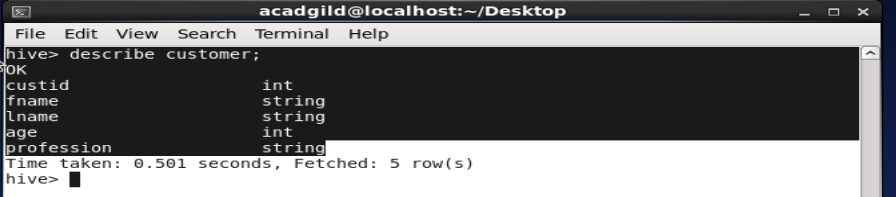
CASE STUDY 2 **CUSTOMER\_TRANSACTION**

Mainly there are two table in this case study in acadgilddb database

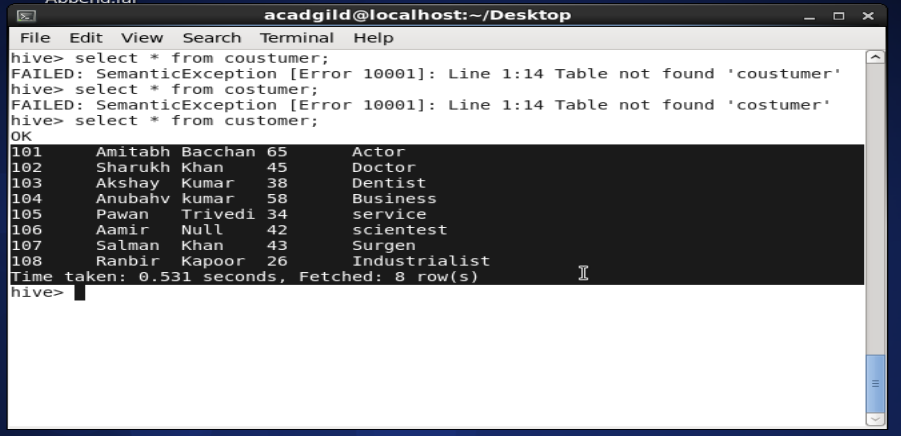
1-customer 2- transaction

**customer** table have five columns consist of **custid, fname, lname, age** and **profession**.

you can find customer schema by typing: **describe customer** as shown below:

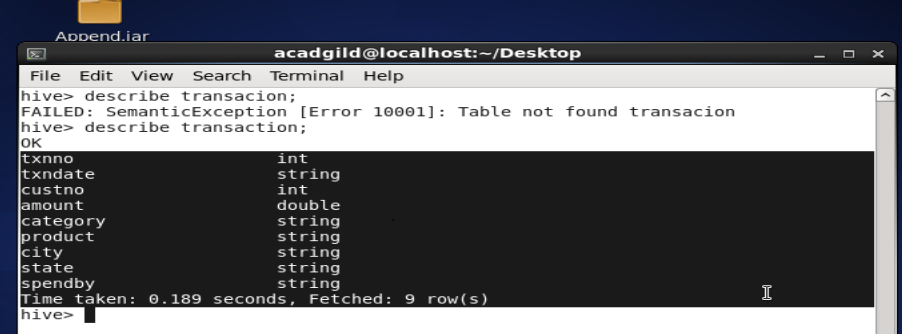


Data in this table is given as shown in screenshot

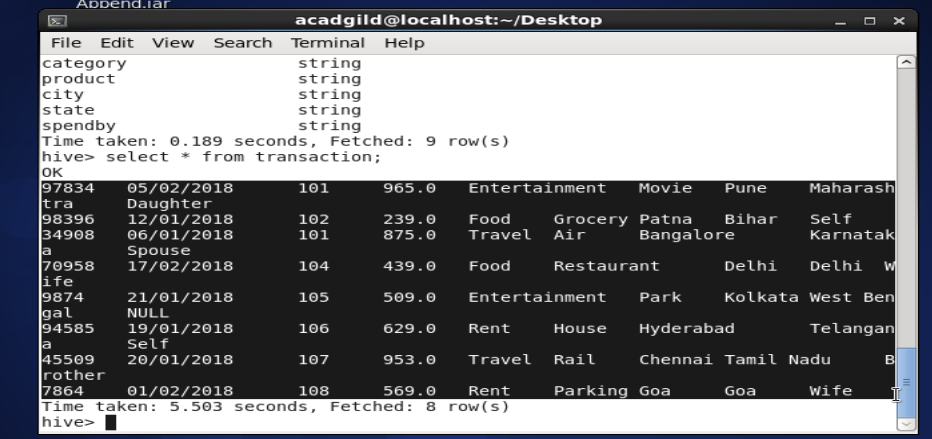


**transaction** table have nine columns consist of transaction **number, transaction date, customer ID, amount,category,product detail,city,state,spendby details**.

you will find this detail about table by: **“describe transaction”** as shown below.



