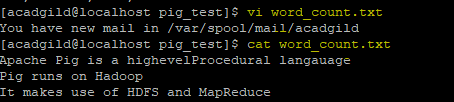
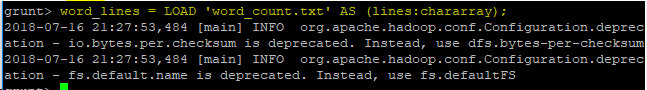
**Exploring Pig:**

**Task 1:**

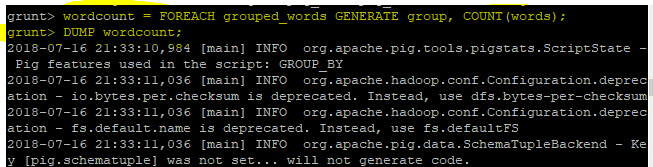
Write a program to implement wordcount using Pig.

**Step 1: Creating a file with some data:**

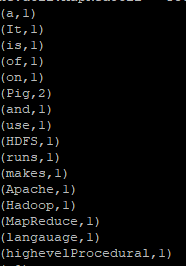






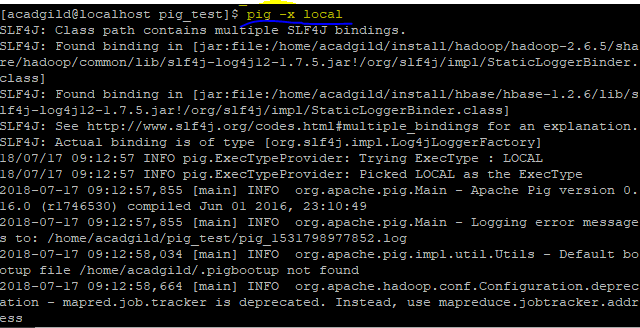


**OUTPUT:**



**Task 2:**

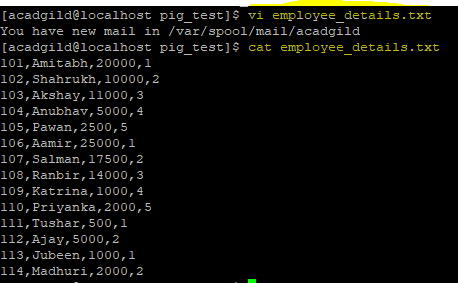
**We have employee\_details and employee\_expenses files. Use local mode while running Pig and write Pig Latin script to get below results:**

Step 1: Running Pig in Local mode

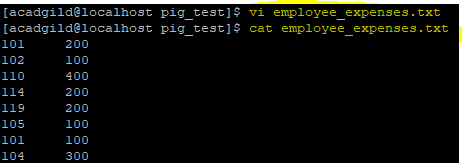
Grunt Shell opens:



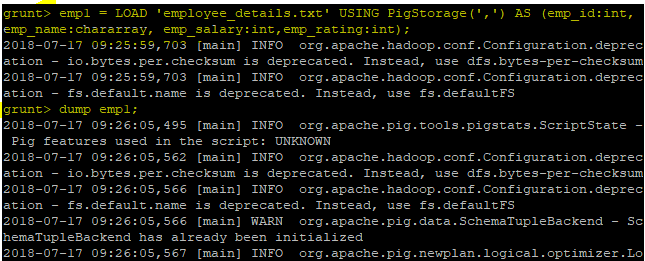
employee\_details (EmpID,Name,Salary,Rating)

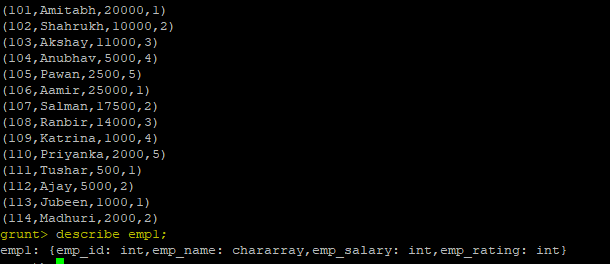


employee\_expenses(EmpID,Expense)

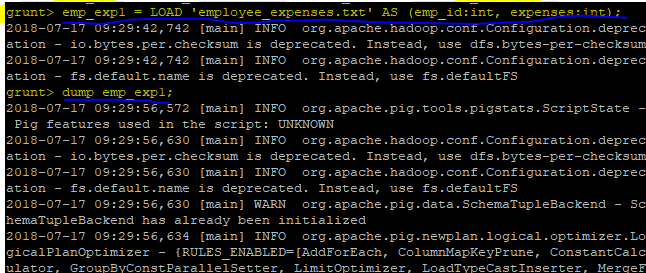


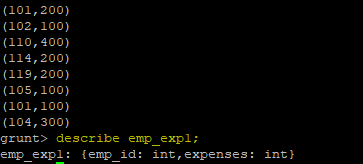
Step 3: Loading the “employee\_details file”





