Big Data Hadoop and Spark Development

Session 9: Advance Hive Assignment 1

Big Data Hadoop and Spark Development

Assignment 1 – You must perform the given tasks.

HIVE Queries create a Database and a table providing the Input Dataset.

CREATE DATABASE olympic;

USE olympic;

CREATE TABLE olympic_data

(Athlete STRING,Age INT,Country STRING,Year INT,Closing_Date STRING,Sport STRING,Gold_Medals INT,Silver_Medals INT,Bronze_MedalsINT,Total_Medals INT)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

LOAD DATA LOCAL INPATH '/home/acadgild/hadoop/olympic_data.json'

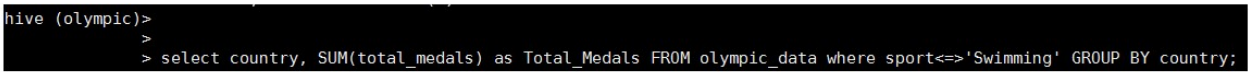
INTO TABLE olympic_data;

Task1

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

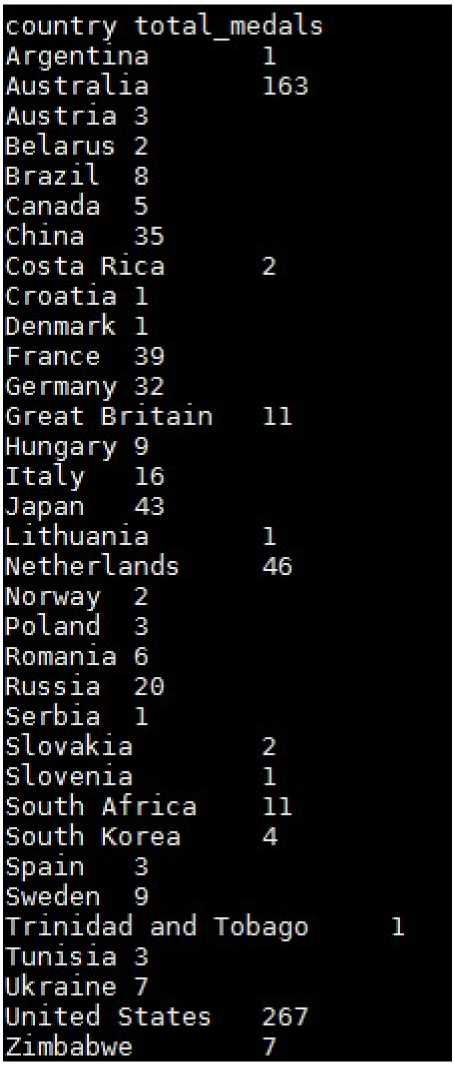
HIVE Query

SELECT country, SUM(total_medals) as Total_Medals FROM olympic_dataWHERE sport<=>'Swimming' GROUP BY country;



```
hive (olympic)>
              >
              > select country, SUM(total_medals) as Total_Medals FROM olympic_data where sport<=>'Swimming' GROUP BY country;
```

Expected Output



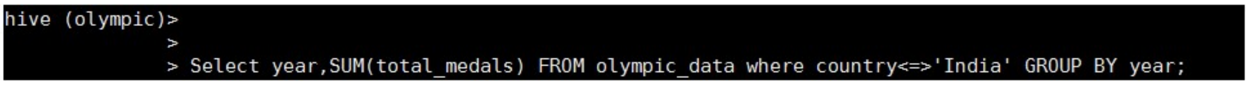
```
country total_medals
Argentina       1
Australia       163
Austria 3
Belarus 2
Brazil  8
Canada  5
China   35
Costa Rica      2
Croatia 1
Denmark 1
France  39
Germany 32
Great Britain   11
Hungary 9
Italy   16
Japan   43
Lithuania       1
Netherlands     46
Norway  2
Poland  3
Romania 6
Russia  20
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
```

TASK2

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

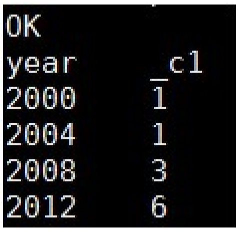
HIVE Query

SELECT year,SUM(total_medals) FROM olympic_dataWhere country<=>'India' GROUP BY year;



```
hive (olympic)>
              >
              > Select year,SUM(total_medals) FROM olympic_data where country<=>'India' GROUP BY year;
```

Expected Output



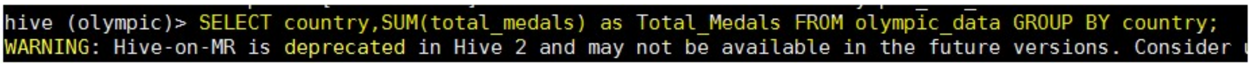
```
OK
year    _c1
2000    1
2004    1
2008    3
2012    6
```

TASK3

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

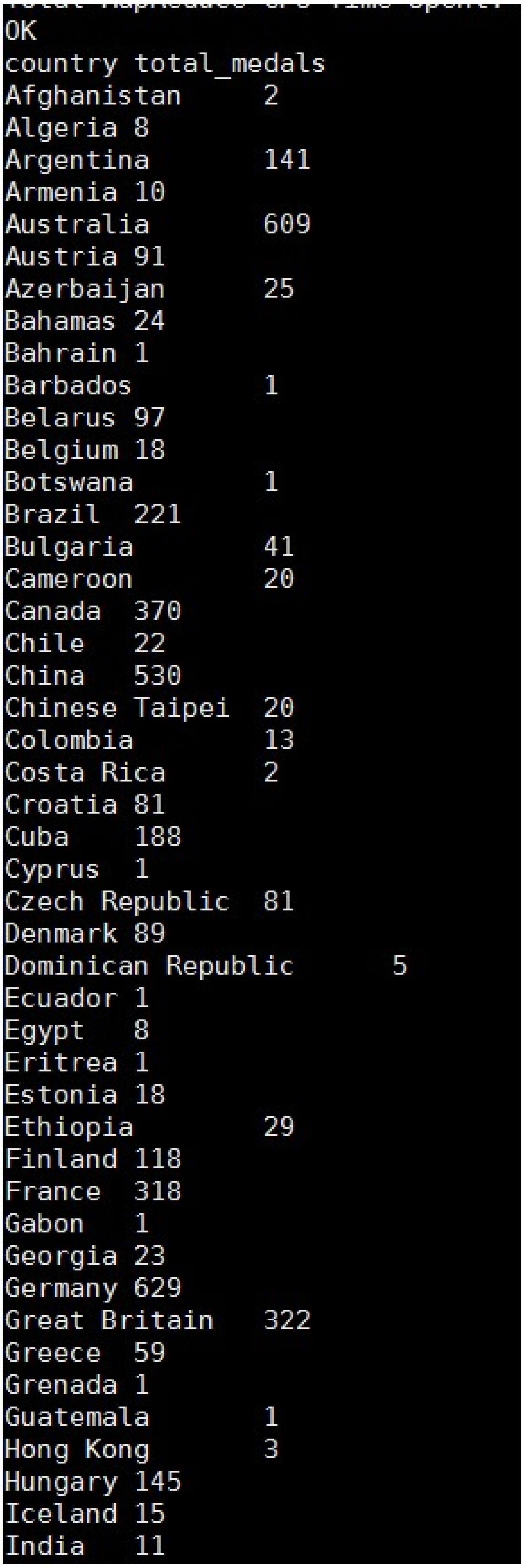
HIVE Query

SELECT country,SUM(total_medals) as Total_Medals FROM olympic_data GROUP BY country;

