Cloudera DataFlow for Data Hub 7.2.12

SQL Stream Builder

Date published: 2019-12-16
Date modified: 2021-10-25

https://docs.cloudera.com/

Legal Notice

Contents

Managing teams in Streaming SQL Console....4

Using the Streaming SQL Console....5
Console Page....6
Compose Tab....6
Tables Tab....7
Functions Tab....8
History Tab....9
SQL Jobs Tab....9
Data Providers Page....9
Materialized Views Page....10

Registering Data Providers in SSB....10
Managing registered Data Providers....11

Concept of tables in SSB....12
Supported tables in SSB....12

Job Lifecycle....13
Configuring SQL job settings....13
Stopping, restarting and editing SQL jobs....14
Sampling data for a running job....18
Managing session for SQL jobs....19
Executing SQL jobs in production mode....20

Creating User Defined Functions....21
Developing JavaScript functions....22
Adding Java to the Functions language option....23

Monitoring SQL Stream jobs....23

SQL Syntax Guide....25

Managing teams in Streaming SQL Console

You can manage your team, team members and invite new team members under the Teams menu on the Streaming SQL Console.

About this task

By default, a new user is assigned to the ssb_default team which is a default team for the administrator within SQL Stream Builder (SSB). Every user who can access the Streaming SQL Console on the same cluster, is automatically added and listed as Team Members in the ssb_default team. Only the administrator has the privilege to change the access level for a team member, and inactivate-activate a team member from the ssb_default team. Team members can create their own team. In this case, only the Team Owner can delete their team. A team can be deleted by the Team Owner of that certain team. A team cannot be deleted if it is a primary team of a user or it is the default team (ssb_default).

Every team member in a team (ssb_default team or user created team) can access the created Tables, User Defined Functions, Materialized Views and API keys within a team. A team member can also view the jobs submitted in a team they are a member of. A user can be a part of multiple teams, and can switch between them. On the Streaming SQL Console, the currently selected team is shown under the Active Team header.

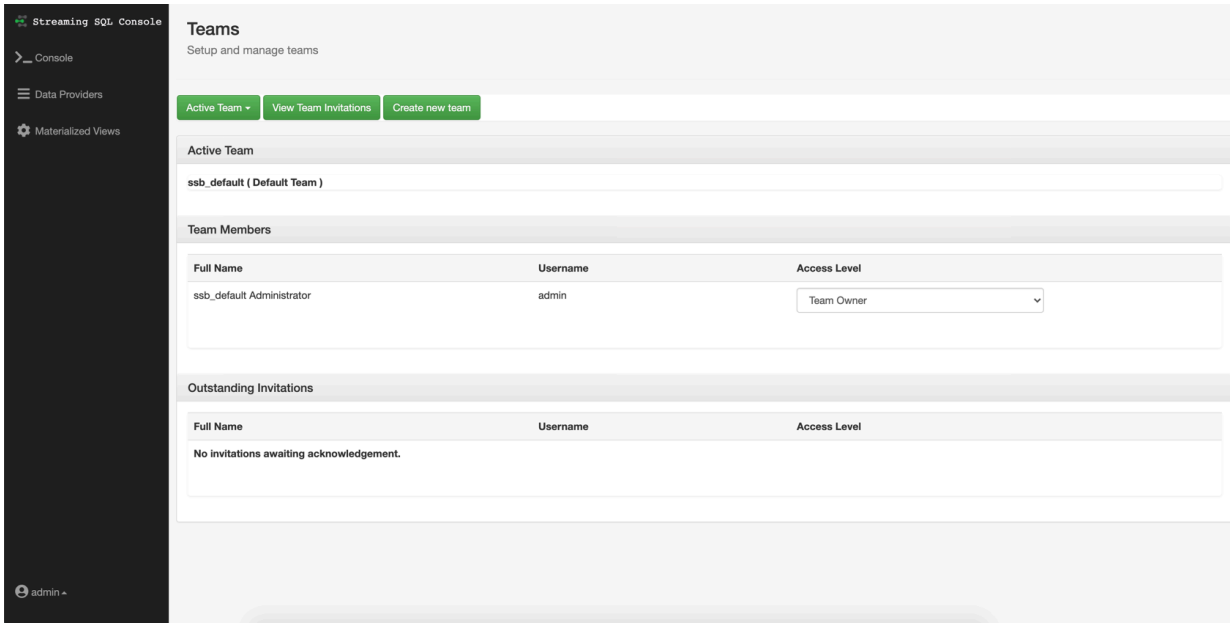
A team member can invite other members to join their team. The invitation can be accepted or ignored. To view the invitation within a team click the View Team Invitations button.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Select your username.
3. Click Teams.

   You are redirected to the **Teams** page.



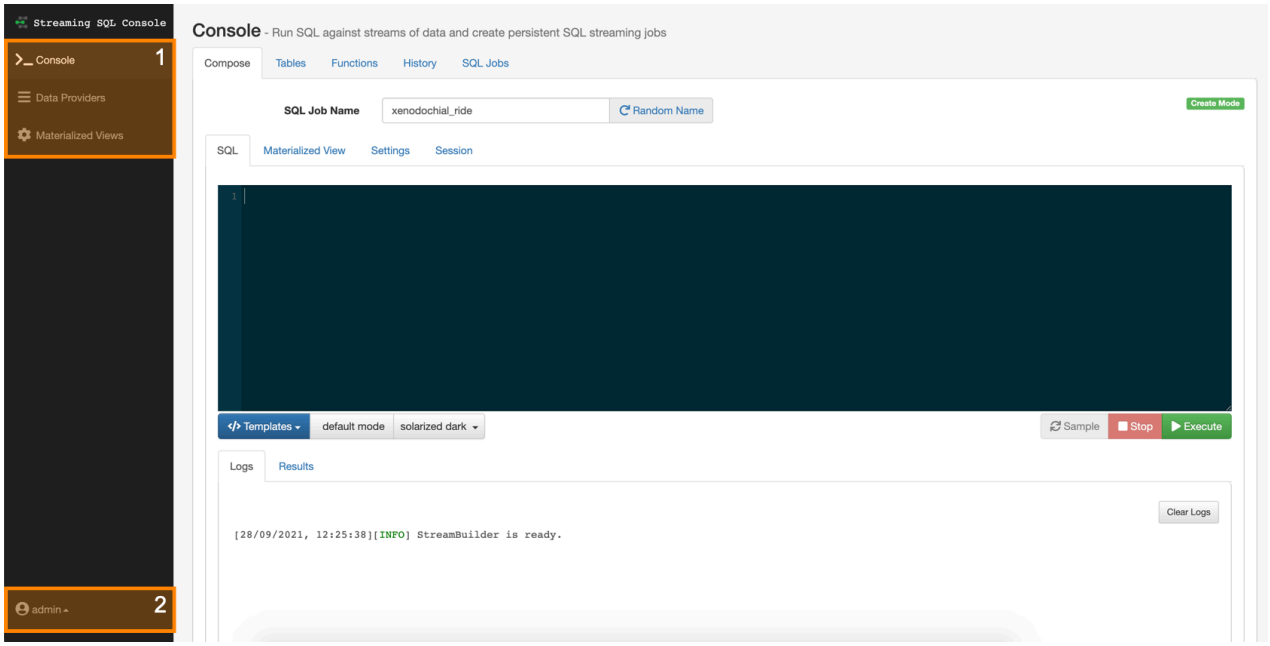
Using the Streaming SQL Console

The Streaming SQL Console is the user interface for the SQL Stream Builder. You can manage your queries, tables, functions and monitor the history of the SQL jobs using the SQL Stream Console.

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select the Streaming Analytics cluster.
3. Click Streaming SQL Console from the services.

   The Streaming SQL Console opens in a new window.

The following illustration details the main menu and the tabs of the user interface.



1. Main Menu

The **Main Menu** consist of the following pages:

- **Console** - The homepage of the Console where you can find the SQL window, the main tabs and audit tabs.
- **Data Providers** - The main page of Data Providers where you can add and manage the data providers and catalogs.
- **Materialized Views** - The main page of Materialized Views where you can review and manage the created Materialized View endpoints and API keys.

