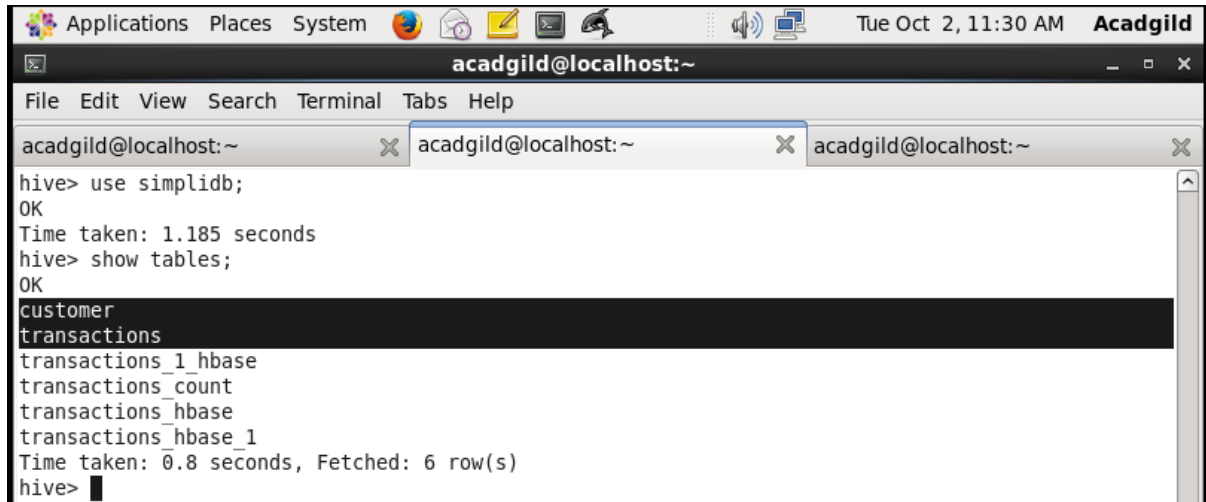


## Case Study II

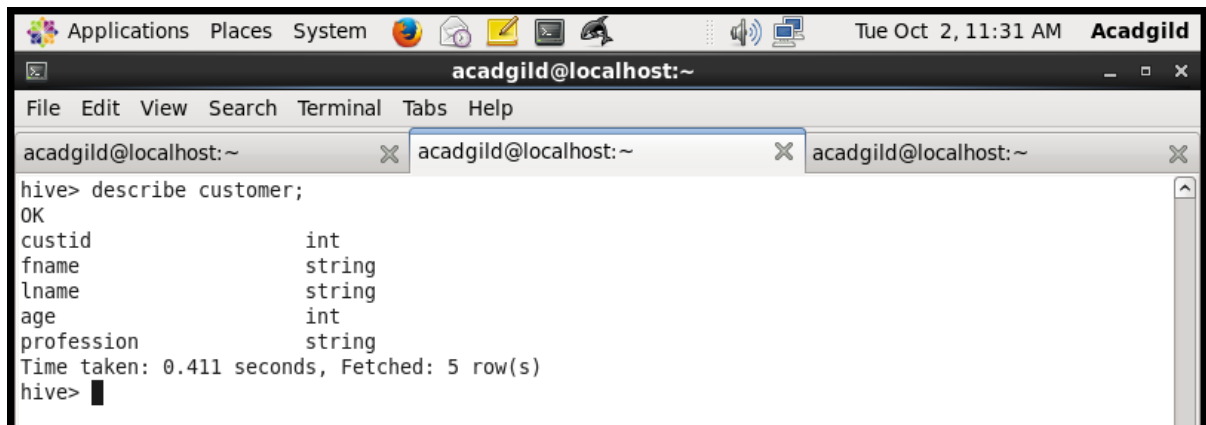
We have two tables **customer** and **transactions** in database **simplidb** as shown in the below screenshot:



```
Applications Places System Tue Oct 2, 11:30 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
hive> use simplidb;
OK
Time taken: 1.185 seconds
hive> show tables;
OK
customer
transactions
transactions_1_hbase
transactions_count
transactions_hbase
transactions_hbase_1
Time taken: 0.8 seconds, Fetched: 6 row(s)
hive>
```

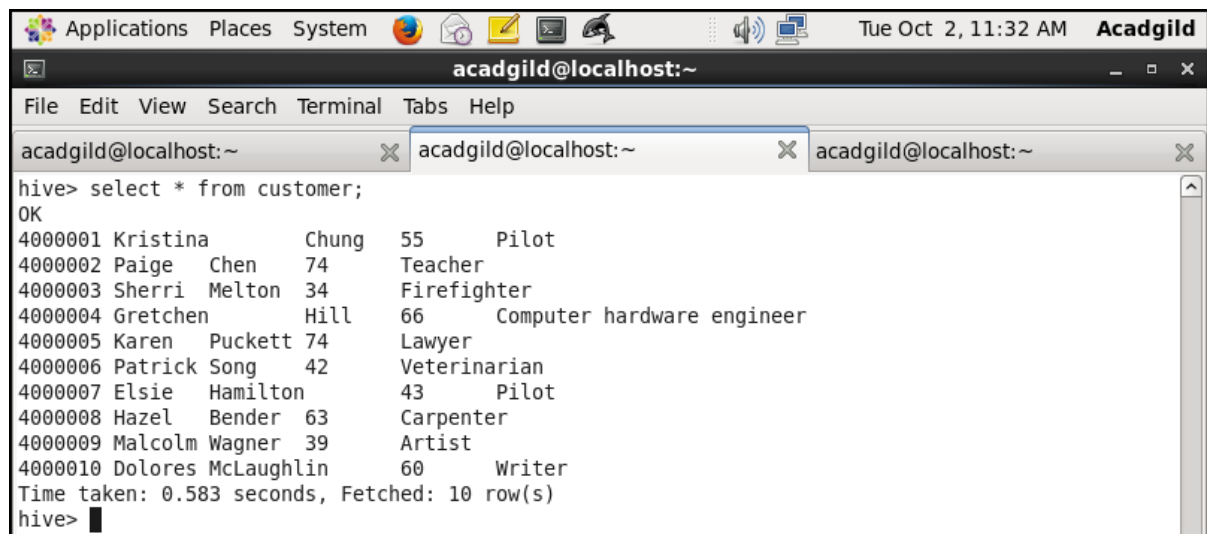
**Customer** table have five columns consist of **customer ID**, **customer first name**, **customer last name**, **age** and **customer profession**.

