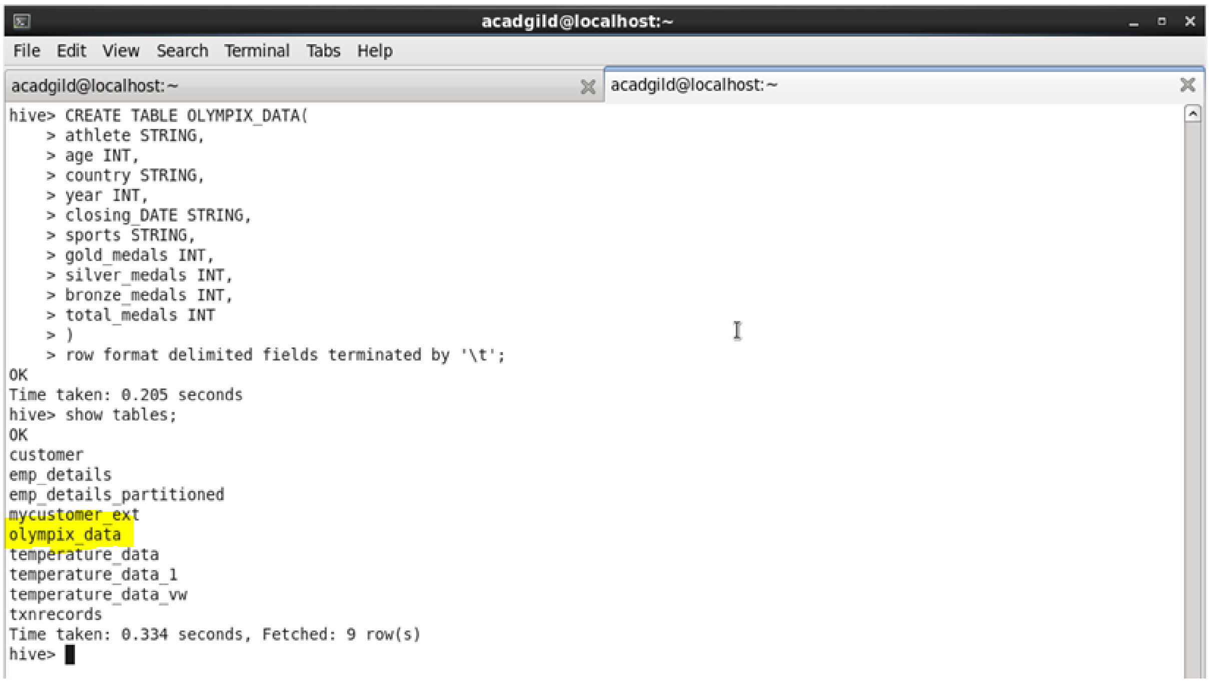
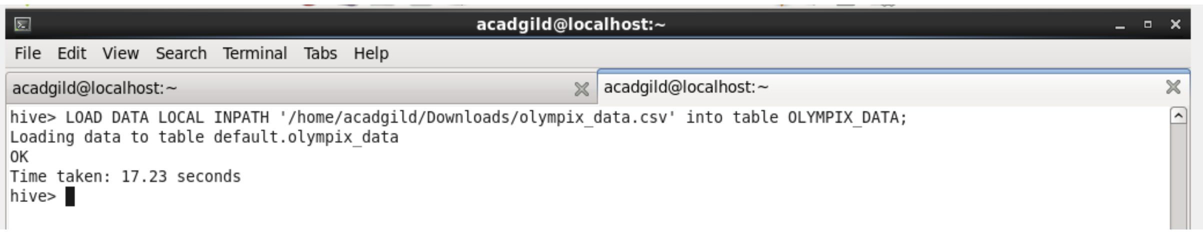
Assignment-9

Table with the name **OLYMPIX_DATA** has been created with the fields given in the assignment.



```
hive> CREATE TABLE OLYMPIX_DATA(
    > athlete STRING,
    > age INT,
    > country STRING,
    > year INT,
    > closing_DATE STRING,
    > sports STRING,
    > gold_medals INT,
    > silver_medals INT,
    > bronze_medals INT,
    > total_medals INT
    > )
    > row format delimited fields terminated by '\t';
OK
Time taken: 0.205 seconds
hive> show tables;
OK
customer
emp_details
emp_details_partitioned
mycustomer_ext
olympix_data
temperature_data
temperature_data_1
temperature_data_vw
txnrecords
Time taken: 0.334 seconds, Fetched: 9 row(s)
hive>
```

Dataset is loaded in to the table **OLYMPIX_DATA** as shown in the below screenshot.



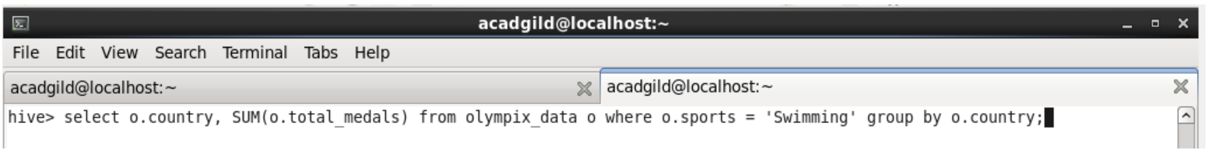
```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/olympix_data.csv' into table OLYMPIX_DATA;
Loading data to table default.olympix_data
OK
Time taken: 17.23 seconds
hive>
```

Task-1:

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

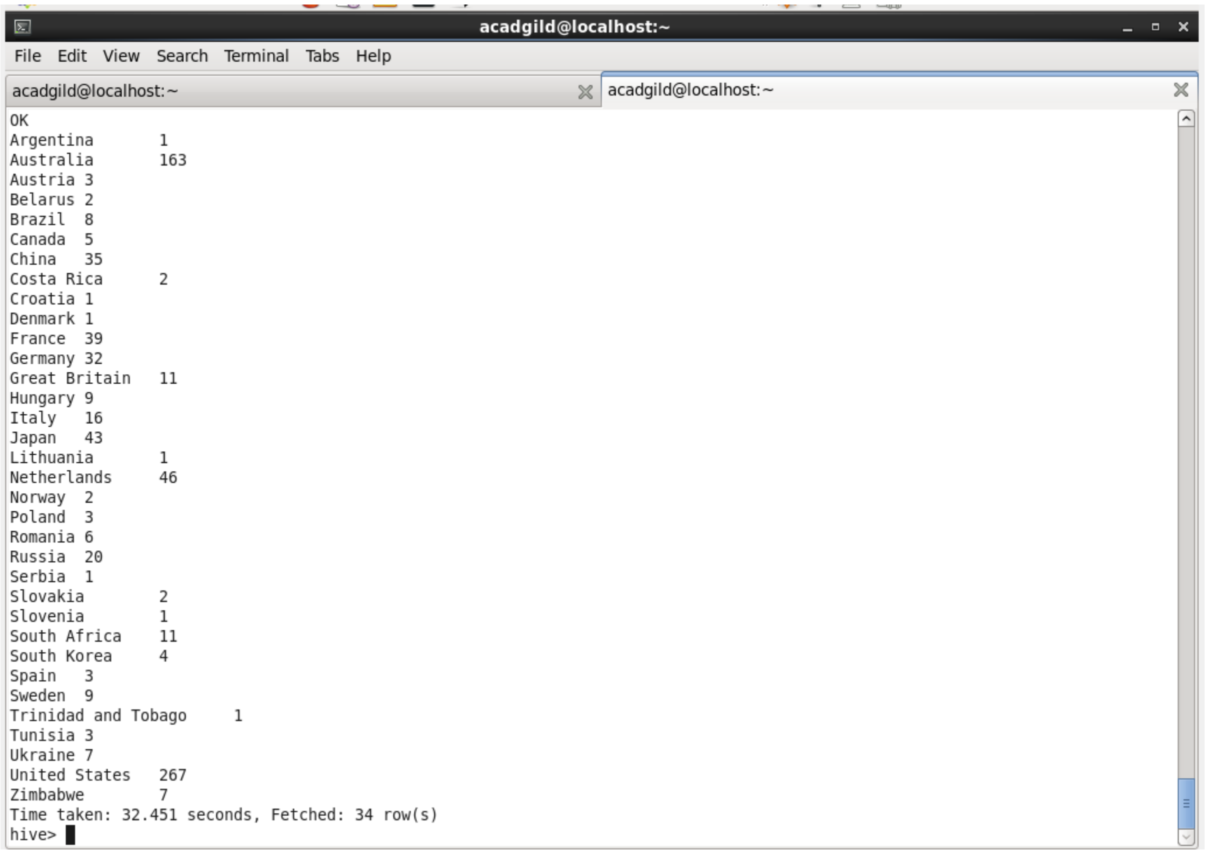
To find the total number of medals won by each country in “**Swimming**” is achieved by using the below query.

