OUTLINE

- Hive
- Our Data Processing Task
- Step 1: Download The Data
- Step 2: Upload The Data Files
- Step 3: Start the Hive View
- Summary
- Further Reading
-

OUR DATA PROCESSING TASK

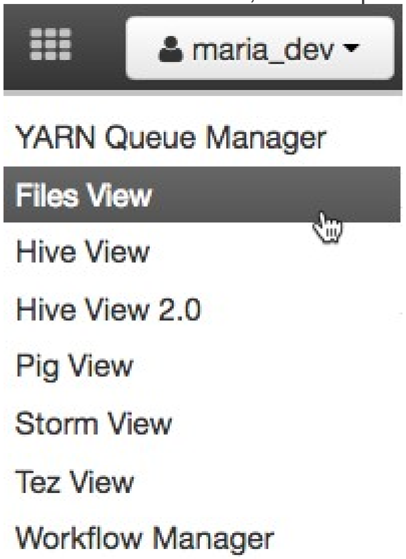
We are going to do the same data processing task as we just did with Pig in the previous tutorial. We have several files of truck driver statistics and we are going to bring them into Hive and do some simple computing with them. We are going to compute the sum of hours and miles logged driven by a truck 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

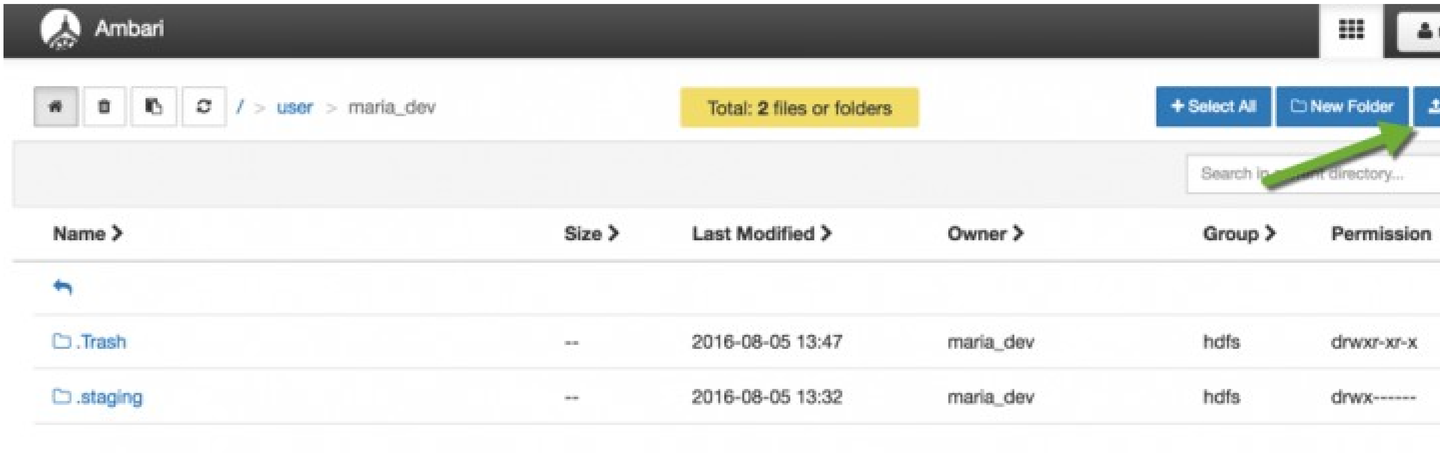
Download the driver data file from here.
Once you have the file you will need to unzip the file into a directory. We will be uploading two csv files – `drivers.csv` and `timesheet.csv`.

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.

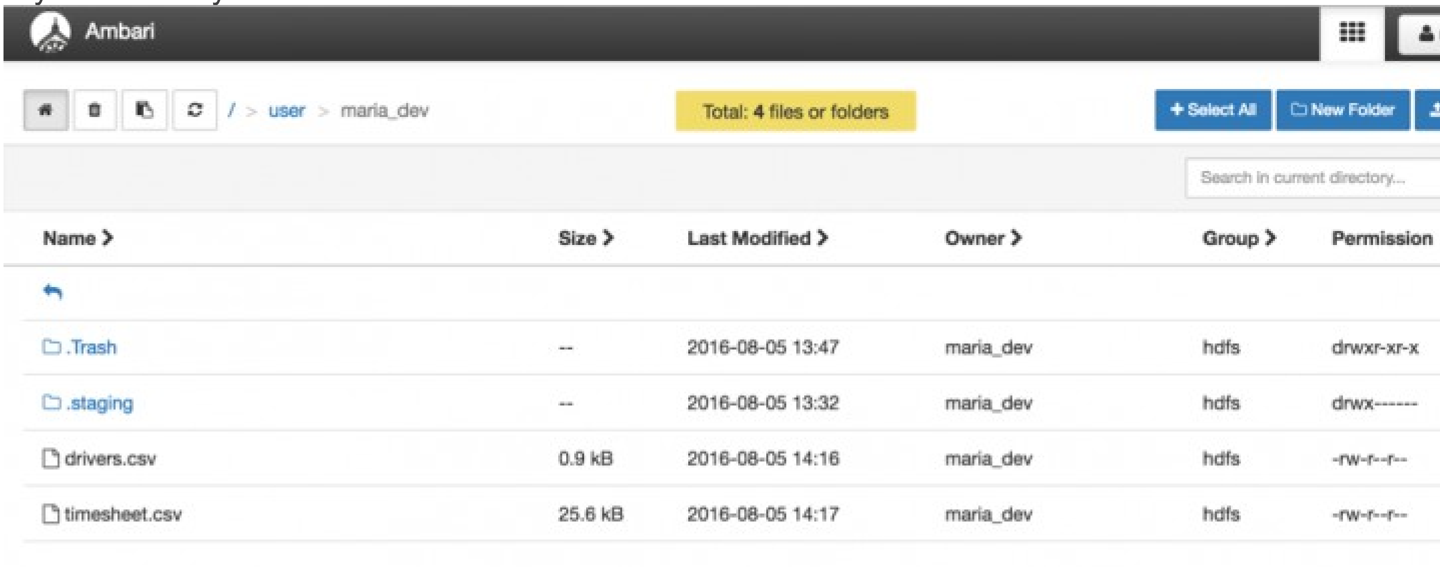


Navigate to `/user/maria_dev` and click on the `Upload` button to select the files we want to upload into the Hortonworks Sandbox environment.



