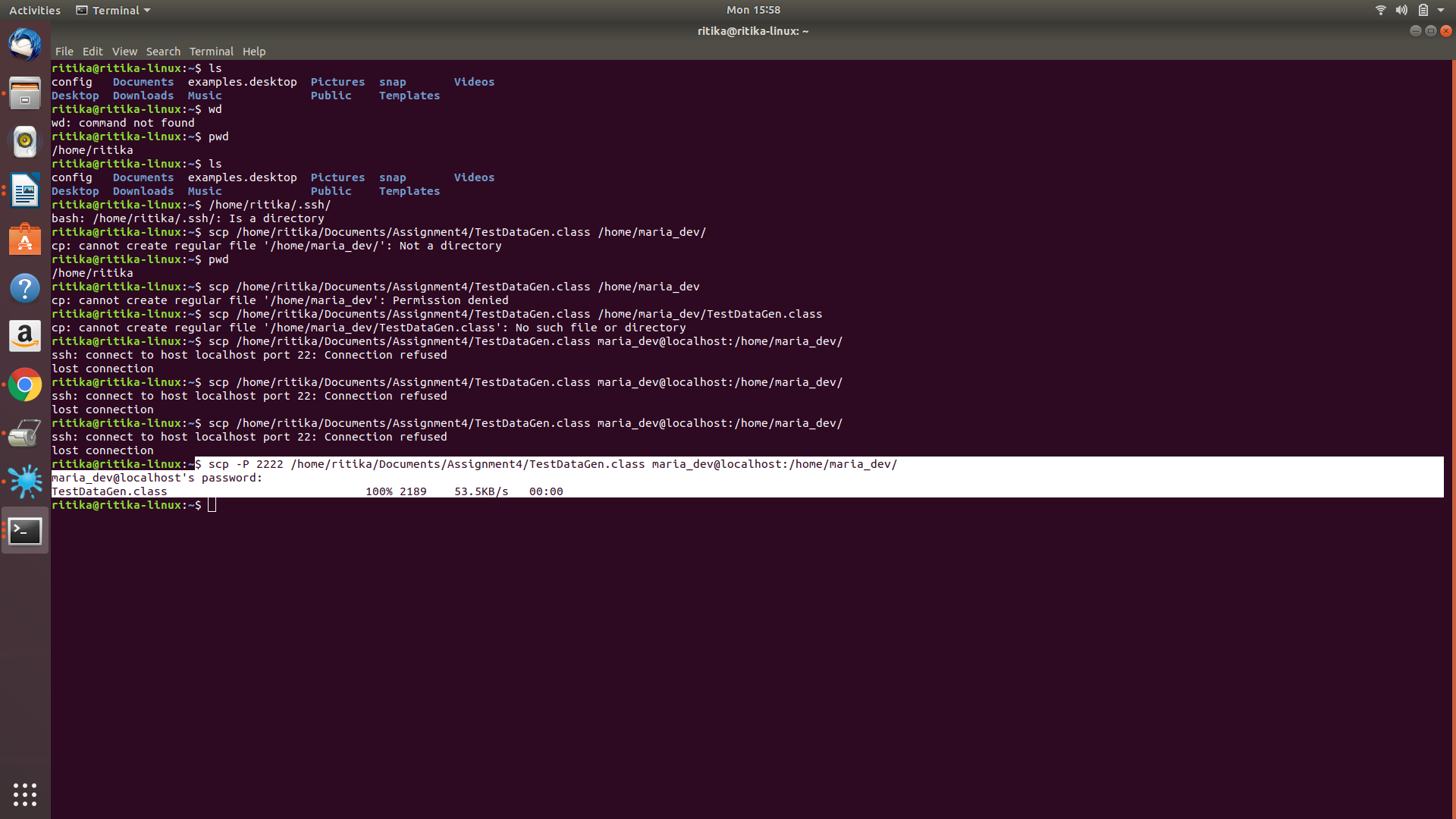
# Ritika Kumari- A20414073

# CSP554—Big Data Technologies

## Assignment #4

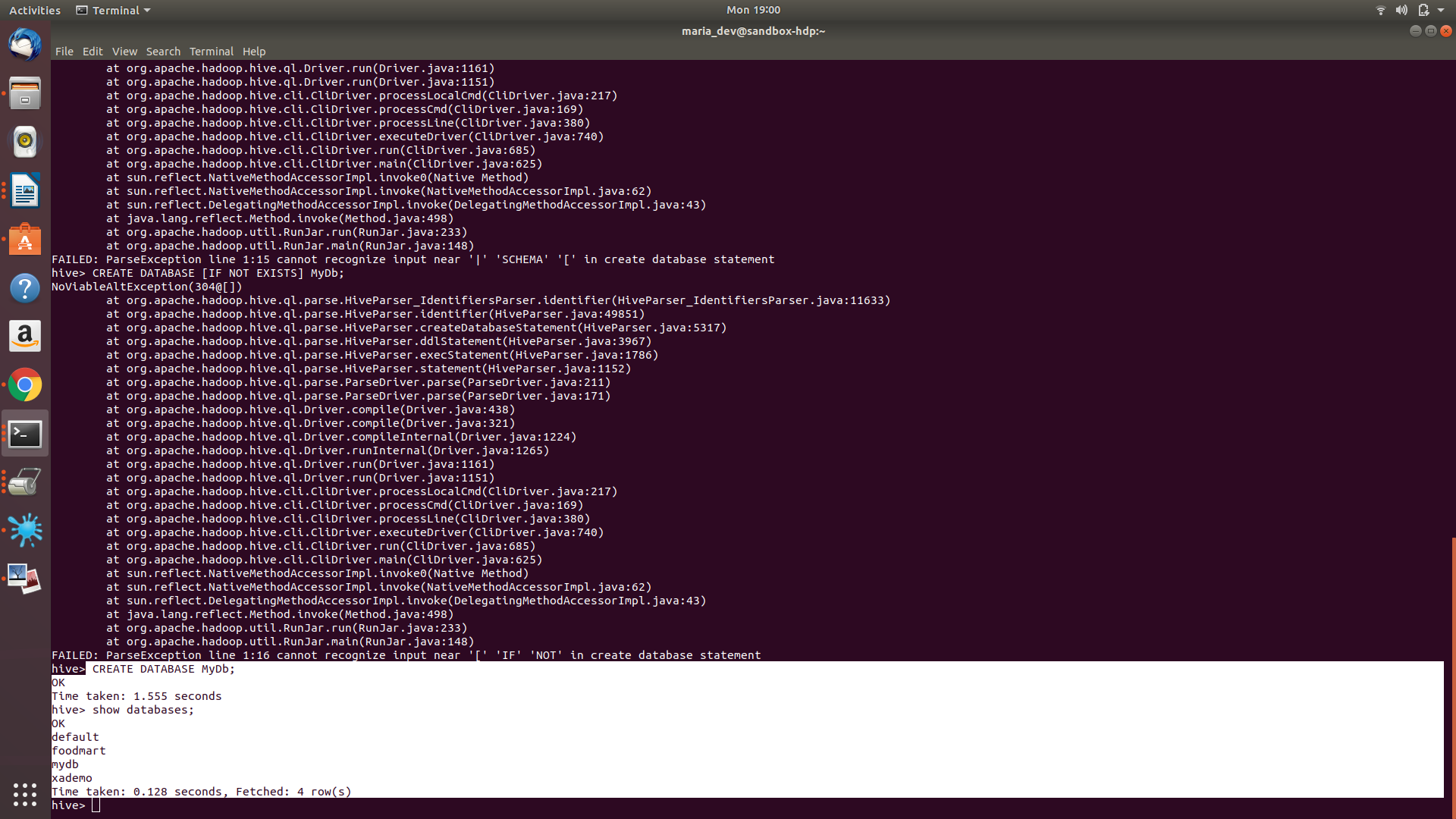
The foodratings<.magic number>.txt file has six comma separated fields. The first field is the name of a food critic. The second through fifth fields are the ratings each critic gives to four food types at each restaurant they review. The ratings are an integer from 1 through 50. The sixth field is the id of the restaurant.

The foodplaces<.magic number>.txt file has two comma separated fields. The first field is the id of a restaurant. The second field is the name of that restaurant.





