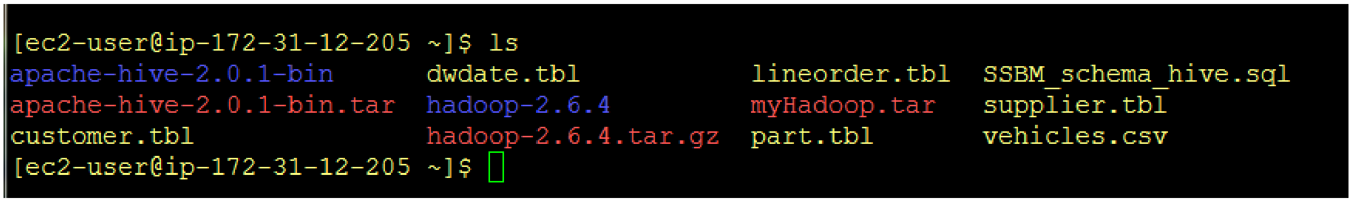
Michal Chowaniak
CSC 555: Mining Big Data
Project, Phase 1 (due Sunday, May 19th)

In this part of the project (which will also serve as our take-home midterm), you will 1) Set up a 4-node cluster and 2) perform data warehousing and transformation queries using Hive, Pig and Hadoop streaming. The modified Hive-style schema is at:
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql
It is based on SSBM benchmark (derived from industry standard TPCH benchmark). The data is at Scale1, or the smallest unit – lineorder is the largest table at about 0.6GB. You can use wget to download the following links. Keep in mind that data is |-separated (not CSV).
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/dwdate.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/supplier.tbl
http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl
Please be sure to submit all code (pig, python and SQL).



```
[ec2-user@ip-172-31-12-205 ~]$ ls
apache-hive-2.0.1-bin      dwdate.tbl           lineorder.tbl  SSBM_schema_hive.sql
apache-hive-2.0.1-bin.tar  hadoop-2.6.4         myHadoop.tar   supplier.tbl
customer.tbl               hadoop-2.6.4.tar.gz  part.tbl       vehicles.csv
[ec2-user@ip-172-31-12-205 ~]$
```

Part 1: Multi-node cluster

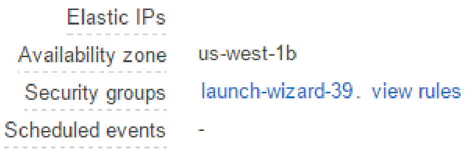
1) Your first step is to setup a multi-node cluster and re-run a simple wordcount. For this part, you will create a 3-node cluster (with a total of 1 master + 2 worker nodes). Include your master node in the “slaves” file, to make sure **all 3** nodes are working.
You need to perform the following steps:

    1. Create a new node of a medium size (you can always switch the size of the node). It is possible, but I do not recommend trying to reconfigure your existing Hadoop into this new cluster (it is much easier to make 3 new nodes for a total of 4 in your AWS account).

        a. **When creating a node I recommend changing the default 8G hard drive to 30G on all nodes.**

        b. Change your security group setting to open firewall access. We need to open the ports in two different ways. We will open port 50070 for the web interface in order to be able to see the cluster status in a browser. We will also set 0-64000 range opening up all ports. However, we will ensure that the ports are open only **within** the cluster and not to the world.

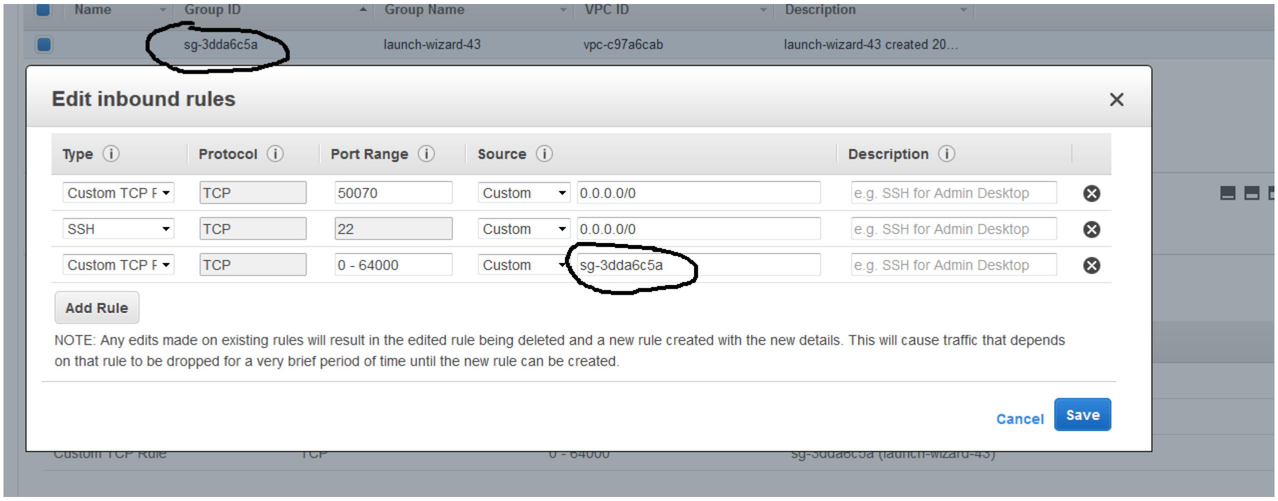
In order to make changes, you need to do the following. Access the cluster security group (launch-wizard-xx).



Elastic IPs	
Availability zone	us-west-1b
Security groups	launch-wizard-39 . view rules
Scheduled events	-

Right click on the security group and choose Edit inbound rules

Note that the first line below is opening port 50070. The second line below is the default (port 22 is required for regular SSH connections). The third line opens all ports but ONLY for the same security group (assuming that all of your nodes in the cluster share the same security group – that will happen automatically if you use the "create more like this" option when creating instances as specified in part 1-c below). We previously had some issues with machines being hacked without that last limitation, so make sure you include it.



Name	Group ID	Group Name	VPC ID	Description
	sg-3dda6c5a	launch-wizard-43	vpc-c97a6cab	launch-wizard-43 created 20...

Edit inbound rules

Type	Protocol	Port Range	Source		Description
Custom TCP F	TCP	50070	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F	TCP	0 - 64000	Custom	sg-3dda6c5a	e.g. SSH for Admin Desktop

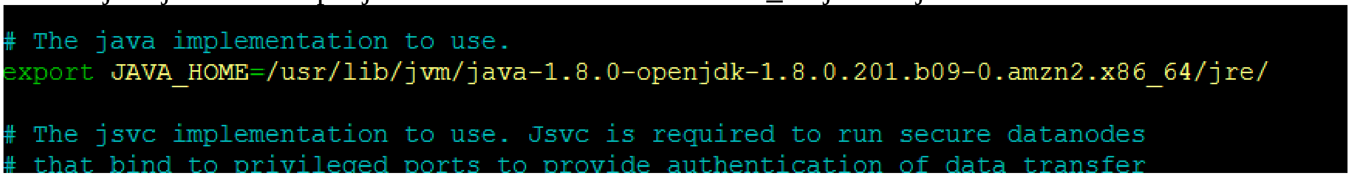
NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

c. Right click on the Master node and choose "create more like this" to create 2 more nodes with same settings. If you configure the network settings on master first, security group information will be copied.
NOTE: Hard drive size will not be copied and default to 8G unless you change it.

2. Connect to the master and set up Hadoop similarly to what you did previously. Do not attempt to repeat these steps on workers yet – you will only need to set up Hadoop once.

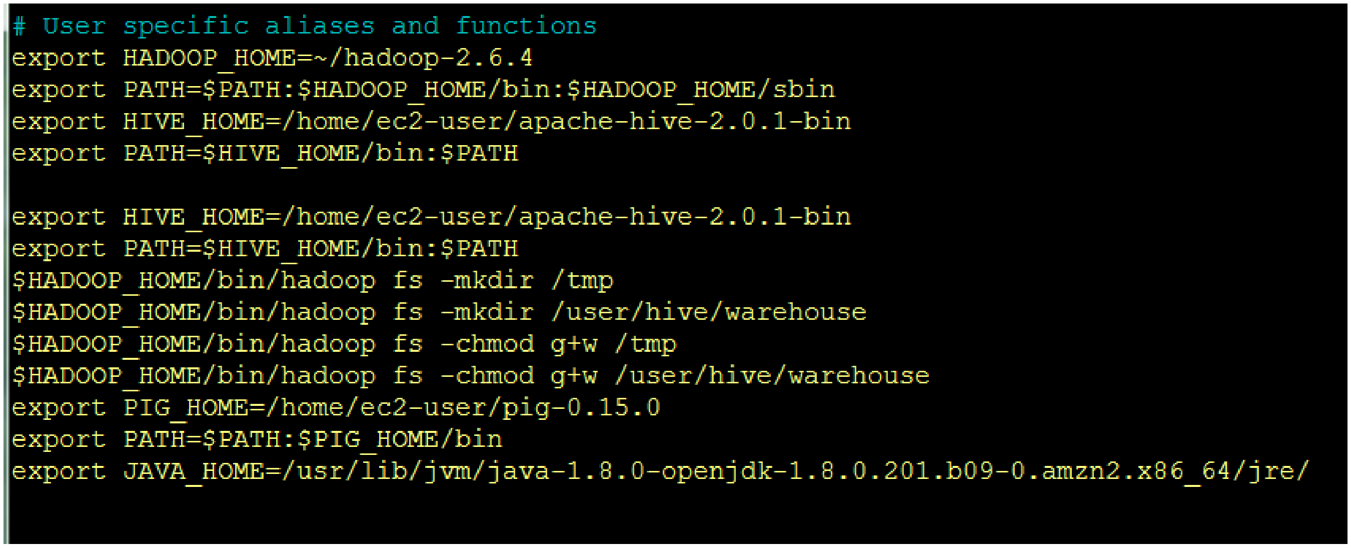
nano hadoop-2.6.4/etc/hadoop/hadoop-env.sh

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/bin/java



```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
```

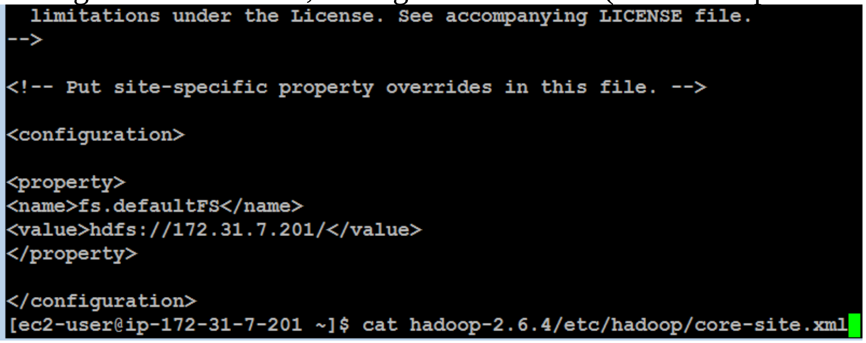
nano ~/.bashrc



```
# User specific aliases and functions
export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HIVE_HOME=/home/ec2-user/apache-hive-2.0.1-bin
export PATH=$HIVE_HOME/bin:$PATH

export HIVE_HOME=/home/ec2-user/apache-hive-2.0.1-bin
export PATH=$HIVE_HOME/bin:$PATH
$HADOOP_HOME/bin/hadoop fs -mkdir /tmp
$HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
$HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
$HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse
export PIG_HOME=/home/ec2-user/pig-0.15.0
export PATH=$PATH:$PIG_HOME/bin
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.amzn2.x86_64/jre/
```

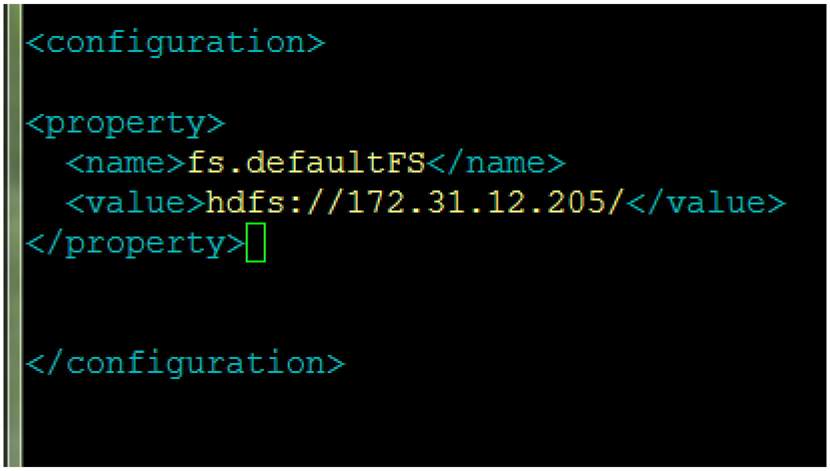
source ~/.bashrc

a. Configure core-site.xml, adding the **PrivateIP** (do not use public IP) of the master.



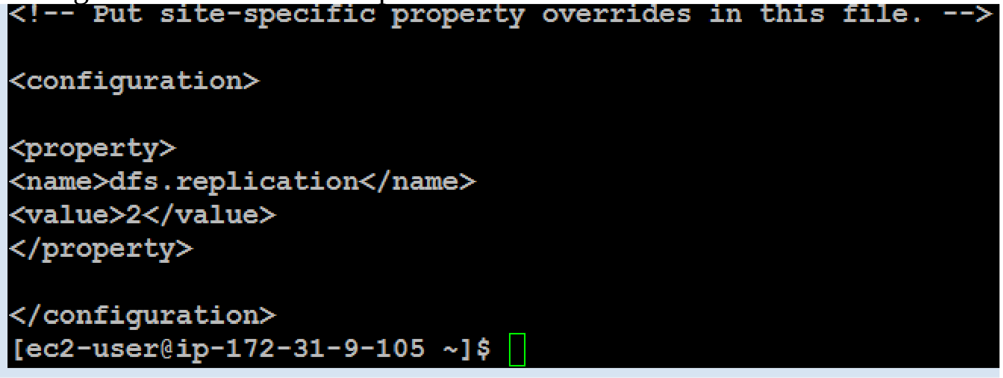
```
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>fs.defaultFS</name>
<value>hdfs://172.31.7.201/</value>
</property>

</configuration>
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/core-site.xml
```

nano hadoop-2.6.4/etc/hadoop/core-site.xml



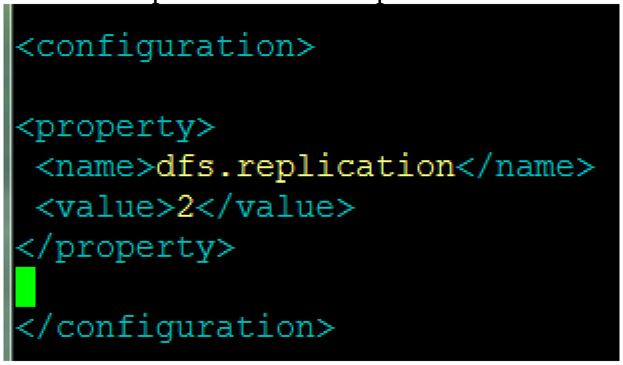
```
<configuration>

<property>
  <name>fs.defaultFS</name>
  <value>hdfs://172.31.12.205/</value>
</property>


</configuration>
```

b. Configure hdfs-site and set replication factor to 2.



