dbt Cloud & Starburst Galaxy hands-on workshop

v1.0.0

Table of Contents

Workshop introduction 1
Lab 1: Create Starburst Galaxy account and data catalogs 5
Lab 2: Discover & review the land zone datasets 18
Lab 3: Create dbt Cloud account and connect to Starburst Galaxy 30
Lab 4: Build the structure zone with dbt Cloud models 35
Lab 5: Materialize the consume zone with a joined & aggregated dbt Cloud model 54
Lab 6: Define a Starburst Galaxy data product 61
Lab 7: Commit changes and create a production environment 76

Workshop introduction

Note: Recordings of the introduction and the labs can be found in this YouTube Playlist.

Workshop objectives

- Introduce dbt Cloud and Starburst Galaxy
- Review the data lakehouse reference architecture
- Present a data pipeline scenario
- Help YOU build it out across several labs
    - Create Starburst Galaxy account and data catalogs
    - Discover & review the land zone datasets
    - Create dbt Cloud account and connect to Starburst Galaxy
    - Build the structure zone with dbt Cloud models
    - Materialize the consume zone with a joined & aggregated dbt Cloud model
    - Define a Starburst Galaxy data product
    - Commit changes and create a production environment

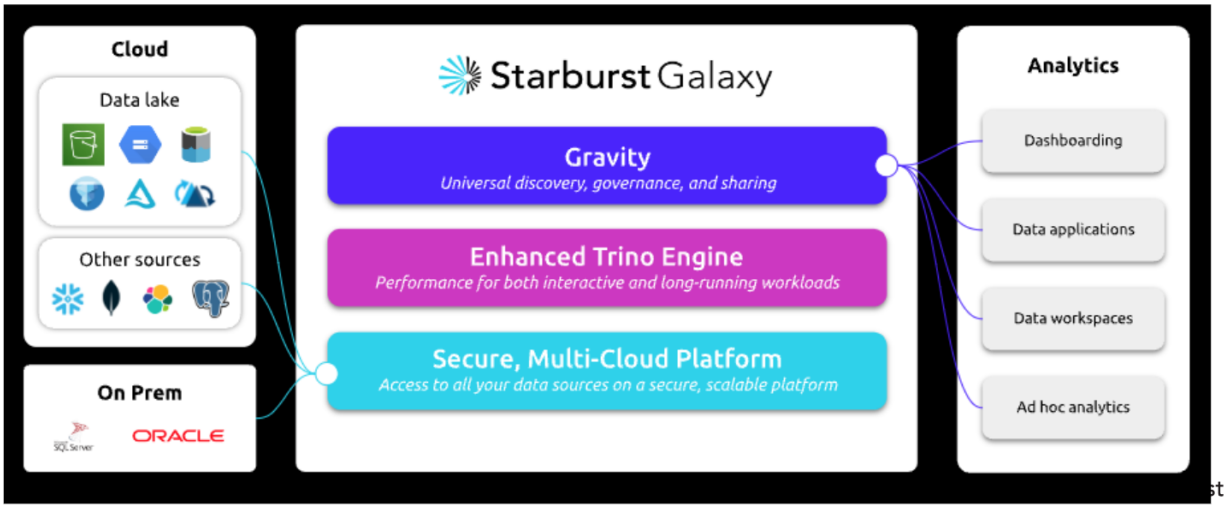
Starburst

dbt™

https://www.getdbt.com/

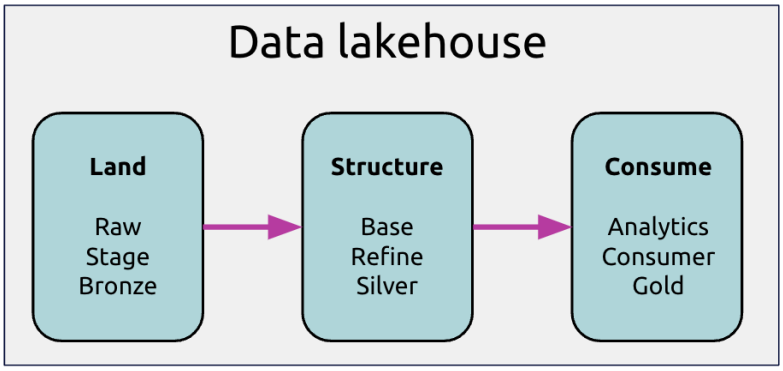
Starburst Galaxy

https://www.starburst.io/platform/starburst-galaxy/



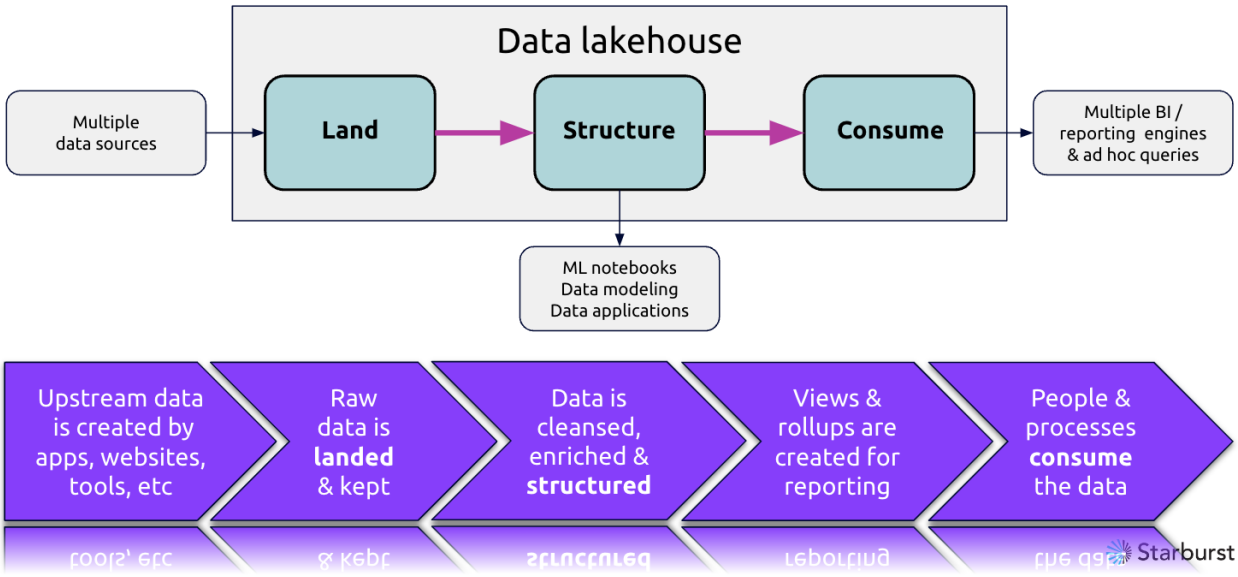
Reference architecture

The reference architecture centers around the data lakehouse and how we classify our data assets into distinct zones. Data pipelines populate the zones.

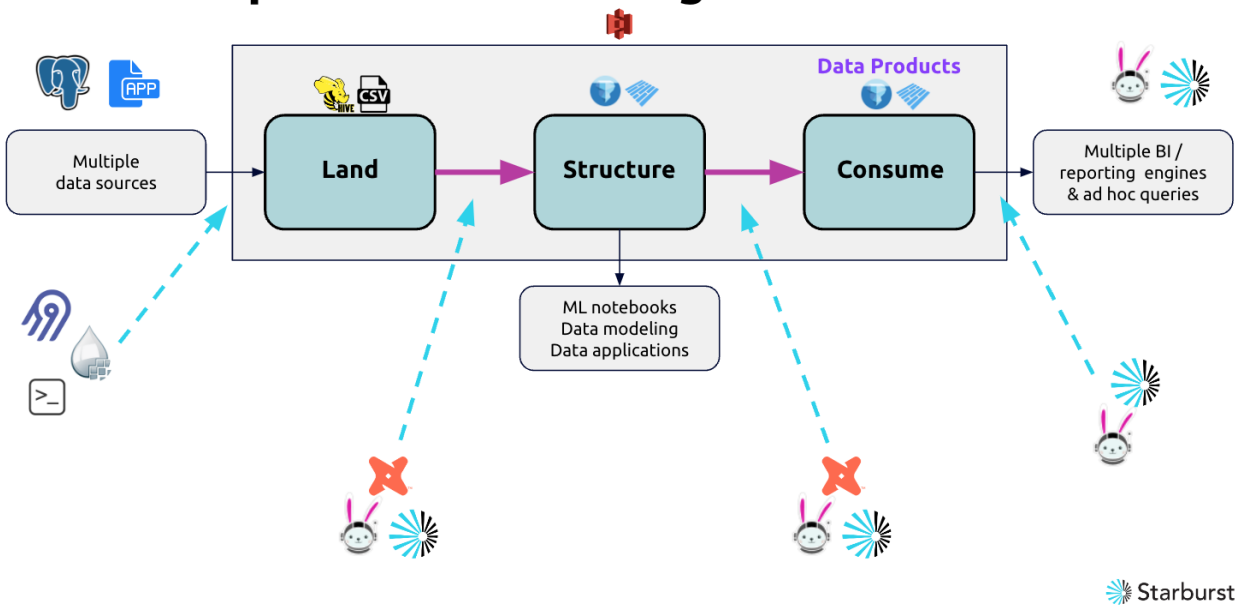


Starburst

Activities across the architecture



Workshop tools & technologies



Pipeline scenario (dbt's Jaffle Shop)

- Leverage land zone datasets
  - Customers (PostgreSQL)
  - Orders (PostgreSQL)
  - Payments (Amazon S3)
- Validate, standardize & build structure zone datasets (Apache Iceberg)
- Define a consume zone model using joined/aggregated structure zone datasets *(View)*
- Verify correct land > structure > consume zone data lineage *(dbt Cloud & Starburst Galaxy)*
- Expose curated datasets as data products *(Starburst Galaxy)*

Starburst

Lab 1: Create Starburst Galaxy account and data catalogs

Estimated completion time

- 15 minutes

Learning objectives

- This lab will walk you through the process of creating a new account within Starburst Galaxy. You will create a domain name and password and set up your account to begin using sample data. Two catalogs will be created as well as initial privileges to these data sources. Finally, you will create a cluster and configure it to access the new catalogs.

Prerequisites

- None.

Activities

1. Sign up for Starburst Galaxy
2. Configure a PostgreSQL catalog
3. Configure a data lakehouse catalog
4. Create a cluster

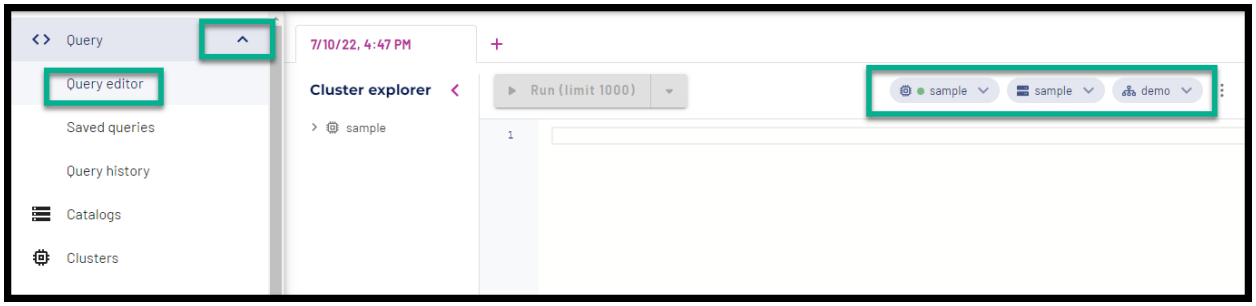
Step 1 - Sign up for Starburst Galaxy

To sign up for Starburst Galaxy, follow the instructions on the free registration page at https://www.starburst.io/platform/starburst-galaxy/start/.

When prompted, choose to not connect your data sources, but choose to use the sample data.

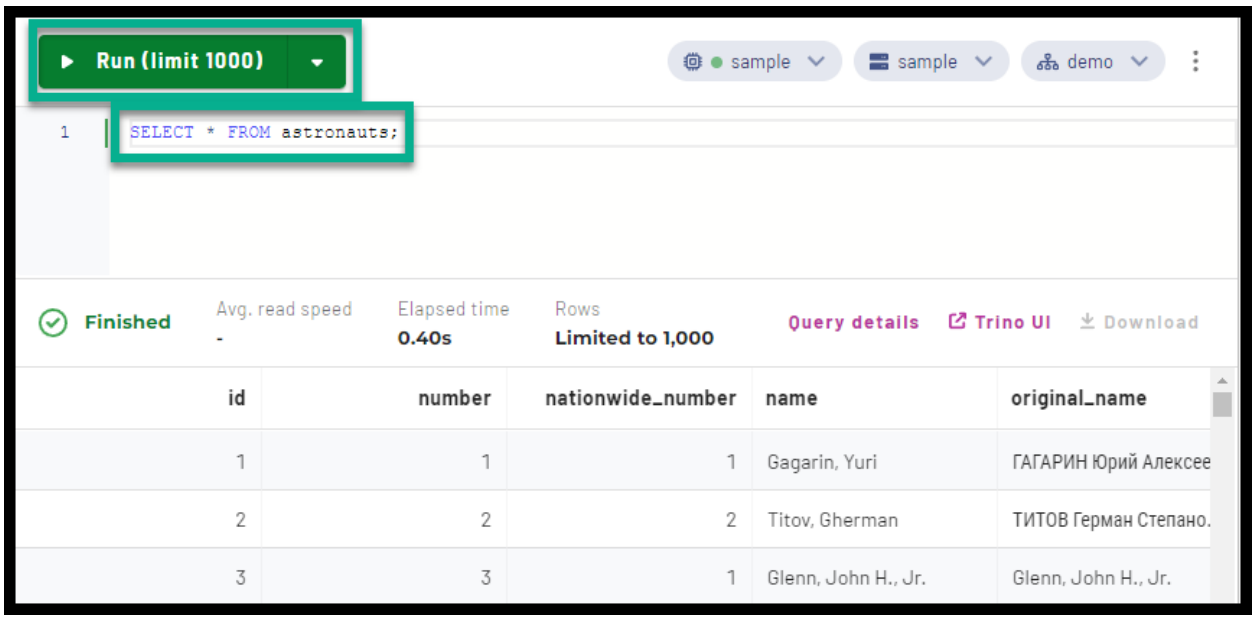
Follow these steps to open the **Query editor**.

- Expand **Query**
- Click **Query editor**
- Ensure the **Select cluster** drop down is set to **free-cluster**
- In the **Select catalog** drop down, select **sample**
- After the previous selection you will see the **Select schema** drop down
- Select the **demo** schema



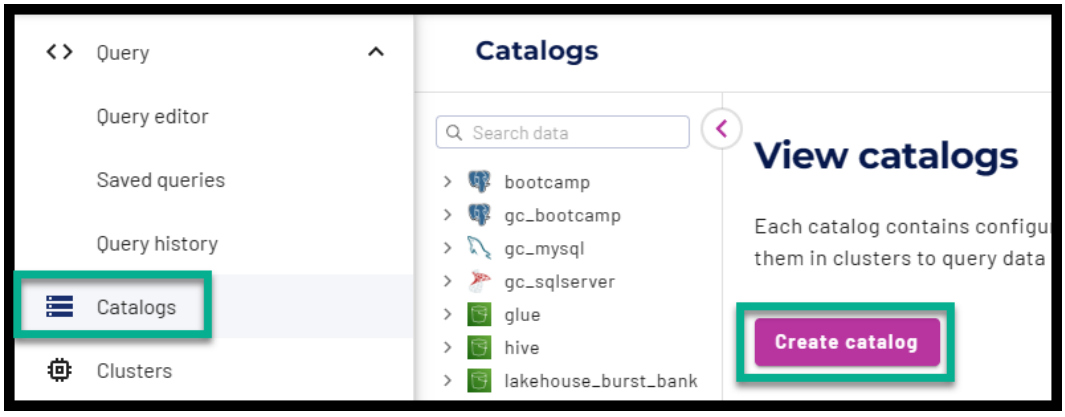
Paste the following SQL into the editor and then click **Run (limit 1000)**.

```
SELECT * FROM astronauts;
```

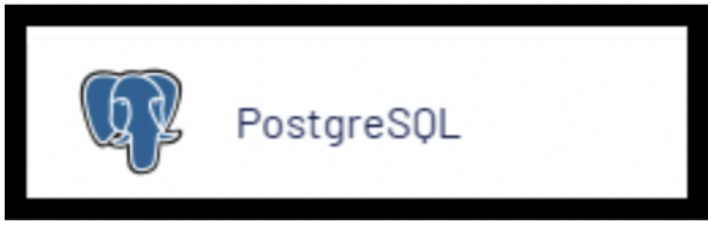


Step 2 - Configure a PostgreSQL catalog

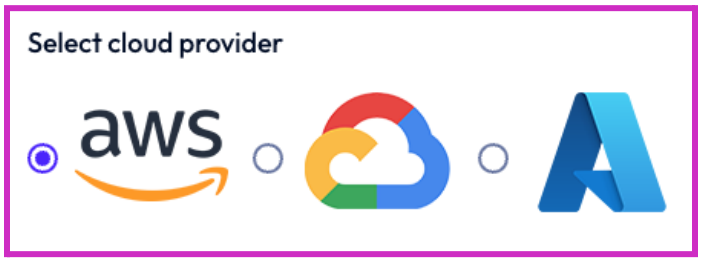
Click **Catalogs** in the menu on the left and then click the **Create catalog** button.



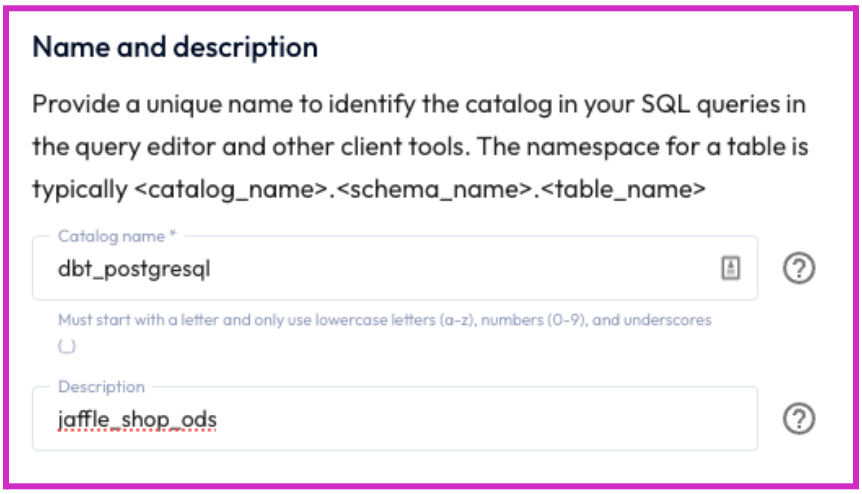
Click the **PostgreSQL** tile.



Ensure the radio button for **aws** is selected.



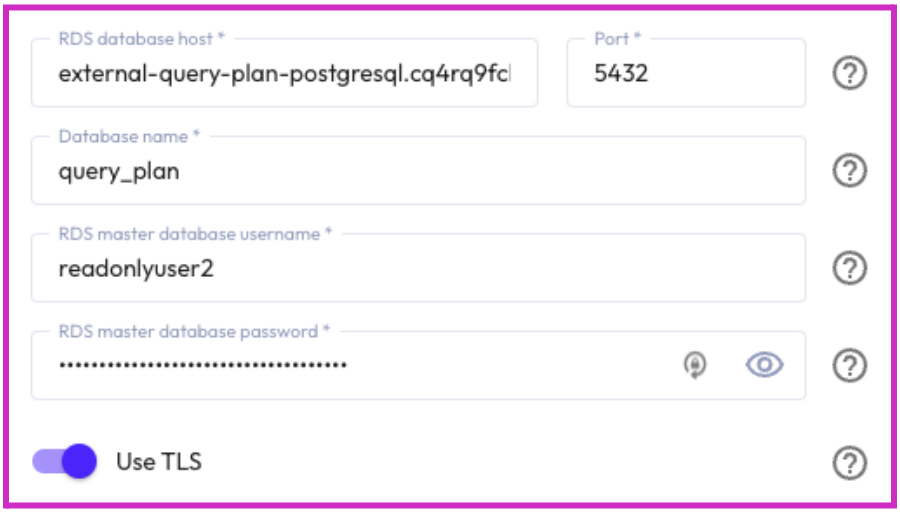
In the box under **Catalog name**, type `dbt_postgresql`. In the box under **Description**, type `jaffle_shop_ods`.



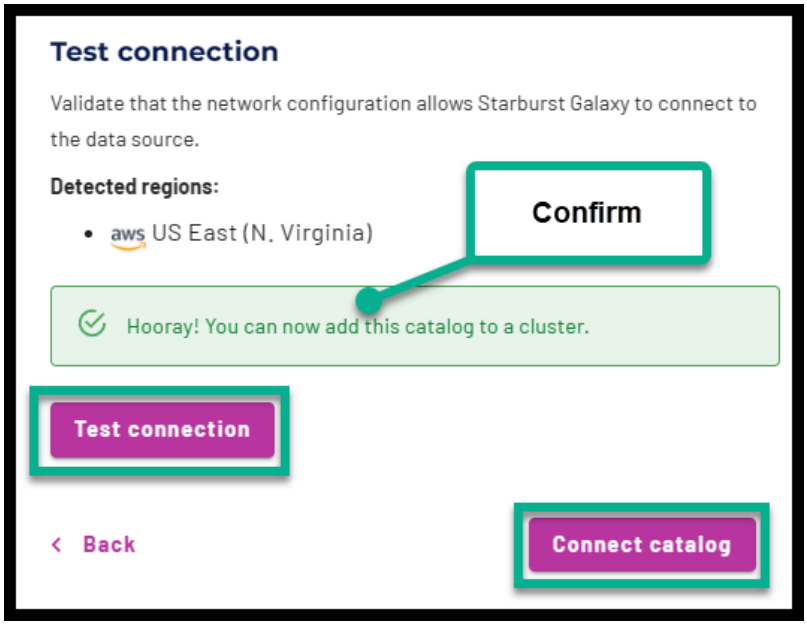
Select the radio button for **Connect directly**.

Enter the following configuration details and ensure the slider for **Use TLS** is to the right.

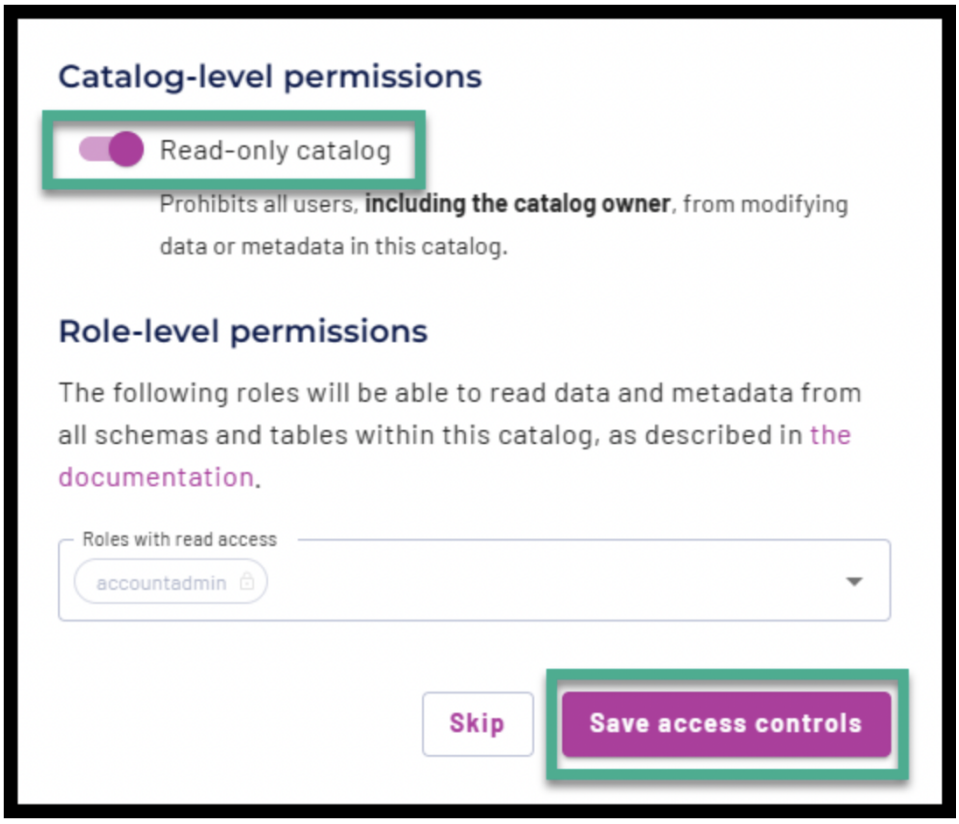
RDS database host	`external-query-plan-postgresql.cq4rq9fclcvt.us-east-1.rds.amazonaws.com`
Port	`5432`
Database name	`query_plan`
RDS master database username	`readonlyuser2`
RDS master database password	`UPXT9jWPq*vG6UdeRg.h@QZKyJpMoiAQXL*n`



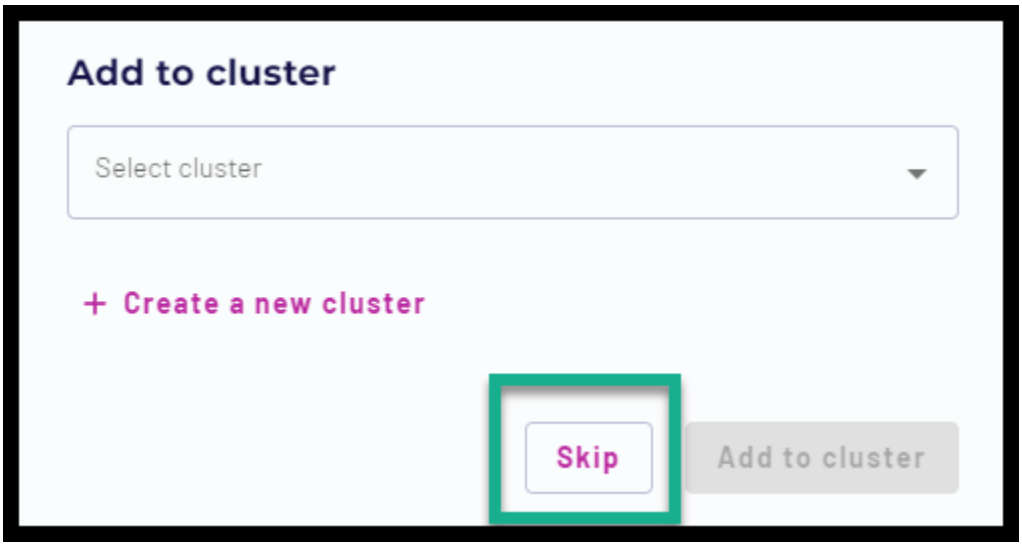
Click the **Test connection** button. Confirm you see the **Hooray! You can now add this catalog to a cluster** message. Click **Connect catalog**.



Move the slider to the right for **Read-only catalog**. Click the **Save access controls** button.

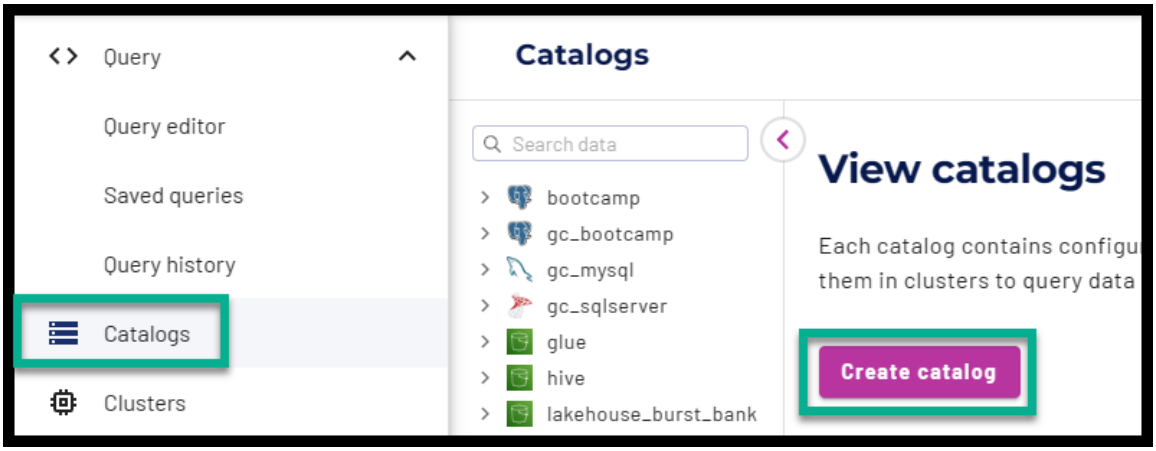


Click the **Skip** button as you will add multiple catalogs to a cluster later.



