databricks × dbt Labs

Step-by-Step Guide

dbt With Databricks

Build a modern data stack
with dbt and Databricks

PRADEEP ANANDAPU | FEI LANG | AMY CHEN | BOBBY BIRSTOCK

Contents

AUTHORS:

Pradeep Anandapu
Senior Solutions Architect, Databricks

Fei Lang
Partner Solutions Architect, Databricks

Amy Chen
Partner Engineering Manager, dbt Labs

Bobby Birstock
Partner Engineer, dbt Labs

databricks × dbt Labs

Part 1: Databricks Setup

01

Create a Databricks trial account with ADMIN access to the workspace

Complete the two steps below following the **Get Started With Databricks Guide** on AWS or Azure before moving to the next section:

- Sign up for a Databricks free trial account
- Set up your workspace

02

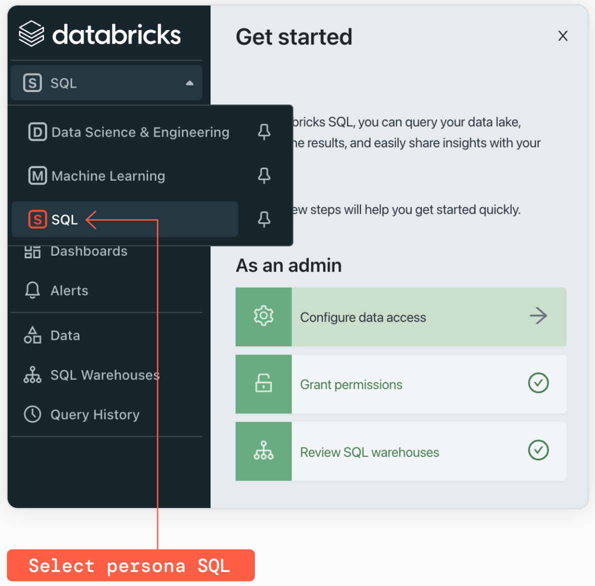
Ensure access to Databricks Partner Connect

Make sure your Databricks account, workspace and the signed-in user all meet the requirements for Partner Connect on AWS or Azure

03

Log in to your Databricks workspace. Navigate to Databricks SQL and SQL Warehouse.

In the Databricks workspace, on the left-side console, select persona **SQL** to enter Databricks SQL UI



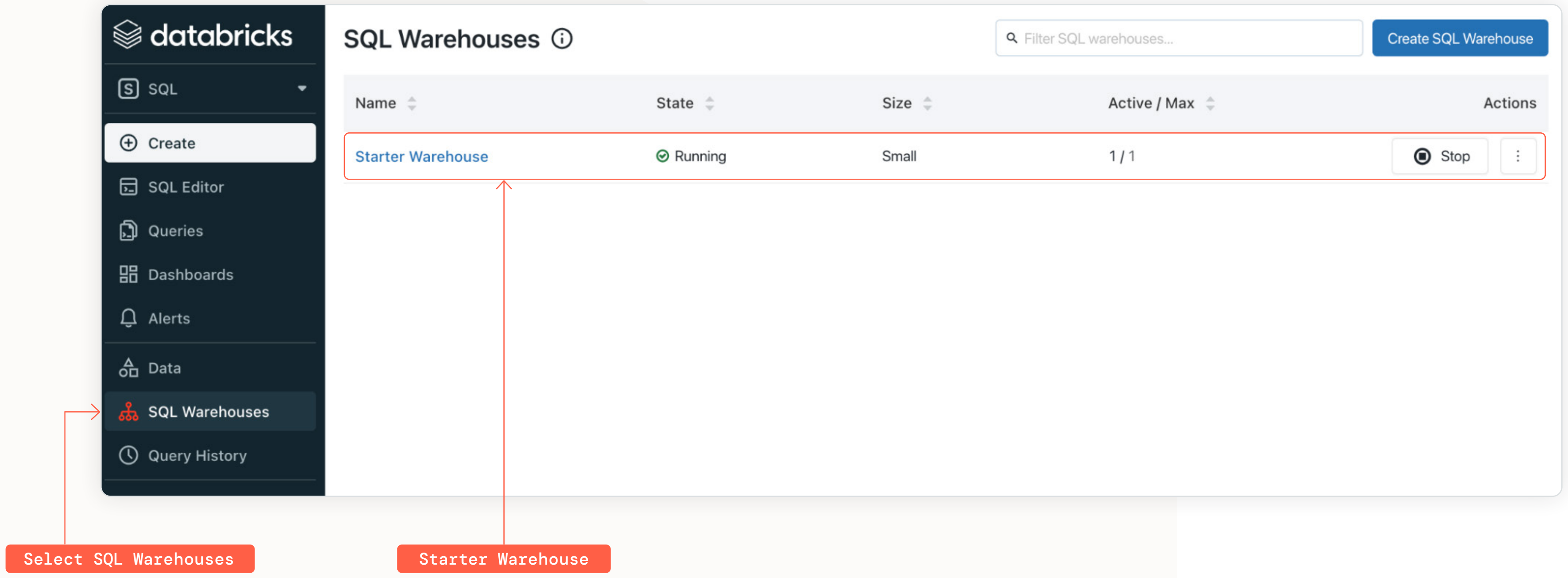
ricks SQL, you can query your data lake, ...e results, and easily share insights with your

...w steps will help you get started quickly.

Part 1: Databricks Setup

03 CONTINUED

Select **SQL Warehouses** from the left-side navigation bar. A SQL warehouse (formerly SQL endpoint) is a compute resource that lets you run SQL commands on data objects within the Databricks environment. A small SQL warehouse called **Starter Warehouse** has been created to help you get started. Click on the start button to start the warehouse.



Name	State	Size	Active / Max	Actions
Starter Warehouse	Running	Small	1 / 1	Stop

Part 1: Databricks Setup

04

Open SQL Editor and load sample data

- In the left-side menu, choose SQL Editor
- Select **Starter Warehouse** from the drop-down menu
- Run the following five lines of code in your SQL Editor. This will create the **retail** schema for the raw sample data and will also create the **dbt_user** schema, which we'll use as part of the development environment.

```sql
-- the default schema will be where our production data lives
DROP SCHEMA IF EXISTS retail cascade;
DROP SCHEMA IF EXISTS dbt_user cascade;
CREATE SCHEMA retail; -- our raw sample data will go here
CREATE SCHEMA dbt_user; -- our development data will go here
GRANT ALL PRIVILEGES ON SCHEMA retail TO users;
GRANT ALL PRIVILEGES ON SCHEMA dbt_user TO users;
USE SCHEMA retail;
```

- Load sample data — run the following three lines of code in your SQL Editor:

```sql
CREATE TABLE retail.customers
    USING csv
    OPTIONS (path"/databricks-datasets/retail-org/customers/customers.csv", header "true");
CREATE TABLE retail.loyalty_segments
    USING csv
    OPTIONS (path"/databricks-datasets/retail-org/loyalty_segments/loyalty_segment.csv", header "true");
CREATE TABLE retail.sales_orders
    USING json
    OPTIONS (path"/databricks-datasets/retail-org/sales_orders/part-00000-tid-1771549084454148016-e2275afd-a5bb-40ed-
b044-1774c0fdab2b-105592-1-c000.json", header "true");
```

- Query loaded sample source data — make sure you can access the tables that you just created by running the following lines:

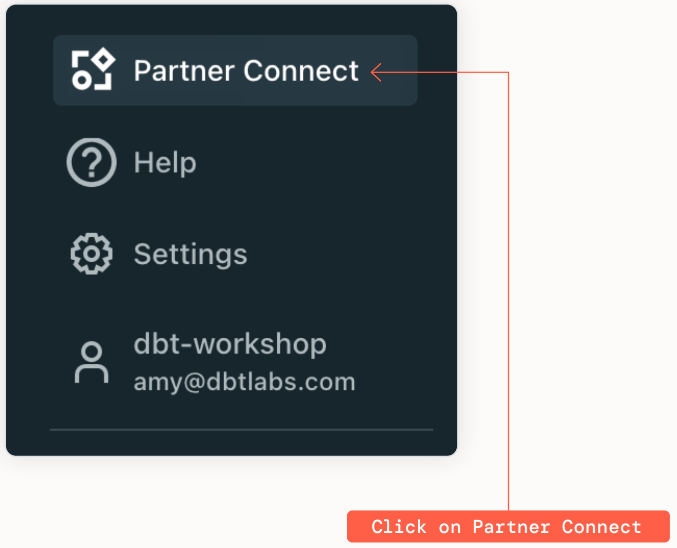
```sql
SELECT * FROM retail.customers limit 10;
SELECT * FROM retail.loyalty_segments limit 10;
SELECT * FROM retail.sales_orders limit 10;
```

Part 1: Databricks Setup

05

Use Partner Connect to set up dbt Cloud

- In the Databricks workspace, on the left-side console, click on **Partner Connect**



- Select the dbt tile under **Data preparation and transformation**
- You should now see a pop-up that says **Connect to partner** for dbt. You will be asked to choose the schemas that you want to use with dbt Cloud. This step is to grant read access to the data and metadata of the selected schemas for our dbt project later.

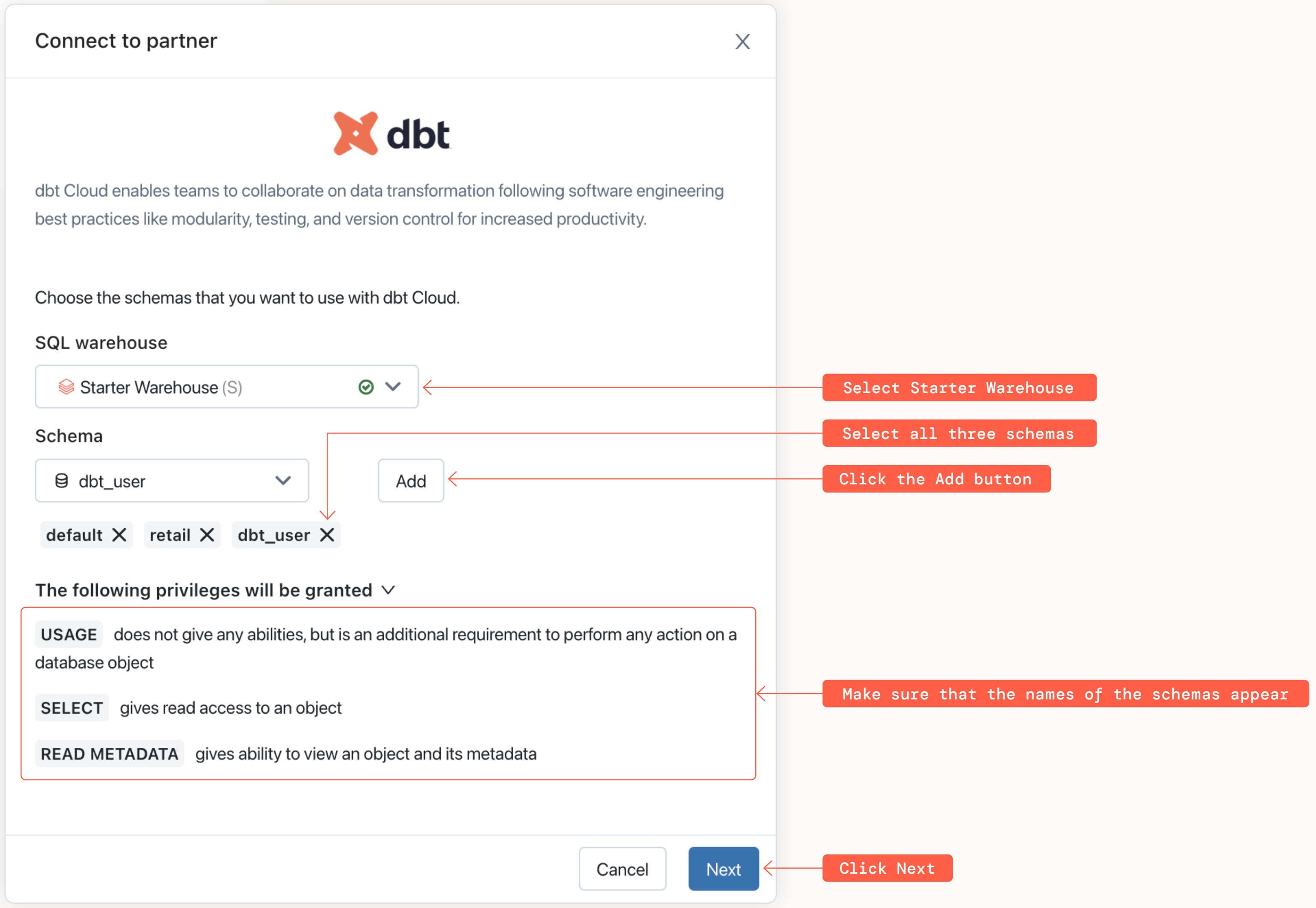
Part 1: Databricks Setup

05 CONTINUED

Complete the following steps on this pop-up:

1. Select Starter Warehouse from the **SQL warehouse** drop-down list and start it if you haven't already. Make sure it's running (when you see a green check mark next to it).
2. Select all three schemas (database) from the **Schema** drop-down list and add them. The three schemas are **default**, **retail** and **dbt_users**. After you select each schema from the drop-down menu, click the **Add** button. Make sure that the names of the schemas appear below the drop-down menu as they do in the screenshot to the right to confirm that you've added all three schemas.
3. **USAGE**, **SELECT** and **READ_METADATA** privileges will be granted to the select schemas

Click on **Next**.



Connect to partner

dbt Cloud enables teams to collaborate on data transformation following software engineering best practices like modularity, testing, and version control for increased productivity.

Choose the schemas that you want to use with dbt Cloud.

SQL warehouse: Starter Warehouse (S)

Schema: dbt_user

default, retail, dbt_user

The following privileges will be granted

USAGE does not give any abilities, but is an additional requirement to perform any action on a database object

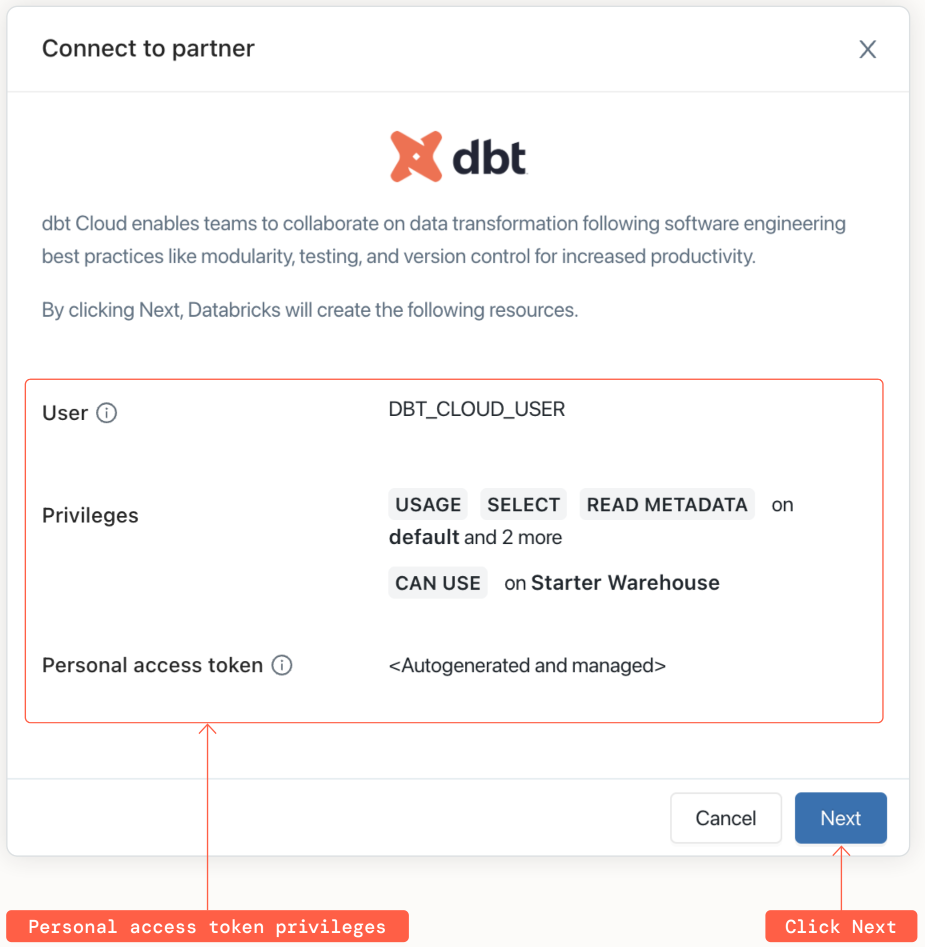
SELECT gives read access to an object

READ METADATA gives ability to view an object and its metadata

Part 1: Databricks Setup

05 CONTINUED

Click on **Next** on the next pop-up. This will create a dedicated **DBT_CLOUD_USER**, **personal access token** with the granted privileges to the select schemas and Databricks SQL warehouse.

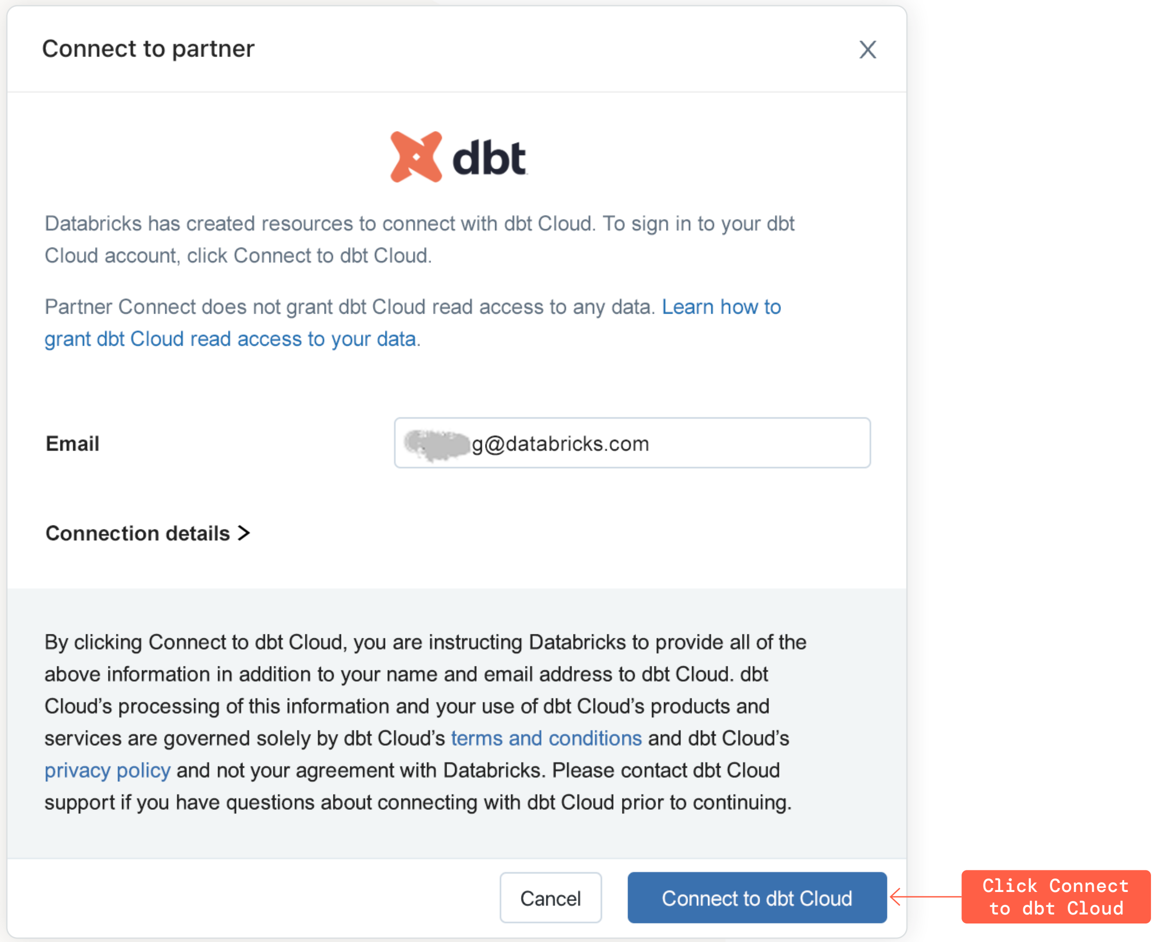


dbt Cloud enables teams to collaborate on data transformation following software engineering best practices like modularity, testing, and version control for increased productivity.

By clicking Next, Databricks will create the following resources.

User	DBT_CLOUD_USER
Privileges	USAGE SELECT READ METADATA on **default** and 2 more CAN USE on **Starter Warehouse**
Personal access token	<Autogenerated and managed>

On the next pop-up, it prepopulates the email address tied to the workspace. This email address will be used for the sign-up for the dbt Cloud trial later. Click on **Connect to dbt Cloud**.



Databricks has created resources to connect with dbt Cloud. To sign in to your dbt Cloud account, click Connect to dbt Cloud.

Partner Connect does not grant dbt Cloud read access to any data. Learn how to grant dbt Cloud read access to your data.

Email: [redacted]g@databricks.com

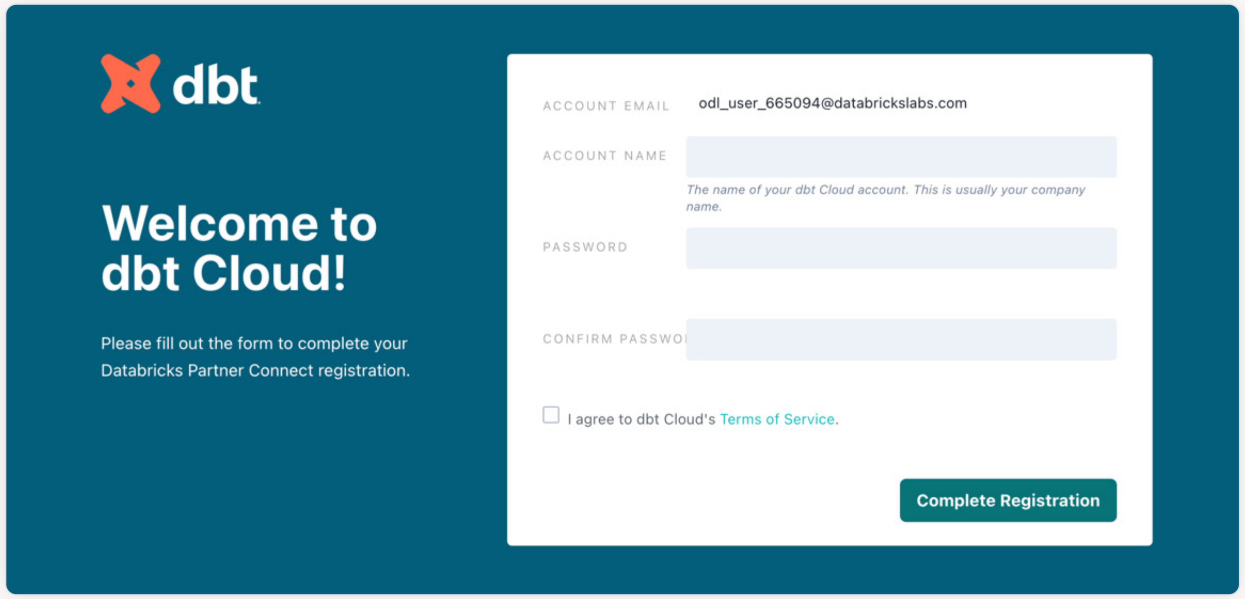
By clicking Connect to dbt Cloud, you are instructing Databricks to provide all of the above information in addition to your name and email address to dbt Cloud. dbt Cloud's processing of this information and your use of dbt Cloud's products and services are governed solely by dbt Cloud's terms and conditions and dbt Cloud's privacy policy and not your agreement with Databricks. Please contact dbt Cloud support if you have questions about connecting with dbt Cloud prior to continuing.

