

## Assignment Submission- Session 9

### DATE 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

### Task 1

**Solution:** First of all, creating a hive table and inserting the data into it.

```
CREATE TABLE OLYMPICS (name STRING, age INT, country STRING, yearParticipated  
STRING, closingDate STRING, sports STRING, goldMedals INT, silverMedals INT, bronzeMedals  
INT, totalMedals INT)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
WITH SERDEPROPERTIES (  
"separatorChar" = "\t"  
)  
STORED AS TEXTFILE;
```

### LOADING DATA INTO TABLE

```
LOAD DATA LOCAL INPATH '/home/acadgild/sumanth/HIVE_SESSIONS/olympix_data.csv'  
INTO TABLE olympics;
```

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

**Hive query:** SELECT country, sum(totalMedals) FROM olympics WHERE sports='Swimming'  
GROUP BY country;

```

Argentina      1.0
Australia      163.0
Austria 3.0
Belarus 2.0
Brazil 8.0
Canada 5.0
China 35.0
Costa Rica     2.0
Croatia 1.0
Denmark 1.0
France 39.0
Germany 32.0
Great Britain  11.0
Hungary 9.0
Italy 16.0
Japan 43.0
Lithuania      1.0
Netherlands    46.0
Norway 2.0
Poland 3.0
Romania 6.0
Russia 20.0
Serbia 1.0
Slovakia       2.0
Slovenia       1.0
South Africa   11.0
South Korea    4.0
Spain 3.0
Sweden 9.0
Trinidad and Tobago 1.0
Tunisia 3.0
Ukraine 7.0
United States  267.0
Zimbabwe       7.0

```

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

**Hive Query:**

```

SELECT yearParticipated, sum(totalMedals) FROM olympics WHERE country='India' GROUP BY
yearParticipated;

```

```

Total MapReduce CPU Time Spent: 10 seconds 20 msec
OK
2000      1.0
2004      1.0
2008      3.0
2012      6.0
Time taken: 62.681 seconds, Fetched: 4 row(s)
hive>

```

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

**Hive Query:**

```
SELECT country, sum(totalMedals) FROM olympics GROUP BY country;
```

**Note:** Output not completely captured in screenshot.

```
Stage: Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 0.0
Total MapReduce CPU Time Spent: 8 seconds 350 msec
OK
Afghanistan      2.0
Algeria 8.0
Argentina        141.0
Armenia 10.0
Australia        609.0
Austria 91.0
Azerbaijan       25.0
Bahamas 24.0
Bahrain 1.0
Barbados         1.0
Belarus 97.0
Belgium 18.0
Botswana         1.0
Brazil 221.0
Bulgaria         41.0
Cameroon         20.0
Canada 370.0
Chile 22.0
China 530.0
Chinese Taipei   20.0
Colombia         13.0
Costa Rica       2.0
Croatia 81.0
Cuba 188.0
Cyprus 1.0
Czech Republic   81.0
Denmark 89.0
Dominican Republic 5.0
Ecuador 1.0
Egypt 8.0
Eritrea 1.0
Estonia 18.0
Ethiopia        29.0
Finland 118.0
France 318.0
Gabon 1.0
Georgia 23.0
Germany 629.0
Great Britain    322.0
Greece 59.0
Grenada 1.0
Guatemala        1.0
Hong Kong        3.0
Hungary 145.0
Iceland 15.0
India 11.0
```

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

**Hive Query:**

```
SELECT country, sum(goldMedals) FROM olympics GROUP BY country;
```

Output not completely captured in screenshot:

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU

Total MapReduce CPU Time Spent: 8 seconds 410 msec

OK

Afghanistan	0.0
Algeria	2.0
Argentina	49.0
Armenia	0.0
Australia	163.0
Austria	36.0
Azerbaijan	6.0
Bahamas	11.0
Bahrain	0.0
Barbados	0.0
Belarus	17.0
Belgium	2.0
Botswana	0.0
Brazil	46.0
Bulgaria	8.0
Cameroon	20.0
Canada	168.0
Chile	3.0
China	234.0
Chinese Taipei	2.0
Colombia	2.0
Costa Rica	0.0
Croatia	35.0
Cuba	57.0
Cyprus	0.0
Czech Republic	14.0
Denmark	46.0
Dominican Republic	3.0
Ecuador	0.0
Egypt	1.0
Eritrea	0.0
Estonia	6.0
Ethiopia	13.0
Finland	11.0
France	108.0
Gabon	0.0
Georgia	6.0
Germany	223.0
Great Britain	124.0
Greece	12.0
Grenada	1.0
Guatemala	0.0
Hong Kong	0.0
Hungary	77.0
Iceland	0.0
India	1.0
Indonesia	5.0
Iran	10.0
Ireland	1.0

