Project Phase 2

CSC 555 Mining Big Data
Michal Chowaniak

(due Sunday June 16th)

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

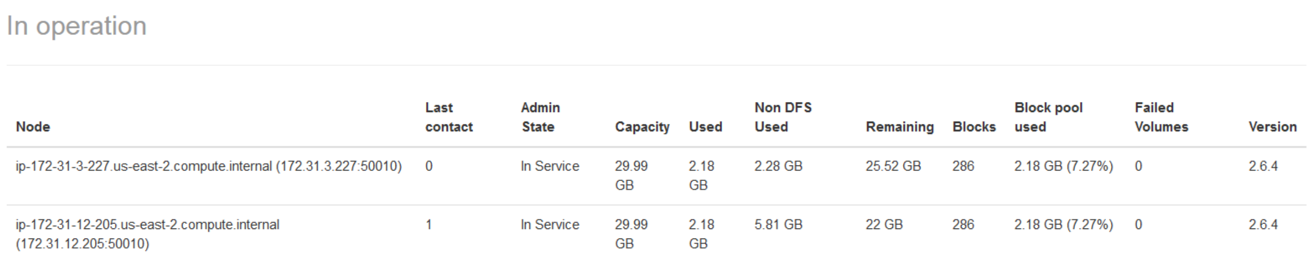
The data is available at (this is Scale1, the smallest denomination of this benchmark)
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/

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.

At the begging I used 4 node cluster, but after I turned off 2 instances, I was not able to get them back, so some of answers were run on 2 node cluster.



In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-3-227.us-east-2.compute.internal (172.31.3.227:50010)	0	In Service	29.99 GB	2.18 GB	2.28 GB	25.52 GB	286	2.18 GB (7.27%)	0	2.6.4
ip-172-31-12-205.us-east-2.compute.internal (172.31.12.205:50010)	1	In Service	29.99 GB	2.18 GB	5.81 GB	22 GB	286	2.18 GB (7.27%)	0	2.6.4

Part 1: Data Transformation (15 pts)

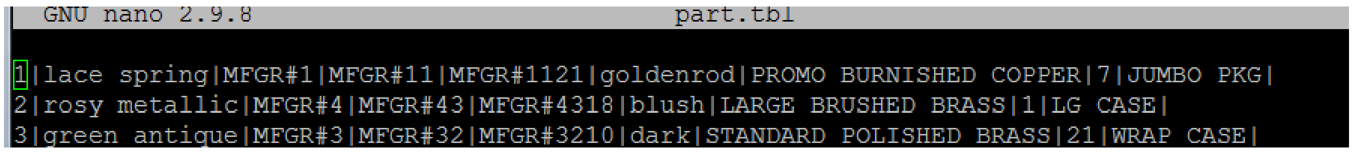
Transform part.tbl table into a ~-separated ('~') file: Use Hive, MapReduce with HadoopStreaming and Pig (i.e. 3 different solutions).

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.

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl

Hive

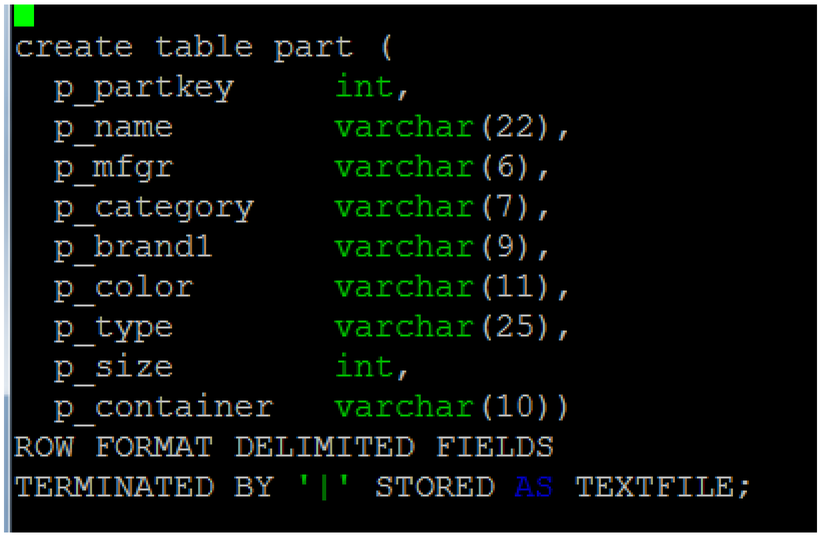
nano part.tbl



```
  GNU nano 2.9.8                              part.tbl

1|lace spring|MFGR#1|MFGR#11|MFGR#1121|goldenrod|PROMO BURNISHED COPPER|7|JUMBO PKG|
2|rosy metallic|MFGR#4|MFGR#43|MFGR#4318|blush|LARGE BRUSHED BRASS|1|LG CASE|
3|green antique|MFGR#3|MFGR#32|MFGR#3210|dark|STANDARD POLISHED BRASS|21|WRAP CASE|
```

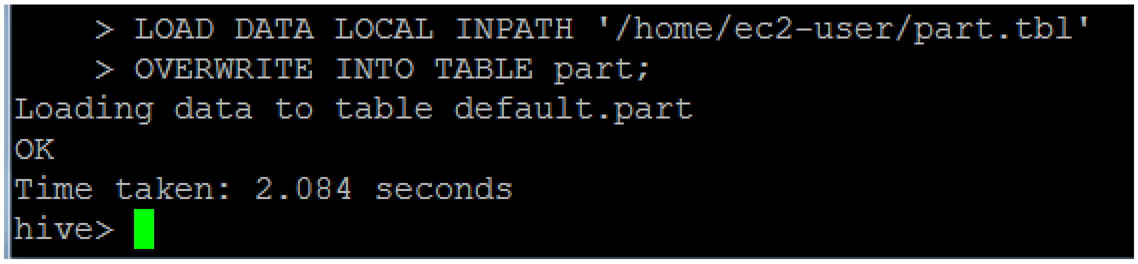
nano SSBM_schema_hive.sql

```
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;
```

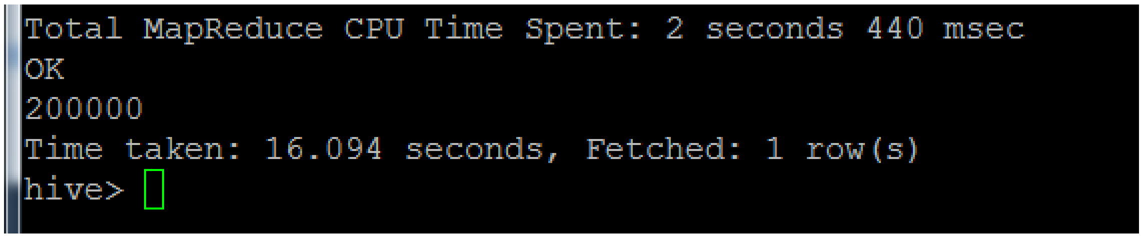

```
create table part (
  p_partkey     int,
  p_name        varchar(22),
  p_mfgr        varchar(6),
  p_category    varchar(7),
  p_brand1      varchar(9),
  p_color       varchar(11),
  p_type        varchar(25),
  p_size        int,
  p_container   varchar(10))
ROW FORMAT DELIMITED FIELDS
TERMINATED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

LOAD DATA

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

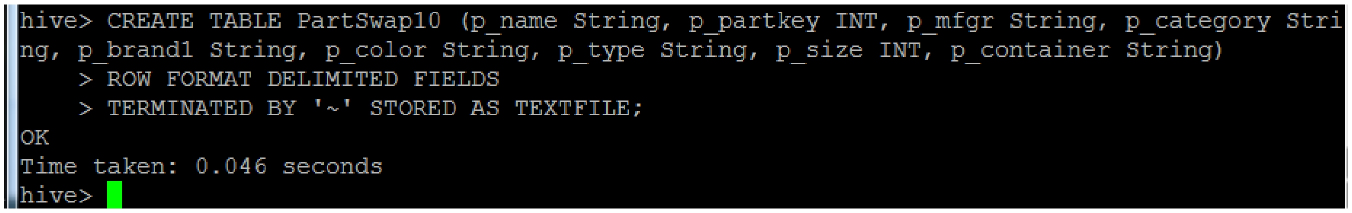

```
    > LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl'
    > OVERWRITE INTO TABLE part;
Loading data to table default.part
OK
Time taken: 2.084 seconds
hive>
```

SELECT COUNT(*) FROM part;


```
Total MapReduce CPU Time Spent: 2 seconds 440 msec
OK
200000
Time taken: 16.094 seconds, Fetched: 1 row(s)
hive>
```

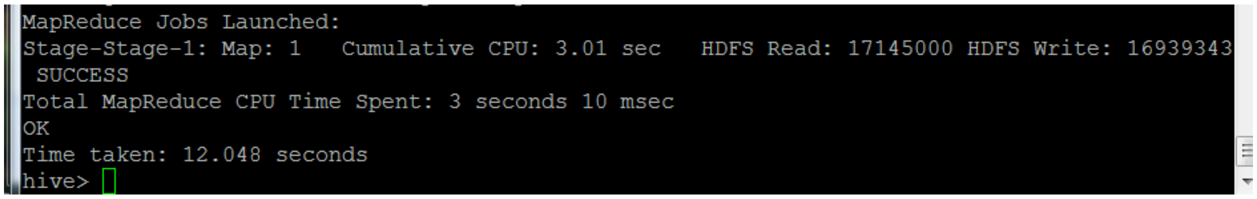
CREATE TABLE PartSwap10 (p_name String, p_partkey INT, p_mfgr String, p_category String, p_brand1 String, p_color String, p_type String, p_size INT, p_container String)
ROW FORMAT DELIMITED FIELDS

TERMINATED BY '~' STORED AS TEXTFILE;



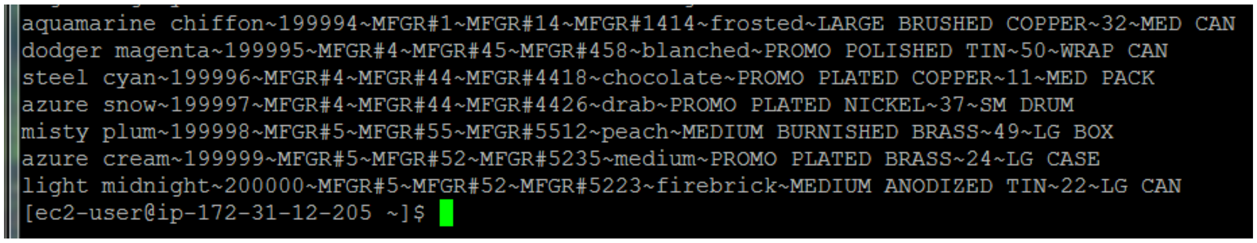
```
hive> CREATE TABLE PartSwap10 (p_name String, p_partkey INT, p_mfgr String, p_category Stri
ng, p_brand1 String, p_color String, p_type String, p_size INT, p_container String)
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '~' STORED AS TEXTFILE;
OK
Time taken: 0.046 seconds
hive>
```

INSERT OVERWRITE TABLE PartSwap10
SELECT p_name, p_partkey, p_mfgr, p_category, p_brand1,
p_color, p_type, p_size, p_container FROM part;



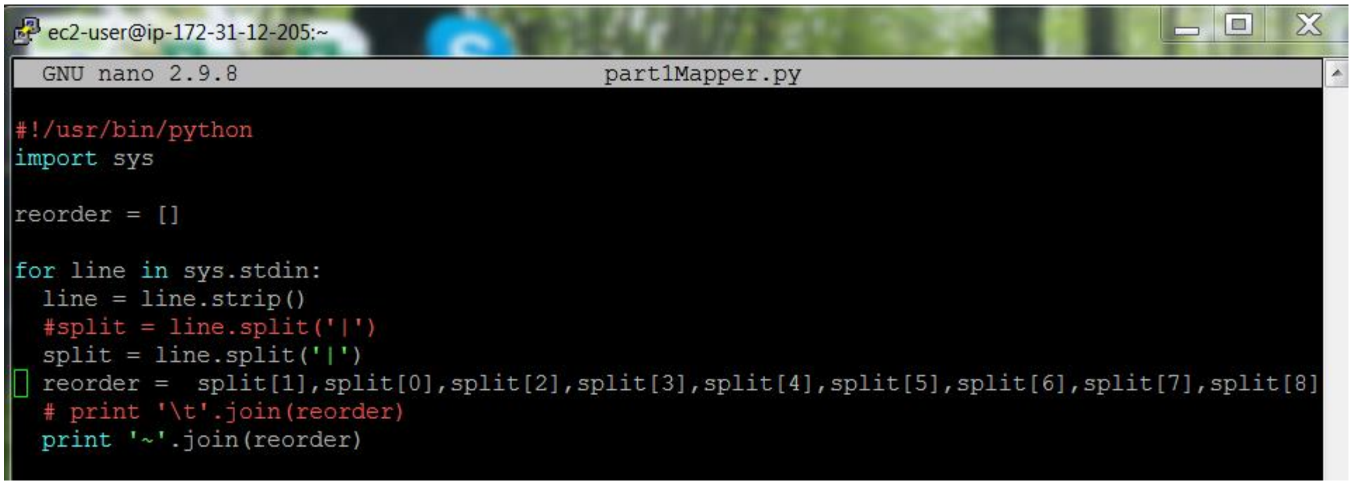
```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 3.01 sec   HDFS Read: 17145000 HDFS Write: 16939343
 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 10 msec
OK
Time taken: 12.048 seconds
hive>
```

hadoop fs -cat /user/hive/warehouse/partswap10/000000_0



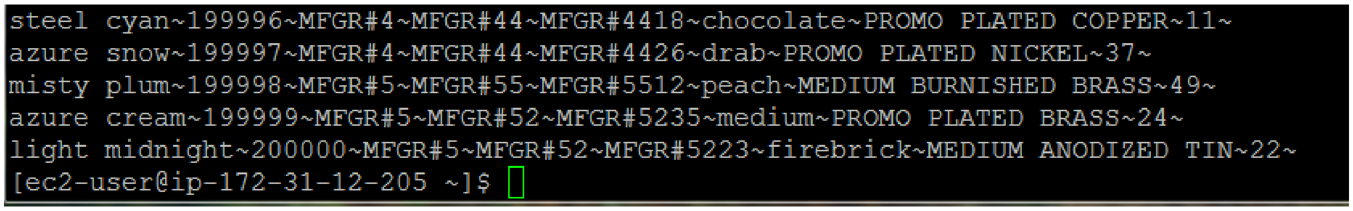
```
aquamarine chiffon~199994~MFGR#1~MFGR#14~MFGR#1414~frosted~LARGE BRUSHED COPPER~32~MED CAN
dodger magenta~199995~MFGR#4~MFGR#45~MFGR#458~blanched~PROMO POLISHED TIN~50~WRAP CAN
steel cyan~199996~MFGR#4~MFGR#44~MFGR#4418~chocolate~PROMO PLATED COPPER~11~MED PACK
azure snow~199997~MFGR#4~MFGR#44~MFGR#4426~drab~PROMO PLATED NICKEL~37~SM DRUM
misty plum~199998~MFGR#5~MFGR#55~MFGR#5512~peach~MEDIUM BURNISHED BRASS~49~LG BOX
azure cream~199999~MFGR#5~MFGR#52~MFGR#5235~medium~PROMO PLATED BRASS~24~LG CASE
light midnight~200000~MFGR#5~MFGR#52~MFGR#5223~firebrick~MEDIUM ANODIZED TIN~22~LG CAN
[ec2-user@ip-172-31-12-205 ~]$
```

MapReduce with HadoopStreaming



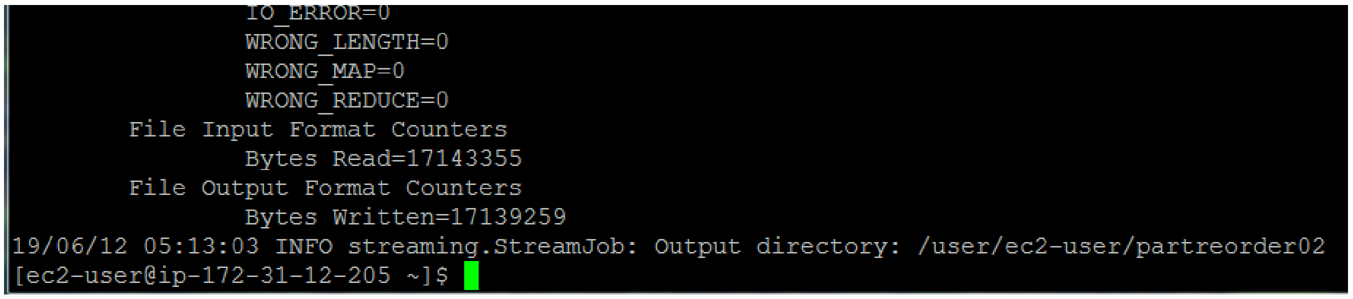
```
ec2-user@ip-172-31-12-205:~
  GNU nano 2.9.8                              part1Mapper.py

