Module 5: Create a DLT Pipeline

The purpose of this module is to explain when to use Delta Live (Streaming) Tables vs Workflows and notebooks. Workflows based on Notebooks are great for static tables or full refreshes or data that doesn't change often. DLT (Streaming) is best for data that changes often. I.e. new data daily or new upserts/deletes daily.

A Delta Live Table is purpose built for data that will change often. Examples include:

- Streaming data such as IOT Sensor data
- CDC files which land in an object store and need to be upserted or deleted in Silver Tables

This DLT workshop will focus on demonstrating how a volume, which is monitored by Databricks' Autoloader, will only process new files as they land.

NOTE: This Workshop is an add-on for the Introduction to Databricks workshop. If performing this workshop outside of that workshop, ensure that there is a catalog named "databricks_workshop" or similar with permissions set to "All Privileges" so users can create schemas.

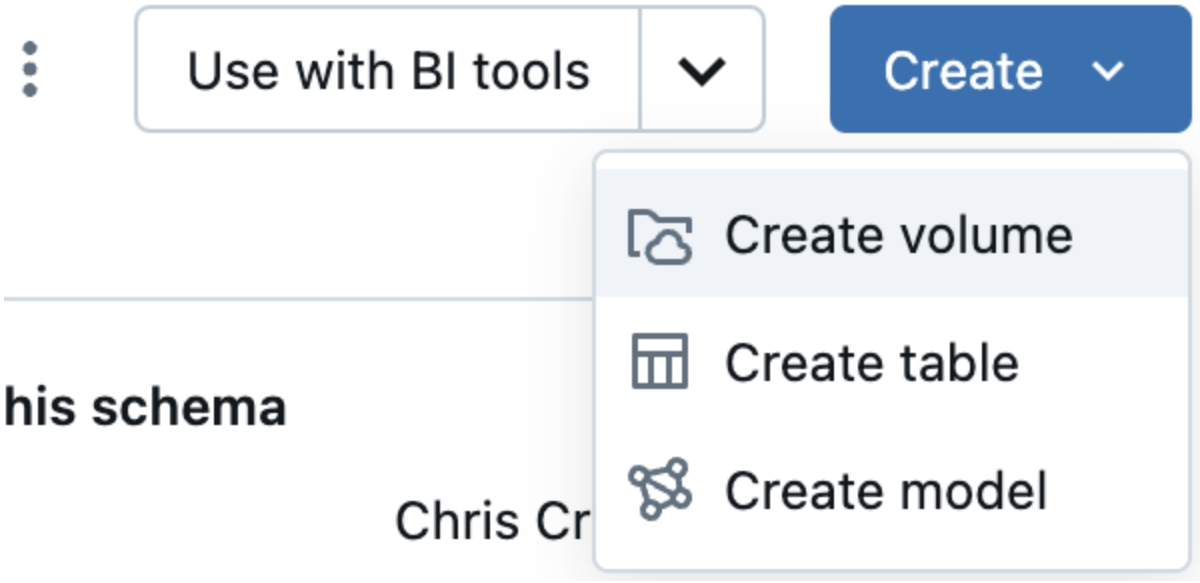
Have each user create a schema that is easy for them to remember in this catalog.

NOTE: The material for this workshop can be found in the following github: https://github.com/chriscrawfordAL/databricks-workshop-material. Recommend having an admin add this git repo to the Shared workspace so all workshop members can download the datasets used.

It is best to have participants download all material from the github to their local machine before beginning the workshop.