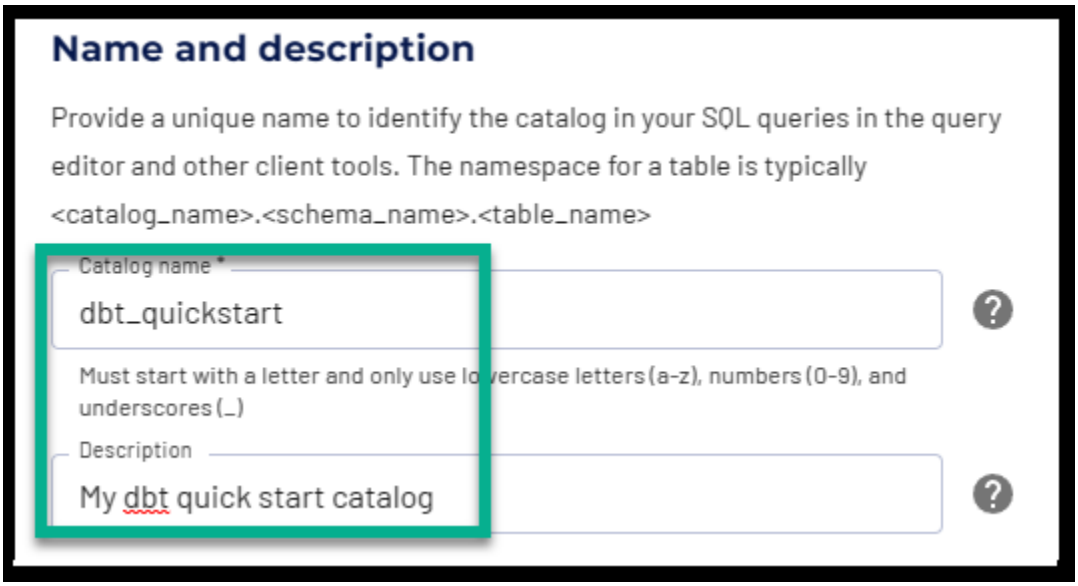
Step 3 - Configure a data lakehouse catalog

Click **Catalogs** in the menu on the left and then click the **Create catalog** button.



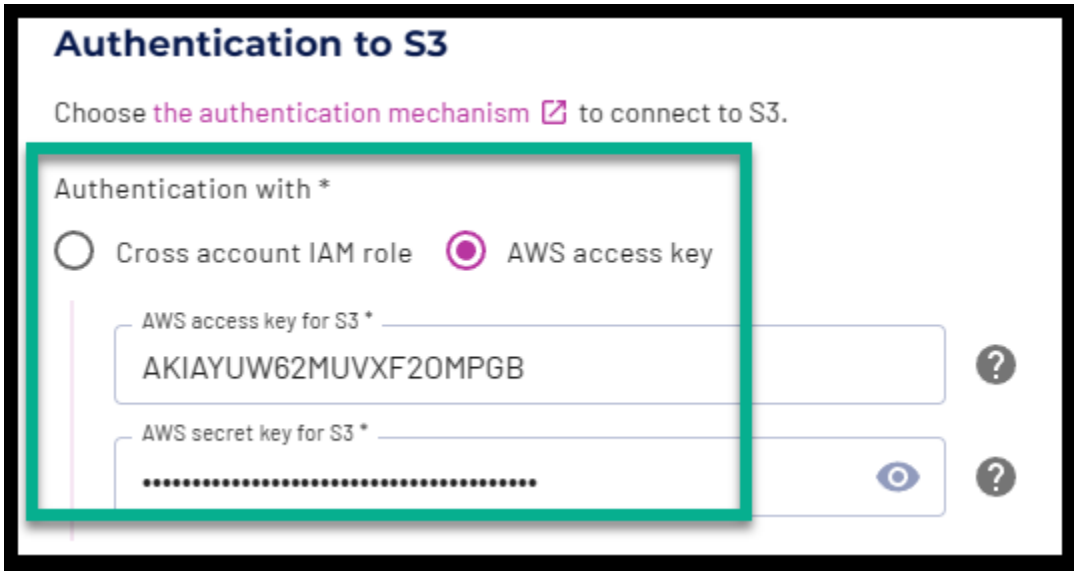
Click the **Amazon S3** tile.

In the box under **Catalog name**, type `dbt_quickstart`. Provide a meaningful **Description**.



Select the radio button for **AWS access key** and enter the following configuration details.

AWS Access key for S3	`AKIAYUW62MUVYMUXUGF4`
AWS secret key for S3	`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`

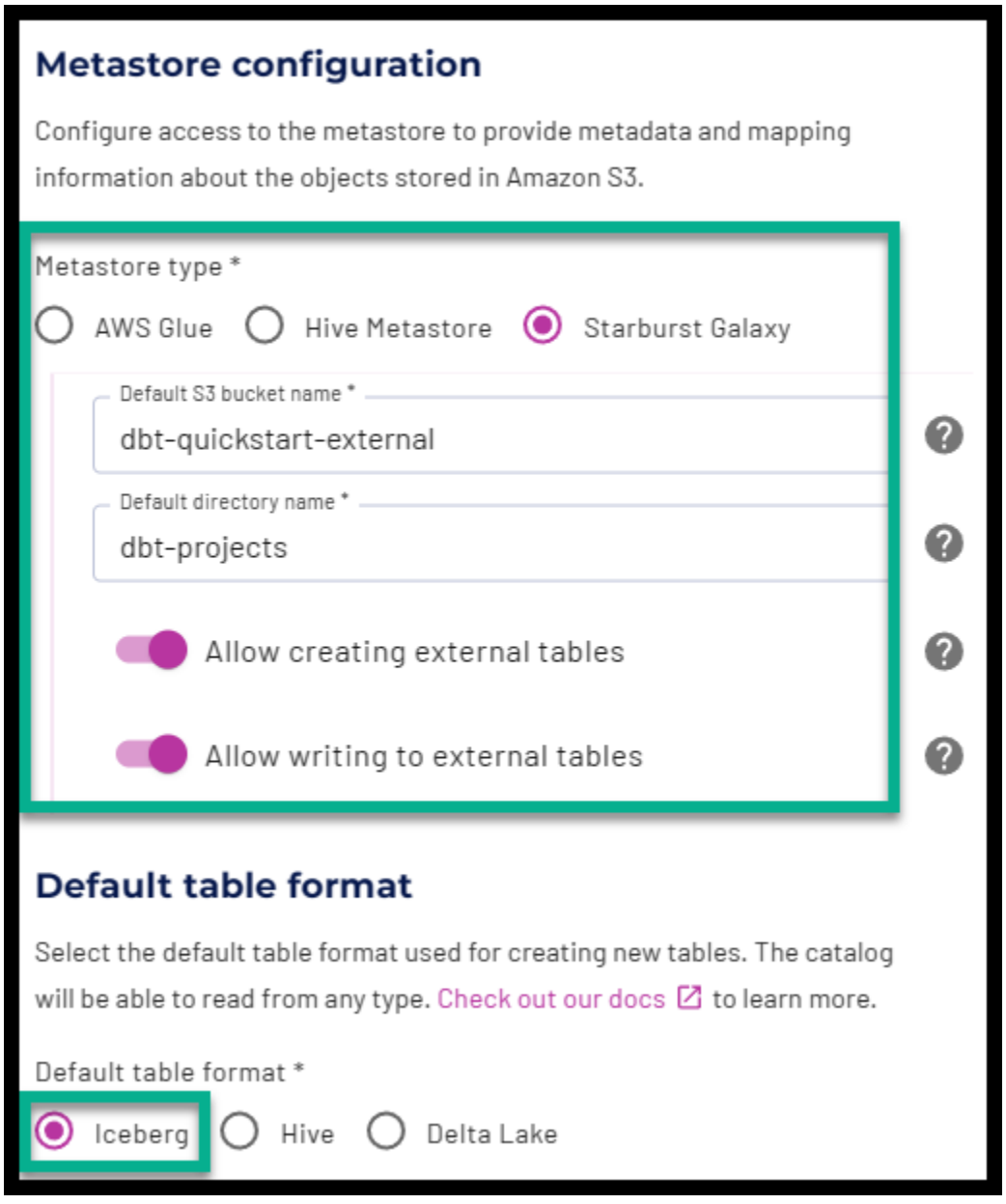


Under the **Metastore type**, select the radio button for **Starburst Galaxy** and enter the following configuration details.

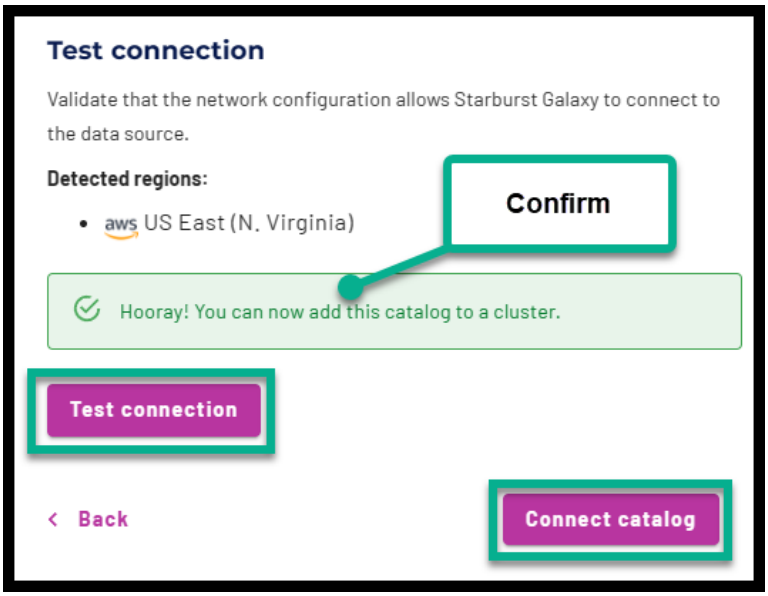
Default S3 bucket for S3	`dbt-quickstart-external`
Default directory name	`dbt-projects`

Move the slider to the right for **Allow creating external tables** and **Allow writing to external tables**.

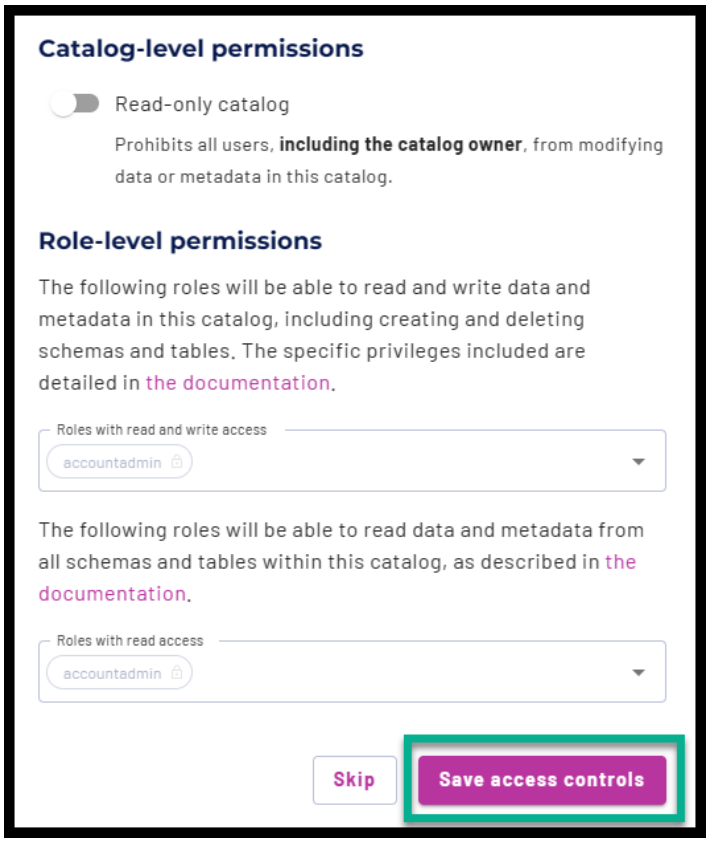
Under **Default table format**, select the radio button in front of **Iceberg**.



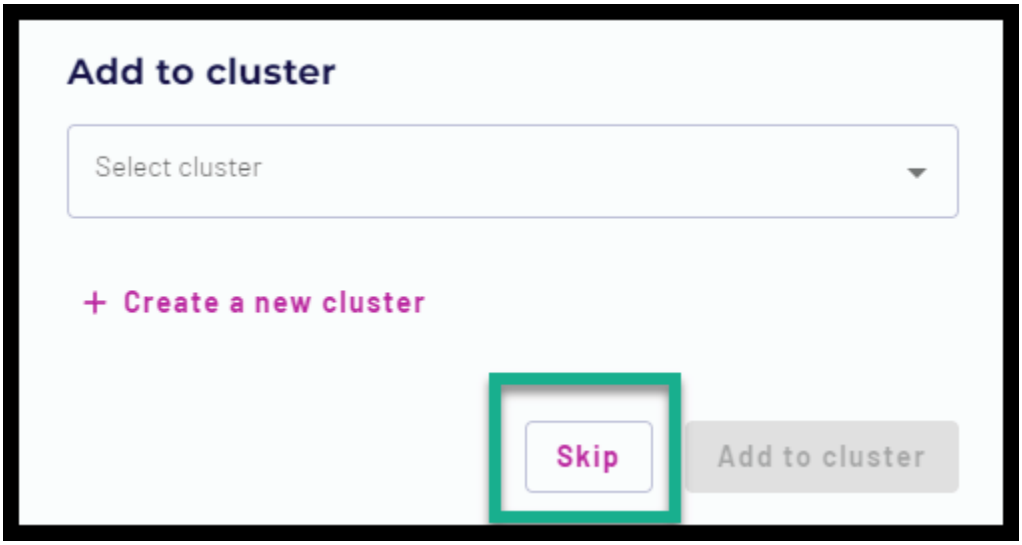
Click the **Test connection** button. Confirm you see the **Hooray! You can now add this catalog to a cluster** message. Click **Connect catalog**.



Click the **Save access controls** button.



Click the **Skip** button as you will add multiple catalogs to a cluster in the next step.

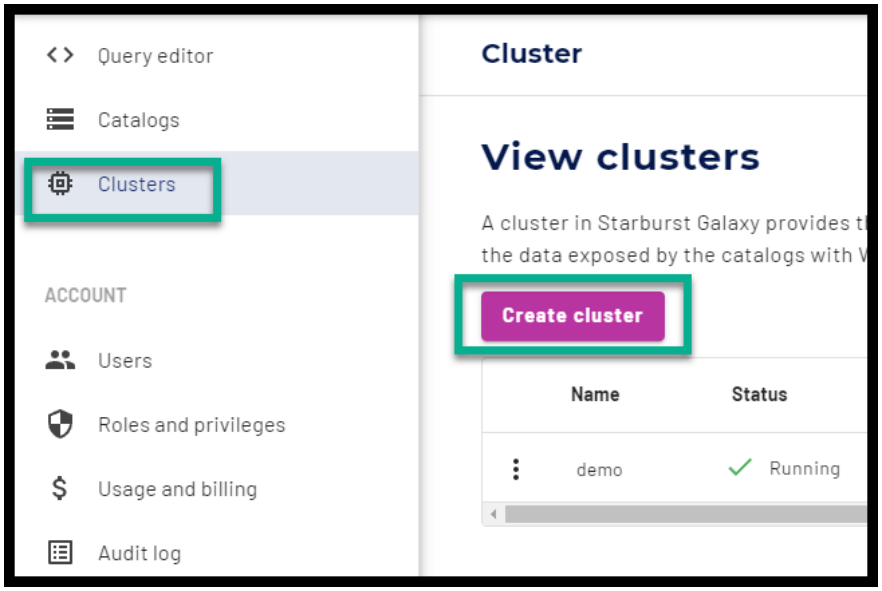


Step 4 - Create a cluster

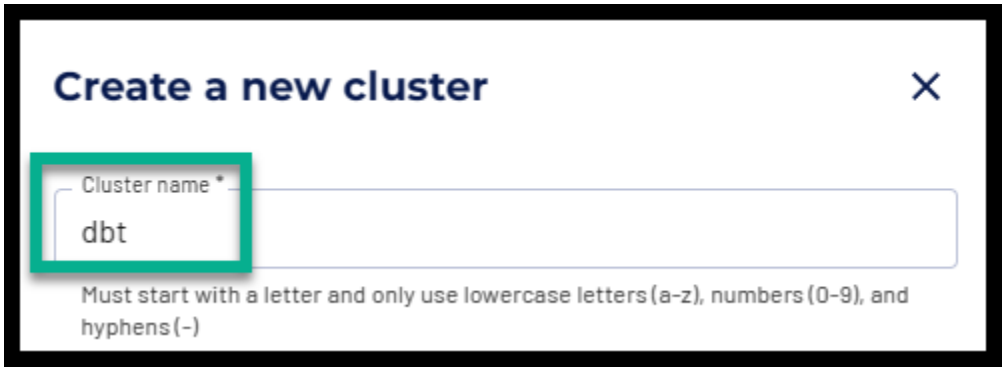
A cluster in Starburst Galaxy provides the resources to run queries against catalogs. In this section, you will create a cluster for the `dbt_quickstart` and `dbt_postgresql` catalogs.

The data sources for both catalogs are in AWS in the US East (N. Virginia) region. In Starburst Galaxy, the data sources and Starburst Galaxy cluster perform best in the same cloud region by default. Therefore, your cluster will be in US East (N. Virginia) (AKA us-east-1).

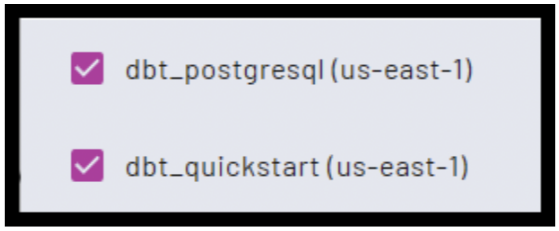
Click **Clusters** in the menu on the left and then click the **Create cluster** button.



To identify the cluster, enter `dbt` in the **Cluster name** textbox.

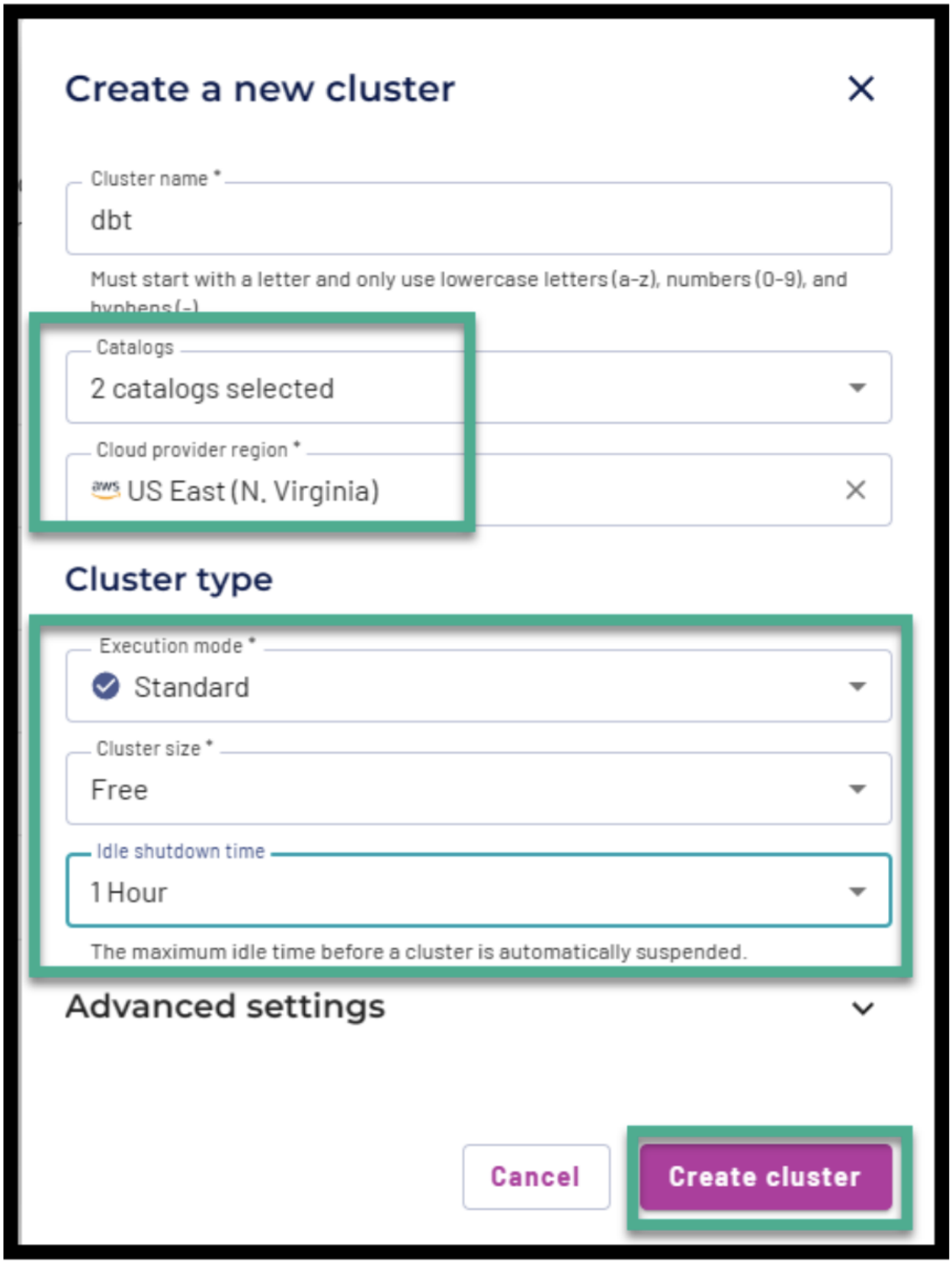


To choose which catalogs to be accessed by this new cluster, click the drop-down arrow in the box under **Catalogs**. Check the boxes in front of `dbt_quickstart` and `dbt_postgresql` catalogs you created earlier. Click outside the selection box to get back to the wizard screen.

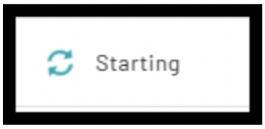


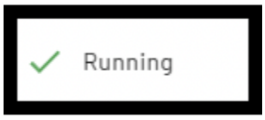
Add the remaining cluster details as identified below.

- In the box under **Cloud provider region**, click the drop-down and select the only region you can. Starburst Galaxy automatically determines the only region you can select based on the catalogs you select.
- In the box under **Execution mode**, select **Standard**.
- In the box under **Cluster size**, select **Free**.
- Choose the **Idle shutdown time** you would like.
- Click **Create cluster**.



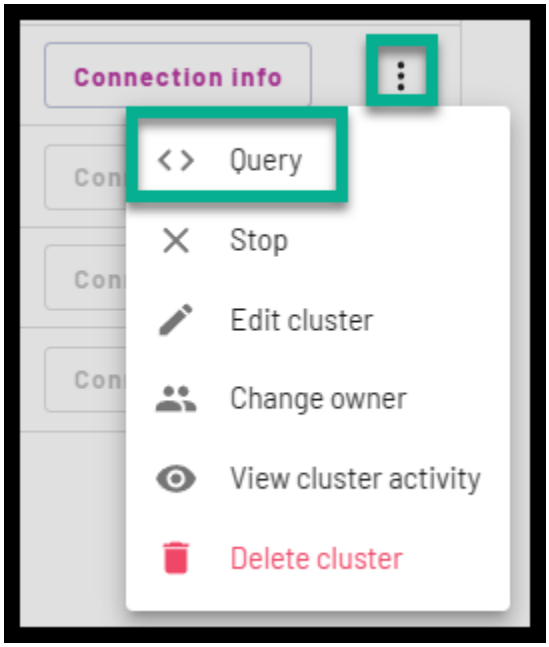
Notice that the cluster will automatically begin **Starting** and shortly be identified as **Running**.





The cluster list can be utilized in the future to restart a stopped cluster.

Click the ellipses to view the drop-down menu and select **Query**. This takes you to a new worksheet in the **Query editor** with your cluster already selected.



You are now ready to run SQL statements to create and/or query datasets.

END OF LAB EXERCISE

Lab 2: Discover & review the land zone datasets

Estimated completion time

- 15 minutes

Learning objectives

- In this lab you will review three land zone datasets. Two will be existing tables from a database. You will leverage the schema discover feature of Starburst Galaxy to automatically create a table based on data you direct the setup wizard to investigate. Lastly, we will peer into the data quality features of Starburst Galaxy.

Prerequisites

- Lab 1 - Create Starburst Galaxy account and data catalogs

Activities

1. Start the cluster
2. Search for existing datasets
3. Use schema discovery on the data lake dataset
4. Validate data quality

Step 1 - Start the cluster

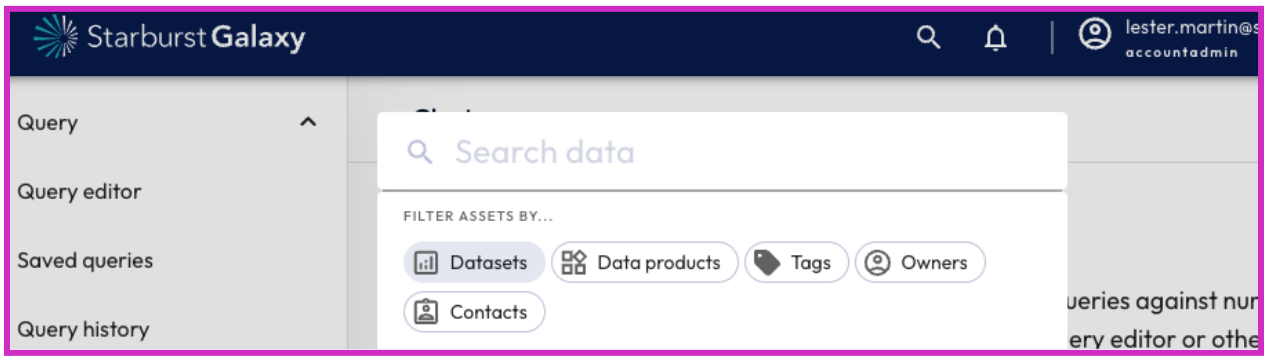
Click **Clusters** from the menu on left and then review the status of the `dbt` cluster.

If identified as **Suspended**, click on **Resume** and wait until the cluster shows as **Running**.

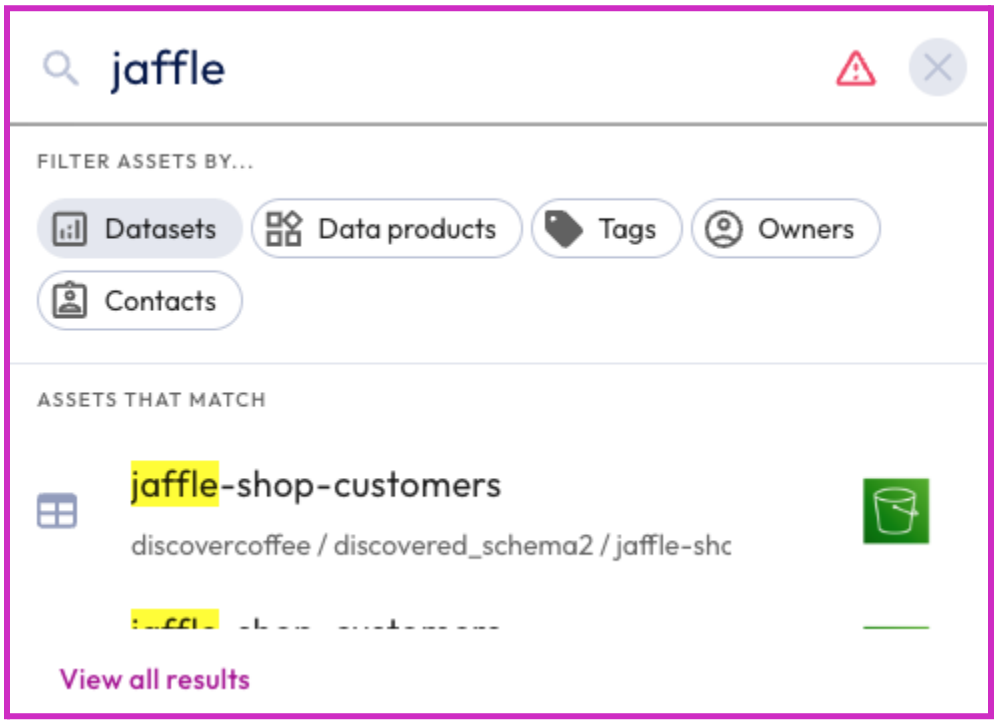
Click on **Query** and then **Query editor** from the menu on the left. Press the **+** on the editor tabs to create an empty worksheet.

Step 2 - Search for existing datasets

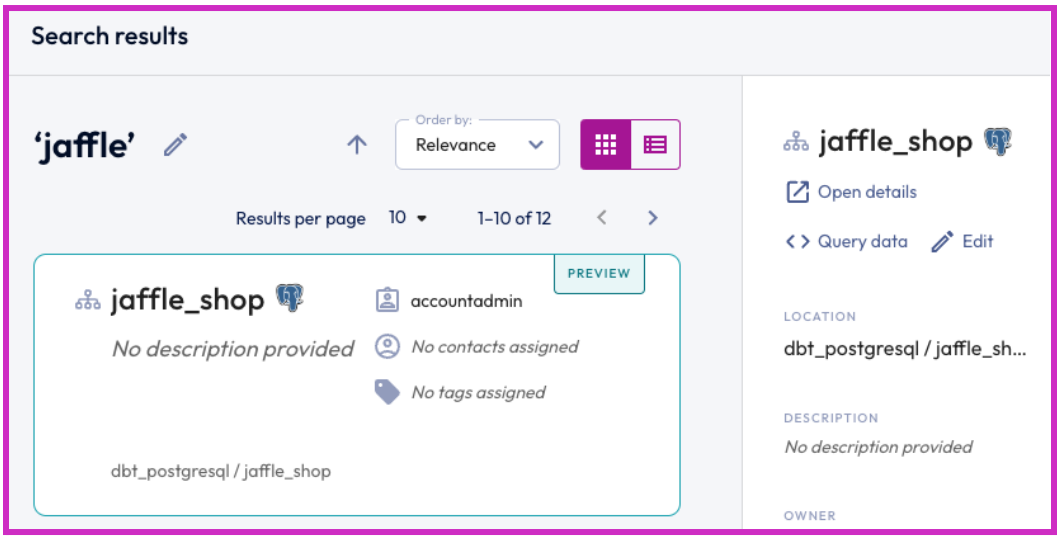
On the top blue bar click on the **magnifying glass icon** to open the search pop-up box.



