**Task 1**

Firstly, Create a table OLYMPICS inside the database custom as below

HIVE query to create table:

*CREATE TABLE OLYMPICS(*

*athelete STRING,*

*age INT,*

*country STRING,*

*year INT,*

*closedate STRING,*

*sport STRING,*

*goldmedal INT,*

*silvermedal INT,*

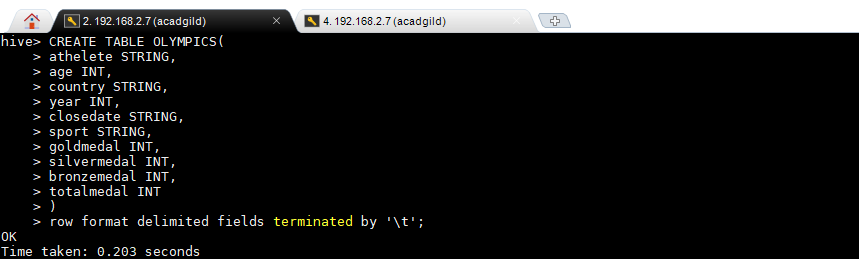
*bronzemedal INT,*

*totalmedal INT*

*)*

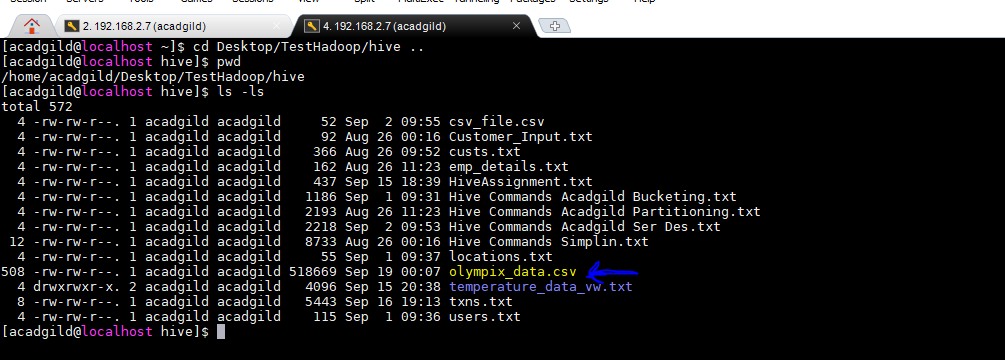
*row format delimited fields terminated by '\t';*





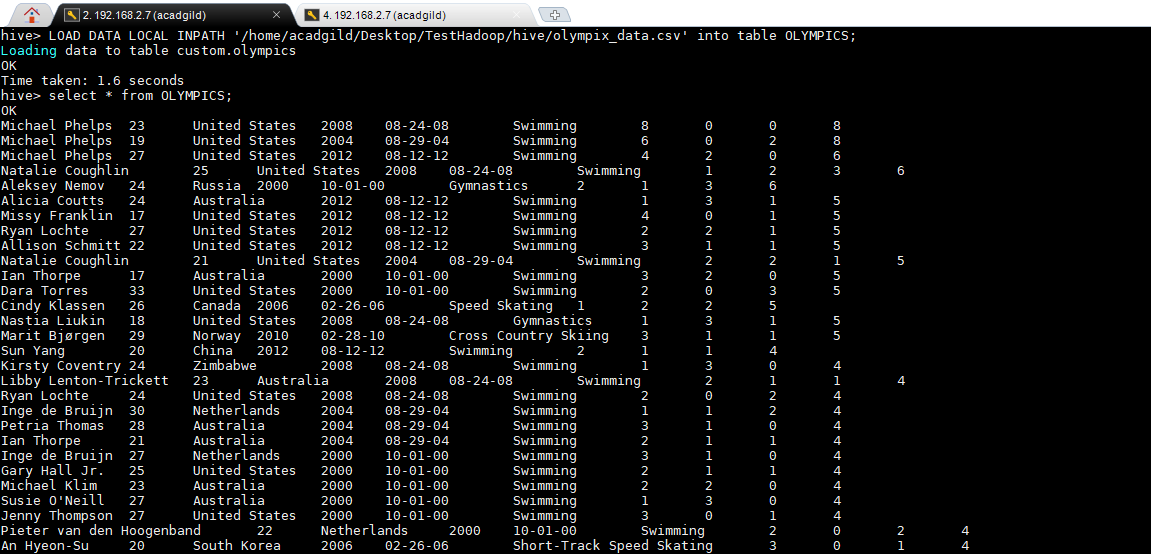
The input file to be loaded to the table is located in the below location

