Assignment Day 9

INPUT Dataset(Task 1):

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

Process:

```
CREATE TABLE olympics_text
(
Athlete STRING,
Age INT,
Country STRING,
Year DOUBLE,
Closing_Date STRING,
Sport STRING,
Gold_Medals INT,
Silver_Medals INT,
Bronze_Medals INT,
Total_Medals INT
```

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

Explanation:

Creating a simple Text Table wherein each fields are delimited by a tab.

LOAD DATA LOCAL INPATH
'/home/acadgild/Desktop/TestHadoop/hive/olympix_data.csv'

INTO TABLE olympics_text;

Explanation:

Loading data from local into the olympics_text Table.

DESC FORMATTED olympics_text;

Explanation:

Checking the detailed properties of the table created.

dfs -du -h dfs://localhost:8020/user/hive/warehouse/custom.db/olympics_text;

506.5 K

Explanation:

Checking the size occupied by newly created table.

```
CREATE TABLE olympics_orc
(
Athlete STRING,
Age INT,
Country STRING,
Year DOUBLE,
Closing_Date STRING,
Sport STRING,
Gold_Medals INT,
Silver_Medals INT,
Bronze_Medals INT,
Total_Medals INT
)
STORED AS ORC;
```

Explanation:

Creating an ORC Table with the same fields as simple text table.

NOTE: An ORC table allows to manage space & makes querying data much more efficient & effective.

```
FROM olympics_text
INSERT INTO olympics_orc SELECT *;
```

Explanation:

Loading Data from simple table to convert them into an orc table.

SELECT * FROM olympics_orc **LIMIT** 10;

Explanation:

Checking top 10 rows of the ORC table created.

dfs -du -h hdfs://localhost:8020/user/hive/warehouse/custom.db/olympics_orc;

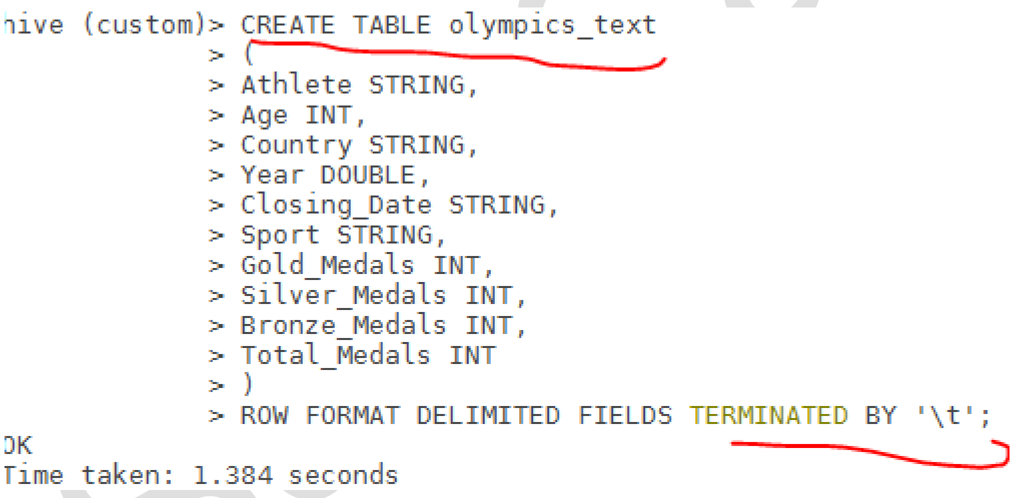
87.6 K

Explanation:

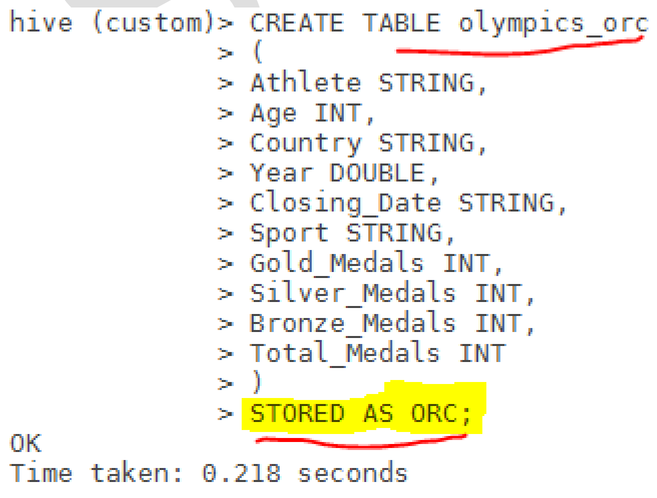
Checking the space occupied by ORC table for the same amount of data as that of a simple table.

Note: It can be observed that, there is almost five fold difference between both the table's data

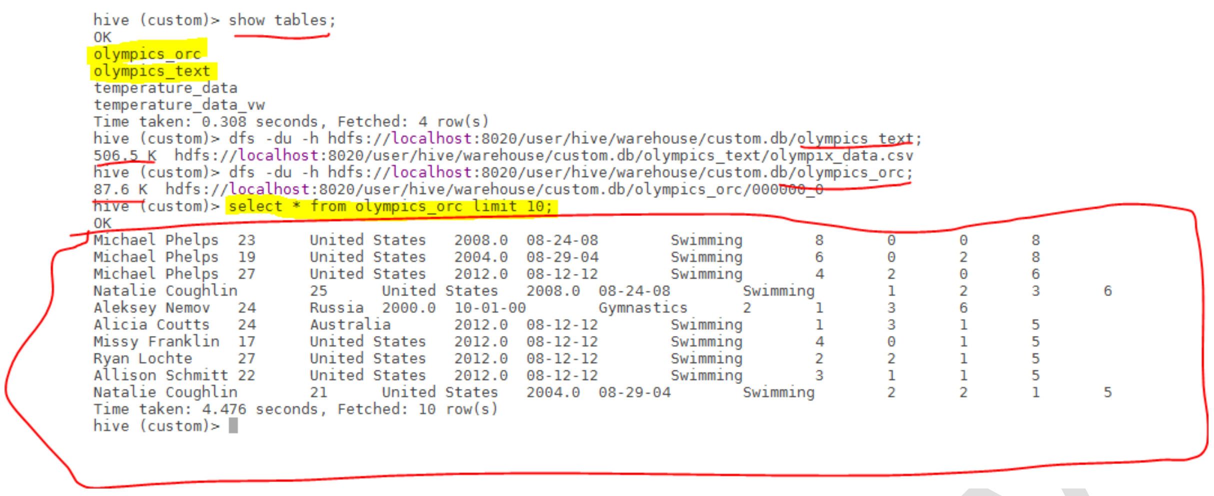
ScreenShot:



```
hive (custom)> CREATE TABLE olympics_text
             > (
             > Athlete STRING,
             > Age INT,
             > Country STRING,
             > Year DOUBLE,
             > Closing_Date STRING,
             > Sport STRING,
             > Gold_Medals INT,
             > Silver_Medals INT,
             > Bronze_Medals INT,
             > Total_Medals INT
             > )
             > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
OK
Time taken: 1.384 seconds
```



```
hive (custom)> CREATE TABLE olympics_orc
             > (
             > Athlete STRING,
             > Age INT,
             > Country STRING,
             > Year DOUBLE,
             > Closing_Date STRING,
             > Sport STRING,
             > Gold_Medals INT,
             > Silver_Medals INT,
             > Bronze_Medals INT,
             > Total_Medals INT
             > )
             > STORED AS ORC;
OK
Time taken: 0.218 seconds
```



```
hive (custom)> show tables;
OK
olympics_orc
olympics_text
temperature_data
temperature_data_vw
Time taken: 0.308 seconds, Fetched: 4 row(s)
hive (custom)> dfs -du -h hdfs://localhost:8020/user/hive/warehouse/custom.db/olympics_text;
506.5 K  hdfs://localhost:8020/user/hive/warehouse/custom.db/olympics_text/olympix_data.csv
hive (custom)> dfs -du -h hdfs://localhost:8020/user/hive/warehouse/custom.db/olympics_orc;
87.6 K  hdfs://localhost:8020/user/hive/warehouse/custom.db/olympics_orc/000000_0
hive (custom)> select * from olympics_orc limit 10;
OK
Michael Phelps  23      United States   2008.0  08-24-08        Swimming        8       0       0       8
Michael Phelps  19      United States   2004.0  08-29-04        Swimming        6       0       2       8
Michael Phelps  27      United States   2012.0  08-12-12        Swimming        4       2       0       6
Natalie Coughlin        25      United States   2008.0  08-24-08        Swimming        1       2       3       6
Aleksey Nemov   24      Russia  2000.0  10-01-00        Gymnastics      2       1       3       6
Alicia Coutts   24      Australia       2012.0  08-12-12        Swimming        1       3       1       5
Missy Franklin  17      United States   2012.0  08-12-12        Swimming        4       0       1       5
Ryan Lochte     27      United States   2012.0  08-12-12        Swimming        2       2       1       5
Allison Schmitt 22      United States   2012.0  08-12-12        Swimming        3       1       1       5
Natalie Coughlin        21      United States   2004.0  08-29-04        Swimming        2       2       1       5
Time taken: 4.476 seconds, Fetched: 10 row(s)
hive (custom)> 
```

