

BIG DATA HADOOP AND SPARK DEVELOPMENT

ASSIGNMENT 9

Table of Contents:

1. Introduction	2
2. Objective	2
3. Associated Data Files	2
4. Problem Statement	3
5. Expected Output	
• Task 1	5
• Task 2	13
• Task 3	17

BIG DATA HADOOP AND SPARK DEVELOPMENT

1. Introduction

In this assignment, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

2. Objective

This Assignment consolidates the deeper understanding of the Session – 9 Advanced concepts of HIVE

3. Associated Data Files

This Data set is about Olympics. You can download the data set from the below link:

<https://drive.google.com/open?id=0ByJLBtmJojjzV1czX3Nha0R3bTQ>

DATA SET DESCRIPTION

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 field 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

4. Problem Statement

- Task 1

1. Write a Hive program to find the number of medals won by each country in swimming.
2. Write a Hive program to find the number of medals that India won year wise.
3. Write a Hive Program to find the total number of medals each country won.
4. Write a Hive program to find the number of gold medals each country won.

- 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.

- Task 3

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

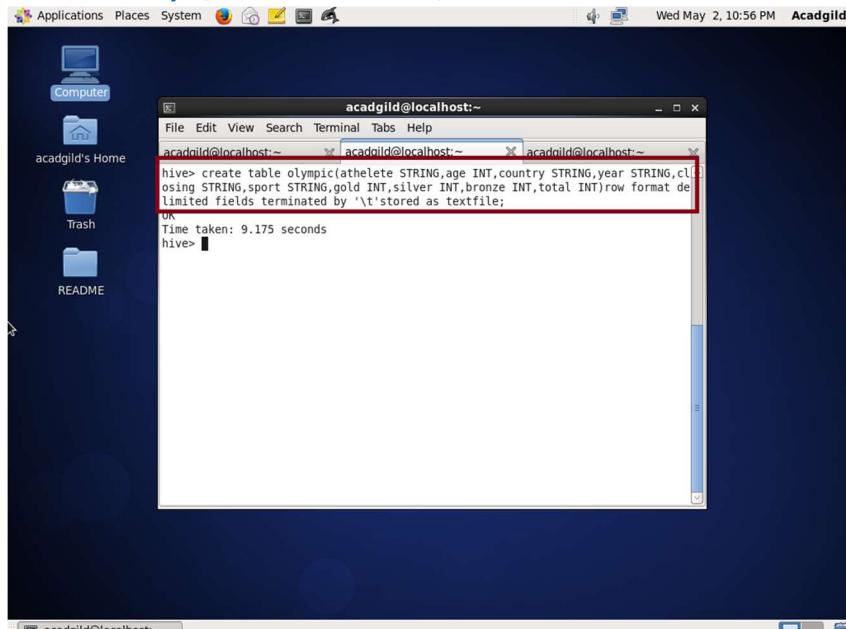
Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

5. Expected Output

For performing the task 1, the Olympic table structure is created and the data is also loaded to the Olympic table.

Table Olympic is created by the following command

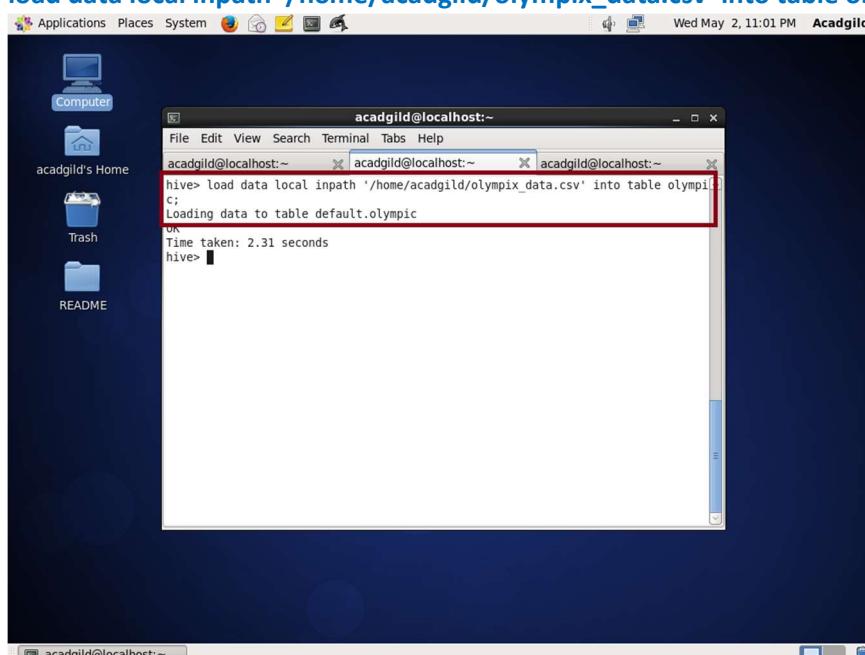
```
create table olympic(athlete STRING, age INT, country STRING, year STRING, closing STRING,  
sport STRING, gold INT, silver INT, bronze INT, total INT)row format delimited  
terminated by '\t' stored as textfile;
```



A screenshot of a Linux desktop environment. In the center is a terminal window titled "acadgild@localhost:~". The window contains the following text:

```
acadgild@localhost:~$ hive> create table olympic(athlete STRING, age INT, country STRING, year STRING, closing STRING,sport STRING,gold INT,silver INT,bronze INT,total INT)row format de  
limited fields terminated by '\t' stored as textfile;  
OK  
Time taken: 9.175 seconds  
hive>
```

```
load data local inpath '/home/acadgild/olympix_data.csv' into table olympic;
```



A screenshot of a Linux desktop environment. In the center is a terminal window titled "acadgild@localhost:~". The window contains the following text:

```
acadgild@localhost:~$ hive> load data local inpath '/home/acadgild/olympix_data.csv' into table olympic;  
C:  
Loading data to table default.olympic  
OK  
Time taken: 2.31 seconds  
hive>
```

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

GROUP BY clause in a SELECT statement:

The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

Syntax

The syntax of GROUP BY clause is as follows:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]]
[LIMIT number];
```

Hive supports the following built-in **aggregate functions**.

count(*), count(expr)

count(*) - Returns the total number of retrieved rows.

sum(col), sum(DISTINCT col)

It returns the sum of the elements in the group or the sum of the distinct values of the column in the group.

avg(col), avg(DISTINCT col)

It returns the average of the elements in the group or the average of the distinct values of the column in the group.

min(col)

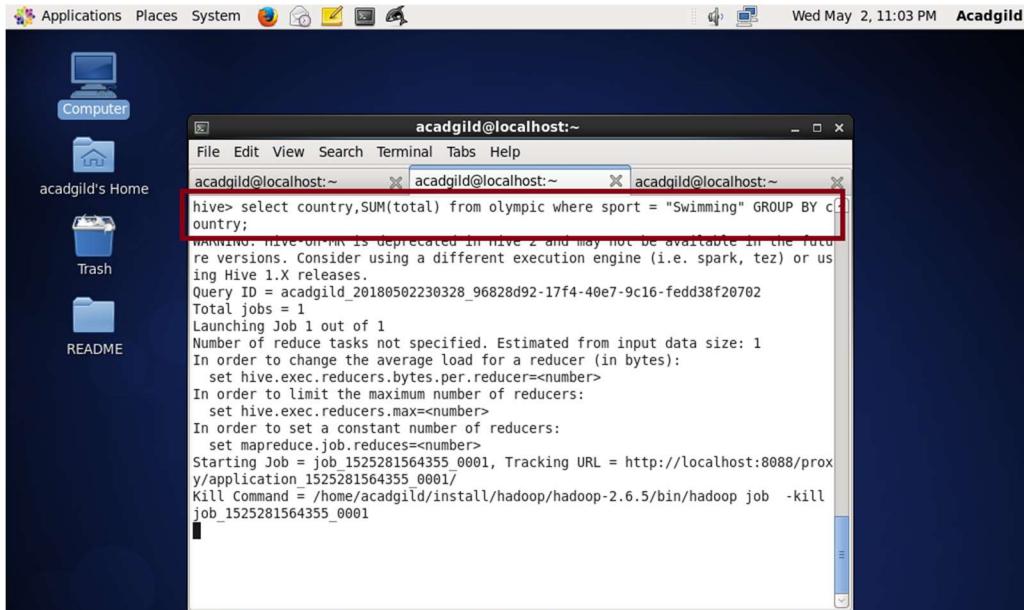
It returns the minimum value of the column in the group.

max(col)

It returns the maximum value of the column in the group.

By using the following command, the number of medals won by each country in swimming.

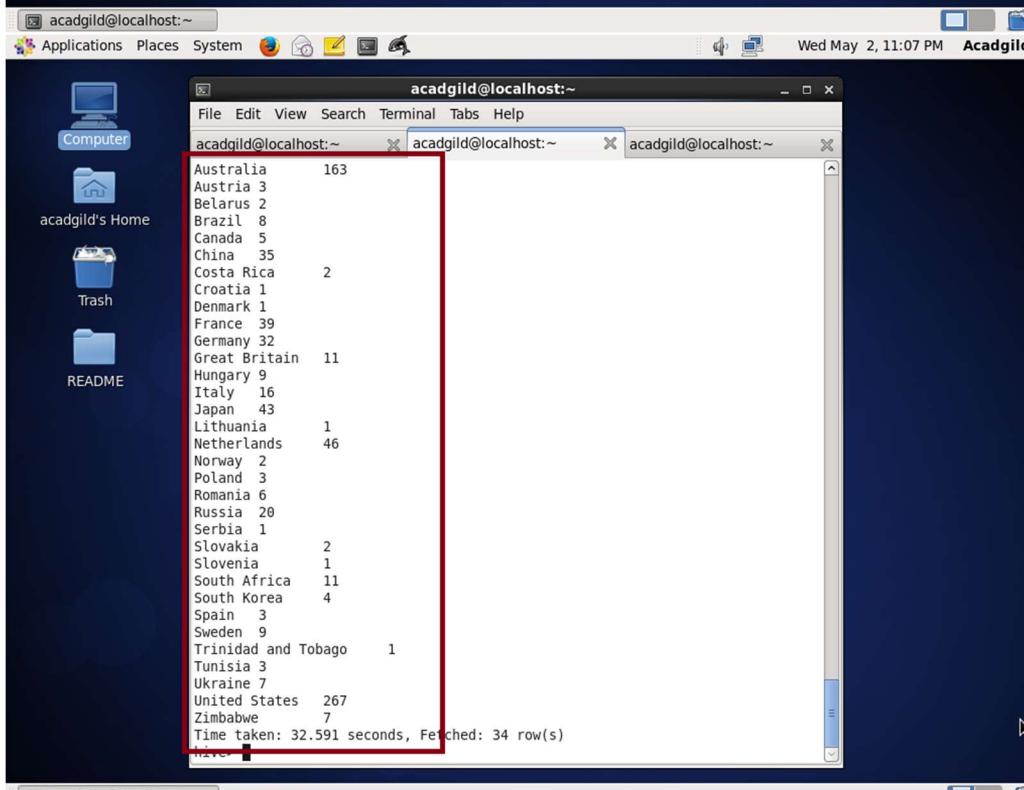
select country, SUM(total) from Olympic where sport = "Swimming" GROUP BY country;



```
acadgild@localhost:~
```

```
hive> select country,SUM(total) from olympic 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 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180502230328_96828d92-17f4-40e7-9c16-fedd38f20702
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_1525281564355_0001, Tracking URL = http://localhost:8088/proxy/application_1525281564355_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525281564355_0001



```
acadgild@localhost:~
```

country	total
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: 32.591 seconds, Fetched: 34 row(s)
```

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

By select query the year and sum of total from Olympic table where country is India.

By using the following command, the number of medals won by India are displayed with year.

select year, SUM(total) from olympic where country = "India" GROUP BY year;

```
hive> select year,SUM(total) from olympic where country = "India" GROUP BY year;
[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_20180502230903_fd3dca42-de89-4238-82dc-94dc88e3b8a7
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_1525281564355_0004, Tracking URL = http://localhost:8088/pr
oxy/application_1525281564355_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kil
l job_1525281564355_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers:
1
2018-05-02 23:09:14,813 Stage-1 map = 0%, reduce = 0%
2018-05-02 23:09:24,831 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.99
sec
2018-05-02 23:09:34,726 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.4
3 sec
MapReduce Total cumulative CPU time: 5 seconds 430 msec
Ended Job = job_1525281564355_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.43 sec HDFS Read: 52854
1 HDFS Write: 163 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 430 msec
OK
2000      1
2004      1
2008      3
2012      6
```

```
hive> select year,SUM(total) from olympic where country = "India" GROUP BY year;
[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_20180502230903_fd3dca42-de89-4238-82dc-94dc88e3b8a7
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_1525281564355_0004, Tracking URL = http://localhost:8088/pr
oxy/application_1525281564355_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kil
l job_1525281564355_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers:
1
2018-05-02 23:09:14,813 Stage-1 map = 0%, reduce = 0%
2018-05-02 23:09:24,831 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.99
sec
2018-05-02 23:09:34,726 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.4
3 sec
MapReduce Total cumulative CPU time: 5 seconds 430 msec
Ended Job = job_1525281564355_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.43 sec HDFS Read: 52854
1 HDFS Write: 163 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 430 msec
OK
2000      1
2004      1
2008      3
2012      6
time taken: 32.739 seconds, Fetched: 4 row(s)
hive>
```

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

By select query the country and sum of total from Olympic table.

By using the following command, the total number of medals won by each country are displayed.

select country, SUM(total) from olympic GROUP BY country;

```
acadgild@localhost:~$ hive> select country,SUM(total) from olympic GROUP BY country;
WARNING: Hive's MapReduce is deprecated in Hive 1.2 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_20180502233216_c26f37a6-0fc4-4c56-82a8-ce174f59177a
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_1525281564355_0007, Tracking URL = http://localhost:8088/proxy/application_1525281564355_0007/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525281564355_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-02 23:32:27,528 Stage-1 map = 0%,  reduce = 0%
2018-05-02 23:32:36,469 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.05 sec
2018-05-02 23:32:46,446 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.4 sec
MapReduce Total cumulative CPU time: 4 seconds 400 msec
Ended Job = job_1525281564355_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1  Cumulative CPU: 4.4 sec  HDFS Read: 527721
HDFS Write: 2742 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 400 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
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
```

Country	Total Medals
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

Applications Places System  Wed May 2, 11:15 PM Acadgild

Computer acadgild's Home Trash README

```
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
```

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
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

Applications Places System  Wed May 2, 11:16 PM Acadgild

Computer acadgild's Home Trash README

```
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
```

Nigeria	39
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

Applications Places System Wed May 2, 11:16 PM Acadgild

Computer
acadgild's Home
Trash
README

```
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~  
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: 33.316 seconds, Fetched: 110 row(s)
```

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

By select query the country and sum of gold medals from Olympic table.

By using the following command, the total number of gold medals won by each country are displayed.

Select country, SUM(gold) from Olympic GROUP BY country;

