

## SQL Stream Job Lifecycle

Date published: 2019-12-17

Date modified: 2022-05-05



# 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

<b>Creating and naming SQL jobs.....</b>	<b>4</b>
<b>Running SQL Stream jobs.....</b>	<b>5</b>
<b>Configuring SQL job settings.....</b>	<b>7</b>
Adjusting logging configuration in Advanced Settings.....	10
Configuring YARN queue for SQL jobs.....	11
<b>Managing SQL jobs and queries.....</b>	<b>12</b>
Loading SQL queries to SQL Editor.....	12
Loading jobs to SQL Editor.....	13
Stopping SQL jobs.....	14
Restarting SQL jobs.....	15
Editing SQL jobs.....	17
Deleting SQL jobs.....	17
<b>Sampling data for a running job.....</b>	<b>18</b>
Configuring sampling sink.....	19
<b>Managing session for SQL jobs.....</b>	<b>21</b>
Executing SQL jobs in production mode.....	22
<b>Enhancing the developer experience.....</b>	<b>23</b>

## Creating and naming SQL jobs

You need to create or select an already existing SQL job on the Streaming SQL Console to be able to submit SQL queries.

The landing page of Streaming SQL Console is the **Getting Started** window. You can use the **Getting Started** window to quickly create a new job or select an already existing job that you can reload to the Console page.

You have the following options to name your SQL job:

- Use the predefined name in the Job Name field when opening the Streaming SQL Console
- Use the Generate Job Name button to generate a new random name for your job
- Manually provide the name of the job to the Job Name field

You can also select a previously created job from the list, and the job details will be loaded to the **SQL Editor** on the **Console** page.

On the Console page of Streaming SQL Console, you can create a job using the + New Job button.

After a new job is created, you can save the job and its settings using the Save button. When you make a change to the SQL query, job settings or create a Materialized View, you have the option to save the changes with the job. You can also revert your changes to the latest saved version of the job using the Revert button.

You can also see the status of the job next to the jobs name on the panel. The following statuses are indicated when managing jobs:

- **Unsaved job** - new job creation is in progress, the job hasn't been saved yet or has unsaved changes
- **Executing** - the submitted query is executed
- **Running** - the query is executed, and the job is successfully running

- **Stopped** - due to some errors the job creation failed, the SQL job has been stopped

## Running SQL Stream jobs

Every time you run an SQL statement in the SQL Stream console, it becomes a job and runs on the deployment as a Flink job. You can manage the running jobs using the Jobs tab on the UI.

### About this task

There are two logical phases to run a job:

1. **Parse:** The SQL is parsed and checked for validity and then compared against the virtual table schema(s) for correct typing and key/columns.
2. **Execution:** If the parse phase is successful, a job is dynamically created, and runs on an open slot on your cluster. The job is a valid Flink job.

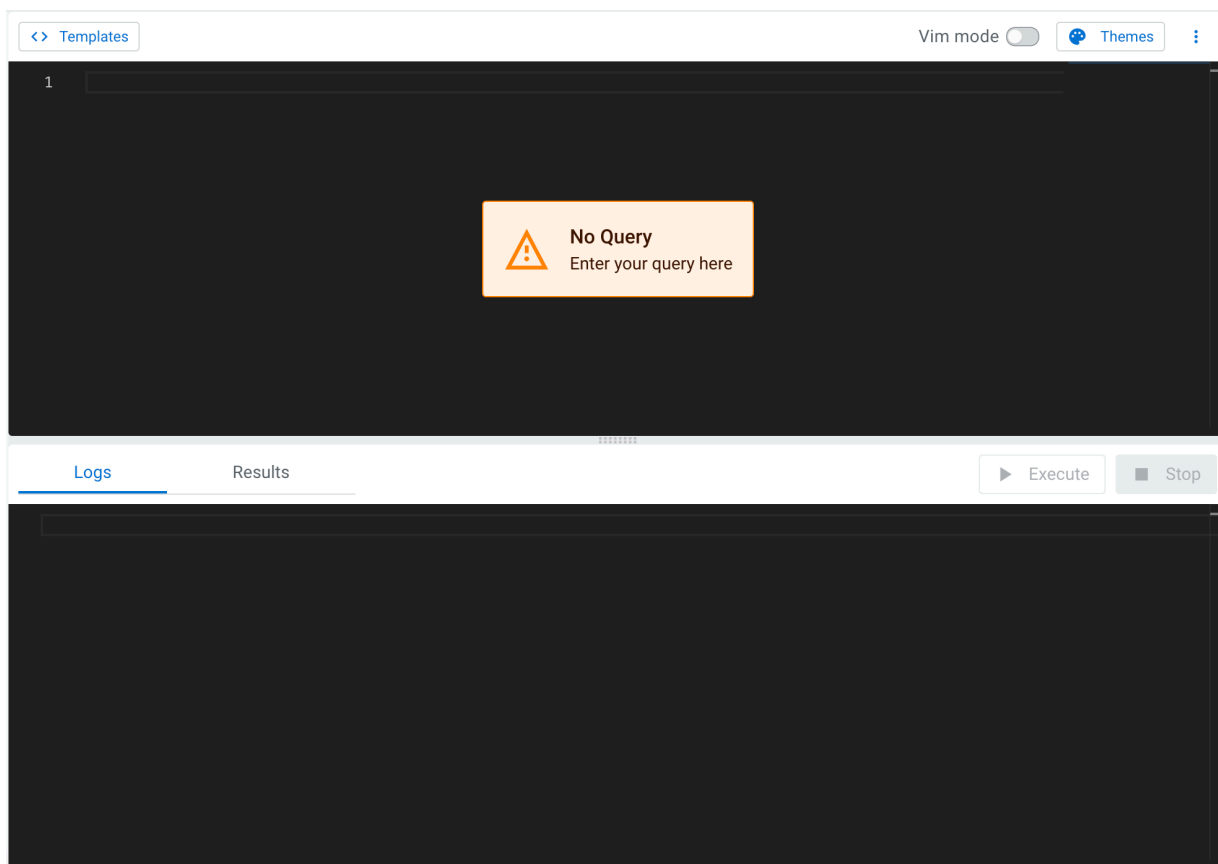
### Before you begin

- Make sure that you have registered a Data Provider if you use the Kafka service on your cluster.
- Make sure that you have added Kudu, Hive or Schema Registry as a catalog if you use them for your SQL job.

### Procedure

1. Navigate to the Streaming SQL Console.
  - a) Go to your cluster in Cloudera Manager.
  - b) Select SQL Stream Builder from the list of services.
  - c) Click SQLStreamBuilder Console .The **Streaming SQL Console** opens in a new window.
2. Provide a name for the SQL job.
  - a) Optionally, you can click Generate Random Name to generate a name for the SQL job.
3. Create a table using one of the following options:
  - Using the Add Kafka or Webhook table wizard.
  - Using the Templates.
  - Add your custom CREATE TABLE statement to the SQL window.