Task 1:

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

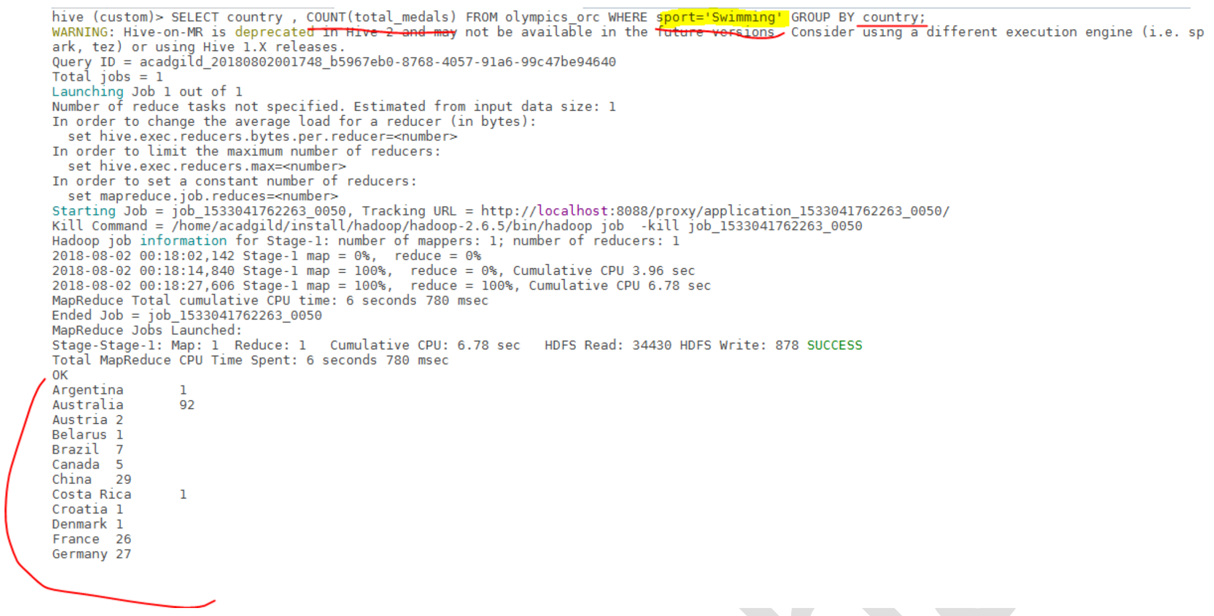
Ans:

SELECT country , **COUNT**(total_medals) **FROM** olympics_orc **WHERE** sport='Swimming' **GROUP BY** country;

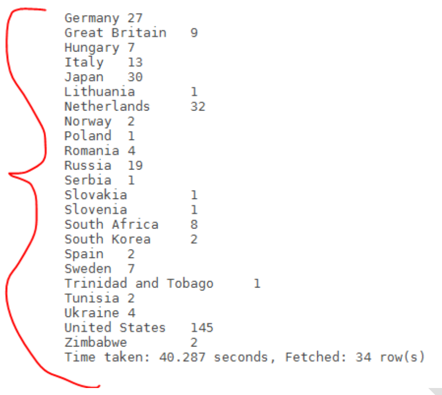
Explanation:

(Grouping by country & counting the total medal won by each country in swimming)

ScreenShot:



```
hive (custom)> SELECT country , COUNT(total_medals) FROM olympics_orc WHERE sport='Swimming' GROUP BY country;
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. sp
ark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180802001748_b5967eb0-8768-4057-91a6-99c47be94640
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
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_1533041762263_0050, Tracking URL = http://localhost:8088/proxy/application_1533041762263_0050/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533041762263_0050
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-02 00:18:02,142 Stage-1 map = 0%,  reduce = 0%
2018-08-02 00:18:14,840 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.96 sec
2018-08-02 00:18:27,606 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.78 sec
MapReduce Total cumulative CPU time: 6 seconds 780 msec
Ended Job = job_1533041762263_0050
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.78 sec   HDFS Read: 34430 HDFS Write: 878 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 780 msec
OK
Argentina       1
Australia       92
Austria 2
Belarus 1
Brazil  7
Canada  5
China   29
Costa Rica      1
Croatia 1
Denmark 1
France  26
Germany 27
```



```
Germany 27
Great Britain   9
Hungary 7
Italy   13
Japan   30
Lithuania       1
Netherlands     32
Norway  2
Poland  1
Romania 4
Russia  19
Serbia  1
Slovakia        1
Slovenia        1
South Africa    8
South Korea     2
Spain   2
Sweden  7
Trinidad and Tobago     1
Tunisia 2
Ukraine 4
United States   145
Zimbabwe        2
Time taken: 40.287 seconds, Fetched: 34 row(s)
```

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

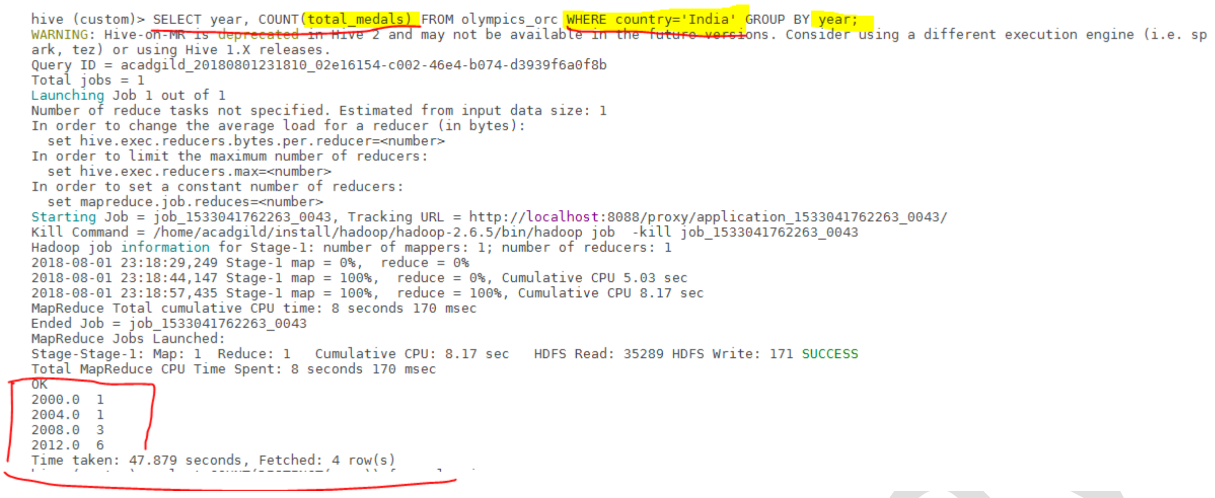
Ans:

SELECT year, **COUNT**(total_medals) **FROM** olympics_orc **WHERE** country='India' **GROUP BY** year;

Explanation:

(Selecting year & counting total number of medals won by INDIA for a particular year)

ScreenShot:



```
hive (custom)> SELECT year, COUNT(total_medals) FROM olympics_orc WHERE country='India' GROUP BY year;
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. sp
ark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180801231810_02e16154-c002-46e4-b074-d3939f6a0f8b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
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_1533041762263_0043, Tracking URL = http://localhost:8088/proxy/application_1533041762263_0043/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533041762263_0043
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-01 23:18:29,249 Stage-1 map = 0%,  reduce = 0%
2018-08-01 23:18:44,147 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.03 sec
2018-08-01 23:18:57,435 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.17 sec
MapReduce Total cumulative CPU time: 8 seconds 170 msec
Ended Job = job_1533041762263_0043
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.17 sec   HDFS Read: 35289 HDFS Write: 171 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 170 msec
OK
2000.0  1
2004.0  1
2008.0  3
2012.0  6
Time taken: 47.879 seconds, Fetched: 4 row(s)
```