Type `jaffle` where you see **Search data** and then click on **View all results** at the bottom of the pop-up.



In the **Search results** pane find the entry (likely the first one) that lists `dbt_postgresql/jaffle_shop` in its lower left corner.



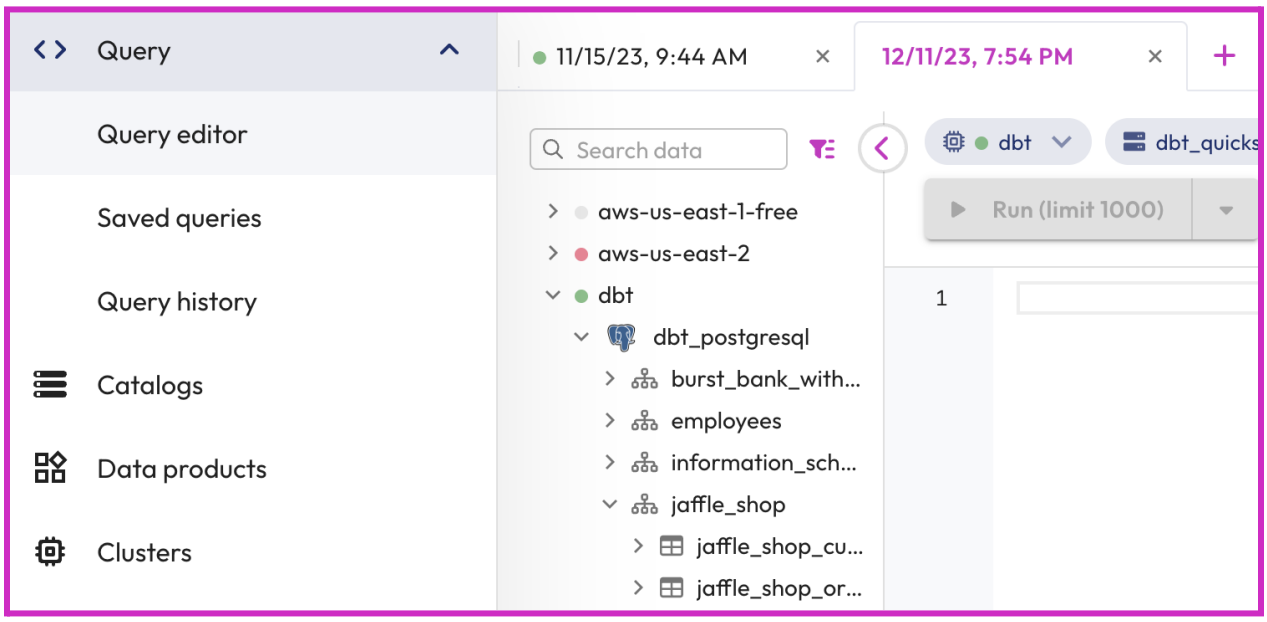
Click on **Open details** as shown in the upper right corner of the previous screenshot. This takes you to the **Catalogs** UI where you can see two tables in this `dbt_postgresql.jaffle_shop` schema.

Click on `jaffle_shop_customers` to see a list of columns.

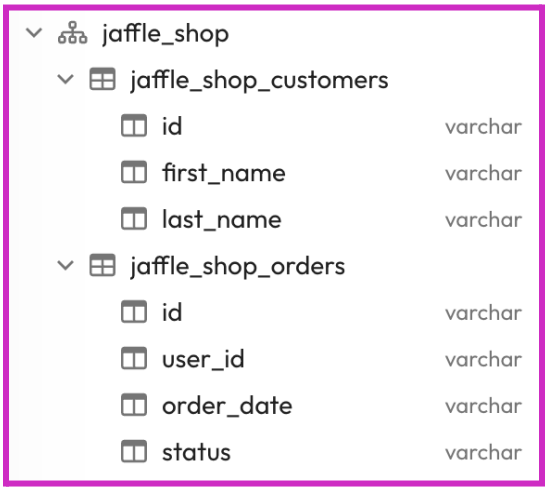
Column	Type	Nullable	Default
first_name	varchar	yes	NULL
id	varchar	yes	NULL
last_name	varchar	yes	NULL

Click on the **vertical ellipses** to the right of the table name near the top of the page. Then select **Query data**.

You are now in the Query editor and the middle pane has `dbt`, `dbt_postgresql`, and `jaffle_shop` expanded so that you can see the two tables in this schema.



Expand each table to see the columns that make up the customers and orders tables.



Run the following SQL for a quick look at the data in these two tables. They are stored in the RDBMS system that is a replicated copy of the point-of-sale system's database.

```
SELECT * FROM jaffle_shop_customers;
SELECT * FROM jaffle_shop_orders;
```

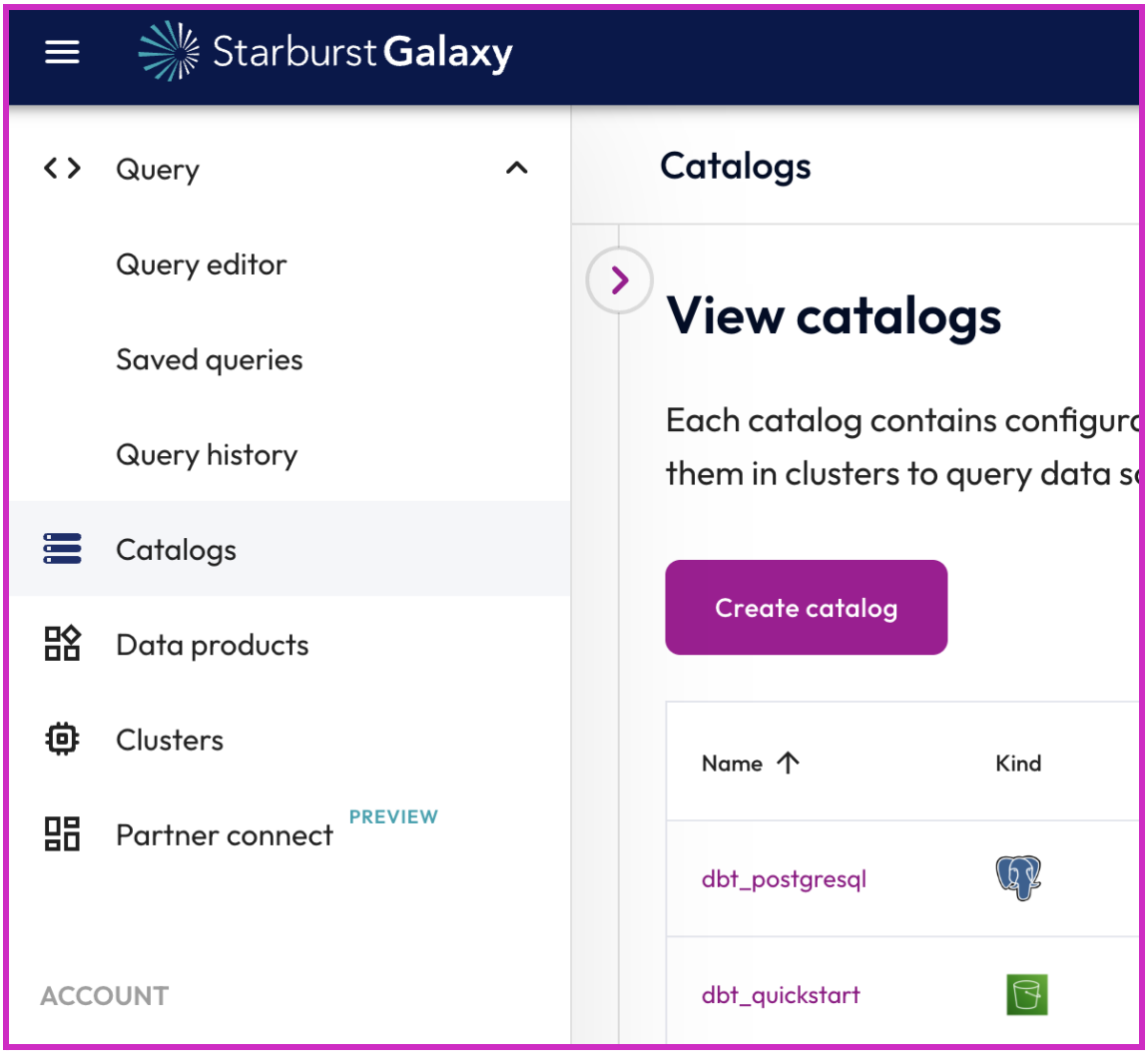
These tables are the start of our land zone datasets. They benefit from Starburst Galaxy's ability to simply connect to the data where it lives instead of needing to copy it first.

Step 3 - Use schema discovery on the data lake dataset

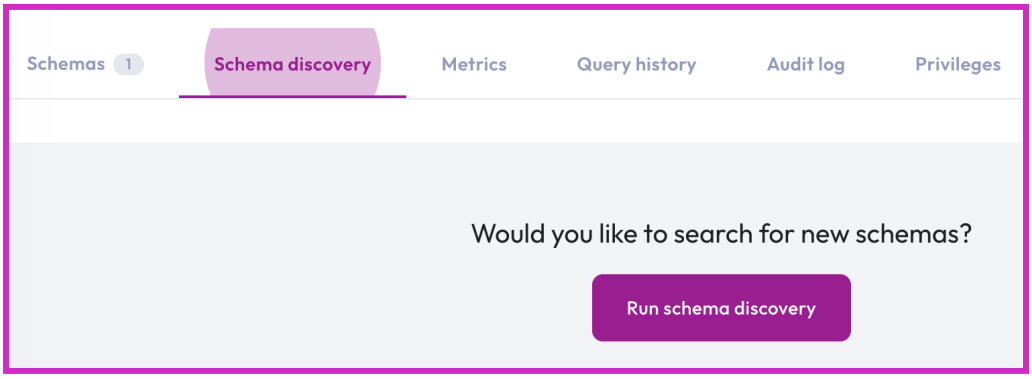
You have been notified that there is a third dataset already ingested onto our data lake. It is at the following location.

While you could manually investigate the contents of this folder for the file format and structure of the table that could be created to align with this dataset, you could also use Starburst Galaxy's schema discovery feature to do this automatically instead. This step will walk you through the process.

Click **Catalogs** in the menu on the left. In the far-right pane labeled with **View catalogs**, click on `dbt_quickstart` in the list of catalogs below the **Create catalog** button.



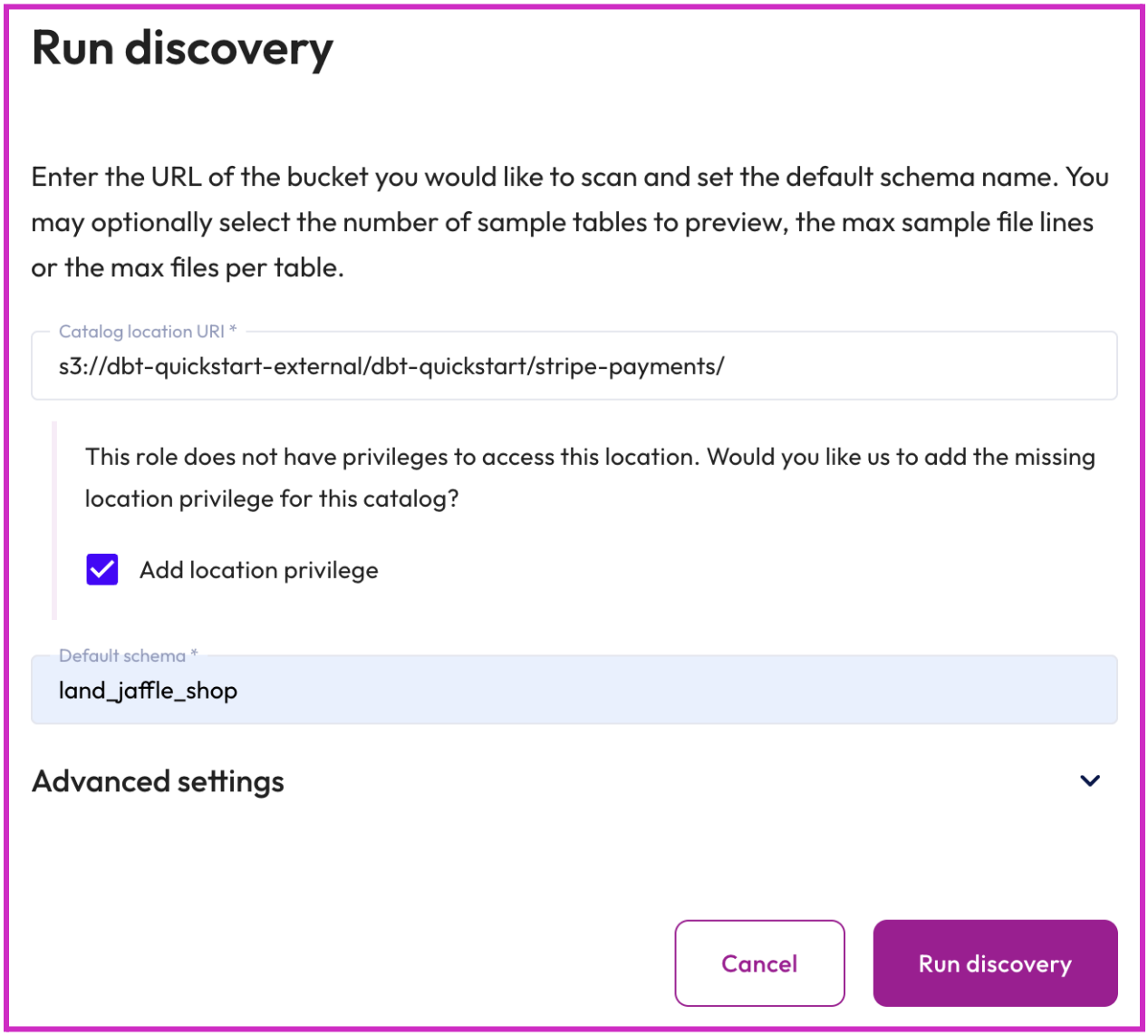
On the **Schema discovery** tab, click the **Run schema discovery** button.



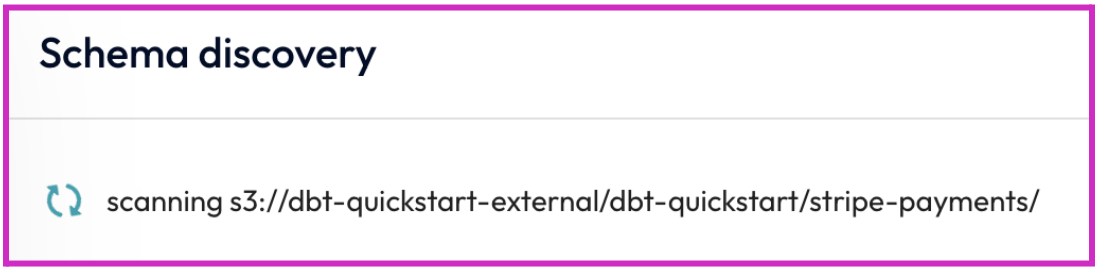
In the **Run discovery** pop-up window, set the following configuration and then press the **Run discovery** button.

Catalog location URI	`s3://dbt-quickstart-external/dbt-quickstart/stripe-payments/`
Default_schema	`land_jaffle_shop`

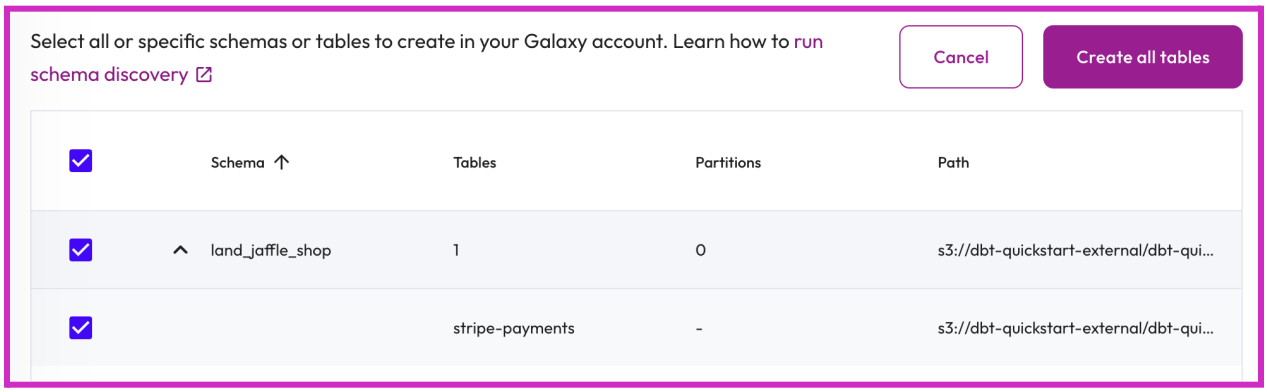
Note: There is a message between the two previous fields that indicates we need to **Add location privilege**. Leave this box checked.



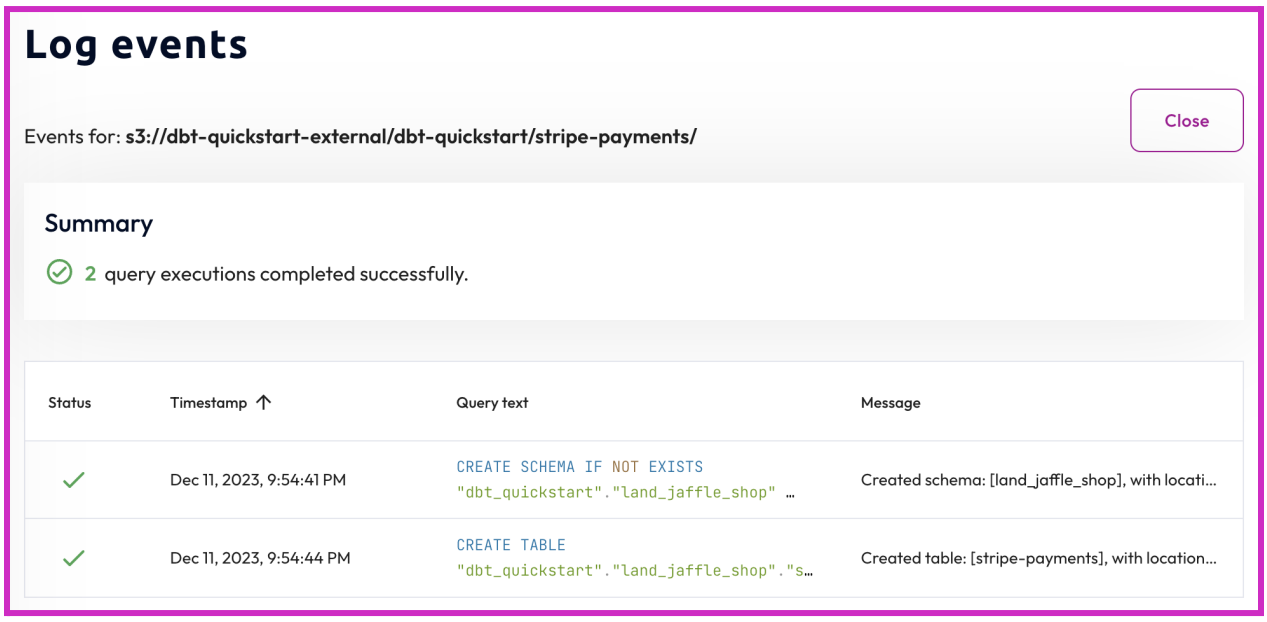
You will see this while the process is running.



Once complete, toggle the `land_jaffle_shop` **Schema** to see that a `stripe-payments` table has been identified. Select the checkbox for the single table found and notice that all checkboxes are now selected. Click on **Create all tables**.



A **Log events** page will surface identifying that `CREATE SCHEMA` and `CREATE TABLE` SQL statements were run.



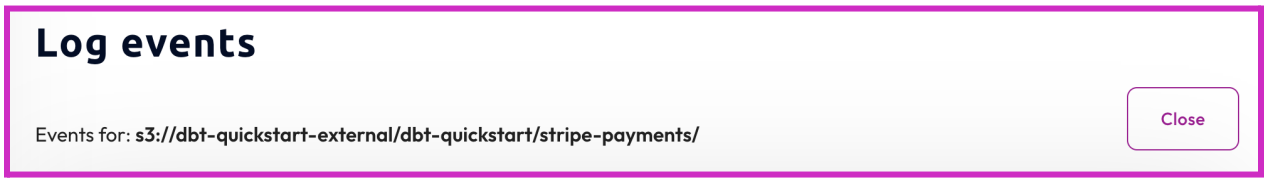
Click on the `CREATE TABLE` SQL statement under the **Query text** column to review the full statement.

```
CREATE TABLE "dbt_quickstart"."land_jaffle_shop"."stripe-payments" (
    "id" int,
    "orderid" int,
    "paymentmethod" varchar,
    "status" varchar,
    "amount" int,
    "created" date
)
WITH (
    type = 'hive',
    format = 'TEXTFILE',
    textfile_field_separator = ',',
    textfile_field_separator_escape = '\',
    skip_header_line_count = 1,
    external_location = 's3://dbt-quickstart-external/dbt-quickstart/stripe-payments/'
)
```

An external Hive table was created for the payment data that is stored in a CSV format on the data lake. This was much faster than investigating the dataset and ultimately creating this DDL statement.

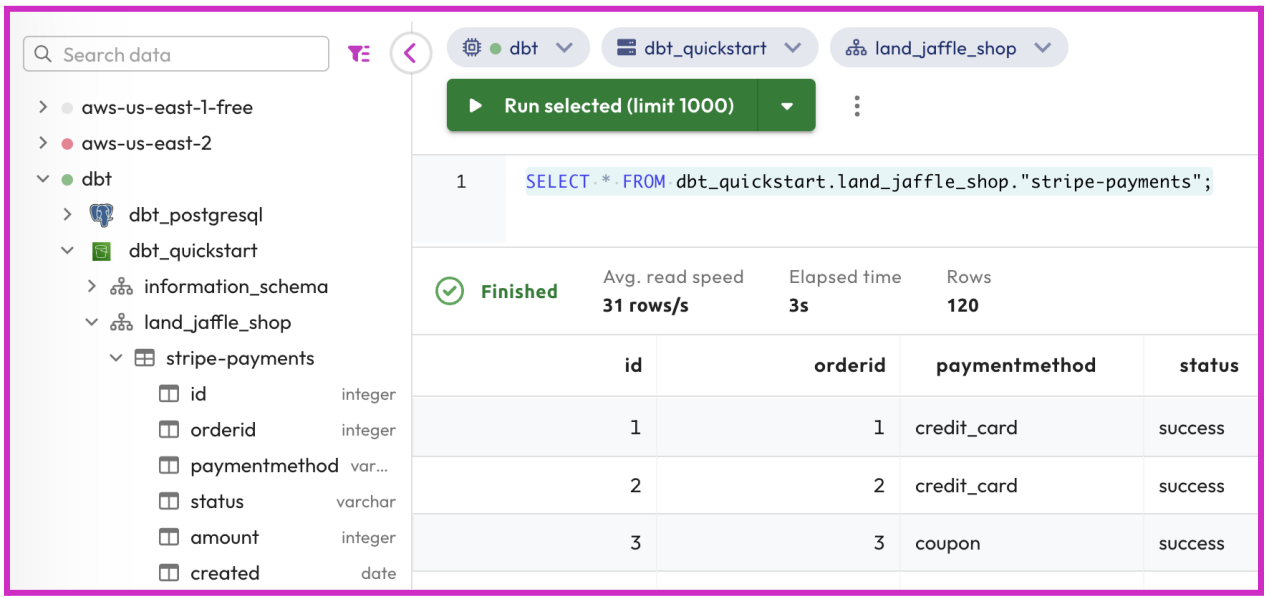
Note: The table was created with the `TEXTFILE` file format. This allows more precise datatypes than the `CSV` file format open which only supports `varchar`. The schema discovery feature effectively selected the most appropriate datatypes. This will help when we create the structure zone's table of this information by limiting or even eliminating the need to cast columns to their most appropriate datatype.

Close the review window by clicking on the **X** in the upper right corner. Click the **Close** button back on the **Log events** page to end the schema discovery process.



Return to the **Query editor** and expand the `dbt_quickstart.land_jaffle_shop` schema's table to review this list of columns. Run a simple query to perform a quick inspection that the schema discovery process worked.

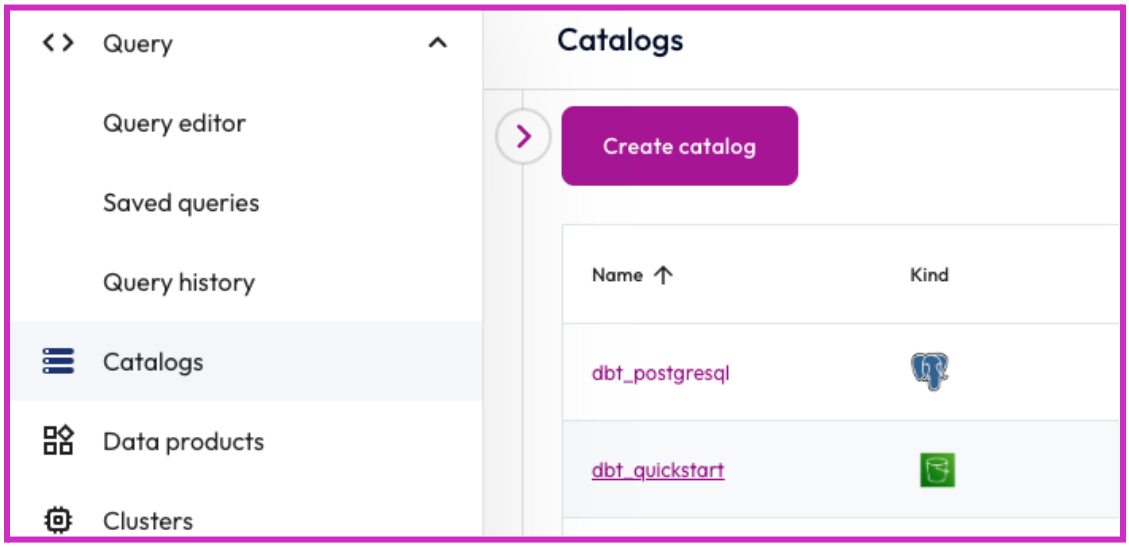
Note: Enclose the `stripe-payments` table with double quotes as the dash included in the table name will cause a query execution error without them. We will correct this in the structure zone's table for payments.



This new external table is the third dataset that makes up our land zone datasets. Again, the automated schema discovery process was much faster than investigating the dataset and ultimately creating the DDL to define all column names and types.

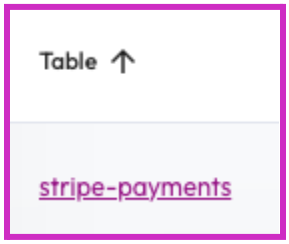
Step 4 - Validate data quality

Click on **Catalogs** in the left nav and under the **Create catalog** button, click on `dbt_quickstart` in the list of catalogs presented.

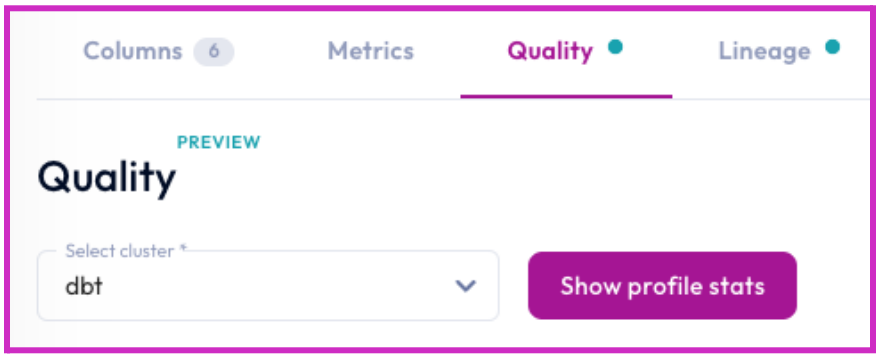


Click on the `land_jaffle_shop` **Schema**.

Click on the `stripe-payments` **Table**.



Click on the **Quality** tab and choose `dbt` for the **Select cluster** pulldown before pressing the **Show profile stats** button.



After reviewing the **Column profile** section, we have decided these statistics seem within normal limits.

Column ↑	Data type	Not null vs null %	Max value	Min value
amount	integer	100%	3000	0
created	date	100%	2018-04-09	2018-01-01
id	integer	100%	120	1
orderid	integer	100%	99	1
paymentmethod	varchar	100%	NULL	NULL
status	varchar	100%	NULL	NULL

END OF LAB EXERCISE

Lab 3: Create dbt Cloud account and connect to Starburst Galaxy

Estimated completion time

- 10 minutes

Learning objectives

- You will create a dbt Cloud account and then configure it to connect to your Starburst Galaxy cluster. Finally, you will create a repository to hold the code you will soon create with dbt Cloud.

