable to get the final result.

Screenshot of T6:



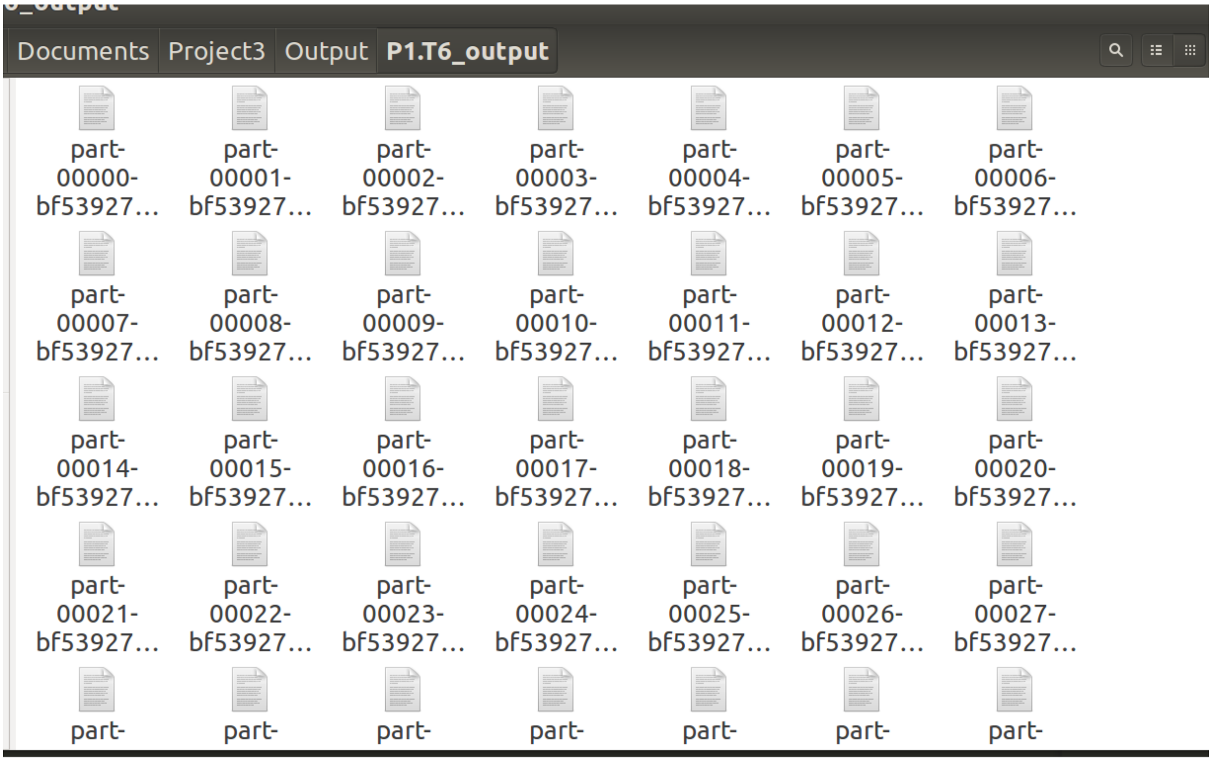
```
------+------------+------------+
CustID|T5_NUM_TRANS|T3_NUM_TRANS|
------+------------+------------+
 14227|          22|          74|
 44629|          26|          80|
 11393|          27|          82|
 24051|          25|          77|
 45069|          27|          83|
 28436|          26|          79|
 25870|          23|          74|
 23088|          15|          60|
 28786|          24|          77|
 47806|          25|          76|
 34246|          21|          68|
 34831|          24|          76|
  7066|          23|          71|
 31053|          21|          68|
   556|          16|          51|
 15160|          23|          72|
 35237|          28|          87|
  2709|          23|          70|
 20226|          21|          70|
   390|          24|          76|
------+------------+------------+
```

Screen of T2 output:



```
part-00000-a815cdc…  part-00001-a815cdc…  part-00002-a815cdc…  part-00003-a815cdc…  part-00004-a815cdc…  part-00005-a815cdc…  part-00006-a815cdc…
part-00007-a815cdc…  part-00008-a815cdc…  part-00009-a815cdc…  part-00010-a815cdc…  part-00011-a815cdc…  part-00012-a815cdc…  part-00013-a815cdc…
part-00014-a815cdc…  part-00015-a815cdc…  part-00016-a815cdc…  part-00017-a815cdc…  part-00018-a815cdc…  part-00019-a815cdc…  part-00020-a815cdc…
part-00021-a815cdc…  part-00022-a815cdc…  part-00023-a815cdc…  part-00024-a815cdc…  part-00025-a815cdc…  part-00026-a815cdc…  part-00027-a815cdc…
part-  part-  part-  part-  part-  part-  part-
```

