

# CSC 555: Mining Big Data

Project, Phase 2 (due Sunday June 16<sup>th</sup>)

Lavinia Wang 1473704

In this part of the project, you will execute queries using Hive, Pig and Hadoop streaming and develop a custom version of KMeans clustering. The schema is available below, but don't forget to change to the correct delimiter:

[http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM\\_schema\\_hive.sql](http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql)

The data is available at (this is Scale1, the smallest denomination of this benchmark)

<http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/>

In your submission, please note what cluster you are using. Please be sure to submit all code (pig, python and Hive). You should also submit the command lines you use and a screenshot of a completed run (just the last page, do not worry about capturing the whole output). An answer without corresponding code will not be counted.

I highly recommend creating a small sample input (e.g., by running head lineorder.tbl > lineorder.tbl.sample, you can create a small version of lineorder with a few lines) and testing your code with it. You can run **head -n 500 lineorder.tbl** to get a specific number of lines.

NOTE: the total number of points adds up to 70.

## Part 1: Data Transformation (15 pts)

Transform part.tbl table into a ~-separated ('~') file: Use Hive, MapReduce with HadoopStreaming and Pig (i.e. 3 different solutions).

In all solutions you must switch the first two columns (i.e., switch the positions of columns 1 and 2). You do not need to transform the columns in any way, just switch them around. Note that this means you do not have to use SELECT TRANSFORM or python in your Hive solution.

Using my multi-node cluster

### Hive

```
CREATE TABLE part (
    p_partkey    INT,
    p_name      VARCHAR(22),
    p_mfgr      VARCHAR(6),
    p_category   VARCHAR(7),
    p_brand1     VARCHAR(9),
    p_color      VARCHAR(11),
    p_type       VARCHAR(25),
    p_size       INT,
    p_container  VARCHAR(10)
)ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;
```

```
LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl' OVERWRITE INTO TABLE part;
```

**Python code (colSwitcher.py):**

```
#!/usr/bin/python
import sys

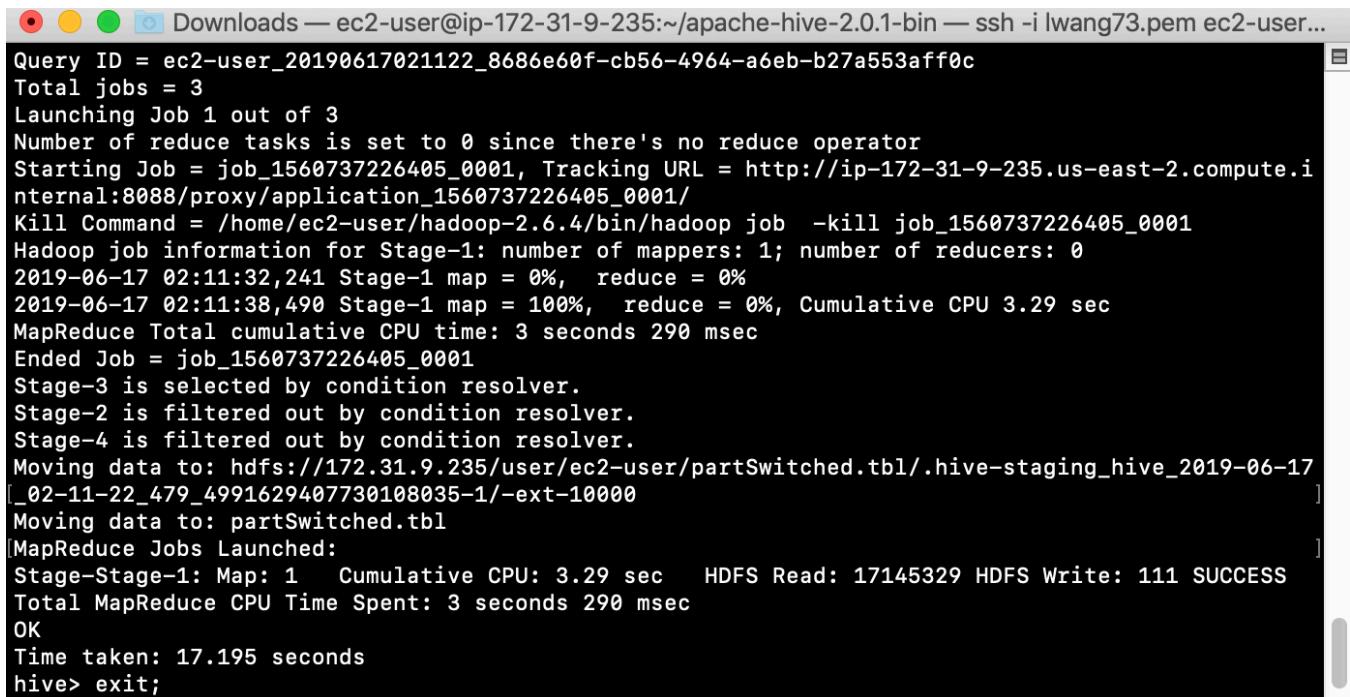
for line in sys.stdin:
    line = line.strip().split('\t')
    print '~'.join([line[1], line[0], line[2], line[3], line[4], line[5], line[6], line[7], line[8]])
```

**Commands:**

```
ADD FILE /home/ec2-user/colSwitcher.py;
```

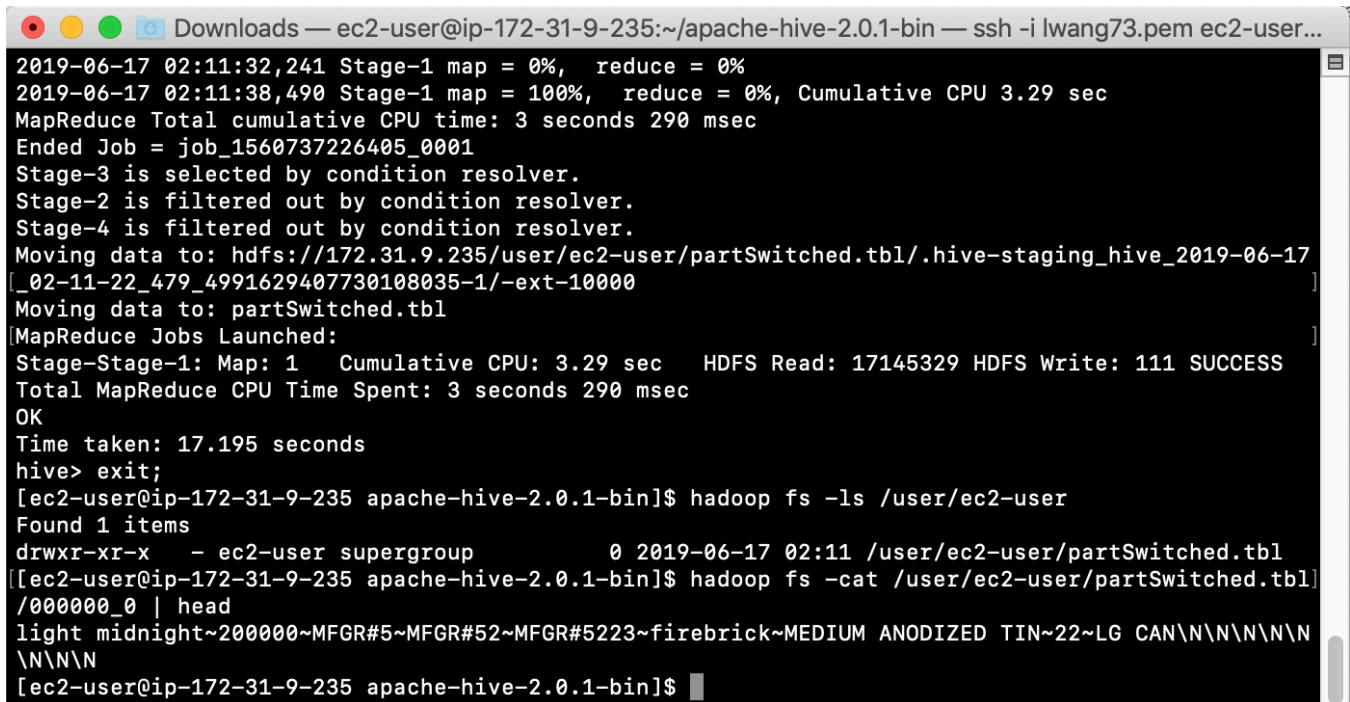
```
INSERT OVERWRITE DIRECTORY 'partSwitched.tbl' SELECT TRANSFORM (p_partkey, p_name,
p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container) USING 'colSwitcher.py' AS
(p_name, p_partkey, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container)
FROM part;
```

