

Case Study 2

Case Study Description

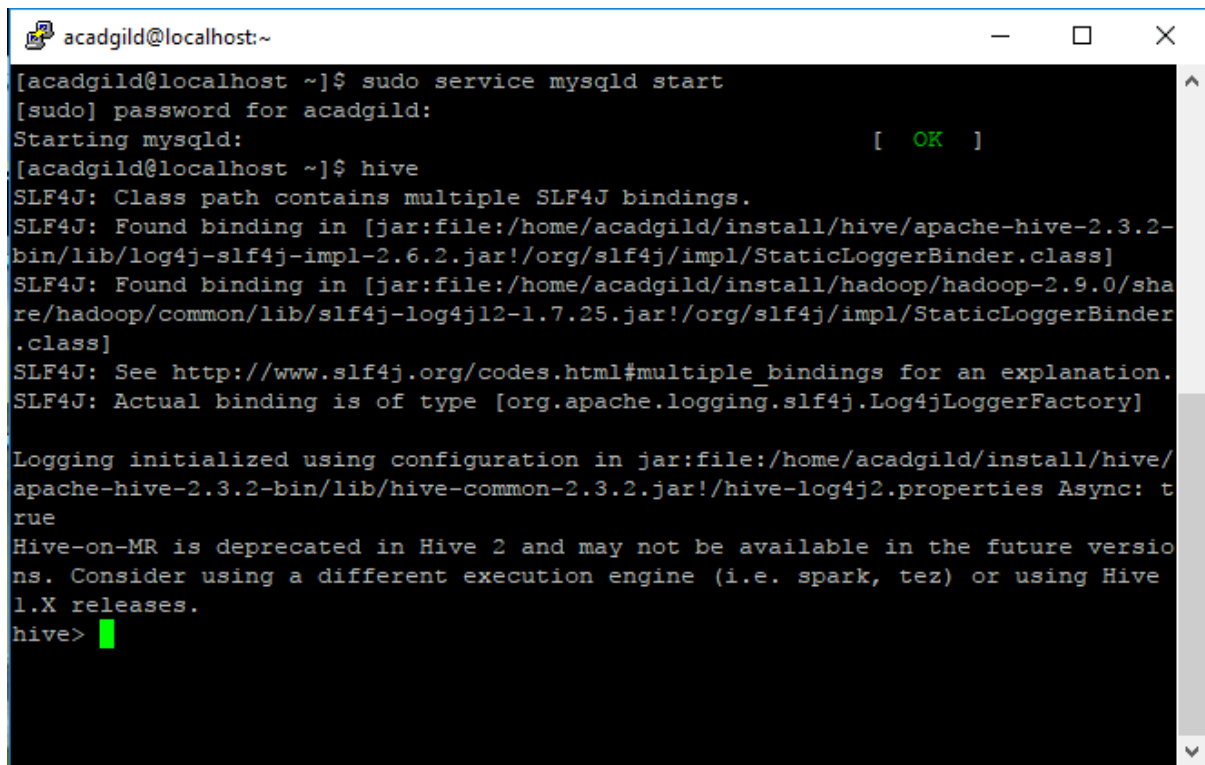
Let us take up the CUSTOMER and TRANSACTIONS table we have created in the Let's Do Together section. Let us solve the following use cases using these tables :-

Step 1 :

