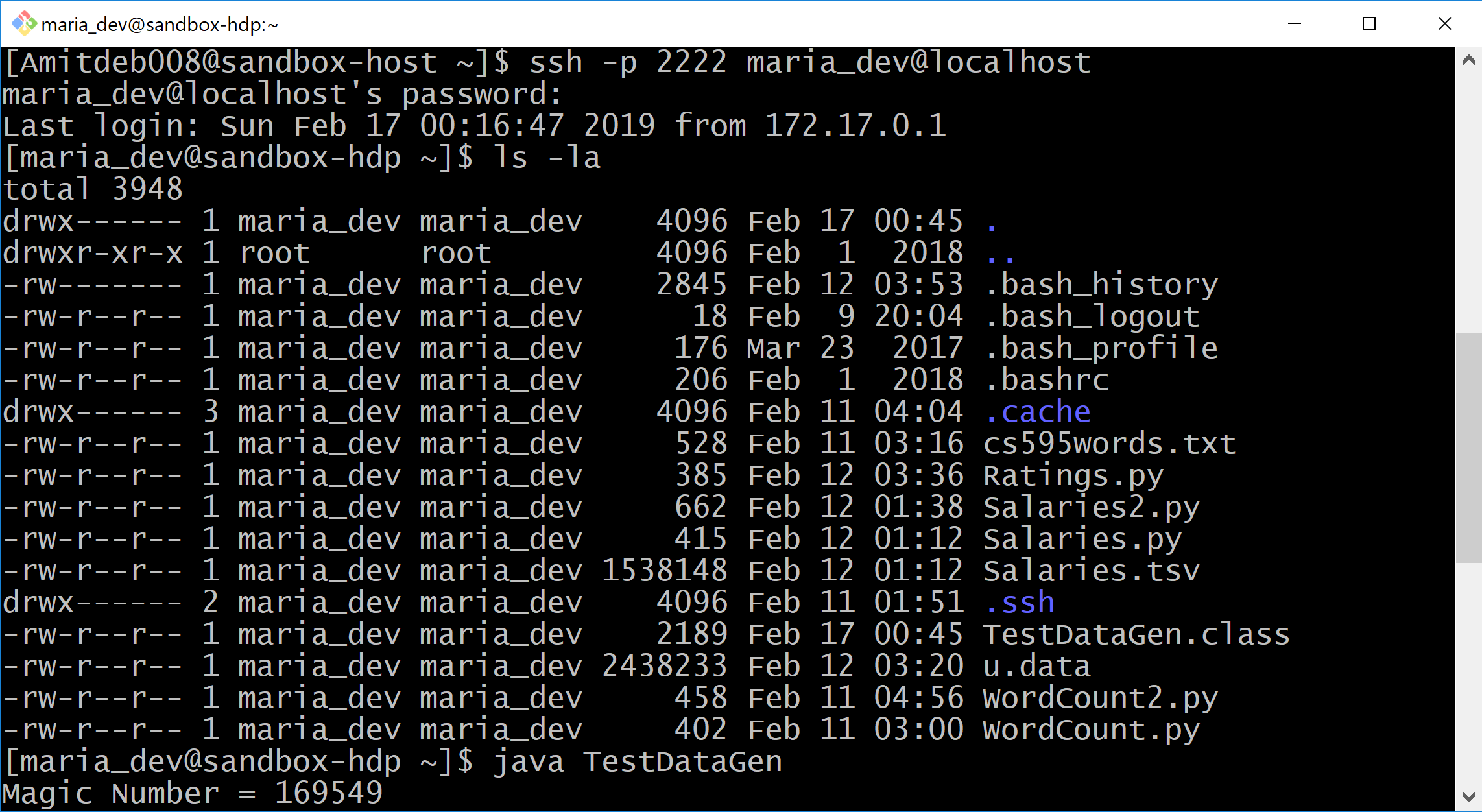
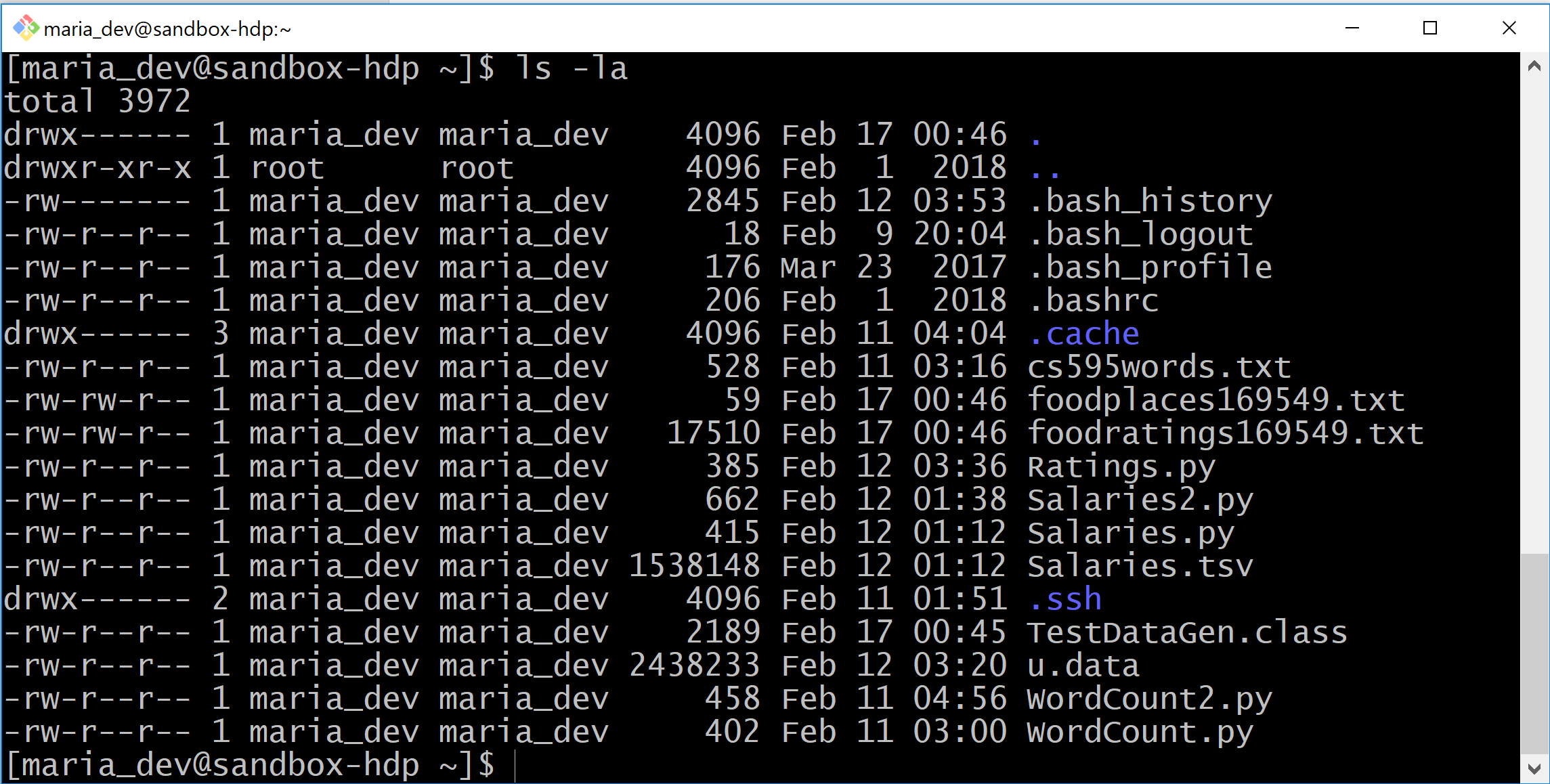
ASSIGNMENT - 4

