

# 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\\_V1dbc](#)

## Table of Contents

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

## Navigating the Catalog Explorer

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

The screenshot shows the Databricks Get started page. On the left is a dark sidebar with a red arrow pointing to the 'Catalog' section. The main area has a 'Get started' heading and four cards:

- Import and transform data**: Create a table by uploading local files, or create a pipeline for continuous data ingestion and transformation.  
Buttons: Create table, Create pipeline.
- Notebook**: Create a new notebook for data analysis, transformation, and machine learning.  
Button: Create notebook.
- SQL query editor**: Create a new query and explore your data in the SQL Editor.  
Button: Create query.
- AutoML**: Accelerate the training of ML models for efficient discovery and iteration.  
Button: Start AutoML.

Below this is a 'Pick up where you left off' section showing a recent notebook entry:

- UC Workshop Test Environment  
Notebook - 20 hours ago

Buttons: Recents, Favorites, Popular (NEW), Provide feedback.

A 'Popular' section is shown with a clock icon and the text: 'Popular assets appear here'.

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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, a sidebar menu includes options like New, Workspace, Recents, Catalog (which is selected and highlighted in red), Workflows, Compute, SQL (with sub-options like SQL Editor, Queries, Dashboards, Alerts, Query History, and SQL Warehouses), Data Engineering (Job Runs, Data Ingestion, Delta Live Tables), Machine Learning (Experiments, Features, Models, Serving), Marketplace, and Partner Connect. A 'Collapse menu' option is at the bottom. The main area is titled 'Catalog Explorer' with a 'Provide Feedback' link. It features a search bar and a 'Catalogs' section with a 'Create catalog' button. The 'Catalogs' table lists four entries:

Name	Created at	Owner
hive_metastore		
main	2023-09-07 03:23:36	databricksadmin@databr... ...
samples		
system	2023-09-07 03:23:36	System user

A 'Delta Sharing' dropdown menu is visible at the bottom left of the main area.

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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, a dark sidebar menu lists various Databricks services: Workspace, Recents, Catalog (selected), Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, Experiments, Features, Models, Serving, Marketplace, Partner Connect, and Collapsible menu. The main area is titled "Catalog Explorer" and contains a "Catalogs" section. It displays a table with four rows:

Name	Created at	Owner
hive_metastore	2023-09-07 03:23:36	databricksadmin@databr... [REDACTED]
main	2023-09-07 03:23:36	databricksadmin@databr... [REDACTED]
samples		
system	2023-09-07 03:23:36	System user

At the top right of the Catalogs section, there is a red arrow pointing towards the "Create catalog" button. The top navigation bar includes Microsoft Azure, the Databricks logo, a search bar, and user information: labs-35381-csca57, odl\_user\_1061106@databricks... (with a red arrow over it), and a dropdown for DBAcademy-W... Serverless 2XS.

Then, give the catalog a name “uc\_demo\_XXXXXXX” **Note: the XXXXXX come from your odl\_user\_XXXXXXX in the top right corner of your workspace**

A large black rectangular box with a white arrow pointing to the right, containing the text "odl\_user\_XXXXXX\_1024490@databricks...".

## Create a new catalog

A catalog is the first layer of Unity Catalog's three-level namespace and is used to organize your data assets. [Learn more](#)

Catalog name

Type

Standard

Storage location (optional)

Select external location  sub/path

Location in cloud storage where data for managed tables will be stored. If not specified, the location will default to the metastore root location.

Comment (optional)

The dialog box has a black border and a white background. The title 'Create a new catalog' is at the top left. A close button 'X' is at the top right. Below the title is a descriptive text block. The 'Catalog name' field is highlighted with a blue border. The 'Type' section shows 'Standard' selected. The 'Storage location (optional)' section includes a dropdown menu and an input field. A note about the storage location is provided below. The 'Comment (optional)' section has a large text area. At the bottom are 'Cancel' and 'Create' buttons.

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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, a sidebar lists various Databricks services: New, Workspace, Recents, Catalog (selected), Workflows, Compute, SQL (SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses), Data Engineering (Job Runs, Data Ingestion, Delta Live Tables), Machine Learning (Experiments, Features, Models, Serving), Marketplace, and Partner Connect. A 'Collapse menu' option is at the bottom. The main area is titled 'Catalog Explorer' and shows the 'Catalogs' section. It displays the catalog 'uc\_demo\_1061106' under 'Catalogs'. The catalog details show it is owned by 'odl\_user\_1061106@databrickslabs.com' and has no tags or comments. The 'Permissions' tab is selected, showing a table with columns 'Principal', 'Privilege', and 'Object'. The table is empty with the message 'No permissions granted yet.'.

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

X

Granted privileges will be inherited by applicable objects (e.g. schemas, tables) in this catalog.

[Learn more](#)

### Principals

odl\_user\_1061106@databrickslabs.com X

X ▾

### Privilege presets

Custom ▾

### Privileges

- |   |  |  |
|---|--|--|
| <input checked="" type="checkbox"/> USE CATALOG | <input checked="" type="checkbox"/> APPLY TAG    | <input checked="" type="checkbox"/> CREATE FUNCTION          |
| <input checked="" type="checkbox"/> USE SCHEMA  | <input checked="" type="checkbox"/> EXECUTE      | <input checked="" type="checkbox"/> CREATE MATERIALIZED VIEW |
|   | <input checked="" type="checkbox"/> MODIFY       | <input checked="" type="checkbox"/> CREATE MODEL             |
|   | <input checked="" type="checkbox"/> READ VOLUME  | <input checked="" type="checkbox"/> CREATE SCHEMA            |
|   | <input checked="" type="checkbox"/> REFRESH      | <input checked="" type="checkbox"/> CREATE TABLE             |
|   | <input checked="" type="checkbox"/> SELECT       | <input checked="" type="checkbox"/> CREATE VOLUME            |
|   | <input checked="" type="checkbox"/> WRITE VOLUME |  |

ALL PRIVILEGES gives all privileges ⓘ

Cancel

Grant

The screenshot shows the Databricks Catalog Explorer interface. On the left, a sidebar lists various navigation options such as Workspace, Recents, Catalog (which is selected), Workflows, Compute, SQL, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, Experiments, Features, Models, Serving, Marketplace, Partner Connect, and a Collapse menu. The main area is titled "Catalog Explorer" and shows the "Catalogs" section. A catalog named "uc\_demo\_1061106" is selected, indicated by a red arrow pointing to the "Create schema" button. The catalog details show the owner as "odl\_user\_1061106@databricks.com". Below this, there are tabs for Schemas, Details, Permissions, and Workspaces. Under the Schemas tab, a table lists two schemas: "default" and "information\_schema". The "default" schema was created on 2023-09-19 12:03:37 by "odl\_user\_1061106@databricks.com". The "information\_schema" schema was created on 2023-09-19 12:03:37 by "System user".

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 statement, 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.

## Create a new Schema

X

A schema is the second layer of Unity Catalog's three-level namespace and organizes tables and views. [Learn more](#)

Schema name

uc

Storage location (optional)

Select external location

sub/path

Location in cloud storage where data for managed tables will be stored. If not specified, the location will default to the metastore root location.

Comment (optional)

Cancel

Create

You will now see your new schema in the catalog.

