* + - 1. **BigLake: Qwik Start**

experimentLabschedule45 minutesuniversal\_currency\_alt1 Creditshow\_chartIntroductory

infoThis lab may incorporate AI tools to support your learning.

**GSP1040**

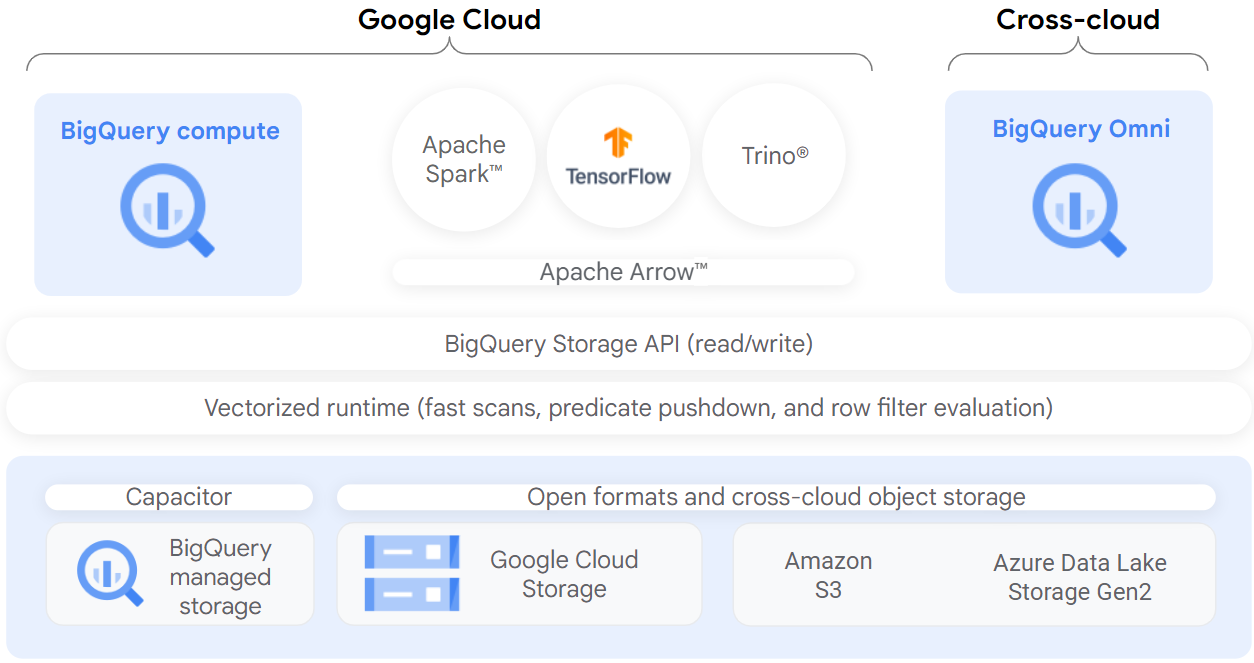


**Overview**

[BigLake](https://cloud.google.com/biglake) is a unified storage engine that simplifies data access for data warehouses and lakes by providing uniform fine-grained access control across multi-cloud storage and open formats.

BigLake extends BigQuery's fine-grained row- and column-level security to tables on data resident object stores such as Amazon S3, Azure Data Lake Storage Gen2, and Google Cloud Storage. BigLake decouples access to the table from the underlying cloud storage data through access delegation. This feature helps you to securely grant row- and column-level access to users and pipelines in your organization without providing them full access to the table.

After you create a BigLake table, you can query it like other BigQuery tables. BigQuery enforces row- and column-level access controls, and every user sees only the slice of data that they are authorized to see. Governance policies are enforced on all access to the data through BigQuery APIs. For example, the [BigQuery Storage API](https://cloud.google.com/bigquery/docs/reference/storage" \t "_blank) lets users access authorized data using open source query engines such as Apache Spark, as the following diagram shows:



**Objectives**

In this lab, you will:

* + - * + Create and view a connection resource.
        + Set up access to a Cloud Storage data lake.
        + Create a BigLake table.
        + Query a BigLake table through BigQuery.
        + Set up access control policies.
        + Upgrade external tables to BigLake tables.

**Setup and requirements**

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources are made available to you.

This hands-on lab lets you do the lab activities in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

* + - * + Access to a standard internet browser (Chrome browser recommended).

**Note:** Use an Incognito (recommended) or private browser window to run this lab. This prevents conflicts between your personal account and the student account, which may cause extra charges incurred to your personal account.

* + - * + Time to complete the lab—remember, once you start, you cannot pause a lab.

**Note:** Use only the student account for this lab. If you use a different Google Cloud account, you may incur charges to that account.

How to start your lab and sign in to the Google Cloud console

* + - * + Click the **Start Lab** button. If you need to pay for the lab, a dialog opens for you to select your payment method. On the left is the Lab Details pane with the following:

The Open Google Cloud console button

Time remaining

The temporary credentials that you must use for this lab

Other information, if needed, to step through this lab

* + - * + Click **Open Google Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

The lab spins up resources, and then opens another tab that shows the Sign in page.

***Tip:*** Arrange the tabs in separate windows, side-by-side.

**Note:**If you see the **Choose an account** dialog, click **Use Another Account**.

* + - * + If necessary, copy the **Username** below and paste it into the **Sign in** dialog.

"Username"

Copied!

You can also find the Username in the Lab Details pane.

* + - * + Click **Next**.
        + Copy the **Password** below and paste it into the **Welcome** dialog.

"Password"

Copied!

You can also find the Password in the Lab Details pane.

* + - * + Click **Next**.

**Important:**You must use the credentials the lab provides you. Do not use your Google Cloud account credentials. **Note:**Using your own Google Cloud account for this lab may incur extra charges.

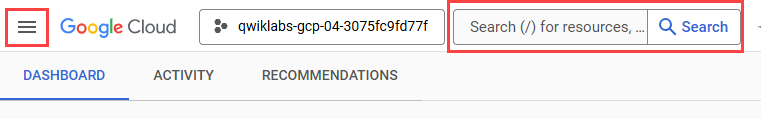
* + - * + Click through the subsequent pages:

Accept the terms and conditions.

Do not add recovery options or two-factor authentication (because this is a temporary account).

Do not sign up for free trials.

After a few moments, the Google Cloud console opens in this tab.

**Note:** To access Google Cloud products and services, click the **Navigation menu** or type the service or product name in the **Search** field.

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

* + - * + Click **Activate Cloud Shell** Activate Cloud Shell icon at the top of the Google Cloud console.
        + Click through the following windows:

Continue through the Cloud Shell information window.

Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project\_ID**, PROJECT\_ID. The output contains a line that declares the **Project\_ID** for this session:

Your Cloud Platform project in this session is set to "PROJECT\_ID"

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

* + - * 1. (Optional) You can list the active account name with this command:

gcloud auth list

Copied!