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

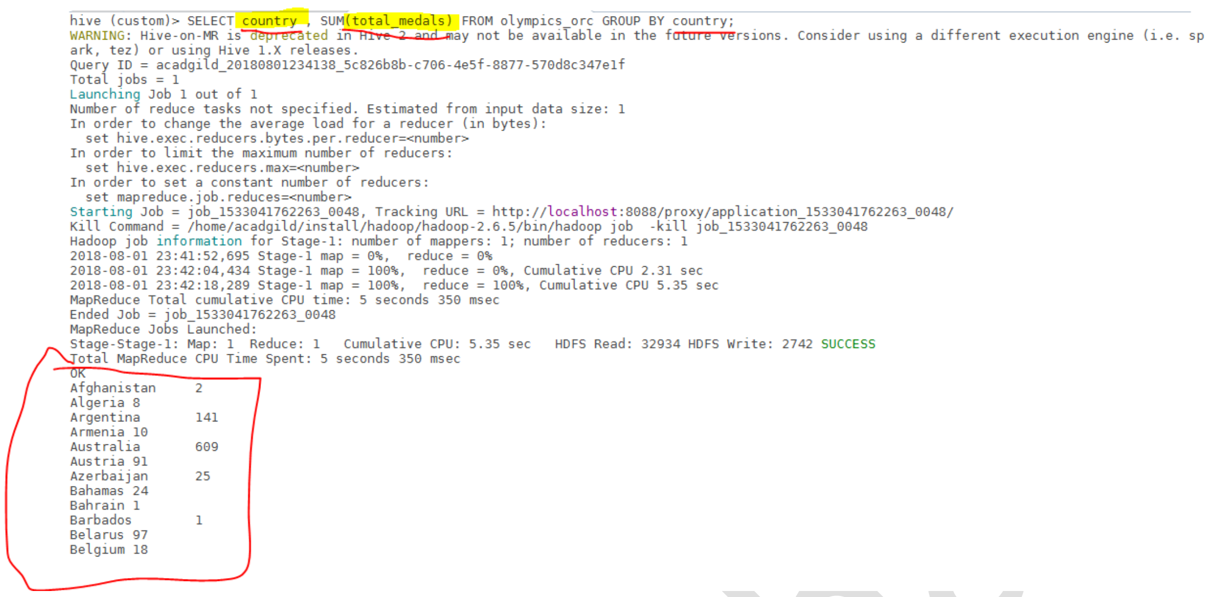
Ans:

SELECT country , **SUM**(total_medals) **FROM** olympics_orc **GROUP BY country**;

Explanation:

(Selecting all the countries & adding up the total medals won by each & displaying the result country wise in a sorted manner).

ScreenShot:

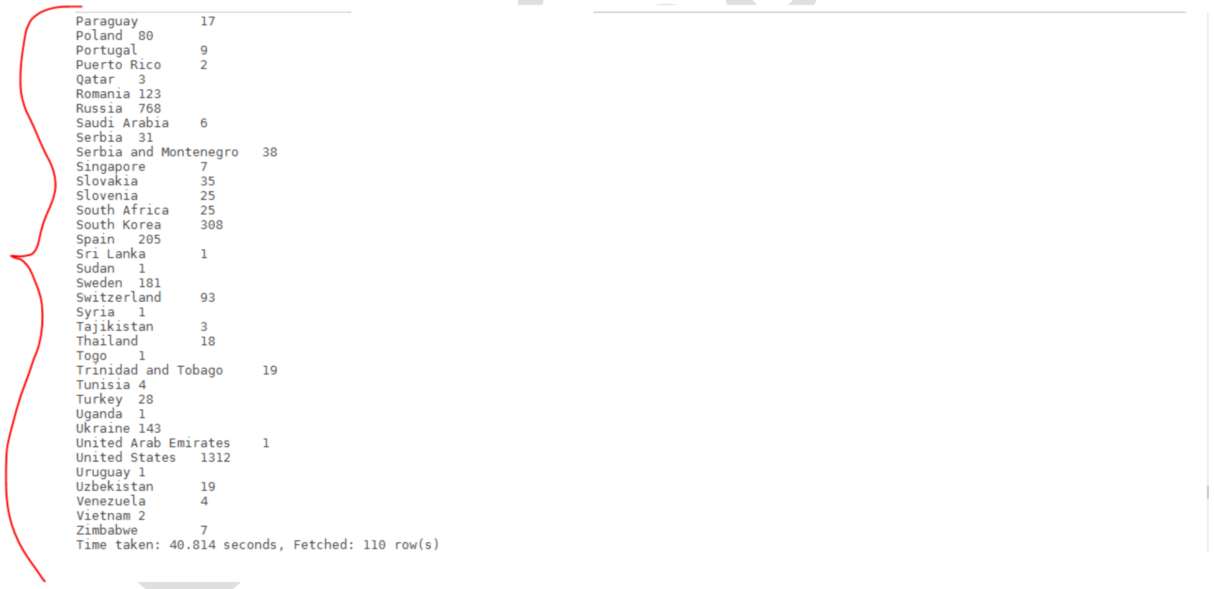


```
hive (custom)> SELECT country , SUM(total_medals) FROM olympics_orc GROUP BY country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. sp
ark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180801234138_5c826b8b-c706-4e5f-8877-570d8c347e1f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
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_1533041762263_0048, Tracking URL = http://localhost:8088/proxy/application_1533041762263_0048/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533041762263_0048
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-01 23:41:52,695 Stage-1 map = 0%,  reduce = 0%
2018-08-01 23:42:04,434 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.31 sec
2018-08-01 23:42:18,289 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.35 sec
MapReduce Total cumulative CPU time: 5 seconds 350 msec
Ended Job = job_1533041762263_0048
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.35 sec   HDFS Read: 32934 HDFS Write: 2742 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 350 msec
OK
Afghanistan     2
Algeria 8
Argentina       141
Armenia 10
Australia       609
Austria 91
Azerbaijan      25
Bahamas 24
Bahrain 1
Barbados        1
Belarus 97
Belgium 18
```

•

•

•



```
Paraguay        17
Poland  80
Portugal        9
Puerto Rico     2
Qatar   3
Romania 123
Russia  768
Saudi Arabia    6
Serbia  31
Serbia and Montenegro   38
Singapore       7
Slovakia        35
Slovenia        25
South Africa    25
South Korea     308
Spain   205
Sri Lanka       1
Sudan   1
Sweden  181
Switzerland     93
Syria   1
Tajikistan      3
Thailand        18
Togo    1
Trinidad and Tobago     19
Tunisia 4
Turkey  28
Uganda  1
Ukraine 143
United Arab Emirates    1
United States   1312
Uruguay 1
Uzbekistan      19
Venezuela       4
Vietnam 2
Zimbabwe        7
Time taken: 40.814 seconds, Fetched: 110 row(s)
```

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

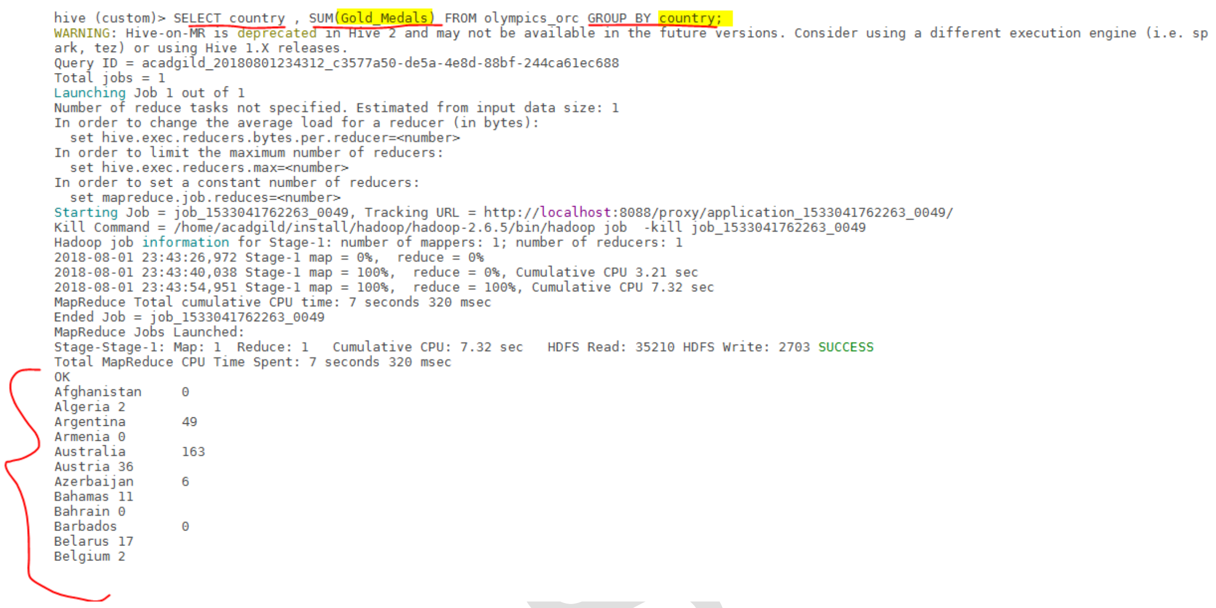
Ans:

SELECT country , **SUM**(Gold_medals) **FROM** olympics_orc **GROUP BY** **country**;

Explanation:

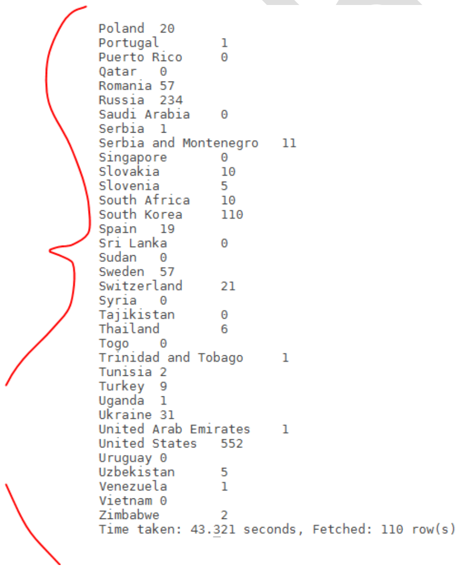
(Selecting all the countries & adding up the total Gold medals won by each & displaying the result country wise in a sorted manner).

