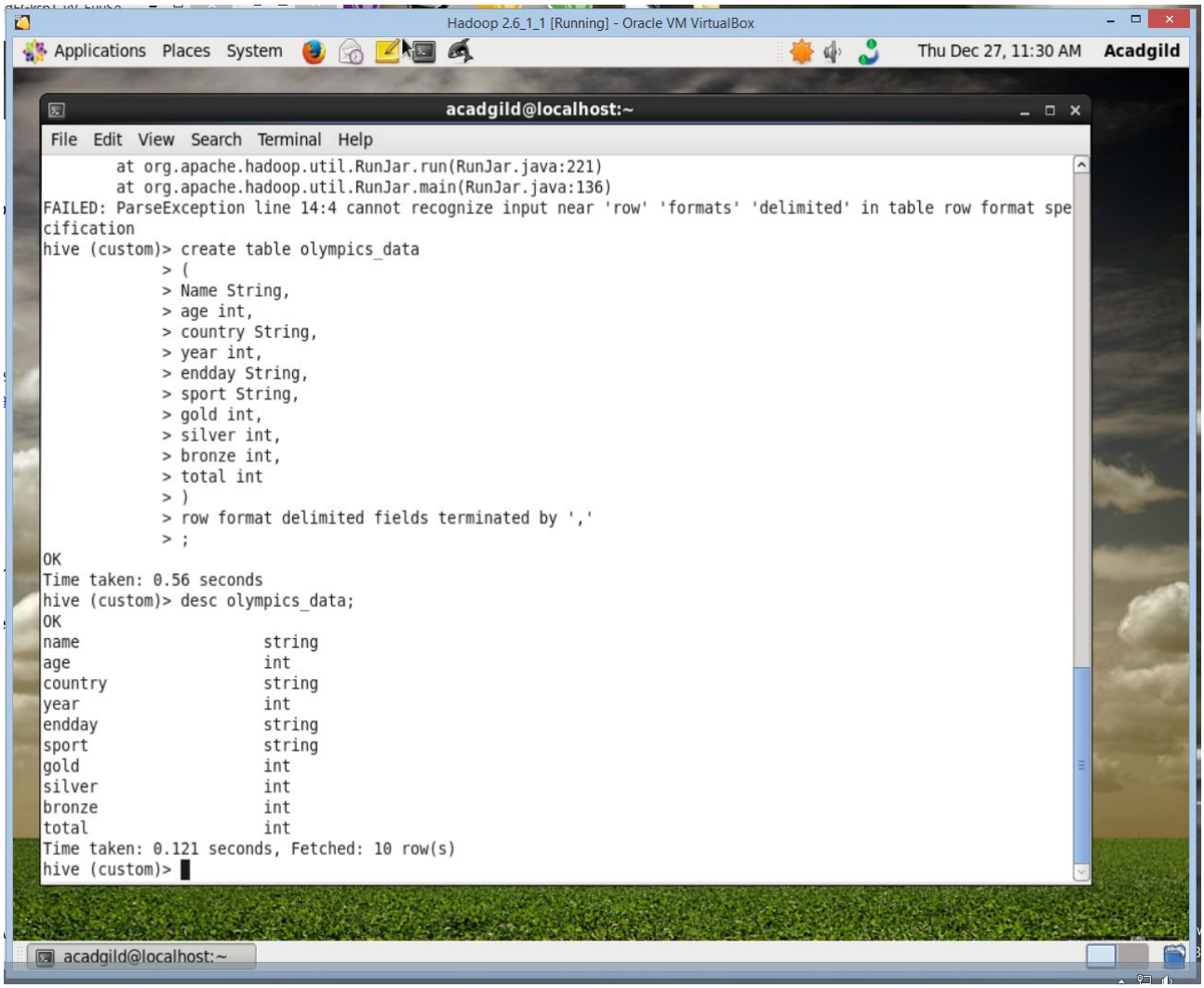