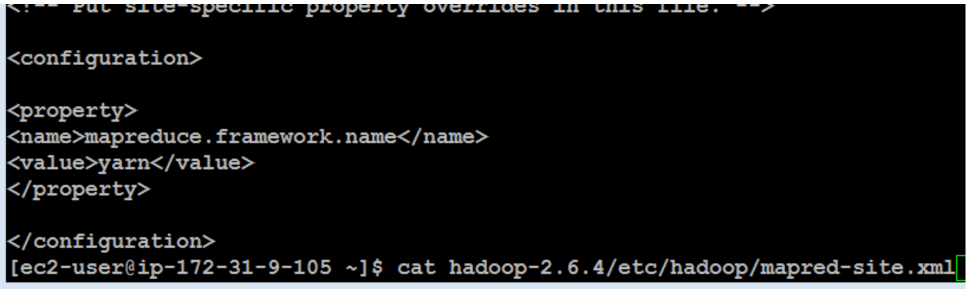
```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>dfs.replication</name>
<value>2</value>
</property>

</configuration>
[ec2-user@ip-172-31-9-105 ~]$
```

nano hadoop-2.6.4/etc/hadoop/hdfs-site.xml



```
<configuration>

<property>
 <name>dfs.replication</name>
 <value>2</value>
</property>

</configuration>
```

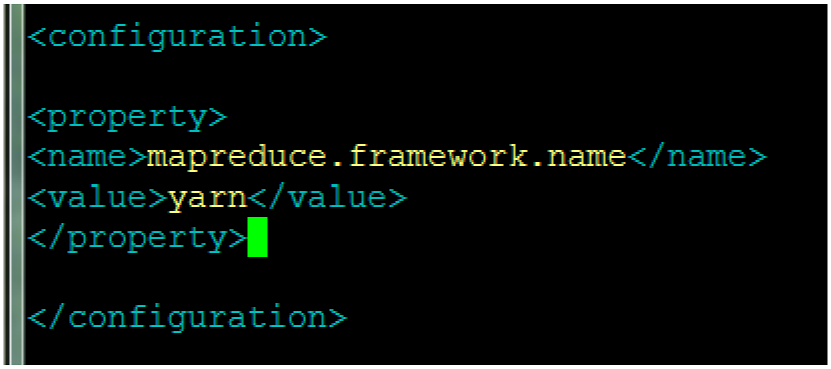
c. cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml and then configure mapred-site.xml



```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>

</configuration>
[ec2-user@ip-172-31-9-105 ~]$ cat hadoop-2.6.4/etc/hadoop/mapred-site.xml
```

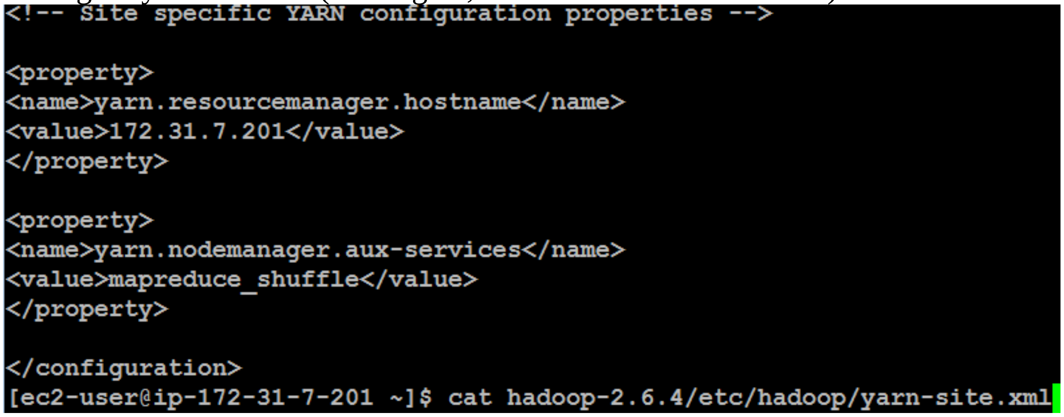
cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml

nano hadoop-2.6.4/etc/hadoop/mapred-site.xml



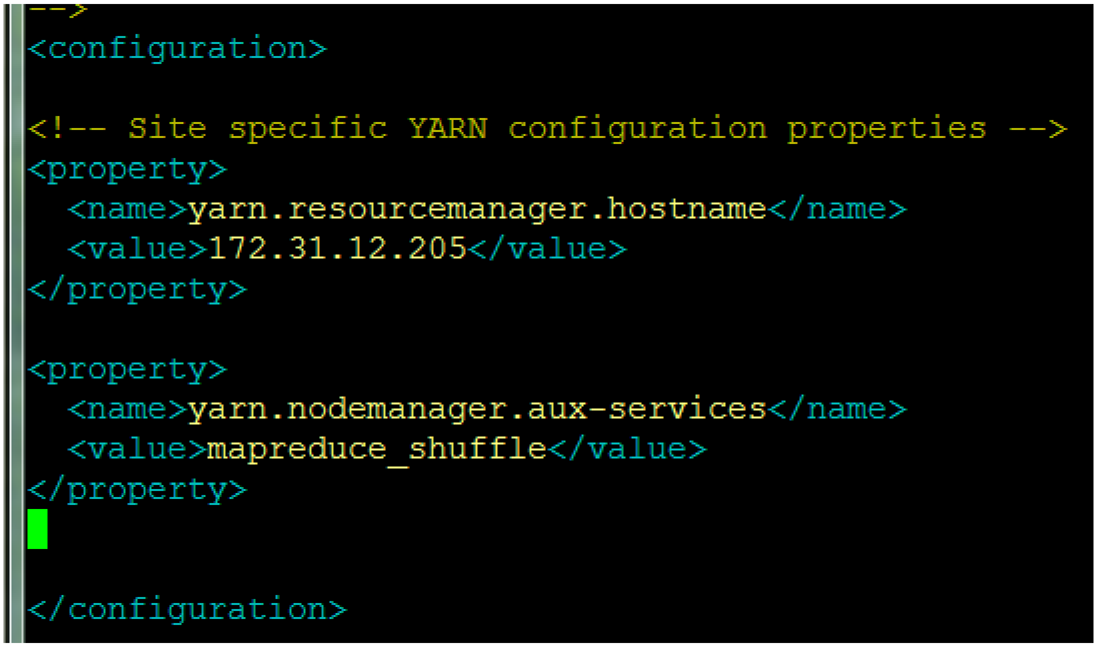
```
<configuration>

<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>

</configuration>
```

d. Configure yarn-site.xml (once again, use PrivateIP of the master)



```
<!-- Site specific YARN configuration properties -->

<property>
<name>yarn.resourcemanager.hostname</name>
<value>172.31.7.201</value>
</property>

<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

</configuration>
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/yarn-site.xml
```

nano hadoop-2.6.4/etc/hadoop/yarn-site.xml



```
-->
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>172.31.12.205</value>
</property>

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>


</configuration>
```

Finally, edit the slaves file and list your 4 nodes (master and 3 workers) using Private IPs

```
[ec2-user@ip-172-31-7-201 ~]$ cat hadoop-2.6.4/etc/hadoop/slaves
172.31.7.201
172.31.5.246
…
```

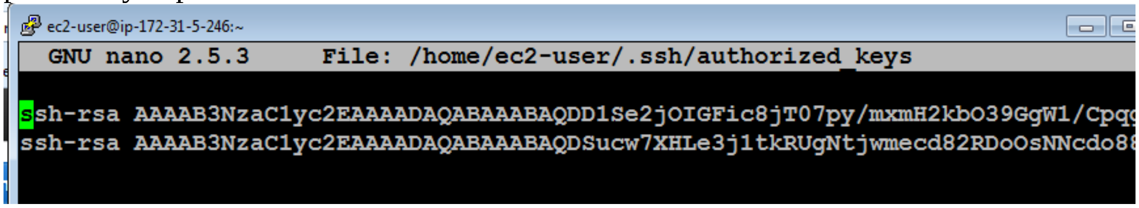
```
nano hadoop-2.6.4/etc/hadoop/slaves
```

```
more .ssh//authorized_keys
```

Make sure that you use private IP (private DNS is also ok) for your configuration files (such as conf/masters and conf/slaves or the other 3 config files). The advantage of the Private IP is that it does not change after your instance is stopped (if you use the Public IP, the cluster would need to be reconfigured every time it is stopped). The downside of the Private IP is that it is only meaningful within the Amazon EC2 network. So all nodes in EC2 can talk to each other using Private IP, but you cannot connect to your instance from the outside (e.g., from your laptop) because Private IP has no meaning for your laptop (since your laptop is not part of the Amazon EC2 network).

Now, we will pack up and move Hadoop to the workers. All you need to do is to generate and then copy the public key to the worker nodes to achieve passwordless access across your cluster.

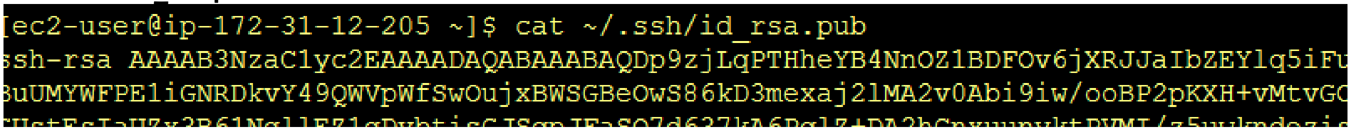
1. Run ssh-keygen -t rsa (and enter empty values for the passphrase) on the master node. That will generate .ssh/id_rsa and .ssh/id_rsa.pub (private and public key). You now need to manually copy the .ssh/id_rsa.pub and append it to ~/.ssh/authorized_keys **on each worker.**
   Keep in mind that this is a single-line public key and accidentally introducing a line break would cause a mismatch.
   Note that the example below is NOT the master, but one of the workers (ip-172-31-5-246). The first public key is the .pem Amazon half and the 2nd public key is the master's public key copied in as one line.



```
ec2-user@ip-172-31-5-246:~
  GNU nano 2.5.3      File: /home/ec2-user/.ssh/authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDD1Se2jOIGFic8jT07py/mxmH2kbO39GgW1/Cpqg
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDSucw7XHLe3j1tkRUgNtjwmecd82RDoOsNNcdo88
```

on Master

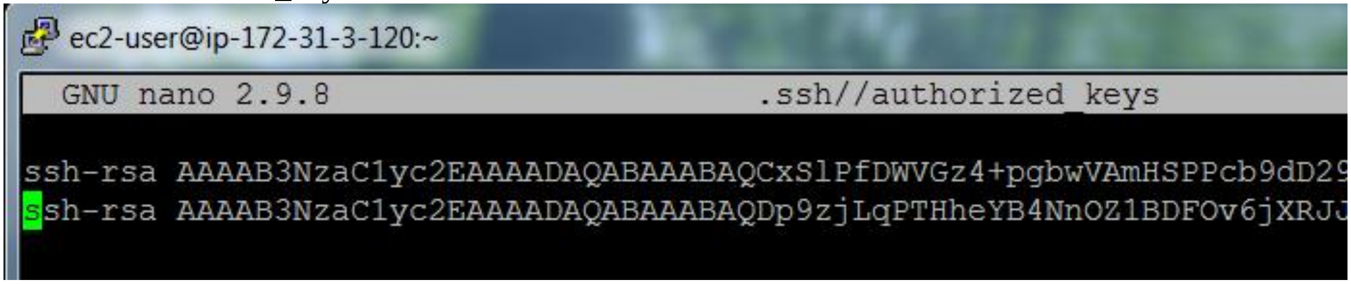
cat ~/.ssh/id_rsa.pub



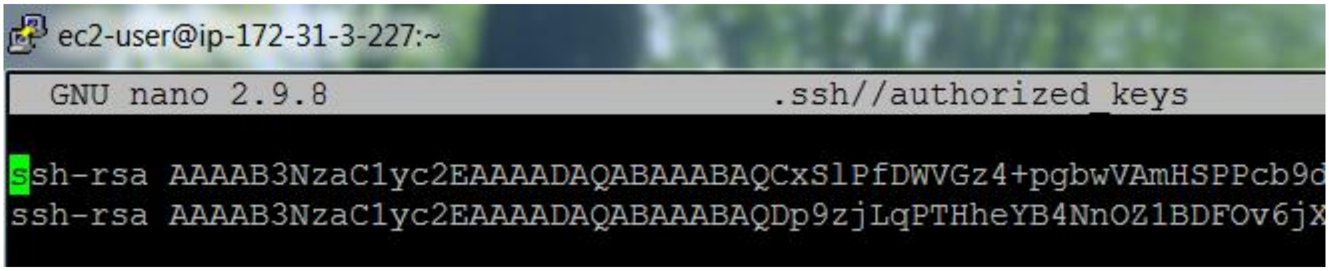
```
[ec2-user@ip-172-31-12-205 ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDp9zjLqPTHheYB4NnOZ1BDFOv6jXRJJaIbZEYlq5iFu
BuUMYWFPE1iGNRDkvY49QWVpWfSwOujxBWSGBeOwS86kD3mexaj21MA2v0Abi9iw/ooBP2pKXH+vMtvGC
```

On each worker

nano .ssh//authorized_keys



```
ec2-user@ip-172-31-3-120:~
  GNU nano 2.9.8                         .ssh//authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxSlPfDWVGz4+pgbwVAmHSPPcb9dD29
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDp9zjLqPTHheYB4NnOZ1BDFOv6jXRJJ
```

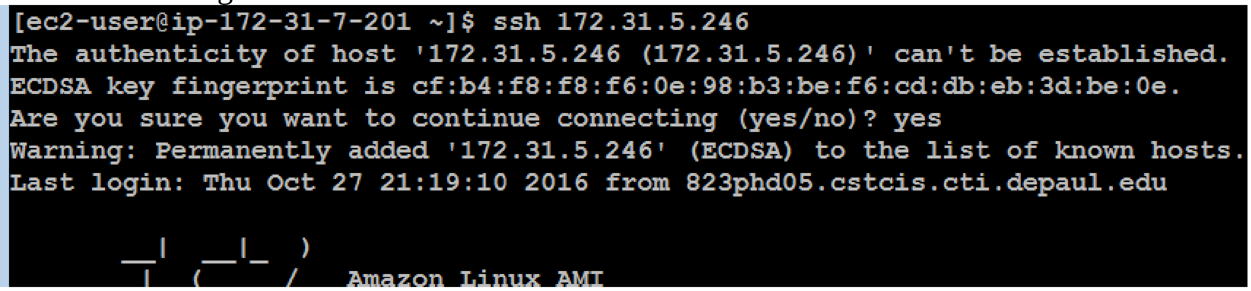


```
ec2-user@ip-172-31-3-227:~
  GNU nano 2.9.8                          .ssh//authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxSlPfDWVGz4+pgbwVAmHSPPcb9d
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDp9zjLqPTHheYB4NnOZ1BDFOv6jX
```

You can add the public key of the master to the master by running this command:

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

Make sure that you can ssh to all of the nodes from the master node (by running ssh 54.186.221.92, where the IP address is your worker node) from the master and ensuring that you were able to login. You can exit after successful ssh connection by typing exit (the command prompt will tell you which machine you are connected to, e.g., ec2-user@ip-172-31-37-113). Here's me ssh-ing from master to worker.



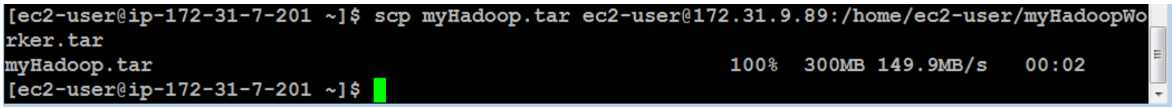
```
[ec2-user@ip-172-31-7-201 ~]$ ssh 172.31.5.246
The authenticity of host '172.31.5.246 (172.31.5.246)' can't be established.
ECDSA key fingerprint is cf:b4:f8:f8:f6:0e:98:b3:be:f6:cd:db:eb:3d:be:0e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.5.246' (ECDSA) to the list of known hosts.
Last login: Thu Oct 27 21:19:10 2016 from 823phd05.cstcis.cti.depaul.edu

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
```

Once you have verified that you can ssh from the master node to every cluster member including the master itself (ssh localhost), you are going to return to the master node (**exit** until your prompt shows the IP address of the master node) and pack the contents of the hadoop directory there. Make sure your Hadoop installation is configured correctly (because from now on, you will have 4 copies of the Hadoop directory and all changes need to be applied in 4 places).