**Magic Number: 53144**



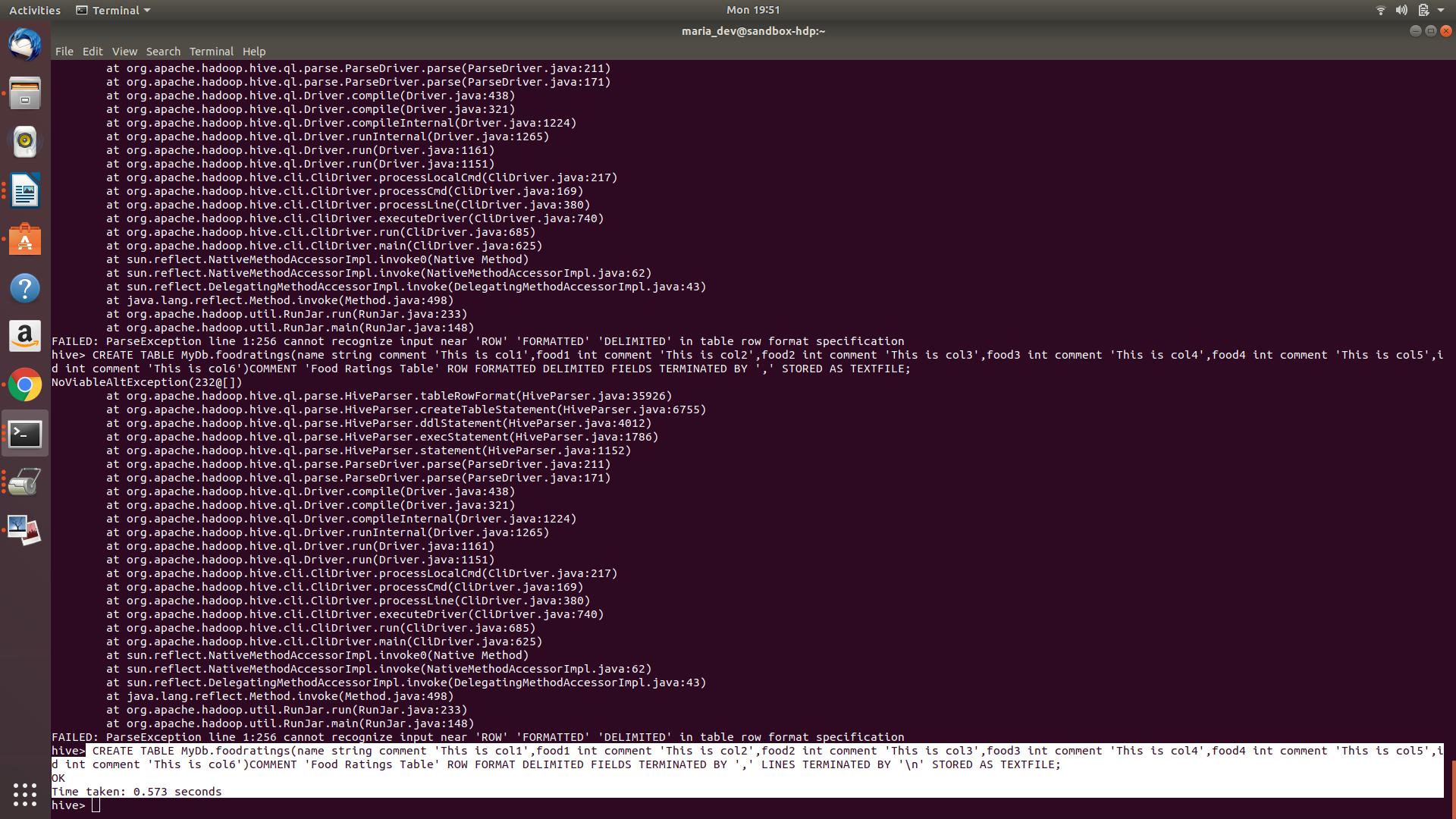
Exercise 1) 2 points

Create a Hive database called MyDb.

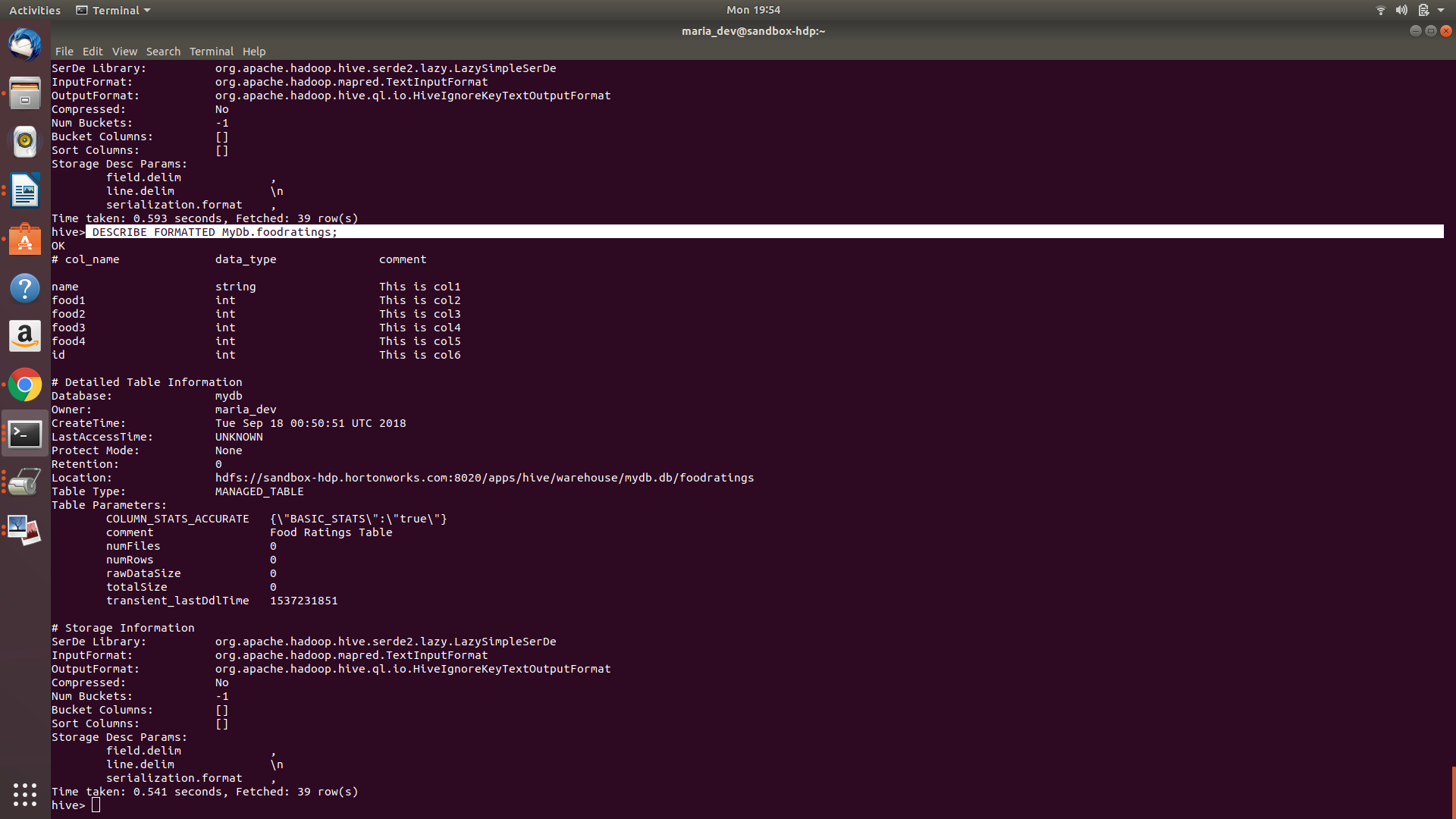
Then in MyDb create a table with name foodratings having six columns with the name of the first ‘name’ and the type of the first a string and the names of the remaining columns food1, food2, food3, food4 and id and indicate their types each as an integer. The table should have storage format TEXTFILE and column separator a “,”. That is the underlying format should be a CSV file. The table itself and each column should include a comment (it does not matter what it says).

Execute a one shot Hive command of ‘DESCRIBE FORMATTED MyDb.foodratings’ and capture its output as one of the results of this exercise.

**Command:** CREATE TABLE MyDb.foodratings(name string comment 'This is col1',food1 int comment 'This is col2',food2 int comment 'This is col3',food3 int comment 'This is col4',food4 int comment 'This is col5',id int comment 'This is col6')COMMENT 'Food Ratings Table' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;



