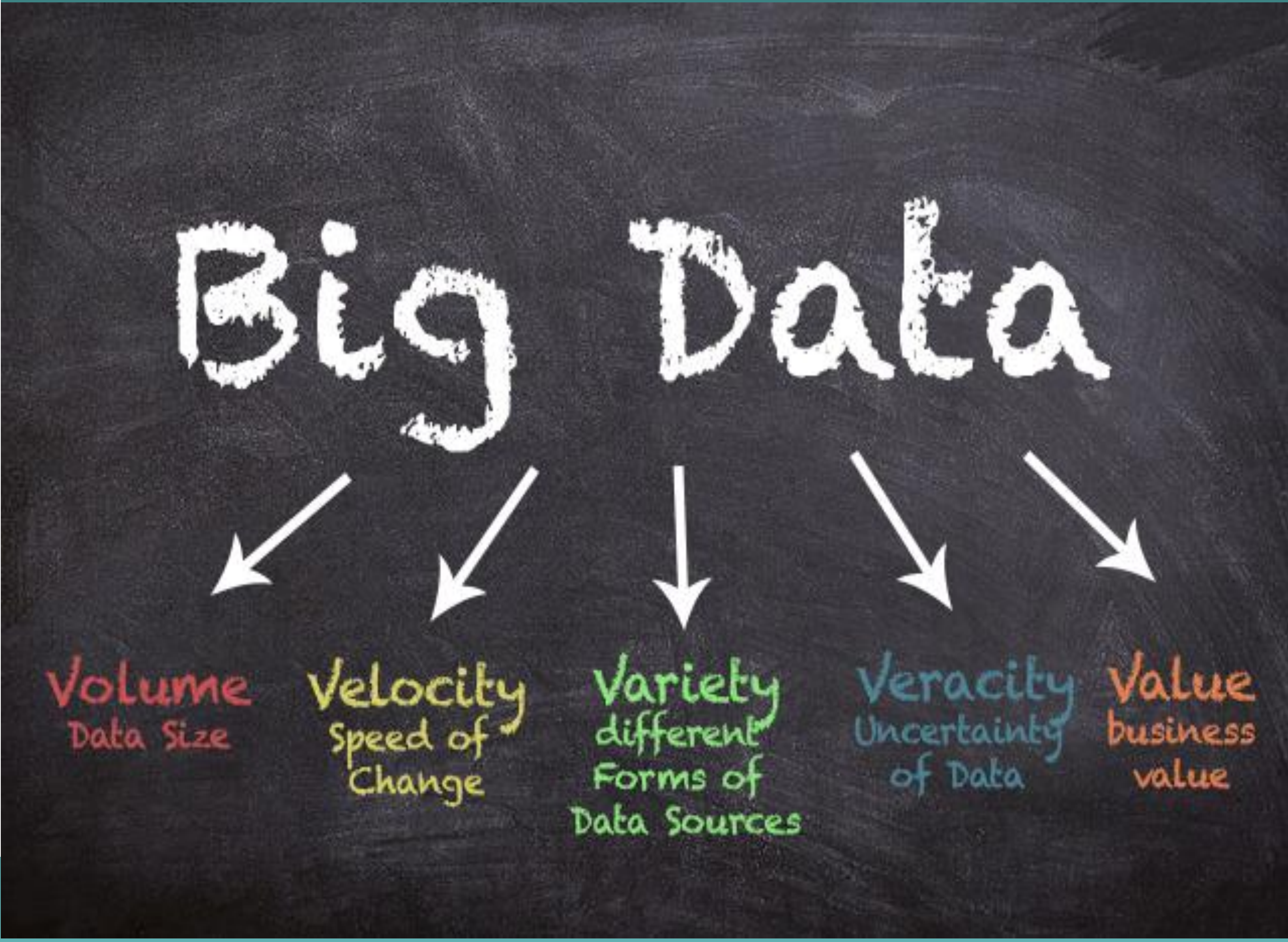
08/08/2018 | By: Ajit K Prasad



Big Data Engineering with Hadoop & Spark

Assignment on Advance Hive

Acadgild

Session 9: Assignment 9.1

This assignment is aimed at consolidating the concepts that was learnt during the Advance Hive session of the course.

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*

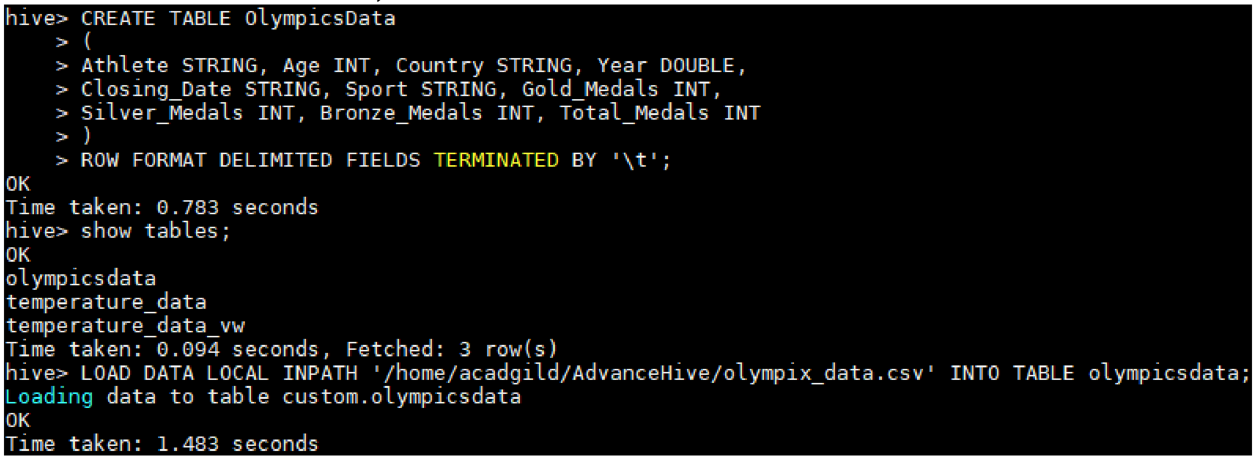
Dataset description:

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 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

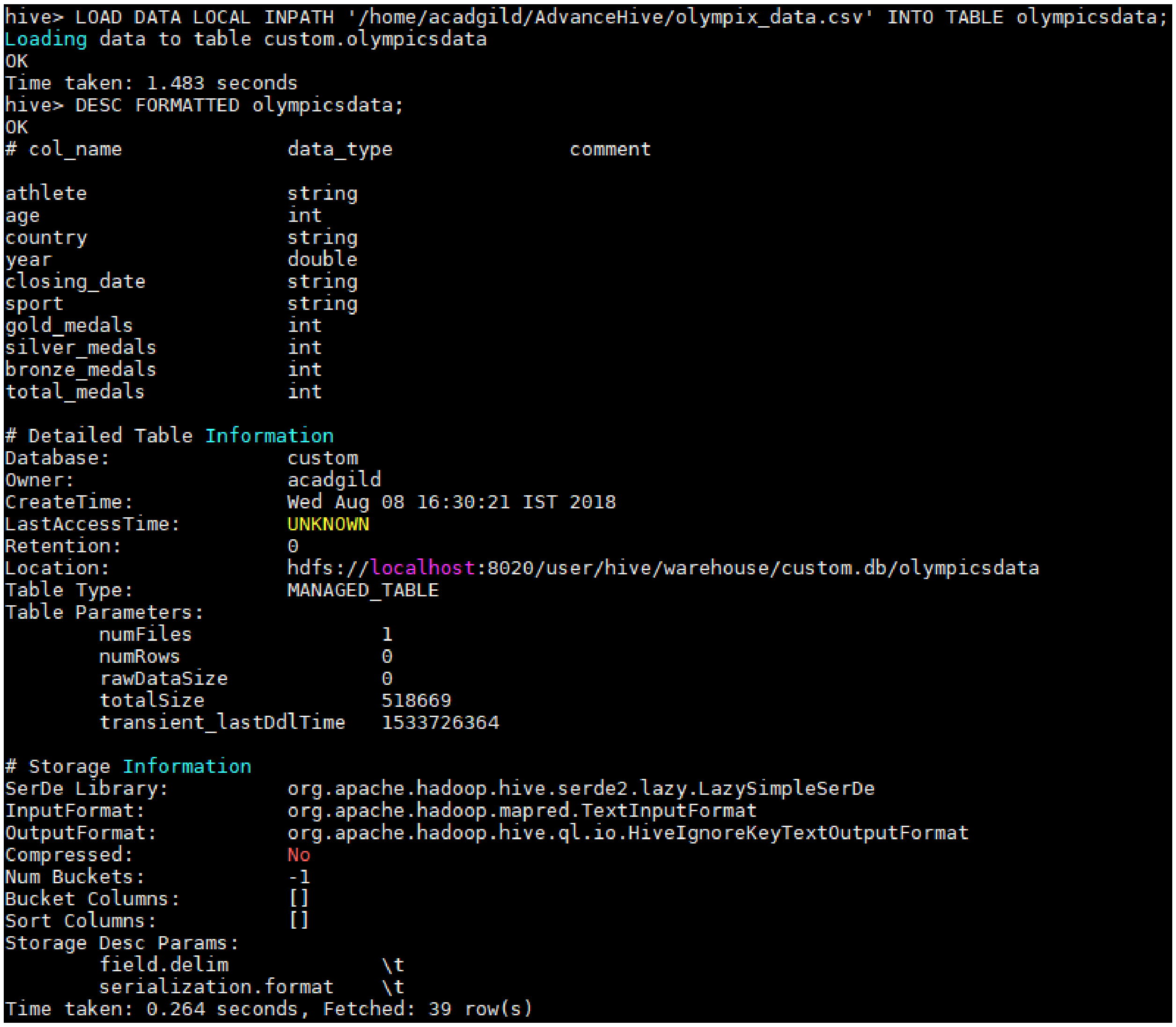
Prerequisites

- Create a table with the query command:
  hive> CREATE TABLE OlympicsData
  (
  Athlete STRING, Age INT, Country STRING, Year DOUBLE, Closing_Date STRING, Sport STRING, Gold_Medals INT, Silver_Medals INT, Bronze_Medals INT, Total_Medals INT
)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
- Check the table created with the query command:
  hive> show tables;



```
hive> CREATE TABLE OlympicsData
    > (
    > Athlete STRING, Age INT, Country STRING, Year DOUBLE,
    > Closing_Date STRING, Sport STRING, Gold_Medals INT,
    > Silver_Medals INT, Bronze_Medals INT, Total_Medals INT
    > )
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
OK
Time taken: 0.783 seconds
hive> show tables;
OK
olympicsdata
temperature_data
temperature_data_vw
Time taken: 0.094 seconds, Fetched: 3 row(s)
hive> LOAD DATA LOCAL INPATH '/home/acadgild/AdvanceHive/olympix_data.csv' INTO TABLE olympicsdata;
Loading data to table custom.olympicsdata
OK
Time taken: 1.483 seconds
```