/home/acadfild/Desktop/TestHadoop/hive/olympix\_data.csv



To load the data to table run the command:

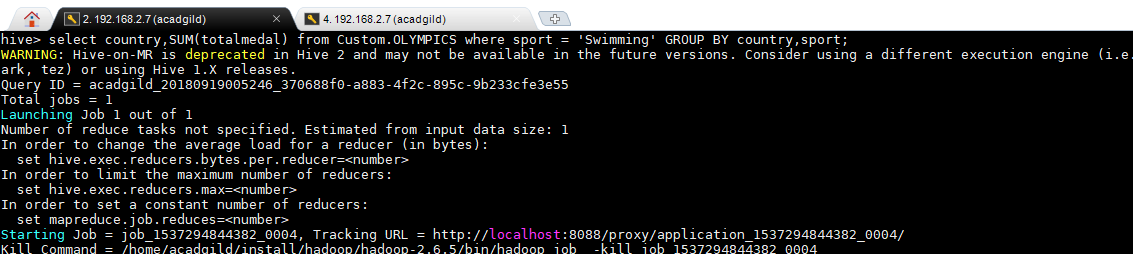
LOAD DATA LOCAL INPATH '/home/acadgild/Desktop/TestHadoop/hive/olympix\_data.csv' into table OLYMPICS;



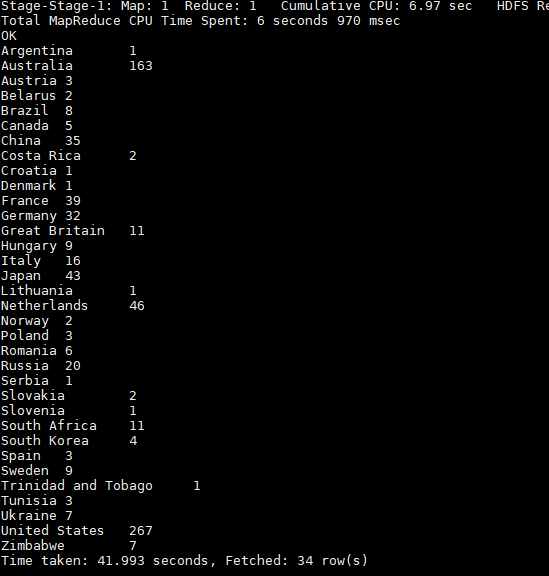
* **Write a Hive program to find the number of medals won by each country in swimming.**

