Using Hive

Contents:

This section describes how you interact through the Trifacta® platform with your Hive data warehouse.

- Hive is an open-source, scalable data warehouse built on top of the Hadoop infrastructure to enable SQL-like access to a datastore where processing is converted to Hadoop map/reduce tasks. Hive users can interact directly with the databases and tables using HiveQL, a querying language similar to SQL. For more information, see *https://en.wikipedia.org/wiki/Apache_Hive*.

Limitations

None.

Uses of Hive

The Trifacta platform can use Hive for the following tasks:

1. Create datasets by reading from Hive tables.
2. Write data to Hive.

Before You Begin Using Hive

- **Read Access:** Your Hadoop administrator must configure read permissions to Hive databases.
  - Your Hadoop administrator should provide a database table or tables for data upload to your Hive datastore.
- **Write Access:** You can write jobs directly to Hive or ad-hoc publish jobs results to Hive at a later time. See *Writing to Hive* below.

Secure Access

Depending on the security features you've enabled, the technical methods by which Trifacta users access Hive may vary. For more information, see *Configure Hadoop Authentication*.

Reading Partitioned Data

The Trifacta platform can read in partitioned tables. However, it cannot read individual partitions of partitioned tables.

> **Tip:** If you are reading data from a partitioned table, one of your early recipe steps in the Transformer page should filter out the unneeded table data so that you are reading only the records of the individual partition.

Storing Data in Hive

Your Hadoop administrator should provide datasets or locations and access for storing datasets within Hive.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

> **NOTE:** The Trifacta platform does not modify source data in Hive. Datasets sourced from Hive are read without modification from their source locations.

Reading from Hive

You can create a Trifacta dataset from a table or view stored in Hive. For more information, see *Hive Browser*.

For more information on how data types are imported from Hive, see *Hive Data Type Conversions*.

Notes on reading from Hive views using custom SQL

If you have enabled custom SQL and are reading data from a Hive view, nested functions are written to a temporary filename, unless they are explicitly aliased.

> **Tip:** If your custom SQL uses nested functions, you should create an explicit alias from the results. Otherwise, the job is likely to fail.

Problematic Example:

```
SELECT
 UPPER(`t1`.`colum1`),
 TRIM(`t1`.`column2`),...
```

When these are ready from a Hive view, the temporary column names are: `_c0`, `_c1`, etc. During job execution, Spark ignores the `column1` and `column2` reference.

Improved Example:

```
SELECT
 UPPER(`t1`.`column1`) as col1,
  TRIM(`t1`.`column2`) as col2,...
```

In this improved example, the two Hive view columns are aliased to the explicit column names, which are correctly interpreted and used by the Spark running environment during job execution.

Writing to Hive

You can write data back to Hive using one of the following methods:

NOTE: You cannot publish to a Hive database that is empty. The database must contain at least one table.

NOTE: If you are writing to unmanaged tables in Hive, create and drop & load actions are not supported.

- Job results can be written directly to Hive as part of the normal job execution. Create a new publishing action to write to Hive. See *Run Job Page*.
- As needed, you can publish results to Hive for previously executed jobs. See *Publishing Dialog*.
- For more information on how data is converted to Hive, see *Hive Data Type Conversions*.