**CSP - 554 BIG DATA TECHNOLOGIES**





Magic Number = 169549

Exercise 1) 2 points.

Create a Hive database called MyDb.

CREATE DATABASE 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).

CREATE TABLE IF NOT EXISTS mydb.foodratings (

name STRING COMMENT 'Food Critic Name',

food1 INT COMMENT 'Ratings for food1',

food2 INT COMMENT 'Ratings for food2',

food3 INT COMMENT 'Ratings for food3',

food4 INT COMMENT 'Ratings for food4',

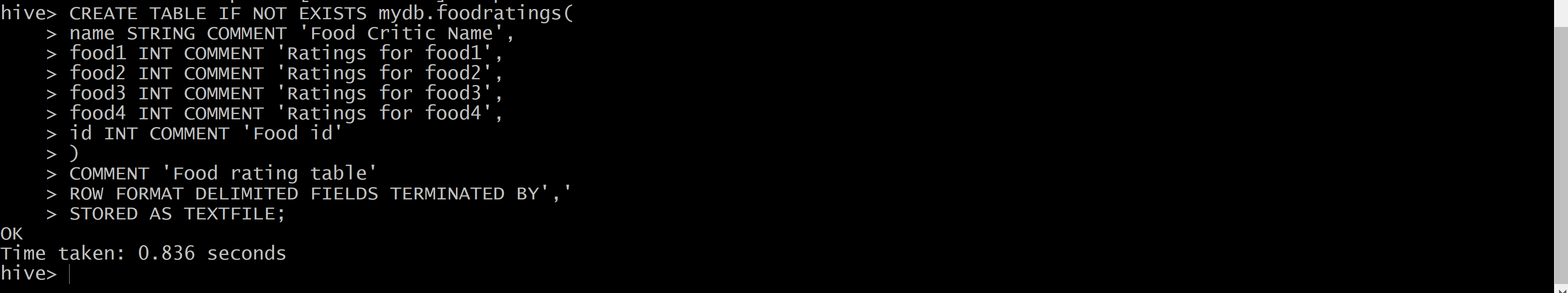
id INT COMMENT 'Food id'

)