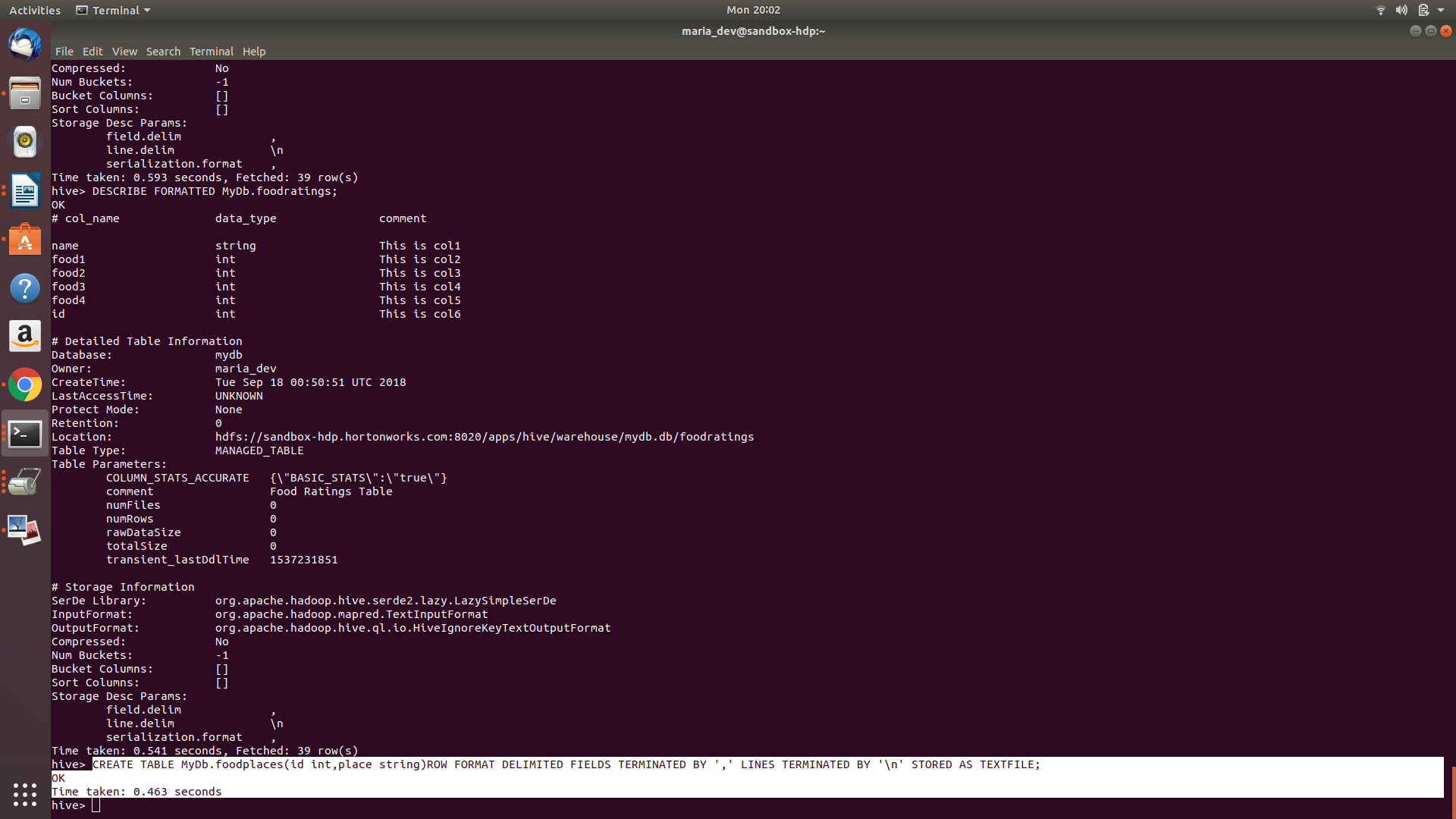
**Command:**DESCRIBE FORMATTED MyDb.foodratings;



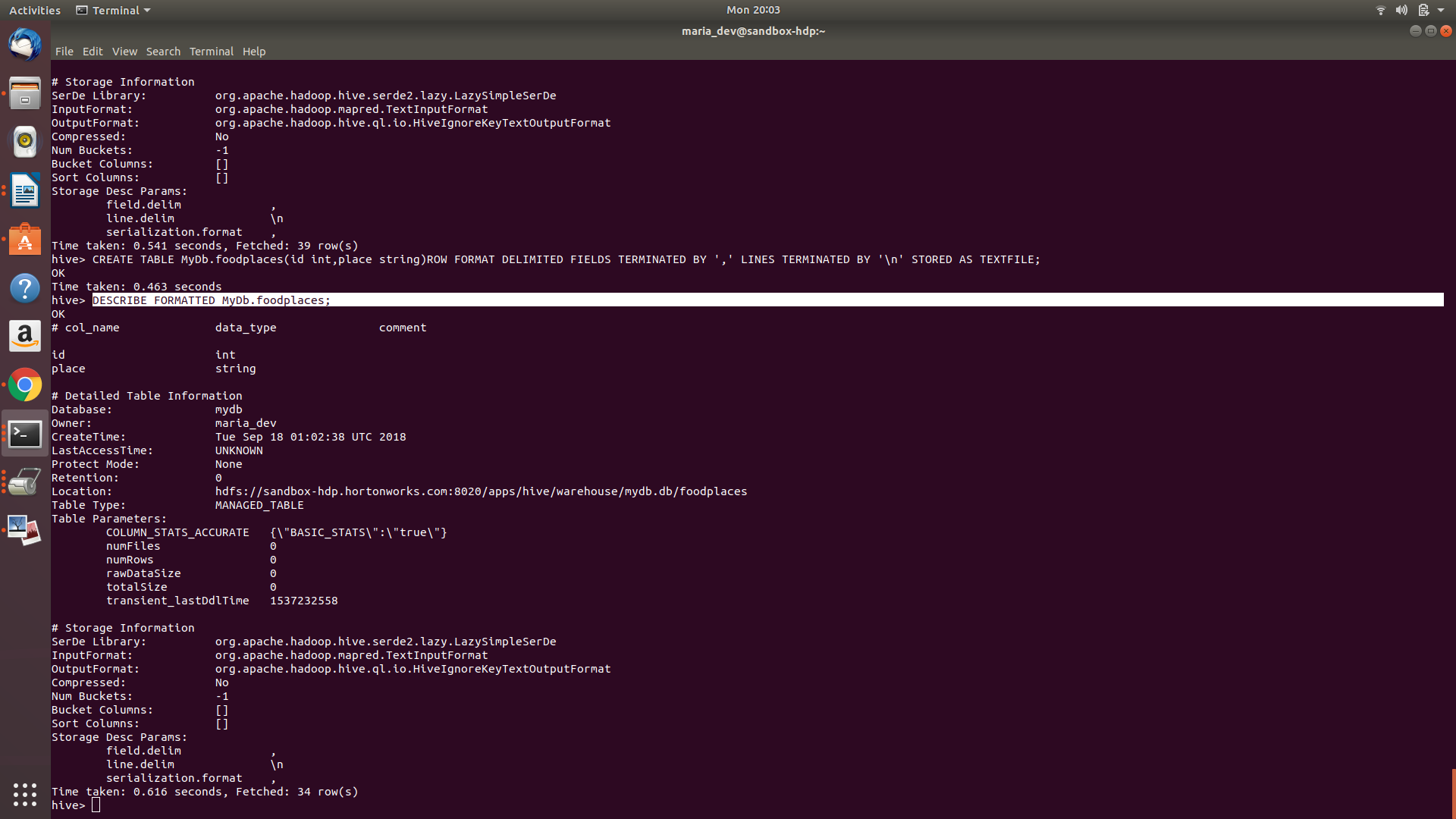
Then in MyDb create a table with name foodplaces having two columns with first called ‘id’ and the type of the first an integer and the second called ‘place’ and the type of the second a string. This table should also have storage format TEXTFILE and column separator a “,”. That is the underlying format should be a CSV file. No comments are needed for this table

Execute a one shot Hive command of ‘DESCRIBE FORMATTED MyDb.foodplaces’ and capture its output as another of the results of this exercise.

**Command:** CREATE TABLE MyDb.foodplaces(id int,place string)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' STORED AS TEXTFILE;



**command:**DESCRIBE FORMATTED MyDb.foodplaces;



Exercise 2)

Load the foodratings<.magic number>.txt file created using TestDataGen from your local file system into the foodratings table.

