# Unity Catalog Workshop Guide

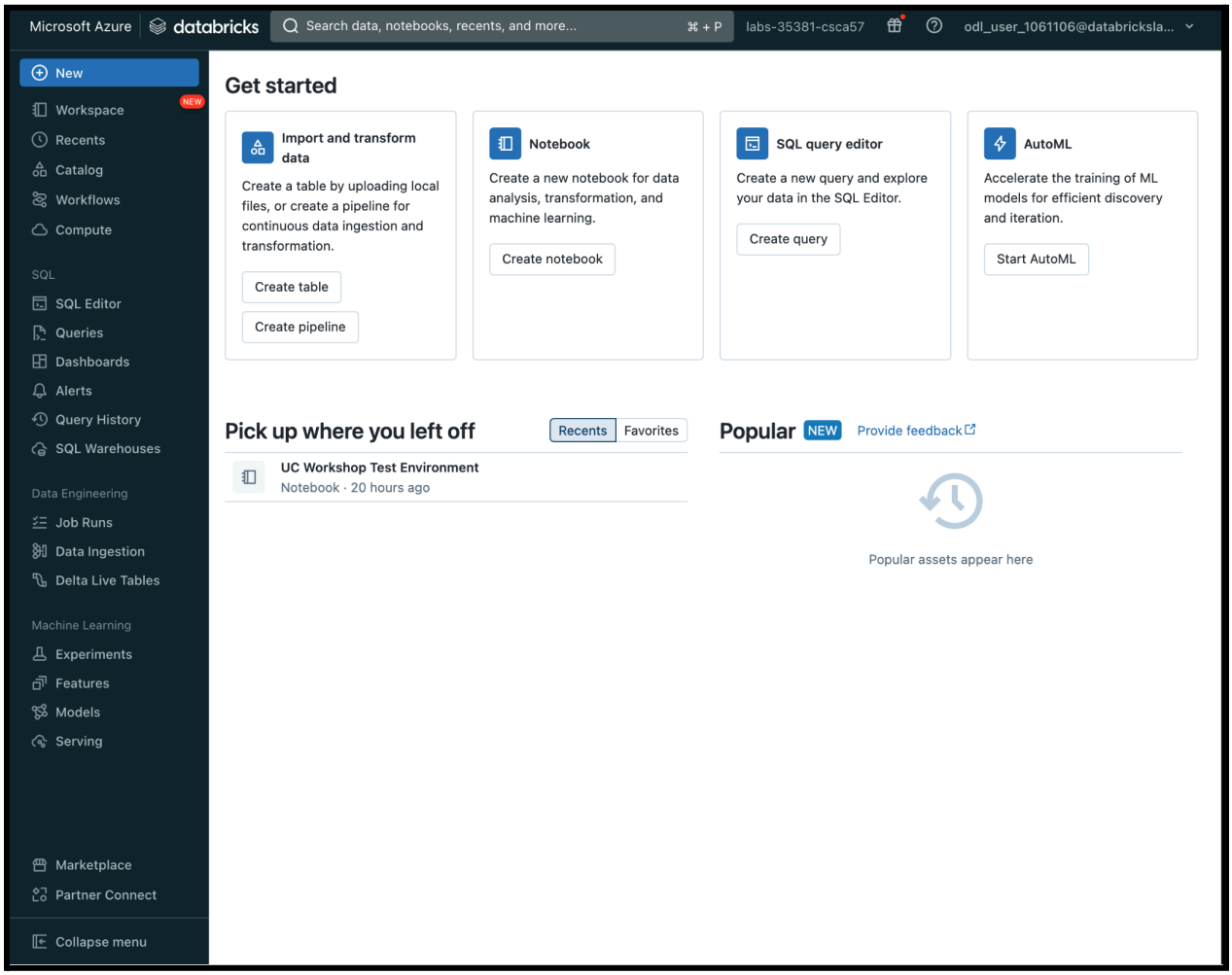
Throughout this guide, we will be working with a sample dataset that covers an online retail platform that captures customer information, customer orders and customer events. We will create a catalog, then create tables and learn how Unity Catalog allows you to securely discover, access and collaborate on trusted data and AI assets. Link to notebook: UC Workshop Lab_V1.dbc

## Table of Contents


Navigating the Data Explorer 1
Creating a Catalog 2
Creating a Schema 7
Creating Tables 11
Lineage 21
Unified Search 26
Creating Volumes 28
Querying the Data using Databricks SQL 32
Row Filters and Column Level Security 43
System Tables 43
Lakehouse Federation 43
Feature Engineering in Unity Catalog 44
Lakehouse Monitoring 44


## Navigating the Catalog Explorer

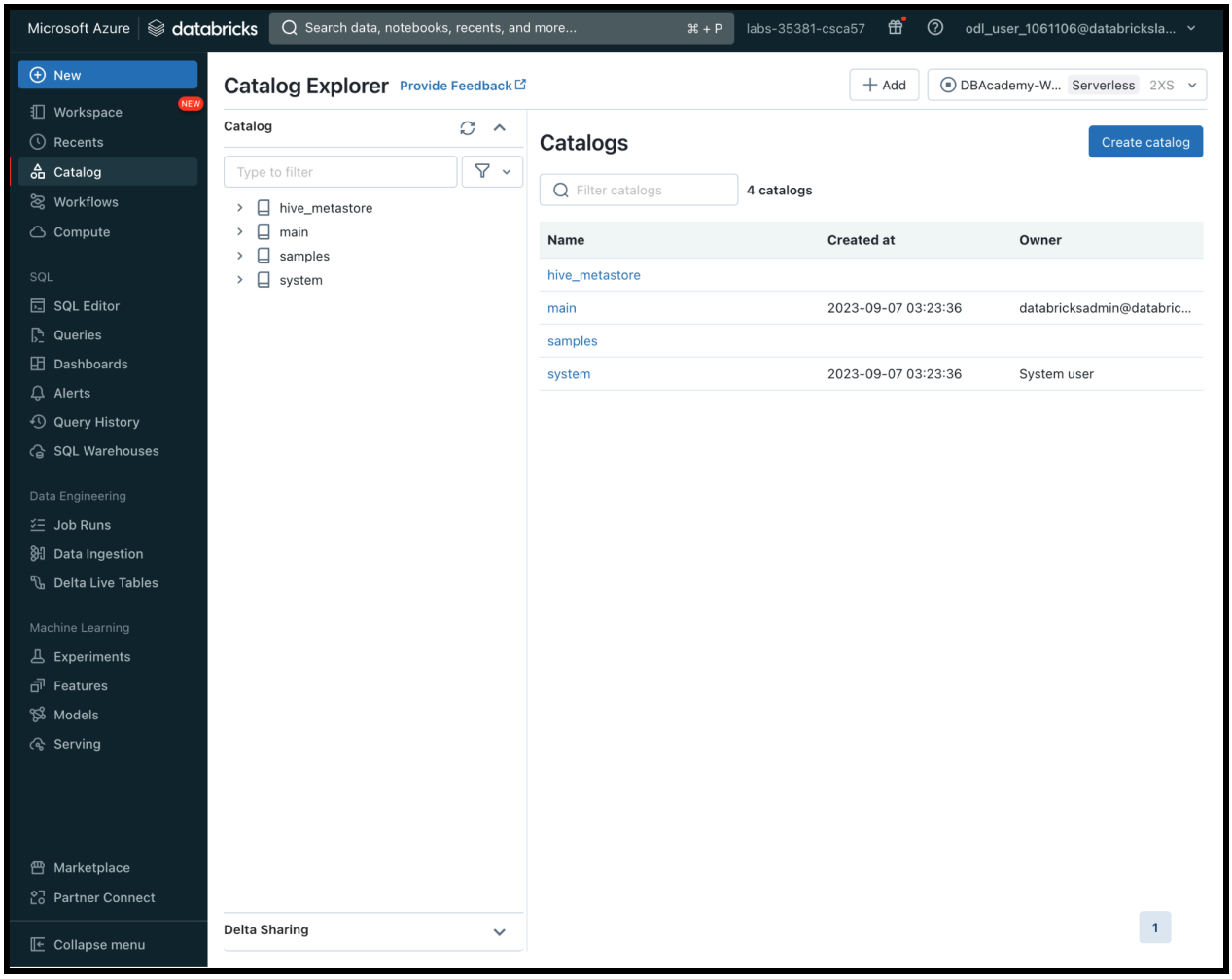
Once you are logged into your workspace, Databricks presents a landing page with unified navigation




In the navigation bar on the left you will see a section titled “Catalog”. This is where we will explore and manage catalogs, schemas (databases), tables, and permissions. Catalog is the main UI for the Unity Catalog object model. Here, you can view schema details, preview sample data, and see table details and their lineage.

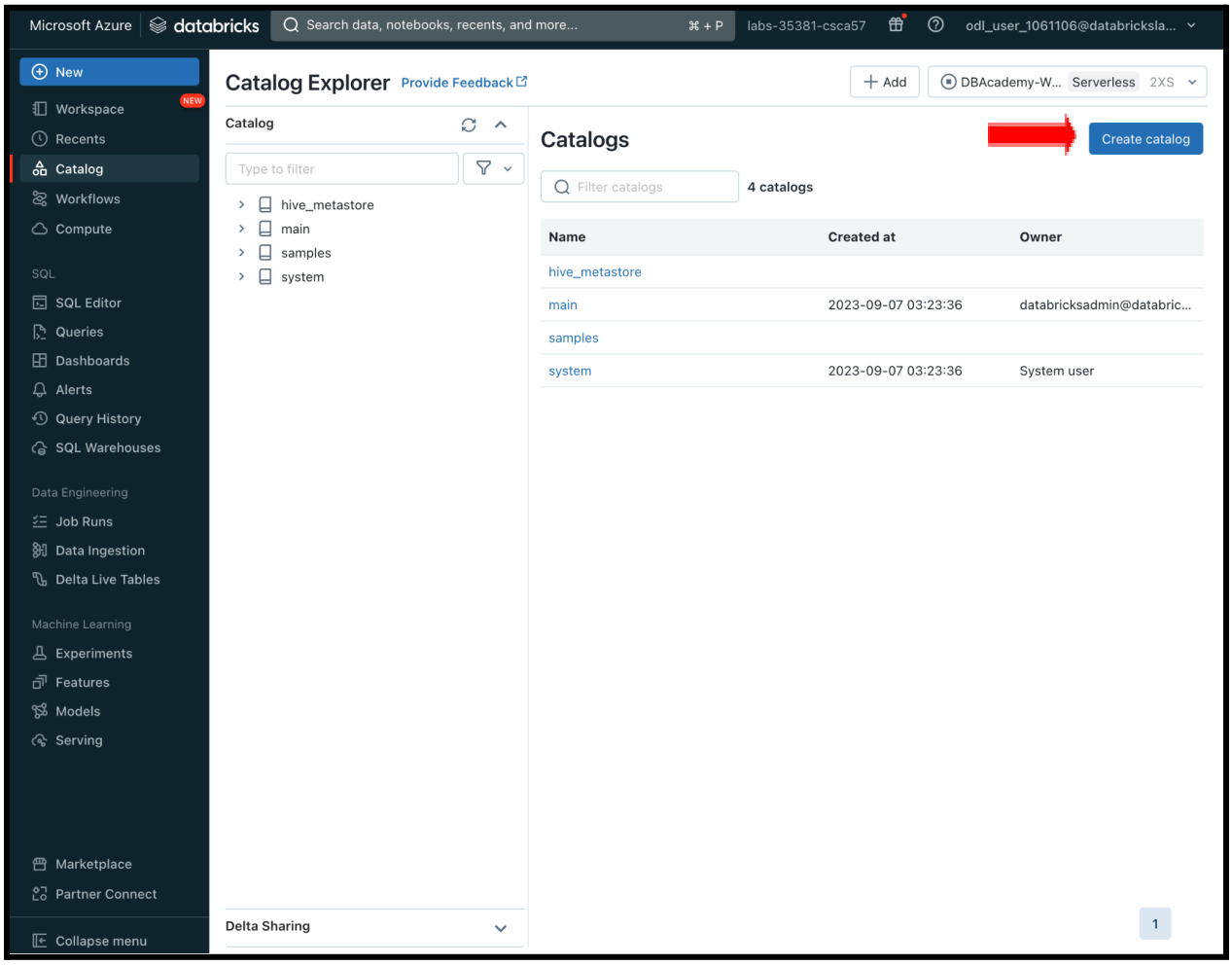
## Creating a Catalog

A catalog is the first layer of the Unity Catalog hierarchy and is used to organize your schemas and subsequently your tables and/or views.

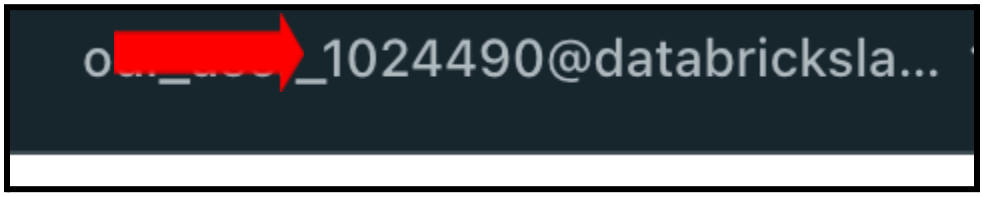


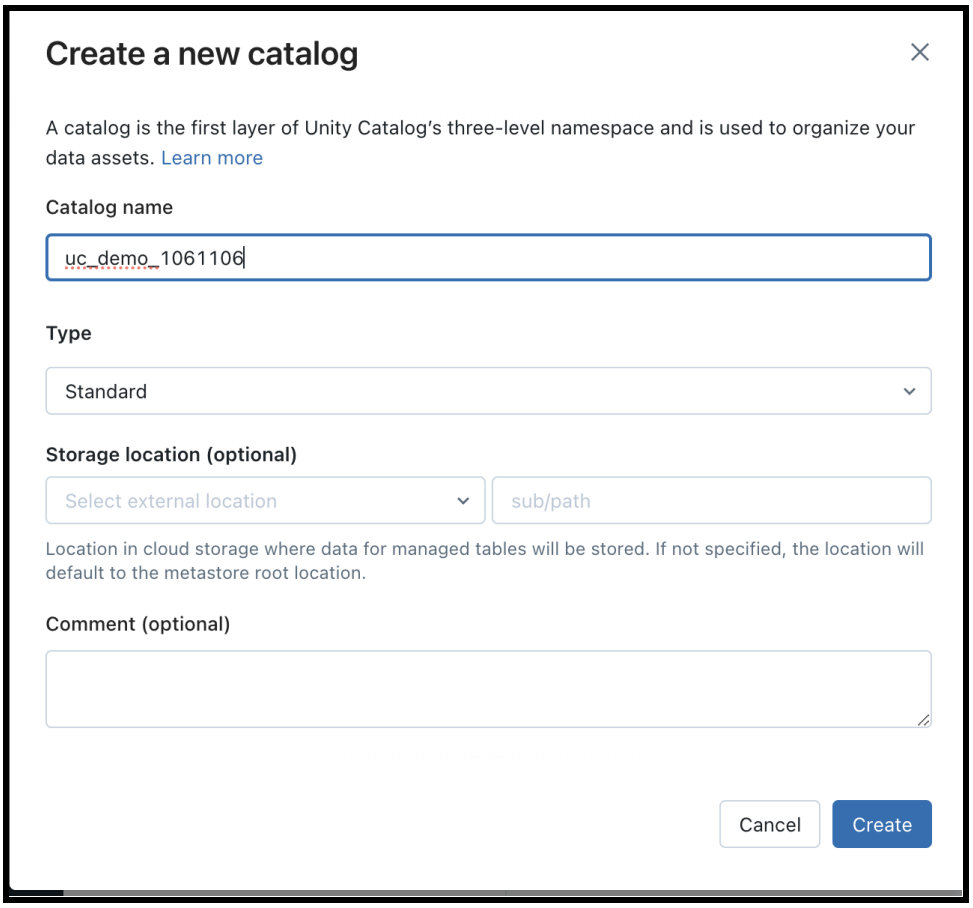

You can create a catalog through the Catalog Explorer UI or a SQL command.

To create a catalog via the Catalog Explorer, click the “Create catalog” button in the upper right corner.




Then, give the catalog a name "uc_demo_XXXXXXX" **Note: the XXXXXXX come from your odl_user_XXXXXXX in the top right corner of your workspace**

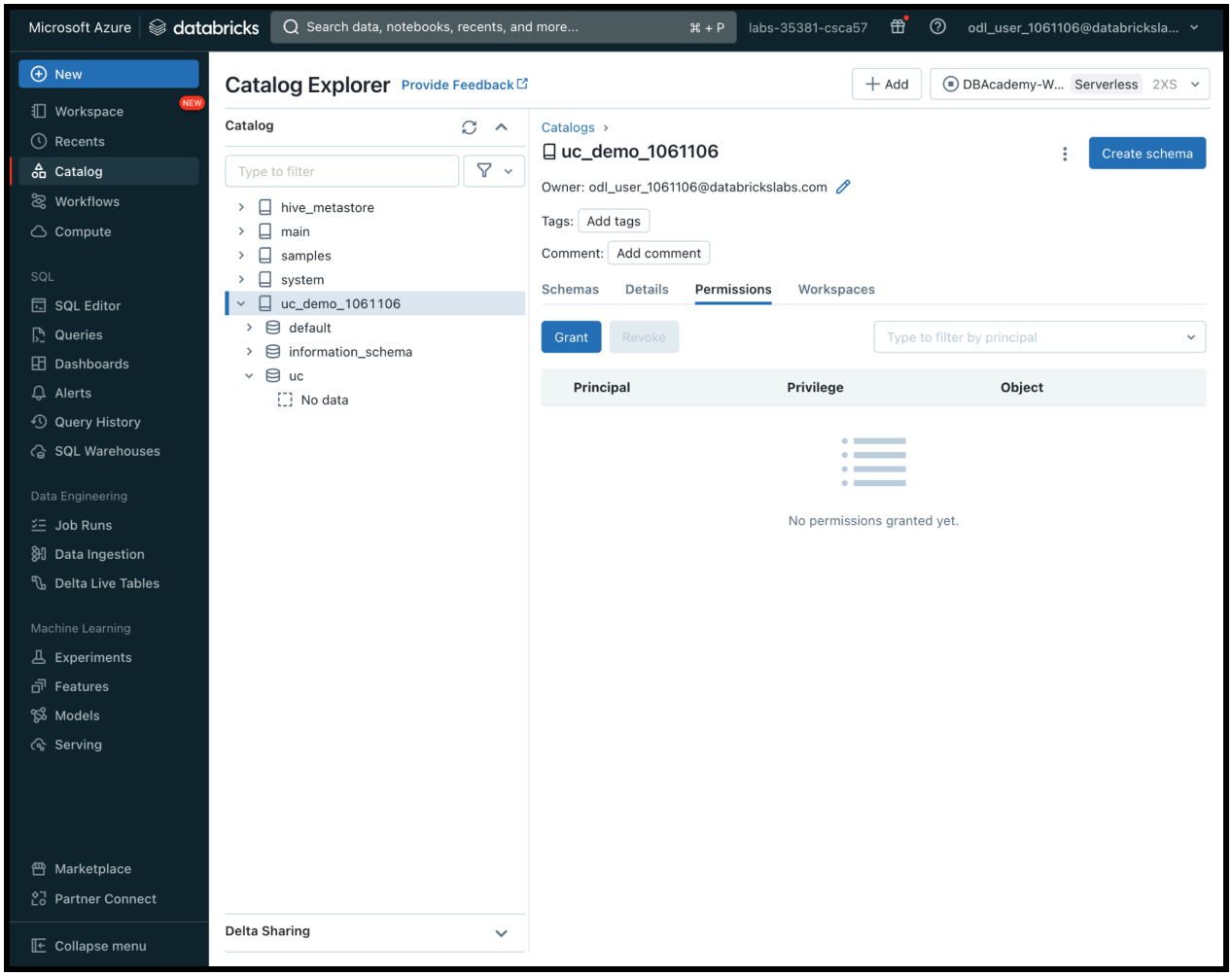





For this workshop the Type of catalog will be Standard meaning that the data will be coming from cloud storage. If you want your managed data stored in a specific external location, it can be optionally added, as well as any comments regarding the catalog.