ScreenShot:



```
hive (custom)> SELECT country , SUM(Gold_Medals) FROM olympics_orc GROUP BY country;
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. sp
ark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180801234312_c3577a50-de5a-4e8d-88bf-244ca61ec688
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
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_1533041762263_0049, Tracking URL = http://localhost:8088/proxy/application_1533041762263_0049/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533041762263_0049
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-08-01 23:43:26,972 Stage-1 map = 0%,  reduce = 0%
2018-08-01 23:43:40,038 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.21 sec
2018-08-01 23:43:54,951 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.32 sec
MapReduce Total cumulative CPU time: 7 seconds 320 msec
Ended Job = job_1533041762263_0049
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.32 sec   HDFS Read: 35210 HDFS Write: 2703 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 320 msec
OK
Afghanistan     0
Algeria 2
Argentina       49
Armenia 0
Australia       163
Austria 36
Azerbaijan      6
Bahamas 11
Bahrain 0
Barbados        0
Belarus 17
Belgium 2
```

-
-
-



```
Poland  20
Portugal        1
Puerto Rico     0
Qatar   0
Romania 57
Russia  234
Saudi Arabia    0
Serbia  1
Serbia and Montenegro   11
Singapore       0
Slovakia        10
Slovenia        5
South Africa    10
South Korea     110
Spain   19
Sri Lanka       0
Sudan   0
Sweden  57
Switzerland     21
Syria   0
Tajikistan      0
Thailand        6
Togo    0
Trinidad and Tobago     1
Tunisia 2
Turkey  9
Uganda  1
Ukraine 31
United Arab Emirates    1
United States   552
Uruguay 0
Uzbekistan      5
Venezuela       1
Vietnam 0
Zimbabwe        2
Time taken: 43.321 seconds, Fetched: 110 row(s)
```

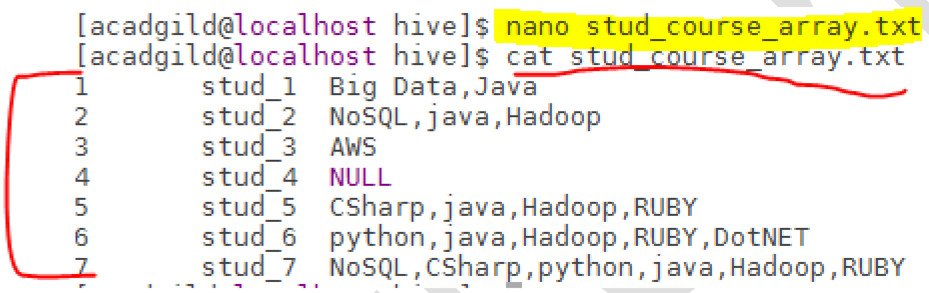
Task 2:

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:

Creating a text file stud_course_array.txt in local & populating the following data to load into a table.

ScreenShot:



```
[acadgild@localhost hive]$ nano stud_course_array.txt
[acadgild@localhost hive]$ cat stud_course_array.txt
1       stud_1  Big Data,Java
2       stud_2  NoSQL,java,Hadoop
3       stud_3  AWS
4       stud_4  NULL
5       stud_5  CSharp,java,Hadoop,RUBY
6       stud_6  python,java,Hadoop,RUBY,DotNET
7       stud_7  NoSQL,CSharp,python,java,Hadoop,RUBY
```

Steps followed for creating a Scenario (Commands in Screenshot):

USE custom;

Explanation:

(Using custom database.)

CREATE table stud_course

(

stud_id int,

stud_name string,

course array<string>

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

COLLECTION ITEMS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED as textfile;

Explanation:

(Creating table stud_course as shown below.)

LOAD DATA LOCAL INPATH
'/home/acadgild/Desktop/TestHadoop/hive/stud_course_array.txt'
INTO table stud_course;

Explanation:

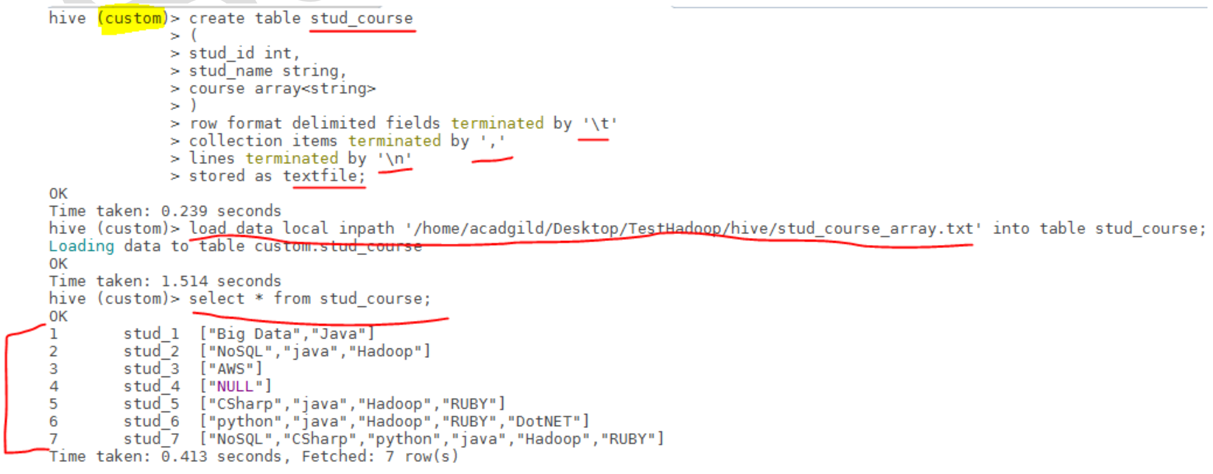
(Loading data from path /home/acadgild/Desktop/TestHadoop/hive/
stud_course_array.txt)

SELECT * FROM stud_course;

Explanation:

(Displaying the table content)

ScreenShot:



```
hive (custom)> create table stud_course
             > (
             > stud_id int,
             > stud_name string,
             > course array<string>
             > )
             > row format delimited fields terminated by '\t'
             > collection items terminated by ','
             > lines terminated by '\n'
             > stored as textfile;
OK
Time taken: 0.239 seconds
hive (custom)> load data local inpath '/home/acadgild/Desktop/TestHadoop/hive/stud_course_array.txt' into table stud_course;
Loading data to table custom.stud_course
OK
Time taken: 1.514 seconds
hive (custom)> select * from stud_course;
OK
1       stud_1  ["Big Data","Java"]
2       stud_2  ["NoSQL","java","Hadoop"]
3       stud_3  ["AWS"]
4       stud_4  ["NULL"]
5       stud_5  ["CSharp","java","Hadoop","RUBY"]
6       stud_6  ["python","java","Hadoop","RUBY","DotNET"]
7       stud_7  ["NoSQL","CSharp","python","java","Hadoop","RUBY"]
Time taken: 0.413 seconds, Fetched: 7 row(s)
```

Ans:

Note: Program files are properly documented for a detailed description of each instruction used within the program.

UDF Program file (***concat_udf.java***) is attached as a separate file.

hive (custom)> ADD JAR /home/acadgild/Desktop/TestHadoop/hive/hive-udf.jar;

hive (custom)> CREATE TEMPORARY FUNCTION concat_ws **AS** 'concat_udf';

Explanation:

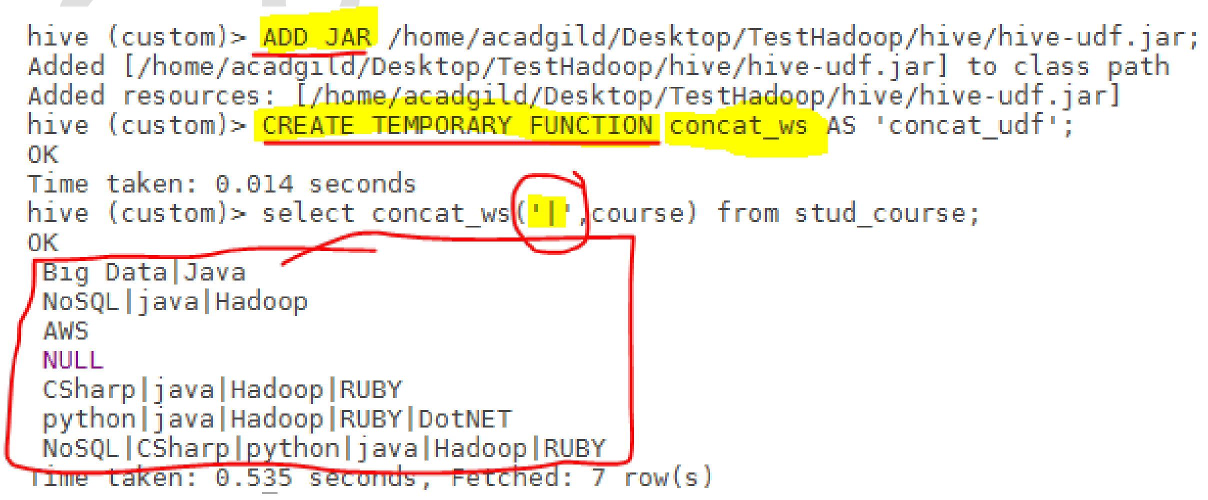
Adding jar containing UDF within hive shell & creating a temporary function concat_ws which would be used over the columns in the table.

