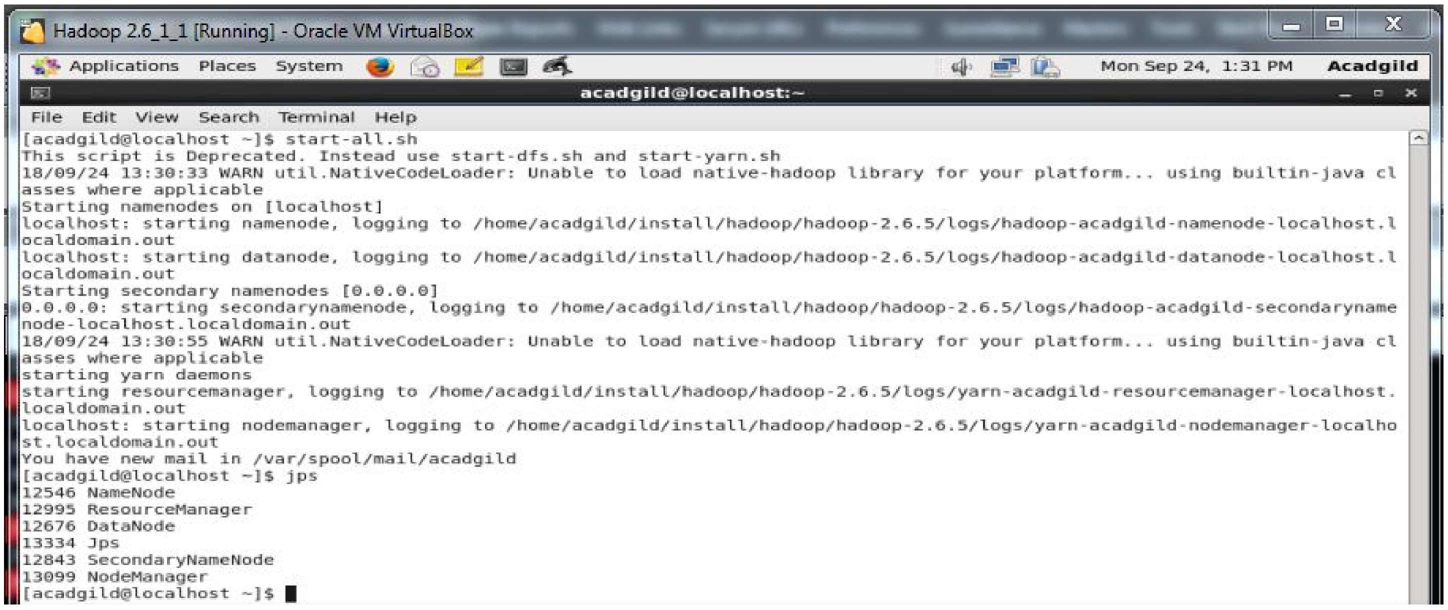
Session 9: Advance Hive Assignment 1.

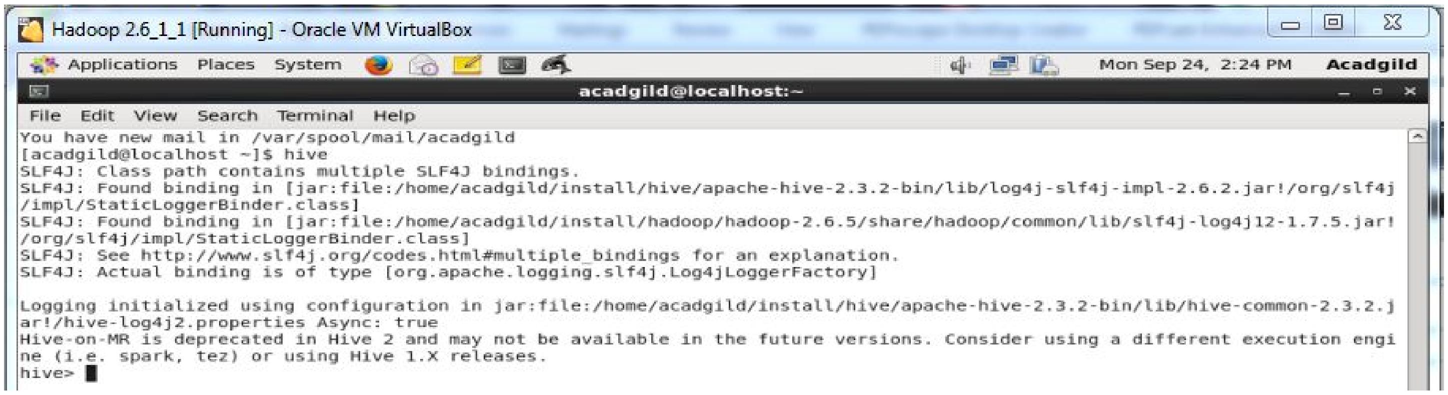
1. To perform the Hive operations, first we need to start the hadoop in VM as follows:
   $start-all.sh



```
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/09/24 13:30:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.l
ocaldomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.l
ocaldomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondaryname
node-localhost.localdomain.out
18/09/24 13:30:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.
localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localho
st.localdomain.out
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ jps
12546 NameNode
12995 ResourceManager
12676 DataNode
13334 Jps
12843 SecondaryNameNode
13099 NodeManager
[acadgild@localhost ~]$
```

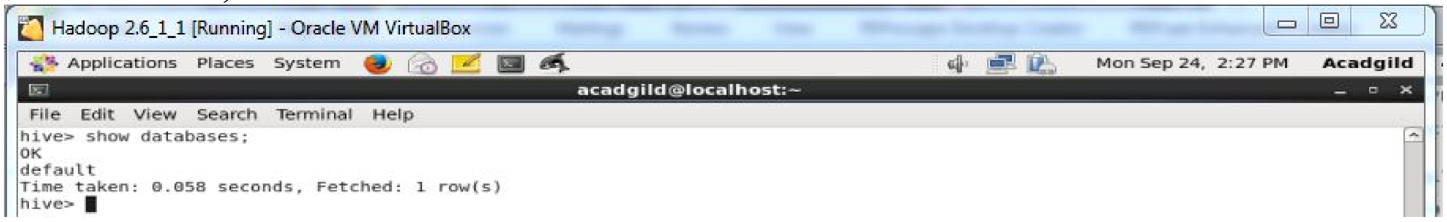
2. Check wither all the daemons are started in the hadoop as shown in the figure with the command:
   $jps

3. Now start the hive terminal with the following command:
   $hive



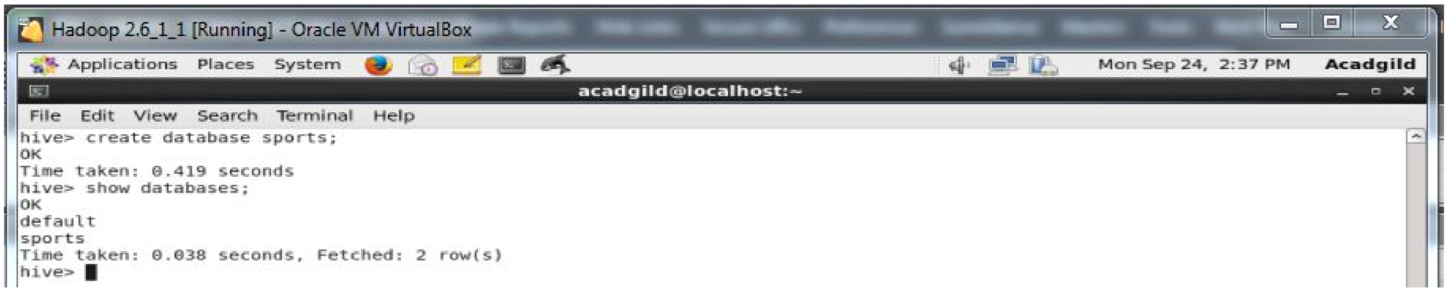
```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j
/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-2.3.2.j
ar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engi
ne (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

4. Now we can see what are all the databases available, with the following command:
$show databases;



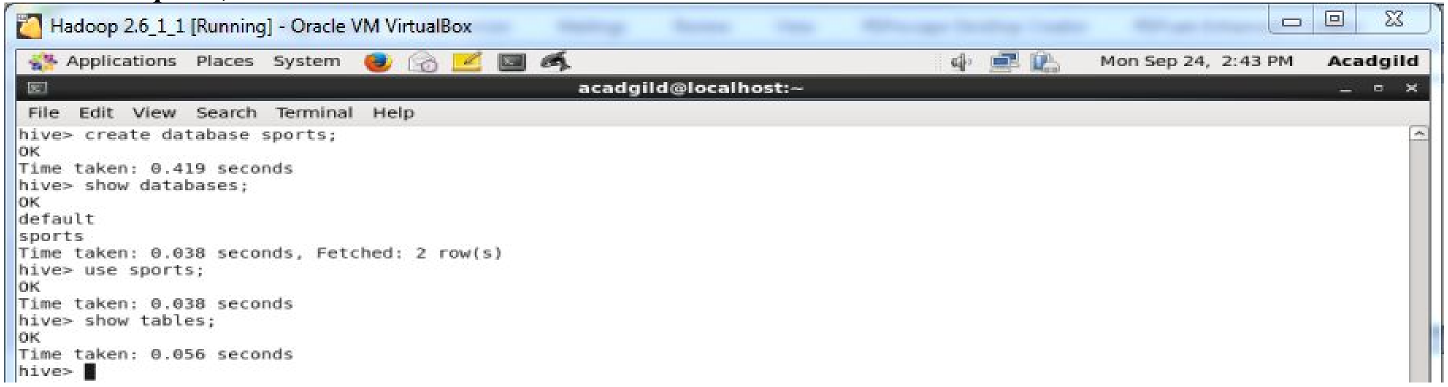
```
hive> show databases;
OK
default
Time taken: 0.058 seconds, Fetched: 1 row(s)
hive>
```

5. We can observe that there is only database available, i.e., default.
6. Now we can create a new database called **sports** as follows:
   hive>create database sports;


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 2:37 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help

```
hive> create database sports;
OK
Time taken: 0.419 seconds
hive> show databases;
OK
default
sports
Time taken: 0.038 seconds, Fetched: 2 row(s)
hive> 
```

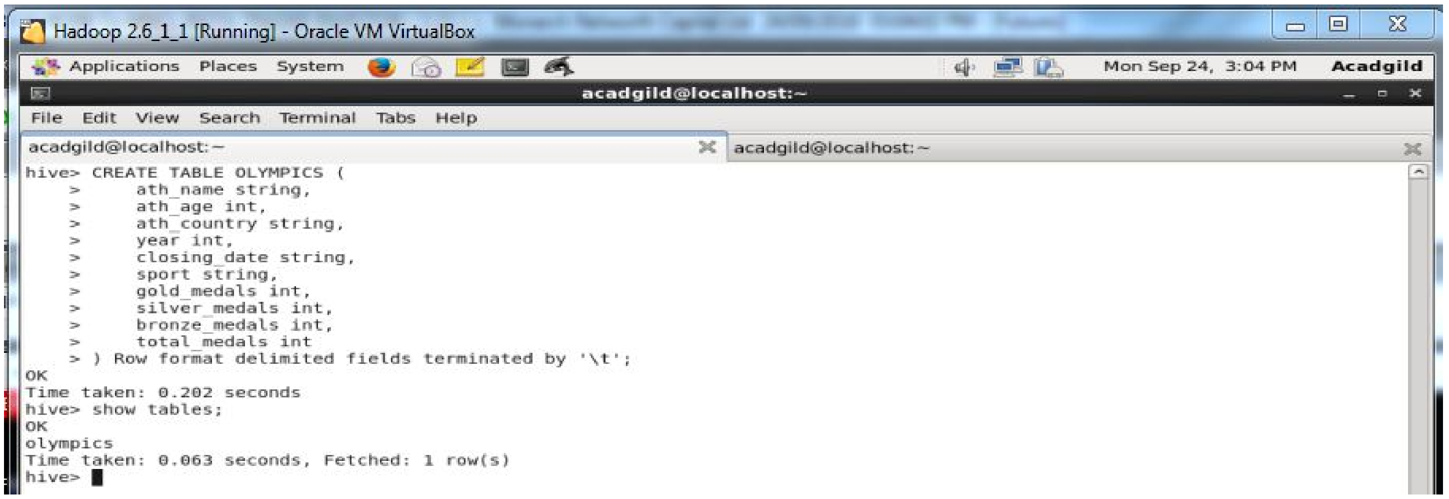
7. No to create tables we need to select the database lets say ***sports***:
   $use sports;


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 2:43 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Help

```
hive> create database sports;
OK
Time taken: 0.419 seconds
hive> show databases;
OK
default
sports
Time taken: 0.038 seconds, Fetched: 2 row(s)
hive> use sports;
OK
Time taken: 0.038 seconds
hive> show tables;
OK
Time taken: 0.056 seconds
hive> 
```

8. We can also see that there are no tables present in the database ***sprots***.
9. Now we can create a table called ***olympics*** inside the database ***sports*** as follows:
   CREATE TABLE OLYMPICS (
   ath_name string,
   ath_age int,
   ath_country string,
   year int,
   closing_date string,
   sport string,
   gold_medals int,
   silver_medals int,
   bronze_medals int,
   total_medals int

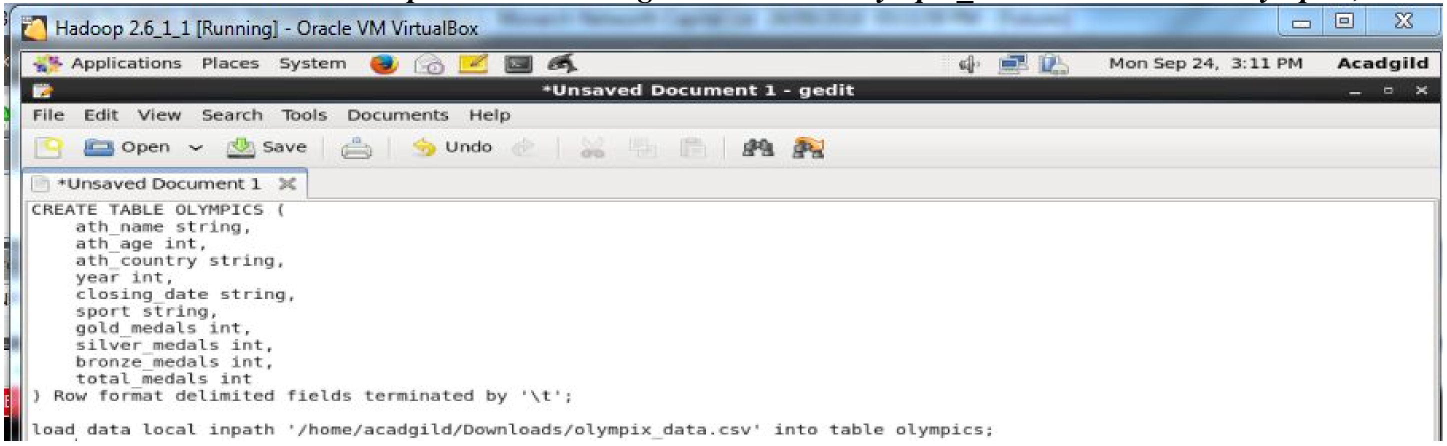
)Row format delimited fields terminated by '\t';


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 3:04 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
hive> CREATE TABLE OLYMPICS (
    >     ath_name string,
    >     ath_age int,
    >     ath_country string,
    >     year int,
    >     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: 0.202 seconds
hive> show tables;
OK
olympics
Time taken: 0.063 seconds, Fetched: 1 row(s)
hive> 
```

