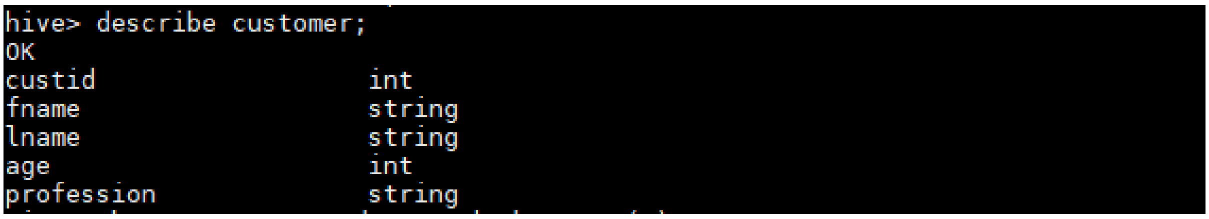
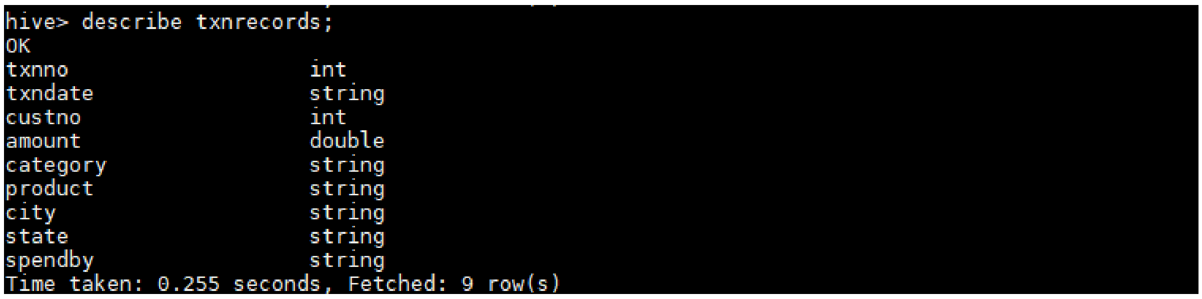
# Assignment Submission-Case Study 2(Customer Transaction)

Let us take up the CUSTOMER and TRANSACTIONS table we have created in the Let's Do Together section. Let us solve the following use cases using these tables :-

Tables created during sessions: **customer, txnrecords**



```
hive> describe customer;
OK
custid                  int
fname                   string
lname                   string
age                     int
profession              string
```

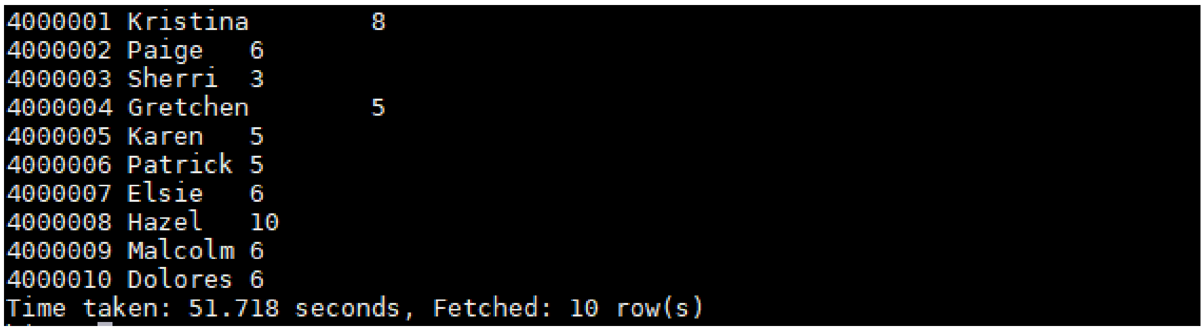


```
hive> describe txnrecords;
OK
txnno                   int
txndate                 string
custno                  int
amount                  double
category                string
product                 string
city                    string
state                   string
spendby                 string
Time taken: 0.255 seconds, Fetched: 9 row(s)
```

1. Find out the number of transaction done by each customer.

***Hive query:***

select a.custid, a.fname, count(a.fname) from CUSTOMER a join TXNRECORDS b on a.custid =b.custno group by a.fname, a.custid;