Prerequisites

- Lab 2 - Discover & review the land zone datasets

Activities

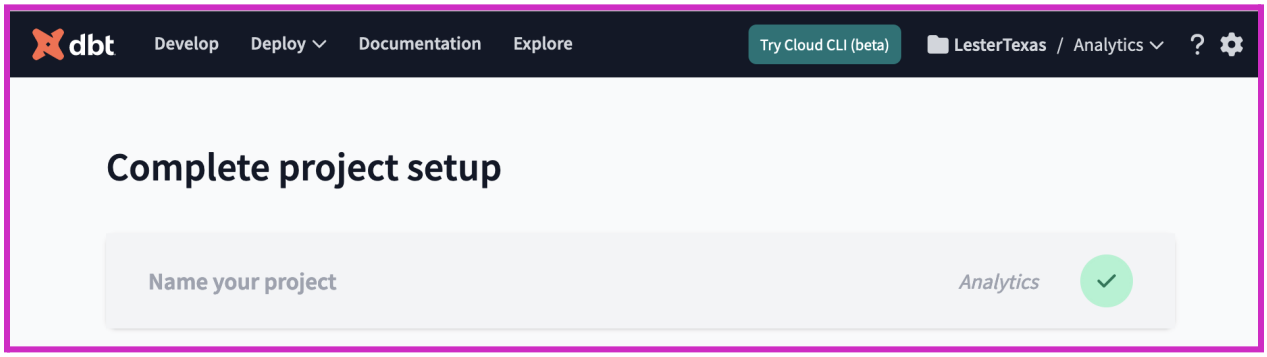
1. Register for a dbt Cloud account
2. View the Starburst Galaxy connection information
3. Configure your environment
4. Set up a repository

Step 1 - Register for a dbt Cloud account

dbt has made it easy to get started with a free, 14-day trial account. This lesson walks you through that process. You can easily upgrade to a paid version on your own at a later time. If you already have a dbt account, please feel free to skip this first step.

Navigate to https://www.getdbt.com/, click on the **Create a free account** button, and follow the registration instructions presented.

After finishing the email verification process you will be presented with a **Complete project setup** page.



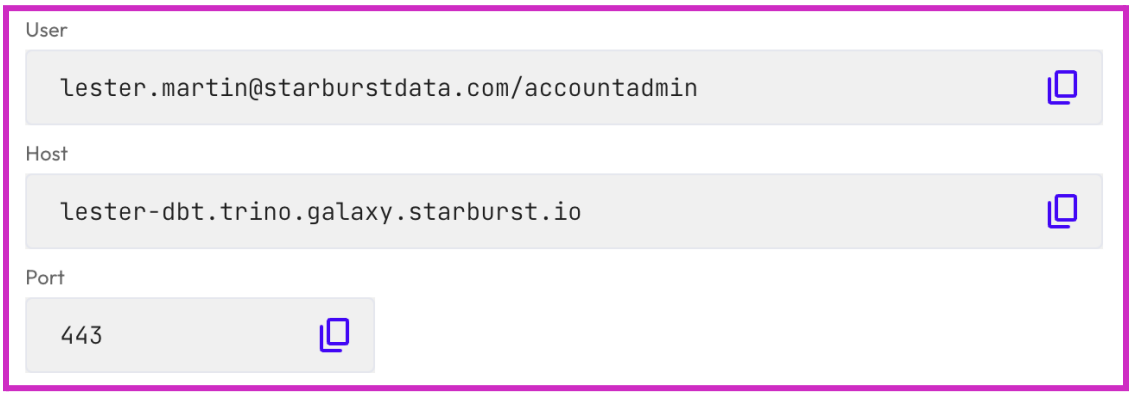
Step 2 - View the Starburst Galaxy connection information

You need to retrieve your Starburst Galaxy cluster's user, host, and port information to connect to dbt Cloud.

Click **Clusters** in the left menu and click on **Connection info** button on the far right of the `dbt` cluster entry in the list.

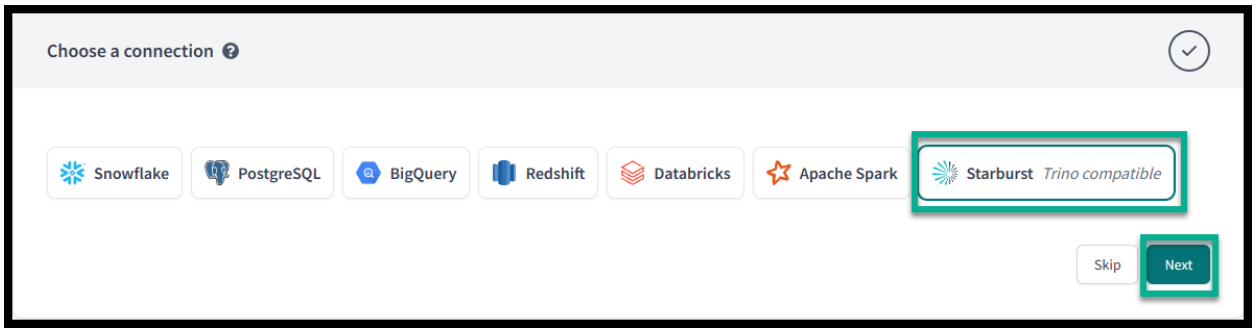
In the **Connection information** pop-up that surfaces, select **dbt** from the pulldown in the **Select a client for custom instructions** section.

Take note of the **User**, **Host**, and **Port** information for the next step.



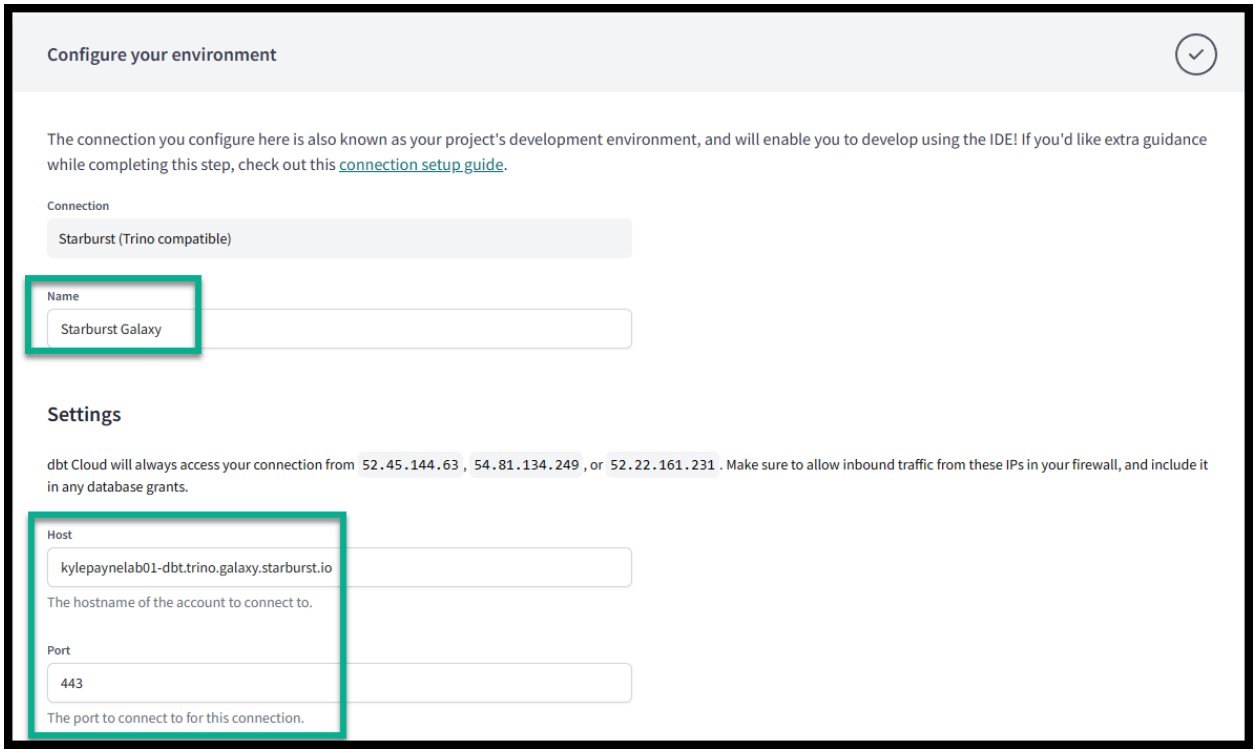
Step 3 - Configure your environment

Switch to the dbt console. In the **Choose a connection** section click **Starbust *Trino compatible*** and then **Next**.



Note: Each dbt project is allow a single connection. From a Starburst perspective, this means you can connect to a single Starburst Galaxy account. However, Starburst allows dbt users to connect to and utilize multiple data sources from within a single project because users can connect multiple data sources to a single cluster in Starburst Galaxy.

Provide a meaningful **Name** for your development environment. Copy the **Host** from your Starburst Galaxy cluster's **Connection information**, and paste it into the box under **Host**. Ensure the **Port** is set to **443**.



Copy the **User** from your Starburst Galaxy cluster's **Connection information**, and paste it into the box under **User**. The user name must include your role at the end (ex. `/accountadmin`).

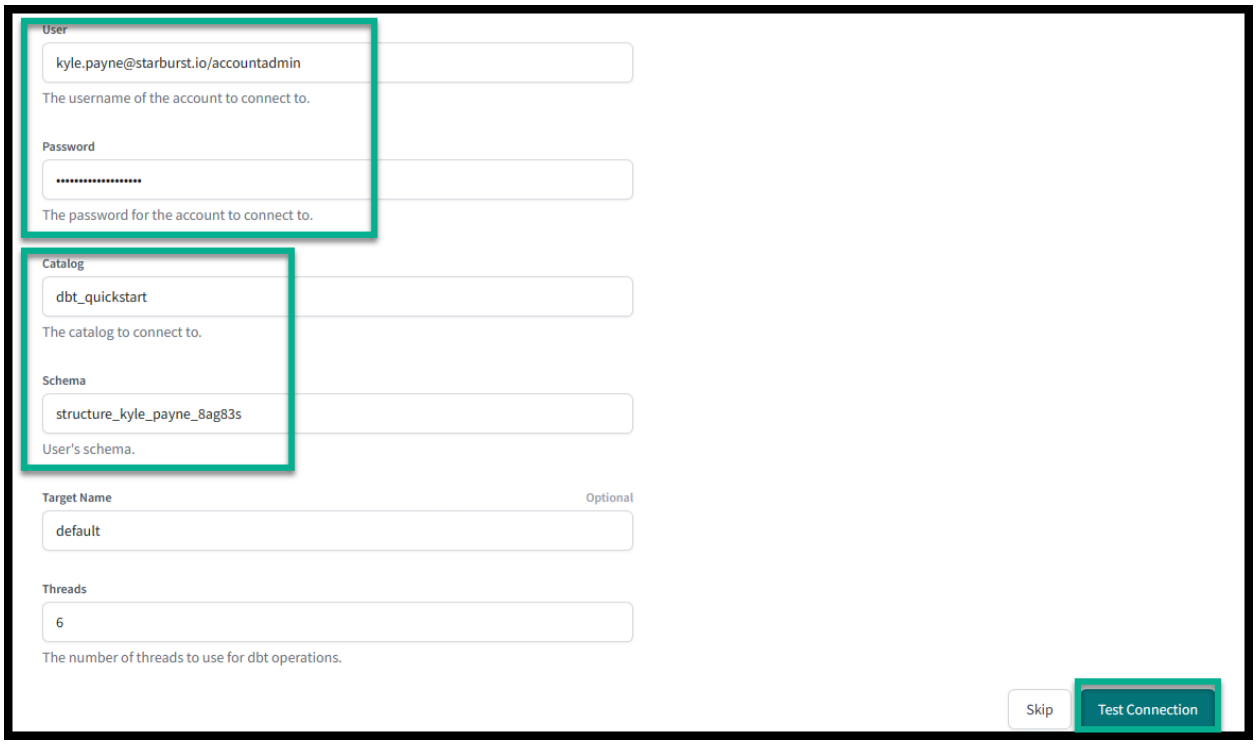
Type in your Starburst Galaxy **Password**. In the box under **Catalog**, type `dbt_quickstart`.

You must provide a globally unique name for the **Schema** because that name will be the directory name in a shared S3 bucket. Use the following guidelines to name the schema.

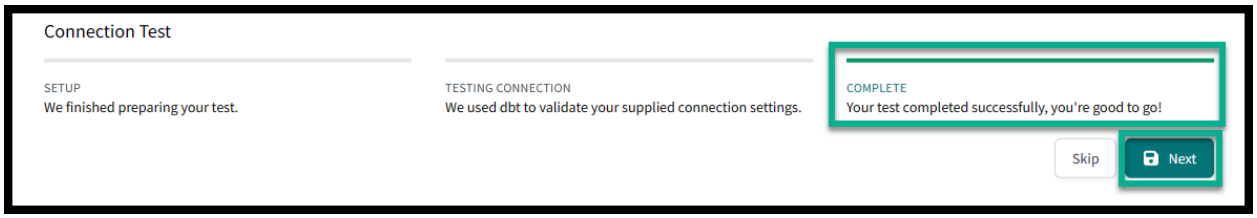
- Start with `structure_`
- Add your first name, followed by an underscore, and then your last name (ex. `kyle_payne`)
- Add another underscore and 6 random characters to the end (ex. `_8ag83s`)
- Your completed schema name should look something like this: `structure_kyle_payne_8ag83s`

When dbt runs for the first time, it will check to see if this schema exists. If it does not exist, dbt will create it for you.

Click the **Test Connection** button.

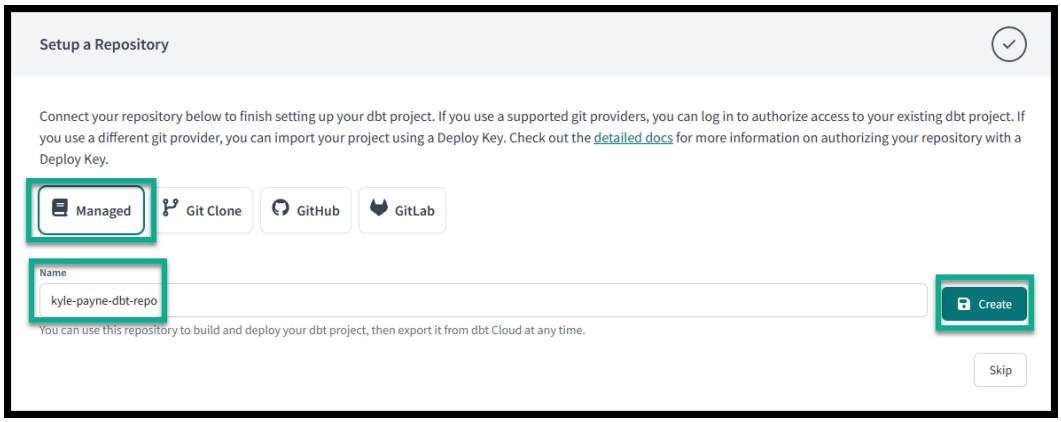


Wait to see the message **Your test completed successfully, you're good to go!** Click **Next**.



Step 4 - Set up a repository

In the **Setup a Repository** section, ensure **Managed** is selected. Provide a meaningful **Name** for your repository (ex. `kyle-payne-dbt-repo`) and click **Create**.



Note: When you develop in dbt Cloud, you can leverage Git to version control your code. To connect to a repository, you can either set up a dbt Cloud-hosted managed repository or directly connect to a supported git provider. Managed repositories are a great way to trial dbt without needing to create a new repository. In the long run, it's better to connect to a supported git provider to use features like automation and continuous integration.

END OF LAB EXERCISE

Lab 4: Build the structure zone with dbt Cloud models

Estimated completion time

- 40 minutes

Learning objectives

- After creating a new dbt project and creating a development branch, you will create the structure zone tables with dbt models.

Prerequisites

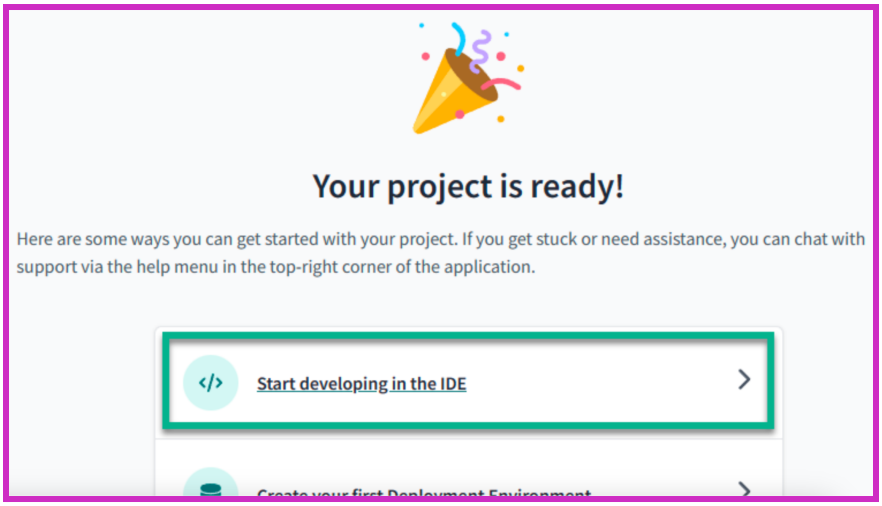
- Lab 3 - Create dbt Cloud account and connect to Starburst Galaxy

Activities

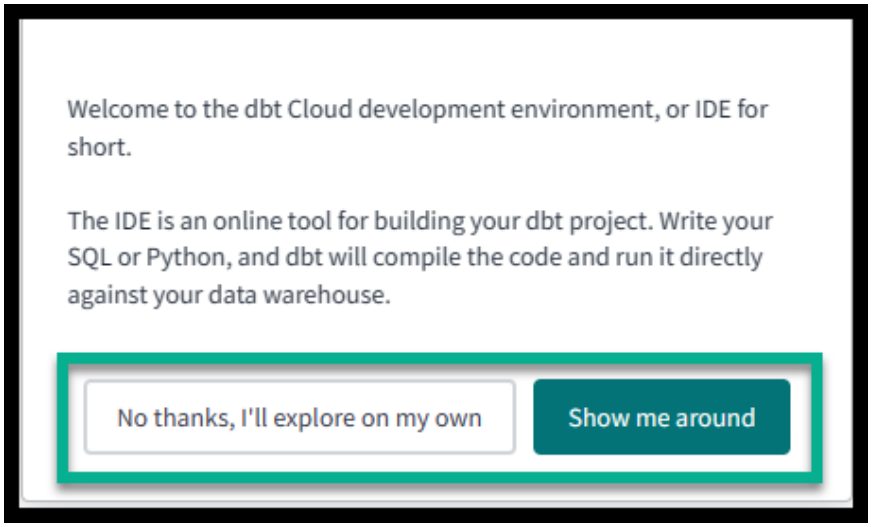
1. Start developing in the IDE
2. Initialize a project
3. Test your connection to Starburst Galaxy
4. Create a branch
5. Delete the example models
6. Edit the dbt project file
7. Create dbt model for structure zone's customers table
8. Query the new table in Starburst Galaxy
9. Create dbt model for structure zone's orders table
10. Create dbt model for structure zone's payments table
11. Add test and documentation to models

Step 1 - Start developing in the IDE

Click **Start developing in the IDE**.

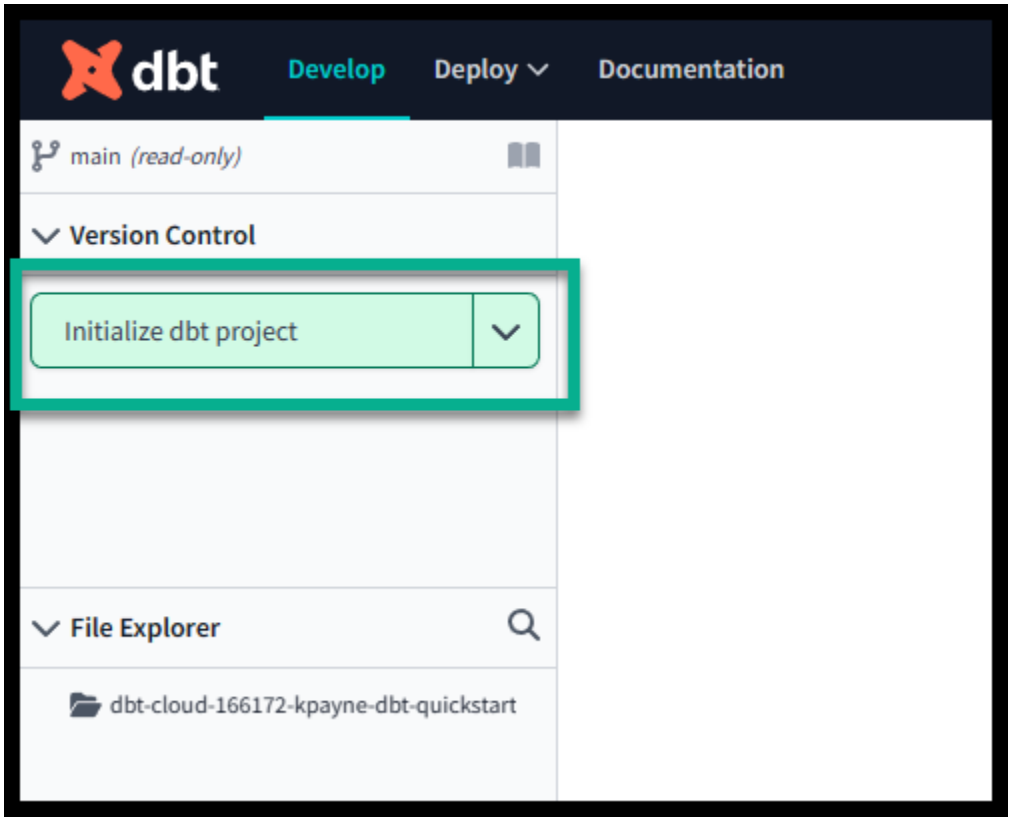


Choose between **Show me around** or **No thanks, I'll explore on my own**. These instructions assume you selected the latter.

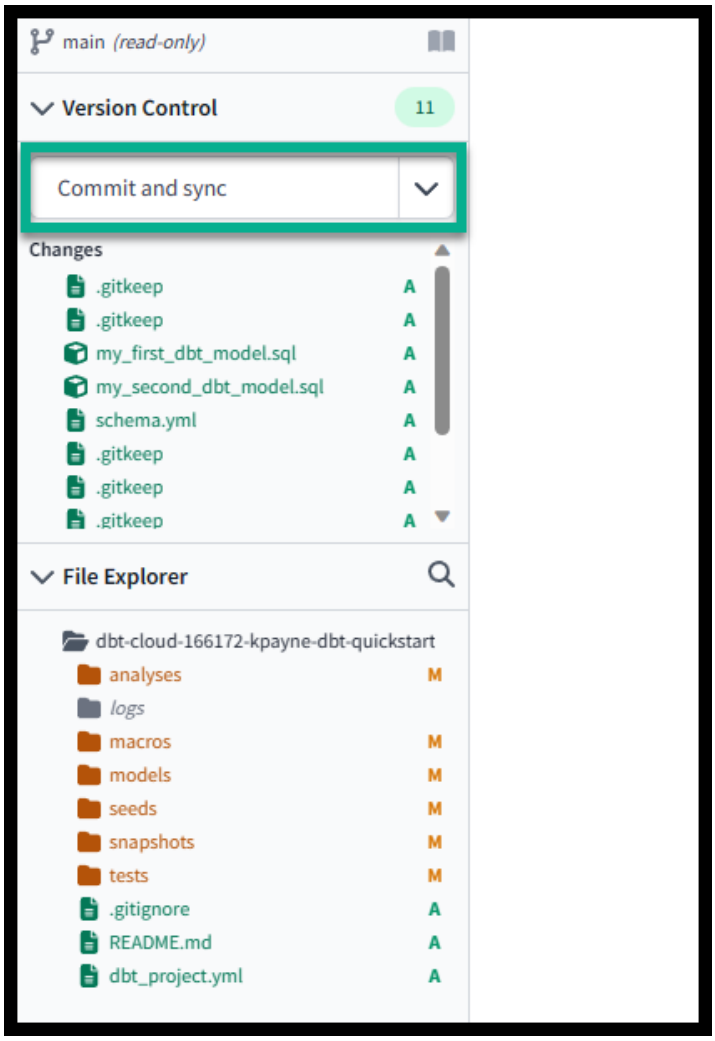


Step 2 - Initialize a project

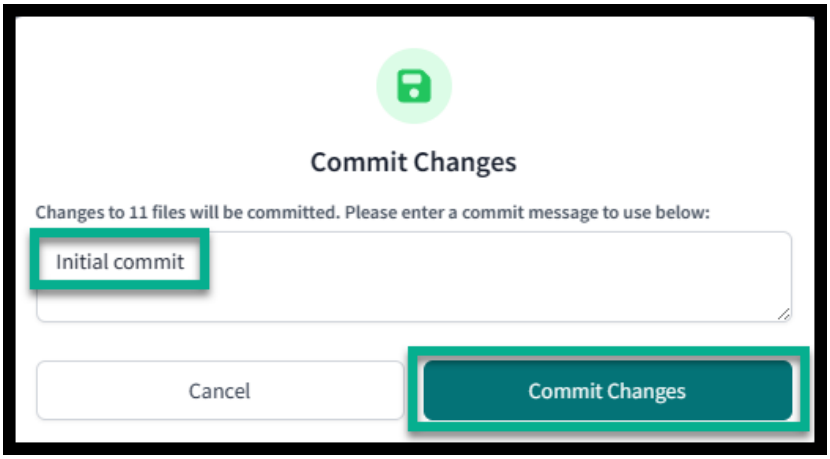
Click **Initialize dbt project**. This builds out the standard dbt folder structure with example models.



Click **Commit and sync**.



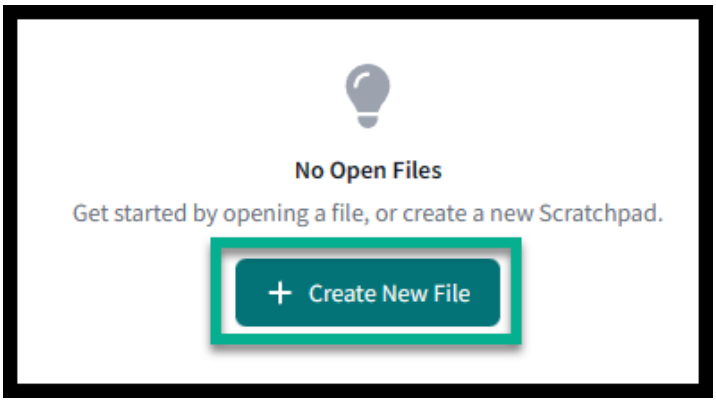
In the box for **Commit message** type `initial commit`. Click **Commit Changes**.



This created the first commit to your managed repo and allows you to open a branch where you can add new dbt code.

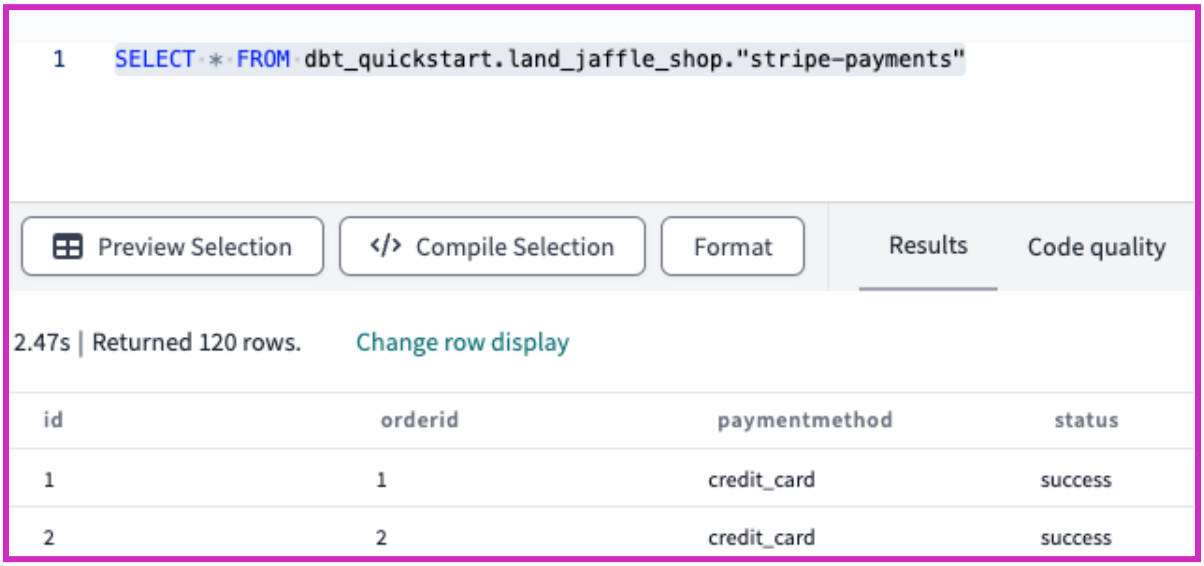
Step 3 - Test your connection to Starburst Galaxy

You can run a simple query to ensure that you can connect to your Starburst Galaxy account. Start by clicking on **Create New File**.



Paste the following SQL into the IDE and click **Preview**.

```
SELECT * FROM dbt_quickstart.land_jaffle_shop."stripe-payments"
```



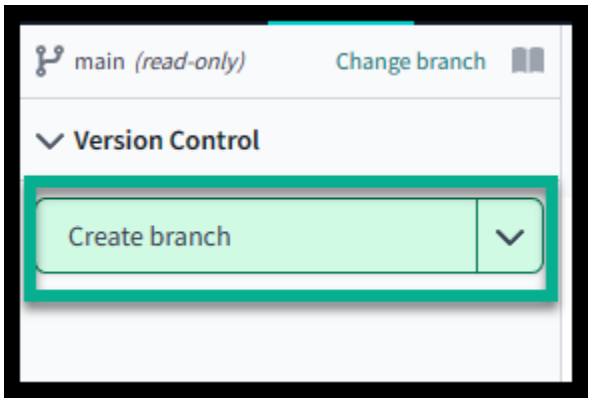
The results presented confirm you can retrieve data from Starburst Galaxy.

