**Huynh Vo**

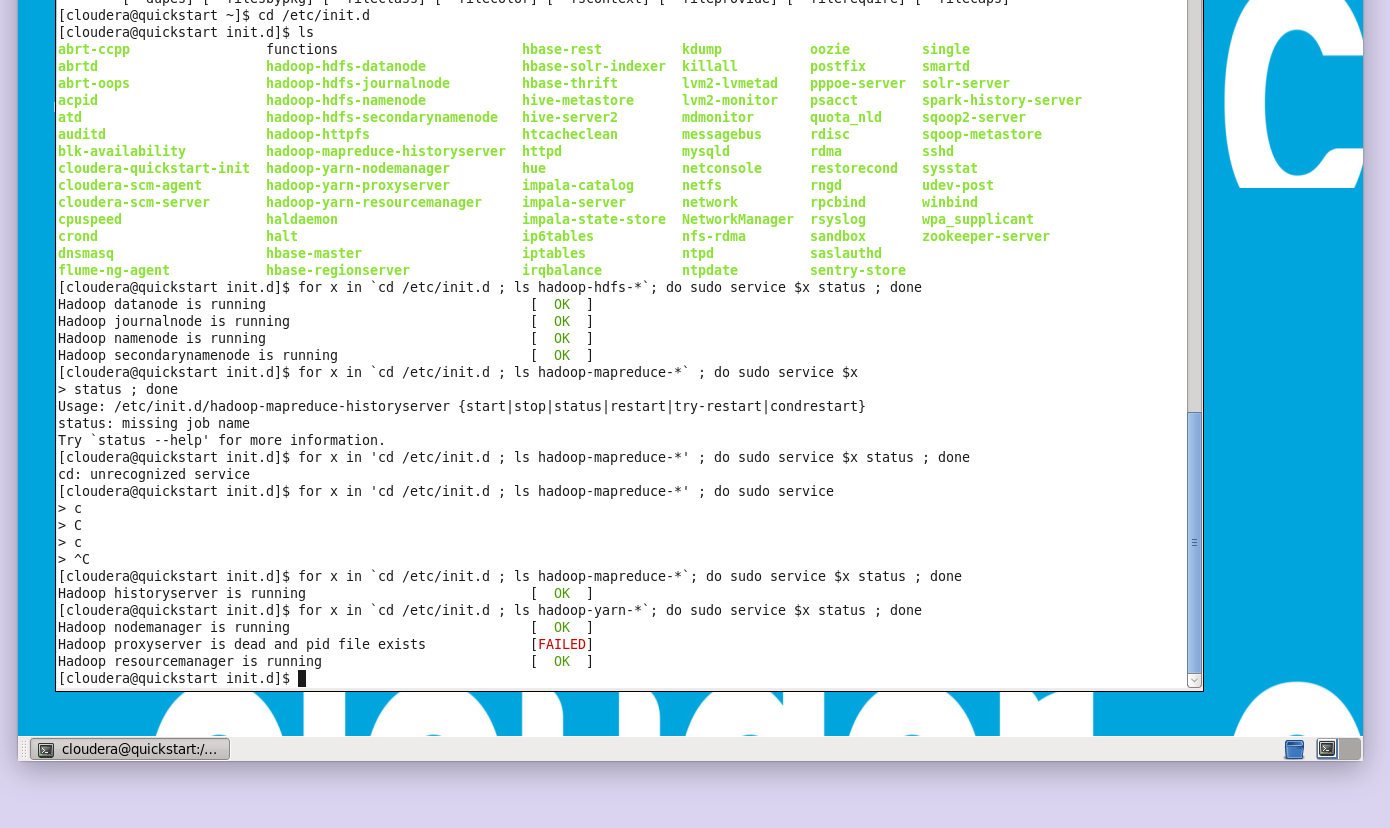
Implement solution for this assignment on Cloudera Quick Start VM with CDH 5.12.

**Problem 1.** Download Quick Start VM for CDH 5.12 from <https://www.cloudera.com/downloads/quickstart_vms/5-8.html>. Start the VM. Please assign to the VM as much memory as you can. Examine whether hadoop-hdfs-\* , hadoop-mapreduce-\* and hadoop-yarn-\* daemons are running. If those daemons are not running start all of them. If any of daemons fails to run, try to fix it.

