Case Study 2

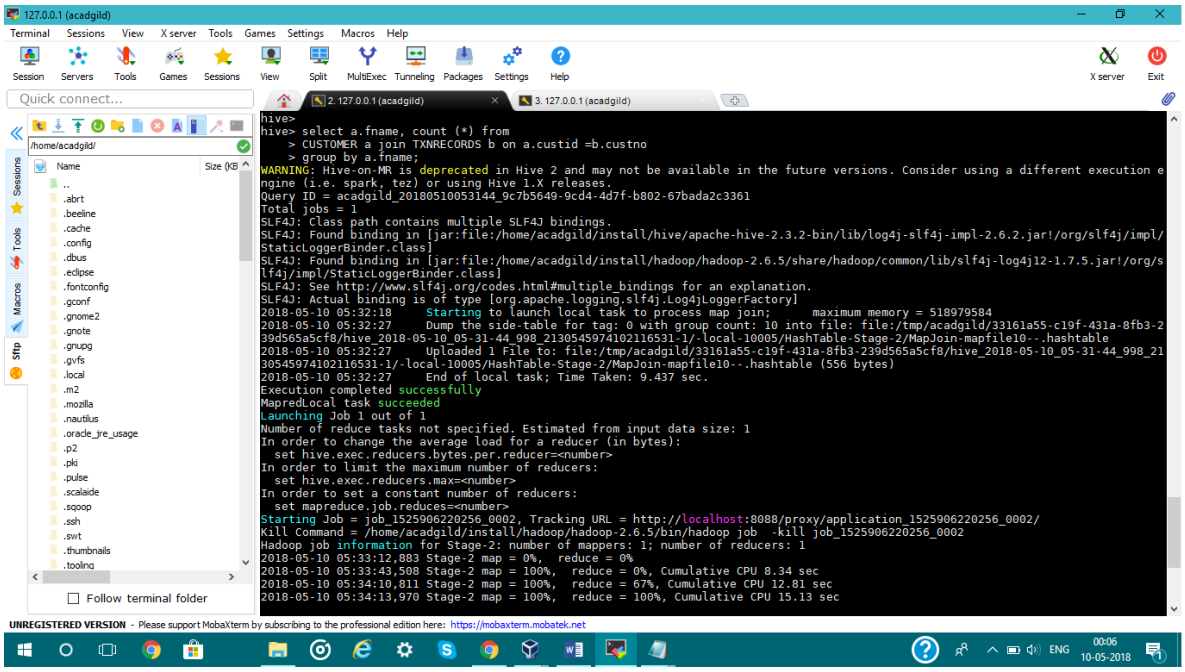
Task1

Find out the number of transaction done by each customer

Command

Select a.fname, count(*) from Customers a join Transactions b on a.custid=b.custno group by a.fname

Screenshot



Output

```
2018-05-10 05:32:27     Uploaded 1 File to: file:/tmp/acadgild/33161a55-c19f-431a-8fb3-239d565a5cf8/hive_2018-05-10_05-31-44_998_21
3054597410211653 1-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile10--.hashtable (556 bytes)
2018-05-10 05:32:27     End of local task; Time Taken: 9.437 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 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_1525906220256_0002, Tracking URL = http://localhost:8088/proxy/application_1525906220256_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525906220256_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-05-10 05:33:12,883 Stage-2 map = 0%,  reduce = 0%
2018-05-10 05:33:43,508 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 8.34 sec
2018-05-10 05:34:10,811 Stage-2 map = 100%,  reduce = 67%, Cumulative CPU 12.81 sec
2018-05-10 05:34:13,970 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 15.13 sec
MapReduce Total cumulative CPU time: 15 seconds 130 msec
Ended Job = job_1525906220256_0002
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 15.13 sec   HDFS Read: 17608 HDFS Write: 301 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 130 msec
OK
Dolores 6
Elsie   6
Gretchen        5
Hazel   10
Karen   5
Kristina        8
Malcolm 6
Paige   6
Patrick 5
Sherri  3
Time taken: 150.644 seconds, Fetched: 10 row(s)
hive>
```

