# Big Data Hadoop Training

Session 16 Assignment 3 Solution:

**Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.**

**Note:** Hive 0.14 should be installed to implement the hive transaction property.

**Transactions in Hive**

Transactions in Hive are introduced in Hive 0.13, but they only partially fulfill the ACID properties like atomicity, consistency, durability, at the partition level. Here, Isolation can be provided by turning on one of the locking mechanisms available with zookeeper or in memory.

But in Hive 0.14, new API’s have been added to completely fulfill the ACID properties while performing any transaction.

Transactions are provided at the row-level in Hive 0.14. The different row-level transactions available in Hive 0.14 are as follows:

**Insert**

**Delete**

**Update**

There are numerous limitations with the present transactions available in Hive 0.14. ORC is the file format supported by Hive transaction. It is now essential to have ORC file format for performing transactions in Hive. The table needs to be bucketed in order to support transactions.

**Row-level Transactions Available in Hive 0.14**

Let’s perform some row-level transactions available in Hive 0.14. Before creating a Hive table that supports transactions, the transaction features present in Hive needs to be turned on, as by default they are turned off.

**The below properties needs to be set appropriately in hive shell, order-wise to work with transactions in Hive:**

hive>set hive.support.concurrency = true;

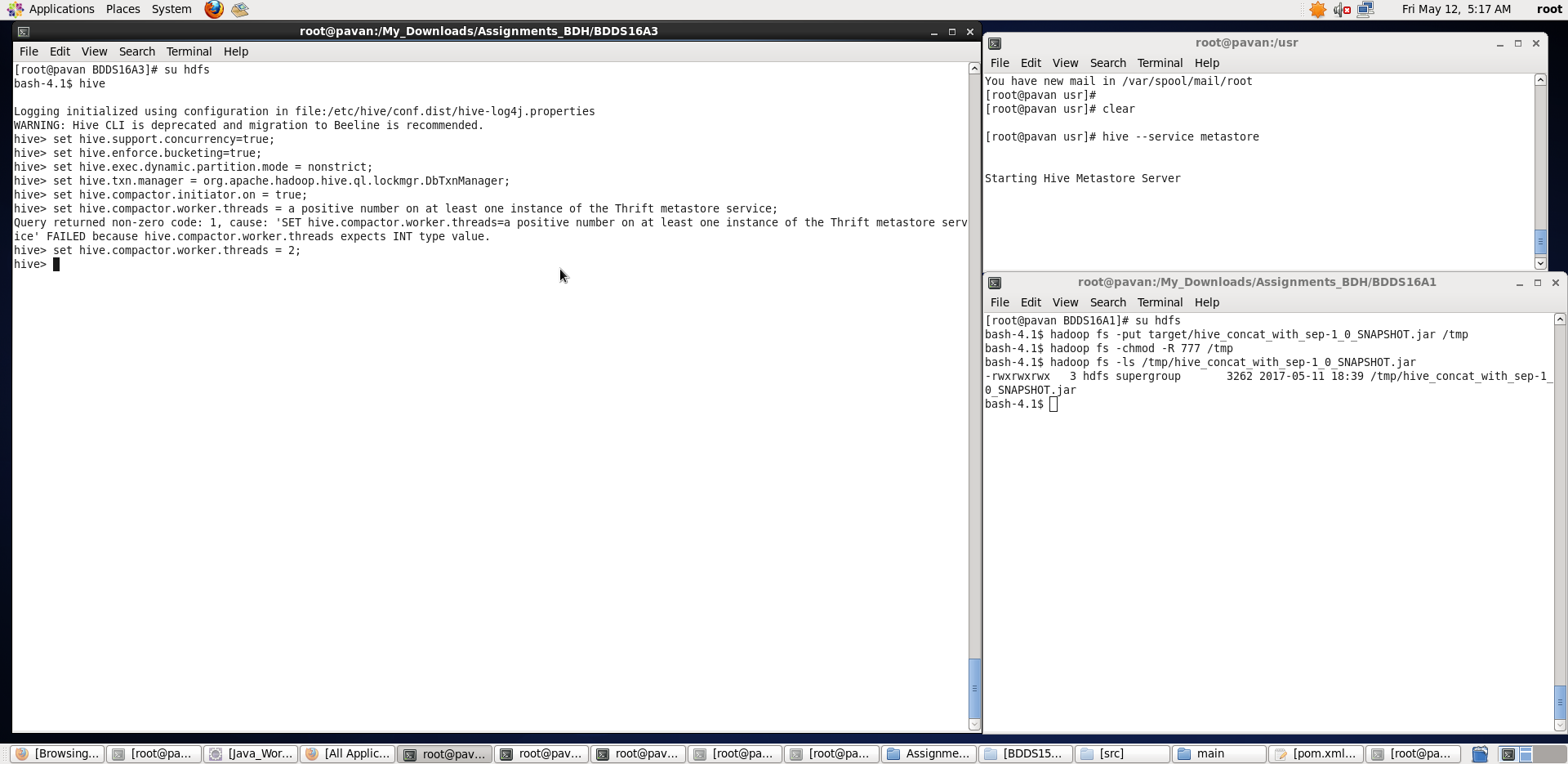
hive>set hive.enforce.bucketing = true;

