**CS 595 - Assignment 4**

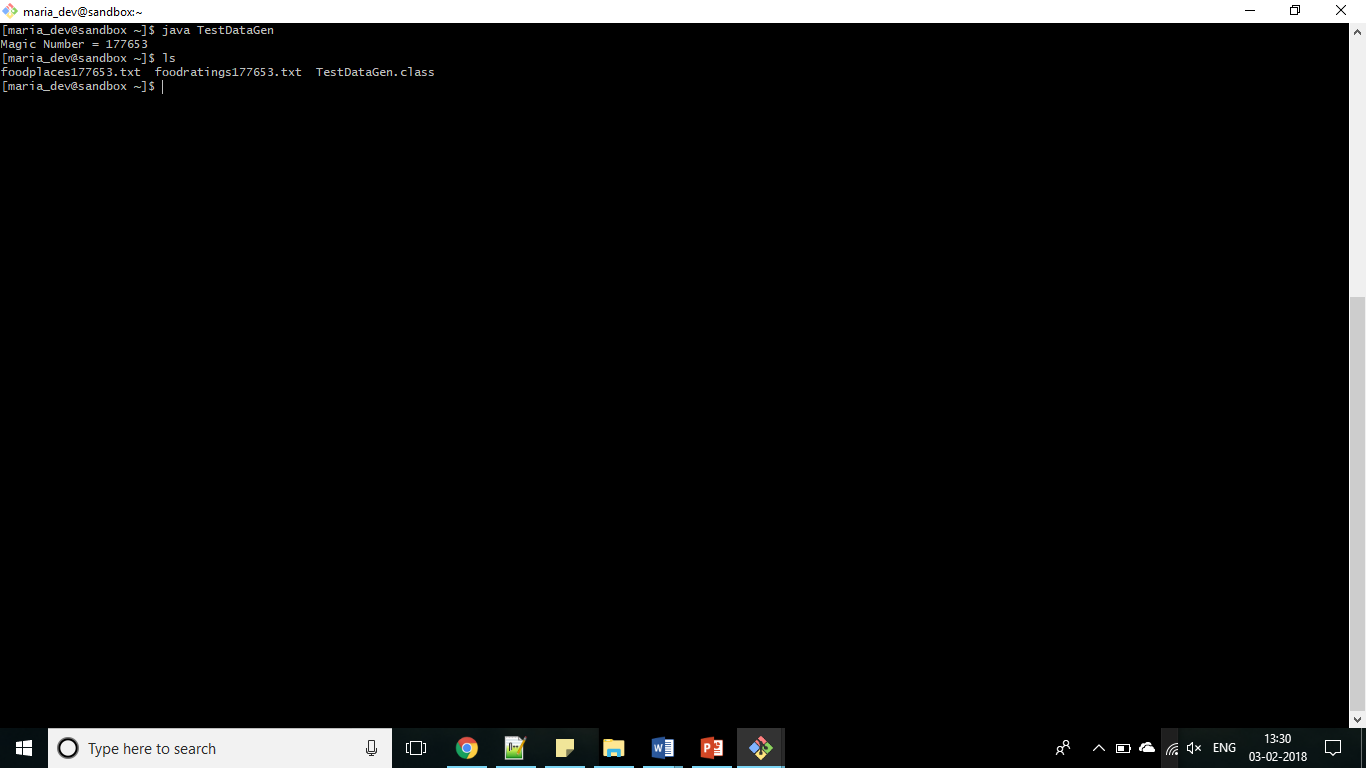
Name: Jagruti Vichare

Email ID: [jvichare@hawk.iit.edu](mailto:jvichare@hawk.iit.edu)

CWID: A20378092

Command: java TestDataGen

Output:



**Exercise 1**

Step 1:

CREATE DATABASE mydb;

Step 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')