**Completed Run:**



```
Downloads — ec2-user@ip-172-31-9-235:~/apache-hive-2.0.1-bin — ssh -i lwang73.pem ec2-user...
Query ID = ec2-user_20190617021122_8686e60f-cb56-4964-a6eb-b27a553aff0c
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1560737226405_0001, Tracking URL = http://ip-172-31-9-235.us-east-2.compute.internal:8088/proxy/application_1560737226405_0001/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job -kill job_1560737226405_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2019-06-17 02:11:32,241 Stage-1 map = 0%, reduce = 0%
2019-06-17 02:11:38,490 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.29 sec
MapReduce Total cumulative CPU time: 3 seconds 290 msec
Ended Job = job_1560737226405_0001
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://172.31.9.235/user/ec2-user/partSwitched.tbl/.hive-staging_hive_2019-06-17
[_02-11-22_479_4991629407730108035-1/-ext-10000]
Moving data to: partSwitched.tbl
[MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 3.29 sec   HDFS Read: 17145329 HDFS Write: 111 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 290 msec
OK
Time taken: 17.195 seconds
hive> exit;
```

**Output:**



Downloads — ec2-user@ip-172-31-9-235:~/apache-hive-2.0.1-bin — ssh -i lwang73.pem ec2-user...  
2019-06-17 02:11:32,241 Stage-1 map = 0%, reduce = 0%  
2019-06-17 02:11:38,490 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.29 sec  
MapReduce Total cumulative CPU time: 3 seconds 290 msec  
Ended Job = job\_1560737226405\_0001  
Stage-3 is selected by condition resolver.  
Stage-2 is filtered out by condition resolver.  
Stage-4 is filtered out by condition resolver.  
Moving data to: hdfs://172.31.9.235/user/ec2-user/partSwitched.tbl/.hive-staging\_hive\_2019-06-17  
[ \_02-11-22\_479\_4991629407730108035-1/-ext-10000  
Moving data to: partSwitched.tbl  
[MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Cumulative CPU: 3.29 sec HDFS Read: 17145329 HDFS Write: 111 SUCCESS  
Total MapReduce CPU Time Spent: 3 seconds 290 msec  
OK  
Time taken: 17.195 seconds  
hive> exit;  
[ec2-user@ip-172-31-9-235 apache-hive-2.0.1-bin]\$ hadoop fs -ls /user/ec2-user  
Found 1 items  
drwxr-xr-x - ec2-user supergroup 0 2019-06-17 02:11 /user/ec2-user/partSwitched.tbl  
[[ec2-user@ip-172-31-9-235 apache-hive-2.0.1-bin]\$ hadoop fs -cat /user/ec2-user/partSwitched.tbl]  
/000000\_0 | head  
light midnight~200000~MFGR#5~MFGR#52~MFGR#5223~firebrick~MEDIUM ANODIZED TIN~22~LG CAN\\N\\N\\N\\N\\N\\N  
\\N\\N\\N  
[ec2-user@ip-172-31-9-235 apache-hive-2.0.1-bin]\$

## Hadoop Streaming

There is no need for a custom mapper for this exercise, so I used the linux cat function as the mapper.  
The reducer code is:

### colSwitcherReducer.py

```
#!/usr/bin/python
import sys

for line in sys.stdin:
    line = line.strip().split(' ')
    print "%s~%s~%s~%s~%s~%s~%s~%s~%s" %
    (line[1],line[0],line[2],line[3],line[4],line[5],line[6],line[7],line[8])
```

### Command:

```
hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/ssbm/part.tbl -output /data/output110 -
mapper /bin/cat -reducer colSwitcherReducer.py -file colSwitcherReducer.py
```

```
Downloads — ec2-user@ip-172-31-9-235:~/hadoop-2.6.4 — ssh -i lwang73.pem ec2-user@ec2-18...
Reduce input records=200000
Reduce output records=200000
Spilled Records=400000
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=270
CPU time spent (ms)=4980
Physical memory (bytes) snapshot=698826752
Virtual memory (bytes) snapshot=6442582016
Total committed heap usage (bytes)=492306432
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=17143355
File Output Format Counters
Bytes Written=17139259
19/06/17 02:44:05 INFO streaming.StreamJob: Output directory: /data/output110
[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$
```

**Output:**

```
Downloads — ec2-user@ip-172-31-9-235:~/hadoop-2.6.4 — ssh -i lwang73.pem ec2-user@ec2-18...
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=17143355
File Output Format Counters
Bytes Written=17139259
19/06/17 02:44:05 INFO streaming.StreamJob: Output directory: /data/output110
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ hadoop fs -cat /data/output110/part-00000 | head
cyan floral~100000~MFGR#5~MFGR#55~MFGR#5535~maroon~LARGE BURNISHED STEEL~17~MED BOX
floral pink~100001~MFGR#5~MFGR#54~MFGR#5436~black~STANDARD BRUSHED TIN~37~JUMBO CASE
olive rose~100002~MFGR#1~MFGR#12~MFGR#1226~peach~STANDARD ANODIZED NICKEL~11~WRAP CAN
light violet~100003~MFGR#1~MFGR#15~MFGR#155~puff~MEDIUM PLATED BRASS~41~SM BOX
gainsboro slate~100004~MFGR#2~MFGR#25~MFGR#2511~hot~SMALL POLISHED TIN~29~SM CASE
drab misty~100005~MFGR#2~MFGR#23~MFGR#235~grey~SMALL POLISHED STEEL~7~MED BAG
honeydew navy~100006~MFGR#1~MFGR#12~MFGR#1237~royal~STANDARD BURNISHED COPPER~23~WRAP CASE
moccasin wheat~100007~MFGR#2~MFGR#22~MFGR#2211~firebrick~PROMO BURNISHED COPPER~4~MED PKG
powder burlywood~100008~MFGR#3~MFGR#35~MFGR#3535~spring~ECONOMY BRUSHED BRASS~19~SM PKG
antique aquamarine~100009~MFGR#5~MFGR#52~MFGR#529~indian~SMALL BURNISHED STEEL~41~WRAP BOX
cat: Unable to write to output stream.
[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$
```

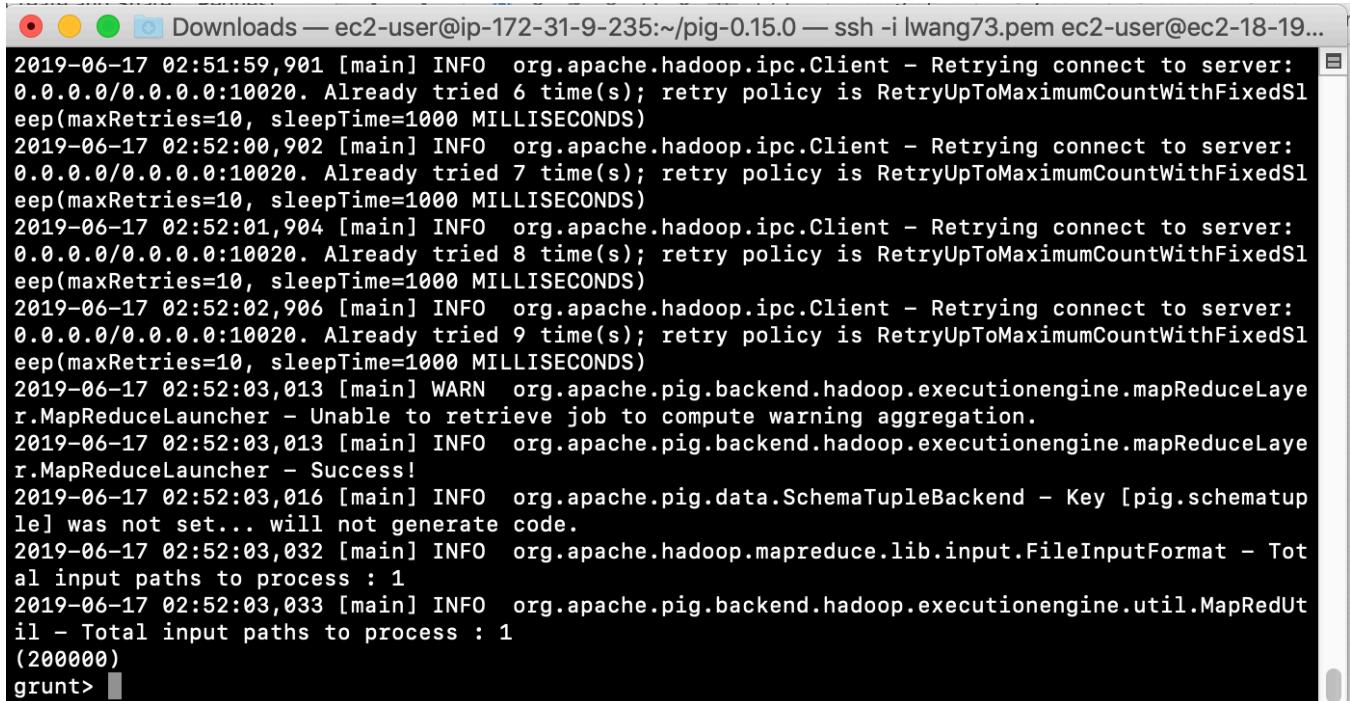
**Pig**

**Load the Data:**

```
PartData = LOAD '/user/ec2-user/ssbm/part.tbl' USING PigStorage('') AS (p_partkey:int, p_name:chararray, p_mfgr:chararray, p_category:chararray, p_brand1:chararray, p_color:chararray, p_type:chararray, p_size:int, p_container:chararray);
```