Screen of the final output:



```
part-00000-bf53927…  part-00001-bf53927…  part-00002-bf53927…  part-00003-bf53927…  part-00004-bf53927…  part-00005-bf53927…  part-00006-bf53927…
part-00007-bf53927…  part-00008-bf53927…  part-00009-bf53927…  part-00010-bf53927…  part-00011-bf53927…  part-00012-bf53927…  part-00013-bf53927…
part-00014-bf53927…  part-00015-bf53927…  part-00016-bf53927…  part-00017-bf53927…  part-00018-bf53927…  part-00019-bf53927…  part-00020-bf53927…
part-00021-bf53927…  part-00022-bf53927…  part-00023-bf53927…  part-00024-bf53927…  part-00025-bf53927…  part-00026-bf53927…  part-00027-bf53927…
part-  part-  part-  part-  part-  part-  part-
```

Problem 2:

P2.A:

Runing generatedata.java

Our dataset contains a set of 2D points and is about 146M large.

P2.B:

Idea: map each point to (supercellID, (1,0, number of neighbor cells)) and (neighborcellID, (0, 1, 0)). In this function, we find out how many points are in each super cell and its neighbor cells.

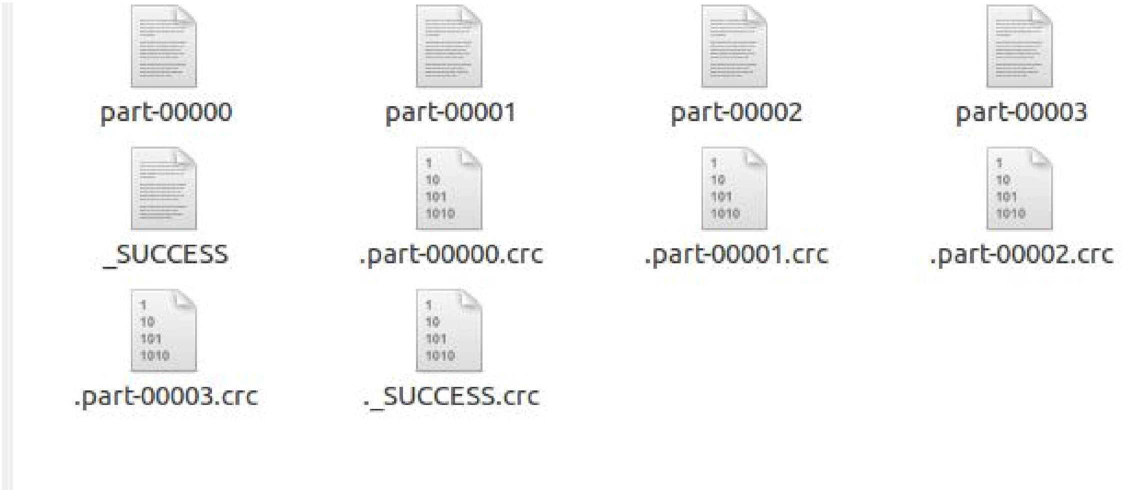
In the reduce function, if there is a value in the list of values whoes first position equals to one which means the point in that cell is not 0. In this situation, sum up all the positions except the last one. Otherwise, sum up all the positions.

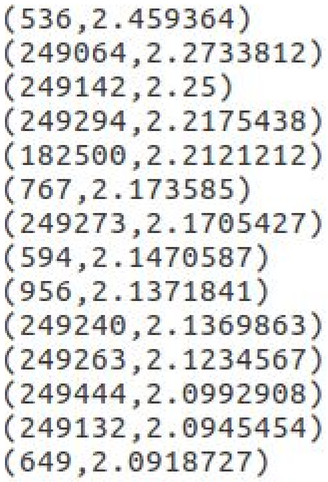
Next map function, calculate the relative-density scores for each cell. After that sort them.

