**CSP 554 – Assignment #4**

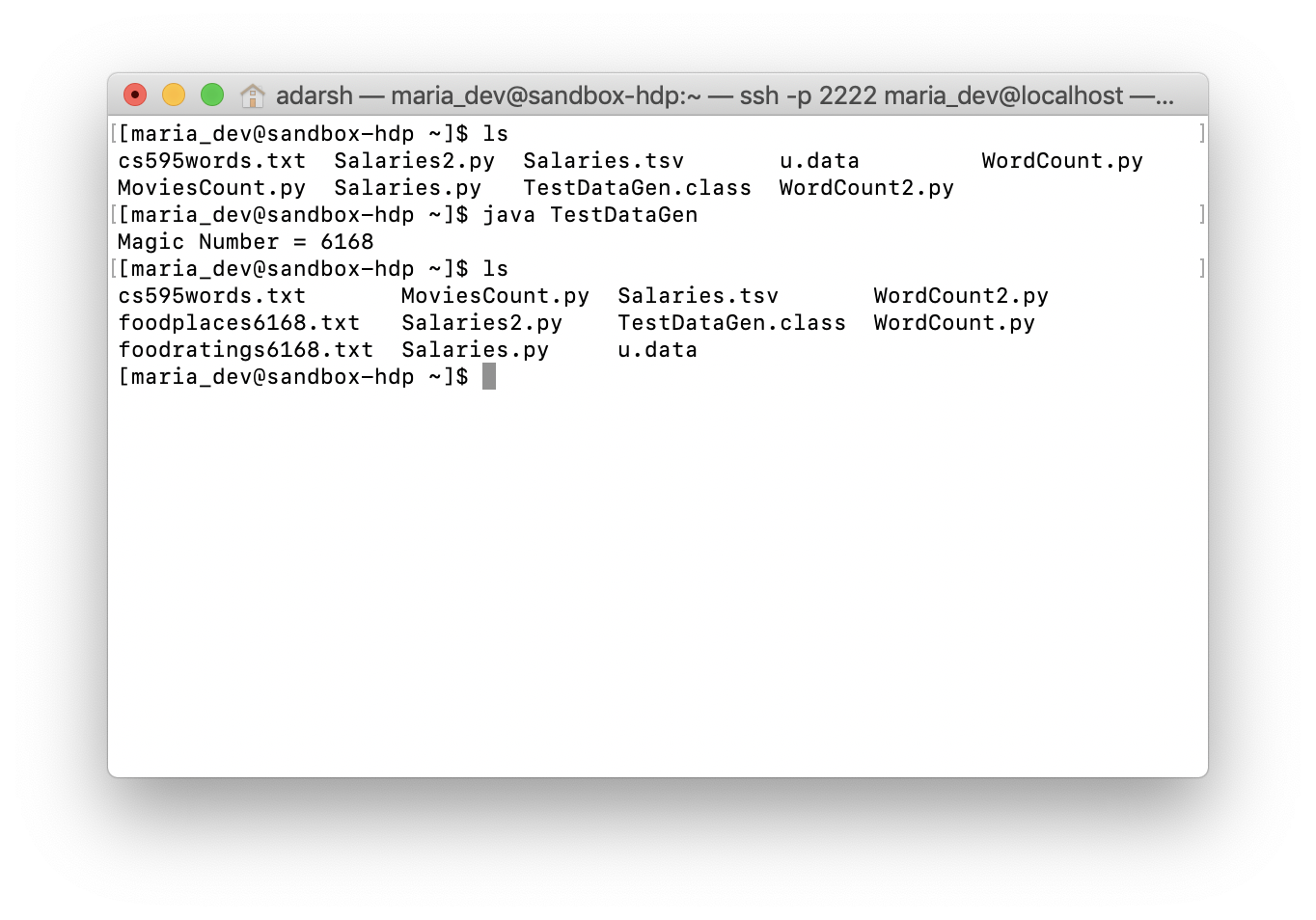
Name: Adarsh Mathad Vijayakumar

CWID: A20424847

Email ID: avijayakumar@hawk.iit.edu

Command: java TestDataGen

Output: Magic Number = 6168



**Exercise 1:**

1. CREATE DATABASE mydb;
2. CREATE DATABASE IF NOT EXISTS mydb;

use mydb;

