Homework IST769 Unit C – Hadoop

## Agenda

1. Your Questions
2. Go Over Problem Set

## 1. Your Questions

1. Getting error while connecting to hive server

Connecting to jdbc:hive2://hive-server:10000/default

24/02/07 22:18:32 [main]: WARN jdbc.HiveConnection: Failed to connect to hive-server:10000

Could not open connection to the HS2 server. Please check the server URI and if the URI is correct, then ask the administrator to check the server status.

Error: Could not open client transport with JDBC Uri: jdbc:hive2://hive-server:10000/default: java.net.ConnectException: Connection refused (Connection refused) (state=08S01,code=0)

Beeline version 2.3.2 by Apache Hive

1. Q7, problems running it in Jupyter, “It looks like the version of Hive is not compatible in Pyspark”
2. Insert overwrite directory not working in Jupyter Lab.
3. Writing Hive table to HDFS
4. Connecting Hive Server to Jupyter Notebook

## 2. Go Over Problem Set C

1. Connect to the Linux shell on the **hive-server** (this is where the Hadoop client has been installed for you.) On this server you will see the **/datasets** folder is mounted. Load the:

a. customers/customers.csv,

b. customers/surveys.csv, and

c. tweets/tweets.psv into HDFS.

Specifically:

Source HDFS Location

customers/customers.csv /user/root/labc/customers/customers.csv

customers/surveys.csv /user/root/labc/surveys/surveys.csv

tweets/tweets.psv /user/root/labc/tweets/tweets.psv

Record the Hadoop commands you entered to complete this task. provide a screenshot of evidence these files are in HDFS. The screen shot can use the Hadoop client output or the HDFS website.

hadoop fs -mkdir /user/root/labc  
 hadoop fs -mkdir /user/root/labc/customers

hadoop fs -mkdir /user/root/labc/surveys

hadoop fs -mkdir /user/root/labc/tweets

hadoop fs -put /datasets/customers/customers.csv /user/root/labc/customers/

hadoop fs -ls /user/root/labc/customers/

hadoop fs -cat /user/root/labc/customers/customers.csv

hadoop fs -put /datasets/customers/surveys.csv /user/root/labc/surveys/

hadoop fs -put /datasets/tweets/tweets.psv /user/root/labc/tweets/

beeline -u jdbc:hive2://hive-server:10000/default

2. Create a Hive database called **labc**. In the **labc** database create an external hive table for the **tweets**. Your external table will point to the existing location on HDFS.

**NOTE:** You will need to view the tweets.psv file to see the format of the file before you can create the table schema correctly.

After you create the table write a SELECT query to display all of the tweets for a user a single user of your choice. Please include the HQL code you wrote to create and query the **tweets** table. Along with screenshots of a **describe tweets** command output along with your SELECT query output.

CREATE DATABASE labc;

USE labc;

beeline>

create external table labc.tweets\_ext

(id BIGINT, col2 string, datetime string, user\_id string, tweet string)

row format delimited

fields terminated by '|'

location '/user/root/labc/tweets';

3. In the **labc** database, let’s create an internal hive table for **customers**. After you create the table, use the LOAD command to move the data from the current HDFS location into the Hive data warehouse.

**NOTE 1:** if you screw up you will need to drop table and reload the file back into HDFS from step 1.

**NOTE 2:** there is a header row in this file, you might need to search the Hive docs on the web for how to exclude this first row.

When you have created the table and imported the data, provide the HQL code you entered to complete the task and provide screenshots of the **describe customers** command, a SELECT output to show data is there, and a screenshot on Web HDFS to show the data is located in **/user/hive/warehouse**.

