Xiao Song
Mining Big Data Project

In this part of the project, I executed queries using Hive, Pig and Hadoop streaming and develop a custom version of KMeans clustering. The schema is available below.
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/

I created a small sample input for testing my code.

Part 1: Data Transformation

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

In all solutions I followed the requirements to switch odd and even columns (i.e., switch the positions of columns 1 and 2, columns 3 and 4, etc.).

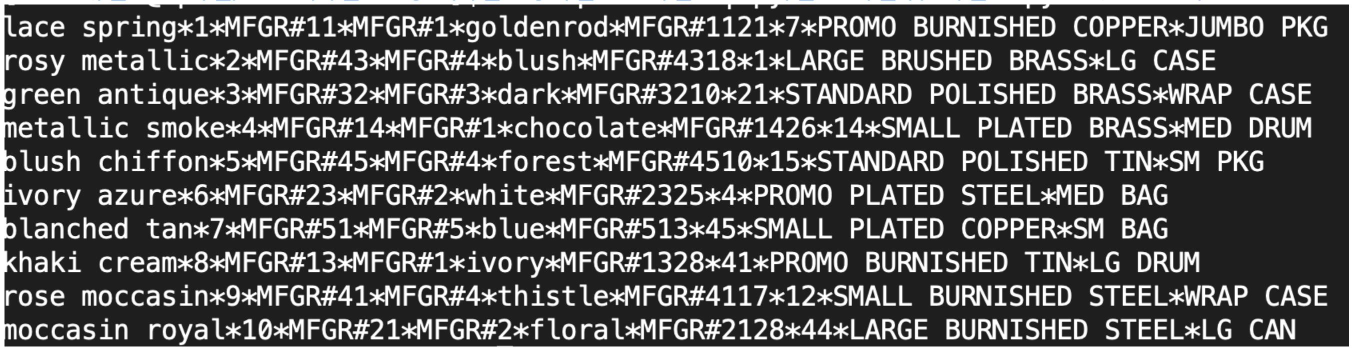
Hive code:

```
Hive code:
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:
#!/usr/bin/python
import sys

for line in sys.stdin:
   line = line.strip()
   vals = line.split('|')
   col1=vals[0]
   col2=vals[1]
   col3=vals[2]
   col4=vals[3]
   col5=vals[4]
   col6=vals[5]
   col7=vals[6]
   col8=vals[7]
   col9=vals[8]
   print '*'.join([col2,col1,col4,col3,col6,col5,col8,col7,col9])
```

```
head part.tbl | python delimiter.py
```

Hive code (part2):

```
create table part2(
    p_name       varchar(22),
    p_partkey    int,
    p_category   varchar(7),
    p_mfgr       varchar(6),
    p_color      varchar(11),
    p_brand1     varchar(9),
    p_size       int,
    p_type       varchar(25),
    p_container  varchar(10))
    ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;
ADD FILE /home/ec2-user/delimiter.py;
hive> INSERT OVERWRITE TABLE part2
   > SELECT TRANSFORM (p_partkey,p_name,p_mfgr,p_category,p_brand1,p_color,p_type,p_size,p_container)
   > USING 'python delimiter.py'
   > AS (p_name,p_partkey,p_category,p_mfgr,p_color,p_brand1,p_size,p_type,p_container) FROM part;
```



```
lace spring*1*MFGR#11*MFGR#1*goldenrod*MFGR#1121*7*PROMO BURNISHED COPPER*JUMBO PKG
rosy metallic*2*MFGR#43*MFGR#4*blush*MFGR#4318*1*LARGE BRUSHED BRASS*LG CASE
green antique*3*MFGR#32*MFGR#3*dark*MFGR#3210*21*STANDARD POLISHED BRASS*WRAP CASE
metallic smoke*4*MFGR#14*MFGR#1*chocolate*MFGR#1426*14*SMALL PLATED BRASS*MED DRUM
blush chiffon*5*MFGR#45*MFGR#4*forest*MFGR#4510*15*STANDARD POLISHED TIN*SM PKG
ivory azure*6*MFGR#23*MFGR#2*white*MFGR#2325*4*PROMO PLATED STEEL*MED BAG
blanched tan*7*MFGR#51*MFGR#5*blue*MFGR#513*45*SMALL PLATED COPPER*SM BAG
khaki cream*8*MFGR#13*MFGR#1*ivory*MFGR#1328*41*PROMO BURNISHED TIN*LG DRUM
rose moccasin*9*MFGR#41*MFGR#4*thistle*MFGR#4117*12*SMALL BURNISHED STEEL*WRAP CASE
moccasin royal*10*MFGR#21*MFGR#2*floral*MFGR#2128*44*LARGE BURNISHED STEEL*LG CAN
```

