**Udemy Analysis Project – Report**

Author: Michela Ieva

Date: 04/17/2017

**Abstract**

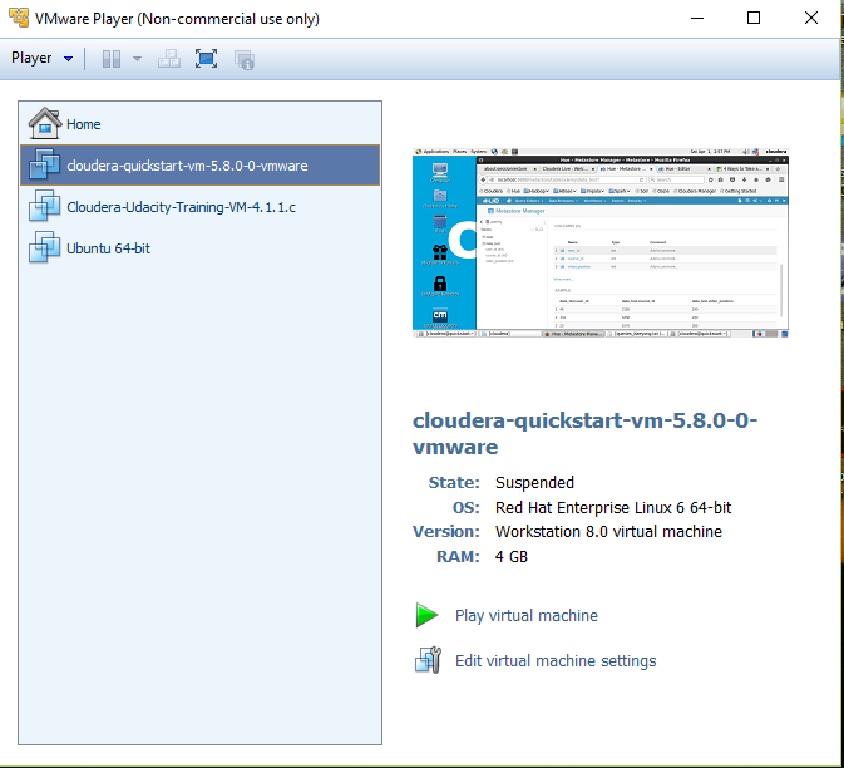
In this document I will report analysis results after a week of work.

**System setting up**

As first step for the analysis project I needed to choose the best way to setting up a system with Hadoop 2.6 and Hive 0.11.

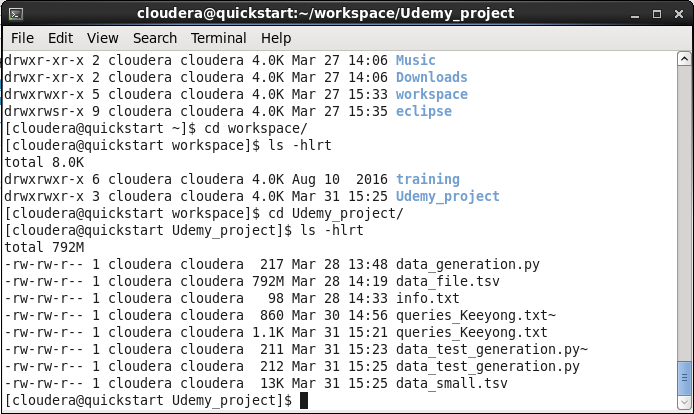
Looking for instructions on the Web I immediately realized that installing each single components, (Java first, then Hadoop, Jar and finally Hive) it could take too much time and results in incompatibility problems with different software versions.

So, I decided to use a Virtual Machine, also because I have a Windows laptop, and go for a Cloudera VM with CDH 5.8. In this way the Hadoop ecosystem was already installed and configured.