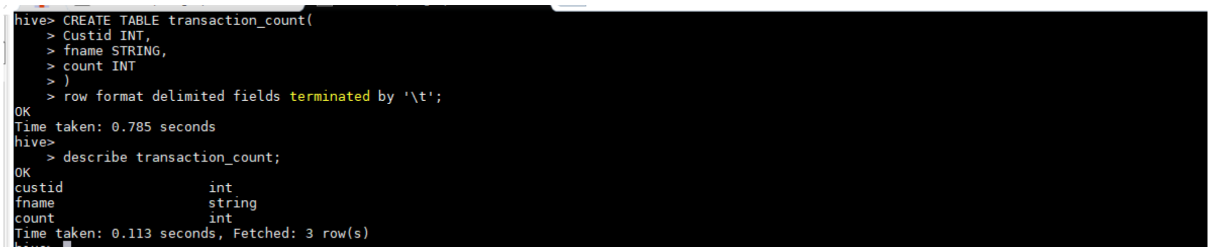


```
4000001 Kristina        8
4000002 Paige   6
4000003 Sherri  3
4000004 Gretchen        5
4000005 Karen   5
4000006 Patrick 5
4000007 Elsie   6
4000008 Hazel   10
4000009 Malcolm 6
4000010 Dolores 6
Time taken: 51.718 seconds, Fetched: 10 row(s)
```

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

***Hive query:***

CREATE TABLE transaction_count(
Custid INT,
fname STRING,
count INT
)
row format delimited fields terminated by '\t';

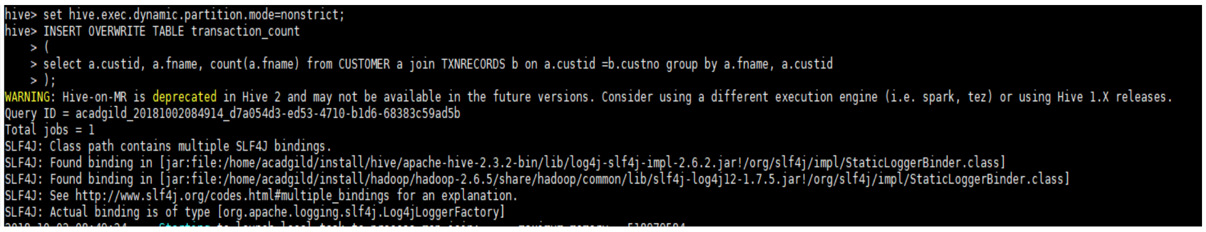


```
hive> CREATE TABLE transaction_count(
    > Custid INT,
    > fname STRING,
    > count INT
    > )
    > row format delimited fields terminated by '\t';
OK
Time taken: 0.785 seconds
hive>
    > describe transaction_count;
OK
custid                  int
fname                   string
count                   int
Time taken: 0.113 seconds, Fetched: 3 row(s)
```

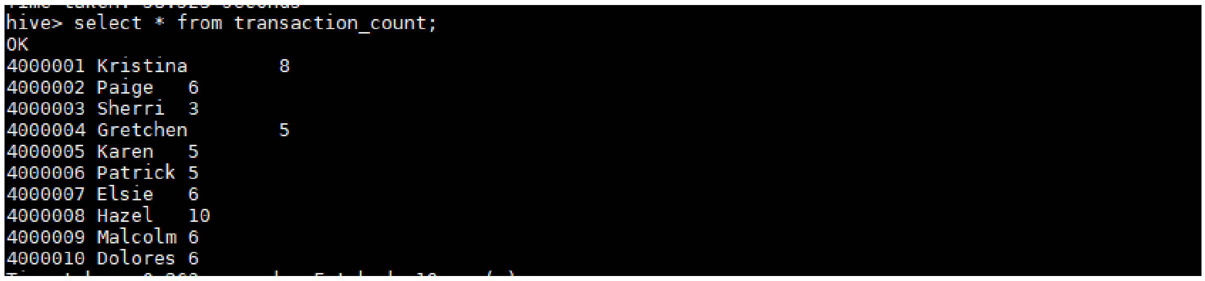
3. Now write a hive query in such a way that the query populates the data obtained in Step 1 above and populate the table in step 2 above.

***Hive query:***

set hive.exec.dynamic.partition.mode=nonstrict;
INSERT OVERWRITE TABLE transaction_count
(
select a.custid, a.fname, count(a.fname) from CUSTOMER a join TXNRECORDS b on a.custid =b.custno group by a.fname, a.custid
);



```
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> INSERT OVERWRITE TABLE transaction_count
    > (
    > select a.custid, a.fname, count(a.fname) from CUSTOMER a join TXNRECORDS b on a.custid =b.custno group by a.fname, a.custid
    > );
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 = acadgild_20181002084914_d7a054d3-ed53-4710-b1d6-68383c59ad5b
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]
```