Task2

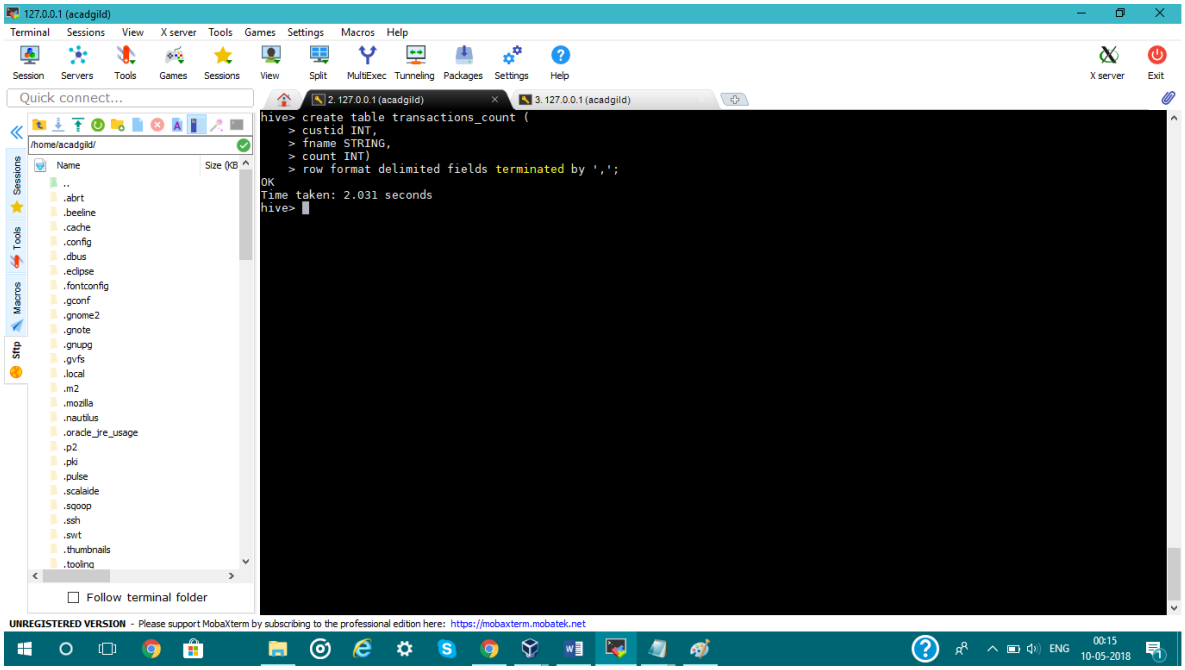
Create a new table called TRANSACTIONS_COUNT. This table should have 3 fields - custid, fname and count.

Command

create table transactions_count (custid INT, fname STRING, count INT)

row format delimited fields terminated by ‘,’;