#!/usr/bin/python
import sys

reorder = []

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

cat part.tbl | python part1Mapper.py



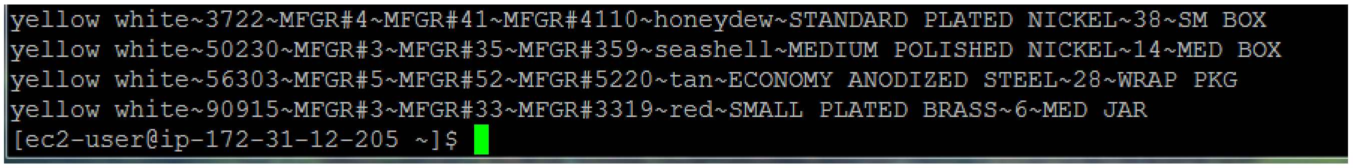
```
steel cyan~199996~MFGR#4~MFGR#44~MFGR#4418~chocolate~PROMO PLATED COPPER~11~
azure snow~199997~MFGR#4~MFGR#44~MFGR#4426~drab~PROMO PLATED NICKEL~37~
misty plum~199998~MFGR#5~MFGR#55~MFGR#5512~peach~MEDIUM BURNISHED BRASS~49~
azure cream~199999~MFGR#5~MFGR#52~MFGR#5235~medium~PROMO PLATED BRASS~24~
light midnight~200000~MFGR#5~MFGR#52~MFGR#5223~firebrick~MEDIUM ANODIZED TIN~22~
[ec2-user@ip-172-31-12-205 ~]$
```

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -input /user/ec2-user/part.tbl -output /user/ec2-user/partreorder02 -mapper part1Mapper.py -file /home/ec2-user/part1Mapper.py



```
                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
19/06/12 05:13:03 INFO streaming.StreamJob: Output directory: /user/ec2-user/partreorder02
[ec2-user@ip-172-31-12-205 ~]$
```

hadoop fs -cat /user/ec2-user/partreorder02/part-00000



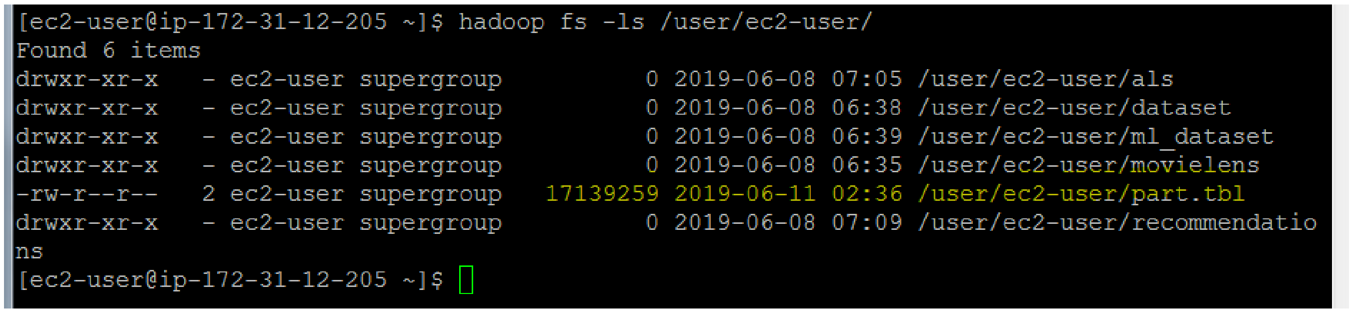
```
yellow white~3722~MFGR#4~MFGR#41~MFGR#4110~honeydew~STANDARD PLATED NICKEL~38~SM BOX
yellow white~50230~MFGR#3~MFGR#35~MFGR#359~seashell~MEDIUM POLISHED NICKEL~14~MED BOX
yellow white~56303~MFGR#5~MFGR#52~MFGR#5220~tan~ECONOMY ANODIZED STEEL~28~WRAP PKG
yellow white~90915~MFGR#3~MFGR#33~MFGR#3319~red~SMALL PLATED BRASS~6~MED JAR
[ec2-user@ip-172-31-12-205 ~]$
```

Pig

cd $PIG_HOME

bin/pig

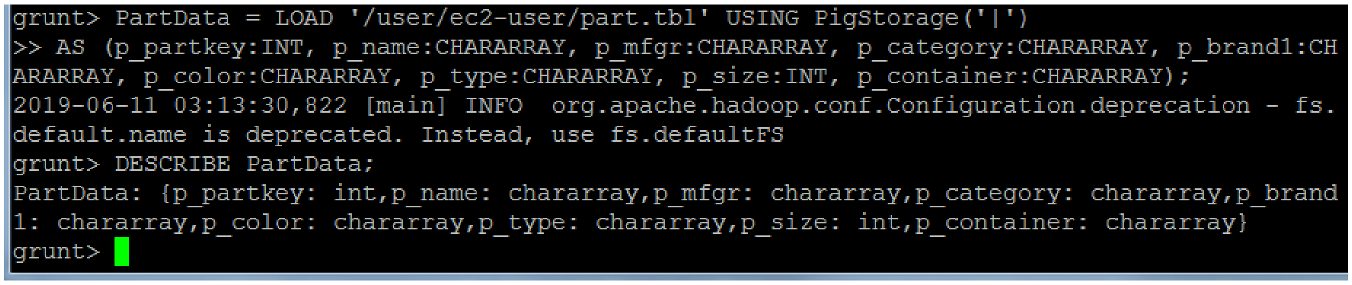
hadoop fs -put part.tbl /user/ec2-user/



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls /user/ec2-user/
Found 6 items
drwxr-xr-x   - ec2-user supergroup          0 2019-06-08 07:05 /user/ec2-user/als
drwxr-xr-x   - ec2-user supergroup          0 2019-06-08 06:38 /user/ec2-user/dataset
drwxr-xr-x   - ec2-user supergroup          0 2019-06-08 06:39 /user/ec2-user/ml_dataset
drwxr-xr-x   - ec2-user supergroup          0 2019-06-08 06:35 /user/ec2-user/movielens
-rw-r--r--   2 ec2-user supergroup   17139259 2019-06-11 02:36 /user/ec2-user/part.tbl
drwxr-xr-x   - ec2-user supergroup          0 2019-06-08 07:09 /user/ec2-user/recommendatio
ns
[ec2-user@ip-172-31-12-205 ~]$
```

PartData = LOAD '/user/ec2-user/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);

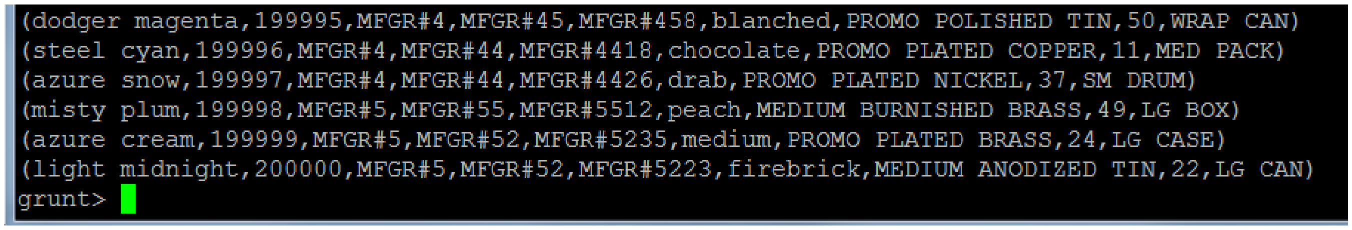
DESCRIBE PartData;



```
grunt> PartData = LOAD '/user/ec2-user/part.tbl' USING PigStorage('|')
>> AS (p_partkey:INT, p_name:CHARARRAY, p_mfgr:CHARARRAY, p_category:CHARARRAY, p_brand1:CH
ARARRAY, p_color:CHARARRAY, p_type:CHARARRAY, p_size:INT, p_container:CHARARRAY);
2019-06-11 03:13:30,822 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.
default.name is deprecated. Instead, use fs.defaultFS
grunt> DESCRIBE PartData;
PartData: {p_partkey: int,p_name: chararray,p_mfgr: chararray,p_category: chararray,p_brand
1: chararray,p_color: chararray,p_type: chararray,p_size: int,p_container: chararray}
grunt>
```

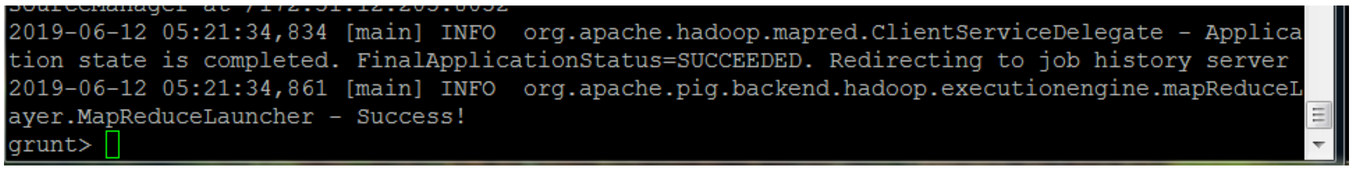
PartDataSwitched = FOREACH PartData GENERATE p_name, p_partkey, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_container;

DUMP PartDataSwitched;



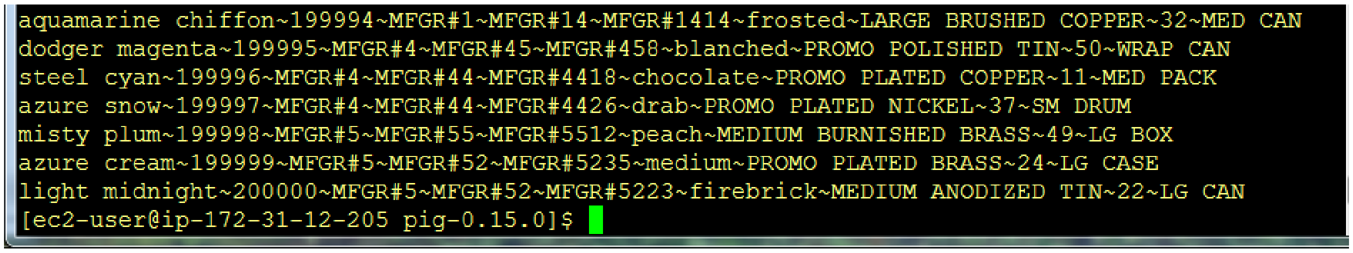
```
(dodger magenta,199995,MFGR#4,MFGR#45,MFGR#458,blanched,PROMO POLISHED TIN,50,WRAP CAN)
(steel cyan,199996,MFGR#4,MFGR#44,MFGR#4418,chocolate,PROMO PLATED COPPER,11,MED PACK)
(azure snow,199997,MFGR#4,MFGR#44,MFGR#4426,drab,PROMO PLATED NICKEL,37,SM DRUM)
(misty plum,199998,MFGR#5,MFGR#55,MFGR#5512,peach,MEDIUM BURNISHED BRASS,49,LG BOX)
(azure cream,199999,MFGR#5,MFGR#52,MFGR#5235,medium,PROMO PLATED BRASS,24,LG CASE)
(light midnight,200000,MFGR#5,MFGR#52,MFGR#5223,firebrick,MEDIUM ANODIZED TIN,22,LG CAN)
grunt>
```

STORE PartDataSwitched INTO 'PartDataNewSwitched2' USING PigStorage('~');



```
2019-06-12 05:21:34,834 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Applica
tion state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2019-06-12 05:21:34,861 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceL
ayer.MapReduceLauncher - Success!
grunt>
```

hadoop fs -cat /user/ec2-user/PartDataNewSwitched2/part-m-00000



```
aquamarine chiffon~199994~MFGR#1~MFGR#14~MFGR#1414~frosted~LARGE BRUSHED COPPER~32~MED CAN
dodger magenta~199995~MFGR#4~MFGR#45~MFGR#458~blanched~PROMO POLISHED TIN~50~WRAP CAN
steel cyan~199996~MFGR#4~MFGR#44~MFGR#4418~chocolate~PROMO PLATED COPPER~11~MED PACK
azure snow~199997~MFGR#4~MFGR#44~MFGR#4426~drab~PROMO PLATED NICKEL~37~SM DRUM
misty plum~199998~MFGR#5~MFGR#55~MFGR#5512~peach~MEDIUM BURNISHED BRASS~49~LG BOX
azure cream~199999~MFGR#5~MFGR#52~MFGR#5235~medium~PROMO PLATED BRASS~24~LG CASE
light midnight~200000~MFGR#5~MFGR#52~MFGR#5223~firebrick~MEDIUM ANODIZED TIN~22~LG CAN
[ec2-user@ip-172-31-12-205 pig-0.15.0]$
```

Part 2: Querying (25 pts)

Implement the following query:

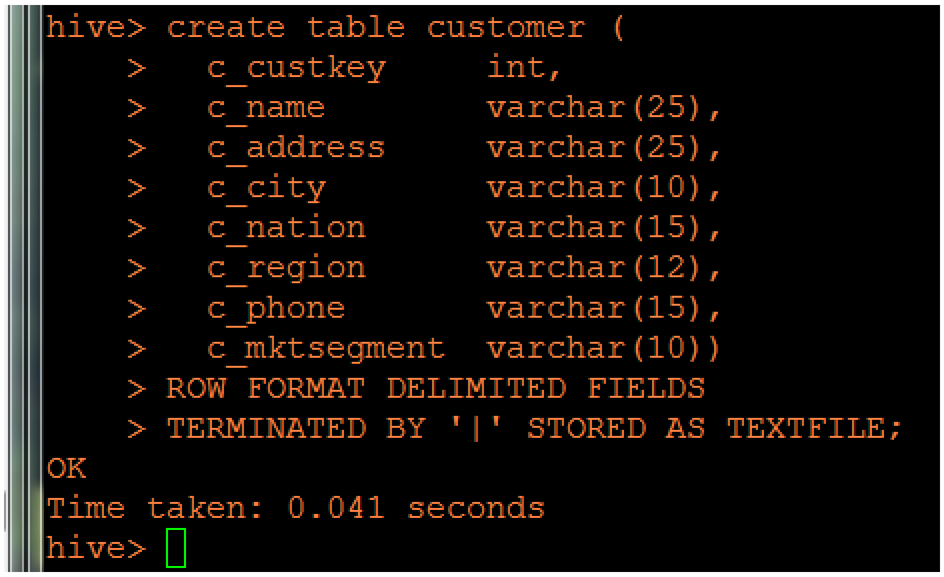
```
select c_nation, sum(lo_revenue)
from customer, lineorder
where lo_custkey = c_custkey
  and c_region = 'AMERICA'
  and lo_discount BETWEEN 4 and 6
group by c_nation;
```

using Hive, MapReduce with HadoopStreaming and Pig (i.e. 3 different solutions). I 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

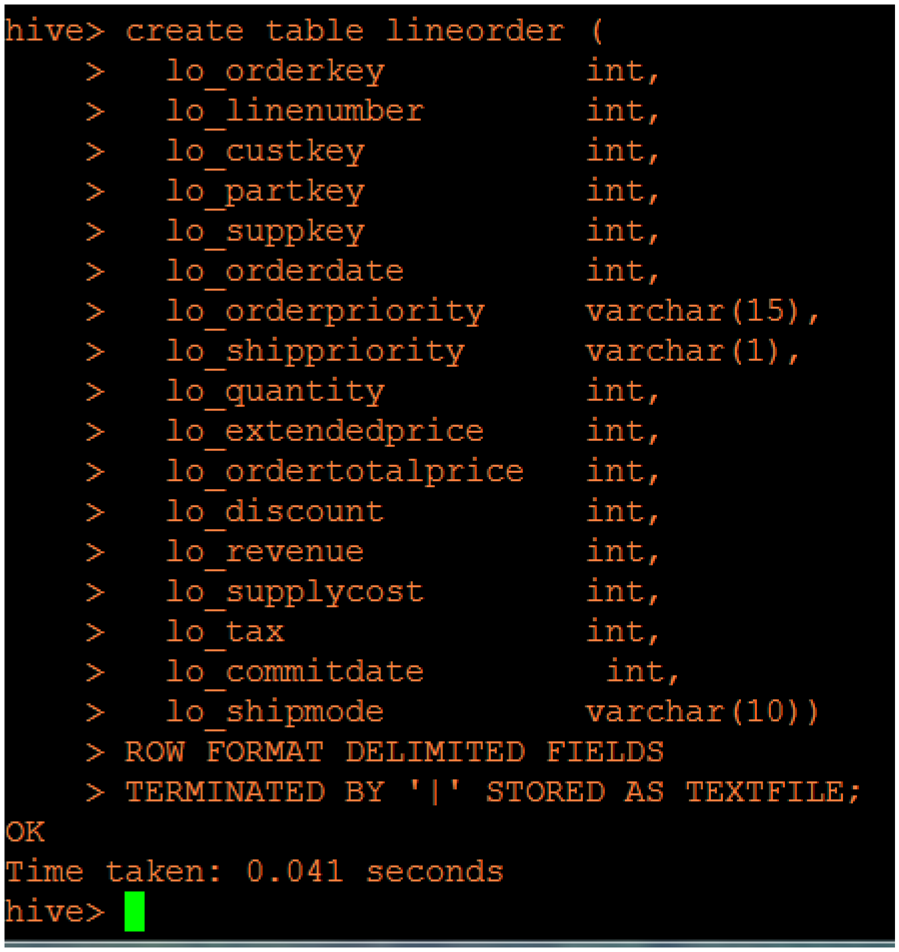
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl

```
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;
```



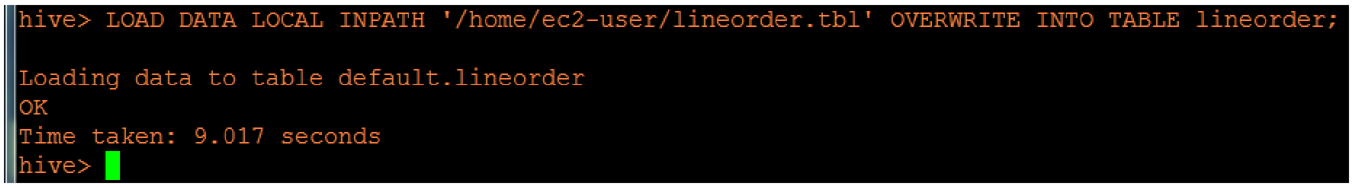
```
hive> create table customer (
    >   c_custkey     int,
    >   c_name        varchar(25),
    >   c_address     varchar(25),
    >   c_city        varchar(10),
    >   c_nation      varchar(15),
    >   c_region      varchar(12),
    >   c_phone       varchar(15),
    >   c_mktsegment  varchar(10))
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '|' STORED AS TEXTFILE;
OK
Time taken: 0.041 seconds
hive>
```

```
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;
```



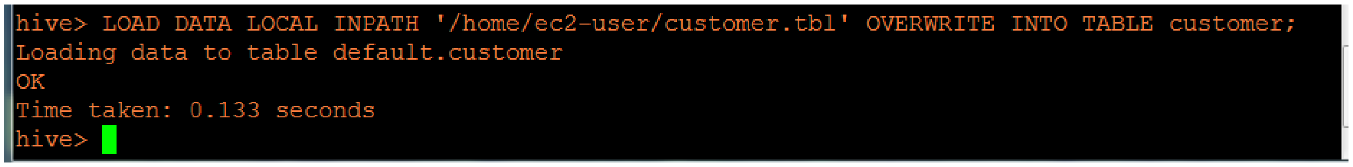
```
hive> create table lineorder (
    >   lo_orderkey          int,
    >   lo_linenumber        int,
    >   lo_custkey           int,
    >   lo_partkey           int,
    >   lo_suppkey           int,
    >   lo_orderdate         int,
    >   lo_orderpriority     varchar(15),
    >   lo_shippriority      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;
OK
Time taken: 0.041 seconds
hive>
```

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



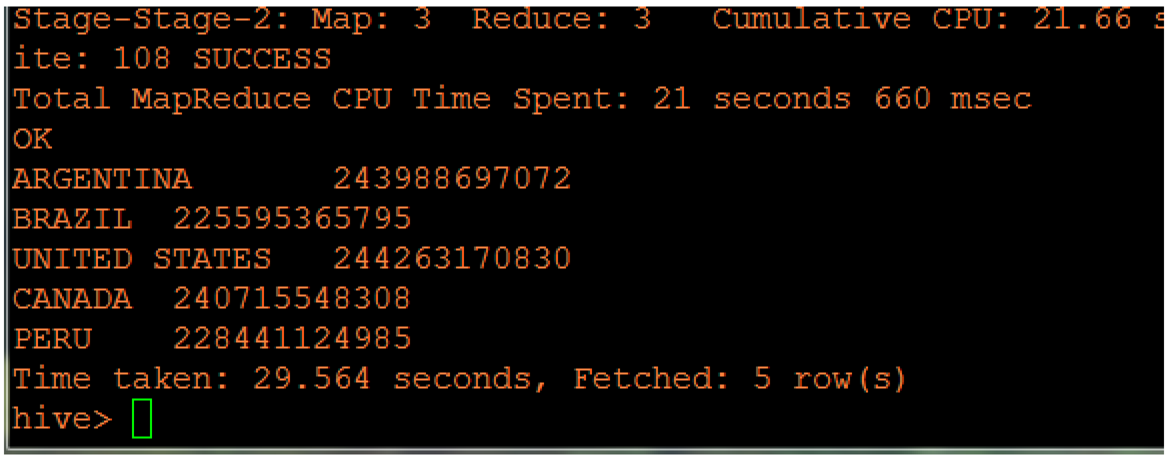
```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl' OVERWRITE INTO TABLE lineorder;

Loading data to table default.lineorder
OK
Time taken: 9.017 seconds
hive>
```

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



```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl' OVERWRITE INTO TABLE customer;
Loading data to table default.customer
OK
Time taken: 0.133 seconds
hive>
```

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;



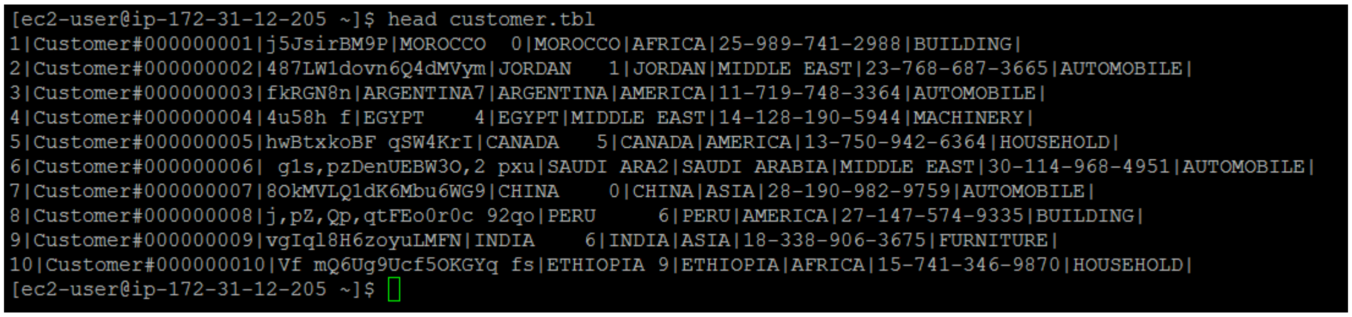
```
Stage-Stage-2: Map: 3  Reduce: 3   Cumulative CPU: 21.66 s
ite: 108 SUCCESS
Total MapReduce CPU Time Spent: 21 seconds 660 msec
OK
ARGENTINA       243988697072
BRAZIL  225595365795
UNITED STATES   244263170830
CANADA  240715548308
PERU    228441124985
Time taken: 29.564 seconds, Fetched: 5 row(s)
hive>
```

MapReduce with HadoopStreaming

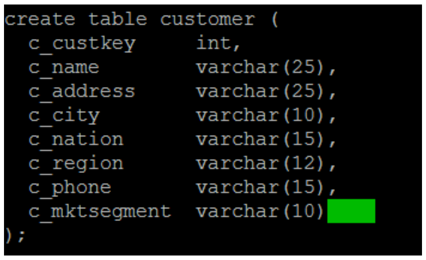
select c_nation, sum(lo_revenue)
from customer, lineorder
where lo_custkey = c_custkey
and c_region = 'AMERICA'
and lo_discount BETWEEN 4 and 6
group by c_nation;

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl

head customer.tbl

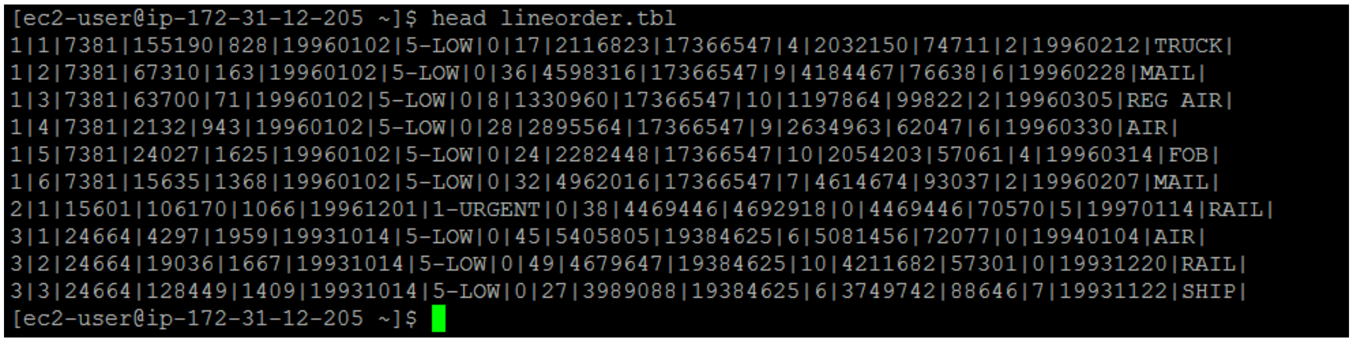


```
[ec2-user@ip-172-31-12-205 ~]$ head customer.tbl
1|Customer#000000001|j5JsirBM9P|MOROCCO  0|MOROCCO|AFRICA|25-989-741-2988|BUILDING|
2|Customer#000000002|487LW1dovn6Q4dMVym|JORDAN   1|JORDAN|MIDDLE EAST|23-768-687-3665|AUTOMOBILE|
3|Customer#000000003|fkRGN8n|ARGENTINA7|ARGENTINA|AMERICA|11-719-748-3364|AUTOMOBILE|
4|Customer#000000004|4u58h f|EGYPT    4|EGYPT|MIDDLE EAST|14-128-190-5944|MACHINERY|
5|Customer#000000005|hwBtxkoBF qSW4KrI|CANADA   5|CANADA|AMERICA|13-750-942-6364|HOUSEHOLD|
6|Customer#000000006| g1s,pzDenUEBW3O,2 pxu|SAUDI ARA2|SAUDI ARABIA|MIDDLE EAST|30-114-968-4951|AUTOMOBILE|
7|Customer#000000007|8OkMVLQ1dK6Mbu6WG9|CHINA    0|CHINA|ASIA|28-190-982-9759|AUTOMOBILE|
8|Customer#000000008|j,pZ,Qp,qtFEo0r0c 92qo|PERU     6|PERU|AMERICA|27-147-574-9335|BUILDING|
9|Customer#000000009|vgIql8H6zoyuLMFN|INDIA    6|INDIA|ASIA|18-338-906-3675|FURNITURE|
10|Customer#000000010|Vf mQ6Ug9Ucf5OKGYq fs|ETHIOPIA 9|ETHIOPIA|AFRICA|15-741-346-9870|HOUSEHOLD|
[ec2-user@ip-172-31-12-205 ~]$
```

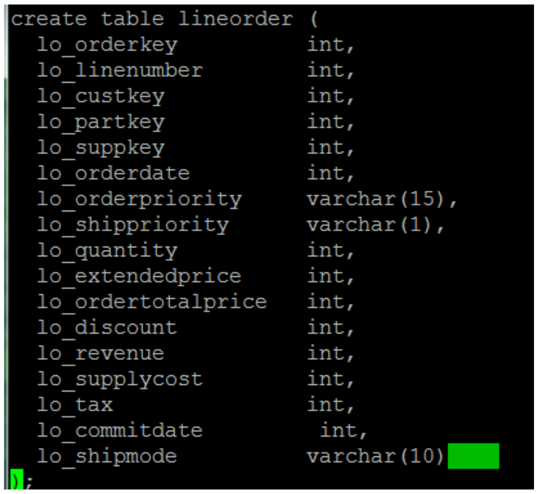


```
create table customer (
  c_custkey     int,
  c_name        varchar(25),
  c_address     varchar(25),
  c_city        varchar(10),
  c_nation      varchar(15),
  c_region      varchar(12),
  c_phone       varchar(15),
  c_mktsegment  varchar(10)
);
```

head lineorder.tbl

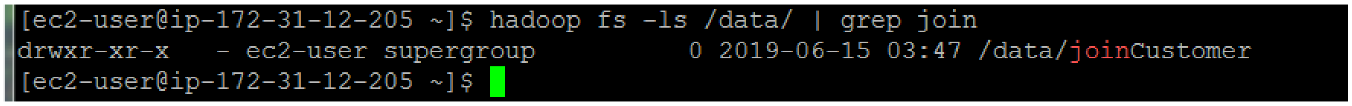


```
[ec2-user@ip-172-31-12-205 ~]$ head lineorder.tbl
1|1|7381|155190|828|19960102|5-LOW|0|17|2116823|17366547|4|2032150|74711|2|19960212|TRUCK|
1|2|7381|67310|163|19960102|5-LOW|0|36|4598316|17366547|9|4184467|76638|6|19960228|MAIL|
1|3|7381|63700|71|19960102|5-LOW|0|8|1330960|17366547|10|1197864|99822|2|19960305|REG AIR|
1|4|7381|2132|943|19960102|5-LOW|0|28|2895564|17366547|9|2634963|62047|6|19960330|AIR|
1|5|7381|24027|1625|19960102|5-LOW|0|24|2282448|17366547|10|2054203|57061|4|19960314|FOB|
1|6|7381|15635|1368|19960102|5-LOW|0|32|4962016|17366547|7|4614674|93037|2|19960207|MAIL|
2|1|15601|106170|1066|19961201|1-URGENT|0|38|4469446|4692918|0|4469446|70570|5|19970114|RAIL|
3|1|24664|4297|1959|19931014|5-LOW|0|45|5405805|19384625|6|5081456|72077|0|19940104|AIR|
3|2|24664|19036|1667|19931014|5-LOW|0|49|4679647|19384625|10|4211682|57301|0|19931220|RAIL|
3|3|24664|128449|1409|19931014|5-LOW|0|27|3989088|19384625|6|3749742|88646|7|19931122|SHIP|
[ec2-user@ip-172-31-12-205 ~]$
```



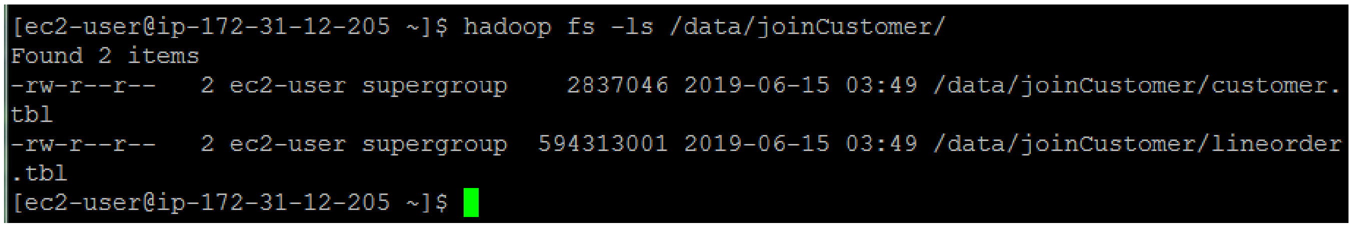
```
create table lineorder (
  lo_orderkey          int,
  lo_linenumber        int,
  lo_custkey           int,
  lo_partkey           int,
  lo_suppkey           int,
  lo_orderdate         int,
  lo_orderpriority     varchar(15),
  lo_shippriority      varchar(1),
  lo_quantity          int,
  lo_extendedprice     int,
  lo_ordertotalprice   int,
  lo_discount          int,
  lo_revenue           int,
  lo_supplycost        int,
  lo_tax               int,
  lo_commitdate         int,
  lo_shipmode          varchar(10)
);
```

hadoop fs -mkdir -p /data/joinCustomer



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls /data/ | grep join
drwxr-xr-x   - ec2-user supergroup          0 2019-06-15 03:47 /data/joinCustomer
[ec2-user@ip-172-31-12-205 ~]$
```