**HIVE Query :**

***select country,SUM(totalmedal) from Custom.OLYMPICS where sport = 'Swimming' GROUP BY country;***

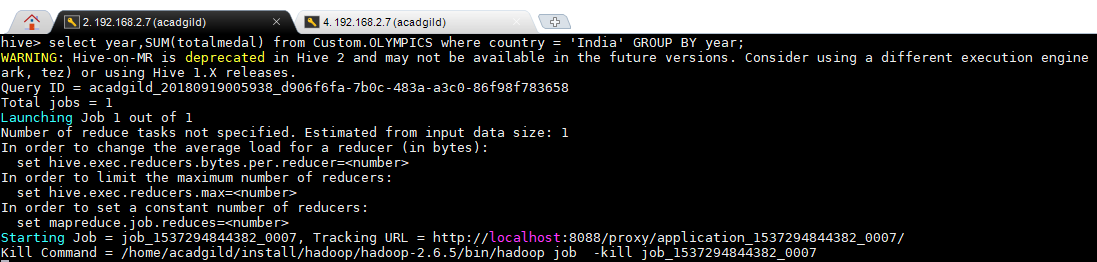
****

**Output**

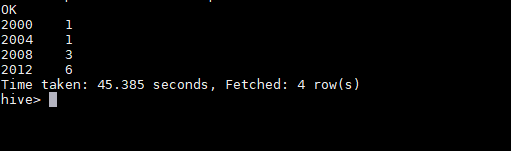
****

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

**Query: *select year,SUM(totalmedal) from Custom.OLYMPICS where country = 'India' GROUP BY year;***

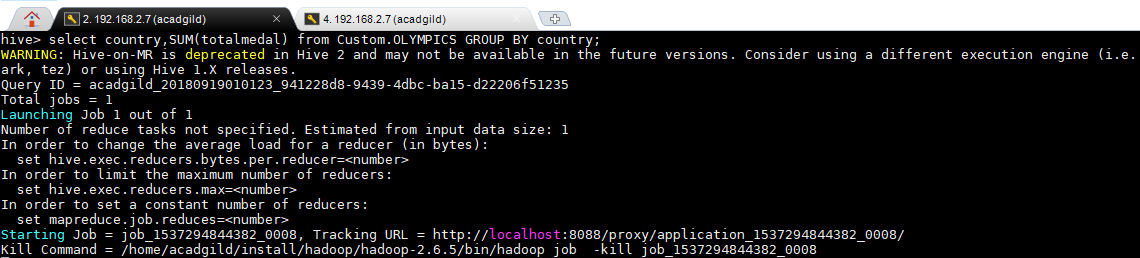
****

**Output**

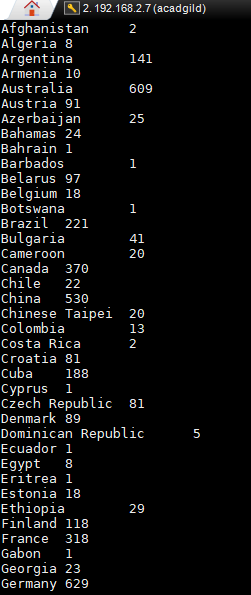
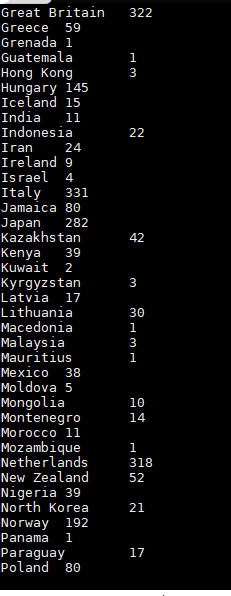
****