Total medals for each country is achieved by grouping data on “**Country**” and used condition as sport equal to “**Swimming**”.



```
hive> select o.country, SUM(o.total_medals) from olympix_data o where o.sports = 'Swimming' group by o.country;
```

Total medals won for each country achieved from the query used above is as shown in the below screenshot.



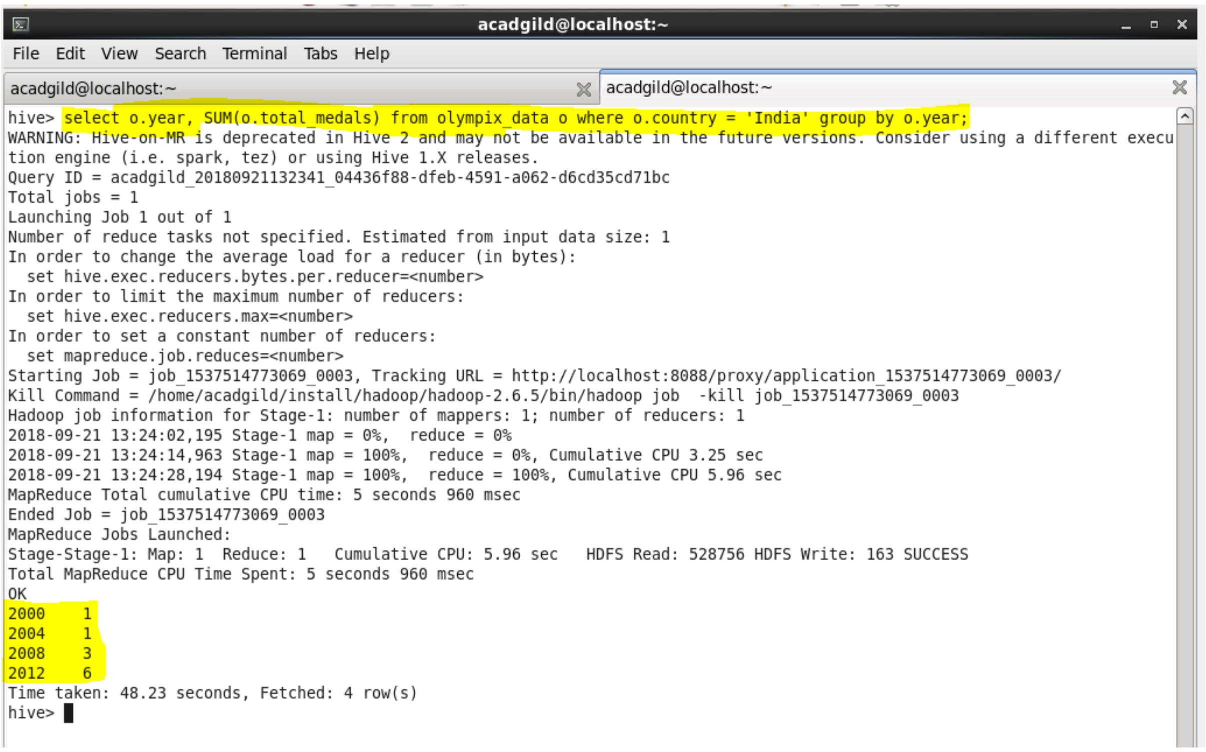
```
OK
Argentina       1
Australia       163
Austria 3
Belarus 2
Brazil  8
Canada  5
China   35
Costa Rica      2
Croatia 1
Denmark 1
France  39
Germany 32
Great Britain   11
Hungary 9
Italy   16
Japan   43
Lithuania       1
Netherlands     46
Norway  2
Poland  3
Romania 6
Russia  20
Serbia  1
Slovakia        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: 32.451 seconds, Fetched: 34 row(s)
hive> 
```

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

To find the total number of medals won by “**India**” over each “**year**” is achieved by using the below query.

Total medals India won is achieved by grouping data on “**Year**” and used condition as country equal to “**India**”.

Data retrieved from the query is shown in the below screenshot with highlighted.

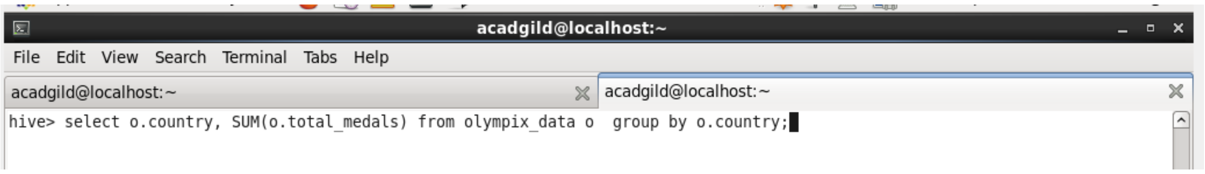


```
hive> select o.year, SUM(o.total_medals) from olympix_data o where o.country = 'India' group by o.year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180921132341_04436f88-dfeb-4591-a062-d6cd35cd71bc
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_1537514773069_0003, Tracking URL = http://localhost:8088/proxy/application_1537514773069_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537514773069_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-21 13:24:02,195 Stage-1 map = 0%,  reduce = 0%
2018-09-21 13:24:14,963 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.25 sec
2018-09-21 13:24:28,194 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.96 sec
MapReduce Total cumulative CPU time: 5 seconds 960 msec
Ended Job = job_1537514773069_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.96 sec   HDFS Read: 528756 HDFS Write: 163 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 960 msec
OK
2000    1
2004    1
2008    3
2012    6
Time taken: 48.23 seconds, Fetched: 4 row(s)
hive> 
```

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

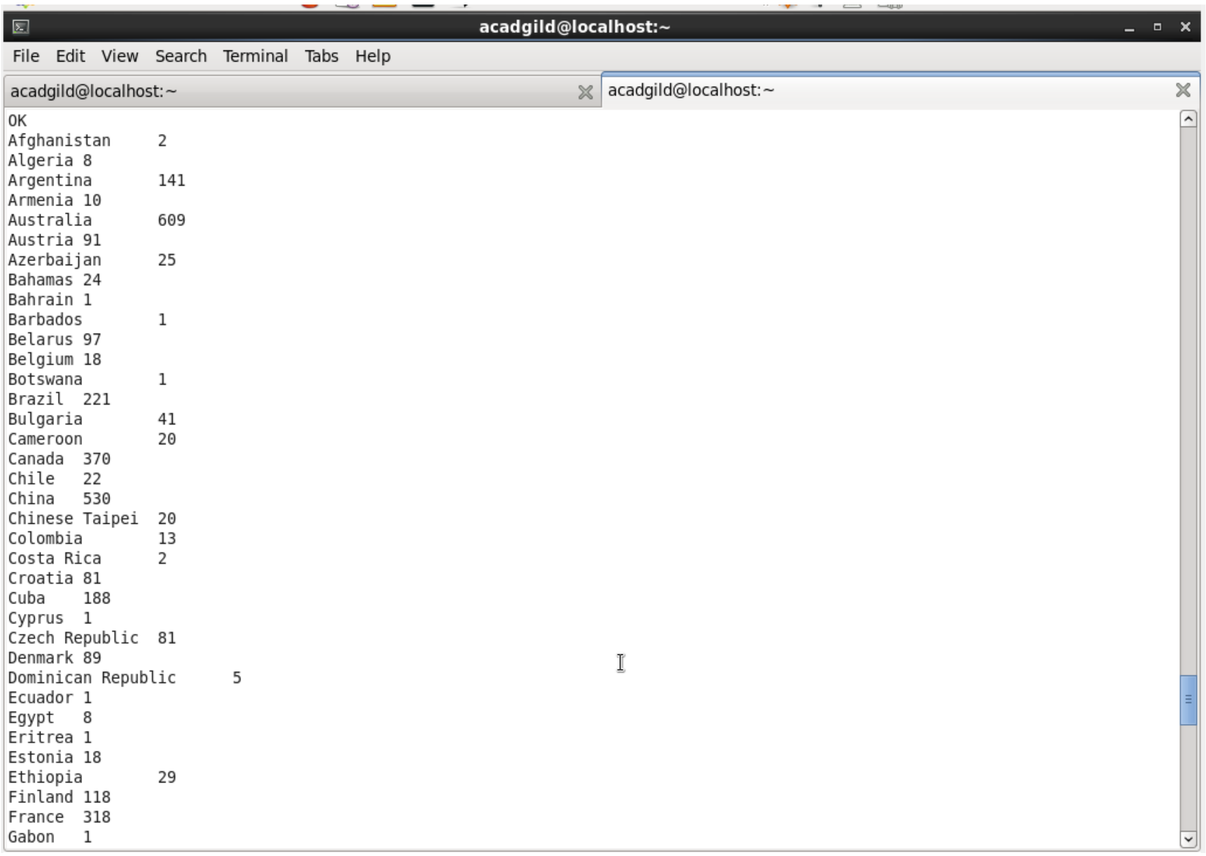
To find the total number of medals won by each country is achieved by using the below query.

