CSC 555 Final Project Phase 2

Nisarg Patel

1) Data Transformation

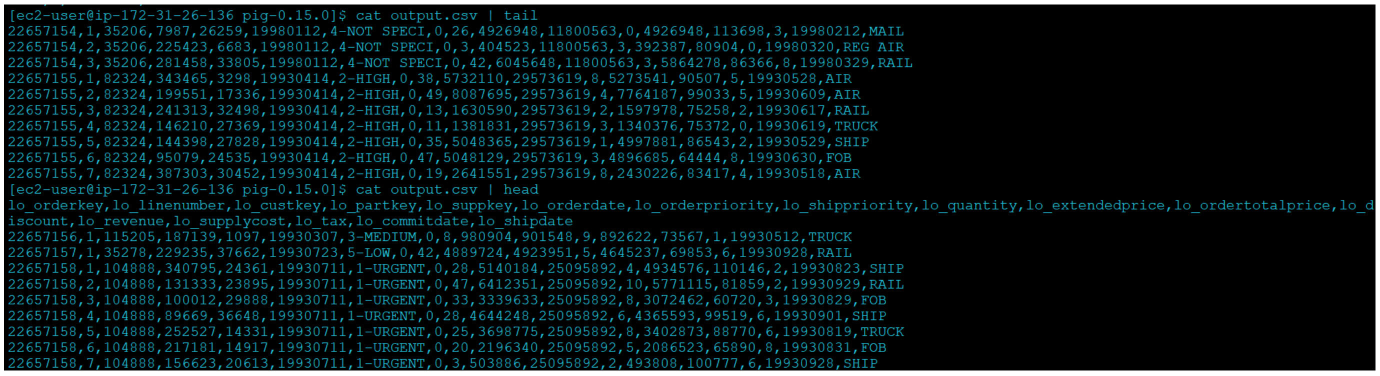
i) <u>Pig</u>

- First copy data from HDFS to Pig Shell as follow:
  grunt>
  lineorder = LOAD '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_shipdate:chararray);

- Now store that output file into HDFS Storage with , as delimiter.
  grunt> store lineorder into '/home/ec2-user/data/lineorder_csv' using PigStorage(‘,' , '-schema');

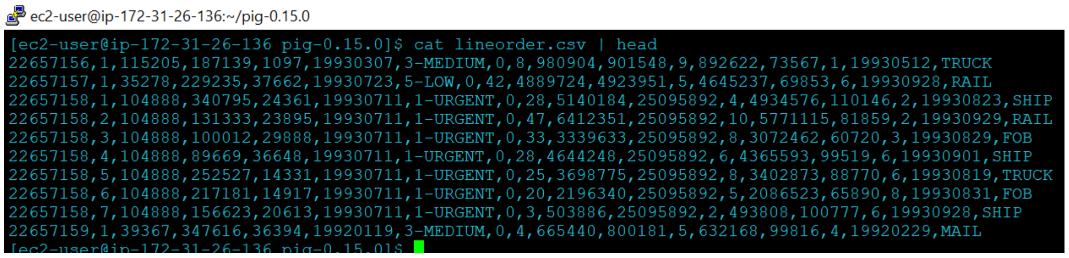
- Now we will remove the pig schema and then store this file to Local directory
  hadoop fs -rm /home/ec2-user/data/lineorder_csv/.pig_schema
  hadoop fs -getmerge /home/ec2-user/data/lineorder_csv /home/ec2-user/output.csv

- Now as you can see from the data head and tail that we got the , (comma) separated file and the columns are no as same as we had in out original file. So we will remove the columns heading that is shown and if we want to use this table we will generate the schema according to this table



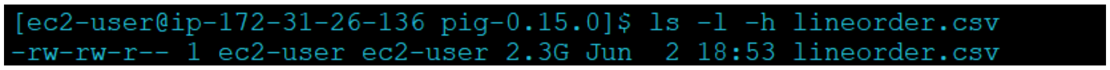
```
[ec2-user@ip-172-31-26-136 pig-0.15.0]$ cat output.csv | tail
22657154,1,35206,7987,26259,19980112,4-NOT SPECI,0,26,4926948,11800563,0,4926948,113698,3,19980212,MAIL
22657154,2,35206,225423,6683,19980112,4-NOT SPECI,0,3,404523,11800563,3,392387,80904,0,19980320,REG AIR
22657154,3,35206,281458,33805,19980112,4-NOT SPECI,0,42,6045648,11800563,3,5864278,86366,8,19980329,RAIL
22657155,1,82324,343465,3298,19930414,2-HIGH,0,38,5732110,29573619,8,5273541,90507,5,19930528,AIR
22657155,2,82324,199551,17336,19930414,2-HIGH,0,49,8087695,29573619,4,7764187,99033,5,19930609,AIR
22657155,3,82324,241313,32498,19930414,2-HIGH,0,13,1630590,29573619,2,1597978,75258,2,19930617,RAIL
22657155,4,82324,146210,27369,19930414,2-HIGH,0,11,1381831,29573619,3,1340376,75372,0,19930619,TRUCK
22657155,5,82324,144398,27828,19930414,2-HIGH,0,35,5048365,29573619,1,4997881,86543,2,19930529,SHIP
22657155,6,82324,95079,24535,19930414,2-HIGH,0,47,5048129,29573619,3,4896685,64444,8,19930630,FOB
22657155,7,82324,387303,30452,19930414,2-HIGH,0,19,2641551,29573619,8,2430226,83417,4,19930518,AIR
[ec2-user@ip-172-31-26-136 pig-0.15.0]$ cat output.csv | head
lo_orderkey,lo_linenumber,lo_custkey,lo_partkey,lo_suppkey,lo_orderdate,lo_orderpriority,lo_shippriority,lo_quantity,lo_extendedprice,lo_ordertotalprice,lo_d
iscount,lo_revenue,lo_supplycost,lo_tax,lo_commitdate,lo_shipdate
22657156,1,115205,187139,1097,19930307,3-MEDIUM,0,8,980904,901548,9,892622,73567,1,19930512,TRUCK
22657157,1,35278,229235,37662,19930723,5-LOW,0,42,4889724,4923951,5,4645237,69853,6,19930928,RAIL
22657158,1,104888,340795,24361,19930711,1-URGENT,0,28,5140184,25095892,4,4934576,110146,2,19930823,SHIP
22657158,2,104888,131333,23895,19930711,1-URGENT,0,47,6412351,25095892,10,5771115,81859,2,19930929,RAIL
22657158,3,104888,100012,29888,19930711,1-URGENT,0,33,3339633,25095892,8,3072462,60720,3,19930829,FOB
22657158,4,104888,89669,36648,19930711,1-URGENT,0,28,4644248,25095892,6,4365593,99519,6,19930901,SHIP
22657158,5,104888,252527,14331,19930711,1-URGENT,0,25,3698775,25095892,8,3402873,88770,6,19930819,TRUCK
22657158,6,104888,217181,14917,19930711,1-URGENT,0,20,2196340,25095892,5,2086523,65890,8,19930831,FOB
22657158,7,104888,156623,20613,19930711,1-URGENT,0,3,503886,25095892,2,493808,100777,6,19930928,SHIP
```

- Now we will remove the first line and then print rest of the data.
  awk 'NR>2 {print}' /home/ec2-user/pig-0.15.0/output.csv > /home/ec2-user/pig-0.15.0/lineorder.csv



```
[ec2-user@ip-172-31-26-136 pig-0.15.0]$ cat lineorder.csv | head
22657156,1,115205,187139,1097,19930307,3-MEDIUM,0,8,980904,901548,9,892622,73567,1,19930512,TRUCK
22657157,1,35278,229235,37662,19930723,5-LOW,0,42,4889724,4923951,5,4645237,69853,6,19930928,RAIL
22657158,1,104888,340795,24361,19930711,1-URGENT,0,28,5140184,25095892,4,4934576,110146,2,19930823,SHIP
22657158,2,104888,131333,23895,19930711,1-URGENT,0,47,6412351,25095892,10,5771115,81859,2,19930929,RAIL
22657158,3,104888,100012,29888,19930711,1-URGENT,0,33,3339633,25095892,8,3072462,60720,3,19930829,FOB
22657158,4,104888,89669,36648,19930711,1-URGENT,0,28,4644248,25095892,6,4365593,99519,6,19930901,SHIP
22657158,5,104888,252527,14331,19930711,1-URGENT,0,25,3698775,25095892,8,3402873,88770,6,19930819,TRUCK
22657158,6,104888,217181,14917,19930711,1-URGENT,0,20,2196340,25095892,5,2086523,65890,8,19930831,FOB
22657158,7,104888,156623,20613,19930711,1-URGENT,0,3,503886,25095892,2,493808,100777,6,19930928,SHIP
22657159,1,39367,347616,36394,19920119,3-MEDIUM,0,4,665440,800181,5,632168,99816,4,19920229,MAIL
[ec2-user@ip-172-31-26-136 pig-0.15.0]$
```

- File size



```
[ec2-user@ip-172-31-26-136 pig-0.15.0]$ ls -l -h lineorder.csv
-rw-rw-r-- 1 ec2-user ec2-user 2.3G Jun  2 18:53 lineorder.csv
```

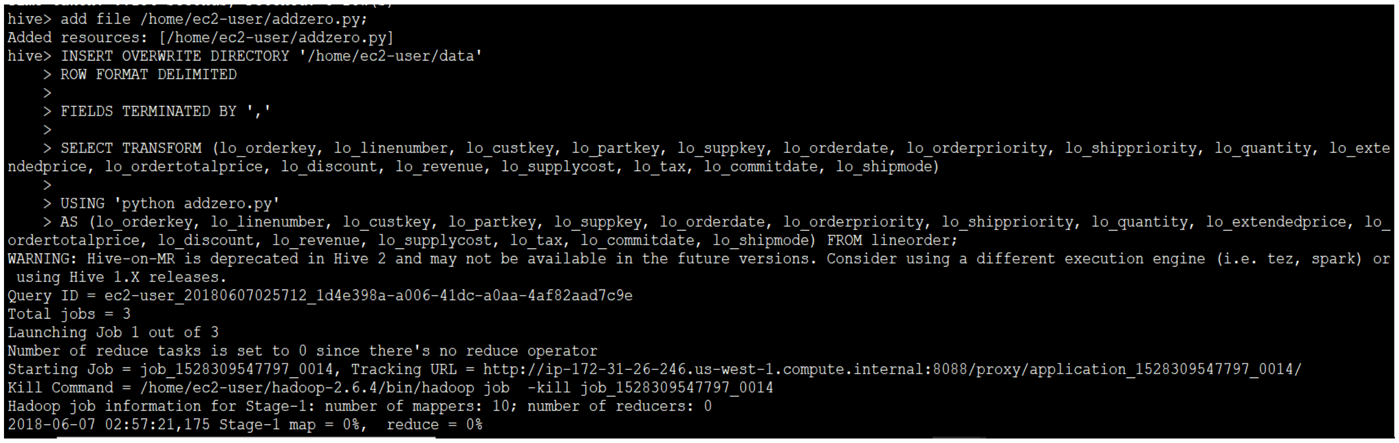
ii) <u>HIVE</u>

- Create table lineorder with the given schema using ‘|’ to seprate the column and add the lineorder.tbl file.
  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;

