Ronaldlee Ejalu

CSC 555: Mining Big Data

Project, Phase 2 (due Tuesday, November 23rd)

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 and data is available at: http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/

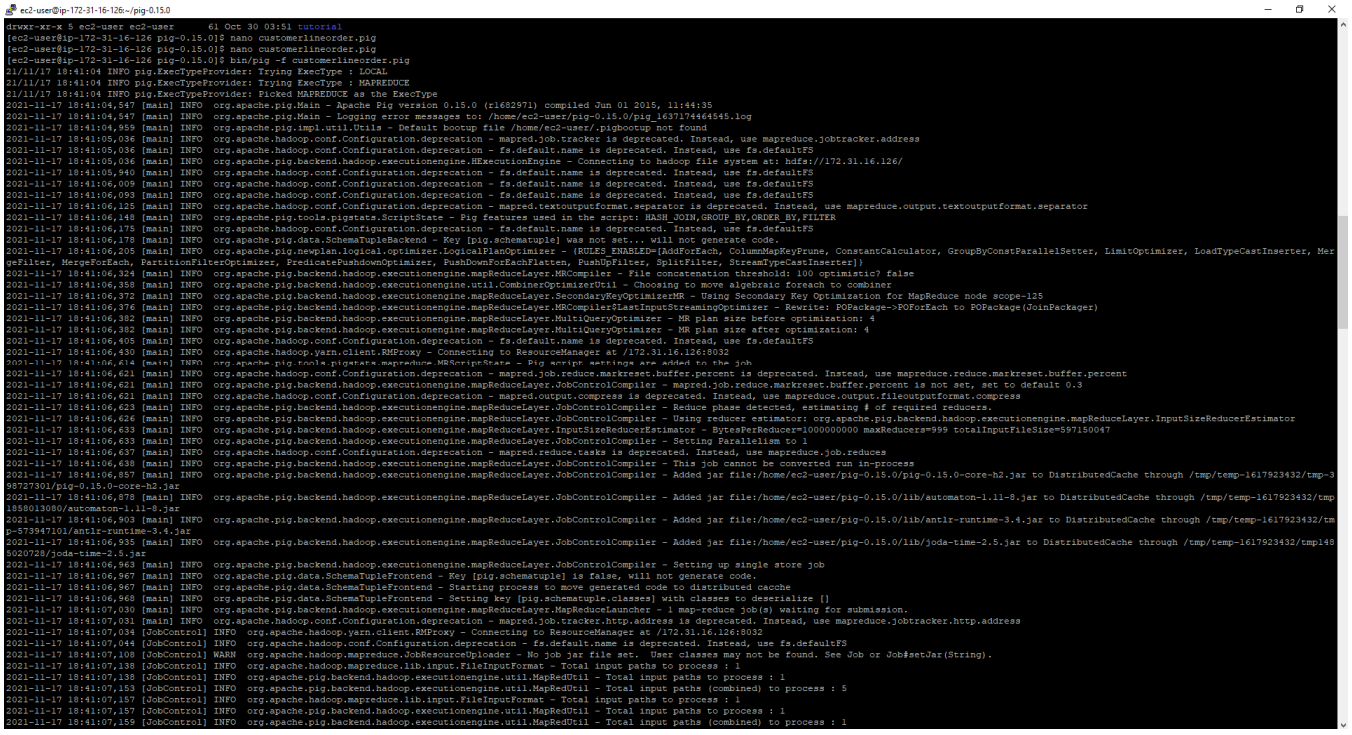
You should use your 3-node cluster for the final. Please be sure to submit all code. You should also submit the command lines you use and a screenshot of a completed run (just the last page, do not worry about capturing the entire output).

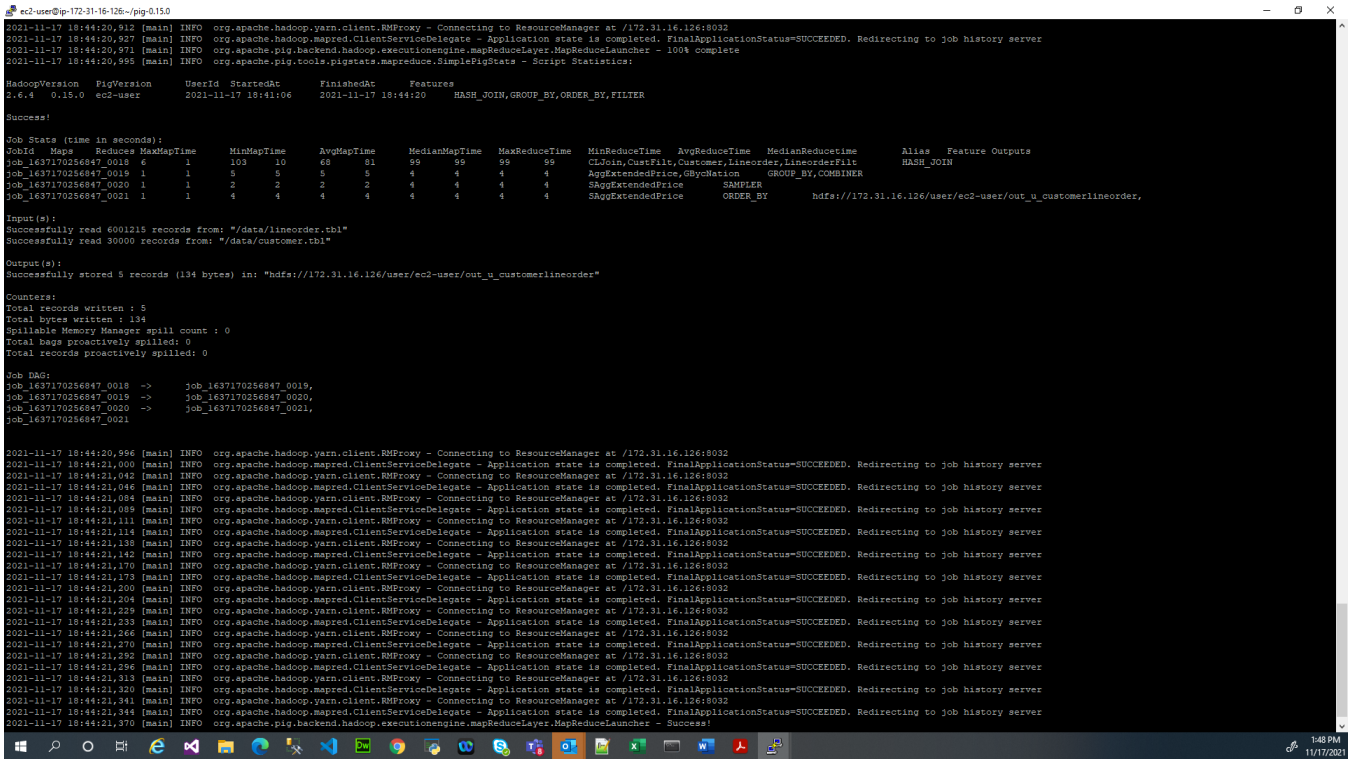
I highly recommend creating a small sample input (e.g., by running head -n 1000 lineorder.tbl > lineorder.tbl.sample, you can create a small version of lineorder with 1000 lines) and testing your code with a smaller file until you can verify that it works.