```
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
hive> select country,SUM(gold) from olympic 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 engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180502232032_f8c92b80-4818-4611-8b35-f22cb6f0fb46
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_1525281564355_0006, Tracking URL = http://localhost:8088/proxy/application_1525281564355_0006/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525281564355_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-05-02 23:20:43,675 Stage-1 map = 0%, reduce = 0%
2018-05-02 23:20:52,637 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.95 sec
2018-05-02 23:21:02,448 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.13 sec
MapReduce Total cumulative CPU time: 4 seconds 130 msec
Ended Job = job_1525281564355_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.13 sec HDFS Read: 52771
9 HDFS Write: 2703 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 130 msec
OK
Afghanistan      0
Algeria         2
Argentina        49
```

```
File Edit View Search Terminal Tabs Help
acadgild@localhost:~ acadgild@localhost:~ acadgild@localhost:~
Armenia 0
Australia 163
Austria 36
Azerbaijan 6
Bahamas 11
Bahrain 0
Barbados 0
Belarus 17
Belgium 2
Botswana 0
Brazil 46
Bulgaria 8
Cameroun 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
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
Norway 97
```

```
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
```

Time taken: 30.712 seconds. Fetched: 110 row(s)

- 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.

Java-based user-defined function (UDF)

Create a Java class for the User Defined Function which extends org.apache.hadoop.hive.sq.exec.UDF and implements more than one evaluate() methods.

concatws.java

```
package concatws;

import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDF;

// Description of the UDF
@Description (name="concatws",
    value="_FUNC_ (string SEP,array<string>)-RETURN_TYPE(STRING)\n")
public class concatws extends UDF {
    // Accepts two arguments, arg1 is string and arg2 is array of string
    public String evaluate (String arg1, String [] arg2) {
        String Output = "";
        // If the arg1 and arg2 are null then return null
        if (arg1== null && arg2== null)
        {
            return null;
        }
        // Length of arg2 is saved in length
        int length = arg2.length;
        for (int i=0; i<length; i++)
        {
            Output += arg2[i];
        }
        // Return the value of Concatenated arg1 string and arg2 array of string
        return(arg1.concat(Output));
    }
}
```

Jar file is created with the above code

Saved as concatws.jar

Using show databases, the databases are listed.

Show databases;

In that databases **custom** database is listed.

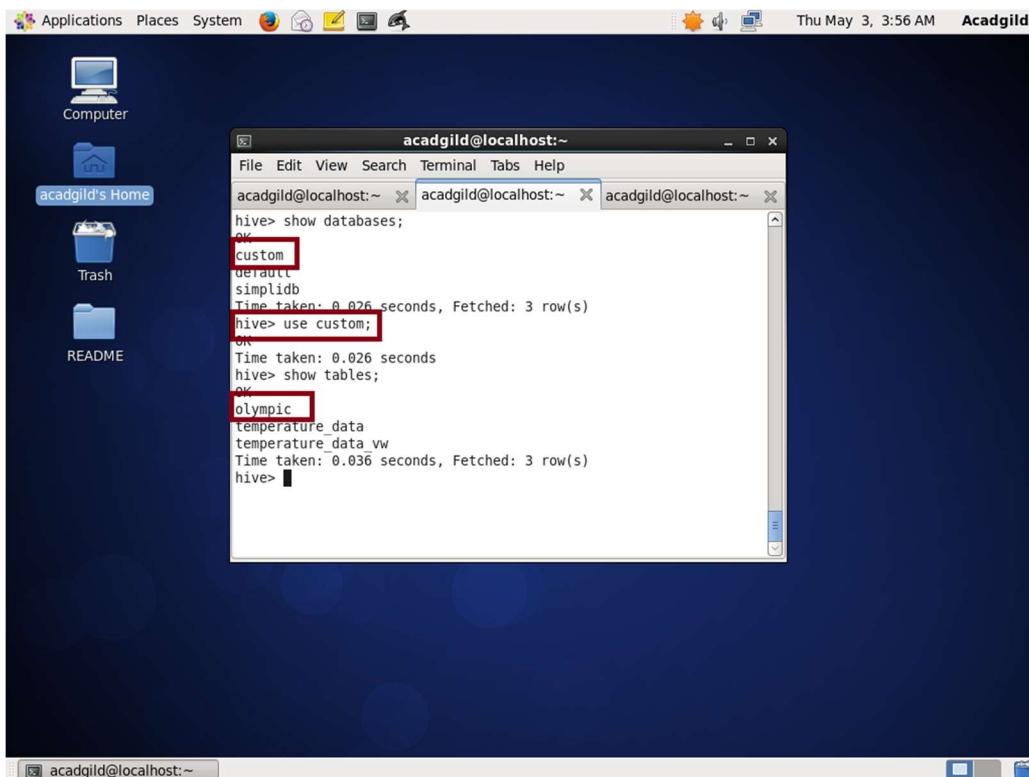
Using the command, use custom which is shown in the databases list.

Use custom;

Using show tables, the tables are listed.

Show tables;

In that databases, **Olympic** table is listed.



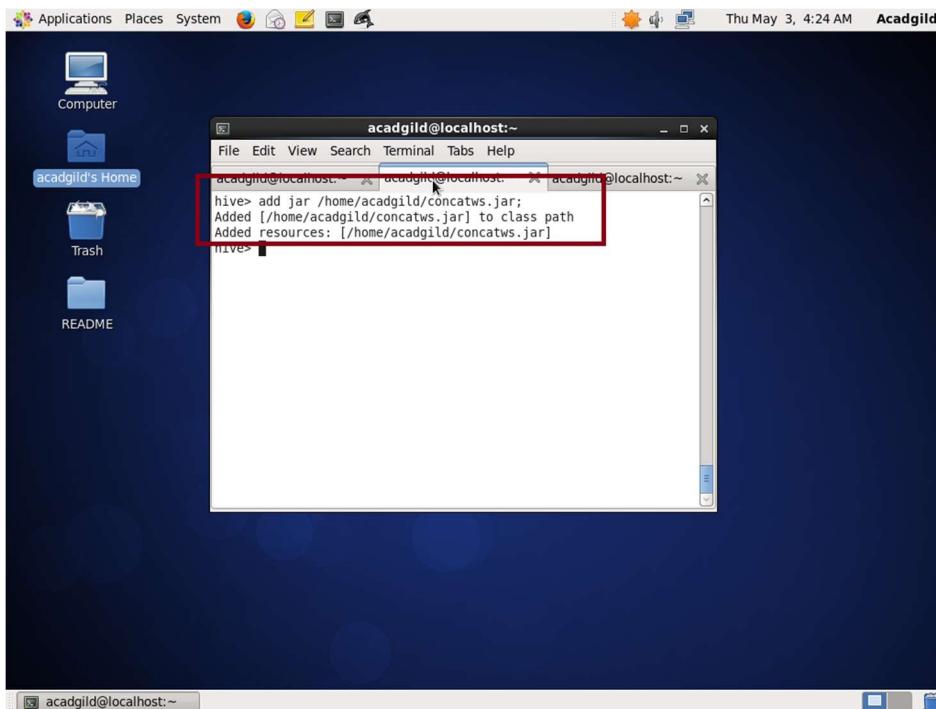
The screenshot shows a Linux desktop environment with a dark blue background. A terminal window titled "acadgild@localhost:~" is open, displaying the following Hive commands and output:

```
hive> show databases;
OK
custom
default
simplidb
Time taken: 0.026 seconds, Fetched: 3 row(s)
hive> use custom;
OK
Time taken: 0.026 seconds
hive> show tables;
OK
olympic
temperature_data
temperature_data_vw
Time taken: 0.036 seconds, Fetched: 3 row(s)
hive>
```

The words "custom" and "olympic" are highlighted with red boxes. The terminal window has three tabs, all showing the same command-line interface. The desktop environment includes a panel at the top with icons for Applications, Places, System, and a clock showing Thu May 3, 3:56 AM. The desktop also features icons for Computer, acadgild's Home, Trash, and README.

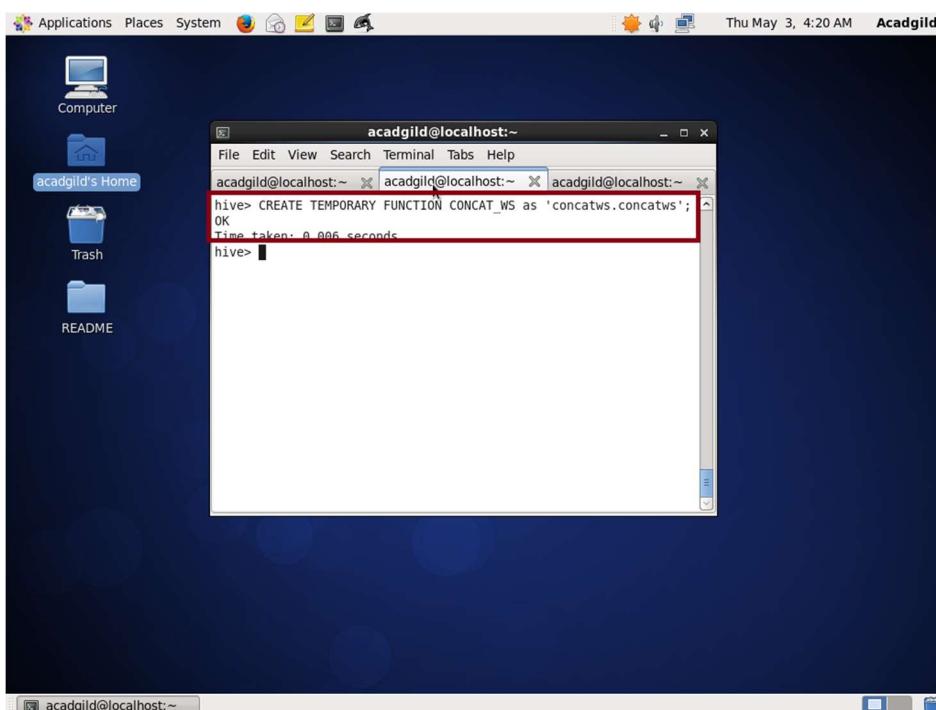
```
add jar /home/acadgild/hadoop/concatws.jar;
```

The concatws.jar which is created as UDF is added to the path file by using the above command and the screenshot is shown below.



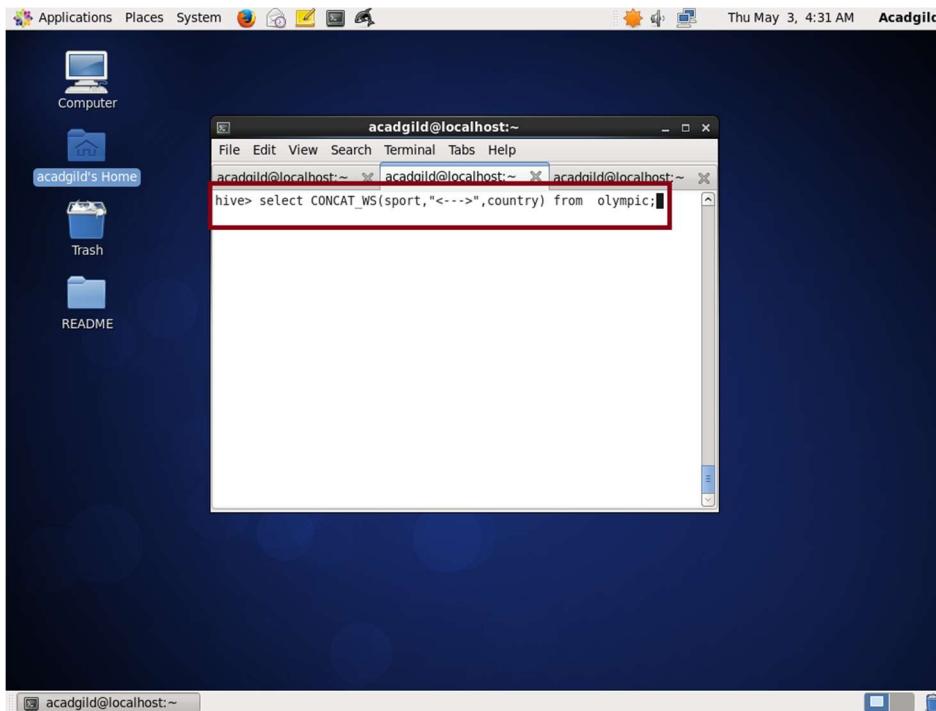
Temporary function is created as following,

```
CREATE TEMPORARY FUNCTION CONCAT_WS AS 'concatws.concatws';
```

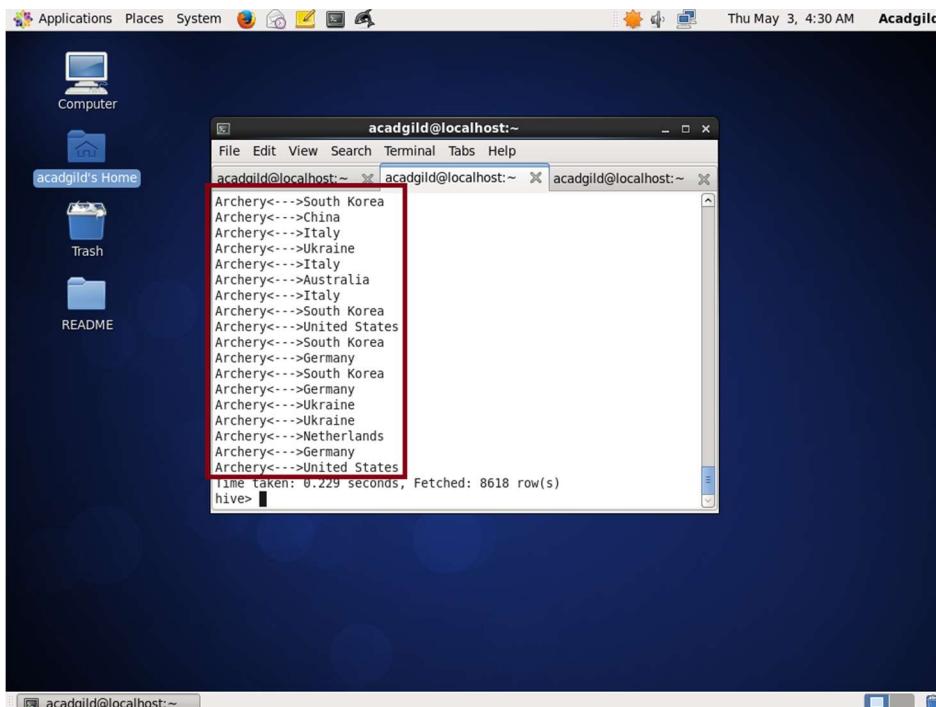


The temporary function is created and used as follows in the Olympic table.