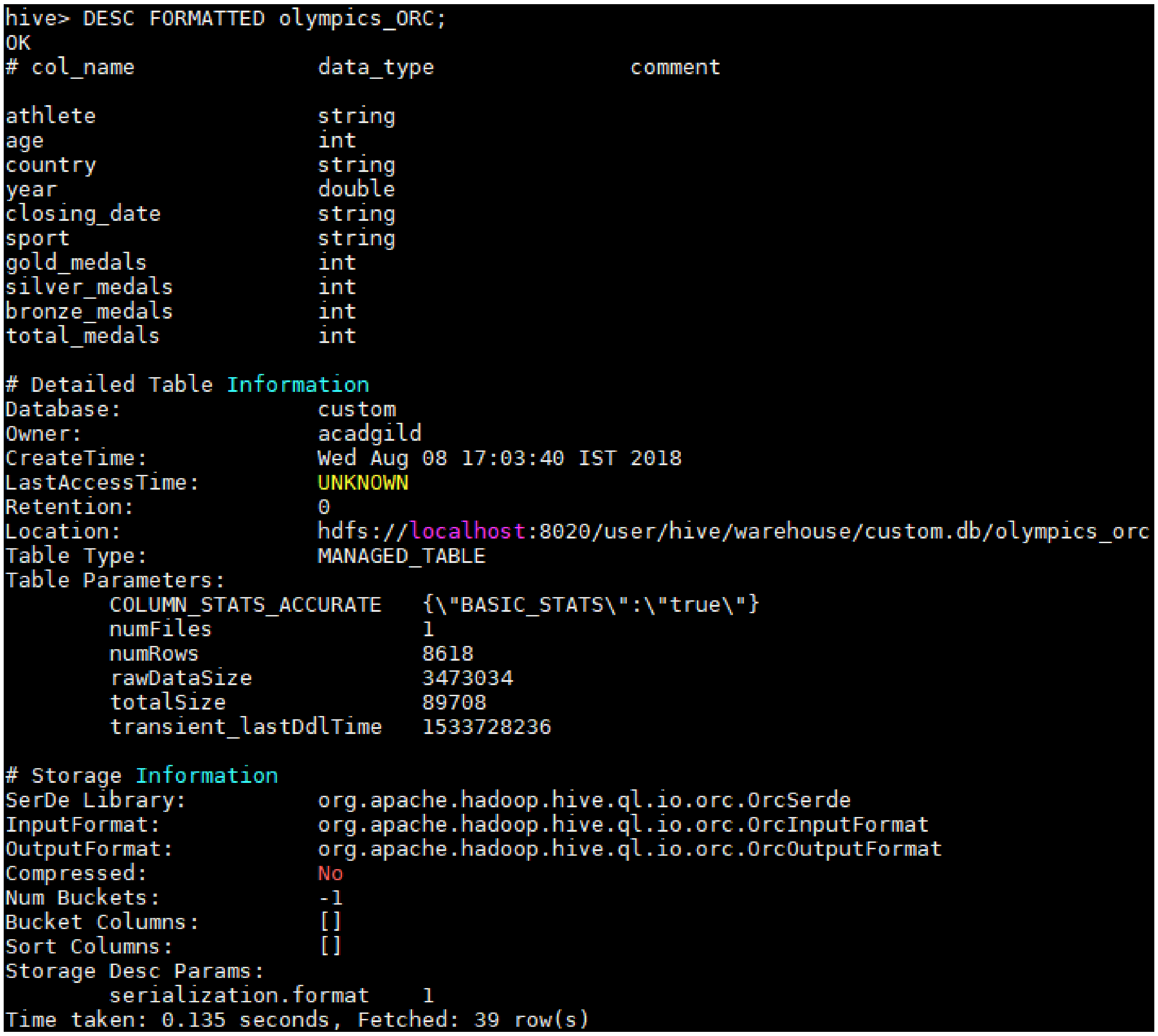
- Load the provided dataset ***"olympix_data"*** into the table ***"OlympicsData"***
  hive> LOAD DATA LOCAL INPATH '/home/acadgild/AdvanceHive/olympix_data.csv' INTO TABLE olympicsdata;
- Checking the detailed properties of the table created using query command:
  hive> DESC FORMATTED olympicsdata;



```
hive> LOAD DATA LOCAL INPATH '/home/acadgild/AdvanceHive/olympix_data.csv' INTO TABLE olympicsdata;
Loading data to table custom.olympicsdata
OK
Time taken: 1.483 seconds
hive> DESC FORMATTED olympicsdata;
OK
# col_name              data_type               comment

athlete                 string
age                     int
country                 string
year                    double
closing_date            string
sport                   string
gold_medals             int
silver_medals           int
bronze_medals           int
total_medals            int

# Detailed Table Information
Database:               custom
Owner:                  acadgild
CreateTime:             Wed Aug 08 16:30:21 IST 2018
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://localhost:8020/user/hive/warehouse/custom.db/olympicsdata
Table Type:             MANAGED_TABLE
Table Parameters:
        numFiles                1
        numRows                 0
        rawDataSize             0
        totalSize               518669
        transient_lastDdlTime   1533726364

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             \t
        serialization.format    \t
Time taken: 0.264 seconds, Fetched: 39 row(s)
```

- While executing the above command it was seen that the data size in ***"OlympicsData"*** table is big, hence create an ***ORC table*** with the same fields.
- ***ORC table*** allows to effectively manage space & makes querying data much more efficient & effective.

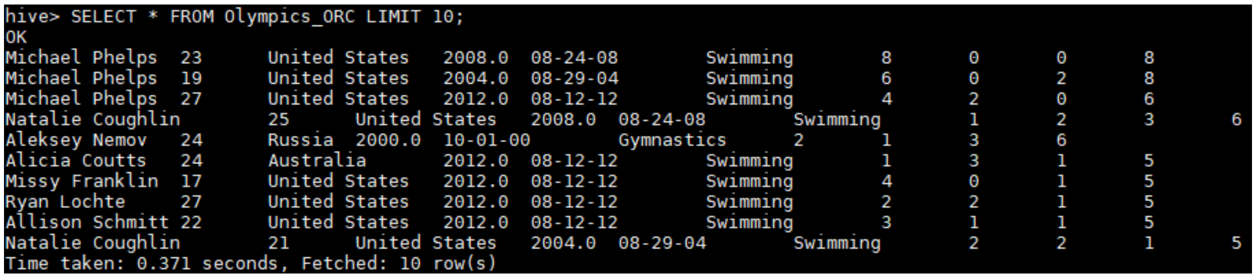
- Create an ***ORC table*** with the query command:
  hive> CREATE TABLE Olympics_ORC
  (
  Athlete STRING, Age INT, Country STRING, Year DOUBLE,
  Closing_Date STRING, Sport STRING, Gold_Medals INT,
  Silver_Medals INT, Bronze_Medals INT, Total_Medals INT
)
  STORED AS ORC;
- Load data from the ***"OlympicsData"*** table into ***"Olympics_ORC"*** table with the query command:
  *hive> FROM OlympicsData INSERT INTO Olympics_ORC SELECT *;*



```
hive> DESC FORMATTED olympics_ORC;
OK
# col_name              data_type               comment

athlete                 string
age                     int
country                 string
year                    double
closing_date            string
sport                   string
gold_medals             int
silver_medals           int
bronze_medals           int
total_medals            int

# Detailed Table Information
Database:               custom
Owner:                  acadgild
CreateTime:             Wed Aug 08 17:03:40 IST 2018
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://localhost:8020/user/hive/warehouse/custom.db/olympics_orc
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        numFiles                1
        numRows                 8618
        rawDataSize             3473034
        totalSize               89708
        transient_lastDdlTime   1533728236

# Storage Information
SerDe Library:          org.apache.hadoop.hive.ql.io.orc.OrcSerde
InputFormat:            org.apache.hadoop.hive.ql.io.orc.OrcInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        serialization.format    1
Time taken: 0.135 seconds, Fetched: 39 row(s)
```

- It can be easily observed that, there is almost five-fold difference between both the table's data

– Check top 10 rows of the ORC table created with query command
  *hive> SELECT * FROM Olympics_ORC LIMIT 10;*



```
hive> SELECT * FROM Olympics_ORC LIMIT 10;
OK
Michael Phelps  23      United States   2008.0  08-24-08        Swimming        8       0       0       8
Michael Phelps  19      United States   2004.0  08-29-04        Swimming        6       0       2       8
Michael Phelps  27      United States   2012.0  08-12-12        Swimming        4       2       0       6
Natalie Coughlin        25      United States   2008.0  08-24-08        Swimming        1       2       3       6
Aleksey Nemov   24      Russia  2000.0  10-01-00        Gymnastics      2       1       3       6
Alicia Coutts   24      Australia       2012.0  08-12-12        Swimming        1       3       1       5
Missy Franklin  17      United States   2012.0  08-12-12        Swimming        4       0       1       5
Ryan Lochte     27      United States   2012.0  08-12-12        Swimming        2       2       1       5
Allison Schmitt 22      United States   2012.0  08-12-12        Swimming        3       1       1       5
Natalie Coughlin        21      United States   2004.0  08-29-04        Swimming        2       2       1       5
Time taken: 0.371 seconds, Fetched: 10 row(s)
```

Problem Statement

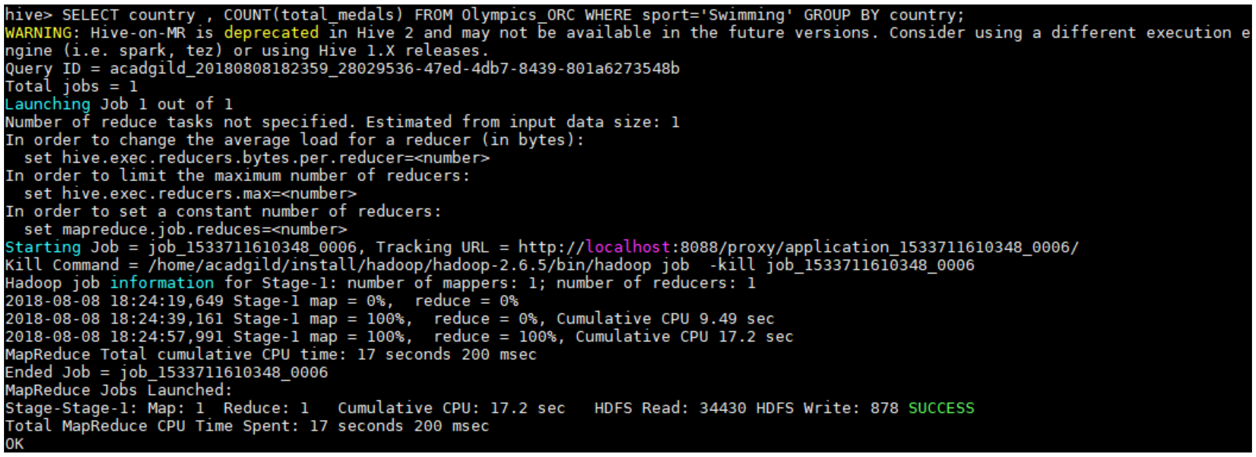
Task 1:

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

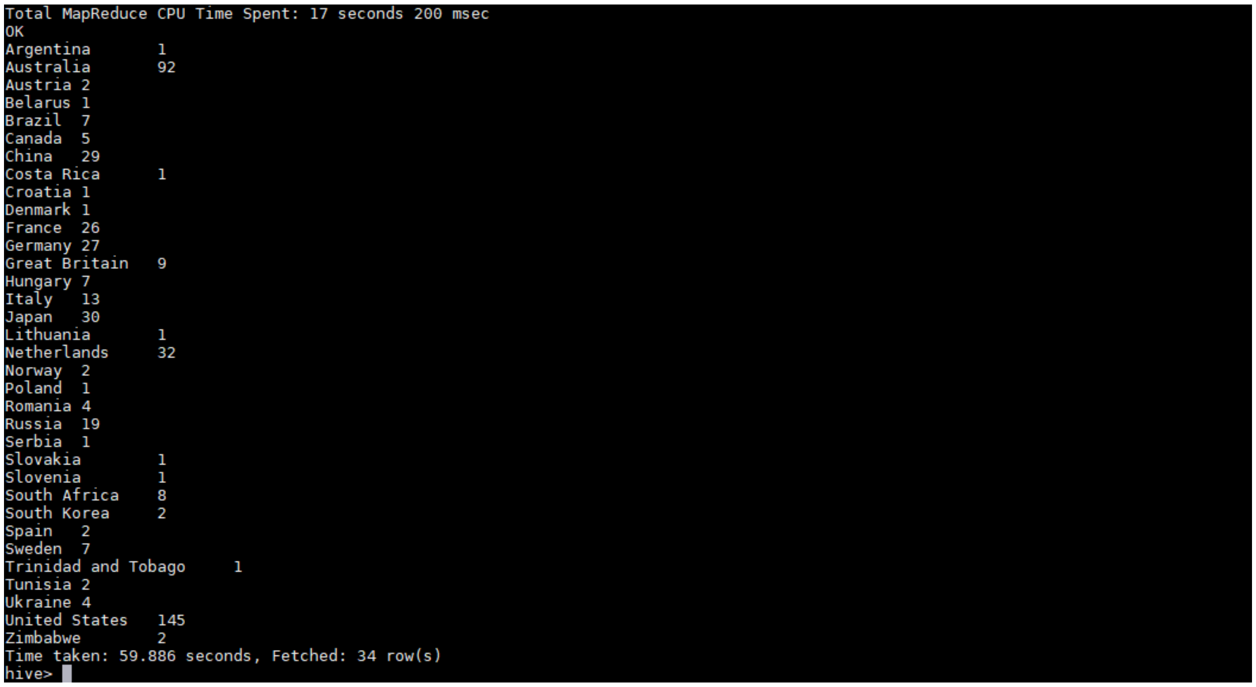
Solution:

– To perform the task, use the query command:
  hive> SELECT country, COUNT(total_medals) FROM Olympics_ORC WHERE sport='Swimming' GROUP BY country;

Output:



```
hive> SELECT country , COUNT(total_medals) FROM Olympics_ORC 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 e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808182359_28029536-47ed-4db7-8439-801a6273548b
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_1533711610348_0006, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0006/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-08 18:24:19,649 Stage-1 map = 0%,  reduce = 0%
2018-08-08 18:24:39,161 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 9.49 sec
2018-08-08 18:24:57,991 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 17.2 sec
MapReduce Total cumulative CPU time: 17 seconds 200 msec
Ended Job = job_1533711610348_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 17.2 sec   HDFS Read: 34430 HDFS Write: 878 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 200 msec
OK
```



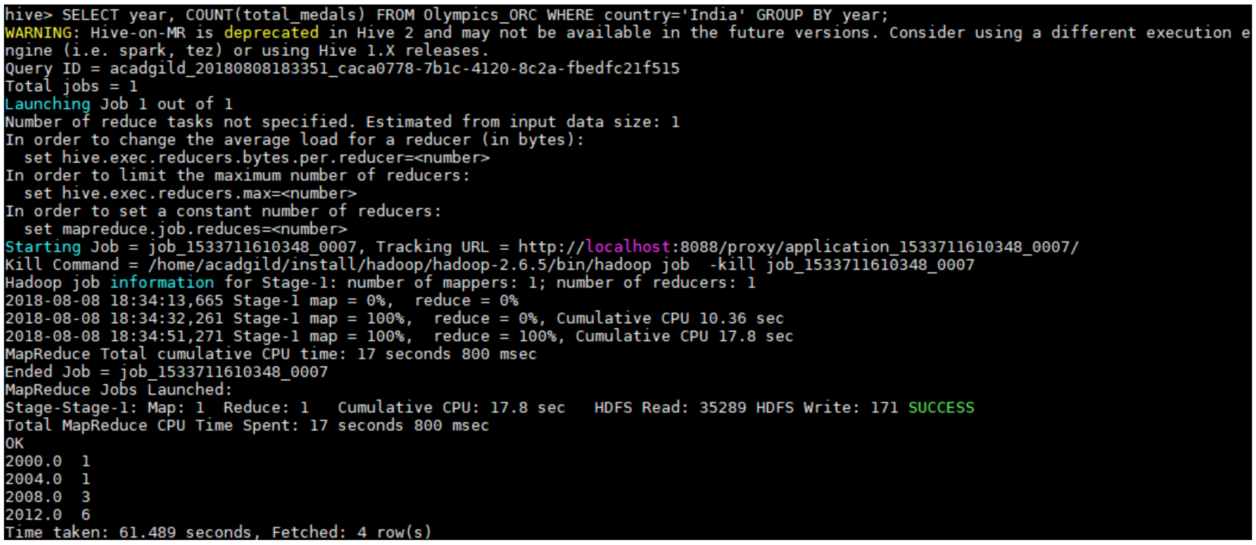
```
Total MapReduce CPU Time Spent: 17 seconds 200 msec
OK
Argentina       1
Australia       92
Austria 2
Belarus 1
Brazil  7
Canada  5
China   29
Costa Rica      1
Croatia 1
Denmark 1
France  26
Germany 27
Great Britain   9
Hungary 7
Italy   13
Japan   30
Lithuania       1
Netherlands     32
Norway  2
Poland  1
Romania 4
Russia  19
Serbia  1
Slovakia        1
Slovenia        1
South Africa    8
South Korea     2
Spain   2
Sweden  7
Trinidad and Tobago     1
Tunisia 2
Ukraine 4
United States   145
Zimbabwe        2
Time taken: 59.886 seconds, Fetched: 34 row(s)
hive>
```

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

Solution:

– To perform the task, use the query command:
  hive> SELECT year, COUNT(total_medals) FROM Olympics_ORC WHERE country='India' GROUP BY year;

Output:



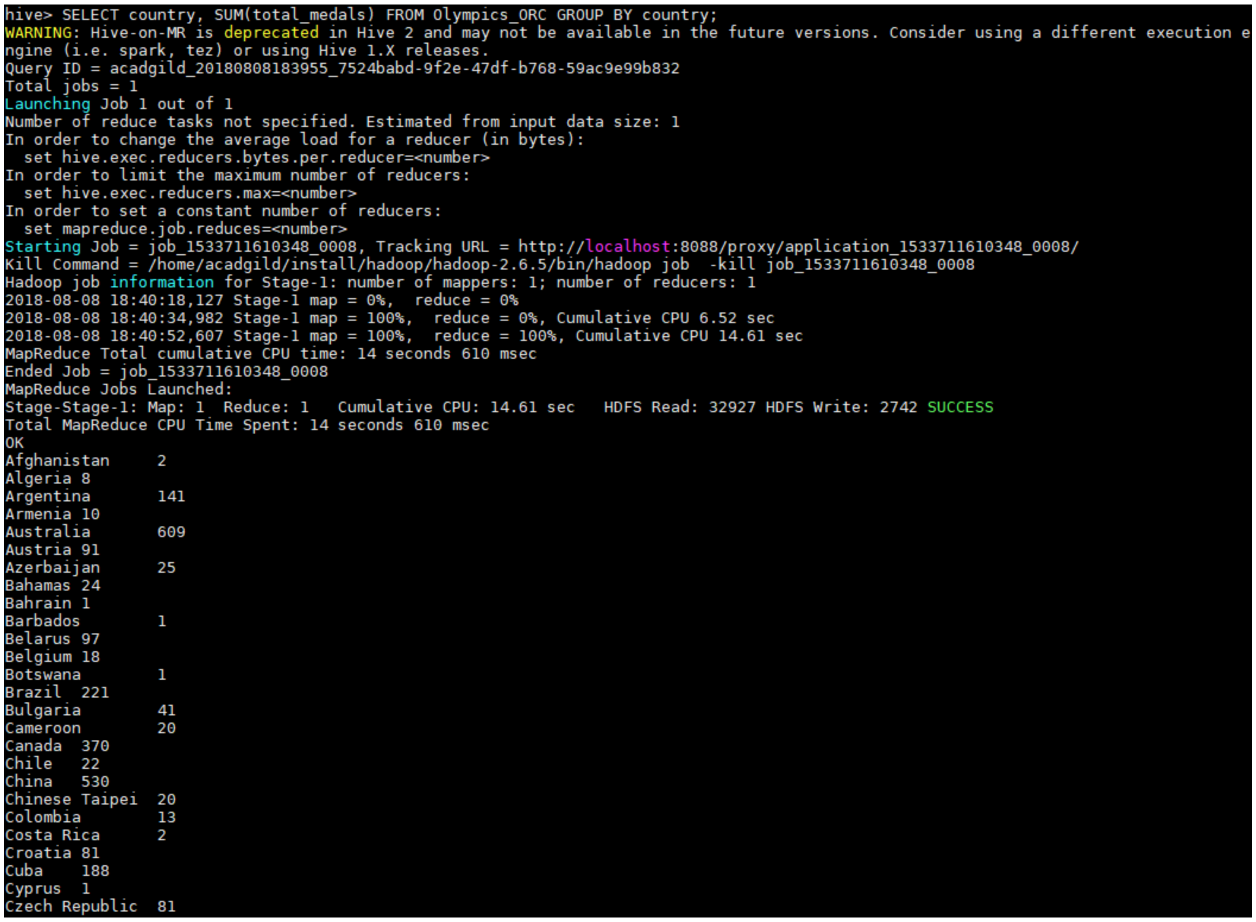
```
hive> SELECT year, COUNT(total_medals) FROM Olympics_ORC WHERE country='India' GROUP BY year;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808183351_caca0778-7b1c-4120-8c2a-fbedfc21f515
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_1533711610348_0007, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0007/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-08 18:34:13,665 Stage-1 map = 0%,  reduce = 0%
2018-08-08 18:34:32,261 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 10.36 sec
2018-08-08 18:34:51,271 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 17.8 sec
MapReduce Total cumulative CPU time: 17 seconds 800 msec
Ended Job = job_1533711610348_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 17.8 sec   HDFS Read: 35289 HDFS Write: 171 SUCCESS
Total MapReduce CPU Time Spent: 17 seconds 800 msec
OK
2000.0  1
2004.0  1
2008.0  3
2012.0  6
Time taken: 61.489 seconds, Fetched: 4 row(s)
```

