New Data Lake/Tutorial 3 ...

Tutorial 3: Working with Hive

This tutorial was built for BDCS-CE version 17.4.1 as part of the New Data Lake User Journey: here (https://github.com/oracle/learning-library/tree/master/workshops/journey2-new-data-lake). Questions and feedback about the tutorial: david.bayard@oracle.com (mailto:david.bayard@oracle.com)

```
Be sure to run the previous Tutorial: "Citi Bike New York Introduction and Setup"
```

This tutorial will illustrate using the Shell interpreter to run a few hive command lines and using the JDBC interpreter to run a few hive queries.

Contents

As a reminder, the documentation for BDCS-CE can be found here (https://docs.oracle.com/cloud/latest/big-data-compute-cloud/index.html)

About Apache Hive

Apache Hive is a component that enables the use of SQL against a variety of data, including data stored locally, in the Hadoop Distributed File

System (HDFS), and in the Object Store.

In this tutorial, we will use Hive to enable SQL queries against our Citi Bike .csv dataset.

Learn more about Hive here (https://hive.apache.org/)

Create a Hive Table

Our first step will be to define a Hive table on top of our CSV file. We will show 2 variations. The first example leverages a “managed” table where Hive manages the storage details (internally Hive will leverage HDFS storage). The second example leverages an “external” table where Hive will read the data directly from the Object Store.

We will use the hive command line running in the shell interpreter to create our tables.

Create a "managed" Bike Table from the CSV file (takes about 2 minutes)

```
%sh

DIRECTORY=citibike
FILENAME=201612-citibike-tripdata

echo "Data Set name (remove .zip or .csv)  :" $FILENAME
echo "-----------------------------------------------------------------"
echo ".."
echo "If this paragraph hangs on the SELECT count(*) FROM bike_trips, see the next paragraph for an explanation/resolution...."
echo ".."

cd $DIRECTORY

# run hive
hive <<EOF
DROP TABLE IF EXISTS bike_trips;

CREATE TABLE bike_trips (
TripDuration int,
StartTime timestamp,
StopTime timestamp,
StartStationID string,
StartStationName string,
StartStationLatitude string,
```

```
Data Set name (remove .zip or .csv)  : 201612-citibike-tripdata
------------------------------------------------------------------
..
If this paragraph hangs on the SELECT count(*) FROM bike_trips, see the next paragraph for an explanation/resolution....
..
WARNING: Use "yarn jar" to launch YARN applications.
Logging initialized using configuration in file:/etc/hive/2.4.2.0-258/0/hive-log4j.properties
hive> DROP TABLE IF EXISTS bike_trips;
OK
Time taken: 1.012 seconds
hive>
    > CREATE TABLE bike_trips (
    > TripDuration int,
    > StartTime timestamp,
    > StopTime timestamp,
    > StartStationID string,
    > StartStationName string,
    > StartStationLatitude string,
    > StartStationLongitude string,
```

```
    > EndStationID string,
    > EndStationName string,
    > EndStationLatitude string,
    > EndStationLongitude string,
    > BikeID int,
    > UserType string,
    > BirthYear int,
    > Gender int
    > )
    > ROW FORMAT delimited
    > FIELDS TERMINATED BY ',' ;
OK
Time taken: 1.228 seconds
hive>
    > LOAD DATA LOCAL INPATH '201612-citibike-tripdata.nh.csv' into table bike_t rips;
Loading data to table default.bike_trips
Table default.bike_trips stats: [numFiles=1, totalSize=136661199]
OK
Time taken: 1.011 seconds
hive>
    > create table bike_trips_small as select * from bike_trips limit 50000;
Query ID = zeppelin_20171115193124_2820bf60-9432-464e-9b01-97536bf47a92
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1510769091693_0002)
Map 1: -/-      Reducer 2: 0/1
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0(+1)/5  Reducer 2: 0/1
Map 1: 1(+0)/5  Reducer 2: 0/1
Map 1: 1(+1)/5  Reducer 2: 0/1
Map 1: 2(+1)/5  Reducer 2: 0/1
Map 1: 3(+1)/5  Reducer 2: 0/1
Map 1: 4(+0)/5  Reducer 2: 0/1
Map 1: 4(+1)/5  Reducer 2: 0/1
Map 1: 5/5      Reducer 2: 0(+1)/1
Map 1: 5/5      Reducer 2: 1/1
Moving data to: hdfs://dcbnov15c-bdcsce-1.compute-gse00002281.oraclecloud.internal:8020/apps/hive/warehouse/bike_trips_small
Table default.bike_trips_small stats: [numFiles=1, numRows=50000, totalSize=8363691, rawDataSize=8313691]
OK
hive> Time taken: 12.409 seconds
    > SELECT count(*) FROM bike_trips;
```

```
Query ID = zeppelin_20171115193136_8a73c6da-f331-4359-91ef-c3cc0e9cf7cf
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1510769091693_0002)
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0(+1)/5  Reducer 2: 0/1
Map 1: 1(+0)/5  Reducer 2: 0/1
Map 1: 1(+1)/5  Reducer 2: 0/1
Map 1: 2(+1)/5  Reducer 2: 0/1
Map 1: 3(+0)/5  Reducer 2: 0/1
Map 1: 3(+1)/5  Reducer 2: 0/1
Map 1: 4(+0)/5  Reducer 2: 0/1
Map 1: 4(+1)/5  Reducer 2: 0/1
Map 1: 5/5      Reducer 2: 0/1
Map 1: 5/5      Reducer 2: 1/1
OK
812192
Time taken: 3.448 seconds, Fetched: 1 row(s)
hive>
    > exit;
```