Step 4: Loading “employee\_expenses.txt” file

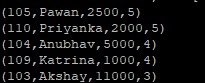




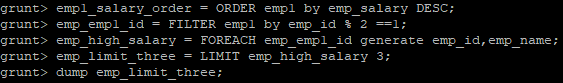
1. **Top 5 employees (employee id and employee name) with highest rating. (In case two employees have same rating, employee with name coming first in dictionary should get preference)**



**Output:**



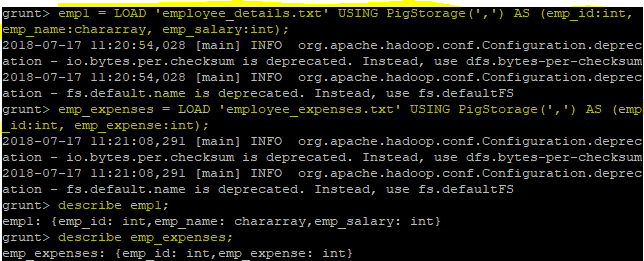
**(b) Top 3 employees (employee id and employee name) with highest salary, whose employee id is an odd number. (In case two employees have same salary, employee with name coming first in dictionary should get preference**

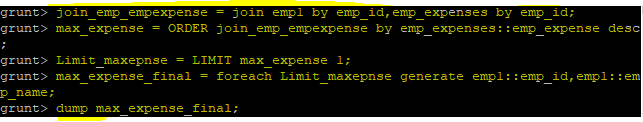


**Output:**



**(c) Employee (employee id and employee name) with maximum expense (In case two employees have same expense, employee with name coming first in dictionary should get preference)**

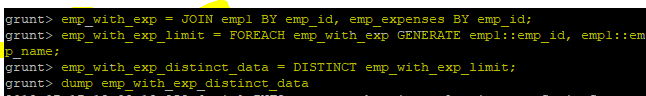




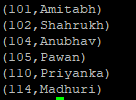
**OUTPUT:**



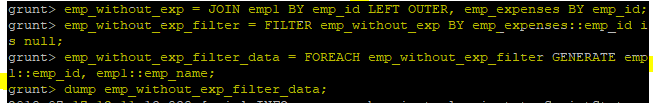
**(d) List of employees (employee id and employee name) having entries in employee\_expenses file.**



**OUTPUT:**



**(e) List of employees (employee id and employee name) having no entry in employee\_expenses file.**



**Output:**



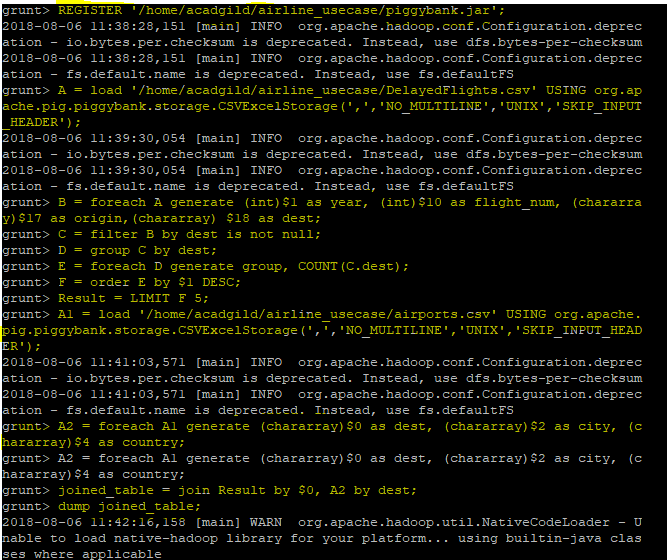
**Task 3:**

**Implement the use case present in below blog link and share the complete steps along with screenshot(s) from your end.**

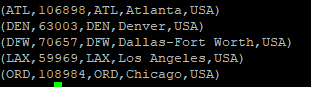
**Problem Statement 1**

Find out the top 5 most visited destinations.

**Commands to execute:**



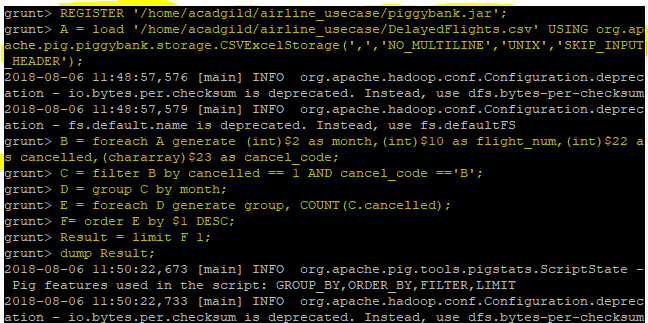
OUTPUT:



**Problem Statement 2**

**Which month has seen the greatest number of cancellations due to bad weather**

**Commands to execute:**



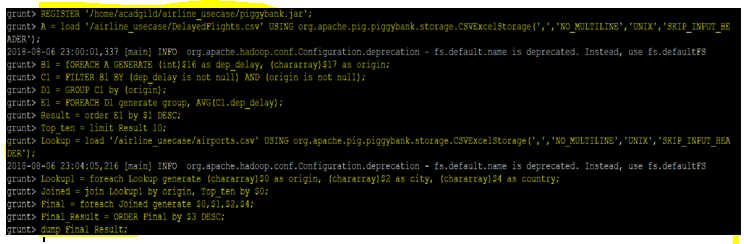
OUTPUT:



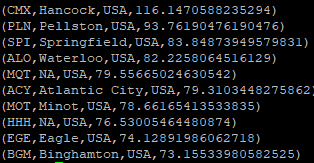
**Problem Statement 3**

**Top ten origins with the highest AVG departure delay**

**Commands to execute:**



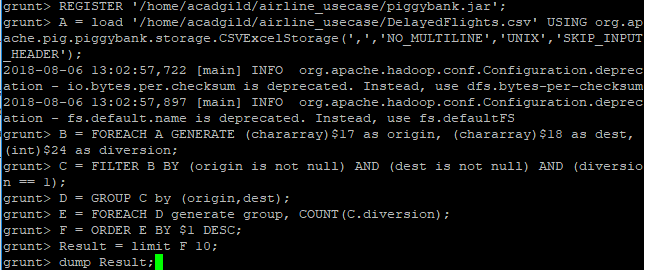
**OUTPUT:**



**Problem Statement 4**

**Which route (origin & destination) has seen the maximum diversion?**

**Commands to execute:**



**OUTPUT:**



**HIVE Assignment**

**Task 1:**

Create a database named 'custom'. Create a table named temperature\_data inside custom having below fields:

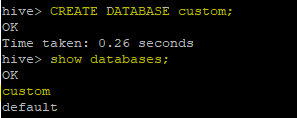
1. date (mm-dd-yyyy) format 2. zip code 3. temperature The table will be loaded from comma-delimited file.

Load the dataset.txt (which is ',' delimited) in the table.-

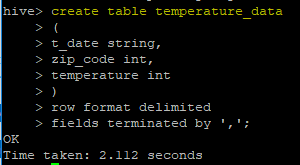
Step 1 : Ensuring database customer exists. “custom database” doesnot exists

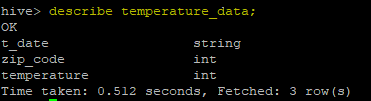


Step 2: Creating the database “custom”

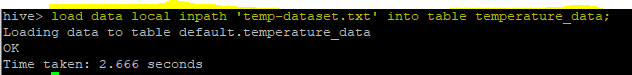


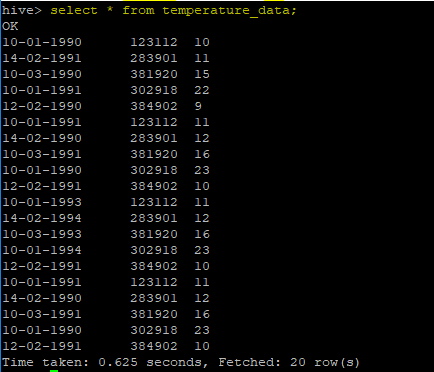
Step 3: Creating table “temperature\_data’ with fields date, zipcode and temperature





Step 4: Loading data into table “temperature\_data”



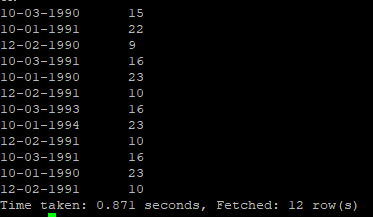


**Task 2:**

1. Fetch date and temperature from temperature\_data where zip code is greater than 300000 and less than 399999.

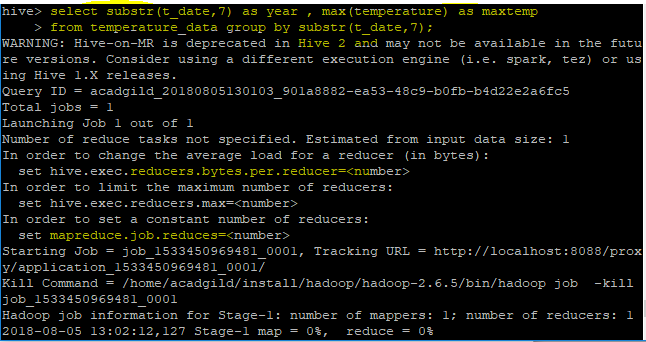


OUTPUT:

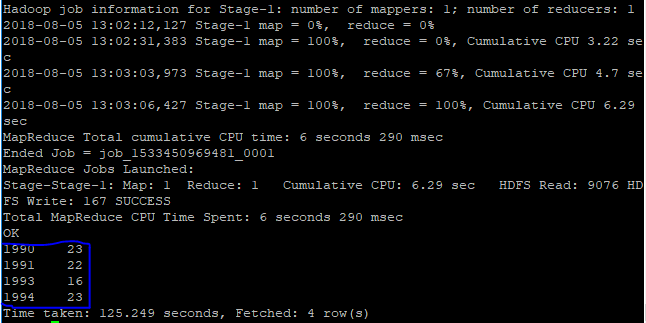


1. Calculate maximum temperature corresponding to every year from temperature\_data table.

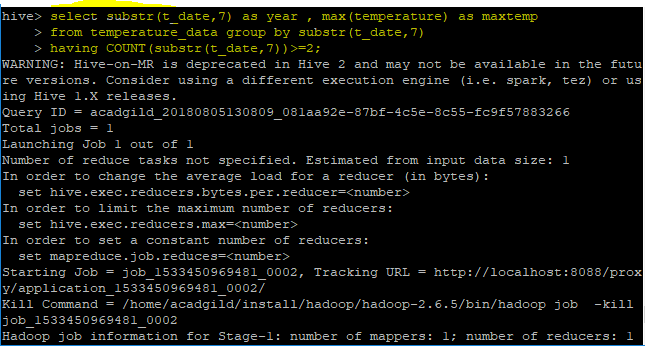
Where MapReduce launches as using aggregate functions like “max”



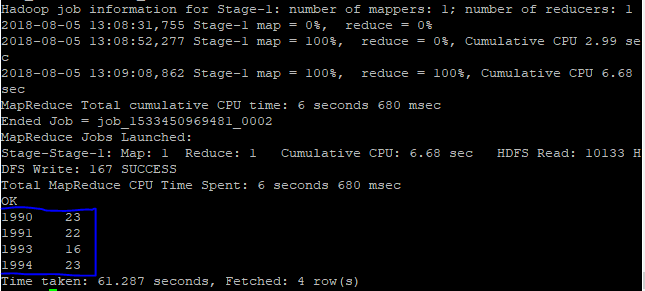
OUTPUT:



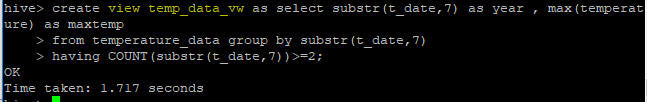
1. Calculate maximum temperature from temperature\_data table corresponding to those years which have at least 2 entries in the table.



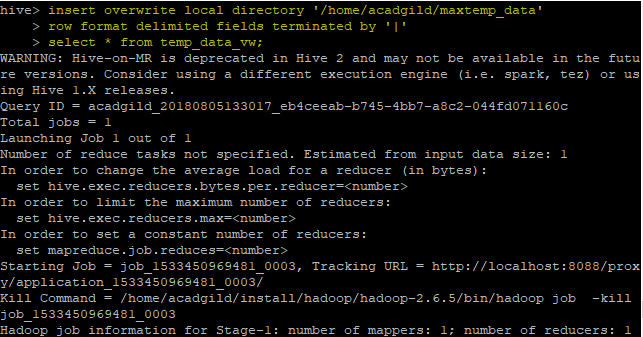
OUTPUT:

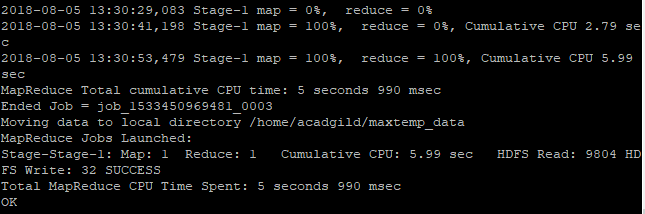


1. **Create a view on the top of last query, name it temperature\_data\_vw.**

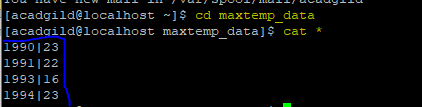


1. **Export contents from temperature\_data\_vw to a file in local file system, such that each file is '|' delimited.**





**OUTPUT :**



**Advanced Hive:**

**Task 1:**

**This Data set is about Olympics. You can download the data set from the below link: https://drive.google.com/open?id=0ByJLBTmJojjzV1czX3Nha0R3bTQ**