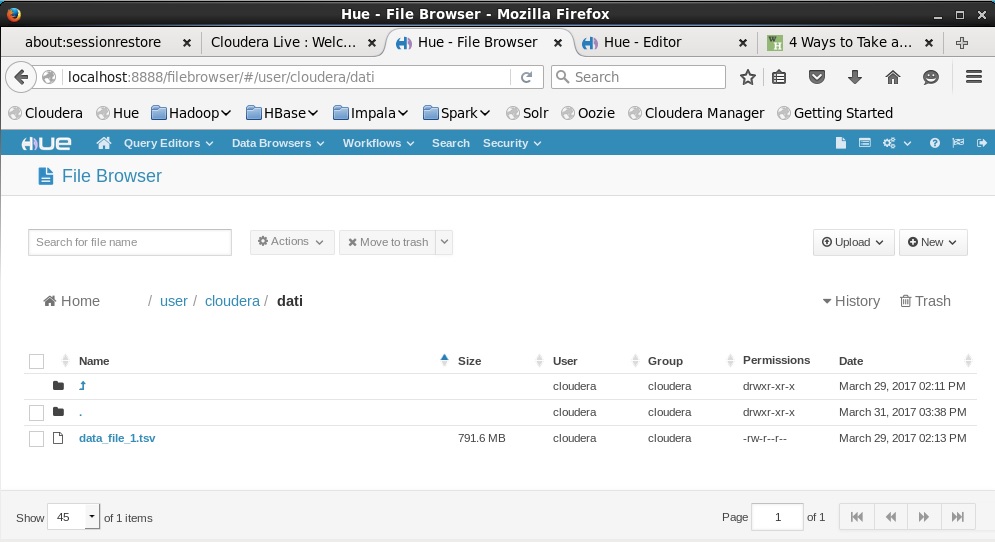
**Simulated Data Generation**

Second step has been generate data to analyze. To do this I copied your code in a python script (data\_generation.py) and created a data file (data\_file.tsv) size 792M.



**Import data on HDFS**

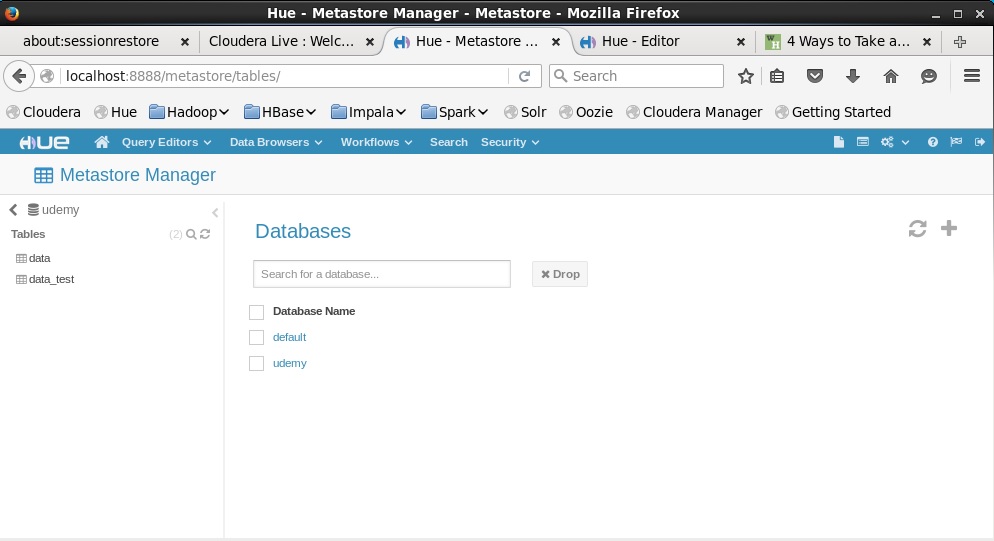
Next step has been to import simulated data into Hadoop. Reading around on the web, I finally decided the simpler and fast way to do this was use all the instruments that Cloudera VM offered, I mean **Hue** graphical interface. Hope this is a *valid* approach…



As you can see from the previous screenshot, I first used the **Hue -** **File Browser** whit the *drag and drops* functionality to upload the data file into HDFS.

**Create a Udemy DB for the analysis**

Then, I used the **Hue - Metastore** wizard to create a Udemy DB adding a table with right columns and types to store simulated data structure.



In this way I created the connection between Hadoop data and Hive to be able to run queries on data.

In the last screenshot you can see the **Hue - Editor** to write and run my queries.

Trying different queries to find the most popular courses, and accomplish with the first task, I immediately realized that 4GB of memory for the Cloudera VM was not enough. Any simple query took too much time and the VM was really busy to perform anything.

To solve the problem I decided to slightly modify your script for the data simulation, changing the limit of the loop and create only 1000 entries, just to test.

So I repeated the procedure to import this new data\_test on HDFS and I added a second table to Udemy DB to run my queries in a quickly way.

