



0 In this workshop we will experience specific capabilities with the Databricks Data Intelligence Platform in the areas of governance, data engineering, task orchestration, SQL analytics, and data visualization. Along the way you will experience DatabricksIQ which is a first-of-its-kind Data Intelligence Engine that uses AI to power all parts of the Databricks Data Intelligence Platform. DatabricksIQ enables natural language interfaces so we can be more productive with data.

SETUP

Prior to the workshop, work with a workspace administrator to ensure that everyone has access to the Databricks workspace and that Unity Catalog is enabled. UC is **required** for this workshop.. As an administrator, create a catalog entitled “databricks_workshop” and add permissions to it for everyone that will participate in the workshop. Give them “all privileges”.

Check connectivity to data.vermont.gov:443 before the workshop.

NOTE: The word “workspace” in Databricks has two meanings. There is the Databricks Workspace which is the UI that you log into with all components. There is also your personal workspace which is where your code for this workshop will be stored.

Have each user log into the Databricks workspace, navigate to catalog on the left pane and then to the databricks_workshop catalog. Have each user create their own schema with a naming convention that is easy to remember such as “userid_schema”.

Email the files for the workshop to each user and have them save the files into their Downloads directory. The “Welcome to the Databricks” workshop may also contain code snippets which we’ll use later as time permits.

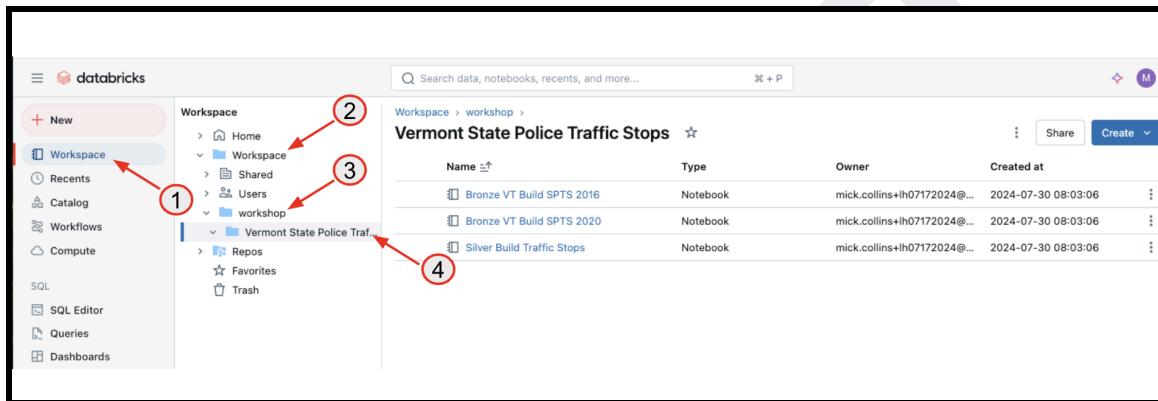
Module 1: Data Engineering and Task Orchestration

1. Each user will need to navigate to their personal workspace and create a folder entitled “workshop”. Open that folder by double clicking it and choose the 3 dots in the upper right hand corner and choose to “import”. Navigate to your Downloads folder and choose the file “Vermont State Police Traffic Stopsdbc”.

NOTE: A “dbc” is a Databricks archive and is a mechanism for sharing notebooks between workspaces and users.

2. Review existing notebooks for ingesting and transforming data from the State of Vermont’s open data portal. Follow the menu navigation to review the notebooks for ingesting the 2016, and 2020 traffic stop data, along with the notebook to consolidate the two results to a single dataset.

Workspace ① > **Workspace** ② > **workshop** ③ > **Vermont State Police Traffic Stops** ④



1. Open each of the *Bronze VT Build SPTS 2016*, *Bronze VT Build SPTS 2020*, and *Silver Build Traffic Stops* notebooks.
 - a. Notice both notebooks producing bronze data use a combination of Markdown (for documentation), SQL and Python to complete their tasks. These notebooks are defaulted to using SQL (see the default setting next to your notebook title)
 - b. Both *bronze* notebooks also receive parameters specifying which catalog and database (schema) to use when persisting data as tables using the default Delta Lake file format. Command cell 2 gets the catalog parameter, and command cell 3 gets the schema parameter.
 - c. The *silver* notebook creates a consolidated table that is a combination of both *bronze* tables while standardizing a few column names.
2. Create a new workflow, leveraging the notebooks just reviewed, to persist data in your own database (schema) using the following instructions.