2. Profile Menu

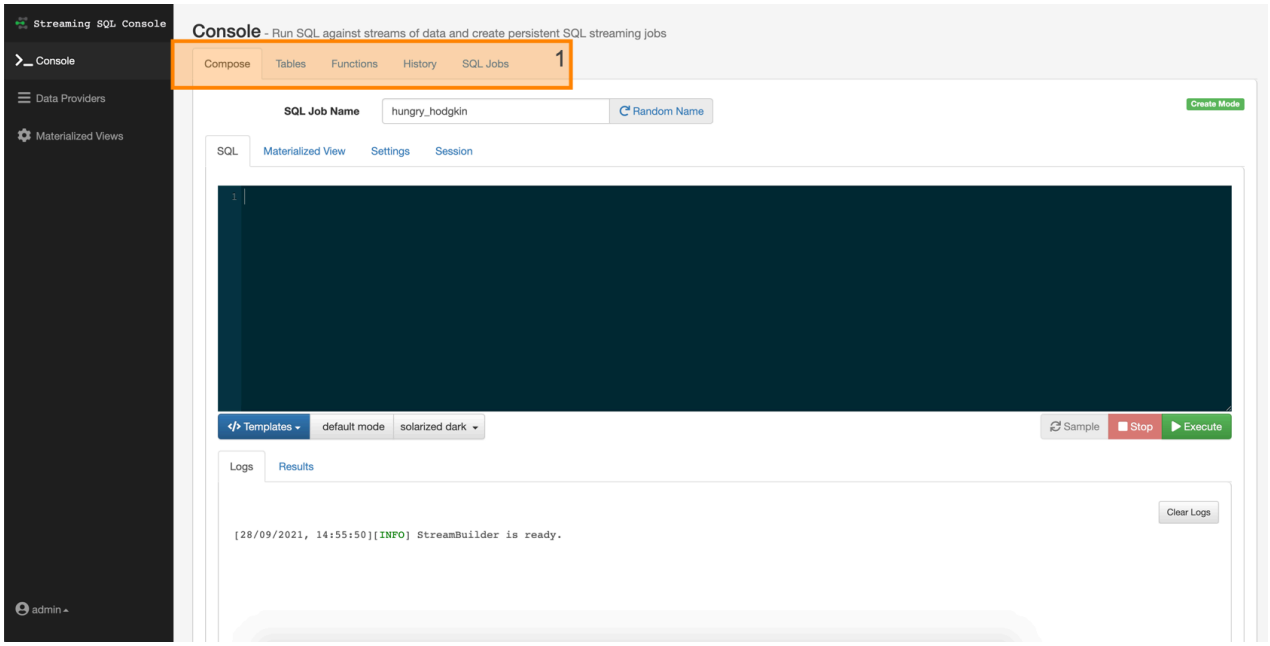
The **Profile Menu** consist of the following pages:

- **Profile** - The main page of the user profile where you can create new password.
- **Teams** - The main page of the Teams where you can create and manage your teams, invite other members to your teams.

Logout - You can use the Logout button from the **Profile Menu** to log out of the Streaming SQL Console.

Console Page

The Console page enables you to create your SQL jobs, and shows every SQL job related tabs, subtabs and audit tabs. By default the Compose tab is displayed with the SQL subtab, so you can easily create and execute your SQL queries.

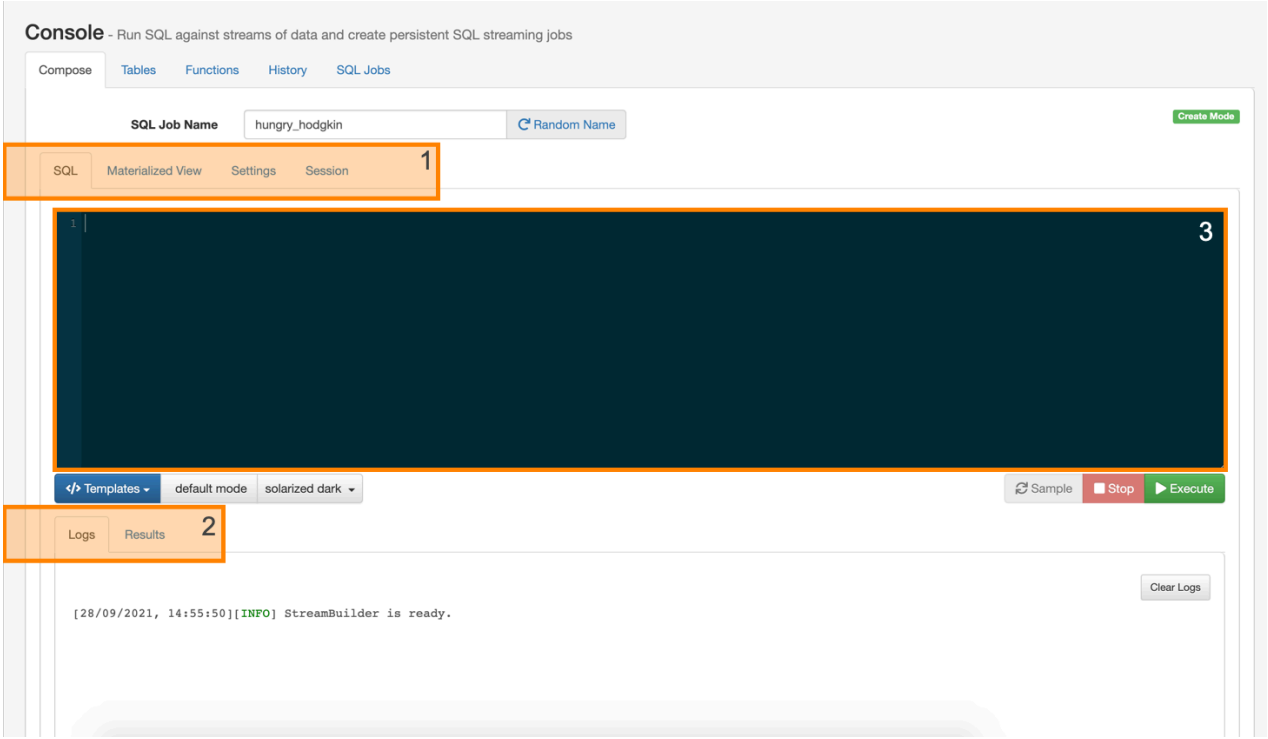


1. Main Tabs

- **Compose** - By default, the Compose tab is selected on the Console.
- **Tables** - You can view, manage and add tables on this tab.
- **Functions** - You can add the User Defined Functions to your SQL query.
- **History** - You can review the history of submitted SQL queries.
- **SQL Jobs** - You can review the history of submitted SQL jobs.

Compose Tab

The Console tab shows the alternate windows to create SQL queries and Materialized Views. You can also set the SQL job settings and review the properties of the running session. With the audit tabs, you can review log information of your SQL jobs, and also check the sampled data of your query results.



1. Subtabs

- **SQL** - By default, the SQL alternate window is displayed on the Compose tab where you can execute, stop and sample your SQL queries.
- **Materialized View** - The Materialized View alternate window displays settings to create Materialized Views for a SQL query.
- **Settings** - The Settings alternate window displays the SQL job relates settings that you can configure before starting a job.
- **Session** - The Session alternate window displays every property that is configurable for the running session.

2. Audit Tabs

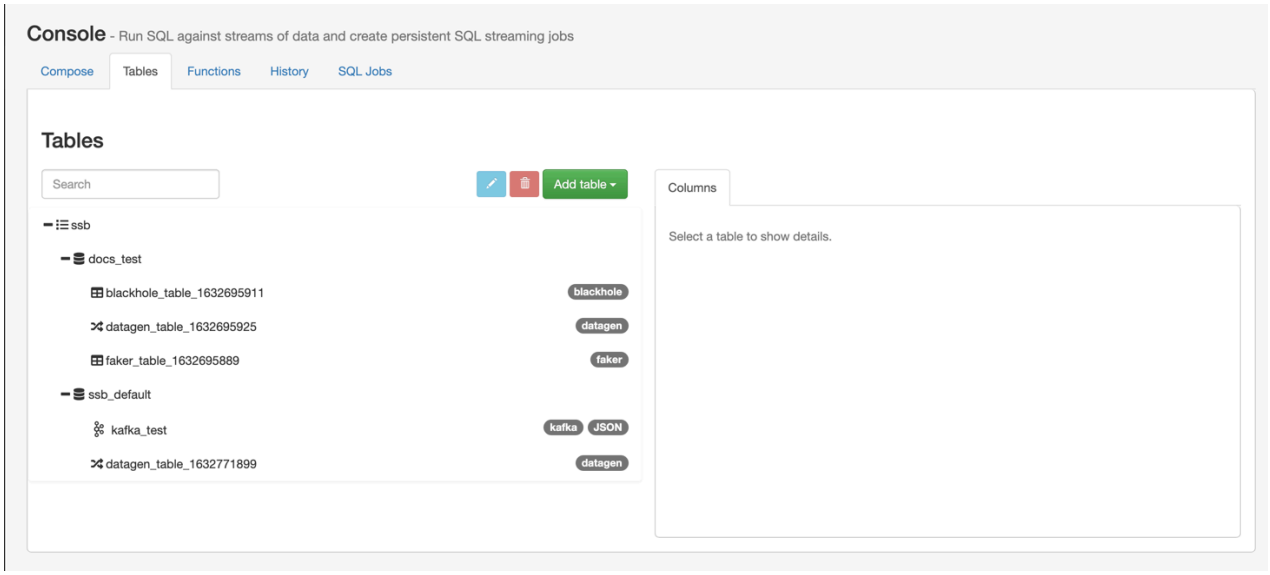
- **Logs** - The **Logs** audit tab displays the status of the SQL Console.
- **Results** - The **Results** audit tab displays the results of the executed SQL queries.