The data in this table is given in the screen shot



The database is created using **create database acadgilddb** the tables are created using **create table costumer(names of column datatype,…..)row format delimited field terminated by ‘,’;**

**create table transaction(names of column datatype,…..)row format delimited field terminated by ‘,’;**

the data is loaded in these tables by using

**load data local inpath ‘<location of file>’ into <tablename>;**

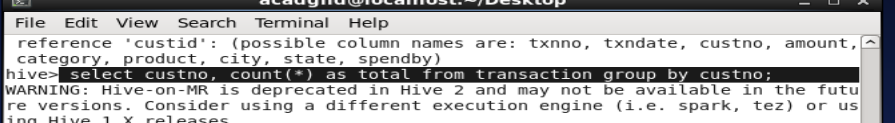
**now lets do the objectives**

**objective1 :-**

**1. Find out the number of transaction done by each customer.**

We can achieve this task by using two queries

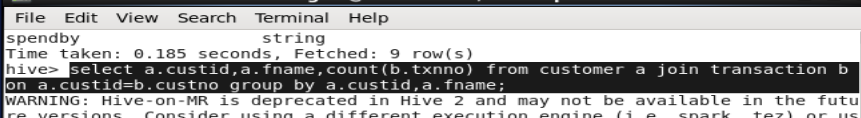
* **SELECT** custno, **COUNT**(\*) as total **FROM** transactions **GROUP BY** custno;



Result:-



* **SELECT** a.custid , a.fname , b.txnno **FROM** customer a **JOIN** transactions b ON c.custid = t.custno **GROUP BY** custid,fname



Result:



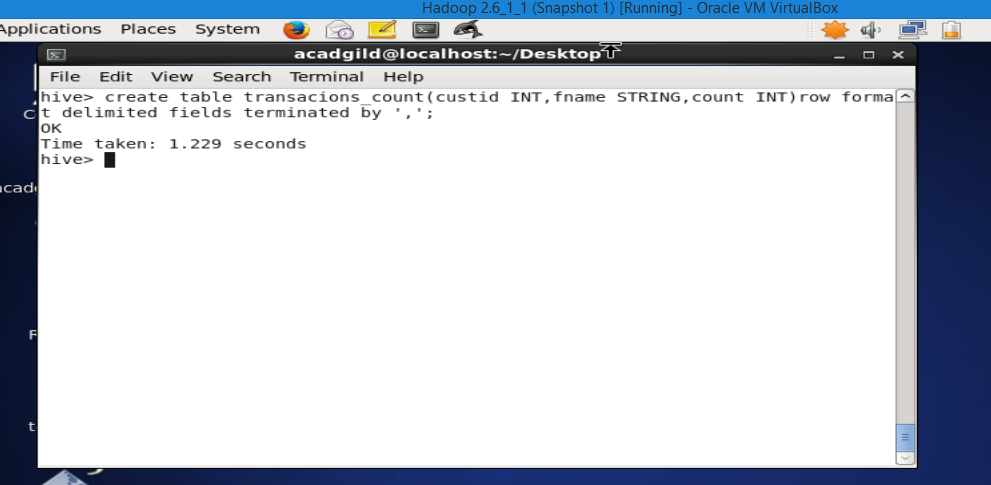
1. **Create a new table called TRANSACTIONS\_COUNT. This table should have three fields - custid, fname and count.**

