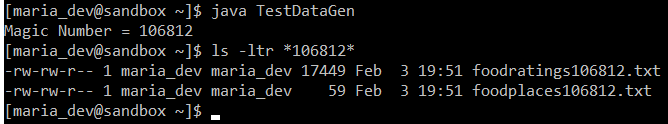
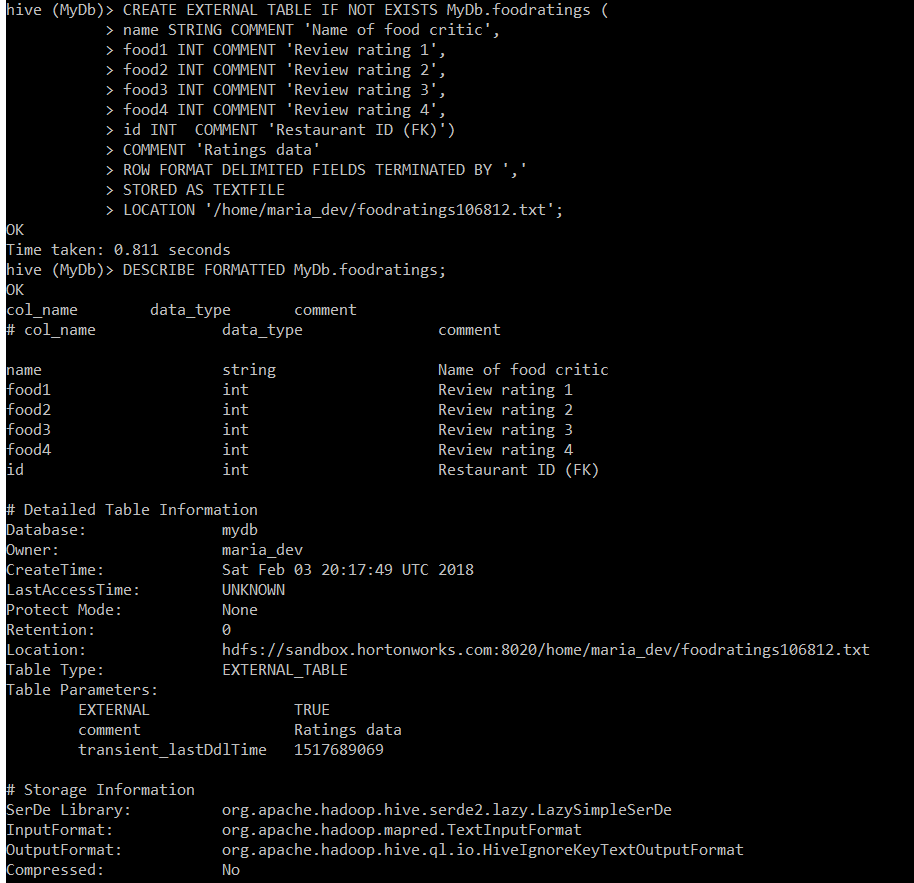
**CS595 - Assignment 4**