Part 1: Databricks Setup

05 CONTINUED

After the new tab loads, you will see a form. Enter your account name (e.g., Acme Labs or Kyle's Sandbox) and password. **Save this username and password if you want to access dbt Cloud after this training.** (Note: Collapse the welcome prompt in the bottom right.)

After you have filled out the form and clicked on **Complete Registration**, you will be logged in to dbt Cloud automatically.



dbt

Welcome to dbt Cloud!

Please fill out the form to complete your Databricks Partner Connect registration.

ACCOUNT EMAIL	odl_user_665094@databrickslabs.com
ACCOUNT NAME	*The name of your dbt Cloud account. This is usually your company name.*
PASSWORD	
CONFIRM PASSWO	

☐ I agree to dbt Cloud's Terms of Service.

Part 2: dbt Setup

Note: If you land in dbt Cloud's **Beta UI**, please switch to **Classic UI** in the drop-down at the top right. Click on **Go back to Classic UI**. Ignore any prompts to switch to Beta UI.

01

Update your Development schema name

In the Databricks interface, navigate to **User Settings** in the left-hand menu

- Select the tab that says **Personal Access Tokens**
- Create a new token with the comment **dbt Cloud** and click **generate**
- Leave the token up in that tab and head over to your dbt Cloud tab. We will be using this to set up our development connection.

If using the classic dbt Cloud UI, do the following:

- Click on your profile icon in the top right corner and click **Profile** in the drop-down menu
- Under **Credentials** on the left, click on your project titled **Databricks Partner Connect Trial**
- Click **edit** to adjust your development credentials
  - For **token**, enter the personal access token from the Databricks workspace
  - For **schema**, enter **dbt_user**
  - For **threads**, you can leave this at 6
- Click **Save** in the top right

If using the new dbt Cloud UI, do the following:

- Click the gear icon in the upper right-hand corner and click **Profile Settings** in the drop-down menu
- On the left-hand side under **Your Profile**, click **Credentials**
- In the middle of the screen in the **Credentials** section, click the box that says **Databricks Partner Connect Trial**
- A window should open on the right-hand side of the screen; click **Edit** in the bottom right-hand corner
  - For **token**, enter the personal access token from the Databricks workspace
  - For **schema**, enter **dbt_user**
  - For **threads**, you can leave this at 6
- Click **Save** in the bottom right

Part 2: dbt Setup

02

Initialize dbt Cloud project

In the classic dbt Cloud UI, click the hamburger menu in the upper left-hand corner and select **Develop**.

In the new dbt Cloud UI, click **Develop** on the upper left side and click **Classic IDE** in the drop-down.

It might take a minute for your project to spin up for the first time as it establishes your git connection, clones your repo and tests the connection to the warehouse.

Above the file tree to the left, click **Initialize your project**. This builds out your folder structure with example models.

Make your initial commit by clicking **Commit**. Use the commit message **initial commit**. This creates the first commit to your managed repo and allows you to open a branch where you can add new dbt code.

03

Test your Databricks connection

Now you should be able to directly query data from your warehouse and execute dbt run. In the Scratchpad 1 tab, delete all text and paste the following warehouse-specific code into Scratchpad 1. Click **Preview**.

```
select * from retail.customers
```

In the command line bar at the bottom, type **dbt run** and click **Enter**. You should see the example models from the project build successfully within the run module.

Once your models have been built, pop back over to the Databricks workspace and refresh your schema list. You should be able to see your **dbt_user** schema now in your workspace, with the two sample models nested within it. Congratulations, you've built your first dbt models!

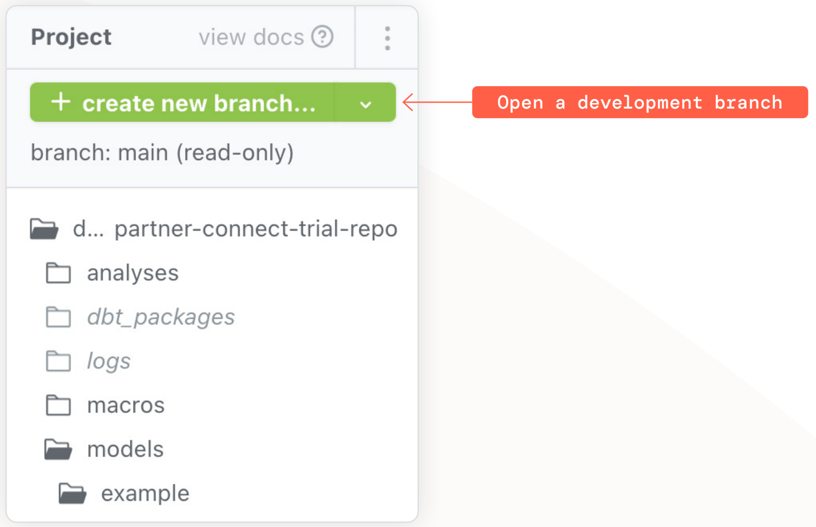
Part 2: dbt Setup

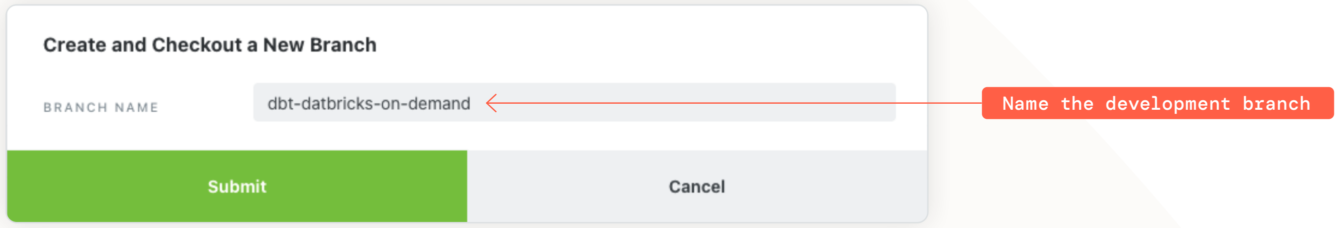
04

Create folders and files

Open a development branch called **dbt-databricks-on-demand**

- Delete the example folder in the models directory
- Create a new folder in models called **staging**
- Create a new folder in staging called **retail**
- Create a new file in retail called **_sources.yml**





Part 2: dbt Setup

05

Declare sources

Configure sources for the raw data that was loaded into Databricks, starting by opening the **_sources.yml** file.

Paste the following code into the file and click **Save** in the upper right corner:

```
version: 2

sources:
  - name: retail
    schema: retail
    tables:
      - name: customers
      - name: loyalty_segments
      - name: sales_order
```

Use the source macro to select from sources in the Scratchpad to confirm they've been declared successfully.

```
select * from {{ source('retail','customers') }}
```

Commit your work with the commit message **configure sources**.

Part 3: Transformation

01

Create staging models

Create two new files in the **models/staging/retail** folder: **stg_loyalty_segments.sql** and **stg_customers.sql**.

Copy and paste the SQL code at right into **stg_customers.sql** and click **Save**:

```
with source as (
    select * from {{ source('retail', 'customers') }}
),

renamed as (
    select
        customer_id as customer_id,
        loyalty_segment as loyalty_segment_id,
        cast(tax_id as int) as tax_id,
        tax_code as tax_code,
        customer_name as customer_name,
        state as state,
        city as city,
        case when postcode like '%-%'
            then cast(left(postcode,5) as int)
            else cast(postcode as int)
        end as postcode,
        street as street,
        case when number like '%.%'
            then cast(number as int)
            else number
        end as number,
        unit as unit,
        region as region,
        district as district,
        cast(lon as double) as longitude,
        cast(lat as double) as latitude,
        ship_to_address as ship_to_address,
        from_unixtime(valid_from,'yyyy-MM-dd') as valid_from_date,
        from_unixtime(valid_to,'yyyy-MM-dd') as valid_to_date,
        cast(units_purchased as int) as units_purchased
    from source
),

de_duped as (
    select * from renamed
    where valid_to_date is null
)

select * from de_duped
```

databricks × dbt Labs

Part 3: Transformation

01 CONTINUED

Copy and paste the SQL code at right into **stg_loyalty_segments.sql** and click **Save**:

```
with source as (
    select * from {{ source('retail','loyalty_segments') }}
),

staged as (
    select
        loyalty_segment_id,
        loyalty_segment_description,
        unit_threshold,
        from_unixtime(valid_from, 'yyyy-MM-dd') as valid_from_date,
        from_unixtime(valid_to, 'yyyy-MM-dd') as valid_to_date
    from source
)

select * from staged
```

Part 3: Transformation

02

Create marts models

Create a new folder in **models** called **marts**.

Create a new folder in **marts** called **core**.

Create a new file in **core** called **dim_loyalty_segments.sql**.

Copy and paste the SQL code at right into **dim_loyalty_segments.sql** and click **Save**:

```sql
with segments as (
    select * from {{ ref('stg_loyalty_segments') }}
),

customers as (
    select * from {{ ref('stg_customers') }}
),

segment_customers as (
    select
        loyalty_segment_id,
        count(*) as number_of_customers
    from customers
    group by 1
    order by 1
),

final as (
    select
        segments.loyalty_segment_id,
        segments.loyalty_segment_description,
        segment_customers.number_of_customers
    from segments
    left join segment_customers on segments.loyalty_segment_id = segment_customers.loyalty_segment_id
)
select * from final
```

Part 3: Transformation

03

Configure models

Open up the **dbt_project.yml** file and scroll to the bottom of the file.

Copy and paste the following code block into the file, making sure that your cursor is not indented when you paste and that you click **Save** when you're done:

```
models:
  my_new_project:
    marts:
      +materialized: table
    staging:
      +materialized: view
```

This configures the models in the marts folder to build as tables and the models in the staging folder to build as views.

04

Execute models

Go to the command line at the bottom of the screen and type in **dbt run**, then click **enter** to run all of the models that you just created.

After the models finish running, pop over to the Databricks workspace and refresh the schema list. You should be able to see your new staging and marts models alongside the example models in your development schema.

Commit with message staging and marts models.

Part 4: Testing and Documentation

01

Implement dbt tests and generate documentation

In the **models/staging/retail** folder create a new file called **_schema.yml**.

