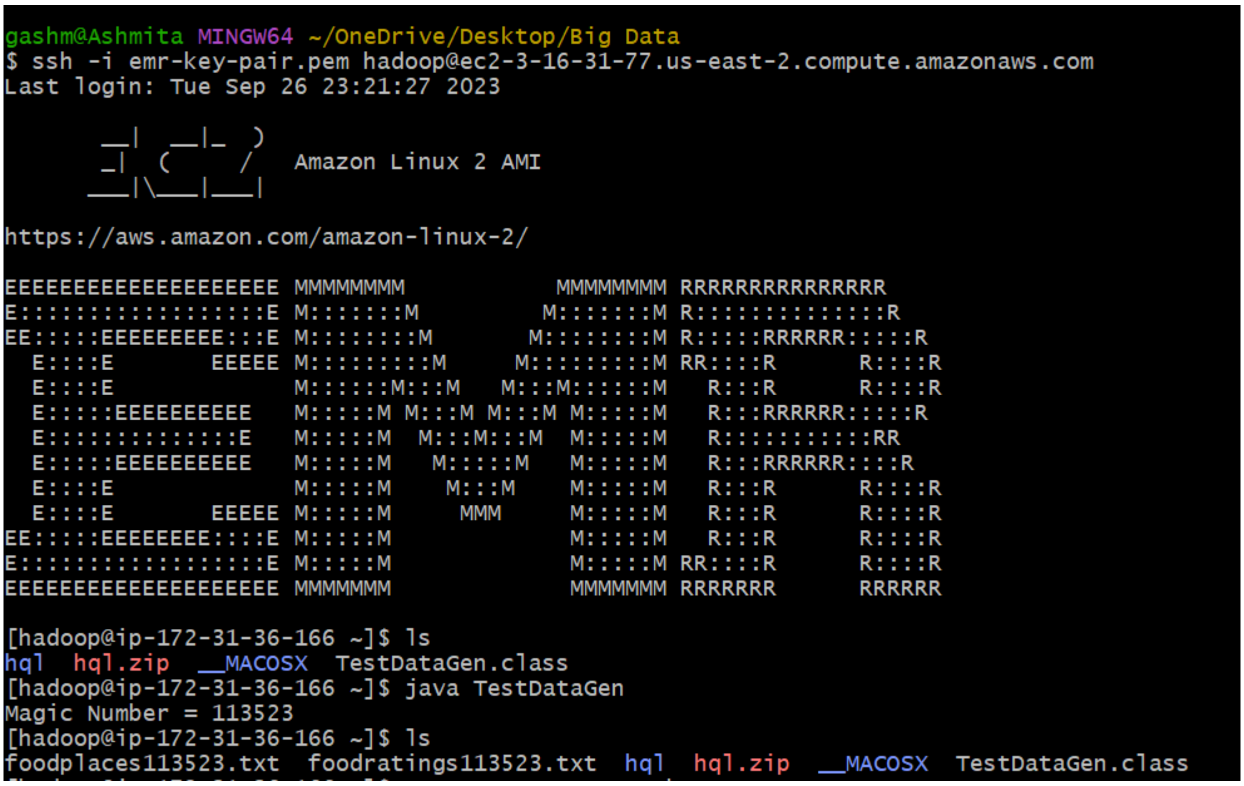
CSP 554

Assignment 4

Ashmita Gupta (A20512498)

1. **Generating magic number in AWS EMR cluster using command 'java TestDataGen'**

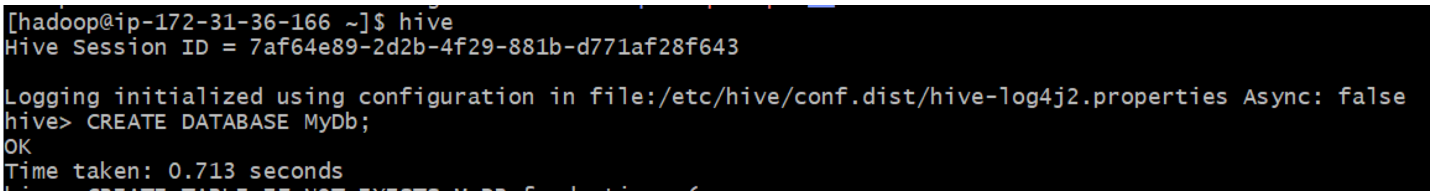


```
gashm@Ashmita MINGW64 ~/OneDrive/Desktop/Big Data
$ ssh -i emr-key-pair.pem hadoop@ec2-3-16-31-77.us-east-2.compute.amazonaws.com
Last login: Tue Sep 26 23:21:27 2023

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-36-166 ~]$ ls
hql  hql.zip  __MACOSX  TestDataGen.class
[hadoop@ip-172-31-36-166 ~]$ java TestDataGen
Magic Number = 113523
[hadoop@ip-172-31-36-166 ~]$ ls
foodplaces113523.txt  foodratings113523.txt  hql  hql.zip  __MACOSX  TestDataGen.class
```

Magic Number generated = 113523

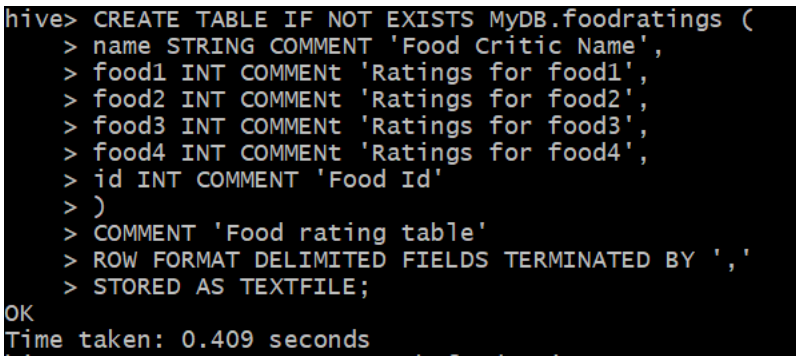
Exercise 1) (2 points) Create a Hive database called "MyDb".

Command: CREATE DATABASE MyDb



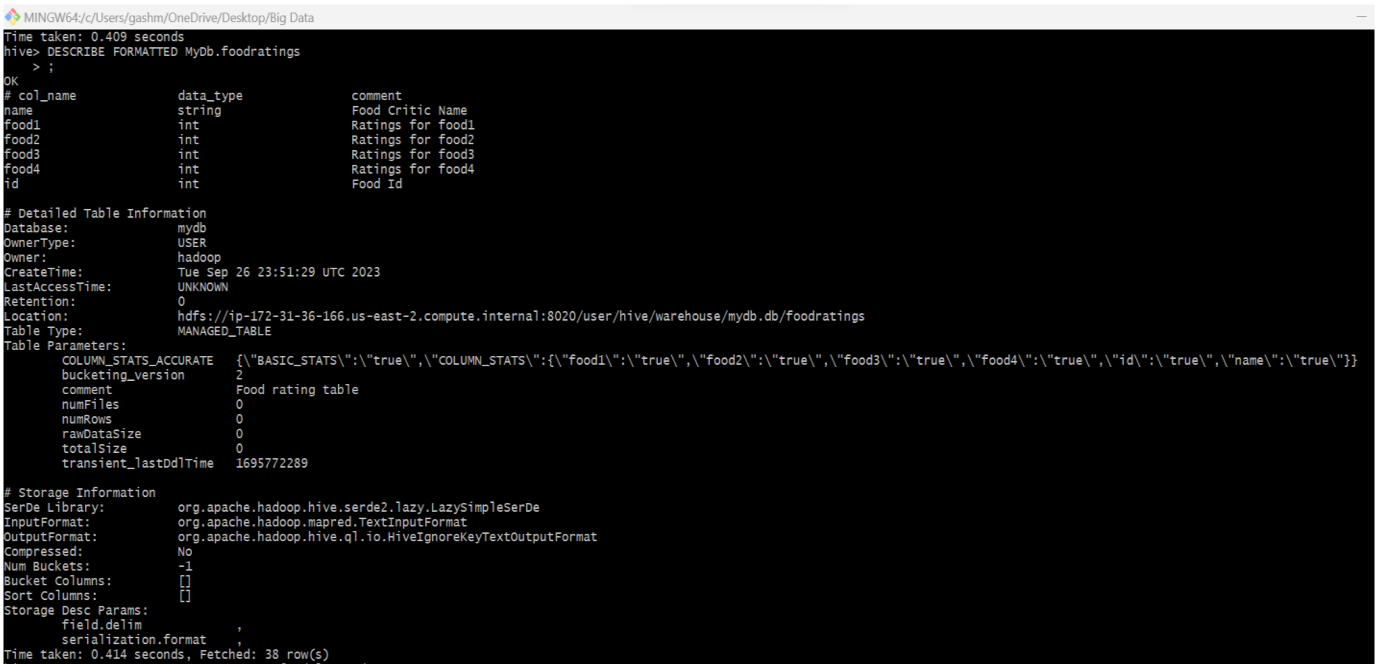
```
[hadoop@ip-172-31-36-166 ~]$ hive
Hive Session ID = 7af64e89-2d2b-4f29-881b-d771af28f643

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> CREATE DATABASE MyDb;
OK
Time taken: 0.713 seconds
```

Creating table "foodratings" in database MyDb:



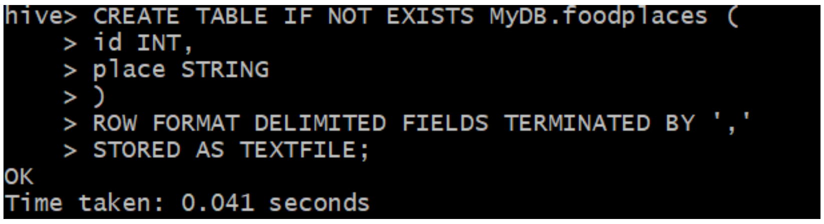
```
hive> CREATE TABLE IF NOT EXISTS MyDB.foodratings (
    > name STRING COMMENT 'Food Critic Name',
    > food1 INT COMMENT 'Ratings for food1',
    > food2 INT COMMENT 'Ratings for food2',
    > food3 INT COMMENT 'Ratings for food3',
    > food4 INT COMMENT 'Ratings for food4',
    > id INT COMMENT 'Food Id'
    > )
    > COMMENT 'Food rating table'
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.409 seconds
```

Executing command "DESCRIBE FORMATTED MyDb.foodratings;"



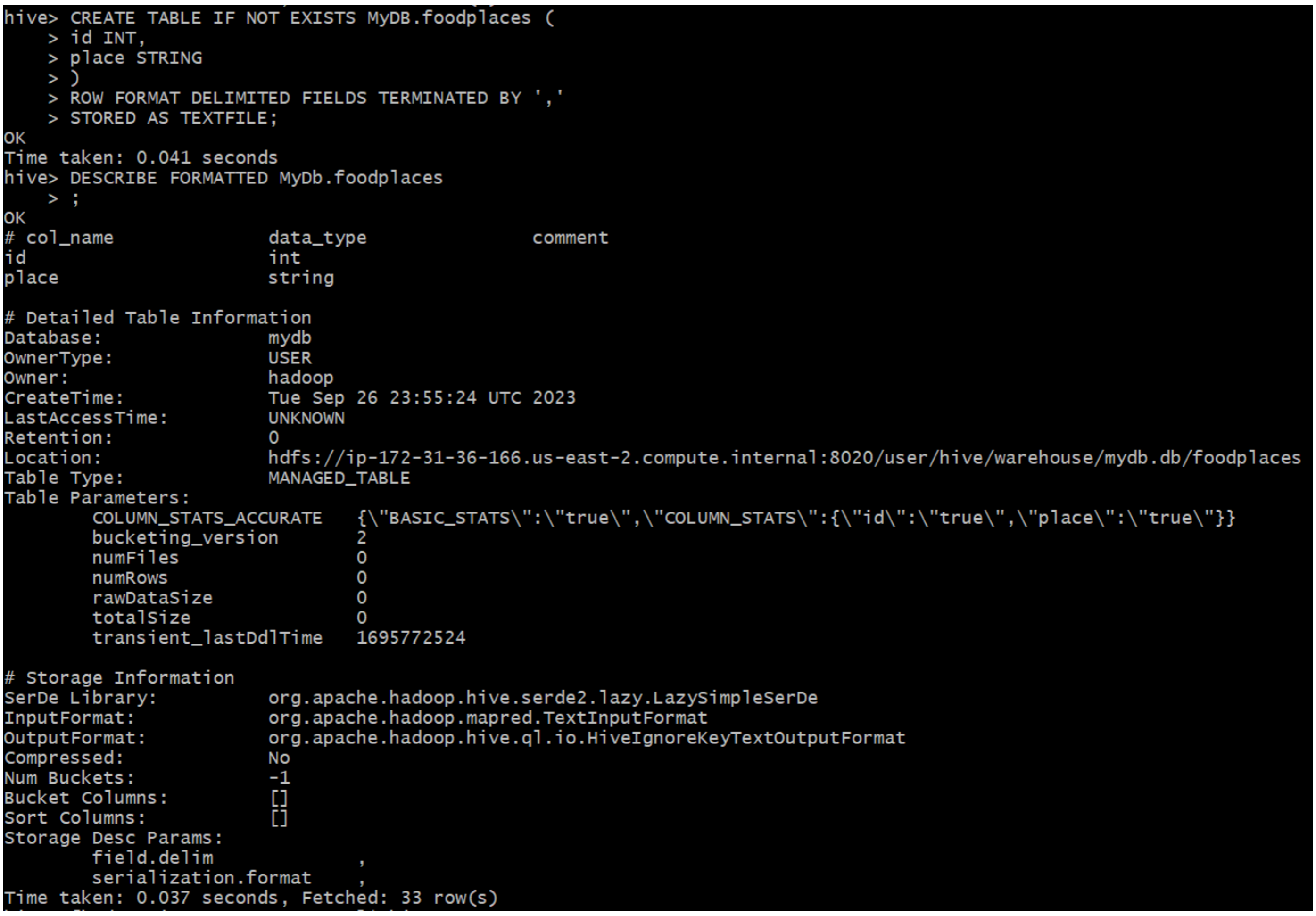
```
Time taken: 0.409 seconds
hive> DESCRIBE FORMATTED MyDb.foodratings
    > ;
OK
# col_name              data_type               comment
name                    string                  Food Critic Name
food1                   int                     Ratings for food1
food2                   int                     Ratings for food2
food3                   int                     Ratings for food3
food4                   int                     Ratings for food4
id                      int                     Food Id

# Detailed Table Information
Database:               mydb
OwnerType:              USER
Owner:                  hadoop
CreateTime:             Tue Sep 26 23:51:29 UTC 2023
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-36-166.us-east-2.compute.internal:8020/user/hive/warehouse/mydb.db/foodratings
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\",\"COLUMN_STATS\":{\"food1\":\"true\",\"food2\":\"true\",\"food3\":\"true\",\"food4\":\"true\",\"id\":\"true\",\"name\":\"true\"}}
        bucketing_version       2
        comment                 Food rating table
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1695772289

# 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: 0.414 seconds, Fetched: 38 row(s)
```

Create table" foodplaces" in the database MyDb



```
hive> CREATE TABLE IF NOT EXISTS MyDB.foodplaces (
    > id INT,
    > place STRING
    > )
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.041 seconds
```

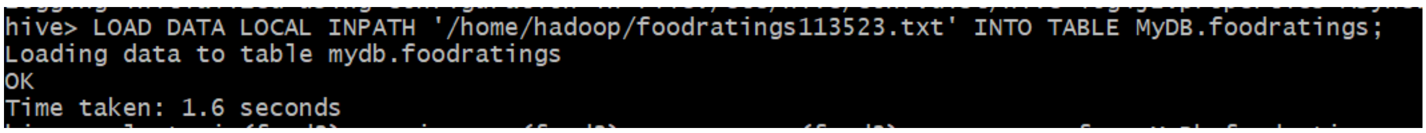
Executing command "DESCRIBE FORMATTED MyDb.foodplaces;"



```
hive> CREATE TABLE IF NOT EXISTS MyDB.foodplaces (
    > id INT,
    > place STRING
    > )
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.041 seconds
hive> DESCRIBE FORMATTED MyDb.foodplaces
    > ;
OK
# col_name              data_type               comment
id                      int
place                   string

# Detailed Table Information
Database:               mydb
OwnerType:              USER
Owner:                  hadoop
CreateTime:             Tue Sep 26 23:55:24 UTC 2023
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-36-166.us-east-2.compute.internal:8020/user/hive/warehouse/mydb.db/foodplaces
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\",\"COLUMN_STATS\":{\"id\":\"true\",\"place\":\"true\"}}
        bucketing_version       2
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1695772524

# 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: 0.037 seconds, Fetched: 33 row(s)
```

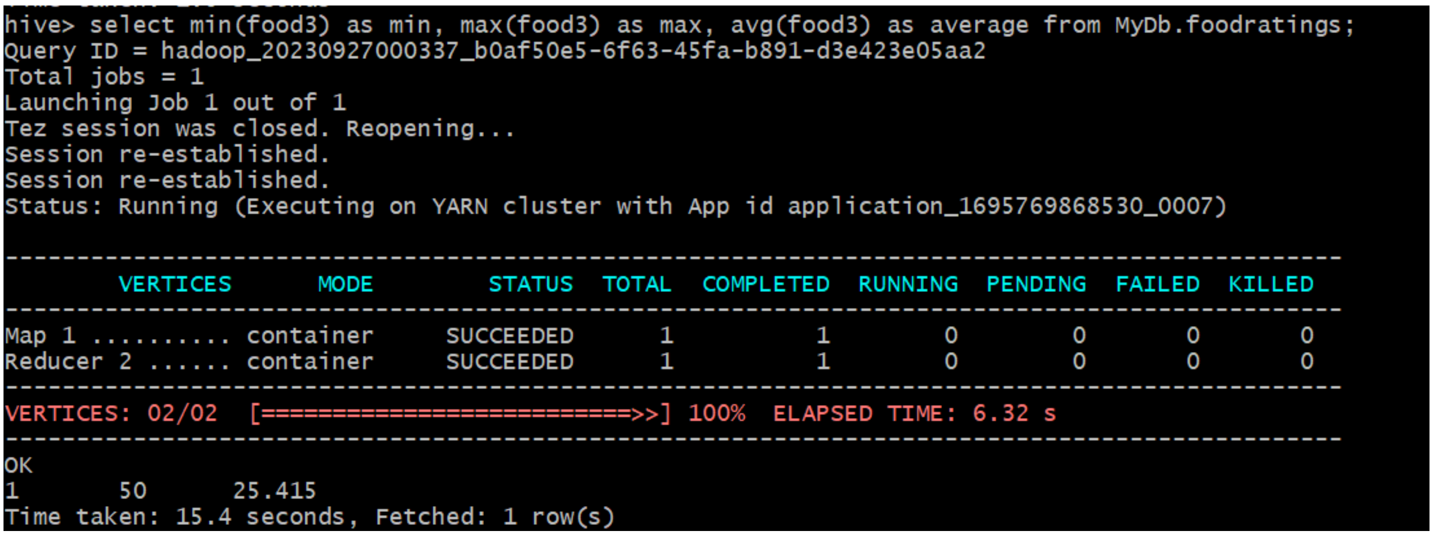
Exercise 2) 2 points

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



```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodratings113523.txt' INTO TABLE MyDB.foodratings;
Loading data to table mydb.foodratings
OK
Time taken: 1.6 seconds
```

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 one



```
hive> select min(food3) as min, max(food3) as max, avg(food3) as average from MyDb.foodratings;
Query ID = hadoop_20230927000337_b0af50e5-6f63-45fa-b891-d3e423e05aa2
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1695769868530_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: 6.32 s
----------------------------------------------------------------------------------------------
OK
1       50      25.415
Time taken: 15.4 seconds, Fetched: 1 row(s)
```

Magic Number= 113523

Exercise 3) 2 points

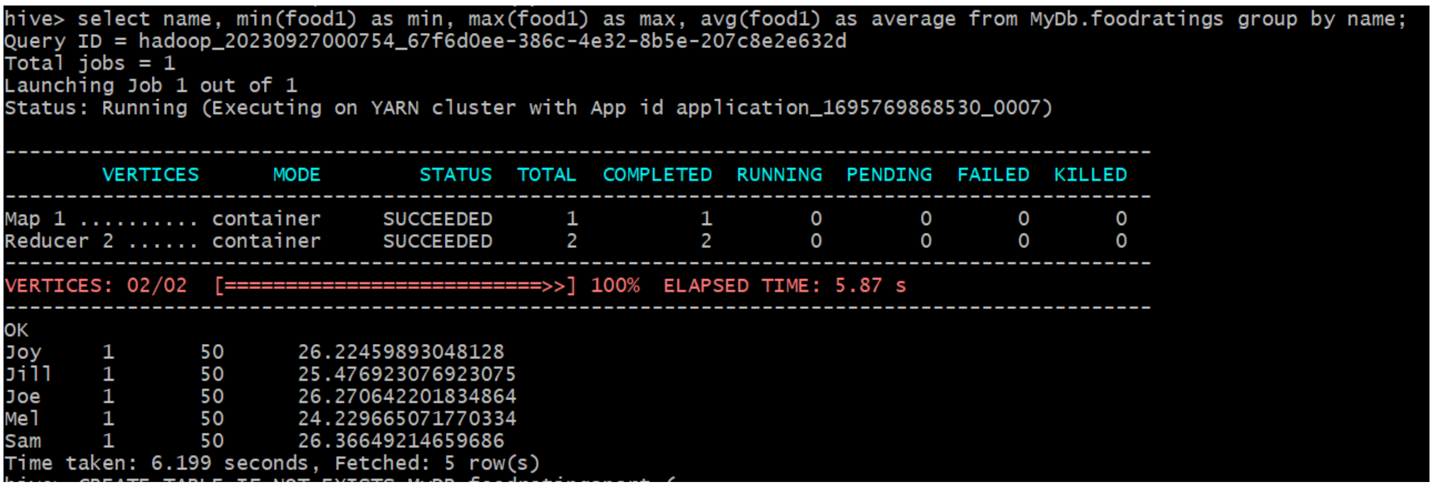
Execute a hive command to output the min, max and average of the values of the food1 column grouped by the first column 'name'. This should be one hive command, not three separate ones.

The output should look something like:

Mel 10 20 15

Bill 20, 30, 24

…

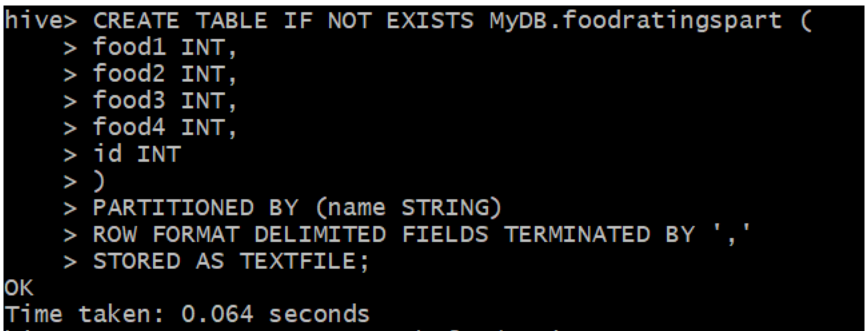


```
hive> select name, min(food1) as min, max(food1) as max, avg(food1) as average from MyDb.foodratings group by name;
Query ID = hadoop_20230927000754_67f6d0ee-386c-4e32-8b5e-207c8e2e632d
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1695769868530_0007)

----------------------------------------------------------------------------------------------
        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: 5.87 s
----------------------------------------------------------------------------------------------
OK
Joy     1       50      26.22459893048128
Jill    1       50      25.476923076923075
Joe     1       50      26.270642201834864
Mel     1       50      24.229665071770334
Sam     1       50      26.36649214659686
Time taken: 6.199 seconds, Fetched: 5 row(s)
```

<u>Magic Number= 113523</u>

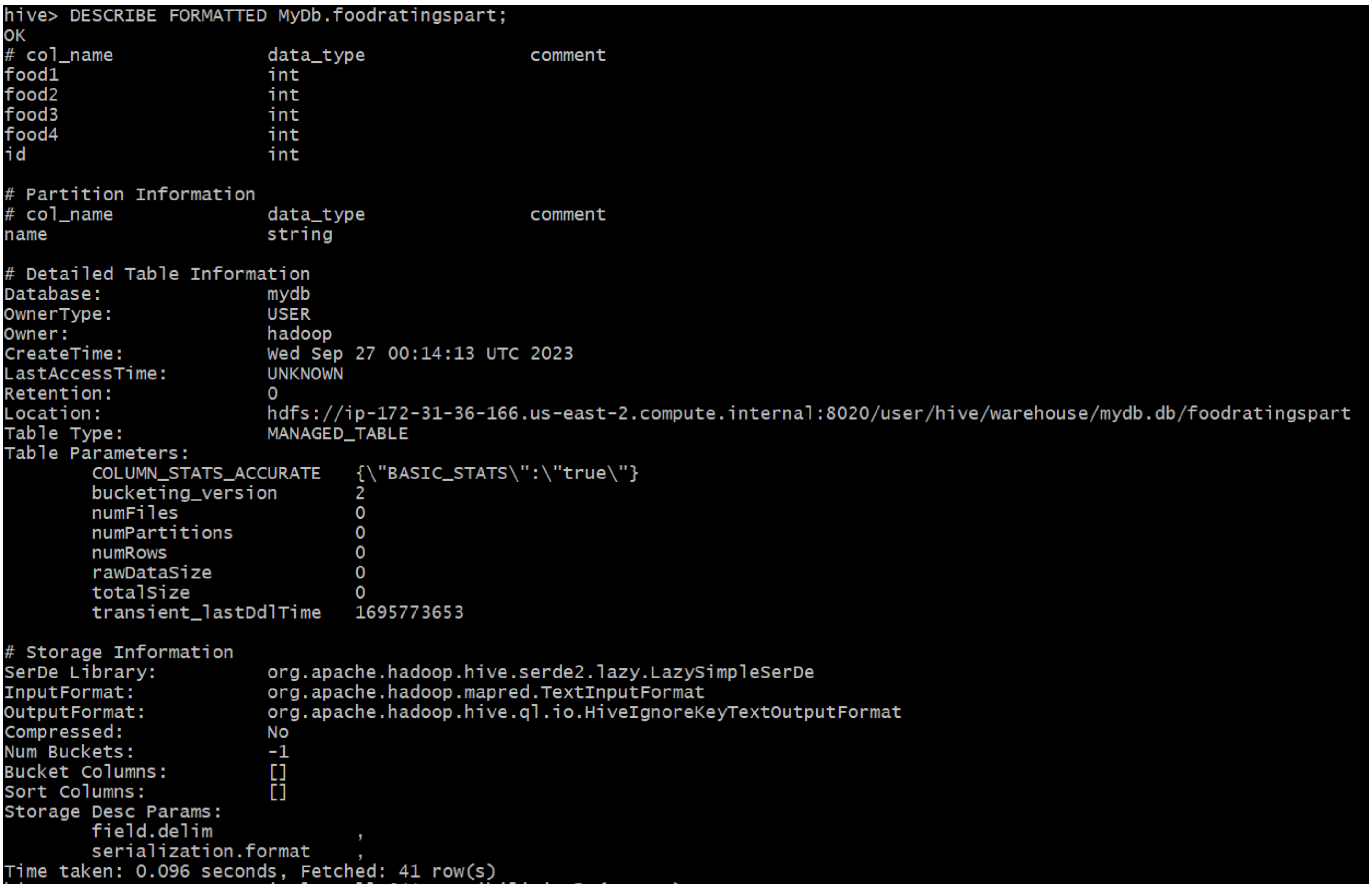
Exercise 4) 2 points

In MyDb create a partitioned table called 'foodratingspart'



```
hive> 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;
OK
Time taken: 0.064 seconds
```

Execute a Hive command of 'DESCRIBE FORMATTED MyDb.foodratingspart;' and capture its output as the result of this exercise.



```
hive> DESCRIBE FORMATTED MyDb.foodratingspart;
OK
# 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
OwnerType:              USER
Owner:                  hadoop
CreateTime:             Wed Sep 27 00:14:13 UTC 2023
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-36-166.us-east-2.compute.internal:8020/user/hive/warehouse/mydb.db/foodratingspart
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        bucketing_version       2
        numFiles                0
        numPartitions           0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1695773653

# 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: 0.096 seconds, Fetched: 41 row(s)
```

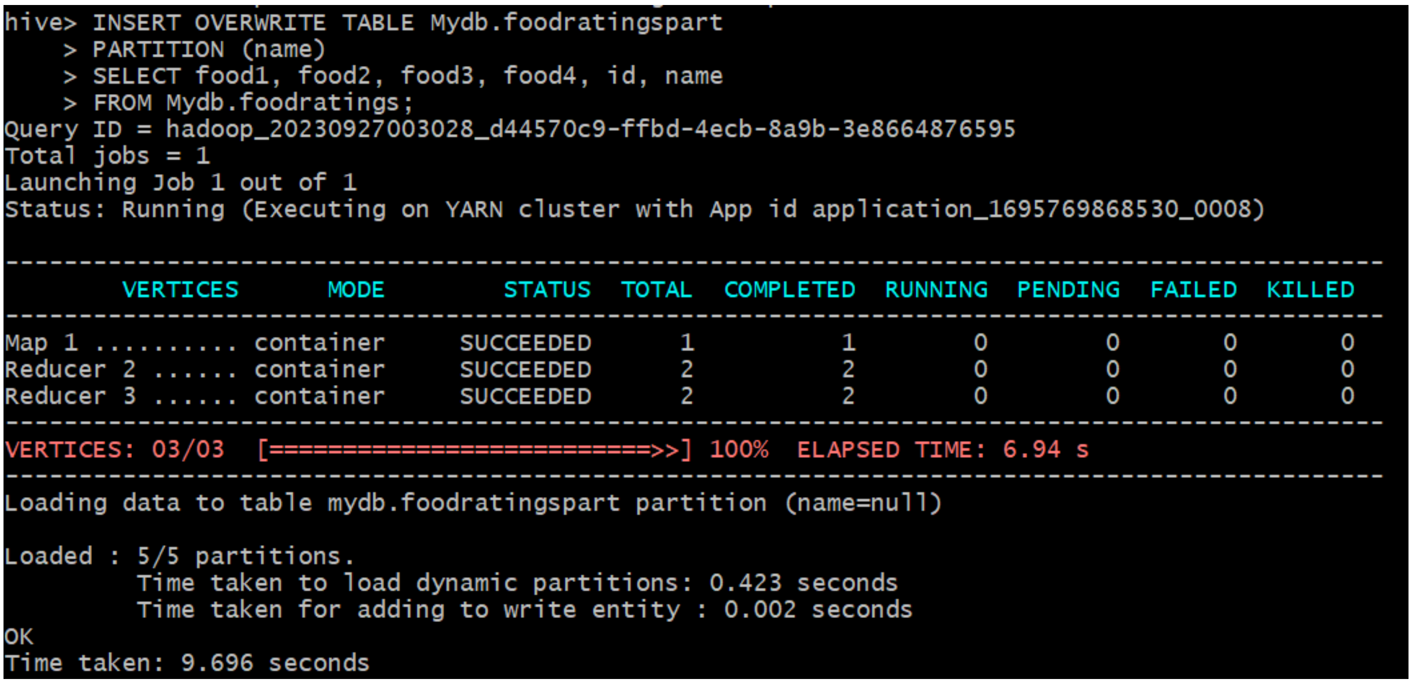
Exercise 5) 2 points

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.

We've selected the "critic name" as the partition field due to the limited number of food critics. By partitioning the table using the critic's name, we can create fewer partitions and this way larger number of records would be distributed under a fewer number of partitions. Alternatively, using an "id" for partitioning would result in excessive partitioning, which is less efficient.

Exercise 6) 2 points

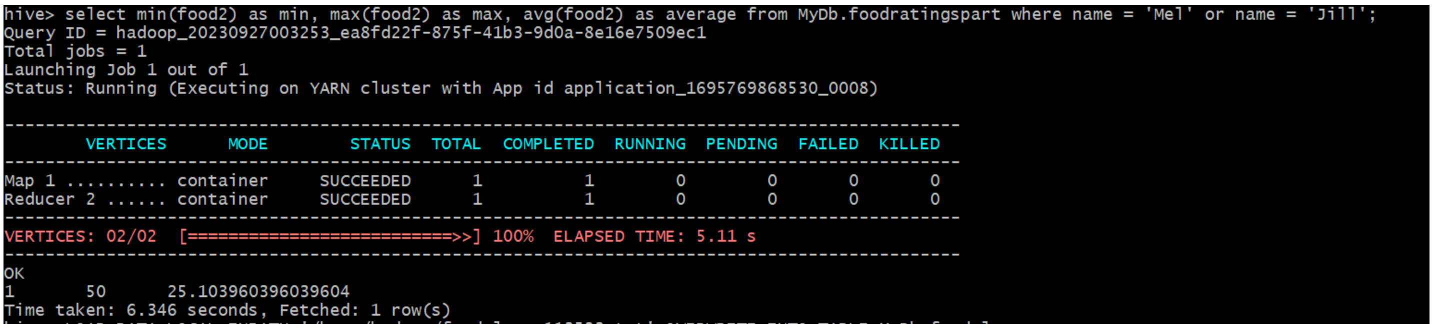
Configure Hive to allow dynamic partition creation as described in the lecture. Now, use a hive command to copy from MyDB.foodratings into MyDB.foodratingspart to create a partitioned table from a non-partitioned one.



```
hive> INSERT OVERWRITE TABLE Mydb.foodratingspart
    > PARTITION (name)
    > SELECT food1, food2, food3, food4, id, name
    > FROM Mydb.foodratings;
Query ID = hadoop_20230927003028_d44570c9-ffbd-4ecb-8a9b-3e8664876595
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1695769868530_0008)

----------------------------------------------------------------------------------------------
        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
Reducer 3 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 6.94 s
----------------------------------------------------------------------------------------------
Loading data to table mydb.foodratingspart partition (name=null)

Loaded : 5/5 partitions.
         Time taken to load dynamic partitions: 0.423 seconds
         Time taken for adding to write entity : 0.002 seconds
OK
Time taken: 9.696 seconds
```

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.

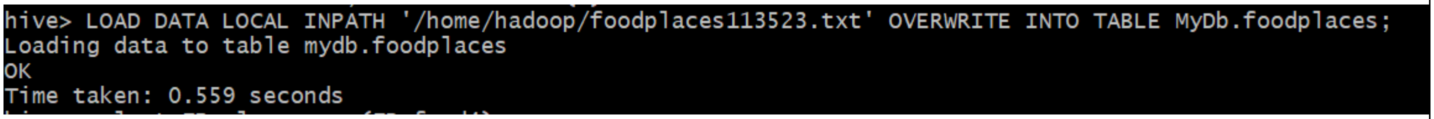
The query and the output of this query are other results of this exercise.



```
hive> select min(food2) as min, max(food2) as max, avg(food2) as average from MyDb.foodratingspart where name = 'Mel' or name = 'Jill';
Query ID = hadoop_20230927003253_ea8fd22f-875f-41b3-9d0a-8e16e7509ec1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1695769868530_0008)

----------------------------------------------------------------------------------------------
        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: 5.11 s
----------------------------------------------------------------------------------------------
OK
1       50      25.103960396039604
Time taken: 6.346 seconds, Fetched: 1 row(s)
```

Exercise 7) 2 points

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

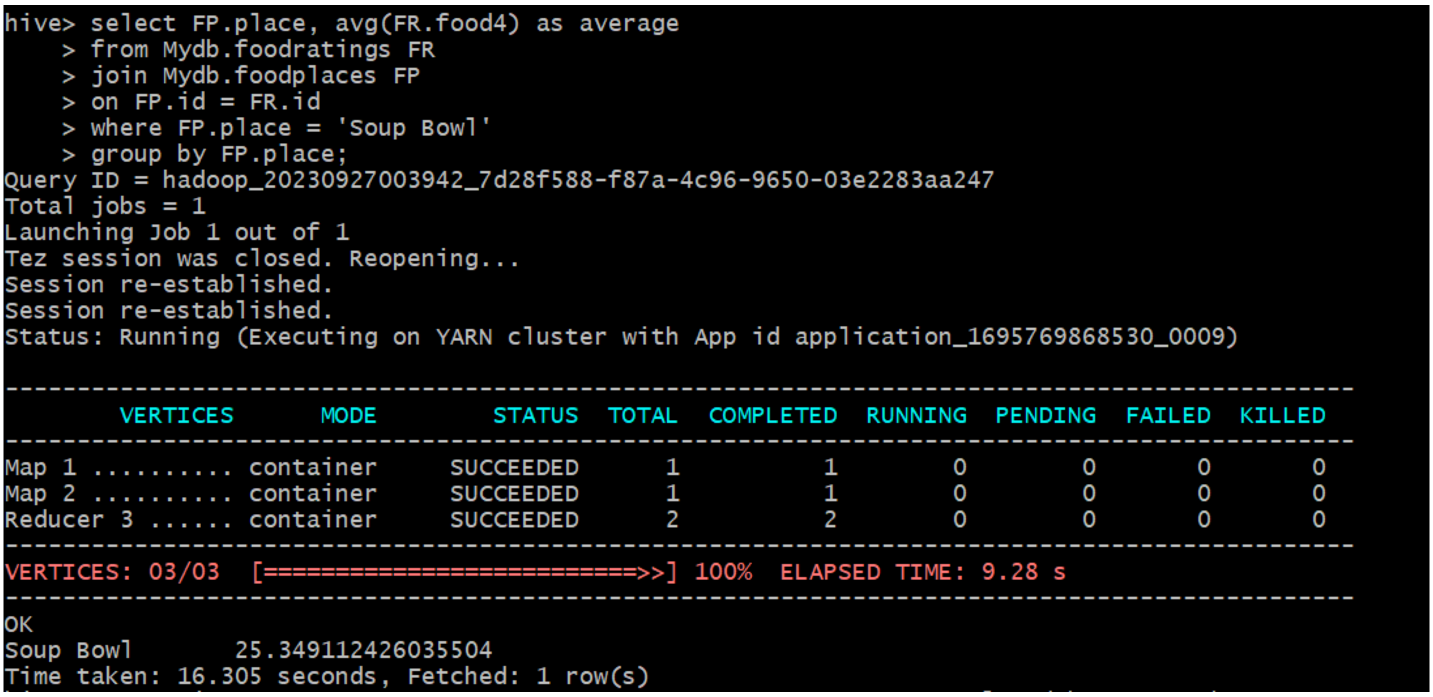


```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces113523.txt' OVERWRITE INTO TABLE MyDb.foodplaces;
Loading data to table mydb.foodplaces
OK
Time taken: 0.559 seconds
```

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



```
hive> 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;
Query ID = hadoop_20230927003942_7d28f588-f87a-4c96-9650-03e2283aa247
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1695769868530_0009)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 2 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 9.28 s
----------------------------------------------------------------------------------------------
OK
Soup Bowl       25.349112426035504
Time taken: 16.305 seconds, Fetched: 1 row(s)
```

Exercise 8) <u>**4 points**</u>

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?**

Ans. The most important consideration while choosing:

- **Row format** is when we are required to execute analytics queries that require a subset of columns examined over a large data set.
- **Column format** is chosen for big data file when the queries are required to access all or most of the columns of each row of data.

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

Ans. Splittability is basically the splitting of larger records into smaller records that can be handled independently. A column-based format will be more appropriate to split into separate jobs if the query calculation is concerned with a single column at a time. Spittability also helps in the **parallelization process** as Datasets are commonly composed of hundreds to thousands of files, each of which may contain thousands to millions of records or more. Furthermore, these file-based chunks of data are often being generated continuously and processing such datasets efficiently usually requires the job up into parts that can be given out to separate processors

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

Ans. Column format data can achieve a better compression rate than row row-based data as storing values by column, with the same data type next to each other, allows for doing more efficient compression on them, instead of storing the data on row. For example, storing all dates together in memory allows for more efficient compression than storing data of various types next to each other such as string, number, date, string, date

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

Ans. Parquet column format is a good choice when we are having a **read-heavy workload** as it enables benefits like splittability, compression, and schema evolution support. Parquet file contains binary data organized by row group and for each row group, the data values are organized by column which enables the compression benefits