Big Data Management - CS585 - Project 2

- **Azharuddin Priyotomo**
- **Congyuan Tang**

We are working under the virtual box (Ubuntu OS) and hadoop environment that the Professor provided us.

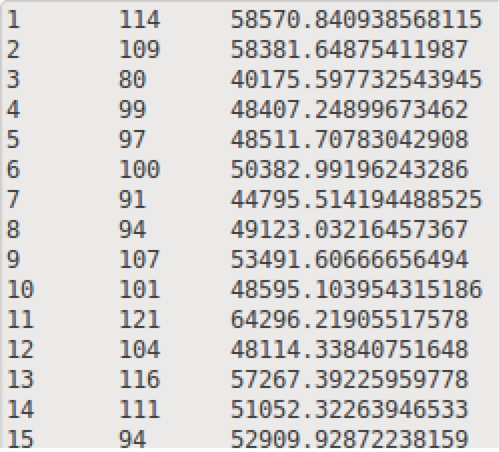
1. Query 1

To perform the query 1, we only need the data set from transaction file, since all output (Customer ID, Number of Transactions and Total Sum) are all available in transaction data.
First we need to load the transaction data into one variable and then group it by customer ID. From the grouped data, we select aggregate count of the transaction and the sum of transaction, and then store it to an output file in HDFS.

Query:

```
SET DEFAULT_PARALLEL 20;
trans = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/transactioninput'
USING PigStorage(',') as (transid:int,custid:int,
trxamt:float,noitem:int,desc:chararray);
A = group trans by custid;
B = foreach A generate group, COUNT(trans) as NumTransactions,
SUM(trans.trxamt) as TotalSum;
STORE B INTO
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/project2/query1output' USING
PigStorage();
```

Sample Output:



```
1       114     58570.840938568115
2       109     58381.64875411987
3       80      40175.597732543945
4       99      48407.24899673462
5       97      48511.70783042908
6       100     50382.99196243286
7       91      44795.514194488525
8       94      49123.03216457367
9       107     53491.60666656494
10      101     48595.103954315186
11      121     64296.21905517578
12      104     48114.33840751648
13      116     57267.39225959778
14      111     51052.32263946533
15      94      52909.92872238159
```

Description: (Customer ID, Number of Transactions, Total Sum)

2. Query 2

To execute the file for the second query:

1. Start Hadoop and Pig: > start-all.sh
2. Clear output folder for query's result:
   > hadoop fs -rmr /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-temp
   > hadoop fs -rmr /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final
3. Be sure to change the path of files (customer and transaction dataset) in the query2.pig file.
4. Run script in batch mode: > pig PATH/query2.pig
5. The final result will be in folder /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final

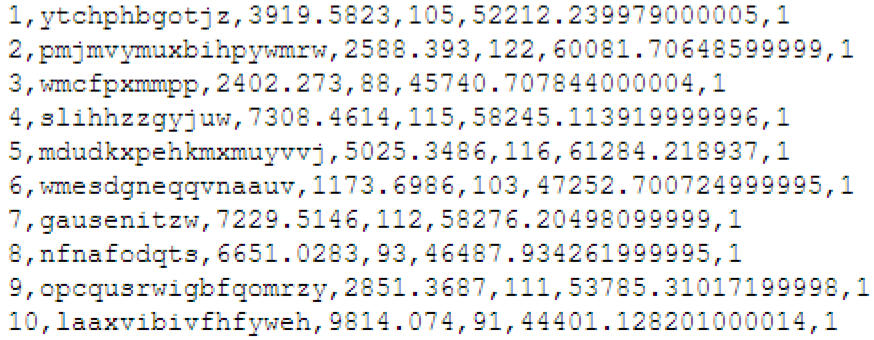
The script file for this query is below:

```
%declare dataSource1
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Customer'
%declare dataSource2
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Transaction'
%declare tempOutputFile
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-temp'
%declare finalOutputFile
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final'

raw1 = LOAD '$dataSource1' USING PigStorage(',') AS (CID1, CName,
Age, CountryCode, Salary);

raw2 = LOAD '$dataSource2' USING PigStorage(',') AS (TID, CID2,
TransTotal, TransNumItem:INT, TransDesc);

sub_raw1 = FOREACH raw1 GENERATE CID1, CName, Salary;

joinedTable = JOIN sub_raw1 BY CID1, raw2 BY CID2;

groupedTable = GROUP joinedTable BY CID1;

temp = FOREACH groupedTable GENERATE group, COUNT(joinedTable.CID1)
AS NumOfTrans, SUM(joinedTable.TransTotal) AS TotalSum,
MIN(joinedTable.TransNumItem) AS MinItems;

outputTable = JOIN temp BY group, sub_raw1 BY CID1;

STORE outputTable INTO '$tempOutputFile' USING PigStorage(',');

raw3 = LOAD '$tempOutputFile' USING PigStorage(',') AS (CID,
NumOfTrans, TotalSum, MinItems, CID1, CName, Salary);

outputTable2 = FOREACH raw3 GENERATE CID, CName, Salary, NumOfTrans,
TotalSum, MinItems;
```

```
STORE outputTable2 INTO '$finalOutputFile' USING PigStorage(',');
```