Screenshot



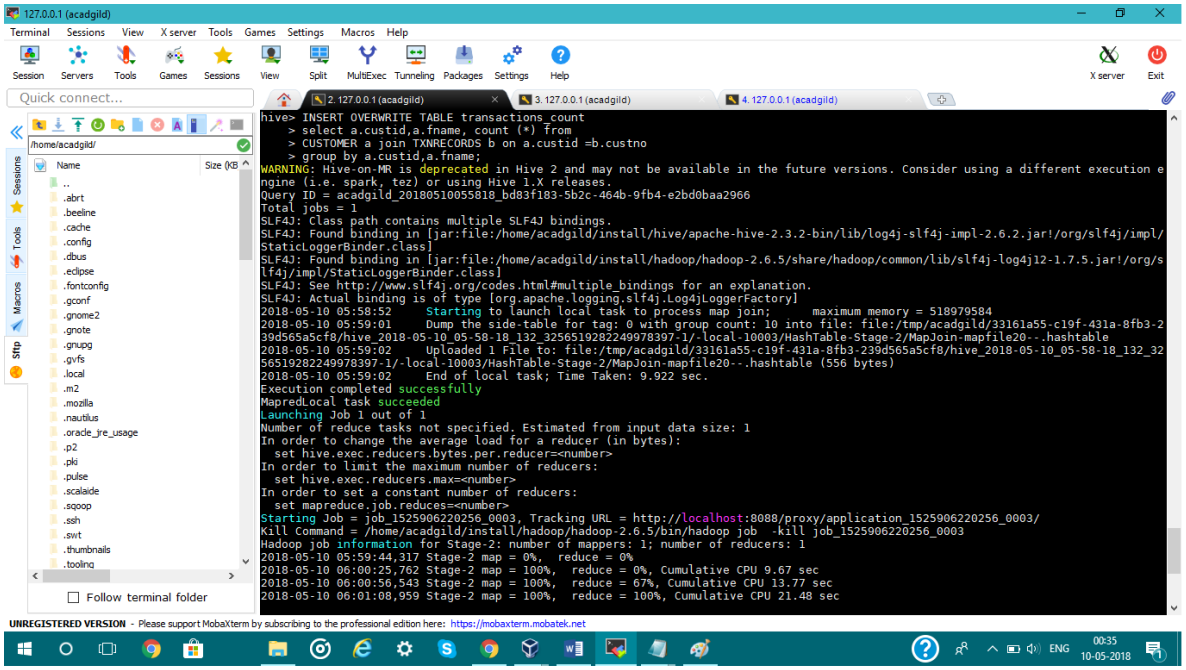
Task3

Now write a hive query in such a way that the query populates the data obtained in Step 1 above and populate the table in step 2 above.

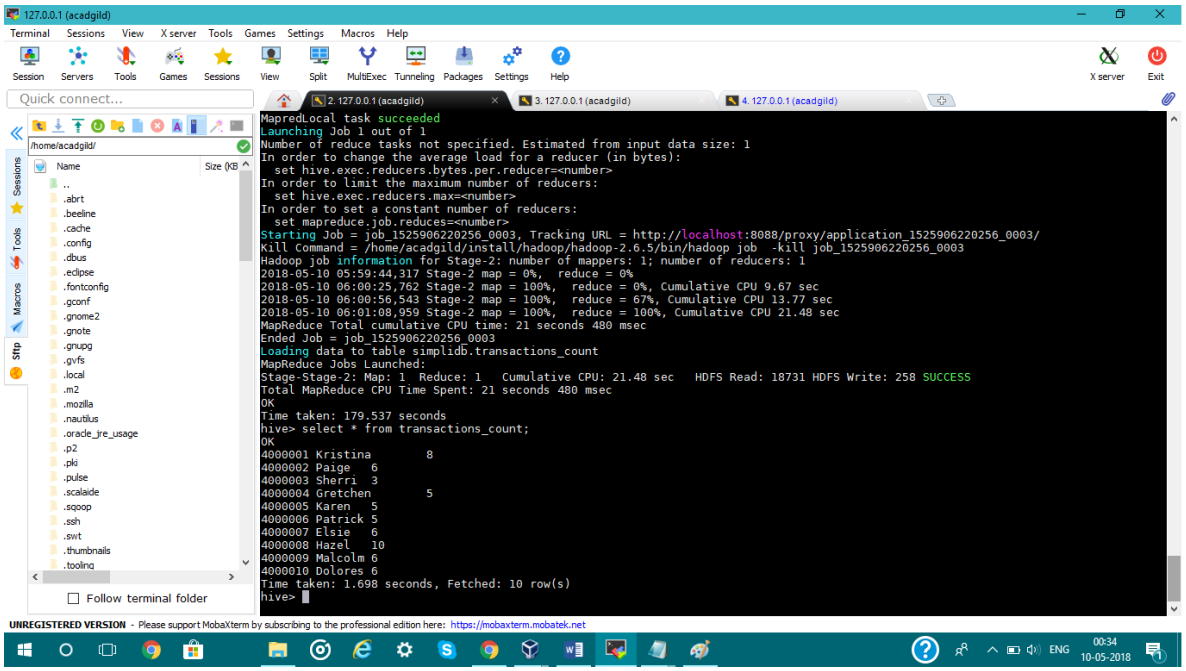
Command

Insert overwrite table transactions_count Select a.custid, a.fname, count(*) from Customers a join Transactions b on a.custid=b.custno group by a.custid, a.fname