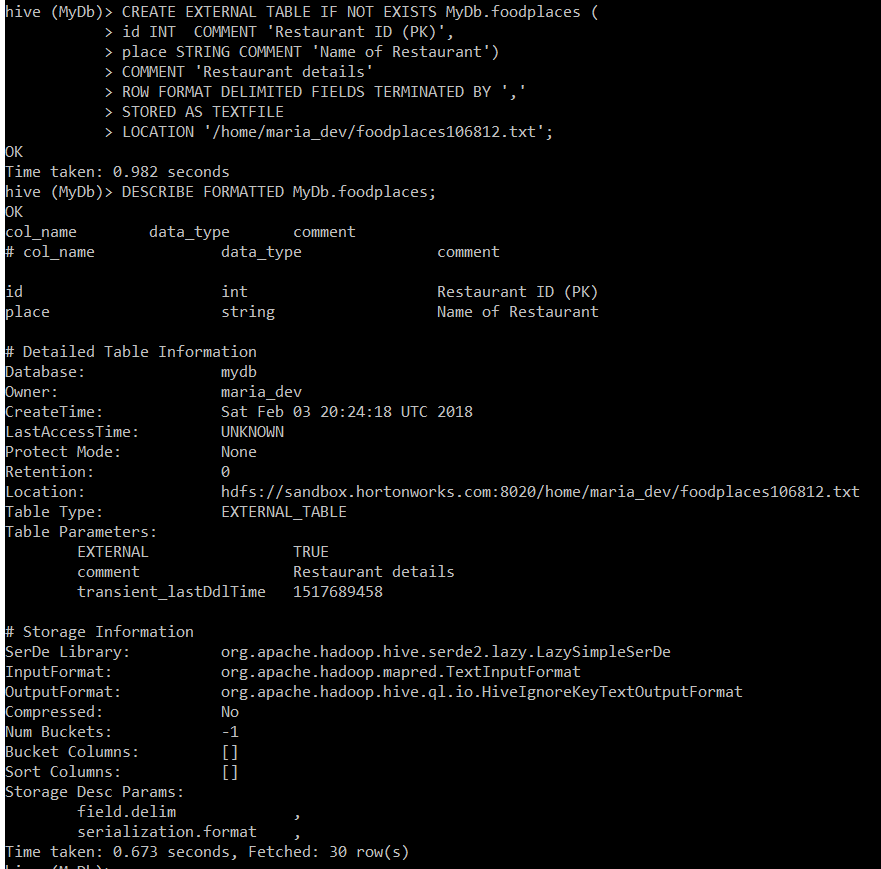
Magic Number = 106812

1. Create a Hive database called MyDb.

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



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



Command Executed:

CREATE DATABASE IF NOT EXISTS MyDb COMMENT 'Sameer Gadne - CS 595 DATABASE';

use MyDb;

CREATE EXTERNAL TABLE IF NOT EXISTS MyDb.foodratings (

name STRING COMMENT 'Name of food critic',

food1 INT COMMENT 'Review rating 1',

food2 INT COMMENT 'Review rating 2',

food3 INT COMMENT 'Review rating 3',

food4 INT COMMENT 'Review rating 4',

id INT COMMENT 'Restaurant ID (FK)')

COMMENT 'Ratings data'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/home/maria\_dev/foodratings106812.txt';

DESCRIBE FORMATTED MyDb.foodratings;

CREATE EXTERNAL TABLE IF NOT EXISTS MyDb.foodplaces (

id INT COMMENT 'Restaurant ID (PK)',

place STRING COMMENT 'Name of Restaurant')

COMMENT 'Restaurant details'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE

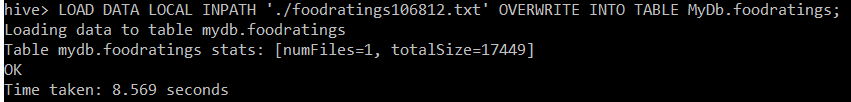
LOCATION '/home/maria\_dev/foodplaces106812.txt';

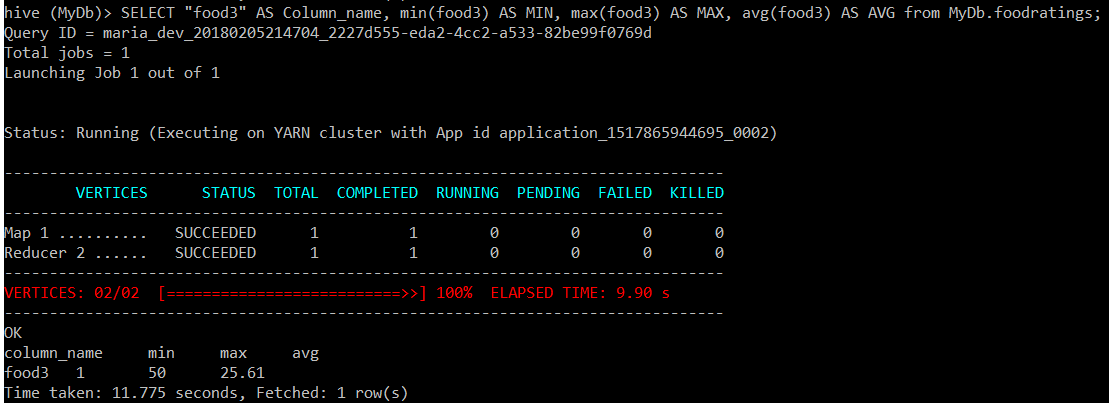
DESCRIBE FORMATTED MyDb.foodplaces;

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

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.





