**HOW TO PROCESS DATA WITH APACHE HIVE**

* [Step 1: Download The Data](https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/#download-the-data)
* [Step 2: Upload The Data Files](https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/#upload-the-data-files)
* [Step 3: Start the Hive View](https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/#start-the-hive-view)

## OUR DATA PROCESSING TASK

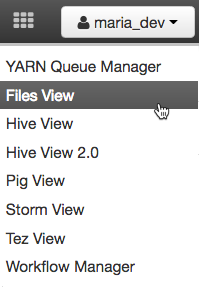
We have several files of truck driver statistics and we are going to bring them into Hive and do some simple computing with them. We are going to compute the sum of hours and miles logged driven by a truck driver for an year. Once we have the sum of hours and miles logged, we will extend the script to translate a driver id field into the name of the drivers by joining two different tables.

## STEP 1: DOWNLOAD THE DATA

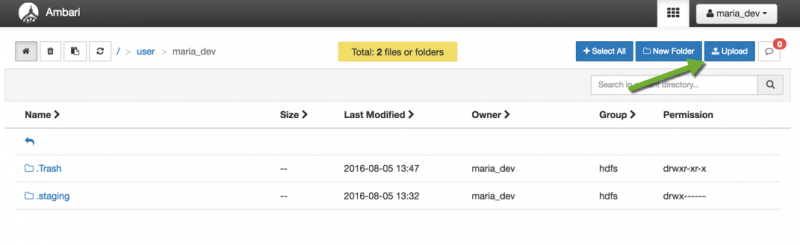
Download the driver data file from [here](https://raw.githubusercontent.com/hortonworks/data-tutorials/master/tutorials/hdp/how-to-process-data-with-apache-hive/assets/driver_data.zip" \t "_blank).  
Once you have the file you will need to unzip the file into a directory. We will be uploading two csv files – drivers.csv and timesheet.csv.

## STEP 2: UPLOAD THE DATA FILES

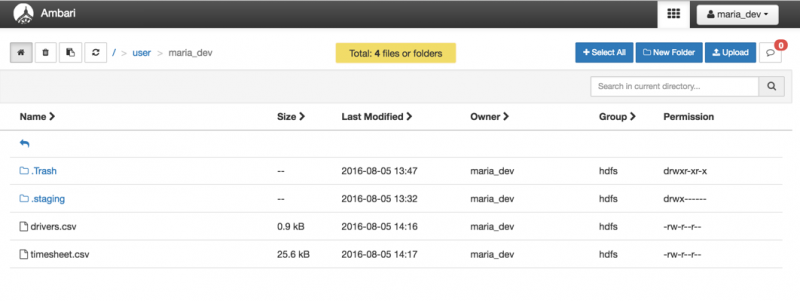
We start by selecting the HDFS Files view from the Off-canvas menu at the top. The HDFS Files view allows us to view the Hortonworks Data Platform(HDP) file store. This is separate from the local file system. For the Hortonworks Sandbox, it will be part of the file system in the Hortonworks Sandbox VM.



Navigate to /user/maria\_dev and click on the Upload button to select the files we want to upload into the Hortonworks Sandbox environment.

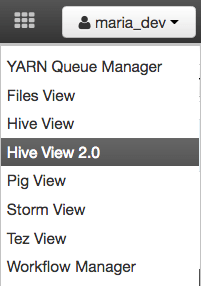


Click on the browse button to open a dialog box. Navigate to where you stored the drivers.csv file on your local disk and select drivers.csv and click open. Do the same thing for timesheet.csv. When you are done you will see there are two new files in your directory.



