Case Study – II. Customer Transaction.

1. Start Hadoop componetns in VM using the following command and check whether all the daemons are running correctly:
   $start-all.hs
   $jps

```
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/10/02 13:09:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.l
ocaldomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.l
ocaldomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondaryname
node-localhost.localdomain.out
18/10/02 13:09:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.
localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localho
st.localdomain.out
[acadgild@localhost ~]$
[acadgild@localhost ~]$ jps
3649 DataNode
4101 NodeManager
3846 SecondaryNameNode
3544 NameNode
4425 Jps
3996 ResourceManager
[acadgild@localhost ~]$
```

2. Now start the Hive shell as follows:

```
[acadgild@localhost ~]$ hive
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/slf4
/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]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.
ar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution eng
ne (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

3. Now check what are the databases available in the hive as follows:
   hive>show databases;

```
hive> show databases;
OK
default
sports
Time taken: 0.064 seconds, Fetched: 2 row(s)
```

4. Now we have 2 databases, called ***default*** and ***sports***. We will use ***default*** database for creation of tables:
hive> default

```
File  Edit  View  Search  Terminal  Help
hive> use default;
OK
Time taken: 0.039 seconds
```

5. Now create a table called ***customer*** as follows:

hive > CREATE TABLE CUSTOMER (

- ***Custid INT,***
- ***Fname string,***
- ***Lname string,***
- ***Age int,***

- ***Profession string***
- ***)***
- ***Row format delimited fields terminated by ‘,’;***

```
hive> CREATE TABLE CUSTOMER(
    > custid INT,
    > fname STRING,
    > lname STRING,
    > age INT,
    > profession STRING
    > )
    > row format delimited fields terminated by ',';
OK
Time taken: 0.162 seconds
```


```
hive> CREATE TABLE CUSTOMER(
    > custid INT,
    > fname STRING,
    > lname STRING,
    > age INT,
    > profession STRING
    > )
    > row format delimited fields terminated by ',';
OK
Time taken: 0.162 seconds
```


6. Now load the data into the file explicitly from a file as follows:
hive > LOAD DATA LOCAL INPUT ‘/home/acadgild/Downloads/hive/custs.txt’ into table CUSTOMER;

```
File  Edit  View  Search  Terminal  Help
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/hive/custs.txt' into table CUSTOMER;
Loading data to table default.customer
OK
Time taken: 0.827 seconds
```

7. Now we will verify data from the table customer as follows:
hive> select * from customer;

```
File  Edit  View  Search  Terminal  Help
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: 3.16 seconds, Fetched: 10 row(s)
```

8. Now create a transaction table called ***TXNRECORDS*** as follows:
hive> CREATE TABLE TXNRECORDS (txno INT, txndate STRING, custono INT, amount DOUBLE, category STRING, product STRING, city STRING, state STRING, spendby STRING) row format delimited fields terminated by ‘,’;

```
File  Edit  View  Search  Terminal  Help
hive> CREATE TABLE TXNRECORDS (txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, city S
RING, state STRING, spendby STRING)
    > row format delimited fields terminated by ',';
OK
Time taken: 0.179 seconds
```

9. Now load the data into the ***txnrecords*** as follows:

hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/hive/txns.txt' INTO TABLE TXNRECORDS;

```
File  Edit  View  Search  Terminal  Help
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/hive/txns.txt' into table TXNRECORDS
    > ;
Loading data to table default.txnrecords
OK
Time taken: 0.848 seconds
```

10. Now display the data from the table ***txnrecords*** as follows:

```
File  Edit  View  Search  Terminal  Help
hive> select * from txnrecords;
OK
0       06-26-2011      4000001 40.33   Exercise & Fitness      Cardio Machine Accessories      Clarksville     Tennessee
redit
1       05-26-2011      4000002 198.44  Exercise & Fitness      Weightlifting Gloves    Long Beach      California      cred
t
2       06-01-2011      4000002 5.58    Exercise & Fitness      Weightlifting Machine Accessories       Anaheim California
redit
3       06-05-2011      4000003 198.19  Gymnastics      Gymnastics Rings        Milwaukee       Wisconsin       credit
4       12-17-2011      4000002 98.81   Team Sports     Field Hockey    Nashville       Tennessee       credit
5       02-14-2011      4000004 193.63  Outdoor Recreation      Camping & Backpacking & Hiking  Chicago Illinois        cred
t
6       10-28-2011      4000005 27.89   Puzzles Jigsaw Puzzles  Charleston      South Carolina  credit
7       07-14-2011      4000006 96.01   Outdoor Play Equipment  Sandboxes       Columbus        Ohio    credit
8       01-17-2011      4000006 10.44   Winter Sports   Snowmobiling    Des Moines      Iowa    credit
9       05-17-2011      4000006 152.46  Jumping Bungee Jumping  St. Petersburg  Florida credit
10      05-29-2011      4000007 180.28  Outdoor Recreation      Archery Reno    Nevada  credit
11      06-18-2011      4000009 121.39  Outdoor Play Equipment  Swing Sets      Columbus        Ohio    credit
12      02-08-2011      4000009 41.52   Indoor Games    Bowling San Francisco   California      credit
13      03-13-2011      4000010 107.8   Team Sports     Field Hockey    Honolulu        Hawaii  credit
14      02-25-2011      4000010 36.81   Gymnastics      Vaulting Horses Los Angeles     California      credit
15      10-20-2011      4000001 137.64  Combat Sports   Fencing Honolulu        Hawaii  credit
16      05-28-2011      4000010 35.56   Exercise & Fitness      Free Weight Bars        Columbia        South Carolina  cred
t
```

Task 1:

Find the number of transactions done by each customer.

1. Write query as follows:

hive> select customer.custid, customer.fname, COUNT(txnrecords.custno) from customer, txnrecords where customer.custid=txnrecords.custno group by customer.custid, customer.fname;

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help
hive> select customer.custid,  customer.fname, COUNT(txnrecords.custno) from customer, txnrecords where customer.custid=txnre
cords.custno group by customer.custid, customer.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181002132257_f647ff1e-5027-42a4-bb05-f50799219624
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]
2018-10-02 13:23:08     Starting to launch local task to process map join;      maximum memory = 477626368
2018-10-02 13:23:11     Dump the side-table for tag: 0 with group count: 10 into file: file:/tmp/acadgild/5298d069-6441-4e0b-
8001-05ee7b082554/hive_2018-10-02_13-22-57_852_1264447518237052877-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile00--.hasht
able
2018-10-02 13:23:11     Uploaded 1 File to: file:/tmp/acadgild/5298d069-6441-4e0b-8001-05ee7b082554/hive_2018-10-02_13-22-57_
852_1264447518237052877-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile00--.hashtable (556 bytes)
2018-10-02 13:23:11     End of local task; Time Taken: 2.644 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_1538465981645_0001, Tracking URL = http://localhost:8088/proxy/application_1538465981645_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1538465981645_0001
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-10-02 13:23:28,994 Stage-2 map = 0%,  reduce = 0%
2018-10-02 13:23:38,125 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 4.07 sec
2018-10-02 13:23:47,044 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.52 sec
MapReduce Total cumulative CPU time: 7 seconds 520 msec
Ended Job = job_1538465981645_0001
MapReduce Jobs Launched:
```

```
File  Edit  View  Search  Terminal  Help
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.52 sec   HDFS Read: 18142 HDFS Write: 381 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 520 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: 51.462 seconds, Fetched: 10 row(s)
hive>
```

2. Here we can observe that all the customer names and their customer numbers and the number of transactions done by each customers.

Task 2:
Create a new table called TRANSACTIONS_COUNT. This table should have 3 fields – custid, fname and count.

