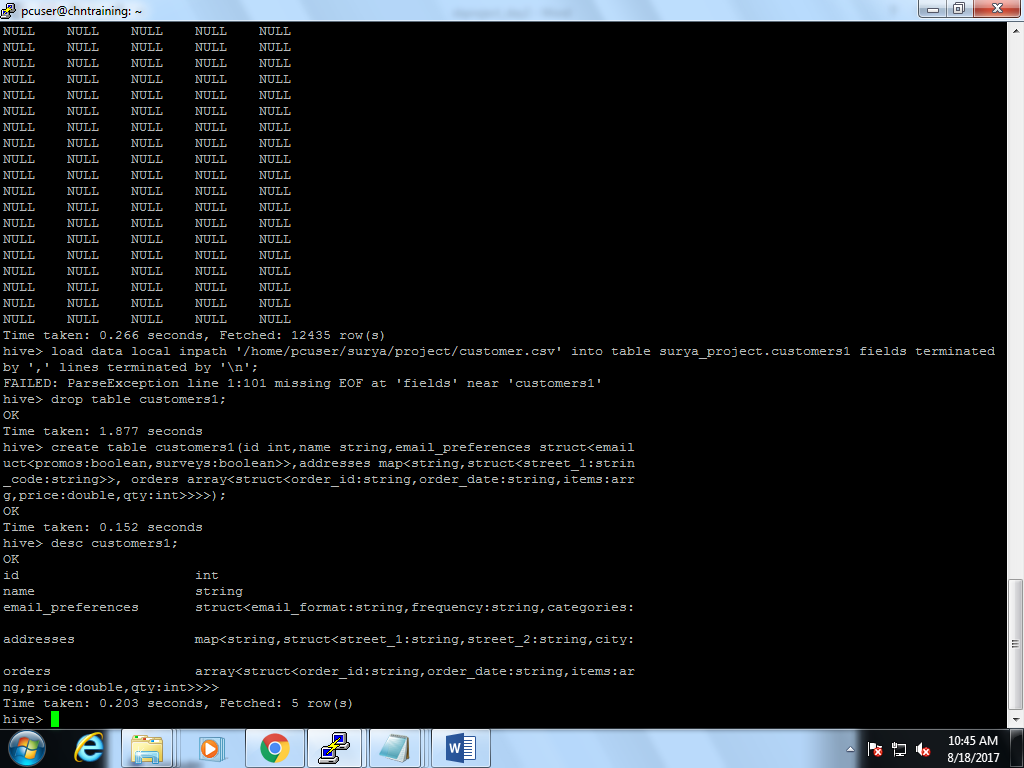
**Creating table in hive using complex data types**

create table customers1(id int,name string,email\_preferences struct<email\_format:string,frequency:string,categories:struct<promos:boolean,surveys:boolean>>,addresses map<string,struct<street\_1:string,street\_2:string,city:string,state:string,zip\_code:string>>,orders array<struct<order\_id:string,order\_date:string,items:array<struct<product\_id:int,sku:string,name:string,price:double,qty:int>>>>);

desc customers1;

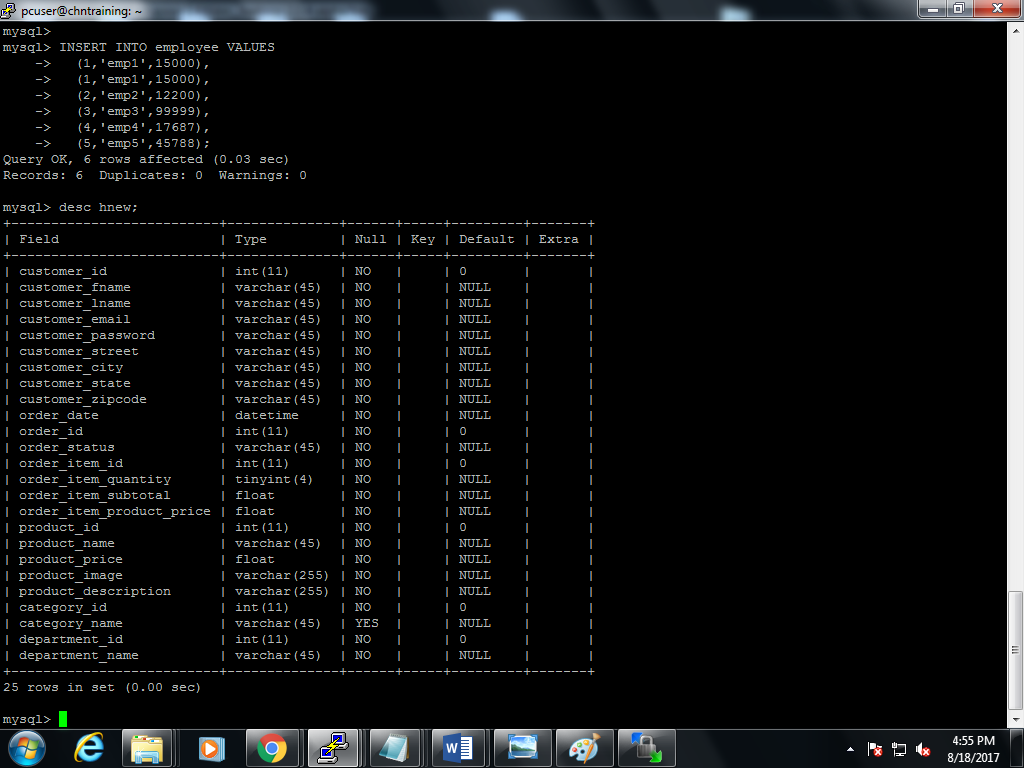
**//** Hive table with complex data type is created.