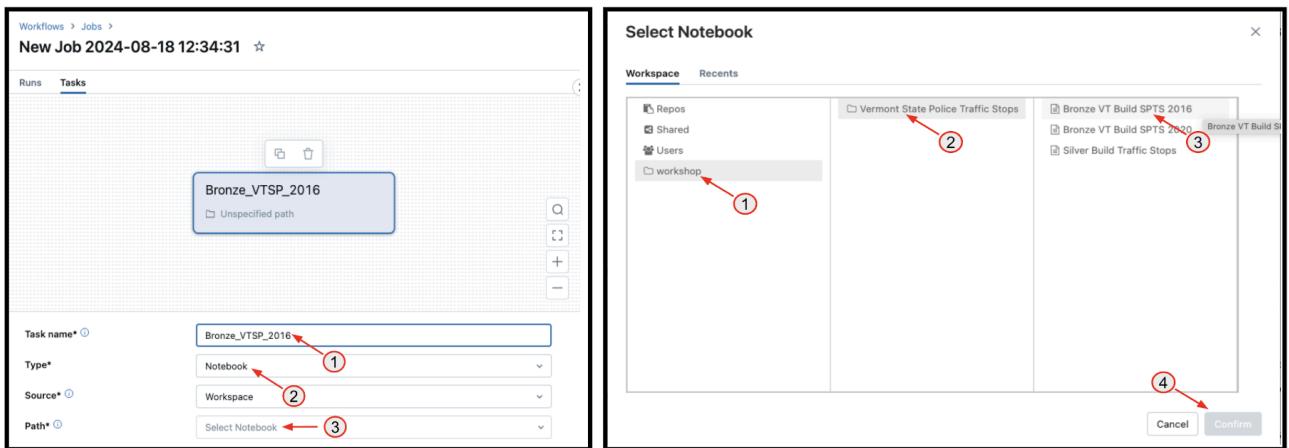
a. To create your workflow navigate to  **Workflows** primary page. Click

Create job

. The workflow canvas will have a blank task to configure.

Give the task an appropriate name, leave the Type as Notebook, and select the **Bronze VT Build SPTS 2016** notebook for this task based on what was reviewed in step 3 above, and as represented in the images below.

After filling in the task details, click [Create task].



- b. Add a second task by selecting the  button, fill in the details to reference the notebook for the 2020 data, remove the task dependency, and click 

Screenshot of the Databricks Jobs interface showing the creation of a new task.

The top navigation bar shows "Workflows > Jobs > New Job 2024-08-18 12:34:31".

The main area displays a flow diagram with two tasks:

```
graph LR; A[Bronze_VTSP_2016] --> B[Bronze_VTSP_2020]
```

The "Bronze_VTSP_2020" task has a tooltip "... Stops/Bronze VT Build SPTS 2020".

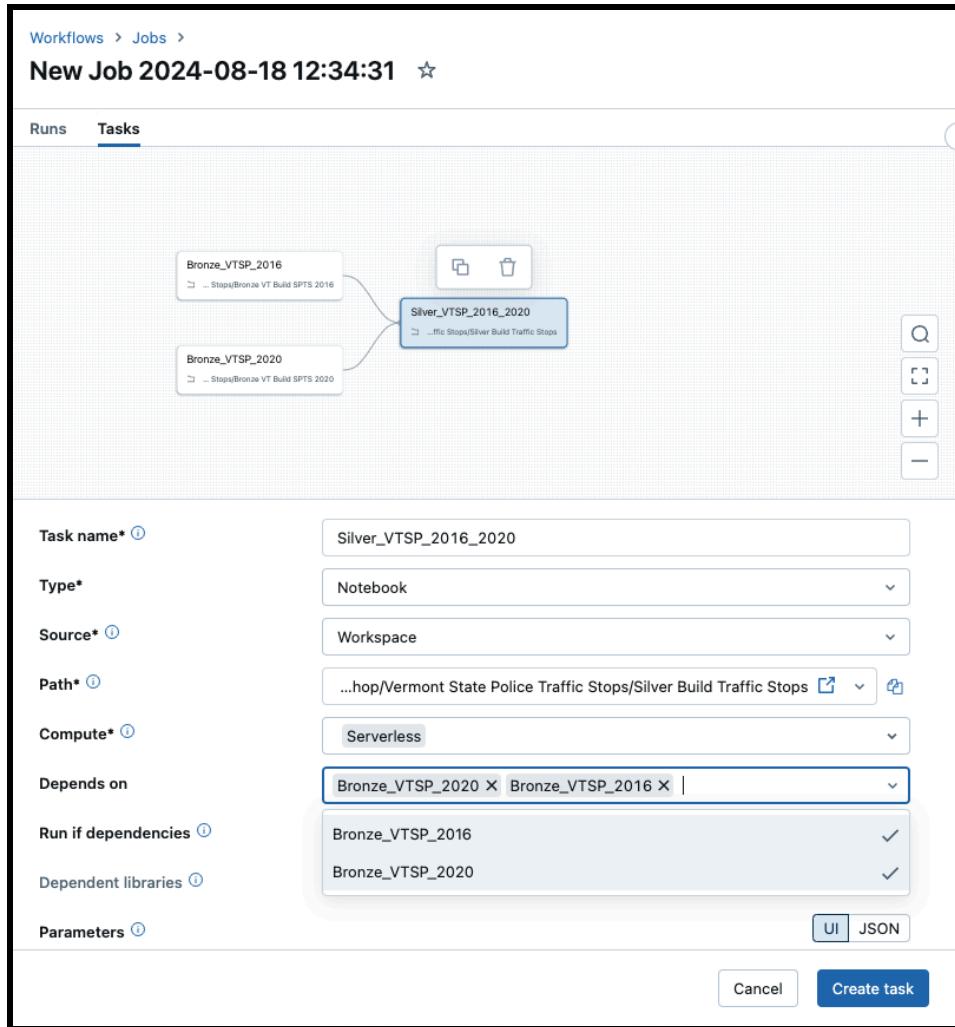
The "Tasks" tab is selected in the header.

