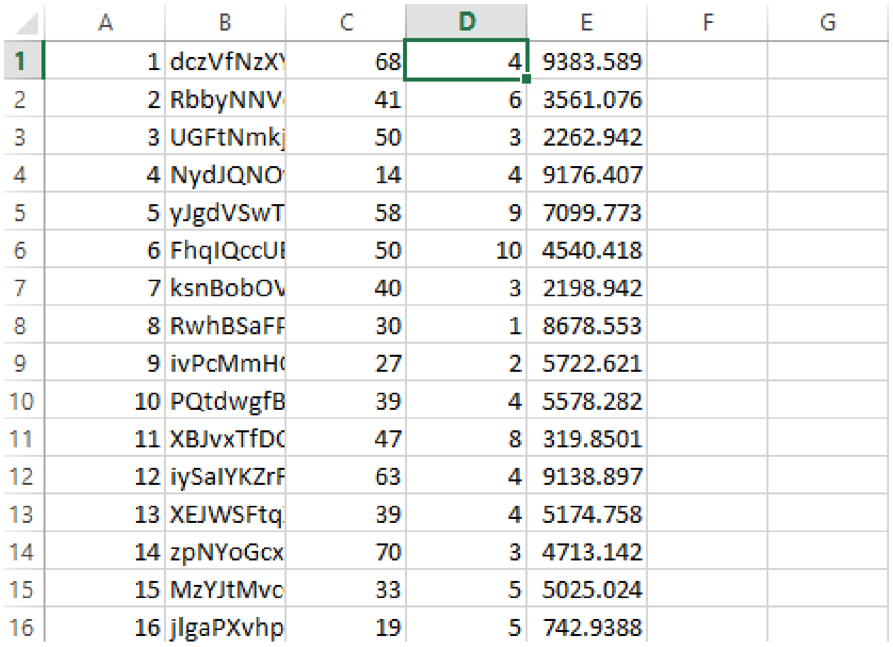
Project 1 February 12, 2017

Big Data Management Spring 2017

Team 14 Wenlei Cao and Caitlin Kuhlman

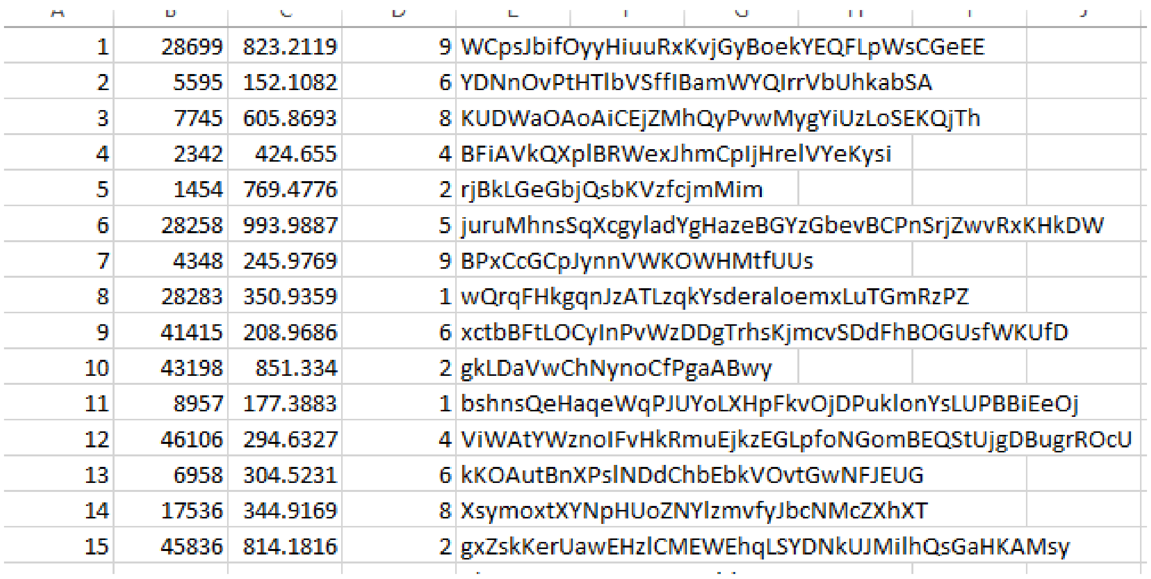
1. The file Q1/DataGenerator.java generates two data files: customers.csv and transactions.csv. Below are screenshots of the resulting data.

Customers Table



	A	B	C	D	E	F	G
1	1	dczVfNzX	68	4	9383.589		
2	2	RbbyNNV	41	6	3561.076		
3	3	UGFtNmkj	50	3	2262.942		
4	4	NydJQNO	14	4	9176.407		
5	5	yJgdVSwT	58	9	7099.773		
6	6	FhqIQccUI	50	10	4540.418		
7	7	ksnBobOV	40	3	2198.942		
8	8	RwhBSaFF	30	1	8678.553		
9	9	ivPcMmH	27	2	5722.621		
10	10	PQtdwgfB	39	4	5578.282		
11	11	XBJvxTfD	47	8	319.8501		
12	12	iySaIYKZrF	63	4	9138.897		
13	13	XEJWSFtq	39	4	5174.758		
14	14	zpNYoGcx	70	3	4713.142		
15	15	MzYJtMvc	33	5	5025.024		
16	16	jlgaPXvhp	19	5	742.9388		

Transactions Table



A	B	C	D	E
1	28699	823.2119	9	WCpsJbifOyyHiuuRxKvjGyBoekYEQFLpWsCGeEE
2	5595	152.1082	6	YDNnOvPtHTlbVSffIBamWYQIrrVbUhkabSA
3	7745	605.8693	8	KUDWaOAoAiCEjZMhQyPvwMygYiUzLoSEKQjTh
4	2342	424.655	4	BFiAVkQXplBRWexJhmCpljHrelVYeKysi
5	1454	769.4776	2	rjBkLGeGbjQsbKVzfcjmMim
6	28258	993.9887	5	juruMhnsSqXcgyladYgHazeBGYzGbevBCPnSrjZwvRxKHkDW
7	4348	245.9769	9	BPxCcGCpJynnVWKOWHMtfUUs
8	28283	350.9359	1	wQrqFHkgqnJzATLzqkYsderaloemxLuTGmRzPZ
9	41415	208.9686	6	xctbBFtLOCyInPvWzDDgTrhsKjmcvSDdFhBOGUsfWKUfD
10	43198	851.334	2	gkLDaVwChNynoCfPgaABwy
11	8957	177.3883	1	bshnsQeHaqeWqPJUYoLXHpFkvOjDPuklonYsLUPBBiEeOj
12	46106	294.6327	4	ViWAtYWznoIFvHkRmuEjkzEGLpfoNGomBEQStUjgDBugrROcU
13	6958	304.5231	6	kKOAutBnXPslNDdChbEbkVOvtGwNFJEUG
14	17536	344.9169	8	XsymoxtXYNpHUoZNYlzmvfyJbcNMcZXhXT
15	45836	814.1816	2	gxZskKerUawEHzlCMEWEhqLSYDNkUJMilhQsGaHKAMsy