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

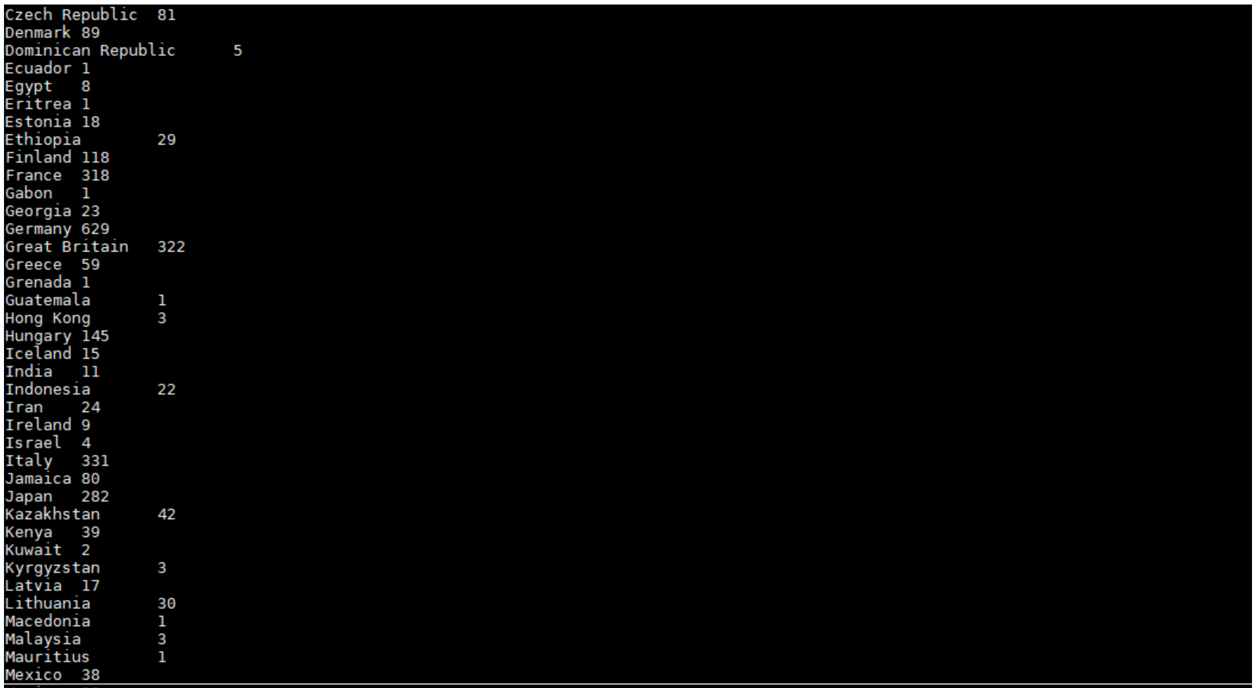
Solution:

– To perform the task, use the query command:
hive> SELECT country, SUM(total_medals) FROM Olympics_ORC GROUP BY country;

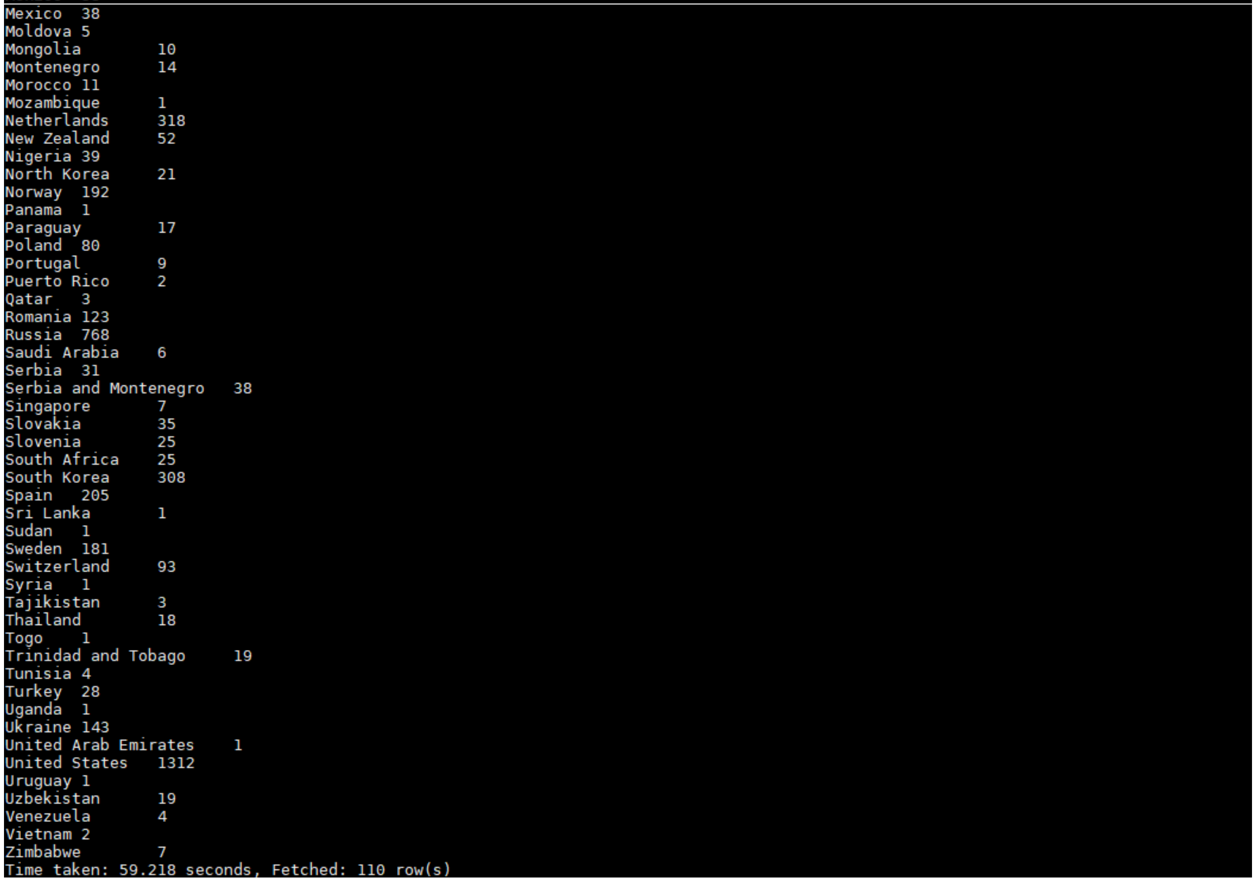
Output:



```
hive> SELECT country, SUM(total_medals) FROM Olympics_ORC 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 e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808183955_7524babd-9f2e-47df-b768-59ac9e99b832
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_1533711610348_0008, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0008/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-08 18:40:18,127 Stage-1 map = 0%,  reduce = 0%
2018-08-08 18:40:34,982 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.52 sec
2018-08-08 18:40:52,607 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 14.61 sec
MapReduce Total cumulative CPU time: 14 seconds 610 msec
Ended Job = job_1533711610348_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 14.61 sec   HDFS Read: 32927 HDFS Write: 2742 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 610 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
```



```
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
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
```



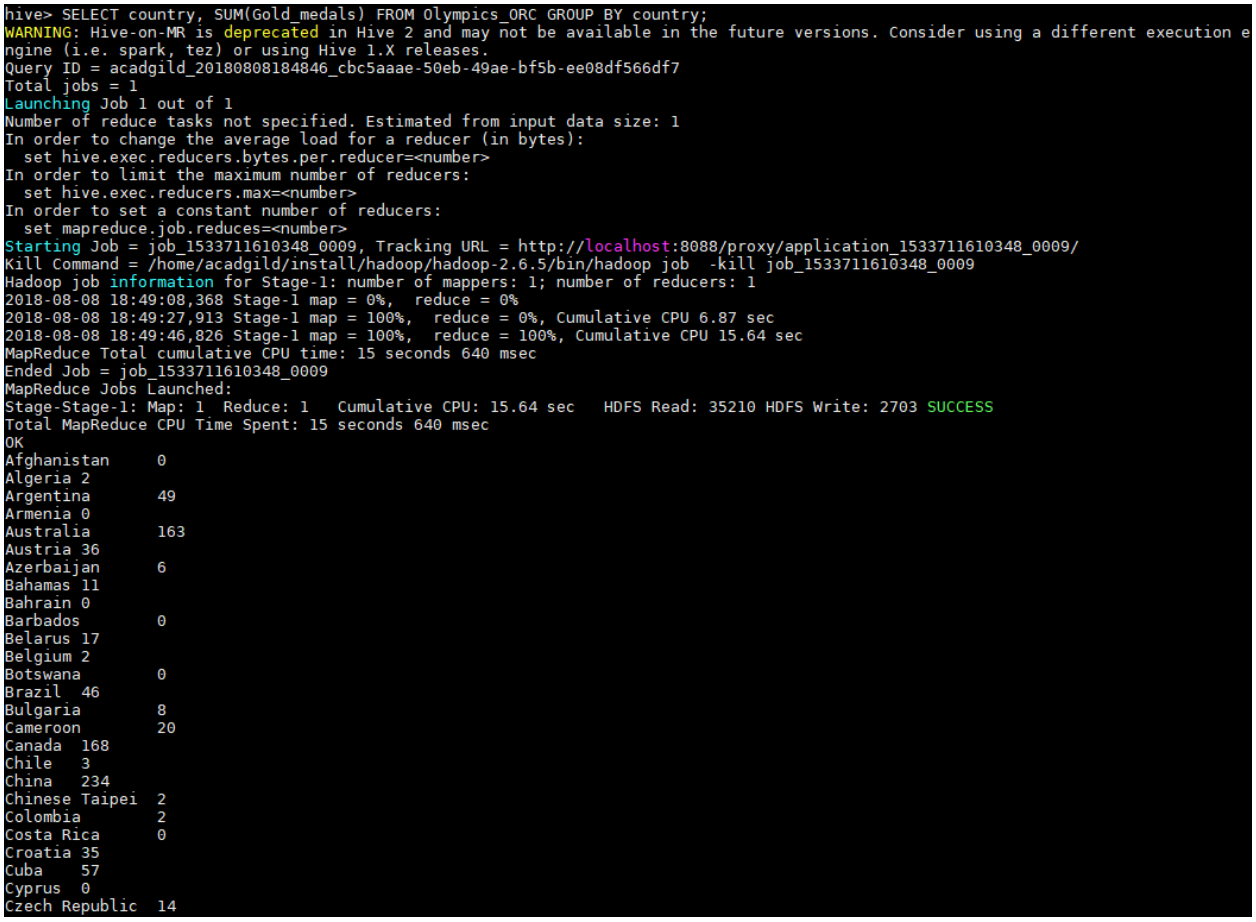
```
Mexico  38
Moldova 5
Mongolia        10
Montenegro      14
Morocco 11
Mozambique      1
Netherlands     318
New Zealand     52
Nigeria 39
North Korea     21
Norway  192
Panama  1
Paraguay        17
Poland  80
Portugal        9
Puerto Rico     2
Qatar   3
Romania 123
Russia  768
Saudi Arabia    6
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: 59.218 seconds, Fetched: 110 row(s)
```

