Alex Teboul
CSC 555 Mining Big Data
Assignment 4

Due Friday, March 6th

1) Consider a Hadoop job that will result in 87 blocks of output to HDFS. Suppose that writing an output block to HDFS takes 1 minute. The HDFS replication factor is set to 3 (for simplicity, we charge reducers for the cost of writing replicated blocks).

a) How long will it take for the reducer to write the job output on a 5-node Hadoop cluster? (ignoring the cost of Map processing, but counting replication cost in the output writing).

- 87 blocks / 5 nodes = 17.4 blocks per node
- 17.4 can round up to ~18 blocks per node
- 18 blocks per node * 3 replication-factor = 54 blocks per node
- 18 blocks per node * 1 min per block per node = 54 min
- **~ 54min**

b) How long will it take for reducer(s) to write the job output to 10 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 1)

- 87 blocks / 10 nodes = 8.7 blocks per node
- 8.7 can round up to ~ 9 blocks per node
- 9 blocks per node * 1 replication-factor = 9 blocks per node
- 9 blocks per node * 1 min per block per node = 9 min
- **~ 9 min**

c) How long will it take for reducer(s) to write the job output to 10 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 3)

- 87 blocks / 10 nodes = 8.7 blocks per node
- 8.7 can round up to ~ 9 blocks per node
- 9 blocks per node * 3 replication-factor = 27 blocks per node
- 27 blocks per node * 1 min per block per node = 27 min
- **~ 27 min**

d) How long will it take for reducer(s) to write the job output to 100 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 1)

- 87 blocks / 100 nodes = 0.87 blocks per node
- 0.87 can round up to ~ 1 blocks per node
- 1 blocks per node * 1 replication-factor = 1 blocks per node
- 1 blocks per node * 1 min per block per node = 1 min
- **~ 1 min**

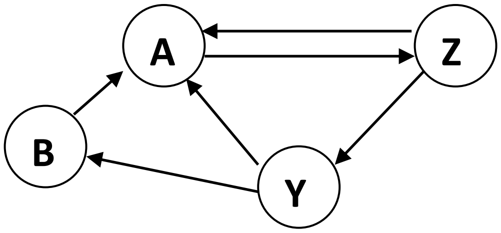
e) How long will it take for reducer(s) to write the job output to 100 Hadoop worker nodes? (Assume that data is distributed evenly and replication factor is set to 3)

- 87 blocks / 100 nodes = 0.87 blocks per node
- 0.87 can round up to ~ 1 blocks per node
- 1 blocks per node * 3 replication-factor = 3 blocks per node
- 3 blocks per node * 1 min per block per node = 3 min
- **~ 3 min**

You can ignore the network transfer costs as well as the possibility of node failure.

2)

a) Consider the following graph



Compute the page rank for the nodes in this graph. If you are multiplying matrices manually, you may stop after 5 steps. If you use a tool (e.g., Matlab, website, etc.) for matrix multiplication, you should get your answer to converge.
*A gets all of B, and a half from Y and Z
*B gets a half from Y
*Y gets a half from Z
*Z gets all of A

Step 1: (take transition matrix and multiply by vector→ calculate rank)

transition	A	B	Y	Z		V		Rank
A	0	1	1/2	1/2	x	1/4	=	1/2
B	0	0	1/2	0		1/4		1/8
Y	0	0	0	1/2		1/4		1/8
Z	1	0	0	0		1/4		1/4

Step 2: (take rank step 1 and use as vector → recalculate rank … repeat until V=Rank)

transition	A	B	Y	Z		V		Rank
A	0	1	1/2	1/2	x	1/2	=	5/16
B	0	0	1/2	0		1/8		1/16
Y	0	0	0	1/2		1/8		1/8
Z	1	0	0	0		1/4		1/2

Step 3: (… repeat until V=Rank)

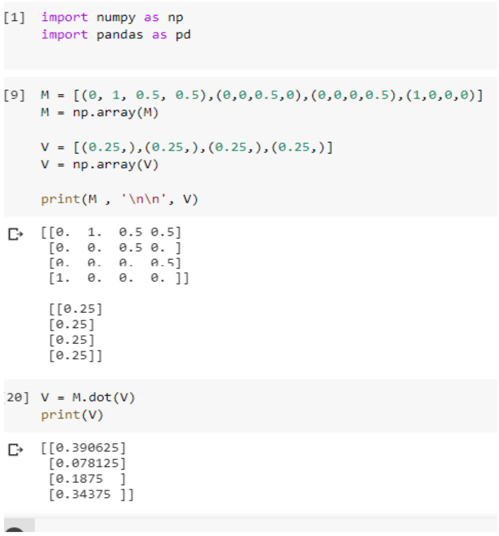
transition	A	B	Y	Z		V		Rank
A	0	1	1/2	1/2	x	5/16	=	3/8
B	0	0	1/2	0		1/16		1/16
Y	0	0	0	1/2		1/8		1/4
Z	1	0	0	0		1/2		5/16

Step 4: (… repeat until V=Rank)

transition	A	B	Y	Z		V		Rank
A	0	1	1/2	1/2	x	3/8	=	11/32
B	0	0	1/2	0		1/16		1/8
Y	0	0	0	1/2		1/4		5/32
Z	1	0	0	0		5/16		3/8

Step 5: (… repeat until V=Rank)

transition	A	B	Y	Z		V		Rank
A	0	1	1/2	1/2	x	11/32	=	25/64
B	0	0	1/2	0		1/8		5/64
Y	0	0	0	1/2		5/32		3/16
Z	1	0	0	0		3/8		11/32



```
[1]  import numpy as np
     import pandas as pd
```

```
[9]  M = [(0, 1, 0.5, 0.5),(0,0,0.5,0),(0,0,0,0.5),(1,0,0,0)]
     M = np.array(M)

     V = [(0.25,),(0.25,),(0.25,),(0.25,)]
     V = np.array(V)

     print(M , '\n\n', V)
```

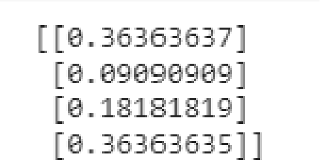
```
[[0.  1.  0.5 0.5]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  0.5]
 [1.  0.  0.  0. ]]

 [[0.25]
 [0.25]
 [0.25]
 [0.25]]
```

```
20]  V = M.dot(V)
     print(V)
```

```
[[0.390625]
 [0.078125]
 [0.1875  ]
 [0.34375 ]]
```

← Step 5 Output

… I tried this in python at it ended up with a rank that seems to be right (order A, B, Y, Z):



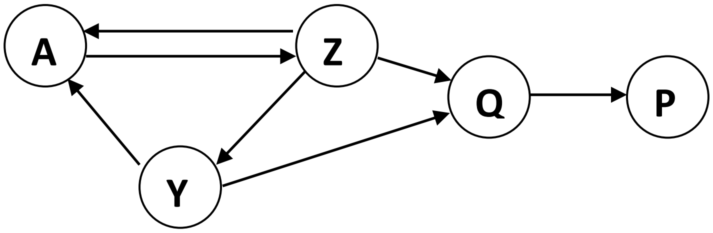
```
[[0.36363637]
 [0.09090909]
 [0.18181819]
 [0.36363635]]
```

← After about 20-30ish iterations it starts to converge

+ **Final Page Rank:**

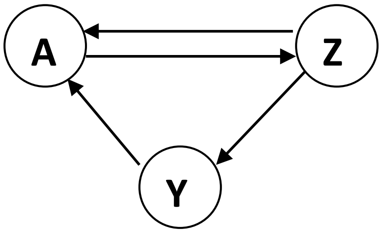
transition	A	B	Y	Z		V		Rank
A	0	1	1/2	1/2	x	4/11	=	4/11
B	0	0	1/2	0		1/11		1/11
Y	0	0	0	1/2		2/11		2/11
Z	1	0	0	0		4/11		4/11

b) Now consider a dead-end node Q and P:



What is the page rank of Q?

* Following the process from Lecture 7, start with the dead end cut off:



Step 1: (take transition matrix and multiply by vector→ calculate rank)

transition	A	Y	Z		V		Rank
A	0	1	1/2	x	1/3	=	1/2
Y	0	0	1/2		1/3		1/3
Z	1	0	0		1/3		1/6

….

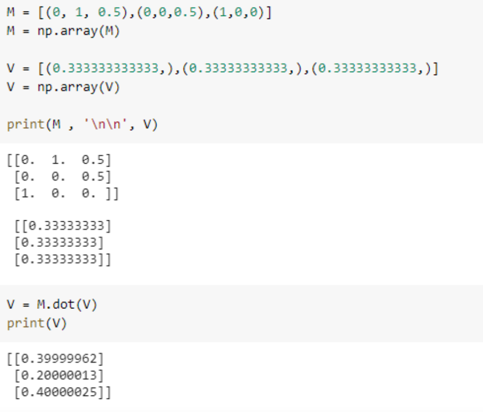
Step 30: it has converged

transition	A	Y	Z		V		Rank
A	0	1	1/2	x	2/5	=	2/5
Y	0	0	1/2		1/5		1/5
Z	1	0	0		2/5		2/5

* Next, to find the page rank of Q, we plug into a formula the rank according to the ranks of the nodes pointing to Q:

- PageRank Q = (1/3) (Rank Z) + (1/2) (Rank Y)
- PageRank Q = (1/3) (2/5) + (1/2) (1/5)
- **PageRank Q = 7/30** ~0.23333

Python code for dead-end cut off page calculation:



```
M = [(0, 1, 0.5),(0,0,0.5),(1,0,0)]
M = np.array(M)

V = [(0.333333333333,),(0.33333333333,),(0.33333333333,)]
V = np.array(V)

print(M , '\n\n', V)
```

```
[[0.  1.  0.5]
 [0.  0.  0.5]
 [1.  0.  0. ]]

 [[0.33333333]
 [0.33333333]
 [0.33333333]]
```

```
V = M.dot(V)
print(V)
```

```
[[0.39999962]
 [0.20000013]
 [0.40000025]]
```

← After about 20-30ish iterations it starts to converge

c) Exercise 5.1.6 from Mining of Massive Datasets

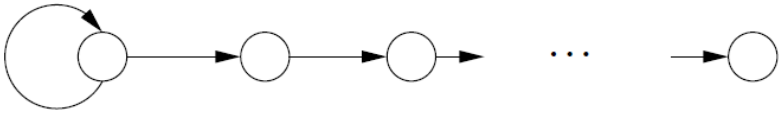
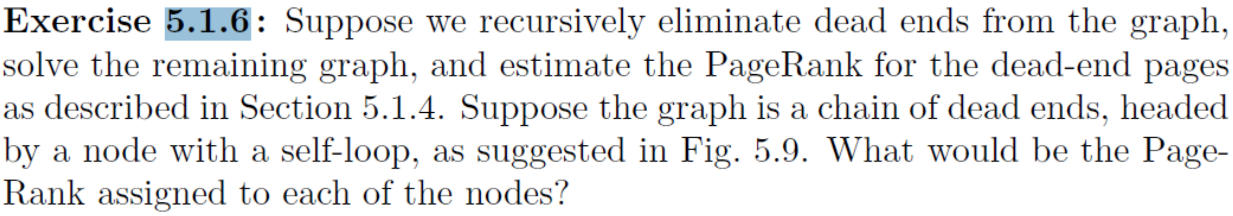


Figure 5.9: A chain of dead ends



Exercise 5.1.6 : Suppose we recursively eliminate dead ends from the graph, solve the remaining graph, and estimate the PageRank for the dead-end pages as described in Section 5.1.4. Suppose the graph is a chain of dead ends, headed by a node with a self-loop, as suggested in Fig. 5.9. What would be the PageRank assigned to each of the nodes?