SELECT CONCAT_WS('|',course**) FROM** stud_course**;**

Explanation:

(Displaying course using HIVE UDF '**CONCAT_WS**' using '|' separator)

ScreenShot:



```
hive (custom)> ADD JAR /home/acadgild/Desktop/TestHadoop/hive/hive-udf.jar;
Added [/home/acadgild/Desktop/TestHadoop/hive/hive-udf.jar] to class path
Added resources: [/home/acadgild/Desktop/TestHadoop/hive/hive-udf.jar]
hive (custom)> CREATE TEMPORARY FUNCTION concat_ws AS 'concat_udf';
OK
Time taken: 0.014 seconds
hive (custom)> select concat_ws('|',course) from stud_course;
OK
 Big Data|Java
 NoSQL|java|Hadoop
 AWS
 NULL
 CSharp|java|Hadoop|RUBY
 python|java|Hadoop|RUBY|DotNET
 NoSQL|CSharp|python|java|Hadoop|RUBY
Time taken: 0.535 seconds, Fetched: 7 row(s)
```

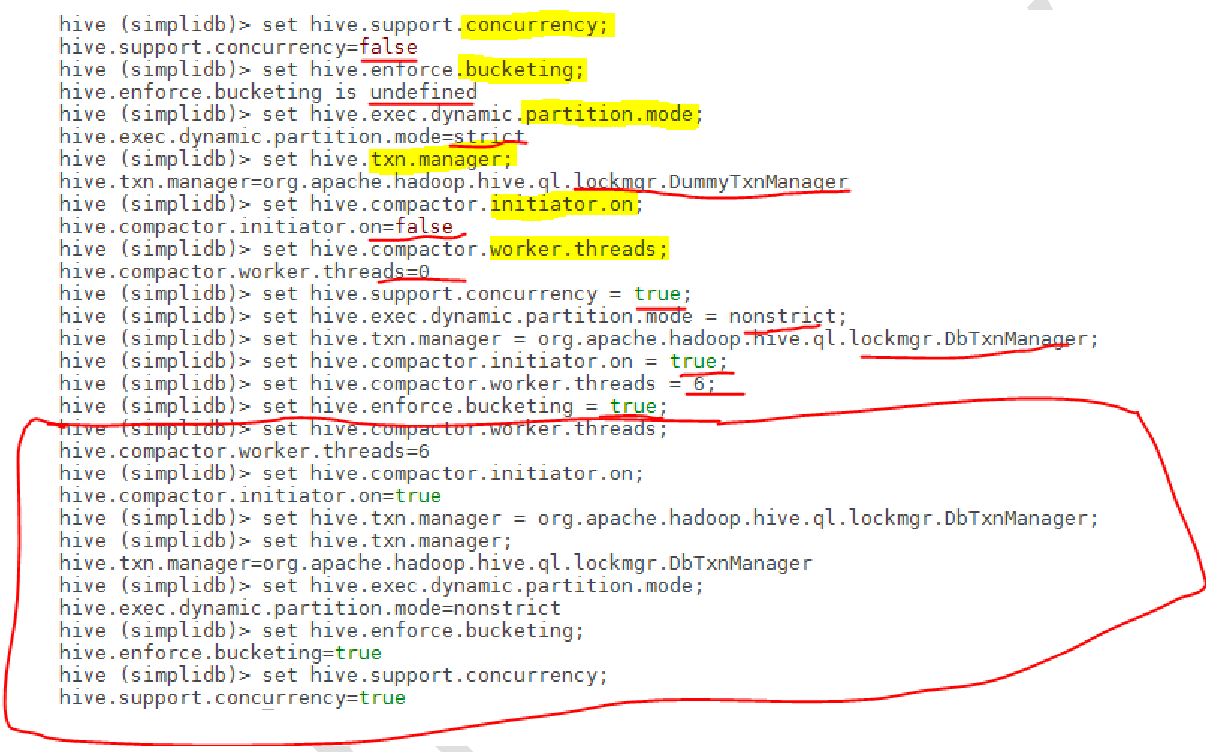
Task 3:

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.

Configuration Steps

Stop-all.sh



```
hive (simplidb)> set hive.support.concurrency;
hive.support.concurrency=false
hive (simplidb)> set hive.enforce.bucketing;
hive.enforce.bucketing is undefined
hive (simplidb)> set hive.exec.dynamic.partition.mode;
hive.exec.dynamic.partition.mode=strict
hive (simplidb)> set hive.txn.manager;
hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DummyTxnManager
hive (simplidb)> set hive.compactor.initiator.on;
hive.compactor.initiator.on=false
hive (simplidb)> set hive.compactor.worker.threads;
hive.compactor.worker.threads=0
hive (simplidb)> set hive.support.concurrency = true;
hive (simplidb)> set hive.exec.dynamic.partition.mode = nonstrict;
hive (simplidb)> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive (simplidb)> set hive.compactor.initiator.on = true;
hive (simplidb)> set hive.compactor.worker.threads = 6;
hive (simplidb)> set hive.enforce.bucketing = true;
hive (simplidb)> set hive.compactor.worker.threads;
hive.compactor.worker.threads=6
hive (simplidb)> set hive.compactor.initiator.on;
hive.compactor.initiator.on=true
hive (simplidb)> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive (simplidb)> set hive.txn.manager;
hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive (simplidb)> set hive.exec.dynamic.partition.mode;
hive.exec.dynamic.partition.mode=nonstrict
hive (simplidb)> set hive.enforce.bucketing;
hive.enforce.bucketing=true
hive (simplidb)> set hive.support.concurrency;
hive.support.concurrency=true
```

Start-all.sh

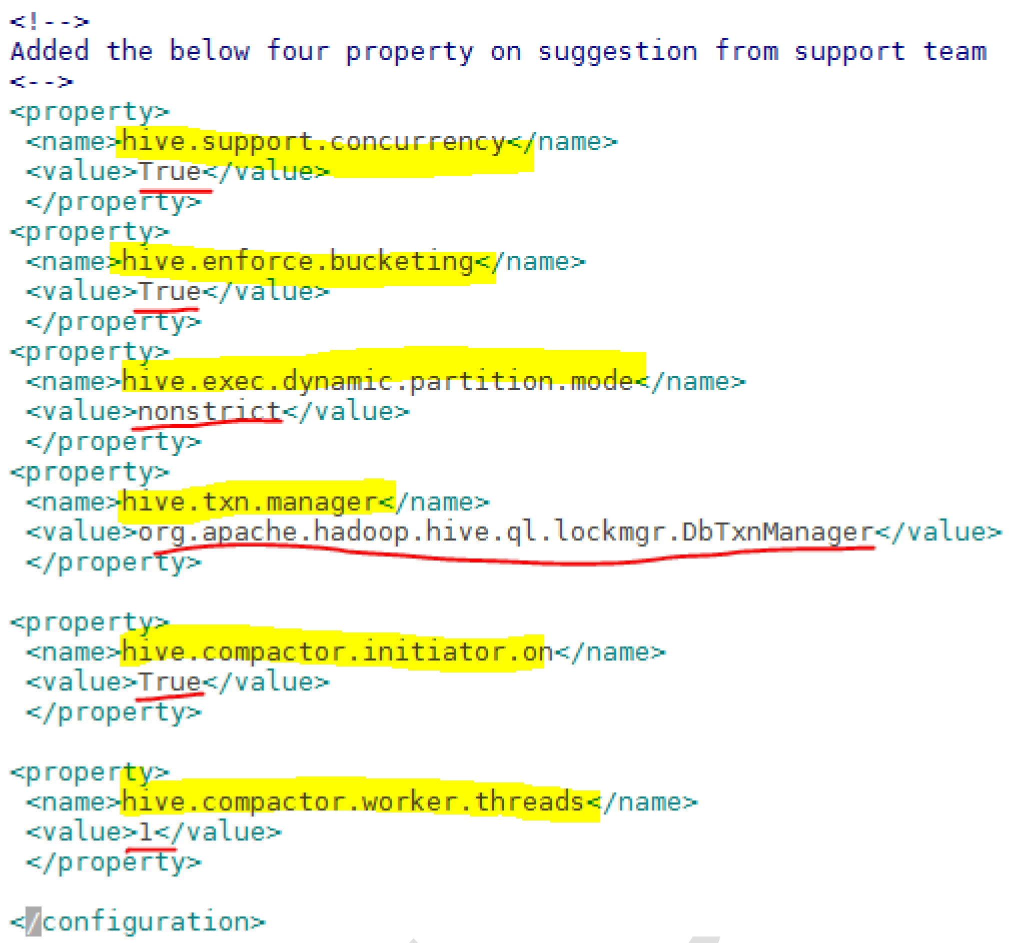
OR

If above does not work:

Stop-all.sh

Try adding following properties into:

/home/acadgild/install/hive/apache-hive-2.3.2-bin/conf/hive-site.xml

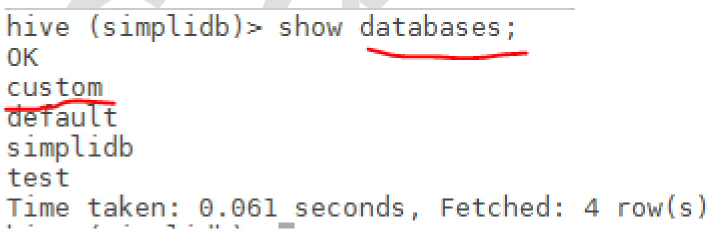


```
<!-->
Added the below four property on suggestion from support team
<-->
<property>
  <name>hive.support.concurrency</name>
  <value>True</value>
  </property>
<property>
  <name>hive.enforce.bucketing</name>
  <value>True</value>
  </property>
<property>
  <name>hive.exec.dynamic.partition.mode</name>
  <value>nonstrict</value>
  </property>
<property>
  <name>hive.txn.manager</name>
  <value>org.apache.hadoop.hive.ql.lockmgr.DbTxnManager</value>
  </property>

<property>
  <name>hive.compactor.initiator.on</name>
  <value>True</value>
  </property>

<property>
  <name>hive.compactor.worker.threads</name>
  <value>1</value>
  </property>

</configuration>
```

Start-all.sh

Ans:

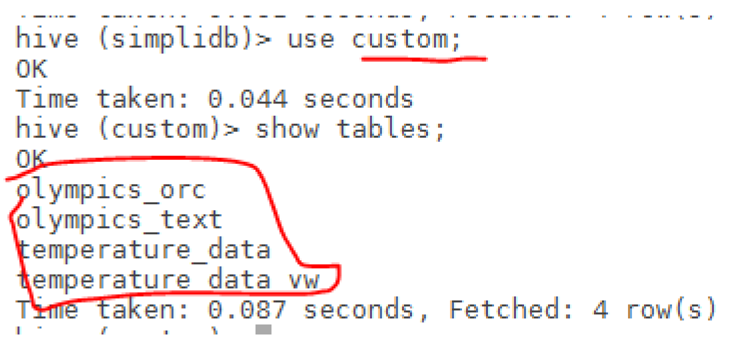
show databases;



```
hive (simplidb)> show databases;
OK
custom
default
simplidb
test
Time taken: 0.061 seconds, Fetched: 4 row(s)
```

use custom;

(Selecting the custom database to create the new table names employee)



```
hive (simplidb)> use custom;
OK
Time taken: 0.044 seconds
hive (custom)> show tables;
OK
olympics_orc
olympics_text
temperature_data
temperature_data_vw
Time taken: 0.087 seconds, Fetched: 4 row(s)
```

CREATE TABLE employee

(

id int,

name string,

salary int,

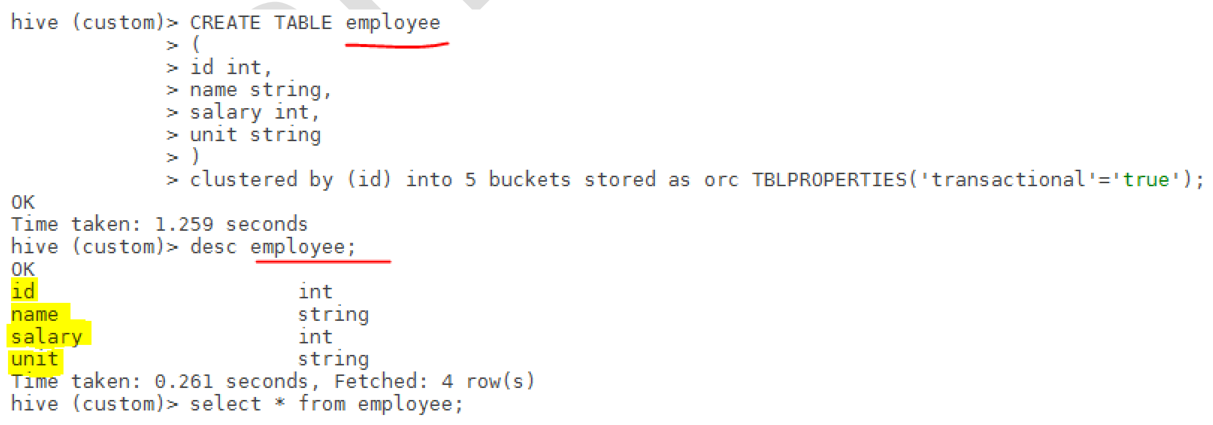
unit string

)

clustered by (id) into 5 buckets stored as orc
TBLPROPERTIES('transactional'='true');

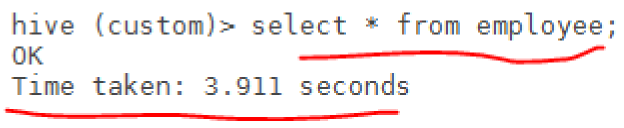
(Creating a table employee bucketing by id & enabling the transactions in the table by specifying it inside the TBLPROPERTIES as ‘transactional’=’true’)

DESC employee;



```
hive (custom)> CREATE TABLE employee
             > (
             > id int,
             > name string,
             > salary int,
             > unit string
             > )
             > clustered by (id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');
OK
Time taken: 1.259 seconds
hive (custom)> desc employee;
OK
id                      int
name                    string
salary                  int
unit                    string
Time taken: 0.261 seconds, Fetched: 4 row(s)
hive (custom)> select * from employee;
```

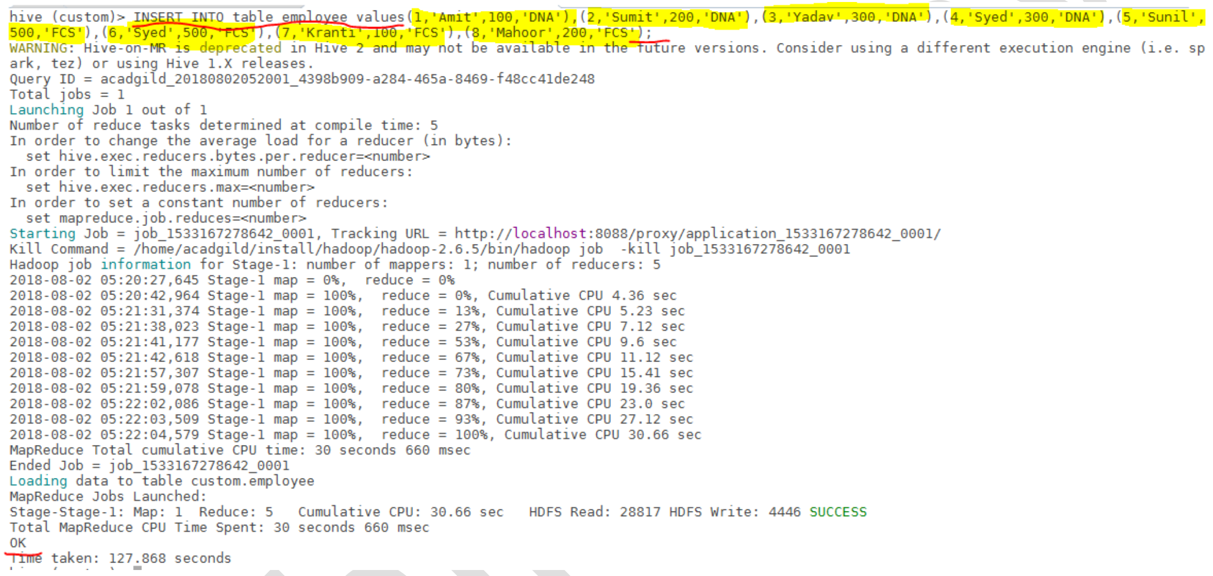
SELECT * from employee;



```
hive (custom)> select * from employee;
OK
Time taken: 3.911 seconds
```

