

Udemy Analysis Project - Report

Author: Michela Ieva

Date: 04/17/2017

Abstract

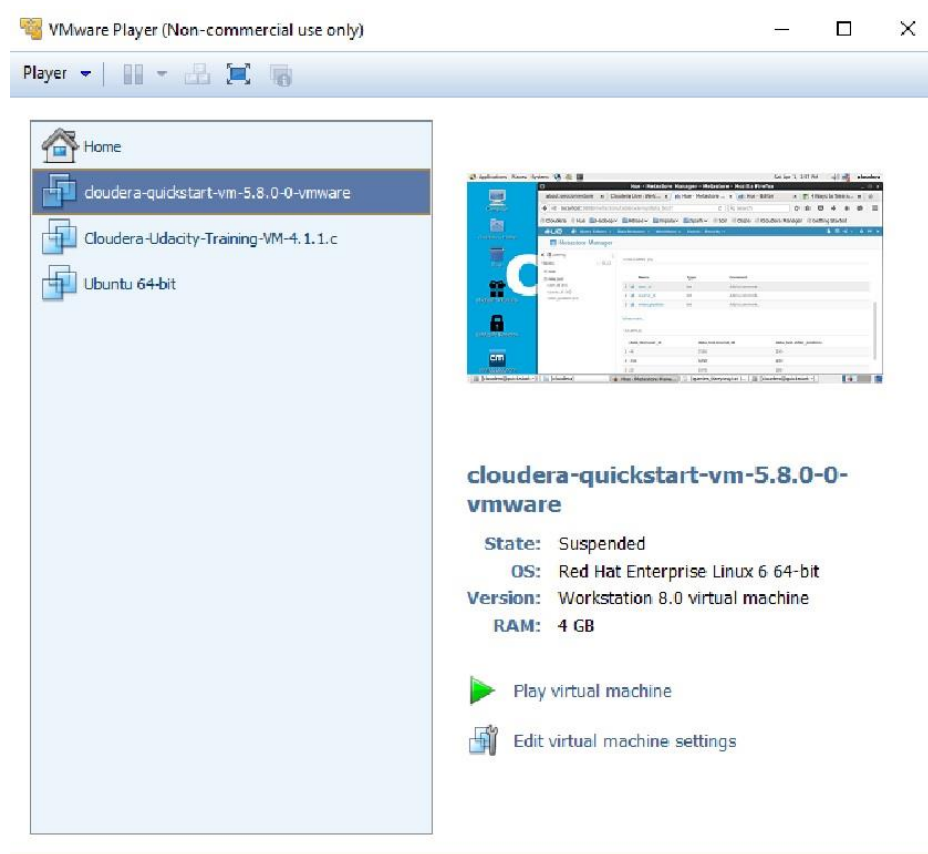
In this document I will report analysis results after a week of work.

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.



