CSE3BDC/CSE5BDC
Lab 03: Hive, Part 2

Department of Computer Science and IT, La Trobe University

Objectives

- Reinforce skills related to basic Hive syntax and execution
- Learn more advanced Hive techniques for optimising queries

This lab will again require the use of the docker container that is used throughout this course. You will also be required to apply knowledge from the previous lab on Hive—**so if you have not completed last week's lab, please do so before continuing.**

Web Traffic Analysis

You have gained some familiarity with basic Hive syntax and execution; using grouping, ordering, and joins in Lab 02. Now it's time to practice these skills further, with a more useful dataset. Included in the directory "`Input_data/1/`" is a set of data files that define four sets of data:

- A set of domains in the file `domains.csv`, comprised of URL and category. Below is an example of the data in the file:

URL	category
www.facebook.com	Social
www.netflix.com	Entertainment
australia.gov.au	Government

- A set of unique IPv4 addresses and their integer representations, used for comparing to other IP address ranges, in the file `ips.csv`. Below is an example of the data in the file:

ipAddress	intAddress
75.73.40.6	1263085574
223.23.156.177	3742866609
91.230.237.94	1541860702

- A set of IPv4 address ranges (min and max), their integer representations (min and max), as well as their region of origin, in the file `regions.csv`. Below is an example of the data in the file.

ipMin	ipMax	intMin	intMax	regionCode	regionName
1.0.0.0	1.0.0.255	16777216	16777471	AU	Australia
1.0.1.0	1.0.3.255	16777472	16778239	CH	China
1.0.4.0	1.0.7.255	16778240	16779263	AU	Australia

- A set of traffic history over the month of February, comprised of URL, IPv4 Address, and a timestamp of occurrence, in the file `traffic.csv`.

URL	ipAddress	time
www.facebook.com	15.116.12.84	2014-02-12 04:25:43
www.netflix.com	35.229.57.93	2014-02-10 12:01:31
australia.gov.au	47.18.175.209	2014-02-06 01:10:38

You will use Hive to join and group these data sets by common keys in order to analyse the data and produce sets of meaningful output.

The datasets are presented in .csv format (comma separated value), which means that there is one row of data per line and each column of data is separated by a comma. You will need to set Hive's default delimiter when creating each table by specifying at the end of the create table statement:

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

Task 1: Load the CSV Data

Because Hive stores its data in tables on HDFS when it loads from a file, it is useful to spend some initial time reading the raw dataset into Hive tables. As such, you will first be required to create a number of tables, one for each dataset, and then load data from the corresponding .csv files provided. Open the file `t1-loaddata.hql` in a text editor to complete the below tasks.

1. **This one has been done for you as an example:** Load the contents of the domains data file, `domains.csv`, to a table called `mydomains`, which stores a web `URL` as a `string`, and a `category` as a `string`.
2. Load the contents of the IPv4 data file, `ips.csv`, to a table called `myips`, which stores an `ipAddress` as a `string`, and the IP integer `intAddress` as a `BIGINT`—this is very important to avoid integer overflow.
3. Load the contents of the IP regions data file, `regions.csv`, to a table called `myregions`, which stores two IPv4 addresses (`ipMin` and `ipMax`) as `strings`, two IPv4 integer (`intMin` and intMax) represented as `BIGINTs`, a `regionCode` as a `string`, and a `regionName` as a `string`.
4. Load in the contents of the traffic data file, `traffic.csv`, to a table called `mytraffic`, which stores a web `URL` as a `string`, an `ipAddress` as a `string`, and a `time` as a `TIMESTAMP`.

   Run the script, then use the Hive interpreter to take a look at the produced tables to make sure the data is loaded correctly.

Task 2: Analysis

Task 2A: Ordered Traffic Count by Domain URL

One of the most essential skills in using Hive is simply to be able to produce a query that will analyse a dataset and provide some key insight to the nature of the data. You are hence required to produce a set of the most popular web domains, as determined by the URLs with the highest amount of traffic. The top 3 should be the following:

URL	**Count**
www.MyFace.com	23006
www.TotalaMad.com.fr	23000
www.Blooger.com	22997

1. Create a new script called `t2-a.hql` and in it do the following operations.

a. Using the `mytraffic` table, count the number of rows for each unique URL using the `group by` function.

b. Order the data by count in descending order.

c. Write the resultant table data to the local directory ./task2a-out/, such that each line contains the URL and traffic count separated with a tab character (\t).