The screenshot shows the Databricks Catalog Explorer interface. On the left, a dark sidebar lists various navigation options: Workspace, Recents, Catalog (which is selected and highlighted with a red arrow), Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning (Experiments, Features, Models, Serving), Marketplace, Partner Connect, and a Collapse menu. The main area is titled "Catalog Explorer" and shows the "Catalog" section. It displays a tree view of catalogs: "Catalog" > "Catalogs" > "uc\_demo\_1061106" > "uc\_demo\_1061106.uc". The "uc\_demo\_1061106.uc" catalog has an owner listed as "odl\_user\_1061106@databrickslabs.com". Below the catalog details, there are tabs for "Tables", "Volumes", "Models", "Functions", "Details", and "Permissions". The "Tables" tab is selected, showing a table header with columns: Name, Created at, Owner, and Popularity (with a "New" badge). A message below the table states: "You either don't have access to any tables or there are no tables in this schema. Either contact your administrator or [create a new sample table](#)." At the bottom of the main area, there is a "Delta Sharing" dropdown.

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](#)

The screenshot shows the Databricks Catalog Explorer interface. On the left, a dark sidebar lists various navigation options such as Workspace, Recents, Catalog (which is currently selected), Workflows, Compute, SQL, Data Engineering, Machine Learning, Marketplace, and Partner Connect. The main area is titled "Catalog Explorer" and shows the "Catalogs > uc\_demo\_1061106 > uc\_demo\_1061106" hierarchy. A search bar at the top allows filtering. To the right, there are tabs for Tables, Volumes, Models, Functions, Details, and Permissions. The Permissions tab is active, showing a table with one entry:

Principal	Privilege	Object
odl_user_1061106@databrick...	ALL PRIVILEGES	uc_demo_1061106

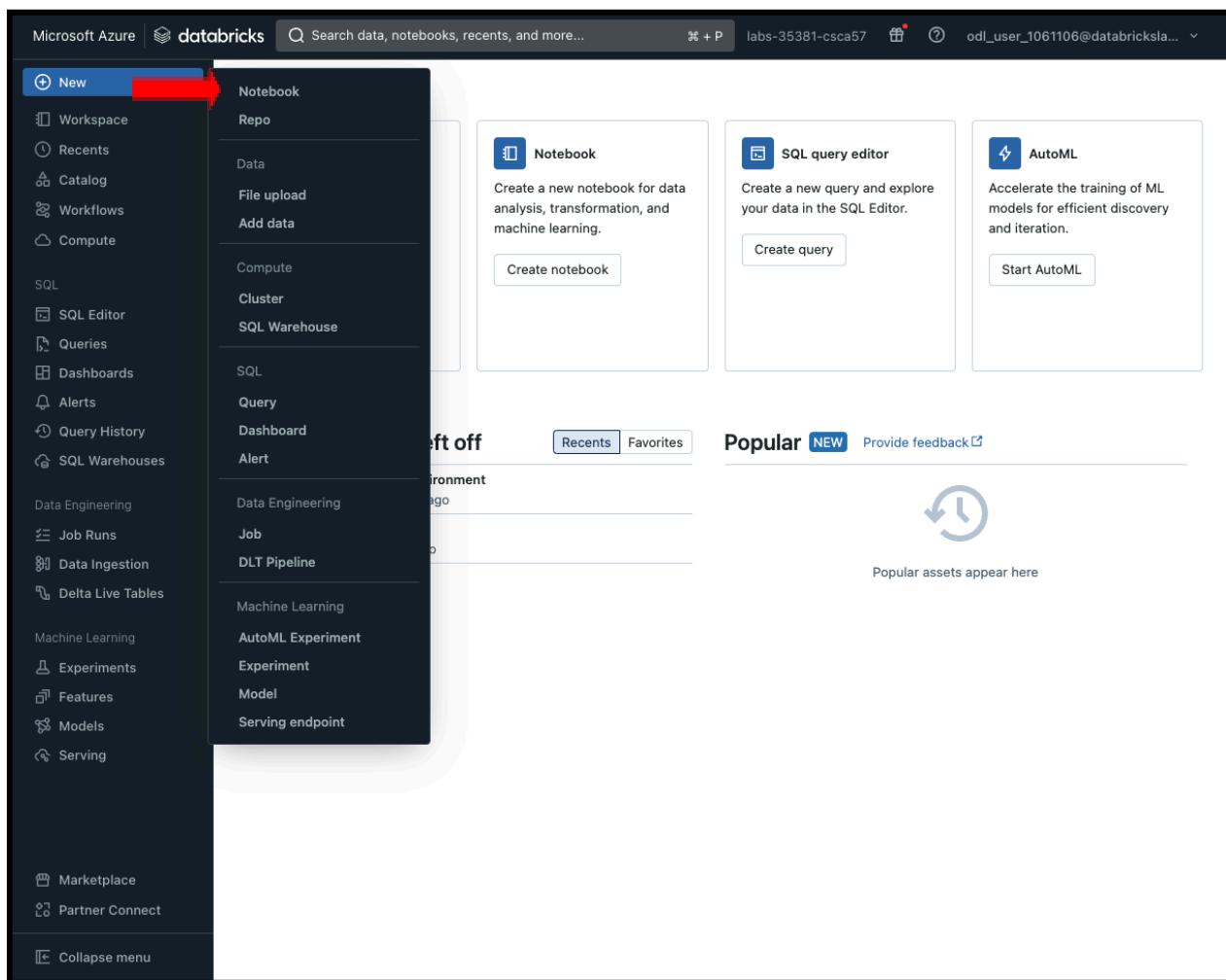
At the bottom of the main area, there is a "Delta Sharing" dropdown menu.

