Session 9: Advance Hive Assignment 1

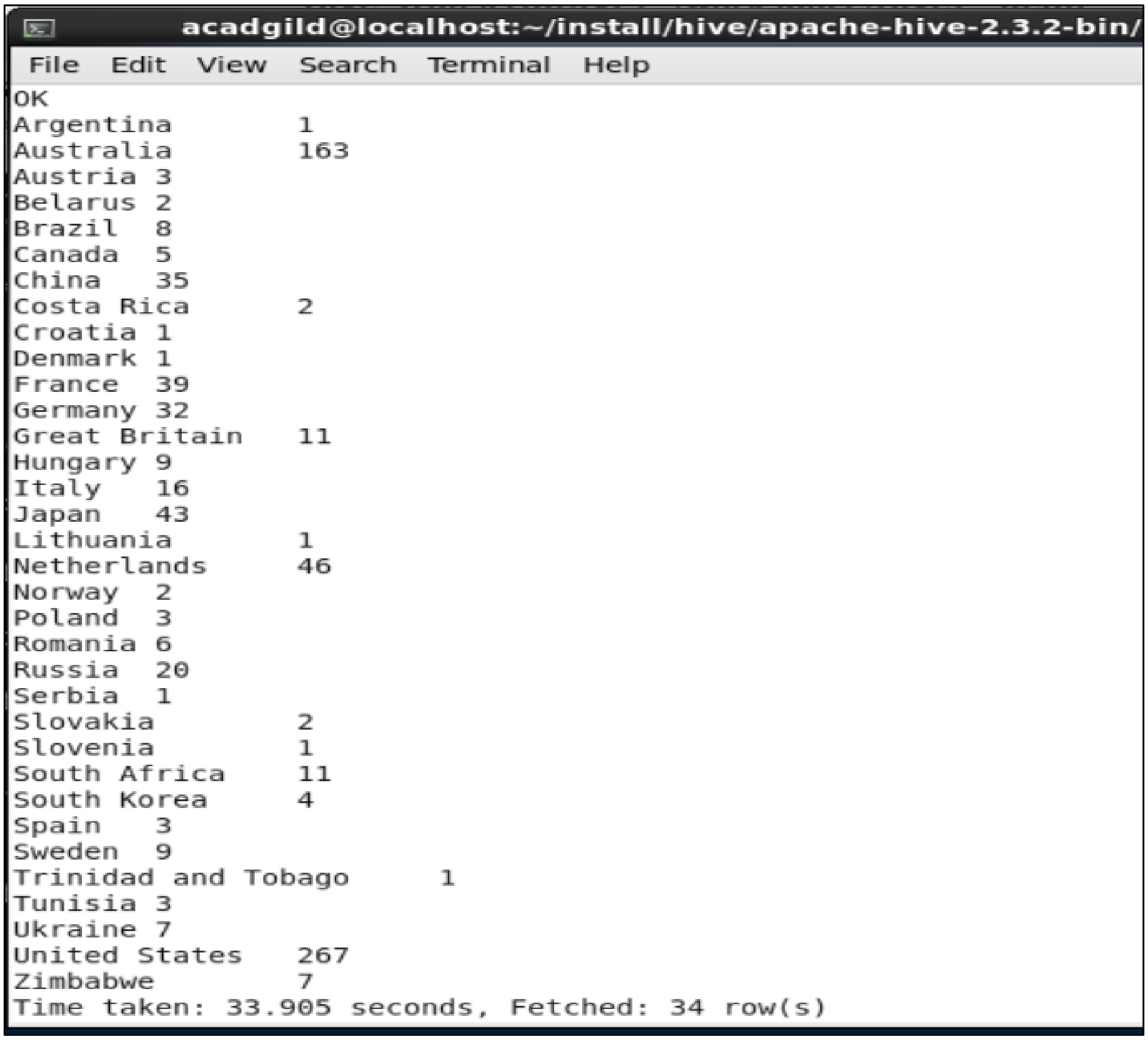
Task 1

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

QUERY:

```
SELECT country,SUM(totalmedal) FROM olympics_data WHERE
sport='Swimming' GROUP BY country;
```

OUTPUT:


acadgild@localhost:~/install/hive/apache-hive-2.3.2-bin/
File Edit View Search Terminal Help

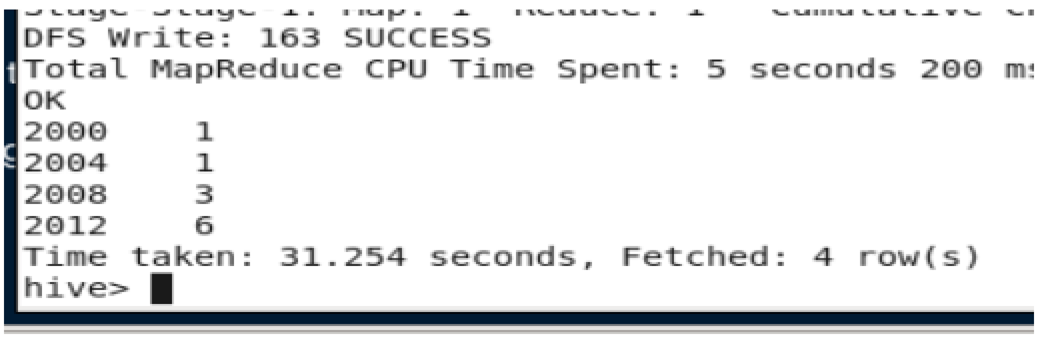
```
OK
Argentina       1
Australia       163
Austria 3
Belarus 2
Brazil  8
Canada  5
China   35
Costa Rica      2
Croatia 1
Denmark 1
France  39
Germany 32
Great Britain   11
Hungary 9
Italy   16
Japan   43
Lithuania       1
Netherlands     46
Norway  2
Poland  3
Romania 6
Russia  20
Serbia  1
Slovakia        2
Slovenia        1
South Africa    11
South Korea     4
Spain   3
Sweden  9
Trinidad and Tobago     1
Tunisia 3
Ukraine 7
United States   267
Zimbabwe        7
Time taken: 33.905 seconds, Fetched: 34 row(s)
```

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

QUERY:

```
SELECT olympicsyear, SUM(totalmedal) FROM olympics_data WHERE country =
'India' GROUP BY olympicsyear;
```

OUTPUT:



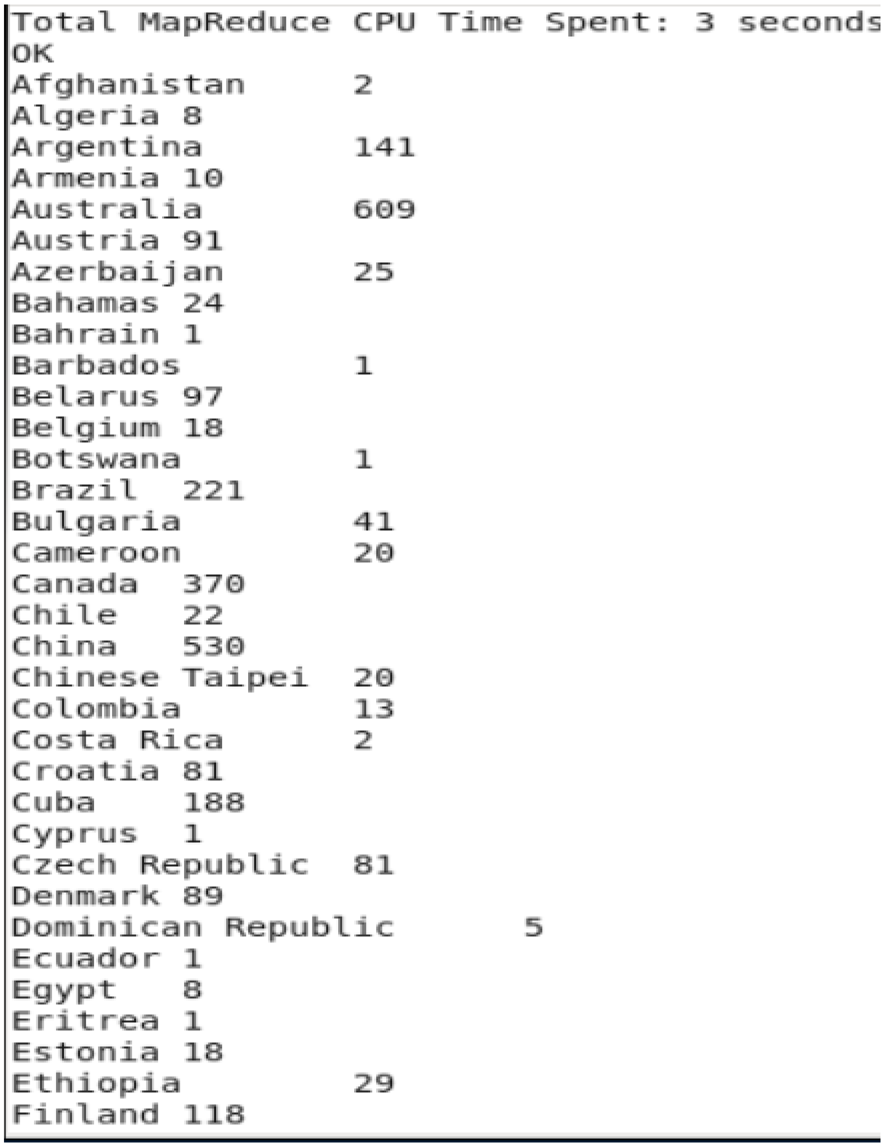
```
DFS Write: 163 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 200 m
OK
2000    1
2004    1
2008    3
2012    6
Time taken: 31.254 seconds, Fetched: 4 row(s)
hive>
```