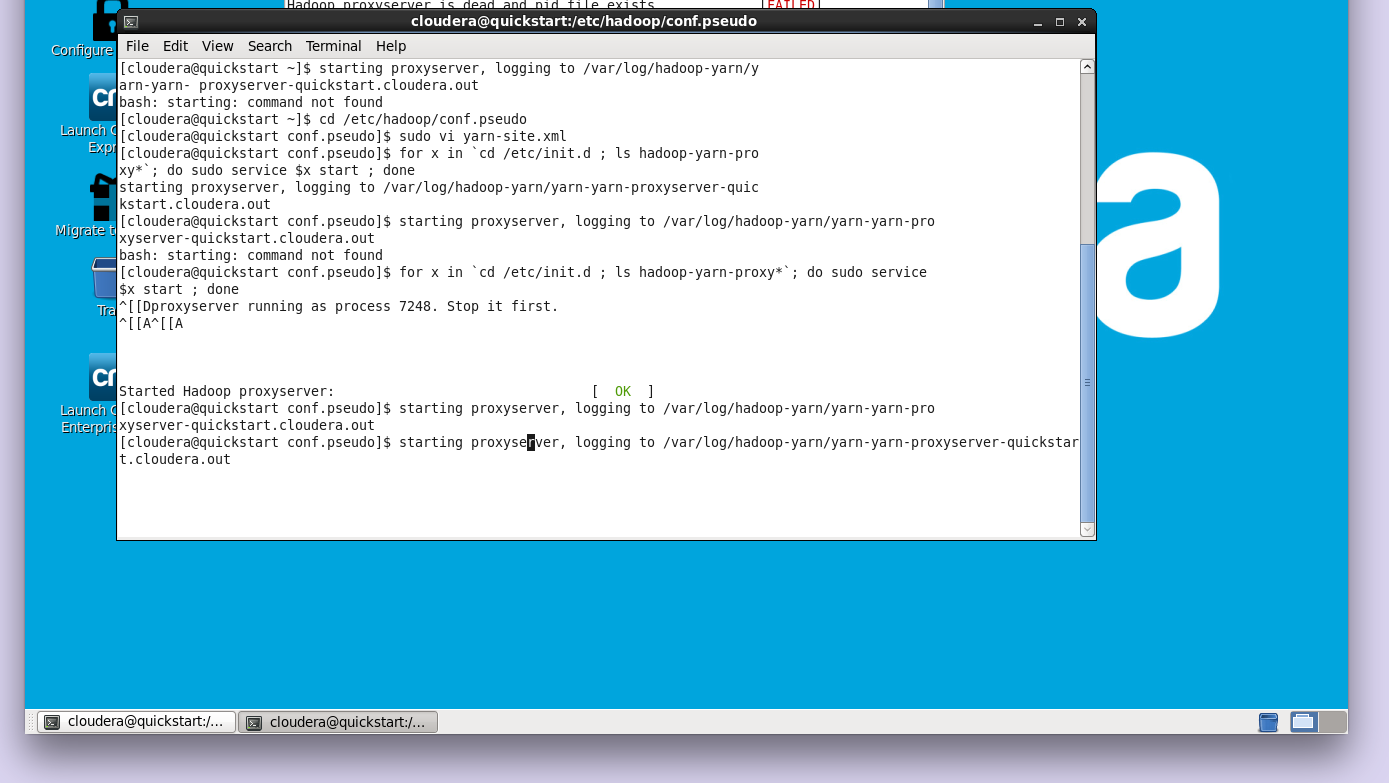
[10%]

Please ZOOM IN Word Doc to see better.



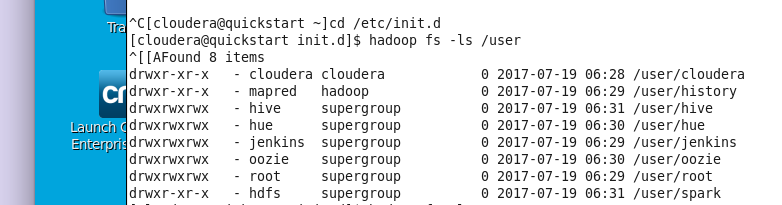


Proxyserver is failed. So I fixed it (below):

****

**Problem 2.** Examine whether there are HDFS directories for users: spark, hive, oozie, and cloudera. If the directories are present, find the content of those directories. If the directories are not present, create them. Please do not format the namenode.

