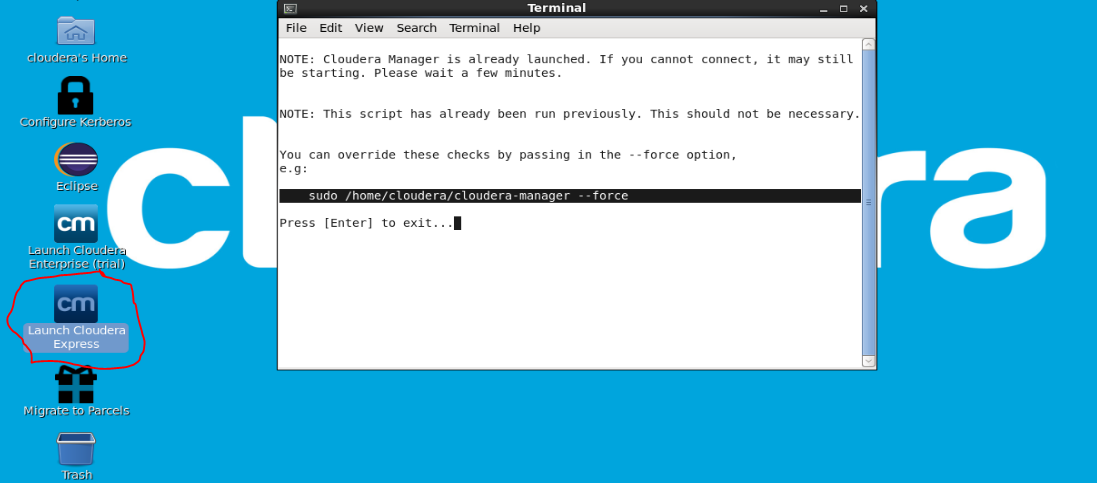
Cloudera QuickStart Guide

# What I’ve done with the QuickStart Machine:

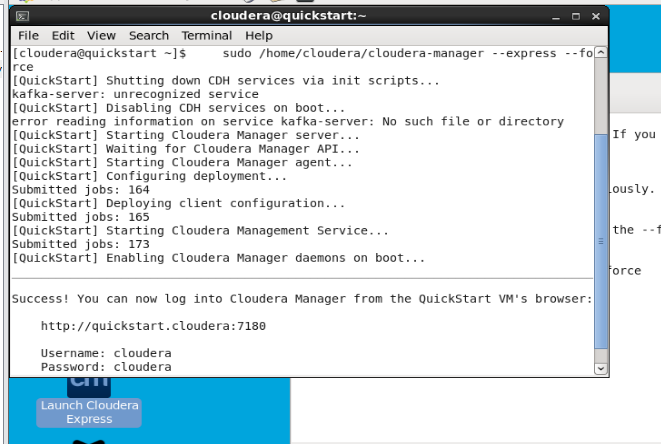
I have Ran the Cloudera QuickStart VM as advised by Trevor to be able to run Hive queries and partitioning the date column into folders on the HDFS Directory.

# 1. Run the Cloudera Express Manager

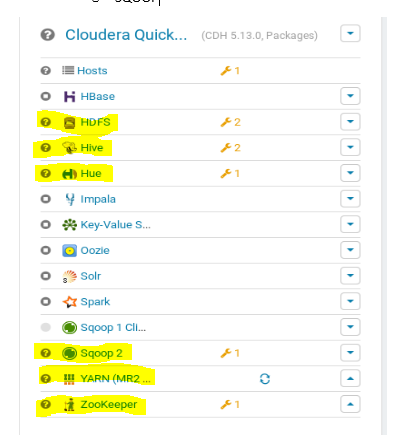
* I ran the cloudera quickstart express command on terminal in order to start the using Cloudera Manager and run the following command highlighted in terminal.



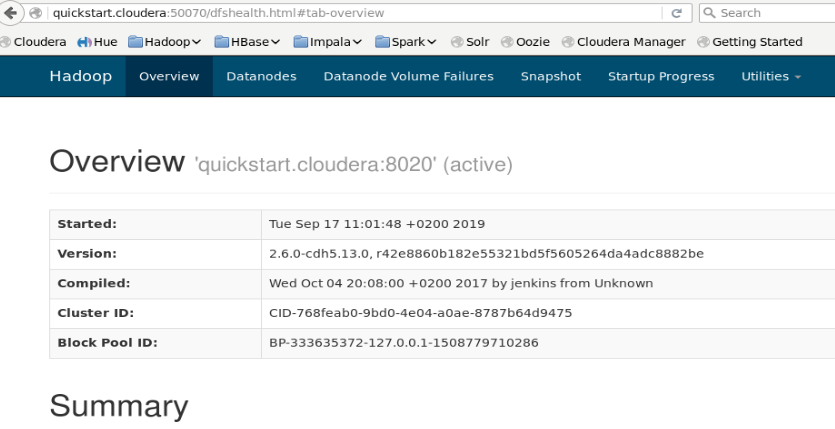
* Just open the link after the services have completed running.



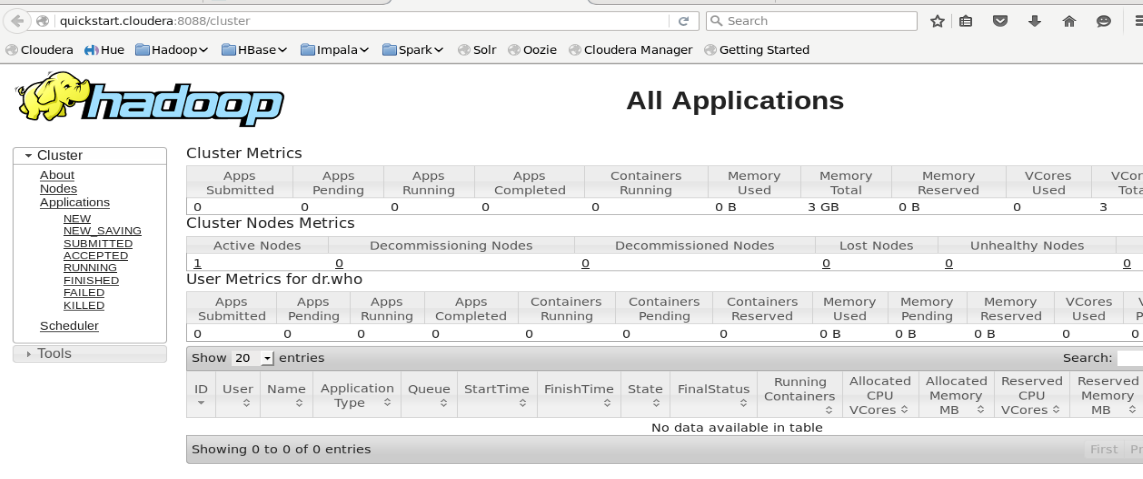
* Started the following services in order to use Hive
  + HDFS
  + SQOOP
  + YARN
  + ZooKeeper
  + Hive
  + Hue (I use this as a graphical User interface for Hive)