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

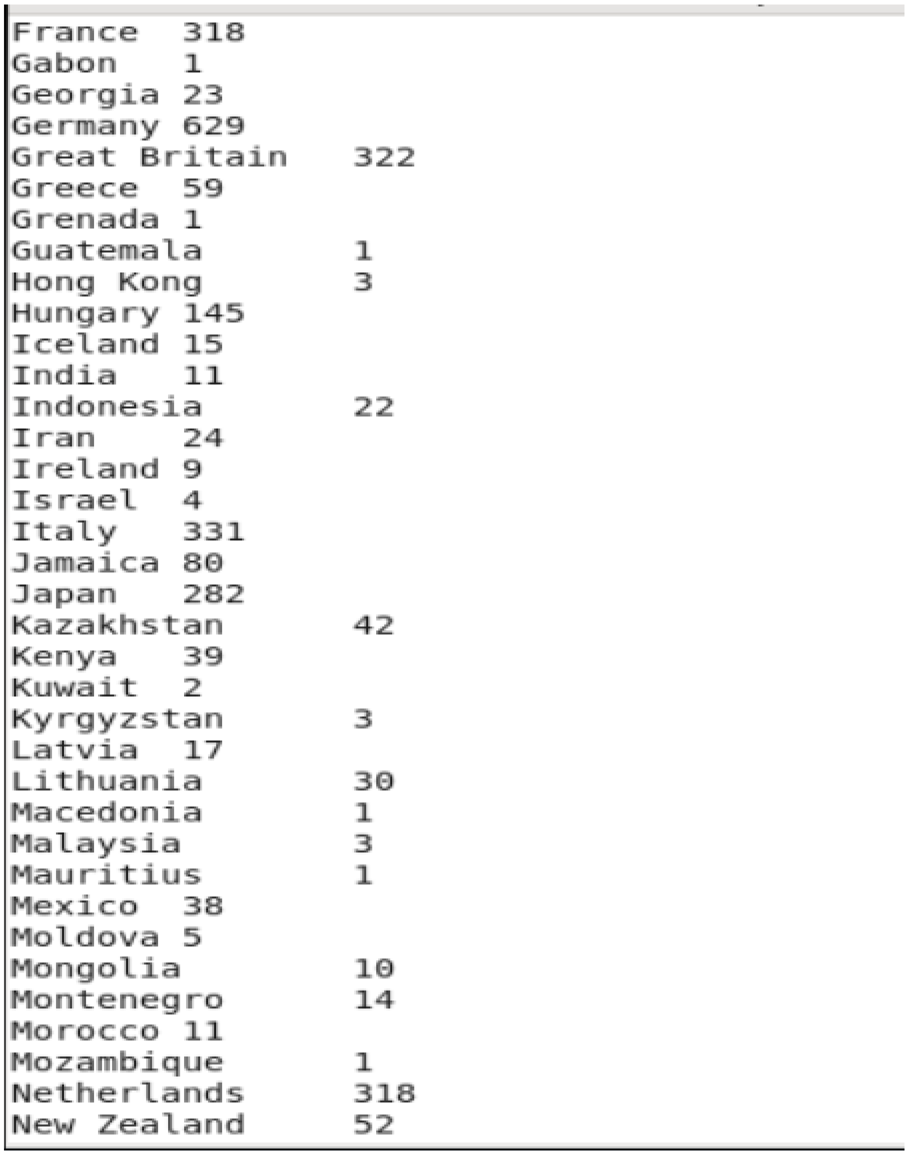
QUERY:

```
SELECT country, SUM(totalmedal) FROM olympics_data GROUP BY country;
```

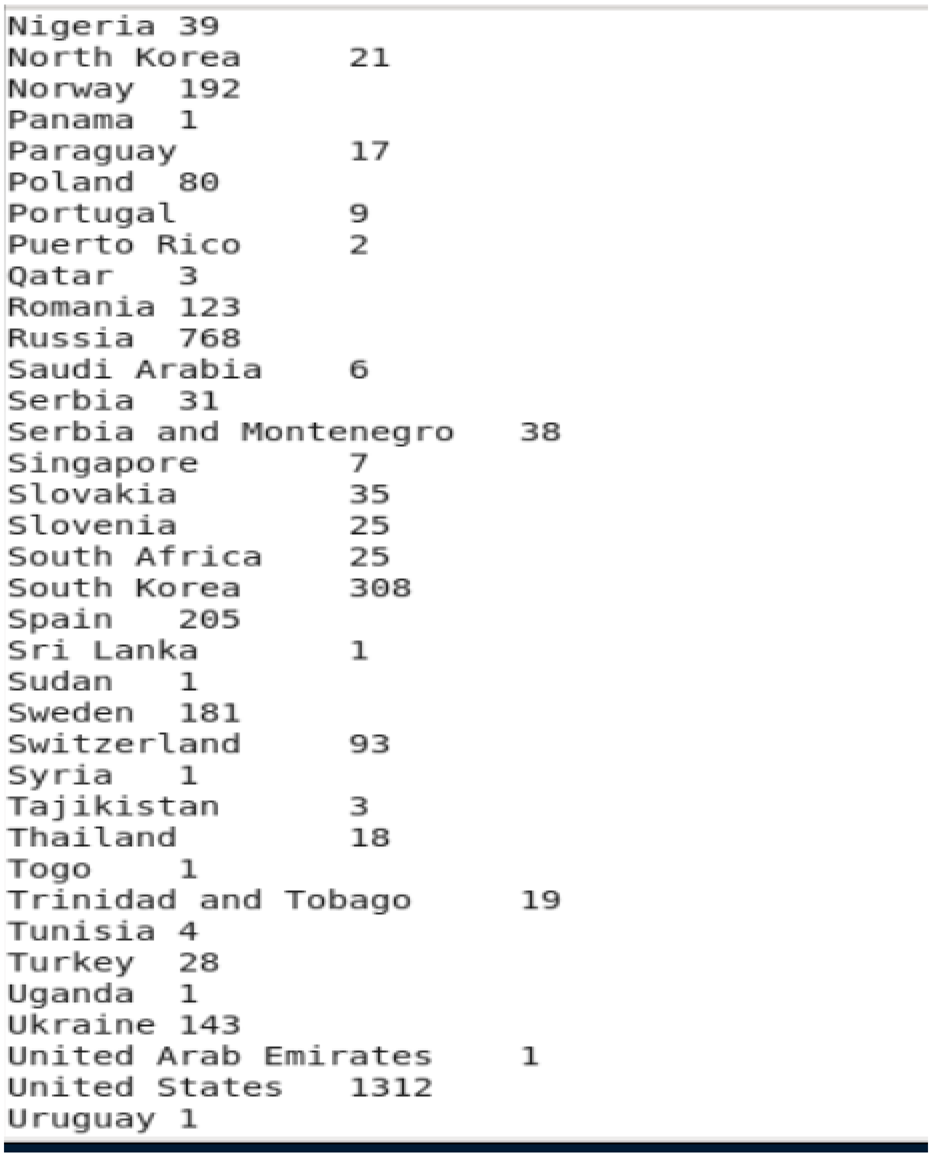
OUTPUT:



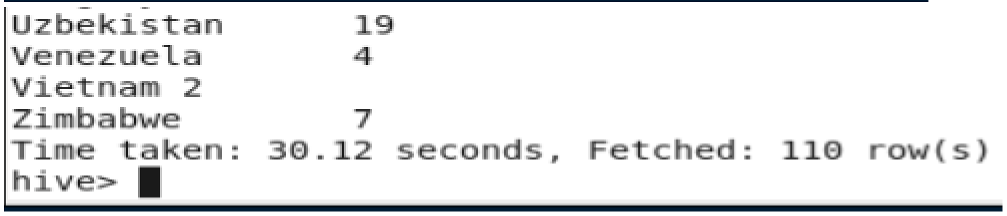
```
Total MapReduce CPU Time Spent: 3 seconds
OK
Afghanistan     2
Algeria 8
Argentina       141
Armenia 10
Australia       609
Austria 91
Azerbaijan      25
Bahamas 24
Bahrain 1
Barbados        1
Belarus 97
Belgium 18
Botswana        1
Brazil  221
Bulgaria        41
Cameroon        20
Canada  370
Chile   22
China   530
Chinese Taipei  20
Colombia        13
Costa Rica      2
Croatia 81
Cuba    188
Cyprus  1
Czech Republic  81
Denmark 89
Dominican Republic      5
Ecuador 1
Egypt   8
Eritrea 1
Estonia 18
Ethiopia        29
Finland 118
```



```
France  318
Gabon   1
Georgia 23
Germany 629
Great Britain   322
Greece  59
Grenada 1
Guatemala       1
Hong Kong       3
Hungary 145
Iceland 15
India   11
Indonesia       22
Iran    24
Ireland 9
Israel  4
Italy   331
Jamaica 80
Japan   282
Kazakhstan      42
Kenya   39
Kuwait  2
Kyrgyzstan      3
Latvia  17
Lithuania       30
Macedonia       1
Malaysia        3
Mauritius       1
Mexico  38
Moldova 5
Mongolia        10
Montenegro      14
Morocco 11
Mozambique      1
Netherlands     318
New Zealand     52
```



```
Nigeria 39
North Korea     21
Norway  192
Panama  1
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
```



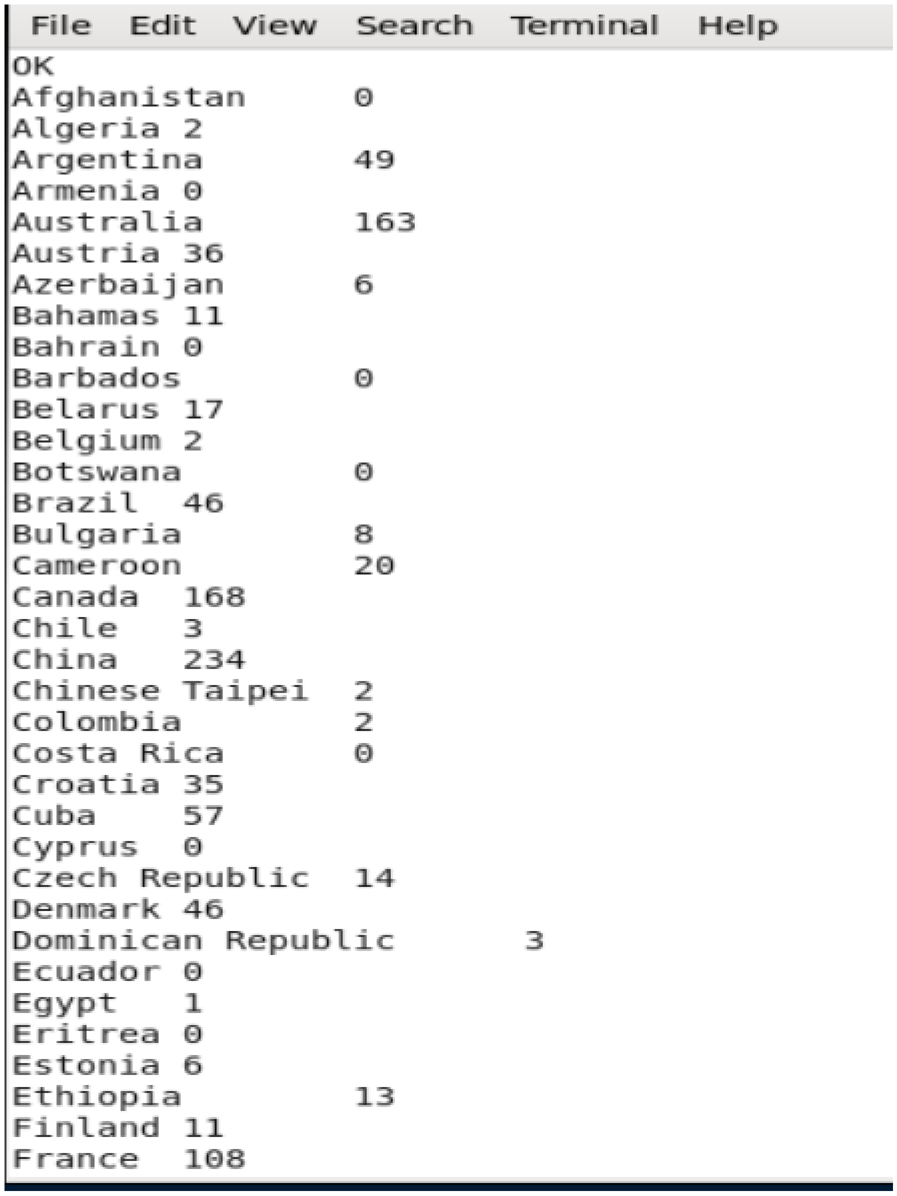
```
Uzbekistan      19
Venezuela       4
Vietnam 2
Zimbabwe        7
Time taken: 30.12 seconds, Fetched: 110 row(s)
hive>
```

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

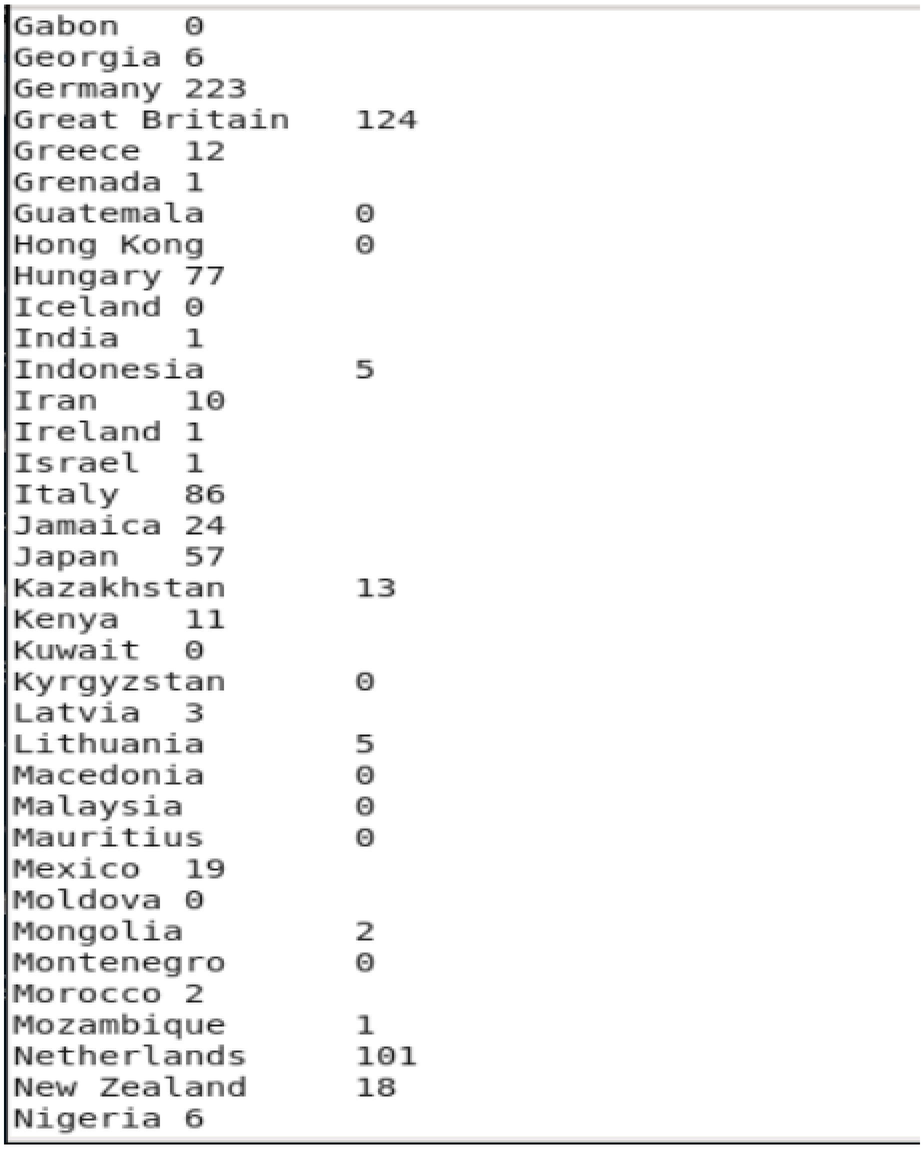
QUERY:

```
SELECT country, SUM(goldmedal) FROM olympics_data GROUP BY country;
```