You will now be able to see your newly created catalog. Let's grant access to this catalog by clicking on the Permissions tab.

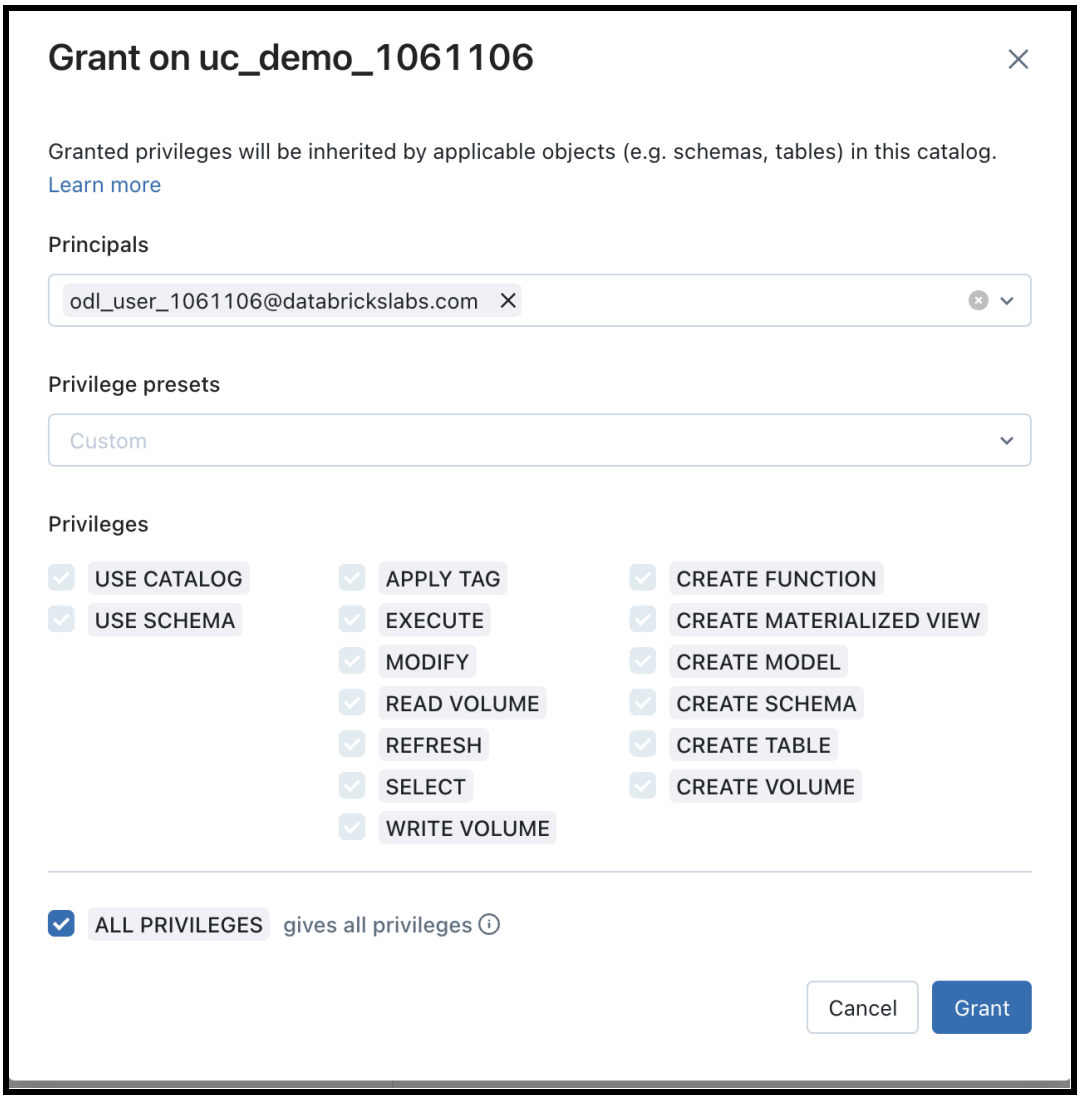
Notice the Owner field below the catalog name. Owners have all privileges on the catalog. You can set the Owner to a user, group, or service principal. Let's leave it set to your user id.

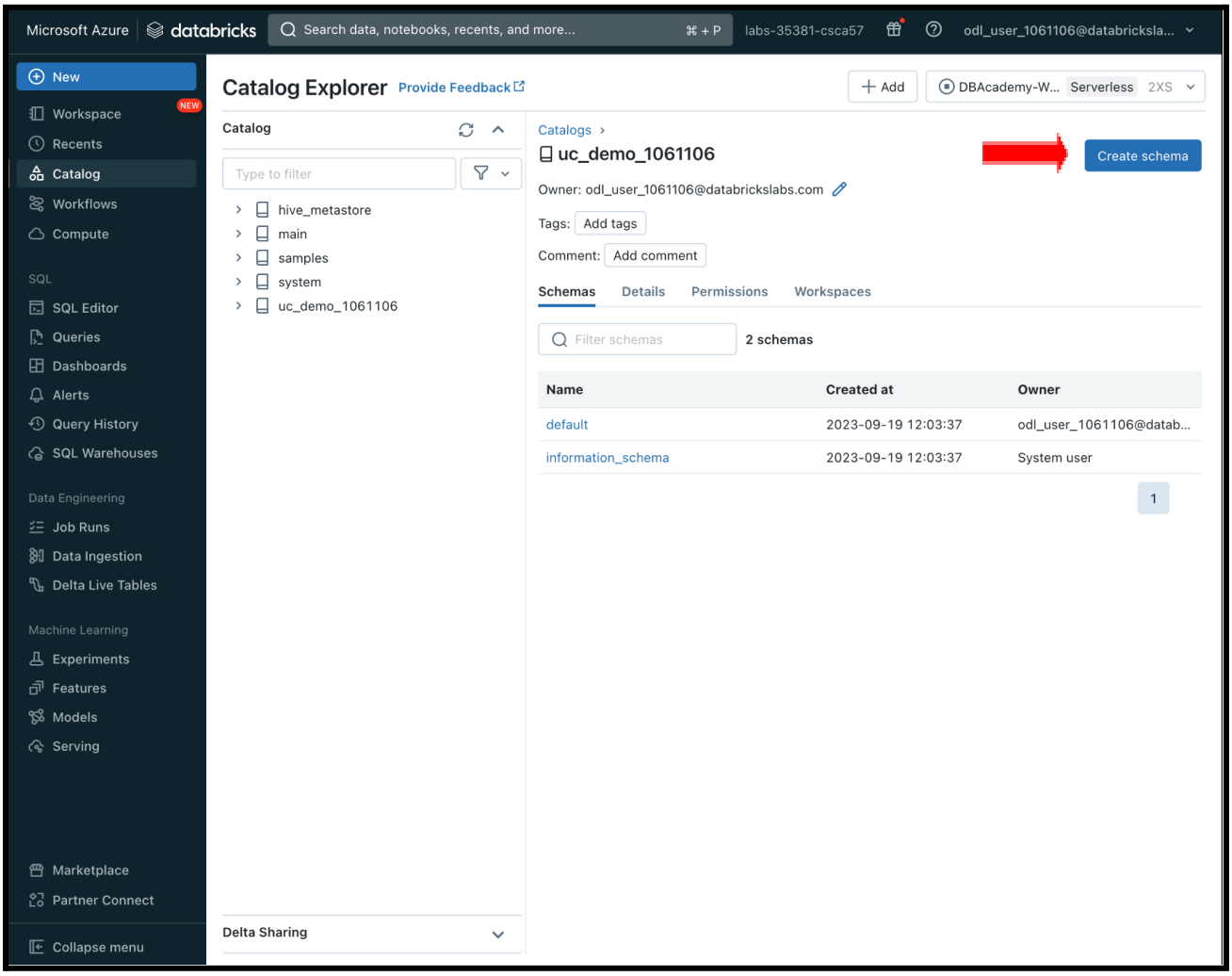



Click “Grant” and you will be able to give other users or groups privileges on your catalog. For this workshop, please type in analysts and give them all privileges.

You now have all privileges on the catalog that will be inherited by all applicable objects such as schemas, tables and views in this catalog

You can also choose one of the typical privilege presets which make it easier to assign the required privileges for users with a data reader or data writer role.

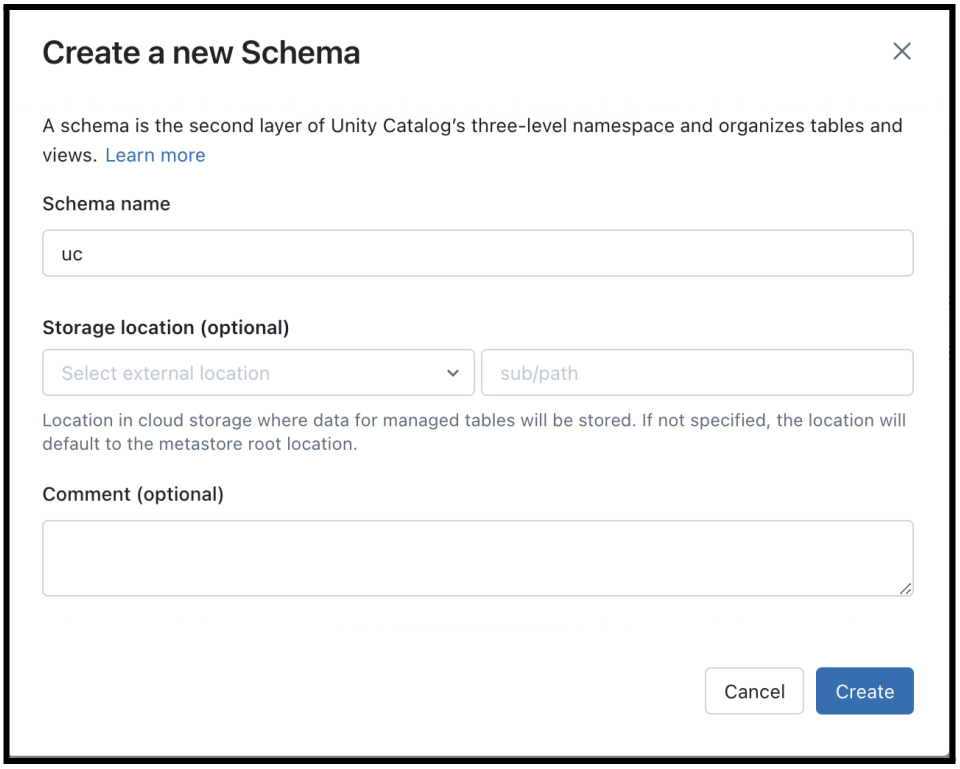

Grant on uc_demo_1061106
Granted privileges will be inherited by applicable objects (e.g. schemas, tables) in this catalog.
Learn more
Principals
odl_user_1061106@databrickslabs.com
Privilege presets
Custom
Privileges
USE CATALOG
USE SCHEMA
APPLY TAG
EXECUTE
MODIFY
READ VOLUME
REFRESH
SELECT
WRITE VOLUME
CREATE FUNCTION
CREATE MATERIALIZED VIEW
CREATE MODEL
CREATE SCHEMA
CREATE TABLE
CREATE VOLUME
ALL PRIVILEGES
gives all privileges
Cancel
Grant




Give the schema the name “uc”. Similarly you can store your data in an external location if you wish and leave comments regarding the schema as well.

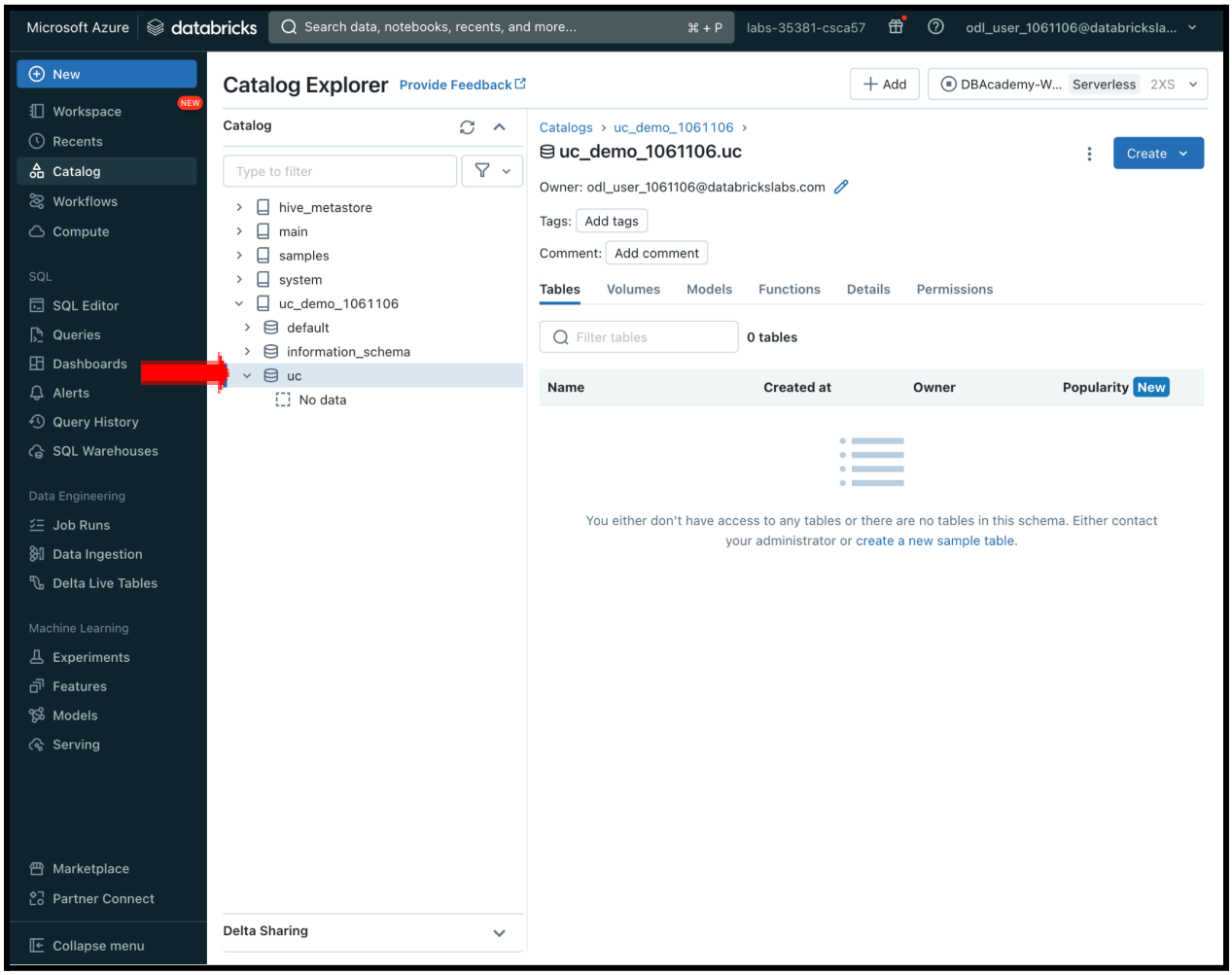
Note that when you create a managed table, Unity Catalog will create the Delta files in the most specific cloud storage location found:

1. Beginning with the path specified on the LOCATION keyword in the CREATE TABLE statment, if provided, or
2. The location specified on the Schema (or database), if specified, or
3. The location specified on the Catalog, if specified, or
4. In the Unity Catalog metastore’s default storage location.



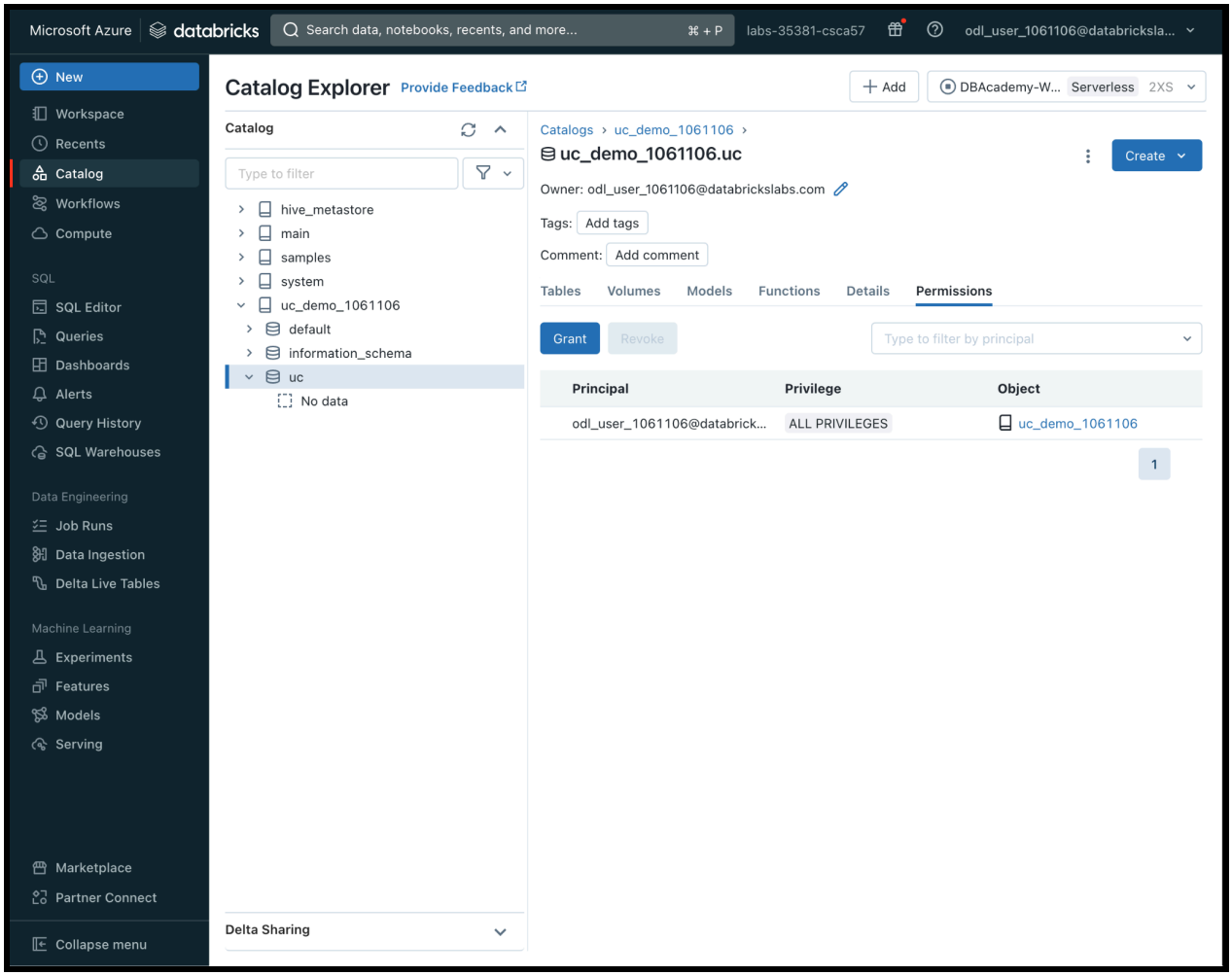

i

You will now see your new schema in the catalog.




