# CSC 555 Mining Big Data
# Ronaldlee Ejalu
## CSC 555 Assignment 3

1) Describe how to implement the following queries in MapReduce:

a) **SELECT a.First, a.Last, e.EID, a.AID, e.Age**
**FROM Employee as e, Agent as a**
**WHERE e.Last = a.Last AND e.First = a.First;**

| | Key | Value |
|---|---|---|
| Mapper1 | First_Last | First, Last, AID |
| Mapper2 | First_Last | EID, Age |
| Reducer | First_Last | First + Last + AID + EID + Age |

Mapper 1 will read data from the employees table and for every instance, it checks if the first and last column values are equal to the first last and last name column values of the Agent table, and If there are equal, set the concatenation of First and Last columns as keys and set First, Last and AID as a value which produces the of key-values pairs in the form of
{First_Last '\t'{First, Last, AID}}

Mapper 2 will read from the Agent table and for every instance, it checks if the first and last column values are equal to the first and last column values of the Employee table and if they are, set the concatenation of First and Last columns as keys, and set EID, age as a value which produces key values pairs in the form of {First_Last '\t' {EID, Age}}

Since Mapper 1 and 2 have the same keys so the reducer's key will be set to first_last and the values will be the concatenation of the First, Last , AID, EID and Age and remember that first_last as the key will be sorted before they are added to the reducer.

b) **SELECT SUM(lo_extendedprice)**
**FROM lineorder, dwdate**

```
WHERE lo_orderdate = d_datekey
  AND d_yearmonth = 'Feb1996'
  AND lo_discount = 6;
```

| | Key | value |
|---|---|---|
| Mapper1lo | lo_orderdate | lo_extendedprice |
| Mapper2d | d_datekey | _d |
| Reducer | lo_orderdate,d_datekey | SUM(lo_extendedprice) |

Mapper1lo will read the data from lineorder with a condition that lo_discount is equal to 6 and this condition or filter will have to be met so that key is set to lo_orderdate and set lo_extendedprice as the value so the output of the key-value pairs will be in the form of {lo_orderdate '\t' {lo_extendedprice_lo} }

Mapper2d will read data from dwdate with a condition that d_yearmonth = 'Feb1996' and this condition or filter will have to be met so that the key is set to d_datekey and set _d as the value so the output of the key-value pairs will be in the form of {d_datekey '\t' {_d}}

Then, a partitioner is modified with a custom range function, which is given the mapper's output key and the number of reducers, and returns the index of the intended reducer, which ensures that all the values of the same key are sent to the same reducer.

For all the keys of lo_orderdate, d_datekey, sum up the values of lo_extendedprice and output an aggregate value which will be written back to HDFS.

c)
```
SELECT d_month, COUNT(d_year)
FROM dwdate
GROUP BY d_month
ORDER BY COUNT(d_year)
```

| | key | value |
|---|---|---|
| Mapper1 | d_month | d_year |
| Reducer1 | d_month | COUNT(d_year) |

Mapper1:
For an input block of data, for every date record identified, set the d_month as the key and set d_year as a value.

Reducer1: For each d_month received, compute and output the count of all years by month

| | | |
|---|---|---|
| Mapper2 | Count(d_year) | d_month |
| Reducer2 | Count(d_year) | d_month |

Mapper2: For an input block of data, for each record with month and count of year, set the count of year as key and corresponding month as the value.

Then, a partitioner is modified with a custom range function,  which is given the mapper’s output key , count(d_year)and the number of reducers, and returns the index of the intended reducer, which ensures that all the values of the same key are sent to the same reducer.

The keys and values for each partition are sorted by Hadoop before being presented to the reducer.

