Advance HBase Assignment

Tasks:

1. Find out the number of transactions by the customer (These should be taken up in module 8 itself)

```
acadgild@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/acadgild/txn.txt' into table TRANSACTIONS;
Loading data to table acadgilddb.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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_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/acadgild/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>
*start_command (~/D...   s8 - Google Drive - Mo...   acadgild@localhost:~   acadgild
```

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)

```
acadgild@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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_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/acadgild/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> 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>
*start_command (~/D...   Session11_Todo_Adva...   acadgild@localhost:~   acadgild
```

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 I module 9)

```
acadgild@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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180918050032_c9d4ac4d-ef92-48e2-97a2-f2fa9f98f0bf
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]
2018-09-18 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/acadgild/7692f44a-e89b-47bb-8
acadgild@localhost:~   *start_command (~/D...   acadgild@localhost:~   Re: [#95900] Big Data...
```

```
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 let's make the TRANSACTIONS_COUNT table HBase complaint. In the sense, use Ser Des and storage handler features of hive to change the TRANSACTIONS_COUNT table to be able to create 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,cf1:username,cf1: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>
[acadgild@loc...   HBase Integrat...   *start_comma...   acadgild@local...   acadgild@local...   CaseStudy2_C...
```

Table in hbase

```
acadgild@localhost:~
File  Edit  View  Search  Terminal  Help

=> ["bulktable", "clicks", "clicks1", "employee"]
hbase(main):003:0> list
TABLE
TRANSACTIONS
bulktable
clicks
clicks1
employee
5 row(s) in 0.0680 seconds

=> ["TRANSACTIONS", "bulktable", "clicks", "clicks1", "employee"]
hbase(main):004:0> describe 'TRANSACTIONS'
Table TRANSACTIONS is ENABLED
TRANSACTIONS
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEE
P_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '655
36', REPLICATION_SCOPE => '0'}
1 row(s) in 1.3430 seconds

hbase(main):005:0>
```

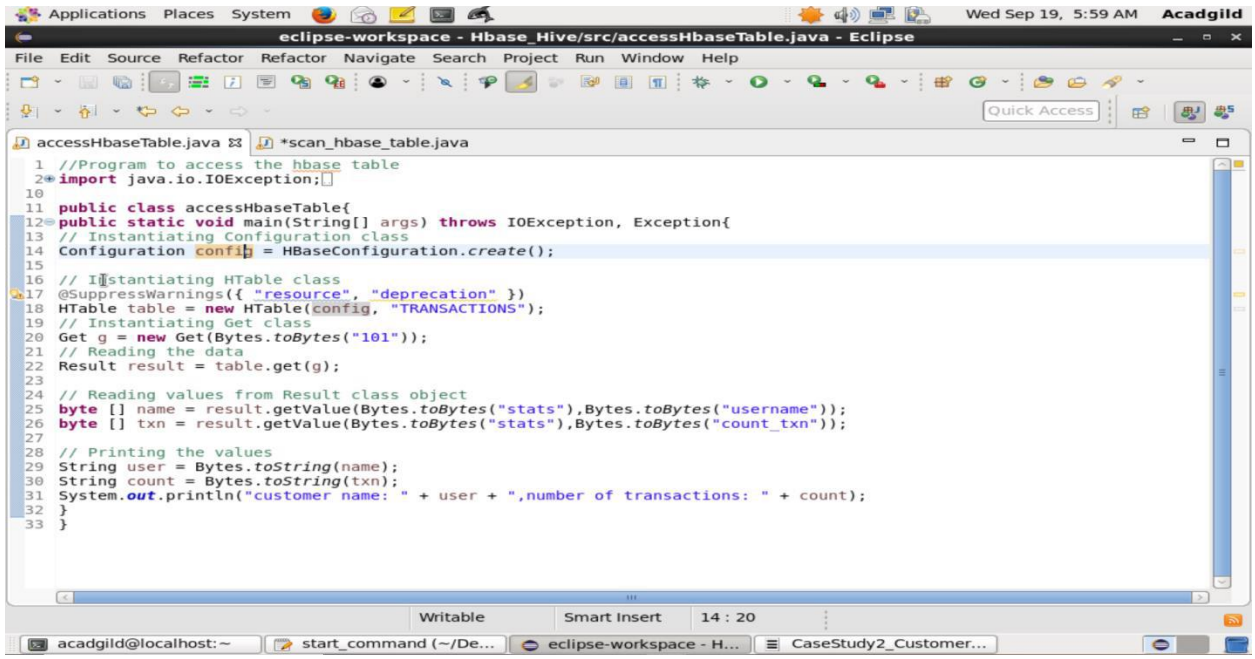
5. Now insert the data in TRANSACTIONS_COUNT table using the query in step 3 again, this should populate the Hbase TRANSACTIONS table automatically (This has to be done in module 10)

```
hive> show tables;
OK
customer
transactions
transactions_count
transactions_hbase
Time taken: 0.207 seconds, Fetched: 4 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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180919052433_91584203-4f83-4b3e-9b08-5fd9c4aad478
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_1537312595756_0002, Tracking URL = http://localhost:8088/proxy/application_1537312595756_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537312595756_0002
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2018-09-19 05:25:04,062 Stage-3 map = 0%,  reduce = 0%
2018-09-19 05:25:30,350 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 5.67 sec
MapReduce Total cumulative CPU time: 5 seconds 670 msec
Ended Job = job_1537312595756_0002
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 5.67 sec   HDFS Read: 11244 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 670 msec
OK
Time taken: 59.123 seconds
```