If you click the “Permissions” tab, you will be able to see that the permissions and privileges you set at the Catalog level have trickled down to the schema level as well; however you are more than welcome to grant or revoke additional privileges to users or groups of users at the schema level as well.

You may also grant privileges and permissions using SQL commands. For more information on how to do this, click here

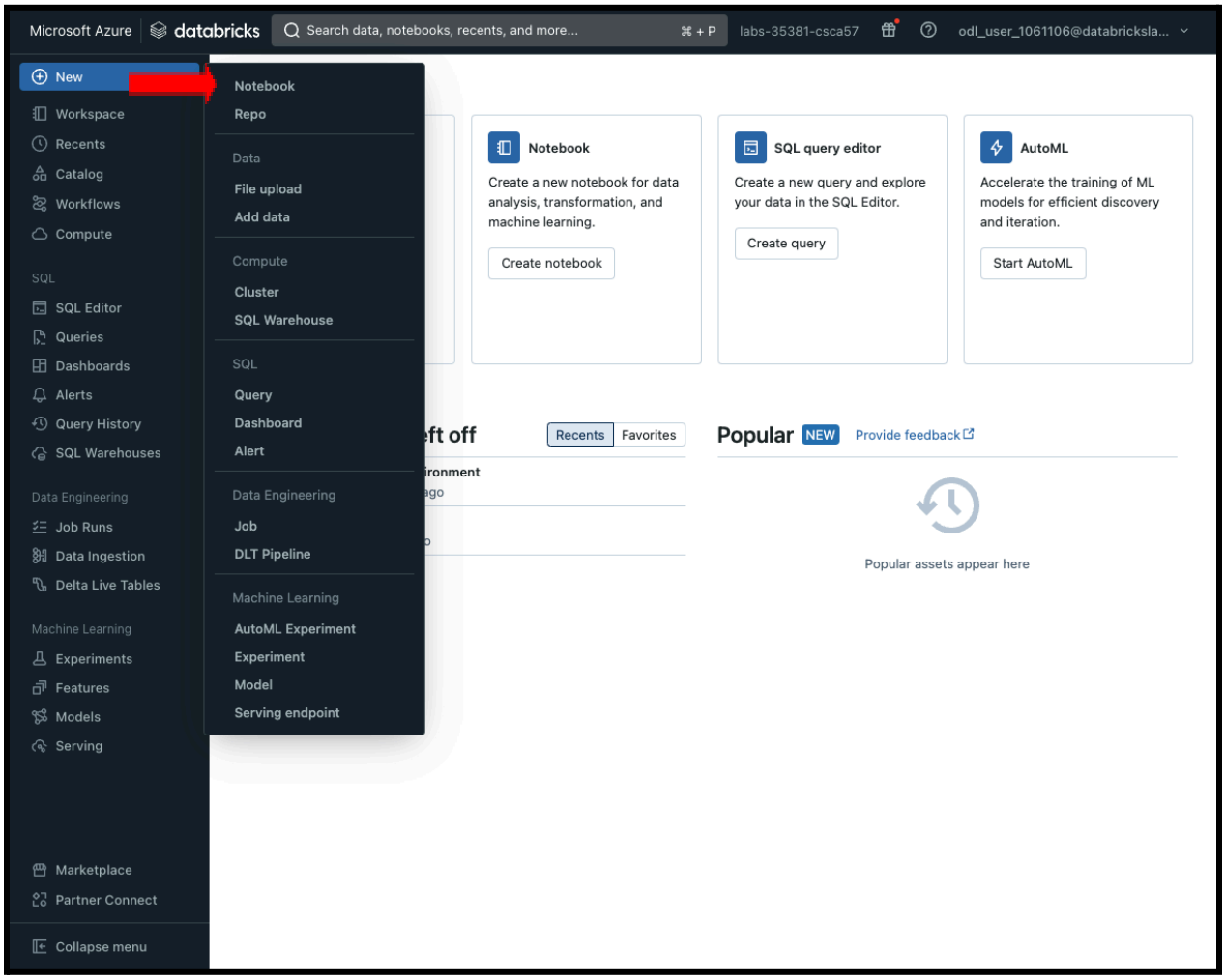



Now let us add some data by creating tables

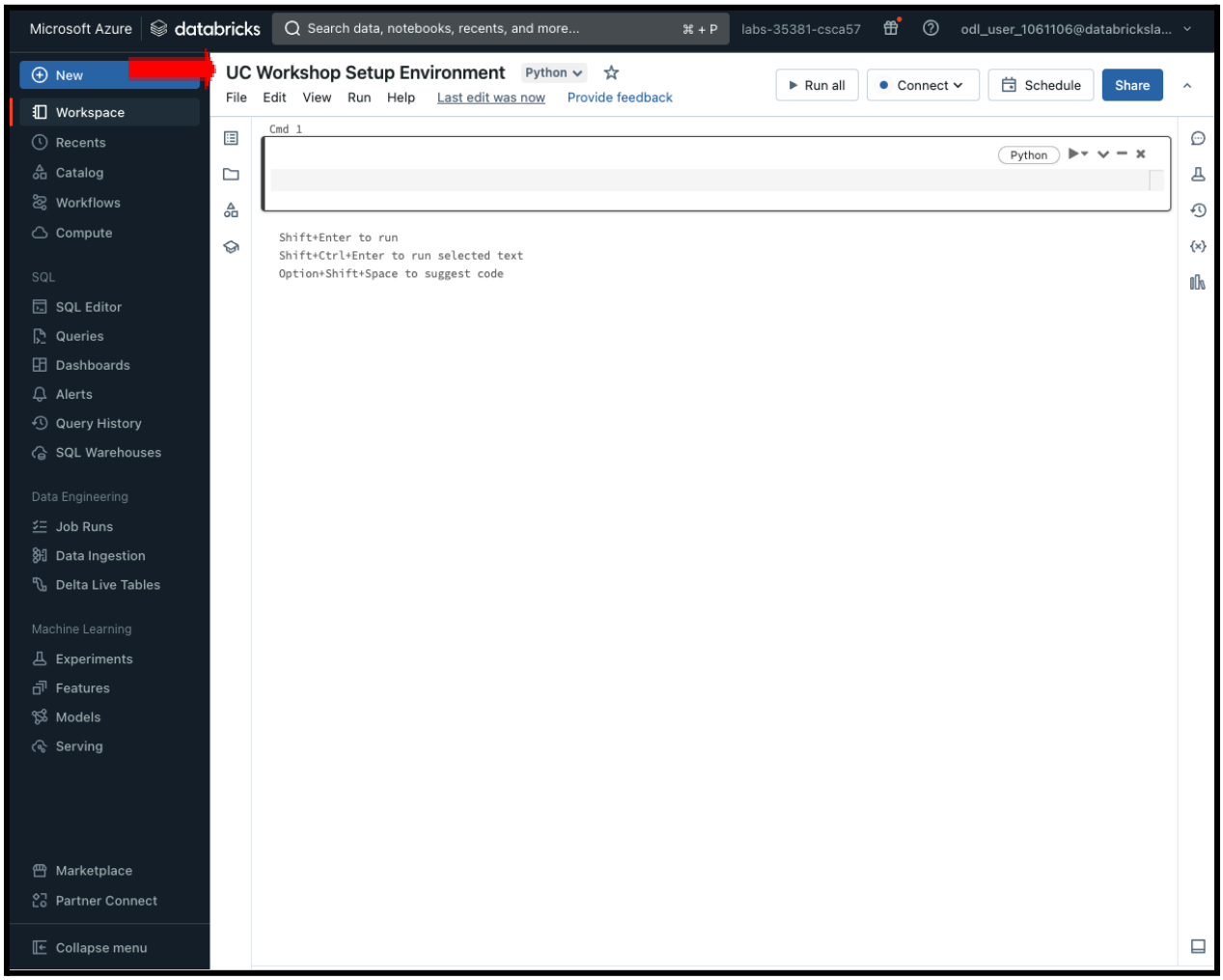
# Creating Tables

For the purpose of this workshop we are going to create our tables using our retail databricks dataset that is already housed in the databricks filesystem. For future purposes you can ingest data using autoloader, copy into, the add data button and volumes.

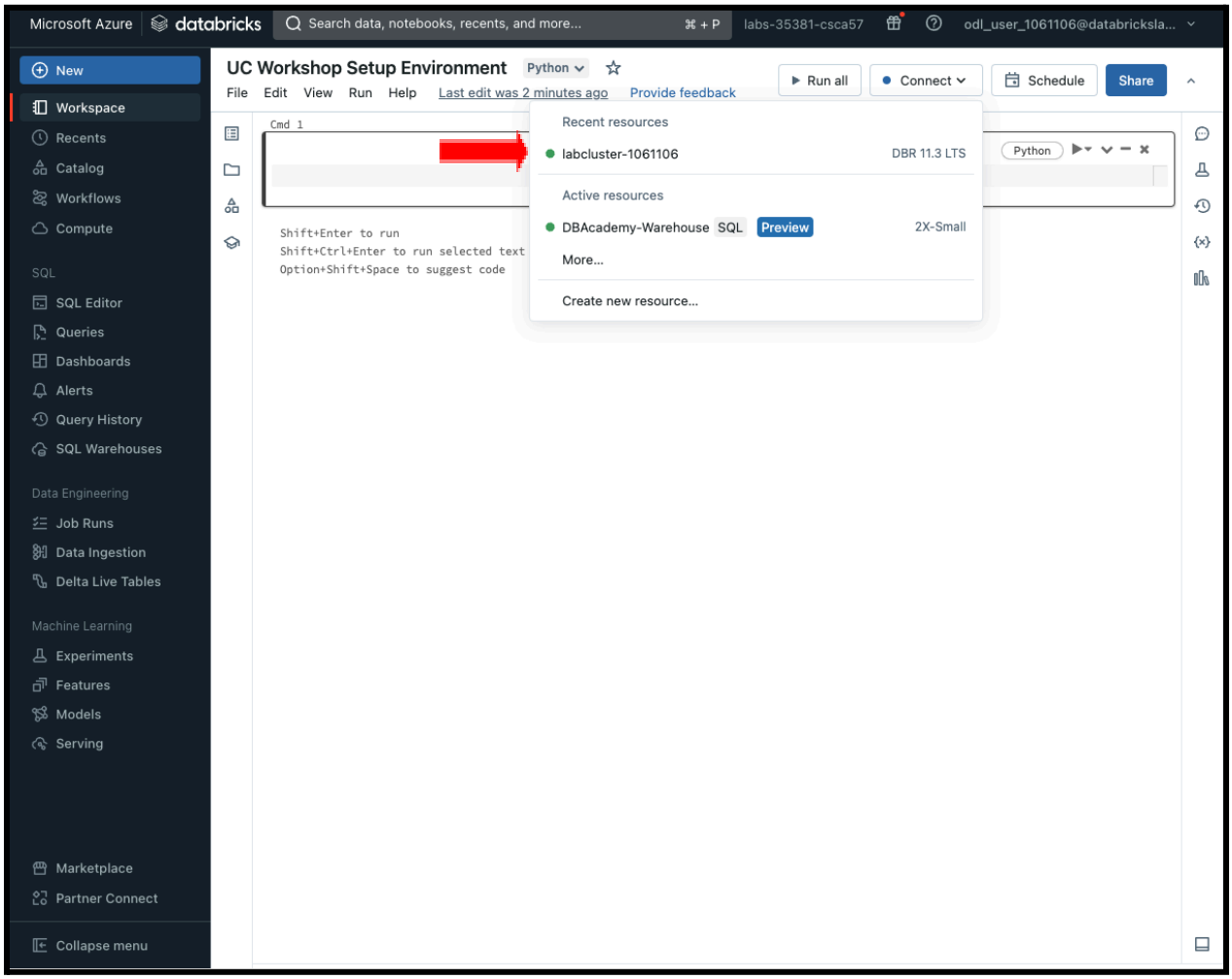
To do this, in the left hand navigation pane click “New” and then “Notebook”



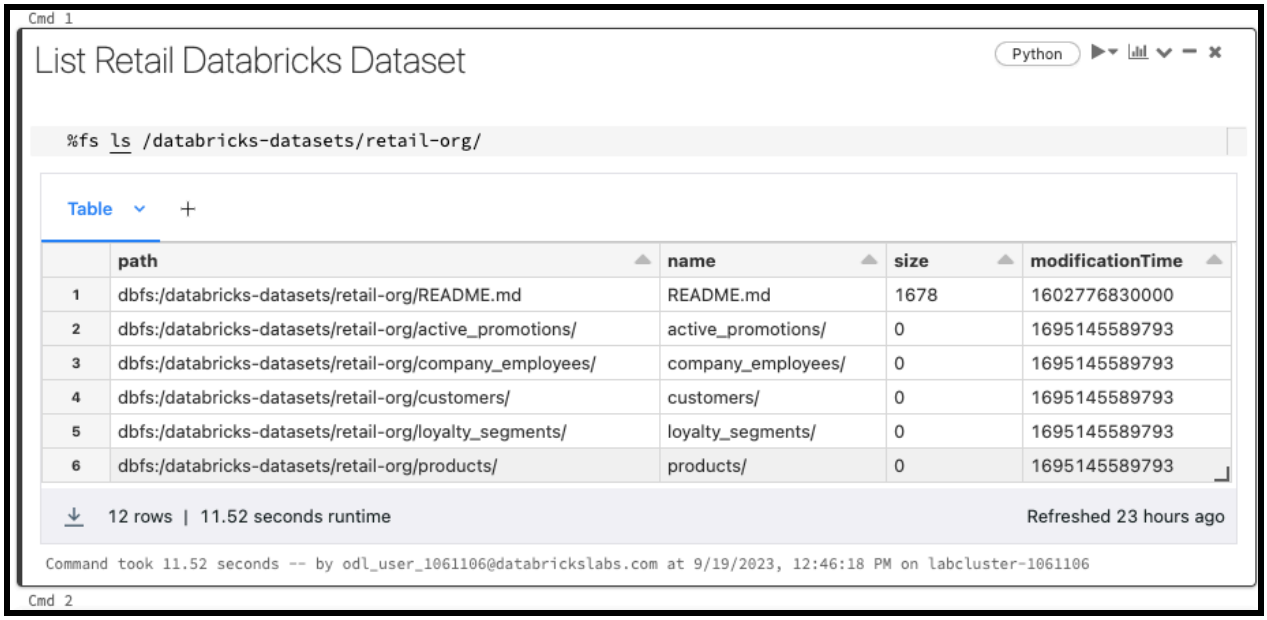

This will open up a blank notebook; so change the title of your notebook for quick reference




and then connect to your cluster that has already been created for you




```
%fs ls /databricks-datasets/retail-org/
```




```
user_id =
dbutils.notebook.entry_point.getDbutils().notebook().getContext().userName().g
et()
user_id = ''.join(filter(str.isdigit, user_id))
print(user_id)
```


```
f = open('/dbfs/databricks-datasets/retail-org/README.md', 'r')
print(f.read())
```

Cmd 2

## Open Read Me file to understand dataset

Python

```
f = open('/dbfs/databricks-datasets/retail-org/README.md', 'r')
print(f.read())
```

```
Synthetic Retail Dataset
================================
This dataset is a collection of files representing different dimensions and facts for a retail organization.

Provenance
================================
This dataset was generated by Databricks.

Data Set Information
====================
* Sales Orders: **sales_orders/sales_orders.json** records the customers' originating purchase order.
* Purchase Orders: **purchase_orders/purchase_orders.xml** contains the raw materials that are being purchased.
* Products: **products/products.csv** contains products that the company sells.
* Goods Receipt: **goods_receipt/goods_receipt.parquet** contains the arrival time of purchased orders.
* Customers: **customers/customers.csv** contains those customers who are located in the US and are buying the
finished products.
* Suppliers: **suppliers/suppliers.csv** contains suppliers that provide raw materials in the US.
* Sales Stream: **sales_stream/sales_stream.json/** is a folder containing JSON files for streaming purposes.
* Promotions: **promotions/promotions.csv** contains additional benefits on top of normal purchases.
* Active Promotions: **active_promotions/active_promotions.parquet** shows how customers are progressing toward
s becoming eligible for promotions.
```

Command took 0.33 seconds -- by odl_user_1061106@databrickslabs.com at 9/19/2023, 12:46:33 PM on labcluster-1061106

```
active_promo =
spark.read.parquet('/databricks-datasets/retail-org/active_promotions/')


active_promo.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{us
er_id}.uc.active_promotions_bronze")
```

Cmd 6

## Create and Ingest Active Promotions bronze table

```
active_promo = spark.read.parquet('/databricks-datasets/retail-org/active_promotions/')
```

▸ (1) Spark Jobs

▸ active_promo: pyspark.sql.dataframe.DataFrame = [promo_customer: string, promo_item: string ... 9 more fields]

Command took 0.53 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:17:44 PM on labcluster-1061106

Cmd 7

```
active_promo.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{user_id}.uc.active_promotions_bronze")
```

▸ (5) Spark Jobs

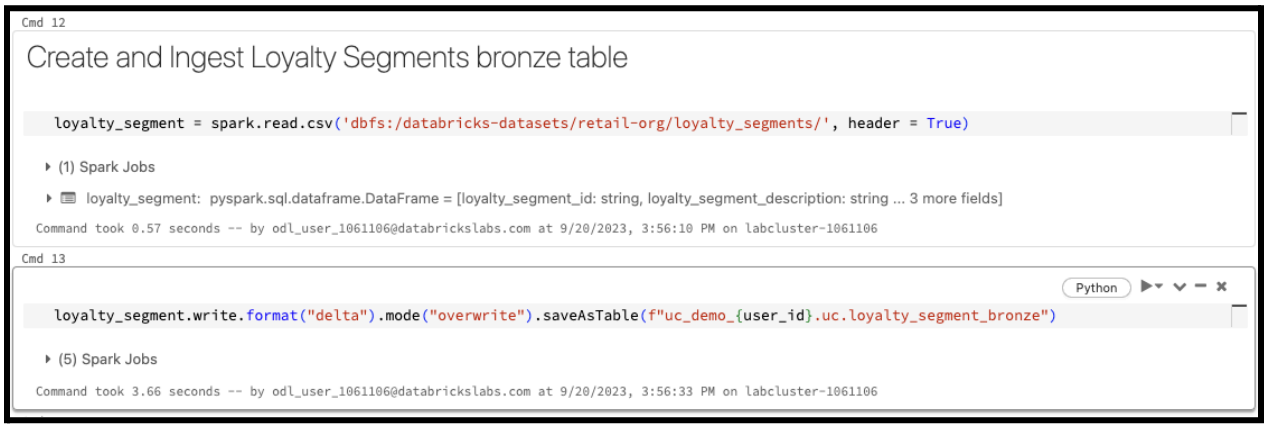
Command took 3.57 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:21:56 PM on labcluster-1061106

