Shraddha Patel A2049171

CSP554—Big Data Technologies

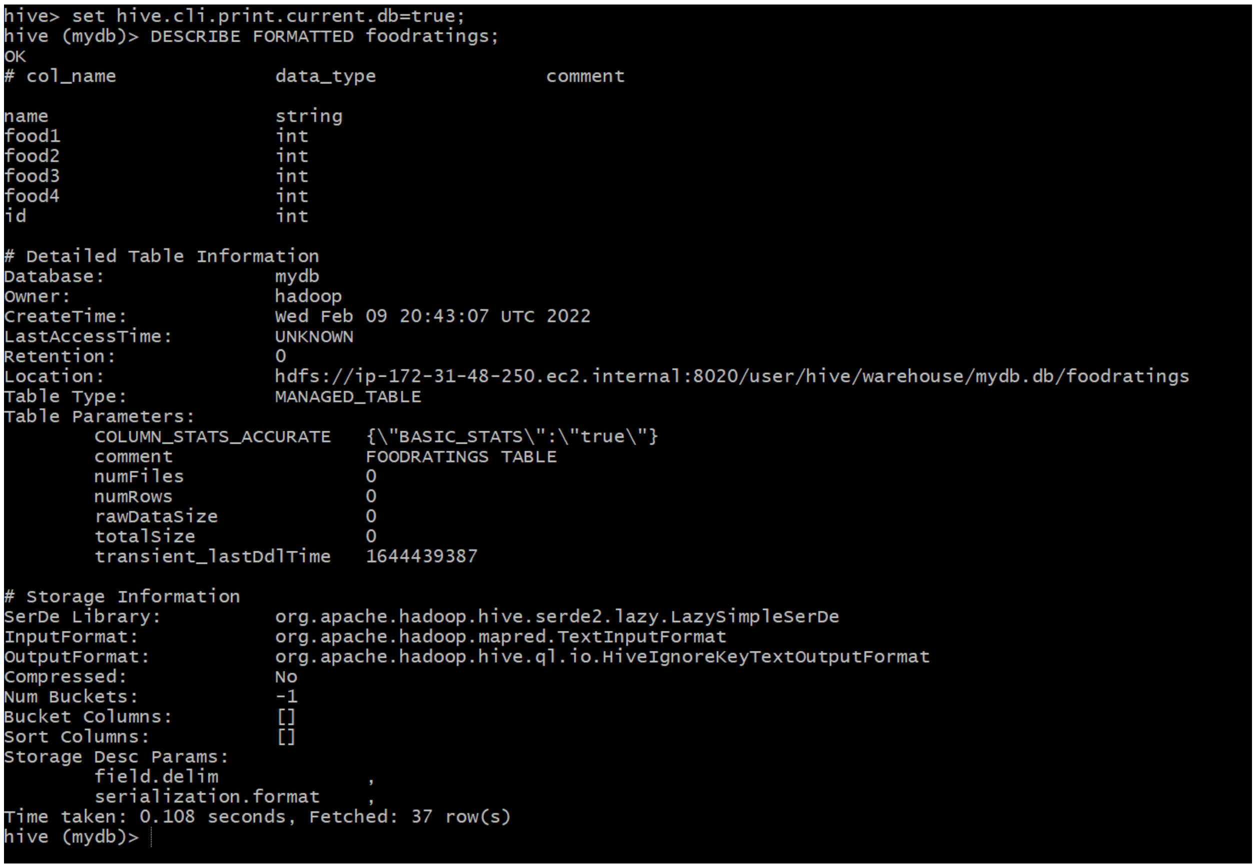
Assignment #4

Exercise 1) 2 points

```
create database MyDb

Use myDb;

set hive.cli.print.current.db=true;

CREATE TABLE IF NOT EXISTS mydb.foodratings (

name STRING, food1 INTEGER, food2 INTEGER, food3 INTEGER, food4 INTEGER, id INTEGER)

COMMENT 'FOODRATINGS TABLE'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE
```