3. SQL Window

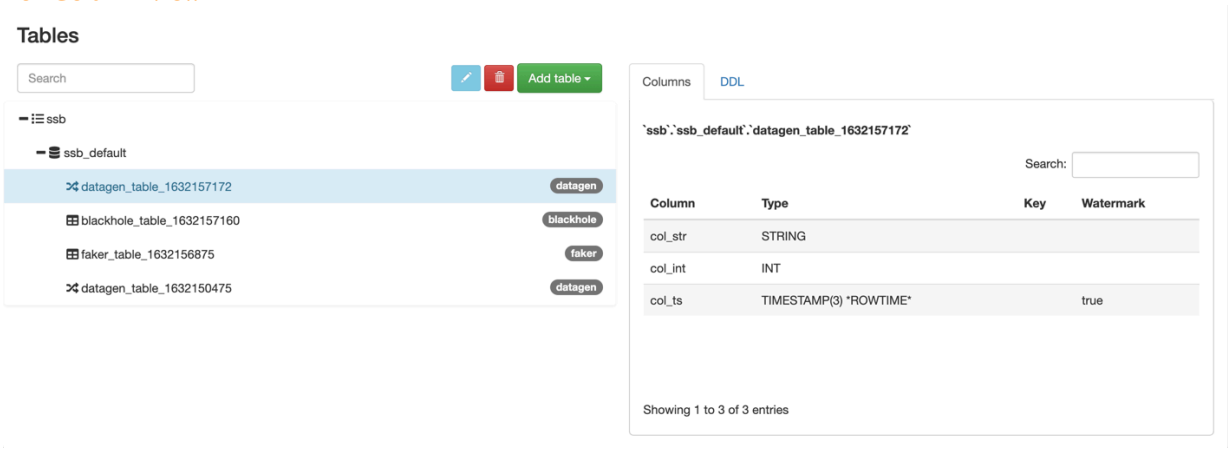
You can use the SQL window to create and add your SQL queries.

Tables Tab

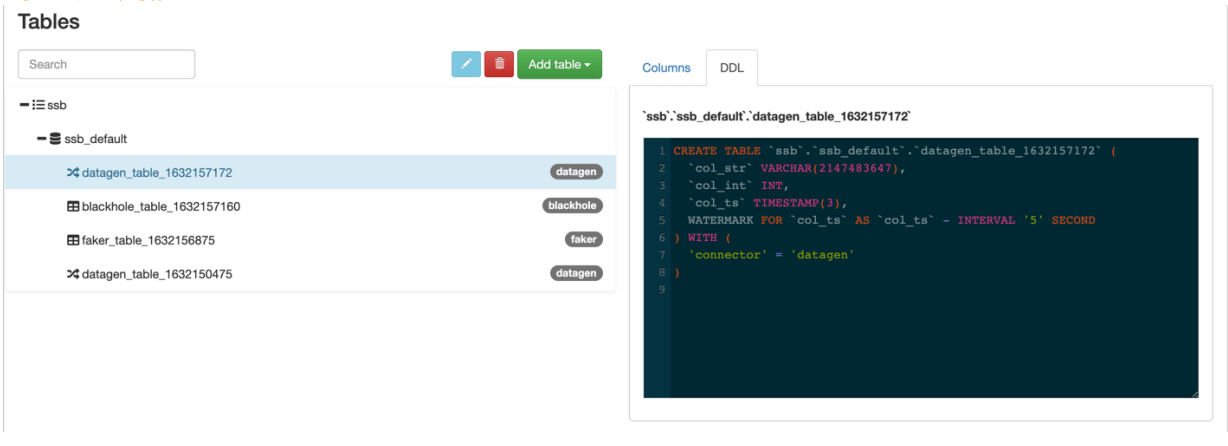
The Tables tab shows the list of created tables. You can switch between Column or DDL view of the tables.



For Column view

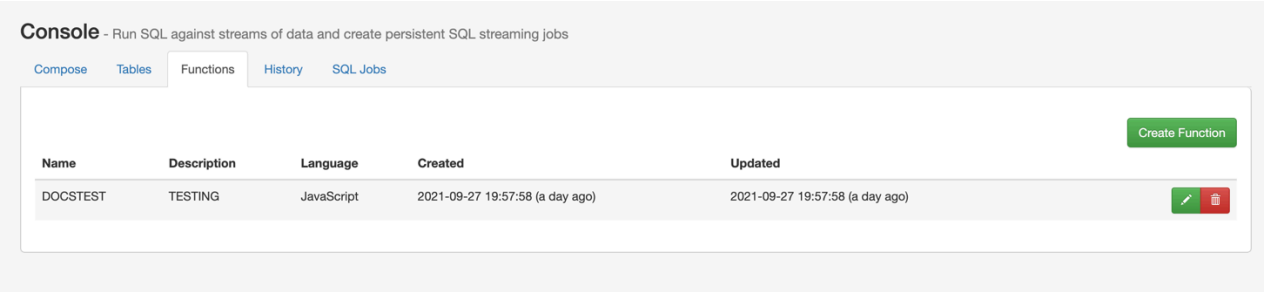


For DDL view



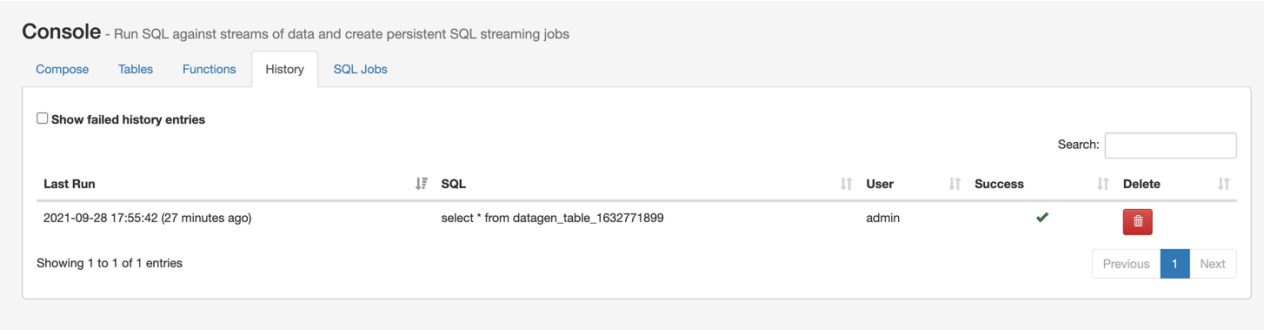
Functions Tab

The Functions tab shows the list of created User Defined Functions (UDF) that you can use in your SQL queries.



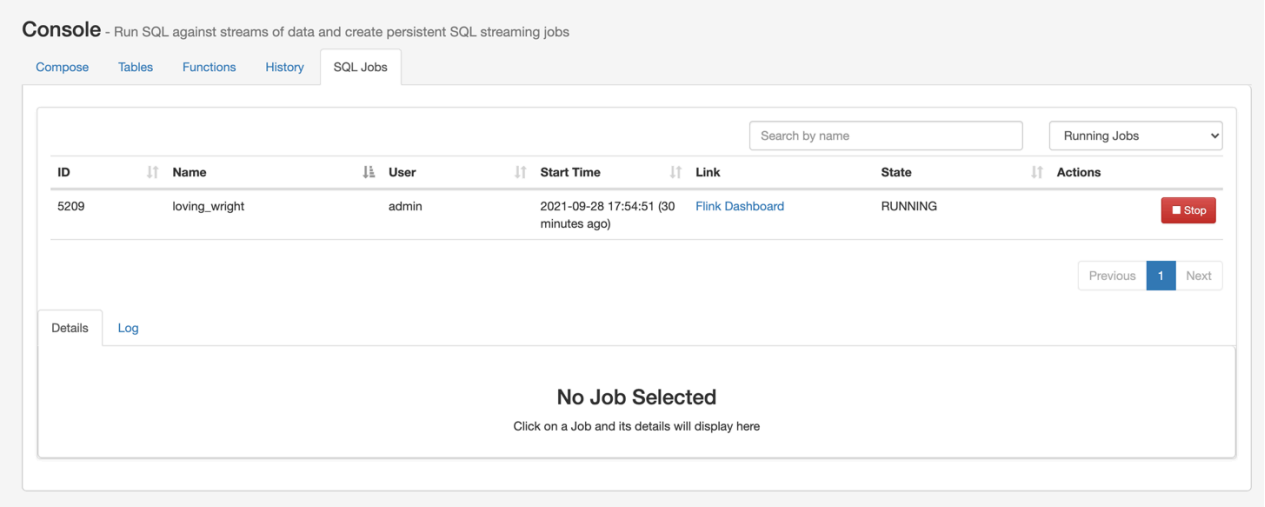
History Tab

The History tab shows the list of executed SQL queries. You have the option to only show the failed history entries. When clicking on an entry from the History, the SQL query is automatically imported to the SQL window.



