**Associated Data Files**

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

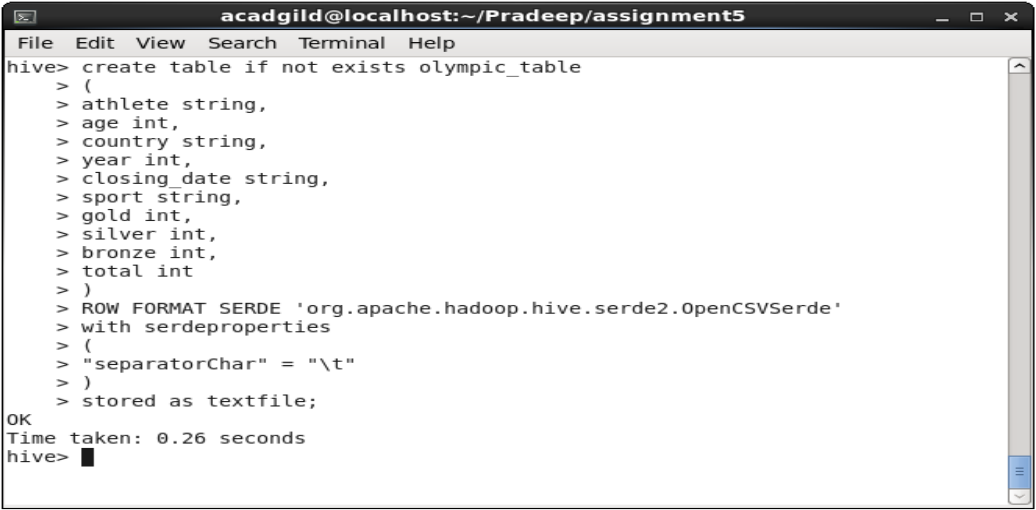
Silver Medals: No. of Silver medals

Bronze Medals: No. of Bronze medals

Total Medals: Consists of total no. of medals

First we will prepare the data set ready to perform all the actions asked in the problem statement below.

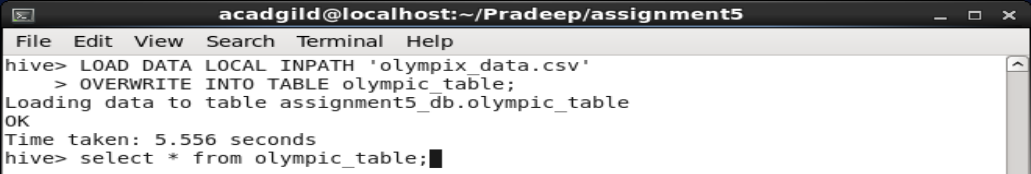
* Hive commands to create database, use the database just created and create a table



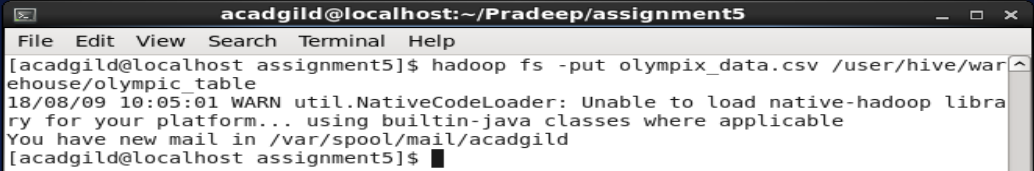
* **There are two ways to load the data into this table that we have created**
  1. By using the LOAD DATA command as following command:
     + LOAD DATA LOCAL INPATH ‘olympix\_data.csv’

INTO TABLE olympic\_table;

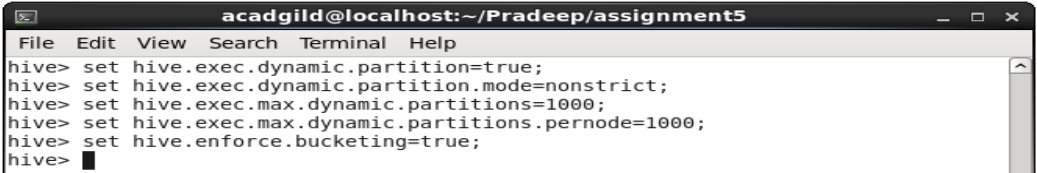
* + - Now verify whether the data is loaded correctly or not by the following command:
    - Select \* from Olympic\_table;