The query's result is attached as file 'query2result.txt'.
The system output (statistics part) is attached as file 'query2SystemOutput.txt'.

Sample output:



```
1,ytchphbgotjz,3919.5823,105,52212.239979000005,1
2,pmjmvymuxbihpywmrw,2588.393,122,60081.70648599999,1
3,wmcfpxmmpp,2402.273,88,45740.707844000004,1
4,slihhzzgyjuw,7308.4614,115,58245.113919999996,1
5,mdudkxpehkmxmuyvvj,5025.3486,116,61284.218937,1
6,wmesdgneqqvnaauv,1173.6986,103,47252.700724999995,1
7,gausenitzw,7229.5146,112,58276.20498099999,1
8,nfnafodqts,6651.0283,93,46487.934261999995,1
9,opcqusrwigbfqomrzy,2851.3687,111,53785.31017199998,1
10,laaxvibivfhfyweh,9814.074,91,44401.128201000014,1
```

Description: (Customer ID, CustomerName, Salary, Number of Transactions, Total Sum, MinItems)

3. Query 3

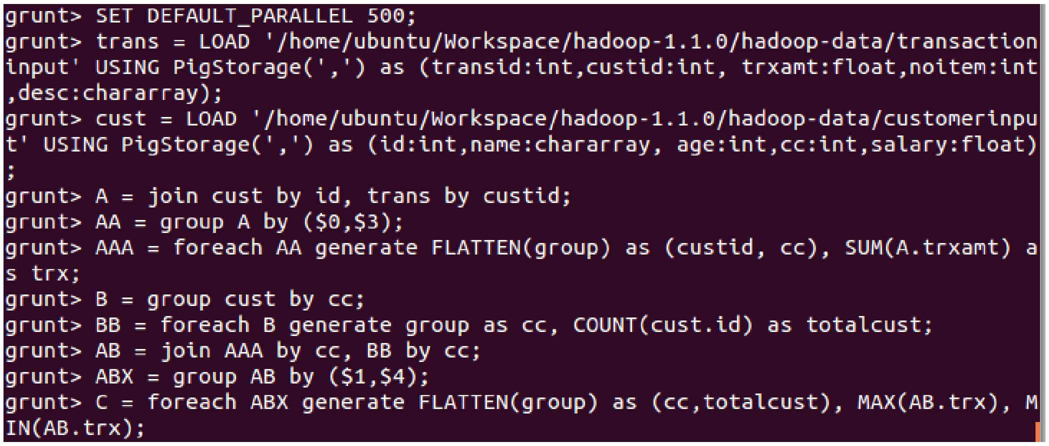
To do query 3, here is the process:
1. We should use two data set (customer and transaction), so we should first join those two data into one variable based on Customer ID.
2. After we have joined data, we group them based on Customer ID and Country Code and aggregate transaction amount using SUM, resulting output of [Customer ID, Country Code, Total Transaction].
3. We perform another separated query of grouping customer data by Country Code and outputting them in format of [Country Code, Total Customer].
4. We join query result from no 3 and no 2 by Country Code. We then group them by Country Code and Total Customer, so we can get the aggregate function of MAX and MIN of the total transaction amount and resulting the final output of [Country Code, Total Customer, Maximum Total Transaction, Minimum Total Transaction].
5. Store the result into file in HDFS.