Reducer2: For each count of year received, output the d_month values as a list such as
1 ‘\t’ Jan Feb Dec Nov Jul
2 ‘\t’Jun Jan Feb  Oct

2) Consider a Hadoop job that processes an input data file of size equal to 72 disk blocks (72 different blocks, not considering HDFS replication factor). The mapper in this job requires 1 minute to read and fully process a single block of data. Reducer requires 1 second (**not** minute) to produce an answer for one key worth of values and there are a total of 7000 **distinct** keys (mappers generate a lot of key-value pairs, but keys only occur in the 1-7000 range for a total of 7000 unique entries). Assume that each node has a reducer and that the keys are distributed evenly.

a) How long will it take to complete the job if you only had one Hadoop worker node?  For simplicity, assume that that only one mapper and only one reducer are created on every node.
Mapper: 72 blocks
1 min to process one block
⇨ 72 blocks * 1 min = 72 mins
Reducer: 1 key => 1 min
⇨ 7000 keys * 1 sec = 7000 secs
⇨ 7000 secs/60 = 116.67 mins
Therefore: it would take 116.67 + 72 = 188.67 mins

b) 30 Hadoop worker nodes?

Mapper:

| | Node 1 | Node 2 | Node 3 | Node 4 | Node 5 | Node 6 | Node 7 | Node 8 | Node 9 | Node 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | B10 |
| 2 | B31 | B32 | B33 | B34 | B35 | B36 | B37 | B38 | B39 | B40 |
| 3 | B61 | B62 | B63 | B64 | B65 | B66 | B67 | B68 | B69 | B70 |

| Node 11 | Node 12 | Node 13 | Node 14 | Node 15 | Node 16 | Node 17 | Node 18 | Node 19 | Node 10 |
|---|---|---|---|---|---|---|---|---|---|
| B11 | B12 | B13 | B14 | B15 | B16 | B17 | B18 | B19 | B20 |
| B41 | B42 | B43 | B44 | B45 | B46 | B47 | B48 | N49 | B50 |
| B71 | B72 | | | | | | | | |

| Node 1 | Node 2 | Node 3 | Node 4 | Node 5 | Node 6 | Node 7 | Node 8 | Node 9 | Node 10 |
|---|---|---|---|---|---|---|---|---|---|
| B21 | B22 | B23 | B24 | B25 | B26 | B27 | B28 | B29 | B30 |
| B51 | B52 | B53 | B54 | B55 | B56 | B57 | B58 | B59 | B60 |

Mapper takes 3 mins

Reducer:
(7000/30) * 1 sec = (7000/30) secs
Therefore 3*60 + (7000/30) = 413.3 seconds which is equivalent to 6.89 mins (413.3/60)

c) 50 Hadoop worker nodes?

Mapper:
72 blocks take 2mins

Reducer: (7000/50) * 1 second = (7000/50) seconds
Therefore: 2 * 60 + (7000/50) = 260 seconds which is equivalent to 4.3 mins.

d) 100 Hadoop worker nodes?

Mapper:
72 blocks take 1 min
Reducer:
(7000/100) * 1 sec = (7000/100) secs
Therefore: 1 * 60 + (7000/100) = 130 secs which is equivalent to 2.2 mins.

e) Would changing the replication factor have any affect your answers for a-d?

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

Changing the replication factor will have no effect on the answers for a – d.

3)

a) Suppose you have an 8-node cluster with replication factor of 3. Describe what MapReduce has to do after it determines that a node has crashed while a job is being processed. For simplicity, assume that the failed node is not replaced and your cluster is reduced to 7 nodes. Specifically:

If a datanode fails while data is being written to it, then the following actions will be taken and these are always transparent to client node writing the data:

First, the pipeline is closed and any packets in the ack queue are added to the front of the data queue so that datanodes that are downstream from the failed node will not miss any packets.

The current block on the good nodes is given a new identity, which is communicated to the namenode, so that the partial block on the failed datanode will be deleted if the failed datanode recovers later on.

The failed datanode is removed from the pipeline, and a new pipeline is constructed from the seven good datanodes.