Step 4 - Create a branch

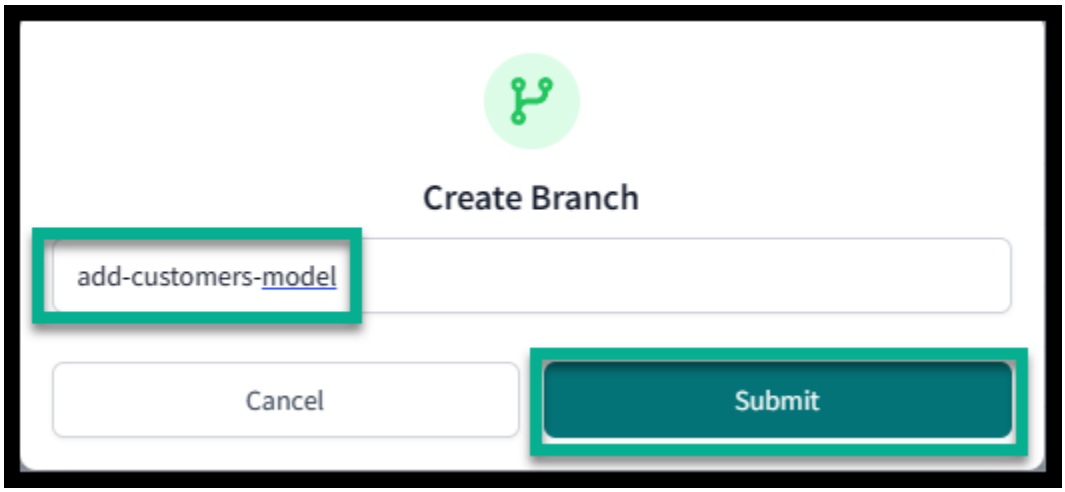
By creating a branch, you are checking out the transformation code for editing. You will create models for the land zone datasets to be transformed into the structure zone.

The land zone datasets in our example are generally presented with high data quality and the transformations will mostly be about technically transforming them into the Iceberg table format. You will add quality tests and documentation to the models as well.

Click **Create branch**.



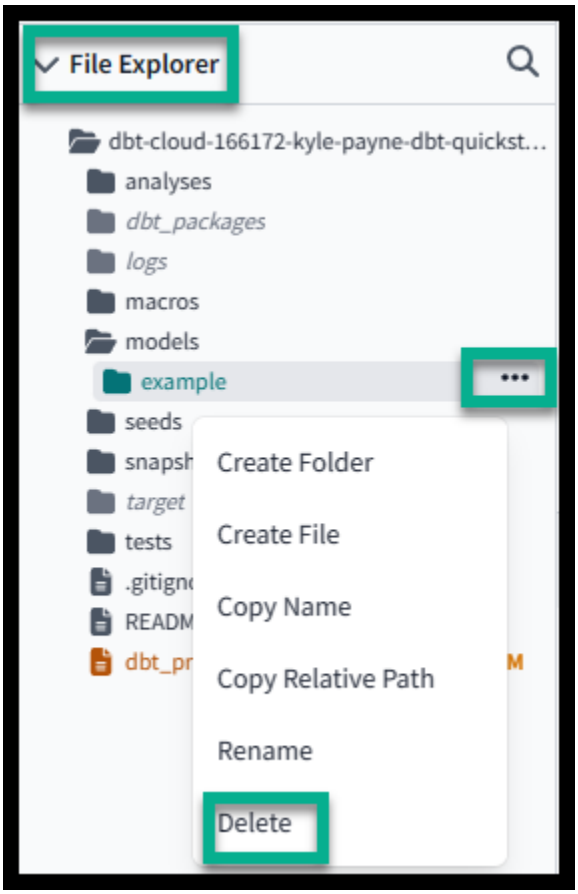
Type `add-customers-model` in the box for the branch name and click **Submit**.



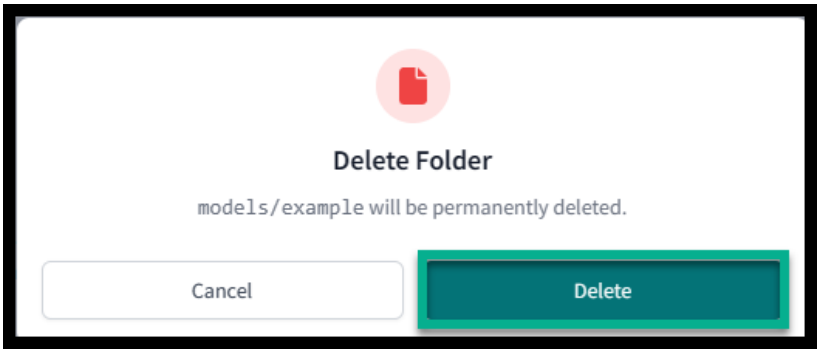
Step 5 - Delete the example models

You do not need the example models provided by dbt for this workshop.

On the bottom-left under **File Explorer**, expand the **models** directory by clicking on it. Hover over the **example** directory. An **ellipses** will appear. Click on the **ellipses** and then click **Delete**.



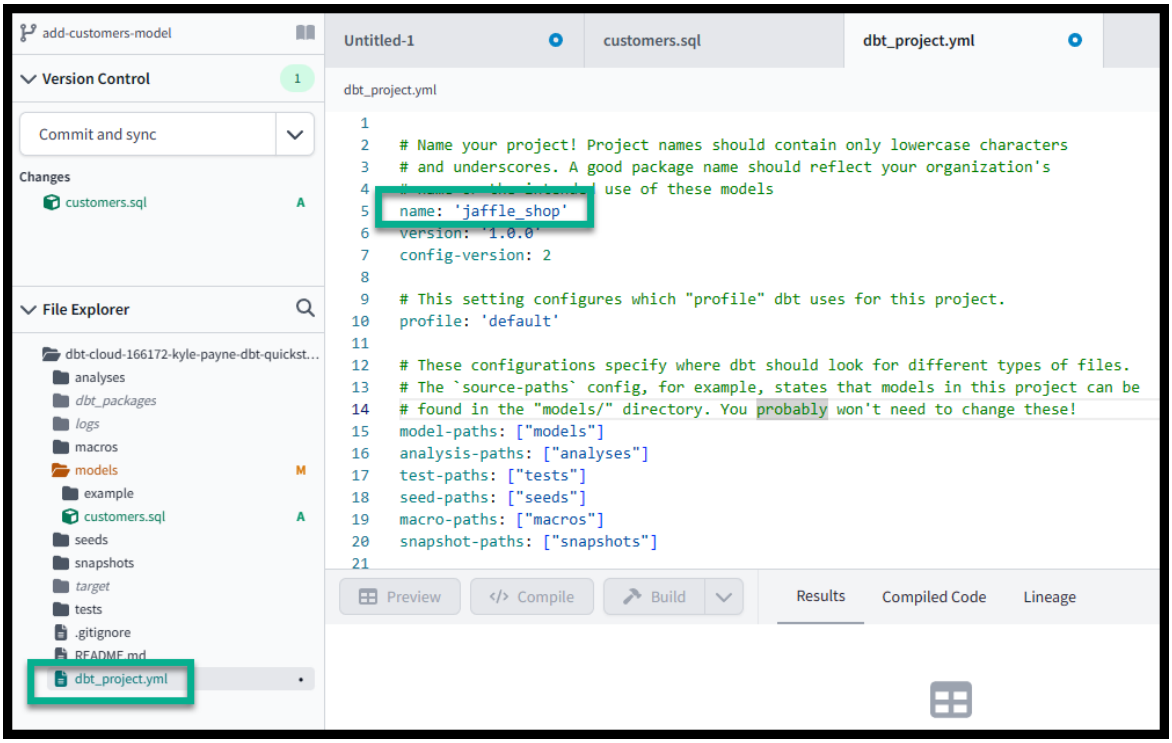
Click **Delete** again.



Step 6 - Edit the dbt project file

Every dbt project needs a `dbt_project.yml` file. It contains important information that directs dbt how to operate on your project.

Under **File explorer**, click `dbt_project.yml` to open it in the editor. Change the `name:` from `my_new_project` to `jaffle_shop`.



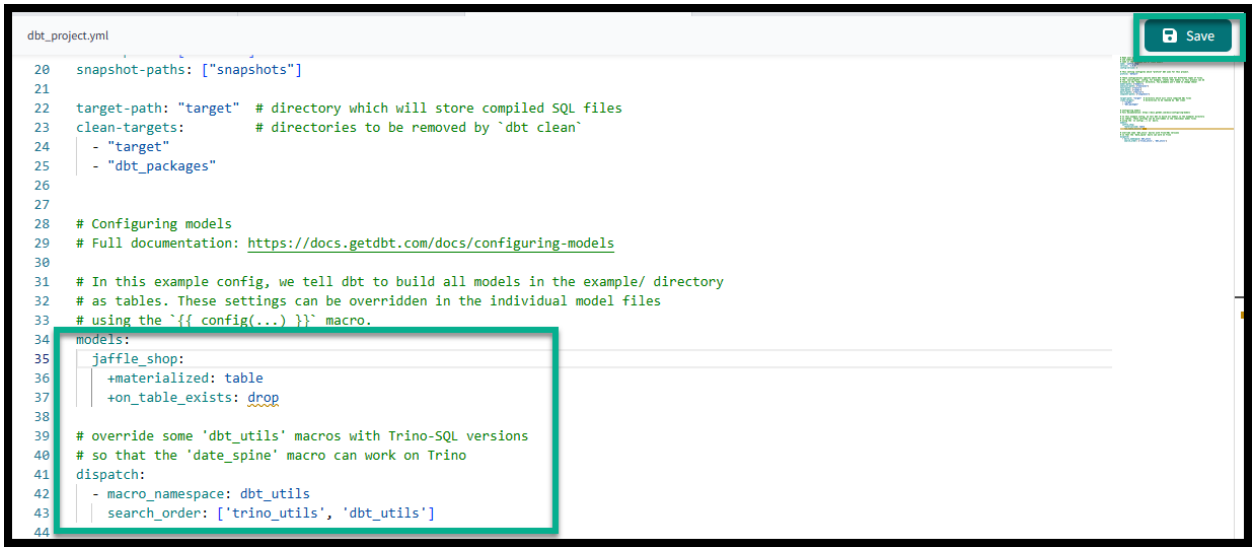
Scroll down and update the `models:` config block to the following.

```
models:
 jaffle_shop:
   +materialized: table
   +on_table_exist: drop
```

Add the following to the end of the file.

```
# override some 'dbt_utils' macros with Trino-SQL versions
# so that the 'date_spine' macro can work on Trino
dispatch:
  - macro_namespace: dbt_utils
    search_order: ['trino_utils', 'dbt_utils']
```


Click **Save**.

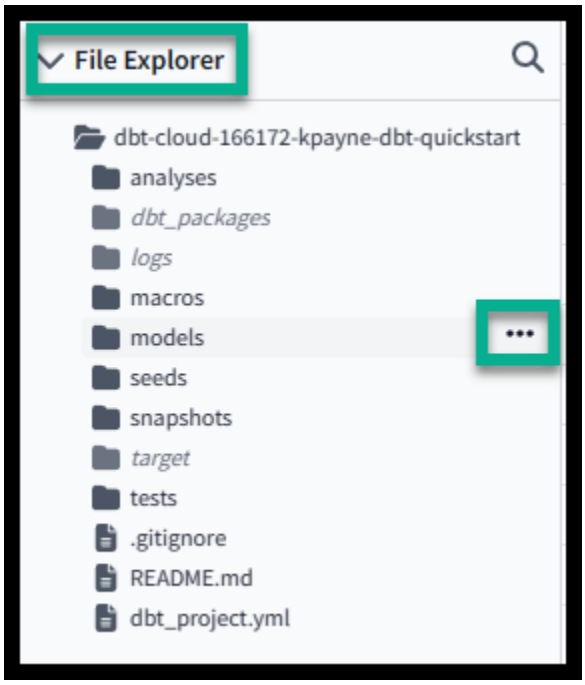


Step 7 - Create dbt model for structure zone's customers table

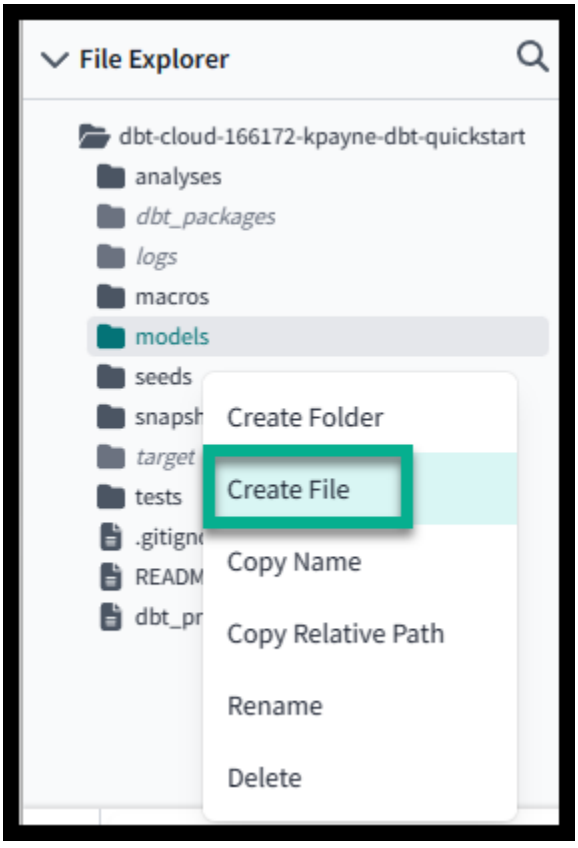
A dbt model is a representation of a table or view in the data model. Our first dbt model will be the transformation of the land zone's customers table to the structure zone.

As mentioned previously, the data quality is high from the land zone. In this example, you will create a simple model that conforms to name & data type standards. You will also be utilizing the Iceberg table format for the new structure zone table.

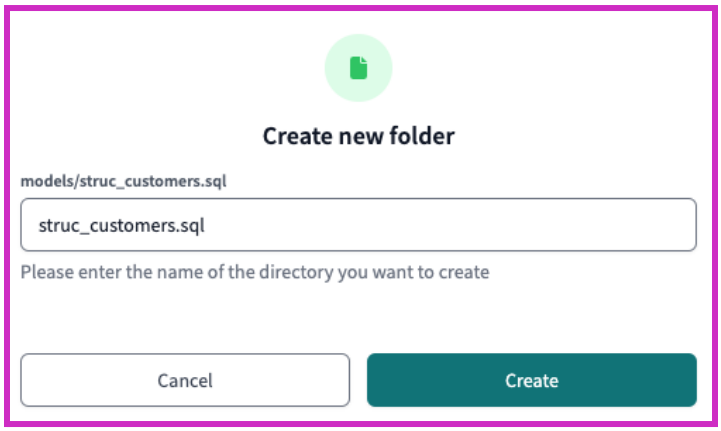
Hover over the **models** directory. An **ellipses** will appear.



Click **Create File**.



Type `struc_customers.sql` in the box for the file name then click **Create**.

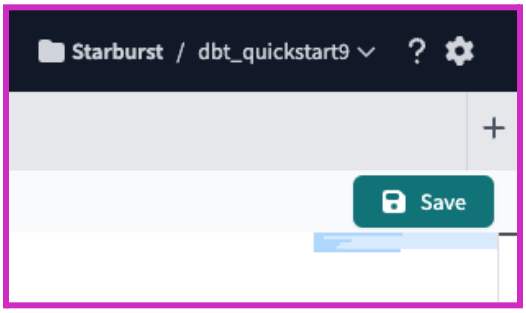


Copy the following SQL and paste it into the IDE.

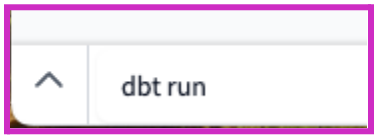
```
-- adapting land zone names & types to struc zone standards
SELECT
    cast(id as integer) AS customer_id,
    first_name,
    last_name
FROM dbt_postgresql.jaffle_shop.jaffle_shop_customers
```

Note: We are assuming that the data quality was validated as not requiring any "clean up" activities. This is mostly due to the simple nature of this example table. If the data contained address information, for example, then we would need to account for more rigorous validation/standardization of the provided address. Additionally, an address would include a postal code allowing us to enrich the `struc_customers` table this model is building with additional lookup information such as an average income based on postal code.

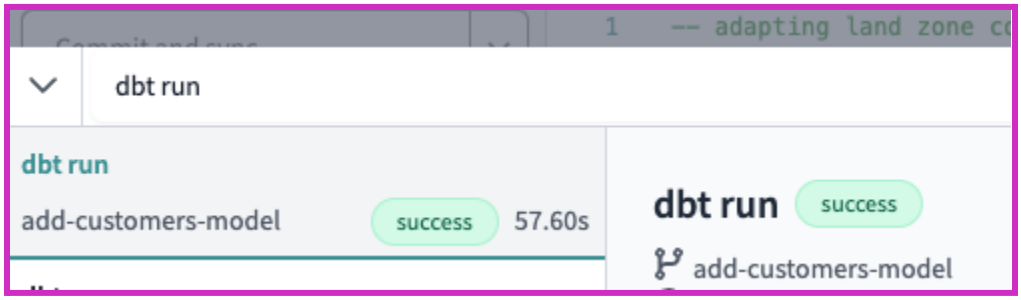
Click **Save** in the upper right.



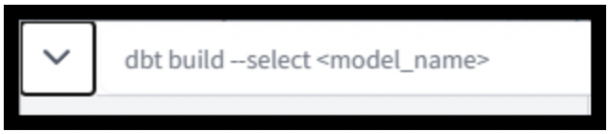
In the command prompt (bottom left) type `dbt run` and press **enter**.



Wait for the `dbt run` to show **success**.

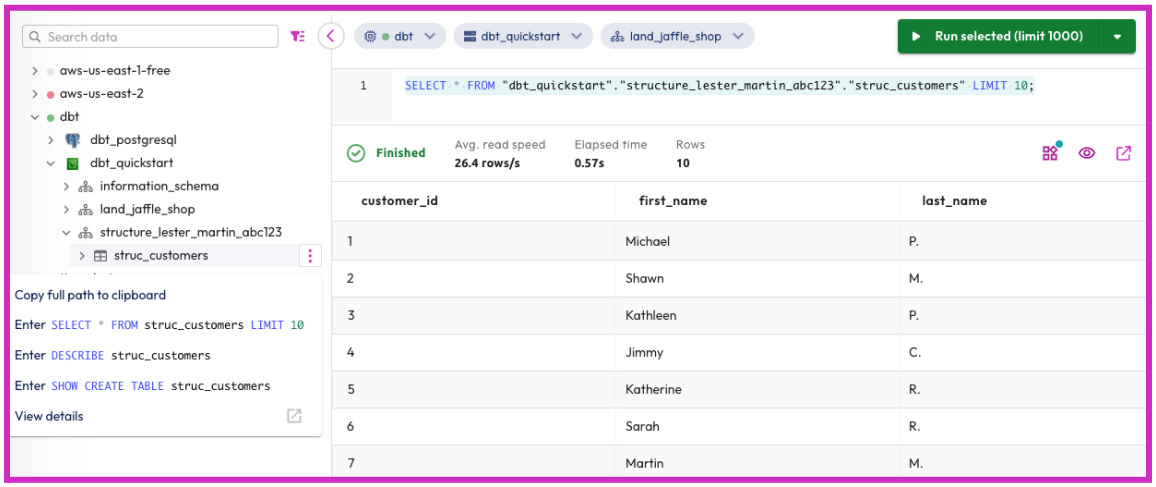


Click the collapse arrow to get back to the main view.

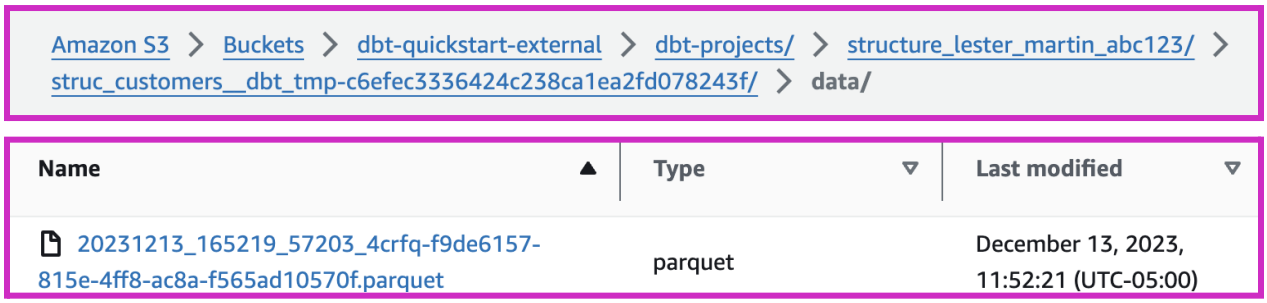


Step 8 - Query the new table in Starburst Galaxy

Return to the Starburst Galaxy UI. Collapse the `dbt_quickstart` catalog and then expand it again to see the new schema dbt created. Expand the new schema dbt created. Hover over the `struc_customers` table and click the **ellipses**. Click on the **Enter SELECT * FROM …** option and then click the **Run selected (limit 1000)** button.



You do not have access to the AWS console for the account where the S3 bucket is located, so the following screenshot is provided for your review. It shows the location of the Iceberg table and that the data was created using the Parquet file format.



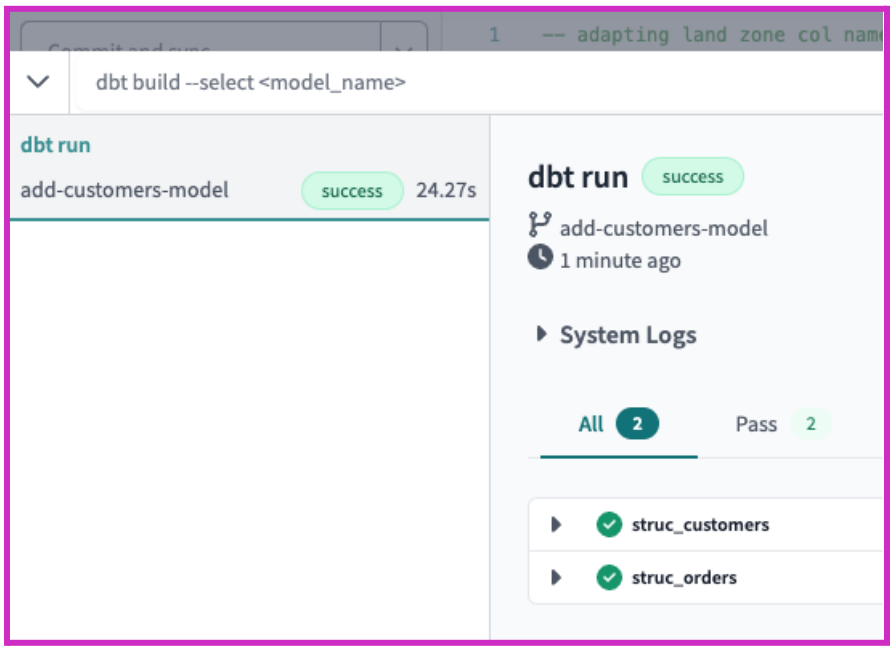
Step 9 - Create dbt model for structure zone's orders table

Like before, hover over the **models** directory. On the **ellipses** that appears, click **Create file**. Type `struc_orders.sql` in the box for the file name then click **Create**.

Copy the following SQL and paste it into the IDE.

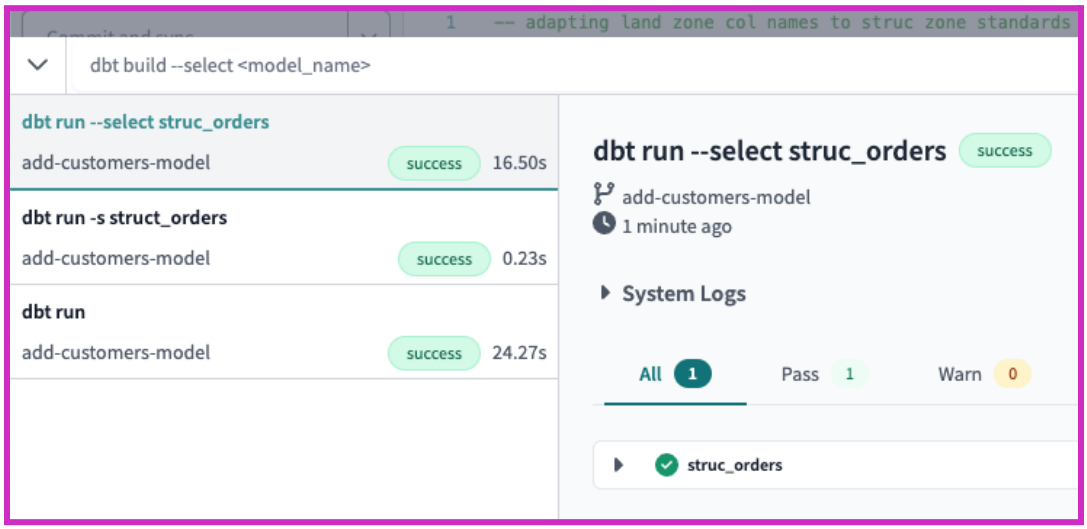
```
-- adapting land zone names & types to struc zone standards
SELECT
    cast(id as integer) AS order_id,
    cast(id as integer) AS customer_id,
    cast(order_date as date) AS order_date,
    status AS order_status
FROM dbt_postgresql.jaffle_shop.jaffle_shop_orders
```


Click **Save** in the upper right. In the command prompt (bottom left) type `dbt run` and press **enter**. Wait for the `dbt run` to show **success**.

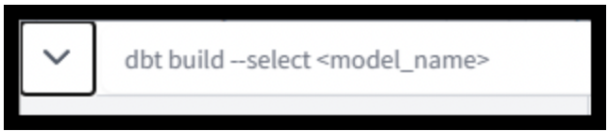


Notice that both models are being executed. To run just the `struc_orders` model, execute the following command.

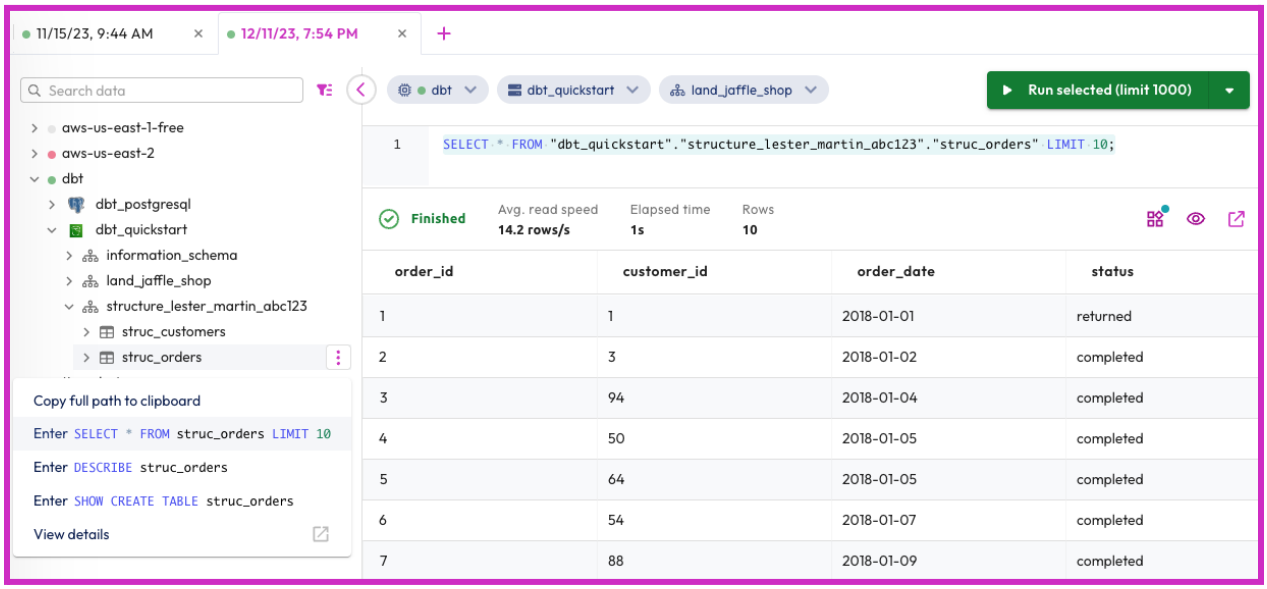
```
dbt run --select struc_orders
```



Click the collapse arrow to get back to the main view.



Return to the Starburst Galaxy UI. Collapse and expand the new schema dbt created. Validate the `struc_orders` table has been created and populated.

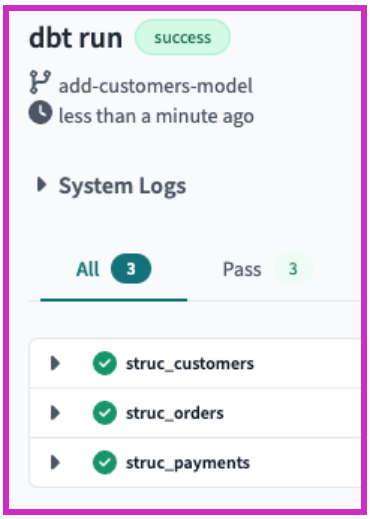


Step 10 - Create dbt model for structure zone's payments table

Create a new model named `struc_payments.sql` and add the following SQL to it.

```
-- adapting land zone names & types to struc zone standards
SELECT
    id AS payment_id,
    orderid AS order_id,
    paymentmethod AS payment_type,
    status AS payment_status,
    amount,
    created AS payment_date
FROM dbt_quickstart.land_jaffle_shop."stripe-payments"
```


Save the model and then execute `dbt run` again in the command prompt. Verify the project executes successfully.