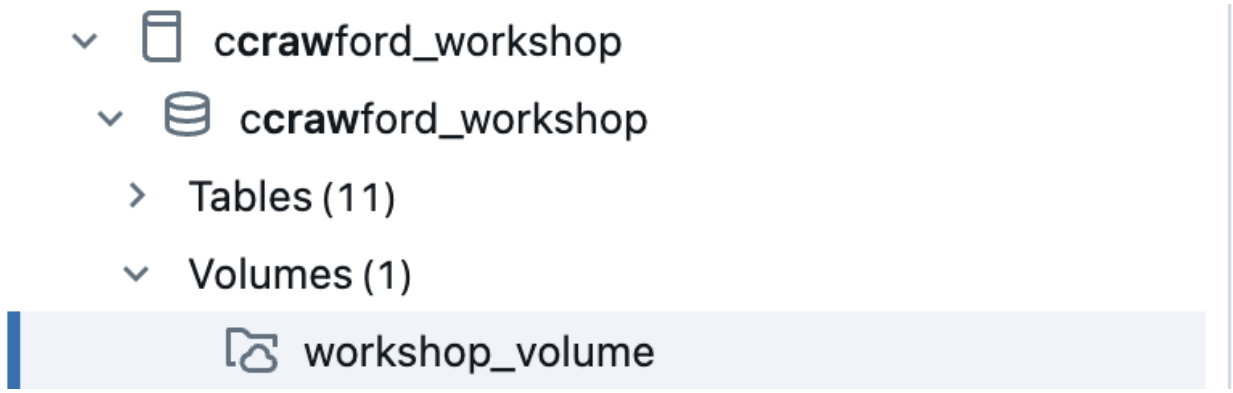
To begin, create a new shared cluster. Have the administrator persona, create one shared cluster which we will use later. Leave the cluster at terminate after 120 minutes. We will terminate at the end of this exercise.

Each user should create a volume in the workshop catalog under their personal schema, choose to "CREATE" and "Create Volume"



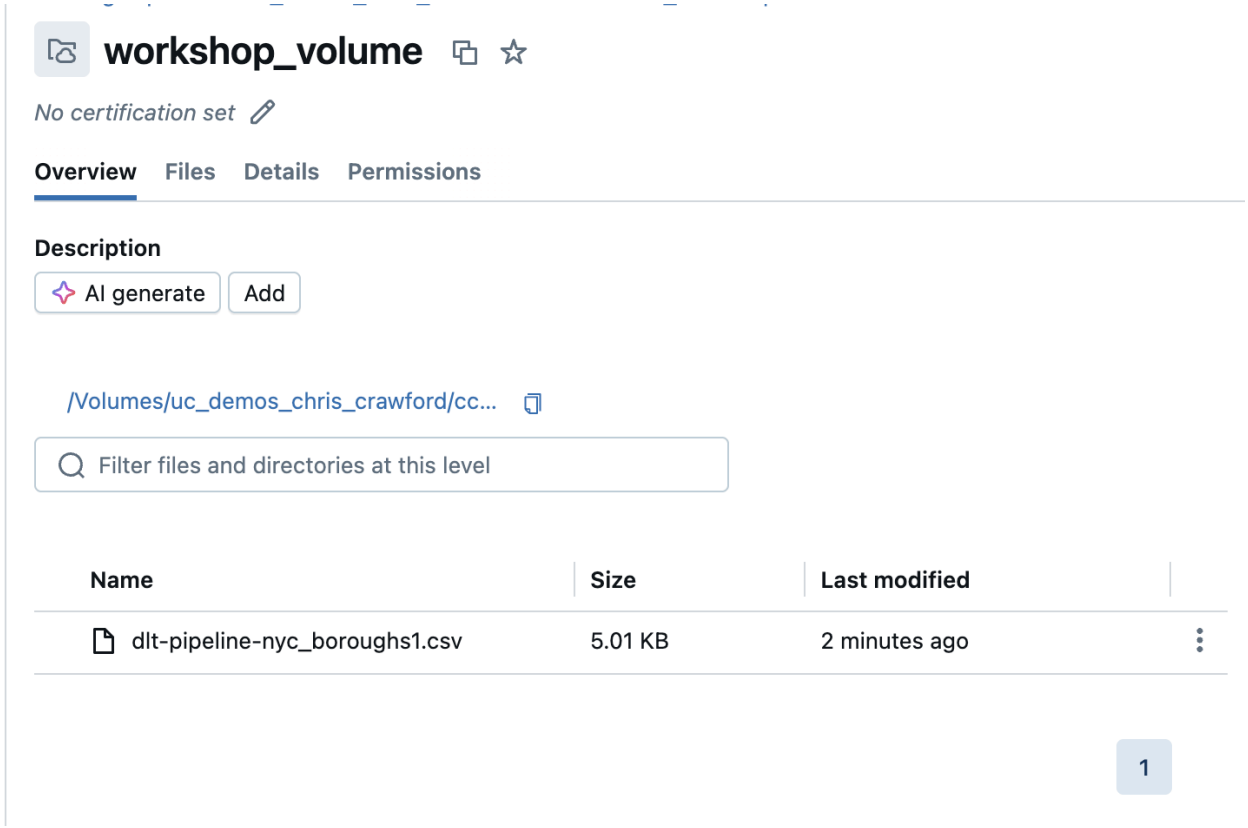
Provide a name for the volume based on the naming convention you have been using.