The remainder of the block's data is written to the good datanodes (7 nodes) in the pipeline. The namenode notices that the block is under-replicated, and it arranges for a further replica to be created on another node. Subsequent blocks are then treated as normal.

i) What does HDFS (the storage layer) have to do in response to node failure in this case?
Fault tolerance is one of the key features of HDFS so data blocks will be replicated to other nodes implying that HDFS will function normally.

ii) What does MapReduce engine have to do to respond to the node failure? Assume that there was a job in progress at the time of the crash (because otherwise MapReduce does not need to do anything).
If a node manager fails by crashing or running very slowly, it will stop sending heartbeats to the resource manager (Namenode). The resource manager will notice a node manager that has stopped sending heartbeats if it hasn't received one for 10 minutes and remove it from its pool of nodes to schedule containers on. Any tasks running on the failed node manager will be channeled to the good datanodes.

b) Where does the Mapper store output key-value pairs before they are sent to Reducers?
The output of the mapper is stored on the local disk of which ever node it is running from.

c) Can Reducers begin processing before Mapper phase is complete? **Why or why not?**
No, the Reducer will have to wait for a Mapper phase to complete because the Reducer's input are the keys and values which are produced by the Mapper**.**

4) Using the SSBM schema (http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql) load the Part table into Hive (data available at http://cdmgcsarprd01.dpu.depaul.edu/CSC555/SSBM1/part.tbl)

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

Load the data using:
LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl'
OVERWRITE INTO TABLE part;
```

**NOTE**: The provided schema is made for Hive, but by default Hive assumes ‘\t’ separated tables. You will need to modify your CREATE TABLE statement to account for the ‘|’ delimiter in the data.

Use Hive user defined function to perform the following transformation on Part table (creating a new PartSwapped table with the same number of columns): in the 7th column/p_type swap the first and last word in the column and replace the space by a comma. For example, STANDARD BRUSHED TIN would become TIN, BRUSHED STANDARD. For the rest of the columns, where applicable, replace space (‘ ‘) and # characters by an underscore (_), so that MFGR#4 becomes MFGR_4 and MED BAG becomes MED_BAG.

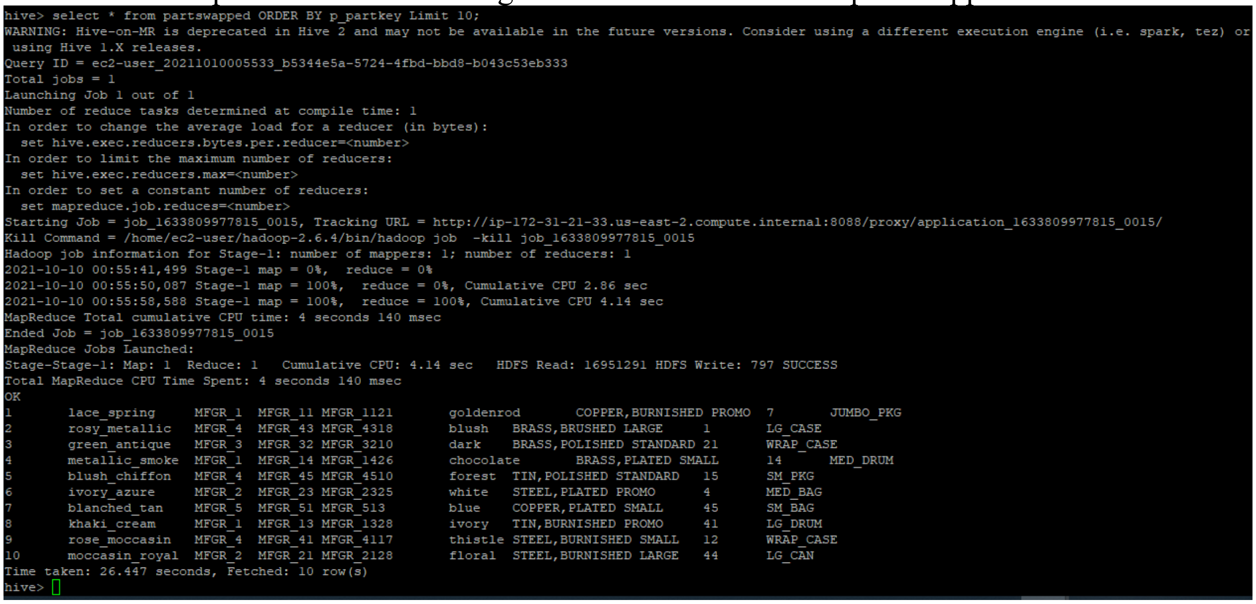
Keep in mind that your transform python code (split/join) should **always** use tab (‘\t’) between fields even if the source data is |-separated. You can also take a look at the transform example included with this assignment for your reference (Examples_Assignment3.doc) which deliberately uses a different delimiter (‘?’).