Cmd 8

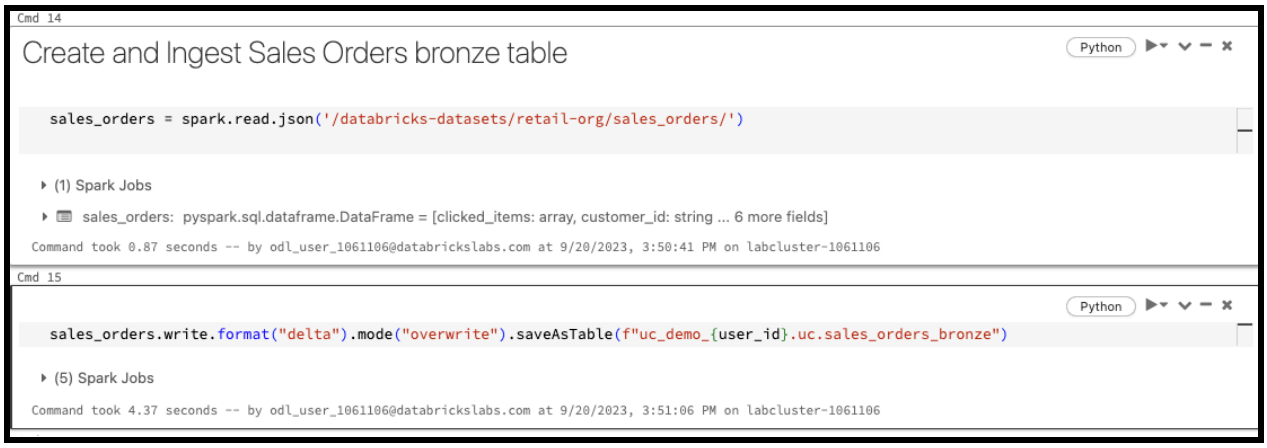
```
customers = spark.read.csv('dbfs:/databricks-datasets/retail-org/customers/', header = True)

customers.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{user_id}.uc.customers_bronze")
```


```
suppliers = spark.read.csv('dbfs:/databricks-datasets/retail-org/suppliers/', header = True)

suppliers.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{user_id}.uc.suppliers_bronze")
```

```
loyalty_segment =
spark.read.csv('dbfs:/databricks-datasets/retail-org/loyalty_segments/',
header = True)

loyalty_segment.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_
{user_id}.uc.loyalty_segment_bronze")
```

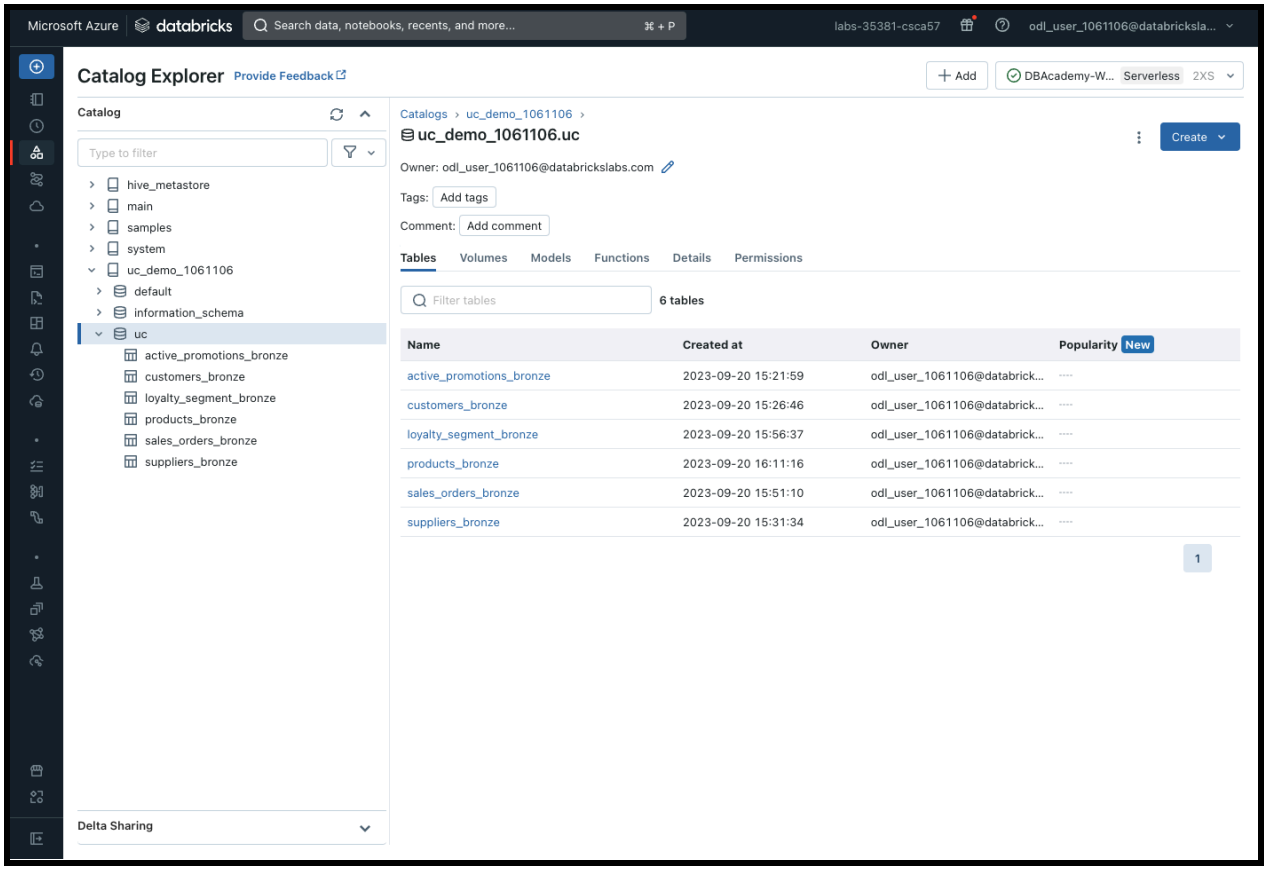



```
sales_orders =
spark.read.json('/databricks-datasets/retail-org/sales_orders/')
sales_orders.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{us
er_id}.uc.sales_orders_bronze")
```

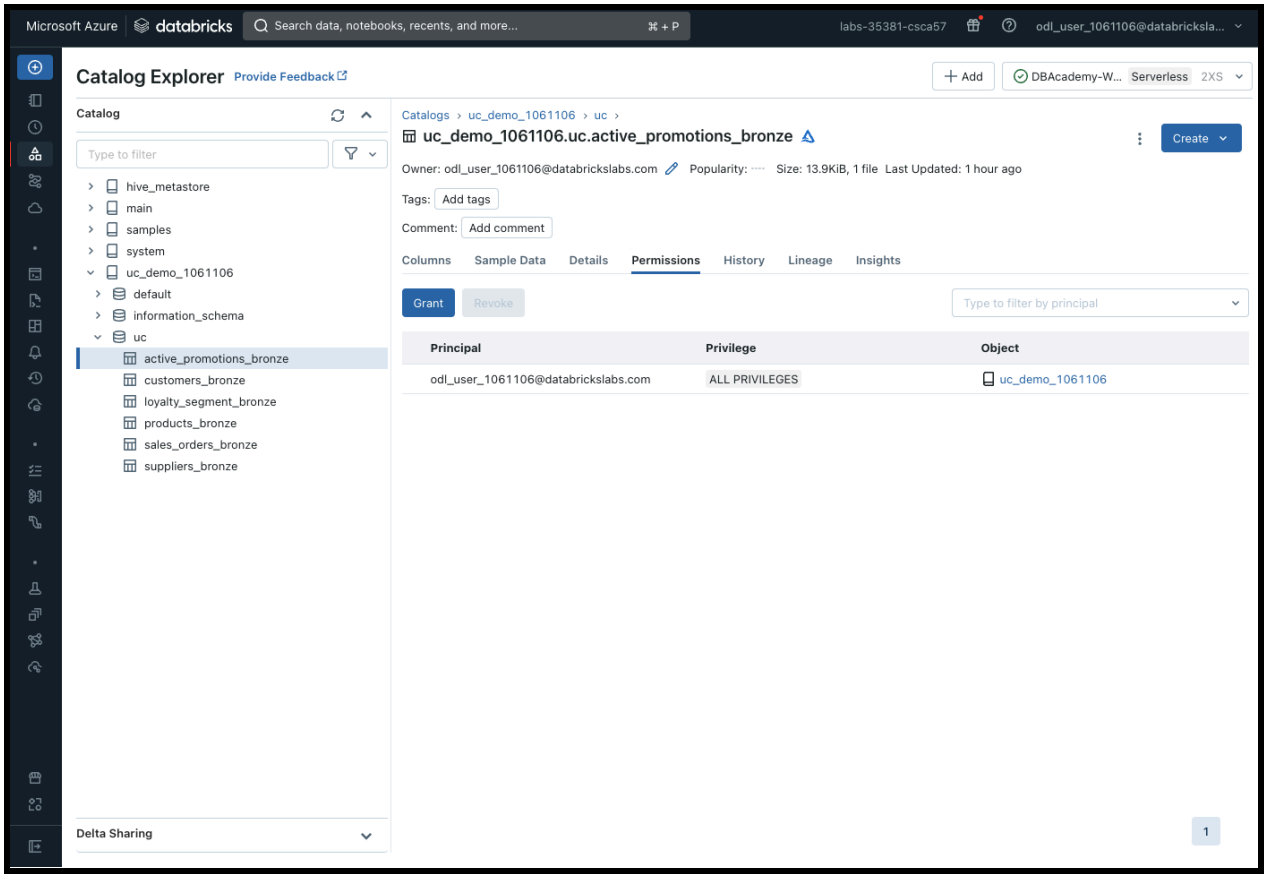



```
products = spark.read.option("delimiter",
";").csv('/databricks-datasets/retail-org/products/', header = True)

products.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{user_i
d}.uc.products_bronze")
```


Lets view our tables. Let's head back to the Catalog Explorer, locate your catalog name, your schema name and there are your tables!



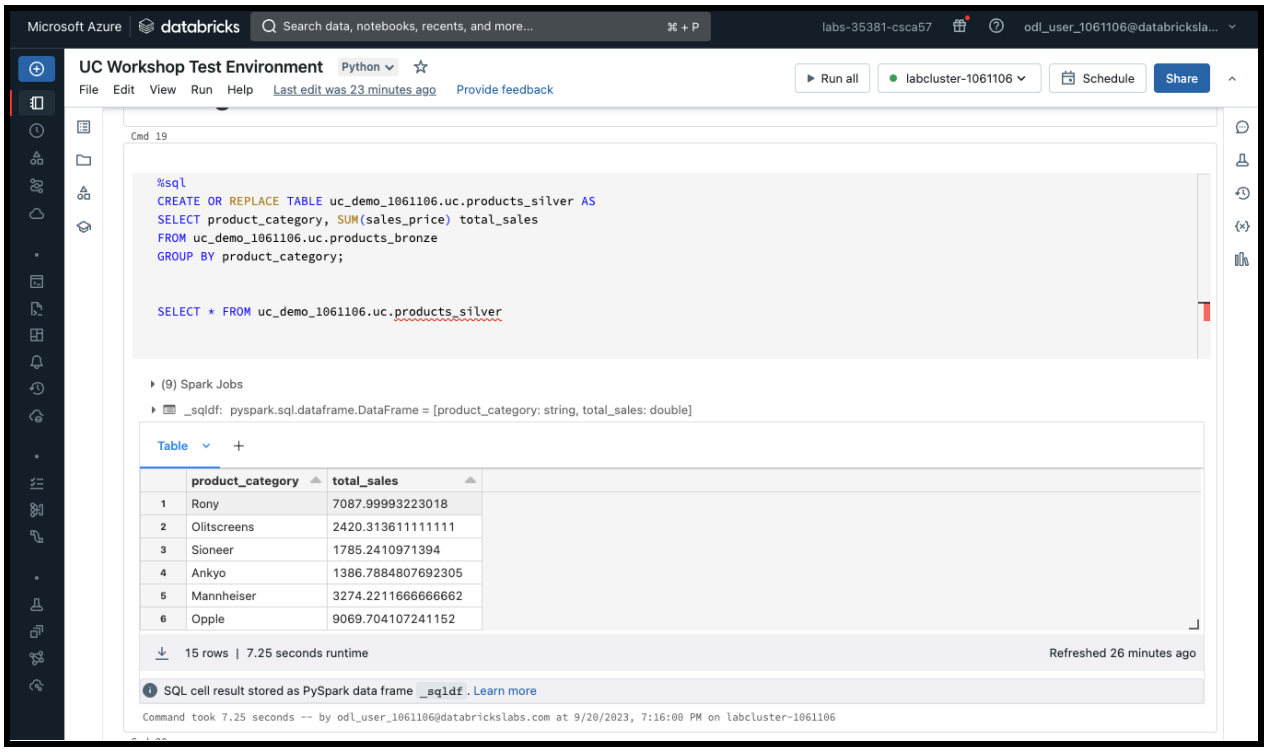

If you click the “active_promotions_bronze” and click the “Permissions” tab, you will be able to see that the permissions and privileges you set at the Catalog level have trickled down to the table level as well; however you are more than welcome to grant or revoke additional privileges to users or groups of users at the table level as well.



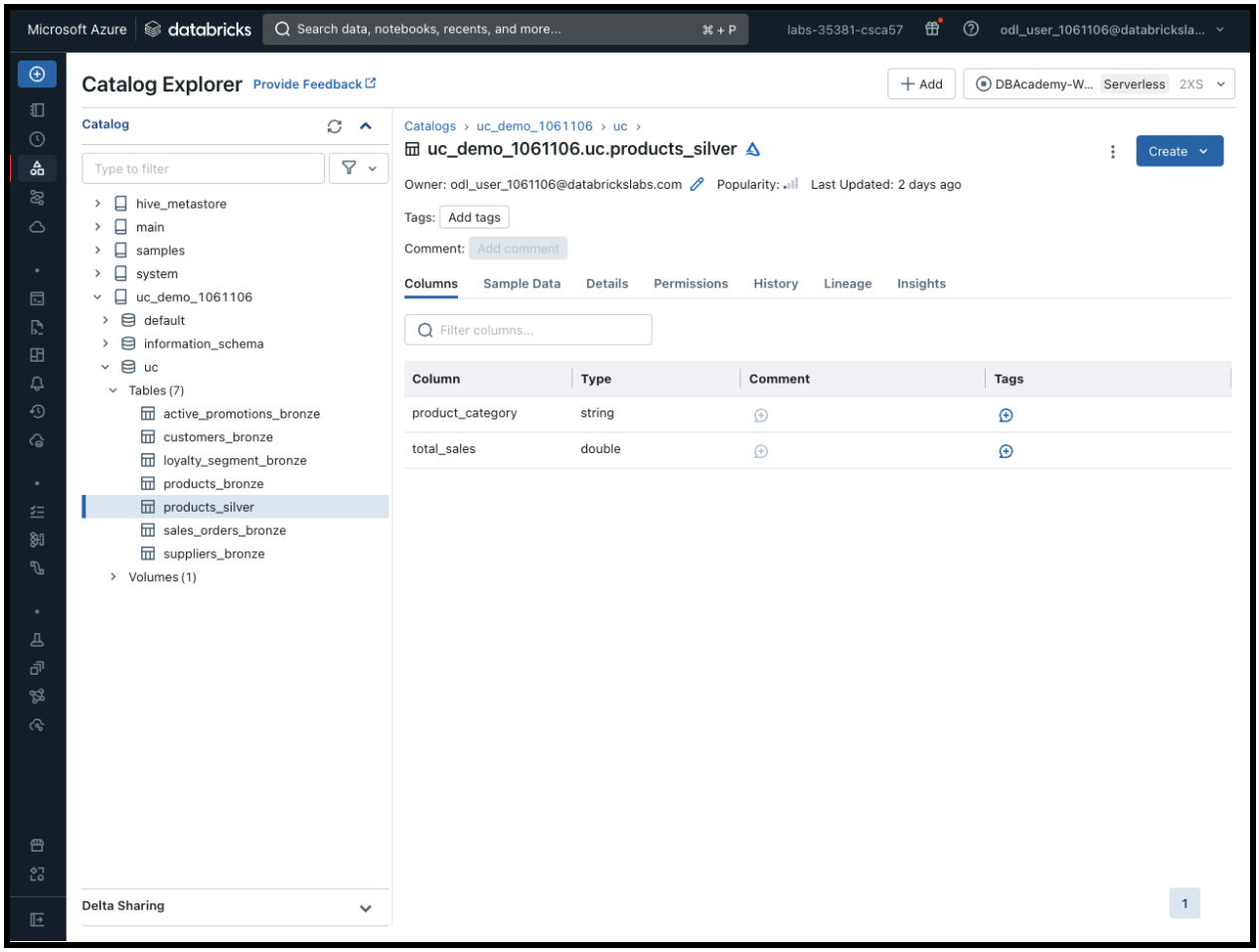

# Lineage

Now let's transform and clean up our data a bit. Back in our notebook, let's create a products table that shows the total sales price per product category

```
%sql
CREATE OR REPLACE TABLE uc_demo_xxxxxx.uc.products_silver AS
SELECT product_category, SUM(sales_price) total_sales
FROM uc_demo_xxxxxx.uc.products_bronze
GROUP BY product_category;



SELECT * FROM uc_demo_xxxxxx.uc.products_silver
```