Once the volume is created, select the volume

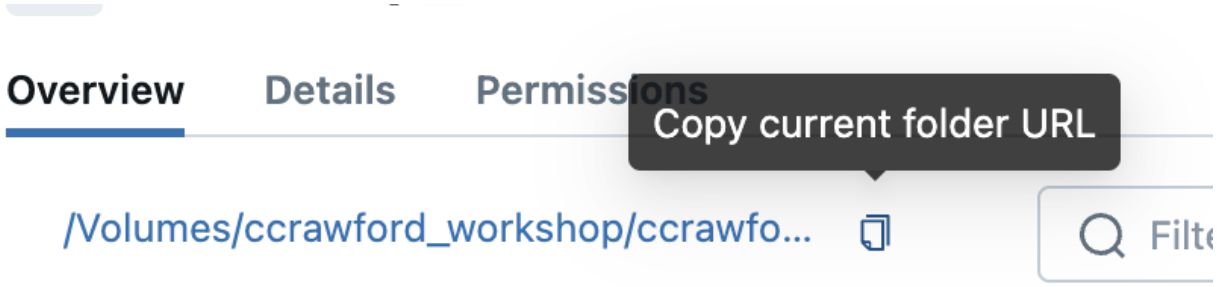


And choose to “Upload to this Volume”

Browse and upload the nyc_boroughs1.csv file (**don’t upload nyc_boroughs2.csv, we’ll do that later**).



Copy the full path of the volume using the “copy” icon



Navigate back to your workshop folder in your workspace and import the “DLT Pipeline.dbc” file.

Leave Cell 1 commented out. This cell is to demonstrate that Volumes act like file storage (even though they are “technically” not directories)

Replace the volume in cell 1 and cell 2 with the volume that you copied to your clipboard earlier.

```
#%fs ls '/Volumes/uc_demos_chris_crawford/ccrawford_workshop/workshop_volume/dlt-pipeline-nyc_boroughs1.csv'
```

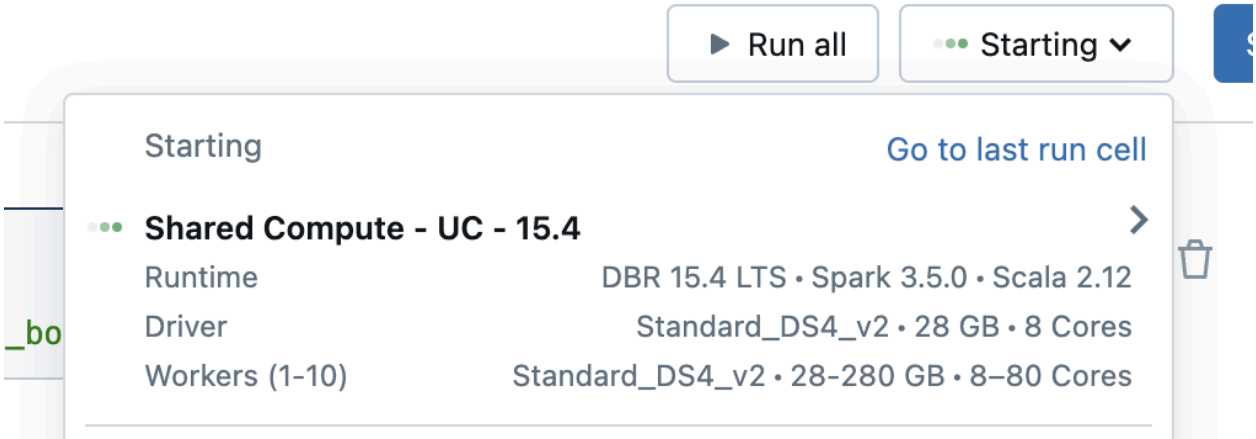
```
import dlt
import pyspark.sql.functions as f

@dlt.table(table_properties={'quality':'bronze'})
def bronze_nyc_boroughs():
  return (
    spark.readStream.format('cloudFiles')
    .option('cloudFiles.format', 'csv')
    .option('cloudFiles.inferColumnTypes', 'true')
    .option('header', 'true')
    .load(f'/Volumes/uc_demos_chris_crawford/ccrawford_workshop/workshop_volume/')
    .withColumn('metadata', f.col('_metadata'))
  )
```