```
select CONCAT_WS(sport,'↔',country) from Olympic;
```



Select one column (sport) input as String and another array(country) as Array of Strings is concatenated by CONCAT_WS(string SEP, array<string>)



- 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.

Transactions in Hive

Transactions in Hive are introduced in **Hive 0.13**, but they only partially fulfill the ACID properties like atomicity, consistency, durability, at the partition level. Here, Isolation can be provided by turning on one of the locking mechanisms available with zookeeper or in memory.

But in **Hive 0.14**, new API's have been added to completely fulfill the ACID properties while performing any transaction.

Transactions are provided at the row-level in Hive 0.14. The different row-level transactions available in Hive 0.14 are as follows:

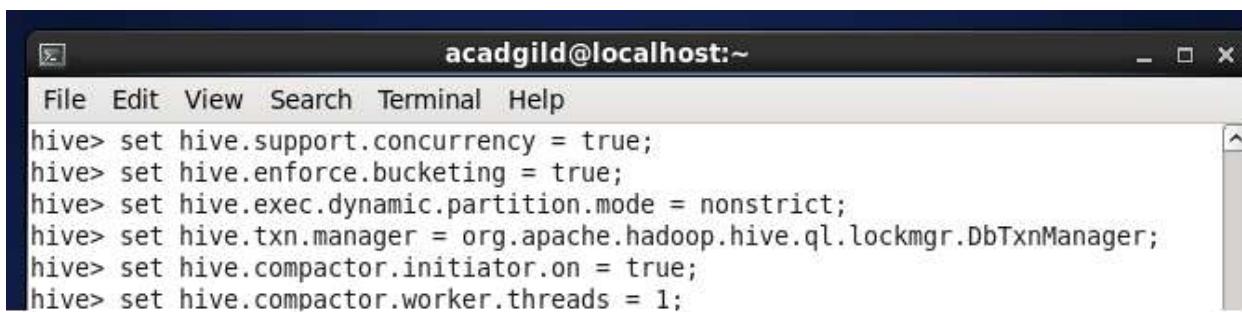
- Insert
- Delete
- Update

There are numerous limitations with the present transactions available in Hive 0.14. ORC is the file format supported by Hive transaction. It is now essential to have ORC file format for performing transactions in Hive. The table needs to be bucketed in order to support transactions.

Row-level Transactions Available in Hive 0.14

Let's perform some row-level transactions available in Hive 0.14. 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:



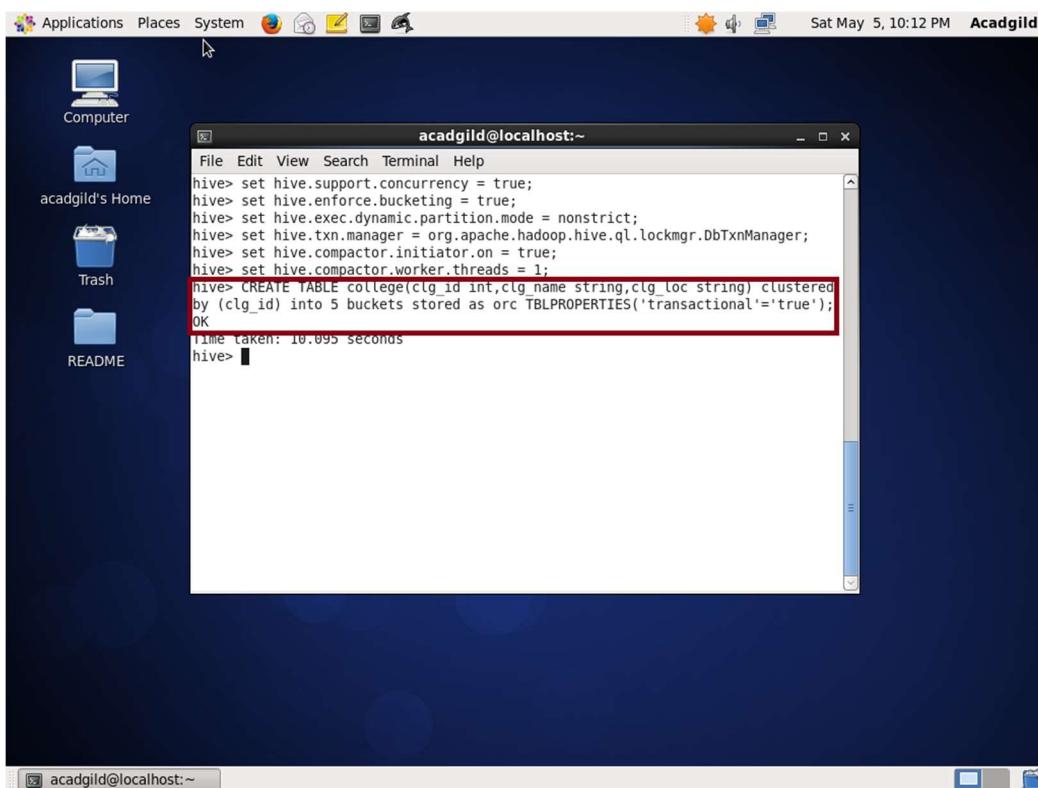
```
acadgild@localhost:~$ File Edit View Search Terminal Help
hive> set hive.support.concurrency = true;
hive> set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on = true;
hive> set hive.compactor.worker.threads = 1;
```

If the above properties are not set properly, the ‘Insert’ operation will work.

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*’. We 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’’



```
acadgild@localhost:~$ 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> 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: 10.095 seconds
hive>
```

We have successfully created a table with name ‘college’ which supports row-level transactions of Hive.

Inserting Data into a Hive Table

```
INSERT INTO table college
values(1,'nec','nlr'),(2,'vit','vlr'),(3,'srm','chen'),(4,'lpu','del'),(5,'stanford','uk'),(6,'JNTUA','atp'),(7,'cambridge','us');
```

The above command is used to insert row wise data into the Hive table. Here, each row is separated by ‘()’ brackets.

```
File Edit View Search Terminal Help
acadgild@localhost:~$ hive> INSERT INTO table college values(1,'nec','nlr'),(2,'vit','vlr'),(3,'srm','chen'),(4,'lpu','del'),(5,'standford','uk'),(6,'JNTUA','atp'),(7,'cambridge','us');
hive>
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_20180505222511_9ab75ac9-df18-4eed-b0eb-a384ca40586b
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_1525531281181_0001, Tracking URL = http://localhost:8088/proxy/application_1525531281181_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525531281181_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-05-05 22:25:46,189 Stage-1 map = 0%,  reduce = 0%
2018-05-05 22:26:01,772 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.34 sec
2018-05-05 22:26:40,941 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 5.12 sec
2018-05-05 22:26:42,422 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 6.13 sec
2018-05-05 22:26:43,792 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 8.51 sec
2018-05-05 22:27:01,513 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 13.15 sec
2018-05-05 22:27:02,989 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 22.13 sec
2018-05-05 22:27:04,289 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 25.58 sec
MapReduce Total cumulative CPU time: 25 seconds 580 msec
Ended Job = job_1525531281181_0001
Loading data to table default.college
```

```
File Edit View Search Terminal Help
acadgild@localhost:~$ hive>
ec
2018-05-05 22:26:42,422 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU
ec
2018-05-05 22:26:43,792 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU
ec
2018-05-05 22:27:01,513 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU
sec
2018-05-05 22:27:02,989 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU
sec
2018-05-05 22:27:04,289 Stage-1 map = 100%,  reduce = 100%, Cumulative CP
sec
MapReduce Total cumulative CPU time: 25 seconds 580 msec
Ended Job = job_1525531281181_0001
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 5 Cumulative CPU: 25.58 sec HDFS Read:
HDFS Write: 4006 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 580 msec
OK
Time taken: 115.39 seconds
hive>
```

Now, we have successfully inserted the data into the Hive table.
The contents of the table can be viewed using the command

```
select * from college
```

```
acadgild@localhost:~$ hive> select * from college;
OK
5      standford      uk
6      JNTUA      atp
1      nec      nlr
7      cambridge      us
2      vit      vlr
3      srm      chen
4      lpu      del
Time taken: 0.384 seconds, Fetched: 7 row(s)
hive>
```

From the above image, we can see that the data has been inserted successfully into the table.

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