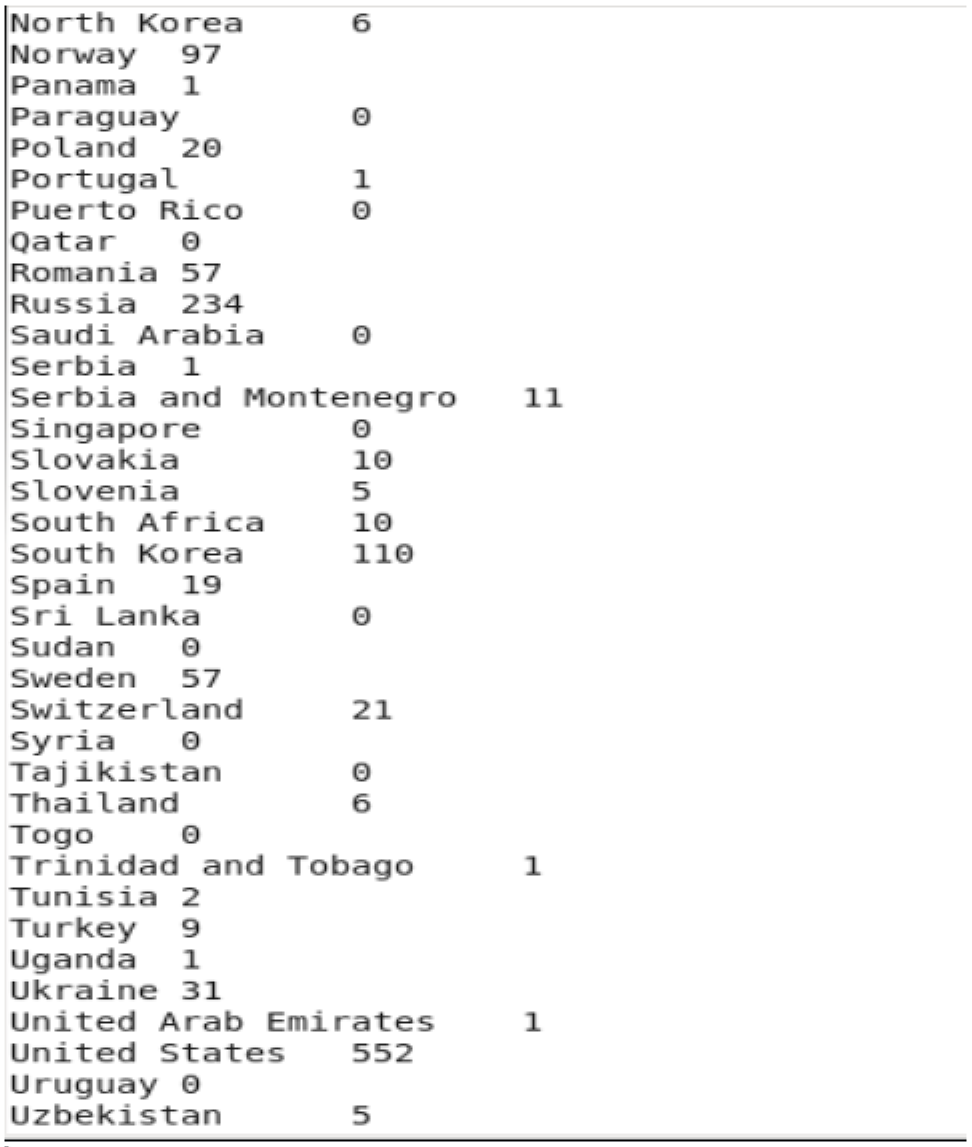
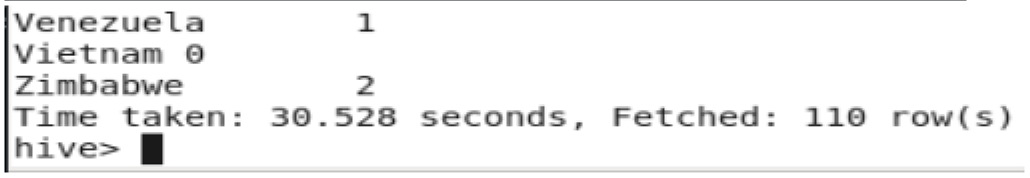
OUTPUT:



```
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
Botswana        0
Brazil  46
Bulgaria        8
Cameroon        20
Canada  168
Chile   3
China   234
Chinese Taipei  2
Colombia        2
Costa Rica      0
Croatia 35
Cuba    57
Cyprus  0
Czech Republic  14
Denmark 46
Dominican Republic      3
Ecuador 0
Egypt   1
Eritrea 0
Estonia 6
Ethiopia        13
Finland 11
France  108
```