We can use below command to create the table

**CREATE TABLE** transactions\_Count

**(** custid **INT,** fname **STRING,** txn\_count **INT )**

**ROW FORMAT DELIMITED FIELDS TERMINATED by ','**

****

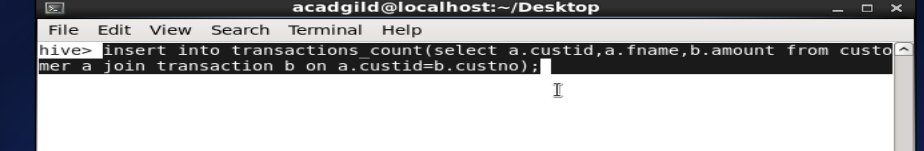
**3. Now write a hive query in such a way that the query populates the data obtained in Step 1 above and populate the table in step 2 above.**

To solve above problem we have to use **insert** query to insert data obtained from the problem number 2 into Transactions\_count

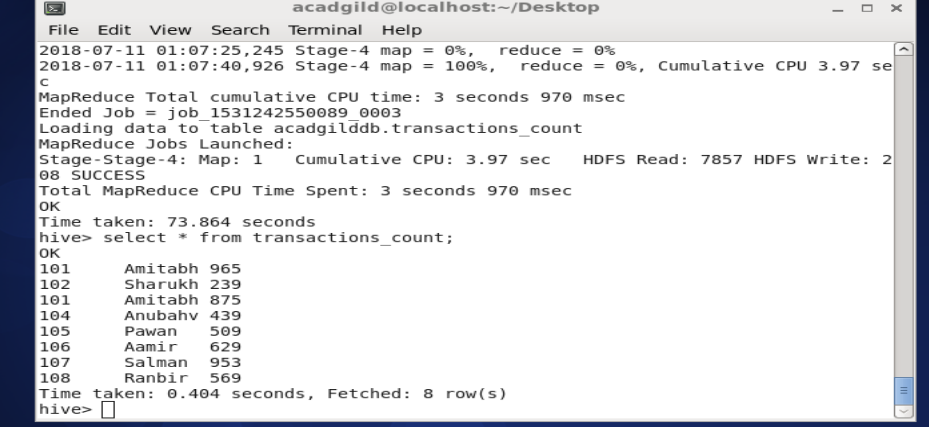
**INSERT INTO** transactions\_Count

**(SELECT** a.custid , a.fname , b.amount

**FROM** customer a **JOIN** transactions b **ON** a.custid = b.custno

****

**Result:** select \* from transaction\_count;

****

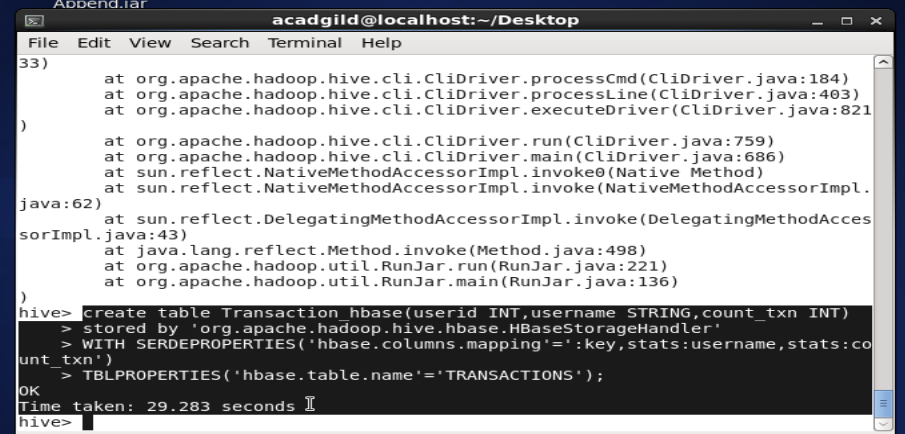
1. **Now let’s make the TRANSACTIONS\_hbase table HBase complaint. In the sense, use SerDes and Storage handler features of hive to change the TRANSACTIONS\_COUNT table to be able to create a TRANSACTIONS table in Hbase.**

