Session 9 - ADVANCED HIVE

Assignment 1

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

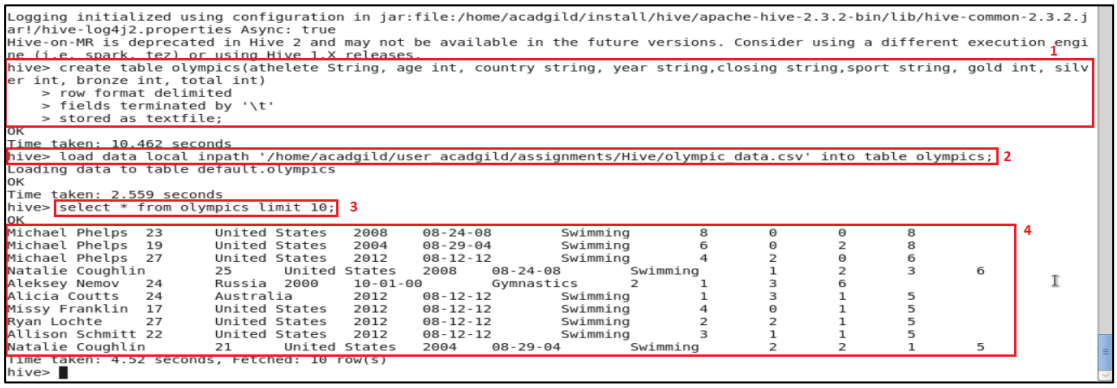
Gold Medals: No. of Gold medals

Silver Medals: No. of Silver medals

Bronze Medals: No. of Bronze medals

Total Medals: Consists of total no. of medals

Create the above table ‘olympics’ with the above structure.



With reference to the screenshot above,

1 : create a table using the following syntax.

create table olympics(athelete string, age int,country string,year string,closing string,sport string, gold int, silver int, bronze int ,total int) row format delimited fields terminated by '\t' stored as textfile;

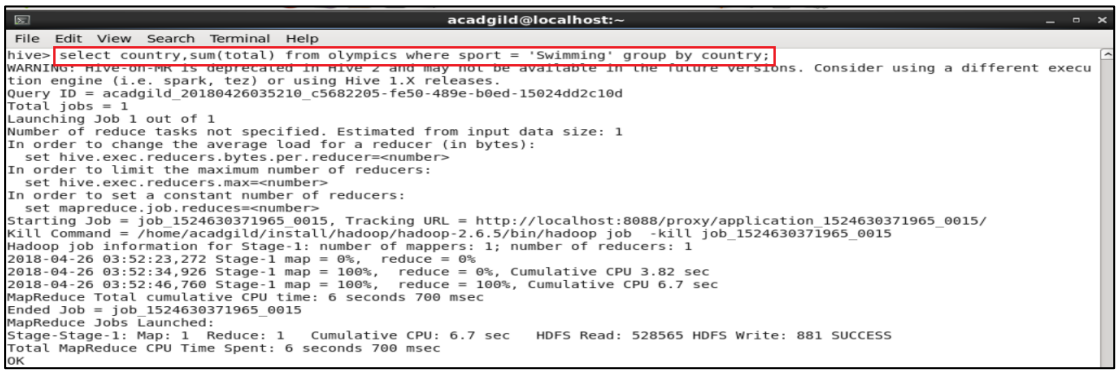
2 : load the data from the text file.

Load data local inpath '/home/acadgild/user_acadgild/assignments/Hive/olympic_data.csv' into table olympics;

3 : displaying the contents of the table.

TASK 1:

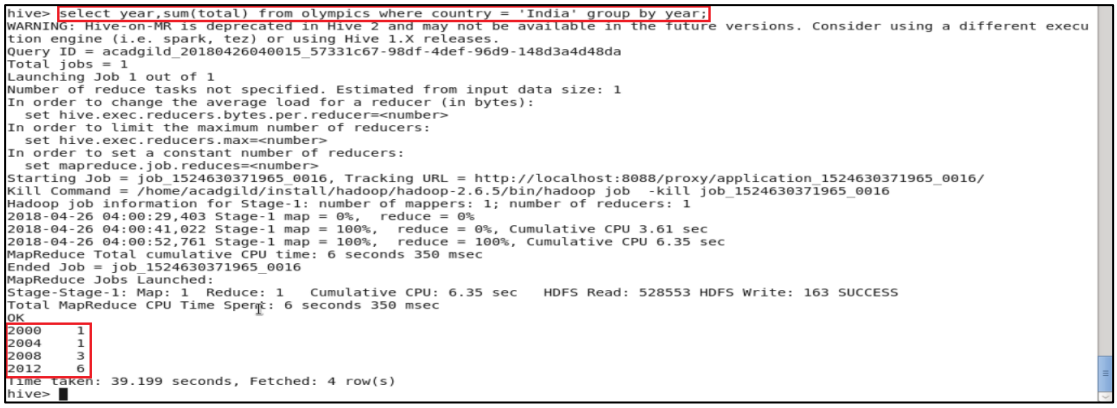
1. Write a Hive program to find the number of medals won by each country in swimming.