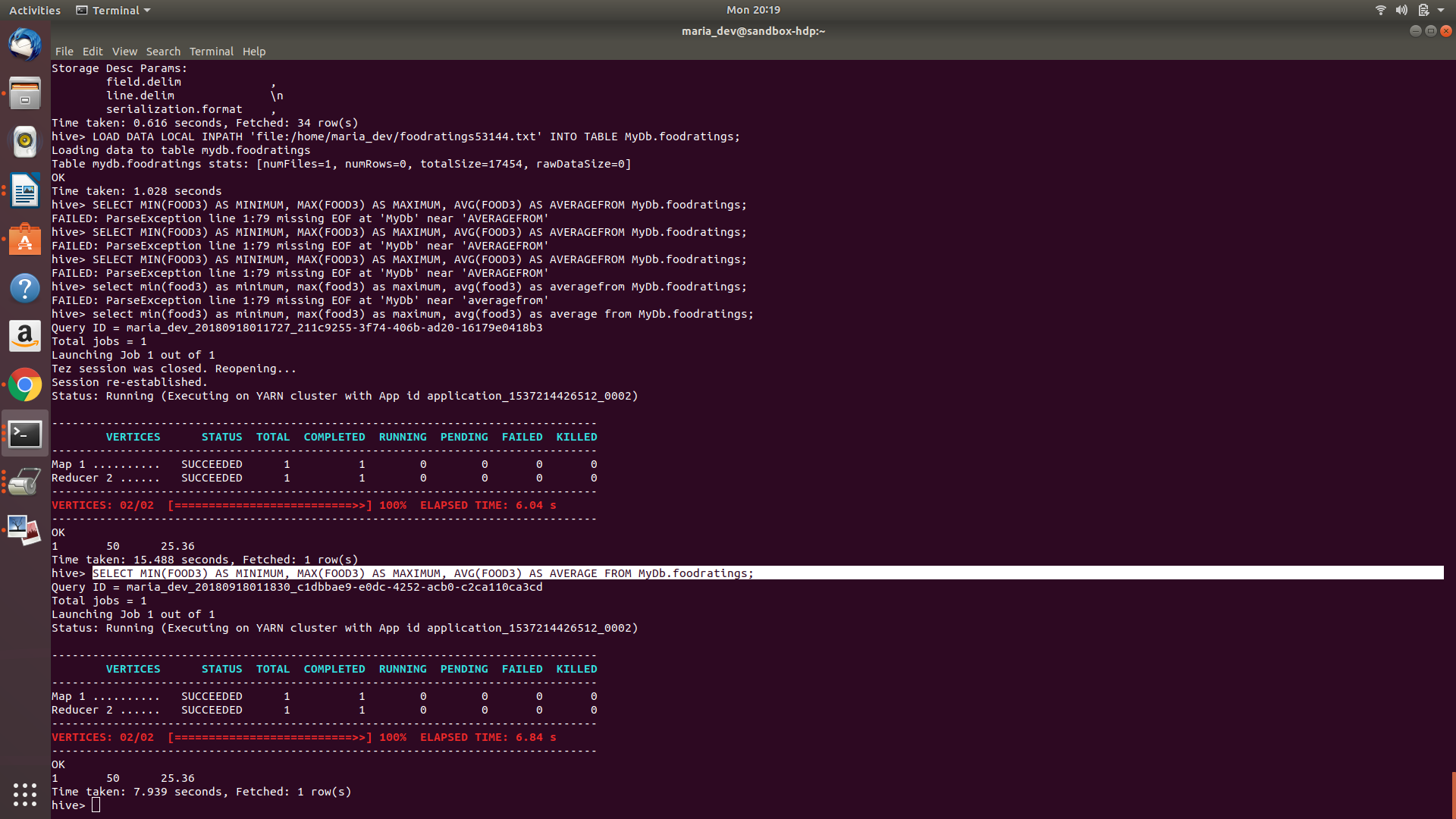
**Command:** LOAD DATA LOCAL INPATH 'file:/home/maria\_dev/foodratings53144.txt' INTO TABLE MyDb.foodratings;



**Magic Number: 53144**

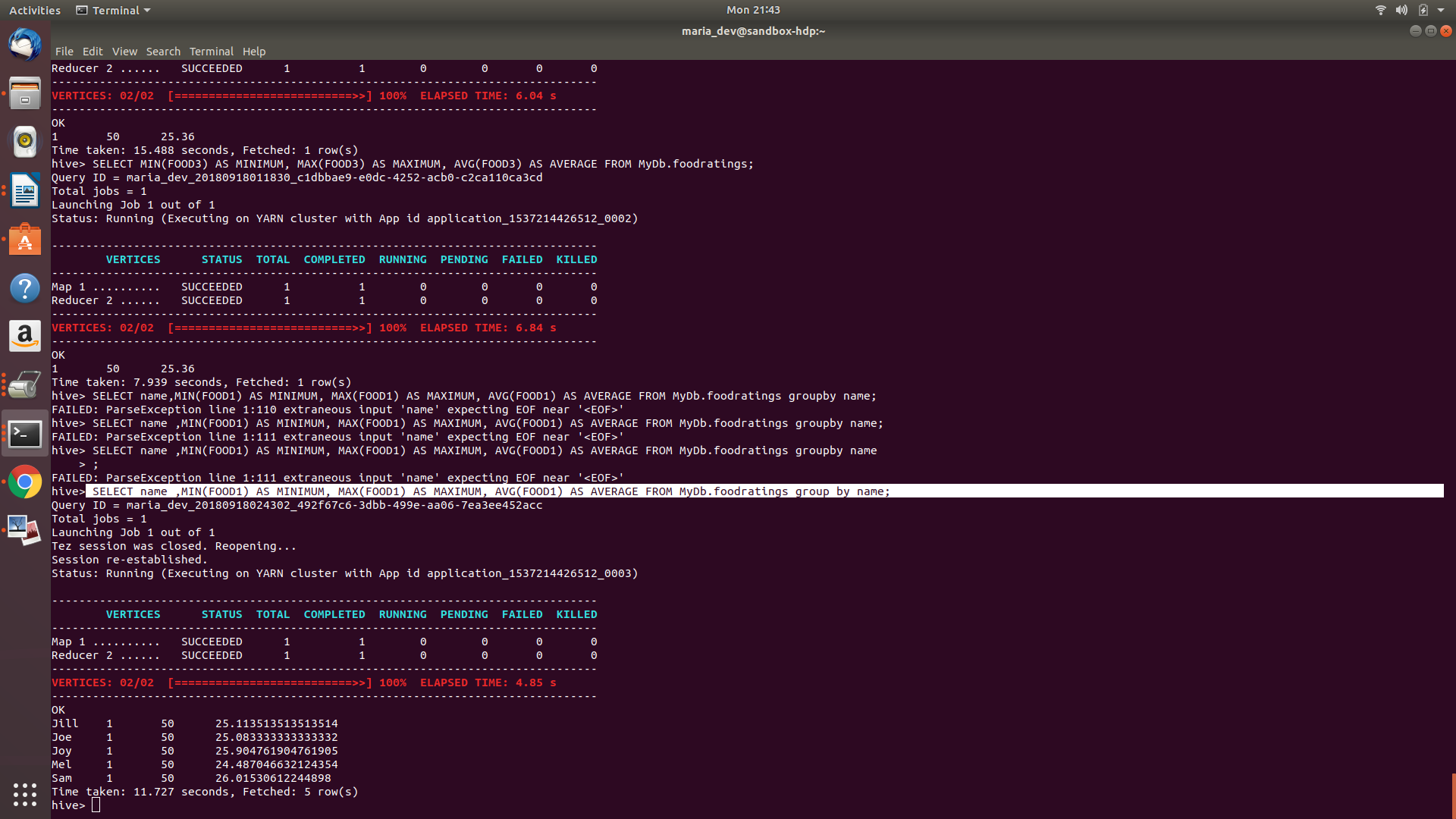
Execute a hive command to output the min, max and average of the values of the food3 column of the foodratings table.

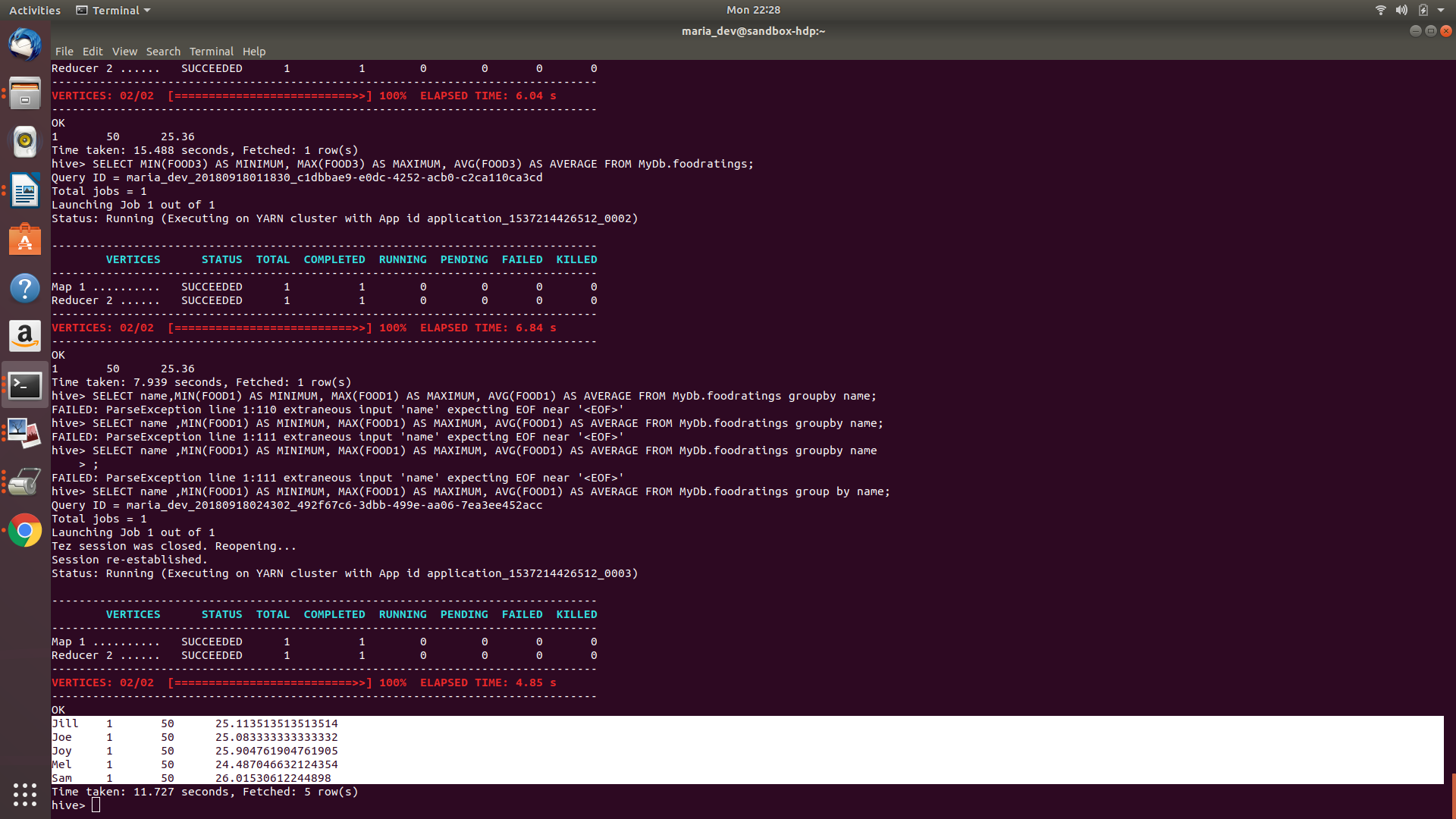
**Command:** SELECT MIN(FOOD3) AS MINIMUM, MAX(FOOD3) AS MAXIMUM, AVG(FOOD3) AS AVERAGE FROM MyDb.foodratings;



Exercise 3)

Execute a hive command to output the min, max and average of the values of the food1 column grouped by the first column ‘name’.

