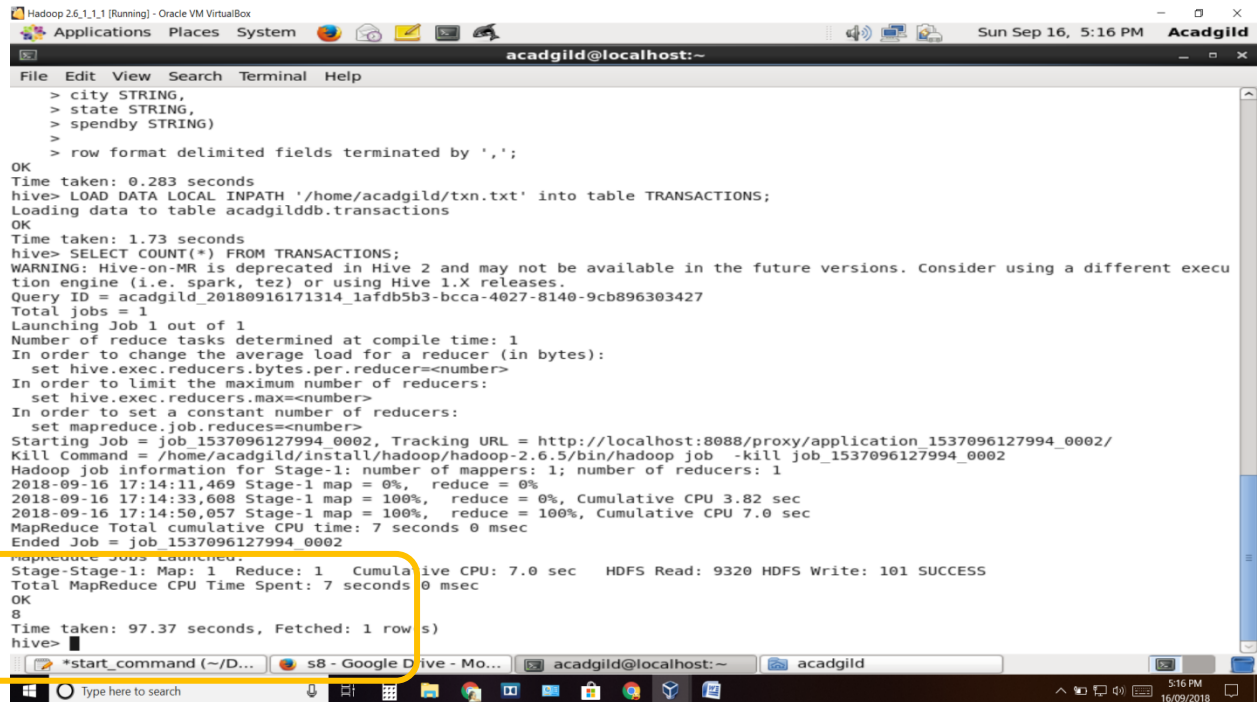


Big Data Hadoop 'Session 11: ADVANCE HBASE Assignment 2'

Tasks

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



```
Hadoop 2.6.1_1.1 [Running] - Oracle VM VirtualBox
Applications Places System
acadmild@localhost:~
File Edit View Search Terminal Help
> city STRING,
> state STRING,
> spendby STRING)
> row format delimited fields terminated by ',';
OK
Time taken: 0.283 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadmild/txn.txt' into table TRANSACTIONS;
Loading data to table acadmildb.transactions
OK
Time taken: 1.73 seconds
hive> SELECT COUNT(*) FROM TRANSACTIONS;
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 = acadmild_20180916171314_1afdb5b3-bcca-4027-8140-9cb896303427
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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_1537096127994_0002, Tracking URL = http://localhost:8088/proxy/application_1537096127994_0002/
Kill Command = /home/acadmild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1537096127994_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-16 17:14:11,469 Stage-1 map = 0%, reduce = 0%
2018-09-16 17:14:33,608 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.82 sec
2018-09-16 17:14:50,057 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.0 sec
MapReduce Total cumulative CPU time: 7 seconds 0 msec
Ended Job = job_1537096127994_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.0 sec HDFS Read: 9320 HDFS Write: 101 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 0 msec
OK
8
Time taken: 97.37 seconds, Fetched: 1 row(s)
hive>
```