Now let us add some data by creating tables

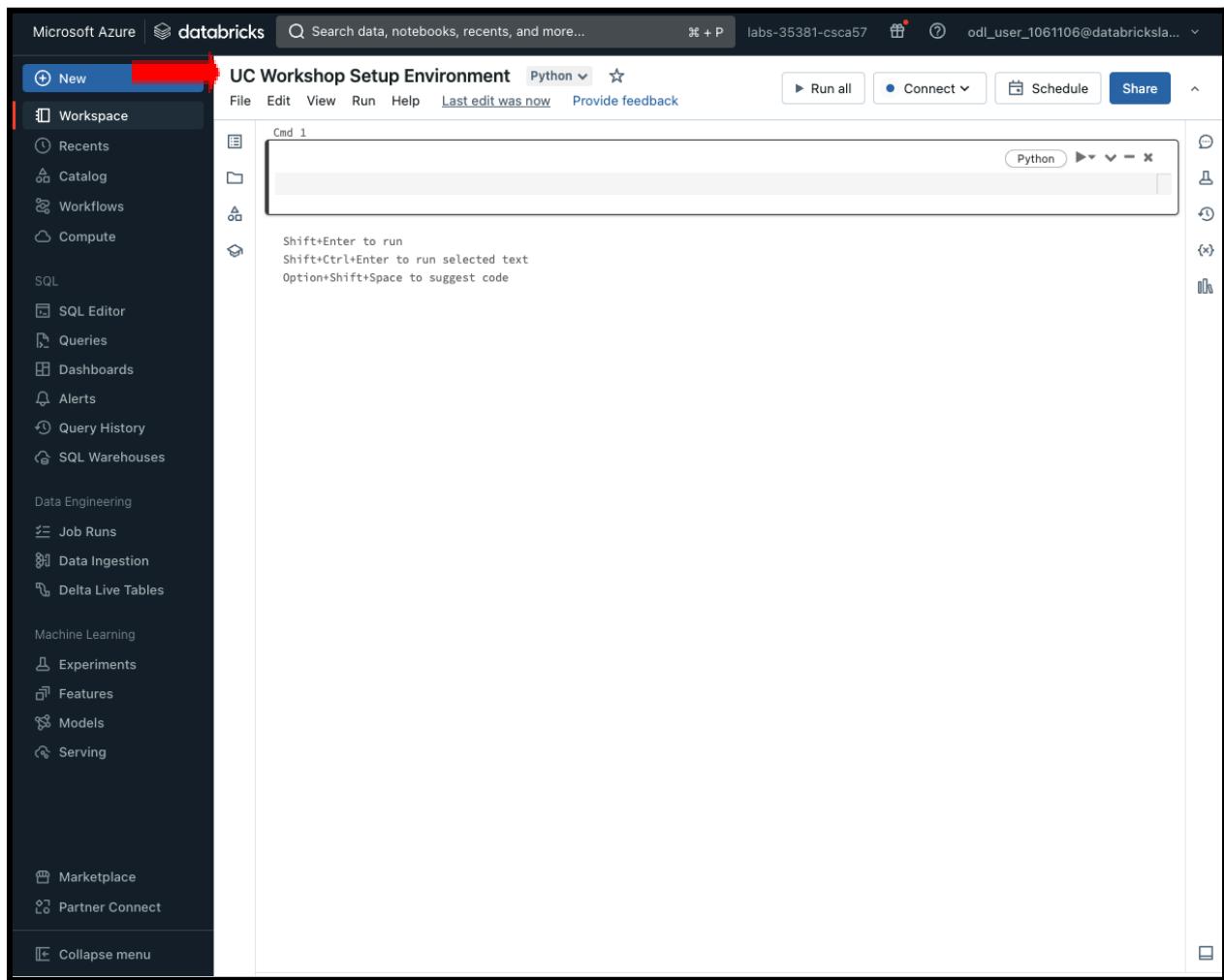
## Creating Tables

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

The screenshot shows the Microsoft Azure Databricks workspace interface. The left sidebar contains a navigation menu with sections like Recents, Catalog, Workflows, Compute, SQL, Data Engineering, Machine Learning, and Marketplace. The main area is titled "UC Workshop Setup Environment" and is set to Python. It displays a command line interface (Cmd 1) with a red arrow pointing to it, indicating where code is being run. Below the command line, there's a section for "Recent resources" showing "labcluster-1061106" (DBR 11.3 LTS) and "Active resources" showing "DBAcademy-Warehouse" (SQL, Preview, 2X-Small). A tooltip provides keyboard shortcuts for running code.

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

Cmd 1

### List Retail Databricks Dataset

Python

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

**Table** +

	path	name	size	modificationTime
1	dbfs:/databricks-datasets/retail-org/README.md	README.md	1678	1602776830000
2	dbfs:/databricks-datasets/retail-org/active_promotions/	active_promotions/	0	1695145589793
3	dbfs:/databricks-datasets/retail-org/company_employees/	company_employees/	0	1695145589793
4	dbfs:/databricks-datasets/retail-org/customers/	customers/	0	1695145589793
5	dbfs:/databricks-datasets/retail-org/loyalty_segments/	loyalty_segments/	0	1695145589793
6	dbfs:/databricks-datasets/retail-org/products/	products/	0	1695145589793

↓ 12 rows | 11.52 seconds runtime      Refreshed 23 hours ago

Command took 11.52 seconds -- by odl\_user\_1061106@databrickslabs.com at 9/19/2023, 12:46:18 PM on labcluster-1061106

Cmd 2

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

Cmd 2

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

1061106

Command took 0.12 seconds -- by odl\_user\_1061106@databrickslabs.com at 9/20/2023, 3:19:39 PM on labcluster-1061106

```
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_{user_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")
```

Cmd 8

Create and Ingest Customers bronze table

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

▶ (1) Spark Jobs
▶ [ ] customers: pyspark.sql.dataframe.DataFrame = [customer_id: string, tax_id: string ... 17 more fields]
Command took 0.80 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:26:03 PM on labcluster-1061106
```

Cmd 9

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

▶ (5) Spark Jobs
Command took 5.41 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:26:40 PM on labcluster-1061106
```

Cmd 10

```
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")
```

Cmd 10

Create and Ingest Suppliers bronze table

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

▶ (1) Spark Jobs
▶ [ ] suppliers: pyspark.sql.dataframe.DataFrame = [SUPPLIER_ID: string, TAX_ID: string ... 12 more fields]
Command took 0.65 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:31:08 PM on labcluster-1061106
```

Cmd 11

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

▶ (5) Spark Jobs
Command took 3.08 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:31:31 PM on labcluster-1061106
```

```
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")
```

Cmd 12

Create and Ingest Loyalty Segments bronze table

```
loyalty_segment = spark.read.csv('dbfs:/databricks-datasets/retail-org/loyalty_segments/', header = True)  
▶ (1) Spark Jobs  
▶ [loyalty_segment: pyspark.sql.dataframe.DataFrame = [loyalty_segment_id: string, loyalty_segment_description: string ... 3 more fields]  
Command took 0.57 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:56:10 PM on labcluster-1061106
```

Cmd 13

```
loyalty_segment.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{user_id}.uc.loyalty_segment_bronze")  
▶ (5) Spark Jobs  
Command took 3.66 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:56:33 PM on labcluster-1061106
```

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

```

Cmd 14
Create and Ingest Sales Orders bronze table

sales_orders = spark.read.json('/databricks-datasets/retail-org/sales_orders/')

▶ (1) Spark Jobs
▶ sales_orders: pyspark.sql.dataframe.DataFrame = [clicked_items: array, customer_id: string ... 6 more fields]
Command took 0.87 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:50:41 PM on labcluster-1061106

Cmd 15

sales_orders.write.format("delta").mode("overwrite").saveAsTable(f"uc_demo_{user_id}.uc.sales_orders_bronze")

▶ (5) Spark Jobs
Command took 4.37 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 3:51:06 PM on labcluster-1061106

```

```

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

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

```

```

Cmd 16
Create and Ingest Products bronze table

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

▶ (1) Spark Jobs
▶ products: pyspark.sql.dataframe.DataFrame = [product_id: string, product_category: string ... 5 more fields]
Command took 0.53 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 4:11:09 PM on labcluster-1061106

Cmd 17

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

▶ (5) Spark Jobs
Command took 3.08 seconds -- by odl_user_1061106@databrickslabs.com at 9/20/2023, 4:11:13 PM on labcluster-1061106

Cmd 18

```

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

The screenshot shows the Databricks Catalog Explorer interface. On the left, there's a sidebar with various icons for navigation. The main area displays the 'Catalog Explorer' with the title 'Catalogs > uc\_demo\_1061106 > uc\_demo\_1061106.uc'. It includes sections for 'Owner', 'Tags', and 'Comment'. Below these are tabs for 'Tables', 'Volumes', 'Models', 'Functions', 'Details', and 'Permissions'. A search bar labeled 'Filter tables' shows '6 tables'. The table list includes:

Name	Created at	Owner	Popularity
active_promotions_bronze	2023-09-20 15:21:59	odl_user_1061106@databrick...	New
customers_bronze	2023-09-20 15:26:46	odl_user_1061106@databrick...	
loyalty_segment_bronze	2023-09-20 15:56:37	odl_user_1061106@databrick...	
products_bronze	2023-09-20 16:11:16	odl_user_1061106@databrick...	
sales_orders_bronze	2023-09-20 15:51:10	odl_user_1061106@databrick...	
suppliers_bronze	2023-09-20 15:31:34	odl_user_1061106@databrick...	

