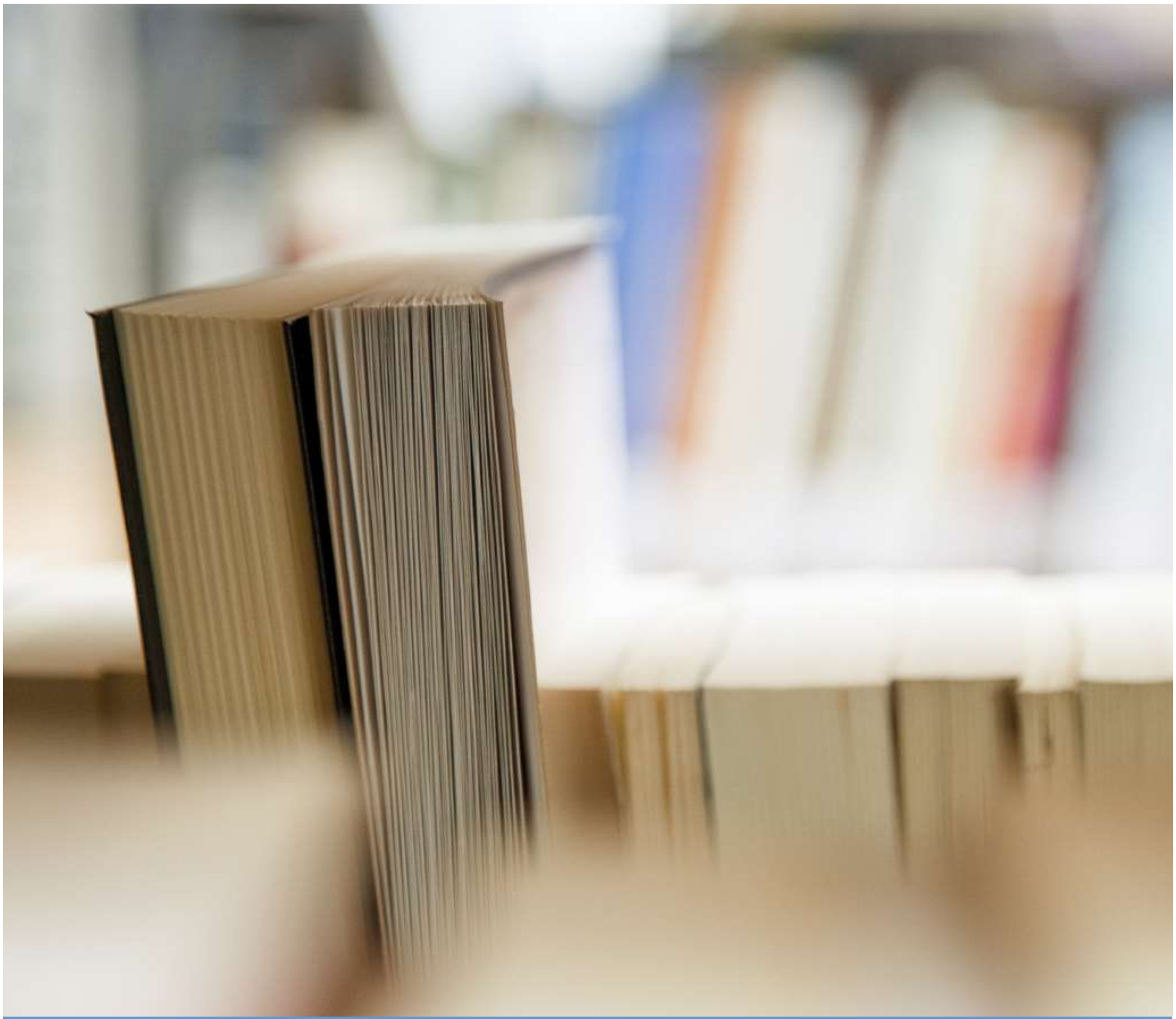


BIG DATA HADOOP & SPARK TRAINING

Assignment 09: Assignment on Advance Hive

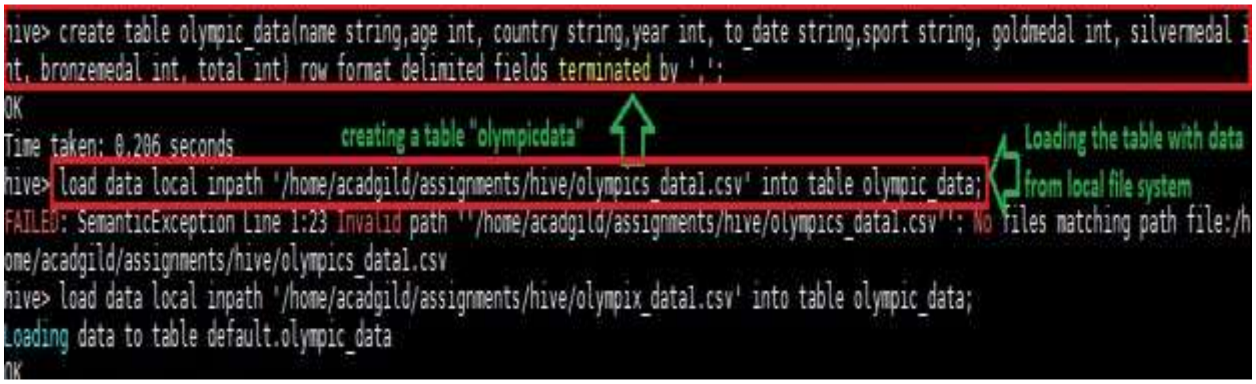
Rashmi Krishna

4/16/18

Olympic Data Analysis in Hive

- Created a table “Olympic_data” with columns *name, age ,country, year, to_date, sport, gold medal, silver medal, bronze medal and total of all medals*.
- Loaded the data from local filesytem to the above created Olympic_data table.
- The data set consists of the following fields.
    - Athlete: This field consists of the athlete name
    - Age: This field consists of athlete ages
    - Country: This fields consists of the country names which participated in Olympics
    - Year: This field consists of the year
    - Closing Date: This field consists of the closing date of ceremony
    - Sport: Consists of the sports name
    - Gold Medals: No. of Gold medals
    - Silver Medals: No. of Silver medals
    - Bronze Medals: No. of Bronze medals
    - Total Medals: Consists of total no. of medals

Above steps are as shown below:



```
ive> create table olympic_data(name string,age int, country string,year int, to_date string,sport string, goldmedal int, silvermedal i
nt, bronzemedal int, total int) row format delimited fields terminated by ',';
OK
Time taken: 0.206 seconds
hive> load data local inpath '/home/acadgild/assignments/hive/olympics_data1.csv' into table olympic_data;
FAILED: SemanticException Line 1:23 Invalid path ''/home/acadgild/assignments/hive/olympics_data1.csv'': No files matching path file:/h
ome/acadgild/assignments/hive/olympics_data1.csv
hive> load data local inpath '/home/acadgild/assignments/hive/olympix_data1.csv' into table olympic_data;
Loading data to table default.olympic_data
OK
```

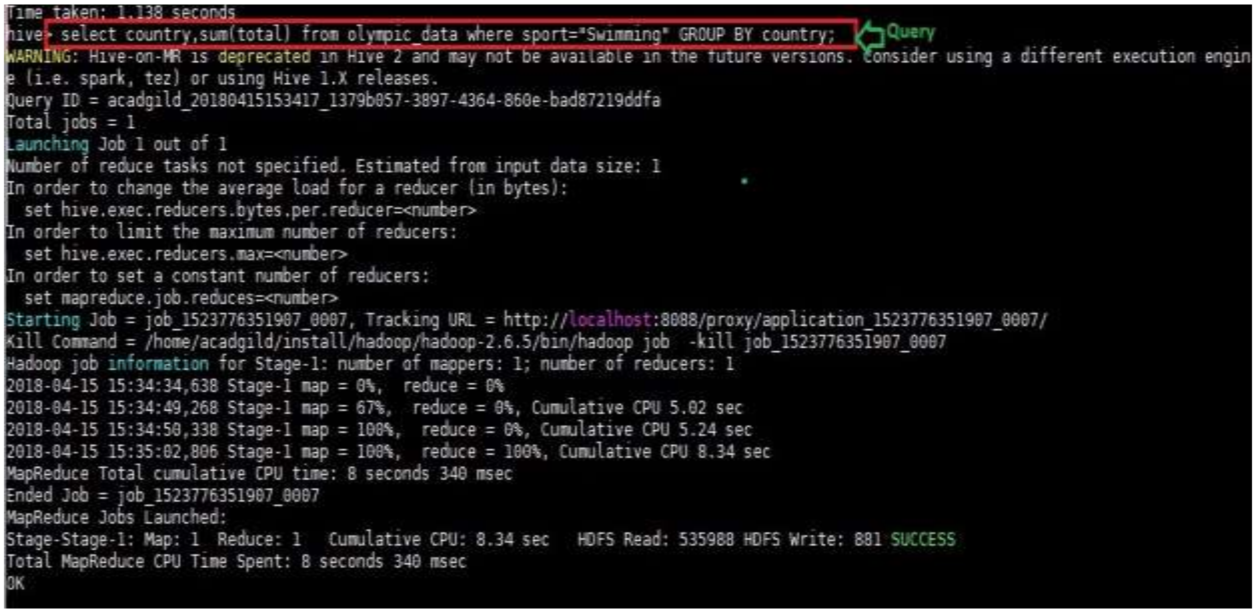
creating a table "olympicdata"
Loading the table with data from local file system