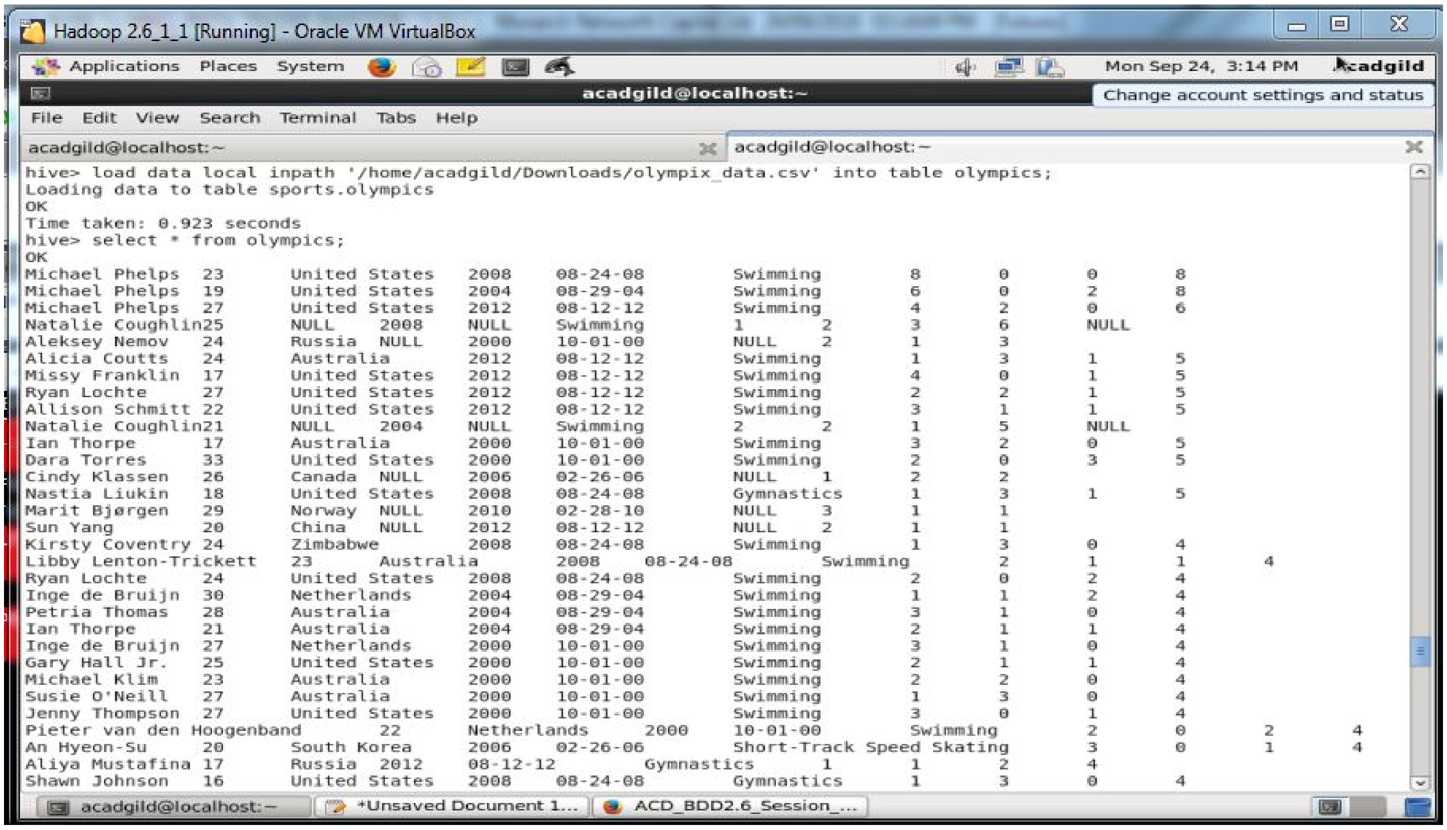
10. We can observe that the table is create with name Olympics and we can verify that table creation with ***show tables*** command as follows:
    hive> show tables.

11. Now enter load data from the .csv file into the hive table as follows:
    hive> load data local inpath '/home/acadgild/Downloads/olympix_data.csv' into table Olympics;


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 3:11 PM Acadgild
*Unsaved Document 1 - gedit
File Edit View Search Tools Documents Help
Open Save Undo
*Unsaved Document 1

```
CREATE TABLE OLYMPICS (
    ath_name string,
    ath_age int,
    ath_country string,
    year int,
    closing_date string,
    sport string,
    gold_medals int,
    silver_medals int,
    bronze_medals int,
    total_medals int
) Row format delimited fields terminated by '\t';

load data local inpath '/home/acadgild/Downloads/olympix_data.csv' into table olympics;
```

12. Verify the records present in the table with ***selec*** command as follows:
$select * from olympics;

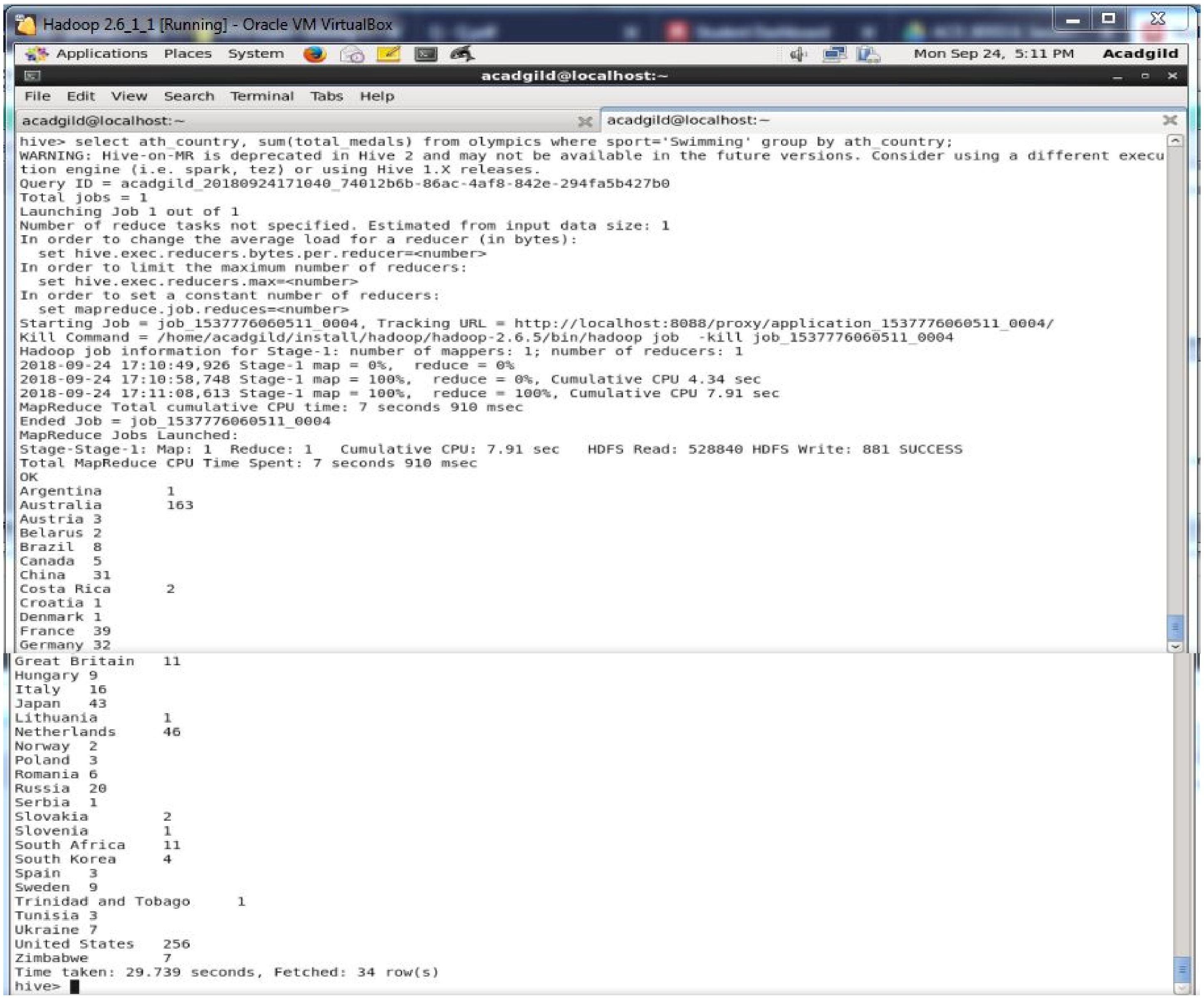

Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 3:14 PM Acadgild
acadgild@localhost:~
Change account settings and status
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~
acadgild@localhost:~
*Unsaved Document 1...
ACD_BDD2.6_Session_...

```
hive> load data local inpath '/home/acadgild/Downloads/olympix_data.csv' into table olympics;
Loading data to table sports.olympics
OK
Time taken: 0.923 seconds
hive> select * from olympics;
OK
Michael Phelps  23      United States   2008    08-24-08        Swimming        8       0       0       8
Michael Phelps  19      United States   2004    08-29-04        Swimming        6       0       2       8
Michael Phelps  27      United States   2012    08-12-12        Swimming        4       2       0       6
Natalie Coughlin25      NULL    2008    NULL    Swimming        1       2       3       6       NULL
Aleksey Nemov   24      Russia  NULL    2000    10-01-00        NULL    2       1       3
Alicia Coutts   24      Australia       2012    08-12-12        Swimming        1       3       1       5
Missy Franklin  17      United States   2012    08-12-12        Swimming        4       0       1       5
Ryan Lochte     27      United States   2012    08-12-12        Swimming        2       2       1       5
Allison Schmitt 22      United States   2012    08-12-12        Swimming        3       1       1       5
Natalie Coughlin21      NULL    2004    NULL    Swimming        2       2       1       5       NULL
Ian Thorpe      17      Australia       2000    10-01-00        Swimming        3       2       0       5
Dara Torres     33      United States   2000    10-01-00        Swimming        2       0       3       5
Cindy Klassen   26      Canada  NULL    2006    02-26-06        NULL    1       2       2
Nastia Liukin   18      United States   2008    08-24-08        Gymnastics      1       3       1       5
Marit Bjørgen   29      Norway  NULL    2010    02-28-10        NULL    3       1       1
Sun Yang        20      China   NULL    2012    08-12-12        NULL    2       1       1
Kirsty Coventry 24      Zimbabwe        2008    08-24-08        Swimming        1       3       0       4
Libby Lenton-Trickett   23      Australia       2008    08-24-08        Swimming        2       1       1       4
Ryan Lochte     24      United States   2008    08-24-08        Swimming        2       0       2       4
Inge de Bruijn  30      Netherlands     2004    08-29-04        Swimming        1       1       2       4
Petria Thomas   28      Australia       2004    08-29-04        Swimming        3       1       0       4
Ian Thorpe      21      Australia       2004    08-29-04        Swimming        2       1       1       4
Inge de Bruijn  27      Netherlands     2000    10-01-00        Swimming        3       1       0       4
Gary Hall Jr.   25      United States   2000    10-01-00        Swimming        2       1       1       4
Michael Klim    23      Australia       2000    10-01-00        Swimming        2       2       0       4
Susie O'Neill   27      Australia       2000    10-01-00        Swimming        1       3       0       4
Jenny Thompson  27      United States   2000    10-01-00        Swimming        3       0       1       4
Pieter van den Hoogenband       22      Netherlands     2000    10-01-00        Swimming        2       0       2       4
An Hyeon-Su     20      South Korea     2006    02-26-06        Short-Track Speed Skating       3       0       1       4
Aliya Mustafina 17      Russia  2012    08-12-12        Gymnastics      1       1       2       4
Shawn Johnson   16      United States   2008    08-24-08        Gymnastics      1       3       0       4
```

Task 1:

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

1. For this we can write a Hive query as follows:
   hive> select ath_country, sum(total_medals) from Olympics where sport='Swimming' group by ath_country;