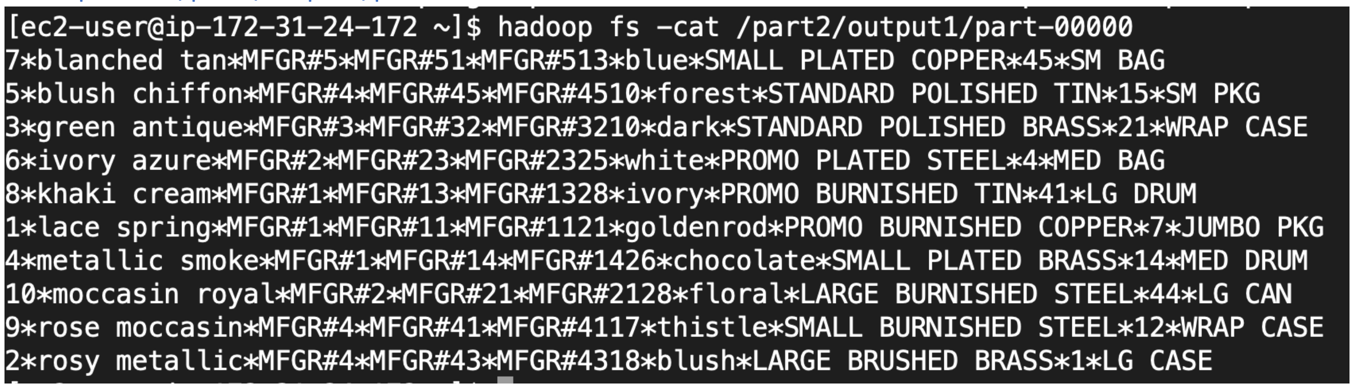
Hadoopstreaming code:

```
head part.tbl > part.tbl.sam
```

```
hadoop fs -mkdir /user/ec2-user/part
hadoop fs -put part.tbl.sample part
```

```
delimitermapper.py:
#!/usr/bin/python
import sys

for line in sys.stdin:
   line = line.strip()
   vals = line.split('|')
   print '|'.join([vals[1],vals[0],vals[3],vals[2],vals[5],vals[4],vals[7],vals[6],vals[8]])

delimiterreducer.py:
#!/usr/bin/python
import sys

for line in sys.stdin:
   line = line.strip()
   vals = line.split('|')
   print '*'.join([vals[1],vals[0],vals[3],vals[2],vals[5],vals[4],vals[7],vals[6],vals[8]])
```

```
head part.tbl.sample | python delimitermapper.py | python delimiterreducer.py

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/part -output /part2/output1  -mapper
delimitermapper.py -reducer delimiterreducer.py -file delimiterreducer.py -file delimitermapper.py

hadoop fs -ls /part2/output1

hadoop fs -cat /part2/output1/part-00000
```

