Case Study 2 Hbase

We have two tables customer and txnrecords(I have created this table name when I was doing the assinment).

Customer has 10 records

Txnrecords have 60 records

```
hive> show databases;
OK
custom
default
simplidb
Time taken: 0.027 seconds, Fetched: 3 row(s)
hive> use simplidb;
OK
Time taken: 0.028 seconds
hive> show tables;
OK
customer
falt
family
mycustomer
mycustomer_ext
txnrecords
Time taken: 0.078 seconds, Fetched: 6 row(s)
hive> desc customer;
OK
custid                  int
fname                   string
lname                   string
age                     int
profession              string
Time taken: 0.11 seconds, Fetched: 5 row(s)
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: 0.203 seconds, Fetched: 10 row(s)
hive>
```

```
hive> desc txnrecords;
OK
txnno                   int
txndate                 string
custno                  int
amount                  double
category                string
product                 string
city                    string
state                   string
spendby                 string
Time taken: 0.08 seconds, Fetched: 9 row(s)
hive> select * from txnrecords;
OK
0       06-26-2011      4000001 40.33   Exercise & Fitness      Cardio Machine Accessories      Clarksville     Tennessee       credit
1       05-26-2011      4000002 198.44  Exercise & Fitness      Weightlifting Gloves    Long Beach      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       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        credit
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  credit
17      10-18-2011      4000008 75.55   Water Sports    Scuba Diving & Snorkeling       Omaha   Nebraska        credit
18      11-18-2011      4000008 88.65   Team Sports     Baseball        Salt Lake City  Utah    credit
19      08-28-2011      4000008 51.81   Water Sports    Life Jackets    Newark  New Jersey      credit
20      06-29-2011      4000005 41.55   Exercise & Fitness      Weightlifting Belts     New Orleans     Louisiana       credit
21      02-14-2011      4000005 45.79   Air Sports      Parachutes      New York        New York        credit
22      10-10-2011      4000009 19.64   Water Sports    Kitesurfing     Saint Paul      Minnesota       credit
23      05-02-2011      4000009 99.5    Gymnastics      Gymnastics Rings        Springfield     Illinois        credit
24      06-10-2011      4000003 151.2   Water Sports    Surfing Plano   Texas   credit
25      10-14-2011      4000009 144.2   Indoor Games    Darts   Phoenix Arizona credit
26      10-11-2011      4000009 31.58   Combat Sports   Wrestling       Orange  California      credit
27      09-29-2011      4000010 66.4    Games   Mahjong Fremont California      credit
28      05-12-2011      4000008 79.78   Team Sports     Cricket Lexington       Kentucky        credit
```

```
29      06-03-2011      4000001 126.9   Outdoor Recreation      Hunting Phoenix Arizona credit
30      03-14-2011      4000001 47.05   Water Sports    Swimming        Lincoln Nebraska        credit
31      11-28-2011      4000008 5.03    Games   Dice & Dice Sets        Los Angeles     California      credit
32      01-29-2011      4000008 20.13   Team Sports     Soccer  Springfield     Illinois        credit
33      06-15-2011      4000008 154.15  Outdoor Recreation      Lawn Games      Nashville       Tennessee       credit
34      05-06-2011      4000008 98.96   Team Sports     Indoor Volleyball       Atlanta Georgia credit
35      04-12-2011      4000008 185.26  Games   Board Games     Centennial      Colorado        credit
36      10-13-2011      4000007 35.66   Team Sports     Football        Saint Paul      Minnesota       credit
37      04-19-2011      4000007 20.2    Outdoor Recreation      Shooting Games  San Diego       California      credit
38      08-05-2011      4000007 150.6   Outdoor Recreation      Camping & Backpacking & Hiking  Hampton         Virginia        credit
39      03-12-2011      4000006 174.36  Outdoor Play Equipment  Swing Sets      Pittsburgh      Pennsylvania    credit
40      11-07-2011      4000005 165.1   Team Sports     Cheerleading    Reno    Nevada  credit
41      04-16-2011      4000004 28.11   Indoor Games    Bowling Westminster     Colorado        cash
42      09-10-2011      4000004 38.52   Outdoor Recreation      Tetherball      Denton  Texas   cash
43      04-22-2011      4000004 32.34   Water Sports    Water Polo      Las Vegas       Nevada  cash
44      09-11-2011      4000001 135.37  Water Sports    Surfing Seattle Washington      credit
45      11-27-2011      4000001 90.04   Exercise & Fitness      Abdominal Equipment     Honolulu        Hawaii  credit
46      05-27-2011      4000001 52.29   Gymnastics      Vaulting Horses Cleveland       Ohio    credit
47      10-23-2011      4000008 100.1   Outdoor Play Equipment  Swing Sets      Everett Washington      credit
48      09-27-2011      4000007 157.94  Exercise & Fitness      Exercise Bands  Philadelphia    Pennsylvania    credit
49      07-12-2011      4000010 144.59  Jumping Jumping Stilts  Cambridge       Massachusetts   credit
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 cash
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       Charleston      South Carolina  cash
54      10-03-2011      4000007 154.87  Outdoor Recreation      Running Long Beach      California      credit
55      12-16-2011      4000006 106.11  Water Sports    Swimming        New York        New York        credit
56      06-21-2011      4000002 176.63  Outdoor Recreation      Geocaching      Boston  Massachusetts   credit
57      12-20-2011      4000003 178.2   Outdoor Recreation      Skating San Jose        California      credit
58      12-29-2011      4000002 194.86  Water Sports    Windsurfing     Oklahoma City   Oklahoma        credit
59      11-07-2011      4000001 21.43   Winter Sports   Snowboarding    Philadelphia    Pennsylvania    cash
Time taken: 0.217 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)

Select custno, count(*) from txnrecords group by custno;