[10%]



$ sudo -u hdfs hdfs dfs -ls -R /user/spark

$ sudo -u hdfs hdfs dfs -ls -R /user/hive

$ sudo -u hdfs hdfs dfs -ls -R /user/oozie

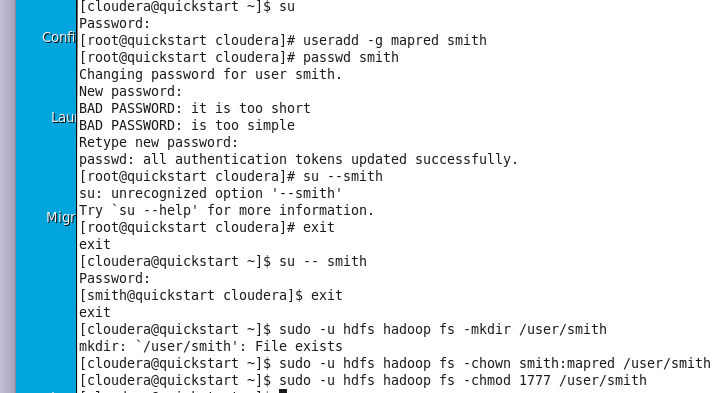
$ sudo -u hdfs hdfs dfs -ls -R /user/cloudera

#The results are very long, so I only put in the commands for answers

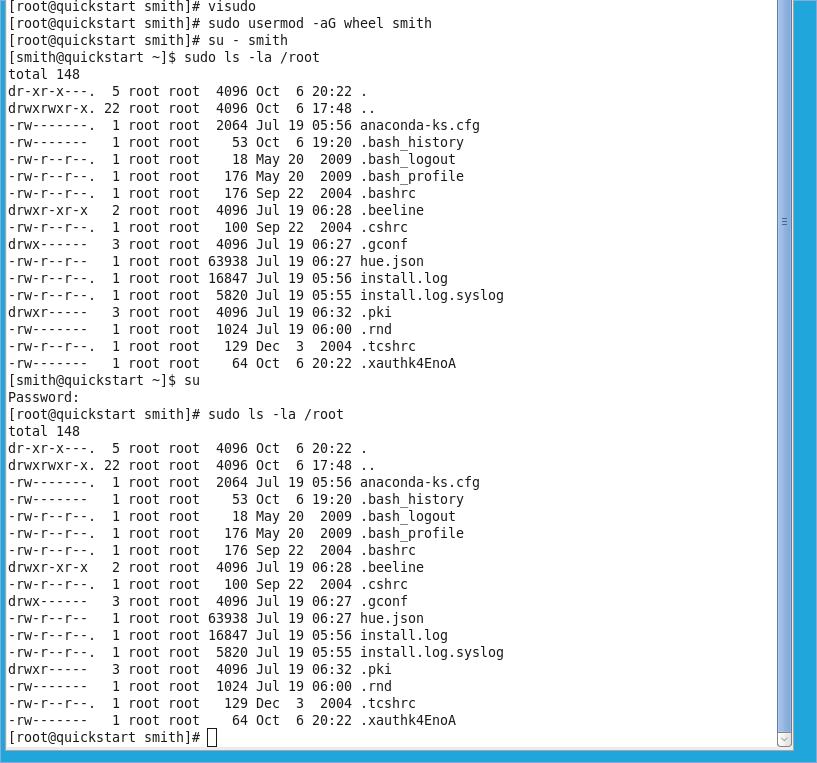
**Problem 3**. Create new Linux user smith. Make that user a member of the mapredLinux group. Make that user a sudo user. Create the home directory of user smith in HDFS. Download provided files bible.tar and shakespeare.tar. Unzip both tar files and copy the resulting files into HDFS directory input of user smith. As user smith run Hadoop grep on both bible and shakespeare texts. Every Hadoop run requires separate output directory. Examine content of first 20 lines of files generated by Hadoop grep.

[15%]

# Create new Linux user smith. Make that user a member of the mapredLinux group also create HDFS directory for user smith.



#Making smith a sudo user, also compared smith sudoers with root to verify smith as a sudo user:



#I extracted all-bible and all-shakespeare and leave them in Cloudera Desktop location.

# Download provided files bible.tar and shakespeare.tar. Unzip both tar files and copy the resulting files into HDFS directory input of user smith. As user smith run Hadoop grep on both bible and shakespeare texts. Every Hadoop run requires separate output directory. Examine content of first 20 lines of files generated by Hadoop grep.