DROP TABLE IF EXISTS foodratings;

CREATE TABLE IF NOT EXISTS mydb.foodratings (

name STRING COMMENT 'Food critic name',

food1 INT COMMENT 'Food1 rating',

food2 INT COMMENT 'Food2 rating',

food3 INT COMMENT 'Food3 rating',

food4 INT COMMENT 'Food4 rating',

id INT COMMENT 'Restaurant ID')

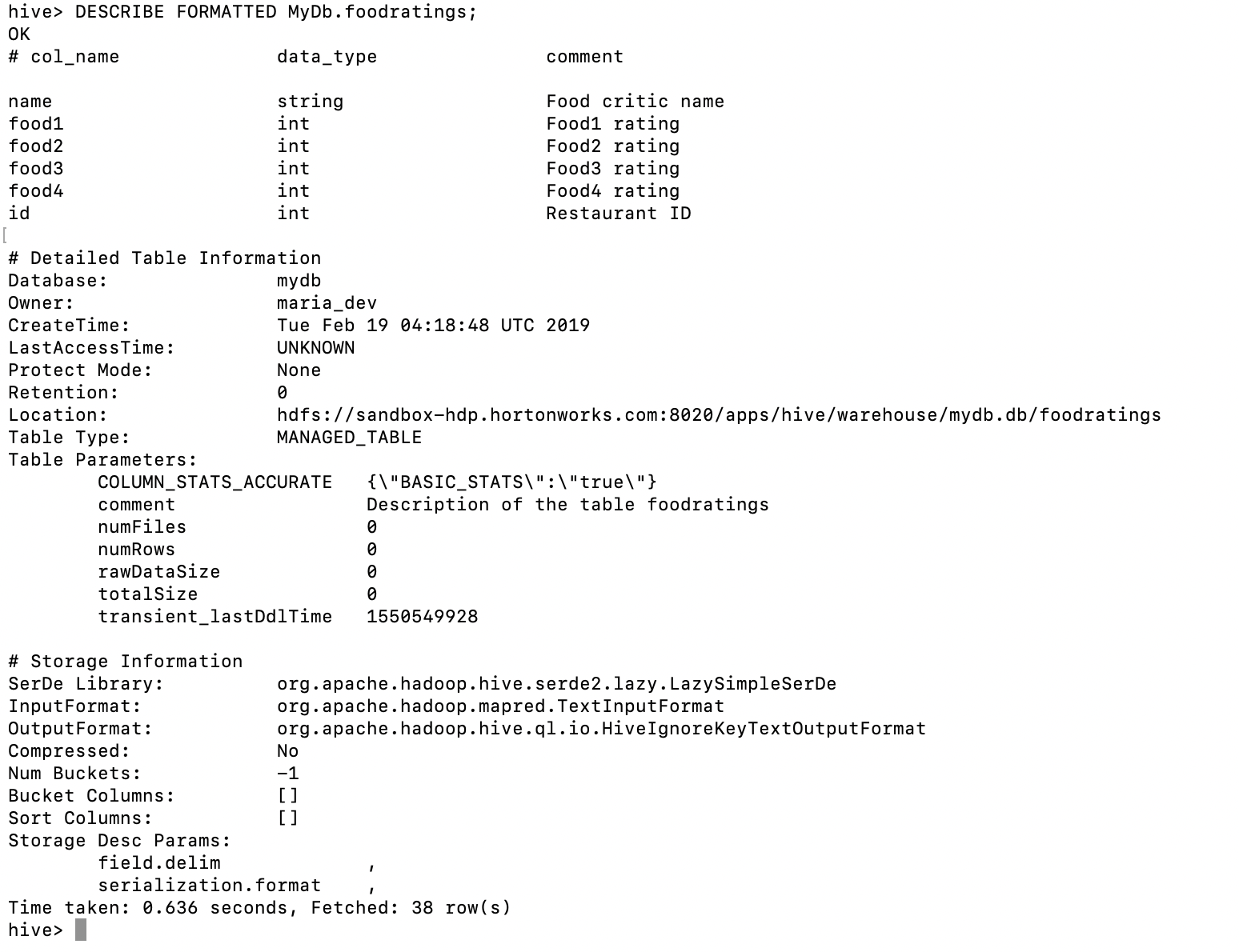
COMMENT 'Description of the table foodratings'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

