Case Study II for Session 8,9,10 11 (for HIVE & HBASE)

- Prachi Mohite

HIVE

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Hive is not

- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Solution Approach –

To execute the HIVE commands we are using HIVE Command line. It has two modes of interaction

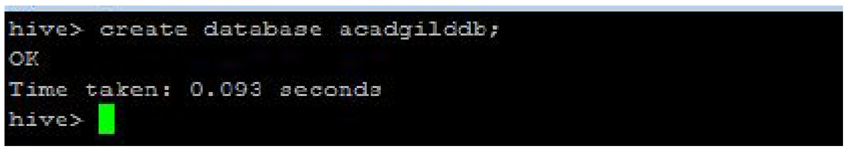
1. **Interactive Mode**
   a. Here we can submit the actual hive commands (queries) on HIVE CLI directly
2. **Non Interactive Mode**
   a. Here we need to execute the HIVE script
   b. e.g HIVE –f name_of_script.q

In this case study we will be using Interactive Mode (through Hive Shell)

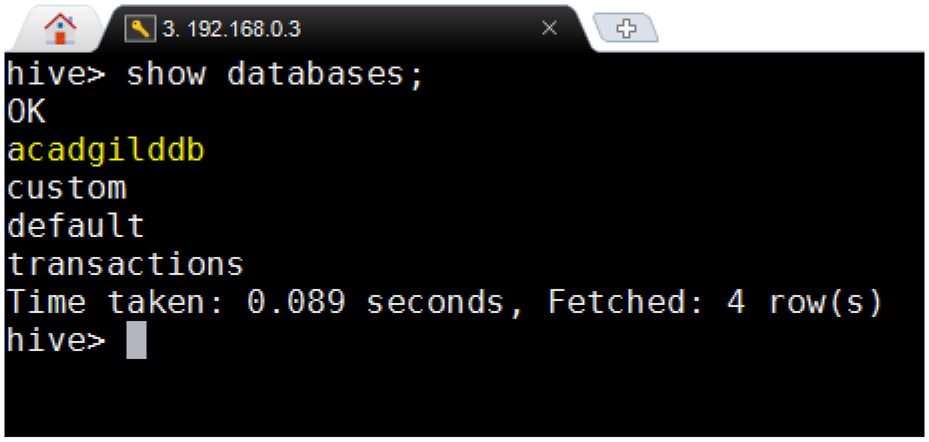
Let us take up the CUSTOMER and TRANSACTIONS table we have created in the Let's Do Together section.

As per prerequisite we should have database and tables created within it as described in to do of session 8

1. Creating Database

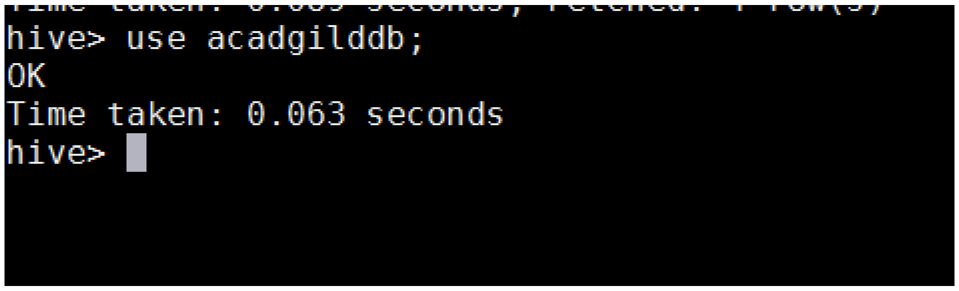


```
hive> create database acadgilddb;
OK
Time taken: 0.093 seconds
hive>
```



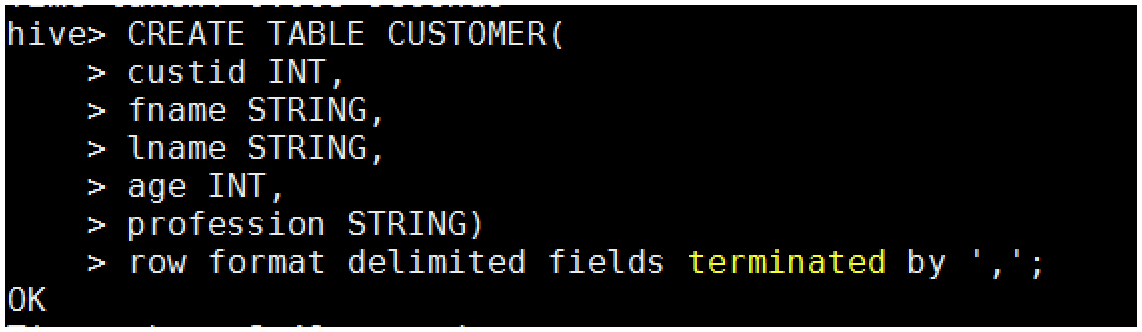
```
hive> show databases;
OK
acadgilddb
custom
default
transactions
Time taken: 0.089 seconds, Fetched: 4 row(s)
hive>
```

2. Making acadgilddb as active db for next query execution



```
hive> use acadgilddb;
OK
Time taken: 0.063 seconds
hive>
```

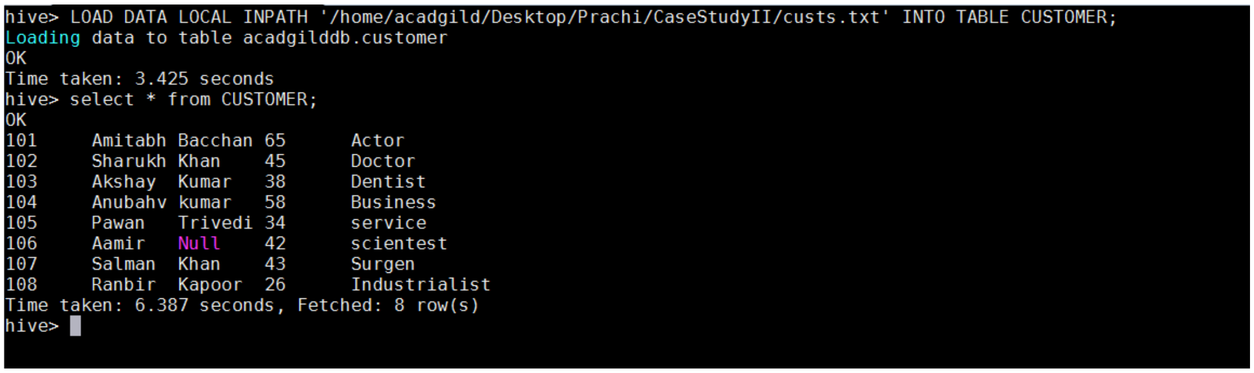
3. Creating tables
   a. Now create an internal table by the name customer



```
hive> CREATE TABLE CUSTOMER(
    > custid INT,
    > fname STRING,
    > lname STRING,
    > age INT,
    > profession STRING)
    > row format delimited fields terminated by ',';
OK
```

