Exercise 1)

Create a Hive database called “MyDb”.

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

Answer-

CREATING foodplaces TABLE

```
Time taken: 0.042 seconds
hive> CREATE TABLE foodplaces (
    >   id INT,
    >   place STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.06 seconds
```

CREATING foodratings TABLE

```
hive> CREATE TABLE foodratings (
    >   name STRING COMMENT 'Name of the critic',
    >   food1 INT COMMENT 'Rating for food1',
    >   food2 INT COMMENT 'Rating for food2',
    >   food3 INT COMMENT 'Rating for food3',
    >   food4 INT COMMENT 'Rating for food4',
    >   id INT COMMENT 'Restaurant ID'
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.392 seconds
```

DESCRIBE FORMATTED MyDb.foodratings;

```
hive> DESCRIBE FORMATTED MyDb.foodratings;
OK
# col_name              data_type               comment
name                    string                  Name of the critic
food1                   int                     Rating for food1
food2                   int                     Rating for food2
food3                   int                     Rating for food3
food4                   int                     Rating for food4
id                      int                     Restaurant ID

# Detailed Table Information
Database:               mydb
OwnerType:              USER
Owner:                  hadoop
CreateTime:             Fri Oct 13 05:04:47 UTC 2023
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-0-200.ec2.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
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1697173487

# 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: 0.405 seconds, Fetched: 37 row(s)
```

DESCRIBE FORMATTED MyDb.foodplaces;

```
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:             Fri Oct 13 05:07:17 UTC 2023
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-0-200.ec2.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   1697173637

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

Exercise 2)

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. This should be one hive command, not three separate ones.

Answer-

LOADING FROM LOCALFILE TO foodratings table:

LOAD DATA INPATH /user/hive/warehouse/MyDb.db/foodratings/foodratings86468.txt INTO TABLE foodratings;

TO VERIFY THE LOAD-

```
hive> SELECT * FROM foodratings LIMIT 10;
OK
Jill    36      29      27      33      2
Joy     48      1       17      31      1
Sam     30      36      3       46      3
Jill    27      27      28      23      3
Sam     26      20      29      20      2
Mel     47      10      4       44      4
Mel     30      38      40      16      2
Mel     25      2       7       8       3
Sam     7       6       26      18      5
Sam     9       7       6       30      4
Time taken: 3.305 seconds, Fetched: 10 row(s)
```

The hive command to output the min, max and average of the values of the food3 column is-

QUERY:

```
SELECT
 MIN(food3) AS min_food3,
 MAX(food3) AS max_food3,
 AVG(food3) AS avg_food3
FROM
 foodratings;
```

OUTPUT:

```
hive> SELECT
    >   MIN(food3) AS min_food3,
    >   MAX(food3) AS max_food3,
    >   AVG(food3) AS avg_food3
    > FROM
    >   foodratings;
Query ID = hadoop_20231013055507_5f8399f8-0e83-49ba-949c-d1f2abe5adf5
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_1697173131529_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: 6.96 s
--------------------------------------------------------------------------------
OK
1       50      26.08
Time taken: 18.707 seconds, Fetched: 1 row(s)
```

The magic number output is:

```
[hadoop@ip-172-31-0-200 ~]$ java TestDataGen
Magic Number = 86468
[hadoop@ip-172-31-0-200 ~]$
```

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’.

QUERY:

```
SELECT name,
  MIN(food1) AS min_food1,
  MAX(food1) AS max_food1,
  AVG(food1) AS avg_food1
FROM foodratings
GROUP BY name;
```

OUTPUT:

```
hive> SELECT name,
    >   MIN(food1) AS min_food1,
    >   MAX(food1) AS max_food1,
    >   AVG(food1) AS avg_food1
    > FROM foodratings
    > GROUP BY name;
Query ID = hadoop_20231013055900_5c93bcbc-d1d7-4dee-82c2-4ab39e87a957
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1697173131529_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
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 7.29 s
--------------------------------------------------------------------------------
OK
Joy     1       50      25.852631578947367
Jill    1       50      23.3463687150838
Joe     1       50      26.171296296296298
Mel     1       50      24.886255924170616
Sam     1       50      25.41176470588235
Time taken: 8.173 seconds, Fetched: 5 row(s)
```

The magic number output is:

```
[hadoop@ip-172-31-0-200 ~]$ java TestDataGen
Magic Number = 86468
[hadoop@ip-172-31-0-200 ~]$
```

Exercise 4)

Execute a Hive command of 'DESCRIBE FORMATTED MyDb.foodratingspart;'

Answer-

In the below screenshot, bucketing_version, numFiles, numPartitions, numRows, rawDataSize and totalSize contain values, as I captured the screenshot after copying from MyDb.foodratings into MyDb.foodratingspart to create a partitioned table from a non-partitioned one.

OUTPUT:

```
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:             Fri Oct 13 05:29:57 UTC 2023
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-0-200.ec2.internal:8020/user/hive/warehouse/mydb.db/foodratingspart
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        bucketing_version       2
        numFiles                5
        numPartitions           5
        numRows                 1000
        rawDataSize             12305
        totalSize               13305
        transient_lastDdlTime   1697174997

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

Exercise 5)

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-

The critic's name is a good choice for a partition field while using the place id is not, using food critics is a good decision as there are not many critics and if we use place id it would create a confusion as there are relatively large and each place id can have many reviews from different critic's which is not a good choice, if we partition using place id it would create many partitions which is less practical for effectively querying and organising data. Hence, using the critic name as partition field can help in uniquely organising data as each critic reviews each place id once.