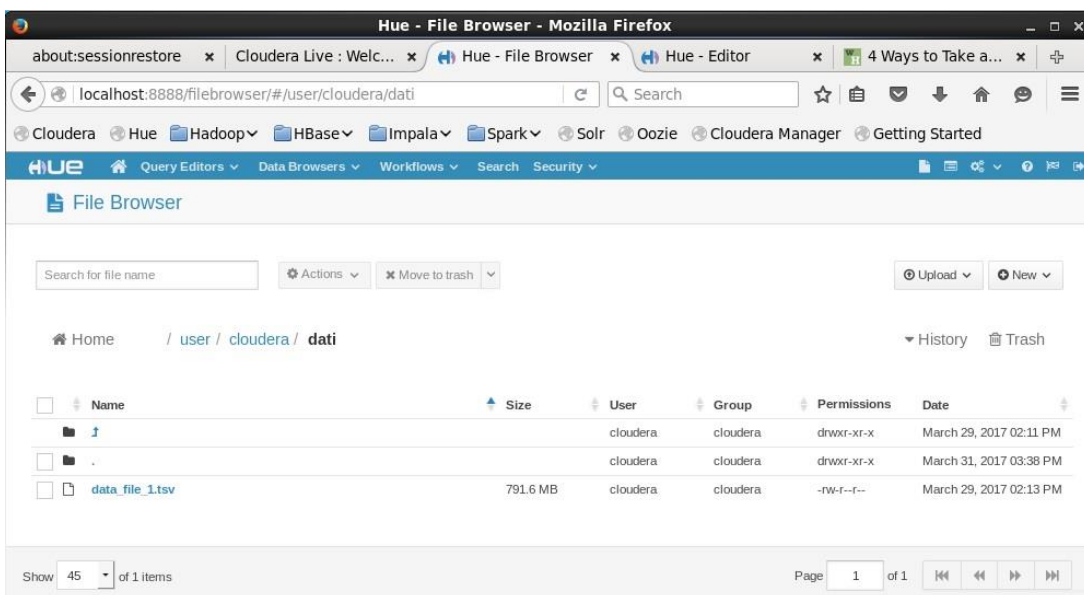
Simulated Data Generation

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.

```
cloudera@quickstart:~/workspace/Udemy_project
File Edit View Search Terminal Help
drwxr-xr-x 2 cloudera cloudera 4.0K Mar 27 14:06 Music
drwxr-xr-x 2 cloudera cloudera 4.0K Mar 27 14:06 Downloads
drwxrwxr-x 5 cloudera cloudera 4.0K Mar 27 15:33 workspace
drwxrwsr-x 9 cloudera cloudera 4.0K Mar 27 15:35 eclipse
[cloudera@quickstart ~]$ cd workspace/
[cloudera@quickstart workspace]$ ls -hlrt
total 8.0K
drwxrwxr-x 6 cloudera cloudera 4.0K Aug 10 2016 training
drwxrwxr-x 3 cloudera cloudera 4.0K Mar 31 15:25 Udemy_project
[cloudera@quickstart workspace]$ cd Udemy_project/
[cloudera@quickstart Udemy_project]$ ls -hlrt
total 792M
-rw-rw-r-- 1 cloudera cloudera 217 Mar 28 13:48 data_generation.py
-rw-rw-r-- 1 cloudera cloudera 792M Mar 28 14:19 data_file.tsv
-rw-rw-r-- 1 cloudera cloudera 98 Mar 28 14:33 info.txt
-rw-rw-r-- 1 cloudera cloudera 860 Mar 30 14:56 queries_Keeyong.txt~
-rw-rw-r-- 1 cloudera cloudera 1.1K Mar 31 15:21 queries_Keeyong.txt
-rw-rw-r-- 1 cloudera cloudera 211 Mar 31 15:23 data_test_generation.py~
-rw-rw-r-- 1 cloudera cloudera 212 Mar 31 15:25 data_test_generation.py
-rw-rw-r-- 1 cloudera cloudera 13K Mar 31 15:25 data_small.tsv
[cloudera@quickstart Udemy_project]$
```