The "Create task" dialog is open, with the following fields filled:

- Task name***: Bronze_VTSP_2020 (circled 1)
- Type***: Notebook
- Source***: Workspace
- Path***: .../Vermont State Police Traffic Stops/Bronze VT Build SPTS 2020 (circled 3)
- Compute***: Serverless
- Depends on**: Bronze_VTSP_2016 (circled 3)
- Run if dependencies**: All succeeded
- Dependent libraries**: + Add
- Parameters**: UI JSON (circled 4)

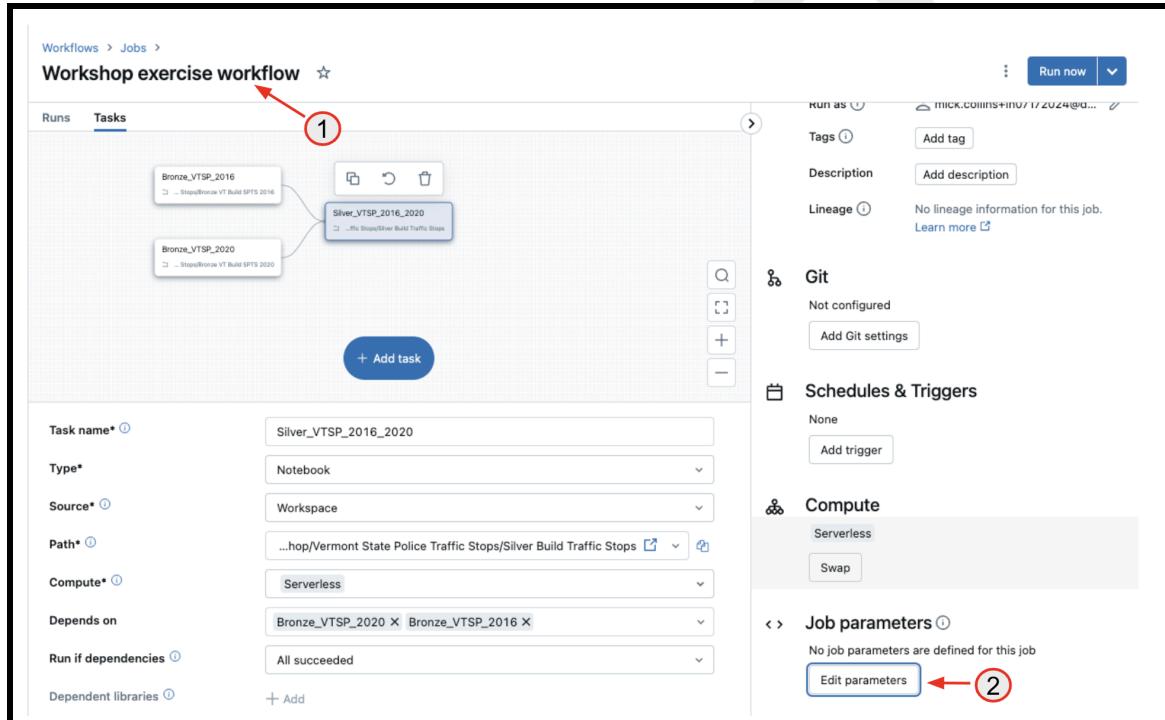
At the bottom right of the dialog are "Cancel" and "Create task" buttons.

- c. Add the final task creating the silver table, ensure the dependency for both the 2016 and 2020 tasks are selected as are represented in the images below, and click Create task.



- d. (See images below) Give your workflow an appropriate name **1**, for workshop purposes use your initials to differentiate from other participants. Also set the parameters **2** so the results will be saved in your new database.
- Important! The parameter keys must be exactly as represented in the image below as the values match the notebook parameters. The value for the catalog parameter is the “databricks_workshop” which we set up earlier. However, the value for the schema parameter should be set to **your** schema. E.g. **userid_schema**.

Save the parameter setting and click **Run now**



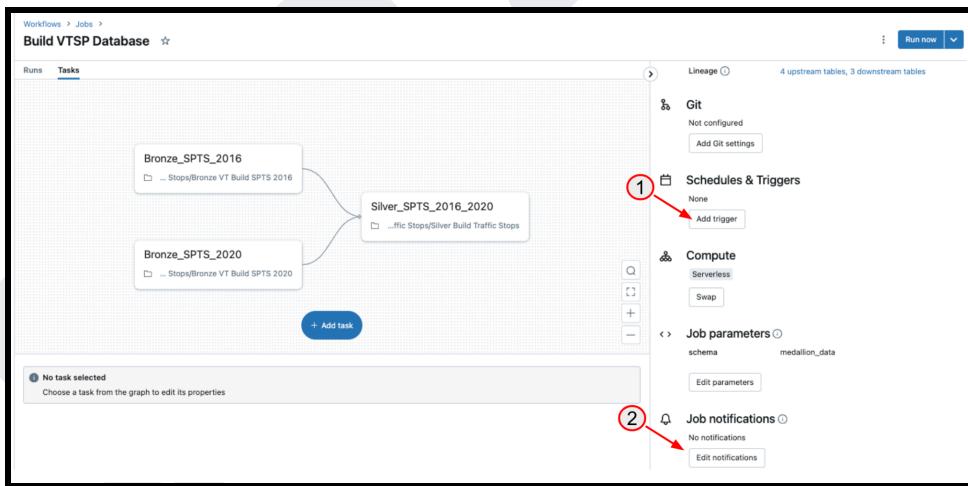
Job parameters

Add job parameters that are automatically pushed down to tasks that take key/value-based parameters. The job parameters can also be referenced explicitly in individual tasks.