1. Create the table ***transactions_count*** table as follows:

hive> CREATE TABLE TRANSACTIONS_COUNT (

- ***custid INT,***
- ***fname STRING,***
- ***count INT***
- ***);***

```
File  Edit  View  Search  Terminal  Help
hive>
    > CREATE TABLE TRANSACTIONS_COUNT(
    > custid INT,
    > fname STRING,
    > count INT
    > );
OK
Time taken: 0.167 seconds
```

2. We created the table called TRANSACTIONS_COUNT.
3. Now lets see the table created or not inside the database ***default*** as follows:

hive> show tables;

```
File  Edit  View  Search  Terminal  Help
hive> show tables;
OK
customer
emp_temp
emp_temp1
transactions_count
txnrecords
Time taken: 0.066 seconds, Fetched: 5 row(s)
hive>
```

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

1. Write a query to populate the data into the table ***transactions_count*** as follows:

hive> insert into transactions_count select customer.custid, customer.fname, COUNT(txnrecords.custno) from customer, txnrecords where customer.custid=txnrecords.custno group by customer.custid, customer.fname;

```
hive> insert into transactions_count select customer.custid,  customer.fname, COUNT(txnrecords.custno) from customer, txnrec
rds where customer.custid=txnrecords.custno group by customer.custid, customer.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different exec
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181002163648_45d60e80-2ec5-48a8-ae10-9b7e894ac1aa
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/slf4
/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 16:36:58     Starting to launch local task to process map join;      maximum memory = 477626368
2018-10-02 16:37:01     Dump the side-table for tag: 0 with group count: 10 into file: file:/tmp/acadgild/5298d069-6441-4e0b
8001-05ee7b082554/hive_2018-10-02_16-36-48_941_840049763826257214-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile10--.hasht
ble
2018-10-02 16:37:01     Uploaded 1 File to: file:/tmp/acadgild/5298d069-6441-4e0b-8001-05ee7b082554/hive_2018-10-02_16-36-48
941_840049763826257214-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile10--.hashtable (556 bytes)
2018-10-02 16:37:01     End of local task; Time Taken: 2.735 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_1538465981645_0002, Tracking URL = http://localhost:8088/proxy/application_1538465981645_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1538465981645_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-10-02 16:37:11,578 Stage-2 map = 0%,  reduce = 0%
2018-10-02 16:37:20,435 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 4.45 sec
2018-10-02 16:37:29,270 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 9.79 sec
MapReduce Total cumulative CPU time: 9 seconds 790 msec
```

```
File  Edit  View  Search  Terminal  Help
2018-10-02 16:37:29,270 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 9.79 sec
MapReduce Total cumulative CPU time: 9 seconds 790 msec
Ended Job = job_1538465981645_0002
Loading data to table default.transactions_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 9.79 sec   HDFS Read: 18781 HDFS Write: 257 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 790 msec
OK
Time taken: 43.165 seconds
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: 0.18 seconds, Fetched: 10 row(s)
```

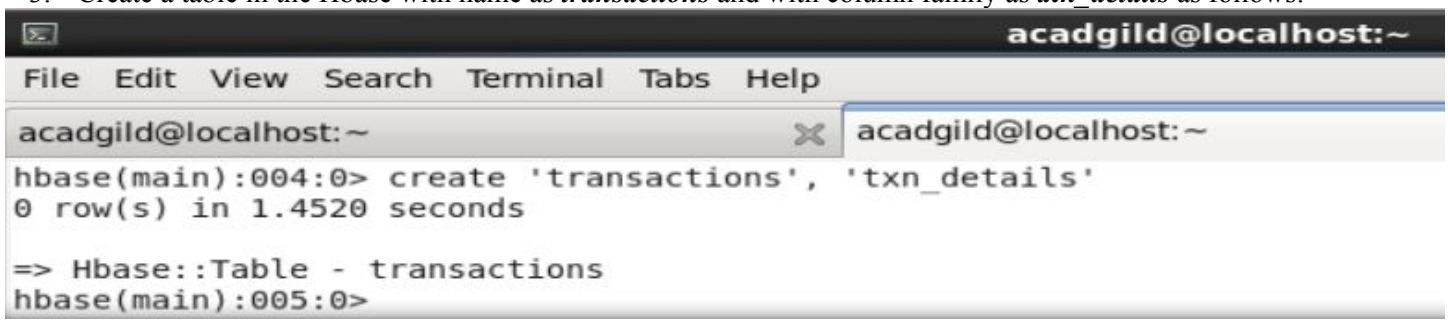
2. We can see the output of the query is populated into the table ***transactions_count*** table. And we displayed the table content using the query ***hive>select * from transactions_count***