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



```
Applications Places System Tue Oct 2, 11:31 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
hive> describe customer;
OK
custid          int
fname           string
lname           string
age             int
profession       string
Time taken: 0.411 seconds, Fetched: 5 row(s)
hive>
```

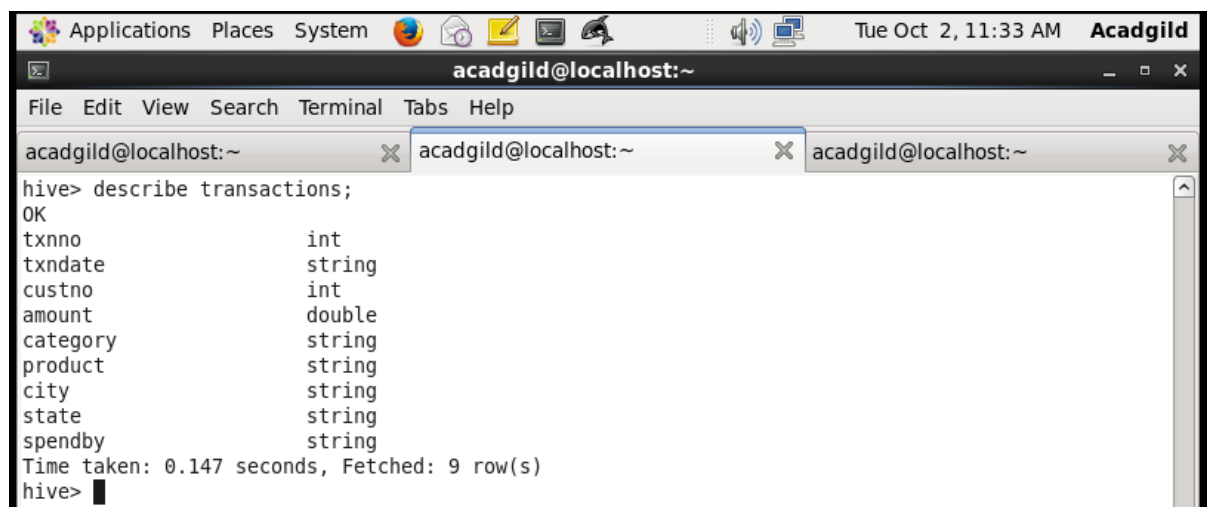
Data present in **customer** table is as shown below:



```
Applications Places System Tue Oct 2, 11:32 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
hive> select * from customer;
OK
4000001 Kristina Chung 55 Pilot
4000002 Paige Chen 74 Teacher
4000003 Sherri Melton 34 Firefighter
4000004 Gretchen Hill 66 Computer hardware engineer
4000005 Karen Puckett 74 Lawyer
4000006 Patrick Song 42 Veterinarian
4000007 Elsie Hamilton 43 Pilot
4000008 Hazel Bender 63 Carpenter
4000009 Malcolm Wagner 39 Artist
4000010 Dolores McLaughlin 60 Writer
Time taken: 0.583 seconds, Fetched: 10 row(s)
hive>
```

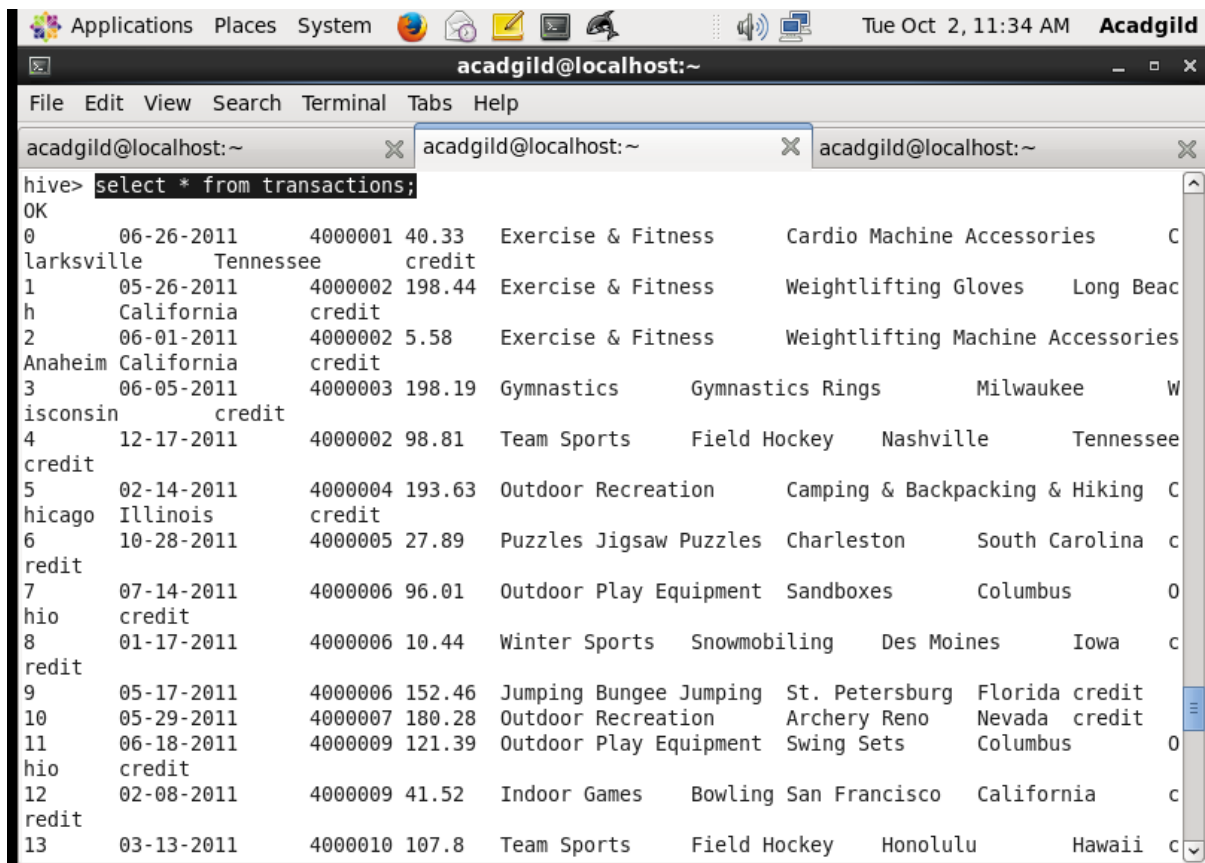
transaction table have nine columns consist of **transaction number, transaction date, customer ID, amount,category,product detail,city,state,spendby details.**

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



```
Applications Places System Tue Oct 2, 11:33 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
hive> describe transactions;
OK
txnno int
txndate string
custno int
amount double
category string
product string
city string
state string
spendby string
Time taken: 0.147 seconds, Fetched: 9 row(s)
hive>
```

Data present in **transactions** table is as shown below:



```
hive> select * from transactions;
```