#### 4. Add a SQL query to the SQL Editor.

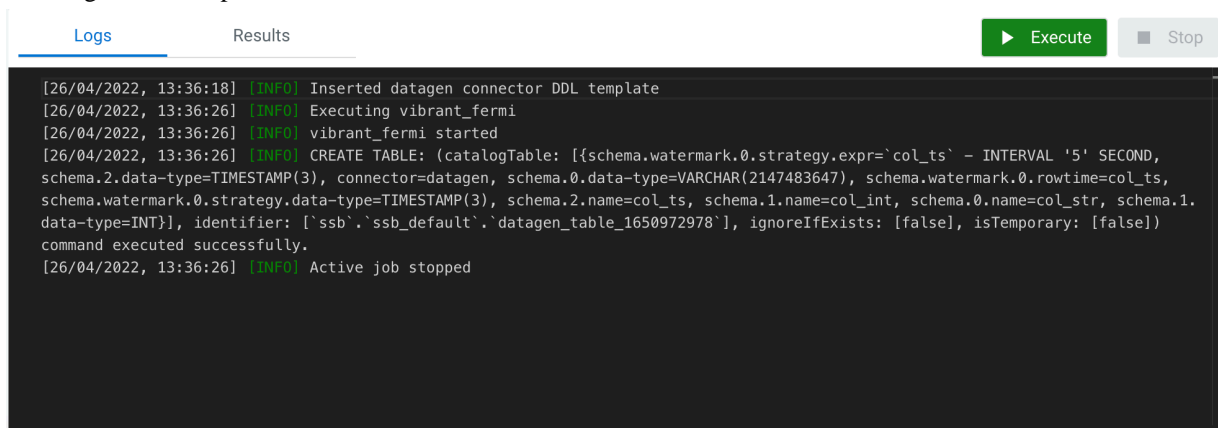


**Note:** You cannot start a job without adding a SQL statement in the SQL editor window. In case, there are no SQL statements provided and you click Execute, the following error message is displayed: You must provide a SQL query.

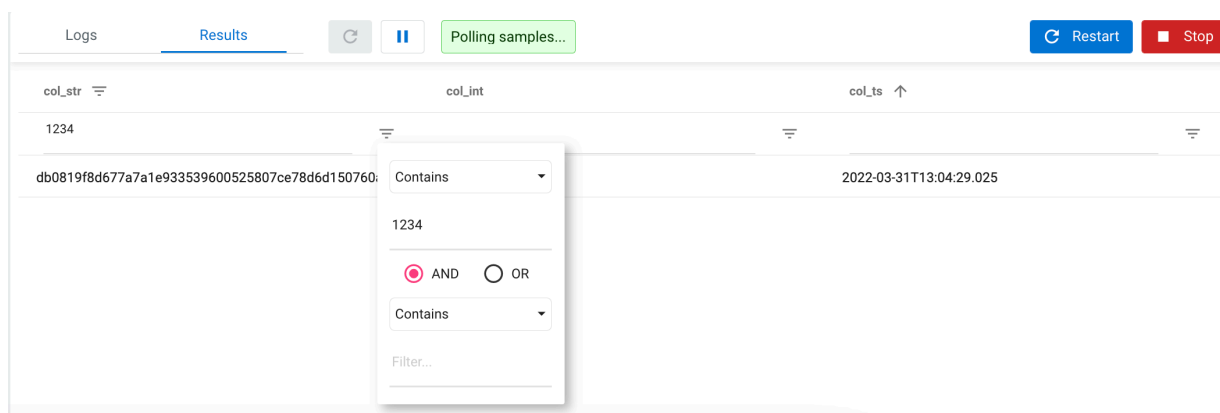
When starting a job, the number of slots consumed on the specified cluster is equal to the parallelism setting. The default is one slot. To change the parallelism setting and more job related configurations, click Job Settings.

#### 5. Click Execute.

The Logs window updates the status of SSB.



6. Click Results to check the sampled data.



You can pause or restart the sampling of a SQL job with the Pause and Refresh buttons. The status of the polling can also be seen next to the Pause button. The Results tab also allows you to change the order of the samples, and to filter the samples using different types of conditions that you can also combine. For changing the order, click on the name of the column, the direction of the ordering is indicated next to the column name by arrows.



For filtering, you can either click the button next to the column name or under the row of the column name. The filtering value can be changed by typing the values directly to the filtering row, the conditions need to be changed using the filtering arrow.

### Results

A job is generated that runs the SQL continuously on the stream of data from a table, and pushes the results to a table, to the Results tab or to a Materialized View.

### Related Information

[Registering Data Providers in SSB](#)

[Creating Tables for SQL Stream jobs](#)

[Monitoring SQL Stream jobs](#)



## 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, exactly once processing and restoring from savepoint.

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

## General and sample settings

## Job Settings

 Settings  Advanced

### General

Job Paralellism (threads)

1

Restore From Savepoint ☐

### Sample

Sample Count ⓘ

100

Sample Window Size ⓘ

100

Sample Behavior

Sample one message every second ▼

**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 provided 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.

**Checkpoint settings****Checkpoint**

Enable Checkpointing ☒

Checkpoint Mode

At Least Once

Checkpoint Interval (ms) ⓘ

600000

Checkpoint Timeout (ms) ⓘ

60000

Tolerable Checkpoint Failures ⓘ

2

Failure Restart Strategy

Enable Auto Recovery

**Enable Checkpointing**

You can enable and disable checkpointing for a SQL job. By default the checkpointing is enabled.

**Checkpoint Mode**

Switching between checkpointing modes. You can choose between At Least Once or Exactly Once.

**Checkpoint Interval**

The time in milliseconds between checkpointing attempts.

**Checkpoint Timeout**

The maximum time in milliseconds until a checkpointing attempt is timed out.

**Tolerable Checkpoint Failures**

The number of checkpointing attempts until the job is aborted.

**Failure Restart Strategy**

Switching between restarting strategies when a job fails. You can choose between enabling and disabling auto recovery. By default auto recovery is enabled for checkpointing.

## Adjusting logging configuration in Advanced Settings

You can customize the logging configurations for the SQL Stream Builder (SSB) job on the Streaming SQL Console in per-job mode or session mode. Adjusting the log configuration enables you to control the log levels of all the underlying libraries: Flink, Hadoop, Kafka, Zookeeper, other common libraries, and connectors to get more or less information in your job's log.

**About this task**

The customization of the log configuration works differently based on the job deployment mode:

**Session mode**

The `execution.target` is set to *yarn-session* mode, this is the default execution mode.

