# APACHE HIVE BEELINE CLIENT, IMPORT CSV FILE INTO HIVE



Beeline has replaced the Hive CLI in what Hive was formally called HiveServer1. Now Hive is called HiveServer2 and the new, improved CLI is Beeline.

Apache Hive says, "HiveServer2 (introduced in Hive 0.11) has its own CLI called Beeline. HiveCLI is now deprecated in favor of Beeline, as it lacks the multi-user, security, and other capabilities of HiveServer2."

Here we are going to show how to start the Hive HiverServer2 and load a CSV file into it. The syntax is much different from HiveServer1, which we wrote about here.

The data we load are weather data downloaded from here https://www.ncdc.noaa.gov/cdo-web/results. There is much data there, so in the screen shown below we picked "daily summaries" for Atlanta, Georgia. Pick just one year, 2017.





| Climate                                    | Data C | nline | Search |               |     |  |                 |
|--|--------|-------|--------|---------------|-----|--|-----------------|
| Start searching here range and select spec |        |       |        | vithin a date |     | Search Guide   |                 |
| Select Weather Observation Type/Dataset    |        |       |        |               |     | Select Type/Dataset  |                 |
| Daily Summaries •                          |        |       |        |               |     | Records of observations including details such as precipitation, wind, snowfall, and radar data. Read more about the datasets and view data samples. |                 |
| Select Date Range •                        |        |       |        |               | - 1 | Select Date Range  |                 |
| 2011-01-01 to 2017-                        | 07-28  |       |        | m             |     | Defaults to the latest available   | year for the    |
| Search For •                               |        |       |        |               |     | selected dataset or product bu<br>date range within the available  |                 |
| Cities                                     |        |       |        |               | 7   | Search For   |                 |
| Enter a Search Term                        | •      |       |        |               |     | Stations: Enter name, WBAN, C<br>NWSLI or COOP identifiers.  | HCND, FAA, ICAO |
| Atlanta                                    |        |       |        |               |     | Locations: Enter name of city, of country or other geographic lo   |                 |

The data looks

like this after the header has been removed. Use a text editor to do that,

GHCND:US1GADK0001,TUCKER 1.3 ENE GA US,20170101,0.30,-9999,-9999 GHCND:US1GADK0001,TUCKER 1.3 ENE GA US,20170102,0.76,-9999,-9999

Now install Hive using the instructions from the <u>Apache web site</u>. And then set \$HIVE\_HOME. Start Hadoop using start\_dfs.sh for a local installation or **start\_yarn.sh** and then start\_dfs.sh for a cluster installation.

### **How Beeline differs from Hive CLI**

One big difference you will notice from HiveServer1 to HiveServer2 is you do not need have to manually create schemas in MySQL or Derby. Instead, with HiveServer2, install MySQL then run this command to create the schema:

\$HIVE\_HOME/bin/schematool -dbType mysql -initSchema

### **Start Hive Server**

Then start the Hive server:

sudo nohup \$HIVE\_HOME/bin/hiveserver2&

### **Beeline Hive Shell**

Here is how to open the Beeline Hive shell>

\$HIVE HOME/bin/beeline -u jdbc:hive2://localhost:10000

Now, the user root will get this error when they run commands:

User: root is not allowed to impersonate anonymous

So when you set up the config files give root permission to do that, if you want to login as user root.

Notice below that we specifically give the root user the rights to execute Hive commands. You can switch the name **root** to your userid or read the instructions to get more details on the other options.

```
cat $HIVE HOME/conf/core-site.xml
<configuration>
cproperty>
<name>hive.server2.enable.doAs
<value>true</value>
cproperty>
<name>hadoop.proxyuser.root.groups</name>
<value>*</value>
cproperty>
<name>hadoop.proxyuser.root.hosts
<value>*</value>
</configuration>
This is the JDBC URL. Because this is a standard it means you can connect to Hive from Java and
SQL clients, like Squirrel.
cat $HIVE HOME/conf/hive-site.xml
<configuration>
cproperty>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true/value>
<description>metadata is stored in a MySQL server</description>
cproperty>
<name>javax.jdo.option.ConnectionDriverName
<value>com.mysql.jdbc.Driver</value>
<description>MySQL JDBC driver class</description>
cproperty>
<name>javax.jdo.option.ConnectionUserName
<value>hiveuser
<description>user name for connecting to mysql server</description>
cproperty>
<name>javax.jdo.option.ConnectionPassword</name>
<value>password</value>
```