**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 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

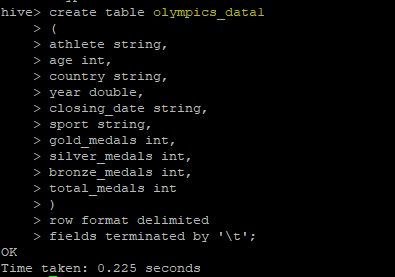
Gold Medals: No. of Gold medals

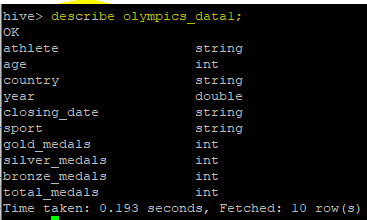
Silver Medals: No. of Silver medals

Bronze Medals: No. of Bronze medals

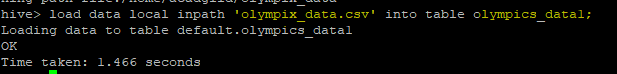
Total Medals: Consists of total no. of medals

Step 1 : creating table Olympics\_data1





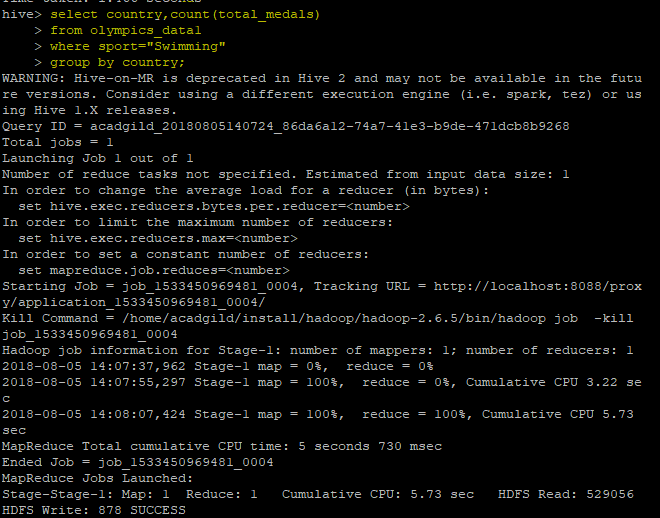
Step 2: loading data into Olympics\_data1 table



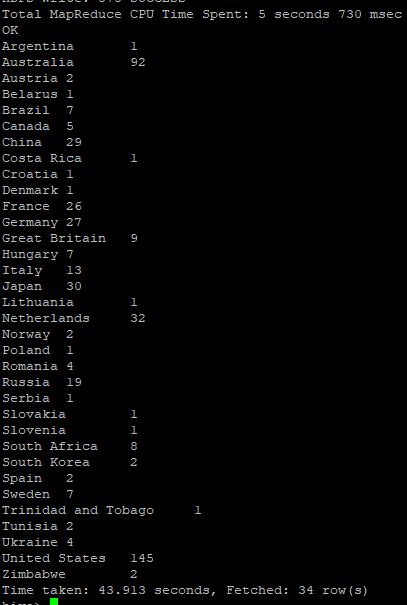
1. **Write a Hive program to find the number of medals won by each country in swimming.**

**Query : select country, count (total\_medals)from Olympics\_data1**

**Where sport = “Swimming” group by country;**



**OUTPUT:**



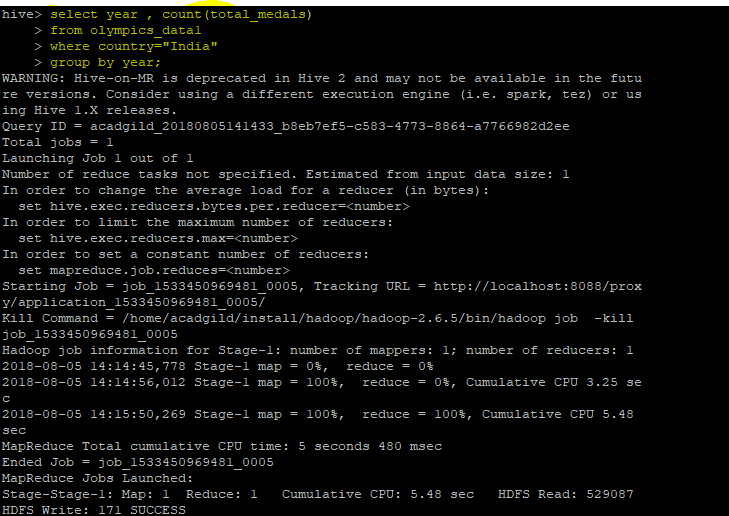
1. **Write a Hive program to find the number of medals that India won year wise.**

**Query executed to get the number of medals that India won year wise :**

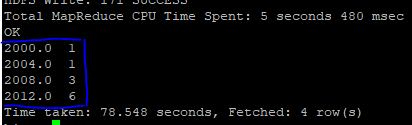
**Select year , count (total\_medals) from Olympics\_data1**