The log configuration is set at the start time if the Flink YARN session is applied to every job execution. For example, the current log configuration is applied if and only if the Flink YARN session is not set on the Session tab of the Compose page.



**Note:** The Reset Session button only resets the SSB Session, not the underlying Flink YARN session. To do that, you have to kill the YARN application that is indicated under Flink Yarn Session on the Session tab.

**Per-job mode**

The `execution.target` is set to *yarn-per-job* mode.

When you change the default execution mode to per-job, the currently applied log configuration is going to be used for the job. To configure the execution mode, you need to start the SQL query with the following line:

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

**Procedure**

1. Click Create Job or select a previous job on the **Getting Started** page.  
You are redirected to the **Console** page.
2. Select Job Settings from the Compose page.

3. Click Advanced.

Job Settings

General **Advanced**

**Custom log configuration** ⚠ Log configuration won't be persisted

```

rootLogger.level = INFO
rootLogger.appenderRef.file.ref = MainAppender
#Uncomment this if you want to _only_ change Flink's logging
#logger.flink.name = org.apache.flink
#logger.flink.level = INFO

# The following lines keep the log level of common libraries/connectors on
# log level INFO. The root logger does not override this. You have to manually
# change the log levels here.
logger.akka.name = akka
logger.akka.level = INFO
logger.kafka.name = org.apache.kafka
logger.kafka.level = INFO
logger.hadoop.name = org.apache.hadoop
logger.hadoop.level = INFO
logger.zookeeper.name = org.apache.zookeeper
logger.zookeeper.level = INFO

# Log all infos in the given file
appender.main.name = MainAppender
appender.main.type = File
appender.main.append = false
appender.main.fileName = ${sys:log.file}
appender.main.layout.type = PatternLayout
appender.main.layout.pattern = %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p %-60c %x - %m%n

# Suppress the irrelevant (wrong) warnings from the Netty channel handler
logger.netty.name = org.apache.flink.shaded.akka.org.jboss.netty.channel.DefaultChannelPipeline
logger.netty.level = OFF
  
```

4. Modify the settings based on your requirements.
5. Close the **Job Settings** window.
6. Click Save to save the job settings.
7. Add and execute a SQL statement.
8. Click **SQL Jobs**.
9. Search for the job you have executed previously.
10. Click **Flink Dashboard**.

The **Flink Dashboard** opens in a new window.

11. Click Task Managers > Logs .

The log information appears in the log window based on your custom configurations.

## Configuring YARN queue for SQL jobs

You can configure the YARN application queue with a custom value for a SQL job using the Streaming SQL Console.

With YARN queues, you can deploy applications to a specific subset of nodes with separate or limited resources, according to configuration. This enables you to have a separate execution environment with limited resources to experiment with new applications without impacting any production operation.

The YARN application queue can only be configured in a per-job execution target using the production mode of SQL Stream Builder. This means that the default value of the YARN queue does not change, and you can only customize it to the specific job that is executed in the production mode.

The default value of YARN application queue can be configured in Cloudera Manager. After accessing the SQL Stream Builder service on your cluster, you need to open the configurations, and search for YARN application queue where you can change the default value. This default value is overwritten when using the SET statement in production mode for specifying the YARN queue for a SQL job.

For more information about running jobs in production mode, see the [Executing SQL jobs in production mode](#) section.

You can configure the YARN application queue using the SET statement in Streaming SQL Console as the following code example shows:

```
--PROD
set 'execution.target' = 'yarn-per-job';
set 'yarn.application.queue' = 'my-queue';
select * from datagen_table_1648801198
```

## Managing SQL jobs and queries

You can restart a SQL Stream job after stopping it. After stopping a job, you can update or change the configurations. You can navigate through the job life cycle using the Streaming SQL Console. To manage your SQL jobs and queries, you can either use the different windows on the Console page or directly access the list of jobs and queries on the SQL Jobs and History pages.

### Loading SQL queries to SQL Editor

You can reuse an already submitted SQL query from the Console page or the History page of Streaming SQL Console.

When you need to reuse a SQL query for a new job, you can load it and execute it in the SQL Editor. You can use the Query History button on the **Console** page or you can select the **History** page from the main menu of Streaming SQL Console.

**For Console page**

### Query History

`SELECT * from datagen_table_1651486213`
2022-05-04 10:43:24 (22 hours ago)

SUCCESS

`CREATE TEMPORARY TABLE heros (  
 id INT,  
 power STRING,  
 aae INT`
2022-05-02 10:32:14 (3 days ago)

SUCCESS

`CREATE TABLE heros (  
 name STRING,  
 power STRING,  
 aae INT`
2022-05-02 10:31:12 (3 days ago)

SUCCESS

`CREATE TABLE `datagen_table_1651486771` (  
 id INTEGER,  
 power VARCHAR,  
 aae INTEGER`
2022-05-02 10:20:32 (3 days ago)

SUCCESS

`SELECT * FROM datagen_table_1651486213`
2022-05-02 10:11:00 (3 days ago)

SUCCESS

`CREATE TABLE `datagen_table_1651486213` (  
 col_str STRING,  
 col_int INT,  
 col_ts TIMESTAMP(3).`
2022-05-02 10:10:15 (3 days ago)

SUCCESS

### For History page

- > Console
- f() Functions
- History**
- SQL Jobs
- Materialized Views
- Data Providers
- Connectors

#### SQL Query History

Reload

Status	SQL	User	ID	Created	Updated	
SUCCESS	SELECT * from datagen_table_1651486213	admin	3061		2022-05-04 10:43:24 (23 hours ago)	
SUCCESS	CREATE TEMPORARY TABLE heros (	admin	3060	2022-05-02 10:32:14 (3 days ago)	2022-05-02 10:32:14 (3 days ago)	
SUCCESS	CREATE TABLE heros (	admin	3058	2022-05-02 10:31:12 (3 days ago)	2022-05-02 10:31:12 (3 days ago)	
SUCCESS	CREATE TABLE `datagen_table_1651486771` (	admin	3055	2022-05-02 10:20:32 (3 days ago)	2022-05-02 10:20:32 (3 days ago)	
SUCCESS	SELECT * FROM datagen_table_1651486213	admin	3054	2022-05-02 10:11:00 (3 days ago)	2022-05-02 10:11:00 (3 days ago)	
SUCCESS	CREATE TABLE `datagen_table_1651486213` (	admin	3053	2022-05-02 10:10:15 (3 days ago)	2022-05-02 10:10:15 (3 days ago)	


You can view the DDL of the submitted query by hovering over the items. You can load the query to the editor from the **Console** page by clicking on the query in the **Query History** window. From the **History** page, you need to click on the query, and select Paste to Active Job.





## Loading jobs to SQL Editor

You can reload the already executed jobs regardless of success to the SQL Editor to make further and additional changes, and execute them again.