3 Mapreduce Jobs.

We have a single package Q3 which contains code for all the MapReduce queries. All the jobs require command line arguments to run secifying input and output paths. This is the usage for each job:

3.1 hadoop jar <JARFILE> Q3.Query1 <PATH TO CUSTOMERS> <OUTPUT PATH>

3.2 hadoop jar <JARFILE> Q3.Query2 <PATH TO TRANSACTIONS> <OUTPUT PATH>

3.3 hadoop jar <JARFILE> Q3.Query3 <PATH TO CUSTOMERS> <PATH TO TRANSACTIONS> <OUTPUT PATH>

Or

3.3 hadoop jar <JARFILE> Q3.query3_w.query3 <PATH TO CUSTOMERS> <PATH TO TRANSACTIONS> <OUTPUT PATH>

3.4 hadoop jar <JARFILE> Q3.Query4 <PATH TO CUSTOMERS> <PATH TO TRANSACTIONS> <OUTPUT PATH>

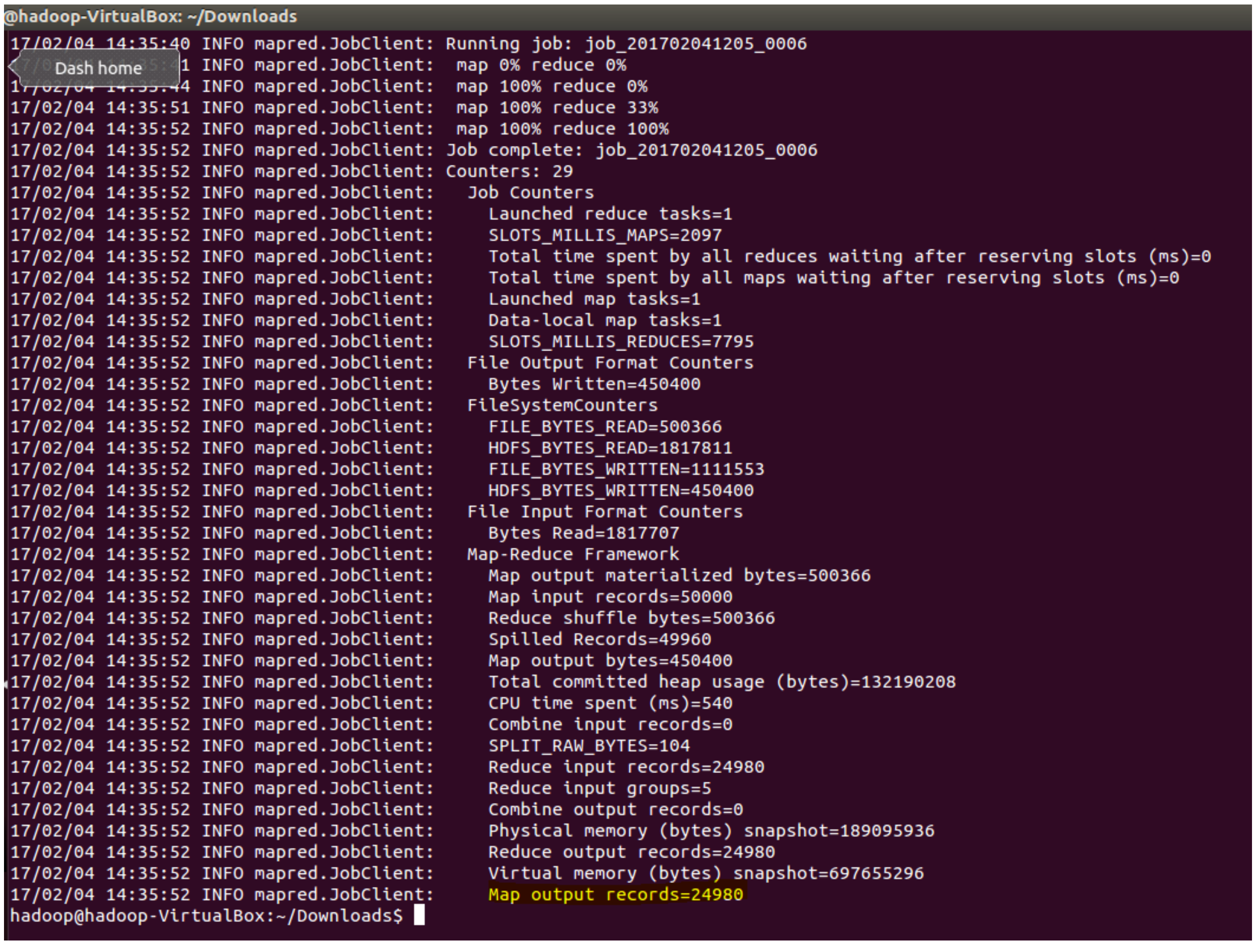
Or

3.4 hadoop jar <JARFILE> Q3.query4.Query4 <PATH TO CUSTOMERS> <PATH TO TRANSACTIONS> <OUTPUT PATH>

3.5 hadoop jar <JARFILE> Q3.query5.Query5 <PATH TO CUSTOMERS> <PATH TO TRANSACTIONS> <OUTPUT PATH>