Import data on HDFS

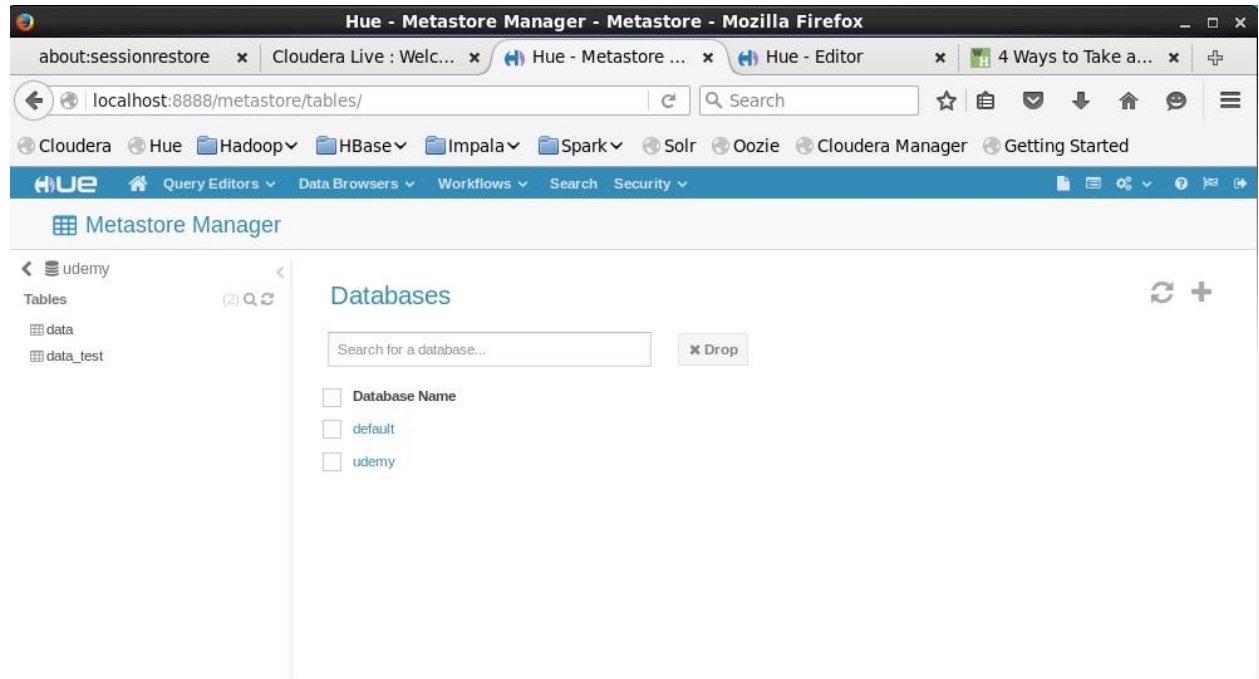
Next step has been to import simulated data into Hadoop. Reading around on the web, I finally decided the simpler and fast way to do this was use all the instruments that Cloudera VM offered, I mean Hue graphical interface. Hope this is a *valid* approach...



As you can see from the previous screenshot, I first used the **Hue - File Browser** with 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.

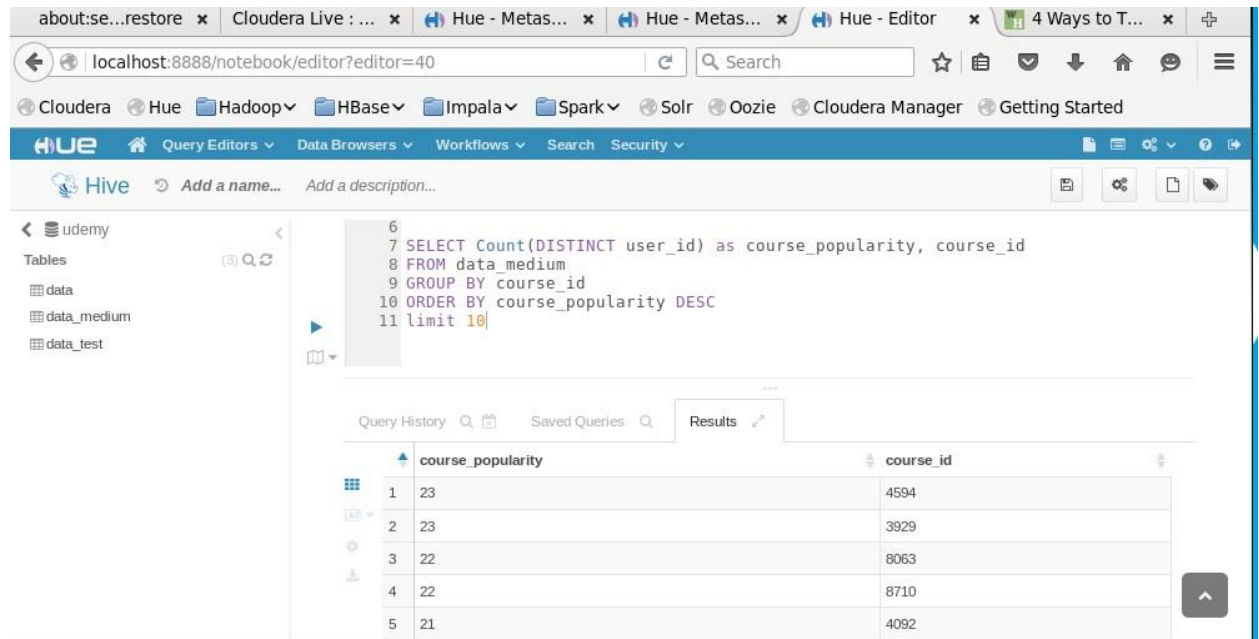
So I repeated the procedure to import this new data_test on HDFS and I added a second table to Udemy DB to run my queries in a quickly way. Indeed you can see on the screenshot of the **Hue - Editor**, on the left, two tables (data and data_test) with same type of structure but different size.