Total medals won for each country is achieved by grouping data on “**Country**”.

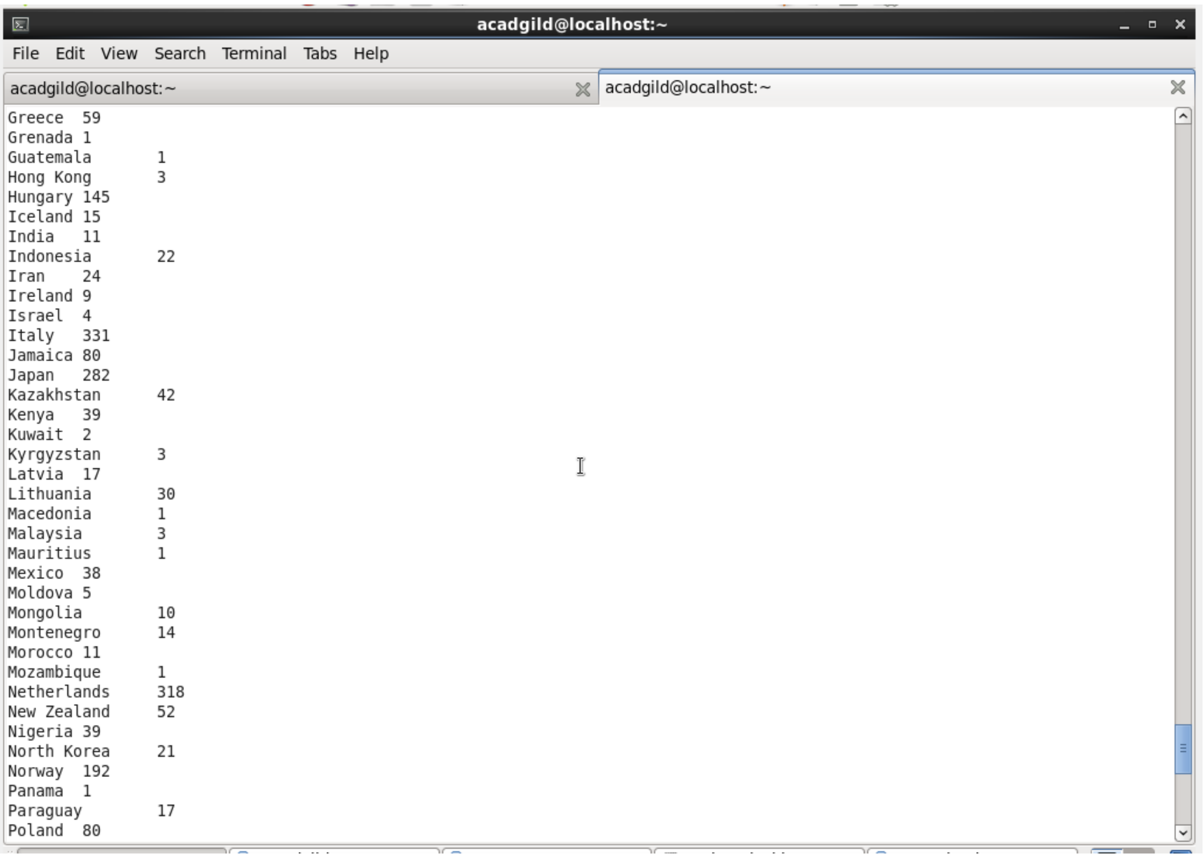


```
hive> select o.country, SUM(o.total_medals) from olympix_data o  group by o.country;
```

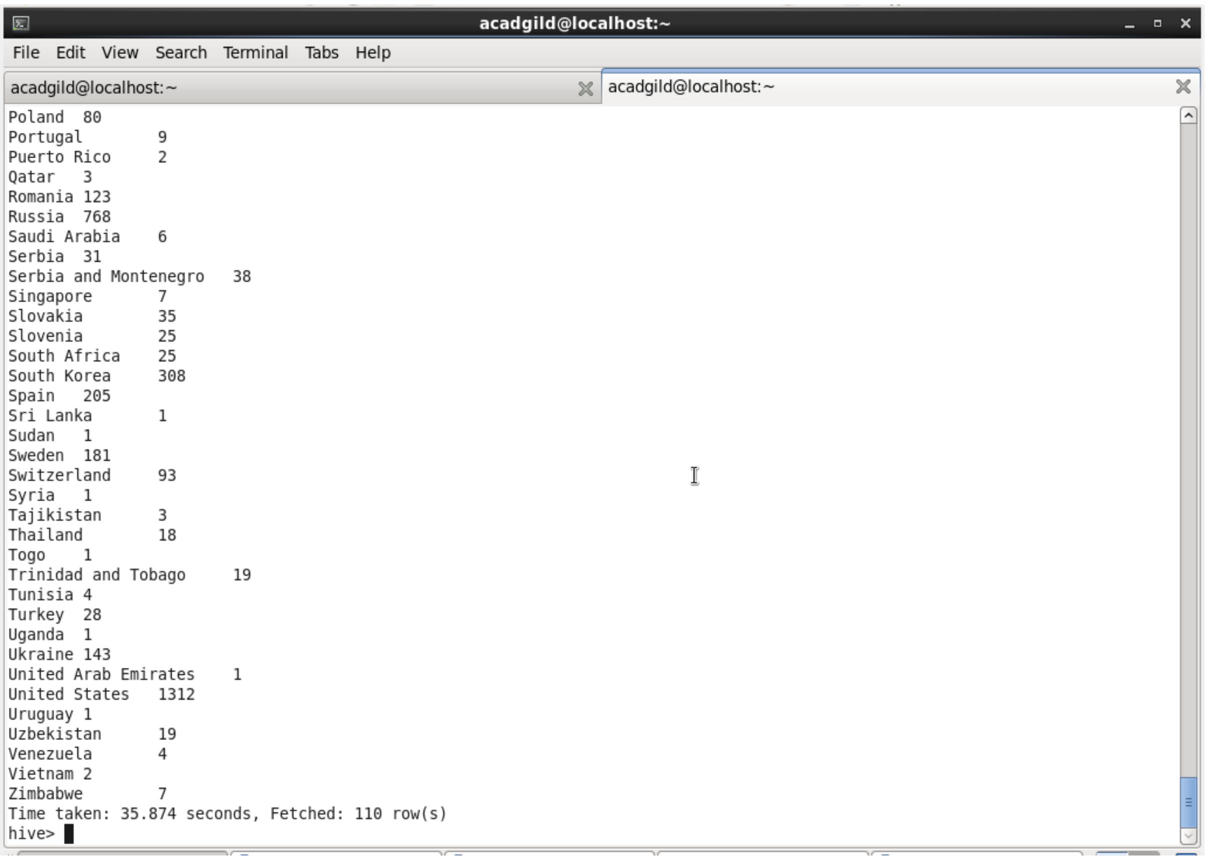
Total medals won for each country achieved from the query used above is as shown in the below screenshots.



```
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
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
```