- Going off of the same process as before, if we start with the node that loops to itself, that has a page rank of 1. The second node then gets a rank of (1/2) (1) = 1/2 . The Third node gets a rank of (1) (1/2) = 1/2 because node 2 point only to it. This continues such that all nodes going to the right all keep that rank of 1/2.
- **Page Ranks by Node:** 1 → 1/2 → 1/2 → …→ 1/2

3) Given the input data [(1pm, $6), (2pm, $15), (3pm, $15), (4pm, $20), (5pm, $10), (6pm, $20), (7pm, $20), (8pm, $24), (9pm, $23), (10pm, $30), (11pm, $30), (12am, $40)].

a) What will the Hive query "compute average price" return? (yes, this question is as obvious as it seems, asked for comparison with part-b)

- 6+15+15+ 20+10+20+ 20+24+23+ 30+30+40 = 253
- 253/12 = 21.08
- **Average price = $21.08**

b) What will a Storm query "compute average price per each 3 hour window" return? (tumbling, i.e., non-overlapping window of tuples, as many as you can fit). For example, the first window would 1pm-4pm. Second window would be 4pm-7pm.

- **1) 1pm-4pm** (1pm, 2pm, 3pm)
    - 6+15+15 = $36 → $36/3
    - **$12.00**
- **2) 4pm-7pm** (4pm, 5pm, 6pm)
    - 20+10+20 = $50 → $50/3
    - **$16.67**
- **3) 7pm-10pm** (7pm, 8pm, 9pm)
    - 20+24+23 = $67 → $67/3
    - **$22.33**
- **4) 10pm-1am** (10pm, 11pm, 12am)
    - 30+30+40 = $100 → $100/3
    - **$33.33**
- *(self-note: $21.08 avg across windows – all points included)*

c) What will a Storm query "compute average price per each 3 hour window" return? (sliding, i.e. overlapping window of tuples, moving the window forward 2 hours each time). First window is 1pm-4pm, second window is 3pm-6pm

- **1) 1pm-4pm** (1pm, 2pm, 3pm)
    - 6+15+15 = $36 → $36/3
    - **$12.00**
- **2) 3pm-6pm** (3pm, 4pm, 5pm)
    - 15+20+10 = $45 → $45/3
    - **$15**
- **3) 5pm-8pm** (5pm, 6pm, 7pm)
    - 10+20+20 = $50 → $50/3
    - **$16.67**
- **4) 7pm-10pm** (7pm, 8pm, 9pm)
    - 20+24+23 = $67 → $67/3
    - **$22.33**
- **5) 9pm-12am** (9pm, 10pm, 11pm)
    - 23+30+30 = $83 → $83/3
    - **$27.67**
- *(self-note: $18.73 avg across windows– missing 12am data point and sliding)*

Note, when Storm does not have a full window, you cannot output anything until the window fills with data.

4) Run another custom MapReduce job, implementing a solution for the following query:

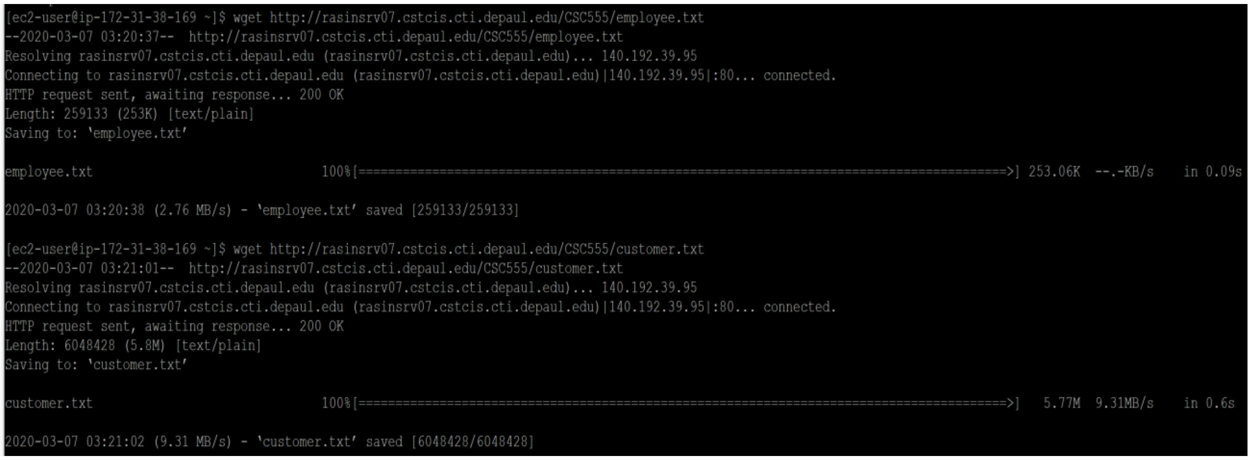
   For Employee(EID, EFirst, ELast, Phone) and Customer(CID, CFirst, CLast, Address), find everyone with the same name using MapReduce:

   ```
   SELECT EFirst, ELast, COUNT(*)
   FROM Employee, Customer
   WHERE EFirst = CFirst AND ELast = CLast;
   GROUP BY EFirst, ELast
   ```

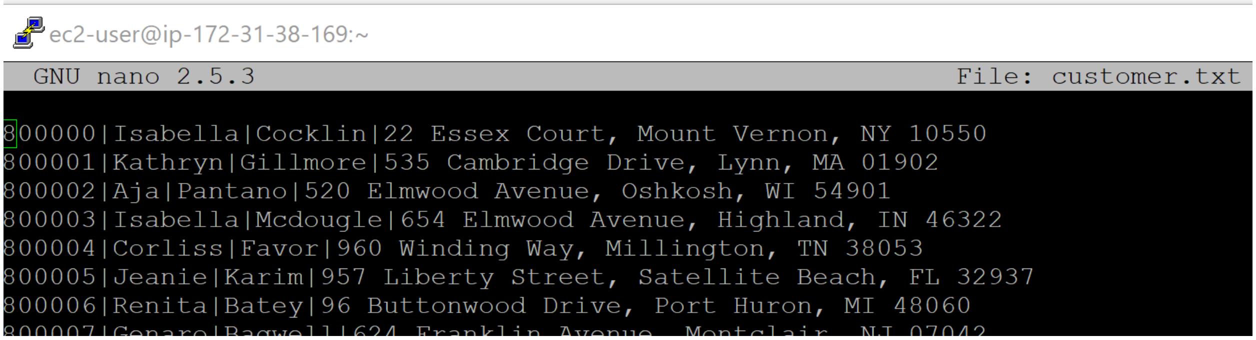
   *Needs to include GROUP BY ^

   You can use this input data:

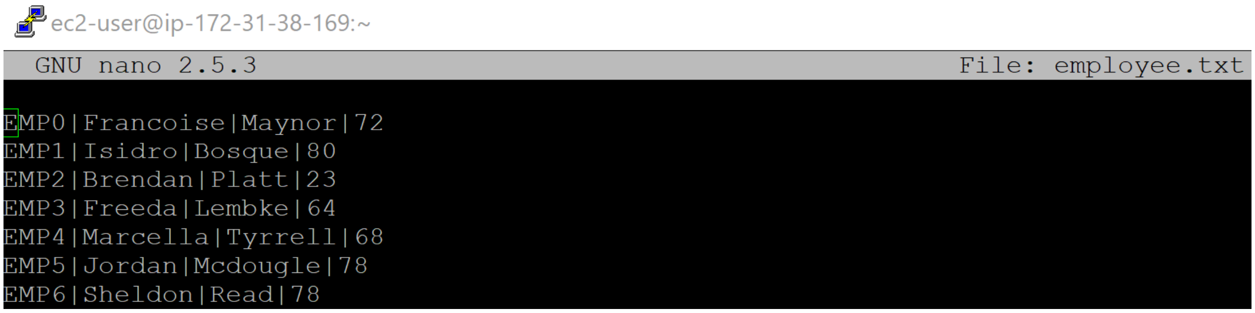
   http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/employee.txt
   http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/customer.txt



```
[ec2-user@ip-172-31-38-169 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/employee.txt
--2020-03-07 03:20:37--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/employee.txt
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.95|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 259133 (253K) [text/plain]
Saving to: 'employee.txt'

employee.txt                    100%[===========================================================================>] 253.06K  --.-KB/s    in 0.09s

2020-03-07 03:20:38 (2.76 MB/s) - 'employee.txt' saved [259133/259133]

[ec2-user@ip-172-31-38-169 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/customer.txt
--2020-03-07 03:21:01--  http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/customer.txt
Resolving rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)... 140.192.39.95
Connecting to rasinsrv07.cstcis.cti.depaul.edu (rasinsrv07.cstcis.cti.depaul.edu)|140.192.39.95|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6048428 (5.8M) [text/plain]
Saving to: 'customer.txt'

customer.txt                    100%[===========================================================================>]   5.77M  9.31MB/s    in 0.6s

2020-03-07 03:21:02 (9.31 MB/s) - 'customer.txt' saved [6048428/6048428]
```



```
  GNU nano 2.5.3                                    File: customer.txt

800000|Isabella|Cocklin|22 Essex Court, Mount Vernon, NY 10550
800001|Kathryn|Gillmore|535 Cambridge Drive, Lynn, MA 01902
800002|Aja|Pantano|520 Elmwood Avenue, Oshkosh, WI 54901
800003|Isabella|Mcdougle|654 Elmwood Avenue, Highland, IN 46322
800004|Corliss|Favor|960 Winding Way, Millington, TN 38053
800005|Jeanie|Karim|957 Liberty Street, Satellite Beach, FL 32937
800006|Renita|Batey|96 Buttonwood Drive, Port Huron, MI 48060
800007|Genaro|Bagwell|624 Franklin Avenue, Montclair, NJ 07042
```

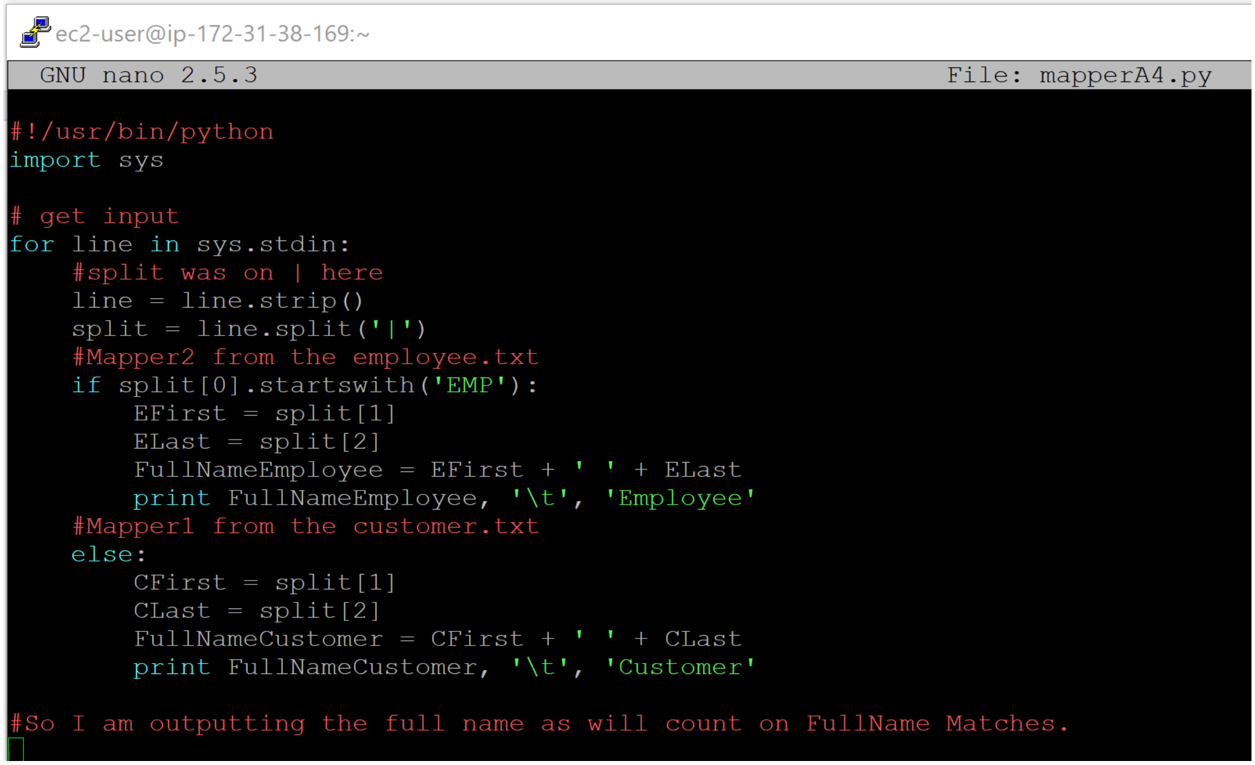


```
ec2-user@ip-172-31-38-169:~
  GNU nano 2.5.3                                        File: employee.txt

EMP0|Francoise|Maynor|72
EMP1|Isidro|Bosque|80
EMP2|Brendan|Platt|23
EMP3|Freeda|Lembke|64
EMP4|Marcella|Tyrrell|68
EMP5|Jordan|Mcdougle|78
EMP6|Sheldon|Read|78
```

Be sure to submit your python code, the command line and the screenshot of successful execution of your code.