Key	Value
catalog	workspace
schema	mac_medallion_data
	{}

Cancel **Save**

- e. After submitting the job for execution you can follow the progress by selecting View run, or select the run from the Workflow Runs page.
- f. Optionally, you may set a schedule for the job **(1)**, or set notifications for the job start, success, failure, or duration warning **(2)**.



Module 2: Governance and Security

1. Open Unity Catalog (Catalog) on the main menu to review your new database (schema).

You will notice that at this point, unless you are an administrator, you will only have visibility to your database, and none of the other newly created databases.

Navigate from the main menu following Catalog ① > workspace ② > *your_database* ③

The screenshot shows the Databricks Catalog interface. On the left, there is a sidebar with the following items:

- + New
- Workspace (highlighted with a red arrow)
- Recents
- Catalog (highlighted with a red circle labeled 1)
- Workflows
- Compute

Below these are sections for SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, and Search. The main area is titled "Catalog" and shows the following schema structure:

- dbdemos-shared-endpoint Serverless
- Type to filter
- demo_drop
- samples
- system
- workspace (highlighted with a red arrow)
 - default
 - information_schema
 - mac_medallion_data
 - bronze_traffic_stops_2020_view
 - bronze_vtsp_traffic_stops_2016
 - bronze_vtsp_traffic_stops_2020
 - silver_traffic_stops_2016_2020 (highlighted with a red circle labeled 3)
 - medallion_data

2. Review tables created by your workflow

Select the **silver_traffic_stops_2016_2020** table.

3. Generate table and column descriptions

Notice that the Databricks Data Intelligence Platform's AI engine automatically suggests a table description if one does not already exist.

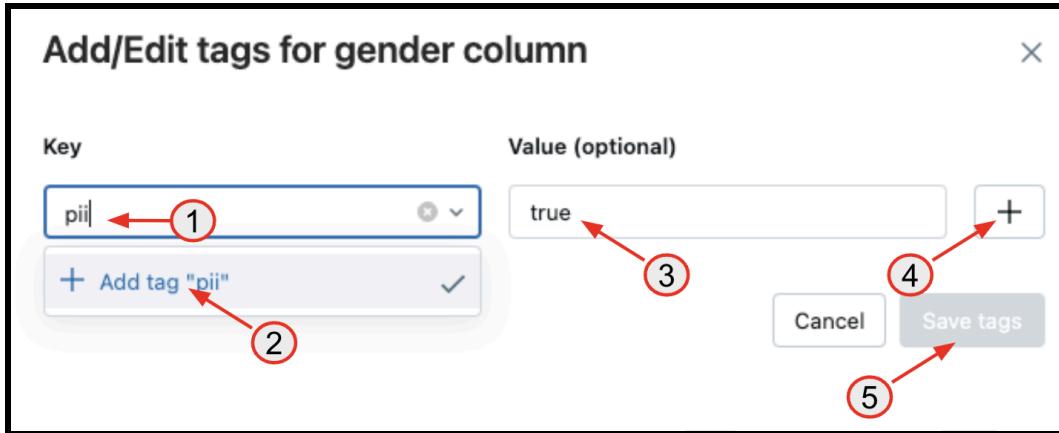
Select to generate column descriptions for the columns missing descriptions.

Accept, edit, or reject descriptions per your preference.

4. Add table and column tags

Click the button to add a tag to either the gender or race columns. Tags are a

key:value pair. Enter the Key ①. If the Key does not yet exist, select the [Add tag](#) option ②, enter the value ③, accept the setting ④, and Save tags ⑤.



5. Review each of the tabs providing access to the table metadata (Overview, Sample Data, Details, Permissions, History, Lineage, Insights, and Quality (you may be prompted to select a compute cluster to see details behind some of the tabs).
 - a. [Sample Data](#): allows you to review the data without executing a separate query.
 - b. [Details](#): is a technical view of the data including the storage location, creation timestamp, file type (Delta is the default), and etc.
 - c. [Permission](#): provides a view of current permissions for users and groups, and allows privileged users to grant access. You will grant access in step 6 below.
 - d. [History](#): displays version history for the table. You may see multiple history levels for your newly created table due to the API building the table in batches.
 - e. [Lineage](#): provides both a tabular and graphical view of both upstream and downstream lineage. Select the [See lineage graph](#) button for the graphical view. Select Show more columns for one of the tables, and select a column to see the column mapping. Table lineage was automatically captured from your workflow.
 - f. [Insights](#): show recent table activity.
 - g. [Quality](#): provides visibility to data quality for the table. This function needs to be explicitly enabled at the table level and will capture data quality over time.
6. Grant access to the '**workshop participant**' group from the [Permissions](#) tab. Refer to the image below. Table level access also requires USE SCHEMA on the

parent schema. Accept the option to also grant this privilege. As access is granted to the group you will begin having visibility to other schemas and content.



7. Add a new table (manually from CSV) following the instructions below.
 - a. Navigate back to the database overview page for your newly created schema **(1)**, click Create **(2)** > Create table **(3)**, and follow the process to import the nyc_boroughs.csv that was downloaded earlier.

- b. Change the data type of the zip_code column from Int to String by selecting the button, and then *string*. Save the table **Create table**.

Module 3: SQL Analytics

1. Open the new table in SQL

From the table overview page create a query of this new table by selecting the

Create button ①, and Query ②. Notice that you will navigate to the **SQL Editor** screen. Select Run if you want to see the results of the **Select * command**.

