Case Study Hbase

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

```
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:

```
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:

```
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.

```
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;
OK
0       06-26-2011      4000001 40.33   Exercise & Fitness      Cardio Machine Accessories      C
larksville      Tennessee       credit
1       05-26-2011      4000002 198.44  Exercise & Fitness      Weightlifting Gloves    Long Beac
h       California      credit
2       06-01-2011      4000002 5.58    Exercise & Fitness      Weightlifting Machine Accessories
Anaheim California      credit
3       06-05-2011      4000003 198.19  Gymnastics      Gymnastics Rings        Milwaukee       W
isconsin        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  C
hicago  Illinois        credit
6       10-28-2011      4000005 27.89   Puzzles Jigsaw Puzzles  Charleston      South Carolina  c
redit
7       07-14-2011      4000006 96.01   Outdoor Play Equipment  Sandboxes       Columbus        O
hio     credit
8       01-17-2011      4000006 10.44   Winter Sports   Snowmobiling    Des Moines      Iowa    c
redit
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        O
hio     credit
12      02-08-2011      4000009 41.52   Indoor Games    Bowling San Francisco   California      c
redit
13      03-13-2011      4000010 107.8   Team Sports     Field Hockey    Honolulu        Hawaii  c
```

There are total 56 records in this table.

```
47      10-23-2011      4000008 100.1   Outdoor Play Equipment  Swing Sets      Everett Washingto
n       credit
48      09-27-2011      4000007 157.94  Exercise & Fitness      Exercise Bands  Philadelphia    P
ennsylvania     credit
49      07-12-2011      4000010 144.59  Jumping Jumping Stilts  Cambridge       Massachusetts   c
redit
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 c
ash
52      02-04-2011      4000005 44.82   Outdoor Play Equipment  Lawn Water Slides       Hampton
Virginia        cash
53      06-12-2011      4000004 44.46   Water Sports    Scuba Diving & Snorkeling       Charlesto
n       South Carolina  cash
54      10-03-2011      4000007 154.87  Outdoor Recreation      Running Long Beach      Californi
a       credit
55      12-16-2011      4000006 106.11  Water Sports    Swimming        New York        New Yorkc
redit
56      06-21-2011      4000002 176.63  Outdoor Recreation      Geocaching      Boston  Massachus
etts    credit
57      12-20-2011      4000003 178.2   Outdoor Recreation      Skating San Jose        Californi
a       credit
58      12-29-2011      4000002 194.86  Water Sports    Windsurfing     Oklahoma City   Oklahomac
redit
59      11-07-2011      4000001 21.43   Winter Sports   Snowboarding    Philadelphia    Pennsylva
nia     cash
Time taken: 0.408 seconds, Fetched: 60 row(s)
hive>
```

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.

```
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. Cons
ider 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_153
8453605204_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
```

```
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

```
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):
```

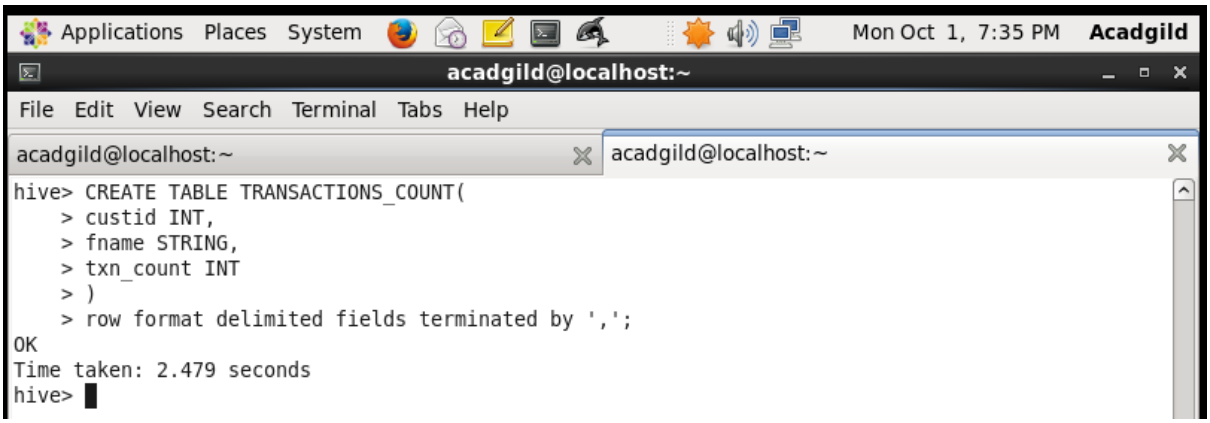
In above screen-shot we are able to see the output containing transaction done by each customer with their **first name** and **customer ID**.

```
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, Fet
hive>
```