0	06-26-2011	4000001	40.33	Exercise & Fitness	Cardio Machine Accessories	C
1	05-26-2011	4000002	198.44	Exercise & Fitness	Weightlifting Gloves	Long Beach
2	06-01-2011	4000002	5.58	Exercise & Fitness	Weightlifting Machine Accessories	
3	06-05-2011	4000003	198.19	Gymnastics	Gymnastics Rings	Milwaukee
4	12-17-2011	4000002	98.81	Team Sports	Field Hockey	Nashville Tennessee
5	02-14-2011	4000004	193.63	Outdoor Recreation	Camping & Backpacking & Hiking	C
6	10-28-2011	4000005	27.89	Puzzles Jigsaw Puzzles	Charleston	South Carolina
7	07-14-2011	4000006	96.01	Outdoor Play Equipment	Sandboxes	Columbus
8	01-17-2011	4000006	10.44	Winter Sports	Snowmobiling	Des Moines Iowa
9	05-17-2011	4000006	152.46	Jumping Bungee Jumping	St. Petersburg	Florida
10	05-29-2011	4000007	180.28	Outdoor Recreation	Archery	Reno Nevada
11	06-18-2011	4000009	121.39	Outdoor Play Equipment	Swing Sets	Columbus
12	02-08-2011	4000009	41.52	Indoor Games	Bowling	San Francisco California
13	03-13-2011	4000010	107.8	Team Sports	Field Hockey	Honolulu Hawaii

There are total 56 records in this table.



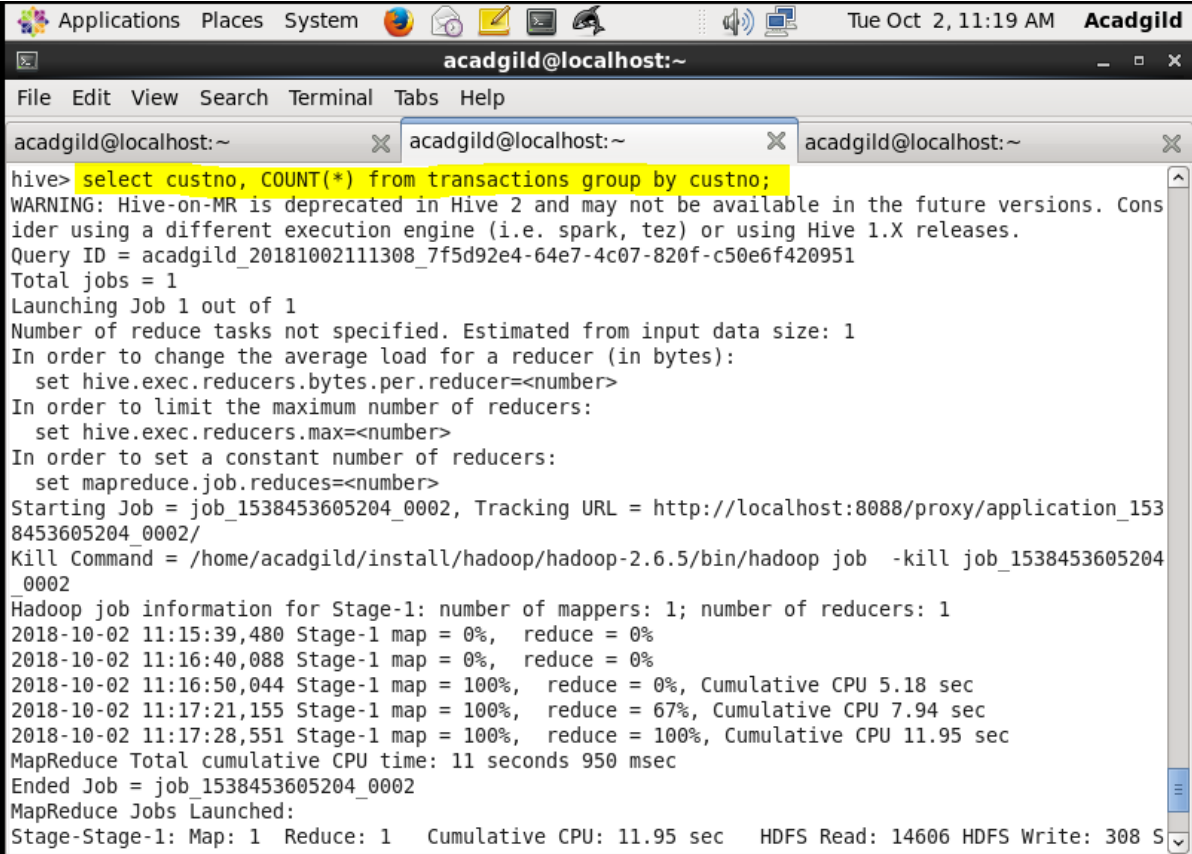
```
hive> select * from transactions;
```

47	10-23-2011	4000008	100.1	Outdoor Play Equipment	Swing Sets	Everett Washington
48	09-27-2011	4000007	157.94	Exercise & Fitness	Exercise Bands	Philadelphia
49	07-12-2011	4000010	144.59	Jumping Jumping Stilts	Cambridge	Massachusetts
50	10-20-2011	4000010	55.93	Jumping Pogo Sticks	Everett Washington	credit
51	02-17-2011	4000002	32.65	Water Sports	Life Jackets	Columbus Georgia
52	02-04-2011	4000005	44.82	Outdoor Play Equipment	Lawn Water Slides	Hampton
53	06-12-2011	4000004	44.46	Water Sports	Scuba Diving & Snorkeling	Charleston
54	10-03-2011	4000007	154.87	Outdoor Recreation	Running	Long Beach California
55	12-16-2011	4000006	106.11	Water Sports	Swimming	New York New York
56	06-21-2011	4000002	176.63	Outdoor Recreation	Geocaching	Boston Massachusetts
57	12-20-2011	4000003	178.2	Outdoor Recreation	Skating	San Jose California
58	12-29-2011	4000002	194.86	Water Sports	Windsurfing	Oklahoma City Oklahoma
59	11-07-2011	4000001	21.43	Winter Sports	Snowboarding	Philadelphia Pennsylvania

