Instructions

For each answer, please include your answer as text, and any screenshot(s) which demonstrate your answer was executed. Most importantly, make sure to include evidence your answer is correct. This will most likely be a screenshot. If you had issues, problems, or had to make assumptions include them in your answer.

Your Answers:

1. Upload all the documents in **datasets/text** into a folder called **text** in HDFS. What HDFS command must you run to verify the files are there after they are uploaded? Your answer should consist of the command you typed to complete the task.

   Explored what is in these directories.
   $ ls datasets
   $ ls datasets/text

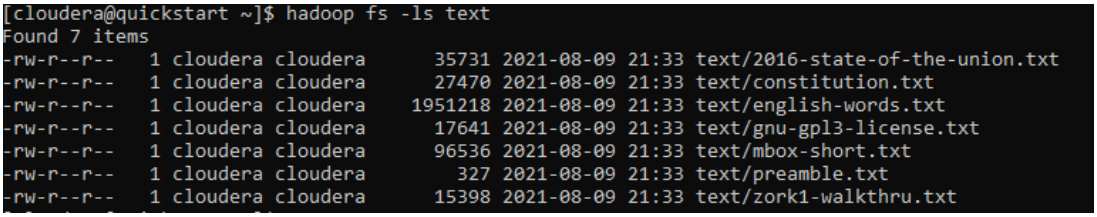
   Looked to see if the text folder already exists in HDFS.
   $ hadoop fs -ls
   $ hadoop fs -ls text

   Removed the text folder because it was already there.
   $ hadoop fs -rm -r text

   Crated a new text folder with nothing in it.
   $ hadoop fs -mkdir text

   Put the text files into the text folder.
   $ hadoop fs -put datasets/text/*.txt text/

   Check to make sure that the files are there.
   $ hadoop fs -ls text



2. In this part you will upload the **clickstream** dataset to HDFS. Specifically, create a **clickstream** folder in HDFS, then create a **logs** and **iplookup** folder inside the clickstream folder. Upload all of the ***.log** files from the **datasets/clickstream** local folder into **clickstream/logs** in HDFS. Upload the **ip_lookup.csv** file from the same folder into **clickstream/iplookup** on HDFS. Verify the files are there. Your answer should consist of the commands you typed to complete the task.

   Remove clickstream folder because it was already there.

$ hadoop fs -rm -r clickstream

Create a new clickstream folder with nothing in it.
$ hadoop fs -mkdir clickstream

Create a logs folder inside the clickstream folder
$ hadoop fs -mkdir clickstream/logs

Create a iplookup folder inside the clickstream folder
$ hadoop fs -mkdir clickstream/iplookup

Put the local *.log files into the clickstreams/logs folder in HDFS.
$ hadoop fs -put datasets/clickstream/*.log clickstream/logs

```
[cloudera@quickstart ~]$ hadoop fs -ls clickstream/logs
Found 3 items
-rw-r--r--   1 cloudera cloudera     137233 2021-08-09 21:45 clickstream/logs/u_ex160211.log
-rw-r--r--   1 cloudera cloudera      78658 2021-08-09 21:45 clickstream/logs/u_ex160212.log
-rw-r--r--   1 cloudera cloudera     105235 2021-08-09 21:45 clickstream/logs/u_ex160213.log
```

Put the local .csv file into the clickstream/iplookup folder in HDFS.
$ hadoop fs -put datasets/clickstream/ip_lookup.csv clickstream/iplookup

```
[cloudera@quickstart ~]$ hadoop fs -ls clickstream/iplookup
Found 1 items
-rw-r--r--   1 cloudera cloudera       1251 2021-08-09 21:46 clickstream/iplookup/ip_lookup.csv
```

3. Use the MapReduce examples:
`export MREX=/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar`
to perform a wordcount on the 2016 State of the Union address, saving the output to the HDFS folder **sotu2016**. Write down the commands to complete the task. How many times does the word **are** appear in the 2016 State of the Union address? Describe a process which could be done to make the wordcount more useful?
Remove the sotu2016 directory because it was already there.
$ hadoop fs -rm -r sotu2016

Set a variable called MREX for this mapreduce file.
$ export MREX=/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar

Run the word count and save it in a folder called sotu2016.
$ yarn jar $MREX wordcount text/2016-state-of-the-union.txt sotu2016/

Show the resulting file from the wordcount operation.
$ hadoop fs -ls sotu2016

```
[cloudera@quickstart ~]$ hadoop fs -ls sotu2016
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2021-08-09 22:59 sotu2016/_SUCCESS
-rw-r--r--   1 cloudera cloudera      19316 2021-08-09 22:59 sotu2016/part-r-00000
```

