Project 1 Documentation
Emily Hao, eyhao@wpi.edu
Daojun Liang, dliang2@wpi.edu

Create datasets
We attached the java code for creating datasets: customer.txt, transactions.txt
Note: transactions.txt takes a bit of time to create.

Uploading data into Hadoop
We used the command
$ CD into where the datasets located in the folder
$ Hadoop fs –put customer.txt /user/Hadoop/input
$ Hadoop fs –put transactions.txt /user/Hadoop/input

MapReduce Jobs

Environment: Intellij + Maven

Query 1:
This is a map-only job.
In the map, the output key is <Text> name of customers; and the output value is <NullWritable> so that we can get the result of the customers' names.

The output is below (the first ten rows and the last ten rows):

```
1     jtauateibbztzr
2     bujfcnlgmvyphrnl
3     xxediscckywryzozakoh
4     loqvnavdqor
5     vfsrsyzxtqzsyiqmefx
6     eyqabdvqwz
7     wrrazhqznuzakhkg
8     ghvsdoaexa
9     ljpuqtrdskuezs
10    zaxdntwnas
```

```
25447    pzuhwerpjprlwumn
25448    xdluczldaepsibmjp
25449    ieodatwfdguhyta
25450    ithsxifukbzcbdmlpssg
25451    ljvdlqtlwi
25452    fdsogjnthbxyza
25453    lhfsewibbbxbcj
25454    xczeliravsblk
25455    xondosopmwvkaxyznoa
25456    zscqiufldq
```

We have 25456 customers whose age is between 20 and 50.

Query 2
We first put the customer.txt into cache.
We use one map function and one reduce function.

In the map function, we join the customer.txt and transaction.txt.
The output key is: customerID
The output value is: <Text> name + “1” + transTotal

In the reduce function, we calculate the numOfTransaction and sumTotal.

The result is below (the first ten rows and the last ten rows):

```
1     1   jtauateibbztzr,100,52615.125
2     10  ngvkcxkfetv,105,46961.566
3     100 ghirnuedsverdgqbu,104,51930.223
4     1000    conrlufrqgrcvi,122,57998.07
5     10000   vmkdmdxckdii,86,46175.586
6     10001   eikfdntaxzroewnwnafw,102,49730.52
7     10002   zcmzwemlrk,103,44929.996
8     10003   xkqxlmdvaj,101,44697.953
9     10004   eyozezyauddyyvwpoyw,93,46224.77
10    10005   kqohecrplb,88,47995.45
```

```
49991 9990    gzjehwnrcokqndancz,90,48094.47
49992 9991    bcirgfcqwzort,89,48455.156
49993 9992    sgynosinkgwa,103,55447.875
49994 9993    mjnyzldpdg,99,50022.812
49995 9994    piwhpxjnkiash,110,53594.08
49996 9995    cjprkfhooclahbibr,110,48725.254
49997 9996    vlrfpgbgptjhmq,98,48350.48
49998 9997    bfcxfpwgru,98,53254.94
49999 9998    xpbvkiltcaedwqpvien,80,37793.207
50000 9999    zroqpohbpdwwklnzzuj,97,50701.19
```

Query 3
We use two maps to read the customer.txt and transaction.txt. The customerMapper is responsible for reading the customer.txt and the transactionMapper is responsible for reading the transaction.txt.

In the customerMapper:
The output key is customerID.
The output value is name, salary.

In the transactionMapper:
The output key is customerID.

The output value is transTotal transNumItems.

In the reduce phase, we join the output value of customerMapper and transactionMapper using the same key - customerID.

In the reduce:
The output key is customerID.
The output value is name, salary, numTransaction, totalSum, minItems.