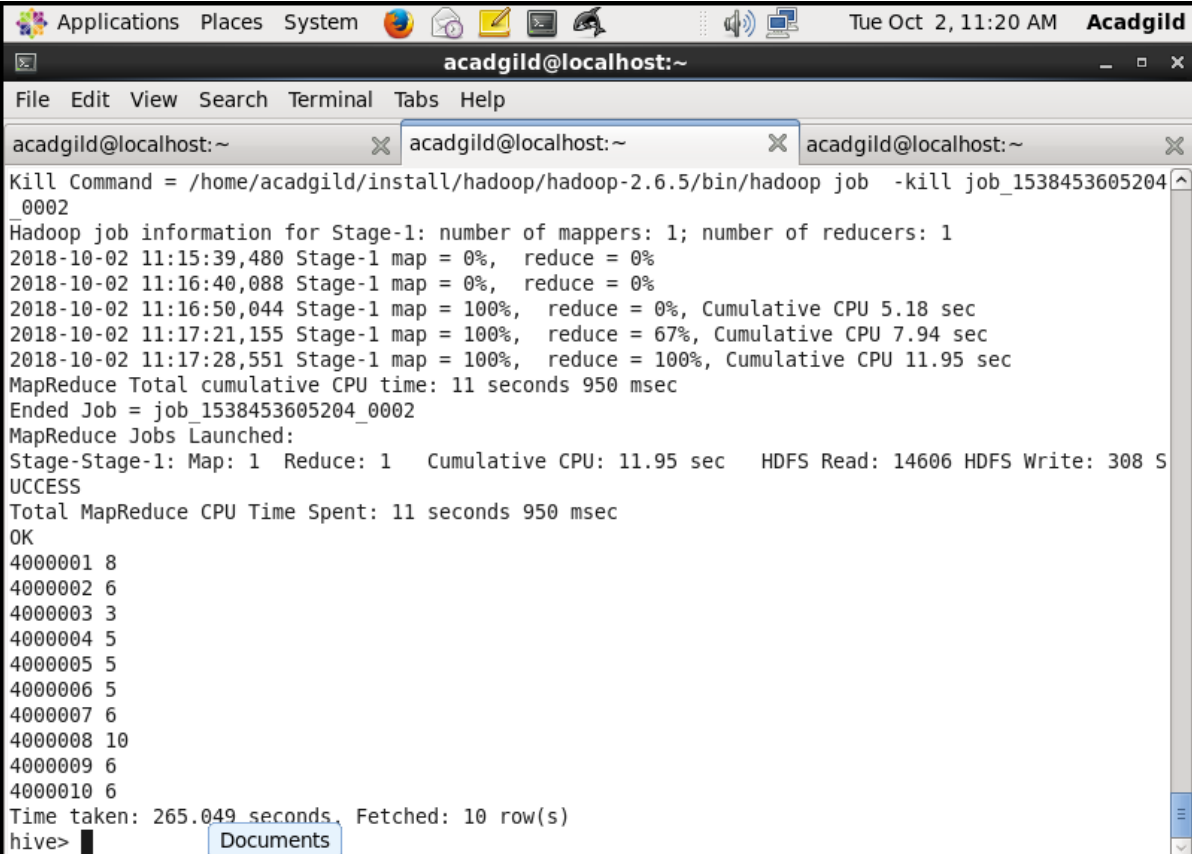
Time taken: 0.408 seconds, Fetched: 60 row(s)

# 1. Find out the number of transaction done by each customer (These should be take up in module 8 itself)

To find the number of transactions done by each customer the following query is used.



```
acadgild@localhost:~  
File Edit View Search Terminal Tabs Help  
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~  
hive> select custno, COUNT(*) from transactions group by custno;  
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.  
Query ID = acadgild_20181002111308_7f5d92e4-64e7-4c07-820f-c50e6f420951  
Total jobs = 1  
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_1538453605204_0002, Tracking URL = http://localhost:8088/proxy/application_1538453605204_0002/  
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1538453605204_0002  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  
2018-10-02 11:15:39,480 Stage-1 map = 0%, reduce = 0%  
2018-10-02 11:16:40,088 Stage-1 map = 0%, reduce = 0%  
2018-10-02 11:16:50,044 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.18 sec  
2018-10-02 11:17:21,155 Stage-1 map = 100%, reduce = 67%, Cumulative CPU 7.94 sec  
2018-10-02 11:17:28,551 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.95 sec  
MapReduce Total cumulative CPU time: 11 seconds 950 msec  
Ended Job = job_1538453605204_0002  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 11.95 sec HDFS Read: 14606 HDFS Write: 308 S
```

A screenshot of a Linux terminal window titled 'acadgild@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. There are three tabs open, all with the title 'acadgild@localhost:~'. The terminal output shows the command to kill a Hadoop job: 'Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job\_1538453605204\_0002'. This is followed by Hadoop job information for Stage-1, showing progress from 0% to 100% for map and reduce tasks, and cumulative CPU time. The output then shows 'MapReduce Total cumulative CPU time: 11 seconds 950 msec', 'Ended Job = job\_1538453605204\_0002', and 'MapReduce Jobs Launched: Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 11.95 sec HDFS Read: 14606 HDFS Write: 308 SUCCESS'. Finally, it shows 'Total MapReduce CPU Time Spent: 11 seconds 950 msec', 'OK', a list of customer IDs and their transaction counts (e.g., '4000001 8'), and 'Time taken: 265.049 seconds. Fetched: 10 row(s)'. The prompt 'hive>' is visible at the bottom.

```
acadgild@localhost:~  
File Edit View Search Terminal Tabs Help  
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~  
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1538453605204_0002  
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  
2018-10-02 11:15:39,480 Stage-1 map = 0%, reduce = 0%  
2018-10-02 11:16:40,088 Stage-1 map = 0%, reduce = 0%  
2018-10-02 11:16:50,044 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.18 sec  
2018-10-02 11:17:21,155 Stage-1 map = 100%, reduce = 67%, Cumulative CPU 7.94 sec  
2018-10-02 11:17:28,551 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 11.95 sec  
MapReduce Total cumulative CPU time: 11 seconds 950 msec  
Ended Job = job_1538453605204_0002  
MapReduce Jobs Launched:  
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 11.95 sec HDFS Read: 14606 HDFS Write: 308 S  
UCCESS  
Total MapReduce CPU Time Spent: 11 seconds 950 msec  
OK  
4000001 8  
4000002 6  
4000003 3  
4000004 5  
4000005 5  
4000006 5  
4000007 6  
4000008 10  
4000009 6  
4000010 6  
Time taken: 265.049 seconds. Fetched: 10 row(s)  
hive>
```

The above screenshot shows the output with customer id and no of transactions.

We can also find the number of transaction done by each customer by getting the name of the customer by using query as shown below