Task 1

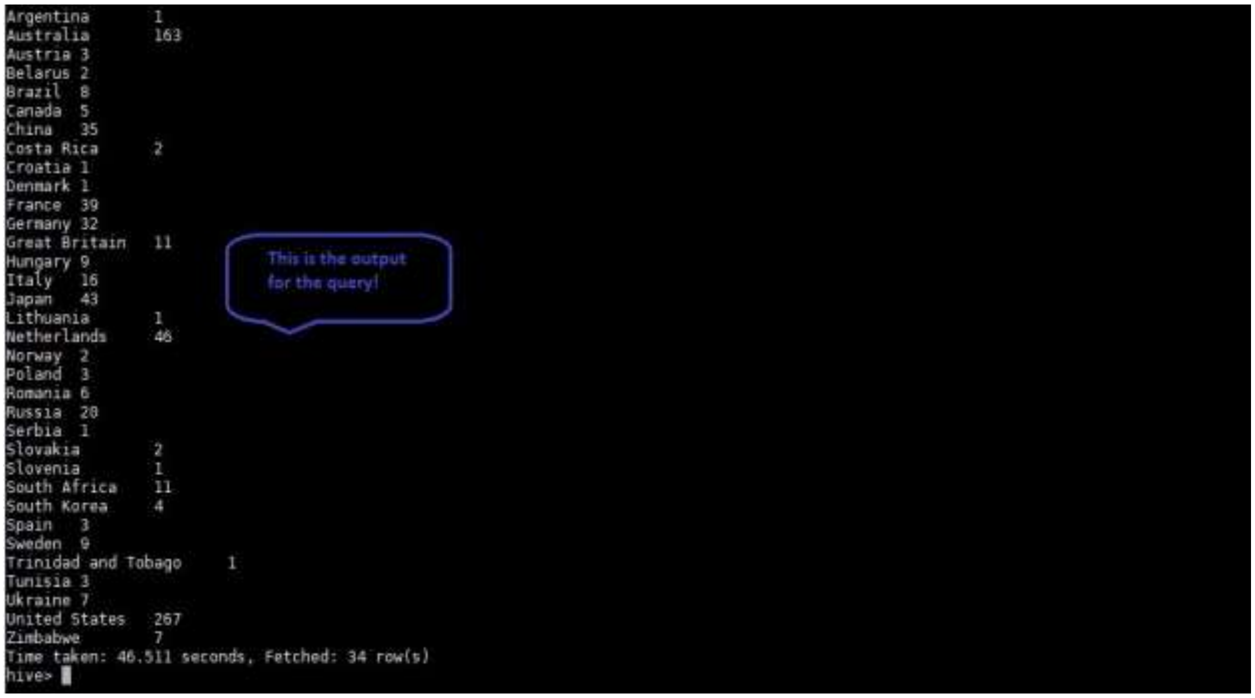
1.Write a Hive program to find the number of medals won by each country in swimming.

Below query will extract the total number of medals won by each country in swimming

Select country, sum(total) from Olympic_data where sport=”Swimming” group by country;



```
Time taken: 1.138 seconds
hive> select country,sum(total) from olympic_data where sport="Swimming" GROUP BY country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415153417_1379b057-3897-4364-860e-bad87219ddfa
Total jobs = 1
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_1523776351907_0007, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0007/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-04-15 15:34:34,638 Stage-1 map = 0%,  reduce = 0%
2018-04-15 15:34:49,268 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 5.02 sec
2018-04-15 15:34:50,338 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.24 sec
2018-04-15 15:35:02,806 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.34 sec
MapReduce Total cumulative CPU time: 8 seconds 340 msec
Ended Job = job_1523776351907_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.34 sec   HDFS Read: 535988 HDFS Write: 881 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 340 msec
OK
```

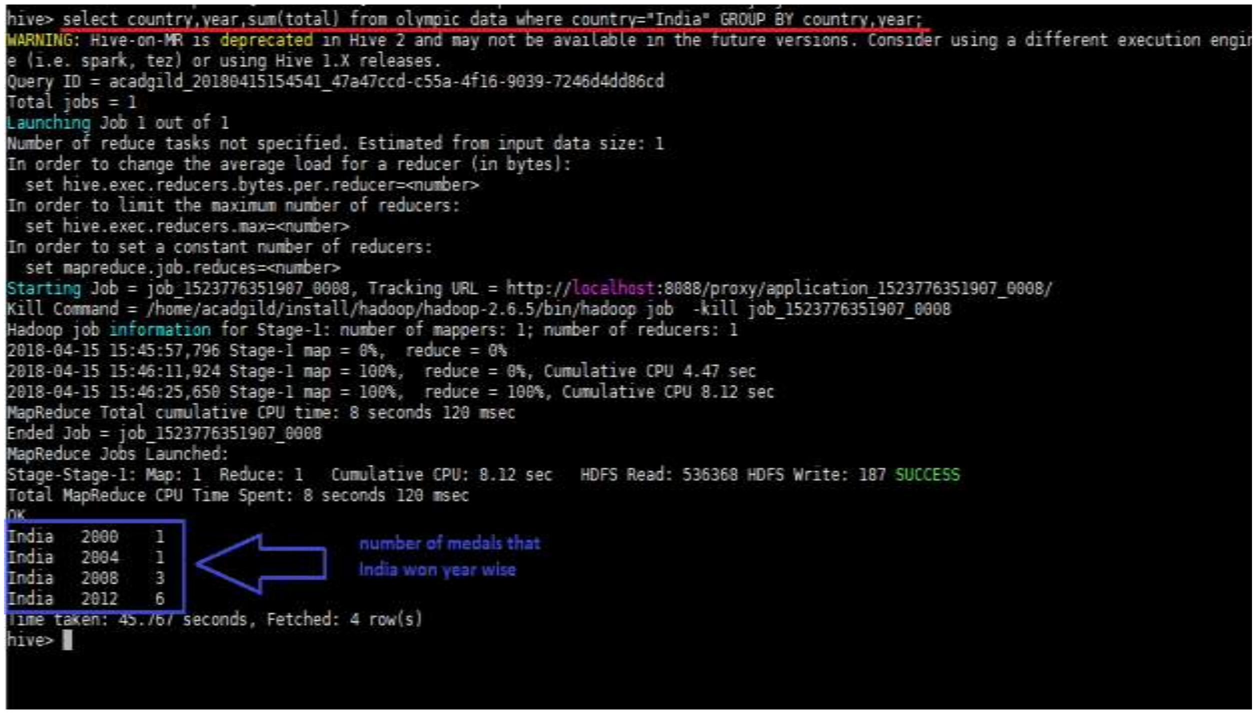


```
Argentina       1
Australia       163
Austria 3
Belarus 2
Brazil  8
Canada  5
China   35
Costa Rica      2
Croatia 1
Denmark 1
France  39
Germany 32
Great Britain   11
Hungary 9
Italy   16
Japan   43
Lithuania       1
Netherlands     46
Norway  2
Poland  3
Romania 6
Russia  20
Serbia  1
Slovakia        2
Slovenia        1
South Africa    11
South Korea     4
Spain   3
Sweden  9
Trinidad and Tobago     1
Tunisia 3
Ukraine 7
United States   267
Zimbabwe        7
Time taken: 46.511 seconds, Fetched: 34 row(s)
hive>
```

2.Write a Hive program to find the number of medals that India won year wise.

The below query will extract the total number of medal that India won every year

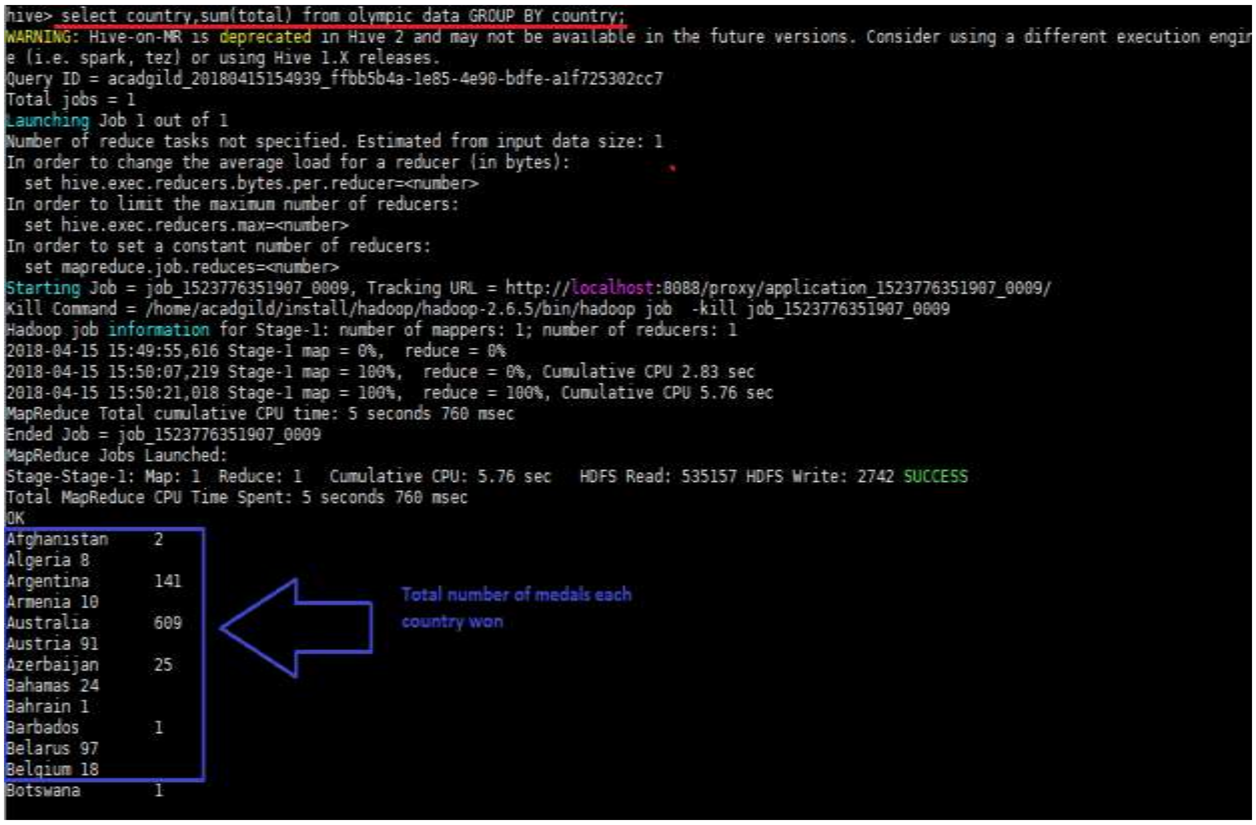
select country, year, sum(total) from olympic_data where country="India" GROUP BY country, year;