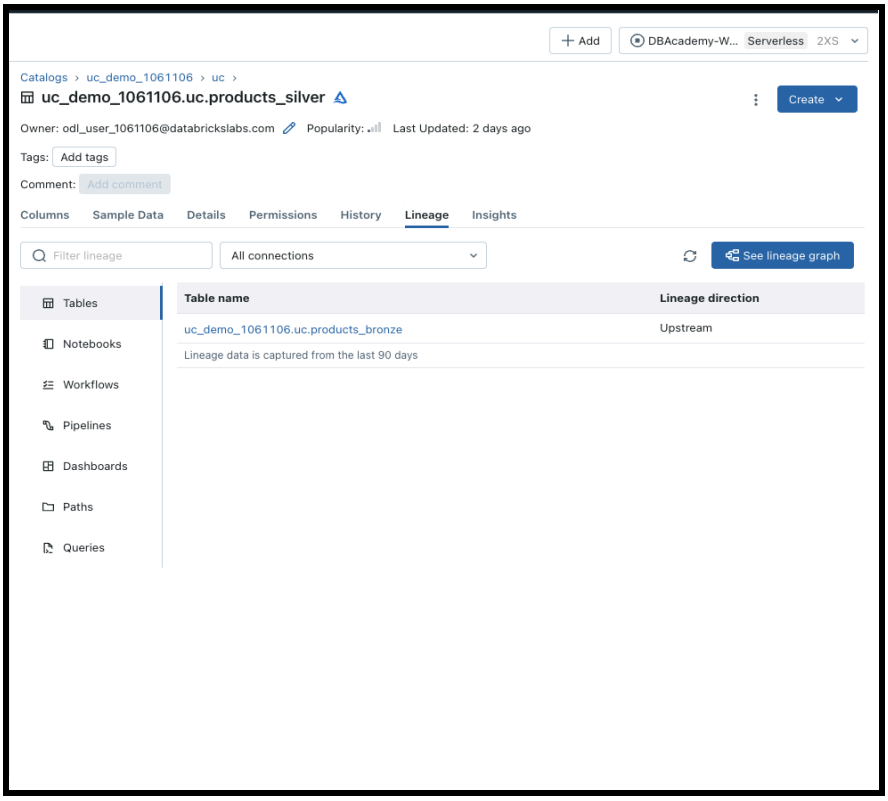



Now let us head back to the Catalog explorer and view our newly created table

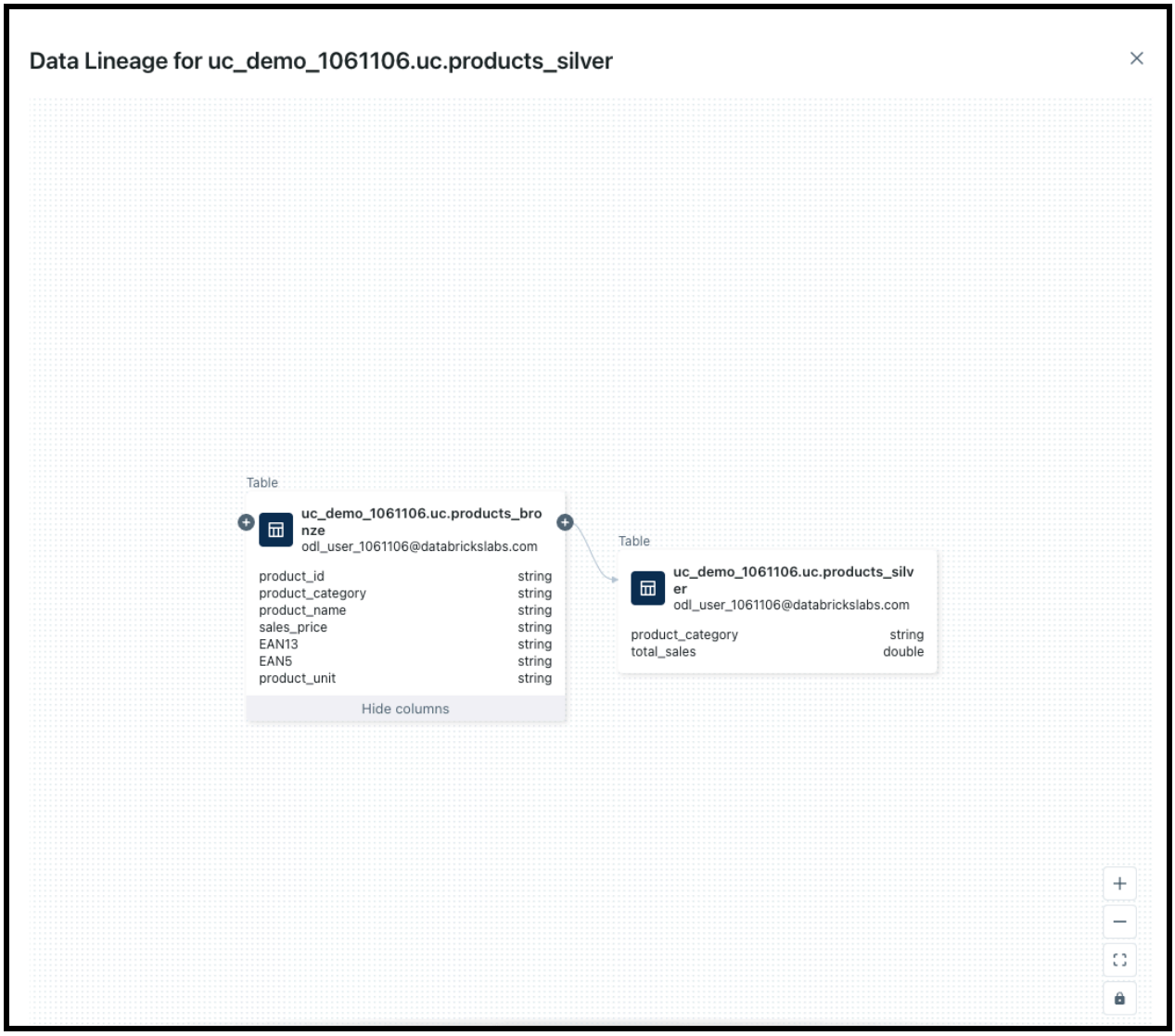



Clicking on the Lineage tab we are able to see the lineage of the table. Lineage is supported for all languages and is captured down to the column level. Lineage data includes notebooks, workflows, and dashboards related to the query. Lineage can be visualized in Catalog Explorer in near real-time, queried with SQL from the Unity Catalog System Tables (system.access.table_lineage), and retrieved with the Databricks REST API.

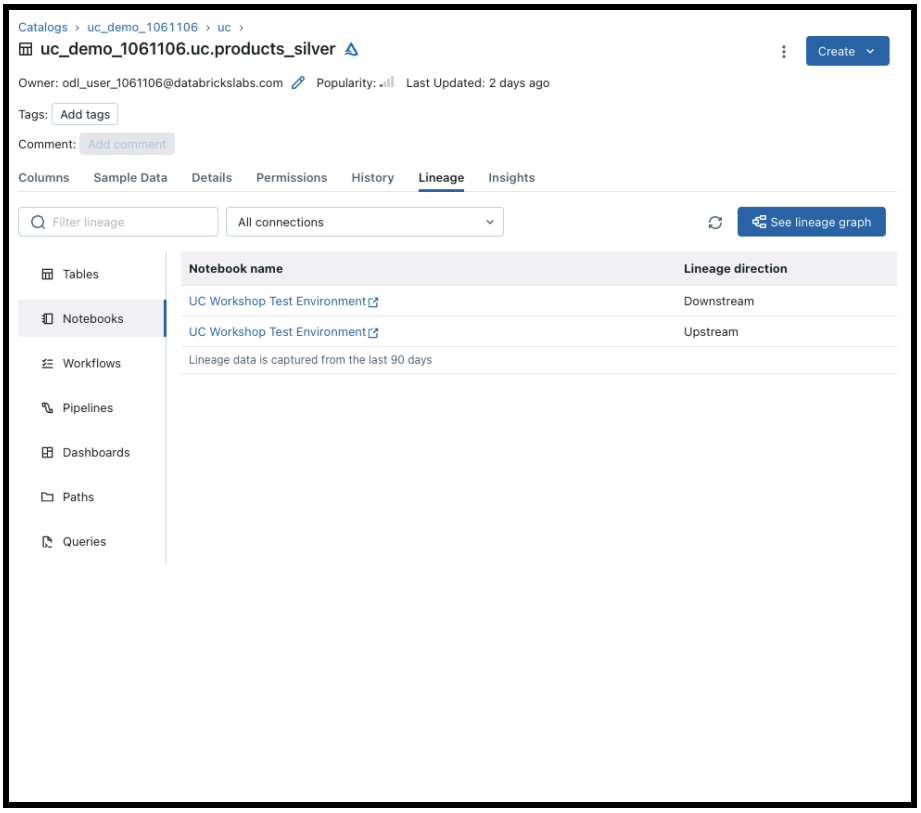
Here we can see that the products_silver table was generated upstream from the products_bronze table




Clicking “see lineage graph”, you are able to see the data lineage for the products_silver table in much more detail. You can even click into the table to get more granular detail like columnar lineage.




You can also see the data lineage from the notebooks that generated the tables upstream and downstream

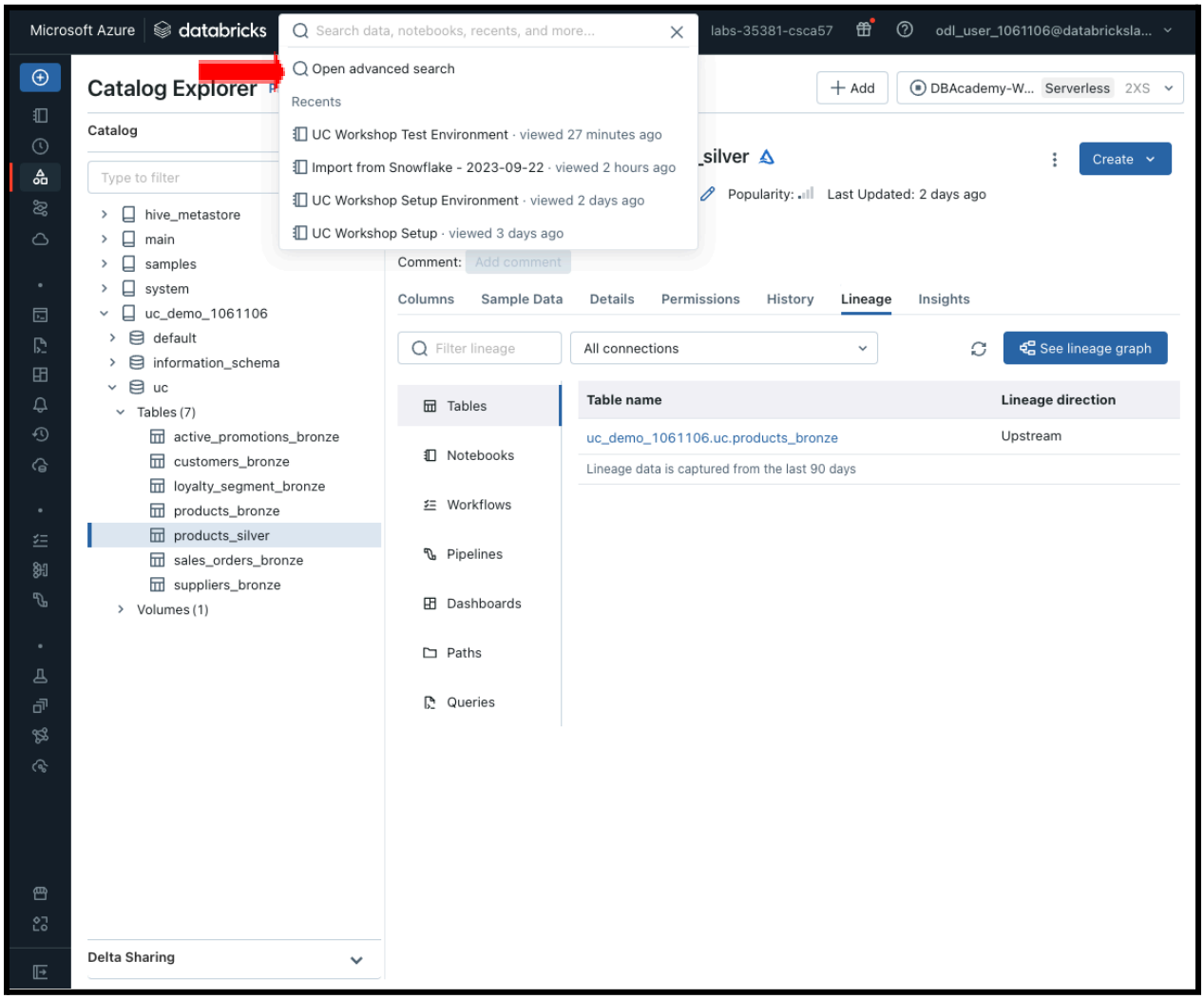



# Unified Search

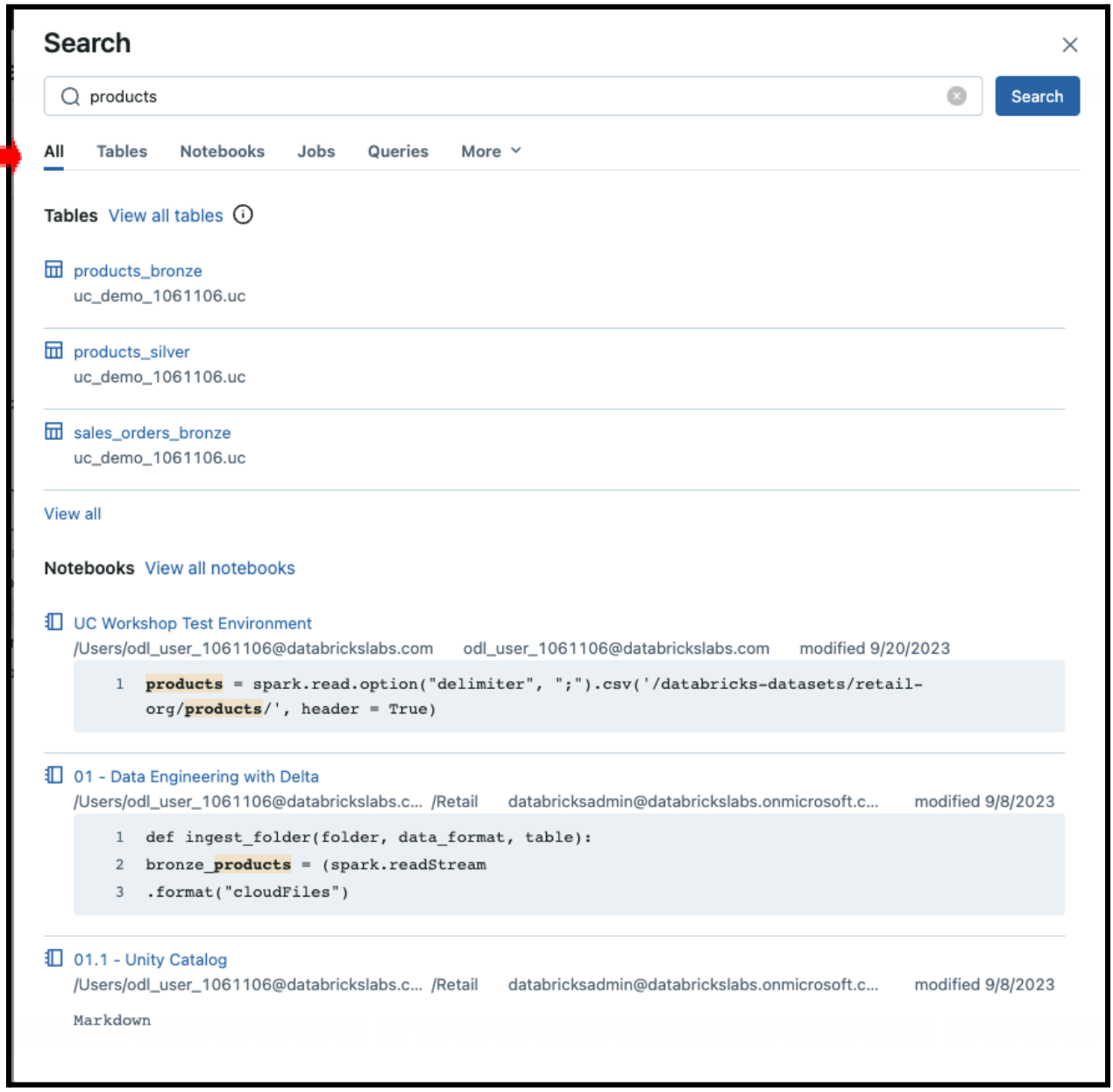
What if you wanted to search for other notebooks in your workspace? This is where unified search comes in.

Just click the Search field in the top of your Databricks workspace, click “Open advanced search” and you can enter your search criteria. You can search for notebooks, queries, dashboards, alerts, files, folders, libraries, jobs or repos in your Databricks workspace; and you can search by text string, by object type or both.

You can also search for Tags, which you can add to Tables and Columns to help users find specific data they are looking for. You need the APPLY TAG permission to add a Tag to a Table or Column.



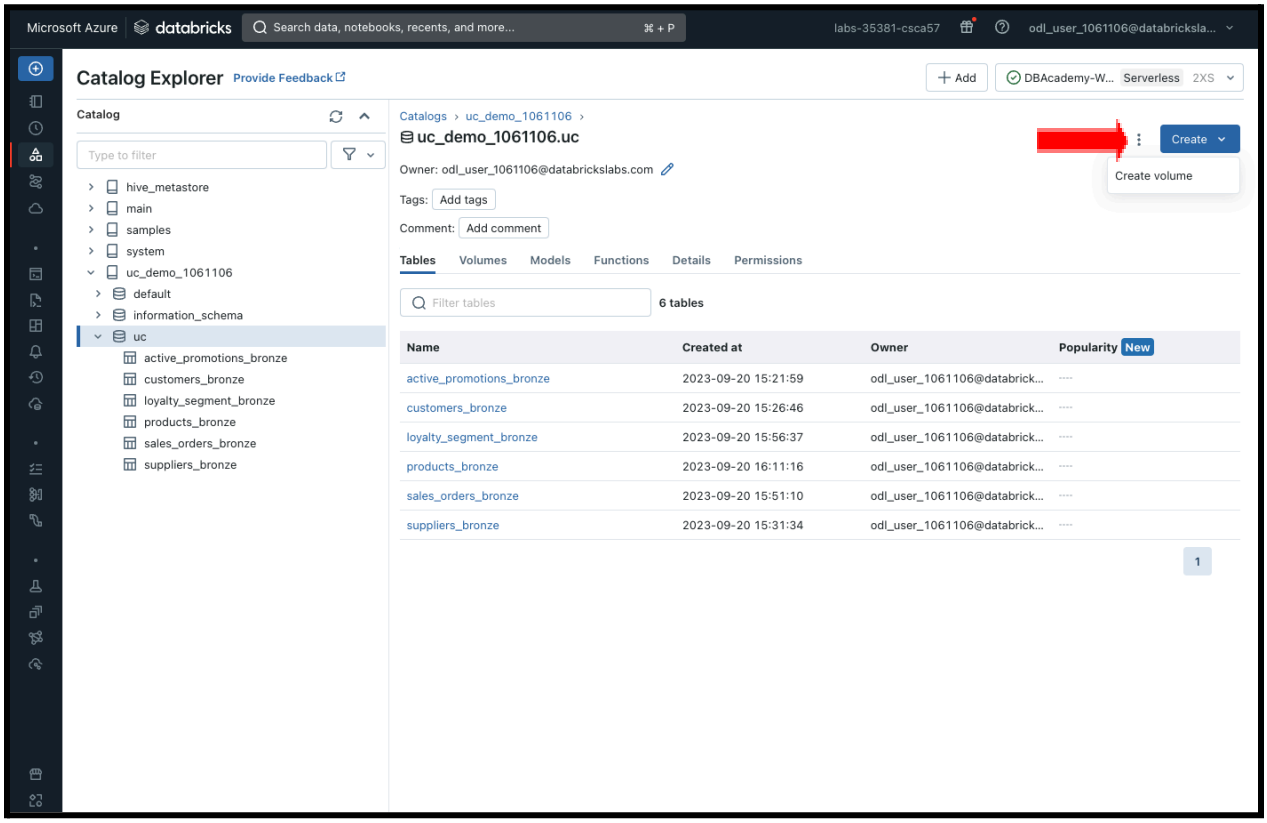

In this example, I am searching for All objects that have the keyword “products” in it. I can see the tables and notebooks are the only objects that have my keyword that I am looking for and if I click into any of them they will bring up the content I need for quick access or reference.