**Day 2**

**1. Create a denormalized hbase table for retail:**

create table hnew as

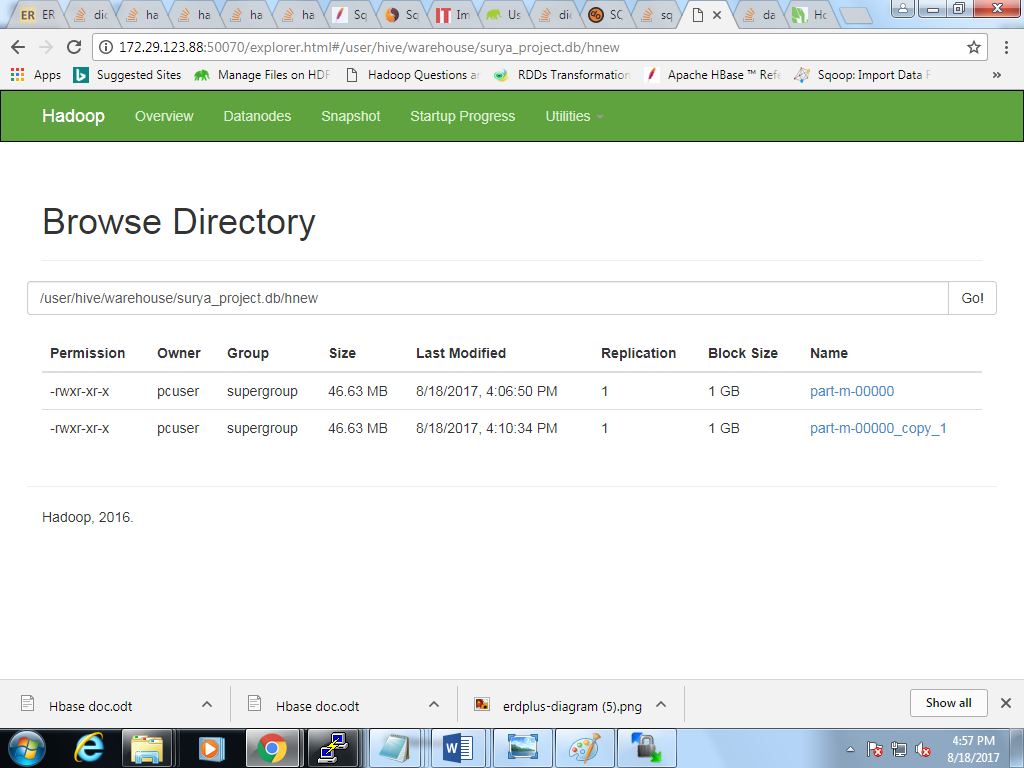
(select c.customer\_id,c.customer\_fname,c.customer\_lname,c.customer\_email,c.customer\_password,c.customer\_street,c.customer\_city,c.customer\_state,c.customer\_zipcode,o.order\_date,o.order\_id,o.order\_status,oi.order\_item\_quantity,oi.order\_item\_id,oi.order\_item\_subtotal,oi.order\_item\_product\_price,p.product\_id,p.product\_name,p.product\_price,p.product\_image,p.product\_description,ca.category\_id,ca.category\_name,d.department\_id,d.department\_name from orders o, products p, order\_items oi,customers c,departments d,category ca where (o.order\_id=oi.order\_item\_order\_id) and (oi.order\_item\_product\_id=p.product\_id) and (c.customer\_id=o.order\_customer\_id) and (p.product\_category\_id=ca.category\_id) and(ca.category\_department\_id=d.department\_id);



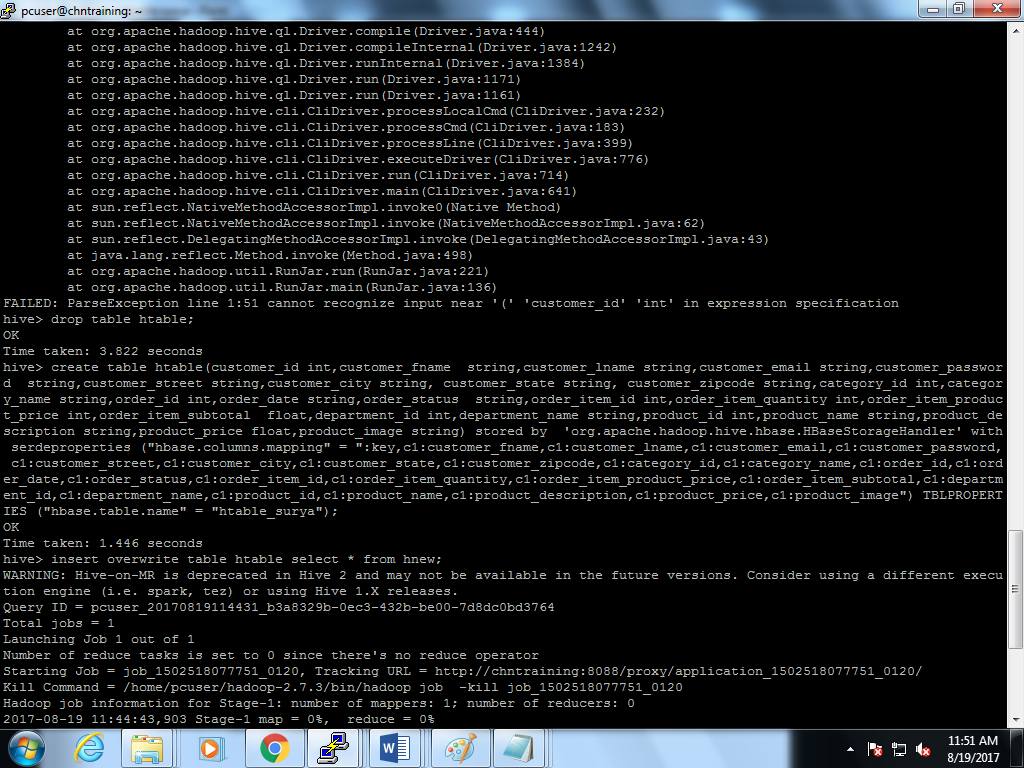
**25 columns.**

**Importing table “hnew” from mysql into hive:-**

Sqoop/bin$:- sqoop import –connect jdbc:mysql://localhost/surya\_project –username root –password root –table hnew –hive-import –hive-table surya\_project.hnew –m 1;

Hdfs :