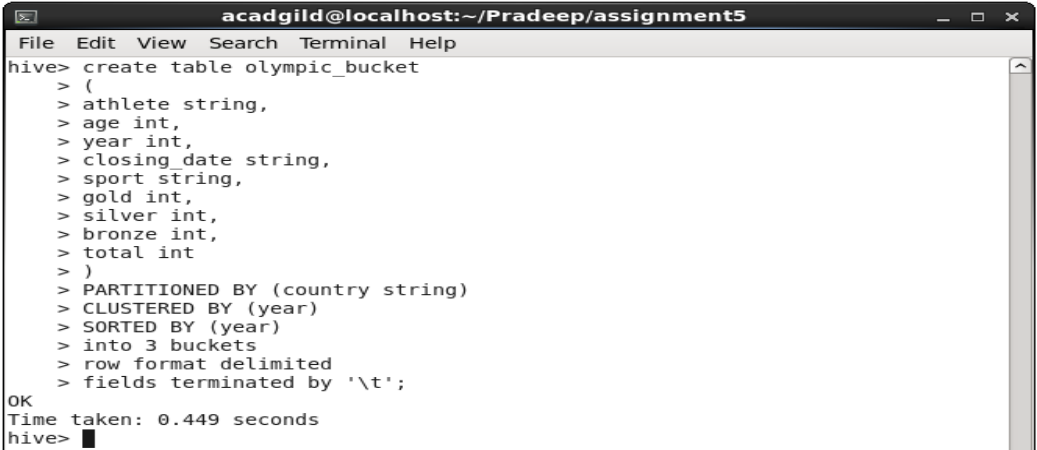
* 1. By using the hadoop filesystem **put** command copy the olympix\_data.csv to the path to which the created table is pointing to:
     + To know where to copy the input file execute the following command:
       - Show create table Olympic\_table;
     + Now in the output of the above command look for **LOCATION** item and note the path
       - ‘hdfs://localhost:8020/usr/hive/warehouse/assignment5\_db.db/olympic\_table’
     + So now you can put the olympix\_data.csv to above path as follows:
       - Hadoop fs –put olympix\_data.csv /usr/hive/warehouse/assignment5\_db.db/Olympic\_table;



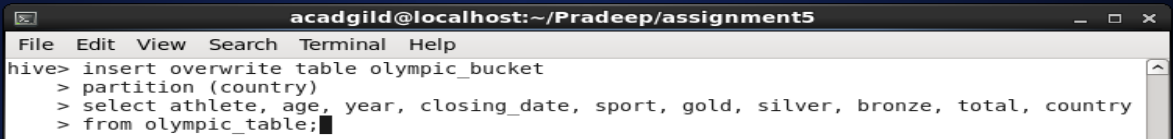
* **Now we will create PARTITION AND BUCKETING operation:**
  1. Hive partition divides table into number of **partitions** and these partitions can be further subdivided into more manageable parts known as **Buckets** or **Clusters**.
  2. In Hive Partition, each partition will be created as directory. But in Hive Buckets, each bucket will be created as file.
* **Setting the Below Properties in Hive Command Line Before Proceeding Further for Bucketing Scripts:**
  1. Set hive.exec.dynamic.partition=true;
  2. Set hive.exec.dynamic.partition.mode=nonstrict;
  3. Set hive.exec.max.dynamic.partitions=1000;
  4. Set hive.exec.max.dynamic.partitions.pernode=1000;
  5. Set hive.enforce.bucketing=true;



* **Then create the table with these optimization technics as shown in the below screen shot:**



* **Now insert the contents of the Olympic\_table into the Olympic\_bucket table with the following command:**

****

**Here I have struggled to remember that partition column should be at the last while loading the data from original table if you selected partition column other than last column of the actual table.**

**i.e.**

insert overwrite table Olympic\_bucket

partition (**country**)

select *all-the-other-columns-separated-by-comma,* **country**

from olympic\_table;

if the partition column is the last column in the actual table then this won’t be necessary by default it will take as partition column. You can do the following:

insert overwrite table Olympic\_bucket

partition (country)

select \* from Olympic\_table;

**Problem Statement**

**Task 1.1**

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

**Script:**

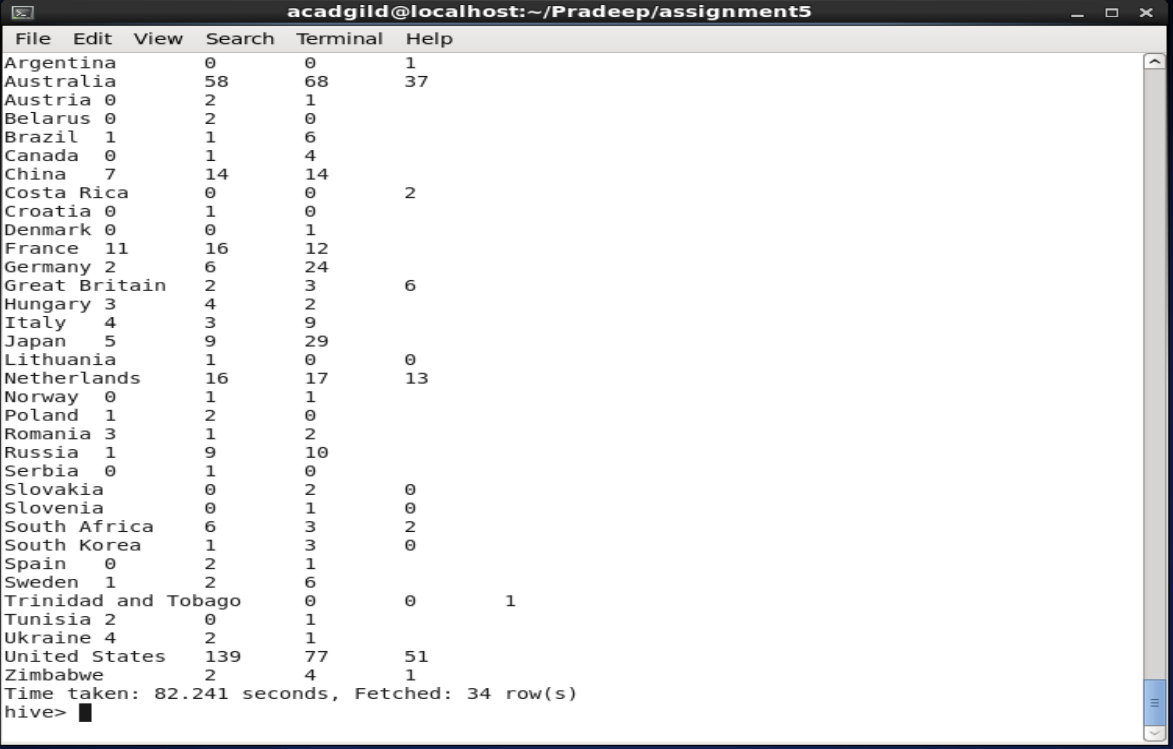
Select country, sum(gold), sum(silver), sum(bronze)