eclipse-workspace - Hbase_Case_Study/src/com/Hbase/Access_table.java - Eclipse

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.



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**

```
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. Cons
ider 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-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]
2018-10-01 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-f17fa6ab91
04/hive_2018-10-01_19-37-47_947_1741443619391091517-1/-local-10003/HashTable-Stage-2/MapJoin-mapf
ile10--.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
ks not specified. Estimated from input data size: 1
```

```
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.

```
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.

```
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>
```

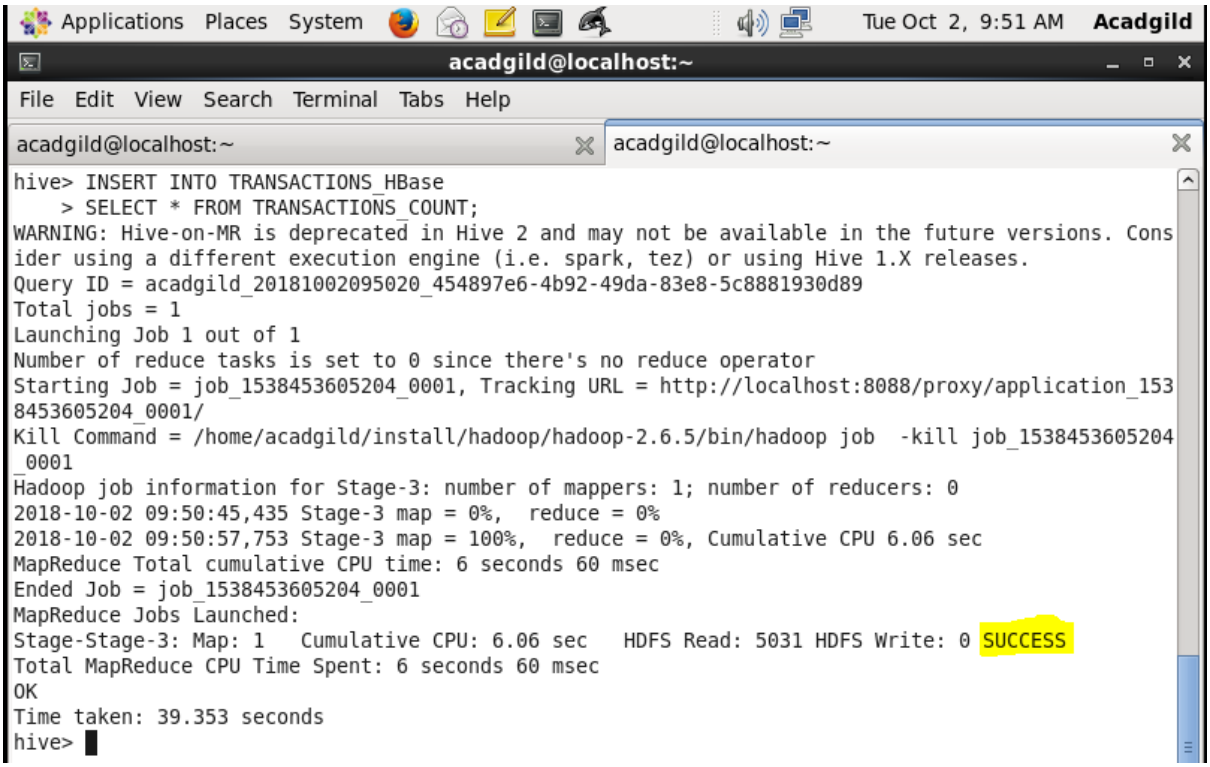
```
hbase(main):004:0> scan 'TRANSACTIONS'
ROW                          COLUMN+CELL
0 row(s) in 0.2080 seconds

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 successed.



Scan the table to see the contents of the table.