## 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:***

- Created a table in hive:

create table skillTable(name string, skill array<string>) row format delimited fields terminated by '\t'  
collection items terminated by ',';

- Loaded data into table:

load data local inpath '/home/acadgild/sumanth/HIVE\_SESSIONS/skillData.txt' into table skillTable;

- Select all data from table:

```
hive> select * from skillTable;
OK
Akshat  ["TCL","MQL","Core Java","Pig","Hive"]
Rahul   ["TCL","MQL","SQL"]
Time taken: 2.58 seconds, Fetched: 2 row(s)
```

- Created a jar file for udf.

### **UDF program code:**

```
import java.util.ArrayList;
import org.apache.hadoop.hive.ql.exec.UDF;
public class ConcatArrayElements extends UDF {
    public String evaluate(String del, ArrayList<String> arrayelements) {
        if (arrayelements == null) {
            return null;
        }
        StringBuffer sbuf = new StringBuffer();
        sbuf.append(arrayelements.get(0));
        for (int i = 1; i < arrayelements.size(); i++) {
            sbuf.append(del);
            sbuf.append(arrayelements.get(i));
        }
        return sbuf.toString();
    }
}
```

- Added jar to hive.

```
hive> add jar /home/acdadgild/sumanth/HIVE_SESSIONS/ConcatArrElements.jar;
Added [/home/acdadgild/sumanth/HIVE_SESSIONS/ConcatArrElements.jar] to class
path
Added resources: [/home/acdadgild/sumanth/HIVE_SESSIONS/ConcatArrElements.jar]
```

- Created temporary function.

Jar hive -exec couldn't be downloaded so couldn't proceed further for running the function.

### 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.

### Solution:

Turning ON features to support transactions:

```
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;
```

### Creating table using ORC format

CREATE TABLE company(cmp\_id int,cmp\_name string,cmp\_loc string) clustered by (cmp\_id) into 3 buckets stored as orc TBLPROPERTIES('transactional'='true');

```
hive> CREATE TABLE company(cmp_id int,cmp_name string,cmp_loc string) clustered by (cmp_id) into 3 buckets stored as orc TBLPROPERTIES('transactional'='true');
OK
Time taken: 2.525 seconds
```

### Inserting data into table created

INSERT INTO table company values

(1,'tata','india'),(2,'facebook','US'),(3,'Sopra','France'),(4,'Barclays','England');

```
hive> INSERT INTO table company values(1,'tata','india'),(2,'facebook','US'),(3,'Sopra','France'),(4,'Barclays','England');
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_20180925214038_0b440751-33ec-44da-a506-d24bca4722fa
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```



Data after inserting in table:

```
OK
3      Sopra      France
4      Barclays   England
1      tata       india
2      facebook   US
```

Updating record: (Update doesn't work on bucketed column)

UPDATE company set cmp\_name = 'wipro' where cmp\_id = 1;

```
hive> UPDATE company set cmp_name = 'wipro' where cmp_id = 1;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in
Query ID = acadgild_20180925214813_bd90eb89-8ad0-410a-8e62-d8e82cf046d0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
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:
```

Deleting record:

delete from company where cmp\_id=3;

```
hive> delete from company where cmp_id=3;
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.
Query ID = acadgild_20180925215135_f1b7b390-5a13-43ac-af3b-46011126b378
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
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>
```

Data post update and deletion:

```
hive> SELECT * FROM company;
OK
4      Barclays   England
1      wipro      india
2      facebook   US
Time taken: 0.524 seconds, Fetched: 3 row(s)
hive>
```