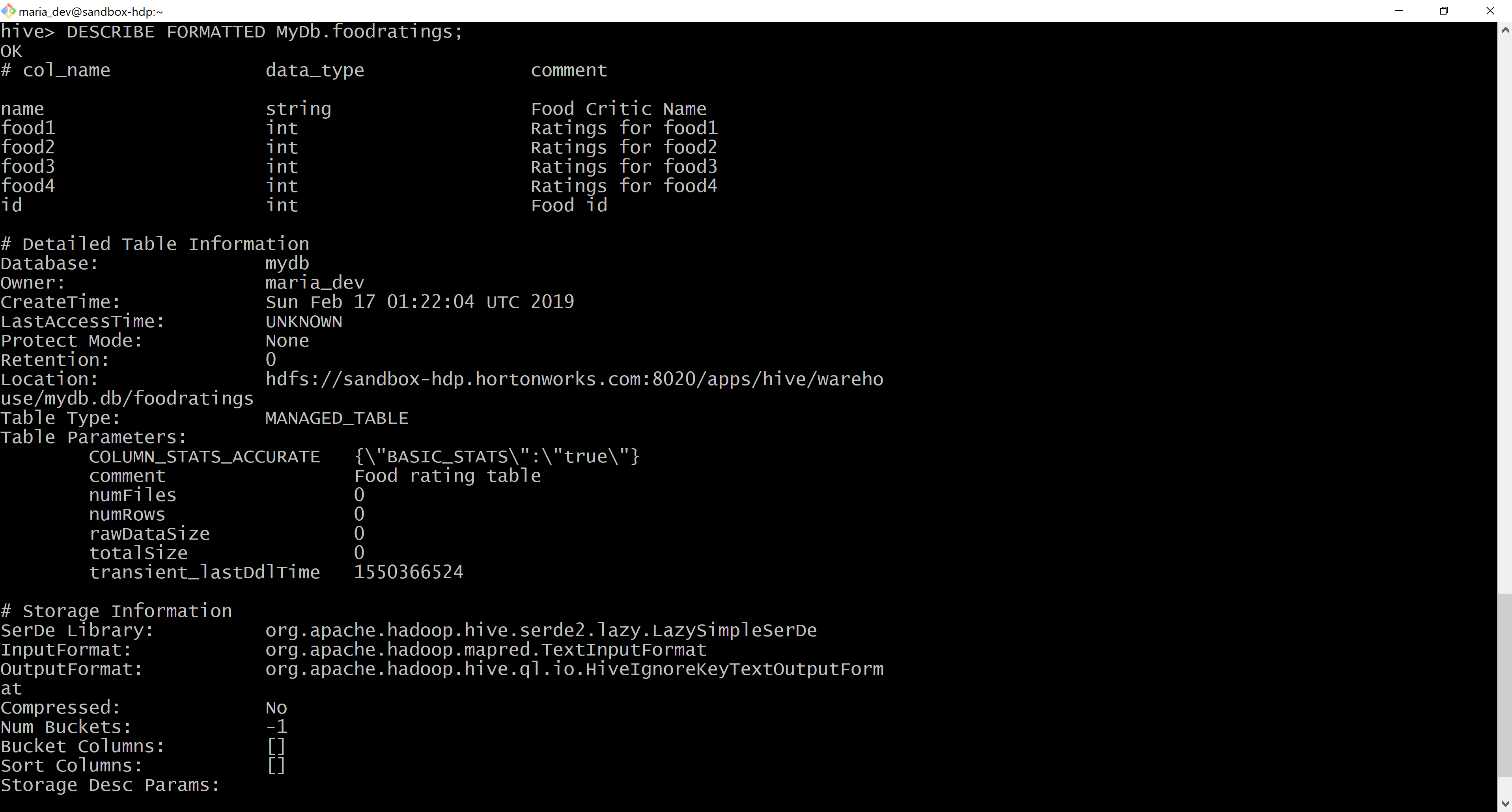
COMMENT 'Food rating table'

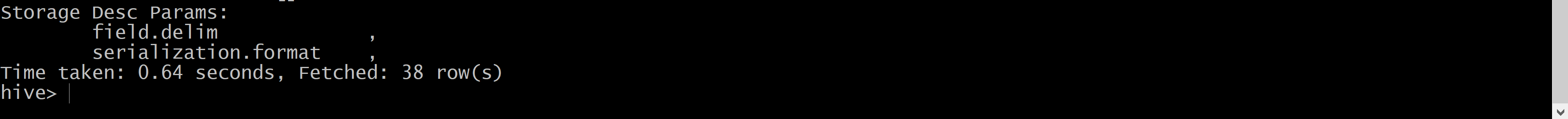
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;



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





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.