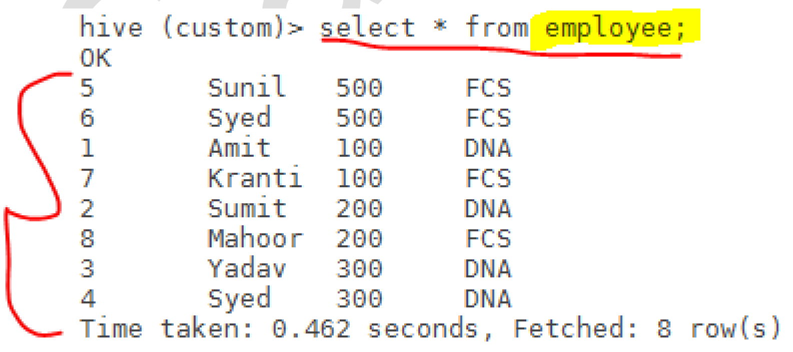
Inserting Data into a Hive Table

INSERT INTO table employee **values**(1,'Amit',100,'DNA'),(2,'Sumit',200,'DNA'),(3,'Yadav',300,'DNA'),(4,'Syed',300,'DNA'),(5,'Sunil',500,'FCS'),(6,'Syed',500,'FCS'),(7,'Kranti',100,'FCS'),(8,'Mahoor',200,'FCS');



```
hive (custom)> INSERT INTO table employee values(1,'Amit',100,'DNA'),(2,'Sumit',200,'DNA'),(3,'Yadav',300,'DNA'),(4,'Syed',300,'DNA'),(5,'Sunil',
500,'FCS'),(6,'Syed',500,'FCS'),(7,'Kranti',100,'FCS'),(8,'Mahoor',200,'FCS');
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. sp
ark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180802052001_4398b909-a284-465a-8469-f48cc41de248
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_1533167278642_0001, Tracking URL = http://localhost:8088/proxy/application_1533167278642_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533167278642_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2018-08-02 05:20:27,645 Stage-1 map = 0%,  reduce = 0%
2018-08-02 05:20:42,964 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.36 sec
2018-08-02 05:21:31,374 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 5.23 sec
2018-08-02 05:21:38,023 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 7.12 sec
2018-08-02 05:21:41,177 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 9.6 sec
2018-08-02 05:21:42,618 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 11.12 sec
2018-08-02 05:21:57,307 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 15.41 sec
2018-08-02 05:21:59,078 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 19.36 sec
2018-08-02 05:22:02,086 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 23.0 sec
2018-08-02 05:22:03,509 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 27.12 sec
2018-08-02 05:22:04,579 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 30.66 sec
MapReduce Total cumulative CPU time: 30 seconds 660 msec
Ended Job = job_1533167278642_0001
Loading data to table custom.employee
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 30.66 sec   HDFS Read: 28817 HDFS Write: 4446 SUCCESS
Total MapReduce CPU Time Spent: 30 seconds 660 msec
OK
Time taken: 127.868 seconds
```

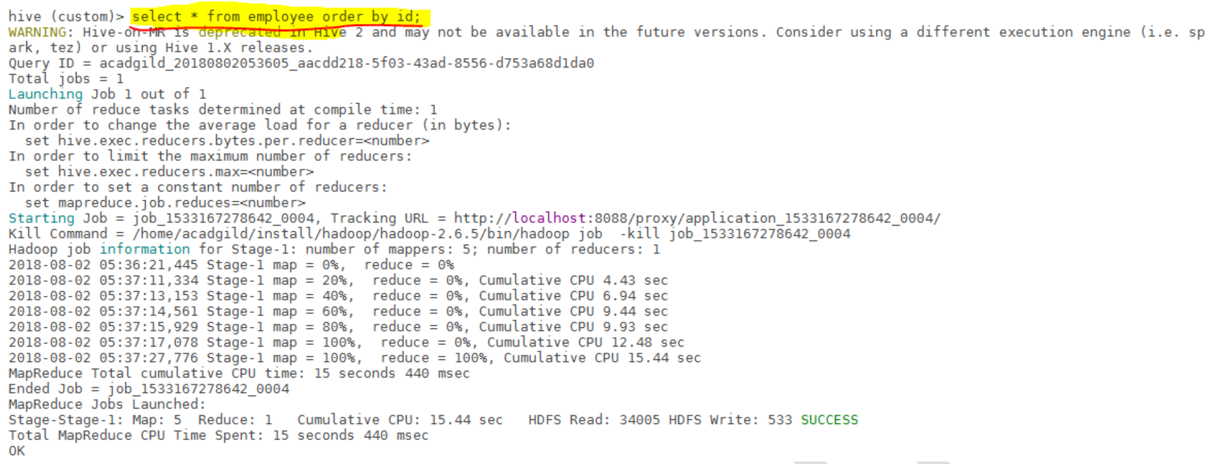
SELECT * from employee;



```
hive (custom)> select * from employee;
OK
5       Sunil   500     FCS
6       Syed    500     FCS
1       Amit    100     DNA
7       Kranti  100     FCS
2       Sumit   200     DNA
8       Mahoor  200     FCS
3       Yadav   300     DNA
4       Syed    300     DNA
Time taken: 0.462 seconds, Fetched: 8 row(s)
```

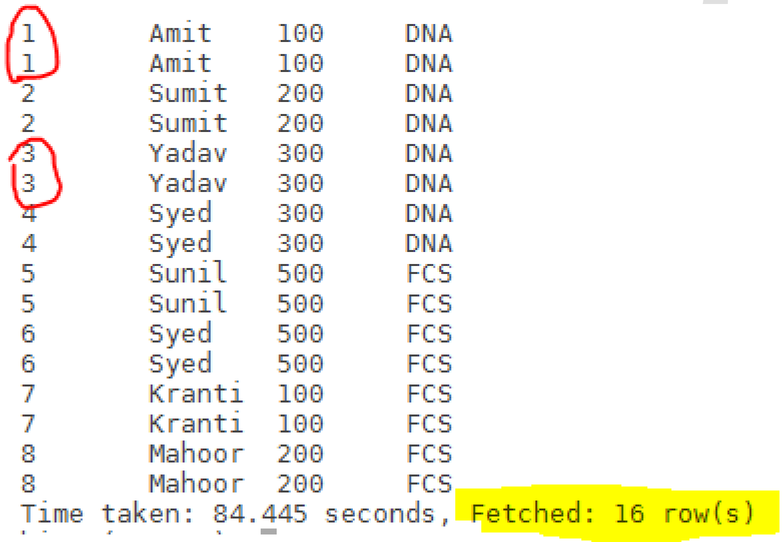
AGAIN INSERTING THE SAME DATA

INSERT INTO table employee **values**(1,'Amit',100,'DNA'),(2,'Sumit',200,'DNA'),(3,'Yadav',300,'DNA'),(4,'Syed',300,'DNA'),(5,'Sunil',500,'FCS'),(6,'Syed',500,'FCS'),(7,'Kranti',100,'FCS'),(8,'Mahoor',200,'FCS');



```
hive (custom)> select * from employee order by id;
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. sp
ark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180802053605_aacdd218-5f03-43ad-8556-d753a68d1da0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
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_1533167278642_0004, Tracking URL = http://localhost:8088/proxy/application_1533167278642_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1533167278642_0004
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 1
2018-08-02 05:36:21,445 Stage-1 map = 0%,  reduce = 0%
2018-08-02 05:37:11,334 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 4.43 sec
2018-08-02 05:37:13,153 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 6.94 sec
2018-08-02 05:37:14,561 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 9.44 sec
2018-08-02 05:37:15,929 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 9.93 sec
2018-08-02 05:37:17,078 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 12.48 sec
2018-08-02 05:37:27,776 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 15.44 sec
MapReduce Total cumulative CPU time: 15 seconds 440 msec
Ended Job = job_1533167278642_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 1   Cumulative CPU: 15.44 sec   HDFS Read: 34005 HDFS Write: 533 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 440 msec
OK
```

Note: Duplicate data Inserted.



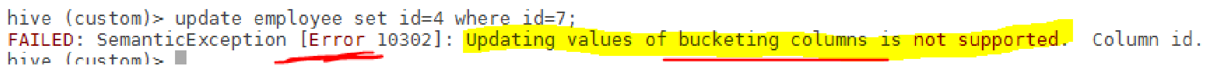
```
1       Amit    100     DNA
1       Amit    100     DNA
2       Sumit   200     DNA
2       Sumit   200     DNA
3       Yadav   300     DNA
3       Yadav   300     DNA
4       Syed    300     DNA
4       Syed    300     DNA
5       Sunil   500     FCS
5       Sunil   500     FCS
6       Syed    500     FCS
6       Syed    500     FCS
7       Kranti  100     FCS
7       Kranti  100     FCS
8       Mahoor  200     FCS
8       Mahoor  200     FCS
Time taken: 84.445 seconds, Fetched: 16 row(s)
```

Updating the Data in Hive Table

UPDATE employee **SET** id=4 **WHERE** id=7;

(The above command is used to update a row in Hive table.)

Update on bucketing columns throws Error.