**Verify Data Loaded:**

```
PartG = GROUP PartData ALL;  
Count = FOREACH PartG GENERATE COUNT(PartData);  
DUMP Count;
```



A screenshot of a terminal window titled "Downloads — ec2-user@ip-172-31-9-235:~/pig-0.15.0 — ssh -i lwang73.pem ec2-user@ec2-18-19...". The window displays a series of log messages from a Pig Latin script. The messages show the client attempting to connect to a server at 0.0.0.0:10020, with retry counts increasing from 6 to 10. A warning message indicates that a job aggregation failed. The script then succeeds in retrieving the job. The final message shows the total number of input paths to process is 1,000,000. The command "grunt>" is visible at the bottom.

```
2019-06-17 02:51:59,901 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 6 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISSECONDS)
2019-06-17 02:52:00,902 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 7 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISSECONDS)
2019-06-17 02:52:01,904 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 8 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISSECONDS)
2019-06-17 02:52:02,906 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISSECONDS)
2019-06-17 02:52:03,013 [main] WARN org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Unable to retrieve job to compute warning aggregation.
2019-06-17 02:52:03,013 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-06-17 02:52:03,016 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2019-06-17 02:52:03,032 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2019-06-17 02:52:03,033 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(200000)
grunt> ■
```

**Switch Columns:**

```
PartSwitchedPig = FOREACH PartData GENERATE p_name, p_partkey, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container;
```

**Write to file:**

```
STORE PartSwitchedPig INTO 'partOutPig' USING PigStorage('~');
```

```
Downloads — ec2-user@ip-172-31-9-235:~/pig-0.15.0 — ssh -i lwang73.pem ec2-user@ec2-18-191-171-26.u...  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:00,381 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server:  
0.0.0.0/0.0.0.0:10020. Already tried 4 time(s); retry policy is RetryUpToMaximumCountWithFixedSl  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:01,383 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server:  
0.0.0.0/0.0.0.0:10020. Already tried 5 time(s); retry policy is RetryUpToMaximumCountWithFixedSl  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:02,384 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server:  
0.0.0.0/0.0.0.0:10020. Already tried 6 time(s); retry policy is RetryUpToMaximumCountWithFixedSl  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:03,399 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server:  
0.0.0.0/0.0.0.0:10020. Already tried 7 time(s); retry policy is RetryUpToMaximumCountWithFixedSl  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:04,401 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server:  
0.0.0.0/0.0.0.0:10020. Already tried 8 time(s); retry policy is RetryUpToMaximumCountWithFixedSl  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:05,402 [main] INFO org.apache.hadoop.ipc.Client - Retrying connect to server:  
0.0.0.0/0.0.0.0:10020. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSl  
eep(maxRetries=10, sleepTime=1000 MILLISECONDS)  
2019-06-17 02:56:05,507 [main] WARN org.apache.pig.backend.hadoop.executionengine.mapReduceLaye  
r.MapReduceLauncher - Unable to retrieve job to compute warning aggregation.  
2019-06-17 02:56:05,507 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLaye  
r.MapReduceLauncher - Success!  
grunt> █
```

### Output:

```
Downloads — ec2-user@ip-172-31-9-235:~ — ssh -i lwang73.pem ec2-user@ec2-18-191-171-26.u...  
[ec2-user@ip-172-31-9-235 ~]$ hadoop fs -cat partOutPig/part-m-00000 | head  
lace spring-1~MFGR#1~MFGR#11~goldenrod~PROMO BURNISHED COPPER~7~JUMBO PKG  
rosy metallic~2~MFGR#4~MFGR#43~MFGR#4318~blush~LARGE BRUSHED BRASS~1~LG CASE  
green antique~3~MFGR#3~MFGR#32~MFGR#3210~dark~STANDARD POLISHED BRASS~21~WRAP CASE  
metallic smoke~4~MFGR#1~MFGR#14~MFGR#1426~chocolate~SMALL PLATED BRASS~14~MED DRUM  
blush chiffon~5~MFGR#4~MFGR#45~MFGR#4510~forest~STANDARD POLISHED TIN~15~SM PKG  
ivory azure~6~MFGR#2~MFGR#23~MFGR#2325~white~PROMO PLATED STEEL~4~MED BAG  
blanched tan~7~MFGR#5~MFGR#51~MFGR#513~blue~SMALL PLATED COPPER~45~SM BAG  
khaki cream~8~MFGR#1~MFGR#13~MFGR#1328~ivory~PROMO BURNISHED TIN~41~LG DRUM  
rose moccasin~9~MFGR#4~MFGR#41~MFGR#4117~thistle~SMALL BURNISHED STEEL~12~WRAP CASE  
moccasin royal~10~MFGR#2~MFGR#21~MFGR#2128~floral~LARGE BURNISHED STEEL~44~LG CAN  
cat: Unable to write to output stream.  
[ec2-user@ip-172-31-9-235 ~]$ █
```

## Part 2: Querying (25 pts)

Implement the following query:

```
select c_nation, sum(lo_revenue)
from customer, lineorder
where lo_custkey = c_custkey
  and c_region = 'AMERICA'
  and lo_discount BETWEEN 4 and 6
group by c_nation;
```

using Hive, MapReduce with HadoopStreaming and Pig (i.e. 3 different solutions). In Hive, this merely requires pasting the query into the Hive prompt and timing it. In Hadoop streaming, this will require a total of 2 passes (one for join and another one for GROUP BY).

**Hive:**

**Create and load tables:**

```
CREATE TABLE lineorder (
  lo_orderkey      INT,
  lo_linenumber    INT,
  lo_custkey       INT,
  lo_partkey       INT,
  lo_suppkey       INT,
  lo_orderdate     INT,
  lo_orderpriority VARCHAR(15),
  lo_shippriority  VARCHAR(1),
  lo_quantity      INT,
  lo_extendedprice INT,
  lo_ordertotalprice INT,
  lo_discount      INT,
  lo_revenue       INT,
  lo_supplycost    INT,
  lo_tax           INT,
  lo_commitdate    INT,
  lo_shipmode      VARCHAR(10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl' OVERWRITE INTO TABLE lineorder;
```

```
CREATE TABLE customer (
    c_custkey    INT,
    c_name      VARCHAR(25),
    c_address    VARCHAR (25),
    c_city      VARCHAR (10),
    c_nation    VARCHAR (15),
    c_region    VARCHAR (12),
    c_phone     VARCHAR (15),
    c_mktsegment VARCHAR (10)
)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl' OVERWRITE INTO TABLE customer;
```

**Execute sql statement:**

```
select c_nation, sum(lo_revenue)
from customer, lineorder
where lo_custkey = c_custkey
and c_region = 'AMERICA'
and lo_discount BETWEEN 4 and 6
group by c_nation;
```

**End of output with time taken:**

```
Downloads — ec2-user@ip-172-31-9-235:~/apache-hive-2.0.1-bin — ssh -i lwang73.pem ec2-user...
set mapreduce.job.reduces=<number>
Starting Job = job_1560737226405_0006, Tracking URL = http://ip-172-31-9-235.us-east-2.compute.internal:8088/proxy/application_1560737226405_0006/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job -kill job_1560737226405_0006
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 3
2019-06-17 03:01:20,140 Stage-2 map = 0%, reduce = 0%
2019-06-17 03:01:28,710 Stage-2 map = 33%, reduce = 0%, Cumulative CPU 6.08 sec
2019-06-17 03:01:30,812 Stage-2 map = 67%, reduce = 0%, Cumulative CPU 11.29 sec
2019-06-17 03:01:31,875 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 16.91 sec
2019-06-17 03:01:33,969 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 21.09 sec
MapReduce Total cumulative CPU time: 21 seconds 90 msec
Ended Job = job_1560737226405_0006
MapReduce Jobs Launched:
Stage-Stage-2: Map: 3 Reduce: 3 Cumulative CPU: 21.09 sec HDFS Read: 594381661 HDFS Write: 108 SUCCESS
Total MapReduce CPU Time Spent: 21 seconds 90 msec
OK
ARGENTINA      243988697072
BRAZIL        225595365795
UNITED STATES  244263170830
CANADA        240715548308
PERU          228441124985
Time taken: 28.547 seconds, Fetched: 5 row(s)
hive> ■
```