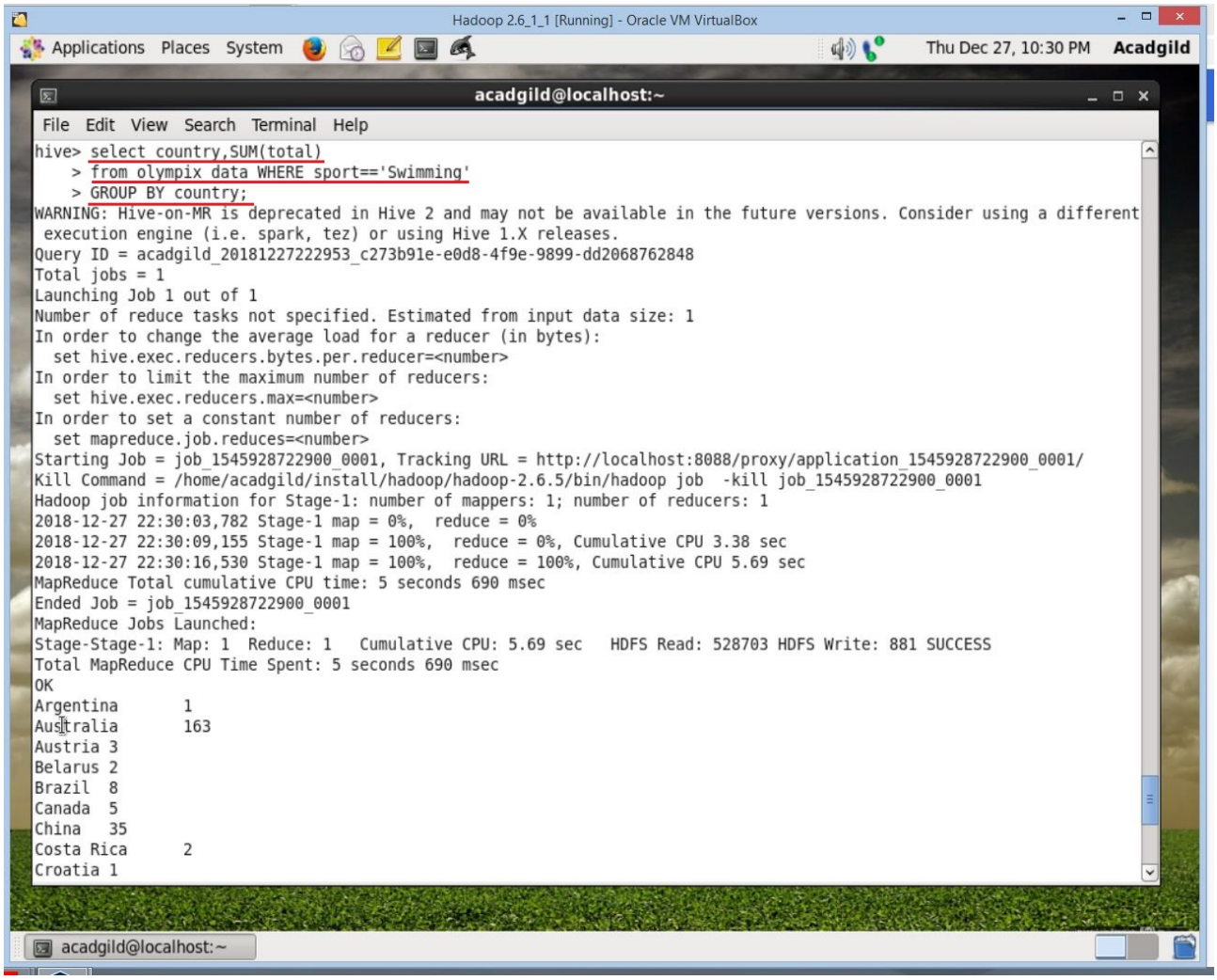
Assignment 1