Name	Size	Last Modified	Owner	Group	Permission
.Trash	--	2016-08-05 13:47	maria_dev	hdfs	drwxr-xr-x
.staging	--	2016-08-05 13:32	maria_dev	hdfs	drwx------

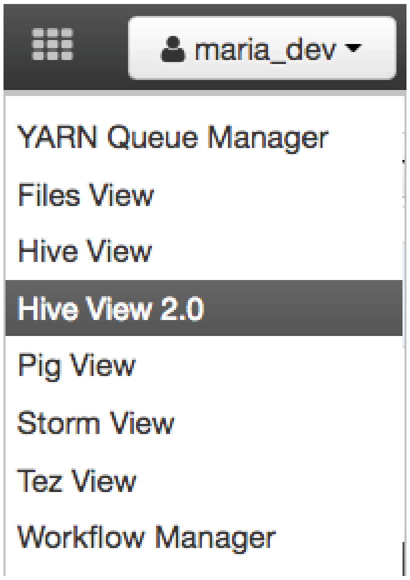
Click on the `browse` button to open a dialog box. Navigate to where you stored the `drivers.csv` file on your local disk and select `drivers.csv` and click `open`. Do the same thing for `timesheet.csv`. When you are done you will see there are two new files in your directory.



Name	Size	Last Modified	Owner	Group	Permission
.Trash	--	2016-08-05 13:47	maria_dev	hdfs	drwxr-xr-x
.staging	--	2016-08-05 13:32	maria_dev	hdfs	drwx------
drivers.csv	0.9 kB	2016-08-05 14:16	maria_dev	hdfs	-rw-r--r--
timesheet.csv	25.6 kB	2016-08-05 14:17	maria_dev	hdfs	-rw-r--r--

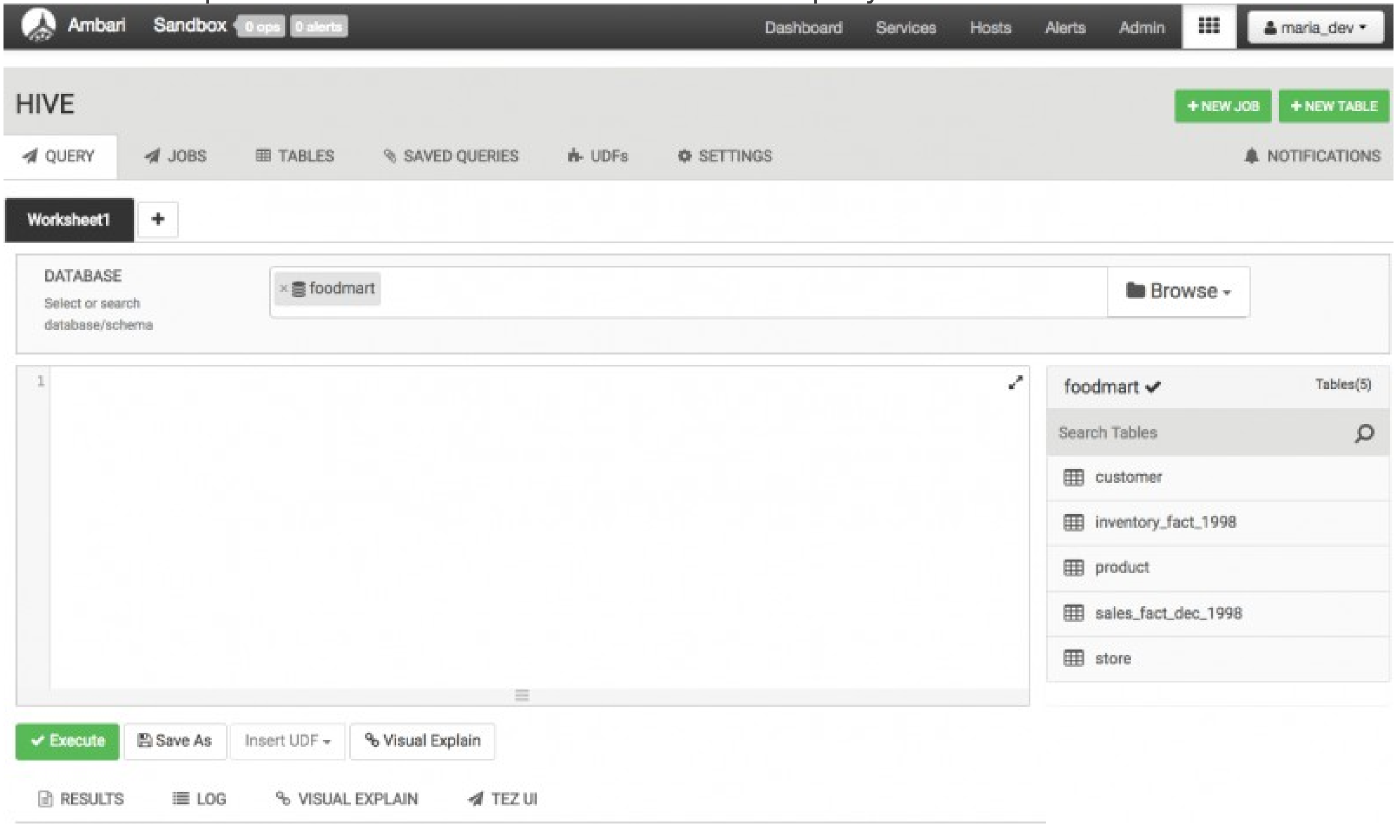
STEP 3: START THE HIVE VIEW

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.