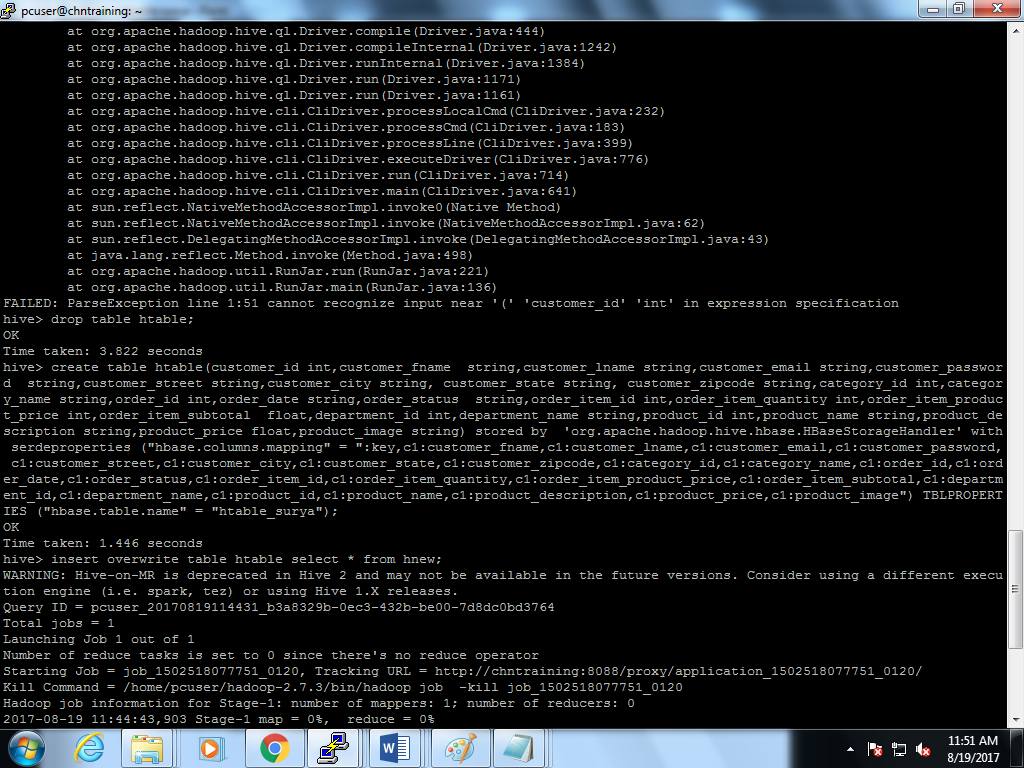
**creating a table in hive for hbase**

create table htable(customer\_id int,customer\_fname string,customer\_lname string,customer\_email string,customer\_password string,customer\_street string,customer\_city string, customer\_state string, customer\_zipcode string,category\_id int,category\_name string,order\_id int,order\_date string,order\_status string,order\_item\_id int,order\_item\_quantity int,order\_item\_product\_price int,order\_item\_subtotal float,department\_id int,department\_name string,product\_id int,product\_name string,product\_description string,product\_price float,product\_image string) stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties ("hbase.columns.mapping" = ":key,c1:customer\_fname,c1:customer\_lname,c1:customer\_email,c1:customer\_password, c1:customer\_street,c1:customer\_city,c1:customer\_state,c1:customer\_zipcode,c1:category\_id,c1:category\_name,c1:order\_id,c1:order\_date,c1:order\_status,c1:order\_item\_id,c1:order\_item\_quantity,c1:order\_item\_product\_price,c1:order\_item\_subtotal,c1:department\_id,c1:department\_name,c1:product\_id,c1:product\_name,c1:product\_description,c1:product\_price,c1:product\_image") TBLPROPERTIES ("hbase.table.name" = "htable\_surya");****

**Integrate hive table with hbase table**

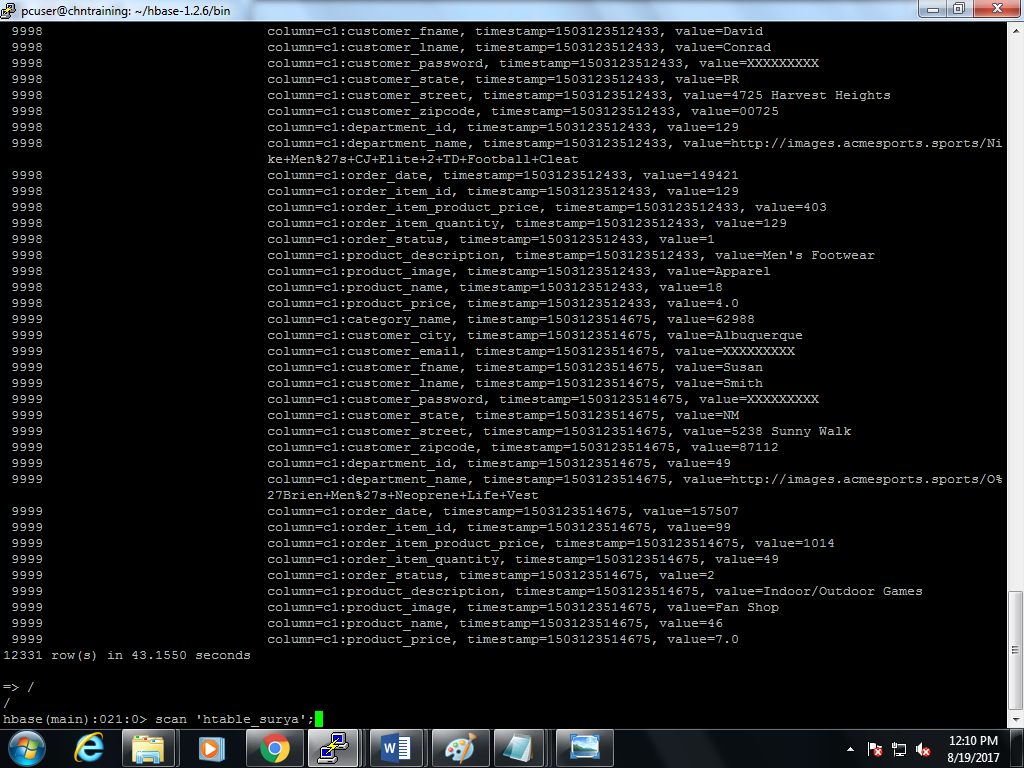
//overwrite into the htable in hive by already using the denormalised table in hive.