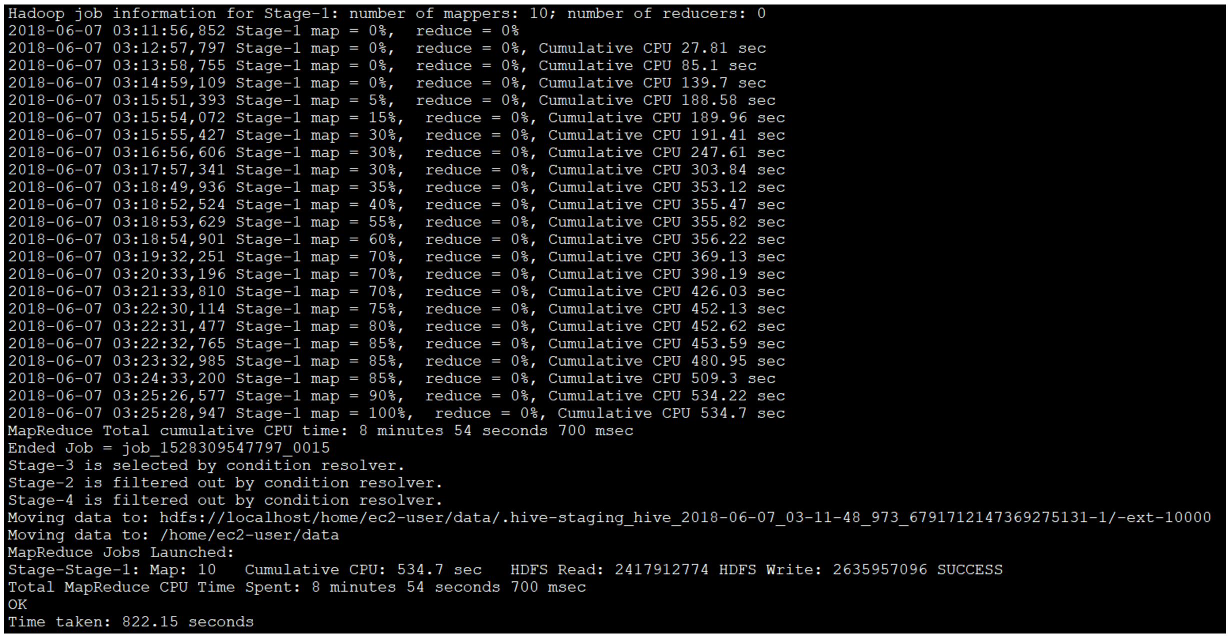
- Adding the file to table

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

- Runing the query show below.

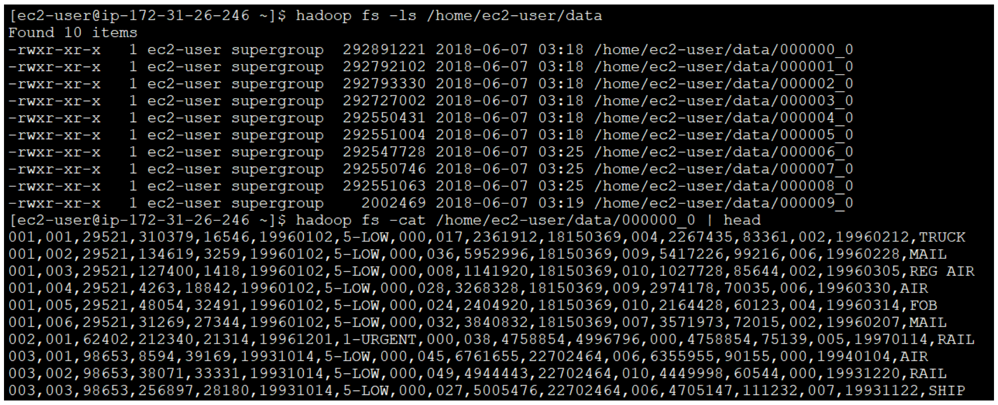


```
hive> add file /home/ec2-user/addzero.py;
Added resources: [/home/ec2-user/addzero.py]
hive> INSERT OVERWRITE DIRECTORY '/home/ec2-user/data'
    > ROW FORMAT DELIMITED
    >
    > FIELDS TERMINATED BY ','
    >
    > SELECT TRANSFORM (lo_orderkey, lo_linenumber, lo_custkey, lo_partkey, lo_suppkey, lo_orderdate, lo_orderpriority, lo_shippriority, lo_quantity, lo_exte
ndedprice, lo_ordertotalprice, lo_discount, lo_revenue, lo_supplycost, lo_tax, lo_commitdate, lo_shipmode)
    >
    > USING 'python addzero.py'
    > AS (lo_orderkey, lo_linenumber, lo_custkey, lo_partkey, lo_suppkey, lo_orderdate, lo_orderpriority, lo_shippriority, lo_quantity, lo_extendedprice, lo_
ordertotalprice, lo_discount, lo_revenue, lo_supplycost, lo_tax, lo_commitdate, lo_shipmode) FROM lineorder;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. tez, spark) or
 using Hive 1.X releases.
Query ID = ec2-user_20180607025712_1d4e398a-a006-41dc-a0aa-4af82aad7c9e
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1528309547797_0014, Tracking URL = http://ip-172-31-26-246.us-west-1.compute.internal:8088/proxy/application_1528309547797_0014/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1528309547797_0014
Hadoop job information for Stage-1: number of mappers: 10; number of reducers: 0
2018-06-07 02:57:21,175 Stage-1 map = 0%,  reduce = 0%
```

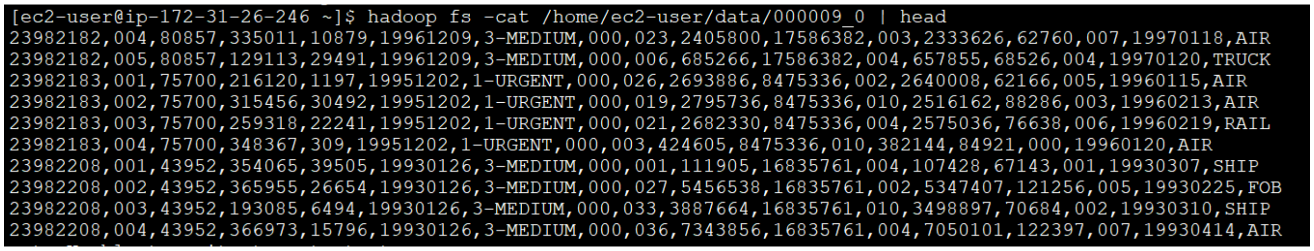


```
Hadoop job information for Stage-1: number of mappers: 10; number of reducers: 0
2018-06-07 03:11:56,852 Stage-1 map = 0%,  reduce = 0%
2018-06-07 03:12:57,797 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 27.81 sec
2018-06-07 03:13:58,755 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 85.1 sec
2018-06-07 03:14:59,109 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 139.7 sec
2018-06-07 03:15:51,393 Stage-1 map = 5%,  reduce = 0%, Cumulative CPU 188.58 sec
2018-06-07 03:15:54,072 Stage-1 map = 15%,  reduce = 0%, Cumulative CPU 189.96 sec
2018-06-07 03:15:55,427 Stage-1 map = 30%,  reduce = 0%, Cumulative CPU 191.41 sec
2018-06-07 03:16:56,606 Stage-1 map = 30%,  reduce = 0%, Cumulative CPU 247.61 sec
2018-06-07 03:17:57,341 Stage-1 map = 30%,  reduce = 0%, Cumulative CPU 303.84 sec
2018-06-07 03:18:49,936 Stage-1 map = 35%,  reduce = 0%, Cumulative CPU 353.12 sec
2018-06-07 03:18:52,524 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 355.47 sec
2018-06-07 03:18:53,629 Stage-1 map = 55%,  reduce = 0%, Cumulative CPU 355.82 sec
2018-06-07 03:18:54,901 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 356.22 sec
2018-06-07 03:19:32,251 Stage-1 map = 70%,  reduce = 0%, Cumulative CPU 369.13 sec
2018-06-07 03:20:33,196 Stage-1 map = 70%,  reduce = 0%, Cumulative CPU 398.19 sec
2018-06-07 03:21:33,810 Stage-1 map = 70%,  reduce = 0%, Cumulative CPU 426.03 sec
2018-06-07 03:22:30,114 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 452.13 sec
2018-06-07 03:22:31,477 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 452.62 sec
2018-06-07 03:22:32,765 Stage-1 map = 85%,  reduce = 0%, Cumulative CPU 453.59 sec
2018-06-07 03:23:32,985 Stage-1 map = 85%,  reduce = 0%, Cumulative CPU 480.95 sec
2018-06-07 03:24:33,200 Stage-1 map = 85%,  reduce = 0%, Cumulative CPU 509.3 sec
2018-06-07 03:25:26,577 Stage-1 map = 90%,  reduce = 0%, Cumulative CPU 534.22 sec
2018-06-07 03:25:28,947 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 534.7 sec
MapReduce Total cumulative CPU time: 8 minutes 54 seconds 700 msec
Ended Job = job_1528309547797_0015
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost/home/ec2-user/data/.hive-staging_hive_2018-06-07_03-11-48_973_6791712147369275131-1/-ext-10000
Moving data to: /home/ec2-user/data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 10   Cumulative CPU: 534.7 sec   HDFS Read: 2417912774 HDFS Write: 2635957096 SUCCESS
Total MapReduce CPU Time Spent: 8 minutes 54 seconds 700 msec
OK
Time taken: 822.15 seconds
```

- This is the generate file to check that go to
  Hadoop fs -ls /home/ec2-user/data