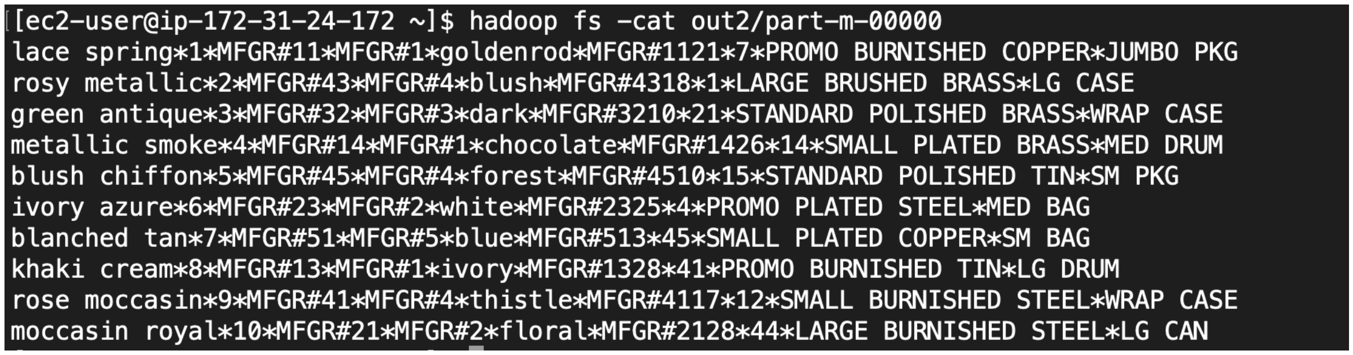


```
[ec2-user@ip-172-31-24-172 ~]$ hadoop fs -cat /part2/output1/part-00000
7*blanched tan*MFGR#5*MFGR#51*MFGR#513*blue*SMALL PLATED COPPER*45*SM BAG
5*blush chiffon*MFGR#4*MFGR#45*MFGR#4510*forest*STANDARD POLISHED TIN*15*SM PKG
3*green antique*MFGR#3*MFGR#32*MFGR#3210*dark*STANDARD POLISHED BRASS*21*WRAP CASE
6*ivory azure*MFGR#2*MFGR#23*MFGR#2325*white*PROMO PLATED STEEL*4*MED BAG
8*khaki cream*MFGR#1*MFGR#13*MFGR#1328*ivory*PROMO BURNISHED TIN*41*LG DRUM
1*lace spring*MFGR#1*MFGR#11*MFGR#1121*goldenrod*PROMO BURNISHED COPPER*7*JUMBO PKG
4*metallic smoke*MFGR#1*MFGR#14*MFGR#1426*chocolate*SMALL PLATED BRASS*14*MED DRUM
10*moccasin royal*MFGR#2*MFGR#21*MFGR#2128*floral*LARGE BURNISHED STEEL*44*LG CAN
9*rose moccasin*MFGR#4*MFGR#41*MFGR#4117*thistle*SMALL BURNISHED STEEL*12*WRAP CASE
2*rosy metallic*MFGR#4*MFGR#43*MFGR#4318*blush*LARGE BRUSHED BRASS*1*LG CASE
```

Pig Code:

```
PartData = LOAD '/user/ec2-user/part/part.tbl.sample' 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);

DUMP PartData;
```

```
newpart = FOREACH PartData GENERATE
p_name,p_partkey,p_category,p_mfgr,p_color,p_brand1,p_size,p_type,p_container;
STORE newpart INTO 'out2' USING PigStorage('*');

hadoop fs -ls
hadoop fs -ls out2/
hadoop fs -cat out2/part-m-00000
```



```
[ec2-user@ip-172-31-24-172 ~]$ hadoop fs -cat out2/part-m-00000
lace spring*1*MFGR#11*MFGR#1*goldenrod*MFGR#1121*7*PROMO BURNISHED COPPER*JUMBO PKG
rosy metallic*2*MFGR#43*MFGR#4*blush*MFGR#4318*1*LARGE BRUSHED BRASS*LG CASE
green antique*3*MFGR#32*MFGR#3*dark*MFGR#3210*21*STANDARD POLISHED BRASS*WRAP CASE
metallic smoke*4*MFGR#14*MFGR#1*chocolate*MFGR#1426*14*SMALL PLATED BRASS*MED DRUM
blush chiffon*5*MFGR#45*MFGR#4*forest*MFGR#4510*15*STANDARD POLISHED TIN*SM PKG
ivory azure*6*MFGR#23*MFGR#2*white*MFGR#2325*4*PROMO PLATED STEEL*MED BAG
blanched tan*7*MFGR#51*MFGR#5*blue*MFGR#513*45*SMALL PLATED COPPER*SM BAG
khaki cream*8*MFGR#13*MFGR#1*ivory*MFGR#1328*41*PROMO BURNISHED TIN*LG DRUM
rose moccasin*9*MFGR#41*MFGR#4*thistle*MFGR#4117*12*SMALL BURNISHED STEEL*WRAP CASE
moccasin royal*10*MFGR#21*MFGR#2*floral*MFGR#2128*44*LARGE BURNISHED STEEL*LG CAN
```