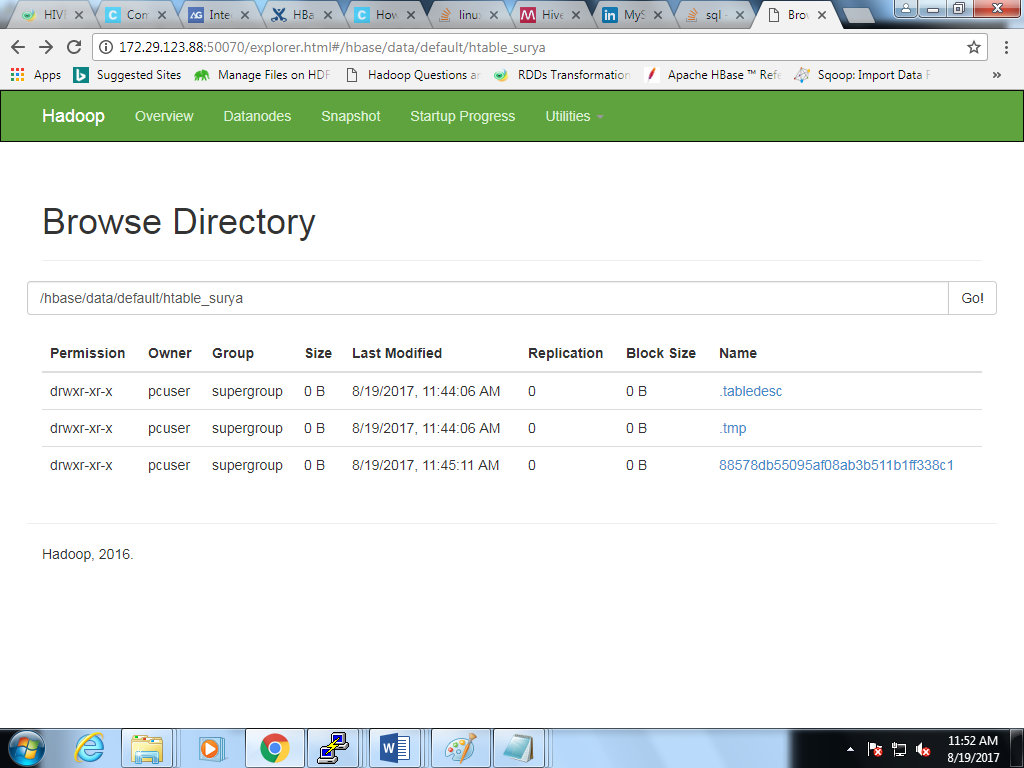
insert overwrite table htable select \* from hnew;

****

**Output in hbase:**

Scan from ‘htable\_surya’

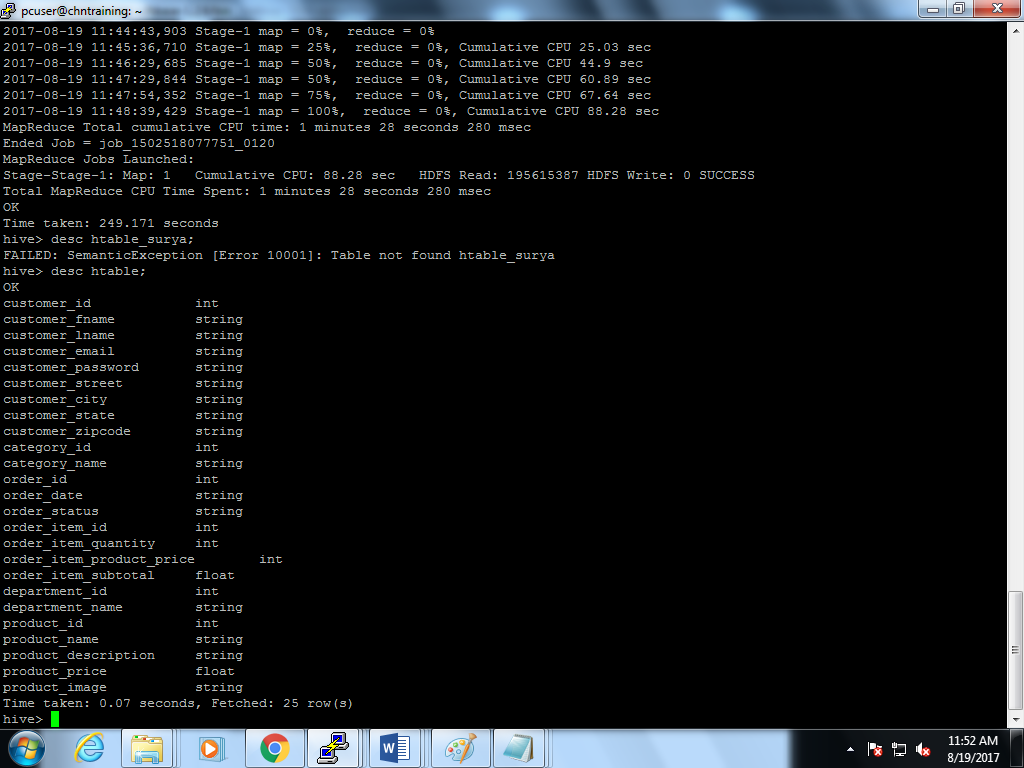
****

The table shown in hdfs 

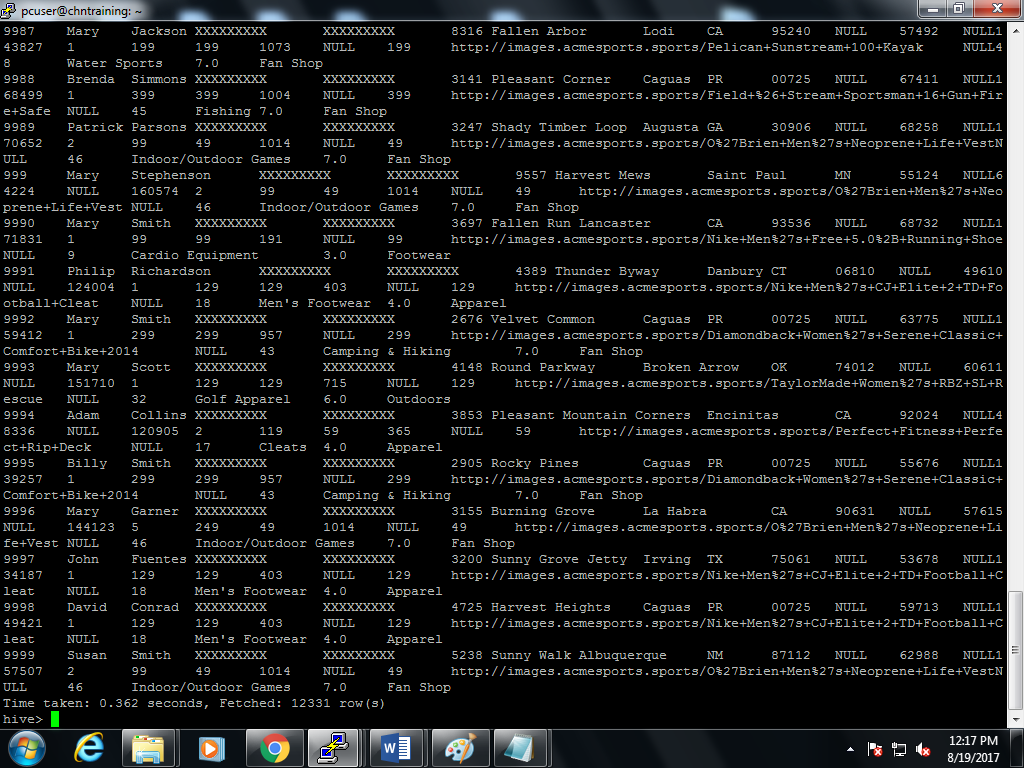
**Mapping of hive and hbase columns Output as seen in hive:**