Task 2B: Traffic History by Date Range for a Domain

On top of analysing data, it is also important to be able to selectively query for data that conforms to particular constraints or requirements. You are required to produce a set of all traffic history that occurred on a particular day, for a particular site. An example of the output is shown below:

URL	**ipAddress**	**Time**
www.Facebook.com	124.241.168.42	2014-02-14 01:41:24
www.Facebook.com	134.15.231.51	2014-02-14 10:51:06
www.Facebook.com	75.73.40.6	2014-02-14 01:47:56
…	…	…

1. Create a new script called `t2-b.hql` and in it do the following operations.

   a. Filter the data from the `mytraffic` table using a `WHERE` clause in order to find **only** the traffic for the domain URL "`www.Facebook.com`".

   b. Filter the data further by extending the `WHERE` clause in order to find **only** the traffic that occurred between `2014-02-14 00:00:00` and `2014-02-15 00:00:00`.

      Note: You will need to use the `unix_timestamp()` function to convert date strings (like `'2014-02-14 00:00:00'`) to TIMESTAMPs before using them in a comparison. So, for example, the expression which matches records where the value of the time column comes after 2014-02-14 is:

      ```
      time > unix_timestamp('2014-02-14 00:00:00')
      ```

   c. Write the resultant table data to ./task2b-out/. Check the results to make sure that they are for the correct URL and are in the expected date range.

Task 2C: Traffic Count by Category

Moving toward the more advanced side of working with big data, it is extremely useful, often even necessary, to be able to join two or more datasets in order to produce some more meaningful result that each individual dataset alone could not provide. You are hence required to produce a list of the top 5 most popular *categories* of domains on the web. This can be determined by the sum of all traffic for every domain that belongs to each category. The list should be in descending order of count. The output should be exactly the following:

Category	**Count**
Education	136232
News	91272
Commerce	68072
Social	67962
Entertainment	67957

Hint: You will need to combine data from the tables mytraffic and mydomains using an inner join. Refer to Lab 02 for guidance on how to use joins.

1. Create a new script called `t2-c.hql` and in it write the script required to complete this task.
2. Make sure that your script writes the results to ./task2c-out/.
3. It is up to you to figure out how to do this. You may collaborate with a neighbour or ask your demonstrator for help, however only minimal assistance will be provided!

Task 2D: Traffic Count by Country (Bonus Task)

It's time we perform the most complicated task yet. However, as we'll see, if we break the problem down into a number of smaller steps, it becomes much easier to conceptualise and code. We will write a script which will give us the total number of visits by each region to Facebook between `2014-02-14 00:00:00` and `2014-02-15 00:00:00`. (Yes, those are the same dates as before, so we will be using the table you created in task 2B). We have provided a minimal skeleton file `t2-d.hql`.

1. Create a table called `ipcountries` which contains two columns. One is the IP addresses from the table `myips`, and the other is its associated region name pulled from `myregions`. Remember, an IP address belongs to a region if its int address falls *between* the int-min and int-max of that region. Due to a limitation in the way Hive processes joins*, we've started writing the solution for you – just fill in the blanks.
2. Create a table called `fbtrafficcountries` which contains two columns. One is the region name – as taken from the table you just created, and the other is the time of a visit to Facebook between the two given dates – as taken from the table you created in task 2B. This should take no more than a simple `INNER JOIN`.

**Hive doesn't allow for inner-joining tables using the less-than or greater-than operator. Thus, we need to perform a cross-join, which creates a table of every combination of entries between the two tables. Then, we filter it using the less-than and greater-than operators to leave only the entries we want.*

3. Create a table called `fbregioncounts` which contains the number of visits to Facebook from each country between the given dates. We've done something similar to this several times already, and only involves `COUNT`, `GROUP BY` and `ORDER BY`.

4. Write the contents of the table you just created to the directory `'./task2d-out/'` and verify its contents. The first few lines should look like the below.

Region Name	**Count**
United States	299
China	48
France	28
…	…