```
CREATE TABLE partswapped
(
  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 '\t';
add FILE partSwapTransform.py;
INSERT OVERWRITE TABLE partswapped
SELECT TRANSFORM (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_typ
e, p_size, p_container)
USING 'python partSwapTransform.py'
AS (p_partkey, p_name, p_mfgr, p_category, p_brand1, p_color, p_type, p_size, p_c
ontainer) FROM part;
```

partSwapTransform.py

```
#! /usr/bin/python
# Author: Ronaldlee Ejalu
# CSC 555 Mining Big Data
# HomeWork Assignment 3
import sys


def transformPart():
    """
    This function swamps the first and last columns in the 7th column and replace
s the space by the comma
    and the rest of the columns where applicable it, replace space(' ') and # cha
racters by an underscore.
    Hive uses this function to perform the mentioned transformations.
    """
    for lines in sys.stdin:# Loop through the list of strings
        cleansedLine =lines.strip()               # remove any white spaces

        # replace the '|' delimeter with '\t' since Hive assumes that everything
is tab separated
        splittedLinesL = cleansedLine.split('\t')         # split the string to c
reate a list of words, in hadoof, it has to be '\t' delimeted
        splittedPtype = splittedLinesL[6].split(' ')
        ptypeTransformed = splittedPtype[2] +',' + splittedPtype[1] + ' ' + split
tedPtype[0]           # swampping the first and last words
        col0 = splittedLinesL[0]
        col1 = splittedLinesL[1].replace(' ','_')
        col2 = splittedLinesL[2].replace('#','_')
        col3 = splittedLinesL[3].replace('#','_')
        col4 = splittedLinesL[4].replace('#','_')
        col5 = splittedLinesL[5]
        col6 = ptypeTransformed
        col7 = splittedLinesL[7]
        col8 = splittedLinesL[8].replace(' ','_')
        print(col0 + '\t' + col1 + '\t' + col2 + '\t' + col3 + '\t' + col4 + '\t'
 + col5 + '\t' + col6 + '\t' + col7 + '\t' + col8)
transformPart()
```

Below is the output screen after loading the transformed data into partswapped:



```
hive> select * from partswapped ORDER BY p_partkey Limit 10;
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. spark, tez) or
 using Hive 1.X releases.
Query ID = ec2-user_20211010005533_b5344e5a-5724-4fbd-bbd8-b043c53eb333
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1633809977815_0015, Tracking URL = http://ip-172-31-21-33.us-east-2.compute.internal:8088/proxy/application_1633809977815_0015/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1633809977815_0015
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-10-10 00:55:41,499 Stage-1 map = 0%,  reduce = 0%
2021-10-10 00:55:50,087 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.86 sec
2021-10-10 00:55:58,588 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.14 sec
MapReduce Total cumulative CPU time: 4 seconds 140 msec
Ended Job = job_1633809977815_0015
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.14 sec   HDFS Read: 16951291 HDFS Write: 797 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 140 msec
OK
1       lace_spring     MFGR_1  MFGR_11 MFGR_1121       goldenrod       COPPER,BURNISHED PROMO  7       JUMBO_PKG
2       rosy_metallic   MFGR_4  MFGR_43 MFGR_4318       blush   BRASS,BRUSHED LARGE     1       LG_CASE
3       green_antique   MFGR_3  MFGR_32 MFGR_3210       dark    BRASS,POLISHED STANDARD 21      WRAP_CASE
4       metallic_smoke  MFGR_1  MFGR_14 MFGR_1426       chocolate       BRASS,PLATED SMALL      14      MED_DRUM
5       blush_chiffon   MFGR_4  MFGR_45 MFGR_4510       forest  TIN,POLISHED STANDARD   15      SM_PKG
6       ivory_azure     MFGR_2  MFGR_23 MFGR_2325       white   STEEL,PLATED PROMO      4       MED_BAG
7       blanched_tan    MFGR_5  MFGR_51 MFGR_513        blue    COPPER,PLATED SMALL     45      SM_BAG
8       khaki_cream     MFGR_1  MFGR_13 MFGR_1328       ivory   TIN,BURNISHED PROMO     41      LG_DRUM
9       rose_moccasin   MFGR_4  MFGR_41 MFGR_4117       thistle STEEL,BURNISHED SMALL   12      WRAP_CASE
10      moccasin_royal  MFGR_2  MFGR_21 MFGR_2128       floral  STEEL,BURNISHED LARGE   44      LG_CAN
Time taken: 26.447 seconds, Fetched: 10 row(s)
hive> 
```

5) Download and install Pig:

```
cd
wget http://cdmgcsarprd01.dpu.depaul.edu/CSC555/pig-0.15.0.tar.gz
gunzip pig-0.15.0.tar.gz
tar xvf pig-0.15.0.tar
```

set the environment variables (this can also be placed in ~/.bashrc to make it permanent)

```
export PIG_HOME=/home/ec2-user/pig-0.15.0
export PATH=$PATH:$PIG_HOME/bin
```

Use the same vehicles file. Copy the vehicles.csv file to the HDFS if it is not already there.

Now run pig (and use the pig home variable we set earlier):

```
cd $PIG_HOME
bin/pig
```

Create the same table as what we used in Hive, assuming that vehicles.csv is in the home directory on HDFS:

```
VehicleData = LOAD '/user/ec2-user/vehicles.csv' USING PigStorage(',')
AS (barrels08:FLOAT, barrelsA08:FLOAT, charge120:FLOAT, charge240:FLOAT, city08:FLOAT);
```

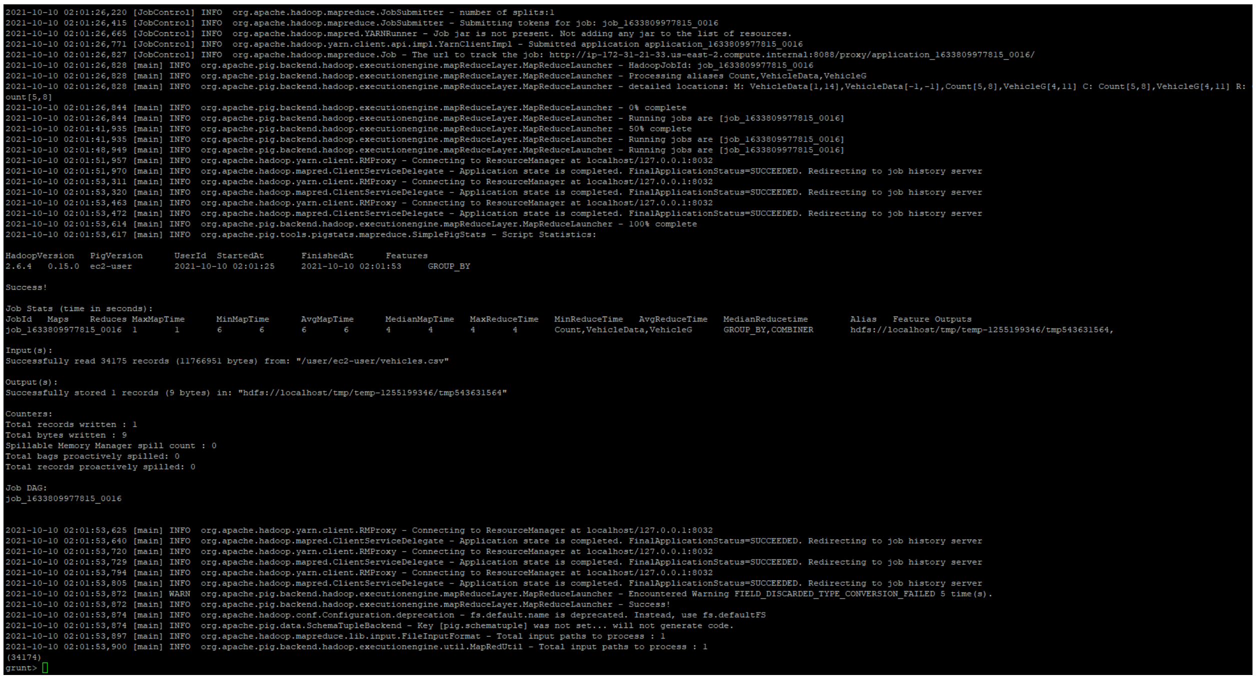
You can see the table description by