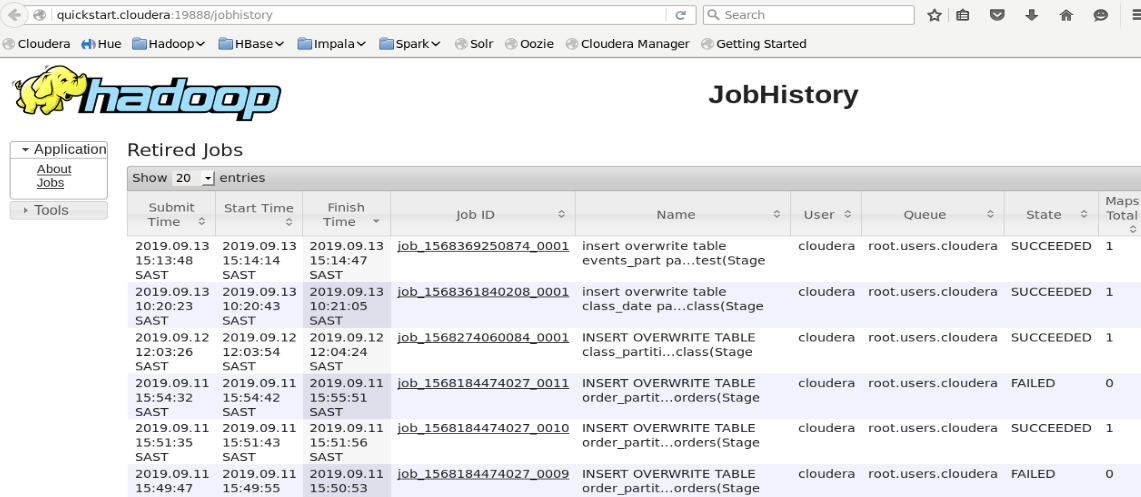


* After this, I could specify the following Services running by accessing their GUI. Cannot specify via command line as it would display all the services as failed, this has been specified from documents and the services can still work as it is a test VM.
  + I can access Namenode from the HDFS Service

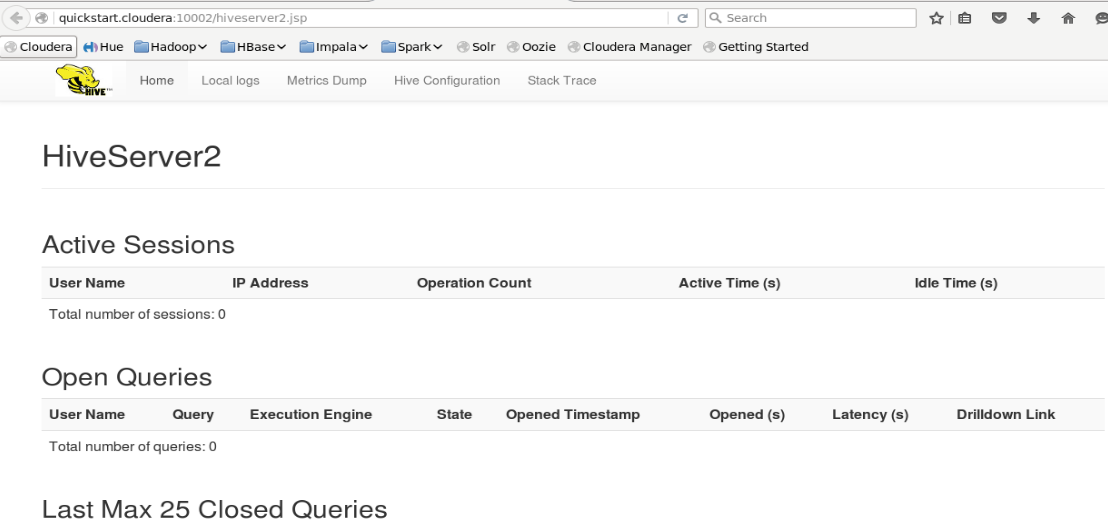


* + I can access the YARN UI of both Resource Manager and History Manager

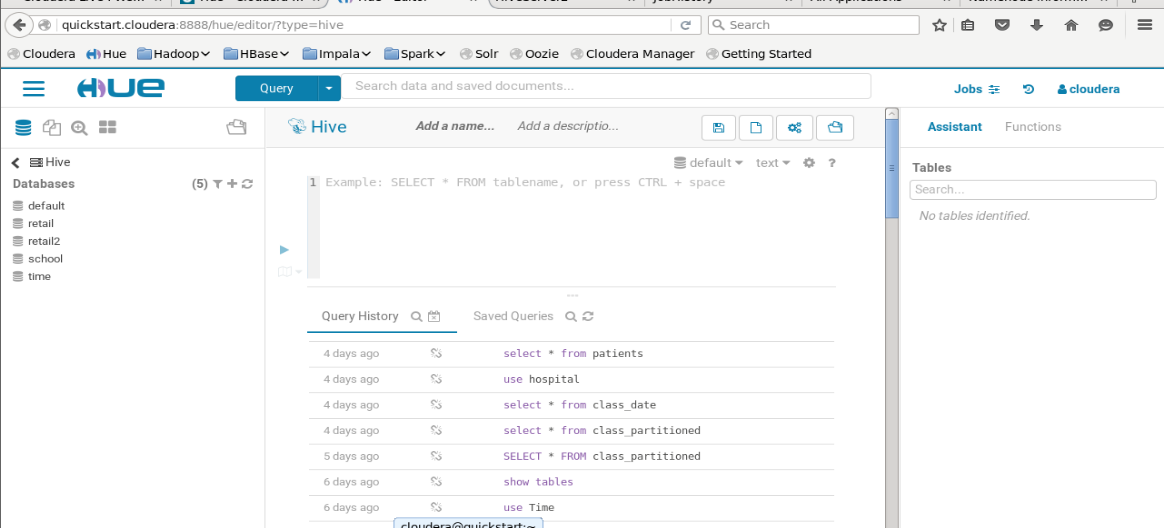




* + I can access the HiveServer2 UI from the Hive Service



* + I can access the HUE UI from HUE Service.