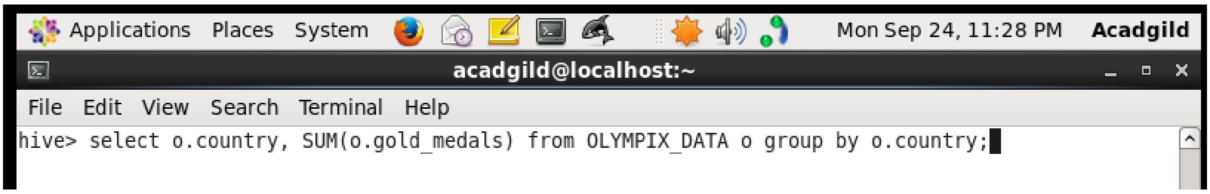
```
Greece  59
Grenada 1
Guatemala       1
Hong Kong       3
Hungary 145
Iceland 15
India   11
Indonesia       22
Iran    24
Ireland 9
Israel  4
Italy   331
Jamaica 80
Japan   282
Kazakhstan      42
Kenya   39
Kuwait  2
Kyrgyzstan      3
Latvia  17
Lithuania       30
Macedonia       1
Malaysia        3
Mauritius       1
Mexico  38
Moldova 5
Mongolia        10
Montenegro      14
Morocco 11
Mozambique      1
Netherlands     318
New Zealand     52
Nigeria 39
North Korea     21
Norway  192
Panama  1
Paraguay        17
Poland  80
```



```
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
Zimbabwe        7
Time taken: 35.874 seconds, Fetched: 110 row(s)
hive> 
```

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

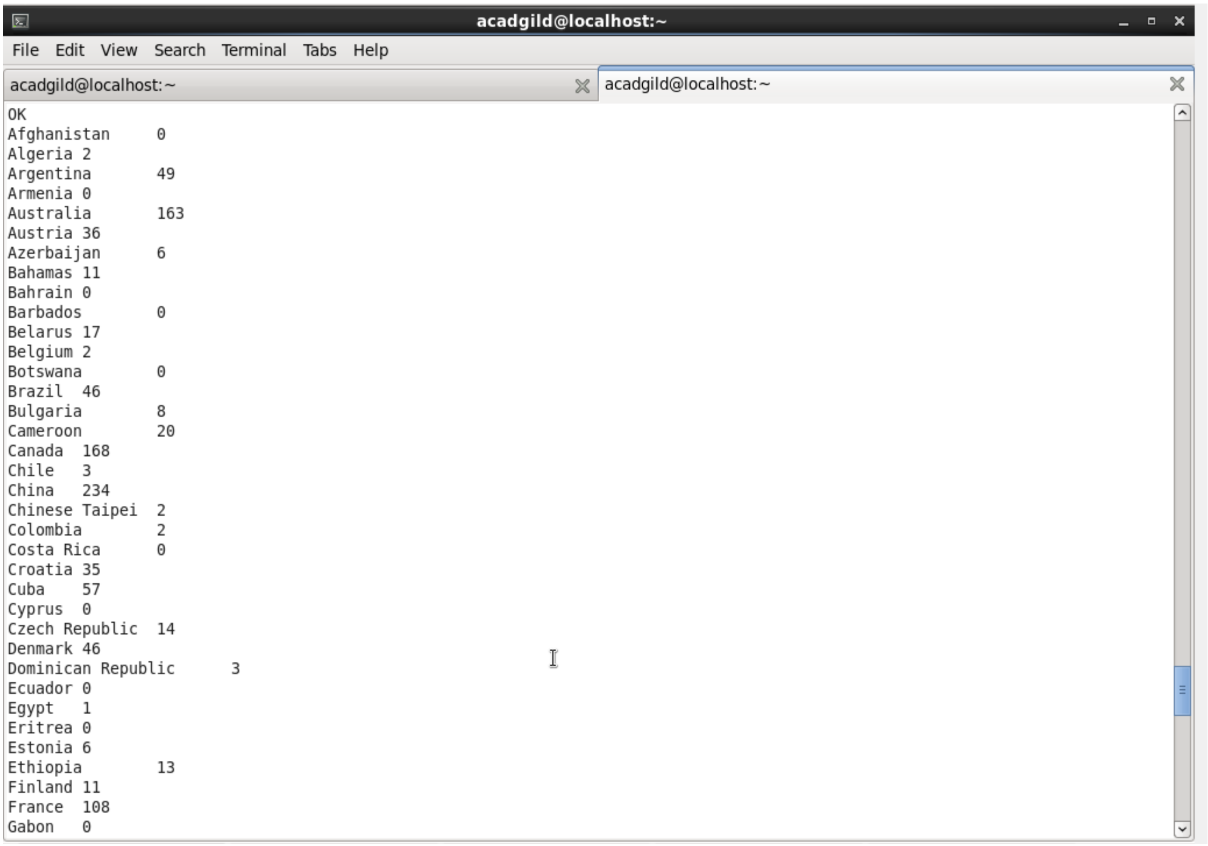
To find the total number of gold medals won by each country is achieved by using the below query.



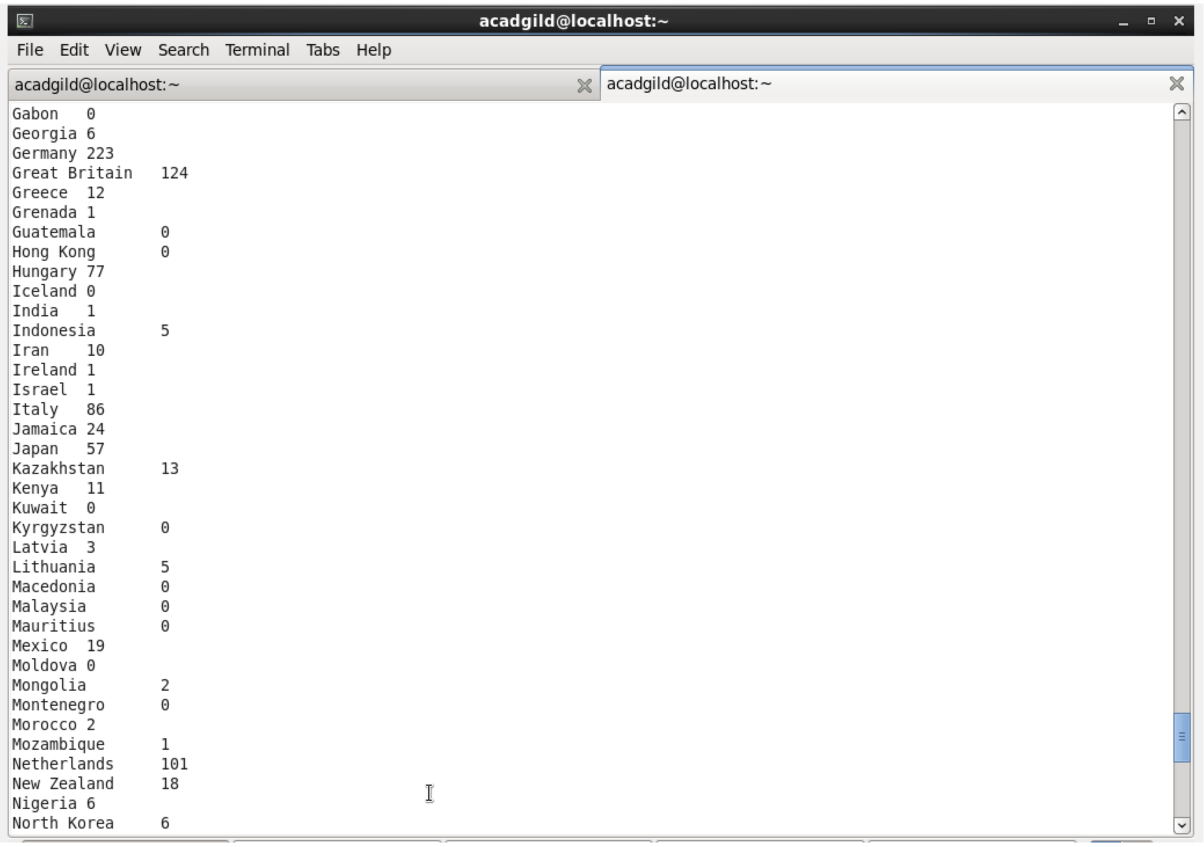
```
hive> select o.country, SUM(o.gold_medals) from OLYMPIX_DATA o group by o.country;
```

Total gold medals won by each country is achieved by grouping data on "**Country**" and used "**SUM(gold_medals)**" to get count.