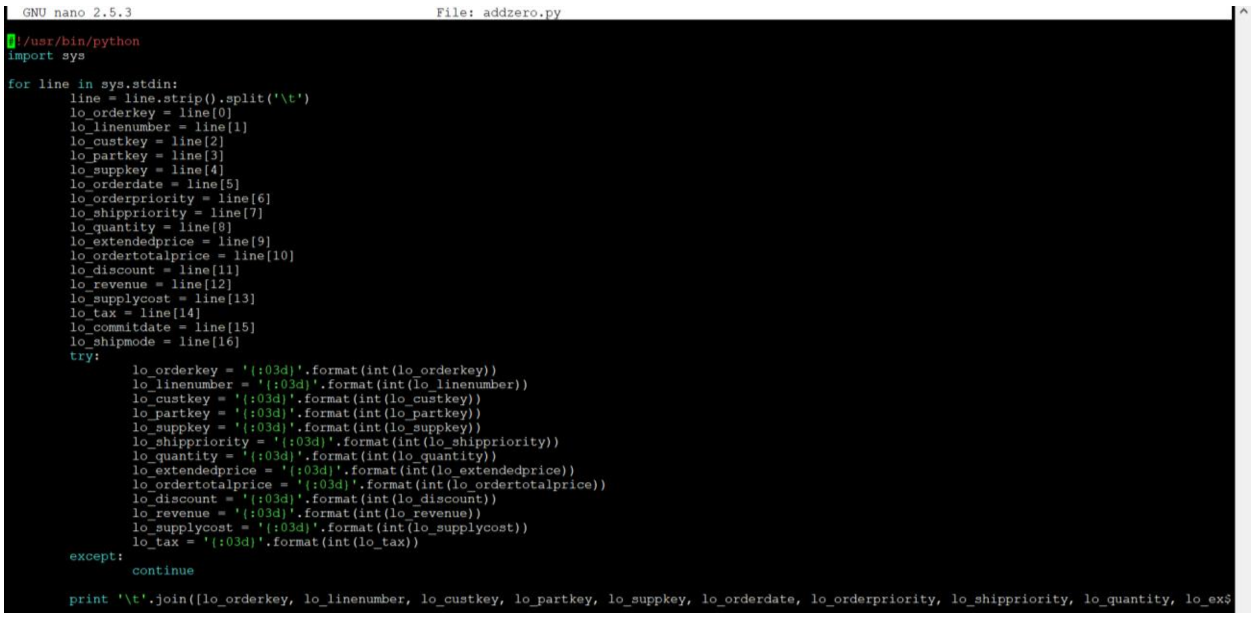


```
[ec2-user@ip-172-31-26-246 ~]$ hadoop fs -ls /home/ec2-user/data
Found 10 items
-rwxr-xr-x   1 ec2-user supergroup  292891221 2018-06-07 03:18 /home/ec2-user/data/000000_0
-rwxr-xr-x   1 ec2-user supergroup  292792102 2018-06-07 03:18 /home/ec2-user/data/000001_0
-rwxr-xr-x   1 ec2-user supergroup  292793330 2018-06-07 03:18 /home/ec2-user/data/000002_0
-rwxr-xr-x   1 ec2-user supergroup  292727002 2018-06-07 03:18 /home/ec2-user/data/000003_0
-rwxr-xr-x   1 ec2-user supergroup  292550431 2018-06-07 03:18 /home/ec2-user/data/000004_0
-rwxr-xr-x   1 ec2-user supergroup  292551004 2018-06-07 03:18 /home/ec2-user/data/000005_0
-rwxr-xr-x   1 ec2-user supergroup  292547728 2018-06-07 03:25 /home/ec2-user/data/000006_0
-rwxr-xr-x   1 ec2-user supergroup  292550746 2018-06-07 03:25 /home/ec2-user/data/000007_0
-rwxr-xr-x   1 ec2-user supergroup  292551063 2018-06-07 03:25 /home/ec2-user/data/000008_0
-rwxr-xr-x   1 ec2-user supergroup    2002469 2018-06-07 03:19 /home/ec2-user/data/000009_0
[ec2-user@ip-172-31-26-246 ~]$ hadoop fs -cat /home/ec2-user/data/000000_0 | head
001,001,29521,310379,16546,19960102,5-LOW,000,017,2361912,18150369,004,2267435,83361,002,19960212,TRUCK
001,002,29521,134619,3259,19960102,5-LOW,000,036,5952996,18150369,009,5417226,99216,006,19960228,MAIL
001,003,29521,127400,1418,19960102,5-LOW,000,008,1141920,18150369,010,1027728,85644,002,19960305,REG AIR
001,004,29521,4263,18842,19960102,5-LOW,000,028,3268328,18150369,009,2974178,70035,006,19960330,AIR
001,005,29521,48054,32491,19960102,5-LOW,000,024,2404920,18150369,010,2164428,60123,004,19960314,FOB
001,006,29521,31269,27344,19960102,5-LOW,000,032,3840832,18150369,007,3571973,72015,002,19960207,MAIL
002,001,62402,212340,21314,19961201,1-URGENT,000,038,4758854,4996796,000,4758854,75139,005,19970114,RAIL
003,001,98653,8594,39169,19931014,5-LOW,000,045,6761655,22702464,006,6355955,90155,000,19940104,AIR
003,002,98653,38071,33331,19931014,5-LOW,000,049,4944443,22702464,010,4449998,60544,000,19931220,RAIL
003,003,98653,256897,28180,19931014,5-LOW,000,027,5005476,22702464,006,4705147,111232,007,19931122,SHIP
```



```
[ec2-user@ip-172-31-26-246 ~]$ hadoop fs -cat /home/ec2-user/data/000009_0 | head
23982182,004,80857,335011,10879,19961209,3-MEDIUM,000,023,2405800,17586382,003,2333626,62760,007,19970118,AIR
23982182,005,80857,129113,29491,19961209,3-MEDIUM,000,006,685266,17586382,004,657855,68526,004,19970120,TRUCK
23982183,001,75700,216120,1197,19951202,1-URGENT,000,026,2693886,8475336,002,2640008,62166,005,19960115,AIR
23982183,002,75700,315456,30492,19951202,1-URGENT,000,019,2795736,8475336,010,2516162,88286,003,19960213,AIR
23982183,003,75700,259318,22241,19951202,1-URGENT,000,021,2682330,8475336,004,2575036,76638,006,19960219,RAIL
23982183,004,75700,348367,309,19951202,1-URGENT,000,003,424605,8475336,010,382144,84921,000,19960120,AIR
23982208,001,43952,354065,39505,19930126,3-MEDIUM,000,001,111905,16835761,004,107428,67143,001,19930307,SHIP
23982208,002,43952,365955,26654,19930126,3-MEDIUM,000,027,5456538,16835761,002,5347407,121256,005,19930225,FOB
23982208,003,43952,193085,6494,19930126,3-MEDIUM,000,033,3887664,16835761,010,3498897,70684,002,19930310,SHIP
23982208,004,43952,366973,15796,19930126,3-MEDIUM,000,036,7343856,16835761,004,7050101,122397,007,19930414,AIR
```

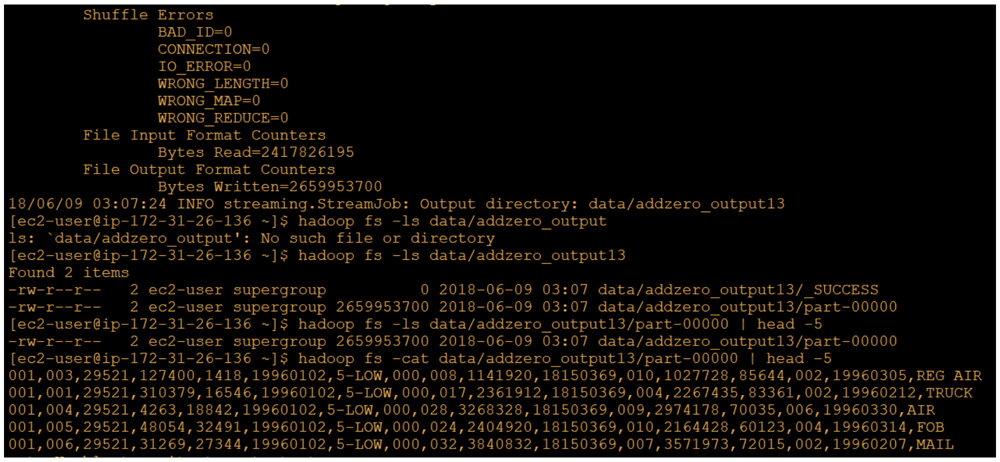
- Python file
  addzero.py



```
  GNU nano 2.5.3                                File: addzero.py

#!/usr/bin/python
import sys

for line in sys.stdin:
        line = line.strip().split('\t')
        lo_orderkey = line[0]
        lo_linenumber = line[1]
        lo_custkey = line[2]
        lo_partkey = line[3]
        lo_suppkey = line[4]
        lo_orderdate = line[5]
        lo_orderpriority = line[6]
        lo_shippriority = line[7]
        lo_quantity = line[8]
        lo_extendedprice = line[9]
        lo_ordertotalprice = line[10]
        lo_discount = line[11]
        lo_revenue = line[12]
        lo_supplycost = line[13]
        lo_tax = line[14]
        lo_commitdate = line[15]
        lo_shipmode = line[16]
        try:
                lo_orderkey = '{:03d}'.format(int(lo_orderkey))
                lo_linenumber = '{:03d}'.format(int(lo_linenumber))
                lo_custkey = '{:03d}'.format(int(lo_custkey))
                lo_partkey = '{:03d}'.format(int(lo_partkey))
                lo_suppkey = '{:03d}'.format(int(lo_suppkey))
                lo_shippriority = '{:03d}'.format(int(lo_shippriority))
                lo_quantity = '{:03d}'.format(int(lo_quantity))
                lo_extendedprice = '{:03d}'.format(int(lo_extendedprice))
                lo_ordertotalprice = '{:03d}'.format(int(lo_ordertotalprice))
                lo_discount = '{:03d}'.format(int(lo_discount))
                lo_revenue = '{:03d}'.format(int(lo_revenue))
                lo_supplycost = '{:03d}'.format(int(lo_supplycost))
                lo_tax = '{:03d}'.format(int(lo_tax))
        except:
                continue

        print '\t'.join([lo_orderkey, lo_linenumber, lo_custkey, lo_partkey, lo_suppkey, lo_orderdate, lo_orderpriority, lo_shippriority, lo_quantity, lo_ex$
```

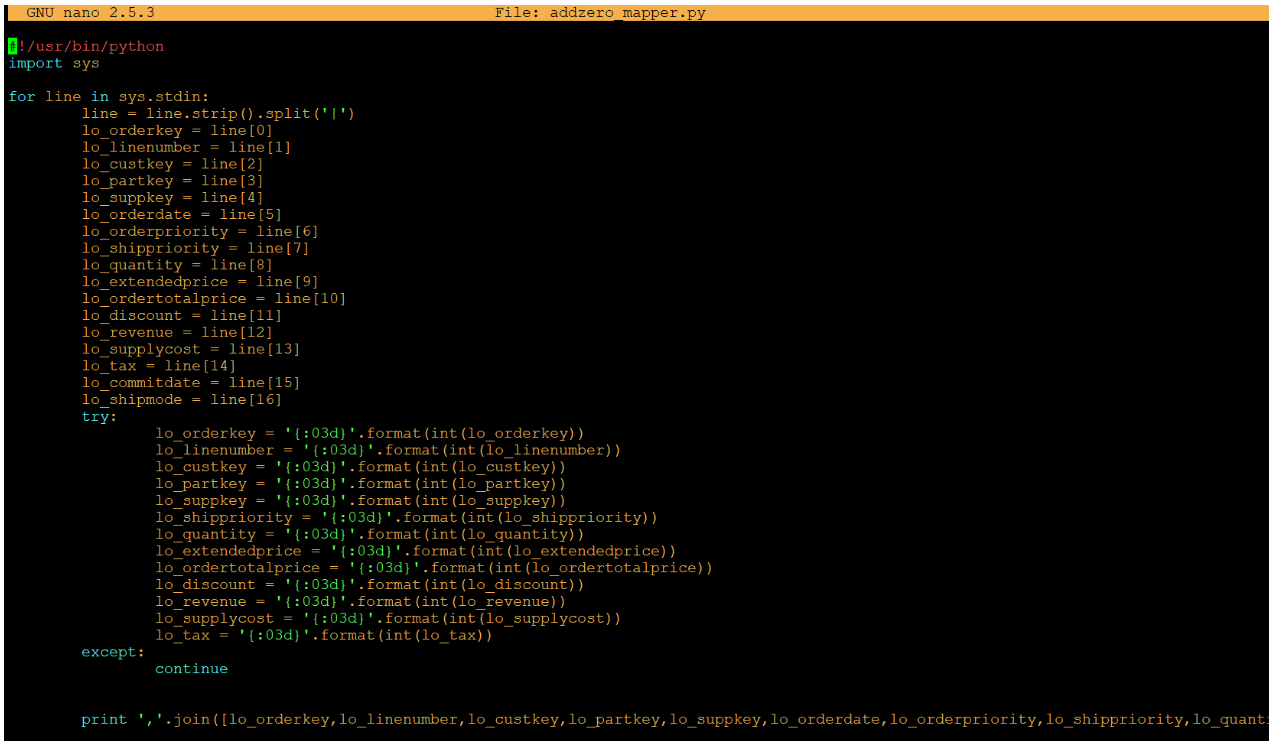
iii) <u>Map Reduce</u>

hadoop jar ./hadoop-2.6.4/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input data/addzero/lineorder.tbl -mapper addzero_mapper.py -file addzero_mapper.py -reducer addzero_reducer.py -file addzero_reducer.py -output data/addzero_output13