[smith@quickstart init.d]$ hadoop fs -mkdir input

[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/Desktop/all-shakespeare.txt /user/smith/input

[smith@quickstart cloudera]$ hadoop fs -put /home/cloudera/Desktop/all-bible.txt /user/smith/input

[smith@quickstart cloudera]$ hadoop fs -ls input

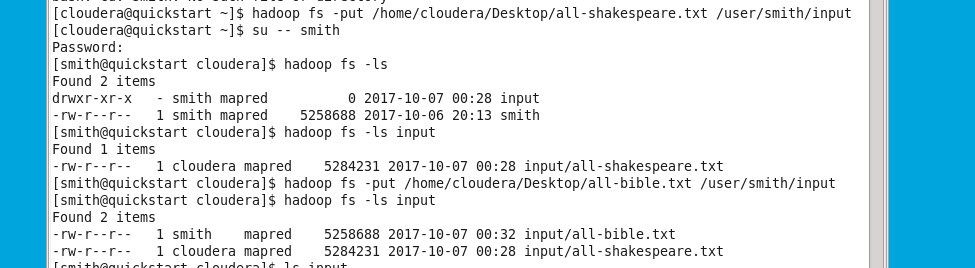
Found 2 items

-rw-r--r-- 1 smith mapred 5258688 2017-10-07 00:32 input/all-bible.txt

-rw-r--r-- 1 cloudera mapred 5284231 2017-10-07 00:28 input/all-shakespeare.txt

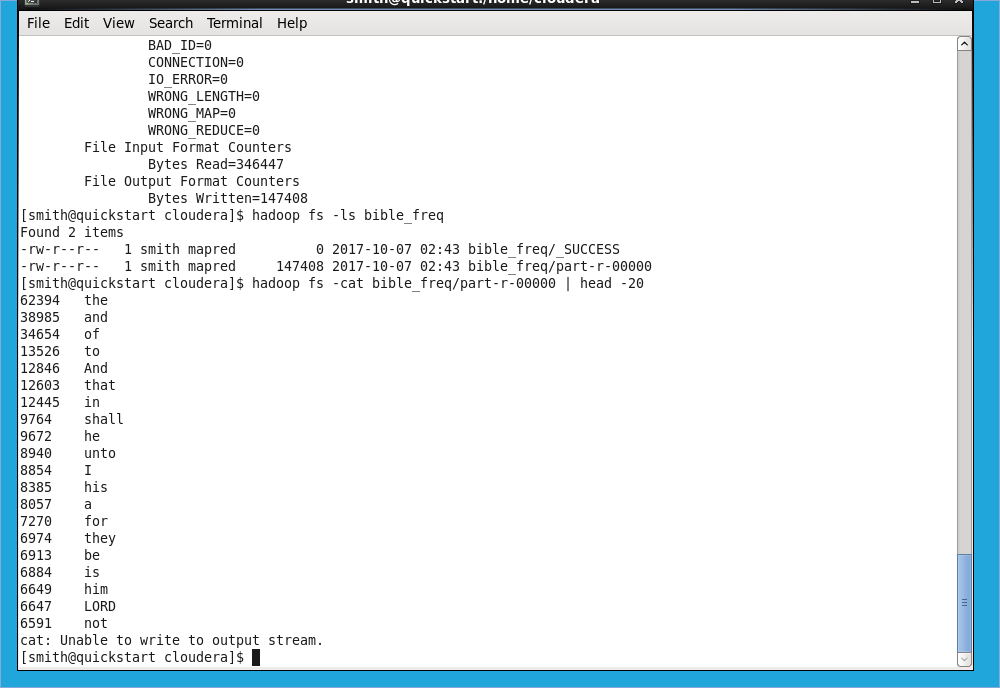
[smith@quickstart cloudera]$ hadoop fs -cat input/all-bible.txt | head