Copy and paste the code at right into **_schema.yml** and click **Save**:

```yaml
version: 2

models:
  - name: stg_customers
    description: Staged data for customers
    columns:
      - name: customer_id
        description: The primary key for customers
        tests:
          - unique
          - not_null
      - name: another column 1
        description: custom description 1
      - name: another column 2
        description: custom description 2

  - name: stg_loyalty_segments
    description: Staged data for loyalty segments
    columns:
      - name: loyalty_segment_id
        description: The primary key for customers
        tests:
          - unique
          - not_null
```

Replace the placeholder column names and descriptions under **stg_customers** with two additional columns from the model.

To test your models, go to the command line and type in **dbt test**, then click **Enter**. You should see the test results the same way that model results appeared in the command box.

To generate documentation for your project, type **dbt docs generate** into the command bar and click **Enter**.

After it completes, a tooltip should appear next to the **view docs** link in the upper left-hand corner of the screen. Click the link to open up the documentation site.

You can browse the documentation site, searching by model name, to view the descriptions that were added to the project in **_schema.yml**.

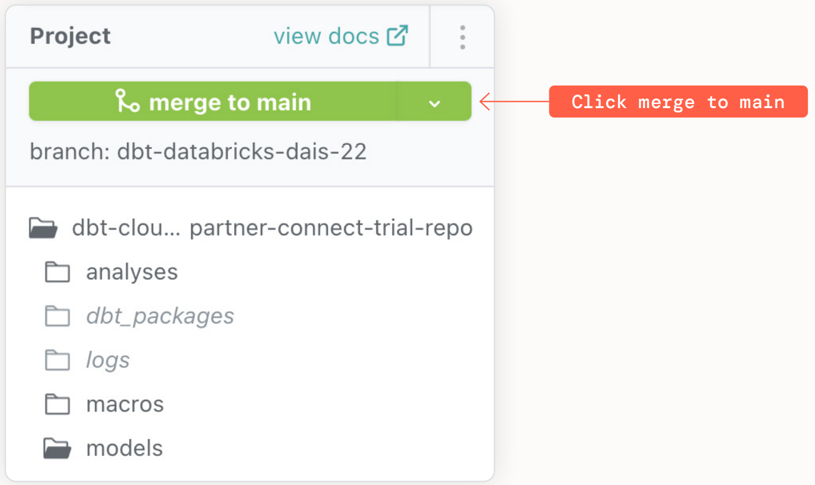
Go back to the IDE and commit your work with the message **add tests and documentation**.

Part 5: Deployment

01

Promote code from development to deployment

Once all of your work has been committed to your feature branch, click the **merge to main** button in the upper left-hand corner of the screen. This will merge your changes from your feature branch to the main branch within the managed repository.

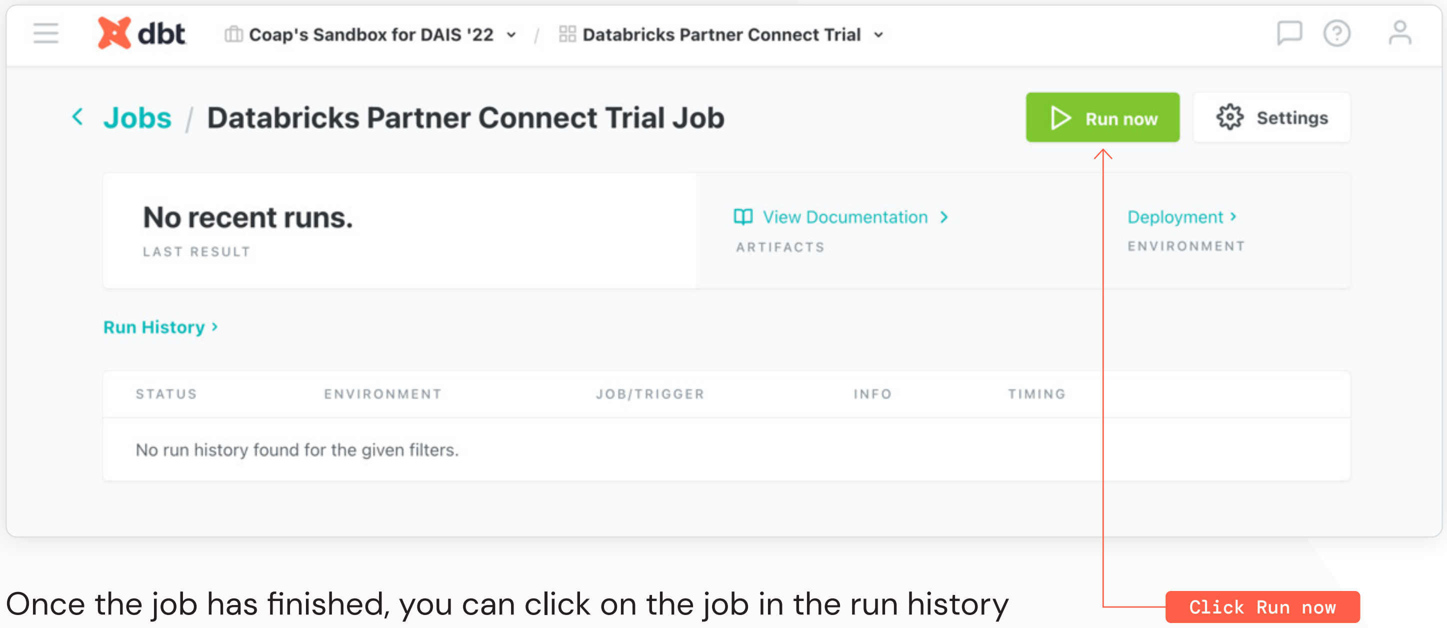


02

Run dbt Cloud job

Click the hamburger menu in the upper left-hand corner of the screen and click **Jobs**.

Click on **Databricks Partner Connect Trial Job** to open it up and then click the **Run now** button in the upper right-hand corner to kick off the job. You should see the job start running, which involves running the dbt commands configured within the job settings using the code on the main branch and building the associated models in your deployment environment.



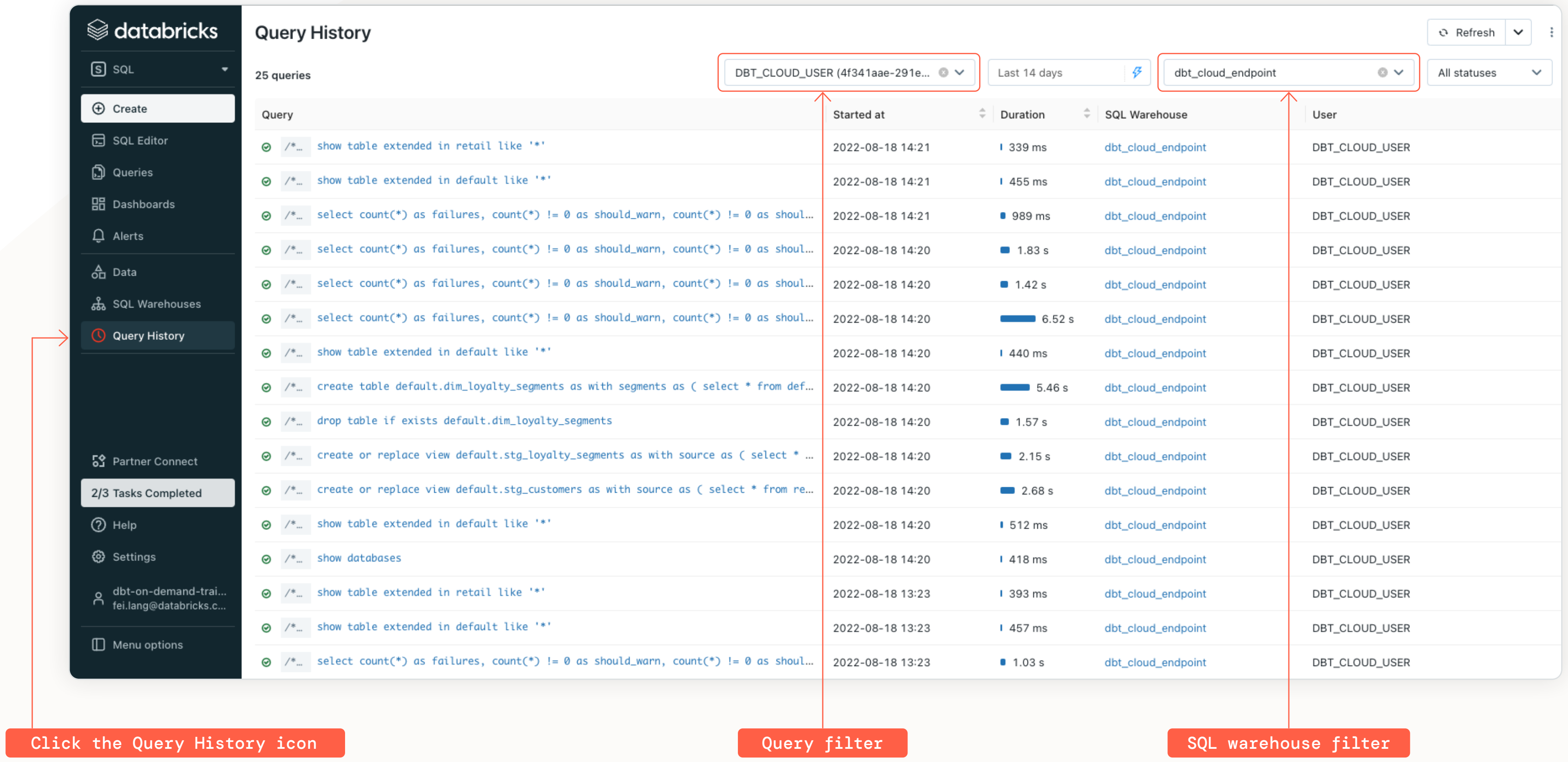
Once the job has finished, you can click on the job in the run history and view all of the logs and artifacts associated with the run.

Part 6: Databricks SQL — Querying and Analysis

01

Review dbt Cloud compiled queries in Databricks SQL Query History

Click the **Query History** icon on the left-side navigation bar to navigate to the **Query History** screen. We can see a list of SQL queries performed using SQL warehouse in this workspace. To find the queries that got pushed down from dbt Cloud, we can filter the list by user as **"DBT_CLOUD_USER"** and SQL warehouse as **"dbt_cloud_endpoint"**. Remember, those were automatically created in Databricks Partner Connect when creating the connection to dbt Cloud.