Task 4:

Now lets make the TRANSACTIONS_COUNT table HBase complaint. In the sence, use Ser Des And Storage handler feature to hive to change the TRANSACTION_COUNT table to be able to create a TRANSACTIONS table in HBase.

1. Now take another terminal and start HBase components in the hadoop as follows:
   $start-hbase.sh
2. Now verity HMaster is started or not using ***jps*** command as follows:
   $jps

3. Create a table in the Hbase with name as ***transactions*** and with column family as ***txn_details*** as follows:



4. Create an External table in hive. We need to create an external table in Hive referring to HBase table. That can be created as shown below:

CREATE EXTERNAL TABLE transactions (
custid string COMMENT 'from deserializer',
fname string COMMENT 'from deserializer',
count string COMMENT 'from deserializer)

ROW FORMAT SERDE
'org.apache.hadoop.hive.hbase.HBaseSerDe'
STORED BY
'org.apache.hadoop.hive.bhase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
'hbase.columns.mapping'=':key,txn_details:fname,txn_details:count',
'serialization.format'='1')
TBLPROPERTIES (
'hbase.table.name'='transactions'
);

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~ | acadgild@localhost:~ | acadgild@localhost:~
hive> CREATE EXTERNAL TABLE transactions(
    >   custid string COMMENT 'from deserializer',
    >   fname string COMMENT 'from deserializer',
    >   count string COMMENT 'from deserializer')
    > ROW FORMAT SERDE
    >   'org.apache.hadoop.hive.hbase.HBaseSerDe'
    > STORED BY
    >   'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES (
    >   'hbase.columns.mapping'=':key,txn_details:fname,txn_details:count',
    >   'serialization.format'='1')
    > TBLPROPERTIES (
    >   'hbase.table.name'='transactions'
    > ) ;
OK
Time taken: 0.272 seconds
```

3. **Inserting Values into HBase Table Through Hive** for inserting data into the HBase table through Hive, you need to specify the HBase table name in the hive shell by using the below property before running the insert command.