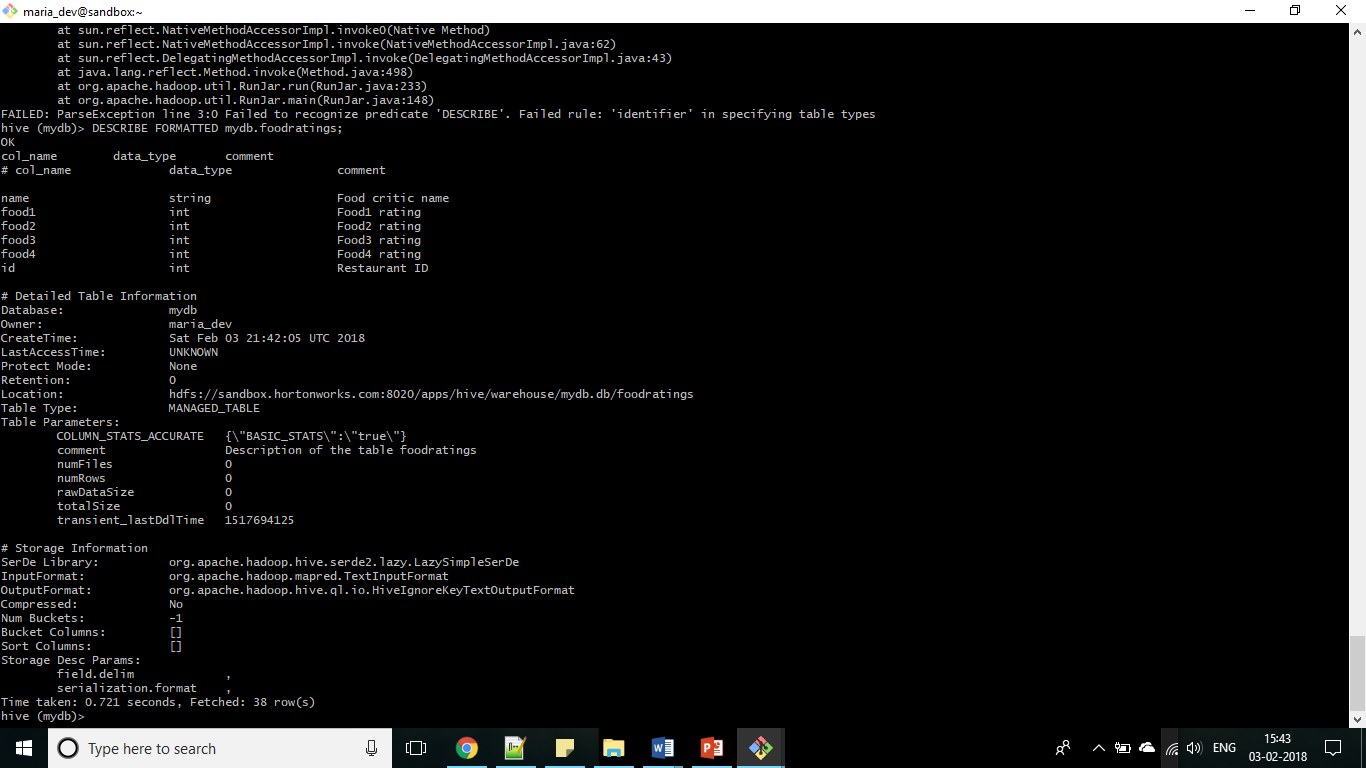
COMMENT 'Description of the table foodratings'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

Step 3:

DESCRIBE FORMATTED MyDb.foodratings

Output:

Step 4:

CREATE DATABASE IF NOT EXISTS mydb;

use mydb;

DROP TABLE IF EXISTS foodplaces;

CREATE TABLE IF NOT EXISTS mydb.foodplaces (

id INT,

place STRING)

