Chirag Khandhar

A20438926 | CSP 554 - Big Data Technologies | Fall 2020 | Assignment 4

Magic Number = **118262**

Q 1. To Create Database:

```
CREATE DATABASE MyDb;
USE MyDB;
```

To Create Table foodratings:

```
CREATE TABLE MyDb.foodratings(
name STRING COMMENT ‘Critic Name’
food1 STRING COMMENT ‘food item1’,
food2 STRING COMMENT ‘food item2’,
food3 STRING COMMENT ‘food item3’,
food4 STRING COMMENT ‘food item4’,
id INT COMMENT ‘Restaurant ID’)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE
LOCATION ‘/home/hadoop’;
```

```
hive> CREATE TABLE MyDb.foodratings(
    > name STRING COMMENT 'Critic Name',
    > food1 INT COMMENT 'food item1',
    > food2 INT COMMENT 'food item2',
    > food3 INT COMMENT 'food item3',
    > food4 INT COMMENT 'food item4',
    > id INT COMMENT 'Restaurant ID')
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > LOCATION '/home/hadoop';
OK
Time taken: 0.307 seconds
hive> DESCRIBE FORMATTED MyDb.foodratings;
OK
# col_name              data_type               comment

name                    string                  Critic Name
food1                   int                     food item1
food2                   int                     food item2
food3                   int                     food item3
food4                   int                     food item4
id                      int                     Restaurant ID

# Detailed Table Information
Database:               mydb
Owner:                  hadoop
CreateTime:             Tue Sep 22 03:56:13 UTC 2020
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-16-9.ec2.internal:8020/home/hadoop
Table Type:             MANAGED_TABLE
Table Parameters:
        transient_lastDdlTime   1600746973

# 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: 0.156 seconds, Fetched: 31 row(s)
```

To Create Table foodplaces:

```
CREATE TABLE MyDb.foodplaces(
id INT,
place STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE
LOCATION ‘/home/hadoop’;
```

```
hive> CREATE TABLE MyDb.foodplaces(
    > id INT,
    > place String)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > LOCATION '/home/hadoop';
OK
Time taken: 0.055 seconds
hive> DESCRIBE FORMATTED MyDb.foodplaces;
OK
# col_name              data_type               comment

id                      int
place                   string

# Detailed Table Information
Database:               mydb
Owner:                  hadoop
CreateTime:             Tue Sep 22 04:04:22 UTC 2020
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-16-9.ec2.internal:8020/home/hadoop
Table Type:             MANAGED_TABLE
Table Parameters:
        numFiles                0
        totalSize               0
        transient_lastDdlTime   1600747462

# 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: 0.06 seconds, Fetched: 29 row(s)
hive>
```

Q 2. Loading Data:

```
LOAD DATA LOCAL INPATH '/home/hadoop/foodratings118262.txt'
OVERWRITE INTO TABLE foodratings;
```

```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodratings118262.txt'
    > OVERWRITE INTO TABLE foodratings;
Loading data to table mydb.foodratings
OK
Time taken: 1.327 seconds
```

Execute MIN, MAX, AVG functions on food3:

```
SELECT MIN(food3) AS MINIMUM, MAX(food3) AS MAXIMUM, AVG(food3) AS
AVERAGE FROM foodratings;
```

```
hive> SELECT MIN(food3) AS MINIMUM, MAX(food3) AS MAXIMUM, AVG(food3) AS AVERAGE FROM foodratings;
Query ID = hadoop_20200922043706_70fcac8d-bc1e-4b2d-902a-c3eb6b2a2d47
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1600743256402_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: 5.20 s
--------------------------------------------------------------------------------
OK
1       50      25.349
Time taken: 5.788 seconds, Fetched: 1 row(s)
hive>
```

Q 3. Execute MIN, MAX, AVG functions on food1, grouped by 'name':

```
SELECT MIN(food1), MAX(food1), AVG(food1) FROM foodratings GROUP BY name;
```

```
hive> SELECT name, MIN(food1), MAX(food1), AVG(food1) FROM foodratings GROUP BY name;
Query ID = hadoop_20200922044631_e4e8cb24-2627-4e04-b20f-522218141f70
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1600743256402_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: 5.86 s
--------------------------------------------------------------------------------
OK
Jill    1       50      26.05235602094241
Joe     1       50      27.214285714285715
Joy     1       50      26.380165289256198
Mel     1       50      25.523076923076925
Sam     1       50      25.178947368421053
Time taken: 6.452 seconds, Fetched: 5 row(s)
hive>
```

Q 4. Create a Partitioned table 'foodratingspart':

```
CREATE TABLE MyDb.foodratingspart(
food1 STRING,
food2 STRING,
food3 STRING,
food4 STRING,
id INT)
PARTITIONED BY (name STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE
LOCATION ‘/home/hadoop’;
```

```
hive>
    >
    > CREATE TABLE foodratingspart(
    > food1 INT,
    > food2 INT,
    > food3 INT,
    > food4 INT,
    > id INT)
    > PARTITIONED BY (name STRING)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE
    > LOCATION '/home/hadoop';
OK
Time taken: 0.051 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:             Tue Sep 22 04:59:11 UTC 2020
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-16-9.ec2.internal:8020/home/hadoop
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        numFiles                0
        numPartitions           0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1600750751

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

Q 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.

- This is because, first of all, as the number of critic name is relatively small, partitioning the table based on name would be having relatively small number of partitions. Thus, our large number of records would be distributed under these few number of partitions.

- As against the name, if we partition our tables using place id, it will create a very large number of partitions with small chunk of data in each partition, causing **Over Partitioning**. It will cause, an increased overhead in data loading and retrieval.

- That is the reason why, we choose (critic) name as our partitioning column.

Q 6. Allowing Dynamic Partition and Copy from MyDb.foodratings into MyDb.foodratingspart to create a partitioned table from a non-partitioned one.

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

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