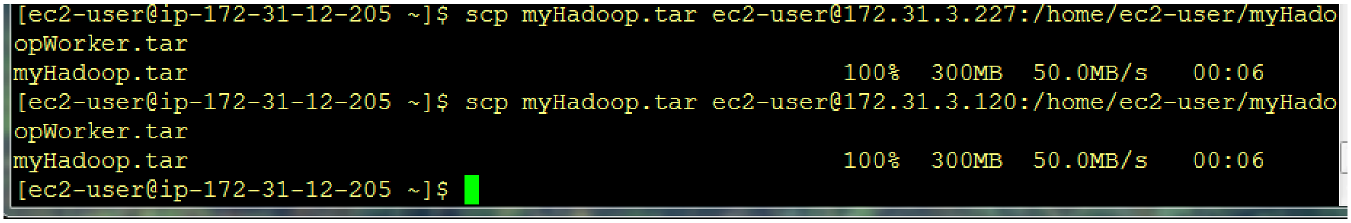
cd (go to root home directory, i.e. /home/ec2-user/)
(pack up the entire Hadoop directory into a single file for transfer. You can optionally compress the file with gzip)
tar cvf myHadoop.tar hadoop-2.6.4
ls -al myHadoop.tar (to verify that the .tar file had been created)

Now, you need to copy the myHadoop.tar file to every non-master node in the cluster. If you had successfully setup public-private key access in the previous step, this command (for each worker node) will do that:

(copies the myHadoop.tar file from the current node to a remote node into a file called myHadoopWorker.tar. Don't forget to replace the IP address with that your worker nodes. By the way, since you are on the Amazon EC2 network, either Public or Private IP will work just fine.)
scp myHadoop.tar ec2-user@172.31.3.227:/home/ec2-user/myHadoopWorker.tar
scp myHadoop.tar ec2-user@172.31.3.120:/home/ec2-user/myHadoopWorker.tar



```
[ec2-user@ip-172-31-7-201 ~]$ scp myHadoop.tar ec2-user@172.31.9.89:/home/ec2-user/myHadoopWo
rker.tar
myHadoop.tar                                         100%  300MB 149.9MB/s   00:02
[ec2-user@ip-172-31-7-201 ~]$
```



```
[ec2-user@ip-172-31-12-205 ~]$ scp myHadoop.tar ec2-user@172.31.3.227:/home/ec2-user/myHado
opWorker.tar
myHadoop.tar                                          100%  300MB  50.0MB/s   00:06
[ec2-user@ip-172-31-12-205 ~]$ scp myHadoop.tar ec2-user@172.31.3.120:/home/ec2-user/myHado
opWorker.tar
myHadoop.tar                                          100%  300MB  50.0MB/s   00:06
[ec2-user@ip-172-31-12-205 ~]$
```

Once the tar file containing your Hadoop installation from master node has been copied to each worker node, you need to login to each non-master node and unpack the .tar file.

Run the following command (on each worker node, not on the master) to untar the hadoop file. We are purposely using a different tar archive name (i.e., **myHadoopWorker.tar**), so if you get "file not found" error, that means you are running this command on the master node or have not yet successfully copied myHadoopWorker.tar file to the worker.

```
tar xvf myHadoopWorker.tar
```

Once you are done, run this on the master (nothing needs to be done on the workers to format the cluster unless you are re-formatting, in which case you'll need to delete the dfs directory).

```
hadoop namenode -format
```

bin/hadoop namenode -format

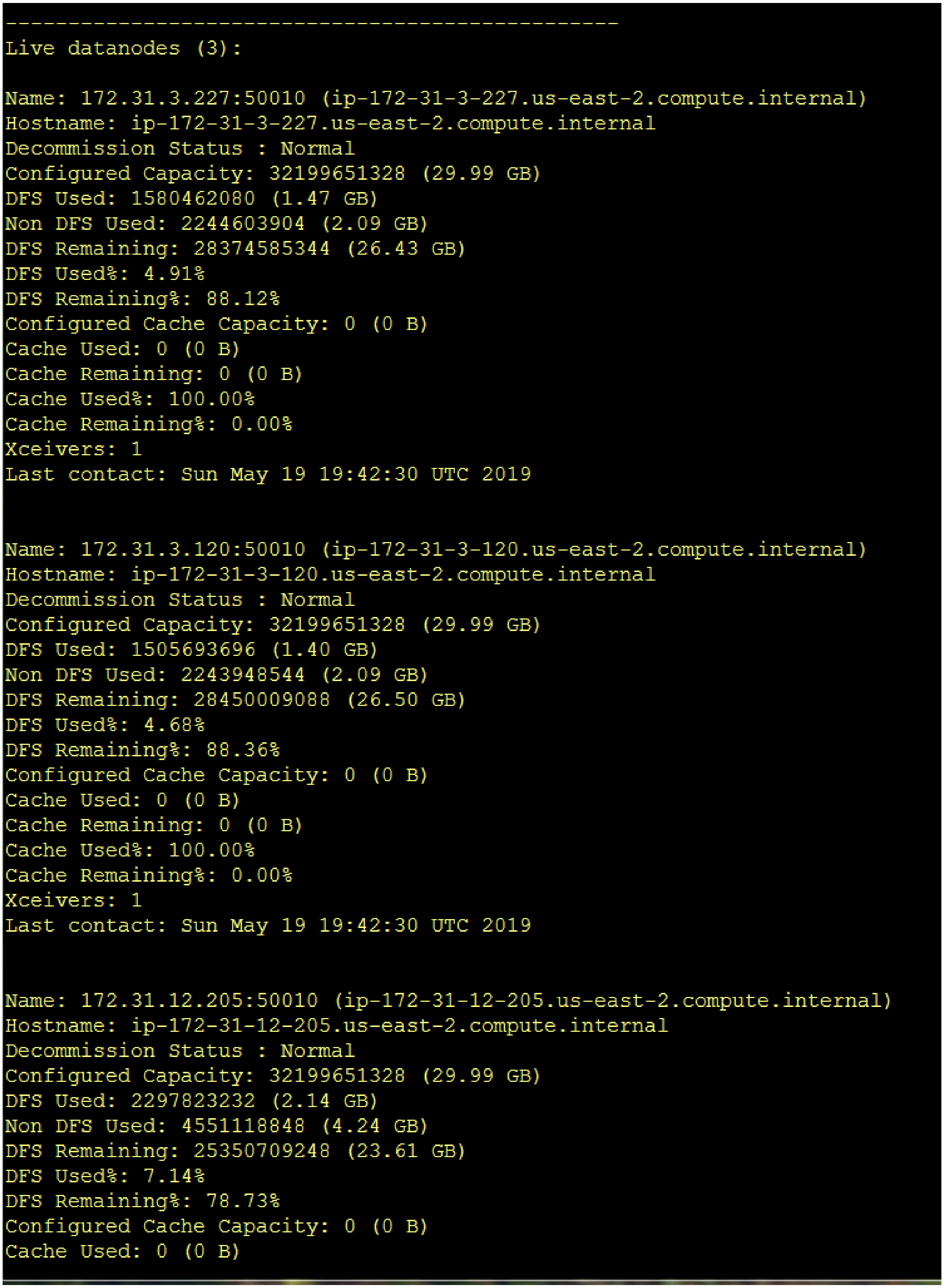
Once you have successfully completed the previous steps, you should can start and use your new cluster by going to the master node and running the start-dfs.sh and start-yarn.sh scripts (you do not need to explicitly start anything on worker nodes – the master will do that for you).

You should verify that the cluster is running by pointing your browser to the link below.

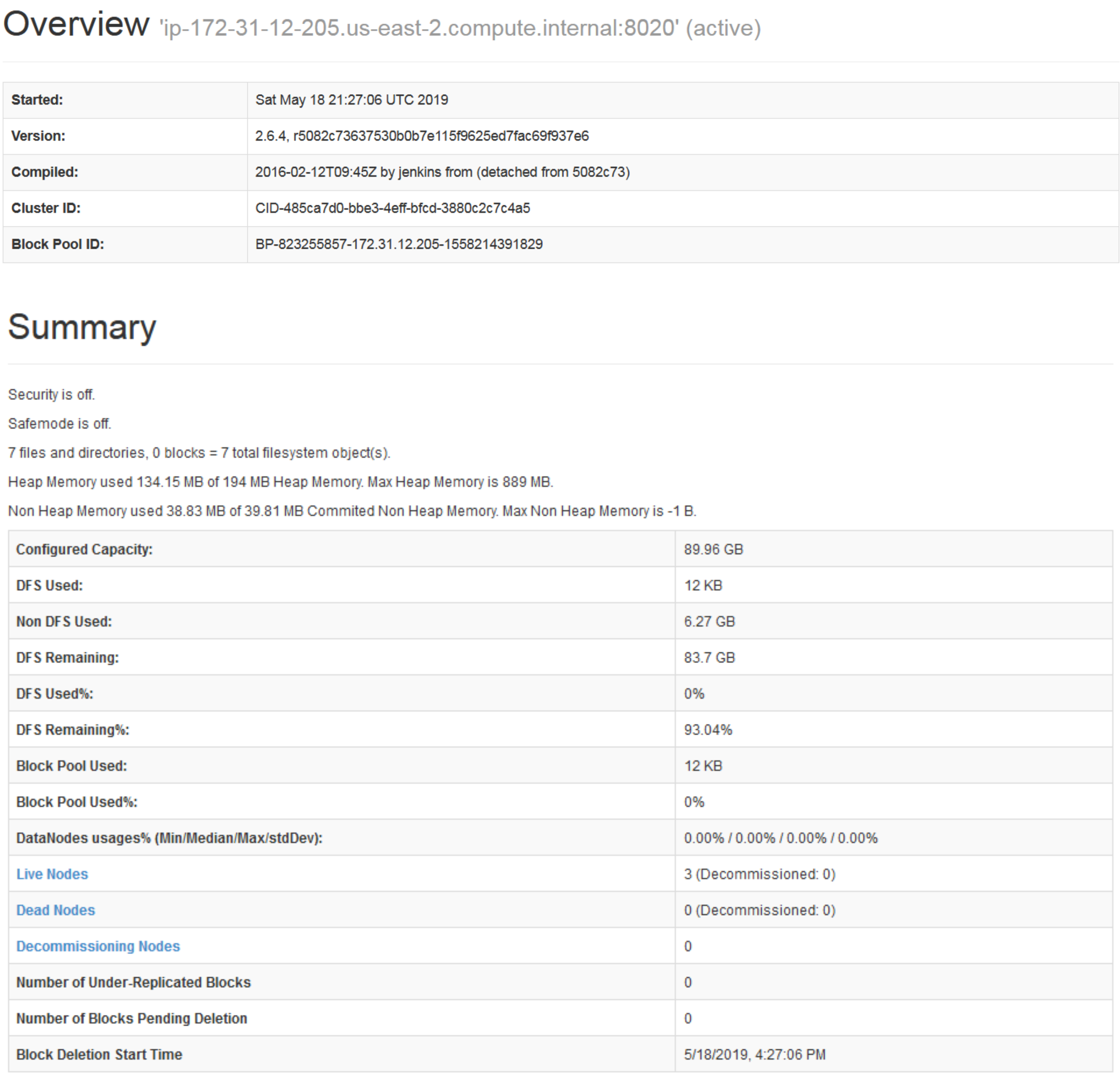
http://18.224.61.240:50070/

Make sure that the cluster is operational (you can see the 4 nodes under Datanodes tab).

Submit a screenshot of your cluster status view.



```
-------------------------------------------------
Live datanodes (3):

Name: 172.31.3.227:50010 (ip-172-31-3-227.us-east-2.compute.internal)
Hostname: ip-172-31-3-227.us-east-2.compute.internal
Decommission Status : Normal
Configured Capacity: 32199651328 (29.99 GB)
DFS Used: 1580462080 (1.47 GB)
Non DFS Used: 2244603904 (2.09 GB)
DFS Remaining: 28374585344 (26.43 GB)
DFS Used%: 4.91%
DFS Remaining%: 88.12%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sun May 19 19:42:30 UTC 2019


Name: 172.31.3.120:50010 (ip-172-31-3-120.us-east-2.compute.internal)
Hostname: ip-172-31-3-120.us-east-2.compute.internal
Decommission Status : Normal
Configured Capacity: 32199651328 (29.99 GB)
DFS Used: 1505693696 (1.40 GB)
Non DFS Used: 2243948544 (2.09 GB)
DFS Remaining: 28450009088 (26.50 GB)
DFS Used%: 4.68%
DFS Remaining%: 88.36%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sun May 19 19:42:30 UTC 2019


Name: 172.31.12.205:50010 (ip-172-31-12-205.us-east-2.compute.internal)
Hostname: ip-172-31-12-205.us-east-2.compute.internal
Decommission Status : Normal
Configured Capacity: 32199651328 (29.99 GB)
DFS Used: 2297823232 (2.14 GB)
Non DFS Used: 4551118848 (4.24 GB)
DFS Remaining: 25350709248 (23.61 GB)
DFS Used%: 7.14%
DFS Remaining%: 78.73%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
```



Overview 'ip-172-31-12-205.us-east-2.compute.internal:8020' (active)

Started:	Sat May 18 21:27:06 UTC 2019
Version:	2.6.4, r5082c73637530b0b7e115f9625ed7fac69f937e6
Compiled:	2016-02-12T09:45Z by jenkins from (detached from 5082c73)
Cluster ID:	CID-485ca7d0-bbe3-4eff-bfcd-3880c2c7c4a5
Block Pool ID:	BP-823255857-172.31.12.205-1558214391829

Summary

Security is off.

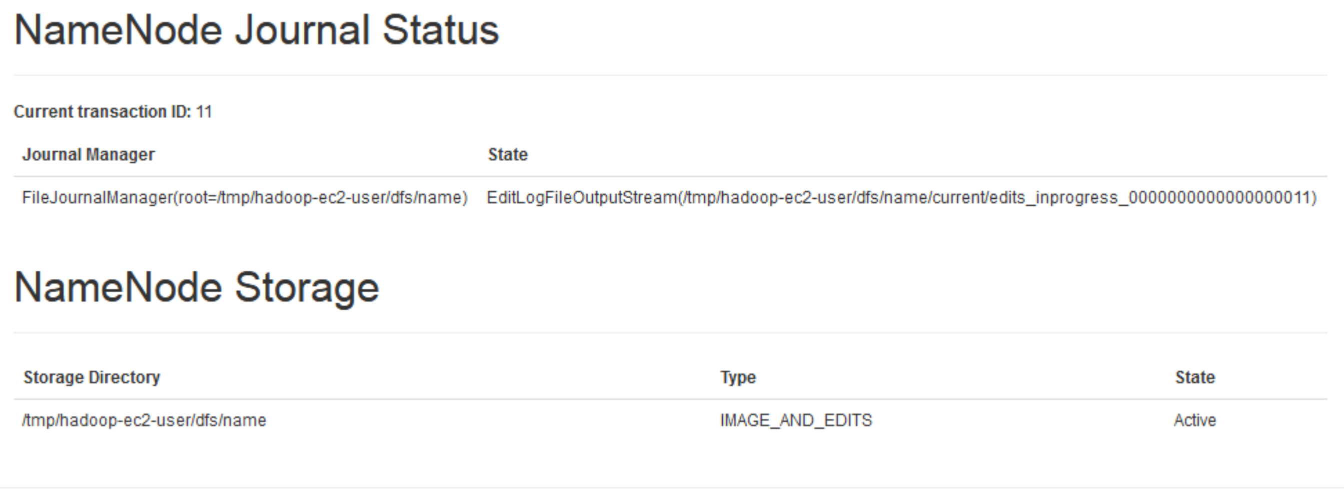
Safemode is off.

7 files and directories, 0 blocks = 7 total filesystem object(s).

