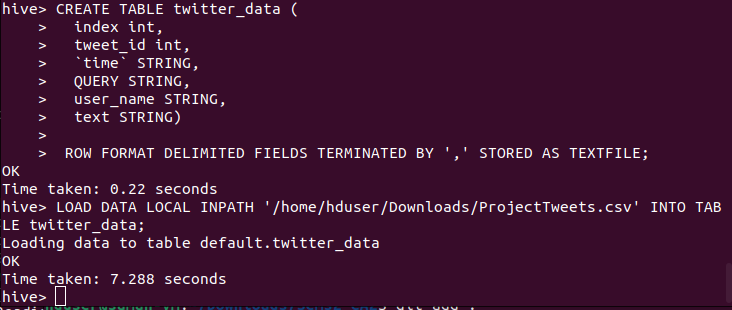
**Big data**

**data storage and processing activities:**

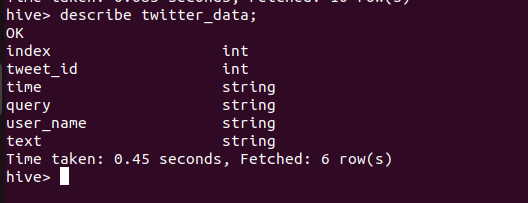
The data file ProjectTweets.csv downloaded from Moodle to be stored and processed using Apache hive. The reason to choose Hive is that it gives SQL like interface to query data and stored in file system integrated with Hadoop so you can leverage Hadoop map reduce without the need to write code.

The dataset contains columns of ids, tweet\_id, tweet text, user name, date and time and flag. It contains about 160000 values.

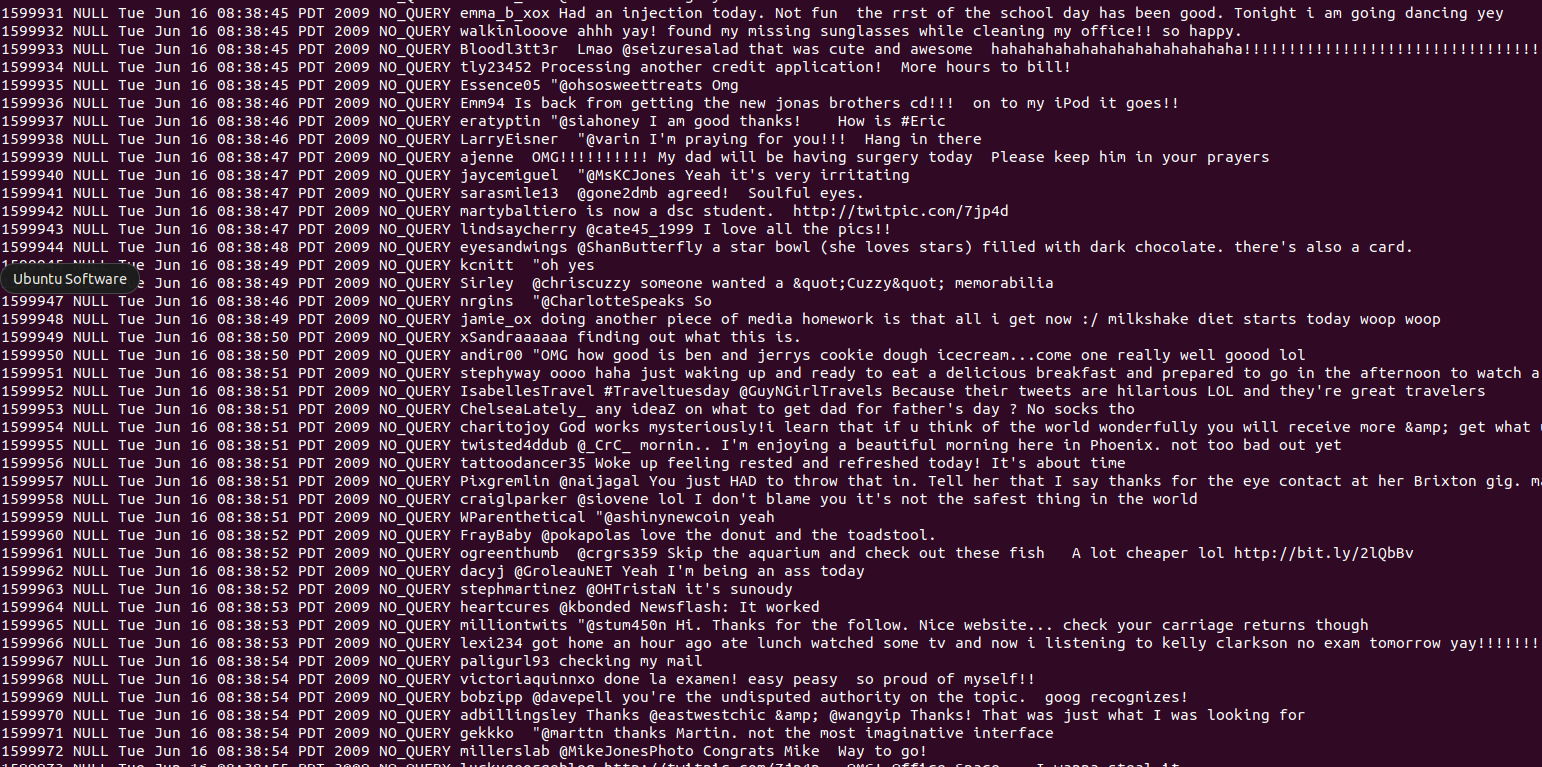
Loading the data as a table in Hive, the screen shot below showing the code and the output of the code.



We can check the schema of the table using the command describe as illustrated in the screenshot:

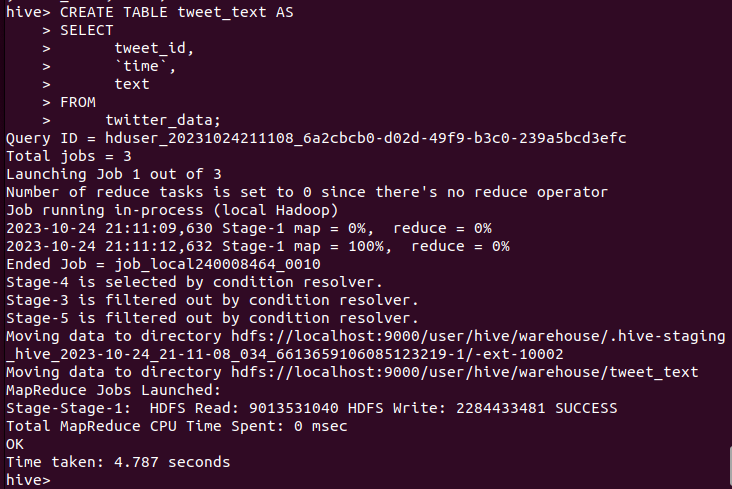
****

Check the table content: tweet text and ids using the command select \* from twitter\_data; the output is in the next screenshot:

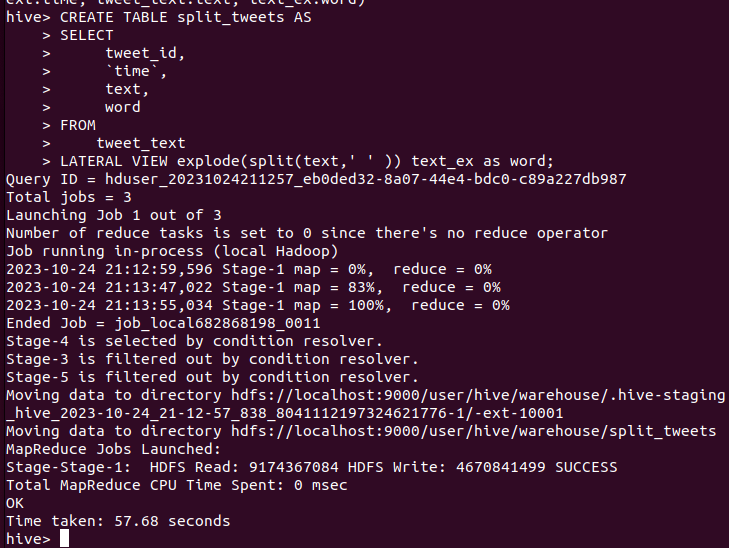


First, cleaning the tweets by getting rid of the hashtags and the usernames mentioned in the tweet text. To do this, create a table and select only the columns of interest form the main table. Only the id, time, tweet\_text are selected because the other columns has no significant impact in the sentiment analysis that will be applied later.

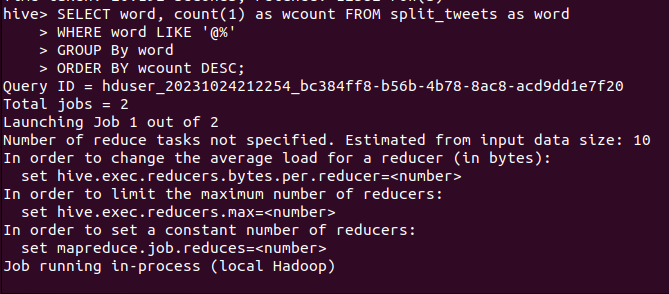
The next screenshot shows the new table with only the ids, time, tweet\_text columns:



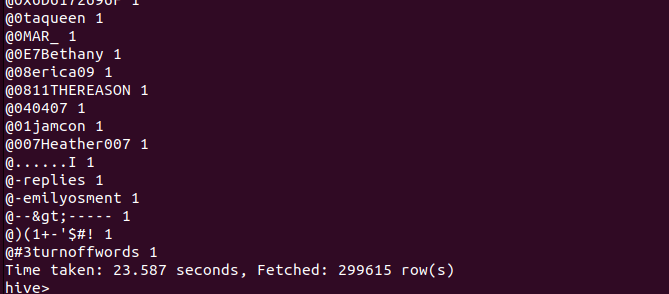
First step to clean the data is to split the tweet text to words then create a table to store the processed data temporary**.**

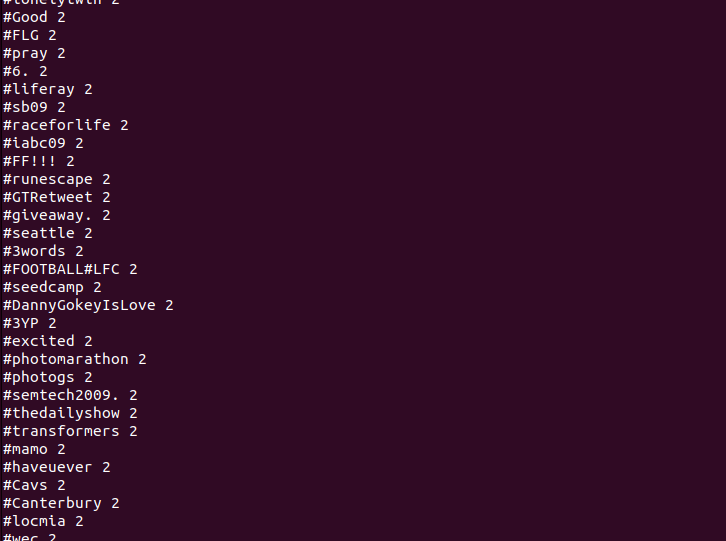
****

**Then extract hashtags and usernames :**

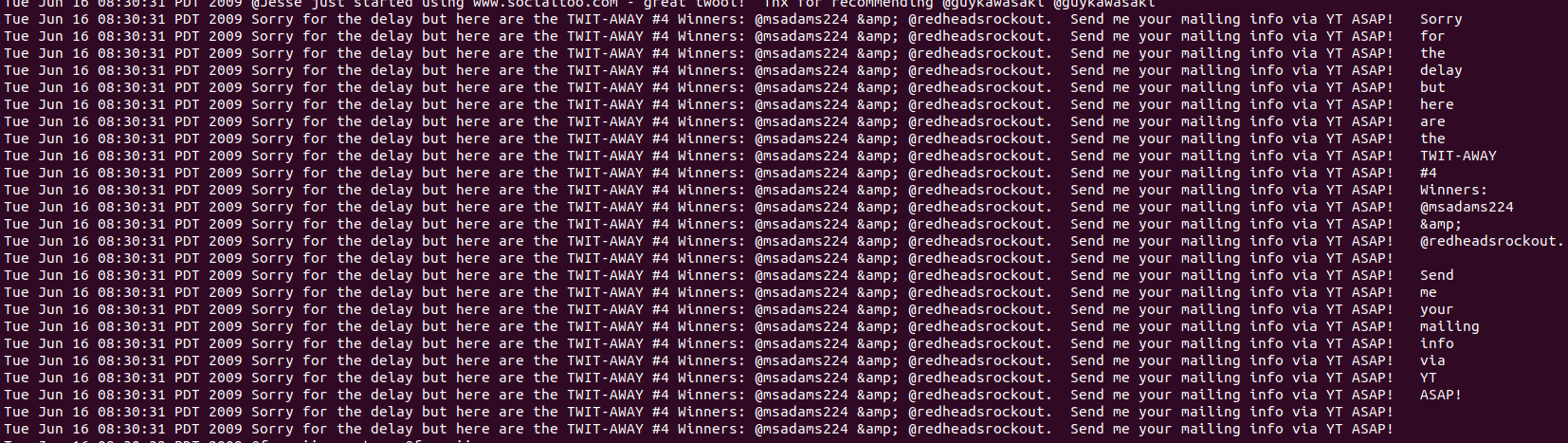
****

**The output for the code shown in the above screen shot is as follows:**

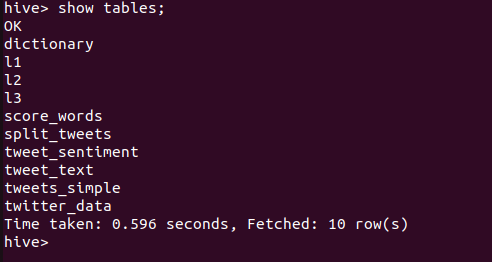
****

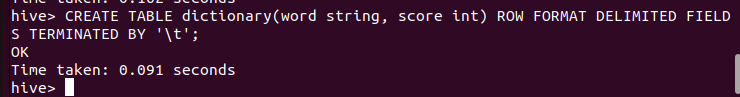
****

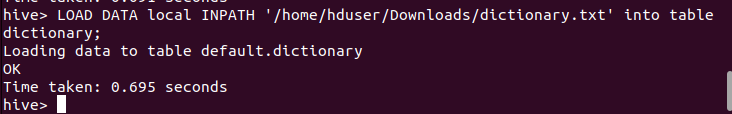
**Splitted tweets to words as illustrated in the next screen shot:**

****

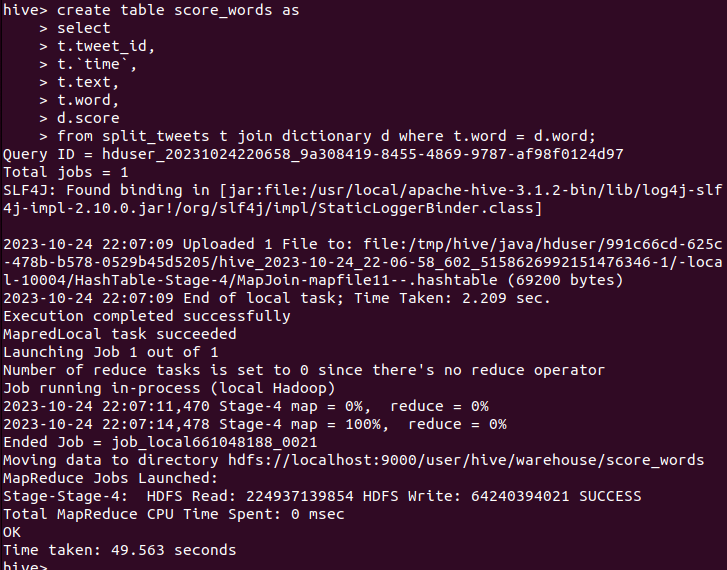
**To show tables created so far use the command shown in the screen shot below:**

****

**To apply the sentiment analysis on the words of each tweet will create a table for the dictionary and load the data by using a downloaded dictionary file from the github account. **

****

**Now, after split the tweet text into individual words to give them a sentiment score. Create another table to do so.**

****

**Comparative analysis for two databases:**

Which databases to compare? And why?

A comparison of the performance of tow open-source NoSQL database, Cassandra and MongoDB will be shown in next paragraph. The reason why particularly Cassandara and MongoDB is that the two databases have different characteristics such as the data model, query language and scalability.

Which test tool? And why?

The tool used in performing the comparison is YCSB. It is standardized benchmarking tool for evaluating the performance of NoSQL databases. It provides a common framework for comparing different database systems, making it easier to obtain consistent and comparable results.

Test strategy:

The two databases will be tested by applying tow workloads from ycsb benchmarking tool. The first workload is workload A: which is for update heavy workload ; 50/50% mix of reads/writes

Workload C: Read-only: 100%. Both workloads have the default settings with number of operations =1000, number of record =1000 and a Zipfian distribution.

**Set metrics for workloada:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Database name** | **Average Latency(us)** | **Runtime (MS)** | **Throughput(ops/sec)** |
| **MongoDB** | 770.414 | 1362 | 734.214390 |
| **Cassandra** | 2456.127 | 6150 | 162.601626 |

**Set metrics for workloadc:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Database name** | **Average Latency(us)** | **Runtime (MS)** | **Throughput(ops/sec)** |
| **MongoDB** | 671.494 | 1297 | 771.01002 |
| **Cassandra** | 1609.546 | 4494 | 222.51891 |

**Perform quantitative analysis:**

For workloada:

MongoDB shows a faster performance in all metrics: for the average latency it is 3 times faster than Cassandra and for the runtime it is 4 times faster than Cassandra. For number of operations per second MongoDB did 734 operations while Cassandra did 162 operations per second.

For workloadc:

The performance is very similar to workloada MongoDB is faster. However, the choice between MongoDB and Cassandra may depend on various factors, including application's specific requirements and performance benchmarks.