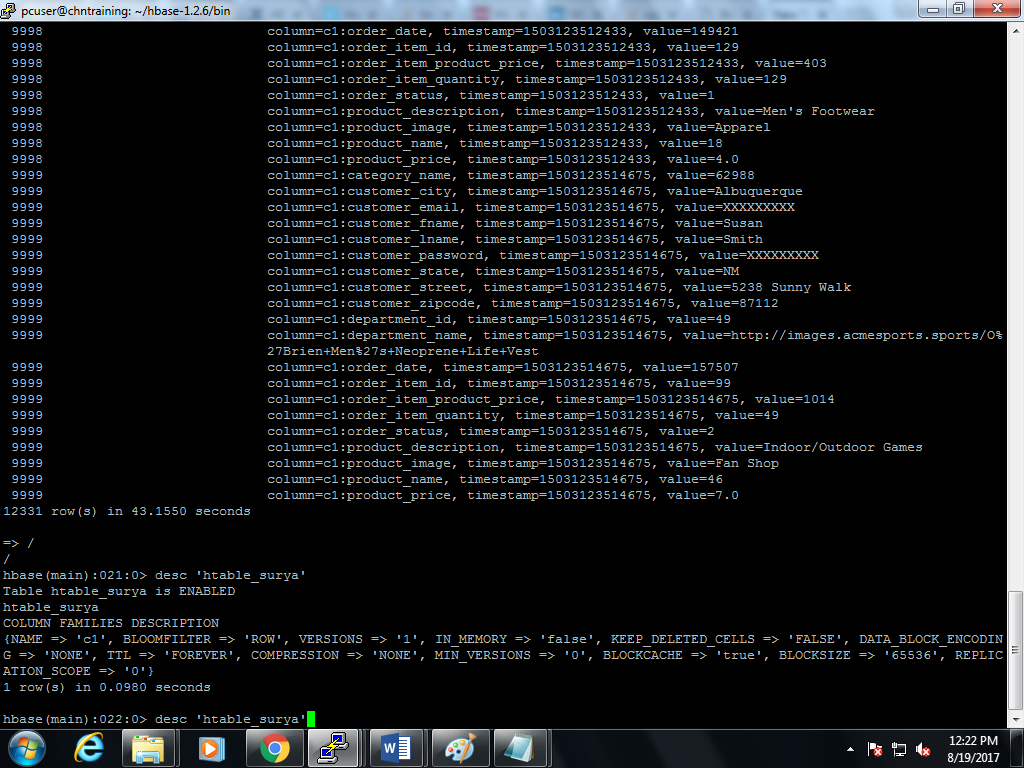
**Output as seen in hive:**

Desc htable;

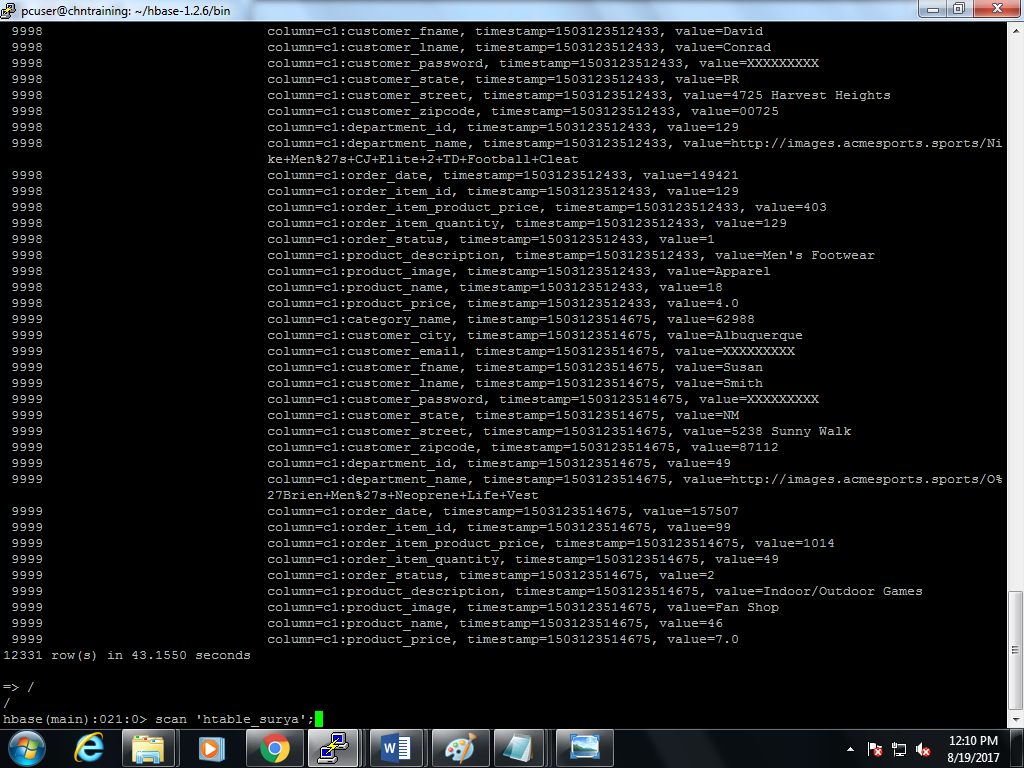


Select \* from htable;