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

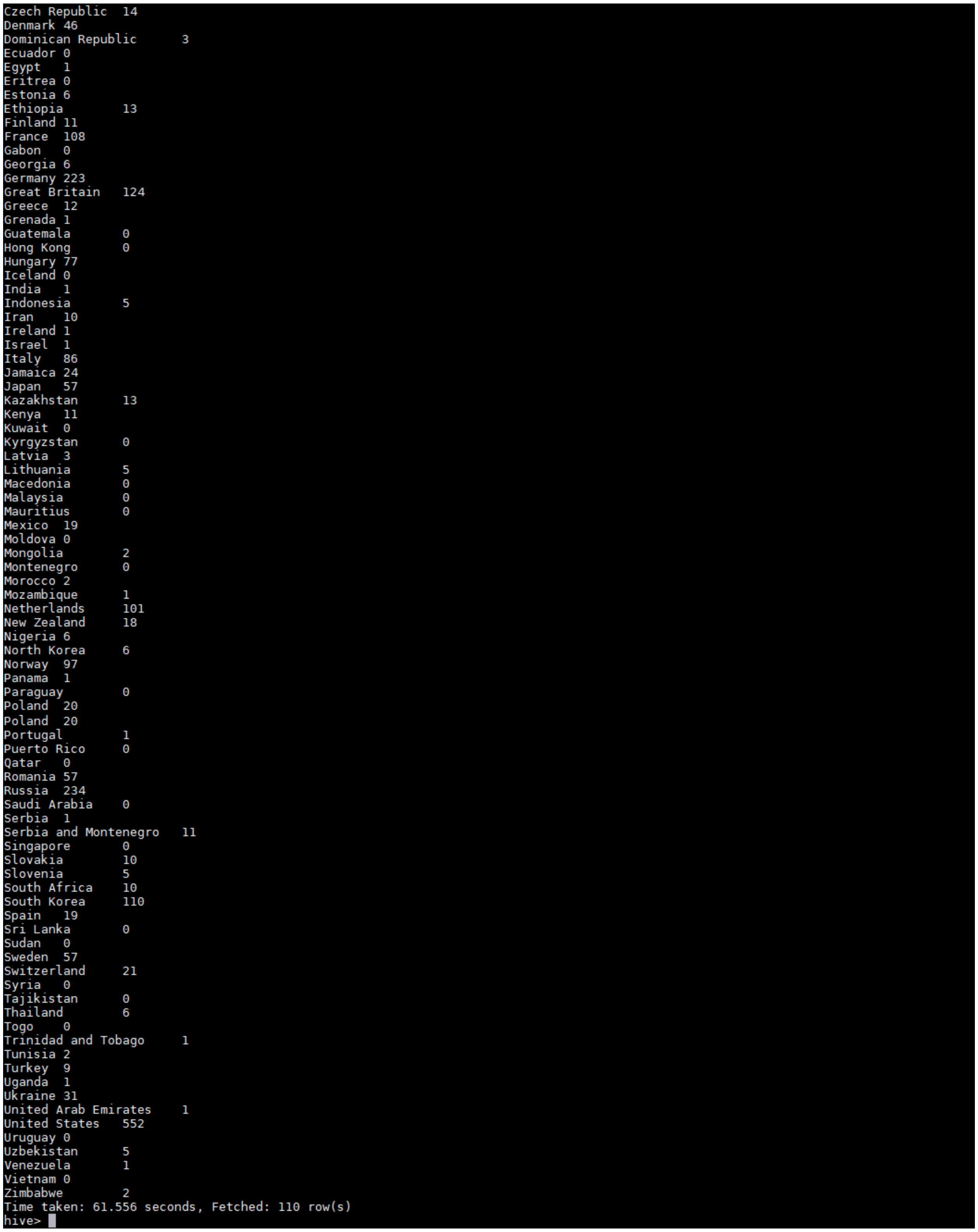
Solution:

– To perform the task, use the query command:
hive> SELECT country, SUM(Gold_medals) FROM Olympics_ORC GROUP BY country;

Output:



```
hive> SELECT country, SUM(Gold_medals) FROM Olympics_ORC 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 e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808184846_cbc5aaae-50eb-49ae-bf5b-ee08df566df7
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_1533711610348_0009, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0009/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-08 18:49:08,368 Stage-1 map = 0%,  reduce = 0%
2018-08-08 18:49:27,913 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.87 sec
2018-08-08 18:49:46,826 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 15.64 sec
MapReduce Total cumulative CPU time: 15 seconds 640 msec
Ended Job = job_1533711610348_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 15.64 sec   HDFS Read: 35210 HDFS Write: 2703 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 640 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
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
```



```
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
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
Panama  1
Paraguay        0
Poland  20
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: 61.556 seconds, Fetched: 110 row(s)
hive>
```

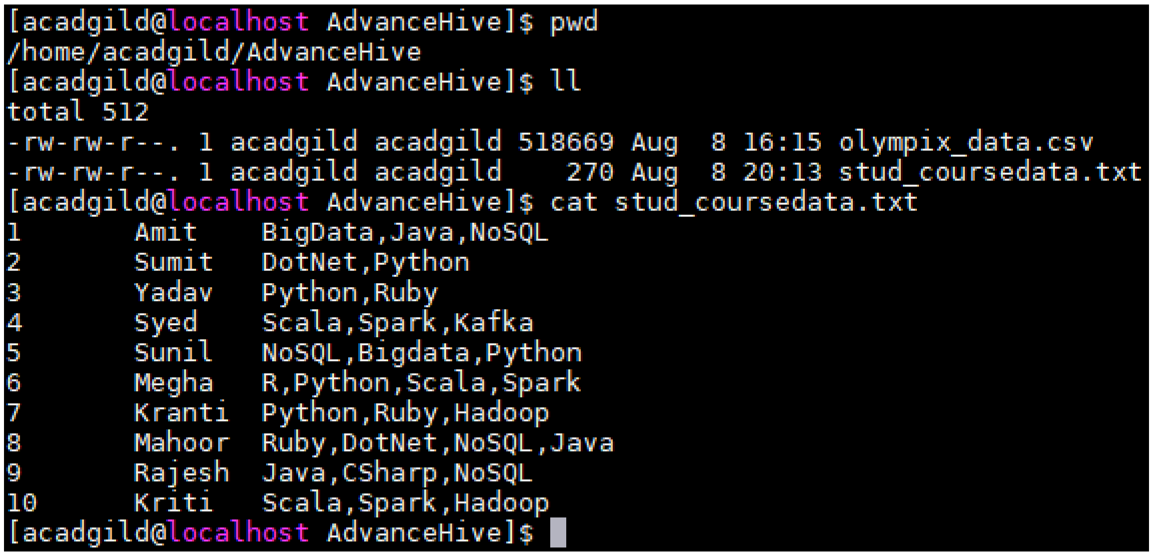
Task 2:

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.

Solution:

– We had a text file ***"stud_coursedata.txt"*** on local which populated the following data to be loaded into a table

– To check the data on the file , use query command
$ cat stud_coursedata.txt



```
[acadgild@localhost AdvanceHive]$ pwd
/home/acadgild/AdvanceHive
[acadgild@localhost AdvanceHive]$ ll
total 512
-rw-rw-r--. 1 acadgild acadgild 518669 Aug  8 16:15 olympix_data.csv
-rw-rw-r--. 1 acadgild acadgild    270 Aug  8 20:13 stud_coursedata.txt
[acadgild@localhost AdvanceHive]$ cat stud_coursedata.txt
1       Amit    BigData,Java,NoSQL
2       Sumit   DotNet,Python
3       Yadav   Python,Ruby
4       Syed    Scala,Spark,Kafka
5       Sunil   NoSQL,Bigdata,Python
6       Megha   R,Python,Scala,Spark
7       Kranti  Python,Ruby,Hadoop
8       Mahoor  Ruby,DotNet,NoSQL,Java
9       Rajesh  Java,CSharp,NoSQL
10      Kriti   Scala,Spark,Hadoop
[acadgild@localhost AdvanceHive]$
```

