Incremental Import from MYSQL to HIVE Using Sqoop

Note: Cloudera quickstart VM is used to perform import and export between MYSQL and HIVE

In Cloudera quickstart VM all daemons are started at the time when we start VM, so there is no need to start all the required daemons manually, like as in acadgild VM we start hadoop daemons with "start-all.sh" command and mysql service with "sudo service mysqld start" command.

Sqoop supports two types of incremental imports:

- **append**
- **lastmodified**

We can use the **--incremental** argument to specify the type of incremental import we want to perform.

When to use append mode?

We should specify the append mode when importing a table, where new rows are continually added with increasing row id values. We must specify the column containing the row's id with **--check-column**. Sqoop imports rows where the check column has a value greater than the one specified with **--last-value**.

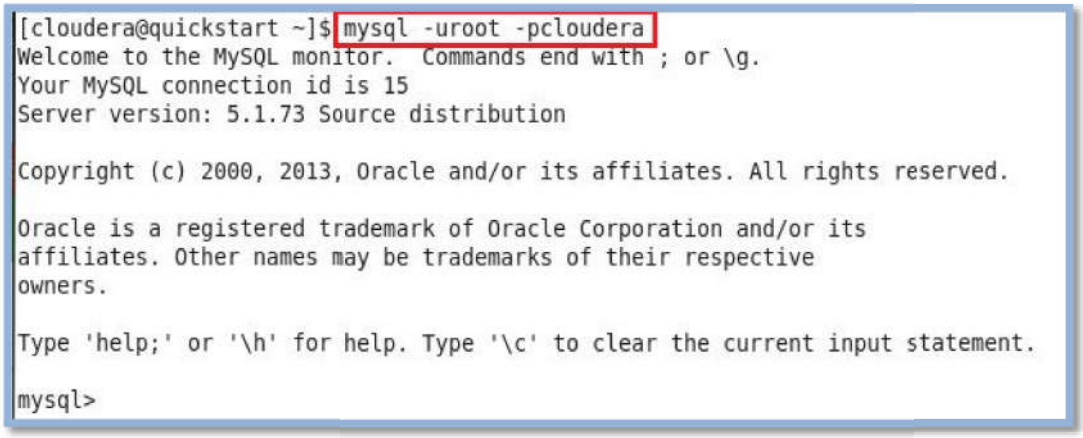
When to use lastmodified mode?

An alternate table **update** strategy supported by Sqoop is called lastmodified mode. This should be used when rows of the source table are updated, and each such update will set the value of a last-modified column to the current timestamp. Rows where the check column holds a timestamp more recent than the timestamp specified with **--last-value** are imported.

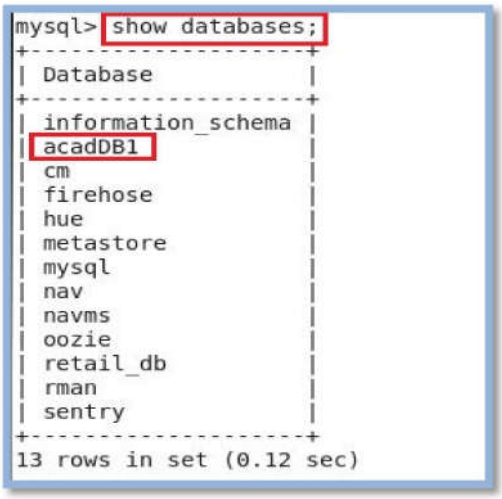
At the end of an incremental import, the value which should be specified as **--last-value** for a subsequent import is printed to the screen. When running a subsequent import, we should specify **--last-value** in such a way to ensure we import only the new or updated data. This is handled automatically by creating an incremental import as a saved job known as **sqoop job**, which is the preferred mechanism for performing a recurring incremental import.