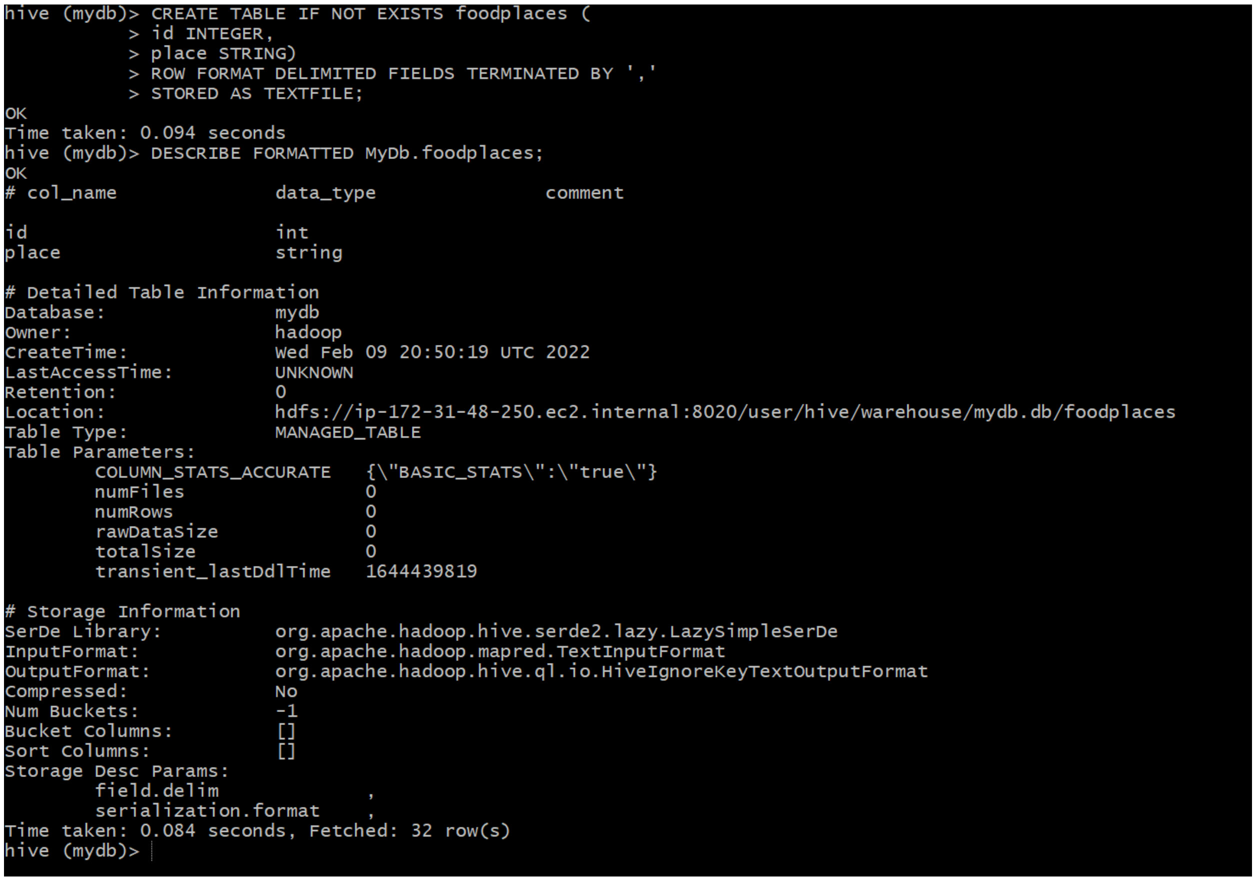
Command Execution:

```
'DESCRIBE FORMATTED MyDb.foodratings;
```



```
hive> set hive.cli.print.current.db=true;
hive (mydb)> DESCRIBE FORMATTED foodratings;
OK
# col_name              data_type               comment

name                    string
food1                   int
food2                   int
food3                   int
food4                   int
id                      int

# Detailed Table Information
Database:               mydb
Owner:                  hadoop
CreateTime:             Wed Feb 09 20:43:07 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-48-250.ec2.internal:8020/user/hive/warehouse/mydb.db/foodratings
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        comment                 FOODRATINGS TABLE
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1644439387

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.108 seconds, Fetched: 37 row(s)
hive (mydb)>
```

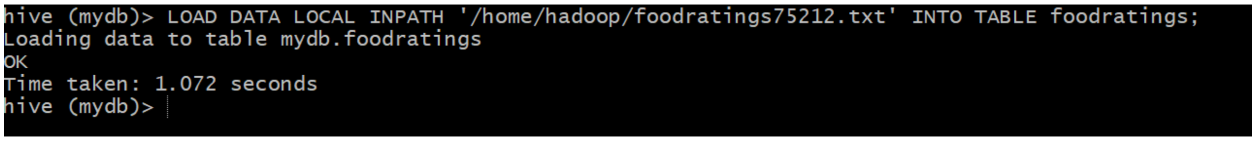
Command Execution:



```
hive (mydb)> CREATE TABLE IF NOT EXISTS foodplaces (
           > id INTEGER,
           > place STRING)
           > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
           > STORED AS TEXTFILE;
OK
Time taken: 0.094 seconds
hive (mydb)> DESCRIBE FORMATTED MyDb.foodplaces;
OK
# col_name              data_type               comment

id                      int
place                   string

# Detailed Table Information
Database:               mydb
Owner:                  hadoop
CreateTime:             Wed Feb 09 20:50:19 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-48-250.ec2.internal:8020/user/hive/warehouse/mydb.db/foodplaces
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1644439819

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.084 seconds, Fetched: 32 row(s)
hive (mydb)>
```