Print out the results of the wordcount file.
$hadoop fs -cat sotu2016/part-r-00000

```
worked  2
worker  1
workers 8
workers,        1
working 4
works   2
world   12
world,  3
world.  7
world's 1
worry,  1
worse   1
worst   1
worst-kept      1
worst?  1
worth.  1
```

4. Type the following command to import the **fudgemart_v3** database into the local **mysql** instance on the Hadoop client:
   `mysql -u root -p < ~/datasets/fudgemart/mysql.sql`
   The password is **cloudera**. Write down the commands you used to complete these tasks:
   Use the **sqoop** utility to verify there are tables in the database by listing them from the **fudgemart_v3** database. Next write a sqoop command to import Fudgemart products in the 'Clothing' department into a HDFS folder **/user/cloudera/fudgemart-clothing**

Use sqoop utility to show the databases
$ sqoop list-databases \
> --connect jdbc:mysql://cloudera --username=root --password=cloudera

```
[cloudera@quickstart ~]$ sqoop list-databases \
> --connect jdbc:mysql://cloudera --username=root --password=cloudera
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail
.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/08/10 00:31:11 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.7.0
21/08/10 00:31:11 WARN tool.BaseSqoopTool: Setting your password on the comman
d-line is insecure. Consider using -P instead.
21/08/10 00:31:11 INFO manager.MySQLManager: Preparing to use a MySQL streamin
g resultset.
information_schema
cm
firehose
fudgemart_v3
hue
metastore
mysql
nav
navms
oozie
retail_db
rman
sentry
twitter
```

Use sqoop utility to show the tables
$ sqoop list-tables
> --connect jdbc:mysql://cloudera/fudgemart_v3 –username=root –password=cloudera

```
[cloudera@quickstart ~]$ sqoop list-tables --connect jdbc:mysql://cloudera/fud
gemart_v3 --username=root --password=cloudera
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail
.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/08/10 00:34:04 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.7.0
21/08/10 00:34:04 WARN tool.BaseSqoopTool: Setting your password on the comman
d-line is insecure. Consider using -P instead.
21/08/10 00:34:05 INFO manager.MySQLManager: Preparing to use a MySQL streamin
g resultset.
fudgemart_creditcards
fudgemart_customer_creditcards
fudgemart_customers
fudgemart_departments_lookup
fudgemart_employee_timesheets
fudgemart_employees
fudgemart_jobtitles_lookup
fudgemart_order_details
fudgemart_orders
fudgemart_products
fudgemart_shipvia_lookup
fudgemart_vendors
[cloudera@quickstart ~]$
```

Delete the HDFS folder because it already exists
$ hadoop fs -rm -r fudgemart-clothing

Query the clothing products and store in a folder called fudgemart-clothing
sqoop import –connect jdbc:mysql://cloudera/fudgemart_v3 --username=root --password=cloudera --query “select * from fudgemart_products where product_department = ‘Clothing’ and \$CONDITIONS” --target-dir /user/cloudera/fudgemart-clothing --as-textfile --split-by product_add_date

* was getting a strange error message that I could not solve.

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql//cloudera/fudgemart_v3 --username=root --password=cloudera --query "select * from fudgemart_products where product_department = 'Clothing' and \$CONDITIONS" --target-dir /user/
cloudera/fudgemart-clothing/ --as-textfile --split-by product_add_date
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/08/10 01:46:20 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.7.0
21/08/10 01:46:20 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
21/08/10 01:46:20 ERROR tool.BaseSqoopTool: Got error creating database manager: java.io.IOException: No manager for connect string: jdbc:mysql//cloudera/fudgemart_v3
        at org.apache.sqoop.ConnFactory.getManager(ConnFactory.java:191)
        at org.apache.sqoop.tool.BaseSqoopTool.init(BaseSqoopTool.java:258)
        at org.apache.sqoop.tool.ImportTool.init(ImportTool.java:89)
        at org.apache.sqoop.tool.ImportTool.run(ImportTool.java:593)
        at org.apache.sqoop.Sqoop.run(Sqoop.java:143)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
        at org.apache.sqoop.Sqoop.runSqoop(Sqoop.java:179)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:218)
        at org.apache.sqoop.Sqoop.runTool(Sqoop.java:227)
        at org.apache.sqoop.Sqoop.main(Sqoop.java:236)

