Kajol Tanesh Shah
A20496724
Fall 2022

CSP554—Big Data Technologies

Assignment #4

- Download the file TestDataGen.class from the Blackboard. It is one of the attachments to the assignment.
- scp the file over to the home directory (/home/hadoop) on your Hadoop VM
- Log on to your VM using ssh and execute the file using "java TestDataGen"
- This will output a magic number which you should copy down and provide with the results of your assignment.
- It will also place the files foodratings<magic number>.txt and foodplaces<magic number>.txt in your VM home directory
- Use them for your exercises

Soln:-

```
(base) kajol@kajol-Lenovo-ideapad-310-15IKB:~$ cd Desktop/CSP554_Big_Data
(base) kajol@kajol-Lenovo-ideapad-310-15IKB:~/Desktop/CSP554_Big_Data$ scp -i e
mr-key-pair.pem TestDataGen.class hadoop@ec2-3-236-152-141.compute-1.amazonaws.
com:/home/hadoop
TestDataGen.class

          100% 2189    46.6KB/s   00:00
(base) kajol@kajol-Lenovo-ideapad-310-15IKB:~/Desktop/CSP554_Big_Data$
```

```
[hadoop@ip-172-31-11-247 ~]$ java TestDataGen
Magic Number = 133657
[hadoop@ip-172-31-11-247 ~]$ ls
foodplaces133657.txt  foodratings133657.txt  TestDataGen.class
[hadoop@ip-172-31-11-247 ~]$
```

Exercise 1) 2 points
Create a Hive database called “MyDb”.
Creating table “foodratings” in database “MyDb”.
Executing command “DESCRIBE FORMATTED MyDb.foodratings;”
Creating table “foodplaces” in database “MyDb”.
Executing command “DESCRIBE FORMATTED MyDb.foodplaces;”

Ans.

```
[hadoop@ip-172-31-11-247 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> CREATE DATABASE MyDb;
OK
Time taken: 2.293 seconds
hive> show databases;
OK
default
mydb
Time taken: 0.425 seconds, Fetched: 2 row(s)
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;
```

```
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: 2.474 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
Owner:                  hadoop
CreateTime:             Thu Sep 29 20:55:52 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-11-247.ec2.internal:8020/user/hive/warehouse/mydb.db/foodratings
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        comment                 Food rating table
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1664484952

# 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: 0.202 seconds, Fetched: 37 row(s)
hive>
```

CREATE TABLE IF NOT EXISTS MyDb.foodplaces (
id INT,
place STRING
)
COMMENT ‘Food places table’
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE;

```
hive> CREATE TABLE IF NOT EXISTS MyDb.foodplaces (
    > id INT,
    > place STRING
    > )
    > COMMENT 'Food places table'
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.137 seconds
```

```
hive> DESCRIBE FORMATTED MyDb.foodplaces
    > ;
OK
# col_name              data_type               comment

id                      int
place                   string

# Detailed Table Information
Database:               mydb
Owner:                  hadoop
CreateTime:             Thu Sep 29 21:04:12 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-11-247.ec2.internal:8020/user/hive/warehouse/mydb.db/foodplaces
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        comment                 Food places table
        numFiles                0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1664485452

# 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: 0.08 seconds, Fetched: 33 row(s)
hive>
```

Exercise 2) 2 points
Load the foodratings<magic number>.txt file created using TestDataGen from your local file system into the foodratings table.
Executing hive command to output the min, max and average of the values of the food3 column of the Foodratings table.

Ans. MAGIC NUMBER = 133657
LOAD DATA LOCAL INPATH '/home/hadoop/foodratings133657.txt' INTO TABLE MyDb.foodratings;

```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodratings133657.txt' INTO TABLE MyDb.foodratings;
Loading data to table mydb.foodratings
OK
Time taken: 0.978 seconds
```

SELECT MIN(food3) as min, MAX(food3) as max, AVG(food3) as average FROM MyDb.foodratings;

```
hive> SELECT MIN(food3) as min, MAX(food3) as max, AVG(food3) as average FROM MyDb.foodratings;
Query ID = hadoop_20220929212302_1345f8ae-3f8a-4028-9227-d4acc8051a9e
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1664483854288_0003)

--------------------------------------------------------------------------------
        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: 5.07 s
--------------------------------------------------------------------------------
OK
1       50      25.143
Time taken: 5.927 seconds, Fetched: 1 row(s)
hive>
```

Exercise 3) 2 points
Execute command to output the min, max and average of the values of the food1 column grouped by the first column 'name'.
Execute a Hive command of 'DESCRIBE FORMATTED MyDb.foodratingspart;'

Ans. Magic number = 133657
SELECT name, MIN(food1) as min, MAX(food1) as max, AVG(food1) as average FROM MyDb.foodratings GROUP BY name;

```
hive> SELECT name, MIN(food1) as min, MAX(food1) as max, AVG(food1) as average FROM MyDb.foodratings GROUP BY name;
Query ID = hadoop_20220929212425_17ccebe3-66fc-48b6-810f-83c5e8229c01
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1664483854288_0003)

--------------------------------------------------------------------------------
        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: 6.19 s
--------------------------------------------------------------------------------
OK
Jill    1       50      24.626373626373628
Joe     1       50      23.302439024390242
Joy     1       50      25.201970443349754
Mel     1       50      25.179372197309416
Sam     1       50      25.262032085561497
Time taken: 6.881 seconds, Fetched: 5 row(s)
hive>
```

Exercise 4) 2 points
In MyDb create a partitioned table called 'foodratingspart'

Ans.

```
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;

```
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: 0.118 seconds
```

```
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
Owner:                  hadoop
CreateTime:             Thu Sep 29 21:40:45 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-11-247.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   1664487645

# 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: 0.173 seconds, Fetched: 41 row(s)
hive>
```

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.

Ans. Using the (critic) name is a good choice for partition field because doing this can increase the efficiency and reduce the partition size. Also, if the number of columns is less, the result is processed faster as compared to a large number of columns. For eg. the processing is faster for results in less than 10 columns in comparison to creating result for 1000 columns.

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.
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.

Ans.
To configure hive for dynamic partition:-
set hive.exec.dynamic.partition.mode=nonstrict;

Create a partitioned table from a non-partitioned one:-

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

```
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> INSERT OVERWRITE TABLE MyDb.foodratingspart
    > PARTITION (name)
    > SELECT food1, food2, food3, food4, id, name
    > FROM MyDb.foodratings;
Query ID = hadoop_20220929214832_bf0d7bf4-6471-4803-bcbb-76eba44c1492
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_1664483854288_0004)

--------------------------------------------------------------------------------
        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: 7.13 s
--------------------------------------------------------------------------------
Loading data to table mydb.foodratingspart partition (name=null)

Loaded : 5/5 partitions.
         Time taken to load dynamic partitions: 0.528 seconds
         Time taken for adding to write entity : 0.002 seconds
OK
Time taken: 16.535 seconds
hive>
```

SELECT MIN(food2) as min, MAX(food2) as max, AVG(food2) as average FROM MyDb.foodratingspart WHERE name='Mel' or name='Jill';

```
    > 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_20220929215649_3a4432da-840d-4100-a677-d5bc79e8b6e7
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_1664483854288_0005)

--------------------------------------------------------------------------------
        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: 6.89 s
--------------------------------------------------------------------------------
OK
1       50      25.664197530864197
Time taken: 14.959 seconds, Fetched: 1 row(s)
hive>
```

Exercise 7) 2 points

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’

Ans.

LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces133657.txt' INTO TABLE MyDb.foodplaces;

```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces133657.txt' INTO TABLE MyDb.foodplaces;
Loading data to table mydb.foodplaces
OK
Time taken: 0.406 seconds
hive>
```

SELECT FPS.place, AVG(FRS.food4) as average
FROM MyDb.foodratings FRS
JOIN MyDb.foodplaces FPS
ON FRS.id = FPS.id
WHERE FPS.place = 'Soup Bowl'
GROUP BY FPS.place;

```
hive>
    > SELECT FPS.place, AVG(FRS.food4) as average
    > FROM MyDb.foodratings FRS
    > JOIN MyDb.foodplaces FPS
    > ON FRS.id = FPS.id
    > WHERE FPS.place = 'Soup Bowl'
    > GROUP BY FPS.place;
Query ID = hadoop_20220929221206_77f4137b-0222-4687-af36-98c7ffad2fee
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_1664483854288_0006)

--------------------------------------------------------------------------------
        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      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 12.91 s
--------------------------------------------------------------------------------
OK
Soup Bowl       25.019704433497537
Time taken: 22.013 seconds, Fetched: 1 row(s)
hive>
```

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:

1. 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 when choosing a row format and when a column format for your big data file is:-

   Column-based format is most useful when performing analytics queries that require only a subset of columns examined over very large data sets.

   Row-based format is most useful if your queries require access to all or most of the columns of each row of data.

2. What is "splitability" for a column file format and why is it important when processing large volumes of data?

   Ans. For a column-based format, splitability means splitting into separate jobs if the query calculation is concerned with a single column at a time. Datasets are commonly composed of hundreds to thousands of files and processing such datasets efficiently usually requires breaking the job up into parts that can be farmed out to separate processors. In fact, large-scale parallelization of processing is key to performance. So, splitability is important as it helps in the process of parallelization.

3. What can files stored in column format achieve better compression than those stored in row format?

   Ans. Columnar data can achieve better compression rates than row-based data. Storing values by column, with the same type next to each other, allows you to do more efficient compression on them than if you're storing rows of data. 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. Therefore column-based formats like Parquet and ORC achieve better compression than Avro.

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

Ans. Parquet is especially adept at analyzing wide datasets with many columns. Each Parquet file contains binary data organized by "row group." For each row group, the data values are organized by column which enables the compression benefits. So, Parquet is a good choice for read-heavy workloads and where data is stored by columns as it can be highly compressed and splittable.