Query:

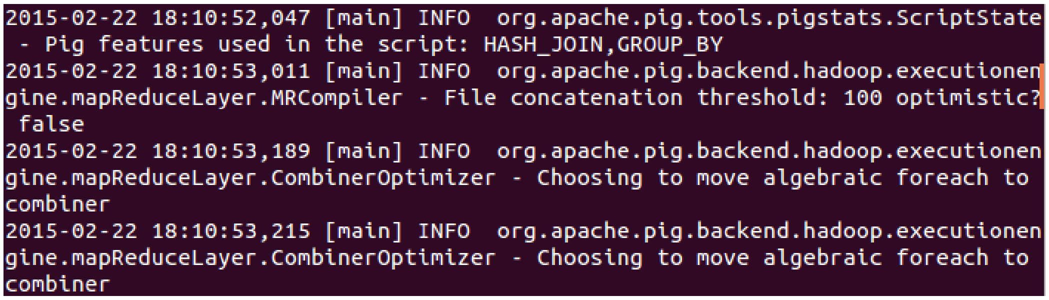
```
SET DEFAULT_PARALLEL 500;
trans = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/transactioninput'
USING PigStorage(',') as (transid:int,custid:int,
trxamt:float,noitem:int,desc:chararray);
cust = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/customerinput'
USING PigStorage(',') as (id:int,name:chararray, age:int,cc:int,salary:float);
A = join cust by id, trans by custid;
AA = group A by ($0,$3);
AAA = foreach AA generate FLATTEN(group) as (custid, cc), SUM(A.trxamt) as trx;
B = group cust by cc;
```

```
BB = foreach B generate group as cc, COUNT(cust.id) as totalcust;
AB = join AAA by cc, BB by cc;
ABX = group AB by ($1,$4);
C = foreach ABX generate FLATTEN(group) as (cc,totalcust), MIN(AB.trx) ,
MAX(AB.trx);
STORE C INTO
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/project2/query3outputs' USING
PigStorage();
```

Screen captures:

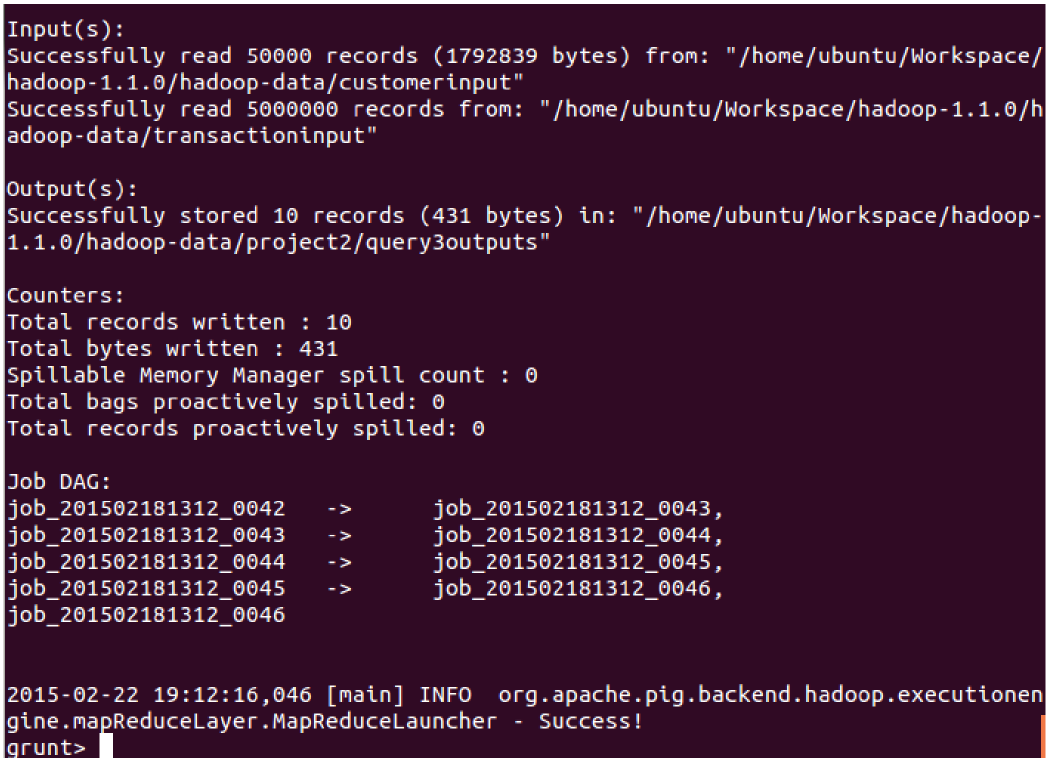


```
grunt> SET DEFAULT_PARALLEL 500;
grunt> trans = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/transaction
input' USING PigStorage(',') as (transid:int,custid:int, trxamt:float,noitem:int
,desc:chararray);
grunt> cust = LOAD '/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/customerinpu
t' USING PigStorage(',') as (id:int,name:chararray, age:int,cc:int,salary:float)
;
grunt> A = join cust by id, trans by custid;
grunt> AA = group A by ($0,$3);
grunt> AAA = foreach AA generate FLATTEN(group) as (custid, cc), SUM(A.trxamt) a
s trx;
grunt> B = group cust by cc;
grunt> BB = foreach B generate group as cc, COUNT(cust.id) as totalcust;
grunt> AB = join AAA by cc, BB by cc;
grunt> ABX = group AB by ($1,$4);
grunt> C = foreach ABX generate FLATTEN(group) as (cc,totalcust), MAX(AB.trx), M
IN(AB.trx);
```



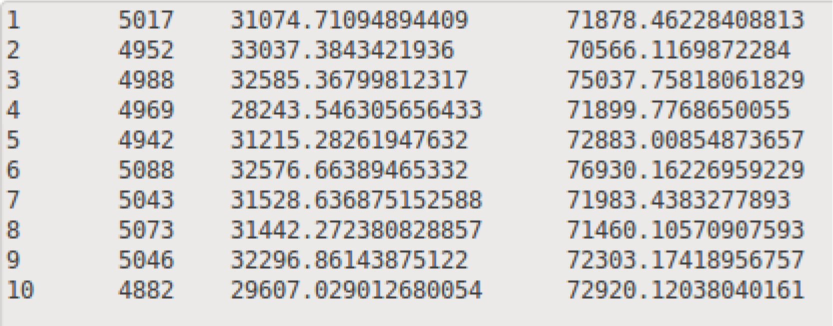
```
2015-02-22 18:10:52,047 [main] INFO  org.apache.pig.tools.pigstats.ScriptState
 - Pig features used in the script: HASH_JOIN,GROUP_BY
2015-02-22 18:10:53,011 [main] INFO  org.apache.pig.backend.hadoop.executionen
gine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic?
 false
2015-02-22 18:10:53,189 [main] INFO  org.apache.pig.backend.hadoop.executionen
gine.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic foreach to
combiner
2015-02-22 18:10:53,215 [main] INFO  org.apache.pig.backend.hadoop.executionen
gine.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic foreach to
combiner
```