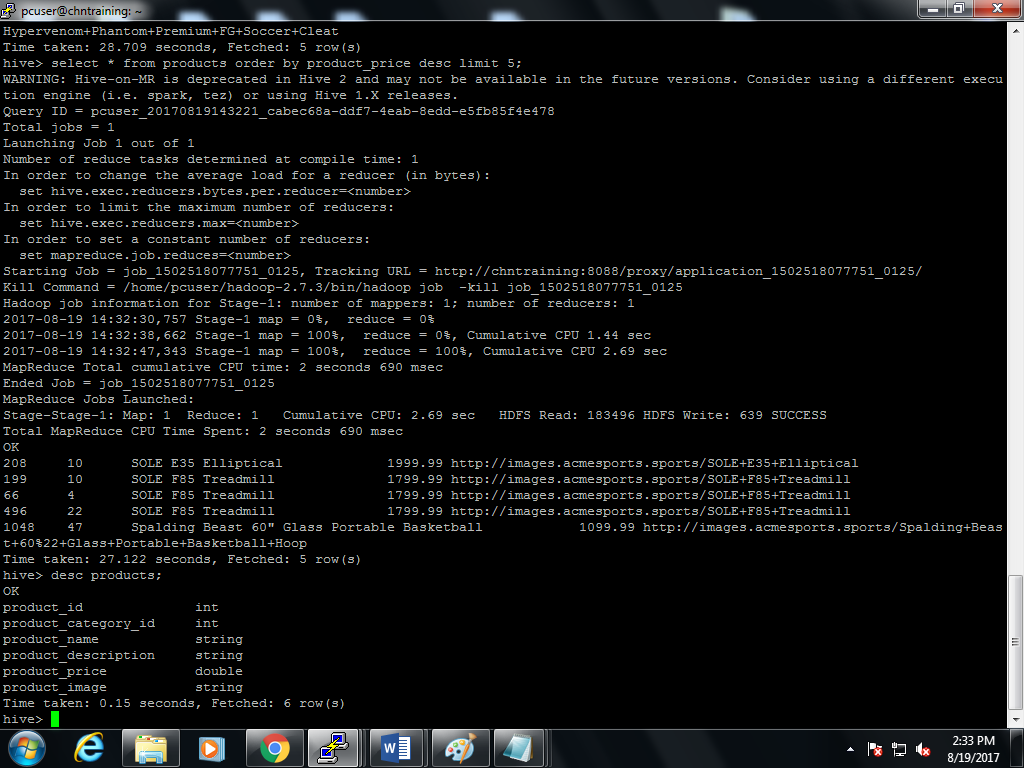
**Output as seen in HBASE:**

Scan 'htable\_surya'



**DML operations**

1) Select \* from products order by product\_price desc limit 5;



//transferring products table from hive to hbase to perform update/delete operations:

//In hive update/delete operations can be achieved by setting ACID property setting transactions=true or from ambari.

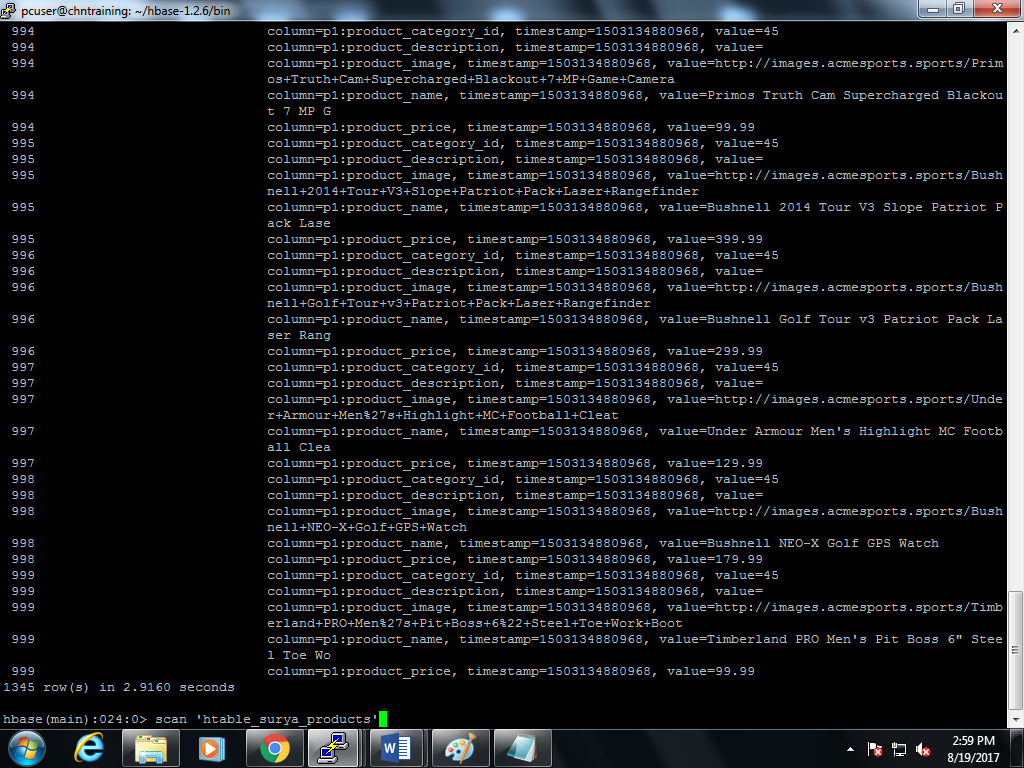
**Creating table in hbase from hive:**

create table hproducts(product\_id int,product\_category\_id int,product\_name string,product\_description string,product\_price double,product\_image string) stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties ("hbase.columns.mapping" = ":key,p1.product\_category\_id,p1.product\_name,p1.product\_description,p1.product\_price,p1.product\_image") TBLPROPERTIES ("hbase.table.name" = "htable\_surya\_products");