Create the table for the dataset given

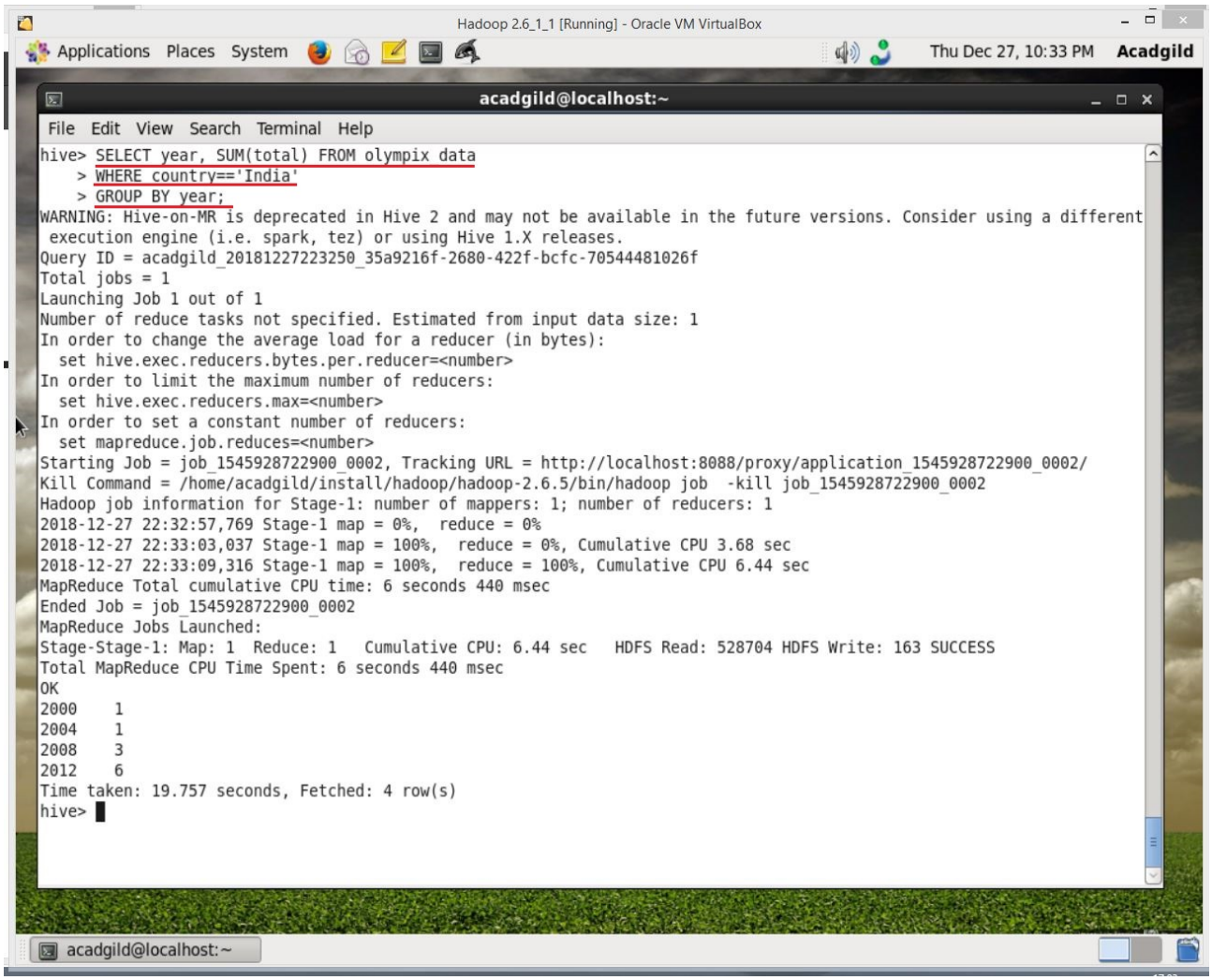


TASK 1