To perform incremental import in hive from mysql using sqoop below steps are followed:

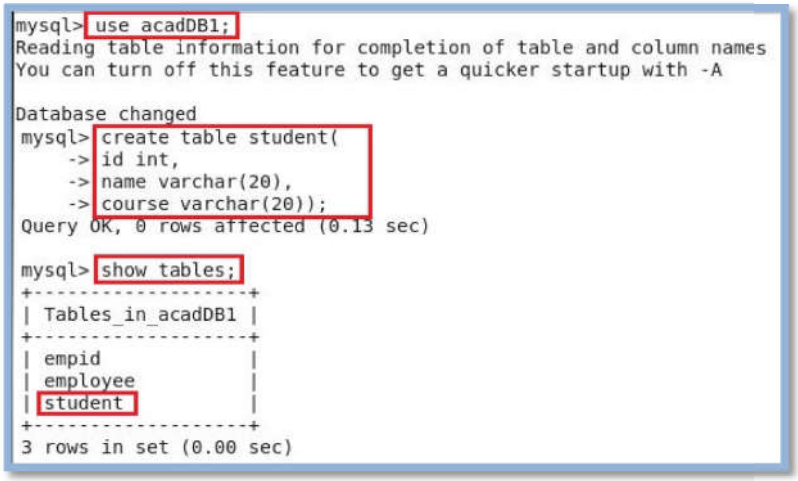
Step 1: Logged into mysql using below command:



Step 2: Checked which databases are present in mysql:



Step 3: Inside acadDB1 database, created new student table with three fields:



Below command displays schema of student table:

```
mysql> describe student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int(11)     | YES  |     | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| course | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

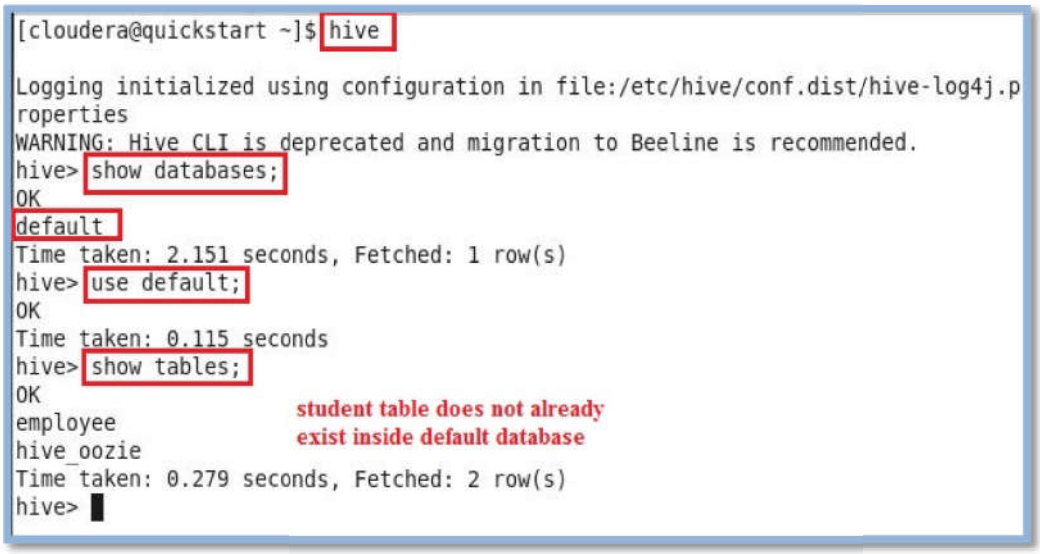
Step 4: Inserted five records in student table using below command, and using select statement checked whether records got inserted successfully inside table or not:

```
mysql> insert into student values
    -> (1,'Harman','Big Data'),
    -> (2,'Simran','Android'),
    -> (3,'Ravi','Spark'),
    -> (4,'Kabir','Sqoop'),
    -> (5,'Raman','Hadoop');
Query OK, 5 rows affected (0.09 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------+--------+----------+
| id   | name   | course   |
+------+--------+----------+
|    1 | Harman | Big Data |
|    2 | Simran | Android  |
|    3 | Ravi   | Spark    |
|    4 | Kabir  | Sqoop    |
|    5 | Raman  | Hadoop   |
+------+--------+----------+
5 rows in set (0.00 sec)