```
hive> Select custno, count(*)  from txnrecords 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 u
sing Hive 1.X releases.
Query ID = acadgild_20181208074445_c115a316-94ae-43b4-bd66-8459e2788690
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_1544146716043_0016, Tracking URL = http://localhost:8088/proxy/application_1544146716043_0016/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1544146716043_0016
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-12-08 07:45:00,934 Stage-1 map = 0%,  reduce = 0%
2018-12-08 07:45:10,995 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.06 sec
2018-12-08 07:45:20,796 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.47 sec
MapReduce Total cumulative CPU time: 4 seconds 470 msec
Ended Job = job_1544146716043_0016
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.47 sec   HDFS Read: 14628 HDFS Write: 308 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 470 msec
OK
4000001 8
4000002 6
4000003 3
4000004 5
4000005 5
4000006 5
4000007 6
4000008 10
4000009 6
4000010 6
Time taken: 37.202 seconds, Fetched: 10 row(s)
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)

For creating the table TRANSACTIONS_COUNT below query is used.

CREATE TABLE TRANSACTIONS_COUNT(

custid INT,

fname string,

txn_count INT

)

Row format delimited fields terminated by ‘,’;

```
hive> CREATE TABLE TRANSACTIONS_COUNT(
    > custid INT,
    > fname string,
    > txn_count INT
    > )
    > Row format delimited fields terminated by ',';
OK
Time taken: 0.431 seconds
hive> show tables;
OK
customer
falt
family
mycustomer
mycustomer_ext
transactions_count
txnrecords
Time taken: 0.092 seconds, Fetched: 7 row(s)
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).3

Insert into TRANSACTIONS_COUNT

Select c.custid,c.fname,count(t.custno) from customer c , txnrecords t where c.custid = t.custno group by c.custid, c.fname;

```
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: 0.243 seconds, Fetched: 10 row(s)
hive>
```

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)

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');

Hive shell

```
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: 4.99 seconds
hive>
```

Hbase shell

```
hbase(main):001:0> list
TABLE
SparkHBasesTable
TRANSACTIONS
bulktable
click
clicks
employee
htest
7 row(s) in 0.5670 seconds

=> ["SparkHBasesTable", "TRANSACTIONS", "bulktable", "click", "clicks", "employee", "htest"]
hbase(main):002:0> scan 'TRANSACTIONS'
ROW                                    COLUMN+CELL
0 row(s) in 0.2100 seconds

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

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)

*hive> INSERT into TRANSACTIONS_Hbase select * from transactions_count;*

Inserting into hive table

```
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: 4.531 seconds, Fetched: 10 row(s)
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 u
sing Hive 1.X releases.
Query ID = acadgild_20181208090033_9a3991a6-dcb8-4a14-a4d9-79642fde951d
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_1544146716043_0019, Tracking URL = http://localhost:8088/proxy/application_1544146716043_0019/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1544146716043_0019
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2018-12-08 09:01:07,826 Stage-3 map = 0%,  reduce = 0%
2018-12-08 09:01:25,082 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 4.9 sec
MapReduce Total cumulative CPU time: 5 seconds 350 msec
Ended Job = job_1544146716043_0019
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 5.35 sec   HDFS Read: 11350 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 350 msec
OK
Time taken: 54.857 seconds
hive>
```