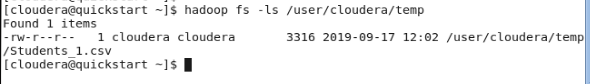


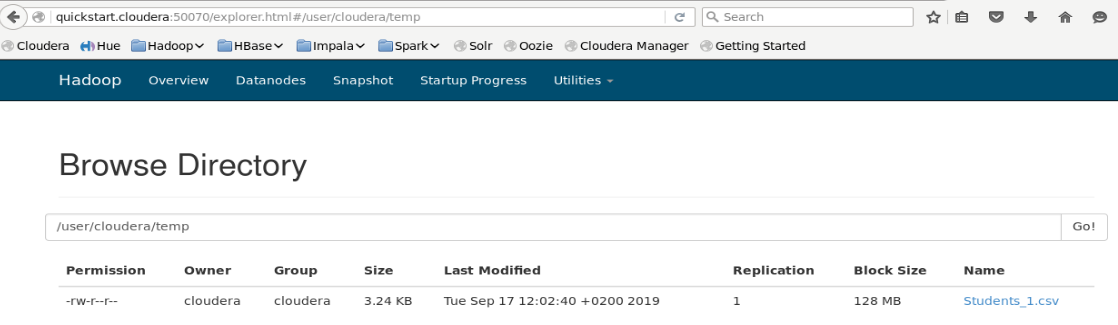
# 2. Copying or Importing of tables from files databases into HDFS.

## Importing tables from folders

I’d first like to explain how I go about importing my own created CSV files into HDFS.

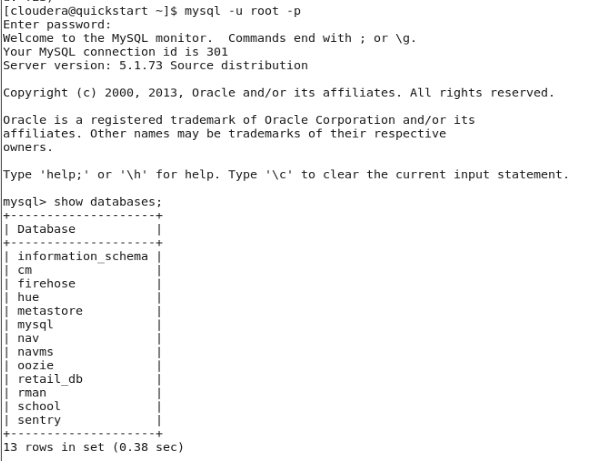
* I first create a folder in HDFS called temp. I then used the following command to create the folder
  + hadoop fs -mkdir /user/cloudera/temp
* I then copied one of my own CSV file called Students\_1.csv that I created from it’s source dir into the targerted HDFS dir
  + hadoop fs -put /home/cloudera/Downloads/Students\_1.csv /user/cloudera/temp
* I can also confirm that the file exists in HDFS





## Importing files from Database

I’d first like to confirm that there already is a MySql instance installed and running in this QuickStart VM.

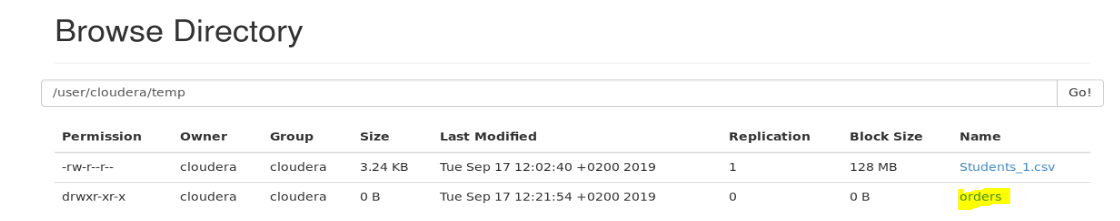


I use this for practice importing tables into HDFS.

I will import a table from the retail\_db database using the sqoop command. The texts rounded in square brackets are the specs of my machine

* sqoop import --connect jdbc:mysql://[192.168.211.132]/retail\_db --username root -P --table orders --target-dir /user/cloudera/temp/orders
* It has completed.



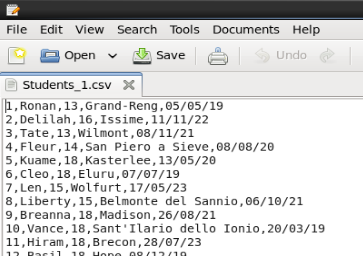


# Hive

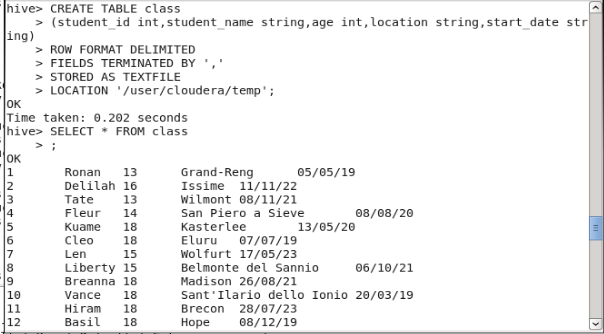
After this I created new tables and external tables on Hive. Since I’ve already copied or imported tables/files into HDFS, I was able to use Hive and create tables from the imported/copied data. I’ll use both the Students\_1.csv file and the orders table that I imported from the MySql database.

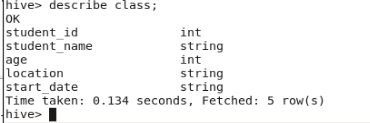
### Students\_1.csv

* This is the information that I’ve created inside the file. I didn’t include any column names as it would disrupt my partitioning.

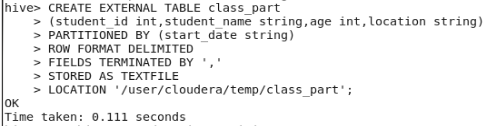


* I’ve created a database called Temp and also a created table called class.