mysql>
```

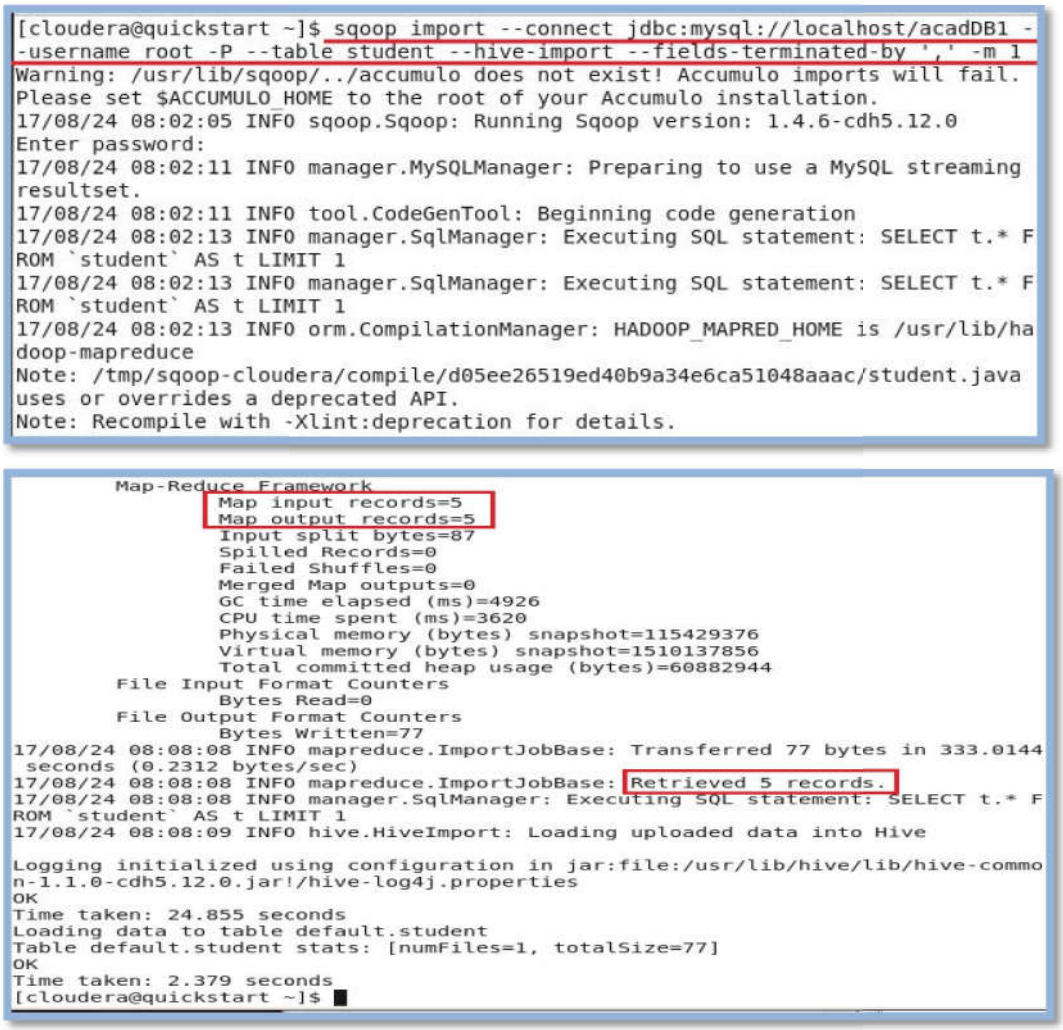
Step 5: Granted permission to root user to access the database over the network, followed by flushing the privileges (The reload/flush privileges command tells the server to reload the grant tables into memory), and committing all changes to database as follows:

```
mysql> grant all on *.* to 'root'@'localhost' with grant option;
Query OK, 0 rows affected (0.09 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.03 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