2. Create a new table called TRANSACTIONS_COUNT. This table should have 3 fields - custid, fname and count. (Again to be done in module 8)

```
Hadoop 2.6.1_1.1 [Running] - Oracle VM VirtualBox
Applications Places System
acadmild@localhost:~
File Edit View Search Terminal Help
Time taken: 1.73 seconds
hive> SELECT COUNT(*) FROM TRANSACTIONS;
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 = acadmild_20180916171314_1afdb5b3-bcca-4027-8140-9cb896303427
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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_1537096127994_0002, Tracking URL = http://localhost:8088/proxy/application_1537096127994_0002/
Kill Command = /home/acadmild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1537096127994_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-16 17:14:11,469 Stage-1 map = 0%, reduce = 0%
2018-09-16 17:14:33,608 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.82 sec
2018-09-16 17:14:50,057 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.0 sec
MapReduce Total cumulative CPU time: 7 seconds 0 msec
Ended Job = job_1537096127994_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.0 sec HDFS Read: 9320 HDFS Write: 101 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 0 msec
OK
Time taken: 97.37 seconds, Fetched: 1 row(s)
hive> CREATE TABLE TRANSACTIONS_COUNT(
  > custid INT,
  > fname STRING,
  > count INT )
  > row format delimited fields terminated by ',';
OK
Time taken: 0.606 seconds
hive> select * from TRANSACTIONS_COUNT;
OK
Time taken: 0.619 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)

```
Hadoop 2.6.1_1.1 [Running] - Oracle VM VirtualBox
Applications Places System
acadmild@localhost:~
File Edit View Search Terminal Help
hive> describe customer
> ;
OK
custid          int
fname           string
lname           string
age             int
profession       string
Time taken: 0.523 seconds, Fetched: 5 row(s)
hive> describe transactions;
OK
txnno           int
txndate         string
custno          int
amount          double
category        string
product         string
city            string
state           string
spendby         string
Time taken: 0.27 seconds, Fetched: 9 row(s)
hive> INSERT INTO TRANSACTIONS_COUNT
  > SELECT t1.id, t1.f,COUNT(t1.txn) FROM