select country,sum(total) from olympics where sport = 'Swimming' group by country;

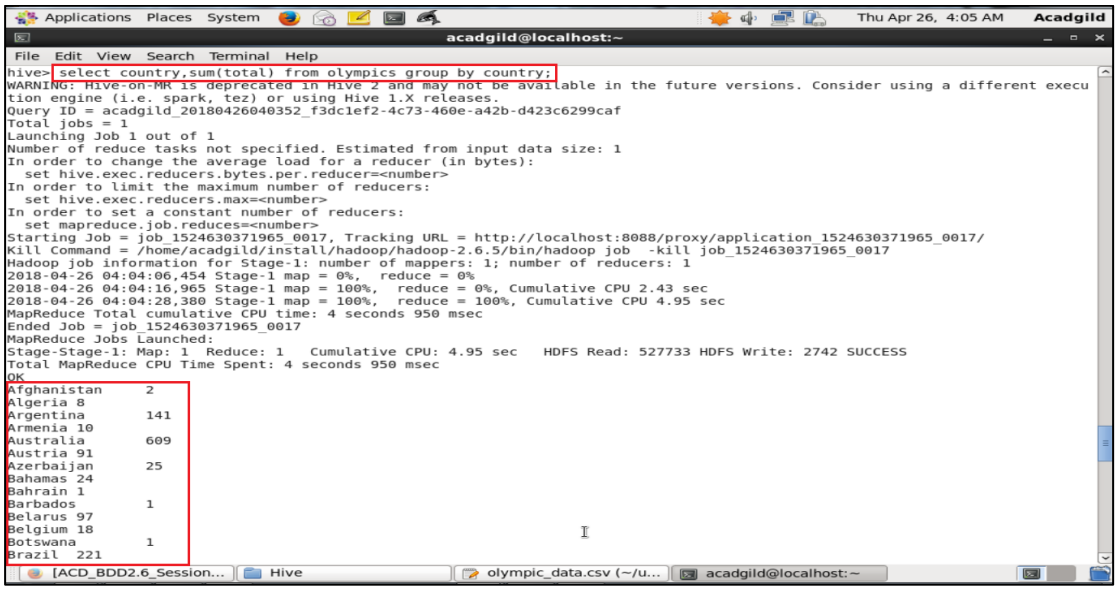
2. Write a Hive program to find the number of medals that India won year wise.

select year,sum(total) from olympics where country = 'India' group by year;



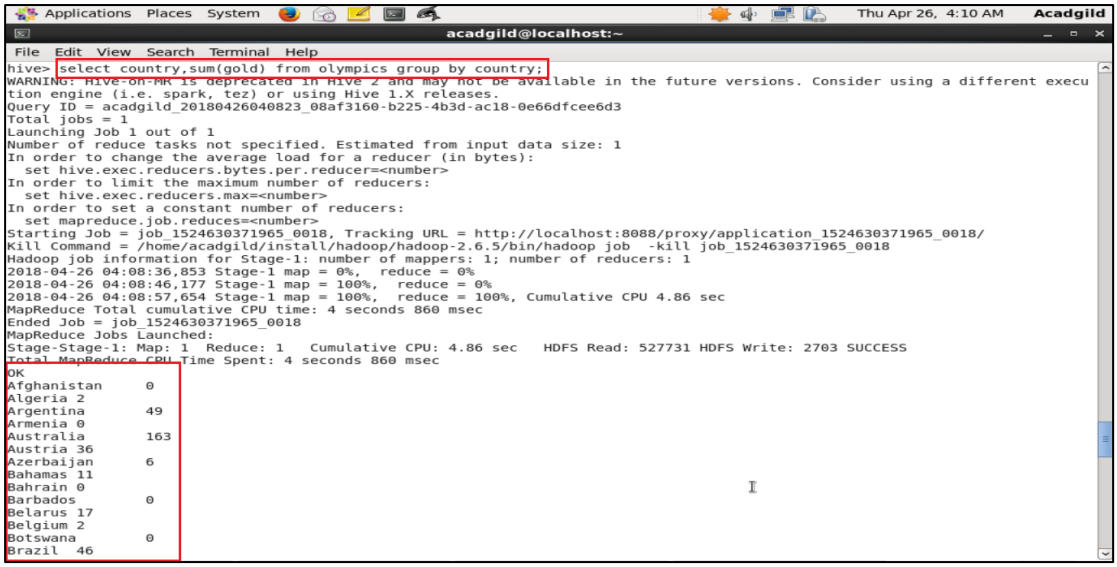
3. Write a Hive Program to find the total number of medals each country won.