* Now since the table has been created from the Students\_1.csv, I will try partitioning it into tables. However, I tend to struggle with the format of the date. Currently the date is formatted as “dd/mm/yy”. I will partition it with the proper format of “yyyy/mm/dd” once I create an entirely new csv and table from scratch.
  + I first created an external table called “class\_part”



* + I then prepared the table to be set for dynamic partitioning with the following query.

SET hive.exec.dynamic.partition = true;

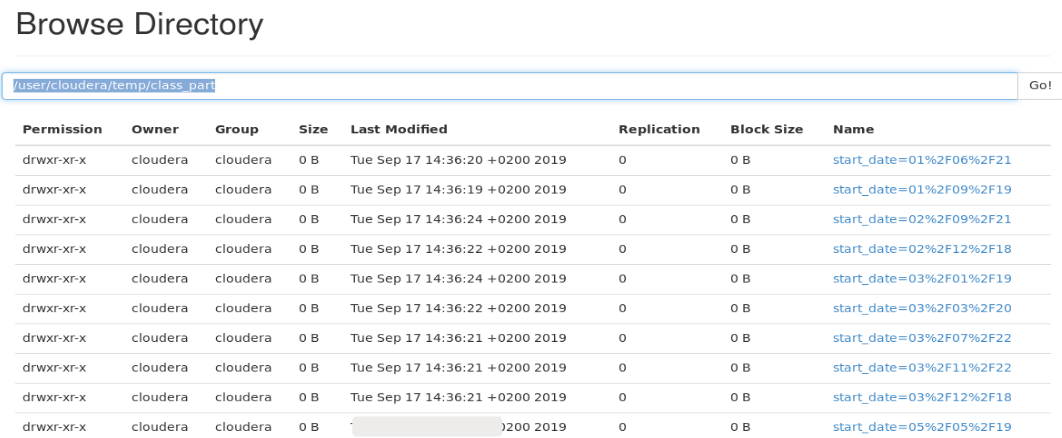
SET hive.exec.dynamic.partition.mode = nonstrict;



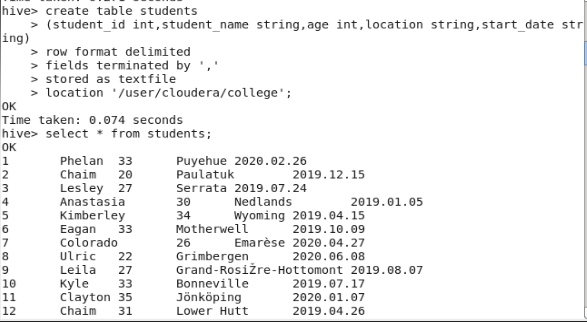
* + I ended up loading all the information form the existing “class” table

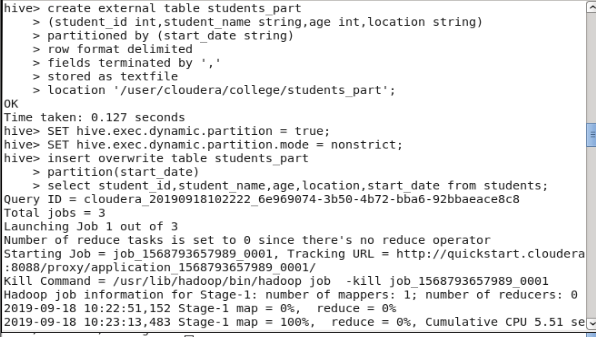


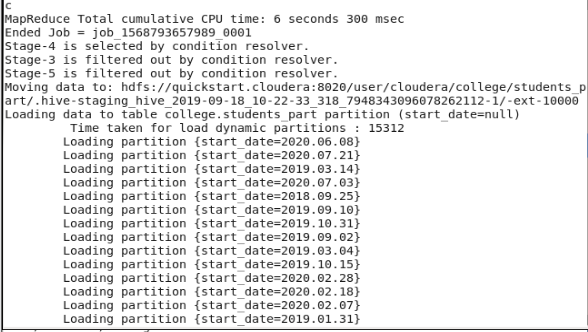
* + After loading the data into the class\_part table, the partitions were created in the HDFS folder “/user/cloudera/temp/class\_part”.

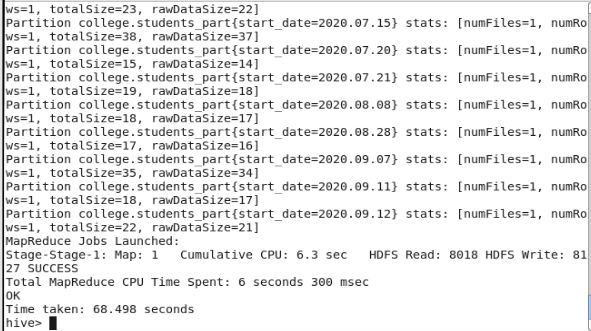


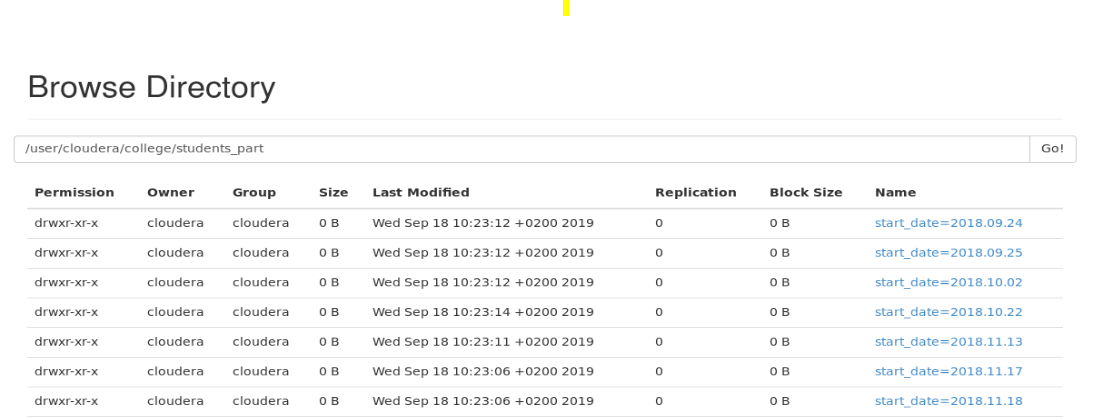
* + As you can see, it I was able to partition, however the format of the folder names is wrong. So I’m currently trying to resolve this query at the moment.
  + Date format partition issue has been resolved, I only set the datatype of the date into string and used a period instead of a forward slash.