Scanning the base transaction count

```
hbase(main):008:0> scan 'TRANSACTIONS'
ROW                                    COLUMN+CELL
 4000001                               column=stats:count_txn, timestamp=1544239885296, value=8
 4000001                               column=stats:username, timestamp=1544239885296, value=Kristina
 4000002                               column=stats:count_txn, timestamp=1544239885296, value=6
 4000002                               column=stats:username, timestamp=1544239885296, value=Paige
 4000003                               column=stats:count_txn, timestamp=1544239885296, value=3
 4000003                               column=stats:username, timestamp=1544239885296, value=Sherri
 4000004                               column=stats:count_txn, timestamp=1544239885296, value=5
 4000004                               column=stats:username, timestamp=1544239885296, value=Gretchen
 4000005                               column=stats:count_txn, timestamp=1544239885296, value=5
 4000005                               column=stats:username, timestamp=1544239885296, value=Karen
 4000006                               column=stats:count_txn, timestamp=1544239885296, value=5
 4000006                               column=stats:username, timestamp=1544239885296, value=Patrick
 4000007                               column=stats:count_txn, timestamp=1544239885296, value=6
 4000007                               column=stats:username, timestamp=1544239885296, value=Elsie
 4000008                               column=stats:count_txn, timestamp=1544239885296, value=10
 4000008                               column=stats:username, timestamp=1544239885296, value=Hazel
 4000009                               column=stats:count_txn, timestamp=1544239885296, value=6
 4000009                               column=stats:username, timestamp=1544239885296, value=Malcolm
 4000010                               column=stats:count_txn, timestamp=1544239885296, value=6
 4000010                               column=stats:username, timestamp=1544239885296, value=Dolores
10 row(s) in 0.2730 seconds

hbase(main):009:0>
```

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

Java API to access

Below code is to retrieve a row from habse table

```java
package com.Hbase;
import java.io.IOException;

public class accessTable {
    public static void main(String[] args) throws IOException, Exception{
    //Configuration Hbase
        Configuration conf = HBaseConfiguration.create();
    //Initiallizing htable class
    @SuppressWarnings({"resource", "deprecation"})
    HTable table = new HTable(conf,"TRANSACTIONS");
    //initializing the get class
    Get g = new Get(Bytes.toBytes("4000002"));
    //reading the data
    Result result = table.get(g);

    //reading values from result object

    byte [] name = result.getValue(Bytes.toBytes("stats"),Bytes.toBytes("username"));
    byte [] txn = result.getValue(Bytes.toBytes("stats"),Bytes.toBytes("count_txn"));
    //print values
    String user = Bytes.toString(name);
    String count = Bytes.toString(txn);
    System.out.println("customer name : " + user + "Number of transactions: " + count);


    }
    }
```

Console out put

```
<terminated> accessTable (1) [Java Application] /usr/java/jdk1.8.0_151/bin/java (Dec 10, 2018, 8:16:30 PM)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/spark/spark-2.2.1-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/i
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
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 : PaigeNumber of transactions: 6
```

Java API for scanning

Below code is for retrieving entire hbase table

scanTable.java | scanTable.java | accessTable.java

```java
import java.io.IOException;

public class scanTable {
    public static void main(String[] args) throws IOException,Exception {
    //Configuration Hbase
        Configuration conf = HBaseConfiguration.create();
    //Initiallizing htable class
    @SuppressWarnings({"deprecation","resource"})
    HTable table = new HTable(conf,"TRANSACTIONS");
    //initializing 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 result
    ResultScanner scanner = table.getScanner(scan);
    //reading from the scan results
    for(Result result = scanner.next();result != null; result = scanner.next()){
    //assign row values in variable row
    String row = Bytes.toString(result.getRow());
    //assign column username values in name
    String name = Bytes.toString(result.getValue("stats".getBytes(),"username".getBytes()));
    //assign column count_txn values in count
    String count = Bytes.toString(result.getValue("stats".getBytes(),"count_txn".getBytes()));
    System.out.println(row + "," + name + "," +  count );


    //closing the scanner
    scanner.close();
    }
```

Output console

```
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
```