Part 1: Pig

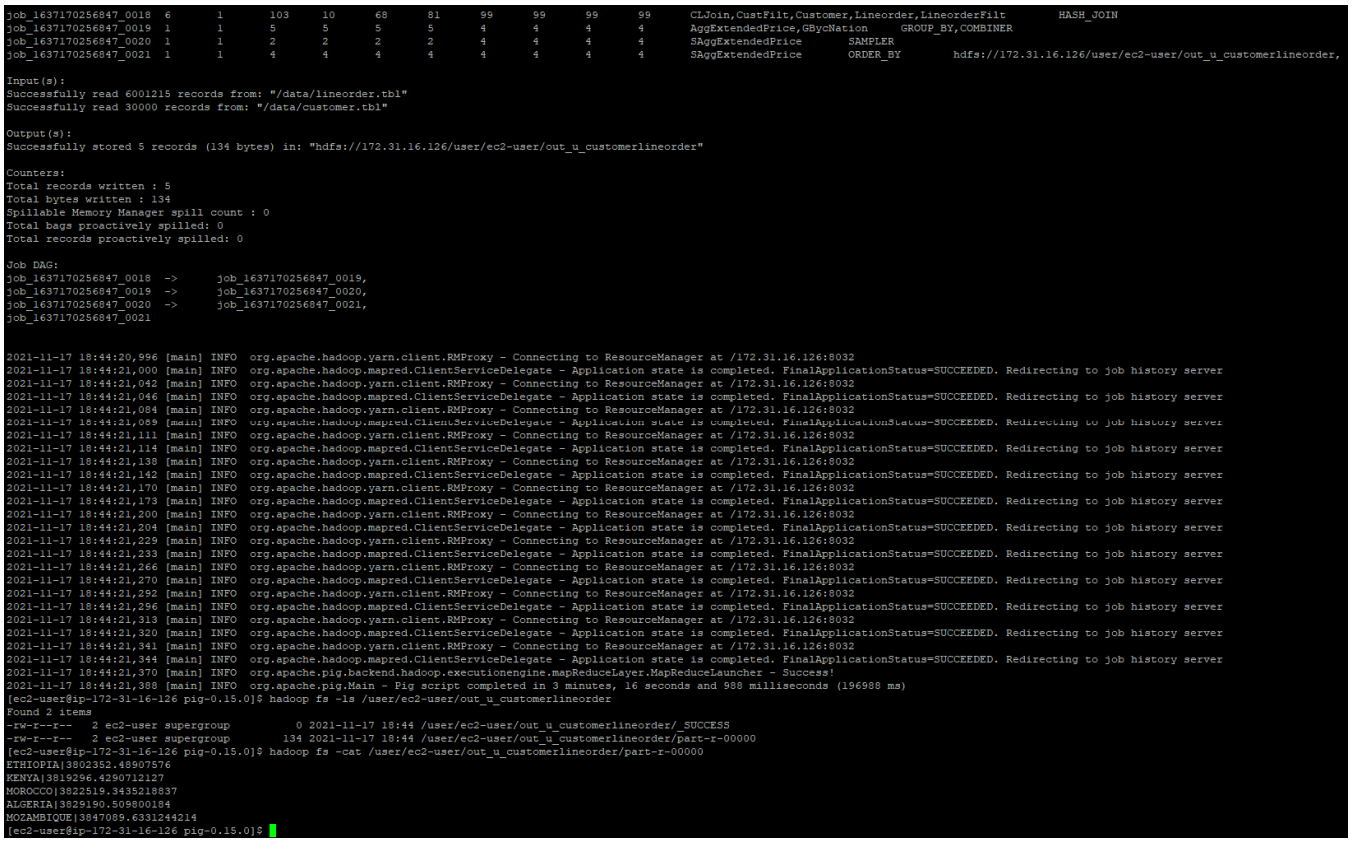
Implement the following query using Pig:

```
select c_nation, AVG(lo_extendedprice) as AVGL
from customer, lineorder
where lo_custkey = c_custkey
  and c_region = 'AFRICA'
  and lo_discount = 6 OR lo_discount > 8
group by c_nation
order by AVGL;
```





The output produced by the aggregation:



The pig script used:

```
/* Part 1

select c_nation, AVG(lo_extendedprice) as AVGL

from customer, lineorder

where lo_custkey = c_custkey

  and c_region = 'AFRICA'

  and lo_discount = 6 OR lo_discount > 8

group by c_nation

order by AVGL;


*/

Customer = LOAD '/data/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);


Lineorder = LOAD '/data/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 );

CustFilt = FILTER Customer BY c_region == 'AFRICA';

LineorderFilt = FILTER Lineorder BY lo_discount == 6 OR lo_discount > 8;

CLJoin = JOIN CustFilt BY c_custkey, LineorderFilt BY lo_custkey;

GBycNation = Group CLJoin BY c_nation;

AggExtendedPrice = FOREACH GBycNation GENERATE group, AVG(CLJoin.lo_extendedprice);

SAggExtendedPrice = ORDER AggExtendedPrice BY $1;

STORE SAggExtendedPrice INTO 'out_u_customerlineorder' USING PigStorage('|');
```

Part 2: Hadoop streaming

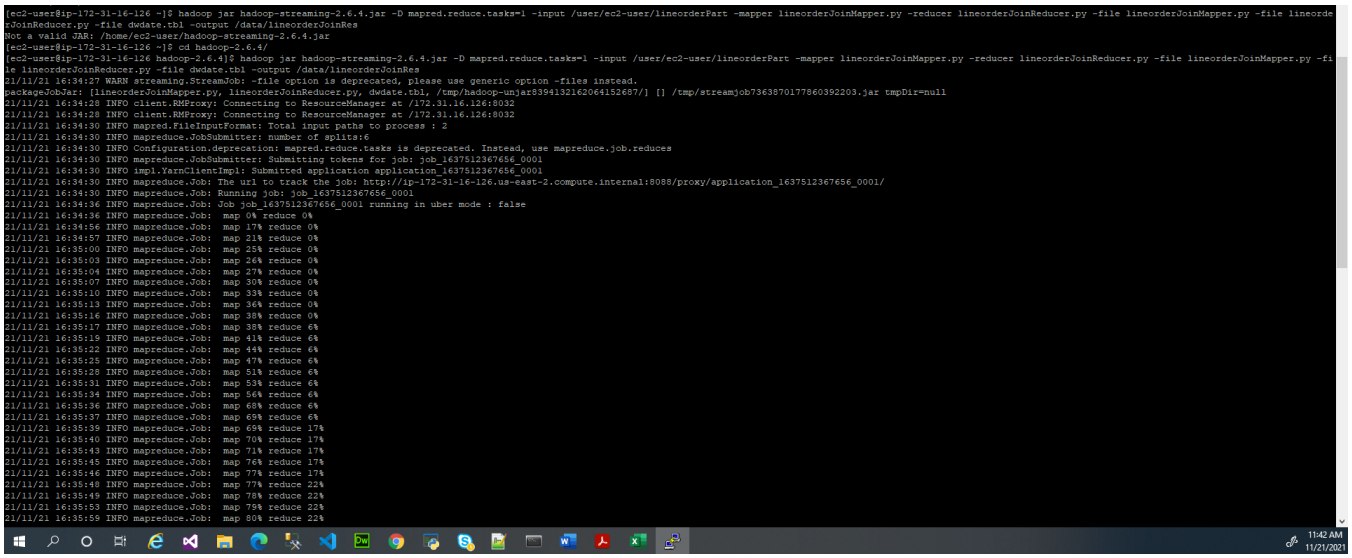
Implement the following query using Hadoop streaming:

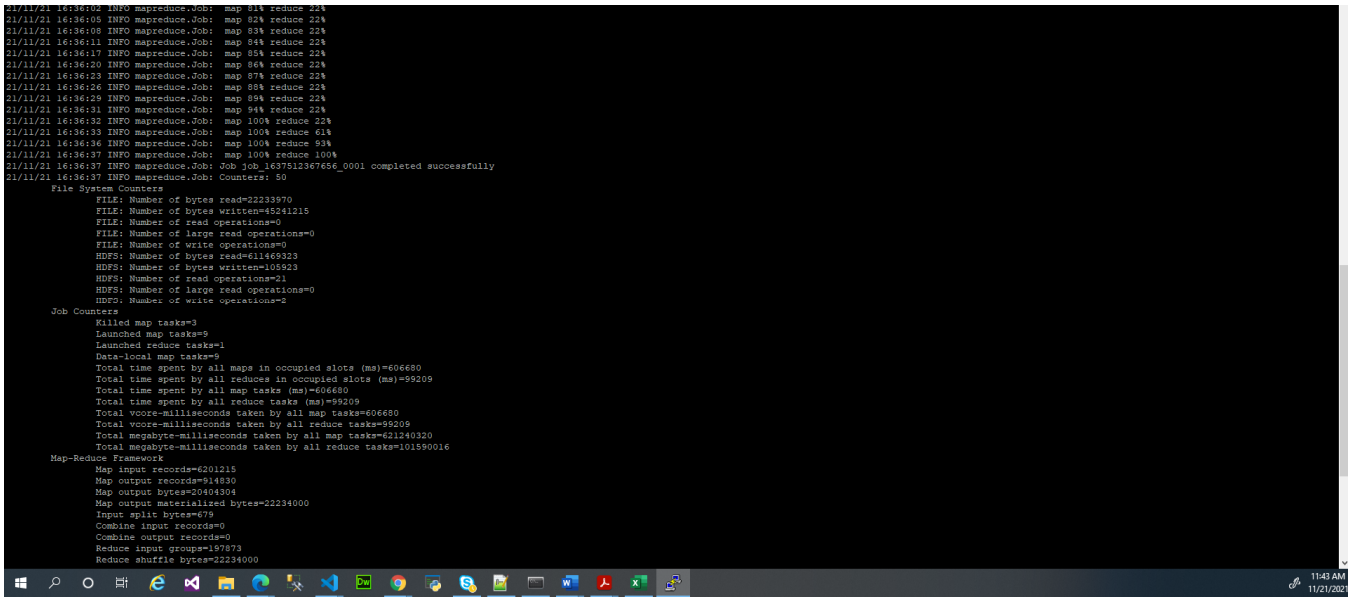
```
select sum(lo_revenue), d_year, p_brand1
from lineorder, dwdate, part
where lo_orderdate = d_datekey
  and lo_partkey = p_partkey
  and d_sellingseason = 'Fall'
  and p_brand1 between 'MFGR#2121'
  and 'MFGR#2138'
group by d_year, p_brand1
```

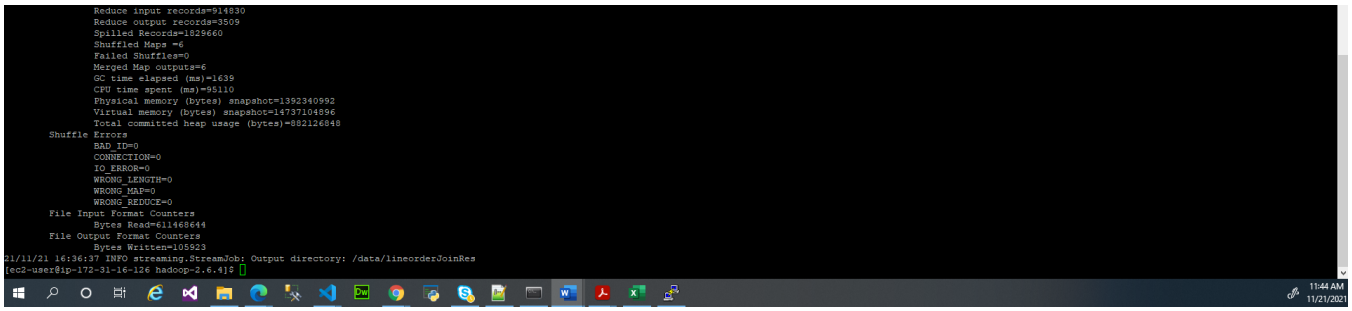
In Hadoop streaming, this will use a total of 3 passes (two joins and another one for GROUP BY). You can also choose to perform a map-side join with dwdate (only dwdate), which would result in a total of 2 passes (one join and one for GROUP BY).

Using map-side join with dwdate.

The first pass:








```
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$ hadoop fs -ls /data/lineorderJoinRes
Found 2 items
-rw-r--r--   2 ec2-user supergroup          0 2021-11-21 16:36 /data/lineorderJoinRes/_SUCCESS
-rw-r--r--   2 ec2-user supergroup     105923 2021-11-21 16:36 /data/lineorderJoinRes/part-00000
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$
```

The code of the first pass:

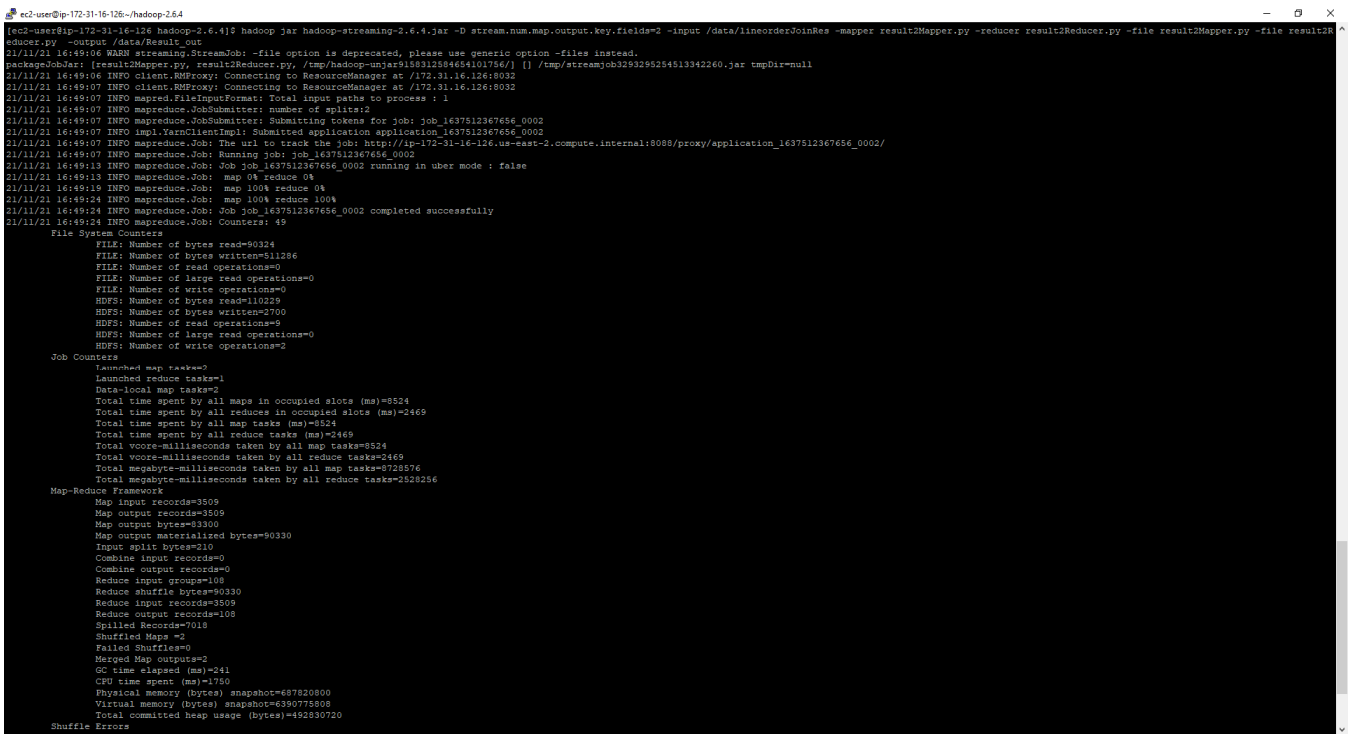
lineorderJoinMapper.py

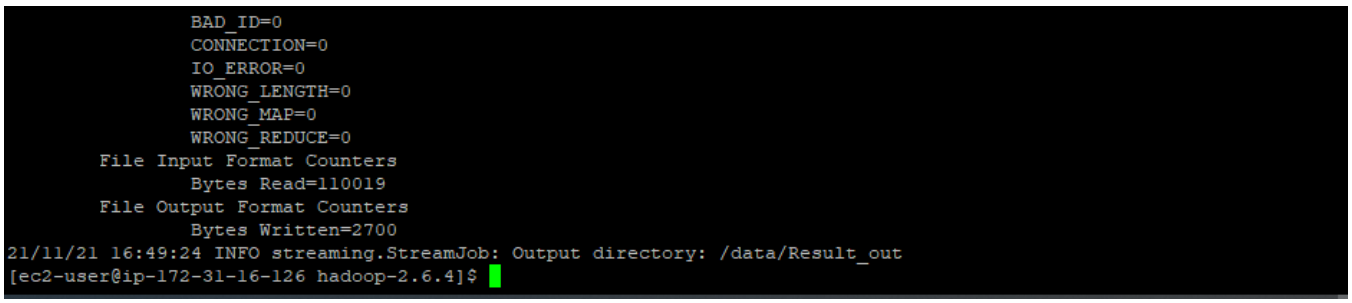
```
#!/usr/bin/python
import sys
fd = open('dwdate.tbl', 'r')
lines = fd.readlines()
fd.close()
dDict = {}

for line in lines:
    vals= line.split('|')
    if vals[12] == 'Fall':
        dDict[vals[0]] = int(vals[4])

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

   # lo_orderdate = d_datekey
   if vals[6].find('-') > 0:
       # partkey, dYear,  revenue
       if vals[5] in dDict.keys():
           print(vals[3] + '\t' + str(dDict[vals[5]]) + '\t' + vals[12] + '\t' +
'LO')
   else:
        if vals[2][:4] == 'MFGR':
            if len(vals[4]) == 9:
                if int('MFGR#2121'[-4:]) <= int(vals[4][-4:])  <=
int('MFGR#2138'[-4:]):
                    # print partKey, brandl
                    print(vals[0] + '\t' + vals[4] + '\t' + 'Part')
```

lineorderJoinReducer.py

```
#!/usr/bin/python
import sys
key = ''
currentKey = None
brandl = None
revenueL = []
dYear = None
partKey = None
for line in sys.stdin:
#for line in listOfWords:
    line = line.strip()
    vals = line.split('\t')
    key = vals[0]
    value = '\t'.join(vals[1:])
    # print(value)
    if currentKey == key:
        if value.endswith('LO'):
            partKey = vals[0]                          # assign string partKey to
the variable
            dYear =   vals[1]                        # assign string orderdate to
the variable
            revenueL.append(int(vals[2]))             # assign string revenue to
the variable
        if value.endswith('Part'):
            brandl = vals[1]                           # assign string brandl to the
variable
    else:
        if currentKey:                               # when the current key is done
            lendYear = len(dYear)       # derive the length of orderdate
            #lenRevenue = len(revenueL)            # derive the length of revenue
            lenBrandl = len(brandl)              # derive the length of brandl

            # this acts as a joins since rows must exist on both sides
            if (lendYear  * lenBrandl) > 0:
                sumOfRev = sum(revenueL)
                print(partKey + '\t' + dYear + '\t'  + brandl + '\t' + str(sumOf-
Rev))
       # reset the variables
        partKey = ''
        brandl = ''
        revenueL = []
        dYear = ''
        currentKey = key
        if value.endswith('LO'):
            partKey = vals[0]
```

Second Pass:


ec2-user@ip-172-31-16-126:~/hadoop-2.6.4
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$ hadoop jar hadoop-streaming-2.6.4.jar -D stream.num.map.output.key.fields=2 -input /data/lineorderJoinRes -mapper result2Mapper.py -reducer result2Reducer.py -file result2Mapper.py -file result2Reducer.py -output /data/Result_out
21/11/21 16:49:06 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [result2Mapper.py, result2Reducer.py, /tmp/hadoop-unjar9158312584684101756/] [] /tmp/streamjob3293295254513342260.jar tmpDir=null
21/11/21 16:49:06 INFO client.RMProxy: Connecting to ResourceManager at /172.31.16.126:8032
21/11/21 16:49:07 INFO client.RMProxy: Connecting to ResourceManager at /172.31.16.126:8032
21/11/21 16:49:07 INFO mapred.FileInputFormat: Total input paths to process : 1
21/11/21 16:49:07 INFO mapreduce.JobSubmitter: number of splits:2
21/11/21 16:49:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1637512367656_0002
21/11/21 16:49:07 INFO impl.YarnClientImpl: Submitted application application_1637512367656_0002
21/11/21 16:49:07 INFO mapreduce.Job: The url to track the job: http://ip-172-31-16-126.us-east-2.compute.internal:8088/proxy/application_1637512367656_0002/
21/11/21 16:49:07 INFO mapreduce.Job: Running job: job_1637512367656_0002
21/11/21 16:49:13 INFO mapreduce.Job: Job job_1637512367656_0002 running in uber mode : false
21/11/21 16:49:13 INFO mapreduce.Job: map 0% reduce 0%
21/11/21 16:49:19 INFO mapreduce.Job: map 100% reduce 0%
21/11/21 16:49:24 INFO mapreduce.Job: map 100% reduce 100%
21/11/21 16:49:24 INFO mapreduce.Job: Job job_1637512367656_0002 completed successfully
21/11/21 16:49:24 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=90324
FILE: Number of bytes written=511286
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=110229
HDFS: Number of bytes written=2700
HDFS: Number of read operations=9
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=2
Launched reduce tasks=1
Data-local map tasks=2
Total time spent by all maps in occupied slots (ms)=8524
Total time spent by all reduces in occupied slots (ms)=2469
Total time spent by all map tasks (ms)=8524
Total time spent by all reduce tasks (ms)=2469
Total vcore-milliseconds taken by all map tasks=8524
Total vcore-milliseconds taken by all reduce tasks=2469
Total megabyte-milliseconds taken by all map tasks=8728576
Total megabyte-milliseconds taken by all reduce tasks=2528256
Map-Reduce Framework
Map input records=3509
Map output records=3509
Map output bytes=83300
Map output materialized bytes=90330
Input split bytes=210
Combine input records=0
Combine output records=0
Reduce input groups=108
Reduce shuffle bytes=90330
Reduce input records=3509
Reduce output records=108
Spilled Records=7018
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=241
CPU time spent (ms)=1750
Physical memory (bytes) snapshot=687820800
Virtual memory (bytes) snapshot=6390775808
Total committed heap usage (bytes)=492830720
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=110019
File Output Format Counters
Bytes Written=2700
21/11/21 16:49:24 INFO streaming.StreamJob: Output directory: /data/Result_out
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$

```
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$ hadoop fs -cat /data/Result_out/part-00000
454533212       1992    MFGR#2121
626727412       1992    MFGR#2122
568103779       1992    MFGR#2123
494498814       1992    MFGR#2124
417927760       1992    MFGR#2125
523950295       1992    MFGR#2126
546897110       1992    MFGR#2127
568178054       1992    MFGR#2128
706453368       1992    MFGR#2129
497167279       1992    MFGR#2130
494035303       1992    MFGR#2131
286139666       1992    MFGR#2132
685781137       1992    MFGR#2133
574199272       1992    MFGR#2134
572281639       1992    MFGR#2135
407246217       1992    MFGR#2136
501143334       1992    MFGR#2137
536526942       1992    MFGR#2138
425635841       1993    MFGR#2121
654925335       1993    MFGR#2122
488832745       1993    MFGR#2123
589070289       1993    MFGR#2124
475208487       1993    MFGR#2125
682561567       1993    MFGR#2126
700059167       1993    MFGR#2127
452869576       1993    MFGR#2128
511023268       1993    MFGR#2129
538625429       1993    MFGR#2130
459138711       1993    MFGR#2131
412340694       1993    MFGR#2132
536534464       1993    MFGR#2133
589673422       1993    MFGR#2134
534429109       1993    MFGR#2135
427040233       1993    MFGR#2136
506915012       1993    MFGR#2137
303866109       1993    MFGR#2138
552432059       1994    MFGR#2121
576208465       1994    MFGR#2122
634137786       1994    MFGR#2123
417365059       1994    MFGR#2124
436739036       1994    MFGR#2125
497479930       1994    MFGR#2126
636189981       1994    MFGR#2127
606550539       1994    MFGR#2128
612948515       1994    MFGR#2129
714257192       1994    MFGR#2130
447910741       1994    MFGR#2131
611048520       1994    MFGR#2132
514087052       1994    MFGR#2133
732492082       1994    MFGR#2134
380981666       1994    MFGR#2135
688019183       1994    MFGR#2136
508284147       1994    MFGR#2137
478975400       1994    MFGR#2138
646081671       1995    MFGR#2121
521302756       1995    MFGR#2122
540714731       1995    MFGR#2123
571034849       1995    MFGR#2124
813760685       1995    MFGR#2125
677519553       1995    MFGR#2126
749947863       1995    MFGR#2127
660934700       1995    MFGR#2128
```

```
660934700       1995    MFGR#2128
443377761       1995    MFGR#2129
470251807       1995    MFGR#2130
508120174       1995    MFGR#2131
478192662       1995    MFGR#2132
370233404       1995    MFGR#2133
558843411       1995    MFGR#2134
499896288       1995    MFGR#2135
431291085       1995    MFGR#2136
534950631       1995    MFGR#2137
416270408       1995    MFGR#2138
543499368       1996    MFGR#2121
497679357       1996    MFGR#2122
676532684       1996    MFGR#2123
541516061       1996    MFGR#2124
739277850       1996    MFGR#2125
693312616       1996    MFGR#2126
422337225       1996    MFGR#2127
665027527       1996    MFGR#2128
479211042       1996    MFGR#2129
571945125       1996    MFGR#2130
492429863       1996    MFGR#2131
457171683       1996    MFGR#2132
630369554       1996    MFGR#2133
527186037       1996    MFGR#2134
532058058       1996    MFGR#2135
556248309       1996    MFGR#2136
566131324       1996    MFGR#2137
661863379       1996    MFGR#2138
370769746       1997    MFGR#2121
734064708       1997    MFGR#2122
609946176       1997    MFGR#2123
455111941       1997    MFGR#2124
492077784       1997    MFGR#2125
482897590       1997    MFGR#2126
627576408       1997    MFGR#2127
647258500       1997    MFGR#2128
591611461       1997    MFGR#2129
417835630       1997    MFGR#2130
493688984       1997    MFGR#2131
389320753       1997    MFGR#2132
547692372       1997    MFGR#2133
502625541       1997    MFGR#2134
456060721       1997    MFGR#2135
620443490       1997    MFGR#2136
517614449       1997    MFGR#2137
542063874       1997    MFGR#2138
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$
```

The code of the second pass:

result2Mapper.py

```
#!/usr/bin/python
import sys
for line in sys.stdin:
   line = line.strip()
   vals = line.split('\t')
   dYear = vals[1]
   brandl = vals[2]
   revenue = vals[3]
   print(dYear + '\t' + brandl + '\t' + revenue)
```

result2Reducer:

```
#!/usr/bin/python
import sys
key = ''
currentKey = None
revenueL = []
dYear = None
brandl = None
for line in sys.stdin:
    line = line.strip()
    vals = line.split('\t')
    key = vals[0] + '|' + vals[1]
    if currentKey == key:
        dYear = vals[0]
        brandl = vals[1]
        revenueL.append(int(vals[2]))
    else:
        if currentKey: #same key
            lenRevenueL = len(revenueL)
            if (lenRevenueL > 0):
                    print('%s\t%s\t%s' %(str(sum(revenueL)), dYear, brandl))
        # re-initialize the variables when the keys are not the same (new key)
before adding$
        dYear = ''
        brandl = ''
        revenueL = []
        currentKey = key
        dYear = vals[0]
        brandl = vals[1]
        revenueL.append(int(vals[2]))

# output the last key
if currentKey == key:
    lenRevenueL = len(revenueL)
    if (lenRevenueL > 0):
        print('%s\t%s\t%s' %(str(sum(revenueL)), dYear, brandl))
```

Part 3: Clustering

Using Hadoop streaming and randomly generated data (similar to what you did in Assignment6, but generate 2,100,000 rows and 6 columns of data) perform four KMeans iterations manually, using 4 centers**.** You can randomly choose the initial centers, such as by picking 4 random points from your data. For each of four KMeans

iterations, include the centers produced by your code. Please do not submit the command line four times, without the corresponding output.