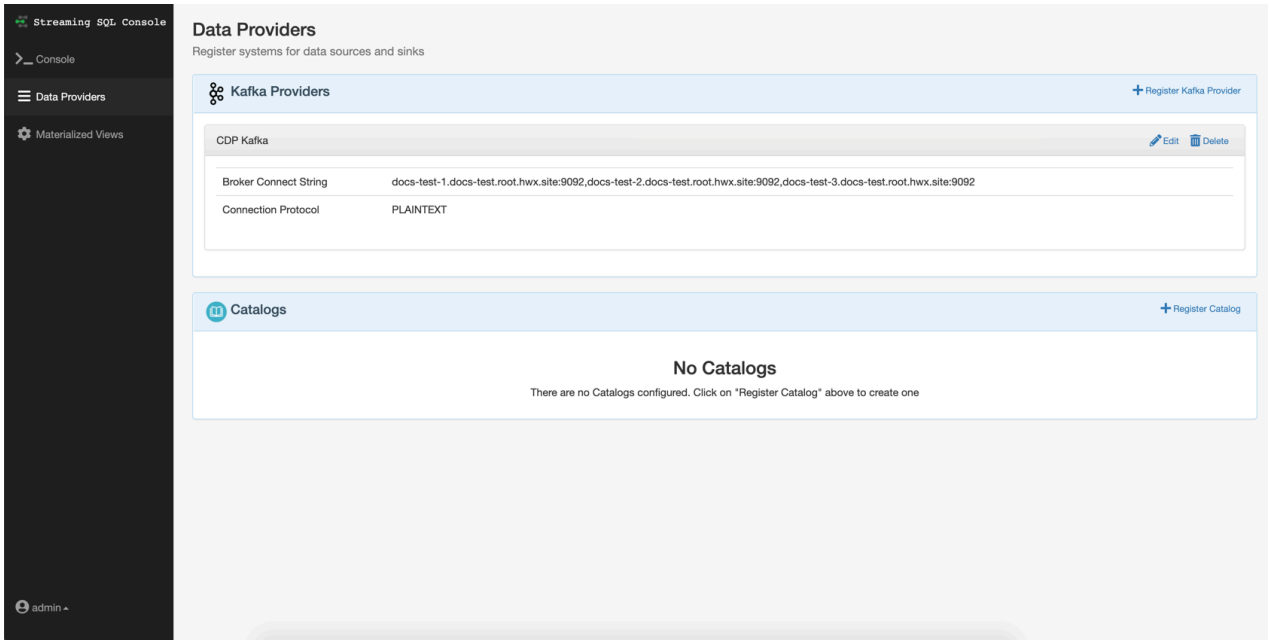
SQL Jobs Tab

The SQL Jobs tab include the list of running and stopped SQL jobs. You can also manage and stop the SQL jobs listed on the SQL Jobs tab. Using the Details and Log windows, you can review more information about a selected job.



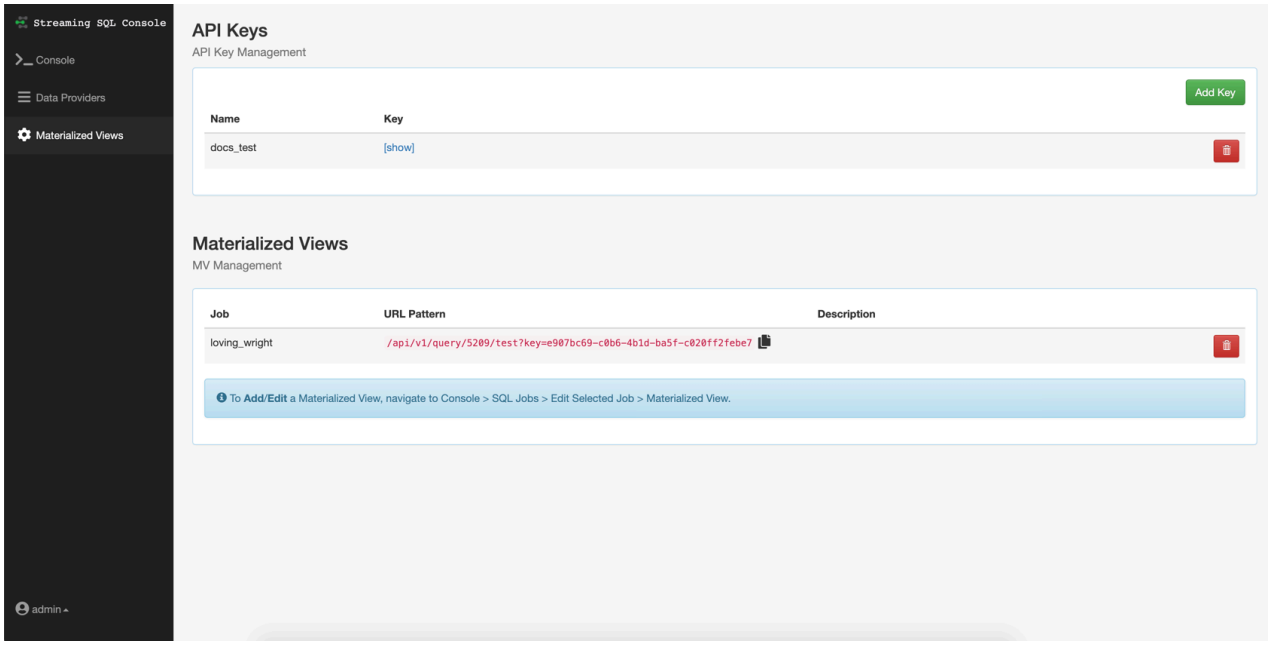
Data Providers Page

The Data Providers page enables you to register, edit and delete the data providers. The providers are displayed in two categories: Kafka providers and Catalogs.



Materialized Views Page

The Materialized Views page enables you to add API keys, manage API keys and manage Materialized Views.



Registering Data Providers in SSB

Data Providers are a set of data endpoints to be used as sources, sinks and catalogs. Data Providers allow you to connect to an already installed component on your cluster, then use that provider for adding tables in SQL Stream Builder (SSB).

You can access the **Data Providers** page through the Streaming SQL Console:

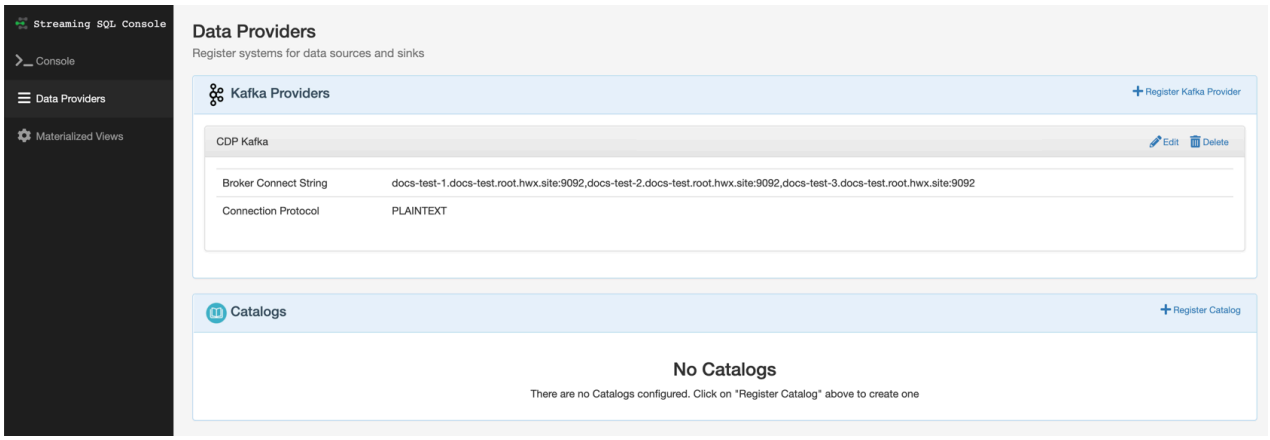
1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select the Streaming Analytics cluster.
3. Click Streaming SQL Console from the services.

   The Streaming SQL Console opens in a new window.
4. Click Data Providers on the main menu.

   You are redirected to the Data Providers page.

You can register Kafka as a data provider, or Kudu, Hive and Schema Registry as a catalog. When registering the components, SSB can access the already existing topics from Kafka, tables from Kudu and Hive, and the schema in Schema Registry. This also means that when you update a data provider, for example add new topics, tables and schemas, SSB automatically detects the changes.

You can also manage your data providers after registering them. You can Edit the providers if there is any change in the connection. You can also Delete them when you no longer need the specific provider.



Related Information

Adding Kafka as Data Provider
Adding catalogs as Data Provider

Managing registered Data Providers

You can edit or delete the registered Data Providers if you need to change their configurations or if you no longer need them.

Editing registered Data Providers

1. Click Data Providers from the main menu.
2. Search for the Kafka provider or catalog you want to modify.
3. Click Edit.

   The Edit Provider or Catalog window appears.
4. Change the settings as required.

   Note: You must validate the modified catalog before saving the changes.
5. Click Save Changes.

Deleting registered Data Providers

1. Click Data Providers from the main menu.

2. Search for the Kafka provider or catalog you want to modify.
3. Click Delete.
4. Click Confirm to delete the provider or catalog.

Concept of tables in SSB

The core abstraction for Streaming SQL is a Table which represents both inputs and outputs of the queries. SQL Stream Builder tables are an extension of the tables used in Flink SQL to allow a bit more flexibility to the users.

A Table is a logical definition of the data source that includes the location and connection parameters, a schema, and any required, context specific configuration parameters. Tables can be used for both reading and writing data in most cases. You can create and manage tables either manually or they can be automatically loaded from one of the catalogs as specified using the Data Providers section.

In SELECT queries the FROM clause defines the table sources which can be multiple tables at the same time in case of JOIN or more complex queries.

When you execute a query, the results go to the table you specify after the INSERT INTO statement in the SQL window. This allows you to create aggregations, filters, joins, and so on, and then route the results to another table. The schema for the results is the schema that you have created when you ran the query.

For example:

```
INSERT INTO air_traffic -- the name of the table sink
SELECT
lat,lon
FROM
airplanes -- the name of the table source
WHERE
icao <> 0;
```

Related Information

Creating Kafka tables

Creating tables with Flink DDL

Supported tables in SSB

SQL Stream Builder (SSB) supports different table types to ease development and access to all kinds of data sources. When creating tables in SQL Stream Builder, you have the option to either add them manually, import them automatically or create them using Flink SQL depending on the connector you want to use.