At the bottom, there's a 'Delta Sharing' section.

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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, there's a sidebar with various icons for navigation. The main area displays a catalog tree under 'Catalog'. A specific table, 'active\_promotions\_bronze', is selected and highlighted with a blue border. At the top right, there are buttons for '+ Add', 'Create', and 'Permissions'. The 'Permissions' tab is active, showing a table with one row:

Principal	Privilege	Object
odl_user_1061106@databrickslabs.com	ALL PRIVILEGES	uc_demo_1061106

## 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_xxxxxxx.uc.products_silver AS  
SELECT product_category, SUM(sales_price) total_sales  
FROM uc_demo_xxxxxxx.uc.products_bronze  
GROUP BY product_category;  
  
SELECT * FROM uc_demo_xxxxxxx.uc.products_silver
```

Microsoft Azure |  databricks | Search data, notebooks, recents, and more...

UC Workshop Test Environment Python ⚡ File Edit View Run Help Last edit was 23 minutes ago Provide feedback Run all labcluster-1061106 Schedule Share

Cmd 19

```
%sql
CREATE OR REPLACE TABLE uc_demo_1061106.uc.products_silver AS
SELECT product_category, SUM(sales_price) total_sales
FROM uc_demo_1061106.uc.products_bronze
GROUP BY product_category;