From Olympic\_bucket

Where sport like ‘Swimming’

Group by country;

**OUTPUT:**

****

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

**Script:**

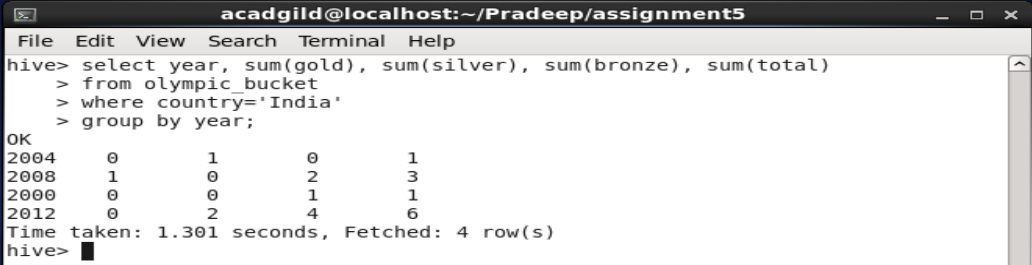
Select year, sum(gold), sum(silver), sum(bronze), sum(total)

From Olympic\_bucket

Where country=‘India’

Group by year;

**OUTPUT:**



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

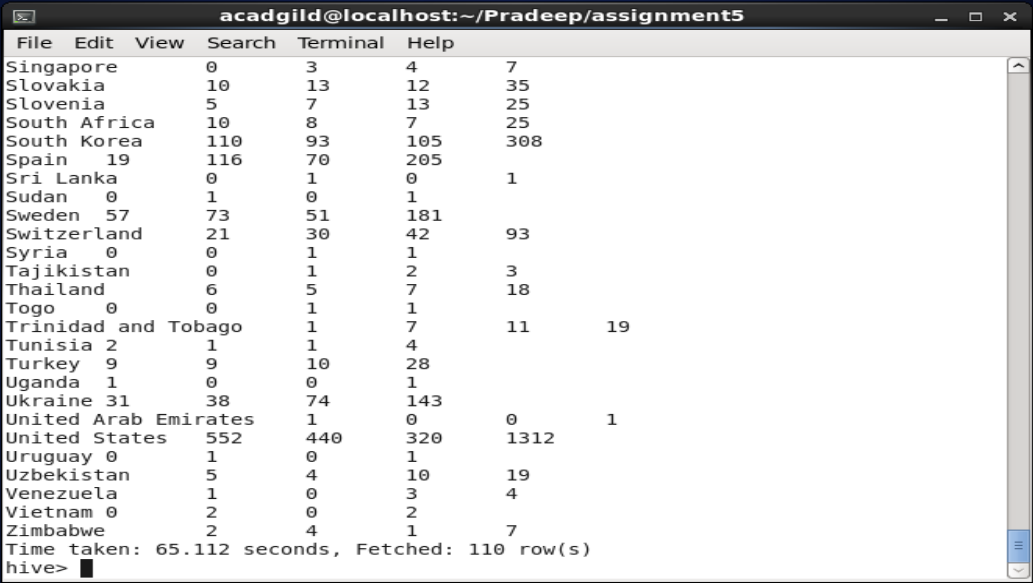
**Script:**

Select country, sum(gold), sum(silver), sum(bronze), sum(total)

From Olympic\_bucket

Group by country;