Part 2: Querying

Implemented the following query:

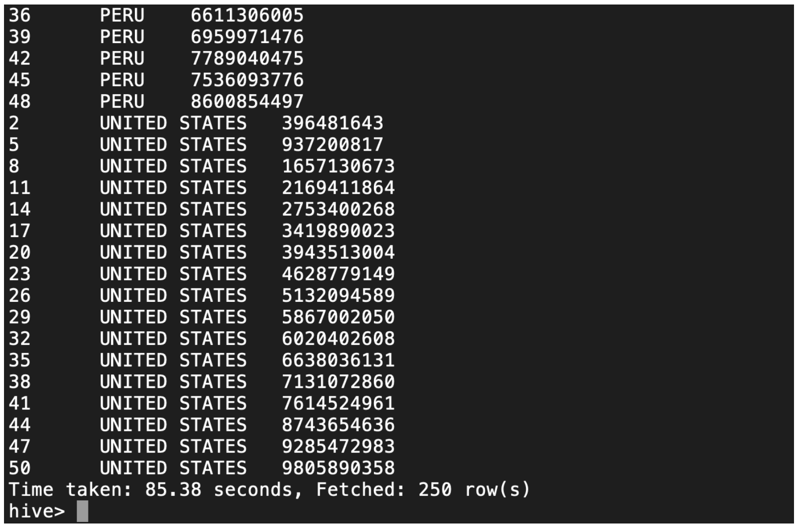
```
select lo_quantity, c_nation, sum(lo_revenue)
from customer, lineorder
where lo_custkey = c_custkey
  and c_region = 'AMERICA'
  and lo_discount BETWEEN 3 and 5
group by lo_quantity, c_nation;
```

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

Hive code:

```
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;

  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;
```

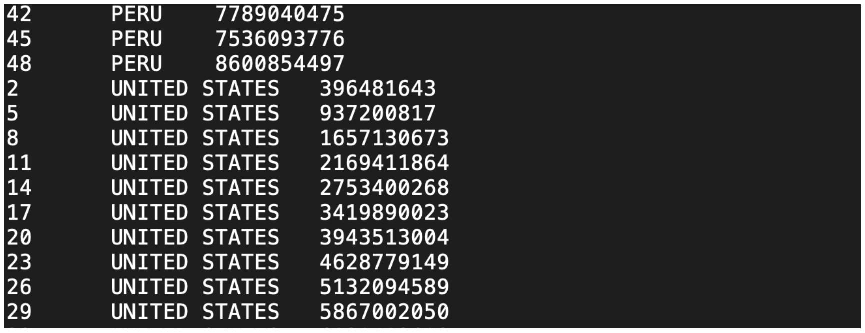
```
select lo_quantity, c_nation, sum(lo_revenue)
   > from customer, lineorder
   > where lo_custkey = c_custkey
   > and c_region = 'AMERICA'
   > and lo_discount BETWEEN 3 and 5
   > group by lo_quantity, c_nation;
```



```
36      PERU    6611306005
39      PERU    6959971476
42      PERU    7789040475
45      PERU    7536093776
48      PERU    8600854497
2       UNITED STATES   396481643
5       UNITED STATES   937200817
8       UNITED STATES   1657130673
11      UNITED STATES   2169411864
14      UNITED STATES   2753400268
17      UNITED STATES   3419890023
20      UNITED STATES   3943513004
23      UNITED STATES   4628779149
26      UNITED STATES   5132094589
29      UNITED STATES   5867002050
32      UNITED STATES   6020402608
35      UNITED STATES   6638036131
38      UNITED STATES   7131072860
41      UNITED STATES   7614524961
44      UNITED STATES   8743654636
47      UNITED STATES   9285472983
50      UNITED STATES   9805890358
Time taken: 85.38 seconds, Fetched: 250 row(s)
hive>
```