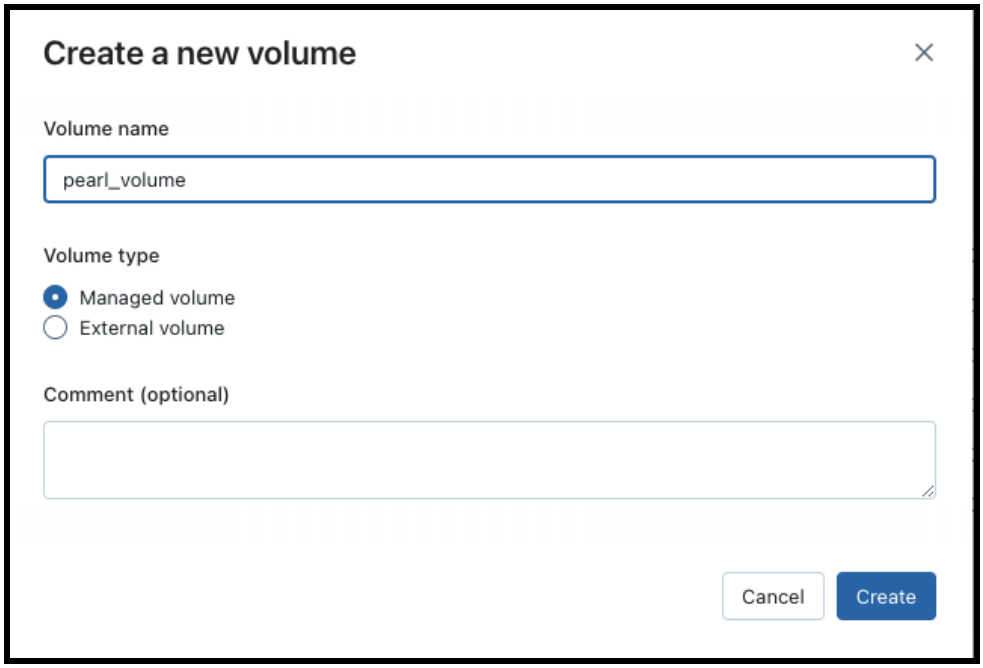

## Creating Volumes

Volumes are Unity Catalog objects representing a logical volume of storage in a cloud object storage location. Volumes provide capabilities for accessing, storing, governing, and organizing files. While tables provide governance over tabular datasets, volumes add governance over non-tabular datasets. You can use volumes to store and access files in any format, including structured, semi-structured, and unstructured data.

To create a volume, head back to your catalog and then schema in the Catalog Explorer and click the “Create” button in the top right corner and click “Create volume”:



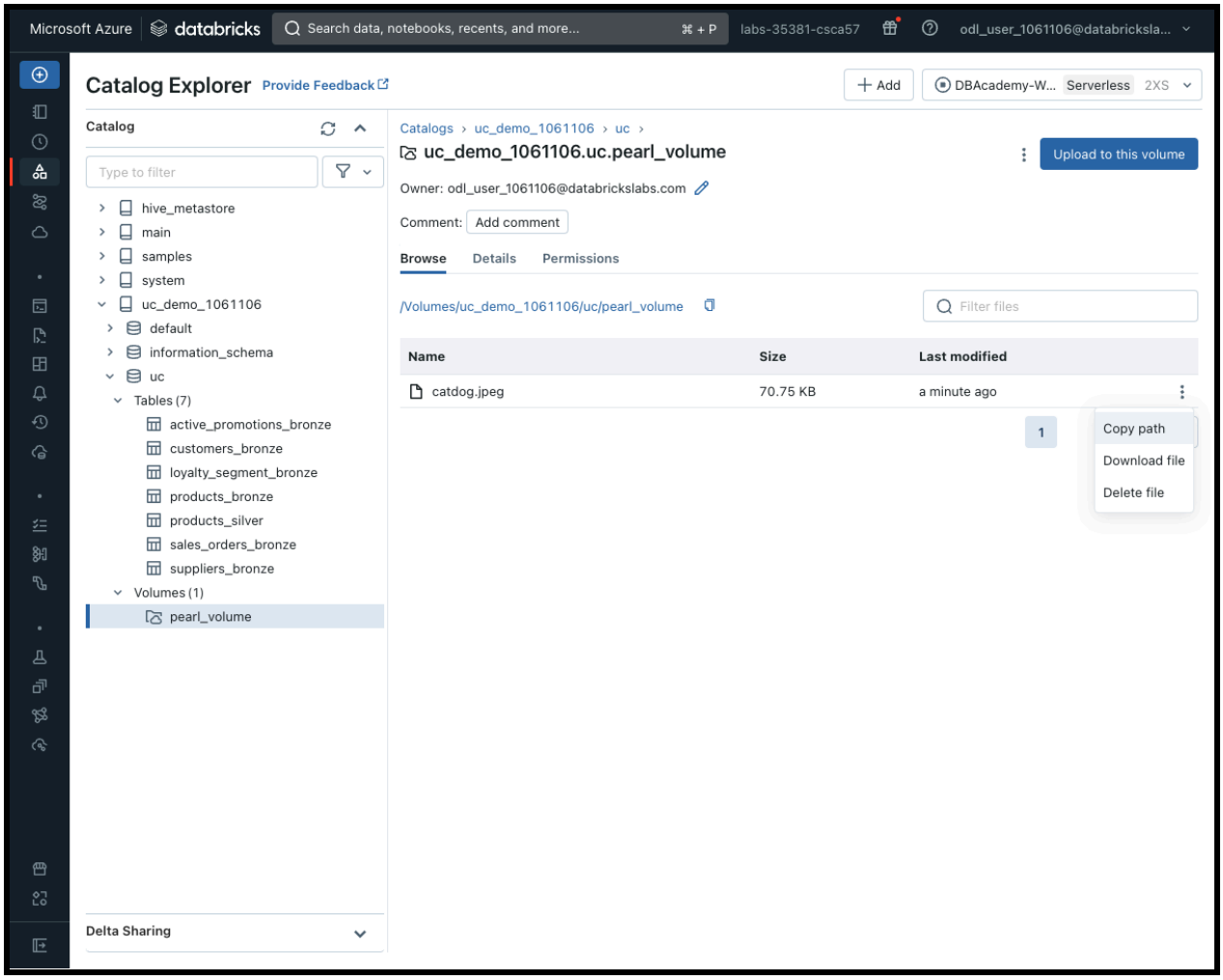

Give your volume the name uploads and click “Create”:



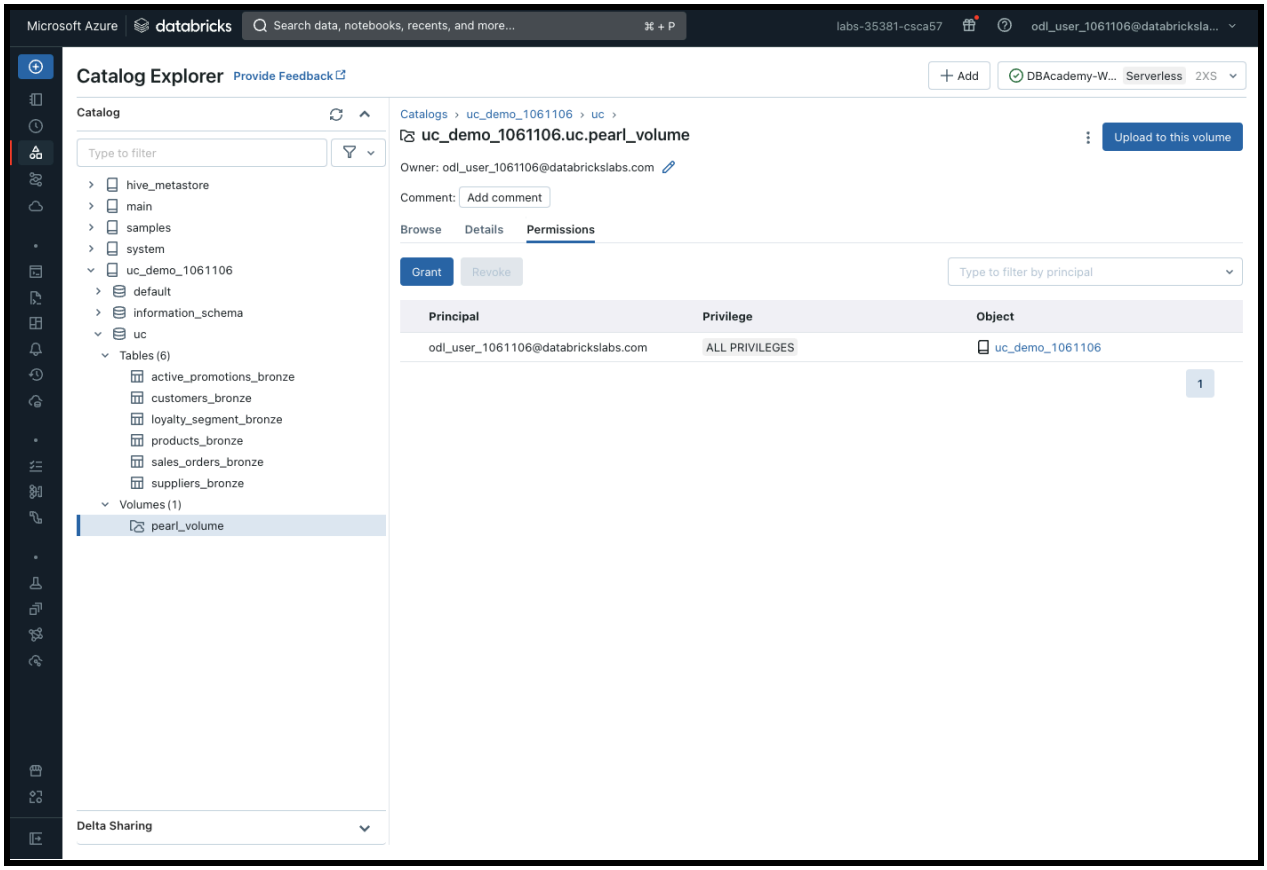

You can have an external source for your volume where you can add unity catalog data governance to existing cloud object storage directories. This can be great to add governance to data files without having to migrate them or even providing governance to files produced by other systems that must be ingested or accessed by Databricks.

For the purpose of this workshop, we will be creating a managed volume which will be created within the default storage location of the containing schema.

Once your volume is created you can upload, store and access files in any format. You can do this by clicking the “Upload to this volume” button and add your files. For the purposes of this workshop, we will not upload any content, however here is an example of unstructured data, an image, that has been uploaded for image recognition for easy reference for an ML project.




You can also set access controls at the volume level as well by clicking the Permissions tab. By doing this you will be able to see that the permissions and privileges you set at the Catalog level have trickled down to the volume level as well; however you are more than welcome to grant or revoke additional privileges to users or groups of users at the volume level as well.




# Querying the Data using Databricks SQL

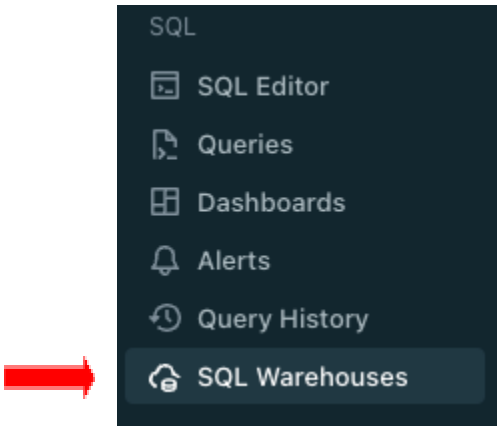
For our analyst team, they may want to query the data further to derive insights and perhaps create a dashboard. Our Data Warehousing solution Databricks SQL allows for this.

Databricks SQL distinguishes itself with its ability to handle massive datasets with speed and efficiency. Utilizing Databricks' next-gen engine, Photon with AI-driven optimizations, ensures rapid data processing and analysis, notably decreasing query execution durations. High performance is crucial for organizations facing data challenges, guaranteeing insights from an extensive variety of data sets. Moreover, Databricks SQL champions collaboration, providing a workspace where data professionals can instantaneously share queries, outcomes, and understandings. This shared setting promotes knowledge exchange and hastens resolution, allowing organizations to capitalize on their teams' collective intelligence.

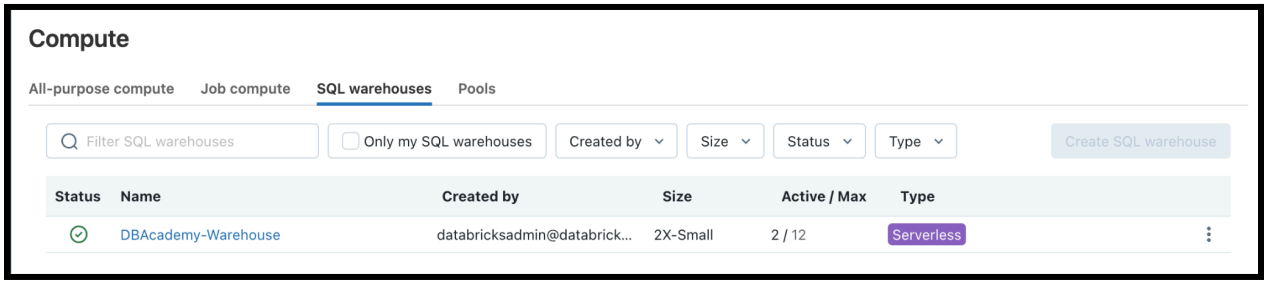
SQL warehouses are the compute resources that let you run SQL commands on data objects within Databricks SQL.

In this workshop, we will be unable to edit SQL warehouses, but lets navigate to the SQL warehouse page to see how everything works.

First, navigate to the SQL warehouse page

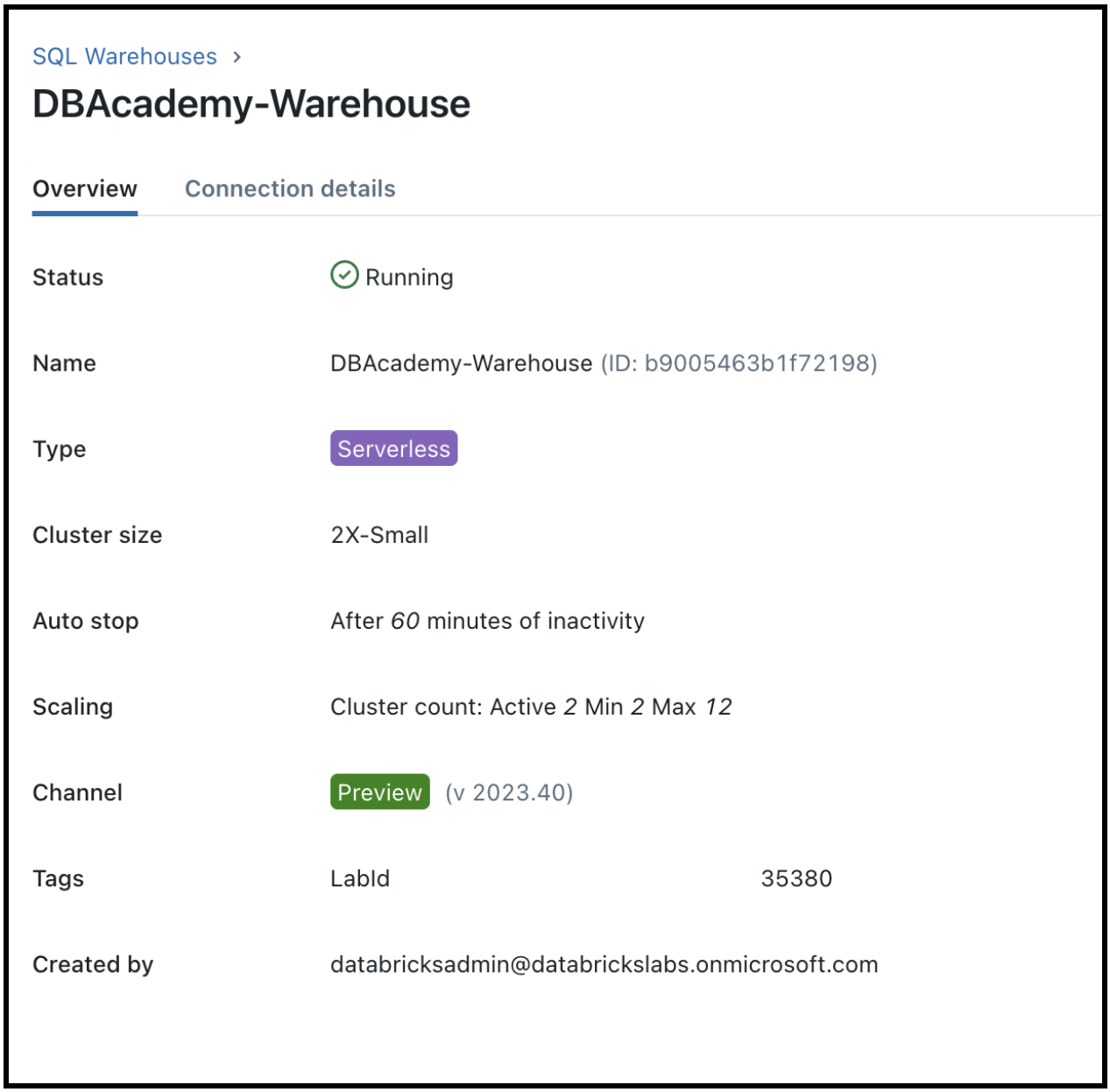



You will notice a pre-configured SQL Warehouse--called “DBAcademy-Warehouse”--that is already started up. If it is not started up, please hit the play icon on the warehouse.

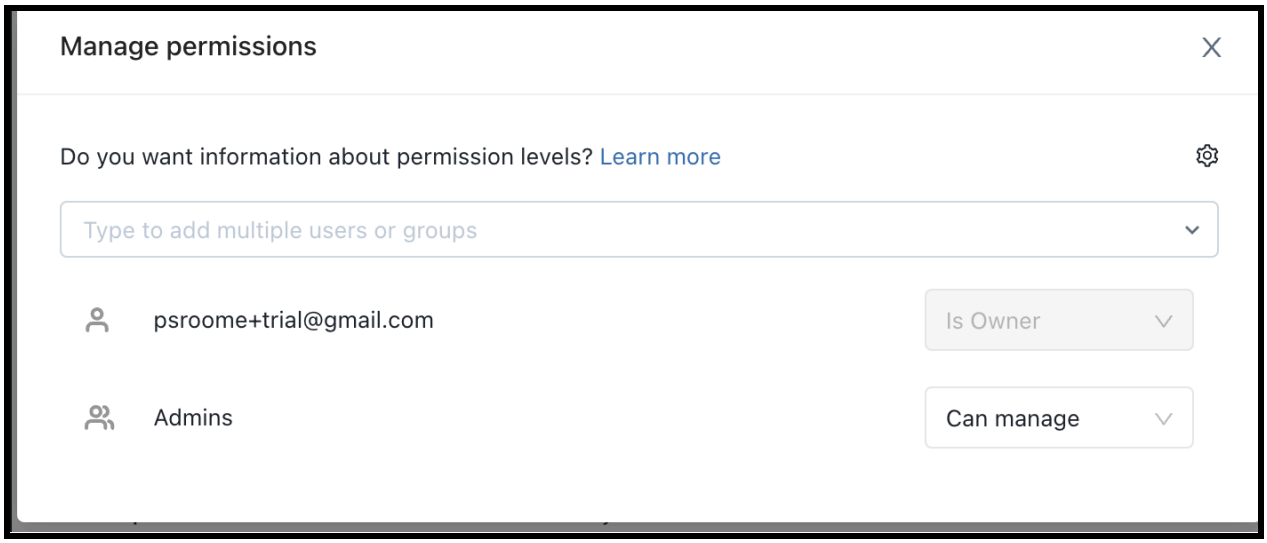



Click into the warehouse to see how it was configured.