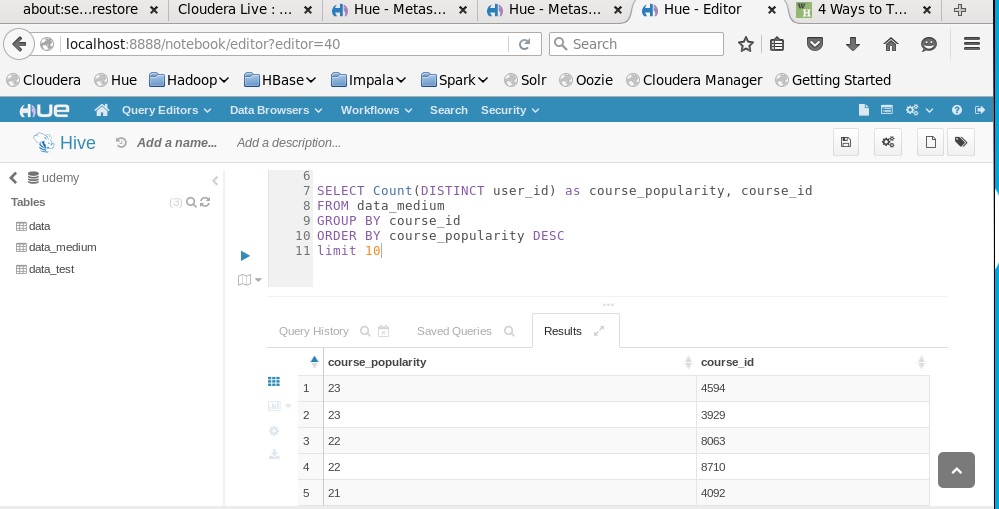
Indeed you can see on the screenshot of the **Hue – Editor**, on the left, two tables (data and data\_test ) with same type of structure but different size.

In this way, queries resulted quite fast, but the output was not so meaningful. So I tried again with a compromise between speed and data size creating a new data set with 100.000 entries.

Now, the queries results seems fine as you can see from the following screenshots.

**First task**

To find most popular courses, I tried different queries and the best one I found is the one written in the middle–top part of the editor:



SELECT Count(DISTINCT user\_id) as course\_popularity, course\_id

FROM data\_medium

GROUP BY course\_id

ORDER BY course\_popularity DESC

I think, this query works quite well.

**Data Summarization**

To see from data the type of summarization you suggested (project point 2), I think I do not still need to use UDF. I tried the query you find in the last screenshot:

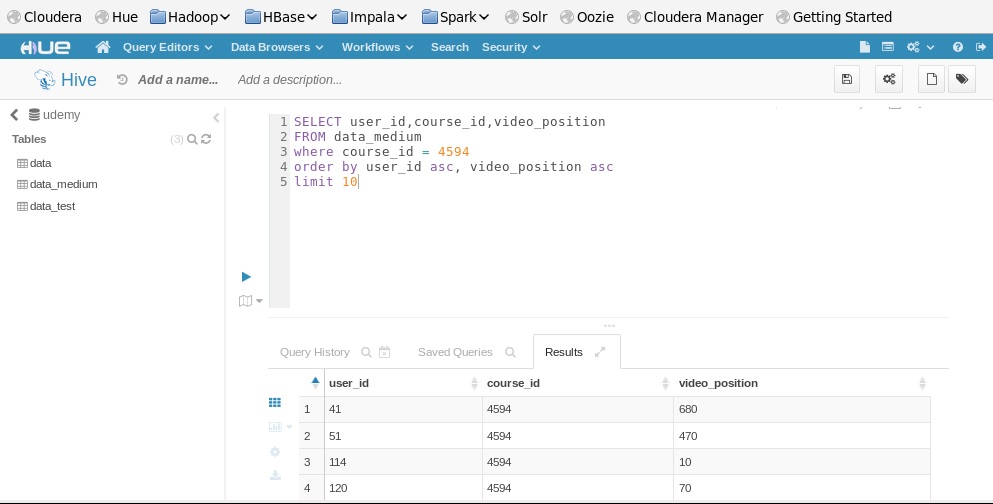
SELECT user\_id,course\_id,video\_position

FROM data\_medium

where course\_id = 4594

order by user\_id asc, video\_position asc

limit 10

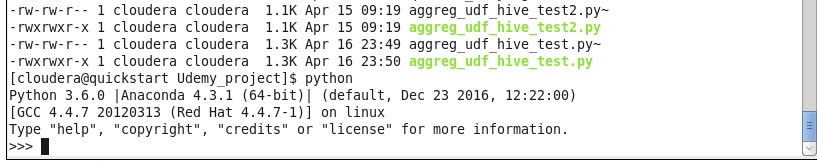


By construction there are no repeated user\_id, but I think is fine at this level to choose a particular course\_id. A possible optimization of this query could be to use a sub-query: ( select course\_id from data\_medium group by course\_id) nested in the main one with WHERE...IN or INNER JOIN.