3.1

In Q3/Query1.java a single job reports the customers whose CountryCode is between 2 and 6 (inclusive). This is completed using only a map phase and no reducer. The output is the same format as in the customers file. Below is a screenshot of the output form this hadoop job. In our generated file there were 24,980 records out of 50,000 customer with country code between 2-6.



```
17/02/04 14:35:40 INFO mapred.JobClient: Running job: job_201702041205_0006
17/02/04 14:35:41 INFO mapred.JobClient:  map 0% reduce 0%
17/02/04 14:35:44 INFO mapred.JobClient:  map 100% reduce 0%
17/02/04 14:35:51 INFO mapred.JobClient:  map 100% reduce 33%
17/02/04 14:35:52 INFO mapred.JobClient:  map 100% reduce 100%
17/02/04 14:35:52 INFO mapred.JobClient: Job complete: job_201702041205_0006
17/02/04 14:35:52 INFO mapred.JobClient: Counters: 29
17/02/04 14:35:52 INFO mapred.JobClient:   Job Counters
17/02/04 14:35:52 INFO mapred.JobClient:     Launched reduce tasks=1
17/02/04 14:35:52 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=2097
17/02/04 14:35:52 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
17/02/04 14:35:52 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
17/02/04 14:35:52 INFO mapred.JobClient:     Launched map tasks=1
17/02/04 14:35:52 INFO mapred.JobClient:     Data-local map tasks=1
17/02/04 14:35:52 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=7795
17/02/04 14:35:52 INFO mapred.JobClient:   File Output Format Counters
17/02/04 14:35:52 INFO mapred.JobClient:     Bytes Written=450400
17/02/04 14:35:52 INFO mapred.JobClient:   FileSystemCounters
17/02/04 14:35:52 INFO mapred.JobClient:     FILE_BYTES_READ=500366
17/02/04 14:35:52 INFO mapred.JobClient:     HDFS_BYTES_READ=1817811
17/02/04 14:35:52 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=1111553
17/02/04 14:35:52 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=450400
17/02/04 14:35:52 INFO mapred.JobClient:   File Input Format Counters
17/02/04 14:35:52 INFO mapred.JobClient:     Bytes Read=1817707
17/02/04 14:35:52 INFO mapred.JobClient:   Map-Reduce Framework
17/02/04 14:35:52 INFO mapred.JobClient:     Map output materialized bytes=500366
17/02/04 14:35:52 INFO mapred.JobClient:     Map input records=50000
17/02/04 14:35:52 INFO mapred.JobClient:     Reduce shuffle bytes=500366
17/02/04 14:35:52 INFO mapred.JobClient:     Spilled Records=49960
17/02/04 14:35:52 INFO mapred.JobClient:     Map output bytes=450400
17/02/04 14:35:52 INFO mapred.JobClient:     Total committed heap usage (bytes)=132190208
17/02/04 14:35:52 INFO mapred.JobClient:     CPU time spent (ms)=540
17/02/04 14:35:52 INFO mapred.JobClient:     Combine input records=0
17/02/04 14:35:52 INFO mapred.JobClient:     SPLIT_RAW_BYTES=104
17/02/04 14:35:52 INFO mapred.JobClient:     Reduce input records=24980
17/02/04 14:35:52 INFO mapred.JobClient:     Reduce input groups=5
17/02/04 14:35:52 INFO mapred.JobClient:     Combine output records=0
17/02/04 14:35:52 INFO mapred.JobClient:     Physical memory (bytes) snapshot=189095936
17/02/04 14:35:52 INFO mapred.JobClient:     Reduce output records=24980
17/02/04 14:35:52 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=697655296
17/02/04 14:35:52 INFO mapred.JobClient:     Map output records=24980
hadoop@hadoop-VirtualBox:~/Downloads$
```

3.2

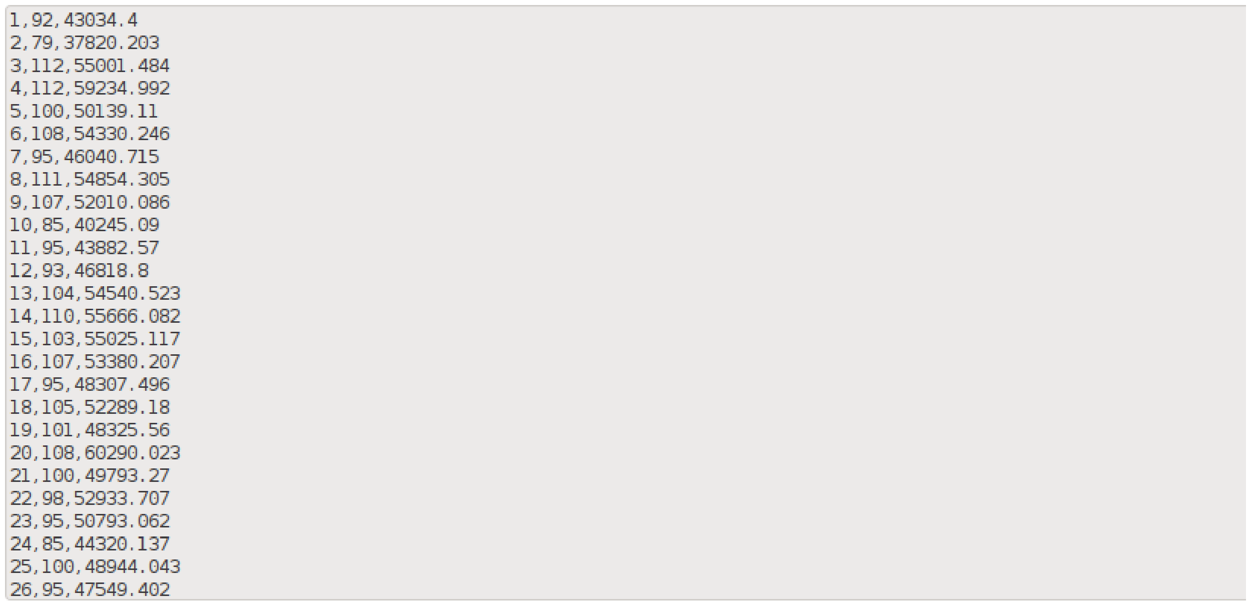
Q3/Query2.java reports for every customer, the number of transactions and the total sum of these transactions. This is done using a custom datatype called TransCountWritable.java which implements the WritableComparable interface in order to hold intermediate values.

The TransCountWritable object holds the transaction data for each customer. In the mapper, for each transaction, the output key is the customer id and the output value is a new TransCountWritable object which holds the totalTrans value for this one transaction.

A combiner is used to aggregate the count and total of transactions for one customer. It is a separate class from the reducer which reads in one customer id and a list of associated TransCountWritable objects. The counts and values of the transactions are aggregated and output in a single TransCountWritable object.

The same aggregation is performed in the reducer, but the output is a NullWritable key (empty) and a Text value with the new record for this customer. The output format is : CustomerID, NumTransactions, TotalSum.