[cloudera@quickstart ~]$
```

5. Let's import HDFS data into MySQL. Record each command you type as your solution:
    a. Load **datasets/tweets/tweets.psv** into the HDFS folder **tweets**
    b. Login to MySQL: `mysql -u root -p` The password is **cloudera**. Create a database **twitter**
    c. Create a table called **tweets** inside the database **twitter** the table should have columns for id, timestamp, date time, username, and tweet_text.

Delete the tweets folder because it was already there.
$ hadoop fs -rm -r tweets

Create a new tweets folder with nothing in it
$ hadoop fs -mkdir tweets

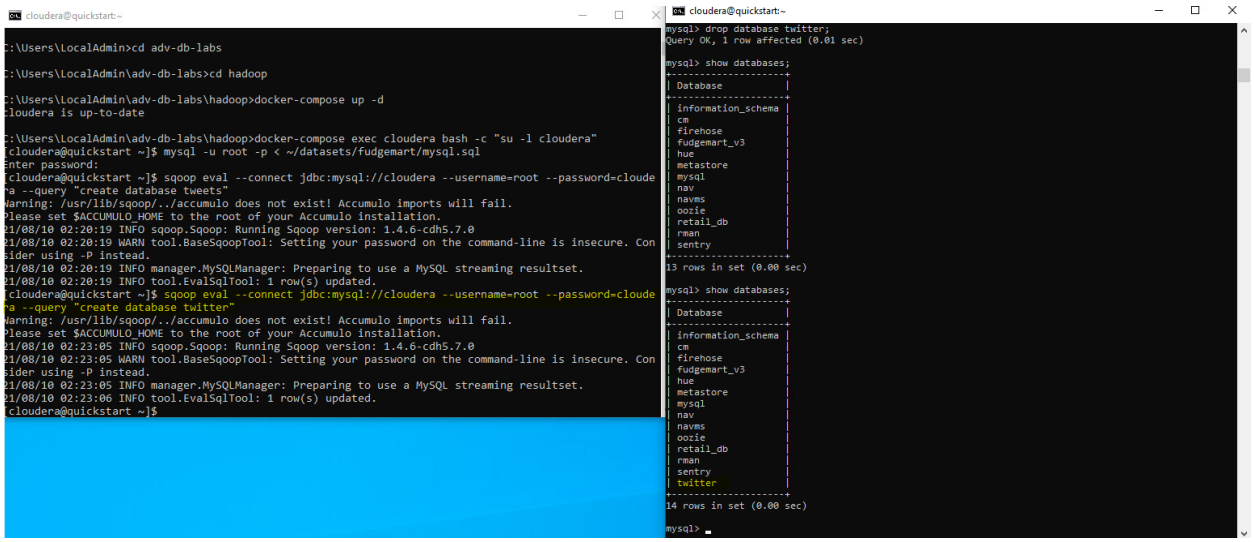
Put the tweets.psv file into the tweets folder in HDFS
$ hadoop fs -put datasets/tweets/tweets.psv tweets/