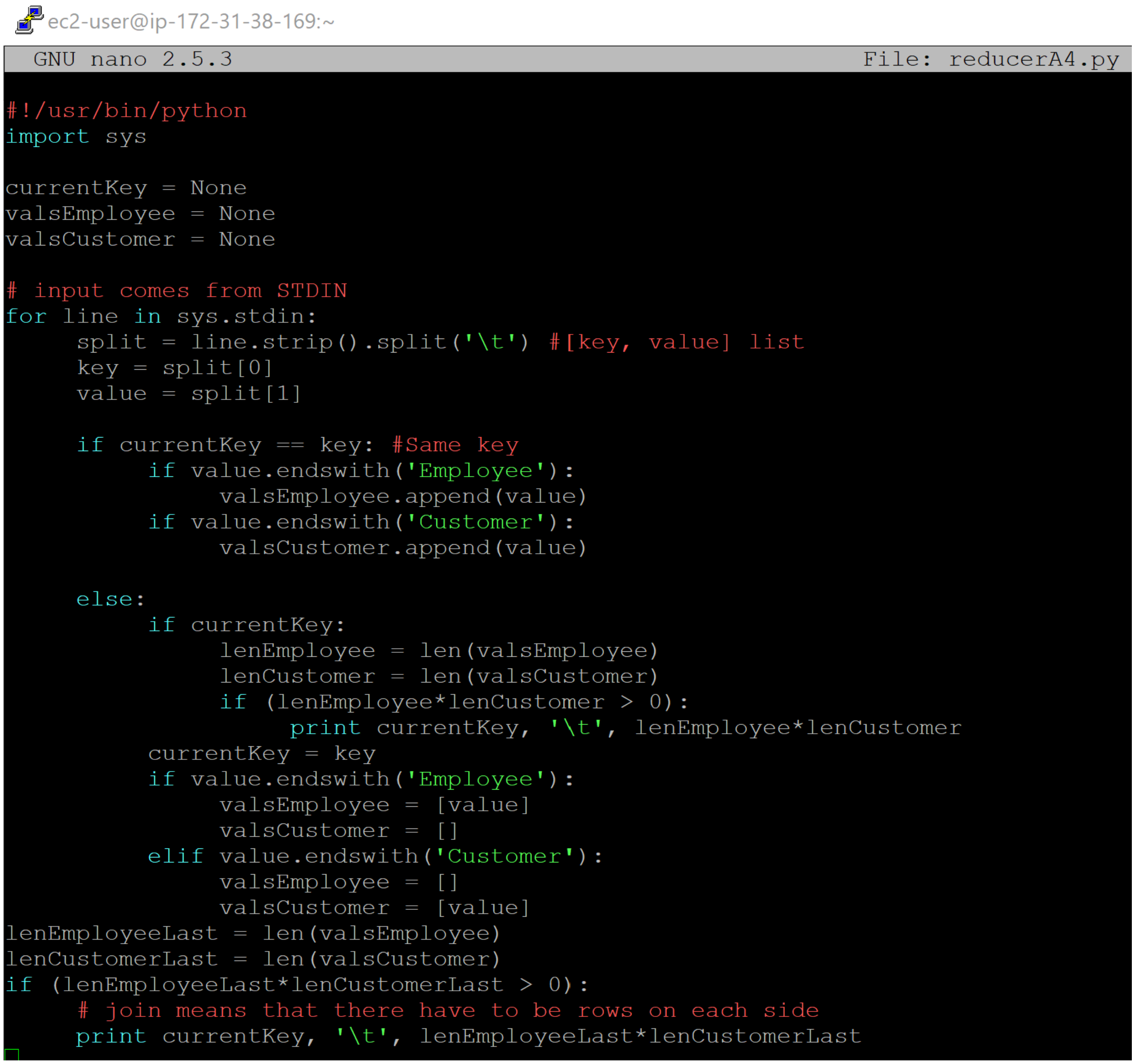
Python Code:

Mapper



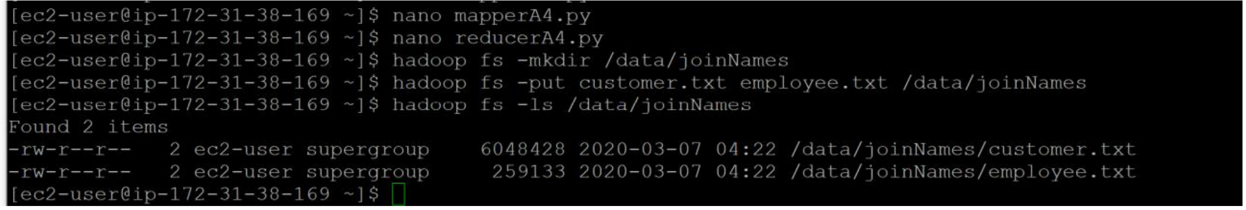
```
ec2-user@ip-172-31-38-169:~
  GNU nano 2.5.3                                       File: mapperA4.py

#!/usr/bin/python
import sys

# get input
for line in sys.stdin:
    #split was on | here
    line = line.strip()
    split = line.split('|')
    #Mapper2 from the employee.txt
    if split[0].startswith('EMP'):
        EFirst = split[1]
        ELast = split[2]
        FullNameEmployee = EFirst + ' ' + ELast
        print FullNameEmployee, '\t', 'Employee'
    #Mapper1 from the customer.txt
    else:
        CFirst = split[1]
        CLast = split[2]
        FullNameCustomer = CFirst + ' ' + CLast
        print FullNameCustomer, '\t', 'Customer'

#So I am outputting the full name as will count on FullName Matches.
```

Reducer



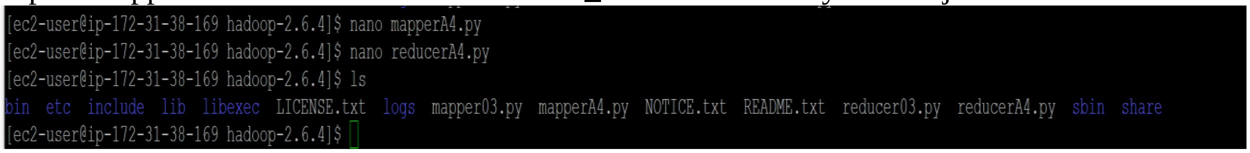
```
ec2-user@ip-172-31-38-169:~
  GNU nano 2.5.3                                          File: reducerA4.py

#!/usr/bin/python
import sys

currentKey = None
valsEmployee = None
valsCustomer = None

# input comes from STDIN
for line in sys.stdin:
     split = line.strip().split('\t') #[key, value] list
     key = split[0]
     value = split[1]

     if currentKey == key: #Same key
          if value.endswith('Employee'):
               valsEmployee.append(value)
          if value.endswith('Customer'):
               valsCustomer.append(value)

     else:
          if currentKey:
               lenEmployee = len(valsEmployee)
               lenCustomer = len(valsCustomer)
               if (lenEmployee*lenCustomer > 0):
                    print currentKey, '\t', lenEmployee*lenCustomer
          currentKey = key
          if value.endswith('Employee'):
               valsEmployee = [value]
               valsCustomer = []
          elif value.endswith('Customer'):
               valsEmployee = []
               valsCustomer = [value]
lenEmployeeLast = len(valsEmployee)
lenCustomerLast = len(valsCustomer)
if (lenEmployeeLast*lenCustomerLast > 0):
     # join means that there have to be rows on each side
     print currentKey, '\t', lenEmployeeLast*lenCustomerLast
```

making directories



```
[ec2-user@ip-172-31-38-169 ~]$ nano mapperA4.py
[ec2-user@ip-172-31-38-169 ~]$ nano reducerA4.py
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -mkdir /data/joinNames
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -put customer.txt employee.txt /data/joinNames
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -ls /data/joinNames
Found 2 items
-rw-r--r--   2 ec2-user supergroup    6048428 2020-03-07 04:22 /data/joinNames/customer.txt
-rw-r--r--   2 ec2-user supergroup     259133 2020-03-07 04:22 /data/joinNames/employee.txt
[ec2-user@ip-172-31-38-169 ~]$
```

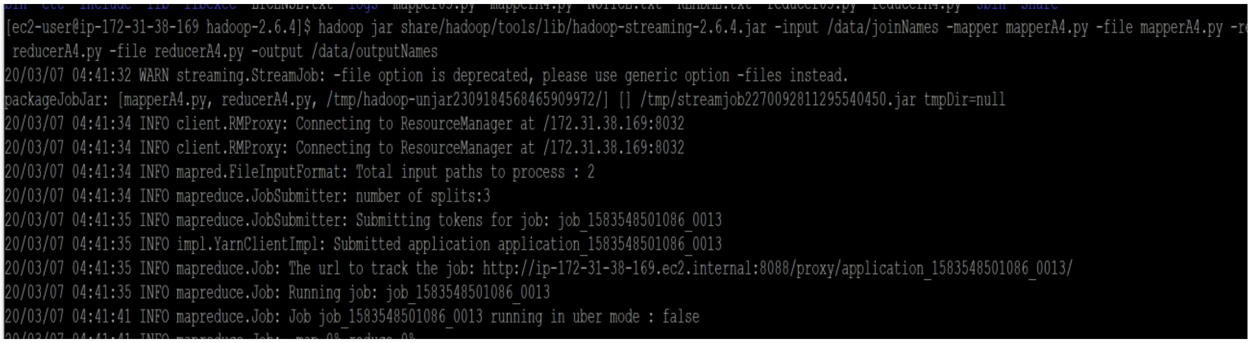
copied mapper and reducer over to HADOOP_HOME to actually run the job.