Below is a sample screenshot of the output.



```
1,92,43034.4
2,79,37820.203
3,112,55001.484
4,112,59234.992
5,100,50139.11
6,108,54330.246
7,95,46040.715
8,111,54854.305
9,107,52010.086
10,85,40245.09
11,95,43882.57
12,93,46818.8
13,104,54540.523
14,110,55666.082
15,103,55025.117
16,107,53380.207
17,95,48307.496
18,105,52289.18
19,101,48325.56
20,108,60290.023
21,100,49793.27
22,98,52933.707
23,95,50793.062
24,85,44320.137
25,100,48944.043
26,95,47549.402
```

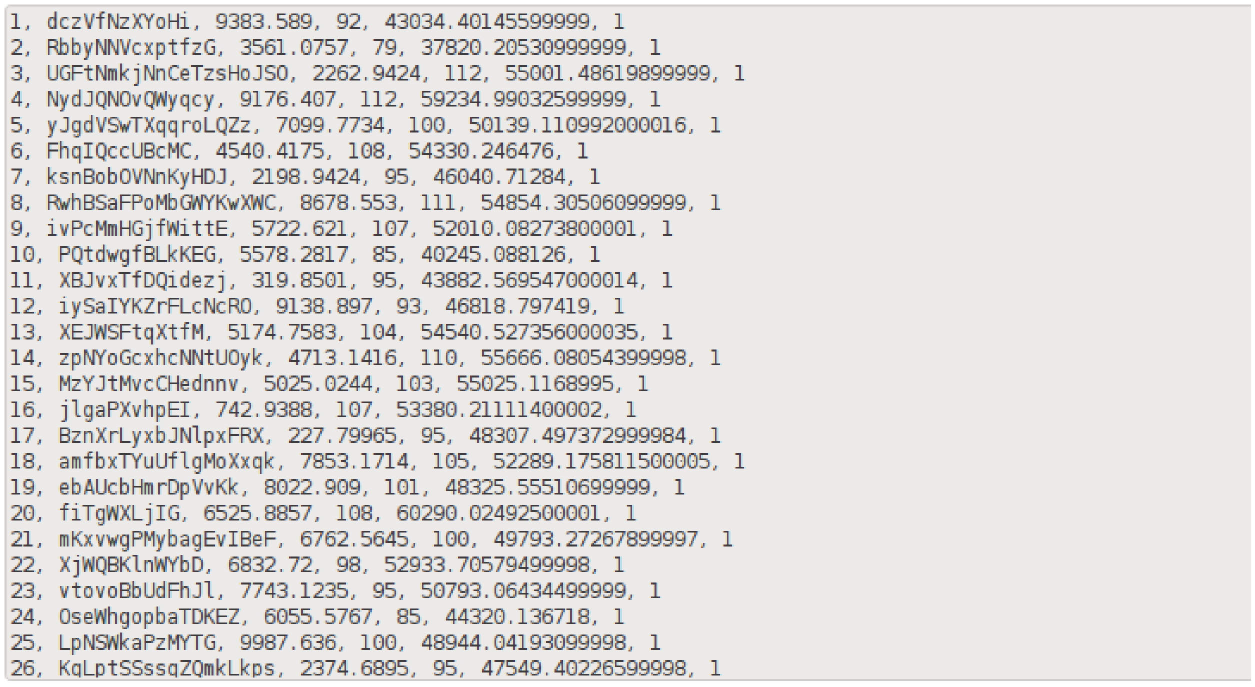
3.3

For Query 3 we completed 2 alternative implementations. Q3/Query3.java also uses the TransCountWritable object to complete the join. MultipleInputs are used to read from customers and transactions files and separate mappers handle each. Both mappers output the key CustomerId, and the results are joined in the reducer.

An alternative implementation is found in package Q3.query3_w. The driver file is query3.java. It follows the logic below:

```
Select
c.customerID,
c.name,
c.salary,
Count(t.transID),
Sum(t.transtotal),
Min (t.transNumItems)
From  customers   c
Join transactions   t on c.ID = t.custID
```

Here is a sample output:



```
1, dczVfNzXYoHi, 9383.589, 92, 43034.40145599999, 1
2, RbbyNNVcxptfzG, 3561.0757, 79, 37820.20530999999, 1
3, UGFtNmkjNnCeTzsHoJSO, 2262.9424, 112, 55001.48619899999, 1
4, NydJQNOvQWyqcy, 9176.407, 112, 59234.99032599999, 1
5, yJgdVSwTXqqroLQZz, 7099.7734, 100, 50139.110992000016, 1
6, FhqIQccUBcMC, 4540.4175, 108, 54330.246476, 1
7, ksnBobOVNnKyHDJ, 2198.9424, 95, 46040.71284, 1
8, RwhBSaFPoMbGWYKwXWC, 8678.553, 111, 54854.30506099999, 1
9, ivPcMmHGjfWittE, 5722.621, 107, 52010.08273800001, 1
10, PQtdwgfBLkKEG, 5578.2817, 85, 40245.088126, 1
11, XBJvxTfDQidezj, 319.8501, 95, 43882.569547000014, 1
12, iySaIYKZrFLcNcRO, 9138.897, 93, 46818.797419, 1
13, XEJWSFtqXtfM, 5174.7583, 104, 54540.527356000035, 1
14, zpNYoGcxhcNNtUOyk, 4713.1416, 110, 55666.08054399998, 1
15, MzYJtMvcCHednnv, 5025.0244, 103, 55025.1168995, 1
16, jlgaPXvhpEI, 742.9388, 107, 53380.21111400002, 1
17, BznXrLyxbJNlpxFRX, 227.79965, 95, 48307.497372999984, 1
18, amfbxTYuUflgMoXxqk, 7853.1714, 105, 52289.175811500005, 1
19, ebAUcbHmrDpVvKk, 8022.909, 101, 48325.55510699999, 1
20, fiTgWXLjIG, 6525.8857, 108, 60290.02492500001, 1
21, mKxvwgPMybagEvIBeF, 6762.5645, 100, 49793.27267899997, 1
22, XjWQBKlnWYbD, 6832.72, 98, 52933.70579499998, 1
23, vtovoBbUdFhJl, 7743.1235, 95, 50793.06434499999, 1
24, OseWhgopbaTDKEZ, 6055.5767, 85, 44320.136718, 1
25, LpNSWkaPzMYTG, 9987.636, 100, 48944.04193099998, 1
26, KqLptSSssqZQmkLkps, 2374.6895, 95, 47549.40226599998, 1
```

3.4

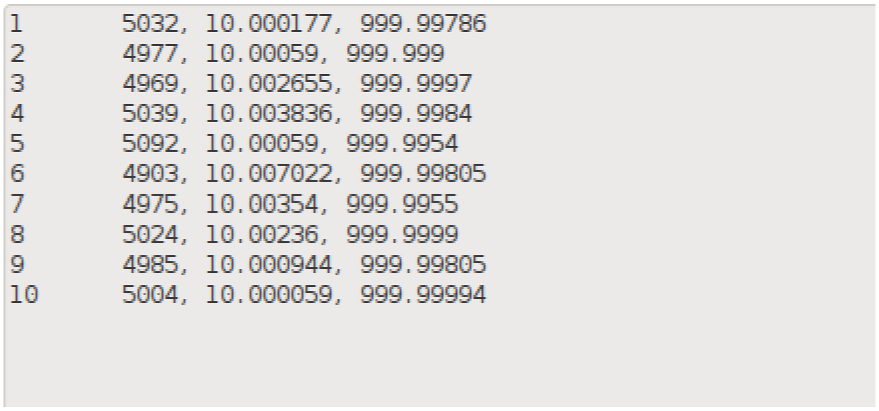
For this task we also implemented 2 alternate versions. In Q3.Query4.java a broadcast join is used to complete the task in a single MapReduce job. Customer info is read in by each Map task and put into a hashtable in a setup function. An object called a Trans4Writable is used to hold intermediate data.

