

Querying Data

Date published: 2020-08-17

Date modified: 2023-11-02



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Query data.....	4
Submit queries with Hue.....	4
View query history.....	6
Create User-defined functions.....	6
Set up dev environment.....	6
Create UDF.....	8
Build project and upload JAR.....	9
Register UDF.....	9
Call UDF in a query.....	10

Querying data in Cloudera Data Warehouse

This topic describes how to query data in your Virtual Warehouse on Cloudera Data Warehouse (CDW).

About this task

The CDW service includes the Hue SQL editor that you can use to submit queries to Virtual Warehouses. For example, you can use Hue to submit queries to an Impala Virtual Warehouse.

Procedure

1. Log in to the Data Warehouse service as DWUser
The **Overview** page is displayed.
2. Click Hue on the Virtual Warehouse tile.
3. Enter your query into the editor and submit it to the Virtual Warehouse.

Submitting queries with Hue

You can write and edit queries for Hive or Impala Virtual Warehouses in the Cloudera Data Warehouse (CDW) service by using Hue.

About this task

For detailed information about using Hue, see [Using Hue](#).

Required role: DWUser

Before you begin

Hue uses your LDAP credentials that you have configured for the CDP cluster.

Procedure

1. Log into the CDP web interface and navigate to the Data Warehouse service.
2. In the Data Warehouse service, navigate to the **Overview** page.

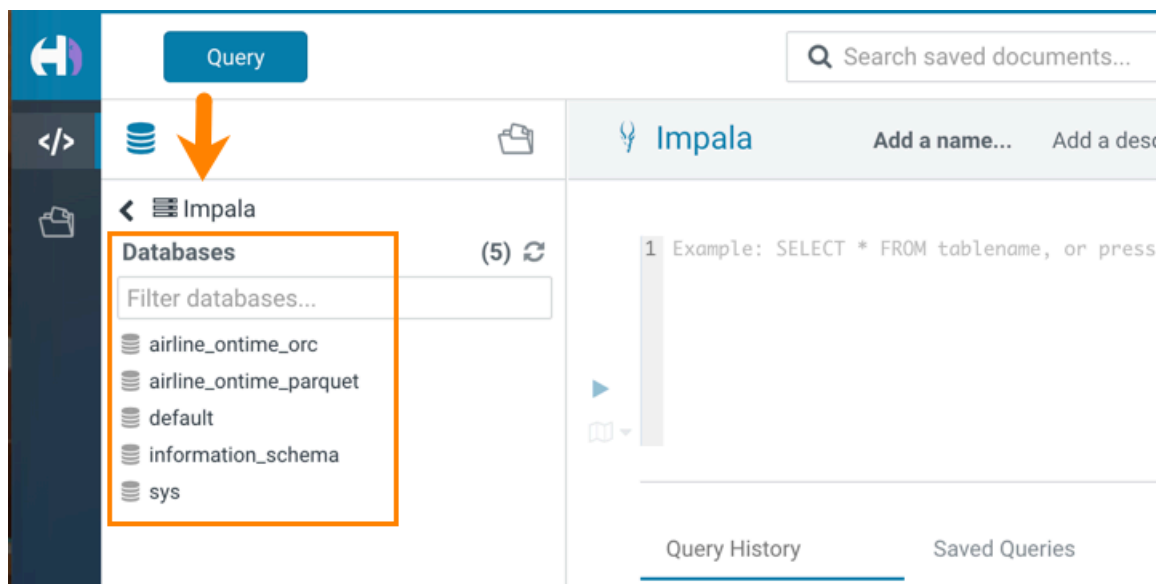


Note: You can also launch Hue from the **Virtual Warehouse** page using the same steps.

3. To run Impala queries:


- a) Click HUE on the Impala Virtual Warehouse tile.


The query editor is displayed:




- b) Click a database to view the tables it contains.

When you click a database, it sets it as the target of your query in the main query editor panel.

- c) Type a query in the editor panel and click the run icon  to run the query.

You can also run multiple queries by selecting them and clicking .

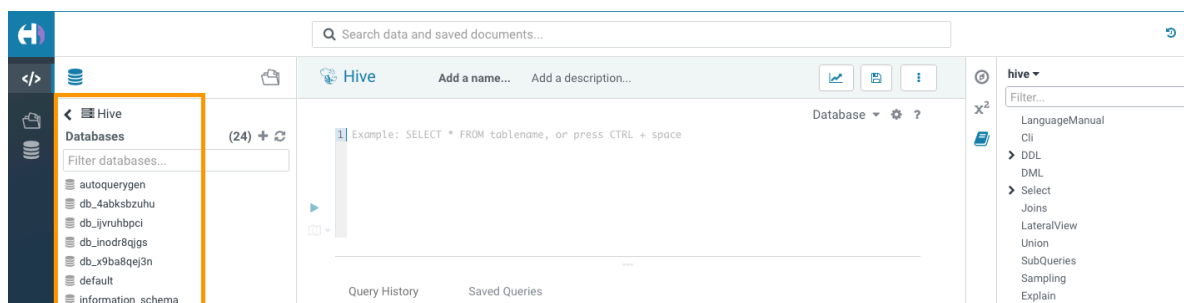


Note: Use the Impala language reference to get information about syntax in addition to the SQL auto-complete feature that is built in. To view the language reference, click the book icon  to the right of the query editor panel.