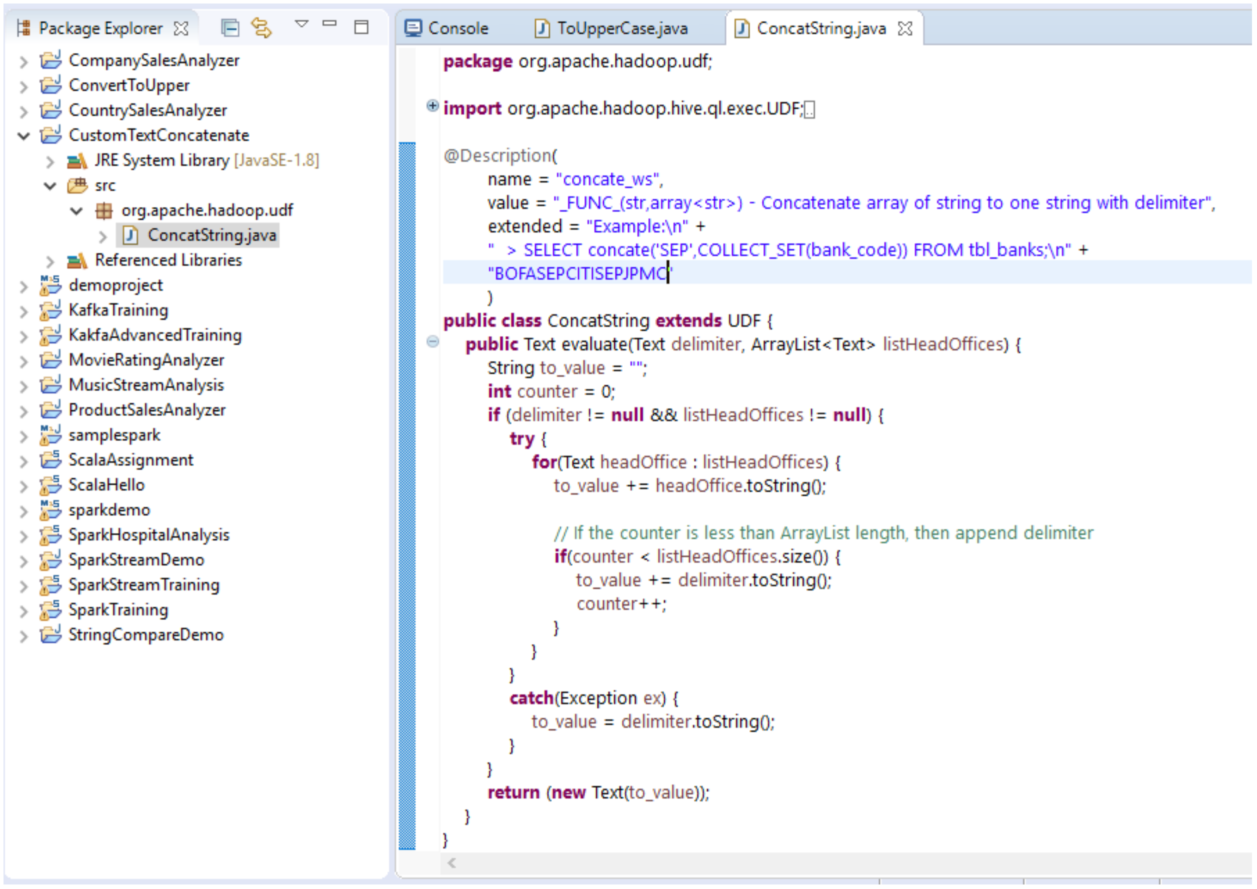
```
Gabon	0
Georgia	6
Germany	223
Great Britain	124
Greece	12
Grenada	1
Guatemala	0
Hong Kong	0
Hungary	77
Iceland	0
India	1
Indonesia	5
Iran	10
Ireland	1
Israel	1
Italy	86
Jamaica	24
Japan	57
Kazakhstan	13
Kenya	11
Kuwait	0
Kyrgyzstan	0
Latvia	3
Lithuania	5
Macedonia	0
Malaysia	0
Mauritius	0
Mexico	19
Moldova	0
Mongolia	2
Montenegro	0
Morocco	2
Mozambique	1
Netherlands	101
New Zealand	18
Nigeria	6
```


```
North Korea     6
Norway  97
Panama  1
Paraguay        0
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: 30.528 seconds, Fetched: 110 row(s)
hive> 
```

Task – 2:

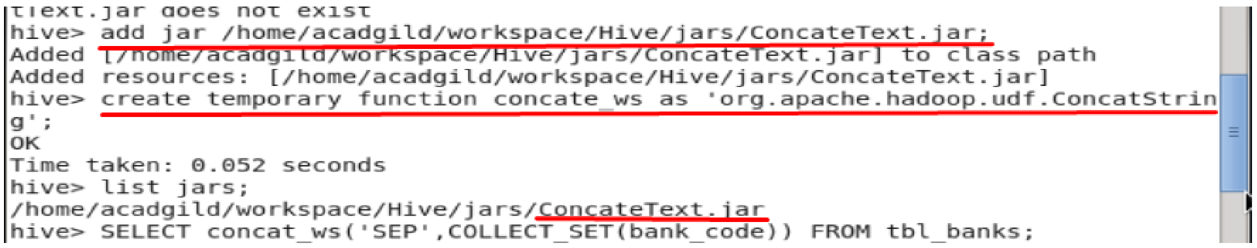
Write a hive UDF that implements functionality of string concat_ws(string SEP, array<string>).
This UDF will accept two arguments, one string and one array of string.
It will return a single string where all the elements of the array are separated by the SEP.

Program:



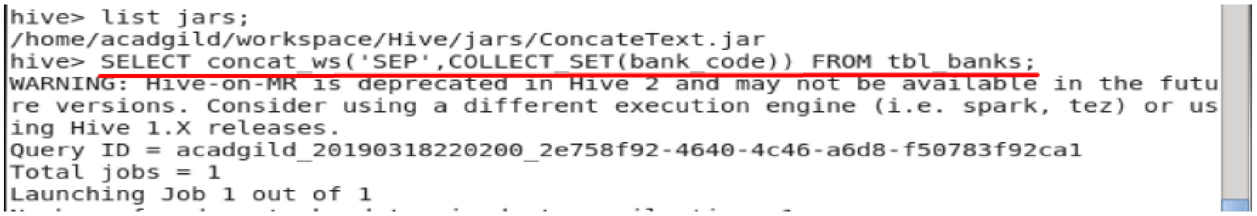
```java
package org.apache.hadoop.udf;

import org.apache.hadoop.hive.ql.exec.UDF;

@Description(
      name = "concate_ws",
      value = "_FUNC_(str,array<str>) - Concatenate array of string to one string with delimiter",
      extended = "Example:\n" +
      "  > SELECT concate('SEP',COLLECT_SET(bank_code)) FROM tbl_banks;\n" +
      "BOFASEPCITISEPJPMC"
      )
public class ConcatString extends UDF {
   public Text evaluate(Text delimiter, ArrayList<Text> listHeadOffices) {
      String to_value = "";
      int counter = 0;
      if (delimiter != null && listHeadOffices != null) {
         try {
            for(Text headOffice : listHeadOffices) {
               to_value += headOffice.toString();

               // If the counter is less than ArrayList length, then append delimiter
               if(counter < listHeadOffices.size()) {
                  to_value += delimiter.toString();
                  counter++;
               }
            }
         }
         catch(Exception ex) {
            to_value = delimiter.toString();
         }
      }
      return (new Text(to_value));
   }
}
```

Build and export as jar. Then register the jar in Hive as temporary function,

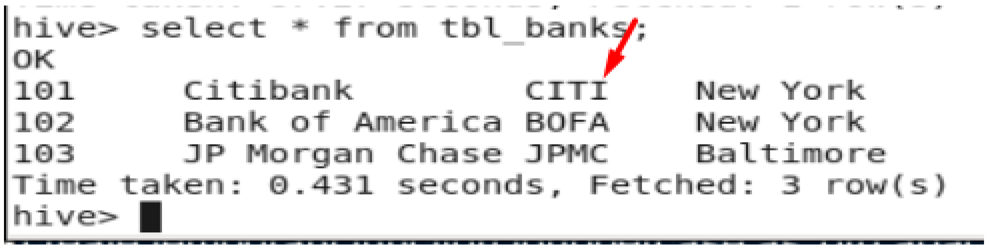


```
Text.jar does not exist
hive> add jar /home/acadgild/workspace/Hive/jars/ConcateText.jar;
Added [/home/acadgild/workspace/Hive/jars/ConcateText.jar] to class path
Added resources: [/home/acadgild/workspace/Hive/jars/ConcateText.jar]
hive> create temporary function concate_ws as 'org.apache.hadoop.udf.ConcatStrin
g';
OK
Time taken: 0.052 seconds
hive> list jars;
/home/acadgild/workspace/Hive/jars/ConcateText.jar
hive> SELECT concat_ws('SEP',COLLECT_SET(bank_code)) FROM tbl_banks;
```

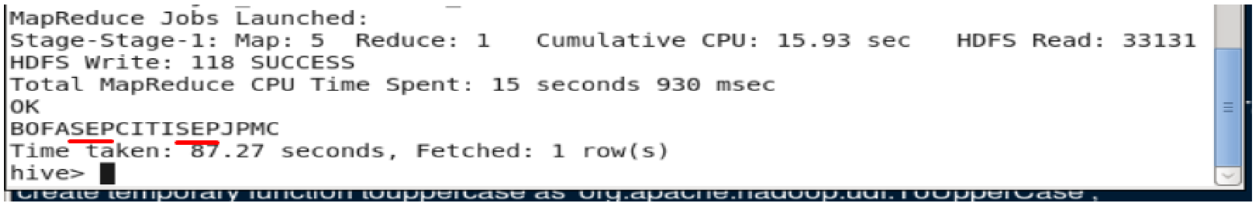
Execution:


```
hive> list jars;
/home/acadgild/workspace/Hive/jars/ConcateText.jar
hive> SELECT concat_ws('SEP',COLLECT_SET(bank_code)) FROM tbl_banks;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20190318220200_2e758f92-4640-4c46-a6d8-f50783f92ca1
Total jobs = 1
Launching Job 1 out of 1
```