Pig code:

```
hadoop fs -mkdir /user/ec2-user/lineorder
hadoop fs -mkdir /user/ec2-user/customer
hadoop fs -put customer.tbl customer
hadoop fs -put lineorder.tbl lineorder

CustomerData = LOAD '/user/ec2-user/customer/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);

LineorderData = LOAD '/user/ec2-user/lineorder/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);

DataJoin = JOIN CustomerData BY (c_custkey),LineorderData BY (lo_custkey);
FilterRegion = FILTER DataJoin BY c_region == 'AMERICA';
FilterDiscount = FILTER FilterRegion BY lo_discount >=3 AND lo_discount<=5;
GroupData = GROUP FilterDiscount BY (LineorderData::lo_quantity, CustomerData::c_nation);
Result = FOREACH GroupData GENERATE lo_quantity,c_nation,SUM(lo_revenue);
```



```
42      PERU    7789040475
45      PERU    7536093776
48      PERU    8600854497
2       UNITED STATES   396481643
5       UNITED STATES   937200817
8       UNITED STATES   1657130673
11      UNITED STATES   2169411864
14      UNITED STATES   2753400268
17      UNITED STATES   3419890023
20      UNITED STATES   3943513004
23      UNITED STATES   4628779149
26      UNITED STATES   5132094589
29      UNITED STATES   5867002050
```

Hadoopstreaming code:

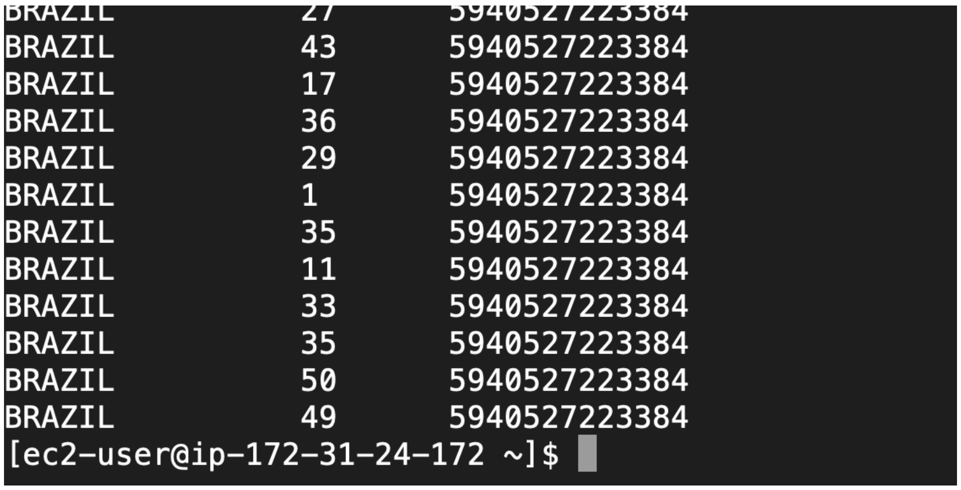
Mapperjoin.py:

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

# input comes from STDIN (standard input)
for line in sys.stdin:
   line = line.strip()
   splits = line.split('|')
   if len(splits[1]) > 2:
      print splits[0],'\t',splits[4], '\t', 'customer'
   else:
         print splits[2],'\t',splits[8],'\t',splits[12],'\t', 'lineorder'
```

Reducerjoin.py:

```
#!/usr/bin/python

import sys

currentKey = None
valsCustomer = None
valsLineorder = None
sumLo_r = 0
# input comes from STDIN
for line in sys.stdin:

    splits = line.strip().split('\t')
    key = splits[0]
    value = '\t'.join(splits[1:])

    if currentKey == key: # Same key
        if value.endswith('customer'):
            valsCustomer.append(value)
        if value.endswith('lineorder'):
            valsLineorder.append(value)
            sumLo_r = sumLo_r + int(splits[-3])
    else:
          if currentKey:
           for C in valsCustomer:
              for Lo in valsLineorder:
                  lenC = len(C)
                  lenLo = len(Lo)
                  if (lenC*lenLo > 0):
                    # Join means that there have to be rows on each side
                         Csplit=C.strip().split('\t')
                         Cvalue='\t'.join(Csplit[:-2])
                         Losplit=Lo.strip().split('\t')
                         Lovalue='\t'.join(Losplit[:-3])
                         print  Cvalue, '\t', Lovalue,'\t',sumLo_r
        valsCustomer = []
        valsLineorder = []
        currentKey = key
        if value.endswith('customer'):
            valsCustomer = [value]
            valsLineorder = []
        elif value.endswith('lineorder'):
            valsCustomer = []
            valsLineorder = [value]