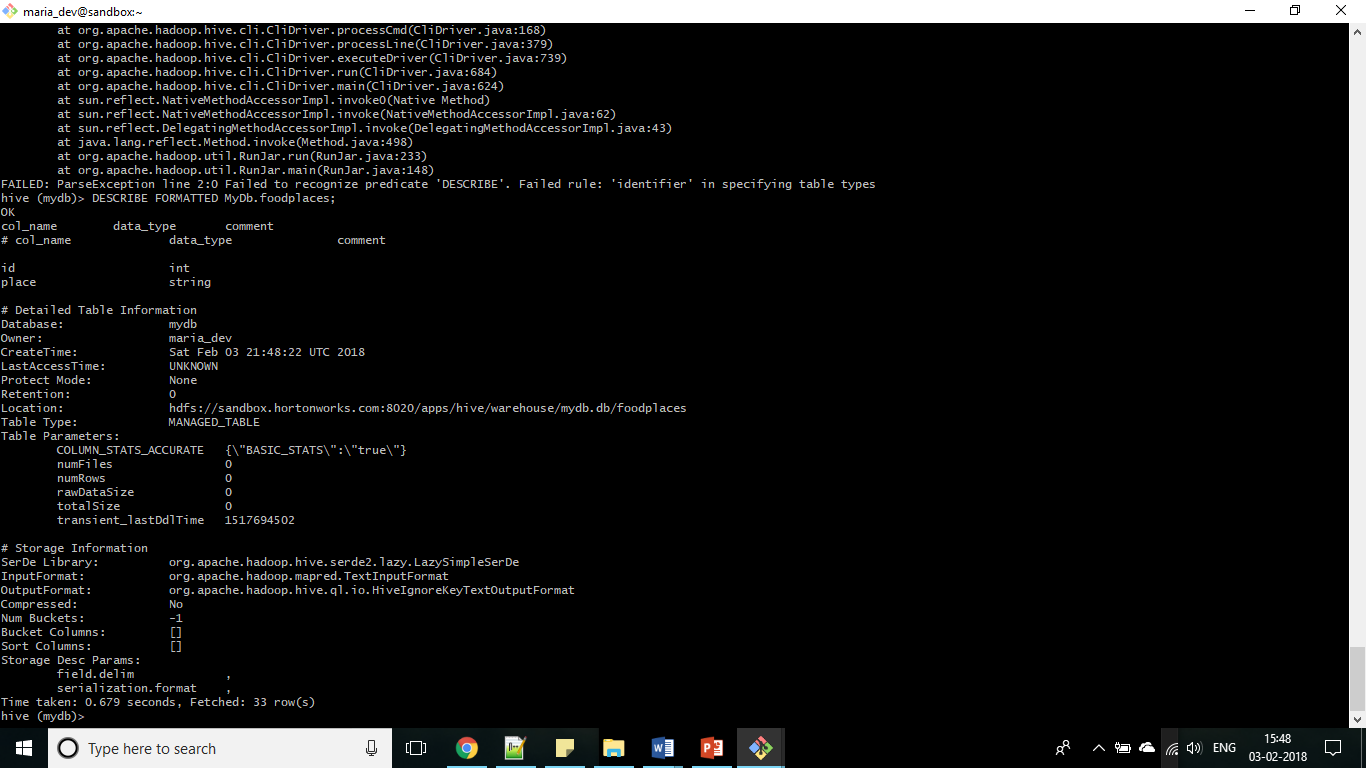
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

Step 5:

DESCRIBE FORMATTED MyDb.foodplaces

Output:



**Exercise 2**

Step 1:

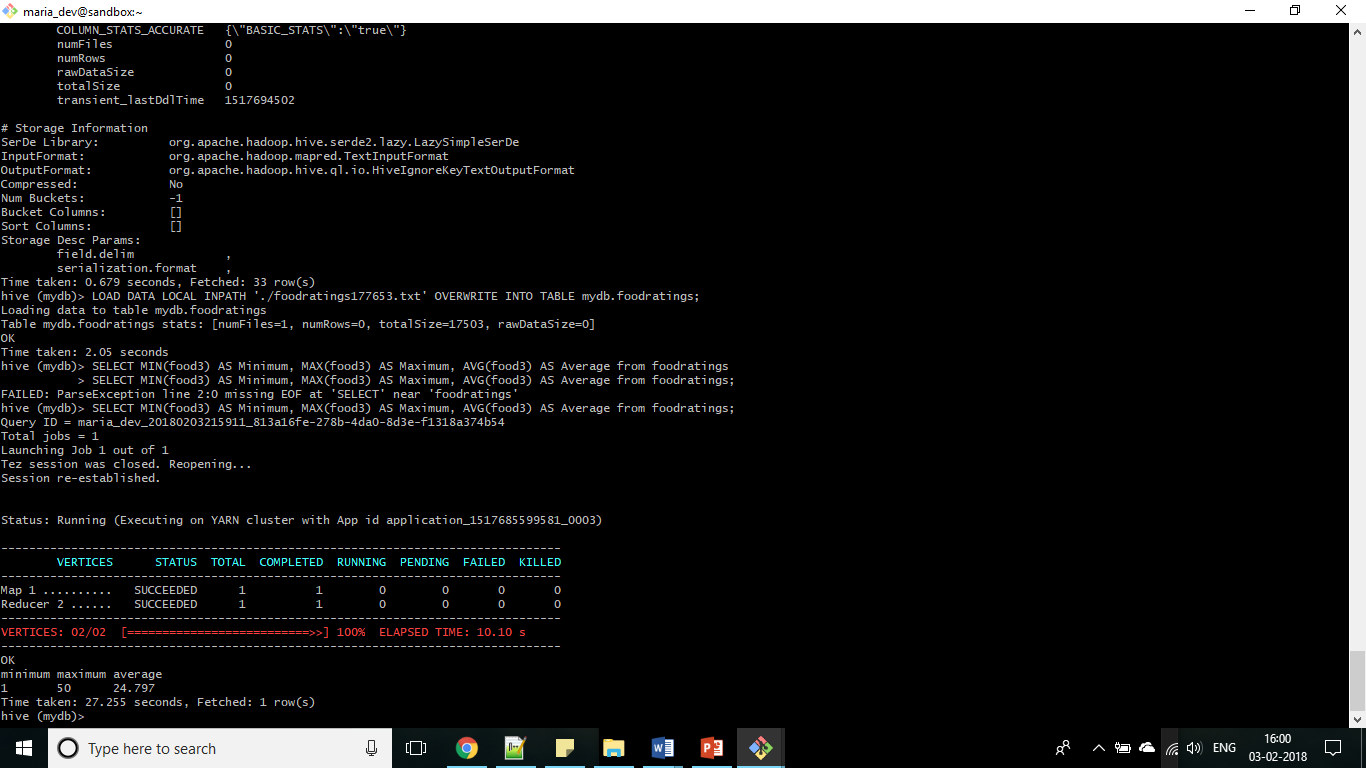
LOAD DATA LOCAL INPATH './foodratings177653.txt' OVERWRITE INTO TABLE mydb.foodratings;

Step 2:

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

Magic number - 177653

Output:



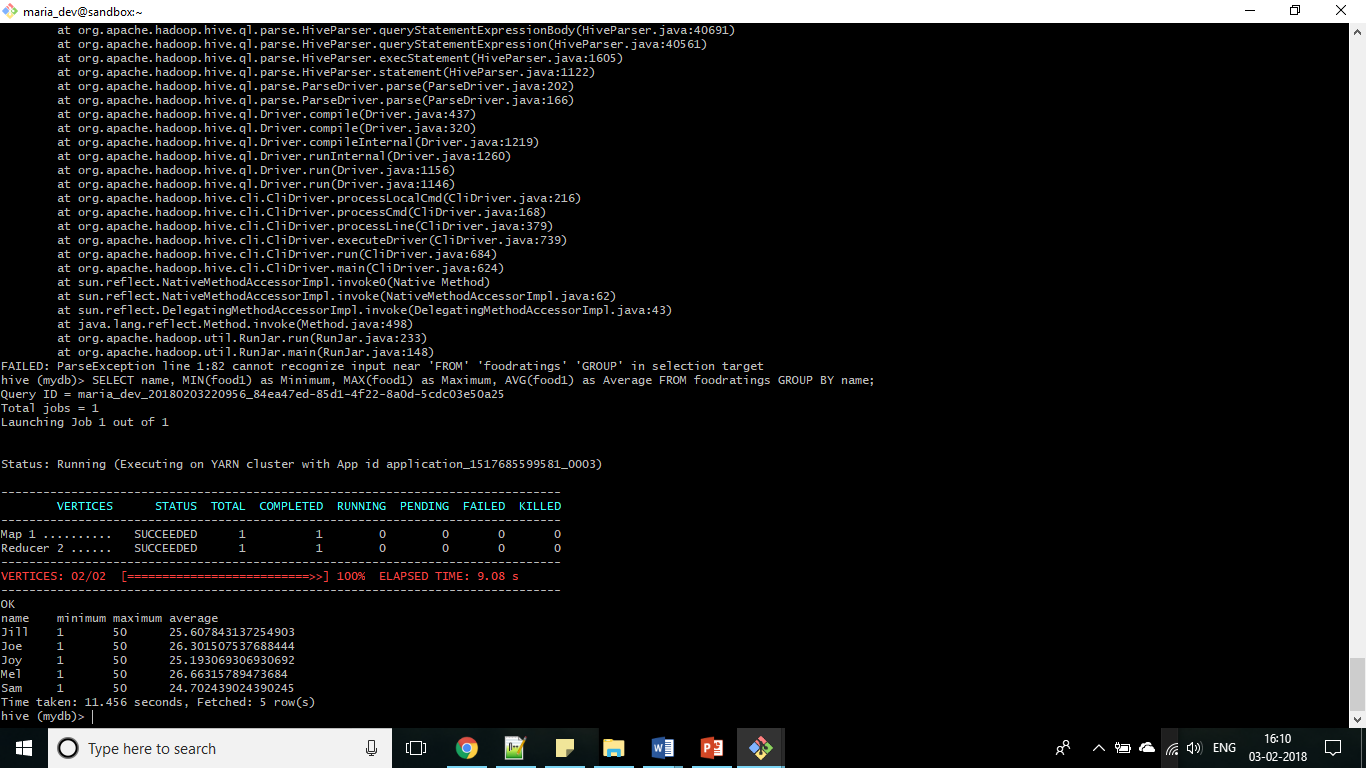
**Exercise 3**

Step 1:

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

Magic number - 177653

Output:



**Exercise 4**

Step 1:

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)

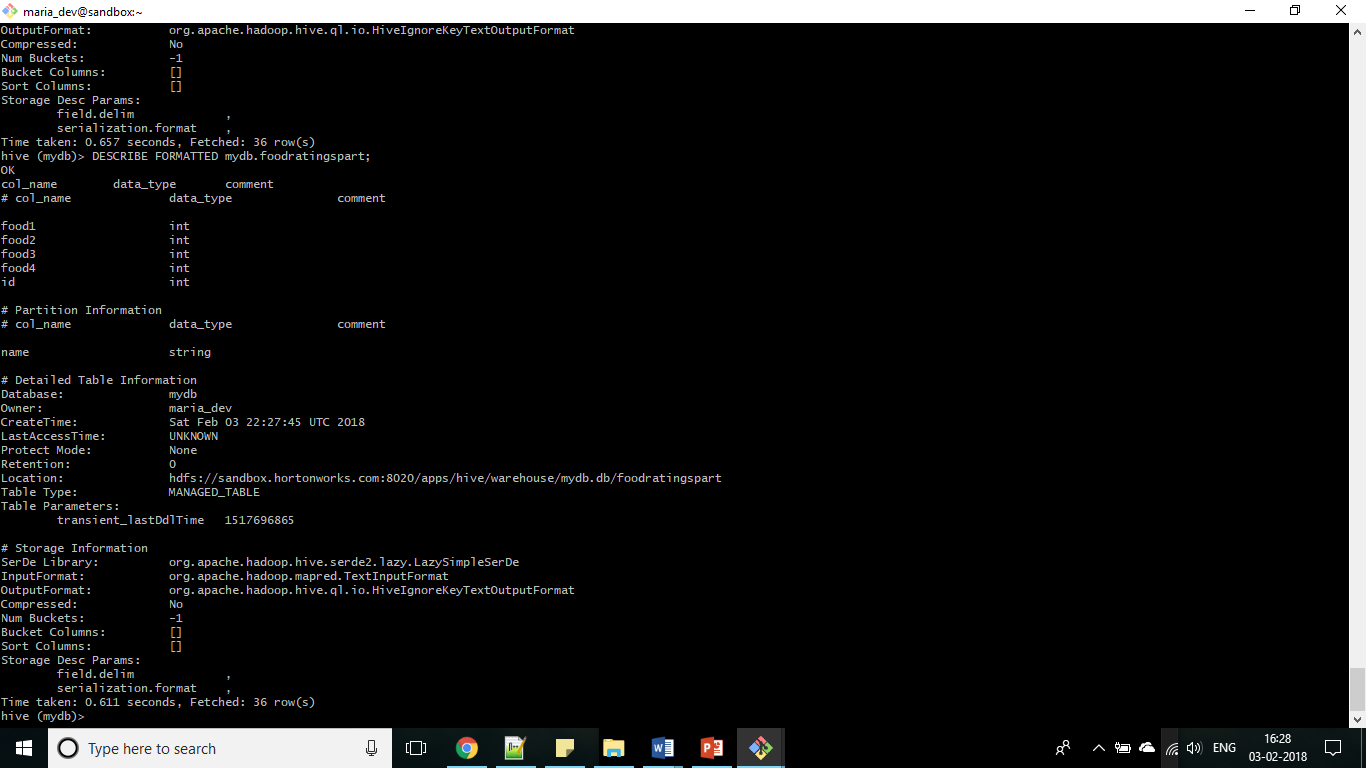
PARTITIONED BY (name STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