In the package Q3.query4 the same task is implemented with 2 MapReduce jobs. The implementation follows the logic below:

Output format: CountryCode, NumberOfCustomers, MinTransTotal, MaxTransTotal

```
Select
c.countrycode,
Count(c.customerID),
min(t.transtotal),
max(t.transtotal)
From  customers   c
Join transactions   t on c.ID = t.custID
Group by c.countrycode
```

Sample output:



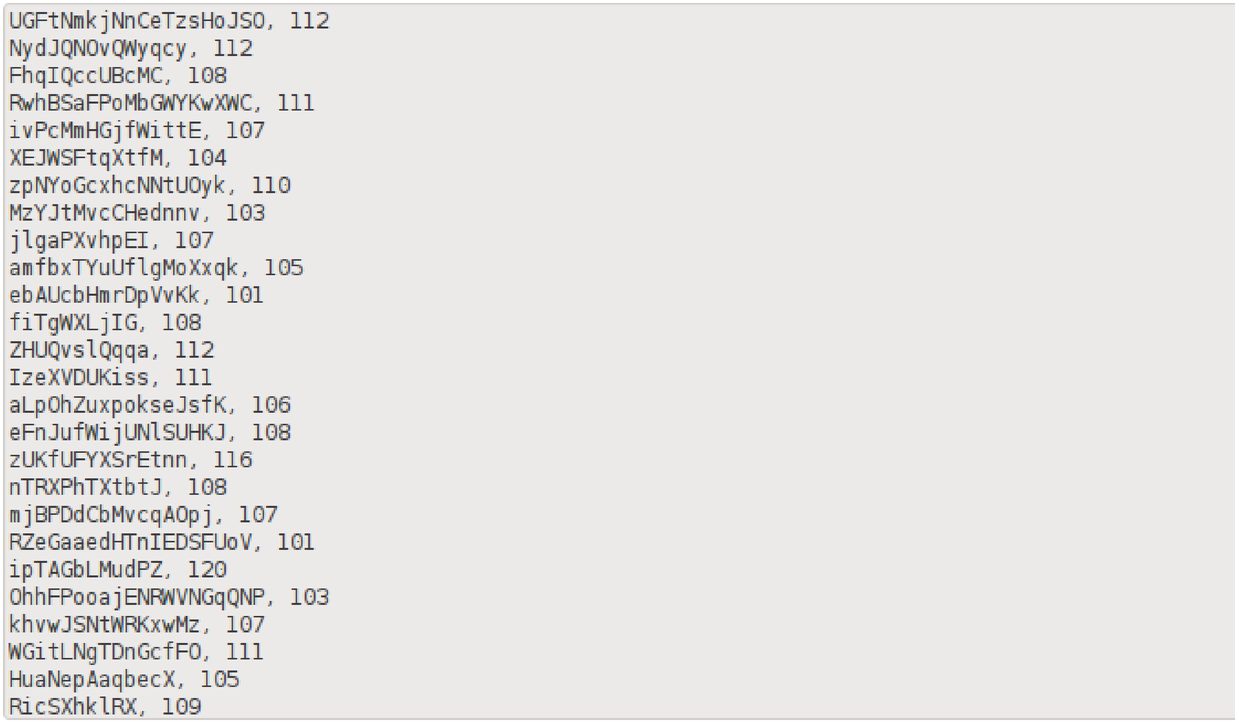
```
1       5032, 10.000177, 999.99786
2       4977, 10.00059, 999.999
3       4969, 10.002655, 999.9997
4       5039, 10.003836, 999.9984
5       5092, 10.00059, 999.9954
6       4903, 10.007022, 999.99805
7       4975, 10.00354, 999.9955
8       5024, 10.00236, 999.9999
9       4985, 10.000944, 999.99805
10      5004, 10.000059, 999.99994
```

3.5

The implementation for query 5 is found in package Q3.query5

Logic

```
Select
c.customername,
count (t.transactioniD)
From  customers   c
Join transactions   t on c.ID = t.custID
Group by c.customername
having count(t.transactioniD) >100   (5000000/50000)
```



```
UGFtNmkjNnCeTzsHoJSO, 112
NydJQNOvQWyqcy, 112
FhqIQccUBcMC, 108
RwhBSaFPoMbGWYKwXWC, 111
ivPcMmHGjfWittE, 107
XEJWSFtqXtfM, 104
zpNYoGcxhcNNtUOyk, 110
MzYJtMvcCHednnv, 103
jlgaPXvhpEI, 107
amfbxTYuUflgMoXxqk, 105
ebAUcbHmrDpVvKk, 101
fiTgWXLjIG, 108
ZHUQvslQqqa, 112
IzeXVDUKiss, 111
aLpOhZuxpokseJsfK, 106
eFnJufWijUNlSUHKJ, 108
zUKfUFYXSrEtnn, 116
nTRXPhTXtbtJ, 108
mjBPDdCbMvcqAOpj, 107
RZeGaaedHTnIEDSFUoV, 101
ipTAGbLMudPZ, 120
OhhFPooajENRWVNGqQNP, 103
khvwJSNtWRKxwMz, 107
WGitLNgTDnGcfFO, 111
HuaNepAaqbecX, 105
RicSXhklRX, 109
```

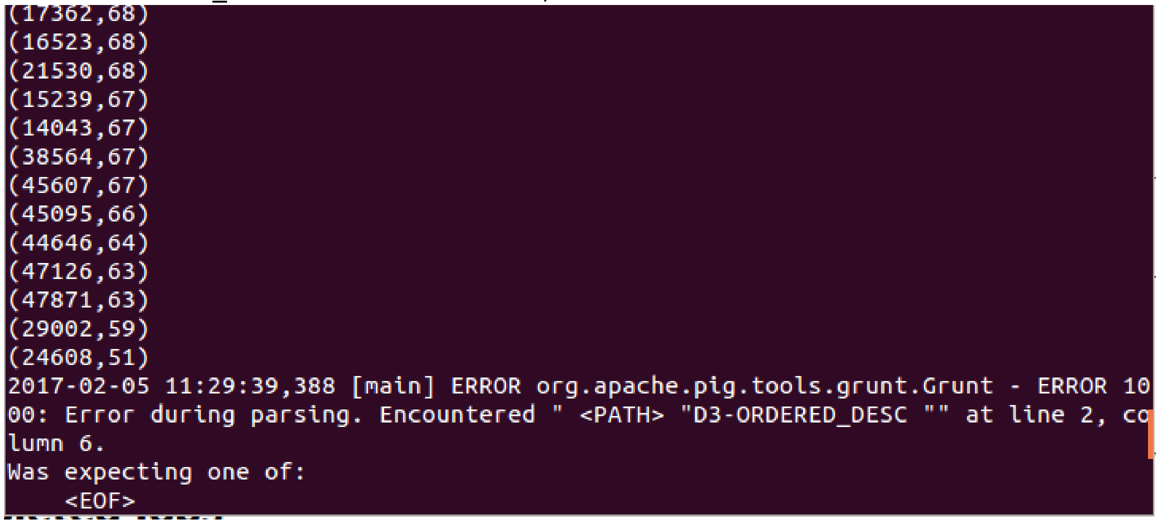
Apache Pig Queries

4.1

/*Write an Apache Pig query that reports the customer names that have the least number of transactions*/