....



```
Input(s):
Successfully read 50000 records (1792839 bytes) from: "/home/ubuntu/Workspace/
hadoop-1.1.0/hadoop-data/customerinput"
Successfully read 5000000 records from: "/home/ubuntu/Workspace/hadoop-1.1.0/h
adoop-data/transactioninput"

Output(s):
Successfully stored 10 records (431 bytes) in: "/home/ubuntu/Workspace/hadoop-
1.1.0/hadoop-data/project2/query3outputs"

Counters:
Total records written : 10
Total bytes written : 431
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_201502181312_0042   ->      job_201502181312_0043,
job_201502181312_0043   ->      job_201502181312_0044,
job_201502181312_0044   ->      job_201502181312_0045,
job_201502181312_0045   ->      job_201502181312_0046,
job_201502181312_0046


2015-02-22 19:12:16,046 [main] INFO  org.apache.pig.backend.hadoop.executionen
gine.mapReduceLayer.MapReduceLauncher - Success!
grunt>
```

Output:



```
1       5017    31074.71094894409       71878.46228408813
2       4952    33037.3843421936        70566.1169872284
3       4988    32585.36799812317       75037.75818061829
4       4969    28243.546305656433      71899.7768650055
5       4942    31215.28261947632       72883.00854873657
6       5088    32576.66389465332       76930.16226959229
7       5043    31528.636875152588      71983.4383277893
8       5073    31442.272380828857      71460.10570907593
9       5046    32296.86143875122       72303.17418956757
10      4882    29607.029012680054      72920.12038040161
```

Description: Country Code, Number of Customers, Minimum Total Transaction, Maximum Total Transaction

4. Query 4

The process of execution of this query is the same as query2.
The script for this query is below:

```
%declare dataSource1
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Customer'
%declare dataSource2
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Transaction'
```

```
%declare tempOutputFile
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-temp'
%declare finalOutputFile
'/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-final'

raw1 = LOAD '$dataSource1' USING PigStorage(',') AS (CID1, CName,
Age, CountryCode, Salary);

raw2 = LOAD '$dataSource2' USING PigStorage(',') AS (TID, CID2,
TransTotal, TransNumItem:INT, TransDesc);

sub_raw1 = FOREACH raw1 GENERATE CID1, CName, Salary;

joinedTable = JOIN raw2 BY CID2, sub_raw1 BY CID1 USING 'replicated';

groupedTable = GROUP joinedTable BY CID1;

temp = FOREACH groupedTable GENERATE group, COUNT(joinedTable.CID1)
AS NumOfTrans, SUM(joinedTable.TransTotal) AS TotalSum,
MIN(joinedTable.TransNumItem) AS MinItems;

outputTable = JOIN temp BY group, sub_raw1 BY CID1 USING
'replicated';

STORE outputTable INTO '$tempOutputFile' USING PigStorage(',');

raw3 = LOAD '$tempOutputFile' USING PigStorage(',') AS (CID,
NumOfTrans, TotalSum, MinItems, CID1, CName, Salary);

outputTable2 = FOREACH raw3 GENERATE CID, CName, Salary, NumOfTrans,
TotalSum, MinItems;

STORE outputTable2 INTO '$finalOutputFile' USING PigStorage(',');
```

The query's result is attached as file 'query4result.txt'.
The system output (statistics part) is attached as file 'query4SystemOutput.txt'.

The statistics from query2 is:

```
HadoopVersion   PigVersion UserId     StartedAt  FinishedAt Features
1.1.0 0.10.0    ubuntu     2015-02-22 04:55:24   2015-02-22 04:58:35
HASH_JOIN,GROUP_BY

Success!

Job Stats (time in seconds):
```