SELECT * FROM uc_demo_1061106.uc.products_silver
```

(9) Spark Jobs

\_sqldf: pyspark.sql.dataframe.DataFrame = [product\_category: string, total\_sales: double]

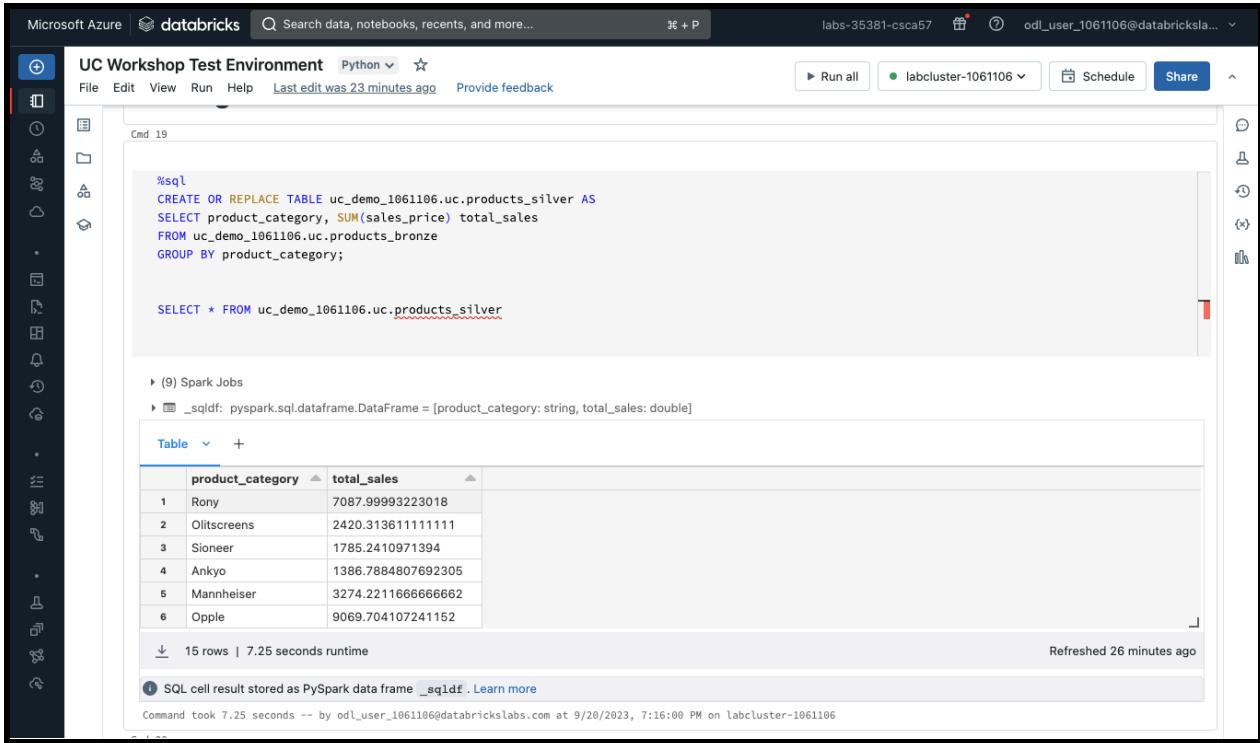
Table +

	product_category	total_sales
1	Rony	7087.99993223018
2	Ollitcreens	2420.313611111111
3	Sioneer	1785.2410971394
4	Ankyo	1386.7884807692305
5	Mannheiser	3274.2211666666662
6	Opple	9069.704107241152

↓ 15 rows | 7.25 seconds runtime Refreshed 26 minutes ago

SQL cell result stored as PySpark data frame \_sqldf. Learn more

Command took 7.25 seconds -- by odl\_user\_1061106@databrickslabs.com at 9/20/2023, 7:16:00 PM on labcluster-1061106



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

The screenshot shows the Databricks Catalog Explorer interface. On the left is a sidebar with various icons for navigation. The main area has a header "Catalog Explorer" and a search bar "Search data, notebooks, recents, and more...". The top right shows session information: "labs-35381-csca57", "odl\_user\_1061106@dbacademy...", and "DBAcademy-W... Serverless 2XS".

The central part displays a catalog structure under "Catalogs > uc\_demo\_1061106 > uc > uc\_demo\_1061106.uc.products\_silver". The table "products\_silver" is selected. The table details show it was last updated 2 days ago by "odl\_user\_1061106@dbacademylabs.com". It has 7 columns: product\_category (string) and total\_sales (double). The "Columns" tab is active.

The lineage tab is visible at the bottom of the table details section. A note at the bottom left says "Delta Sharing" and a note at the bottom right says "1".

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

The screenshot shows the Databricks lineage interface for the table `uc_demo_1061106.uc.products_silver`. The top navigation bar includes buttons for '+ Add', 'DBAcademy-W...', 'Serverless', and '2XS'. Below the navigation, the table name is displayed with a blue triangle icon indicating it's a new or updated table. The owner is listed as `odl_user_1061106@databrickslabs.com`, popularity is shown as '0.00', and it was last updated 2 days ago. There are buttons for 'Create' and 'Add tags'. A comment field with 'Add comment' is also present.

The main interface has tabs for 'Columns', 'Sample Data', 'Details', 'Permissions', 'History', 'Lineage', and 'Insights'. The 'Lineage' tab is selected. A search bar for 'Filter lineage' and a dropdown for 'All connections' are available. A large button labeled 'See lineage graph' with a magnifying glass icon is prominently displayed.

The lineage data table lists one upstream connection:

Table name	Lineage direction
<code>uc_demo_1061106.uc.products_bronze</code>	Upstream

A note below the table states: 'Lineage data is captured from the last 90 days'.

The sidebar on the left contains links for 'Tables', 'Notebooks', 'Workflows', 'Pipelines', 'Dashboards', 'Paths', and 'Queries'.

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

The screenshot shows the Databricks interface for a specific table. At the top, it displays the catalog path: Catalogs > uc\_demo\_1061106 > uc > uc\_demo\_1061106.uc.products\_silver. It also shows the owner (odl\_user\_1061106@datarickslabs.com), popularity (Last Updated: 2 days ago), and various metadata fields like Tags and Comment.

The main navigation bar includes tabs for Columns, Sample Data, Details, Permissions, History, Lineage (which is currently selected), and Insights. Below the navigation bar is a search bar labeled "Filter lineage" and a dropdown menu set to "All connections". To the right of the search bar is a button labeled "See lineage graph".

The left sidebar contains links to other Databricks objects: Tables, Notebooks (which is currently selected), Workflows, Pipelines, Dashboards, Paths, and Queries. The "Notebooks" section lists two entries: "UC Workshop Test Environment" with "Downstream" lineage direction and "Upstream" lineage direction.

A note at the bottom of the lineage table states: "Lineage data is captured from the last 90 days".

## 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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, there's a sidebar with various icons and a search bar at the top. The main area is titled "Catalog Explorer" and shows a tree view of catalogs. Under the "Catalog" section, there are several entries: "hive\_metastore", "main", "samples", "system", "uc\_demo\_1061106" (which is expanded to show "default", "information\_schema", and "uc" subfolders, with "Tables (7)" further expanded to show "active\_promotions\_bronze", "customers\_bronze", "loyalty\_segment\_bronze", "products\_bronze", "products\_silver", "sales\_orders\_bronze", and "suppliers\_bronze"), and "Volumes (1)". A red box highlights the "catalog" icon in the sidebar.

In the center, there's a search bar with the placeholder "Search data, notebooks, recents, and more..." and a "Recent" section listing recent items like "UC Workshop Test Environment", "Import from Snowflake", and "UC Workshop Setup Environment".

The right side of the interface shows details for a specific table named "products\_silver". It has a "silver" status, a popularity rating, and was last updated 2 days ago. Below this, there are tabs for "Columns", "Sample Data", "Details", "Permissions", "History", "Lineage" (which is currently selected), and "Insights".