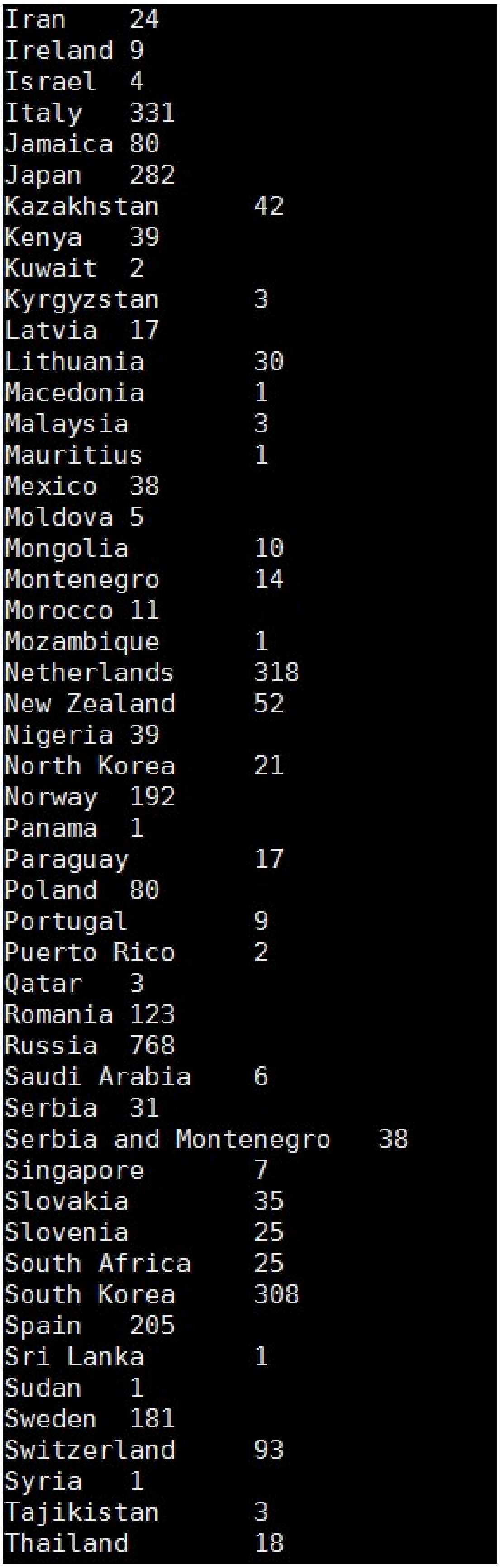


```
hive (olympic)> SELECT country,SUM(total_medals) as Total_Medals FROM olympic_data GROUP BY country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
```

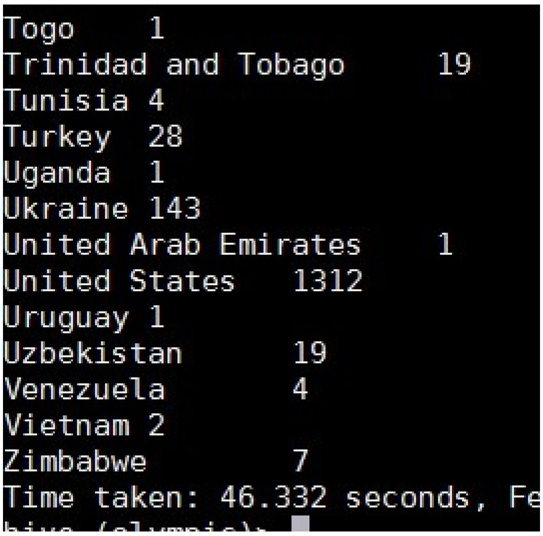
Expected Output



```
Total MapReduce CPU Time Spent:
OK
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
Gabon   1
Georgia 23
Germany 629
Great Britain   322
Greece  59
Grenada 1
Guatemala       1
Hong Kong       3
Hungary 145
Iceland 15
India   11
```



```
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
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
```



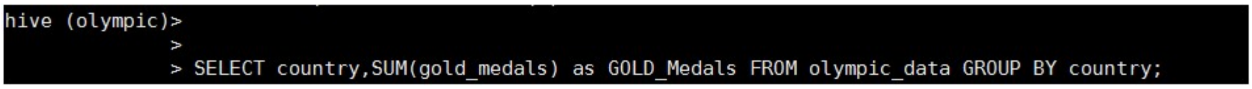
```
Togo    1
Trinidad and Tobago     19
Tunisia 4
Turkey  28
Uganda  1
Ukraine 143
United Arab Emirates    1
United States   1312
Uruguay 1
Uzbekistan      19
Venezuela       4
Vietnam 2
Zimbabwe        7
Time taken: 46.332 seconds, Fe
hive (olympic)>
```