  > (SELECT c.custid as id, c.fname as f, t.txnno as txn
  > FROM customer c JOIN transactions t ON c.custid = t.custno)t1
  > GROUP BY t1.id, t1.f;
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 = acadmild_20180918050032_c9d4ac4d-ef92-48e2-97a2-f2fa9f98f0bf
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadmild/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/acadmild/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]
2018-09-18 05:01:11 Starting to launch local task to process map join; maximum memory = 518979584
2018-09-18 05:01:17 Dump the side-table for tag: 0 with group count: 8 into file: file:/tmp/acadmild/7692f44a-e89b-47bb-8
```

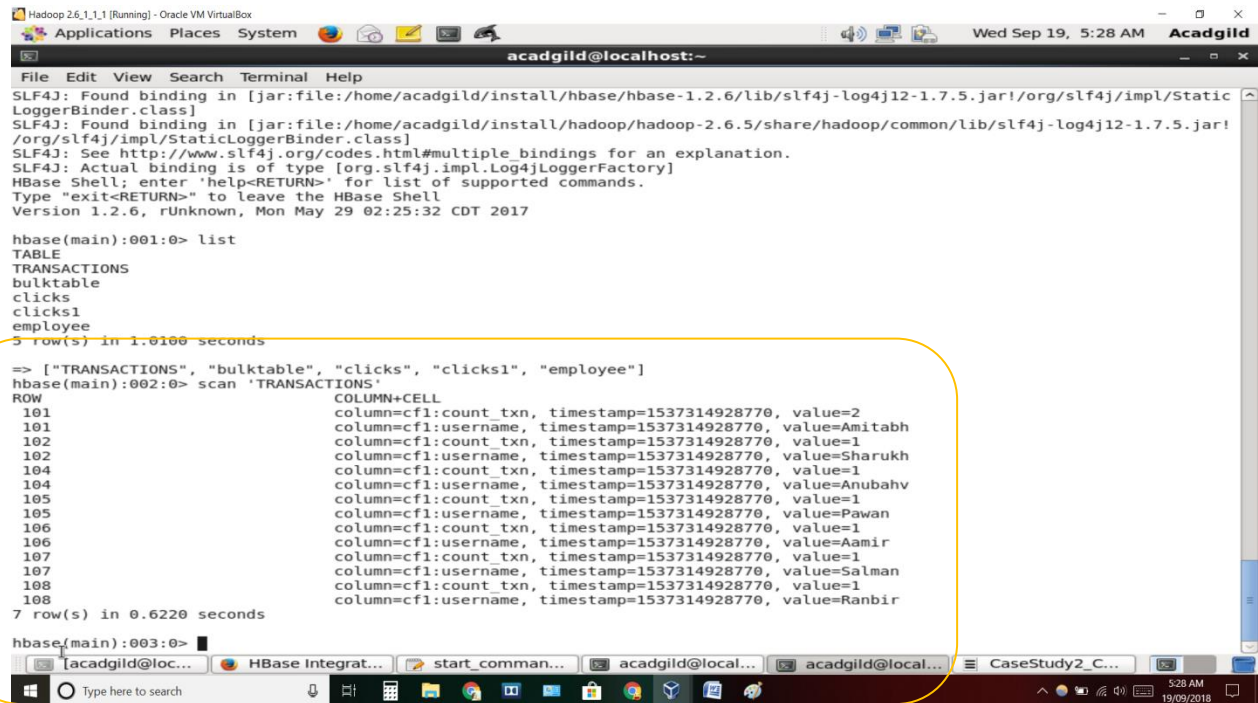
```
Hadoop 2.6.1_1.1 [Running] - Oracle VM VirtualBox
Applications Places System
acadgild@localhost:~
File Edit View Search Terminal Help
ble
2018-09-18 05:01:17 Uploaded 1 File to: file:/tmp/acadgild/7692f44a-e89b-47bb-87b6-ff2d1fc6e176/hive_2018-09-18_05-00-32_
216_8429270956180067724-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile00--.hashtable (469 bytes)
2018-09-18 05:01:17 End of local task; Time Taken: 5.779 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_1537226246851_0001, Tracking URL = http://localhost:8088/proxy/application_1537226246851_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1537226246851_0001
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-18 05:01:48,620 Stage-2 map = 0%, reduce = 0%
2018-09-18 05:02:10,392 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 4.43 sec
2018-09-18 05:02:27,863 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 8.83 sec
MapReduce Total cumulative CPU time: 8 seconds 830 msec
Ended Job = job_1537226246851_0001
Loading data to table acadgilddb.transactions_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 8.83 sec HDFS Read: 14092 HDFS Write: 177 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 830 msec
OK
Time taken: 120.018 seconds
hive> SELECT * FROM TRANSACTIONS_COUNT;
OK
101 Amitabh 2
102 Sharukh 1
104 Anubahv 1
105 Pawan 1
106 Aamir 1
107 Salman 1
108 Ranbir 1
Time taken: 0.414 seconds, Fetched: 7 row(s)
hive>
```

4. Now lets make the TRANSACTIONS_COUNT table Hbase complaint. In the sence, use Ser Des And Storage 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)