**Command:**SELECT name ,MIN(FOOD1) AS MINIMUM, MAX(FOOD1) AS MAXIMUM, AVG(FOOD1) AS AVERAGE FROM MyDb.foodratings group by name;

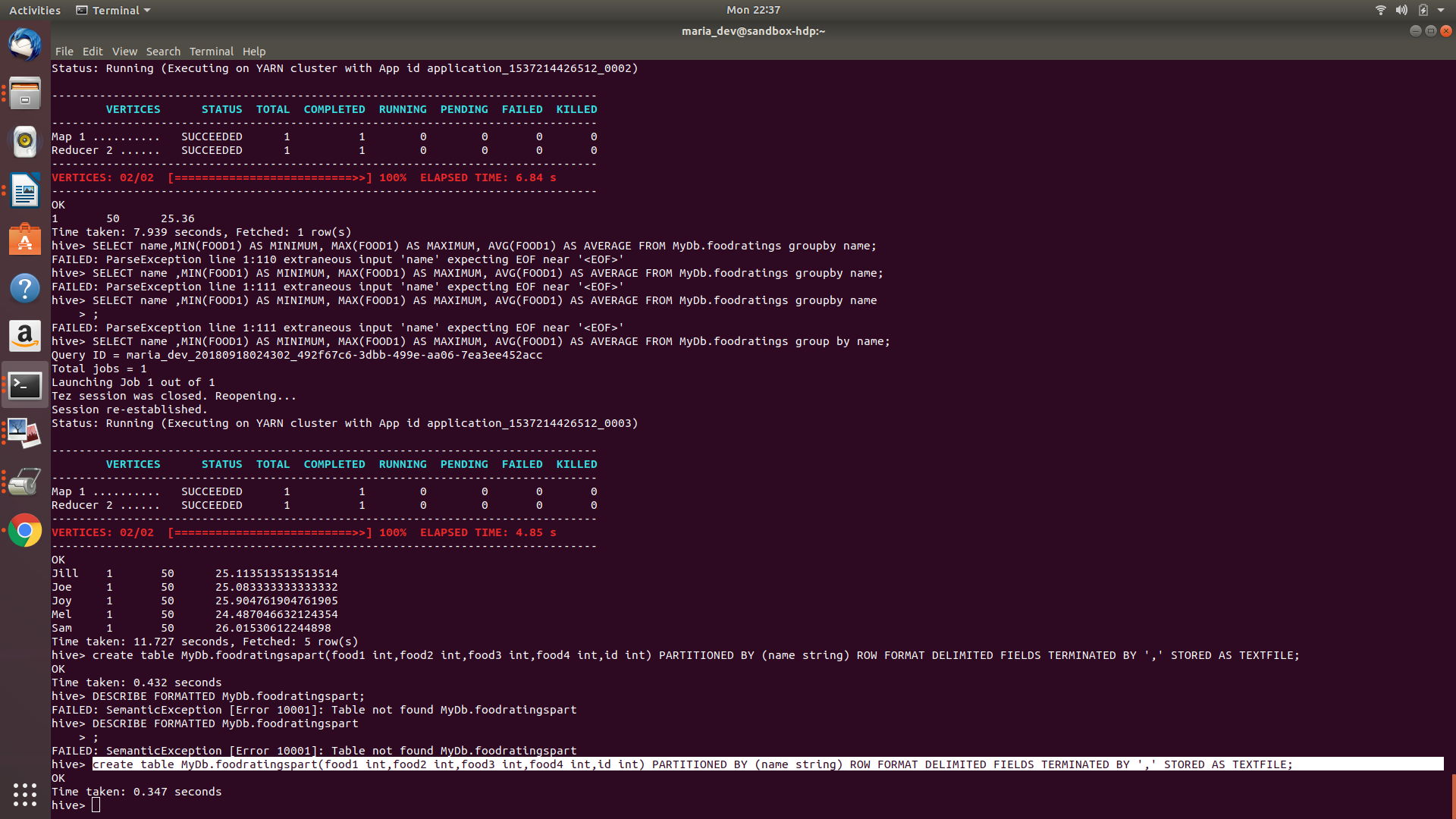


Exercise 4) 2 points

In MyDb create a partitioned table called ‘foodratingspart’

The partition field should be called ‘name’ and its type should be a string. The names of the non-partition columns should be food1, food2, food3, food4 and id and their types each an integer. The table should have storage format TEXTFILE and column separator a “,”. That is the underlying format should be a CSV file. No comments are needed for this table.

**Command:** create table MyDb.foodratingspart(food1 int,food2 int,food3 int,food4 int,id int) PARTITIONED BY (name string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

Execute a one shot Hive command of ‘DESCRIBE FORMATTED MyDb.foodratingspart’ and capture its output as the result of this exercise.