1. DESCRIBE FORMATTED MyDb.foodratings;

Output:



1. CREATE DATABASE IF NOT EXISTS mydb;

use mydb;

DROP TABLE IF EXISTS foodplaces;

CREATE TABLE IF NOT EXISTS mydb.foodplaces (

id INT,

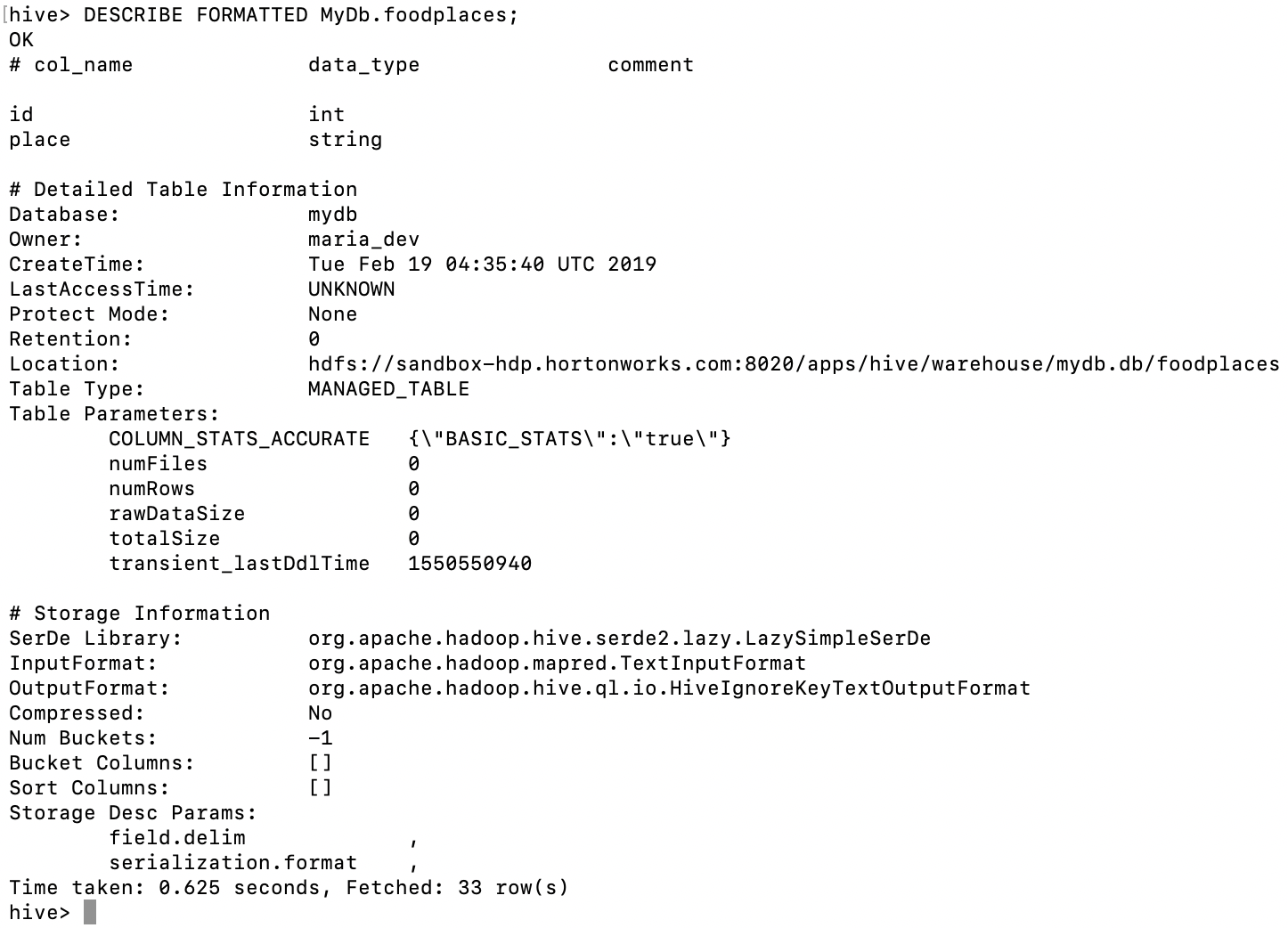
place STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