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:



The screenshot shows the Hue web interface with a Hive query editor. The query is as follows:

```
6  
7 SELECT Count(DISTINCT user_id) as course_popularity, course_id  
8 FROM data_medium  
9 GROUP BY course_id  
10 ORDER BY course_popularity DESC  
11 limit 10
```

The results are displayed in a table with two columns: **course_popularity** and **course_id**. The results are ordered by popularity in descending order.

	course_popularity	course_id
1	23	4594
2	23	3929
3	22	8063
4	22	8710
5	21	4092

```
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
```

The screenshot shows the Hue web interface for Cloudera. The top navigation bar includes links for Cloudera, Hue, Hadoop, HBase, Impala, Spark, Solr, Oozie, Cloudera Manager, and Getting Started. The main interface is divided into three sections: a left sidebar for navigation, a central query editor, and a bottom results section.

Left Sidebar: Shows the 'udemy' database with tables 'data', 'data_medium', and 'data_test'.

Central Query Editor: Contains the following SQL query:

```
1 SELECT user_id, course_id, video_position
2 FROM data_medium
3 where course_id = 4594
4 order by user_id asc, video_position asc
5 limit 10
```

Bottom Results Section: Displays the query results in a table format.

	user_id	course_id	video_position
1	41	4594	680
2	51	4594	470
3	114	4594	10
4	120	4594	70

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.