```
[cloudera@quickstart ~]$ hadoop fs -ls tweets
Found 1 items
-rw-r--r--   1 cloudera cloudera      25167 2021-08-10 01:52 tweets/tweets.psv
[cloudera@quickstart ~]$
```

Start

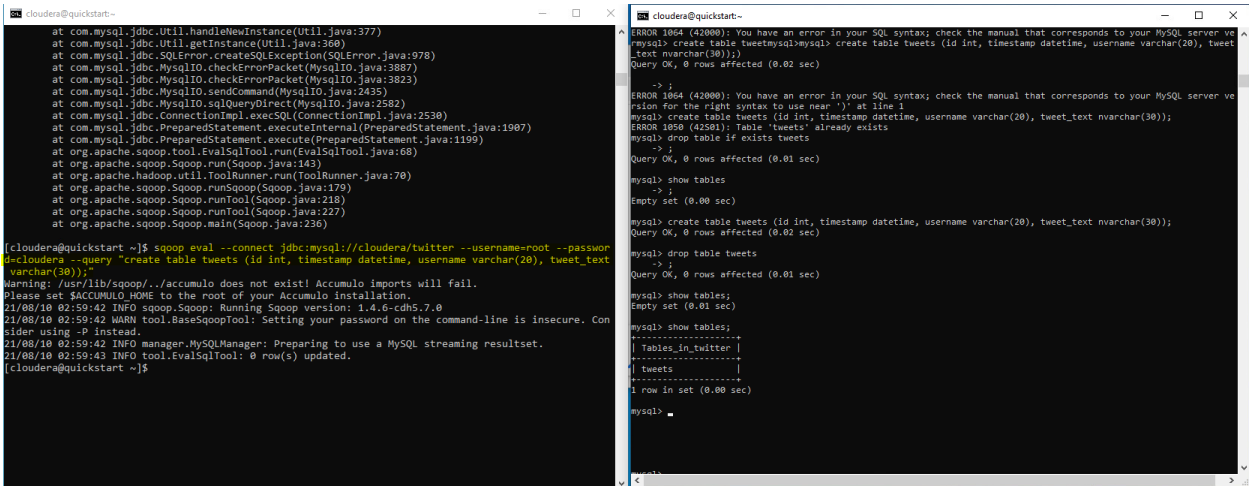
Delete the twitter database in MySQL server because it already existed.
drop database twitter;

Create twitter database in MySQL server via sqoop
$ sqoop eval --connect jdbc:mysql://cloudera --username=root --password=cloudera --query "create database twitter"



Create tweets table in MySQL server via sqoop
$ sqoop eval --connect jdbc:mysql://cloudera/twitter --username=root --password=cloudera --query “create table tweets (id int, timestamp datetime, username varchar(20), tweet_text varchar(30));”

* note I went back and re ran this with tweet_text varchar(200) to be able to fit

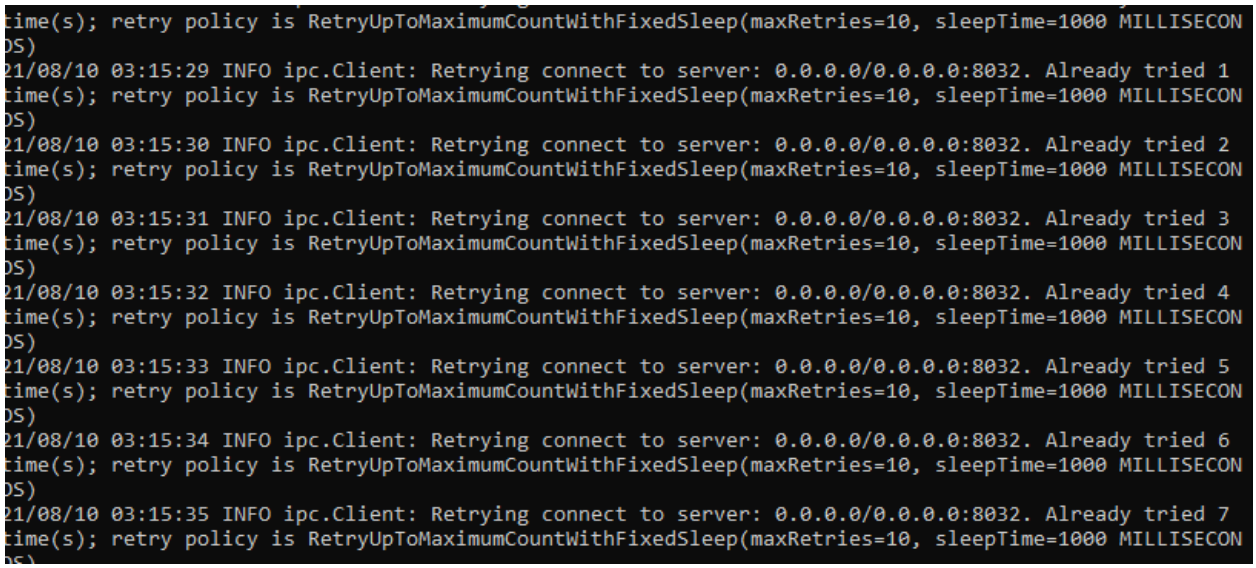


Export the data from HDFS into the MySQL table.
TIPS: If your SQOOP job fails it is likely due to the table constraints such as selecting a data type too small for the imported data. Try to insert a row in the table using a sample from the HDFS data. This will help you to ensure your chosen data types will work.

Take a look at whats in the tweets.psv file
$ hadoop fs -cat tweets/tweets.psv

Import the data into mysql
$ sqoop export --connect jdbc:mysql://cloudera/twitter --username=root --password=cloudera --table tweets --export-dir /tweets/tweets.psv/ --input-fields-terminated-by "|"

* not sure why just kept getting this and it kept looping



Seemed to be having some issues with the VM but I am not sure the cause. I troubleshooted for several hours but could not get it to work. I believe that my code is correct and something else is the problem.