Heap Memory used 134.15 MB of 194 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 38.83 MB of 39.81 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	89.96 GB
DFS Used:	12 KB
Non DFS Used:	6.27 GB
DFS Remaining:	83.7 GB
DFS Used%:	0%
DFS Remaining%:	93.04%
Block Pool Used:	12 KB
Block Pool Used%:	0%
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	5/18/2019, 4:27:06 PM



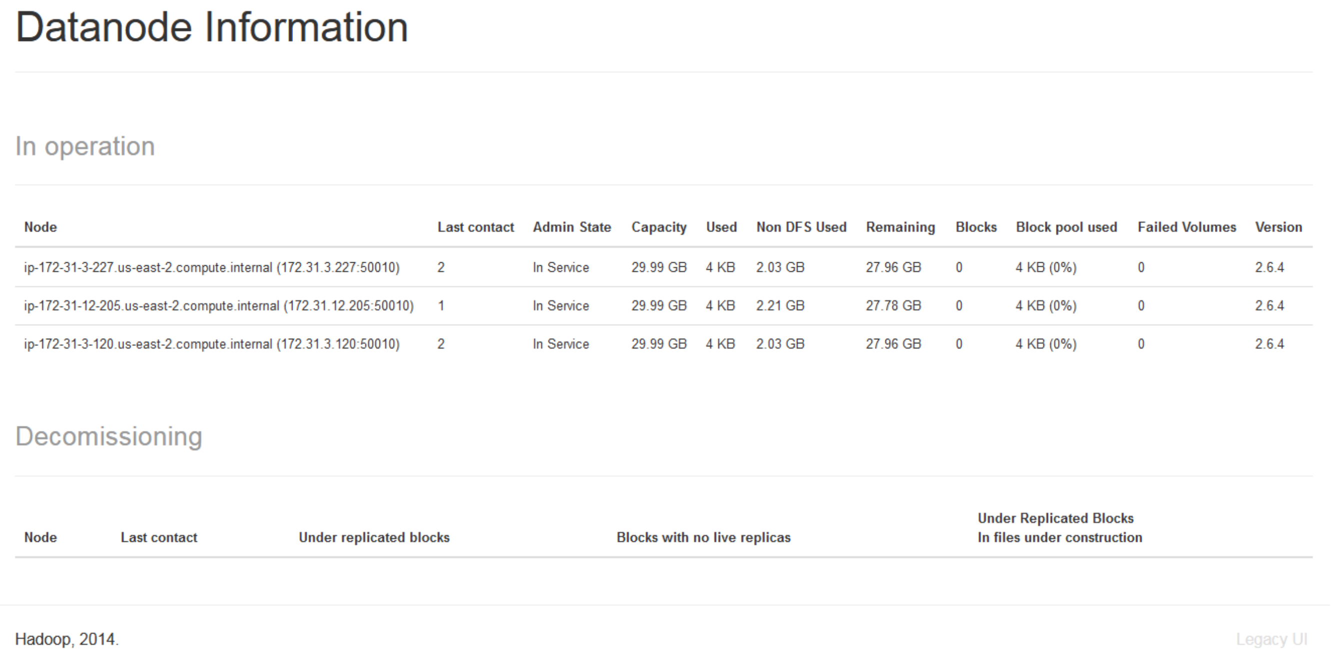
NameNode Journal Status

Current transaction ID: 11

Journal Manager	State
FileJournalManager(root=/tmp/hadoop-ec2-user/dfs/name)	EditLogFileOutputStream(/tmp/hadoop-ec2-user/dfs/name/current/edits_inprogress_0000000000000000011)

NameNode Storage

Storage Directory	Type	State
/tmp/hadoop-ec2-user/dfs/name	IMAGE_AND_EDITS	Active



Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
ip-172-31-3-227.us-east-2.compute.internal (172.31.3.227:50010)	2	In Service	29.99 GB	4 KB	2.03 GB	27.96 GB	0	4 KB (0%)	0	2.6.4
ip-172-31-12-205.us-east-2.compute.internal (172.31.12.205:50010)	1	In Service	29.99 GB	4 KB	2.21 GB	27.78 GB	0	4 KB (0%)	0	2.6.4
ip-172-31-3-120.us-east-2.compute.internal (172.31.3.120:50010)	2	In Service	29.99 GB	4 KB	2.03 GB	27.96 GB	0	4 KB (0%)	0	2.6.4

Decomissioning

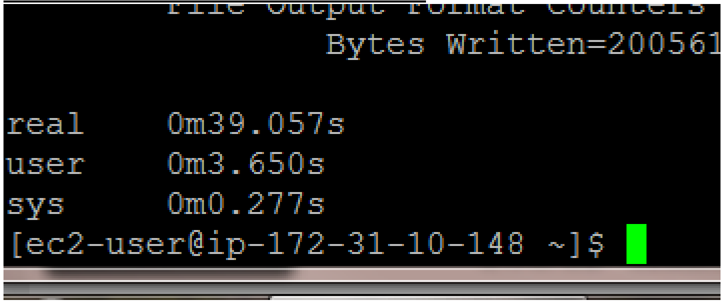
Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction

Hadoop, 2014.

Repeat the steps for wordcount using bioproject.xml from Assignment 1 and submit screenshots of running it.

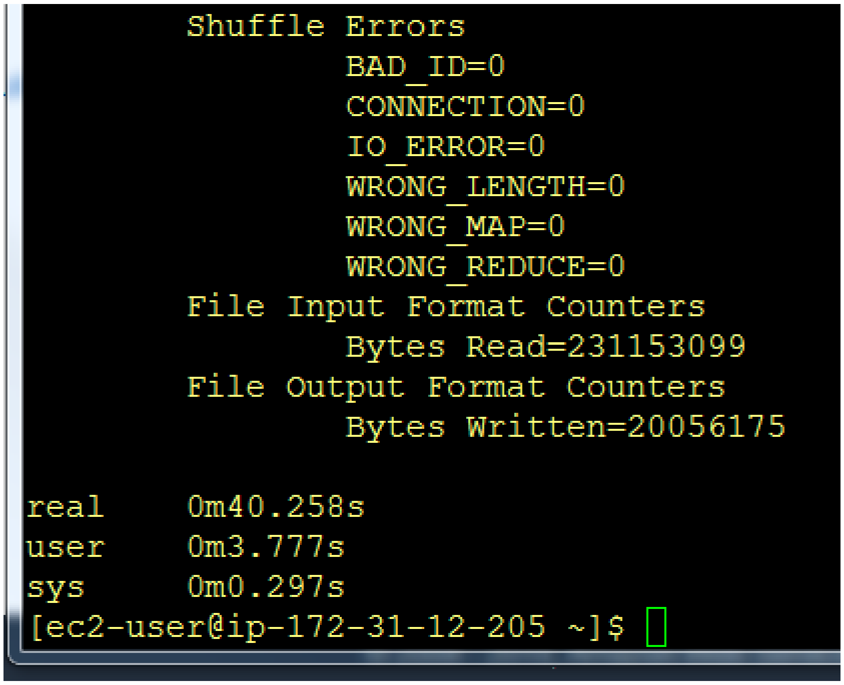
Submit a short paragraph with a discussion about how the results compare (faster? slower? How much faster/slower? Due to what?)

Wordcount run on 1 node cluster

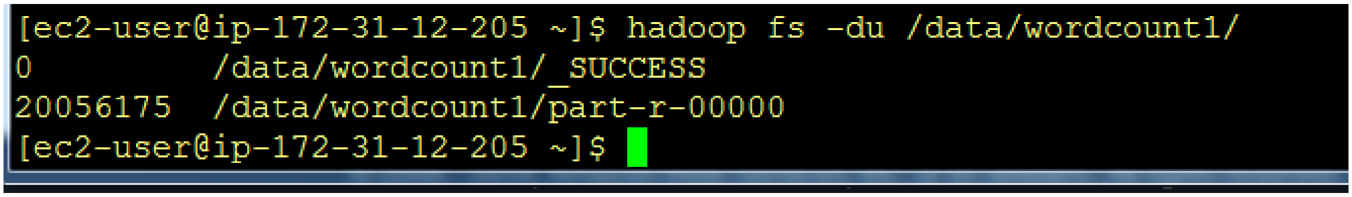


```
        File Output Format Counters
                Bytes Written=200561

real    0m39.057s
user    0m3.650s
sys     0m0.277s
[ec2-user@ip-172-31-10-148 ~]$
```

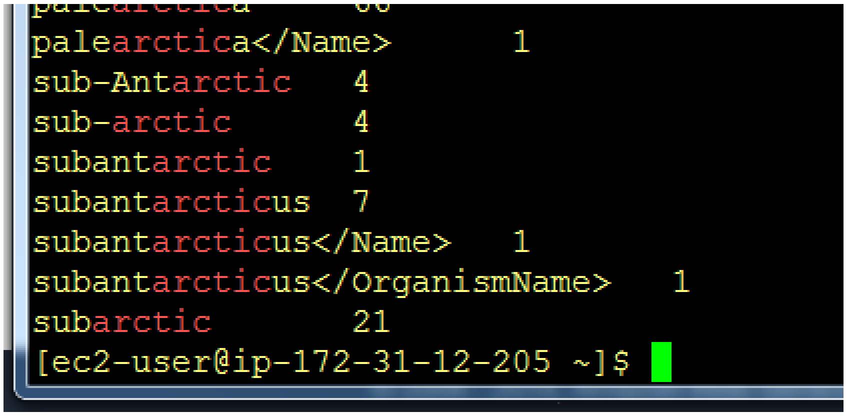
Wordcount run on 3 node cluster



```
        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=231153099
        File Output Format Counters
                Bytes Written=20056175

real    0m40.258s
user    0m3.777s
sys     0m0.297s
[ec2-user@ip-172-31-12-205 ~]$
```



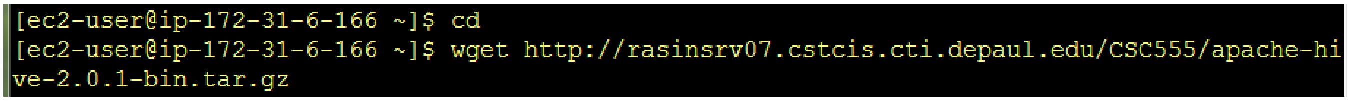
```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -du /data/wordcount1/
0          /data/wordcount1/_SUCCESS
20056175   /data/wordcount1/part-r-00000
[ec2-user@ip-172-31-12-205 ~]$
```



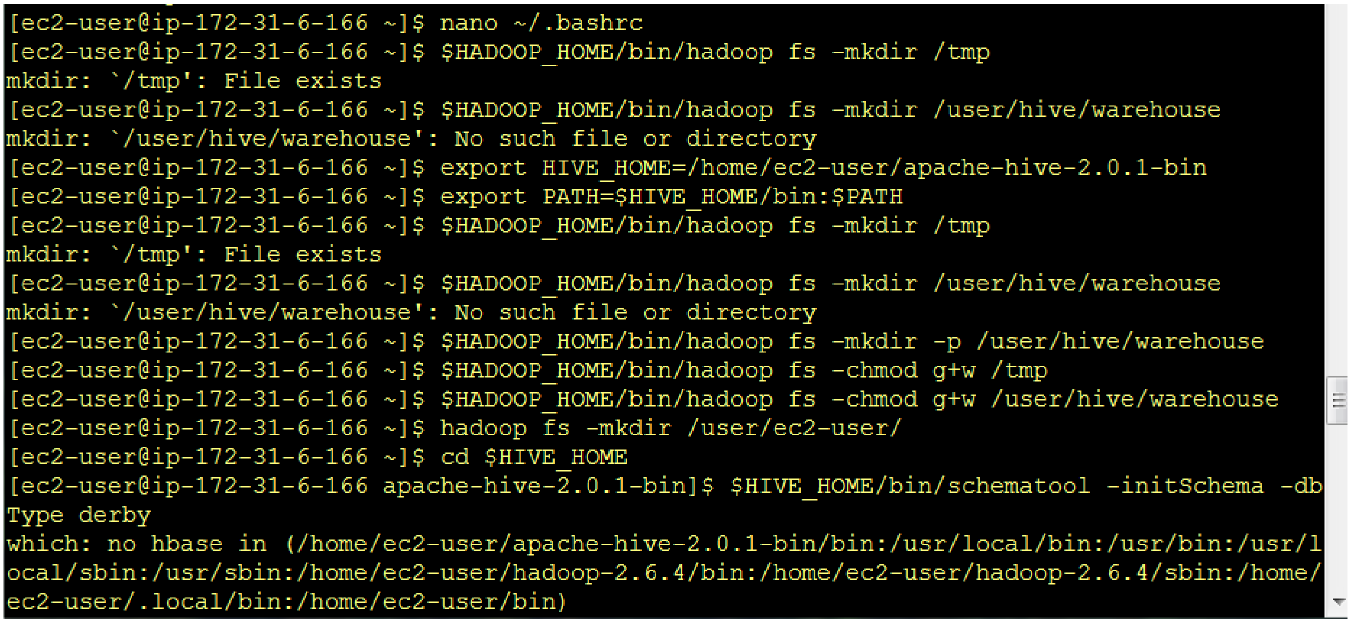
```
palearctica     66
palearctica</Name>      1
sub-Antarctic   4
sub-arctic      4
subantarctic    1
subantarcticus  7
subantarcticus</Name>   1
subantarcticus</OrganismName>   1
subarctic       21
[ec2-user@ip-172-31-12-205 ~]$
```

I expected that wordcount would take less time when run on 3 node cluster compared to 2 node cluster, however it took around 1 second longer. It is possible that this was caused by replication, which was set up as 2. Instances in both cases were the same type.