select country, sum(total) from olympics group by country;



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

select country, sum(gold) from olympics group by country;



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.

The UDF has to be created by extending the *org.apache.hadoop.hive.ql.exec.UDF* class.

```
package com.acadgild.hiveudf;

import java.util.ArrayList;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public class StringConcatUDF extends UDF {
    private Text result = new Text();
    public Text evaluate(String sep, ArrayList<String> stringChars) {
        if (sep == null) {
            return null;
        }
        String tempstr = "";
        for (int i = 0; i <= stringChars.size() - 1; i++) {
            tempstr = tempstr + (stringChars.get(i) + sep);
        }
```

```
        String finalstr = tempstr.substring(0, tempstr.length() - 1);
        result.set(finalstr);
        return result;
    }
    public Text evaluate(Text str) {
        if (str == null) {
            return null;
        }
        result.set(StringUtils.strip(str.toString()));
        return result;
    }
}
```

- Create a jar file for the java file.
- Add the jar in hive list of jars.

    add jar '/location/of/the/jar/file'

Create a table with a column with array datatype.

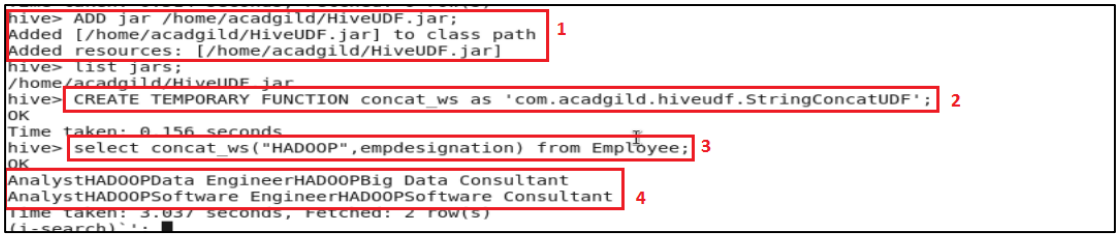
With reference to the screenshot above,

1 : creating a table employee where the fields are delimited using a tab space and the values in an array are separated using comma.

2 : the datatype of the column is array.

3 : sample data from a text file is loaded.

4 : The table is loaded with the data and the array can be seen.



With reference to the screenshot above,

1 : Adding jar to hive. Verifying the jar is added to hive, using 'list jars'.

2 : A temporary function is created with the classname to be used.

CREATE TEMPORARY FUNCTION concat_ws as 'com.acadgild.hiveudf.StringConcatUDF';

3 : Using the method.

select concat_ws("HADOOP",empdesignation) from Employee;

4 : The word HADOOP (1st arguement) is concatenated between each field in the array.

TASK 3:

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

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

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

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

Creating a table to support 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: 11.824 seconds
hive> show tables;
OK
college
employee
olympics
Time taken: 0.653 seconds, Fetched: 3 row(s)
hive>
```

The table name is displayed the list of 'show tables;'

Inserting data to 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');