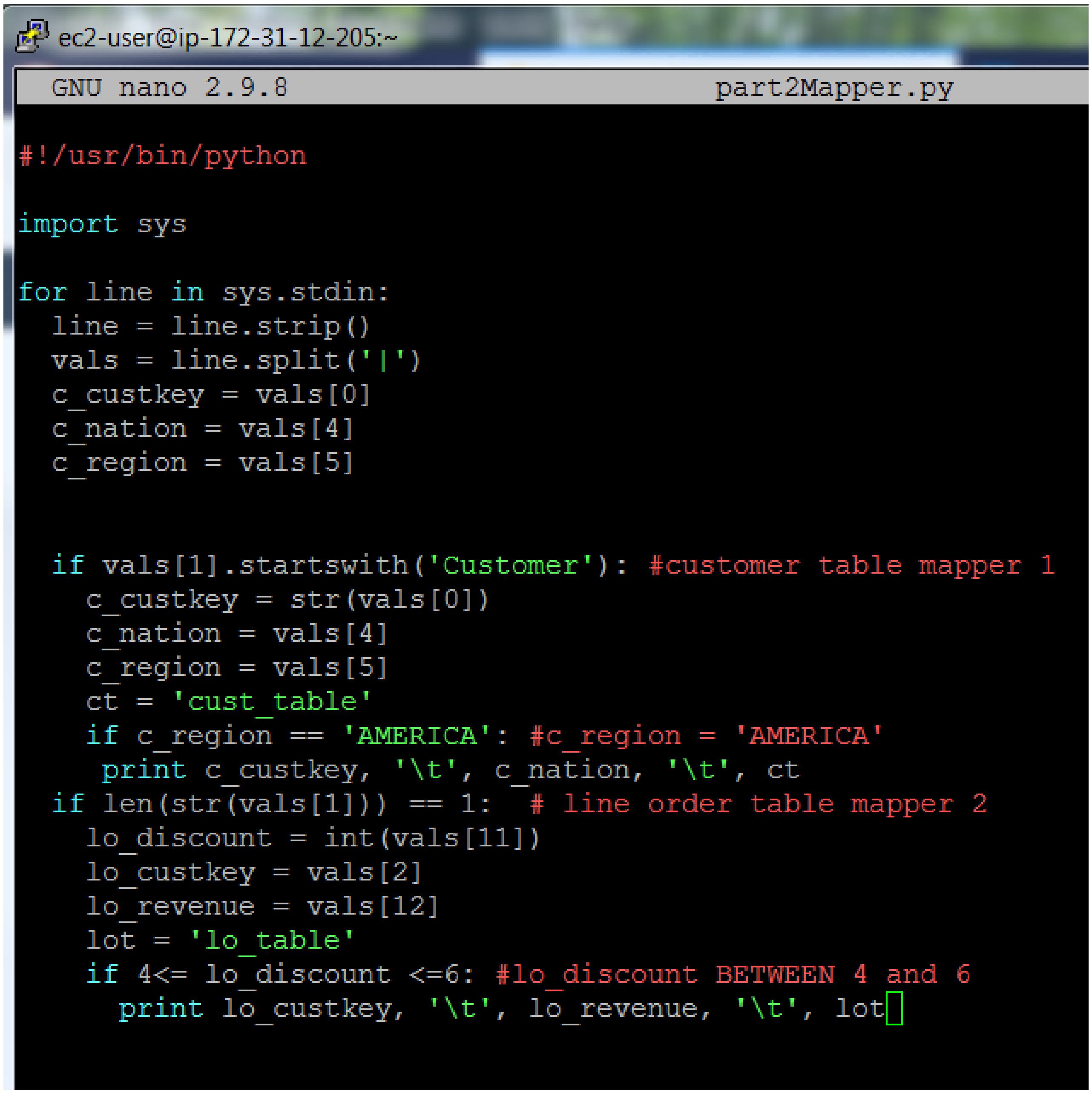
hadoop fs -put customer.tbl lineorder.tbl /data/joinCustomer



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls /data/joinCustomer/
Found 2 items
-rw-r--r--   2 ec2-user supergroup    2837046 2019-06-15 03:49 /data/joinCustomer/customer.
tbl
-rw-r--r--   2 ec2-user supergroup  594313001 2019-06-15 03:49 /data/joinCustomer/lineorder
.tbl
[ec2-user@ip-172-31-12-205 ~]$
```

Pass 1

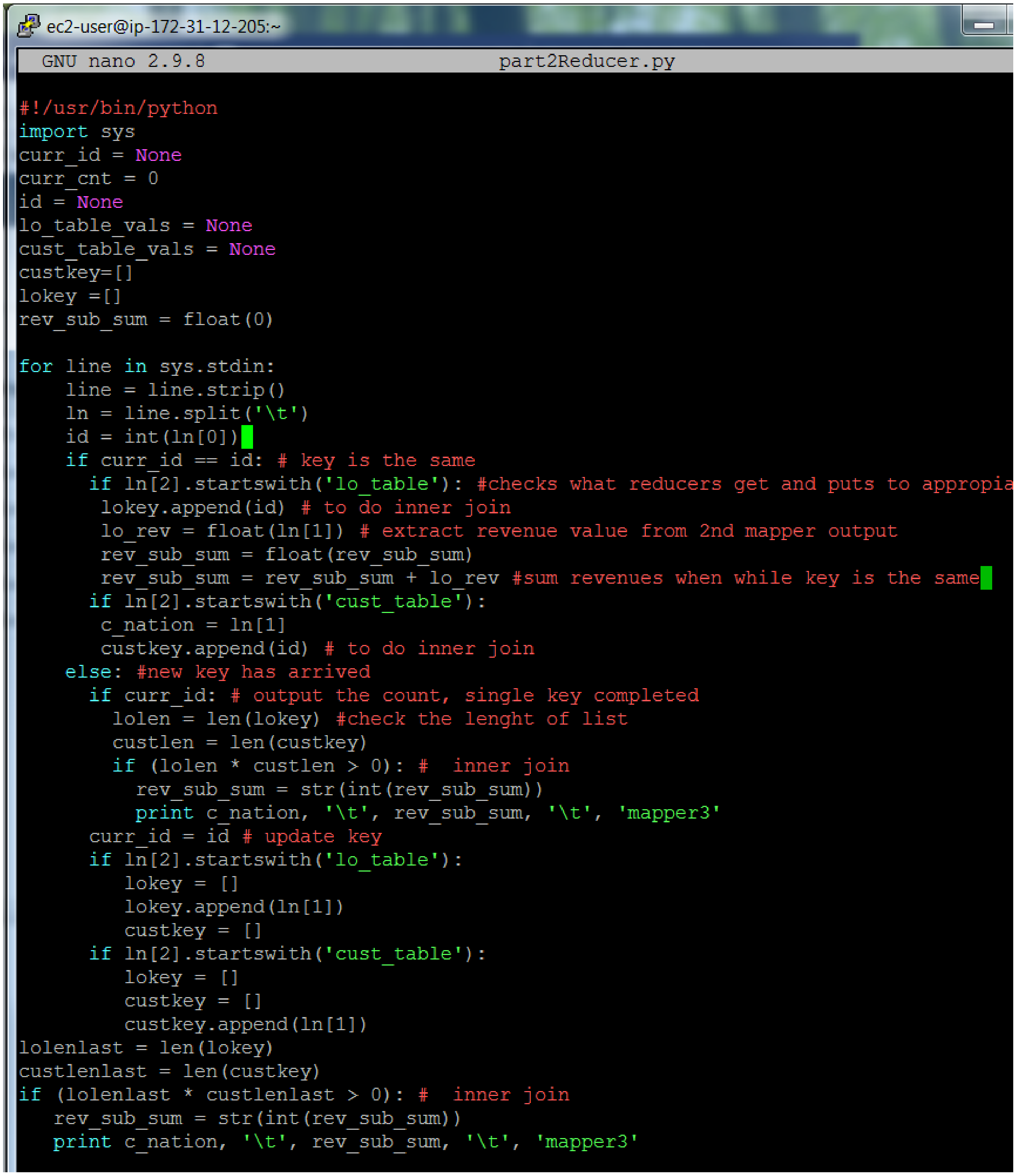
part2Mapper.py



ec2-user@ip-172-31-12-205:~

```
 GNU nano 2.9.8                          part2Mapper.py

#!/usr/bin/python

import sys

for line in sys.stdin:
  line = line.strip()
  vals = line.split('|')
  c_custkey = vals[0]
  c_nation = vals[4]
  c_region = vals[5]


  if vals[1].startswith('Customer'): #customer table mapper 1
    c_custkey = str(vals[0])
    c_nation = vals[4]
    c_region = vals[5]
    ct = 'cust_table'
    if c_region == 'AMERICA': #c_region = 'AMERICA'
     print c_custkey, '\t', c_nation, '\t', ct
  if len(str(vals[1])) == 1:  # line order table mapper 2
    lo_discount = int(vals[11])
    lo_custkey = vals[2]
    lo_revenue = vals[12]
    lot = 'lo_table'
    if 4<= lo_discount <=6: #lo_discount BETWEEN 4 and 6
      print lo_custkey, '\t', lo_revenue, '\t', lot
```

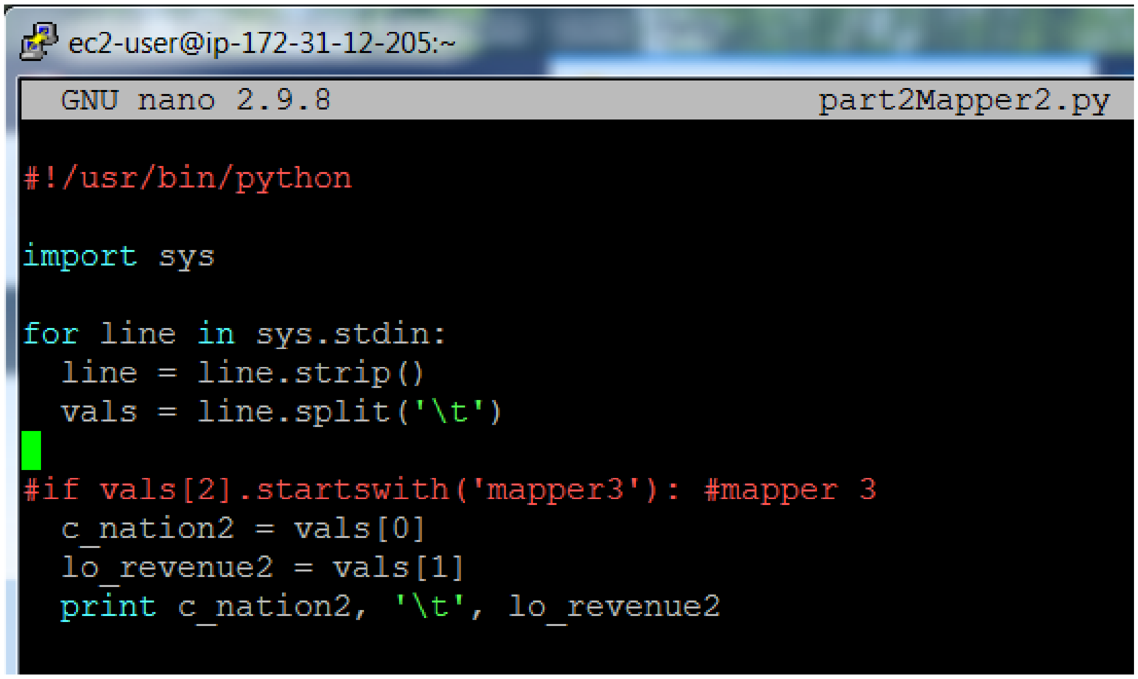
part2Reducer.py



```
ec2-user@ip-172-31-12-205:~
  GNU nano 2.9.8                              part2Reducer.py

#!/usr/bin/python
import sys
curr_id = None
curr_cnt = 0
id = None
lo_table_vals = None
cust_table_vals = None
custkey=[]
lokey =[]
rev_sub_sum = float(0)

for line in sys.stdin:
    line = line.strip()
    ln = line.split('\t')
    id = int(ln[0])
    if curr_id == id: # key is the same
      if ln[2].startswith('lo_table'): #checks what reducers get and puts to appropia
       lokey.append(id) # to do inner join
       lo_rev = float(ln[1]) # extract revenue value from 2nd mapper output
       rev_sub_sum = float(rev_sub_sum)
       rev_sub_sum = rev_sub_sum + lo_rev #sum revenues when while key is the same
      if ln[2].startswith('cust_table'):
       c_nation = ln[1]
       custkey.append(id) # to do inner join
    else: #new key has arrived
      if curr_id: # output the count, single key completed
        lolen = len(lokey) #check the lenght of list
        custlen = len(custkey)
        if (lolen * custlen > 0): #  inner join
          rev_sub_sum = str(int(rev_sub_sum))
          print c_nation, '\t', rev_sub_sum, '\t', 'mapper3'
      curr_id = id # update key
      if ln[2].startswith('lo_table'):
         lokey = []
         lokey.append(ln[1])
         custkey = []
      if ln[2].startswith('cust_table'):
         lokey = []
         custkey = []
         custkey.append(ln[1])
lolenlast = len(lokey)
custlenlast = len(custkey)
if (lolenlast * custlenlast > 0): #  inner join
   rev_sub_sum = str(int(rev_sub_sum))
   print c_nation, '\t', rev_sub_sum, '\t', 'mapper3'
```

Pass 2

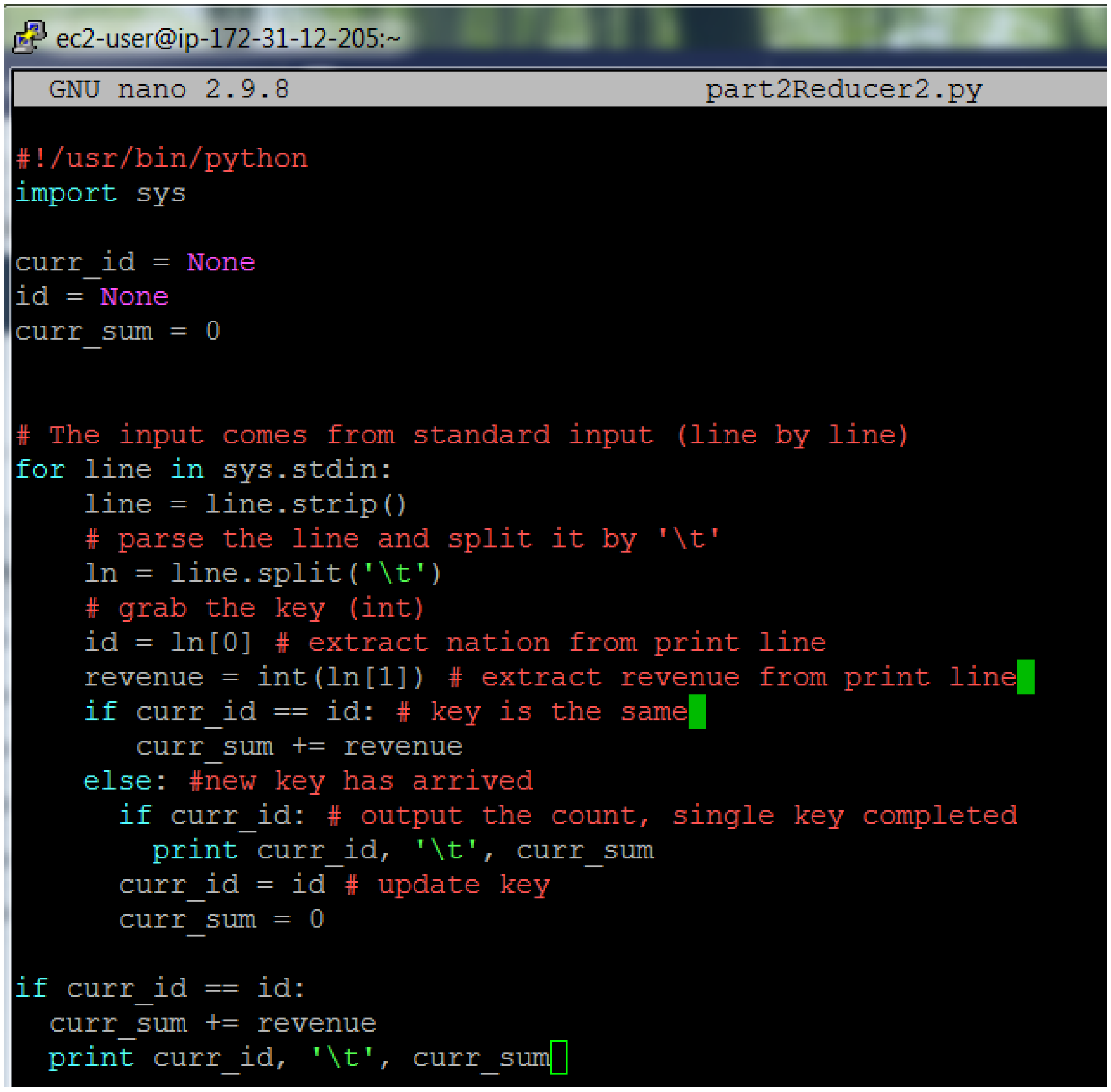
part2Mapper2.py



```
#!/usr/bin/python

import sys

for line in sys.stdin:
  line = line.strip()
  vals = line.split('\t')