Code used to generate the file:

```
#!/usr/bin/python
import numpy as np
from numpy import savetxt
import random
arr = np.random.randint(50, size = (2100000,6))
np.savetxt('numericGeneratedFile.csv', arr, fmt= '%i', delimiter = '|')
```

This would require passing a text file with cluster centers using -file option as discussed in class, 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 by averaging the points, which would conclude the iteration. At that point, the output of the reducer with new centers can be given to the next pass of the same map reduce code using the -file option (you would need to get the output from HDFS into a local file for that).

The only difference between first and subsequent iterations is that in first iteration you have to pick the initial centers. Starting from the 2nd iteration, the centers will be given to you by a previous pass of KMeans, and so on. Include the centers you computed at each iteration in your answer.

These were my initial centers I manually picked up:

```
  GNU nano 2.9.8

C1|10,43,34,13,3,33
C2|1,41,7,43,1,8
C3|49,10,42,42,15,36
C4|23,1,12,44,33,9
```

ec2-user@ip-172-31-16-126:~/hadoop-2.6.4

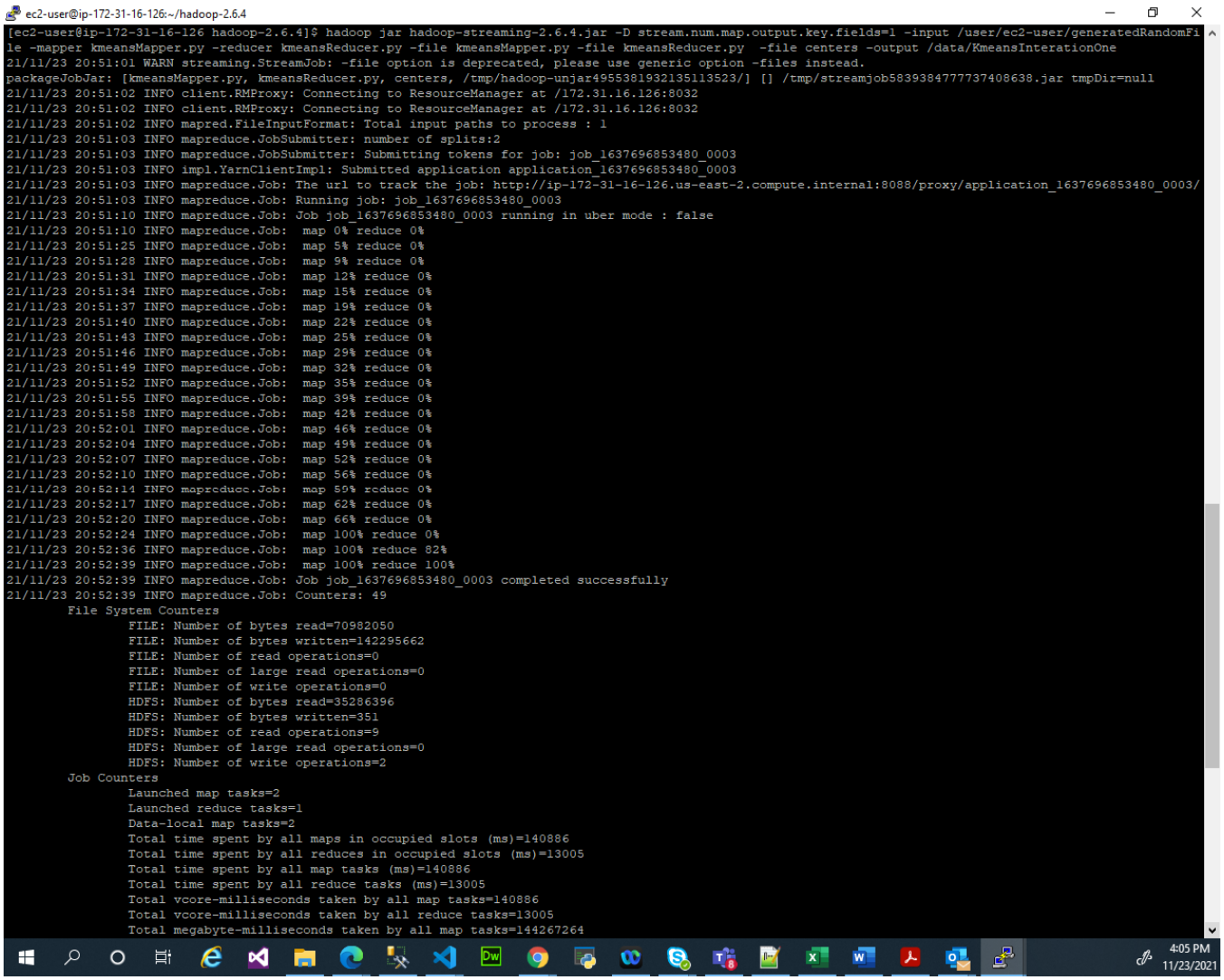
```
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$ hadoop jar hadoop-streaming-2.6.4.jar -D stream.num.map.output.key.fields=1 -input /user/ec2-user/generatedRandomFi
le -mapper kmeansMapper.py -reducer kmeansReducer.py -file kmeansMapper.py -file kmeansReducer.py  -file centers -output /data/KmeansInterationOne
21/11/23 20:33:27 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [kmeansMapper.py, kmeansReducer.py, centers, /tmp/hadoop-unjar2715302473109111895/] [] /tmp/streamjob9771498844031910 45.jar tmpDir=null
21/11/23 20:33:28 INFO client.RMProxy: Connecting to ResourceManager at /172.31.16.126:8032
21/11/23 20:33:28 INFO client.RMProxy: Connecting to ResourceManager at /172.31.16.126:8032
21/11/23 20:33:28 INFO mapred.FileInputFormat: Total input paths to process : 1
21/11/23 20:33:28 INFO mapreduce.JobSubmitter: number of splits:2
21/11/23 20:33:29 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1637696853480_0002
21/11/23 20:33:29 INFO impl.YarnClientImpl: Submitted application application_1637696853480_0002
21/11/23 20:33:29 INFO mapreduce.Job: The url to track the job: http://ip-172-31-16-126.us-east-2.compute.internal:8088/proxy/application_1637696853480_0002/
21/11/23 20:33:29 INFO mapreduce.Job: Running job: job_1637696853480_0002
21/11/23 20:33:36 INFO mapreduce.Job: Job job_1637696853480_0002 running in uber mode : false
21/11/23 20:33:36 INFO mapreduce.Job:  map 0% reduce 0%
21/11/23 20:33:51 INFO mapreduce.Job:  map 5% reduce 0%
21/11/23 20:33:54 INFO mapreduce.Job:  map 7% reduce 0%
21/11/23 20:33:55 INFO mapreduce.Job:  map 8% reduce 0%
21/11/23 20:33:58 INFO mapreduce.Job:  map 12% reduce 0%
21/11/23 20:34:01 INFO mapreduce.Job:  map 15% reduce 0%
21/11/23 20:34:04 INFO mapreduce.Job:  map 18% reduce 0%
21/11/23 20:34:07 INFO mapreduce.Job:  map 22% reduce 0%
21/11/23 20:34:10 INFO mapreduce.Job:  map 25% reduce 0%
21/11/23 20:34:13 INFO mapreduce.Job:  map 28% reduce 0%
21/11/23 20:34:16 INFO mapreduce.Job:  map 32% reduce 0%
21/11/23 20:34:19 INFO mapreduce.Job:  map 35% reduce 0%
21/11/23 20:34:22 INFO mapreduce.Job:  map 39% reduce 0%
21/11/23 20:34:25 INFO mapreduce.Job:  map 42% reduce 0%
21/11/23 20:34:28 INFO mapreduce.Job:  map 46% reduce 0%
21/11/23 20:34:31 INFO mapreduce.Job:  map 49% reduce 0%
21/11/23 20:34:34 INFO mapreduce.Job:  map 52% reduce 0%
21/11/23 20:34:37 INFO maprcduce.Job:  map 56% rcduce 0%
21/11/23 20:34:40 INFO mapreduce.Job:  map 59% reduce 0%
21/11/23 20:34:43 INFO mapreduce.Job:  map 63% reduce 0%
21/11/23 20:34:46 INFO mapreduce.Job:  map 66% reduce 0%
21/11/23 20:34:49 INFO mapreduce.Job:  map 83% reduce 0%
21/11/23 20:34:50 INFO mapreduce.Job:  map 100% reduce 0%
21/11/23 20:35:03 INFO mapreduce.Job:  map 100% reduce 82%
21/11/23 20:35:06 INFO mapreduce.Job:  map 100% reduce 100%
21/11/23 20:35:07 INFO mapreduce.Job: Job job_1637696853480_0002 completed successfully
21/11/23 20:35:07 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=70982050
                FILE: Number of bytes written=142295656
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=35286396
                HDFS: Number of bytes written=351
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=140576
                Total time spent by all reduces in occupied slots (ms)=14141
                Total time spent by all map tasks (ms)=140576
                Total time spent by all reduce tasks (ms)=14141
                Total vcore-milliseconds taken by all map tasks=140576
```