Catalog Explorer > workspace > mac_medallion_data > **nyc_boroughs**

Column	Type	Comment	Tags	Column mas...
uhf_code	bigint			
borough	string			
neighbo...	string			
zip_code	string			

About this table
Owner: mic
Data source: Genie space
Popularity: 18
Last updated: 18 hours ago
Size: 3.7KB, 1 file

Tags
Add tags
Row filter

2. Open Databricks Assistant to modify this query as is represented in the image below.

Run (1000) workspace. mac_medallion_data

```
1 SELECT * FROM nyc_boroughs
```

- Type **how many zip codes by borough** in the dialog box ①, hit return, accept the results ②, and run ③ the command.

Run (1000) workspace. mac_medallion_data

how many zip codes by borough ①

```
1 SELECT * FROM nyc_boroughs
```

3. Create a new SQL statement

We will create a query using a Common Table Expression (CTE)

- Open a new tab in the SQL Editor by selecting the + icon.
- On your workstation, open the CTE_query.txt document. Copy the full contents (^A, ^C). Paste the text in the new query tab.
- In the CTE query replace **MY_SCHEMA** with the name of the schema you created. This is on two lines (lines 15, and 17). Run the query and review the results.
- Format the query to improve readability by selecting the kabob menu ①, and **Format query** ②

```

3 | hour(tpipe_pickup_datetime) as trip_hour,
4 | date(tpipe_pickup_datetime) as trip_date,
5 | cast(date_format(tpipe_pickup_datetime, 'yyyyMM'
6 | as trip_month,
7 | year(tpipe_pickup_datetime) as trip_year,
8 | date_format(tpipe_pickup_datetime, 'EEEE') as
9 | trip_day_of_week,
10 | b.borough as pickup_borough,
11 | b.neighborhood as pickup_neighborhood,
12 | c.borough as dropoff_borough,
13 | c.neighborhood as dropoff_neighborhood
14 | from samples.nyctaxi.trips a
15 | join workspace.mac_medallion_data.nyc_boroughs
16 | on a.pickup_zip = b.zip_code
17 | join workspace.mac_medallion_data.nyc_boroughs
18 | on a.dropoff_zip = c.zip_code),
19 | trips_agg as (
20 | select

```

4. Save query

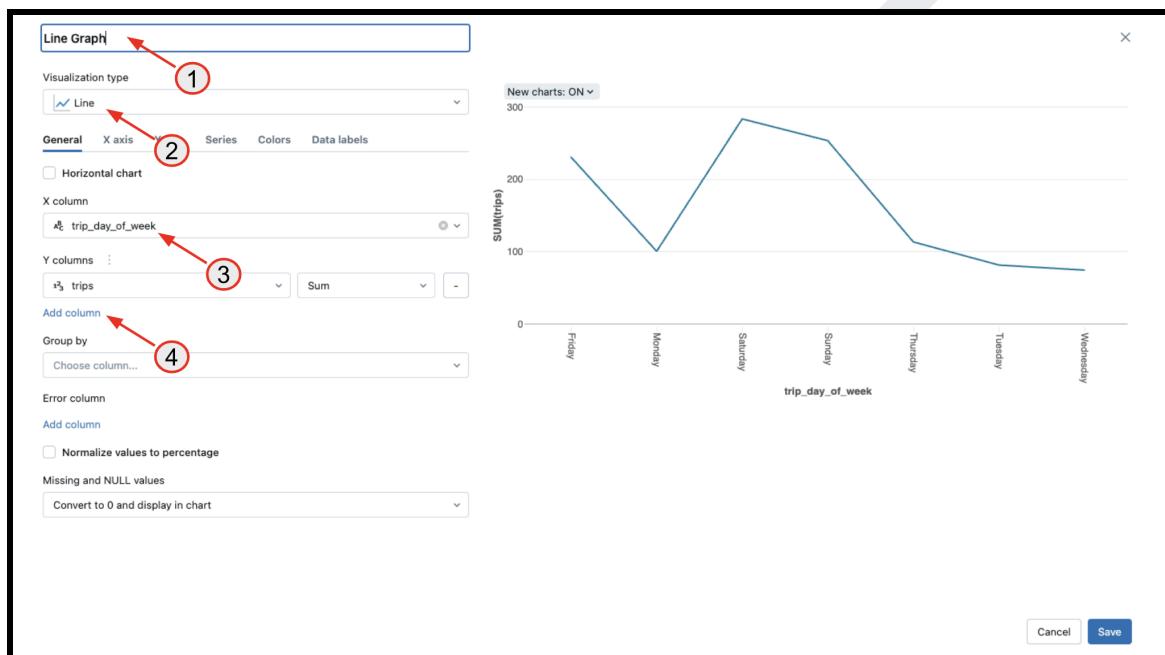
Double click **New query** in the query tab, and give your query a name, and click

Save*

Your query will be saved in your default workspace folder. You can see the results by selecting **Workspace** in the main menu.