The output is below (the first ten rows and the last ten rows):

```
1   jtauateibbztzr,6174.858,100,52615.13,1
10  ngvkcxkfetv,2540.4087,105,46961.562,1
100 ghirnuedsverdgqbu,6077.049,104,51930.23,1
1000    conrlufrqgrcvi,1965.8826,122,57998.074,1
10000   vmkdmdxckdii,1174.788,86,46175.59,1
10001   eikfdntaxzroewnwnafw,4951.029,102,49730.516,1
10002   zcmzwemlrk,6536.537,103,44929.99,1
10003   xkqxlmdvaj,5015.994,101,44697.957,1
10004   eyozezyauddyyvwpoyw,6189.213,93,46224.777,1
10005   kqohecrplb,6059.0522,88,47995.457,1
```

```
49990 999 lqtibondxl,1036.5986,126,60714.29,1
49991 9990    gzjehwnrcokqndancz,3662.1284,90,48094.465,1
49992 9991    bcirgfcqwzort,9258.528,89,48455.152,1
49993 9992    sgynosinkgwa,9993.744,103,55447.883,1
49994 9993    mjnyzldpdg,7181.8213,99,50022.812,1
49995 9994    piwhpxjnkiash,1876.3394,110,53594.082,1
49996 9995    cjprkfhooclahbibr,3031.0732,110,48725.258,1
49997 9996    vlrfpgbgptjhmq,1410.7155,98,48350.47,1
49998 9997    bfcxfpwgru,7044.8276,98,53254.945,1
49999 9998    xpbvkiltcaedwqpvien,4131.915,80,37793.207,1
50000 9999    zroqpohbpdwwklnzzuj,115.256676,97,50701.203,1
```

Query 4
We first put customer.txt into cache.
In this query, we use two maps: Mapper and SecondMapper. And we have one reduce: IntSumReducer.

In the Mapper (the first map):
The output key is customerID.
The output value is name (from customer.txt), transTotal (from the transaction.txt)

In the SecondMapper (the second map):
The output key is CountryCode (from customer.txt).
The output value is a tag: “customer” (type: String). We use this tag in the reduce phase to identify which <key, value> we have.

In the reduce:
If we read data from the SecondMapper(the second map), then we can calculate the number of transactions. If we read data from the Mapper (the first map), then we can calculate the minTransTotal and the maxTransTotal.

In this way, we only use one job (two map and one reduce) to get the final result.

The result is below (we only have ten rows because we only have ten country codes):

```
1    5031,10.001534,999.9955
10   4900,10.002655,999.9977
2    5037,10.001534,999.9994
3    4967,10.000118,999.99854
4    4987,10.002242,999.993
5    5003,10.001416,999.9993
6    5012,10.003599,999.99963
7    5047,10.0011215,999.99536
8    4996,10.000826,999.99915
9    5020,10.002773,999.9991
```

Query 5
We first put customer.txt into cache.
We have only one map and one reduce in this query.

In the map:
The output key is: <Text> AgeRange + “,” + gender
The output value is: transTotal (from the transaction.txt).

In the reduce:
The output key is: <Text> AgeRange + “,” + gender
The output value is: minTransTotal, maxTransTotal, avgTransTotal.

The result is below (we have 12 rows because we have 6 groups of age range and 2 kinds of gender):

```
[10,20),female  10.005134,999.99963,505.0725
[10,20),male    10.004602,999.99915,505.051
[20,30),female  10.003599,999.99835,505.46198
[20,30),male    10.000826,999.99725,505.1287
[30,40),female  10.004484,999.99554,504.7009
[30,40),male    10.0011215,999.9993,504.88217
[40,50),female  10.000354,999.99536,504.70895
[40,50),male    10.004249,999.9988,505.09467
[50,60),female  10.009264,999.9955,505.746
[50,60),male    10.001416,999.99866,505.4535
[60,70],female  10.000118,999.9986,504.82663
[60,70],male    10.001181,999.9994,504.93118
```

Apache-Pig

Note:
All of my inputs are loaded from the path 'input/_____'
All of my outputs are stored to the path 'output/_____'
Feel free to change the path if needed.

I attached the 4 Apache-Pig scripts in the folder.
To run the scripts,
$ Sudo start-all.sh
$ Pig
you can either paste the whole script into grunt command.
Or
$ Sudo start-all.sh
$ cd <to the folder>
$ Pig PigQuery1.pig
$ Pig PigQuery2.pig
$ Pig PigQuery3.pig
$ Pig PigQuery4.pig

Also note the output for query 4 is in output/PigQuery4, but in multiple files, each file contains the information for each age group.