```
hive> CREATE TABLE TRANSACTIONS_hbase(id int, username string, count_txn string)
> STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
> WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cfl:username,cfl:count_txn")
> TBLPROPERTIES ("hbase.table.name" = "TRANSACTIONS");
OK
Time taken: 7.059 seconds
hive> show tables;
OK
customer
transactions
transactions_count
transactions_hbase
Time taken: 0.207 seconds, Fetched: 4 row(s)
hive>
```

Table in hbase

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)



```
Hadoop 2.6.1-1.1 [Running] - Oracle VM VirtualBox
Applications Places System
acadmild@localhost:~
File Edit View Search Terminal Help
SLF4J: Found binding in [jar:file:/home/acadmild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadmild/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.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

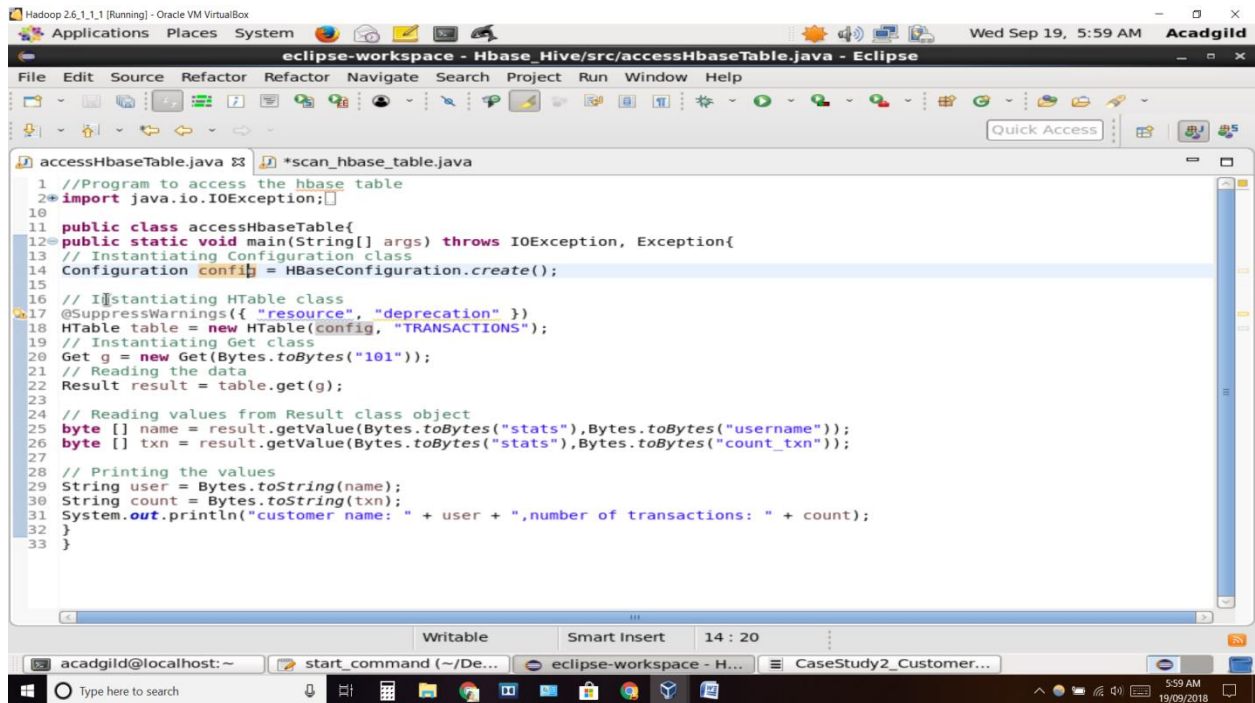
hbase(main):001:0> list
TABLE
TRANSACTIONS
bulktable
clicks
clicks1
employee
5 row(s) in 1.0100 seconds

=> ["TRANSACTIONS", "bulktable", "clicks", "clicks1", "employee"]
hbase(main):002:0> scan 'TRANSACTIONS'
ROW COLUMN+CELL
101 column=cf1:count_txn, timestamp=1537314928770, value=2
101 column=cf1:username, timestamp=1537314928770, value=Amitabh
102 column=cf1:count_txn, timestamp=1537314928770, value=1
102 column=cf1:username, timestamp=1537314928770, value=Sharukh
104 column=cf1:count_txn, timestamp=1537314928770, value=1
104 column=cf1:username, timestamp=1537314928770, value=Anubahv
105 column=cf1:count_txn, timestamp=1537314928770, value=1
105 column=cf1:username, timestamp=1537314928770, value=Pawan
106 column=cf1:count_txn, timestamp=1537314928770, value=1
106 column=cf1:username, timestamp=1537314928770, value=Aamir
107 column=cf1:count_txn, timestamp=1537314928770, value=1
107 column=cf1:username, timestamp=1537314928770, value=Salman
108 column=cf1:count_txn, timestamp=1537314928770, value=1
108 column=cf1:username, timestamp=1537314928770, value=Randhir
7 row(s) in 0.6220 seconds