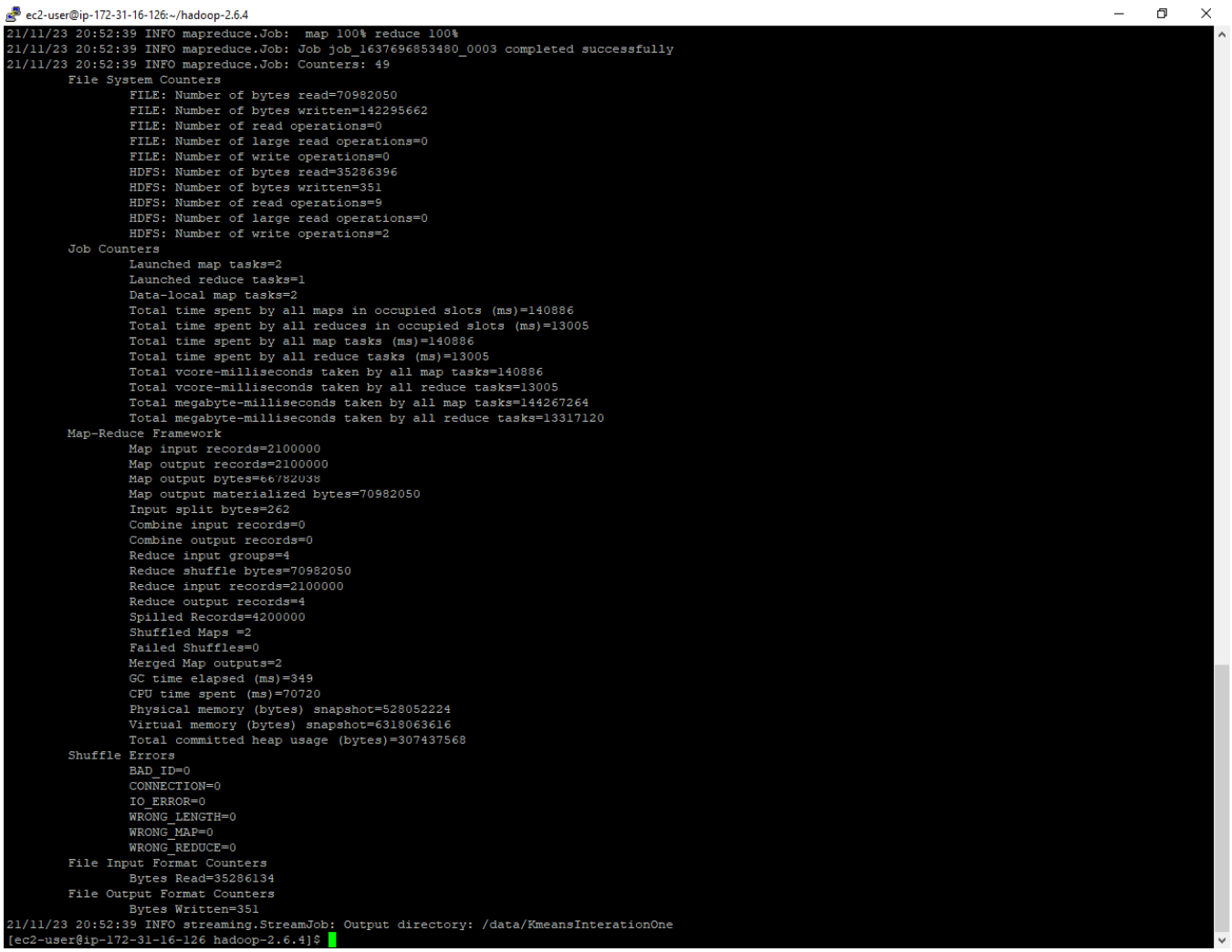

```
                Total vcore-milliseconds taken by all reduce tasks=14141
                Total megabyte-milliseconds taken by all map tasks=143949824
                Total megabyte-milliseconds taken by all reduce tasks=14480384
        Map-Reduce Framework
                Map input records=2100000
                Map output records=2100000
                Map output bytes=66782038
                Map output materialized bytes=70982050
                Input split bytes=262
                Combine input records=0
                Combine output records=0
                Reduce input groups=4
                Reduce shuffle bytes=70982050
                Reduce input records=2100000
                Reduce output records=4
                Spilled Records=4200000
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=368
                CPU time spent (ms)=66960
                Physical memory (bytes) snapshot=524685312
                Virtual memory (bytes) snapshot=6317883392
                Total committed heap usage (bytes)=307437568
        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=35286134
        File Output Format Counters
                Bytes Written=351
21/11/23 20:35:07 INFO streaming.StreamJob: Output directory: /data/KmeansInterationOne
```

This produced the converged centers:

```
  GNU nano 2.9.8

C1|19.613449668,32.4283702757,27.6826997018,15.9880729702,20.7611762322,28.2497025387
C2|15.0935707254,35.6426219491,13.5186134401,35.0165936839,15.5087296469,15.2288661077
C3|37.3543120402,19.8140405032,32.6988393981,29.4433680194,23.0440194894,30.9909670138
C4|23.5509704159,15.8742777889,18.0121552547,27.9935796908,32.1852957823,18.1514565082
```