Task4

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

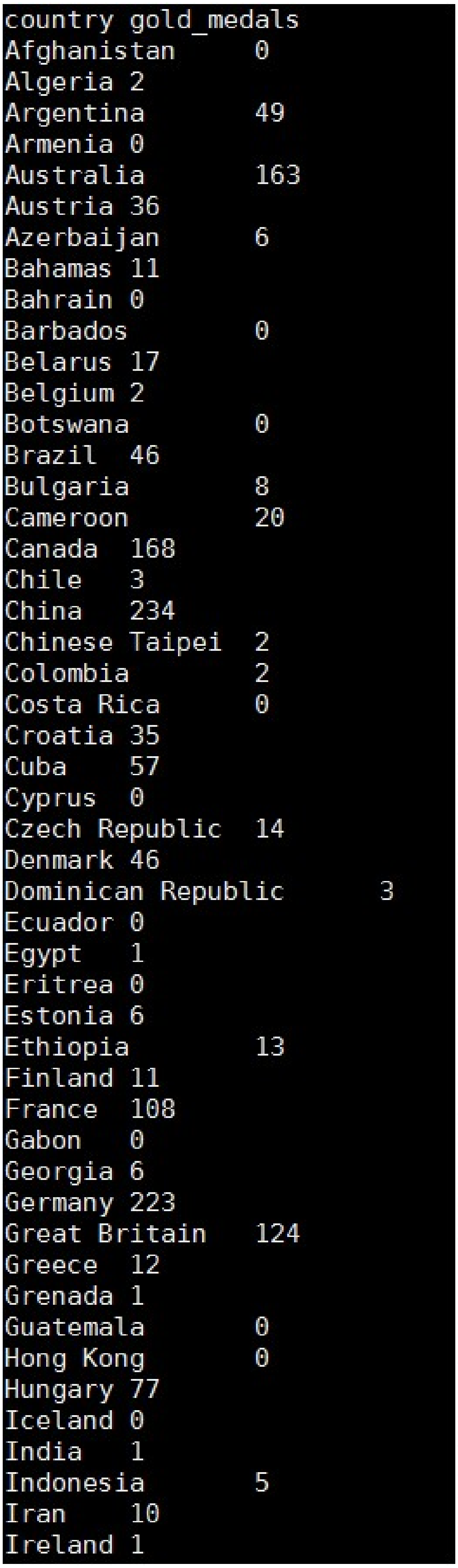
HIVE Query

SELECT country,SUM(gold_medals) as GOLD_Medals FROM olympic_data GROUP BY country;

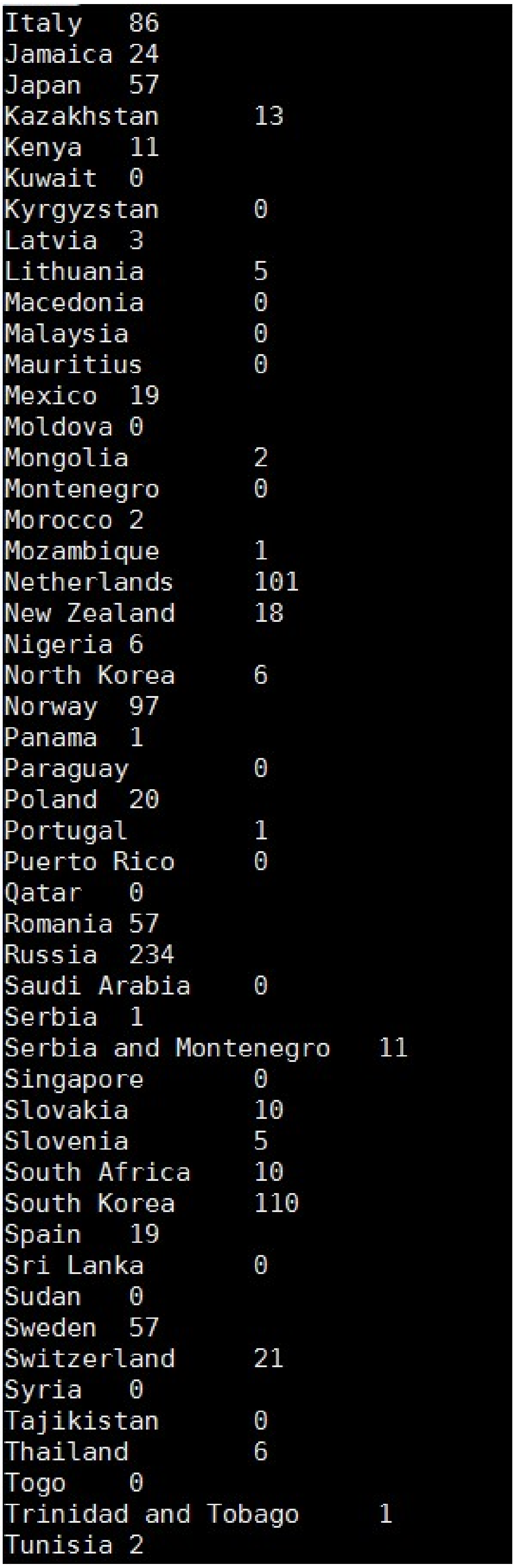


```
hive (olympic)>
              >
              > SELECT country,SUM(gold_medals) as GOLD_Medals FROM olympic_data GROUP BY country;
```

Expect Output



```
country gold_medals
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
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
```



```
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
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
```

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.

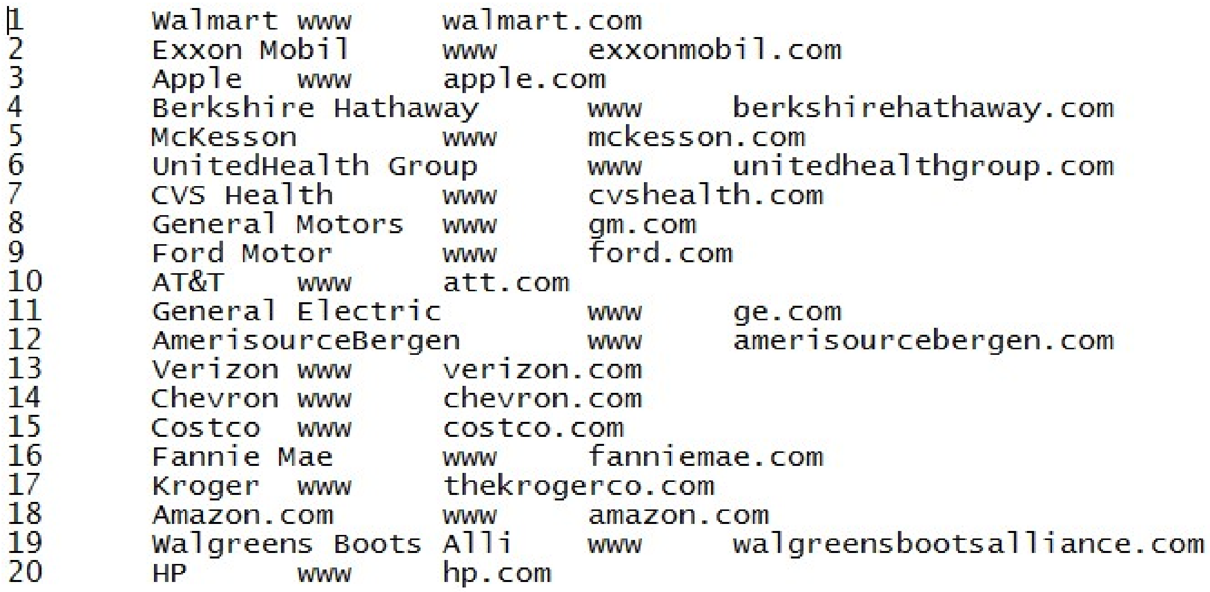
We have fortune 20 companies list and its company website URL, but the ‘www’ and the remaining domain are separated. In our output we try to achieve the output as below,