CREATE TABLE IF NOT EXISTS mydb.foodplaces (

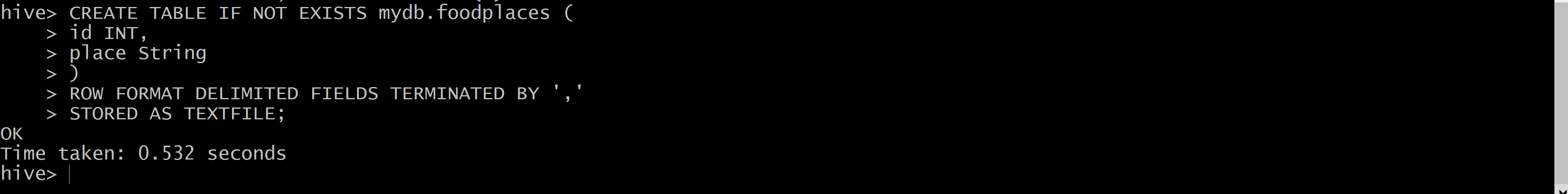
id INT,

place String

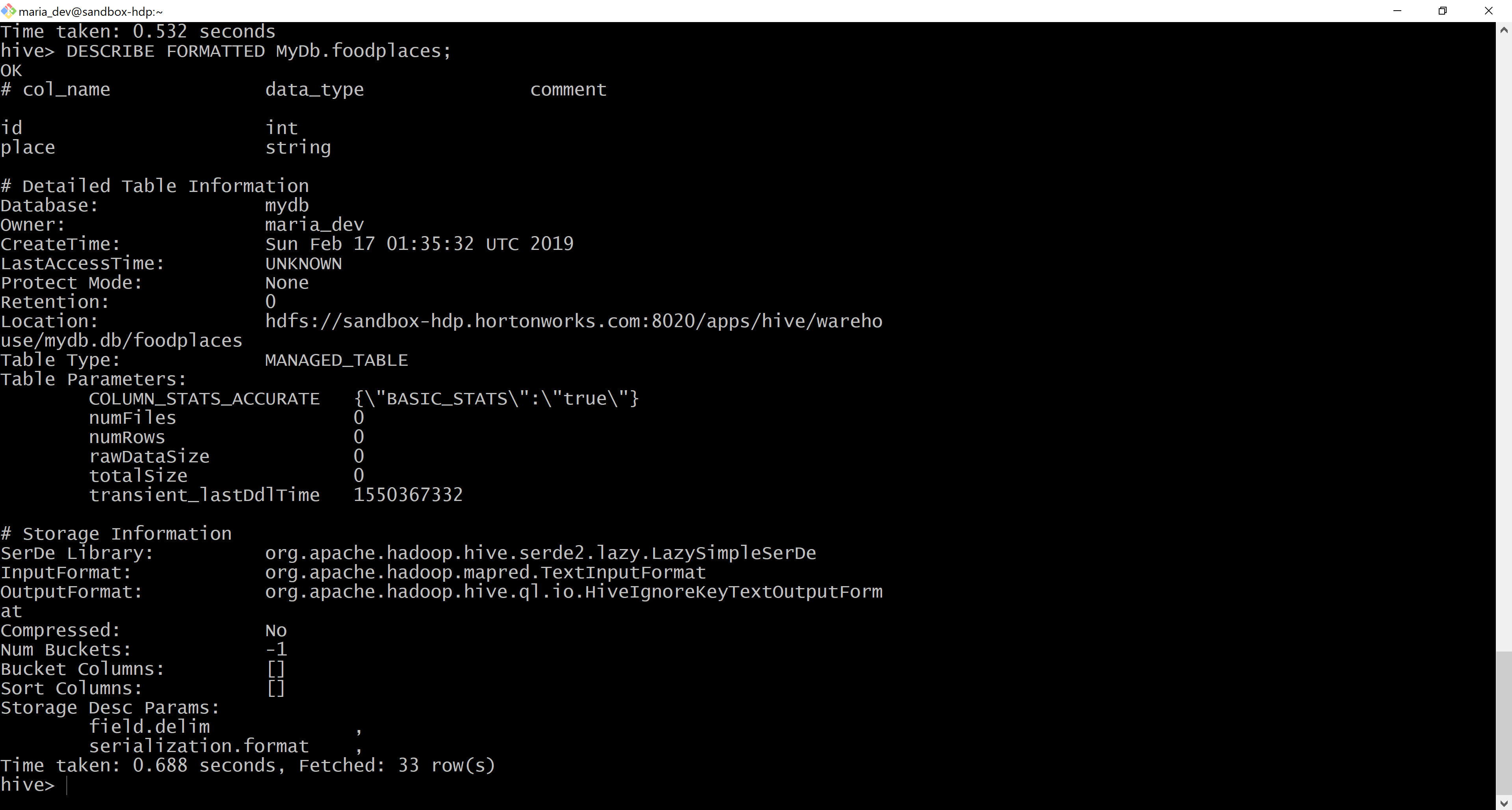
)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;



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

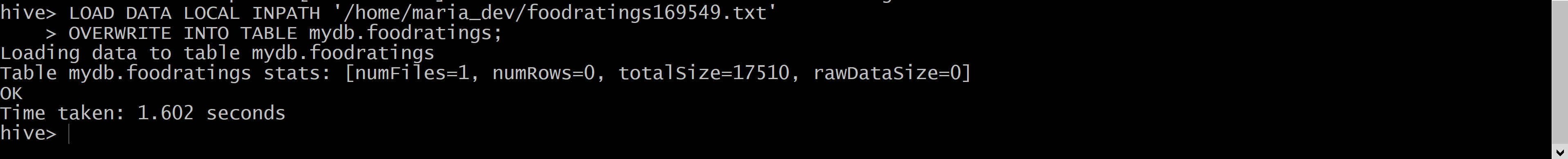


Exercise 2) 2 points

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

LOAD DATA LOCAL INPATH '/home/maria\_dev/foodratings169549.txt'

OVERWRITE INTO TABLE mydb.foodratings;



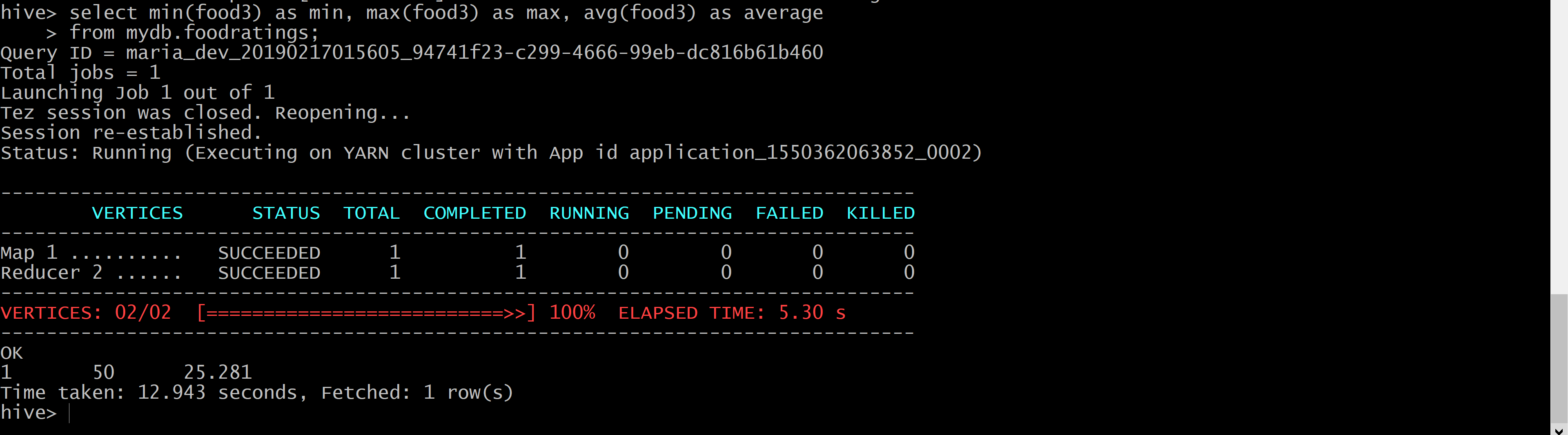
Execute a hive command to output the min, max and average of the values of the food3 column of the foodratings table.