1. DESCRIBE FORMATTED MyDb.foodplaces;

Output:



**Exercise 2:**

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

LOAD DATA LOCAL INPATH './foodratings6168.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.

Magic Number = 6168

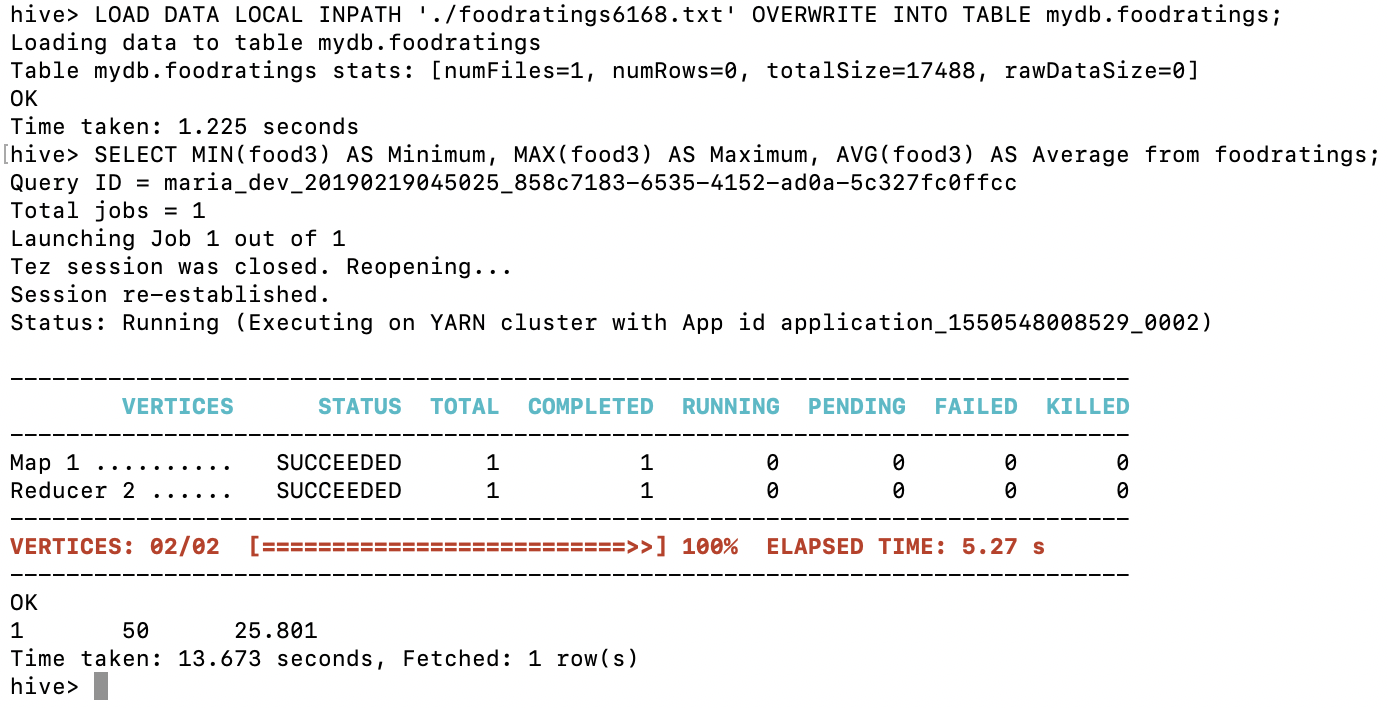
Hive command:

SELECT MIN(food3) AS Minimum, MAX(food3) AS Maximum, AVG(food3) AS Average from foodratings;

Output:

Minimum 1, Maximum 50, Average 25.801

Output Screenshot:



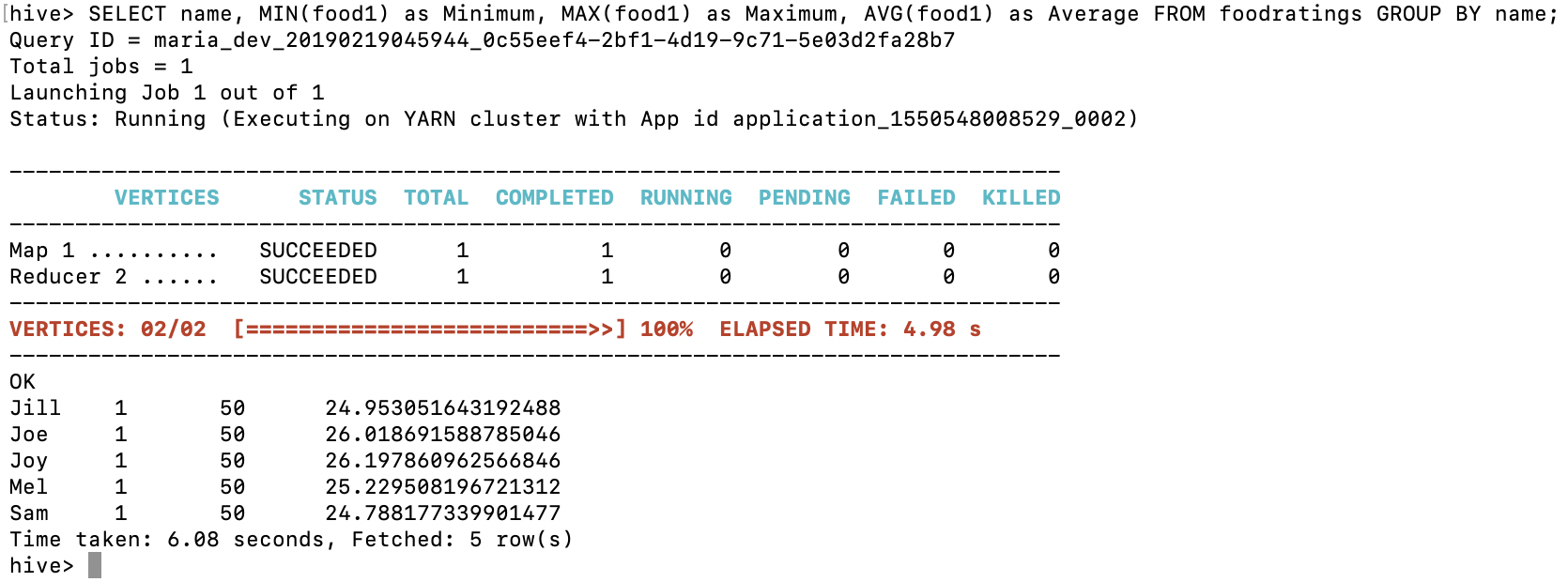
**Exercise 3:**

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

SELECT name, MIN(food1) as Minimum, MAX(food1) as Maximum, AVG(food1) as Average FROM foodratings GROUP BY name;