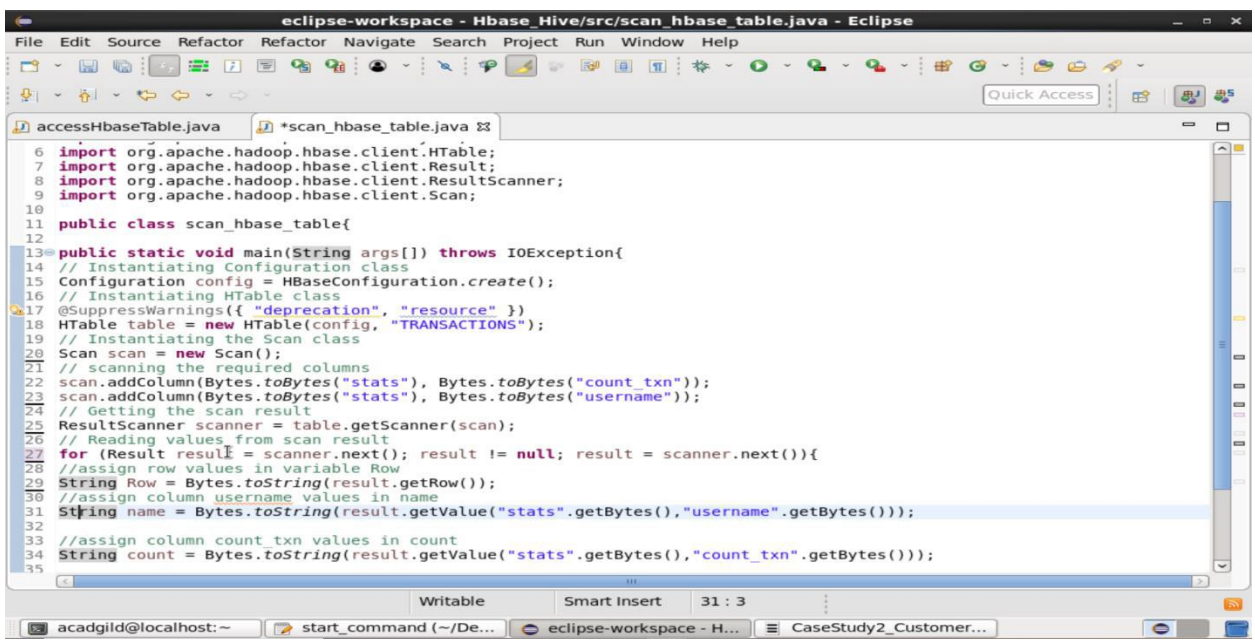
```
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.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=Ranbir
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



Scan Hbase table



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