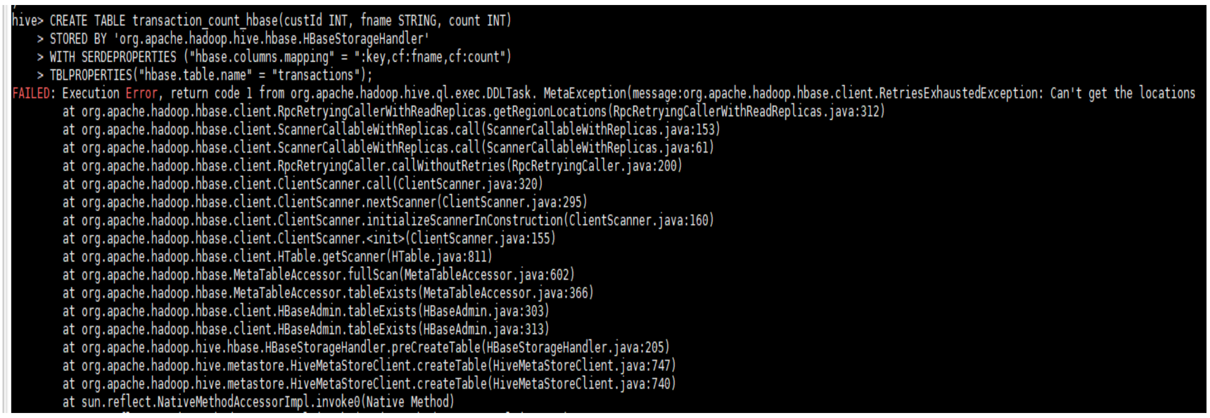
```
hive> select * from transaction_count;
OK
4000001 Kristina        8
4000002 Paige   6
4000003 Sherri  3
4000004 Gretchen        5
4000005 Karen   5
4000006 Patrick 5
4000007 Elsie   6
4000008 Hazel   10
4000009 Malcolm 6
4000010 Dolores 6
```

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 a TRANSACTIONS table in Hbase.

***Hive query:***

CREATE TABLE transaction_count_hbase(custId INT, fname STRING, count INT)
STORED BY ‘org.apache.hadoop.hive.hbase.HBaseStorageHandler’
WITH SERDEPROPERTIES (“hbase.columns.mapping” = “:key,f:fname,f:count”)
TBLPROPERTIES(“hbase.table.name” = “transactions”);

**Couldn’t run the command due to issue on my VM:**



```
hive> CREATE TABLE transaction_count_hbase(custId INT, fname STRING, count INT)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cf:fname,cf:count")
    > TBLPROPERTIES("hbase.table.name" = "transactions");
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:org.apache.hadoop.hbase.client.RetriesExhaustedException: Can't get the locations
        at org.apache.hadoop.hbase.client.RpcRetryingCallerWithReadReplicas.getRegionLocations(RpcRetryingCallerWithReadReplicas.java:312)
        at org.apache.hadoop.hbase.client.ScannerCallableWithReplicas.call(ScannerCallableWithReplicas.java:153)
        at org.apache.hadoop.hbase.client.ScannerCallableWithReplicas.call(ScannerCallableWithReplicas.java:61)
        at org.apache.hadoop.hbase.client.RpcRetryingCaller.callWithoutRetries(RpcRetryingCaller.java:200)
        at org.apache.hadoop.hbase.client.ClientScanner.call(ClientScanner.java:320)
        at org.apache.hadoop.hbase.client.ClientScanner.nextScanner(ClientScanner.java:295)
        at org.apache.hadoop.hbase.client.ClientScanner.initializeScannerInConstruction(ClientScanner.java:160)
        at org.apache.hadoop.hbase.client.ClientScanner.<init>(ClientScanner.java:155)
        at org.apache.hadoop.hbase.client.HTable.getScanner(HTable.java:811)
        at org.apache.hadoop.hbase.MetaTableAccessor.fullScan(MetaTableAccessor.java:602)
        at org.apache.hadoop.hbase.MetaTableAccessor.tableExists(MetaTableAccessor.java:366)
        at org.apache.hadoop.hbase.client.HBaseAdmin.tableExists(HBaseAdmin.java:303)
        at org.apache.hadoop.hbase.client.HBaseAdmin.tableExists(HBaseAdmin.java:313)
        at org.apache.hadoop.hive.hbase.HBaseStorageHandler.preCreateTable(HBaseStorageHandler.java:205)
        at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.createTable(HiveMetaStoreClient.java:747)
        at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.createTable(HiveMetaStoreClient.java:740)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

5. Now insert the data in TRANSACTIONS_COUNT table using the query in step 3 again, this should populate the Hbase TRANSACTIONS table automatically.

***Hive query:***

INSERT OVERWRITE transaction_count_hbase
(
select a.custid, a.fname, count(a.fname) from CUSTOMER a join TXNRECORDS b on a.custid =b.custno group by a.fname, a.custid
);

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