**Command:** DESCRIBE FORMATTED MyDb.foodratingspart;

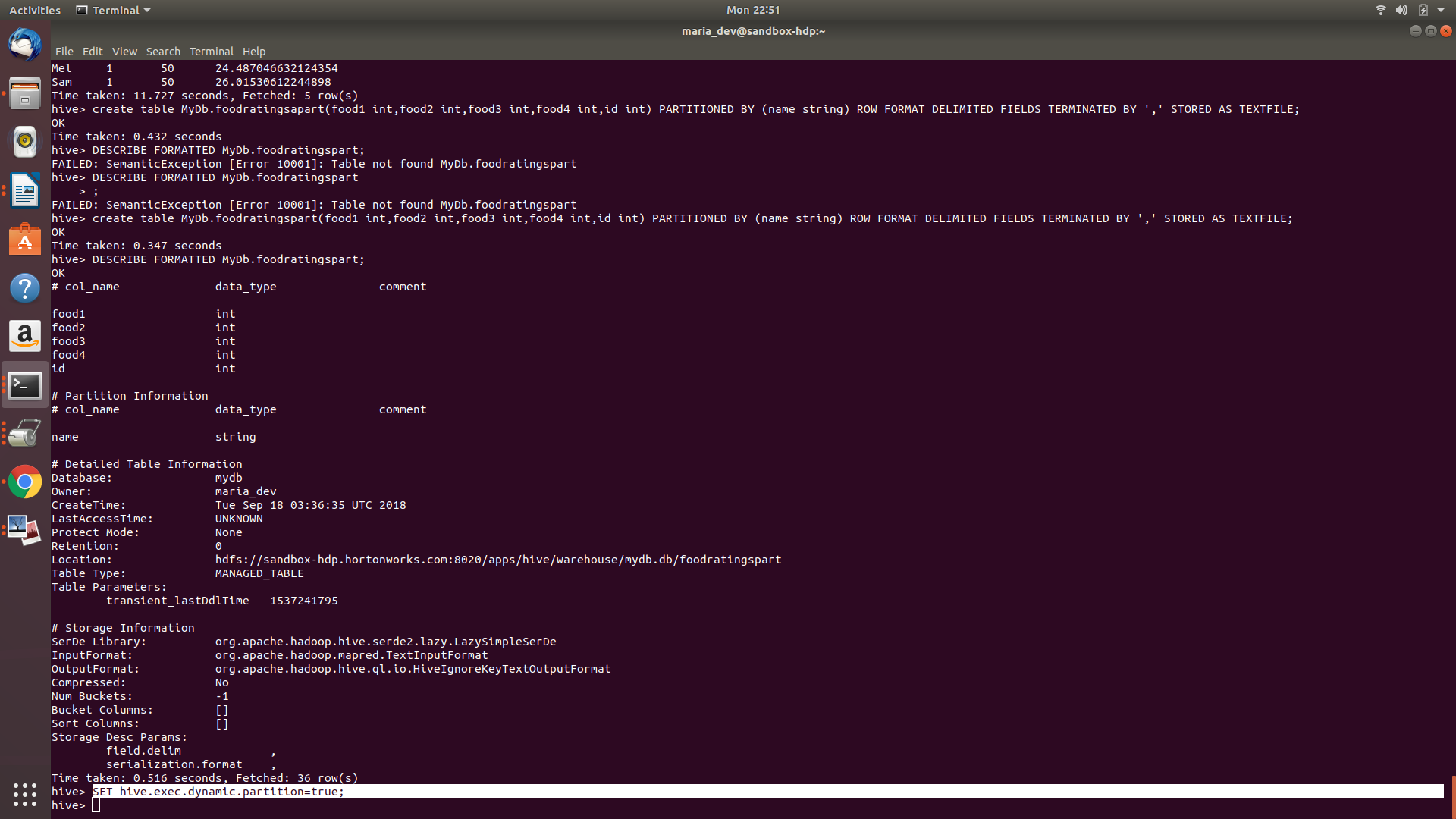


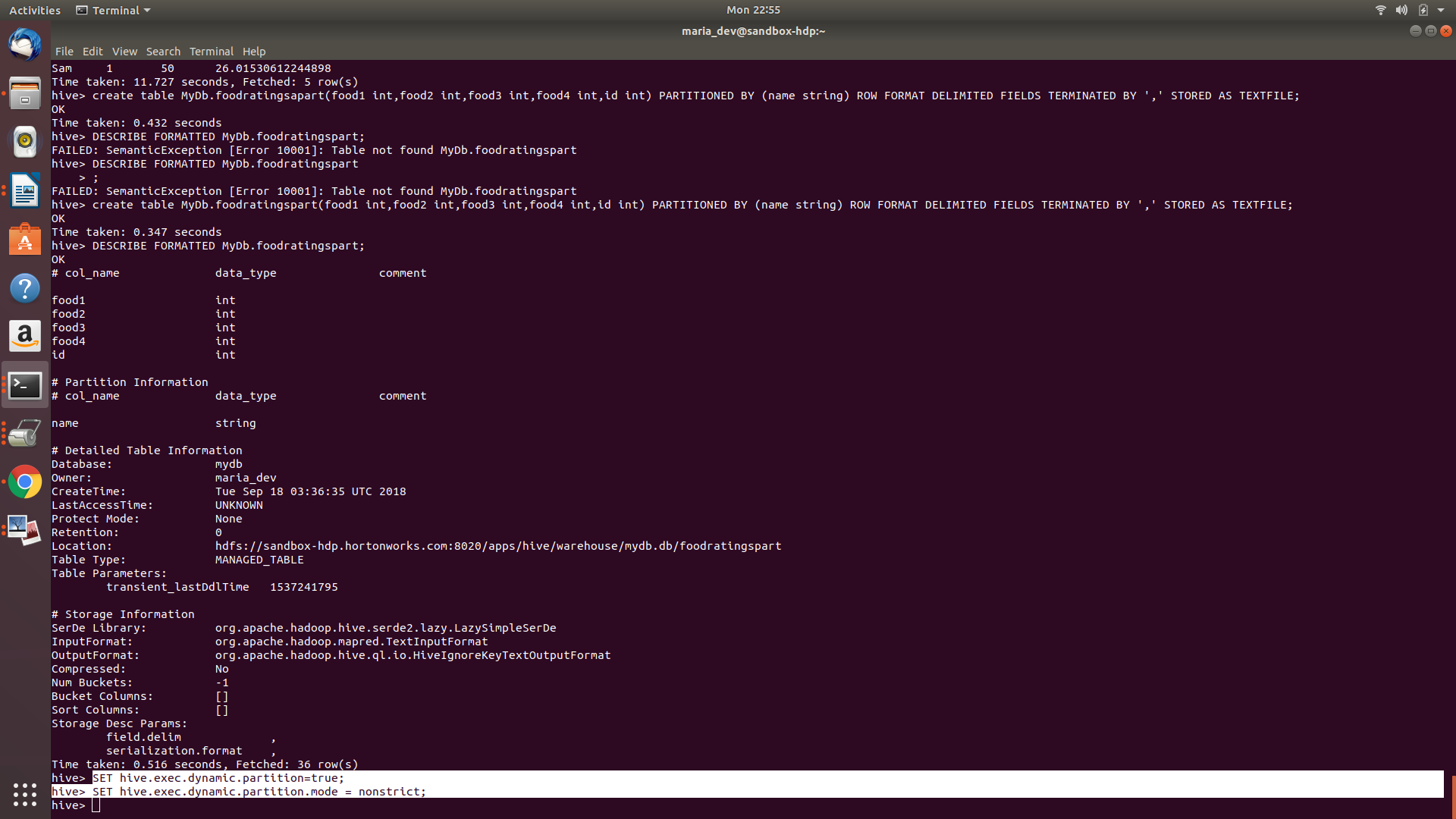
Exercise 5)

Configure Hive to allow dynamic partition creation as described in the lecture.

**Command:**

SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode = nonstrict;

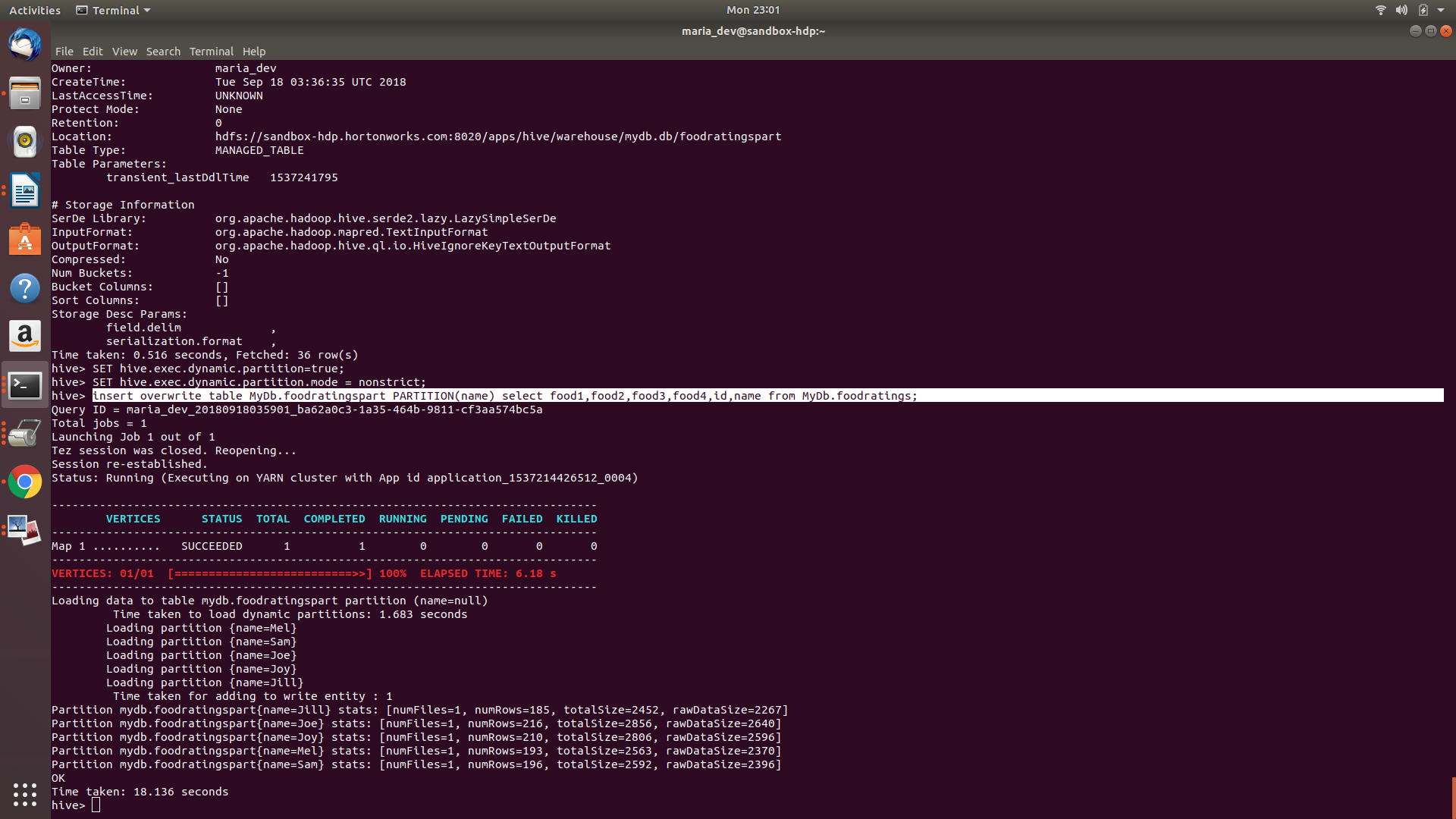


Use a hive command to copy from MyDB.foodratings into MyDB.foodratingspart to create a partitioned table from a non-partitioned one.

Hint: The ‘name’ column from MyDB.foodratings should be mentioned last in this command (whatever it is).