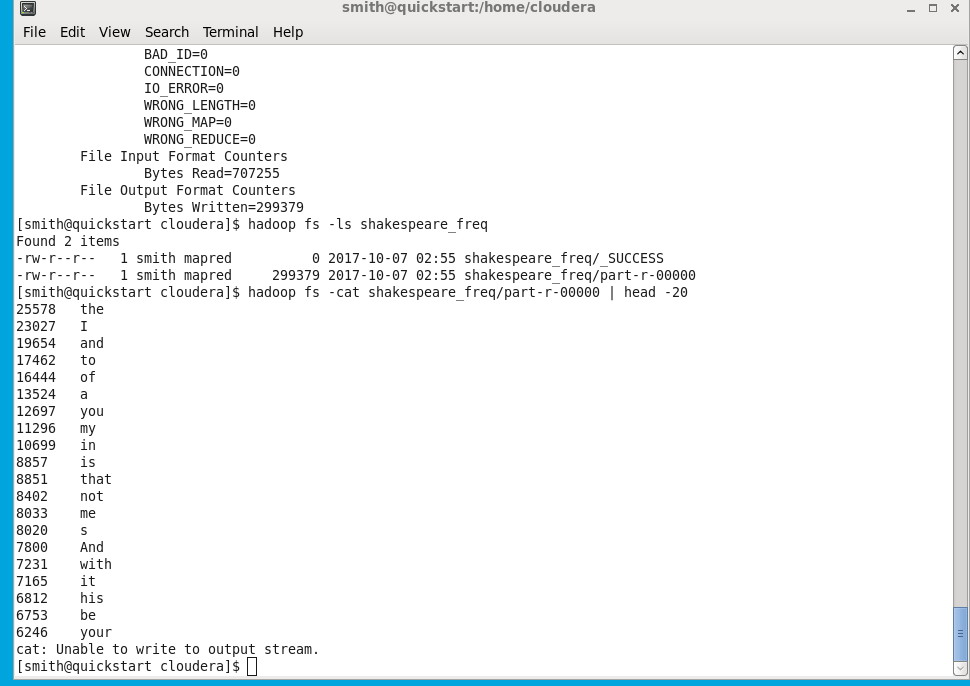
[smith@quickstart cloudera]$ hadoop fs -cat input/all-shakespeare.txt | head

****

[smith@quickstart cloudera]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input/all-bible.txt bible\_freq '\w+'

[smith@quickstart cloudera]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input/all-shakespeare.txt shakespeare\_freq '\w+'



****

**Problem 4**. Create your own version of “Hadoop grep” program using Spark. Compare your results with the results of Hadoop grep when applied to the texts of King James Bible, and all of Shakespeare’s works, contained in files bible.tar and shakespear.tar respectively. Notice small differences between results obtained by your Spark program and Hadoop grep. Try to explain what causes those differences. Save results of your Spark grep operations both in HDFS and on your local file system. You can implement your solution using one of interactive shells or a standalone program.

[20%]

Start pyspark

[cloudera@quickstart ~]$ cd /usr/lib/spark/sbin

[cloudera@quickstart sbin]$ sudo ./start-master.sh

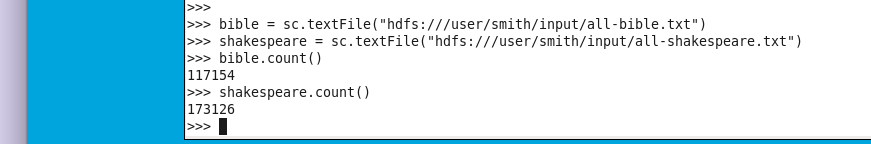
starting org.apache.spark.deploy.master.Master, logging to /var/log/spark/spark-root-org.apache.spark.deploy.master.Master-1-quickstart.cloudera.out

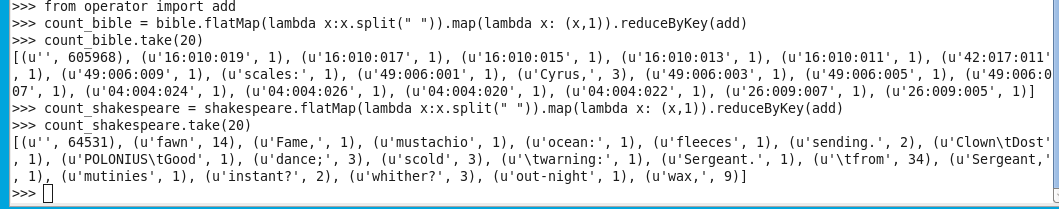
[cloudera@quickstart sbin]$ ~ cd

bash: /home/cloudera: is a directory

[cloudera@quickstart sbin]$ cd ~

[cloudera@quickstart ~]$ pyspark



****

# Saving Spark Object in File System:

**Screen%20Shot%202017-10-07%20at%2011.04.18%20AM.png**

***#*** *Answer:* ***What I have noticed the differences between using Hadoop grep and Spark program is that, Spark took less time to run commands. We can still run Spark then save the hdfs file (saveAsTextFile), that way we still secure the textFile as much as Hadoop grep, in addition to saving time. Hadoop grep program took me about 5 minutes to show 1 result, spark program took me about 5 seconds.***