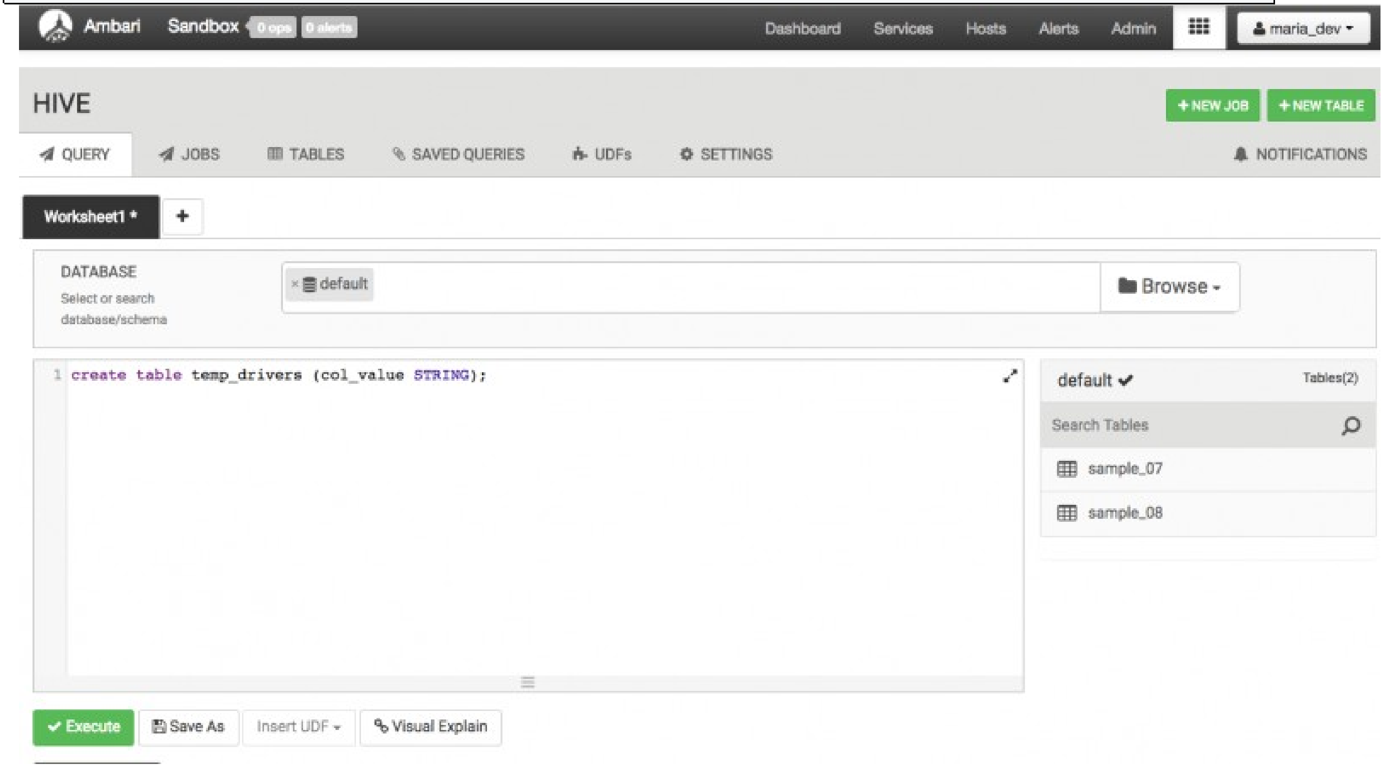
HIVE AND PIG DATA MODEL DIFFERENCES

Before we get started let’s take a look at how `Pig and Hive` data models differ. In the case of Pig all data objects exist and are operated on in the script. 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.

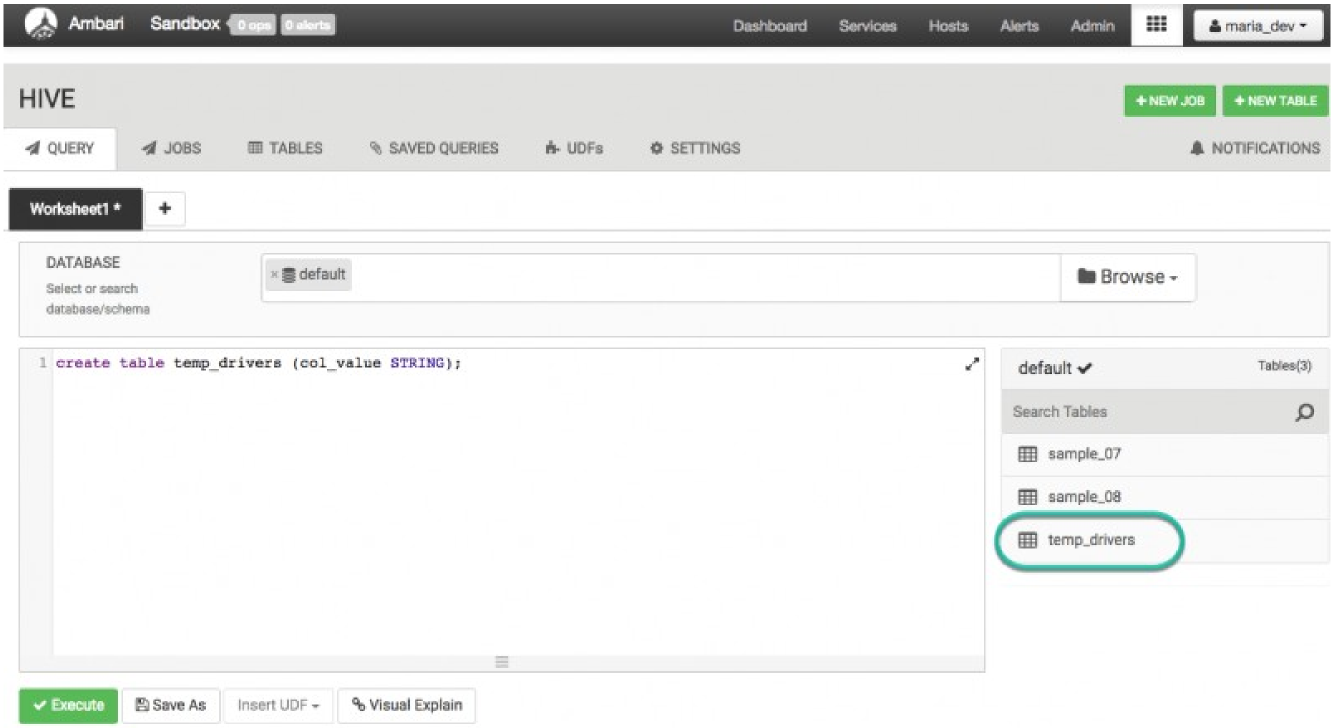
```
create table temp_drivers (col_value STRING);
```



```
create table temp_drivers (col_value STRING);
```

Hint: press `CTRL` + `Space` for autocompletion

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`.

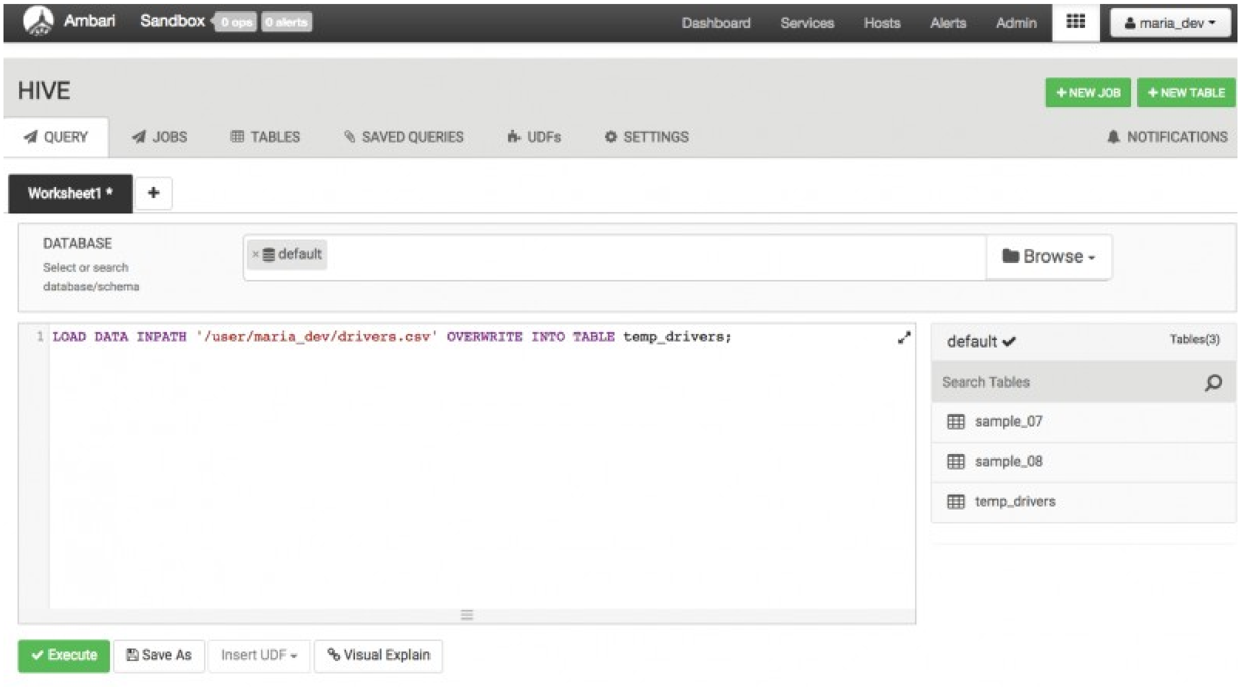


```
create table temp_drivers (col_value STRING);
```