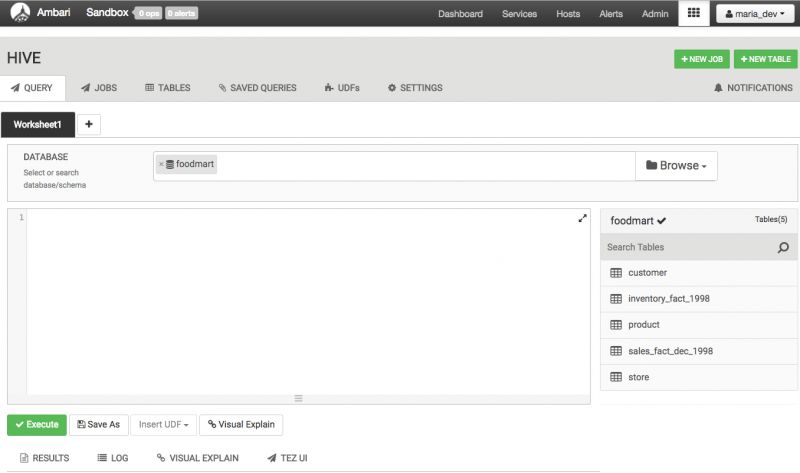
## STEP 3: START THE HIVE VIEW

Let’s open the Hive View 2.0 by clicking on the Views icon on the top bar. Hive View 2.0 provides a user interface to the Hive data warehouse system for Hadoop.



### 3.1 EXPLORE THE HIVE USER INTERFACE

Below is the Query Editor. A query may span multiple lines. At the bottom, there are buttons to Execute the query, Visual Explain the query, Save As the query with a name and to open a new Worksheet window for another query.

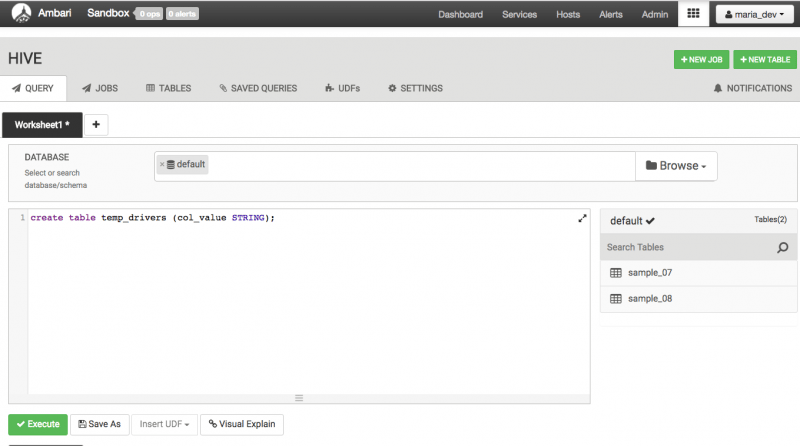


Once the script is complete all data objects are deleted unless you stored them. In the case of Hive we are operating on the Apache Hadoop data store. Any query you make, table that you create, data that you copy persists from query to query. You can think of Hive as providing a data workbench where you can examine, modify and manipulate the data in Apache Hadoop. So when we perform our data processing task we will execute it one query or line at a time. Once a line successfully executes you can look at the data objects to verify if the last operation did what you expected. All your data is live, compared to Pig, where data objects only exist inside the script unless they are copied out to storage. This kind of flexibility is Hive’s strength. You can solve problems bit by bit and change your mind on what to do next depending on what you find.

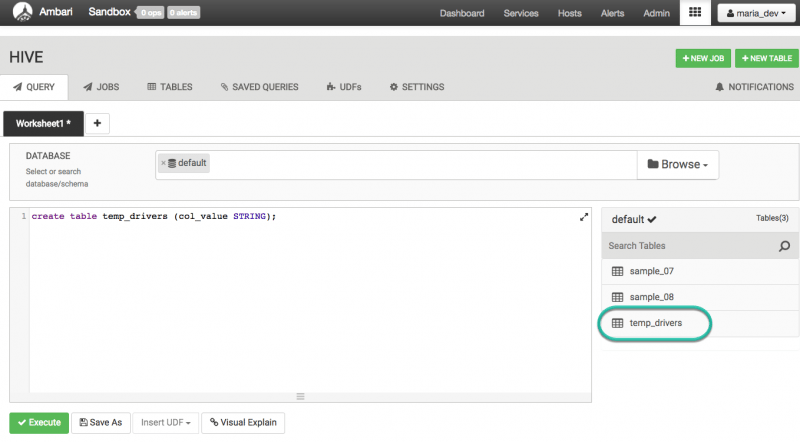
### 3.2 CREATE TABLE TEMP\_DRIVERS

The first task we will do is create a table to hold the data. We will type the query into the Query Editor. Once you have typed in the query hit the Execute button at the bottom.

create table temp\_drivers (col\_value STRING);



The query does not return any results because at this point we just created an empty table and we have not copied any data in it.  
Once the query has executed we can refresh the Database by re-selecting the Database. We will see the new table called temp\_drivers.