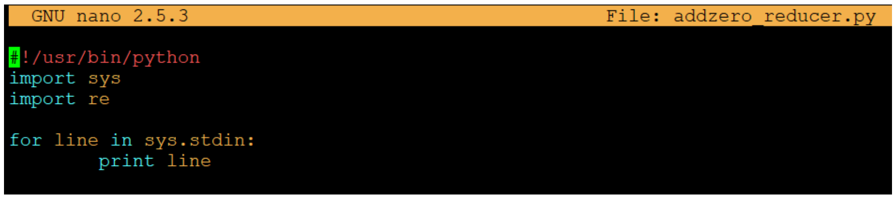
```
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2417826195
        File Output Format Counters
                Bytes Written=2659953700
18/06/09 03:07:24 INFO streaming.StreamJob: Output directory: data/addzero_output13
[ec2-user@ip-172-31-26-136 ~]$ hadoop fs -ls data/addzero_output
ls: `data/addzero_output': No such file or directory
[ec2-user@ip-172-31-26-136 ~]$ hadoop fs -ls data/addzero_output13
Found 2 items
-rw-r--r--   2 ec2-user supergroup          0 2018-06-09 03:07 data/addzero_output13/_SUCCESS
-rw-r--r--   2 ec2-user supergroup 2659953700 2018-06-09 03:07 data/addzero_output13/part-00000
[ec2-user@ip-172-31-26-136 ~]$ hadoop fs -ls data/addzero_output13/part-00000 | head -5
-rw-r--r--   2 ec2-user supergroup 2659953700 2018-06-09 03:07 data/addzero_output13/part-00000
[ec2-user@ip-172-31-26-136 ~]$ hadoop fs -cat data/addzero_output13/part-00000 | head -5
001,003,29521,127400,1418,19960102,5-LOW,000,008,1141920,18150369,010,1027728,85644,002,19960305,REG AIR
001,001,29521,310379,16546,19960102,5-LOW,000,017,2361912,18150369,004,2267435,83361,002,19960212,TRUCK
001,004,29521,4263,18842,19960102,5-LOW,000,028,3268328,18150369,009,2974178,70035,006,19960330,AIR
001,005,29521,48054,32491,19960102,5-LOW,000,024,2404920,18150369,010,2164428,60123,004,19960314,FOB
001,006,29521,31269,27344,19960102,5-LOW,000,032,3840832,18150369,007,3571973,72015,002,19960207,MAIL
```

- Python file for mapper and reducer
  addzero_mapper.py



```
 GNU nano 2.5.3                                         File: addzero_mapper.py

#!/usr/bin/python
import sys

for line in sys.stdin:
        line = line.strip().split('|')
        lo_orderkey = line[0]
        lo_linenumber = line[1]
        lo_custkey = line[2]
        lo_partkey = line[3]
        lo_suppkey = line[4]
        lo_orderdate = line[5]
        lo_orderpriority = line[6]
        lo_shippriority = line[7]
        lo_quantity = line[8]
        lo_extendedprice = line[9]
        lo_ordertotalprice = line[10]
        lo_discount = line[11]
        lo_revenue = line[12]
        lo_supplycost = line[13]
        lo_tax = line[14]
        lo_commitdate = line[15]
        lo_shipmode = line[16]
        try:
                lo_orderkey = '{:03d}'.format(int(lo_orderkey))
                lo_linenumber = '{:03d}'.format(int(lo_linenumber))
                lo_custkey = '{:03d}'.format(int(lo_custkey))
                lo_partkey = '{:03d}'.format(int(lo_partkey))
                lo_suppkey = '{:03d}'.format(int(lo_suppkey))
                lo_shippriority = '{:03d}'.format(int(lo_shippriority))
                lo_quantity = '{:03d}'.format(int(lo_quantity))
                lo_extendedprice = '{:03d}'.format(int(lo_extendedprice))
                lo_ordertotalprice = '{:03d}'.format(int(lo_ordertotalprice))
                lo_discount = '{:03d}'.format(int(lo_discount))
                lo_revenue = '{:03d}'.format(int(lo_revenue))
                lo_supplycost = '{:03d}'.format(int(lo_supplycost))
                lo_tax = '{:03d}'.format(int(lo_tax))
        except:
                continue


        print ','.join([lo_orderkey,lo_linenumber,lo_custkey,lo_partkey,lo_suppkey,lo_orderdate,lo_orderpriority,lo_shippriority,lo_quant
```

addzero_reducer.py



```
 GNU nano 2.5.3                         File: addzero_reducer.py

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

for line in sys.stdin:
        print line
```

2) **Quering**

i) <u>Hive query output</u>

Code :
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;

CREATE TABLE part (p_partkey int, p_name varchar(22), p_mfgr varchar(6), p_category varchar(7), p_brand1 varchar(9), p_color varchar(11), p_type varchar(25), p_size int, p_container varchar(10))

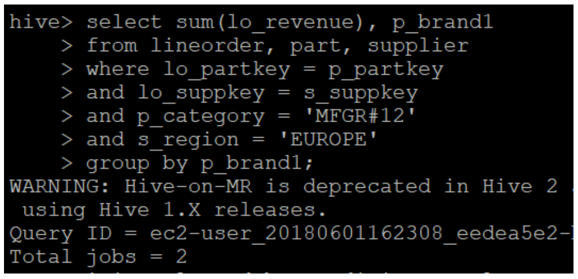
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

CREATE TABLE supplier (s_suppkey int, s_name varchar(25), s_address varchar(25), s_city varchar(10), s_nation varchar(15), s_region varchar(12), s_phone varchar(15))
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' STORED AS TEXTFILE;

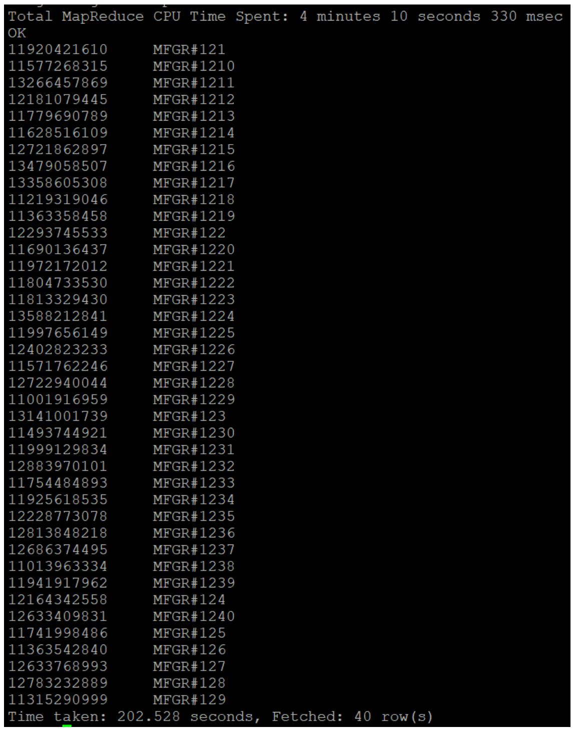
LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl'
OVERWRITE INTO TABLE lineorder;

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

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



```
hive> select sum(lo_revenue), p_brand1
    > from lineorder, part, supplier
    > where lo_partkey = p_partkey
    > and lo_suppkey = s_suppkey
    > and p_category = 'MFGR#12'
    > and s_region = 'EUROPE'
    > group by p_brand1;
WARNING: Hive-on-MR is deprecated in Hive 2
 using Hive 1.X releases.
Query ID = ec2-user_20180601162308_eedea5e2-
Total jobs = 2
```



```
Total MapReduce CPU Time Spent: 4 minutes 10 seconds 330 msec
OK
11920421610     MFGR#121
11577268315     MFGR#1210
13266457869     MFGR#1211
12181079445     MFGR#1212
11779690789     MFGR#1213
11628516109     MFGR#1214
12721862897     MFGR#1215
13479058507     MFGR#1216
13358605308     MFGR#1217
11219319046     MFGR#1218
11363358458     MFGR#1219
12293745533     MFGR#122
11690136437     MFGR#1220
11972172012     MFGR#1221
11804733530     MFGR#1222
11813329430     MFGR#1223
13588212841     MFGR#1224
11997656149     MFGR#1225
12402823233     MFGR#1226
11571762246     MFGR#1227
12722940044     MFGR#1228
11001916959     MFGR#1229
13141001739     MFGR#123
11493744921     MFGR#1230
11999129834     MFGR#1231
12883970101     MFGR#1232
11754484893     MFGR#1233
11925618535     MFGR#1234
12228773078     MFGR#1235
12813848218     MFGR#1236
12686374495     MFGR#1237
11013963334     MFGR#1238
11941917962     MFGR#1239
12164342558     MFGR#124
12633409831     MFGR#1240
11741998486     MFGR#125
11363542840     MFGR#126
12633768993     MFGR#127
12783232889     MFGR#128
11315290999     MFGR#129
Time taken: 202.528 seconds, Fetched: 40 row(s)
```