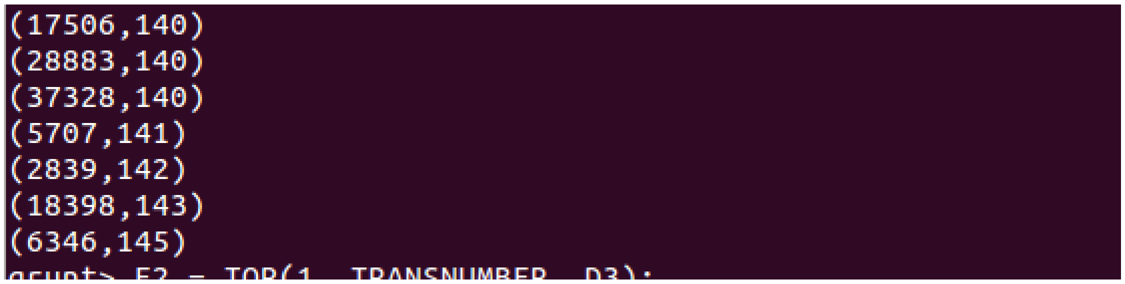
Sample output:

See the customer_id and transaction number, min 51



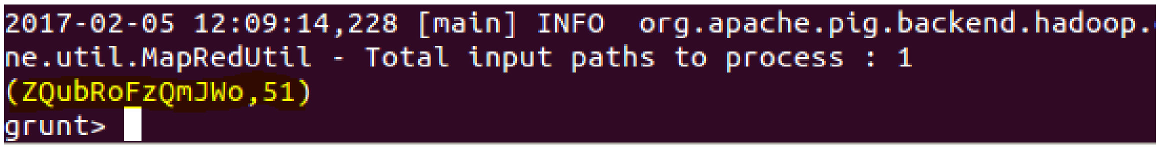
```
(17362,68)
(16523,68)
(21530,68)
(15239,67)
(14043,67)
(38564,67)
(45607,67)
(45095,66)
(44646,64)
(47126,63)
(47871,63)
(29002,59)
(24608,51)
2017-02-05 11:29:39,388 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 10
00: Error during parsing. Encountered " <PATH> "D3-ORDERED_DESC "" at line 2, co
lumn 6.
Was expecting one of:
    <EOF>
```

Maxium 145



```
(17506,140)
(28883,140)
(37328,140)
(5707,141)
(2839,142)
(18398,143)
(6346,145)
grunt> E2 = TOP(1, TRANSNUMBER, D3);
```

Retrieve the lowest customer name



```
2017-02-05 12:09:14,228 [main] INFO  org.apache.pig.backend.hadoop.
ne.util.MapRedUtil - Total input paths to process : 1
(ZQubRoFzQmJWo,51)
grunt> 
```

4.2

/*Write an Apache Pig query that join Customers and Transactions using Broadcast (replicated) join. The query reports for each customer the following info:

CustomerID, Name, Salary, NumOf Transactions, TotalSum, MinItem number of transaction total number of transaction,