```
hive> desc college;
OK
clg_id                  int
clg_name                string
clg_loc                 string
Time taken: 0.384 seconds, Fetched: 3 row(s)
hive> 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');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180430012309_50a10c62-56be-4e2f-8016-b0291ba5b3fb
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_1525006545565_0001, Tracking URL = http://localhost:8088/proxy/application_1525006545565_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1525006545565_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-04-30 01:23:51,625 Stage-1 map = 0%,  reduce = 0%
2018-04-30 01:24:11,439 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.83 sec
2018-04-30 01:25:00,789 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 9.28 sec
2018-04-30 01:25:01,900 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 10.66 sec
2018-04-30 01:25:22,323 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 27.24 sec
MapReduce Total cumulative CPU time: 28 seconds 630 msec
Ended Job = job_1525006545565_0001
Loading data to table default.college
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 28.63 sec   HDFS Read: 26737 HDFS Write: 4001 SUCCESS
Total MapReduce CPU Time Spent: 28 seconds 630 msec
OK
Time taken: 140.7 seconds
hive>
```

Data is inserted into the table.

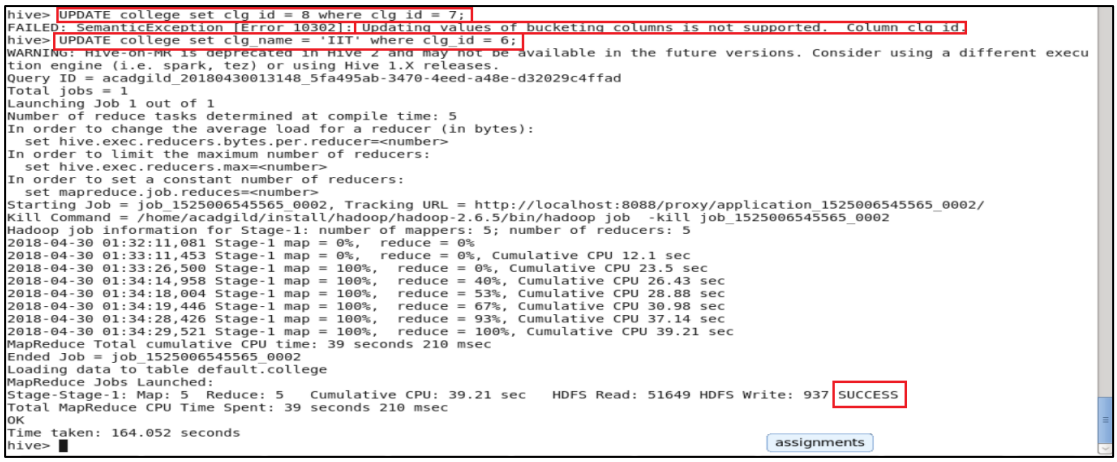
```
hive> select * from college;
OK
5       stanford        uk
6       JNTUA   atp
1       nec     nlr
7       cambridge       us
2       vit     vlr
3       srm     chen
4       lpu     del
Time taken: 1.483 seconds, Fetched: 7 row(s)
```

Updating Data in Hive table :

UPDATE college set clg_id = 8 where clg_id = 7;

Bucketed column cannot be udpated. Only non bucketed columns can be updated.

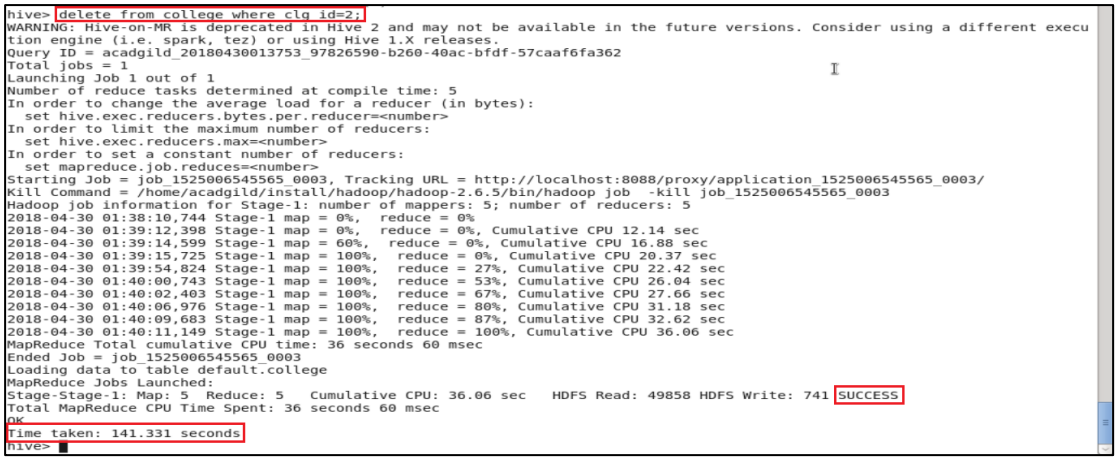
UPDATE college set clg_name = 'IIT' where clg_id = 6;



The updated values are reflected in the table.

Deleting a row from the table :

delete from college where clg_id = 2;



The data is reflected in the table.

```
hive> select * from college;
OK
5       stanford        uk
6       IIT     atp
1       nec     nlr
7       cambridge       us
3       srm     chen
4       lpu     del
Time taken: 0.514 seconds, Fetched: 6 row(s)
hive>
```

Row with clg_id 2 is deleted from the table.