CREATE TABLE CUSTOMERS\_int(

first\_name STRING,

last\_name STRING,

email STRING,

gender STRING,

ip\_address STRING,

city STRING,

state STRING,

total\_orders INT,

total\_purchased INT,

customer\_months INT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

TBLPROPERTIES("skip.header.line.count"="1");

LOAD DATA INPATH 'labc/customers/customers.csv' INTO TABLE labc.customers\_int;

Explain

select state, sum(total\_orders) as total\_orders

from customers\_int

where gender = 'F'

group by state;

spark.sql("select \* from labc.customers").show()

Df = pd.read\_Csv(“file./csv”, header=4)

4. Like the previous step, import the surveys.csv into a Hive internal table in the **labc** database called **surveys**. When you have created the table and imported the data, provide all the commands you entered to complete the task, a screenshot of the table description, the select statement output, and Web HDFS location.

CREATE TABLE labc.surveys\_int(

email STRING,

twitter\_username STRING,

marital\_status STRING,

household\_income INT,

own\_house STRING,

education STRING,

fav\_department STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

TBLPROPERTIES("skip.header.line.count"="1");

LOAD DATA INPATH 'labc/surveys/surveys.csv' INTO TABLE labc.surveys\_int;

5. Open Jupyter Lab. Create a new notebook called **labc.** Copy over the code from an example to create a spark session connected to Hive.

In a separate cell, write Spark SQL code to join customers to surveys on email address. Include all rows and columns and show output in the notebook. Provide a screenshot of the notebook cell with a reasonable amount of output (doesn’t need to be the entire set of rows and columns as that will be too large).

q5 = '''SELECT c.First

, c.Last

, c.Email

, c.Gender

, c.Last\_IP\_Address

, c.City

, c.State

, c.Total\_Orders

, c.Total\_Purchased

, c.Months\_Customer

, s.Twitter\_Username

, s.Marital\_Status

, s.Household\_Income

, s.Own\_Home

, s.Education

, s.Favorite\_Department

FROM customers\_i as c

FULL OUTER JOIN labc.surveys\_i as s

ON c.Email = s.Email;

'''

plan = spark.sql(q5)

plan.explain()

plan.show()

spark.sql("""

SELECT c.\*,s.\*

FROM labc.customers\_int c

JOIN labc.surveys\_int s

WHERE c.email = s.email

""").show(3)

6. The marketing department would like a dataset of customers / surveys for analysis. In a separate cell in the **labc** Juypter Notebook, write a Spark SQL query to create a hive table called **marketing** in **AVRO** file format from a SELECT query that once again joins customers and surveys on email addresses. Include the following columns in the new table: **Household Income**, **Education**, **Marital Status**, **Gender**, **City** and **State**.

Provide a screenshot of the Jupyter cell and output that creates the new table, and another of the cell and output of executing a SELECT on the table.

plan2 = plan.select(‘Household\_Income’, ‘Education’,’Marital\_Status’,’Gender’, ‘City’, ‘State’)

plan2.explain()

plan2.show()  
  
query = """

CREATE TABLE IF not EXISTS default.marketing

STORED AS AVRO

AS

SELECT

s.`Household Income` AS Household\_Income,

s.Education,

s.`Marital Status` AS Marital\_Status,

c.Gender,

c.City,

c.State

FROM default.customers c

JOIN default.surveys s ON c.Email = s.Email

"""

spark.sql("select \* from default.marketing").show()

7. Stupid marketing doesn’t know what they want! Now they would like the same query in the previous step, only output as a Comma-Delimited file instead of a Hive table. In a new Jupyter Lab cell, write Spark SQL to execute the Hive query but save the output back to HDFS in the folder **/user/root/marketing**.  
  
Provide a screenshot of the Spark code cell and its output, as well as a screenshot of the file on Web HD

#Write to HDFS

df.write.csv("webhdfs://namenode:50070/user/root/labc/marketing")

Notes:

spark = SparkSession.builder\

.master('local')\ #

.appName('jupyter-pyspark')\ #

.config('hive.metastore.uris', 'thrift://hive-metastore:9083')\ # Hive server?

.enableHiveSupport()\

.getOrCreate()