3.3 CREATE QUERY TO POPULATE HIVE TABLE TEMP_DRIVERS WITH DRIVERS.CSV DATA

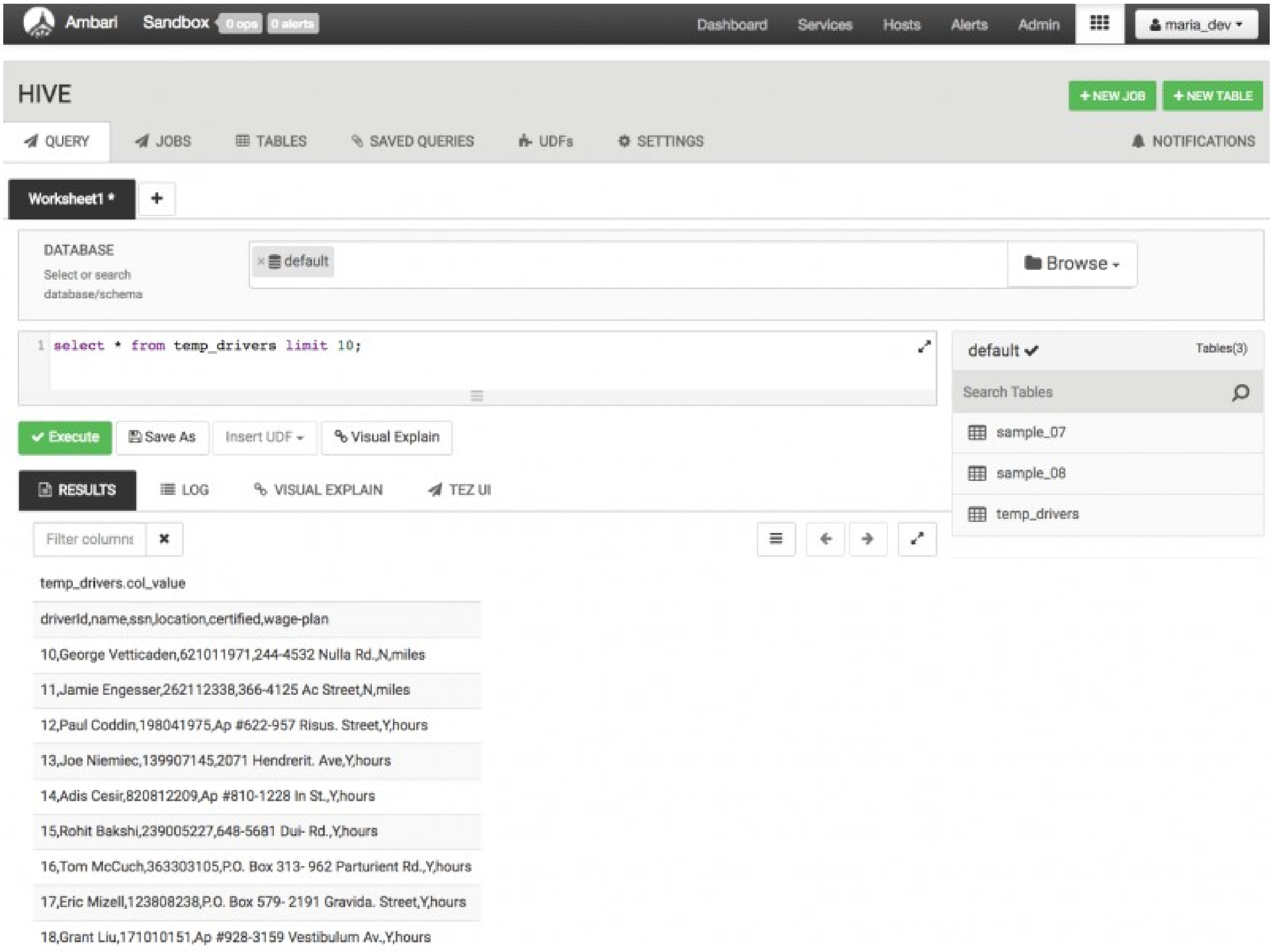
The next line of code will load the data file `drivers.csv` into the table `temp_drivers`.

```
LOAD DATA INPATH '/user/maria_dev/drivers.csv' OVERWRITE INTO TABLE
temp_drivers;
```



```
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.



DATABASE
Select or search database/schema
default