Screenshot



Output



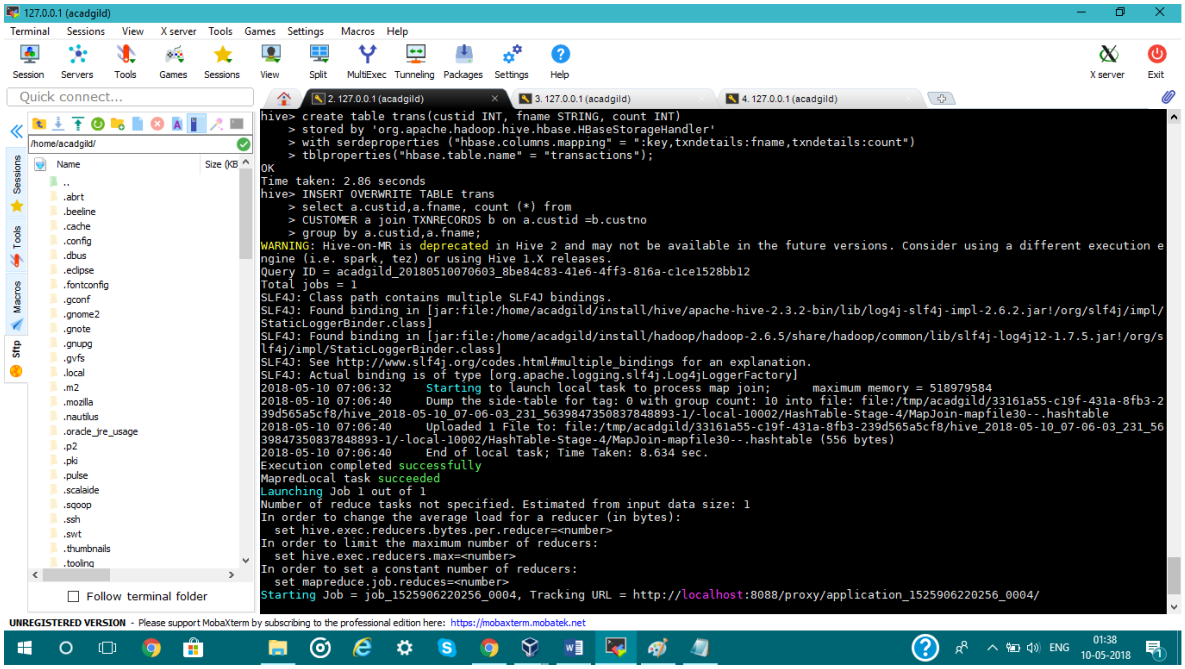
Task4

Make the TRANSACTIONS_COUNT table Hbase complaint

Command

Create table trans (custid INT, fname STRING, count INT) stored by ‘org.apache.hadoop.hive.hbase.HBasestorageHandler’ with serdeproperties (‘hbase.columns.mapping”=:”key.txndetails:fname, txndetails:count”) tblproperties(“hbase.table.name”=”transactions”);