1 walmart www.walmart.com

Dataset

The below data contains, the column name as,

Rank, company_name, website, protocal.



```
1       Walmart www     walmart.com
2       Exxon Mobil     www     exxonmobil.com
3       Apple   www     apple.com
4       Berkshire Hathaway      www     berkshirehathaway.com
5       McKesson        www     mckesson.com
6       UnitedHealth Group      www     unitedhealthgroup.com
7       CVS Health      www     cvshealth.com
8       General Motors  www     gm.com
9       Ford Motor      www     ford.com
10      AT&T    www     att.com
11      General Electric        www     ge.com
12      AmerisourceBergen       www     amerisourcebergen.com
13      Verizon www     verizon.com
14      Chevron www     chevron.com
15      Costco  www     costco.com
16      Fannie Mae      www     fanniemae.com
17      Kroger  www     thekrogerco.com
18      Amazon.com      www     amazon.com
19      Walgreens Boots Alli    www     walgreensbootsalliance.com
20      HP      www     hp.com
```

Prerequisites

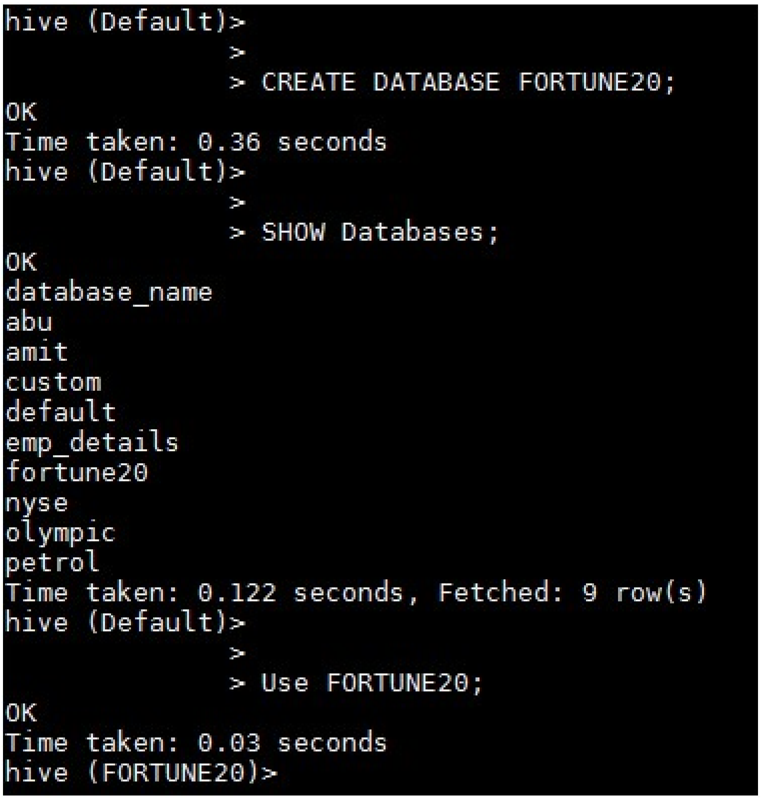
Create Database and Table

Create Database FORTUNE20

HIVE QL

CREATE DATABASE FORTUNE20

Use FORTUNE20;



```
hive (Default)>
              >
              > CREATE DATABASE FORTUNE20;
OK
Time taken: 0.36 seconds
hive (Default)>
              >
              > SHOW Databases;
OK
database_name
abu
amit
custom
default
emp_details
fortune20
nyse
olympic
petrol
Time taken: 0.122 seconds, Fetched: 9 row(s)
hive (Default)>
              >
              > Use FORTUNE20;
OK
Time taken: 0.03 seconds
hive (FORTUNE20)>
```

Create Table Fortune_company

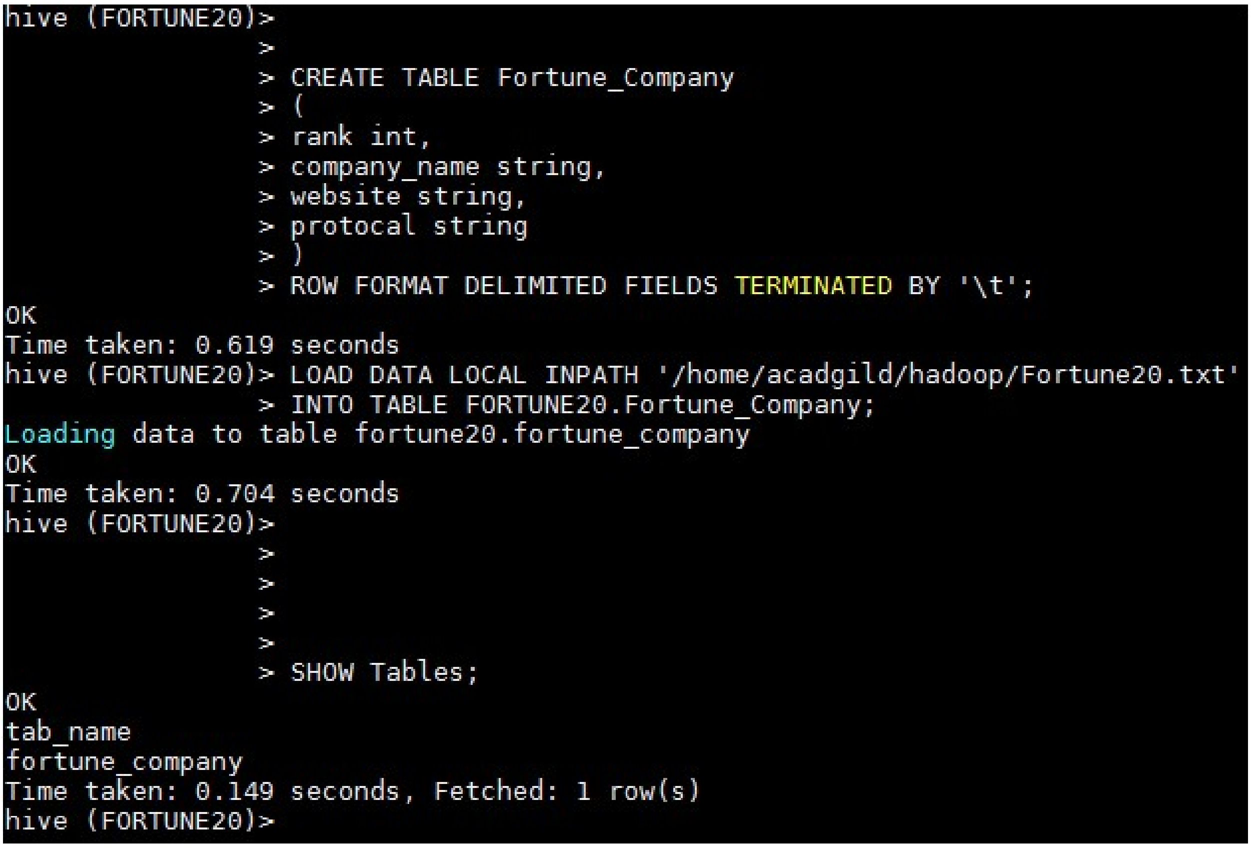
HIVE QL

CREATE TABLE fortune_company(rank int, company_namestring,website string, protocal string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