Return to the Starburst Galaxy UI and verify the payments table is present and loaded.

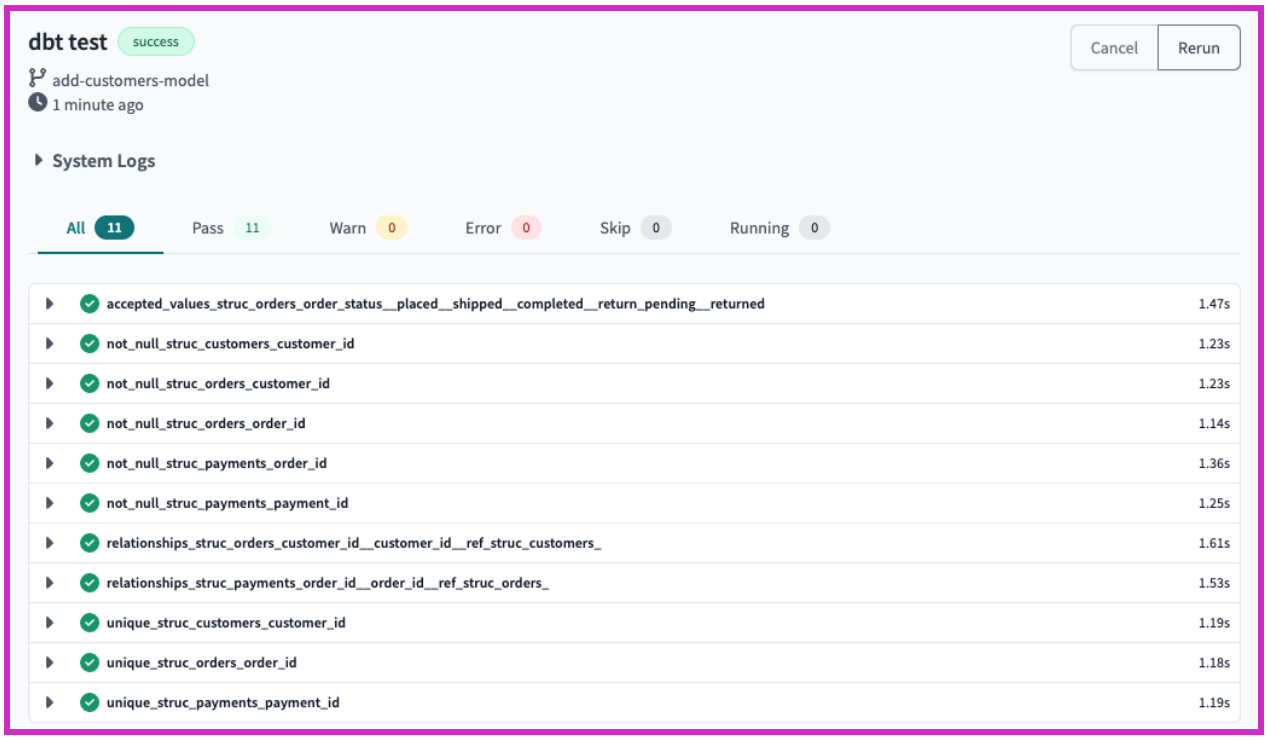
Step 11 - Add tests and documentation to models

Adding tests to a project helps to ensure data quality. Adding documentation to your project allows you to describe your models in rich detail and share that information with your team. Here, you're going to add tests and basic documentation to your project.

This configuration information belongs in a file with a `.yml` extension saved into the **models** directory. Create a new file named `schema.yml` and paste the following code into it.


```
version: 2

models:

 - name: struc_customers
   description: This model cleans up customer data
   columns:
     - name: customer_id
       description: Primary key
       tests:
         - unique
         - not_null

 - name: struc_orders
   description: This model cleans up order data
   columns:
     - name: order_id
       description: Primary key
       tests:
         - unique
         - not_null
     - name: order_status
       tests:
         - accepted_values:
             values: ['placed', 'shipped', 'completed',
                      'return_pending', 'returned']
     - name: customer_id
       tests:
         - not_null
         - relationships:
             to: ref('struc_customers')
             field: customer_id

 - name: struc_payments
   description: This model cleans up payment data
   columns:
     - name: payment_id
       description: Primary key
       tests:
         - unique
         - not_null
     - name: order_id
       tests:
         - not_null
         - relationships:
             to: ref('struc_orders')
             field: order_id
```


After reading through the code above to understand the purpose of the tests, save the file and then execute `dbt test` in the command prompt. This triggers dbt to construct a query for each test. Each of these queries will return the number of records that fail the test. If this number is 0, then the test was successful. Wait for `dbt test` to show **success**.



To run tests on a single model use the `--select` flag followed by the name of the model as shown in the following example.

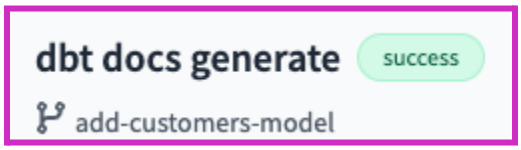
```
dbt test --select struc_customers
```

For additional information, check out the [model selection syntax documentation] for full syntax, and [test selection examples] in particular.

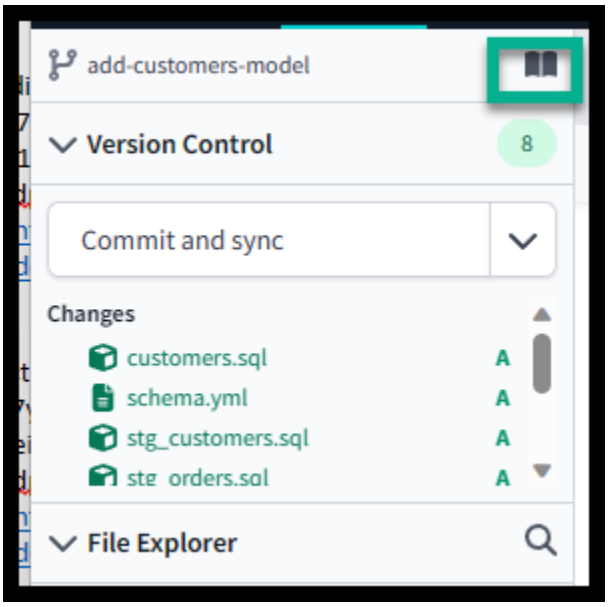
Review the schema.yml file again. The `description:` properties that are included are used for the documentation generation process. Execute the following in the command prompt.

```
dbt docs generate
```

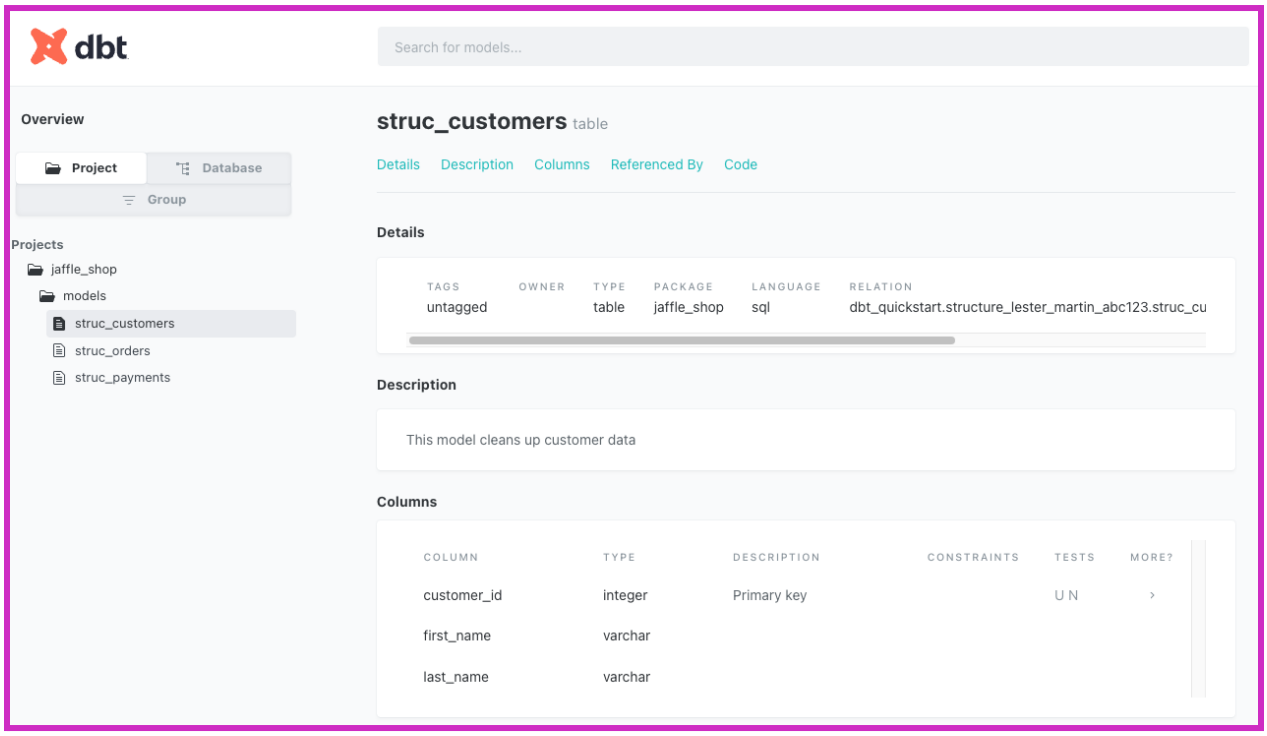

Verify that this ran successfully. There is no need to look further at the output.



Collapse the command prompt arrow to get back to the main view. Click on the **book icon** next to the **add-customer-model** branch name in the top-left.



A new browser tab appears. Expand your project and click on `struc_customers`. Notice the documentation that appears on the right side of the screen.



END OF LAB EXERCISE

Lab 5: Materialize the consume zone with a joined & aggregated dbt Cloud model

Estimated completion time

- 20 minutes

Learning objectives

- After reviewing a query created to solve a reporting requirement, you will implement the SQL code into a dbt model. This model will be the first dataset you create in the structure zone. You will enhance the model to integrate with existing models, convert it to generate a view instead of a table, and add appropriate tests & documentation definitions.

Prerequisites

- Lab 4 - Build the structure zone with dbt Cloud models

Activities

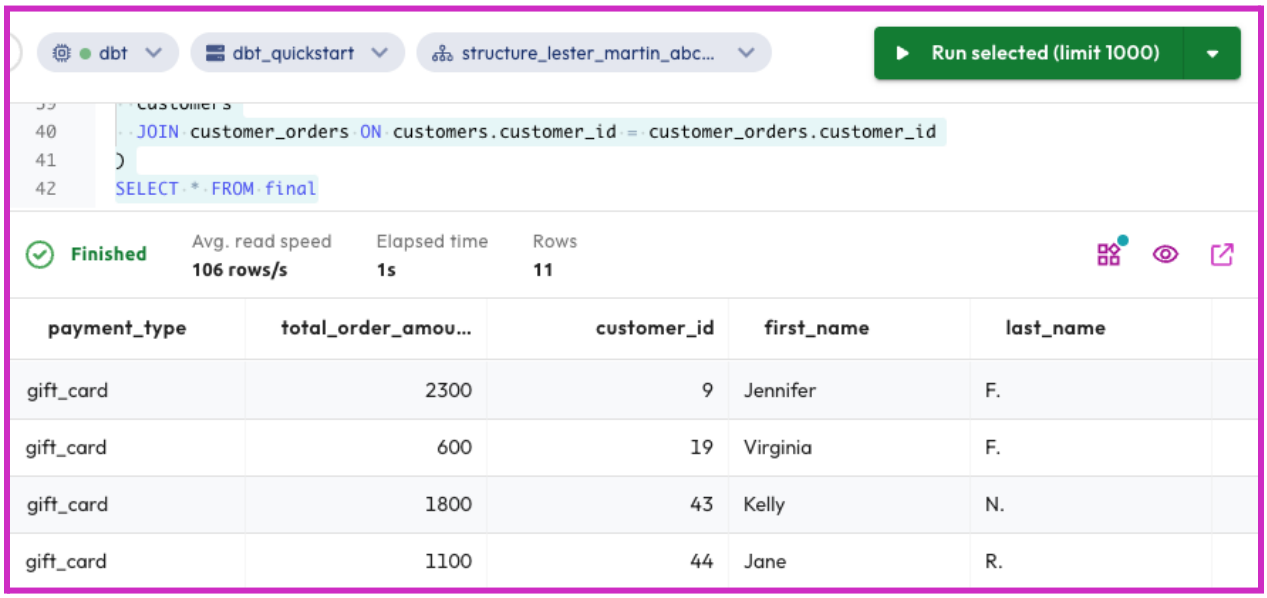
1. Build query for reporting request
2. Create an initial dbt model for the consume zone dataset
3. Build models on top of other models
4. Change structure zone dataset to be a view
5. Add tests and documentation to the consume zone model

Step 1 - Build query for reporting request

You create the following query to solve a business request to report on the total number of orders & total order amount by customer for those customers that have successfully used gift cards for the form of payment.

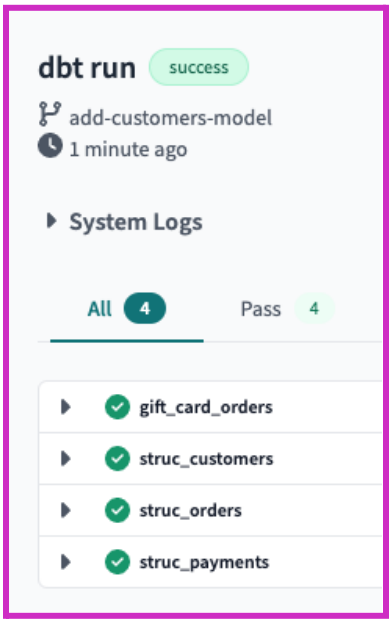
```
WITH customers AS (
SELECT customer_id, first_name, last_name
 FROM struc_customers
),
orders AS (
SELECT order_id, customer_id, order_date, order_status
 FROM struc_orders
),
payments AS (
SELECT payment_id, order_id, payment_type, amount
 FROM struc_payments
WHERE payment_type = 'gift_card'
  AND payment_status = 'success'
),
customer_orders AS (
SELECT
 p.payment_type,
 o.customer_id,
 MIN(o.order_date) AS first_order_date,
 MAX(o.order_date) AS most_recent_order_date,
 COUNT(o.order_id) AS number_of_orders,
 SUM(p.amount) AS total_order_amount
FROM
 orders o
 JOIN payments p ON o.order_id = p.order_id
GROUP BY 1, 2
),
final AS (
SELECT
 customer_orders.payment_type,
 customer_orders.total_order_amount,
 customers.customer_id,
 customers.first_name,
 customers.last_name,
 customer_orders.first_order_date,
 customer_orders.most_recent_order_date,
 COALESCE(customer_orders.number_of_orders, 0) AS number_of_orders
FROM
 customers
 JOIN customer_orders
   ON customers.customer_id = customer_orders.customer_id
)
SELECT * FROM final
```


Verify the query solves the report problem executing it in the Starburst Galaxy UI. Don't forget to select the appropriate pull-down options for the `dbt_quickstart` catalog and the schema dbt created before you run the query.

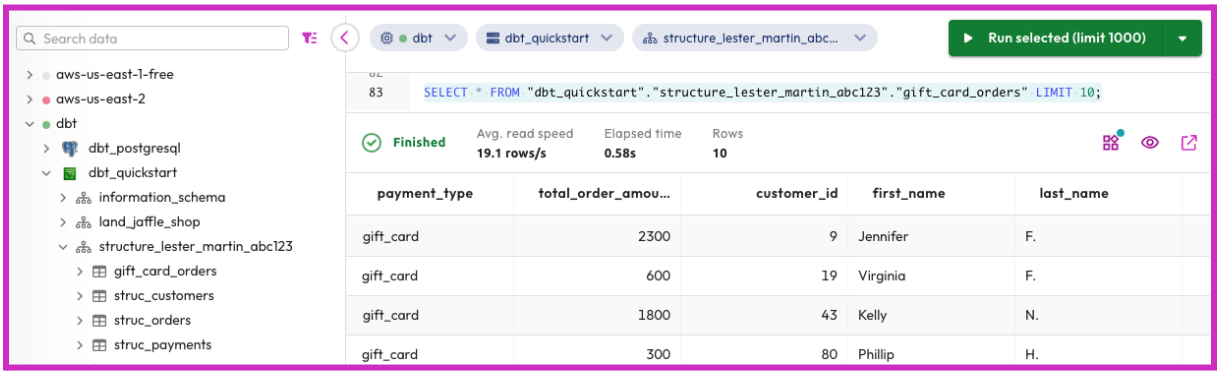


Step 2 - Create an initial dbt model for the consume zone dataset

Return to dbt and create a new model named `gift_card_orders.sql` and paste the query from the prior step into it. Verify the `dbt run` command still functions successfully and now includes a fourth dataset.

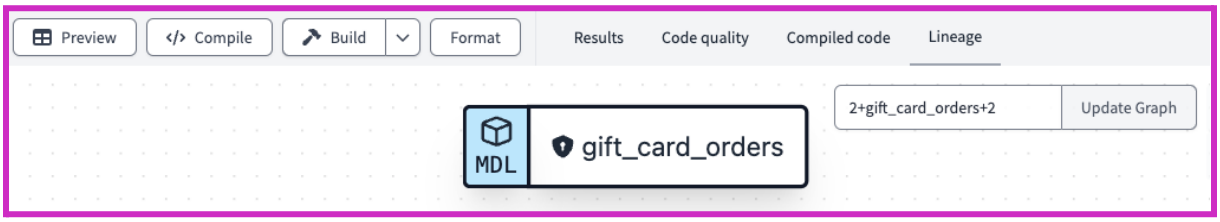


Use Starburst Galaxy UI to verify the dataset is created and populated.



Step 3 - Build models on top of other models

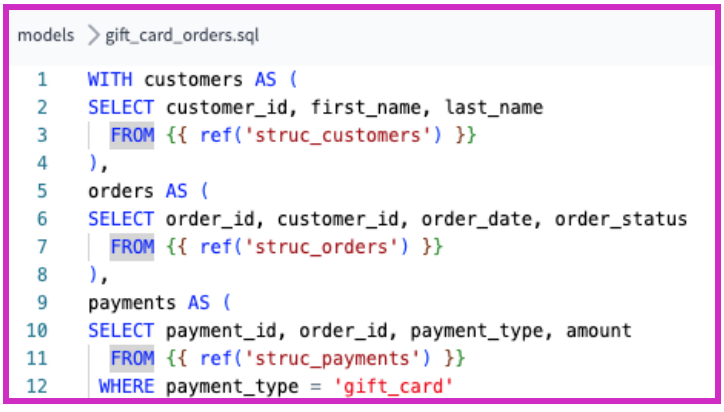
Return to the dbt code editor with the `gift_card_orders.sql` model open. Select **Lineage** in the bottom pane to see this graphical view of the structure zone dataset.



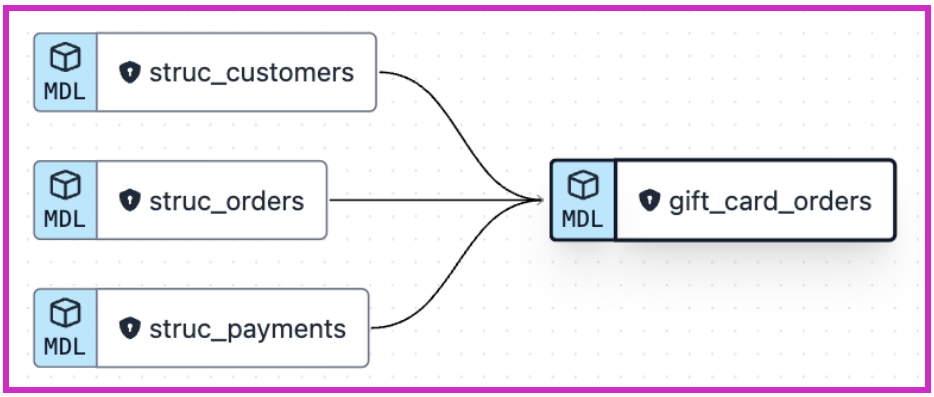
This data lineage visualization does not indicate that it is derived from the 3 `struc_*.sql` models previously created.

To remedy this, templatize the model to wrap the table name of these to look like the following. Replace *`modelname`* with the appropriate model names, sans their `.sql` extensions. See lines 3, 7, and 11 below for how this should now appear in your editor.

```
{{ ref('modelname') }}
```



After saving the file notice the **Lineage** output shows how this structure zone model is built on top of the prior structure zone models.



Run `dbt run` again to ensure all still function correctly. From Starburst Galaxy, verify the models are successfully built and populated.

Step 4 - Change structure zone dataset to be a view

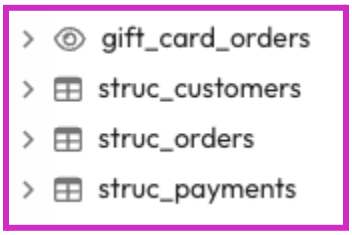
After remembering when starting this lab that the query being used within `gift_card_orders.sql` was not a performance issue, you decide to change how the model is materialized. It makes more sense to use a classic view in this case which eliminates the possibility of staleness by always retrieving the underlying structure zone tables when building the results.

Note: dbt ships with four materializations; `view`, `table`, `incremental`, and `ephemeral`. Visit the documentation on materializations for more information on each of these options.

Add the following to the top of `gift_card_orders.sql` and save it.

```
{{
  config(
    materialized='view'
  )
}}
```


Verify `dbt run` executes successfully in dbt. Then switch to Starburst Galaxy to see the icon for `gift_card_orders` has changed to now indicated a view instead of a table.



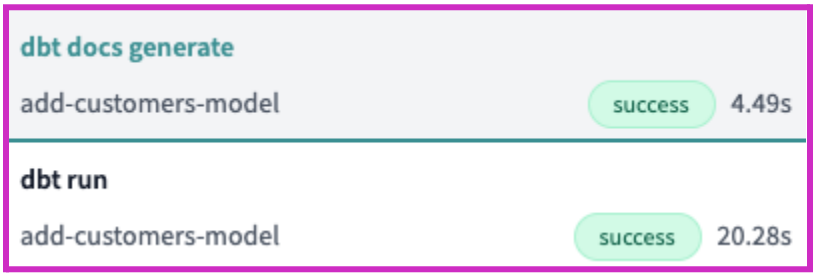
Run a query on the view to verify it functions identically as before.

Step 5 - Add tests and documentation to the consume zone model

Add the following to the top of the `schema.yml` file below the `models:` line.

```
- name: gift_card_orders
  description: >
    Total number of orders & total order amount
    by customer for those customers that have
    successfully used gift cards for the form
    of payment
  columns:
    - name: customer_id
      tests:
        - unique
        - not_null
```

Verify `dbt test, dbt docs generate`, and `dbt run` execute successfully.



END OF LAB EXERCISE

Lab 6: Define a Starburst Galaxy data product

Estimated completion time

- 30 minutes

Learning objectives

- This lab will.

Prerequisites

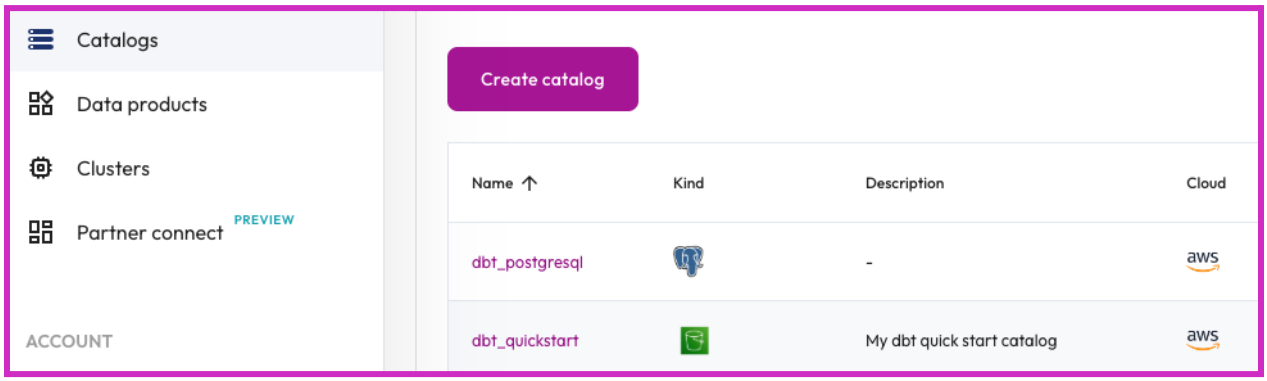
- Lab 5 - Materialize the consume zone with dbt Cloud models

Activities

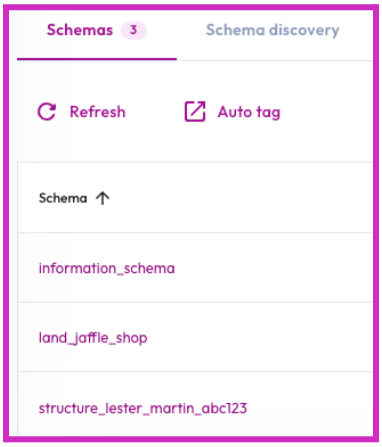
1. Verify lineage of the consume zone dataset
2. Promote schema to a data product
3. View the data product's metadata
4. Create & view metadata for the data product's datasets
5. Query the data product's datasets

Step 1 - Verify lineage of the consume zone dataset

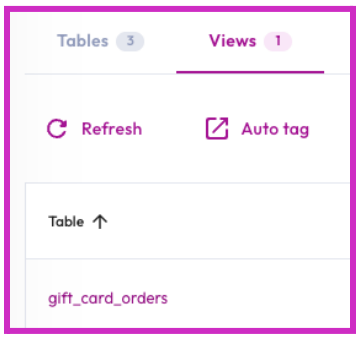
Click on **Catalogs** in the left-side menu. In the list of catalogs below the **Create catalog** button, click on `dbt_quickstart`.



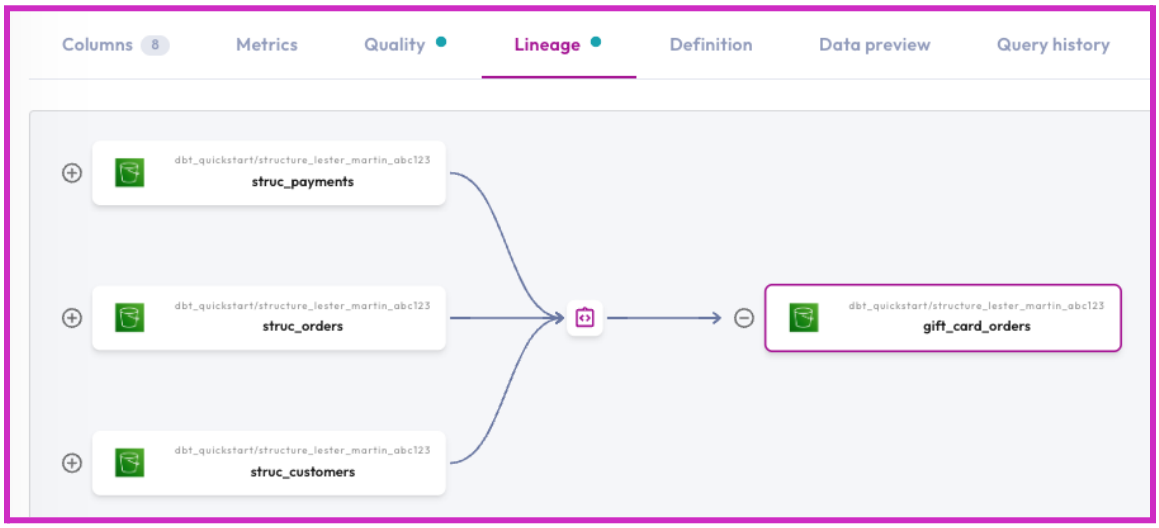
Under **Schemas**, click on the `structure_*` schema that dbt created.



Click on **Views** and then `gift_card_orders`.



Click on **Lineage** and then verify the graphical lineage previously reviewed in dbt for the `gift_card_orders.sql` model is also present in Starburst Galaxy.