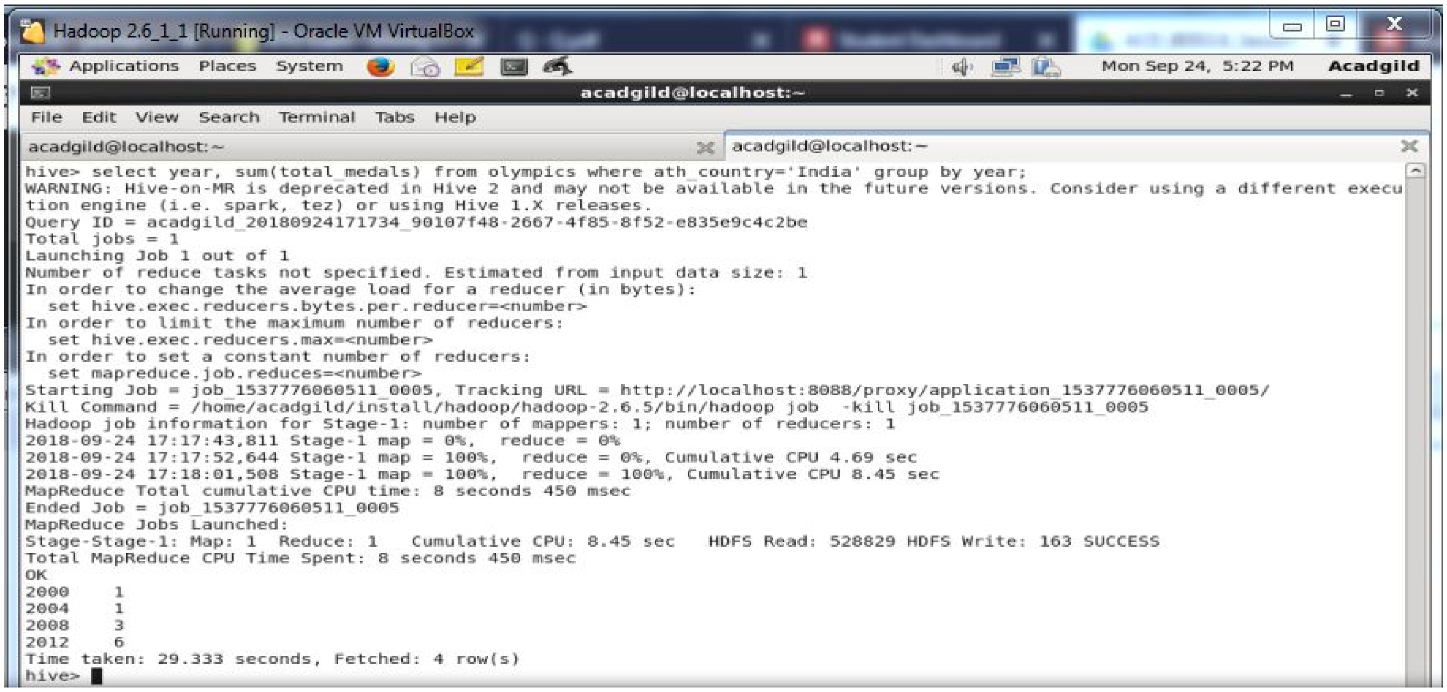

Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 5:11 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
hive> select ath_country, sum(total_medals) from olympics where sport='Swimming' group by ath_country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180924171040_74012b6b-86ac-4af8-842e-294fa5b427b0
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_1537776060511_0004, Tracking URL = http://localhost:8088/proxy/application_1537776060511_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537776060511_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-24 17:10:49,926 Stage-1 map = 0%,  reduce = 0%
2018-09-24 17:10:58,748 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.34 sec
2018-09-24 17:11:08,613 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.91 sec
MapReduce Total cumulative CPU time: 7 seconds 910 msec
Ended Job = job_1537776060511_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.91 sec   HDFS Read: 528840 HDFS Write: 881 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 910 msec
OK
Argentina       1
Australia       163
Austria 3
Belarus 2
Brazil  8
Canada  5
China   31
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   256
Zimbabwe        7
Time taken: 29.739 seconds, Fetched: 34 row(s)
hive>
```

2. We can observe the output that each country and corresponding total number of medals.

3. Write a hive program to find the number of medals that India won year wise.

1. For this task we need to write a hive query as follows:
   hive>select year, sum(total_medals) from Olympics where ath_country='India' group by year;

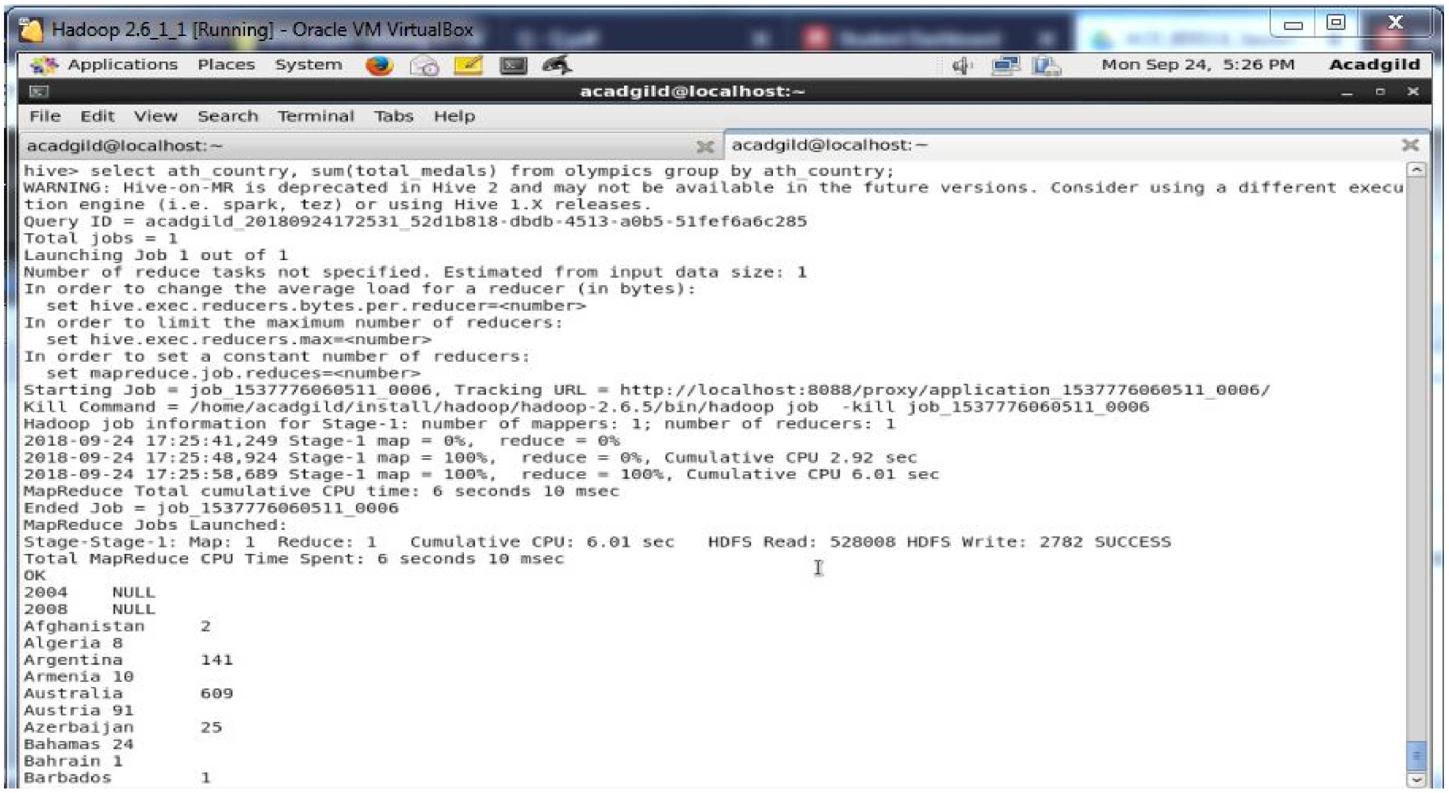

Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 5:22 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
hive> select year, sum(total_medals) from olympics where ath_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 execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180924171734_90107f48-2667-4f85-8f52-e835e9c4c2be
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_1537776060511_0005, Tracking URL = http://localhost:8088/proxy/application_1537776060511_0005/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537776060511_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-24 17:17:43,811 Stage-1 map = 0%,  reduce = 0%
2018-09-24 17:17:52,644 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.69 sec
2018-09-24 17:18:01,508 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.45 sec
MapReduce Total cumulative CPU time: 8 seconds 450 msec
Ended Job = job_1537776060511_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.45 sec   HDFS Read: 528829 HDFS Write: 163 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 450 msec
OK
2000    1
2004    1
2008    3
2012    6
Time taken: 29.333 seconds, Fetched: 4 row(s)
hive>
```

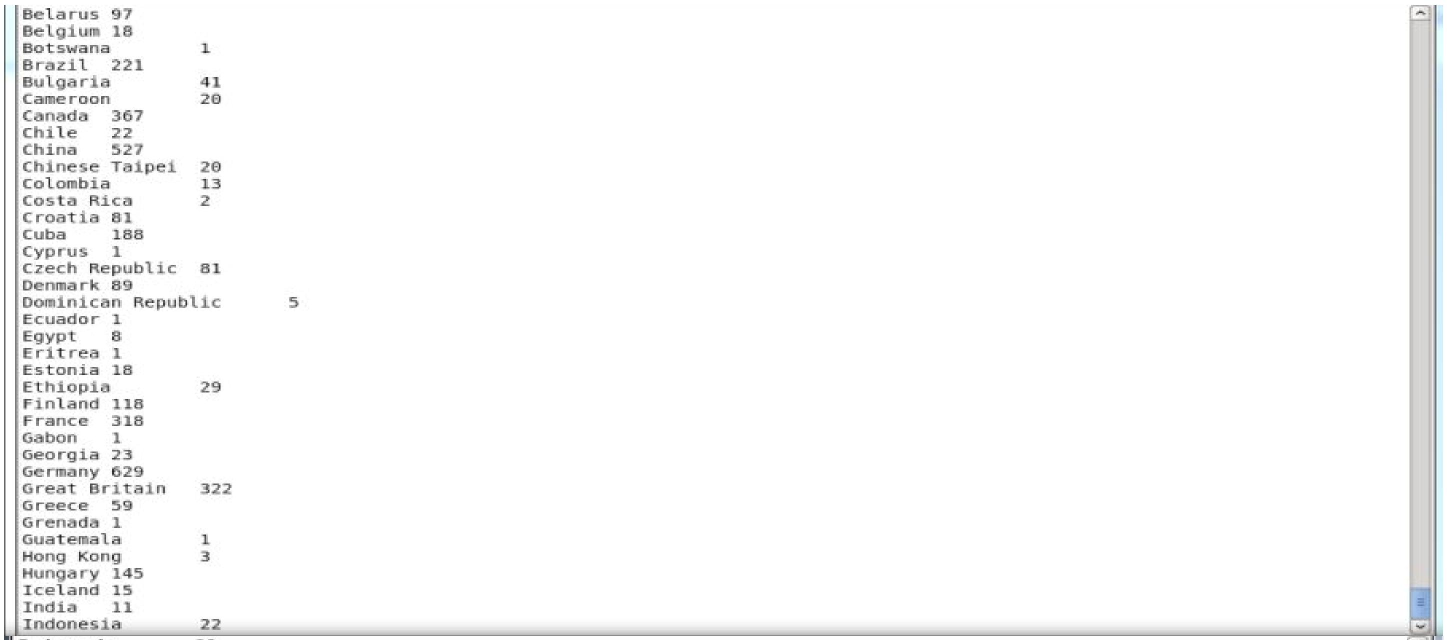
2. With the above figure we can see that the total number of medals own by India on each year.

3. Write a hive program to find the total number of medals each country own.

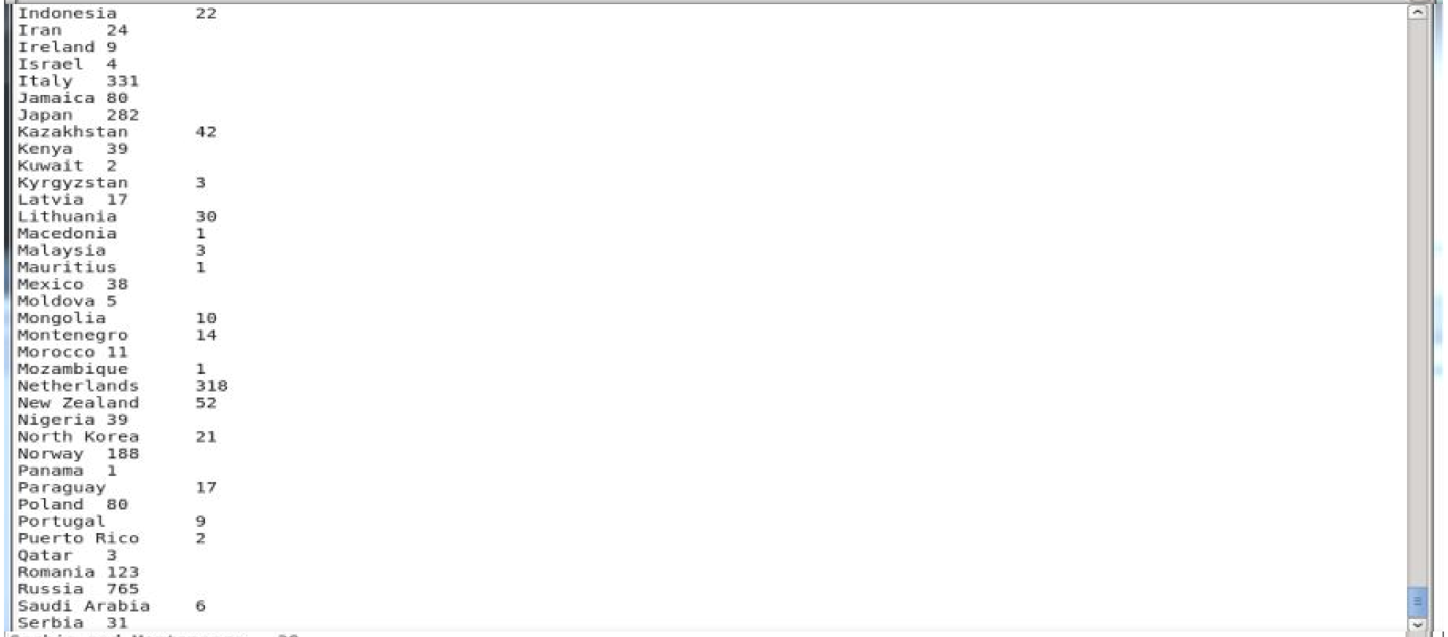
1. For this we need to write Hive query as follows:
   hive> select ath_country, sum(total_medals) from Olympics group by ath_country;


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 5:26 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

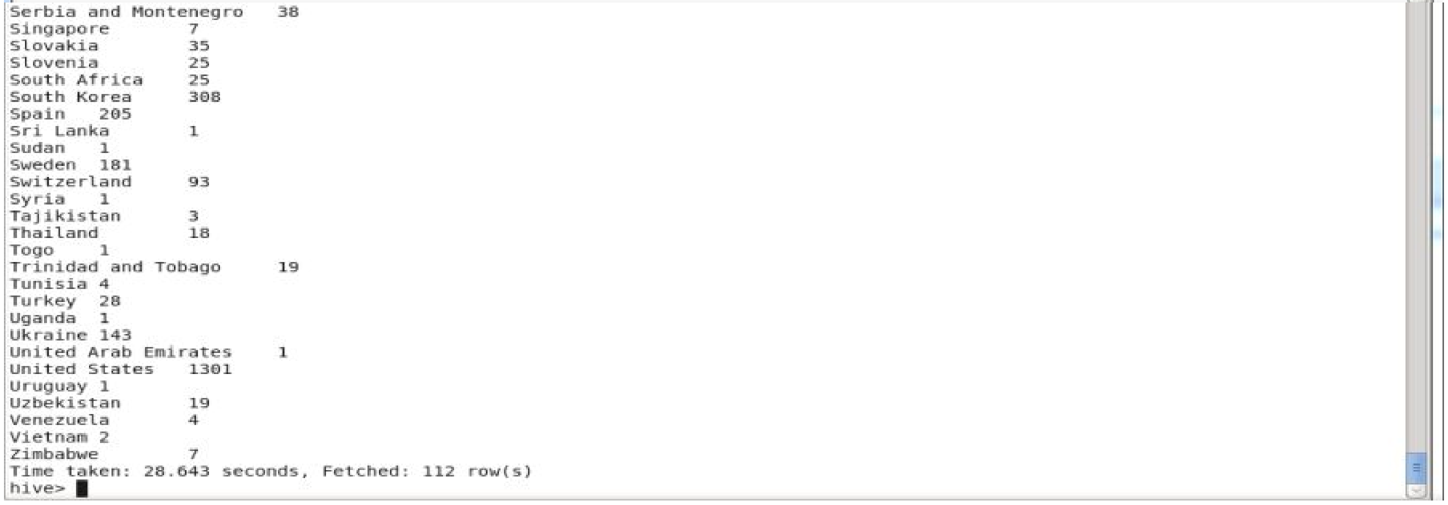
```
hive> select ath_country, sum(total_medals) from olympics group by ath_country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180924172531_52d1b818-dbdb-4513-a0b5-51fef6a6c285
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_1537776060511_0006, Tracking URL = http://localhost:8088/proxy/application_1537776060511_0006/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537776060511_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-24 17:25:41,249 Stage-1 map = 0%,  reduce = 0%
2018-09-24 17:25:48,924 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.92 sec
2018-09-24 17:25:58,689 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.01 sec
MapReduce Total cumulative CPU time: 6 seconds 10 msec
Ended Job = job_1537776060511_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.01 sec   HDFS Read: 528008 HDFS Write: 2782 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 10 msec
OK
2004    NULL
2008    NULL
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  367
Chile   22
China   527
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
```



```
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  188
Panama  1
Paraguay        17
Poland  80
Portugal        9
Puerto Rico     2
Qatar   3
Romania 123
Russia  765
Saudi Arabia    6
Serbia  31
```



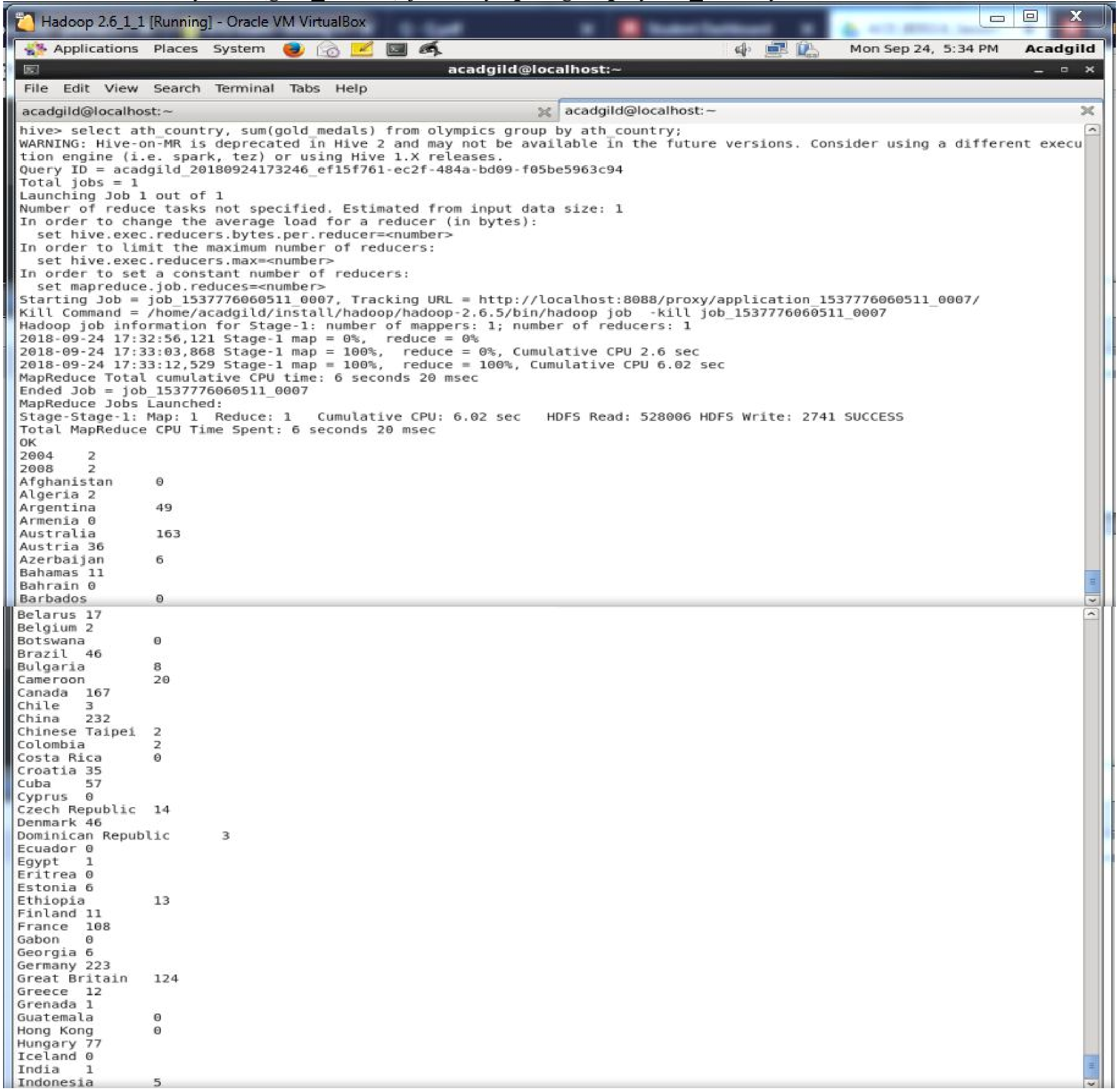
```
Serbia and Montenegro   38
Singapore       7
Slovakia        35
Slovenia        25
South Africa    25
South Korea     308
Spain   205
Sri Lanka       1
Sudan   1
Sweden  181
Switzerland     93
Syria   1
Tajikistan      3
Thailand        18
Togo    1
Trinidad and Tobago     19
Tunisia 4
Turkey  28
Uganda  1
Ukraine 143
United Arab Emirates    1
United States   1301
Uruguay 1
Uzbekistan      19
Venezuela       4
Vietnam 2
Zimbabwe        7
Time taken: 28.643 seconds, Fetched: 112 row(s)
hive> 
```