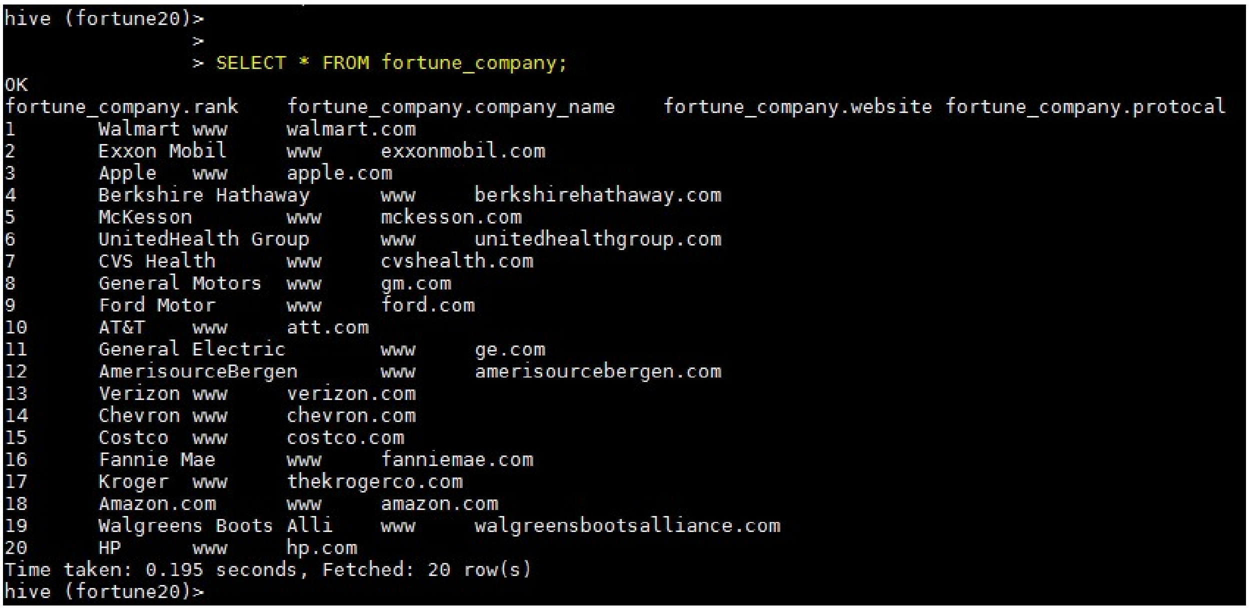
LOAD DATA LOCAL INPATH '/home/acadgild/hadoop/fortune20.txt'

INTO TABLE fortune20.fortune_company;



```
hive (FORTUNE20)>
                > 
                > CREATE TABLE Fortune_Company
                > (
                > rank int,
                > company_name string,
                > website string,
                > protocal string
                > )
                > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
OK
Time taken: 0.619 seconds
hive (FORTUNE20)> LOAD DATA LOCAL INPATH '/home/acadgild/hadoop/Fortune20.txt'
                > INTO TABLE FORTUNE20.Fortune_Company;
Loading data to table fortune20.fortune_company
OK
Time taken: 0.704 seconds
hive (FORTUNE20)>
                > 
                > 
                > 
                > 
                > SHOW Tables;
OK
tab_name
fortune_company
Time taken: 0.149 seconds, Fetched: 1 row(s)
hive (FORTUNE20)>
```

Viewing the data in the table fortune_company,
SELECT * FROM fortune_company;



```
hive (fortune20)>
                > 
                > SELECT * FROM fortune_company;
OK
fortune_company.rank    fortune_company.company_name    fortune_company.website fortune_company.protocal
1       Walmart www     walmart.com
2       Exxon Mobil     www     exxonmobil.com
3       Apple   www     apple.com
4       Berkshire Hathaway      www     berkshirehathaway.com
5       McKesson        www     mckesson.com
6       UnitedHealth Group      www     unitedhealthgroup.com
7       CVS Health      www     cvshealth.com
8       General Motors  www     gm.com
9       Ford Motor      www     ford.com
10      AT&T    www     att.com
11      General Electric        www     ge.com
12      AmerisourceBergen       www     amerisourcebergen.com
13      Verizon www     verizon.com
14      Chevron www     chevron.com
15      Costco  www     costco.com
16      Fannie Mae      www     fanniemae.com
17      Kroger  www     thekrogerco.com
18      Amazon.com      www     amazon.com
19      Walgreens Boots Alli    www     walgreensbootsalliance.com
20      HP      www     hp.com
Time taken: 0.195 seconds, Fetched: 20 row(s)
hive (fortune20)>
```

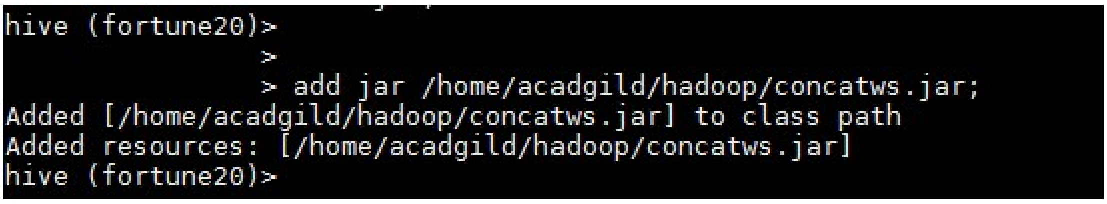
HIVE UDF Java code

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


Pulic class CONCAT_WS extend UDF {
public String evaluate(String sep, ArrayList<String> arr)
{
StringBuffer strBuffer;

if (arr=null)
{
return null;
}
strBuffer = new StringBuffer();
strBuffer.append(arr.get(0));
for(int i =1; i <arr.size(); i ++)
{
strBuffer.append(sep);
strBuffer.append(arr.get());
}
return strBuffer.tostring();
```

} After that we are adding JAR created from the JAVA class which is defining the UDF using below syntax-

HIVE UDF CONCAT_WS function

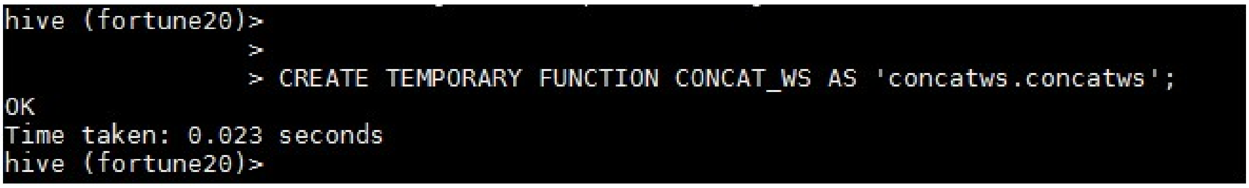
add jar /home/acadgild/hadoop/concatws.jar;