#if vals[2].startswith('mapper3'): #mapper 3
  c_nation2 = vals[0]
  lo_revenue2 = vals[1]
  print c_nation2, '\t', lo_revenue2
```

part2Reducer2.py

part2Reducer2.py

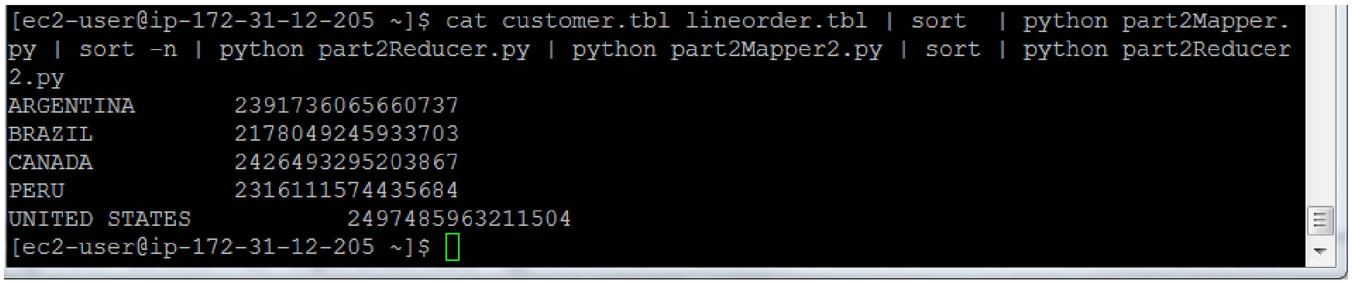


ec2-user@ip-172-31-12-205:~

```
  GNU nano 2.9.8                        part2Reducer2.py

#!/usr/bin/python
import sys

curr_id = None
id = None
curr_sum = 0


# The input comes from standard input (line by line)
for line in sys.stdin:
    line = line.strip()
    # parse the line and split it by '\t'
    ln = line.split('\t')
    # grab the key (int)
    id = ln[0] # extract nation from print line
    revenue = int(ln[1]) # extract revenue from print line
    if curr_id == id: # key is the same
       curr_sum += revenue
    else: #new key has arrived
      if curr_id: # output the count, single key completed
        print curr_id, '\t', curr_sum
      curr_id = id # update key
      curr_sum = 0

if curr_id == id:
  curr_sum += revenue
  print curr_id, '\t', curr_sum
```

<u>All 4 python files (part2Mapper.py, part2Reducer.py, part2Mapper2.py, part2Reducer2.py) run without errors in command line using below command.</u>

cat customer.tbl lineorder.tbl | sort | python part2Mapper.py | sort -n | python part2Reducer.py | python part2Mapper2.py | sort | python part2Reducer2.py

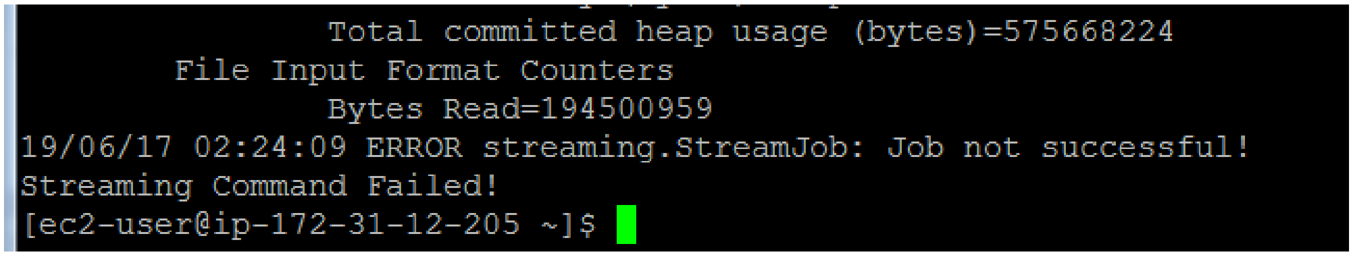


```
[ec2-user@ip-172-31-12-205 ~]$ cat customer.tbl lineorder.tbl | sort  | python part2Mapper.
py | sort -n | python part2Reducer.py | python part2Mapper2.py | sort | python part2Reducer
2.py
ARGENTINA       2391736065660737
BRAZIL          2178049245933703
CANADA          2426493295203867
PERU            2316111574435684
UNITED STATES           2497485963211504
[ec2-user@ip-172-31-12-205 ~]$
```

However their fail when running first pass hadoop jar command. If you see why, please let me know.

PASS 1 command

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -D mapred.text.key.comparator.options=-n -input /data/joinCustomer -output /data/outputCustomer007 -mapper part2Mapper.py -file /home/ec2-user/part2Mapper.py -reducer par2Reducer.py -file /home/ec2-user/part2Reducer.py



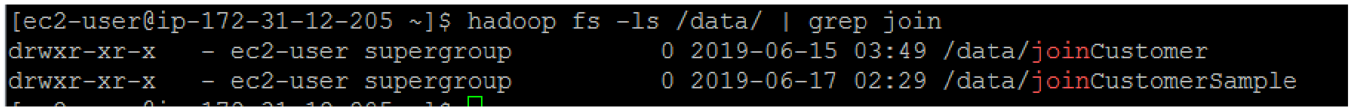
```
                Total committed heap usage (bytes)=575668224
        File Input Format Counters
                Bytes Read=194500959
19/06/17 02:24:09 ERROR streaming.StreamJob: Job not successful!
Streaming Command Failed!
[ec2-user@ip-172-31-12-205 ~]$
```

PASS 2 command would be:

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -input /data/outputCustomer007/part-00000 -output /data/outputCustomerPass2 -mapper part2Mapper2.py -file /home/ec2-user/part2Mapper2.py -reducer par2Reducer2.py -file /home/ec2-user/part2Reducer2.py

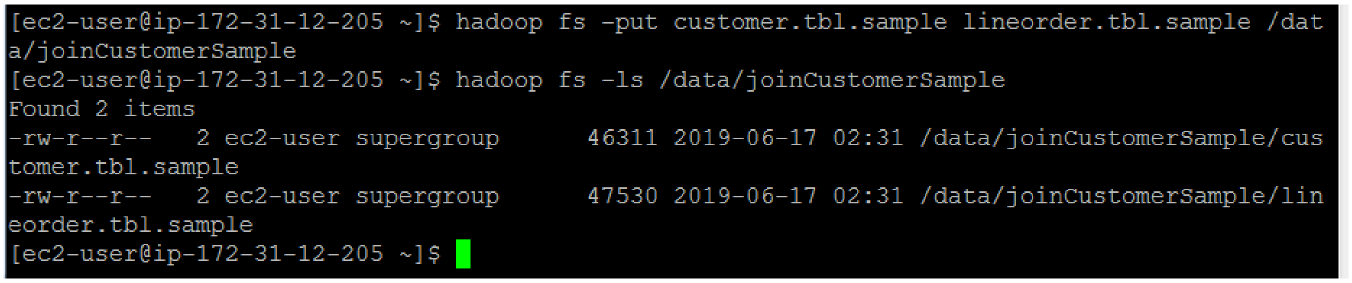
Testing on smaller samples

hadoop fs -mkdir -p /data/joinCustomerSample



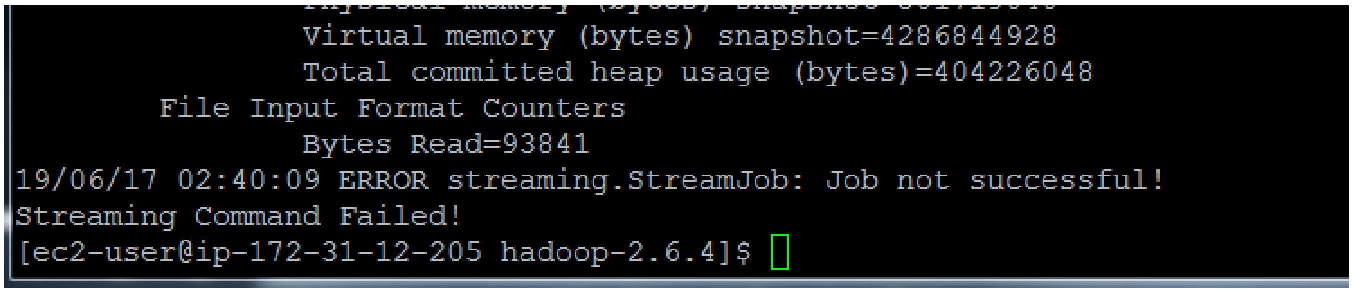
```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls /data/ | grep join
drwxr-xr-x   - ec2-user supergroup          0 2019-06-15 03:49 /data/joinCustomer
drwxr-xr-x   - ec2-user supergroup          0 2019-06-17 02:29 /data/joinCustomerSample
```

hadoop fs -put customer.tbl.sample lineorder.tbl.sample /data/joinCustomerSample



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -put customer.tbl.sample lineorder.tbl.sample /dat
a/joinCustomerSample
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls /data/joinCustomerSample
Found 2 items
-rw-r--r--   2 ec2-user supergroup      46311 2019-06-17 02:31 /data/joinCustomerSample/cus
tomer.tbl.sample
-rw-r--r--   2 ec2-user supergroup      47530 2019-06-17 02:31 /data/joinCustomerSample/lin
eorder.tbl.sample
[ec2-user@ip-172-31-12-205 ~]$ 
```

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -D mapred.text.key.comparator.options=-n -input /data/joinCustomerSample -output /data/outputCustomerSample002 -mapper part2Mapper.py -file /home/ec2-user/part2Mapper.py -reducer par2Reducer.py -file /home/ec2-user/part2Reducer.py



```
                Virtual memory (bytes) snapshot=4286844928
                Total committed heap usage (bytes)=404226048
        File Input Format Counters
                Bytes Read=93841
19/06/17 02:40:09 ERROR streaming.StreamJob: Job not successful!
Streaming Command Failed!
[ec2-user@ip-172-31-12-205 hadoop-2.6.4]$ 
```

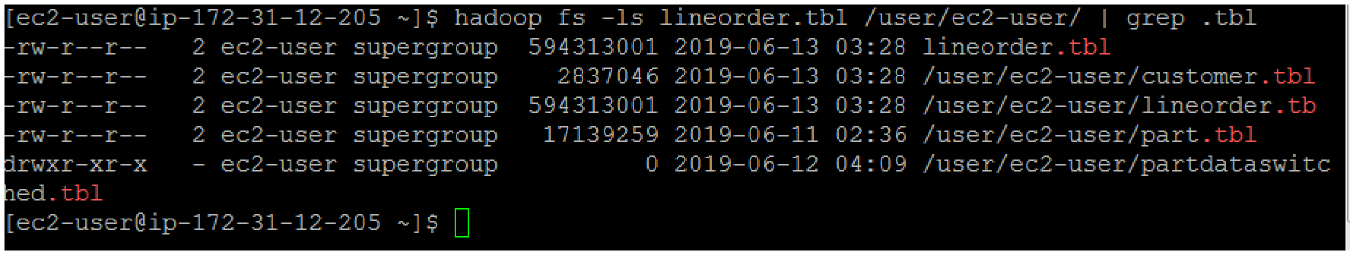
Pig

cd $PIG_HOME

bin/pig

hadoop fs -put customer.tbl /user/ec2-user/

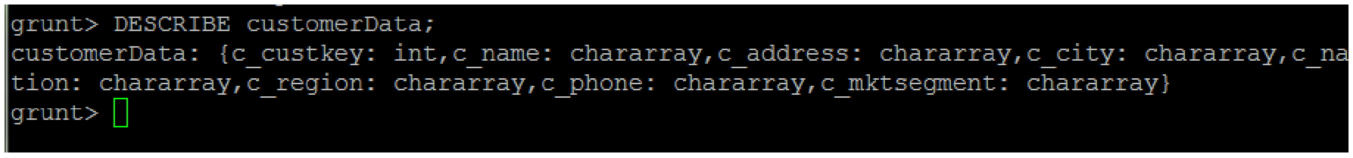
hadoop fs -put lineorder.tbl /user/ec2-user/



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls lineorder.tbl /user/ec2-user/ | grep .tbl
-rw-r--r--   2 ec2-user supergroup  594313001 2019-06-13 03:28 lineorder.tbl
-rw-r--r--   2 ec2-user supergroup    2837046 2019-06-13 03:28 /user/ec2-user/customer.tbl
-rw-r--r--   2 ec2-user supergroup  594313001 2019-06-13 03:28 /user/ec2-user/lineorder.tb
-rw-r--r--   2 ec2-user supergroup   17139259 2019-06-11 02:36 /user/ec2-user/part.tbl
drwxr-xr-x   - ec2-user supergroup          0 2019-06-12 04:09 /user/ec2-user/partdataswitc
hed.tbl
[ec2-user@ip-172-31-12-205 ~]$
```

**customerData = LOAD '/user/ec2-user/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);**

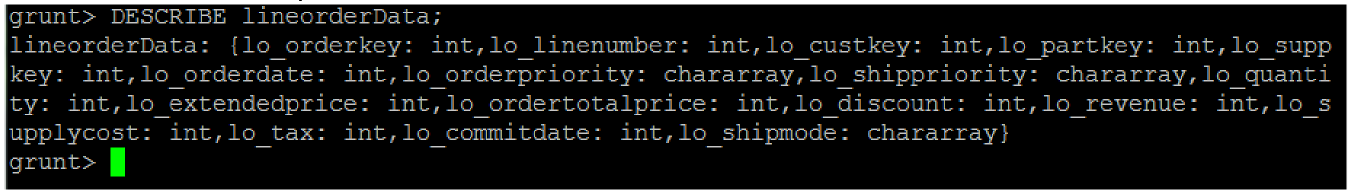
DESCRIBE customerData;



```
grunt> DESCRIBE customerData;
customerData: {c_custkey: int,c_name: chararray,c_address: chararray,c_city: chararray,c_na
tion: chararray,c_region: chararray,c_phone: chararray,c_mktsegment: chararray}
grunt>
```

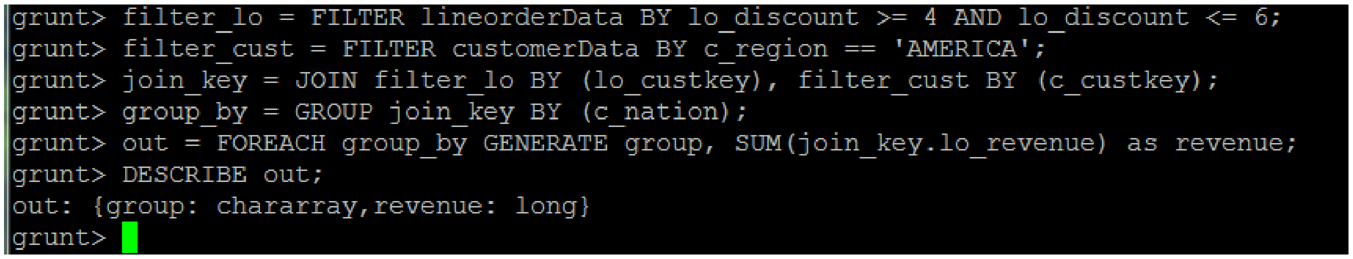
**lineorderData = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS (lo_orderkey:INT, lo_linenumber:INT, lo_custkey:INT, lo_partkey:INT, lo_suppkey:INT, lo_orderdate:INT, lo_orderpriority:CHARARRAY, lo_shippriority:CHARARRAY, lo_quantity:INT, lo_extendedprice:INT, lo_ordertotalprice:INT, lo_discount:INT, lo_revenue:INT, lo_supplycost:INT, lo_tax:INT, lo_commitdate:INT, lo_shipmode: CHARARRAY);**

DESCRIBE lineorderData;



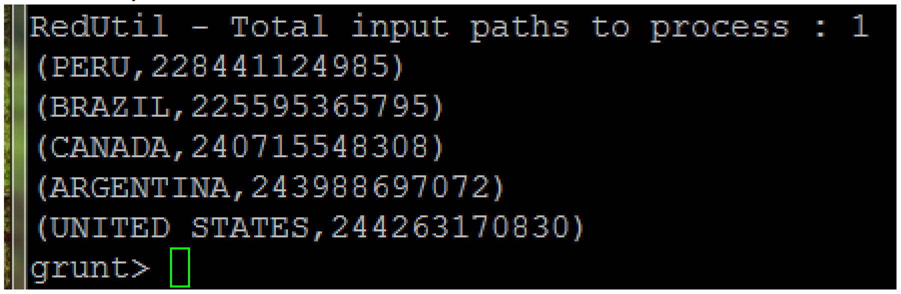
```
grunt> DESCRIBE lineorderData;
lineorderData: {lo_orderkey: int,lo_linenumber: int,lo_custkey: int,lo_partkey: int,lo_supp
key: int,lo_orderdate: int,lo_orderpriority: chararray,lo_shippriority: chararray,lo_quanti
ty: int,lo_extendedprice: int,lo_ordertotalprice: int,lo_discount: int,lo_revenue: int,lo_s
upplycost: int,lo_tax: int,lo_commitdate: int,lo_shipmode: chararray}
grunt>
```

filter_lo = FILTER lineorderData BY lo_discount >= 4 AND lo_discount <= 6;
filter_cust = FILTER customerData BY c_region == 'AMERICA';
join_key = JOIN filter_lo BY (lo_custkey), filter_cust BY (c_custkey);
group_by = GROUP join_key BY (c_nation);
out = FOREACH group_by GENERATE group, SUM(join_key.lo_revenue) as revenue;
DESCRIBE out;



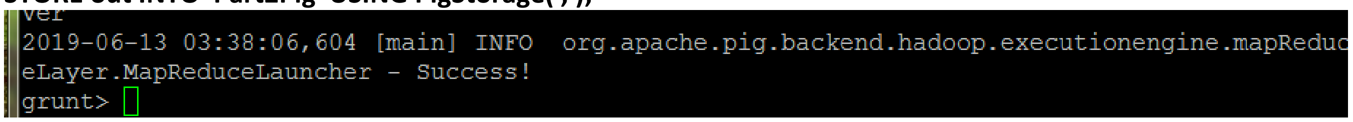
```
grunt> filter_lo = FILTER lineorderData BY lo_discount >= 4 AND lo_discount <= 6;
grunt> filter_cust = FILTER customerData BY c_region == 'AMERICA';
grunt> join_key = JOIN filter_lo BY (lo_custkey), filter_cust BY (c_custkey);
grunt> group_by = GROUP join_key BY (c_nation);
grunt> out = FOREACH group_by GENERATE group, SUM(join_key.lo_revenue) as revenue;
grunt> DESCRIBE out;
out: {group: chararray,revenue: long}
grunt>
```

DUMP out;



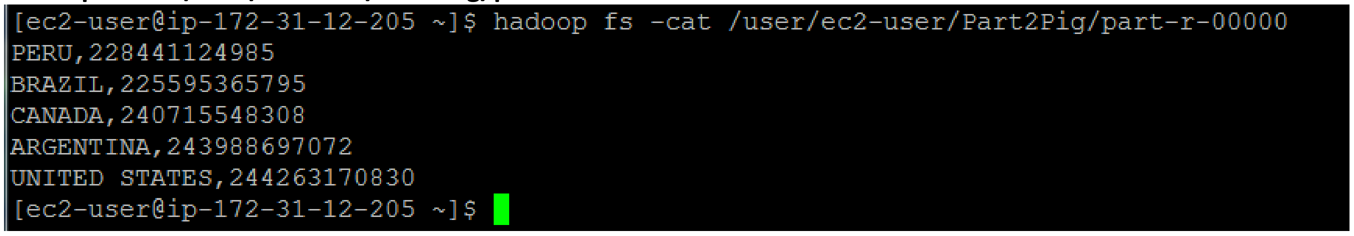
```
RedUtil - Total input paths to process : 1
(PERU,228441124985)
(BRAZIL,225595365795)
(CANADA,240715548308)
(ARGENTINA,243988697072)
(UNITED STATES,244263170830)
grunt>
```

STORE out INTO 'Part2Pig' USING PigStorage(',');