The above command intends to concatenate bank code with delimiter as “SEP”


```
hive> select * from tbl_banks;
OK
101     Citibank        CITI    New York
102     Bank of America BOFA    New York
103     JP Morgan Chase JPMC    Baltimore
Time taken: 0.431 seconds, Fetched: 3 row(s)
hive>
```

Final result will be as follows,

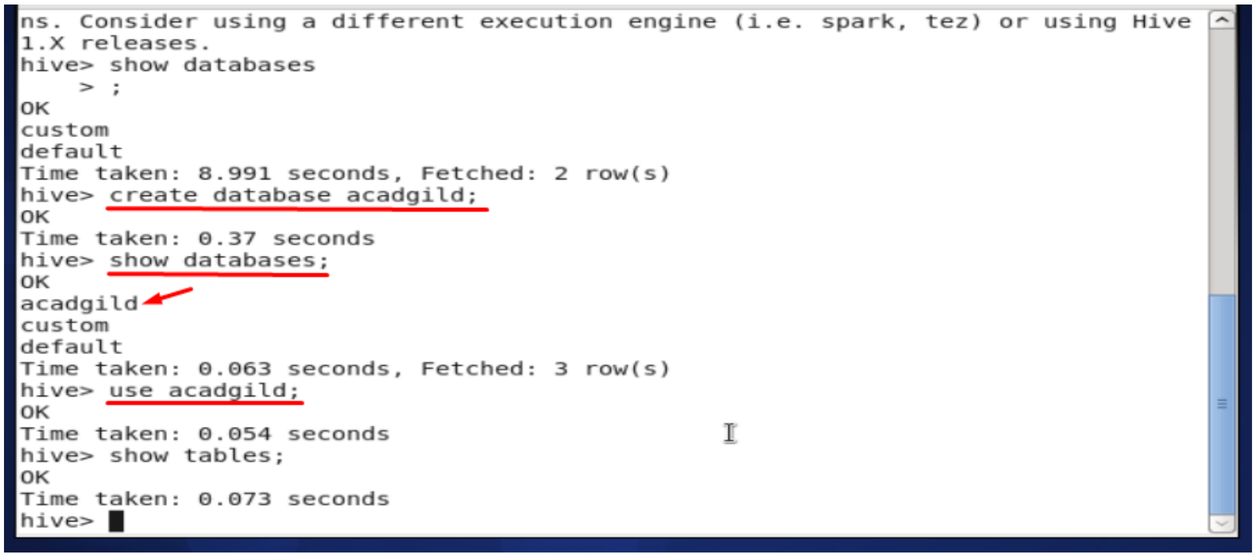

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 1   Cumulative CPU: 15.93 sec   HDFS Read: 33131
HDFS Write: 118 SUCCESS
Total MapReduce CPU Time Spent: 15 seconds 930 msec
OK
BOFASEPCITISEPJPMC
Time taken: 87.27 seconds, Fetched: 1 row(s)
hive>
```

Task – 3:

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

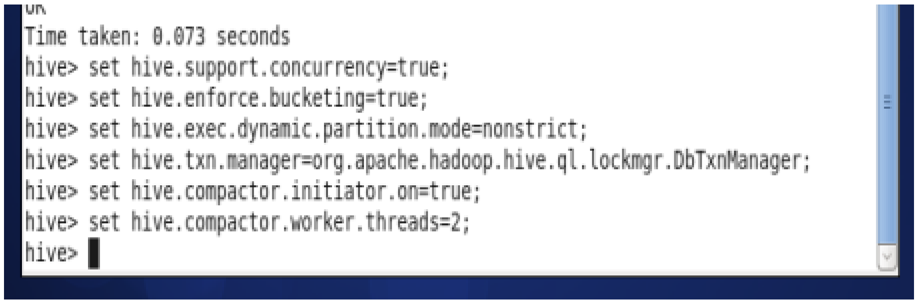
Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

Execution:



```
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive> show databases
    > ;
OK
custom
default
Time taken: 8.991 seconds, Fetched: 2 row(s)
hive> create database acadgild;
OK
Time taken: 0.37 seconds
hive> show databases;
OK
acadgild
custom
default
Time taken: 0.063 seconds, Fetched: 3 row(s)
hive> use acadgild;
OK
Time taken: 0.054 seconds
hive> show tables;
OK
Time taken: 0.073 seconds
hive>
```

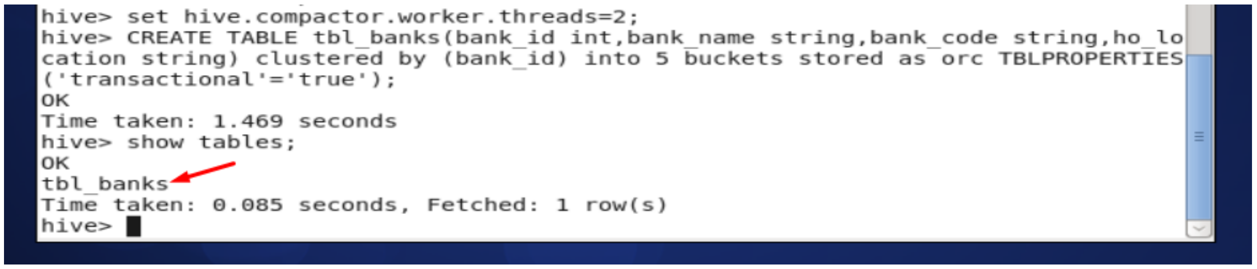
Properties to set appropriately in Hive shell to work with hive transactions in table,