The "Lineage" tab displays a table of objects that influenced the "products\_silver" table. The table has columns for "Table name" and "Lineage direction". One entry is shown: "uc\_demo\_1061106.uc.products\_bronze" with an "Upstream" direction. A note below the table states "Lineage data is captured from the last 90 days".

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.

The screenshot shows the Databricks search interface with a red arrow pointing to the left margin. The search bar at the top contains the query "products". Below the search bar, there are tabs for "All", "Tables", "Notebooks", "Jobs", "Queries", and "More". The "Tables" tab is selected. The results section starts with "Tables View all tables" and lists three tables: "products\_bronze", "products\_silver", and "sales\_orders\_bronze", all from the "uc\_demo\_1061106.uc" database. Below this is a "View all" link. The next section is "Notebooks View all notebooks", which lists three notebooks: "UC Workshop Test Environment", "01 - Data Engineering with Delta", and "01.1 - Unity Catalog". Each notebook entry includes the author, last modified date, and a snippet of the code or notebook content.

Table	Author	Last Modified
products_bronze	odl_user_1061106@databrickslabs.com	modified 9/20/2023
products_silver	odl_user_1061106@databrickslabs.com	modified 9/20/2023
sales_orders_bronze	odl_user_1061106@databrickslabs.com	modified 9/20/2023

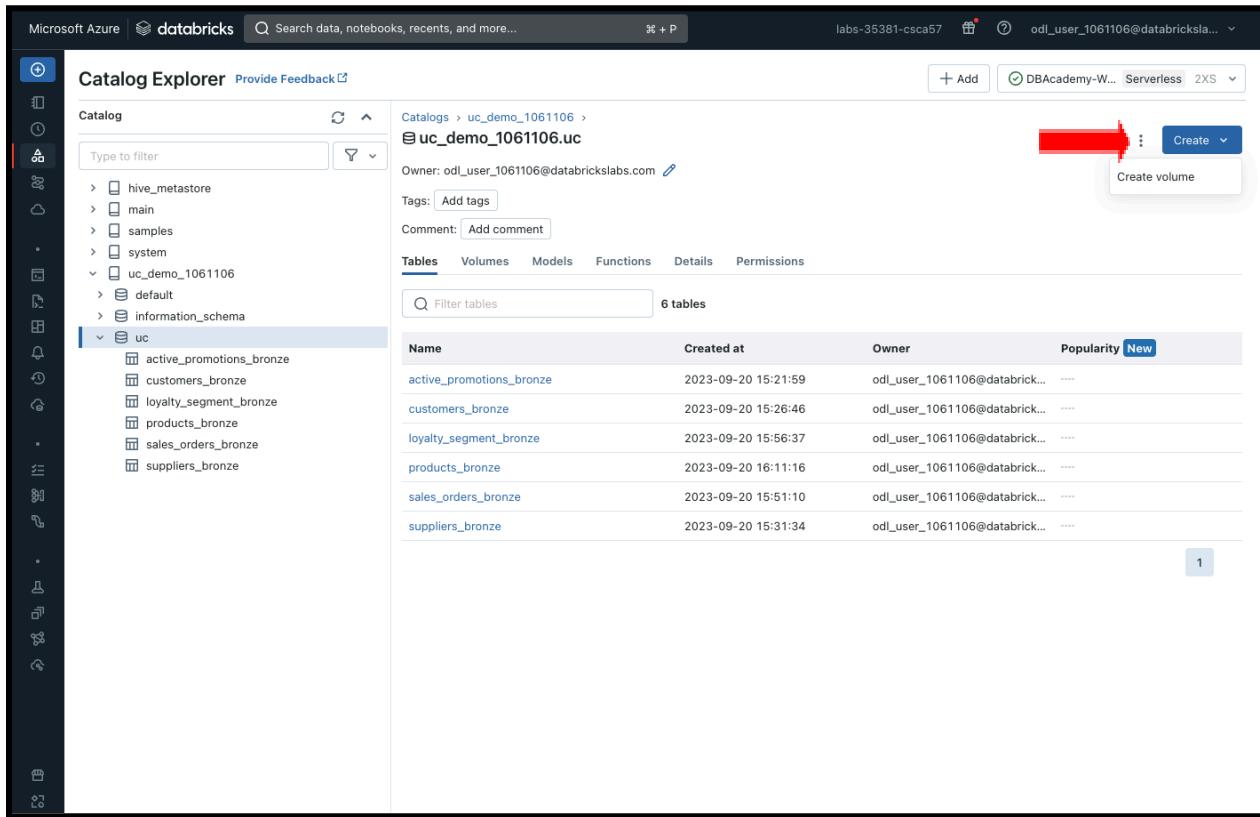
  

Notebook	Author	Last Modified
UC Workshop Test Environment	odl_user_1061106@databrickslabs.com	modified 9/20/2023
01 - Data Engineering with Delta	databricksadmin@databrickslabs.onmicrosoft.com	modified 9/8/2023
01.1 - Unity Catalog	databricksadmin@databrickslabs.onmicrosoft.com	modified 9/8/2023

## 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”:



The screenshot shows the Databricks Catalog Explorer interface. On the left is a sidebar with various icons. The main area shows a tree view of catalogs and schemas. Under the 'Catalogs' section, 'uc\_demo\_1061106' is expanded, showing 'main', 'samples', 'system', and 'uc'. The 'uc' schema is also expanded, showing tables like 'active\_promotions\_bronze', 'customers\_bronze', etc. At the top right, there is a toolbar with several buttons. A red arrow points to the 'Create volume' button, which is located next to other buttons like '+ Add', 'DBAcademy-W...', 'Serverless', and '2XS'. Below the toolbar, there is a 'Create' dropdown menu.

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

**Create a new volume**

Volume name  
pearl\_volume

Volume type  
 Managed volume  
 External volume

Comment (optional)

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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, there's a sidebar with various icons and a search bar at the top. The main area displays a catalog structure under 'Catalogs > uc\_demo\_1061106 > uc >'. A specific volume, 'uc\_demo\_1061106.uc.pearl\_volume', is selected. The volume details show it was created by 'odl\_user\_1061106@databrickslabs.com'. A file named 'catdog.jpeg' is listed in the volume, with a size of 70.75 KB and a timestamp of 'a minute ago'. A context menu is open over the file, offering options like 'Copy path', 'Download file', and 'Delete file'.

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.

The screenshot shows the Databricks Catalog Explorer interface. On the left, there's a sidebar with various icons. The main area shows a tree view of catalogs and databases. Under the 'uc' database, there are tables like active\_promotions\_bronze, customers\_bronze, loyalty\_segment\_bronze, products\_bronze, sales\_orders\_bronze, and suppliers\_bronze. A 'Volumes' section is also present. The right side of the screen displays the details for the 'uc\_demo\_1061106.uc.pearl\_volume' catalog. It shows the owner as odl\_user\_1061106@datarockslabs.com and a comment field. The 'Permissions' tab is selected, showing a table with one row:

Principal	Privilege	Object
odl_user_1061106@datarockslabs.com	ALL PRIVILEGES	uc_demo_1061106

At the bottom, there's a 'Delta Sharing' dropdown.

## Querying the Data using Databricks SQL

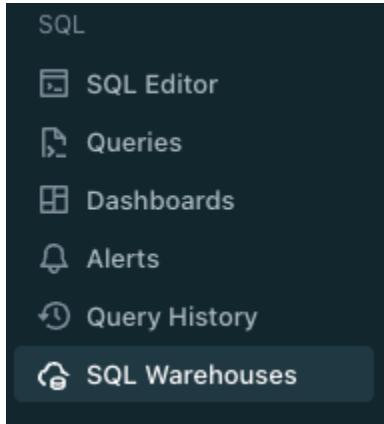
For our analyst team, they may want to query the data further to derive insights and perhaps create a dashboard. Our Data Warehousing solution Databricks SQL allows for this.

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.

The image shows the 'Compute' page in Databricks, specifically the 'SQL warehouses' tab. The page has a search bar and filters for 'Status', 'Name', 'Created by', 'Size', 'Status', and 'Type'. A single row is listed in the table:

Status	Name	Created by	Size	Active / Max	Type
Green checkmark	DBAcademy-Warehouse	databricksadmin@databrick...	2X-Small	2 / 12	Serverless

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

SQL Warehouses >

## DBAcademy-Warehouse

### Overview

### Connection details

Status	 Running
Name	DBAcademy-Warehouse (ID: b9005463b1f72198)
Type	Serverless
Cluster size	2X-Small
Auto stop	After 60 minutes of inactivity
Scaling	Cluster count: Active 2 Min 2 Max 12
Channel	 (v 2023.40)
Tags	LabId 35380
Created by	databricksadmin@databrickslabs.onmicrosoft.com

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.

### Manage permissions

Do you want information about permission levels? [Learn more](#)

Type to add multiple users or groups

psroome+trial@gmail.com	Is Owner
Admins	Can manage

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.

Total Revenue

Run (1000) main.Select schema DBAcadem... Serverless 2XS Save Schedule Share