**Where country = “India”**

**Group by country;**



**OUTPUT :**



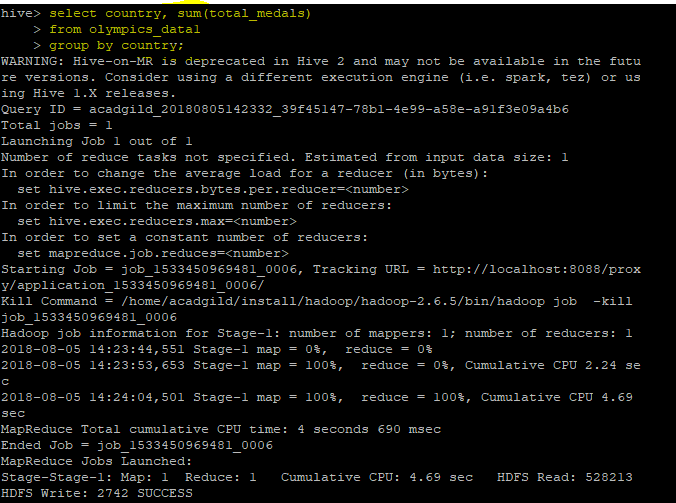
1. **Write a Hive Program to find the total number of medals each country won.**

**Query executed to get the total number of medals won by each country**

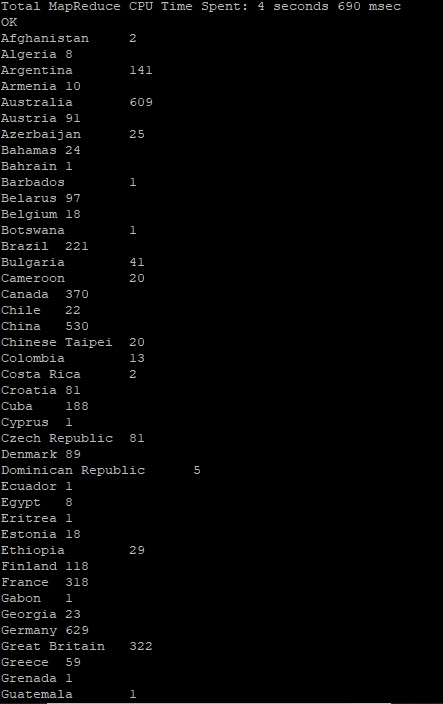
**Select country , sum(total\_medals)**

**From Olympics\_data1**

**Group by country;**



**OUTPUT :**





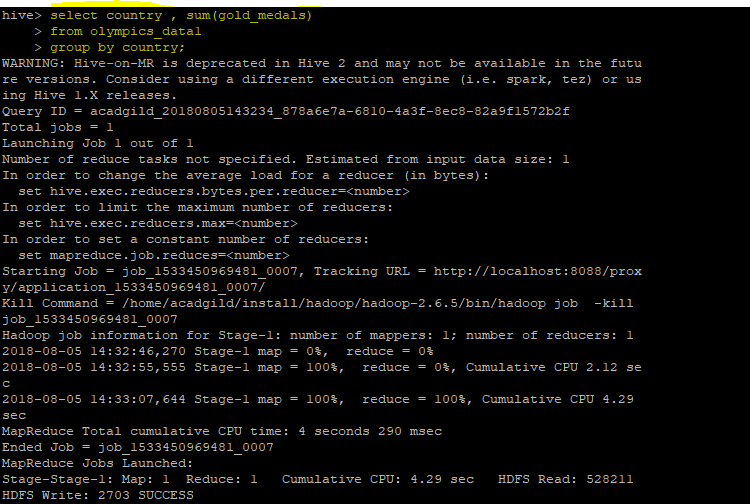


1. **Write a Hive program to find the number of gold medals each country won.**

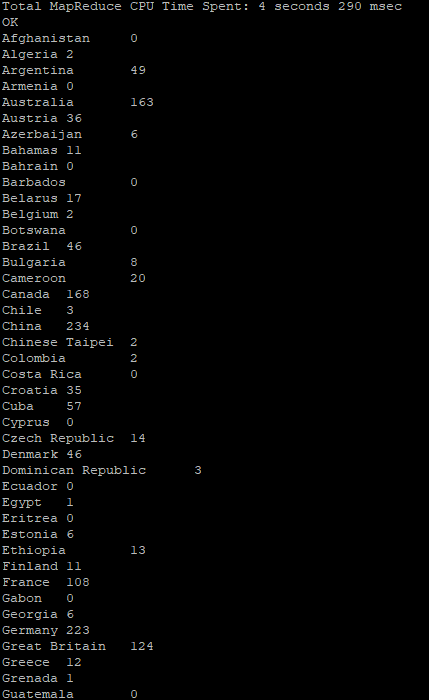
**Query executed to get the total number of gold medals each country won**

**Selectcountry,sum(gold\_medals)**

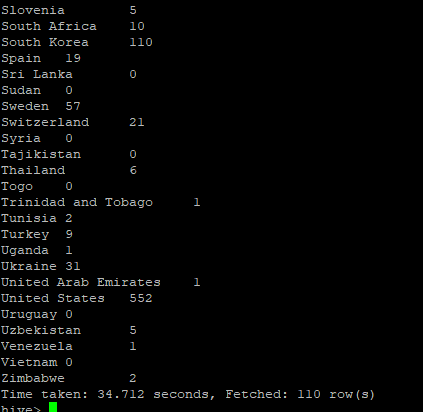
**From Olympics\_data1 group by country;**