            for C in valsCustomer:
              for Lo in valsLineorder:
                  lenC = len(C)
            lenLo = len(Lo)
            if (lenC*lenLo > 0):
             # Join means that there have to be rows on each side
                Csplit=C.strip().split('\t')
                Cvalue='\t'.join(Csplit[:-2])
                Losplit=Lo.strip().split('\t')
                Lovalue='\t'.join(Losplit[:-3])
                print  Cvalue, '\t', Lovalue,'\t',sumLo_r

  for C in valsCustomer:
    for Lo in valsLineorder:
      lenC = len(C)
      lenLo = len(Lo)
      if (lenC*lenLo > 0):
            # Join means that there have to be rows on each side
          Csplit=C.strip().split('\t')
          Cvalue='\t'.join(Csplit[:-2])
          Losplit=Lo.strip().split('\t')
          Lovalue='\t'.join(Losplit[:-3])
          print  Cvalue, '\t', Lovalue,'\t',sumLo_r
```

```
  hadoop fs -mkdir -p /joinset
  hadoop fs -put customer.tbl lineorder.tbl /joinset
  hadoop jar hadoop-streaming-2.6.4.jar -input /joinset -mapper mapperjoin.py -file mapperjoin.py -reducer
reducerjoin.py -file reducerjoin.py -output /output4
  hadoop fs -cat /output4/part-00000
```



```
BRAZIL          27      5940527223384
BRAZIL          43      5940527223384
BRAZIL          17      5940527223384
BRAZIL          36      5940527223384
BRAZIL          29      5940527223384
BRAZIL          1       5940527223384
BRAZIL          35      5940527223384
BRAZIL          11      5940527223384
BRAZIL          33      5940527223384
BRAZIL          35      5940527223384
BRAZIL          50      5940527223384
BRAZIL          49      5940527223384
[ec2-user@ip-172-31-24-172 ~]$
```

Part 3: Clustering (Codes Test just for random sample but not for customer orders)

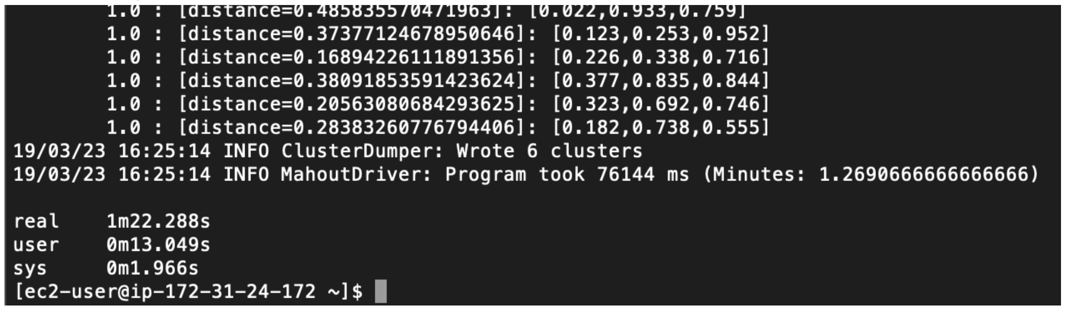
I created a new numeric file with 25,000 rows and 3 columns, separated by space in Jupyter Notebook.

a. Used Mahout synthetic clustering on sample data.

Python (Jupyter Notebook) code:

```
import numpy as np
a = np.random.random((25000,3))
b=list(a)
file = open('test5.txt','w')
file.write('\n'.join(' '.join('{:}'.format(item) for item in row) for row in b))
file.close()
```

```
hadoop fs -put test5.txt testdata
time mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job
mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints -- output
clusteranalyze.txt
```