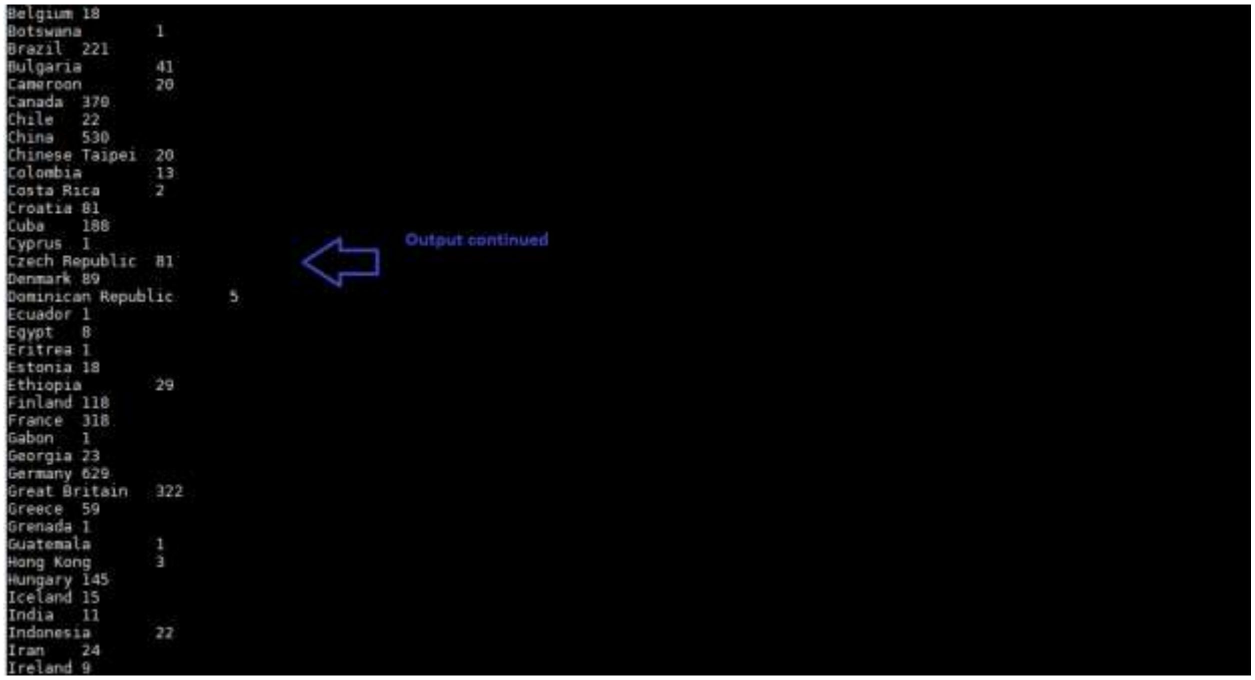
```
hive> select country,year,sum(total) from olympic data where country="India" GROUP BY country,year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engir
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415154541_47a47ccd-c55a-4f16-9039-7246d4dd86cd
Total jobs = 1
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_1523776351907_0008, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0008/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-04-15 15:45:57,796 Stage-1 map = 0%,  reduce = 0%
2018-04-15 15:46:11,924 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.47 sec
2018-04-15 15:46:25,650 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.12 sec
MapReduce Total cumulative CPU time: 8 seconds 120 msec
Ended Job = job_1523776351907_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.12 sec   HDFS Read: 536368 HDFS Write: 187 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 120 msec
OK
India   2000    1
India   2004    1
India   2008    3
India   2012    6
Time taken: 45.767 seconds, Fetched: 4 row(s)
hive>
```

3.Write a Hive Program to find the total number of medals each country won.

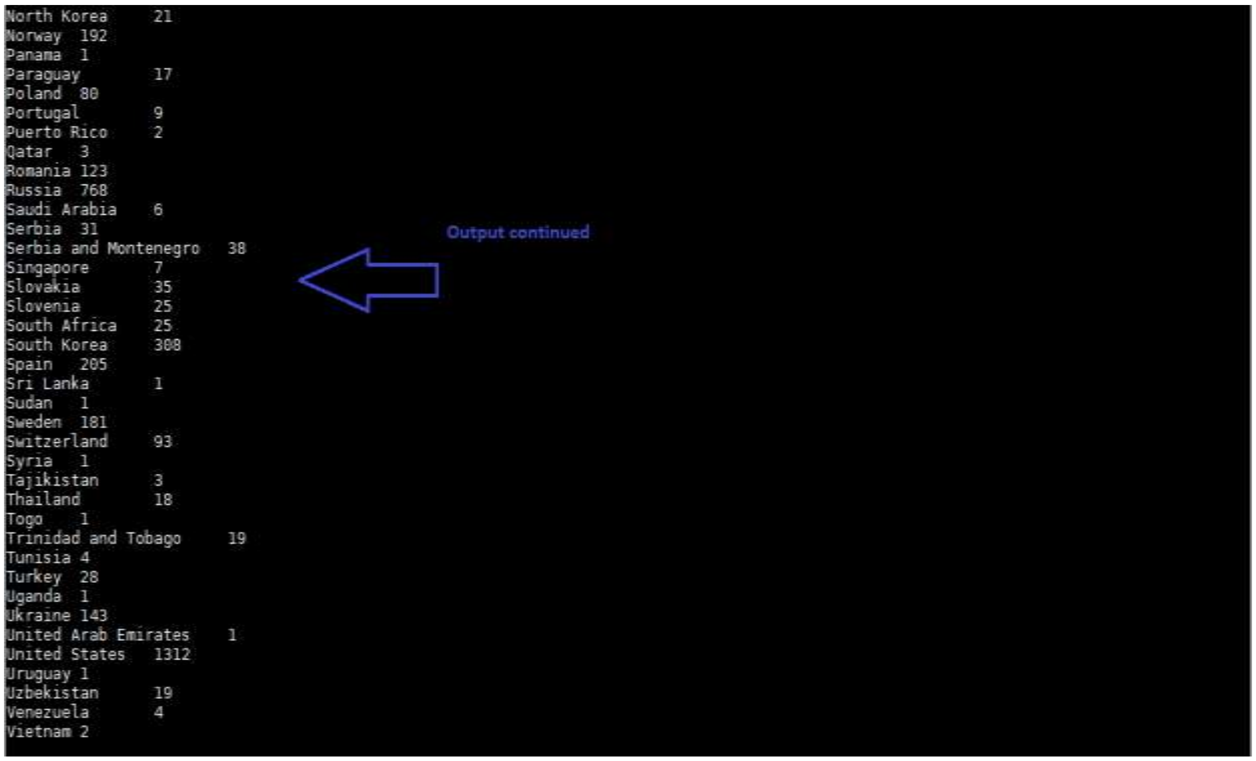
Select country, sum(total) from Olympic_data GROUP BY country;



```
hive> select country,sum(total) from olympic_data GROUP BY country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415154939_ffbb5b4a-1e85-4e90-bdfe-a1f725302cc7
Total jobs = 1
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_1523776351907_0009, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0009/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-04-15 15:49:55,616 Stage-1 map = 0%,  reduce = 0%
2018-04-15 15:50:07,219 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.83 sec
2018-04-15 15:50:21,018 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.76 sec
MapReduce Total cumulative CPU time: 5 seconds 760 msec
Ended Job = job_1523776351907_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.76 sec   HDFS Read: 535157 HDFS Write: 2742 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 760 msec
OK
Afghanistan     2
Algeria 8
Argentina       141
Armenia 10
Australia       609
Austria 91
Azerbaijan      25
Bahamas 24
Bahrain 1
Barbados        1
Belarus 97
Belgium 18
Botswana        1
```



```
Belgium 18
Botswana        1
Brazil  221
Bulgaria        41
Cameroon        20
Canada  370
Chile   22
China   530
Chinese Taipei  20
Colombia        13
Costa Rica      2
Croatia 81
Cuba    188
Cyprus  1
Czech Republic  81
Denmark 89
Dominican Republic      5
Ecuador 1
Egypt   8
Eritrea 1
Estonia 18
Ethiopia        29
Finland 118
France  318
Gabon   1
Georgia 23
Germany 629
Great Britain   322
Greece  59
Grenada 1
Guatemala       1
Hong Kong       3
Hungary 145
Iceland 15
India   11
Indonesia       22
Iran    24
Ireland 9
```