```
hive (fortune20)>
                 >
                 > add jar /home/acadgild/hadoop/concatws.jar;
Added [/home/acadgild/hadoop/concatws.jar] to class path
Added resources: [/home/acadgild/hadoop/concatws.jar]
hive (fortune20)>
```

After that we are creating a temporary function “CONCAT_WS” using below syntax-

CREATE TEMPORARY FUNCTION CONCAT_WS AS 'concatws.concatws';

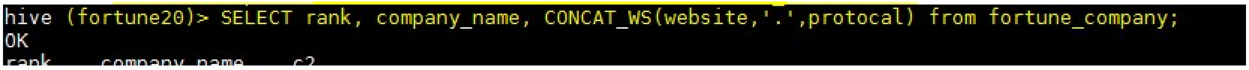


```
hive (fortune20)>
                 >
                 > CREATE TEMPORARY FUNCTION CONCAT_WS AS 'concatws.concatws';
OK
Time taken: 0.023 seconds
hive (fortune20)>
```

After that we run below query to take one column (company_name) input as String and another array(website,’.’,protocal) as Array of Strings and concatenate them,

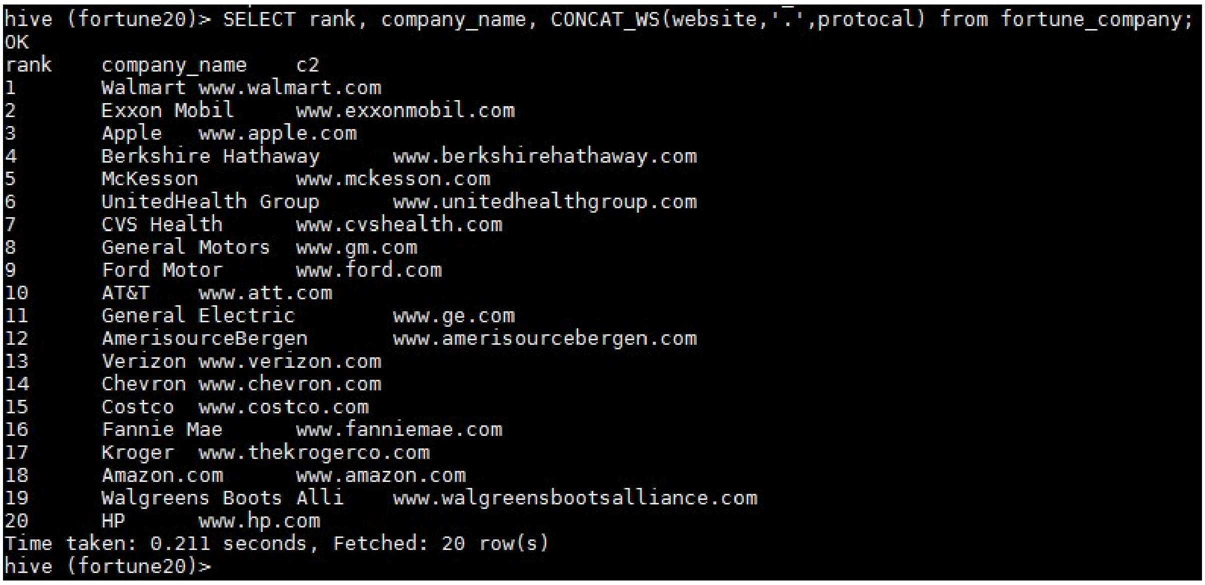
HIVE QL

SELECT rank, company_name, CONCAT_WS(website,'.',protocal) from fortune_company;SELECT rank, company_name, CONCAT_WS(website,'.',protocal) from fortune_company;



```
hive (fortune20)> SELECT rank, company_name, CONCAT_WS(website,'.',protocal) from fortune_company;
OK
rank    company_name    c2
```

Required Output



```
hive (fortune20)> SELECT rank, company_name, CONCAT_WS(website,'.',protocal) from fortune_company;
OK
rank    company_name    c2
1       Walmart www.walmart.com
2       Exxon Mobil     www.exxonmobil.com
3       Apple   www.apple.com
4       Berkshire Hathaway      www.berkshirehathaway.com
5       McKesson        www.mckesson.com
6       UnitedHealth Group      www.unitedhealthgroup.com
7       CVS Health      www.cvshealth.com
8       General Motors  www.gm.com
9       Ford Motor      www.ford.com
10      AT&T    www.att.com
11      General Electric        www.ge.com
12      AmerisourceBergen       www.amerisourcebergen.com
13      Verizon www.verizon.com
14      Chevron www.chevron.com
15      Costco  www.costco.com
16      Fannie Mae      www.fanniemae.com
17      Kroger  www.thekrogerco.com
18      Amazon.com      www.amazon.com
19      Walgreens Boots Alli    www.walgreensbootsalliance.com
20      HP      www.hp.com
Time taken: 0.211 seconds, Fetched: 20 row(s)
hive (fortune20)>
```

Task 3

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

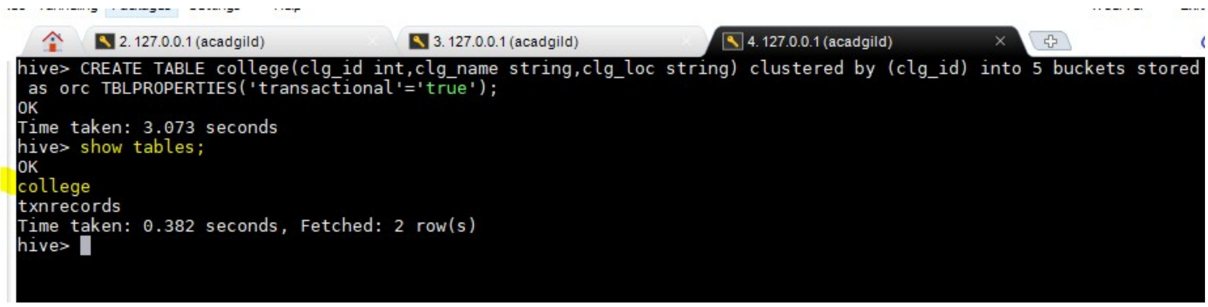
Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

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

1. Insert
2. Delete
3. Update

Creating a Table That Supports Hive Transactions

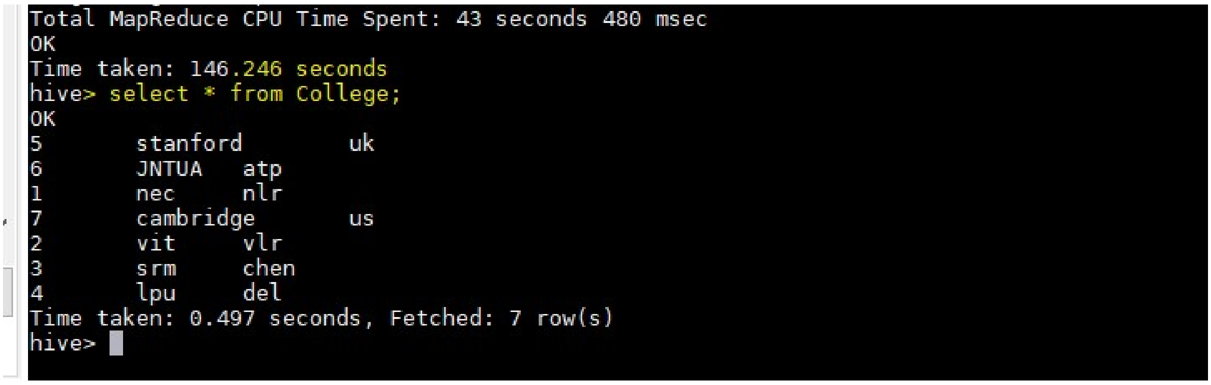
CREATE TABLE college(clg_id int,clg_name string,clg_loc string) clustered by (clg_id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');