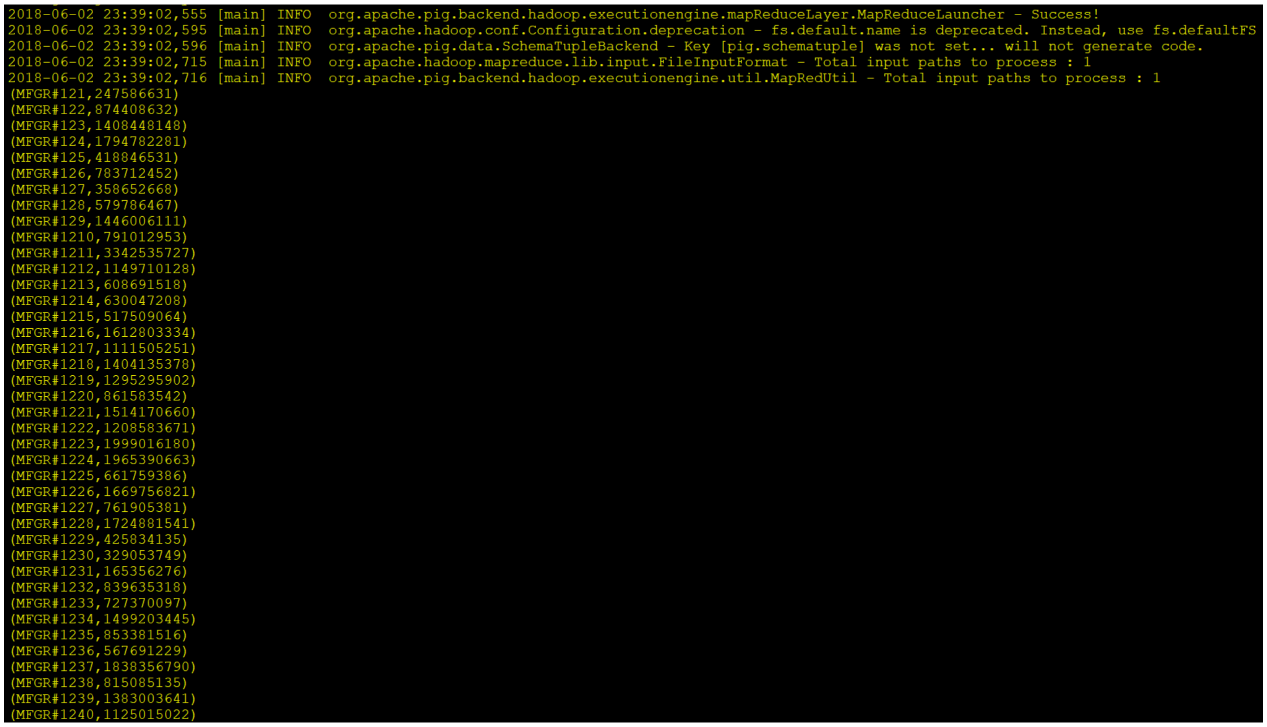
ii) <u>Pig</u>

grunt> supplier = LOAD 'supplier.tbl' USING PigStorage('|') AS (s_suppkey:int,s_name:chararray ,s_address:chararray,s_city:chararray,s_nation:chararray,s_region:chararray,s_phone:chararray);

grunt> part = LOAD 'part.tbl' USING PigStorage('|') AS (p_partkey:int, p_name:chararray, p_mfgr:chararray,p_category:chararray,p_brand1:chararray,p_color:chararray,p_type:chararray, p_size:int,p_container:chararray);

grunt> lineorder = LOAD '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_shipdate:chararray);

grunt> group_join = JOIN lineorder BY lo_partkey , part BY p_partkey , supplier BY s_suppkey;
grunt> describe group_join;
grunt> final_filter = FILTER group_join BY part::p_category == 'MFGR#12' AND supplier::s_region == 'EUROPE' ;
grunt> group_part = group final_filter BY part::p_brand1;
grunt> group_out = FOREACH group_part GENERATE group ,SUM(final_filter.lo_revenue);



```
2018-06-02 23:39:02,555 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2018-06-02 23:39:02,595 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2018-06-02 23:39:02,596 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2018-06-02 23:39:02,715 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2018-06-02 23:39:02,716 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(MFGR#121,247586631)
(MFGR#122,874408632)
(MFGR#123,1408448148)
(MFGR#124,1794782281)
(MFGR#125,418846531)
(MFGR#126,783712452)
(MFGR#127,358652668)
(MFGR#128,579786467)
(MFGR#129,1446006111)
(MFGR#1210,791012953)
(MFGR#1211,3342535727)
(MFGR#1212,1149710128)
(MFGR#1213,608691518)
(MFGR#1214,630047208)
(MFGR#1215,517509064)
(MFGR#1216,1612803334)
(MFGR#1217,1111505251)
(MFGR#1218,1404135378)
(MFGR#1219,1295295902)
(MFGR#1220,861583542)
(MFGR#1221,1514170660)
(MFGR#1222,1208583671)
(MFGR#1223,1999016180)
(MFGR#1224,1965390663)
(MFGR#1225,661759386)
(MFGR#1226,1669756821)
(MFGR#1227,761905381)
(MFGR#1228,1724881541)
(MFGR#1229,425834135)
(MFGR#1230,329053749)
(MFGR#1231,165356276)
(MFGR#1232,839635318)
(MFGR#1233,727370097)
(MFGR#1234,1499203445)
(MFGR#1235,853381516)
(MFGR#1236,567691229)
(MFGR#1237,1838356790)
(MFGR#1238,815085135)
(MFGR#1239,1383003641)
(MFGR#1240,1125015022)
```

iii) **<u>Map Reduce</u>**

This map reduce can be perform in three passes

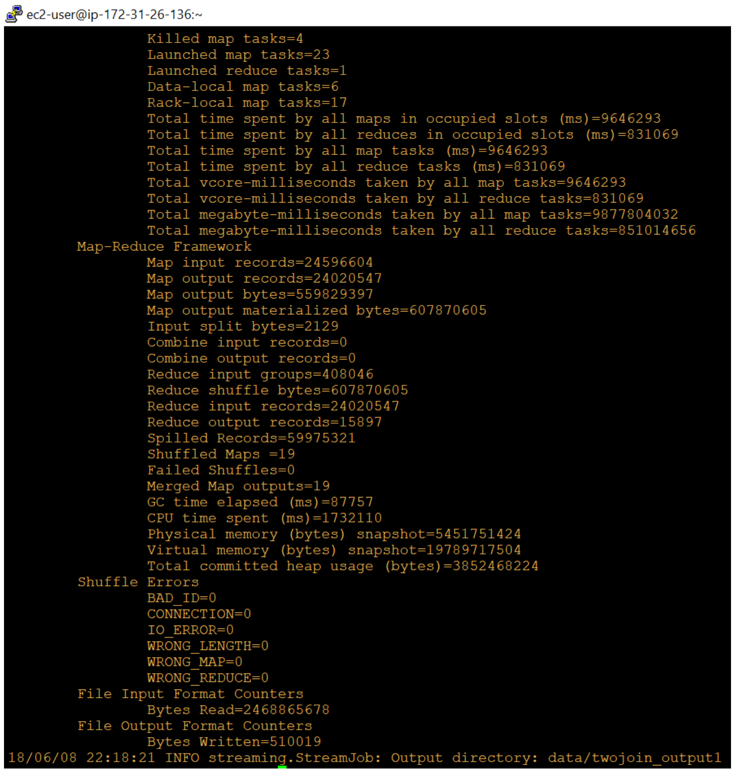
i) In first pass we will join lineorder table and part table.

ii) In second pass we will join the output of first pass and supplier table.

iii) In third pass atlast add all the revenue that has same brand

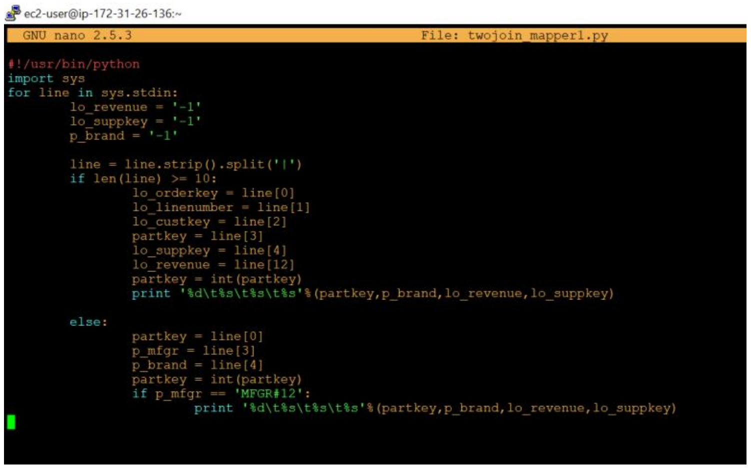
First pass

hadoop jar ./hadoop-2.6.4/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input data/addzero/lineorder.tbl part.tbl -mapper twojoin_mapper1.py -file twojoin_mapper1.py -reducer twojoin_reducer1.py -file twojoin_reducer1.py -output data/twojoin_output1



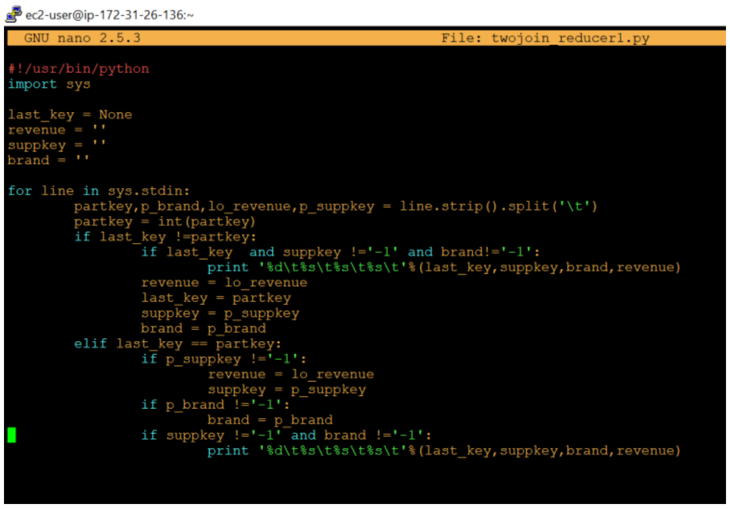
ec2-user@ip-172-31-26-136:~

```
                Killed map tasks=4
                Launched map tasks=23
                Launched reduce tasks=1
                Data-local map tasks=6
                Rack-local map tasks=17
                Total time spent by all maps in occupied slots (ms)=9646293
                Total time spent by all reduces in occupied slots (ms)=831069
                Total time spent by all map tasks (ms)=9646293
                Total time spent by all reduce tasks (ms)=831069
                Total vcore-milliseconds taken by all map tasks=9646293
                Total vcore-milliseconds taken by all reduce tasks=831069
                Total megabyte-milliseconds taken by all map tasks=9877804032
                Total megabyte-milliseconds taken by all reduce tasks=851014656
        Map-Reduce Framework
                Map input records=24596604
                Map output records=24020547
                Map output bytes=559829397
                Map output materialized bytes=607870605
                Input split bytes=2129
                Combine input records=0
                Combine output records=0
                Reduce input groups=408046
                Reduce shuffle bytes=607870605
                Reduce input records=24020547
                Reduce output records=15897
                Spilled Records=59975321
                Shuffled Maps =19
                Failed Shuffles=0
                Merged Map outputs=19
                GC time elapsed (ms)=87757
                CPU time spent (ms)=1732110
                Physical memory (bytes) snapshot=5451751424
                Virtual memory (bytes) snapshot=19789717504
                Total committed heap usage (bytes)=3852468224
        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=2468865678
        File Output Format Counters
                Bytes Written=510019
18/06/08 22:18:21 INFO streaming.StreamJob: Output directory: data/twojoin_output1
```



ec2-user@ip-172-31-26-136:~

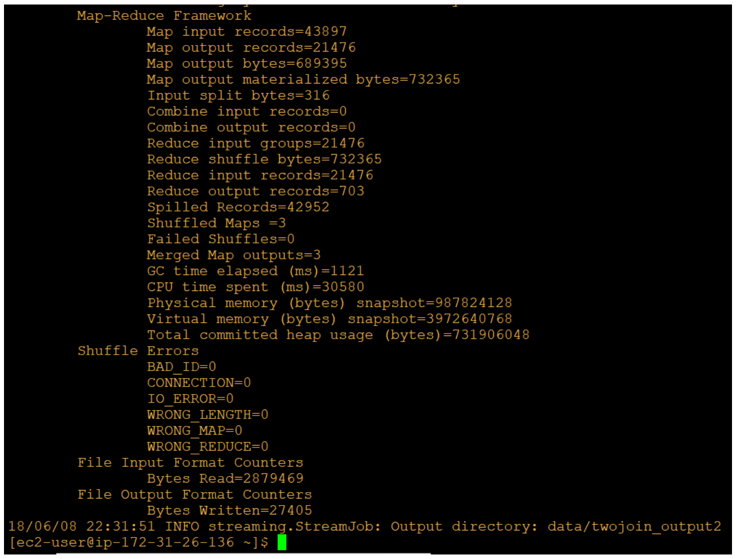
```
  GNU nano 2.5.3                                         File: twojoin_mapper1.py

#!/usr/bin/python
import sys
for line in sys.stdin:
        lo_revenue = '-1'
        lo_suppkey = '-1'
        p_brand = '-1'

        line = line.strip().split('|')
        if len(line) >= 10:
                lo_orderkey = line[0]
                lo_linenumber = line[1]
                lo_custkey = line[2]
                partkey = line[3]
                lo_suppkey = line[4]
                lo_revenue = line[12]
                partkey = int(partkey)
                print '%d\t%s\t%s\t%s'%(partkey,p_brand,lo_revenue,lo_suppkey)

        else:
                partkey = line[0]
                p_mfgr = line[3]
                p_brand = line[4]
                partkey = int(partkey)
                if p_mfgr == 'MFGR#12':
                        print '%d\t%s\t%s\t%s'%(partkey,p_brand,lo_revenue,lo_suppkey)
```