Exercise 2) 2 points

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

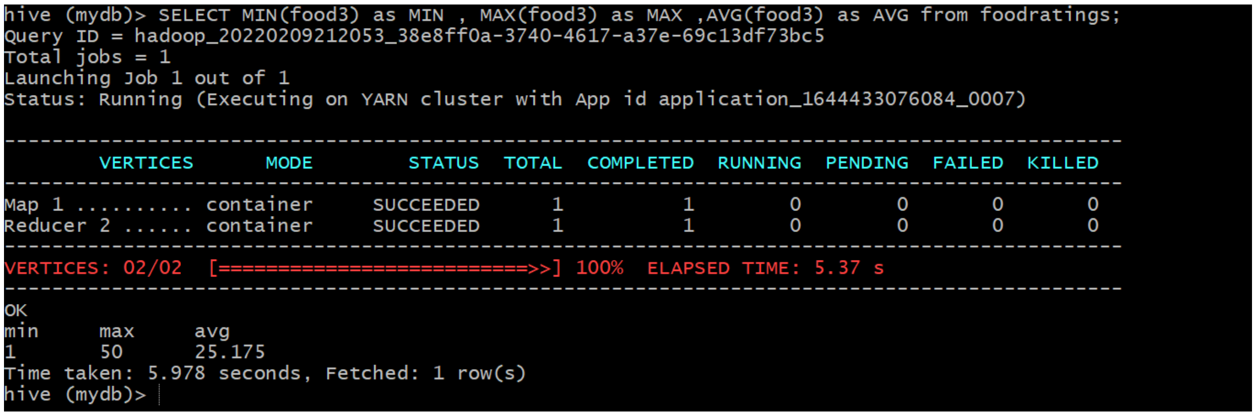


```
hive (mydb)> LOAD DATA LOCAL INPATH '/home/hadoop/foodratings75212.txt' INTO TABLE foodratings;
Loading data to table mydb.foodratings
OK
Time taken: 1.072 seconds
hive (mydb)>
```

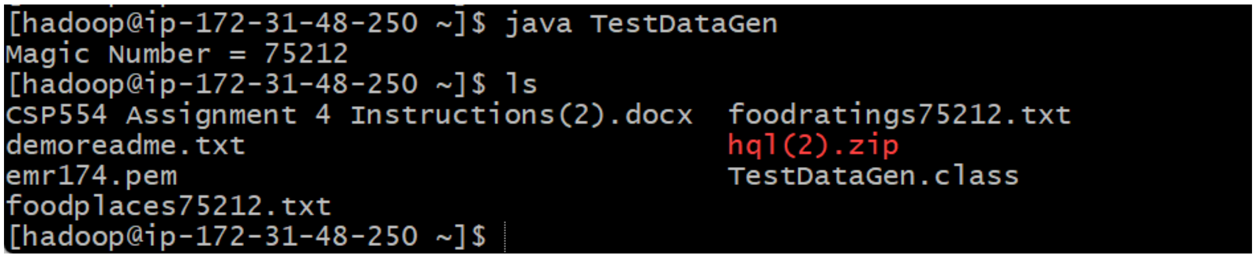
Execute a hive command to output the min, max and average of the values of the food3 column of the foodratings table. This should be one hive command, not three separate ones.

SELECT MIN (food3) as MIN , MAX(food3) as MAX ,AVG(food3) as AVG from foodratings;

Magic Number: 75212



```
hive (mydb)> SELECT MIN(food3) as MIN , MAX(food3) as MAX ,AVG(food3) as AVG from foodratings;
Query ID = hadoop_20220209212053_38e8ff0a-3740-4617-a37e-69c13df73bc5
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644433076084_0007)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5.37 s
----------------------------------------------------------------------------------------------
OK
min     max     avg
1       50      25.175
Time taken: 5.978 seconds, Fetched: 1 row(s)
hive (mydb)>
```