2. We can observe that each country and their number of medals own.

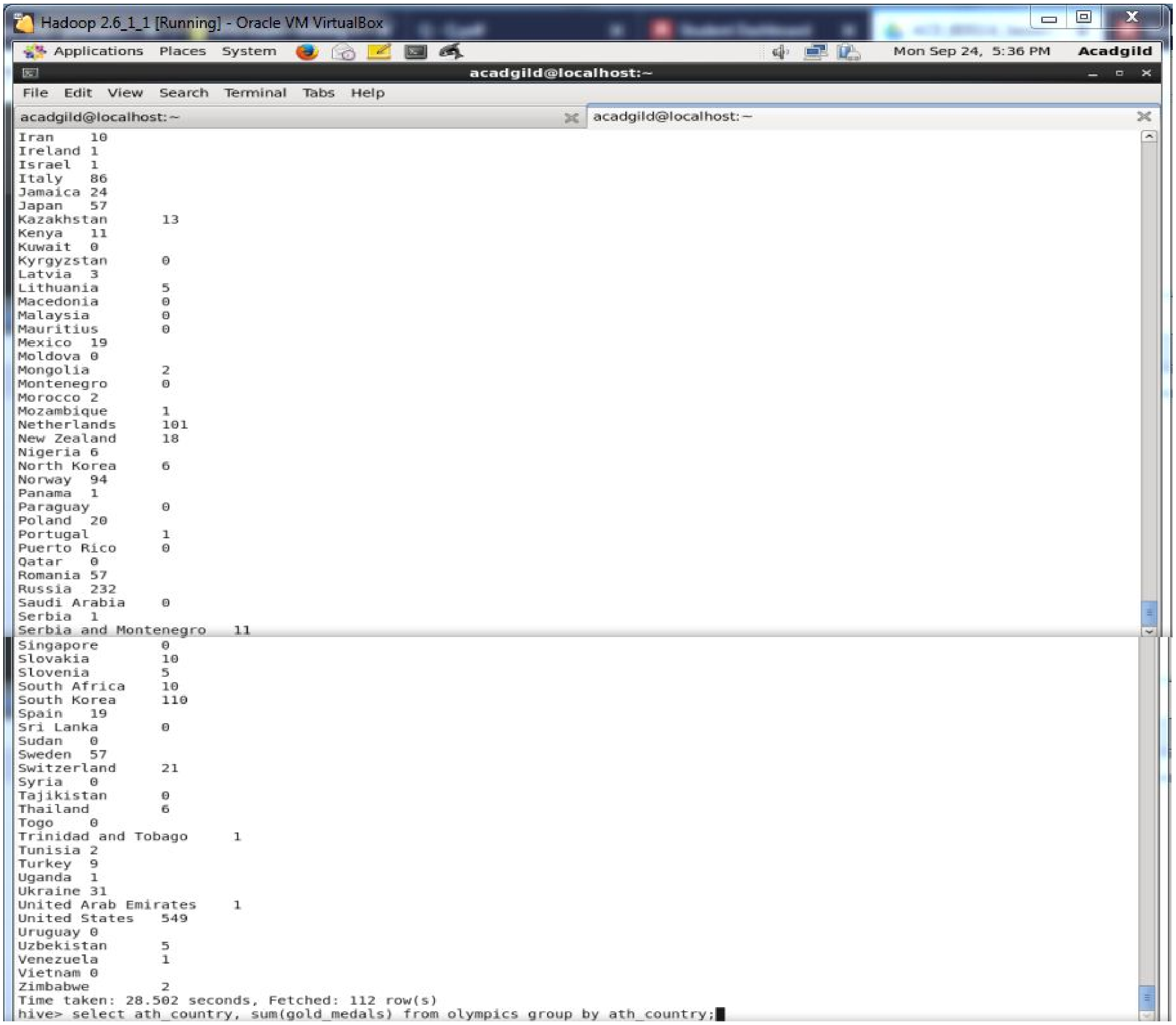
4. Write a hive program to find the number of gold medals each country own.

1. For this we need to write hive query as follows:

hive> select country, sum(gold_medals) from olympics group by ath_country;


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 5:34 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
hive> select ath_country, sum(gold_medals) from olympics group by ath_country;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180924173246_ef15f761-ec2f-484a-bd09-f05be5963c94
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_1537776060511_0007, Tracking URL = http://localhost:8088/proxy/application_1537776060511_0007/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537776060511_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-24 17:32:56,121 Stage-1 map = 0%,  reduce = 0%
2018-09-24 17:33:03,868 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.6 sec
2018-09-24 17:33:12,529 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.02 sec
MapReduce Total cumulative CPU time: 6 seconds 20 msec
Ended Job = job_1537776060511_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.02 sec   HDFS Read: 528006 HDFS Write: 2741 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 20 msec
OK
2004    2
2008    2
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  167
Chile   3
China   232
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
```


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
Mon Sep 24, 5:36 PM
Acadgild
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
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  94
Panama  1
Paraguay        0
Poland  20
Portugal        1
Puerto Rico     0
Qatar   0
Romania 57
Russia  232
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   549
Uruguay 0
Uzbekistan      5
Venezuela       1
Vietnam 0
Zimbabwe        2
Time taken: 28.502 seconds, Fetched: 112 row(s)
hive> select ath_country, sum(gold_medals) from olympics group by ath_country;
```

3. With this we can observe that the countries along with number of gold medals own.

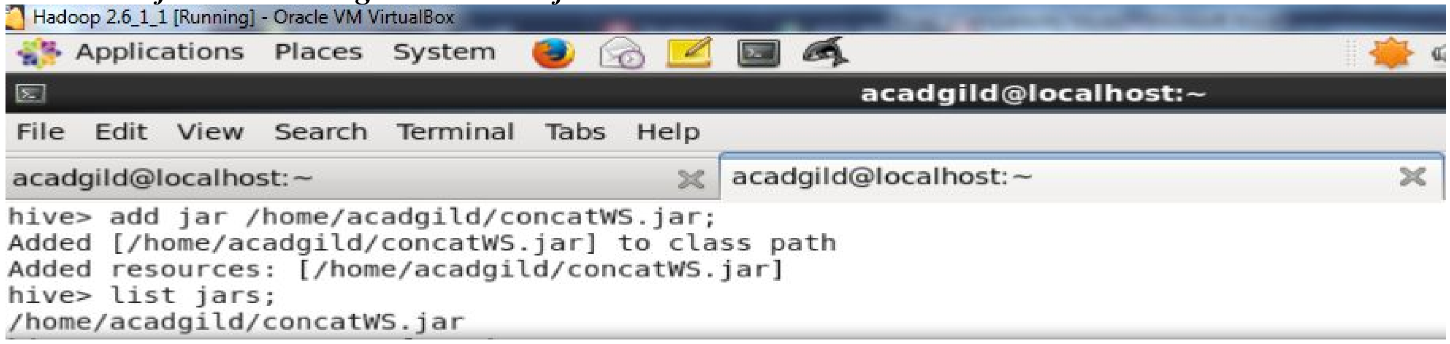
Task 2 – Write a hive UDF that implements functionality of string concat_ws(string SEP, array<string>). This UDF will accept two arguments, one string and one array of string. It will return a single string where all the elements of the array are separated by the SEP.

1. For this task first we will write the UDF in Java and the source code is as follows:

```
import org.apache.hadoop.hive.ql.udf.generic;

import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.exec.UDFArgumentLengthException;
import org.apache.hadoop.hive.ql.exec.UDFArgumentTypeException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF.DeferredObject;
import org.apache.hadoop.hive.serde.serdeConstants;
import org.apache.hadoop.hive.serde2.objectinspector.ListObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector.Category;
import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.PrimitiveObjectInspector.PrimitiveCategory;
import org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorFactory;
import org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorUtils;
import
org.apache.hadoop.hive.serde2.objectinspector.primitive.PrimitiveObjectInspectorUtils.PrimitiveGrou
ping;
import org.apache.hadoop.io.Text;

public class GenericUDFConcatWS extends GenericUDF {
  private transient ObjectInspector[] argumentOIs;

  @Override
  public ObjectInspector initialize(ObjectInspector[] arguments) throws UDFArgumentException {
    if (arguments.length < 2) {
      throw new UDFArgumentLengthException(
          "The function CONCAT_WS(separator,[string | array(string)]+) "
            + "needs at least two arguments.");
    }

    // check if argument is a string or an array of strings
    for (int i = 0; i < arguments.length; i++) {
      switch(arguments[i].getCategory()) {
        case LIST:
          if (isStringOrVoidType(
              ((ListObjectInspector) arguments[i]).getListElementObjectInspector())) {
            break;
          }
        case PRIMITIVE:
          if (isStringOrVoidType(arguments[i])) {
          break;
          }
        default:
          throw new UDFArgumentTypeException(i, "Argument " + (i + 1)
            + " of function CONCAT_WS must be \"" + serdeConstants.STRING_TYPE_NAME
            +       "     or     "     +     serdeConstants.LIST_TYPE_NAME     +     "<"     +
serdeConstants.STRING_TYPE_NAME
            + ">\", but \"" + arguments[i].getTypeName() + "\" was found.");
      }
    }

    argumentOIs = arguments;
    return PrimitiveObjectInspectorFactory.writableStringObjectInspector;
  }

  protected boolean isStringOrVoidType(ObjectInspector oi) {
    if (oi.getCategory() == Category.PRIMITIVE) {
      if (PrimitiveGrouping.STRING_GROUP
        == PrimitiveObjectInspectorUtils.getPrimitiveGrouping(
            ((PrimitiveObjectInspector) oi).getPrimitiveCategory())
        || ((PrimitiveObjectInspector) oi).getPrimitiveCategory() == PrimitiveCategory.VOID) {
      return true;
    }
  }
  return false;
}

private final Text resultText = new Text();

@Override
public Object evaluate(DeferredObject[] arguments) throws HiveException {
  if (arguments[0].get() == null) {
    return null;
  }
  String separator = PrimitiveObjectInspectorUtils.getString(
      arguments[0].get(), (PrimitiveObjectInspector)argumentOIs[0]);

  StringBuilder sb = new StringBuilder();
  boolean first = true;
  for (int i = 1; i < arguments.length; i++) {
    if (arguments[i].get() != null) {
      if (first) {
        first = false;
      } else {
        sb.append(separator);
      }
      if (argumentOIs[i].getCategory().equals(Category.LIST)) {
        Object strArray = arguments[i].get();
        ListObjectInspector strArrayOI = (ListObjectInspector) argumentOIs[i];
        boolean strArrayFirst = true;
        for (int j = 0; j < strArrayOI.getListLength(strArray); j++) {
          if (strArrayFirst) {
            strArrayFirst = false;
          } else {
            sb.append(separator);
          }
          sb.append(strArrayOI.getListElement(strArray, j));
        }
      } else {
        sb.append(PrimitiveObjectInspectorUtils.getString(
            arguments[i].get(), (PrimitiveObjectInspector)argumentOIs[i]));
      }
    }
  }

  resultText.set(sb.toString());
  return resultText;
}

@Override
```

```
  public String getDisplayString(String[] children) {
    assert (children.length >= 2);
    return getStandardDisplayString("concat_ws", children);
  }
}
```

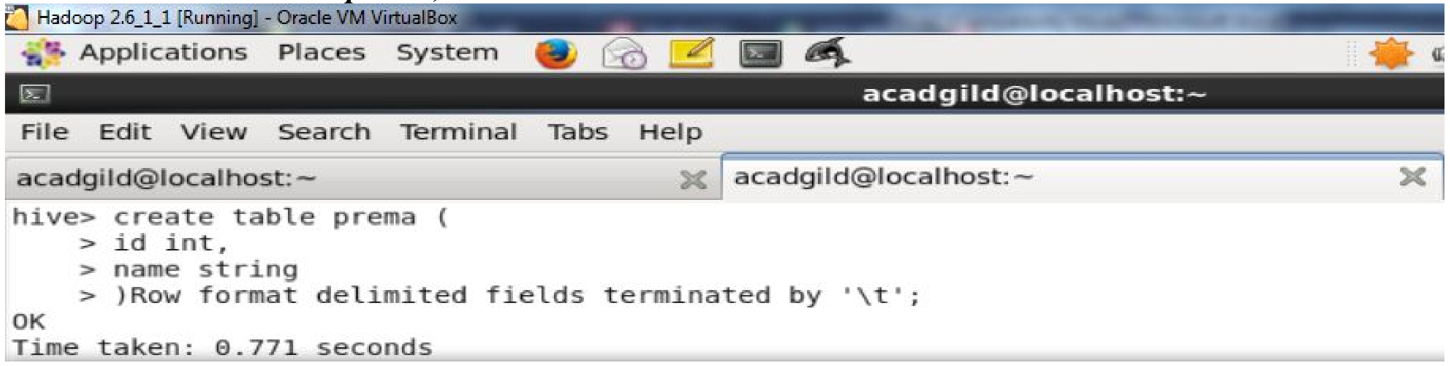
2. Now convert the above source code into ***concateWS.jar*** and load into hive as follows:
   hive> jar /home/acadgild/concatWS.jar