If the above paragraph seems to hang...

If you created the smallest BDCS-CE cluster (using the OC2M shape), then the hadoop scheduling system (YARN) can sometimes run out of available slots (virtual cores) to run applications. If the hive job takes more than 3 minutes, this is likely the cause. The solution is this:

- Go to the Jobs tab
- Look for the a running/processing non-Hive job, such as the Thrift server or the SparkSession labelled “Zeppelin”. Using the pop-up menu to the right of the job, choose to Abort the job.
- Then navigate back to this notebook

Create an "external" table against the Object Store (takes about 3 minutes)

```
%sh

CONTAINER=journeyC
DIRECTORY=citibike
FILENAME=201612-citibike-tripdata
```

```
Object Storage Container Name        : journeyC
Data Set name (remove .zip or .csv)  : 201612-citibike-tripdata
---------------------------------------------------------------
WARNING: Use "yarn jar" to launch YARN applications.
Logging initialized using configuration in file:/etc/hive/2.4.2.0-258/0/hive-log4j.properties
```

```
hive> DROP TABLE bike_trips_objectstore;
OK
Time taken: 0.931 seconds
hive>
    > CREATE external TABLE bike_trips_objectstore (
    > TripDuration int,
    > StartTime timestamp,
    > StopTime timestamp,
    > StartStationID string,
    > StartStationName string,
    > StartStationLatitude string,
    > StartStationLongitude string,
    > EndStationID string,
    > EndStationName string,
    > EndStationLatitude string,
    > EndStationLongitude string,
    > BikeID int,
    > UserType string,
    > BirthYear int,
    > Gender int
    > )
    > ROW FORMAT delimited
    > FIELDS TERMINATED BY ','
    > location 'swift://journeyC.default/citibike/modified/';
OK
Time taken: 2.303 seconds
hive>
    >
    > SELECT count(*) FROM bike_trips_objectstore;
Query ID = zeppelin_20171115193229_8e998e71-a7b3-402a-a121-f36b88e5d43a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1510769091693_0003)
Map 1: -/-      Reducer 2: 0/1
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0(+1)/5  Reducer 2: 0/1
Map 1: 1(+0)/5  Reducer 2: 0/1
Map 1: 1(+1)/5  Reducer 2: 0/1
Map 1: 2(+0)/5  Reducer 2: 0/1
Map 1: 2(+1)/5  Reducer 2: 0/1
```

```
Map 1: 3(+0)/5  Reducer 2: 0/1
Map 1: 3(+1)/5  Reducer 2: 0/1
Map 1: 4(+0)/5  Reducer 2: 0/1
Map 1: 4(+1)/5  Reducer 2: 0/1
Map 1: 5/5      Reducer 2: 0/1
Map 1: 5/5      Reducer 2: 0(+1)/1
Map 1: 5/5      Reducer 2: 1/1
OK
812192
Time taken: 18.677 seconds, Fetched: 1 row(s)
hive>
    > exit;
```