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

**Query : *select country,SUM(totalmedal) from Custom.OLYMPICS GROUP BY country;***

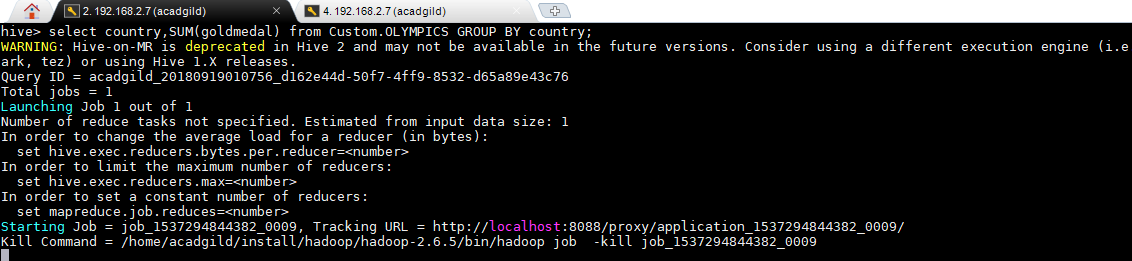
****

**Output:**

**  **

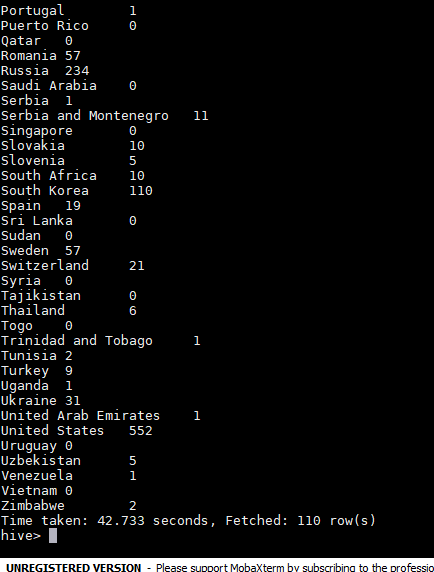
* **Write a Hive program to find the number of gold medals each country won.**

**Query : select country,SUM(goldmedal) from Custom.OLYMPICS GROUP BY country;**

****

**Output:**

** **

****

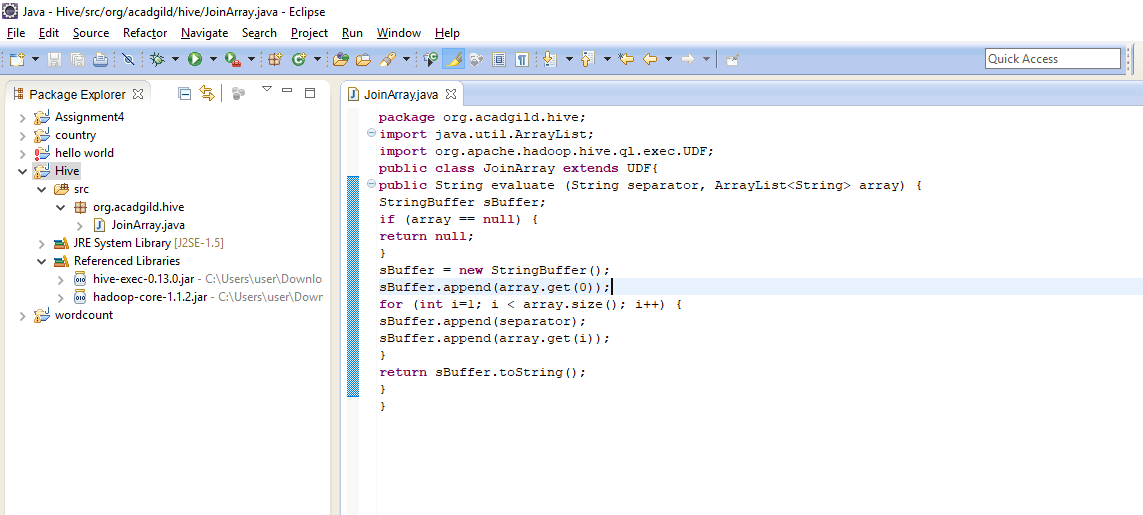
**Task 2: Write a hive UDF that implements functionality of string concat\_ws(string SEP, array<string>).**

**This UDF will accept two arguments, one string and one array of string.**

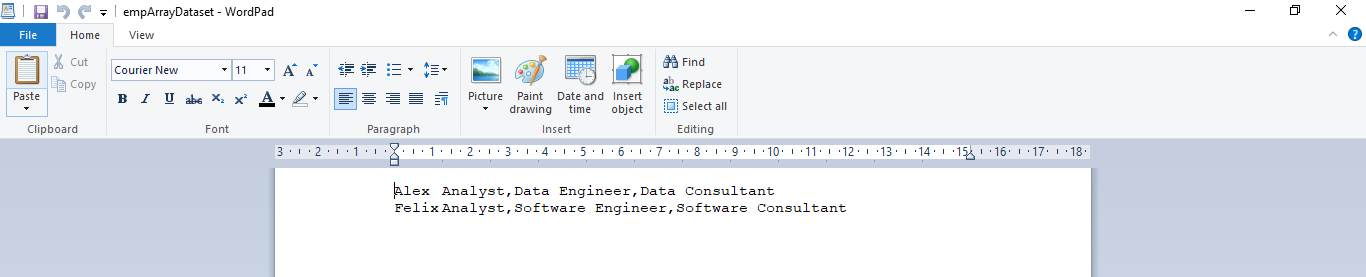
**It will return a single string where all the elements of the array are separated by the SEP.**