```
acadgild@localhost:~$ hive> INSERT INTO table college values(1,'nec','nlr'),(2,'vit','vlr'),(3,'srm','chen'),(4,'lpu','del'),(5,'standford','uk'),(6,'JNTUA','atp'),(7,'cambridge','us');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180505223344_0c4a5949-d269-4402-a8de-68e02a7888e8
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_1525531281181_0002, Tracking URL = http://localhost:8088/proxy/application_1525531281181_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525531281181_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-05-05 22:33:59,007 Stage-1 map = 0%,  reduce = 0%
2018-05-05 22:34:09,563 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.92 sec
2018-05-05 22:34:41,765 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 4.43 sec
2018-05-05 22:34:43,197 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 5.58 sec
2018-05-05 22:34:45,926 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 6.65 sec
2018-05-05 22:34:48,704 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 8.23 sec
2018-05-05 22:34:50,104 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU
```

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "acadgild@localhost:~". The terminal displays the following Hive query execution:

```
File Edit View Search Terminal Help
13.37 sec
2018-05-05 22:35:00,839 Stage-1 map = 100%, reduce = 80%, Cumulative CPU
15.96 sec
2018-05-05 22:35:03,406 Stage-1 map = 100%, reduce = 93%, Cumulative CPU
21.15 sec
2018-05-05 22:35:04,483 Stage-1 map = 100%, reduce = 100%, Cumulative CPU
U 23.8 sec
MapReduce Total cumulative CPU time: 23 seconds 800 msec
Ended Job = job_1525531281181_0002
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 5 Cumulative CPU: 23.8 sec HDFS Read:
26598 HDFS Write: 4005 SUCCESS
Total MapReduce CPU Time Spent: 23 seconds 800 msec
OK
Time taken: 82.049 seconds
hive> select * from college;
+-----+-----+-----+
| clg_id | name | country |
+-----+-----+-----+
| 5      | stanford | uk      |
| 5      | stanford | uk      |
| 6      | JNTUA    | atp     |
| 1      | nec      | nlr     |
| 6      | JNTUA    | atp     |
| 1      | nec      | nlr     |
| 7      | cambridge | us      |
| 2      | vit      | vlr     |
| 7      | cambridge | us      |
| 2      | vit      | vlr     |
| 3      | srm      | chen   |
| 3      | srm      | chen   |
| 4      | lpu      | del     |
| 4      | lpu      | del     |
+-----+-----+-----+
Time taken: 0.318 seconds, Fetched: 14 row(s)
hive>
```

Updating the Data in Hive Table

UPDATE college set clg_id = 8 where clg_id = 7;

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

The screenshot shows a terminal window with the title "acadgild@localhost:~". The terminal displays the following error message:

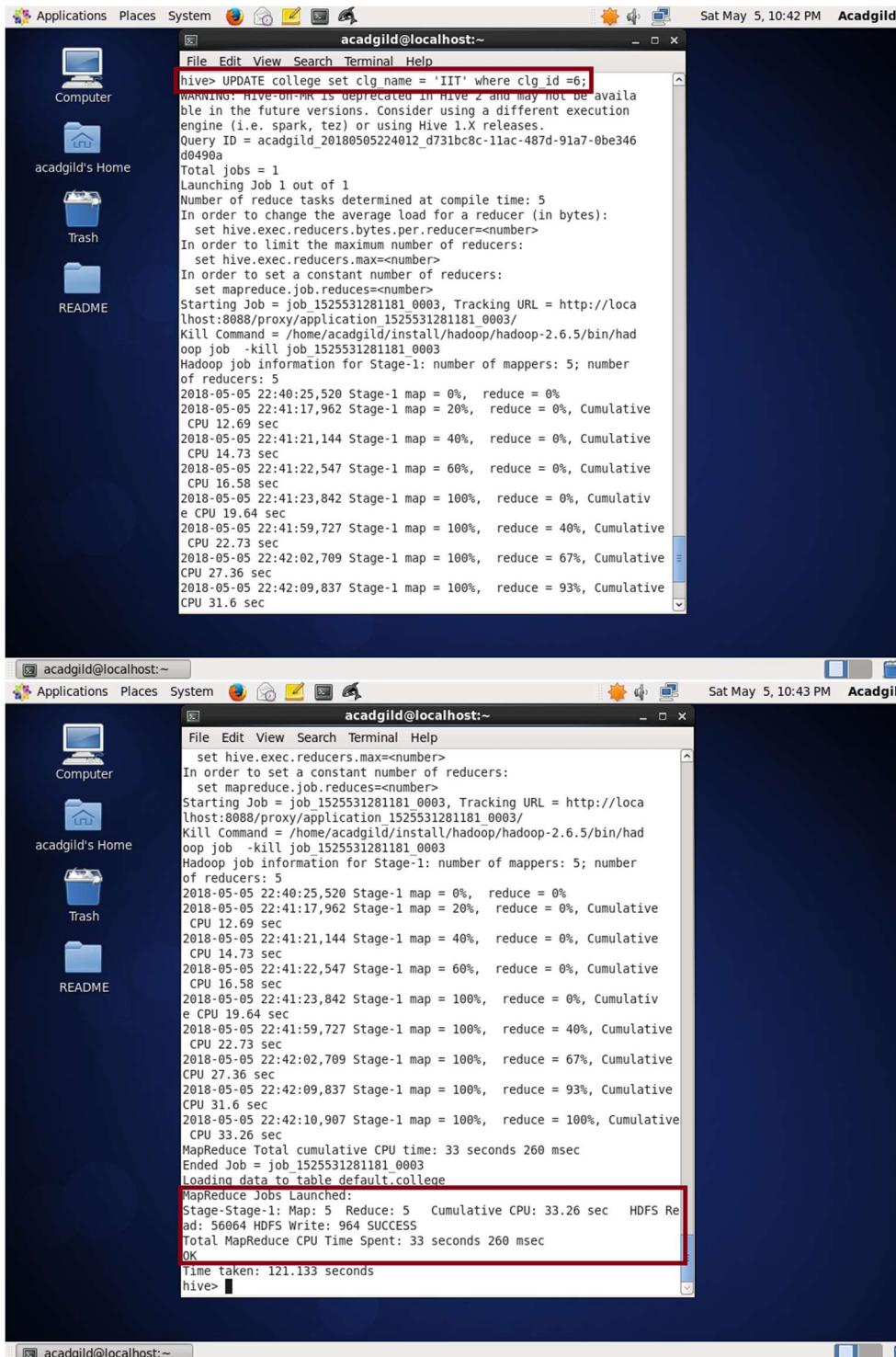
```
File Edit View Search Terminal Help
hive> UPDATE college set clg_id = 8 where clg_id =7;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported. Column clg_id.
hive>
```

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.

In this table, we have bucketed the '`clg_id`' column and performing the Update operation on the same column, so we have go the error

**FAILED: SemanticException[Error 10302]: Updating values of bucketing columns is not supported.
Column clg_id**

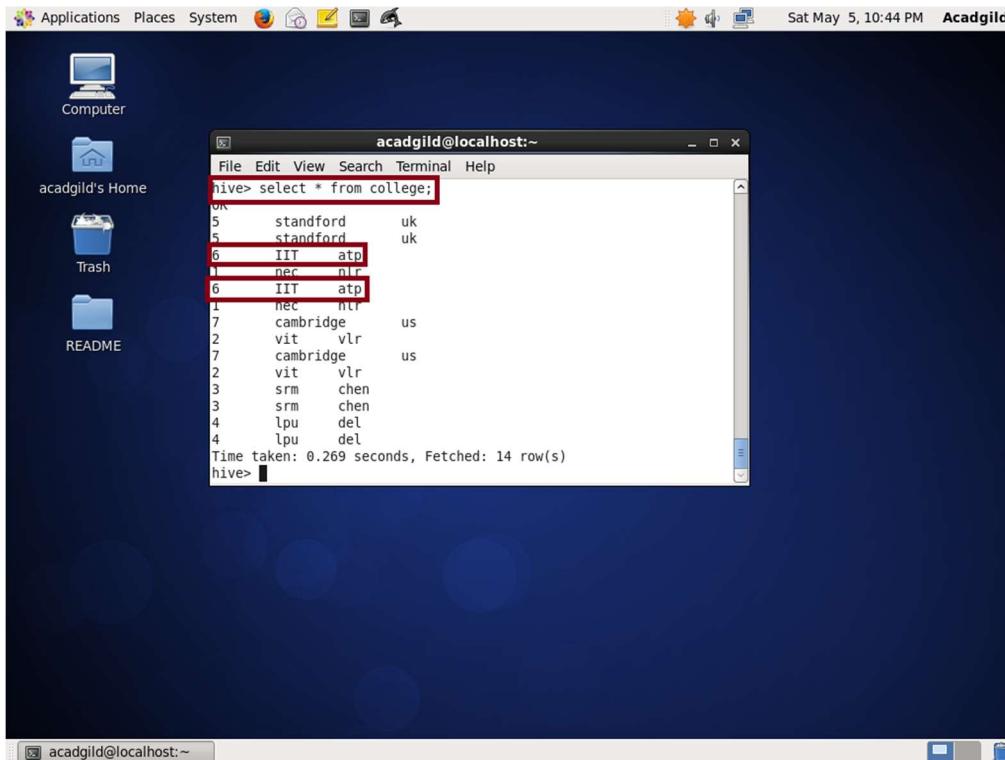
Now let's perform the update operation on Non bucketed column