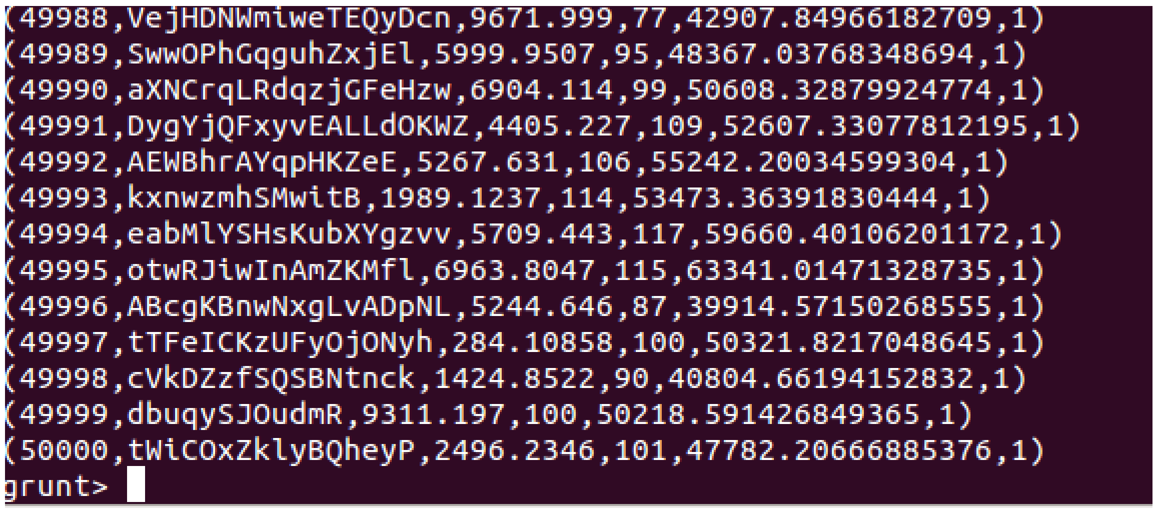
totalsum is sum of transtotal

minum number of item in transaction

*/

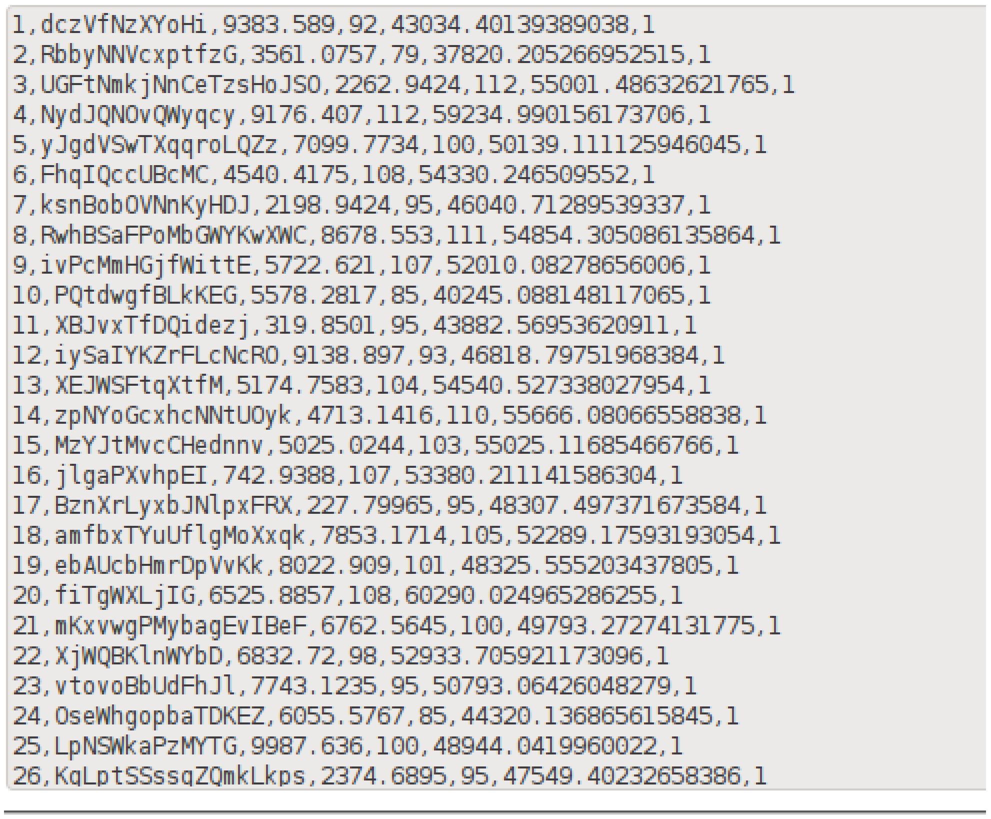
Code using 'replicated'

Sample Result :



```
(49988,VejHDNWmiweTEQyDcn,9671.999,77,42907.84966182709,1)
(49989,SwwOPhGqguhZxjEl,5999.9507,95,48367.03768348694,1)
(49990,aXNCrqLRdqzjGFeHzw,6904.114,99,50608.32879924774,1)
(49991,DygYjQFxyvEALLdOKWZ,4405.227,109,52607.33077812195,1)
(49992,AEWBhrAYqpHKZeE,5267.631,106,55242.20034599304,1)
(49993,kxnwzmhSMwitB,1989.1237,114,53473.36391830444,1)
(49994,eabMlYSHsKubXYgzvv,5709.443,117,59660.40106201172,1)
(49995,otwRJiwInAmZKMfl,6963.8047,115,63341.01471328735,1)
(49996,ABcgKBnwNxgLvADpNL,5244.646,87,39914.57150268555,1)
(49997,tTFeICKzUFyOjONyh,284.10858,100,50321.8217048645,1)
(49998,cVkDZzfSQSBNtnck,1424.8522,90,40804.66194152832,1)
(49999,dbuqySJOudmR,9311.197,100,50218.591426849365,1)
(50000,tWiCOxZklyBQheyP,2496.2346,101,47782.20666885376,1)
grunt>
```

Show CustomerID, Name, Salary, NumOf Transactions, TotalSum, MinItem



```
1,dczVfNzXYoHi,9383.589,92,43034.40139389038,1
2,RbbyNNVcxptfzG,3561.0757,79,37820.205266952515,1
3,UGFtNmkjNnCeTzsHoJSO,2262.9424,112,55001.48632621765,1
4,NydJQNOvQWyqcy,9176.407,112,59234.990156173706,1
5,yJgdVSwTXqqroLQZz,7099.7734,100,50139.111125946045,1
6,FhqIQccUBcMC,4540.4175,108,54330.246509552,1
7,ksnBobOVNnKyHDJ,2198.9424,95,46040.71289539337,1
8,RwhBSaFPoMbGWYKwXWC,8678.553,111,54854.305086135864,1
9,ivPcMmHGjfWittE,5722.621,107,52010.08278656006,1
10,PQtdwgfBLkKEG,5578.2817,85,40245.088148117065,1
11,XBJvxTfDQidezj,319.8501,95,43882.56953620911,1
12,iySaIYKZrFLcNcRO,9138.897,93,46818.79751968384,1
13,XEJWSFtqXtfM,5174.7583,104,54540.527338027954,1
14,zpNYoGcxhcNNtUOyk,4713.1416,110,55666.08066558838,1
15,MzYJtMvcCHednnv,5025.0244,103,55025.11685466766,1
16,jlgaPXvhpEI,742.9388,107,53380.211141586304,1
17,BznXrLyxbJNlpxFRX,227.79965,95,48307.497371673584,1
18,amfbxTYuUflgMoXxqk,7853.1714,105,52289.17593193054,1
19,ebAUcbHmrDpVvKk,8022.909,101,48325.555203437805,1
20,fiTgWXLjIG,6525.8857,108,60290.024965286255,1
21,mKxvwgPMybagEvIBeF,6762.5645,100,49793.27274131775,1
22,XjWQBKlnWYbD,6832.72,98,52933.705921173096,1
23,vtovoBbUdFhJl,7743.1235,95,50793.06426048279,1
24,OseWhgopbaTDKEZ,6055.5767,85,44320.136865615845,1
25,LpNSWkaPzMYTG,9987.636,100,48944.0419960022,1
26,KqLptSSssqZQmkLkps,2374.6895,95,47549.40232658386,1
```