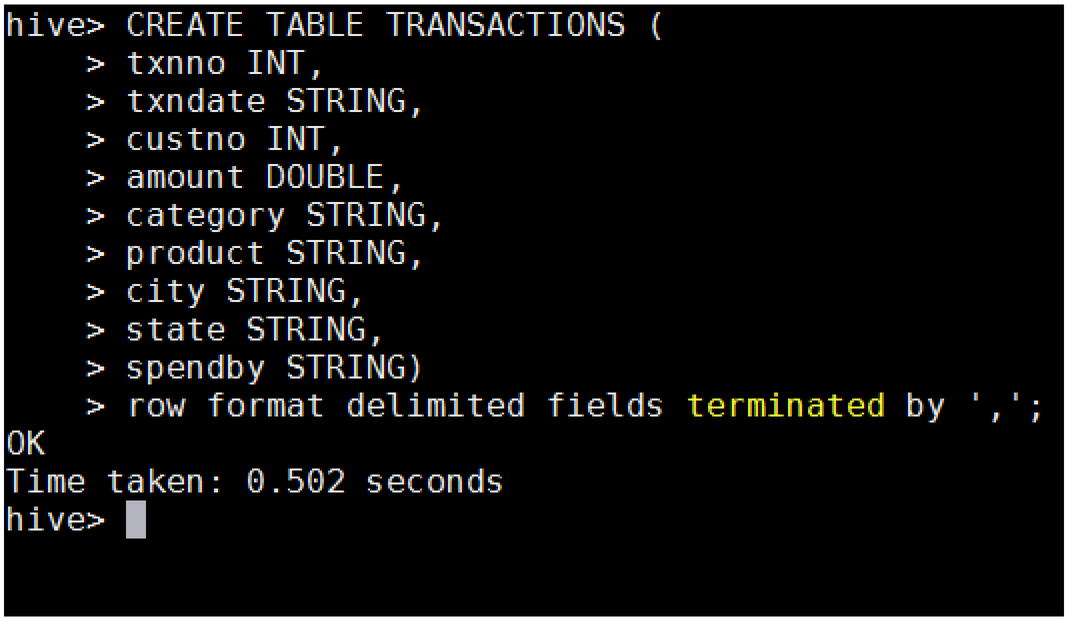
   b. Load data into Customer Table

LOAD DATA LOCAL INPATH
'home/acadgild/Desktop/Prachi/CaseStudyII/custs.txt'
into table CUSTOMER;



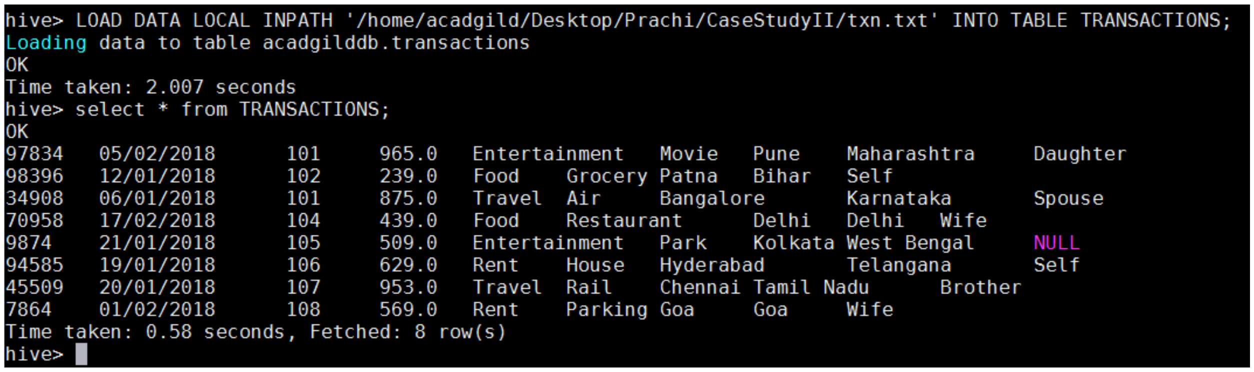
```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Desktop/Prachi/CaseStudyII/custs.txt' INTO TABLE CUSTOMER;
Loading data to table acadgilddb.customer
OK
Time taken: 3.425 seconds
hive> select * from CUSTOMER;
OK
101     Amitabh Bacchan 65      Actor
102     Sharukh Khan    45      Doctor
103     Akshay  Kumar   38      Dentist
104     Anubahv kumar   58      Business
105     Pawan   Trivedi 34      service
106     Aamir   Null    42      scientest
107     Salman  Khan    43      Surgen
108     Ranbir  Kapoor  26      Industrialist
Time taken: 6.387 seconds, Fetched: 8 row(s)
hive>
```

4. Create table Transactions



```
hive> CREATE TABLE TRANSACTIONS (
    > txnno INT,
    > txndate STRING,
    > custno INT,
    > amount DOUBLE,
    > category STRING,
    > product STRING,
    > city STRING,
    > state STRING,
    > spendby STRING)
    > row format delimited fields terminated by ',';
OK
Time taken: 0.502 seconds
hive>
```

Load data into transactions table



```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Desktop/Prachi/CaseStudyII/txn.txt' INTO TABLE TRANSACTIONS;
Loading data to table acadgilddb.transactions
OK
Time taken: 2.007 seconds
hive> select * from TRANSACTIONS;
OK
97834   05/02/2018      101     965.0   Entertainment   Movie   Pune    Maharashtra     Daughter
98396   12/01/2018      102     239.0   Food    Grocery Patna   Bihar   Self
34908   06/01/2018      101     875.0   Travel  Air     Bangalore       Karnataka       Spouse
70958   17/02/2018      104     439.0   Food    Restaurant      Delhi   Delhi   Wife
9874    21/01/2018      105     509.0   Entertainment   Park    Kolkata West Bengal     NULL
94585   19/01/2018      106     629.0   Rent    House   Hyderabad       Telangana       Self
45509   20/01/2018      107     953.0   Travel  Rail    Chennai Tamil Nadu      Brother
7864    01/02/2018      108     569.0   Rent    Parking Goa     Goa     Wife
Time taken: 0.58 seconds, Fetched: 8 row(s)
hive>
```

Task 1

Find out the number of transaction done by each customer

Solution Approach

As we have to find customers having transactions, we need **to INNER JOIN transactions** table with **CUSTOMER** table on **customer id**.

INNER JOIN:

The INNER JOIN in Hive uses JOIN keywords, which return rows meeting the JOIN conditions from both left and right tables.