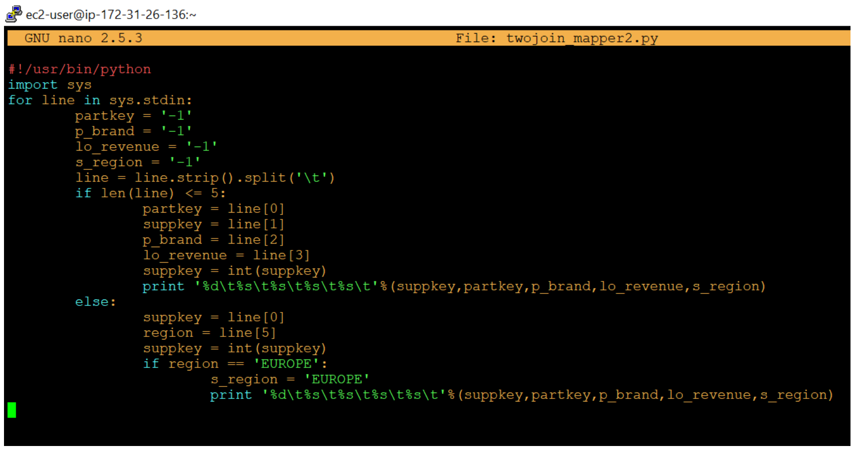
```
ec2-user@ip-172-31-26-136:~
  GNU nano 2.5.3                                     File: twojoin_reducer1.py

#!/usr/bin/python
import sys

last_key = None
revenue = ''
suppkey = ''
brand = ''

for line in sys.stdin:
        partkey,p_brand,lo_revenue,p_suppkey = line.strip().split('\t')
        partkey = int(partkey)
        if last_key !=partkey:
                if last_key  and suppkey !='-1' and brand!='-1':
                        print '%d\t%s\t%s\t%s\t'%(last_key,suppkey,brand,revenue)
                revenue = lo_revenue
                last_key = partkey
                suppkey = p_suppkey
                brand = p_brand
        elif last_key == partkey:
                if p_suppkey !='-1':
                        revenue = lo_revenue
                        suppkey = p_suppkey
                if p_brand !='-1':
                        brand = p_brand
                if suppkey !='-1' and brand !='-1':
                        print '%d\t%s\t%s\t%s\t'%(last_key,suppkey,brand,revenue)
```

Second Pass

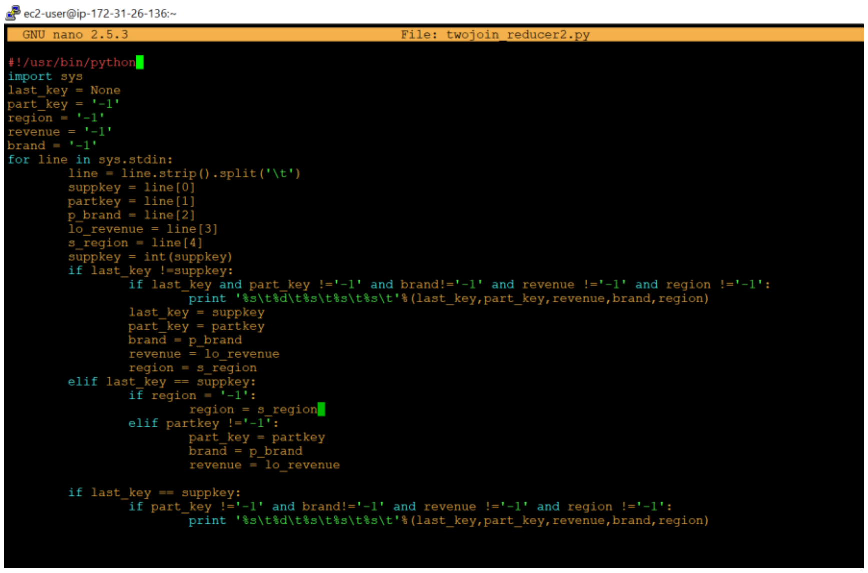
hadoop jar ./hadoop-2.6.4/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input data/twojoin_output1/part-00000 -mapper twojoin_mapper2.py -file twojoin_mapper2.py -reducer twojoin_reducer2.py -file twojoin_reducer2.py -output data/twojoin_output2



```
        Map-Reduce Framework
                Map input records=43897
                Map output records=21476
                Map output bytes=689395
                Map output materialized bytes=732365
                Input split bytes=316
                Combine input records=0
                Combine output records=0
                Reduce input groups=21476
                Reduce shuffle bytes=732365
                Reduce input records=21476
                Reduce output records=703
                Spilled Records=42952
                Shuffled Maps =3
                Failed Shuffles=0
                Merged Map outputs=3
                GC time elapsed (ms)=1121
                CPU time spent (ms)=30580
                Physical memory (bytes) snapshot=987824128
                Virtual memory (bytes) snapshot=3972640768
                Total committed heap usage (bytes)=731906048
        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=2879469
        File Output Format Counters
                Bytes Written=27405
18/06/08 22:31:51 INFO streaming.StreamJob: Output directory: data/twojoin_output2
[ec2-user@ip-172-31-26-136 ~]$
```



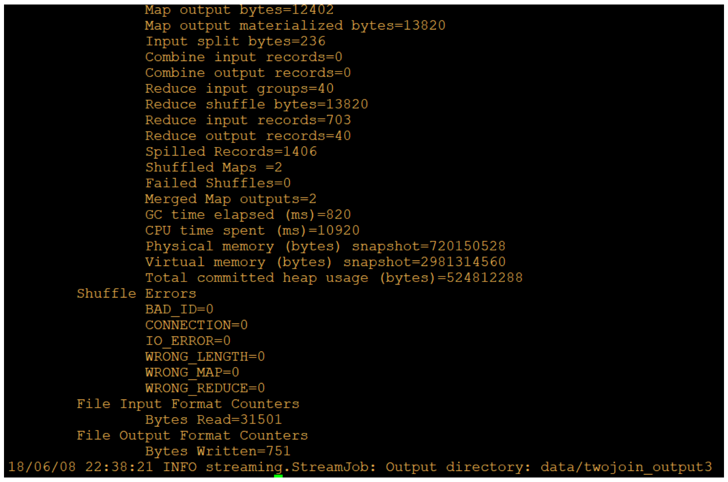
```
ec2-user@ip-172-31-26-136:~
  GNU nano 2.5.3                                         File: twojoin_mapper2.py

#!/usr/bin/python
import sys
for line in sys.stdin:
        partkey = '-1'
        p_brand = '-1'
        lo_revenue = '-1'
        s_region = '-1'
        line = line.strip().split('\t')
        if len(line) <= 5:
                partkey = line[0]
                suppkey = line[1]
                p_brand = line[2]
                lo_revenue = line[3]
                suppkey = int(suppkey)
                print '%d\t%s\t%s\t%s\t%s\t'%(suppkey,partkey,p_brand,lo_revenue,s_region)
        else:
                suppkey = line[0]
                region = line[5]
                suppkey = int(suppkey)
                if region == 'EUROPE':
                        s_region = 'EUROPE'
                        print '%d\t%s\t%s\t%s\t%s\t'%(suppkey,partkey,p_brand,lo_revenue,s_region)
```



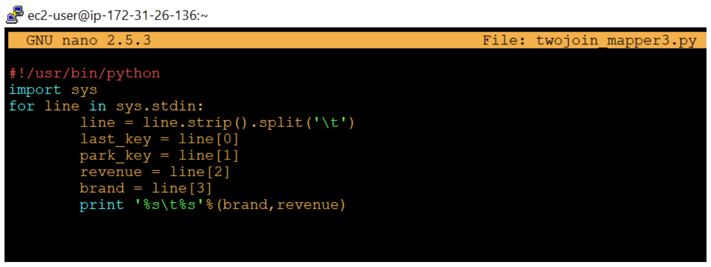
```
ec2-user@ip-172-31-26-136:~
  GNU nano 2.5.3                                         File: twojoin_reducer2.py

#!/usr/bin/python
import sys
last_key = None
part_key = '-1'
region = '-1'
revenue = '-1'
brand = '-1'
for line in sys.stdin:
        line = line.strip().split('\t')
        suppkey = line[0]
        partkey = line[1]
        p_brand = line[2]
        lo_revenue = line[3]
        s_region = line[4]
        suppkey = int(suppkey)
        if last_key !=suppkey:
                if last_key and part_key !='-1' and brand!='-1' and revenue !='-1' and region !='-1':
                        print '%s\t%d\t%s\t%s\t%s\t'%(last_key,part_key,revenue,brand,region)
                last_key = suppkey
                part_key = partkey
                brand = p_brand
                revenue = lo_revenue
                region = s_region
        elif last_key == suppkey:
                if region = '-1':
                        region = s_region
                elif partkey !='-1':
                        part_key = partkey
                        brand = p_brand
                        revenue = lo_revenue

        if last_key == suppkey:
                if part_key !='-1' and brand!='-1' and revenue !='-1' and region !='-1':
                        print '%s\t%d\t%s\t%s\t%s\t'%(last_key,part_key,revenue,brand,region)
```