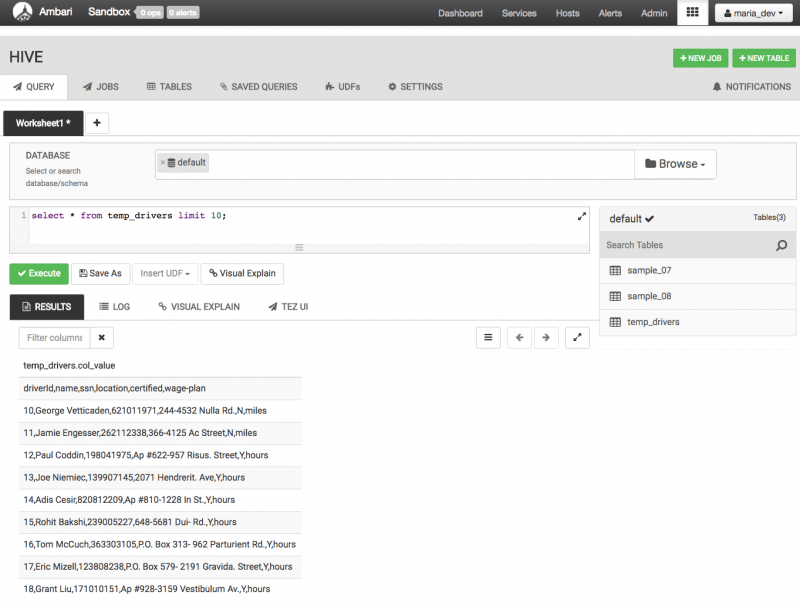


### 3.3 CREATE QUERY TO POPULATE HIVE TABLE TEMP\_DRIVERS WITH DRIVERS.CSV DATA

The next line of code will load the data file drivers.csv into the table temp\_drivers.

LOAD DATA INPATH '/user/maria\_dev/drivers.csv' OVERWRITE INTO TABLE temp\_drivers;

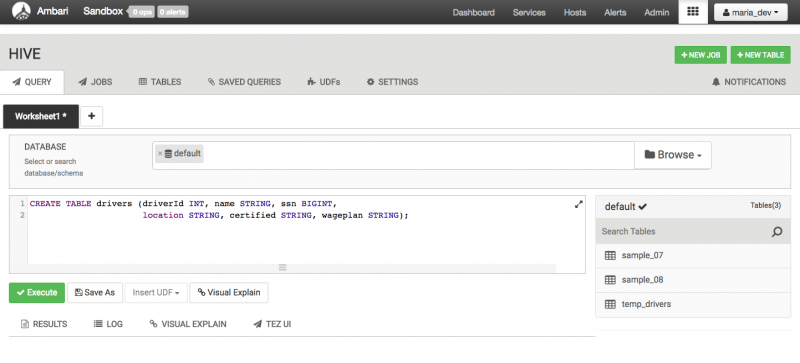
After executing LOAD DATA we can see table temp\_drivers was populated with data from drivers.csv. Note that Hive consumed the data file drivers.csv during this step. If you look in the File Browser you will see drivers.csv is no longer there.



### 3.4 CREATE TABLE DRIVERS

Now that we have read the data in we can start working with it. The next thing we want to do extract the data. So first we will type in a query to create a new table called drivers to hold the data. That table will have six columns for driverId,name, ssn, location, certified and the wage-plan of drivers.