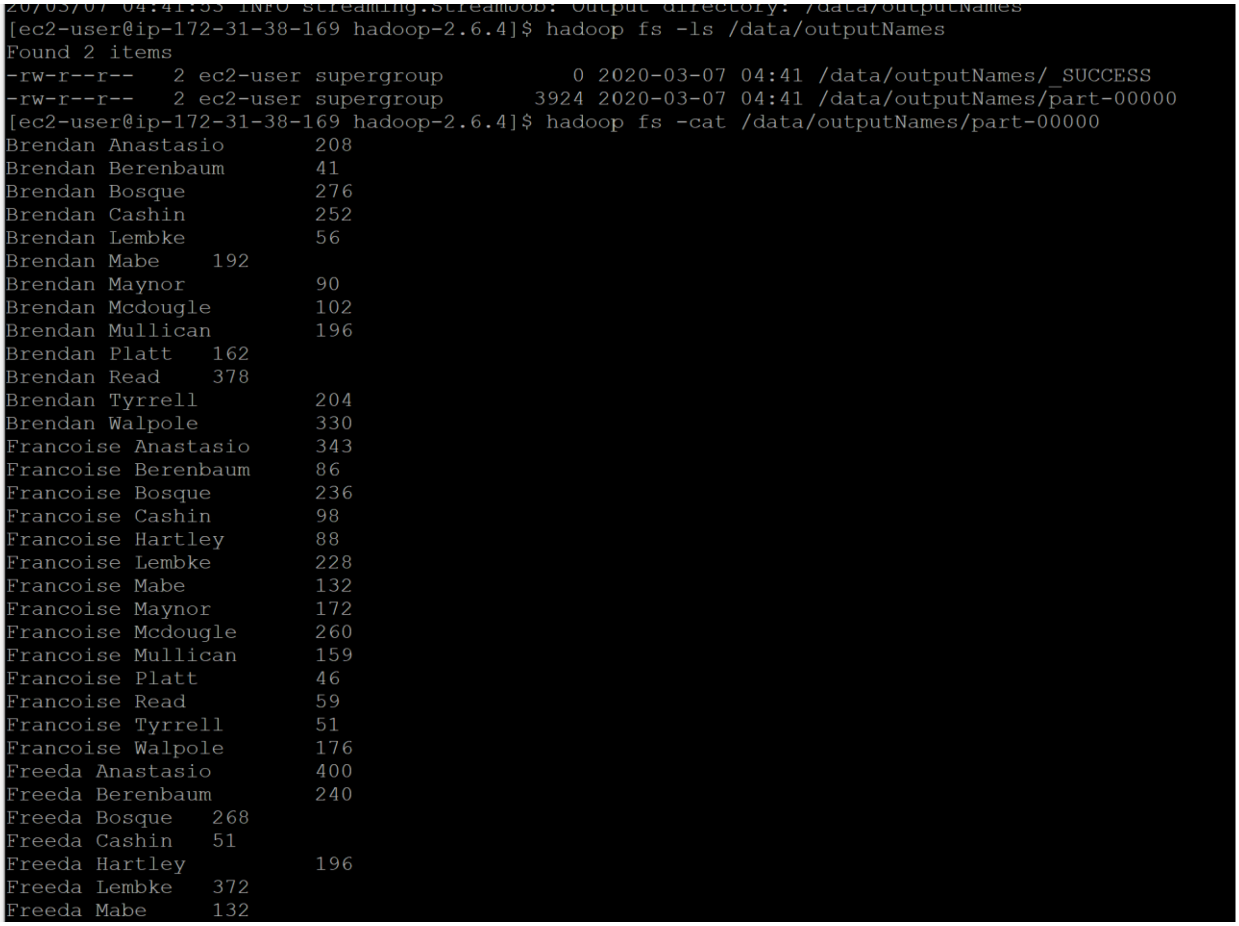
```
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ nano mapperA4.py
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ nano reducerA4.py
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ ls
bin  etc  include  lib  libexec  LICENSE.txt  logs  mapper03.py  mapperA4.py  NOTICE.txt  README.txt  reducer03.py  reducerA4.py  sbin  share
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$
```

<u>RUN:</u>

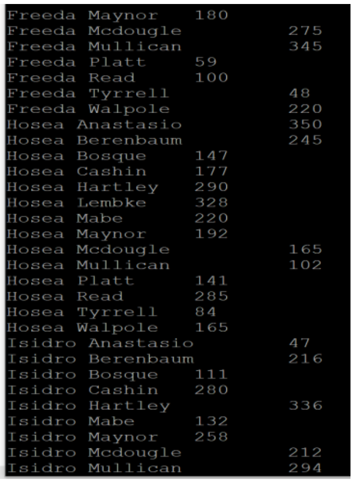
hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input /data/joinNames -mapper mapperA4.py -file mapperA4.py -reducer reducerA4.py -file reducerA4.py -output /data/outputNames



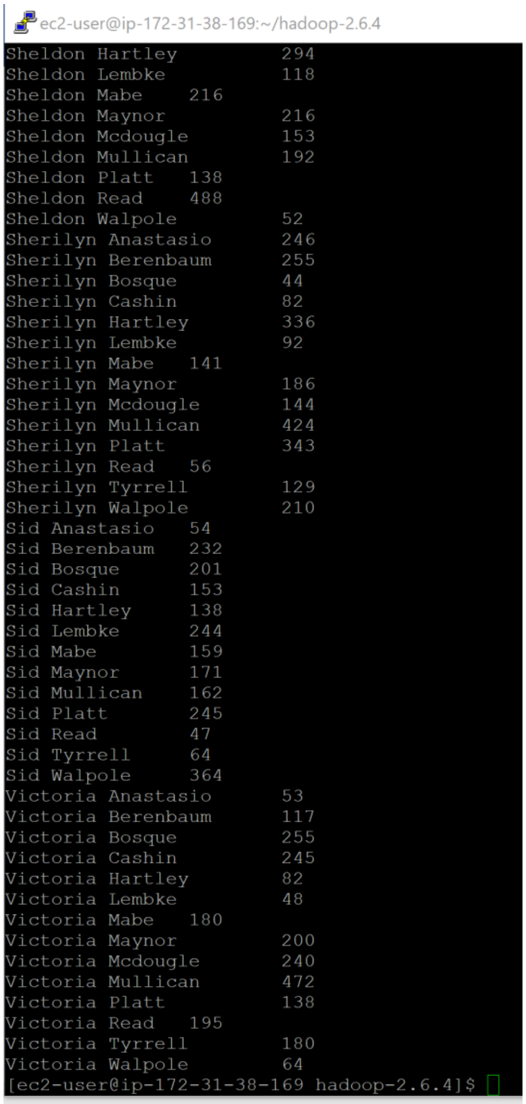
```
bin  etc  include  lib  libexec  LICENSE.txt  logs  mapperA3.py  mapperA4.py  NOTICE.txt  README.txt  reducerA3.py  reducerA4.py  sbin  share
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input /data/joinNames -mapper mapperA4.py -file mapperA4.py -re
 reducerA4.py -file reducerA4.py -output /data/outputNames
20/03/07 04:41:32 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapperA4.py, reducerA4.py, /tmp/hadoop-unjar2309184568465909972/] [] /tmp/streamjob2270092811295540450.jar tmpDir=null
20/03/07 04:41:34 INFO client.RMProxy: Connecting to ResourceManager at /172.31.38.169:8032
20/03/07 04:41:34 INFO client.RMProxy: Connecting to ResourceManager at /172.31.38.169:8032
20/03/07 04:41:34 INFO mapred.FileInputFormat: Total input paths to process : 2
20/03/07 04:41:34 INFO mapreduce.JobSubmitter: number of splits:3
20/03/07 04:41:35 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1583548501086_0013
20/03/07 04:41:35 INFO impl.YarnClientImpl: Submitted application application_1583548501086_0013
20/03/07 04:41:35 INFO mapreduce.Job: The url to track the job: http://ip-172-31-38-169.ec2.internal:8088/proxy/application_1583548501086_0013/
20/03/07 04:41:35 INFO mapreduce.Job: Running job: job_1583548501086_0013
20/03/07 04:41:41 INFO mapreduce.Job: Job job_1583548501086_0013 running in uber mode : false
20/03/07 04:41:41 INFO mapreduce.Job:  map 0% reduce 0%
```



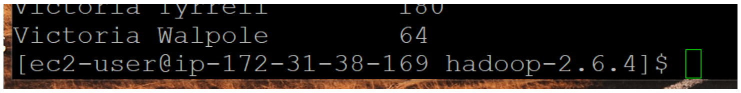
```
20/03/07 04:41:53 INFO streaming.StreamJob: Output directory: /data/outputNames
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ hadoop fs -ls /data/outputNames
Found 2 items
-rw-r--r--   2 ec2-user supergroup          0 2020-03-07 04:41 /data/outputNames/_SUCCESS
-rw-r--r--   2 ec2-user supergroup       3924 2020-03-07 04:41 /data/outputNames/part-00000
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ hadoop fs -cat /data/outputNames/part-00000
Brendan Anastasio       208
Brendan Berenbaum       41
Brendan Bosque          276
Brendan Cashin          252
Brendan Lembke          56
Brendan Mabe    192
Brendan Maynor          90
Brendan Mcdougle        102
Brendan Mullican        196
Brendan Platt   162
Brendan Read    378
Brendan Tyrrell         204
Brendan Walpole         330
Francoise Anastasio     343
Francoise Berenbaum     86
Francoise Bosque        236
Francoise Cashin        98
Francoise Hartley       88
Francoise Lembke        228
Francoise Mabe          132
Francoise Maynor        172
Francoise Mcdougle      260
Francoise Mullican      159
Francoise Platt         46
Francoise Read          59
Francoise Tyrrell       51
Francoise Walpole       176
Freeda Anastasio        400
Freeda Berenbaum        240
Freeda Bosque   268
Freeda Cashin   51
Freeda Hartley          196
Freeda Lembke   372
Freeda Mabe     132
```



```
Freeda Maynor   180
Freeda Mcdougle         275
Freeda Mullican         345
Freeda Platt    59
Freeda Read     100
Freeda Tyrrell          48
Freeda Walpole          220
Hosea Anastasio         350
Hosea Berenbaum         245
Hosea Bosque    147
Hosea Cashin    177
Hosea Hartley   290
Hosea Lembke    328
Hosea Mabe      220
Hosea Maynor    192
Hosea Mcdougle          165
Hosea Mullican          102
Hosea Platt     141
Hosea Read      285
Hosea Tyrrell   84
Hosea Walpole   165
Isidro Anastasio        47
Isidro Berenbaum        216
Isidro Bosque   111
Isidro Cashin   280
Isidro Hartley          336
Isidro Mabe     132
Isidro Maynor   258
Isidro Mcdougle         212
Isidro Mullican         294
```

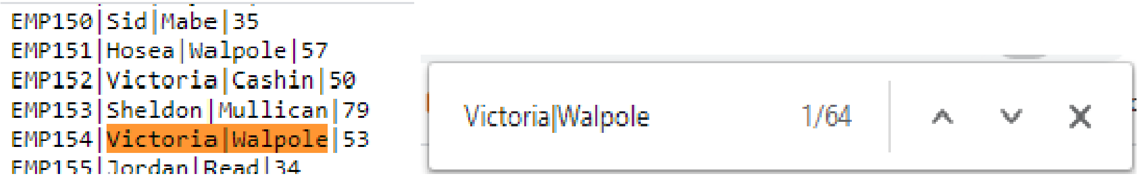


```
Sheldon Hartley         294
Sheldon Lembke          118
Sheldon Mabe    216
Sheldon Maynor          216
Sheldon Mcdougle        153
Sheldon Mullican        192
Sheldon Platt   138
Sheldon Read    488
Sheldon Walpole         52
Sherilyn Anastasio      246
Sherilyn Berenbaum      255
Sherilyn Bosque         44
Sherilyn Cashin         82
Sherilyn Hartley        336
Sherilyn Lembke         92
Sherilyn Mabe   141
Sherilyn Maynor         186
Sherilyn Mcdougle       144
Sherilyn Mullican       424
Sherilyn Platt          343
Sherilyn Read   56
Sherilyn Tyrrell        129
Sherilyn Walpole        210
Sid Anastasio   54
Sid Berenbaum   232
Sid Bosque      201
Sid Cashin      153
Sid Hartley     138
Sid Lembke      244
Sid Mabe        159
Sid Maynor      171
Sid Mullican    162
Sid Platt       245
Sid Read        47
Sid Tyrrell     64
Sid Walpole     364
Victoria Anastasio      53
Victoria Berenbaum      117
Victoria Bosque         255
Victoria Cashin         245
Victoria Hartley        82
Victoria Lembke         48
Victoria Mabe   180
Victoria Maynor         200
Victoria Mcdougle       240
Victoria Mullican       472
Victoria Platt          138
Victoria Read   195
Victoria Tyrrell        180
Victoria Walpole        64
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$
```



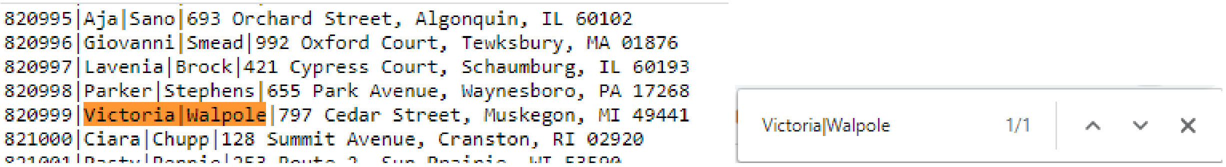
```
Victoria Tyrrell         180
Victoria Walpole         64
[ec2-user@ip-172-31-38-169 hadoop-2.6.4]$ 
```

Sanity Check:
Victoria Walpole = 64 ?



```
EMP150|Sid|Mabe|35
EMP151|Hosea|Walpole|57
EMP152|Victoria|Cashin|50
EMP153|Sheldon|Mullican|79
EMP154|Victoria|Walpole|53
EMP155|Jordan|Read|34
```

^64 instances in the Employee.txt file from http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/employee.txt



```
820995|Aja|Sano|693 Orchard Street, Algonquin, IL 60102
820996|Giovanni|Smead|992 Oxford Court, Tewksbury, MA 01876
820997|Lavenia|Brock|421 Cypress Court, Schaumburg, IL 60193
820998|Parker|Stephens|655 Park Avenue, Waynesboro, PA 17268
820999|Victoria|Walpole|797 Cedar Street, Muskegon, MI 49441
821000|Ciara|Chupp|128 Summit Avenue, Cranston, RI 02920
821001|Rusty|Bonnie|253 Route 2, Sun Prairie, WI 53590
```

^1 instance in the Customer.txt from http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/customer.txt

➔ I conclude that the MapReduce Job ran effectively given that it worked for Victoria Walpole. Those names were correctly counted. I ran this on the 3-node cluster from Phase1 of the project.