**Hadoop Streaming:**

**Join**

**lineCustMapJoin.py**

```
#!/usr/bin/python
import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    line = line.strip().split('|')
    if line[1].startswith('Customer#'):
        if line[5] == 'AMERICA': # Return on matching records
            print line[0], '\t', line[4], '\t', 'customer'
    # lineorder
    else:
        if 4 <= int(line[11]) <= 6: # Return on matching records
            print line[2], '\t', line[12], '\t', 'lineorder'
```

**lineCustReduceJoin.py**

```
#!/usr/bin/python

import sys

currentKey = None
revenue = []
nation = ""

# input comes from STDIN
for line in sys.stdin:

    split = line.strip().split('\t')
    key = split[0] # key is customer id
    value = '\t'.join(split[1:])

    if currentKey == key: # Same key
        if value.endswith('lineorder'):
            revenue.extend([split[1]])
        if value.endswith('customer'):
            nation = split[1]
    else:
        # Do not print anything until all records
        # for a key have been seen, this is signaled
        # by currentKey != key
        # Check for values and then iterate results
        if currentKey:
            print currentKey, '\t', nation, '\t', len(revenue)
        currentKey = key
        revenue = [split[1]]
        nation = split[1]
```

```
lenNation = len(nation)
if (lenNation > 0):
    i = 0
    while i < lenNation:
        print nation, '\t', revenue[i]
        i += 1

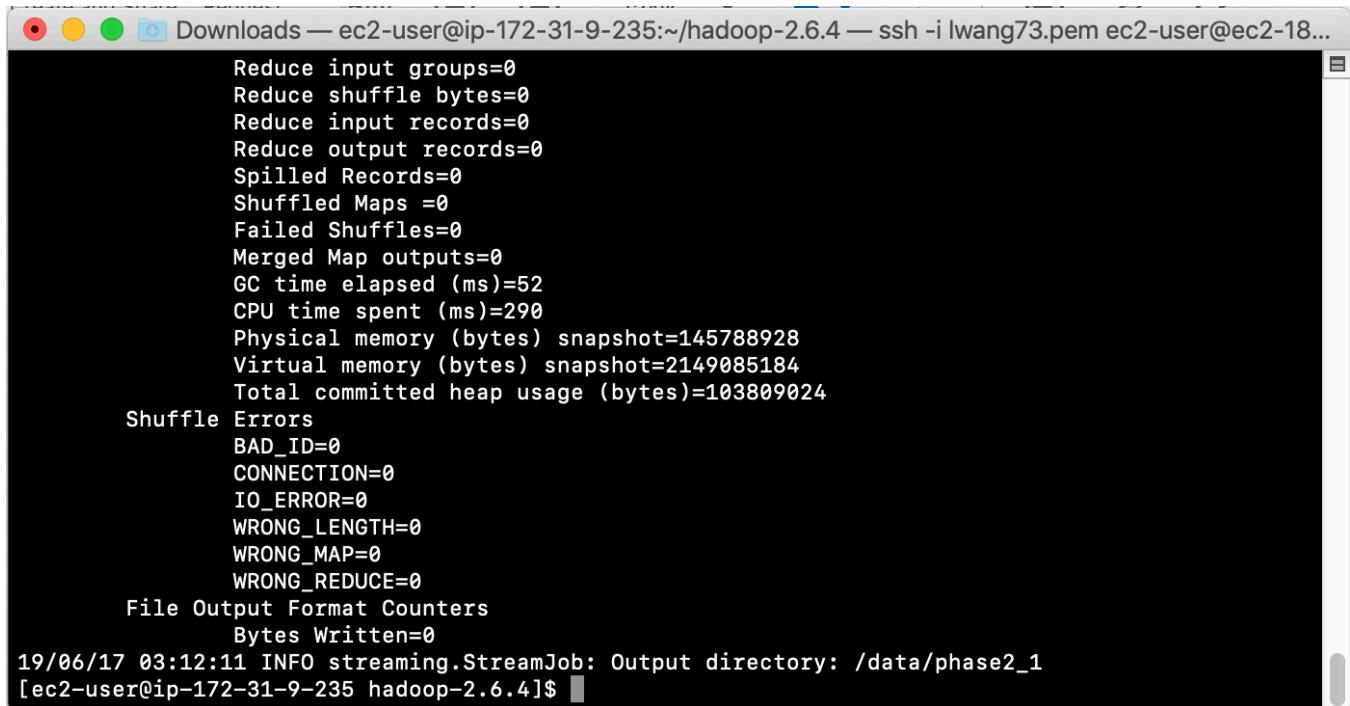
# reset values
revenue = []
nation = ""

if value.endswith('lineorder'):
    revenue.extend([split[1]])
if value.endswith('customer'):
    nation = split[1]

# set the current key at the end of each iteration
currentKey = key
```

**Commands:**

```
hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/phase2 -output /data/phase2_1 -
mapper lineCustMapJoin.py -reducer lineCustReduceJoin.py -file lineCustMapJoin.py -file
lineCustReduceJoin.py
```