Third Pass

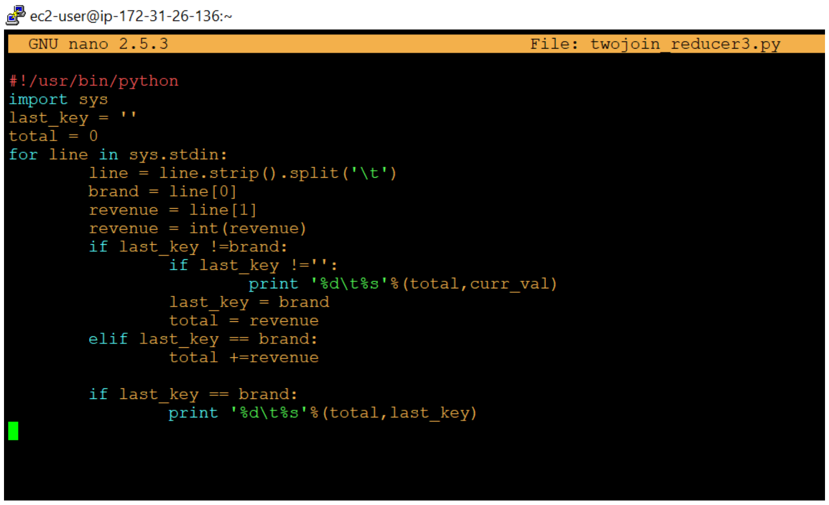
hadoop jar ./hadoop-2.6.4/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -input data/twojoin_output2/part-00000 -mapper twojoin_mapper3.py -file twojoin_mapper3.py -reducer twojoin_reducer3.py -file twojoin_reducer3.py -output data/twojoin_output3



```
                Map output bytes=12402
                Map output materialized bytes=13820
                Input split bytes=236
                Combine input records=0
                Combine output records=0
                Reduce input groups=40
                Reduce shuffle bytes=13820
                Reduce input records=703
                Reduce output records=40
                Spilled Records=1406
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=820
                CPU time spent (ms)=10920
                Physical memory (bytes) snapshot=720150528
                Virtual memory (bytes) snapshot=2981314560
                Total committed heap usage (bytes)=524812288
        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=31501
        File Output Format Counters
                Bytes Written=751
18/06/08 22:38:21 INFO streaming.StreamJob: Output directory: data/twojoin_output3
```

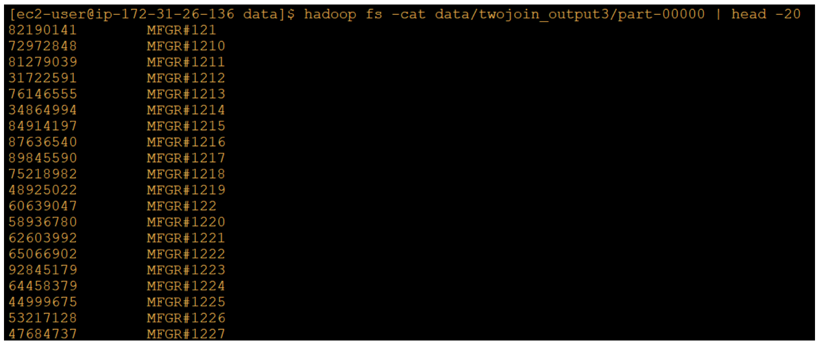


```
ec2-user@ip-172-31-26-136:~
  GNU nano 2.5.3                                      File: twojoin_mapper3.py

#!/usr/bin/python
import sys
for line in sys.stdin:
        line = line.strip().split('\t')
        last_key = line[0]
        park_key = line[1]
        revenue = line[2]
        brand = line[3]
        print '%s\t%s'%(brand,revenue)
```



```
ec2-user@ip-172-31-26-136:~
  GNU nano 2.5.3                                      File: twojoin_reducer3.py

#!/usr/bin/python
import sys
last_key = ''
total = 0
for line in sys.stdin:
        line = line.strip().split('\t')
        brand = line[0]
        revenue = line[1]
        revenue = int(revenue)
        if last_key !=brand:
                if last_key !='':
                        print '%d\t%s'%(total,curr_val)
                last_key = brand
                total = revenue
        elif last_key == brand:
                total +=revenue

        if last_key == brand:
                print '%d\t%s'%(total,last_key)
```

Output

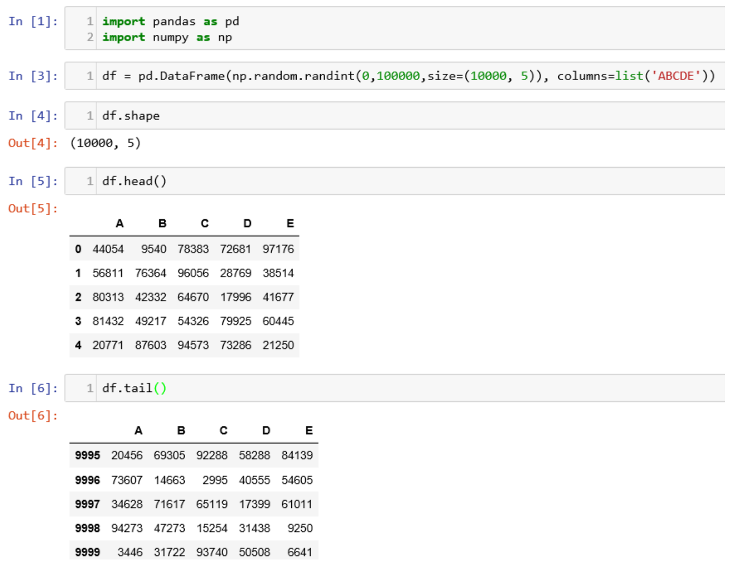


```
[ec2-user@ip-172-31-26-136 data]$ hadoop fs -cat data/twojoin_output3/part-00000 | head -20
82190141        MFGR#121
72972848        MFGR#1210
81279039        MFGR#1211
31722591        MFGR#1212
76146555        MFGR#1213
34864994        MFGR#1214
84914197        MFGR#1215
87636540        MFGR#1216
89845590        MFGR#1217
75218982        MFGR#1218
48925022        MFGR#1219
60639047        MFGR#122
58936780        MFGR#1220
62603992        MFGR#1221
65066902        MFGR#1222
92845179        MFGR#1223
64458379        MFGR#1224
44999675        MFGR#1225
53217128        MFGR#1226
47684737        MFGR#1227
```

3) **Clustering**

A). Using Mahout synthetic clustering

- Now we will generate the sample data using the python notebook as shown below in the figure:



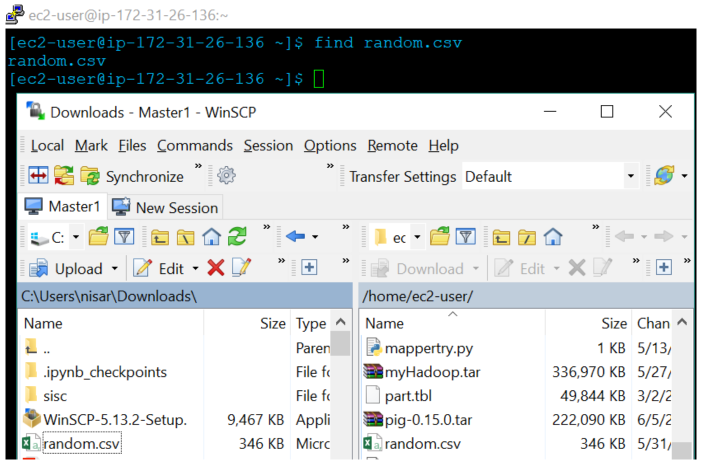
```
In [1]:  import pandas as pd
         import numpy as np

In [3]:  df = pd.DataFrame(np.random.randint(0,100000,size=(10000, 5)), columns=list('ABCDE'))

In [4]:  df.shape
Out[4]:  (10000, 5)

In [5]:  df.head()
Out[5]:
```

	A	B	C	D	E
0	44054	9540	78383	72681	97176
1	56811	76364	96056	28769	38514
2	80313	42332	64670	17996	41677
3	81432	49217	54326	79925	60445
4	20771	87603	94573	73286	21250

```
In [6]:  df.tail()
Out[6]:
```

	A	B	C	D	E
9995	20456	69305	92288	58288	84139
9996	73607	14663	2995	40555	54605
9997	34628	71617	65119	17399	61011
9998	94273	47273	15254	31438	9250
9999	3446	31722	93740	50508	6641

- Now we will generate the output as csv file from python notebook.
    - We will export the csv with ( )space delimiter
      df.to_csv("random.csv", sep=' ')
- Now we will transfer this file from local machine to AWS machine using WinSCP
  Using **find** command we can check if the file is added or not.
  Fig show below:



```
ec2-user@ip-172-31-26-136:~
[ec2-user@ip-172-31-26-136 ~]$ find random.csv
random.csv
[ec2-user@ip-172-31-26-136 ~]$
```

Downloads - Master1 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Transfer Settings Default
Master1
New Session
C:
ec
Upload
Edit
Download
Edit

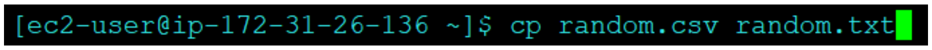
C:\Users\nisar\Downloads\

Name	Size	Type
..		Paren
.ipynb_checkpoints		File fc
sisc		File fc
WinSCP-5.13.2-Setup.	9,467 KB	Appli
random.csv	346 KB	Micrc

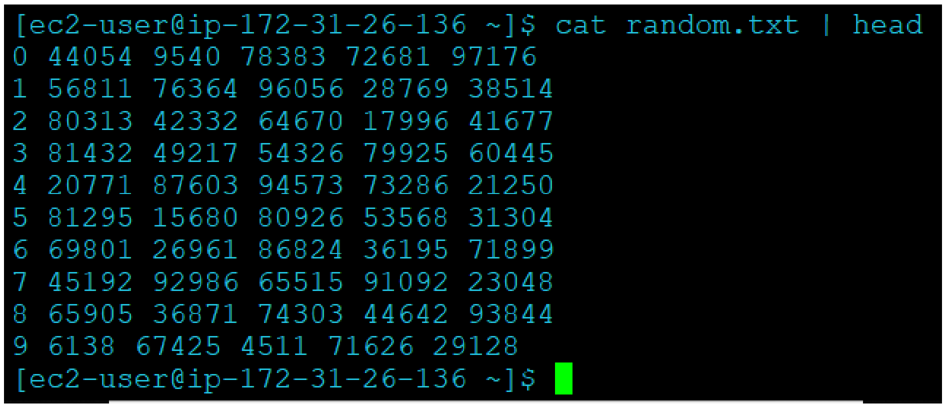
/home/ec2-user/

Name	Size	Chan
mappertry.py	1 KB	5/13/
myHadoop.tar	336,970 KB	5/27/
part.tbl	49,844 KB	3/2/2
pig-0.15.0.tar	222,090 KB	6/5/2
random.csv	346 KB	5/31/

- Copy this file as text file from csv file .