Kafka Tables

Apache Kafka Tables represent data contained in a single Kafka topic in JSON, AVRO or CSV format. It can be defined using the Streaming SQL Console wizard or you can create Kafka tables from the following pre-defined templates:

- CDP Kafka
- Kafka
- Upsert Kafka

Tables from Catalogs

SSB supports Kudu, Hive and Schema Registry as catalog providers. After registering them using the Streaming SQL Console, the tables are automatically imported to SSB, and can be used in the SQL window for computations.

Note: You cannot edit the properties of the already existing tables that are automatically imported from the catalogs. To distinguish between editable and not editable tables, in other words, user defined and catalog tables, the Edit and Delete table options are not available on the Tables page as the illustration shows below:

Flink Tables

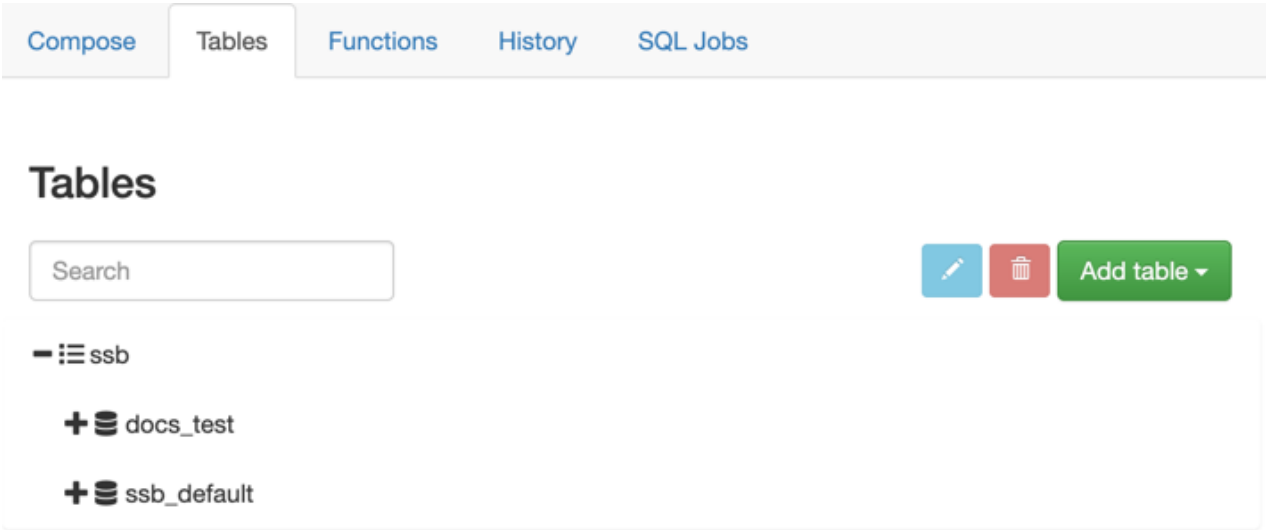
Flink SQL tables represent tables created by the standard CREATE TABLE syntax. This supports full flexibility in defining new or derived tables and views. You can either provide the syntax by directly adding it to the SQL window or use one of the predefined DDL templates:

- Blackhole
- Datagen
- Faker
- Filesystem
- JDBC
- MySQL CDC
- Oracle CDC
- PostgreSQL CDC

Webhook Tables

Webhooks can only be used as tables to write results to. When you use the Webhook Tables the result of your SQL query is sent to a specified webhook.

After creating your tables for the SQL jobs, you can review and manage them on the Tables tab in Streaming SQL Console. The created tables are organized based on the teams a user is assigned to.

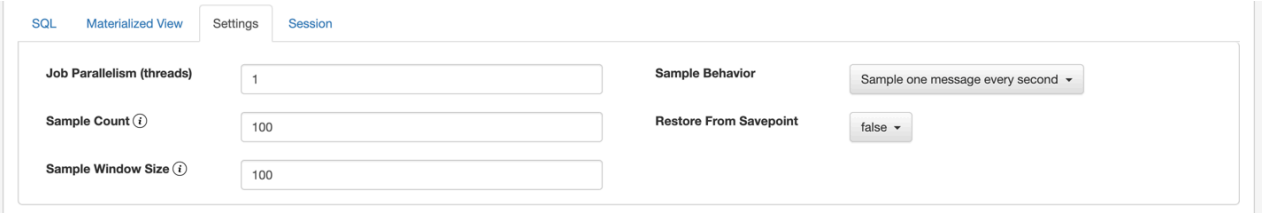


Job Lifecycle

Configuring SQL job settings

If you need to further customize your SQL Stream job, you can add more advanced features to configure the job restarting method and time, threads for parallelism, sample behavior, exatly once processing and restoring from savepoint.

Before running a SQL query, you can configure advanced features by clicking on the Settings tabs at the SQL window.



Job parallelism (threads)

The number of threads to start to process the job. Each thread consumes a slot on the cluster. When the Job Parallelism is set to 1, the job consumes the least resources. If the data provired supports parallel reads, increasing the parallelism can raise the maximum throughput. For example, when using Kafka as a data provider, setting the parallelism to the equal number as the partitions of the topic can be a starting point for performance tuning.

Sample Count

The number of sample entries shown under the Results tab. To have an unlimited number of sample entries, add 0 to the Sample Count value.

Sample Window Size

The number of sample entries to keep in under the Results tab. To have an unlimited number of sample entries, add 0 to the Sample Window Size value.

Sample Behavior

You have the following options to choose the behavior of the sampled data under the Results tab:

- Sample all messages
- Sample one message every second
- Sample one message every five seconds

Restore From Savepoint

You can enable or disable restoring a SQL job from a Flink savepoint after stopping it. The savepoint is saved under hdfs:///user/flink/savepoints by default.

Stopping, restarting and editing SQL jobs

As a SQL Stream job processes streaming data, you need to stop the job to finish the process. You can restart a SQL Stream job after stopping it. In case you need to update or change the configurations that you have set for a SQL Stream job, you can restart it. You can navigate through the job life cycle using the Streaming SQL Console.

Stopping a SQL Stream job

You can stop a running job either on the **Compose** or the **SQL Jobs** page.
Stopping job from Compose page

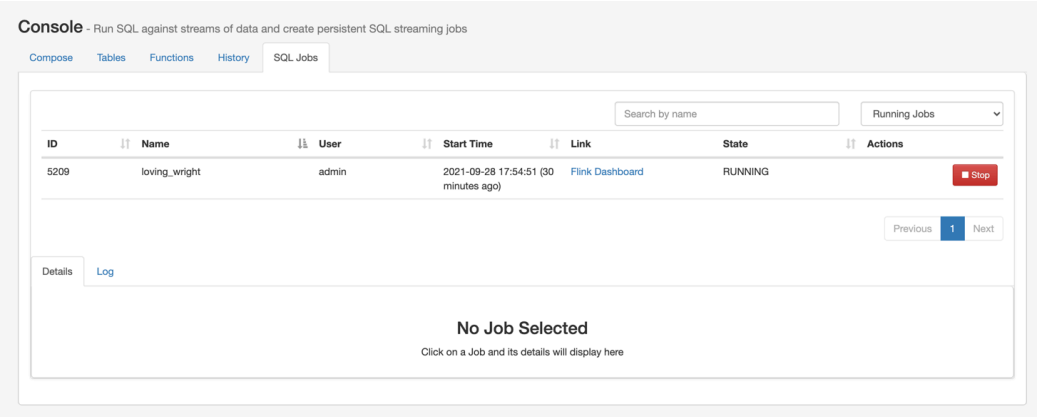
1. Select Console from the main menu.

   By default, you are on the Compose page when selecting Console from the main menu.
2. Click Stop under the SQL window.

Stopping job from SQL Jobs page

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.



3. Click on the job you want to stop.
   a. You can further filter down the results, by directly searching for the job name in the Search field.
4. Click Stop under Actions.

Restarting a running SQL Stream job

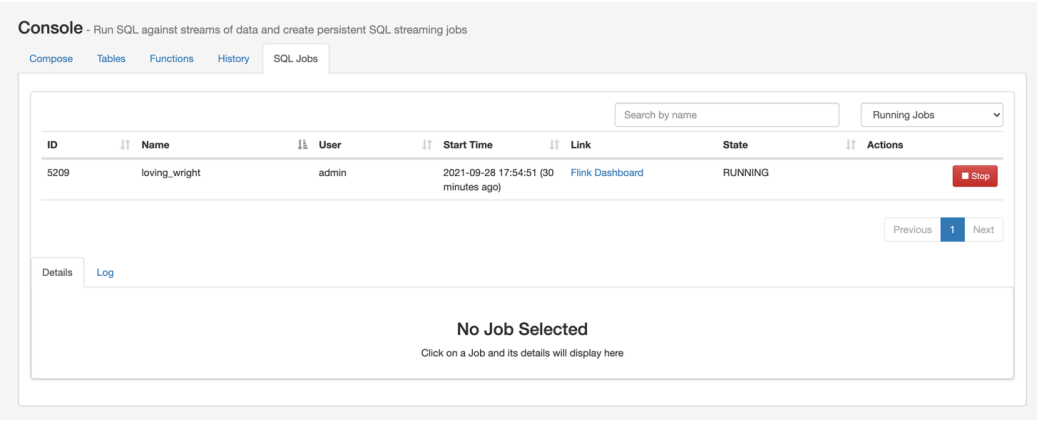
You can restart a running SQL job using the Restart button under the SQL window.