– Create a table to load the data file above

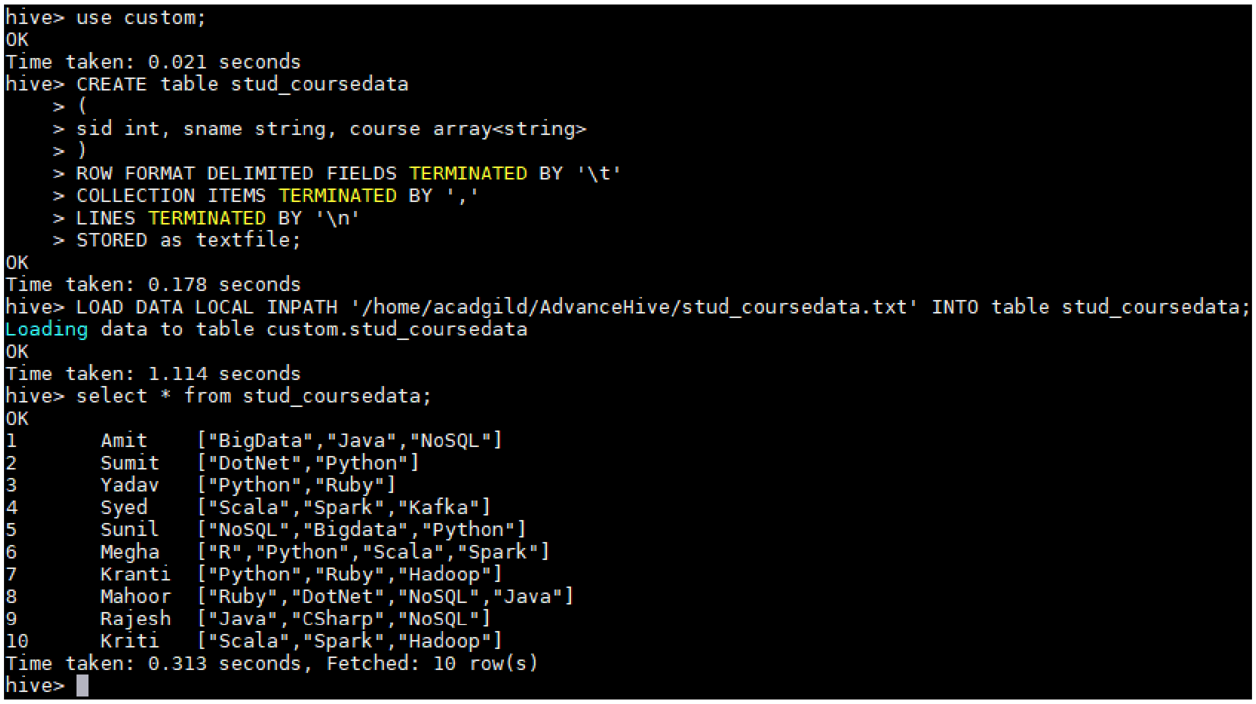
```
hive> CREATE table stud_coursedata
      (
      sid int, sname string, course array<string>
      )
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
      COLLECTION ITEMS TERMINATED BY ','
      LINES TERMINATED BY '\n'
      STORED as textfile;
```

– Load the data into the table created

```
hive> LOAD DATA LOCAL INPATH
      '/home/acadgild/AdvanceHive/stud_coursedata.txt' INTO table
      stud_coursedata;
```

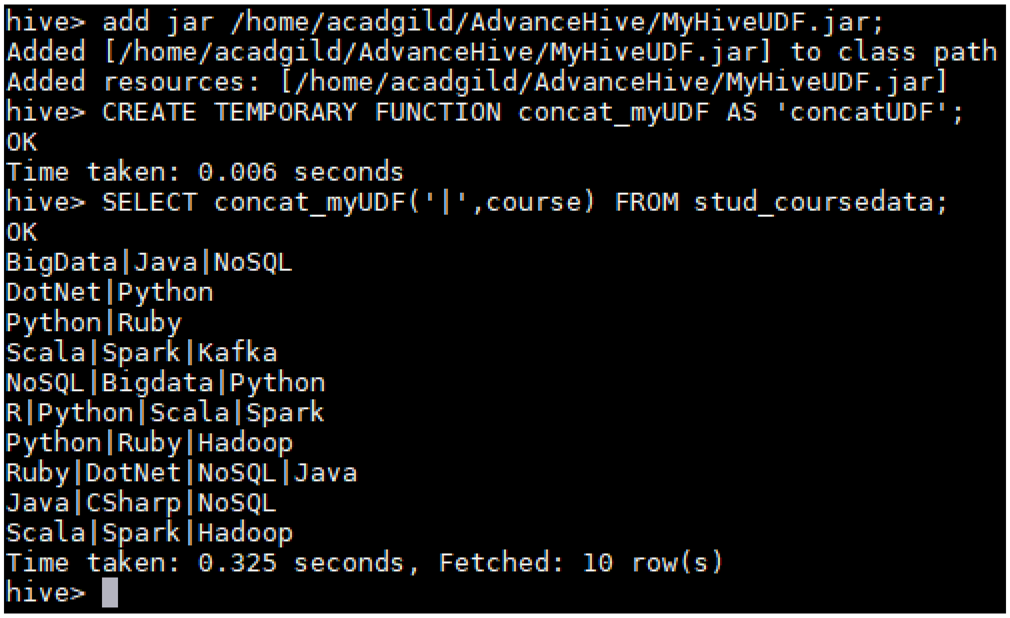
– To check the data loaded in the table, use query command

```
hive> select * from stud_coursedata;
```



```
hive> use custom;
OK
Time taken: 0.021 seconds
hive> CREATE table stud_coursedata
    > (
    > sid int, sname string, course array<string>
    > )
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > COLLECTION ITEMS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED as textfile;
OK
Time taken: 0.178 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadgild/AdvanceHive/stud_coursedata.txt' INTO table stud_coursedata;
Loading data to table custom.stud_coursedata
OK
Time taken: 1.114 seconds
hive> select * from stud_coursedata;
OK
1       Amit    ["BigData","Java","NoSQL"]
2       Sumit   ["DotNet","Python"]
3       Yadav   ["Python","Ruby"]
4       Syed    ["Scala","Spark","Kafka"]
5       Sunil   ["NoSQL","Bigdata","Python"]
6       Megha   ["R","Python","Scala","Spark"]
7       Kranti  ["Python","Ruby","Hadoop"]
8       Mahoor  ["Ruby","DotNet","NoSQL","Java"]
9       Rajesh  ["Java","CSharp","NoSQL"]
10      Kriti   ["Scala","Spark","Hadoop"]
Time taken: 0.313 seconds, Fetched: 10 row(s)
hive> 
```

- Java program with class ***concatUDF.java*** was created, which is provided along with this report, which was used to export a jar file ***MyHiveUDF.jar***
- This jar needs to be added to Hive, this can be done by using the command in hive
  hive> add jar /home/acadgild/AdvanceHive/MyHiveUDF.jar
- Create a temporary function concatUDF which would be used over the columns in the table
  hive> CREATE TEMPORARY FUNCTION concat_myUDF AS 'concatUDF';
- To display course using HIVE UDF 'concat_udf' using '|' separator
  hive> SELECT concat_myUDF('|',course) FROM stud_coursedata;



```
hive> add jar /home/acadgild/AdvanceHive/MyHiveUDF.jar;
Added [/home/acadgild/AdvanceHive/MyHiveUDF.jar] to class path
Added resources: [/home/acadgild/AdvanceHive/MyHiveUDF.jar]
hive> CREATE TEMPORARY FUNCTION concat_myUDF AS 'concatUDF';
OK
Time taken: 0.006 seconds
hive> SELECT concat_myUDF('|',course) FROM stud_coursedata;
OK
BigData|Java|NoSQL
DotNet|Python
Python|Ruby
Scala|Spark|Kafka
NoSQL|Bigdata|Python
R|Python|Scala|Spark
Python|Ruby|Hadoop
Ruby|DotNet|NoSQL|Java
Java|CSharp|NoSQL
Scala|Spark|Hadoop
Time taken: 0.325 seconds, Fetched: 10 row(s)
hive> 
```

Task 3:

Case Study link for Hive Transactions:

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.

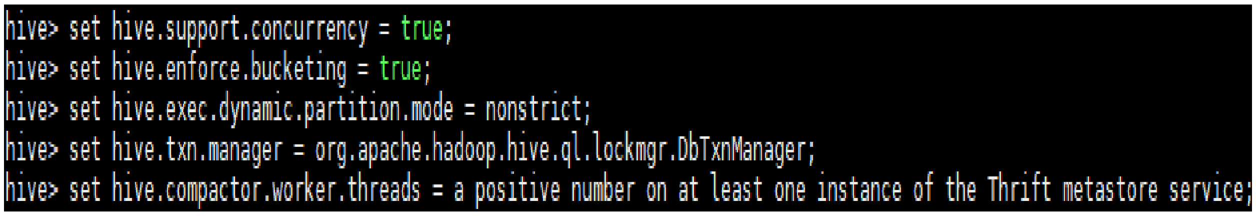
The different row-level transactions available in Hive are as follows:

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 need 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;

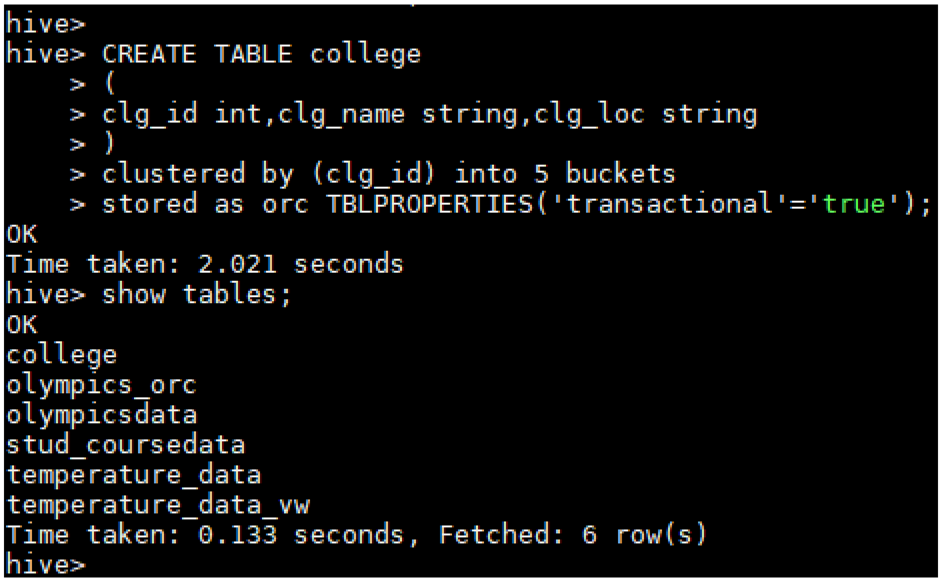


```
hive> set hive.support.concurrency = true;
hive> set hive.enforce.bucketing = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.worker.threads = a positive number on at least one instance of the Thrift metastore service;
```

Creating a Table That Supports Hive Transactions:

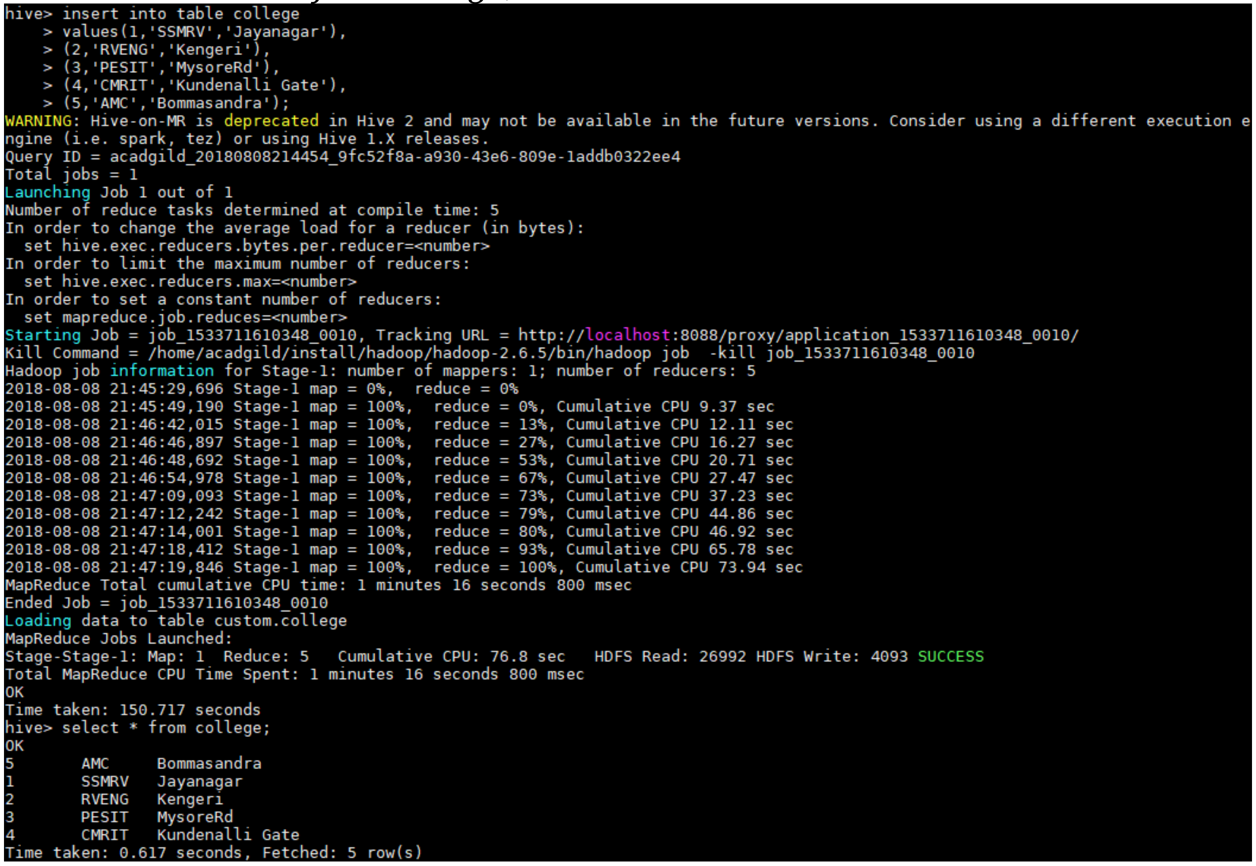
- The below query command will create a table with name ‘college’ and the columns present in the table are ‘clg_id, clg_name, clg_loc’.
- We are bucketing the table by ‘clg_id’ and the table format is ‘orc’, also we are enabling the transactions in the table by specifying it inside the TBLPROPERTIES as ‘transactional’=’true’

  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');



```
hive>
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: 2.021 seconds
hive> show tables;
OK
college
olympics_orc
olympicsdata
stud_coursedata
temperature_data
temperature_data_vw
Time taken: 0.133 seconds, Fetched: 6 row(s)
hive>
```

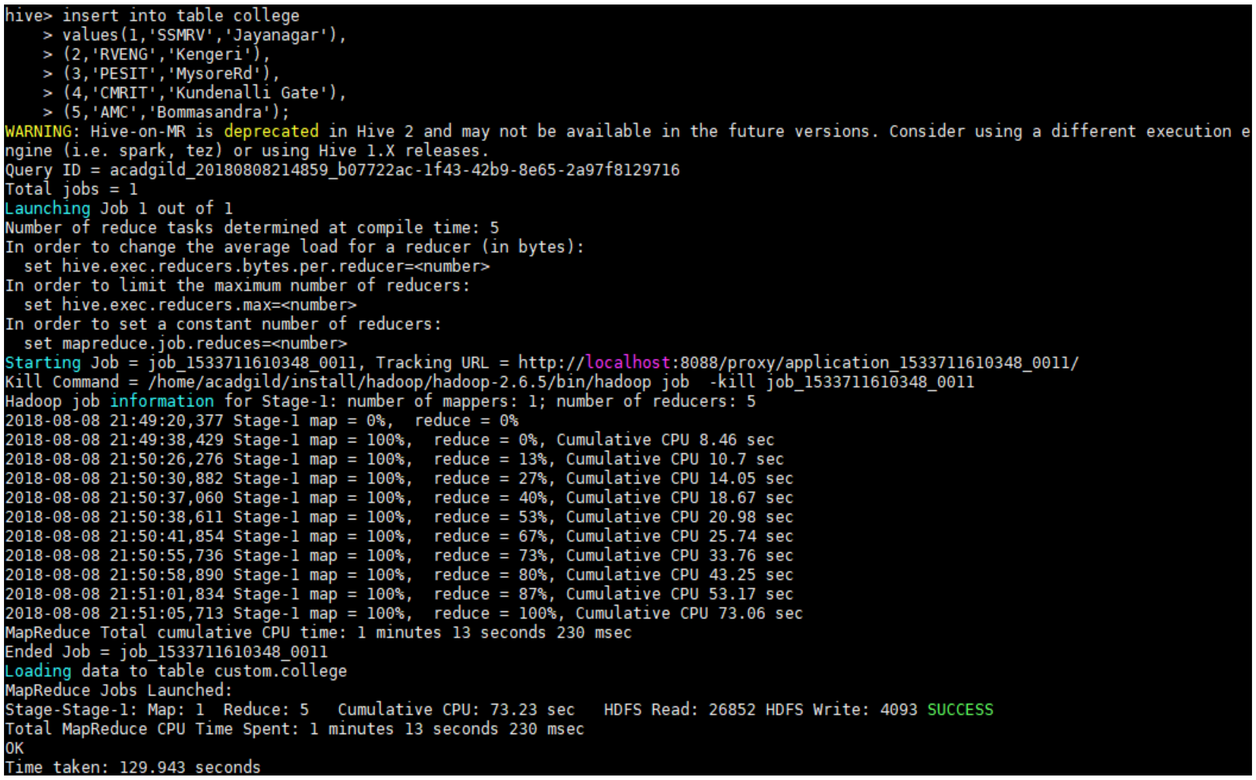
– The query command below is used to insert row-wise data into the Hive table. Here, each row is separated by '( )' brackets.
  hive> insert into table college values (1,'SSMRV','Jayanagar'),(2,'RVENG','Kengeri'),(3,'PESIT','MysoreRd'),(4,'CMRIT','Kundenalli Gate'),(5,'AMC','Bommasandra');
– The contents of the table can be viewed using the command
  *hive> select * from college;*



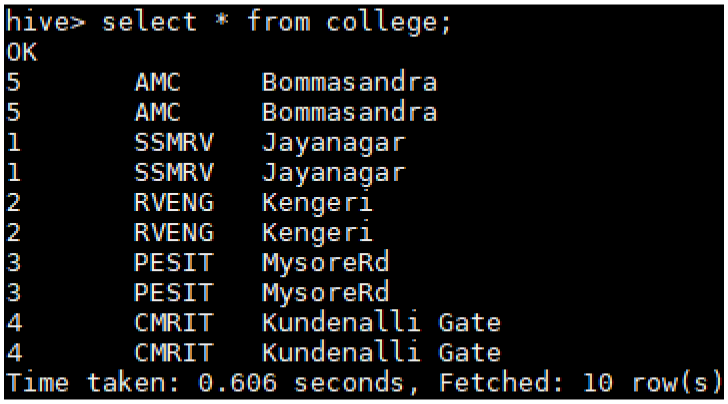
```
hive> insert into table college
    > values(1,'SSMRV','Jayanagar'),
    > (2,'RVENG','Kengeri'),
    > (3,'PESIT','MysoreRd'),
    > (4,'CMRIT','Kundenalli Gate'),
    > (5,'AMC','Bommasandra');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808214454_9fc52f8a-a930-43e6-809e-1addb0322ee4
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_1533711610348_0010, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0010/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-08-08 21:45:29,696 Stage-1 map = 0%,  reduce = 0%
2018-08-08 21:45:49,190 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 9.37 sec
2018-08-08 21:46:42,015 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 12.11 sec
2018-08-08 21:46:46,897 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 16.27 sec
2018-08-08 21:46:48,692 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 20.71 sec
2018-08-08 21:46:54,978 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 27.47 sec
2018-08-08 21:47:09,093 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 37.23 sec
2018-08-08 21:47:12,242 Stage-1 map = 100%,  reduce = 79%, Cumulative CPU 44.86 sec
2018-08-08 21:47:14,001 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 46.92 sec
2018-08-08 21:47:18,412 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 65.78 sec
2018-08-08 21:47:19,846 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 73.94 sec
MapReduce Total cumulative CPU time: 1 minutes 16 seconds 800 msec
Ended Job = job_1533711610348_0010
Loading data to table custom.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 76.8 sec   HDFS Read: 26992 HDFS Write: 4093 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 16 seconds 800 msec
OK
Time taken: 150.717 seconds
hive> select * from college;
OK
5       AMC     Bommasandra
1       SSMRV   Jayanagar
2       RVENG   Kengeri
3       PESIT   MysoreRd
4       CMRIT   Kundenalli Gate
Time taken: 0.617 seconds, Fetched: 5 row(s)
```

– 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 screenshots:



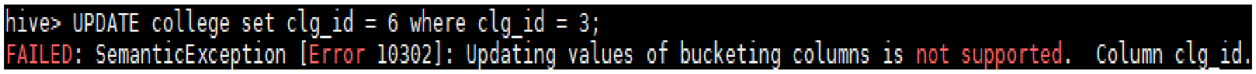
```
hive> insert into table college
    > values(1,'SSMRV','Jayanagar'),
    > (2,'RVENG','Kengeri'),
    > (3,'PESIT','MysoreRd'),
    > (4,'CMRIT','Kundenalli Gate'),
    > (5,'AMC','Bommasandra');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808214859_b07722ac-1f43-42b9-8e65-2a97f8129716
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_1533711610348_0011, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-08-08 21:49:20,377 Stage-1 map = 0%,  reduce = 0%
2018-08-08 21:49:38,429 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.46 sec
2018-08-08 21:50:26,276 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 10.7 sec
2018-08-08 21:50:30,882 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 14.05 sec
2018-08-08 21:50:37,060 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 18.67 sec
2018-08-08 21:50:38,611 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 20.98 sec
2018-08-08 21:50:41,854 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 25.74 sec
2018-08-08 21:50:55,736 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 33.76 sec
2018-08-08 21:50:58,890 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 43.25 sec
2018-08-08 21:51:01,834 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 53.17 sec
2018-08-08 21:51:05,713 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 73.06 sec
MapReduce Total cumulative CPU time: 1 minutes 13 seconds 230 msec
Ended Job = job_1533711610348_0011
Loading data to table custom.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 73.23 sec   HDFS Read: 26852 HDFS Write: 4093 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 13 seconds 230 msec
OK
Time taken: 129.943 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: 0.606 seconds, Fetched: 10 row(s)
```