\



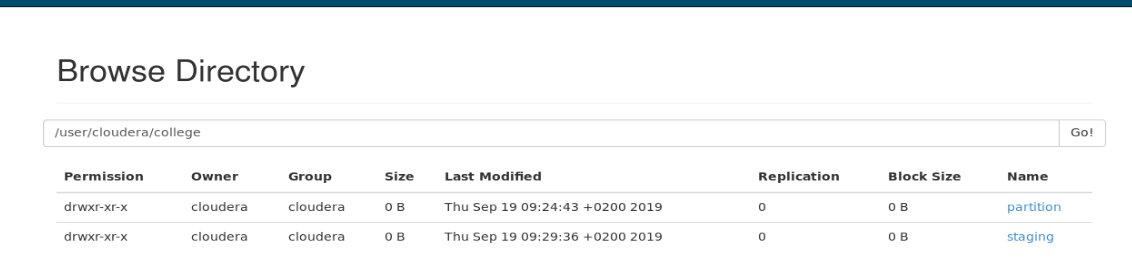






# Things I’ve recently discovered about partitioning tables 19/09/2019:

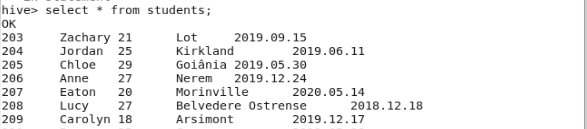
* I’ve managed to automatically load data from files into Hive by creating a folder in HDFS which I called “staging” and a second folder called “partition” to which the staging folder handles all the files which will are loaded into it to immediately load the data information into a staging table in hive which I called students.

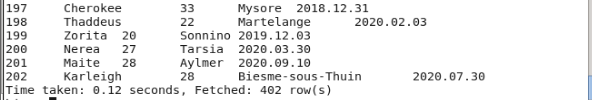


* I then have to data from the staging table into the partitioned table in order to update the partitioned table

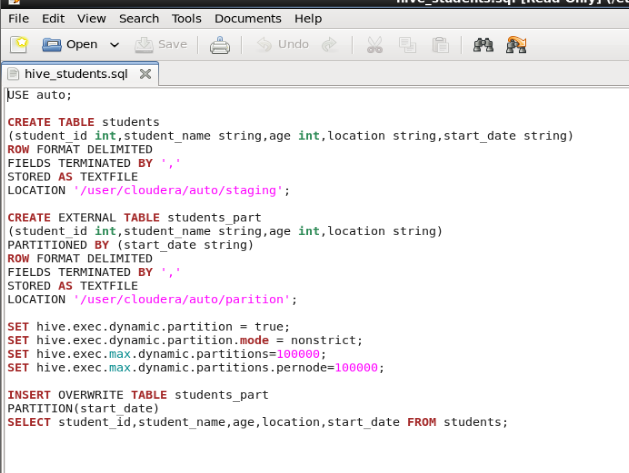


* Hive partitions are set to max out to 100 partitions. So I’ve set the maximum to exceed 100 by running these SETS:
  + SET hive.exec.max.dynamic.partitions=100000;
  + SET hive.exec.max.dynamic.partitions.pernode=100000;
* I’ve also managed to copy multiple files into hdfs manually from a folder called source in the local downloads directory which I created.
  + hadoop fs -put /home/cloudera/Downloads/source/\* /user/cloudera/college/staging
  + To which I now have seemed to have loaded about 300 records into the staging table and simply loaded data from the staging table into the partitioned table





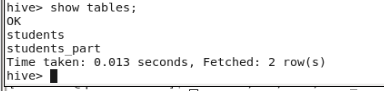
* Was also able to create a Hive.SQL script to simply create the staging table and also create the partitioned table whilst and running the necessary SETs to load data into the partitioned table

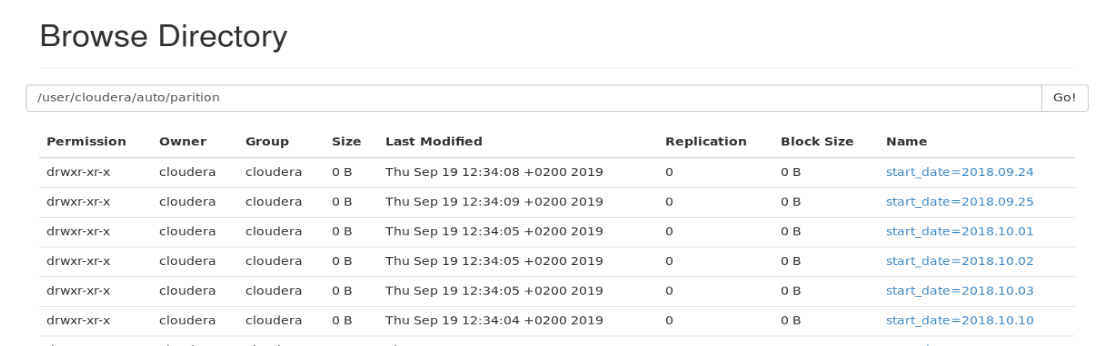


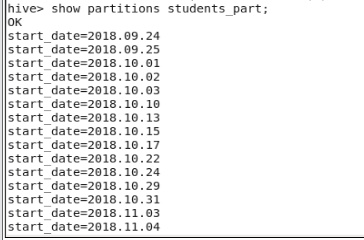
* Ran this command to run the Hive query
  + hive -f /etc/hive/hive\_students.sql



* The tables were created and the Partition was able to run and also loaded the data.







Testing Sample files

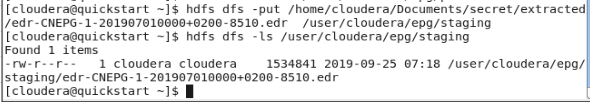
Tshifhiwa has provided me with some of the data that we collect to test the Hive Partition.

Here are the steps that I’ve done to see if it works.