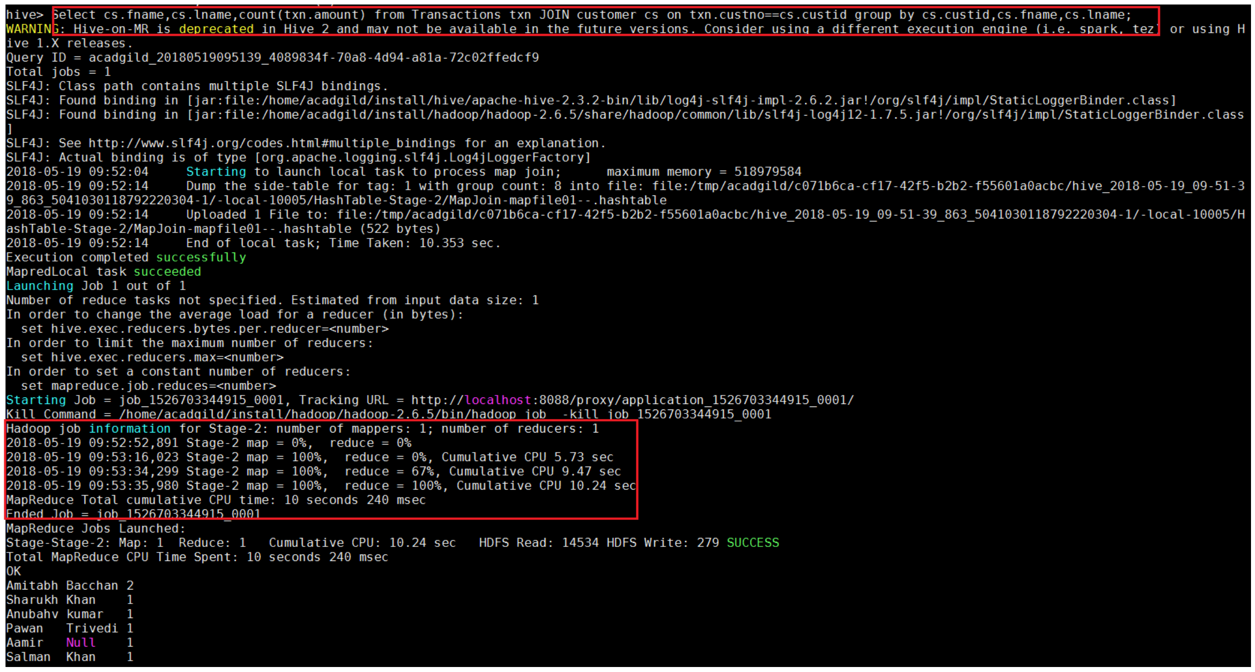
Group By Clause

This chapter explains the details of GROUP BY clause in a SELECT statement. The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

Command

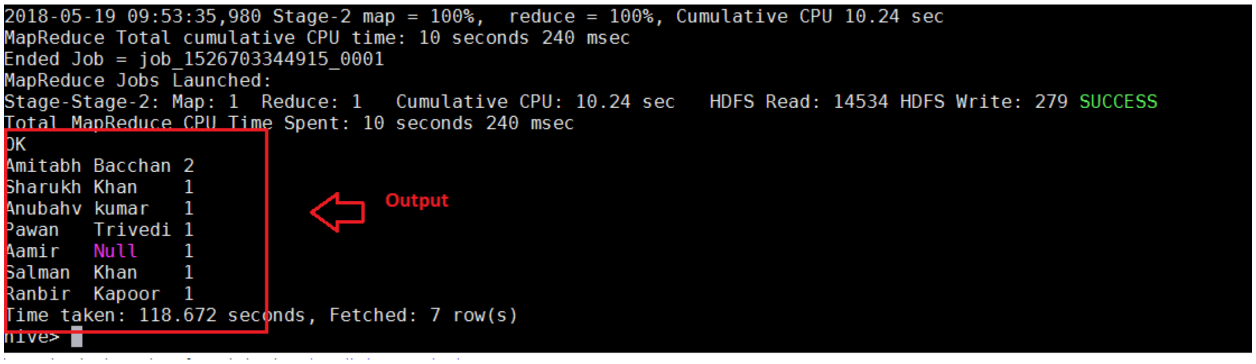
Select cs.fname,cs.lname,count(txn.amount) from Transactions txn JOIN customer cs on txn.custno==cs.custid group by cs.custid.cs.fname,cs.lname;

Execution of Command



```
hive> Select cs.fname,cs.lname,count(txn.amount) from Transactions txn JOIN customer cs on txn.custno==cs.custid group by cs.custid,cs.fname,cs.lname;
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 H
ive 1.X releases.
Query ID = acadgild_20180519095139_4089834f-70a8-4d94-a81a-72c02ffedcf9
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]
2018-05-19 09:52:04     Starting to launch local task to process map join;      maximum memory = 518979584
2018-05-19 09:52:14     Dump the side-table for tag: 1 with group count: 8 into file: file:/tmp/acadgild/c071b6ca-cf17-42f5-b2b2-f55601a0acbc/hive_2018-05-19_09-51-3
9_863_5041030118792220304-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile01--.hashtable
2018-05-19 09:52:14     Uploaded 1 File to: file:/tmp/acadgild/c071b6ca-cf17-42f5-b2b2-f55601a0acbc/hive_2018-05-19_09-51-39_863_5041030118792220304-1/-local-10005/H
ashTable-Stage-2/MapJoin-mapfile01--.hashtable (522 bytes)
2018-05-19 09:52:14     End of local task; Time Taken: 10.353 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_1526703344915_0001, Tracking URL = http://localhost:8088/proxy/application_1526703344915_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1526703344915_0001
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-05-19 09:52:52,891 Stage-2 map = 0%,  reduce = 0%
2018-05-19 09:53:16,023 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 5.73 sec
2018-05-19 09:53:34,299 Stage-2 map = 100%,  reduce = 67%, Cumulative CPU 9.47 sec
2018-05-19 09:53:35,980 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 10.24 sec
MapReduce Total cumulative CPU time: 10 seconds 240 msec
Ended Job = job_1526703344915_0001
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 10.24 sec   HDFS Read: 14534 HDFS Write: 279 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 240 msec
OK
Amitabh Bacchan 2
Sharukh Khan    1
Anubahv kumar   1
Pawan   Trivedi 1
Aamir   Null    1
Salman  Khan    1
```

Output


```
2018-05-19 09:53:35,980 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 10.24 sec
MapReduce Total cumulative CPU time: 10 seconds 240 msec
Ended Job = job_1526703344915_0001
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 10.24 sec   HDFS Read: 14534 HDFS Write: 279 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 240 msec
OK
Amitabh Bacchan 2
Sharukh Khan    1
Anubahv kumar   1
Pawan   Trivedi 1
Aamir   Null    1
Salman  Khan    1
Ranbir  Kapoor  1
Time taken: 118.672 seconds, Fetched: 7 row(s)
hive>
```