4. To run Hive queries:


- a) Click HUE on the Hive Virtual Warehouse tile.


The Hive query editor is displayed:




- b) Click a database to view the tables it contains.

When you click a database, it sets it as the target of your query in the main query editor panel.

- c) Type a query in the editor panel and click the run icon  to run the query.

You can also run multiple queries by selecting them and clicking .



Note: Use the Hive language reference to get information about syntax in addition to the SQL auto-complete feature that is built in. To view the language reference, click the book icon  to the right of the query editor panel.

Related Information

[Advanced Hue configurations \(safety valves\) in CDW](#)

Viewing query history in Cloudera Data Warehouse

In Cloudera Data Warehouse (CDW), you can view all queries that were run against a Database Catalog from Hue, Beeline, Hive Warehouse Connector (HWC), Tableau, Impala-shell, Impyla, and so on.

About this task

You need to set up Query Processor administrators to view the list of all queries from all users, or to restrict viewing of queries.

Procedure

1. Log in to the CDP web interface and navigate to the Data Warehouse service.
2. In the Data Warehouse service, navigate to the **Overview** page.
3. From a Virtual Warehouse, launch Hue.
4. Click on the Jobs icon on the left-assist panel.
The **Job Browser** page is displayed.
5. Go to the **Queries** tab to view query history and query details.

Related Information

[Viewing Hive query details](#)

[Viewing Impala query details](#)

[Adding Query Processor Administrator users and groups in Cloudera Data Warehouse](#)

Creating a user-defined function in Cloudera Data Warehouse Private Cloud

You export user-defined functionality (UDF) to a JAR from a Hadoop- and Hive-compatible Java project and store the JAR on your cluster. Using Hive commands, you register the UDF based on the JAR, and call the UDF from a Hive query.

Before you begin

- You must have access rights to upload the JAR to your cluster. Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator).
- Make sure Hive on Tez or Hive LLAP is running on the cluster.
- Make sure that you have installed Java and a Java integrated development environment (IDE) tool on the machine, or virtual machine, where you want to create the UDF.

Setting up the development environment

You can create a Hive UDF in a development environment using IntelliJ, for example, and build the UDF. You define the Cloudera Maven Repository in your POM, which accesses necessary JARS `hadoop-common-<version>.jar` and `hive-exec-<version>.jar`.

Procedure

1. Open IntelliJ and create a new Maven-based project. Click Create New Project. Select Maven and the supported Java version as the Project SDK. Click Next.
2. Add archetype information.
For example:
 - GroupId: com.mycompany.hiveudf
 - ArtifactId: hiveudf
3. Click Next and Finish.
The generated pom.xml appears in sample-hiveudf.
4. To the pom.xml, add properties to facilitate versioning.
For example:

```
<properties>
  <hadoop.version>TBD</hadoop.version>
  <hive.version>TBD</hive.version>
</properties>
```

5. In the pom.xml, define the repositories.
Use internal repositories if you do not have internet access.

```
<repositories>
  <repository>
    <releases>
      <enabled>true</enabled>
      <updatePolicy>always</updatePolicy>
      <checksumPolicy>warn</checksumPolicy>
    </releases>
    <snapshots>
      <enabled>false</enabled>
      <updatePolicy>never</updatePolicy>
      <checksumPolicy>fail</checksumPolicy>
    </snapshots>
    <id>HDPReleases</id>
    <name>HDP Releases</name>
    <url>http://repo.hortonworks.com/content/repositories/releases/</u
rl>
    <layout>default</layout>
  </repository>
  <repository>
    <id>public.repo.hortonworks.com</id>
    <name>Public Hortonworks Maven Repo</name>
    <url>http://repo.hortonworks.com/content/groups/public/</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
  <repository>
    <id>repository.cloudera.com</id>
    <url>https://repository.cloudera.com/artifactory/cloudera-repos/<
/url>
  </repository>
</repositories>
```

6. Define dependencies.
For example:

```
<dependencies>
  <dependency>
    <groupId>org.apache.hive</groupId>
```

```

    <artifactId>hive-exec</artifactId>
    <version>${hive.version}</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>${hadoop.version}</version>
  </dependency>
</dependencies>

```

Creating the UDF class

You define the UDF logic in a new class that returns the data type of a selected column in a table.

Procedure

1. In IntelliJ, click the vertical project tab, and expand hiveudf: hiveudf src main . Select the java directory, and on the context menu, select **New Java Class** , and name the class, for example, **TypeOf**.
2. Extend the **GenericUDF** class to include the logic that identifies the data type of a column.