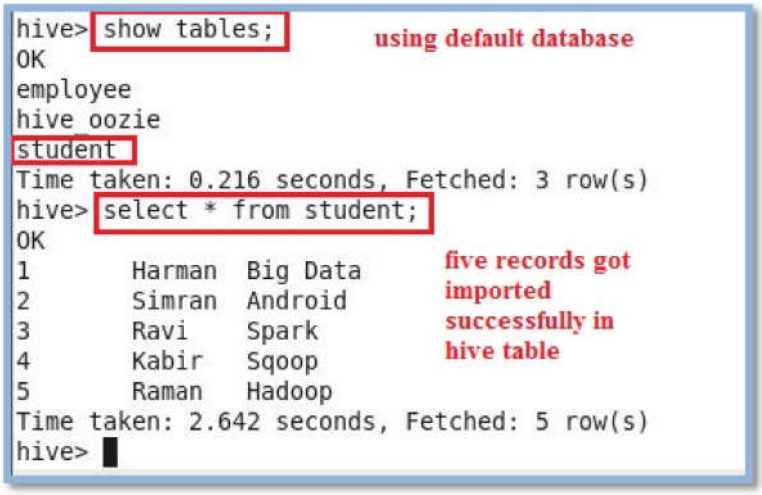
Step 6: In hive, checked whether "student" table inside default database already exist or not:



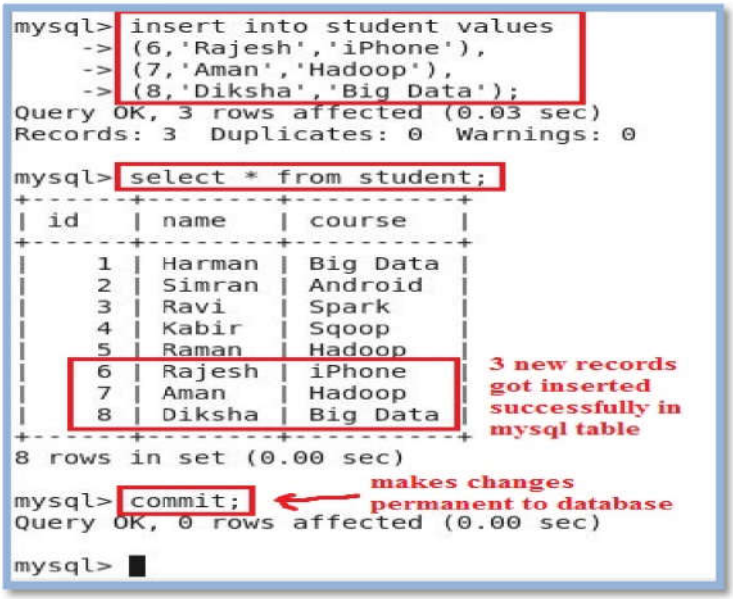
Step 7: Since "student" table does not already exist in hive, therefore ran "sqoop import" command to import the data in hive table:



Step 8: Above screenshots show that data got imported successfully inside hive table, so using below command checked in hive whether table with same name i.e. "student" inside default database created or not:



Step 9: As table has been created inside hive, so again inserted few records in mysql table to show incremental import:



There is no need to grant and flush privileges again, because same session of mysql is used, new session of mysql is not started.

Step 10: In this step, incremental import is performed with append mode, where last value of column "id" is checked, and only those records having greater value than last value of "id" are imported:

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/acadDB1 -
-username root -P --table student --hive-import --fields-terminated-by ',' --inc
remental append --check-column id --last-value 5 -m 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
17/08/24 08:13:45 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.12.0
Enter password:
17/08/24 08:14:00 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
17/08/24 08:14:01 INFO tool.CodeGenTool: Beginning code generation
17/08/24 08:14:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
17/08/24 08:14:04 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
17/08/24 08:14:04 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/ha
doop-mapreduce
```

```
Note: Recompile with -Xlint:deprecation for details.
17/08/24 08:14:18 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-clou
dera/compile/c0972f43e24059e589243c8bf4810c81/student.jar
17/08/24 08:14:25 INFO tool.ImportTool: Maximal id query for free form increment
al import: SELECT MAX(`id`) FROM `student`
17/08/24 08:14:25 INFO tool.ImportTool: Incremental import based on column `id`
17/08/24 08:14:25 INFO tool.ImportTool: Lower bound value: 5
17/08/24 08:14:25 INFO tool.ImportTool: Upper bound value: 8
17/08/24 08:14:25 WARN manager.MySQLManager: It looks like you are importing fro
m mysql.
```