**OUTPUT :**





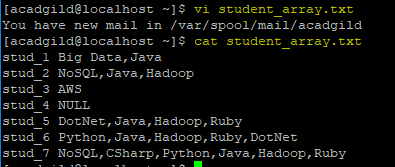


**Task 2:**

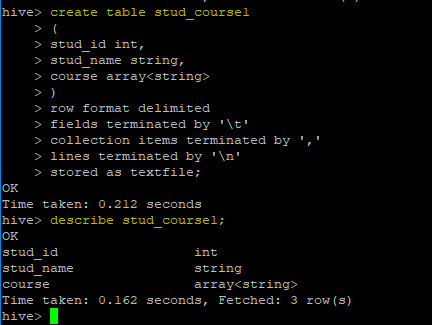
**Write a hive UDF that implements functionality of string concat\_ws(string SEP, array<string>). This UDF will accept two arguments, one string and one array of string. It will return a single string where all the elements of the array are separated by the SEP.**

**Hive.udf.jar will be attached in the Git link**

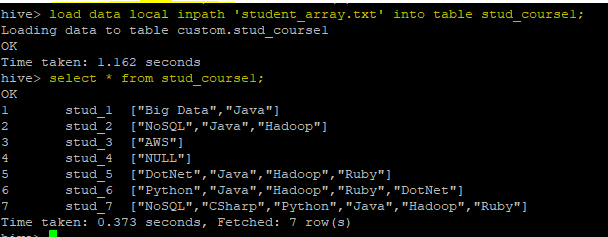
Step 1: Creating a text file stud\_course\_array.txt in local & populating the following data to load into a table



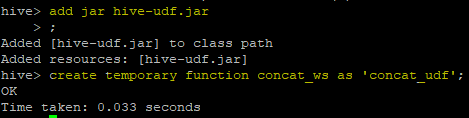
Step 2: Creating table stud\_course



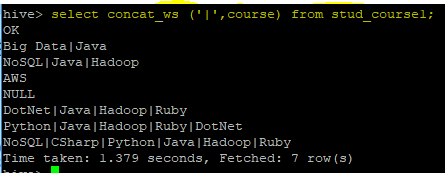
Step 3: Loading data into table “stud\_course1”



Step 4: Adding hive-udf.jar file and creating a function concat\_ws as concat\_udf



Step 5: Displaying course using HIVE UDF ‘CONCAT\_WS’ using ‘|’ separator

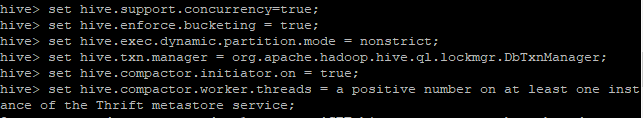


**Task 3:**

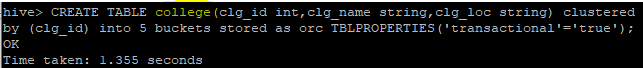
**Link: https://acadgild.com/blog/transactions-in-hive/** Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

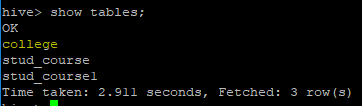
**Row-level Transactions Available in Hive 0.14**