Start Hive

sudo service mysqld start

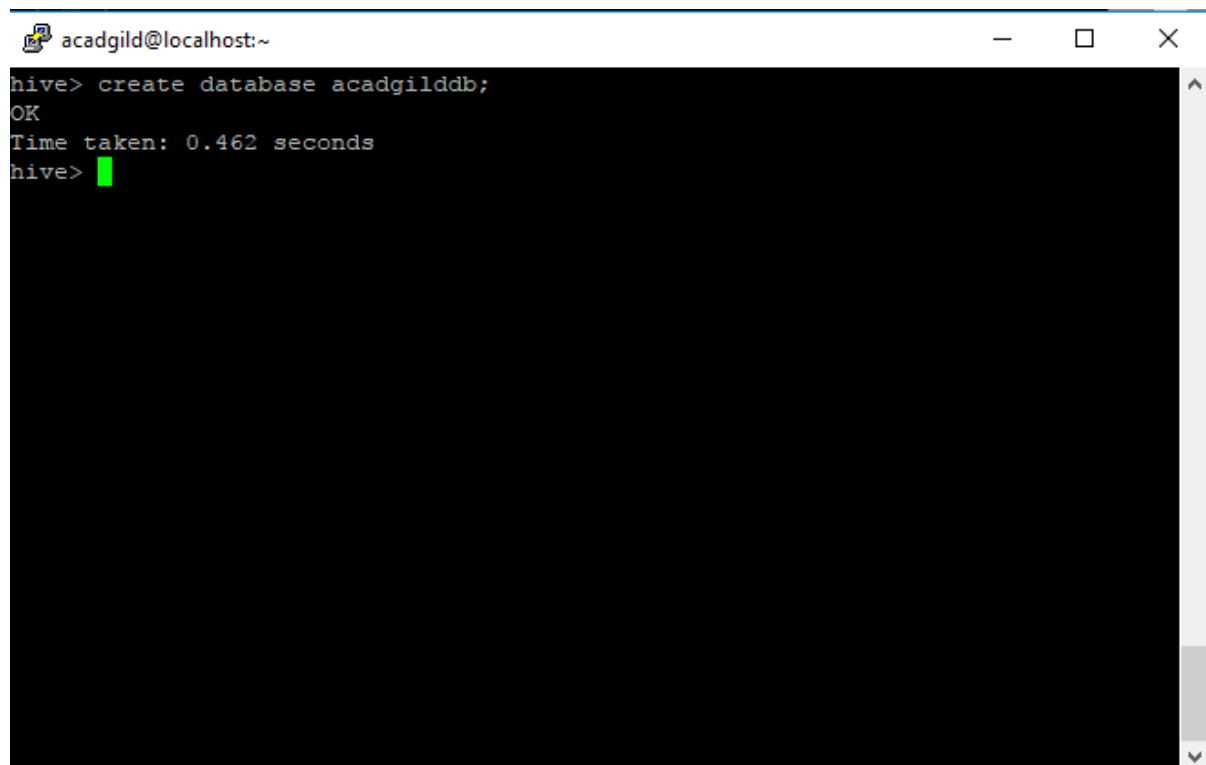
hive

A terminal window titled 'acadgild@localhost:~' showing the execution of two commands. The first command is 'sudo service mysqld start', which prompts for a password and returns '[OK]'. The second command is 'hive', which outputs several SLF4J warnings about multiple bindings and deprecated Hive features, followed by a green cursor prompt 'hive>'.

```
acadgild@localhost:~  
[acadgild@localhost ~]$ sudo service mysqld start  
[sudo] password for acadgild:  
Starting mysqld: [ OK ]  
[acadgild@localhost ~]$ hive  
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-2.9.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.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]  
  
Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.jar!/hive-log4j2.properties Async: true  
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.  
hive>
```

Step2: Create database

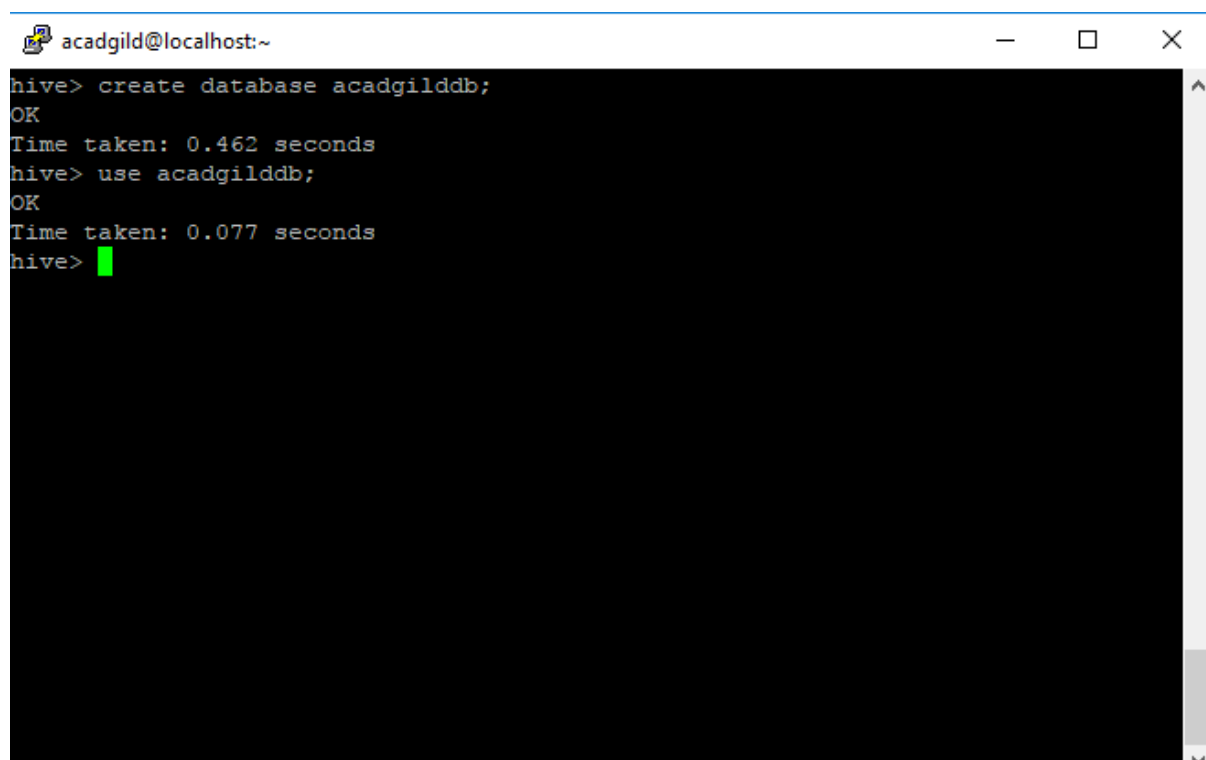
create database acadgilddb;