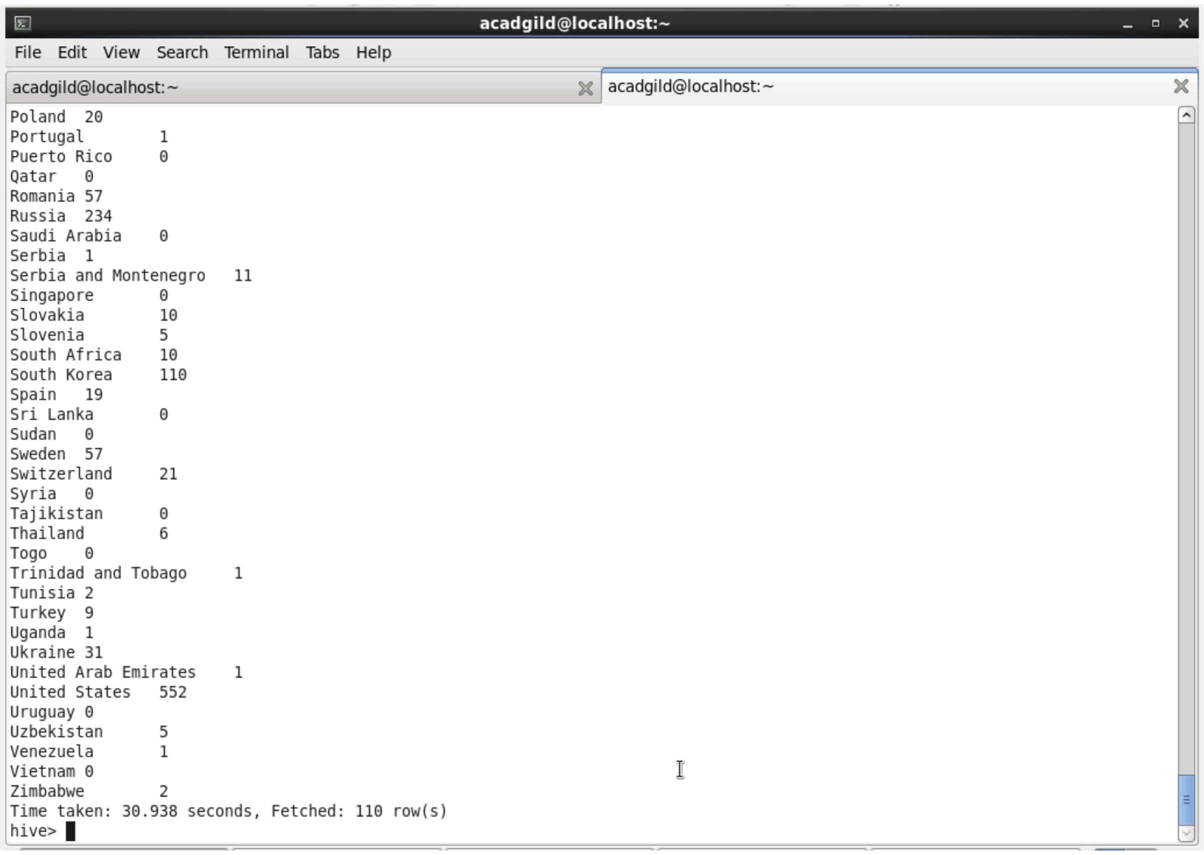
Total gold medals won by each country achieved from the query used above is as shown in the below screenshots.



```
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
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
```



```
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
Israel  1
Italy   86
Jamaica 24
Japan   57
Kazakhstan      13
Kenya   11
Kuwait  0
Kyrgyzstan      0
Latvia  3
Lithuania       5
Macedonia       0
Malaysia        0
Mauritius       0
Mexico  19
Moldova 0
Mongolia        2
Montenegro      0
Morocco 2
Mozambique      1
Netherlands     101
New Zealand     18
Nigeria 6
North Korea     6
```



```
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: 30.938 seconds, Fetched: 110 row(s)
hive>
```

Task -2:

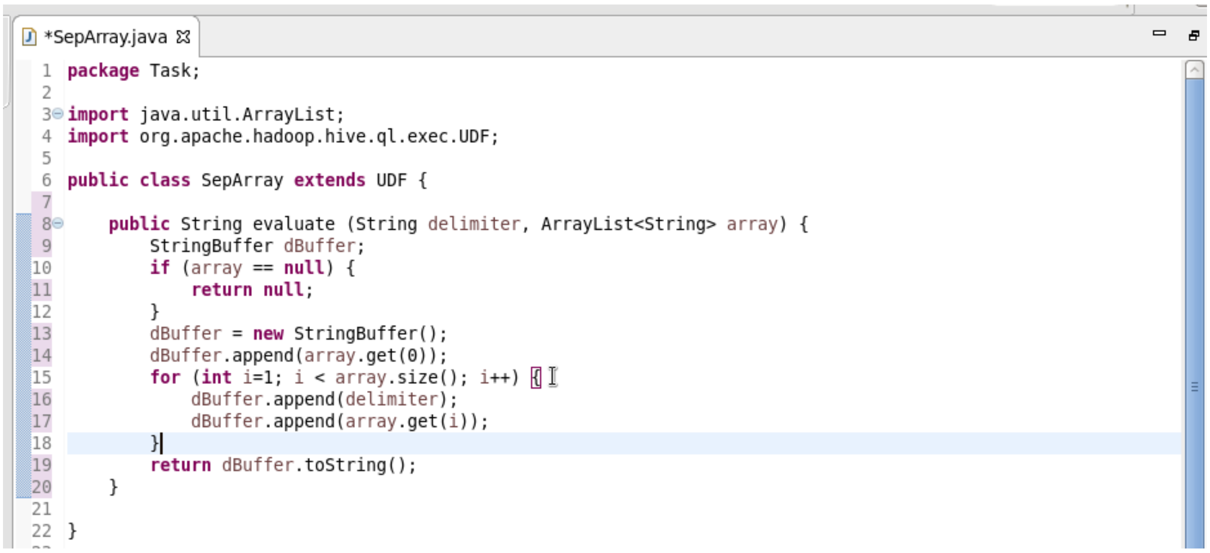
Write a hive UDF that implements functionality of string concat_ws(string SEP, array<string>).

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.

Created an UDF as shown in the below screenshot. Where the UDF will take the delimiter given and will delimit the data present in array. I have used a StringBuffer stored all the delimited data in that.

This UDF is exported into a jar file with name “**SepArray.jar**”

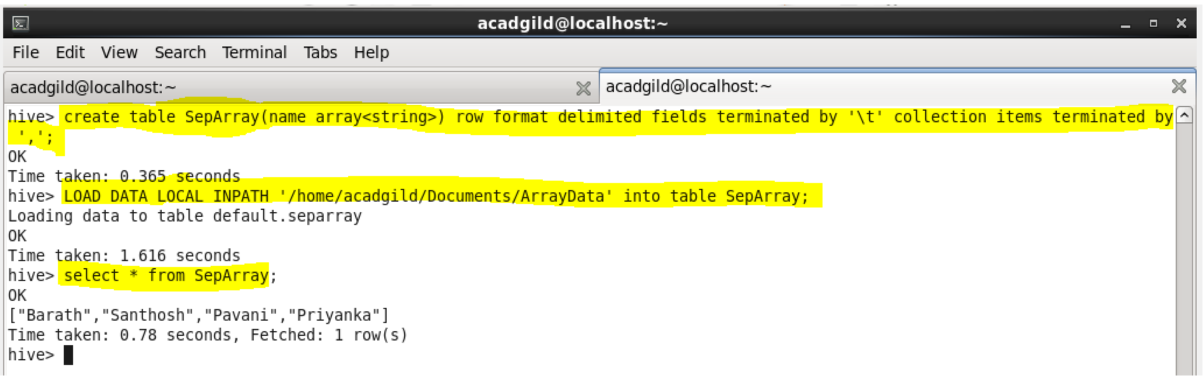


```java
package Task;

import java.util.ArrayList;
import org.apache.hadoop.hive.ql.exec.UDF;

public class SepArray extends UDF {

    public String evaluate (String delimiter, ArrayList<String> array) {
        StringBuffer dBuffer;
        if (array == null) {
            return null;
        }
        dBuffer = new StringBuffer();
        dBuffer.append(array.get(0));
        for (int i=1; i < array.size(); i++) {
            dBuffer.append(delimiter);
            dBuffer.append(array.get(i));
        }
        return dBuffer.toString();
    }

}
```

Created table with name “**SepArray**” and with column “**name**”.

Loaded the data into the table “**SepArray**”.

Data present in SepArray is shown in the below screeshot.



