Lab: Exploring Spark SQL

About this Lab

Objective: Create DataFrame structure from Hive tables & HDFS files and utilize both the DataFrame API and SQL to refine returned data.

File locations: /root/devph/labs/Spark

Successful outcome: You will have successfully queries data from multiple DataFrame objects as well as joined them together.

Before you begin: Your HDP 2.3 cluster should be up and running within your VM.

Related lesson: ***Spark SQL and DataFrames***

Lab Steps

1) Run a query on an existing Hive table.

a. Load file into HDFS and create a Hive table mapping to it.

```
[root@sandbox Spark]# hdfs dfs -mkdir spark/cust_fav
[root@sandbox Spark]# hdfs dfs -put cust_fav.csv spark/cust_fav/
[root@sandbox Spark]# hive -f cust_fav.hive
[root@sandbox Spark]# hive -e 'select * from cust_fav limit 5;'
OK
Irvin Riesling
Owen Pinot Noir
August Sauvignon Blanc
Christian Merlot
Arlen Pinot Noir
Time taken: 3.023 seconds, Fetched: 5 row(s)
```

b. Create a DataFrame from querying the Hive table created in the prior step after starting up pyspark again.

```
>>> from pyspark.sql import HiveContext
>>> hc = HiveContext(sc)
>>> custFavDF = hc.sql("SELECT * FROM cust_fav")
>>> custFavDF.show(5)
+---------+---------------+
|cust_name|      wine_type|
+---------+---------------+
|    Irvin|       Riesling|
|     Owen|     Pinot Noir|
|   August|Sauvignon Blanc|
|Christian|         Merlot|
|    Arlen|     Pinot Noir|
+---------+---------------+
```

2) Use customer data from the prior lab to find average length of customers by gender and state.

a. Import the necessary Row definition and then create a RDD from the customer.csv file previously loaded into HDFS.

```
>>> from pyspark.sql import Row
>>> customerRaw =
sc.textFile("hdfs://sandbox:8020/user/root/spark/customer.csv")
>>> customerRaw.take(2)
[u'Irvin,M,Maryland,5.06', u'Owen,M,Illinois,2.01']
```

b. Break each long string representing a row from the input file into discrete customer records.

```
>>> customerRecords = customerRaw.map(lambda line:
line.split(','))
>>> customerRecords.take(2)
[[u'Irvin', u'M', u'Maryland', u'5.06'], [u'Owen', u'M',
u'Illinois', u'2.01']]
```

c. Convert that the RDD to a DataFrame.

```
>>> customerDF = customerRecords.map(lambda c: Row(name=c[0],
gender=c[1], state=c[2], length=float(c[3]))).toDF()
>>> customerDF.show(2)
+------+------+-----+--------+
|gender|length| name|   state|
+------+------+-----+--------+
|     M|  5.06|Irvin|Maryland|
|     M|  2.01| Owen|Illinois|
+------+------+-----+--------+
```

d. Search for the final results with the DataFrame API.

```
>>>
customerDF.select("gender","state","length").groupBy("gender","state").a
vg("length").show()
+------+-----------+-----------------+
|gender|      state|      AVG(length)|
+------+-----------+-----------------+
|     F|   Illinois|             7.49|
|     F| New Jersey|             4.04|
|     F|  Minnesota|            6.204|
|     F|   Michigan| 4.207142857142857|
|     M|    Indiana| 4.369999999999999|
|     M|   Maryland| 5.326250000000001|
|     M|   Nebraska|            6.516|
|     M|   Illinois| 4.858571428571428|
|     M| New Jersey|4.6499999999999995|
|     M|  Minnesota|           5.4375|
|     F|  Wisconsin|          6.29125|
|     M|   Michigan|5.0337499999999995|
|     F|Pennsylvania|            5.575|
|     F|       Ohio|             5.93|
|     F|       Iowa| 7.430000000000001|
|     M|  Wisconsin|             0.33|
|     M|Pennsylvania|            3.665|
|     M|       Ohio|             4.25|
|     M|       Iowa| 5.753333333333333|
|     F|   Maryland| 5.003749999999999|
+------+-----------+-----------------+
```

3) Join the prior two DataFrames.

a. Utilize the DataFrame API to perform the join.

```
>>> customerDF.join(custFavDF, customerDF.name ==
custFavDF.cust_name).show(5)
+------+------+--------+----------+---------+--------------+
|gender|length|    name|     state|cust_name|     wine_type|
+------+------+--------+----------+---------+--------------+
|     M|  5.06|   Irvin|  Maryland|    Irvin|      Riesling|
|     M|  2.01|    Owen|  Illinois|     Owen|    Pinot Noir|
|     M|  1.42|  August|  Illinois|   August|Sauvignon Blanc|
|     M|  8.17|Christian|New Jersey|Christian|        Merlot|
|     M|  2.24|   Arlen|   Indiana|    Arlen|    Pinot Noir|
+------+------+--------+----------+---------+--------------+
```

Result

Successful creation of a DataFrame from a Hive tables and a HDFS file; as well as joining these two DataFrames.