**Problem 5**. Create your own tables KINGJAMES with columns for words and frequencies and insert into the table the result of your Spark grep program which produces word counts in file bible. Find all words in table KINGJAMES which start with letter “w” and are 4 or more characters long and appear more than 250 times. Write a query that will tell us the number of such words.

#Im’m creating bible and Shakespeare grep Job in cloudera. The same steps were done before but they were in user/smith.

[cloudera@quickstart ~]$ hadoop fs -mkdir input

[cloudera@quickstart ~]$ hadoop fs -put all-bible.txt input

[cloudera@quickstart ~]$ hadoop fs -put all-shakespeare.txt input

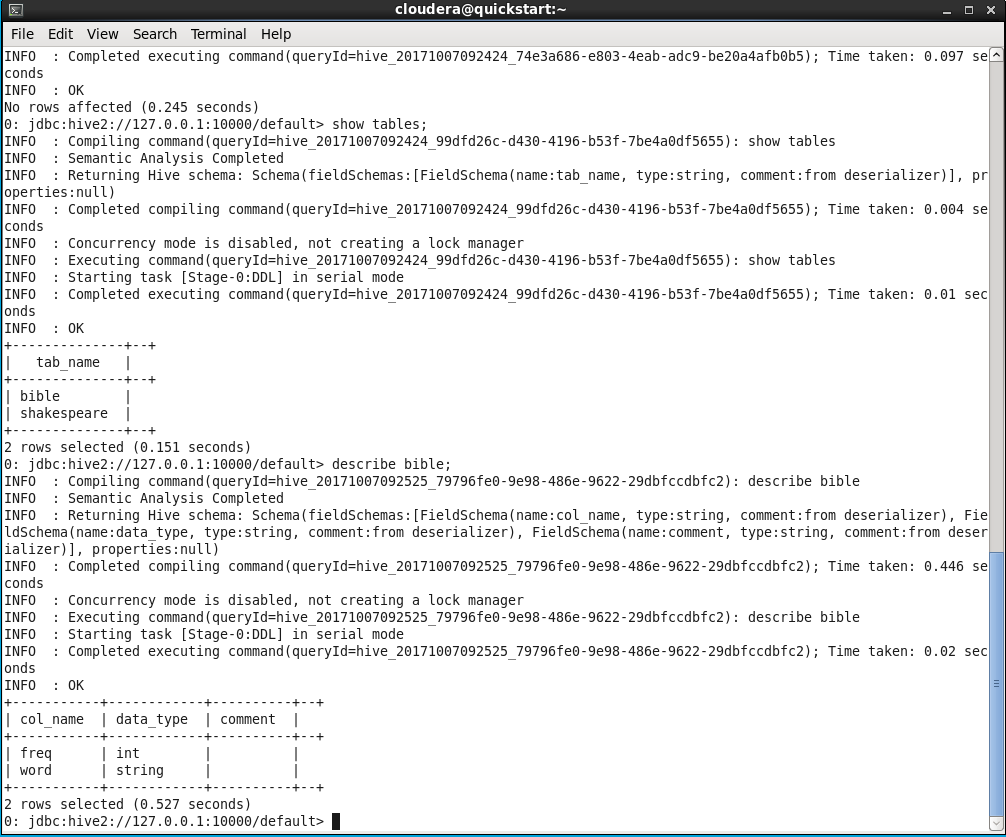
[cloudera@quickstart ~]$ hadoop fs -ls input

*Found 2 items*

*-rw-r--r-- 1 cloudera cloudera 5258688 2017-10-07 08:53 input/all-bible.txt*

*-rw-r--r-- 1 cloudera cloudera 5284231 2017-10-07 08:56 input/all-shakespeare.txt*

[cloudera@quickstart ~]$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input/all-bible.txt bible\_freq '\w+'