**CREATE TABLE** Transaction\_hbase **(**userID **STRING,** username **STRING,** count\_txn **STRING)**

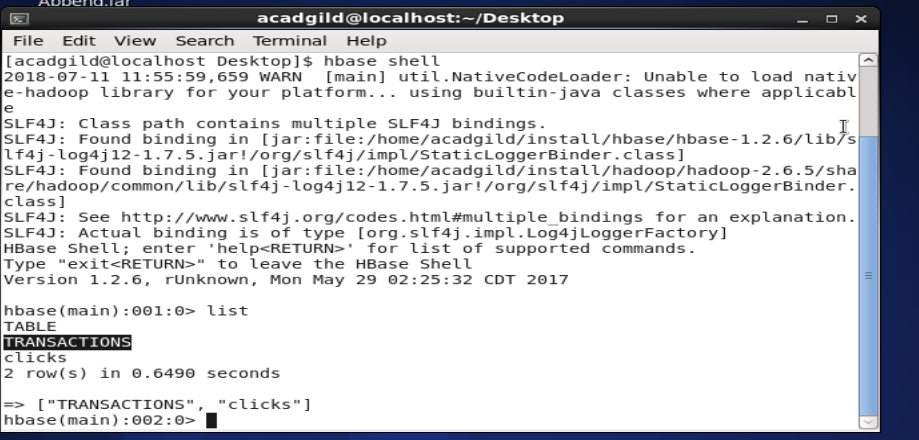
**STORED BY '**org.apache.hadoop.hive.hbase.HBaseStorageHandler**'**

**WITH SERDEPROPERTIES (**'hbase.columns.mapping' = ': key,stats:username,stats:count\_txn' )

**TBLPROPERTIES ('hbase.table.name' = 'TRANSACTION');**



In hbase we will check the list in hbase. After starting hbase shell write **list** in terminal to see the result

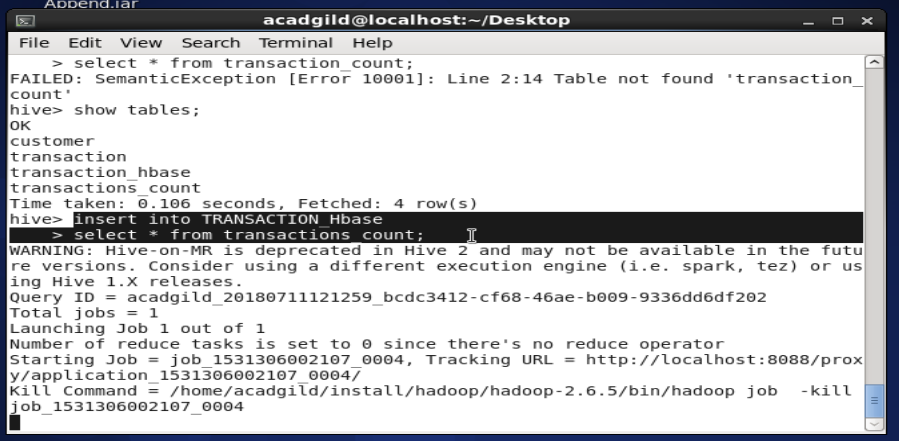


1. **Now insert the data in TRANSACTIONS\_HBase table using the query in step-3 again, this should populate the Hbase TRANSACTIONS table automatically.**

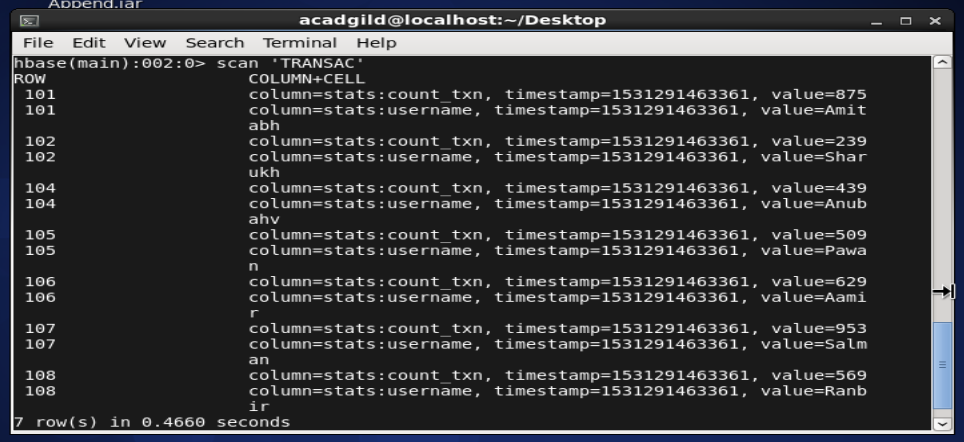
To solve above problem we use insert query to transfer data from TRANSACTIONS\_COUNT into TRANSACTIONS\_HBASE.