   hive> set hbase.mapred.output.outputtable=transactions;

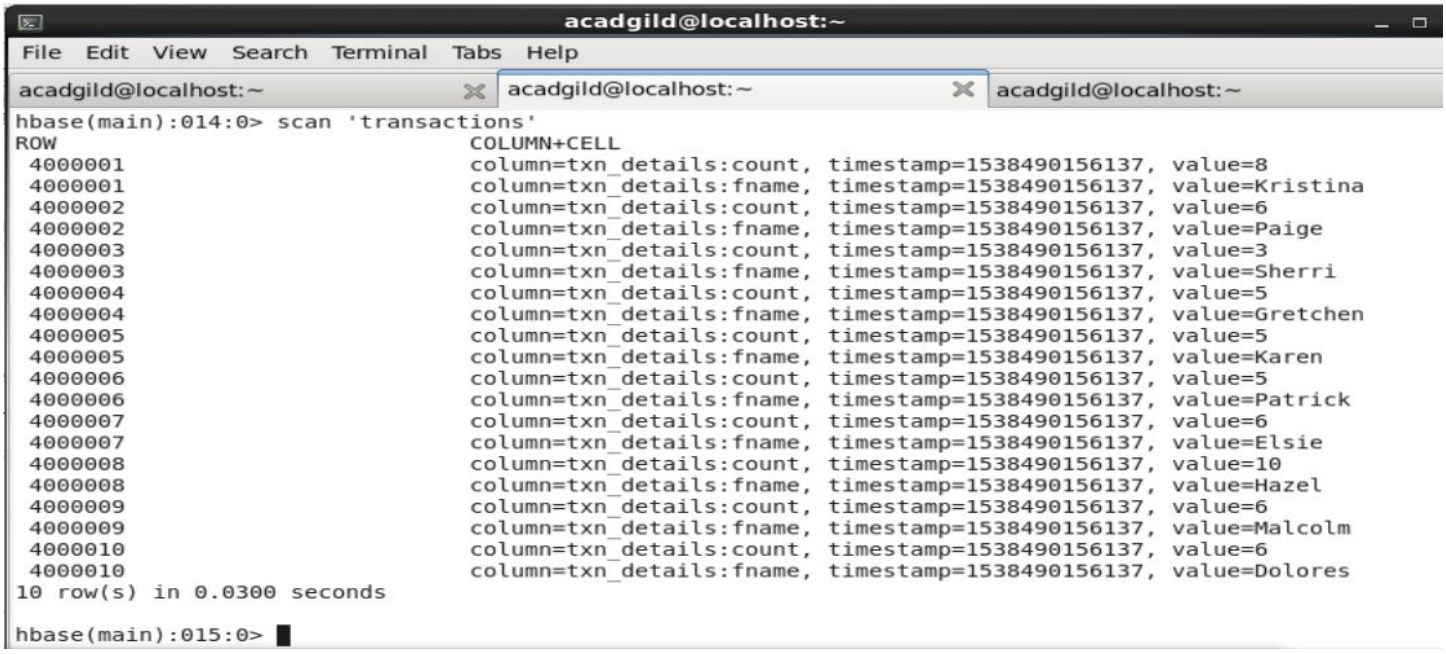
```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~ | acadgild@localhost:~
hive> set hbase.mapred.output.outputtable=transactions;
```

Task 5:
Now insert the data in TRANSACTIONS_COUNT table using the query in step 3 again, this should populate the HBase TRANSACTIONS table automatically (This has to be done in module 10).

1. Write a hive query to write the data populated from the table ***transactiosn_count*** into ***transactions*** table which is present in the HBase, as follows:
   hive>insert overwrite table transactions select * from transactions_count;

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~ | acadgild@localhost:~ | acadgild@localhost:~
hive> insert overwrite table transactions 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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181002195211_6be7d25a-b9a7-48e0-b102-0a75dfe1c1e3
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_1538487872665_0007, Tracking URL = http://localhost:8088/proxy/application_1538487872665_0007/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1538487872665_0007
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2018-10-02 19:52:24,926 Stage-3 map = 0%,  reduce = 0%
2018-10-02 19:52:36,908 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 6.52 sec
MapReduce Total cumulative CPU time: 6 seconds 520 msec
Ended Job = job_1538487872665_0007
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 6.52 sec   HDFS Read: 11234 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 520 msec
OK
Time taken: 26.943 seconds
hive>
```

2. We can see that the process ran successfully. Now we will check the table we already created in HBase, ***transactions.***
   hbase (main):014:0> scan 'transactions'