A terminal window titled 'acadgild@localhost:~' with standard window controls. The Hive prompt 'hive>' is followed by the command 'create database acadgilddb;'. The output shows 'OK' and 'Time taken: 0.462 seconds'. The prompt 'hive>' is followed by a green cursor.

```
acadgild@localhost:~  
hive> create database acadgilddb;  
OK  
Time taken: 0.462 seconds  
hive> █
```

Step 3:

use acadgilddb;

A terminal window titled 'acadgild@localhost:~' with standard window controls. The Hive prompt 'hive>' is followed by the command 'create database acadgilddb;'. The output shows 'OK' and 'Time taken: 0.462 seconds'. The prompt 'hive>' is followed by the command 'use acadgilddb;'. The output shows 'OK' and 'Time taken: 0.077 seconds'. The prompt 'hive>' is followed by a green cursor.

```
acadgild@localhost:~  
hive> create database acadgilddb;  
OK  
Time taken: 0.462 seconds  
hive> use acadgilddb;  
OK  
Time taken: 0.077 seconds  
hive> █
```

Step 4: Create Customer Table

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

```
acadgild@localhost:~  
hive> CREATE TABLE CUSTOMER(custid INT,fname STRING,lname STRING,age INT,profession STRING)row format delimited fields terminated by ',';  
OK  
Time taken: 2.442 seconds  
hive> █
```

Step 5: Create custs.txt file

```
acadgild@localhost:~  
[acadgild@localhost ~]$ cat custs.txt  
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  
[acadgild@localhost ~]$ █
```

Step 6: Load custs.txt into Customer table

LOAD DATA LOCAL INPATH '/home/acadgild/customer.txt' into table CUSTOMER;

```
acadmild@localhost:~  
hive> LOAD DATA LOCAL INPATH '/home/acadmild/custs.txt' into table CUSTOMER;  
Loading data to table acadmildddb.customer  
OK  
Time taken: 3.407 seconds  
hive> █
```

Step 7 :

Select * from CUSTOMER;

```
acadmild@localhost:~  
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: 0.649 seconds, Fetched: 8 row(s)  
hive> █
```

Step 8 : Create Transaction Table

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 ',';

```
acadgild@localhost:~  
hive> CREATE TABLE TRANSACTIONS(txnno INT,txndate STRING,custno INT,amount DOUBL  
E,category STRING,product STRING,city STRING,state STRING,spendby STRING)row for  
mat delimited fields terminated by ',';  
OK  
Time taken: 0.389 seconds  
hive>
```