5) In this section you will practice using HBase and setup Mahout and run the curve-clustering example.

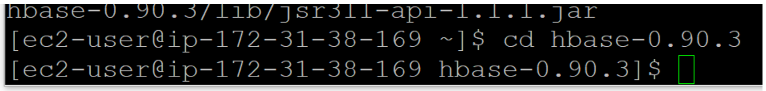
   a) Note that HBase runs on top of HDFS, bypassing MapReduce (so only NameNode and DataNode need to be running). You can use your 4-node cluster or the 1-node cluster to run HBase, but be sure to specify which one you used.

<u>I used the 3-node setup</u>

```
cd
```

(Download HBase)

```
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/hbase-0.90.3.tar.gz
gunzip hbase-0.90.3.tar.gz
tar xvf hbase-0.90.3.tar

cd hbase-0.90.3
```



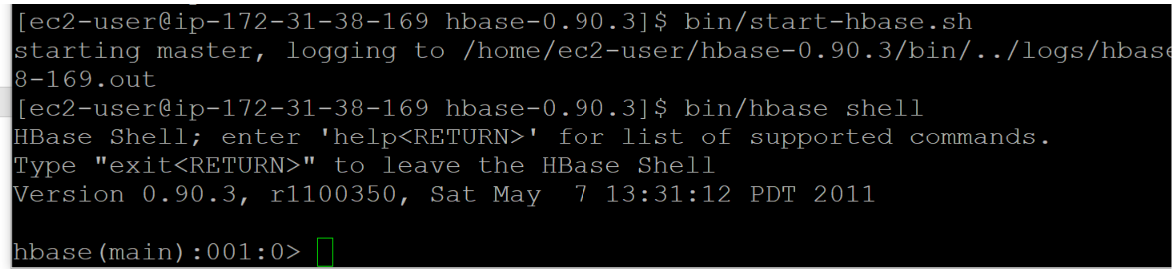
```
hbase-0.90.3/lib/jsr311-api-1.1.1.jar
[ec2-user@ip-172-31-38-169 ~]$ cd hbase-0.90.3
[ec2-user@ip-172-31-38-169 hbase-0.90.3]$ 
```

(Start HBase service, there is a corresponding stop service and this assumes Hadoop home is set)

```
bin/start-hbase.sh
```

(Open the HBase shell – at this point jps should show HMaster)

bin/hbase shell

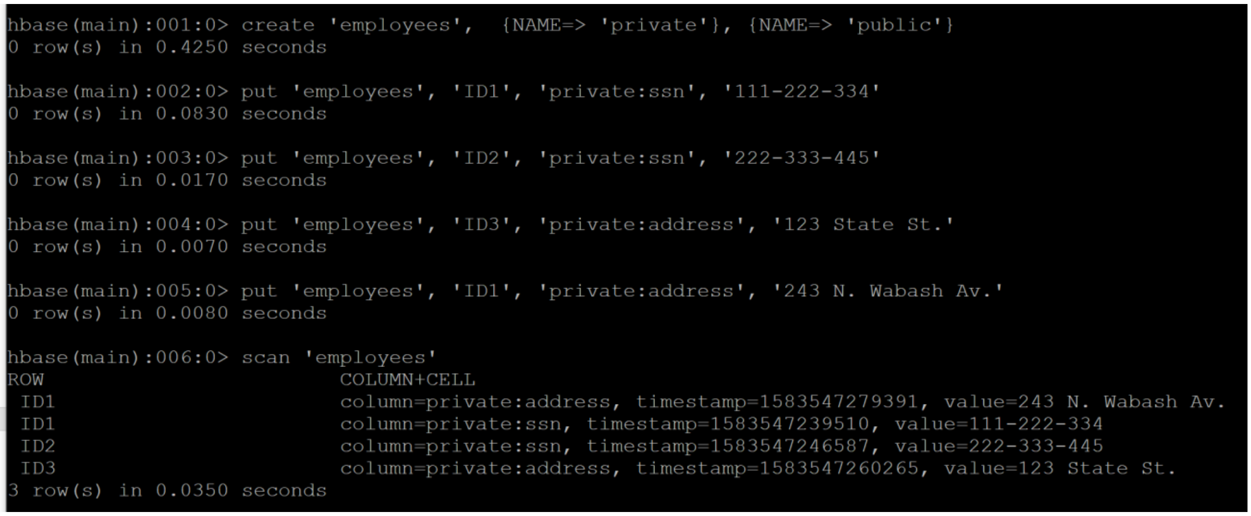


```
[ec2-user@ip-172-31-38-169 hbase-0.90.3]$ bin/start-hbase.sh
starting master, logging to /home/ec2-user/hbase-0.90.3/bin/../logs/hbase
8-169.out
[ec2-user@ip-172-31-38-169 hbase-0.90.3]$ bin/hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.90.3, r1100350, Sat May  7 13:31:12 PDT 2011

hbase(main):001:0>
```

(Create an employee table and two column families – private and public. Please watch the quotes, if ' turns into ‘, the commands will not work)

create 'employees', {NAME=> 'private'}, {NAME=> 'public'}
put 'employees', 'ID1', 'private:ssn', '111-222-334'
put 'employees', 'ID2', 'private:ssn', '222-333-445'
put 'employees', 'ID3', 'private:address', '123 State St.'
put 'employees', 'ID1', 'private:address', '243 N. Wabash Av.'

scan 'employees'



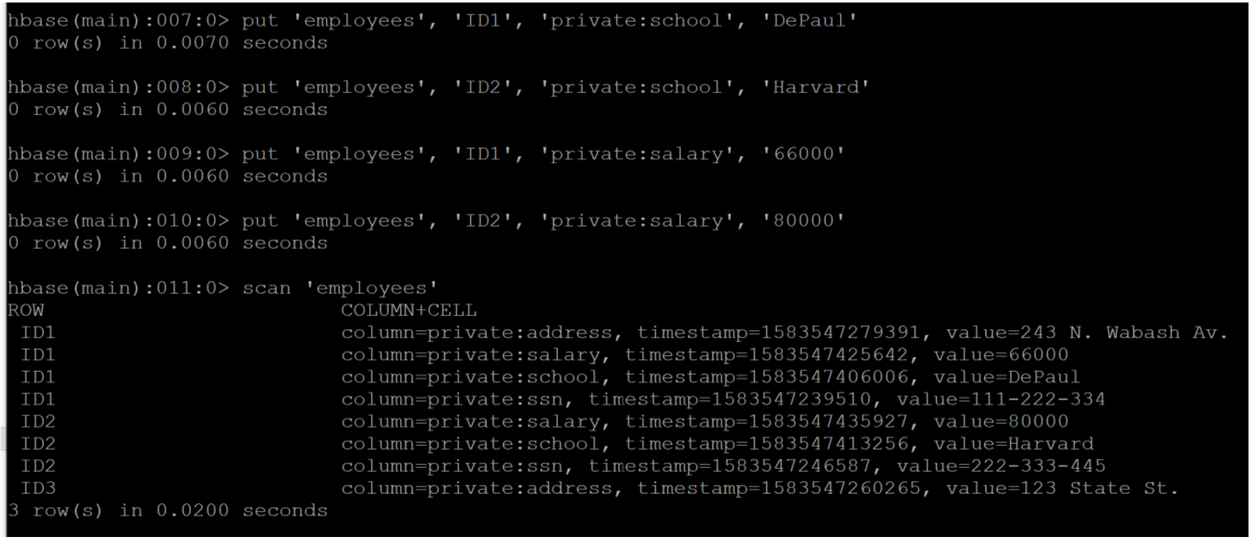
```
hbase(main):001:0> create 'employees',  {NAME=> 'private'}, {NAME=> 'public'}
0 row(s) in 0.4250 seconds

hbase(main):002:0> put 'employees', 'ID1', 'private:ssn', '111-222-334'
0 row(s) in 0.0830 seconds

hbase(main):003:0> put 'employees', 'ID2', 'private:ssn', '222-333-445'
0 row(s) in 0.0170 seconds

hbase(main):004:0> put 'employees', 'ID3', 'private:address', '123 State St.'
0 row(s) in 0.0070 seconds

hbase(main):005:0> put 'employees', 'ID1', 'private:address', '243 N. Wabash Av.'
0 row(s) in 0.0080 seconds

hbase(main):006:0> scan 'employees'
ROW                          COLUMN+CELL
 ID1                         column=private:address, timestamp=1583547279391, value=243 N. Wabash Av.
 ID1                         column=private:ssn, timestamp=1583547239510, value=111-222-334
 ID2                         column=private:ssn, timestamp=1583547246587, value=222-333-445
 ID3                         column=private:address, timestamp=1583547260265, value=123 State St.
3 row(s) in 0.0350 seconds
```

Now that we have filled in a couple of values, add 2 new columns to the private family, 1 new column to the public family and create a brand new family with at least 3 columns. For each of these you should introduce at least 2 values -- so a total of (2+1+3) * 2 = 12 values inserted.