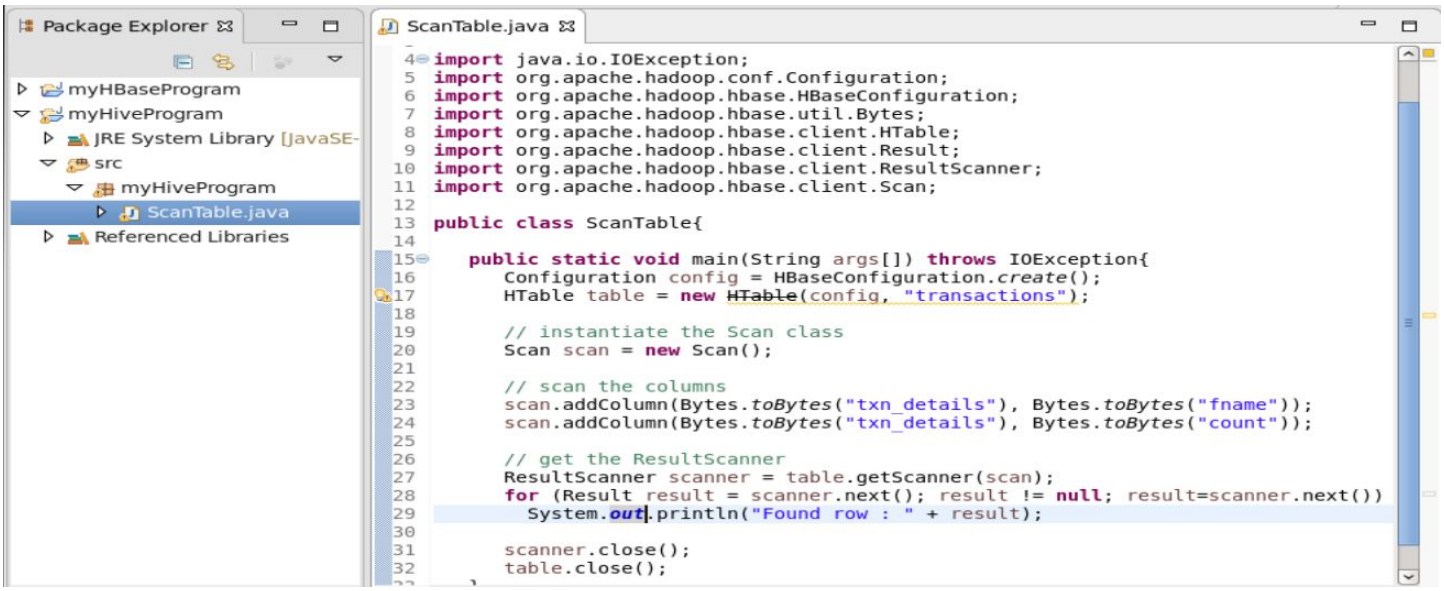
3. We can observe that the populated data from the above table contains all the details that are populated from Hive.

Task 6:
Now from the HBase level, write the HBase java API code to access and scan the TRANSACTIONS table data from java level.

1. Write Java Program for scan the TRANSACTIONS table as follows:

```
Package myHiveProgram;

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.HTables;
import org.apache.haboop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;

public class ScanTable {

    public static void main(String args[ ]) throuws IOExceptions {
            Configuration config = HBaseConfiguration.create();
    HTabe table =new HTable(config, “transactions”);
    //instantiate the Scan class
    Scan scan = new Scan();

    //scan the columns
    scan.addColumn(Bytes.toBytes(“txn_details”), Bytes.toBytes(“fname”));
    scan.addColumn(Bytes.toBytes(“txn_details”), Bytes.toBytes(“count”));

    //Get the ResultScanner
    ResultScanner scanner = table.getScanner(scan);
    for(Result result = scanner.next(); result != null; result=scanner.next())
            System.out.println(“Found row:” + result);

    scanner.close();
```

```
        table.close();
    }
}
```



2. When we run the above program, we will get the following output showing that the column family values, that is the customer id, customer count and its length, customer name and its length.

```
<terminated> ScanTable [Java Application] /usr/java/jdk1.8.0_151/bin/java (Oct 2, 2018, 9:49:23 PM)
keyvalues={4000001/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000001/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000002/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000002/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000003/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000003/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000004/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000004/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000005/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000005/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000006/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000006/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000007/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000007/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000008/txn_details:count/1538490156137/Put/vlen=2/seqid=0, 4000008/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000009/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000009/txn_details:fname/1538490156137/Put/vlen
keyvalues={4000010/txn_details:count/1538490156137/Put/vlen=1/seqid=0, 4000010/txn_details:fname/1538490156137/Put/vlen
```