Show the Hive tables defined

```
%sh

hive  <<EOF
show tables;
EOF
```

```
WARNING: Use "yarn jar" to launch YARN applications.
Logging initialized using configuration in file:/etc/hive/2.4.2.0-258/0/hive-log4j.properties
hive> show tables;
OK
bike_trips
bike_trips_objectstore
bike_trips_small
Time taken: 1.126 seconds, Fetched: 3 row(s)
hive>
```

Sample query via Hive Command Line

```
%sh

hive  <<EOF
select
 case when a.gender=1 then 'Male' when a.gender=2 then 'Female' else 'unknown' end gender ,
        a.trip_count
from (select gender, count(*) trip_count from bike_trips
```

```
WARNING: Use "yarn jar" to launch YARN applications.
Logging initialized using configuration in file:/etc/hive/2.4.2.0-258/0/hive-log4j.properties
hive> select
    >  case when a.gender=1 then 'Male' when a.gender=2 then 'Female' else 'unkn own' end gender ,
    >         a.trip_count
    > from (select gender, count(*) trip_count from bike_trips
    > group by gender) a;
Query ID = zeppelin_20171115193443_ed948f09-b59a-4273-b09e-6e0da5e8cc67
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1510769091693_0005)
Map 1: -/-      Reducer 2: 0/3
Map 1: 0/5      Reducer 2: 0/3
Map 1: 0(+1)/5  Reducer 2: 0/3
Map 1: 1(+1)/5  Reducer 2: 0/3
Map 1: 2(+0)/5  Reducer 2: 0/3
Map 1: 2(+1)/5  Reducer 2: 0/3
Map 1: 3(+0)/5  Reducer 2: 0/3
Map 1: 3(+1)/5  Reducer 2: 0/3
Map 1: 4(+0)/5  Reducer 2: 0/3
Map 1: 4(+1)/5  Reducer 2: 0/3
Map 1: 5/5      Reducer 2: 0(+1)/3
Map 1: 5/5      Reducer 2: 1(+0)/3
Map 1: 5/5      Reducer 2: 2(+0)/3
Map 1: 5/5      Reducer 2: 3/3
OK
```

Working with the JDBC(Hive) interpreter

Zeppelin includes a JDBC interpreter that allows you run a query as a paragraph and do some nice formating of the results. In BDCS-CE, the JDBC interpreter has been pre-configured to connect to Hive.

You can work with the JDBC interpreter and connect to Hive like this:

```
%jdbc(hive)
select * from my_table;
```

```
%jdbc(hive)
show tables
```

tab_name
bike_trips
bike_trips_objectstore
bike_trips_small

```
%jdbc(hive)
describe bike_trips
```

col_name	data_type
tripduration	int
starttime	timestamp
stoptime	timestamp
startstationid	string
startstationname	string
startstationlatitude	string
startstationlongitude	string
endstationid	string
endstationname	string

