YCBS-257 - Data at Scale

Workshop 5

Hive – Impala

General Instructions:

The purpose of this workshop is to get you started with Hadoop Analyzing tools. Here you will learn how to query data using Apache Hive and Apache Impala.

Start your Cloudera QuickStart VM to complete the workshop

Online resources:

https://hive.apache.org/
https://impala.apache.org/index.html

Part 1 - Hive

This exercise aims to familiarize you with HiveQL Language.

Exercise 1: Analyzing the Bixi data using Hive

In this exercise you will create a new database and a new table then populate this table with Bixi data.

You can download the Bixi data from this link:
https://www.bixi.com/en/open-data

Part I:

1. Create a new directory /hivelab1 on HDFS and place the Stations_2018.csv file into it.
2. Open a new Terminal window and run Hive using this command:

```
$ beeline –u jdbc:hive2://localhost:10000
```

3. Create a new database bixi

```
create database bixi;
```

4. Make this base the current database

```
use bixi;
```

5. Create a new table Stations defining the following properties:
   a. using the coma (',') as fields separator

b. Skip the header line of the CSV file using
`TBLPROPERTIES ("skip.header.line.coount"="1")`

6. Open a new Terminal window and list recursively the content of the default HDFS Hive path :
`/user/hive/warehouse`
7. Populate the Stations table with the content of Stations_2017.csv file.
8. Print the 10 first line of the Stations table
9. How many Stations in the table?

10. Again, list recursively the content of the default HDFS Hive path, do you see any difference?
11. List the content of /hivelab1, do you see any file in this directory? Why?

12. Drop the Stations table
13. List recursively the content of the default HDFS Hive path, do you see any difference?

Part II:

1. Put back the Stations_2018.csv file into /hivelab1 directory
2. Create a new directory /hivelab2 on HDFS and place all the OD_2018-*.csv files into this directory.
3. Create a new External Hive table Stations meeting the following:
   a. Use the coma (`','`) as fields separator
   b. Set the data location to /hivelab1
   c. Add a TBLPROPERTIES statement to skip the header line of the CSV file.
4. Print the 10 first line of the Stations table
5. How many Stations in the table?

Do you see any difference with the previous section?
We didn't populate the table with the data? Why?

6. Create a new External Hive table BixData meeting the following:
   a. Use the coma (`','`) as fields separator
   b. Set the data location to /hivelab2
   c. Add a TBLPROPERTIES statement to skip the header line of the CSV file

6. How many rows are in BixiData table?
7. How long is the longest ride?
8. Print the top 10 longest ride.
9. Print the top 10 stations where rides start

10. Print the top 10 stations where rides end

Exercise 2: Hive Dynamic data partitioning and Bucketing

Dynamic partition in Apache Hive allows to divide data into partitions while loading the data into Hive partitioned tables.

In this exercise you will use the real_estate dataset. This dataset define these columns:

```
Street string, City string, Zip int, State string, Beds int, Baths int,
Sq__ft int, Type string, Price int
```

Part I: Dynamic partitioning

In Hive, dynamic partitioning mode is disabled by default. In this exercise you will learn how to enable it and how to use it.

1. Create a new HDFS directory /hivelab3 and place real_estate.csv file into this directory.
2. To enable dynamic partitioning in Hive type the following commands (into the Hive interface):

   ```
   set hive.exec.dynamic.partition = true;
   ```

   ```
   set hive.exec.dynamic.partition.mode=nonstrict
   ```
3. Create a new Hive table RealEstate using coma (`','`) as field separator and remove the CSV file header.
4. Populate RealEstate table with data from real_estate.csv file.
5. Create a new Hive table RealEstate_Part having the same columns as RealEstate table and Partitioned By 'City'.
6. Populate RealEstate_Part table with data from RealEstate table.
7. Explore recursively the content of the default HDFS Hive path. What did you found?

8. Print all addresses for the 'ANTELOPE' city.
9. Explore the content of 'ANTELOPE' partition directory on HDFS
   a. Use the Linux command `wc -l` to count the line into the file in this partition
   b. Compare it to the result you obtained in point 8.
      i. Did you found any difference?

Part 2: Dynamic Bucketing

The data inside partitions can be grouped and organized in Buckets. In this exercise you will learn how to create and organize Hive partition data into Buckets in order to optimize data searches and extractions.

1. Create a new table RealEstate_Bucket :
   a. Having the same columns as RealEstate table

b. Partitioned By 'City'
c. Clustered By 'Street' and Sorted by 'Price'
d. Buckets count : 4

2. Enforce dynamic Bucketing and Sorting using the following commands:

```
set hive.enforce.bucketing = true;
set hive.enforce.sorting=true;
```

3. Populate RealEstate_Bucket table from RealEstate table
4. Explore recursively the content of the RealEstate_Bucket HDFS path. What did you found inside each partition directory?