[Magic Number: 6168]

Output:



**Exercise 4:**

In MyDb create a partitioned table called ‘foodratingspart’

CREATE DATABASE IF NOT EXISTS mydb;

use mydb;

DROP TABLE IF EXISTS foodratingspart;

CREATE TABLE IF NOT EXISTS mydb.foodratingspart (

food1 INT,

food2 INT,

food3 INT,

food4 INT,

id INT)

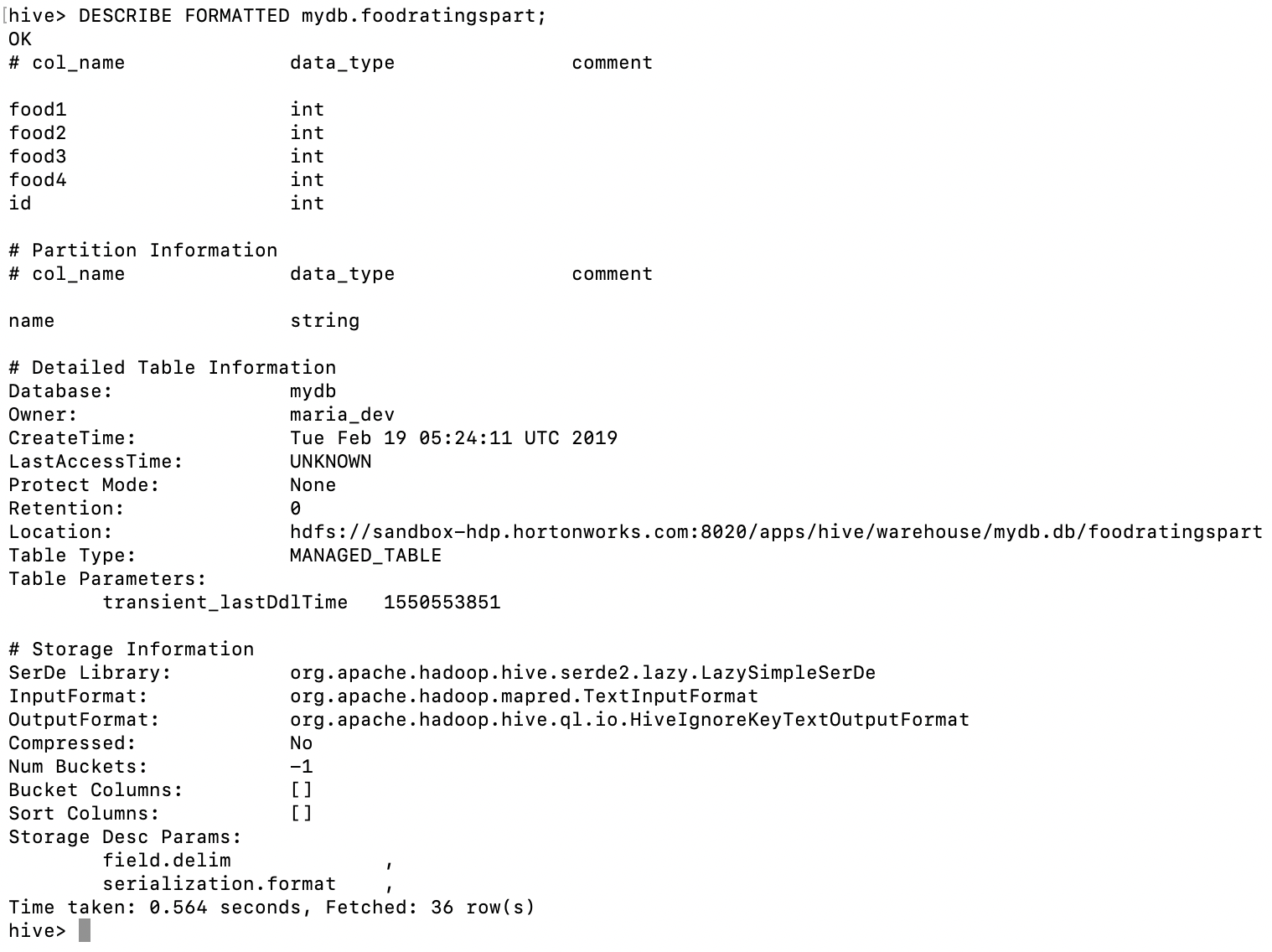
PARTITIONED BY (name STRING)