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
hive> add jar /home/acadgild/concatWS.jar;
Added [/home/acadgild/concatWS.jar] to class path
Added resources: [/home/acadgild/concatWS.jar]
hive> list jars;
/home/acadgild/concatWS.jar
```

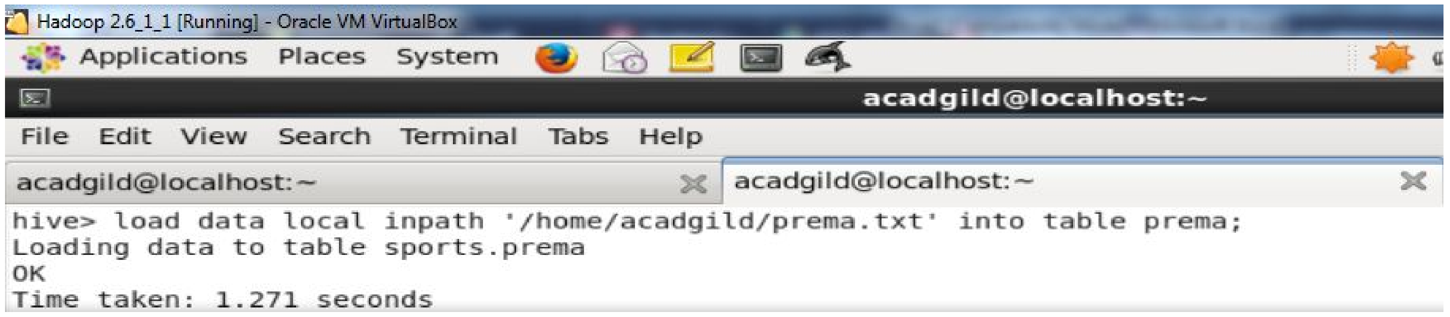
3. And we can list the jars with ***list jars*** command.

4. Now create a table ***prema***, as follows:


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

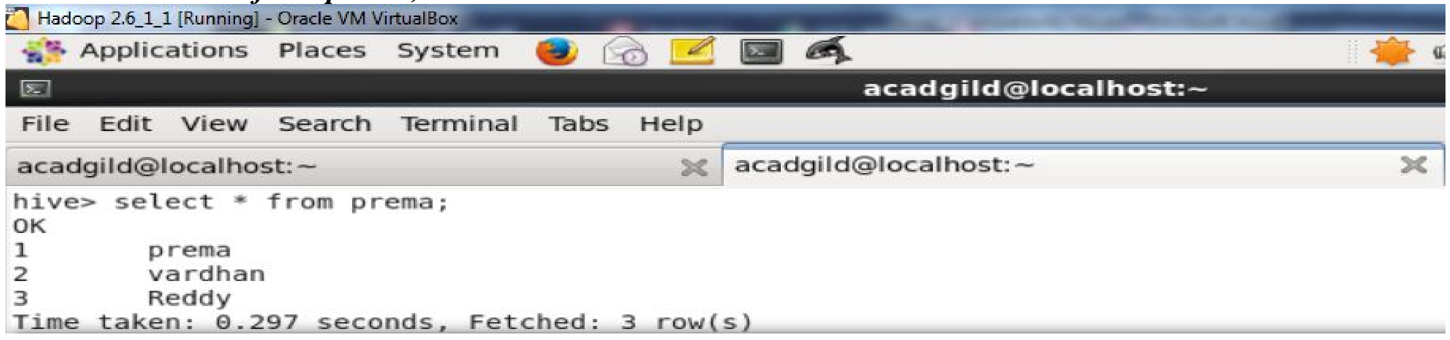
```
hive> create table prema (
    > id int,
    > name string
    > )Row format delimited fields terminated by '\t';
OK
Time taken: 0.771 seconds
```

5. Now the table is created and load some data into the table as follows:


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

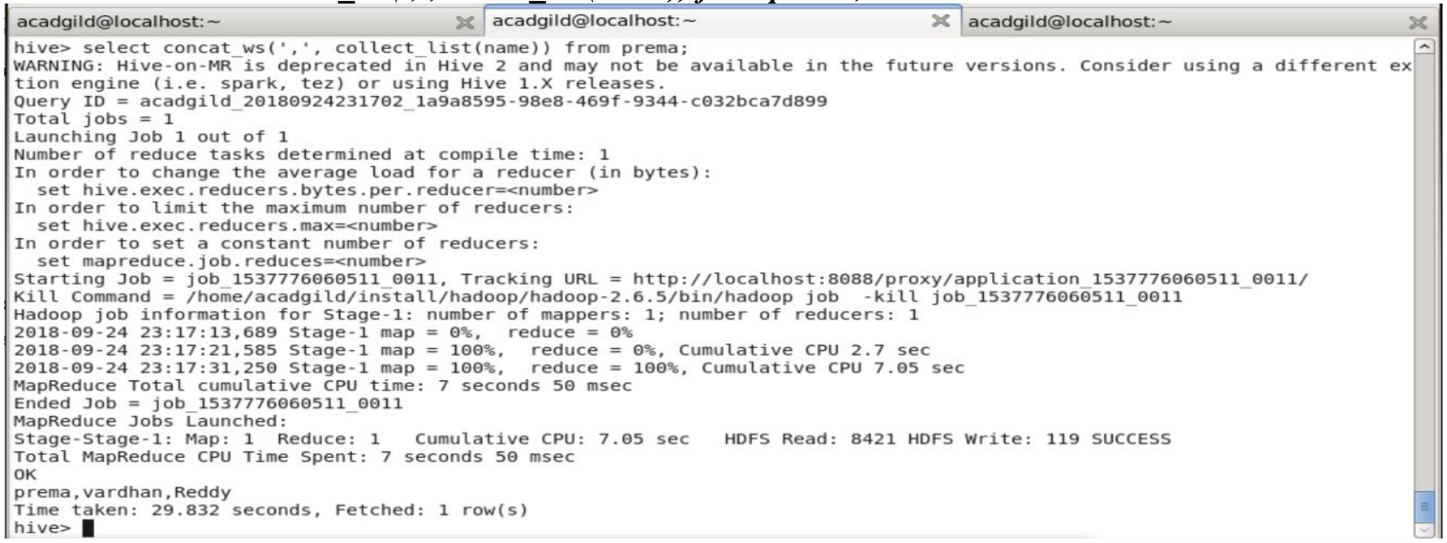
```
hive> load data local inpath '/home/acadgild/prema.txt' into table prema;
Loading data to table sports.prema
OK
Time taken: 1.271 seconds
```

6. Now select the contents from the table ***prema*** as follows:
   hive>select * from prema;


Hadoop 2.6_1_1 [Running] - Oracle VM VirtualBox
Applications Places System
acadgild@localhost:~
File Edit View Search Terminal Tabs Help
acadgild@localhost:~
acadgild@localhost:~

```
hive> select * from prema;
OK
1       prema
2       vardhan
3       Reddy
Time taken: 0.297 seconds, Fetched: 3 row(s)
```

7. Now apply the ***concat_ws*** function on names which are present in the table ***prema*** as follows:
   hive>select concat_ws(',', collect_list(name)) from prema;



```
hive> select concat_ws(',', collect_list(name)) from prema;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different ex
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180924231702_1a9a8595-98e8-469f-9344-c032bca7d899
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_1537776060511_0011, Tracking URL = http://localhost:8088/proxy/application_1537776060511_0011/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1537776060511_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-24 23:17:13,689 Stage-1 map = 0%,  reduce = 0%
2018-09-24 23:17:21,585 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.7 sec
2018-09-24 23:17:31,250 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.05 sec
MapReduce Total cumulative CPU time: 7 seconds 50 msec
Ended Job = job_1537776060511_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.05 sec   HDFS Read: 8421 HDFS Write: 119 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 50 msec
OK
prema,vardhan,Reddy
Time taken: 29.832 seconds, Fetched: 1 row(s)
hive>
```

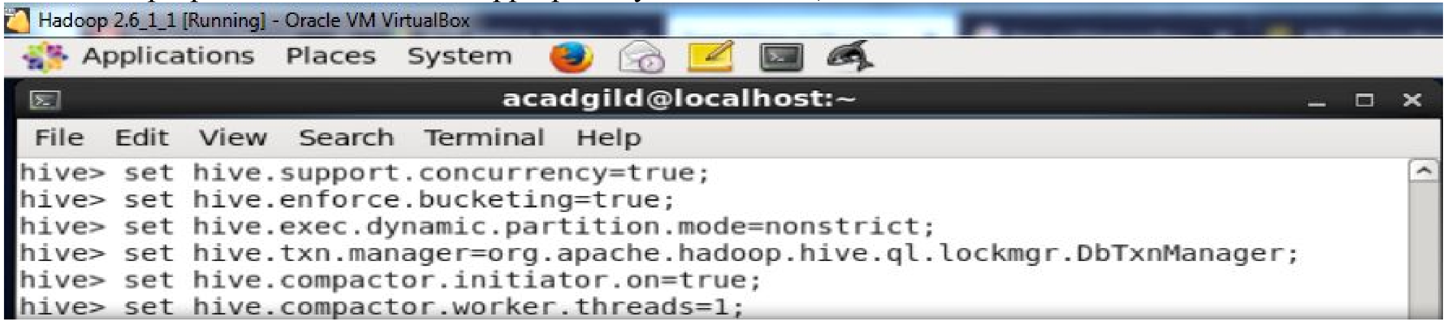
8. We can observe that in the output all the names in the table ***prema*** are separated by **','** as ***prema,vardhan,Reddy***

Task 3 – Row-level transactions available in Hive.

Transactions are provided at the row-level in Hive 0.14. The different row-level transactions available in Hive 0.14 are as follows:

1. Insert.
2. Delete.
3. Update.

The below properties needs to be set appropriately in ***hive shell,*** order-wise to work with transaction in Hive:



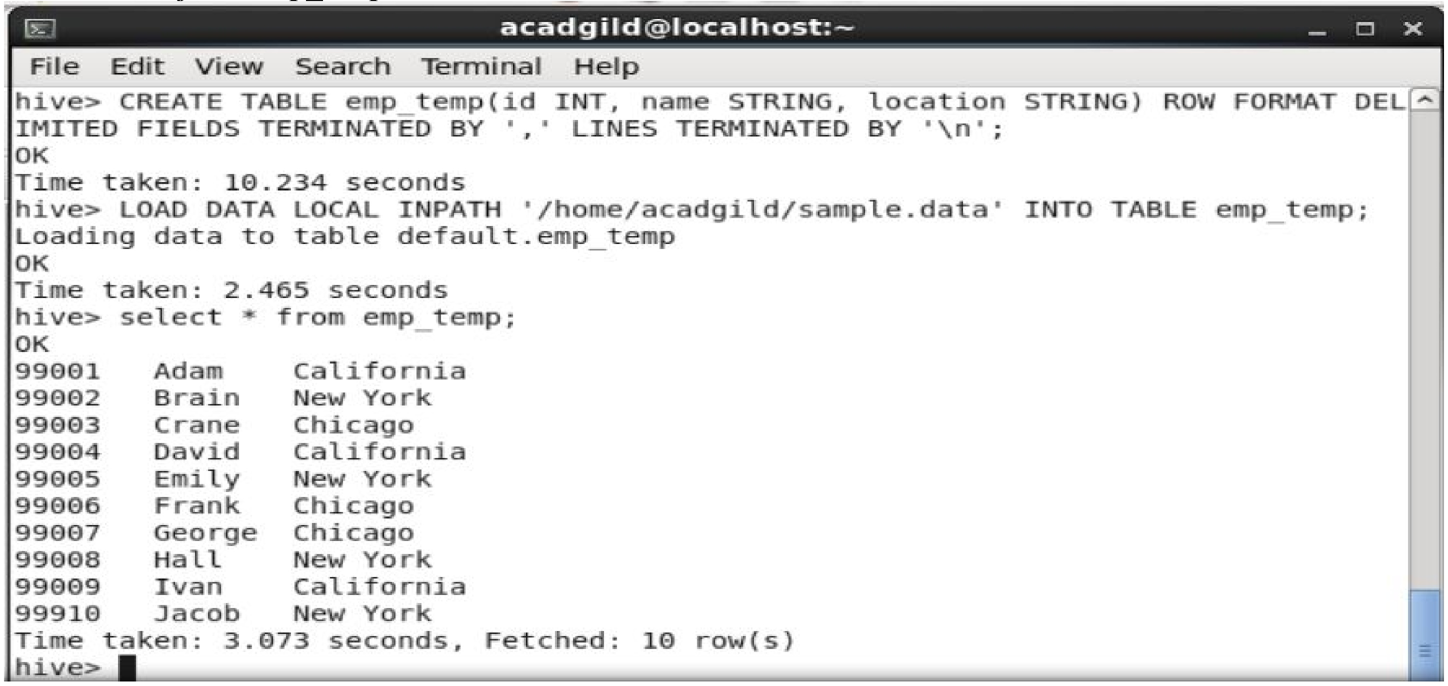
```
hive> set hive.support.concurrency=true;
hive> set hive.enforce.bucketing=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> set hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on=true;
hive> set hive.compactor.worker.threads=1;
```

The sample file ***sample.data:***
99001, Adam, California
99002, Brain, New York
99003, Crane, Chicago
99004, David, California
99005, Emily, New York
99006, Frank, Chicago
99007, George, Chicago
99008, Hall, New York
99009, Ivan, California
99010, Jacob, New York

Now create the table ***emp_temp*** that supports hive transactions as follows:
CREATE TABLE emp_temp(id INT, name STRING, location STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';

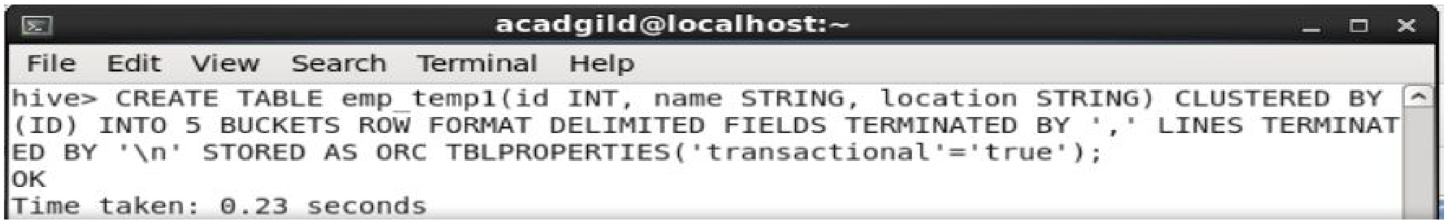
Load the data ***sample.data*** into the table ***emp_temp*** with the following command:
hive>LOAD DATA LOCAL INPATH '/home/acadgild/sample.data' INTO TABLE emp_temp;

Now display all the data that loaded into the table ***emp_temp*** as follows:
hive>select * from emp_temp;



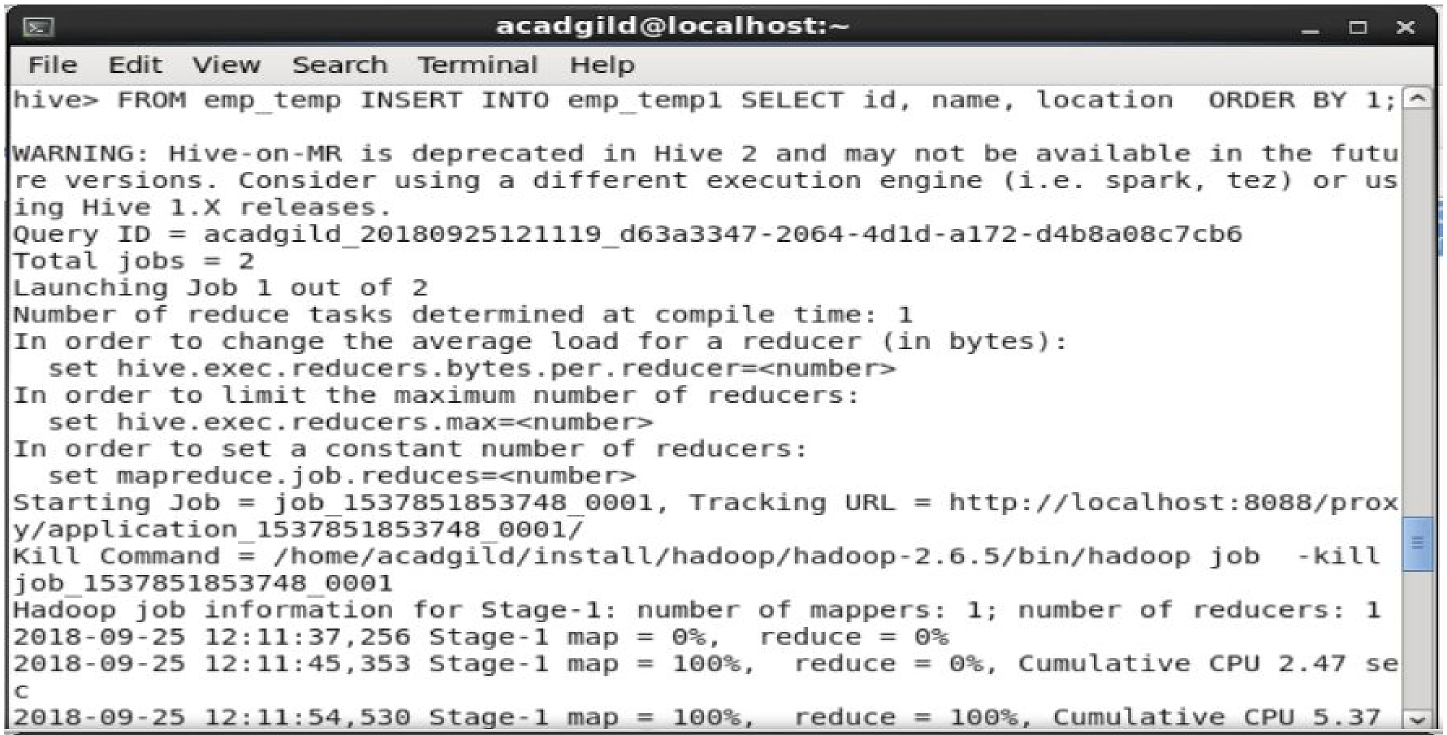
```
hive> CREATE TABLE emp_temp(id INT, name STRING, location STRING) ROW FORMAT DEL
IMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';
OK
Time taken: 10.234 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadgild/sample.data' INTO TABLE emp_temp;
Loading data to table default.emp_temp
OK
Time taken: 2.465 seconds
hive> select * from emp_temp;
OK
99001   Adam    California
99002   Brain   New York
99003   Crane   Chicago
99004   David   California
99005   Emily   New York
99006   Frank   Chicago
99007   George  Chicago
99008   Hall    New York
99009   Ivan    California
99910   Jacob   New York
Time taken: 3.073 seconds, Fetched: 10 row(s)
hive> 
```

Now create another table called ***emp_temp1*** as follows:
hive>CREATE TABLE emp_temp1(id INT, name STRING, location STRING) CLUSTERED BY (ID) INTO 5 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS ORC TBLPROPERTIES('transactional'='true');

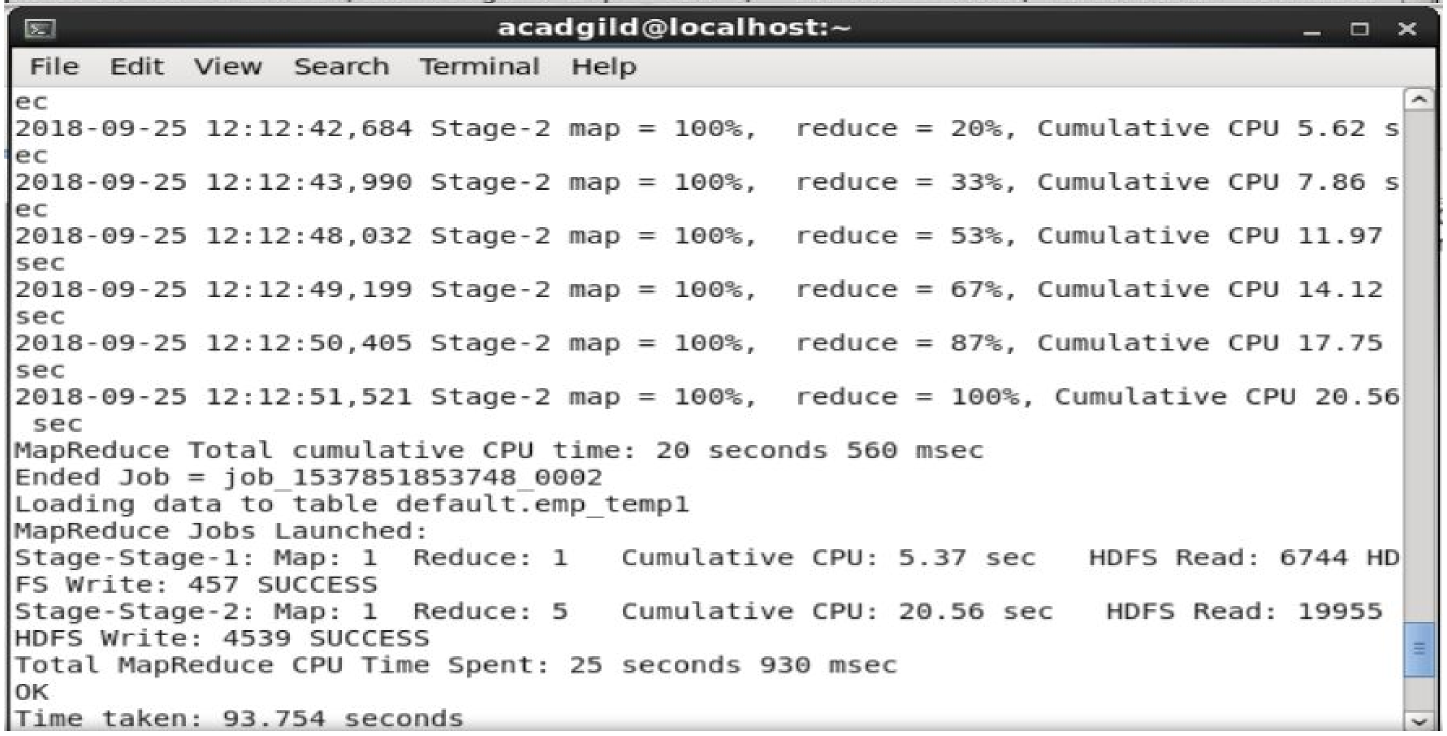