```
ver
2019-06-13 03:38:06,604 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduc
eLayer.MapReduceLauncher - Success!
grunt>
```

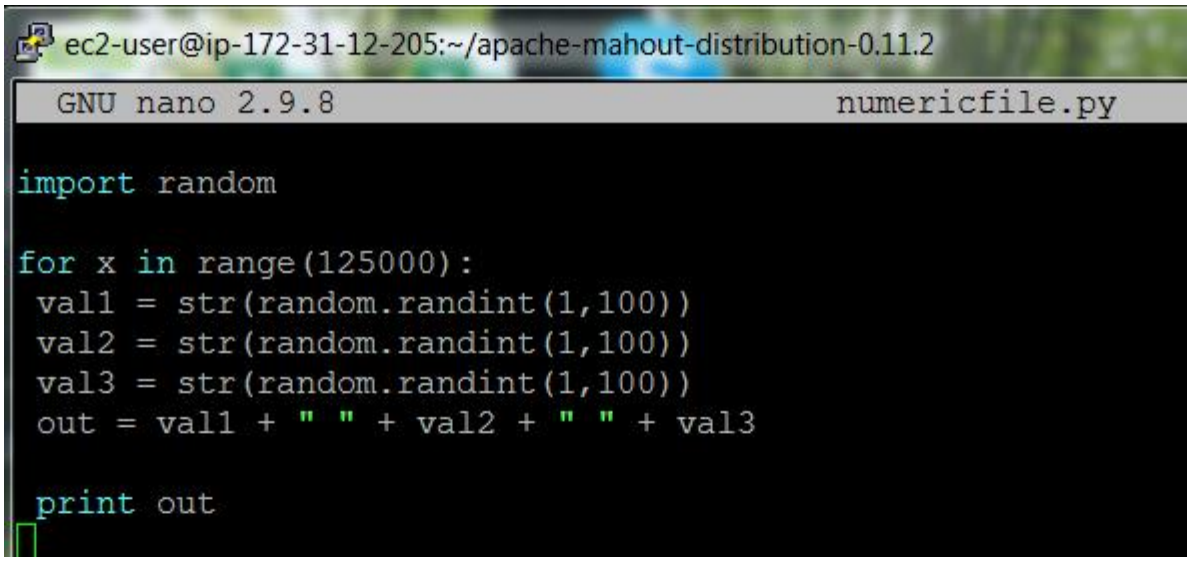
hadoop fs -cat /user/ec2-user/Part2Pig/part-r-00000



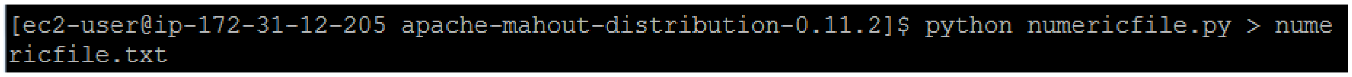
```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -cat /user/ec2-user/Part2Pig/part-r-00000
PERU,228441124985
BRAZIL,225595365795
CANADA,240715548308
ARGENTINA,243988697072
UNITED STATES,244263170830
[ec2-user@ip-172-31-12-205 ~]$
```

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.



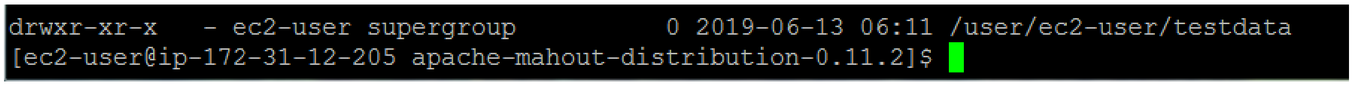
```
ec2-user@ip-172-31-12-205:~/apache-mahout-distribution-0.11.2
  GNU nano 2.9.8                        numericfile.py

import random

for x in range(125000):
 val1 = str(random.randint(1,100))
 val2 = str(random.randint(1,100))
 val3 = str(random.randint(1,100))
 out = val1 + " " + val2 + " " + val3

 print out
```



```
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$ python numericfile.py > nume
ricfile.txt
```

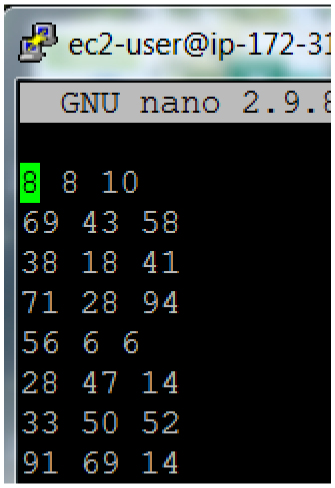
A. **(5 pts) Using Mahout synthetic clustering as you have in a previous assignment on sample data. This entails running the same clustering command, but substituting your own input data and the right number of clusters.**

hadoop fs -mkdir -p testdata



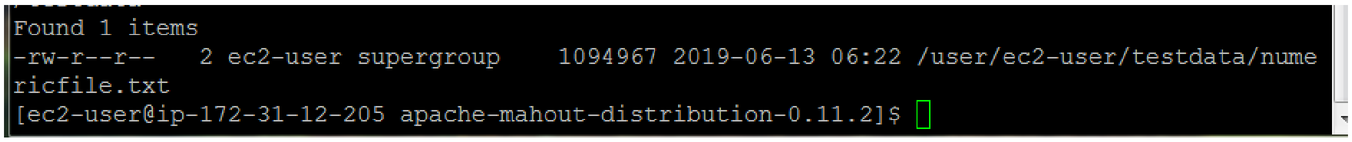
```
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:11 /user/ec2-user/testdata
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$ 
```

Nano numericfile.txt



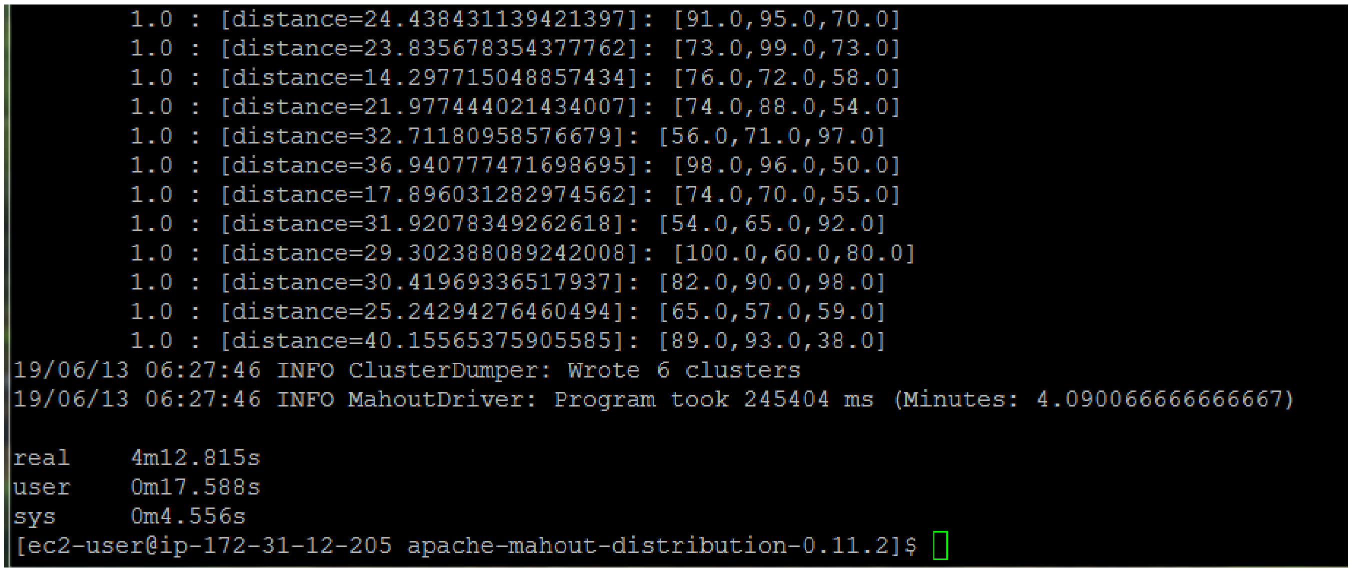
```
ec2-user@ip-172-31
 GNU nano 2.9.

8 8 10
69 43 58
38 18 41
71 28 94
56 6 6
28 47 14
33 50 52
91 69 14
```

hadoop fs -put numericfile.txt testdata/



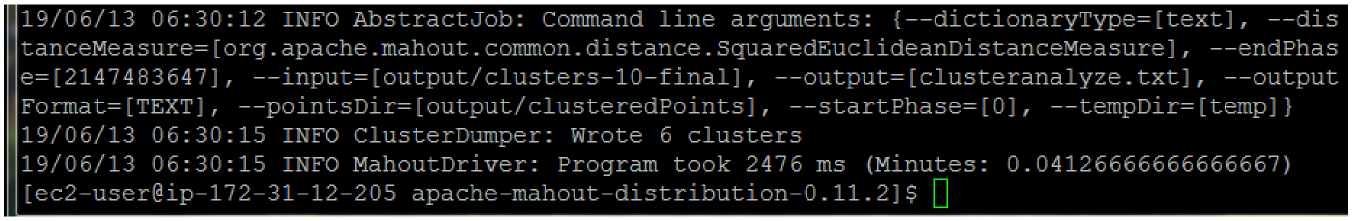
```
Found 1 items
-rw-r--r--   2 ec2-user supergroup    1094967 2019-06-13 06:22 /user/ec2-user/testdata/nume
ricfile.txt
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$
```

time mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job



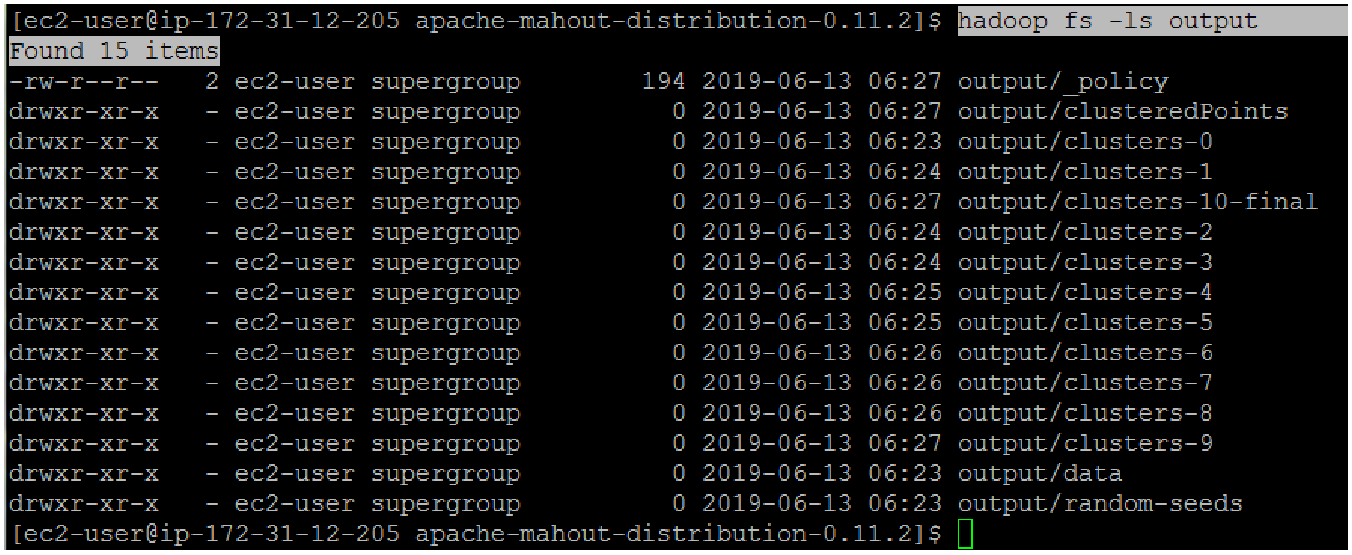
```
        1.0 : [distance=24.438431139421397]: [91.0,95.0,70.0]
        1.0 : [distance=23.835678354377762]: [73.0,99.0,73.0]
        1.0 : [distance=14.297715048857434]: [76.0,72.0,58.0]
        1.0 : [distance=21.977444021434007]: [74.0,88.0,54.0]
        1.0 : [distance=32.71180958576679]: [56.0,71.0,97.0]
        1.0 : [distance=36.940777471698695]: [98.0,96.0,50.0]
        1.0 : [distance=17.896031282974562]: [74.0,70.0,55.0]
        1.0 : [distance=31.92078349262618]: [54.0,65.0,92.0]
        1.0 : [distance=29.302388089242008]: [100.0,60.0,80.0]
        1.0 : [distance=30.41969336517937]: [82.0,90.0,98.0]
        1.0 : [distance=25.24294276460494]: [65.0,57.0,59.0]
        1.0 : [distance=40.15565375905585]: [89.0,93.0,38.0]
19/06/13 06:27:46 INFO ClusterDumper: Wrote 6 clusters
19/06/13 06:27:46 INFO MahoutDriver: Program took 245404 ms (Minutes: 4.090066666666667)

real    4m12.815s
user    0m17.588s
sys     0m4.556s
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$
```

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