hive>set hive.exec.dynamic.partition.mode = nonstrict;

hive>set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;

hive>set hive.compactor.initiator.on = true;

hive>set hive.compactor.worker.threads = 2;



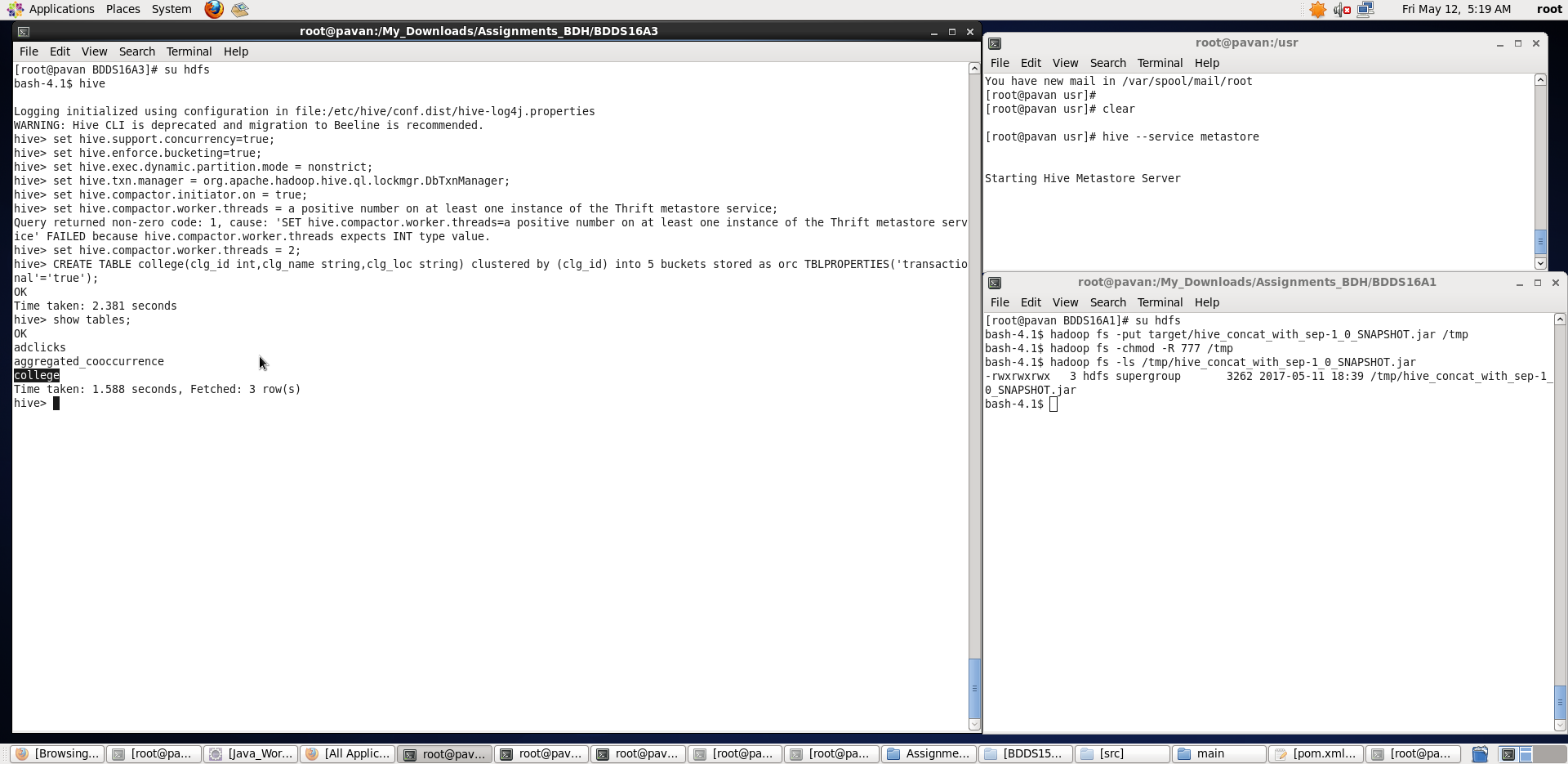
## Once, the properties are set, we will be creating a table that supports Hive Transactions:

hive>CREATE TABLE college(clg\_id int,clg\_name string,clg\_loc string) clustered by (clg\_id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');

The above syntax will create a table with name ‘*college’*and the columns present in the table are ‘*clg\_id, clg\_name, clg\_loc’. W*e are *bucketing* the table by ‘*clg\_id’*and the table format is ‘*orc’,*also we are enabling the transactions in the table by specifying it inside the *TBLPROPERTIES* as *‘transactional’=’true’*

The create table can be checked using the command ***show tables.***

We have successfully created a table with name ‘*college’*which supports row-level transactions of Hive.



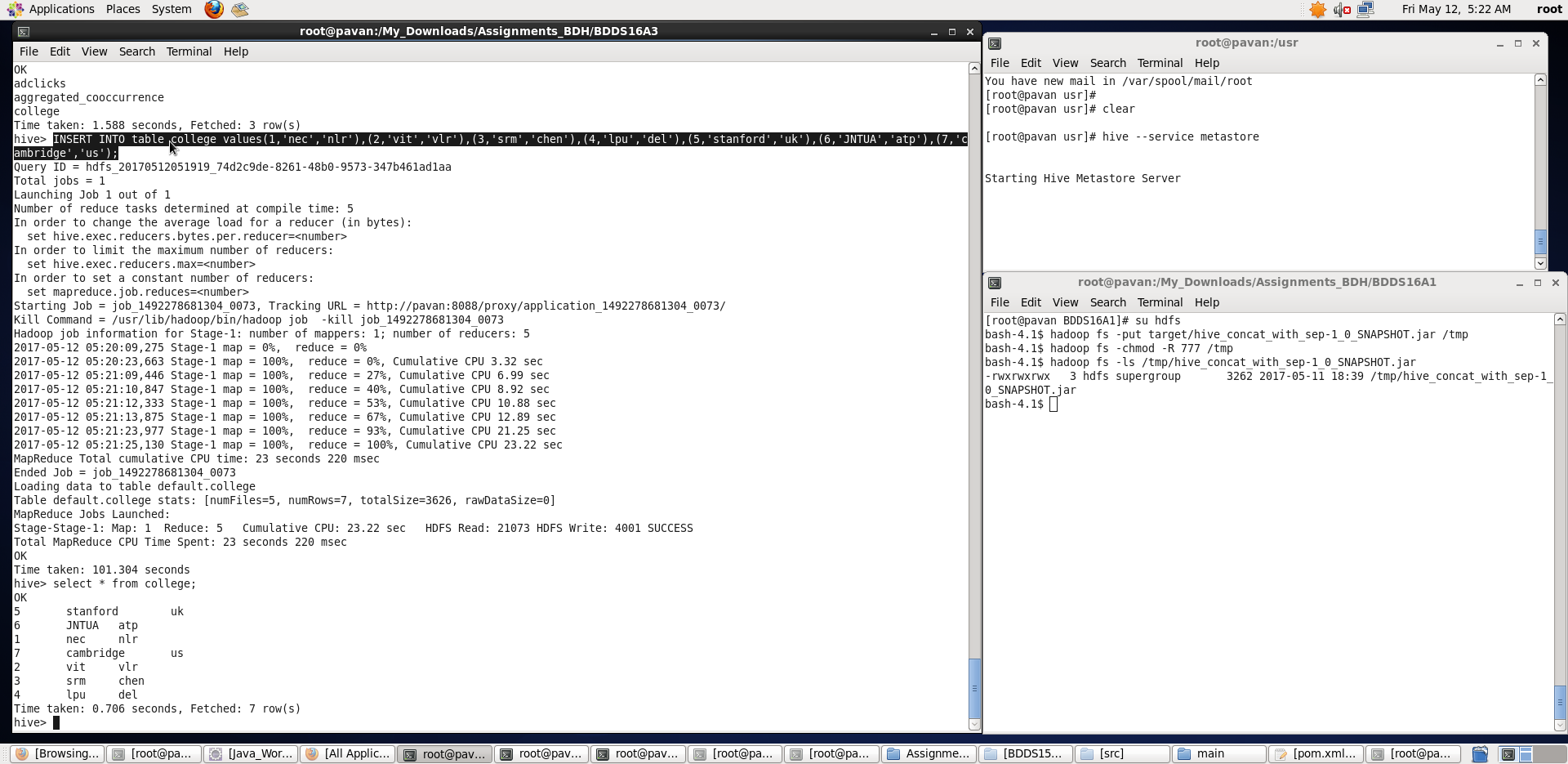
## **Inserting Data into a Hive Table**

hive> INSERT INTO table college values (1,'nec','nlr'),(2,'vit','vlr'),(3,'srm','chen'),(4,'lpu','del'),(5,'stanford','uk'),(6,'JNTUA','atp'),(7,'cambridge','us');

The above command is used to insert row wise data into the Hive table. Here, each row is separated by ‘( )’ brackets.