Step 9 : Create txn.txt file

```
acadgild@localhost:~  
[acadgild@localhost ~]$ cat txn.txt  
97834,05/02/2018,101,965,Entertainment,Movie,Pune,Maharashtra, Daughter  
98396,12/01/2018,102,239,Food,Grocery,Patna,Bihar,Self  
34908,06/01/2018,101,875,Travel,Air,Bangalore,Karnataka,Spouse  
70958,17/02/2018,104,439,Food,Restaurant,Delhi,Delhi,Wife  
09874,21/01/2018,105,509,Entertainment,Park,Kolkata,West Bengal  
94585,19/01/2018,106,629,Rent,House,Hyderabad,Telangana,Self  
45509,20/01/2018,107,953,Travel,Rail,Chennai,Tamil Nadu,Brother  
07864,01/02/2018,108,569,Rent,Parking,Goa,Goa,Wife  
[acadgild@localhost ~]$
```

Step 10: Load txn.txt file into TRANSACTIONS Table

LOAD DATA LOCAL INPATH '/home/acadgild/transactions.txt'
into table TRANSACTIONS;

```
acadgild@localhost:~  
hive> LOAD DATA LOCAL INPATH '/home/acadgild/txn.txt' into table TRANSACTIONS;  
Loading data to table acadgild.db.transactions  
OK  
Time taken: 1.206 seconds  
hive>
```

Step 11:

Select * From TRANSACTIONS;

```
acadgild@localhost:~  
hive> select * from TRANSACTIONS;  
OK  
97834 05/02/2018 101 965.0 Entertainment Movie Pune Maharashtra  
tra Daughter  
98396 12/01/2018 102 239.0 Food Grocery Patna Bihar Self  
34908 06/01/2018 101 875.0 Travel Air Bangalore Karnataka  
a Spouse  
70958 17/02/2018 104 439.0 Food Restaurant Delhi Delhi W  
ife  
9874 21/01/2018 105 509.0 Entertainment Park Kolkata West Ben  
gal NULL  
94585 19/01/2018 106 629.0 Rent House Hyderabad Telangan  
a Self  
45509 20/01/2018 107 953.0 Travel Rail Chennai Tamil Nadu B  
rother  
7864 01/02/2018 108 569.0 Rent Parking Goa Goa Wife  
Time taken: 0.659 seconds, Fetched: 8 row(s)  
hive> █
```

Step 12 : Show tables

```
acadgild@localhost:~  
hive> show tables;  
OK  
customer  
transactions  
Time taken: 0.317 seconds, Fetched: 2 row(s)  
hive> █
```

1. Find out the number of transaction done by each customer (These should be take up in module 8 itself)

select a.custid,a.fname,COUNT(b.amount) from CUSTOMER a join TRANSACTIONS b on a.custid = b.custno GROUP BY custid,fname ;

```
acagild@localhost:~$ hive>
hive> select a.custid,a.fname,COUNT(b.amount) from CUSTOMER a join TRANSACTIONS b on a.custid = b.custno GROUP BY custid,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 Hive
1.X releases.
Query ID = acagild_20180503211909_lcfb5a8e-e482-4f45-bl65-c5ech8b57f5c
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acagild/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/acagild/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]
2018-05-03 21:19:41 Starting to launch local task to process map join; maximum memory = 518979584
2018-05-03 21:19:46 Dump the side-table for tag: 0 with group count: 8 into file: file:/tmp/acagild/478d7988-e66f-4ff3-9160-9a8bd38af626/hive_2018-05-03_21-19-09_5
07_7996452272595681947-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile40--.hashtable
2018-05-03 21:19:46 Uploaded 1 File to: file:/tmp/acagild/478d7988-e66f-4ff3-9160-9a8bd38af626/hive_2018-05-03_21-19-09_507_7996452272595681947-1/-local-10005/Hash
Table-Stage-2/MapJoin-mapfile40--.hashtable (469 bytes)
2018-05-03 21:19:46 End of local task; Time Taken: 5.75 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>

2018-05-03 21:20:48,754 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 6.48 sec
2018-05-03 21:21:16,644 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 10.45 sec
MapReduce Total cumulative CPU time: 10 seconds 450 msec
Ended Job = job_1525358071900_0008
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 10.45 sec HDFS Read: 13398 HDFS Write: 263 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 450 msec
OK
101 Amitabh 2
102 Sharukh 1
104 Anubhav 1
105 Pawan 1
106 Aamir 1
107 Salman 1
108 Ranbir 1
Time taken: 128.588 seconds, Fetched: 7 row(s)
hive>
```

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)