You can load the running and stopped jobs to the **SQL Editor** in case you need to stop them, restart them or change their configurations. You have the following options to load the SQL jobs to the **SQL Editor** on the **Console** page:

- On the **Console** page, click on SQL Jobs button, and click on a specific job from the list.
- 

On the **SQL Jobs** page, click on the  button next to the job status, and select Load Job.

<b>RUNNING</b>	 Execute	ted_einstein	5199
<b>STOPPED</b>	 Load Job	cious_nobel	5198
<b>STOPPED</b>	 Delete Job	ective_hermann	5197
<b>STOPPED</b> 		trusting_hopper	5200

The details of the job are reloaded to the SQL Editor with the latest saved configuration and Materialized Views.

### Stopping SQL jobs


As a SQL Stream job processes streaming data, you need to stop the job to finish the process, and not let the job run for an unlimited time. You can stop a running job either on the Console page or the SQL Jobs page.

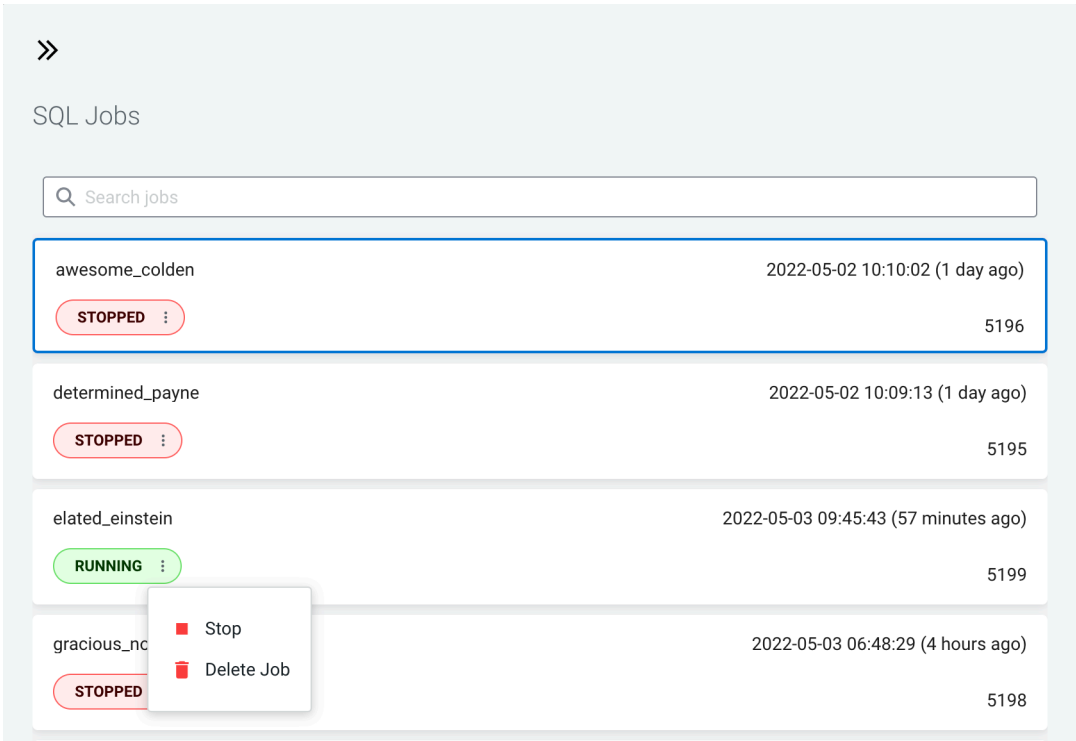
#### Stopping job from Console page

If the job is currently loaded to the **SQL Editor**, you need to click on the Stop button to terminate the job.




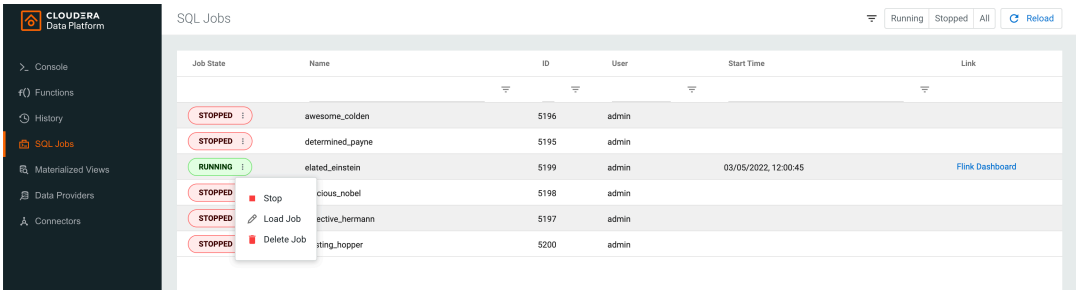
If the job is not loaded to the **SQL Editor**, you need to click on the SQL Jobs button to open the list

of SQL jobs window. Without selecting the running job, click on the  button next to the job status, and select Stop.





Stopping job from SQL Jobs page

On the **SQL Jobs** page, you need to click on the  button next to the job status, and select **Stop**.



You can filter down the list of job only to the running jobs by selecting Running from the filter. For changing the order, click on the name of the column, the direction of the ordering is indicated

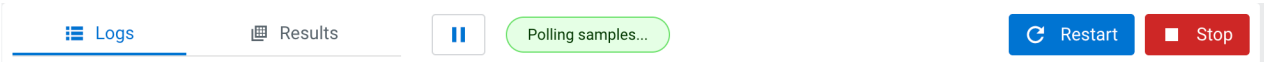
 next to the column name by arrows. For further filtering, you can either click the  button next to the column name or under the row of the column name. The filtering value can be changed by typing the values directly to the filtering row, the conditions need to be changed using the filtering arrow.

Restarting SQL jobs

You can restart an already running job when it is active in the SQL Editor. You can also restart a stopped job by executing it again either from the Console page or the SQL Jobs page.

Restarting a running SQL job


If the job is currently loaded to the **SQL Editor**, you need to click on the Restart button to restart the job.

