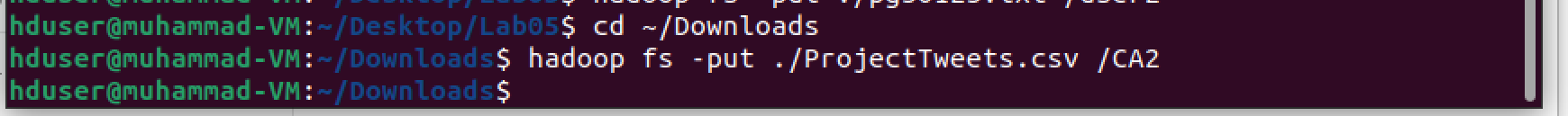
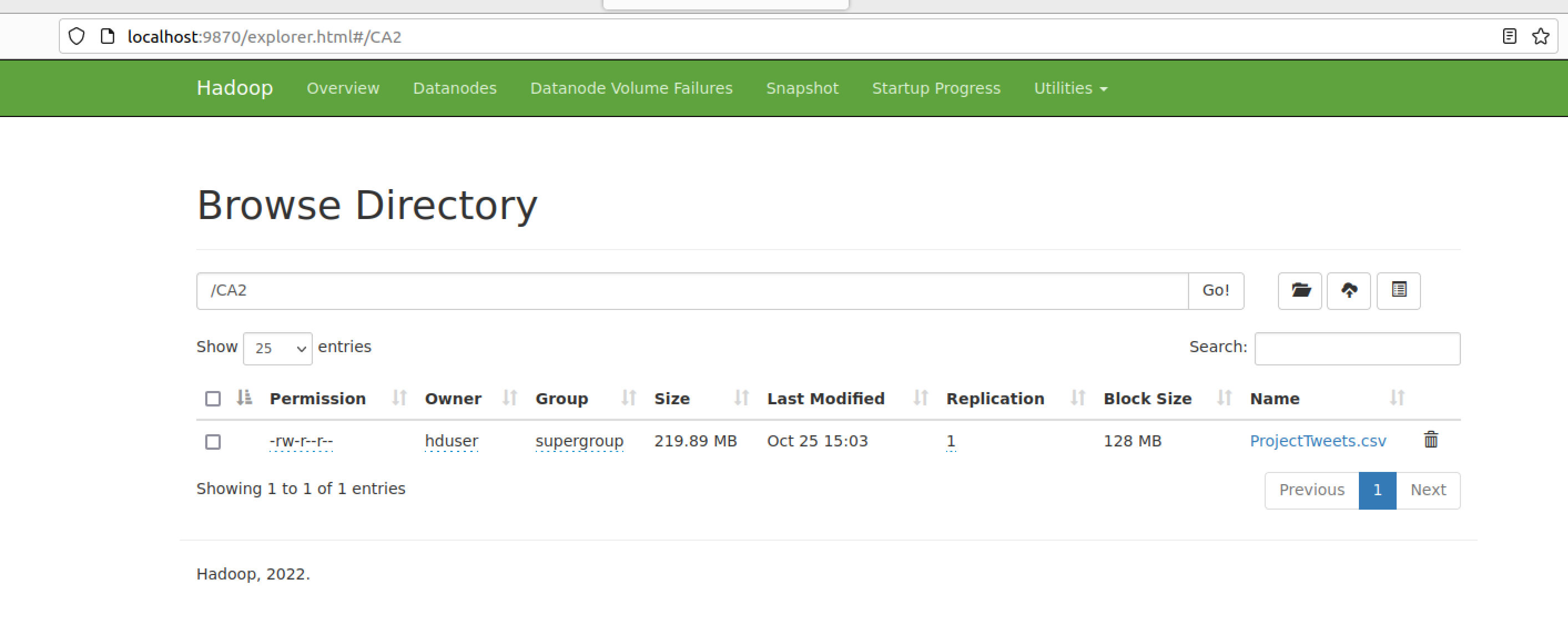
First I put the file on HDFS in a folder called CA2:



Then I checked to make sure it was present in the HDFS:

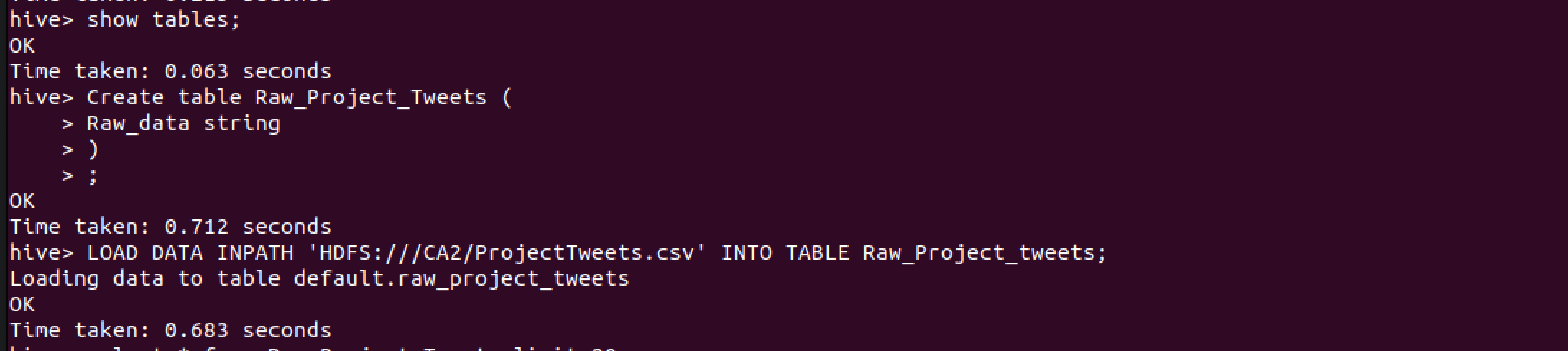


I loaded the data into Hive and also brought the data into Spark for processing using the Pyspark library.

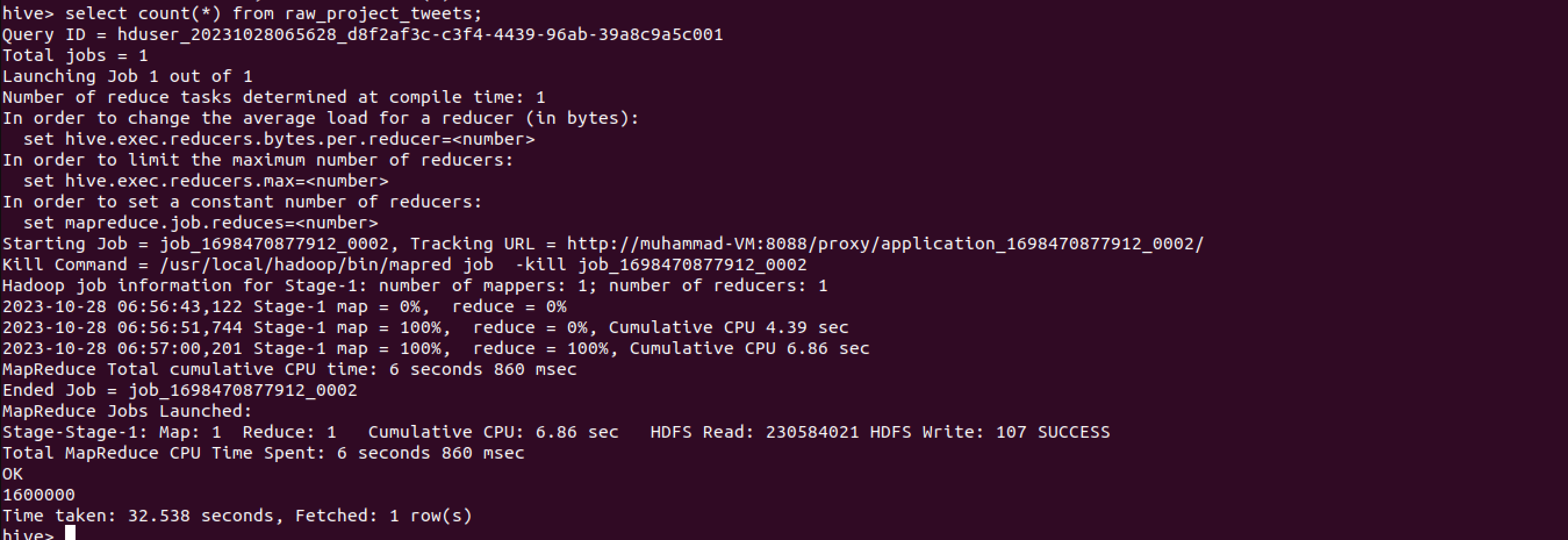
***Hive***

Note: I intentionally brought the data into a single column table in hive because even though the data is csv, I’m aware that there is a free text column that probably contains commas and I don’t want the data to be parsed or corrupted before taking a look at it. It’s important to make no assumptions about the contents of the data before first taking a look at it in its raw state.

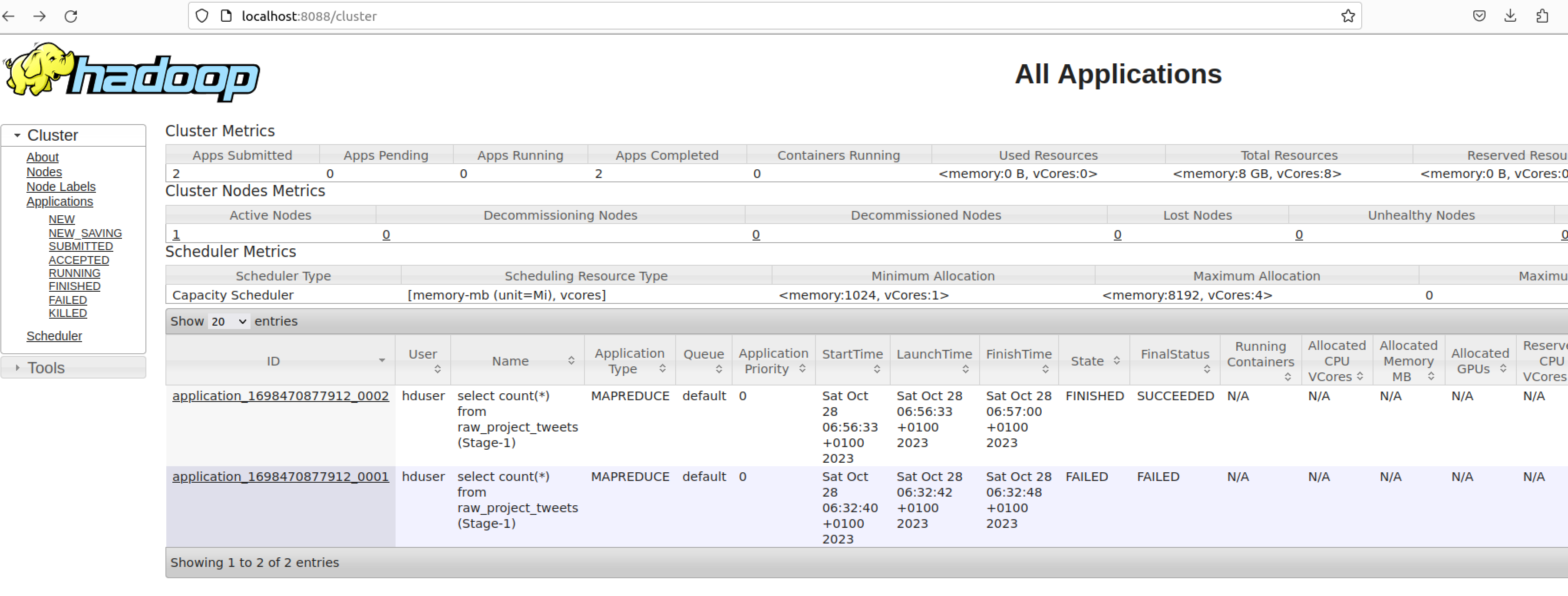
A single column table was created in Hive so that the data could be viewed unparsed in its raw state.



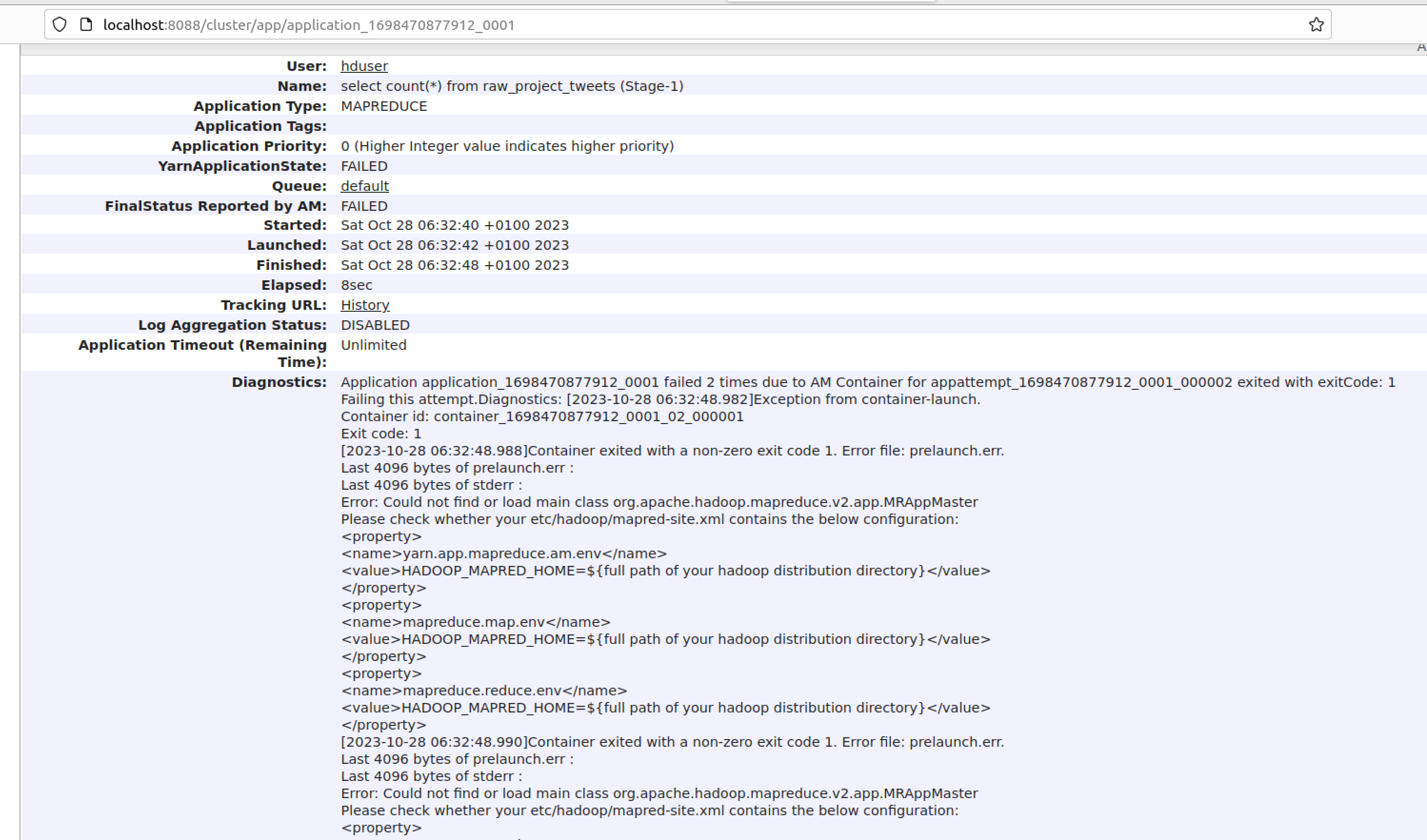
I checked to ensure the data was loaded correctly by first of all selecting a count of the rows:



It can be seen that 1.6 million tweets are present in the table. The mapreduce job failed at first because I didn’t have the relevant properties set up in the Mapred-site.xml file. This was evident in the cluster manager (found on localhost:8088):

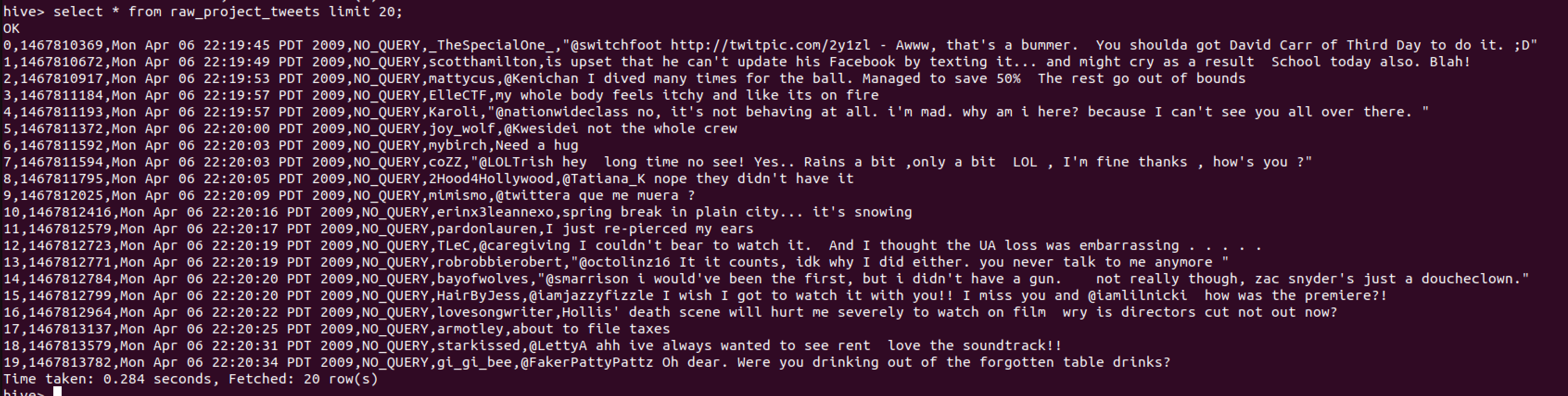


After clicking into the job itself, further details on the fail are visable:



Once I updated the XML file, the job ran successfully.

After seeing that all records made it to the table successfully, I wanted to get a very initial view of the type of data that was present by doing a select \* (limited to 20 rows):



Features such as Tweet id, date and username appear to have a consistent structure with very little noise. However the final column will need a lot of tidying before any worthwhile sentiment analysis can be performed.

Next I started a Pyspark session in Python to Preprocess and clean the data.

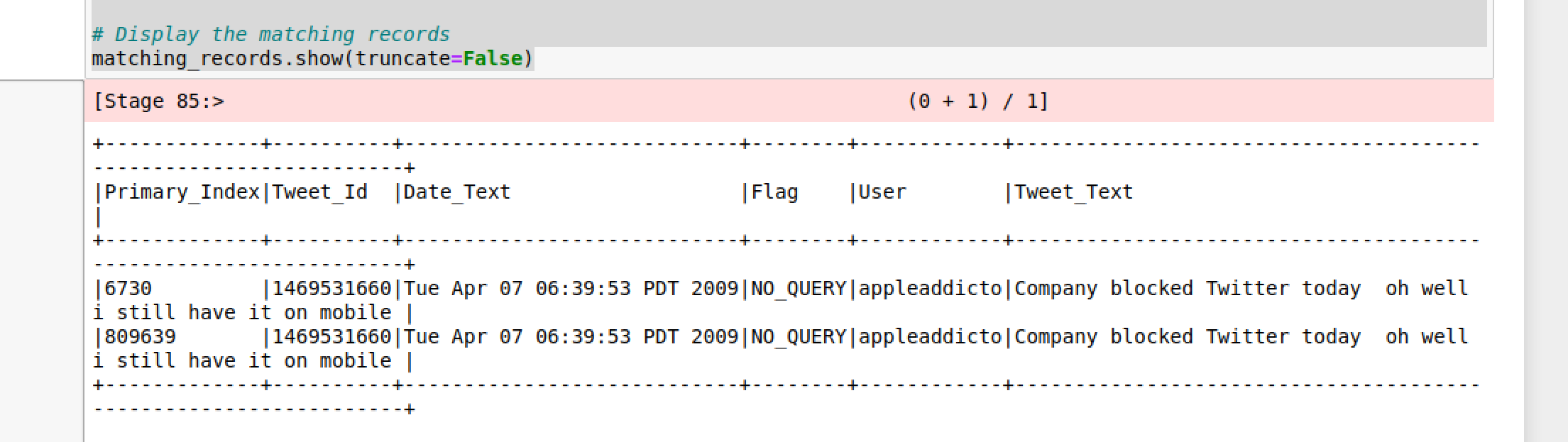
The data was loaded into the Pyspark session and the commas from the final column were removed. Commas don’t have any effect on a sentiment score or forecast, and can cause problems when moving data from one system to another while CSV delimited.

*Early Data Analysis and Preprocessing*

It is important to familiarise oneself with the dataset so further analysis was carried out on the Pyspark dataframe. Max length of each column was found.

A function was run to see if there were any nulls found in any columns. No null values were found.

A function was run to find the number of unique values in each column. This showed up a surprising result. There were only 1,598,315 unique “Tweet Ids” which suggested a level of duplication. A function was run to show an example of this which proved there were duplicate tweets in the dataset:



The reason why this is very relevant is because having duplicate tweets with, for example, a negative sentiment might skew the average sentiment score for a given day.

A new dataframe was created with unique records across the four columns:

Tweet\_Id,

Date\_Text,

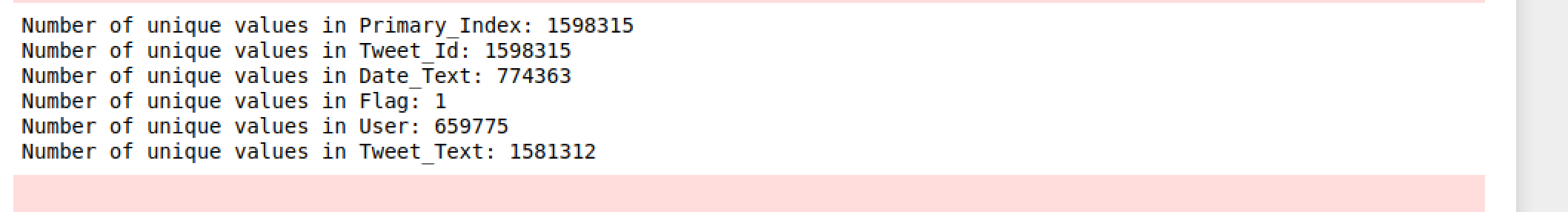
Flag,

User

And Tweet\_Text

To test the outcome, the original example was queried and was found to be unique in the new table.

Again a function was run to count the number of unique values in each column:



It can be seen that there are still some duplicates in the Tweet\_Text field, but it was deemed necessary to keep these since they have unique Tweet Ids and could be different users posting common tweets such as “Happy Birthday”/ “What a goal!” etc.

Before formatting the date data, I checked to ensure all tweets were made in the same timezone (PDT) and that was the case.

Pre-processing Data Analysis

Missing Values

There are only 80 days between the start of the dataset (‘2009-04-06’) and the end of the dataset (‘2009-06-25’). 33 of those 80 days have no tweets, and therefore no sentiment. With another dataset, it might be prudent to query with the people in the business domain as to whether the null values are “true” null values and there were simply no tweets on that day in which case the sentiment score could arguably set to ‘0’. In this case however, since “Twitter” famously has many tweets per day, it will be assumed that some tweets are missing and we should take action to impute the data.

It is worth noting that to have 33 values missing out of 80 is quite a large portion. Imputing missing values generally adds noise to the dataset and therefore the less imputing the better.

There are many ways we could impute the missing data. There are options such as forward fill, backward fill, linear interpolation, cubic interpolation as well as seasonal means. The approach taken here is to test each of these approaches against the data we know is correct, to see which approach brought about the most accurate score. That approach will then be applied to the rest of the dataset.

<https://builtin.com/data-science/time-series-forecasting-python>

^ good for seasonality vs trend and gives a link to description. Plus it talks through a few Time Series approaches

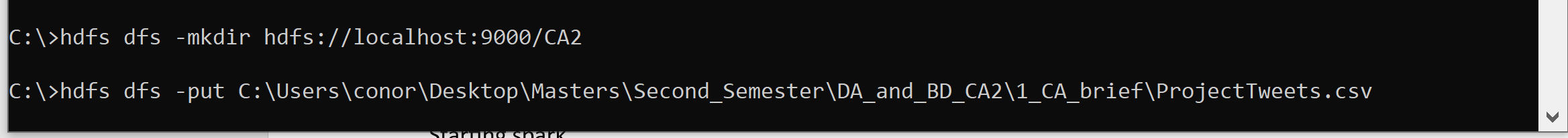
Time Series

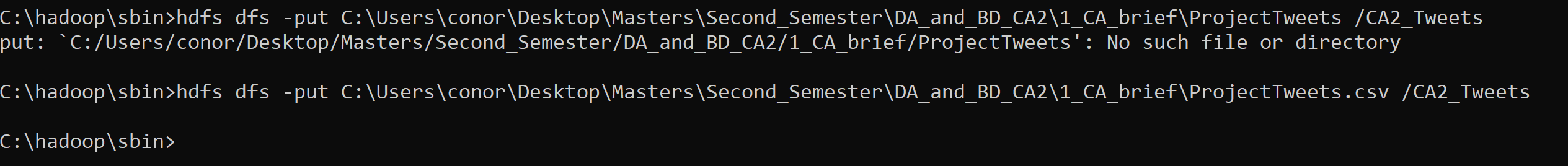
Now that the missing values are filled in, we can continue with the time series

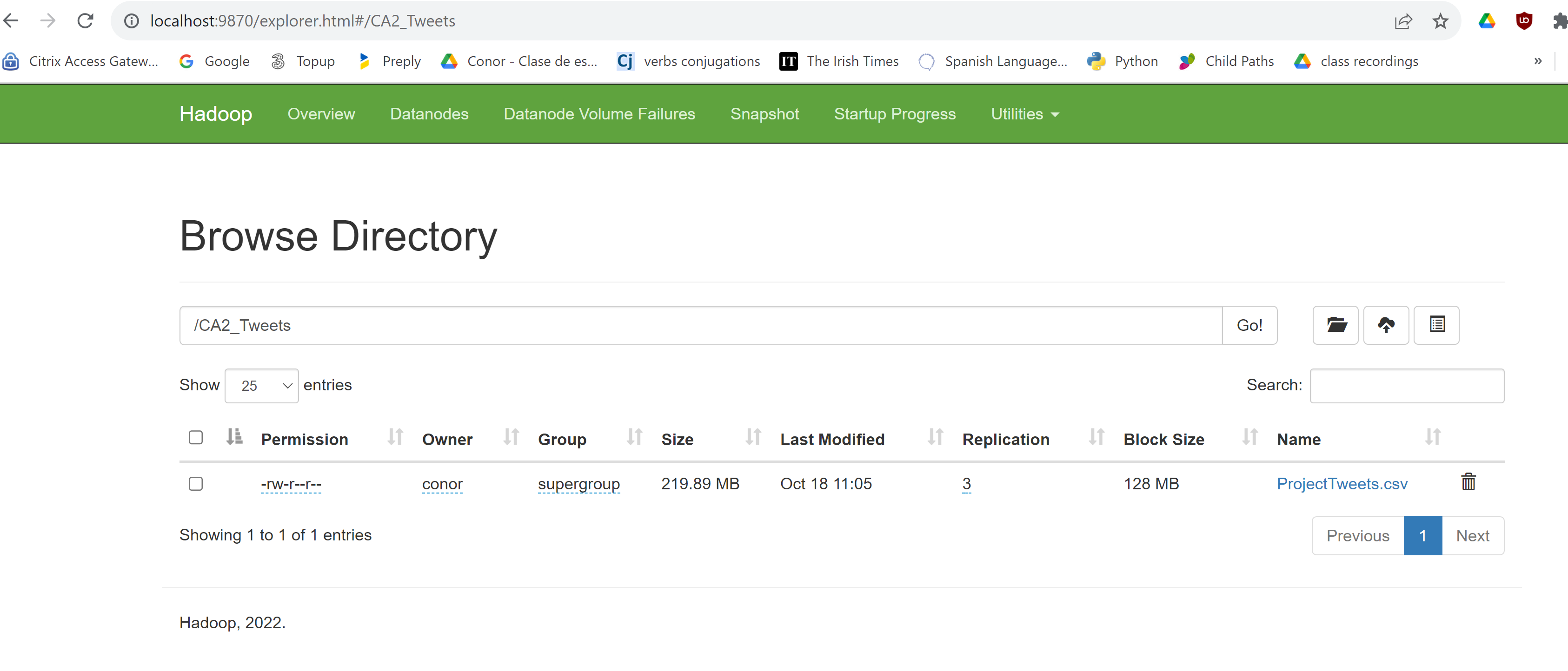
Loading the file into HDFS

I opened HDFS using the CMD prompt (ran as an administrator) and I uploaded the ProjectTweets dataset using the put function in the CMD line:

Initially there were issues because the file path had spaces within it but after those spaces were removed, the file successfully uploaded into HDFS:



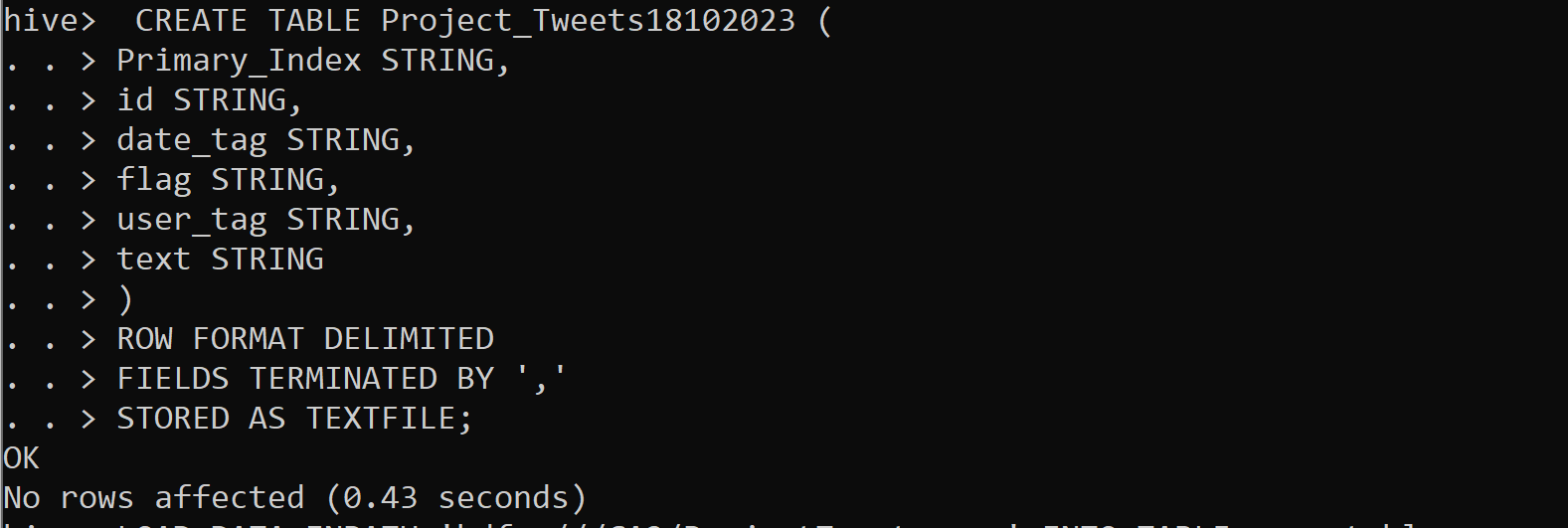


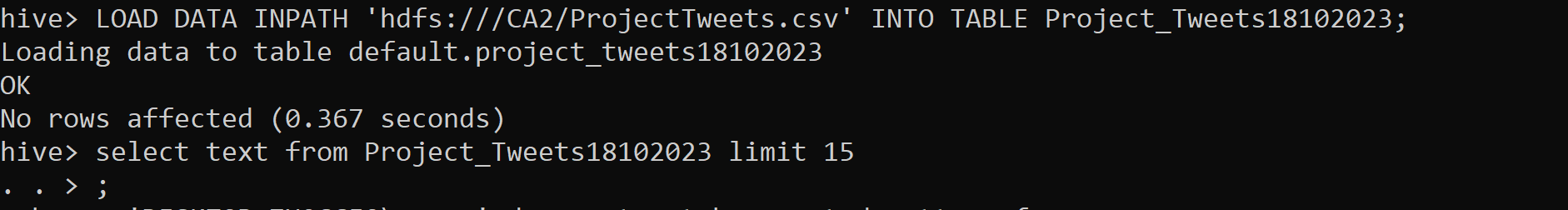


We are using a standalone Hadoop distribution. But it’s possible to check the cluster detail, so it’s possible to check how many clusters on the standalone machine Hadoop is developing.

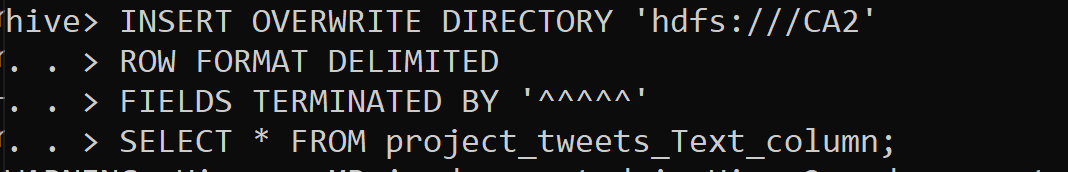
If we had different machines we could decide how to distribute the data, Hadoop automatically takes responsibility that but there are. Hadoop does it all internally.

HIVE





Send back the one column for the map reduce job



Mapreduce:

C:\>hadoop jar C:\hadoop\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.10.2.jar wordcount /CA2/000000\_0 /output/part\_1