* + - * 1. Click **Authorize**.

**Output:**

ACTIVE: \*

ACCOUNT: "ACCOUNT"

To set the active account, run:

$ gcloud config set account `ACCOUNT`

* + - * 1. (Optional) You can list the project ID with this command:

gcloud config list project

Copied!

**Output:**

[core]

project = "PROJECT\_ID"

**Note:**For full documentation of gcloud, in Google Cloud, refer to [the gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud).

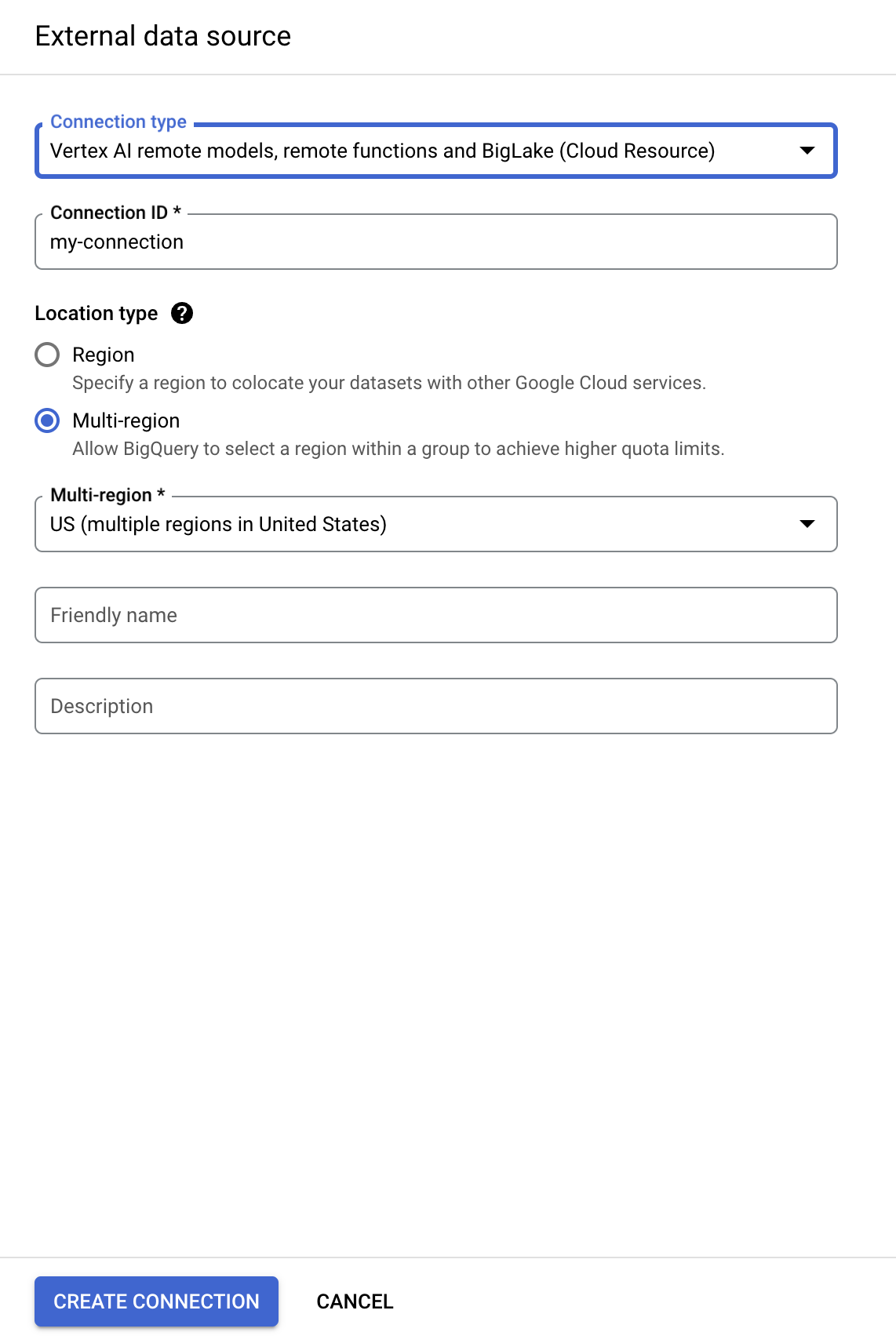
**Task 1. Create a connection resource**

BigLake tables access Google Cloud Storage data using a [connection resource](https://cloud.google.com/bigquery/docs/working-with-connections). A connection resource can be associated with a single table or an arbitrary group of tables in the project.

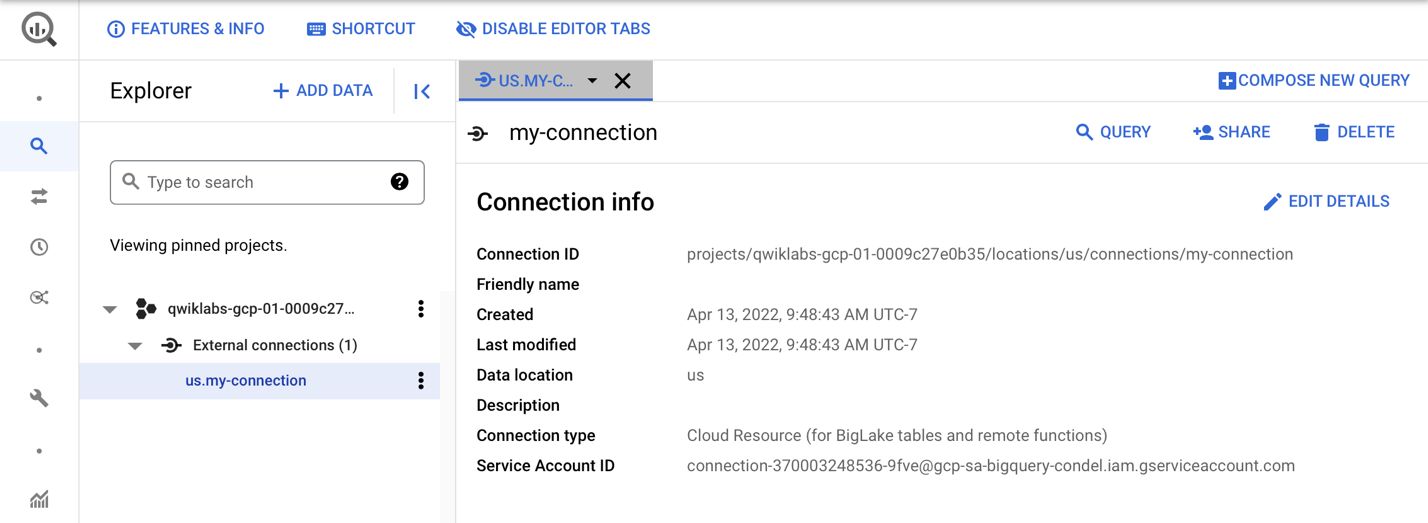
* + - * + From the Navigation Menu, go to **BigQuery** > **BigQuery Studio**. Click **Done**.
        + To create a connection, click **+ADD**, and then click **Connections to external data sources**.

**Note:** If you are prompted to enable the BigQuery Connection API click **Enable API**.

* + - * 1. In the Connection type list, select **Vertex AI remote models, remote functions and BigLake (Cloud Resource)**.



* + - * 1. In the Connection ID field, enter my-connection.
        2. For Location type, choose **Multi-region** and select **US (multiple regions in United States)** from dropdown.
        3. Click **Create connection**.
        4. To view your connection information, select the connection in the navigation menu.



* + - * 1. In the **Connection info** section, copy the Service Account ID. You will need this in the following section.

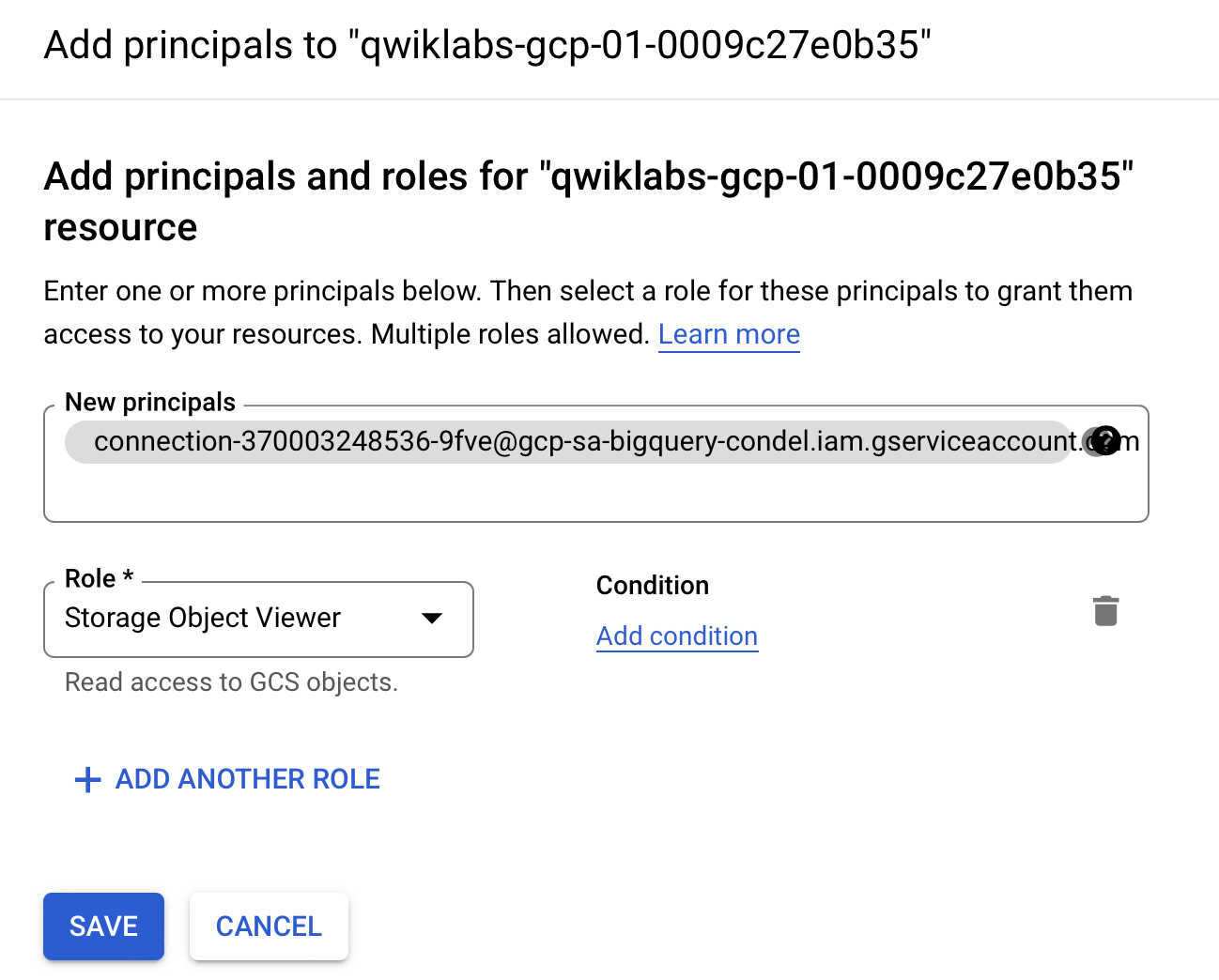
Click *Check my progress* to verify the objective.

Create the connection resource

**Task 2. Set up access to a Cloud Storage data lake**

In this section, you will give the new connection resource read-only access to the Cloud Storage data lake so that BigQuery can access Cloud Storage files on behalf of users. We recommend that you grant the connection resource service account the [Storage Object Viewer](https://cloud.google.com/storage/docs/access-control/iam-roles) IAM role, which lets the service account access Cloud Storage buckets.

* + - * + From the Navigation Menu, go to **IAM & Admin** > **IAM**.
        + Click **+GRANT ACCESS**.
        + In the **New principals** field, enter the service account ID that you copied earlier.
        + In the **Select a role** field, select **Cloud Storage**, and then select **Storage Object Viewer**.



* + - * 1. Click **Save**.

**Note:** After you migrate users to BigLake tables, remove direct Cloud Storage permissions from existing users. Direct file access allows users to bypass governance policies (such as row- and column-level security) set on BigLake tables.

Click *Check my progress* to verify the objective.

Set up access to a Cloud Storage data lake

**Task 3. Create a BigLake table**

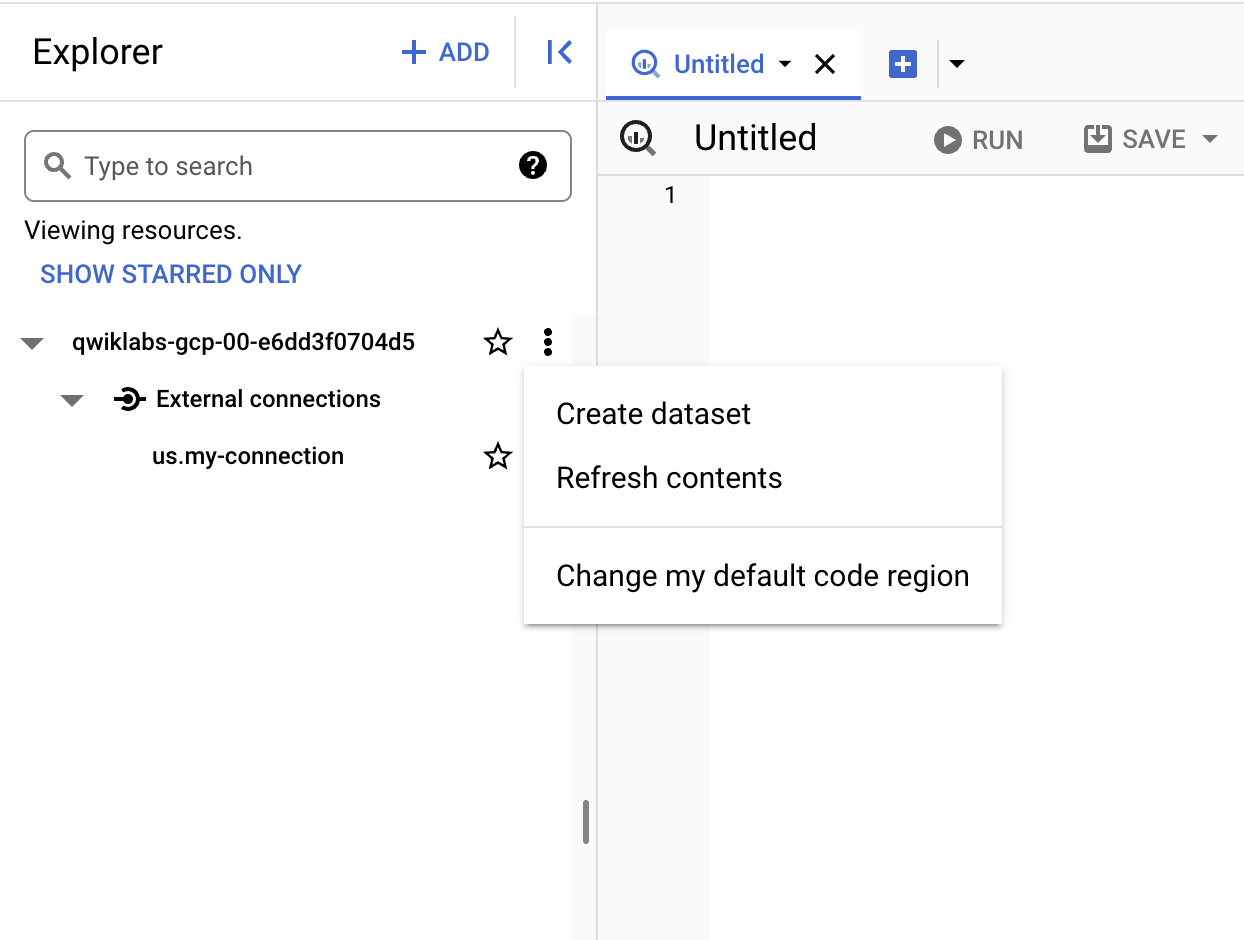
The following example uses the CSV file format, but you can use any format supported by BigLake, as shown in [Limitations](https://www.cloudskillsboost.google/focuses/Limitations). If you're familiar with creating tables in BigQuery, then this process should be similar. The only difference is that you specify the associated cloud resource connection.

**Note:** For optimal performance, we recommend using Cloud Storage single-region or dual-region buckets and not multi-region buckets.

If no schema was provided and the service account was not granted access to the bucket in the previous step, this step will fail with an access denied message.

Create a dataset

* + - * + Navigate back to **BigQuery** > **BigQuery Studio**.
        + Click the three dots next to your project name and select **Create dataset**.

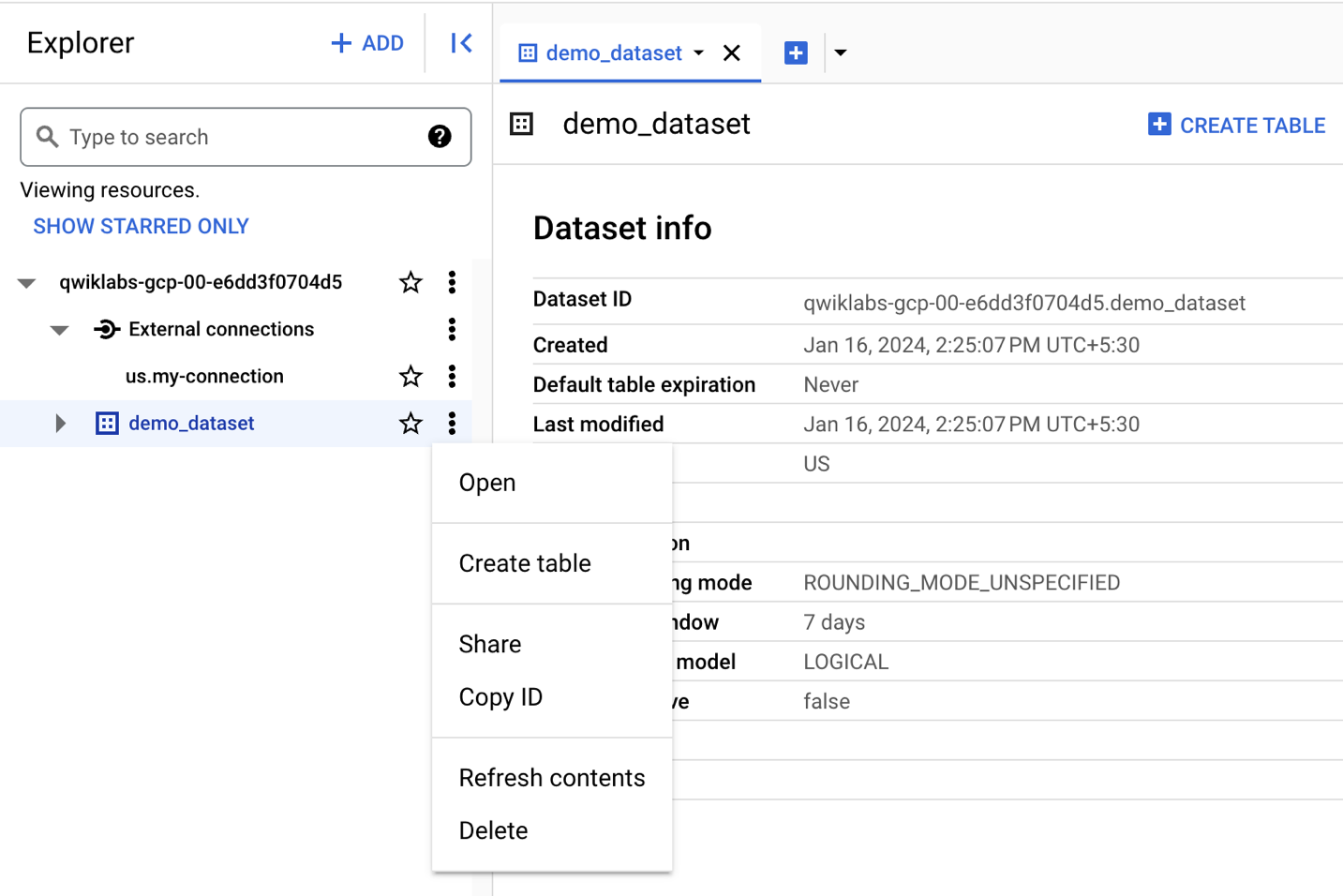


* + - * 1. For the **Dataset ID**, use demo\_dataset.
        2. For Location type, choose **Multi-region** and select **US (multiple regions in United States)** from dropdown.
        3. Leave the rest of the fields as default and click **Create Dataset**.

Now that you have a dataset created, you can copy an existing dataset from Cloud Storage into BigQuery.

Create the table

* + - * + Click three dots next to **demo\_dataset**, then choose **Create table**.

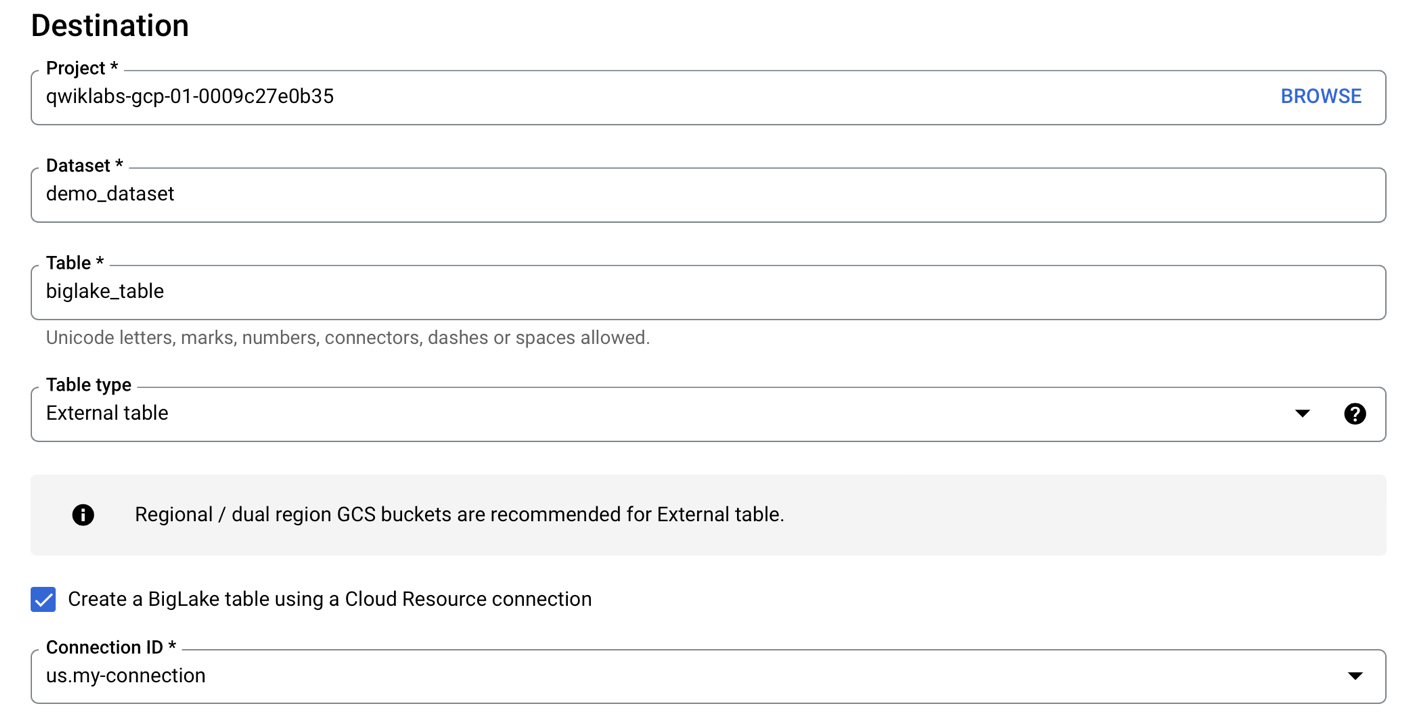


* + - * 1. Under Source for **Create table from**, choose **Google Cloud Storage**.

**Note:** A Cloud Storage bucket has been created with two datasets that you will use in this lab.

* + - * 1. Click **Browse** to select the dataset. Navigate to the bucket named as Project ID and then customer.csv file to import it into BigQuery, and click **Select**.
        2. Under **Destination**, verify your lab project has been selected and you're using the **demo\_dataset**.
        3. For the table name, use biglake\_table.
        4. Set the table type to **External Table**.
        5. Select the box to **Create a BigLake table using a Cloud Resource connection**.

Verify that your connection ID **us.my-connection** is selected. Your configuration should resemble the following:



* + - * 1. Under **Schema**, enable **Edit as text** and copy and paste the following schema into the text box:

[

{

"name": "customer\_id",

"type": "INTEGER",

"mode": "REQUIRED"

},

{

"name": "first\_name",

"type": "STRING",

"mode": "REQUIRED"

},

{

"name": "last\_name",

"type": "STRING",

"mode": "REQUIRED"

},

{

"name": "company",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "address",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "city",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "state",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "country",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "postal\_code",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "phone",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "fax",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "email",

"type": "STRING",

"mode": "REQUIRED"

},

{

"name": "support\_rep\_id",

"type": "INTEGER",

"mode": "NULLABLE"

}

]

Copied!

**Note:** Typically data lakes **do not** have a predefined schema. For this lab's purposes, we are using one to make setting column-level policies clearer.

* + - * 1. Click **Create Table**.

Click *Check my progress* to verify the objective.

Create the BigLake table

**Task 4. Query a BigLake table through BigQuery**

Now that you've created the BigLake table, you can use any BigQuery client to submit a query.

* + - * + From the **biglake\_table** preview toolbar, click **Query** > **In new tab**.
        + Run the following to query the BigLake table through the BigQuery Editor:

SELECT \* FROM `Project ID.demo\_dataset.biglake\_table`

Copied!

* + - * 1. Click **Run**.
        2. Verify you can see all of the columns and data in the resulting table.

**Task 5. Set up access control policies**

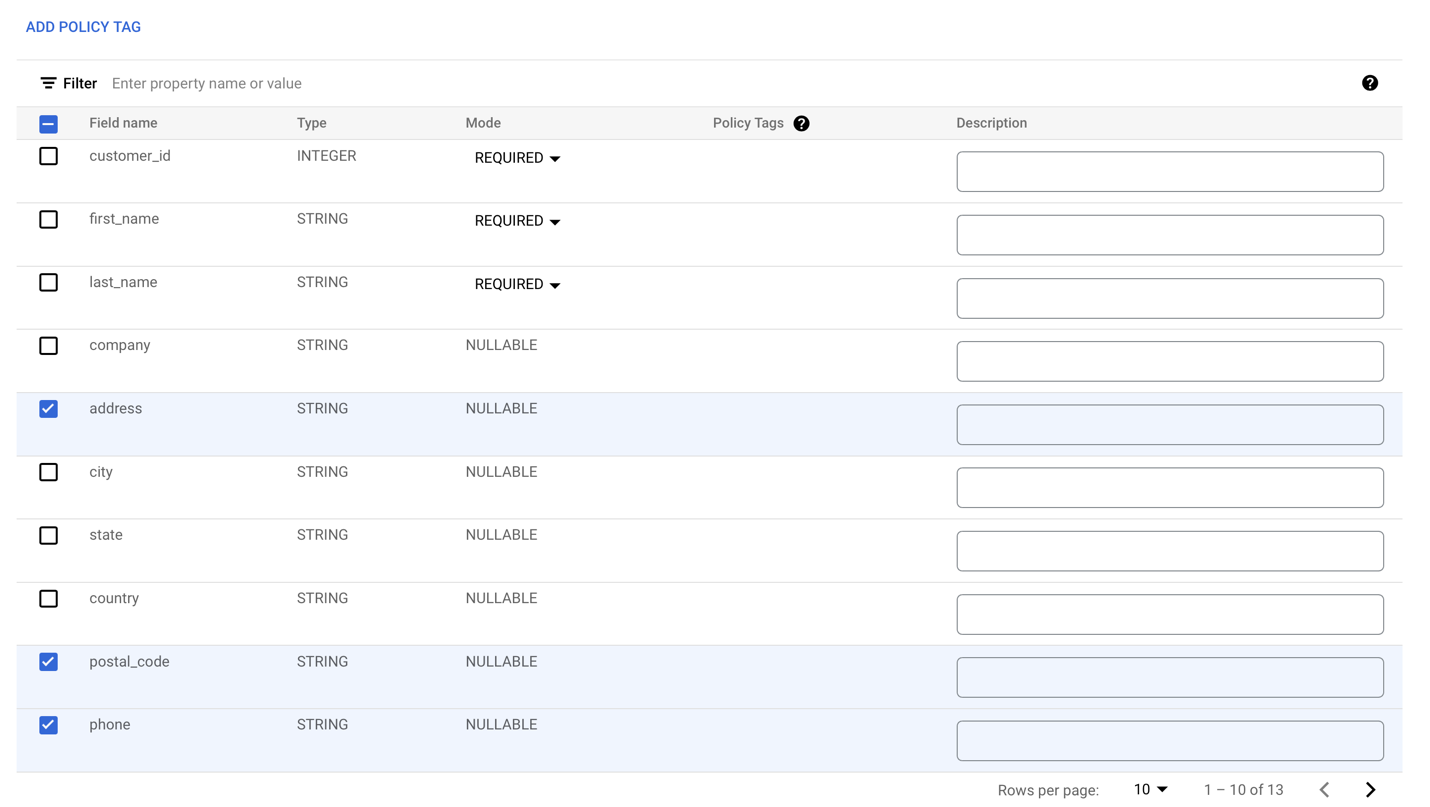
Once a BigLake table has been created, it can be managed in a similar fashion to BigQuery tables. To create access control policies for BigLake tables, you'll first create a taxonomy of policy tags in [BigQuery](https://cloud.google.com/bigquery/docs/best-practices-policy-tags" \t "_blank). Then, apply the policy tags to the sensitive rows or columns. In this section, you will create a column level policy. For directions on setting up row-level security, see the [row-level security guide](https://cloud.google.com/bigquery/docs/managing-row-level-security).

For these purposes, a BigQuery taxonomy named taxonomy name and an associated policy tag named **biglake-policy** has been created for you.

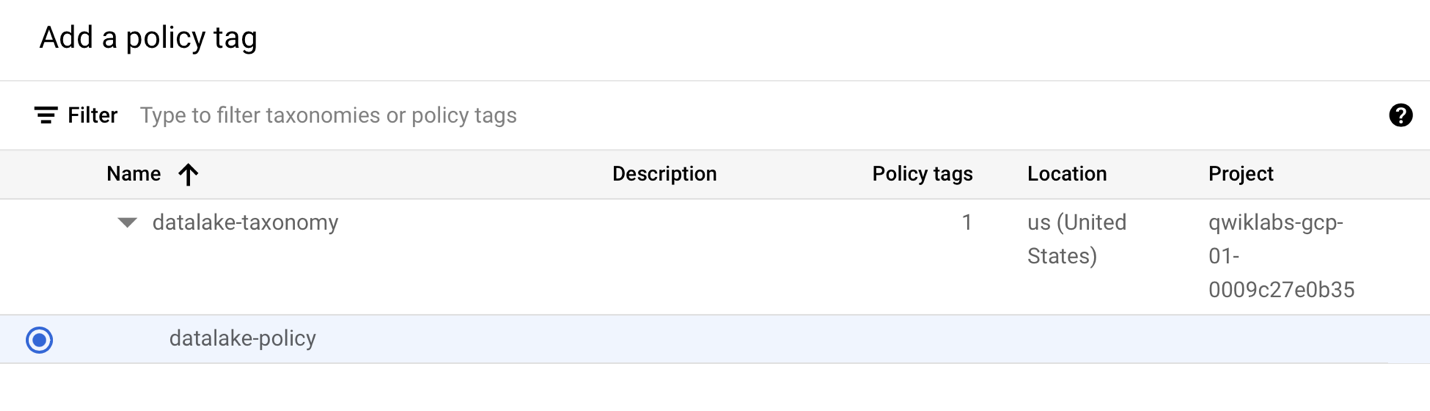
Add policy tags to columns

You will now use the policy tag you created to restrict access to certain columns within the BigQuery table. For this example, you will restrict access to sensitive information such as address, postal code, and phone number.

* + - * + From the Navigation Menu, go to **BigQuery** > **BigQuery Studio**.
        + Navigate to **demo-dataset** > **biglake\_table** and click the table to open the table schema page.
        + Click **Edit Schema**.
        + Check the boxes next to the **address**, **postal\_code**, and **phone** fields.

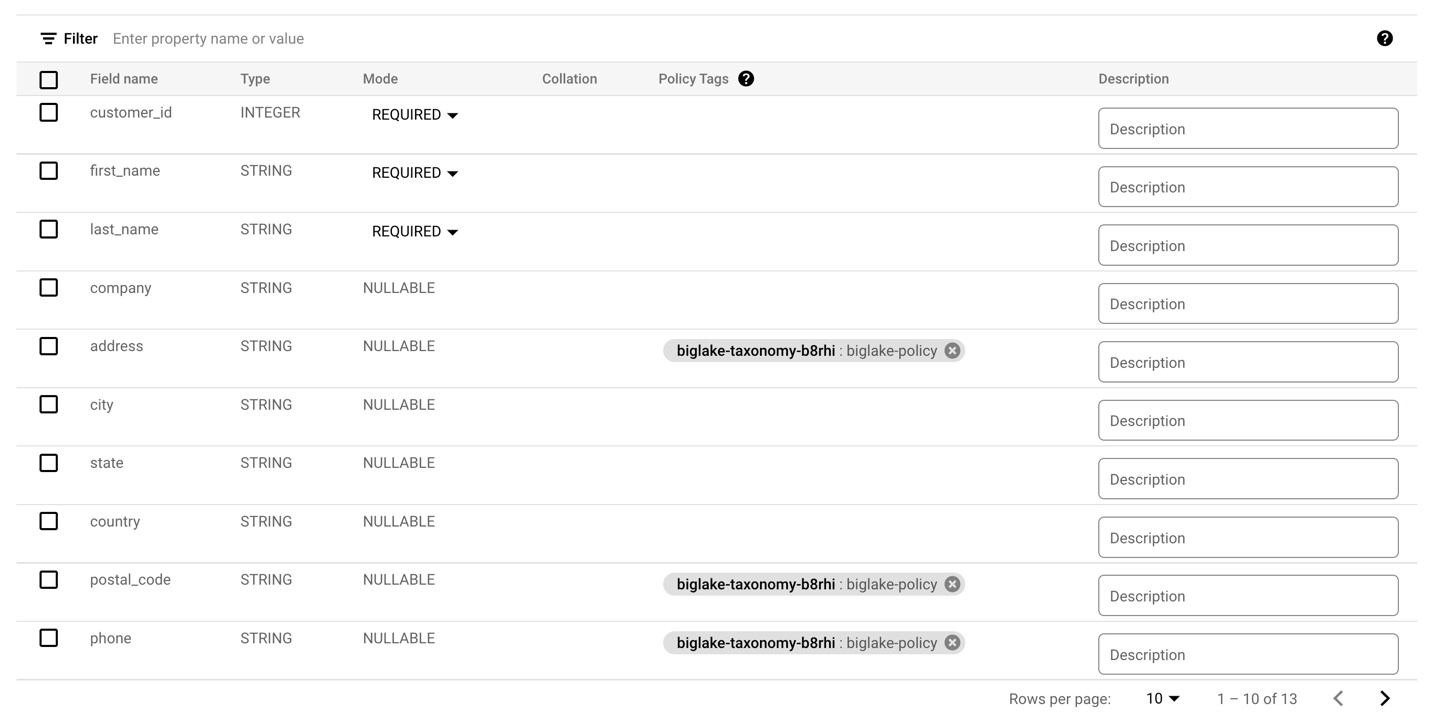


* + - * 1. Click **Add policy tag**.
        2. Click taxonomy name to expand it to select **biglake-policy**.

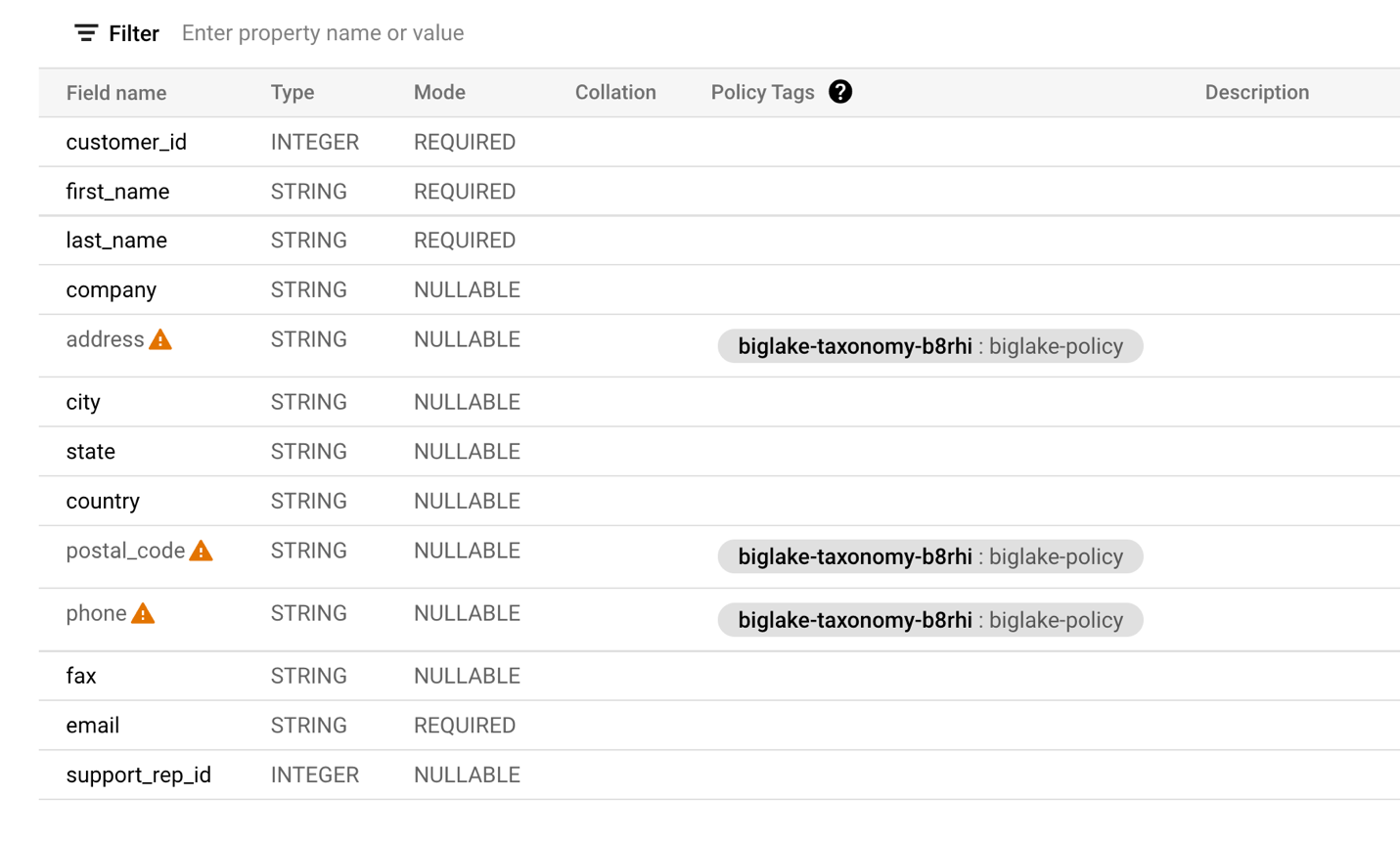


* + - * 1. Click **Select**.

Your columns should now have the policy tags attached to them.



* + - * 1. Click **Save**.
        2. Verify your table schema now resembles the following.



**Note:** The warning signs on the columns signal you don't have access to those particular fields based on the security policies in place.

Verify the column level security

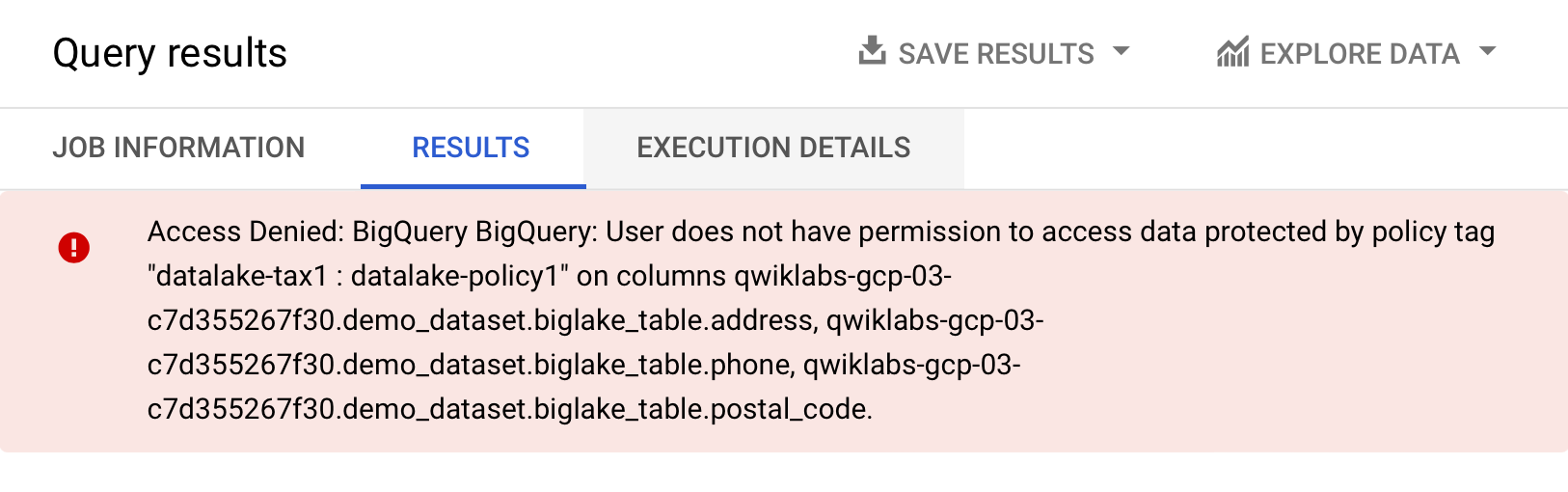
* + - * + Open the query editor for the **biglake\_table**.
        + Run the following to query the BigLake table through the BigQuery Editor:

SELECT \* FROM `Project ID.demo\_dataset.biglake\_table`

Copied!

* + - * 1. Click **Run**.

You should receive an error access denied error:



* + - * 1. Now, run the following query, omitting the columns you don't have access to:

SELECT \* EXCEPT(address, phone, postal\_code)

FROM `Project ID.demo\_dataset.biglake\_table`

Copied!

The query should run without any issues and return the columns you have access to. This example shows that column level security enforced through BigQuery can also be applied to BigLake tables.

**Task 6. Upgrade external tables to BigLake tables**

You can upgrade existing tables to BigLake tables by associating the existing table to a cloud resource connection. For a complete list of flags and arguments, see [bq update](https://cloud.google.com/bigquery/docs/reference/bq-cli-reference" \l "bq_update" \t "_blank)and [bq mkdef](https://cloud.google.com/bigquery/docs/reference/bq-cli-reference" \l "bq_mkdef" \t "_blank).

Create the external table

* + - * + Click three dots next to **demo\_dataset**, then choose **Create table**.
        + Under Source for **Create table from**, choose **Google Cloud Storage**.
        + Click **Browse** to select the dataset. Navigate to the bucket named Project IDand then invoice.csv file to import it into BigQuery, and click **Select**.
        + Under **Destination**, verify your lab project has been selected and you're using the **demo\_dataset**.
        + For the table name, use external\_table.
        + Set the table type to **External Table**.

**Note:** Do not specify a Cloud Resource connection yet.

* + - * 1. Under **Schema**, enable **Edit as text** and copy and paste the following schema into the text box:

[

{

"name": "invoice\_id",

"type": "INTEGER",

"mode": "REQUIRED"

},

{

"name": "customer\_id",

"type": "INTEGER",

"mode": "REQUIRED"

},

{

"name": "invoice\_date",

"type": "TIMESTAMP",

"mode": "REQUIRED"

},

{

"name": "billing\_address",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "billing\_city",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "billing\_state",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "billing\_country",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "billing\_postal\_code",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "total",

"type": "NUMERIC",

"mode": "REQUIRED"

}

]

Copied!

* + - * 1. Click **Create Table**.

Click *Check my progress* to verify the objective.

Create the external table

Update external table to Biglake table

* + - * + Open a new Cloud Shell window and run the following command to generate a new [external table definition](https://cloud.google.com/bigquery/external-table-definition#table-definition) that specifies the connection to use:

export PROJECT\_ID=$(gcloud config get-value project)

bq mkdef \

--autodetect \

--connection\_id=$PROJECT\_ID.US.my-connection \

--source\_format=CSV \

"gs://$PROJECT\_ID/invoice.csv" > /tmp/tabledef.json

Copied!

* + - * 1. Verify your table definition has been created:

cat /tmp/tabledef.json

Copied!

* + - * 1. Get the schema from your table:

bq show --schema --format=prettyjson demo\_dataset.external\_table > /tmp/schema

Copied!

* + - * 1. Update the table using the new external table definition:

bq update --external\_table\_definition=/tmp/tabledef.json --schema=/tmp/schema demo\_dataset.external\_table

Copied!

Click *Check my progress* to verify the objective.

Update external table to Biglake table

Verify the updated table

* + - * + From the Navigation Menu, go to **BigQuery** > **BigQuery Studio**.
        + Navigate to **demo-dataset** > and double click **external\_table**.
        + Open the **Details** tab.
        + Verify under External Data Configuration that the table is now using the proper Connection ID.



Great! You successfully upgraded the existing external table to a BigLake table by associating it to a cloud resource connection.