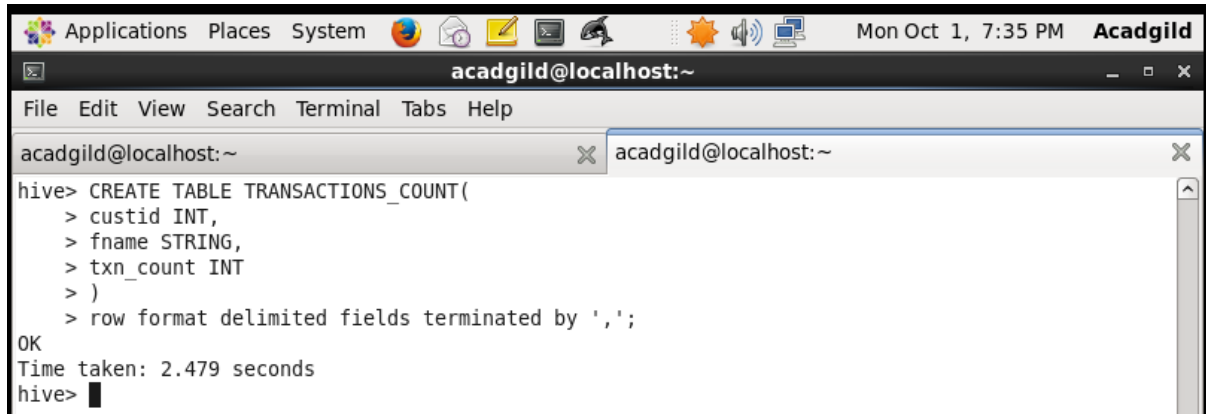
```
Applications Places System Tue Oct 2, 11:27 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
hive> select c.custid, c.fname, COUNT(t.custno) from customer c, transactions t where c.custid =
t.custno group by c.custid, c.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Cons
ider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181002112515_6c424b96-443c-4a66-ae15-8e54935b4404
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-slf
4j-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/slf4j/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]
2018-10-02 11:25:42 Starting to launch local task to process map join; maximum memory =
477626368
2018-10-02 11:25:47 Dump the side-table for tag: 0 with group count: 10 into file: file:/tmp/
acadgild/2d7121a6-69e6-4888-ab69-6ecf0cfd3ea5/hive_2018-10-02_11-25-15_278_6935035279810519606-1/
-local-10005/HashTable-Stage-2/MapJoin-mapfile00--.hashtable
2018-10-02 11:25:47 Uploaded 1 File to: file:/tmp/acadgild/2d7121a6-69e6-4888-ab69-6ecf0cfd3e
a5/hive_2018-10-02_11-25-15_278_6935035279810519606-1/-local-10005/HashTable-Stage-2/MapJoin-mapf
ile00--.hashtable (556 bytes)
2018-10-02 11:25:47 End of local task; Time Taken: 5.295 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):
```

In above screen-shot we are able to see the output containing transaction done by each customer with their **first name** and **customer ID**.

```
Applications Places System Tue Oct 2, 11:29 AM Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
Starting Job = job_1538453605204_0004, Tracking URL = http://localhost:8088/proxy/application_153
8453605204_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1538453605204
_0004
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-10-02 11:26:15,847 Stage-2 map = 0%, reduce = 0%
2018-10-02 11:26:43,705 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 5.84 sec
2018-10-02 11:27:00,696 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 10.19 sec
MapReduce Total cumulative CPU time: 10 seconds 190 msec
Ended Job = job_1538453605204_0004
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 10.19 sec HDFS Read: 18188 HDFS Write: 381 S
UCCESS
Total MapReduce CPU Time Spent: 10 seconds 190 msec
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: 109.895 seconds, Fetched 10000 rows in 109.895 seconds
hive>
```

**2. Create a new table called TRANSACTIONS\_COUNT. This table should have fields - custid, fname and count. (Again to be done in module 8)**

To create the table **TRANSACTIONS\_COUNT** below query is used.



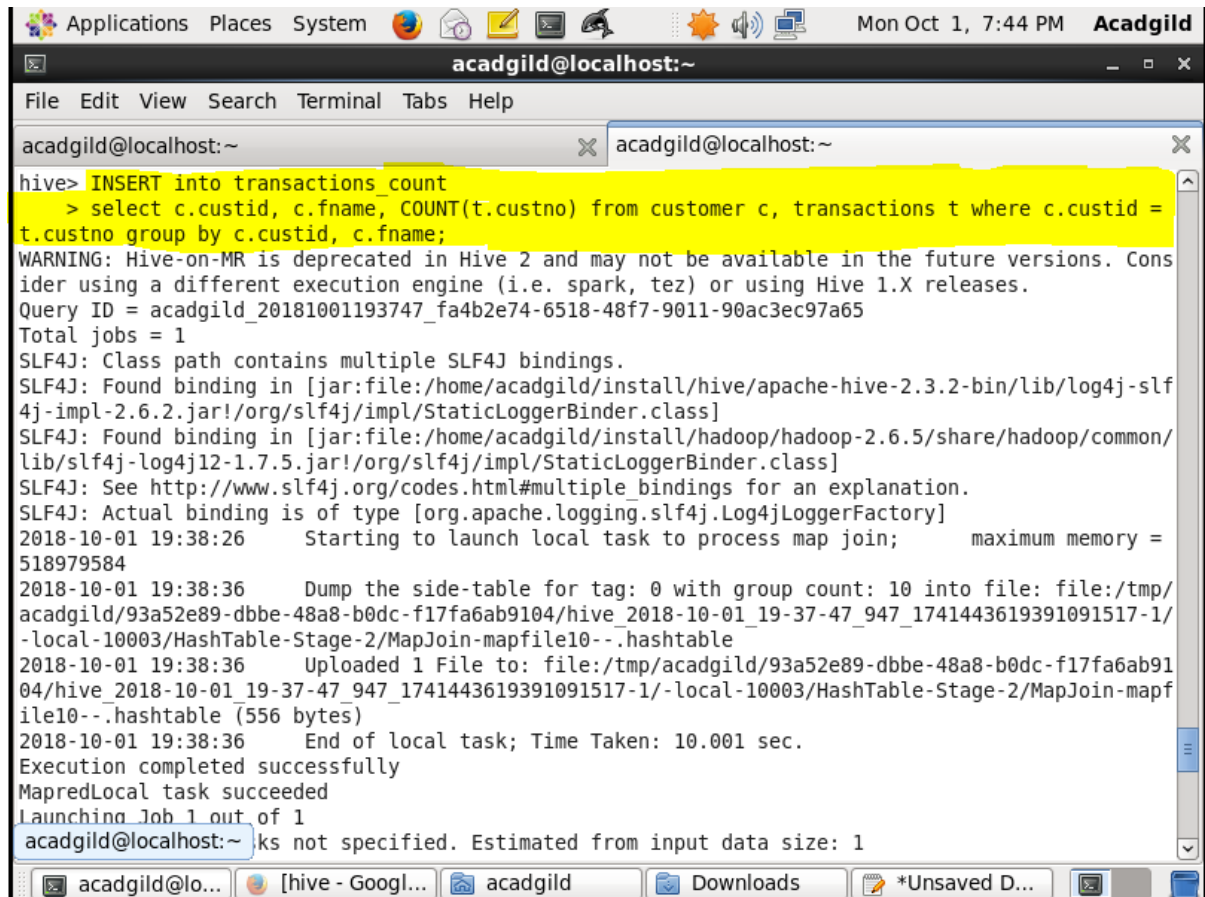
The screenshot shows a terminal window titled 'acadgild@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. Below the menu bar, there are two tabs, both labeled 'acadgild@localhost:~'. The terminal content shows a Hive prompt 'hive>' followed by the command to create a table: 'CREATE TABLE TRANSACTIONS\_COUNT('). The command is indented with '>' for each line: '> custid INT,', '> fname STRING,', '> txn\_count INT', and '> )'. The final line of the command is '> row format delimited fields terminated by ',';'. The terminal then shows 'OK' and 'Time taken: 2.479 seconds'. The prompt 'hive>' is followed by a cursor.

```
acadgild@localhost:~  
File Edit View Search Terminal Tabs Help  
acadgild@localhost:~ acadgild@localhost:~  
hive> CREATE TABLE TRANSACTIONS_COUNT(  
  > custid INT,  
  > fname STRING,  
  > txn_count INT  
  > )  
  > row format delimited fields terminated by ',';  
OK  
Time taken: 2.479 seconds  
hive> █
```