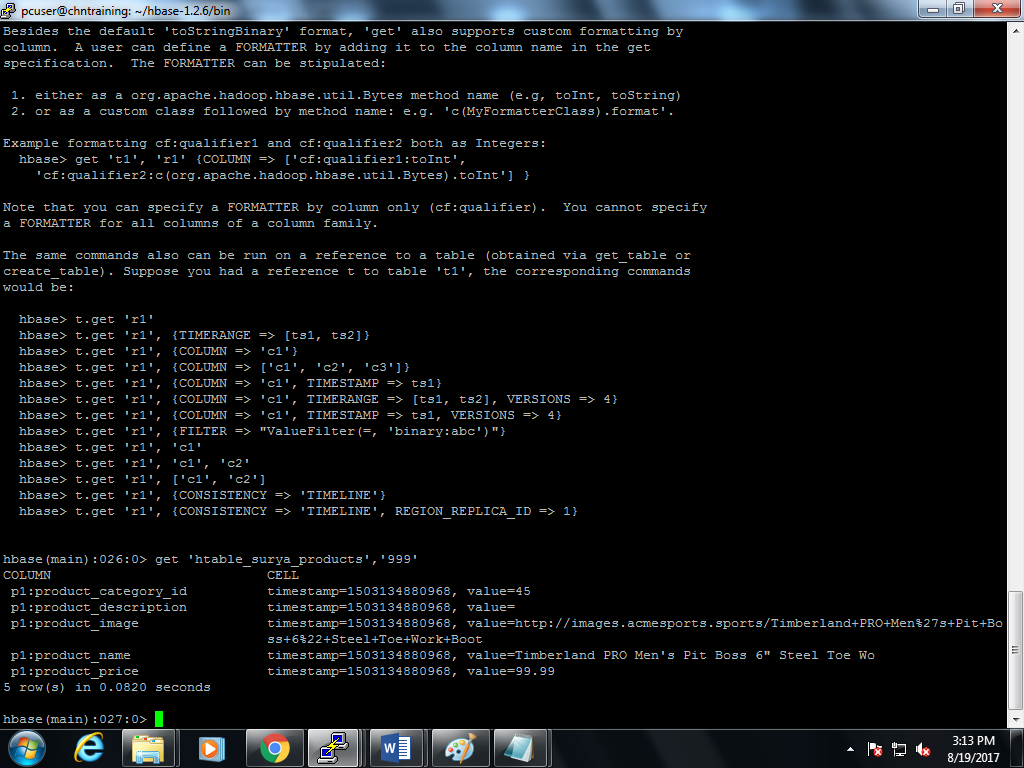
insert overwrite table hproducts select \* from products;

**hbase products table output:**



**Select operations**

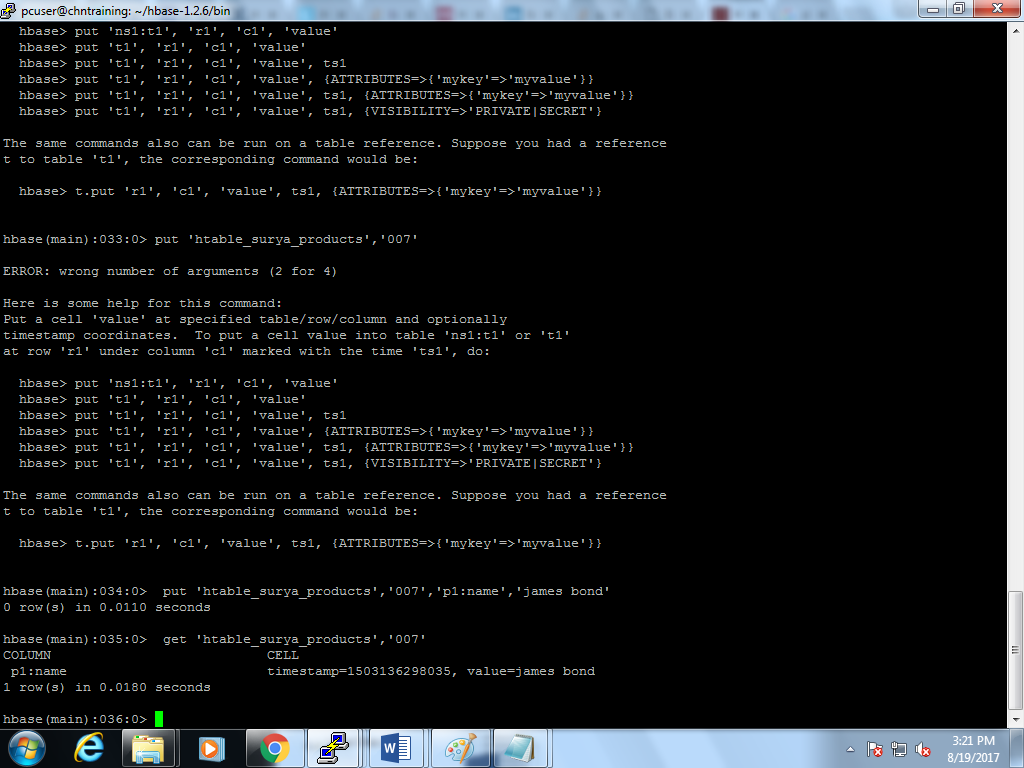
hbase>get 'htable\_surya\_products','999'



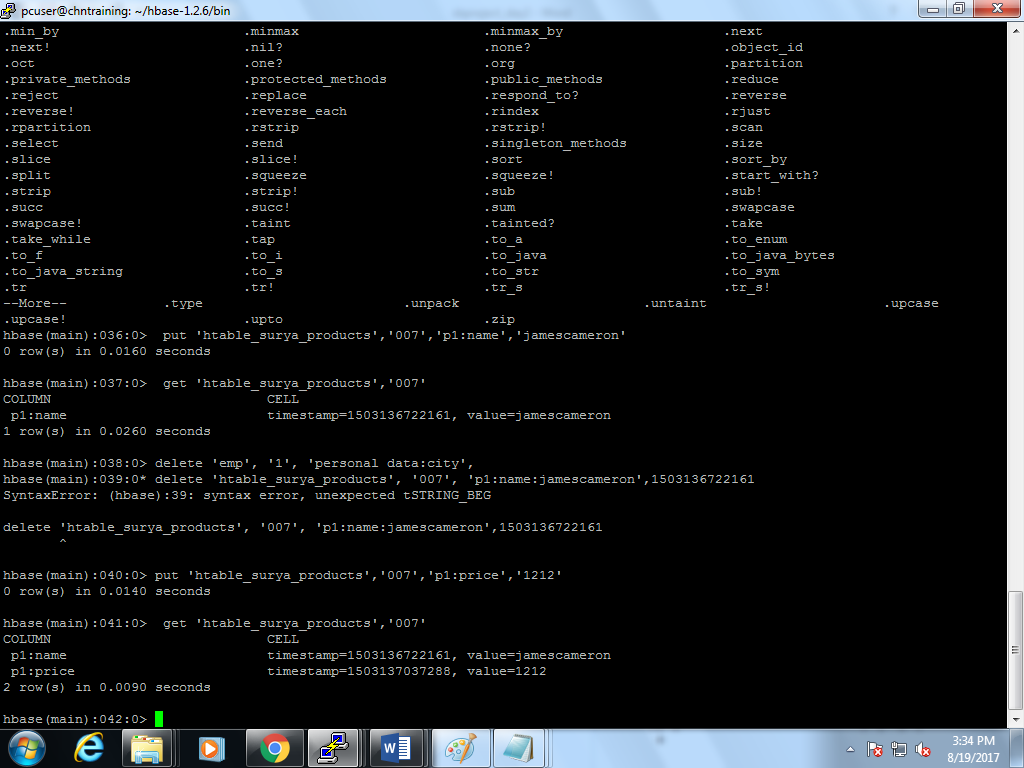
**Insert operations**

hbase>put 'htable\_surya\_products','007','p1:name','james bond'

hbase>get 'htable\_surya\_products','007'