```
1 | SELECT SUM(sales_price) AS total_revenue FROM uc_demo_1061106.uc.products_bronze
```

Results Total Product Revenue +

#	total_revenue
1	42424.28

```
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

The screenshot shows a software interface with a results table. At the top left is a blue "Results" button with a dropdown arrow, followed by a plus sign (+). Below it is a table with one row and two columns. The first column is labeled "#" and contains the value "1". The second column is labeled "total\_r" and is currently empty. A context menu is open over the "total\_r" cell, featuring three options: "Visualization" with a chart icon, "Filter" with a funnel icon, and "Parameter" with a brace icon.

#	total_r
1	

Make sure your edited visualization matches what you see below.

Total Product Revenue X

Visualization type  
10k Counter

**General** Format

Label  
Total Revenue

Value column  
1.2 total\_revenue

Row  
1

Count rows

Target column Row  
1

Cancel Save

# 42,424

## Total Revenue

### Total Product Revenue

Visualization type

10k Counter

General Format

Formatting decimal place

2

Formatting decimal character

.

Formatting thousands separator

,

Formatting string prefix

Formatting string suffix

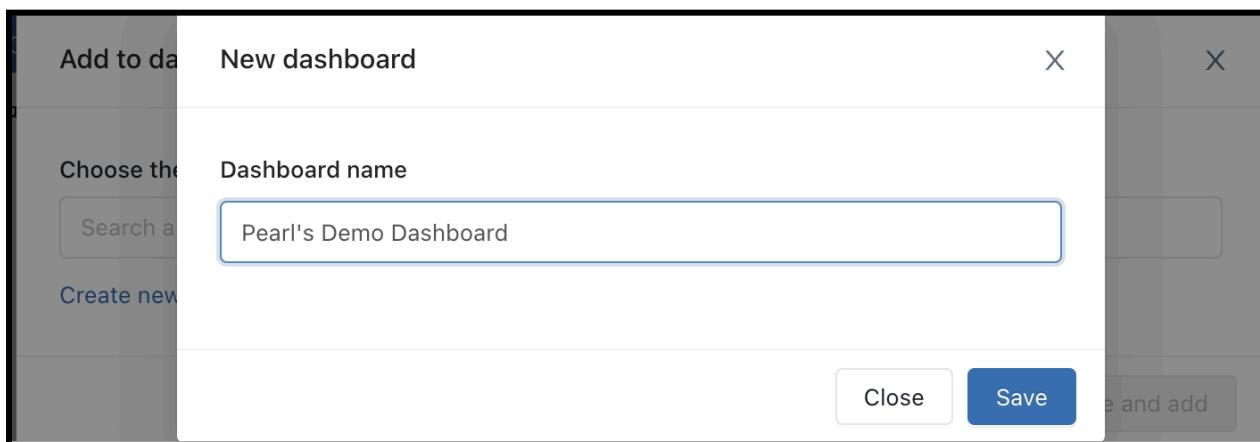
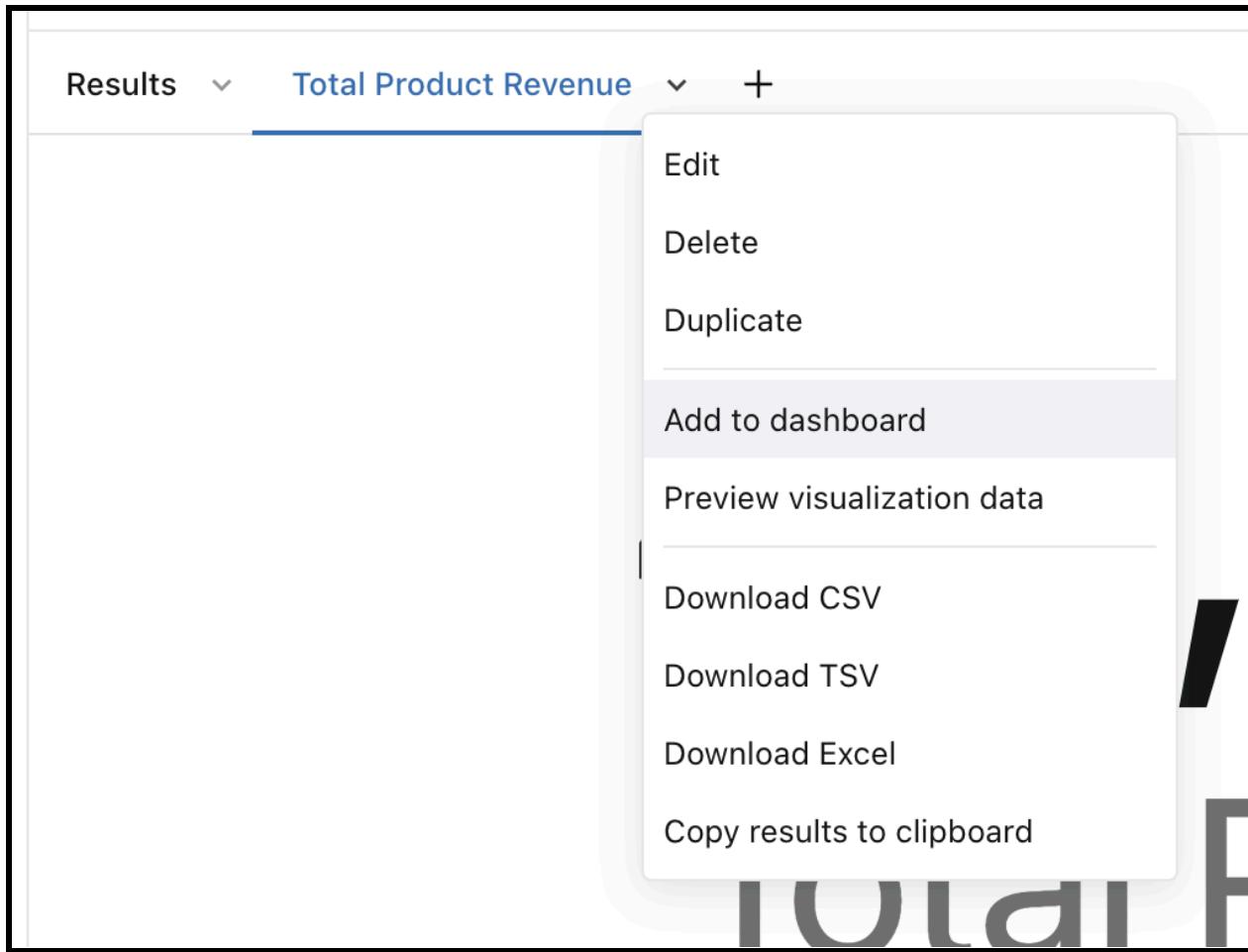
Format target value