```
DESCRIBE VehicleData;
```

Verify that your data has loaded by running:

```
VehicleG = GROUP VehicleData ALL;
```

**Count = FOREACH VehicleG GENERATE COUNT(VehicleData);**
**DUMP Count;**



```
2021-10-10 02:01:26,220 [JobControl] INFO  org.apache.hadoop.mapreduce.JobSubmitter - number of splits:1
2021-10-10 02:01:26,415 [JobControl] INFO  org.apache.hadoop.mapreduce.JobSubmitter - Submitting tokens for job: job_1633809977815_0016
2021-10-10 02:01:26,665 [JobControl] INFO  org.apache.hadoop.mapred.YARNRunner - Job jar is not present. Not adding any jar to the list of resources.
2021-10-10 02:01:26,771 [JobControl] INFO  org.apache.hadoop.yarn.client.api.impl.YarnClientImpl - Submitted application application_1633809977815_0016
2021-10-10 02:01:26,827 [JobControl] INFO  org.apache.hadoop.mapreduce.Job - The url to track the job: http://ip-172-31-21-33.us-east-2.compute.internal:8088/proxy/application_1633809977815_0016/
2021-10-10 02:01:26,828 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - HadoopJobId: job_1633809977815_0016
2021-10-10 02:01:26,828 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Processing aliases Count,VehicleData,VehicleG
2021-10-10 02:01:26,828 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed locations: M: VehicleData[1,14],VehicleData[-1,-1],Count[5,8],VehicleG[4,11] C: Count[5,8],VehicleG[4,11] R:
ount[5,8]
2021-10-10 02:01:26,844 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
2021-10-10 02:01:26,844 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Running jobs are [job_1633809977815_0016]
2021-10-10 02:01:41,935 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 50% complete
2021-10-10 02:01:41,935 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Running jobs are [job_1633809977815_0016]
2021-10-10 02:01:48,949 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Running jobs are [job_1633809977815_0016]
2021-10-10 02:01:51,957 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 02:01:51,970 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-10 02:01:53,311 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 02:01:53,320 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-10 02:01:53,463 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 02:01:53,472 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-10 02:01:53,614 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2021-10-10 02:01:53,617 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.4   0.15.0  ec2-user        2021-10-10 02:01:25     2021-10-10 02:01:53     GROUP_BY

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime        Alias   Feature Outputs
job_1633809977815_0016  1       1       6       6       6       6       4       4       4       4       Count,VehicleData,VehicleG      GROUP_BY,COMBINER       hdfs://localhost/tmp/temp-1255199346/tmp543631564,

Input(s):
Successfully read 34175 records (11766951 bytes) from: "/user/ec2-user/vehicles.csv"

Output(s):
Successfully stored 1 records (9 bytes) in: "hdfs://localhost/tmp/temp-1255199346/tmp543631564"

Counters:
Total records written : 1
Total bytes written : 9
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1633809977815_0016