**INSERT INTO** Transaction\_hbase

**SELECT \* FROM** transactions\_Count**;**

****

**Now we will go to the hbase shell and by using scan we will check the table weather the data is available or not**

****

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

To solve above problem two-java program coded in the eclipse platform to scan and access the Transaction table.

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.client.Get;

import org.apache.hadoop.hbase.client.HTable;

import org.apache.hadoop.hbase.client.Result;

import org.apache.hadoop.hbase.util.Bytes;

public class accessHbaseTable{

public static void main(String[] args) throws IOException, Exception{ // Instantiating Configuration class

Configuration config = HBaseConfiguration.create()@SuppressWarnings({ "resource", "deprecation" })

HTable table = new HTable(config, "TRANSACTIONS");

* Instantiating Get class

Get g = new Get(Bytes.toBytes("101"));

// Reading the data

Result result = table.get(g);

// Reading values from Result class object

byte [] name = result.getValue(Bytes.toBytes("stats"),Bytes.toBytes("username")); byte [] txn = result.getValue(Bytes.toBytes("stats"),Bytes.toBytes("count\_txn"));

// Printing the values

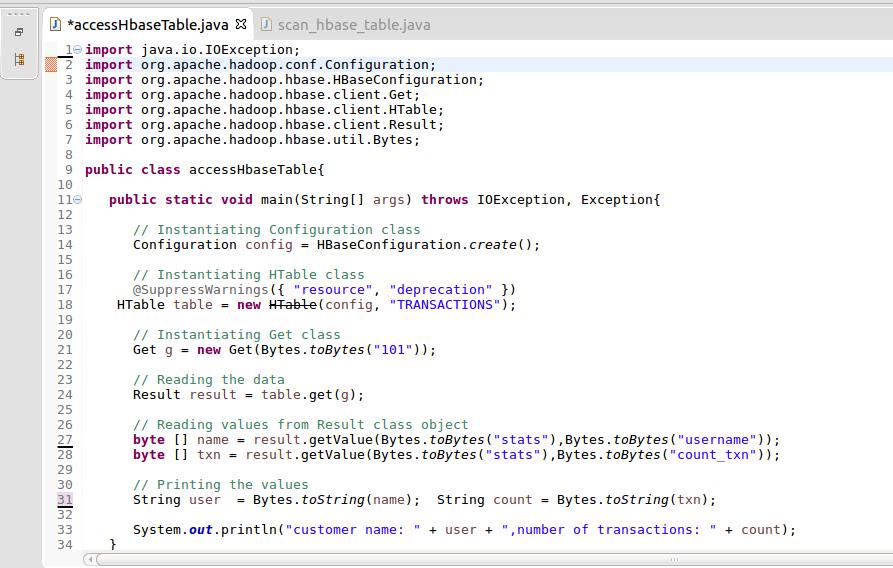
String user = Bytes.toString(name);

String count = Bytes.toString(txn);

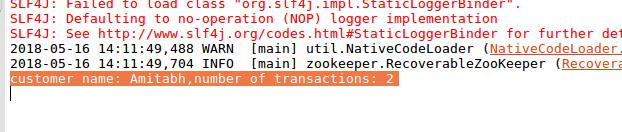
System.out.println("customer name: " + user + ",number of transactions: " + count);

}

}



**Output:** Access table program shows the value of row key 101



**Program to scan the hbase TRANSACTIONS table:**

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.util.Bytes;

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;

public class scan\_hbase\_table{

public static void main(String args[]) throws IOException{

// Instantiating Configuration class

Configuration config = HBaseConfiguration.create();

* Instantiating HTable class @SuppressWarnings({ "deprecation", "resource" })

HTable table = new HTable(config, "TRANSACTIONS");

// Instantiating the Scan class

Scan scan = new Scan();

* scanning the required columns scan.addColumn(Bytes.toBytes("stats"), Bytes.toBytes("count\_txn")); scan.addColumn(Bytes.toBytes("stats"), Bytes.toBytes("username"));
* Getting the scan result

ResultScanner scanner = table.getScanner(scan);

// Reading values from scan result

for (Result result = scanner.next(); result != null; result = scanner.next())

{

//assign row values in variable Row

String Row = Bytes.toString(result.getRow());