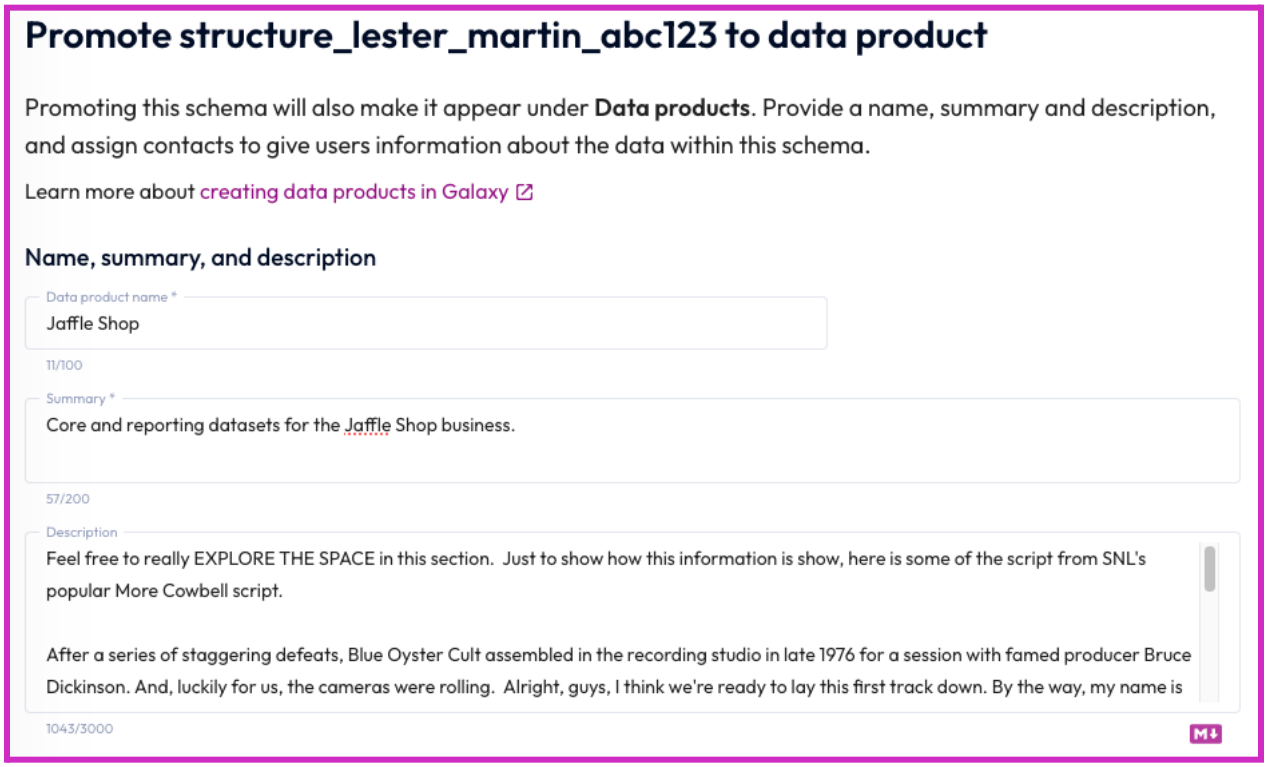


Step 2 - Promote schema to a data product

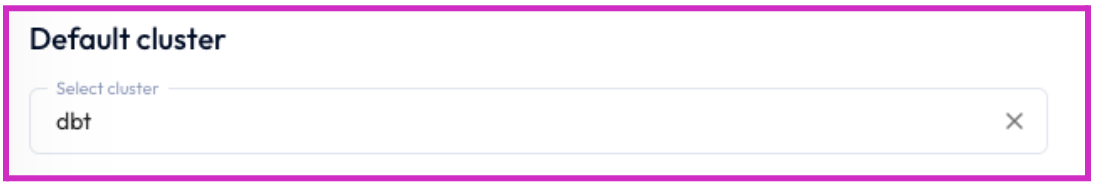
Click on the `structure_*` schema name in the breadcrumbs presented at the top of the page.

Click the **Promote to data product** button in the upper right of the screen.

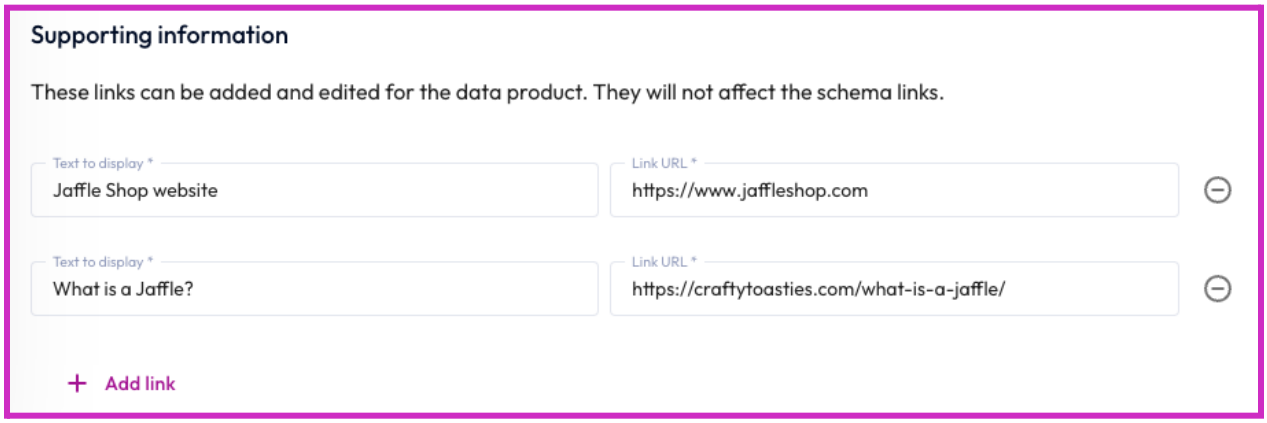
Enter `Jaffle Shop` for the **Data product name** and something for **Summary**. Optionally, include something for **Description**.



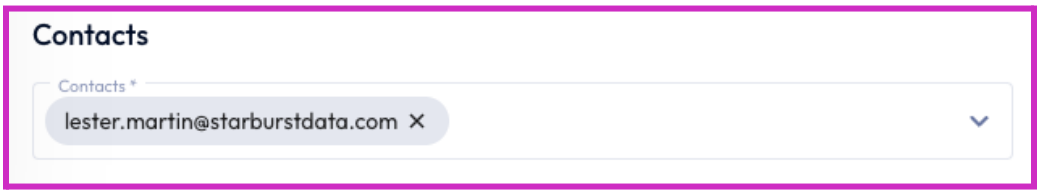
Choose `dbt` in the pulldown below **Default cluster**.



Supply a set of **Text to display** and **Link URL** values, or click the **minus-sign icon** to the far right to remove any helpful links for this data product. Optionally, click on **Add link** to create one, or more, additional links.



Select your username in the **Contacts** pulldown.



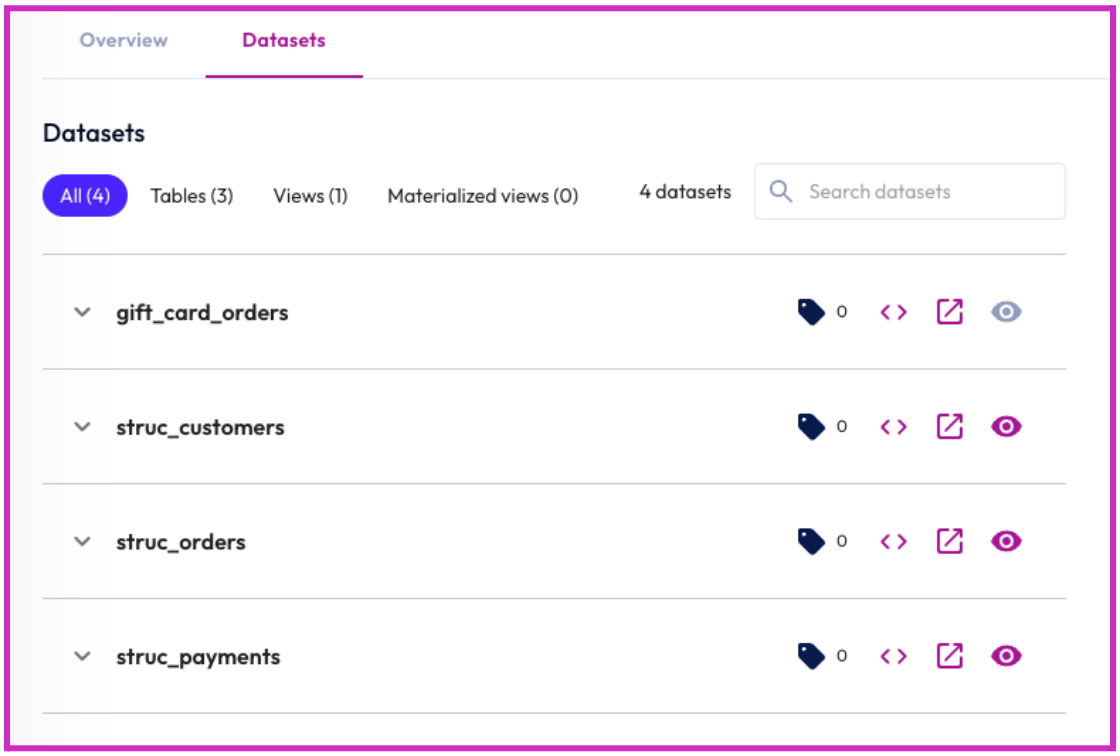
Click on **Promote to Data Product** at the bottom right of the screen.

Verify your `Jaffle Shop` data product tile is present on the **Data products** landing page that you were routed to. It is also accessible from the **Data products** link in the left navigation menu.

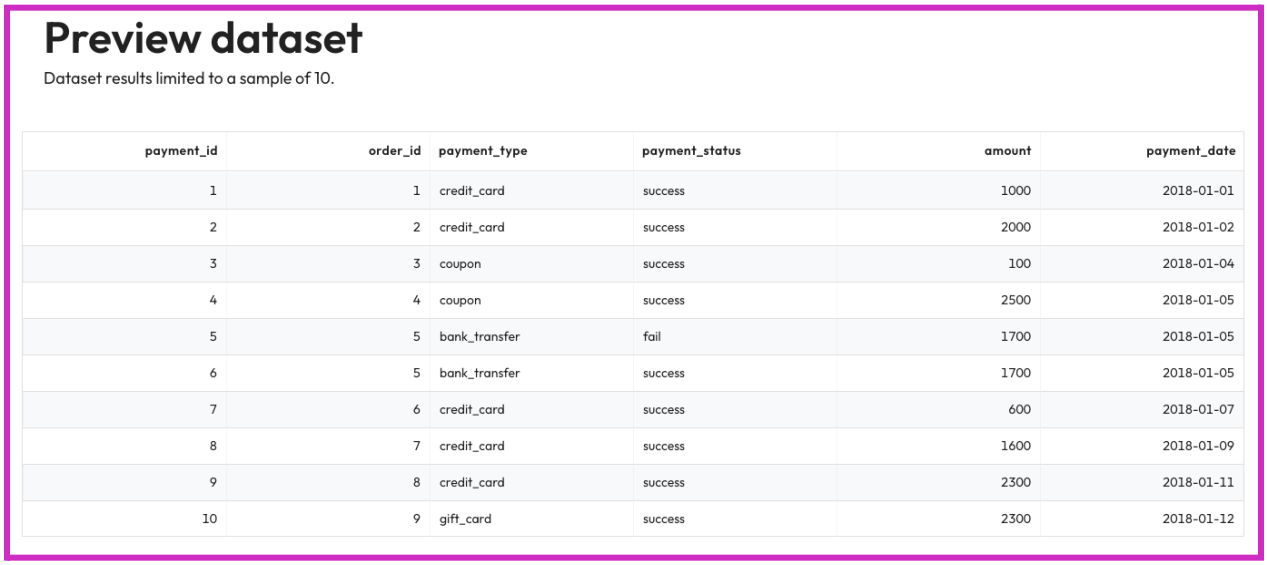
Step 3 - View the data product's metadata

Click in the middle of the data product tile presented in the last screenshot. Review the overarching information and metadata presented on this details page.

Click on the **Datasets** tab and you will see the 4 datasets in this data product which are comprised of 3 tables and 1 view. You will also see a list of them.



Click on the **eye icon** on the far right of the `struc_payments` row at the bottom of the list to see the **Preview dataset** pop-up.



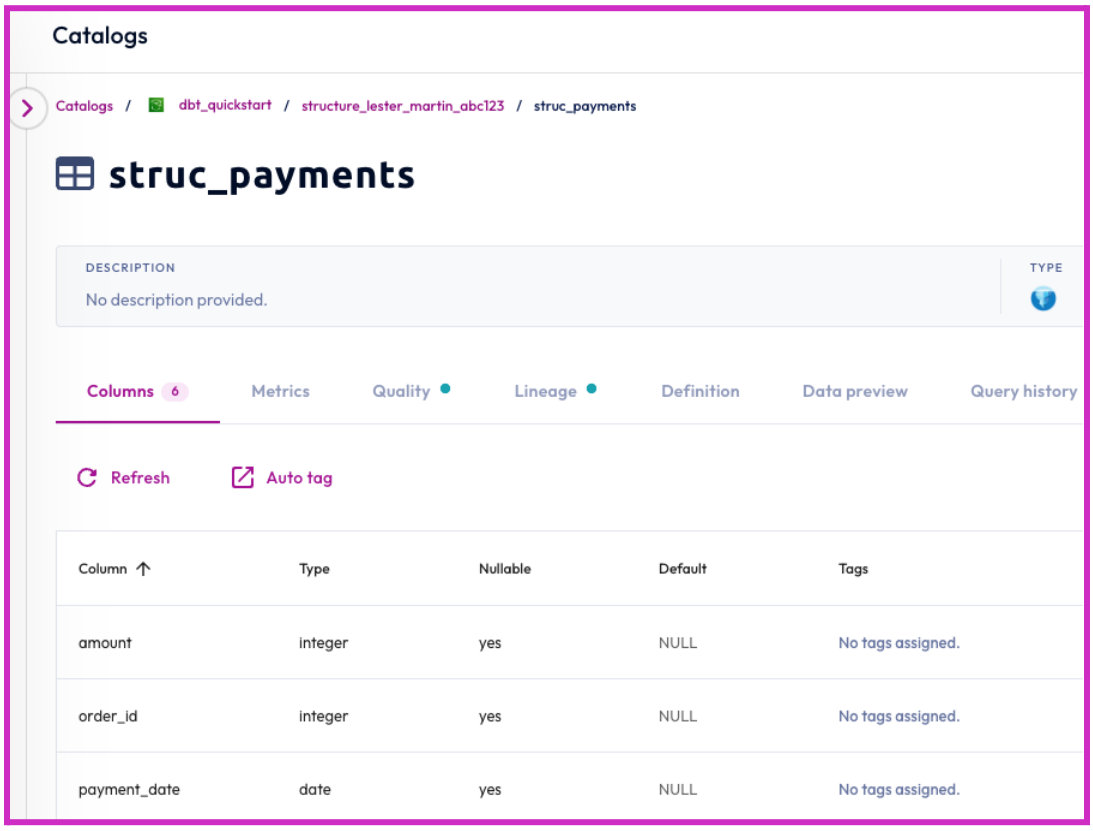
Preview dataset

Dataset results limited to a sample of 10.

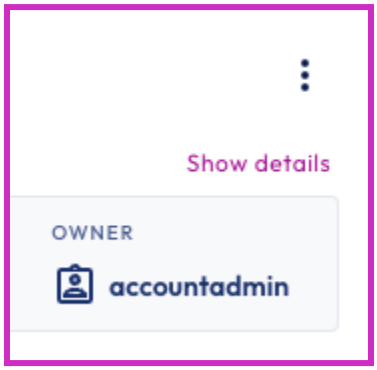
payment_id	order_id	payment_type	payment_status	amount	payment_date
1	1	credit_card	success	1000	2018-01-01
2	2	credit_card	success	2000	2018-01-02
3	3	coupon	success	100	2018-01-04
4	4	coupon	success	2500	2018-01-05
5	5	bank_transfer	fail	1700	2018-01-05
6	5	bank_transfer	success	1700	2018-01-05
7	6	credit_card	success	600	2018-01-07
8	7	credit_card	success	1600	2018-01-09
9	8	credit_card	success	2300	2018-01-11
10	9	gift_card	success	2300	2018-01-12

Step 4 - Create & view metadata for the data product's datasets

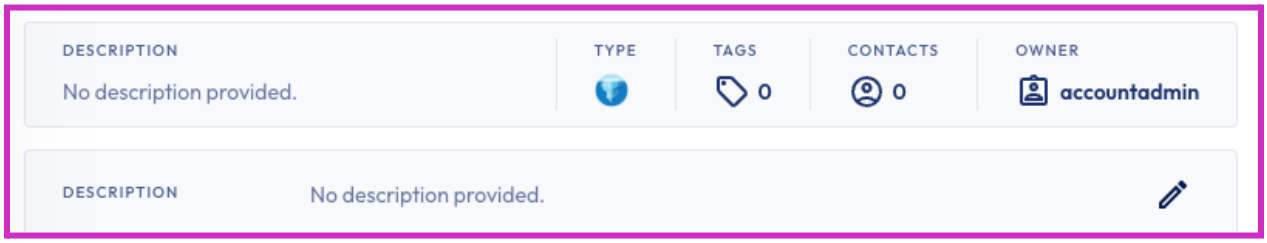
After closing the pop-up, click on the icon on the same dataset which will take you to the **Catalogs** UI and automatically drill down the **Columns** list for the `struc_payments` table.



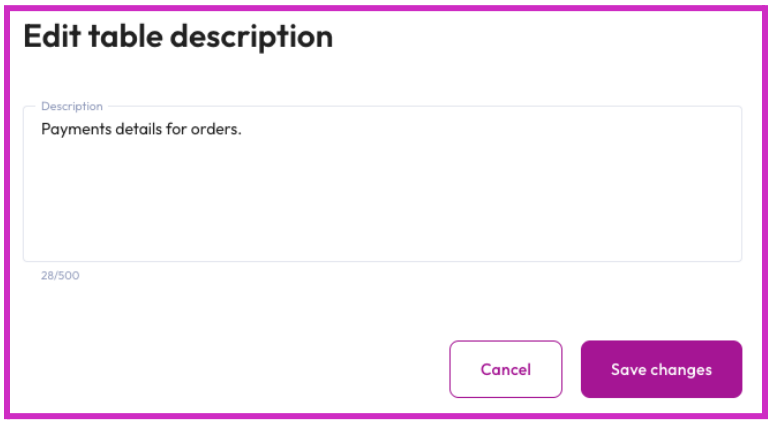
Click on **Show details** in the upper right just below the vertical ellipses.



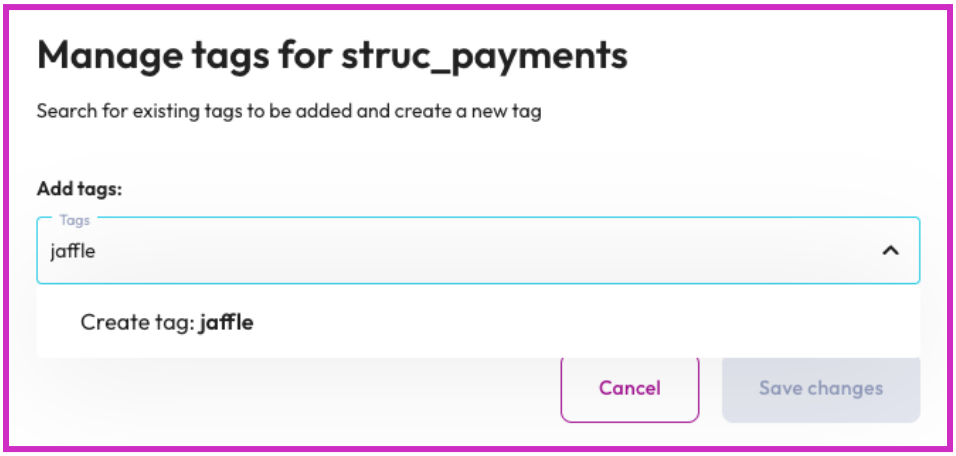
Click the **pencil icon** on the far right of the **DESCRIPTION** field.



In the **Edit table description** pop-up that surfaces, add a meaningful **Description** before you **Save changes**.

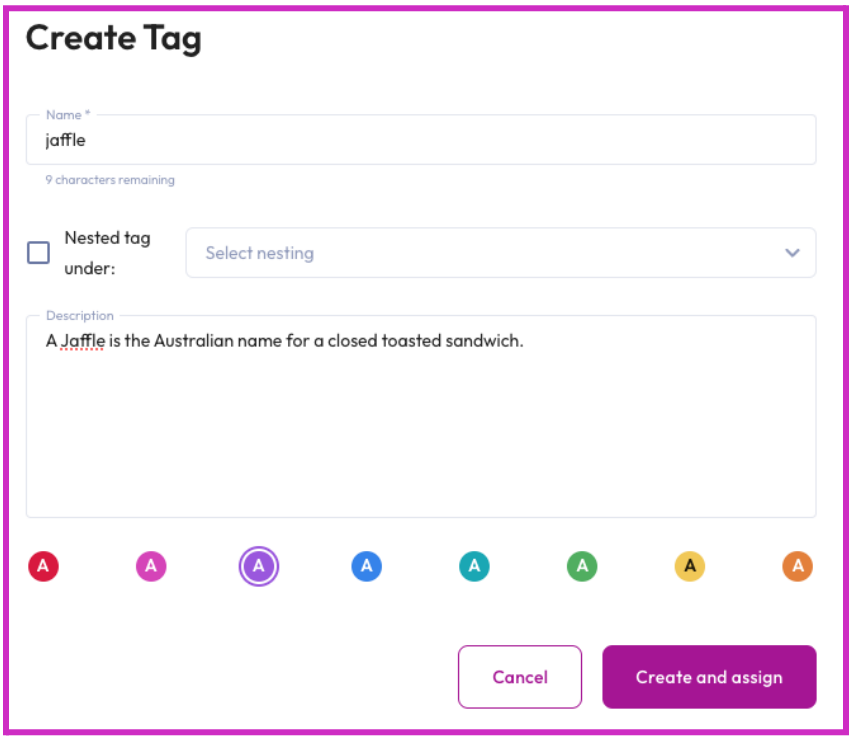


When back on the `struc_payments` **Catalogs** page, click on the **pencil icon** to the far right of **TAGS** to allow the **Manage tags for struct_payments** pop-up to surface. In the **Add tags** field, type in `jaffle`.

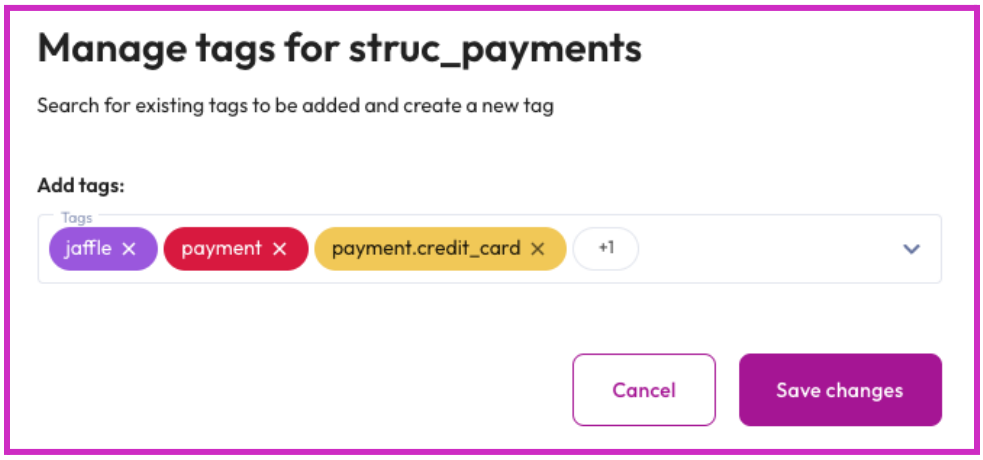


Click on **Create tag: jaffle** that surfaces just below the **Tags** field you typed in. Since this is a new tag you will be routed to the **Create Tag** pop-up *(which is also available from **Access***

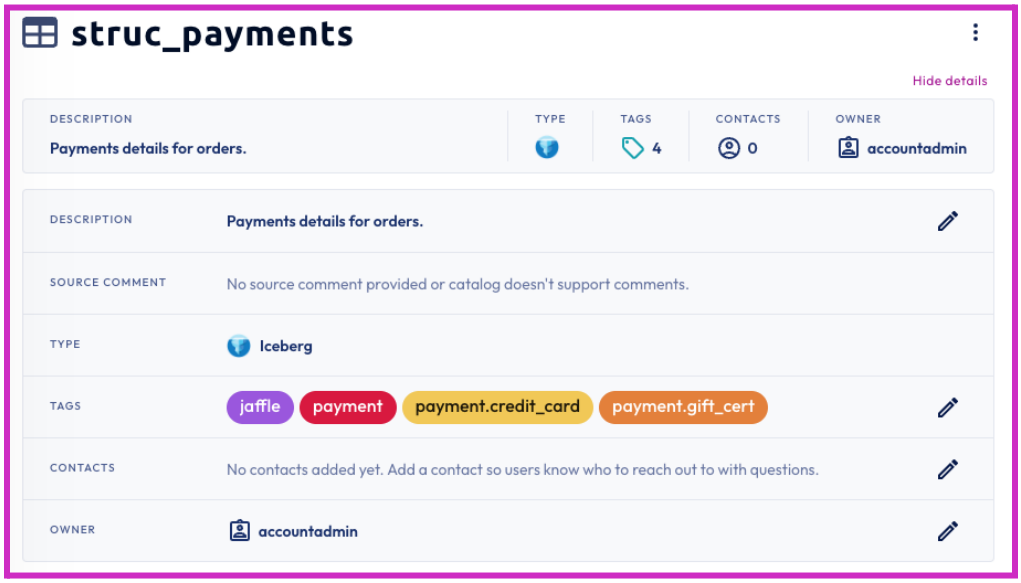
control > ***Tags*** > ***Create tag)*** where you can modify the **Name**, add a **Description**, and/or change the color. Close this window by clicking on **Create and assign**.



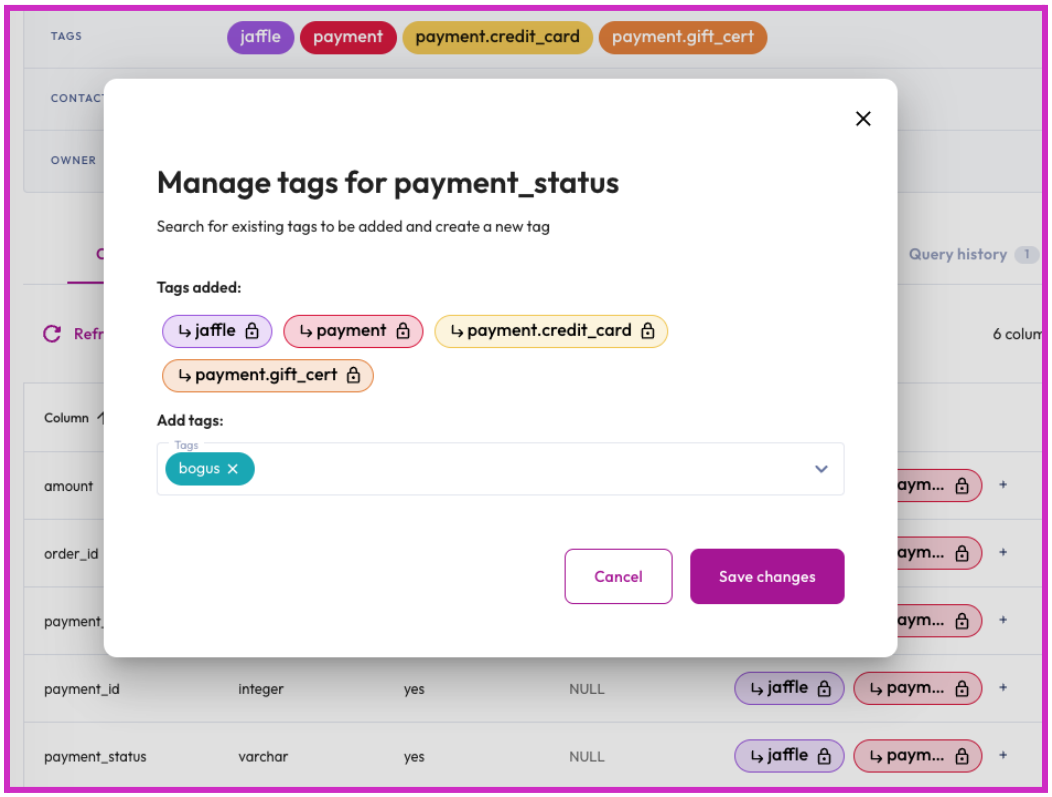
You are returned to **Manage tags for struc_payments** where you optionally can add and/or create additional tags before you click on **Save changes**.



The schema has much more metadata available now.



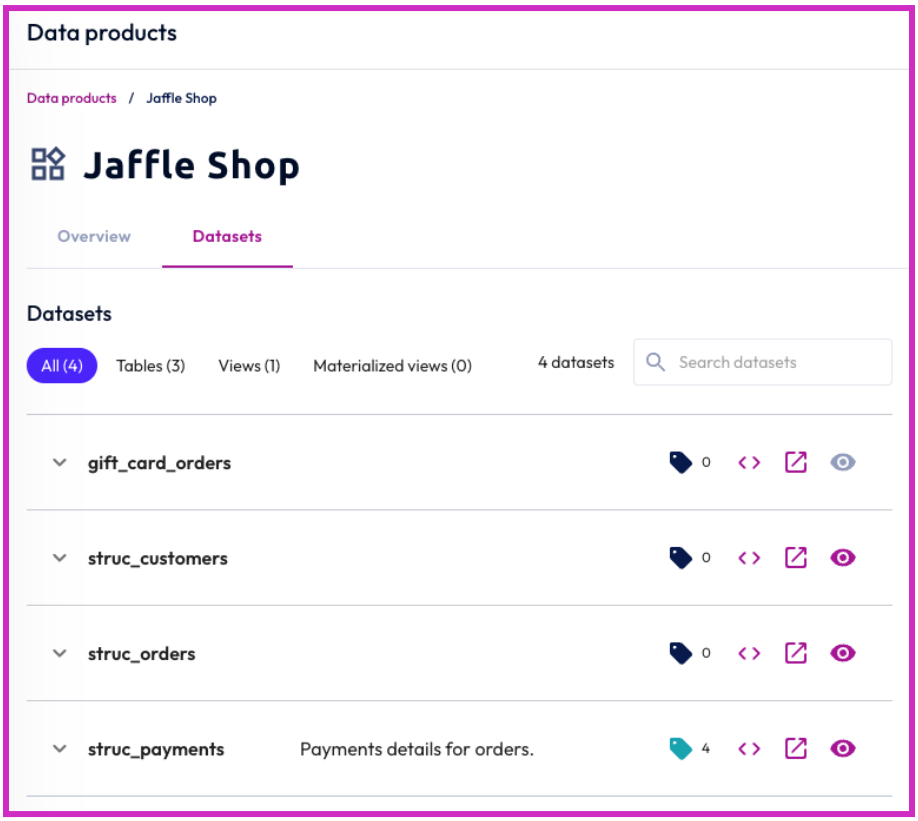
Below that top-level info, in the **Columns** tab, you can see the columns are inheriting the table's tags. Click on the **+** icon under the **Tags** column header allows you to add/create column-level tags once you **Save changes**.