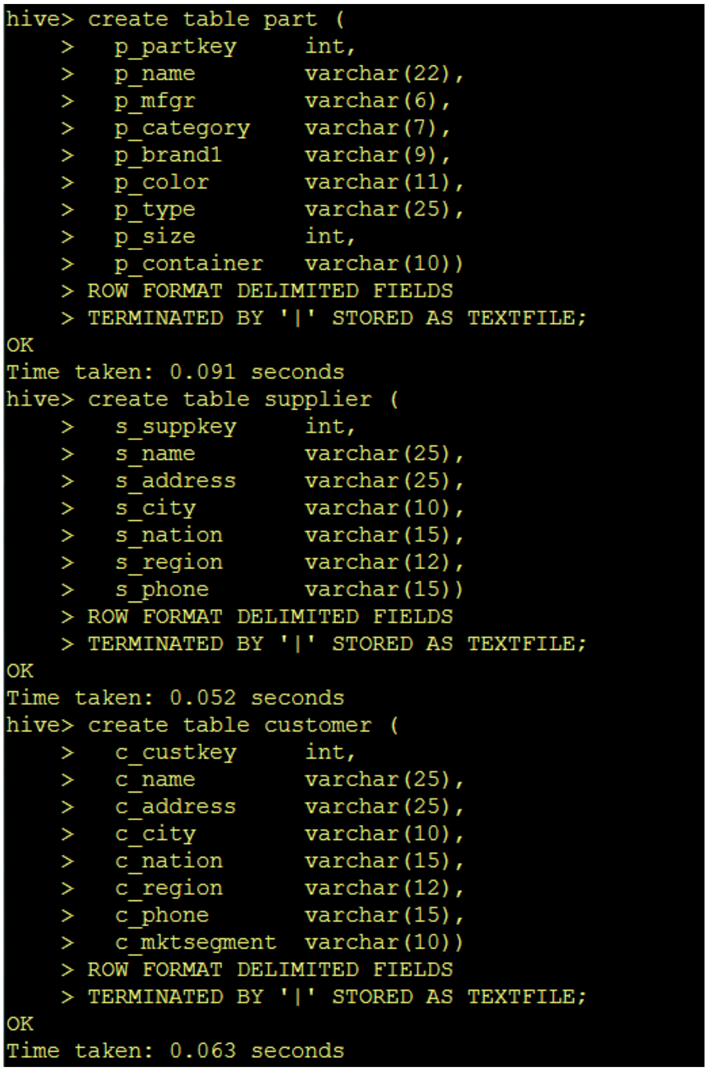
Part 2: Hive



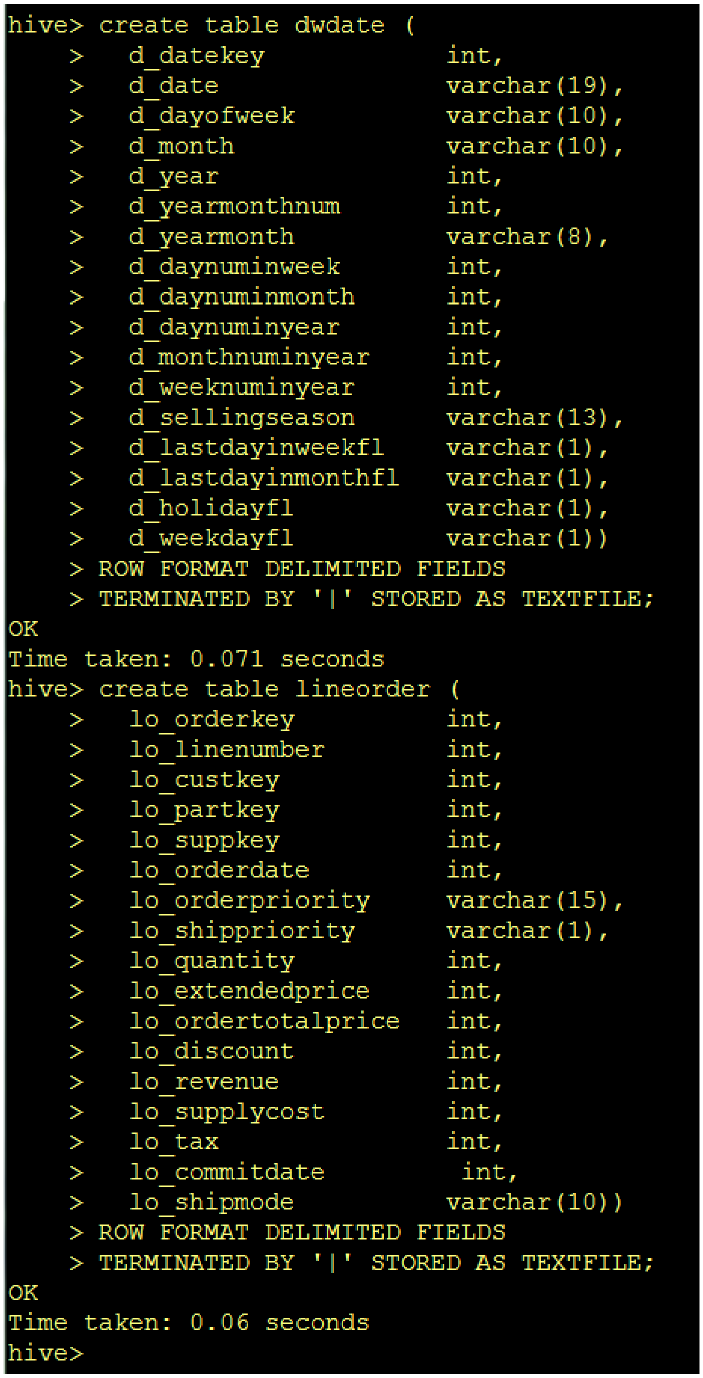
```
[ec2-user@ip-172-31-6-166 ~]$ cd
[ec2-user@ip-172-31-6-166 ~]$ wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-hi
ve-2.0.1-bin.tar.gz
```



```
[ec2-user@ip-172-31-6-166 ~]$ nano ~/.bashrc
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
mkdir: `/tmp': File exists
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
mkdir: `/user/hive/warehouse': No such file or directory
[ec2-user@ip-172-31-6-166 ~]$ export HIVE_HOME=/home/ec2-user/apache-hive-2.0.1-bin
[ec2-user@ip-172-31-6-166 ~]$ export PATH=$HIVE_HOME/bin:$PATH
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
mkdir: `/tmp': File exists
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
mkdir: `/user/hive/warehouse': No such file or directory
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
[ec2-user@ip-172-31-6-166 ~]$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse
[ec2-user@ip-172-31-6-166 ~]$ hadoop fs -mkdir /user/ec2-user/
[ec2-user@ip-172-31-6-166 ~]$ cd $HIVE_HOME
[ec2-user@ip-172-31-6-166 apache-hive-2.0.1-bin]$ $HIVE_HOME/bin/schematool -initSchema -db
Type derby
which: no hbase in (/home/ec2-user/apache-hive-2.0.1-bin/bin:/usr/local/bin:/usr/bin:/usr/l
ocal/sbin:/usr/sbin:/home/ec2-user/hadoop-2.6.4/bin:/home/ec2-user/hadoop-2.6.4/sbin:/home/
ec2-user/.local/bin:/home/ec2-user/bin)
```



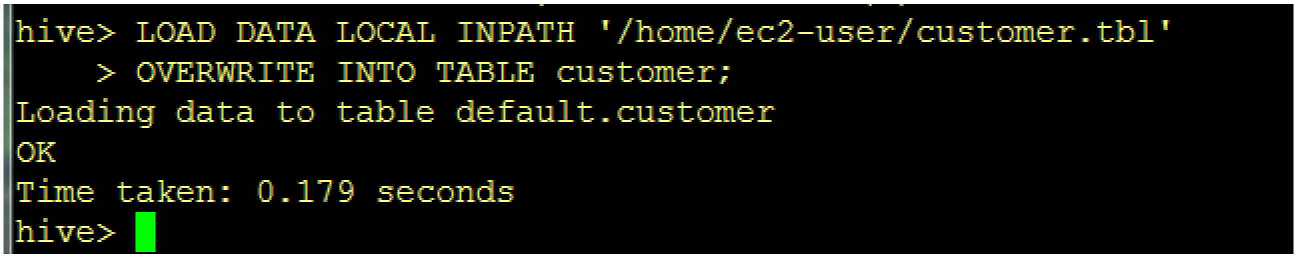
```
hive> create table part (
    >   p_partkey     int,
    >   p_name        varchar(22),
    >   p_mfgr        varchar(6),
    >   p_category    varchar(7),
    >   p_brand1      varchar(9),
    >   p_color       varchar(11),
    >   p_type        varchar(25),
    >   p_size        int,
    >   p_container   varchar(10))
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '|' STORED AS TEXTFILE;
OK
Time taken: 0.091 seconds
hive> 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;
OK
Time taken: 0.052 seconds
hive> create table customer (
    >   c_custkey     int,
    >   c_name        varchar(25),
    >   c_address     varchar(25),
    >   c_city        varchar(10),
    >   c_nation      varchar(15),
    >   c_region      varchar(12),
    >   c_phone       varchar(15),
    >   c_mktsegment  varchar(10))
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '|' STORED AS TEXTFILE;
OK
Time taken: 0.063 seconds
```



```
hive> create table dwdate (
    >   d_datekey            int,
    >   d_date               varchar(19),
    >   d_dayofweek          varchar(10),
    >   d_month              varchar(10),
    >   d_year               int,
    >   d_yearmonthnum       int,
    >   d_yearmonth          varchar(8),
    >   d_daynuminweek       int,
    >   d_daynuminmonth      int,
    >   d_daynuminyear       int,
    >   d_monthnuminyear     int,
    >   d_weeknuminyear      int,
    >   d_sellingseason      varchar(13),
    >   d_lastdayinweekfl    varchar(1),
    >   d_lastdayinmonthfl   varchar(1),
    >   d_holidayfl          varchar(1),
    >   d_weekdayfl          varchar(1))
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '|' STORED AS TEXTFILE;
OK
Time taken: 0.071 seconds
hive> 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;
OK
Time taken: 0.06 seconds
hive>
```

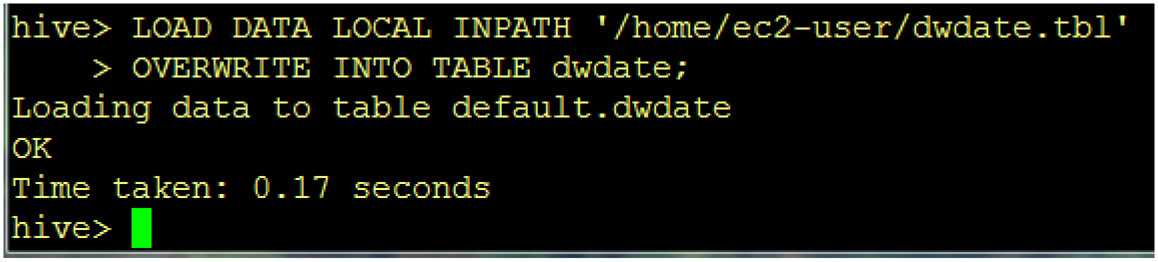
LOAD DATA

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



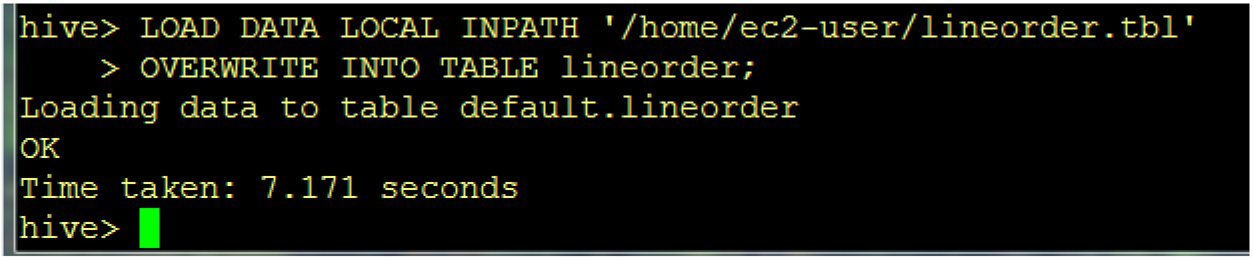
```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl'
    > OVERWRITE INTO TABLE customer;
Loading data to table default.customer
OK
Time taken: 0.179 seconds
hive> 
```

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



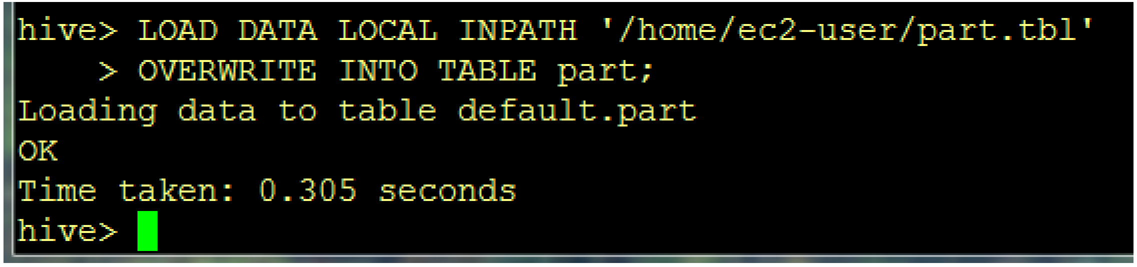
```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/dwdate.tbl'
    > OVERWRITE INTO TABLE dwdate;
Loading data to table default.dwdate
OK
Time taken: 0.17 seconds
hive> 
```

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



```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl'
    > OVERWRITE INTO TABLE lineorder;
Loading data to table default.lineorder
OK
Time taken: 7.171 seconds
hive> 
```

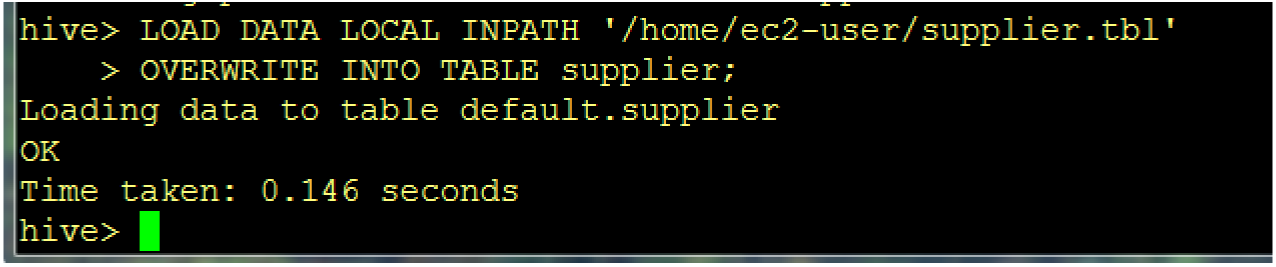
**LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl'
OVERWRITE INTO TABLE part;**



```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl'
    > OVERWRITE INTO TABLE part;
Loading data to table default.part
OK
Time taken: 0.305 seconds
hive> 
```

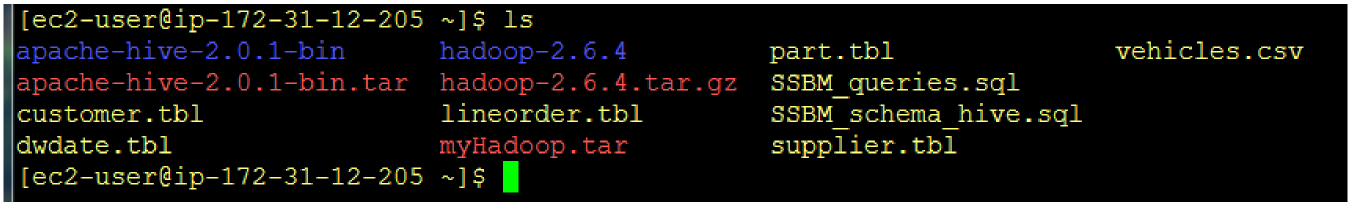
LOAD DATA LOCAL INPATH '/home/ec2-user/supplier.tbl'

OVERWRITE INTO TABLE supplier;



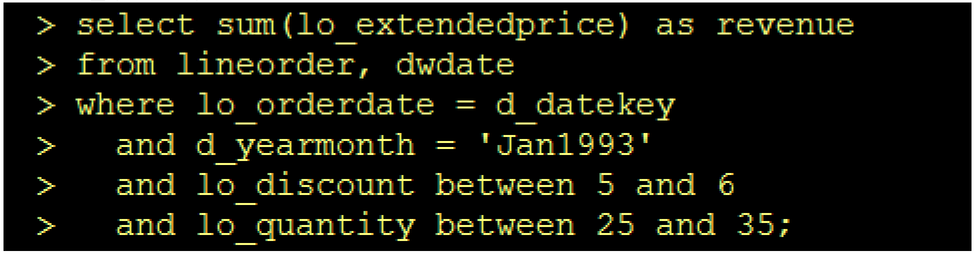
```
hive> LOAD DATA LOCAL INPATH '/home/ec2-user/supplier.tbl'
    > OVERWRITE INTO TABLE supplier;
Loading data to table default.supplier
OK
Time taken: 0.146 seconds
hive>
```

Run the following three (1.2, 1.3 and 2.1) queries in Hive and record the time they take to execute: http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_queries.sql

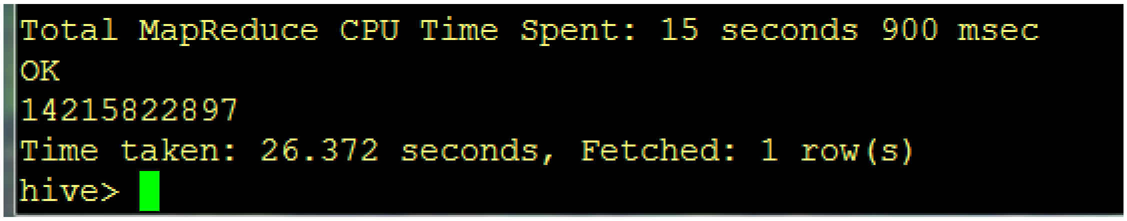


```
[ec2-user@ip-172-31-12-205 ~]$ ls
apache-hive-2.0.1-bin      hadoop-2.6.4         part.tbl             vehicles.csv
apache-hive-2.0.1-bin.tar  hadoop-2.6.4.tar.gz  SSBM_queries.sql
customer.tbl               lineorder.tbl        SSBM_schema_hive.sql
dwdate.tbl                 myHadoop.tar         supplier.tbl
[ec2-user@ip-172-31-12-205 ~]$
```

--Q1.2 Simplified to remove expression in sum
select sum(lo_extendedprice) as revenue
from lineorder, dwdate
where lo_orderdate = d_datekey
  and d_yearmonth = 'Jan1993'
  and lo_discount between 5 and 6
  and lo_quantity between 25 and 35;

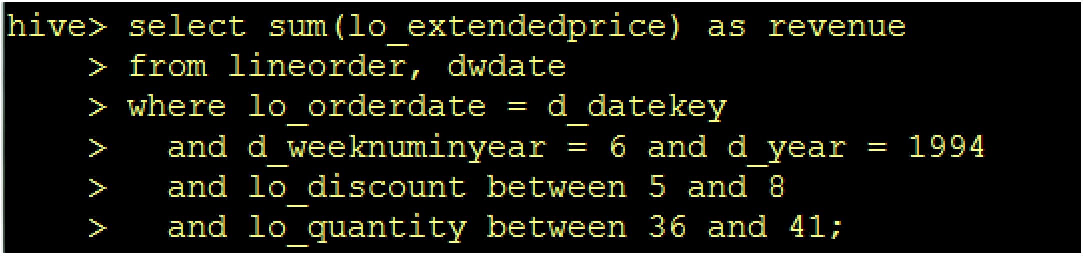


```
 > select sum(lo_extendedprice) as revenue
 > from lineorder, dwdate
 > where lo_orderdate = d_datekey
 >   and d_yearmonth = 'Jan1993'
 >   and lo_discount between 5 and 6
 >   and lo_quantity between 25 and 35;
```