5. Add visualization

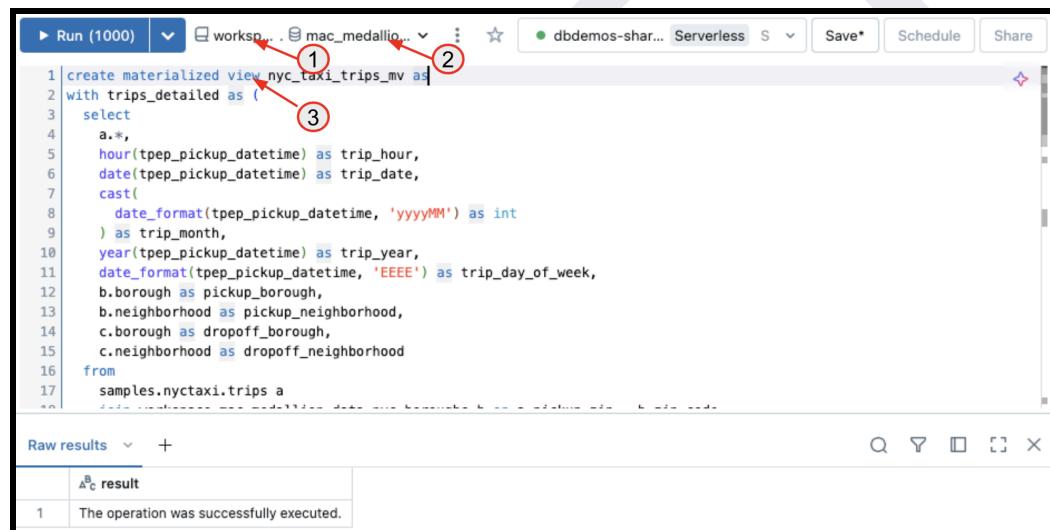
Add a visualization to this query. Select and Visualization in the query results pane. Provide a title **1**, set the visualization type as **Line** **2**, select `trip_day_of_week` for the X column **3**, add a column for the Y axis and set the value to `trips` **4**.



6. Schedule query (optional)

Click to schedule your query.

7. Create a materialized view from the CTE query
 - a. Select and Copy the entire query text
 - b. Create a new query by selecting  on the query tabs.
 - c. Set the catalog **1** and database (schema) **2** to reference your database.
 - d. Paste the CTE as the new query.
 - e. Add a new line at the top of the query with the following text:
create materialized view nyc_taxi_trips_mv as **3**
 (this text is also in the mv.txt file from the workshop resources).
 - f. Select  to execute the CREATE statement (this operation will take approximately 90 seconds)



The screenshot shows the Databricks SQL interface. A query is being run in a notebook cell:

```

1 create materialized view nyc_taxi_trips_mv as
2 with trips_detailed as (
3 select
4   a.*,
5   hour(tpep_pickup_datetime) as trip_hour,
6   date(tpep_pickup_datetime) as trip_date,
7   cast(
8     date_format(tpep_pickup_datetime, 'yyyyMM') as int
9   ) as trip_month,
10  year(tpep_pickup_datetime) as trip_year,
11  date_format(tpep_pickup_datetime, 'EEEE') as trip_day_of_week,
12  b.borough as pickup_borough,
13  b.neighborhood as pickup_neighborhood,
14  c.borough as dropoff_borough,
15  c.neighborhood as dropoff_neighborhood
16  from
17  samples.nyctaxi.trips a

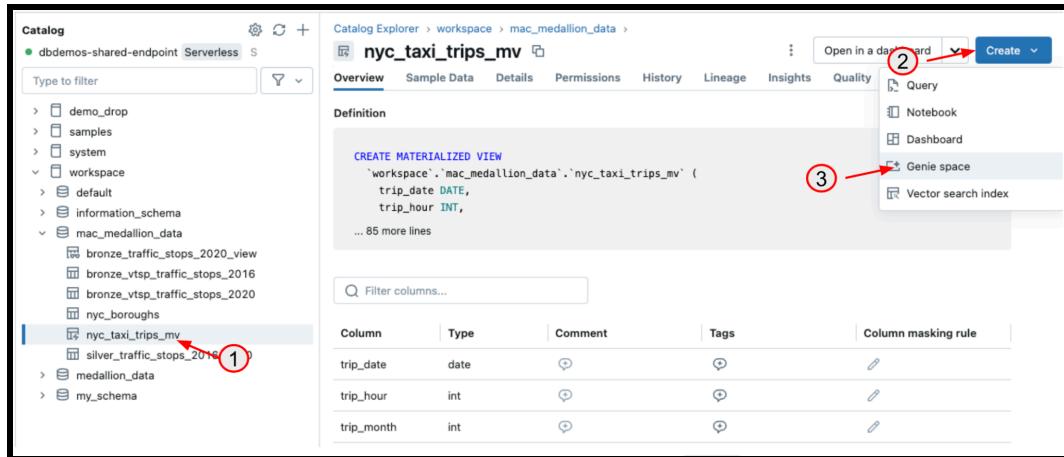
```

The interface includes a toolbar with 'Run (1000)', workspace dropdown, catalog dropdown, and other buttons like 'Save*', 'Schedule', and 'Share'. The catalog dropdown has '1' circled, and the schema dropdown has '2' circled. The line number '3' is circled in the query text. Below the cell, the results pane shows:

Raw results	+ 
1	The operation was successfully executed.

Module 4: Converse with the data by creating a Genie Space

1. In Unity Catalog, navigate to the schema where you created your materialized view (MV) **(1)**
2. Select the MV **(2)**, and create a Genie Space **(3)**



3. We will minimally configure the Genie Space. Provide a title ①, select the default warehouse as the compute ②, the MV is already included ③. Optionally you can add other related tables (skip this), and Save ④

New
Create a new space by giving it a name and define which tables make sense for the use case.

Title
MAC Workshop Genie Space ①

Description
Describe what data is available in this space and what type of questions users can ask.