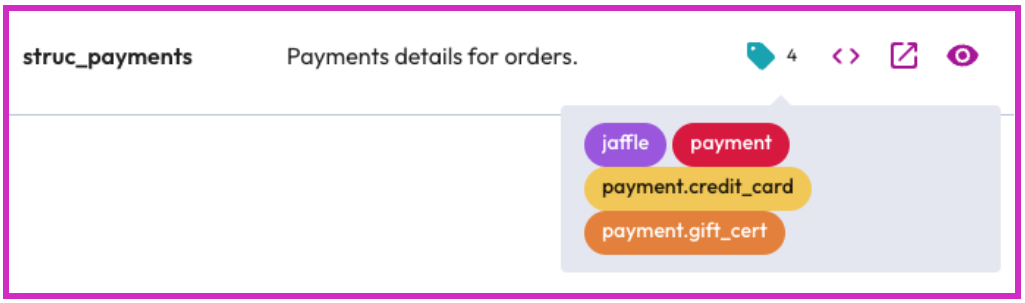
The column-level tags will be listed before the inherited ones.

On the same `payment_status` **Column** row noticed the **No description provided** message under the **Description** column. Click on the **pencil icon** to add a useful message and then click the **disk icon** to save it.

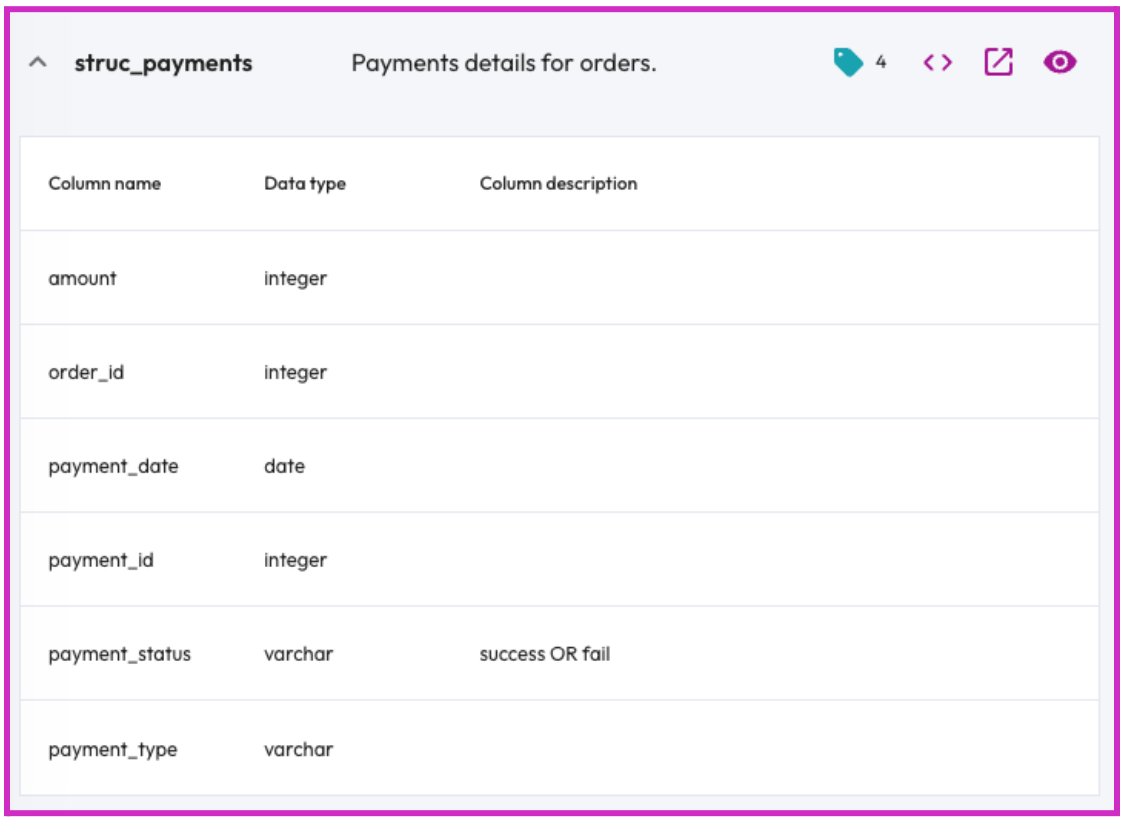
Click on **Data products** on the left navigation again, click on the `Jaffle Shop` tile, and then select the **Datasets** tab.



Notice that `struct_payments` is reporting that it has 4 tags. *Remember, your number may be different.* Hover over the **tag icon** to see a preview of what is assigned.



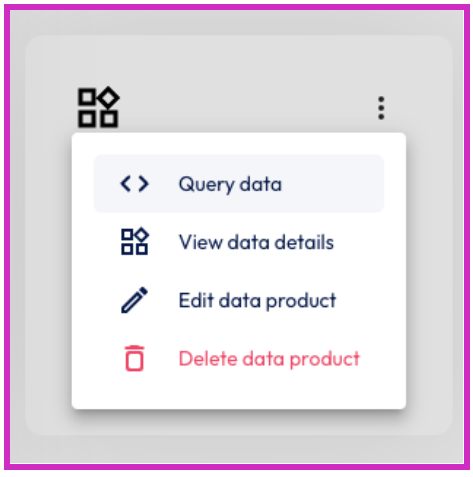
Toggling the **down arrow** to the left of the `struct_payments` item in the list will open up the column list so that in addition to the description you provided for the table, you can see the **Column description** that was added.



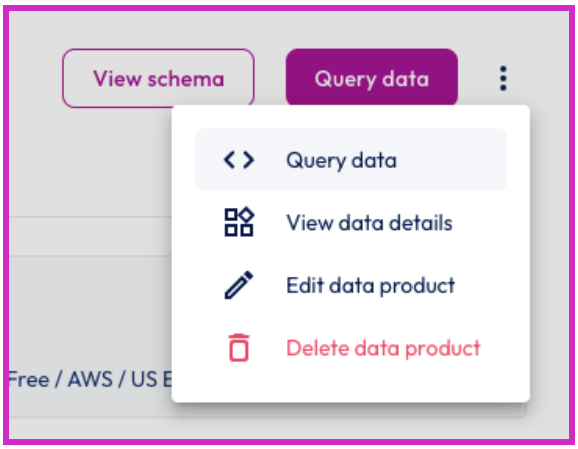
Click on the **< > icon** next to the others we have just explored to open the **Query editor**.

Step 5 - Query the data product's datasets

The end of the last step opened up the **Query editor** making it easy to query the datasets, but there are a few other ways to get there. On the **Data products** UI you can click on the **vertical ellipses** in the upper-right of the tile nd then select **Query data** to get the same behavior.

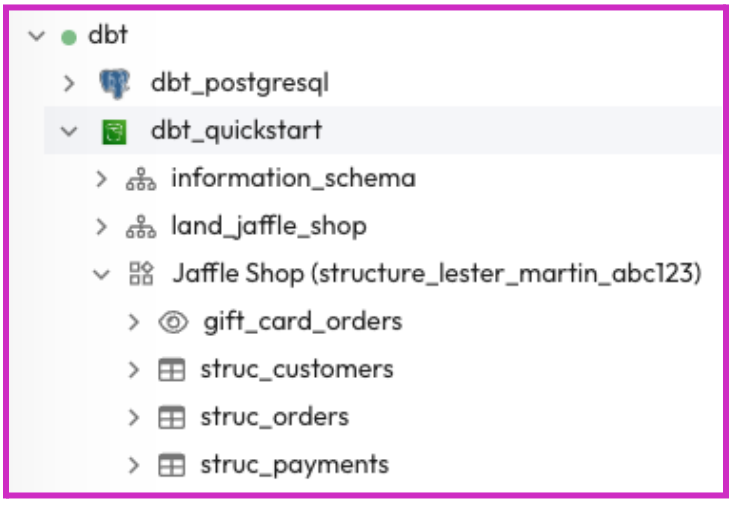


Clicking on the center of the tile instead shows a **Query data** button on the specific data product page. There is also a **Query data** submenu item of the **vertical ellipses** to the right of the button. Both of these options yield the same results – routing you to the **Query editor**.

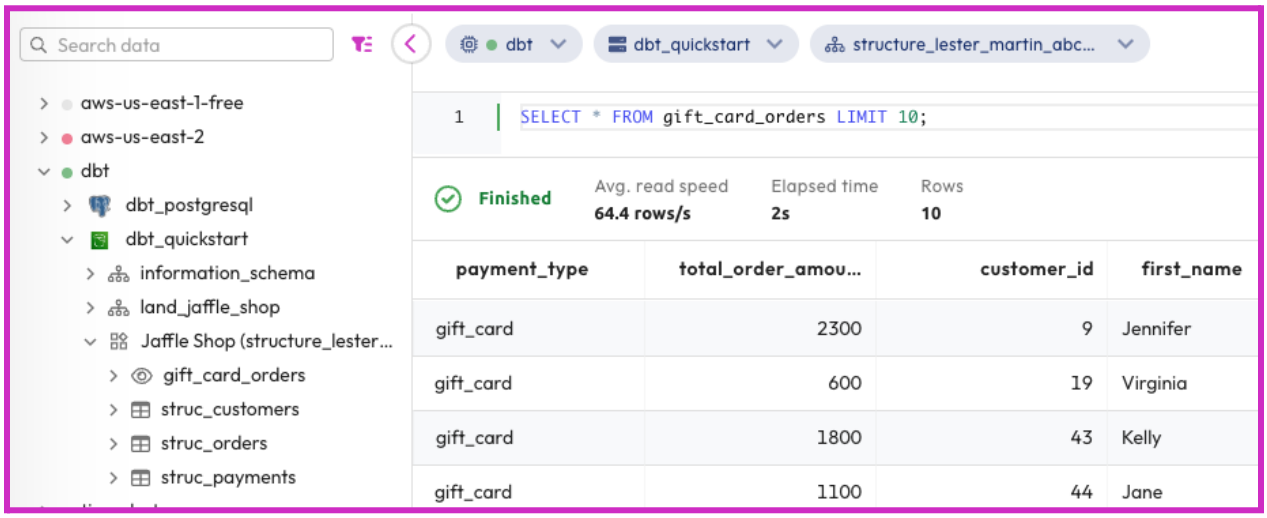


The data products UI provides multiple ways to document, highlight, search, explore, and share curated datasets, but at the end of the day, a Starburst Galaxy data product is aligned to a particular schema. That means, that any way you can query data in that schema is an appropriate way to query the data.

Notice how the icon has changed for the structure zone's schema below and that the schema name is wrapped by the data product name.



Query the consume zone's `gift_card_orders` view.



END OF LAB EXERCISE

Lab 7: Commit changes and create a production environment

Estimated completion time

- 15 minutes

Learning objectives

- In this dbt Cloud lab you will commit your project changes so that the repository will have your latest version of the code. You will also create a production environment and schedule a job.

Prerequisites

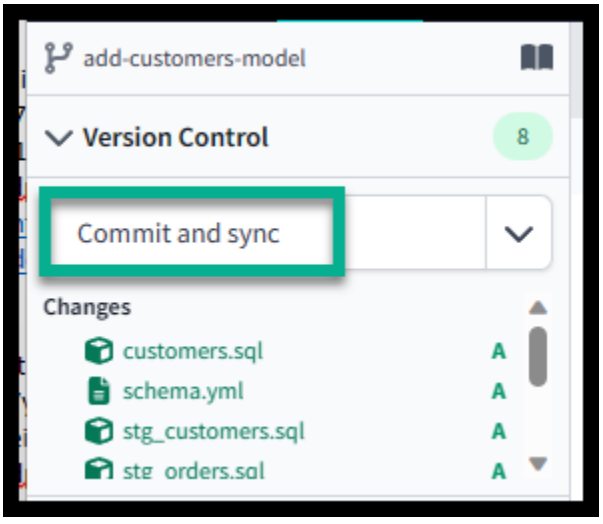
- Lab 5 - Materialize the consume zone with dbt Cloud models

Activities

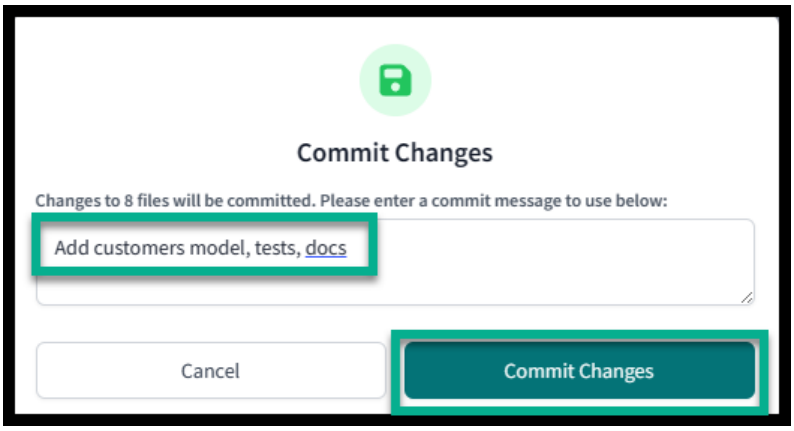
1. Commit your changes
2. Create a production environment
3. Create a job
4. Run a job

Step 1 - Commit your changes

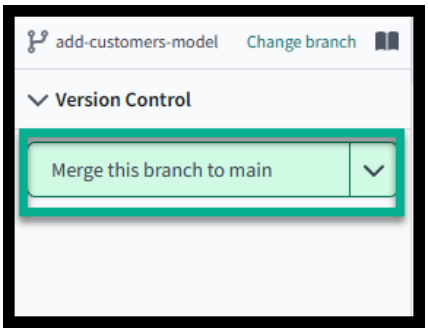
In the dbt Cloud IDE click on **Commit and sync** under **Version control** in the upper-right corner.



Provide a meaningful commit message and click **Commit Changes**.



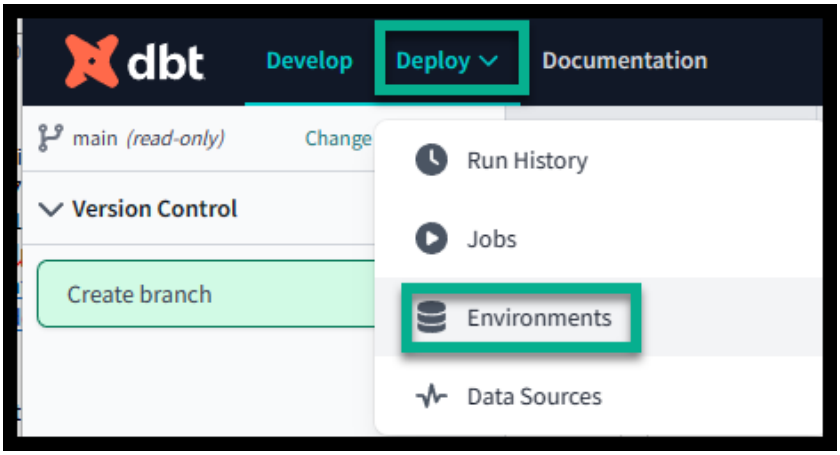
Click **Merge this branch to main** which will add the changes to the main branch of your repo.



Step 2 - Create a production environment

The environment that you first created is a development environment. It is a best practice to have a production environment where you can run your code on a schedule as a job.

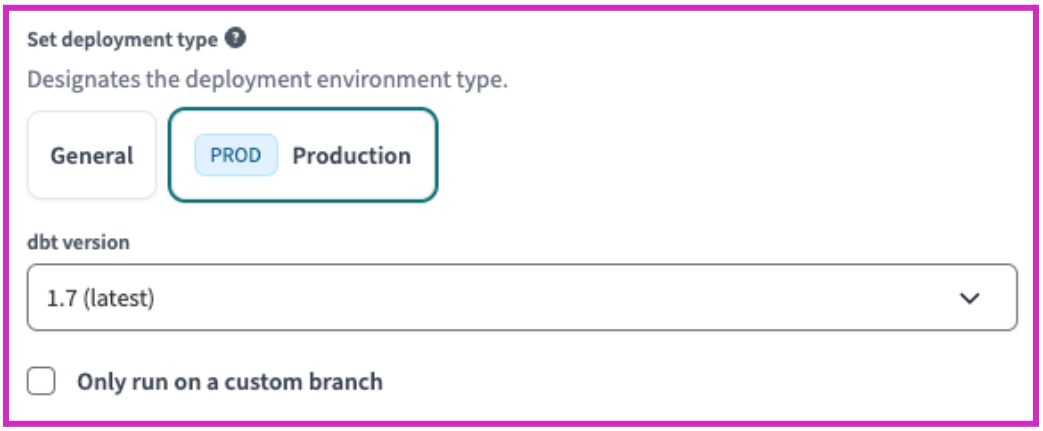
Click **Deploy** to begin, and then click **Environments**.



Click **+ Create environment**.

In the box under **Name**, type `Production`.

Keep **Production** selected in the **Set deployment type** section.



Note: You need your Connection information from your Starburst Galaxy account again to continue.

Use these steps to fill out the **Deployment credentials** section.

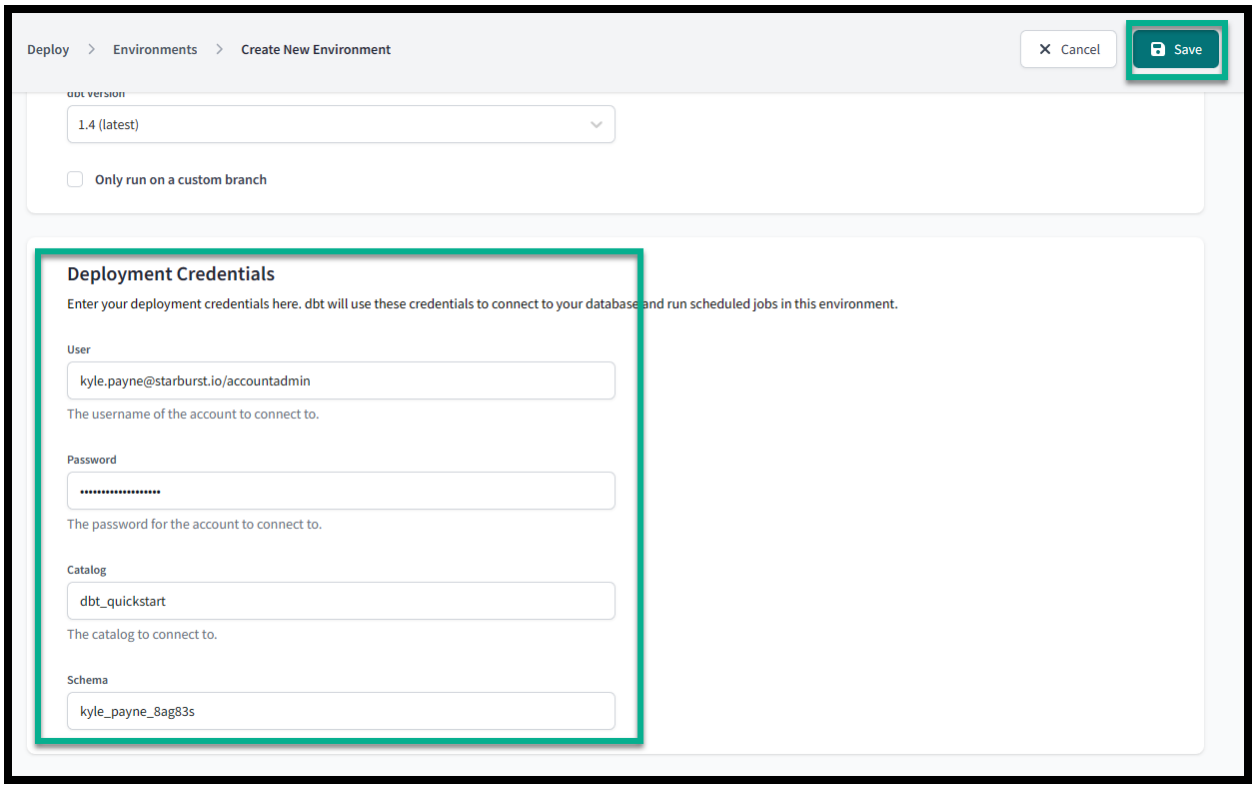
- Copy the **User** from your Starburst Galaxy cluster's **Connection information** and paste it into the box under **User**. Remember, the user name must include your role at the end (ex. `/accountadmin`).

- In the box under **Password**, type the password for your Starburst Galaxy user.

- In the box under **Catalog**, type `dbt_quickstart`.

- In the box under **Schema**, type the schema name you have been using for these labs (ex. `structure_kyle_payne_8ag83s`).

- Click **Save** (skip **Test Connection**).

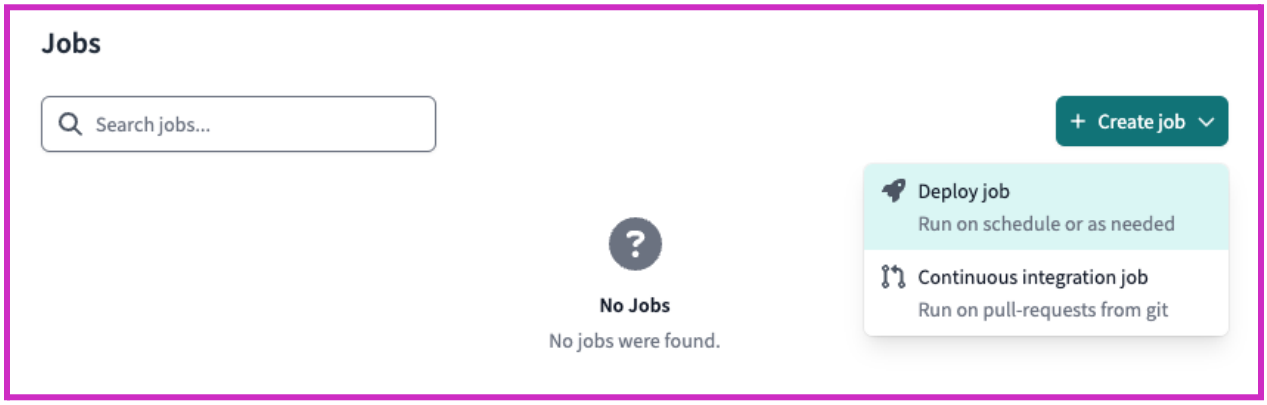


Step 3 - Create a job

Jobs are a set of dbt commands that you want to run on a schedule. For example, `dbt run` and `dbt test`. As the Jaffle Shop business gains more customers, and those customers create more orders, you will see more records added to your source data.

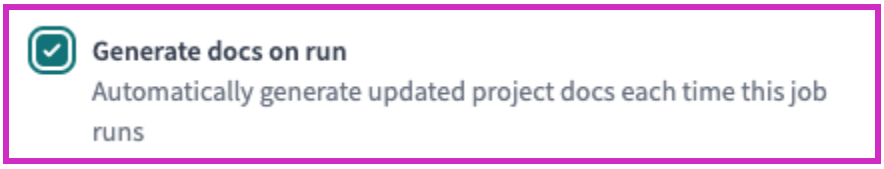
Because you materialized the three `struc_*` models as tables, you'll need to periodically rebuild them to ensure that the data stays up-to-date. This update will happen when you run a job.

Select the Deploy job option associated with the **+ Create job** button which is on the right side of the **Jobs** section.



In the **Job settings** section, provide a meaningful **Job name**.

In the **Execution settings** section, check the box for **Generate docs on run**.

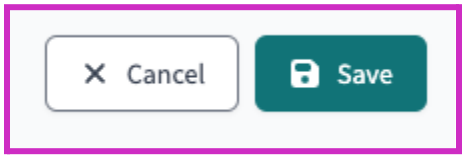


Just above this, notice **Commands** already includes `dbt build`.

Use the **+ Add command** link to include two more commands; `dbt run` and `dbt test`.

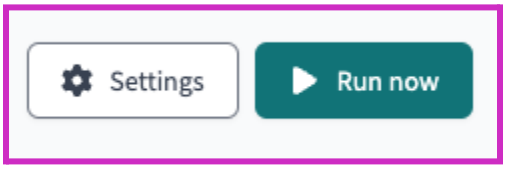
For this exercise, do *not* set a schedule for your project to run. While a true production project should run regularly, there is no need to run this example on a schedule. Scheduling a job is sometimes referred to as *deploying a project*.

Click **Save** in the upper-right of the screen.

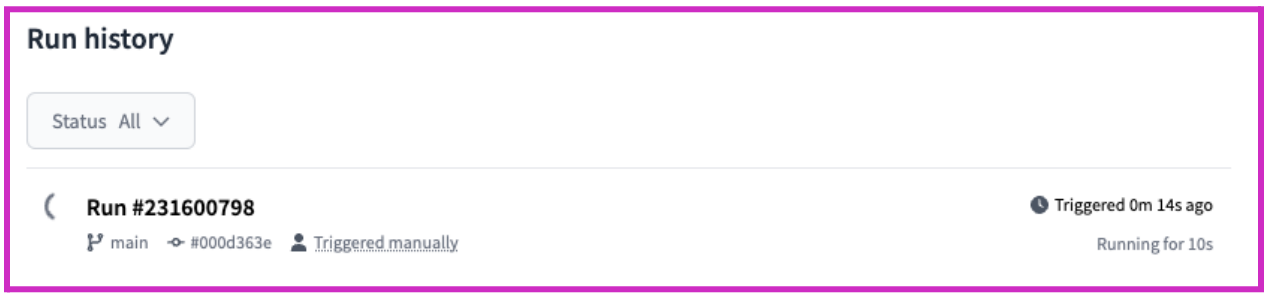


Step 4 - Run a job

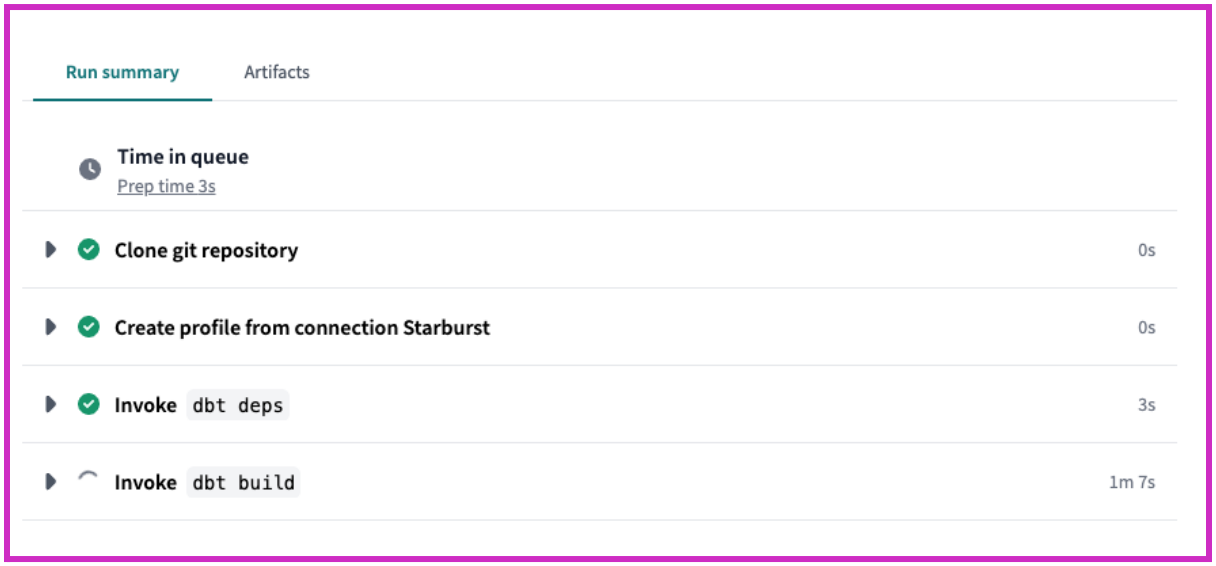
Click **Run now**.



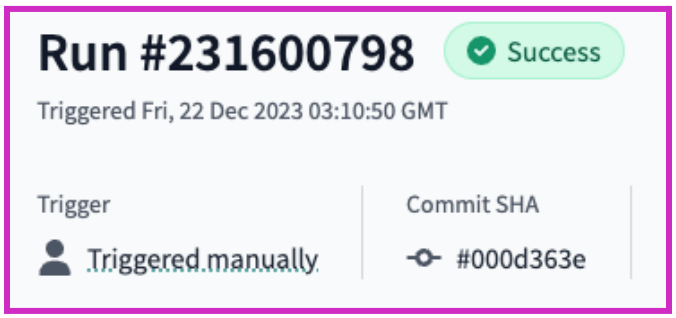
Refresh your browser to monitor the job and see the status change.



Once it is running, click on the job entry in the list (there should only be one job entry as shown above) to see the **Run summary**.



Once all the steps are complete, scroll back to the top to confirm the **Success**.



When you finish editing a model, you should always commit your changes to make sure the repository has an up-to-date version of your code. In a production environment, you can create jobs to run commands on a schedule, saving you time and effort.

You have now completed your dbt project, which means you are finished building a data transformation pipeline with dbt and Starburst.

END OF LAB EXERCISE