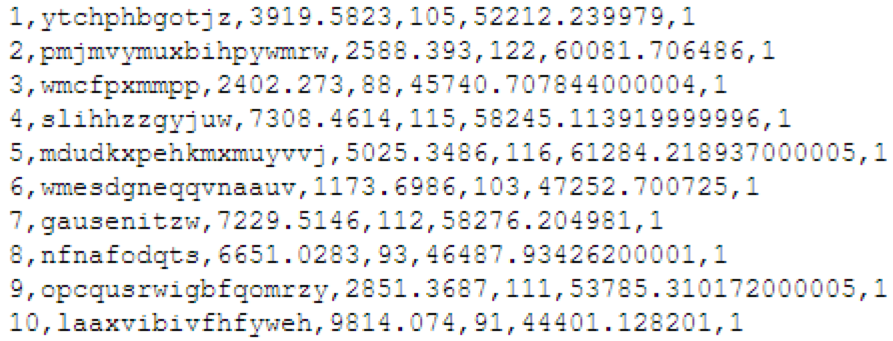
```
JobId	Maps	Reduces	MaxMapTime	MinMapTIme	AvgMapTime	MaxReduceTime
MinReduceTime	AvgReduceTime	Alias	Feature	Outputs
job_201502220216_0030	1	0	2	2	2	0	0
0raw1,sub_raw1	MAP_ONLY
job_201502220216_0031	6	1	10	0	7	57	57	57
joinedTable,raw2	HASH_JOIN
job_201502220216_0032	8	1	7	3	5	41	41	41
groupedTable,temp	GROUP_BY,COMBINER
job_201502220216_0033	2	1	1	1	1	10	10	10
outputTable	HASH_JOIN
/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-temp,
job_201502220216_0034	1	0	2	2	2	0	0
0outputTable2,raw3	MAP_ONLY
/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query2-final,
```

The statistics from query4 is:

```
HadoopVersion	PigVersion	UserId	StartedAt	FinishedAt	Features
1.1.0	0.10.0	ubuntu	2015-02-22 05:33:57	2015-02-22 05:36:29
REPLICATED_JOIN,GROUP_BY

Success!

Job Stats (time in seconds):
JobId	Maps	Reduces	MaxMapTime	MinMapTIme	AvgMapTime	MaxReduceTime
MinReduceTime	AvgReduceTime	Alias	Feature	Outputs
job_201502220216_0053	1	0	2	2	2	0	0
0raw1,sub_raw1	MULTI_QUERY,MAP_ONLY
job_201502220216_0054	5	1	26	12	20	86	86	86
groupedTable,joinedTable,outputTable,raw2,temp
REPLICATED_JOIN,GROUP_BY,COMBINER
/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-temp,
job_201502220216_0055	1	0	2	2	2	0	0
0outputTable2,raw3	MAP_ONLY
/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Query4-final,
```

Compared the system output of this query and query2, we can clearly see query2 consists of 5 Map and/or MapReduce jobs, whereas query4 requires 3 Map and/or MapReduce jobs. Also, the time cost for query2 & 4 is 3mins 11secs and 2mins 42secs respectively. Using replicated JOIN increase the efficiency by nearly 15%.

Sample output:



```
1,ytchphbgotjz,3919.5823,105,52212.239979,1
2,pmjmvymuxbihpywmrw,2588.393,122,60081.706486,1
3,wmcfpxmmpp,2402.273,88,45740.707844000004,1
4,slihhzzgyjuw,7308.4614,115,58245.113919999996,1
5,mdudkxpehkmxmuyvvj,5025.3486,116,61284.218937000005,1
6,wmesdgneqqvnaauv,1173.6986,103,47252.700725,1
7,gausenitzw,7229.5146,112,58276.204981,1
8,nfnafodqts,6651.0283,93,46487.93426200001,1
9,opcqusrwigbfqomrzy,2851.3687,111,53785.310172000005,1
10,laaxvibivfhfyweh,9814.074,91,44401.128201,1
```

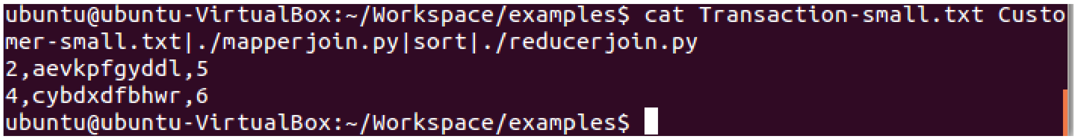
Description: (Customer ID, CustomerName, Salary, Number of Transactions, Total Sum, MinItems)