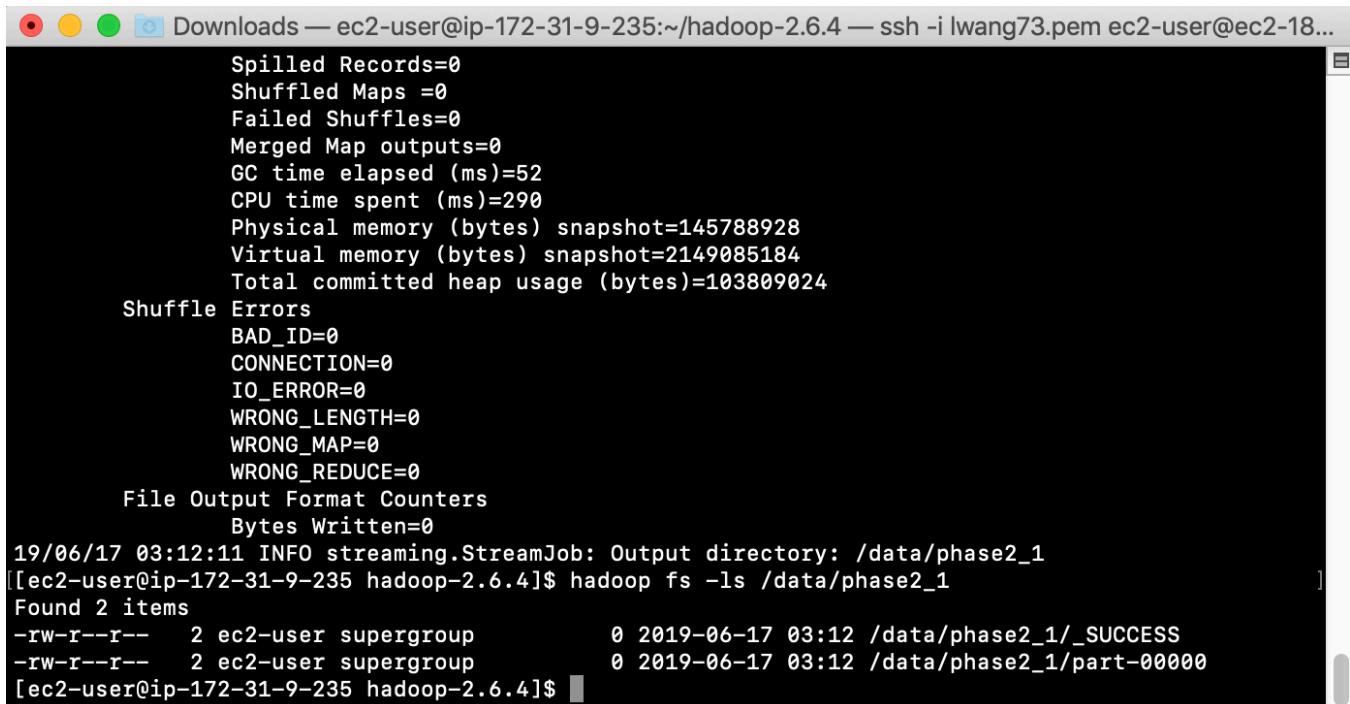


The screenshot shows a terminal window with the following content:

```
Downloads — ec2-user@ip-172-31-9-235:~/hadoop-2.6.4 — ssh -i lwang73.pem ec2-user@ec2-18...
```

```
Reduce input groups=0
Reduce shuffle bytes=0
Reduce input records=0
Reduce output records=0
Spilled Records=0
Shuffled Maps =0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=52
CPU time spent (ms)=290
Physical memory (bytes) snapshot=145788928
Virtual memory (bytes) snapshot=2149085184
Total committed heap usage (bytes)=103809024
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
Bytes Written=0
19/06/17 03:12:11 INFO streaming.StreamJob: Output directory: /data/phase2_1
[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$
```

```
hadoop fs -ls /data/phase2_1
```



```

Downloads — ec2-user@ip-172-31-9-235:~/hadoop-2.6.4 — ssh -i lwang73.pem ec2-user@ec2-18...
Spilled Records=0
Shuffled Maps =0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=52
CPU time spent (ms)=290
Physical memory (bytes) snapshot=145788928
Virtual memory (bytes) snapshot=2149085184
Total committed heap usage (bytes)=103809024
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
Bytes Written=0
19/06/17 03:12:11 INFO streaming.StreamJob: Output directory: /data/phase2_1
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ hadoop fs -ls /data/phase2_1
Found 2 items
-rw-r--r-- 2 ec2-user supergroup 0 2019-06-17 03:12 /data/phase2_1/_SUCCESS
-rw-r--r-- 2 ec2-user supergroup 0 2019-06-17 03:12 /data/phase2_1/part-00000
[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ ]

```

**Group:**

**lineCustReduceGroup.py**

```

#!/usr/bin/python
import sys

curr_id = None
curr_tot = 0
id = None

# The input comes from standard input (line by line)
for line in sys.stdin:
    # parse the line and split it by '\t'
    line = line.strip().split('\t')

    # grab the key
    # values include some whitespace, removing here
    id = line[0].strip() + '\t' + line[1].strip()

    # grab the value (int)
    val = int(line[2])

    if curr_id == id:
        curr_tot += val

    else:

```

```

if curr_id: # output the sum, single key completed
    print '%s\t%d' % (curr_id, curr_tot)
    curr_tot = val

# set curr_id to id at end of each iteration
curr_id = id

# output the last key
if curr_id == id:
    print '%s\t%d' % (curr_id, curr_tot)

hadoop jar hadoop-streaming-2.6.4.jar -D stream.num.map.output.key.fields=2 -input
/data/phase2_01/part-00000 -output /data/phase2_06 -mapper /bin/cat -reducer
lineCustReduceGroup.py -file lineCustReduceGroup.py

```

```

Downloads — ec2-user@ip-172-31-9-235:~/hadoop-2.6.4 — ssh -i lwang73.pem ec2-user@ec2-18...
File Output Format Counters
Bytes Written=0
19/06/17 03:12:11 INFO streaming.StreamJob: Output directory: /data/phase2_1
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ hadoop fs -ls /data/phase2_1
Found 2 items
-rw-r--r-- 2 ec2-user supergroup 0 2019-06-17 03:12 /data/phase2_1/_SUCCESS
-rw-r--r-- 2 ec2-user supergroup 0 2019-06-17 03:12 /data/phase2_1/part-00000
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ hadoop fs -cat /data/phase2_1/part-00000 | head
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ nano lineCustReduceGroup.py
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ hadoop jar hadoop-streaming-2.6.4.jar -D stream.num.map
.output.key.fields=2 -input /data/phase2_01/part-00000 -output /data/phase2_06 -mapper /bin/cat
-reducer lineCustReduceGroup.py -file lineCustReduceGroup.py
19/06/17 03:15:15 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [lineCustReduceGroup.py, /tmp/hadoop-unjar1800942159855886489/] [] /tmp/streamjob
7684805959649632280.jar tmpDir=null
19/06/17 03:15:16 INFO client.RMProxy: Connecting to ResourceManager at /172.31.9.235:8032
19/06/17 03:15:16 INFO client.RMProxy: Connecting to ResourceManager at /172.31.9.235:8032
19/06/17 03:15:17 INFO mapreduce.JobSubmitter: Cleaning up the staging area /tmp/hadoop-yarn/staging/ec2-user/.staging/job_1560737226405_0009
19/06/17 03:15:17 ERROR streaming.StreamJob: Error Launching job : Input path does not exist: hdfs://172.31.9.235/data/phase2_01/part-00000
Streaming Command Failed!
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ 

```

This one didn't work.

**Pig:**

#### Load Tables:

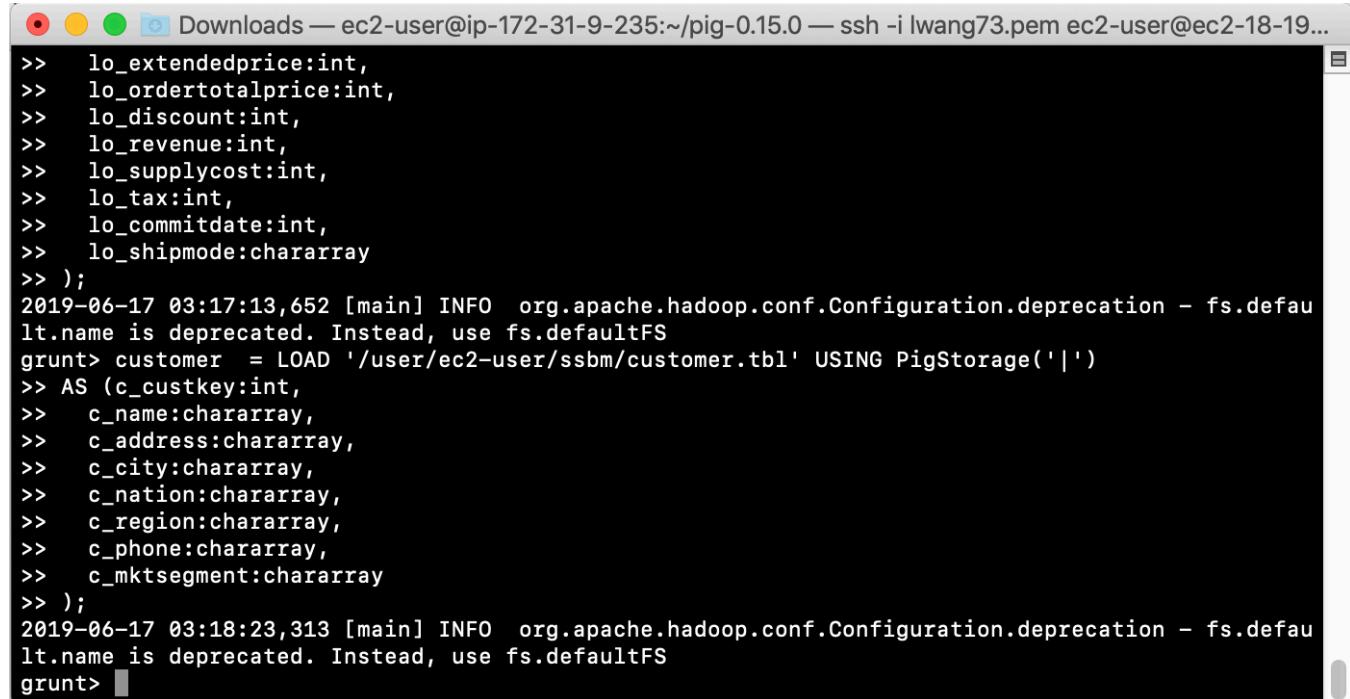
```

lineorder = LOAD '/user/ec2-user/ssbm/lineorder.tbl' USING PigStorage('|')
AS (lo_orderkey:int,
    lo_linenumber:int,
    lo_custkey:int,
    lo_partkey:int,
    lo_suppkey:int,

```

```
lo_orderdate:int,
lo_orderpriority:chararray,
lo_shipppriority:chararray,
lo_quantity:int,
lo_extendedprice:int,
lo_ordertotalprice:int,
lo_discount:int,
lo_revenue:int,
lo_supplycost:int,
lo_tax:int,
lo_commitdate:int,
lo_shipmode:chararray
);

customer = LOAD '/user/ec2-user/ssbm/customer.tbl' USING PigStorage('|')
AS (c_custkey:int,
c_name:chararray,
c_address:chararray,
c_city:chararray,
c_nation:chararray,
c_region:chararray,
c_phone:chararray,
c_mktsegment:chararray
);
```