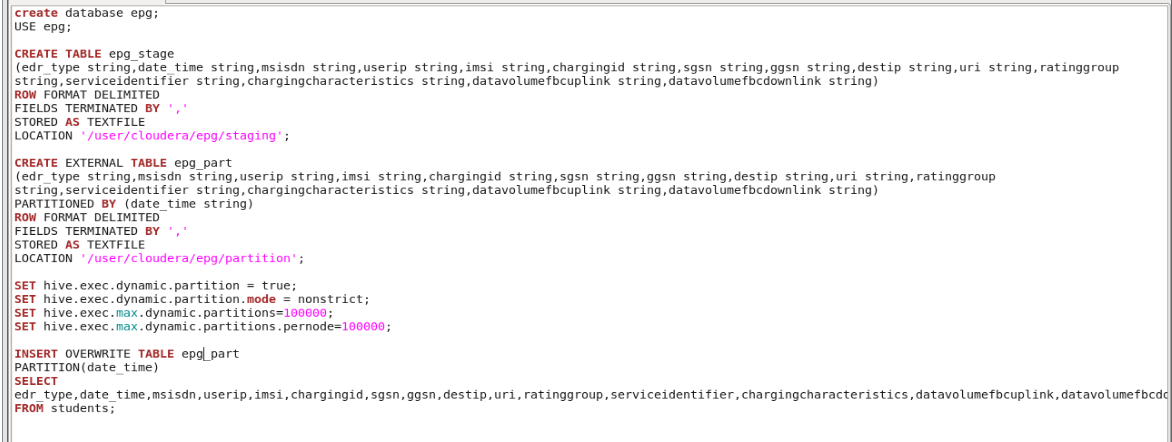
* I created the following hdfs directories.



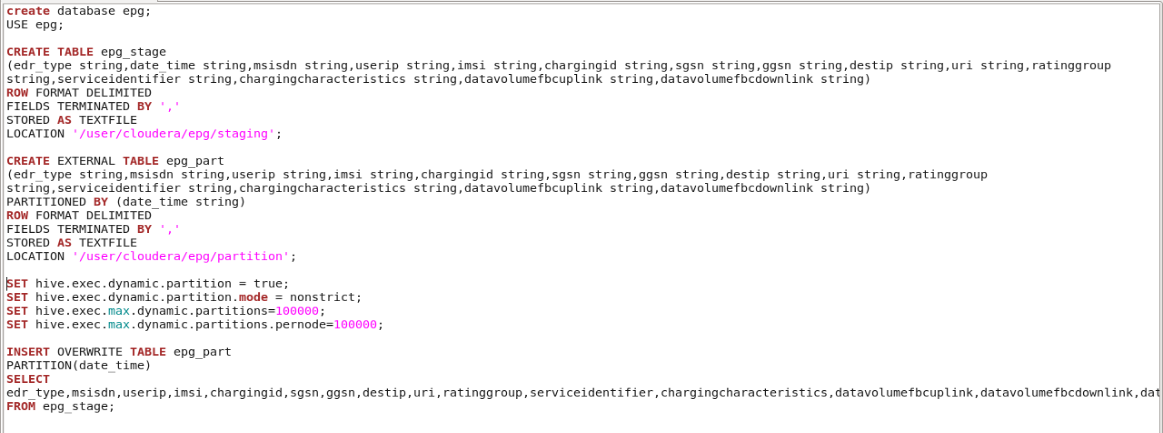
* Added one of the epg files to the following directory



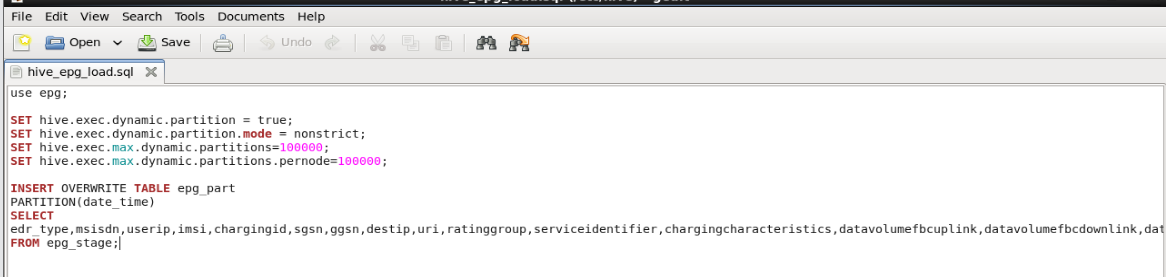
* Created the following Hive script.



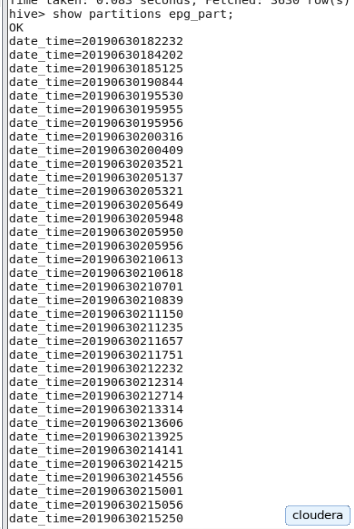
* I created 2 scripts.
  + One is to do all the tasks which are
    - Create a database
    - Use the database
    - Create the staging table
    - Create the partitioning table
    - Set the partition rules
    - Load the data

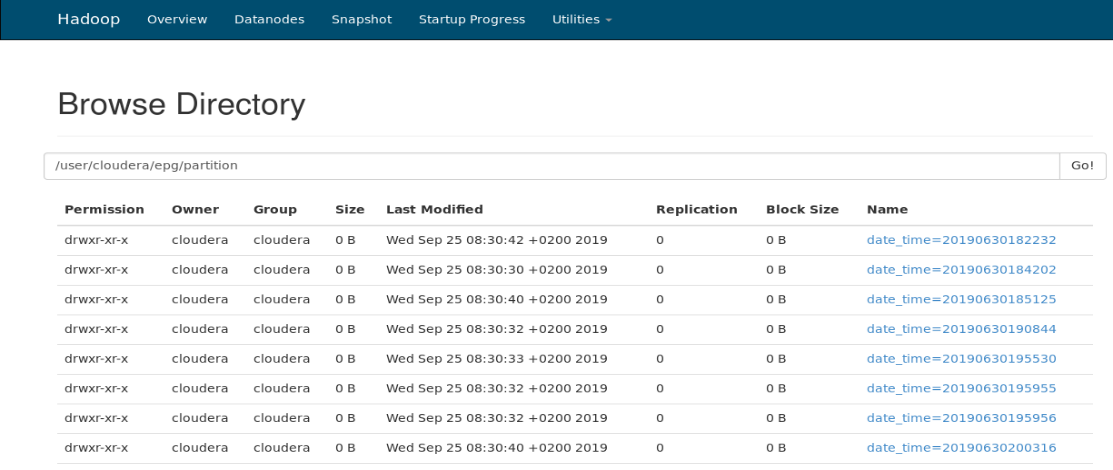


* + The other script simply loads the data from the staging table to the partitioned table.