Command Executed:

Magic Number = 106812

LOAD DATA LOCAL INPATH './foodratings106812.txt' OVERWRITE INTO TABLE MyDb.foodratings;

SELECT "food3" AS Column\_name, min(food3) AS MIN, max(food3) AS MAX, avg(food3) AS AVG from MyDb.foodratings;

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

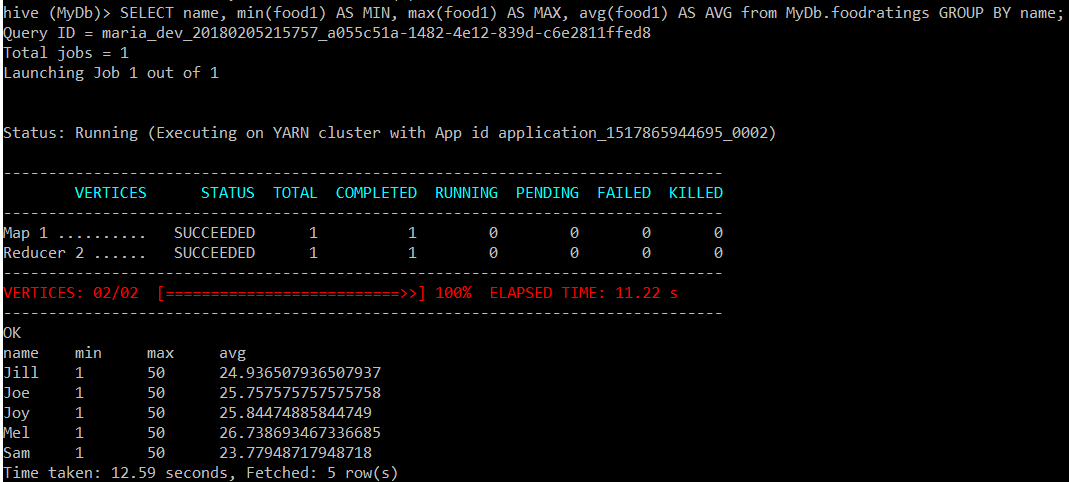
The output should look something like:

Mel 10 20 15

Bill 20, 30, 24

…

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



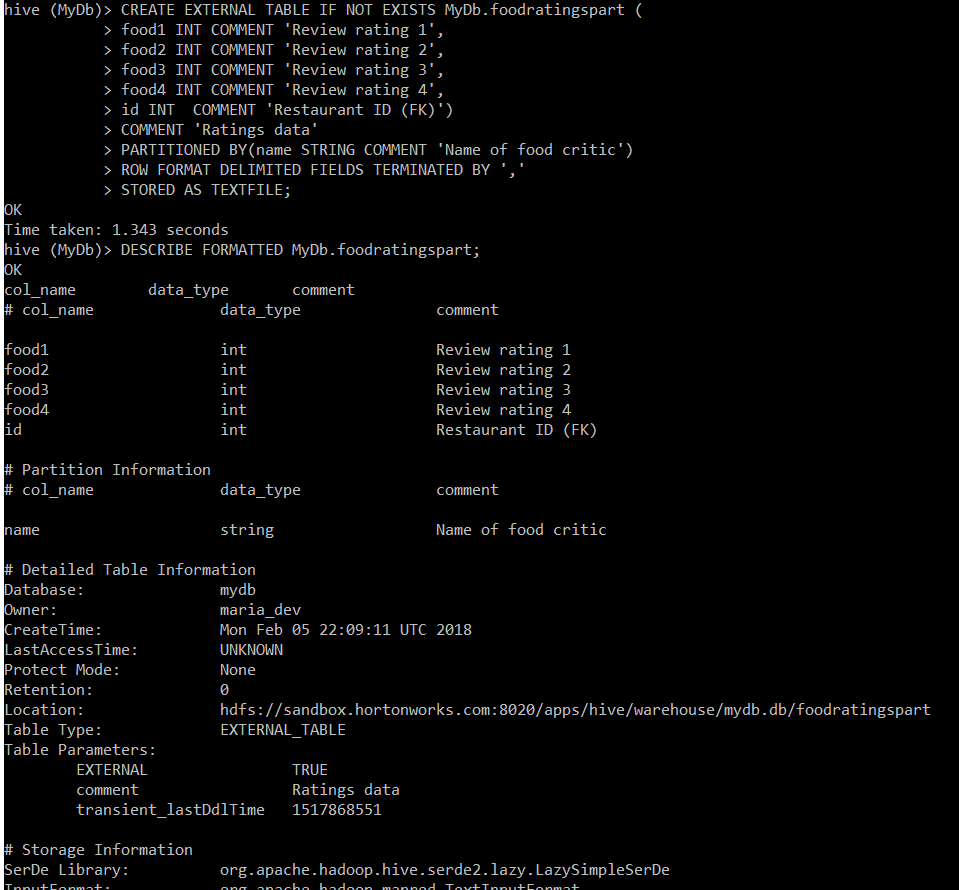
Command Executed:

Magic Number = 106812

SELECT name, min(food1) AS MIN, max(food1) AS MAX, avg(food1) AS AVG from MyDb.foodratings GROUP BY name;

1. In MyDb create a partitioned table called ‘foodratingspart’

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



Command Executed:

CREATE EXTERNAL TABLE IF NOT EXISTS MyDb.foodratingspart (

food1 INT COMMENT 'Review rating 1',

food2 INT COMMENT 'Review rating 2',

food3 INT COMMENT 'Review rating 3',

food4 INT COMMENT 'Review rating 4',

id INT COMMENT 'Restaurant ID (FK)')

COMMENT 'Ratings data'

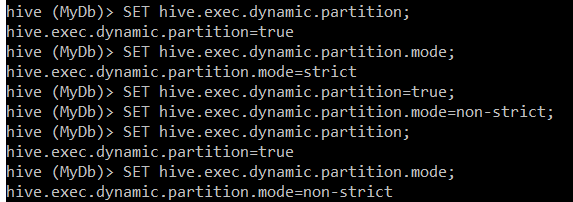
PARTITIONED BY(name STRING COMMENT 'Name of food critic')

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

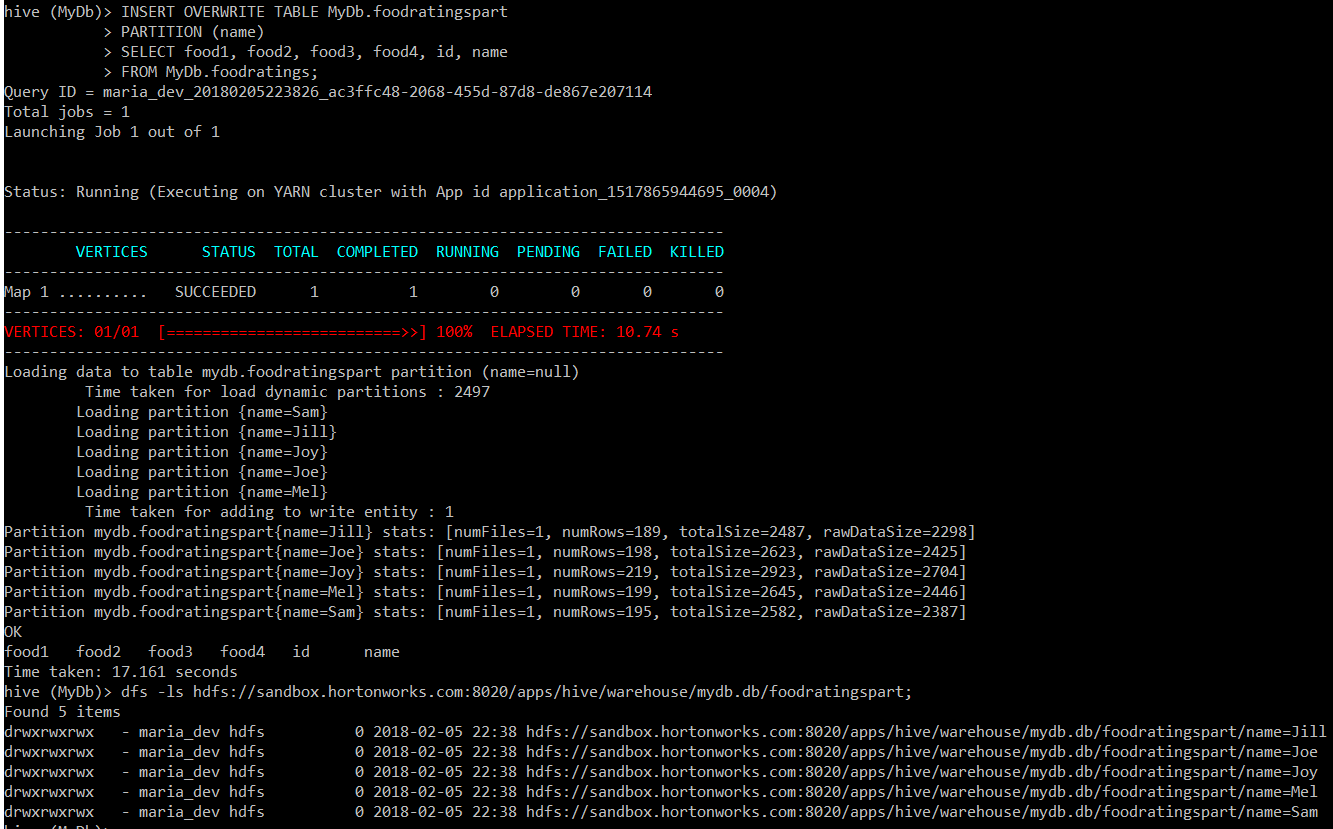
STORED AS TEXTFILE;