```
select * from temp_drivers limit 10;
```

default ✓

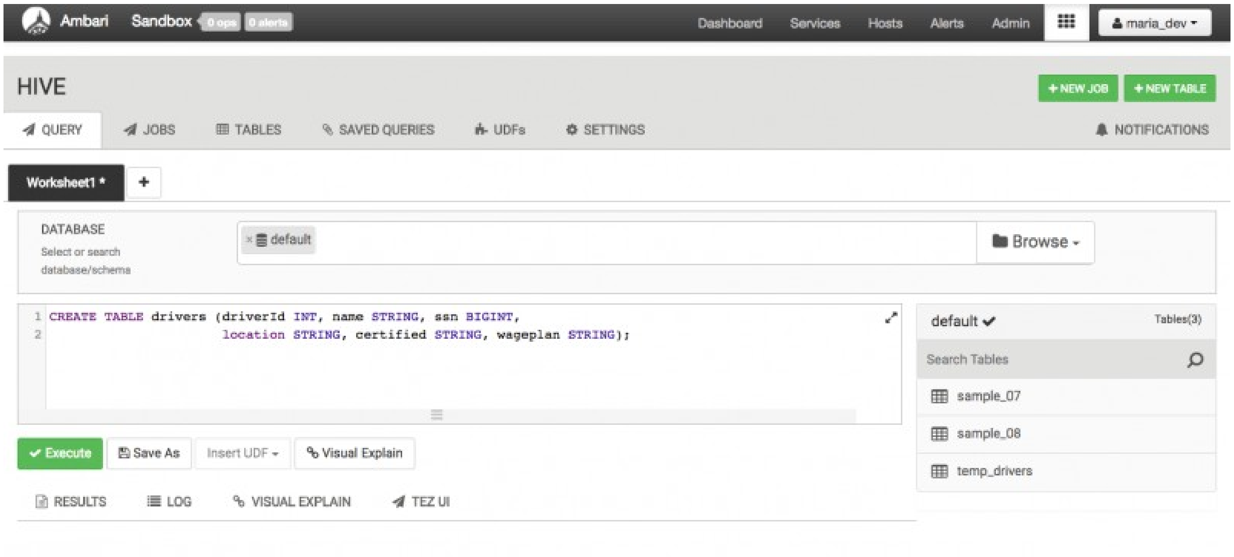
- sample_07
- sample_08
- temp_drivers

temp_drivers.col_value
driverId,name,ssn,location,certified,wage-plan
10,George Vetticaden,621011971,244-4532 Nulla Rd.,N,miles
11,Jamie Engesser,262112338,366-4125 Ac Street,N,miles
12,Paul Coddin,198041975,Ap #622-957 Risus. Street,Y,hours
13,Joe Niemiec,139907145,2071 Hendrerit. Ave,Y,hours
14,Adis Cesir,820812209,Ap #810-1228 In St.,Y,hours
15,Rohit Bakshi,239005227,648-5681 Dui- Rd.,Y,hours
16,Tom McCuch,363303105,P.O. Box 313- 962 Parturient Rd.,Y,hours
17,Eric Mizell,123808238,P.O. Box 579- 2191 Gravida. Street,Y,hours
18,Grant Liu,171010151,Ap #928-3159 Vestibulum Av.,Y,hours

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);
```

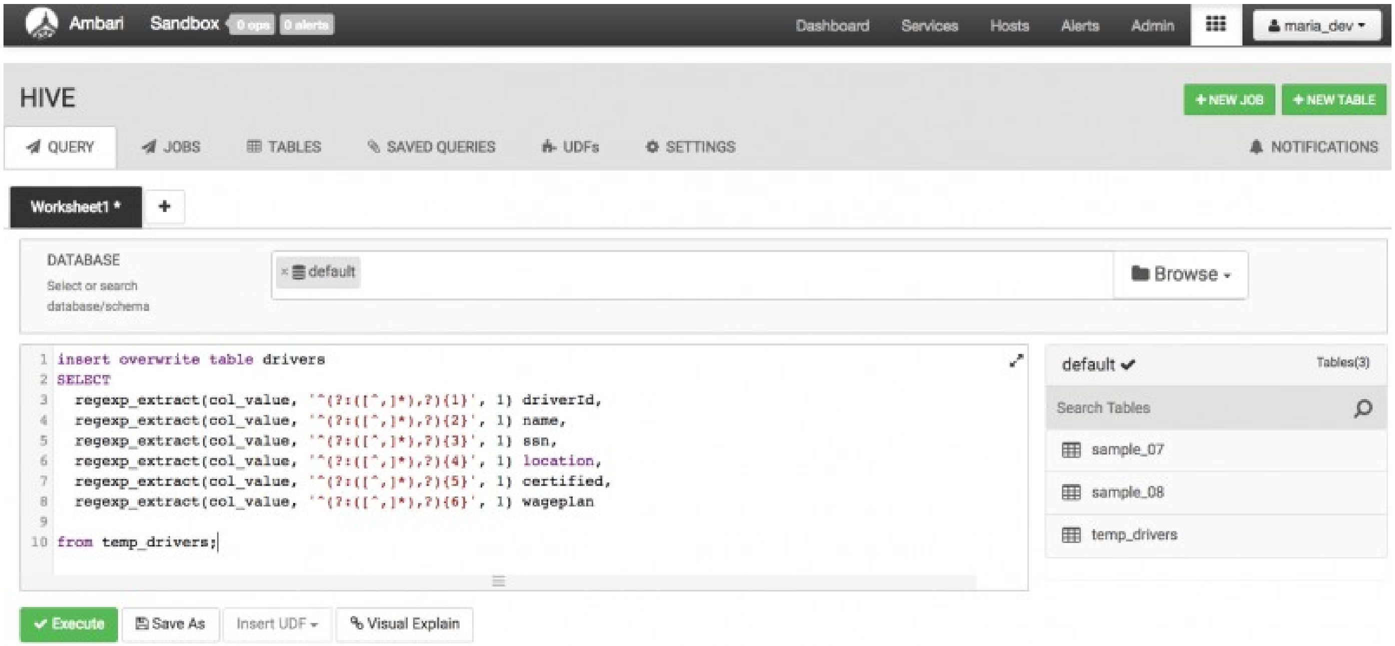


```
CREATE TABLE drivers (driverId INT, name STRING, ssn BIGINT,
                      location STRING, certified STRING, wageplan STRING);
```