**Python UD(A)F in Hive**

In order to aggregate data and to obtain some type of video consumption we decided to use a Python UD(A)F. That means write a python script to perform aggregation and run the script using Hive.

So, first of all I installed Anaconda 4.3 (Python 3.6 inside) on the Cloudera VM, as you can see from the following screenshot.

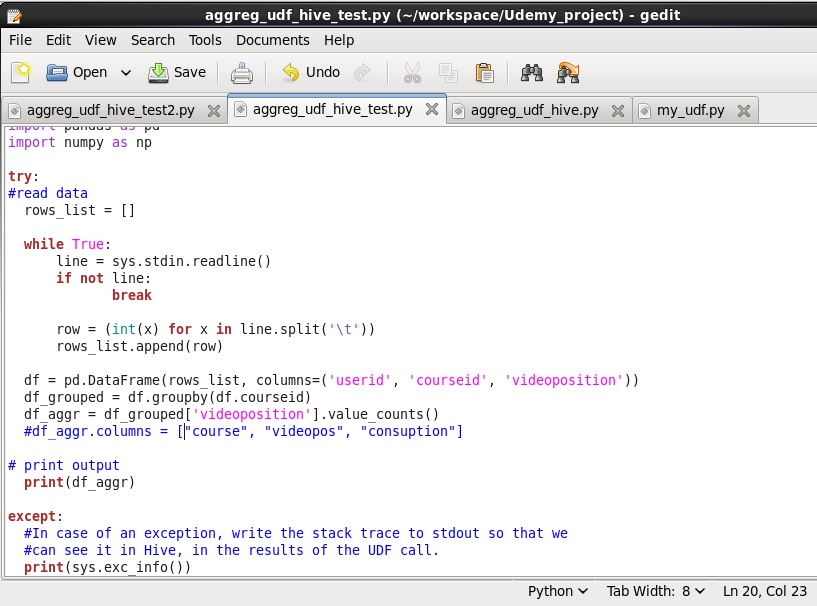


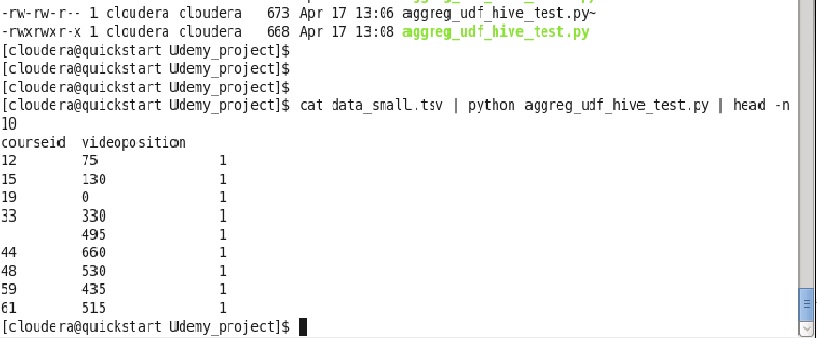
Next step has been write a Python script that read from standard input and write on standard output counting how many times a given course has been stopped at given time position. Next screenshot shows the script named: aggreg\_udf\_hive\_test.py.

I tested the script using the command:

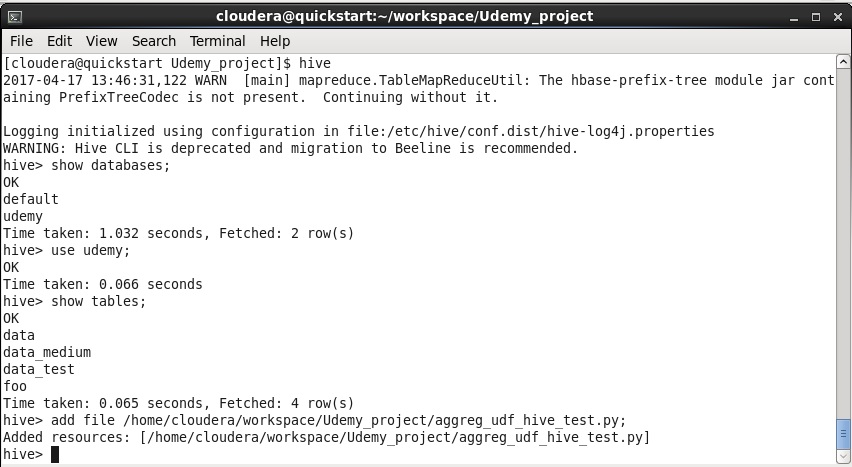
“cat data\_small.tsv | python aggreg\_udf\_hive\_test.py | head -n 10 “