Part 2 – HCatalog

The goal of this lab is to familiarize you with the using of Apache HCatalog. You will be performing HCatalog instructions to perform table creation on the Hive metastore.

Exercise 1: Creating Table in HCatalog

This section describes, how to create a table using the HCatalog command line tool:

1. Open a new Terminal window
2. Create a new table employee using the following command:

```
$ hcat -e "CREATE TABLE IF NOT EXISTS employee( eid int, name String, salary String, position String) ROW FORMAT DEIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;"
```

3. Check whether it is created or not

```
$ hcat -e "show tables;"
```

4. Create a new empty document named employee.csv

```
$ gedit employee.csv
```

5. Fill employee.csv with this sample content

```
101,Micheal,45000,Technical manager
102,Rita,45000,Administrator
103,Sandra,25000,Technical support
104,John,40000,Hr manager
105,David,30000,DB Admin
```

6. Execute the load operation

```
$ hcat -e "LOAD DATA LOCAL INPATH '/home/<user>/employee.csv' OVERWRITE INTO TABLE employee;"
```

7. Show all rows in the table

```
$ hcat -e "SELECT * FROM employee;"
```

Exercise 2: Reading Hive table from Pig:

This section describes, how to read a Hive table from Pig.

1. Run Apache Pig (local mode is recommended)

```
$ pig -x local
```

2. Write the following Pig Latin Script to Load the employee table from the Hive metastore and store it as 'employee_data' using HCatLoader() and HCatStorer respectively to load and store the table.

```
table = LOAD 'employee' using org.apache.hive.hcatalog.pig.HCatLoader();
STORE table INTO 'employee_data' using
org.apache.hive.hcatalog.pig.HCatStorer();
```

Part 3 – Impala

In this part, we will be performing certain Impala queries to perform data analysis on the Pokemon Go dataset. The goal of this lab is to familiarize you with the using of Apache Impala.

Exercise 1: Pokémon Go characters analysis

In this section, we will be performing certain Impala queries to perform data analysis on Pokémon Go characters. Pokémon Go is a free-to-play, location-based augmented reality game developed by Niantic for iOS and Android devices. It was released only in July 2016 and only in selected countries.

Pokémon Dataset description

The dataset consists of 11 columns separated by a `','` and their respective description is as follows:

Columns	Type	Description
Number	int	id of each Pokémon
Name	string	the name of the Pokémon
Type 1		the property of a Pokémon
Type 2	string	the extended property of the same Pokémon *A Pokémon may be one or both the types. For instance, Charmander is a Fire type, while Bulbasaur is both a Grass type as well as a Poison type. With the current 18-type system, there are 324 possible ways to assign these types to Pokémon, along with 171 unique combinations. As of Generation VI, 133 different type combinations have been used.*
Total	int	represents the sum of all character points of a Pokémon (HP, Attack, Defense, Sp. Atk, Sp. Def, and Speed)
HP (Hit Points)	int	Pokémon Hit Points, which is a value that determines how much damage a Pokémon can receive. When a Pokémon's HP is down to '0', the Pokémon will faint. HP is the most frequently affected stat of them all, as a depleting HP is a key factor in winning a battle.
Attack	int	represents the Attack stat
Defense	int	represents the Defense stat
Sp. Atk	int	represents a Pokémon's Special Attack stat
Sp. Def	int	represents a Pokémon's Special Defense stat
Speed	int	represents the speed stat of a Pokémon

1. Create a new HDFS directory /Impalalab1 and place Pokemon.csv file into this directory.
2. Use Hive to create a new external table pokemon pointing to the Pokemon csv file.
3. Fill the pokemon table with rows from Pokemon.csv file

4. Start a new Impala session and refresh the Impala meta-store cache using this command:

```
invalidate metadata;
```

Problems Statements:

5. Find out the average HP (Hit points) of all the Pokémon.

We need to segregate the Pokemon. In order to segregate the Pokémon, you will use **if** condition inside the **select** statement, which will create one more column in our dataset. The **if** condition should be used in the following manner inside a HiveQL query.

```
if(expr1,expr2,expr3)
```

Now, you will create a table based on the condition that

- If the HP `is greater` than the average HP, then it is `Powerful`,
- If the HP `is less` than the average, then it is `Moderate`
- A neutral condition is considered as `Powerless`.

6. Create a new table 'pokemon1' and insert values of existing table 'pokemon' into this table, with an additional column 'power_rate' to find the count of 'powerful' and 'moderate' from the table 'pokemon1'.

7. Find out the number of `Powerful` and `Moderate` HP Pokémons present in the dataset.
8. Find out the top 10 Pokémons according to their HP's
9. Find out the top 10 Pokémons based on their Attack stat
10. Find out the top 10 Pokémons based on their Defense stat
11. Find out the top 10 Pokémons based on their total power