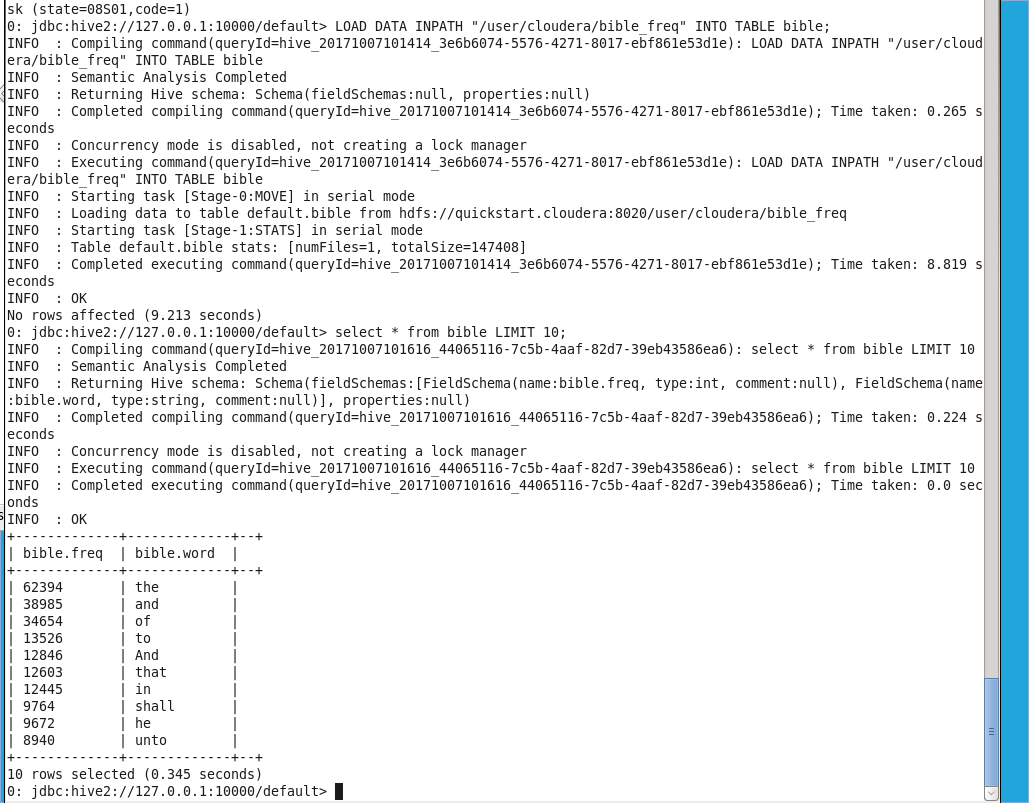


#Go back to cloudera, change permissions on HDFS directory /user/clouder/bible\_freq to 777 (gain permission)

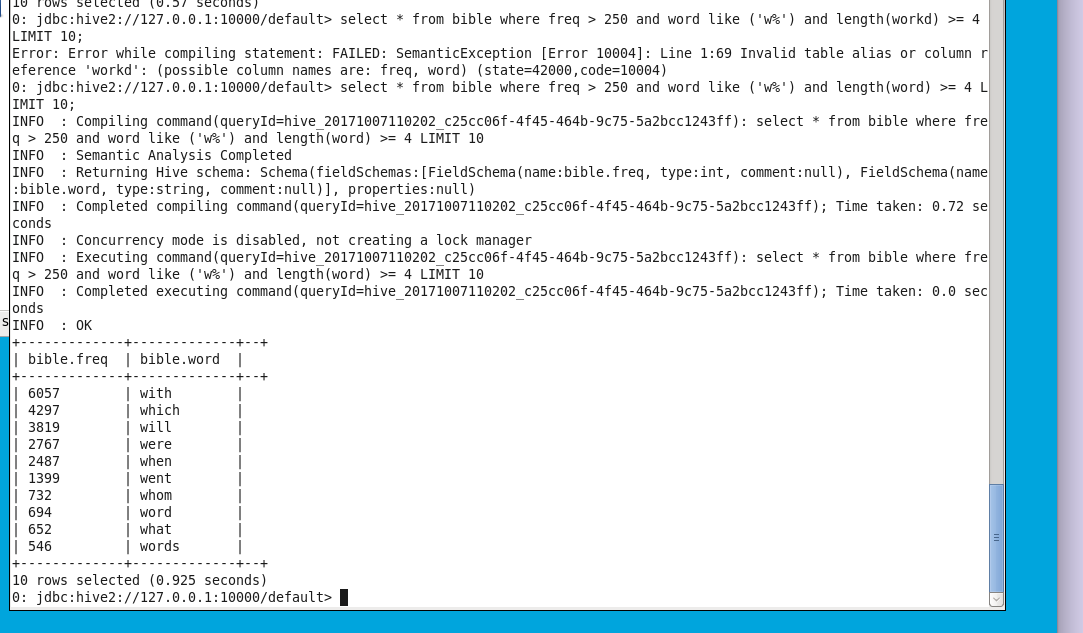
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod 777 /user/cloudera/bible\_freq