5. Query 5

We choose Python language combined with the Hadoop Streaming. To do the query, we need one mapper (mapperjoin.py) and one reducer (reducerjoin.py), where the joining data set will happen in reducer side. The mapper will read the input from both of transaction and customer file. Initial value of (-1) is set for each of the attribute that we need (customer _id, customer_name, country_code, transaction) and will be replaced with real value after reading the input file. So for example, when mapper read data from customer file, the value of transaction_id will still be -1 while other attributes will have the real values. The output from the mapper will follow the format of [customer_id, customer_name, country_code, transaction_id]. The output will be sorted and then sent to reducer. Reducer will group the output from the mapper based on **customer ID**, filter the result using condition (if country code = 5) and then sent to stdout.

We first test the codes using the local Python environment first using a small dataset:

```
cat Transaction-small.txt Customer-small.txt|./mapperjoin.py|sort|./reducerjoin.py
```

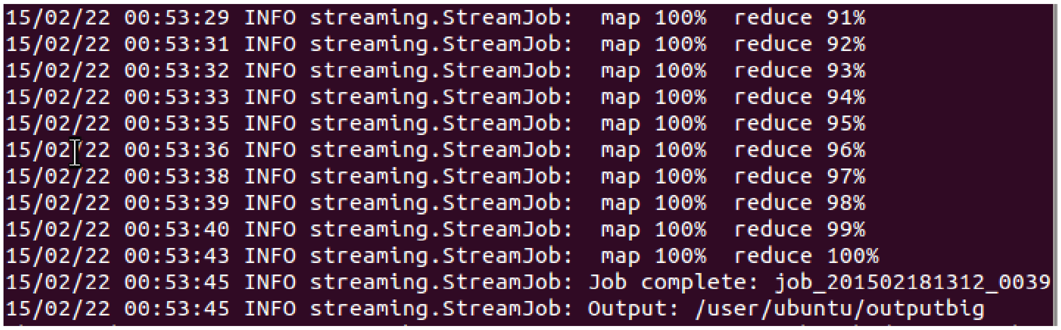


```
ubuntu@ubuntu-VirtualBox:~/Workspace/examples$ cat Transaction-small.txt Custo
mer-small.txt|./mapperjoin.py|sort|./reducerjoin.py
2,aevkpfgyddl,5
4,cybdxdfbhwr,6
ubuntu@ubuntu-VirtualBox:~/Workspace/examples$
```

After we make sure it works, then we run the mapper and reducer script using Hadoop with the real data set using following command:

```
$HADOOP_HOME/bin/hadoop jar /home/ubuntu/Workspace/hadoop-
1.1.0/contrib/streaming/hadoop-streaming-1.1.0.jar -D mapred.reduce.tasks=1 -file
/home/ubuntu/Workspace/examples/python/mapperjoin.py -mapper
/home/ubuntu/Workspace/examples/python/mapperjoin.py -file
/home/ubuntu/Workspace/examples/python/reducerjoin.py -reducer
/home/ubuntu/Workspace/examples/python/reducerjoin.py -input /user/ubuntu/inputbig/*
-output /user/ubuntu/outputbig
```

Screen capture:

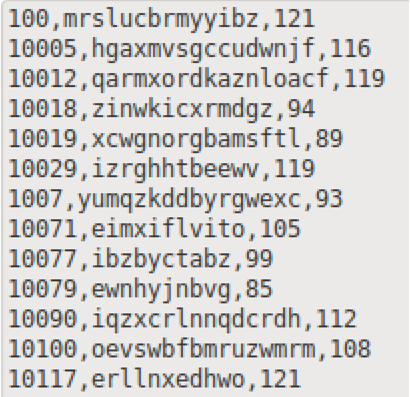


```
15/02/22 00:53:29 INFO streaming.StreamJob:  map 100%  reduce 91%
15/02/22 00:53:31 INFO streaming.StreamJob:  map 100%  reduce 92%
15/02/22 00:53:32 INFO streaming.StreamJob:  map 100%  reduce 93%
15/02/22 00:53:33 INFO streaming.StreamJob:  map 100%  reduce 94%
15/02/22 00:53:35 INFO streaming.StreamJob:  map 100%  reduce 95%
15/02/22 00:53:36 INFO streaming.StreamJob:  map 100%  reduce 96%
15/02/22 00:53:38 INFO streaming.StreamJob:  map 100%  reduce 97%
15/02/22 00:53:39 INFO streaming.StreamJob:  map 100%  reduce 98%
15/02/22 00:53:40 INFO streaming.StreamJob:  map 100%  reduce 99%
15/02/22 00:53:43 INFO streaming.StreamJob:  map 100%  reduce 100%
15/02/22 00:53:45 INFO streaming.StreamJob: Job complete: job_201502181312_0039
15/02/22 00:53:45 INFO streaming.StreamJob: Output: /user/ubuntu/outputbig
```