Step 2:

DESCRIBE FORMATTED mydb.foodratingspart

Output:

**Exercise 5**

Step 1:

SET hive.exec.dynamic.partition=true;

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

Step 2:

INSERT OVERWRITE TABLE foodratingspart

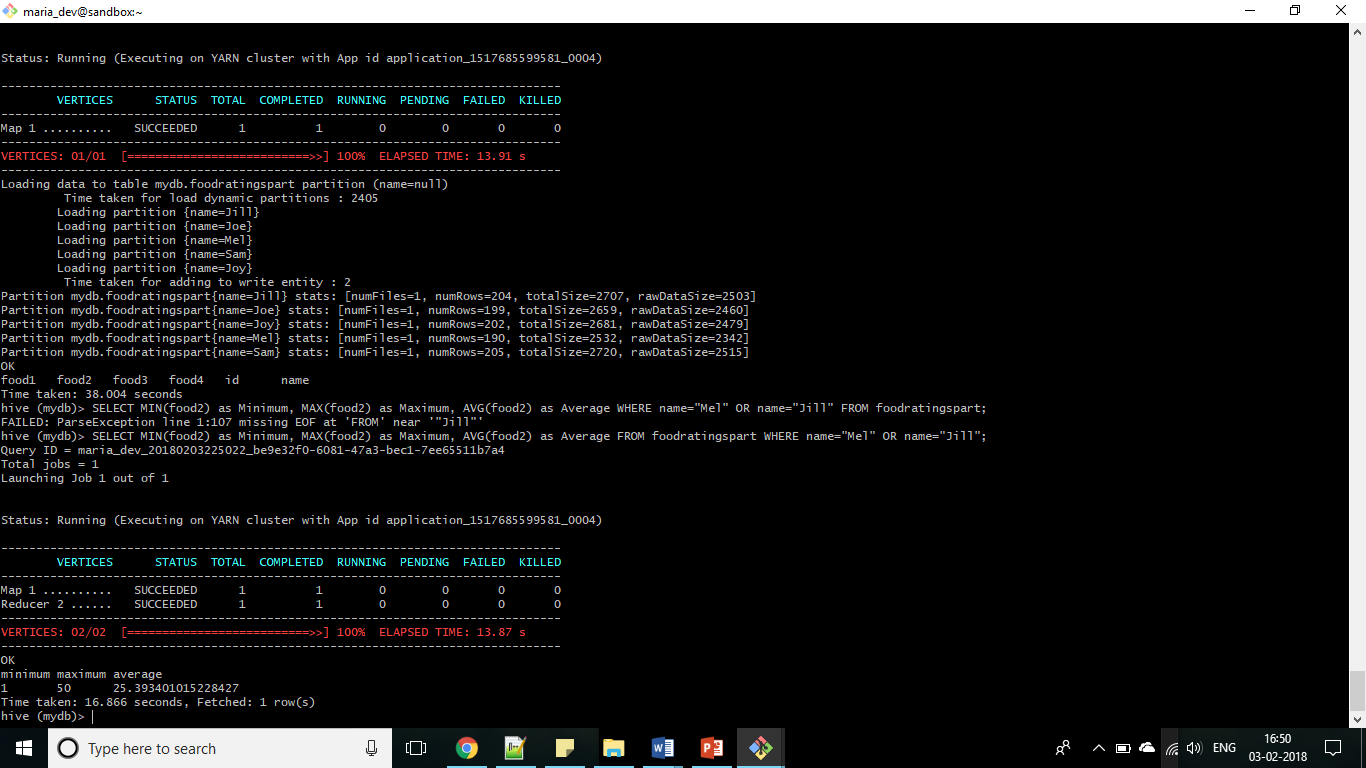
PARTITION (name)

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

FROM foodratings;

Step 3:

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

Output:

**Exercise 6**

Step 1:

LOAD DATA LOCAL INPATH './foodplaces177653.txt' OVERWRITE INTO TABLE mydb.foodplaces;

Step 2:

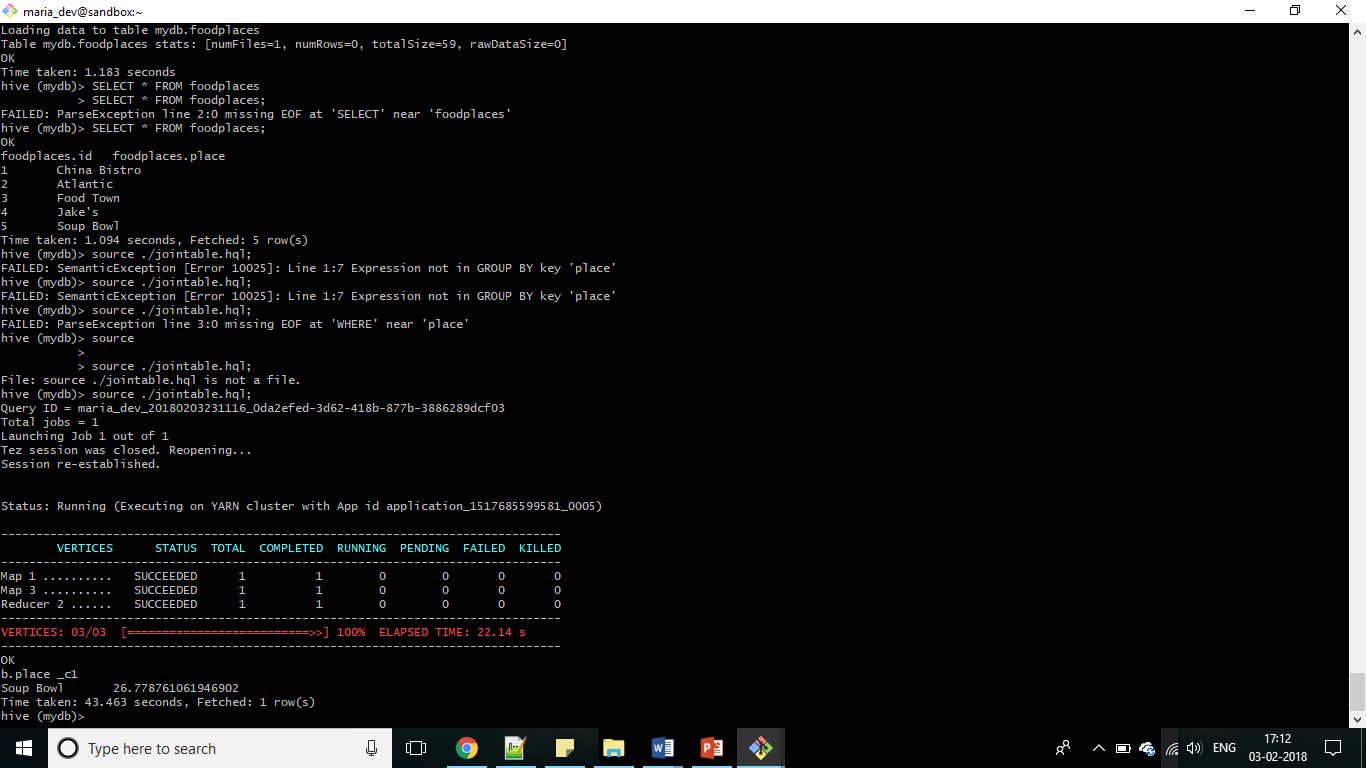
SELECT b.place, AVG(a.food4)

FROM foodratings a JOIN foodplaces b ON a.id = b.id

WHERE b.place = 'Soup Bowl'

GROUP BY b.place;

Output:



**Exercise 7**

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

Thus article describes a new language called Pig Latin deployed at Yahoo! which is a combination of declarative style of SQL and low level, procedural style of map-reduce. It is fully implemented and 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 integrated novel debugging environment that leads to even higher productivity. It is an open-source, Apache-incubator project, and available for general use.

This article also shows comparative analysis of SQL and Pig and concludes that in Pig, it is easier for programmers to understand and control how their data processing task is executed. It also supports a flexible, fully nested data model, user-defined functions, and the ability to operate over plain input files without any schema information. It allows complex, non-atomic data types such as set, map, and tuple to occur as fields of a table.

Pig is meant for offline, ad-hoc, scan-centric workloads. This article also compares Pig against other data processing languages and systems and about its scope of improvement.