Sample code:

```
./spark-submit --class bigdata.Problem2 /home/yifan/hw3.jar /home/yifan/spark/data.txt
```

The result of P2.B is stored in p2b file





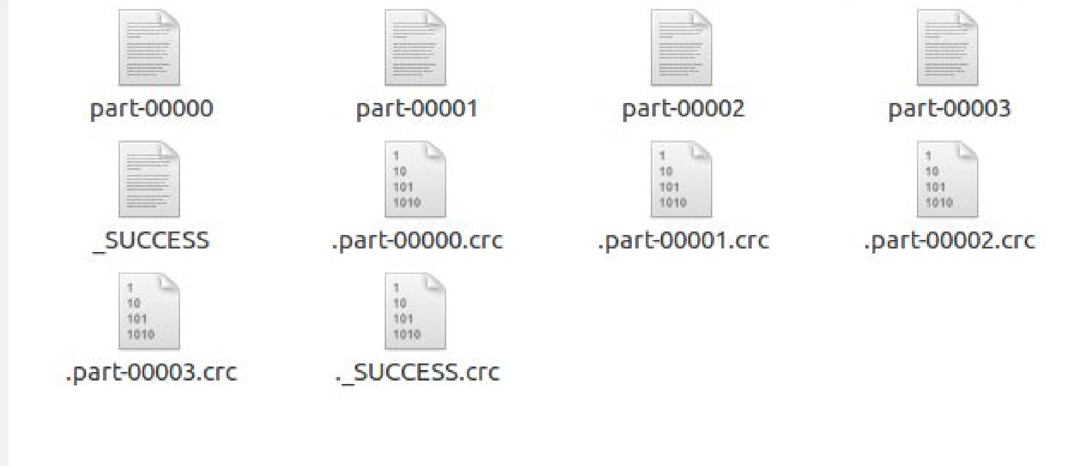
```
(536,2.459364)
(249064,2.2733812)
(249142,2.25)
(249294,2.2175438)
(182500,2.2121212)
(767,2.173585)
(249273,2.1705427)
(594,2.1470587)
(956,2.1371841)
(249240,2.1369863)
(249263,2.1234567)
(249444,2.0992908)
(249132,2.0945454)
(649,2.0918727)
```

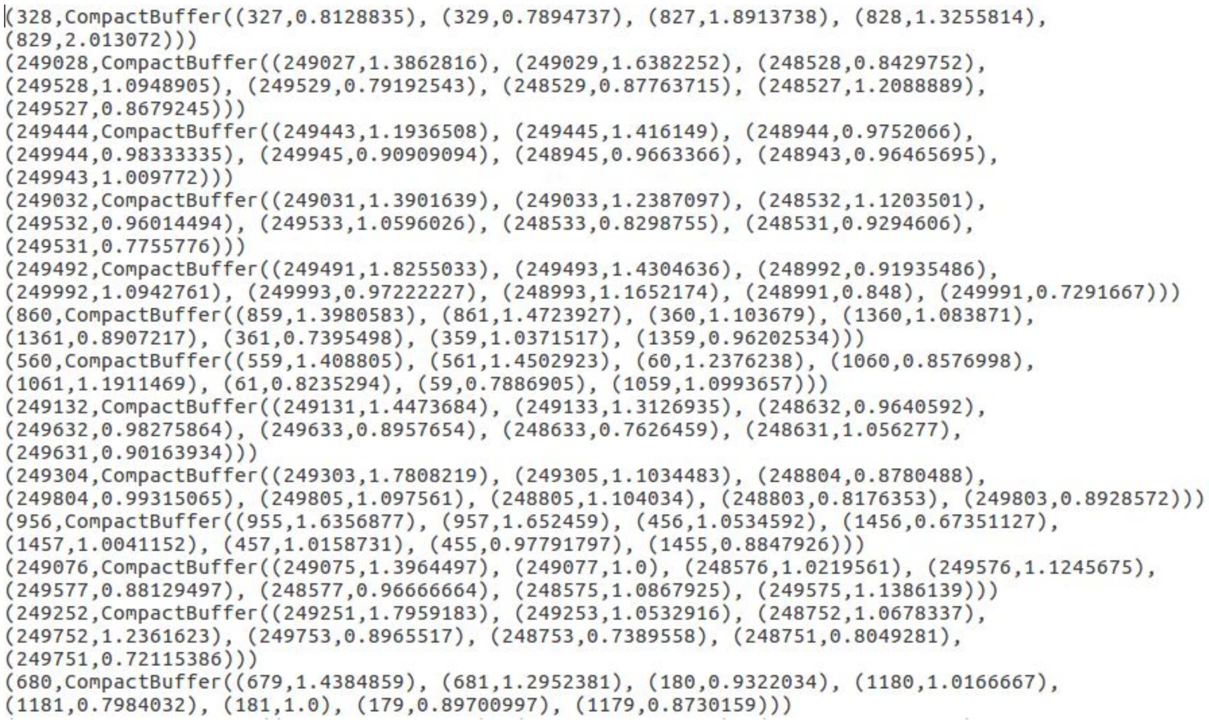
P2.C:

Idea: Get the neighbor cells for Top 100 super cells and find their neighbor cells' relative-density scores from above.