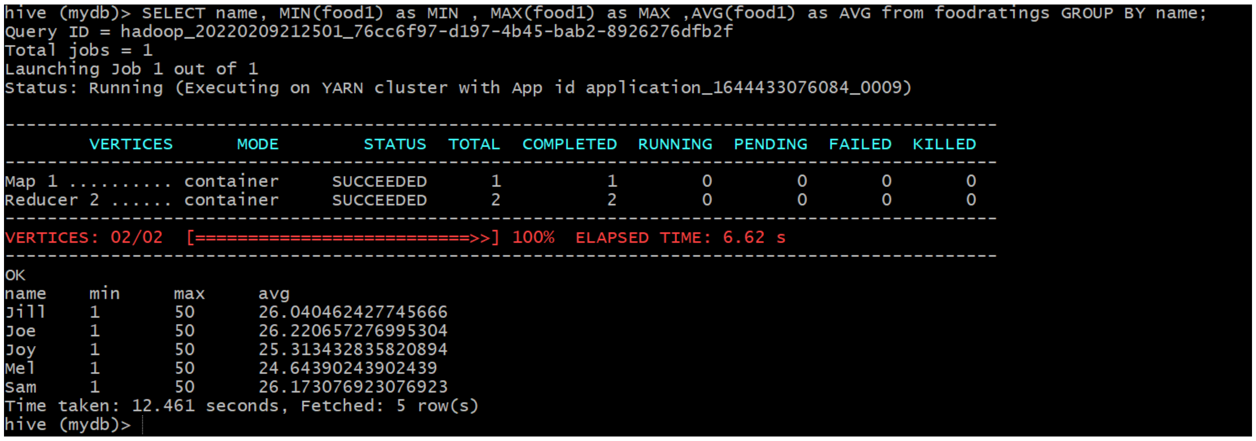
```
[hadoop@ip-172-31-48-250 ~]$ java TestDataGen
Magic Number = 75212
[hadoop@ip-172-31-48-250 ~]$ ls
CSP554 Assignment 4 Instructions(2).docx  foodratings75212.txt
demoreadme.txt                            hql(2).zip
emr174.pem                                TestDataGen.class
foodplaces75212.txt
[hadoop@ip-172-31-48-250 ~]$
```

Exercise 3) 2 points

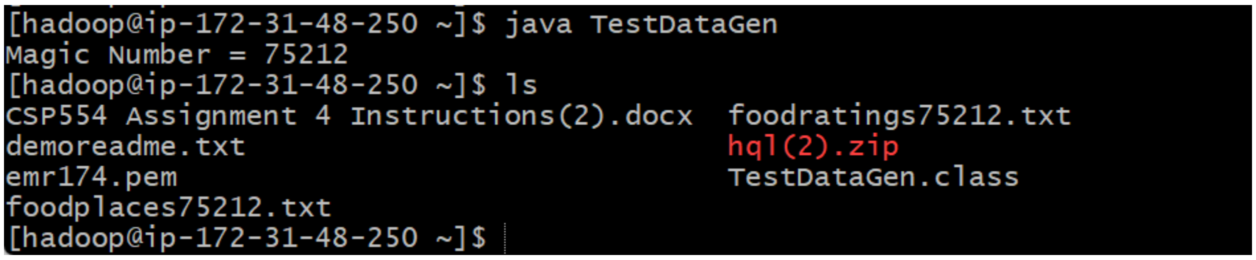
Execute a hive command to output the min, max and average of the values of the food1 column grouped by the first column 'name'. This should be one hive command, not three separate ones.

SELECT name, MIN (food1) as MIN , MAX(food1) as MAX ,AVG(food1) as AVG from foodratings GROUP BY name;

OUTPUT:



```
hive (mydb)> SELECT name, MIN(food1) as MIN , MAX(food1) as MAX ,AVG(food1) as AVG from foodratings GROUP BY name;
Query ID = hadoop_20220209212501_76cc6f97-d197-4b45-bab2-8926276dfb2f
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644433076084_0009)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 6.62 s
----------------------------------------------------------------------------------------------
OK
name    min     max     avg
Jill    1       50      26.040462427745666
Joe     1       50      26.220657276995304
Joy     1       50      25.313432835820894
Mel     1       50      24.64390243902439
Sam     1       50      26.173076923076923
Time taken: 12.461 seconds, Fetched: 5 row(s)
hive (mydb)>
```