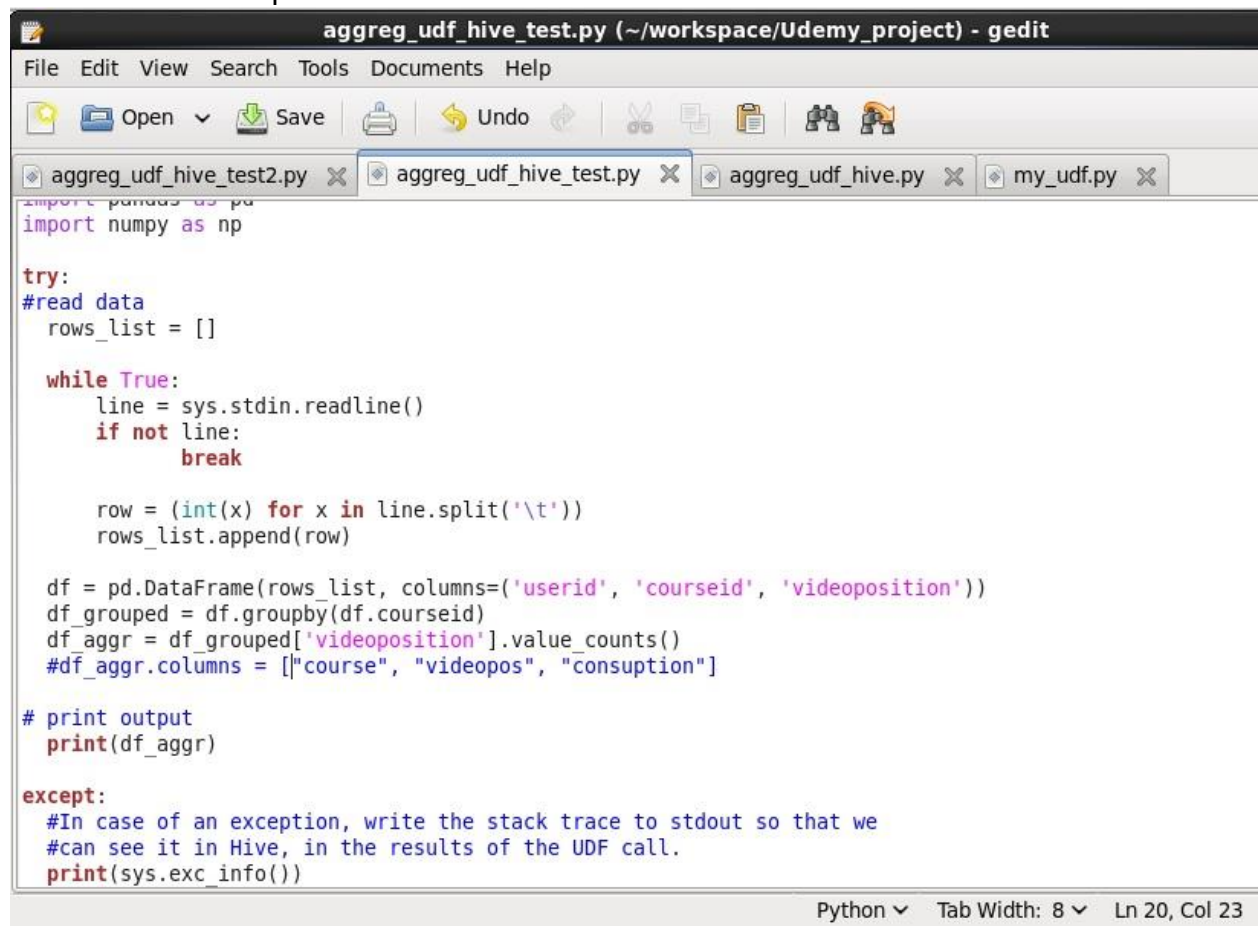
```
-rw-rw-r-- 1 cloudera cloudera 1.1K Apr 15 09:19 aggreg_udf_hive_test2.py~
-rwxrwxr-x 1 cloudera cloudera 1.1K Apr 15 09:19 aggreg_udf_hive_test2.py
-rw-rw-r-- 1 cloudera cloudera 1.3K Apr 16 23:49 aggreg_udf_hive_test.py~
-rwxrwxr-x 1 cloudera cloudera 1.3K Apr 16 23:50 aggreg_udf_hive_test.py
[cloudera@quickstart Udemey_project]$ python
Python 3.6.0 [Anaconda 4.3.1 (64-bit)] (default, Dec 23 2016, 12:22:00)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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...



```
aggreg_udf_hive_test.py (~/workspace/Udemy_project) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
aggreg_udf_hive_test2.py aggreg_udf_hive_test.py aggreg_udf_hive.py my_udf.py
import pandas as pd
import numpy as np

try:
    #read data
    rows_list = []

    while True:
        line = sys.stdin.readline()
        if not line:
            break

        row = (int(x) for x in line.split('\t'))
        rows_list.append(row)

    df = pd.DataFrame(rows_list, columns=('userid', 'courseid', 'videoposition'))
    df_grouped = df.groupby(df.courseid)
    df_aggr = df_grouped['videoposition'].value_counts()
    #df_aggr.columns = ["course", "videopos", "consumption"]

    # print output
    print(df_aggr)

except:
    #In case of an exception, write the stack trace to stdout so that we
    #can see it in Hive, in the results of the UDF call.
    print(sys.exc_info())

Python Tab Width: 8 Ln 20, Col 23
```

```
-rw-rw-r-- 1 cloudera cloudera 673 Apr 17 13:06 aggreg_udf_hive_test.py~
-rwxrwxr-x 1 cloudera cloudera 668 Apr 17 13:08 aggreg_udf_hive_test.py
[cloudera@quickstart Udemy_project]$
[cloudera@quickstart Udemy_project]$
[cloudera@quickstart Udemy_project]$
[cloudera@quickstart Udemy_project]$ cat data_small.tsv | python aggreg_udf_hive_test.py | head -n
10
courseid videoposition
12      75          1
15     130          1
19       0          1
33     330          1
      495          1
44     660          1
48     530          1
59     435          1
61     515          1
[cloudera@quickstart Udemy_project]$
```

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.