Do not include the filename in Cell 2. Explain that “Cloudfiles” is Databrick’s Autloader and it must “glob”. You *could* do *.csv, but it must be a glob, not an individual file. Explain that Autoloader will keep track of what files have been processed and which ones have not.

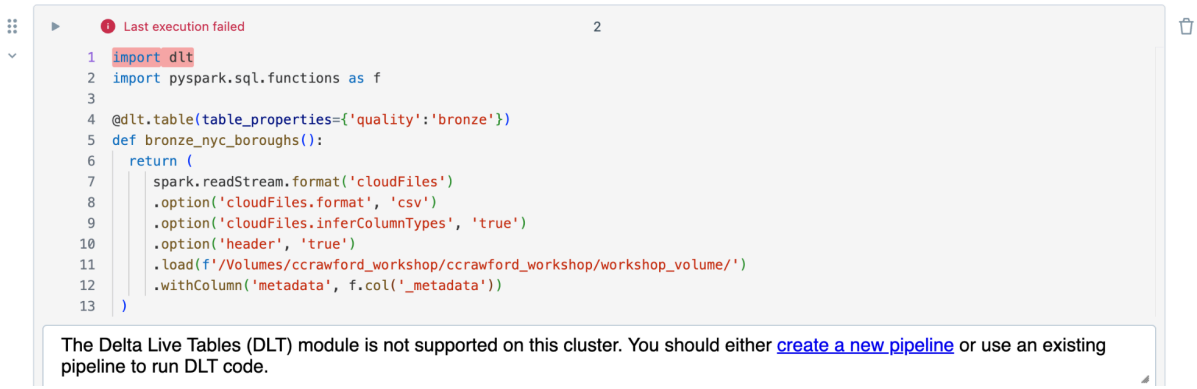
```
    .load(f'/Volumes/uc_demos_chris_crawford/ccrawford_workshop/workshop_volume/')
```

We are now going to demonstrate that a DLT Pipeline **cannot** be run on an all-purpose cluster.

In the notebook, attach to the Shared Compute created earlier in this workshop

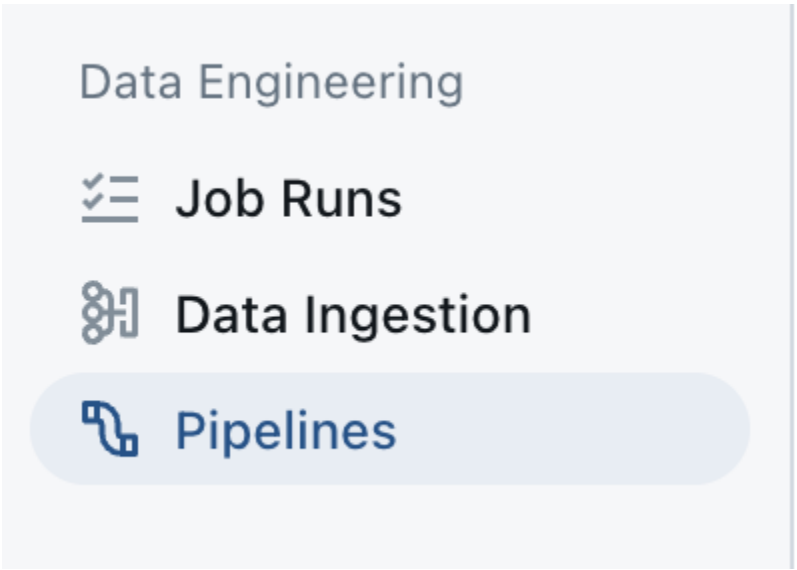


Attempt to run cell 2. Note that it fails with the following error:



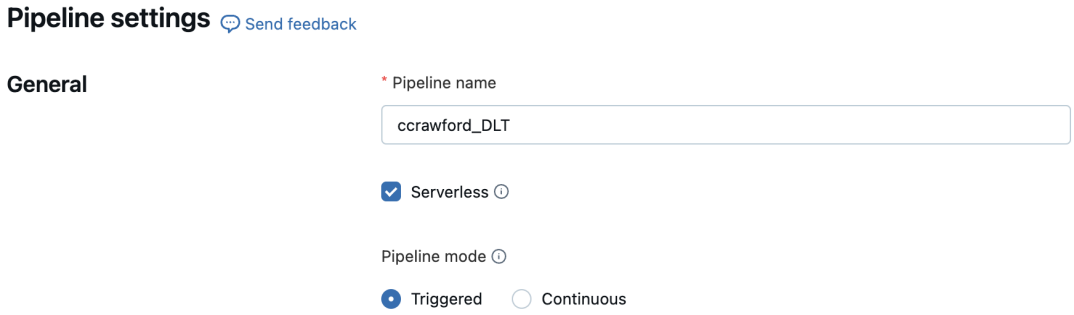
A notebook with DLT code/syntax/decorator will **not** run on an all-purpose computer, you must have a DLT *Pipeline*.

On the menu bar on the right, choose “Pipeline” or “Delta Live Table” (depending on your UI version) and create a new pipeline.

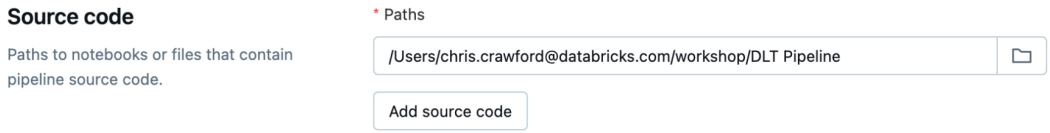


When you create the pipeline, you will be presented with the pipeline configuration.

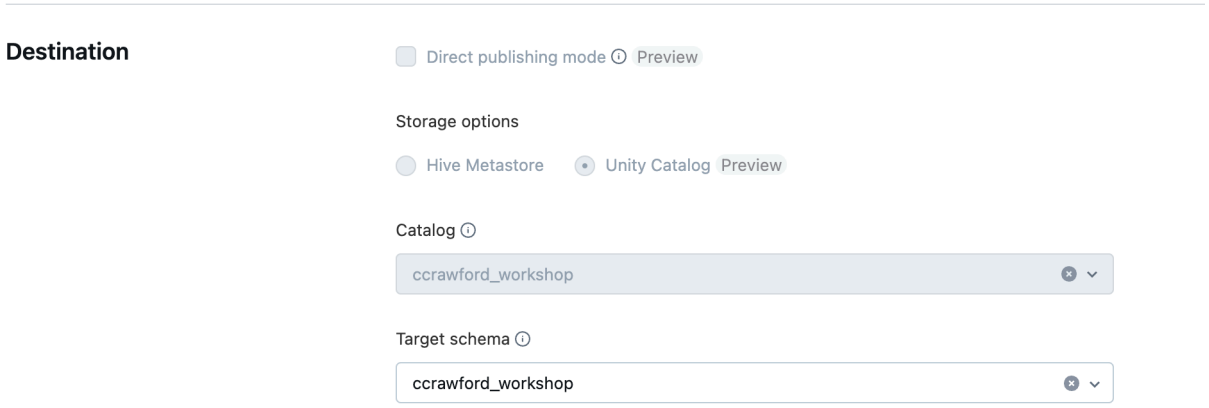
Give it a name, choose serverless, and leave it as “triggered”