2021-10-10 02:01:53,625 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 02:01:53,640 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-10 02:01:53,720 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 02:01:53,729 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-10 02:01:53,794 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 02:01:53,805 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2021-10-10 02:01:53,872 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD_DISCARDED_TYPE_CONVERSION_FAILED 5 time(s).
2021-10-10 02:01:53,872 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2021-10-10 02:01:53,874 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2021-10-10 02:01:53,874 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2021-10-10 02:01:53,897 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2021-10-10 02:01:53,900 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(34174)
grunt> 
```

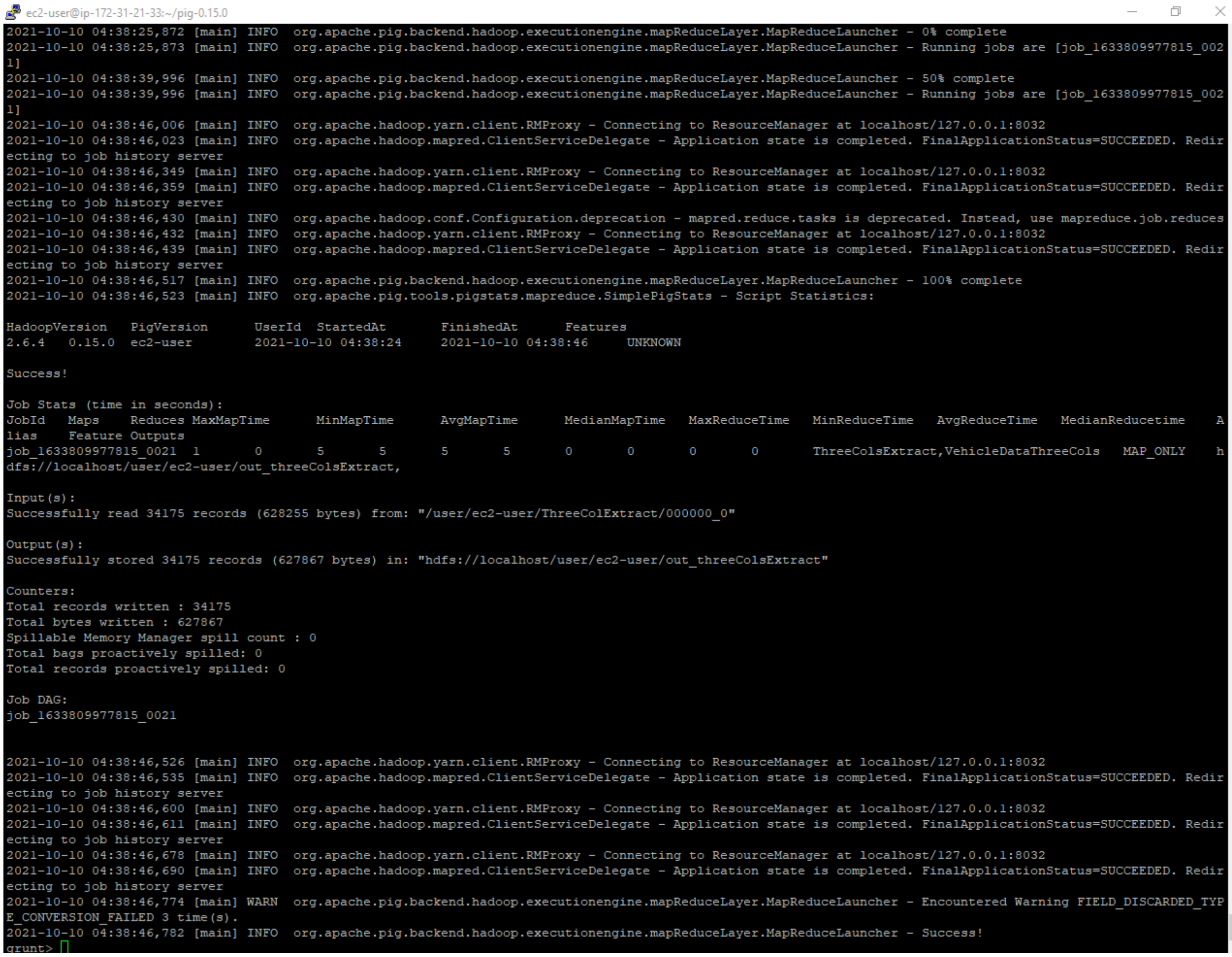
How many rows did you get? (if you get an error here, it is likely because vehicles.csv is not in HDFS)

34,174 records.

Create the same ThreeColExtract file that you have in the previous assignment, by placing barrels08, city08 and charge120 into a new file using PigStorage .You want the STORE command to record output in HDFS. (discussed in p457, Pig Chapter, “Data Processing Operator section)

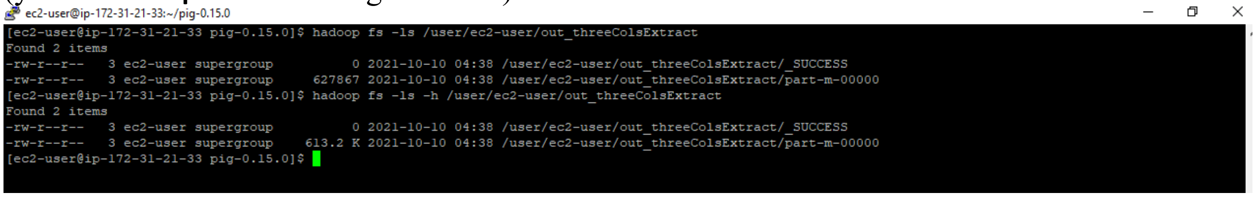
For example, you can use this to get one column (multiple columns are comma-separated)

**OneCol = FOREACH VehicleData GENERATE barrels08;**