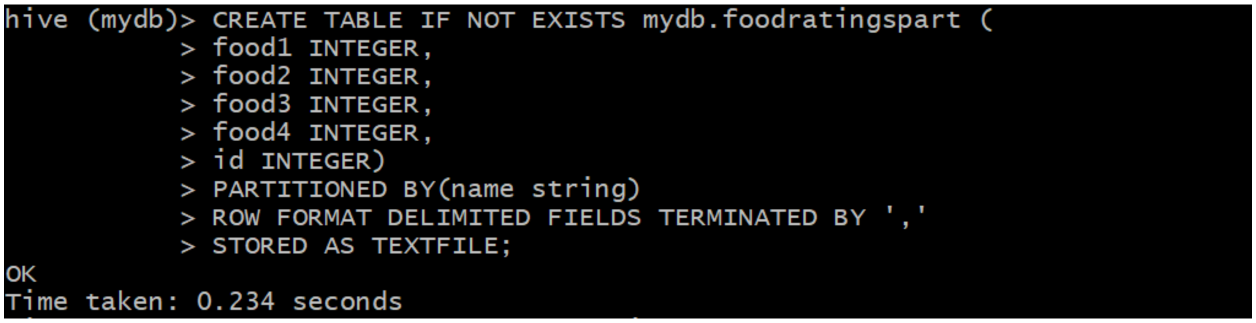


```
[hadoop@ip-172-31-48-250 ~]$ java TestDataGen
Magic Number = 75212
[hadoop@ip-172-31-48-250 ~]$ ls
CSP554 Assignment 4 Instructions(2).docx  foodratings75212.txt
demoreadme.txt                            hql(2).zip
emr174.pem                                TestDataGen.class
foodplaces75212.txt
[hadoop@ip-172-31-48-250 ~]$
```

Exercise 4) 2 points

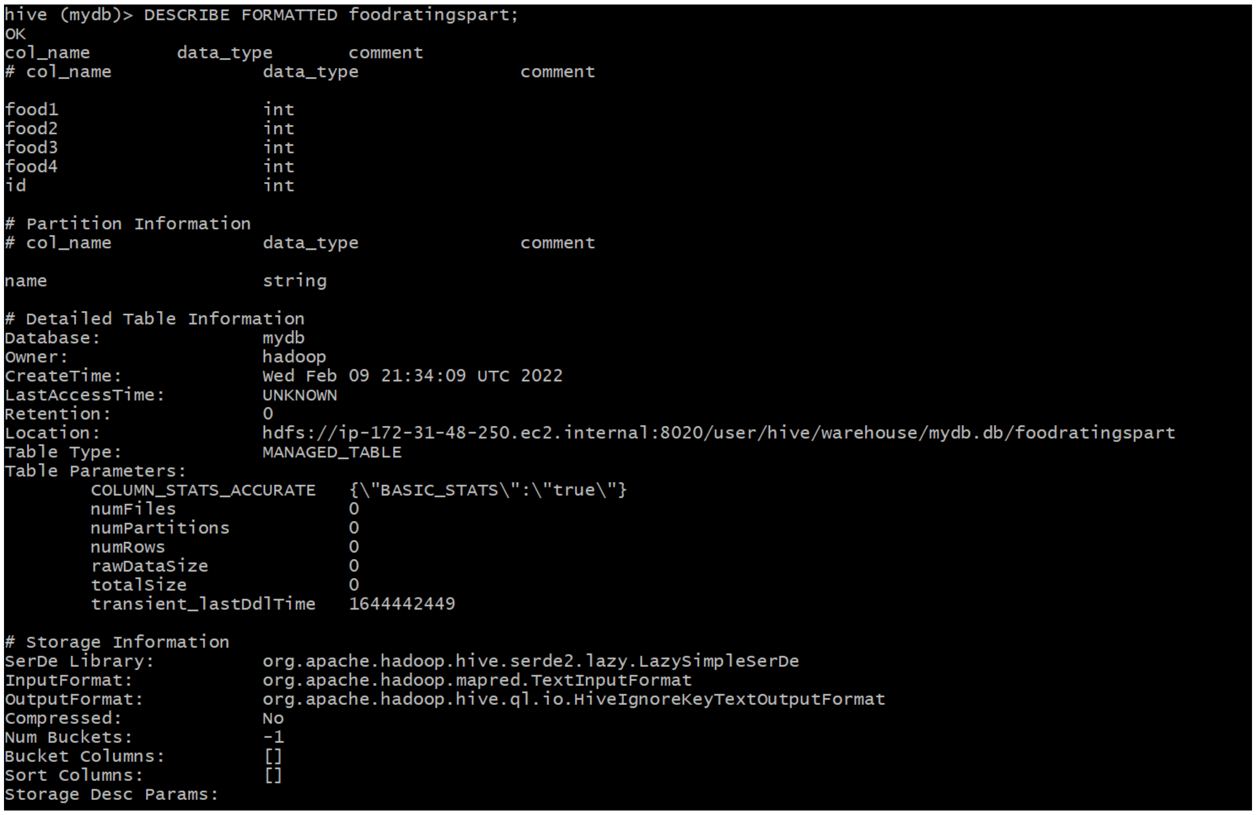
In MyDb create a partitioned table called ‘foodratingspart’