Task 2

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

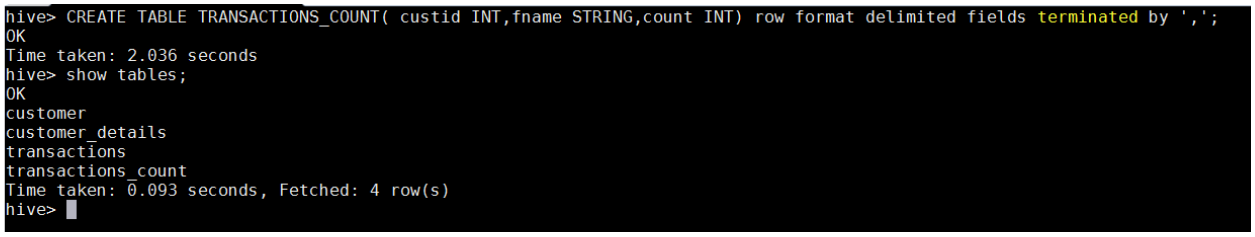
CREATE TABLE TRANSACTIONS_COUNT(

custid INT,

Fname STRING,

Count INT) row format delimited fields terminated by ‘,’;

Execution and Output


```
hive> CREATE TABLE TRANSACTIONS_COUNT( custid INT,fname STRING,count INT) row format delimited fields terminated by ',';
OK
Time taken: 2.036 seconds
hive> show tables;
OK
customer
customer_details
transactions
transactions_count
Time taken: 0.093 seconds, Fetched: 4 row(s)
hive>
```

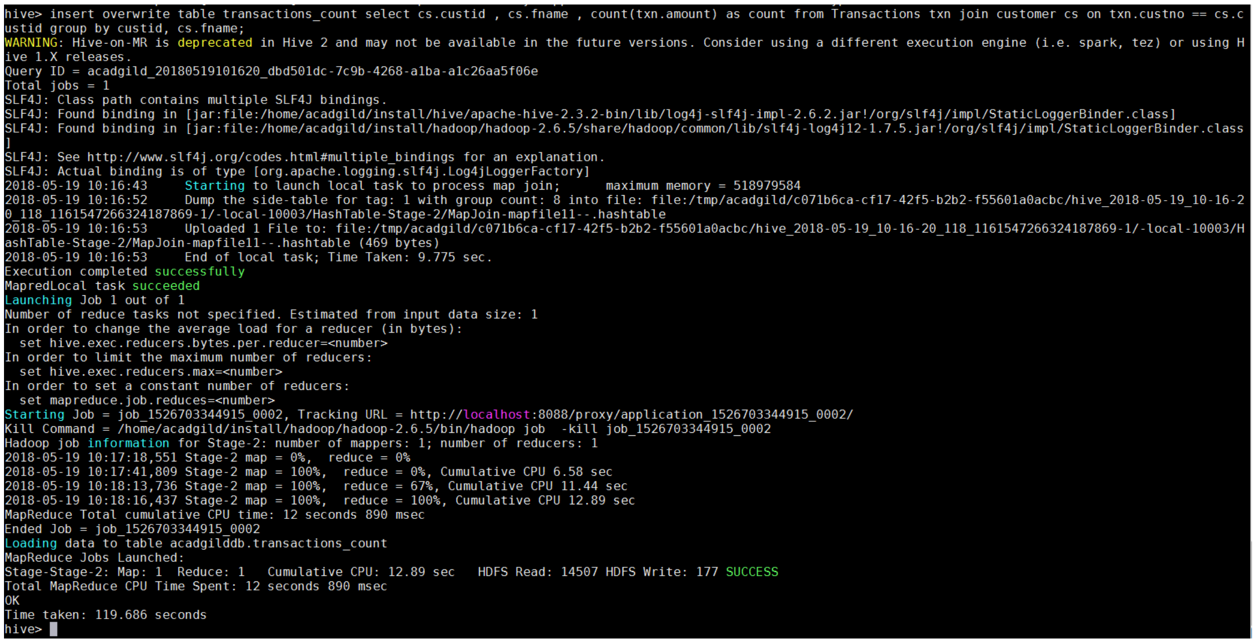
Task 3.1

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).

Command

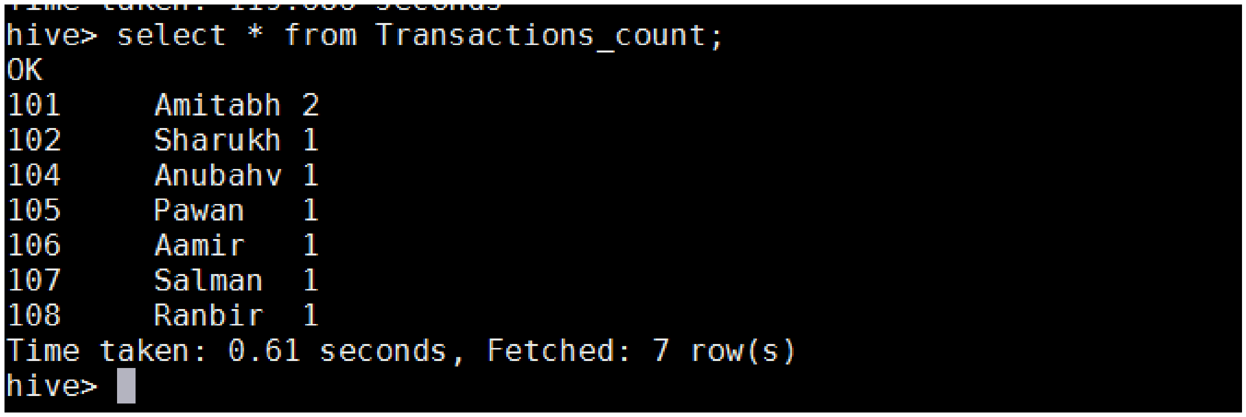
insert overwrite table transactions_count select cs.custid , cs.fname , count(txn.amount) as count from Transactions txn join customer cs on txn.custno == cs.custid group by custid, cs.fname;

Execution of Command



```
hive> insert overwrite table transactions_count select cs.custid , cs.fname , count(txn.amount) as count from Transactions txn join customer cs on txn.custno == cs.c
ustid group by custid, cs.fname;
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 H
ive 1.X releases.
Query ID = acadgild_20180519101620_dbd501dc-7c9b-4268-a1ba-a1c26aa5f06e
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]
2018-05-19 10:16:43     Starting to launch local task to process map join;      maximum memory = 518979584
2018-05-19 10:16:52     Dump the side-table for tag: 1 with group count: 8 into file: file:/tmp/acadgild/c071b6ca-cf17-42f5-b2b2-f55601a0acbc/hive_2018-05-19_10-16-2
0_118_1161547266324187869-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile11--.hashtable
2018-05-19 10:16:53     Uploaded 1 File to: file:/tmp/acadgild/c071b6ca-cf17-42f5-b2b2-f55601a0acbc/hive_2018-05-19_10-16-20_118_1161547266324187869-1/-local-10003/H
ashTable-Stage-2/MapJoin-mapfile11--.hashtable (469 bytes)
2018-05-19 10:16:53     End of local task; Time Taken: 9.775 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_1526703344915_0002, Tracking URL = http://localhost:8088/proxy/application_1526703344915_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1526703344915_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-05-19 10:17:18,551 Stage-2 map = 0%,  reduce = 0%
2018-05-19 10:17:41,809 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 6.58 sec
2018-05-19 10:18:13,736 Stage-2 map = 100%,  reduce = 67%, Cumulative CPU 11.44 sec
2018-05-19 10:18:16,437 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 12.89 sec
MapReduce Total cumulative CPU time: 12 seconds 890 msec
Ended Job = job_1526703344915_0002
Loading data to table acadgilddb.transactions_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 12.89 sec   HDFS Read: 14507 HDFS Write: 177 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 890 msec
OK
Time taken: 119.686 seconds
hive>
```