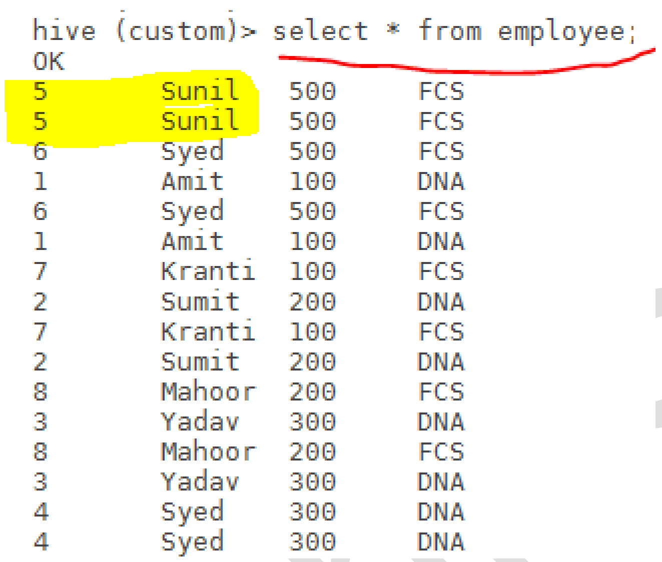
```
hive (custom)> update employee set id=4 where id=7;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported.  Column id.
hive (custom)>
```

Update on Non-Bucketing Columns.

UPDATE employee **SET** name='Sahil Sahay' **WHERE** id=5;

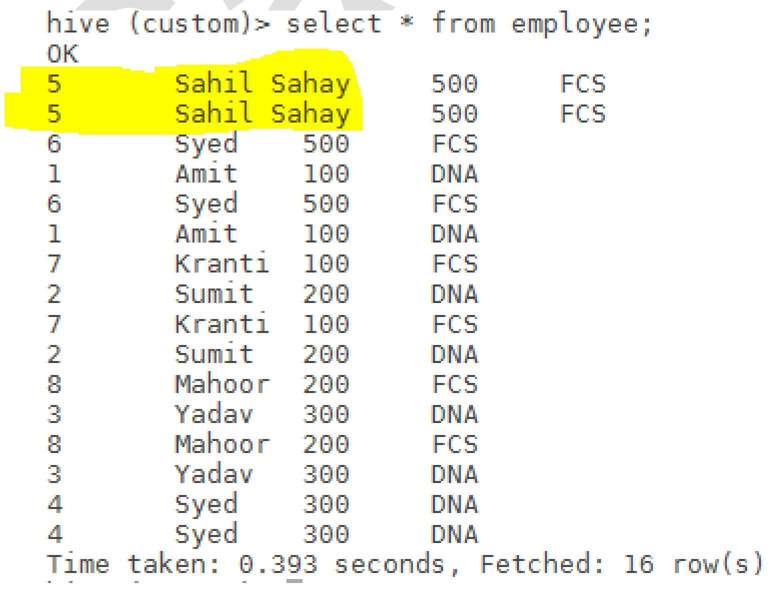
(Updating name for id 5)

Before Update:



```
hive (custom)> select * from employee;
OK
5         Sunil    500      FCS
5         Sunil    500      FCS
6         Syed     500      FCS
1         Amit     100      DNA
6         Syed     500      FCS
1         Amit     100      DNA
7         Kranti   100      FCS
2         Sumit    200      DNA
7         Kranti   100      FCS
2         Sumit    200      DNA
8         Mahoor   200      FCS
3         Yadav    300      DNA
8         Mahoor   200      FCS
3         Yadav    300      DNA
4         Syed     300      DNA
4         Syed     300      DNA
```

After Update:



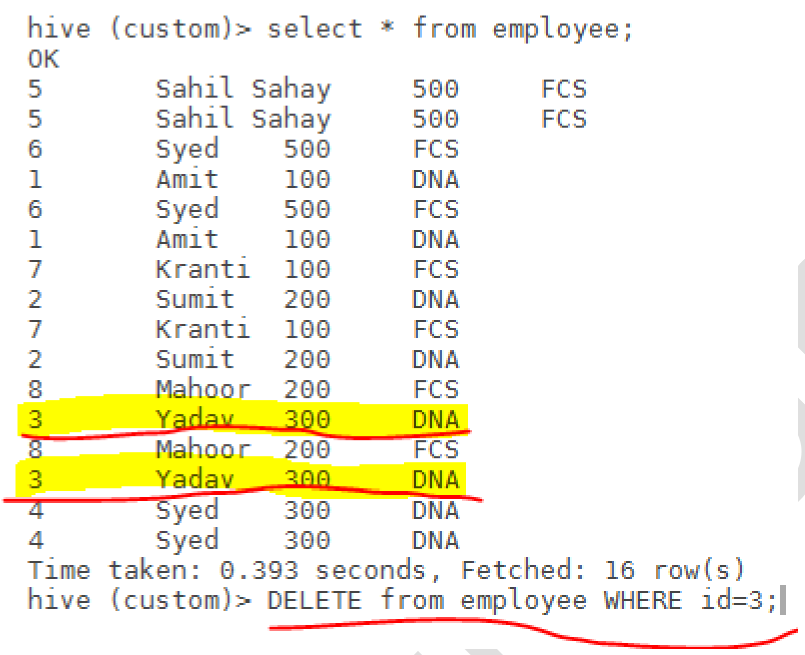
```
hive (custom)> select * from employee;
OK
5         Sahil Sahay      500      FCS
5         Sahil Sahay      500      FCS
6         Syed     500      FCS
1         Amit     100      DNA
6         Syed     500      FCS
1         Amit     100      DNA
7         Kranti   100      FCS
2         Sumit    200      DNA
7         Kranti   100      FCS
2         Sumit    200      DNA
8         Mahoor   200      FCS
3         Yadav    300      DNA
8         Mahoor   200      FCS
3         Yadav    300      DNA
4         Syed     300      DNA
4         Syed     300      DNA
Time taken: 0.393 seconds, Fetched: 16 row(s)
```

Deleting a Row from Hive Table

DELETE from employee **WHERE** id=3;

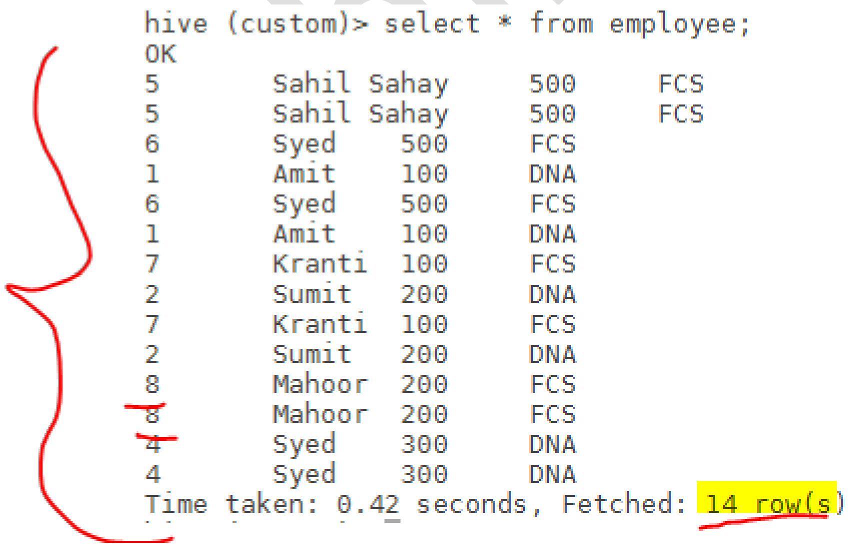
(Deleting entire row for id 5)

Before Delete:



```
hive (custom)> select * from employee;
OK
5       Sahil Sahay     500     FCS
5       Sahil Sahay     500     FCS
6       Syed    500     FCS
1       Amit    100     DNA
6       Syed    500     FCS
1       Amit    100     DNA
7       Kranti  100     FCS
2       Sumit   200     DNA
7       Kranti  100     FCS
2       Sumit   200     DNA
8       Mahoor  200     FCS
3       Yadav   300     DNA
8       Mahoor  200     FCS
3       Yadav   300     DNA
4       Syed    300     DNA
4       Syed    300     DNA
Time taken: 0.393 seconds, Fetched: 16 row(s)
hive (custom)> DELETE from employee WHERE id=3;
```

After Delete:



```
hive (custom)> select * from employee;
OK
5       Sahil Sahay     500     FCS
5       Sahil Sahay     500     FCS
6       Syed    500     FCS
1       Amit    100     DNA
6       Syed    500     FCS
1       Amit    100     DNA
7       Kranti  100     FCS
2       Sumit   200     DNA
7       Kranti  100     FCS
2       Sumit   200     DNA
8       Mahoor  200     FCS
8       Mahoor  200     FCS
4       Syed    300     DNA
4       Syed    300     DNA
Time taken: 0.42 seconds, Fetched: 14 row(s)
```

******************************** **End** ********************************