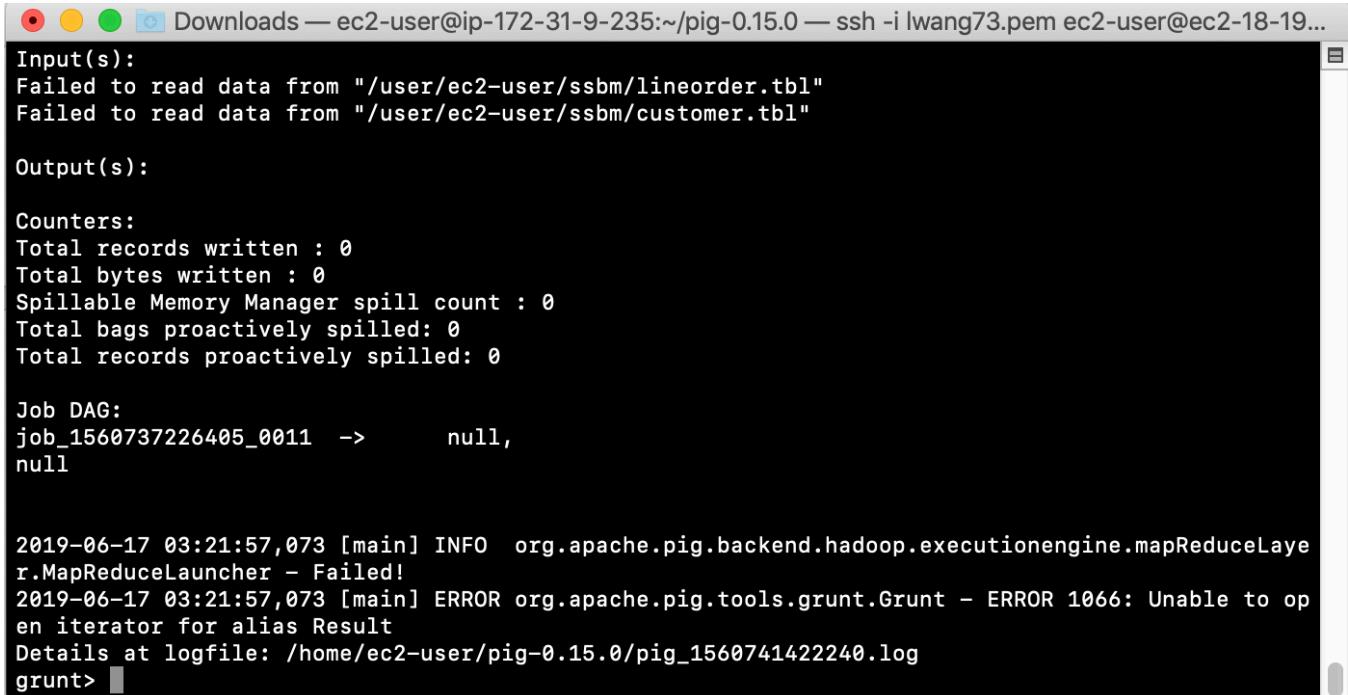


A screenshot of a terminal window titled "Downloads — ec2-user@ip-172-31-9-235:~/pig-0.15.0 — ssh -i lwang73.pem ec2-user@ec2-18-19...". The terminal displays the following Pig Latin script:

```
>> lo_extendedprice:int,
>> lo_ordertotalprice:int,
>> lo_discount:int,
>> lo_revenue:int,
>> lo_supplycost:int,
>> lo_tax:int,
>> lo_commitdate:int,
>> lo_shipmode:chararray
>> );
2019-06-17 03:17:13,652 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.defau
lt.name is deprecated. Instead, use fs.defaultFS
grunt> customer = LOAD '/user/ec2-user/ssbm/customer.tbl' USING PigStorage('|')
>> AS (c_custkey:int,
>> c_name:chararray,
>> c_address:chararray,
>> c_city:chararray,
>> c_nation:chararray,
>> c_region:chararray,
>> c_phone:chararray,
>> c_mktsegment:chararray
>> );
2019-06-17 03:18:23,313 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.defau
lt.name is deprecated. Instead, use fs.defaultFS
grunt> █
```

**Execution steps:**

```
FilteredLineorder = FILTER lineorder BY lo_discount >= 4 AND lo_discount <= 6;
FilteredCustomer = FILTER customer BY c_region == 'AMERICA';
JoinedData = JOIN FilteredLineorder BY (lo_custkey), FilteredCustomer BY (c_custkey);
GroupedData = GROUP JoinedData BY (c_nation);
Result = FOREACH GroupedData GENERATE group, SUM(JoinedData.lo_revenue) as rev;
DUMP Result;
```



```
Downloads — ec2-user@ip-172-31-9-235:~/pig-0.15.0 — ssh -i lwang73.pem ec2-user@ec2-18-19...
Input(s):
Failed to read data from "/user/ec2-user/ssbm/lineorder.tbl"
Failed to read data from "/user/ec2-user/ssbm/customer.tbl"

Output(s):

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

Job DAG:
job_1560737226405_0011  ->      null,
null

2019-06-17 03:21:57,073 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Failed!
2019-06-17 03:21:57,073 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1066: Unable to open iterator for alias Result
Details at logfile: /home/ec2-user/pig-0.15.0/pig_1560741422240.log
grunt> 
```

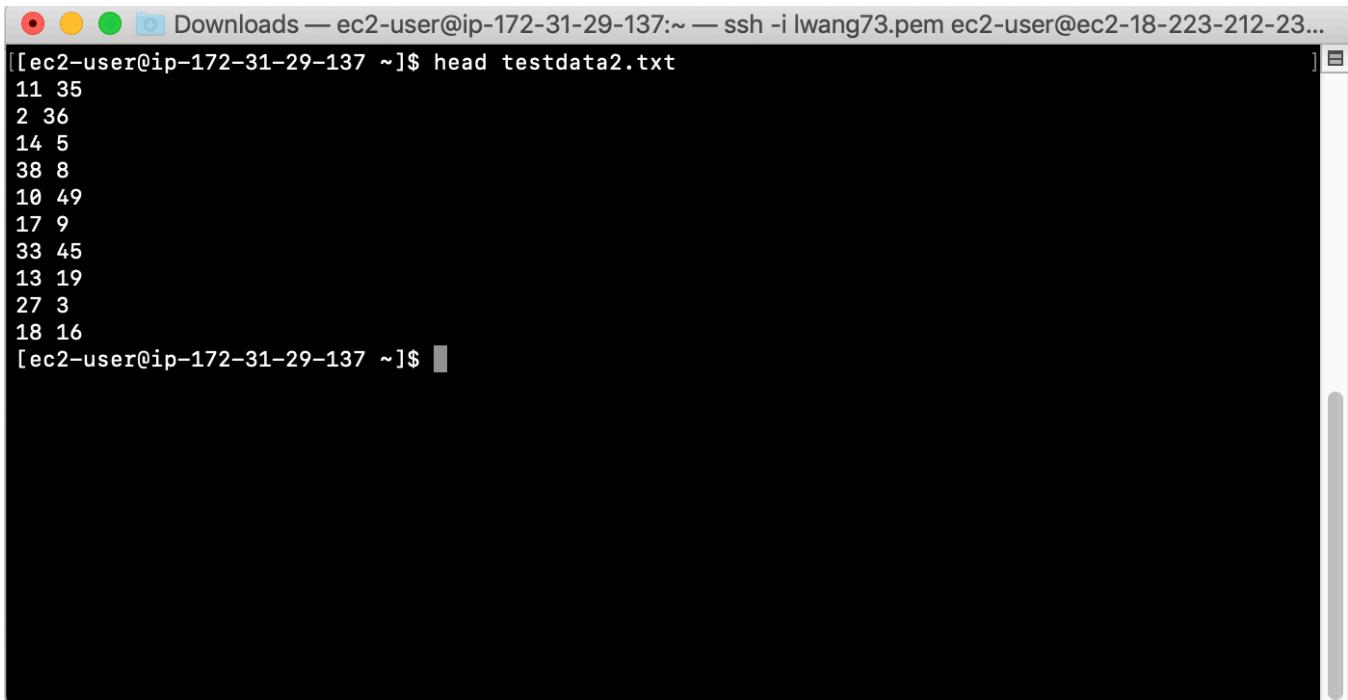
## Part 3: Clustering (30 pts)

Create a new numeric file with 125,000 rows and 3 columns, separated by space – you can generate numeric data as you prefer, but submit the code that you have used.

- A. (5 pts) Using Mahout synthetic clustering as you have in a previous assignment on sample data. This entails running the **same** clustering command, but substituting your own input data and the right number of clusters.