on a small data sample. And it seems to work fine…





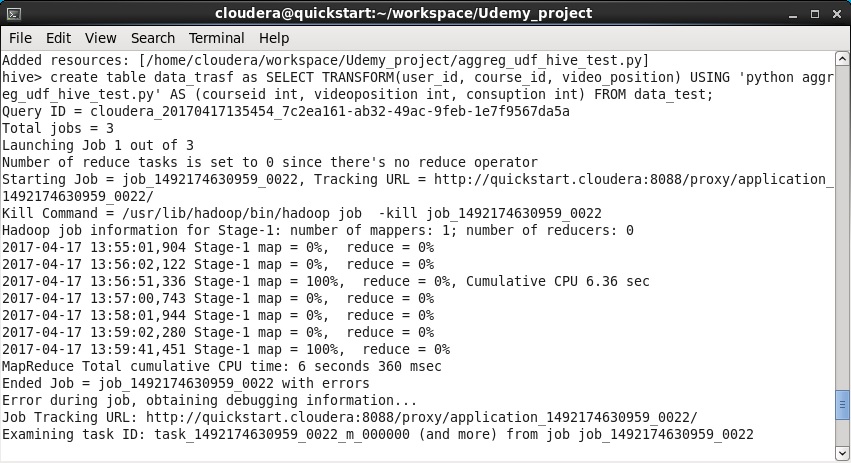
Next step has been to upload the python script into HDFS. From Hive command prompt, I first of all checked the udemy db and tables, than I added the file, as you can see from following screenshot.

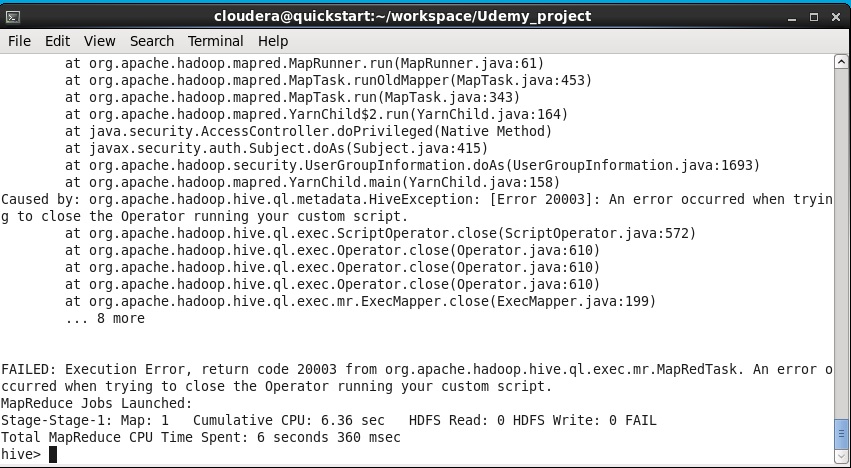


Then I tried the query to run the UDAF:

create table data\_trasf as SELECT TRANSFORM(user\_id, course\_id, video\_position) USING 'python aggreg\_udf\_hive\_test.py' AS (courseid int, videoposition int, consuption int) FROM data\_test;

But it does not work…



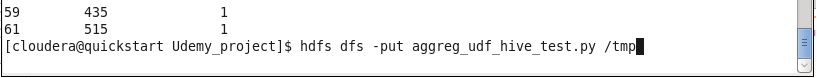


I checked this type of error on the web, looking for possible solution, but without luck, up to now.

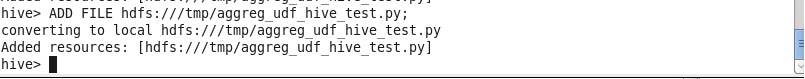
According different forum, it seems this type of error occur when the python scripts is not loaded into HDFS or it does not have the correct rights.

But I did it! And I, also, changed the right of the script with “chmod +x “ command.

I tried also to import the script in a different way using the commands:



And then from the Hive command prompt:



But running again the Hive query, this does not fix the bug!

I changed the query in order not to create the table as result, and use only the part whith SELCT (), TRANSFORM () … but with the same result.

Obviously I also try to run very simple UDF python script (the one of the example you suggested me) and Hive query but nothing change, at the moment I do not be able to run a python UDF, on my system, at all.

Sorry, I am still trying to find a solution…