A copy of the hive command you wrote, the output of this query and the magic number are the result of this exercise.

select min(food3) as min, max(food3) as max, avg(food3) as average

from mydb.foodratings;

Magic Number: 169549



Exercise 3) 2 points

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

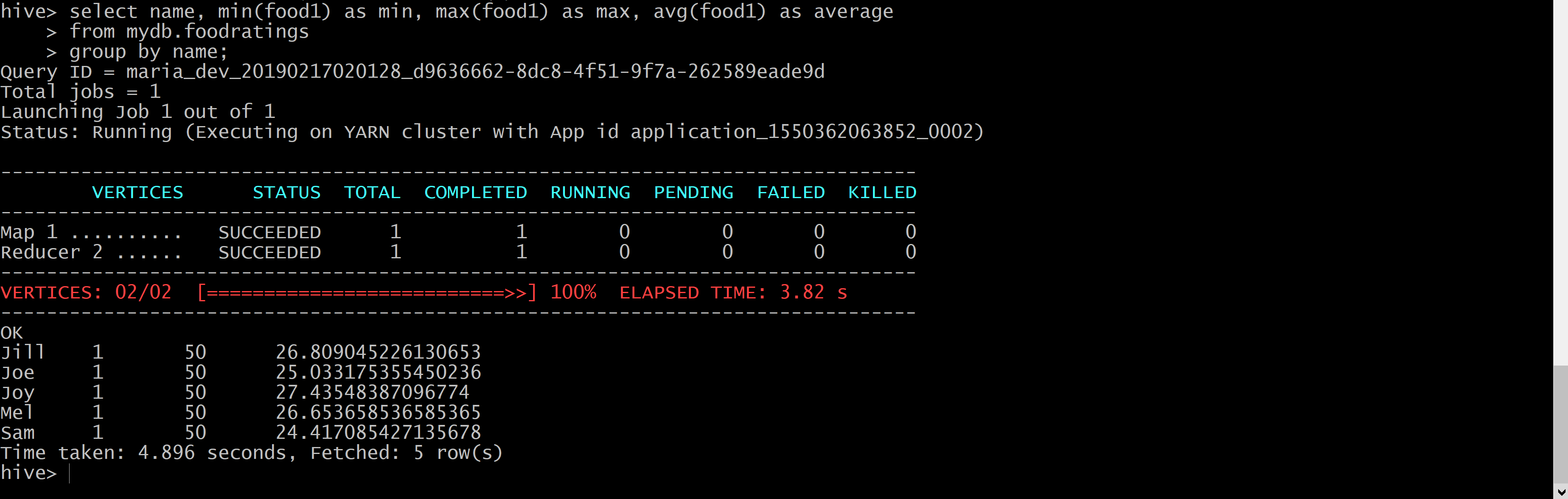
A copy of the hive command you wrote, the output of this query and the magic number are the result of this exercise.

select name, min(food1) as min, max(food1) as max, avg(food1) as average

from mydb.foodratings

group by name;

Magic Number: 169549



Exercise 4) 2 points.

In MyDb create a partitioned table called ‘foodratingspart’

CREATE TABLE IF NOT EXISTS mydb.foodratingspart (

food1 INT,

food2 INT,

food3 INT,

food4 INT,

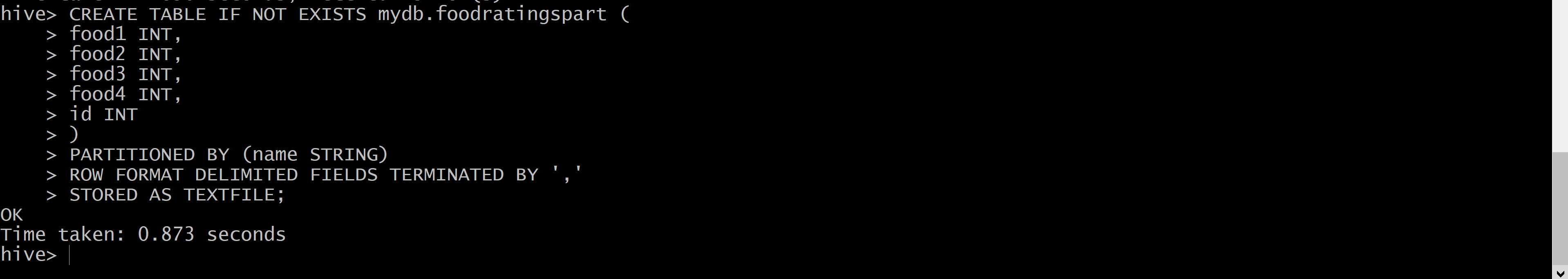
id INT

)

PARTITIONED BY (name STRING)