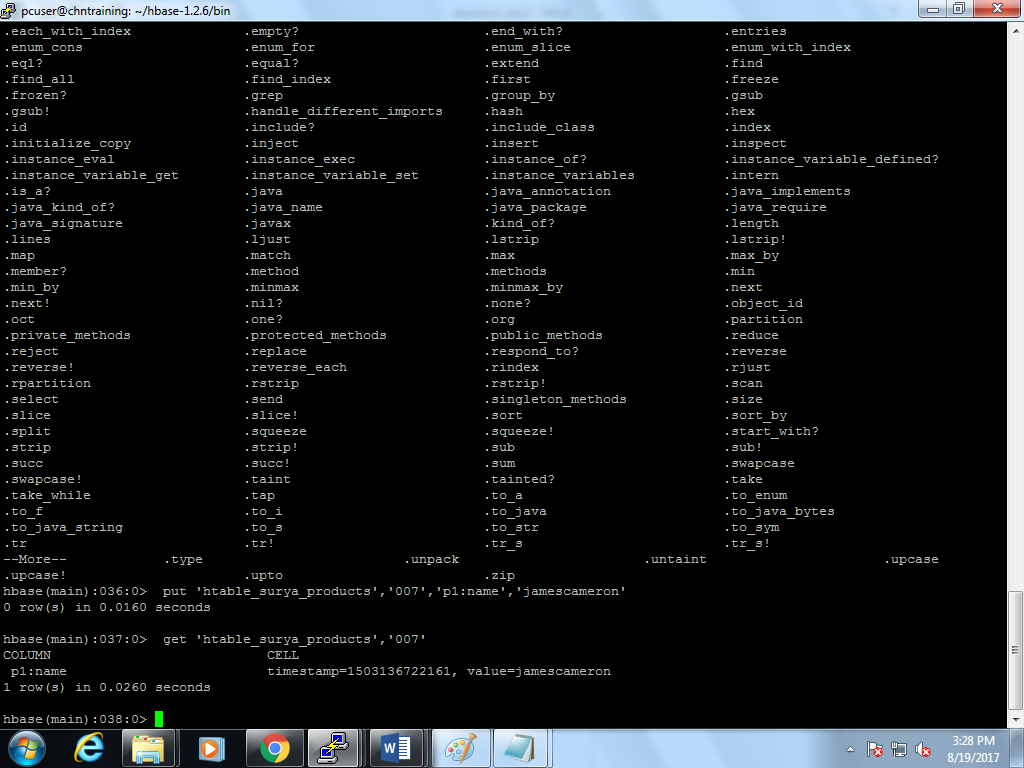


hbase>put 'htable\_surya\_products','007','p1:price','1212'



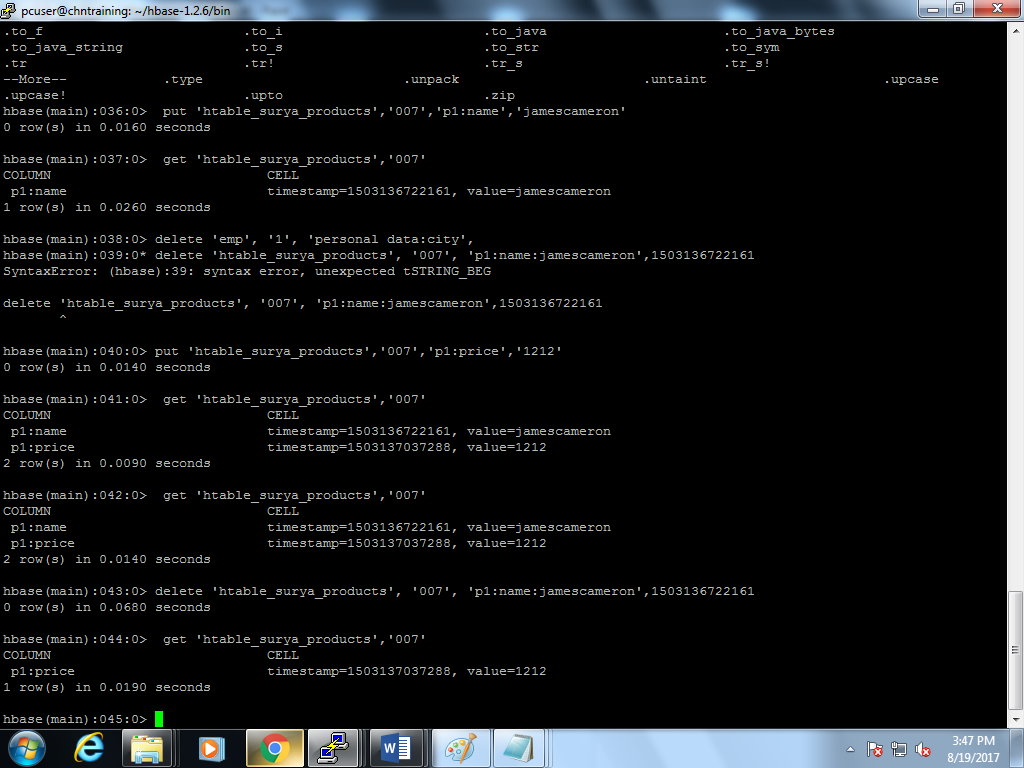
**Updating**

hbase>put 'htable\_surya\_products','007','p1:name','james cameron’



**Deleting**

hbase>Delete'htable\_surya\_products','007','p1:name:jamescameron','1503136722161'



**Access the hive data in spark-sql:**

In hive my database name is surya\_project;

In spark bin:-./spark-sql

show databases;

use surya\_project;

select customer\_id,product\_id from hnew;