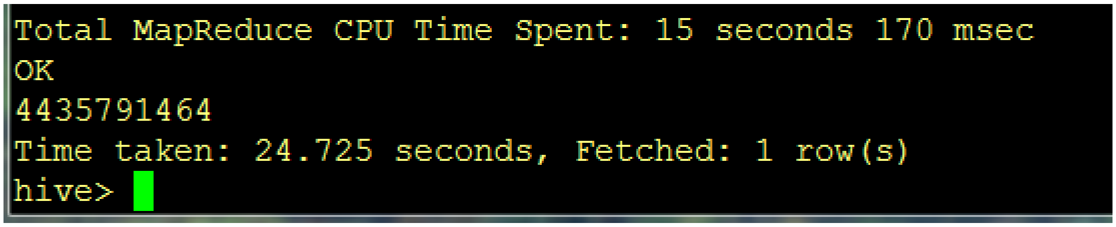


```
Total MapReduce CPU Time Spent: 15 seconds 900 msec
OK
14215822897
Time taken: 26.372 seconds, Fetched: 1 row(s)
hive>
```

--Q1.3 Simplified to remove expression in sum
select sum(lo_extendedprice) as revenue
from lineorder, dwdate
where lo_orderdate = d_datekey
  and d_weeknuminyear = 6 and d_year = 1994
  and lo_discount between 5 and 8
  and lo_quantity between 36 and 41;

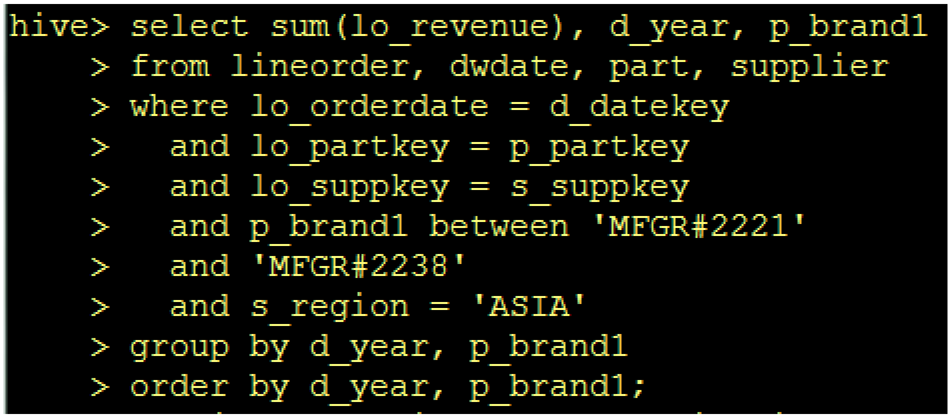


```
hive> select sum(lo_extendedprice) as revenue
    > from lineorder, dwdate
    > where lo_orderdate = d_datekey
    >   and d_weeknuminyear = 6 and d_year = 1994
    >   and lo_discount between 5 and 8
    >   and lo_quantity between 36 and 41;
```

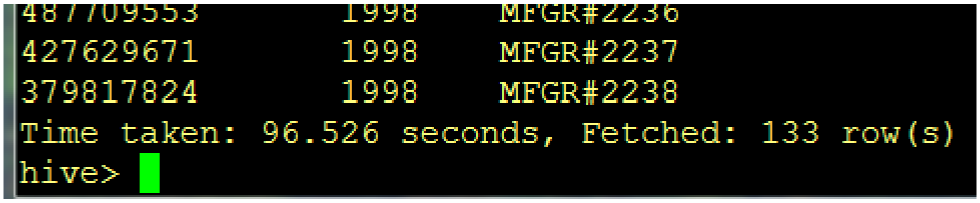


```
Total MapReduce CPU Time Spent: 15 seconds 170 msec
OK
4435791464
Time taken: 24.725 seconds, Fetched: 1 row(s)
hive>
```

--Q2.2 No simpifications
select sum(lo_revenue), d_year, p_brand1
from lineorder, dwdate, part, supplier
where lo_orderdate = d_datekey
  and lo_partkey = p_partkey
  and lo_suppkey = s_suppkey
  and p_brand1 between 'MFGR#2221'
  and 'MFGR#2238'
  and s_region = 'ASIA'
group by d_year, p_brand1
order by d_year, p_brand1;



```
hive> select sum(lo_revenue), d_year, p_brand1
    > from lineorder, dwdate, part, supplier
    > where lo_orderdate = d_datekey
    >   and lo_partkey = p_partkey
    >   and lo_suppkey = s_suppkey
    >   and p_brand1 between 'MFGR#2221'
    >   and 'MFGR#2238'
    >   and s_region = 'ASIA'
    > group by d_year, p_brand1
    > order by d_year, p_brand1;
```

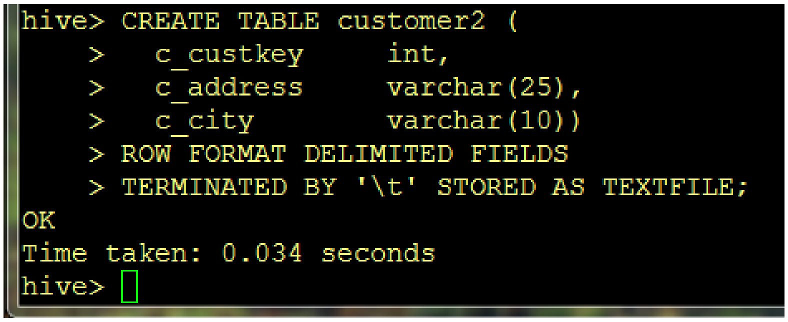


```
487709553       1998    MFGR#2236
427629671       1998    MFGR#2237
379817824       1998    MFGR#2238
Time taken: 96.526 seconds, Fetched: 133 row(s)
hive>
```

Perform the following transform operation using SELECT TRANSFORM on the customer table by creating a new table. Your new target table should have only three columns, c_custkey (no changes), c_address, and c_city.

For the c_address column, shorten it to 8 characters (i.e., if the value is longer, remove extra characters, but otherwise keep it as-is). For c_city, add a space and a # to indicate the digit at the end (e.g., UNITED KI2 => UNITED KI #2, or INDONESIA4 => INDONESIA #4). Make sure to modify the columns of the target table accordingly (since you are introducing longer columns).

```
CREATE TABLE customer2 (
  c_custkey     int,
  c_address     varchar(25),
  c_city        varchar(30))
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' STORED AS TEXTFILE;
```

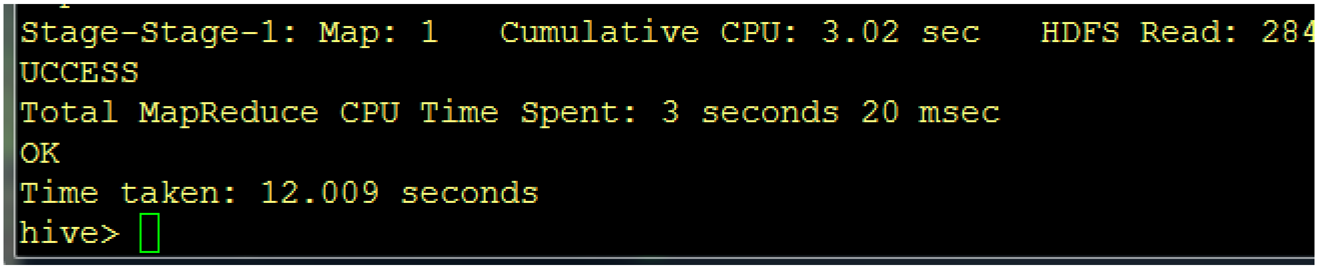


```
hive> CREATE TABLE customer2 (
    >   c_custkey     int,
    >   c_address     varchar(25),
    >   c_city        varchar(10))
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY '\t' STORED AS TEXTFILE;
OK
Time taken: 0.034 seconds
hive>
```

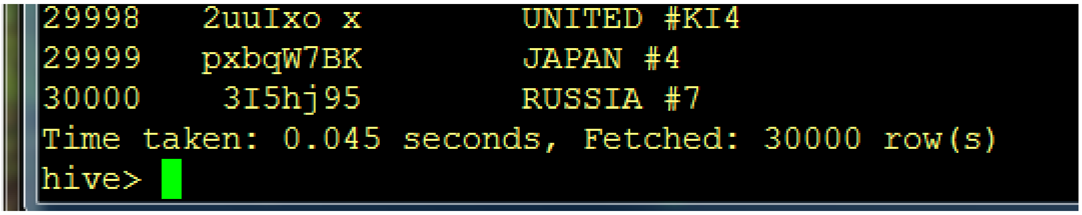
cat /home/ec2-user/customer.tbl | python /home/ec2-user/apache-hive-2.0.1-bin/mapper.py add FILE /home/ec2-user/apache-hive-2.0.1-bin/mapper.py;

```
INSERT OVERWRITE TABLE customer2
SELECT TRANSFORM (c_custkey, c_name, c_address, c_city, c_nation, c_region, c_phone,
c_mktsegment) USING 'python mapper.py'
AS (c_custkey, c_address, c_city) FROM customer;
```

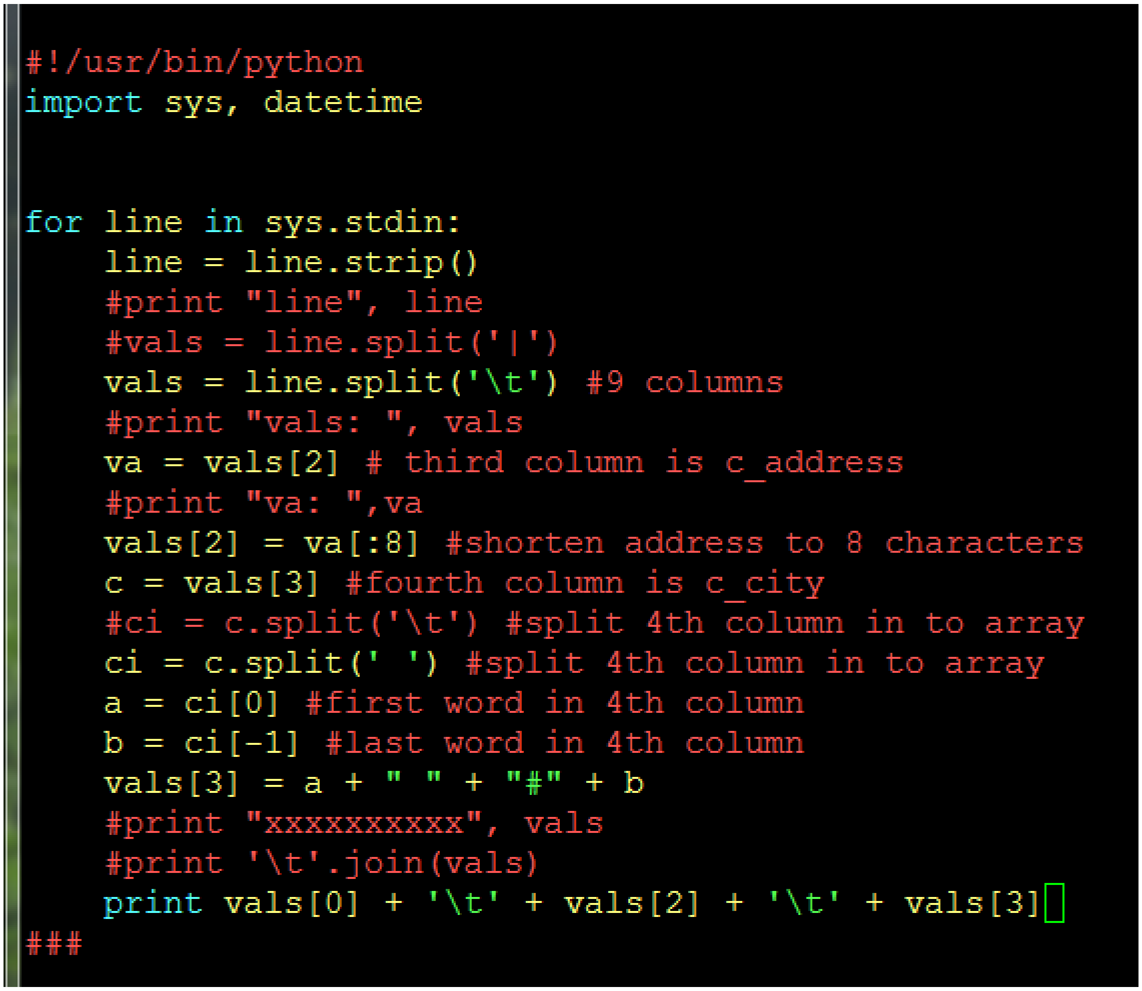
SELECT c_custkey, c_address, c_city FROM customer2;



```
Stage-Stage-1: Map: 1   Cumulative CPU: 3.02 sec   HDFS Read: 284
UCCESS
Total MapReduce CPU Time Spent: 3 seconds 20 msec
OK
Time taken: 12.009 seconds
hive>
```



```
29998   2uuIxo x        UNITED #KI4
29999   pxbqW7BK        JAPAN #4
30000    3I5hj95        RUSSIA #7
Time taken: 0.045 seconds, Fetched: 30000 row(s)
hive>
```

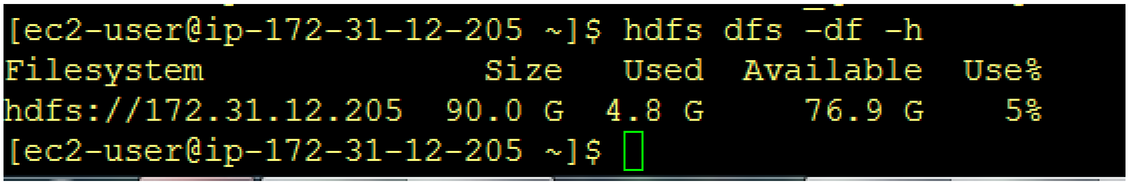


```
#!/usr/bin/python
import sys, datetime


for line in sys.stdin:
    line = line.strip()
    #print "line", line
    #vals = line.split('|')
    vals = line.split('\t') #9 columns
    #print "vals: ", vals
    va = vals[2] # third column is c_address
    #print "va: ",va
    vals[2] = va[:8] #shorten address to 8 characters
    c = vals[3] #fourth column is c_city
    #ci = c.split('\t') #split 4th column in to array
    ci = c.split(' ') #split 4th column in to array
    a = ci[0] #first word in 4th column
    b = ci[-1] #last word in 4th column
    vals[3] = a + " " + "#" + b
    #print "xxxxxxxxxx", vals
    #print '\t'.join(vals)
    print vals[0] + '\t' + vals[2] + '\t' + vals[3]
###
```

Part 3: Pig

Convert and load the data into Pig, implementing only queries 0.1, 0.2, 0.3. Do not implement all queries.