Cancel Save

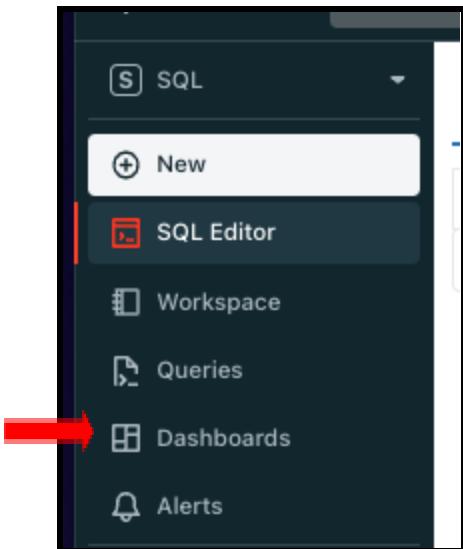
42,424.28

Total Revenue

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

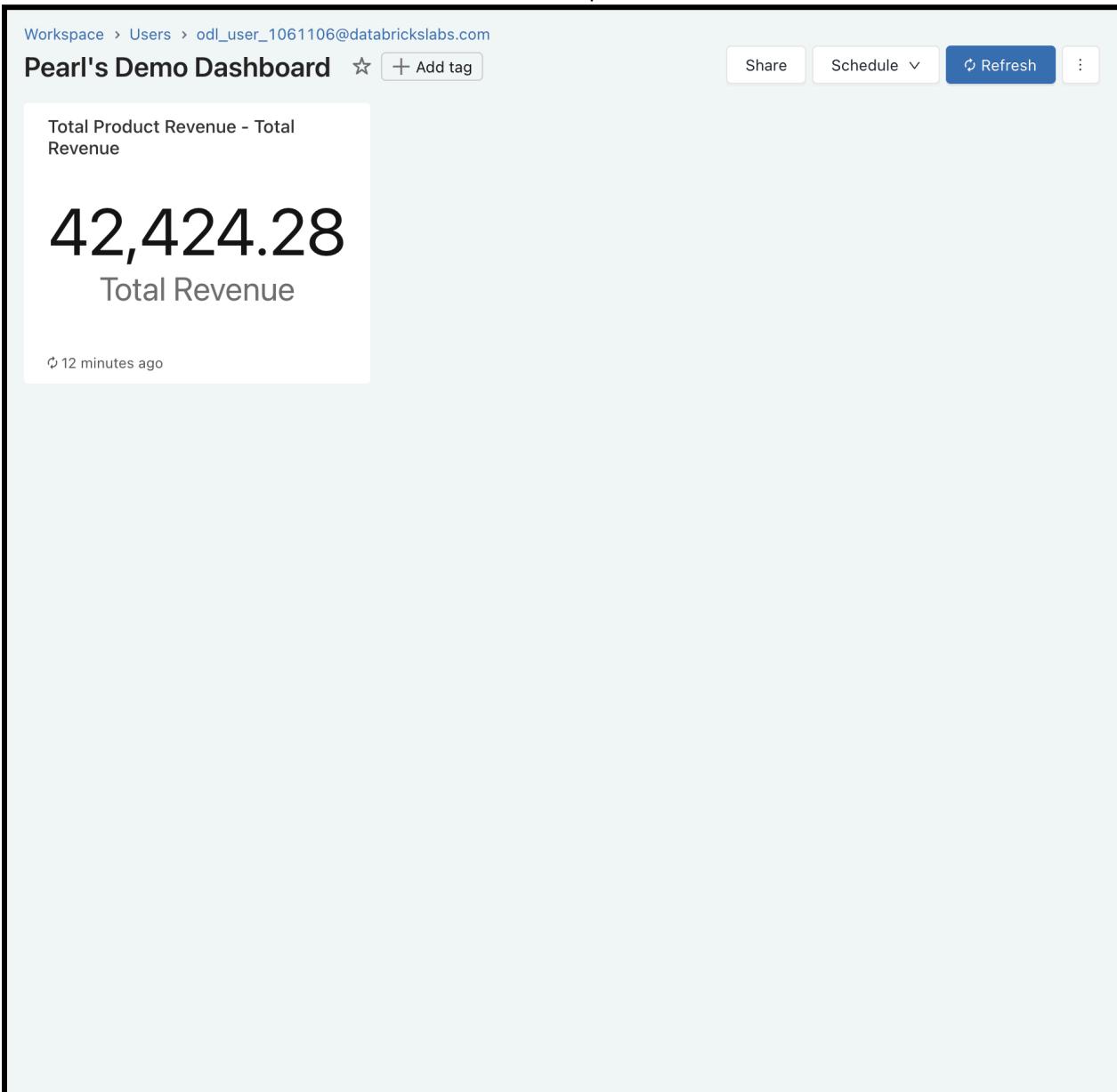


There you should see your newly created dashboard

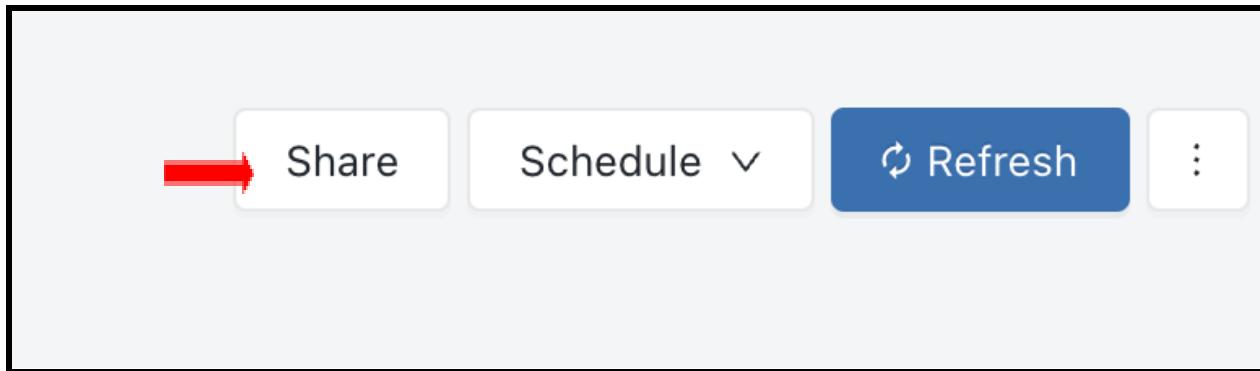
A screenshot of the Databricks Dashboards page. The page title is "Dashboards". There is one dashboard listed:

Name	Tags	Created by	Created at
Pearl's Demo Dashboard		odl_user_1061106@databri...	2023-09-26 10:16

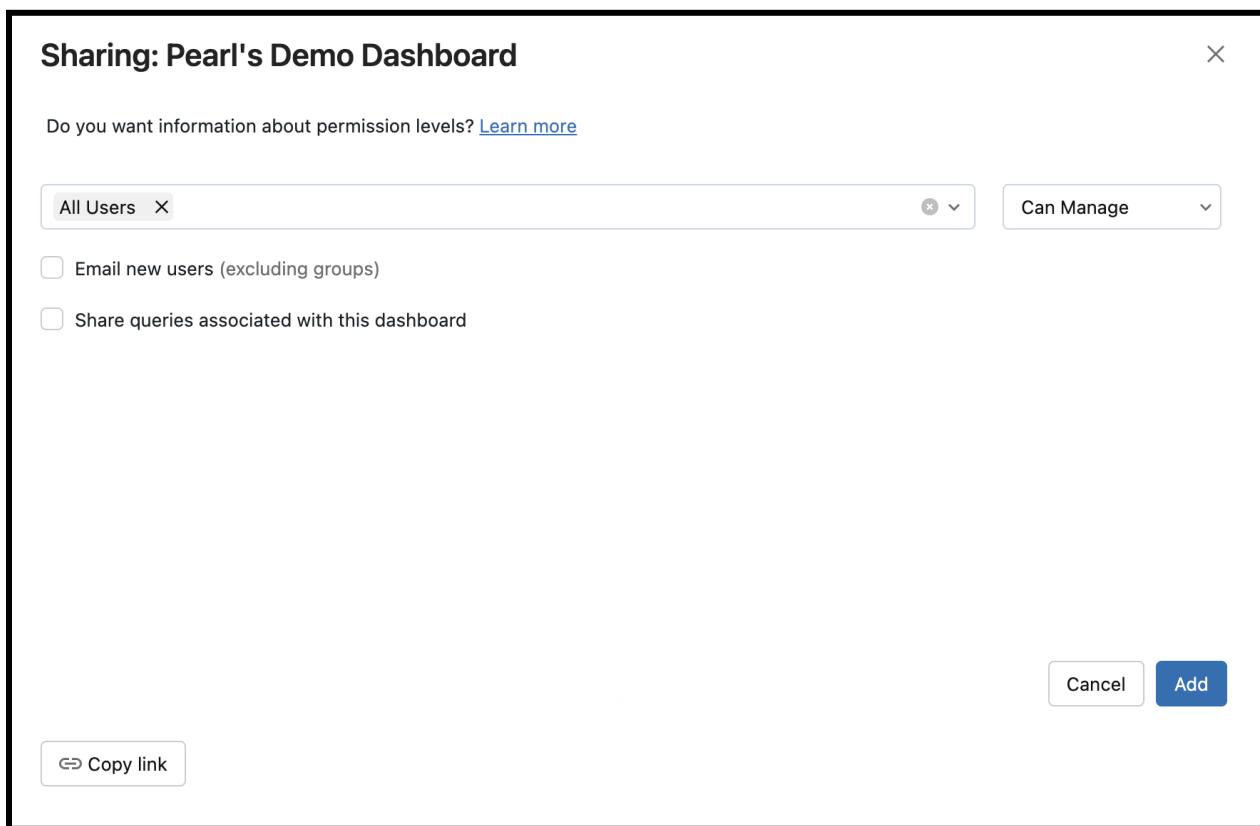
Clicking into it you will see the visualization that you created as part of your new dashboard. For the sake of time for this workshop we will not build out a full dashboard.



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 Feature 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 \*\***