```
        1.0 : [distance=0.485835570471963]: [0.022,0.933,0.759]
        1.0 : [distance=0.37377124678950646]: [0.123,0.253,0.952]
        1.0 : [distance=0.16894226111891356]: [0.226,0.338,0.716]
        1.0 : [distance=0.38091853591423624]: [0.377,0.835,0.844]
        1.0 : [distance=0.20563080684293625]: [0.323,0.692,0.746]
        1.0 : [distance=0.28383260776794406]: [0.182,0.738,0.555]
19/03/23 16:25:14 INFO ClusterDumper: Wrote 6 clusters
19/03/23 16:25:14 INFO MahoutDriver: Program took 76144 ms (Minutes: 1.2690666666666666)

real    1m22.288s
user    0m13.049s
sys     0m1.966s
[ec2-user@ip-172-31-24-172 ~]$
```

b. Applied hadoop streaming perform four iterations manually **using 6 centers** (initially with randomly chosen centers).

This required 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.

My reducer then computed the new centers, and at that point the iteration was done and the output of the reducer with new centers was given to the next pass of the same code.

The only difference between first and subsequent iteration was that in first iteration I had to pick the initial centers. Starting from 2nd iteration, the centers were given by a previous pass of KMeans.

Mapper code:

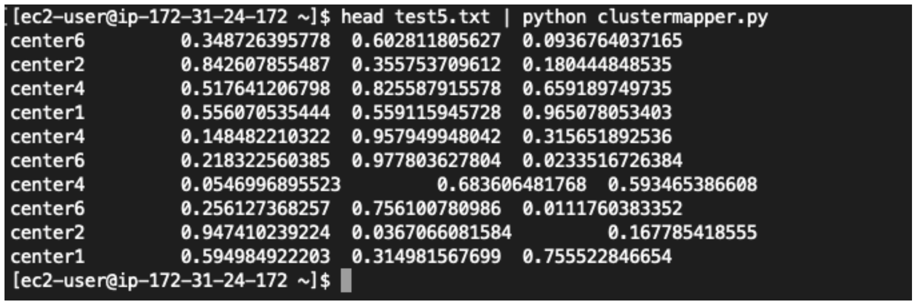
```
#!/usr/bin/python

import sys
import math
```

```
fd = open('initialCenter.txt', 'r')
center = fd.readlines()
fd.close()

# read every line in center file
lst_center1 = str(center[0]).strip().split()
x_center1 =  float(lst_center1[0])
y_center1 =  float(lst_center1[1])
z_center1 =  float(lst_center1[2])

lst_center2 = str(center[1]).strip().split()
x_center2 =  float(lst_center2[0])
y_center2 =  float(lst_center2[1])
z_center2 =  float(lst_center2[2])

lst_center3 = str(center[2]).strip().split()
x_center3 =  float(lst_center3[0])
y_center3 =  float(lst_center3[1])
z_center3 =  float(lst_center3[2])

lst_center4 = str(center[3]).strip().split()
x_center4 =  float(lst_center4[0])
y_center4 =  float(lst_center4[1])
z_center4 =  float(lst_center4[2])

lst_center5 = str(center[4]).strip().split()
x_center5 =  float(lst_center5[0])
y_center5 =  float(lst_center5[1])
z_center5 =  float(lst_center5[2])

lst_center6 = str(center[5]).strip().split()
x_center6 =  float(lst_center6[0])
y_center6 =  float(lst_center6[1])
z_center6 =  float(lst_center6[2])


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

    x_point = float(lists[0])
    y_point = float(lists[1])
    z_point = float(lists[2])

    dist1 = math.sqrt((x_center1-x_point)**2+(y_center1-y_point)**2+(z_center1-z_point)**2)
    dist2 = math.sqrt((x_center2-x_point)**2+(y_center2-y_point)**2+(z_center2-z_point)**2)
    dist3 = math.sqrt((x_center3-x_point)**2+(y_center3-y_point)**2+(z_center3-z_point)**2)
    dist4 = math.sqrt((x_center4-x_point)**2+(y_center4-y_point)**2+(z_center4-z_point)**2)
    dist5 = math.sqrt((x_center5-x_point)**2+(y_center5-y_point)**2+(z_center5-z_point)**2)
    dist6 = math.sqrt((x_center6-x_point)**2+(y_center6-y_point)**2+(z_center6-z_point)**2)