Screenshot



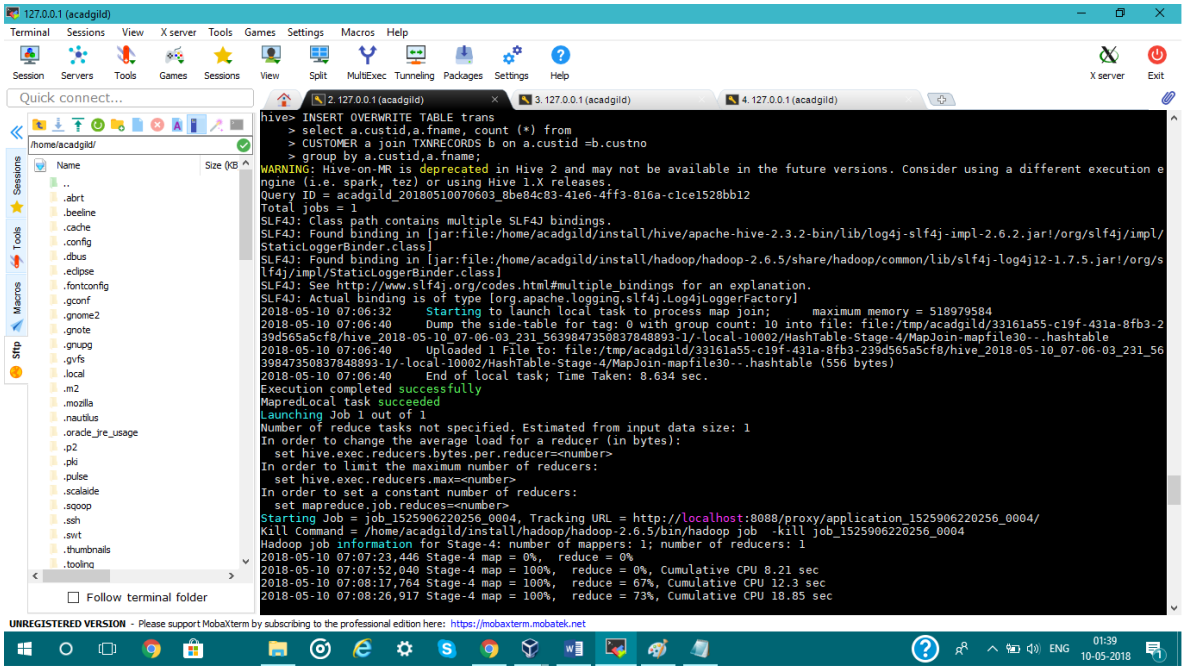
Task5

Now insert the data in TRANSACTIONS_COUNT table using the query in step 3 again, this should populate the Hbase TRANSACTIONS table automatically

Command

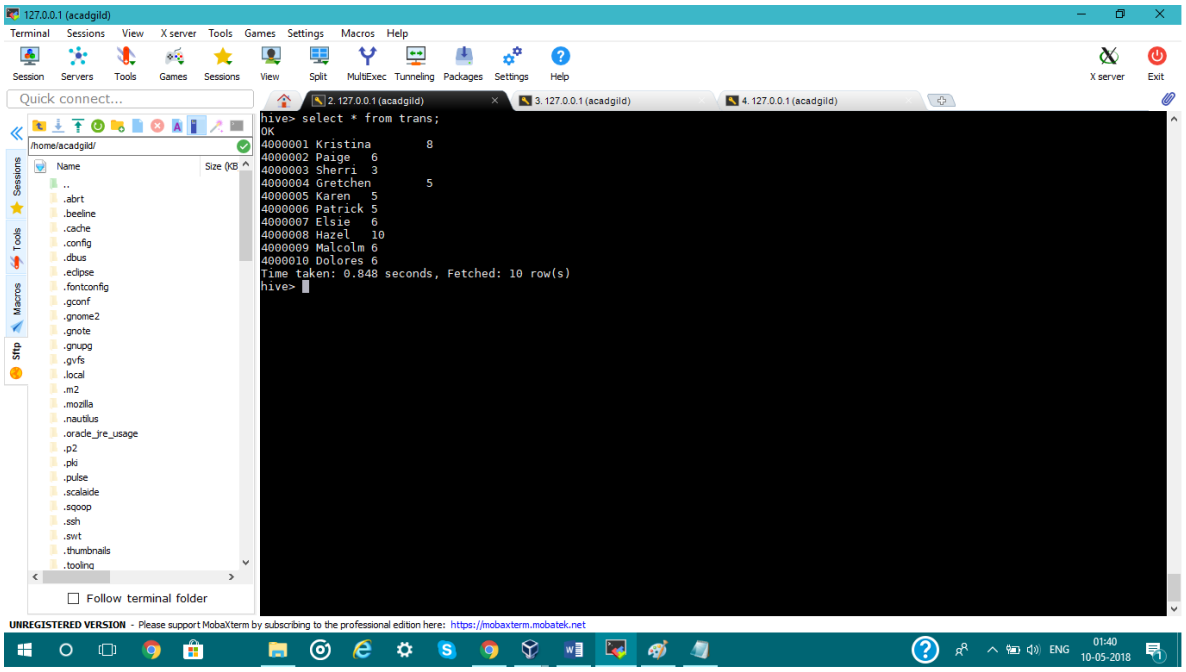
Insert overwrite table trans Select a.custid, a.fname, count(*) from Customers a join Transactions b on a.custid=b.custno group by a.custid, a.fname