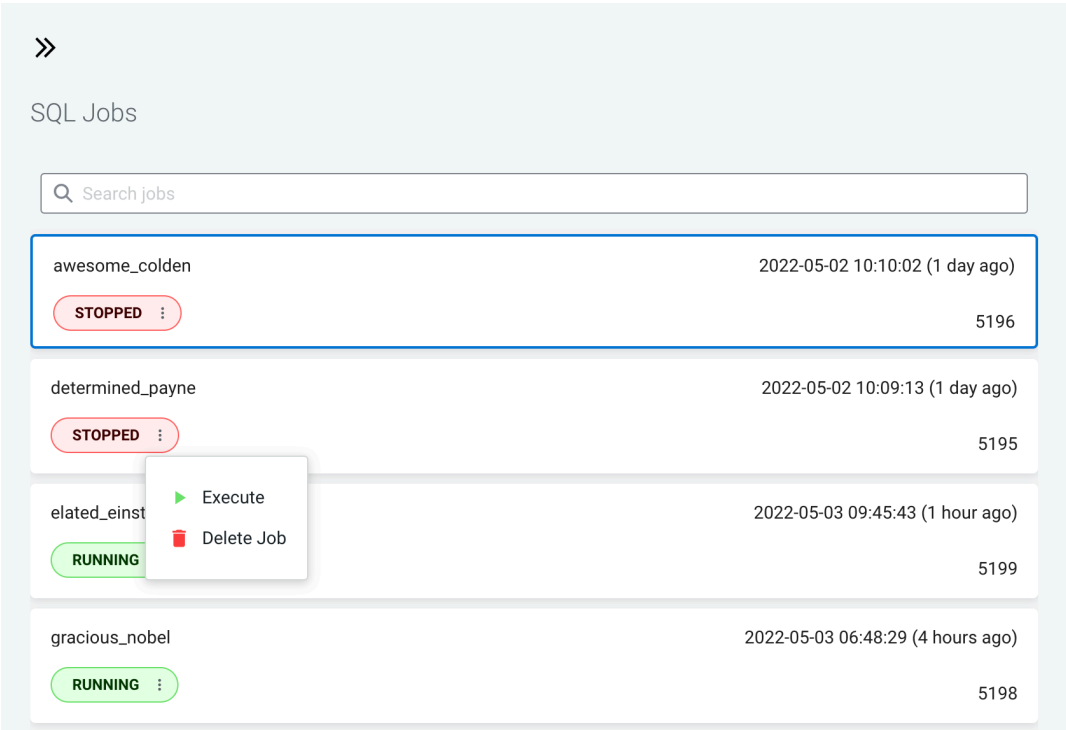


Executing a stopped SQL job


Executing stopped job from Console page

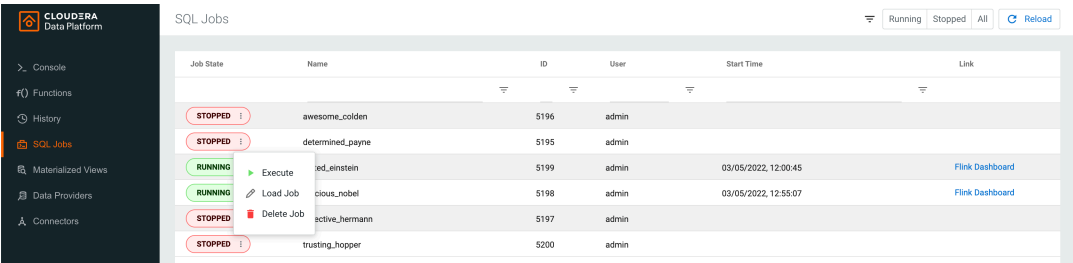
You need to click on the SQL Jobs button to open the list of SQL jobs window. Without selecting

the stopped job, click on the  button next to the job status, and select Execute.



Executing job from SQL Jobs page

On the **SQL Jobs** page, you need to click on the  button next to the job status, and select Execute.




You can filter down the list of job only to the stopped jobs by selecting Stopped from the filters.



For changing the order, click on the name of the column, the direction of the ordering is indicated



next to the column name by arrows. For further filtering, you can either click the  button next to the column name or under the row of the column name. The filtering value can be changed by typing the values directly to the filtering row, the conditions need to be changed using the filtering arrow.



**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 SQL jobs

You can edit every detail of your SQL jobs after stopping and reloading it to the SQL Editor on the Console page.

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

1. Load the stopped job to the **SQL Editor** either from the **Console** or **SQL Jobs** page.
2. Click Job Settings to edit any configuration of the selected SQL job.

You can modify the job settings, Materialized Views and SQL query when editing a job.

3. Click Save.
4. Click Execute.



**Note:** Before saving a job, you can revert any type of job related change using the Revert Changes button.

## Deleting SQL jobs

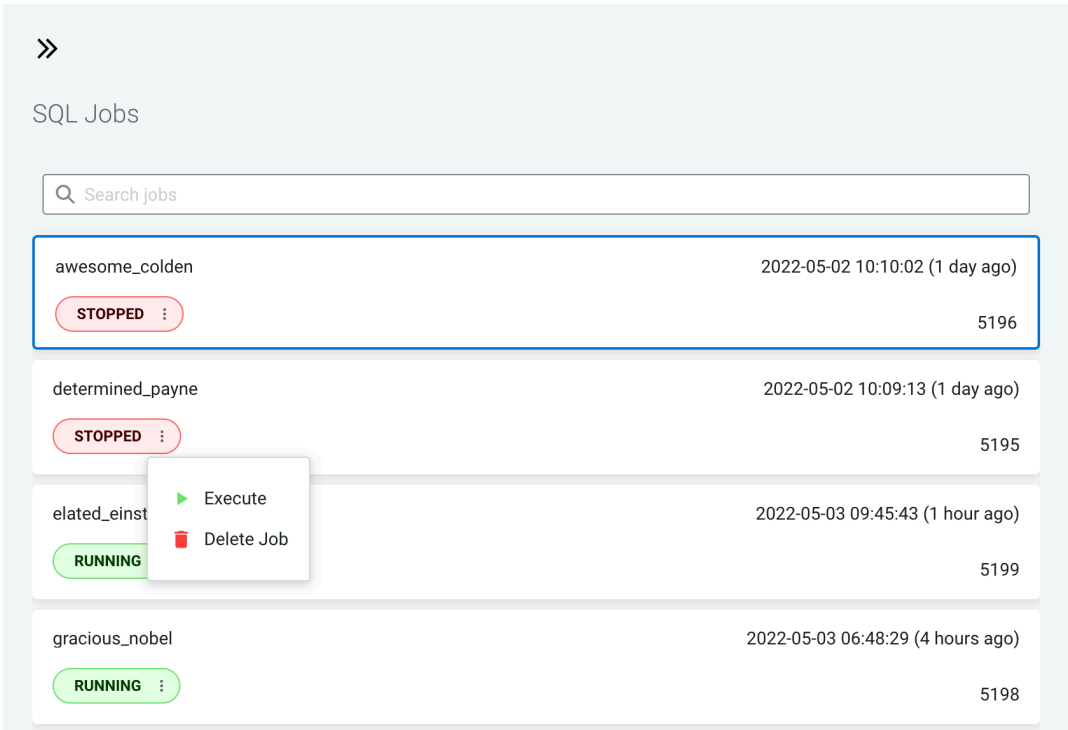
You can delete running and stopped jobs from the Console and SQL Jobs page. The deleted job no longer appears on the Streaming SQL Console.