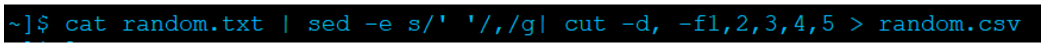


```
[ec2-user@ip-172-31-26-136 ~]$ cp random.csv random.txt
```



```
[ec2-user@ip-172-31-26-136 ~]$ cat random.txt | head
0 44054 9540 78383 72681 97176
1 56811 76364 96056 28769 38514
2 80313 42332 64670 17996 41677
3 81432 49217 54326 79925 60445
4 20771 87603 94573 73286 21250
5 81295 15680 80926 53568 31304
6 69801 26961 86824 36195 71899
7 45192 92986 65515 91092 23048
8 65905 36871 74303 44642 93844
9 6138 67425 4511 71626 29128
[ec2-user@ip-172-31-26-136 ~]$
```

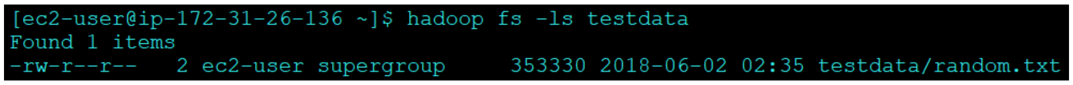
- This is **another approach** you can also use , delimiter while getting file from python to Aws and then use this code to copy it to new file using space( ) delimiter.



```
~]$ cat random.txt | sed -e s/' '/,/g| cut -d, -f1,2,3,4,5 > random.csv
```

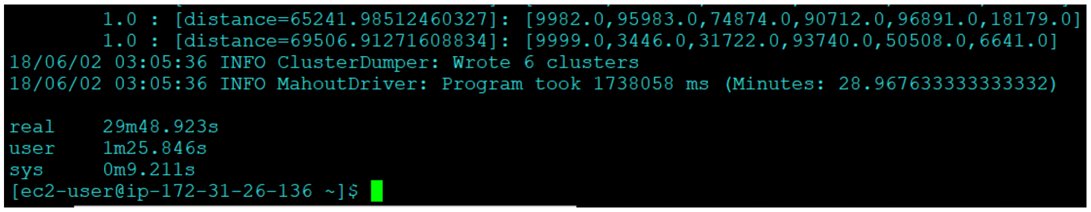
- No we will copy the txt file into the HDFS into testdata folder using command:

  hadoop fs -put random.txt testdata/

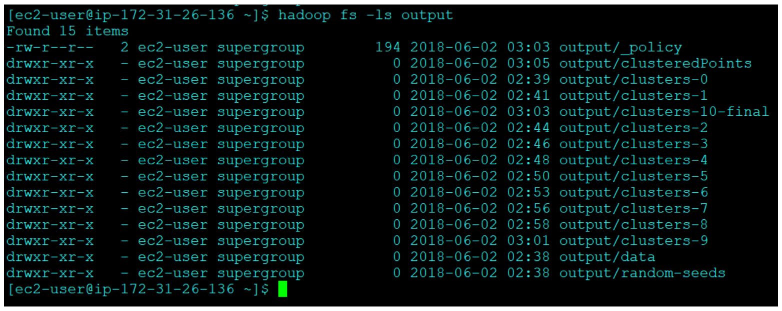


```
[ec2-user@ip-172-31-26-136 ~]$ hadoop fs -ls testdata
Found 1 items
-rw-r--r--   2 ec2-user supergroup     353330 2018-06-02 02:35 testdata/random.txt
```

- **time mahout org.apache.mahout.clustering.syntheticcontrol.kmeans.Job**
  Run this code to do K-means iteration's and then note down the time.

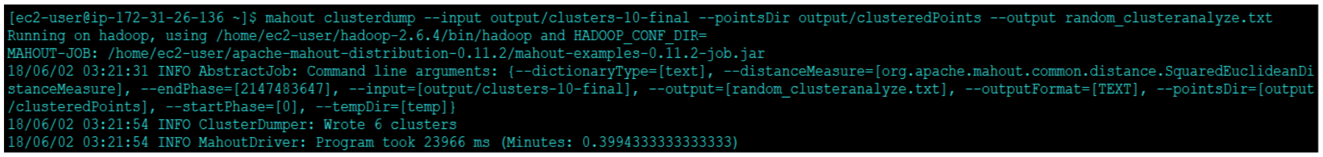


```
        1.0 : [distance=65241.98512460327]: [9982.0,95983.0,74874.0,90712.0,96891.0,18179.0]
        1.0 : [distance=69506.91271608834]: [9999.0,3446.0,31722.0,93740.0,50508.0,6641.0]
18/06/02 03:05:36 INFO ClusterDumper: Wrote 6 clusters
18/06/02 03:05:36 INFO MahoutDriver: Program took 1738058 ms (Minutes: 28.967633333333332)

real    29m48.923s
user    1m25.846s
sys     0m9.211s
[ec2-user@ip-172-31-26-136 ~]$
```

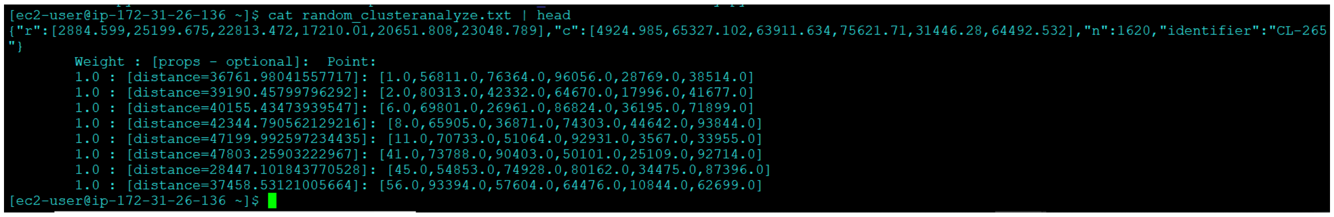


```
[ec2-user@ip-172-31-26-136 ~]$ hadoop fs -ls output
Found 15 items
-rw-r--r--   2 ec2-user supergroup        194 2018-06-02 03:03 output/_policy
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 03:05 output/clusteredPoints
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:39 output/clusters-0
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:41 output/clusters-1
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 03:03 output/clusters-10-final
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:44 output/clusters-2
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:46 output/clusters-3
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:48 output/clusters-4
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:50 output/clusters-5
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:53 output/clusters-6
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:56 output/clusters-7
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:58 output/clusters-8
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 03:01 output/clusters-9
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:38 output/data
drwxr-xr-x   - ec2-user supergroup          0 2018-06-02 02:38 output/random-seeds
[ec2-user@ip-172-31-26-136 ~]$
```

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



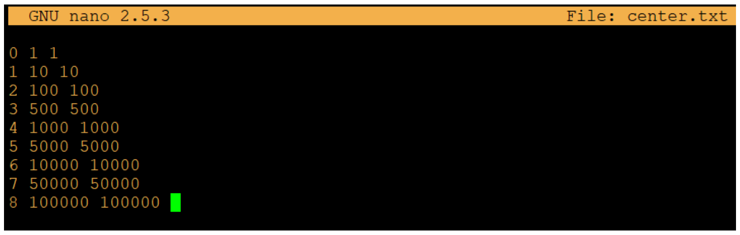
```
[ec2-user@ip-172-31-26-136 ~]$ mahout clusterdump --input output/clusters-10-final --pointsDir output/clusteredPoints --output random_clusteranalyze.txt
Running on hadoop, using /home/ec2-user/hadoop-2.6.4/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/ec2-user/apache-mahout-distribution-0.11.2/mahout-examples-0.11.2-job.jar
18/06/02 03:21:31 INFO AbstractJob: Command line arguments: {--dictionaryType=[text], --distanceMeasure=[org.apache.mahout.common.distance.SquaredEuclideanDi
stanceMeasure], --endPhase=[2147483647], --input=[output/clusters-10-final], --output=[random_clusteranalyze.txt], --outputFormat=[TEXT], --pointsDir=[output
/clusteredPoints], --startPhase=[0], --tempDir=[temp]}
18/06/02 03:21:54 INFO ClusterDumper: Wrote 6 clusters
18/06/02 03:21:54 INFO MahoutDriver: Program took 23966 ms (Minutes: 0.3994333333333333)
```



```
[ec2-user@ip-172-31-26-136 ~]$ cat random_clusteranalyze.txt | head
{"r":[2884.599,25199.675,22813.472,17210.01,20651.808,23048.789],"c":[4924.985,65327.102,63911.634,75621.71,31446.28,64492.532],"n":1620,"identifier":"CL-265
"}
        Weight : [props - optional]:  Point:
        1.0 : [distance=36761.98041557717]: [1.0,56811.0,76364.0,96056.0,28769.0,38514.0]
        1.0 : [distance=39190.45799796292]: [2.0,80313.0,42332.0,64670.0,17996.0,41677.0]
        1.0 : [distance=40155.43473939547]: [6.0,69801.0,26961.0,86824.0,36195.0,71899.0]
        1.0 : [distance=42344.790562129216]: [8.0,65905.0,36871.0,74303.0,44642.0,93844.0]
        1.0 : [distance=47199.992597234435]: [11.0,70733.0,51064.0,92931.0,3567.0,33955.0]
        1.0 : [distance=47803.25903222967]: [41.0,73788.0,90403.0,50101.0,25109.0,92714.0]
        1.0 : [distance=28447.101843770528]: [45.0,54853.0,74928.0,80162.0,34475.0,87396.0]
        1.0 : [distance=37458.53121005664]: [56.0,93394.0,57604.0,64476.0,10844.0,62699.0]
[ec2-user@ip-172-31-26-136 ~]$
```

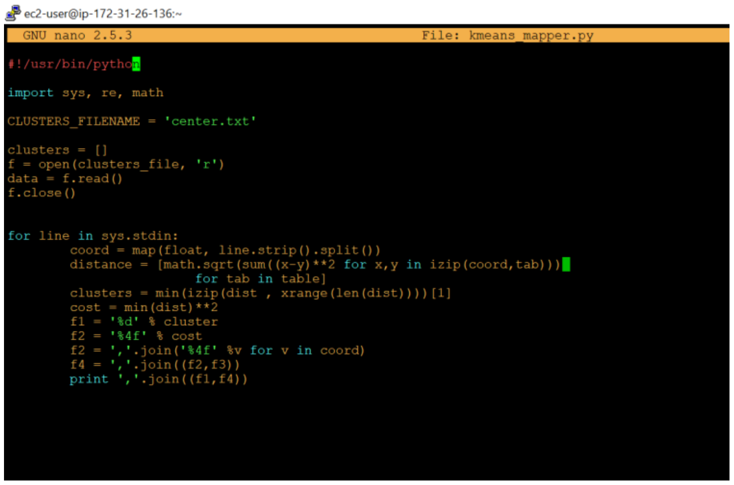
B. Hadoop streaming Map reduce

Creating center.txt file manually with 9 centers



```
  GNU nano 2.5.3                                    File: center.txt

0 1 1
1 10 10
2 100 100
3 500 500
4 1000 1000
5 5000 5000
6 10000 10000
7 50000 50000
8 100000 100000
```

Kmeans_mapper.py



```
ec2-user@ip-172-31-26-136:~
  GNU nano 2.5.3                                    File: kmeans_mapper.py

#!/usr/bin/python

import sys, re, math

CLUSTERS_FILENAME = 'center.txt'

clusters = []
f = open(clusters_file, 'r')
data = f.read()
f.close()


for line in sys.stdin:
        coord = map(float, line.strip().split())
        distance = [math.sqrt(sum((x-y)**2 for x,y in izip(coord,tab)))
                        for tab in table]
        clusters = min(izip(dist , xrange(len(dist))))[1]
        cost = min(dist)**2
        f1 = '%d' % cluster
        f2 = '%4f' % cost
        f2 = ','.join('%4f' %v for v in coord)
        f4 = ','.join((f2,f3))
        print ','.join((f1,f4))
```