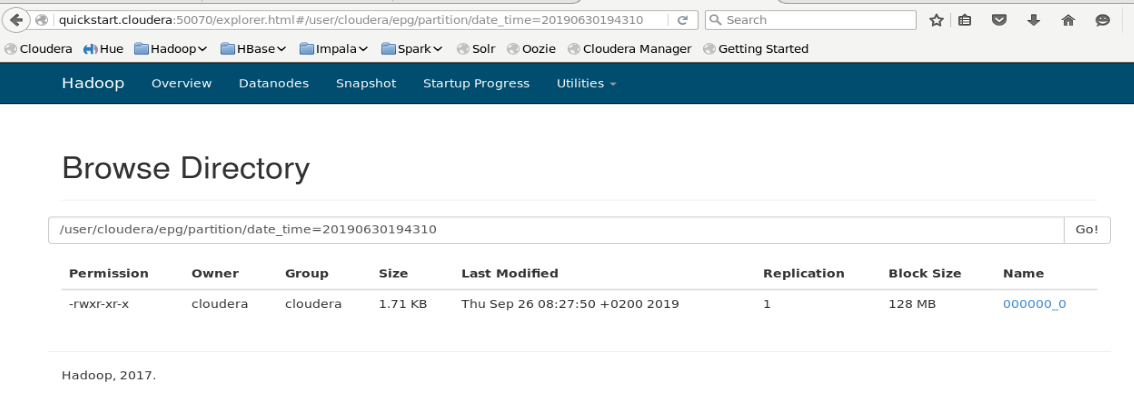


* NB!! Make sure that the partitioned column is the last item on the select query when loading the data.
* Partition has been created

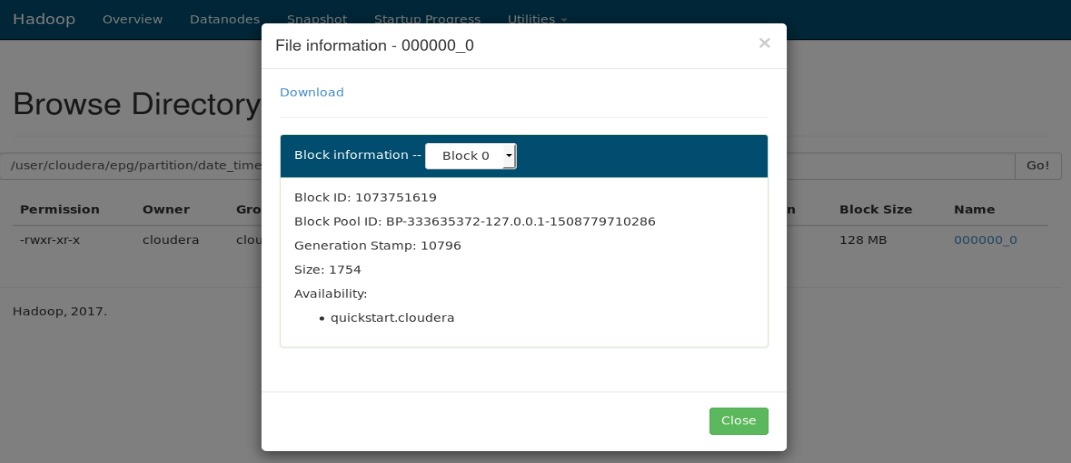


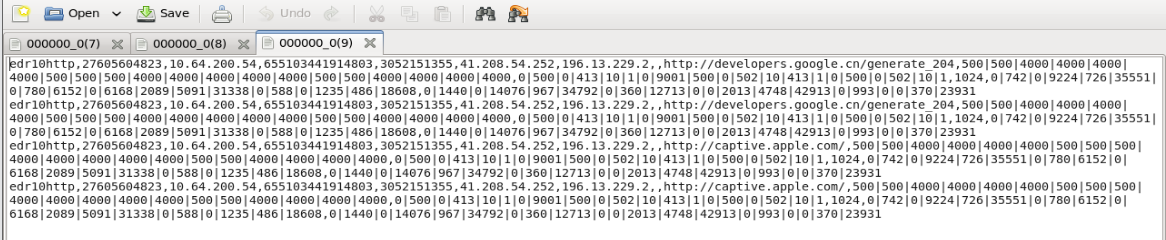


* As I click on one of the partitioned folders, it will show a file

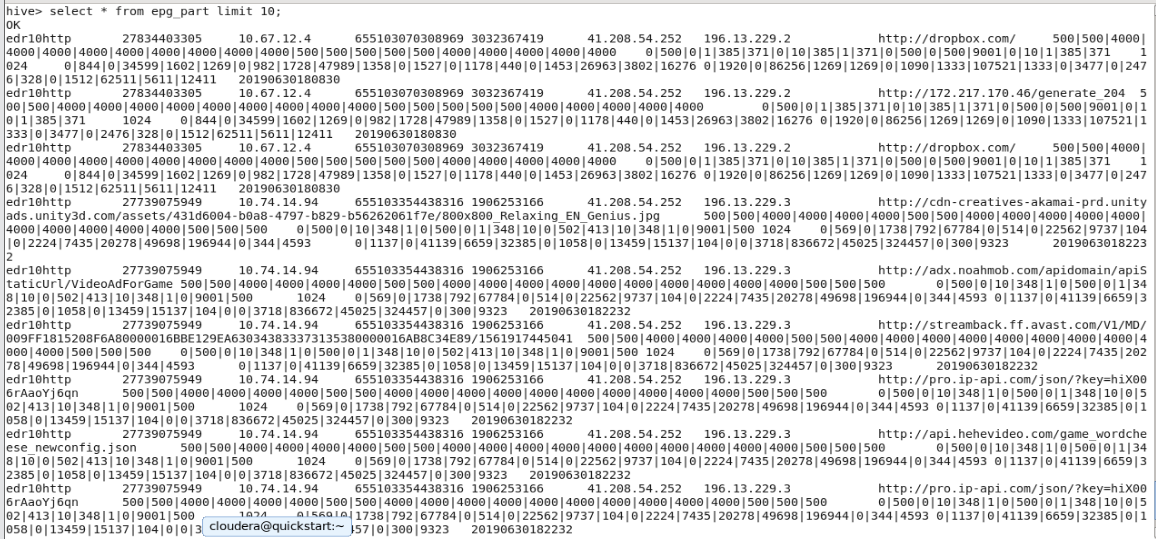


* As I try to open this file, it will prompt me to download the file and once I save it and open the file, it will display all the info regarding the selected time.