Output

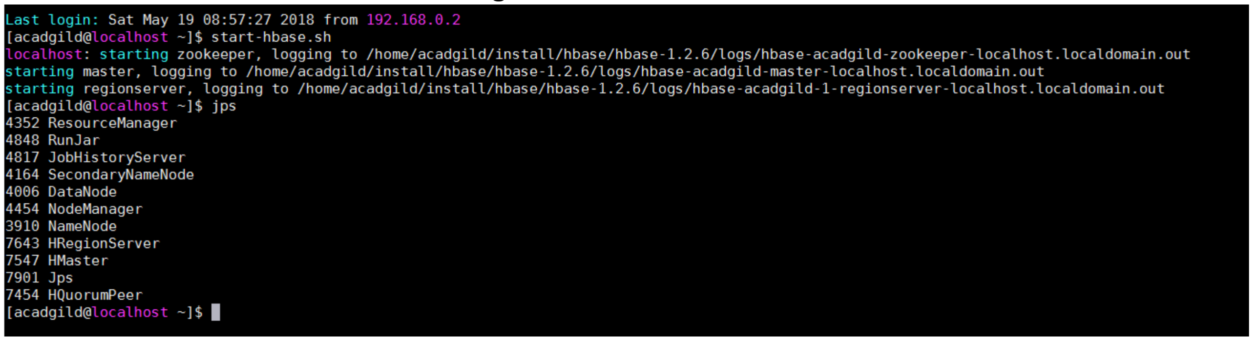


```
hive> select * from Transactions_count;
OK
101     Amitabh 2
102     Sharukh 1
104     Anubahv 1
105     Pawan   1
106     Aamir   1
107     Salman  1
108     Ranbir  1
Time taken: 0.61 seconds, Fetched: 7 row(s)
hive>
```

Task 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)

Make sure hbase deamons are running.



```
Last login: Sat May 19 08:57:27 2018 from 192.168.0.2
[acadgild@localhost ~]$ start-hbase.sh
localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.localdomain.out
starting master, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-master-localhost.localdomain.out
starting regionserver, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-1-regionserver-localhost.localdomain.out
[acadgild@localhost ~]$ jps
4352 ResourceManager
4848 RunJar
4817 JobHistoryServer
4164 SecondaryNameNode
4006 DataNode
4454 NodeManager
3910 NameNode
7643 HRegionServer
7547 HMaster
7901 Jps
7454 HQuorumPeer
[acadgild@localhost ~]$ 
```

Create table in HIVe which is HBASE compliant by using below command

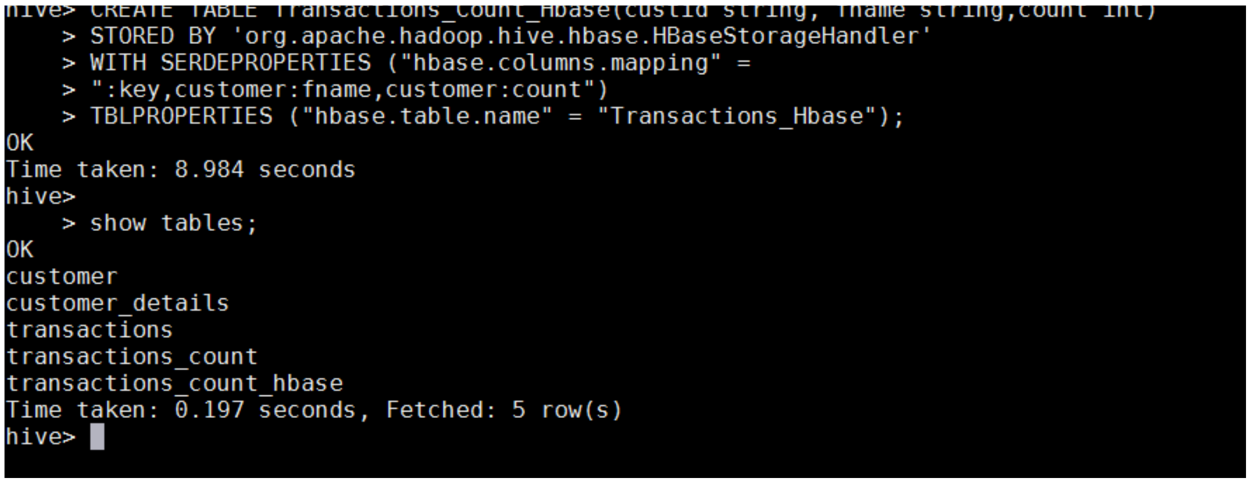
CREATE TABLE Transactions_Count_Hbase(custid string, fname string,count int)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

WITH SERDEPROPERTIES ("hbase.columns.mapping" =

":key,customer:fname,customer:count")

TBLPROPERTIES ("hbase.table.name" = "Transactions_Hbase");



```
hive> CREATE TABLE Transactions_Count_Hbase(custid string, fname string,count int)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES ("hbase.columns.mapping" =
    > ":key,customer:fname,customer:count")
    > TBLPROPERTIES ("hbase.table.name" = "Transactions_Hbase");
OK
Time taken: 8.984 seconds
hive>
    > show tables;
OK
customer
customer_details
transactions
transactions_count
transactions_count_hbase
Time taken: 0.197 seconds, Fetched: 5 row(s)
hive>
```

<u>Column Mapping Details</u>

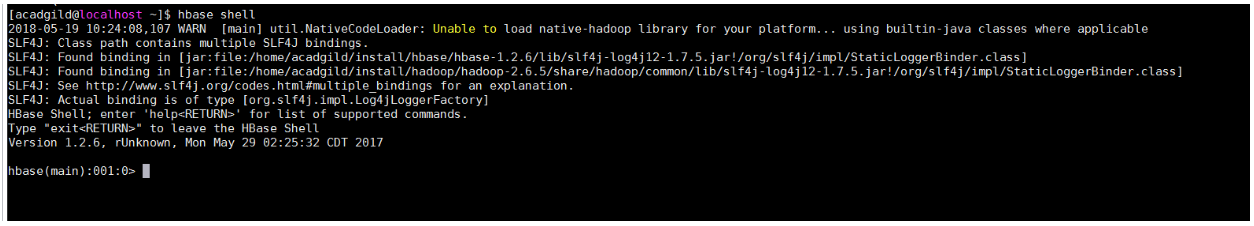
There are two SERDEPROPERTIES that control the mapping of HBase columns to Hive:

- hbase.columns.mapping
- hbase.table.default.storage.type: Can have a value of either string (the default) or binary, this option is only available as of Hive 0.9 and the string behavior is the only one available in earlier versions

The column mapping support currently available is somewhat cumbersome and restrictive:

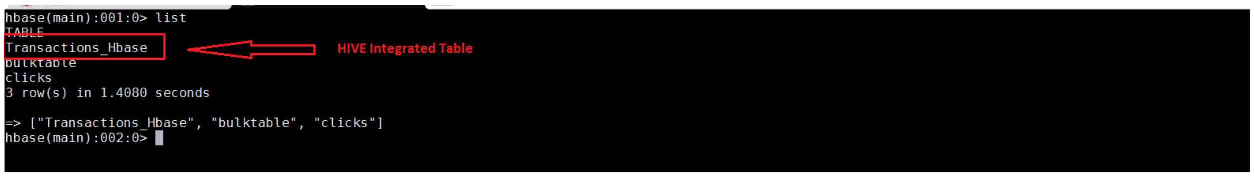
- for each Hive column, the table creator must specify a corresponding entry in the comma-delimited hbase.columns.mapping string (so for a Hive table with n columns, the string should have n entries); whitespace should not be used in between entries
- a mapping entry must be either :key or of the form column-family-name:[column-name][#(binary|string)
- If no type specification is given the value from hbase.table.default.storage.type will be used
- there must be exactly one :key mapping
- if no column-name is given, then the Hive column will map to all columns in the corresponding HBase column family, and the Hive MAP datatype must be used to allow access to these (possibly sparse) columns

Run the Hbase Shell

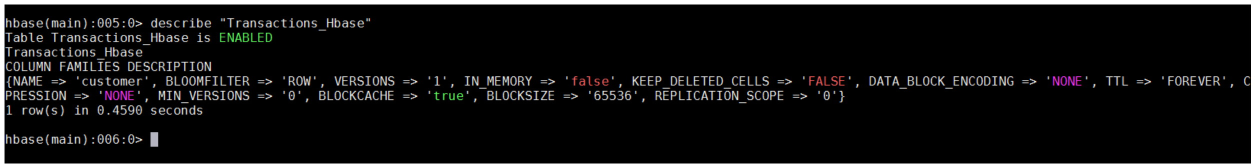


```
[acadgild@localhost ~]$ hbase shell
2018-05-19 10:24:08,107 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/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.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 Hbase tables



```
hbase(main):001:0> list
TABLE
Transactions_Hbase
bulktable
clicks
3 row(s) in 1.4080 seconds

=> ["Transactions_Hbase", "bulktable", "clicks"]
hbase(main):002:0>
```



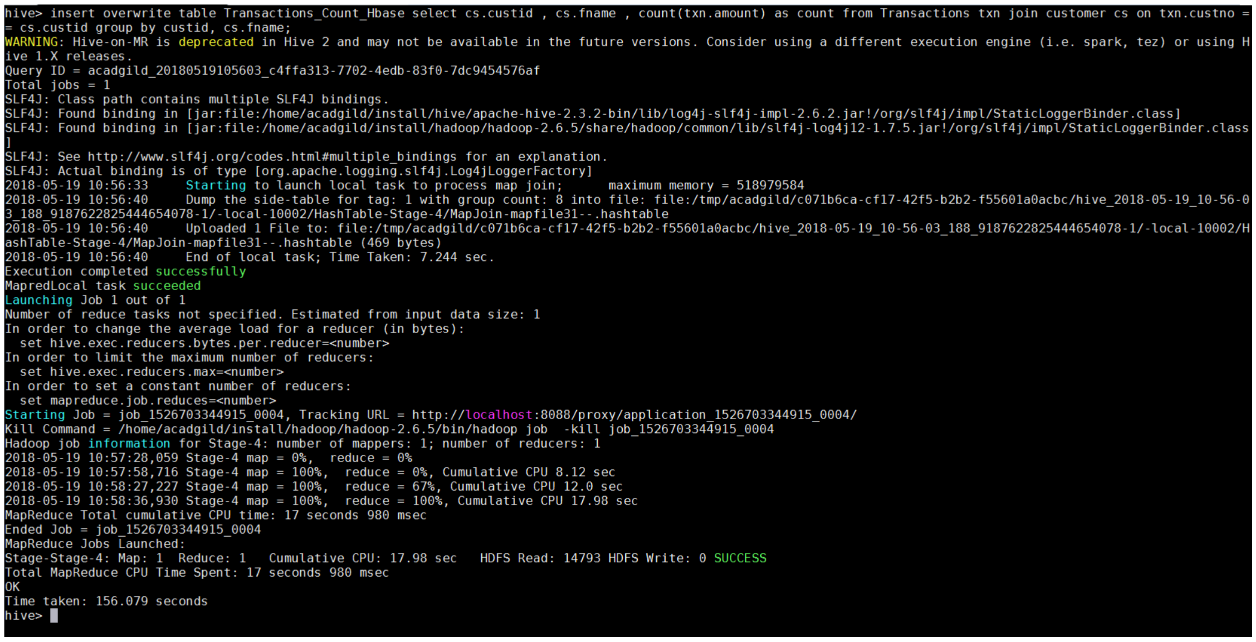
```
hbase(main):005:0> describe "Transactions_Hbase"
Table Transactions_Hbase is ENABLED
Transactions_Hbase
COLUMN FAMILIES DESCRIPTION
{NAME => 'customer', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', C
PRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.4590 seconds