```
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: 3.073 seconds
hive> show tables;
OK
college
txnrecords
Time taken: 0.382 seconds, Fetched: 2 row(s)
hive>
```

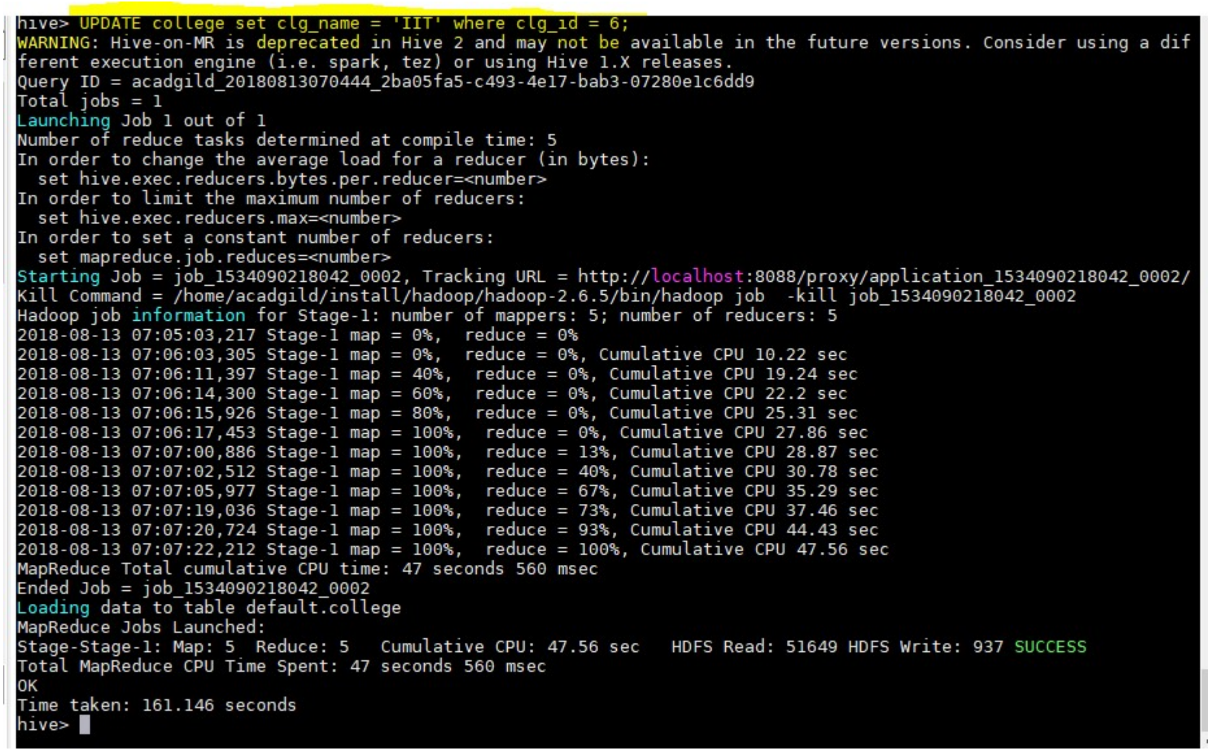
Inserting Data into a Hive Table

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');

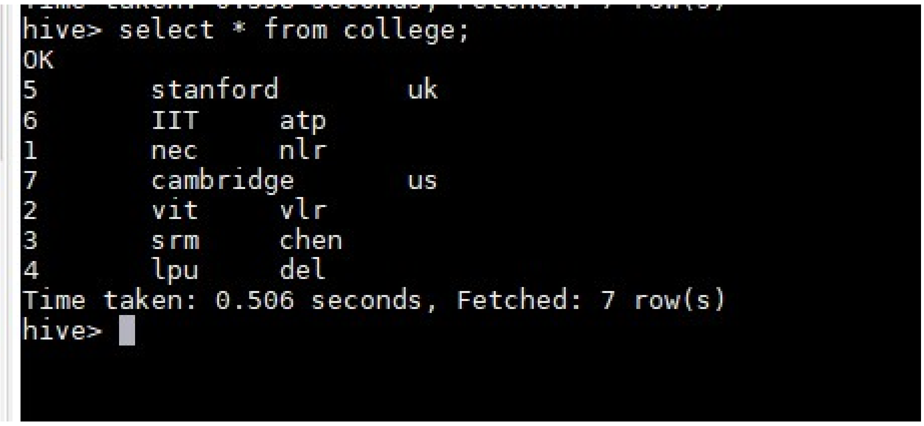