**Input file and the jar file is present in the location:**

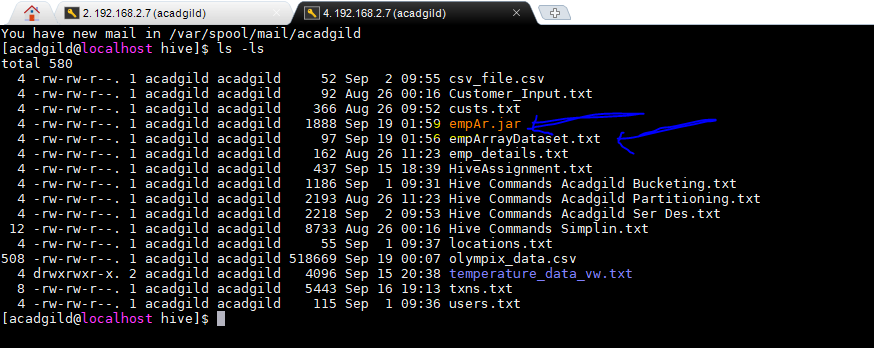
**/home/acadgild/Desktop/TestHadoop/hive/empAr.jar**

****

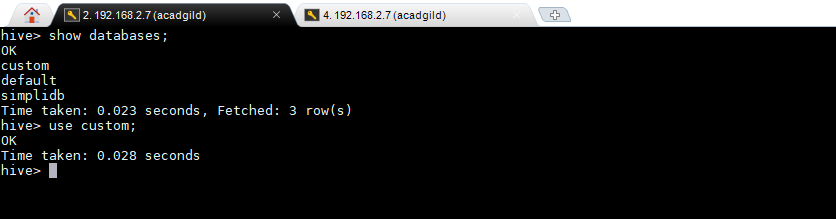
**/home/acadgild/Desktop/TestHadoop/hive/empArrayDataset.txt**

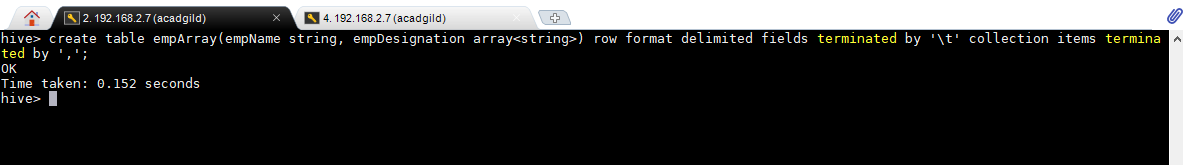
****

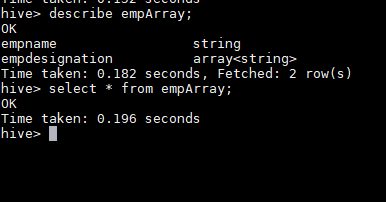
**In VM**

****

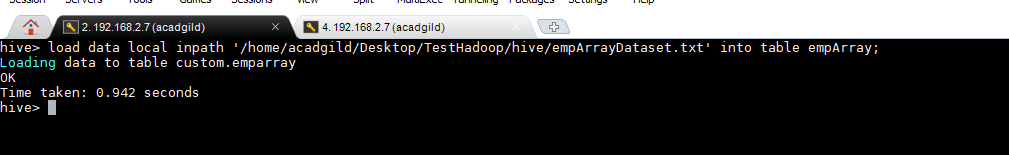
**Create table ‘empArray’ inside database ‘custom’**