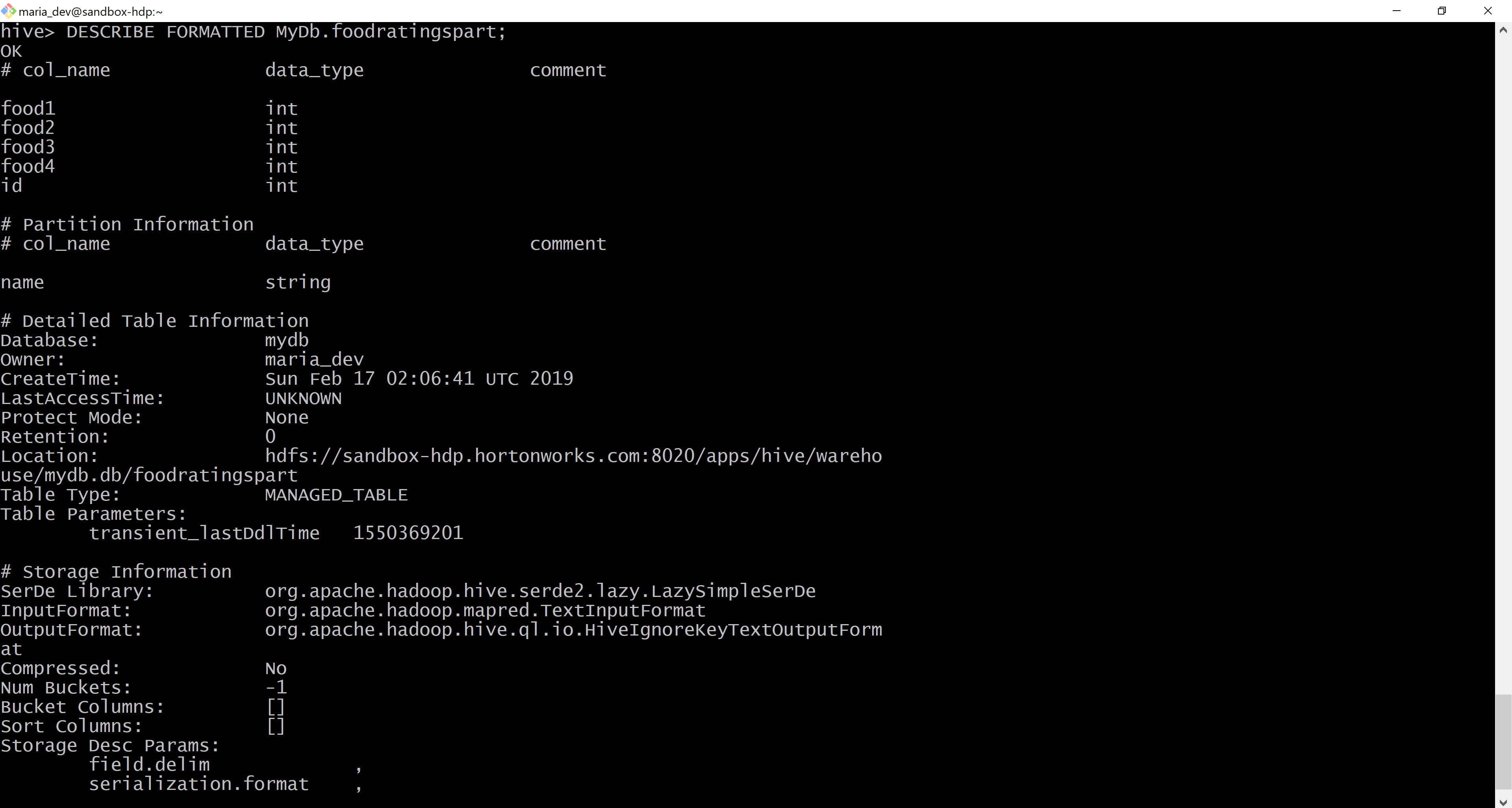
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

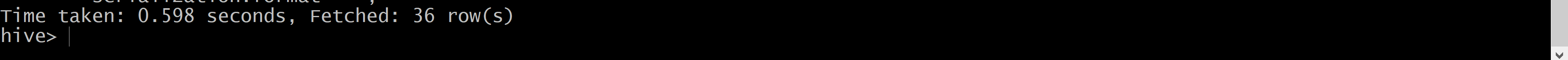
STORED AS TEXTFILE;



Execute a one shot Hive command of ‘DESCRIBE FORMATTED MyDb.foodratingspart’ and capture its