```
hbase(main):002:0> scan 'TRANSACTIONS'
ROW                          COLUMN+CELL
 4000001                     column=stats:count_txn, timestamp=1538454056936, value=8
 4000001                     column=stats:username, timestamp=1538454056936, value=Kristina
 4000002                     column=stats:count_txn, timestamp=1538454056936, value=6
 4000002                     column=stats:username, timestamp=1538454056936, value=Paige
 4000003                     column=stats:count_txn, timestamp=1538454056936, value=3
 4000003                     column=stats:username, timestamp=1538454056936, value=Sherri
 4000004                     column=stats:count_txn, timestamp=1538454056936, value=5
 4000004                     column=stats:username, timestamp=1538454056936, value=Gretchen
 4000005                     column=stats:count_txn, timestamp=1538454056936, value=5
 4000005                     column=stats:username, timestamp=1538454056936, value=Karen
 4000006                     column=stats:count_txn, timestamp=1538454056936, value=5
 4000006                     column=stats:username, timestamp=1538454056936, value=Patrick
 4000007                     column=stats:count_txn, timestamp=1538454056936, value=6
 4000007                     column=stats:username, timestamp=1538454056936, value=Elsie
 4000008                     column=stats:count_txn, timestamp=1538454056936, value=10
 4000008                     column=stats:username, timestamp=1538454056936, value=Hazel
 4000009                     column=stats:count_txn, timestamp=1538454056936, value=6
 4000009                     column=stats:username, timestamp=1538454056936, value=Malcolm
 4000010                     column=stats:count_txn, timestamp=1538454056936, value=6
 4000010                     column=stats:username, timestamp=1538454056936, value=Dolores
10 row(s) in 0.1380 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.

Java code to access Hbase table:

Data retrieved for customerId **'4000001'** by accessing the **Hbase** table using the below java api code.

Access_table.java | Scan_table.java

```java
package com.Hbase;
import java.io.IOException;
public class Access_table{
public static void main(String[] args) throws IOException, Exception{
// Instantiating Configuration class
Configuration config = HBaseConfiguration.create();
// Instantiating HTable clasls
@SuppressWarnings({ "resource", "deprecation" })
HTable table = new HTable(config, "TRANSACTIONS");
// Instantiating Get class
Get g = new Get(Bytes.toBytes("4000001"));
// 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: " + coun
```

Output of the data retrieved is as shown below.

Problems | Javadoc | Declaration | Console

```
<terminated> Access_table [Java Application] /usr/java/jdk1.8.0_151/bin/java (Oct 2, 2018, 6:26:00 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.
customer name: Kristina,number of transactions: 8
```

Java code to scan Hbase table:

Data retrieved from the **Hbase** table by **scanning** the table by using below java api code.

Access_table.java | Scan_table.java

```java
package com.Hbase;
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_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"));
```

Access_table.java | Scan_table.java

```java
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".getBy
//assign column count_txn values in count
String count = Bytes.toString(result.getValue("stats".getBytes(),"count_txn".get
System.out.println( Row + "," + name + "," + count );
//closing the scanner
scanner.close();
}
}}
```

Output of the above code is as shown below.

```
Problems  @ Javadoc  Declaration  Console
<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
```