Updating the Data in Hive Table

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



```
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.
```

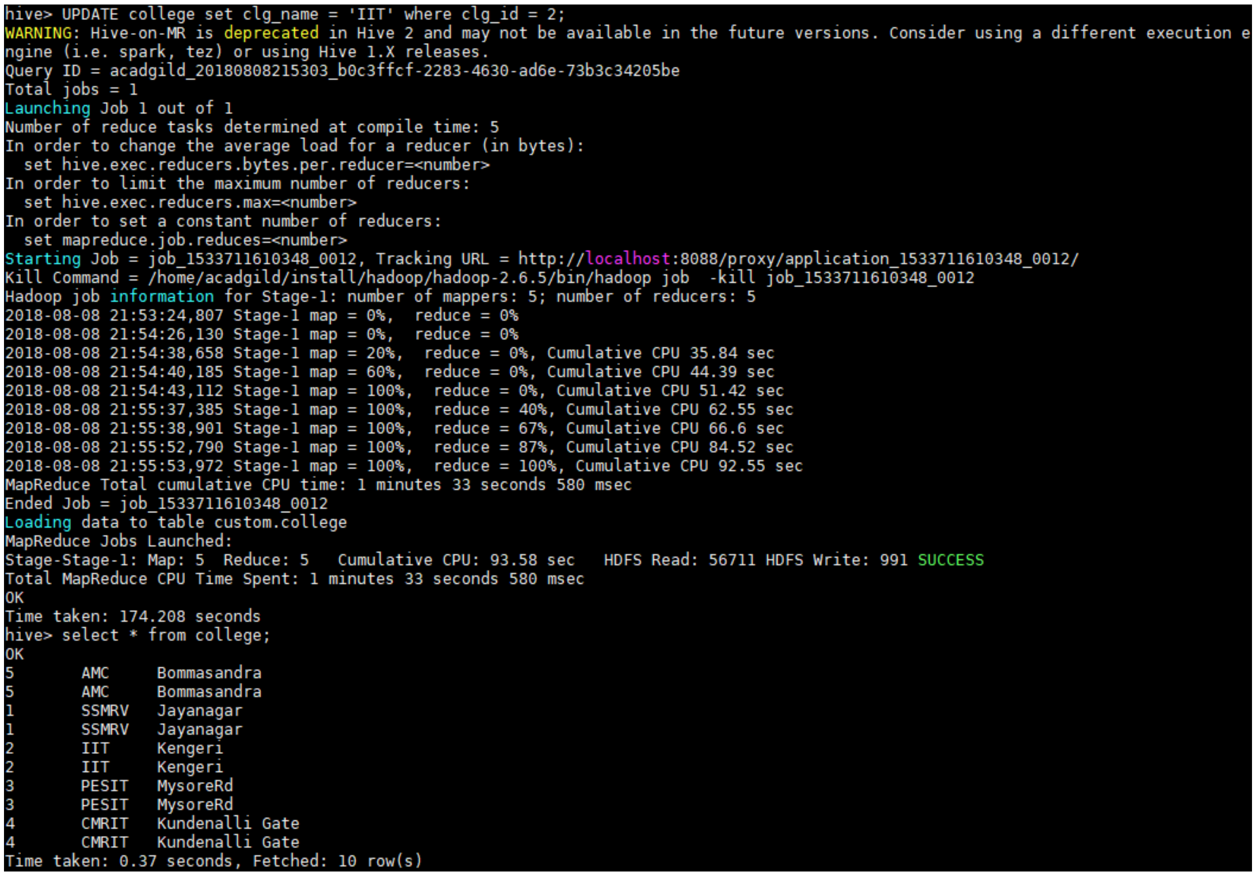
– We can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed.

– 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

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

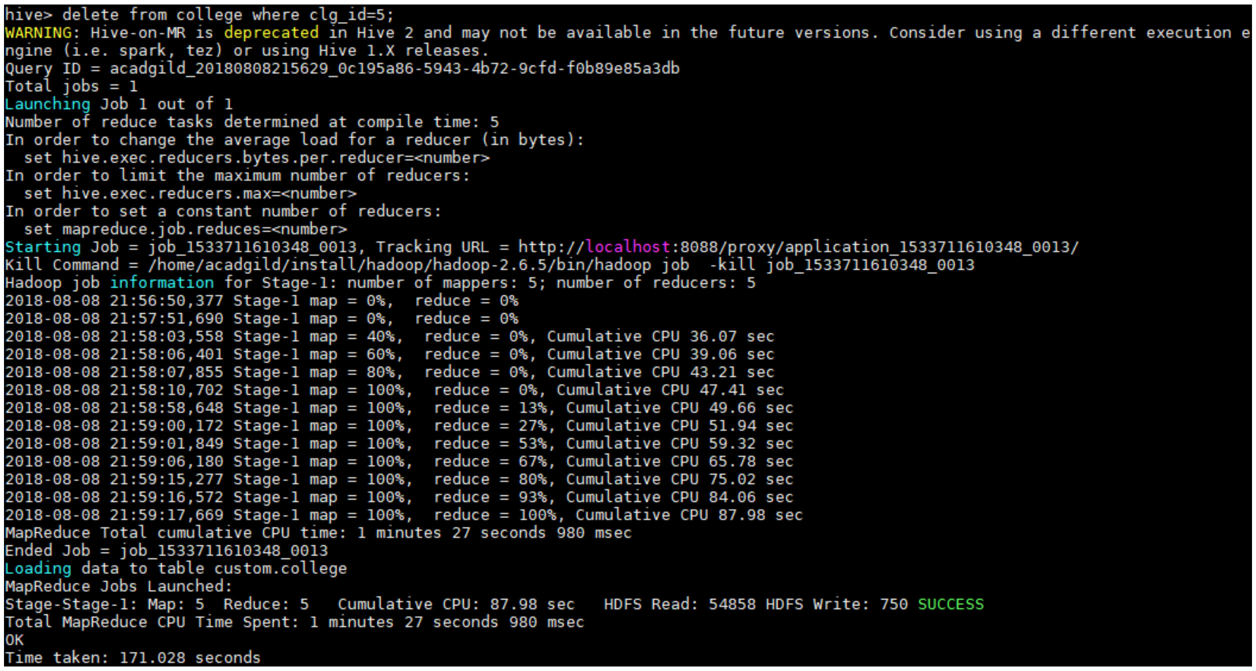
– The updated data can be checked using the query command
  *hive> select * from college;*



```
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 e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808215303_b0c3ffcf-2283-4630-ad6e-73b3c34205be
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_1533711610348_0012, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0012/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0012
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-08-08 21:53:24,807 Stage-1 map = 0%,  reduce = 0%
2018-08-08 21:54:26,130 Stage-1 map = 0%,  reduce = 0%
2018-08-08 21:54:38,658 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 35.84 sec
2018-08-08 21:54:40,185 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 44.39 sec
2018-08-08 21:54:43,112 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 51.42 sec
2018-08-08 21:55:37,385 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 62.55 sec
2018-08-08 21:55:38,901 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 66.6 sec
2018-08-08 21:55:52,790 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 84.52 sec
2018-08-08 21:55:53,972 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 92.55 sec
MapReduce Total cumulative CPU time: 1 minutes 33 seconds 580 msec
Ended Job = job_1533711610348_0012
Loading data to table custom.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 93.58 sec   HDFS Read: 56711 HDFS Write: 991 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 33 seconds 580 msec
OK
Time taken: 174.208 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: 0.37 seconds, Fetched: 10 row(s)
```

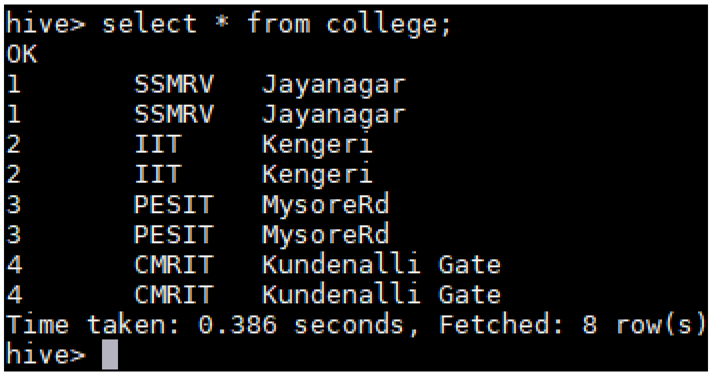
Deleting a Row from Hive Table:

hive> 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 e
ngine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180808215629_0c195a86-5943-4b72-9cfd-f0b89e85a3db
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_1533711610348_0013, Tracking URL = http://localhost:8088/proxy/application_1533711610348_0013/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533711610348_0013
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-08-08 21:56:50,377 Stage-1 map = 0%,  reduce = 0%
2018-08-08 21:57:51,690 Stage-1 map = 0%,  reduce = 0%
2018-08-08 21:58:03,558 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 36.07 sec
2018-08-08 21:58:06,401 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 39.06 sec
2018-08-08 21:58:07,855 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 43.21 sec
2018-08-08 21:58:10,702 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 47.41 sec
2018-08-08 21:58:58,648 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 49.66 sec
2018-08-08 21:59:00,172 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 51.94 sec
2018-08-08 21:59:01,849 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 59.32 sec
2018-08-08 21:59:06,180 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 65.78 sec
2018-08-08 21:59:15,277 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 75.02 sec
2018-08-08 21:59:16,572 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 84.06 sec
2018-08-08 21:59:17,669 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 87.98 sec
MapReduce Total cumulative CPU time: 1 minutes 27 seconds 980 msec
Ended Job = job_1533711610348_0013
Loading data to table custom.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 87.98 sec   HDFS Read: 54858 HDFS Write: 750 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 27 seconds 980 msec
OK
Time taken: 171.028 seconds
```

- We have now successfully deleted a row from the Hive table.
- Checked the same using the query command
  *hive> select * from college;*



```
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: 0.386 seconds, Fetched: 8 row(s)
hive>
```

- We can see that there is no row with clg_id =5.
- This is how the transactions or row-wise operations are performed in Hive.