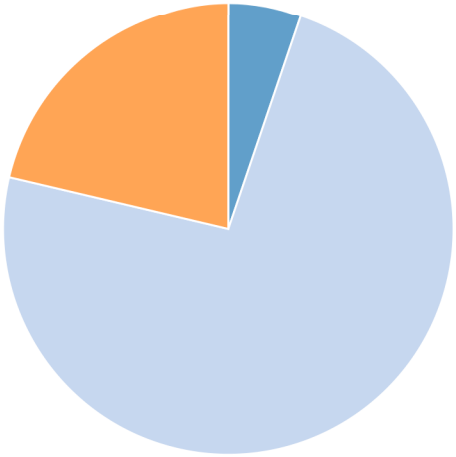
```
%jdbc(hive)
show create table bike_trips
```

createtab_stmt
CREATE TABLE `bike_trips`(
`tripduration` int,

createtab_stmt
`starttime` timestamp,
`stoptime` timestamp,
`startstationid` string,
`startstationname` string,
`startstationlatitude` string,
`startstationlongitude` string,
`endstationid` string,
`endstationname` string,
`endstationlatitude` string,
`endstationlongitude` string,
`bikeid` int,
`usertype` string,
`birthyear` int,
`gender` int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

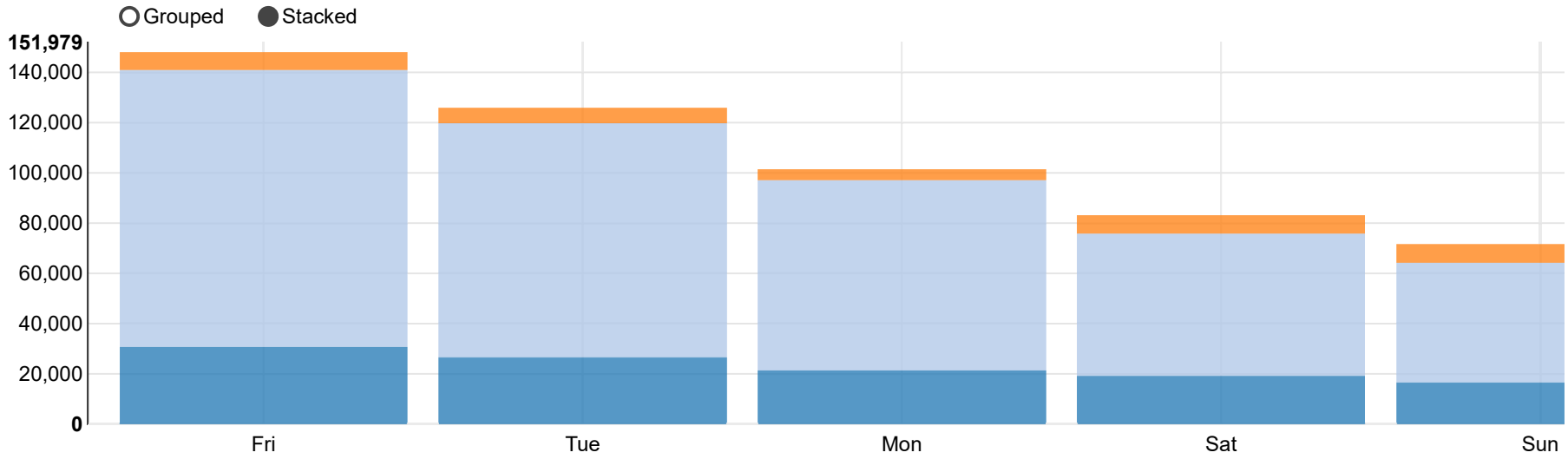
Query to show Data as a Pie Chart (Riders by Gender)

```
%jdbc(hive)
```