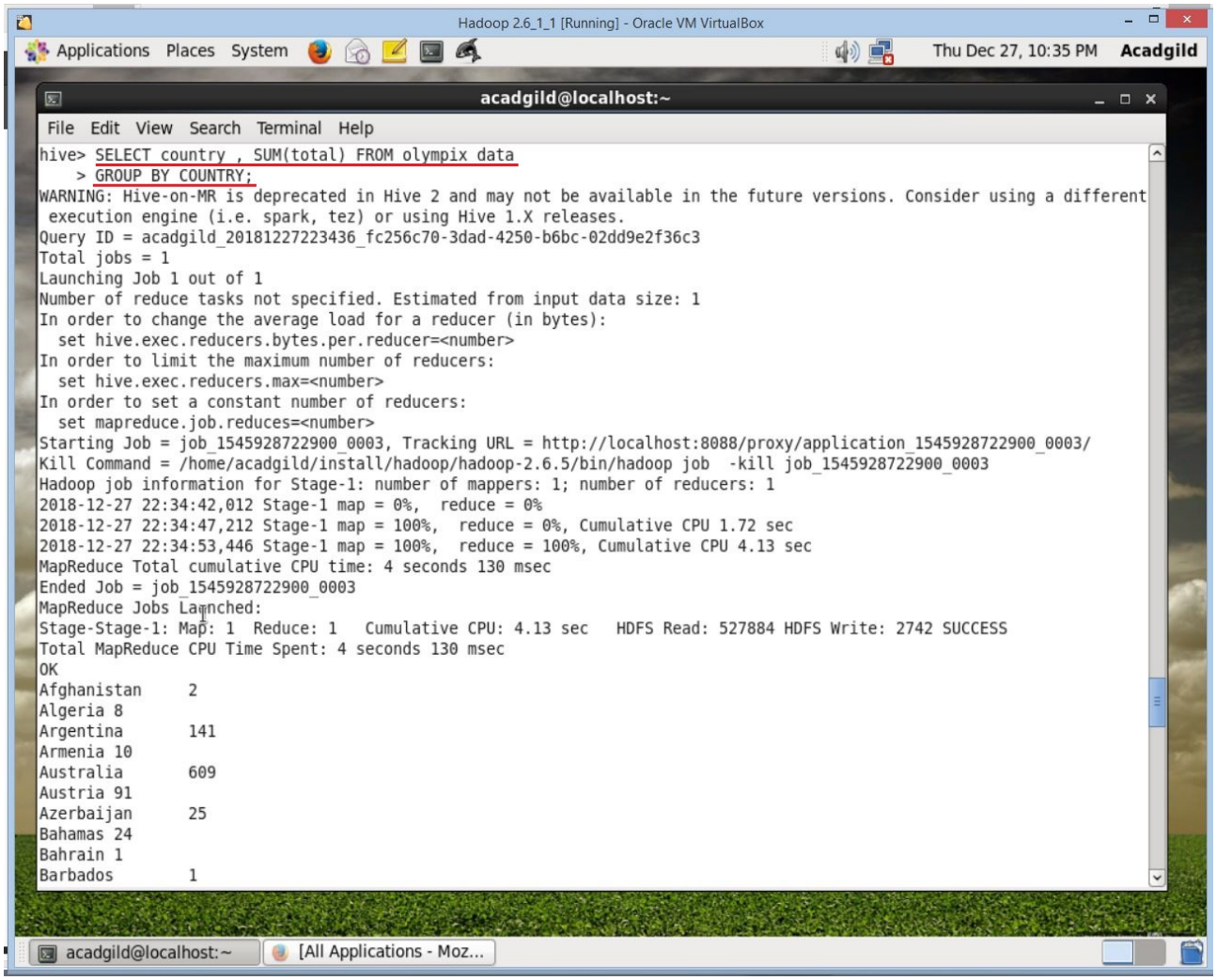
1) Hive program to find the number of medals won by each country in swimming