**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. (This has to be done in module 9).**

To solve above problem we have to use insert query to insert data obtained from the task-1 into **Transactions\_count**



```
acadgild@localhost:~  
File Edit View Search Terminal Tabs Help  
acadgild@localhost:~ acadgild@localhost:~  
hive> INSERT into transactions_count  
      > select c.custid, c.fname, COUNT(t.custno) from customer c, transactions t where c.custid =  
      t.custno group by c.custid, c.fname;  
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.  
Query ID = acadgild_20181001193747_fa4b2e74-6518-48f7-9011-90ac3ec97a65  
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/slf4j/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]  
2018-10-01 19:38:26 Starting to launch local task to process map join; maximum memory = 518979584  
2018-10-01 19:38:36 Dump the side-table for tag: 0 with group count: 10 into file: file:/tmp/acadgild/93a52e89-dbbe-48a8-b0dc-f17fa6ab9104/hive_2018-10-01_19-37-47_947_1741443619391091517-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile10--.hashtable  
2018-10-01 19:38:36 Uploaded 1 File to: file:/tmp/acadgild/93a52e89-dbbe-48a8-b0dc-f17fa6ab9104/hive_2018-10-01_19-37-47_947_1741443619391091517-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile10--.hashtable (556 bytes)  
2018-10-01 19:38:36 End of local task; Time Taken: 10.001 sec.  
Execution completed successfully  
MapredLocal task succeeded  
Launching Job 1 out of 1  
acacgild@localhost:~ ks not specified. Estimated from input data size: 1
```



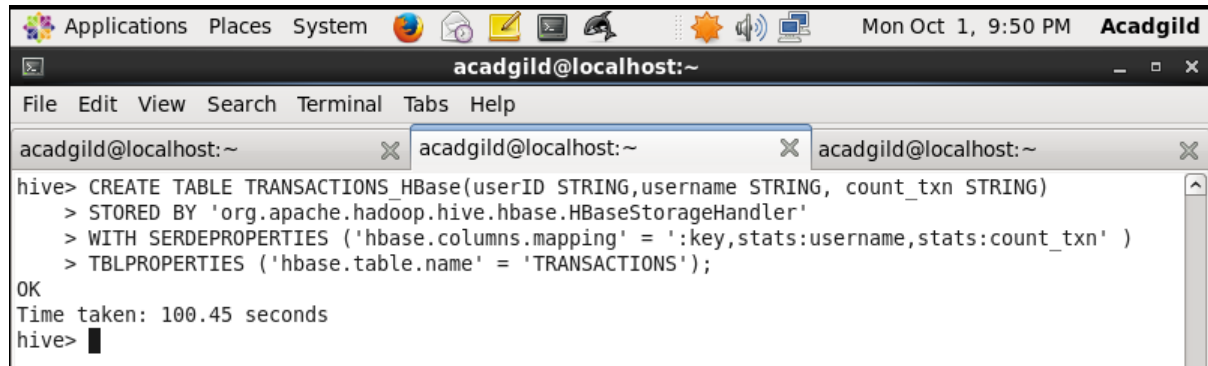
The screenshot shows a terminal window titled 'acadgild@localhost:~'. The window displays the output of a Hive job and a subsequent query. The job output includes progress for Stage-2, cumulative CPU time, and job completion status. The query 'select \* from transactions\_count;' is executed, returning 10 rows of data. The data is highlighted in yellow in the original image.

```
acadgild@localhost:~
2018-10-01 19:39:56,590 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 6.0 sec
2018-10-01 19:40:33,495 Stage-2 map = 100%, reduce = 67%, Cumulative CPU 10.28 sec
2018-10-01 19:40:34,793 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 11.58 sec
MapReduce Total cumulative CPU time: 11 seconds 580 msec
Ended Job = job_1538382707980_0003
Loading data to table simplidb.transactions_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 11.58 sec HDFS Read: 18922 HDFS Write: 258 S
UCCESS
Total MapReduce CPU Time Spent: 11 seconds 580 msec
OK
Time taken: 174.847 seconds
hive>
hive> select * from transactions_count;
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: 0.823 seconds, Fetched: 10 row(s)
hive>
```

Above screen, shot shows that data obtained from query in case1 has successfully inserted in table **transactions\_count**.

4. Now lets make the TRANSACTIONS\_COUNT table Hbase complaint. In the sence, use Ser Des And Storate handler features of hive to change the TRANSACTIONS\_COUNT table to be able to create a TRANSACTIONS table in Hbase. (This has to be done in module 10)

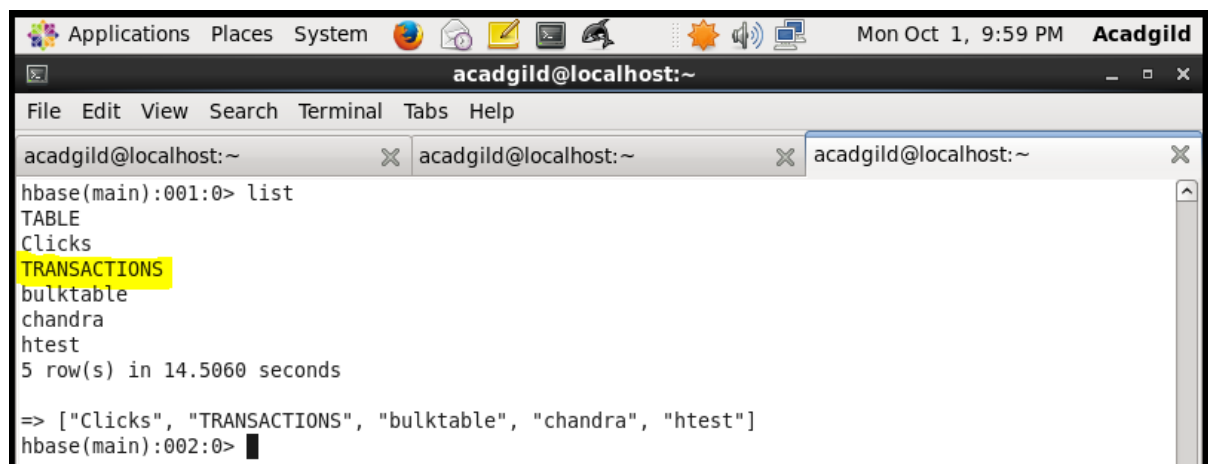
Below query is used to create the table in **Hbase** same as table in **Hive** with **serde** properties.