Note, I used the single-node Hadoop instance for this exercise.

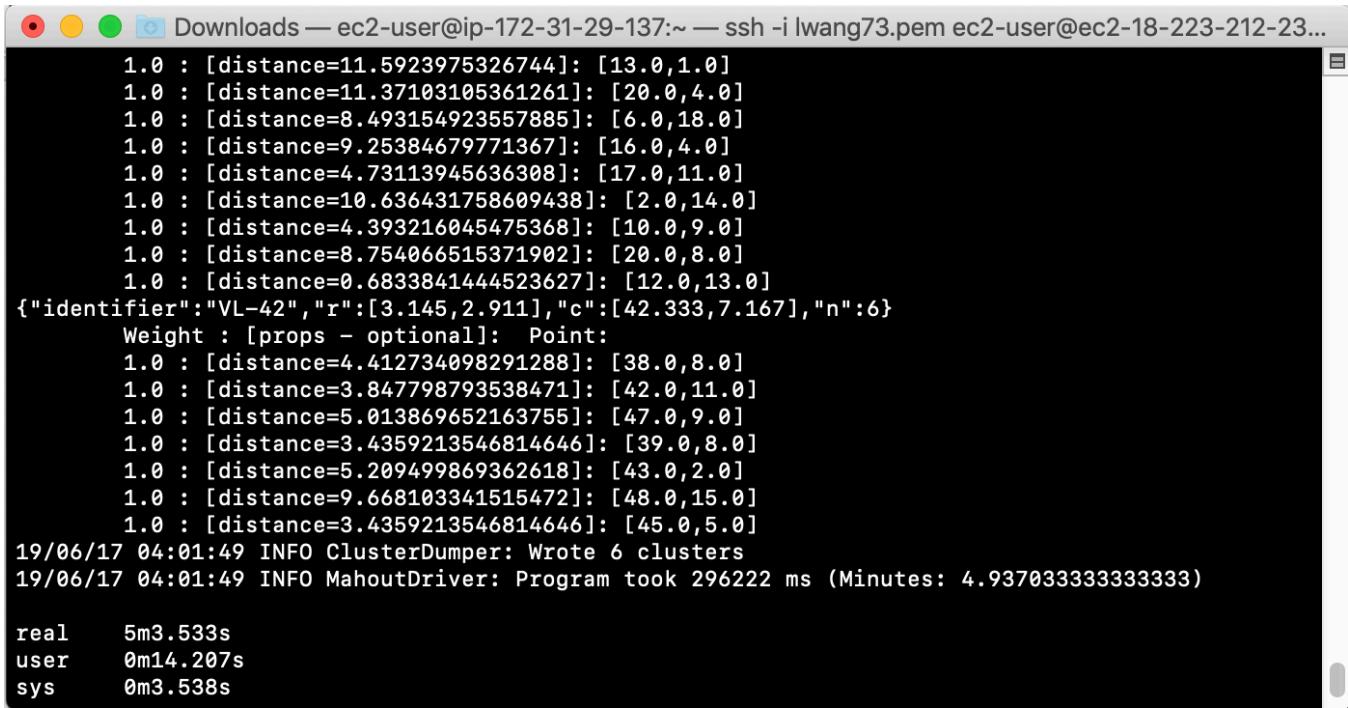
First, I used an online random sequence generator to generate 100 x and y variables. Here's a screen shot of the first 10 records. The full list is at the end of this document.



```
Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@ec2-18-223-212-23...
[[ec2-user@ip-172-31-29-137 ~]$ head testdata2.txt
11 35
2 36
14 5
38 8
10 49
17 9
33 45
13 19
27 3
18 16
[ec2-user@ip-172-31-29-137 ~]$ ]
```

**Commands:**

```
hadoop fs -put testdata2.txt testdata/
time mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job
```



```
Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@ec2-18-223-212-23...
1.0 : [distance=11.5923975326744]: [13.0,1.0]
1.0 : [distance=11.37103105361261]: [20.0,4.0]
1.0 : [distance=8.493154923557885]: [6.0,18.0]
1.0 : [distance=9.25384679771367]: [16.0,4.0]
1.0 : [distance=4.73113945636308]: [17.0,11.0]
1.0 : [distance=10.636431758609438]: [2.0,14.0]
1.0 : [distance=4.393216045475368]: [10.0,9.0]
1.0 : [distance=8.754066515371902]: [20.0,8.0]
1.0 : [distance=0.6833841444523627]: [12.0,13.0]
{"identifier":"VL-42","r":[3.145,2.911],"c":[42.333,7.167],"n":6}
Weight : [props - optional]: Point:
1.0 : [distance=4.412734098291288]: [38.0,8.0]
1.0 : [distance=3.847798793538471]: [42.0,11.0]
1.0 : [distance=5.013869652163755]: [47.0,9.0]
1.0 : [distance=3.4359213546814646]: [39.0,8.0]
1.0 : [distance=5.209499869362618]: [43.0,2.0]
1.0 : [distance=9.668103341515472]: [48.0,15.0]
1.0 : [distance=3.4359213546814646]: [45.0,5.0]
19/06/17 04:01:49 INFO ClusterDumper: Wrote 6 clusters
19/06/17 04:01:49 INFO MahoutDriver: Program took 296222 ms (Minutes: 4.937033333333333)
real      5m3.533s
user      0m14.207s
sys       0m3.538s
```

# Lavinia Wang 1473704

```
mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints --
output clusteranalyze.txt
```

```
Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@ec2-18-223-212-23...
1.0 : [distance=4.412734098291288]: [38.0,8.0]
1.0 : [distance=3.847798793538471]: [42.0,11.0]
1.0 : [distance=5.013869652163755]: [47.0,9.0]
1.0 : [distance=3.4359213546814646]: [39.0,8.0]
1.0 : [distance=5.209499869362618]: [43.0,2.0]
1.0 : [distance=9.668103341515472]: [48.0,15.0]
1.0 : [distance=3.4359213546814646]: [45.0,5.0]
19/06/17 04:01:49 INFO ClusterDumper: Wrote 6 clusters
19/06/17 04:01:49 INFO MahoutDriver: Program took 296222 ms (Minutes: 4.937033333333333)

real      5m3.533s
user     0m14.207s
sys      0m3.538s
[ec2-user@ip-172-31-29-137 ~]$ mahout clusterdump --input output/clusters-10-final --pointsDir o
utput/clusteredPoints --output clusteranalyze.txt
Running on hadoop, using /home/ec2-user/hadoop-2.6.4/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/ec2-user/apache-mahout-distribution-0.11.2/mahout-examples-0.11.2-job.jar
19/06/17 04:02:52 INFO AbstractJob: Command line arguments: {--dictionaryType=[text], --distance
Measure=[org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure], --endPhase=[2147483
647], --input=[output/clusters-10-final], --output=[clusteranalyze.txt], --outputFormat=[TEXT],
--pointsDir=[output/clusteredPoints], --startPhase=[0], --tempDir=[temp]}
19/06/17 04:02:54 INFO ClusterDumper: Wrote 6 clusters
19/06/17 04:02:54 INFO MahoutDriver: Program took 1876 ms (Minutes: 0.031266666666666665)
[ec2-user@ip-172-31-29-137 ~]$
```