**OUTPUT:**



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

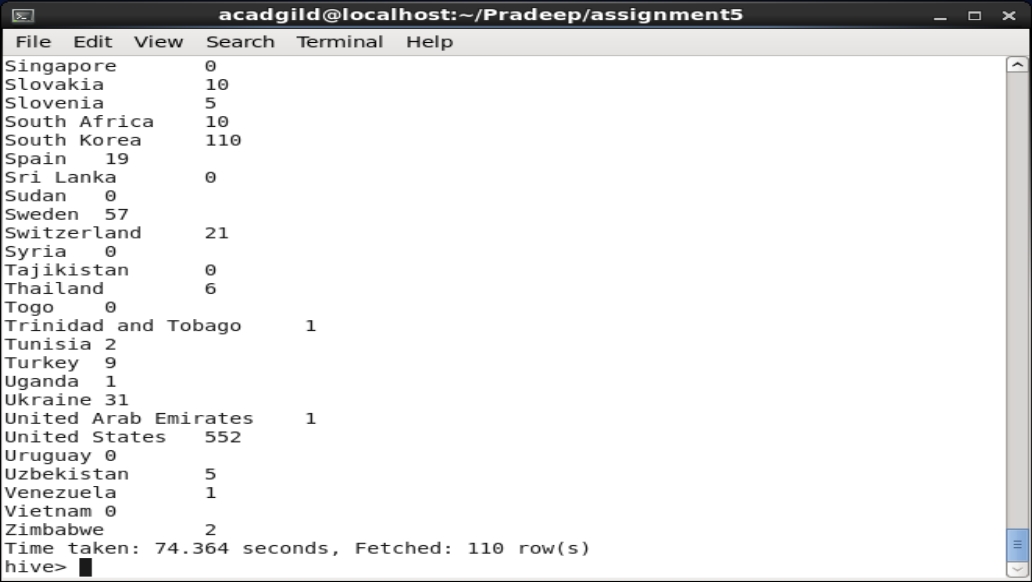
**Script:**

Select country, sum(gold)

From Olympic\_bucket

Group by country;

**OUTPUT:**



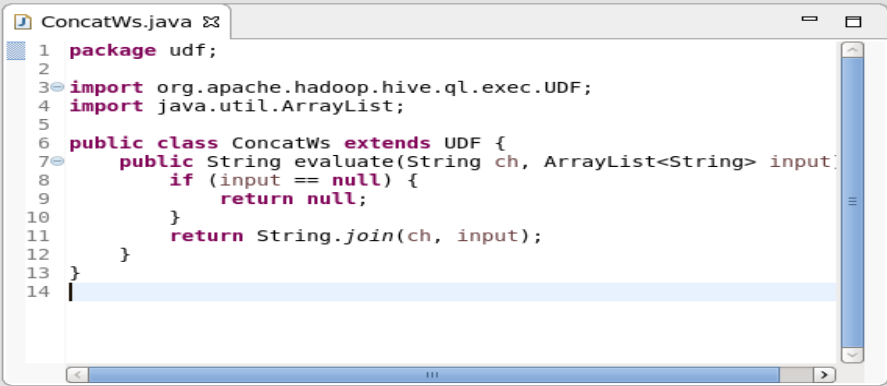
**Task 1.2**

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.

**Java Program:**

Following is the java program that will take two arguments: separator string as ch and **array list of string as input** and returns the joined string to the calling command:



Then I have created a jar file from the above program with the name **ConcatWS.jar**

Now I have created the emp\_skills.txt file which contains the following information:



Now I have created a table to store the details of employee with the following command:

* Create table emp
* (
* eId int,
* eName String,
* eSkills array<String>
* )

Row format delimited

Fields terminated by ‘\t’

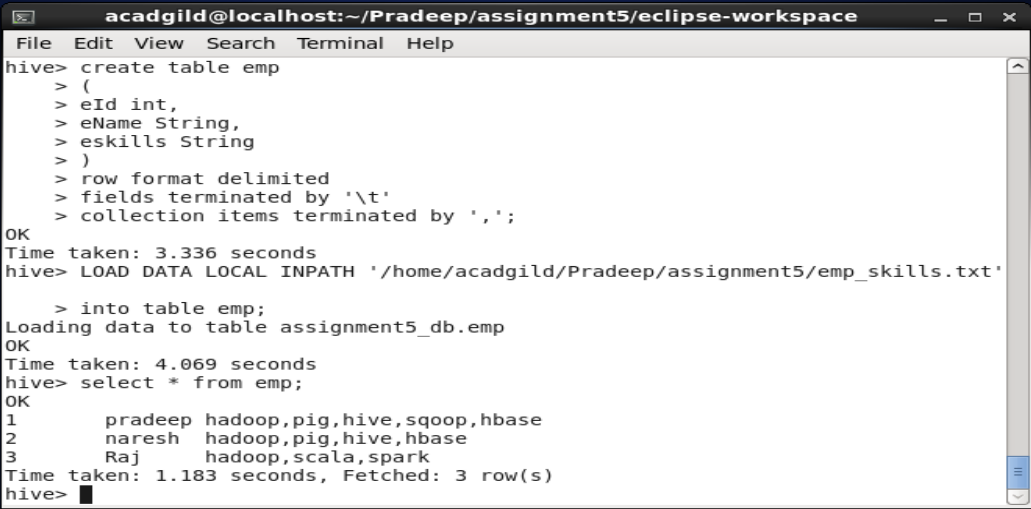
Collection items terminated by ‘,’;