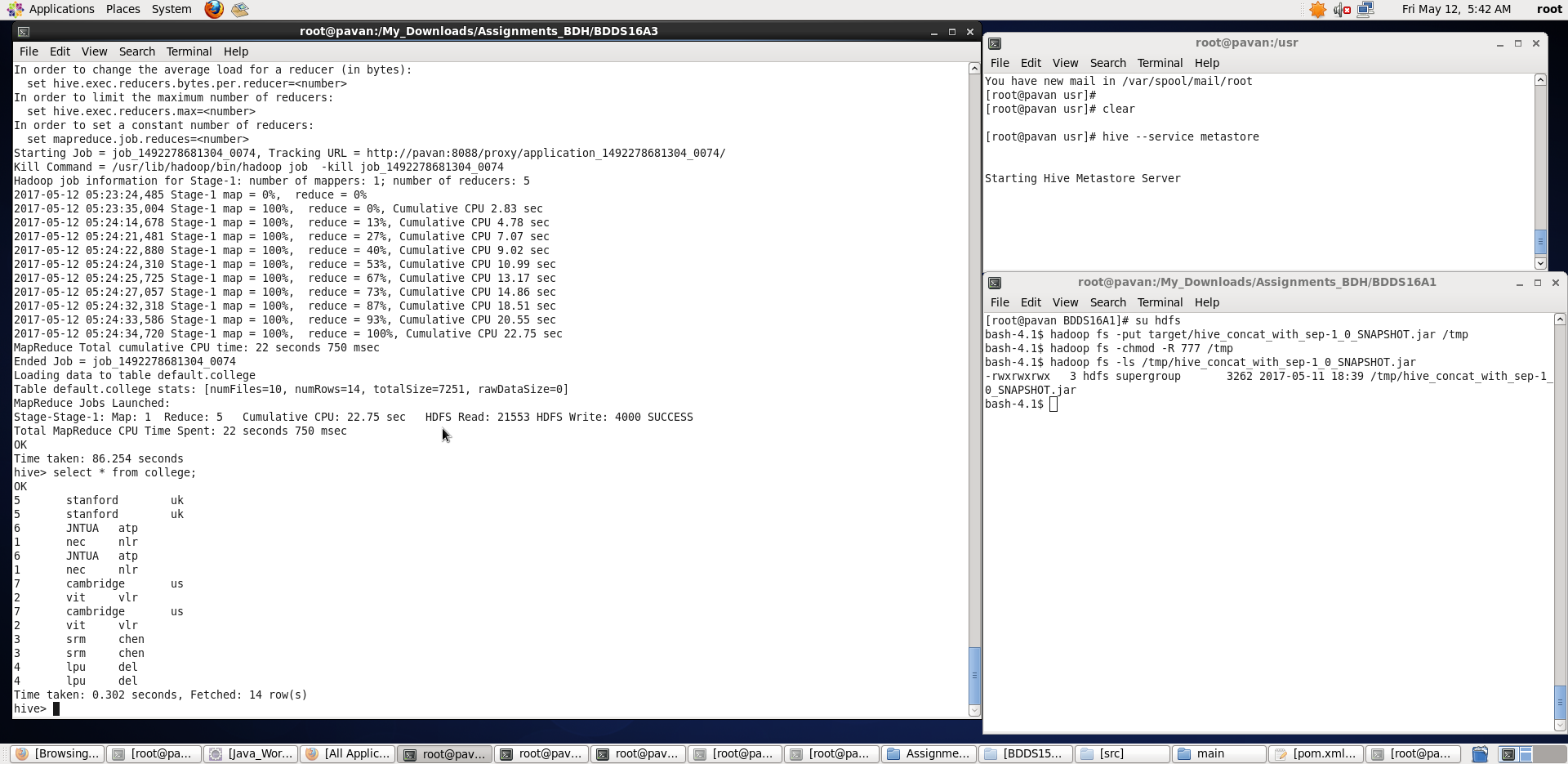
Now, we have successfully inserted the data into the Hive table.

The contents of the table can be viewed using the command **select \* from college**



From the above image, we can see that the data has been inserted successfully into the table.