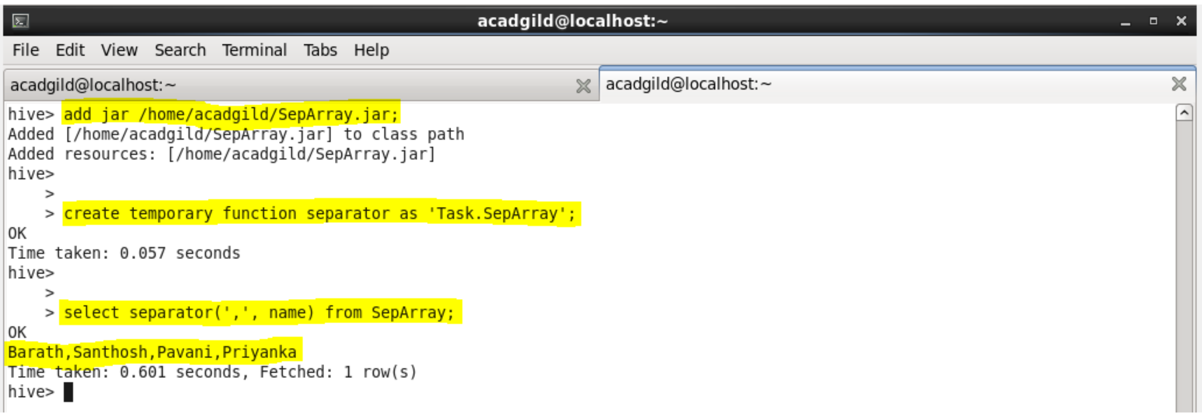
```
hive> create table SepArray(name array<string>) row format delimited fields terminated by '\t' collection items terminated by
 ',';
OK
Time taken: 0.365 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Documents/ArrayData' into table SepArray;
Loading data to table default.separray
OK
Time taken: 1.616 seconds
hive> select * from SepArray;
OK
["Barath","Santhosh","Pavani","Priyanka"]
Time taken: 0.78 seconds, Fetched: 1 row(s)
hive> 
```

Added the jar in hive shell.

Created temporary function “**separator**”.

Using the select query in the below query, data present in the table with column datatype array is delimited based on comma(**‘,’**).

Result after using **UDF** is shown in the below screenshot.

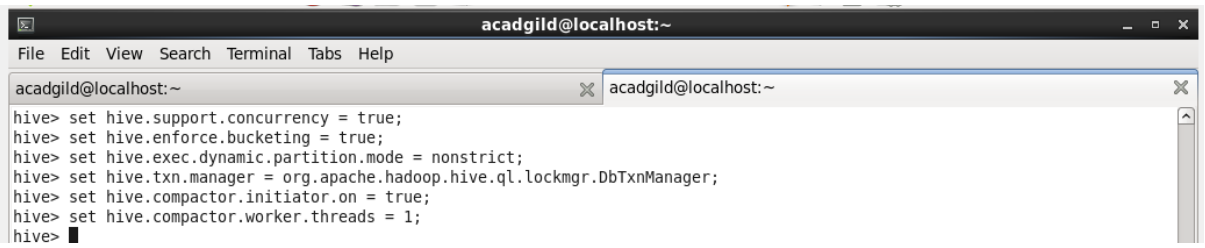


```
hive> add jar /home/acadgild/SepArray.jar;
Added [/home/acadgild/SepArray.jar] to class path
Added resources: [/home/acadgild/SepArray.jar]
hive>
    >
    > create temporary function separator as 'Task.SepArray';
OK
Time taken: 0.057 seconds
hive>
    >
    > select separator(',', name) from SepArray;
OK
Barath,Santhosh,Pavani,Priyanka
Time taken: 0.601 seconds, Fetched: 1 row(s)
hive>
```

Task -3:

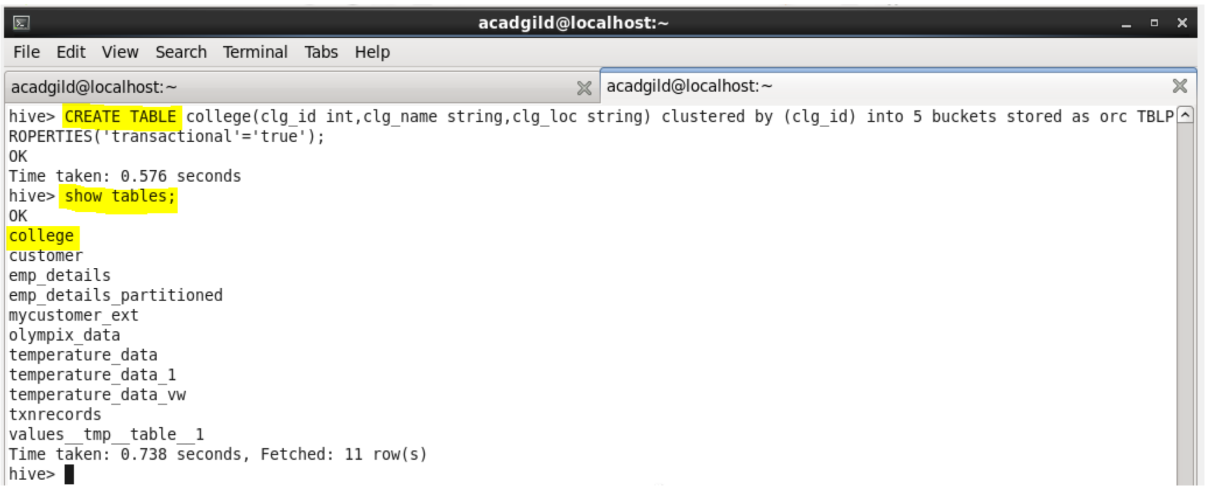
Implementation of transactions in hive.

The below properties needs to be set appropriately in hive shell, order-wise to work with transactions in Hive



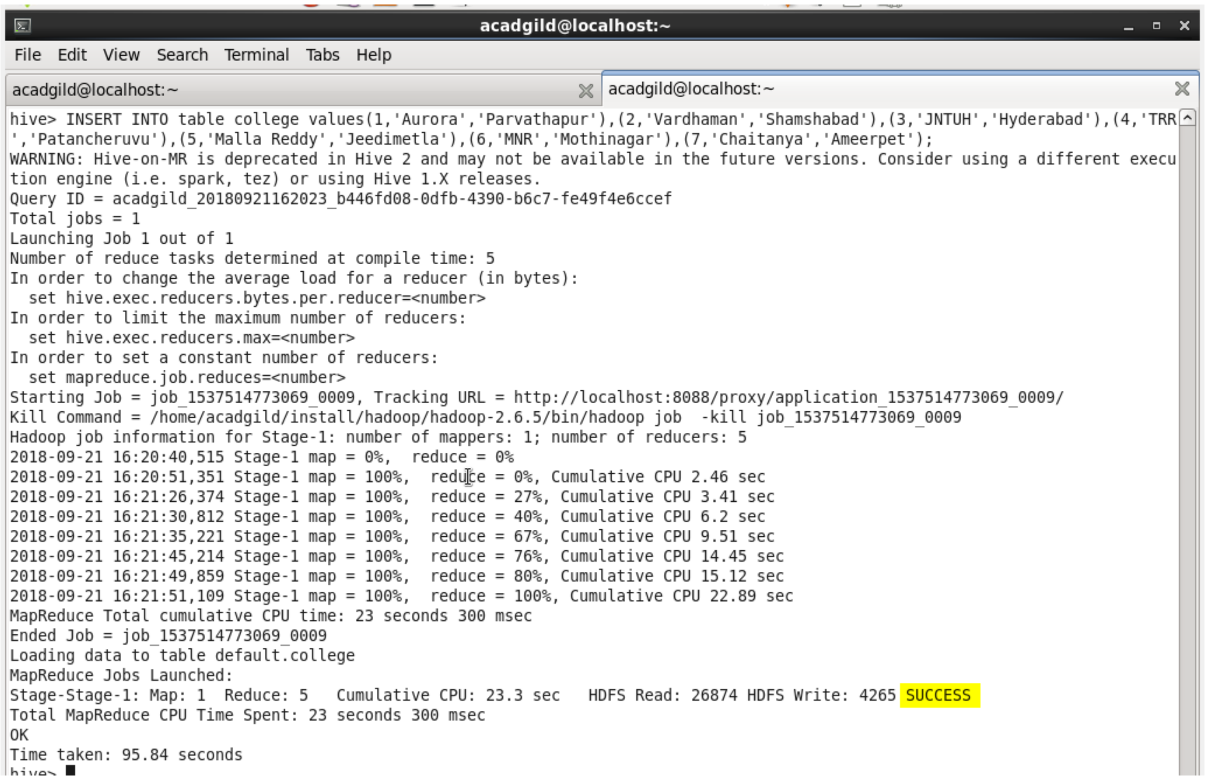
```
hive> set hive.support.concurrency = true;
hive> set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on = true;
hive> set hive.compactor.worker.threads = 1;
hive> 
```

Created table with name "college" and columns present are '**clg_id**', '**clg_name**' and '**clg_loc**' and the table bucketed with '**clg_id**' and table format is '**orc**'