Choose the notebook you just imported

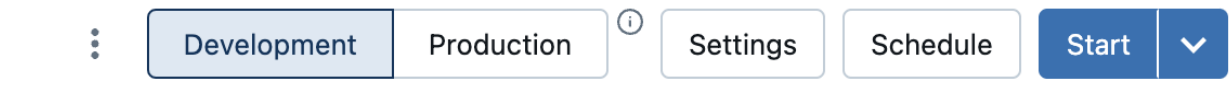


Select Unity Catalog and choose your catalog and schema



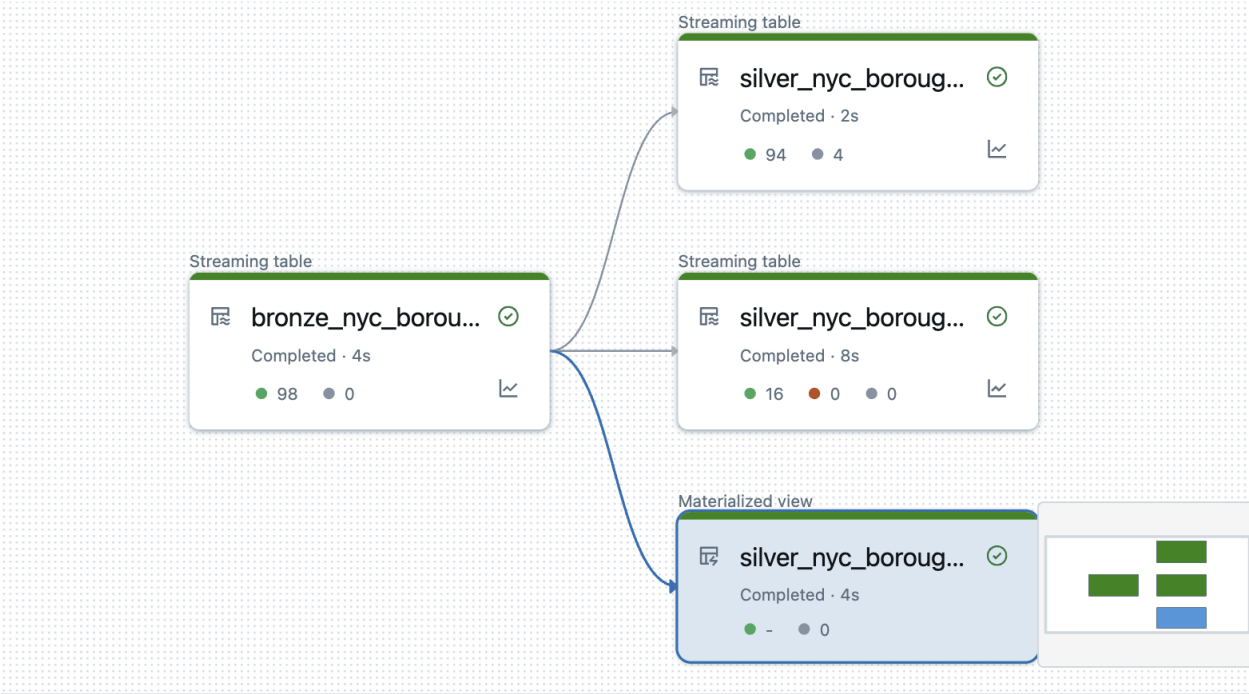
And save.

Explain the difference between Development and Production mode and choose to “Start”



When it finishes, you will have created four tables:

- bronze_nyc_boroughs - raw ingest
- silver_nyc_boroughs - first pass at cleaning with constraints
- silver_nyc_boroughs_cdc - Demonstration of SCD Type 1
- Silver_nyc_boroughs_mv - Materialized View