hbase(main):006:0>
```

Task 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)

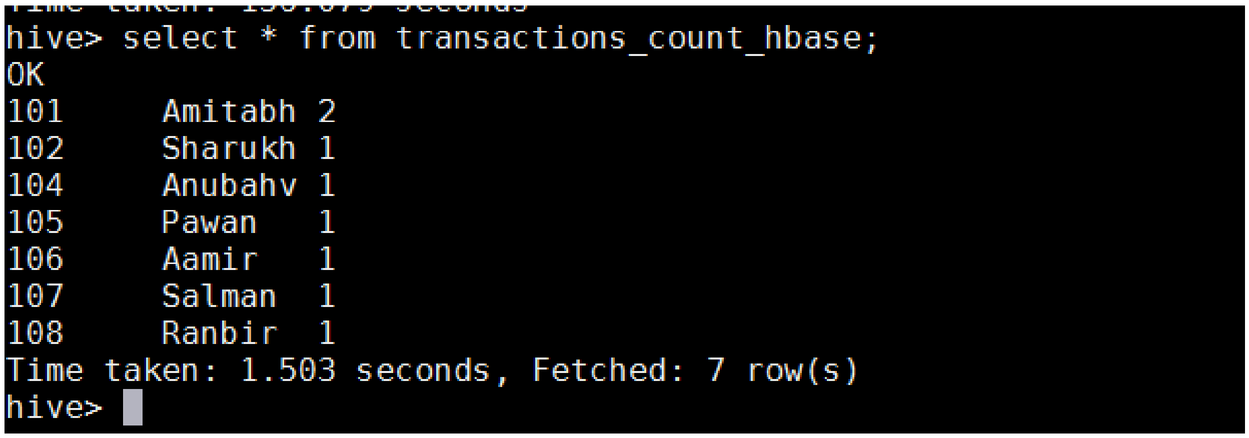
insert overwrite table Transactions_Count_Hbase select cs.custid , cs.fname , count(txn.amount) as count from Transactions txn join customer cs on txn.custno == cs.custid group by custid, cs.fname;

Execution OF Command



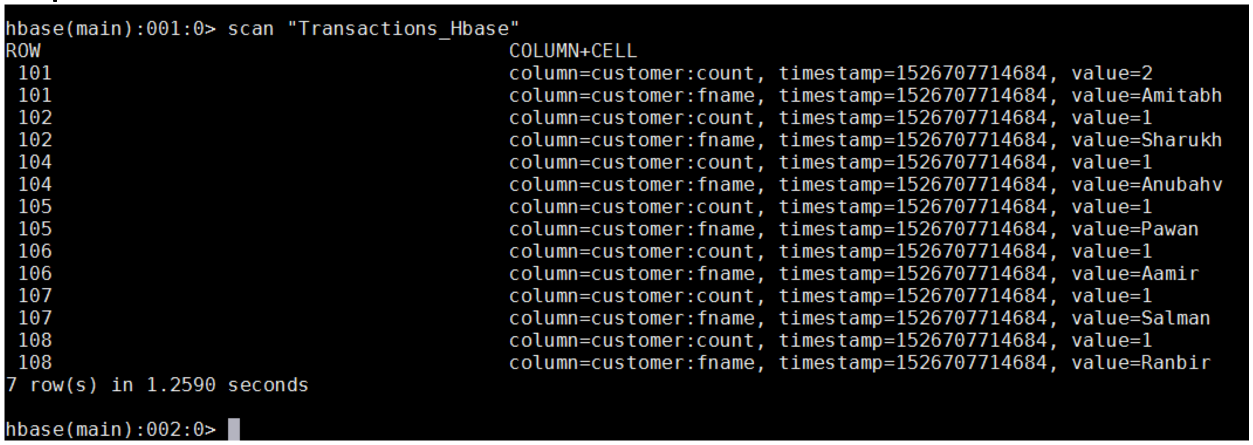
```
hive> insert overwrite table Transactions_Count_Hbase select cs.custid , cs.fname , count(txn.amount) as count from Transactions txn join customer cs on txn.custno =
= cs.custid group by custid, cs.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using H
ive 1.X releases.
Query ID = acadgild_20180519105603_c4ffa313-7702-4edb-83f0-7dc9454576af
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]
2018-05-19 10:56:33     Starting to launch local task to process map join;      maximum memory = 518979584
2018-05-19 10:56:40     Dump the side-table for tag: 1 with group count: 8 into file: file:/tmp/acadgild/c071b6ca-cf17-42f5-b2b2-f55601a0acbc/hive_2018-05-19_10-56-0
3_188_9187622825444654078-1/-local-10002/HashTable-Stage-4/MapJoin-mapfile31--.hashtable
2018-05-19 10:56:40     Uploaded 1 File to: file:/tmp/acadgild/c071b6ca-cf17-42f5-b2b2-f55601a0acbc/hive_2018-05-19_10-56-03_188_9187622825444654078-1/-local-10002/H
ashTable-Stage-4/MapJoin-mapfile31--.hashtable (469 bytes)
2018-05-19 10:56:40     End of local task; Time Taken: 7.244 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_1526703344915_0004, Tracking URL = http://localhost:8088/proxy/application_1526703344915_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1526703344915_0004
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
2018-05-19 10:57:28,059 Stage-4 map = 0%,  reduce = 0%
2018-05-19 10:57:58,716 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 8.12 sec
2018-05-19 10:58:27,227 Stage-4 map = 100%,  reduce = 67%, Cumulative CPU 12.0 sec
2018-05-19 10:58:36,930 Stage-4 map = 100%,  reduce = 100%, Cumulative CPU 17.98 sec
MapReduce Total cumulative CPU time: 17 seconds 980 msec
Ended Job = job_1526703344915_0004
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 17.98 sec   HDFS Read: 14793 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 980 msec
OK
Time taken: 156.079 seconds
hive>
```

Output
From HIVE Shell



```
hive> select * from transactions_count_hbase;
OK
101     Amitabh 2
102     Sharukh 1
104     Anubahv 1
105     Pawan   1
106     Aamir   1
107     Salman  1
108     Ranbir  1
Time taken: 1.503 seconds, Fetched: 7 row(s)
hive>
```

Output from HBase Shell



```
hbase(main):001:0> scan "Transactions_Hbase"
ROW                                         COLUMN+CELL
 101                                        column=customer:count, timestamp=1526707714684, value=2
 101                                        column=customer:fname, timestamp=1526707714684, value=Amitabh
 102                                        column=customer:count, timestamp=1526707714684, value=1
 102                                        column=customer:fname, timestamp=1526707714684, value=Sharukh
 104                                        column=customer:count, timestamp=1526707714684, value=1
 104                                        column=customer:fname, timestamp=1526707714684, value=Anubahv
 105                                        column=customer:count, timestamp=1526707714684, value=1
 105                                        column=customer:fname, timestamp=1526707714684, value=Pawan
 106                                        column=customer:count, timestamp=1526707714684, value=1
 106                                        column=customer:fname, timestamp=1526707714684, value=Aamir
 107                                        column=customer:count, timestamp=1526707714684, value=1
 107                                        column=customer:fname, timestamp=1526707714684, value=Salman
 108                                        column=customer:count, timestamp=1526707714684, value=1
 108                                        column=customer:fname, timestamp=1526707714684, value=Ranbir
7 row(s) in 1.2590 seconds