Query History

25 queries

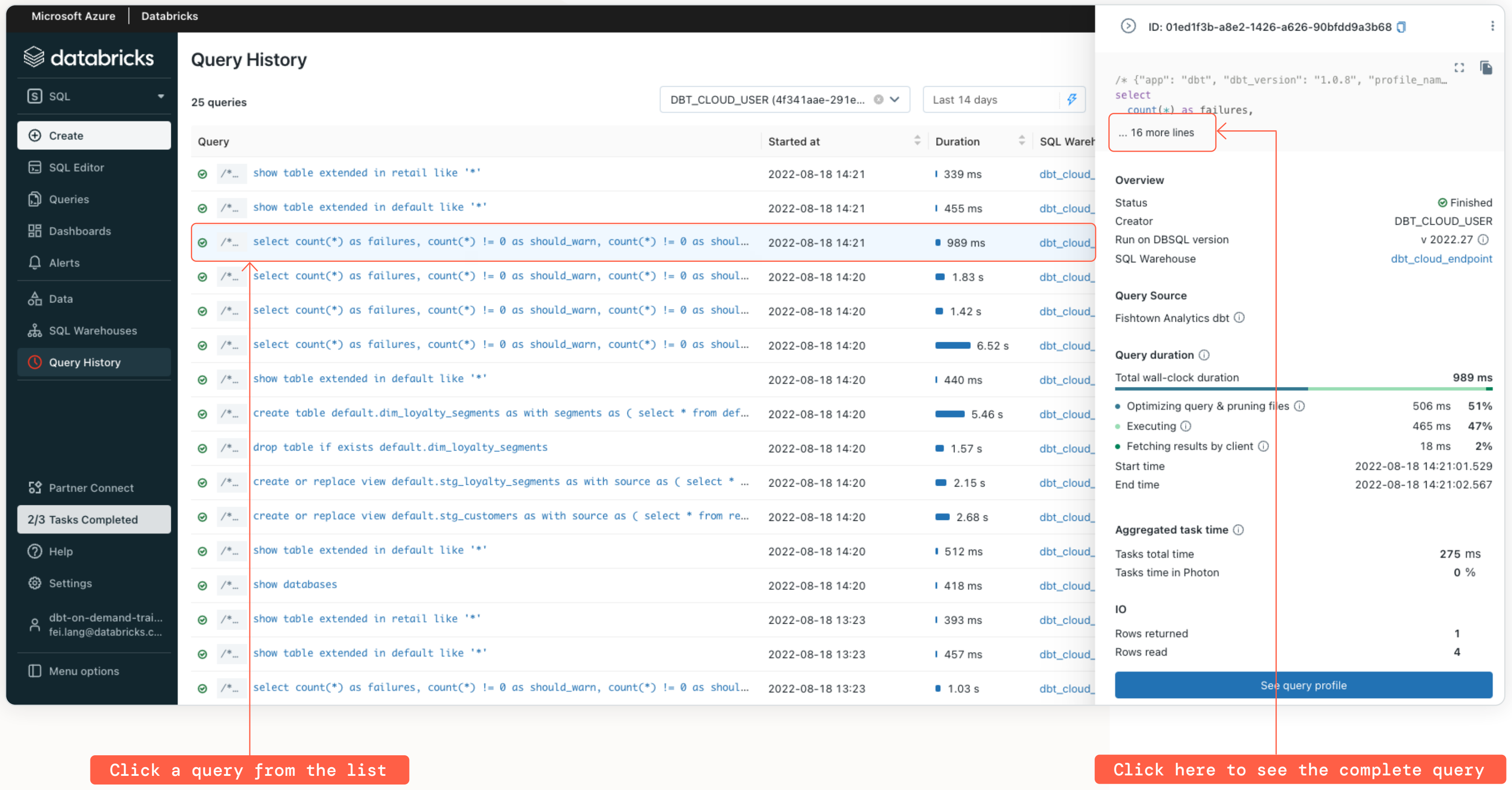
DBT_CLOUD_USER (4f341aae-291e... | Last 14 days | dbt_cloud_endpoint | All statuses

Query	Started at	Duration	SQL Warehouse	User
/*... show table extended in retail like '*'	2022-08-18 14:21	339 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... show table extended in default like '*'	2022-08-18 14:21	455 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:21	989 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:20	1.83 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:20	1.42 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:20	6.52 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... show table extended in default like '*'	2022-08-18 14:20	440 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... create table default.dim_loyalty_segments as with segments as (select * from def...	2022-08-18 14:20	5.46 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... drop table if exists default.dim_loyalty_segments	2022-08-18 14:20	1.57 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... create or replace view default.stg_loyalty_segments as with source as (select * ...	2022-08-18 14:20	2.15 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... create or replace view default.stg_customers as with source as (select * from re...	2022-08-18 14:20	2.68 s	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... show table extended in default like '*'	2022-08-18 14:20	512 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... show databases	2022-08-18 14:20	418 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... show table extended in retail like '*'	2022-08-18 13:23	393 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... show table extended in default like '*'	2022-08-18 13:23	457 ms	dbt_cloud_endpoint	DBT_CLOUD_USER
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 13:23	1.03 s	dbt_cloud_endpoint	DBT_CLOUD_USER

Part 6: Databricks SQL — Querying and Analysis

01 CONTINUED

Click a query from the list to view the details of it, including its duration, I/O performance, etc.



Query History

25 queries

DBT_CLOUD_USER (4f341aae-291e... | Last 14 days

Query	Started at	Duration	SQL Wareh
/*... show table extended in retail like '*'	2022-08-18 14:21	339 ms	dbt_cloud_
/*... show table extended in default like '*'	2022-08-18 14:21	455 ms	dbt_cloud_
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:21	989 ms	dbt_cloud_
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:20	1.83 s	dbt_cloud_
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:20	1.42 s	dbt_cloud_
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 14:20	6.52 s	dbt_cloud_
/*... show table extended in default like '*'	2022-08-18 14:20	440 ms	dbt_cloud_
/*... create table default.dim_loyalty_segments as with segments as (select * from def...	2022-08-18 14:20	5.46 s	dbt_cloud_
/*... drop table if exists default.dim_loyalty_segments	2022-08-18 14:20	1.57 s	dbt_cloud_
/*... create or replace view default.stg_loyalty_segments as with source as (select * ...	2022-08-18 14:20	2.15 s	dbt_cloud_
/*... create or replace view default.stg_customers as with source as (select * from re...	2022-08-18 14:20	2.68 s	dbt_cloud_
/*... show table extended in default like '*'	2022-08-18 14:20	512 ms	dbt_cloud_
/*... show databases	2022-08-18 14:20	418 ms	dbt_cloud_
/*... show table extended in retail like '*'	2022-08-18 13:23	393 ms	dbt_cloud_
/*... show table extended in default like '*'	2022-08-18 13:23	457 ms	dbt_cloud_
/*... select count(*) as failures, count(*) != 0 as should_warn, count(*) != 0 as shoul...	2022-08-18 13:23	1.03 s	dbt_cloud_

ID: 01ed1f3b-a8e2-1426-a626-90bfdd9a3b68

```
/* {"app": "dbt", "dbt_version": "1.0.8", "profile_nam…
select
  count(*) as failures,
... 16 more lines
```

Overview

Status	Finished
Creator	DBT_CLOUD_USER
Run on DBSQL version	v 2022.27
SQL Warehouse	dbt_cloud_endpoint

Query Source

Fishtown Analytics dbt

Query duration

Total wall-clock duration		989 ms
Optimizing query & pruning files	506 ms	51%
Executing	465 ms	47%
Fetching results by client	18 ms	2%
Start time		2022-08-18 14:21:01.529
End time		2022-08-18 14:21:02.567

Aggregated task time

Tasks total time	275 ms
Tasks time in Photon	0 %

IO

Rows returned	1
Rows read	4

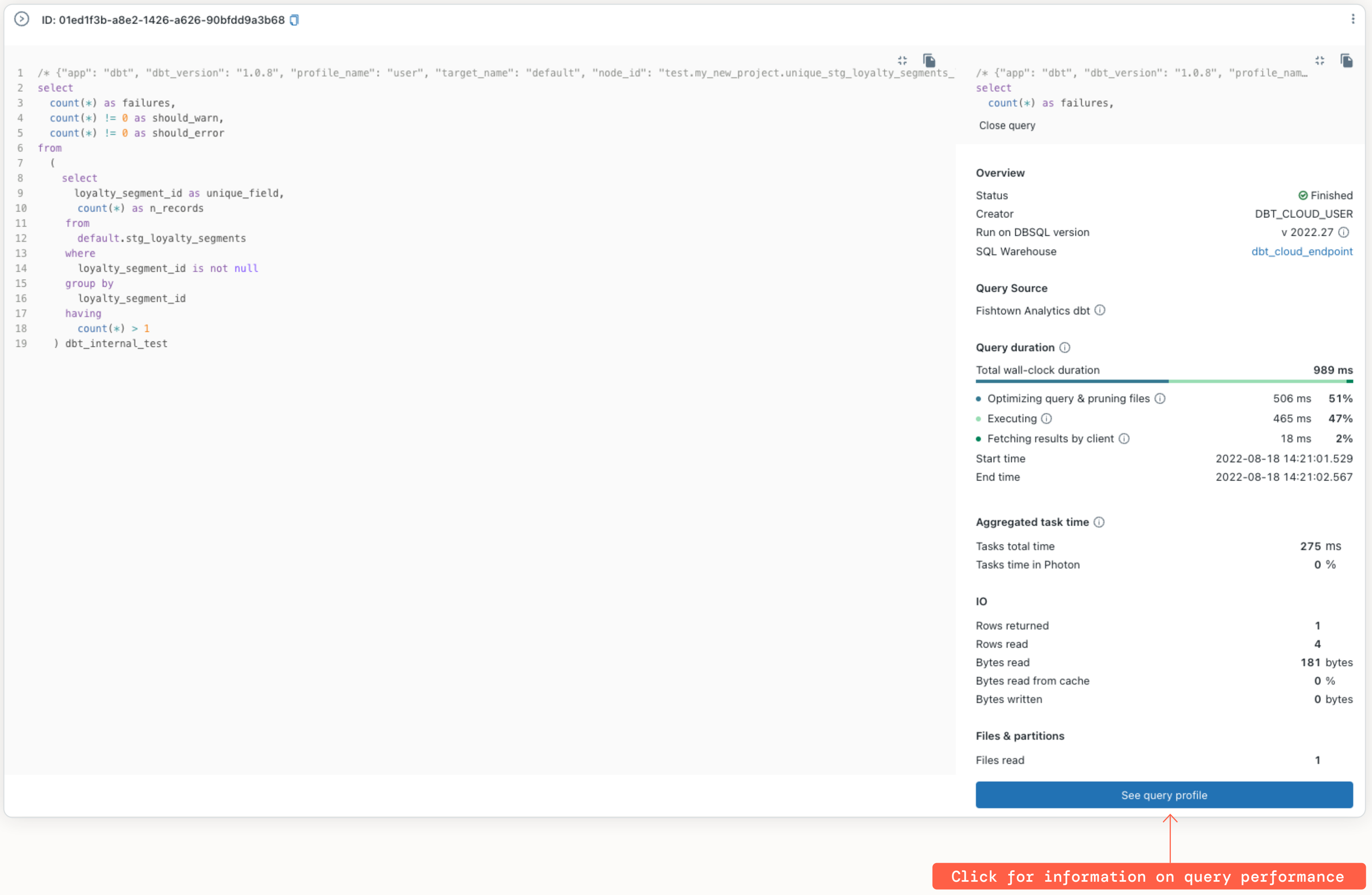
See query profile

Part 6: Databricks SQL — Querying and Analysis

01 CONTINUED

You can see the complete query by clicking "… x more lines" if the query is not shown completely. Like this, we can see the entire query got pushed down from dbt Cloud.

Click **View query profile** at the bottom for more detailed information about the query's performance, such as its execution plan and so on.