A terminal window titled 'acadgild@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). It shows a Hive command to create a table in HBase. The command is: `hive> CREATE TABLE TRANSACTIONS_HBase(userID STRING,username STRING, count_txn STRING)`. The next line is `> STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'`. The third line is `> WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,stats:username,stats:count_txn' )`. The fourth line is `> TBLPROPERTIES ('hbase.table.name' = 'TRANSACTIONS');`. The output is 'OK' and 'Time taken: 100.45 seconds'. The prompt 'hive>' is followed by a cursor.

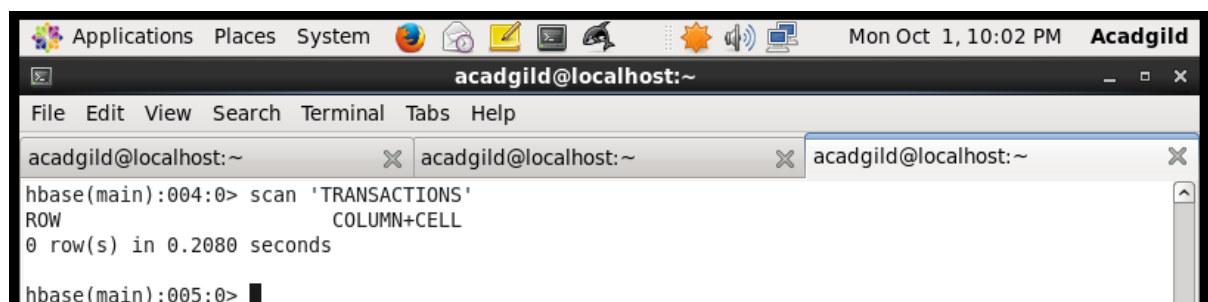
```
acadgild@localhost:~
acadgild@localhost:~
acadgild@localhost:~
hive> CREATE TABLE TRANSACTIONS_HBase(userID STRING,username STRING, count_txn STRING)
> STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
> WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,stats:username,stats:count_txn' )
> TBLPROPERTIES ('hbase.table.name' = 'TRANSACTIONS');
OK
Time taken: 100.45 seconds
hive>
```

Below screenshot shows that table has been created in hbase.



A terminal window titled 'acadgild@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). It shows the output of the 'list' command in HBase. The command is `hbase(main):001:0> list`. The output is: `TABLE`, `Clicks`, `TRANSACTIONS` (highlighted in yellow), `bulktable`, `chandra`, `htest`. Below this is `5 row(s) in 14.5060 seconds`. The next line is `=> ["Clicks", "TRANSACTIONS", "bulktable", "chandra", "htest"]`. The prompt 'hbase(main):002:0>' is followed by a cursor.

```
acadgild@localhost:~
acadgild@localhost:~
acadgild@localhost:~
hbase(main):001:0> list
TABLE
Clicks
TRANSACTIONS
bulktable
chandra
htest
5 row(s) in 14.5060 seconds
=> ["Clicks", "TRANSACTIONS", "bulktable", "chandra", "htest"]
hbase(main):002:0>
```



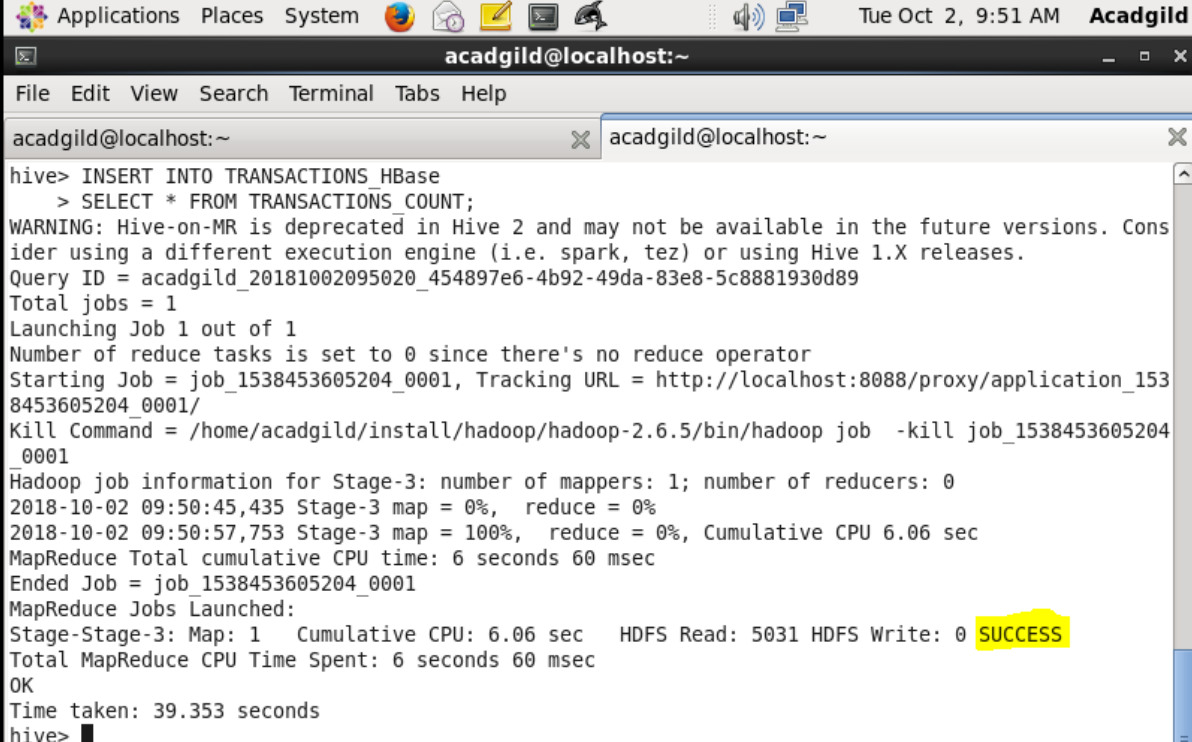
A terminal window titled 'acadgild@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). It shows the output of the 'scan' command in HBase. The command is `hbase(main):004:0> scan 'TRANSACTIONS'`. The output is: `ROW`, `COLUMN+CELL`. Below this is `0 row(s) in 0.2080 seconds`. The prompt 'hbase(main):005:0>' is followed by a cursor.

```
acadgild@localhost:~
acadgild@localhost:~
acadgild@localhost:~
hbase(main):004:0> scan 'TRANSACTIONS'
ROW
COLUMN+CELL
0 row(s) in 0.2080 seconds
hbase(main):005:0>
```

5. 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.