hbase(main):002:0>
```

<u>Task 6</u>

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

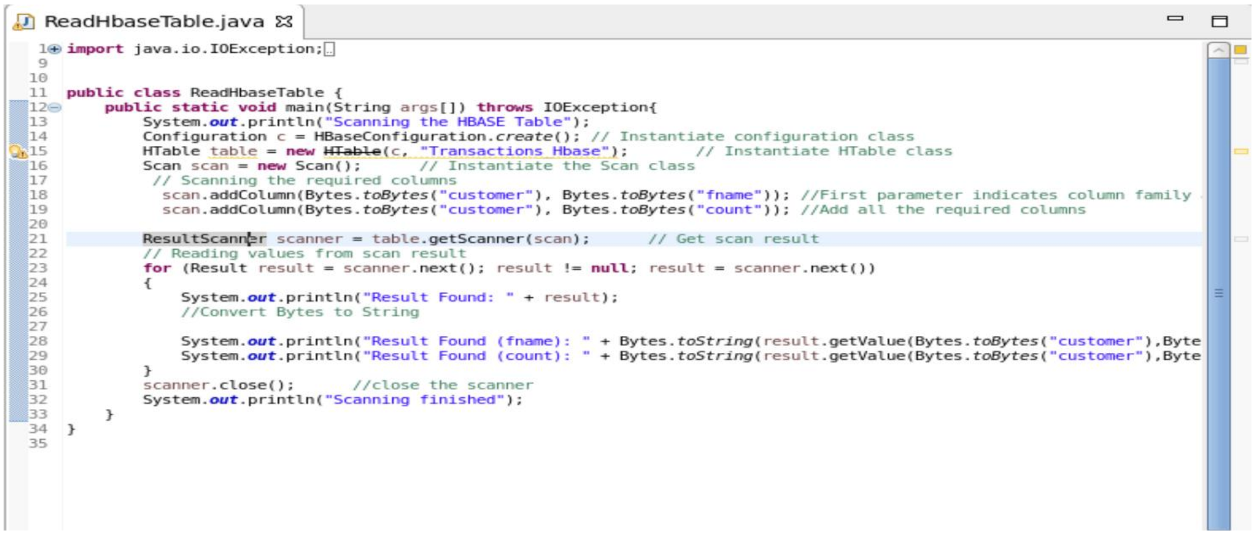
HBase provides java API to communicate with HBase. Java API is the fastest way to communicate with HBase.

- static org.apache.hadoop.conf.Configuration create()
    - This method creates a Configuration with HBase resources.
- HTable(TableName tableName, ClusterConnection connection, ExecutorService pool)
    - Using this constructor, you can create an object to access an HBase table.
- org.apache.hadoop.hbase.client.Scan
    - This creates a object to read / scan the table

Solution Approach

- Create Configuration Object
- Once Configuration Object is created , create Hbase Table using (HTable)
- Create scan object and add columns to be read in the Scan object. While adding make sure add Column family and then the name of column.
- By using getScanner method of the Scan object read the the table and store the result in object of type ResultScanner
- Iterate through the ResultScanner Object to get the row values
- Convert Byte to String by using getValue method of result object

Complete Code Of JAVA API to Read HBase Table

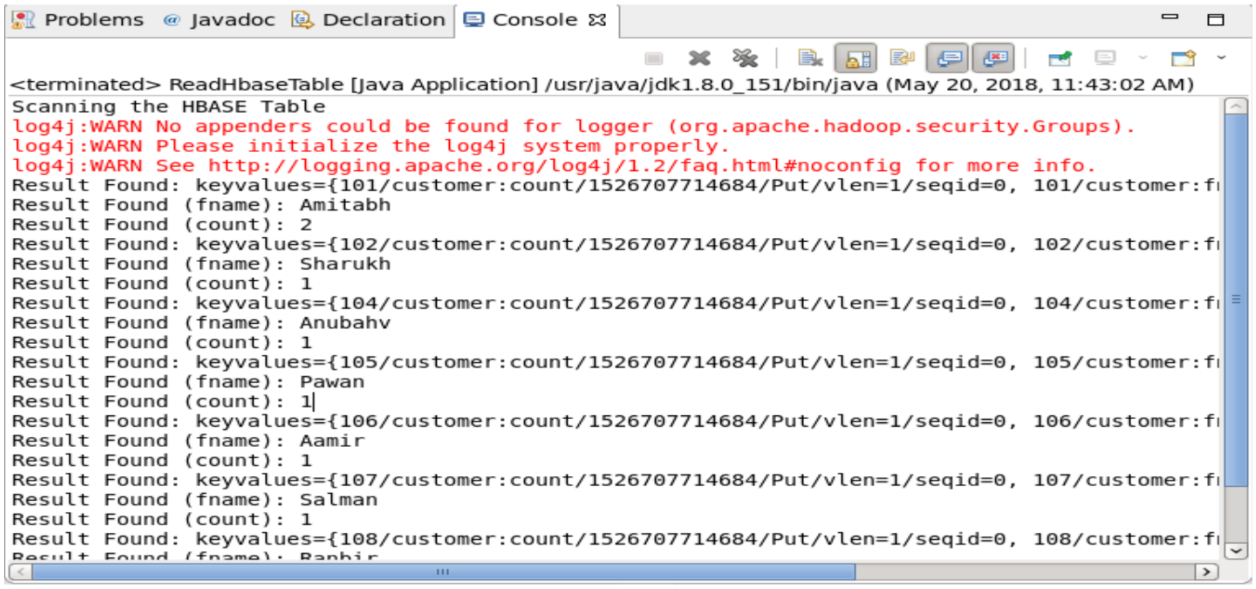


```java
import java.io.IOException;


public class ReadHbaseTable {
    public static void main(String args[]) throws IOException{
        System.out.println("Scanning the HBASE Table");
        Configuration c = HBaseConfiguration.create(); // Instantiate configuration class
        HTable table = new ~~HTable~~(c, "Transactions Hbase");       // Instantiate HTable class
        Scan scan = new Scan();      // Instantiate the Scan class
         // Scanning the required columns
          scan.addColumn(Bytes.toBytes("customer"), Bytes.toBytes("fname")); //First parameter indicates column family
          scan.addColumn(Bytes.toBytes("customer"), Bytes.toBytes("count")); //Add all the required columns

        ResultScanner scanner = table.getScanner(scan);       // Get scan result
        // Reading values from scan result
        for (Result result = scanner.next(); result != null; result = scanner.next())
        {
            System.out.println("Result Found: " + result);
            //Convert Bytes to String

            System.out.println("Result Found (fname): " + Bytes.toString(result.getValue(Bytes.toBytes("customer"),Byte
            System.out.println("Result Found (count): " + Bytes.toString(result.getValue(Bytes.toBytes("customer"),Byte
        }
        scanner.close();      //close the scanner
        System.out.println("Scanning finished");
    }
}
```

Execution Of Code

Can be executed directly from Eclipse (running on the same machine where hadoop is installed and make sure HBASe deamons are running)

Console will display the desired **output** as below



```
<terminated> ReadHbaseTable [Java Application] /usr/java/jdk1.8.0_151/bin/java (May 20, 2018, 11:43:02 AM)
Scanning the HBASE Table
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.
Result Found: keyvalues={101/customer:count/1526707714684/Put/vlen=1/seqid=0, 101/customer:fı
Result Found (fname): Amitabh
Result Found (count): 2
Result Found: keyvalues={102/customer:count/1526707714684/Put/vlen=1/seqid=0, 102/customer:fı
Result Found (fname): Sharukh
Result Found (count): 1
Result Found: keyvalues={104/customer:count/1526707714684/Put/vlen=1/seqid=0, 104/customer:fı
Result Found (fname): Anubahv
Result Found (count): 1
Result Found: keyvalues={105/customer:count/1526707714684/Put/vlen=1/seqid=0, 105/customer:fı
Result Found (fname): Pawan
Result Found (count): 1
Result Found: keyvalues={106/customer:count/1526707714684/Put/vlen=1/seqid=0, 106/customer:fı
Result Found (fname): Aamir
Result Found (count): 1
Result Found: keyvalues={107/customer:count/1526707714684/Put/vlen=1/seqid=0, 107/customer:fı
Result Found (fname): Salman
Result Found (count): 1
Result Found: keyvalues={108/customer:count/1526707714684/Put/vlen=1/seqid=0, 108/customer:fı
Result Found (fname): Ranbir
```