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.

DESCRIBE FORMATTED mydb.foodratingspart;



**Exercise 5:**

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

SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode=non-strict

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

INSERT OVERWRITE TABLE foodratingspart

PARTITION (name)

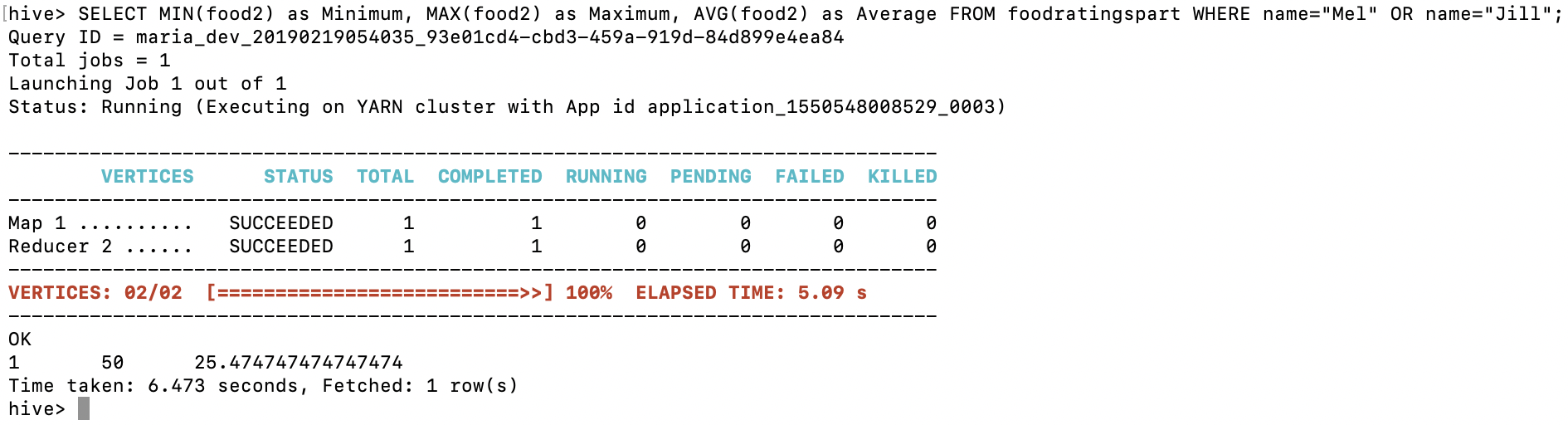
SELECT food1, food2, food3, food4, id, name

FROM 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.

SELECT MIN(food2) as Minimum, MAX(food2) as Maximum, AVG(food2) as Average FROM foodratingspart WHERE name="Mel" OR name="Jill";

Output:



**Exercise 6:**

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

LOAD DATA LOCAL INPATH './foodplaces6168.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’.