```
                Map input records=3
                Map output records=3
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=3135
                CPU time spent (ms)=3060
                Physical memory (bytes) snapshot=98463744
                Virtual memory (bytes) snapshot=1510170624
                Total committed heap usage (bytes)=60882944
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=48
17/08/24 08:19:13 INFO mapreduce.ImportJobBase: Transferred 48 bytes in 287.8141
 seconds (0.1668 bytes/sec)
17/08/24 08:19:13 INFO mapreduce.ImportJobBase: Retrieved 3 records.
17/08/24 08:19:14 INFO util.AppendUtils: Creating missing output directory - stu
dent
17/08/24 08:19:14 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
17/08/24 08:19:14 INFO hive.HiveImport: Loading uploaded data into Hive

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-commo
n-1.1.0-cdh5.12.0.jar!/hive-log4j.properties
OK
Time taken: 9.753 seconds
Loading data to table default.student
Table default.student stats: [numFiles=2, totalSize=125]
OK
Time taken: 2.995 seconds
```

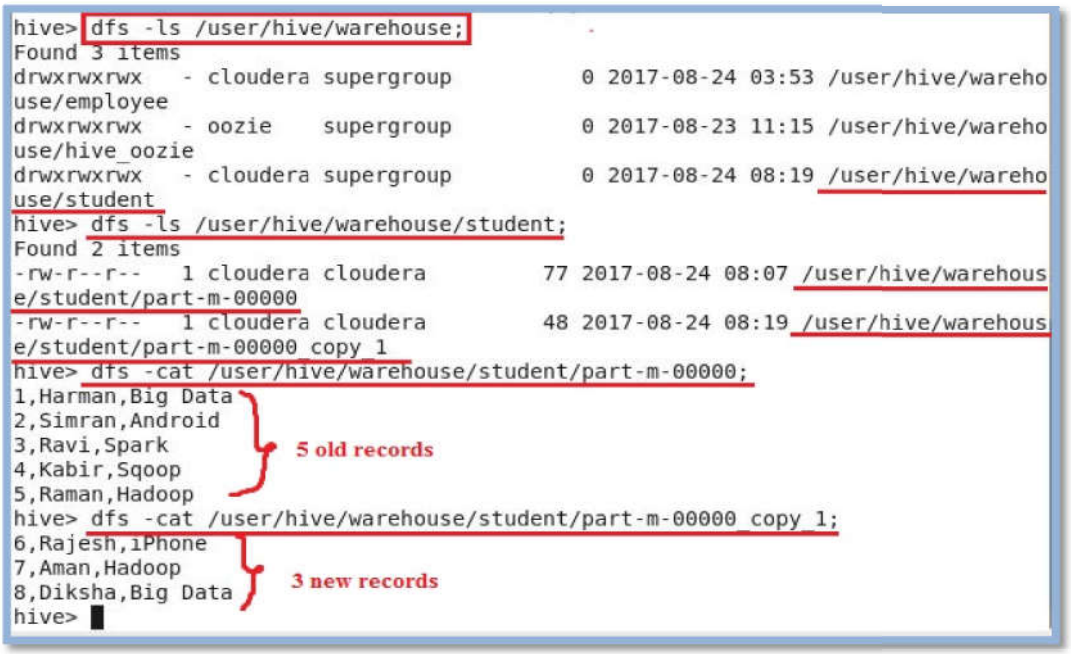
student directory in /user/hive/warehouse contains 2 files, one with old records another with new records

Step 11: Below command shows that latest records got appended successfully inside hive table "student":

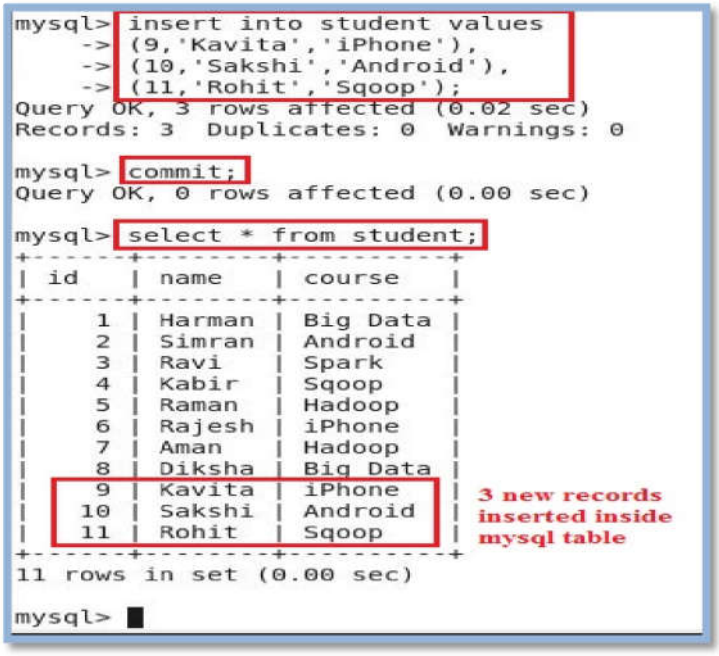
```
hive> select * from student;
OK
1       Harman  Big Data
2       Simran  Android
3       Ravi    Spark
4       Kabir   Sqoop
5       Raman   Hadoop
6       Rajesh  iPhone
7       Aman    Hadoop
8       Diksha  Big Data
Time taken: 0.193 seconds, Fetched: 8 row(s)
hive>
```

3 new records have been appended successfully

Step 12: Below screenshot shows that inside HDFS, two copies of records are maintained in student table directory, one with old records and other with new records:



Step 13: To show how sqoop job works, inserted few records inside mysql table:



Step 14: Created sqoop job "mysqoopjob" which does incremental import by checking itself the last value of --check-column argument:

```
[cloudera@quickstart ~]$ sqoop job --create mysqoopjob -- import --connect jdbc:
mysql://localhost/acadDB1 --username root -P --table student --hive-import --fie
lds-terminated-by ',' --incremental append --check-column id -m 1
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
17/08/24 08:31:48 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.12.0
Enter password:
[cloudera@quickstart ~]$
```

sqoop job "mysqoopjob" is created successfully

Step 15: Executed sqoop job using below command:

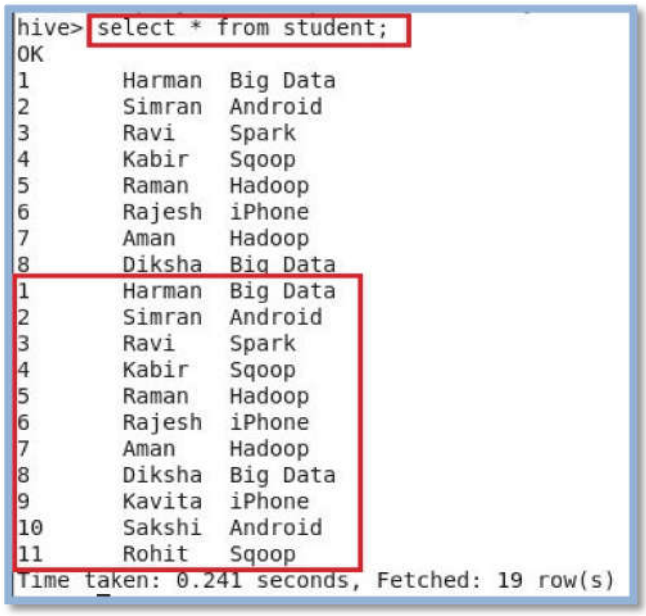
```
[cloudera@quickstart ~]$ sqoop job --exec mysqoopjob
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
17/08/24 08:33:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.12.0
Enter password:
17/08/24 08:33:07 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
17/08/24 08:33:07 INFO tool.CodeGenTool: Beginning code generation
17/08/24 08:33:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
17/08/24 08:33:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
17/08/24 08:33:08 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/ha
doop-mapreduce
```

```
17/08/24 08:33:19 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-clou
dera/compile/45319fca4c8876d9453ad40f37bfd216/student.jar
17/08/24 08:33:32 INFO tool.ImportTool: Maximal id query for free form increment
al import: SELECT MAX(`id`) FROM `student`
17/08/24 08:33:32 INFO tool.ImportTool: Incremental import based on column `id`
17/08/24 08:33:32 INFO tool.ImportTool: Upper bound value: 11
17/08/24 08:33:32 WARN manager.MySQLManager: It looks like you are importing fro
m mysql.
```

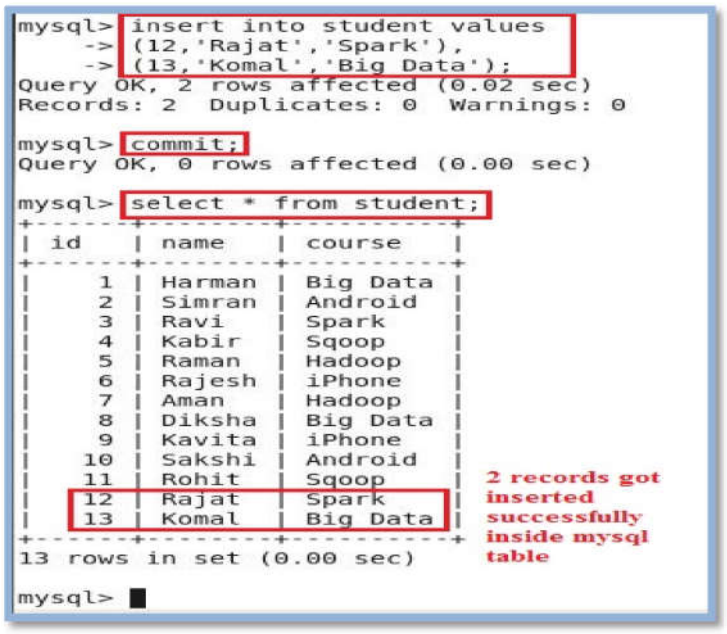
```
        Map-Reduce Framework
                Map input records=11
                Map output records=11
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=1154
                CPU time spent (ms)=2950
                Physical memory (bytes) snapshot=121921536
                Virtual memory (bytes) snapshot=1510166528
                Total committed heap usage (bytes)=60882944
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=174
17/08/24 08:36:42 INFO mapreduce.ImportJobBase: Transferred 174 bytes in 186.051
5 seconds (0.9352 bytes/sec)
17/08/24 08:36:42 INFO mapreduce.ImportJobBase: Retrieved 11 records.
17/08/24 08:36:42 INFO util.AppendUtils: Creating missing output directory - stu
dent
17/08/24 08:36:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `student` AS t LIMIT 1
17/08/24 08:36:42 INFO hive.HiveImport: Loading uploaded data into Hive

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-commo
n-1.1.0-cdh5.12.0.jar!/hive-log4j.properties
OK
Time taken: 8.701 seconds
Loading data to table default.student
Table default.student stats: [numFiles=3, totalSize=299]
OK
```

sqoop job imports total records from mysql table when it is run first time, in our case total records are 11

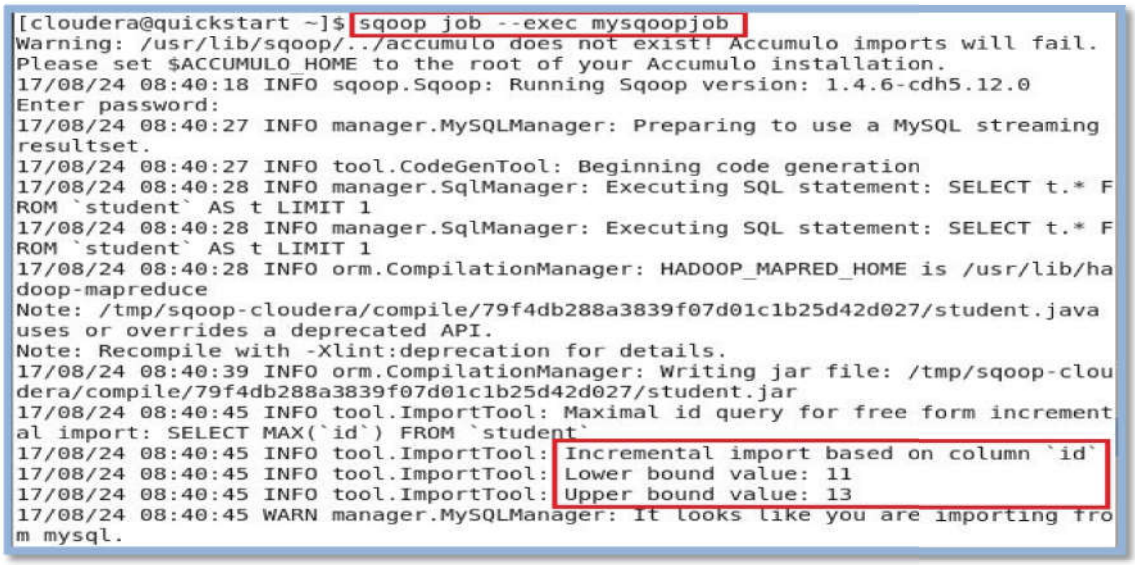
Step 16: Below screenshot shows that, latest 3 records are not imported, however, all records are imported in hive table when sqoop job is run first time:

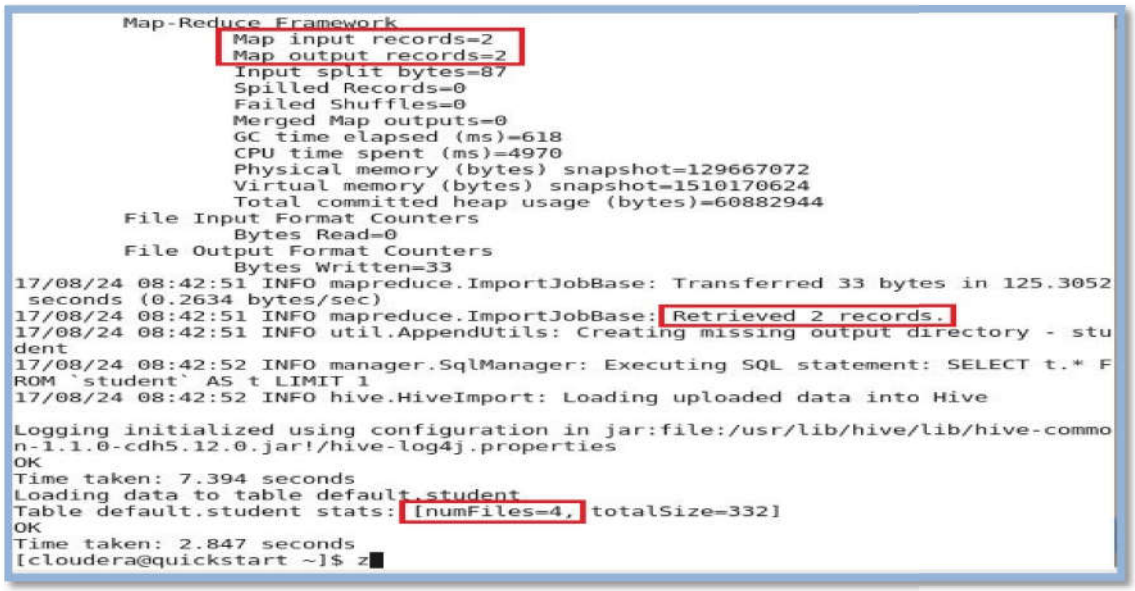


Step 17: Again inserted 2 new records inside mysql table, to show how sqoop job worked when it was executed second time:



Step 18: Executed sqoop job "mysqoopjob":





Step 19: Below command shows that only latest records are appended in hive table, when sqoop job was run second time, we do not need to provide –last-value of –check-column argument as sqoop job automatically picks latest value of –check-column: