**Assignment 1 - Data Analysis** CA675 - Cloud Technologies

Name: Likitha Shivananjegowda

Student-Id: 22265298

Email : [likitha\_shivananjegowda2@mail.dcu.ie](mailto:likitha_shivananjegowda2@mail.dcu.ie)

Link to Git Repository: <https://github.com/liki2124/Cloud_spamdetection>

Link to project on cloud system: <https://console.cloud.google.com/compute/instances?project=festive-shield-362616>

<https://console.cloud.google.com/dataproc/clusters?project=festive-shield-362616>

Task 1: Installation of Hadoop , Hive and Pig on GCP environment

A small understanding of each tool according to my understanding before looking into to the installation of each tools

Hadoop: Using straightforward programming languages like java, the Apache Hadoop software library provides a framework for the distributed processing and storage of massive data sets across computer clusters, for optimization of the Map Reduce jobs, java provides flexibility.

PIG is a high-level data-flow language and has a parallel computing execution framework. For those who are proficient in Python-like scripting languages, it is a good exploration to work with hive.

Hive is a data warehouse infrastructure that offers ad-hoc querying and data summarization. It is advised for those with programming experience in SQL because the syntax is similar.

For developers, PIG and HIVE hide the complexity of Map Reduce Jobs. They provide a higher level of abstraction to solve business problems.

Both PIG and HIVE convert scripts/queries to a series of MAP reduce jobs. They are inefficient in terms of performance when compared to traditional MAP reduce jobs implemented in Java.

For java installation:

likitha\_shivananjegowda2@hadoop1:~$ sudo apt update

likitha\_shivananjegowda2@hadoop1:~$ sudo apt install openjdk-8-jdk -y

likitha\_shivananjegowda2@hadoop1:~$ java -version; javac -version

likitha\_shivananjegowda2@hadoop1:~$ sudo apt install openssh-server openssh-client -y

likitha\_shivananjegowda2@hadoop1:~$ sudo adduser likitha

likitha\_shivananjegowda2@hadoop1:~$ su - likitha

likitha\_shivananjegowda2@hadoop1:~$ sudo adduser likitha

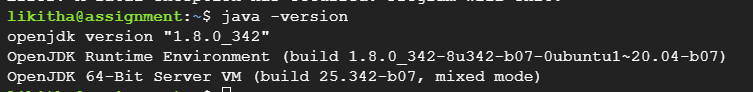
likitha\_shivananjegowda2@hadoop1:~$ sudo ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa

likitha\_shivananjegowda2@hadoop1:~$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

likitha\_shivananjegowda2@hadoop1:~$ chmod 0600 ~/.ssh/authorized\_keys

likitha\_shivananjegowda2@hadoop1:~$ ssh localhost

Screenshots of java and hadoop installation:



Installation of Hadoop:

likitha\_shivananjegowda2@hadoop1:~$ wget <https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz>

likitha\_shivananjegowda2@hadoop1:~$ sudo tar xvzf hadoop-3.2.3.tar.gz

likitha\_shivananjegowda2@hadoop1:~$ sudo nano .bashrc

export HADOOP\_HOME=/home/likitha/hadoop-3.2.3

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export PATH=$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/bin

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

likitha\_shivananjegowda2@hadoop1:~$ source ~/.bashrc

likitha\_shivananjegowda2@hadoop1:~$ sudo nano $HADOOP\_HOME/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

likitha\_shivananjegowda2@hadoop1:~$ sudo nano $HADOOP\_HOME/etc/hadoop/core-site.xml

<property>

<name>hadoop.tmp.dir</name>

<value>/home/likitha/tmpdata</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

<description>The name of the default file system></description>

</property>

likitha\_shivananjegowda2@hadoop1:~ $ sudo nano $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

<property>

<name>dfs.data.dir</name>

<value>/home/likitha/dfsdata/namenode</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/home/likitha/dfsdata/datanode</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

likitha\_shivananjegowda2@hadoop1:~$ sudo nano $HADOOP\_HOME/etc/hadoop/mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

likitha\_shivananjegowda2@hadoop`:~$ sudo nano $HOME\_HOME/etc/hadoop/yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>127.0.0.1</value>

</property>

<property>

<name>yarn.acl.enable</name>

<value>0</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PERPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>

</property>

likitha\_shivananjegowda2@hadoop1:~$ hdfs name -format

likitha\_shivananjegowda2@hadoop1:~$ ./start-dfs.sh

likitha\_shivananjegowda2@hadoop1:~$ ./start-yarn.sh

likitha\_shivananjegowda2@hadoop1:~$ jps

Installation of Pig:

likitha\_shivananjegowda2@hadoop1:~$ wget <https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz>

likitha\_shivananjegowda2@hadoop1:~$ tar xvzf pig-0.16.0.tar.gz

likitha\_shivananjegowda2@hadoop1:~$ sudo mv /home/likitha\_shivananjegowda2/pig-0.16.0 /home/likitha\_shivananjegowda2/pig

likitha\_shivananjegowda2@hadoop1:~$ sudo nano .bashrc

export PIG\_HOME=/home/likitha/pig

export PATH=$PATH:$PIG\_HOME/bin

export PIG\_CLASSPATH=$PIG\_HOME/conf:$HADOOP\_INSTALL/etc/hadoop/

export PIG\_CONF\_DIR=$PIG\_HOME/conf

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export PIG\_CLASSPATH=$PIG\_CONF\_DIR:$PATH

likitha\_shivananjegowda2@hadoop1:~$ source .bashrc

likitha\_shivananjegowda2@hadoop1:~$ pig

Installation of Hive:

likitha\_shivananjegowda2@hadoop1:~$ wget <https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz>

likitha\_shivananjegowda2@hadoop1:~$ tar xvzf apache-hive-3.1.2-bin.tar.gz

likitha\_shivananjegowda2@hadoop1:~$ sudo mv apache-hive-3.1.2-bin hive

likitha\_shivananjegowda2@hadoop1:~$ sudo nano ~/.bashrc

export HIVE\_HOME=/usr/local/hive

export PATH=$PATH:$HIVE\_HOME/bin

likitha\_shivananjegowda2@hadoop1:~$ source ~/.bashrc

likitha\_shivananjegowda2@hadoop1:~$ sudo nano $HIVE\_HOME/bin/hive-config.sh

export HADOOP\_HOME=/home/likitha/hadoop-3.2.1

likitha\_shivananjegowda2@hadoop1:~$ hdfs dfs -mkdir /tmp

likitha\_shivananjegowda2@hadoop1:~$ hdfs dfs -chmod g+w/tmp

likitha\_shivananjegowda2@hadoop1:~$ hdfs dfs -mkdir -p /user/hive/warehouse

likitha\_shivananjegowda2@hadoop1:~$ hdfs dfs -chmod g+w /user/hive/warehouse

likitha\_shivananjegowda2@hadoop1:~$ rm $HIVE\_HOME /lib/guava-19.0.jar

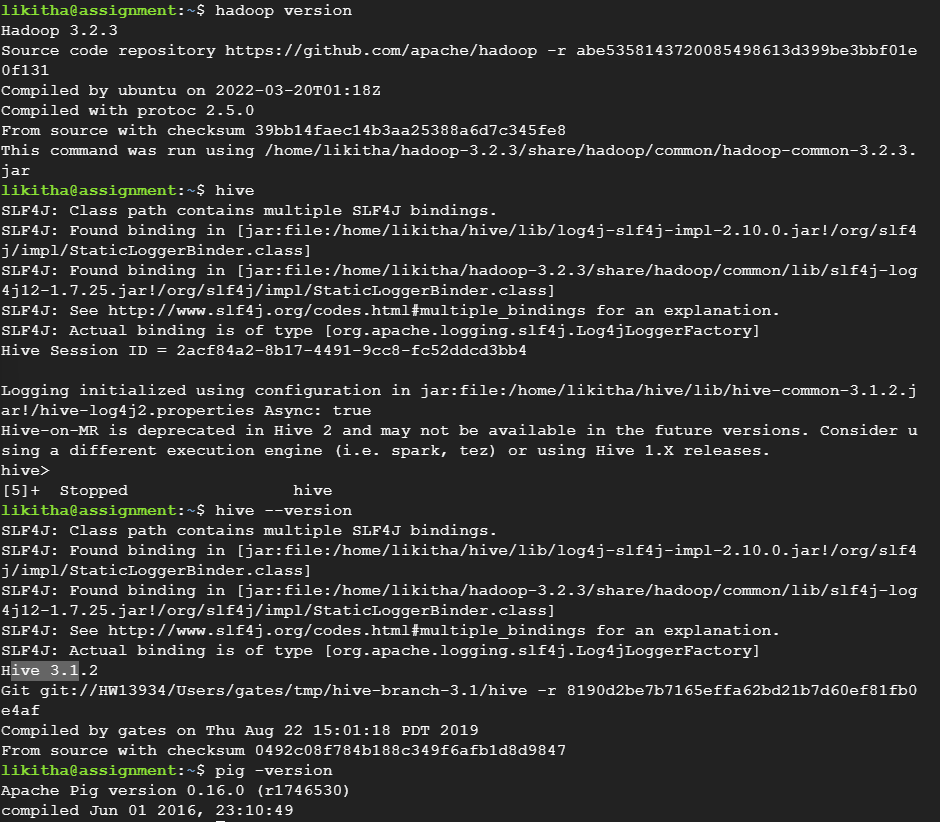
likitha\_shivananjegowda2@hadoop1:~$cp $HADOOP\_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE\_HOME/lib/schematool -initSchema -dbType derby

likitha\_shivananjegowda2@hadoop1:~$ cd $HIVE\_HOME/conf

likitha\_shivananjegowda2@hadoop1:~$ cp hive-default.xml.template hive-site.xml

likitha\_shivananjegowda2@hadoop1:~$sudo nano hive-site.xml

likitha\_shivananjegowda2@hadoop1:~$ hive



TASK 2:

Get dataset from Public repository provided link to respective repo:

Link to the dataset used :

<https://www.kaggle.com/datasets/naveedhn/amazon-product-review-spam-and-non-spam?select=Toys_and_Games>

In the dataset used which has amazon product reviews labeled as Spam or Ham. There are around million reviewers in this large corpus. The class labels are Spam and Ham, with a "0" indicating no spam and a "1" indicating spam reviews.

Since I have got json file so used python convert csv file:

import pandas as pd

import numpy

import json

df1= pd.read\_json("C:/Users/likit/Downloads/Toys\_and\_Games/Toys\_and\_Games.json", lines=True, nrows=500)

cnt=0

for df in pd.read\_json('C:/Users/likit/Downloads/Toys\_and\_Games/Toys\_and\_Games.json', lines=True, chunksize=1997140):

print(df.shape)

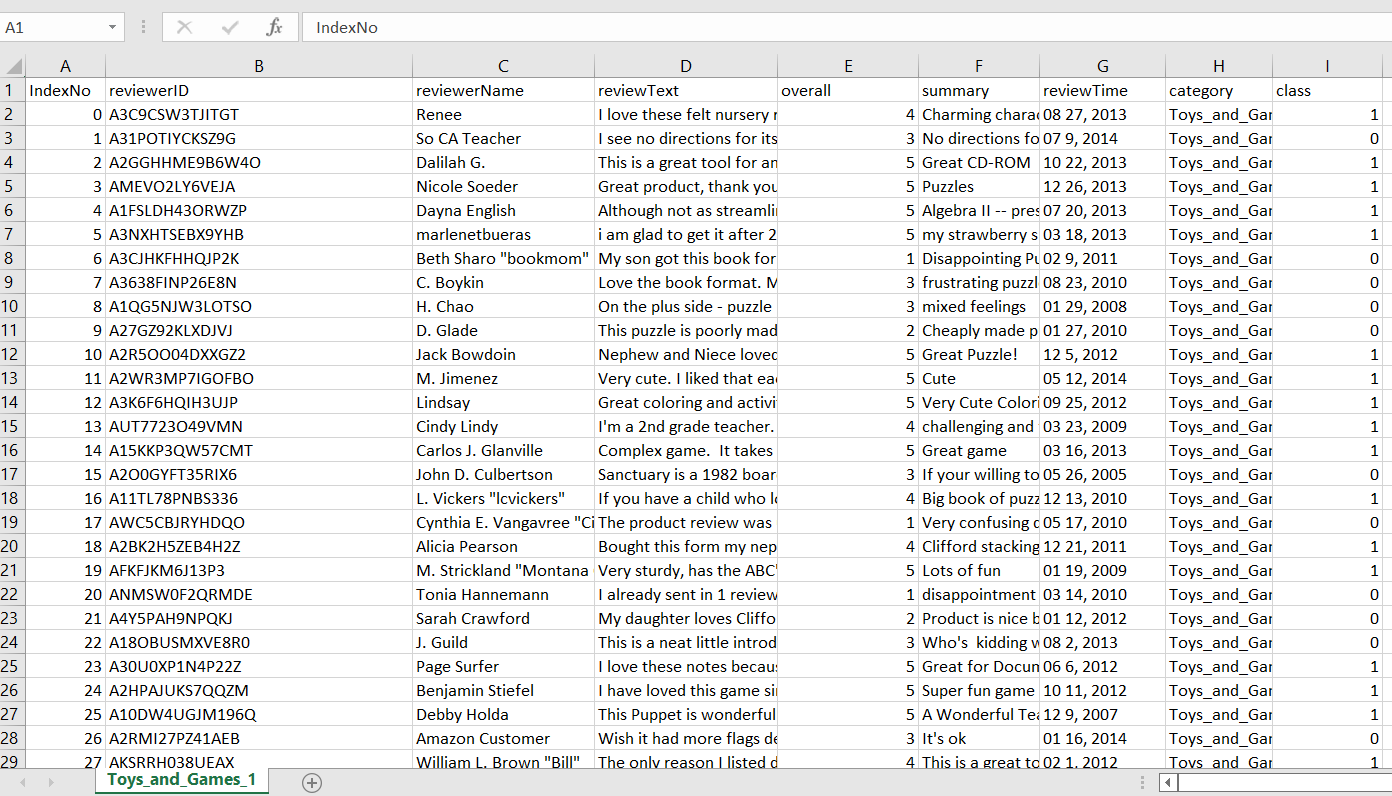
cnt=cnt+1

path=f"Toys\_and\_Games\_{cnt}.csv"

df.to\_csv(path)

df = pd.read\_csv('C:/Users/likit/Toys\_and\_Games.csv')

Uploading the csv file based on class of the emails



Using this csv file I have manually updated the data into the dataproc cluster

Task 3:

Data cleaning is the process of removing or correcting incorrect records from a table. The rules used to clean the data are determined by the business requirements.

In the dataset I have taken the date must be within a class to differenciate between ham and spam. I believe the Dataset may contain records with dates that are outside of the proper accuracy at some point, we can filter those records in a Relation in Pig and use that relation to implement Logic.To filter out records where columns have NULL values and the logic does not want those values to be NULL. You can update those records in a relation that have the string value NA by hard coding Field values and creating a separate relation for NULL values.

When I check the length of a specific field it was seen that there may be some junk values in String fields. While it was exceeding the threshold, it can be filtered by going through the records.

Everything depends on the data, so one must perform data validation to ensure that corrupt records can be cleaned by integrating various data validation checks into the Pig Script.

To clean and process the data I have used pig and hive queries

1st to upload files from local home directory to hadoop using commands

Likitha\_shivananjegowda2@spamdetectionengine:~$ ls

Likitha\_shivananjegowda2@spamdetectionengine:~$ hadoop fs -put Toys\_and\_Games\_1.csv

Likitha\_shivananjegowda2@spamdetectionengine:~$cat Toys\_and\_Games.csv

2nd used pig to clean data like replaced it with newlines where there were spaces

Likitha\_shivananjegowda2@spamdetectionengine:~$ sed':a;N;$!ba;s/\n//g' Toys\_and\_Games > Toys\_And\_Games.csv

> pig – useHCatalog

> REGISTER /usr/lib/pig/piggybank.jar

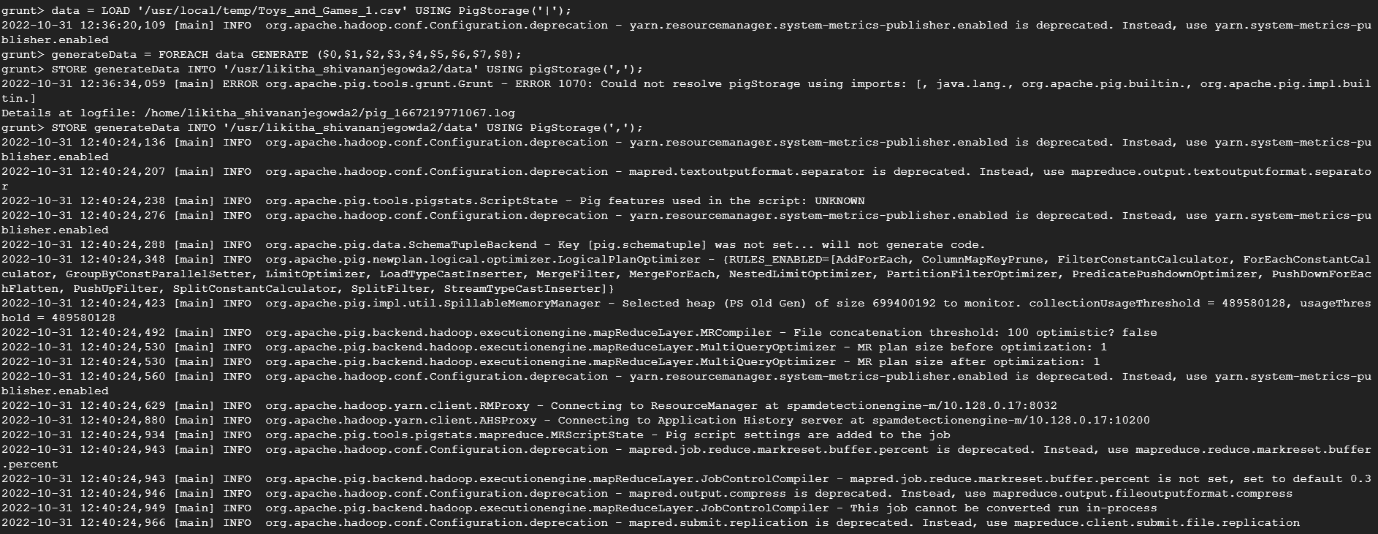
> DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader();

In order to load data from Hadoop to Pig

Likitha\_shivananjegowda2@spamdetectionengine:~$LOAD'/Toys\_and\_Games'USING org.apache.pig.piggybank.storage.CSVExcelStorage (',','YES\_MULTILINE','NOCHANGE','SKIP\_INPUT\_HEADER') AS (IndexNo BIGINT,reviewerID String,reviewerName varchar(255),reviewText varchar(255),overall int,summary varchar(255),reviewTime varchar(255),category varchar(255),class String);

To remove newlines form the body column and dump clean data:

Likitha\_shivananjegowda2@spamdetectionengine:~$ cleaneddata FOREACH stackdata GENERATE IndexNo, reviewerID, reviewerName, reviewText, overall, summary,class (REPLACE(body,'[\r\n]+',' ')) AS body

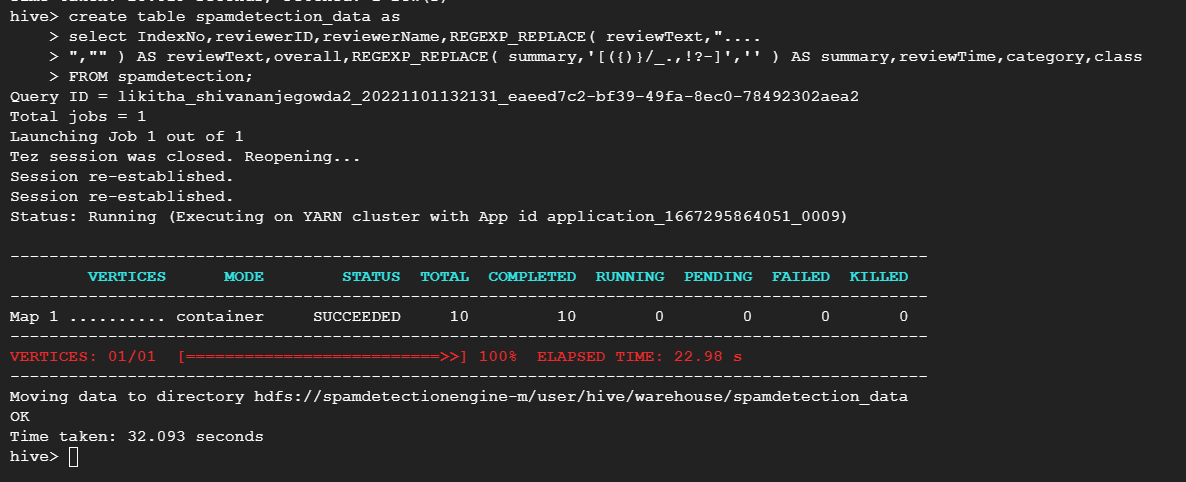


Step to load cleaned data from Pig to Hive we create database and table in Hive

hive> CREATE DATABASE spamdetection;

hive> USE spamdetection;

hive> CREATE TABLE spamdetection (IndexNo BIGINT,reviewerID String,reviewerName varchar(255),reviewText varchar(255),overall int,summary varchar(255),reviewTime varchar(255),category varchar(255),class String);



hive> describe spamdetection;

hive> select \* from spamdetection;

//load for all table working

hive> LOAD DATA INPATH '/user/likitha\_shivananjegowda2/mydir/Toys\_and\_Games.csv' INTO TABLE spamdetection; // LOADING THE FROM HDFS

hive> SELECT \* FROM spamdetection LIMIT 10; // TO CHECK THE DATA ACCRUARCY

//creating new enriched table structure

hive> create table spamdetection\_data as

hive> select IndexNo,reviewerID,reviewerName,REGEXP\_REPLACE( reviewText,'[({)}/\_.,!?-]',"" ) AS reviewText,overall,REGEXP\_REPLACE( summary,'[({)}/\_.,!?-]',"" ) AS summary,reviewTime,category,class

FROM spamdetection ;

//enriched loading data

Hive> insert overwrite table (spamdetection\_data select IndexNo,reviewerID,reviewerName,REGEXP\_REPLACE( reviewText,'[({)}/\_.,!?-]',"" ) AS reviewText,overall,REGEXP\_REPLACE( summary,'[({)}/\_.,!?-]',"" ) AS summary,reviewTime,category,class ) FROM spamdetection;

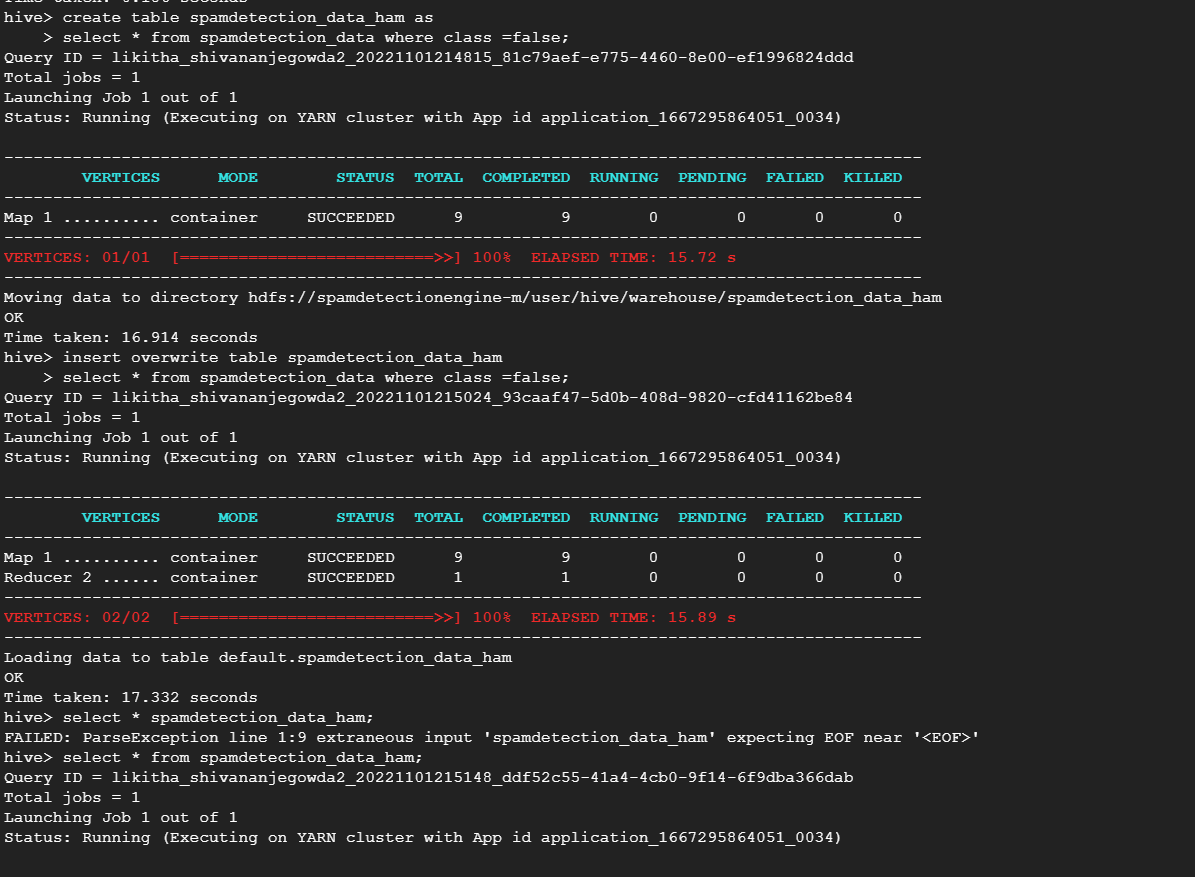
//creating table for spam and ham

Hive> create table spamdetection\_data\_spam as

Hive> select \* from spamdetection\_data where class =true;

Hive>create table spamdetection\_data\_ham as

Hive> select \* from spamdetection\_data where class =false;



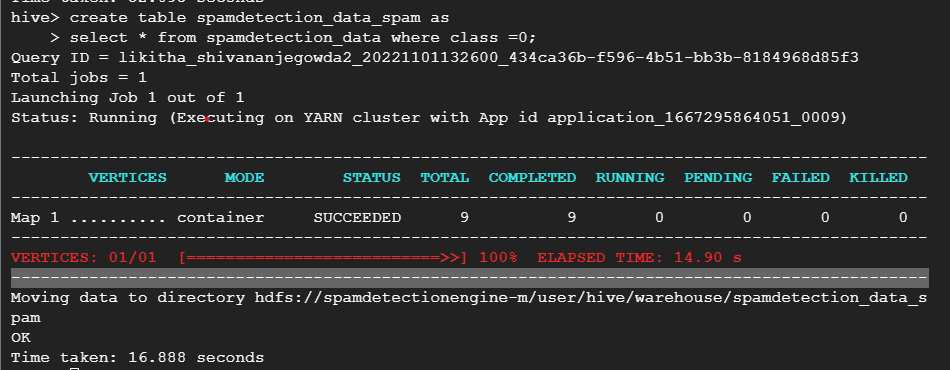
/inserting values to spam table

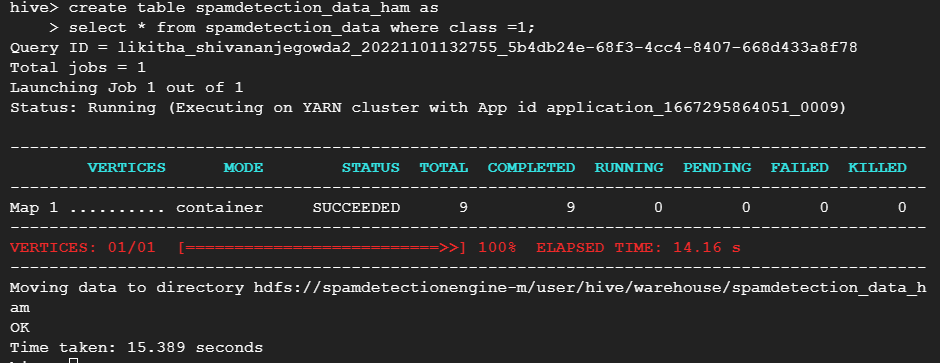
Hive> insert overwrite table spamdetection\_data\_spam

Hive> select \* from spamdetection\_data where class =true;

Hive> insert overwrite table spamdetection\_data\_ham

Hive> select \* from spamdetection\_data where class =false;





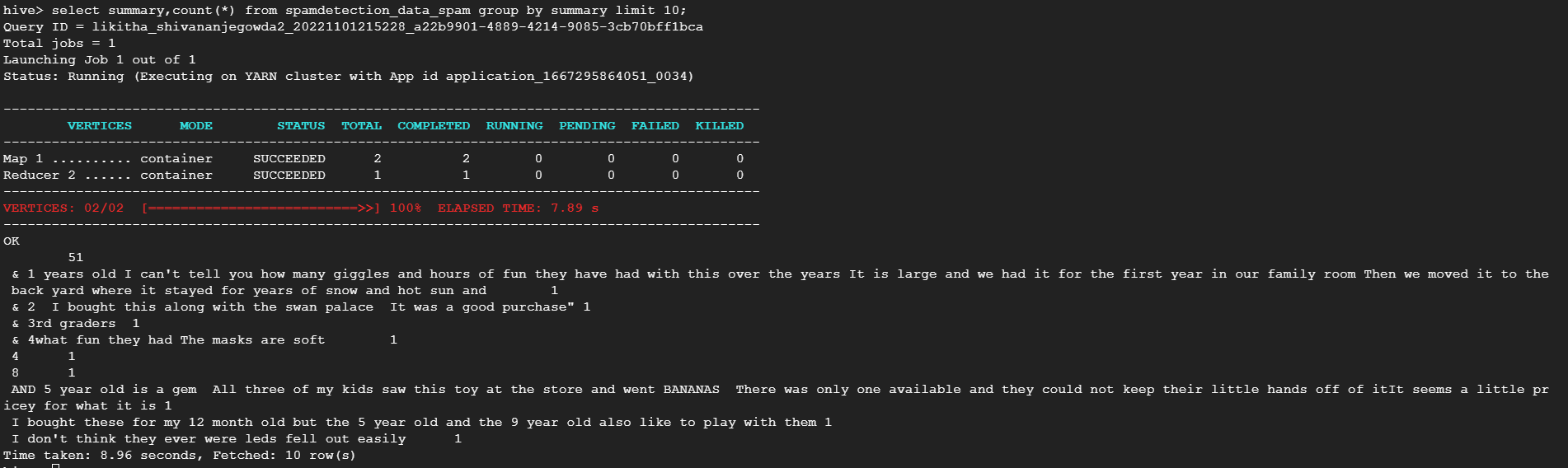
// query to list top 10 spam messages

Hive> select summary,count(\*) from spamdetection\_data\_spam group by summary limit 10;

Tried the two queries to check if it works fine soo uploaded both the queries :

Hive> With topspam as(Select reviewerID,count(\*)cnt from spamdetection\_data\_spam GROUP BY reviewerID)

Select reviewerID,cnt from topspam ORDER BY cnt DESC limit 10;



Task 5:

TF-IDF using MapReduce

tf-idf can be calculated with tfidf(t,d,D) = tf(t,d) \* idf(t, D) where tf denotes the term frequency and idf the inverse frequency, while the tf can be calculated in different for me it was about classifying the dataset with spam or ham so I used Boolean expression. Also I find a normalized form where the data can be divided by making use of frequency found on counting the words.

In order to implement tf-idf using mapreduce I have divided the task into 2 as mapper and reducer in which I have a logic on implementing mapper and reducer programs in python and tried first implementing on local and I dint find any bugs in the code and delivered the same inside my Hadoop but im not excepting the output as the data is huge its failing displaying runtime error soo I had to end up with the same output I have given that screenshot respectively.

Commands I used to run the mapreduce program:

Likitha\_shivananjegowda2@spamdetectionengine:~$ $HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar -input mapReduce1 -output mapReduce2 -file /home/likitha\_shivananjegowda2/mapper1.py -mapper "python mapper1.py" -file /home/likitha\_shivananjegowda2/reducer1.py -reducer "python reducer1.py"

Likitha\_shivananjegowda2@spamdetectionengine:~$ $HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar -input mapReduce3 -output mapReduce4 -file /home/likitha\_shivananjegowda2/mapper2.py -mapper "python mapper2.py" -file /home/likitha\_shivananjegowda2/reducer2.py -reducer "python reducer2.py"

Likitha\_shivananjegowda2@spamdetectionengine:~$ $HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/hadoop-streaming.jar -input mapReduce5 -output mapReduce6 -file /home/likitha\_shivananjegowda2/mapper3.py -mapper "python mapper3.py" -file /home/likitha\_shivananjegowda2/reducer3.py -reducer "python reducer3.py"