ID: 01ed1f3b-a8e2-1426-a626-90bfdd9a3b68

```
/* {"app": "dbt", "dbt_version": "1.0.8", "profile_name": "user", "target_name": "default", "node_id": "test.my_new_project.unique_stg_loyalty_segments_
select
  count(*) as failures,
  count(*) != 0 as should_warn,
  count(*) != 0 as should_error
from
  (
    select
      loyalty_segment_id as unique_field,
      count(*) as n_records
    from
      default.stg_loyalty_segments
    where
      loyalty_segment_id is not null
    group by
      loyalty_segment_id
    having
      count(*) > 1
  ) dbt_internal_test
```

```
/* {"app": "dbt", "dbt_version": "1.0.8", "profile_nam…
select
  count(*) as failures,
```

Close query

Overview

Status	Finished
Creator	DBT_CLOUD_USER
Run on DBSQL version	v 2022.27
SQL Warehouse	dbt_cloud_endpoint

Query Source

Fishtown Analytics dbt

Query duration

Total wall-clock duration		989 ms
Optimizing query & pruning files	506 ms	51%
Executing	465 ms	47%
Fetching results by client	18 ms	2%
Start time		2022-08-18 14:21:01.529
End time		2022-08-18 14:21:02.567

Aggregated task time

Tasks total time	275 ms
Tasks time in Photon	0 %

IO

Rows returned	1
Rows read	4
Bytes read	181 bytes
Bytes read from cache	0 %
Bytes written	0 bytes

Files & partitions

Files read	1

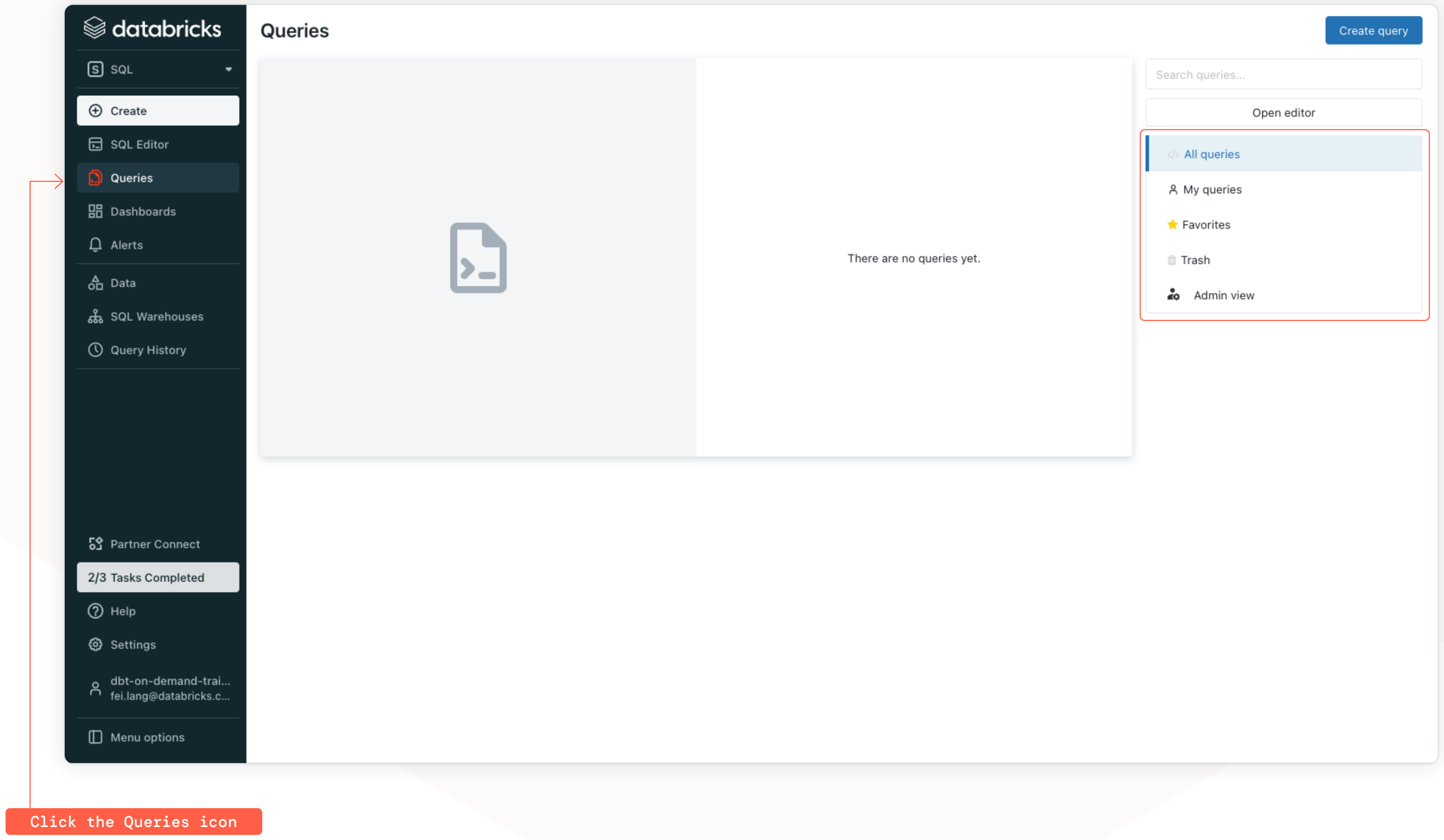
Part 6: Databricks SQL — Querying and Analysis

02

Ad hoc data analysis and visualization in Databricks SQL

Click the **Queries** icon on the left-side navigation bar to navigate to the **Queries** page. For each query listed, you can see who created it, when it was created, when it was last executed or if there is a refresh schedule attached. Within this view, you can create a new query or search for existing queries. Different from the **Query History** page, which we just went through earlier, the queries shown in the list on the **Queries** page here are saved ones which can be rerun manually or automatically.

You can view queries you've created by clicking **My queries** or **Favorites** or deleted queries that have been sent to **Trash**. The **Admin view** gives you access to see all queries created and delete queries in this workspace. However, an admin can't edit a query if it is not shared with the admin.



There are no queries yet.

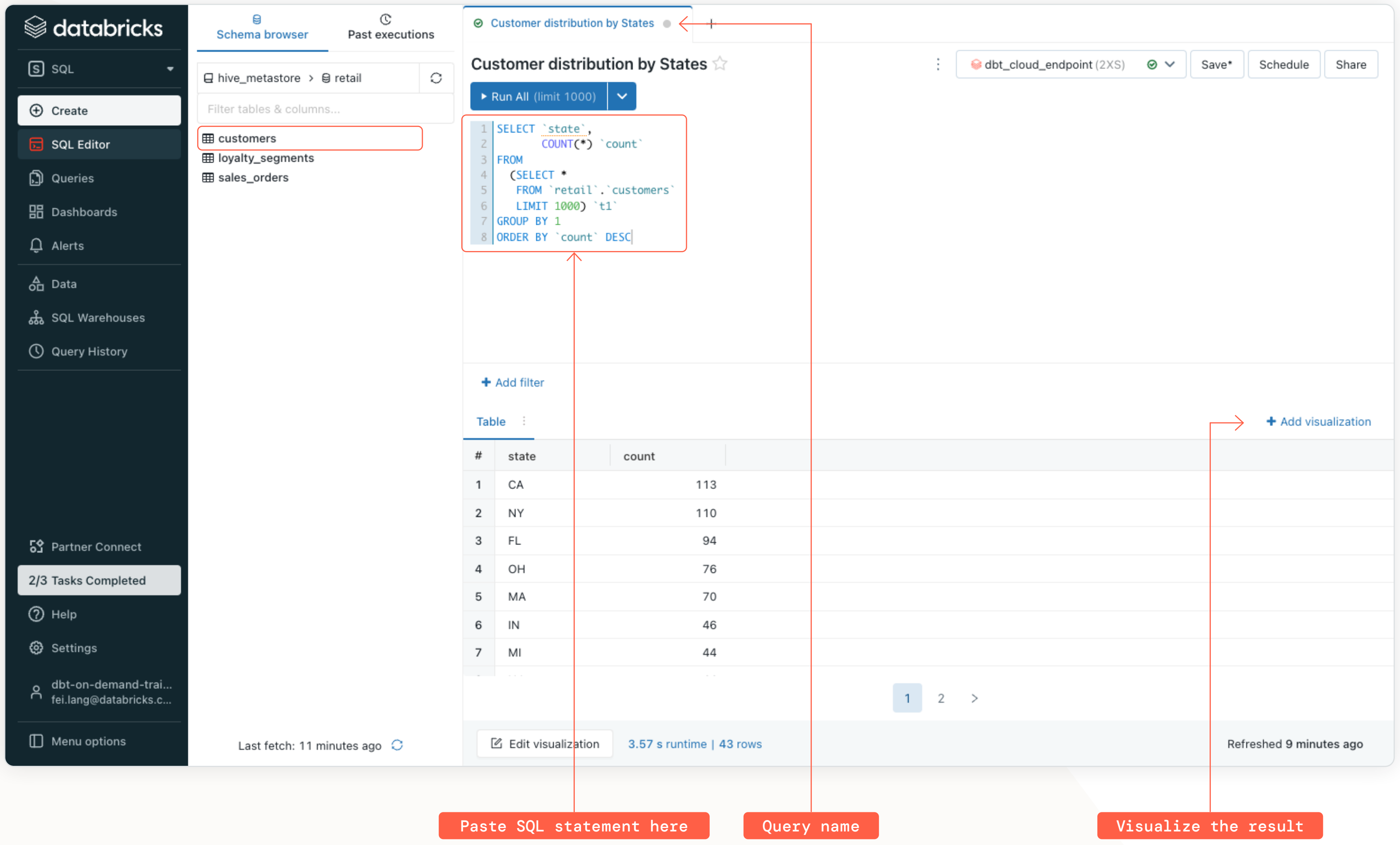
Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

We want to create a simple customer distribution by states using data set customers. To start, click **Create query** from the top right corner to open a new query editor. Then copy and paste the following SQL statement. Replace **New Query** in the tab with a meaningful name — for example, "Customer distribution by States."

```sql
SELECT `state`,
       COUNT(*) `count`
FROM
  (SELECT *
   FROM `retail`.`fct_customer`
   LIMIT 1000) `t1`
GROUP BY 1
ORDER BY `count` DESC
```

Click **run** until you see the query result at the bottom of the screen. Then, click **Add visualization** to visualize the result.



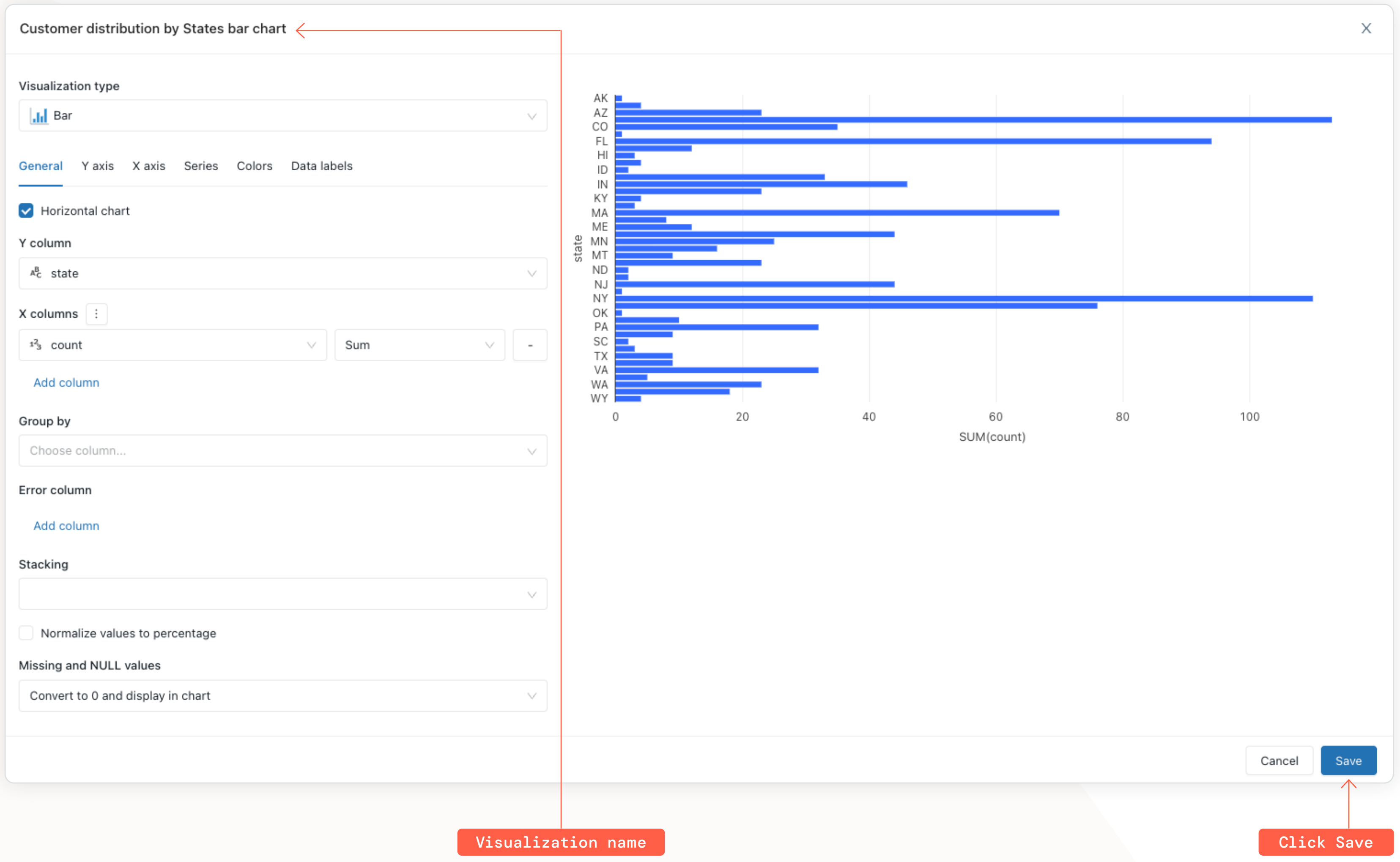
```sql
SELECT `state`,
       COUNT(*) `count`
FROM
  (SELECT *
   FROM `retail`.`customers`
   LIMIT 1000) `t1`
GROUP BY 1
ORDER BY `count` DESC
```