CREATE TABLE TRANSACTIONS_COUNT(custid INT,fname STRING,count INT)row format delimited fields terminated by ',';

```

acadgild@localhost:~
hive> CREATE TABLE TRANSACTIONS_COUNT(custid INT,fname STRING,count INT)row format delimited fields terminated by ',';
OK
Time taken: 0.453 seconds
hive> █

```

- 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).

insert into table TRANSACTIONS_COUNT select
a.custid,a.fname,COUNT(b.amount) from CUSTOMER a join
TRANSACTIONS b on a.custid = b.custno GROUP BY custid,fname;

```

acadgild@localhost:~
hive> insert into table TRANSACTIONS_COUNT select a.custid,a.fname,COUNT(b.amount) from CUSTOMER a join TRANSACTIONS b on a.custid = b.custno GROUP BY custid,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 Hive
1.X releases.
Query ID = acadgild_20180503212310_6dcea29c-5d10-4f21-9c68-9afffd5bde
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]
2018-05-03 21:33:33 Starting to launch local task to process map join: maximum memory = 518979584
2018-05-03 21:33:42 Dump the side-table for tag: 0 with group count: 8 into file: file:/tmp/acadgild/478d7988-e66f-4ff3-9160-9a8bd38af626/hive_2018-05-03_21-23-10_8
2018-05-03 21:33:43 Uploaded 1 File to: file:/tmp/acadgild/478d7988-e66f-4ff3-9160-9a8bd38af626/hive_2018-05-03_21-23-10_828_2157845336841837900-1/-local-10003/Hash
Table-Stage-2/MapJoin-mapfile50--.hashtable (469 bytes)
2018-05-03 21:33:43 End of local task; Time Taken: 9.511 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.reducers=<number>
Starting Job = job_1525358071900_0009, Tracking URL = http://localhost:8088/proxy/application_1525358071900_0009/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525358071900_0009
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-05-03 21:34:10,625 Stage-2 map = 0%, reduce = 0%
2018-05-03 21:34:35,182 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 5.12 sec
2018-05-03 21:35:01,565 Stage-2 map = 100%, reduce = 67%, Cumulative CPU 9.45 sec
2018-05-03 21:35:05,118 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 11.32 sec
MapReduce Total cumulative CPU time: 11 seconds 320 msec
Ended Job = job_1525358071900_0009
Loading data to table acadgildddb.transactions_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 11.32 sec HDFS Read: 14132 HDFS Write: 177 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 320 msec
OK
Time taken: 118.721 seconds
hive> █

```

select * from TRANSACTIONS_COUNT;


```

acadgild@localhost:~
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: 0.677 seconds, Fetched: 7 row(s)
hive> █

```

- Now let's make the TRANSACTIONS_COUNT table Hbase compliant. In the sense, 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(ID INT,username STRING, count INT) STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler'WITH
SERDEPROPERTIES
('hbase.columns.mapping':key,cf1:username,cf2:count');

```

acadgild@localhost:~
hive> CREATE TABLE TRANSACTIONS_HBase(ID INT,username STRING, count INT)
> STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
> WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,cf1:username,cf2:count');
OK
Time taken: 2.904 seconds
hive> █

```

- 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 into table TRANSACTIONS_HBase
select * from TRANSACTIONS_COUNT;

```
acadgild@localhost:~$ hive> insert into table 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 using Hive
1.X releases.
Query ID = acadgild_20180513204539_0031d7b5-4576-4b78-9458-54134b16a142
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_1526220777807_0003, Tracking URL = http://localhost:8088/proxy/application_1526220777807_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1526220777807_0003
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2018-05-13 20:46:19,154 Stage-3 map = 0%, reduce = 0%
2018-05-13 20:47:19,390 Stage-3 map = 0%, reduce = 0%
2018-05-13 20:47:29,895 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 6.96 sec
MapReduce Total cumulative CPU time: 8 seconds 480 msec
Ended Job = job_1526220777807_0003
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Cumulative CPU: 8.48 sec HDFS Read: 11053 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 480 msec
OK
Time taken: 124.154 seconds
hive>
```