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

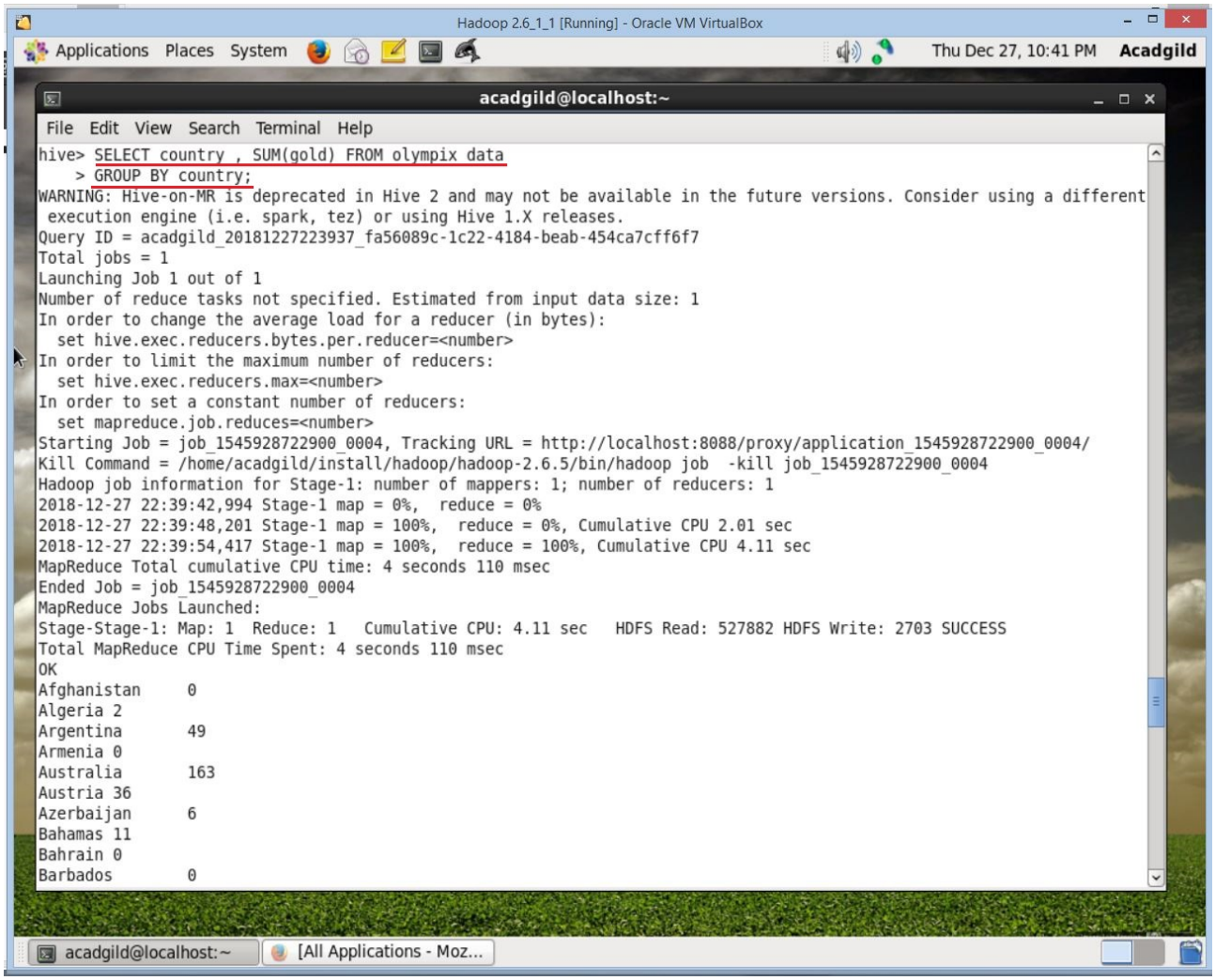


3) Hive Program to find the total number of medals each country won.



```
hive> SELECT country , SUM(total) FROM olympix data
    > 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. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20181227223436_fc256c70-3dad-4250-b6bc-02dd9e2f36c3
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_1545928722900_0003, Tracking URL = http://localhost:8088/proxy/application_1545928722900_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1545928722900_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-12-27 22:34:42,012 Stage-1 map = 0%,  reduce = 0%
2018-12-27 22:34:47,212 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.72 sec
2018-12-27 22:34:53,446 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.13 sec
MapReduce Total cumulative CPU time: 4 seconds 130 msec
Ended Job = job_1545928722900_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.13 sec   HDFS Read: 527884 HDFS Write: 2742 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 130 msec
OK
Afghanistan     2
Algeria 8
Argentina       141
Armenia 10
Australia       609
Austria 91
Azerbaijan      25
Bahamas 24
Bahrain 1
Barbados        1
```