In case the job is running in the background, you can load it with its properties from the SQL Jobs tab.

Restarting job from SQL Jobs page

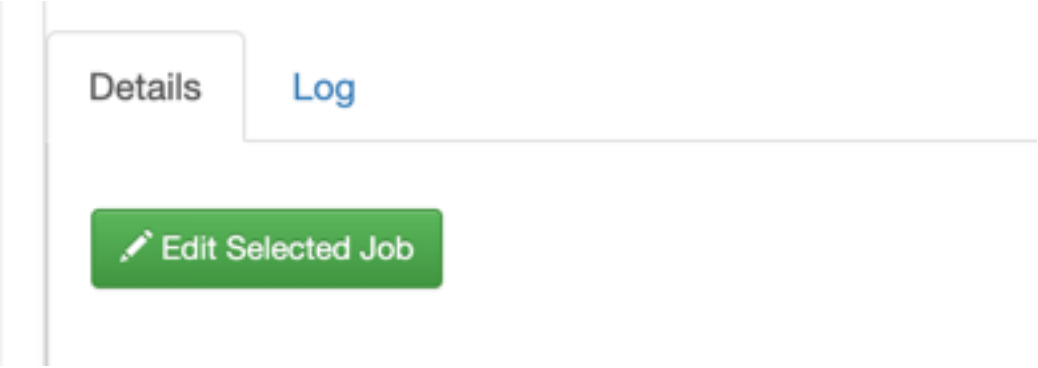
1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.



3. Click on the job you want to restart.
   a. You can further filter down the results, by directly searching for the job name in the Search field.

4. Click Edit Selected Job under the **Details** tab.



The SQL job and its configuration is loaded in the SQL window on the **Compose** page.

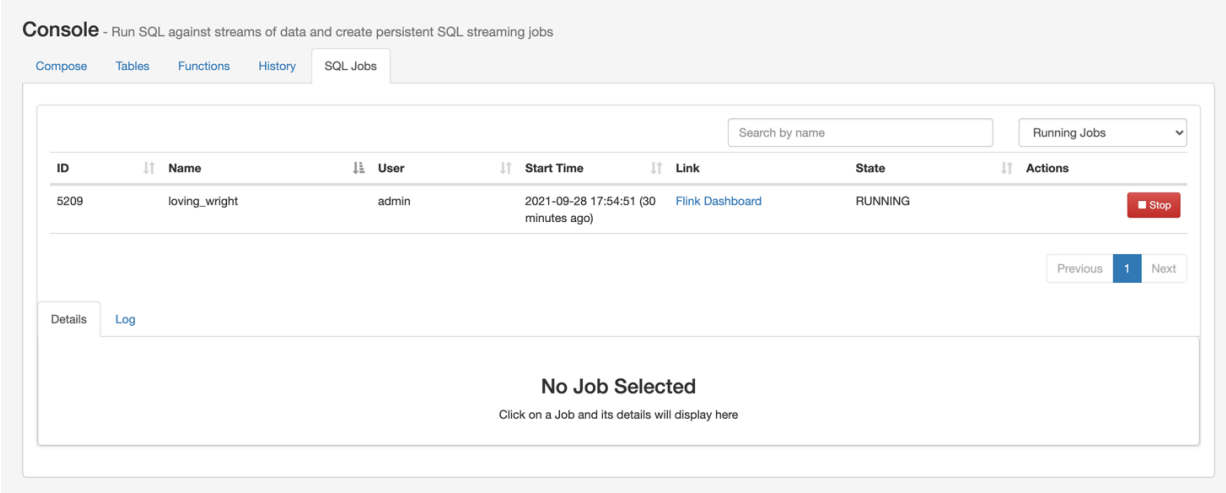
5. Click Restart.

Restarting a stopped SQL Stream job

You can restart a SQL job that was previously stopped by locating it in the SQL Jobs page.

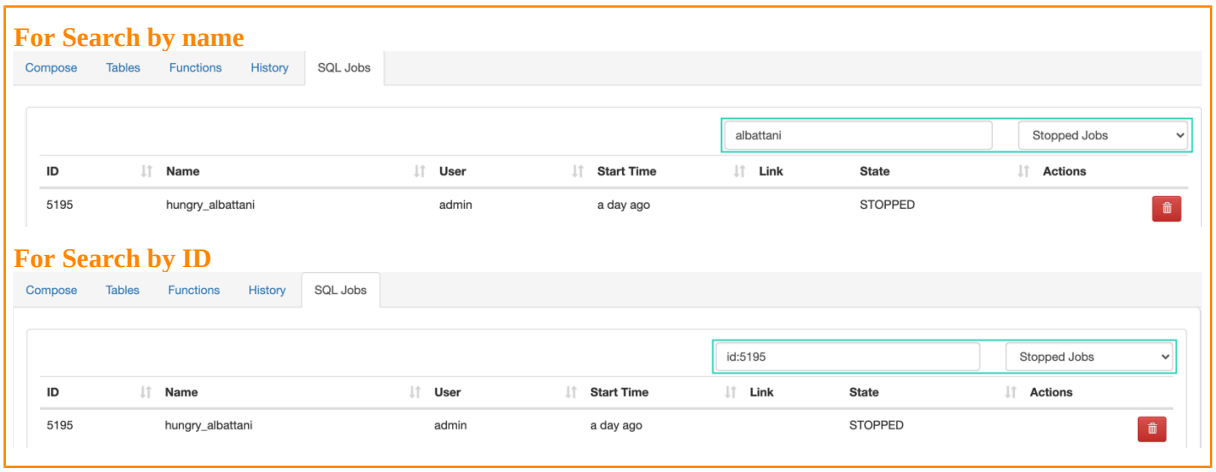
1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.

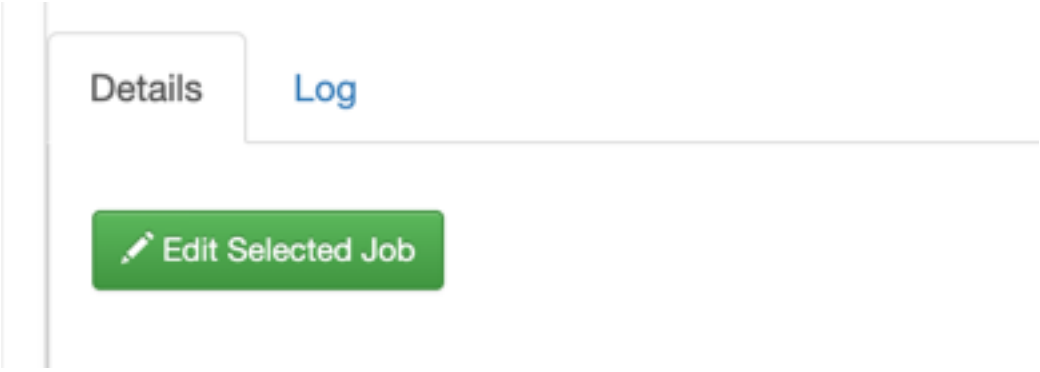


3. Select Stopped Jobs from the drop-down menu.

4. Click on the job you want to restart.
   a. You can further filter down the job list by searching for the job name, or locate a specific job ID by prefixing your search with id.



5. Click Edit Selected Job under the **Details** tab.



   The SQL job and its configuration is loaded in the SQL window on the **Compose** page.
6. Click on Execute.

Note: If you are using Materialized Views, the same Materialized View parameters will be selected. Make sure that the configured parameters are still appropriate for the job if the source schemas have been modified.

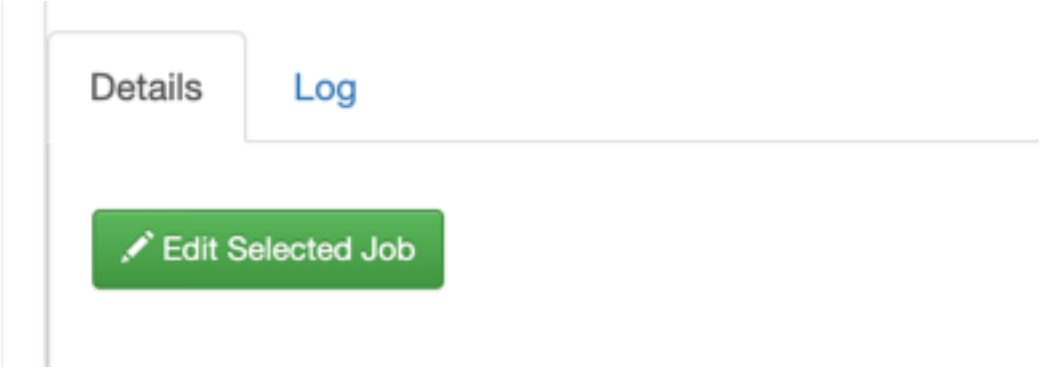
Editing a SQL Stream job

Before editing, you must stop the running SQL job. After modifying the properties of the job, you need to execute them again to apply the changes.

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.
3. Select Stopped Jobs from the drop-down menu.
4. Click on the job you want to restart.
   a. You can further filter down the results, by directly searching for the job name in the Search field.

5. Click Edit Selected Job under the **Details** tab.



The SQL job and its configuration is loaded in the SQL window on the **Compose** page.

6. Edit any configuration of the selected SQL job.

You need to click on Advanced Settings to display more configuration of the SQL job.