Notice that we set the number of reducer to be equal 1 in the above command (-D mapred.reduce.tasks=1). This is because by default, hadoop streaming will use 3 reducers which will divide the result and group the count result separately in each file. So we need to specify the number of reducer to be in a single reducer so that the result will be grouped in one single output.

The query resulted **4942 records** (customer which has country code = 5).

Sample output:



```
100,mrslucbrmyyibz,121
10005,hgaxmvsgccudwnjf,116
10012,qarmxordkaznloacf,119
10018,zinwkicxrmdgz,94
10019,xcwgnorgbamsftl,89
10029,izrghhtbeewv,119
1007,yumqzkddbyrgwexc,93
10071,eimxiflvito,105
10077,ibzbyctabz,99
10079,ewnhyjnbvg,85
10090,iqzxcrlnnqdcrdh,112
10100,oevswbfbmruzwmrm,108
10117,erllnxedhwo,121
```

Description: Customer ID, Customer Name, Count Transactions

6. Query 6

Still, we choose Python to do this query. The mapper function is attached as file 'PMapper.py' and the reducer function is 'PReducer.py'.

To execute the file for the sixth query:

1. Start Hadoop and Pig
2. Clear output folder for query's result: > hadoop fs -rmr /home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/query6
3. Go to hadoop local directory.
4. Make sure our Mapper and Reducer function is executable.
5. Specify the number of reducer to be 1 and run: > bin/hadoop jar contrib/streaming/hadoop-streaming-1.1.0.jar -D mapred.reduce.tasks=1 -file /home/ubuntu/Desktop/PMapper.py -mapper /home/ubuntu/Desktop/PMapper.py -file /home/ubuntu/Desktop/PReducer.py -reducer /home/ubuntu/Desktop/PReducer.py -input

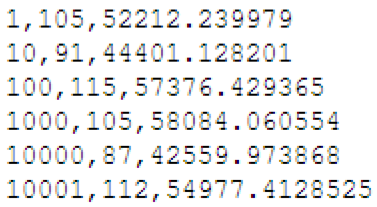
/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/Transaction -output
/home/ubuntu/Workspace/hadoop-1.1.0/hadoop-data/query6

The reason of specifying the reducer number to be 1 is that, the reducers input under the Hadoop streaming mode is just a stream. These means all the Key-Value pairs will fit into all reducers, instead of each Key-Value pair goes to different reducer. So, we limit the number of reducer in order to fit all our result in one file, which is attached as ‘query6result.txt’.

Compared to query1, query6 is more Hadoop visible, more close to MapReduce algorithm, and query1 is more like a SQL language. Also, as described above the input of reducer in query6 is totally different from typical JAVA MapReduce framework.

In a word, the Hadoop streaming mode makes our MapReduce algorithm even difficult to perform the SQL queries. Running in streaming mode, it takes 48 seconds to finish this query, whereas in query1, it takes 58 seconds. So, streaming in Python is no more convenient than JAVA but is more efficient than Pig.

Sample output:



```
1,105,52212.239979
10,91,44401.128201
100,115,57376.429365
1000,105,58084.060554
10000,87,42559.973868
10001,112,54977.4128525
```

Description: CustomerID, NumTransactions, TotalSum