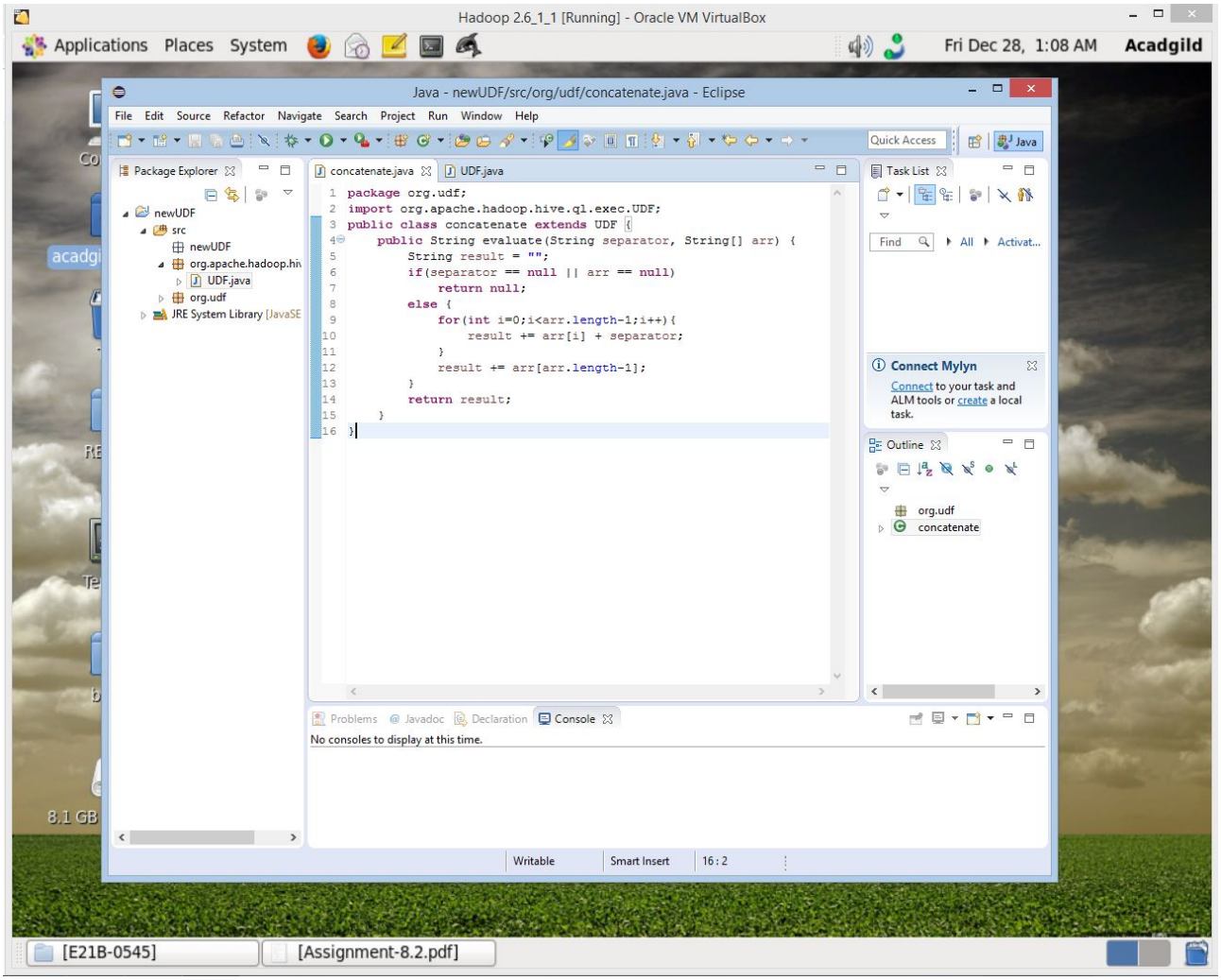
4) Hive program to find the number of gold medals each country won.



TASK 2

Problem Statement:
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.

Solution:



TASK 3

Problem Statement: 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.

Solution:

```
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 = 5;
hive>
```

```
hive> CREATE TABLE smartphones_data
    > (phone_id INT,
    > manufacturer STRING,
    > phone_model STRING,
    > phone_price INT)
    > CLUSTERED BY (phone_id) INTO 5 BUCKETS
    > STORED AS orc
    > TBLPROPERTIES('transactional'='true');
OK
Time taken: 0.649 seconds
hive> SHOW TABLES;
OK
employee
olympics_data
smartphones_data
temperature_data
temperature_data_new
Time taken: 0.1 seconds, Fetched: 5 row(s)
hive>
```

INSERT DATA INTO TABLE

INSERT INTO TABLE smartphones_data VALUES(101, 'Samsung', 'S6 Edge', 30000), (102, 'Apple', 'iPhone X', 89000), (103, 'Motorola', 'G5 Plus', 15000), (104, 'Coolpad', 'Note 3', 9000);

The contents of the table can be viewed using the command **select * from smartphones_data;**

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S6 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
```

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S6 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
101     Samsung S6 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
Time taken: 6.508 seconds, Fetched: 8 row(s)
hive>
```

```
hive> DELETE FROM smartphones_data
    > WHERE phone_id = 104;
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_20171217135153_816a9e06-8a8b-49fa-8981-95fa3c008bcd
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_1513491304174_0004, Tracking URL = http://localhost:8088/proxy/application_1513491304174_0004/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1513491304174_0004
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
```

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

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S7 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
101     Samsung S7 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
Time taken: 2.931 seconds, Fetched: 6 row(s)
hive>
```