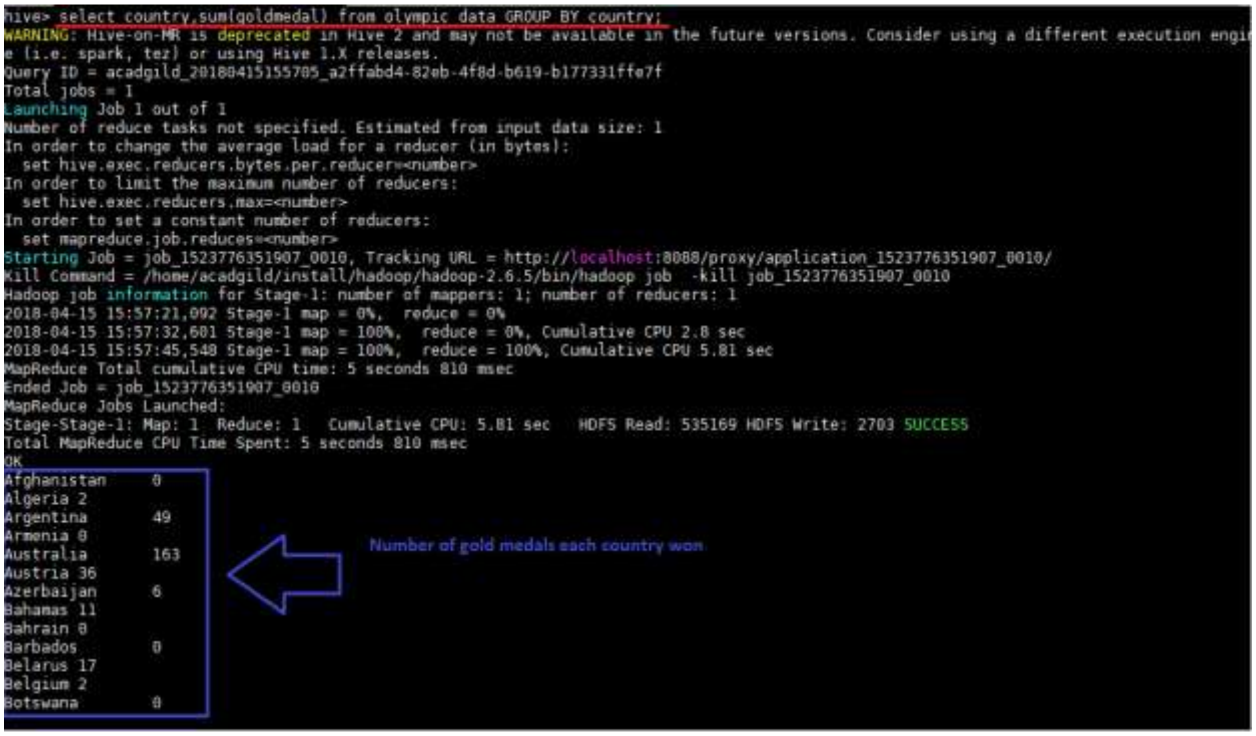


```
North Korea     21
Norway  192
Panama  1
Paraguay        17
Poland  80
Portugal        9
Puerto Rico     2
Qatar   3
Romania 123
Russia  768
Saudi Arabia    6
Serbia  31
Serbia and Montenegro   38
Singapore       7
Slovakia        35
Slovenia        25
South Africa    25
South Korea     308
Spain   205
Sri Lanka       1
Sudan   1
Sweden  181
Switzerland     93
Syria   1
Tajikistan      3
Thailand        18
Togo    1
Trinidad and Tobago     19
Tunisia 4
Turkey  28
Uganda  1
Ukraine 143
United Arab Emirates    1
United States   1312
Uruguay 1
Uzbekistan      19
Venezuela       4
Vietnam 2
```

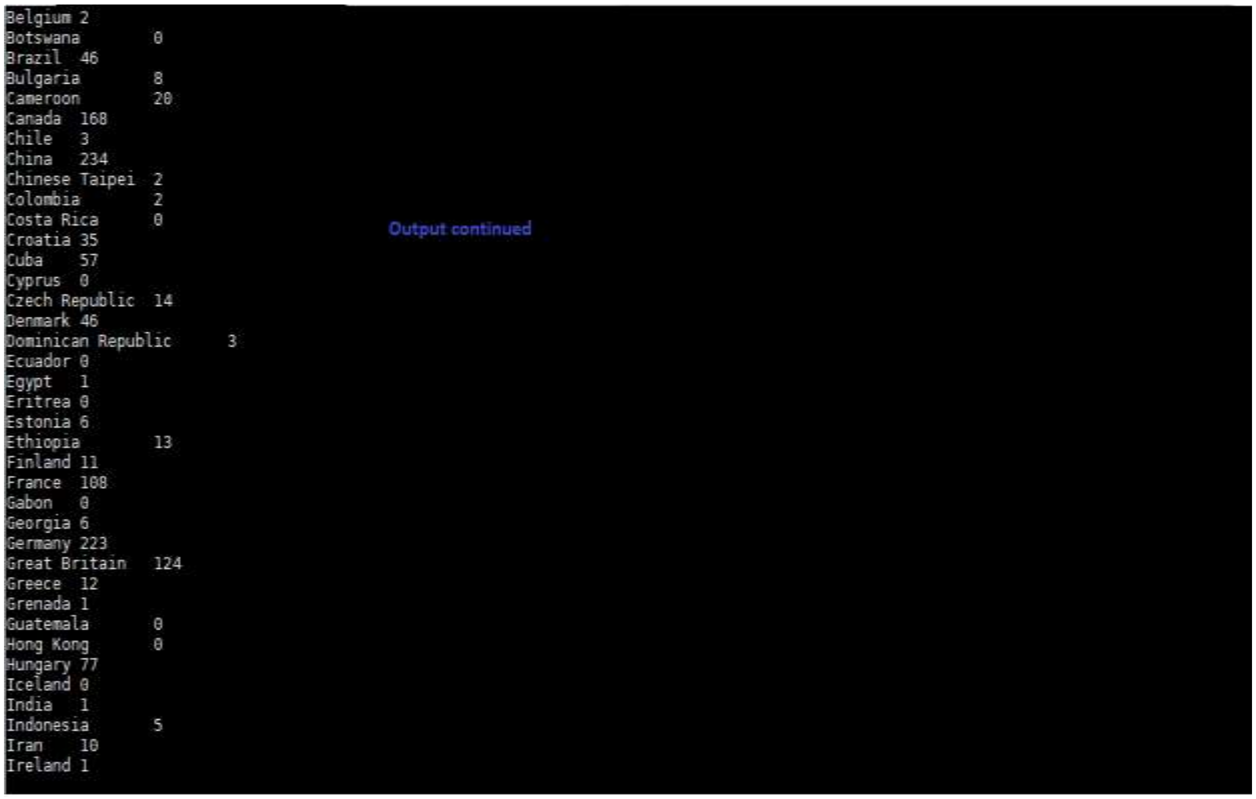
4. Write a Hive program to find the number of gold medals each country won.

The below query will extract the total gold medal won by each country.

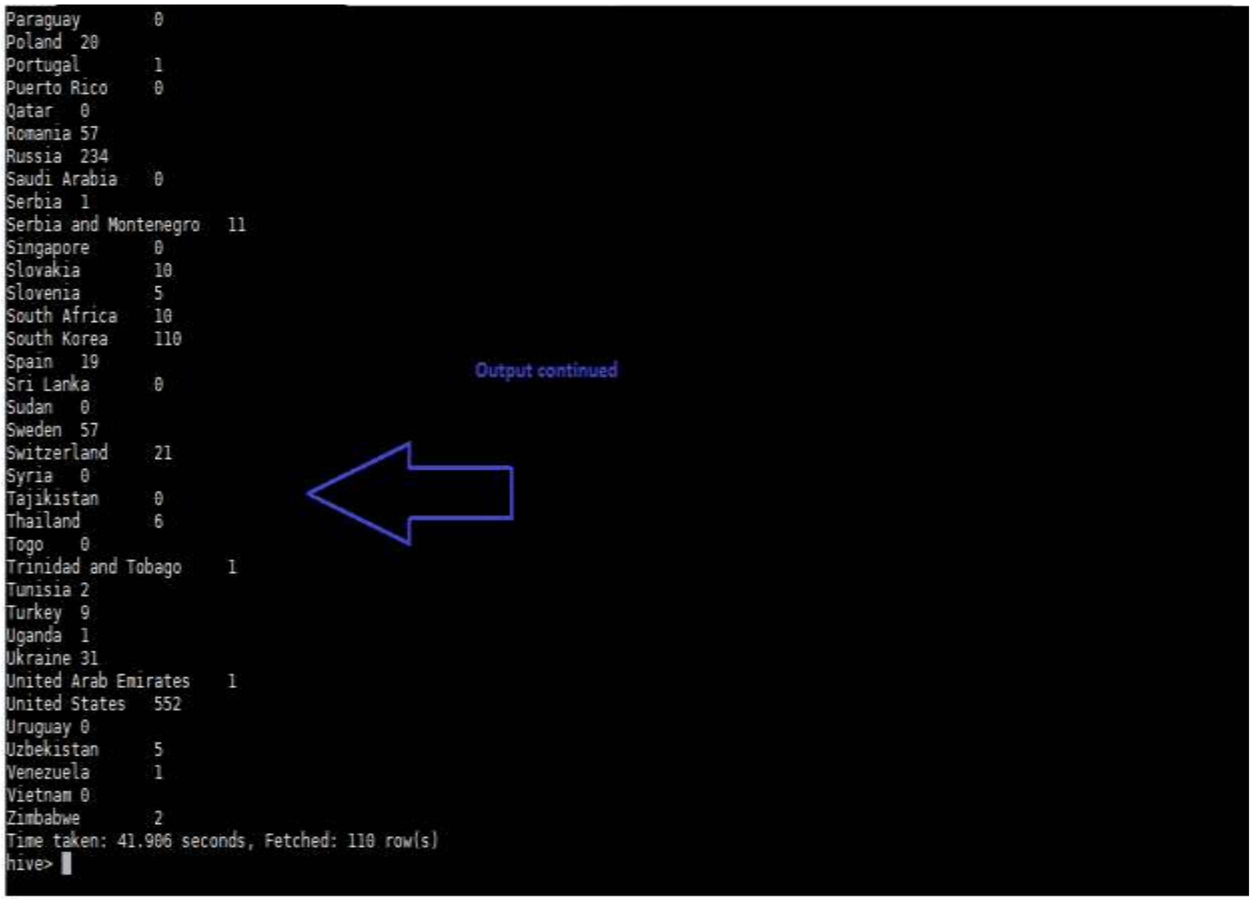
select country,sum(goldmedal) from olympic_data GROUP BY country;



```
hive> select country,sum(goldmedal) from olympic_data GROUP BY country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415155705_a2ffabd4-82eb-4f8d-b619-b177331ffe7f
Total jobs = 1
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_1523776351907_0010, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0010/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-04-15 15:57:21,092 Stage-1 map = 0%,  reduce = 0%
2018-04-15 15:57:32,601 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.8 sec
2018-04-15 15:57:45,548 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.81 sec
MapReduce Total cumulative CPU time: 5 seconds 810 msec
Ended Job = job_1523776351907_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.81 sec   HDFS Read: 535169 HDFS Write: 2703 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 810 msec
OK
Afghanistan     0
Algeria 2
Argentina       49
Armenia 0
Australia       163
Austria 36
Azerbaijan      6
Bahamas 11
Bahrain 0
Barbados        0
Belarus 17
Belgium 2
Botswana        0
```



```
Belgium 2
Botswana        0
Brazil  46
Bulgaria        8
Cameroon        20
Canada  168
Chile   3
China   234
Chinese Taipei  2
Colombia        2
Costa Rica      0
Croatia 35
Cuba    57
Cyprus  0
Czech Republic  14
Denmark 46
Dominican Republic      3
Ecuador 0
Egypt   1
Eritrea 0
Estonia 6
Ethiopia        13
Finland 11
France  108
Gabon   0
Georgia 6
Germany 223
Great Britain   124
Greece  12
Grenada 1
Guatemala       0
Hong Kong       0
Hungary 77
Iceland 0
India   1
Indonesia       5
Iran    10
Ireland 1
```



```
Paraguay        0
Poland  20
Portugal        1
Puerto Rico     0
Qatar   0
Romania 57
Russia  234
Saudi Arabia    0
Serbia  1
Serbia and Montenegro   11
Singapore       0
Slovakia        10
Slovenia        5
South Africa    10
South Korea     110
Spain   19
Sri Lanka       0
Sudan   0
Sweden  57
Switzerland     21
Syria   0
Tajikistan      0
Thailand        6
Togo    0
Trinidad and Tobago     1
Tunisia 2
Turkey  9
Uganda  1
Ukraine 31
United Arab Emirates    1
United States   552
Uruguay 0
Uzbekistan      5
Venezuela       1
Vietnam 0
Zimbabwe        2
Time taken: 41.906 seconds, Fetched: 110 row(s)
hive>
```