```
cloudera@quickstart:~/workspace/Udemy_project
File Edit View Search Terminal Help
[cloudera@quickstart Udemy_project]$ hive
2017-04-17 13:46:31,122 WARN [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing PrefixTreeCodec is not present. Continuing without it.

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
udemy
Time taken: 1.032 seconds, Fetched: 2 row(s)
hive> use udemy;
OK
Time taken: 0.066 seconds
hive> show tables;
OK
data
data_medium
data_test
foo
Time taken: 0.065 seconds, Fetched: 4 row(s)
hive> add file /home/cloudera/workspace/Udemy_project/aggreg_udf_hive test.py;
Added resources: [/home/cloudera/workspace/Udemy_project/aggreg_udf_hive_test.py]
hive>
```

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, consumption
int) FROM data_test;
```

But it does not work...

```
cloudera@quickstart:~/workspace/Udemy_project
File Edit View Search Terminal Help
Added resources: [/home/cloudera/workspace/Udemy_project/aggreg_udf_hive_test.py]
hive> 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, consumption int) FROM data_test;
Query ID = cloudera_20170417135454_7c2ea161-ab32-49ac-9feb-1e7f9567da5a
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1492174630959_0022, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1492174630959_0022/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1492174630959_0022
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2017-04-17 13:55:01,904 Stage-1 map = 0%, reduce = 0%
2017-04-17 13:56:02,122 Stage-1 map = 0%, reduce = 0%
2017-04-17 13:56:51,336 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.36 sec
2017-04-17 13:57:00,743 Stage-1 map = 0%, reduce = 0%
2017-04-17 13:58:01,944 Stage-1 map = 0%, reduce = 0%
2017-04-17 13:59:02,280 Stage-1 map = 0%, reduce = 0%
2017-04-17 13:59:41,451 Stage-1 map = 100%, reduce = 0%
MapReduce Total cumulative CPU time: 6 seconds 360 msec
Ended Job = job_1492174630959_0022 with errors
Error during job, obtaining debugging information...
Job Tracking URL: http://quickstart.cloudera:8088/proxy/application_1492174630959_0022/
Examining task ID: task_1492174630959_0022_m_000000 (and more) from job job_1492174630959_0022
```

```
cloudera@quickstart:~/workspace/Udemy_project
File Edit View Search Terminal Help
at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:61)
at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:453)
at org.apache.hadoop.mapred.MapTask.run(MapTask.java:343)
at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:415)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1693)
at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)
Caused by: org.apache.hadoop.hive.ql.metadata.HiveException: [Error 20003]: An error occurred when trying
g to close the Operator running your custom script.
at org.apache.hadoop.hive.ql.exec.ScriptOperator.close(ScriptOperator.java:572)
at org.apache.hadoop.hive.ql.exec.Operator.close(Operator.java:610)
at org.apache.hadoop.hive.ql.exec.Operator.close(Operator.java:610)
at org.apache.hadoop.hive.ql.exec.Operator.close(Operator.java:610)
at org.apache.hadoop.hive.ql.exec.mr.ExecMapper.close(ExecMapper.java:199)
... 8 more

FAILED: Execution Error, return code 20003 from org.apache.hadoop.hive.ql.exec.mr.MapRedTask. An error o
ccurred when trying to close the Operator running your custom script.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 6.36 sec HDFS Read: 0 HDFS Write: 0 FAIL
Total MapReduce CPU Time Spent: 6 seconds 360 msec
hive>
```

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:

```
59      435      1
61      515      1
[cloudera@quickstart Udemy_project]$ hdfs dfs -put aggreg_udf_hive_test.py /tmp
```

And then from the Hive command prompt:

```
hive> ADD FILE hdfs:///tmp/aggreg_udf_hive_test.py;
converting to local hdfs:///tmp/aggreg_udf_hive_test.py
Added resources: [hdfs:///tmp/aggreg_udf_hive_test.py]
hive>
```

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 with SELECT (), 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...