output as the result of this exercise





Exercise 5) 2 points

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

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

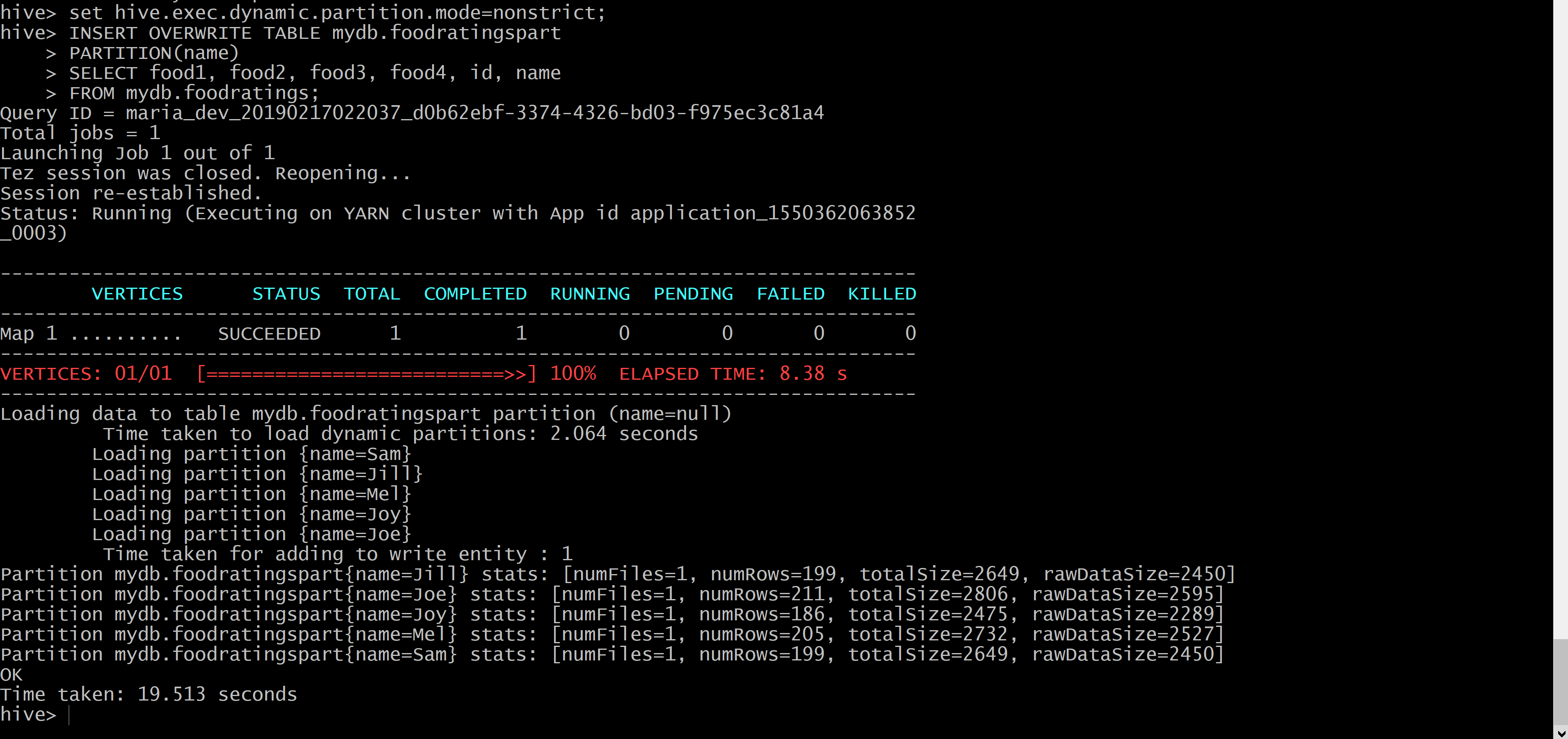
exercise.

INSERT OVERWRITE TABLE mydb.foodratingspart

PARTITION (name)

SELECT food1, food2, food3, food4, id, name

FROM mydb.foodratings;



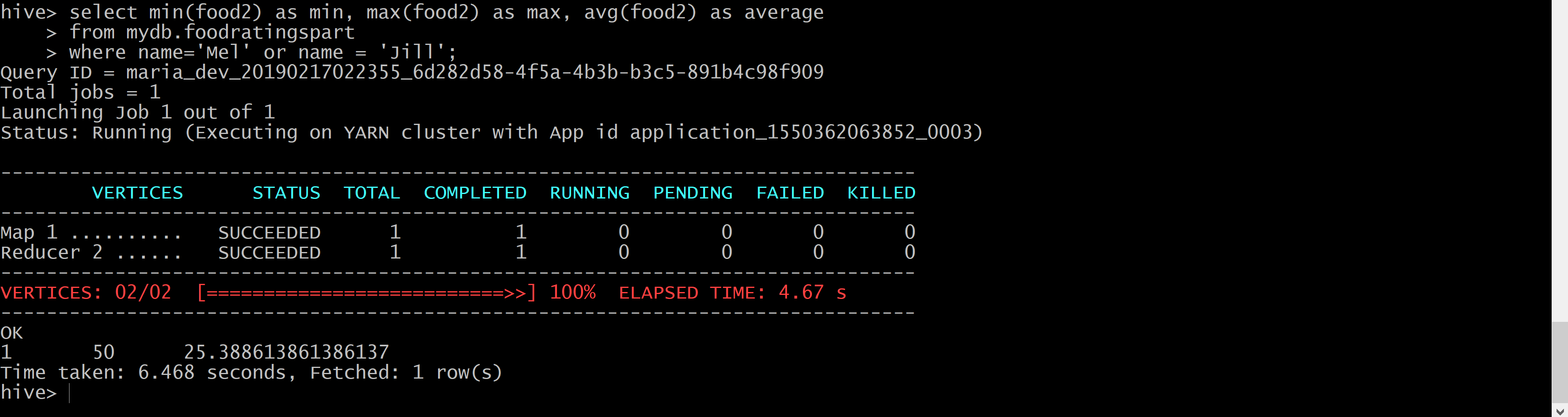
Execute a hive command to output the min, max and average of the values of the food2 column of MyDB.foodratingspart where the food critic ‘name’ is either Mel or Jill.

The query and the output of this query are other results of this exercise.

select min(food2) as min, max(food2) as max, avg(food2) as average

from mydb.foodratingspart

where name='Mel' or name = 'Jill';



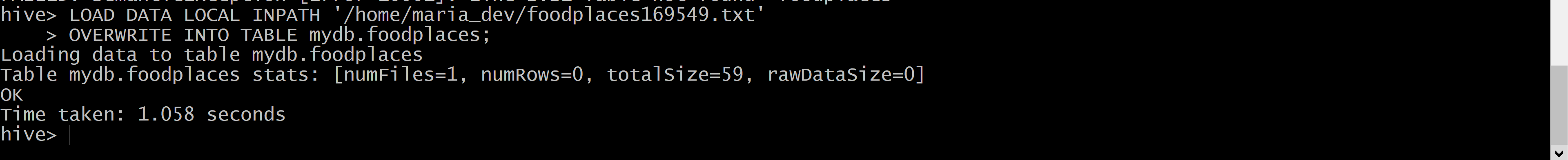
Exercise 6) 2 points

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

system into the foodplaces table.