```
2021-10-10 04:38:25,872 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
2021-10-10 04:38:25,873 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Running jobs are [job_1633809977815_002
1]
2021-10-10 04:38:39,996 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 50% complete
2021-10-10 04:38:39,996 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Running jobs are [job_1633809977815_002
1]
2021-10-10 04:38:46,006 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 04:38:46,023 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redir
ecting to job history server
2021-10-10 04:38:46,349 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 04:38:46,359 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redir
ecting to job history server
2021-10-10 04:38:46,430 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
2021-10-10 04:38:46,432 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 04:38:46,439 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redir
ecting to job history server
2021-10-10 04:38:46,517 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2021-10-10 04:38:46,523 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.4   0.15.0  ec2-user        2021-10-10 04:38:24     2021-10-10 04:38:46     UNKNOWN

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime    A
lias    Feature Outputs
job_1633809977815_0021  1       0       5       5       5       5       0       0       0       0       ThreeColsExtract,VehicleDataThreeCols   MAP_ONLY    h
dfs://localhost/user/ec2-user/out_threeColsExtract,

Input(s):
Successfully read 34175 records (628255 bytes) from: "/user/ec2-user/ThreeColExtract/000000_0"

Output(s):
Successfully stored 34175 records (627867 bytes) in: "hdfs://localhost/user/ec2-user/out_threeColsExtract"

Counters:
Total records written : 34175
Total bytes written : 627867
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1633809977815_0021


2021-10-10 04:38:46,526 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 04:38:46,535 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redir
ecting to job history server
2021-10-10 04:38:46,600 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 04:38:46,611 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redir
ecting to job history server
2021-10-10 04:38:46,678 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at localhost/127.0.0.1:8032
2021-10-10 04:38:46,690 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redir
ecting to job history server
2021-10-10 04:38:46,774 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD_DISCARDED_TYP
E_CONVERSION_FAILED 3 time(s).
2021-10-10 04:38:46,782 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
grunt>
```

Verify that the new file has been created and report the size of the newly created file. (you can use **quit** to exit the grunt shell)



```
[ec2-user@ip-172-31-21-33 pig-0.15.0]$ hadoop fs -ls /user/ec2-user/out_threeColsExtract
Found 2 items
-rw-r--r--   3 ec2-user supergroup          0 2021-10-10 04:38 /user/ec2-user/out_threeColsExtract/_SUCCESS
-rw-r--r--   3 ec2-user supergroup     627867 2021-10-10 04:38 /user/ec2-user/out_threeColsExtract/part-m-00000
[ec2-user@ip-172-31-21-33 pig-0.15.0]$ hadoop fs -ls -h /user/ec2-user/out_threeColsExtract
Found 2 items
-rw-r--r--   3 ec2-user supergroup          0 2021-10-10 04:38 /user/ec2-user/out_threeColsExtract/_SUCCESS
-rw-r--r--   3 ec2-user supergroup    613.2 K 2021-10-10 04:38 /user/ec2-user/out_threeColsExtract/part-m-00000
[ec2-user@ip-172-31-21-33 pig-0.15.0]$
```

The size of the file is 613.2 K.

Submit a single document containing your written answers. Be sure that this document contains your name and "CSC 555 Assignment 3" at the top.