```
hive> SET hive.exec.dynamic.partition = true;
hive> SET hive.exec.dynamic.partition.mode = non-strict;
hive> INSERT OVERWRITE TABLE foodratingspart
    > PARTITION (name) SELECT food1, food2, food3, food4, id, name FROM foodratings;
Query ID = hadoop_20200922050848_810b6f18-3569-4f45-b8db-d72a286d57dd
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_1600743256402_0005)

--------------------------------------------------------------------------------
        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: 0.436 seconds
         Time taken for adding to write entity : 0.001 seconds
OK
Time taken: 14.871 seconds
hive>
```

Execute MIN, MAX, AVG, on ‘food2’, of MyDb.foodratingspart

```
SELECT MIN(food2), MAX(food2), AVG(food2) FROM foodratingspart
WHERE name = ‘Mel’ OR name = ‘Jill’;
```

```
hive> SELECT MIN(food2), MAX(food2), AVG(food2) FROM foodratingspart WHERE name = 'Mel' OR name = 'Jill';
Query ID = hadoop_20200922051224_f5adaa9f-cb69-4e21-ba29-61ef9c882395
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1600743256402_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: 6.48 s
--------------------------------------------------------------------------------
OK
1       50      26.531088082901555
Time taken: 7.773 seconds, Fetched: 1 row(s)
hive>
```

Q 7. Load the foodplaces118262.txt file created using TestDataGen from your local file system into the foodplaces table.

```
LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces118262.txt'
OVERWRITE INTO TABLE foodplaces;
```

```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces118262.txt'
    > OVERWRITE INTO TABLE foodplaces;
Loading data to table mydb.foodplaces
OK
Time taken: 0.907 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'

```
SELECT FP.place, AVG(food4) FROM foodratings FR JOIN foodplaces FP ON
(FR.id = FP.id) WHERE FP.place  = 'Soup Bowl' GROUP BY FP.place;
```

```
hive> SELECT FP.place, AVG(FR.food4) FROM foodratings FR JOIN foodplaces FP ON (FR.id = FP.id) WHERE FP.place = 'Soup Bowl' GROUP BY FP.place;
Query ID = hadoop_20200922054723_8cf2e7f2-f288-4386-8c77-371b026115bc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1600743256402_0007)

--------------------------------------------------------------------------------
        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: 11.19 s
--------------------------------------------------------------------------------
OK
Time taken: 11.887 seconds
hive>
```

Cross Verifying:

```
SELECT * FROM foodplaces where place = 'Super Bowl';

SELECT AVG(food4) from foodratings where id = 5;

SELECT food4 FROM foodratinngs WHERE ID = 5;
```

```
hive> select * from foodplaces where place = 'Soup Bowl';
OK
5       Soup Bowl
Time taken: 0.105 seconds, Fetched: 1 row(s)
hive> select AVG(food4) from foodratings where id = 5;
Query ID = hadoop_20200922055754_35551f7b-2dfa-4cd2-813e-58554f8e44ec
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1600743256402_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: 6.01 s
--------------------------------------------------------------------------------
OK
NULL
Time taken: 6.539 seconds, Fetched: 1 row(s)
hive> select food4 from foodratings where id = 5;
OK
Time taken: 0.103 seconds
hive>
```

Q 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?
   - At the highest level, column-based storage is most useful when performing analytics queries that require only a subset of columns examined over very large datasets.
   - If your queries require access to all or most of the columns of each row of data, row-based storage will be better suited to your needs.

b. What is "splitability" for a column file format and why is it important when processing large volumes of data?
   - The ability of a file to be decomposed into smaller records that can be handled independently is called "Splitability".
   - A column-based format will be more amenable to splitting into separate jobs if the query calculation is concerned with a single column at a time (Row Columnar).
   - This means, they take a batch of rows and store that batch in columnar format. These batches then become split boundaries.
   - Thus, processing large volumes of data efficiently, requires breaking the job up into parts that can be farmed out to separate processors through large-scale parallelization.

c. What can files stored in column format achieve better compression than those stored in row format?
   - Columnar data can achieve better compression rates than row-based data.
   - Storing values by column, with the same type next to each other, allows us to do more efficient compression on them than if we're storing rows of data.

d. Under what circumstances would it be the best choice to use the "Parquet" column file format?
   - Parquet is commonly used with Apache Impala, an analytics database for Hadoop.
   - Impala is designed for low latency and high concurrency queries on Hadoop.
   - Parquet is especially adept at analyzing wide datasets with many columns.
   - As each Parquet file contains binary data organized by "row group." For each row group, the data values are organized by column thus enabling compression benefits.
   - Thus, Parquet is a good choice for read-heavy workloads.