```
OK
Time taken: 0.073 seconds
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=2;
hive>
```

CREATE NEW TABLE:

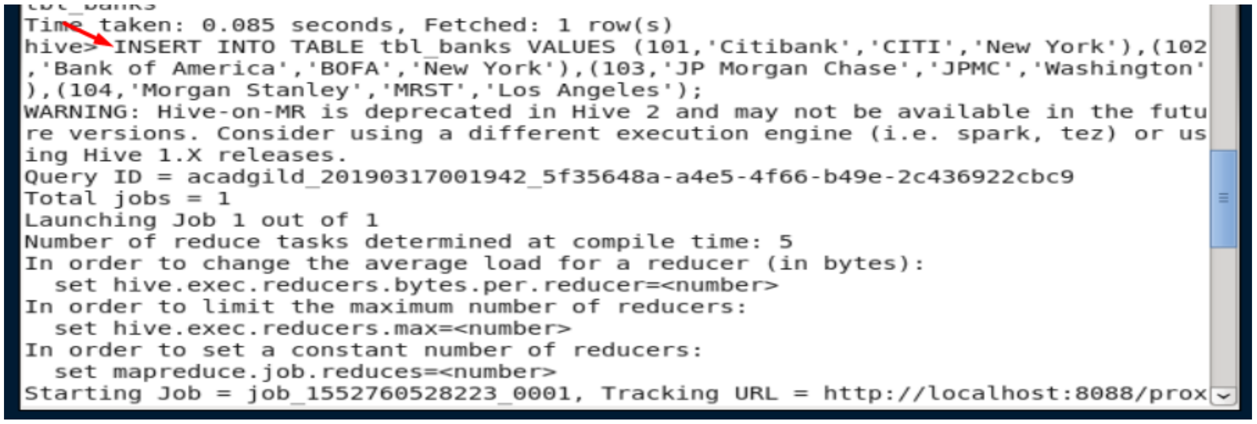
CREATE TABLE tbl_banks(bank_id int,bank_name string,bank_code string,ho_location string) clustered by (bank_id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');



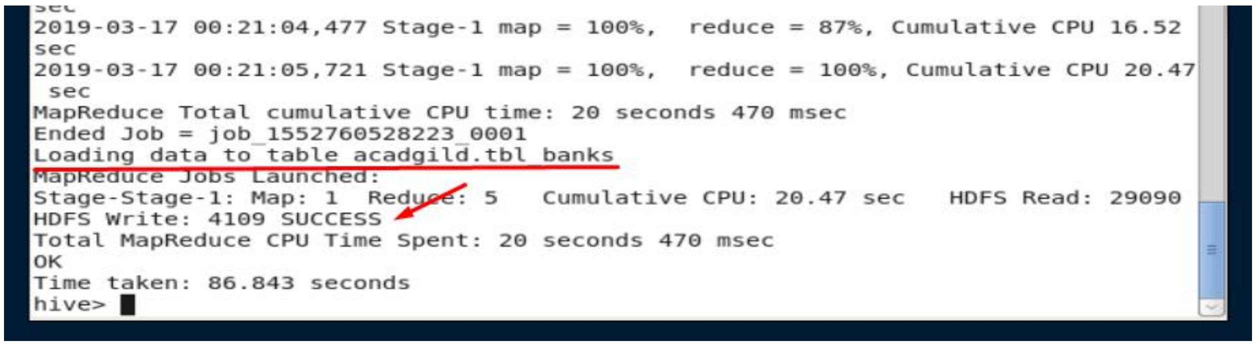
```
hive> set hive.compactor.worker.threads=2;
hive> CREATE TABLE tbl_banks(bank_id int,bank_name string,bank_code string,ho_lo
cation string) clustered by (bank_id) into 5 buckets stored as orc TBLPROPERTIES
('transactional'='true');
OK
Time taken: 1.469 seconds
hive> show tables;
OK
tbl_banks
Time taken: 0.085 seconds, Fetched: 1 row(s)
hive>
```

Insert data in to Hive table,

INSERT INTO TABLE tbl_banks VALUES (101,'Citibank','CITI','New York'),(102,'Bank of America','BOFA','New York'),(103,'JP Morgan Chase','JPMC','Washington'),(104,'Morgan Stanley','MRST','Los Angeles');



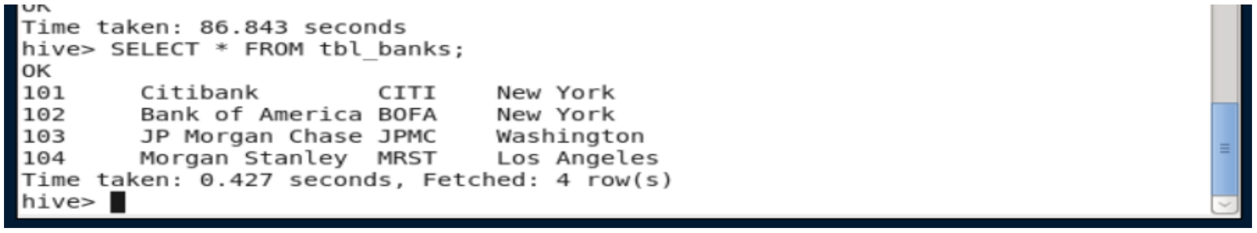
```
Time taken: 0.085 seconds, Fetched: 1 row(s)
hive> INSERT INTO TABLE tbl_banks VALUES (101,'Citibank','CITI','New York'),(102
,'Bank of America','BOFA','New York'),(103,'JP Morgan Chase','JPMC','Washington'
),(104,'Morgan Stanley','MRST','Los Angeles');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20190317001942_5f35648a-a4e5-4f66-b49e-2c436922cbc9
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_1552760528223_0001, Tracking URL = http://localhost:8088/prox
```



```
2019-03-17 00:21:04,477 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 16.52
sec
2019-03-17 00:21:05,721 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 20.47
 sec
MapReduce Total cumulative CPU time: 20 seconds 470 msec
Ended Job = job_1552760528223_0001
Loading data to table acadgild.tbl_banks
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 20.47 sec   HDFS Read: 29090
HDFS Write: 4109 SUCCESS
Total MapReduce CPU Time Spent: 20 seconds 470 msec
OK
Time taken: 86.843 seconds
hive>
```

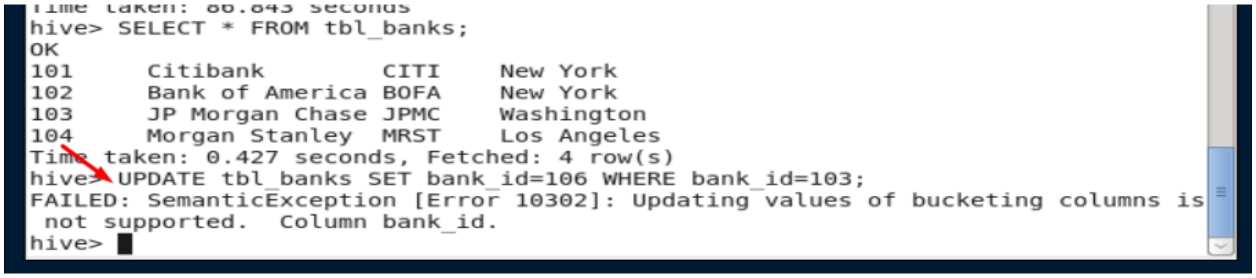
Query the inserted data and display using SELECT statement,

SELECT * FROM tbl_banks;



```
Time taken: 86.843 seconds
hive> SELECT * FROM tbl_banks;
OK
101     Citibank        CITI    New York
102     Bank of America BOFA    New York
103     JP Morgan Chase JPMC    Washington
104     Morgan Stanley  MRST    Los Angeles
Time taken: 0.427 seconds, Fetched: 4 row(s)
hive>
```

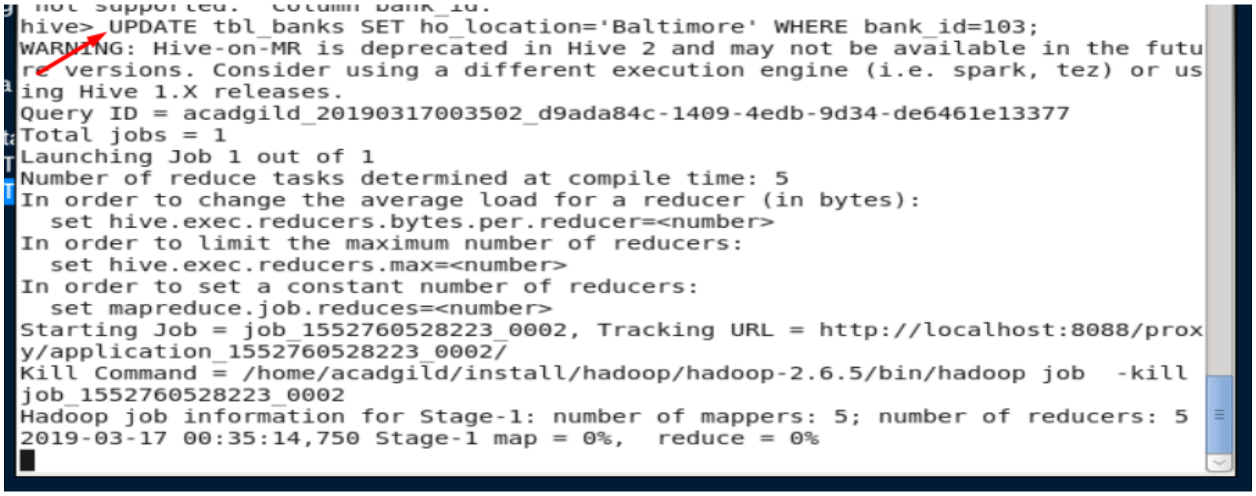
Try to update a bucketed column. In tbl_banks table, bank_id column is classified as bucketed. So, it should not be allowed to update and show error message as below,



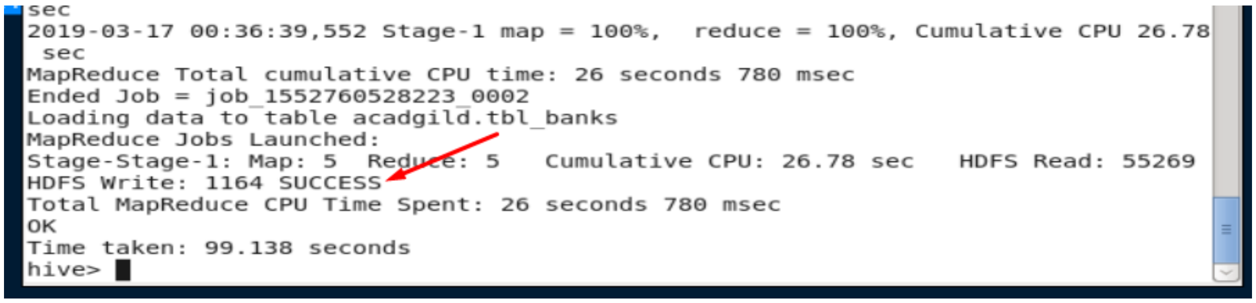
```
hive> SELECT * FROM tbl_banks;
OK
101     Citibank         CITI    New York
102     Bank of America  BOFA    New York
103     JP Morgan Chase  JPMC    Washington
104     Morgan Stanley   MRST    Los Angeles
Time taken: 0.427 seconds, Fetched: 4 row(s)
hive> UPDATE tbl_banks SET bank_id=106 WHERE bank_id=103;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is
 not supported.  Column bank_id.
hive>
```

Update a non-bucketed column value and the result will appear below,

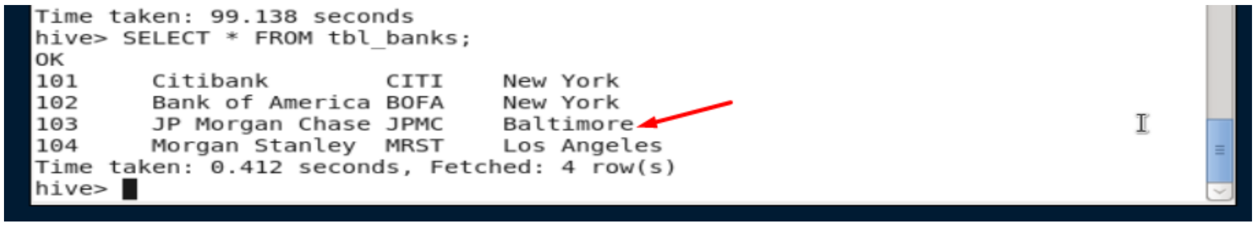
UPDATE tbl_banks SET ho_location='Baltimore' WHERE bank_id=103;