**Step 1: Configurations:** The below properties needs to be set appropriately in ***hive shell***

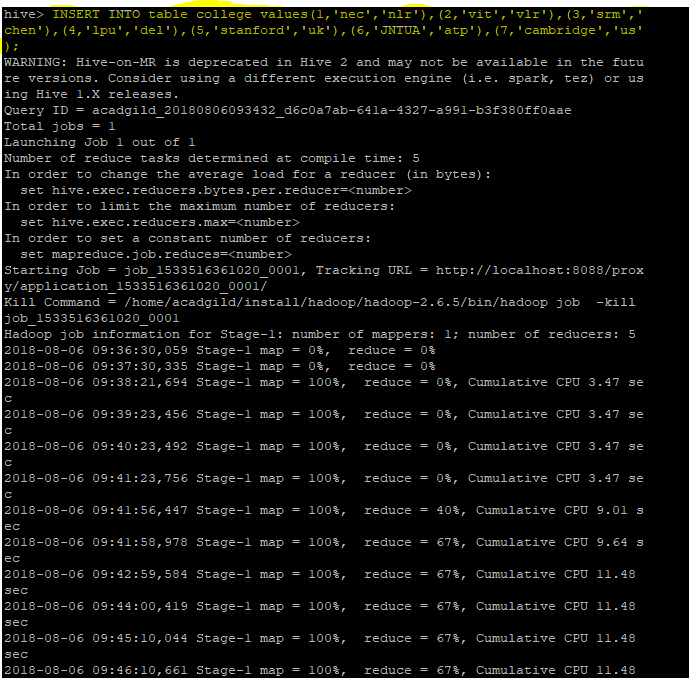


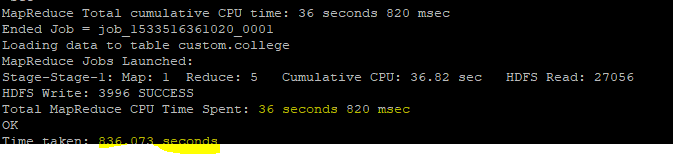
**Step 2: Creating table “college”**



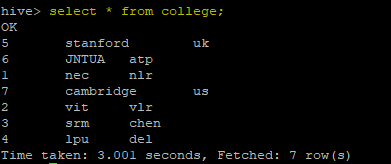


Step 3 : Inserting Data into a Hive Table





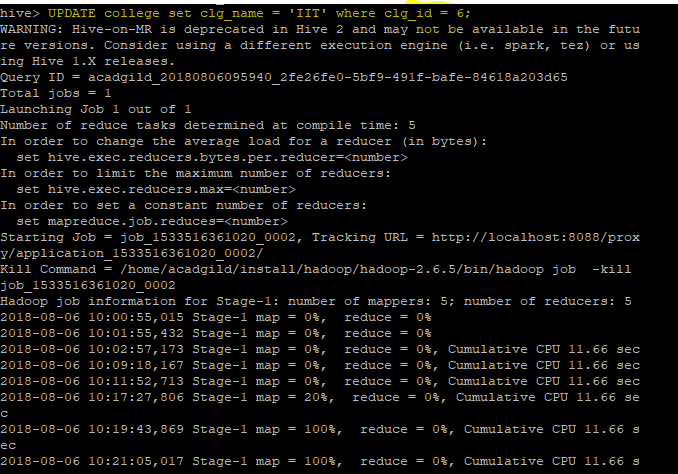
OUTPUT:

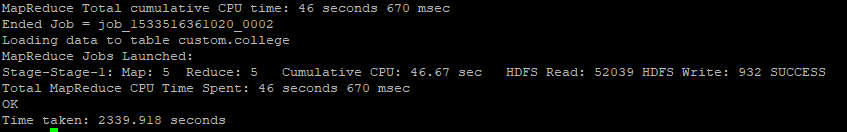


Step 4: Updating the Data in Hive Table

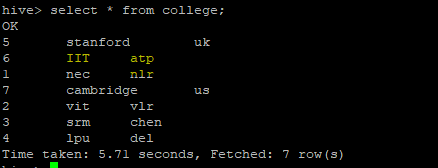
used to update a row in Hive table.





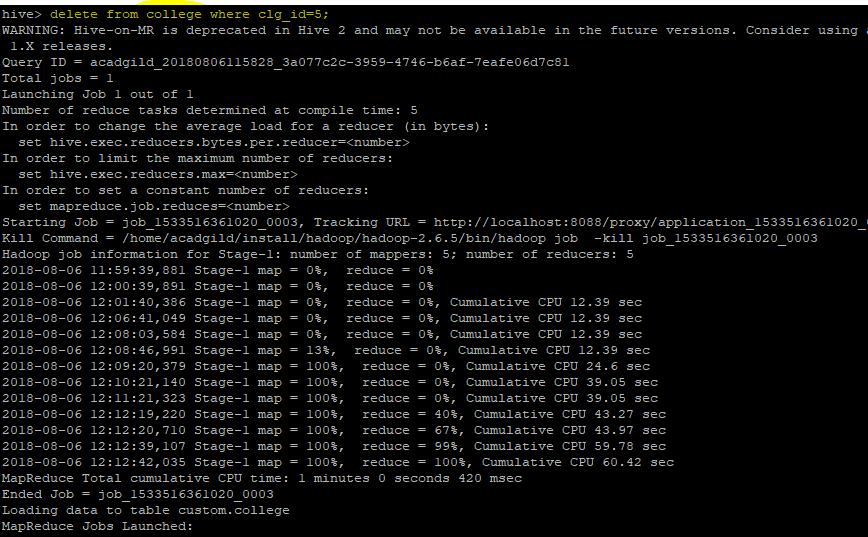


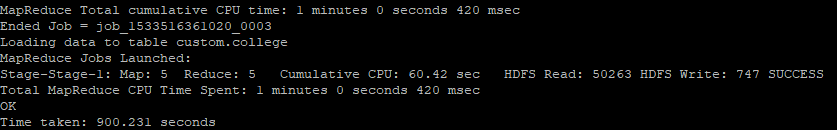
OUTPUT:



**Step 5: Deleting a Row from Hive Table**

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





OUTPUT:

