

Quickstart: Query data in Amazon S3

Users can analyze data stored in Amazon S3 by running SQL queries from JupyterLab notebooks using the SQL extension. The extension integrates with Athena enabling the functionality for data in Amazon S3 with a few extra steps.

This section walks you through the steps to load data from Amazon S3 into Athena and then query that data from JupyterLab using the SQL extension. You will create an Athena data source and AWS Glue crawler to index your Amazon S3 data, configure the proper IAM permissions to enable JupyterLab access to Athena, and connect JupyterLab to Athena to query the data. Following those few steps, you will be able to analyze Amazon S3 data using the SQL extension in JupyterLab notebooks.

Prerequisites

- Sign in to the AWS Management Console using an AWS Identity and Access Management (IAM) user account with admin permissions. For information on how to sign up for an AWS account and create a user with administrative access, see [the section called “Complete Amazon SageMaker AI prerequisites”](#).
- Have a SageMaker AI domain and user profile to access SageMaker Studio. For information on how to set a SageMaker AI environment, see [the section called “Use quick setup”](#).
- Have an Amazon S3 bucket and folder to store Athena query results, using the same AWS Region and account as your SageMaker AI environment. For information on how to create a bucket in Amazon S3, see [Creating a bucket](#) in the Amazon S3 documentation. You will configure this bucket and folder to be your query output location.

To access and query your data in Amazon S3:

- [Step 1: Set up an Athena data source and AWS Glue crawler for your Amazon S3 data](#)
- [Step 2: Grant Studio the permissions to access Athena](#)
- [Step 3: Enable Athena default connection in JupyterLab](#)
- [Step 4: Query data in Amazon S3 from JupyterLab notebooks using the SQL extension](#)

Step 1: Set up an Athena data source and AWS Glue crawler for your Amazon S3 data

Follow these steps to index your data in Amazon S3 and create tables in Athena.

Note

To avoid collisions between table names from different Amazon S3 locations, create a separate data source and crawler for each location. Each data source creates a table named after the folder that contain them unless prefixed.

1. Configure a query result location
 - a. Go to the Athena console: <https://console.aws.amazon.com/athena/>.
 - b. From the left menu, choose **Workgroups**.
 - c. Follow the link for the primary workgroup and choose **Edit**.
 - d. In the **Query result configuration** section, enter the Amazon S3 path for your output directory and then choose **Save changes**.
2. Create an Athena data source for your Amazon S3 data
 - a. From the left menu in the Athena console, choose **Data sources** and then **Create Data Source**.
 - b. Choose **S3 - AWS Glue Data Catalog** and then **Next**.
 - c. Leave the default **AWS Glue Data Catalog in this account**, choose **Create a crawler in AWS Glue** and then **Create in AWS Glue**. This opens the AWS Glue console.
3. Use AWS Glue to crawl your data source
 - a. Enter a name and a description for your new crawler and then choose **Next**.
 - b. Under **Data Sources**, choose **Add a data source**.
 - i. If the Amazon Amazon S3 bucket containing your data is in a different AWS account than your SageMaker AI environment, choose **In a different account** for the **Location of the S3 data**.
 - ii. Enter the path to your dataset in Amazon S3. For example:

```
s3://dsoaws/nyc-taxi-orig-cleaned-split-parquet-per-year-multiple-files/ride-info/year=2019/
```

- iii. Keep all other default values and then choose **Add an Amazon S3 data source**. You should see a new Amazon S3 data source in the data sources table.
- iv. Choose **Next**.

- c. Configure the IAM role for the crawler to access your data.

 Note

Each role is scoped down to the data source you specify. When reusing a role, edit the JSON policy to add any new resource you want to grant access to or create a new role for this data source.

- i. Choose **Create new IAM role**.
- ii. Enter a name for the role and then choose **Next**.

4. Create or select a database for your tables

- a. If you do not have an existing database in Athena, choose **Add database** and then **Create a new database**.
- b. Back to your previous crawler creation tab, in **Output configuration**, choose the **Refresh** button. You should now see your newly created database in the list.
- c. Select your database, add an optional prefix in **Table name prefix** and then choose **Next**.

 Note

For the previous example where your data is located at `s3://dsoaws/nyc-taxi-orig-cleaned-split-parquet-per-year-multiple-files/ride-info/year=2019/`, adding the prefix `taxi-ride-` will create a table named `taxi-ride-year_2019`. Adding a prefix helps prevent table name collisions when multiple data locations have identically named folders.

5. Choose **Create crawler**.

6. Run your crawler to index your data. Wait for the crawler run to reach a **Completed** status, which may take a few minutes.

To ensure that a new table was created, go to the left menu in AWS Glue and choose **Databases** then **Tables**. You should now see a new table containing your data.

Step 2: Grant Studio the permissions to access Athena

In the following steps you grant the execution role of your user profile permissions to access Athena.

1. Retrieve the ARN of the execution role associated with your user profile
 - a. Go to the SageMaker AI console at <https://console.aws.amazon.com/sagemaker/> and choose **Domains** in the left menu.
 - b. Follow the name for your domain name.
 - c. In the **User profiles** list, follow the name for your user profile.
 - d. On the **User details** page, copy the ARN of the execution role.
2. Update the policy of your execution role
 - a. Find your AWS region and account ID at the top right of the SageMaker AI console. Use these values and your database name to update the placeholders in the following JSON policy in a text editor.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetS3AndDataSourcesMetadata",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabases",
        "glue:GetSchema",
        "glue:GetTables",
        "s3:ListBucket",
        "s3:GetObject",
        "s3:GetBucketLocation",
```



```

        "glue:GetDatabase",
        "glue:GetTable",
        "glue:ListSchemas",
        "glue:GetPartitions"
    ],
    "Resource": [
        "arn:aws:s3:::*",
        "arn:aws:glue:us-east-1:111122223333:catalog",
        "arn:aws:glue:us-east-1:111122223333:database/db-name"
    ]
},
{
    "Sid": "ExecuteAthenaQueries",
    "Effect": "Allow",
    "Action": [
        "athena:ListDataCatalogs",
        "athena:ListDatabases",
        "athena:ListTableMetadata",
        "athena:StartQueryExecution",
        "athena:GetQueryExecution",
        "athena:RunQuery",
        "athena:StartSession",
        "athena:GetQueryResults",
        "athena:ListWorkGroups",
        "s3:ListMultipartUploadParts",
        "s3:ListBucket",
        "s3:GetBucketLocation",
        "athena:GetDataCatalog",
        "s3:AbortMultipartUpload",
        "s3:GetObject",
        "s3:PutObject",
        "athena:GetWorkGroup"
    ],
    "Resource": [
        "arn:aws:s3:::*"
    ]
},
{
    "Sid": "GetGlueConnectionsAndSecrets",
    "Effect": "Allow",
    "Action": [
        "glue:GetConnections",
        "glue:GetConnection"
    ],

```

```
        "Resource": [
            "*"
        ]
    }
}
```

- b. Go to the IAM console: <https://console.aws.amazon.com/iam/> and choose **Roles** in the left menu.
- c. Search for your role by role name.

 Note

You can retrieve an execution role name from its Amazon Resource Name (ARN) by splitting the ARN on '/' and taking the last element. For example, in the following example of an ARN `arn:aws:iam::112233445566:role/SageMakerStudio-SQLExtension-ExecutionRole`, the name of the execution role is `SageMakerStudio-SQLExtension-ExecutionRole`.

- d. Follow the link for your role.
- e. In the **Permissions** tab, choose **Add permissions** then **Create inline policy**.
- f. Choose the JSON format in the **Policy editor** section.
- g. Copy the policy above and then choose **Next**. Ensure that you have replaced all the `account-id`, `region-name`, and `db-name` with their values.
- h. Enter a name for your policy and then choose **Create policy**.

Step 3: Enable Athena default connection in JupyterLab

In the following steps, you enable a `default-athena-connection` in your JupyterLab application. The default Athena connection allows running SQL queries in Athena directly from JupyterLab, without needing to manually create a connection.

To enable the default Athena connection

1. Go to the SageMaker AI console at <https://console.aws.amazon.com/sagemaker/> and choose **Studio** in the left menu. Using your domain and user profile, launch Studio.
2. Choose the JupyterLab application.

3. If you have not created a space for your JupyterLab application, choose **Create a JupyterLab space**. Enter a name for the space, keep the space as **Private**, and then choose **Create space**. Run your space using the latest version of the SageMaker AI Distribution image.

Otherwise, choose **Run space** on your space to launch a JupyterLab application.

4. Enable Athena default connection:
 - a. In your JupyterLab application, navigate to the **Settings** menu in the top navigation bar and open the **Settings Editor** menu.
 - b. Choose **Data Discovery**.
 - c. Check the box for **Enable default Athena connection**.
 - d. In your JupyterLab application, choose the SQL extension icon
()
in the left navigation pane to open the SQL extension.
 - e. Choose the **Refresh** button at the bottom of the data discovery panel. You should see a `default-athena-connection` in the list of connections.

Step 4: Query data in Amazon S3 from JupyterLab notebooks using the SQL extension

You are ready to query your data using SQL in your JupyterLab notebooks.

1. Open the connection `default-athena-connection` and then **AWSDataCatalog**.
2. Navigate to your database and choose the three dots icon
()
on its right. Select **Query in notebook**.

This automatically populates a notebook cell in JupyterLab with the relevant `%%sm_sql` magic command to connect to the data source. It also adds a sample SQL statement to help you start querying right away.

Note

Ensure to load the extension in the top cell before you run an SQL query.

You can further refine the SQL query using the auto-complete and highlighting features of the extension. See [the section called “SQL editor”](#) for more information on using the SQL extension SQL editor.

SQL extension features and usage

This section details the various features of the JupyterLab SQL extension in Studio, and provides instructions on how to use them. Before you can use the SQL extension to access and query data from your JupyterLab notebooks, an administrator must first configure the connection to your data sources. For information on how administrators can create connections to data sources, see [the section called “Data source connections”](#).

Note

To use the SQL extension, your JupyterLab application must run on a [SageMaker AI distribution](#) image version 1.6 or higher. These SageMaker images have the extension pre-installed.

The extension provides two components to help you access, discover, query, and analyze data from pre-configured data sources.

- Use the *user interface* of the SQL extension to discover and explore your data sources. The UI capabilities can be further divided into the following subcategories.
 - With the **data exploration** UI element, you can browse your data sources and explore their tables, columns, and metadata. For details on the data exploration features of the SQL extension, see [the section called “Browse data”](#).
 - The **connection caching** element caches connections for quick access. For details on connection caching in the SQL extension, see [the section called “Connection caching”](#).
- Use the *SQL Editor and Executor* to write, edit, and run SQL queries against connected data sources.
 - With the **SQL editor** element, you can write, format, and validate SQL statements within the notebooks of your JupyterLab application in Studio. For details on the SQL editor features, see [the section called “SQL editor”](#).