    if min(dist1,dist2,dist3,dist4,dist5,dist6) == dist1:
        print 'center1','\t',lists
    elif min(dist1,dist2,dist3,dist4,dist5,dist6) == dist2:
        print 'center2','\t',lists
    elif min(dist1,dist2,dist3,dist4,dist5,dist6) == dist3:
        print 'center3','\t',lists
    elif min(dist1,dist2,dist3,dist4,dist5,dist6) == dist4:
        print 'center4','\t',lists
    elif min(dist1,dist2,dist3,dist4,dist5,dist6) == dist5:
        print 'center5','\t',lists
    else:
            print 'center6','\t',lists
```

Reducer code:

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

curr_id = None
curr_sum_x = 0
curr_sum_y = 0
curr_sum_z = 0
id = None
point_x = []
point_y = []
point_z = []


for line in sys.stdin:
    line = line.strip()
    ln = line.split('\t')
    id = ln[0]
    x=ln[1]
    y=ln[2]
    z=ln[3]

    if curr_id == id:
        if curr_id == 'center1':
            curr_sum_x = curr_sum_x+float(x)
            curr_sum_y = curr_sum_y+float(y)
            curr_sum_z = curr_sum_z+float(z)
            point_x.append(x)
            point_y.append(y)
            point_z.append(x)
        if curr_id == 'center2':
            curr_sum_x = curr_sum_x+float(x)
            curr_sum_y = curr_sum_y+float(y)
            curr_sum_z = curr_sum_z+float(z)
            point_x.append(x)
            point_y.append(y)
            point_z.append(x)
        if curr_id == 'center3':
            curr_sum_x = curr_sum_x+float(x)
            curr_sum_y = curr_sum_y+float(y)
            curr_sum_z = curr_sum_z+float(z)
            point_x.append(x)
            point_y.append(y)
            point_z.append(x)
        if curr_id == 'center4':
            curr_sum_x = curr_sum_x+float(x)
            curr_sum_y = curr_sum_y+float(y)
            curr_sum_z = curr_sum_z+float(z)
            point_x.append(x)
            point_y.append(y)
            point_z.append(x)
        if curr_id == 'center5':
            curr_sum_x = curr_sum_x+float(x)
            curr_sum_y = curr_sum_y+float(y)
            curr_sum_z = curr_sum_z+float(z)
            point_x.append(x)
            point_y.append(y)
            point_z.append(x)
        if curr_id == 'center6':
            curr_sum_x = curr_sum_x+float(x)
            curr_sum_y = curr_sum_y+float(y)
            curr_sum_z = curr_sum_z+float(z)
            point_x.append(x)
            point_y.append(y)
            point_z.append(x)
    else:
              if curr_id:
            avgx = curr_sum_x/int(len(point_x))
            avgy = curr_sum_y/int(len(point_y))
            avgz = curr_sum_z/int(len(point_z))

        curr_id = id
        curr_sum_x = float(x)
        curr_sum_y = float(y)
        curr_sum_z = float(z)
        point_x = ln[1]
        point_y = ln[2]
        point_z = ln[3]
avgx = curr_sum_x/int(len(point_x))
avgy = curr_sum_y/int(len(point_y))
```

```
avgz = curr_sum_z/int(len(point_z))

if curr_id == 'center1':
   print 'center1','\t',avgx,'\t',avgy,'\t',avgz
if curr_id == 'center2':
   print 'center2','\t',avgx,'\t',avgy,'\t',avgz
if curr_id == 'center3':
   print 'center3','\t',avgx,'\t',avgy,'\t',avgz
if curr_id == 'center4':
   print 'center4','\t',avgx,'\t',avgy,'\t',avgz
if curr_id == 'center5':
   print 'center5','\t',avgx,'\t',avgy,'\t',avgz
if curr_id == 'center6':
   print 'center6','\t',avgx,'\t',avgy,'\t',avgz
```



```
[ec2-user@ip-172-31-24-172 ~]$ head test5.txt | python clustermapper.py
center6         0.348726395778  0.602811805627  0.0936764037165
center2         0.842607855487  0.355753709612  0.180444848535
center4         0.517641206798  0.825587915578  0.659189749735
center1         0.556070535444  0.559115945728  0.965078053403
center4         0.148482210322  0.957949948042  0.315651892536
center6         0.218322560385  0.977803627804  0.0233516726384
center4         0.0546996895523         0.683606481768  0.593465386608
center6         0.256127368257  0.756100780986  0.0111760383352
center2         0.947410239224  0.0367066081584         0.167785418555
center1         0.594984922203  0.314981567699  0.755522846654
[ec2-user@ip-172-31-24-172 ~]$
```