#	state	count
1	CA	113
2	NY	110
3	FL	94
4	OH	76
5	MA	70
6	IN	46
7	MI	44

Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

On the next screen, give the visualization a meaningful name and click the **Save** button at the bottom.



Customer distribution by States bar chart

Visualization type: Bar

Horizontal chart: ✓

Y column: state

X columns: count, Sum

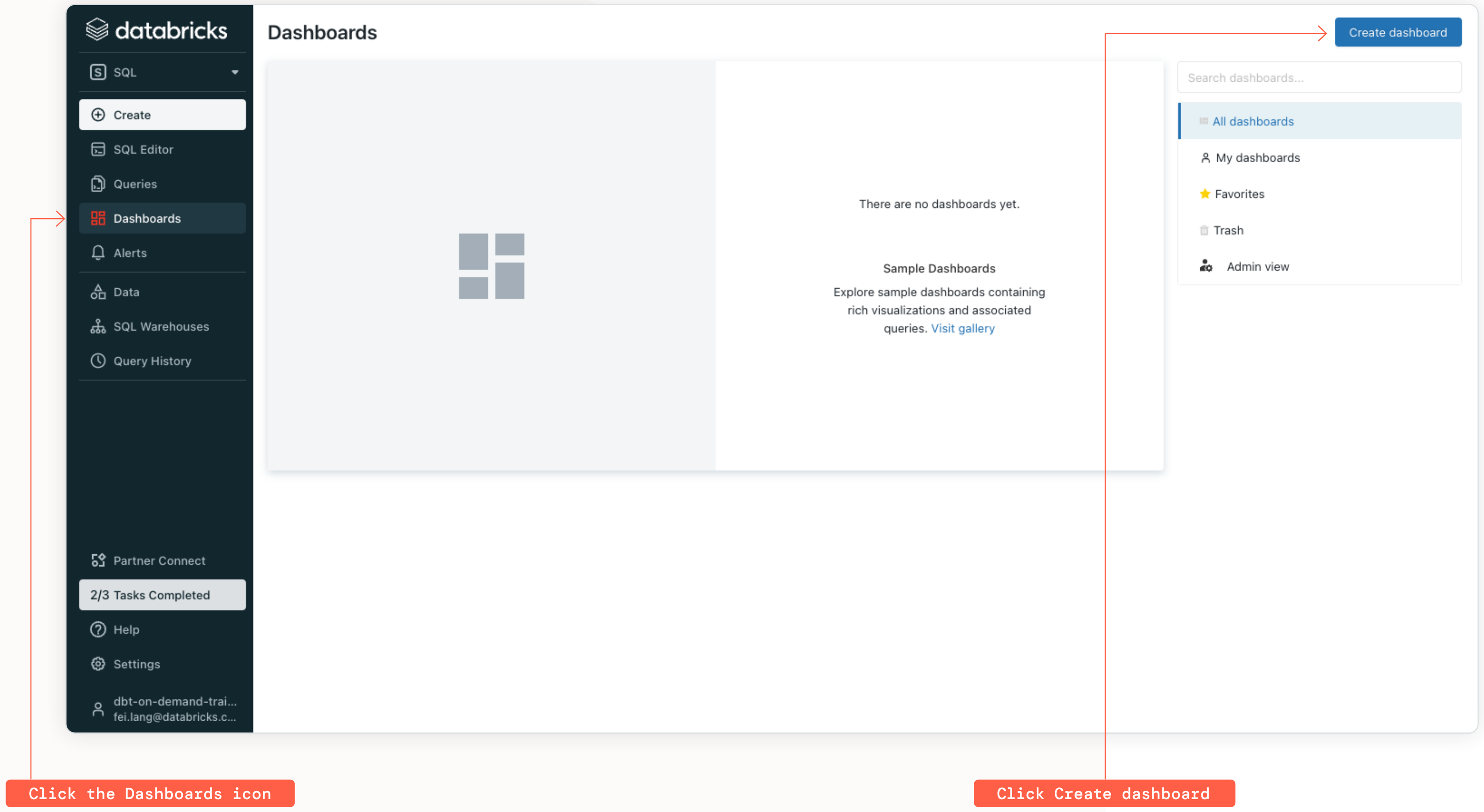
Group by: Choose column...

Missing and NULL values: Convert to 0 and display in chart

Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

Now we go to the Dashboards UI by clicking the **Dashboards** icon on the left-side navigation bar. Then click **Create dashboard**.



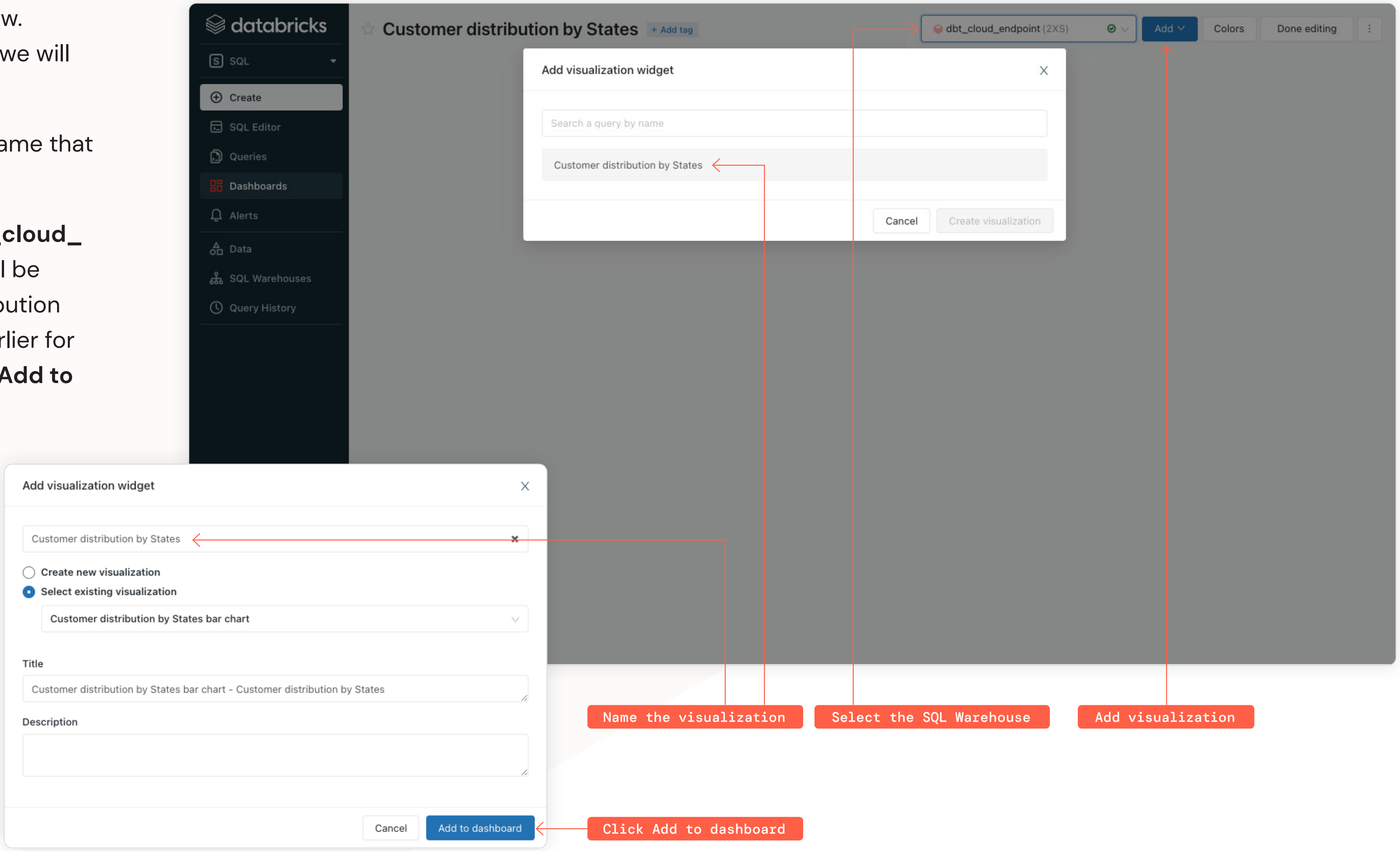
Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

Give it a name on the next pop-out window. “Customer distribution by States” is what we will use. Save it.

It creates an empty dashboard with the name that we just gave it.

On this screen, pick SQL Warehouse **dbt_cloud_endpoint**. Click **Add visualization**. You will be able to see Visualization “Customer distribution by States bar chart,” which we created earlier for the same query saved. Select it and click **Add to dashboard**.