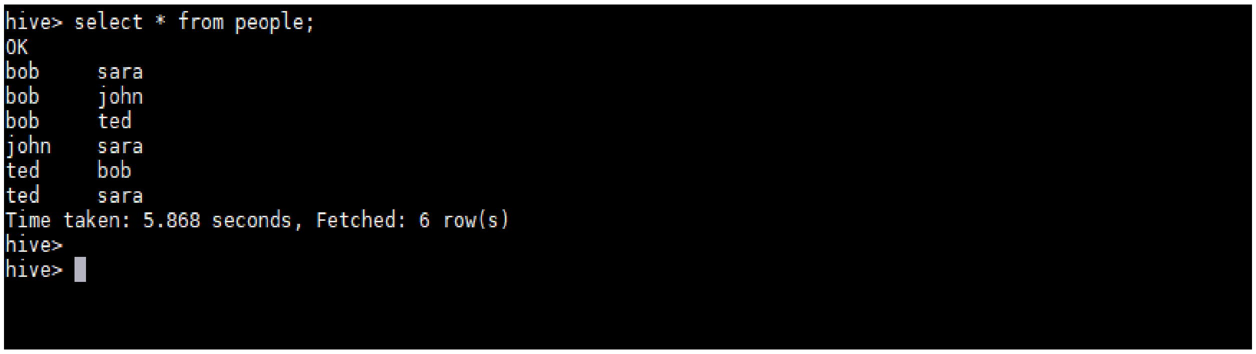
Task 2

Write a hive UDF that implements functionality of string concat_ws(string SEP, array<string>).This UDF will accept two arguments, one string and one array of string.It will return a single string where all the elements of the array are separated by the SEP.

```
package concatws;

import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.exec.UDFArgumentLengthException;
import org.apache.hadoop.hive.ql.exec.UDFArgumentTypeException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.serde.serdeConstants;
import org.apache.hadoop.hive.serde2.objectinspector.ListObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector.Category;
import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector;
import
org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector.PrimitiveCategor
y;
import
org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory;
import
org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorUtils;
import
org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorUtils.Pri
mitiveGrouping;
import org.apache.hadoop.io.Text;

public class concatenatews extends GenericUDF {
 private transient ObjectInspector[] argumentOIs;

 @Override
 public ObjectInspector initialize(ObjectInspector[] arguments) throws
UDFArgumentException {
   if (arguments.length < 2) {
     throw new UDFArgumentLengthException(
       "The function CONCAT_WS(separator,[string | array(string)]+) "
         + "needs at least two arguments.");
   }

   // check if argument is a string or an array of strings
```

```
    for (int i = 0; i < arguments.length; i++) {
      switch(arguments[i].getCategory()) {
        case LIST:
          if (isStringOrVoidType(
              ((ListObjectInspector) arguments[i]).getListElementObjectInspector())) {
            break;
          }
        case PRIMITIVE:
          if (isStringOrVoidType(arguments[i])) {
          break;
          }
        default:
          throw new UDFArgumentTypeException(i, "Argument " + (i + 1)
            + " of function CONCAT_WS must be \"" + serdeConstants.STRING_TYPE_NAME
            + " or " + serdeConstants.LIST_TYPE_NAME + "<" +
serdeConstants.STRING_TYPE_NAME
            + ">\", but \"" + arguments[i].getTypeName() + "\" was found.");
      }
    }

    argumentOIs = arguments;
    return PrimitiveObjectInspectorFactory.writableStringObjectInspector;
  }

  protected boolean isStringOrVoidType(ObjectInspector oi) {
    if (oi.getCategory() == Category.PRIMITIVE) {
      if (PrimitiveGrouping.STRING_GROUP
          == PrimitiveObjectInspectorUtils.getPrimitiveGrouping(
              ((PrimitiveObjectInspector) oi).getPrimitiveCategory())
          || ((PrimitiveObjectInspector) oi).getPrimitiveCategory() == PrimitiveCategory.VOID)
{
        return true;
      }
    }
    return false;
  }

  private final Text resultText = new Text();

  @Override
  public Object evaluate(DeferredObject[] arguments) throws HiveException {
    if (arguments[0].get() == null) {
      return null;
    }
    String separator = PrimitiveObjectInspectorUtils.getString(
        arguments[0].get(), (PrimitiveObjectInspector)argumentOIs[0]);
    StringBuilder sb = new StringBuilder();
    boolean first = true;
    for (int i = 1; i < arguments.length; i++) {
      if (arguments[i].get() != null) {
        if (first) {
          first = false;
        } else {
          sb.append(separator);
        }
        if (argumentOIs[i].getCategory().equals(Category.LIST)) {
          Object strArray = arguments[i].get();
          ListObjectInspector strArrayOI = (ListObjectInspector) argumentOIs[i];
          boolean strArrayFirst = true;
          for (int j = 0; j < strArrayOI.getListLength(strArray); j++) {
            if (strArrayFirst) {
              strArrayFirst = false;
            } else {
              sb.append(separator);
            }
            sb.append(strArrayOI.getListElement(strArray, j));
          }
        } else {
          sb.append(PrimitiveObjectInspectorUtils.getString(
              arguments[i].get(), (PrimitiveObjectInspector)argumentOIs[i]));
        }
      }
    }

    resultText.set(sb.toString());
    return resultText;
  }

  @Override
  public String getDisplayString(String[] children) {
    assert (children.length >= 2);
    return getStandardDisplayString("concat_ws", children);
  }
}
```