### Deleting job from Console page


You need to click on the SQL Jobs button to open the list of SQL jobs window. Without selecting

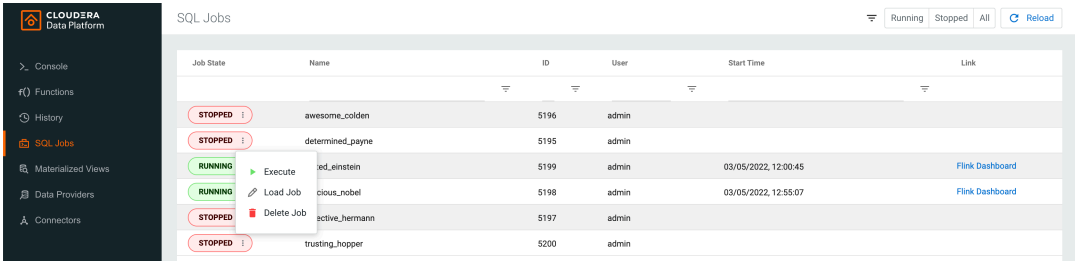


the stopped job, click on the  button next to the job status, and select Delete.





Deleting job from SQL Jobs page

On the **SQL Jobs** page, you need to click on the  button next to the job status, and select Delete Job.



You can filter down the list of jobs by selecting Stopped or Running from the filters. For changing the order, click on the name of the column, the direction of the ordering is indicated

 next to the column name by arrows. For further filtering, you can either click the  button next to the column name or under the row of the column name. The filtering value can be changed by typing the values directly to the filtering row, the conditions need to be changed using the filtering arrow.

# Sampling data for a running job

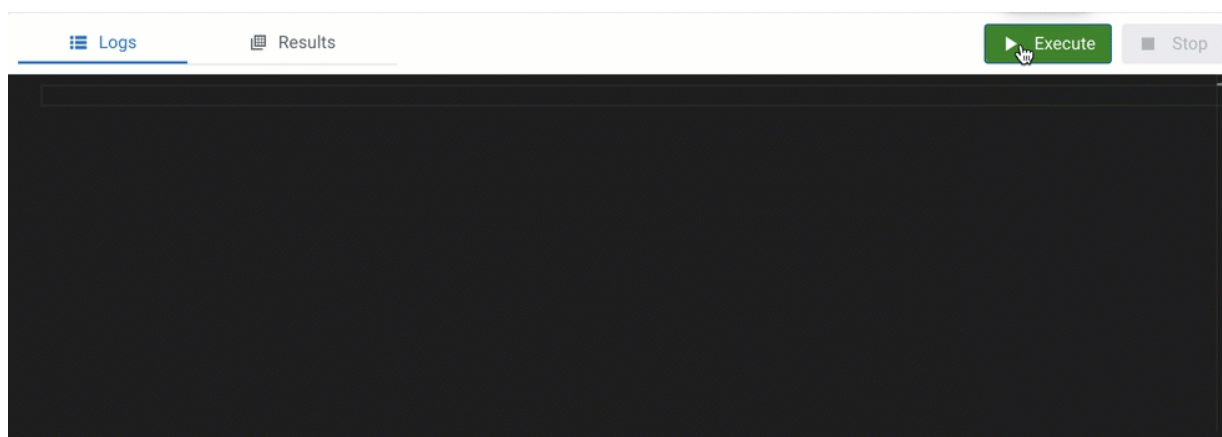
You can sample data from a running job to the Streaming SQL Console without writing it to any sink to inspect the data and to make sure the job is running correctly, and producing the results you expect.

## 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 under the **SQL Editor** on the **Console** page. Even if you do not specify any sink to the SQL job, the results automatically appear in the **Results** tab.

## Procedure

1. Navigate to the Streaming SQL Console.
  - a) Go to your cluster in Cloudera Manager.
  - b) Select SQL Stream Builder from the list of services.
  - c) Click SQLStreamBuilder Console .The **Streaming SQL Console** opens in a new window.
2. Create a new job or select an existing job on the **Getting Started** window.
3. Execute the job.



The samples are automatically polled to the Console under the **Results** tab until the samples reach the maximum number of result or you stop the job.

## Results

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

## Configuring sampling sink

The `sample.sink.parameters` configuration allows you to configure the sink used to retrieve sample results of a SQL query.

The parameter introduces key/value pairs which the system can use to configure the sink.

The current implementation uses a Kafka sink as the physical medium for storing the sample results, which means that you can use the configuration parameters of Kafka producers to configure the sink. You can configure any Kafka producer parameter based on your requirements.

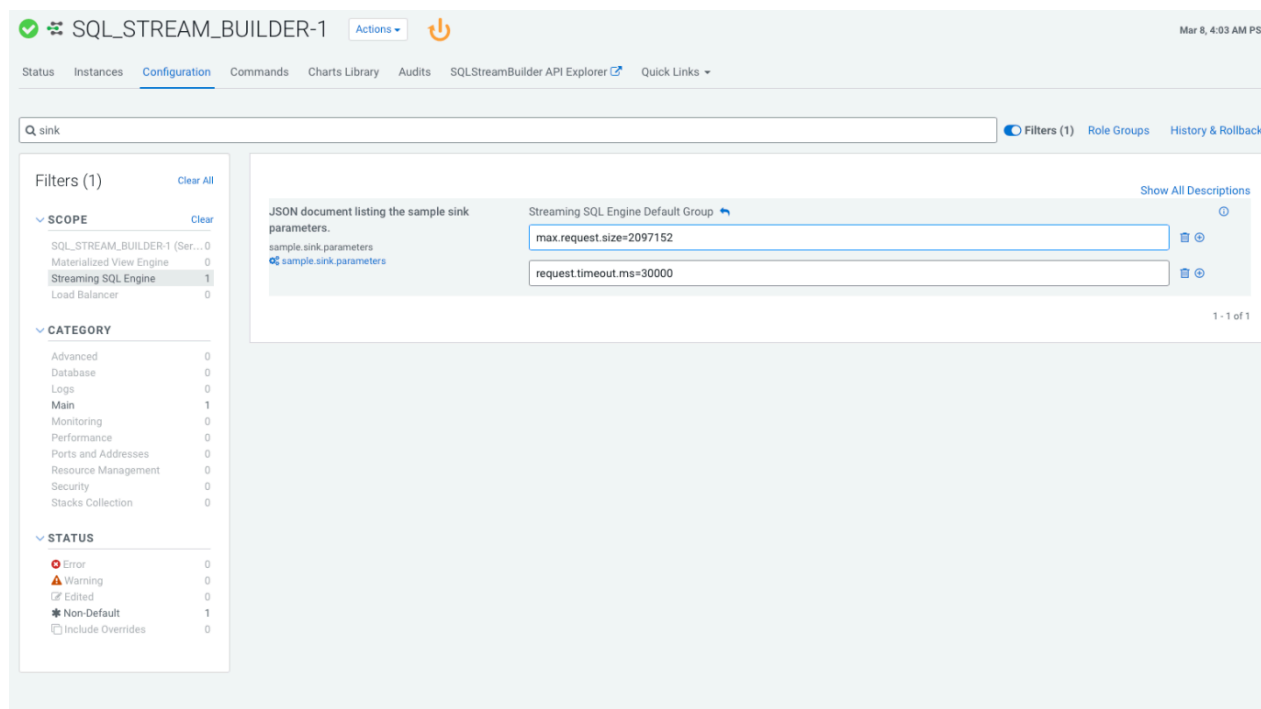
For more information about Kafka producer configurations, see the [Apache Kafka documentation](#).