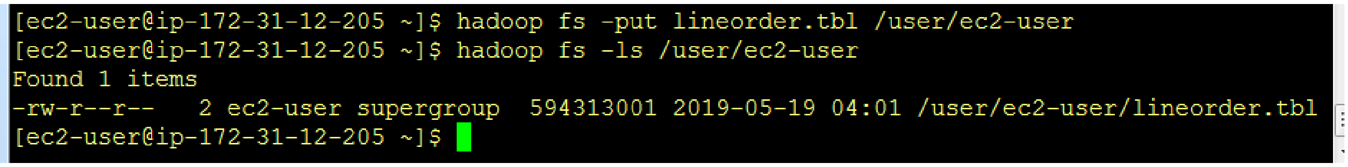
Check disk storage space in HDFS, if your disk usage is over 90% Pig may hang without an error or a warning.



```
[ec2-user@ip-172-31-12-205 ~]$ hdfs dfs -df -h
Filesystem                  Size    Used  Available  Use%
hdfs://172.31.12.205  90.0 G  4.8 G     76.9 G     5%
[ec2-user@ip-172-31-12-205 ~]$
```

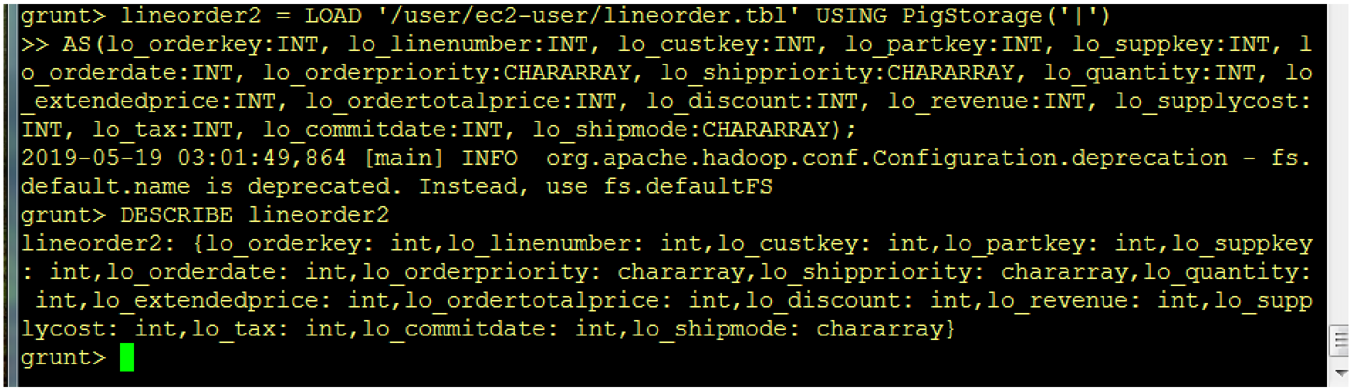
One easy way to time Pig is as follows: put your sequence of pig commands into a text file and then run, from command line in pig directory (e.g., [ec2-user@ip-172-31-6-39 pig-0.15.0]$), **bin/pig -f pig_script.pig** (which will inform you how long the pig script took to run).

```
cd
wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/pig-0.15.0.tar.gz
gunzip pig-0.15.0.tar.gz
tar xvf pig-0.15.0.tar
export PIG_HOME=/home/ec2-user/pig-0.15.0
export PATH=$PATH:$PIG_HOME/bin
cd $PIG_HOME
bin/pig
```



```
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -put lineorder.tbl /user/ec2-user
[ec2-user@ip-172-31-12-205 ~]$ hadoop fs -ls /user/ec2-user
Found 1 items
-rw-r--r--   2 ec2-user supergroup  594313001 2019-05-19 04:01 /user/ec2-user/lineorder.tbl
[ec2-user@ip-172-31-12-205 ~]$
```

lineorder2 = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS(lo_orderkey:INT, lo_linenumber:INT, lo_custkey:INT, lo_partkey:INT, lo_suppkey:INT, lo_orderdate:INT, lo_orderpriority:CHARARRAY, lo_shippriority:CHARARRAY, lo_quantity:INT, lo_extendedprice:INT, lo_ordertotalprice:INT, lo_discount:INT, lo_revenue:INT, lo_supplycost:INT, lo_tax:INT, lo_commitdate:INT, lo_shipmode:CHARARRAY);

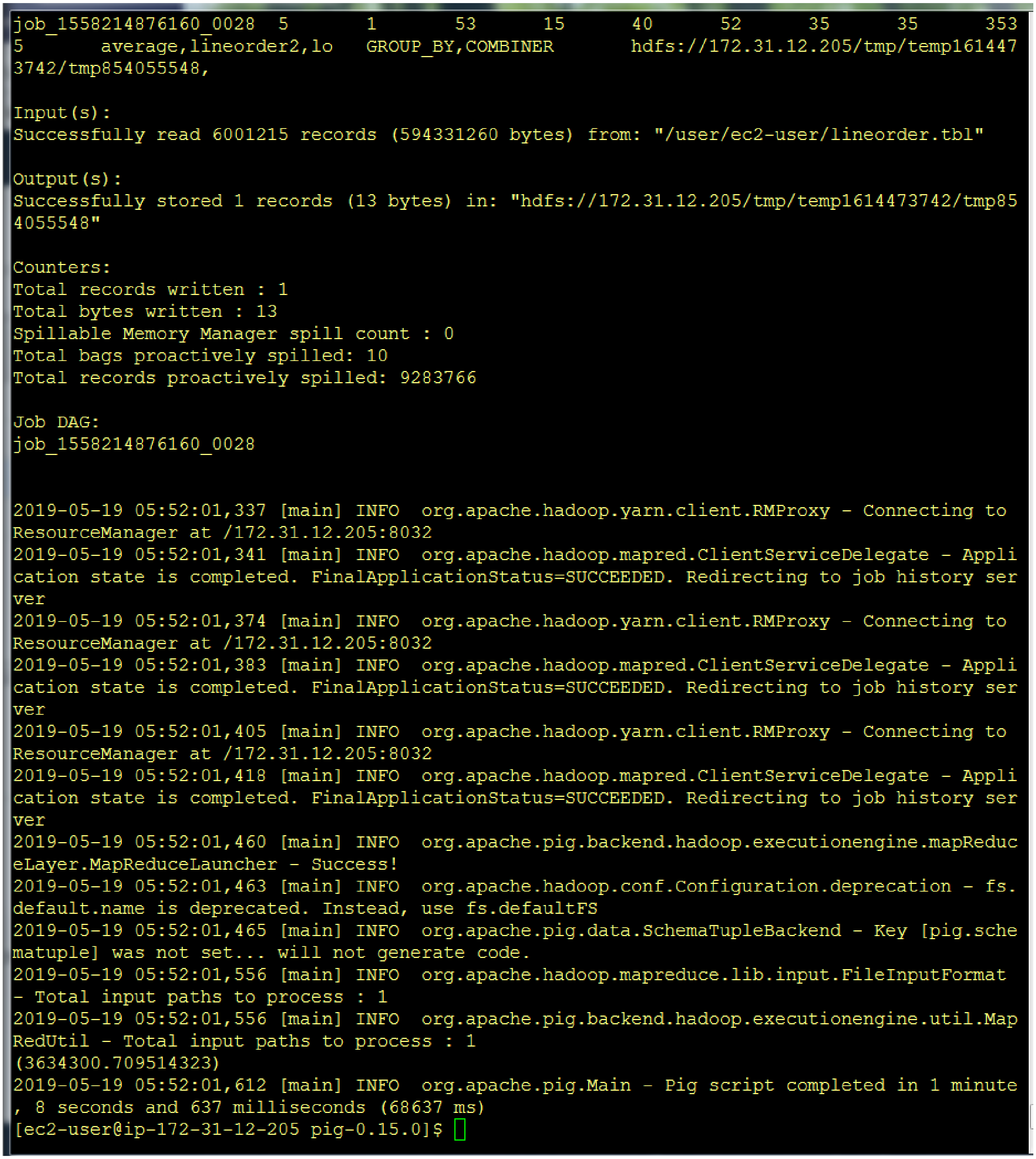


```
grunt> lineorder2 = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
>> AS(lo_orderkey:INT, lo_linenumber:INT, lo_custkey:INT, lo_partkey:INT, lo_suppkey:INT, l
o_orderdate:INT, lo_orderpriority:CHARARRAY, lo_shippriority:CHARARRAY, lo_quantity:INT, lo
_extendedprice:INT, lo_ordertotalprice:INT, lo_discount:INT, lo_revenue:INT, lo_supplycost:
INT, lo_tax:INT, lo_commitdate:INT, lo_shipmode:CHARARRAY);
2019-05-19 03:01:49,864 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.
default.name is deprecated. Instead, use fs.defaultFS
grunt> DESCRIBE lineorder2
lineorder2: {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_supp
lycost: int,lo_tax: int,lo_commitdate: int,lo_shipmode: chararray}
grunt> 
```

```
--Q0.1 Added simple test query
SELECT AVG(lo_revenue)
FROM lineorder;
```

```
lineorder2 = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS(lo_orderkey:INT, lo_linenumber:INT, lo_custkey:INT, lo_partkey:INT, lo_suppkey:INT, lo_orderdate:INT,
lo_orderpriority:CHARARRAY, lo_shippriority:CHARARRAY, lo_quantity:INT, lo_extendedprice:INT,
lo_ordertotalprice:INT, lo_discount:INT, lo_revenue:INT, lo_supplycost:INT, lo_tax:INT, lo_commitdate:INT,
lo_shipmode:CHARARRAY);
lo = GROUP lineorder2 ALL;
average = FOREACH lo GENERATE AVG(lineorder2.lo_revenue);
DUMP average;
STORE average INTO 'query01extract' USING PigStorage(',');
```

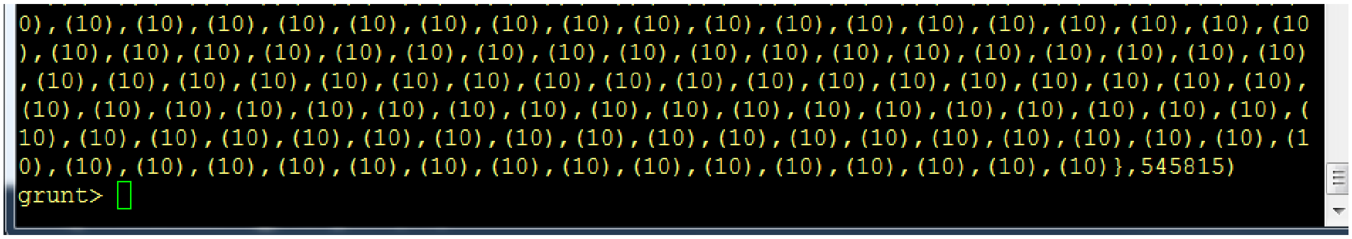
bin/pig -f pig_script1.pig



```
job_1558214876160_0028  5       1       53      15      40      52      35      35      353
5       average,lineorder2,lo   GROUP_BY,COMBINER       hdfs://172.31.12.205/tmp/temp161447
3742/tmp854055548,

Input(s):
Successfully read 6001215 records (594331260 bytes) from: "/user/ec2-user/lineorder.tbl"

Output(s):
Successfully stored 1 records (13 bytes) in: "hdfs://172.31.12.205/tmp/temp1614473742/tmp85
4055548"

Counters:
Total records written : 1
Total bytes written : 13
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 10
Total records proactively spilled: 9283766

Job DAG:
job_1558214876160_0028


2019-05-19 05:52:01,337 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to
ResourceManager at /172.31.12.205:8032
2019-05-19 05:52:01,341 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Appli
cation state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history ser
ver
2019-05-19 05:52:01,374 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to
ResourceManager at /172.31.12.205:8032
2019-05-19 05:52:01,383 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Appli
cation state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history ser
ver
2019-05-19 05:52:01,405 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to
ResourceManager at /172.31.12.205:8032
2019-05-19 05:52:01,418 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Appli
cation state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history ser
ver
2019-05-19 05:52:01,460 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduc
eLayer.MapReduceLauncher - Success!
2019-05-19 05:52:01,463 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.
default.name is deprecated. Instead, use fs.defaultFS
2019-05-19 05:52:01,465 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.sche
matuple] was not set... will not generate code.
2019-05-19 05:52:01,556 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat
- Total input paths to process : 1
2019-05-19 05:52:01,556 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.Map
RedUtil - Total input paths to process : 1
(3634300.709514323)
2019-05-19 05:52:01,612 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute
, 8 seconds and 637 milliseconds (68637 ms)
[ec2-user@ip-172-31-12-205 pig-0.15.0]$
```

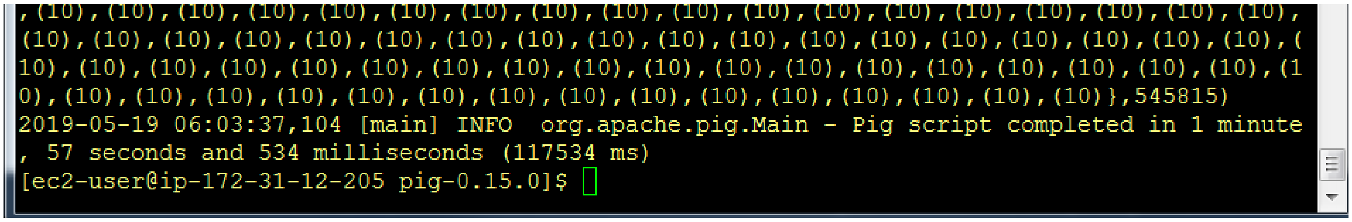
--Q0.2 Added simple test query
SELECT lo_discount, COUNT(lo_extendedprice)
FROM lineorder
GROUP BY lo_discount;

```
lineorder2 = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS(lo_orderkey:INT, lo_linenumber:INT, lo_custkey:INT, lo_partkey:INT, lo_suppkey:INT, lo_orderdate:INT,
lo_orderpriority:CHARARRAY, lo_shippriority:CHARARRAY, lo_quantity:INT, lo_extendedprice:INT,
lo_ordertotalprice:INT, lo_discount:INT, lo_revenue:INT, lo_supplycost:INT, lo_tax:INT, lo_commitdate:INT,
lo_shipmode:CHARARRAY);
lo = GROUP lineorder2 BY lo_discount;
discount = FOREACH lo GENERATE lineorder2.lo_discount, COUNT(lineorder2.lo_extendedprice);
DUMP discount;
STORE discount INTO 'query02extract' USING PigStorage(',');
```



```
0),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10
),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10)
,(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),
(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(
10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(1
0),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10)},545815)
grunt> 
```

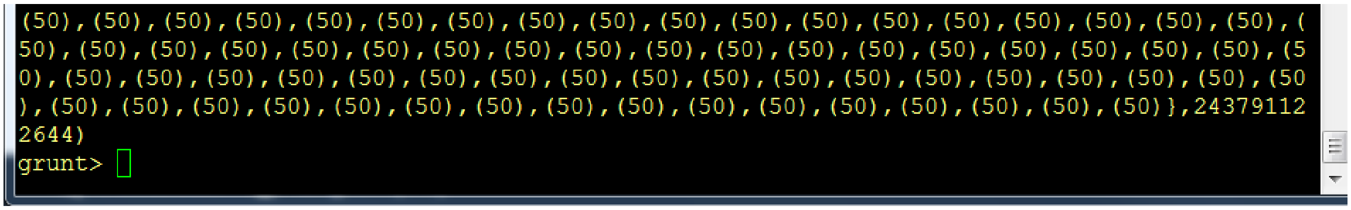
bin/pig -f pig_script2.pig



```
,(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),
(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(
10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(1
0),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10),(10)},545815)
2019-05-19 06:03:37,104 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute
, 57 seconds and 534 milliseconds (117534 ms)
[ec2-user@ip-172-31-12-205 pig-0.15.0]$ 
```

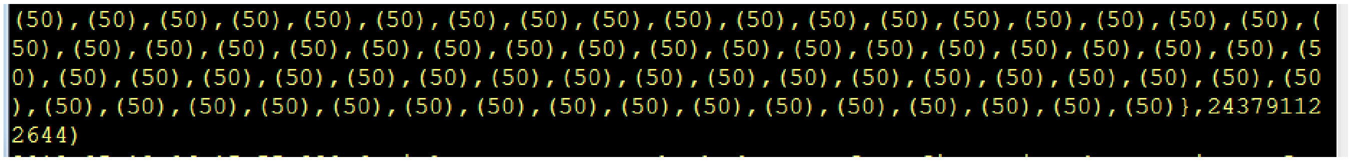
--Q0.3 Added simple test query
SELECT lo_quantity, SUM(lo_revenue)
FROM lineorder
WHERE lo_discount < 3
GROUP BY lo_quantity;

```
lineorder2 = LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')
AS(lo_orderkey:INT, lo_linenumber:INT, lo_custkey:INT, lo_partkey:INT, lo_suppkey:INT, lo_orderdate:INT,
lo_orderpriority:CHARARRAY, lo_shippriority:CHARARRAY, lo_quantity:INT, lo_extendedprice:INT,
lo_ordertotalprice:INT, lo_discount:INT, lo_revenue:INT, lo_supplycost:INT, lo_tax:INT, lo_commitdate:INT,
lo_shipmode:CHARARRAY);
discount = FILTER lineorder2 BY lo_discount < 3;
gr = GROUP discount BY lo_quantity;
res = FOREACH gr GENERATE discount.lo_quantity, SUM(discount.lo_revenue);
DUMP res;
STORE res INTO 'query03extract' USING PigStorage(',');
```