3.5 CREATE QUERY TO EXTRACT DATA FROM TEMP_DRIVERS AND STORE IT TO DRIVERS

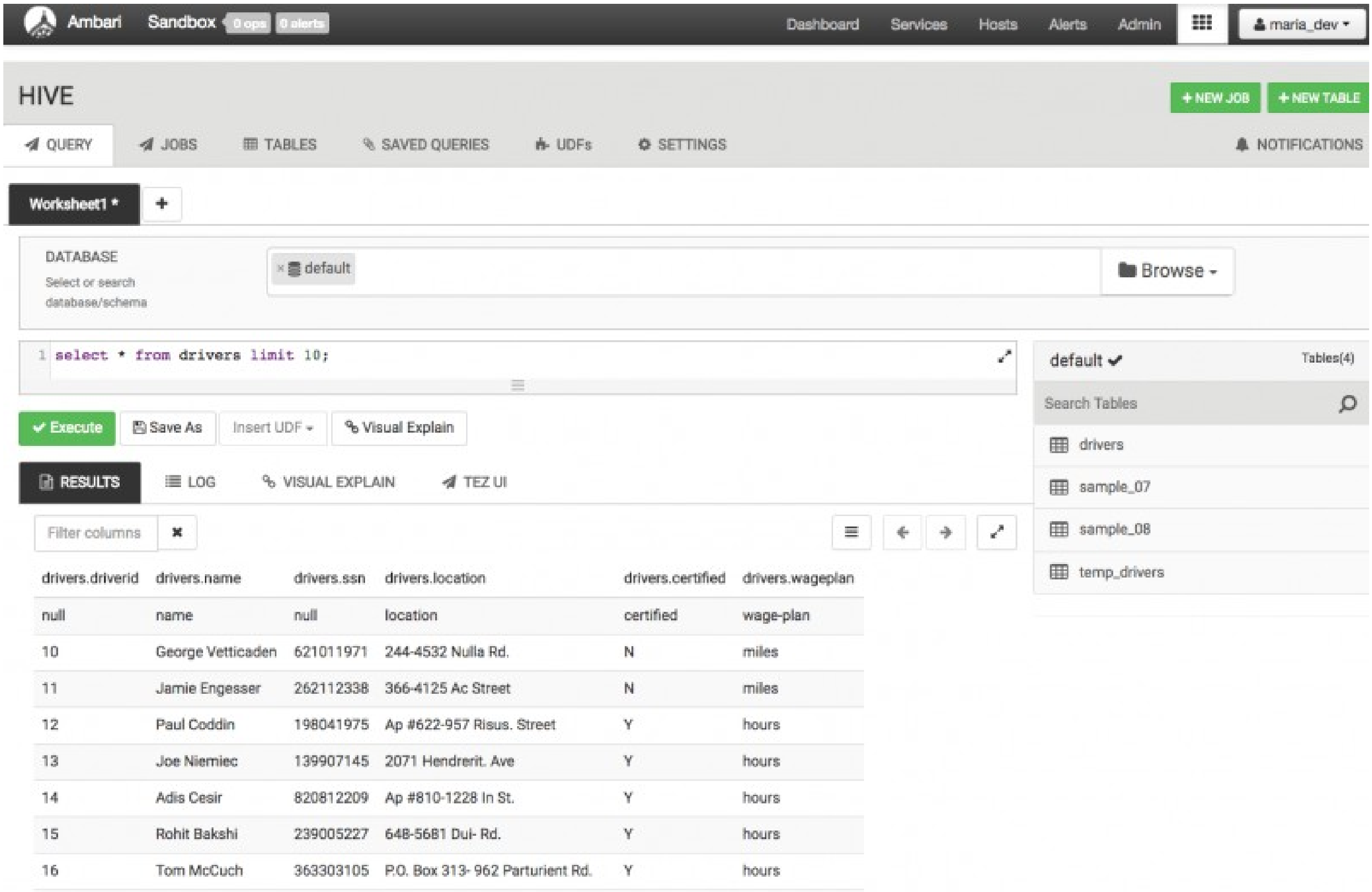
Then we extract the data we want from `temp_drivers` and copy it into `drivers`. We will do this with a `regexp` pattern. To do this we are going to build up a multi-line query. The six regexp_extract calls are going to extract the `driverId, name,ssn, location, certified and the wage-plan` fields from 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;
```



```
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;
```

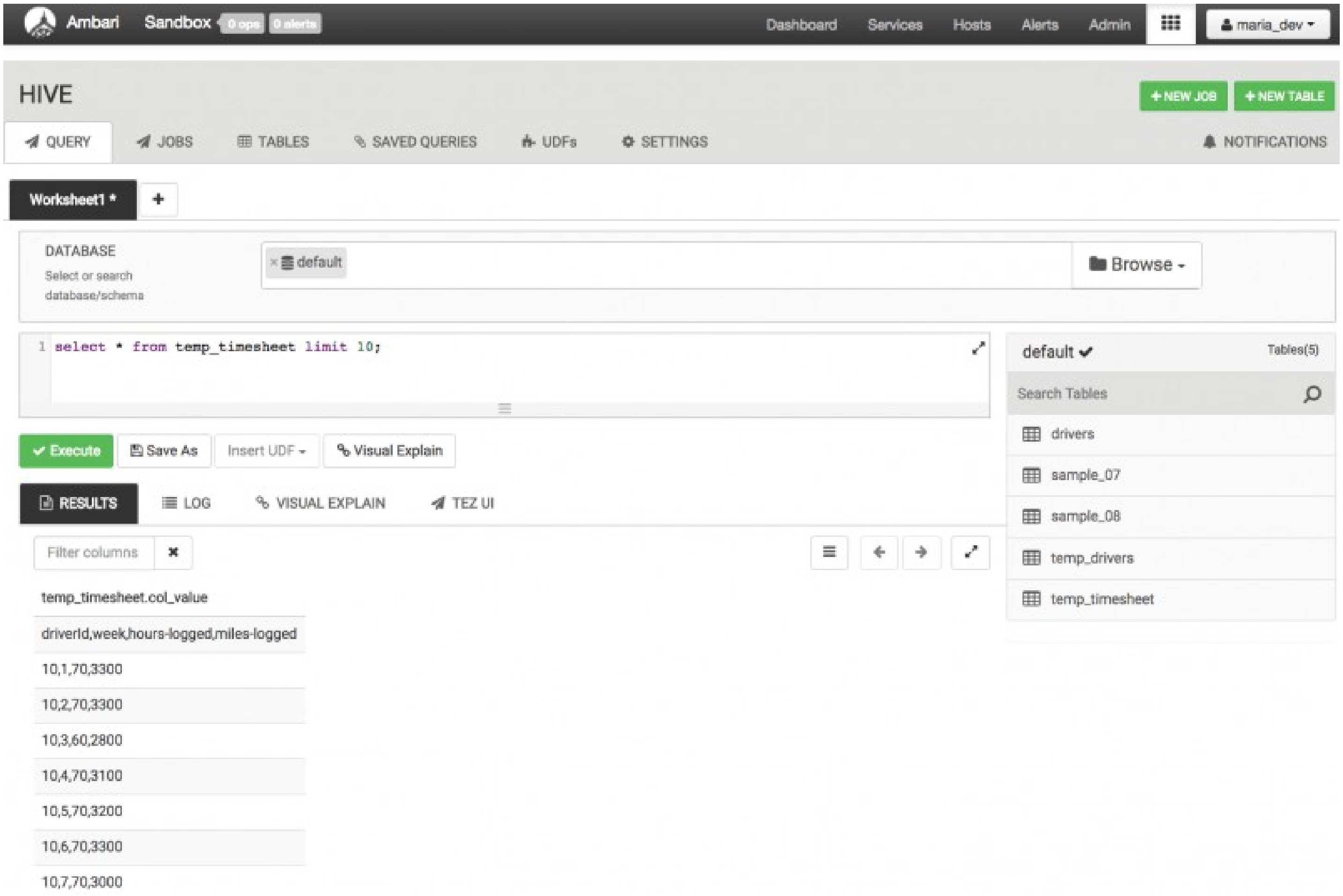
drivers.driverid	drivers.name	drivers.ssn	drivers.location	drivers.certified	drivers.wageplan
null	name	null	location	certified	wage-plan
10	George Vetticaden	621011971	244-4532 Nulla Rd.	N	miles
11	Jamie Engesser	262112338	366-4125 Ac Street	N	miles
12	Paul Coddin	198041975	Ap #622-957 Risus. Street	Y	hours
13	Joe Niemiec	139907145	2071 Hendrerit. Ave	Y	hours
14	Adis Cesir	820812209	Ap #810-1228 In St.	Y	hours
15	Rohit Bakshi	239005227	648-5681 Dui- Rd.	Y	hours
16	Tom McCuch	363303105	P.O. Box 313- 962 Parturient Rd.	Y	hours

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:



```
select * from temp_timesheet limit 10;
```

temp_timesheet.col_value
driverId,week,hours-logged,miles-logged
10,1,70,3300
10,2,70,3300
10,3,60,2800
10,4,70,3100
10,5,70,3200
10,6,70,3300
10,7,70,3000

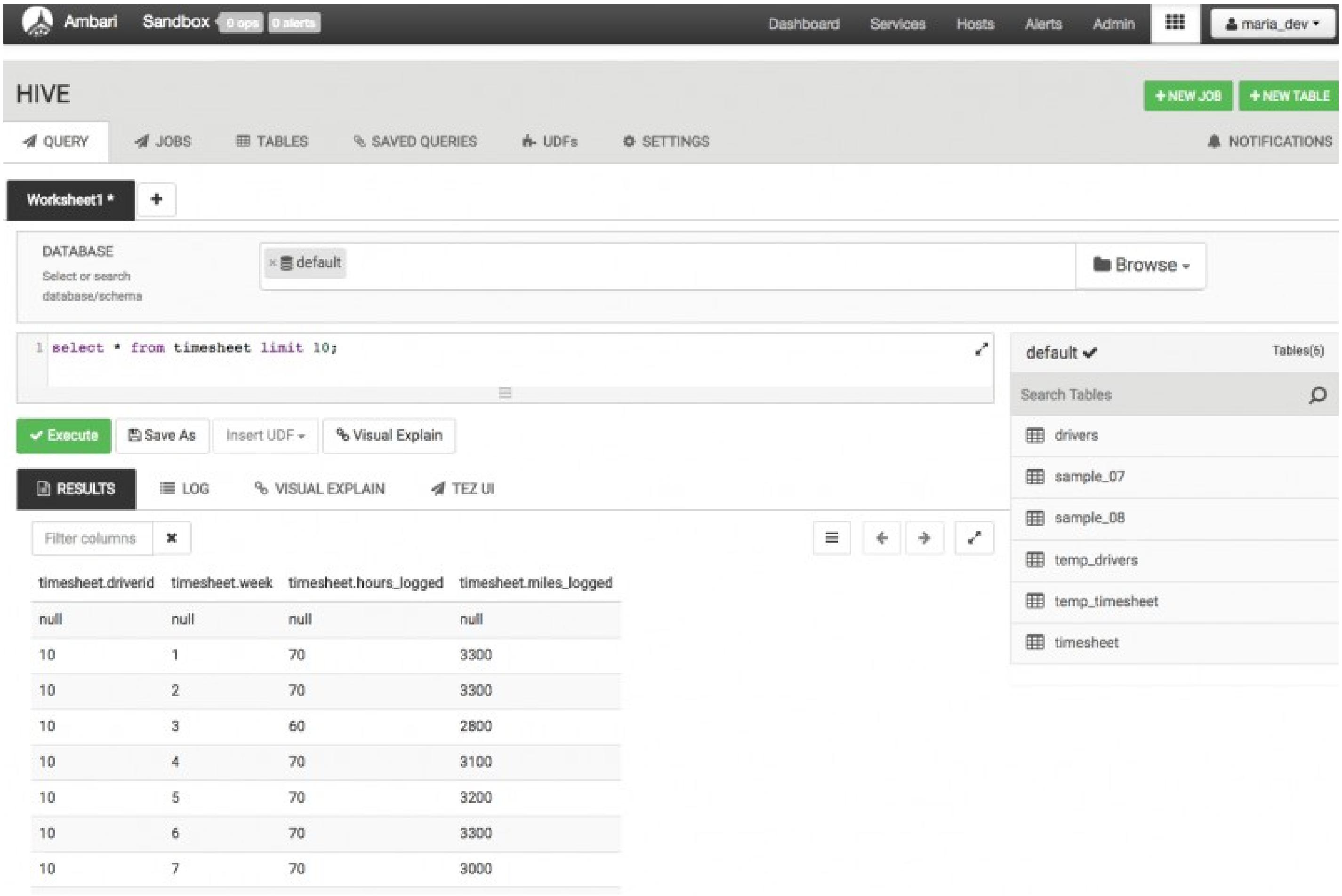
Now create the table `timesheet` using the following query:

```
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:



```
select * from timesheet limit 10;
```

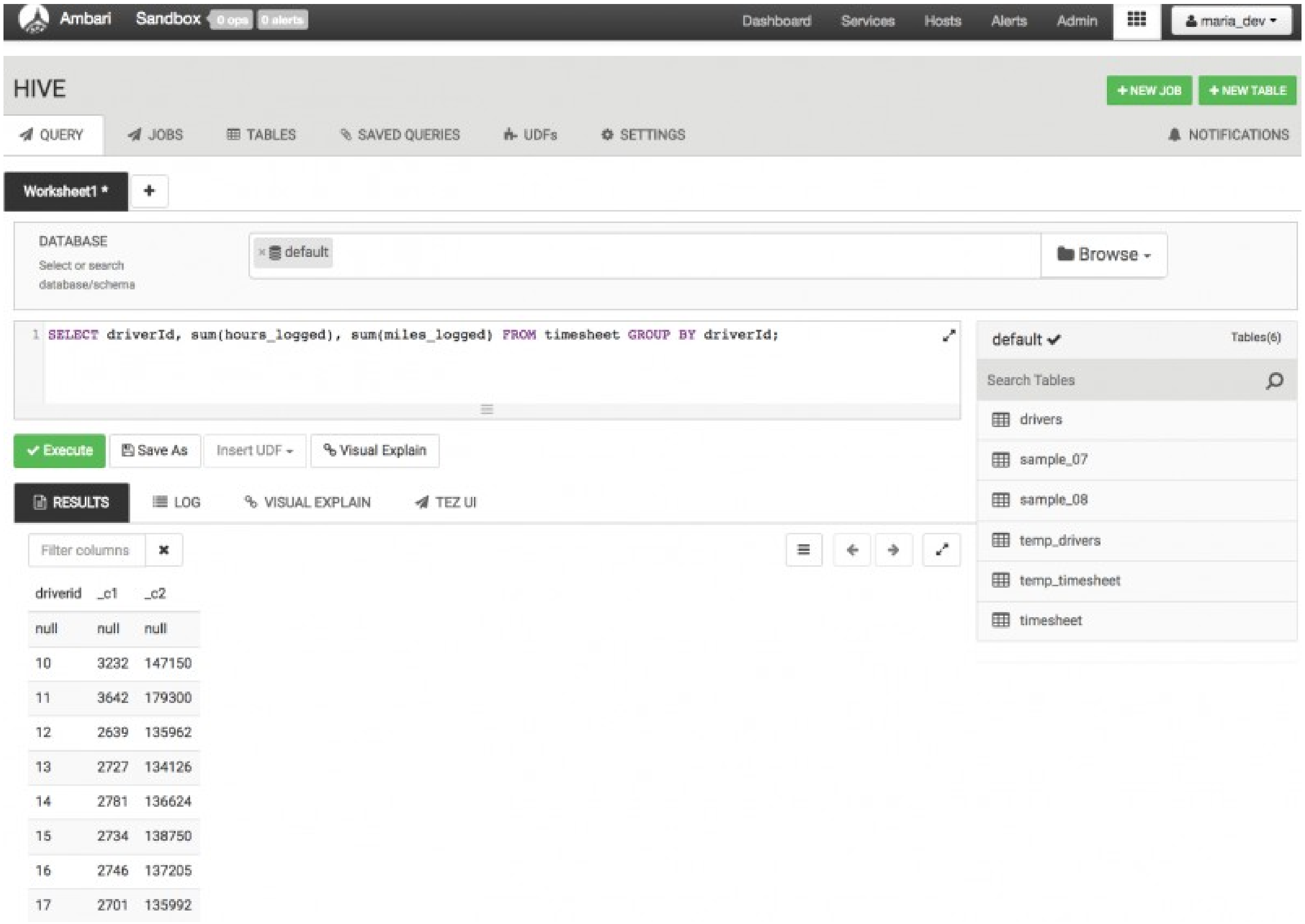
timesheet.driverid	timesheet.week	timesheet.hours_logged	timesheet.miles_logged
null	null	null	null
10	1	70	3300
10	2	70	3300
10	3	60	2800
10	4	70	3100
10	5	70	3200
10	6	70	3300
10	7	70	3000