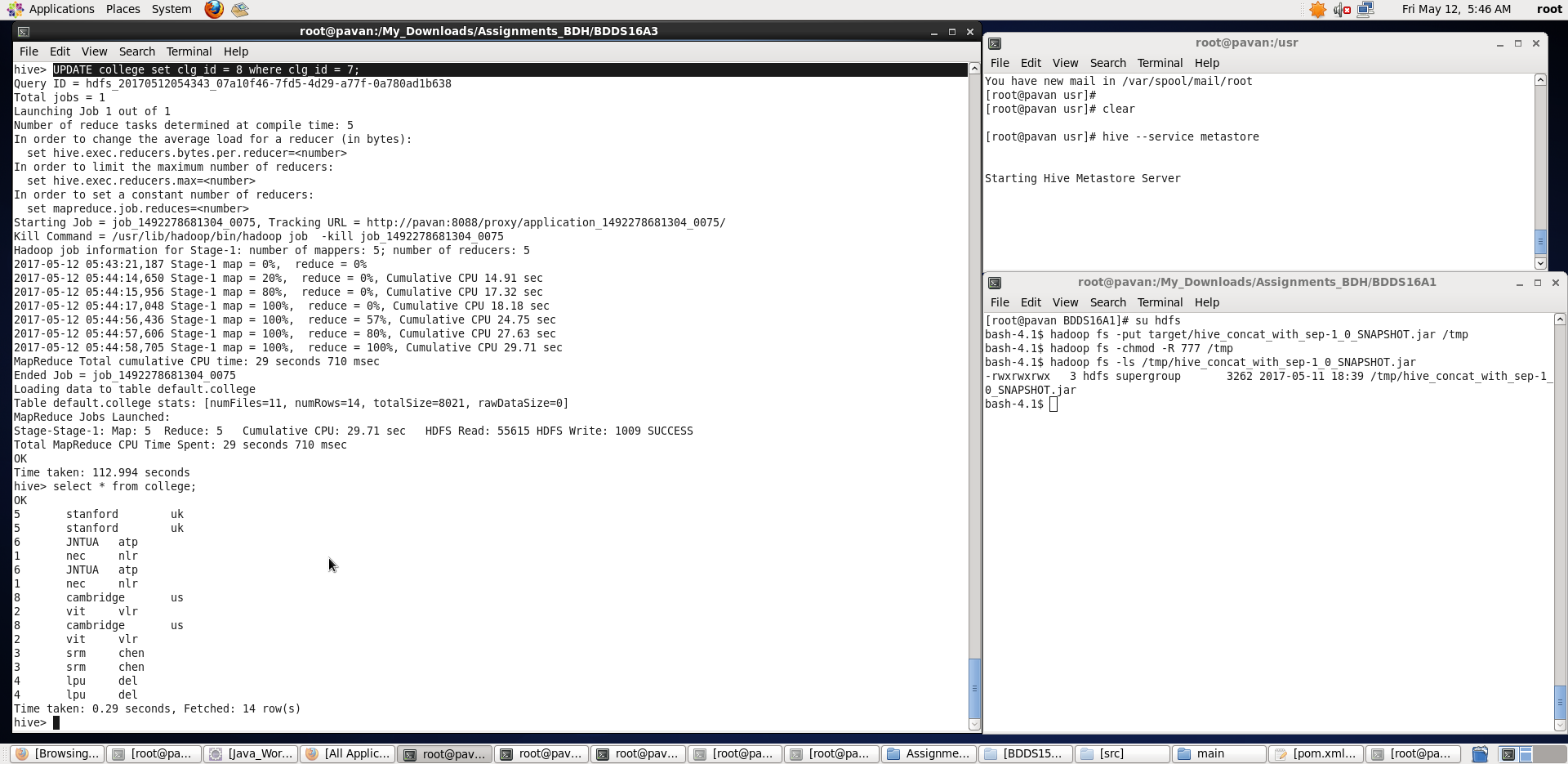
Now if we try to re-insert the same data again, it will be appended to the previous data as shown below:



## **Updating the Data in Hive Table**

hive> UPDATE college set clg\_id = 8 where clg\_id = 7;

The above command is used to update a row in Hive table. We can see that column has been updated.