****

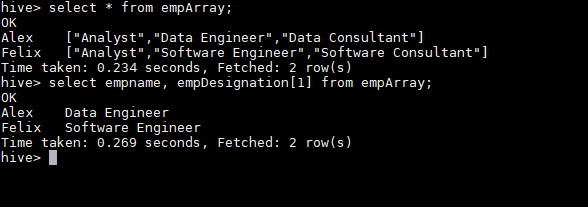
****

****

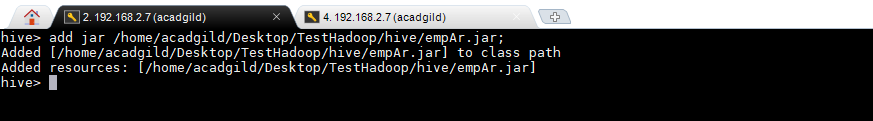
**Load the table from the input file present at /home/acadgild/Desktop/TestHadoop/hive/empArrayDataset.txt**

****

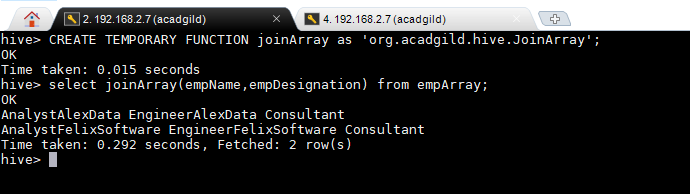
**Checking the data has been loaded or not using select statement.**

****

**Add the jar file with complete path of the jar made in the above case.**

****

**Create a temporary function as shown below and use the select statement to check the output**

****

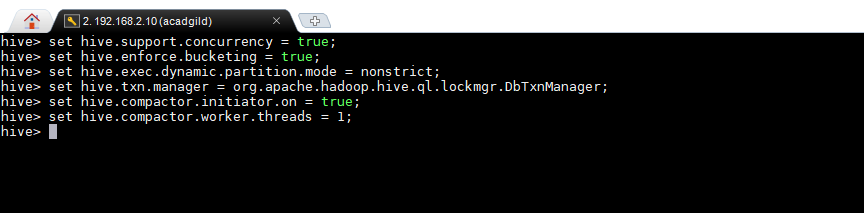
**Task 3**

**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

****

**Creating a Table That Supports Hive Transactions**

**HIVE Query :**

*CREATE TABLE sports(*

*person\_id int,*

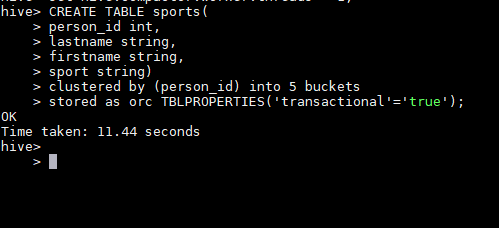
*lastname string,*

*firstname string,*

*sport string)*

*clustered by (person\_id) into 5 buckets*

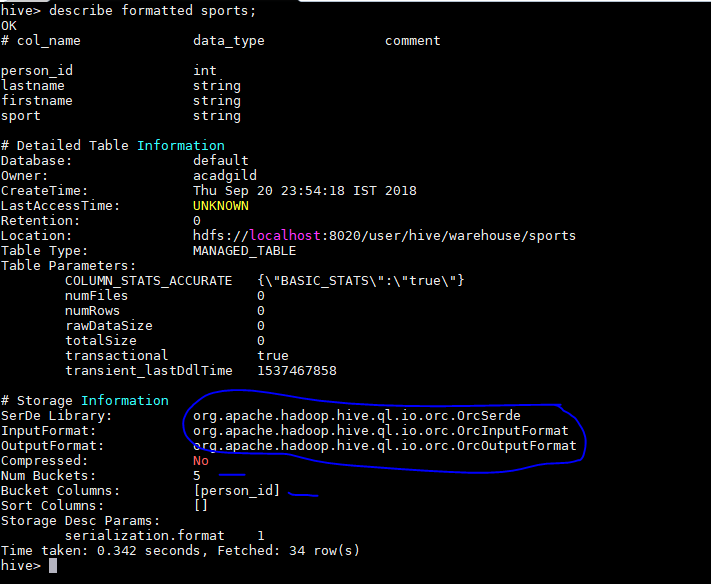
*stored as orc TBLPROPERTIES('transactional'='true');*

****

The above syntax will create a table with name ‘*sports’*and the columns present in the table are ‘*person\_id,lastname,firstname,sport’. W*e are *bucketing* the table by ‘*person\_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’*

**Command :** *describe formatted sports;*

Will give the test format,serde library and bucket columns etc.