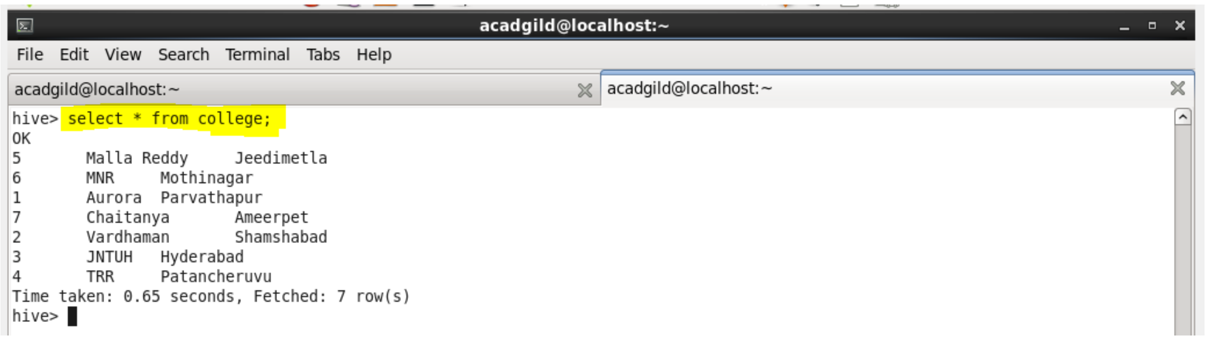
```
hive> CREATE TABLE college(clg_id int,clg_name string,clg_loc string) clustered by (clg_id) into 5 buckets stored as orc TBLP
ROPERTIES('transactional'='true');
OK
Time taken: 0.576 seconds
hive> show tables;
OK
college
customer
emp_details
emp_details_partitioned
mycustomer_ext
olympix_data
temperature_data
temperature_data_1
temperature_data_vw
txnrecords
values__tmp__table__1
Time taken: 0.738 seconds, Fetched: 11 row(s)
hive> 
```

Successfully inserted data into the table **'College'**.



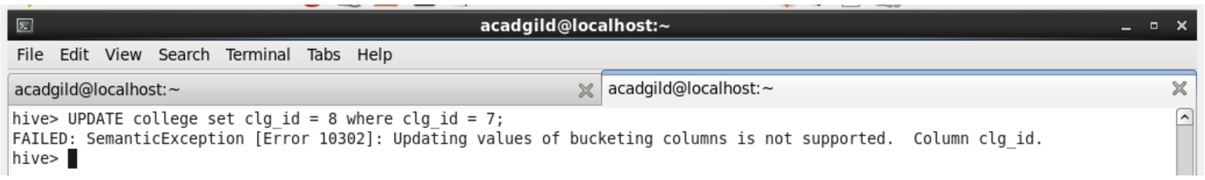
```
hive> INSERT INTO table college values(1,'Aurora','Parvathapur'),(2,'Vardhaman','Shamshabad'),(3,'JNTUH','Hyderabad'),(4,'TRR
','Patancheruvu'),(5,'Malla Reddy','Jeedimetla'),(6,'MNR','Mothinagar'),(7,'Chaitanya','Ameerpet');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180921162023_b446fd08-0dfb-4390-b6c7-fe49f4e6ccef
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_1537514773069_0009, Tracking URL = http://localhost:8088/proxy/application_1537514773069_0009/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537514773069_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-09-21 16:20:40,515 Stage-1 map = 0%,  reduce = 0%
2018-09-21 16:20:51,351 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.46 sec
2018-09-21 16:21:26,374 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 3.41 sec
2018-09-21 16:21:30,812 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 6.2 sec
2018-09-21 16:21:35,221 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 9.51 sec
2018-09-21 16:21:45,214 Stage-1 map = 100%,  reduce = 76%, Cumulative CPU 14.45 sec
2018-09-21 16:21:49,859 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 15.12 sec
2018-09-21 16:21:51,109 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 22.89 sec
MapReduce Total cumulative CPU time: 23 seconds 300 msec
Ended Job = job_1537514773069_0009
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 23.3 sec   HDFS Read: 26874 HDFS Write: 4265 SUCCESS
Total MapReduce CPU Time Spent: 23 seconds 300 msec
OK
Time taken: 95.84 seconds
hive> 
```

Data inserted into table is as shown below.



```
hive> select * from college;
OK
5       Malla Reddy     Jeedimetla
6       MNR     Mothinagar
1       Aurora  Parvathapur
7       Chaitanya       Ameerpet
2       Vardhaman       Shamshabad
3       JNTUH   Hyderabad
4       TRR     Patancheruvu
Time taken: 0.65 seconds, Fetched: 7 row(s)
hive> 
```

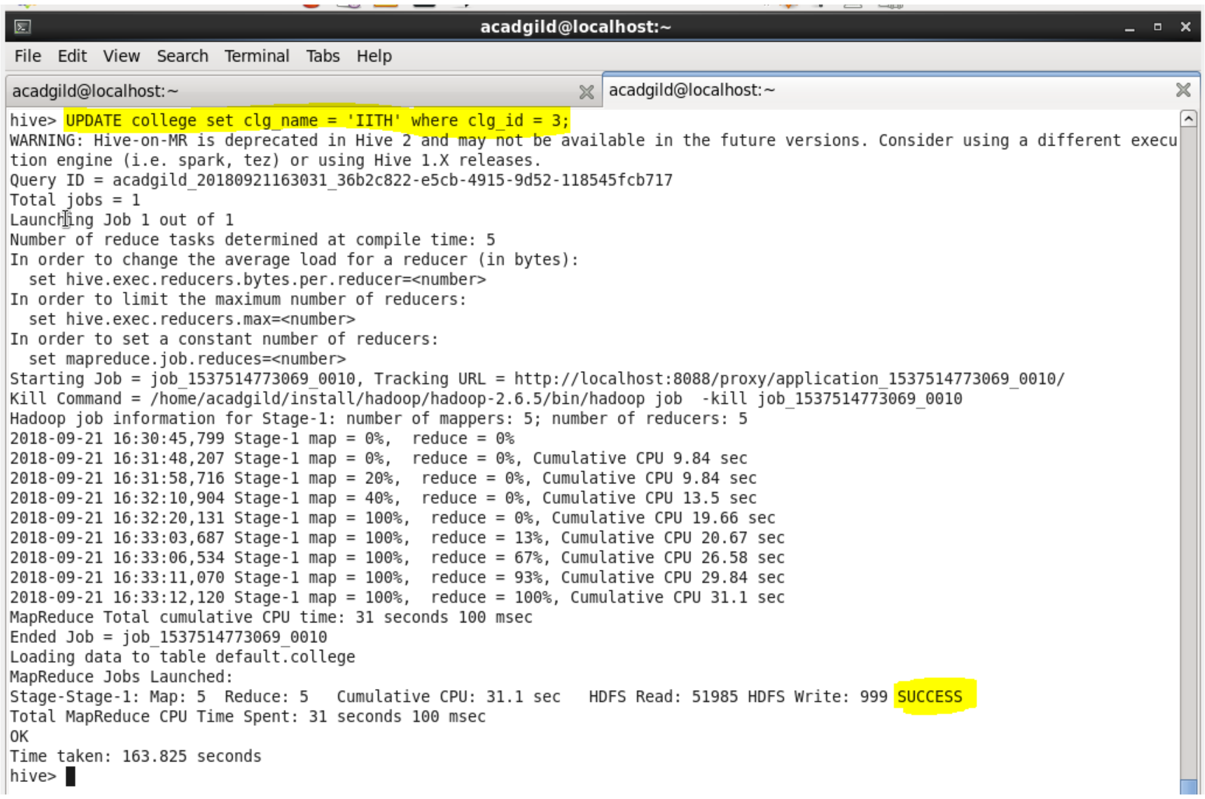
The below command is used to update a row in Hive table.



```
hive> UPDATE college set clg_id = 8 where clg_id = 7;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported.  Column clg_id.
hive> 
```

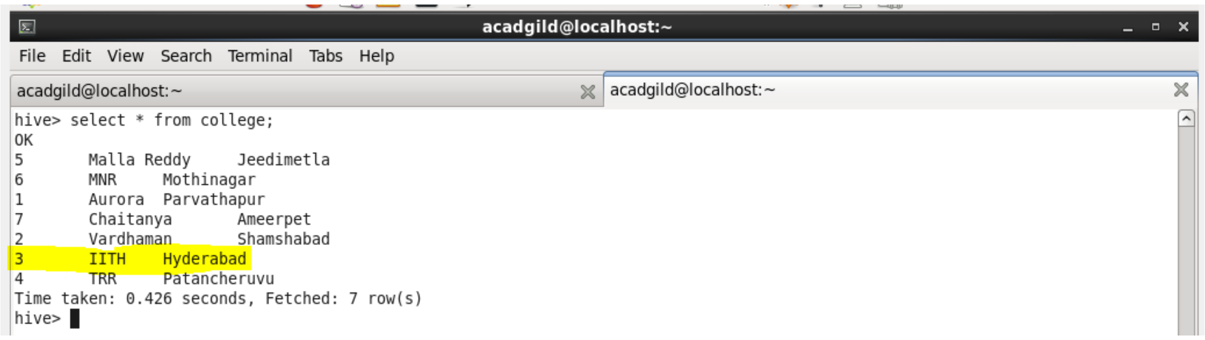
From the above image, we can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed.