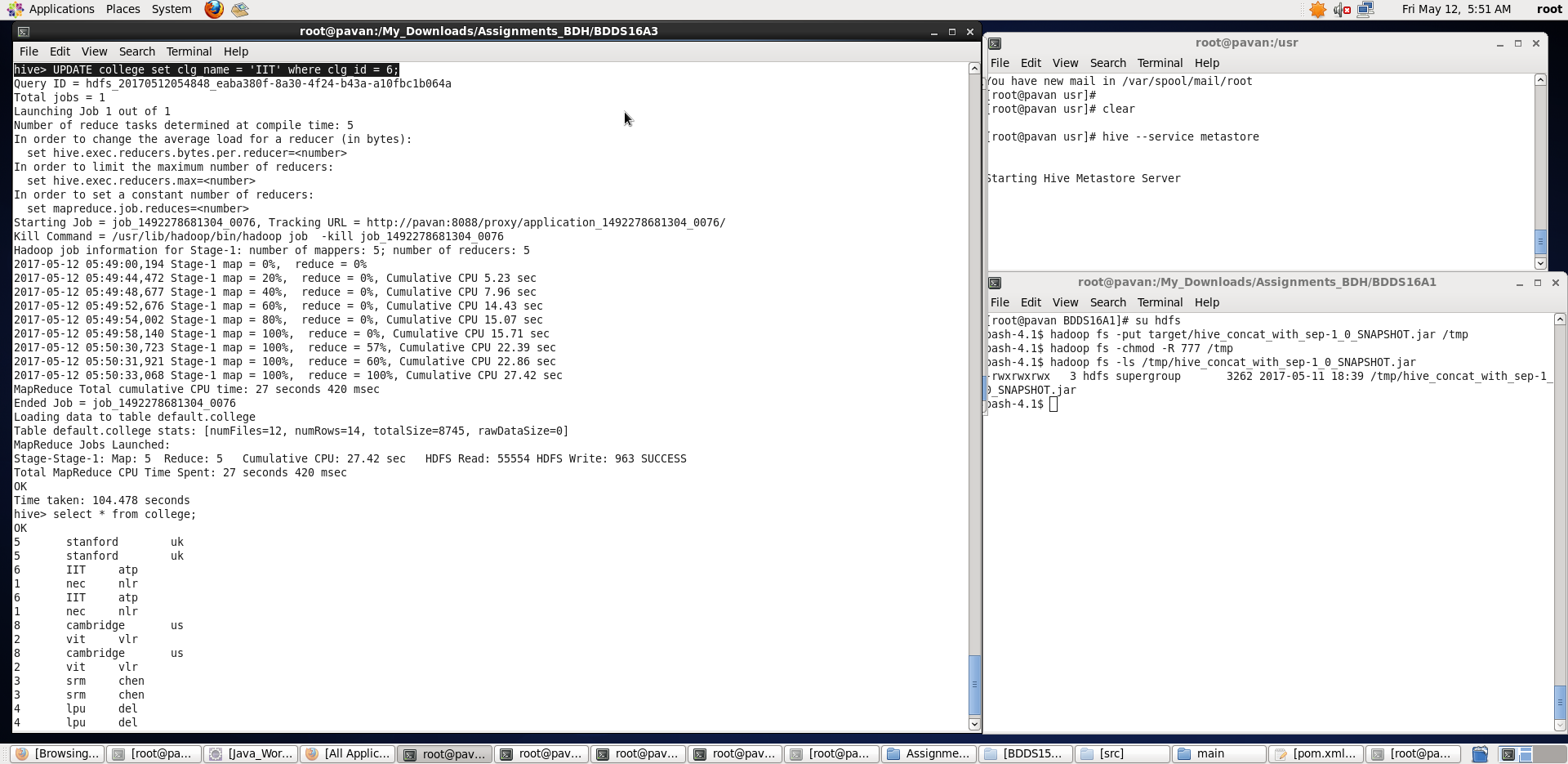


**Now let’s perform the update operation on Non bucketed column:**

hive> UPDATE college set clg\_name = 'IIT' where clg\_id = 6;

We have successfully updated the data.

The updated data can be checked using the command **select \* from college.**



We can see that the data has been updated successfully.