For example:

```

package com.mycompany.hiveudf;

import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.primitive.\
PrimitiveObjectInspectorFactory;
import org.apache.hadoop.io.Text;
public class TypeOf extends GenericUDF {
    private final Text output = new Text();
    @Override
    public ObjectInspector initialize(ObjectInspector[] arguments) throws U
DFArgumentException {
        checkArgsSize(arguments, 1, 1);
        checkArgPrimitive(arguments, 0);
        ObjectInspector outputOI = PrimitiveObjectInspectorFactory.writableSt
ringObjectInspector;
        return outputOI;
    }

    @Override
    public Object evaluate(DeferredObject[] arguments) throws HiveException
    {
        Object obj;
        if ((obj = arguments[0].get()) == null) {
            String res = "Type: NULL";
            output.set(res);
        } else {
            String res = "Type: " + obj.getClass().getName();
            output.set(res);
        }
        return output;
    }

    @Override
    public String getDisplayString(String[] children) {
        return getStandardDisplayString("TYPEOF", children, ",");
    }
}

```


}

Building the project and uploading the JAR

You compile the UDF code into a JAR and add the JAR to the classpath on the cluster.

About this task

Use the direct reference method to configure the cluster to find the JAR. It is a straight-forward method, but recommended for development only.

Procedure

1. Build the IntelliJ project.

```
...
[INFO] Building jar: /Users/max/IdeaProjects/hiveudf/target/TypeOf-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 14.820 s
[INFO] Finished at: 2019-04-03T16:53:04-07:00
[INFO] Final Memory: 26M/397M
[INFO] -----
```

Process finished with exit code 0

2. In IntelliJ, navigate to the JAR in the /target directory of the project.
3. Configure the cluster so that Hive can find the JAR using the direct reference method.
 - a) Upload the JAR to HDFS.
 - b) Move the JAR into the Hive warehouse. For example, in CDP Data Center:

```
$ hdfs dfs -put TypeOf-1.0-SNAPSHOT.jar /warehouse/tablespace/managed/hiveudf-1.0-SNAPSHOT.jar
```

4. In IntelliJ, click Save.
5. Click Actions Deploy Client Configuration .
6. Restart the Hive service.

Registering the UDF

In Cloudera Data Warehouse (CDW), you run a command from Hue to make the UDF functional in Hive queries. The UDF persists between HiveServer restarts.

Before you begin

You need to set up UDF access using a Ranger policy as follows:

1. Log in to the Data Warehouse service and open Ranger from the Database Catalog associated with your Hive Virtual Warehouse.
2. On the **Service Manager** page, under the HADOOP SQL section, select the Database Catalog associated with the Hive Virtual Warehouse in which you want to run the UDFs.

The list of policies is displayed.

3. Select the all - database, udf policy and add the users needing access to Hue. To add all users, you can specify {USER}.

About this task

In this task, the registration command differs depending on the method you choose to configure the cluster for finding the JAR. If you use the Hive aux library directory method that involves a symbolic link, you need to restart the HiveServer pod after registration. If you use the direct JAR reference method, you do not need to restart HiveServer. You must recreate the symbolic link after any patch or maintenance upgrades that deploy a new version of Hive.

Procedure

1. Open Hue from the Hive Virtual Warehouse in CDW.
2. Run the registration command by including the JAR location in the command as follows:

```
CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.TypeOf01' USING JAR
'hdfs:///warehouse/tablespace/managed/TypeOf01-1.0-SNAPSHOT.jar';
```

3. Restart the HiveServer.

You can either delete the hiveserver2-0 pod using Kubernetes, or, you can edit an HS2 related configuration, and CDP restarts the HiveServer pod.



Note: If you plan to run UDFs on LLAP, you must restart the query executor and query coordinator pods after registering the UDF.

4. Verify whether the UDF is registered.

```
SHOW FUNCTIONS;
```

You scroll through the output and find default.typeof.

Calling the UDF in a query

After registration of a UDF, you do not need to restart Hive before using the UDF in a query. In this example, you call the UDF you created in a SELECT statement, and Hive returns the data type of a column you specify.

Before you begin

- For the example query in this task, you need to create a table in Hive and insert some data.

This task assumes you have the following example table in Hive:

students.name	students.age	students.gpa
fred flintstone	35	1.28
barney rubble	32	2.32

- As a user, you need to have permission to call a UDF, which a Ranger policy can provide.

Procedure

1. Use the database in which you registered the UDF.

```
USE default;
```

2. Query Hive using the direct reference method:

```
SELECT students.name, udftypeof(students.name) AS type FROM students WHERE  
age=35;
```

You get the data type of the name column in the students table:

students.name	type
fred flintstone	Type: org.apache.hadoop.hive.serde2.io.HiveVarcharWri table