Screenshot



```
hive> INSERT OVERWRITE TABLE trans
    > select a.custid,a.fname, count (*) from
    > CUSTOMER a join TXNRECORDS b on a.custid =b.custno
    > group by a.custid,a.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180510070603_8be84c83-41e6-4ff3-816a-c1ce1528bb12
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/s
lf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2018-05-10 07:06:32     Starting to launch local task to process map join;      maximum memory = 518979584
2018-05-10 07:06:40     Dump the side-table for tag: 0 with group count: 10 into file: file:/tmp/acadgild/33161a55-c19f-431a-8fb3-2
39d565a5cf8/hive_2018-05-10_07-06-03_231_5639847350837848893-1/-local-10002/HashTable-Stage-4/MapJoin-mapfile30--.hashtable
2018-05-10 07:06:40     Uploaded 1 File to: file:/tmp/acadgild/33161a55-c19f-431a-8fb3-239d565a5cf8/hive_2018-05-10_07-06-03_231_56
39847350837848893-1/-local-10002/HashTable-Stage-4/MapJoin-mapfile30--.hashtable (556 bytes)
2018-05-10 07:06:40     End of local task; Time Taken: 8.634 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 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_1525906220256_0004, Tracking URL = http://localhost:8088/proxy/application_1525906220256_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525906220256_0004
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
2018-05-10 07:07:23,446 Stage-4 map = 0%,  reduce = 0%
2018-05-10 07:07:52,040 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 8.21 sec
2018-05-10 07:08:17,764 Stage-4 map = 100%,  reduce = 67%, Cumulative CPU 12.3 sec
2018-05-10 07:08:26,917 Stage-4 map = 100%,  reduce = 73%, Cumulative CPU 18.85 sec
```

Output



```
hive> select * from trans;
OK
4000001 Kristina        8
4000002 Paige   6
4000003 Sherri  3
4000004 Gretchen        5
4000005 Karen   5
4000006 Patrick 5
4000007 Elsie   6
4000008 Hazel   10
4000009 Malcolm 6
4000010 Dolores 6
Time taken: 0.848 seconds, Fetched: 10 row(s)
hive>
```

Task6

Now from the Hbase level, write the Hbase java API code to access and scan the TRANSACTIONS table data from java level

Code

```
//importing all the packages
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.ZooKeeperConnectionException;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;

public class HbaseTable {                          //defining a class
 private static Configuration conf = null;         //declaring configurations
static {
      conf = HBaseConfiguration.create();          //initialising configuration
   }
```

```
//scanning a HBase table
   public static void getAllRecord (String tableName) {
      try{
          HTable table = new HTable(conf, tableName);
          Scan s = new Scan();
          ResultScanner ss = table.getScanner(s);
          for(Result r:ss){
             for(KeyValue kv : r.raw()){
                System.out.print(new String(kv.getRow()) + " ");       //for fetching
rowkey
                System.out.print(new String(kv.getFamily()) + ":");    //for fetching
column family
                System.out.print(new String(kv.getQualifier()) + " ");  //for fetching
columns
                System.out.print(kv.getTimestamp() + " ");
                System.out.println(new String(kv.getValue()));
             }
          }
      } catch (IOException e){
         e.printStackTrace();
      }
   }

   public static void main(String[] agrs) {
      try {
         String tablename = "transactions";      //initialising the name of the table
in HBase
```

```
        String[] families = { "custid", "fname","count" };     //initialising the column families
        System.out.println("===========show all record========");
        HbaseTable.getAllRecord(tablename);
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Screenshot