7. Click on Execute.

Deleting a SQL Stream job

You can delete a stopped SQL job from SQL Stream Builder on the SQL Jobs tab. By deleting the SQL job, you remove them from the list of stopped jobs.

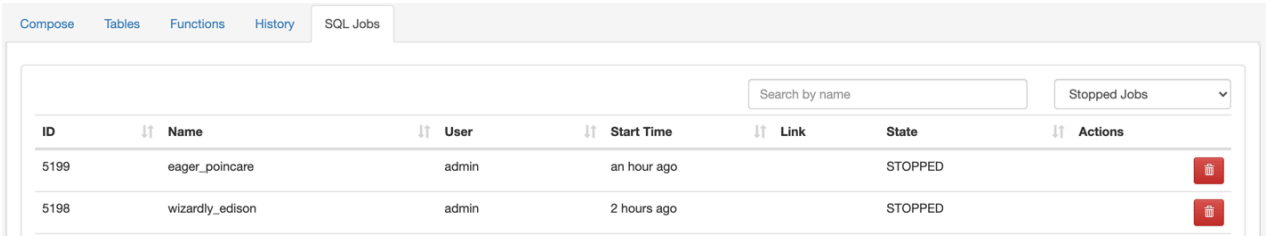
1. Click Console on the main menu.
2. Select SQL Jobs tab.

By default, Running Jobs are displayed on the **SQL Jobs** page.

3. Select Stopped Jobs from the drop-down menu.

The list of stopped jobs is diplayed on the **SQL Jobs** page.

4. Click on Delete under Actions.



Sampling data for a running job

You can sample data from a running job. This is useful if you want to inspect the data to make sure the job is running correctly and producing the results you expect.

About this task

Sampling the results to your browser allows you to inspect the queried data and iterate on your query. You can sample 100 rows in the Results tab by clicking on the Sample button in the Console. In case you do not add any sink to the SQL job, the results automatically appear in the Results tab.

Procedure

1. Select Console on the main menu.
2. Go to the SQL Jobs tab.

3. Select the job you want to edit.
4. Go to the Details tab at the bottom.
5. Click Edit Selected Job.

   The SQL window in Edit Mode appears.
6. Click Sample.

Results

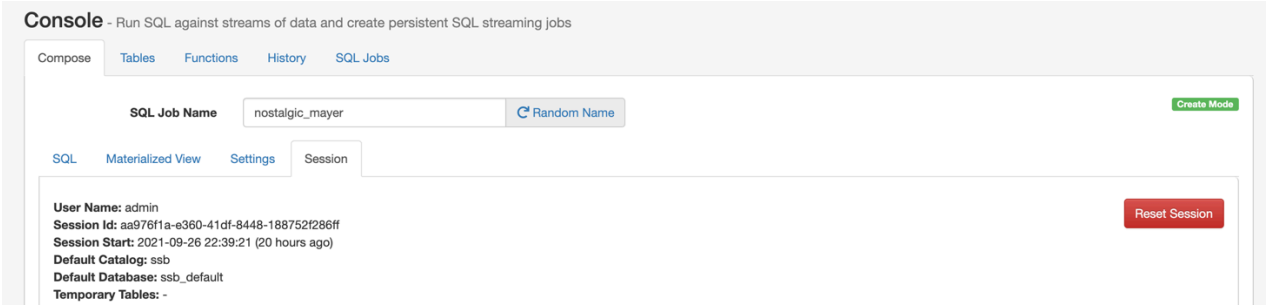
Sample results are displayed in the results window. If there is no data meeting the SQL query, sampling stops after a few attempts.

Managing session for SQL jobs

By default, the SQL Stream jobs are running in a session cluster. This means that multiple Flink jobs run in the same YARN session sharing the cluster, allocated resources, the Job Manager and Task Managers. The session starts when you open the Streaming SQL Console. You can reset the session, and set the properties of the session using the Streaming SQL Console.

Resetting a session

When you reset your session, every configuration, temporary table and view, default database and catalog will be lost.



1. Navigate to the **Streaming SQL Console**.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Select Session on the Console page.
3. Click Reset.

Configuring properties for a session

You can configure the session properties using the SET statement in the SQL window.

1. Navigate to the **Streaming SQL Console**.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.

2. Use the SET statement in the **SQL window** to configure a session property.

   Example:

   ```
   SET state.backend=rocksdb
   ```

3. Click Execute.

Executing SQL jobs in production mode

As by default the SQL jobs are running in a session cluster, there is a risk in case of a cluster failure that every job is affected within that cluster. However, you can set a per-job production mode in SQL Stream Builder to create a dedicated environment for your production jobs.

Production mode means that separately from the running session in SSB, you deploy a SQL job (Flink job) in per-job mode with a dedicated YARN cluster that is configured specifically to that particular production job.

To run your SQL jobs in production mode, you need to add a --PROD prefix to the SQL window at the beginning of your SQL query, and execute the SQL query after setting the execution target and properties in the same window:

```
--PROD
set 'execution.target' = 'yarn-per-job';
set 'logging.configuration.file' = '/tmp/log4j.properties';
select * from datagen_table_1631781644;
```

In the above example, the production mode is indicated as --PROD, and the execution target is set to per-job to create a new YARN application for the job. Setting the execution target to per-job allows you to have an individual cluster for the specific job. The additional properties that you configure using the SET statement overwrites the properties that are configured for the running session. However, when you set properties for the production mode, the settings of the session cluster are not affected.

1. Navigate to the **Streaming SQL Console**.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Specify a job name in the SQL Job Name field, or click Random Name.
3. Add --PROD to the **SQL window**.
4. Set the execution mode to per-job.

   ```
   set 'execution.target' = 'yarn-per-job';
   ```

5. Add additional configuration to the production job.

   For the list of configurable parameters, see *Session cluster properties* section.
6. Add a SQL statement you want to execute

   Example:

   ```
   --PROD
   set 'execution.target' = 'yarn-per-job';
   set 'state.backend' = 'rocksdb';
   select * from faker_table_1631781644;
   ```

7. Click Execute.

Creating User Defined Functions

With SQL Stream Builder, you can create user functions to write powerful functions in JavaScript, that you can use to enhance the functionality of SQL.

About this task

User functions can be simple translation functions like Celsius to Fahrenheit, more complex business logic, or even looking up data from external sources. User functions are written in JavaScript. When you write them, you create a library of useful functions.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Select Console from the left-hand menu.
3. Select Functions > Add .
4. Name the function HELLO_WORLD, give it a short description, select JavaScript as the language.
5. Select STRING as output type and add STRING to the input type by selecting it from the list and clicking the plus button.
6. Paste this code into the JavaScript editor:

```
// check to see if the card is VISA
function HELLO_WORLD(card){
    var cardType = "Other";
    if (card.charAt(0) == 4){
      cardType = "Visa";
    }
    return cardType;
}
HELLO_WORLD($p0); // this line must exist
```

7. Click Save.
8. Once created, you can use a User Function in your SQL statement:

```
-- simple usage
SELECT HELLO_WORLD(card) AS IS_VISA
FROM ev_sample_fraud;

-- in the predicate
SELECT amount, card
FROM ev_sample_fraud
```

```
WHERE HELLO_WORLD(card) = "Visa";
```

Note: Valid inputs can be a field in the source virtual table or any other valid input. Functions must be in upper case.

Note: User Functions have access to the Java 8 API, this increases the overall usefulness and power. For example:

```
function GETPLANE(icao) {
  try {
    var c = new java.net.URL('http://yyyyyy.io' + icao).openConnec
tion();
    c.requestMethod='GET';
    var reader = new java.io.BufferedReader(new java.io.InputStreamRe
ader(c.inputStream));
    return reader.readLine();
  } catch(err) {
    return "Unknown: " + err;
  }
}
GETPLANE($p0);
```

Developing JavaScript functions

When developing JavaScript functions that are more complicated than just simple logic, it is recommended to use the jjs command-line utility to create and iterate while writing functions.

About this task

After the function performs the required task, migrate it to the console. Additionally these files/functions can be saved in a source code control system like git/Github.

Procedure

1. Create a file for your function.

   Note: It is recommended to name the file with the same name as that of the function.

2. Create some sample input when calling the function.
3. Call jjs on the command line to test the function.

```
$>cat TO_EPOCH.js
function TO_EPOCH(strDate) {
   var strFmt = "yyyy-MM-dd HH:ss:mm";
   var c = new java.text.SimpleDateFormat(strFmt).parse(strDate).getTime()
/1000;
   return c.toString();
}

print(TO_EPOCH("2019-02-02 22:23:13"));

then
$>jjs TO_EPOCH.js
1549167203
```

What to do next

After you have successfully developed the JavaScript code, copy and paste only the function to your code window when creating the JavaScript function in SQL Stream Builder.

Adding Java to the Functions language option

You can create User Defined Functions (UDF) using Java after manually adding the UDF function JAR file that contains the UDF class to the Flink connectors. After creating the function in the SQL window, you can select Java as a language for the UDFs.

Procedure

1. Create a Java UDF function JAR file based on the following example:

```
package udf;
import org.apache.flink.table.functions.ScalarFunction;

public class UdfTest extends ScalarFunction {
   public String eval(String input){
       return "Hello World " + input;
   }
}
```

Note: The function needs to be named as ‘eval’ in the JAR file.