more clusteranalyze.txt

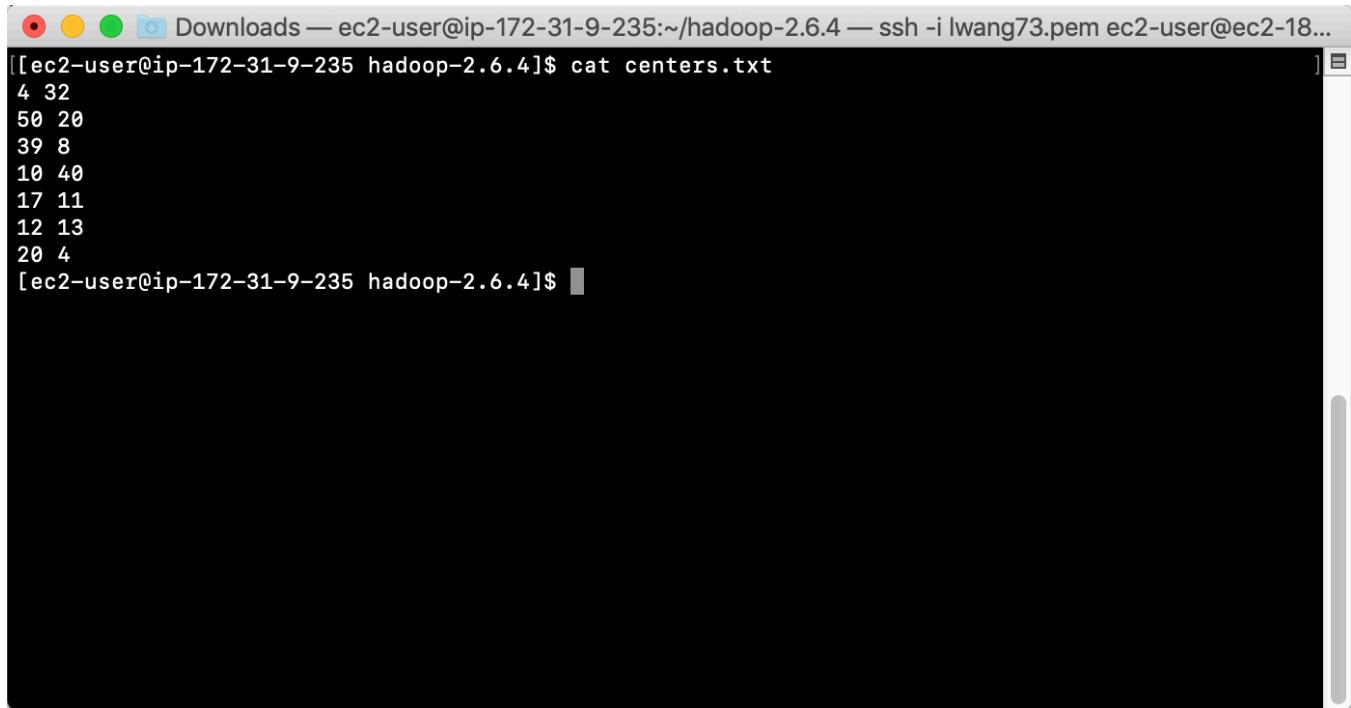
```
Downloads — ec2-user@ip-172-31-29-137:~ — ssh -i lwang73.pem ec2-user@ec2-18-223-212-23...
{"identifier": "CL-85", "r": [4.957, 4.351], "c": [43.882, 23.882], "n": 17}
  Weight : [props - optional]: Point:
  1.0 : [distance=2.394486169064421]: [46.0,25.0]
  1.0 : [distance=2.0788937702437997]: [43.0,22.0]
  1.0 : [distance=10.118331030232685]: [44.0,34.0]
  1.0 : [distance=5.3218229309098515]: [39.0,26.0]
  1.0 : [distance=3.014383053016168]: [41.0,23.0]
  1.0 : [distance=2.3944861690642307]: [45.0,26.0]
  1.0 : [distance=4.630274977448127]: [46.0,28.0]
  1.0 : [distance=8.883132026603901]: [35.0,24.0]
  1.0 : [distance=7.245568983606047]: [50.0,20.0]
  1.0 : [distance=4.835351366485131]: [40.0,21.0]
  1.0 : [distance=7.9759586856387665]: [49.0,30.0]
  1.0 : [distance=11.086479811106678]: [33.0,26.0]
  1.0 : [distance=6.7626595370342875]: [50.0,21.0]
  1.0 : [distance=3.1198660249824663]: [47.0,24.0]
  1.0 : [distance=5.193154930433593]: [49.0,23.0]
  1.0 : [distance=6.550575135213391]: [41.0,18.0]
{"identifier": "VL-20", "r": [8.56, 6.191], "c": [34.826, 40.565], "n": 23}
  Weight : [props - optional]: Point:
  1.0 : [distance=4.796028602830576]: [33.0,45.0]
  1.0 : [distance=6.849447703094687]: [28.0,40.0]
  1.0 : [distance=7.746088713613348]: [37.0,48.0]
--More-- (20%)
```

- B. (25 pts) Using Hadoop streaming perform two iterations manually **using 7 centers** (initially with randomly chosen centers). As discussed in class, this would require passing a text file with cluster centers using -file option, opening the centers.txt in the mapper with open('centers.txt', 'r') and assigning a key to each point based on which center is the closest to each particular point. Your reducer would then compute the new centers, and at that point the iteration is done and the output of the reducer with new centers can be given to the next pass of the same code.

The only difference between first and second iteration is that in first iteration you have to pick the initial centers. In the 2<sup>nd</sup> iteration, the centers will be given to you by a previous pass of KMeans.

Note: I used the multi-node Hadoop instance for this exercise.

Using the same 100 points from exercise 3.A, I randomly picked seven starting centers from the list:



```
Downloads — ec2-user@ip-172-31-9-235:~/hadoop-2.6.4 — ssh -i lwang73.pem ec2-user@ec2-18...  
[[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ cat centers.txt  
4 32  
50 20  
39 8  
10 40  
17 11  
12 13  
20 4  
[ec2-user@ip-172-31-9-235 hadoop-2.6.4]$ ]
```

I used the same testdate2.txt file for this exercise as I did for part 3.A.

**Code:**

**kmeansMapper.py**

```
#!/usr/bin/python
```

```
import sys
import math

fd = open('centers.txt', 'r')
centers = []

for line in fd:
    line = line.strip()
    vals = line.split(' ')
    centers.extend([vals])
fd.close()

for line in sys.stdin:
    line = line.strip()
    vals = line.split(' ')

clusterNum = None
distance = None
i = 0

#compare to each center and store the smallest distance
for center in centers:

    euclidDist = math.sqrt( (float(vals[0])-float(center[0]))**2 + (float(v$

        if clusterNum:
            if euclidDist < distance:
                clusterNum = i+1
                distance = euclidDist
        else: #always record the first cluster
            clusterNum = i+1
            distance = euclidDist

    i += 1

print clusterNum, '\t', vals[0], '\t', vals[1]
```

**kmeansReducer.py**

```
#!/usr/bin/python

import sys

currId = None # this is the "current" key
currXs = []
currYs = []
id = None
```

```
# The input comes from standard input (line by line)
for line in sys.stdin:

    line = line.strip()
    ln = line.split('\t')
    id = ln[0]

    if currId == id:
        currXs.append(float(ln[1]))
        currYs.append(float(ln[2]))
    else:
        if currId:
            #calculate center
            centerX = sum(currXs)/len(currXs)
            centerY = sum(currYs)/len(currYs)
            print '%s %s %s %s' % (centerX, centerY, currId, zip(currXs, cu$)

            currXs = []
            currYs = []

        currId = id
        currXs.append(float(ln[1]))
        currYs.append(float(ln[2]))

    # output the last key
if currId == id:
    #calculate center
    centerX = sum(currXs)/len(currXs)
    centerY = sum(currYs)/len(currYs)
    print '%s %s %s %s' % (centerX, centerY, currId, zip(currXs, currYs))
```

**Executions:**

**Execution 1:**

```
hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/data/testdata2.txt -file centers.txt -
mapper kmeansMapper.py -file kmeansMapper.py -reducer kmeansReducer.py -file
kmeansReducer.py -output /data/kmeans1
```

**Execution 2:**

Replace the centers file:

```
rm centers.txt
hadoop fs -get /data/kmeans1/part-00000 centers.txt
```

Run with new centers:

```
hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/data/testdata2.txt -file  
centers.txt -mapper kmeansMapper.py -file kmeansMapper.py -reducer kmeansReducer.py -  
file kmeansReducer.py -output /data/kmeans2
```

**testdata2.txt**

```
11 35  
2 36  
14 5  
38 8  
10 49  
17 9  
33 45  
13 19  
27 3  
18 16  
28 40  
4 32  
46 25  
21 15  
29 7  
43 22  
37 48  
44 34  
20 12  
31 6  
39 26  
41 23  
50 47  
24 30  
1 42  
45 26  
46 28  
1 48  
27 44  
35 24  
29 32  
7 23  
16 5  
14 18  
50 20  
39 36  
40 21
```

2 33  
10 31  
4 22  
42 11  
6 13  
47 9  
30 8  
37 17  
3 19  
12 38  
34 15  
25 43  
49 41  
17 7  
35 11  
39 8  
43 2  
48 15  
24 25  
23 50  
41 38  
3 16  
12 21  
22 46  
13 1  
20 4  
37 33  
36 19  
5 29  
34 47  
14 28  
42 44  
49 30  
6 18  
10 40  
31 45  
27 26  
32 9  
28 30  
33 26  
43 42  
16 4  
50 21  
47 24  
27 29  
7 35  
15 31

49 23  
41 18  
17 11  
2 14  
10 9  
25 3  
22 34  
20 8  
44 38  
40 46  
36 19  
48 39  
12 13  
1 37  
6 32  
45 5

Submit a single document containing your written answers. Be sure that this document contains your name and “CSC 555 Project Phase 2” at the top.