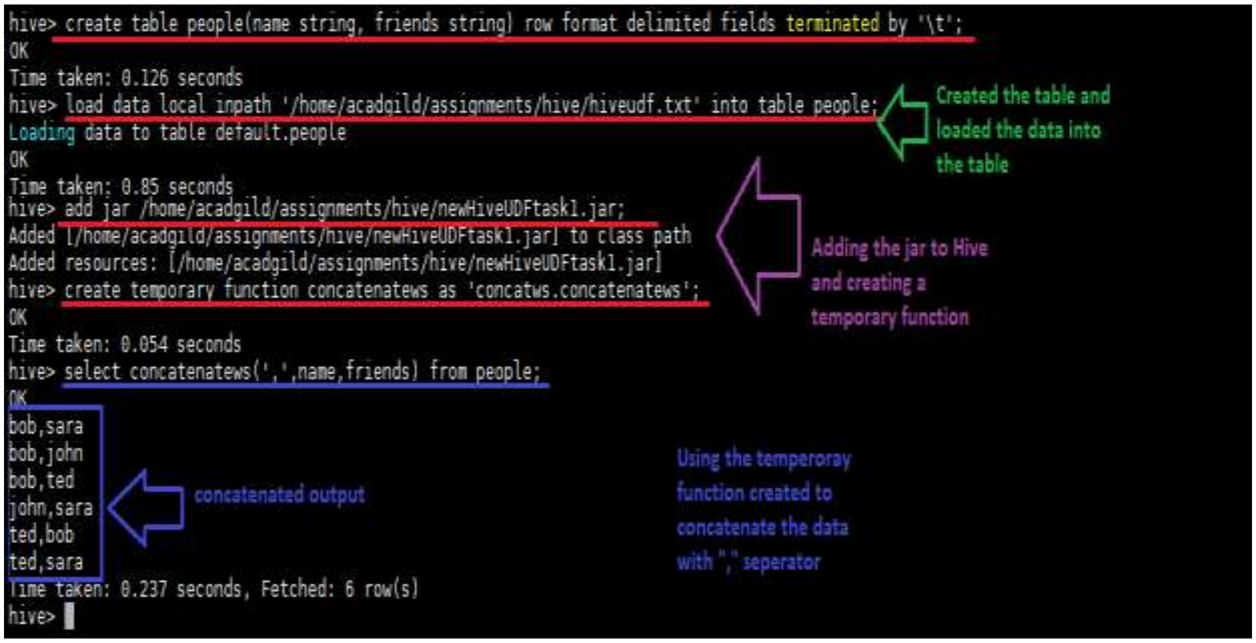
- The above program is written in eclipse and exported as jar named **"newHiveUDFtask1.jar"**
- This jar needs to added to Hive, this can be done by using the command in hive " *add jar <local path of jar file saved>*"
- We have to create a temporary function to use this function only for the current instance of hive. This can be done using the command "*create temporary function <class_name> <'package_name.class_name'>"*
- We have a table in hive "people" which has two columns "*name, friend's name"* this table represents name of a person and his/her friend's name.



```
hive> select * from people;
OK
bob     sara
bob     john
bob     ted
john    sara
ted     bob
ted     sara
Time taken: 5.868 seconds, Fetched: 6 row(s)
hive>
hive>
```

- We will use the above UDF concatenatews to concatenate the above data with "," separator.

Above steps are as shown below:



```
hive> create table people(name string, friends string) row format delimited fields terminated by '\t';
OK
Time taken: 0.126 seconds
hive> load data local inpath '/home/acadgild/assignments/hive/hiveudf.txt' into table people;
Loading data to table default.people
OK
Time taken: 0.85 seconds
hive> add jar /home/acadgild/assignments/hive/newHiveUDFtask1.jar;
Added [/home/acadgild/assignments/hive/newHiveUDFtask1.jar] to class path
Added resources: [/home/acadgild/assignments/hive/newHiveUDFtask1.jar]
hive> create temporary function concatenatews as 'concatws.concatenatews';
OK
Time taken: 0.054 seconds
hive> select concatenatews(',',name,friends) from people;
OK
bob,sara
bob,john
bob,ted
john,sara
ted,bob
ted,sara
Time taken: 0.237 seconds, Fetched: 6 row(s)
hive>
```

Created the table and loaded the data into the table
Adding the jar to Hive and creating a temporary function
concatenated output
Using the temperoray function created to concatenate the data with "," seperator

Task 3

Link: https://acadgild.com/blog/transactions-in-hive/

Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

The different row-level transactions available in Hive are as follows:

1. Insert
2. Delete
3. Update

Row-level Transactions Available in Hive:

Before creating a Hive table that supports transactions, the transaction features present in Hive needs to be turned on, as by default they are turned off.
The below properties needs to be set appropriately in ***hive shell*** , order-wise to work with transactions in Hive:
hive>set hive.support.concurrency = true;

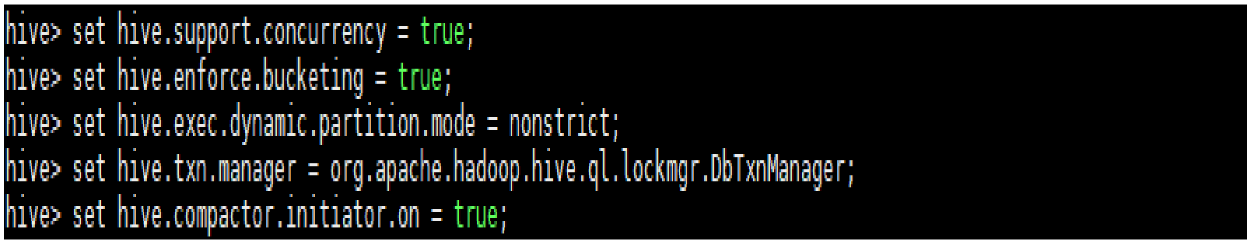
hive>set hive.enforce.bucketing = true;

hive>set hive.exec.dynamic.partition.mode = nonstrict;

hive>set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;

hive>set hive.compactor.initiator.on = true;

hive>set hive.compactor.worker.threads = a positive number on at least one instance of the Thrift metastore service;

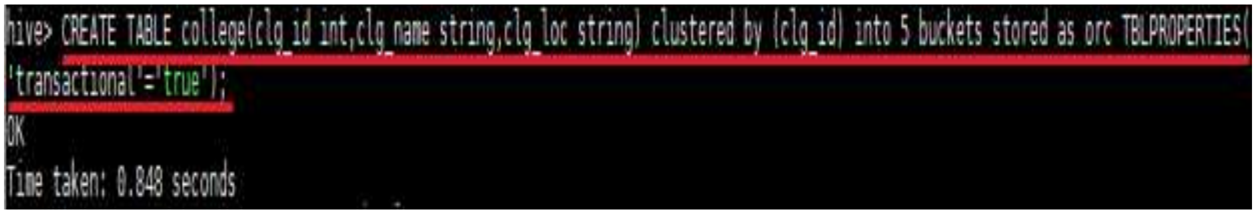


```
hive> set hive.support.concurrency = true;
hive> set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on = true;
```

Creating a Table That Supports Hive Transactions

CREATE TABLE college(clg_id int,clg_name string,clg_loc string) clustered by (clg_id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');

The above syntax will create a table with name '*college'* and the columns present in the table are '*clg_id, clg_name, clg_loc'. W*e are *bucketing* the table by '*clg_id'* and the table format is '*orc',* also we are enabling the transactions in the table by specifying it inside the *TBLPROPERTIES* as *'transactional'='true'*

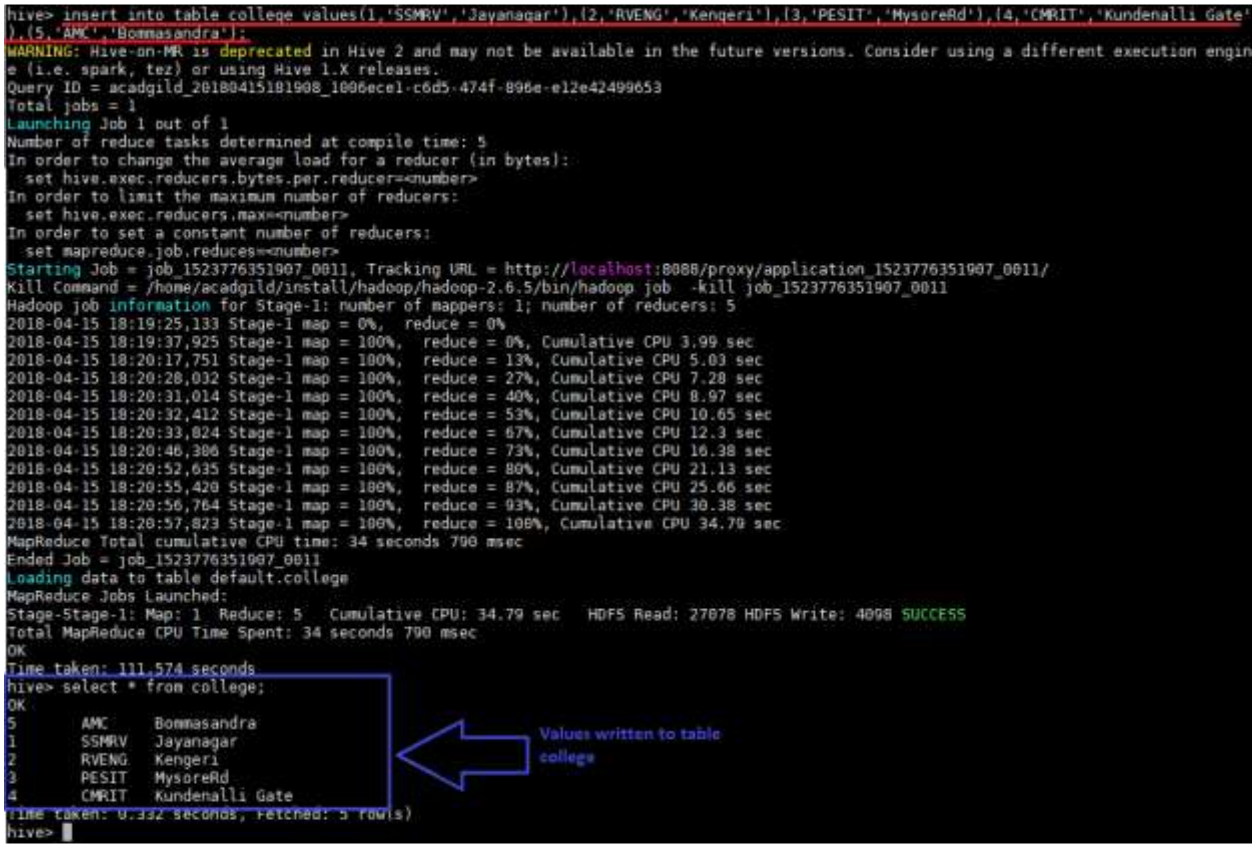


```
hive> CREATE TABLE college(clg id int,clg name string,clg loc string) clustered by (clg id) into 5 buckets stored as orc TBLPROPERTIES(
'transactional'='true');
OK
Time taken: 0.848 seconds
```