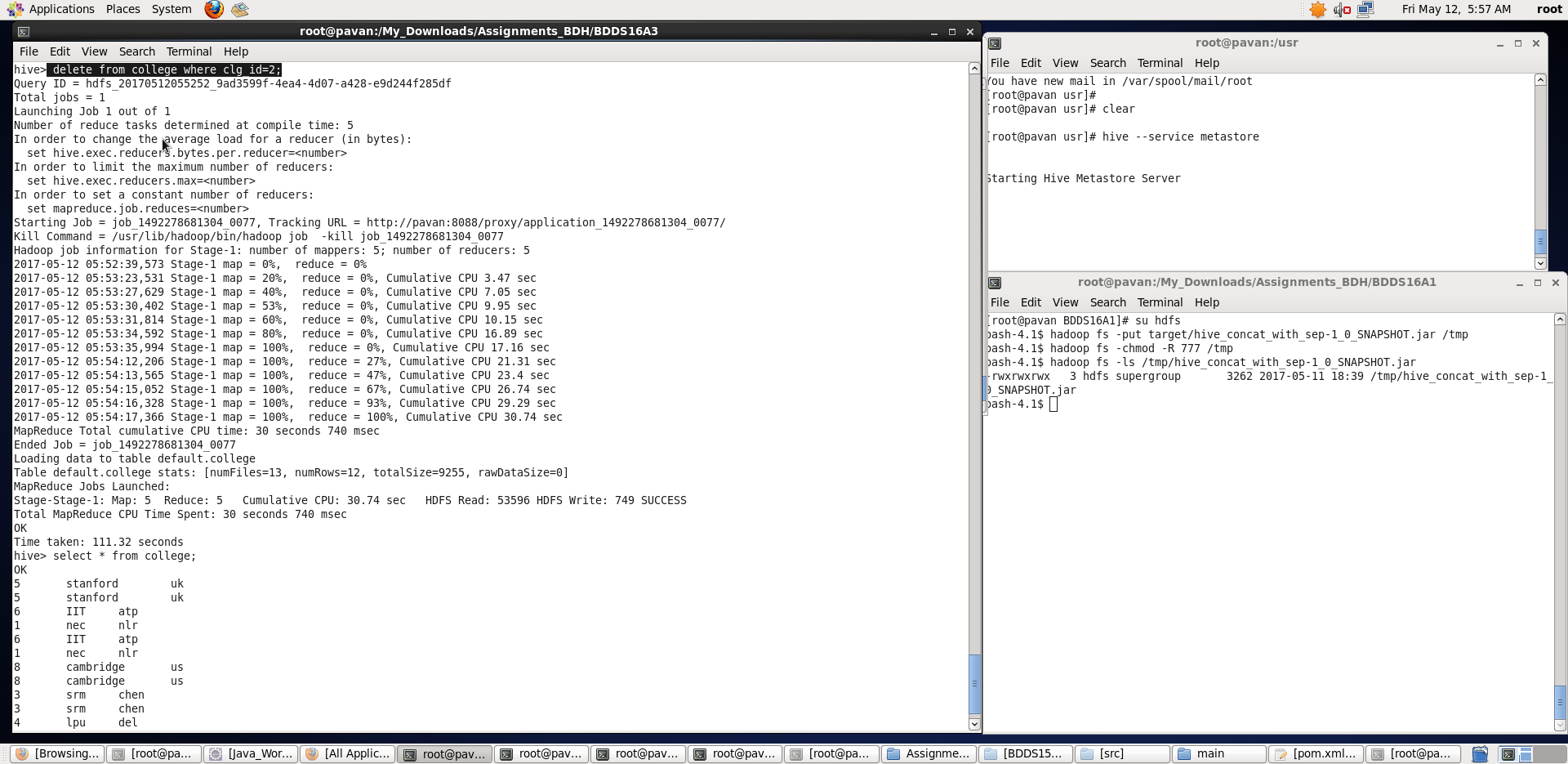
Now let’s perform the Delete operation on the same table.

**Deleting a Row from Hive Table**

hive> delete from college where clg\_id=2;

The above command will delete all the rows with clg\_id = 2 in the Hive table.

We have now successfully deleted a row from the Hive table. This can be checked using the command **select \* from college.**



Thus, we have performed the transactions or row-wise operations in Hive on a sample dataset.