CREATE TABLE drivers (driverId INT, name STRING, ssn BIGINT, location STRING, certified STRING, wageplan STRING);



### 3.5 CREATE QUERY TO EXTRACT DATA FROM TEMP\_DRIVERS AND STORE IT TO DRIVERS

Then we extract the data we want from temp\_drivers and copy it into drivers. We will do this with a regexp pattern. To do this we are going to build up a multi-line query. The six regexp\_extract calls are going to extract the driverId, name,ssn, location, certified and the wage-plan fields from the table temp\_drivers. When you are done typing the query it will look like this. Be careful as there are no spaces in the regular expression pattern.

insert overwrite table drivers

SELECT

regexp\_extract(col\_value, '^(?:([^,]\*),?){1}', 1) driverId,

regexp\_extract(col\_value, '^(?:([^,]\*),?){2}', 1) name,

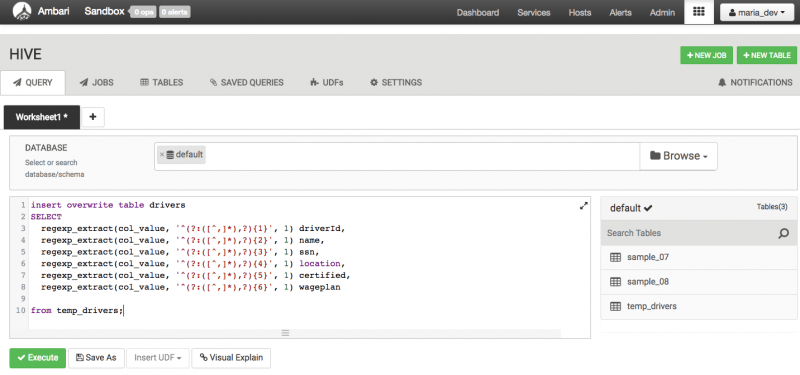
regexp\_extract(col\_value, '^(?:([^,]\*),?){3}', 1) ssn,

regexp\_extract(col\_value, '^(?:([^,]\*),?){4}', 1) location,

regexp\_extract(col\_value, '^(?:([^,]\*),?){5}', 1) certified,

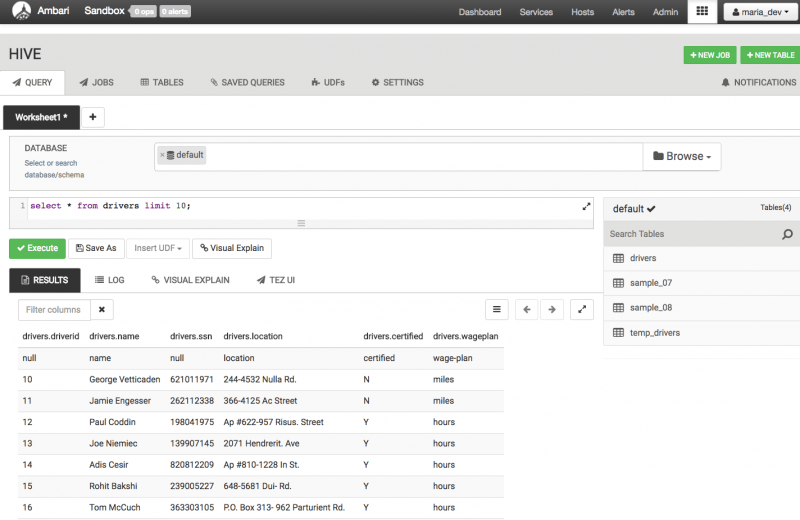
regexp\_extract(col\_value, '^(?:([^,]\*),?){6}', 1) wageplan

from temp\_drivers;



Execute the query and look at the drivers table. You should see data that looks like this.

Select \* from drivers limit 10;