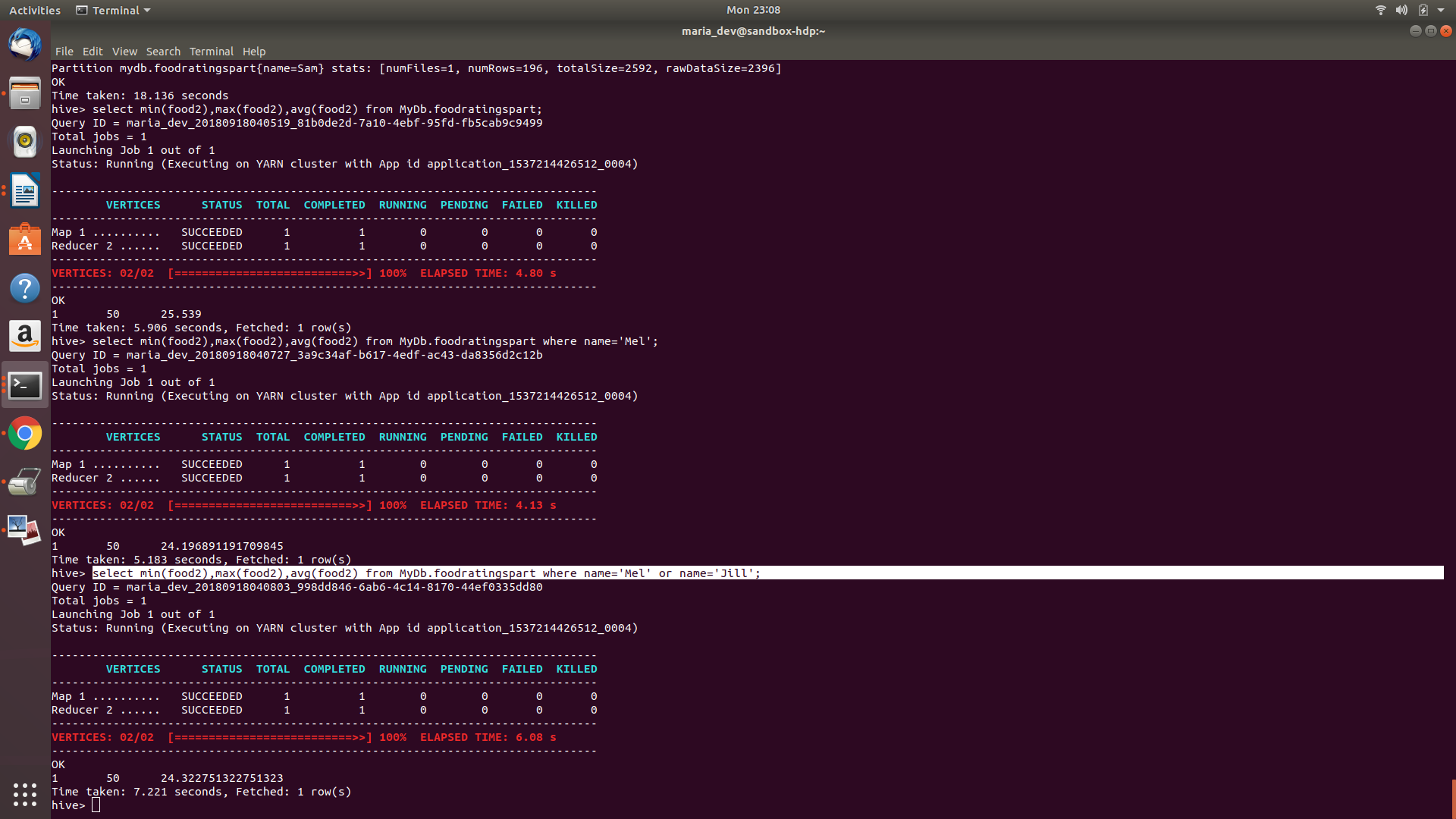
Provide a copy of the command you use to load the ‘foodratingspart’ table as a result of this exercise.

**Command:** insert overwrite table MyDb.foodratingspart PARTITION(name) select food1,food2,food3,food4,id,name from MyDb.foodratings;

****

Execute a hive command to output the min, max and average of the values of the food2 column of MyDB.foodratingspart where the food critic ‘name’ is either Mel or Jill.

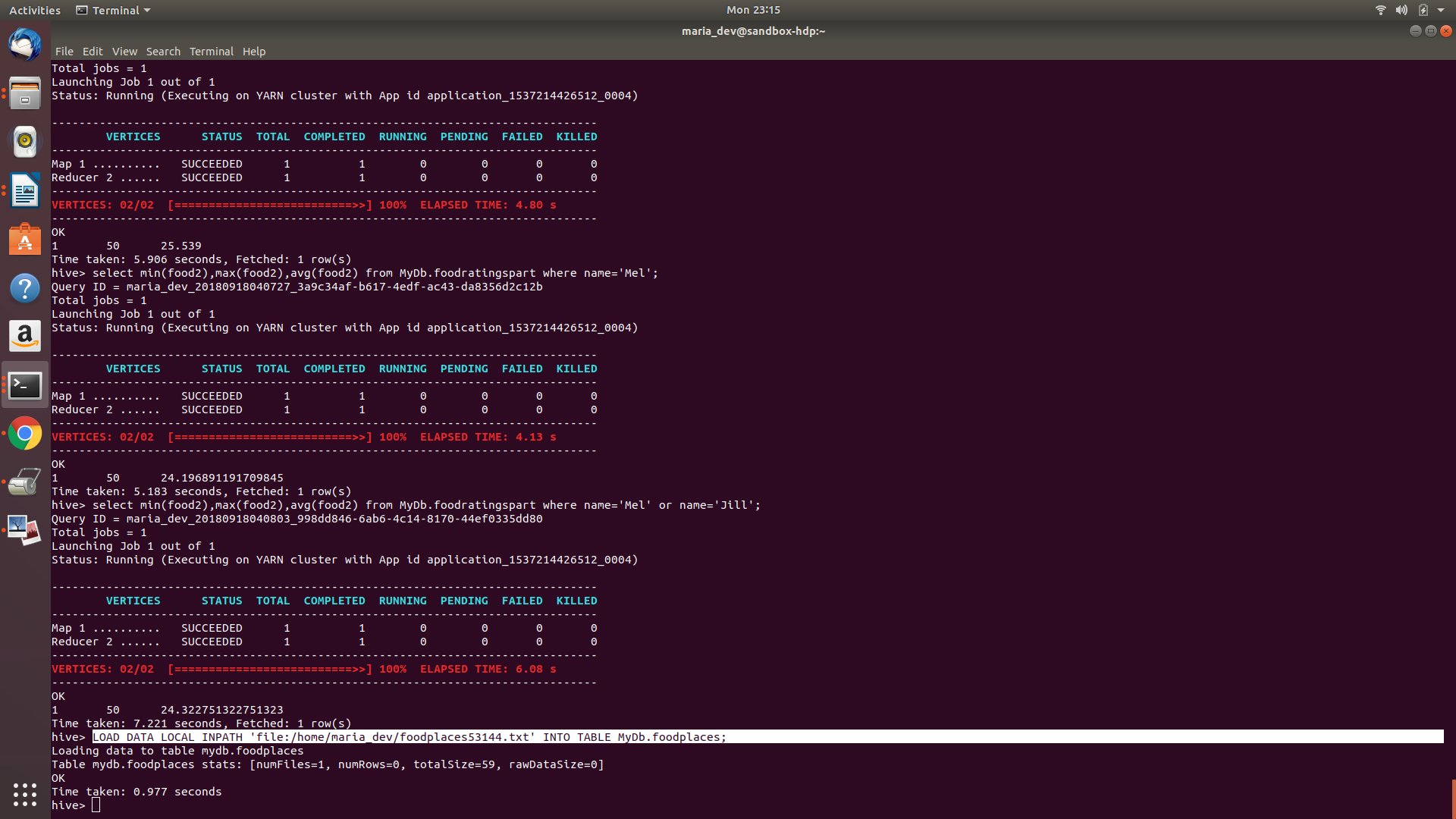
**Command:** select min(food2),max(food2),avg(food2) from MyDb.foodratingspart where name='Mel' or name='Jill';



Exercise 6)