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:



```
SELECT driverId, sum(hours_logged), sum(miles_logged) FROM timesheet GROUP BY driverId;
```

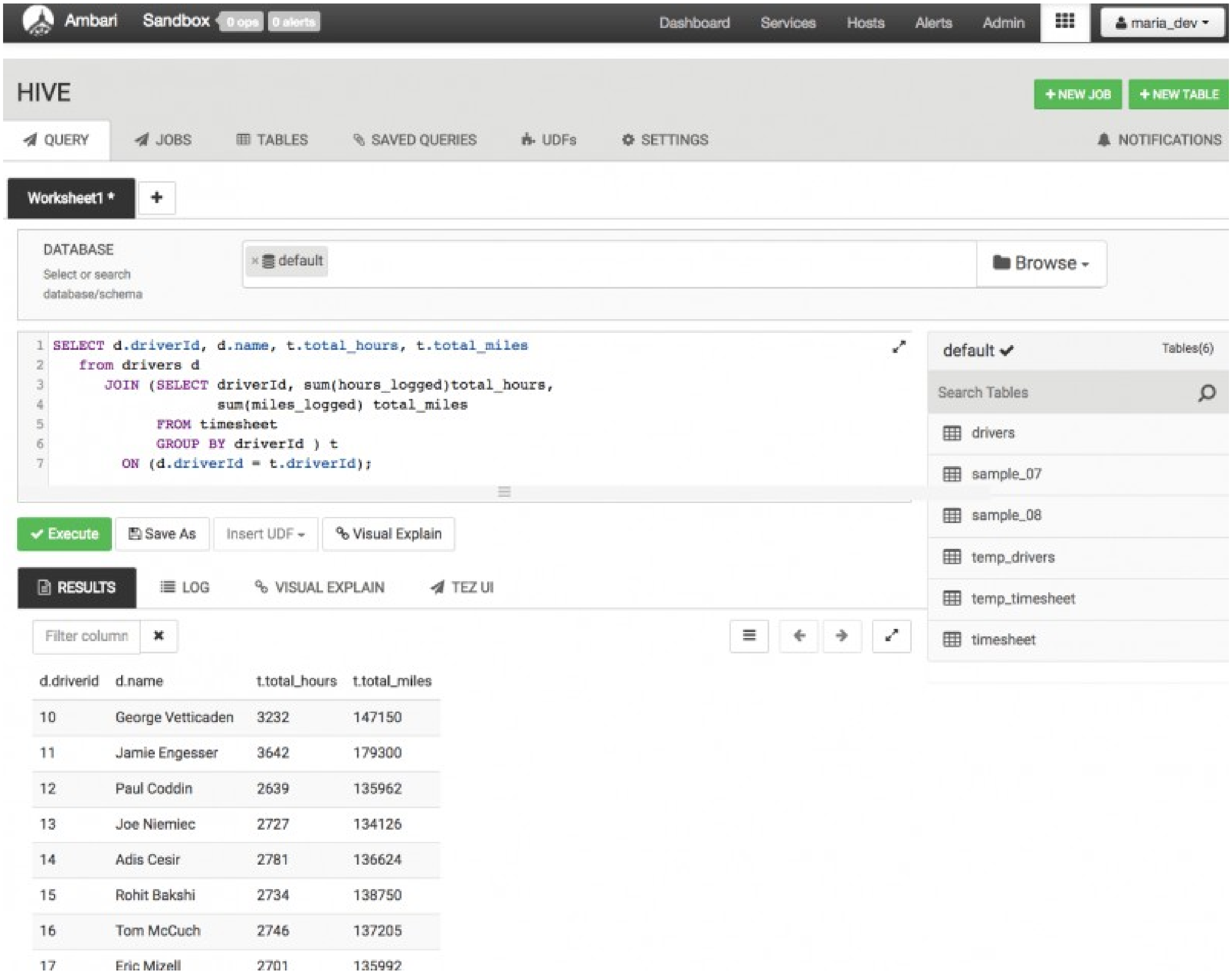
driverid	_c1	_c2
null	null	null
10	3232	147150
11	3642	179300
12	2639	135962
13	2727	134126
14	2781	136624
15	2734	138750
16	2746	137205
17	2701	135992

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:



```
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);
```

d.driverid	d.name	t.total_hours	t.total_miles
10	George Vetticaden	3232	147150
11	Jamie Engesser	3642	179300
12	Paul Coddin	2639	135962
13	Joe Niemiec	2727	134126
14	Adis Cesir	2781	136624
15	Rohit Bakshi	2734	138750
16	Tom McCuch	2746	137205
17	Eric Mizell	2701	135992

So now we have our results. As described earlier we solved this problem using Hive step by step. At any time we were free to look around at the data, decide we needed to do another task and come back. At all times the data is live and accessible to us.

SUMMARY

Congratulations on completing this tutorial! 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.

Reference:

https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/