Then I have loaded the data into the emp table with the following command:

* LOAD DATA LOCAL INPATH ‘/home/acadgild/Pradeep/assignment5/emp\_skills.txt’
* Into table emp;

Then I have verified whether the data is loaded as per expectations with the following command:

* Select \* from emp;

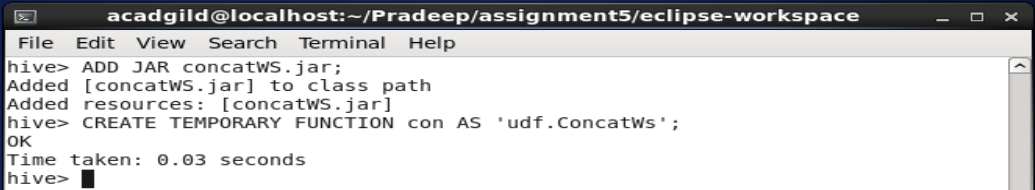


Then I have registered this jar to existing hive session with the following command:

* ADD JAR ConcatWS.jar;

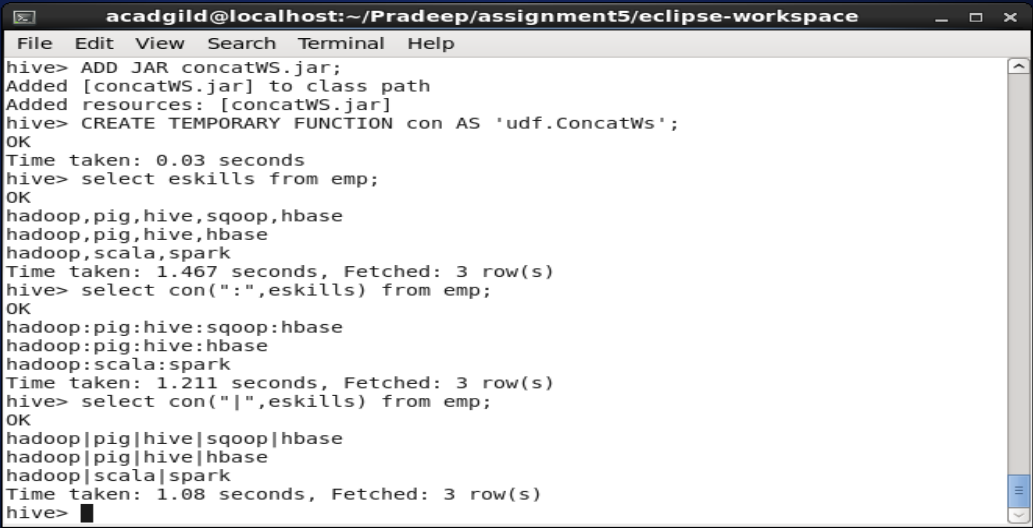
Then I have created a temporary function with the following command:

* CREATE TEMPORARY FUNCTION con AS ‘udf.ConcatWs’;



Then I have executed the following command to work with udf function:

* Select eskills from emp;
* Select con(“:”,eskills) from emp;
* Select con(“|”,eskills) from emp;



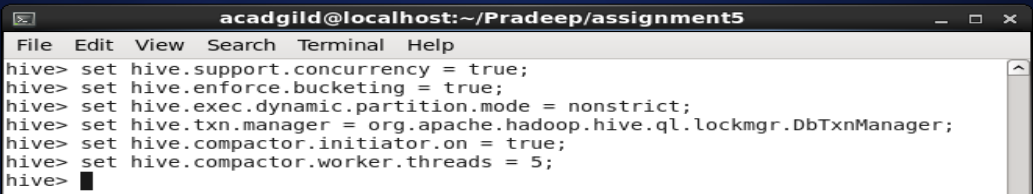
As you can see in the above screen shot I am able to add the different separators for the eskills string array that I am passing to the Hive udf function.

**Task 1.3**

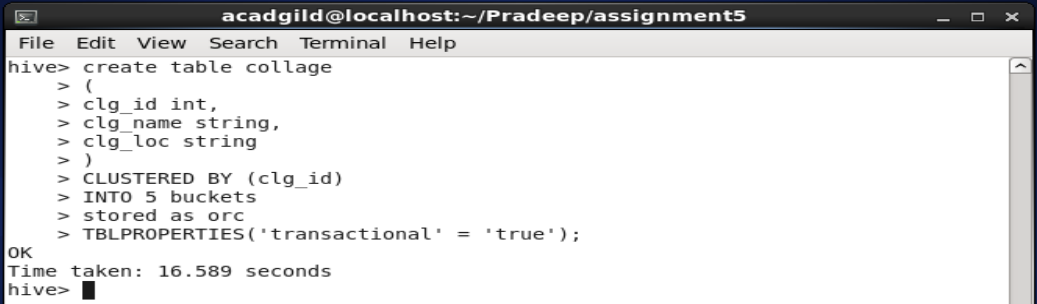
Link: https://acadgild.com/blog/transactions-in-hive/

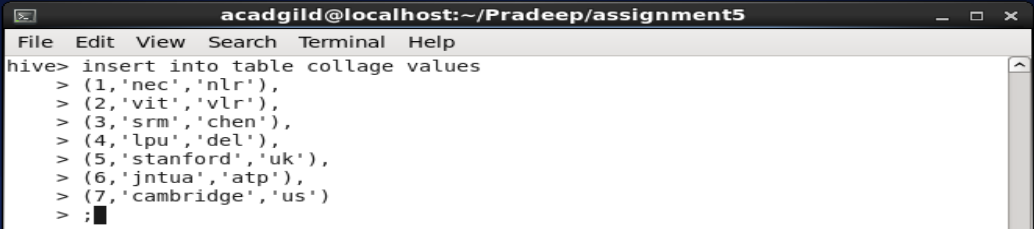
Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

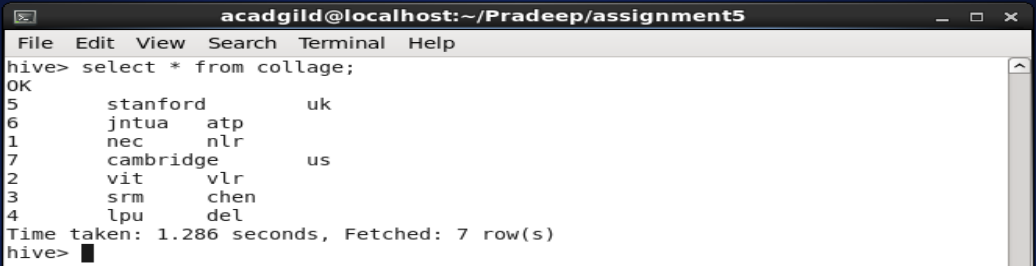
**Settings:**



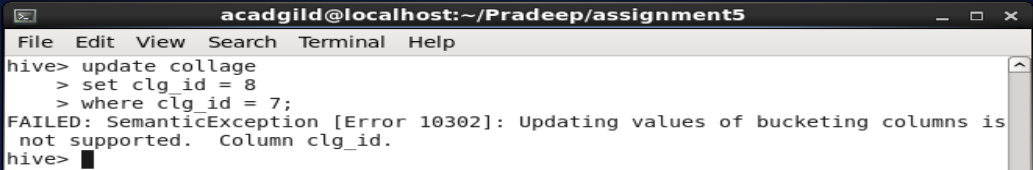
**Script:**

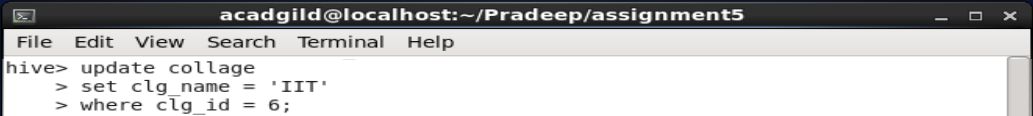
****

****

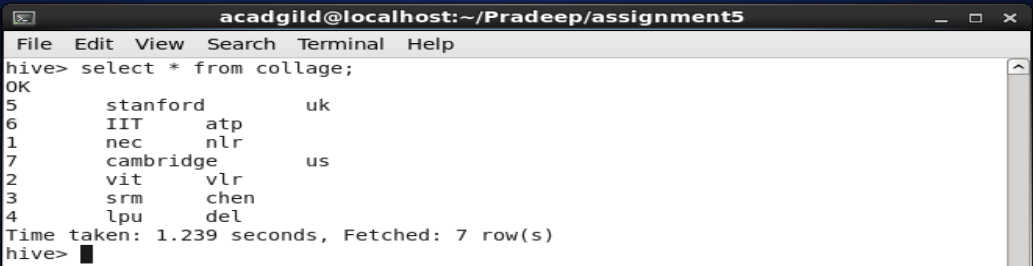
****

**Updating the Data in Hive Table**

****

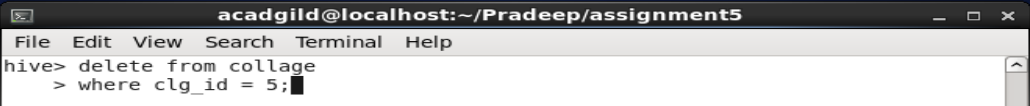
****

**OUTPUT:**

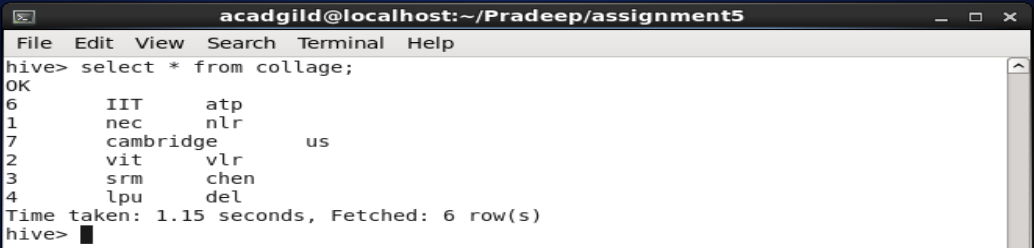
****

**Deleting a Row from Hive Table**

**Script:**



**OUTPUT:**



**Task 2.1**

Answer in your own words with example.

