Starburst 101: Hands-on Lab

Starburst 101 Hands-on Lab

Table of Contents

Starburst 101 Data Engineer/Data Administrator

Lab 1: Introduction and setup

Learning objectives

- Describe the lab scenario and goals.
- Demonstrate how to set up a free Starburst Galaxy account.
- Understand how to continue using Starburst Galaxy after the end of the lab.

Activities

1. Lab overview
2. Create a Starburst Galaxy account
3. Housekeeping items

Part 1: Lab overview

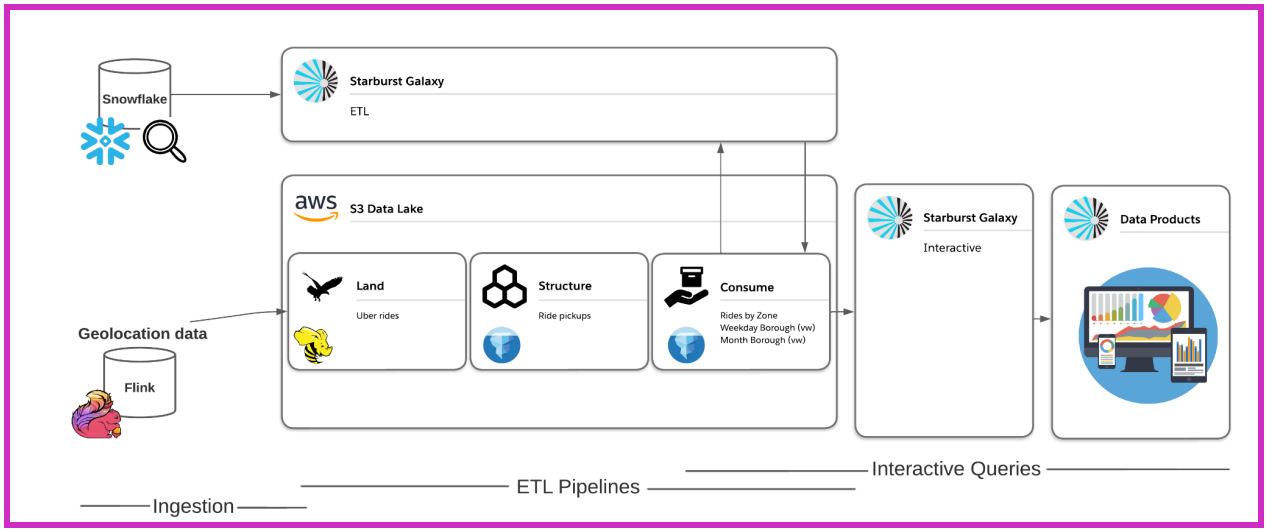
You are a data engineer at Uber. The data science team is running into an issue with not having enough data to train their models. The marketing team doesn't know where to advertise in NYC. You need to help both teams by discovering, transforming, and cleaning the data from multiple sources. You will create a data product for each specific team.

Step 1 - Purpose of lab

This lab uses Uber pickup data from New York City, January to June 2015 and is ingested into AWS S3. The dataset contains information about each pickup, including pickup_time, location_id, dispatching_base, and affiliated_base. Because you will be simulating near-real-time pickup information, you will notice many rows. To help manage this, the ingested data is partitioned in AWS S3 by both year and month. Importantly, you do not have any information about the specific location statistics. This is contained in separate snowflake lookup tables. This information is derived from each zone and is recorded by location_id.

Step 2 - Description of activities

To make sense of data from multiple sources, you will need to federate across Snowflake and AWS S3. To do this, you will begin by creating a reporting structure in your data lake. You will use Starburst Galaxy to read the data in the Bronze layer, then clean and optimize that data into more performance Parquet files in the Silver layer. In the last step, you will join the Location information from Snowflake with the geolocation pickup information in AWS S3 into a single table that is cleaned and ready to be utilized by our teams. After completing the discovery, location, governance, and query stages, you will end the lab by creating data products, which package the dataset in a curated way for easy consumption.



Step 3 - Data challenge

This data challenge involves two key missions:

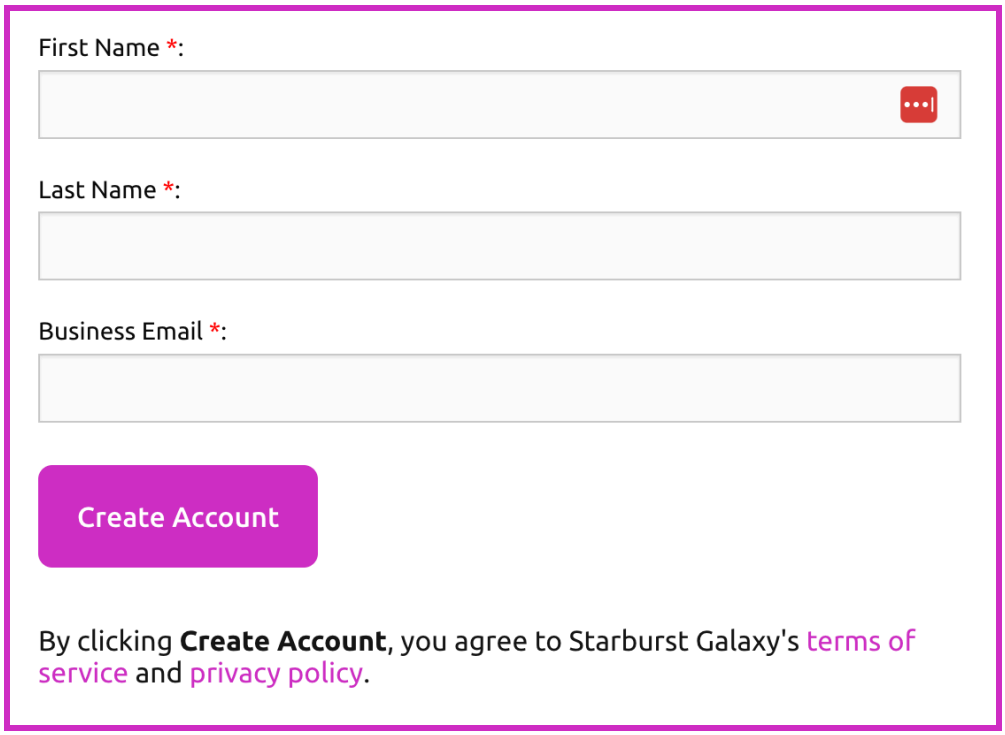
- Create a final table output for our data scientists to run models.
- Answer the following business question from the marketing department. This will be used to plan marketing campaigns that generate more business.
    - **Question:** What is the most popular day of the week for each borough?
    - **Objective:** Plan marketing campaigns for each borough based on popularity.
    - **Approach:** You create a specific view for each borough and day.

Part 2: Create a Starburst Galaxy account

Now it's time to create a Starburst Galaxy account. If you already have one, you can use that. If not, follow the steps below.

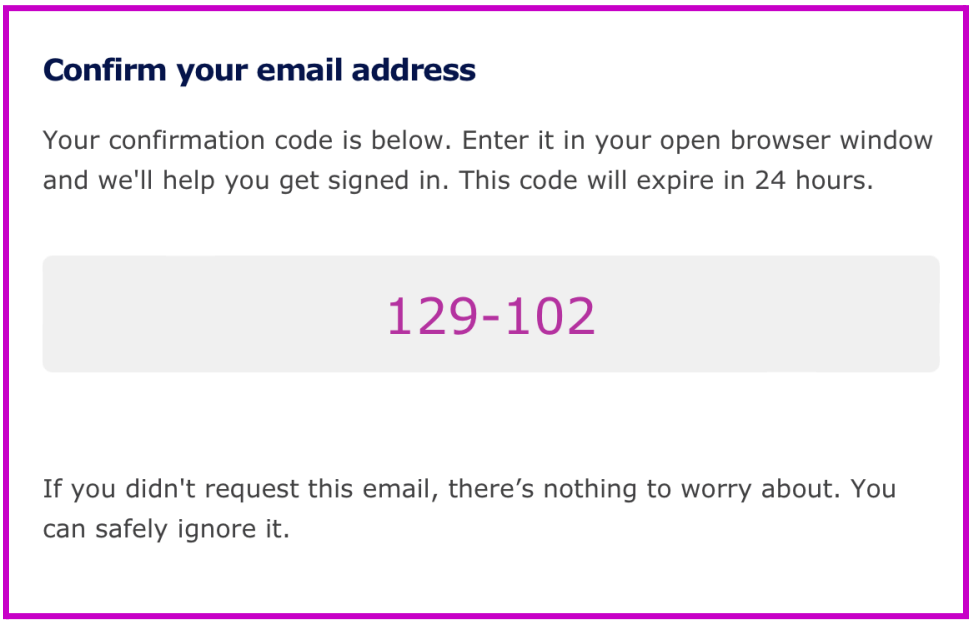
Step 1 - Create a Starburst Galaxy domain

Navigate to the Starburst Galaxy homepage. Enter your First Name, Last Name, and Email address. Click the **Create Account** button.

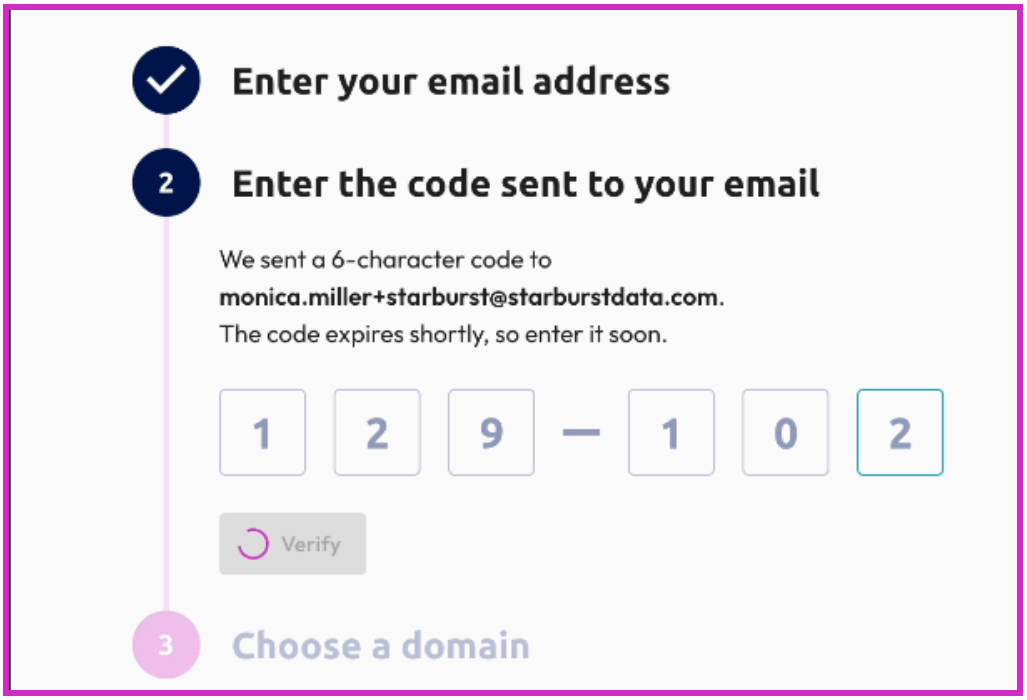


Step 2 - Enter your confirmation code

Go to your email inbox and get the code you were just sent from the subject line of the email.

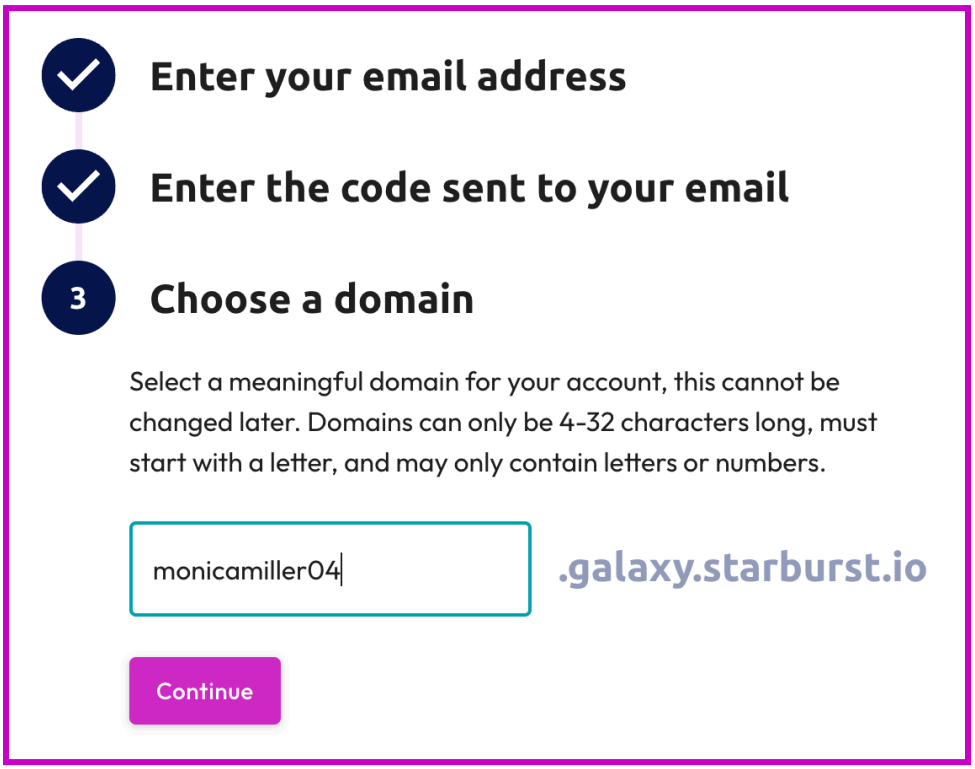


Go back to the Starburst Galaxy login, and enter the code. As soon as you type the last digit the verification process will start.



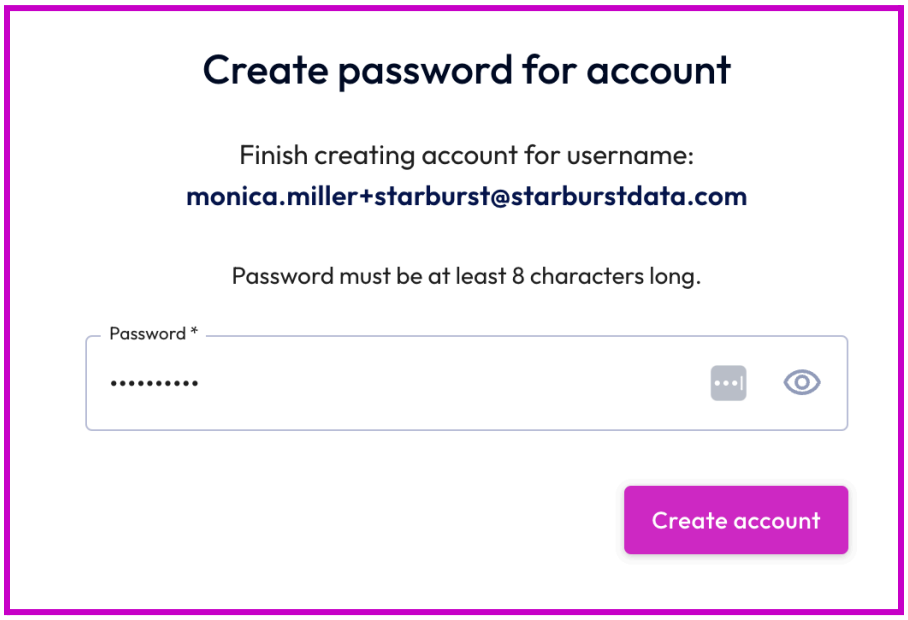
Step 3 - Choose a domain name

Select a meaningful domain name according to the guidelines provided. Click the **Create account** button.



Step 4 - Create a password

Enter a memorable and meaningful password. Click the **Create account** button.

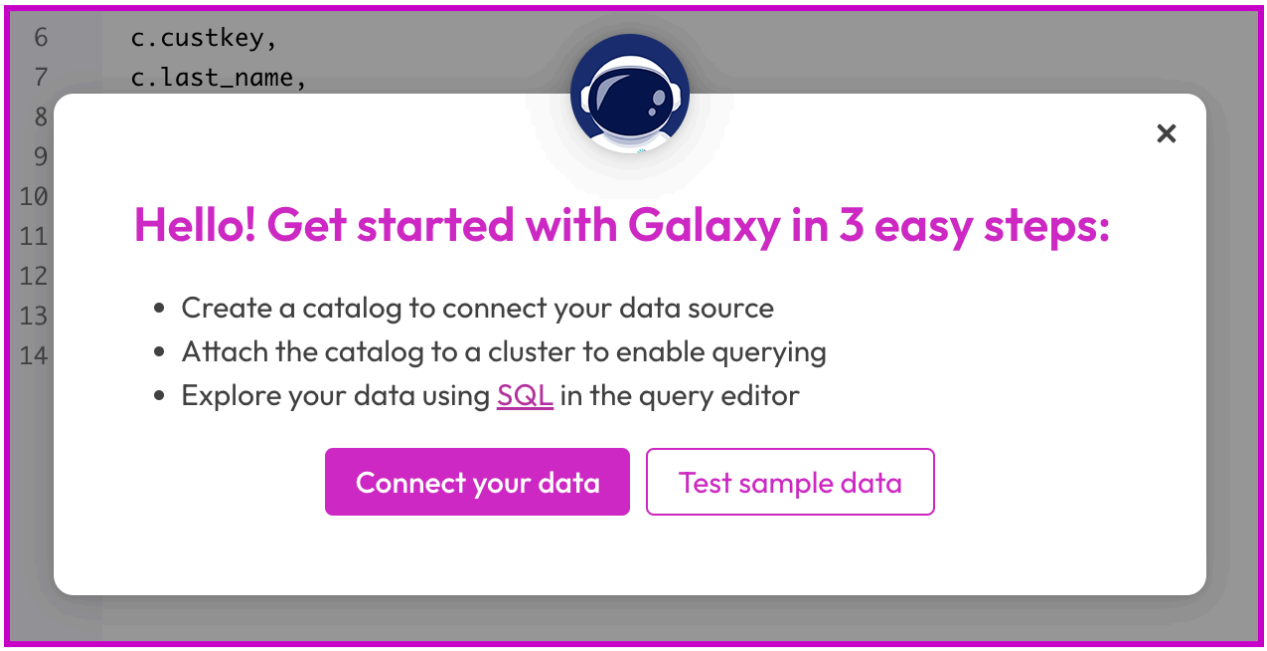


Step 5 - Answer the entrance survey

Read through the introductory onboarding information. Provide answers accordingly. If you do not know what to answer for a sign-up reason, put **Starburst training or certification**.

Step 6 - Navigate around the UI

Close the in-app product tour as you will go through this throughout the lab.



You will land on the query editor - let the lab take it from here.

Part 3: Housekeeping

As part of Starburst 101, the credentials for Amazon S3 and Snowflake will be available for one week. It is critical to understand that after 11/20/2024, **YOU WILL BE UNABLE TO RUN ANY QUERIES AGAINST THE TWO CATALOGS CREATED IN THIS LAB**.

If you want to continue exploring Starburst Galaxy, here are some other free projects and helpful links you can utilize with your Starburst Galaxy account:

- Federate multiple data sources tutorial
- Starburst Academy
  - Starburst Galaxy courses
  - Data foundations courses
  - Learn SQL courses
  - Starburst foundations
- Starburst Galaxy documentation
- Near Real-Time Ingestion tutorial

For questions about the lab material, please reach out to monica.miller@starburstdata.com.

END OF LAB EXERCISE

Lab 2: Connect to data sources

Learning objectives

- Describe the process for creating catalogs that connect AWS S3 and Snowflake.
- Demonstrate how to create a cluster in Starburst Galaxy.

Activities

1. Create Amazon S3 catalog
2. Create Snowflake catalog
3. Create a cluster

Part 1: Create Amazon S3 catalog

Objective

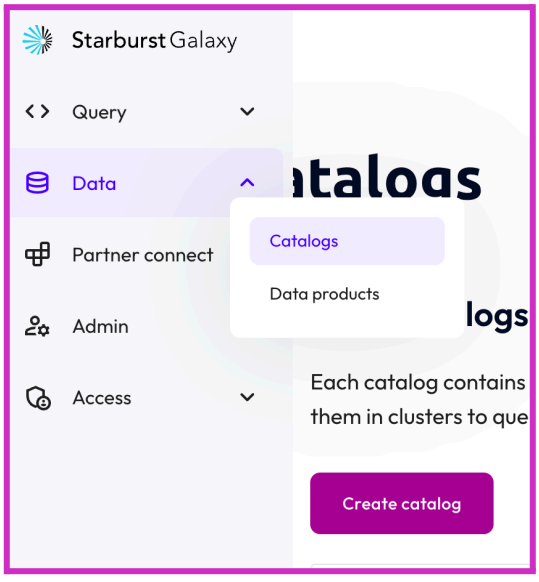
You're going to begin by setting up an AWS S3 catalog in Starburst Galaxy and connect the Uber ride share data.

Step 1 - Sign in and verify your role

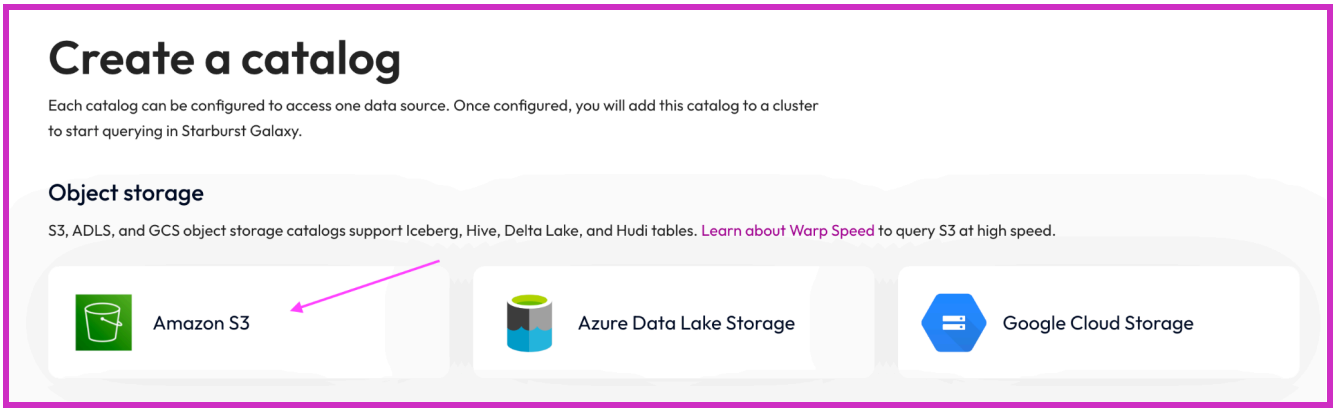
Sign in to Starburst Galaxy. Use the account credentials you previously created. In the upper right corner of the screen, confirm that your role is set as `accountadmin`.

Step 2 - Create Amazon S3 catalog

Navigate to the **Catalogs** pane on the left-hand side of Starburst Galaxy. Select the **Create catalog** button to create your first catalog.

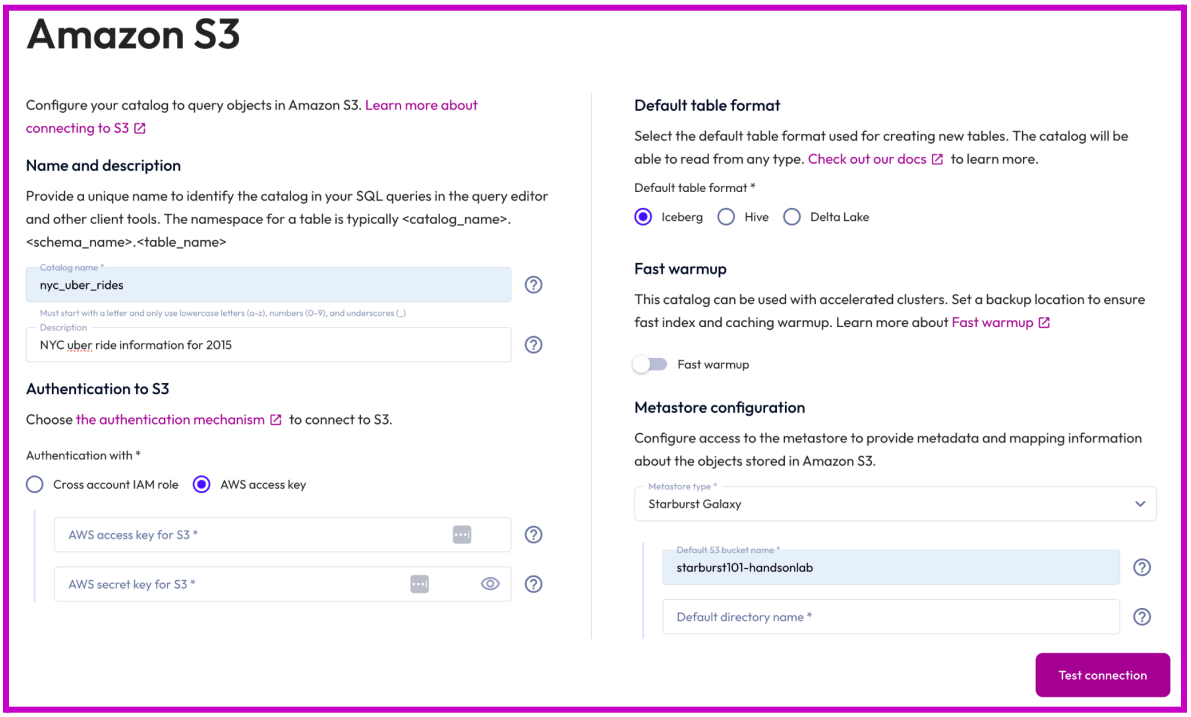


Select **Amazon S3** as your first data source.



Use the information below to configure your catalog. It will query objects in Amazon S3, specifically the NYC rideshare data in your data lake. Provide the necessary credentials to authenticate the connection.

Catalog Name: nyc_uber_rides
Description: NYC uber ride information for 2015
Authentication with: AWS ACCESS KEY (select the radio button AWS access key)
Access Key: AKIAYUW62MUVYUC3IVOB
Secret Key: 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
Default table format: Iceberg
Metastore type: Starburst Galaxy
Default S3 bucket name: starburst101-handsonlab
Default directory name: your-name (ex: dennis-mcquilken)
Allow creating external tables - YES
Allow writing to external tables - YES

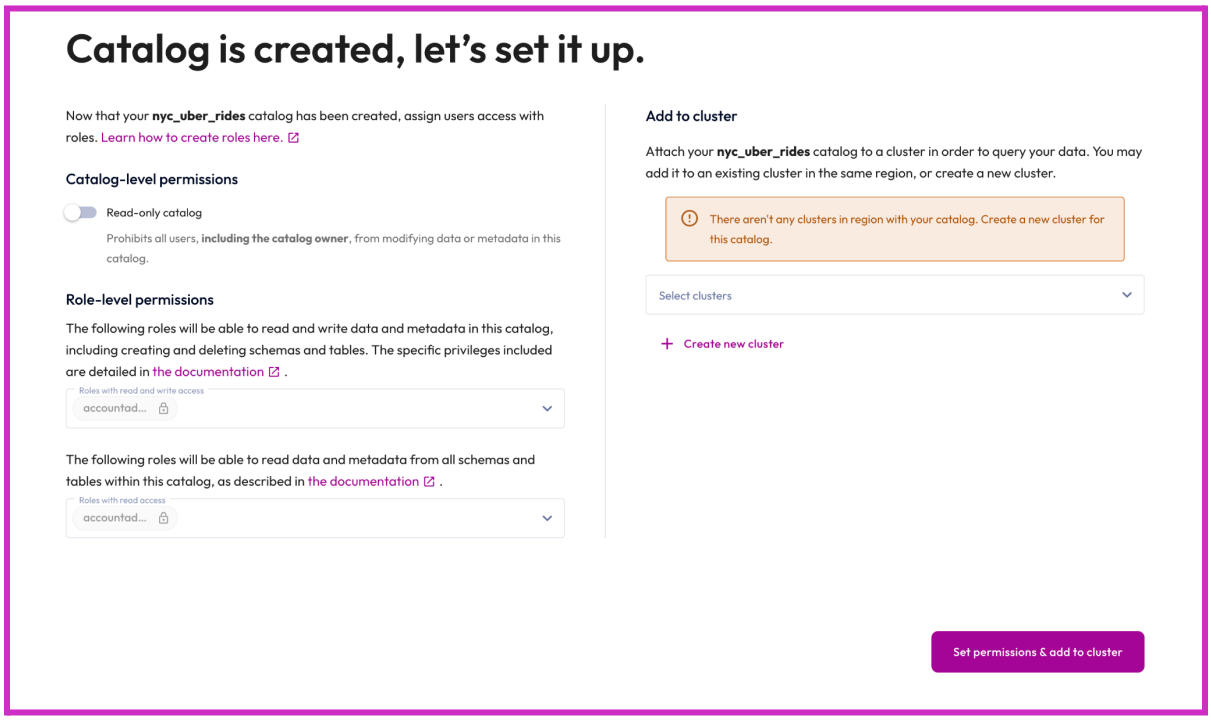


Validate the connection by hitting **Test connection.** Your catalog should return the same message indicating that you can now add the catalog.

Select **Connect catalog**. This will save the credentials for your Amazon S3 catalog.

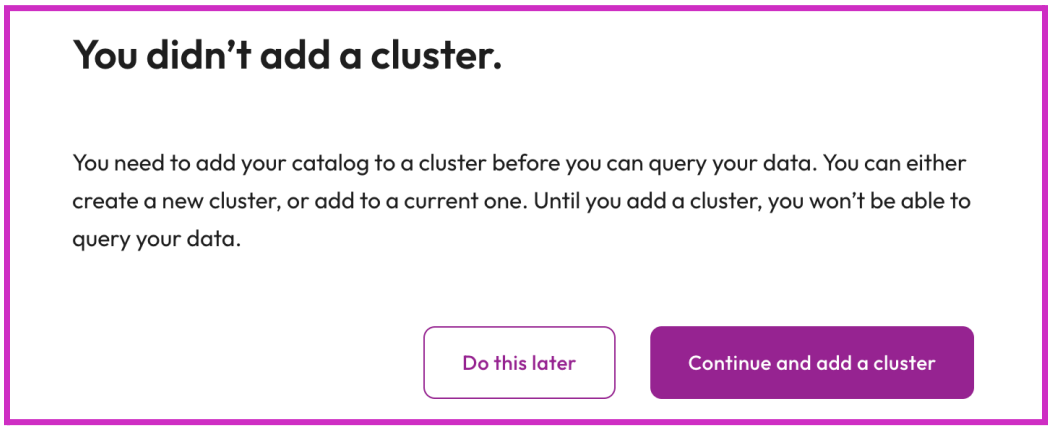
Step 3 - Set permissions

Next, accept the default permissions for your catalog by selecting the button **Set permissions & add to cluster**.



Step 4 - Skip Add to cluster

Finally, you will be asked if you want to add your newly created catalog to a cluster. Select **Do this later**. You will create a cluster later in the lab exercise.



Part 2: Create Snowflake catalog

Objective

Now it's time to create a Snowflake catalog alongside your AWS S3 catalog. Later, this will allow us to federate across the two data sources.

Step 1 - Create Snowflake catalog

From the catalog page, select the **Create catalog** button to create your second catalog. Choose **Snowflake**.

Using the list below as a guide, configure your catalog to query objects in Snowflake, specifically the zone information. Provide the necessary credentials to authenticate the connection.

Cloud Provider: AWS
Catalog Name: taxi_zone_lookup
Description: Taxi zone lookup information for NYC
Snowflake account identifier: TB03263.us-east-2.aws
Authenticate with: Key pair
Username: PUB_KEY_USR

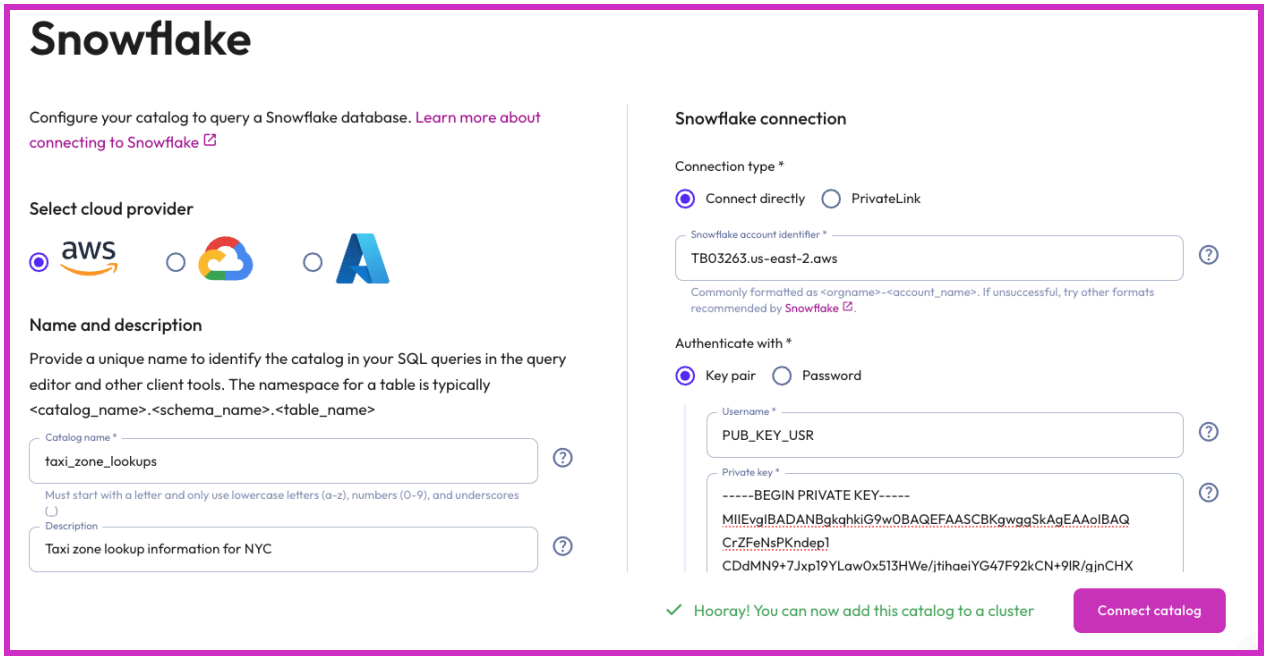
Private key:

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCrZFeNsPKndep1
CDdMN9+7Jxp19YLaw0x513HWe/jtihaeiYG47F92kCN+9lR/gjnCHXiAu4YwpZ5V
QJSGVIFhjUyO3FD+9mdNgMR6jU2VC2X/Sbt9AQelj8wZtl4woUkSphr9QVFTswau
Io13241ndcNFkTtYEGth9zBvN1xPFDu1/d8Rdap3bmD3CIERq2WcaIZZPvnO8tF+
WTYe2vTmIXmQkOI96pK2WuB0PxVbrrFtMbNHzj67MorskjkPidDoUeKF342oxU/K
IryHqpClwQgZ5mPLTaQZ24B+D2x/+MmrHIRhmh7npTa3JqlcZEDF2WF/3L4z7aaG
duGwHLe9AgMBAAECggEAAy8UaEzUQAXubh3ta5/fqvMzHy0SlbMBKIPbizOpco/w
Cka0KtOZAtOpuknj+VDZrec6Upmbf4bsrfSvWRXquhjz6r4zzkcCnsTIpi91mRtk
/lqxHiDm8qRhroNQPfwZnXMcAFDENcBynEkrZNPv1lzwgVNNP9CF/HG7cJKd0FfO
D9pgngw554a5eiCxb3/DQFfBUEfdol88n0/Xm5EozrBTePxxI77u1+obAjhUkGhT
A7aGvPJdajfxP4oZdDkS/OFeLhHYN6Uj67heuohdsccuxqZZ2tz+Y3t5YBckUK1S
s6xY7Z5rY3NnLcOju9R/SRicErElAfS2GCyVR6tVsQKBgQDTzRK/oafrD89n2UBV
xHE2wAK+BKco2epKnDv9tV7Yun66AOm8uqBSBxy7fGCGG/x7VcD0rIThpUP6p6aF
KvF+yaJJ2d0GIBbRVIoxWBgCsY38a0K7K6L9xFZDEo3E42qZaUyUzVT3V95/D100
L7hUAG40kJ+cmD5PbEEUKZ7GcQKBgQDPKIQrmIJpkVkd3r8ug1LK3IQxtvxUY0+l
PCYGEC7HSiKD7ZpWp51OisVnWoV06DshaMzNSUvECrhEfA8wC0OqAEWPnP4izNzQ
uHJWWkaLq7IukfZ/bsW6whSOUvA9npGrH/xIOz52OTmgf2pHjyLPXanU1ZHcAlef
jO/YF1DkDQKBgQC8bPUBBxAcTzR1bxGNO/Z8ftXxtrH/5d5KAjRcwb5fkYqsb6OC
A4Ydc7ZcFYrJxJwaHXBMPC77j8uqcvvx/QJbiYaXrPq1OFcUY+GNhJHX0549si7r
1SZsHP+hLCBzyN5NHSDEQOgWJFYNM933T3ztPnTMzt0ws0C759L93QkkAQKBgQCj
Pk5I/HloLz2GqE0HnjzT10nk00msRkYw82ETThVOvkDMraP9vT/c8MX2WALqyUYv
YaJh8uxdpATpEsn1FitP/77RGOm7CqzHXVHccm89CSP0Rtl0OhpevSntrbXbGOW5
cA8h7G2gRCABNebOVSEdP7XhHhs+wCnEdQ9bvaPk0QKBgF03ZzVrZzjj2zbLvIzY
th7vUdMpkOYcsMSohZUDMdExrXg/Al/l3ZIZER9aWRxrmjvN5rH6DklRdBgqgb92
nnbtx2MjBS+t5XMtSze5U8Xhbm1nuYGmmiNzfmcJeqi2sD2hLSi7l2vIFiXOdTmr
MqlCKelHYTIM+Wy240dzWNVJ
-----END PRIVATE KEY-----
```

Private key passphrase: *<empty sting>*
Database name: NYC_RIDESHARE
Warehouse name: SB_101
Snowflake role: STARBURST_101

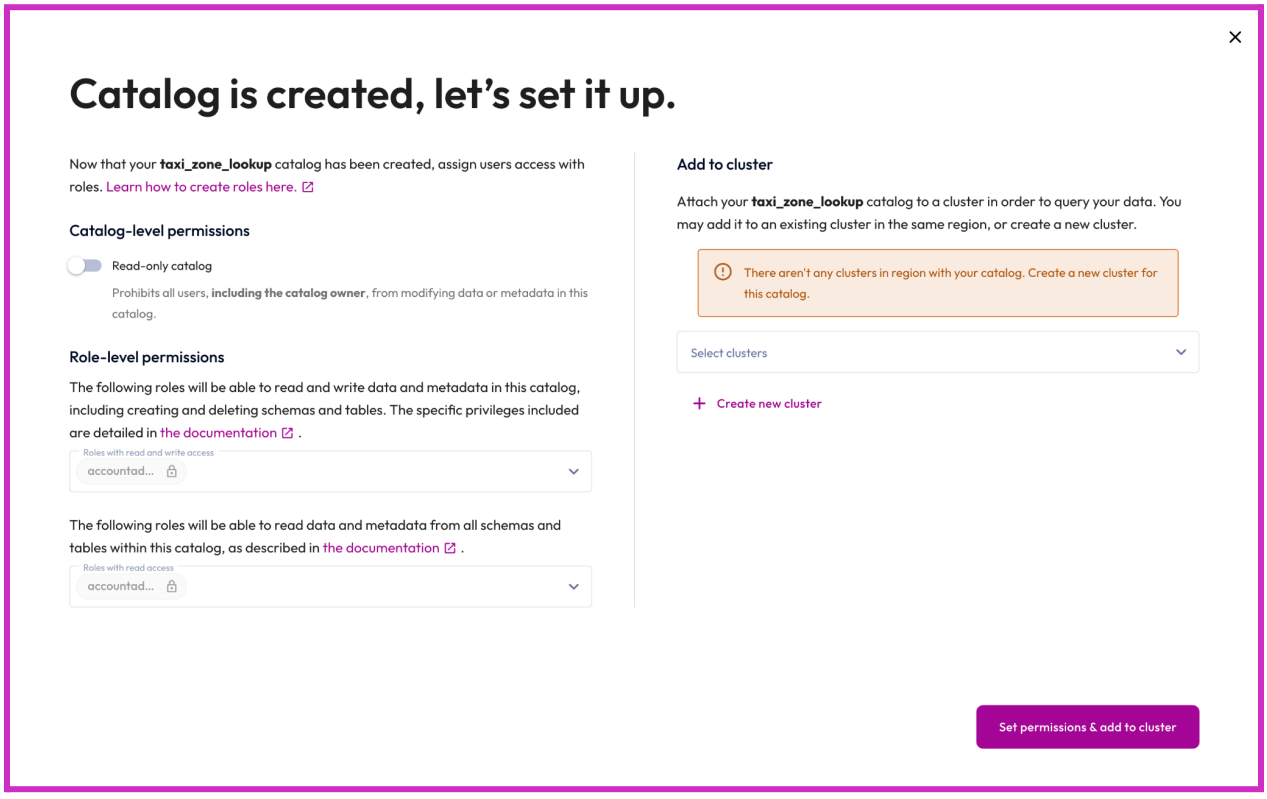
Test the connection to ensure that the setup is correct.

Select **Connect catalog** to save the credentials for your Snowflake catalog.



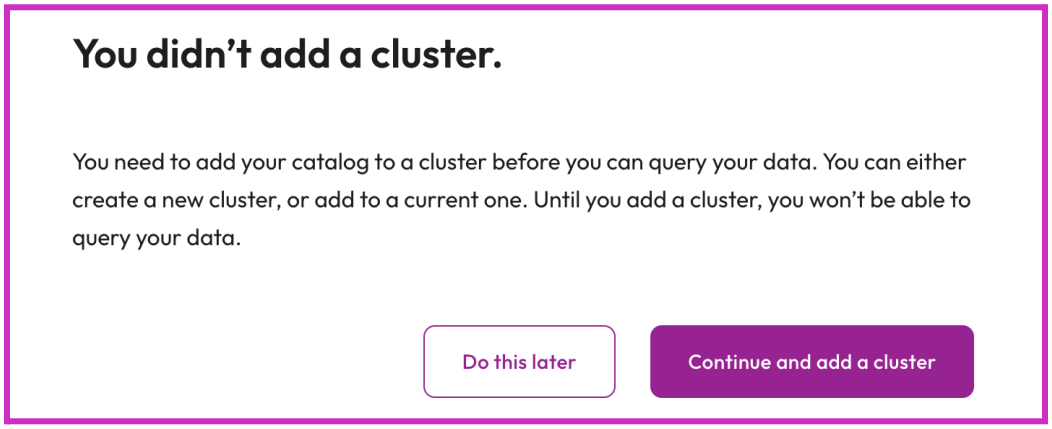
Step 2 - Set permissions

Next, accept the default permissions for your catalog by selecting the button **Set permissions & add to cluster**.



Step 3 - Skip Add to cluster

Finally, you will be asked if you want to add your newly created catalog to a cluster. Select **Do this later.**. You will create a cluster later in the lab exercise.



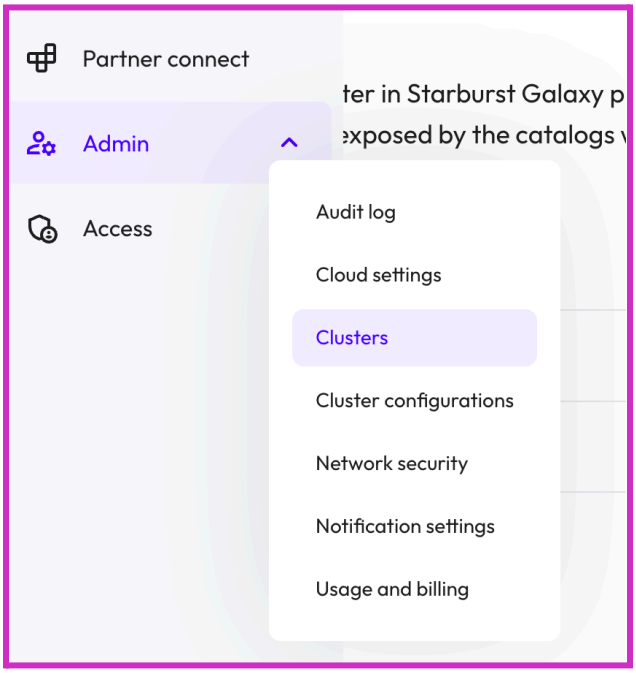
Part 3: Create a cluster

Objective

Now it's time to create a cluster to help you execute queries against your catalogs. To federate data from multiple sources, you need to add catalogs to the same cluster.

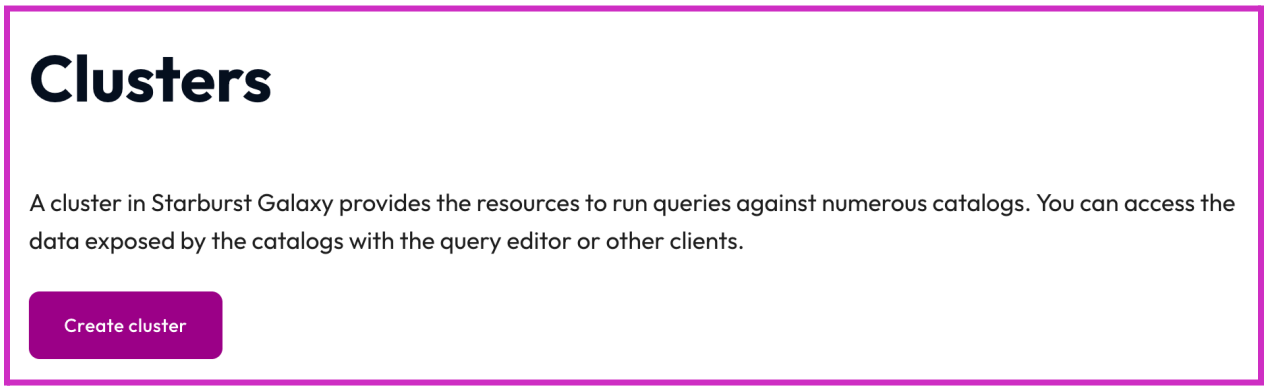
Step 1 - Navigate to the Clusters pane

To begin, navigate to the Clusters pane on the left-hand side of your Galaxy browser window.

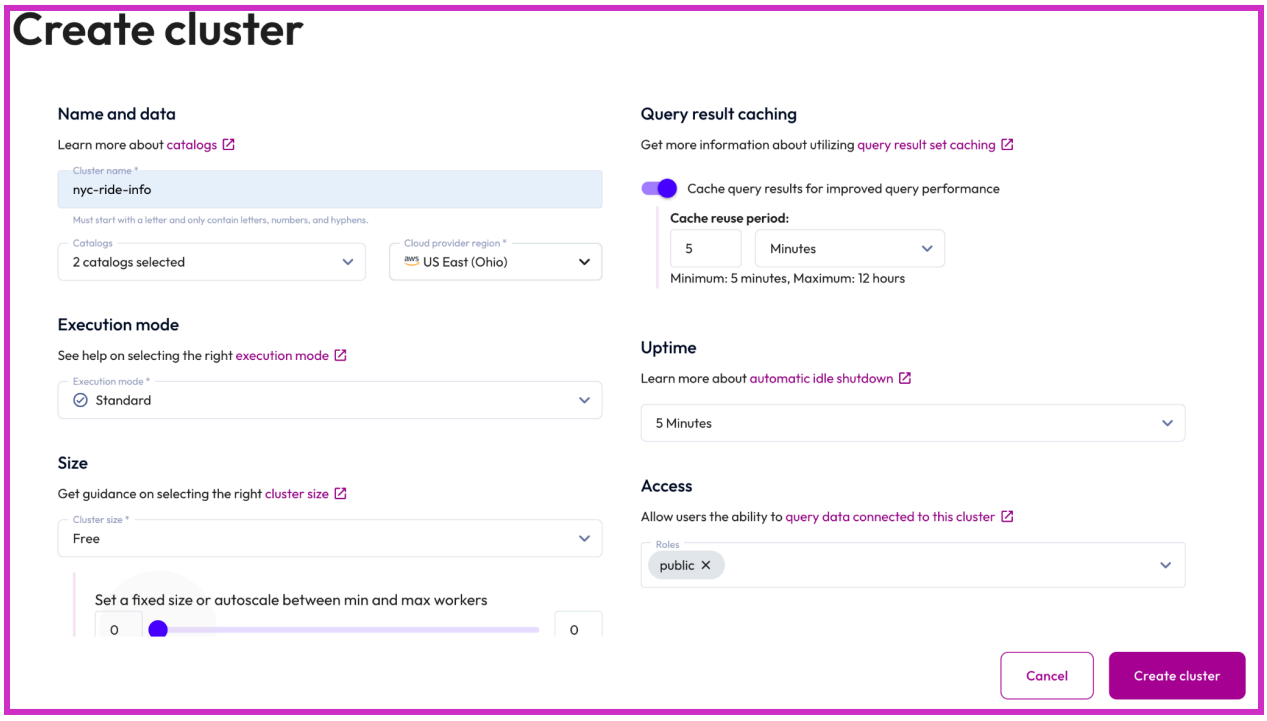


Step 2 - Create cluster

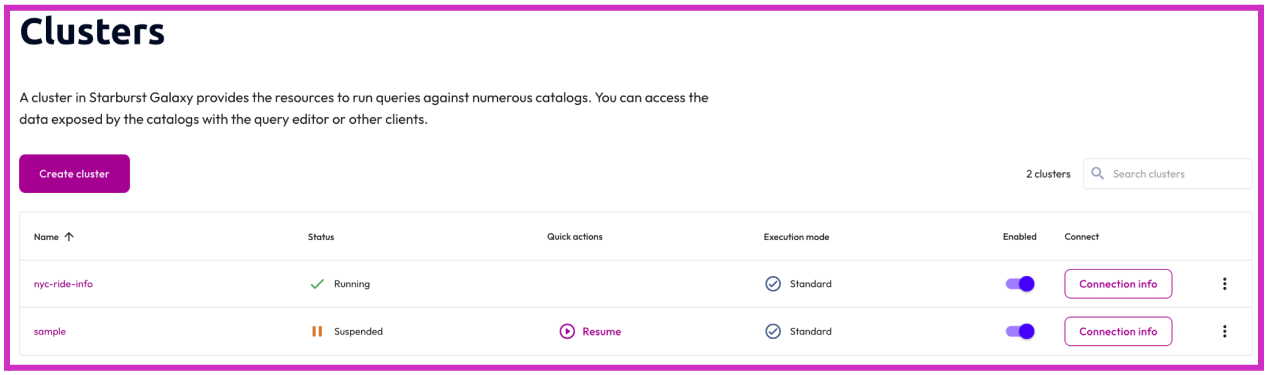
Select **Create cluster.** Populate the cluster with the information below. Use the check boxes to select the catalogs. When complete, select **Create cluster.**



Cluster name: nyc-ride-info
Catalogs: nyc_uber_rides & taxi_zone_lookup
Cloud provider region: US East (Ohio)
Execution mode: Standard
min/max workers: Edit the text boxes to both show 0 (sets **Cluster size** as Custom)
Cluster size: Free (Can upgrade to anything you like if you are just signing up)
Idle Shutdown time: 5 Minutes



You will now see your newly created cluster, similar to the image below.



Part 4: Use Schema Discovery to Ingest your Raw Data

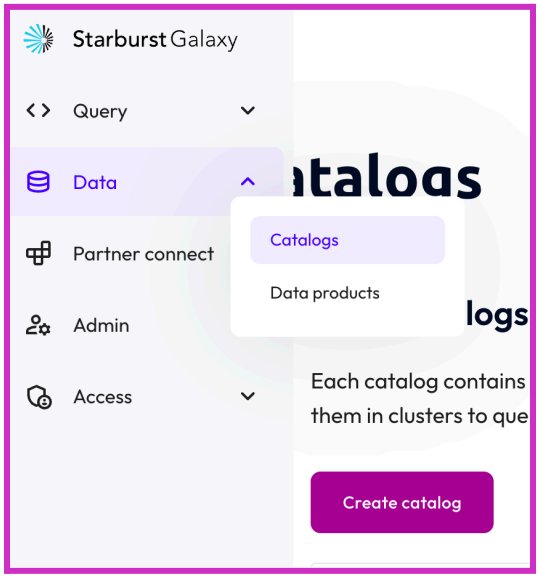
Objective

One of the most difficult parts of data engineering is discovering where your data lives, or convincing someone to let you view the data so you can accurately build the data pipeline requested by another team. Starburst Galaxy utilizes schema discovery so that you can automatically discover and access the data located in your data lake.

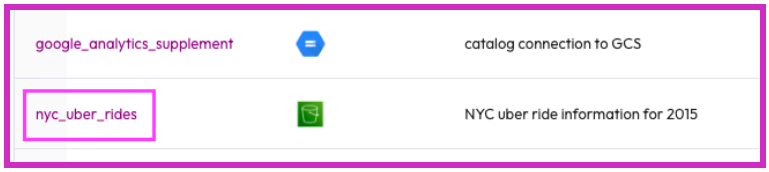
In this section, you will use schema discovery S3 object storage location so that you can execute queries within the data lake. Schema discovery will act as the Bronze layer, our raw data layer that is the first level in our data lake reporting structure.

Step 1 - Navigate to your newly created AWS catalog

Navigate to the **Catalogs** section on the left-hand side of Starburst Galaxy.



Click on the **nyc_uber_rides** catalog to navigate to the catalog metadata page.

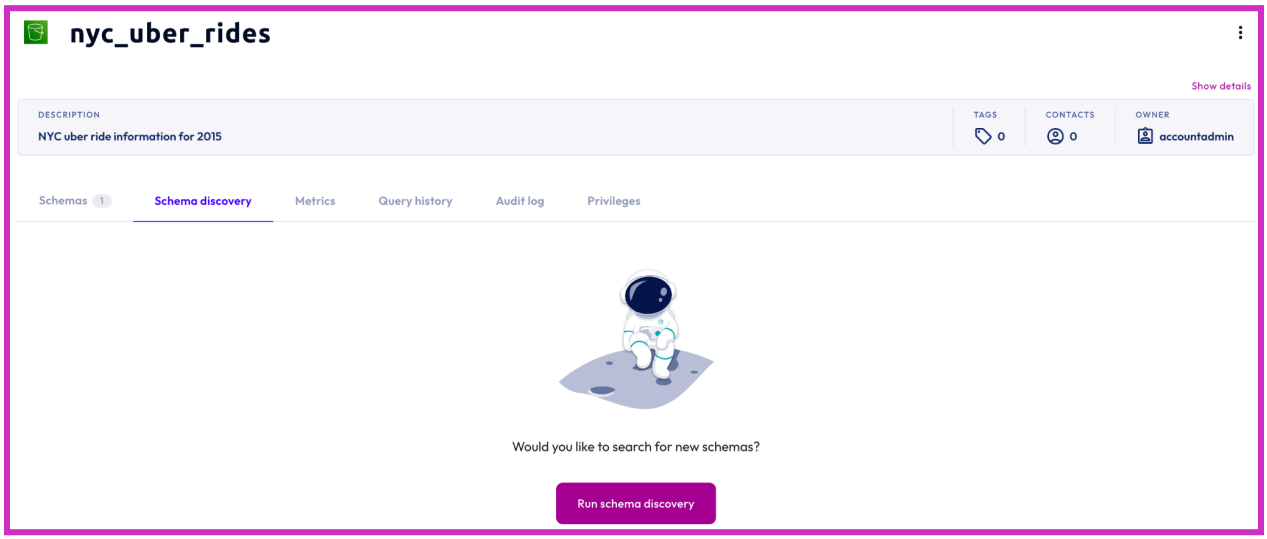


The catalog metadata is critical for storing all the information surrounding your data. This information will help you create a well informed, curated data product later in the lab. It is an important data practice to always provide as much information as possible to the data you are discovering, transforming, and curating.

Step 2 - Run Schema discovery

As part of Gravity, you can see all the metrics, schemas, query history, audit log, privileges, and more within the catalog metadata!

Click on the **Schema discovery** tab. Hit the **Run schema discovery** button.

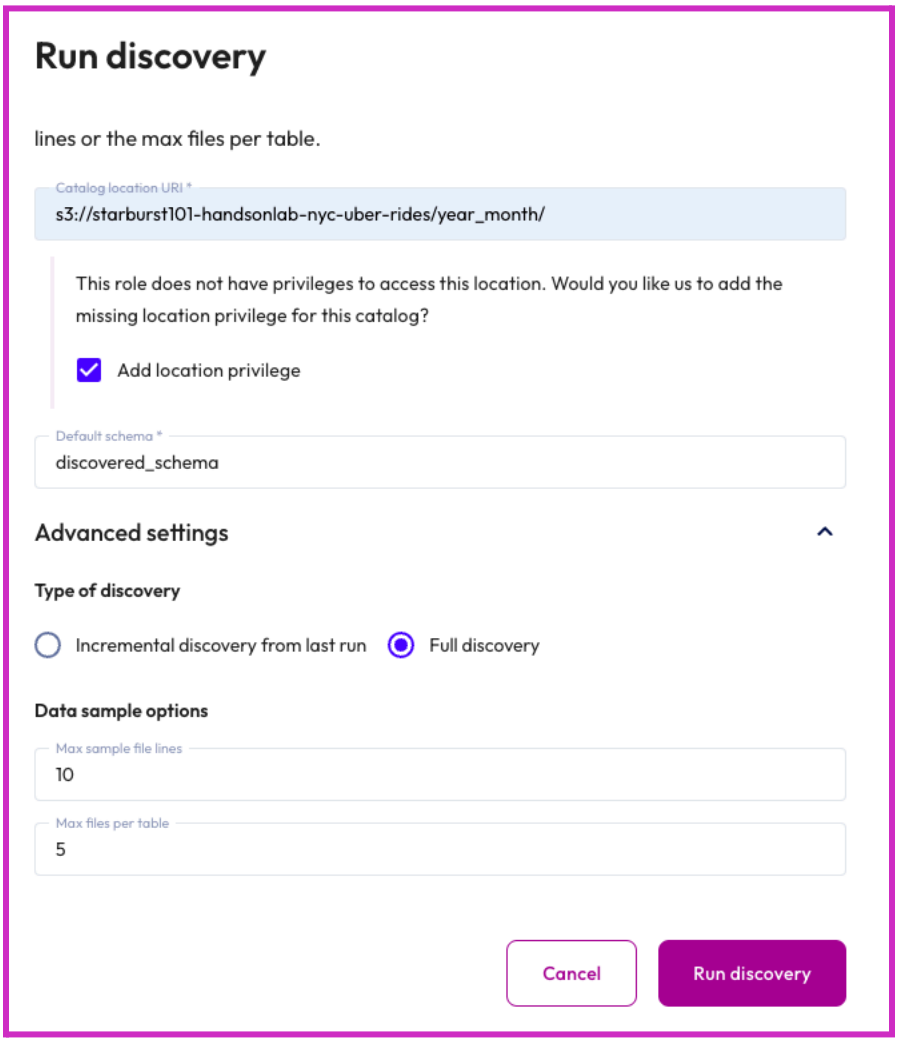


The Schema discovery pane lets you examine the metadata of the specified object storage location. Schema discovery is for catalogs in object storage data sources only.

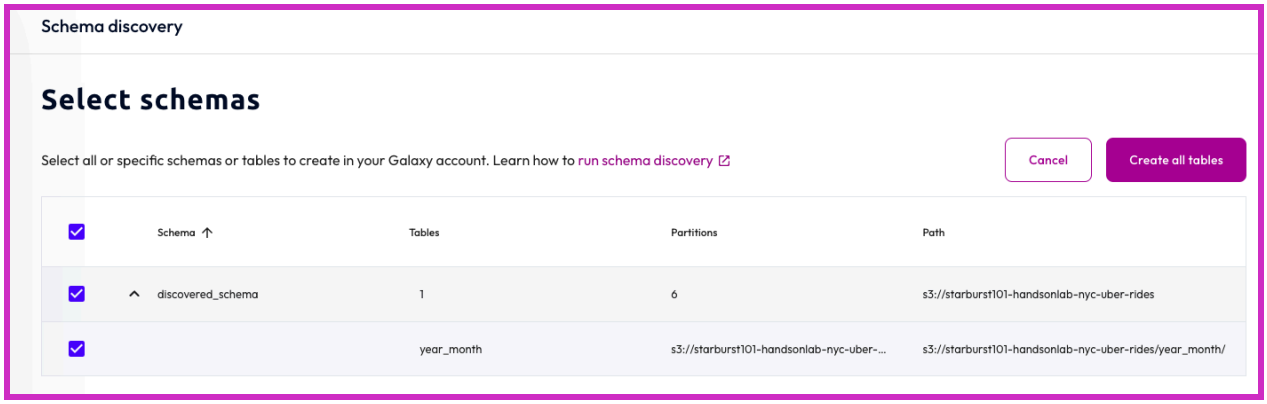
Use schema discovery to identify and register tables or views that are newly added to a known schema location. For example, a logging process might drop a new log file every hour, rolling over from the previous hour's log file. The purpose of schema discovery is to find the newly added files to make sure Starburst Galaxy knows how to query them.

Enter the following information, including the catalog location URI, the default schema name, the type of discovery.

Catalog location URI: `s3://starburst101-handsonlab-nyc-uber-rides/year_month/`
Check the Add location privilege checkbox? `YES`
Default Schema: `discovered_schema`
Type of discovery: `Full discovery`

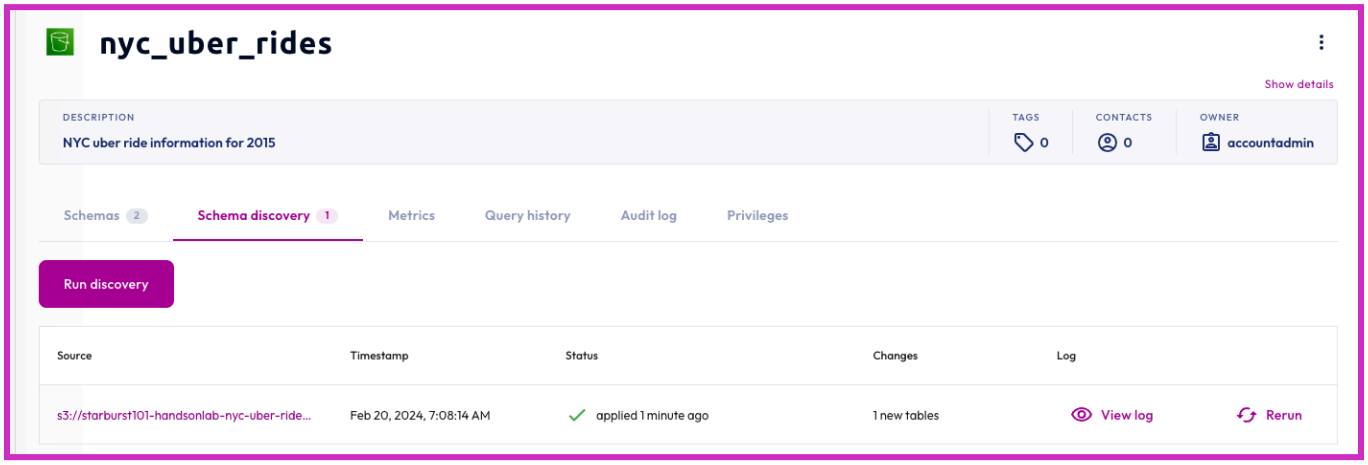


Hit **Run discovery.** Starburst will start scanning for you. Then, it will return code to create your desired schema and table. Notice the 6 partitions that will create the year_month table in the discovered schema. Select all instances and then hit **Create all tables**.



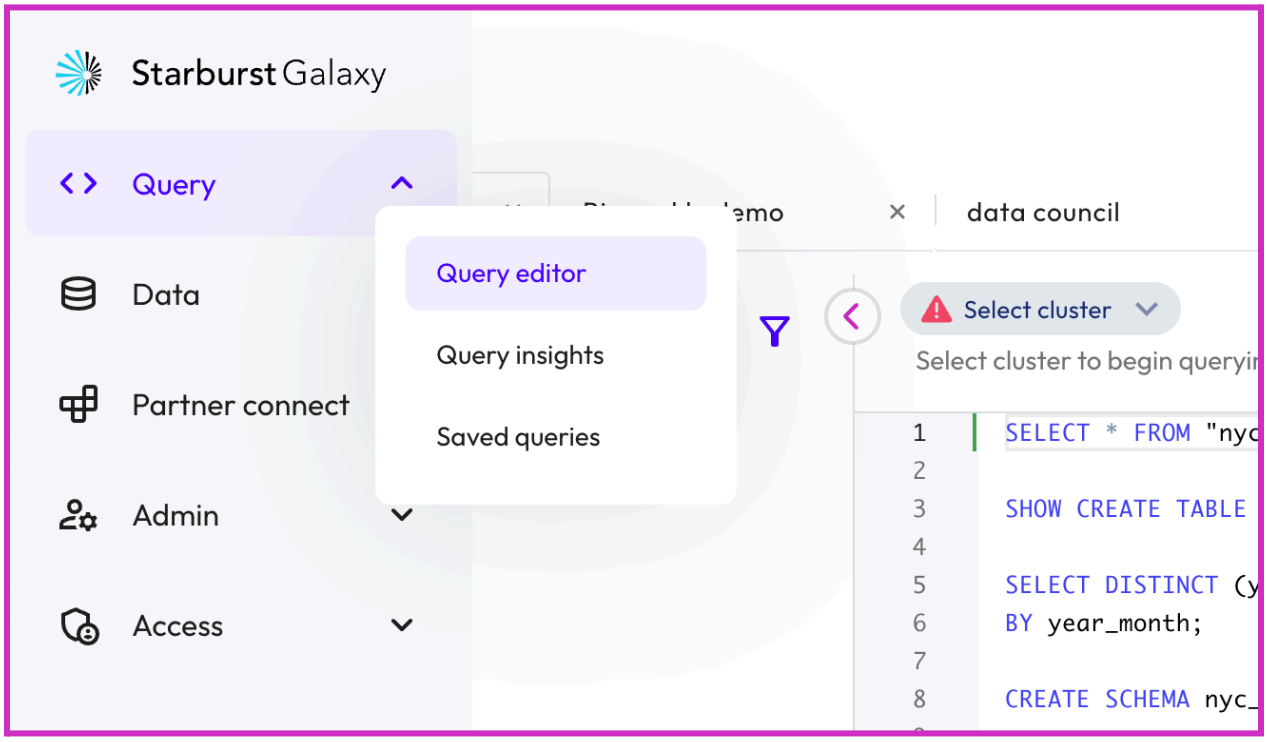
Schema discovery has done the heavy lifting so you don't have to spend time trying to investigate what columns exist, or bother your AWS administrator to give you details about the file.

Click on the Query text to view the full queries. Your first query created the desired schema, and the second query created a Hive table. After you are done, Hit **Finish**. You will see a record of schema discovery in the catalog metadata.



Step 3 - Validate schema discovery

Navigate to the **Query editor**.



If you already have queries, add a new tab + using the fuschia plus sign. Change the location drop-downs in the top left hand corner to match the cluster and catalog previously created.

Cluster: nyc-ride-info
Catalog: nyc_uber_rides

Run the following query to validate the table you created using schema discovery.

```
SELECT * FROM "nyc_uber_rides"."discovered_schema"."year_month" LIMIT 10;
```

Run a command to view the CREATE TABLE statement:

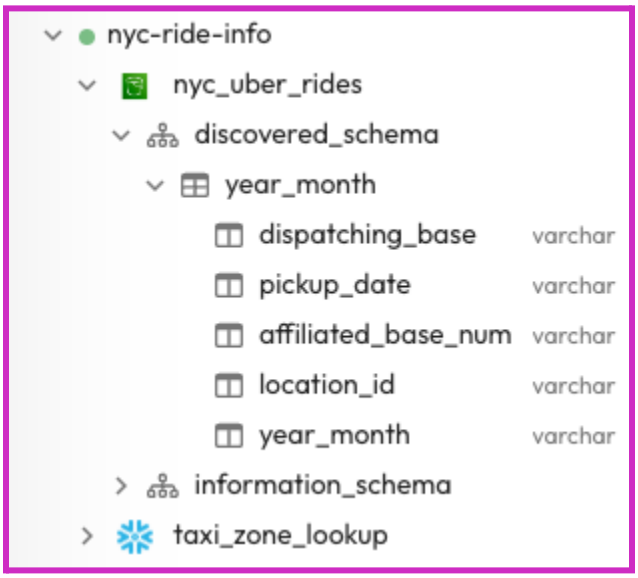
```
SHOW CREATE TABLE discovered_schema.year_month;
```

You should return the same code as run with Schema discovery. Notice that the columns are already utilizing proper data types. Also notice that the table format is HIVE. You will update this as you build your reporting structure in your data lake using Great Lakes connectivity. Recall that the default table format you set for this catalog was Iceberg. However, you are still

able to create a table with the Hive table format due to Great Lakes connectivity in Starburst Galaxy. Great Lakes connectivity abstracts the details of using different table formats and file types when using certain write access statements for object storage systems.

Step 4 - Verify schema in catalog

In the catalog explorer, expand the `nyc_uber_rides` catalog to verify your new schema and table are present. There will be other schemas listed.



END OF LAB EXERCISE

If you are unable to create your schema using schema discovery - refer to appendix section A.

Lab 3: Build within your data lake

Learning objectives

- Demonstrate the process needed to run schema discovery to analyze a root object in an object storage location.
- Show how to use open table formats.
- Demonstrate the steps needed to build a reporting structure in your data lake, and secure your team's access.

Prerequisites

- Lab 1: Introduction and setup
- Lab 2: Connect to data sources

Activities

1. Validate the Bronze layer
2. Create your schema
3. Build the Silver layer
4. Build the Gold layer
5. Secure access to your Gold layer

Now it's time to use both your data sources and create a reporting structure in S3.

- **Bronze layer** - This is the raw data you were ingesting that's landing in S3. Thanks to schema discovery, this layer is already created and all you will need to do is validate.
- **Silver layer** - This is the enriched, cleaned, and cleansed data.
- **Gold layer** - This is the data that is ready to be queried and utilized by an end consumer.

Starburst is special because it allows you to build this reporting structure not just with data that already exists in your data lake, but also with data that exists in other data sources in your orbit - like our Snowflake location data.

Part 1: Validate the Bronze layer

This layer stores raw data and is the first of the three-part data lake reporting structure. With schema discovery, the heavy lifting for building the Bronze layer is complete. All you need to do in this section is learn some basic information about the data. Run some queries to learn more about the data you will be working with.

```
SELECT * FROM discovered_schema.year_month LIMIT 10;
```

Your table should return records of a similar format as the image below.

dispatching_base	pickup_date	affiliated_base_num	location_id	year_month
B02682	2015-02-15 20:18:04	B02682	50	2015-02
B02682	2015-02-15 20:02:14	B02682	163	2015-02
B02682	2015-02-15 16:58:27	B02682	246	2015-02
B02682	2015-02-15 04:13:17	B02682	75	2015-02
B02682	2015-02-15 21:05:00	B02682	256	2015-02

Recall that your table has six partitions. It is considered best practice to check that all partitions are visible.

Use the query below to check each partition. It uses the `SELECT DISTINCT` command to validate partitions individually.

```
SELECT DISTINCT (year_month) FROM discovered_schema.year_month ORDER BY year_month;
```

Your result should resemble the output below.

year_month
2015-01
2015-02
2015-03
2015-04
2015-05
2015-06

Part 2: Create your schema

Objective

You're going to begin by creating your schema. Schemas control the structure of the data inside them. You will use this to ensure that the rideshare data is structured appropriately.

Step 1 - Select the rideshare cluster

Navigate to the **Query editor**. Add a new tab using the fuschia plus sign +. Validate the location drop-downs in the top left hand corner match the cluster and catalog previously created.

Cluster: nyc-ride-info
Catalog: nyc_uber_rides

Step 2 - Create lab schema

In the Query editor pane, open a new tab and create a new schema with the following command. Highlight and run the command below. Replace <yourname> with your actual name, or another identifier that you prefer.
Note: Only use lowercase characters and numbers; special characters or spaces are not allowed, so remember to replace the < > characters in the example below. You'll use this throughout the lab, so choose something easy to remember and type.

```
CREATE SCHEMA nyc_uber_rides.<yourname>;
```

Step 3 - Verify schema in catalog

Once this query is **Finished**, in the catalog explorer, expand the `nyc_uber_rides` catalog to verify your new schema is present and is empty. You should have 3 total schemas listed in your AWS data lake.

Note: *If you named your schema as* `yourname`*, create the schema again with the instructions above.*

Step 4 - Select schema after creation

In the top left hand corner, select your schema. Validate the location drop-downs in the top left hand corner match the cluster and catalog previously created.

Cluster: nyc-ride-info
Catalog: nyc_uber_rides
Schema: `<yourname>` `(ex: monica_miller)`

Part 3: Build the Silver layer

Objective

Now it's time to build the Silver layer. This is the second of the three data lake reporting structure layers and holds data once it has been transformed.

Step 1 - Create the Silver table

The table you just created is not yet fully optimized and transformed. For example, the CSV file format uses the varchar data type as a default for every field, even when it is not accurate. You must build a Silver layer, creating a new table and casting the necessary values into more accurate data types with the Parquet file format and Iceberg table format.

Run the following command to create the Silver layer table. Make sure you are in the schema you just created.

```
CREATE TABLE
 ride_pickups (
   dispatching_base varchar,
   pickup_date timestamp (6),
   affiliated_base_num varchar,
   location_id integer,
   year_month varchar
 )
WITH
 (
   type = 'iceberg',
   format = 'parquet',
   partitioning = array['year_month']
 );
```

Step 2 - Insert data into the Silver table

Once the table is created, you need to insert the data into it. **Note:** You can automate the process in the future using an orchestration tool like Airflow, Prefect, or Dagster.

```
INSERT INTO
 ride_pickups
SELECT
 dispatching_base,
 cast(pickup_date as timestamp (6)),
 affiliated_base_num,
 cast(location_id as INT),
 year_month
FROM
 discovered_schema.year_month;
```

You should insert 12205653 rows.

Step 3 - Validate the Silver table

It is considered best practice to validate that the new table has been created. To do this, run a simple select statement followed by a describe statement.

```
SELECT * FROM ride_pickups LIMIT 10;

DESCRIBE ride_pickups;
```

Column	Type
dispatching_base	varchar
pickup_date	timestamp(6)
affiliated_base_num	varchar
location_id	integer
year_month	varchar

Recall, this table is created as an Iceberg table. Run any of your favorite Iceberg statements to reveal statistics on your newly created table.

```
SELECT * FROM "ride_pickups$snapshots";
SELECT * FROM "ride_pickups$partitions";
SELECT * FROM "ride_pickups$history";
```

If you're looking to learn more about Iceberg, check out this awesome hands-on Iceberg lab!

Step 4: Federate your data

Run the following query to federate the data between S3 and Snowflake. You will do this before creating the final table so that you can see Starburst's ability to access data in and around the data lake.

```
SELECT
  p.dispatching_base,
  p.pickup_date,
  p.location_id,
  l.borough,
  l.zone
FROM
  ride_pickups p
INNER JOIN taxi_zone_lookup.taxi_zones.zone_lookup l ON p.location_id
= l.location_id;
```

Here you can see the query output is derived by federating two different data sources: S3 and Snowflake, which is evident by the join in the SQL statement. Anything with the **p.**prefix is pulled from S3, while the **l.**prefix fields are from Snowflake. This is quite impactful to run interactive queries and federate data from multiple different data sources.

Part 4: Build the Gold layer

Objective

Now it's time to construct the last of the three layers of the data lake reporting structure, the Gold layer. This will ready the layer for final consumption by data consumers.

Step 1 - Create the Gold table

Run the following query to create the Gold table, This constructs a new table from two separate data sources–Amazon S3 and Snowflake.

Note: Notice that the weekday and month columns are derived from the pickup_date column.

```
CREATE TABLE
 rides_by_zone
WITH
 (type = 'iceberg', format = 'parquet') AS
SELECT
 p.dispatching_base,
 p.pickup_date,
 p.location_id,
 date_format(p.pickup_date, '%W') weekday,
 date_format(p.pickup_date, '%M') month,
 l.borough,
 l.zone
FROM
 ride_pickups p
 INNER JOIN taxi_zone_lookup.taxi_zones.zone_lookup l ON
p.location_id = l.location_id;
```

The results should show 12205653 rows. Run a `SELECT` statement to validate that the table's values are similar to the image below.

```
SELECT * FROM rides_by_zone LIMIT 10;
```

dispatching_base	pickup_date	location_id	weekday	month	borough
B02764	2015-02-07 18:57:53....	181	Saturday	February	Brooklyn
B02764	2015-02-07 20:39:12....	244	Saturday	February	Manhattan
B02764	2015-02-07 01:42:53....	79	Saturday	February	Manhattan
B02764	2015-02-07 16:46:58....	163	Saturday	February	Manhattan
B02764	2015-02-07 23:02:12....	144	Saturday	February	Manhattan

Step 2 - Run interactive analytics

Now, you need to derive two different views for the marketing department. Utilize a WITH statement to run a subquery and only select the weekday that is most popular for each borough.

```
WITH
 weekly AS (
   SELECT
     borough,
     weekday,
     COUNT(*) AS total_rides,
     RANK() OVER (PARTITION BY borough ORDER BY COUNT(weekday) DESC
     ) AS rank_column
   FROM
     rides_by_zone
   GROUP BY
     borough,
     weekday
   ORDER BY
     borough,
     COUNT(*) DESC
 )
SELECT
 borough,
 weekday,
 total_rides
FROM
 weekly
WHERE
 rank_column = 1;
```

borough	weekday	total_rides
Queens	Sunday	192760
Bronx	Saturday	32941
EWR	Thursday	20
Brooklyn	Saturday	408026
Manhattan	Friday	1428080
Staten Island	Saturday	1157
Unknown	Saturday	940

Step 3 - Create the marketing views

Finally, you need to create a view to make the data visible to the marketing team. Use the SQL statement below.

```
CREATE OR REPLACE VIEW borough_most_pop_weekday_vw AS
WITH
 weekly AS (
   SELECT
     borough,
     weekday,
     COUNT(*) AS total_rides,
     RANK() OVER (PARTITION BY borough ORDER BY COUNT(weekday) DESC
     ) AS rank_column
   FROM
     rides_by_zone
   GROUP BY
     borough,
     weekday
   ORDER BY
     borough,
     COUNT(*) DESC
 )
SELECT
 borough,
 weekday,
 total_rides
FROM
 weekly
WHERE
 rank_column = 1;
```

Run a select statement to validate the view was created properly.

```
SELECT * FROM borough_most_pop_weekday_vw;
```

Following the same logic but with a different level of granularity (monthly), create the second view.

```
CREATE OR REPLACE VIEW borough_most_pop_month_vw AS
WITH
 monthly AS (
   SELECT
     borough,
     month,
     COUNT(*) AS total_rides,
     rank() OVER (PARTITION BY borough ORDER BY COUNT(month) DESC
     ) AS rank_column
   FROM
     rides_by_zone
   GROUP BY
     borough,
     month
   ORDER BY
     borough,
     COUNT(*) DESC
 )
SELECT
 borough,
 month,
 total_rides
FROM
 monthly
WHERE
 rank_column = 1;
```

Run a select statement to validate the view was created properly. The results should be similar to the image below.

```
SELECT * FROM borough_most_pop_month_vw;
```

Part 5: Secure access to your Gold layer

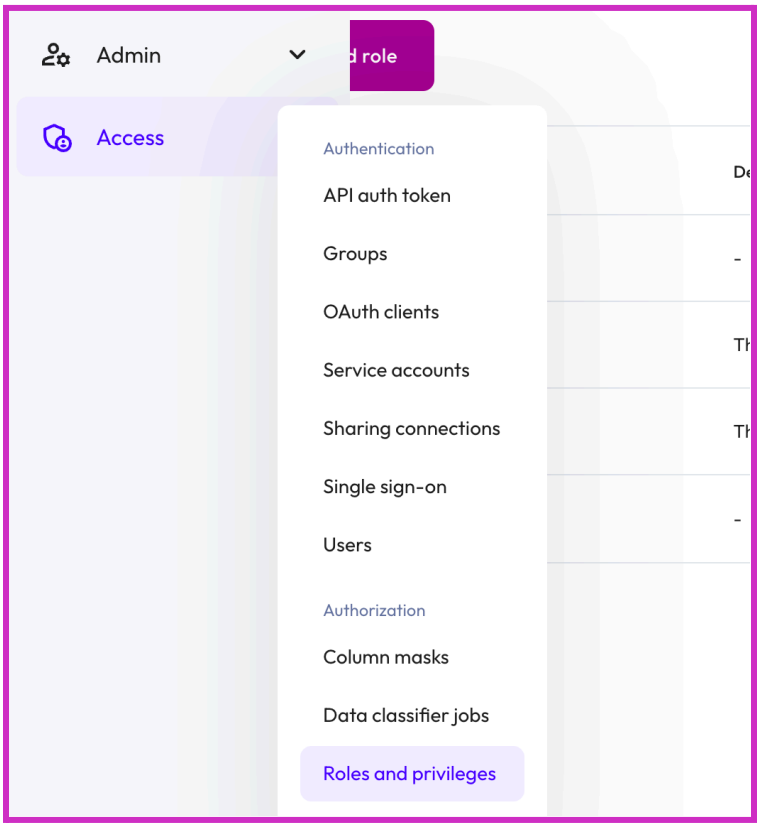
Objective

The Gold layer has been created. Now it's time to ensure that access to this data is restricted to the appropriate users - this will allow you to create data products for specific groups of users.

Step 1 - Create a data science role

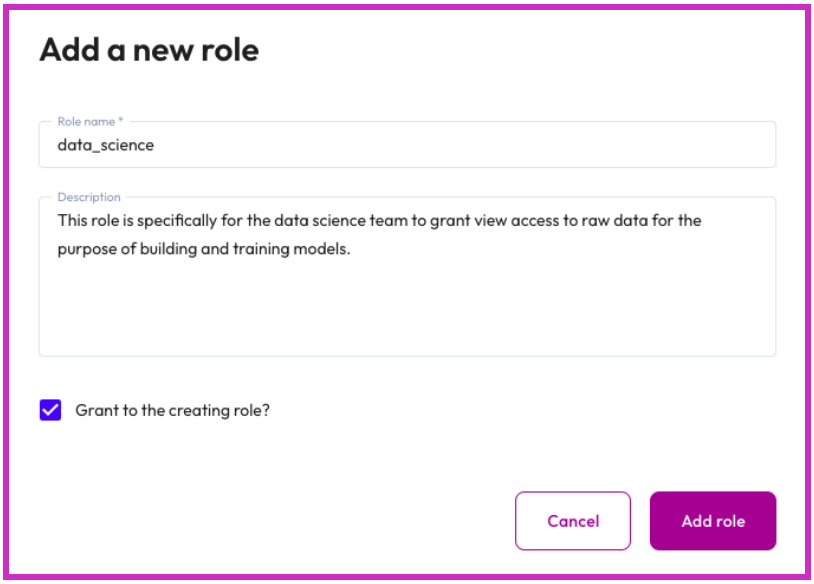
To appropriately grant access to the Bronze layer, you're going to create a specific role for the data science team. This will grant access to the team members with the appropriate rights, and restrict access for everyone else.

To do this, navigate to the **Roles and privileges** tab.



Select **Add role**. Enter the following information:

Role name: data_science
Description: This role is specifically for the data science team to grant view access to raw data for the purpose of building and training models.
Grant to the creating role? Yes

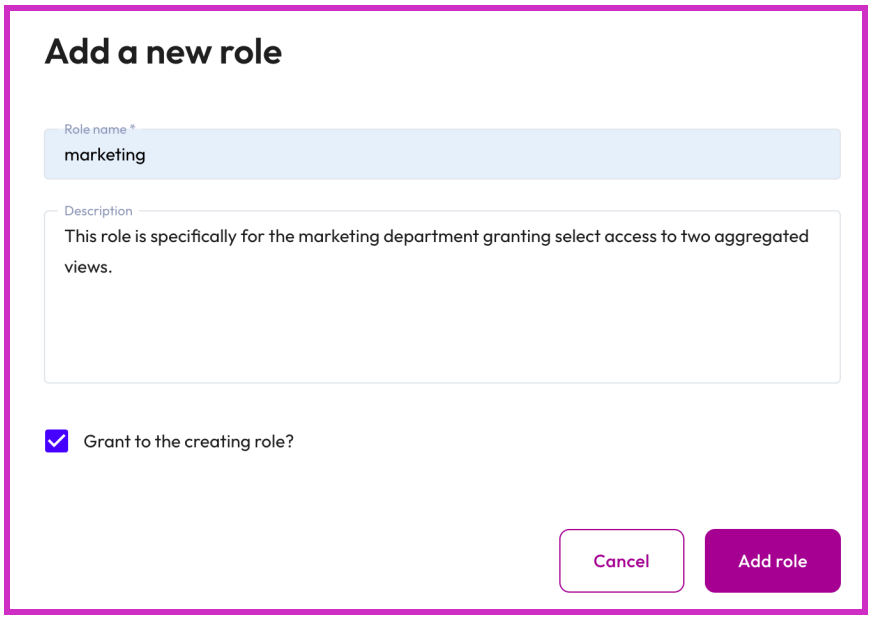


Step 2 - Create a marketing role

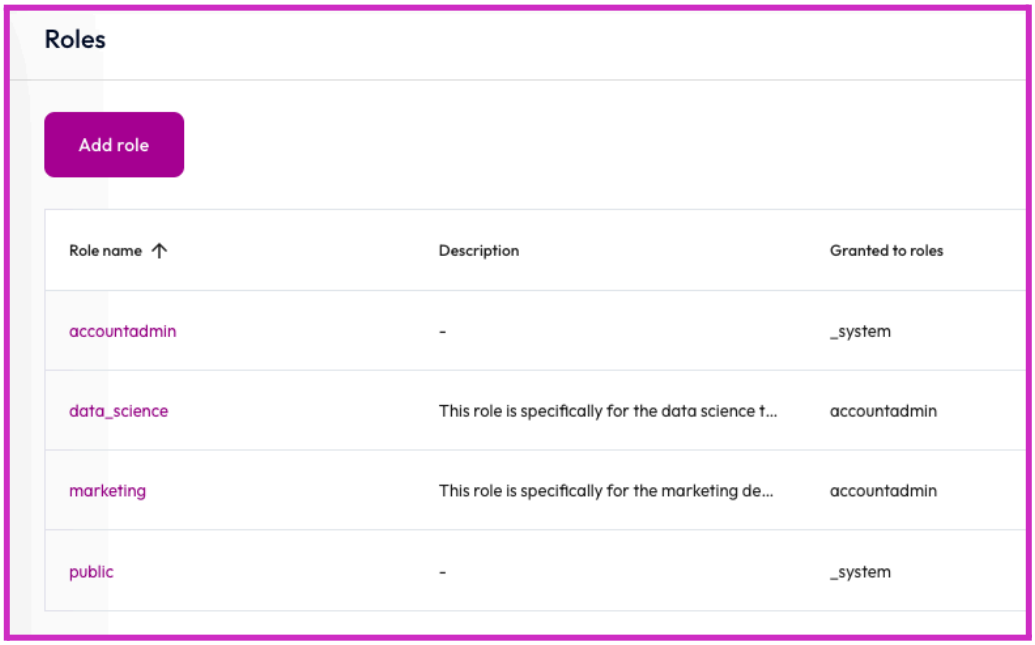
To restrict access to the Gold layer, you're going to create a specific role for the marketing department. This will restrict access to the data to team members with the appropriate rights, and restrict their access to the two newly created views.

To do this, navigate to the **Roles and privileges** tab. Select **Add role**. Enter the following information:

Role name: marketing
Description: This role is specifically for the marketing department granting view access to two aggregated views.
Grant to the creating role? Yes



You should now have two new roles created for a total of four roles plus any roles previously created.



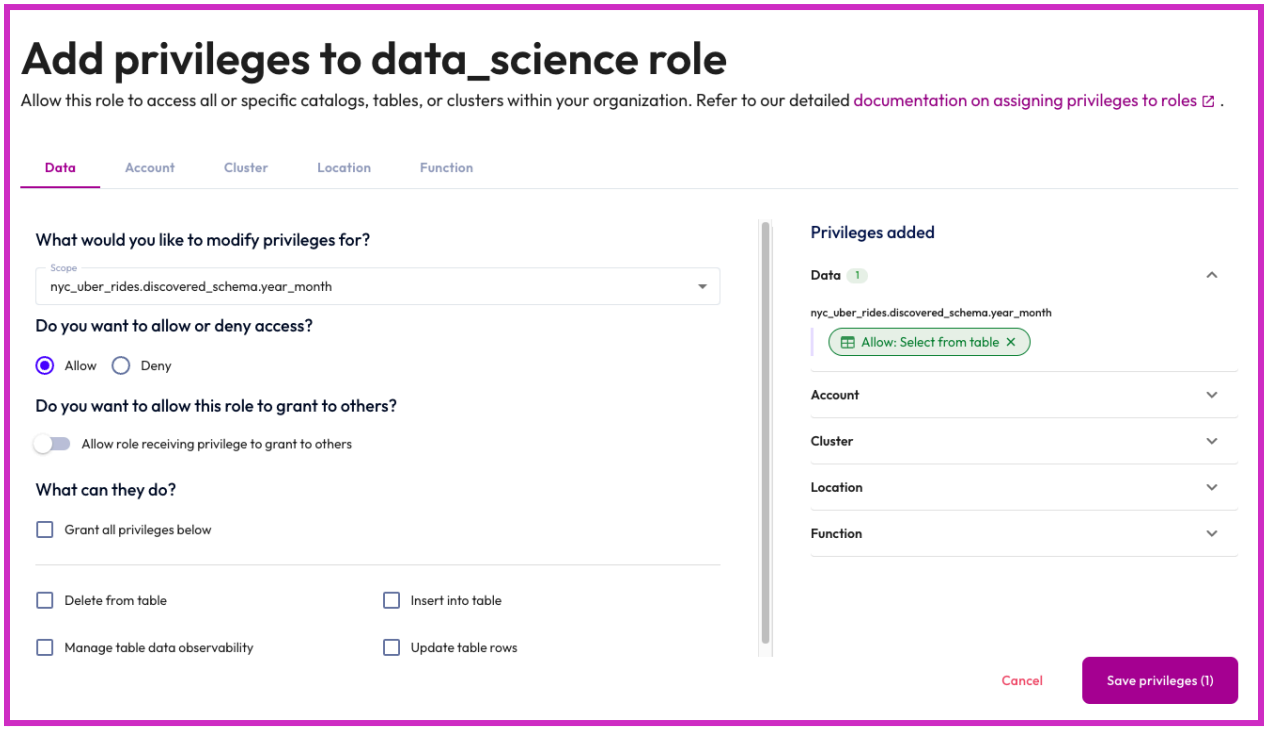
Step 3 - Assign data science privileges

Next, select the newly created data_science role. This allows you to assign proper privileges. To do this, navigate to the **Privileges** tab. Select **Add privilege**.

What would you like to modify privileges for?

`nyc_uber_rides.discovered_schema.year_month`

Do you want to allow or deny access? *Allow*
Do you want to allow this role to grant to others? *No*
What can they do? *Select from table*



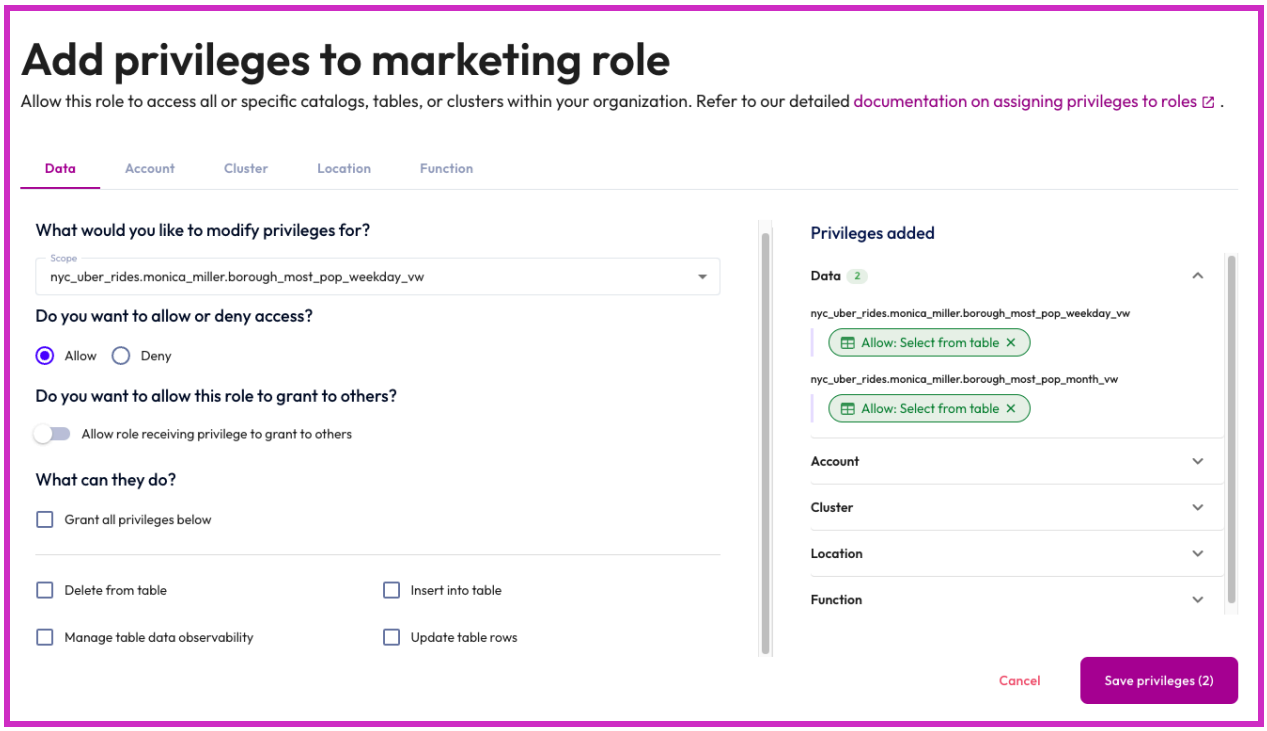
Select **Save privileges(1)**.

Step 4 - Assign marketing privileges

Next, select the newly created marketing role. This allows you to assign proper privileges. To do this, navigate to the **Privileges** tab. Select **Add privilege**.

What would you like to modify privileges for?
`nyc_uber_rides.yourname.borough_most_pop_month_vw`
Do you want to allow or deny access? *Allow*
Do you want to allow this role to grant to others? *No*
What can they do? *Select from table*

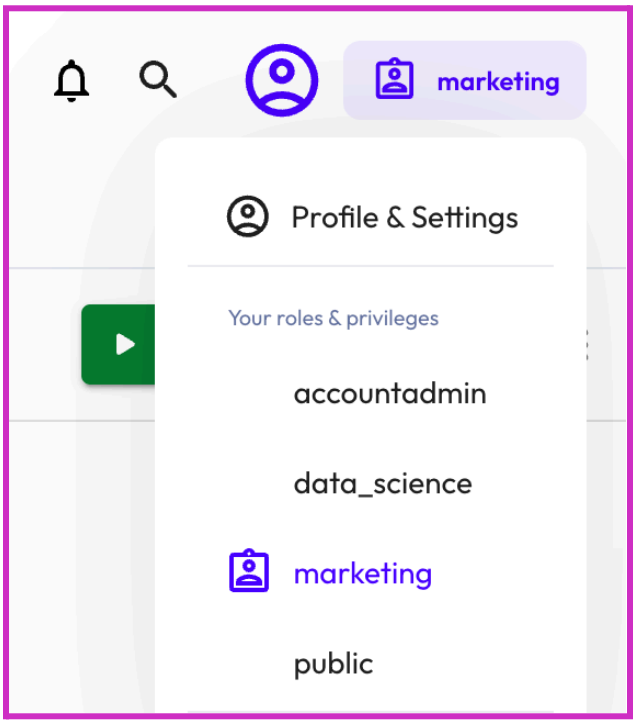
Repeat the process for the table `borough_most_pop_weekday_vw`.
What would you like to modify privileges for?
`nyc_uber_rides.yourname.borough_most_pop_weekday_vw`
Do you want to allow or deny access? *Allow*
Do you want to allow this role to grant to others? *No*
What can they do? *Select from table*



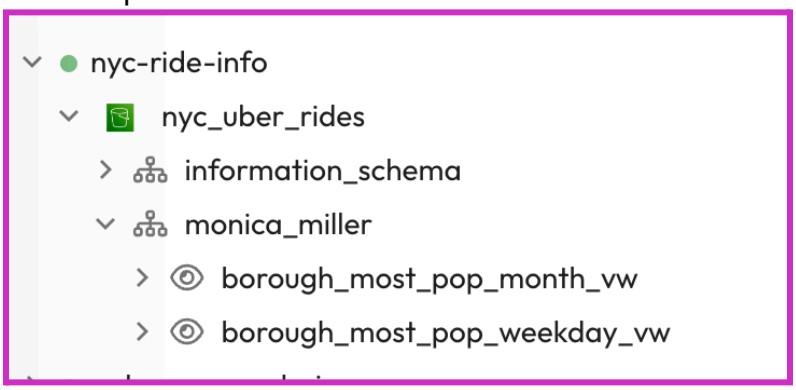
Then hit **Save Privileges(2).**

Step 5 - Test your new roles

Now it's time to test that the new roles are working correctly. Navigate back to the **Query editor** and switch to the **marketing** role in the top right-hand corner.



Notice that the marketing role view has fewer capabilities available in the left-hand navigation bar. Also, the cluster explorer shows only two views in the `nyc-ride-info` cluster. This is exactly what you would expect.



Run a select statement to validate the newly created role has access to view the tables.

```
SELECT * FROM borough_most_pop_month_vw;
```

Now, try recreating the view using the marketing role. Run the command shared in Lab 3 Part 4 Step 4. This will fail because you have only granted the marketing role select permissions.

Also try accessing the year_month table.

```
SELECT * FROM discovered_schema.year_month;
```

Navigate to the **data_science** role in the upper right hand corner. Test select access for the year_month table, and test access for the newly created views.

```
SELECT * FROM discovered_schema.year_month;

DELETE FROM discovered_schema.year_month where year_month =

'2015-09';

SELECT * FROM borough_most_pop_month_vw;
```

Navigate back to the **account_admin** role.

END OF LAB EXERCISE

Lab 4: Create data products

Learning objectives

- Demonstrate how to execute a global search.
- Demonstrate how to create a data product.
- Demonstrate how to create tags.

Prerequisites

- Lab 1: Introduction and setup
- Lab 2: Connect to data sources
- Lab 3: Build within your data lake

Activities

1. Execute global search
2. Create a data science data product
3. Create a marketing data product

Part 1: Execute global search

Objective

Global search lets users find datasets quickly and intuitively. It is a powerful tool that helps keep better track of your data.

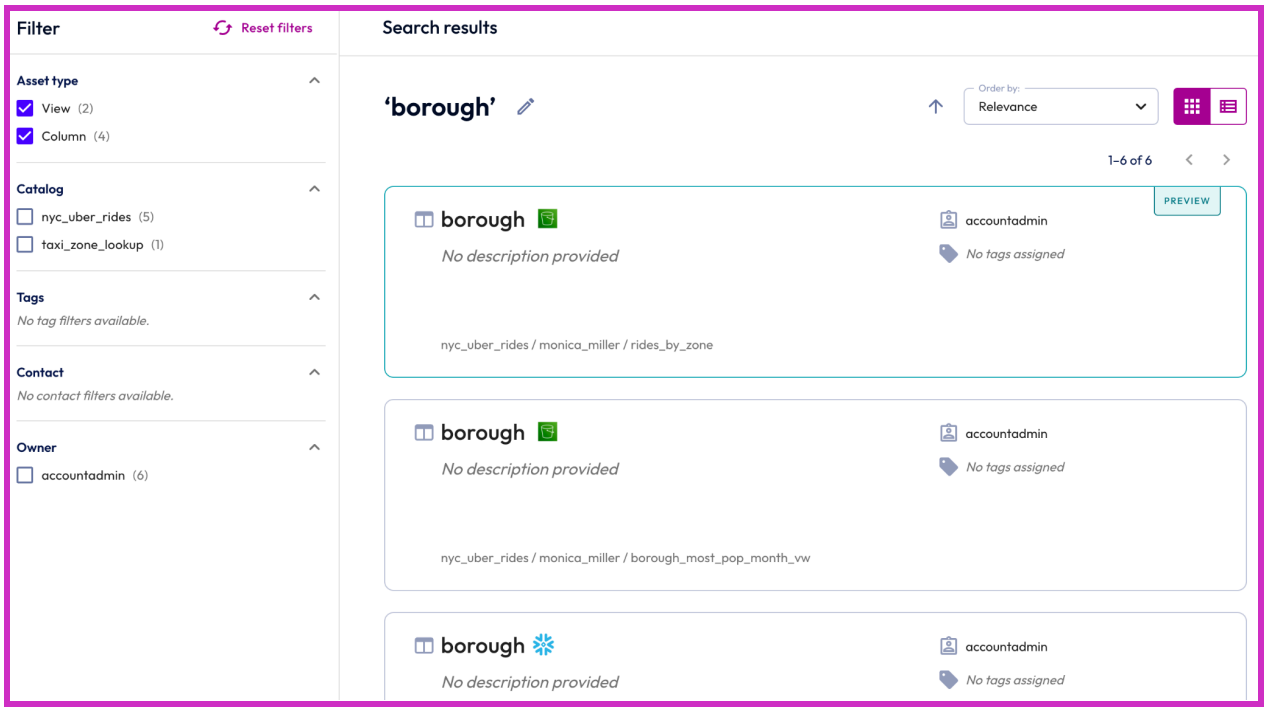
Step 1 - Navigate Starburst UI

To use global search, select the magnifying glass icon in the upper-right corner.

Step 2 - Execute global search

Enter the word `borough` and select **View all results**.

Starburst Galaxy displays multiple instances matching your search criteria drawn from multiple data sources. Some occur in tables and views you created throughout the lab. Others are simply populated through your catalog connection.



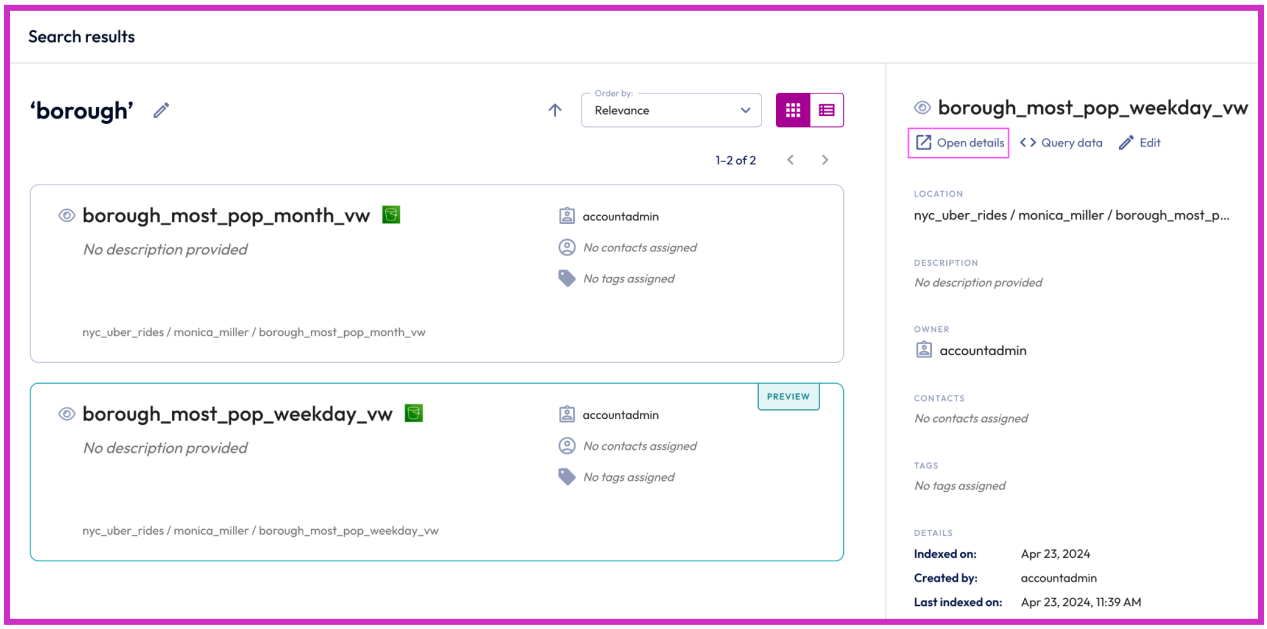
Step 3 - Filter global search

Global search can also be filtered to help refine your search.

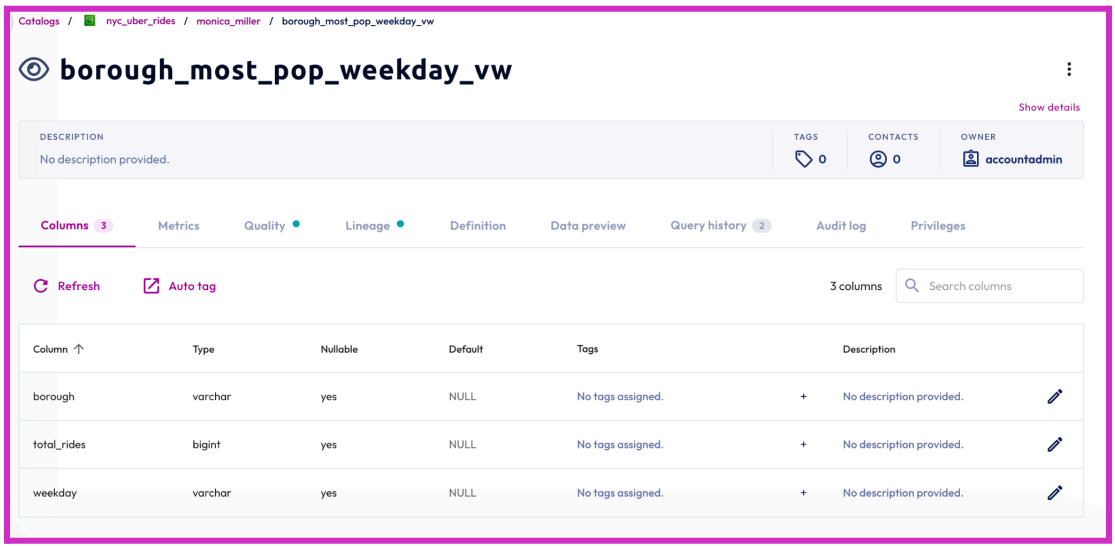
Select the View filter on the left to see how different search criteria impact the results.

Step 3 - Navigate to your newly created view

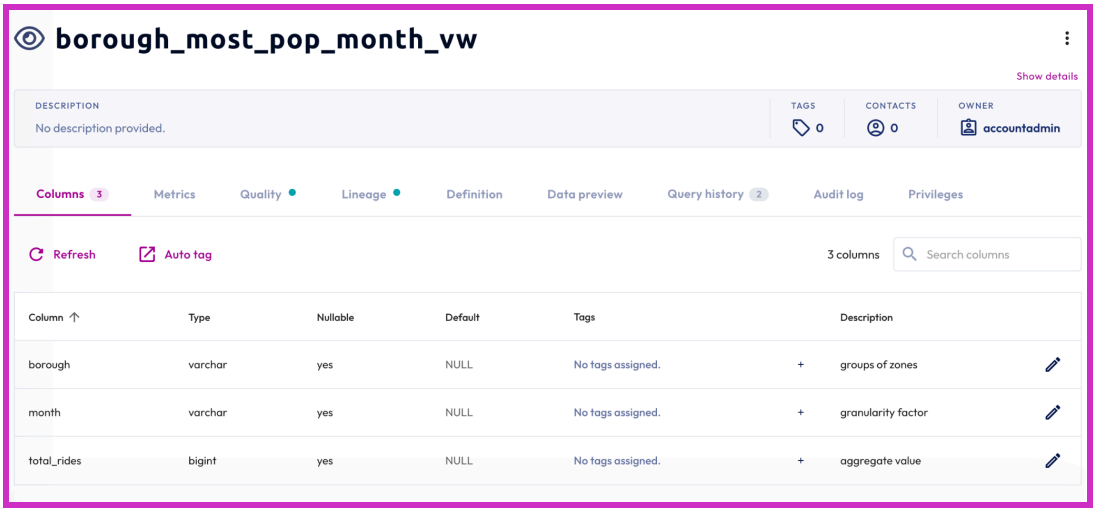
Click on the view `borough_most_pop_weekday_vw`. Select **Open Details**.



Here you can see **Column** information, **Metrics**, **Quality, Lineage, Definition**, **Data preview**, **Query history**, **Audit log**, and **Privileges**. Navigate through each tab.



This page contains information specific to the `borough_most_pop_weekday_vw` view. If you have extra time, go through the tables and views created and add meaningful descriptions to each table/view and the columns within them.



Navigate to your < yourname> schema in the top left-hand corner.

The information for the entire schema is available to you, including **Tables**, **Views**, **Metrics**, **Definition**, **Query history**, and **Privileges**. Navigate through each tab to see the available features.

Part 2: Create your data products

Objective

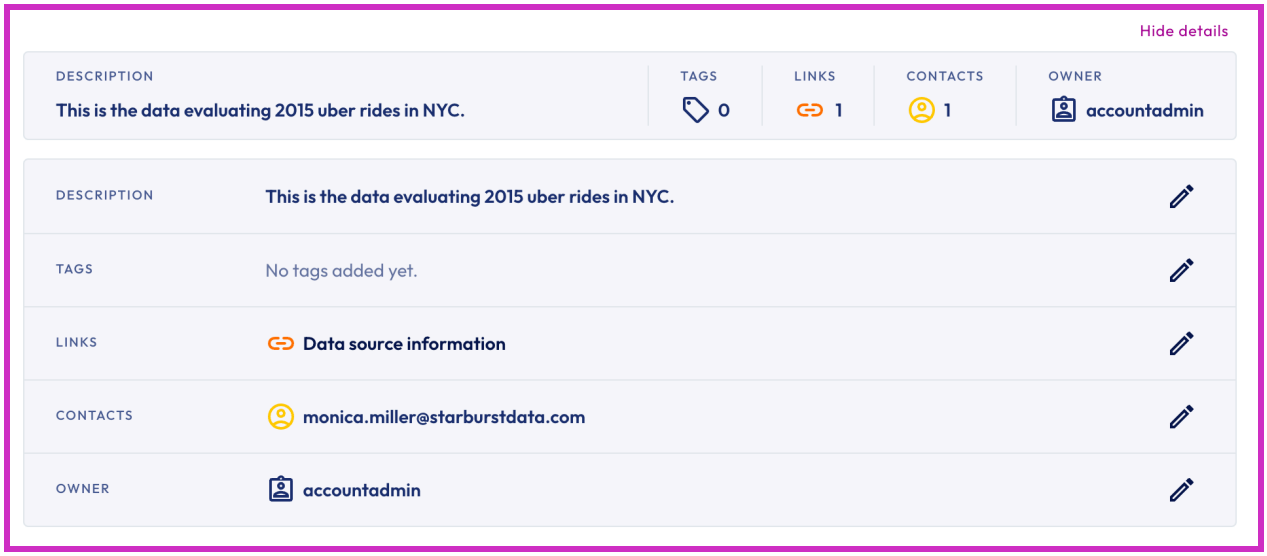
Now it's time to create a data product using your datasets. Data products curate data in a way that makes it more accessible and useful, and can be shared across teams.

Step 1 - Enter additional information

Make sure you are in your schema as mentioned in the previous step. Before creating a data product, enter some information into your catalog.

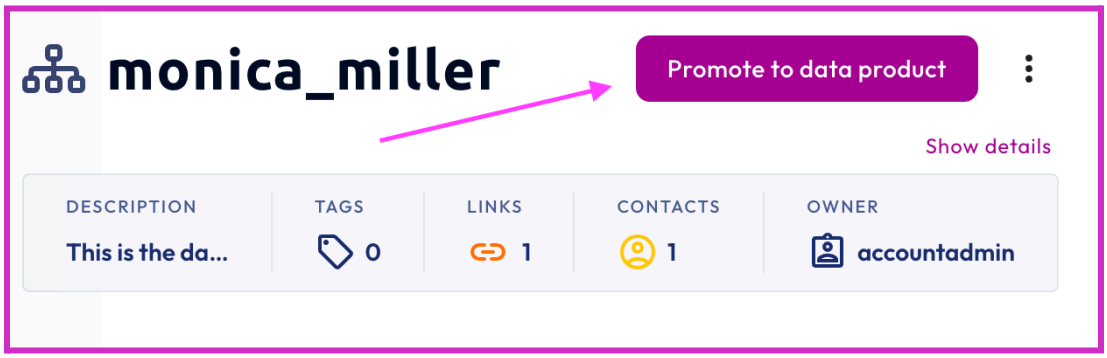
Select **Show details** on the right hand side. Edit the following information:

Description: This is the data evaluating 2015 uber rides in NYC.
Links: Text to display: Data source information
Link URL:
https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city?select=other-FHV-services_jan-aug-2015.csv
Contacts: yourname

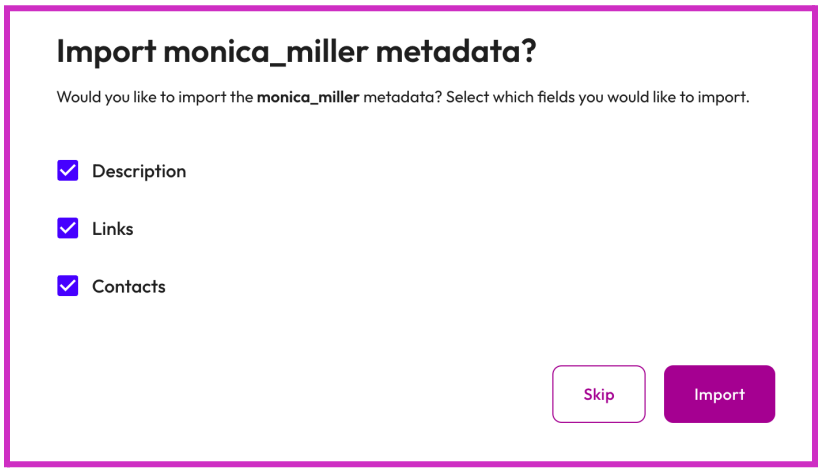


Step 2 - Promote your data product

Navigate back to your schema and select **Promote to data product**.



Import all the information you've already added to the schema. Select **Import**.



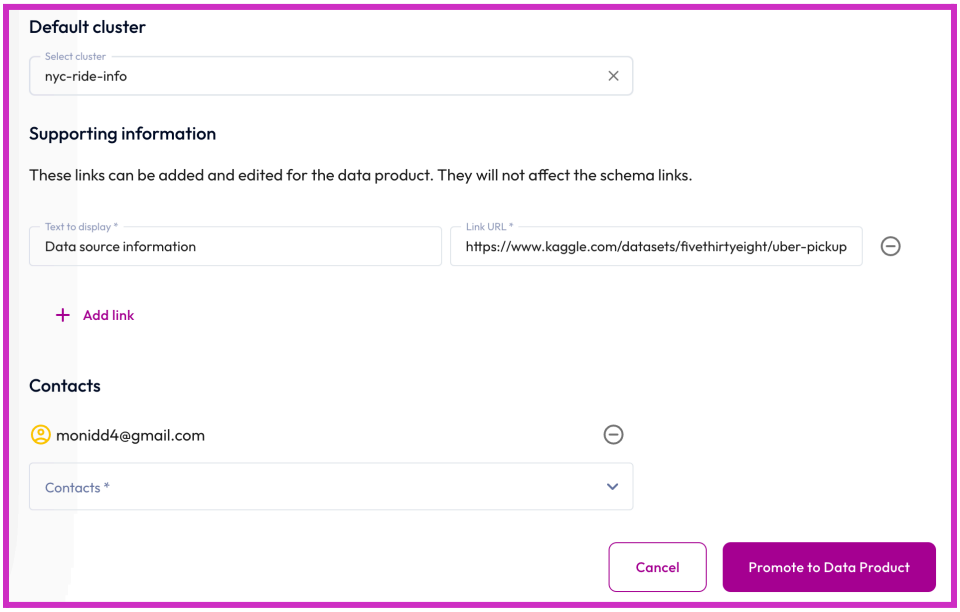
Add a descriptive name like **NYC-2015-uber-rides-aggregate**. The summary has already been populated based on the information you added to the schema.

Add the following description:

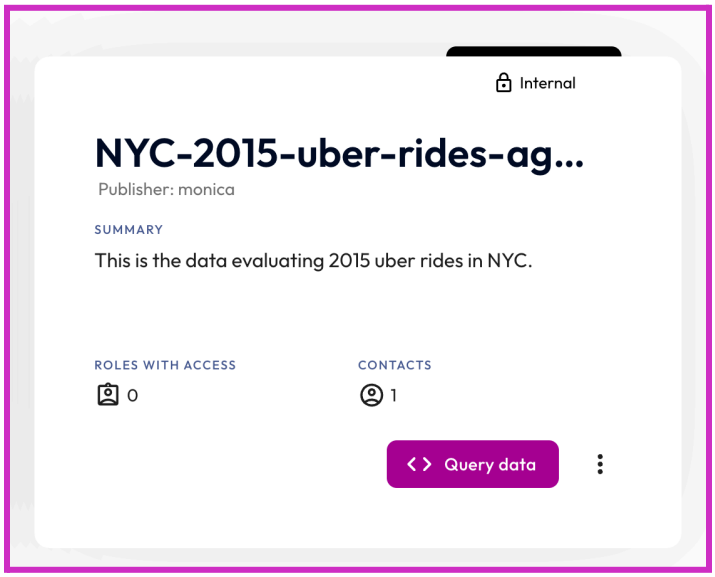
Use this data product to plan marketing campaigns around geo-location based on borough.

- ***Question:*** *What is the most popular day of the week for each borough?*
- ***Objective:*** *Plan marketing campaigns for each borough based on popularity.*
- ***Approach:*** *You create a specific view for each borough and day.*

Select the **nyc-ride-info** cluster as the default cluster. Your contacts and supporting information have been automatically populated.
Select **Promote to data product**.

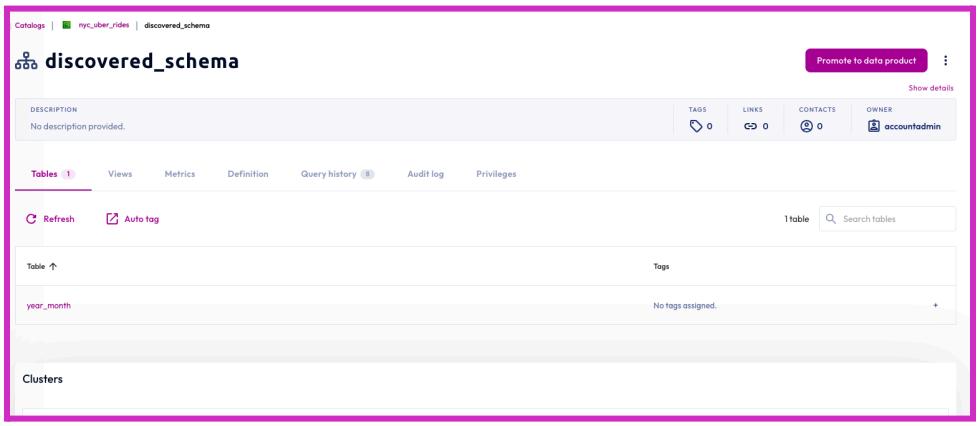


Congratulations! You have created and promoted your first data product. Navigate to the Data products tab to view your work.



Step 3 - Create a data product for the data scientists

Use global search to navigate to the Bronze layer- the schema known as **discovered_schema**. Click on the returned schema.



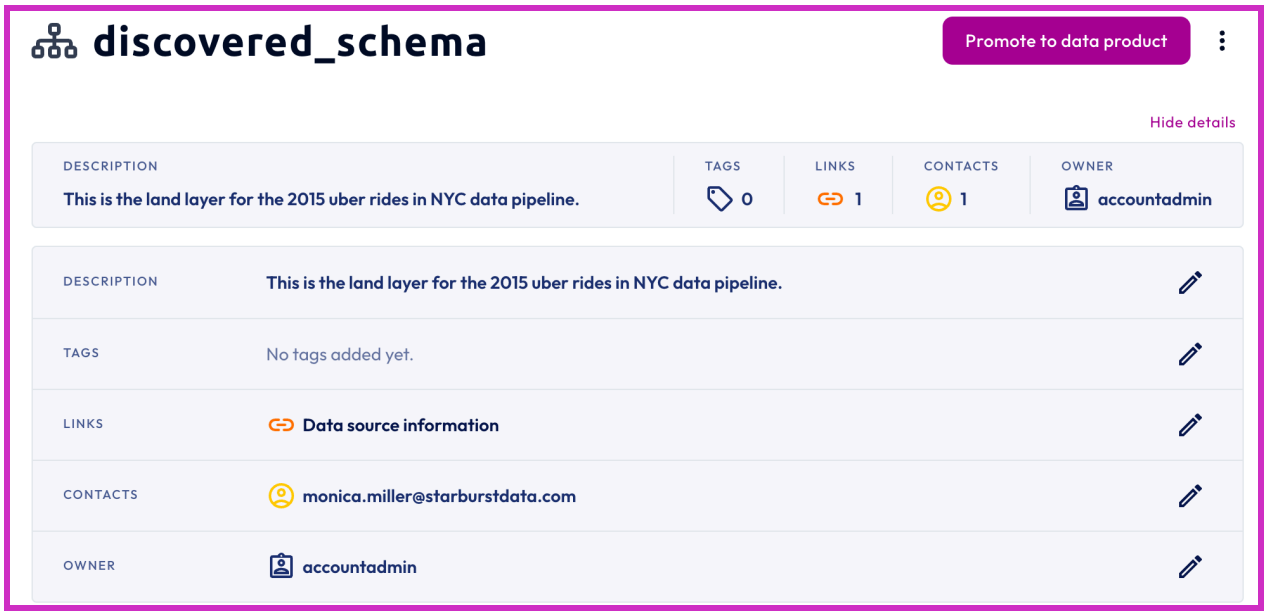
Step 4 - Enter additional information

Make sure you are in your schema as mentioned in the previous step. Before creating a data product, enter some information into your catalog.

Select **Show details** on the right hand side.

Edit the following information:

Description: This is the Bronze layer for the 2015 uber rides in NYC data pipeline.
Links: Text to display: Data source information
Link URL:
https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city?select=other-FHV-services_jan-aug-2015.csv
Contacts: yourname



Next, click the **Promote to data product** button. Import the metadata.

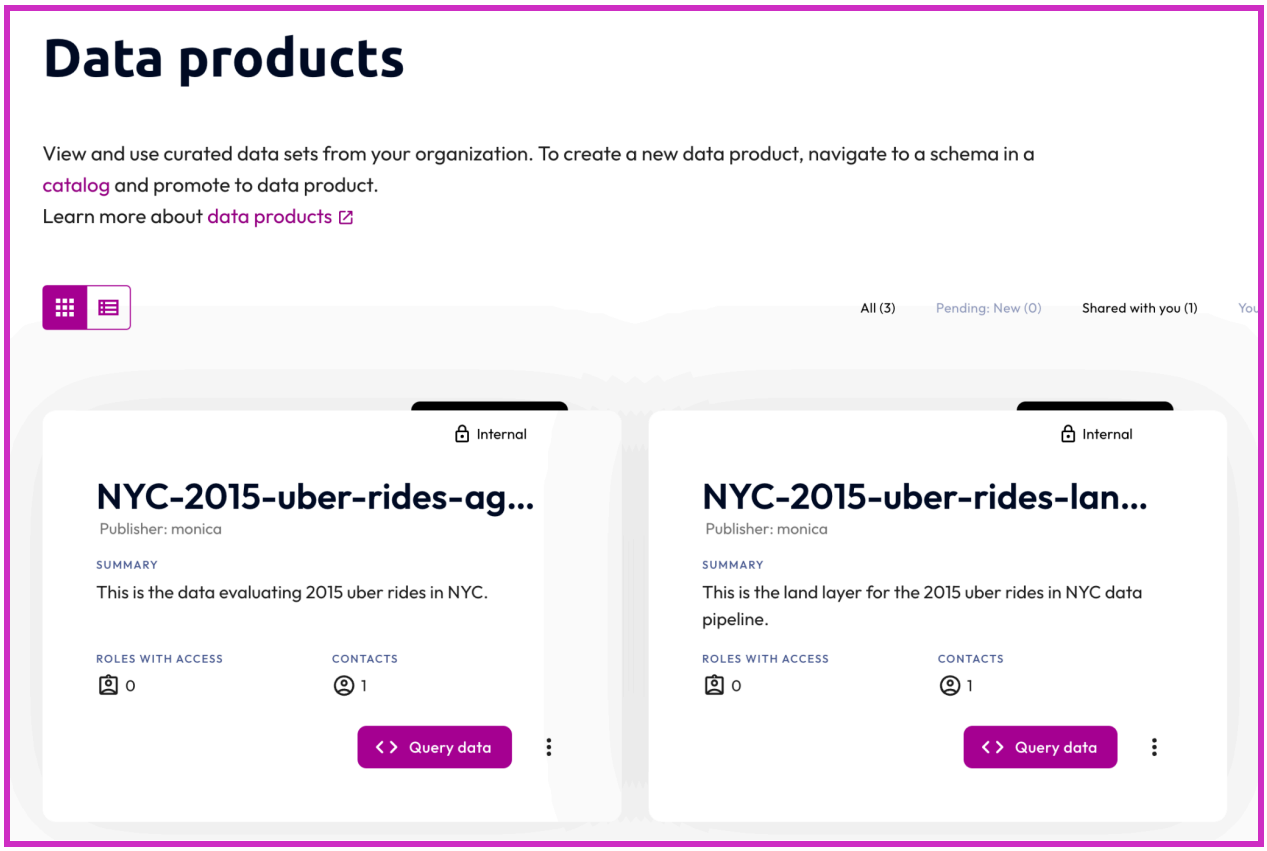
Add a descriptive name like **NYC-2015-uber-rides-bronze-layer**. The summary has already been populated based on the information you added to the schema.

Add the following description:

Use this data product to build and train models for the data science team around geo-location based on borough. Utilized by the data_science role. Built for specific investigation.

Select the **nyc-ride-info** cluster as the default cluster. Your contacts and supporting information have been automatically populated.
Select **Promote to data product**.

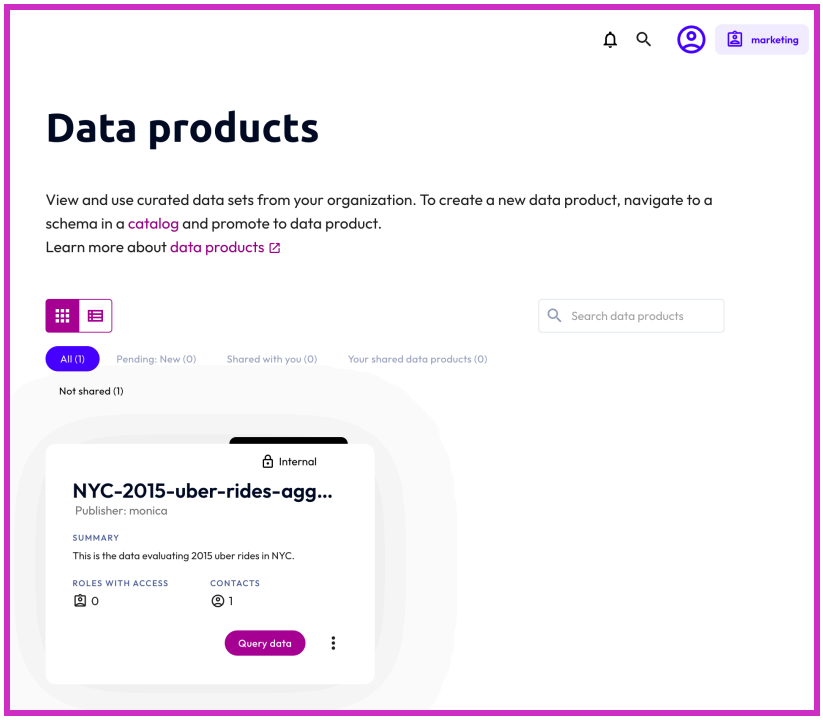
Hooray! Now you should be able to see two different data products in the **accountadmin** role.



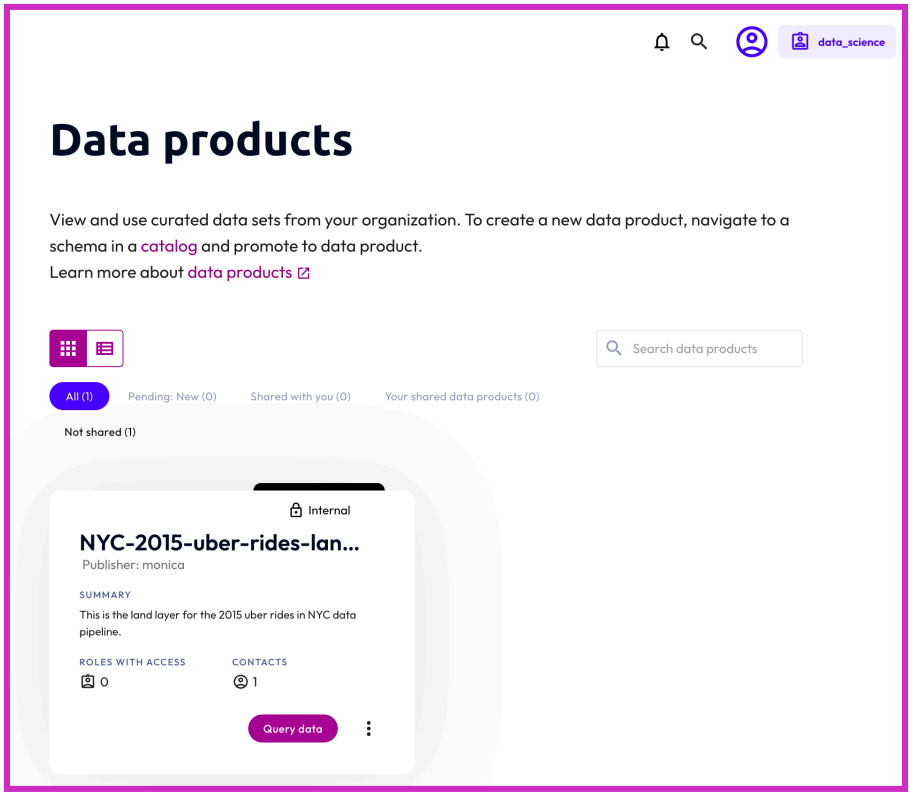
Step 5 - Utilize access control alongside your data products

Starburst Galaxy provides built-in access control using customizable levels of granularity. You can control this feature using both role-based access control (RBAC) and attribute-based access control (ABAC). Starburst Galaxy's built-in access control allows you to define multiple users, roles, groups, privileges, and policies. These access options encourage businesses to create security policies that make sense for their organization. Explore the different data products permissions for the three roles: **accountadmin, data_science, marketing**.

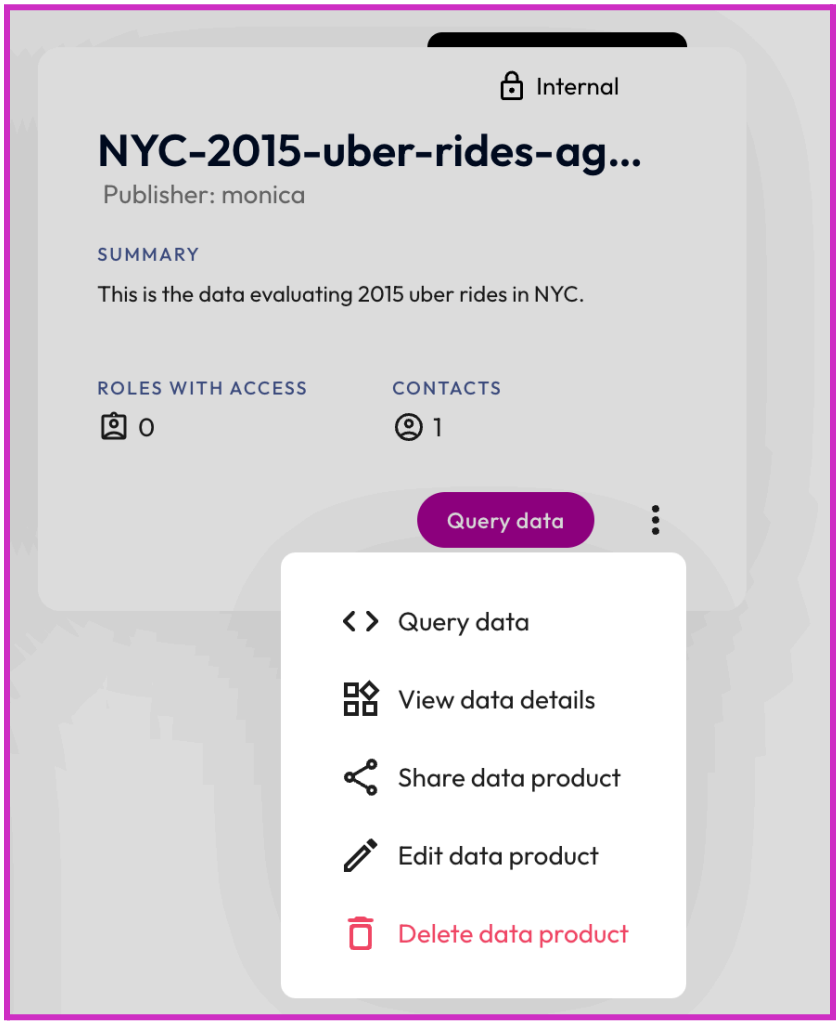
Switch to the marketing role and view the data products screen. Notice how only one data product is viewable.



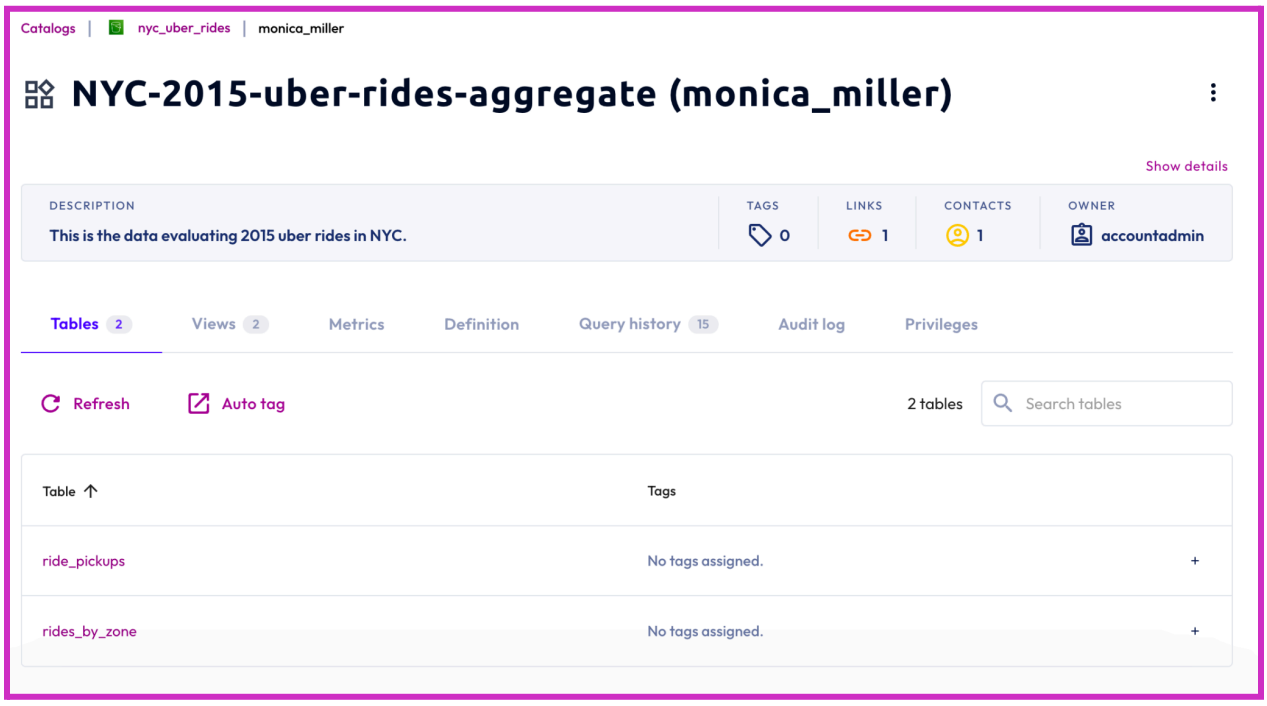
Now, switch to the data_science role. The data_science role also has only one viewable data product, notice it is a different data product than the marketing team.



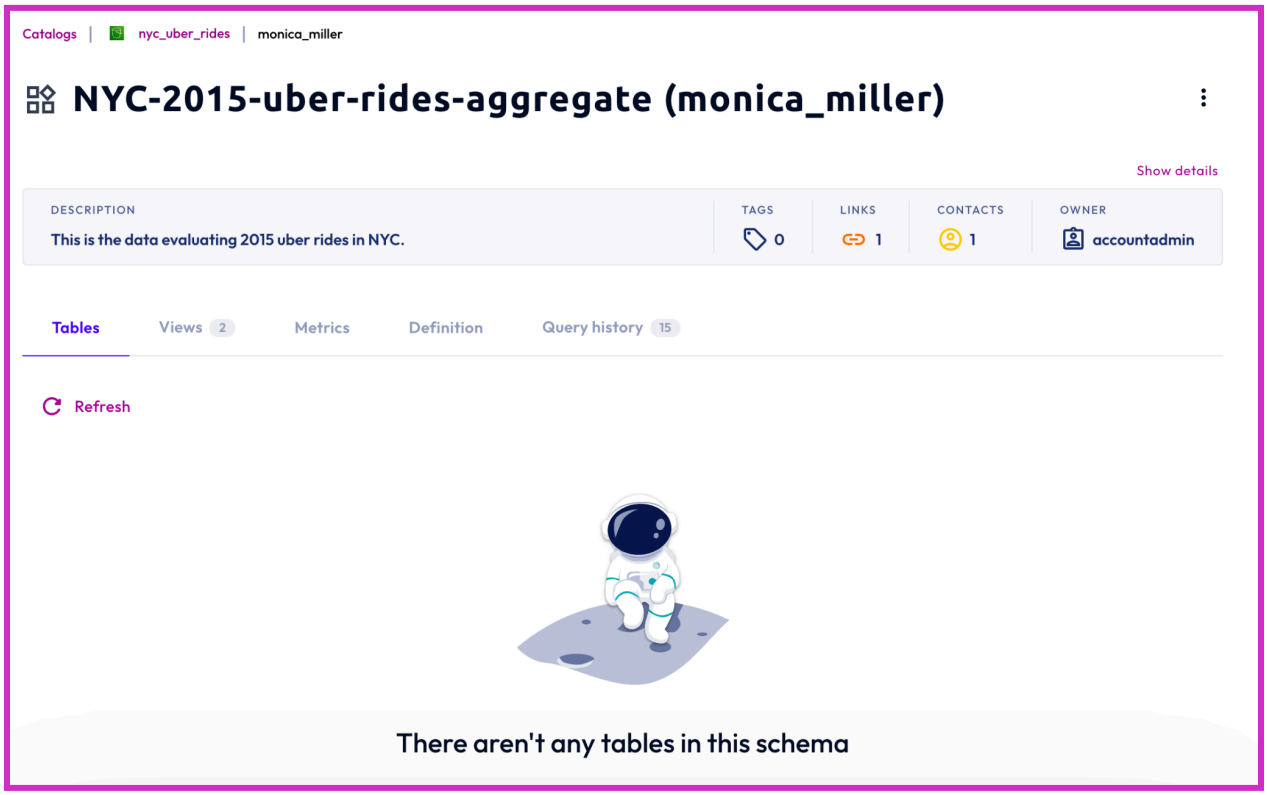
Now, dive into the data products as an **accountadmin** and as each role to view the differences. Go back to the **accountadmin** role and view the **NYC-2015–uber-rides-aggregate** data product by clicking the three dots and selecting **View data details.**



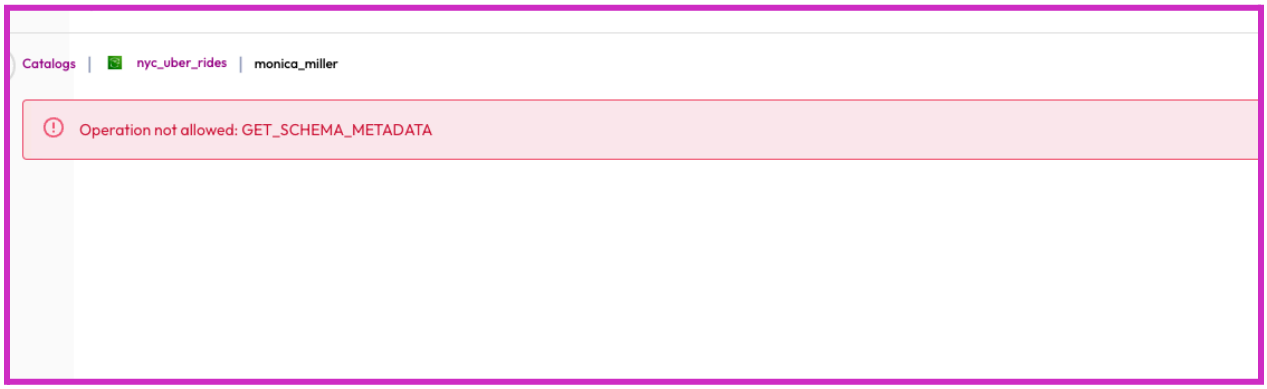
The **accountadmin** role shows two tables and two views that create the data product.



Switch to the marketing role next. Now, you will only see two views.



Finally, try the data_science role. You will see you cannot view the metadata information.



This is an example of how RBAC and data products can work together to provide a granulated data management solution. For another example of an RBAC application, watch this video.

Appendix: Section A

If you are unable to create your Bronze layer with schema discovery, utilize the code below. This will create the `discovered_schema` and the `year_month` table manually.

Step 1 - Select the rideshare cluster

Navigate to the **Query editor**. Add a new tab using the fuschia plus sign +. Validate the location drop-downs in the top left hand corner match the cluster and catalog previously created.

Cluster: nyc-ride-info
Catalog: nyc_uber_rides

Step 2 - Create lab schema

In the Query editor pane, open a new tab and create a new schema with the following command.

```
CREATE SCHEMA nyc_uber_rides.discovered_schema;
```

Step 3 - Verify schema in catalog

Once this query is **Finished**, in the catalog explorer, expand the `nyc_uber_rides` catalog to verify your new schema is present and is empty. There will be other schemas listed.

Step 4 - Select schema after creation

In the top left hand corner, select your schema. Validate the location drop-downs in the top left hand corner match the cluster and catalog previously created.

Cluster: nyc-ride-info
Catalog: nyc_uber_rides
Schema: `discovered_schema`

Step 2 - Create the external table

Run the following command to create an external table from the rideshare data stored in the data lake.
This table is partitioned by year_month with a retention period of six months.

```
CREATE TABLE year_month (
   dispatching_base varchar,
   pickup_date varchar,
   affiliated_base_num varchar,
   location_id varchar,
   year_month varchar
 )
WITH
 (
   format = 'csv',
   type = 'hive',
   external_location =
's3://starburst101-handsonlab-nyc-uber-rides/year_month',
   skip_header_line_count = 1,
   partitioned_by = array['year_month']
 );
```

Recall that the default table format you set for this catalog was Iceberg. However, you are still able to create a table with the Hive table format due to Great Lakes connectivity in Starburst Galaxy. Great Lakes connectivity abstracts the details of using different table formats and file types when using certain write access statements for object storage systems.

Since you are using Starburst Galaxy's metastore, run the following command to sync the metadata.

```
call system.sync_partition_metadata('discovered_schema',
'year_month', 'ADD');
```