DESCRIBE FORMATTED MyDb.foodratingspart;

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



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.

Command Executed:

SET hive.exec.dynamic.partition=true;

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

INSERT OVERWRITE TABLE MyDb.foodratingspart

PARTITION (name)

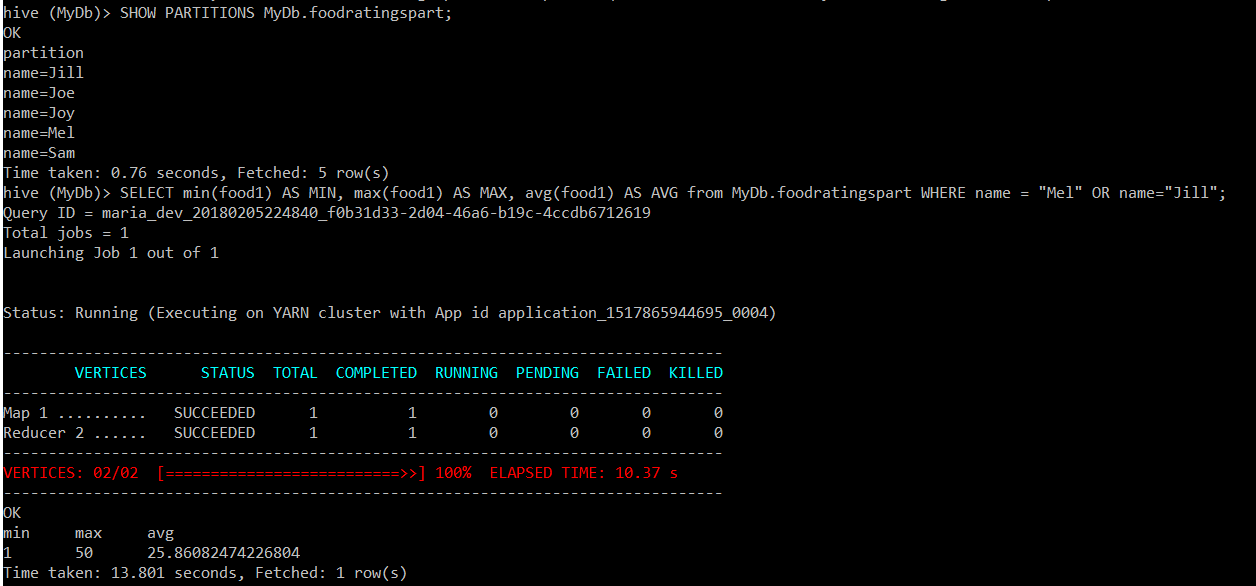
SELECT food1, food2, food3, food4, id, name

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. It should look something like

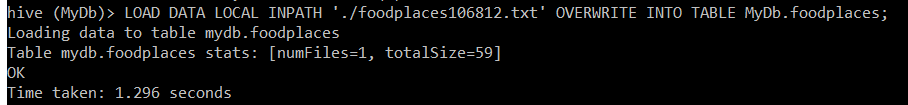
10 20 15



Command Executed:

SELECT min(food1) AS MIN, max(food1) AS MAX, avg(food1) AS AVG from MyDb.foodratingspart WHERE name = "Mel" OR name="Jill";