As an example, the following parameters are configured for the sampling sink:

- `max.request.size`
- `request.timeout.ms`

The sample sink configuration in the example can be used when there is a need to increase the amount of data allowed to be sent with the producer to the Kafka sink. For example, the default configuration of the amount of data is set to 1M (1048576), which needs to be increased as more result records are expected. In this case, you need to set the

sample sink configuration to a bigger amount in Cloudera Manager, for example 2M (2097152), as shown in the following image:



### Configuring sampling parameter in Cloudera Manager

1. Go to your cluster in Cloudera Manager.
2. Select SQL Stream Builder from the list of services.
3. Select Configuration.
4. Search for sink using the search bar.

The sample.sink.parameters configuration is listed.

5. Click the plus icon to add the Producer configuration.
6. Add parameters to the sampling configuration.

For example, max.request.size and request.timeout.ms parameter.

7. Set the value based on your requirements.
8. Click Save.

### Sample sink configuration per job

When you configure the sample.sink.parameters in Cloudera Manager for SQL Stream Builder (SSB), the configuration is implemented on every sampling job no matter which user submits the SQL query. However, you can choose to configure the session to contain the specific sample.sink.parameter that only applies to the jobs submitted within the session.

To configure the sample.sink.parameters for a session, you need to use the SET commands in the SQL window on the Streaming SQL Console. You need to use the “sample.sink” prefix before the parameters to indicate that the setting is applicable only for the sample sink. For example to increase the size of records to the Producer of the Kafka sample sink, you need to add the Kafka parameter name after the “sample.sink” prefix and add a needed value:

```
SET sample.sink.max.request.size=2097152
```

This means that the command sets the max.request.size parameter to 2097152 for the current session only.

## 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.

Session	
<input type="text" value="Search"/> <span>Reset Session</span>	
<a href="#">General</a> <a href="#">Properties</a>	
Default Catalog	ssb
Default Database	ssb_default
Flink Yarn Session	
cluster_id	application_1651070259224_0001
tracking_url	http://docs-test-1.docs-test.root.hwx.site:8088/proxy/application_1651070259224_0001/
Session ID	3ec127be-b921-4cee-b5c4-329e6a9e2541
Session Start	2022-04-27 14:53:14 (2 days ago)
Temporary Tables	
Temporary Views	
Username	admin

1. Navigate to the **Streaming SQL Console**.
  - a. Go to your cluster in Cloudera Manager.
  - b. Click on SQL Stream Builder from the list of Services.
  - c. Click on the SQLStreamBuilder Console.

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

2. Click Create Job or select a previous job on the Getting Started page. You are redirected to the Console page.  
You are redirected to the **Console** page.
3. Select Session.
4. 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. Go to your cluster in Cloudera Manager.
  - b. Click on SQL Stream Builder from the list of Services.
  - c. Click on the SQLStreamBuilder Console.

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

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

Example:

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

You can review all of the configurable parameters on the Properties tab of the **Session** window.

Property	Value
atlas.collection.enabled	false
execution.attached	true
execution.buffer-timeout	100
execution.checkpointing.externalized-checkpoint-retention	RETAIN_ON_CANCELLATION
execution.checkpointing.max-concurrent-checkpoints	1
execution.checkpointing.min-pause	0
execution.checkpointing.mode	EXACTLY_ONCE
execution.checkpointing.snapshot-compression	false
execution.checkpointing.timeout	60000
execution.runtime-mode	STREAMING
execution.target	yarn-session
high-availability	ZOOKEEPER
high-availability.storageDir	hdfs:///user/flink/ha
high-availability.zookeeper.client.acl	open
high-availability.zookeeper.quorum	docs-test-1.docs-test.root.hwx.site:2181
historyserver.cli.fallback	true
historyserver.cluster.fetcher	YARN
jobmanager.archive.fs.dir	hdfs:///user/flink/applicationHistory
jobmanager.archive.per-user	true
jobmanager.heap.size	1073741824
kerberos.auth.enabled	false
parallelism.default	1
pipeline.auto-watermark-interval	200
pipeline.generic-types	true

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. Go to your cluster in Cloudera Manager.
- b. Click on SQL Stream Builder from the list of Services.
- c. Click on the SQLStreamBuilder Console.

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

2. Click Create Job or select a previous job on the **Getting Started** page.

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

3. Add `--PROD` to the **SQL Editor**.

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.

## Enhancing the developer experience

You can change the views of the Console page to have a better developer experience. The SQL Editor also has some SQL developer features that makes searching code and writing SQL queries easier.

The following features are available for Streaming SQL Console to enhance the developer experience of SSB:

### VIM mode

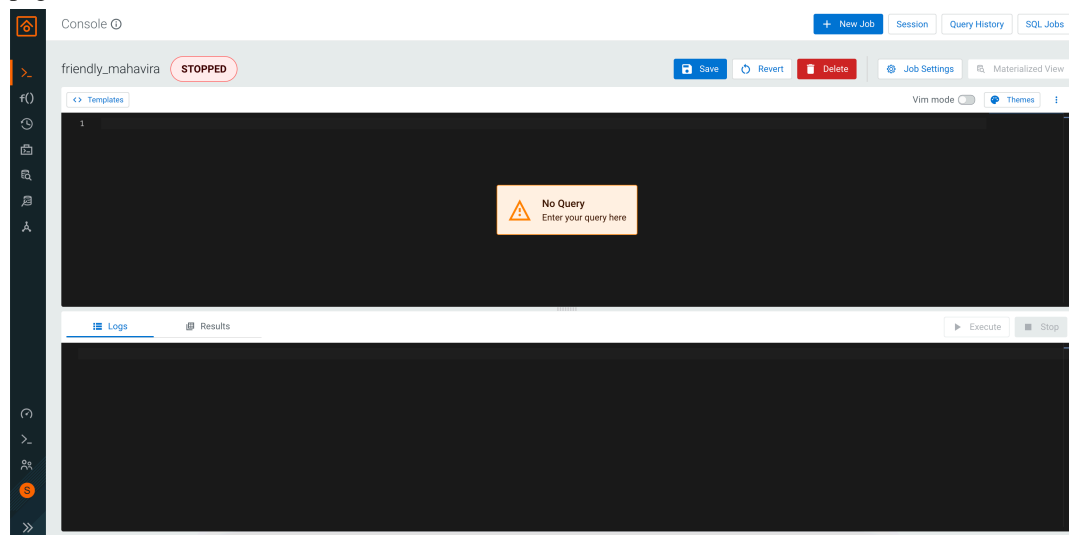
You can edit the SQL statements and queries either in default or in VIM mode. To use the VIM editor, you need to enable it with Vim mode toggle. For more information about the VIM editor, see the [official VIM documentation](#).