Default warehouse
[Running] dbdemos-shared-endpoint ②

Tables
Choose tables to use for answering questions in the space. It is best to keep the scope for each space as small as possible. Data access is governed with the viewers Unity Catalog permissions.

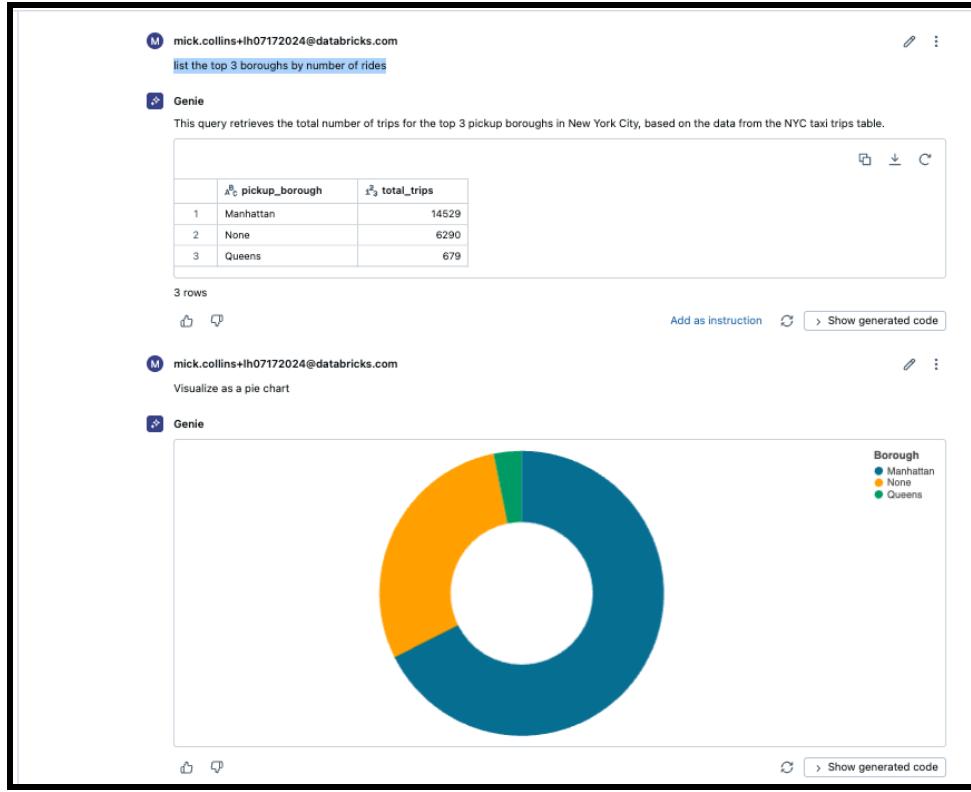
Catalog	Schema	Table
workspace.mac_medallion_data	nyc_trips	mv ③

Sample questions
Sample questions will be presented in new chat windows for users to ask the Space.
E.g. What is our annual revenue? ④

Cancel Save

4. Ask a question of the data (e.g. *list the top 3 boroughs by number of rides*)

5. Graph the results as a pie chart (click the Pie Chart button)



6. Provide instructions for Genie to understand terms that may be organizationally specific, such as fiscal year, etc. For our example we will tell Genie that **a lot of rides is more than 200**. (Use 200, not 500 as in the image below)

Select the Instructions button . Enter the instruction and Save

General Instructions
Add general instructions on how you want Genie to behave.

a lot of rides is more than 500

Discard **Save**



7. Return to the chat  and ask a new question, such as **which boroughs had a lot of rides**

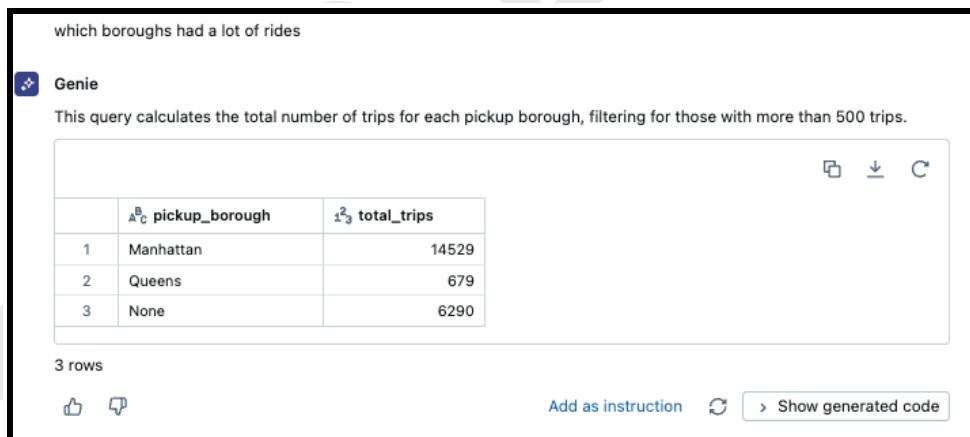
which boroughs had a lot of rides

Genie
This query calculates the total number of trips for each pickup borough, filtering for those with more than 500 trips.

	pickup_borough	total_trips
1	Manhattan	14529
2	Queens	679
3	None	6290

3 rows

  Add as instruction  > Show generated code



8. Notice, you can show the generated code if desired

Module 5: Data Visualization

1. Return to your MV in Unity Catalog
2. Use the **Create** button, and select Dashboard
3. Use Databricks Assistant to add a pie chart to the dashboard as represented in the image below ①.