➔ <u>add 2 new columns to the private family</u>

put 'employees', 'ID1', 'private:school', 'DePaul'
put 'employees', 'ID2', 'private:school', 'Harvard'

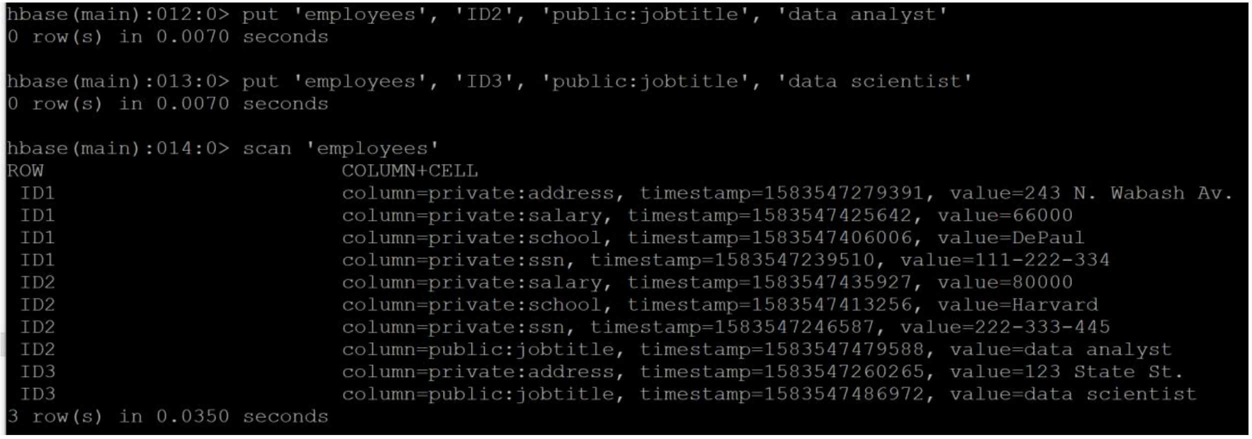
put 'employees', 'ID1', 'private:salary', '66000'
put 'employees', 'ID2', 'private:salary', '80000'



```
hbase(main):007:0> put 'employees', 'ID1', 'private:school', 'DePaul'
0 row(s) in 0.0070 seconds

hbase(main):008:0> put 'employees', 'ID2', 'private:school', 'Harvard'
0 row(s) in 0.0060 seconds

hbase(main):009:0> put 'employees', 'ID1', 'private:salary', '66000'
0 row(s) in 0.0060 seconds

hbase(main):010:0> put 'employees', 'ID2', 'private:salary', '80000'
0 row(s) in 0.0060 seconds

hbase(main):011:0> scan 'employees'
ROW                           COLUMN+CELL
 ID1                          column=private:address, timestamp=1583547279391, value=243 N. Wabash Av.
 ID1                          column=private:salary, timestamp=1583547425642, value=66000
 ID1                          column=private:school, timestamp=1583547406006, value=DePaul
 ID1                          column=private:ssn, timestamp=1583547239510, value=111-222-334
 ID2                          column=private:salary, timestamp=1583547435927, value=80000
 ID2                          column=private:school, timestamp=1583547413256, value=Harvard
 ID2                          column=private:ssn, timestamp=1583547246587, value=222-333-445
 ID3                          column=private:address, timestamp=1583547260265, value=123 State St.
3 row(s) in 0.0200 seconds
```

➔ 1 new column to the public family

put 'employees', 'ID2', 'public:jobtitle', 'data analyst'
put 'employees', 'ID3', 'public:jobtitle', 'data scientist'



```
hbase(main):012:0> put 'employees', 'ID2', 'public:jobtitle', 'data analyst'
0 row(s) in 0.0070 seconds

hbase(main):013:0> put 'employees', 'ID3', 'public:jobtitle', 'data scientist'
0 row(s) in 0.0070 seconds

hbase(main):014:0> scan 'employees'
ROW                           COLUMN+CELL
 ID1                          column=private:address, timestamp=1583547279391, value=243 N. Wabash Av.
 ID1                          column=private:salary, timestamp=1583547425642, value=66000
 ID1                          column=private:school, timestamp=1583547406006, value=DePaul
 ID1                          column=private:ssn, timestamp=1583547239510, value=111-222-334
 ID2                          column=private:salary, timestamp=1583547435927, value=80000
 ID2                          column=private:school, timestamp=1583547413256, value=Harvard
 ID2                          column=private:ssn, timestamp=1583547246587, value=222-333-445
 ID2                          column=public:jobtitle, timestamp=1583547479588, value=data analyst
 ID3                          column=private:address, timestamp=1583547260265, value=123 State St.
 ID3                          column=public:jobtitle, timestamp=1583547486972, value=data scientist
3 row(s) in 0.0350 seconds
```

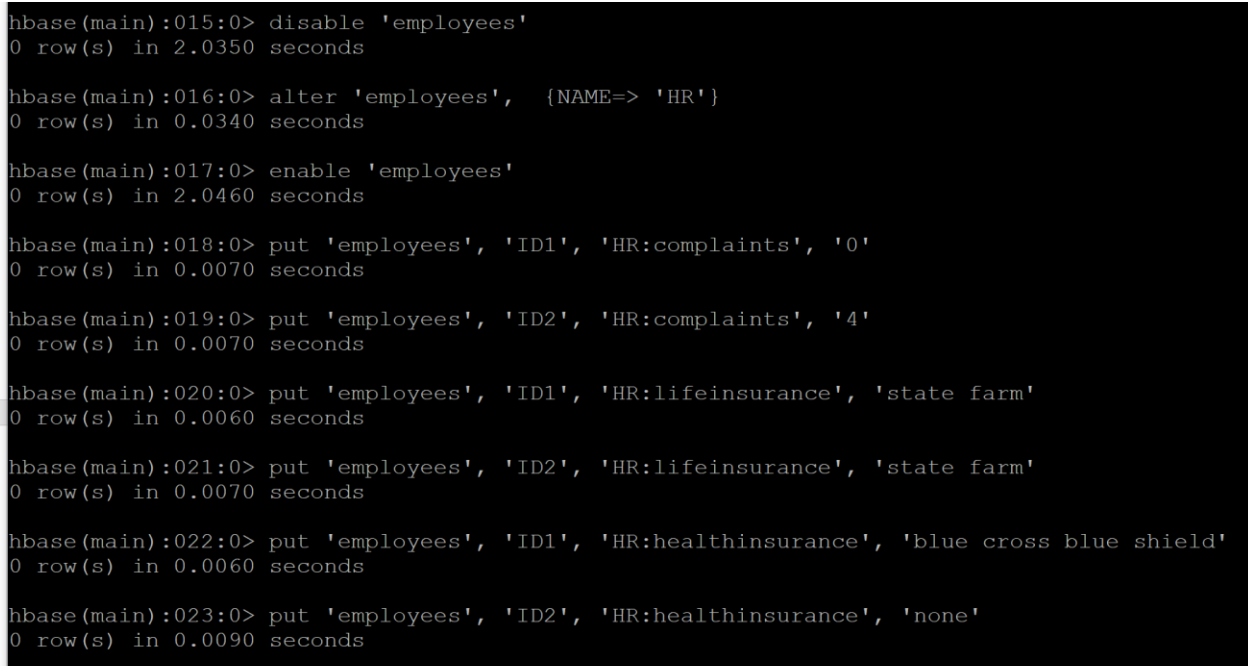
➔ create a brand new family with at least 3 columns

disable 'employees'
alter 'employees', {NAME=> 'HR'}
enable 'employees'

put 'employees', 'ID1', 'HR:complaints', '0'
put 'employees', 'ID2', 'HR:complaints', '4'

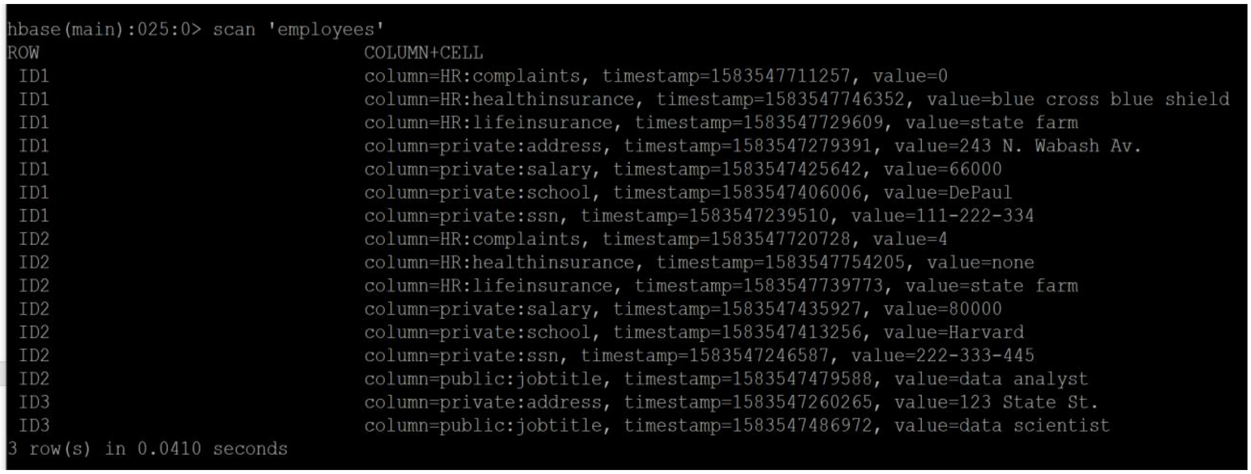
put 'employees', 'ID1', 'HR:lifeinsurance', 'state farm'
put 'employees', 'ID2', 'HR:lifeinsurance', 'state farm'

put 'employees', 'ID1', 'HR:healthinsurance', 'blue cross blue shield'
put 'employees', 'ID2', 'HR:healthinsurance', 'none'



```
hbase(main):015:0> disable 'employees'
0 row(s) in 2.0350 seconds

hbase(main):016:0> alter 'employees',  {NAME=> 'HR'}
0 row(s) in 0.0340 seconds

hbase(main):017:0> enable 'employees'
0 row(s) in 2.0460 seconds

hbase(main):018:0> put 'employees', 'ID1', 'HR:complaints', '0'
0 row(s) in 0.0070 seconds

hbase(main):019:0> put 'employees', 'ID2', 'HR:complaints', '4'
0 row(s) in 0.0070 seconds

hbase(main):020:0> put 'employees', 'ID1', 'HR:lifeinsurance', 'state farm'
0 row(s) in 0.0060 seconds

hbase(main):021:0> put 'employees', 'ID2', 'HR:lifeinsurance', 'state farm'
0 row(s) in 0.0070 seconds

hbase(main):022:0> put 'employees', 'ID1', 'HR:healthinsurance', 'blue cross blue shield'
0 row(s) in 0.0060 seconds

hbase(main):023:0> put 'employees', 'ID2', 'HR:healthinsurance', 'none'
0 row(s) in 0.0090 seconds
```

Verify that the table has been filled in properly with scan command and submit a screenshot.



```
hbase(main):025:0> scan 'employees'
ROW                               COLUMN+CELL
 ID1                              column=HR:complaints, timestamp=1583547711257, value=0
 ID1                              column=HR:healthinsurance, timestamp=1583547746352, value=blue cross blue shield
 ID1                              column=HR:lifeinsurance, timestamp=1583547729609, value=state farm
 ID1                              column=private:address, timestamp=1583547279391, value=243 N. Wabash Av.
 ID1                              column=private:salary, timestamp=1583547425642, value=66000
 ID1                              column=private:school, timestamp=1583547406006, value=DePaul
 ID1                              column=private:ssn, timestamp=1583547239510, value=111-222-334
 ID2                              column=HR:complaints, timestamp=1583547720728, value=4
 ID2                              column=HR:healthinsurance, timestamp=1583547754205, value=none
 ID2                              column=HR:lifeinsurance, timestamp=1583547739773, value=state farm
 ID2                              column=private:salary, timestamp=1583547435927, value=80000
 ID2                              column=private:school, timestamp=1583547413256, value=Harvard
 ID2                              column=private:ssn, timestamp=1583547246587, value=222-333-445
 ID2                              column=public:jobtitle, timestamp=1583547479588, value=data analyst
 ID3                              column=private:address, timestamp=1583547260265, value=123 State St.
 ID3                              column=public:jobtitle, timestamp=1583547486972, value=data scientist
3 row(s) in 0.0410 seconds
```

➔ worked

b) Download and setup Mahout:

cd
(download mahout zip package)
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-mahout-distribution-0.11.2.zip
(Unzip the file)

unzip apache-mahout-distribution-0.11.2.zip

set the environment variables (as always, you can put these commands in ~/.bashrc to automatically set these variables every time you open a new connection, source ~/.bashrc to refresh)

export MAHOUT_HOME=/home/ec2-user/apache-mahout-distribution-0.11.2

export PATH=/home/ec2-user/apache-mahout-distribution-0.11.2/bin:$PATH

be absolutely sure you set Hadoop home variable (if you haven't):

Download and prepare synthetic data – it represents a list of 2D curves, represented as a 50-point vector.