* The date\_time column, which is the partitioned column, will be displayed as the last column in the Partitioned table.

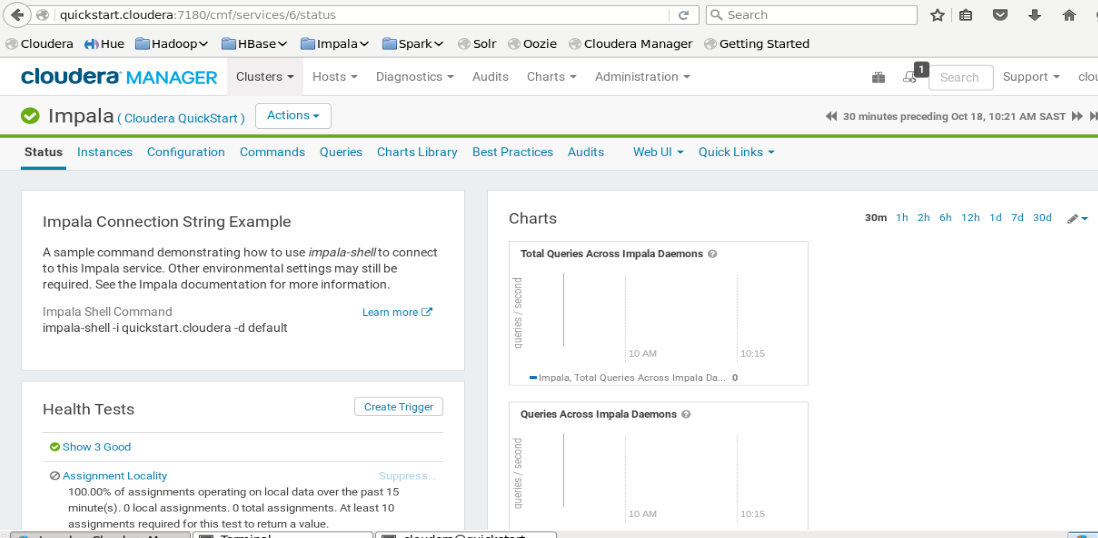


Impala

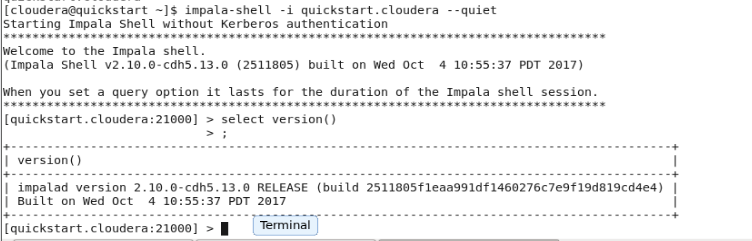
# What I’ve discovered using Impala

My Understanding of Impala is that it can query live stream data from hdfs without the need of mapreduce, meaning it won’t need to use YARN. It is also meant to be faster than Hive in terms of querying data, however it is not fault tolerant, meaning if a query failed, it won’t continue to run and that would mean you’ll need to run the entire query from start. Hive is fault tolerant.

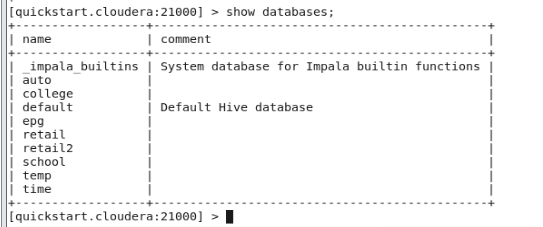
* Ran the Service on Cloudera Manager



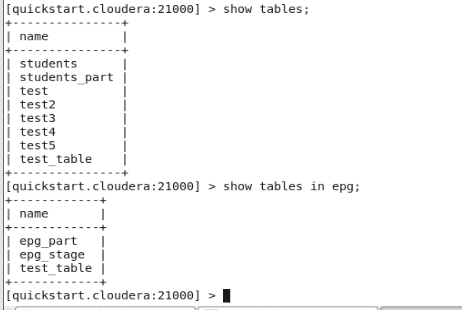
* Also logged in the the Command Line Interface of the Impala Service.



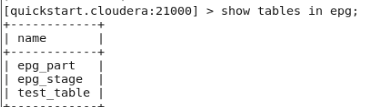
* Viewed the databases, so far all the databases that are displayed were created from Hive as Impala is able to us both it’s own databases and the Hive databases

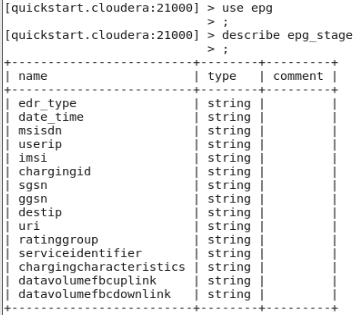


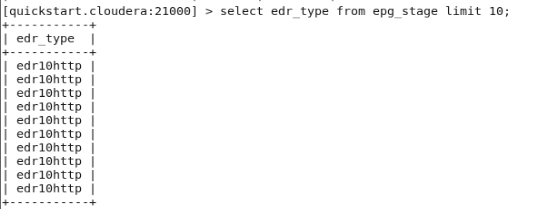
* Can even view the tables which I’ve created from Hive.

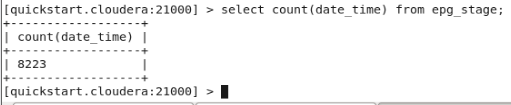


* Can even run Hive syntax queries on Impala.









* I created a test database and table