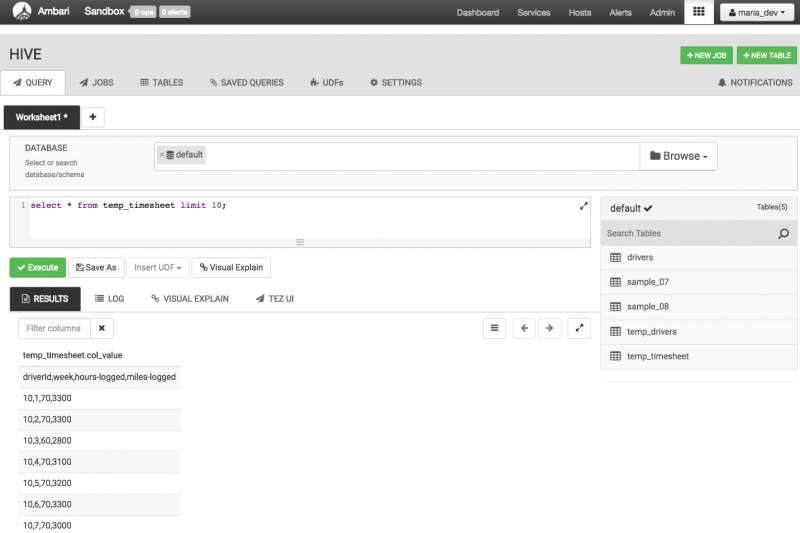
### 3.6 CREATE TEMP\_TIMESHEET AND TIMESHEET TABLES SIMILARLY

Similarly, we have to create a table called temp\_timesheet, then load the sample timesheet.csv file. Type the following queries one by one:

CREATE TABLE temp\_timesheet (col\_value string);

LOAD DATA INPATH '/user/maria\_dev/timesheet.csv' OVERWRITE INTO TABLE temp\_timesheet;

You should see the data like this:



Now create the table timesheet using the following query:

CREATE TABLE timesheet (driverId INT, week INT, hours\_logged INT , miles\_logged INT);

Insert the data into the table timesheet from temp\_timesheet table using the same regexp\_extract as we did earlier.

insert overwrite table timesheet

SELECT

regexp\_extract(col\_value, '^(?:([^,]\*),?){1}', 1) driverId,

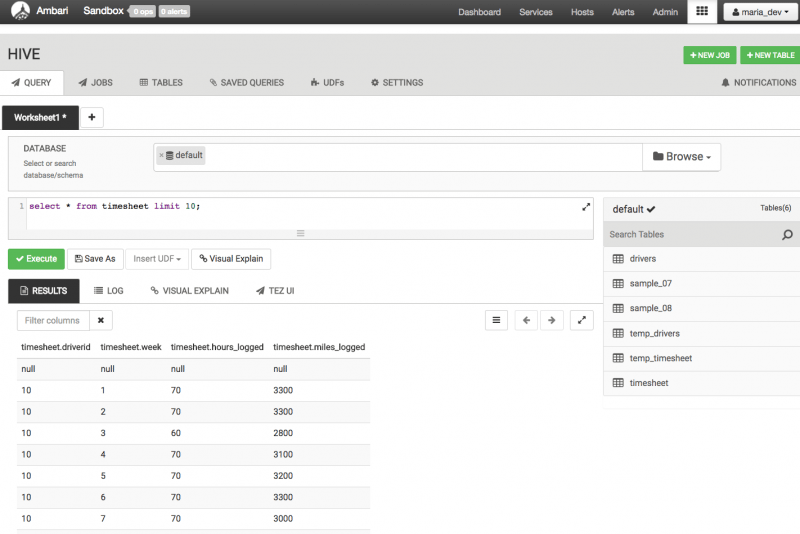
regexp\_extract(col\_value, '^(?:([^,]\*),?){2}', 1) week,

regexp\_extract(col\_value, '^(?:([^,]\*),?){3}', 1) hours\_logged,

regexp\_extract(col\_value, '^(?:([^,]\*),?){4}', 1) miles\_logged

from temp\_timesheet;

You should see the data like this:

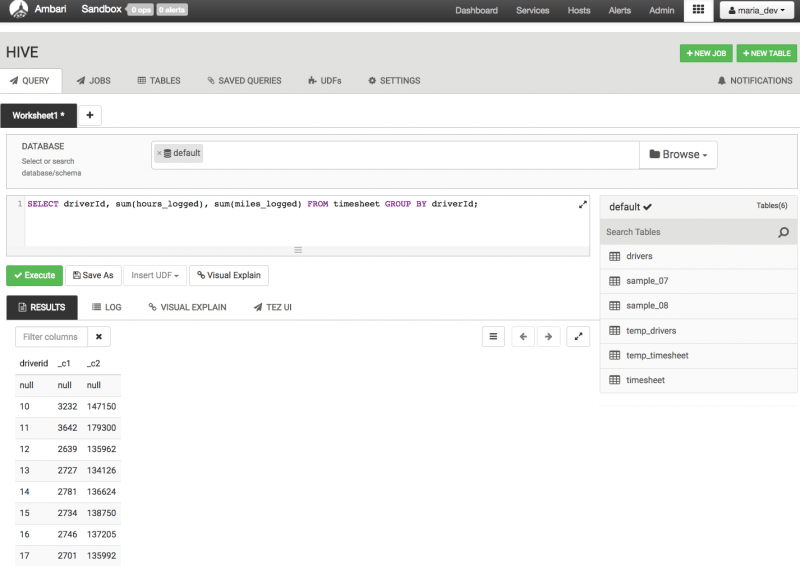


### 3.7 CREATE QUERY TO FILTER THE DATA (DRIVERID, HOURS\_LOGGED, MILES\_LOGGED)

Now we have the data fields we want. The next step is to group the data by driverId so we can find the sum of hours and miles logged score for an year. This query first groups all the records by driverId and then selects the driver with the sum of the hours and miles logged runs for that year.

SELECT driverId, sum(hours\_logged), sum(miles\_logged) FROM timesheet GROUP BY driverId;

The results of the query look like this:



### 3.8 CREATE QUERY TO JOIN THE DATA (DRIVERID, NAME, HOURS\_LOGGED, MILES\_LOGGED)

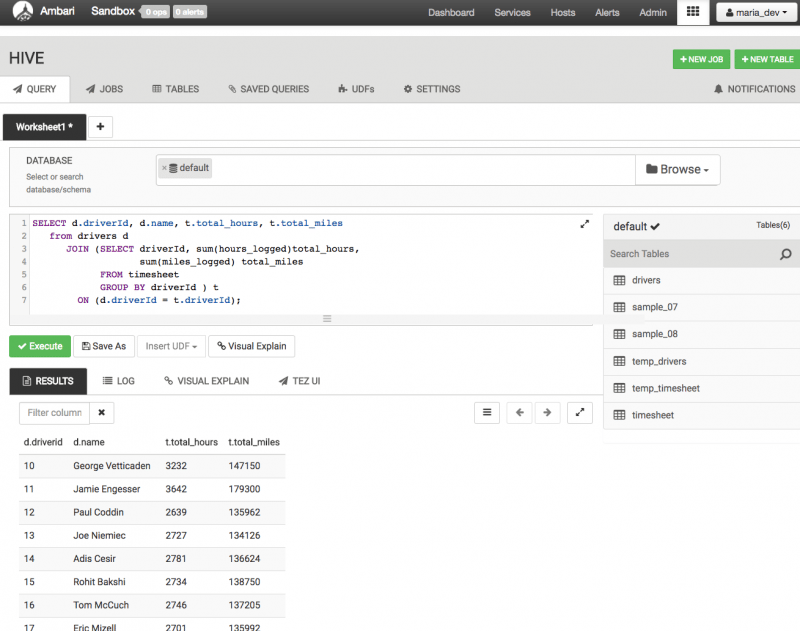
Now we need to go back and get the driverId(s) so we know who the driver(s) was. We can take the previous query and join it with the drivers records to get the final table which will have the driverId, name and the sum of hours and miles logged.

SELECT d.driverId, d.name, t.total\_hours, t.total\_miles from drivers d

JOIN (SELECT driverId, sum(hours\_logged)total\_hours, sum(miles\_logged)total\_miles FROM timesheet GROUP BY driverId ) t

ON (d.driverId = t.driverId);

The resulting data looks like:



So now we have our results. As described earlier we solved this problem using Hive step by step.

## SUMMARY

We just learned how to upload data into HDFS Files View and create hive queries to manipulate data. Let’s review all the queries that were utilized in this tutorial: **create**, **load**, **insert**, **select**, **from**, **group by**, **join** and **on**. With these queries, we created a table temp\_drivers to store the data. We created another table drivers, so we can overwrite that table with extracted data from the temp\_drivers table we created earlier. Then we did the same for temp\_timesheet and timesheet.Finally, created queries to filter the data to have the result show the sum of hours and miles logged by each driver.