2. Copy the JAR file to the flink connectors folder:

```
scp <jar_location>/<jar_filename> <workload_username>@<datahub_hostname>:/
usr/share/flink-connectors
```

3. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
4. Run the following query to create the function:

```
CREATE FUNCTION myFunction AS 'udf.UdfTest’ LANGUAGE java;
```

5. Test the function by running a SELECT query.

```
SELECT myFunction('test');
```

Monitoring SQL Stream jobs

You can use the Streaming SQL Console to review the status, properties and log of your SQL Stream jobs executed in SQL Stream Builder. Using the Flink Dashboard, you can also monitor the Flink job that is submitted when you execute a SQL query.

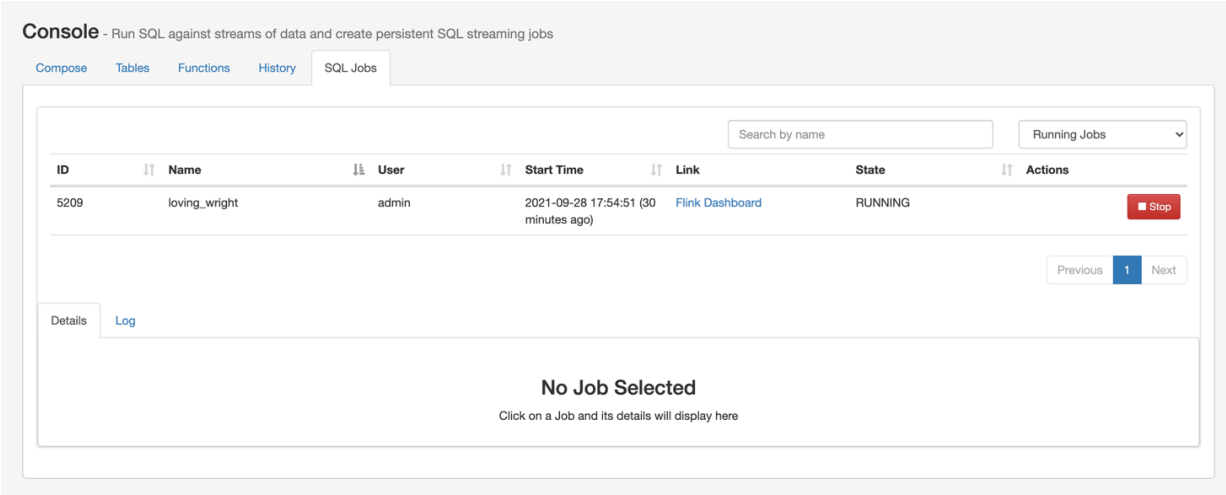
Using the Streaming SQL Console

When using the **SQL Jobs** page in Streaming SQL Console to monitor your SQL jobs, you can review the ID, the Name, the Start time, the State of the submitted jobs, and the User who submitted the SQL jobs. When monitoring running jobs, you are also able to open the Flink Dashboard, and stop the job using the Stop button under Actions. The Flink Dashboard link and Stop button are not available for Stopped Jobs.

1. Click Console on the main menu.

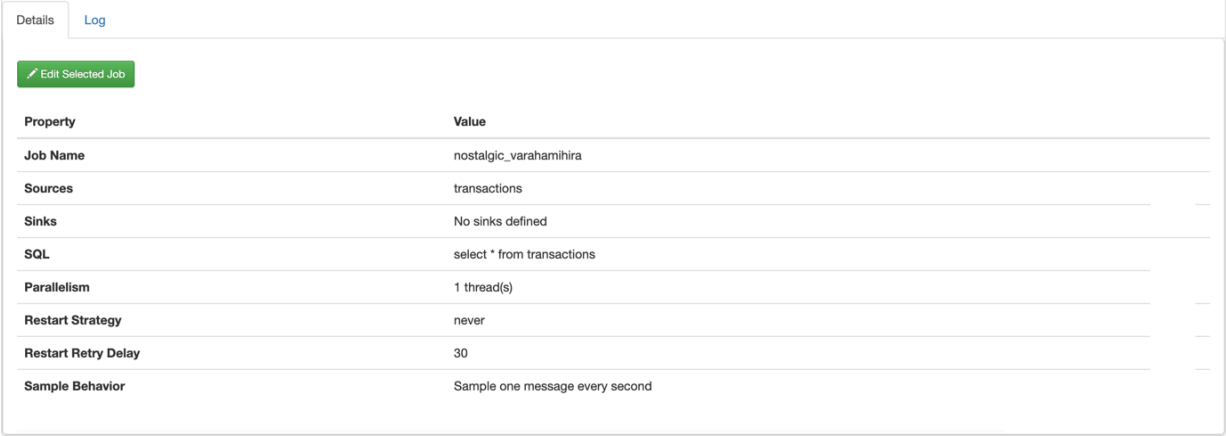
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.

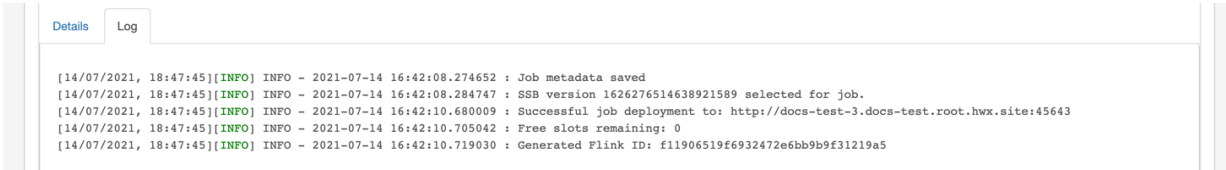


3. Select Running Jobs or Stopped Jobs from the drop-down menu.
4. Click on the job you want to monitor.

   a. You can further filter down the results, by directly searching for the job name in the Search field.
5. Select Details tab on the bottom panel to display additional details and configurations about the SQL job.

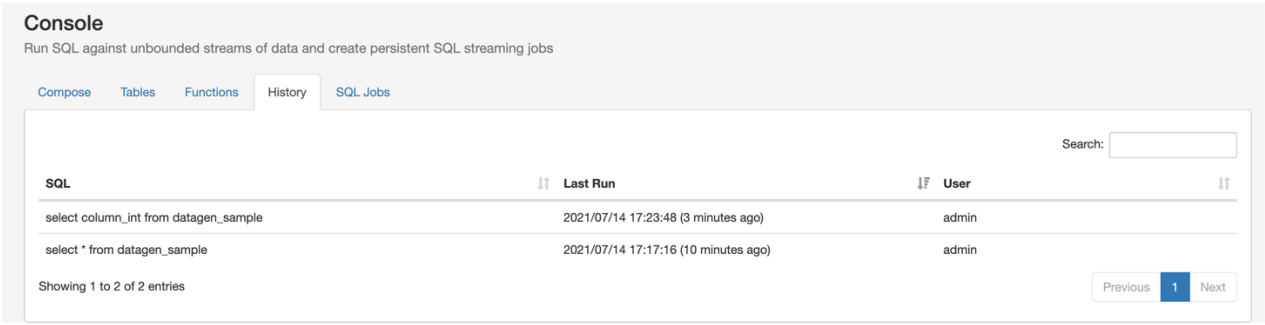


6. Select Log tab on the bottom panel to review the log information of the job submission.



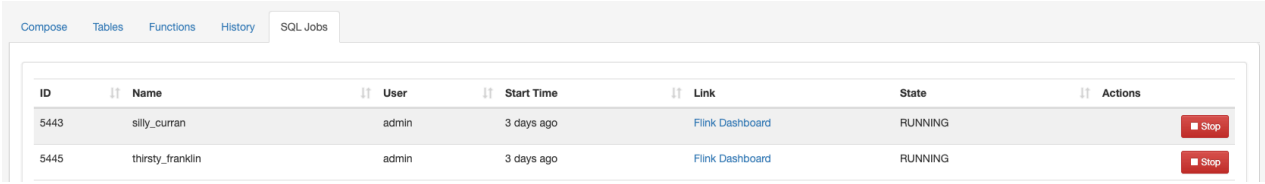
History of SQL queries

You can review and reuse the SQL queries that were previosuly executed on the **History** page. When you click on one of the SQL queries, it is automatically imported to the SQL window for execution. You can filter the SQL queries by the time they were last run or by the user who run them. You can also search for a type of SQL query using the Search field.



Using the Flink Dashboard

You can also monitor your running SQL jobs using the Flink Dashboard. You can easily reach the Flink Dashboard on the SQL jobs tab below the Console main menu. After clicking on the Flink Dashboard, you are redirected to the Flink Dashboard interface.



SQL Syntax Guide

The SQL Syntax Guide provides a collection of SQL syntaxes that is supported in SQL Stream Builder.

SQL Syntax

- Filtering WHERE
- Projections FOO || 'baz'
- Tumble, Hopping window expressions TUMBLE(), HOP()
- Grouping GROUP BY
- Filter after aggregation HAVING
- Join JOIN ON..
- Comparison operators
- Logical operators
- Arithmetic operators and functions
- Character string operators and functions
- Binary string operators and functions
- Date/Time functions
- Type conversion
- Aggregate functions
- Grouped window functions
- Grouped Auxiliary functions

Data types

- Datatypes

Metadata commands

- show tables - list virtual tables.
- desc <vtable> - describe the specified virtual table, showing columns and types.
- show jobs - list current running SQL jobs.
- show history - show SQL query history (only successfully parsed/executed).
- help - show help.