```
hive> UPDATE tbl_banks SET ho_location='Baltimore' WHERE bank_id=103;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20190317003502_d9ada84c-1409-4edb-9d34-de6461e13377
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_1552760528223_0002, Tracking URL = http://localhost:8088/prox
y/application_1552760528223_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1552760528223_0002
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2019-03-17 00:35:14,750 Stage-1 map = 0%,  reduce = 0%
```



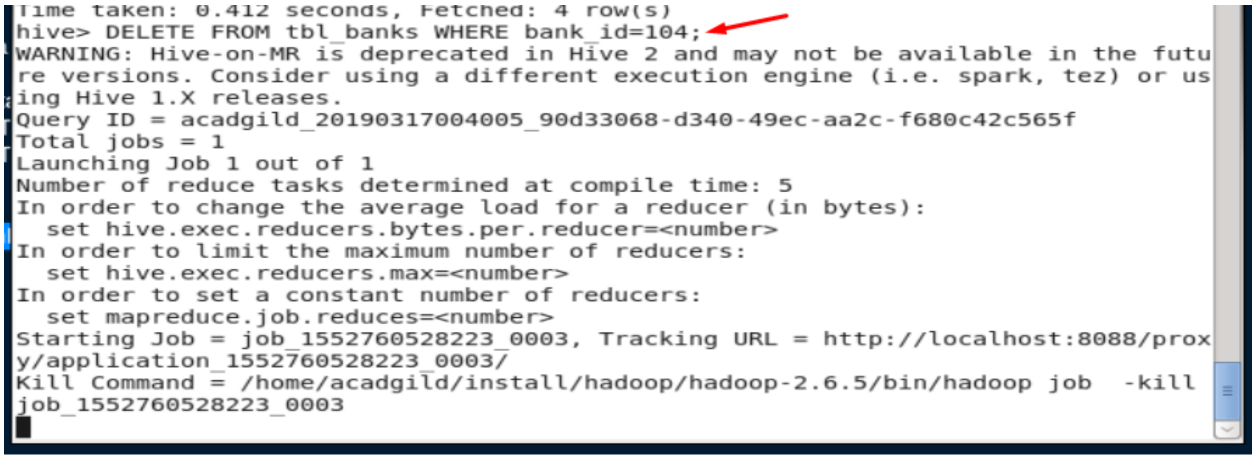
```
sec
2019-03-17 00:36:39,552 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 26.78
 sec
MapReduce Total cumulative CPU time: 26 seconds 780 msec
Ended Job = job_1552760528223_0002
Loading data to table acadgild.tbl_banks
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5   Reduce: 5   Cumulative CPU: 26.78 sec   HDFS Read: 55269
HDFS Write: 1164 SUCCESS
Total MapReduce CPU Time Spent: 26 seconds 780 msec
OK
Time taken: 99.138 seconds
hive>
```



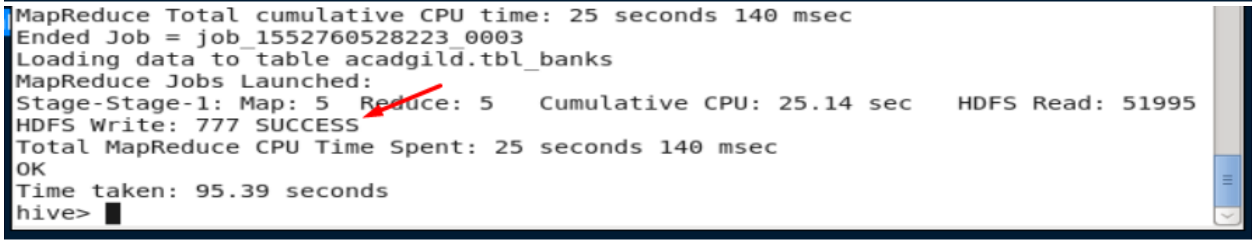
```
Time taken: 99.138 seconds
hive> SELECT * FROM tbl_banks;
OK
101     Citibank        CITI    New York
102     Bank of America BOFA    New York
103     JP Morgan Chase JPMC    Baltimore
104     Morgan Stanley  MRST    Los Angeles
Time taken: 0.412 seconds, Fetched: 4 row(s)
hive>
```

Delete an existing row in hive table,

DELETE FROM tbl_banks WHERE bank_id=104;

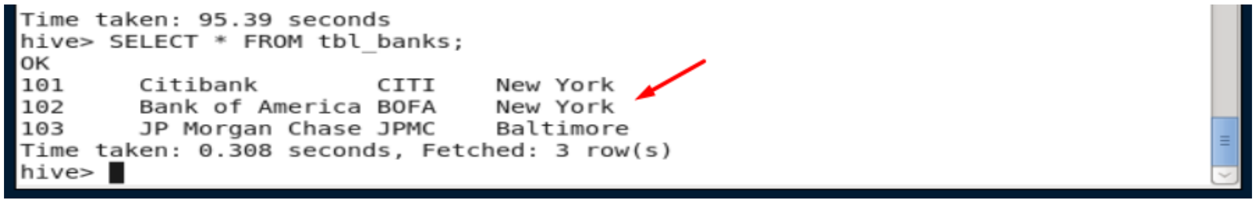


```
Time taken: 0.412 seconds, Fetched: 4 row(s)
hive> DELETE FROM tbl_banks WHERE bank_id=104;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20190317004005_90d33068-d340-49ec-aa2c-f680c42c565f
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_1552760528223_0003, Tracking URL = http://localhost:8088/prox
y/application_1552760528223_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1552760528223_0003
```



```
MapReduce Total cumulative CPU time: 25 seconds 140 msec
Ended Job = job_1552760528223_0003
Loading data to table acadgild.tbl_banks
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5   Reduce: 5   Cumulative CPU: 25.14 sec   HDFS Read: 51995
HDFS Write: 777 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 140 msec
OK
Time taken: 95.39 seconds
hive>
```

Result:



```
Time taken: 95.39 seconds
hive> SELECT * FROM tbl_banks;
OK
101     Citibank        CITI    New York
102     Bank of America BOFA    New York
103     JP Morgan Chase JPMC    Baltimore
Time taken: 0.308 seconds, Fetched: 3 row(s)
hive>
```