hbase(main):003:0>
```

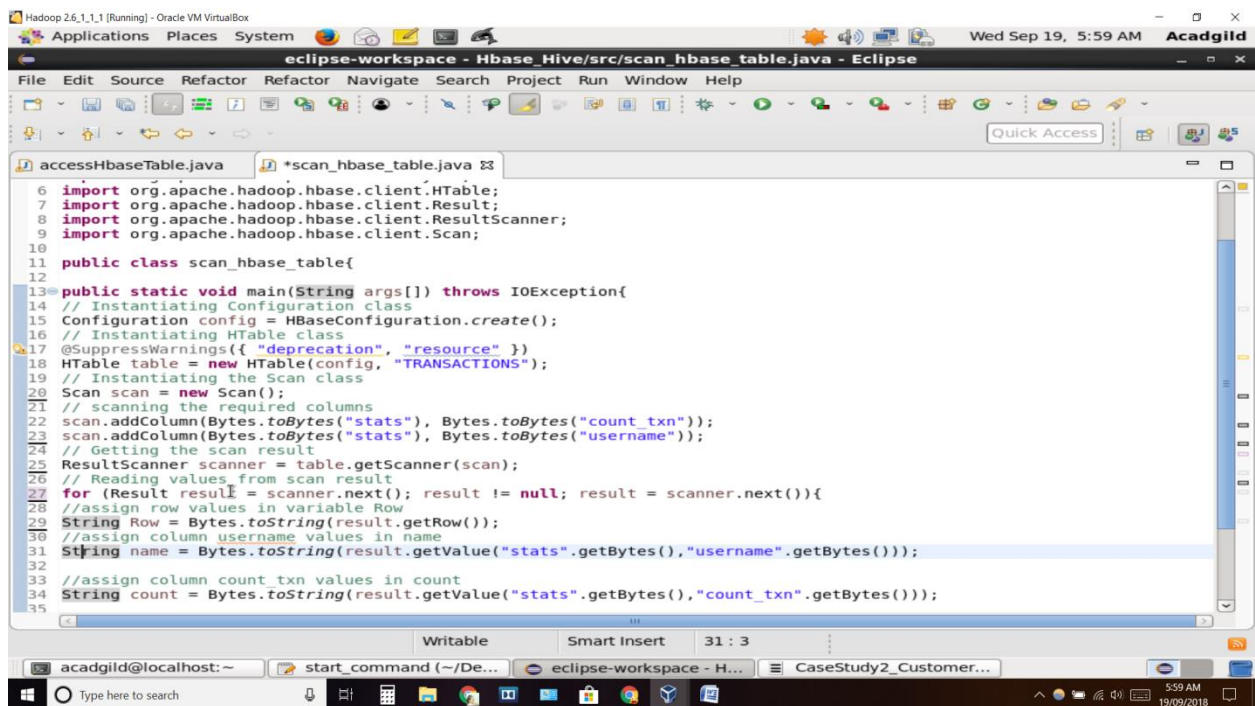
6. Now from the Hbase level, write the Hbase java API code to access and scan the TRANSACTIONS table data from java level.

Access Hbase table



```
1 //Program to access the hbase table
2 import java.io.IOException;
3
4 public class accessHbaseTable{
5     public static void main(String[] args) throws IOException, Exception{
6         // Instantiating Configuration class
7         Configuration config = HBaseConfiguration.create();
8
9         // Instantiating HTable class
10        @SuppressWarnings({ "resource", "deprecation" })
11        HTable table = new HTable(config, "TRANSACTIONS");
12        // Instantiating Get class
13        Get g = new Get(Bytes.toBytes("101"));
14        // Reading the data
15        Result result = table.get(g);
16
17        // Reading values from Result class object
18        byte [] name = result.getValue(Bytes.toBytes("stats"),Bytes.toBytes("username"));
19        byte [] txn = result.getValue(Bytes.toBytes("stats"),Bytes.toBytes("count_txn"));
20
21        // Printing the values
22        String user = Bytes.toString(name);
23        String count = Bytes.toString(txn);
24        System.out.println("customer name: " + user + ",number of transactions: " + count);
25    }
26 }
```

Scan Hbase table



```
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 public class scan_hbase_table{
12
13     public static void main(String args[]) throws IOException{
14         // Instantiating Configuration class
15         Configuration config = HBaseConfiguration.create();
16         // Instantiating HTable class
17         @SuppressWarnings({ "deprecation", "resource" })
18         HTable table = new HTable(config, "TRANSACTIONS");
19         // Instantiating the Scan class
20         Scan scan = new Scan();
21         // scanning the required columns
22         scan.addColumn(Bytes.toBytes("stats"), Bytes.toBytes("count_txn"));
23         scan.addColumn(Bytes.toBytes("stats"), Bytes.toBytes("username"));
24         // Getting the scan result
25         ResultScanner scanner = table.getScanner(scan);
26         // Reading values from scan result
27         for (Result result = scanner.next(); result != null; result = scanner.next()){
28             //assign row values in variable Row
29             String Row = Bytes.toString(result.getRow());
30             //assign column username values in name
31             String name = Bytes.toString(result.getValue("stats".getBytes(),"username".getBytes()));
32
33             //assign column count txn values in count
34             String count = Bytes.toString(result.getValue("stats".getBytes(),"count_txn".getBytes()));
35         }
36     }
37 }
```

```
101,Amitabh,2
102,Sharukh,1
104,Anubhav,1
105,Pawan,1
106,Aamir,1
107,Salman,1
108,Ranbir,1
```