### Using Themes

You can customize the color coding of the SQL Editor by selecting one of the supported themes.

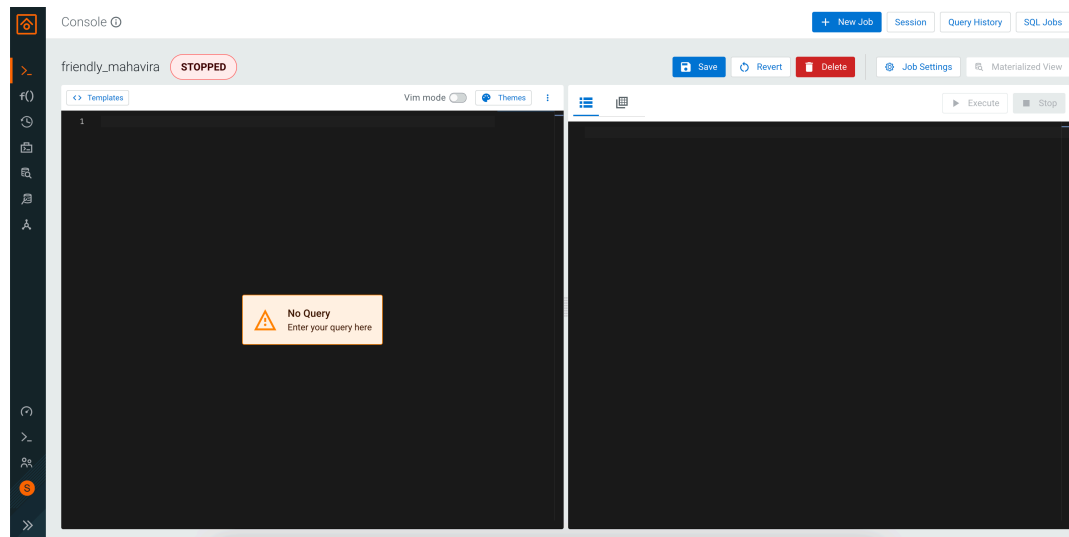
### Hide Tables and main menu

You can close the Tables panel next to the SQL Editor to create a bigger Editor window for your SQL queries. Additionally, the left main menu can also be closed to have the SQL Editor in full page size.



### Switch to Vertical Layout

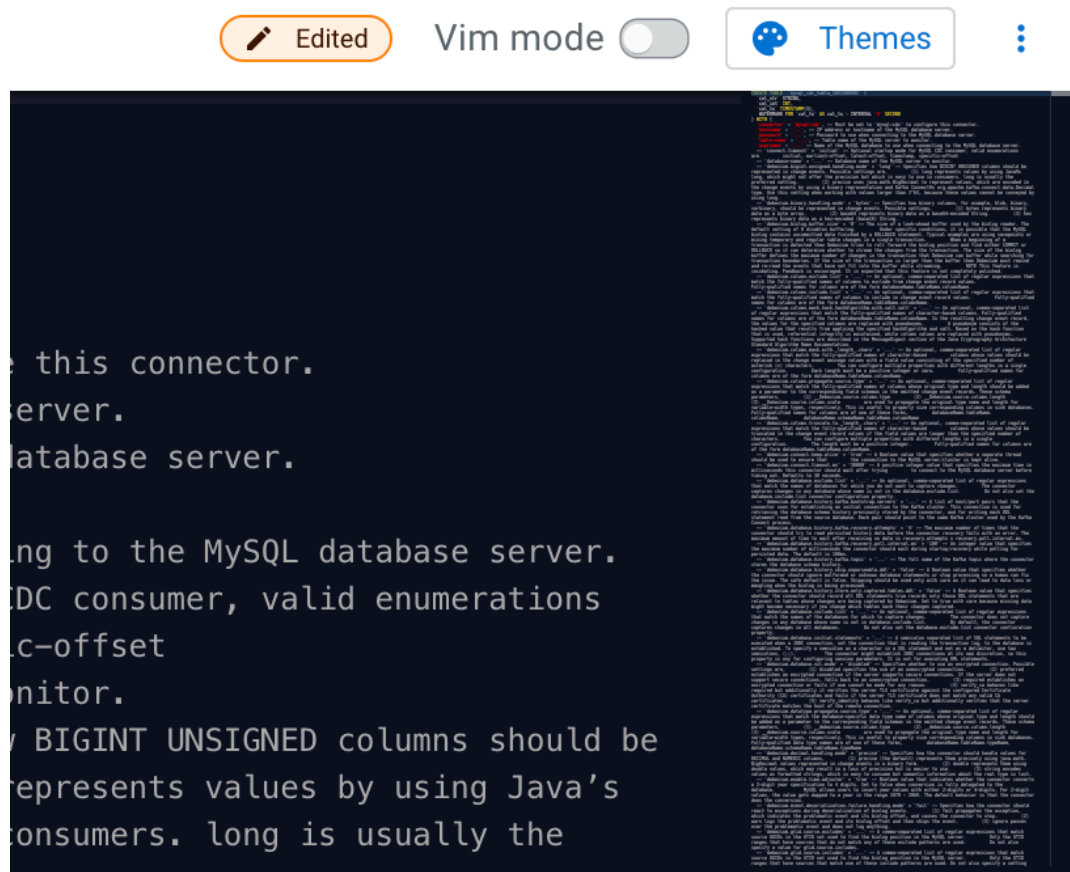
You can switch to a vertical layout of the SQL Editor which means that the Editor window and the Logs/Results window are next to each other.



### Sidebar of SQL Editor

You can see the miniature version of the SQL code at the sidebar of the SQL Editor window. This is helpful when you have longer SQL syntaxes in a batch, and need to quickly jump in the middle or at the end of the code. In addition, the sidebar also shows the appearance of specific values in the code. For example, if you want to see where a given value appears, you can click in the SQL Editor at the given value, and its occurrence is highlighted in the sidebar.





### Autocomplete for SQL

When you start typing SQL statements to the SQL Editor, an autocomplete window appears to help you with the SQL syntax. You can select any statement from the list by using the enter key on your keyboard.