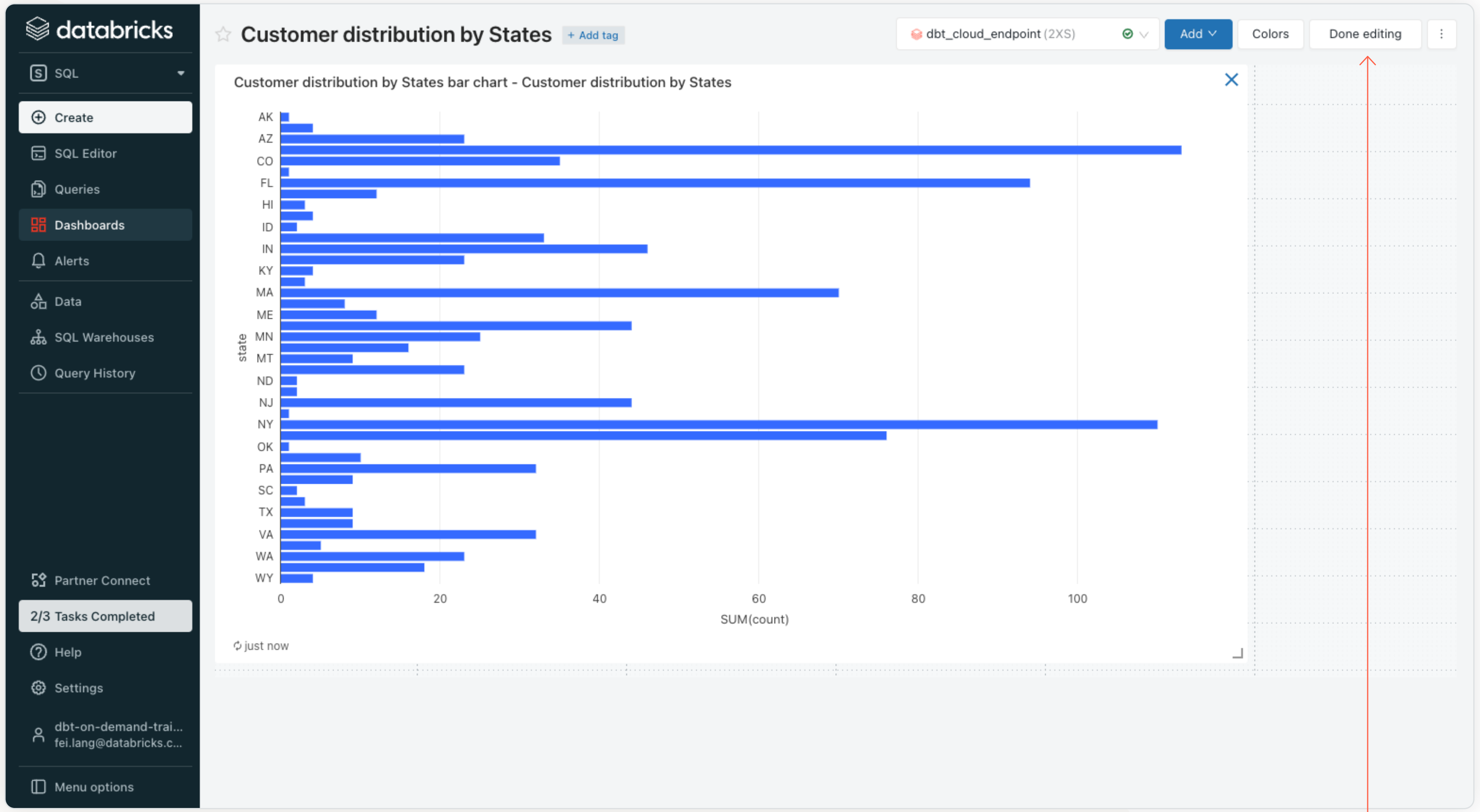


Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

A simple dashboard is created!

Click **Done editing**.

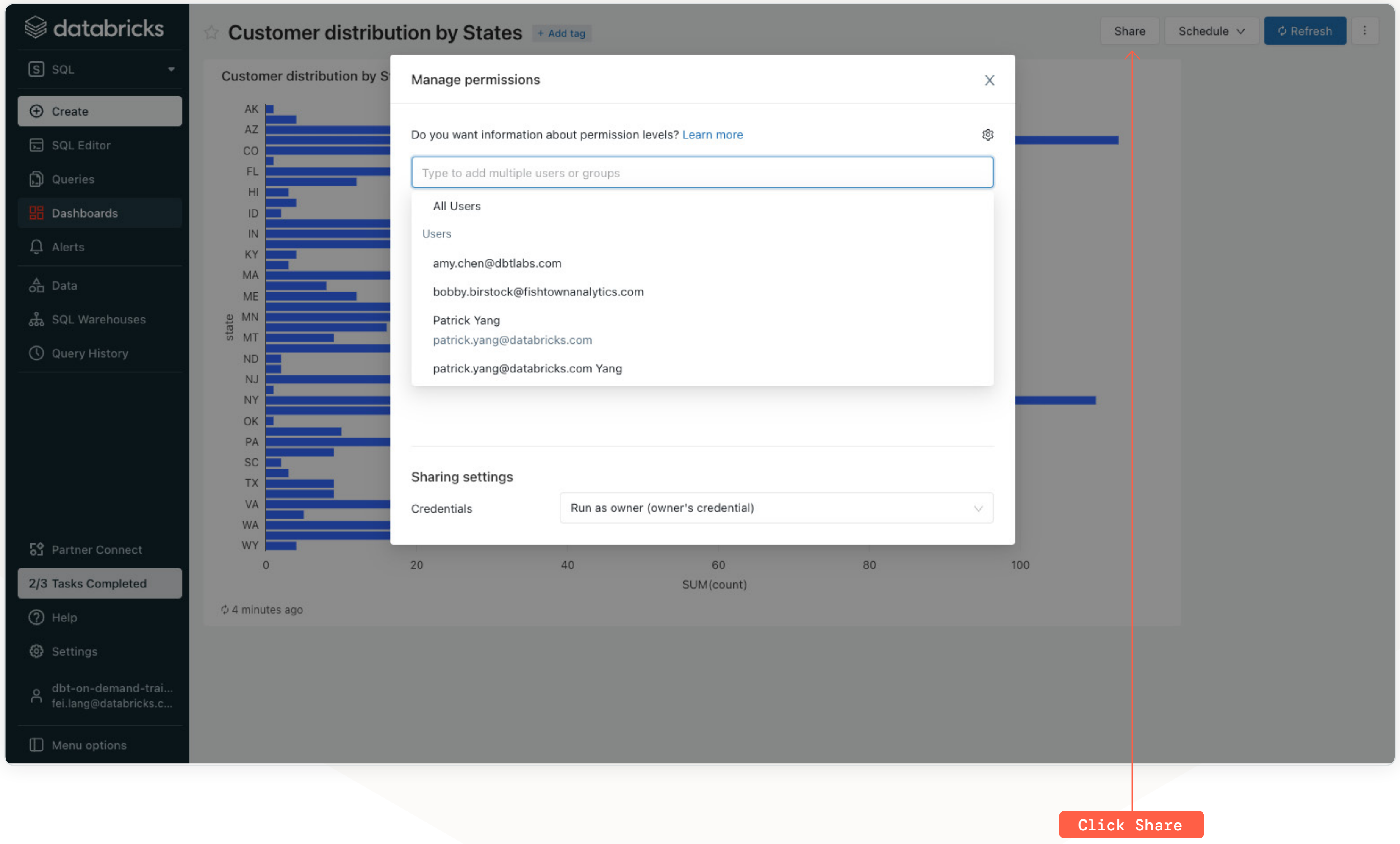


Click Done editing

Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

You can click **Share** to share this dashboard with someone in the same workspace with the corresponding access permissions.



Manage permissions

Do you want information about permission levels? Learn more

Type to add multiple users or groups

All Users

Users

amy.chen@dbtlabs.com

bobby.birstock@fishtownanalytics.com

Patrick Yang
patrick.yang@databricks.com

patrick.yang@databricks.com Yang

Sharing settings

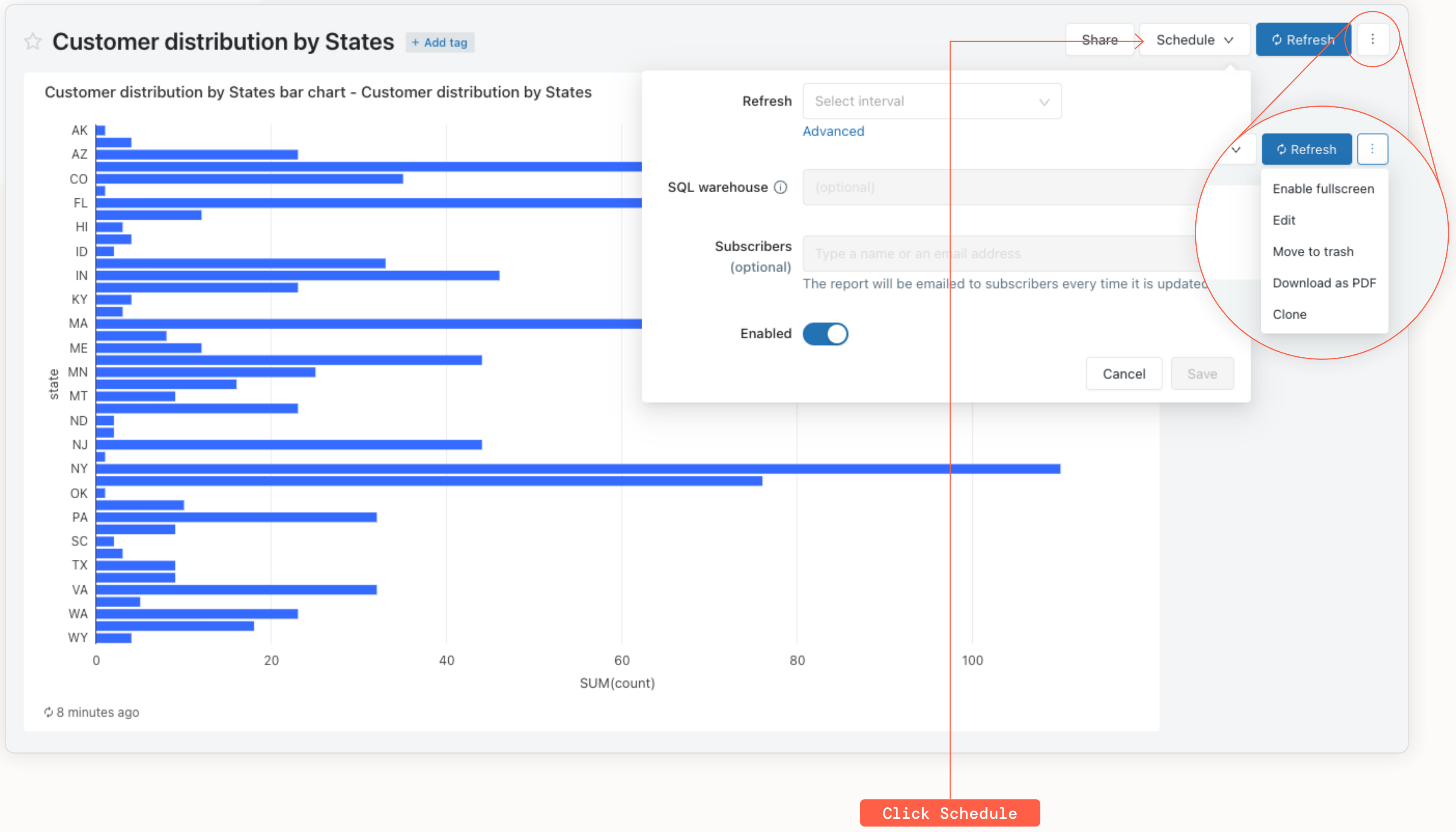
Credentials: Run as owner (owner's credential)

Part 6: Databricks SQL — Querying and Analysis

02 CONTINUED

You can also set a dashboard refresh schedule by clicking **Schedule** and entering the right settings.

And, of course, you can enable fullscreen for the dashboard, or edit, delete, clone or download it as PDF by clicking the three-dot icon next to the **Refresh** button and finding the right option.



databricks × dbt Labs

Additional Resources

dbt integration with Databricks

- Join our dbt community Slack which contains more than 18,000 data practitioners today.
  We have a dedicated Slack channel, **#db-databricks-and-spark**, for Databricks-related content.
- To continue to learn to use dbt more effectively, check out the dbt Learn site
- Contact the dbt Cloud Sales team if you're interested in exploring dbt Cloud for your team or organization

Databricks SQL

- To learn more about Databricks SQL
- On-demand hands-on workshop: Using Databricks SQL for Analytics on Your Lakehouse