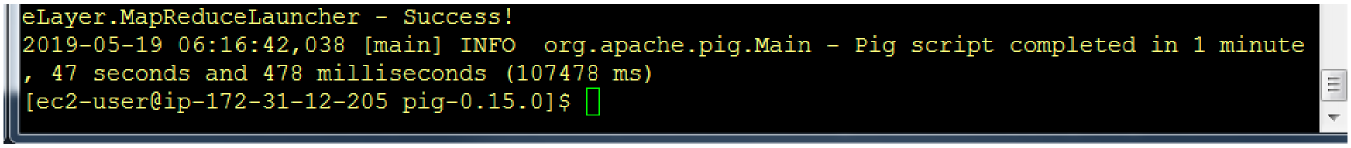


```
(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(
50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(5
0),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50
),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50)},24379112
2644)
grunt>
```

bin/pig -f pig_script3.pig

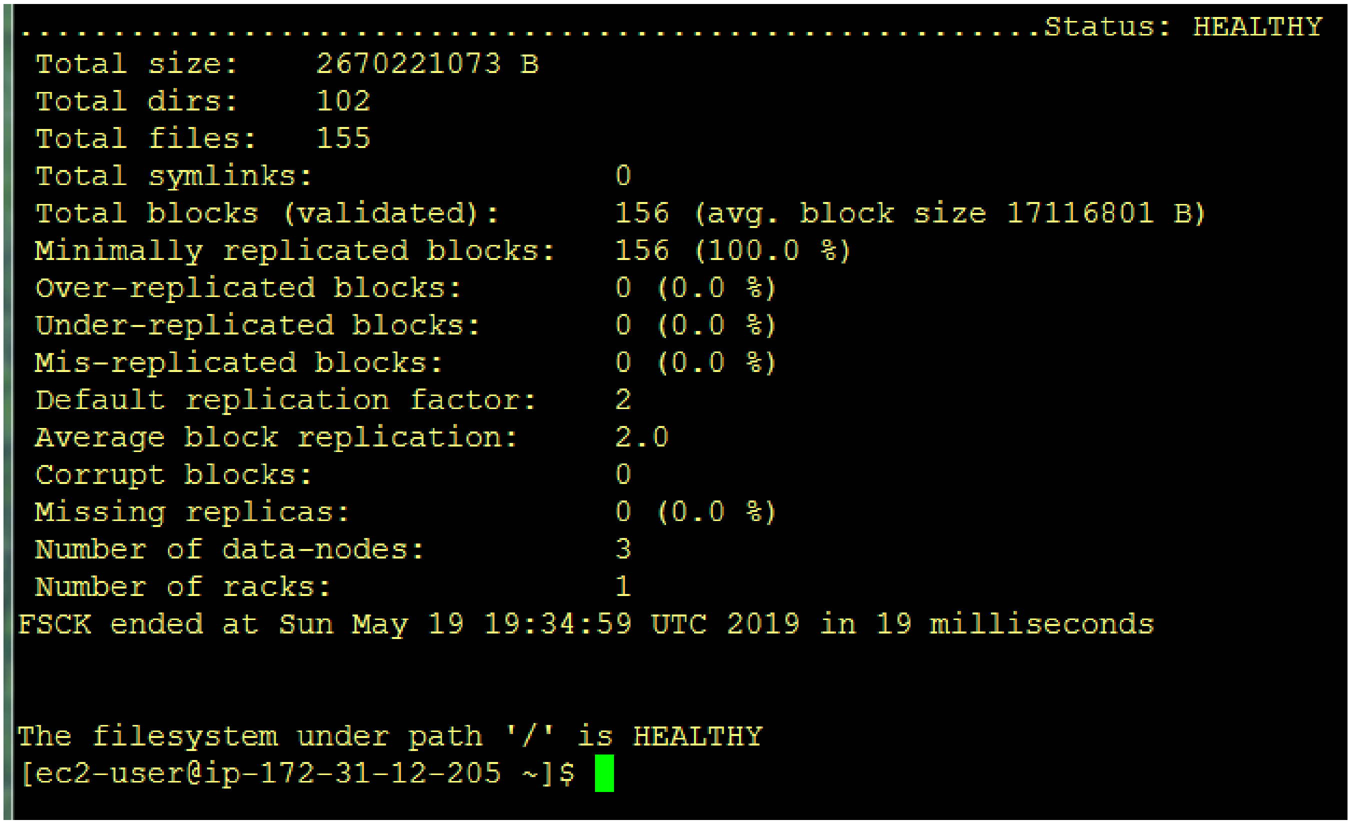


```
(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(
50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(5
0),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50
),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50),(50)},24379112
2644)
```



```
eLayer.MapReduceLauncher - Success!
2019-05-19 06:16:42,038 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute
, 47 seconds and 478 milliseconds (107478 ms)
[ec2-user@ip-172-31-12-205 pig-0.15.0]$
```

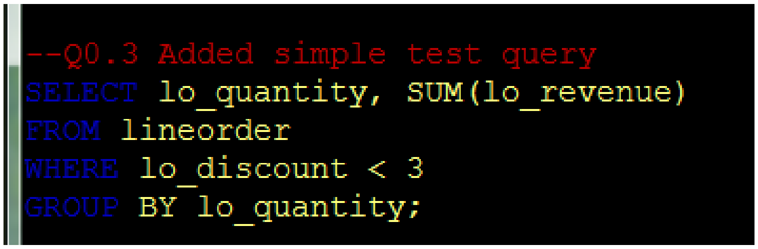
Part 4: Hadoop Streaming



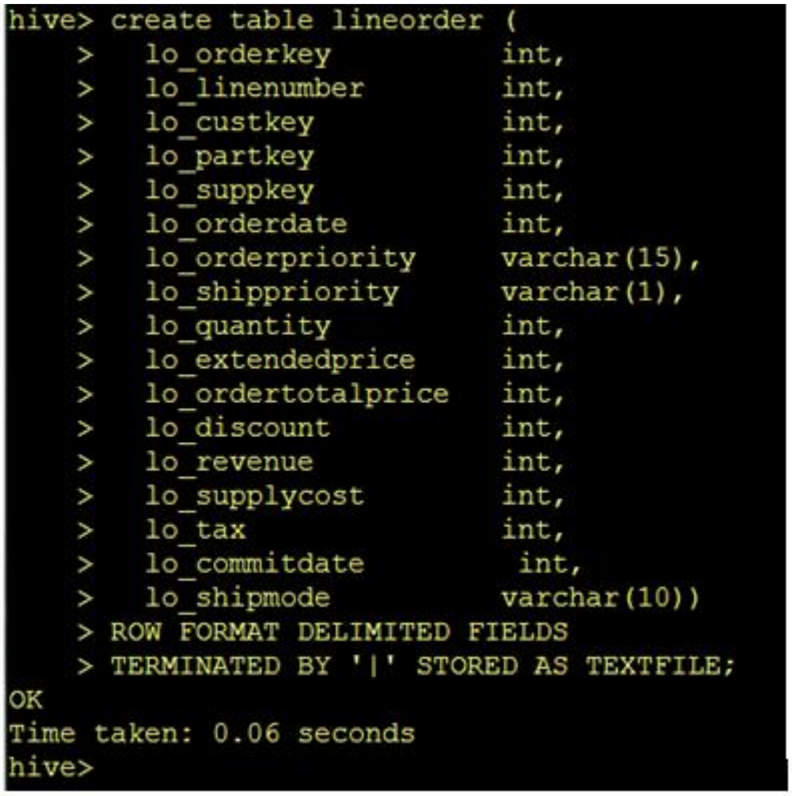
```
......................................................Status: HEALTHY
 Total size:    2670221073 B
 Total dirs:    102
 Total files:   155
 Total symlinks:                0
 Total blocks (validated):      156 (avg. block size 17116801 B)
 Minimally replicated blocks:   156 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     2.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          3
 Number of racks:               1
FSCK ended at Sun May 19 19:34:59 UTC 2019 in 19 milliseconds


The filesystem under path '/' is HEALTHY
[ec2-user@ip-172-31-12-205 ~]$
```

Implement query **0.3** using Hadoop streaming with python. You don't need to implement other queries.

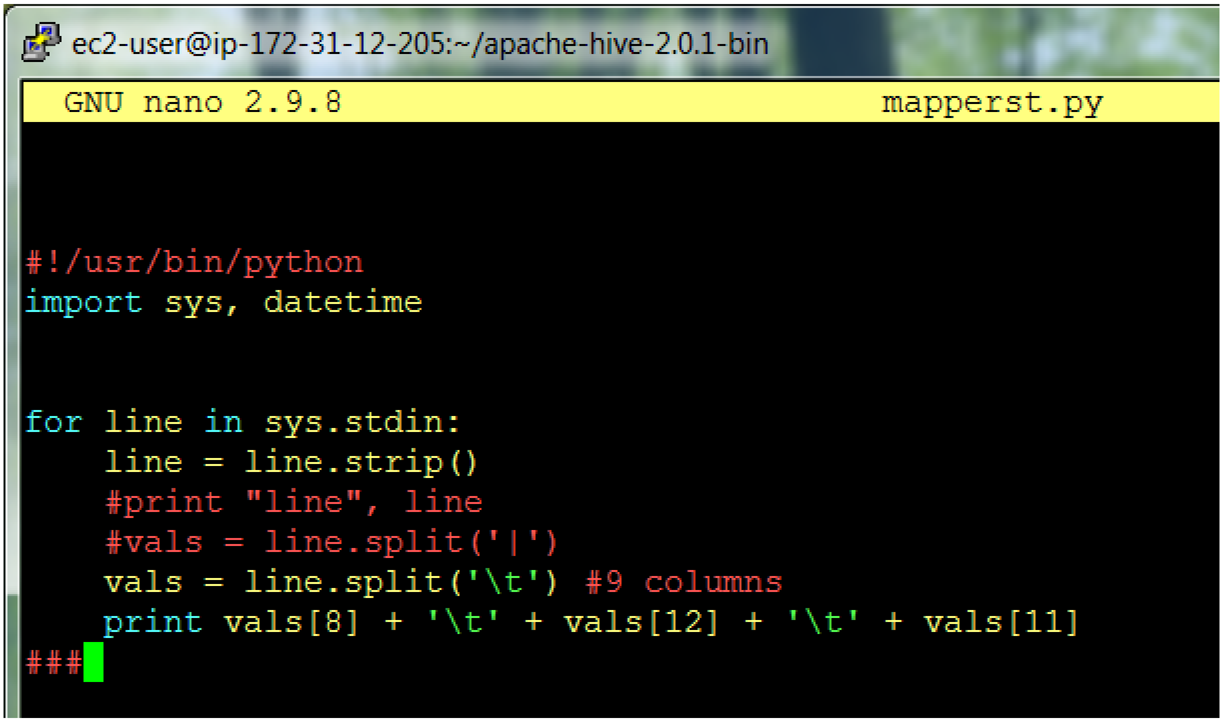
```
--Q0.3
SELECT lo_quantity, SUM(lo_revenue)
FROM lineorder
WHERE lo_discount < 3
GROUP BY lo_quantity;
```



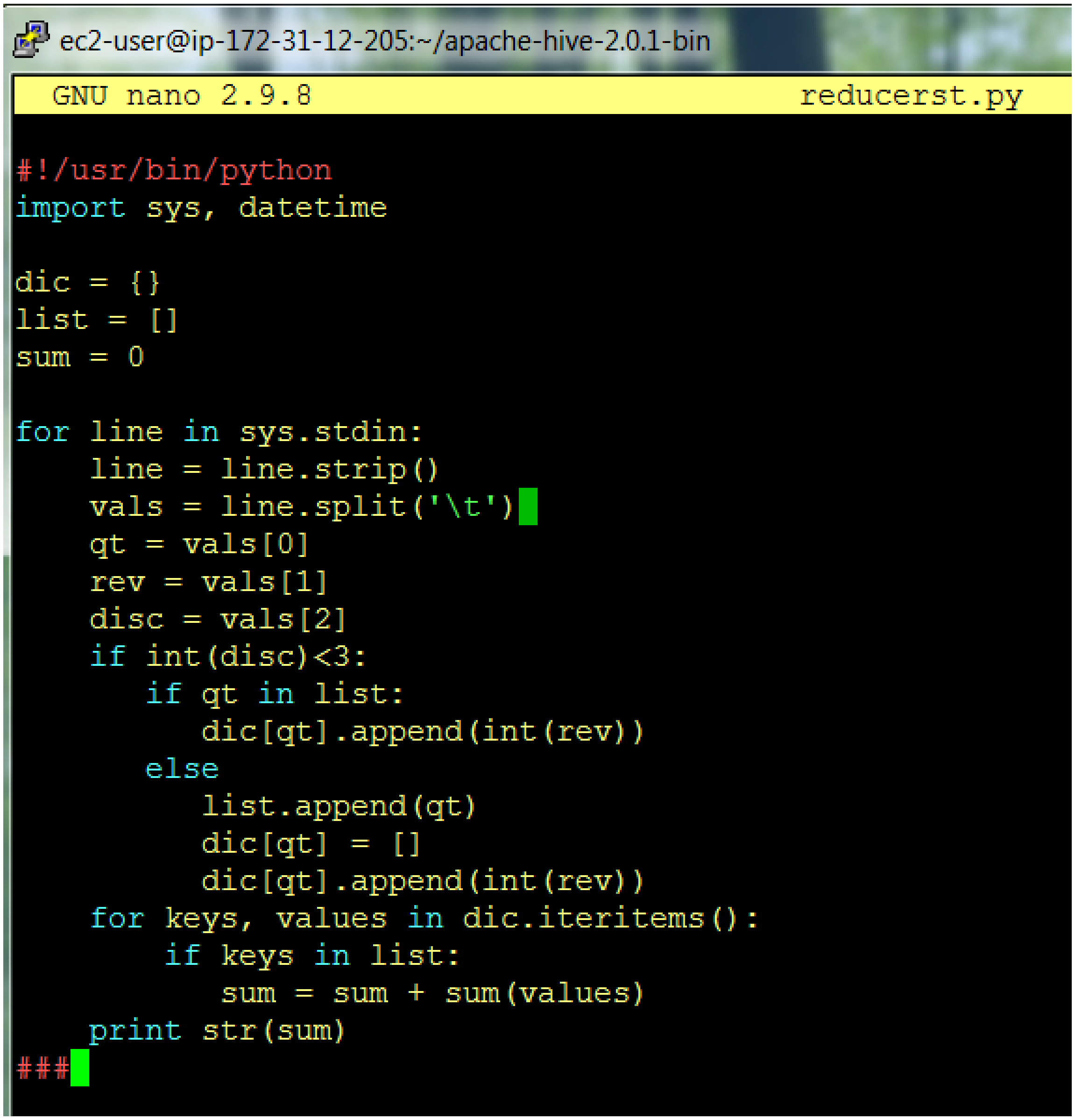
```
--Q0.3 Added simple test query
SELECT lo_quantity, SUM(lo_revenue)
FROM lineorder
WHERE lo_discount < 3
GROUP BY lo_quantity;
```



```
hive> 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;
OK
Time taken: 0.06 seconds
hive>
```



```
  GNU nano 2.9.8                          mapperst.py



#!/usr/bin/python
import sys, datetime


for line in sys.stdin:
    line = line.strip()
    #print "line", line
    #vals = line.split('|')
    vals = line.split('\t') #9 columns
    print vals[8] + '\t' + vals[12] + '\t' + vals[11]
###
```



```
  GNU nano 2.9.8                                 reducerst.py

#!/usr/bin/python
import sys, datetime

dic = {}
list = []
sum = 0

for line in sys.stdin:
    line = line.strip()
    vals = line.split('\t')
    qt = vals[0]
    rev = vals[1]
    disc = vals[2]
    if int(disc)<3:
       if qt in list:
          dic[qt].append(int(rev))
       else
          list.append(qt)
          dic[qt] = []
          dic[qt].append(int(rev))
    for keys, values in dic.iteritems():
        if keys in list:
           sum = sum + sum(values)
    print str(sum)
###
```

NOTE: You may implement this part in Java if you prefer.

Submit a single document containing your written answers. Be sure that this document contains your name and “CSC 555 Project Phase 1” at the top.