Now the update operation is performed on non bucketed column.



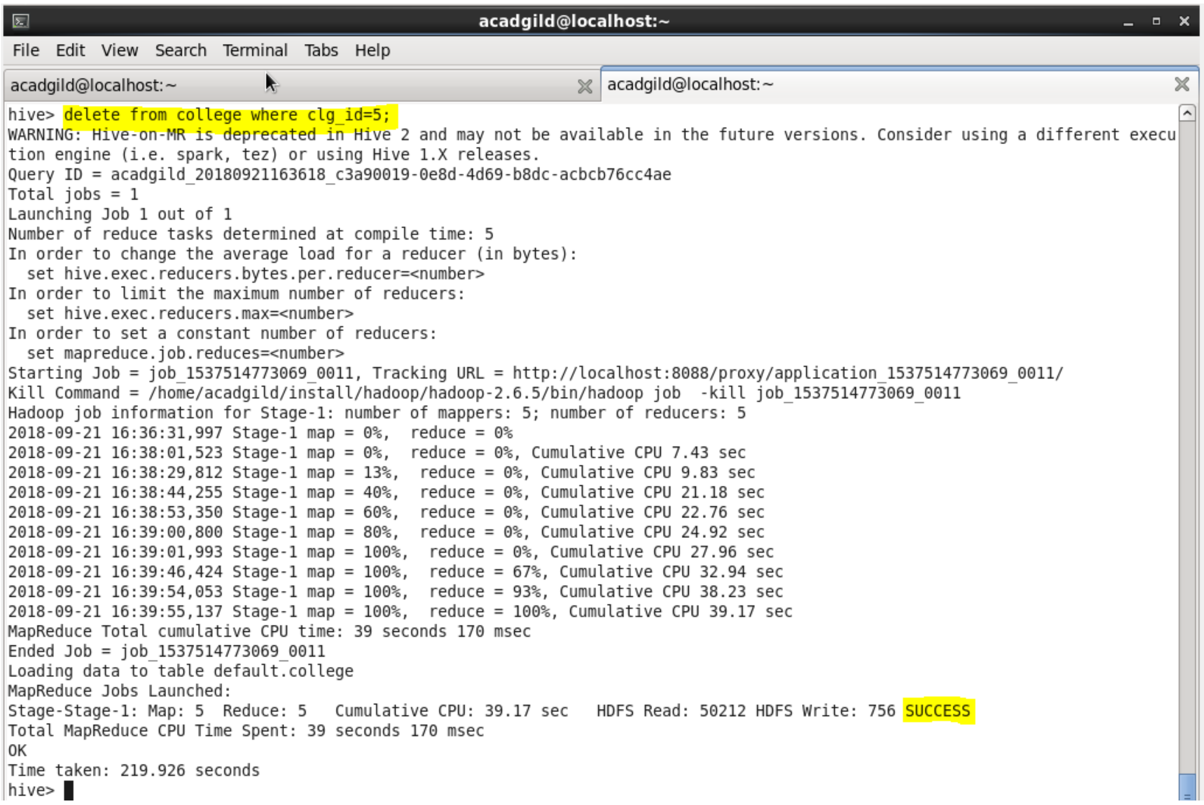
```
hive> UPDATE college set clg_name = 'IITH' where clg_id = 3;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180921163031_36b2c822-e5cb-4915-9d52-118545fcb717
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_1537514773069_0010, Tracking URL = http://localhost:8088/proxy/application_1537514773069_0010/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537514773069_0010
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-09-21 16:30:45,799 Stage-1 map = 0%,  reduce = 0%
2018-09-21 16:31:48,207 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 9.84 sec
2018-09-21 16:31:58,716 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 9.84 sec
2018-09-21 16:32:10,904 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 13.5 sec
2018-09-21 16:32:20,131 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 19.66 sec
2018-09-21 16:33:03,687 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 20.67 sec
2018-09-21 16:33:06,534 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 26.58 sec
2018-09-21 16:33:11,070 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 29.84 sec
2018-09-21 16:33:12,120 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 31.1 sec
MapReduce Total cumulative CPU time: 31 seconds 100 msec
Ended Job = job_1537514773069_0010
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 31.1 sec   HDFS Read: 51985 HDFS Write: 999 SUCCESS
Total MapReduce CPU Time Spent: 31 seconds 100 msec
OK
Time taken: 163.825 seconds
hive> █
```

Updated data in the table is as shown. Earlier it was **‘JNTUH’** and it was updated to **‘IITH’**.



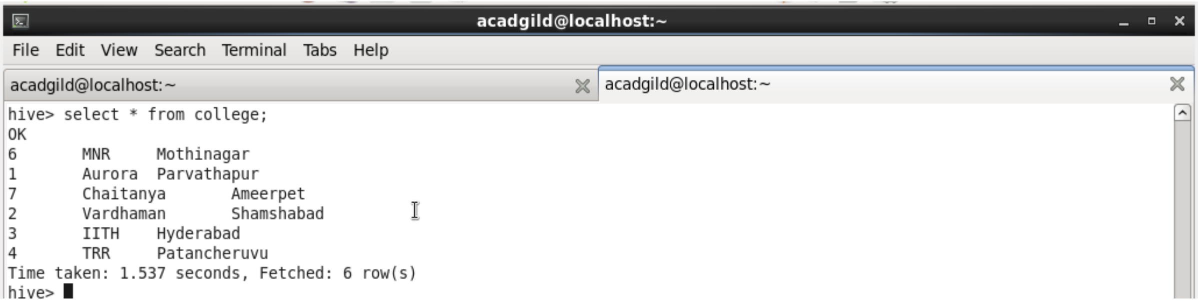
```
hive> select * from college;
OK
5       Malla Reddy     Jeedimetla
6       MNR     Mothinagar
1       Aurora  Parvathapur
7       Chaitanya       Ameerpet
2       Vardhaman       Shamshabad
3       IITH    Hyderabad
4       TRR     Patancheruvu
Time taken: 0.426 seconds, Fetched: 7 row(s)
hive> █
```

Now the delete operation is performed on the same table.



```
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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180921163618_c3a90019-0e8d-4d69-b8dc-acbcb76cc4ae
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_1537514773069_0011, Tracking URL = http://localhost:8088/proxy/application_1537514773069_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537514773069_0011
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-09-21 16:36:31,997 Stage-1 map = 0%,  reduce = 0%
2018-09-21 16:38:01,523 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 7.43 sec
2018-09-21 16:38:29,812 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 9.83 sec
2018-09-21 16:38:44,255 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 21.18 sec
2018-09-21 16:38:53,350 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 22.76 sec
2018-09-21 16:39:00,800 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 24.92 sec
2018-09-21 16:39:01,993 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 27.96 sec
2018-09-21 16:39:46,424 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 32.94 sec
2018-09-21 16:39:54,053 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 38.23 sec
2018-09-21 16:39:55,137 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 39.17 sec
MapReduce Total cumulative CPU time: 39 seconds 170 msec
Ended Job = job_1537514773069_0011
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 39.17 sec   HDFS Read: 50212 HDFS Write: 756 SUCCESS
Total MapReduce CPU Time Spent: 39 seconds 170 msec
OK
Time taken: 219.926 seconds
hive>
```

'Clg_id =5' has been deleted from the table as shown in the below screenshot.



```
hive> select * from college;
OK
6       MNR     Mothinagar
1       Aurora  Parvathapur
7       Chaitanya       Ameerpet
2       Vardhaman       Shamshabad
3       IITH    Hyderabad
4       TRR     Patancheruvu
Time taken: 1.537 seconds, Fetched: 6 row(s)
hive>
```