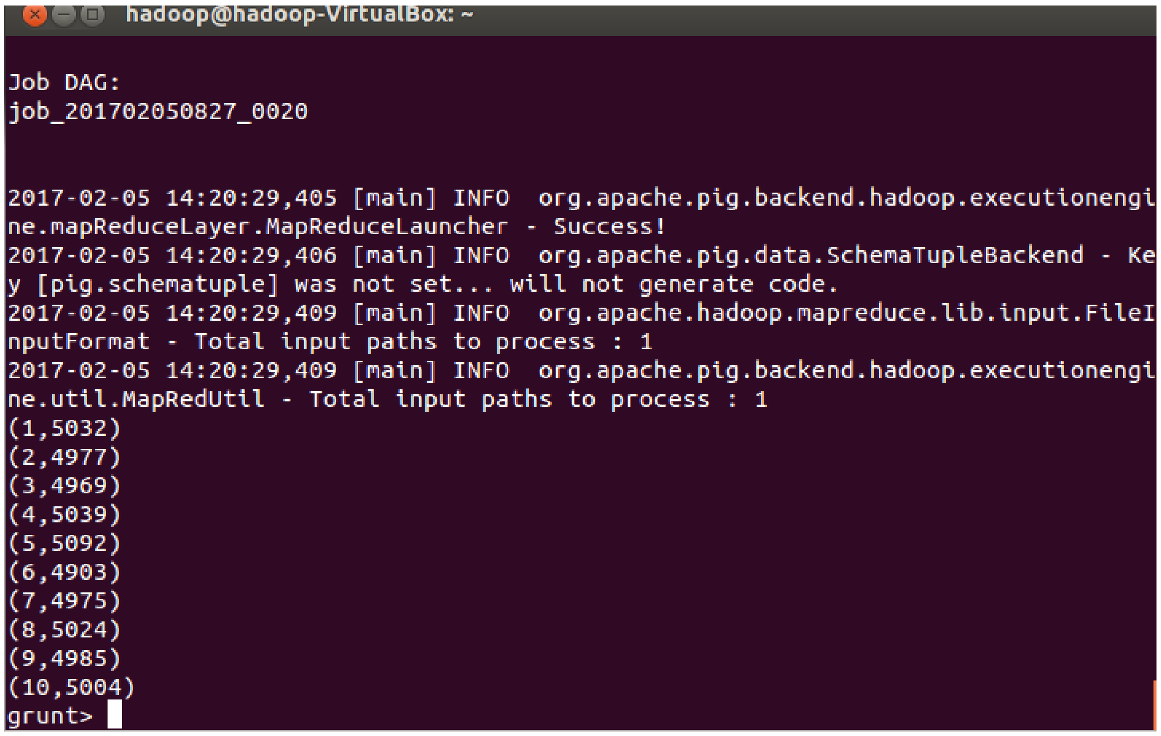
4.3

/*

Write an Apache Pig query that reports the Country Codes having number of customers greater than 5000 or less than 2000

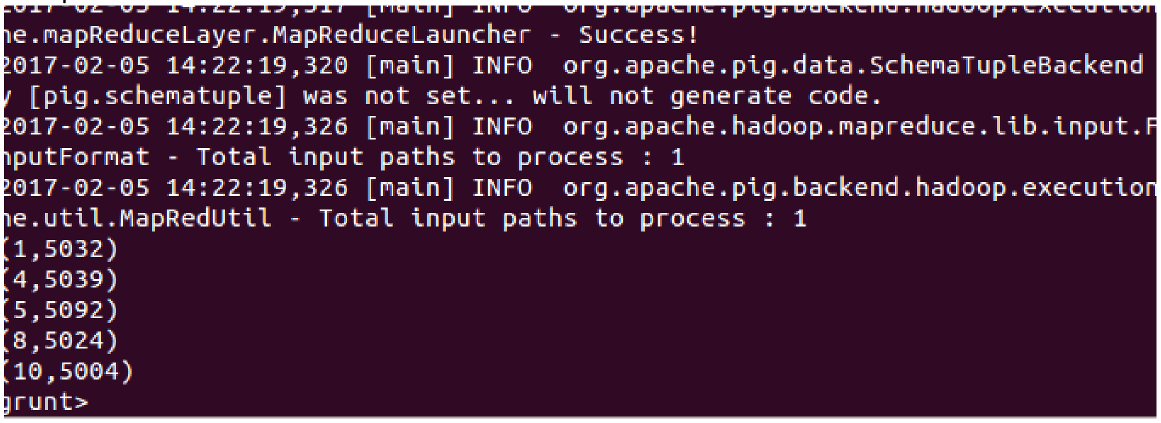
*/

Country code and customer number



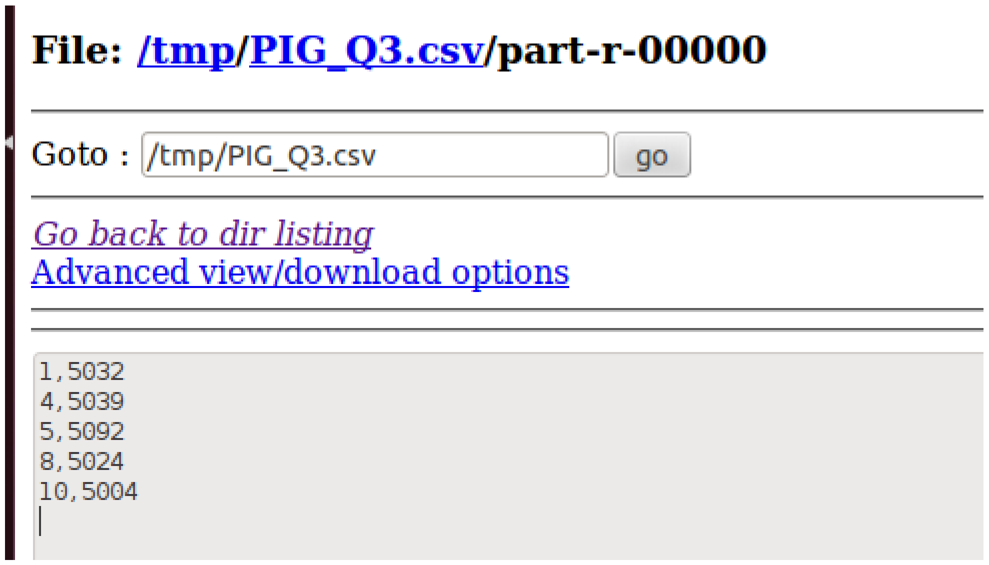
```
hadoop@hadoop-VirtualBox: ~

Job DAG:
job_201702050827_0020


2017-02-05 14:20:29,405 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Success!
2017-02-05 14:20:29,406 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Ke
y [pig.schematuple] was not set... will not generate code.
2017-02-05 14:20:29,409 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileI
nputFormat - Total input paths to process : 1
2017-02-05 14:20:29,409 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.util.MapRedUtil - Total input paths to process : 1
(1,5032)
(2,4977)
(3,4969)
(4,5039)
(5,5092)
(6,4903)
(7,4975)
(8,5024)
(9,4985)
(10,5004)
grunt>
```

Dump show



```
ne.mapReduceLayer.MapReduceLauncher - Success!
2017-02-05 14:22:19,320 [main] INFO  org.apache.pig.data.SchemaTupleBackend
y [pig.schematuple] was not set... will not generate code.
2017-02-05 14:22:19,326 [main] INFO  org.apache.hadoop.mapreduce.lib.input.F
nputFormat - Total input paths to process : 1
2017-02-05 14:22:19,326 [main] INFO  org.apache.pig.backend.hadoop.execution
ne.util.MapRedUtil - Total input paths to process : 1
(1,5032)
(4,5039)
(5,5092)
(8,5024)
(10,5004)
grunt>
```

Filter >5000 or < 2000



File: /tmp/PIG_Q3.csv/part-r-00000

Goto : /tmp/PIG_Q3.csv go

Go back to dir listing
Advanced view/download options

```
1,5032
4,5039
5,5092
8,5024
10,5004
```