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.



```
hive (mydb)> CREATE TABLE IF NOT EXISTS mydb.foodratingspart (
           > food1 INTEGER,
           > food2 INTEGER,
           > food3 INTEGER,
           > food4 INTEGER,
           > id INTEGER)
           > PARTITIONED BY(name string)
           > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
           > STORED AS TEXTFILE;
OK
Time taken: 0.234 seconds
```

Command Execution:



```
hive (mydb)> DESCRIBE FORMATTED foodratingspart;
OK
col_name        data_type       comment
# col_name              data_type               comment

food1                   int
food2                   int
food3                   int
food4                   int
id                      int

# Partition Information
# col_name              data_type               comment

name                    string

# Detailed Table Information
Database:               mydb
Owner:                  hadoop
CreateTime:             Wed Feb 09 21:34:09 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-48-250.ec2.internal:8020/user/hive/warehouse/mydb.db/foodratingspart
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        numFiles                0
        numPartitions           0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1644442449

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
```

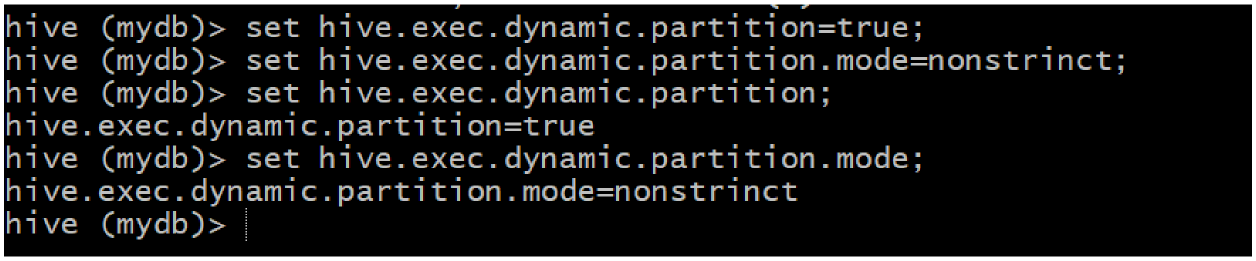
Exercise 5) 2 points

Assume that the number of food critics is relatively small, say less than 10 and the number places to eat is very large, say more than 10,000. In a few short sentences explain why using the (critic) name is a good choice for a partition field while using the place id is not.

Answer:

Here The cardinality for column place id is very high (10,000), we cannot use that column for partitioning as it would be end up having 10,000 partitions. we can partition by number of places column as data in that partition is less than 10 so analysis can be done on relevant dataset only and resulting in improved performance of HIVE queries.

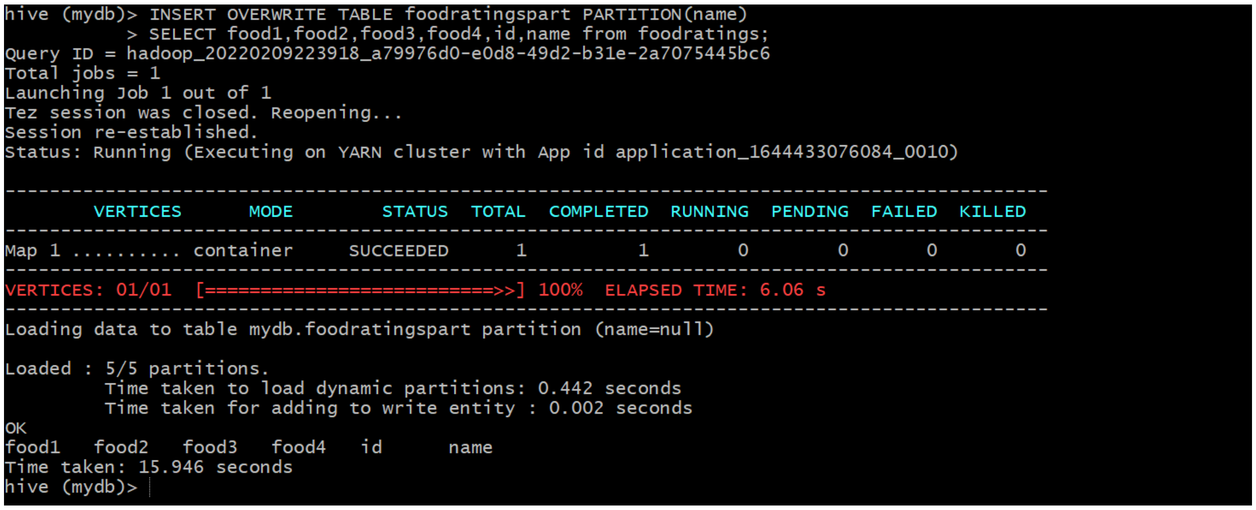
Exercise 6) 2 points



```
hive (mydb)> set hive.exec.dynamic.partition=true;
hive (mydb)> set hive.exec.dynamic.partition.mode=nonstrinct;
hive (mydb)> set hive.exec.dynamic.partition;
hive.exec.dynamic.partition=true
hive (mydb)> set hive.exec.dynamic.partition.mode;
hive.exec.dynamic.partition.mode=nonstrinct
hive (mydb)>
```

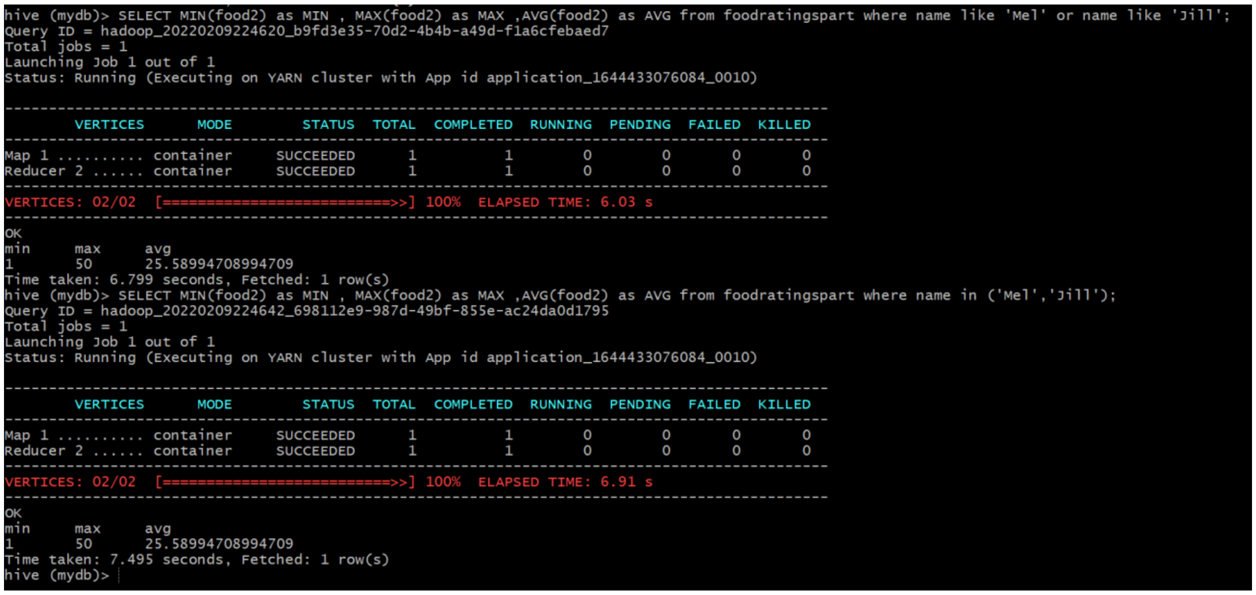
Provide a copy of the command you use to load the ‘foodratings’ table as a result of this exercise.

INSERT OVERWRITE TABLE foodratings PARTITION (name) SELECT food1,food2,food3,food4,id,name from foodratings;



```
hive (mydb)> INSERT OVERWRITE TABLE foodratingspart PARTITION(name)
           > SELECT food1,food2,food3,food4,id,name from foodratings;
Query ID = hadoop_20220209223918_a79976d0-e0d8-49d2-b31e-2a7075445bc6
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1644433076084_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 6.06 s
----------------------------------------------------------------------------------------------
Loading data to table mydb.foodratingspart partition (name=null)

Loaded : 5/5 partitions.
         Time taken to load dynamic partitions: 0.442 seconds
         Time taken for adding to write entity : 0.002 seconds
OK
food1   food2   food3   food4   id      name
Time taken: 15.946 seconds
hive (mydb)>
```

Query Output:



```
hive (mydb)> SELECT MIN(food2) as MIN , MAX(food2) as MAX ,AVG(food2) as AVG from foodratingspart where name like 'Mel' or name like 'Jill';
Query ID = hadoop_20220209224620_b9fd3e35-70d2-4b4b-a49d-f1a6cfebaed7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644433076084_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 6.03 s
----------------------------------------------------------------------------------------------
OK
min     max     avg
1       50      25.58994708994709
Time taken: 6.799 seconds, Fetched: 1 row(s)
hive (mydb)> SELECT MIN(food2) as MIN , MAX(food2) as MAX ,AVG(food2) as AVG from foodratingspart where name in ('Mel','Jill');
Query ID = hadoop_20220209224642_698112e9-987d-49bf-855e-ac24da0d1795
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644433076084_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 6.91 s
----------------------------------------------------------------------------------------------
OK
min     max     avg
1       50      25.58994708994709
Time taken: 7.495 seconds, Fetched: 1 row(s)
hive (mydb)>
```

Exercise 7) 2 points

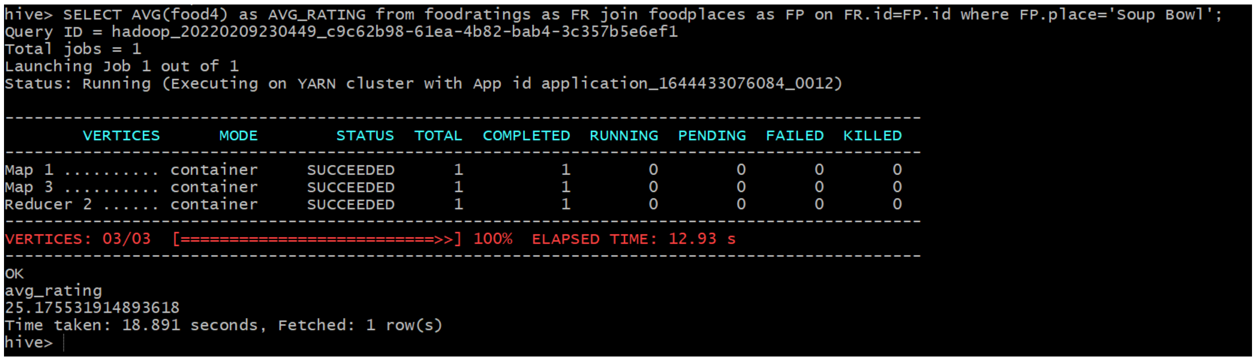
Load the foodplaces<.magic number>.txt file created using TestDataGen from your local file system into the foodplaces table.

LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces75212.txt' INTO TABLE 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 AVG(food4) as AVG_RATING from foodratings as FR join foodplaces as FP on FR.id=FP.id where FP.place='Soup Bowl';

Query Output:



```
hive> SELECT AVG(food4) as AVG_RATING from foodratings as FR join foodplaces as FP on FR.id=FP.id where FP.place='Soup Bowl';
Query ID = hadoop_20220209230449_c9c62b98-61ea-4b82-bab4-3c357b5e6ef1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644433076084_0012)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 3 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 12.93 s
----------------------------------------------------------------------------------------------
OK
avg_rating
25.175531914893618
Time taken: 18.891 seconds, Fetched: 1 row(s)
hive>
```

Exercise 8) 4 points

Read the article "An Introduction to Big Data Formats" found on the blackboard in section "Articles" and provide short (2 to 4 sentence) answers to the following questions:

a) When is the most important consideration when choosing a row format and when a column format for your big data file?

   Most important consideration while choosing row format or column format is rely on our objective. If we need access to all or most of the columns of each row of data, row format is better choice, column-based format is most beneficial when we need to retrieve information from certain columns examined over very large data sets which provides faster scan of data. It is also ideal for sparse data sets.

b) What is "splittability" for a column file format and why is it important when processing large volumes of data?

   Splittability in a column file format is taking batches of rows and store these batches in columnar format. This allow data to be processed in parallel which is key to performance. When query calculation is concerned with single column at a time, this method is more amenable.

c) What can files stored in column format achieve better compression than those stored in row format?

Storing values by column allows us to compress the data more efficiently than storing rows of data. For example, storing all the dates together in memory provides better compression than storing data of different type next to each other. This way, it can save storage cost for the user.

d) Under what circumstances would it be the best choice to use the "Parquet" column file format?

Parquet is proficient at analyzing huge datasets with several columns. In each Parquet file, binary data is organized by "row group." For each row group, the data values are organized by column Parquet offers flexible compression benefits. Parquet is smart choice for read heavy workloads.