1. What is NoSQL data base?

NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

1. How does data get stored in NoSQL database?

Data Storage is not based on a single data model. Most outstanding ones are key-value pair, graph, document, and columnar.

1. What is a column family in HBase?

A column family defines shared features to all columns that are created within them (think of it almost as a sub-table within your larger table). You will notice that HBase columns are composed of a combination of the column family and column qualifier (or column key): 'family:qualifier'  
  
Where the qualifier can be an arbitrary array of bytes, the column family has to be composed of printable characters. Also, on HDFS, the column family is what is stored in human-readable format, example: '/hbase/table/region/<colfamX>'  
  
Remember - all columns families must be created up-front whereas columns can be added on the fly. Hence understanding the design of your data access pattern is crucial when first building your HBase instance.

1. How many maximum number of columns can be added to HBase table?

It has been recommended to keep the number of column families under three. But there is no magic number like this. Why not two? Why not four? Technically, HBase can manage more than three of four column families. However, you need to understand how column families work to make the best use of them.

1. Why columns are not defined at the time of table creation in HBase?

An HBase table is made of column families which are the logical and physical grouping of columns. The columns in one family are stored separately from the columns in another family.

A single column family contains one or more columns, Column families must be defined at table creation time but columns can be added dynamically after table creation (if an insert statement states a column that does not exist for a column family it will create it).

1. How does data get managed in HBase?

The data in the HBASE is managed in such a way that,

* **WAL:** Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.
* **BlockCache:** is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.
* **MemStore:** is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.
* **Hfiles** store the rows as sorted KeyValues on disk.

All the above are sub components of Region server which manages the data in the HBASE.

1. What happens internally when new data gets inserted into HBase table?

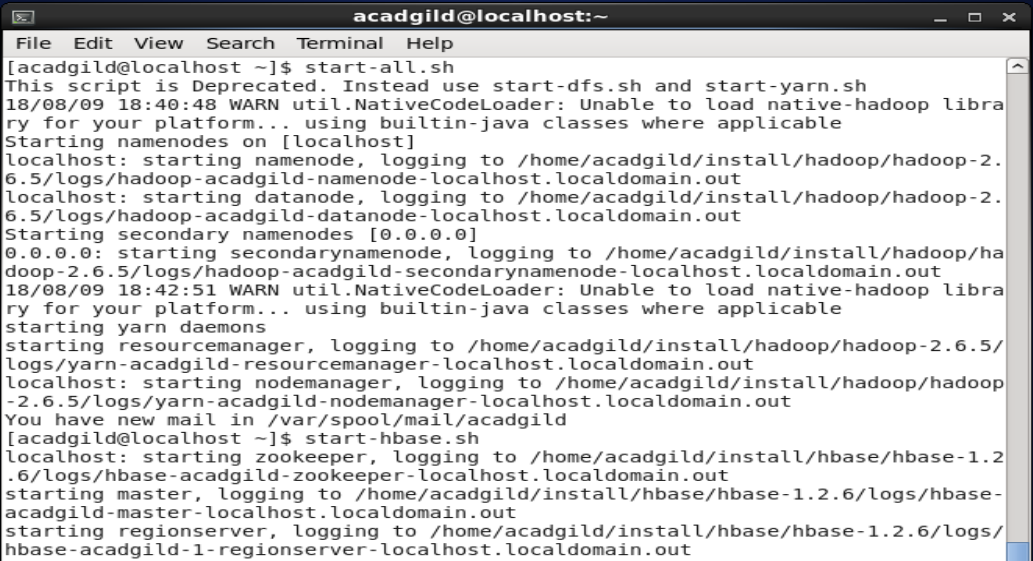
* The client gets the Region server that hosts the META table from ZooKeeper.
* The client will query the .META. Server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location.
* It will get the Row from the corresponding Region Server.

For future reads, the client uses the cache to retrieve the META location and previously read row keys. Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will re-query and update the cache.

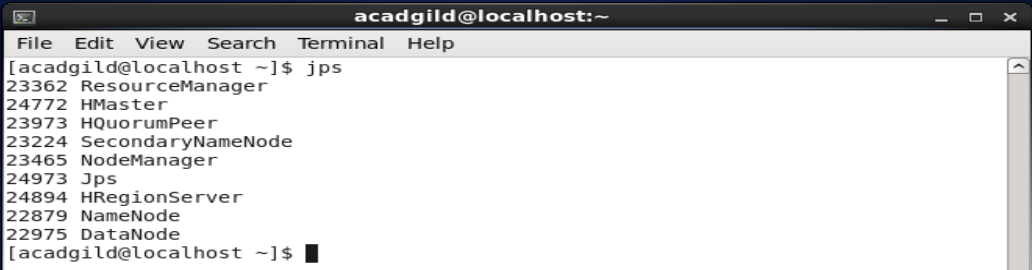
**Task 2.2**

Before working with HBase actual commands we need to start the Hadoop and HBase services with the following commands:

* Start-all.sh
* Start-hbase.sh



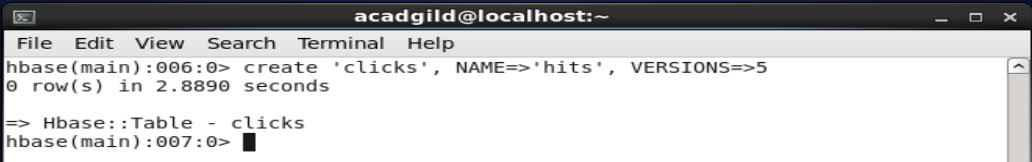
* Jps command to check whether the services started or not



1. Create an HBase table named 'clicks' with a column family 'hits' such that it should be able to store last 5 values of qualifiers inside 'hits' column family.

Following is the command to create a table ‘clicks’ with column family ‘hits’ and having 5 versions:

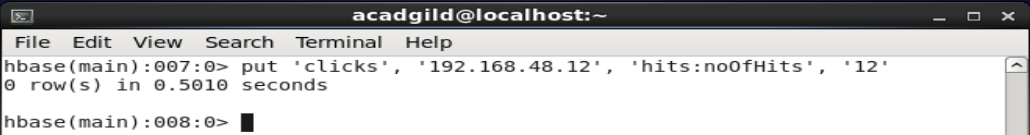
* Create ‘clicks’, NAME=>‘hits’, VERSIONS=>5



2. Add few records in the table and update some of them. Use IP Address as row-key. Scan the table to view if all the previous versions are getting displayed.

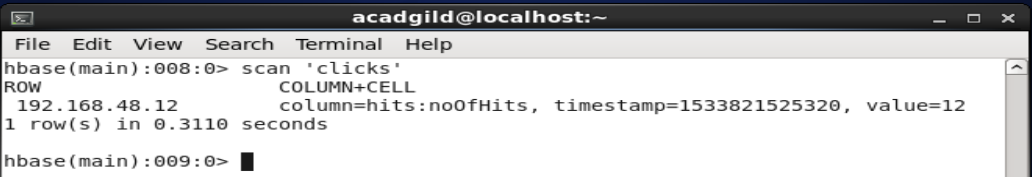
To add records in the clicks table we use below put command and we use row ip address 192.168.48.12 as row key:

* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘12’



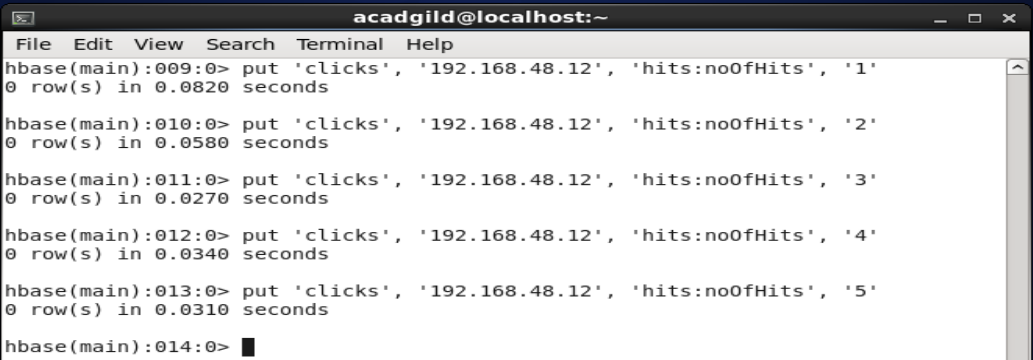
To view the contents of the table use scan command as show below:

* Scan ‘clicks’



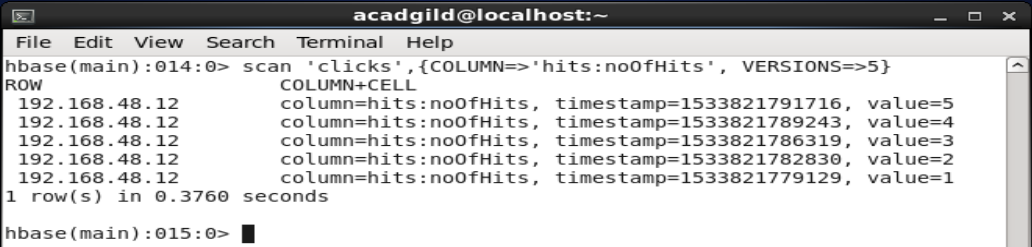
Now we will try to update the same row key for 5 times and check whether we are getting all the version values or not with the help of following commands:

* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘1’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘2’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘3’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘4’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘5’



Now we are going to see the last 5 qualifiers inside hits column family with the help of the following command:

* Scan ‘clicks’ ,{COLUMN=>'hits:NoOfHits',VERSIONS=>5}



Thank you,

Pradeep