```
hive> CREATE TABLE emp_temp1(id INT, name STRING, location STRING) CLUSTERED BY
(ID) INTO 5 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINAT
ED BY '\n' STORED AS ORC TBLPROPERTIES('transactional'='true');
OK
Time taken: 0.23 seconds
```

Now load data into table '**emp_temp1**' from table '**emp_temp**': as follows:
hive> FROM emp_temp INSERT INTO emp_temp1 SELECT id,name,location ORDER BY 1;

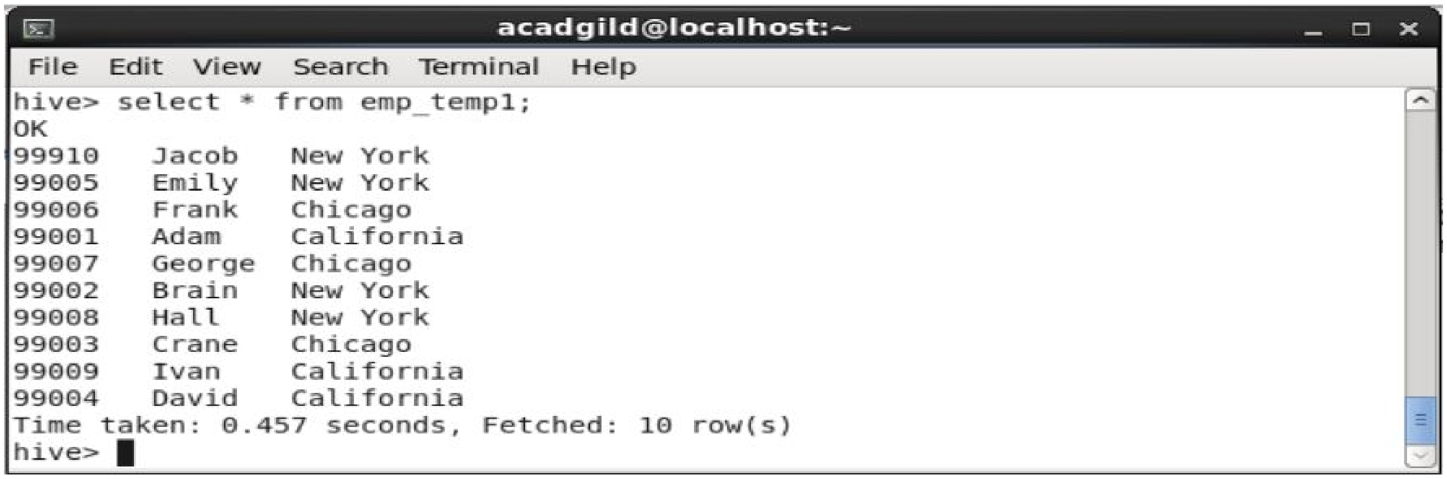


```
hive> FROM emp_temp INSERT INTO emp_temp1 SELECT id, name, location  ORDER BY 1;

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_20180925121119_d63a3347-2064-4d1d-a172-d4b8a08c7cb6
Total jobs = 2
Launching Job 1 out of 2
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_1537851853748_0001, Tracking URL = http://localhost:8088/prox
y/application_1537851853748_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1537851853748_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-25 12:11:37,256 Stage-1 map = 0%,  reduce = 0%
2018-09-25 12:11:45,353 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.47 se
c
2018-09-25 12:11:54,530 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.37
```



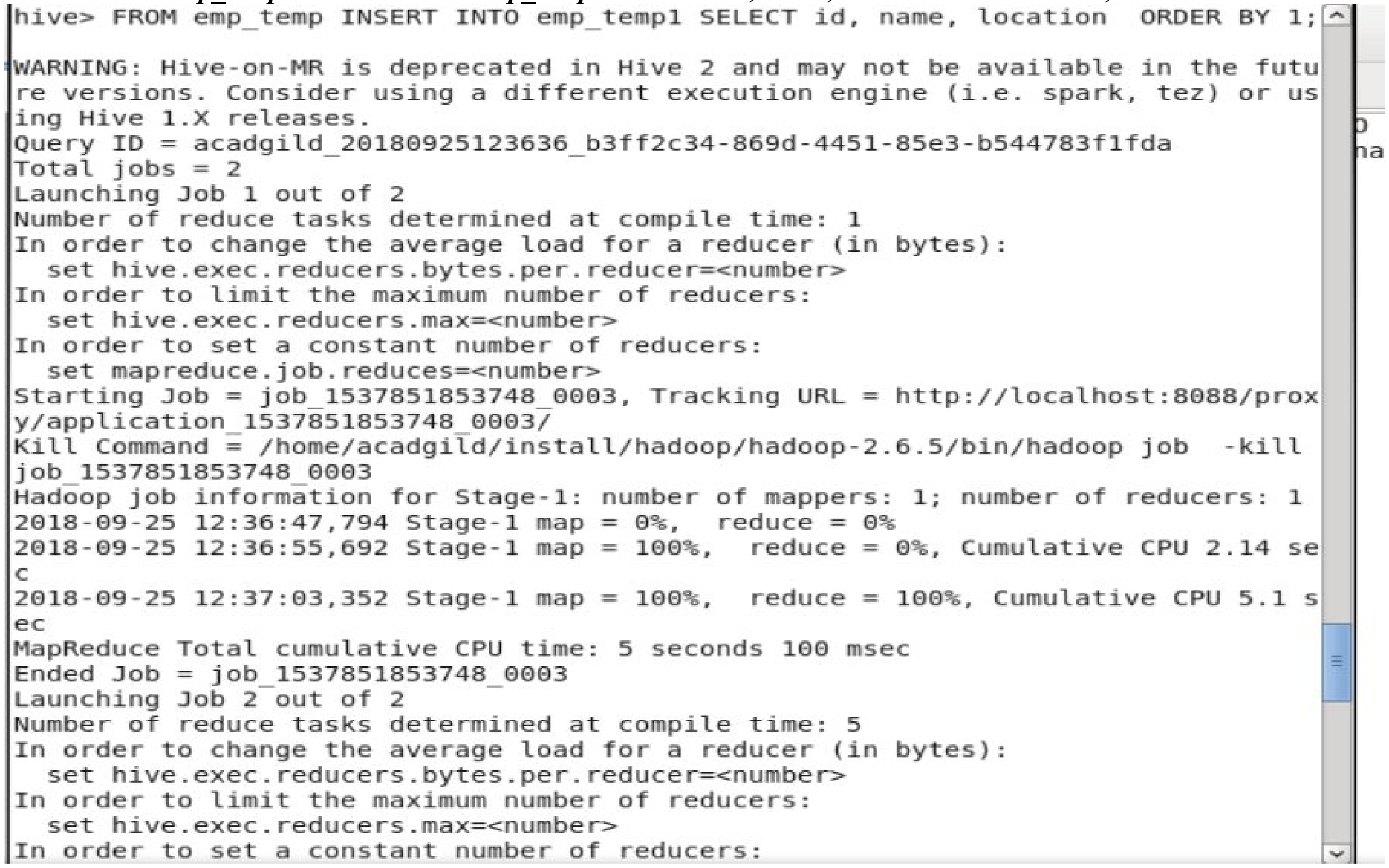
```
ec
2018-09-25 12:12:42,684 Stage-2 map = 100%,  reduce = 20%, Cumulative CPU 5.62 s
ec
2018-09-25 12:12:43,990 Stage-2 map = 100%,  reduce = 33%, Cumulative CPU 7.86 s
ec
2018-09-25 12:12:48,032 Stage-2 map = 100%,  reduce = 53%, Cumulative CPU 11.97
sec
2018-09-25 12:12:49,199 Stage-2 map = 100%,  reduce = 67%, Cumulative CPU 14.12
sec
2018-09-25 12:12:50,405 Stage-2 map = 100%,  reduce = 87%, Cumulative CPU 17.75
sec
2018-09-25 12:12:51,521 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 20.56
 sec
MapReduce Total cumulative CPU time: 20 seconds 560 msec
Ended Job = job_1537851853748_0002
Loading data to table default.emp_temp1
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.37 sec   HDFS Read: 6744 HD
FS Write: 457 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 5   Cumulative CPU: 20.56 sec   HDFS Read: 19955
HDFS Write: 4539 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 930 msec
OK
Time taken: 93.754 seconds
```

Now print the elements from the table as follows:
hive>select * from emp_temp1;

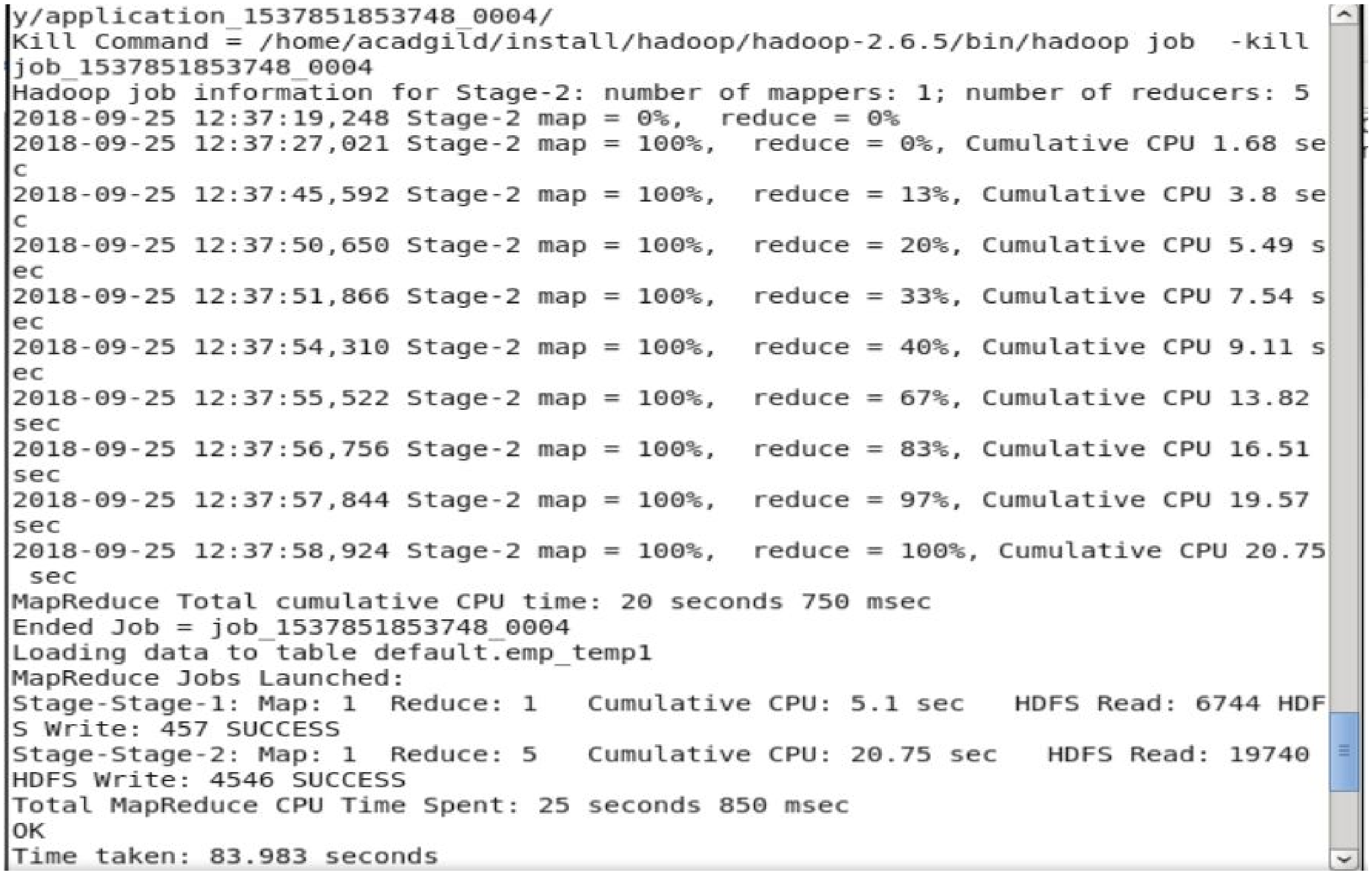