In HBASE:
describe 'acadgilddb.transactions_hbase'

```
acadgild@localhost:~/install/hbase/hbase-1.2.6/bin$ hbase(main):008:0> describe 'acadgilddb.transactions_hbase'
Table acadgilddb.transactions_hbase is ENABLED
acadgilddb.transactions_hbase
COLUMN FAMILIES DESCRIPTION
(NAME => 'cf1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION
=> 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0')
(NAME => 'cf2', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION
=> 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0')
2 row(s) in 0.3700 seconds
hbase(main):009:0>
```

scan 'acadgilddb.transactions_hbase'

```

acadgild@localhost:~/install/hbase/hbase-1.2.6/bin
hbase(main):009:0> scan 'acadgilddb.transactions_hbase'
ROW                                COLUMN+CELL
101                                column=cf1:username, timestamp=1526224654778, value=Amitabh
101                                column=cf2:count, timestamp=1526224654778, value=2
102                                column=cf1:username, timestamp=1526224654778, value=Sharukh
102                                column=cf2:count, timestamp=1526224654778, value=1
104                                column=cf1:username, timestamp=1526224654778, value=Anubahv
104                                column=cf2:count, timestamp=1526224654778, value=1
105                                column=cf1:username, timestamp=1526224654778, value=Pawan
105                                column=cf2:count, timestamp=1526224654778, value=1
106                                column=cf1:username, timestamp=1526224654778, value=Aamir
106                                column=cf2:count, timestamp=1526224654778, value=1
107                                column=cf1:username, timestamp=1526224654778, value=Salman
107                                column=cf2:count, timestamp=1526224654778, value=1
108                                column=cf1:username, timestamp=1526224654778, value=Ranbir
108                                column=cf2:count, timestamp=1526224654778, value=1
7 row(s) in 0.2250 seconds
hbase(main):010:0>

```

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

Step 1:

```
import org.apache.spark.{SparkConf, SparkContext}
```

```
import org.apache.spark.rdd.RDD
```

```
import org.apache.spark.streaming.{Seconds, StreamingContext, Time}
```

```
import org.apache.spark.sql.SparkSession
```

```
import org.apache.log4j.{Level, Logger}
```

```
object SqlNetworkWordCount {
```

```
  def main(args: Array[String]): Unit = {
```

```
    println("hey Spark SQL Streaming")
```

```
    val conf = new
```

```
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
```

```
    val sc = new SparkContext(conf)
```

```
    val rootLogger =Logger.getRootLogger()
```

```

rootLogger.setLevel(Level.ERROR)

println("hey Spark Streaming ---> 1")

//val sparkConf = new SparkConf().setAppName("NetworkWordCount")
println("hey Spark Streaming ---> 2")

val ssc = new StreamingContext(sc, Seconds(10))
val lines = ssc.socketTextStream("localhost", 9999)
println("hey Spark Streaming ---> 3")
val words = lines.flatMap(_.split(" "))

// Convert RDDs of the words DStream to DataFrame and run SQL query
words.foreachRDD { (rdd: RDD[String], time: Time) =>
    val spark = SparkSessionSingleton.getInstance(rdd.sparkContext.getConf)
    import spark.implicits._

    // Convert RDD[String] to RDD[case class] to DataFrame
    val wordsDataFrame = rdd.map(w => Record(w)).toDF()

    // Creates a temporary view using the DataFrame
    wordsDataFrame.createOrReplaceTempView("words")

    // Do word count on table using SQL and print it
    val wordCountsDataFrame =
        spark.sql("select word, count(*) as total from words group by word")
    println(s"===== $time =====")
    wordCountsDataFrame.show()
}

ssc.start()