Exercise 6)
Configure Hive to allow dynamic partition creation. Now, use a hive command to copy from MyDb.foodratings into MyDb.foodratingspart to create a partitioned table from a non-partitioned one. 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.
Answer-
The hive command to copy from MyDb.foodratings into MyDb.foodratingspart-

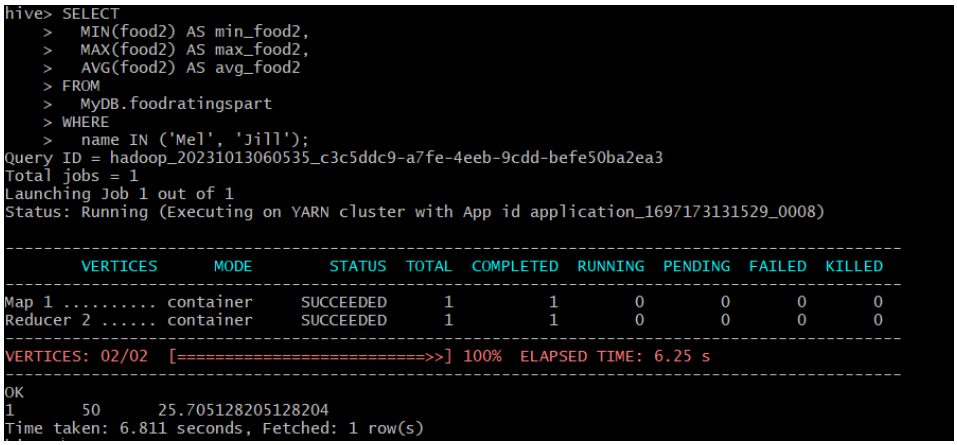
```
INSERT OVERWRITE TABLE MyDb.foodratingspart PARTITION (name)
SELECT food1, food2, food3, food4, id, name
FROM MyDb.foodratings;
```

The 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 –

QUERY:

```
SELECT
  MIN(food2) AS min_food2,
  MAX(food2) AS max_food2,
  AVG(food2) AS avg_food2
FROM
  MyDB.foodratingspart
WHERE
  name IN ('Mel', 'Jill');
```

OUTPUT:



Exercise 7)
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'.

QUERY:

```
SELECT p.place, AVG(r.food4) AS average_rating
FROM foodplaces p
JOIN foodratings r ON p.id = r.id
WHERE p.place = 'Soup Bowl'
GROUP BY p.place;
```

OUTPUT:

```
hive> SELECT p.place, AVG(r.food4) AS average_rating
    > FROM foodplaces p
    > JOIN foodratings r ON p.id = r.id
    > WHERE p.place = 'Soup Bowl'
    > GROUP BY p.place;
Query ID = hadoop_20231013074042_6b7e361d-2789-4751-b198-2d8b88cac70c
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1697173131529_0014)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 3 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 13.80 s
--------------------------------------------------------------------------------
OK
Soup Bowl       25.396551724137932
Time taken: 20.749 seconds, Fetched: 1 row(s)
```

Exercise 8)
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?
Answer-
The objectives are main factors to take into account when selecting a row format and a column format. When running analytics queries that only need a portion of the columns in extremely big data sets to be analysed, column-based storage is most advantageous. Row-based storage is more appropriate if queries need access to all or most of the columns in each row of data.

b) What is "splittability" for a column file format and why is it important when processing large volumes of data?
Answer-
In the context of columnar file formats, if a single column is being considered by the query computation at a time, a column-based approach will be more receptive to splitting into independent jobs. Due to its ability to support parallelism, scalability, efficient resource consumption, reduced latency, cost effectiveness, and ease of maintenance, splitability is essential when processing huge amounts of data.

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

Answer-

Files stored in column format achieve better compression than those in row format as, column format makes storing values of same type next to each other thus allowing efficient compression. Storing all dates together in memory(column format) allows for more efficient compression than storing data of various types next to each other(row format).

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

Answer-

The “Parquet” column file format would be best choice under the following circumstances:

- Read-Heavy Workloads.
- Data Serialization (self-describing -store metadata).
- Big Data Processing.
- Schema Evolution.
- Highly compressed and splitable.