Below screenshot shows inserting data is succeeded.



```
acadgild@localhost:~  
File Edit View Search Terminal Tabs Help  
acadgild@localhost:~ acadgild@localhost:~  
hive> INSERT INTO TRANSACTIONS_HBase  
  > SELECT * FROM TRANSACTIONS_COUNT;  
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.  
Query ID = acadgild_20181002095020_454897e6-4b92-49da-83e8-5c8881930d89  
Total jobs = 1  
Launching Job 1 out of 1  
Number of reduce tasks is set to 0 since there's no reduce operator  
Starting Job = job_1538453605204_0001, Tracking URL = http://localhost:8088/proxy/application_1538453605204_0001/  
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1538453605204_0001  
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0  
2018-10-02 09:50:45,435 Stage-3 map = 0%, reduce = 0%  
2018-10-02 09:50:57,753 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 6.06 sec  
MapReduce Total cumulative CPU time: 6 seconds 60 msec  
Ended Job = job_1538453605204_0001  
MapReduce Jobs Launched:  
Stage-Stage-3: Map: 1 Cumulative CPU: 6.06 sec HDFS Read: 5031 HDFS Write: 0 SUCCESS  
Total MapReduce CPU Time Spent: 6 seconds 60 msec  
OK  
Time taken: 39.353 seconds  
hive>
```

6.6. 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.

#### Program to access the hbase 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();
```

```
// Instantiating HTable class
```

```
@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
```

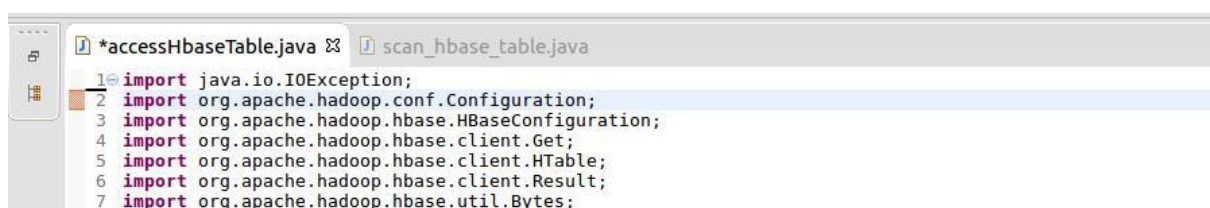
```
String user = Bytes.toString(name);
```

```
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

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details
2018-05-16 14:11:49,488 WARN [main] util.NativeCodeLoader (NativeCodeLoader.java:111) - Unable to load native code library: /lib/x86_64-linux-gnu/libc.so.6
2018-05-16 14:11:49,704 INFO [main] zookeeper.RecoverableZooKeeper (RecoverableZooKeeper.java:111) - customer name: Amitabh,number of transactions: 2
```

**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();
```

```
}
```

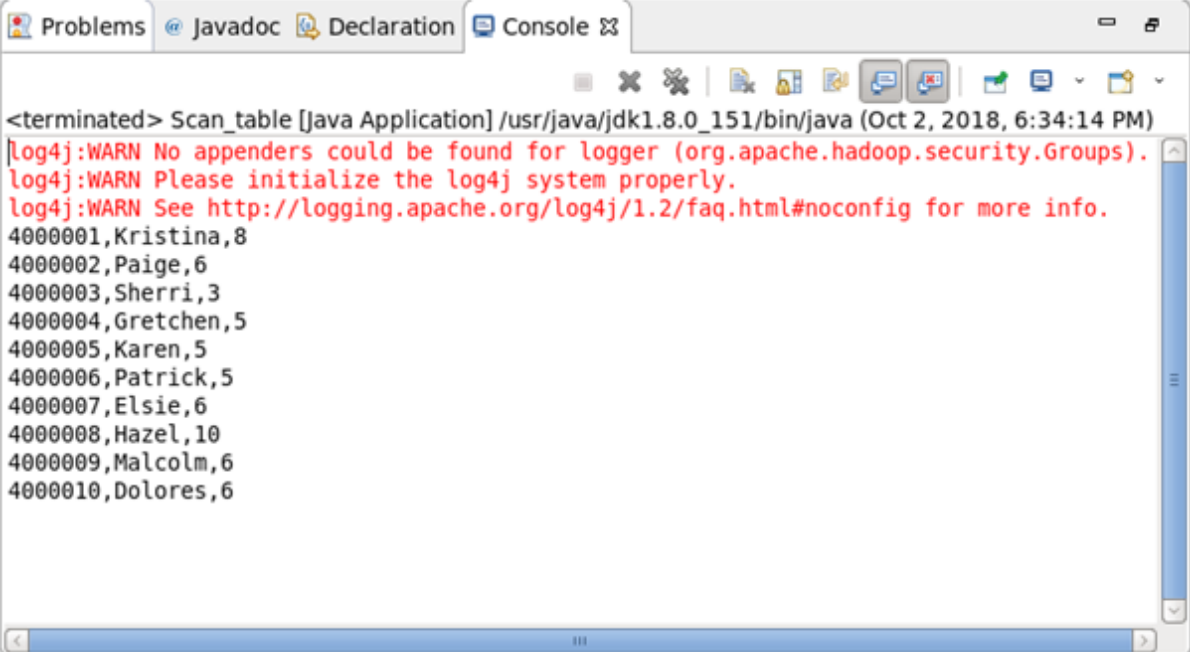
```
}}
```

\*accessHbaseTable.java

scan\_hbase\_table.java

```
1 import java.io.IOException;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.hbase.HBaseConfiguration;
5 import org.apache.hadoop.hbase.util.Bytes;
6 import org.apache.hadoop.hbase.client.HTable;
7 import org.apache.hadoop.hbase.client.Result;
8 import org.apache.hadoop.hbase.client.ResultScanner;
9 import org.apache.hadoop.hbase.client.Scan;
10
11
12 public class scan_hbase_table{
13
14     public static void main(String args[]) throws IOException{
15
16         // Instantiating Configuration class
17         Configuration config = HBaseConfiguration.create();
18
19         // Instantiating HTable class
20         @SuppressWarnings({ "deprecation", "resource" })
21
22         HTable table = new HTable(config, "TRANSACTIONS");
23
24         // Instantiating the Scan class
25         Scan scan = new Scan();
26
27         // Scanning the required columns
28         scan.addColumn(Bytes.toBytes("stats"), Bytes.toBytes("count_txn"));
29         scan.addColumn(Bytes.toBytes("stats"), Bytes.toBytes("username"));
30
31         // Getting the scan result
32         ResultScanner scanner = table.getScanner(scan);
33
34         // Getting the scan result
35         ResultScanner scanner = table.getScanner(scan);
36
37         // Reading values from scan result
38         for (Result result = scanner.next(); result != null; result = scanner.next())
39         {
40             //assign row values in variable Row
41             String Row = Bytes.toString(result.getRow());
42
43             //assign column username values in name
44             String name = Bytes.toString(result.getValue("stats".getBytes(), "username".getBytes()));
45
46             //assign column count_txn values in count
47             String count = Bytes.toString(result.getValue("stats".getBytes(), "count_txn".getBytes()));
48
49             System.out.println( Row + "," + name + "," + count );
50
51             //closing the scanner
52             scanner.close();
53         }
54     }
```

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



```
<terminated> Scan_table [Java Application] /usr/java/jdk1.8.0_151/bin/java (Oct 2, 2018, 6:34:14 PM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.security.Groups).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
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
```