The screenshot shows the Databricks Assistant interface for creating a dashboard. At the top, it says "Dashboard for nyc_taxi_trips_mv table". Below that is a text area: "The dashboard you created for workspace.mac_medallion_data.nyc_taxi_trips_mv is ready! The widget to the right of this one allows you to generate a chart based on this dataset using natural language. Click the widget to start typing a description of the chart you'd like to see. Alternatively, you can edit any selected widget using the". To the right, there's a text input field with "add a pie chart of the top 5 boroughs by number of rides" and a red circle with the number 1 above it. Below the input field are two buttons: "Average Distance by Pickup Borough" and "Total Fare by Dropoff Borough". A large table below the text area shows taxi trip data from January 2016.

trip_date	trip_hour	trip_month	trip_year	trip_day_of_week	pickup_borough	pickup_neighborhood
2016-01-01	0	201601	2016	Friday	Brooklyn	Bedford Stuyvesant - Crown Heights
2016-01-29	0	201601	2016	Friday	Brooklyn	Downtown - Heights - Park Slope
2016-01-01	0	201601	2016	Friday	Brooklyn	Downtown - Heights - Park Slope
2016-01-22	0	201601	2016	Friday	Brooklyn	Downtown - Heights - Park Slope
2016-01-22	0	201601	2016	Friday	Brooklyn	Greenpoint
2016-01-01	0	201601	2016	Friday	Brooklyn	Williamsburg - Bushwick
2016-01-01	0	201601	2016	Friday	Manhattan	Central Harlem - Morningside Heights

This screenshot shows the same Databricks Assistant interface as the previous one, but now with a pie chart added to the dashboard. The pie chart is titled "Top 5 Boroughs by Number of Rides" and is labeled "Sum of trips". It includes a legend for pickup_borough: Manhattan (blue), None (orange), Queens (green), Brooklyn (red), Bronx (light blue), and Staten Island (dark red). The dashboard title is "Dashboard for nyc_taxi_trips_mv table". The table below shows the same taxi trip data as the first screenshot.

trip_date	trip_hour	trip_month	trip_year	trip_day_of_week	pickup_borough	pickup_neighborhood
2016-01-01	0	201601	2016	Friday	Brooklyn	Bedford Stuyvesant - Crown Heights
2016-01-29	0	201601	2016	Friday	Brooklyn	Downtown - Heights - Park Slope
2016-01-01	0	201601	2016	Friday	Brooklyn	Downtown - Heights - Park Slope
2016-01-22	0	201601	2016	Friday	Brooklyn	Downtown - Heights - Park Slope
2016-01-22	0	201601	2016	Friday	Brooklyn	Greenpoint
2016-01-01	0	201601	2016	Friday	Brooklyn	Williamsburg - Bushwick
2016-01-01	0	201601	2016	Friday	Manhattan	Central Harlem - Morningside Heights

Module 6: Connecting partner tools (optional)

1. Select Partner Connect at the bottom of the main menu ①.

The screenshot shows the Databricks homepage. On the left, there is a sidebar with various navigation options: Dashboards, Genie, Alerts, Query History, SQL Warehouses, Data Engineering (Job Runs, Data Ingestion, Delta Live Tables), Machine Learning (Playground, Experiments, Features, Models, Serving), Marketplace, and Partner Connect (marked with a red circle and arrow). The main area is titled "Welcome to Databricks" and features a search bar, three circular buttons ("For you", "Mosaic AI", "What's new"), and three cards: "Easily compare DBRX and other LLMs head-to-head", "Explore how AI accelerates Dashboard creation", and "Converse with your data using natural language". Below these is a "Quick access" section with "Recents", "Favorites" (marked with a red circle), and "Popular" buttons. A large watermark "DRAFT" is visible across the page.

2. Explore the options for leveraging external tooling for data visualization, and other categories.

The screenshot shows the Databricks Partner Connect interface. On the left, there's a sidebar with navigation links for Workspace, Recents, Catalog, Workflows, Compute, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, Playground, Experiments, Features, Models, Serving, Marketplace, and Partner Connect. The main area displays a grid of partner cards:

- Rivery**: Provides a cloud-native ELT+ platform that accelerates the entire Databricks workflow through data ingestion, transformation, orchestration, and reverse ETL.
- rudderstack**: A lakehouse native customer data platform that helps data teams collect, unify, and activate first party data securely.
- SNOWPLOW**: Creates 1st party customer data in real time from web, mobile, IoT & serverside apps into a raw events table and a series of modelled tables.
- BI and visualization** section:
 - Microsoft Power BI**: Supports all users. Quickly find meaningful insights within your data and easily build rich, visual analytic reports.
 - Tableau**: Supports all users. Helps people see and understand data with the world's broadest and deepest analytics platform.
 - Hex**: A modern Data Workspace for teams. Analysts and Data Scientists can explore, analyze, and visualize in collaborative data notebooks, and then share their work as interactive apps and stories that anyone can use.
- preset**: A free tool to quickly build interactive charts and dashboards on top of your data workspace. Preset (powered by Apache Superset) is the visualization layer for your modern data stack.
- Qlik**: Sense delivers best-in-class cloud analytics that help people of all skill levels to make data-driven decisions and take action.
- ThoughtSpot**: The Modern Analytics Cloud enabling answers and insights through natural language search and AI for all users drawing on all data from the Lakehouse.
- sigma**: Unlocks the value of data by delivering cloud-scale analytics to empower organizations with data-driven insights.