<description>password for connecting to mysql server</description>

</property>

```
</configuration>
Now create the weather table:
CREATE table weather(
STATION STRING,
STATION NAME STRING,
WDATE STRING,
PRCP FLOAT,
WIND INT,
SNOW INT
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
Then load the data from the local file system.
LOAD DATA INPATH '/root/Documents/1031764.csv' OVERWRITE INTO TABLE weather;
Now show that there is some data in it.
select count(*) from weather;
_C0
4235
1 row selected (1.914
seconds)
```

## **Using SQL With Hive**

The SQL language Reference manual for Hive is <u>here</u>. It will be useful to follow along.

Let's see what is the rainiest day on the month for any month of the year.

First lets show the schema of our table:

col\_name data\_type comment

describe weather:

# station string station\_name string wdate string prcp float wind int

int

Now, we cannot group on columns on which we have run functions. And we cannot create views. That means we cannot divide this into simpler, concrete steps. All we can do is make nested SQL statements. In this case we do three:

- 1. Turn the date field into day of month.
- 2. Using that query total rainfall per day of month.
- 3. Finally print the results sorted by amount of rain per day of month.

Here is the complete query.

```
Select b.rain, b.day from
(select sum(prcp) as rain, a.day as day from
(select prcp, dayofmonth(to_date(from_unixtime(unix_timestamp(wdate,
"yyyymmddd")))) as day from weather where prcp > 0) a
Group by a.day) b
Order by b.rain;
```

Looking at the innermost query first, turn unix time into a date object using to\_date(). Then use dayofmonth() to get day or month.

```
select prcp,
dayofmonth(to_date(from_unixtime(unix_timestamp(wdate, "yyyymmddd")))) as day
from weather where prcp > 0
```

That result lacks the sum of precipitation. After all we cannot sum the precipitation by day of month until we have grouped the data into day of month. And remember we said we cannot order by calculated values. We so we need another step for that.

This next SQL shows the rainfall by day. So we need one step more after this to order it by total rainfall and not day.

```
select sum(prcp) as rain, a.day as day from
(select prcp, dayofmonth(to_date(from_unixtime(unix_timestamp(wdate,
"yyyymmddd")))) as day from weather where prcp > 0) a
Group by a.day
```

Finally, running the first query about Looks like the 6th day of the month is the rainiest.

| b.rain             | b.day |
|--------------------|-------|
| 0.9000000040978193 | 27    |
| 2.240000018849969  | 17    |
| 3.7899999916553497 | 29    |
| 5.250000001862645  | 9     |
| 5.830000016838312  | 13    |
| 5.970000043511391  | 11    |
| 7.929999986663461  | 25    |
| 9.790000105276704  | 16    |
| 10.079999938607216 | 10    |
| 10.439999951049685 | 14    |
| 10.459999982267618 | 18    |
| 11.820000018924475 | 26    |
|                    |       |

| b.rain             | b.day |
|--------------------|-------|
| 14.159999972209334 | 28    |
| 14.389999927952886 | 12    |
| 15.17000013589859  | 19    |
| 18.720000019297004 | 15    |
| 20.27999988757074  | 1     |
| 20.329999884590507 | 4     |
| 21.089999973773956 | 31    |
| 21.169999765232205 | 7     |
| 26.680000007152557 | 20    |
| 35.399999901652336 | 30    |
| 35.699999986216426 | 8     |
| 36.230000007897615 | 2     |
| 37.28000004403293  | 24    |
| 44.0399998370558   | 23    |
| 47.20999996364117  | 3     |
| 63.980000307783484 | 5     |
| 77.87000086344779  | 21    |
| 82.32000004313886  | 22    |
| 82.78000020980835  | 6     |