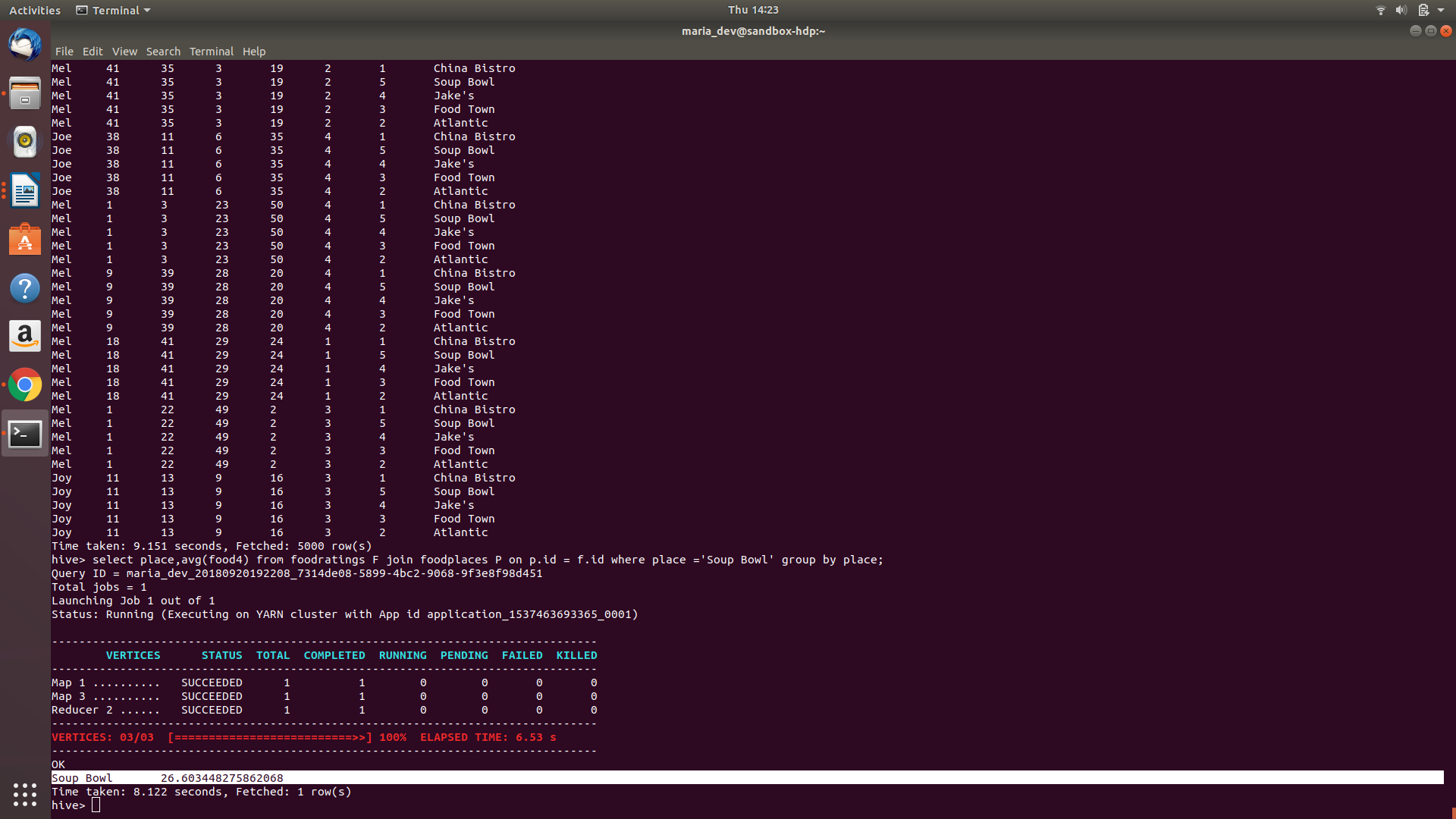
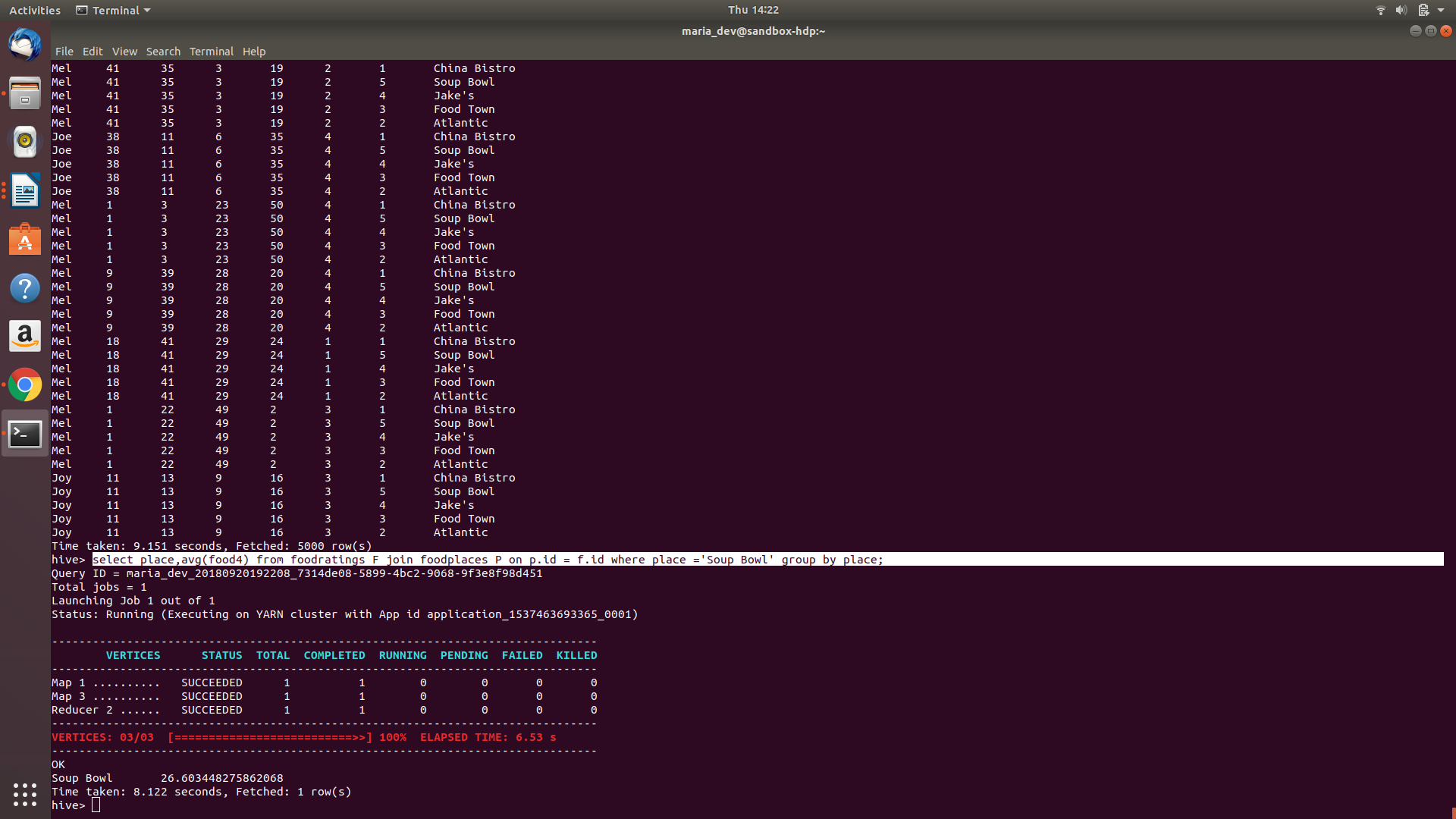
Load the foodplaces<.magic number>.txt file created using TestDataGen from your local file system into the foodplaces table.

**Command:**LOAD DATA LOCAL INPATH 'file:/home/maria\_dev/foodplaces53144.txt' INTO TABLE MyDb.foodplaces;



Use a join operation between the two tables (foodratings and foodplaces) to provide the average rating for field food4 for the restaurant ‘Soup Bowl’

**Command:** select place,avg(food4) from foodratings F join foodplaces P on p.id = f.id where place ='Soup Bowl' group by place;



Exercise 7)

Write a half page summary of the following article on the blackboard in section “Articles:”

Pig Latin: A Not-So-Foreign Language for Data Processing

**Answer:** PIG is fully implemented language, and compiles Pig Latin into physical plans that are executed over Hadoop. A Pig Latin program is a sequence of steps where each step carrying out a single high-level transformation. Effectively the program specifies the query execution plan itself. The program then compiles into map-reduce jobs which are run on.

Unlike traditional DBMS, Pig does not require data to be imported into system managed tables as it meant for ad-hoc analysis of data. To accommodate specialized data processing tasks, Pig Latin has extensive support for user-defined functions (UDFs). Since it targets only parallel processing, there is no inbuilt support for operations like non-equijoins or correlated sub queries. These operations can still be performed using User Defined Functions. Pig supports a flexible, nested data model with 4 supported types: ATOM, TUPLE, BAG and MAP. The inbuilt functions in PIG are FOREACH and GENERATE. For each COGROUP command, the compiler generates a map-reduce job where the map function assigns keys for grouping and the reduce function is initially a non-operative. Undesirable data is discarded using FILTER command, logical connectors and operators. Other PIG functions such as  LOAD, STORE, FILTER, JOIN, CROSS, DISTINCT, UNION and ORDER are similar in operation to equivalent SQL operations. This article also introduces Pig Pen ,which is and interactive debugging environment, that also helps to debug Pig Latin program. Typically, the programmer would write a program and run it on a dataset, or a subset of the dataset to check for correctness and change the program accordingly.

A lot of new functions have been added and it now comes with an interactive shell. Moreover, now User Defined Functions can be written in various scripting languages and not just Java. All these changes have made Pig more powerful and accessible than before.