```
hive> select * from emp_temp1;
OK
99910   Jacob   New York
99005   Emily   New York
99006   Frank   Chicago
99001   Adam    California
99007   George  Chicago
99002   Brain   New York
99008   Hall    New York
99003   Crane   Chicago
99009   Ivan    California
99004   David   California
Time taken: 0.457 seconds, Fetched: 10 row(s)
hive> 
```

Now if we try to insert the same data again, it will be append to the previous data as shown below:
hive> FROM emp_temp INSERT INTO emp_temp1 SELECT id, name, location ORDER BY 1;

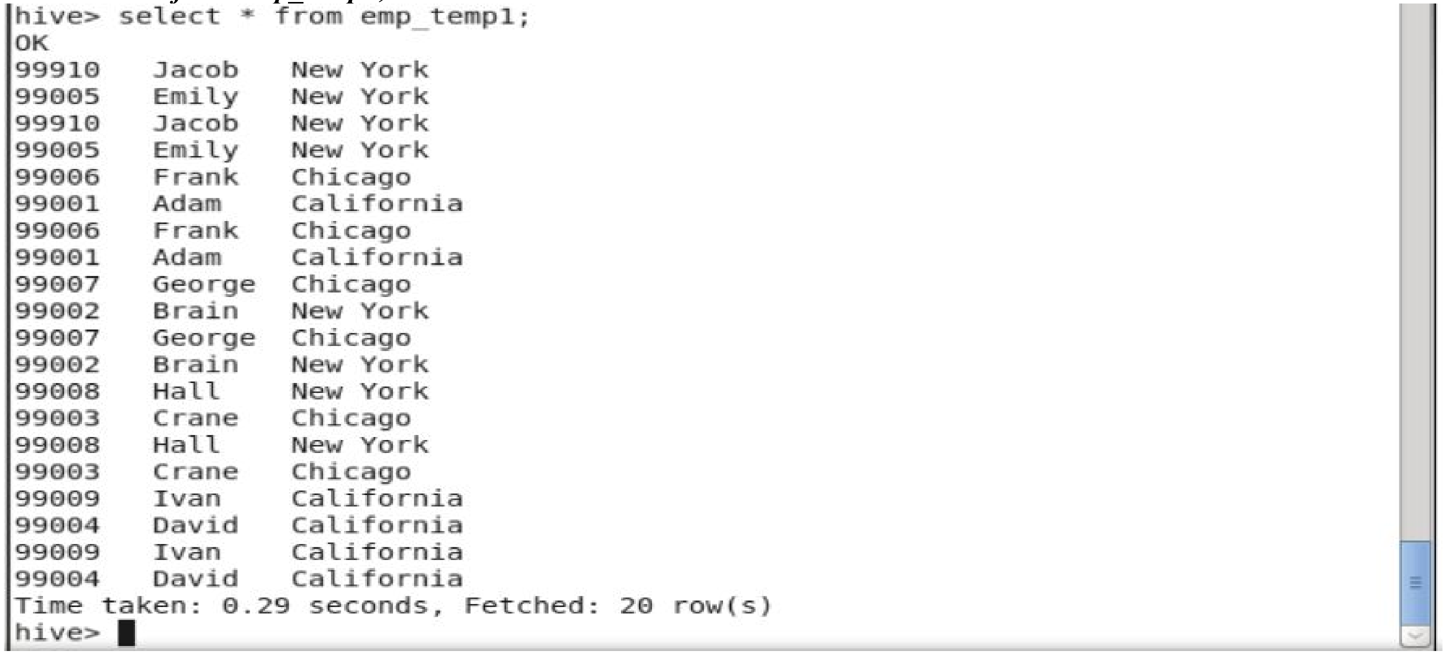


```
hive> FROM emp_temp INSERT INTO emp_temp1 SELECT id, name, location  ORDER BY 1;

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_20180925123636_b3ff2c34-869d-4451-85e3-b544783f1fda
Total jobs = 2
Launching Job 1 out of 2
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_1537851853748_0003, Tracking URL = http://localhost:8088/prox
y/application_1537851853748_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1537851853748_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-09-25 12:36:47,794 Stage-1 map = 0%,  reduce = 0%
2018-09-25 12:36:55,692 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.14 se
c
2018-09-25 12:37:03,352 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.1 s
ec
MapReduce Total cumulative CPU time: 5 seconds 100 msec
Ended Job = job_1537851853748_0003
Launching Job 2 out of 2
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:
```



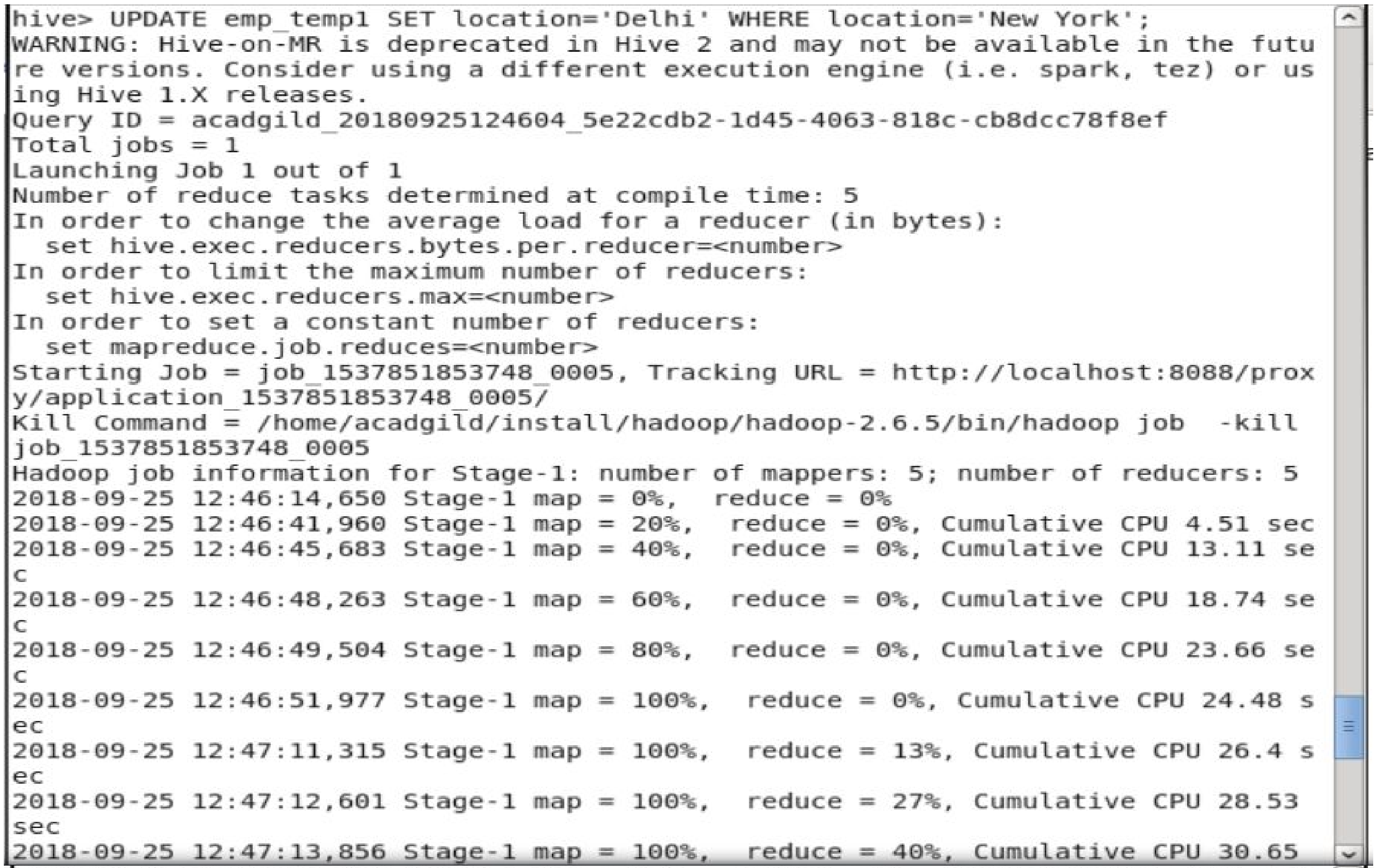
```
y/application_1537851853748_0004/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1537851853748_0004
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 5
2018-09-25 12:37:19,248 Stage-2 map = 0%,  reduce = 0%
2018-09-25 12:37:27,021 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.68 se
c
2018-09-25 12:37:45,592 Stage-2 map = 100%,  reduce = 13%, Cumulative CPU 3.8 se
c
2018-09-25 12:37:50,650 Stage-2 map = 100%,  reduce = 20%, Cumulative CPU 5.49 s
ec
2018-09-25 12:37:51,866 Stage-2 map = 100%,  reduce = 33%, Cumulative CPU 7.54 s
ec
2018-09-25 12:37:54,310 Stage-2 map = 100%,  reduce = 40%, Cumulative CPU 9.11 s
ec
2018-09-25 12:37:55,522 Stage-2 map = 100%,  reduce = 67%, Cumulative CPU 13.82
sec
2018-09-25 12:37:56,756 Stage-2 map = 100%,  reduce = 83%, Cumulative CPU 16.51
sec
2018-09-25 12:37:57,844 Stage-2 map = 100%,  reduce = 97%, Cumulative CPU 19.57
sec
2018-09-25 12:37:58,924 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 20.75
 sec
MapReduce Total cumulative CPU time: 20 seconds 750 msec
Ended Job = job_1537851853748_0004
Loading data to table default.emp_temp1
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.1 sec   HDFS Read: 6744 HDF
S Write: 457 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 5   Cumulative CPU: 20.75 sec   HDFS Read: 19740
HDFS Write: 4546 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 850 msec
OK
Time taken: 83.983 seconds
```

Now use print the data from table as follows:
hive> select * from emp_temp1;

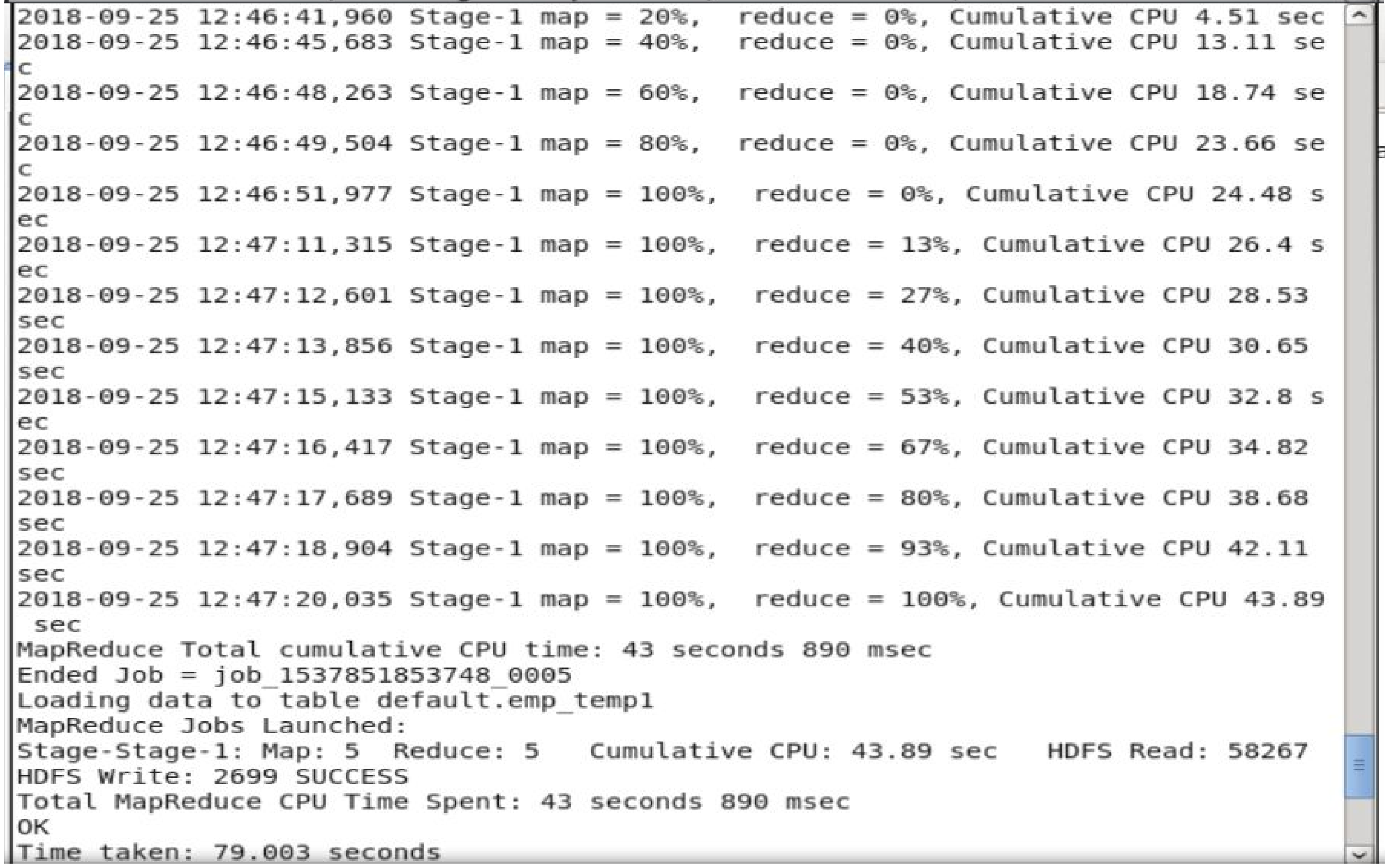


```
hive> select * from emp_temp1;
OK
99910   Jacob   New York
99005   Emily   New York
99910   Jacob   New York
99005   Emily   New York
99006   Frank   Chicago
99001   Adam    California
99006   Frank   Chicago
99001   Adam    California
99007   George  Chicago
99002   Brain   New York
99007   George  Chicago
99002   Brain   New York
99008   Hall    New York
99003   Crane   Chicago
99008   Hall    New York
99003   Crane   Chicago
99009   Ivan    California
99004   David   California
99009   Ivan    California
99004   David   California
Time taken: 0.29 seconds, Fetched: 20 row(s)
hive> 
```

Updating the Data in Hive Table
Update the table record as follows:
hive> UPDATE emp_temp1 SET location='Delhi' WHERE location='New York';



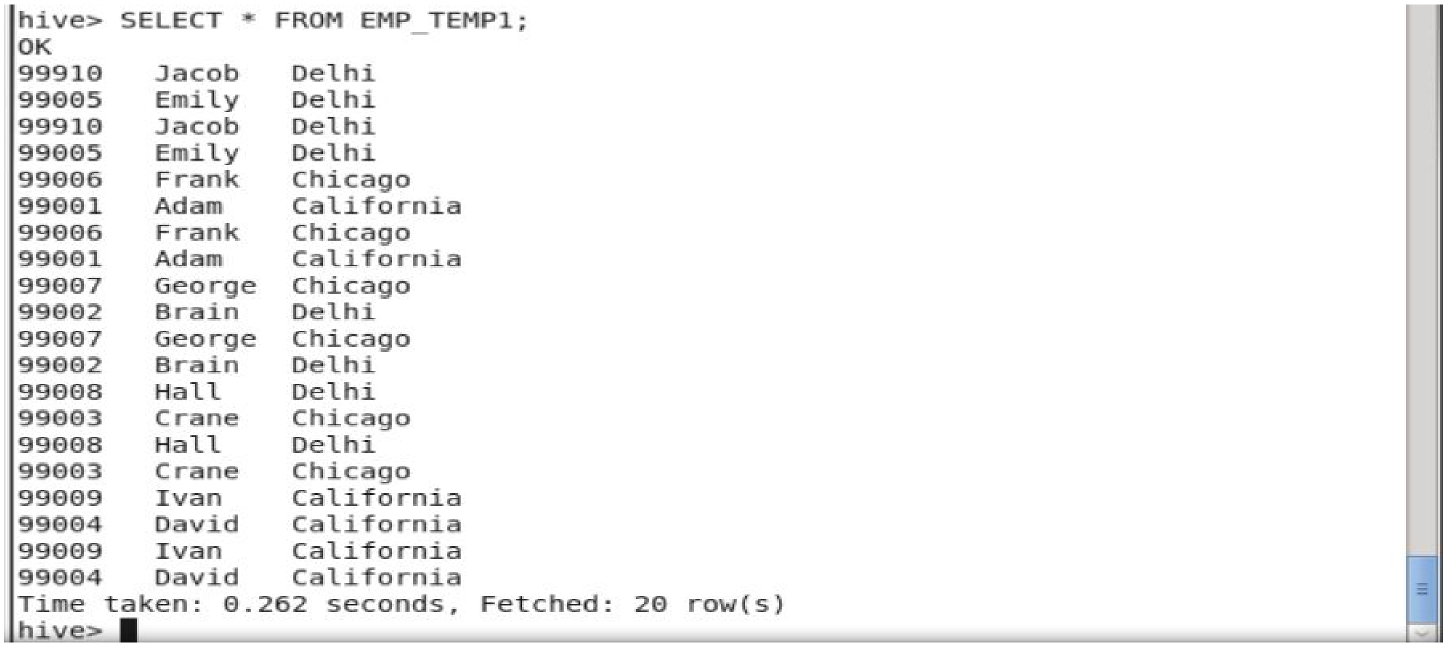
```
hive> UPDATE emp_temp1 SET location='Delhi' WHERE location='New York';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futu
re versions. Consider using a different execution engine (i.e. spark, tez) or us
ing Hive 1.X releases.
Query ID = acadgild_20180925124604_5e22cdb2-1d45-4063-818c-cb8dcc78f8ef
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_1537851853748_0005, Tracking URL = http://localhost:8088/prox
y/application_1537851853748_0005/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1537851853748_0005
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-09-25 12:46:14,650 Stage-1 map = 0%,  reduce = 0%
2018-09-25 12:46:41,960 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 4.51 sec
2018-09-25 12:46:45,683 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 13.11 se
c
2018-09-25 12:46:48,263 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 18.74 se
c
2018-09-25 12:46:49,504 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 23.66 se
c
2018-09-25 12:46:51,977 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 24.48 s
ec
2018-09-25 12:47:11,315 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 26.4 s
ec
2018-09-25 12:47:12,601 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 28.53
sec
2018-09-25 12:47:13,856 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 30.65
```