```
19/06/13 06:30:12 INFO AbstractJob: Command line arguments: {--dictionaryType=[text], --dis
tanceMeasure=[org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure], --endPhas
e=[2147483647], --input=[output/clusters-10-final], --output=[clusteranalyze.txt], --output
Format=[TEXT], --pointsDir=[output/clusteredPoints], --startPhase=[0], --tempDir=[temp]}
19/06/13 06:30:15 INFO ClusterDumper: Wrote 6 clusters
19/06/13 06:30:15 INFO MahoutDriver: Program took 2476 ms (Minutes: 0.04126666666666667)
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$
```

hadoop fs -ls output



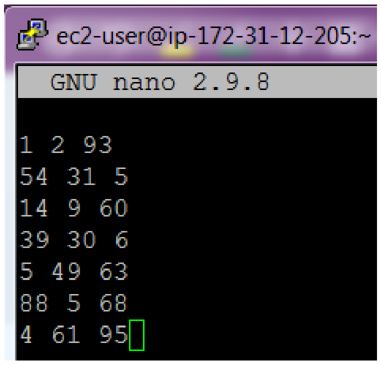
```
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$ hadoop fs -ls output
Found 15 items
-rw-r--r--   2 ec2-user supergroup        194 2019-06-13 06:27 output/_policy
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:27 output/clusteredPoints
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:23 output/clusters-0
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:24 output/clusters-1
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:27 output/clusters-10-final
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:24 output/clusters-2
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:24 output/clusters-3
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:25 output/clusters-4
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:25 output/clusters-5
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:26 output/clusters-6
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:26 output/clusters-7
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:26 output/clusters-8
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:27 output/clusters-9
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:23 output/data
drwxr-xr-x   - ec2-user supergroup          0 2019-06-13 06:23 output/random-seeds
[ec2-user@ip-172-31-12-205 apache-mahout-distribution-0.11.2]$
```

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 2nd iteration, the centers will be given to you by a previous pass of KMeans.

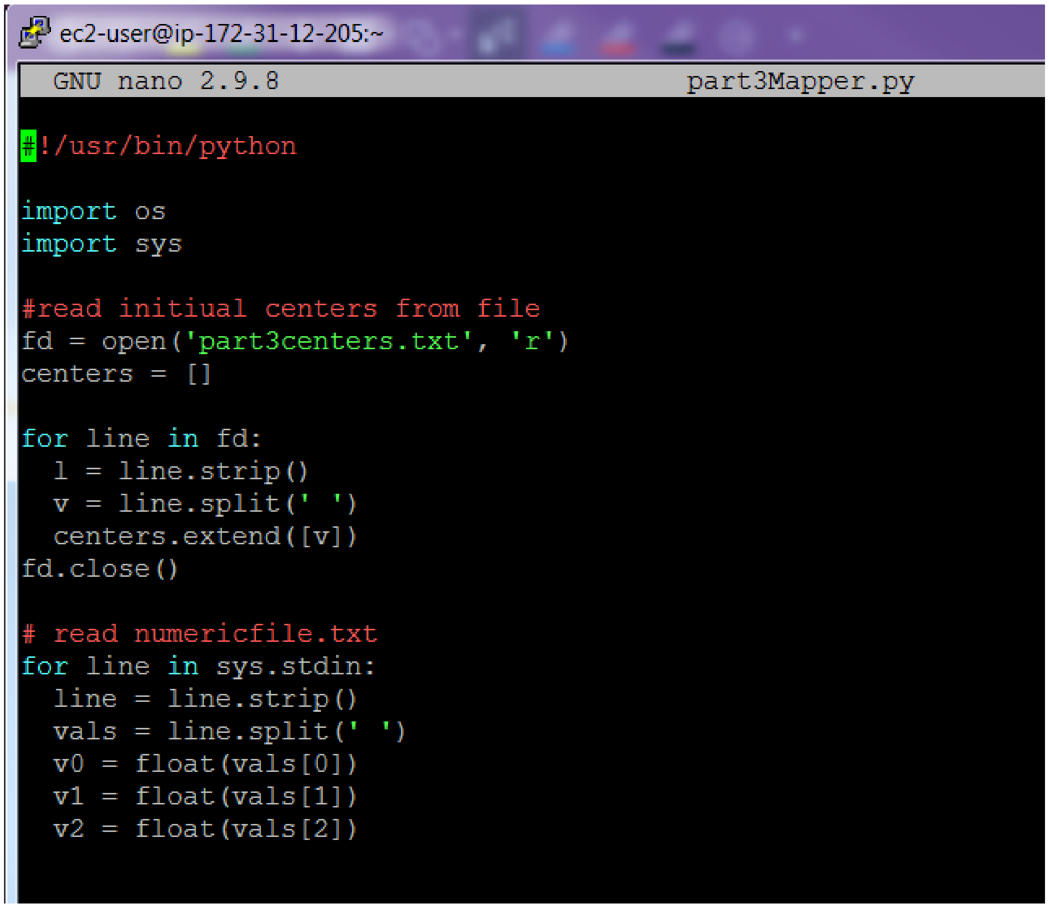
cp /home/ec2-user/centers.txt ./part3centers.txt

hadoop fs -put part3numericfile.txt /data/

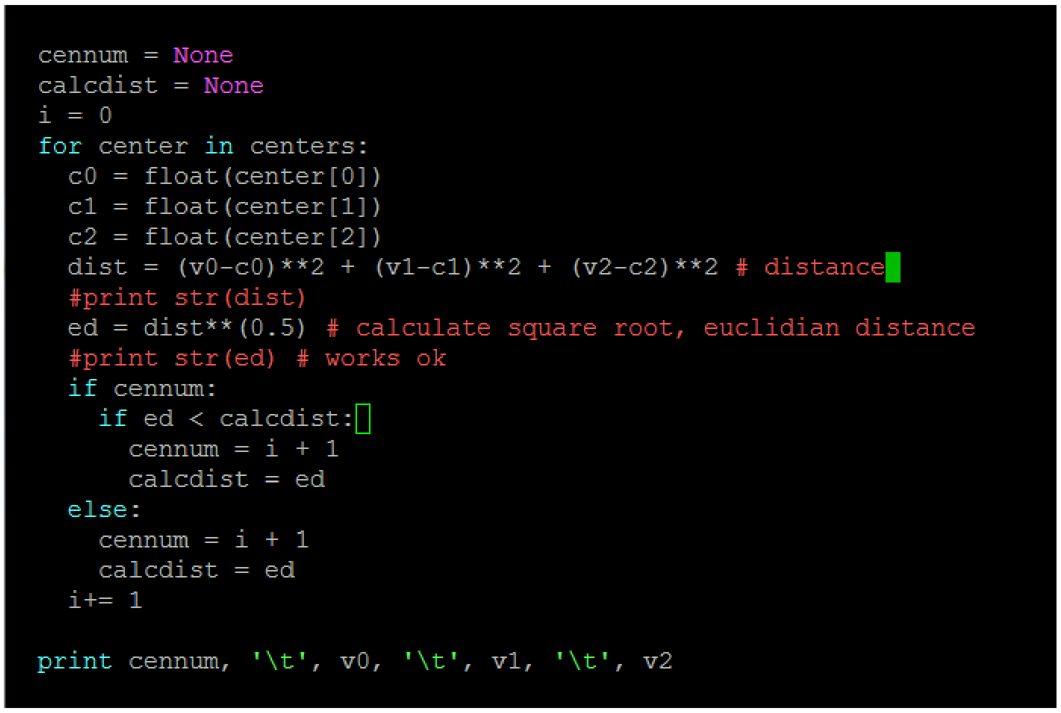


```
ec2-user@ip-172-31-12-205:~
 GNU nano 2.9.8

1 2 93
54 31 5
14 9 60
39 30 6
5 49 63
88 5 68
4 61 95
```

Mapper: part3Mapper.py

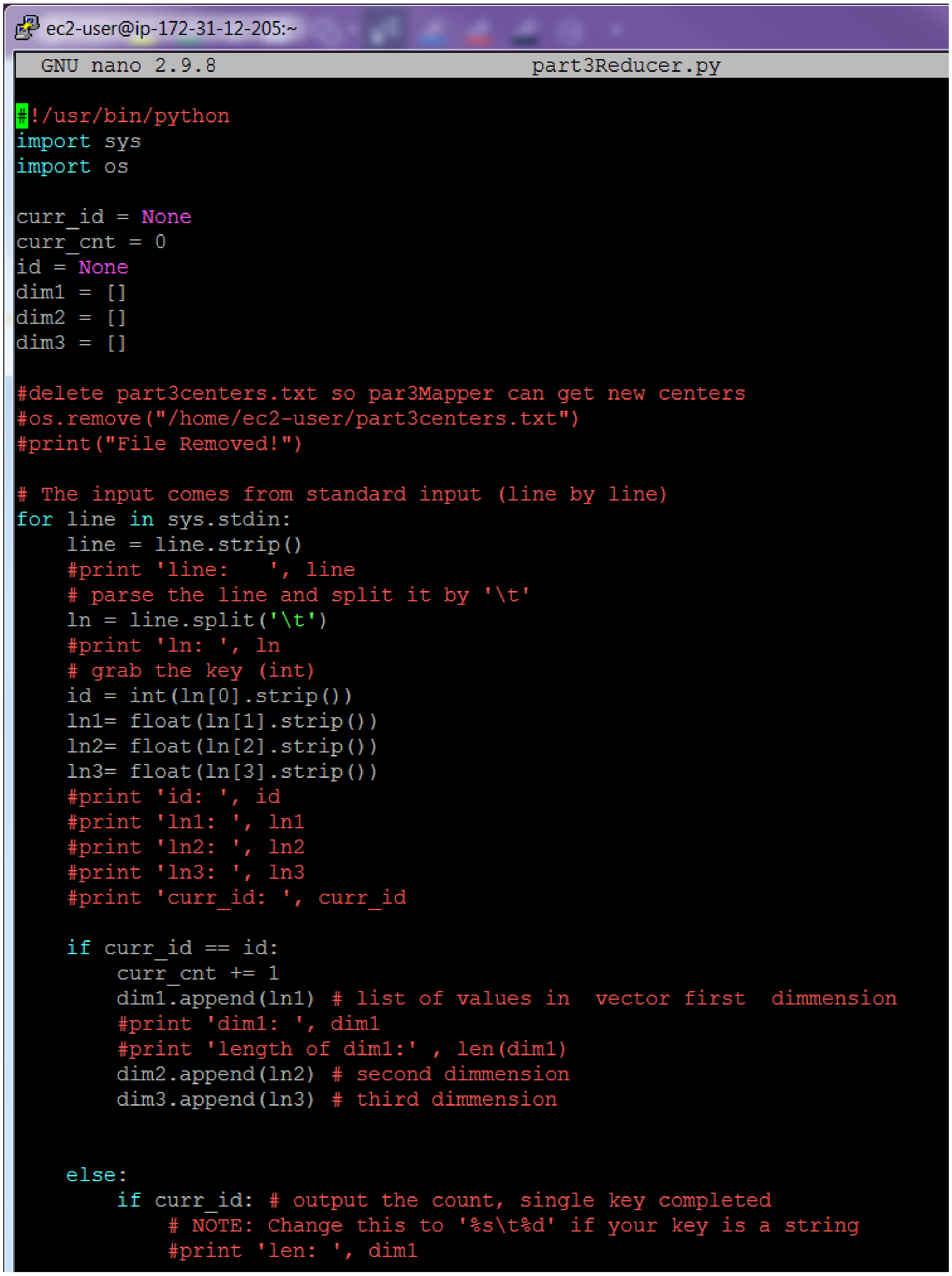


```
ec2-user@ip-172-31-12-205:~
 GNU nano 2.9.8                              part3Mapper.py

#!/usr/bin/python

import os
import sys

#read initiual centers from file
fd = open('part3centers.txt', 'r')
centers = []

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

# read numericfile.txt
for line in sys.stdin:
  line = line.strip()
  vals = line.split(' ')
  v0 = float(vals[0])
  v1 = float(vals[1])
  v2 = float(vals[2])
```



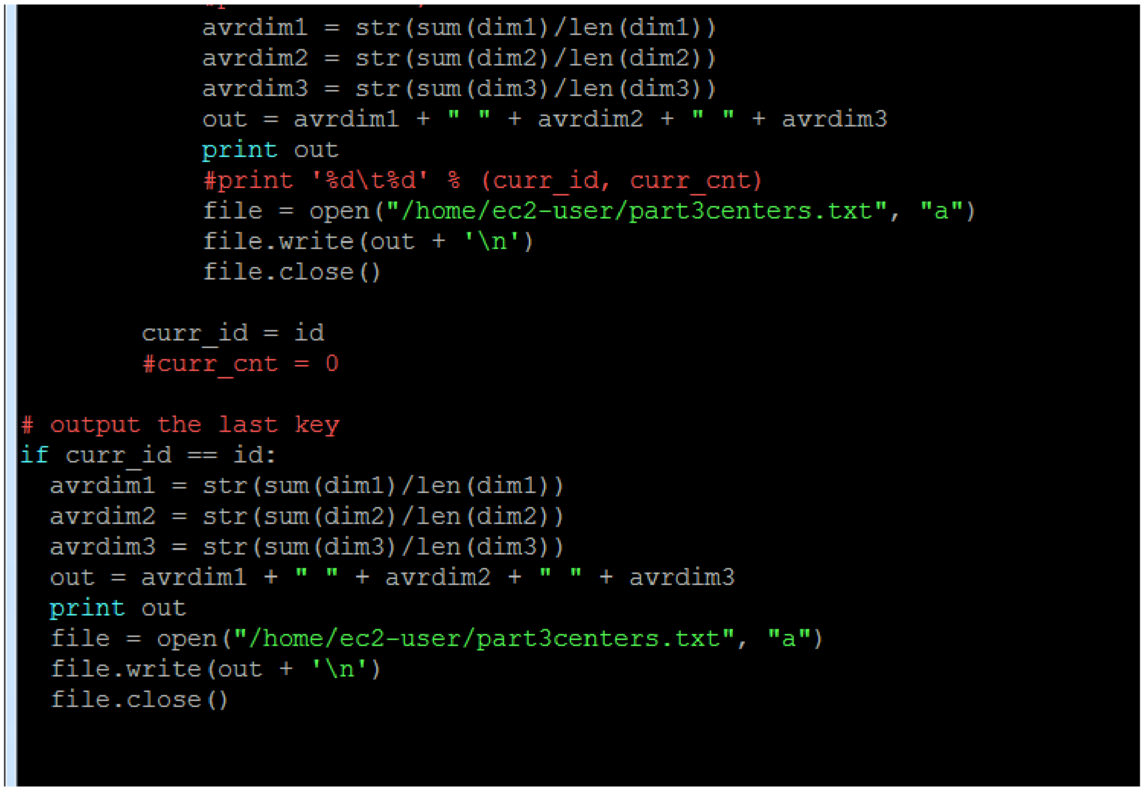
```
cennum = None
calcdist = None
i = 0
for center in centers:
  c0 = float(center[0])
  c1 = float(center[1])
  c2 = float(center[2])
  dist = (v0-c0)**2 + (v1-c1)**2 + (v2-c2)**2 # distance
  #print str(dist)
  ed = dist**(0.5) # calculate square root, euclidian distance
  #print str(ed) # works ok
  if cennum:
    if ed < calcdist:
      cennum = i + 1
      calcdist = ed
  else:
    cennum = i + 1
    calcdist = ed
  i+= 1

print cennum, '\t', v0, '\t', v1, '\t', v2
```

Reducer: part3Reducer.py



ec2-user@ip-172-31-12-205:~

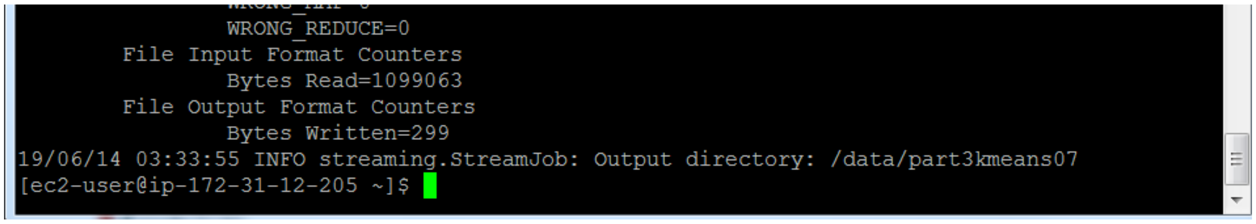
```
  GNU nano 2.9.8                       part3Reducer.py

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

curr_id = None
curr_cnt = 0
id = None
dim1 = []
dim2 = []
dim3 = []

#delete part3centers.txt so par3Mapper can get new centers
#os.remove("/home/ec2-user/part3centers.txt")
#print("File Removed!")