****

## Inserting Data into a Hive Table

## Hive Query:

*INSERT INTO table sports values*

*(1,'Messi','Lionel','Football'),*

*(2,'Federer','Roger','Tennis'),*

*(3,'Tendulkar','Sachin','Cricket'),*

*(4,'Bolt','Usain','Sprint'),*

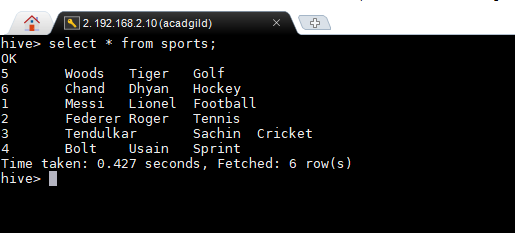
*(5,'Woods','Tiger','Golf'),*

*(6,'Chand','Dhyan','Hockey');*

## 

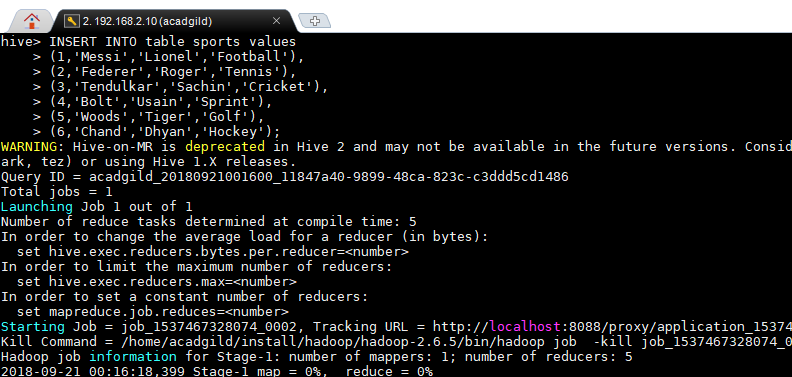
Now, we have successfully inserted the data into the Hive table.

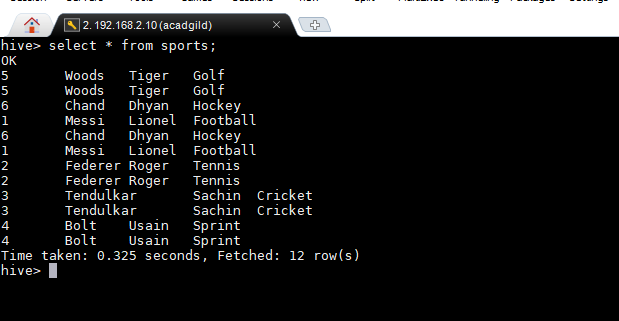
The contents of the table can be viewed using the command **select \* from sports**

****

From the above image, we can see that the data has been inserted successfully into the table.

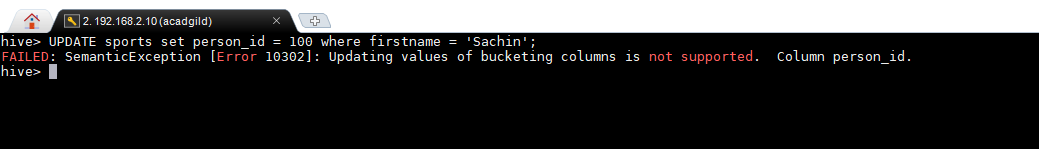
Now if we try to re-insert the same data again, it will be appended to the previous data as shown below:

****

****

## Updating the Data in Hive Table

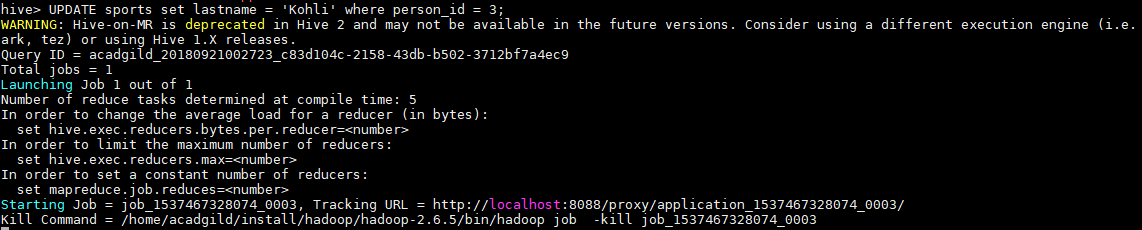
*Command: UPDATE sports set person\_id = 100 where firstname = 'Sachin';*

****

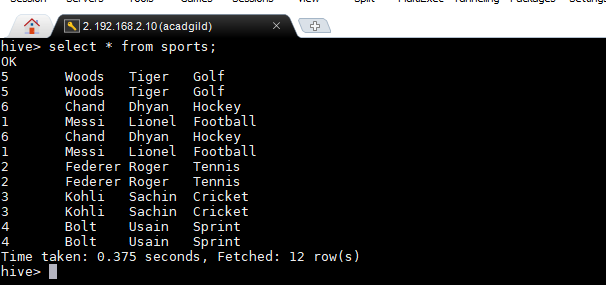
From the above image, we can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed.

**Now let’s perform the update operation on Non bucketed column**

*Command: UPDATE sports set lastname = ‘Kohli’ where person\_id =3;*

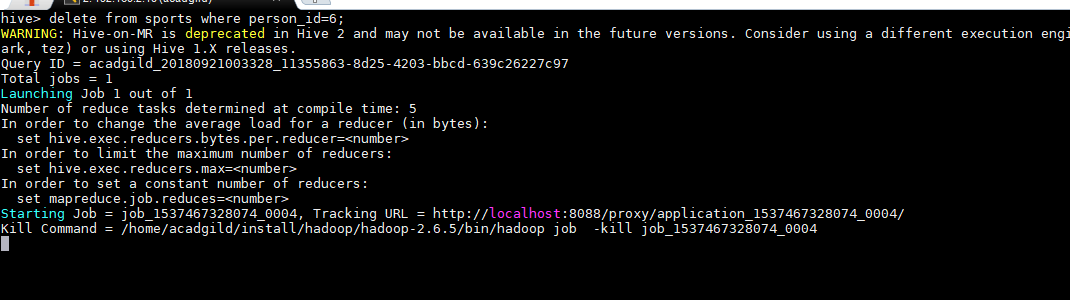
****

The updated data can be checked using the command ***select \* from sports.***

****

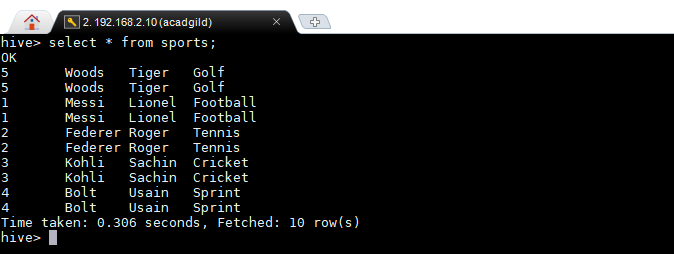
## Deleting a Row from Hive Table

**Command :** *delete from sports where person\_id=6;*

****

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

We have now successfully deleted a row from the Hive table. This can be checked using the command select \* from sports.

**

We can see that there is no row with**person\_id =6**. This means that we have successfully deleted the row from the Hive table.

This is how the transactions or row-wise operations are performed in Hive.