Inserting Data into a Hive Table

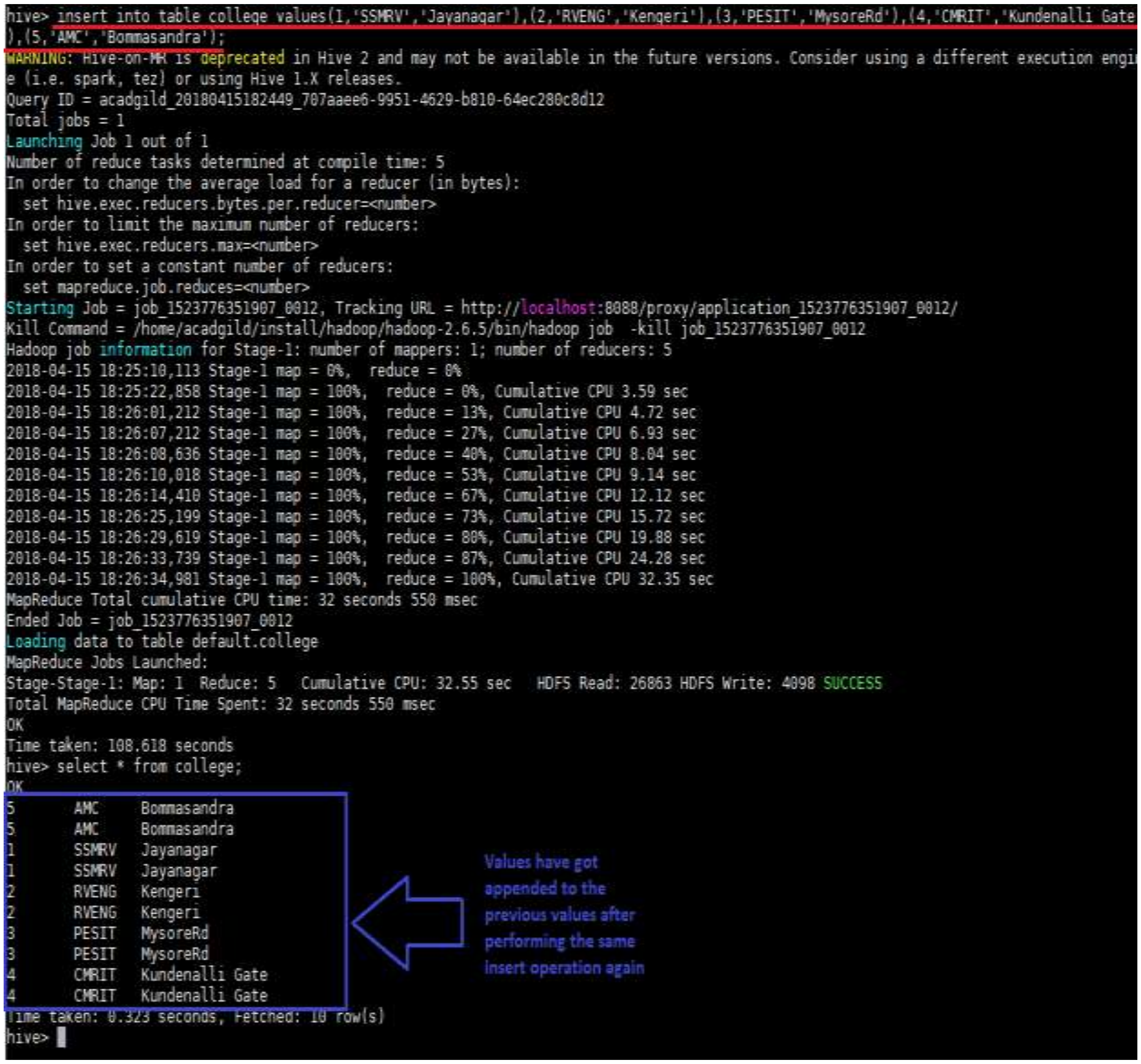
insert into table college values(1,'SSMRV','Jayanagar'),(2,'RVENG','Kengeri'),(3,'PESIT','MysoreRd'),(4,'CMRIT','Kundenalli Gate'),(5,'AMC','Bommasandra');

The above command is used to insert row wise data into the Hive table. Here, each row is separated by '*( )' brackets. The contents of the table can be viewed using the command* ***select * from college***



```
hive> insert into table college values(1,'SSMRV','Jayanagar'),(2,'RVENG','Kengeri'),(3,'PESIT','MysoreRd'),(4,'CMRIT','Kundenalli Gate'
),(5,'AMC','Bommasandra');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415181908_1006ece1-c6d5-474f-896e-e12e42499653
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1523776351907_0011, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-04-15 18:19:25,133 Stage-1 map = 0%,  reduce = 0%
2018-04-15 18:19:37,925 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.99 sec
2018-04-15 18:20:17,751 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 5.03 sec
2018-04-15 18:20:28,032 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 7.28 sec
2018-04-15 18:20:31,014 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 8.97 sec
2018-04-15 18:20:32,412 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 10.65 sec
2018-04-15 18:20:33,824 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 12.3 sec
2018-04-15 18:20:46,306 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 16.38 sec
2018-04-15 18:20:52,635 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 21.13 sec
2018-04-15 18:20:55,420 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 25.66 sec
2018-04-15 18:20:56,764 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 30.38 sec
2018-04-15 18:20:57,823 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 34.79 sec
MapReduce Total cumulative CPU time: 34 seconds 790 msec
Ended Job = job_1523776351907_0011
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 34.79 sec   HDFS Read: 27078 HDFS Write: 4098 SUCCESS
Total MapReduce CPU Time Spent: 34 seconds 790 msec
OK
Time taken: 111.574 seconds
hive> select * from college;
OK
5       AMC     Bommasandra
1       SSMRV   Jayanagar
2       RVENG   Kengeri
3       PESIT   MysoreRd
4       CMRIT   Kundenalli Gate
Time taken: 0.332 seconds, Fetched: 5 row(s)
hive>
```

we will re-insert the same data again, it will be appended to the previous data as shown below:



```
hive> insert into table college values(1,'SSMRV','Jayanagar'),(2,'RVENG','Kengeri'),(3,'PESIT','MysoreRd'),(4,'CMRIT','Kundenalli Gate
),(5,'AMC','Bommasandra');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415182449_707aaee6-9951-4629-b810-64ec280c8d12
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1523776351907_0012, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0012/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-04-15 18:25:10,113 Stage-1 map = 0%,  reduce = 0%
2018-04-15 18:25:22,858 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.59 sec
2018-04-15 18:26:01,212 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 4.72 sec
2018-04-15 18:26:07,212 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 6.93 sec
2018-04-15 18:26:08,636 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 8.04 sec
2018-04-15 18:26:10,018 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 9.14 sec
2018-04-15 18:26:14,410 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 12.12 sec
2018-04-15 18:26:25,199 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 15.72 sec
2018-04-15 18:26:29,619 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 19.88 sec
2018-04-15 18:26:33,739 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 24.28 sec
2018-04-15 18:26:34,981 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 32.35 sec
MapReduce Total cumulative CPU time: 32 seconds 550 msec
Ended Job = job_1523776351907_0012
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 32.55 sec   HDFS Read: 26863 HDFS Write: 4098 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 550 msec
OK
Time taken: 108.618 seconds
hive> select * from college;
OK
5       AMC     Bommasandra
5       AMC     Bommasandra
1       SSMRV   Jayanagar
1       SSMRV   Jayanagar
2       RVENG   Kengeri
2       RVENG   Kengeri
3       PESIT   MysoreRd
3       PESIT   MysoreRd
4       CMRIT   Kundenalli Gate
4       CMRIT   Kundenalli Gate
Time taken: 0.323 seconds, Fetched: 10 row(s)
hive> 
```

Values have got appended to the previous values after performing the same insert operation again

Updating the Data in Hive Table

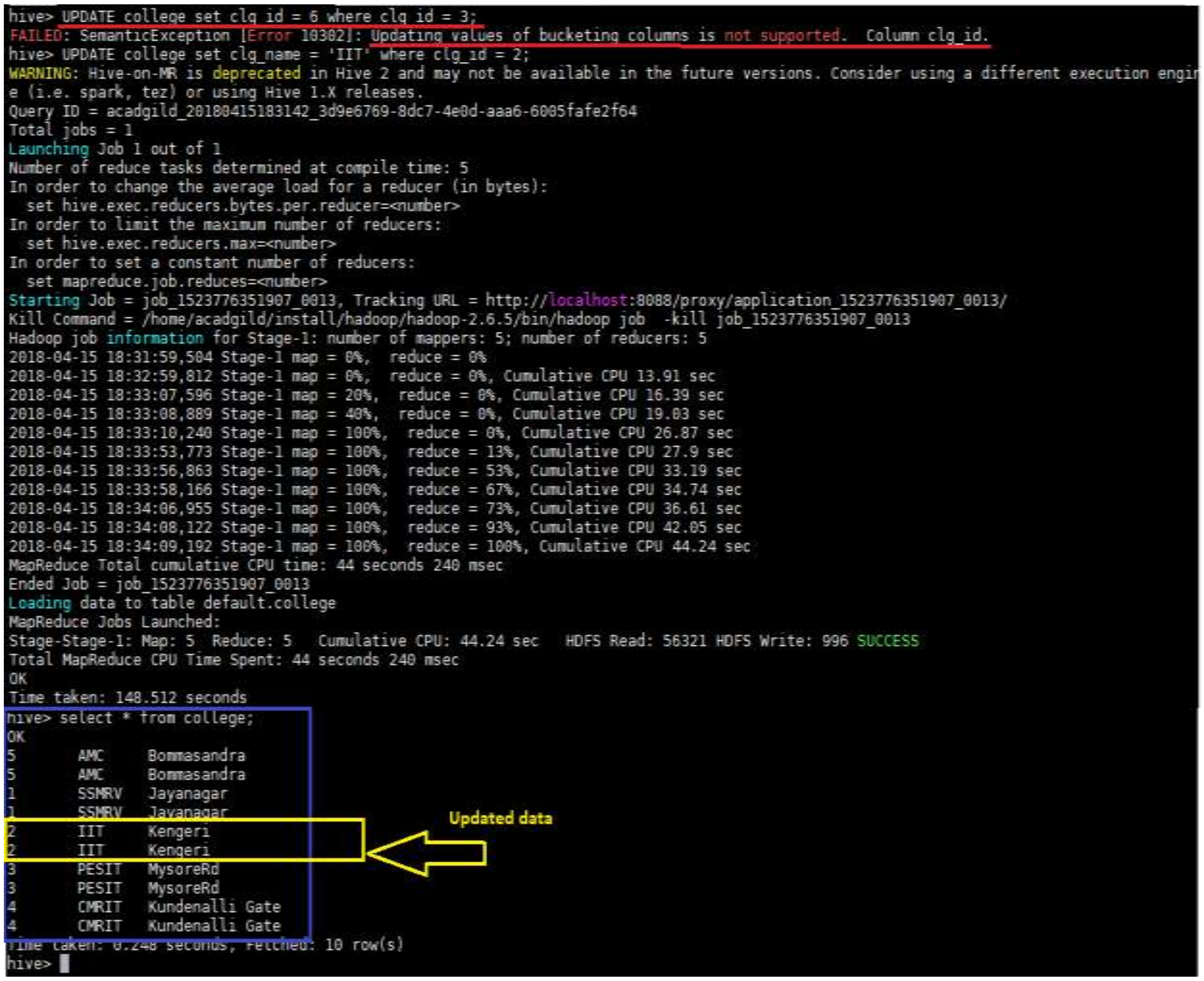
UPDATE college set clg_id = 6 where clg_id = 3;//not supported because of bucketing

we can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed. we have bucketed the ***'clg_id'*** column and performing the Update operation on the same column, so we got the error.

But we can perform **update operation on Non bucketed column**

UPDATE college set clg_name = 'IIT' where clg_id = 2;

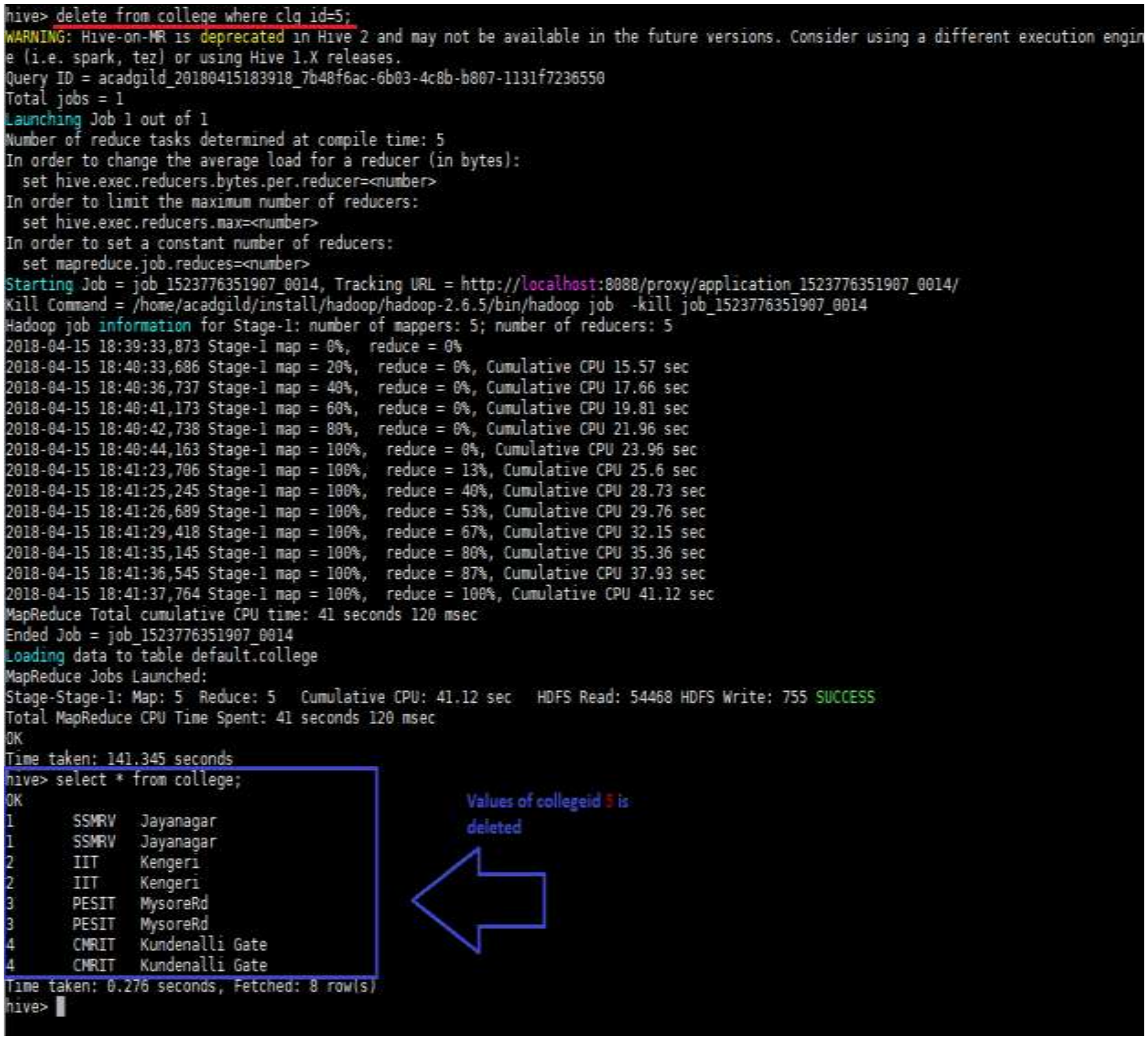
The updated data can be checked using the command ***select * from college.***



```
hive> UPDATE college set clg id = 6 where clg id = 3;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported.  Column clg_id.
hive> UPDATE college set clg_name = 'IIT' where clg_id = 2;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415183142_3d9e6769-8dc7-4e0d-aaa6-6005fafe2f64
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1523776351907_0013, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0013/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0013
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-04-15 18:31:59,504 Stage-1 map = 0%,  reduce = 0%
2018-04-15 18:32:59,812 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 13.91 sec
2018-04-15 18:33:07,596 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 16.39 sec
2018-04-15 18:33:08,889 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 19.03 sec
2018-04-15 18:33:10,240 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 26.87 sec
2018-04-15 18:33:53,773 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 27.9 sec
2018-04-15 18:33:56,863 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 33.19 sec
2018-04-15 18:33:58,166 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 34.74 sec
2018-04-15 18:34:06,955 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 36.61 sec
2018-04-15 18:34:08,122 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 42.05 sec
2018-04-15 18:34:09,192 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 44.24 sec
MapReduce Total cumulative CPU time: 44 seconds 240 msec
Ended Job = job_1523776351907_0013
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 44.24 sec   HDFS Read: 56321 HDFS Write: 996 SUCCESS
Total MapReduce CPU Time Spent: 44 seconds 240 msec
OK
Time taken: 148.512 seconds
hive> select * from college;
OK
5       AMC     Bommasandra
5       AMC     Bommasandra
1       SSMRV   Jayanagar
1       SSMRV   Jayanagar
2       IIT     Kengeri
2       IIT     Kengeri
3       PESIT   MysoreRd
3       PESIT   MysoreRd
4       CMRIT   Kundenalli Gate
4       CMRIT   Kundenalli Gate
Time taken: 0.248 seconds, Fetched: 10 row(s)
hive>
```

Deleting a Row from Hive Table:

delete from college where clg_id=5;



```
hive> delete from college where clg id=5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engin
e (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180415183918_7b48f6ac-6b03-4c8b-b807-1131f7236550
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1523776351907_0014, Tracking URL = http://localhost:8088/proxy/application_1523776351907_0014/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523776351907_0014
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-04-15 18:39:33,873 Stage-1 map = 0%,  reduce = 0%
2018-04-15 18:40:33,686 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 15.57 sec
2018-04-15 18:40:36,737 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 17.66 sec
2018-04-15 18:40:41,173 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 19.81 sec
2018-04-15 18:40:42,738 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 21.96 sec
2018-04-15 18:40:44,163 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 23.96 sec
2018-04-15 18:41:23,706 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 25.6 sec
2018-04-15 18:41:25,245 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 28.73 sec
2018-04-15 18:41:26,689 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 29.76 sec
2018-04-15 18:41:29,418 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 32.15 sec
2018-04-15 18:41:35,145 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 35.36 sec
2018-04-15 18:41:36,545 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 37.93 sec
2018-04-15 18:41:37,764 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 41.12 sec
MapReduce Total cumulative CPU time: 41 seconds 120 msec
Ended Job = job_1523776351907_0014
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 41.12 sec   HDFS Read: 54468 HDFS Write: 755 SUCCESS
Total MapReduce CPU Time Spent: 41 seconds 120 msec
OK
Time taken: 141.345 seconds
hive> select * from college;
OK
1       SSMRV   Jayanagar
1       SSMRV   Jayanagar
2       IIT     Kengeri
2       IIT     Kengeri
3       PESIT   MysoreRd
3       PESIT   MysoreRd
4       CMRIT   Kundenalli Gate
4       CMRIT   Kundenalli Gate
Time taken: 0.276 seconds, Fetched: 8 row(s)
hive> 
```

We have now successfully deleted a row from the Hive table. This can be checked using the command **select * from college, as shown above.** We can see that there is no row with ***clg_id =5***.