Big Data Technologies-CSP554

Assignment-4

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

```
[hadoop@ip-172-31-58-143 ~]$ ls
[hadoop@ip-172-31-58-143 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-58-143 ~]$ java TestDataGen
Magic Number = 62032
[hadoop@ip-172-31-58-143 ~]$ ls
foodplaces62032.txt  foodratings62032.txt  TestDataGen.class
[hadoop@ip-172-31-58-143 ~]$
```

Magic Number generated =62032

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

Command Used= CREATE DATABASE MyDb

```
[hadoop@ip-172-31-58-143 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.
properties Async: false
hive> CREATE DATABASE MyDb;
OK
Time taken: 2.036 seconds
hive>
```

Create 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: 1.654 seconds
hive>
```

Executing command "DESCRIBE FORMATTED MyDb.foodratings;"

```
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 03:51:29 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-58-143.ec2.internal:8020/user/hive/ware
house/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   1664423489

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputForm
at
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.694 seconds, Fetched: 37 row(s)
hive>
```

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: 0.174 seconds
hive>
```

Executing command "DESCRIBE FORMATTED MyDb.foodplaces;"

```
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 03:55:48 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-58-143.ec2.internal:8020/user/hive/ware
house/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   1664423748

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputForm
at
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.161 seconds, Fetched: 32 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.

```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodratings62032.txt' INTO TABLE MyDb
.foodratings;
Loading data to table mydb.foodratings
OK
Time taken: 1.714 seconds
hive>
```

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.

```
hive> select min(food3) as min, max(food3) as max, avg(food3) as average from My
Db.foodratings;
Query ID = hadoop_20220929040507_25b48478-5cc9-4e33-b423-1b70daf3bfb5
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_1664422922857
_0002)

Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1
OK
1       50      26.196
Time taken: 18.693 seconds, Fetched: 1 row(s)
hive>
```

Magic Number= 62032

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 f
rom MyDb.foodratings group by name;
Query ID = hadoop_20220929040901_aeddb07a-7b6f-4829-900b-0987796bf7ea
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1664422922857
_0002)

Map 1: 0/1      Reducer 2: 0/2
Map 1: 0(+1)/1  Reducer 2: 0/2
Map 1: 1/1      Reducer 2: 0(+1)/2
Map 1: 1/1      Reducer 2: 1(+0)/2
Map 1: 1/1      Reducer 2: 2/2
OK
Jill    1       50      25.45026178010471
Joe     1       50      24.670212765957448
Joy     1       50      26.552511415525114
Mel     1       50      25.395833333333332
Sam     1       50      26.419047619047618
Time taken: 7.091 seconds, Fetched: 5 row(s)
hive>
```

Magic Number= 62032

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: 0.151 seconds
hive>
```

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

```
Time taken: 0.151 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 04:13:28 UTC 2022
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://ip-172-31-58-143.ec2.internal:8020/user/hive/ware
house/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   1664424808

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputForm
at
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             ,
        serialization.format    ,
Time taken: 0.153 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.

We choose the critic name for the partition field because the number of food critics is small, and partitioning the table on the basis of the critic name will have a smaller number of partitions hence, a larger number of records would be distributed under a fewer number of partitions. Also, if we use id in place of critic name then it will result in over-partitioning.

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_20220929043335_ec3ae4c0-3819-4f28-aff5-17d2282a70da
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_1664422922857
_0004)

Map 1: -/-      Reducer 2: 0/2
Map 1: 0/1      Reducer 2: 0/2
Map 1: 0/1      Reducer 2: 0/2
Map 1: 0(+1)/1  Reducer 2: 0/2
Map 1: 1/1      Reducer 2: 0(+1)/2
Map 1: 1/1      Reducer 2: 1(+1)/2
Map 1: 1/1      Reducer 2: 2/2
Loading data to table mydb.foodratingspart partition (name=null)


        Time taken to load dynamic partitions: 0.946 seconds
        Time taken for adding to write entity : 0.003 seconds
OK
Time taken: 26.888 seconds
hive>
```

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 My
Db.foodratingspart where name = 'Mel' or name = 'Jill';
Query ID = hadoop_20220929043922_05c67cff-d4c7-4e74-aaf0-6ef08875bccb
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_1664422922857
_0005)

Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1
OK
1       50      24.840731070496084
Time taken: 16.292 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.

```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/foodplaces62032.txt' OVERWRITE INTO T
ABLE MyDb.foodplaces;
Loading data to table mydb.foodplaces
OK
Time taken: 0.536 seconds
hive>
```

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_20220929044652_a2785ade-58b3-4672-b41a-5bbcbcf41db2
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_1664422922857_0006)

Map 1: 0/1      Map 3: 0/1      Reducer 2: 0/2
Map 1: 0/1      Map 3: 0/1      Reducer 2: 0/2
Map 1: 0/1      Map 3: 0(+1)/1  Reducer 2: 0/2
Map 1: 0(+1)/1  Map 3: 0(+1)/1  Reducer 2: 0/2
Map 1: 0(+1)/1  Map 3: 0(+1)/1  Reducer 2: 0/2
Map 1: 0(+1)/1  Map 3: 0/1      Reducer 2: 0/2
Map 1: 0(+1)/1  Map 3: 1/1      Reducer 2: 0/2
Map 1: 1/1      Map 3: 1/1      Reducer 2: 0(+2)/2
Map 1: 1/1      Map 3: 1/1      Reducer 2: 1(+1)/2
Map 1: 1/1      Map 3: 1/1      Reducer 2: 2/2
OK
Soup Bowl       25.82587064676617
Time taken: 21.164 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?**
The most important consideration when choosing a Row format is when we are required to execute analytics queries that require a subset of columns examined over a large data set.
The most important consideration when choosing a Column format for your big data file is when our 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?**
Splittability is defined as 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.

c) **What can files stored in column format achieve better compression than those stored in row format?**
Column format data can achieve a better compression rate than row row-based data. 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.

Under what circumstances would it be the best choice to use the "Parquet" column file format?
Parquet column format is used when we are having a heavy workload with important factors like splittability, compression, and schema evolution support. Parquet file contains binary data organized by row group.

Submitted By: -
Aastha Dhir
adhir2@hawk.iit.edu
CWID- A20468022