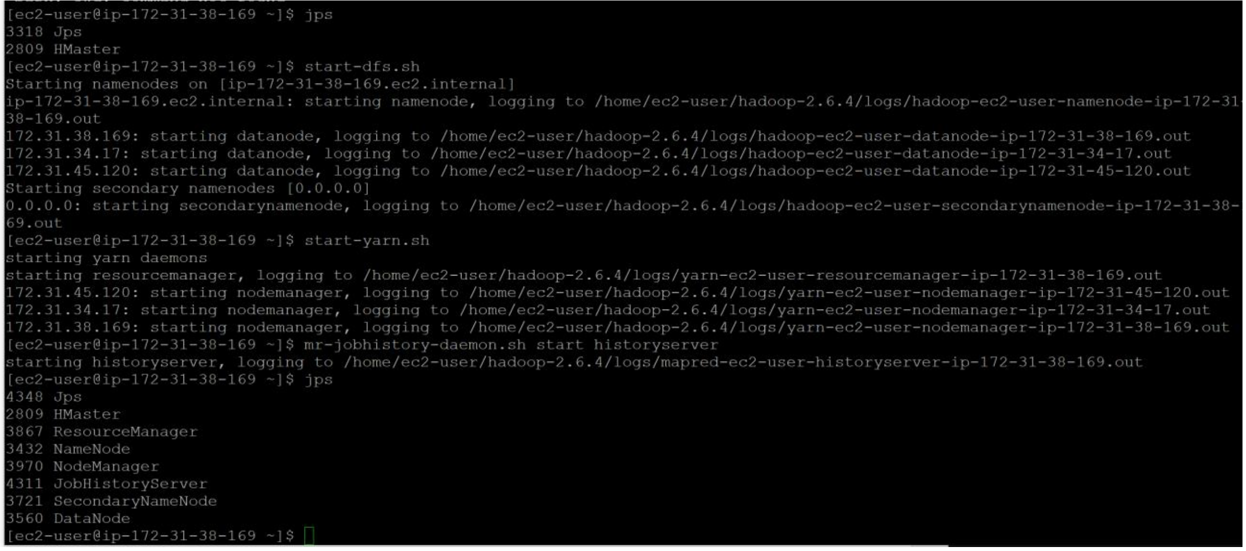
Download the synthetic data example:

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/synthetic_control.data

(make a testdata directory in HDFS, the example KMeans algorithm assumes the data lives there by default.

*

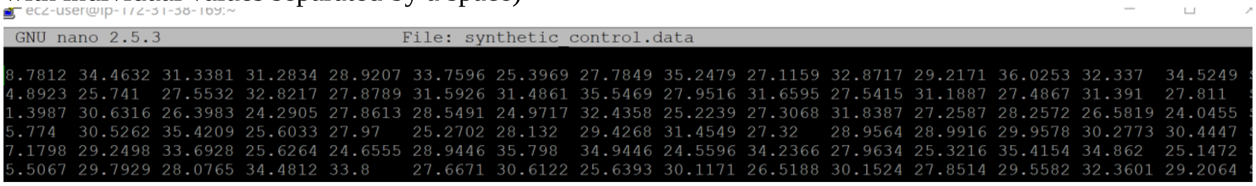
start-dfs.sh
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver
jps



```
[ec2-user@ip-172-31-38-169 ~]$ jps
3318 Jps
2809 HMaster
[ec2-user@ip-172-31-38-169 ~]$ start-dfs.sh
Starting namenodes on [ip-172-31-38-169.ec2.internal]
ip-172-31-38-169.ec2.internal: starting namenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-ec2-user-namenode-ip-172-31-
38-169.out
172.31.38.169: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-ec2-user-datanode-ip-172-31-38-169.out
172.31.34.17: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-ec2-user-datanode-ip-172-31-34-17.out
172.31.45.120: starting datanode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-ec2-user-datanode-ip-172-31-45-120.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ec2-user/hadoop-2.6.4/logs/hadoop-ec2-user-secondarynamenode-ip-172-31-38-
69.out
[ec2-user@ip-172-31-38-169 ~]$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/ec2-user/hadoop-2.6.4/logs/yarn-ec2-user-resourcemanager-ip-172-31-38-169.out
172.31.45.120: starting nodemanager, logging to /home/ec2-user/hadoop-2.6.4/logs/yarn-ec2-user-nodemanager-ip-172-31-45-120.out
172.31.34.17: starting nodemanager, logging to /home/ec2-user/hadoop-2.6.4/logs/yarn-ec2-user-nodemanager-ip-172-31-34-17.out
172.31.38.169: starting nodemanager, logging to /home/ec2-user/hadoop-2.6.4/logs/yarn-ec2-user-nodemanager-ip-172-31-38-169.out
[ec2-user@ip-172-31-38-169 ~]$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/ec2-user/hadoop-2.6.4/logs/mapred-ec2-user-historyserver-ip-172-31-38-169.out
[ec2-user@ip-172-31-38-169 ~]$ jps
4348 Jps
2809 HMaster
3867 ResourceManager
3432 NameNode
3970 NodeManager
4311 JobHistoryServer
3721 SecondaryNameNode
3560 DataNode
[ec2-user@ip-172-31-38-169 ~]$
```

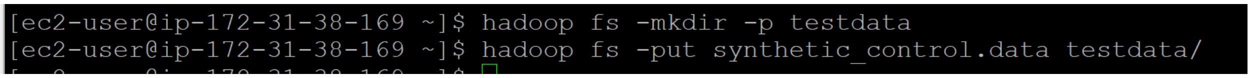
hadoop fs -mkdir -p testdata

(copy the synthetic data over to the testdata directory on HDFS side. You can inspect the contents of the file by running **nano synthetic_control.data** – as you can see this is a list of 600 vectors, with individual values separated by a space)



```
 GNU nano 2.5.3                          File: synthetic_control.data

8.7812 34.4632 31.3381 31.2834 28.9207 33.7596 25.3969 27.7849 35.2479 27.1159 32.8717 29.2171 36.0253 32.337  34.5249
4.8923 25.741  27.5532 32.8217 27.8789 31.5926 31.4861 35.5469 27.9516 31.6595 27.5415 31.1887 27.4867 31.391  27.811
1.3987 30.6316 26.3983 24.2905 27.8613 28.5491 24.9717 32.4358 25.2239 27.3068 31.8387 27.2587 28.2572 26.5819 24.0455
5.774  30.5262 35.4209 25.6033 27.97   25.2702 28.132  29.4268 31.4549 27.32   28.9564 28.9916 29.9578 30.2773 30.4447
7.1798 29.2498 33.6928 25.6264 24.6555 28.9446 35.798  34.9446 24.5596 34.2366 27.9634 25.3216 35.4154 34.862  25.1472
5.5067 29.7929 28.0765 34.4812 33.8    27.6671 30.6122 25.6393 30.1171 26.5188 30.1524 27.8514 29.5582 32.3601 29.2064
```

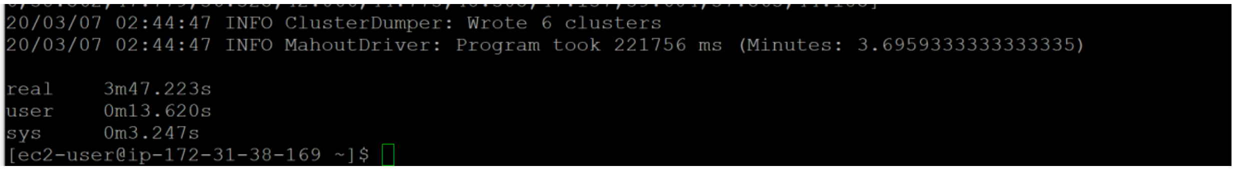
hadoop fs -put synthetic_control.data testdata/



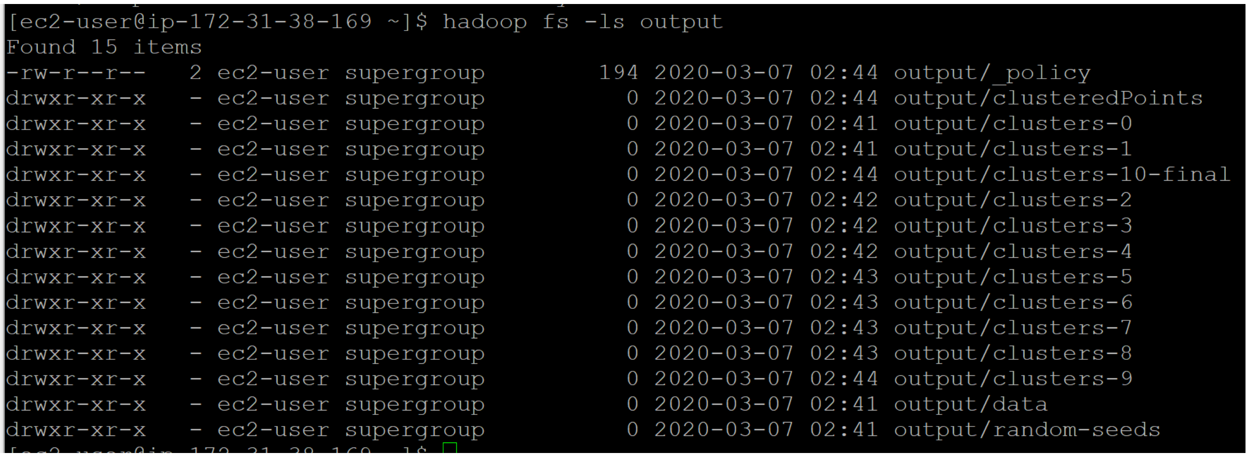
```
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -mkdir -p testdata
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -put synthetic_control.data testdata/
```

Please be sure to report the runtime of any command that includes "time"

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



```
20/03/07 02:44:47 INFO ClusterDumper: Wrote 6 clusters
20/03/07 02:44:47 INFO MahoutDriver: Program took 221756 ms (Minutes: 3.6959333333333335)

real    3m47.223s
user    0m13.620s
sys     0m3.247s
[ec2-user@ip-172-31-38-169 ~]$
```

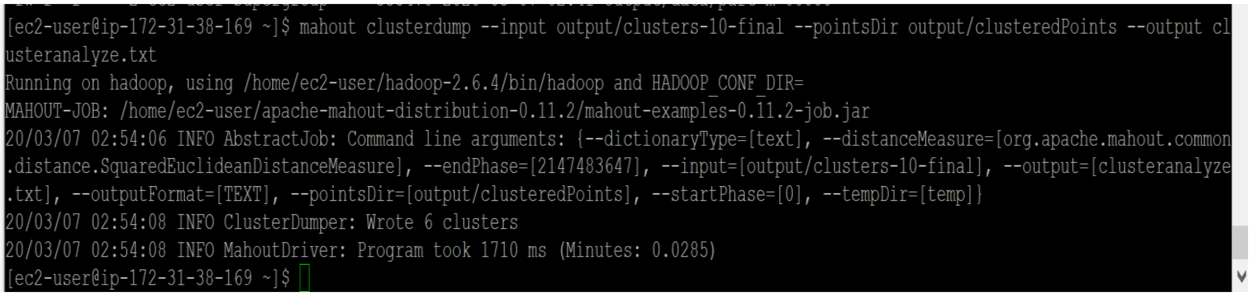


```
[ec2-user@ip-172-31-38-169 ~]$ hadoop fs -ls output
Found 15 items
-rw-r--r--   2 ec2-user supergroup        194 2020-03-07 02:44 output/_policy
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:44 output/clusteredPoints
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:41 output/clusters-0
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:41 output/clusters-1
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:44 output/clusters-10-final
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:42 output/clusters-2
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:42 output/clusters-3
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:42 output/clusters-4
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:43 output/clusters-5
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:43 output/clusters-6
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:43 output/clusters-7
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:43 output/clusters-8
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:44 output/clusters-9
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:41 output/data
drwxr-xr-x   - ec2-user supergroup          0 2020-03-07 02:41 output/random-seeds
```

^it worked

(clusterdump is a built-in Mahout command that will produce the result of KMeans. Output file is written to clusters-10-final because that is where the output is written after 10 iterations. The center points are placed in a separate file, called clusteredPoints)