LOAD DATA LOCAL INPATH '/home/maria\_dev/foodplaces169549.txt'

OVERWRITE 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’

The output of this query is the result of this exercise.

select fp.place, avg(fr.food4) as average

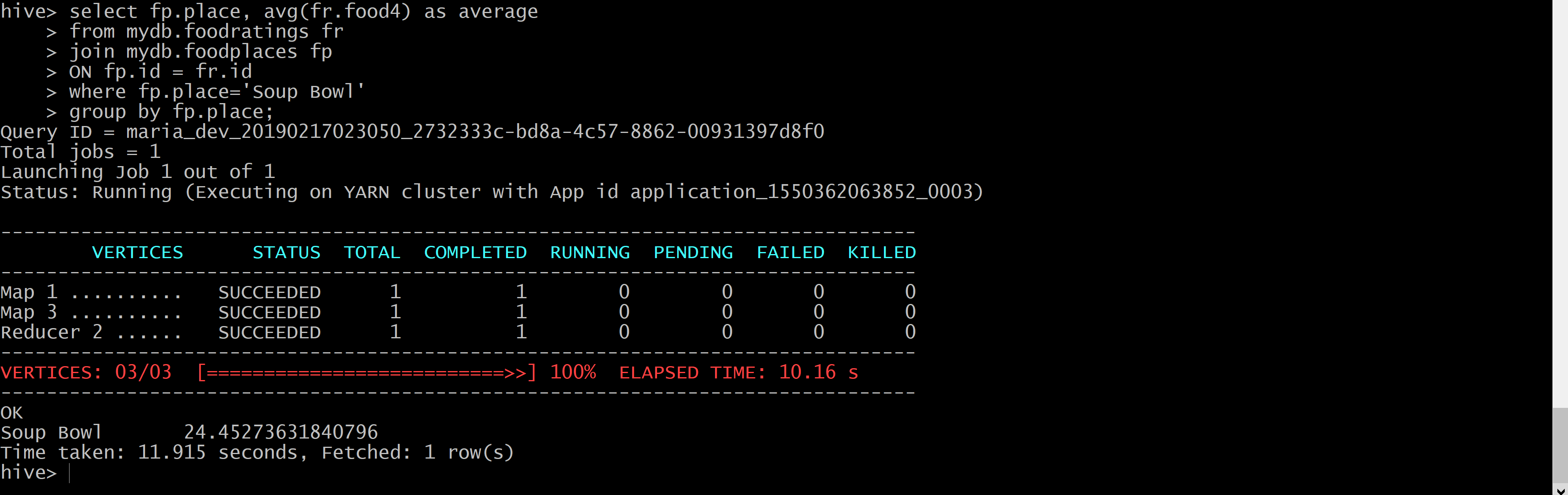
from mydb.foodratings fr

join mydb.foodplaces fp

ON fp.id = fr.id

where fp.place='Soup Bowl'

group by fp.place;



Exercise 7) 2 points Extra Credit

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

**Summary on Pig Latin**

The article discuses about a new data processing environment developed by Yahoo! called as Pig and it’s language Pig Latin which is an open source project in Apache incubator. Pig Latin can process massive amount of data set and streaming data which has made it more popular. Pig Latin is implemented on Hadoop and MapReduce to utilize the parallelization. Pig Latin consists of the spirit of SQL and procedural programming used in MapReduce. A program written in Pig Latin is much like SQL where everything is explained like a data flow graph thereby making it easier for an ad-hoc data analyst and it is more flexible because it comes with a novel debugging environment which helps in dealing large data sets. It is stylistically different from the SQL approach where each step specifies a single high-level data transformation which makes it so optimistic language. Pig uses high-level relational algebra style primitives like "GROUP," "FILTER" helps it for optimization. Pig Latin also creates a Nested data model which is hard to do in traditional SQL queries and better than normalization. Another advantage of Pig Latin nested data model is that it allows users to write user-defined functions (UDF) which helps a user to create custom processing tasks. Pig UDF's can perform all aspects of processing like grouping, filtering, joining and per tuple processing and takes non-atomic parameters as input and shows output non-atomic values this explains how flexibility of Pig Latin. Pig Latin has four data models atom (simple atomic value), Tuple (sequence of fields), Bag (collection of tuples), Map(collection of data items with associated key).

Keywords used in Pig Latin:

Loading input data: LOAD

per-tuple processing: FOREACH

Filtering data: FILTER

Grouping related data: COGROUP or GROUP

Joining of two data sets: JOIN

We can create a MapReduce application using FOREACH and GROUP keywords and some of the other keywords are UNION, CROSS, ORDER, DISTINCT, and STORE for asking output.

Pig Latin uses pig pen debugging environment which creates a side dataset automatically for avoiding debugging errors. This dataset is dynamically constructed and called as sandbox data set. Three primary objectives in selecting a sandbox dataset realism (sandbox data set should be a subset of the actual dataset), conciseness (small as possible) and completeness(should completely illustrate the key semantics of each command).

Pig Latin is widely used in Yahoo!, Facebook and other internet companies for ad-hoc analysis but still there are the number of promising directions that are yet unexplored in Pig system like a safe optimizer where Pig Latin depends on unknown data characteristics, but it needs to follow the Pig Latin sequence written by the user only. Other problems like user interface that need to be more user-friendly.