```
2018-09-25 12:46:41,960 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 4.51 sec
2018-09-25 12:46:45,683 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 13.11 se
c
2018-09-25 12:46:48,263 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 18.74 se
c
2018-09-25 12:46:49,504 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 23.66 se
c
2018-09-25 12:46:51,977 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 24.48 s
ec
2018-09-25 12:47:11,315 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 26.4 s
ec
2018-09-25 12:47:12,601 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 28.53
sec
2018-09-25 12:47:13,856 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 30.65
sec
```

```
2018-09-25 12:47:15,133 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 32.8 s
ec
2018-09-25 12:47:16,417 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 34.82
sec
2018-09-25 12:47:17,689 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 38.68
sec
2018-09-25 12:47:18,904 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 42.11
sec
2018-09-25 12:47:20,035 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 43.89
 sec
MapReduce Total cumulative CPU time: 43 seconds 890 msec
Ended Job = job_1537851853748_0005
Loading data to table default.emp_temp1
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 43.89 sec   HDFS Read: 58267
HDFS Write: 2699 SUCCESS
Total MapReduce CPU Time Spent: 43 seconds 890 msec
OK
Time taken: 79.003 seconds
```

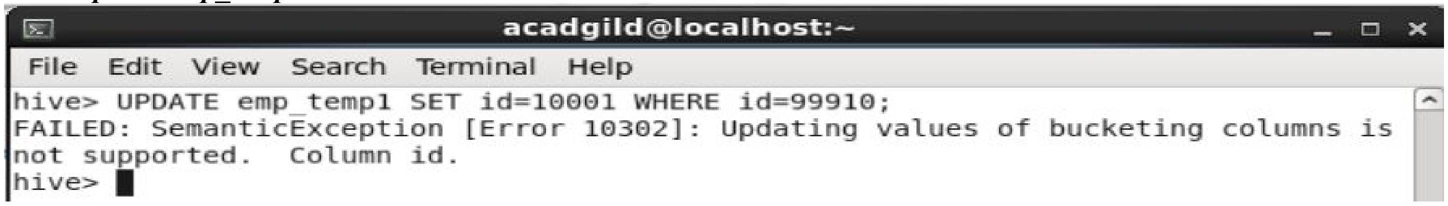
This will update the Records, with ***'New York'*** to **'Delhi'** we will seethe the output as follows:
hive>select * from emp_temp1;



```
hive> SELECT * FROM EMP_TEMP1;
OK
99910   Jacob   Delhi
99005   Emily   Delhi
99910   Jacob   Delhi
99005   Emily   Delhi
99006   Frank   Chicago
99001   Adam    California
99006   Frank   Chicago
99001   Adam    California
99007   George  Chicago
99002   Brain   Delhi
99007   George  Chicago
99002   Brain   Delhi
99008   Hall    Delhi
99003   Crane   Chicago
99008   Hall    Delhi
99003   Crane   Chicago
99009   Ivan    California
99004   David   California
99009   Ivan    California
99004   David   California
Time taken: 0.262 seconds, Fetched: 20 row(s)
hive>
```

We have successfully updated the data, and we can observe that the updated data.

Now we try to update the ID of the employee as follows:
hive> update emp_temp1 set id=10001 where id=99910.

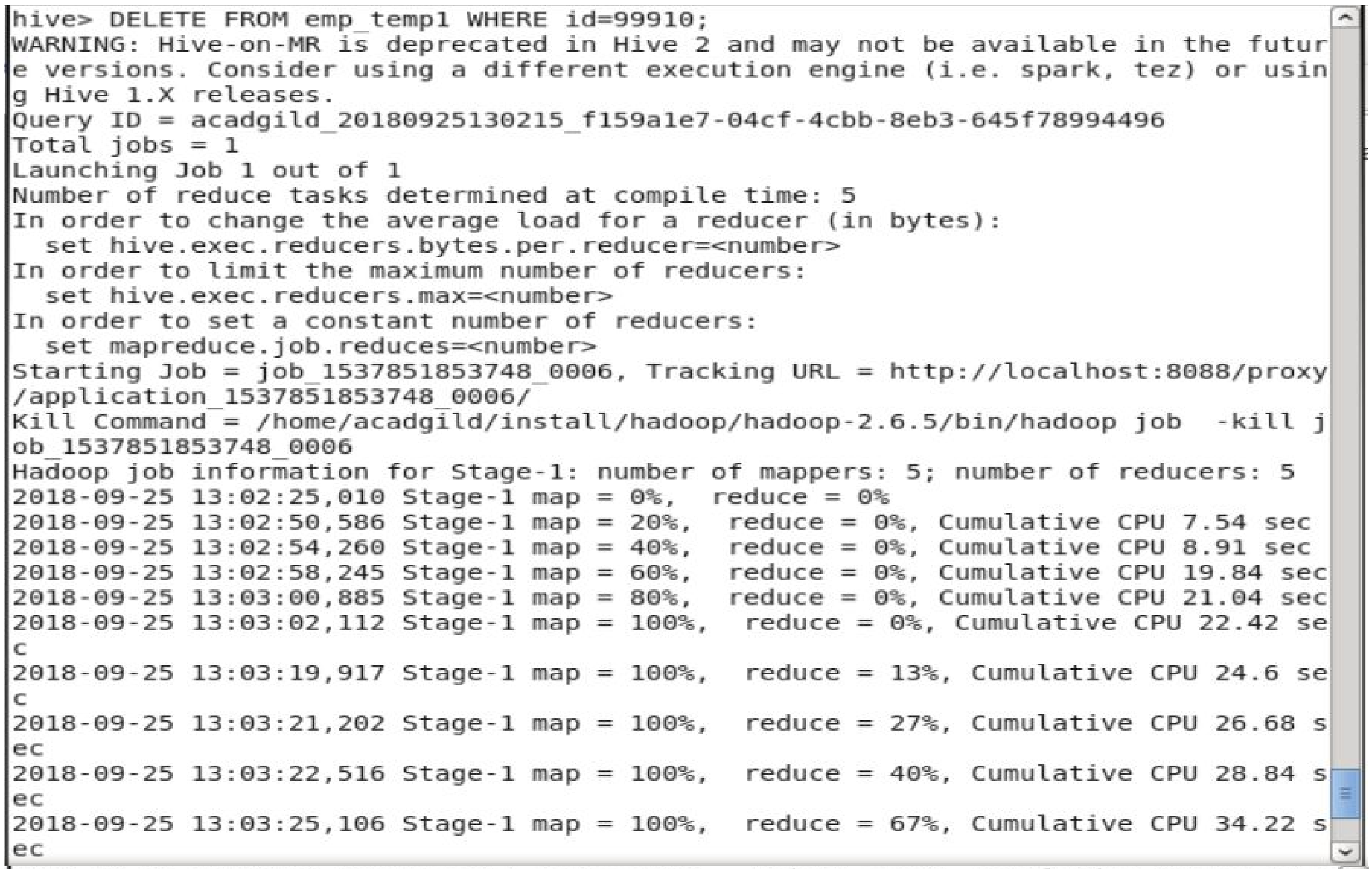


```
hive> UPDATE emp_temp1 SET id=10001 WHERE id=99910;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is
not supported.  Column id.
hive>
```

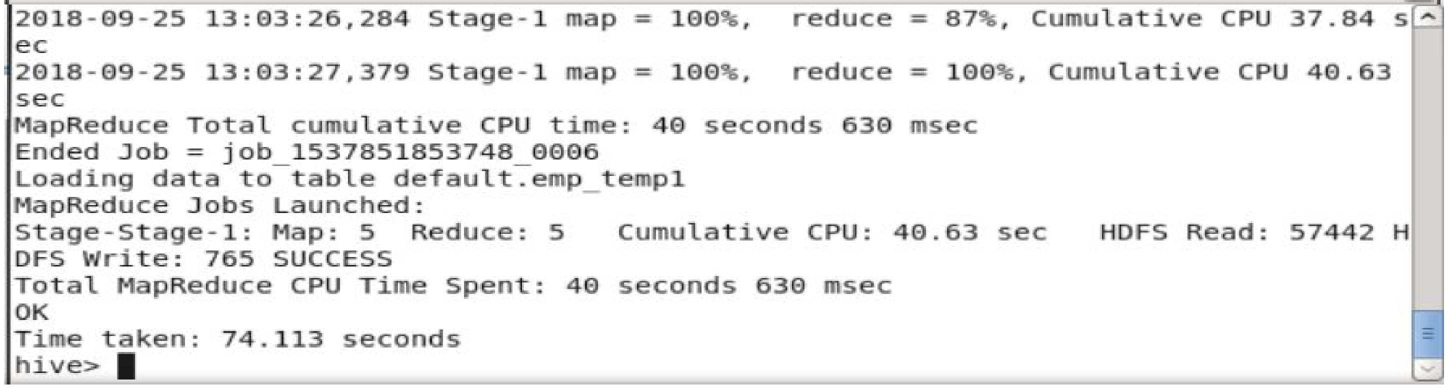
From the above image we can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed.

Deleting a Row from Hive Table

Now let's perform the Delete operation on the same table as follows:
hive>DELETE FROM emp_temp1 WHERE id=99910;

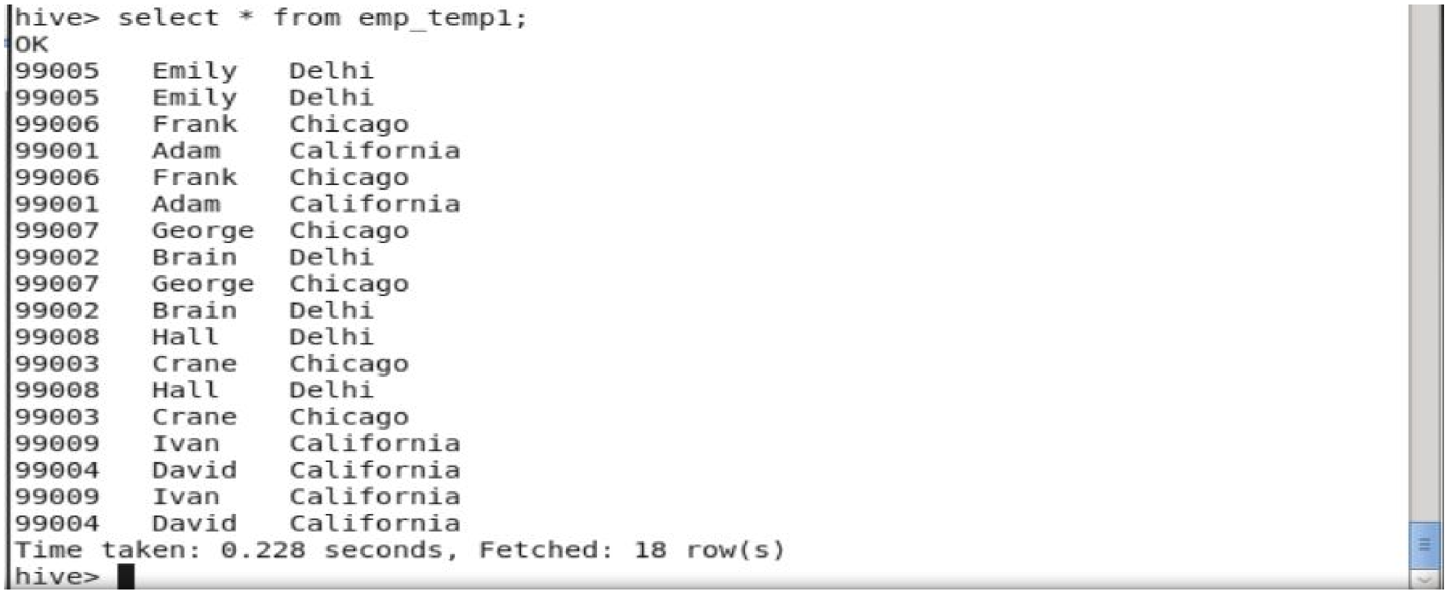


```
hive> DELETE FROM emp_temp1 WHERE id=99910;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the futur
e versions. Consider using a different execution engine (i.e. spark, tez) or usin
g Hive 1.X releases.
Query ID = acadgild_20180925130215_f159a1e7-04cf-4cbb-8eb3-645f78994496
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_1537851853748_0006, Tracking URL = http://localhost:8088/proxy
/application_1537851853748_0006/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill j
ob_1537851853748_0006
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2018-09-25 13:02:25,010 Stage-1 map = 0%,  reduce = 0%
2018-09-25 13:02:50,586 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 7.54 sec
2018-09-25 13:02:54,260 Stage-1 map = 40%,  reduce = 0%, Cumulative CPU 8.91 sec
2018-09-25 13:02:58,245 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 19.84 sec
2018-09-25 13:03:00,885 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 21.04 sec
2018-09-25 13:03:02,112 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 22.42 se
c
2018-09-25 13:03:19,917 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 24.6 se
c
2018-09-25 13:03:21,202 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 26.68 s
ec
2018-09-25 13:03:22,516 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 28.84 s
ec
2018-09-25 13:03:25,106 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 34.22 s
ec
```



```
2018-09-25 13:03:26,284 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 37.84 s
ec
2018-09-25 13:03:27,379 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 40.63
sec
MapReduce Total cumulative CPU time: 40 seconds 630 msec
Ended Job = job_1537851853748_0006
Loading data to table default.emp_temp1
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 40.63 sec   HDFS Read: 57442 H
DFS Write: 765 SUCCESS
Total MapReduce CPU Time Spent: 40 seconds 630 msec
OK
Time taken: 74.113 seconds
hive> 
```

Now we can see that the record was deleted. We can observe this by printing the table records as follows:
hive>select * from emp_temp1;



```
hive> select * from emp_temp1;
OK
99005   Emily   Delhi
99005   Emily   Delhi
99006   Frank   Chicago
99001   Adam    California
99006   Frank   Chicago
99001   Adam    California
99007   George  Chicago
99002   Brain   Delhi
99007   George  Chicago
99002   Brain   Delhi
99008   Hall    Delhi
99003   Crane   Chicago
99008   Hall    Delhi
99003   Crane   Chicago
99009   Ivan    California
99004   David   California
99009   Ivan    California
99004   David   California
Time taken: 0.228 seconds, Fetched: 18 row(s)
hive> 
```

We can see that there is no row with ***id=99910.*** 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.