- **Cluster Size**: for this Section, we chose a 2X-small cluster size (keeps the cost down!)
    - (To reduce the latency of queries, you can increase the size. Larger sized clusters have a larger coordinator and doubles the number of cluster workers.)
- **Auto Stop**: whether the warehouse stops if it’s idle for the specified number of minutes.
- **Scaling**:This scaling piece here is for horizontal scaling, so typically a cluster will be able to handle 10 concurrent queries before doing any queuing. If you have more than 10 concurrent queries you could have it auto-scale up and down to handle concurrency horizontally.
- **Serverless**: run your SQL and BI workloads on a fully managed secure compute environment that starts, scales up & down within seconds
- **Tags**: allow you to easily monitor the cost of cloud resources used by various groups in your organization. You can specify tags as key-value pairs when you create an warehouse, and Databricks applies these tags to cloud resources.
- **Channel**: allows you to choose whether to use the current SQL warehouse compute version or the preview version. Preview versions let you try out functionality before it becomes the Databricks SQL standard



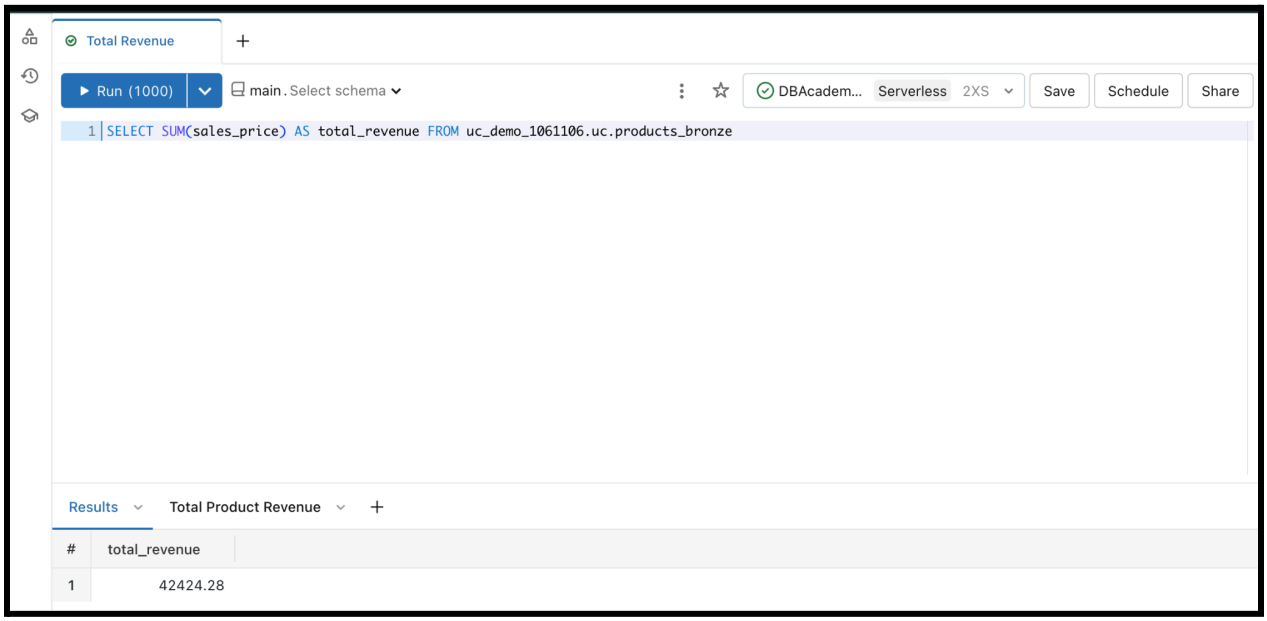

Although we cannot edit the permissioning on this particular warehouse, there is an ability also to provide access to other users or groups to run or manage the SQL warehouse.




For more information on configuring a SQL warehouse, click here.

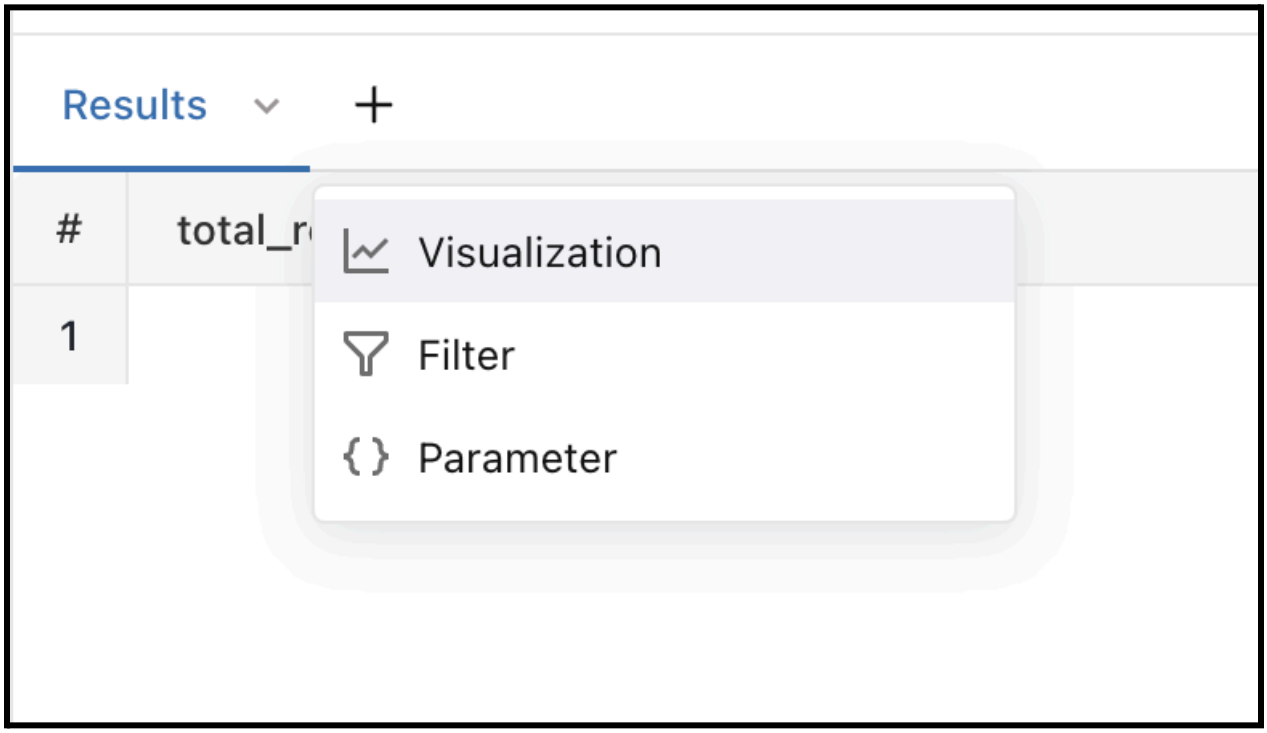
Now that our compute resource is configured and running, lets head to the SQL editor to run some queries and build our visualizations.

First we will change the title of our query so that it is easy to find. Let's title it Total Revenue. Then we will run our select statement to generate the total revenue.

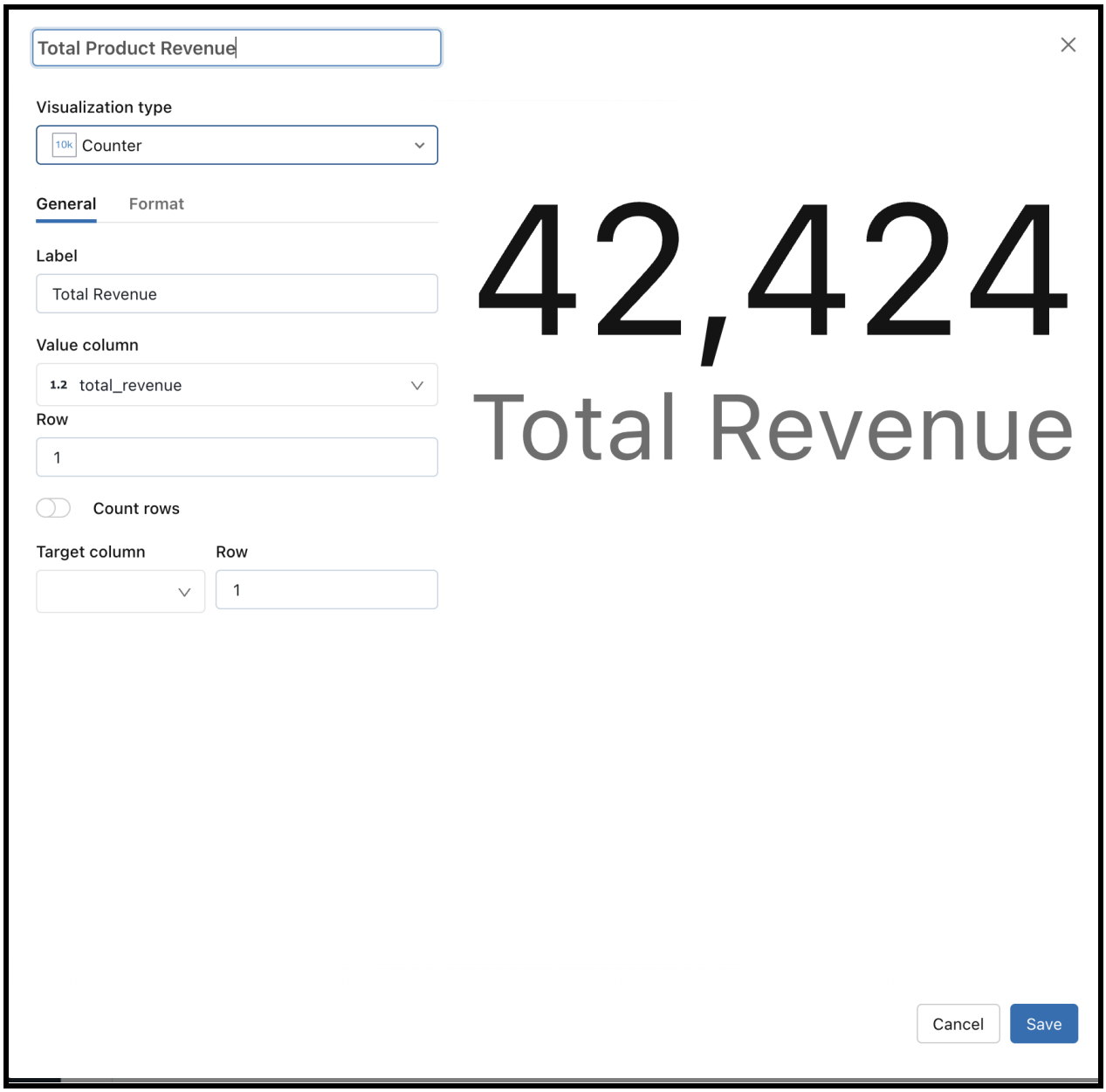


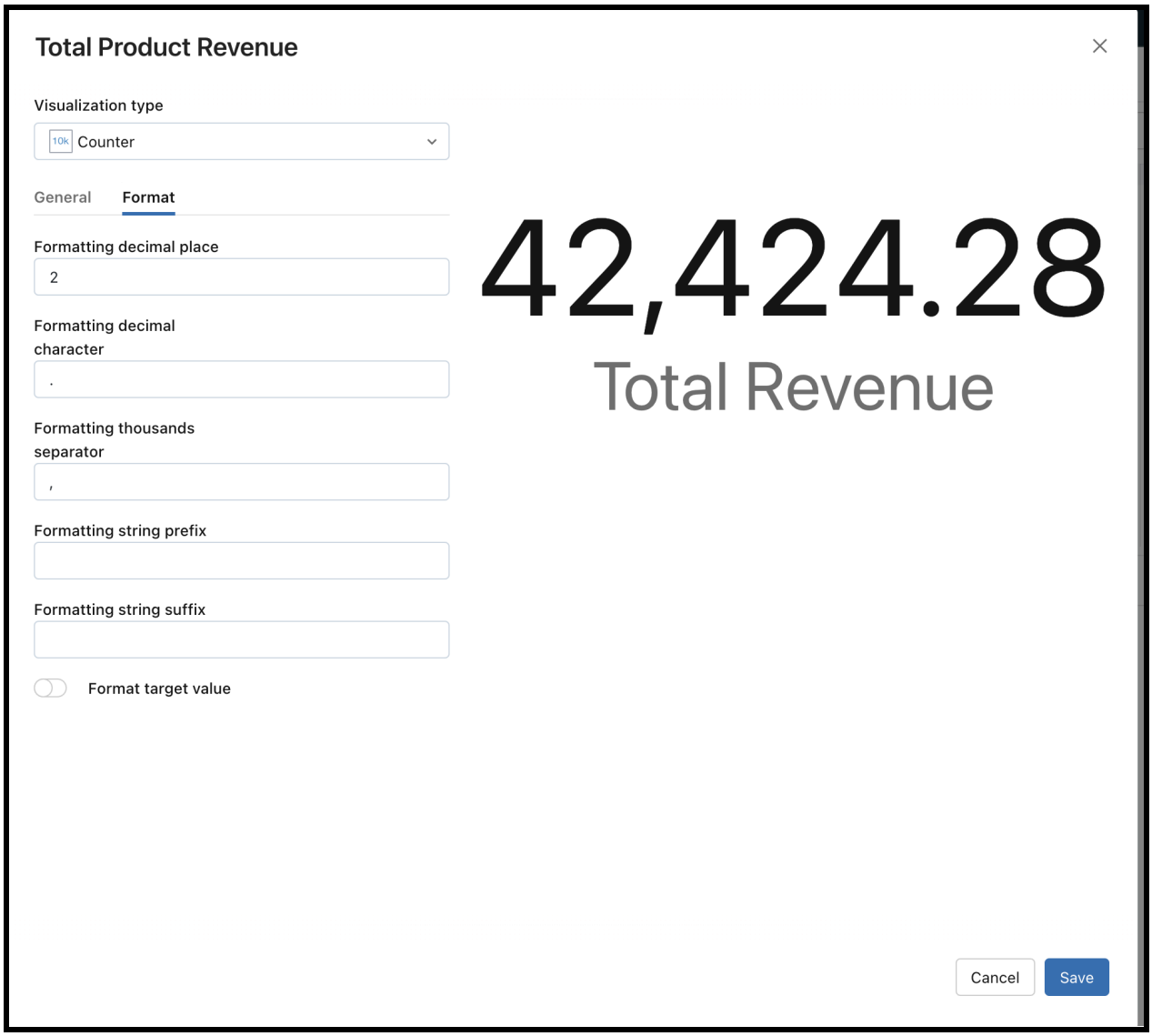

```
SELECT SUM(sales_price) AS total_revenue FROM uc_demo_xxxxxx.uc.products_bronze
```

To create a visualization, click the plus sign beside the results

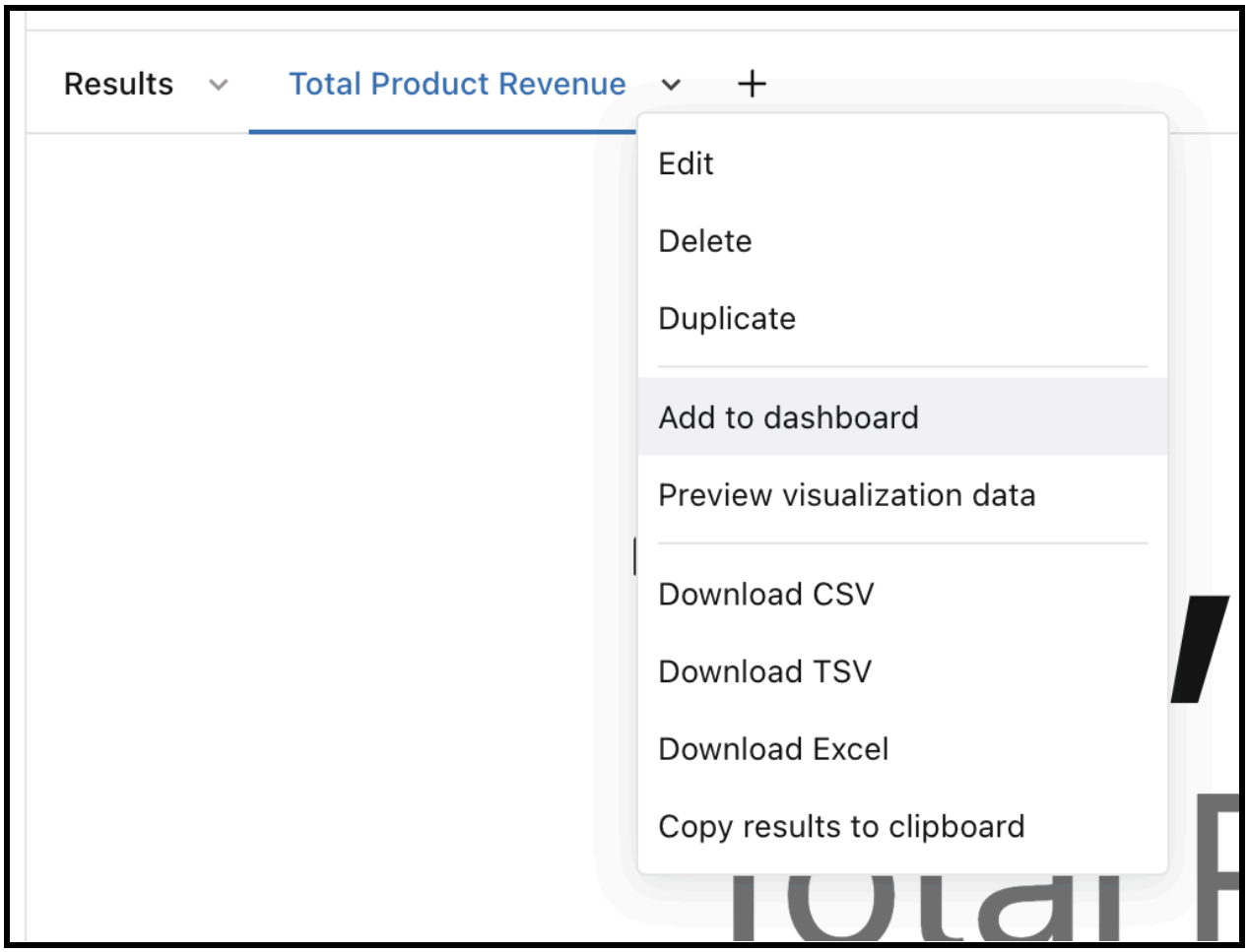


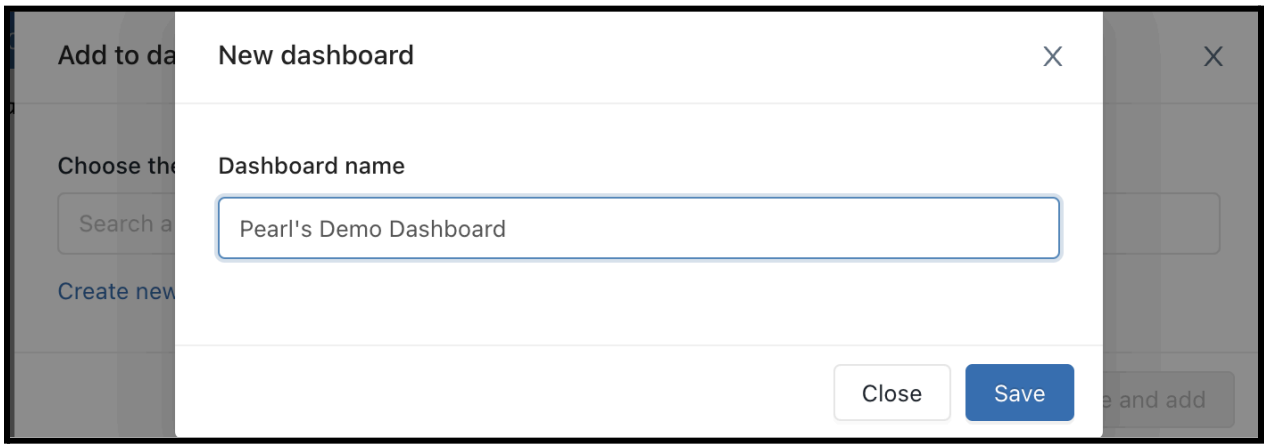

Make sure your edited visualization matches what you see below.