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

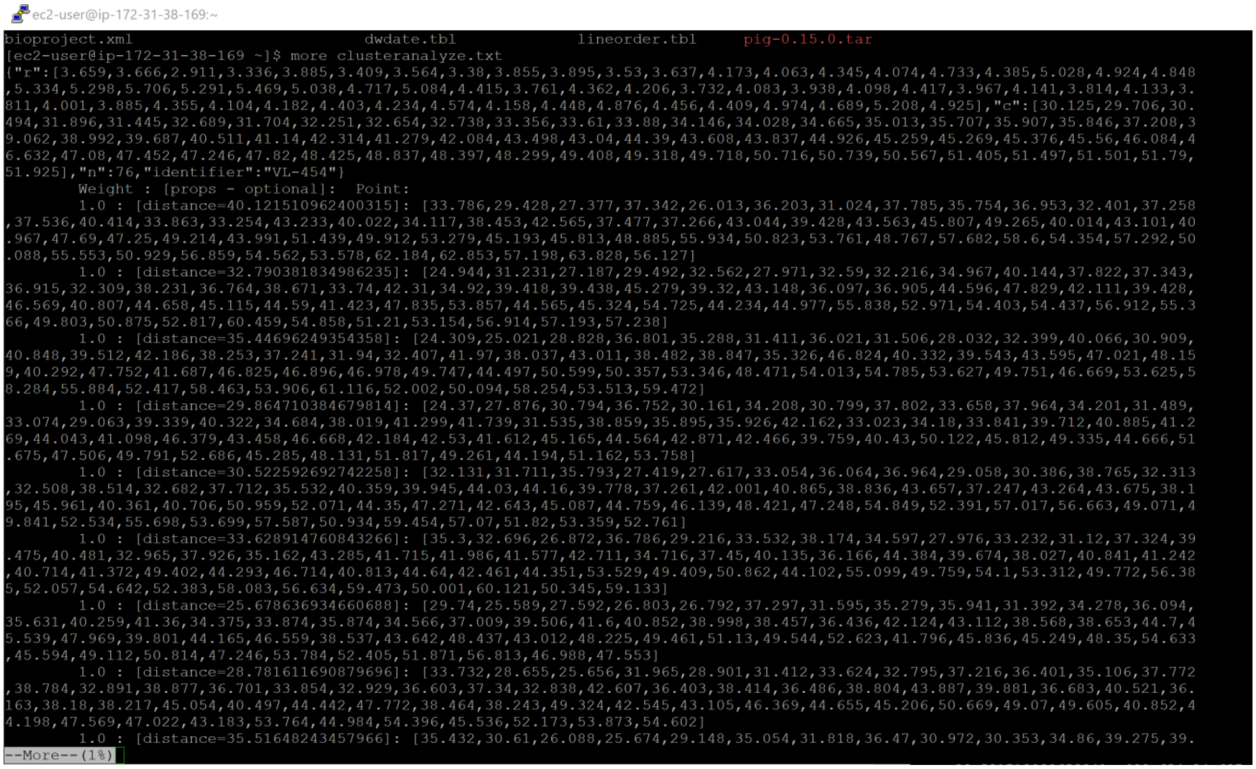


```
[ec2-user@ip-172-31-38-169 ~]$ mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints --output cl
usteranalyze.txt
Running on hadoop, using /home/ec2-user/hadoop-2.6.4/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/ec2-user/apache-mahout-distribution-0.11.2/mahout-examples-0.11.2-job.jar
20/03/07 02:54:06 INFO AbstractJob: Command line arguments: {--dictionaryType=[text], --distanceMeasure=[org.apache.mahout.common
.distance.SquaredEuclideanDistanceMeasure], --endPhase=[2147483647], --input=[output/clusters-10-final], --output=[clusteranalyze
.txt], --outputFormat=[TEXT], --pointsDir=[output/clusteredPoints], --startPhase=[0], --tempDir=[temp]}
20/03/07 02:54:08 INFO ClusterDumper: Wrote 6 clusters
20/03/07 02:54:08 INFO MahoutDriver: Program took 1710 ms (Minutes: 0.0285)
[ec2-user@ip-172-31-38-169 ~]$
```

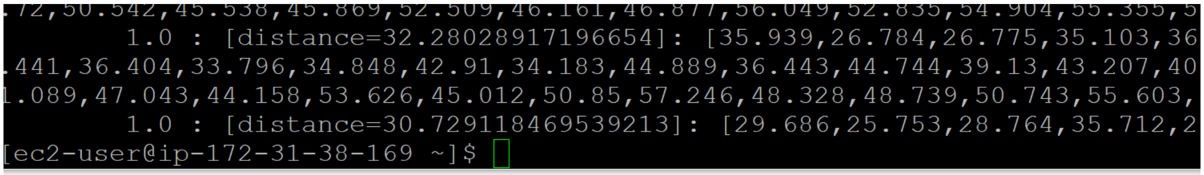
The file clusteranalyze.txt contains the results of the Kmeans run after 10 iterations.

Submit the screenshot of the first page from clusteranalyze.txt (e.g., from more clusteranalyze.txt)

more clusteranalyze.txt



```
bioproject.xml                    dwdate.tbl            lineorder.tbl     pig-0.15.0.tar
[ec2-user@ip-172-31-38-169 ~]$ more clusteranalyze.txt
{"r":[3.659,3.666,2.911,3.336,3.885,3.409,3.564,3.38,3.855,3.895,3.53,3.637,4.173,4.063,4.345,4.074,4.733,4.385,5.028,4.924,4.848
,5.334,5.298,5.706,5.291,5.469,5.038,4.717,5.084,4.415,3.761,4.362,4.206,3.732,4.083,3.938,4.098,4.417,3.967,4.141,3.814,4.133,3.
811,4.001,3.885,4.355,4.104,4.182,4.403,4.234,4.574,4.158,4.448,4.876,4.456,4.409,4.974,4.689,5.208,4.925],"c":[30.125,29.706,30.
494,31.896,31.445,32.689,31.704,32.251,32.654,32.738,33.356,33.61,33.88,34.146,34.028,34.665,35.013,35.707,35.907,35.846,37.208,3
9.062,38.992,39.687,40.511,41.14,42.314,41.279,42.084,43.498,43.04,44.39,43.608,43.837,44.926,45.259,45.269,45.376,45.56,46.084,4
6.632,47.08,47.452,47.246,47.82,48.425,48.837,48.397,48.299,49.408,49.318,49.718,50.716,50.739,50.567,51.405,51.497,51.501,51.79,
51.925],"n":76,"identifier":"VL-454"}
        Weight : [props - optional]:  Point:
        1.0 : [distance=40.121510962400315]: [33.786,29.428,27.377,37.342,26.013,36.203,31.024,37.785,35.754,36.953,32.401,37.258
,37.536,40.414,33.863,33.254,43.233,40.022,34.117,38.453,42.565,37.477,37.266,43.044,39.428,43.563,45.807,49.265,40.014,43.101,40
.967,47.69,47.25,49.214,43.991,51.439,49.912,53.279,45.193,45.813,48.885,55.934,50.823,53.761,48.767,57.682,58.6,54.354,57.292,50
.088,55.553,50.929,56.859,54.562,53.578,62.184,62.853,57.198,63.828,56.127]
        1.0 : [distance=32.790381834986235]: [24.944,31.231,27.187,29.492,32.562,27.971,32.59,32.216,34.967,40.144,37.822,37.343,
36.915,32.309,38.231,36.764,38.671,33.74,42.31,34.92,39.418,39.438,45.279,39.32,43.148,36.097,36.905,44.596,47.829,42.111,39.428,
46.569,40.807,44.658,45.115,44.59,41.423,47.835,53.857,44.565,45.324,54.725,44.234,44.977,55.838,52.971,54.403,54.437,56.912,55.3
66,49.803,50.875,52.817,60.459,54.858,51.21,53.154,56.914,57.193,57.238]
        1.0 : [distance=35.44696249354358]: [24.309,25.021,28.828,36.801,35.288,31.411,36.021,31.506,28.032,32.399,40.066,30.909,
40.848,39.512,42.186,38.253,37.241,31.94,32.407,41.97,38.037,43.011,38.482,38.847,35.326,46.824,40.332,39.543,43.595,47.021,48.15
9,40.292,47.752,41.687,46.825,46.896,46.978,49.747,44.497,50.599,50.357,53.346,48.471,54.013,54.785,53.627,49.751,46.669,53.625,5
8.284,55.884,52.417,58.463,53.906,61.116,52.002,50.094,58.254,53.513,59.472]
        1.0 : [distance=29.864710384679814]: [24.37,27.876,30.794,36.752,30.161,34.208,30.799,37.802,33.658,37.964,34.201,31.489,
33.074,29.063,39.339,40.322,34.684,38.019,41.299,41.739,31.535,38.859,35.895,35.926,42.162,33.023,34.18,33.841,39.712,40.885,41.2
69,44.043,41.098,46.379,43.458,46.668,42.184,42.53,41.612,45.165,44.564,42.871,42.466,39.759,40.43,50.122,45.812,49.335,44.666,51
.675,47.506,49.791,52.686,45.285,48.131,51.817,49.261,44.194,51.162,53.758]
        1.0 : [distance=30.522592692742258]: [32.131,31.711,35.793,27.419,27.617,33.054,36.064,36.964,29.058,30.386,38.765,32.313
,32.508,38.514,32.682,37.712,35.532,40.359,39.945,44.03,44.16,39.778,37.261,42.001,40.865,38.836,43.657,37.247,43.264,43.675,38.1
95,45.961,40.361,40.706,50.959,52.071,44.35,47.271,42.643,45.087,44.759,46.139,48.421,47.248,54.849,52.391,57.017,56.663,49.071,4
9.841,52.534,55.698,53.699,57.587,50.934,59.454,57.07,51.82,53.359,52.761]
        1.0 : [distance=33.628914760843266]: [35.3,32.696,26.872,36.786,29.216,33.532,38.174,34.597,27.976,33.232,31.12,37.324,39
.475,40.481,32.965,37.926,35.162,43.285,41.715,41.986,41.577,42.711,34.716,37.45,40.135,36.166,44.384,39.674,38.027,40.841,41.242
,40.714,41.372,49.402,44.293,46.714,40.813,44.64,42.461,44.351,53.529,49.409,50.862,44.102,55.099,49.759,54.1,53.312,49.772,56.38
5,52.057,54.642,52.383,58.083,56.634,59.473,50.001,60.121,50.345,59.133]
        1.0 : [distance=25.678636934660688]: [29.74,25.589,27.592,26.803,26.792,37.297,31.595,35.279,35.941,31.392,34.278,36.094,
35.631,40.259,41.36,34.375,33.874,35.874,34.566,37.009,39.506,41.6,40.852,38.998,38.457,36.436,42.124,43.112,38.568,38.653,44.7,4
5.539,47.969,39.801,44.165,46.559,38.537,43.642,48.437,43.012,48.225,49.461,51.13,49.544,52.623,41.796,45.836,45.249,48.35,54.633
,45.594,49.112,50.814,47.246,53.784,52.405,51.871,56.813,46.988,47.553]
        1.0 : [distance=28.781611690879696]: [33.732,28.655,25.656,31.965,28.901,31.412,33.624,32.795,37.216,36.401,35.106,37.772
,38.784,32.891,38.877,36.701,33.854,32.929,36.603,37.34,32.838,42.607,36.403,38.414,36.486,38.804,43.887,39.881,36.683,40.521,36.
163,38.18,38.217,45.054,40.497,44.442,47.772,38.464,38.243,49.324,42.545,43.105,46.369,44.655,45.206,50.669,49.07,49.605,40.852,4
4.198,47.569,47.022,43.183,53.764,44.984,54.396,45.536,52.173,53.873,54.602]
        1.0 : [distance=35.51648243457966]: [35.432,30.61,26.088,25.674,29.148,35.054,31.818,36.47,30.972,30.353,34.86,39.275,39.
--More--(1%)
```



```
.72,50.542,45.538,45.869,52.509,46.161,46.877,56.049,52.835,54.904,55.355,5
         1.0 : [distance=32.28028917196654]: [35.939,26.784,26.775,35.103,36
.441,36.404,33.796,34.848,42.91,34.183,44.889,36.443,44.744,39.13,43.207,40
1.089,47.043,44.158,53.626,45.012,50.85,57.246,48.328,48.739,50.743,55.603,
         1.0 : [distance=30.729118469539213]: [29.686,25.753,28.764,35.712,2
[ec2-user@ip-172-31-38-169 ~]$
```

➔ worked

Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 4" at the top.