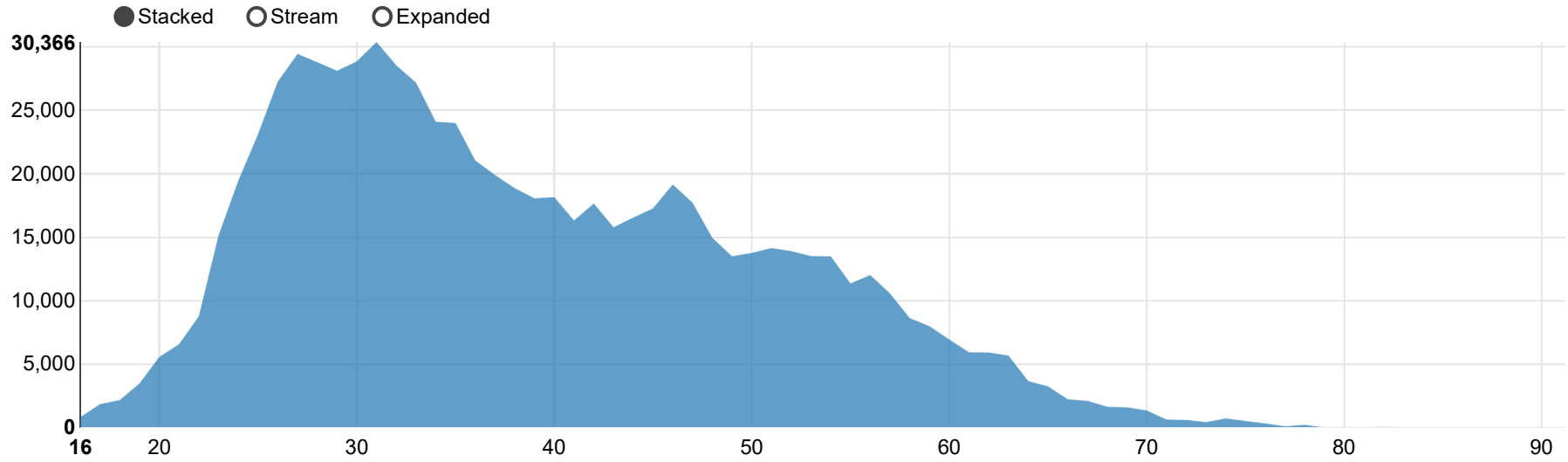
Query to Show a Chart with Stacks or Groupings (Riders by Gender by Day)

```
%jdbc(hive)

select gender, dayofweek, count(*)
from (    select date_format(`StartTime`,"E") dayofweek,
          case when gender=1 then 'Male' when gender=2 then 'Female' else 'Unknown' end gender
from bike_trips) bike_times
group by gender, dayofweek
```



Hive Query to Show a Chart (Age Distribution)

```
%jdbc(hive)

select (2016 - a.birthyear) age, count(*) number_trips from bike_trips as a
where birthyear is not null
group by birthyear
```



Next Steps

So far, we

1) have downloaded a Citi Bike data zip and created csv files. Then we stored the data into the Object Storage.
2) set up a Hive table using the Hive command line
3) queried Hive via Zepellin's JDBC(Hive) Interpreter

In the next tutorial we use Spark for processing and query.

Tip: Running Hive and Zeppelin/Spark on a 2 OCPUs instance

Some of you may be running this tutorial on the smallest configuration size (2 OCPUs). If you are using this small shape, there can be scenarios where your Hive commands get "Accepted" but do not progress to "Processing". You can see this behavior in the Jobs tab. If this happens, then your Hive commands will hang indefinitely. The reason this happens is that with 2 OCPUs, all of the available processing threads in the YARN resource manager can be occupied by other jobs. There is a simple solution if you see this happen: simply go to the Jobs tab, find the "Zeppelin" job or the "Thrift server" job, and abort it. You will find that your Hive job will then move from "Accepted" to "Processing" and then finish. The

"Zeppelin" job will relaunch itself automatically the next time you run something with the Spark Interpreter inside Zeppelin.

An alternate solution is to add another node to your BDCS-CE cluster. BDCS-CE is scalable, so you can add a node which will provide additional processing threads for YARN to work with. You can also drop the node later if you want.

Change Log

September 7, 2017 - Confirmed it works on 17.3.5-20. Tweaked the hive managed table script
August 23, 2017 - A few minor tweaks
August 13, 2017 - Confirmed it works on 17.3.3-20
August 11, 2017 - Journey v2
July 28, 2017 - Confirmed that it works on 17.3.1-20

```
%md
```