2nd run:





The centers in the /data/kmeansInterationOne:

```
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$ hadoop fs -ls /data/KmeansInterationOne
Found 2 items
-rw-r--r--   2 ec2-user supergroup          0 2021-11-23 20:52 /data/KmeansInterationOne/_SUCCESS
-rw-r--r--   2 ec2-user supergroup        351 2021-11-23 20:52 /data/KmeansInterationOne/part-00000
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$ hadoop fs -cat /data/KmeansInterationOne/part-00000
C1|19.1910927966,32.0075953507,28.1658619567,13.6250387157,22.1795999352,28.6692429962
C2|17.6719763378,34.5076093348,15.881441222,34.9480941594,16.3698414262,17.1947983728
C3|36.7570306444,20.2199925086,32.6358695451,29.4368920157,22.7028289466,30.8688330039
C4|22.9581189304,14.6212234279,18.0037413801,25.8194487316,33.3603766481,18.3682410086
[ec2-user@ip-172-31-16-126 hadoop-2.6.4]$
```

The code used for the KmeansMapper.py and kmeansReducer.py

ec2-user@ip-172-31-16-126:~/hadoop-2.6.4

GNU nano 2.9.8 kmeansMapper.py

```
#!/usr/bin/python
import sys
import math
fd = fd = open('centers', 'r')
lines = fd.readlines()
fd.close()
centroids = []
dDict = {}

for line in lines: # loop through the lines and populate the centers  dictionary, dDict
    line = line.strip()
    vals = line.split('|')
    dDict[vals[0]] = vals[1]

# for line in listOfWords:
for line in sys.stdin:
    line = line.strip()
    record = line.split(',')
    minDist = 200000000000000
    index = -1
    for key, value in dDict.items():
        try:
            cent = dDict[key]
            centEl = cent.split(',')
            record[0] = float(record[0])
            record[1] = float(record[1])
            record[2] = float(record[2])
            record[3] = float(record[3])
            record[4] = float(record[4])
            record[5] = float(record[5])
        except ValueError: # ignore any errors if they any empty spaces
            continue
        distEuclid = math.sqrt(math.pow(record[0] - float(centEl[0]),2) + math.pow(record[1] - float(centEl[1]), 2) + math.pow(record[2] - float(centEl[2]), 2) \
             + math.pow(record[3] - float(centEl[3]), 2) + math.pow(record[4] - float(centEl[4]), 2) + math.pow(record[5] - float(centEl[5]), 2))

        # determine the point which is closer to the centroid
        if distEuclid <= minDist:
            minDist = distEuclid
            index = key
            points = str(record[0]) + '\t' + str(record[1]) + '\t' + str(record[2]) + '\t' + str(record[3]) + '\t' + str(record[4]) + '\t' + str(record[5])
    print('%s\t%s' %(index, points))
```

KmeansMapper.py:

```
#!/usr/bin/python
import sys
import math
fd = fd = open('centers', 'r')
lines = fd.readlines()
fd.close()
centroids = []
dDict = {}

for line in lines: # loop through the lines and populate the centers  dictionary,
dDict
    line = line.strip()
    vals = line.split('|')
    dDict[vals[0]] = vals[1]

# for line in listOfWords:
for line in sys.stdin:
    line = line.strip()
    record = line.split(',')
    minDist = 200000000000000
    index = -1
    for key, value in dDict.items():
        try:
            cent = dDict[key]
            centEl = cent.split(',')
            record[0] = float(record[0])
            record[1] = float(record[1])
            record[2] = float(record[2])
            record[3] = float(record[3])
            record[4] = float(record[4])
            record[5] = float(record[5])
        except ValueError: # ignore any errors if there are any empty spaces.
            continue
        distEuclid = math.sqrt(math.pow(record[0] - float(centEl[0]),2) +
math.pow(record[1] - float(centEl[1]), 2) + math.pow(record[2] -
float(centEl[2]), 2) \
             + math.pow(record[3] - float(centEl[3]), 2) + math.pow(record[4] -
float(centEl[4]), 2) + math.pow(record[5] - float(centEl[5]), 2))

        # determine the point which is closer to the centroid
        if distEuclid <= minDist:
            minDist = distEuclid
            index = key
```

kmeansReducer.py

```
#!/usr/bin/python
import sys
currentKey = None
sumA = 0
sumB = 0
sumC = 0
sumD = 0
sumE = 0
sumF = 0
cnt = 0
centerKey = None

for line in sys.stdin:
# for line in listOfWords:
    key, a, b, c, d, e, f = line.strip().split('\t')
    try:
        a = float(a)
        b = float(b)
        c = float(c)
        d = float(d)
        e = float(e)
        f = float(f)
    except ValueError: # if a value wasn't a number, so silently ignore/discard
this line.
        continue
    if currentKey == key:
        sumA += a
        sumB += b
        sumC += c
        sumD += d
        sumE += e
        sumF += f
        cnt += 1
        centerKey = key
    else:
        if currentKey:
            numericStr = str(sumA/cnt) + ',' + str(sumB/cnt) + ',' +
str(sumC/cnt) + ',' + str(sumD/cnt) + ',' + str(sumE/cnt) + ',' + str(sumF/cnt)
            print('%s|%s' %(centerKey, numericStr))
        # re-initialize the variables when the keys are not the same before
adding.
        sumA = 0
        sumB = 0
        sumC = 0
        sumD = 0
```

ec2-user@ip-172-31-16-126:~/hadoop-2.6.4

GNU nano 2.9.8 kmeansReducer.py

```
#!/usr/bin/python
import sys
currentKey = None
sumA = 0
sumB = 0
sumC = 0
sumD = 0
sumE = 0
sumF = 0
cnt = 0
centerKey = None

for line in sys.stdin:
# for line in listOfWords:
    key, a, b, c, d, e, f = line.strip().split('\t')
    try:
        a = float(a)
        b = float(b)
        c = float(c)
        d = float(d)
        e = float(e)
        f = float(f)
    except ValueError: # if a value wasn't a number, so silently ignore/discard this line.
        continue
    if currentKey == key:
        sumA += a
        sumB += b
        sumC += c
        sumD += d
        sumE += e
        sumF += f
        cnt += 1
        centerKey = key
    else:
        if currentKey:
            numericStr = str(sumA/cnt) + ',' + str(sumB/cnt) + ',' + str(sumC/cnt) + ',' + str(sumD/cnt) + ',' + str(sumE/cnt) + ',' + str(sumF/cnt)
            print('%s|%s' %(centerKey, numericStr))
        # re-initialize the variables when the keys are not the same before adding.
        sumA = 0
        sumB = 0
        sumC = 0
        sumD = 0
        sumE = 0
        sumF = 0
        cnt = 0
        centerKey = ''
        currentKey = key
        sumA = a
        sumB = b
        sumC = c
        sumD = d
        sumE = e
        sumF = f
        cnt = 1
        centerKey = key
# print the last rows
if currentKey == key:
    numericStr = str(sumA/cnt) + ',' + str(sumB/cnt) + ',' + str(sumC/cnt) + ',' + str(sumD/cnt) + ',' + str(sumE/cnt) + ',' + str(sumF/cnt)
    print('%s|%s' %(centerKey, numericStr))
```

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.