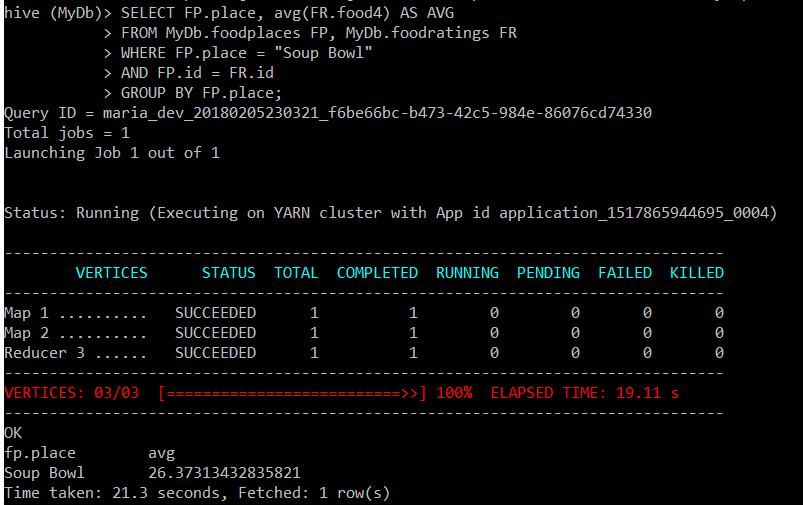
1. Load the foodplaces<.magic number>.txt file created using TestDataGen from your local file system into the foodplaces table.



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. It should look something like

Soup Bowl 20



Command Executed:

LOAD DATA LOCAL INPATH './foodplaces106812.txt' OVERWRITE INTO TABLE MyDb.foodplaces;

SELECT FP.place, avg(FR.food4) AS AVG

FROM MyDb.foodplaces FP, MyDb.foodratings FR

WHERE FP.place = "Soup Bowl"

AND FP.id = FR.id

GROUP BY FP.place;

SELECT FP.place, AVG(FR.food4) AS AVG, SUM(FR.food4) AS SUM, COUNT(FR.food4) AS COUNT

FROM MyDb.foodplaces FP, MyDb.foodratings FR

WHERE FP.place = "Soup Bowl"

AND FP.id = FR.id

GROUP BY FP.place;

***Extra Credit:***

1. 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:** Following is the summary of “Pig Latin: A Not-So-Foreign Language for Data Processing” article:

* The article introduces a new language called Pig Latin that was designed to fit in a sweet spot between the declarative style of SQL, and the low-level, procedural style of map-reduce. The article also provides us with details of a novel debugging environment Pig Pen that allows users to write & debug Pig scripts/programs. Pig is an open-source, Apache-incubator project, and available for general use.
* The accompanying system, Pig, is fully implemented, and compiles Pig Latin into physical plans that are executed over Hadoop, an open-source, map-reduce implementation. Pig can be used to dramatically reduce the time required for the development and execution of data analysis tasks, compared to using Hadoop (rigid Map-Reduce framework) directly. Pig also comes integrated with a novel debugging environment that can lead to even higher productivity gains.

**Business case for Pig:**

* At a growing number of organizations (primarily Internet companies such as Amazon, Google, Microsoft, and Yahoo!), innovation revolves around the collection and analysis of enormous data sets such as web crawls, search logs, and click streams.
* The sheer size of these data sets dictates that it be stored and processed on highly parallel systems, such as shared-nothing clusters. Parallel database products, e.g., Teradata, Oracle RAC, Netezza, offer a solution by providing a simple SQL query interface and hiding the complexity of the physical cluster. These products however, can be prohibitively expensive at web scale. Besides, they wrench programmers away from their preferred method of analyzing data, namely writing imperative scripts or code, toward writing declarative queries in SQL, which they often find unnatural, and overly restrictive.
* As evidence of the above, programmers have been flocking to the more procedural map-reduce programming model. A map-reduce program essentially performs a group-by-aggregation in parallel over a cluster of machines. The Map-Reduce programming model also allows us to take advantage of its associated scalable implementations on commodity hardware, thereby allowing the ability to deal with enormous datasets with (Structured, Unstructured & Semi-structured) data formats in a cost-effective manner.
* Pig's targeted 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 at achieving a middle ground between two extremes, by offering high-level data manipulation primitives such as projection and join, but in a much less declarative style than SQL.

**Conclusion:**

* Pig was developed to deal with ad-hoc analysis of very large datasets generated by IoT in a cost & computationally effective manner as compared to the expensive and more restrictive Parallel database management systems. The language Pig Latin was developed on top of Pig to take advantage of map-reduce programming model and simultaneously allowing the simplicity of a declarative query language (to internally implement the functionality in map-reduce paradigm) and to provide programmers with more fine-grained control over the step-by-step execution of query as opposed to simple declarative SQL.