Build and Execute MySQL, PostgreSQL, and SQLServer to Data Catalog Connectors

cloudskillsboost.google/focuses/11999

GSP814

Overview

Data Catalog is a fully managed and scalable metadata management service that empowers organizations to quickly discover, understand, and manage all their data.

It offers a simple and easy-to-use search interface for data discovery, a flexible and powerful cataloging system for capturing both technical and business metadata, and a strong security and compliance foundation with Cloud Data Loss Prevention (DLP) and Cloud Identity and Access Management (IAM) integrations.

Using Data Catalog

There are two main ways you interact with Data Catalog:

- Searching for data assets that you have access to.
- Tagging assets with metadata.

What you will learn

In this lab, you will learn how to:

- Enable the Data Catalog API so that you can use this service in your Google Cloud project.
- Execute SQLServer to Data Catalog connector.
- Execute PostgreSQL to Data Catalog connector.
- Execute MySQL to Data Catalog connector.

Prerequisites

Very Important: Before starting this lab, log out of your personal or corporate gmail account, or run this lab in Incognito. This prevents sign-in confusion while the lab is running.

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 will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that 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 or private browser window to run this lab. This prevents any 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: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

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

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
    - The **Open Google Console** button
    - Time remaining
    - The temporary credentials that you must use for this lab
    - Other information, if needed, to step through this lab
2. Click **Open Google Console**. 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**.
3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.

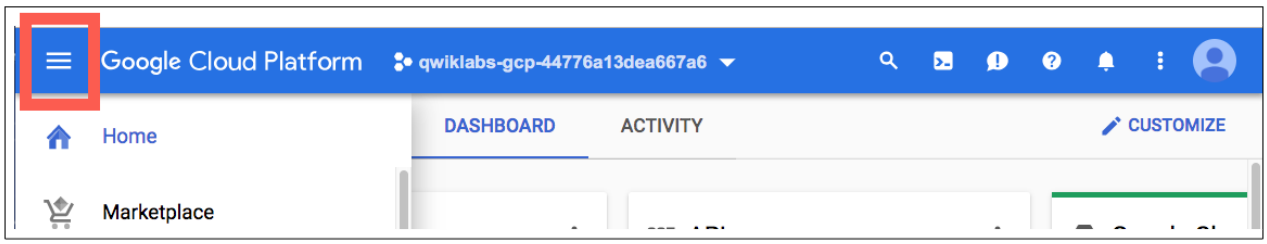
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

   Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.
5. 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 Cloud Console opens in this tab.

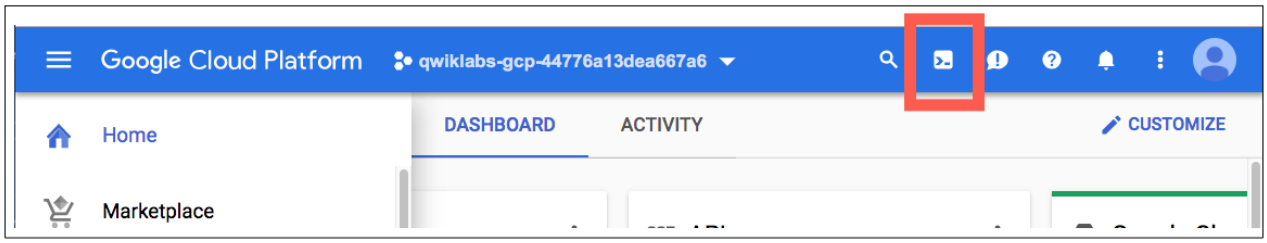
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



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.

1. In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



2. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your **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 YOUR_PROJECT_ID
`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

gcloud auth list
(Output)

ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net To set the active account, run: $ gcloud config set account `ACCOUNT`

4. (Optional) You can list the project ID with this command:

gcloud config list project
(Output)

[core] project = <project_ID>
(Example output)

[core] project = qwiklabs-gcp-44776a13dea667a6 For full documentation of `gcloud`, in Google Cloud, Cloud SDK documentation, see the gcloud command-line tool overview.

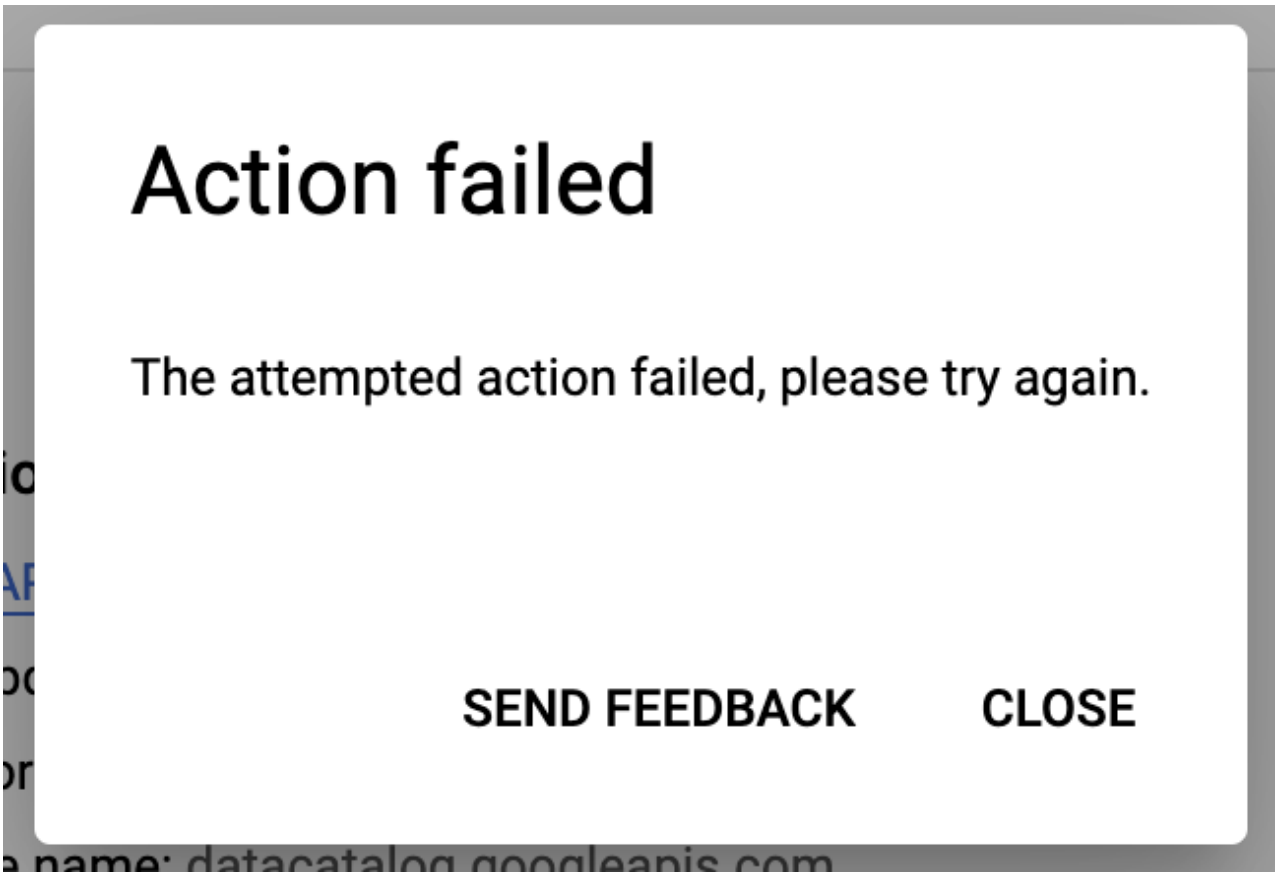
Enable the Data Catalog API

Open the **Navigation menu** and select **APIs and Services** > **Library**.

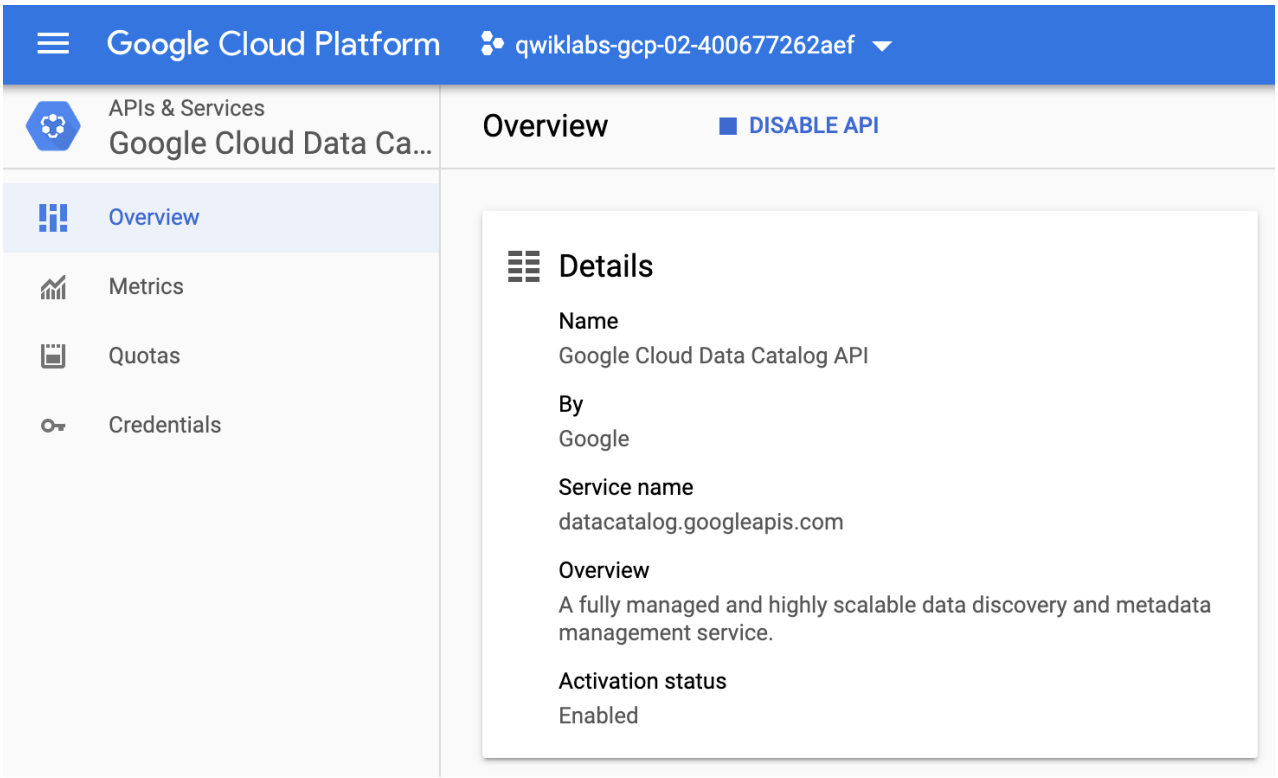
In the search bar, enter in "Data Catalog" and select the `Google Cloud Data Catalog API`.

Then click **Enable**.

If you run into the following error after trying to enable the Data Catalog API:



Click **Close** and **refresh** your browser tab. Then click **Enable** again. The Data Catalog API should be successfully enabled:



Click *Check my progress* to verify the objective. Enable the Data Catalog API

SQLServer to Data Catalog

Start by setting up your environment. Run the following command to set your Project ID, replacing `<YOUR_PROJECT_ID>` with the Project ID found in the connection details panel:

gcloud config set project <YOUR_PROJECT_ID>
Next set it as an environment variable:

export PROJECT_ID=$(gcloud config get-value project)

Create the SQLServer Database

In your Cloud Shell session, run the following command to download the scripts to create and populate your SQLServer instance:

gsutil cp gs://spls/gsp814/cloudsql-sqlserver-tooling.zip . unzip cloudsql-sqlserver-tooling.zip
Now change your current working directory to the downloaded directory:

cd cloudsql-sqlserver-tooling
Now run the `init-db.sh` script.

bash init-db.sh
This will create your SQLServer instance and populate it with a random schema.

If you get an `Error: Failed to load "tfplan" as a plan file`, re-run the `init-db` script.
This will take `around 5 to 10 minutes` to complete. You can move on when you receive the following output:

CREATE TABLE factory_warehouse15797.employees53b82dc5 (school80581 REAL, reason91250 DATETIME, randomdata32431 BINARY, phone_number52754 REAL, person66471 REAL, credit_card75527 DATETIME) COMPLETED
Click *Check my progress* to verify the objective. Create the SQLServer Database

Set Up the Service Account

Run the following command to create a Service Account:

gcloud iam service-accounts create sqlserver2dc-credentials \ --display-name "Service Account for SQLServer to Data Catalog connector" \ --project $PROJECT_ID
Now create and download the Service Account Key.

gcloud iam service-accounts keys create "sqlserver2dc-credentials.json" \ --iam-account "sqlserver2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com"
Add the Data Catalog admin role to the Service Account:

gcloud projects add-iam-policy-binding $PROJECT_ID \ --member "serviceAccount:sqlserver2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com" \ --

quiet \ --project $PROJECT_ID \ --role "roles/datacatalog.admin"
Click *Check my progress* to verify the objective. Set Up the Service Account for SQLServer

Execute SQLServer to Data Catalog connector.

You can build the SQLServer connector yourself by going to this GitHub repository.

To facilitate its usage, we are going to use a docker image.

The variables needed were output by the Terraform config.

Change directories into the location of the Terraform scripts:

cd infrastructure/terraform/
Grab the environment variables:

public_ip_address=$(terraform output -raw public_ip_address) username=$(terraform output -raw username) password=$(terraform output -raw password) database=$(terraform output -raw db_name)
Change back to the root directory for the example code:

cd ~/cloudsql-sqlserver-tooling
Run the following command to execute the connector:

docker run --rm --tty -v \ "$PWD":/data mesmacosta/sqlserver2datacatalog:stable \ --datacatalog-project-id=$PROJECT_ID \ --datacatalog-location-id=us-central1 \ --sqlserver-host=$public_ip_address \ --sqlserver-user=$username \ --sqlserver-pass=$password \ --sqlserver-database=$database
Soon after you should receive the following output:

==========End sqlserver-to-datacatalog==========
Click *Check my progress* to verify the objective. Execute SQLServer to Data Catalog connector

Search for the SQLServer Entries in Data Catalog

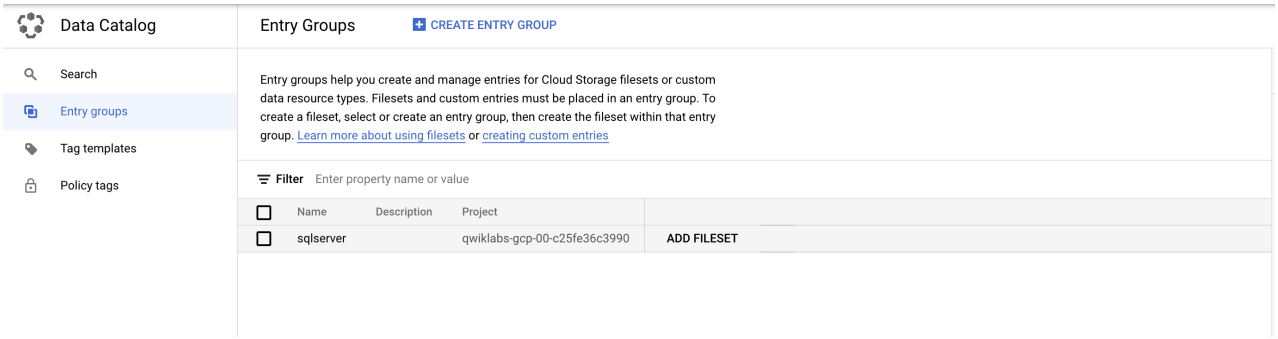
After the script finishes, open the navigation menu and select **Data Catalog** from the list of services.

In the the **Data Catalog** page, click on **Tag Templates**.

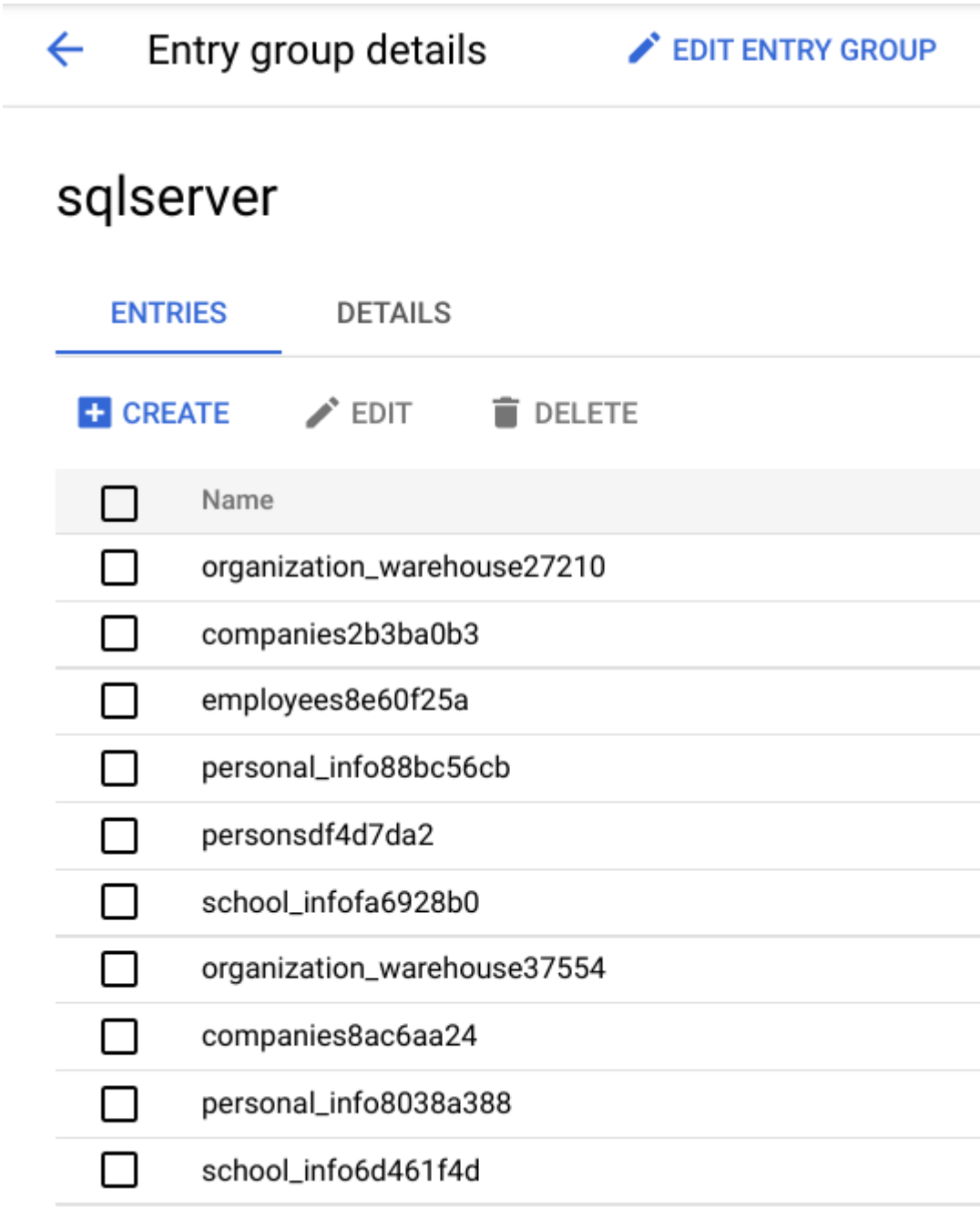
You should see your **sqlserver** Tag Templates listed.

Next, select **Entry Groups**.

You should see the **sqlserver** Entry Group:



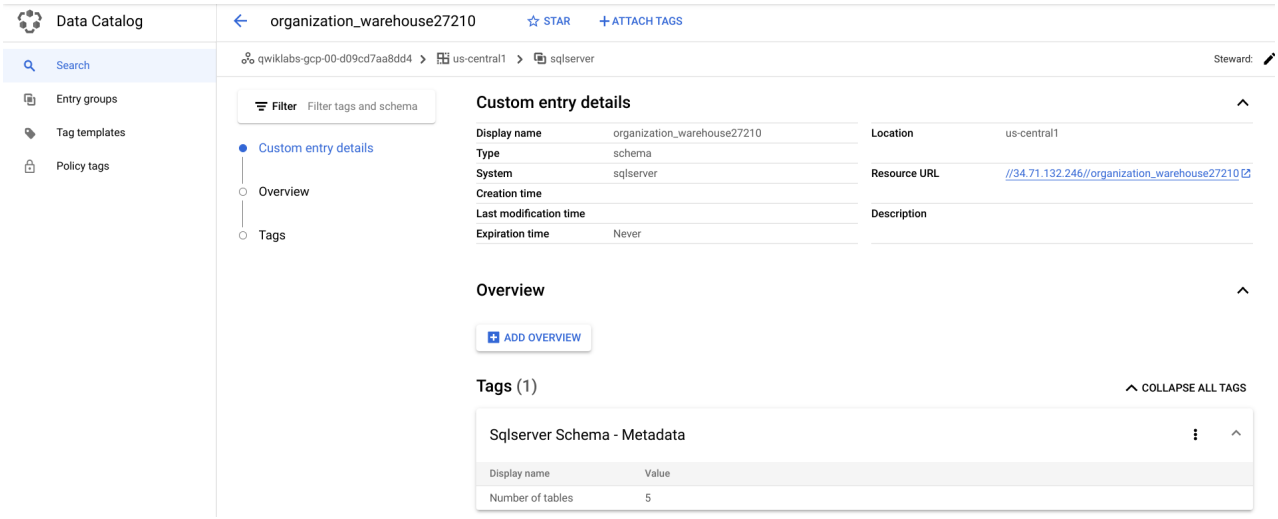
Now click on the `sqlserver` Entry Group. Your console should resemble the following:



This is the real value of an Entry Group—you can see all entries that belong to sqlserver using the UI.

Click on one of the `warehouse` entries. Look at the Custom entry details and tags:

Custom entry details and Tags:



This is the real value the connector adds — it allows you to have the metadata searchable in Data Catalog.

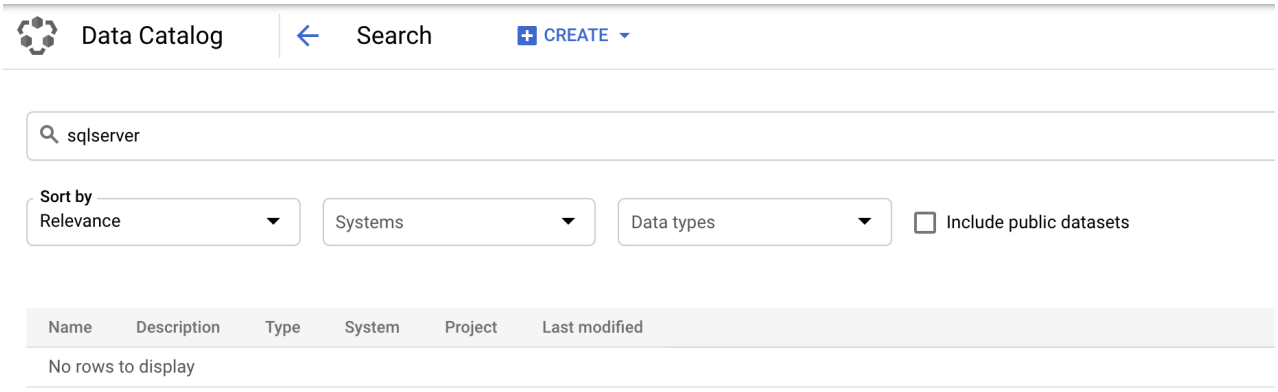
Cleaning up

To delete the created resources, run the following command to delete the SQLServer metadata:

./cleanup-db.sh
Now execute the cleaner container:

docker run --rm --tty -v \ "$PWD":/data mesmacosta/sqlserver-datacatalog-cleaner:stable \ --datacatalog-project-ids=$PROJECT_ID \ --rdbms-type=sqlserver \ --table-container-type=schema
Now run the following command to delete the SQLServer database:

./delete-db.sh
From the **Navigation menu** click **Data Catalog**. Search for **sqlserver**. You will no longer see the SQLServer Tag Templates in the results:



Ensure you see the following output in Cloud Shell before you move on:

Cloud SQL Instance deleted COMPLETED

You will now learn how to do the same thing with a PostgreSQL instance.

PostgreSQL to Data Catalog

Create the PostgreSQL Database

Run the following command in Cloud Shell to return to your home directory:

cd
Run the following command to clone the Github repository:

gsutil cp gs://spls/gsp814/cloudsql-postgresql-tooling.zip . unzip cloudsql-postgresql-tooling.zip
Now change your current working directory to the cloned repo directory:

cd cloudsql-postgresql-tooling
Now execute the `init-db.sh` script:

bash init-db.sh
This will create your PostgreSQL instance and populate it with a random schema. This can take `around 10 to 15 minutes` to complete.

If you get an `Error: Failed to load "tfplan" as a plan file`, re-run the `init-db` script.
Soon after you should receive the following output:

CREATE TABLE factory_warehouse69945.home17e97c57 (house57588 DATE, paragraph64180 SMALLINT, ip_address61569 JSONB, date_time44962 REAL, food19478 JSONB, state8925 VARCHAR(25), cpf75444 REAL, date_time96090 SMALLINT, reason7955 CHAR(5), phone_number96292 INT, size97593 DATE, date_time609 CHAR(5), location70431 DATE) COMPLETED
Click *Check my progress* to verify the objective. Create the PostgreSQL Database

Set Up the Service Account

Create a Service Account.

gcloud iam service-accounts create postgresql2dc-credentials \ --display-name "Service Account for PostgreSQL to Data Catalog connector" \ --project $PROJECT_ID
Next create and download the Service Account Key.

gcloud iam service-accounts keys create "postgresql2dc-credentials.json" \ --iam-account "postgresql2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com"
Next add Data Catalog admin role to the Service Account.

gcloud projects add-iam-policy-binding $PROJECT_ID \ --member "serviceAccount:postgresql2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com" \ --quiet \ --project $PROJECT_ID \ --role "roles/datacatalog.admin"

Click *Check my progress* to verify the objective. Create a Service Account for postgresql

Execute PostgreSQL to Data Catalog connector

You can build the PostgreSQL connector yourself by going to this GitHub repository.

To facilitate its usage, we are going to use a docker image.

The variables needed were output by the Terraform config.

Change directories into the location of the Terraform scripts:

cd infrastructure/terraform/
Grab the environment variables:

public_ip_address=$(terraform output -raw public_ip_address) username=$(terraform output -raw username) password=$(terraform output -raw password) database=$(terraform output -raw db_name)
Change back to the root directory for the example code:

cd ~/cloudsql-postgresql-tooling
Execute the connector:

docker run --rm --tty -v \ "$PWD":/data mesmacosta/postgresql2datacatalog:stable \ --datacatalog-project-id=$PROJECT_ID \ --datacatalog-location-id=us-central1 \ --postgresql-host=$public_ip_address \ --postgresql-user=$username \ --postgresql-pass=$password \ --postgresql-database=$database
Soon after you should receive the following output:

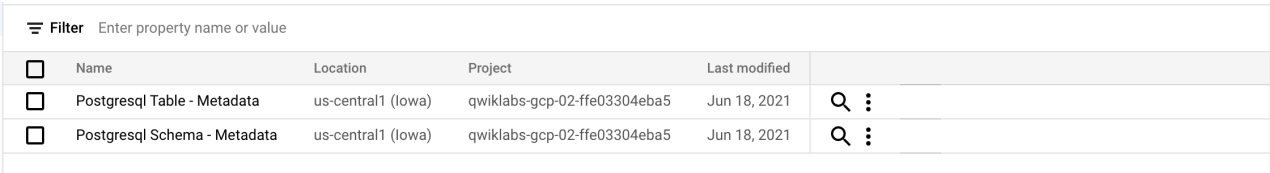
==========End postgresql-to-datacatalog==========
Click *Check my progress* to verify the objective. Execute PostgreSQL to Data Catalog connector

Check the results of the script

Ensure that you are in the Data Catalog home page.

Click on **Tag Templates**.

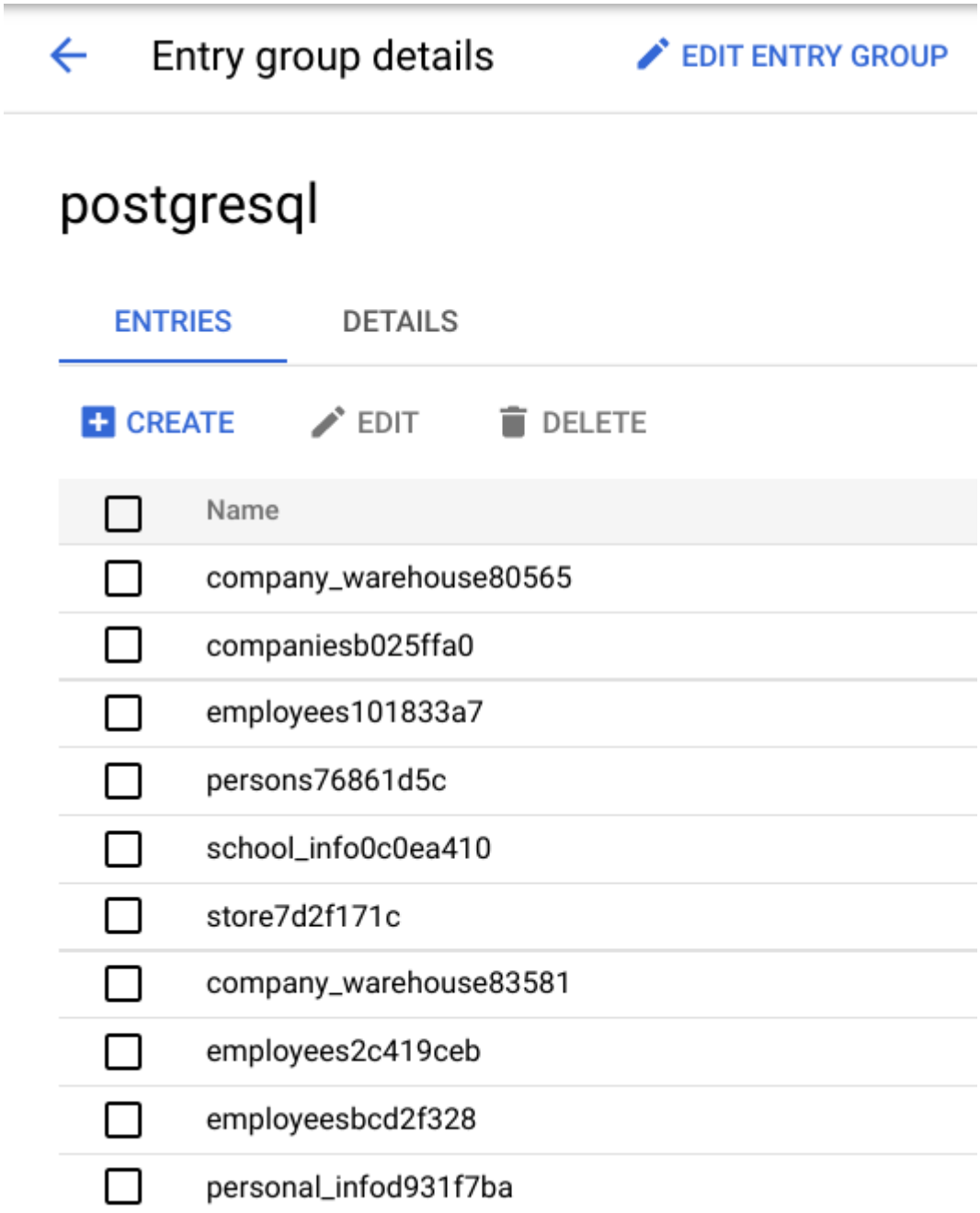
You should see the following postgresql Tag Templates:



Click on **Entry groups**.

You should see the following postgresql Entry Group:

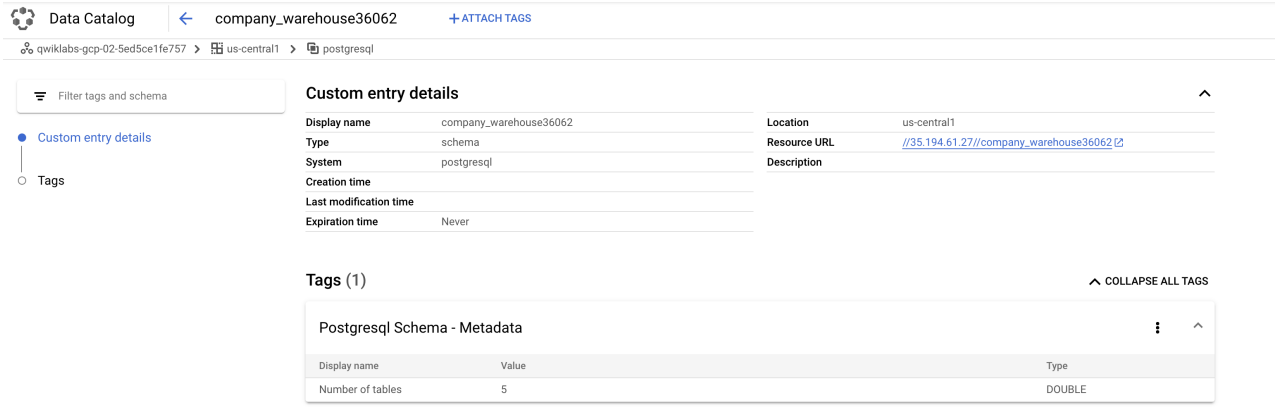
Now click on the `postgresql` Entry Group. Your console should resemble the following:



This is the real value of an Entry Group — you can see all entries that belong to postgresql using the UI.

Click on one of the `warehouse` entries. Look at the Custom entry details and tags:

Custom entry details and Tags:



This is the real value the connector adds—it allows you to have the metadata searchable in Data Catalog.

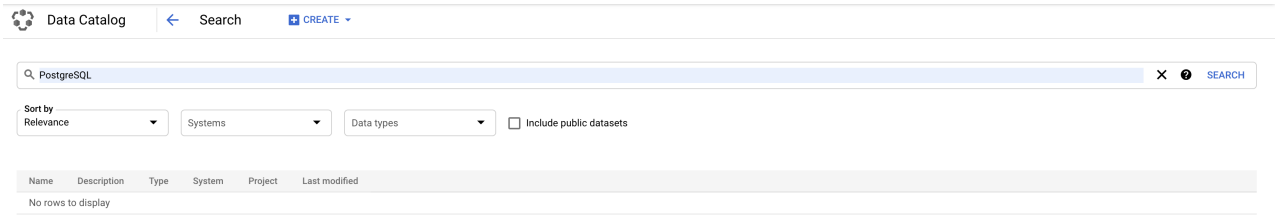
Cleaning up

To delete the created resources, run the following command to delete the PostgreSQL metadata:

./cleanup-db.sh
Now execute the cleaner container:

docker run --rm --tty -v \ "$PWD":/data mesmacosta/postgresql-datacatalog-cleaner:stable \ --datacatalog-project-ids=$PROJECT_ID \ --rdbms-type=postgresql \ --table-container-type=schema
Finally, delete the PostgreSQL database:

./delete-db.sh
Now, from the **Navigation menu** click on **Data Catalog**. Search for **PostgreSQL**. You will no longer see the PostgreSQL Tag Templates in the results:



Ensure you see the following output in Cloud Shell before you move on:

Cloud SQL Instance deleted COMPLETED
You will now learn how to do the same thing with a MySQL instance.

MySQL to Data Catalog

Create the MySQL Database

Run the following command in Cloud Shell to return to your home directory:

cd
Run the following command to download the scripts to create and populate your MySQL instance:

gsutil cp gs://spls/gsp814/cloudsql-mysql-tooling.zip . unzip cloudsql-mysql-tooling.zip
Now change your current working directory to the cloned repo directory:

cd cloudsql-mysql-tooling
Next execute the `init-db.sh` script:

bash init-db.sh
This will create your MySQL instance and populate it with a random schema. After a few minutes, you should receive the following output:

CREATE TABLE factory_warehouse14342.persons88a5ebc4 (address9634 TEXT, cpf12934 FLOAT, food88799 BOOL, food4761 LONGTEXT, credit_card44049 FLOAT, city8417 TINYINT, name76076 DATETIME, address19458 TIME, reason49953 DATETIME) COMPLETED If you get an `Error: Failed to load "tfplan" as a plan file` , re-run the `init-db` script.
Click *Check my progress* to verify the objective. Create the MySQL Database

Set Up the Service Account

Run the following to create a Service Account:

gcloud iam service-accounts create mysql2dc-credentials \ --display-name "Service Account for MySQL to Data Catalog connector" \ --project $PROJECT_ID
Next, create and download the Service Account Key:

gcloud iam service-accounts keys create "mysql2dc-credentials.json" \ --iam-account "mysql2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com"
Next add Data Catalog admin role to the Service Account:

gcloud projects add-iam-policy-binding $PROJECT_ID \ --member "serviceAccount:mysql2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com" \ --quiet \ --project $PROJECT_ID \ --role "roles/datacatalog.admin"
Click *Check my progress* to verify the objective. Create a Service Account for MySQL

Execute MySQL to Data Catalog connector

You can build the MySQL connector yourself by going to this GitHub repository.

To facilitate its usage, this lab uses a docker image.

The variables needed were output by the Terraform config.

Change directories into the location of the Terraform scripts:

cd infrastructure/terraform/

Grab the environment variables:

public_ip_address=$(terraform output -raw public_ip_address) username=$(terraform output -raw username) password=$(terraform output -raw password) database=$(terraform output -raw db_name)
Change back to the root directory for the example code:

cd ~/cloudsql-mysql-tooling
Execute the connector:

docker run --rm --tty -v \ "$PWD":/data mesmacosta/mysql2datacatalog:stable \ --datacatalog-project-id=$PROJECT_ID \ --datacatalog-location-id=us-central1 \ --mysql-host=$public_ip_address \ --mysql-user=$username \ --mysql-pass=$password \ --mysql-database=$database
Soon after you should receive the following output:

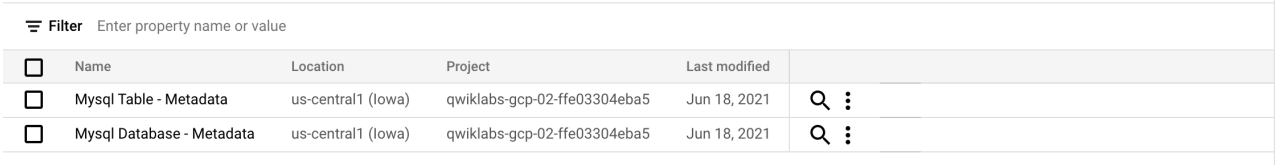
============End mysql-to-datacatalog============
Click *Check my progress* to verify the objective. Execute MySQL to Data Catalog connector

Check the results of the script

Ensure that you are in the Data Catalog home page.

Click on **Tag Templates**.

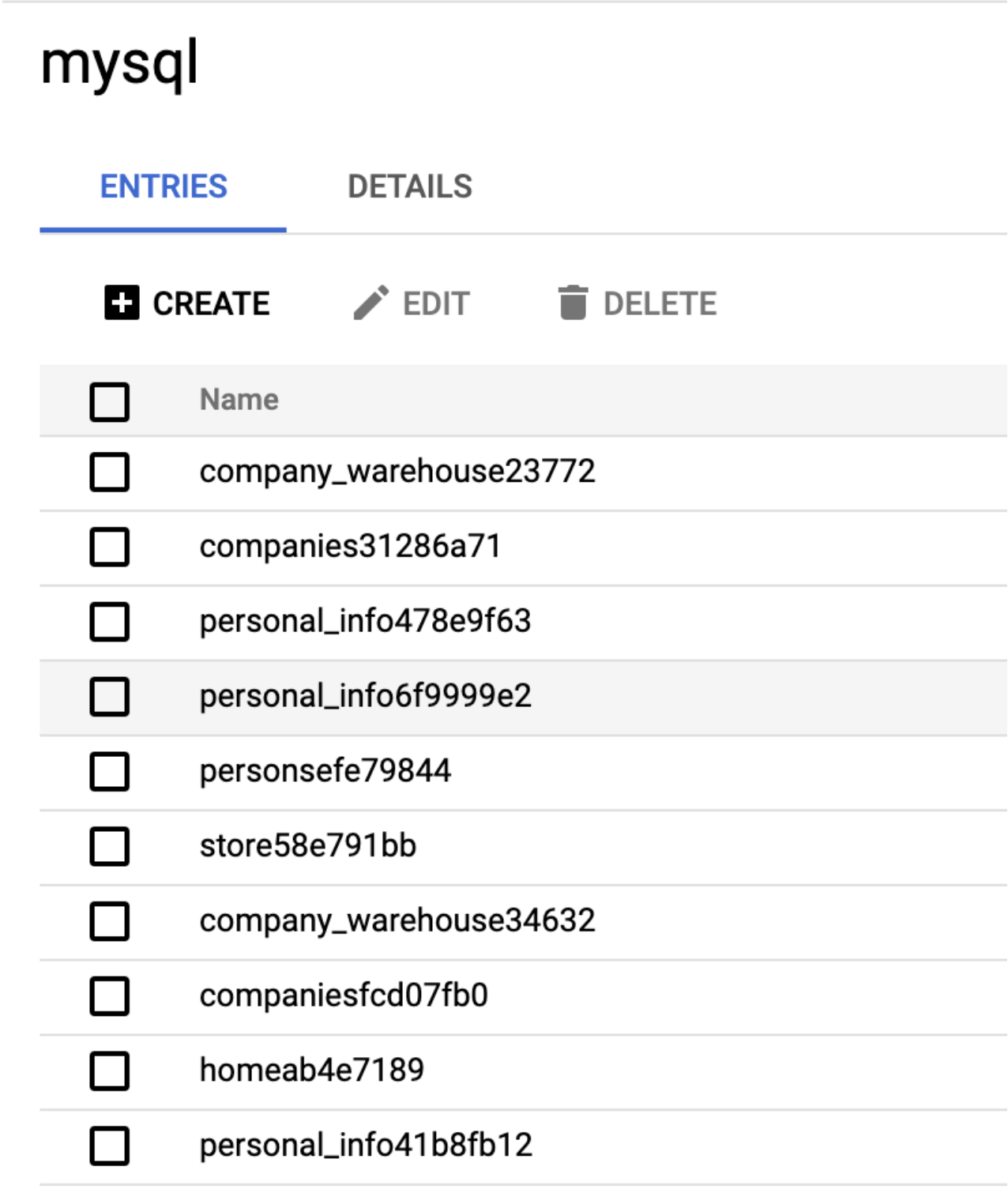
You should see the following mysql Tag Templates:



Click on **Entry groups**.

You should see the following mysql Entry Group:

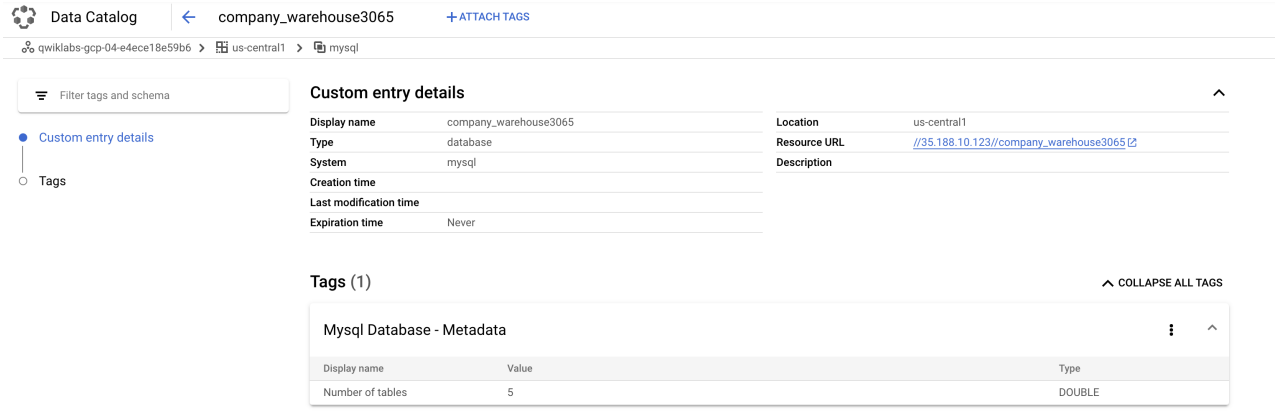
Now click on the `mysql` Entry Group. Your console should resemble the following:



This is the real value of an Entry Group — you can see all entries that belong to MySQL using the UI.

Click on one of the `warehouse` entries. Look at the Custom entry details and tags:

Custom entry details and Tags:



This is the real value the connector adds — it allows you to have the metadata searchable in Data Catalog.

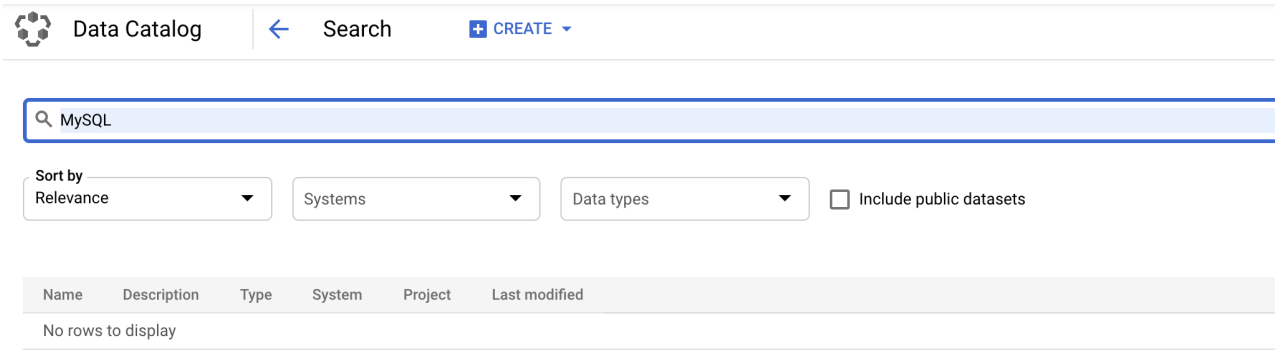
Cleaning up

To delete the created resources, run the following command to delete the MySQL metadata:

./cleanup-db.sh
Now execute the cleaner container:

docker run --rm --tty -v \ "$PWD":/data mesmacosta/mysql-datacatalog-cleaner:stable \ --datacatalog-project-ids=$PROJECT_ID \ --rdbms-type=mysql \ --table-container-type=database
Finally, delete the PostgreSQL database:

./delete-db.sh
From the **Navigation menu** click **Data Catalog**. Search for **MySQL**. You will no longer see the MySQL Tag Templates in the results:



Ensure you see the following output in Cloud Shell before you move on:

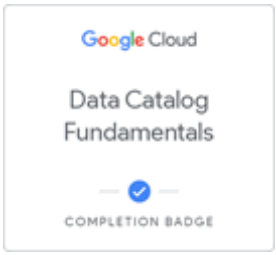
Cloud SQL Instance deleted COMPLETED

Congratulations!

Great job! You received hands-on practice with Data Catalog connectors.

In this lab, you learned how to:

- Enable the Data Catalog API.
- Create a dataset.
- Copy a public New York Taxi table to your dataset.
- Create a tag template and attach the tag to your table.



Finish Your Quest

This self-paced lab is part of the BigQuery for Data Warehousing, BigQuery for Marketing Analysts, and Data Catalog Fundamentals Quests. A Quest is a series of related labs that form a learning path. Completing a Quest earns you a badge to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in a Quest and get immediate completion credit if you've taken this lab. See other available Quests.

Next Steps / Learn More

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated March 31, 2022

Lab Last Tested February 22, 2022