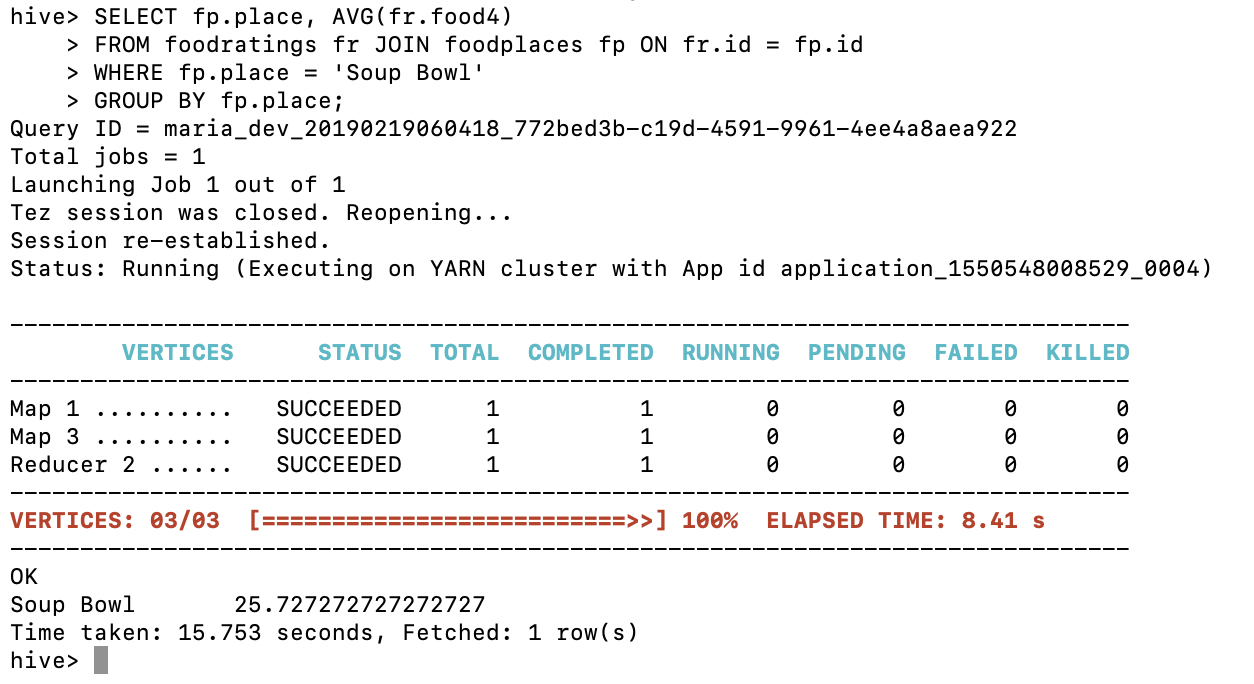
SELECT fp.place, AVG(fr.food4)

FROM foodratings fr JOIN foodplaces fp ON fr.id = fp.id

WHERE fp.place = 'Soup Bowl'

GROUP BY fp.place;

Output:



**Exercise 7:** (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

The authors described a new data processing environment being deployed at Yahoo! called Pig, and its associated language, Pig Latin. Pig's target demographic is experienced procedural programmers who prefer map-reduce style programming over the more declarative, SQL-style programming, for stylistic reasons as well as the ability to control the execution plan. Pig aims for a sweet spot between these two extremes, offering high-level data manipulation primitives such as projection and join, but in a much less declarative style than SQL, such as projection.

The author also described a novel debugging environment we are developing for Pig, called Pig Pen. In conjunction with the step-by-step nature of our Pig Latin language, Pig Pen makes it easy and fast for users to construct and debug their programs in an incremental fashion. The author says user can write a prefix of their overall program, examine the output on the sandbox data set, and iterate until the output matches what they intended.

While Pig Pen is still in early stages of development, the core Pig system is fully implemented and available as an open-source Apache incubator project. The Pig system compiles Pig Latin expressions into a sequence of map-reduce jobs, and orchestrates the execution of these jobs on Hadoop, an open-source scalable map-reduce implementation. Pig has an active and growing user base inside Yahoo!, and with their recent open-source release they are beginning to attract users in the broader community.