Sample code:

./spark-submit --class bigdata.Problem2 /home/yifan/hw3.jar /home/yifan/spark/data.txt
The result of P2.C is stored in p2c file





```
(328,CompactBuffer((327,0.8128835), (329,0.7894737), (827,1.8913738), (828,1.3255814),
(829,2.013072)))
(249028,CompactBuffer((249027,1.3862816), (249029,1.6382252), (248528,0.8429752),
(249528,1.0948905), (249529,0.79192543), (248529,0.87763715), (248527,1.2088889),
(249527,0.8679245)))
(249444,CompactBuffer((249443,1.1936508), (249445,1.416149), (248944,0.9752066),
(249944,0.98333335), (249945,0.90909094), (248945,0.9663366), (248943,0.96465695),
(249943,1.009772)))
(249032,CompactBuffer((249031,1.3901639), (249033,1.2387097), (248532,1.1203501),
(249532,0.96014494), (249533,1.0596026), (248533,0.8298755), (248531,0.9294606),
(249531,0.7755776)))
(249492,CompactBuffer((249491,1.8255033), (249493,1.4304636), (248992,0.91935486),
(249992,1.0942761), (249993,0.97222227), (248993,1.1652174), (248991,0.848), (249991,0.7291667)))
(860,CompactBuffer((859,1.3980583), (861,1.4723927), (360,1.103679), (1360,1.083871),
(1361,0.8907217), (361,0.7395498), (359,1.0371517), (1359,0.96202534)))
(560,CompactBuffer((559,1.408805), (561,1.4502923), (60,1.2376238), (1060,0.8576998),
(1061,1.1911469), (61,0.8235294), (59,0.7886905), (1059,1.0993657)))
(249132,CompactBuffer((249131,1.4473684), (249133,1.3126935), (248632,0.9640592),
(249632,0.98275864), (249633,0.8957654), (248633,0.7626459), (248631,1.056277),
(249631,0.90163934)))
(249304,CompactBuffer((249303,1.7808219), (249305,1.1034483), (248804,0.8780488),
(249804,0.99315065), (249805,1.097561), (248805,1.104034), (248803,0.8176353), (249803,0.8928572)))
(956,CompactBuffer((955,1.6356877), (957,1.652459), (456,1.0534592), (1456,0.67351127),
(1457,1.0041152), (457,1.0158731), (455,0.97791797), (1455,0.8847926)))
(249076,CompactBuffer((249075,1.3964497), (249077,1.0), (248576,1.0219561), (249576,1.1245675),
(249577,0.88129497), (248577,0.96666664), (248575,1.0867925), (249575,1.1386139)))
(249252,CompactBuffer((249251,1.7959183), (249253,1.0532916), (248752,1.0678337),
(249752,1.2361623), (249753,0.8965517), (248753,0.7389558), (248751,0.8049281),
(249751,0.72115386)))
(680,CompactBuffer((679,1.4384859), (681,1.2952381), (180,0.9322034), (1180,1.0166667),
(1181,0.7984032), (181,1.0), (179,0.89700997), (1179,0.8730159)))
```

Contribution:

Both Brendan and Jiaxing worked on the problem 1 with Scala. And we used diferent approach to do it. Brendan choose to query directly on the data frame, while Jiaxing mostly create a SQL table structure then query on the table. And both of the two codes worked well.

Xiaoting finished problem 2 in Scala and Java. Write report for problem 2 and review Jiaxing's code for problem2.

The problem 2 codes include two parts of code, the P2B_Jack was did by Jiaxing Zhang with Scala. The other codes was did by Xiaoting including one Scala solution and one Java solution. You just need to copy the code of P2B_Jack to the terminal in order to run it. For the rest codes which finished by Xiaoting Cui, you need to do in the Spark way.