```

```
    ssc.awaitTermination()
  }

  /** Case class for converting RDD to DataFrame */
  case class Record(word: String)

  /** Lazily instantiated singleton instance of SparkSession */
  object SparkSessionSingleton {

    @transient private var instance: SparkSession = _

    def getInstance(sparkConf: SparkConf): SparkSession = {
      if (instance == null) {
        instance = SparkSession
          .builder
          .config(sparkConf)
          .getOrCreate()
      }
      instance
    }
  }
}
```

```
SqlNetworkWordCount.scala 8
1
2+ import org.apache.spark.{SparkConf, SparkContext}
7
8= object SqlNetworkWordCount {
9
10= def main(args: Array[String]): Unit = {
11     println("hey Spark SQL Streaming");
12
13     val conf = new SparkConf().setMaster("local[2]").setApp
14     val sc = new SparkContext(conf);
15     val rootLogger = Logger.getRootLogger();
16     rootLogger.setLevel(Level.ERROR);
17
18
19     println("hey Spark Streaming ---> 1");
20     //val sparkConf = new SparkConf().setAppName("NetworkW
21     println("hey Spark Streaming ---> 2");
22     val ssc = new StreamingContext(sc, Seconds(10));
23     val lines = ssc.socketTextStream("localhost", 9999);
24     println("hey Spark Streaming ---> 3");
```

```
SqlNetworkWordCount.scala 8
25
26     val words = lines.flatMap(_.split(" "));
27
28     // Convert RDDs of the words DStream to DataFrame and
29     words.foreachRDD { (rdd: RDD[String], time: Time) =>
30         val spark = SparkSessionSingleton.getInstance(rdd.sp
31         import spark.implicits._;
32
33         // Convert RDD[String] to RDD[case class] to DataFrame
34         val wordsDataFrame = rdd.map(w => Record(w)).toDF();
35
36         // Creates a temporary view using the DataFrame
37         wordsDataFrame.createOrReplaceTempView("words");
38
39         // Do word count on table using SQL and print it
40         val wordCountsDataFrame =
41             spark.sql("select word, count(*) as total from words")
42         println(s"===== $time =====")
43         wordCountsDataFrame.show()
44     }
45
```

```
SqlNetworkWordCount.scala  ✖
46
47   ssc.start()
48   ssc.awaitTermination()
49
50 }
51
52
53 /** Case class for converting RDD to DataFrame */
54 case class Record(word: String)
55
56 /** Lazily instantiated singleton instance of SparkSession */
57 object SparkSessionSingleton {
58
59   @transient private var instance: SparkSession = _
60
61   def getInstance(sparkConf: SparkConf): SparkSession = {
62     if (instance == null) {
63       instance = SparkSession
64         .builder
65         .config(sparkConf)
```

```
SqlNetworkWordCount.scala  ✖
55
56 /** Lazily instantiated singleton instance of SparkSession */
57 object SparkSessionSingleton {
58
59   @transient private var instance: SparkSession = _
60
61   def getInstance(sparkConf: SparkConf): SparkSession = {
62     if (instance == null) {
63       instance = SparkSession
64         .builder
65         .config(sparkConf)
66         .getOrCreate()
67     }
68     instance
69   }
70 }
71
72 }
73
74
```

Step 2: Start netcat from a terminal

nc -lk 9999

```
acadmild@localhost:~  
[acadmild@localhost ~]$ nc -lk 9999  
101 Abhishek  
101 Abhishek  
102 Sharukh  
104 Anubahv  
105 Pawan  
106 Aamir  
107 Salman  
108 Ranbir  
█
```

Step 3: Display Results:

101 Abhishek 2

102 Sharukh 1

104 Anubahv 1

105 Pawan 1

106 Aamir 1

107 Salman 1

108 Ranbir 1


```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (May 19, 20

===== 1526706160000 ms =====
+-----+
|word|total|
+-----+
+-----+

===== 1526706170000 ms =====
+-----+
|  word|total|
+-----+
|   102|    1|
|Sharukh|    1|
+-----+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (May 19, 20

===== 1526706210000 ms =====
+-----+
|word|total|
+-----+
+-----+

===== 1526706220000 ms =====
+-----+
|  word|total|
+-----+
|   104|    1|
|Anubahv|    1|
+-----+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (May 19, 20

===== 1526706260000 ms =====
+---+---+
|word|total|
+---+---+
+---+---+

===== 1526706270000 ms =====
+---+---+
| word|total|
+---+---+
| Pawan|    1|
|  105|    1|
+---+---+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (May 19, 20

===== 1526706300000 ms =====
+---+---+
|word|total|
+---+---+
+---+---+

===== 1526706310000 ms =====
+---+---+
| word|total|
+---+---+
|  106|    1|
|Aamir|    1|
+---+---+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (May 19, 20

===== 1526706340000 ms =====
+---+---+
|word|total|
+---+---+
+---+---+

===== 1526706350000 ms =====
+---+---+
|  word|total|
+---+---+
|Salman|    1|
|   107|    1|
+---+---+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (May 19, 20

===== 1526706390000 ms =====
+---+---+
|word|total|
+---+---+
+---+---+

===== 1526706400000 ms =====
+---+---+
|  word|total|
+---+---+
|Ranbir|    1|
|   108|    1|
+---+---+
```