# The input comes from standard input (line by line)
for line in sys.stdin:
    line = line.strip()
    #print 'line:   ', line
    # parse the line and split it by '\t'
    ln = line.split('\t')
    #print 'ln: ', ln
    # grab the key (int)
    id = int(ln[0].strip())
    ln1= float(ln[1].strip())
    ln2= float(ln[2].strip())
    ln3= float(ln[3].strip())
    #print 'id: ', id
    #print 'ln1: ', ln1
    #print 'ln2: ', ln2
    #print 'ln3: ', ln3
    #print 'curr_id: ', curr_id

    if curr_id == id:
        curr_cnt += 1
        dim1.append(ln1) # list of values in  vector first  dimmension
        #print 'dim1: ', dim1
        #print 'length of dim1:' , len(dim1)
        dim2.append(ln2) # second dimmension
        dim3.append(ln3) # third dimmension


    else:
        if curr_id: # output the count, single key completed
            # NOTE: Change this to '%s\t%d' if your key is a string
            #print 'len: ', dim1
```



```
            avrdim1 = str(sum(dim1)/len(dim1))
            avrdim2 = str(sum(dim2)/len(dim2))
            avrdim3 = str(sum(dim3)/len(dim3))
            out = avrdim1 + " " + avrdim2 + " " + avrdim3
            print out
            #print '%d\t%d' % (curr_id, curr_cnt)
            file = open("/home/ec2-user/part3centers.txt", "a")
            file.write(out + '\n')
            file.close()

        curr_id = id
        #curr_cnt = 0

# output the last key
if curr_id == id:
  avrdim1 = str(sum(dim1)/len(dim1))
  avrdim2 = str(sum(dim2)/len(dim2))
  avrdim3 = str(sum(dim3)/len(dim3))
  out = avrdim1 + " " + avrdim2 + " " + avrdim3
  print out
  file = open("/home/ec2-user/part3centers.txt", "a")
  file.write(out + '\n')
  file.close()
```

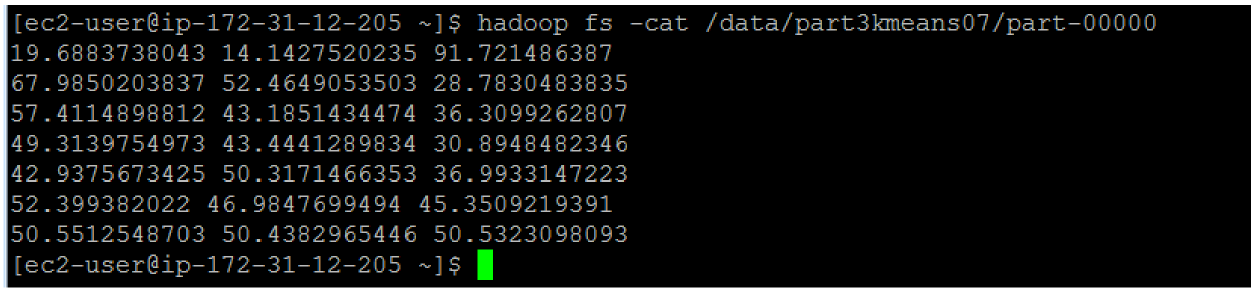
1st run

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -input /data/part3numericfile.txt -file part3centers.txt -output /data/part3kmeans07 -mapper part3Mapper.py -file /home/ec2-user/part3Mapper.py -reducer part3Reducer.py -file /home/ec2-user/part3Reducer.py



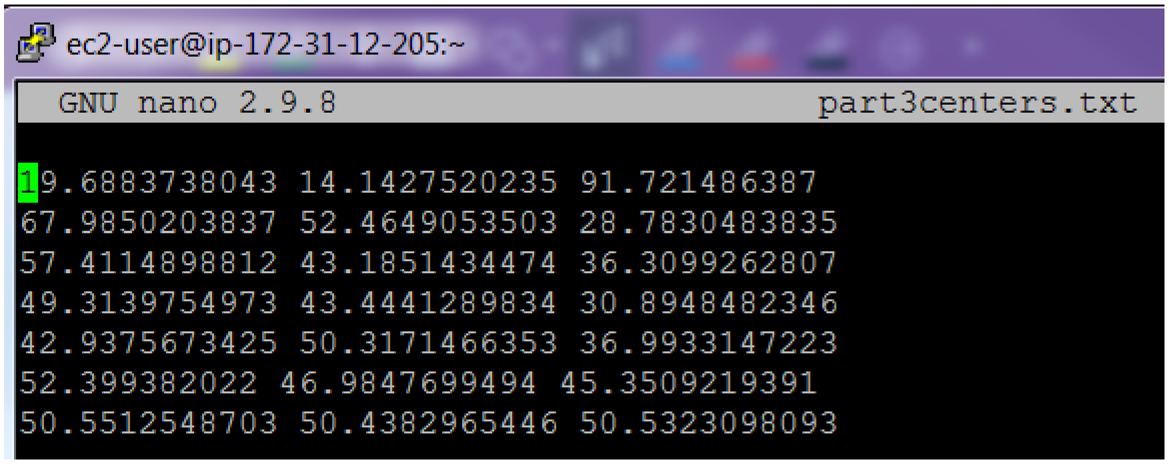
```
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1099063
        File Output Format Counters
                Bytes Written=299
19/06/14 03:33:55 INFO streaming.StreamJob: Output directory: /data/part3kmeans07
[ec2-user@ip-172-31-12-205 ~]$
```

hadoop fs -cat /data/part3kmeans07/part-00000



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -cat /data/part3kmeans07/part-00000
19.6883738043 14.1427520235 91.721486387
67.9850203837 52.4649053503 28.7830483835
57.4114898812 43.1851434474 36.3099262807
49.3139754973 43.4441289834 30.8948482346
42.9375673425 50.3171466353 36.9933147223
52.399382022 46.9847699494 45.3509219391
50.5512548703 50.4382965446 50.5323098093
[ec2-user@ip-172-31-12-205 ~]$
```

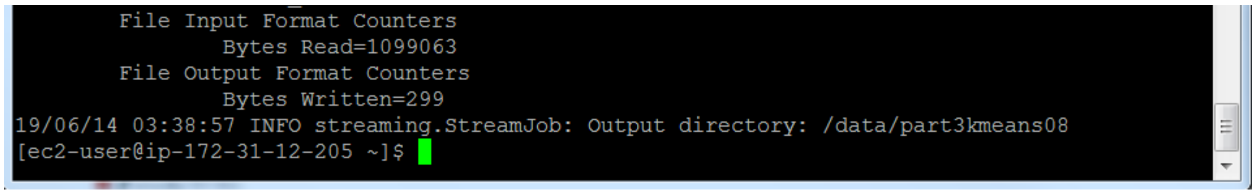
rm part3centers.txt
hadoop fs -get /data/part3kmeans07/part-00000 part3centers.txt
nano part3centers.txt



```
ec2-user@ip-172-31-12-205:~
  GNU nano 2.9.8                          part3centers.txt

19.6883738043 14.1427520235 91.721486387
67.9850203837 52.4649053503 28.7830483835
57.4114898812 43.1851434474 36.3099262807
49.3139754973 43.4441289834 30.8948482346
42.9375673425 50.3171466353 36.9933147223
52.399382022 46.9847699494 45.3509219391
50.5512548703 50.4382965446 50.5323098093
```

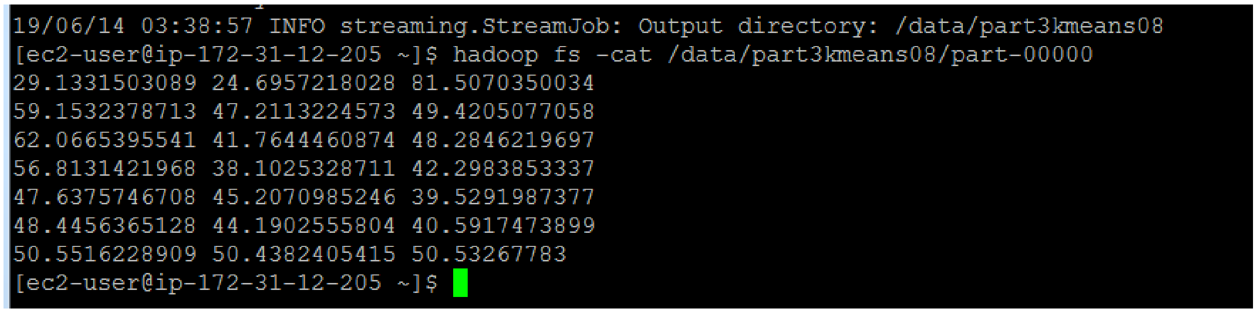
2nd run

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -input /data/part3numericfile.txt -file part3centers.txt -output /data/part3kmeans08 -mapper part3Mapper.py -file /home/ec2-user/part3Mapper.py -reducer part3Reducer.py -file /home/ec2-user/part3Reducer.py



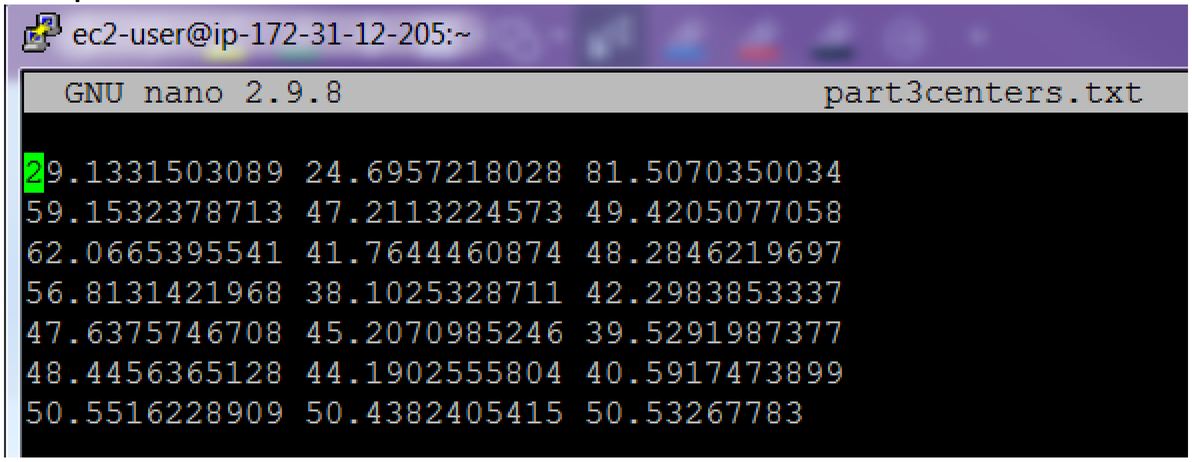
```
        File Input Format Counters
                Bytes Read=1099063
        File Output Format Counters
                Bytes Written=299
19/06/14 03:38:57 INFO streaming.StreamJob: Output directory: /data/part3kmeans08
[ec2-user@ip-172-31-12-205 ~]$
```

hadoop fs -cat /data/part3kmeans08/part-00000



```
19/06/14 03:38:57 INFO streaming.StreamJob: Output directory: /data/part3kmeans08
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -cat /data/part3kmeans08/part-00000
29.1331503089 24.6957218028 81.5070350034
59.1532378713 47.2113224573 49.4205077058
62.0665395541 41.7644460874 48.2846219697
56.8131421968 38.1025328711 42.2983853337
47.6375746708 45.2070985246 39.5291987377
48.4456365128 44.1902555804 40.5917473899
50.5516228909 50.4382405415 50.53267783
[ec2-user@ip-172-31-12-205 ~]$
```

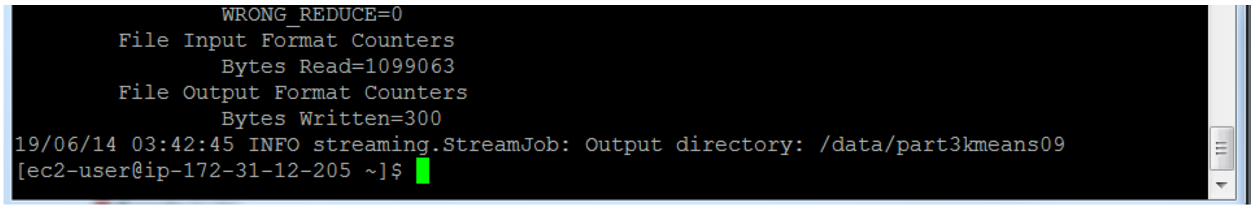
rm part3centers.txt
hadoop fs -get /data/part3kmeans08/part-00000 part3centers.txt
nano part3centers.txt



```
ec2-user@ip-172-31-12-205:~
  GNU nano 2.9.8                    part3centers.txt

29.1331503089 24.6957218028 81.5070350034
59.1532378713 47.2113224573 49.4205077058
62.0665395541 41.7644460874 48.2846219697
56.8131421968 38.1025328711 42.2983853337
47.6375746708 45.2070985246 39.5291987377
48.4456365128 44.1902555804 40.5917473899
50.5516228909 50.4382405415 50.53267783
```

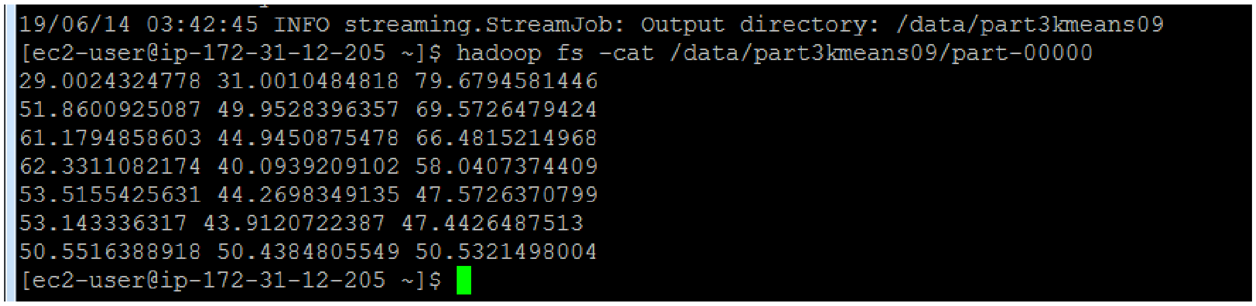
3rd run

hadoop jar /home/ec2-user/hadoop-2.6.4/hadoop-streaming-2.6.4.jar -input /data/part3numericfile.txt -file part3centers.txt -output /data/part3kmeans09 -mapper part3Mapper.py -file /home/ec2-user/part3Mapper.py -reducer part3Reducer.py -file /home/ec2-user/part3Reducer.py



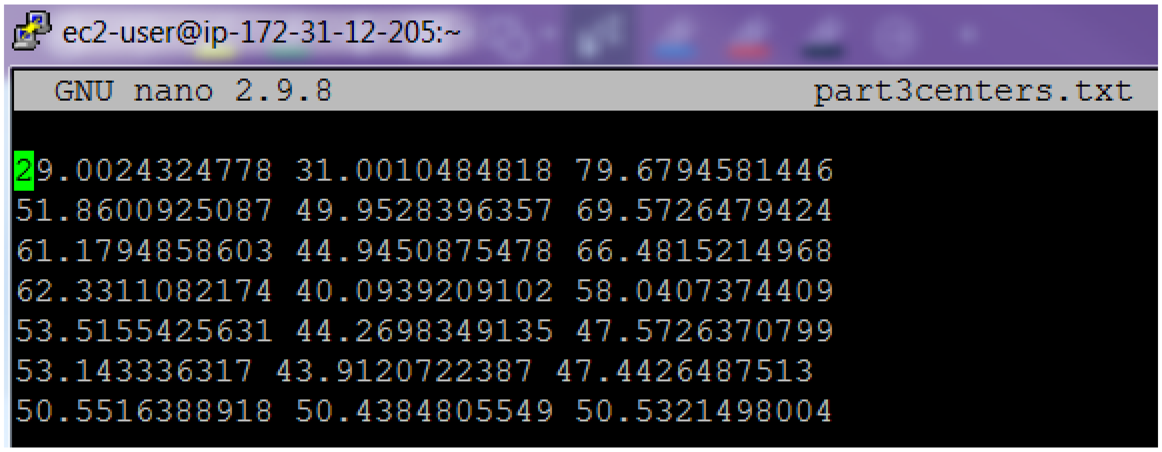
```
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1099063
        File Output Format Counters
                Bytes Written=300
19/06/14 03:42:45 INFO streaming.StreamJob: Output directory: /data/part3kmeans09
[ec2-user@ip-172-31-12-205 ~]$
```

hadoop fs -cat /data/part3kmeans09/part-00000



```
19/06/14 03:42:45 INFO streaming.StreamJob: Output directory: /data/part3kmeans09
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -cat /data/part3kmeans09/part-00000
29.0024324778 31.0010484818 79.6794581446
51.8600925087 49.9528396357 69.5726479424
61.1794858603 44.9450875478 66.4815214968
62.3311082174 40.0939209102 58.0407374409
53.5155425631 44.2698349135 47.5726370799
53.143336317 43.9120722387 47.4426487513
50.5516388918 50.4384805549 50.5321498004
[ec2-user@ip-172-31-12-205 ~]$
```

rm part3centers.txt
hadoop fs -get /data/part3kmeans09/part-00000 part3centers.txt
nano part3centers.txt



```
ec2-user@ip-172-31-12-205:~
  GNU nano 2.9.8                          part3centers.txt

29.0024324778 31.0010484818 79.6794581446
51.8600925087 49.9528396357 69.5726479424
61.1794858603 44.9450875478 66.4815214968
62.3311082174 40.0939209102 58.0407374409
53.5155425631 44.2698349135 47.5726370799
53.143336317 43.9120722387 47.4426487513
50.5516388918 50.4384805549 50.5321498004
```

<u>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.</u>