```
Total MapReduce CPU Time Spent: 43 seconds 480 msec
OK
Time taken: 146.246 seconds
hive> select * from College;
OK
5       stanford        uk
6       JNTUA   atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 0.497 seconds, Fetched: 7 row(s)
hive>
```

Updating the Data in Hive Table

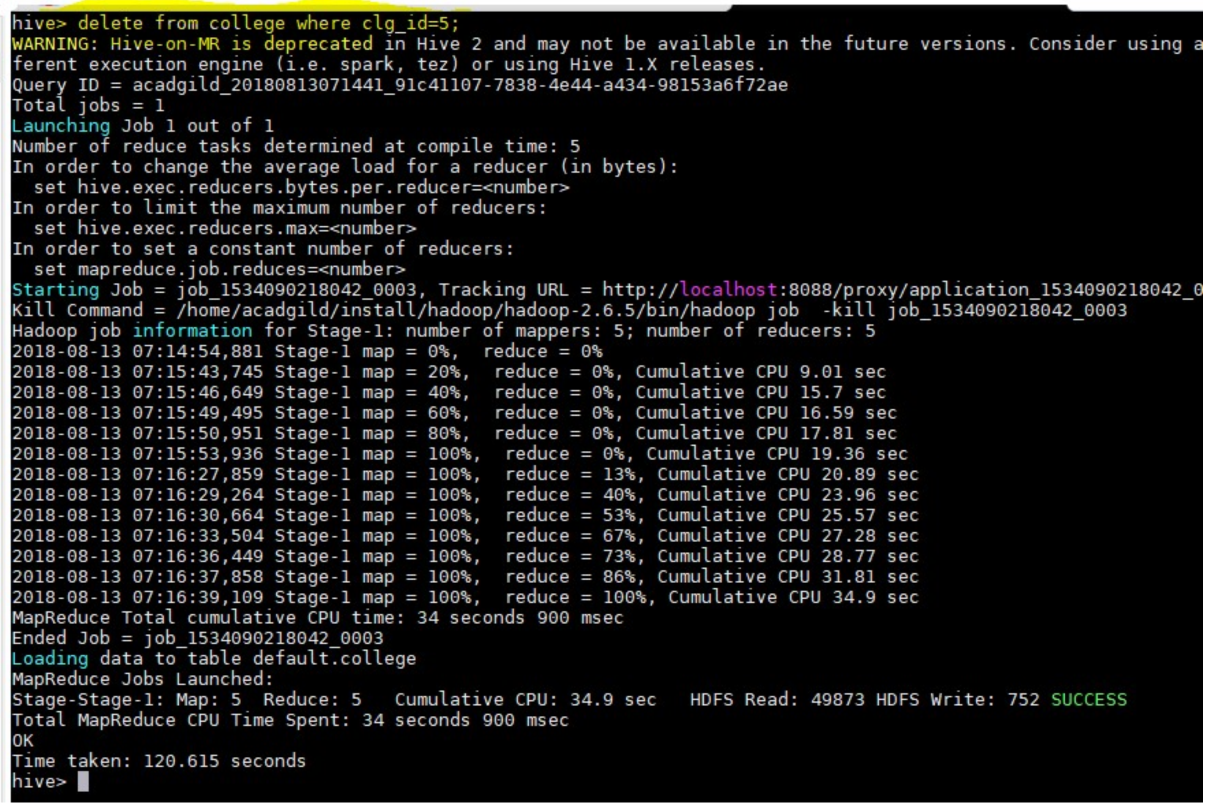


```
hive> UPDATE college set clg_name = 'IIT' where clg_id = 6;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a dif
ferent execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180813070444_2ba05fa5-c493-4e17-bab3-07280e1c6dd9
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_1534090218042_0002, Tracking URL = http://localhost:8088/proxy/application_1534090218042_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1534090218042_0002
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-08-13 07:05:03,217 Stage-1 map = 0%,  reduce = 0%
2018-08-13 07:06:03,305 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 10.22 sec
2018-08-13 07:06:11,397 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 19.24 sec
2018-08-13 07:06:14,300 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 22.2 sec
2018-08-13 07:06:15,926 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 25.31 sec
2018-08-13 07:06:17,453 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 27.86 sec
2018-08-13 07:07:00,886 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 28.87 sec
2018-08-13 07:07:02,512 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 30.78 sec
2018-08-13 07:07:05,977 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 35.29 sec
2018-08-13 07:07:19,036 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 37.46 sec
2018-08-13 07:07:20,724 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 44.43 sec
2018-08-13 07:07:22,212 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 47.56 sec
MapReduce Total cumulative CPU time: 47 seconds 560 msec
Ended Job = job_1534090218042_0002
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 47.56 sec   HDFS Read: 51649 HDFS Write: 937 SUCCESS
Total MapReduce CPU Time Spent: 47 seconds 560 msec
OK
Time taken: 161.146 seconds
hive>
```

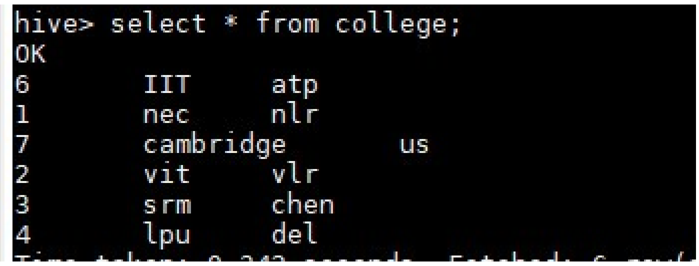


```
hive> select * from college;
OK
5       stanford        uk
6       IIT     atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 0.506 seconds, Fetched: 7 row(s)
hive>
```

Deleting a Row from Hive 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
ferent execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180813071441_91c41107-7838-4e44-a434-98153a6f72ae
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_1534090218042_0003, Tracking URL = http://localhost:8088/proxy/application_1534090218042_0
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1534090218042_0003
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-08-13 07:14:54,881 Stage-1 map = 0%,  reduce = 0%
2018-08-13 07:15:43,745 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 9.01 sec
2018-08-13 07:15:46,649 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 15.7 sec
2018-08-13 07:15:49,495 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 16.59 sec
2018-08-13 07:15:50,951 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 17.81 sec
2018-08-13 07:15:53,936 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 19.36 sec
2018-08-13 07:16:27,859 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 20.89 sec
2018-08-13 07:16:29,264 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 23.96 sec
2018-08-13 07:16:30,664 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 25.57 sec
2018-08-13 07:16:33,504 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 27.28 sec
2018-08-13 07:16:36,449 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 28.77 sec
2018-08-13 07:16:37,858 Stage-1 map = 100%,  reduce = 86%, Cumulative CPU 31.81 sec
2018-08-13 07:16:39,109 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 34.9 sec
MapReduce Total cumulative CPU time: 34 seconds 900 msec
Ended Job = job_1534090218042_0003
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 34.9 sec   HDFS Read: 49873 HDFS Write: 752 SUCCESS
Total MapReduce CPU Time Spent: 34 seconds 900 msec
OK
Time taken: 120.615 seconds
hive> 
```



```
hive> select * from college;
OK
6       IIT     atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 0.343 seconds, Fetched: 6 row(
```