```
acadgild@localhost:~$ Applications Places System Terminal Help
acadgild@localhost:~$ hive> UPDATE college set clg_name = 'IIT' where clg_id =6;
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_20180505224012_d731bc8c-1lac-487d-91a7-0be346d0490a
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_1525531281181_0003, Tracking URL = http://localhost:8088/proxy/application_1525531281181_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525531281181_0003
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-05-05 22:40:25,520 Stage-1 map = 0%,  reduce = 0%
2018-05-05 22:41:17,962 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 12.69 sec
2018-05-05 22:41:21,144 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 14.73 sec
2018-05-05 22:41:22,547 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 16.58 sec
2018-05-05 22:41:23,842 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 19.64 sec
2018-05-05 22:41:59,727 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 22.73 sec
2018-05-05 22:42:02,709 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 27.36 sec
2018-05-05 22:42:09,837 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 31.6 sec
acadgild@localhost:~$ Applications Places System Terminal Help
acadgild@localhost:~$ set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1525531281181_0003, Tracking URL = http://localhost:8088/proxy/application_1525531281181_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job -kill job_1525531281181_0003
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-05-05 22:40:25,520 Stage-1 map = 0%,  reduce = 0%
2018-05-05 22:41:17,962 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 12.69 sec
2018-05-05 22:41:21,144 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 14.73 sec
2018-05-05 22:41:22,547 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 16.58 sec
2018-05-05 22:41:23,842 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 19.64 sec
2018-05-05 22:41:59,727 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 22.73 sec
2018-05-05 22:42:02,709 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 27.36 sec
2018-05-05 22:42:09,837 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 31.6 sec
2018-05-05 22:42:10,907 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 33.26 sec
MapReduce Total cumulative CPU time: 33 seconds 260 msec
Ended Job = job_1525531281181_0003
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5 Reduce: 5 Cumulative CPU: 33.26 sec  HDFS Read: 56664 HDFS Write: 964 SUCCESS
Total MapReduce CPU Time Spent: 33 seconds 260 msec
OK
Time taken: 121.133 seconds
hive>
```

We have successfully updated the data.

The updated data can be checked using the command [select * from college](#).



A screenshot of a Linux desktop environment. The desktop background is dark blue with a blurred circular pattern. At the top, there is a panel with icons for Applications, Places, System, and network status. The date and time are shown as Sat May 5, 10:44 PM. The user is logged in as Acadgild. On the left, there is a Nautilus file manager window showing the home directory with icons for Computer, acadgild's Home, Trash, and README. In the center, there is a terminal window titled "acadgild@localhost:~". The terminal shows the following output:

```
hive> select * from college;
OK
5      standford    uk
5      standford    uk
6      IIT          atp
1      nec          nlr
6      IIT          atp
1      nec          nlr
7      cambridge    us
2      vit          vlr
7      cambridge    us
2      vit          vlr
3      srm          chen
3      srm          chen
4      lpu          del
4      lpu          del
Time taken: 0.269 seconds, Fetched: 14 row(s)
hive>
```

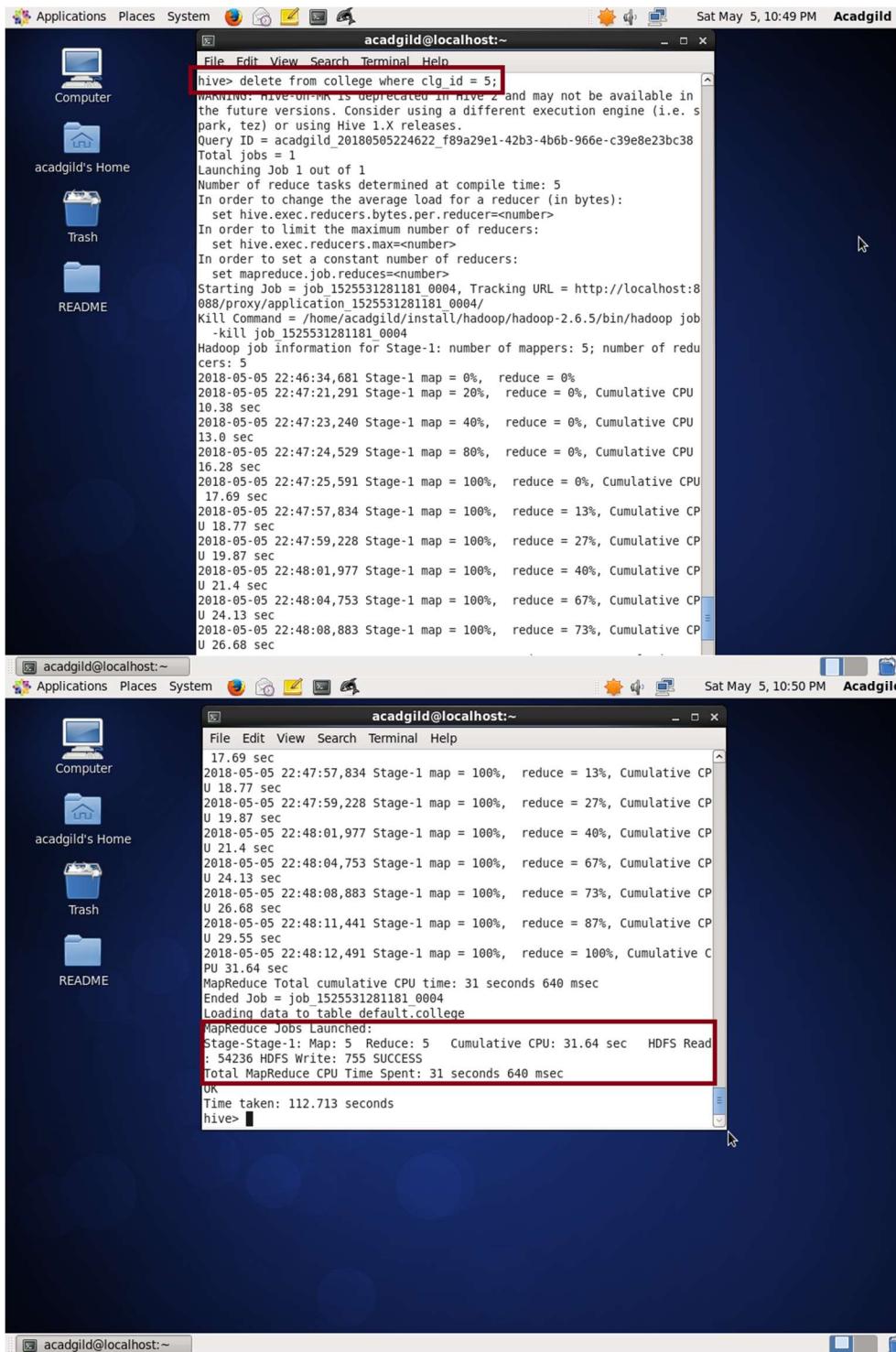
We can see that the data has been updated successfully.

Now let's perform the Delete operation on the same table.

Deleting a Row from Hive Table

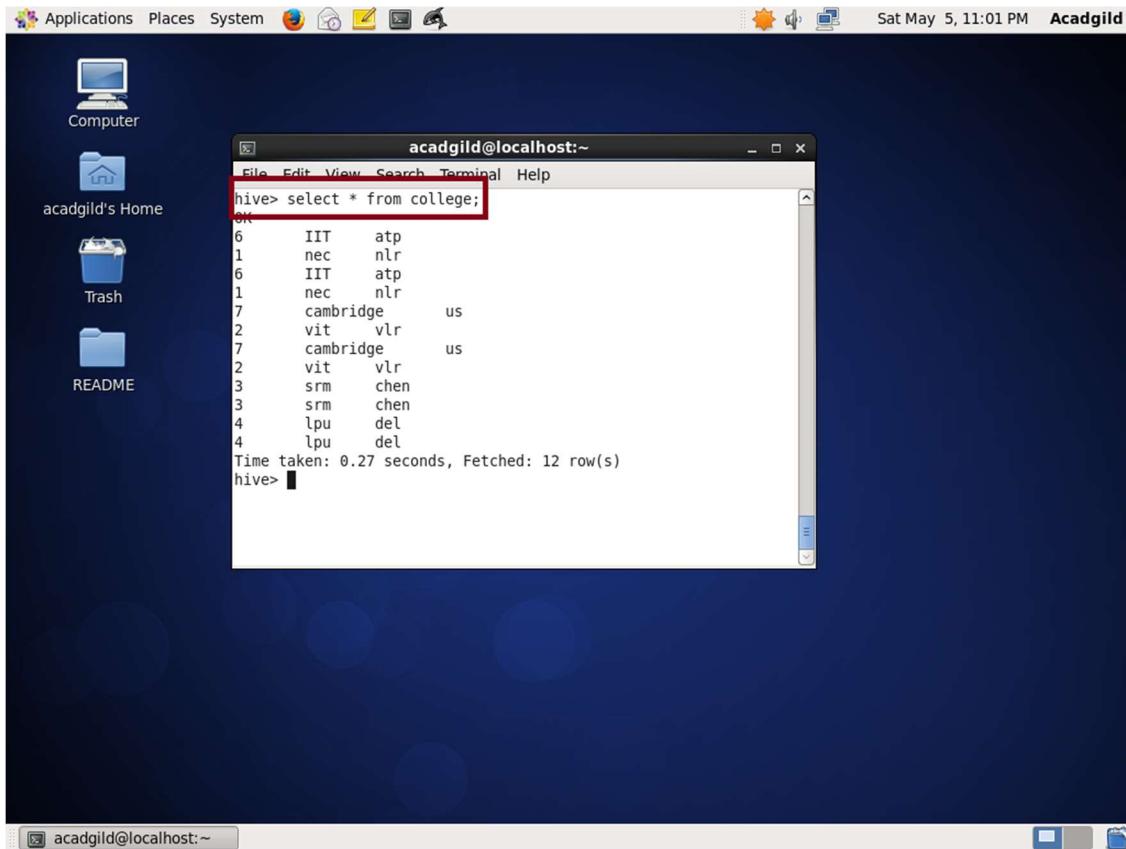
The above command will delete a single row in the Hive table.

delete from college where clg_id = 5;



```
File Edit View Search Terminal Help
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 engine (i.e. s
park, tez) or using Hive 1.X releases.
Query ID = acadgild_20180505224622_f89a29e1-42b3-4b6b-966e-c39e8e23bc38
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_1525531281181_0004, Tracking URL = http://localhost:8
088/proxy/application_1525531281181_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job
 -kill job_1525531281181_0004
Hadoop job information for Stage-1: number of mappers: 5; number of reduc
ers: 5
2018-05-05 22:46:34,681 Stage-1 map = 0%,  reduce = 0%
2018-05-05 22:47:21,291 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU
10.38 sec
2018-05-05 22:47:23,240 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU
13.0 sec
2018-05-05 22:47:24,529 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU
16.28 sec
2018-05-05 22:47:25,591 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU
17.69 sec
2018-05-05 22:47:57,834 Stage-1 map = 100%,  reduce = 13%, Cumulative CP
U 18.77 sec
2018-05-05 22:47:59,228 Stage-1 map = 100%,  reduce = 27%, Cumulative CP
U 19.87 sec
2018-05-05 22:48:01,977 Stage-1 map = 100%,  reduce = 40%, Cumulative CP
U 21.4 sec
2018-05-05 22:48:04,753 Stage-1 map = 100%,  reduce = 67%, Cumulative CP
U 24.13 sec
2018-05-05 22:48:08,883 Stage-1 map = 100%,  reduce = 73%, Cumulative CP
U 26.68 sec
File Edit View Search Terminal Help
17.69 sec
2018-05-05 22:47:57,834 Stage-1 map = 100%,  reduce = 13%, Cumulative CP
U 18.77 sec
2018-05-05 22:47:59,228 Stage-1 map = 100%,  reduce = 27%, Cumulative CP
U 19.87 sec
2018-05-05 22:48:01,977 Stage-1 map = 100%,  reduce = 40%, Cumulative CP
U 21.4 sec
2018-05-05 22:48:04,753 Stage-1 map = 100%,  reduce = 67%, Cumulative CP
U 24.13 sec
2018-05-05 22:48:08,883 Stage-1 map = 100%,  reduce = 73%, Cumulative CP
U 29.55 sec
2018-05-05 22:48:12,491 Stage-1 map = 100%,  reduce = 100%, Cumulative C
PU 31.64 sec
MapReduce Total cumulative CPU time: 31 seconds 640 msec
Ended Job = job_1525531281181_0004
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5 Reduce: 5 Cumulative CPU: 31.64 sec  HDFS Read
: 54236 HDFS Write: 755 SUCCESS
Total MapReduce CPU Time Spent: 31 seconds 640 msec
OK.
Time taken: 112.713 seconds
hive>
```

We have now successfully deleted a row from the Hive table. This can be checked using the command **select * from college**



A screenshot of a Linux desktop environment. The desktop background is dark blue with a blurred circular pattern. At the top, there is a horizontal menu bar with icons for Applications, Places, System, and several system status indicators. The date and time "Sat May 5, 11:01 PM" and the user name "Acadgild" are visible on the right side of the menu bar. Below the menu bar is a dock containing icons for Computer, Home, Trash, and README. A terminal window titled "acadgild@localhost:~" is open in the foreground. The window shows the output of a Hive query:

```
hive> select * from college;
OK
6    IIT    atp
1    nec    nlr
6    IIT    atp
1    nec    nlr
7    cambridge    us
2    vit    vlr
7    cambridge    us
2    vit    vlr
3    srm    chen
3    srm    chen
4    lpu    del
4    lpu    del
Time taken: 0.27 seconds, Fetched: 12 row(s)
hive>
```

We can see that there is no row with **clg_id =5**. This means that we have successfully deleted the row from the Hive table.

This is how the transactions or row-wise operations are performed in Hive.