#Go back to Hive server prompt:

0: jdbc:hive2://127.0.0.1:10000/default> LOAD DATA INPATH "/user/cloudera/bible\_freq" INTO TABLE bible;



0: jdbc:hive2://127.0.0.1:10000/default> select \* from bible where freq > 250 and word like ('w%') and length(workd) >= 4 LIMIT 10;



**Problem 6**. Transfer content of your Hive KINGJAMES table to a Spark DataFrame. Perform the analysis from problem 6 using any available API in Spark. Please note that you are working with Spark 1.6.

[20%]

[root@quickstart cloudera]# hiveserver2 &

[root@quickstart cloudera]# cp /etc/hive/conf/hive-site.xml /etc/spark/conf

[root@quickstart cloudera]# pyspark

>>> hivecontext = HiveContext(sc)

>>> bibleDF = hivecontext.sql("select \* from bible")

>>> bibleDF.count()

14330

>>> bibleDF.first()

Row(freq=62394, word=u'the')

>>> bibleDF.printSchema()

root

|-- freq: integer (nullable = true)

|-- word: string (nullable = true)

**Problem 7.** Use Sqoop to transfer the content of MySQL database retail\_db which is present on the Cloudera VM into Hive. Demonstrate that new Hive tables are created and correspond to the original MySQL tables. Find the number of rows in each table. Compare those row counts with row counts in MySQL database.

[15%]

[cloudera@quickstart ~]$ mysql --user=retail\_dba -p

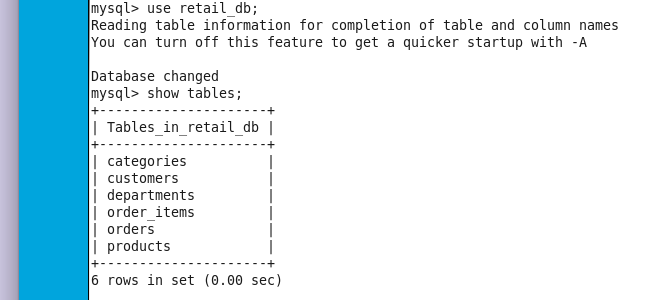
Enter password: cloudera

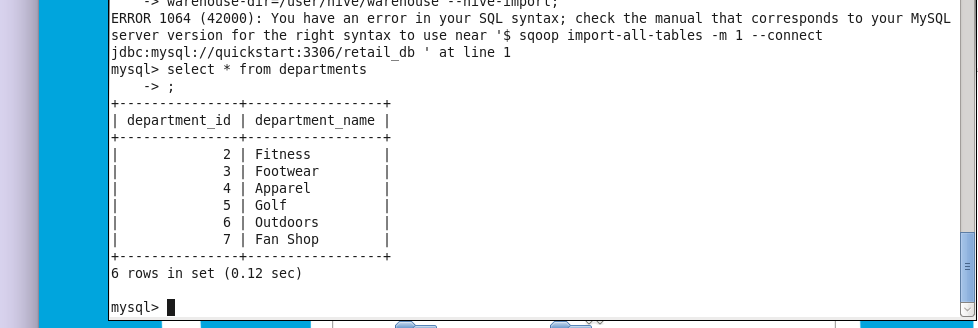
mysql> show databases;

mysql> use retail\_db;

mysql> show tables;

# columns and rows in mysql prompt:





# Exit out of mysql prompt

#Go to cloudera@quickstart:

$sqoop import-all-tables -m 1 –connect jdbc:mysql://quickstart:3306/retail\_db --username=retail\_dba --password=cloudera --compression-codec=snappy --as-parquetfile --warehouse-dir=/user/hive/warehouse --hive-import

#Go to hive server prompt:

0: jdbc:hive2://127.0.0.1:10000/default> show tables;