Total Product Revenue
Visualization type
Counter
General
Format
Label
Total Revenue
Value column
total_revenue
Row
1
Count rows
Target column
Row
1
42,424
Total Revenue
Cancel
Save

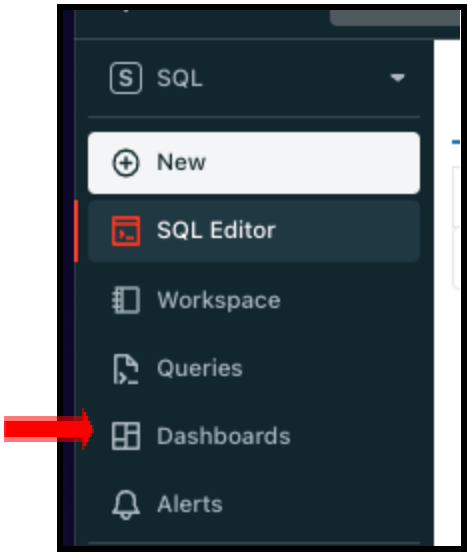



Click Save and then now we can add the viz to our dashboard.

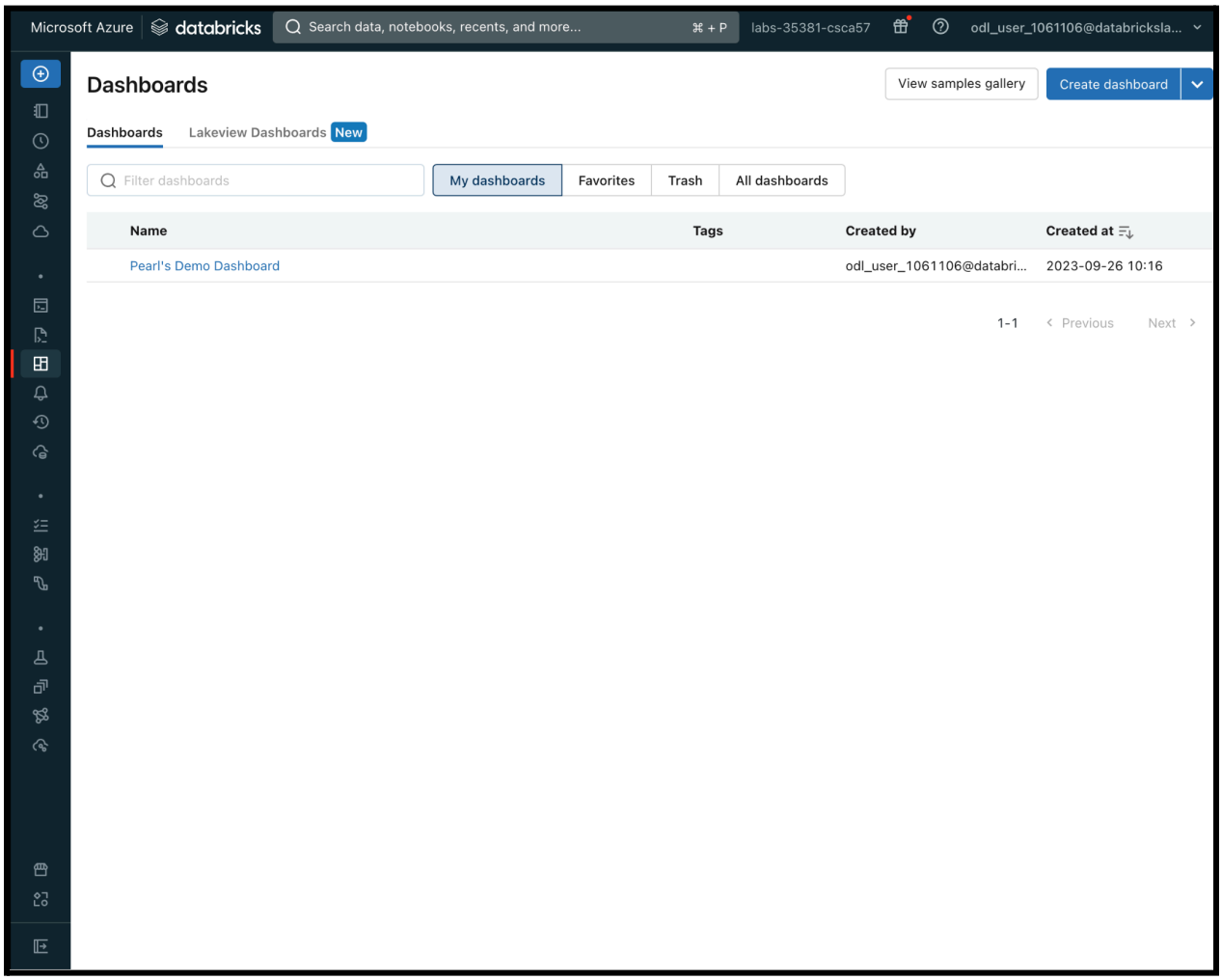


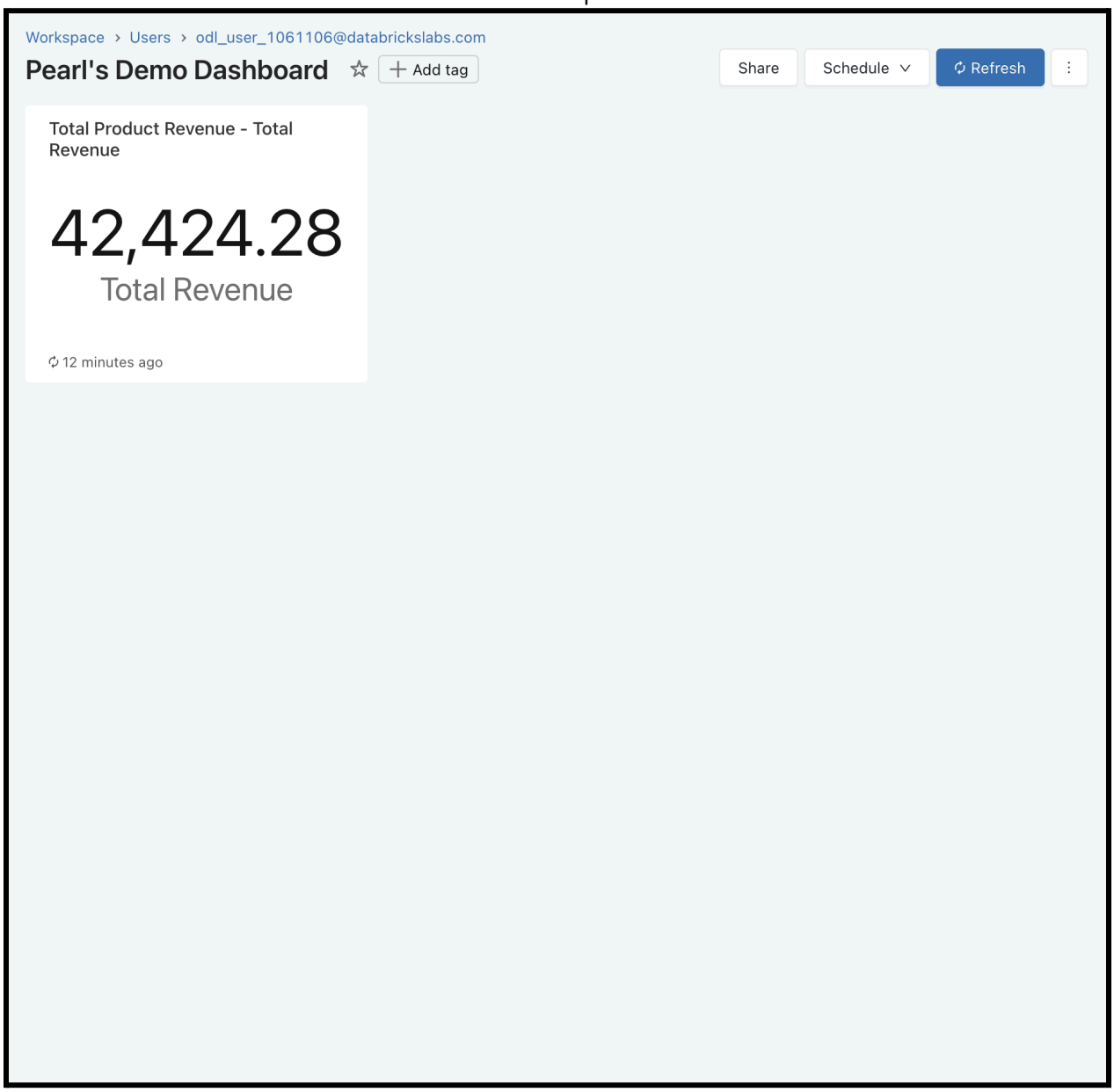

Now lets navigate to our dashboard by clicking Dashboards in the left hand navigation pane



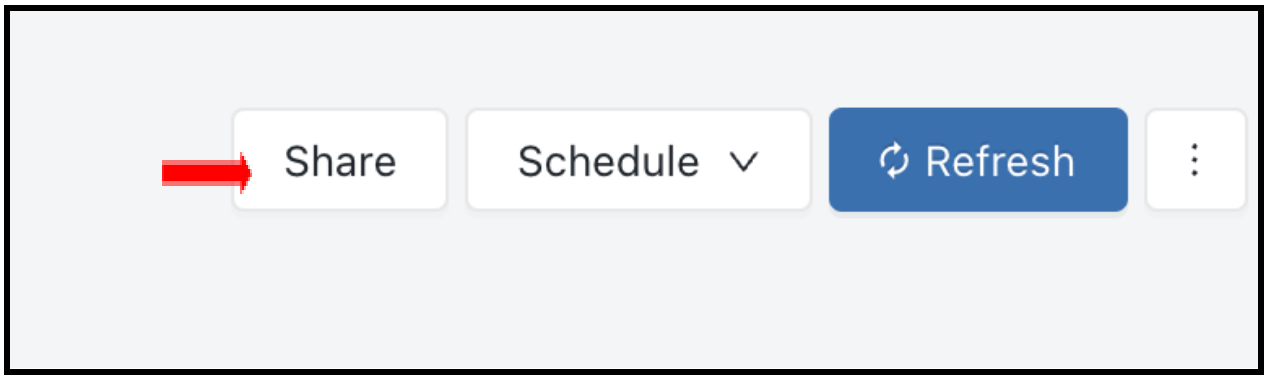

There you should see your newly created dashboard



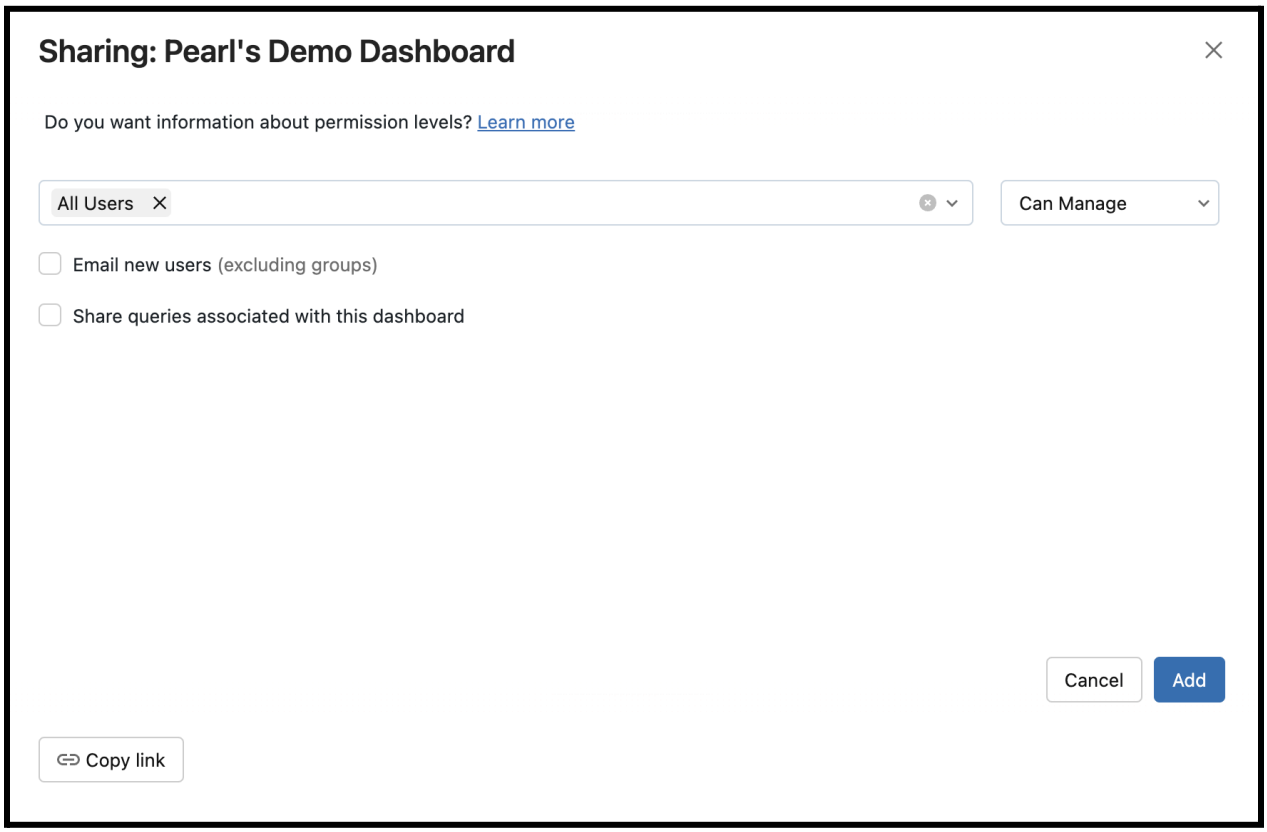
Clicking into it you will see the visualization that you created as part of your new dashboard. For the sake of time for this workshop we will not build out a full dashboard.




Permissioning can be set by clicking the Share button in the top right corner.




And similarly, you can give can manage, can edit, can run or can view privileges to users and/or groups. Additionally, you can notify new users that you are sharing the dashboard with them and you can share the queries that are associated with the visualizations behind the dashboard.




**** End of Hands-On Portion of Workshop ****

## Row Filters and Column Level Security

In Unity Catalog, Row filters allow you to apply a filter to a table so that subsequent queries only return rows for which the filter predicate evaluates to true. Both row and column level security are implemented by functions that are SQL user-defined functions (UDF).

Column masks let you apply a masking function to a table column. The masking function gets evaluated at query runtime, substituting each reference of the target column with the results of the masking function.

- Live Demo
- Table ACL Notebook

## System Tables

System tables are a Databricks-hosted analytical store of your account's operational data. System tables can be used for historical observability across your account.

Since system tables are governed by Unity Catalog, you need to have at least one Unity Catalog-enabled workspace in your account to enable and access system tables. System tables include data from all workspaces in your account but they can only be accessed from a Unity Catalog-enabled workspace. System tables must be enabled by an account admin. You can enable system tables using the Unity Catalog REST API.

- Live Demo
- System Tables Demo Tutorial - Demo Center

## Lakehouse Federation

Lakehouse Federation capabilities in Unity Catalog allow you to discover, query, and govern data across data platforms including MySQL, PostgreSQL, Amazon Redshift, Snowflake, Azure SQL Database, Azure Synapse, Google's BigQuery, and more from within Databricks without moving or copying the data, all within a simplified and unified experience. This means Unity Catalog's advanced security features such as row and column level access controls, discovery features like tags, and data lineage will be available across these external data sources, ensuring consistent governance.

- Live Demo
- Lakehouse Federation Product Tour

## Feature Engineering in Unity Catalog

Feature engineering includes steps such as scaling or normalizing data, encoding non-numeric data (such as text or images), aggregating data by time or entity, joining data from different sources, or even transferring knowledge from other models. The goal of these transformations is to increase the ability of machine learning algorithms to learn from the data set and thus make more accurate predictions. With Geature Engineering in Unity Catalog, you can search for feature tables by feature table name, feature, comment or tag; filter feature tables by tags and explore and manage feature tables with the Catalog Explorer.

Feature Engineering in Unity Catalog requires Databricks Runtime 13.2 ML or above. In addition, the Unity Catalog metastore must have Privilege Model Version 1.0.

- Live Demo
- Feature Engineering Notebook Demo

## Lakehouse Monitoring

Databricks Lakehouse Monitoring lets you monitor the statistical properties and quality of the data in all of the tables in your account. You can also use it to track the performance of machine learning models and model-serving endpoints by monitoring inference tables that contain model inputs and predictions. The diagram shows the flow of data through data and ML pipelines in Databricks, and how you can use monitoring to continuously track data quality and model performance.

To draw useful insights from your data, you must have confidence in the quality of your data. Monitoring your data provides quantitative measures that help you track and confirm the quality and consistency of your data over time. When you detect changes in your table's data distribution or corresponding model's performance, the tables created by Databricks Lakehouse Monitoring can capture and alert you to the change and can help you identify the cause.

Databricks Lakehouse Monitoring helps you answer questions like, what does data integrity look like, and how does it change over time? For example, what is the fraction of null or zero values in the current data, and has it increased? Or, what does the statistical distribution of the data look like, and how does it change over time? In addition, Databricks Lakehouse Monitoring lets you control the time granularity of observations and set up custom metrics.

- Live Demo
- Lakehouse Monitoring Demo Video - Demo Center

**** End of Demo Portion of Workshop ****