//assign column username values in name

String name = Bytes.toString(result.getValue("stats".getBytes(),"username".getBytes()));

//assign column count\_txn values in count

String count = Bytes.toString(result.getValue("stats".getBytes(),"count\_txn".getBytes()));

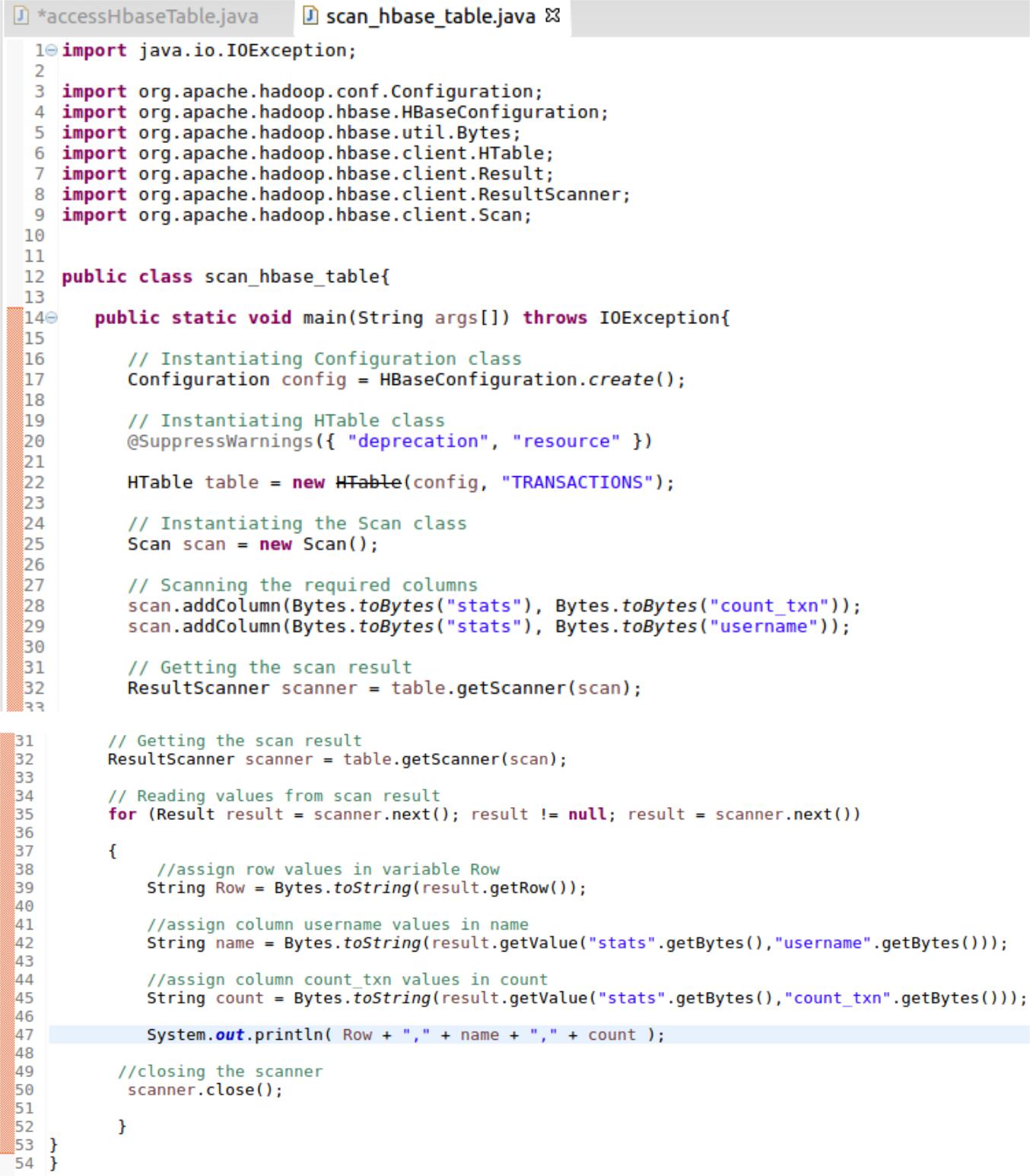
System.out.println( Row + "," + name + "," + count );

//closing the scanner

scanner.close();

}

}}



**Output:** scan program shows the content of the TRANSACTIONS table