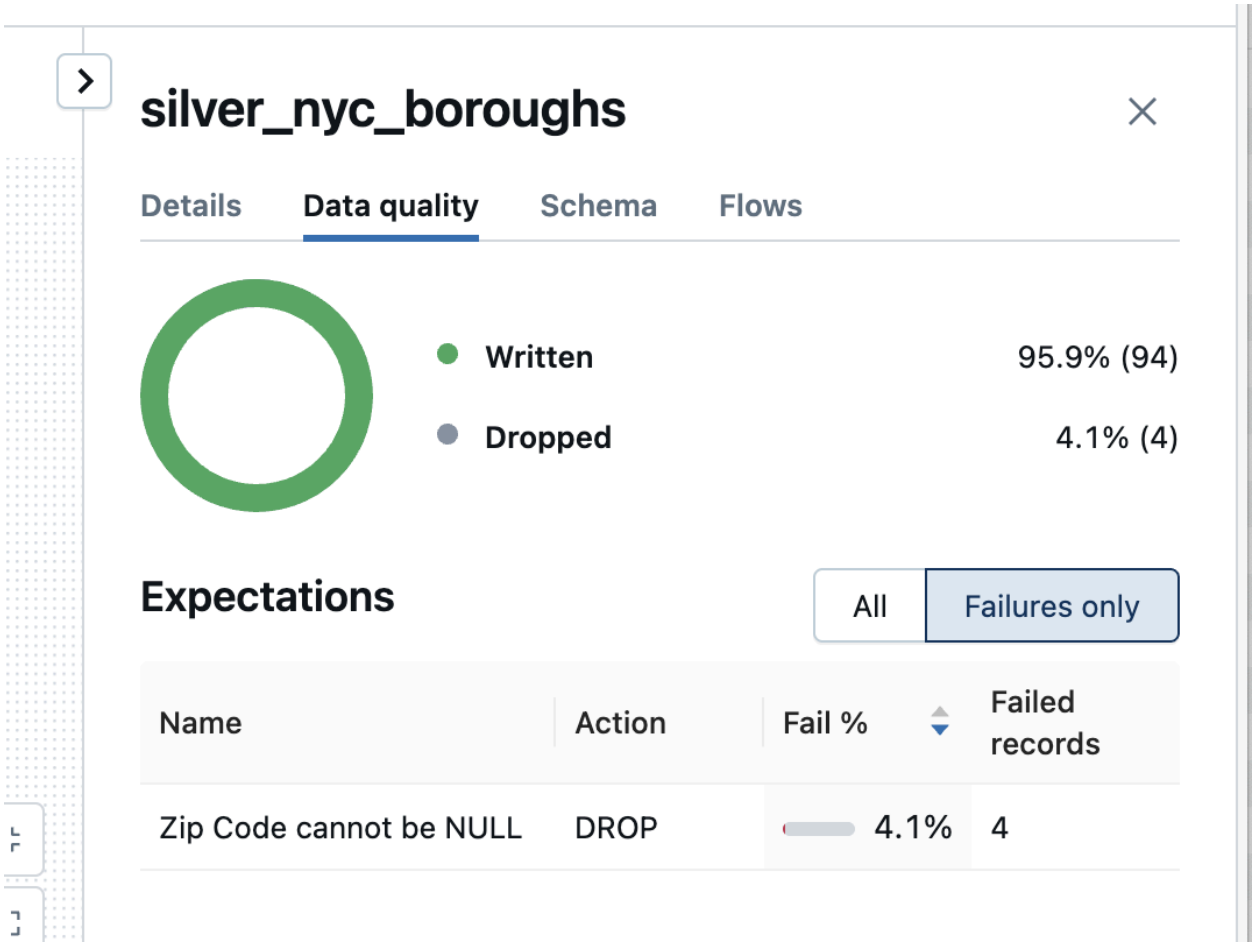


Now, let's demonstrate Databricks data quality capabilities and how to do a "Full Table Refresh."

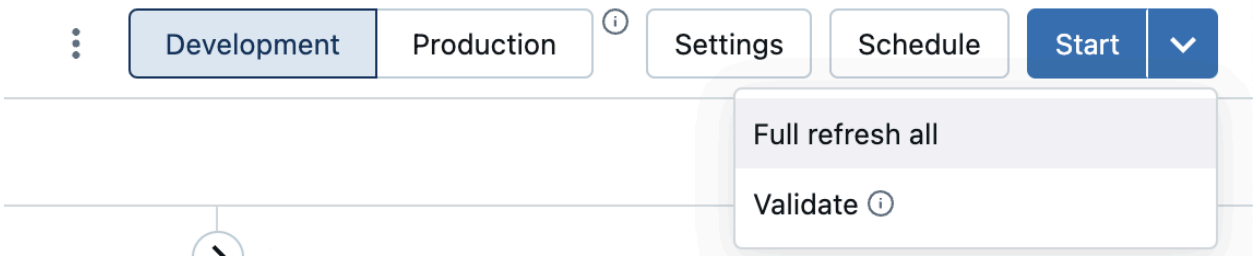
```
▶                                                                    3


@dlt.expect_or_drop("Zip Code cannot be NULL", "(zip_code IS NOT NULL)")
@dlt.table(table_properties={'quality': 'silver'})
def silver_nyc_boroughs():
  return (
    dlt.readStream('bronze_nyc_boroughs')
  )
```

If you look at the code for the silver_nyc_boroughs table, you'll see there is one constraint. Zip Code cannot be NULL, but if it is, drop the record. We can also see that on the graphical display.

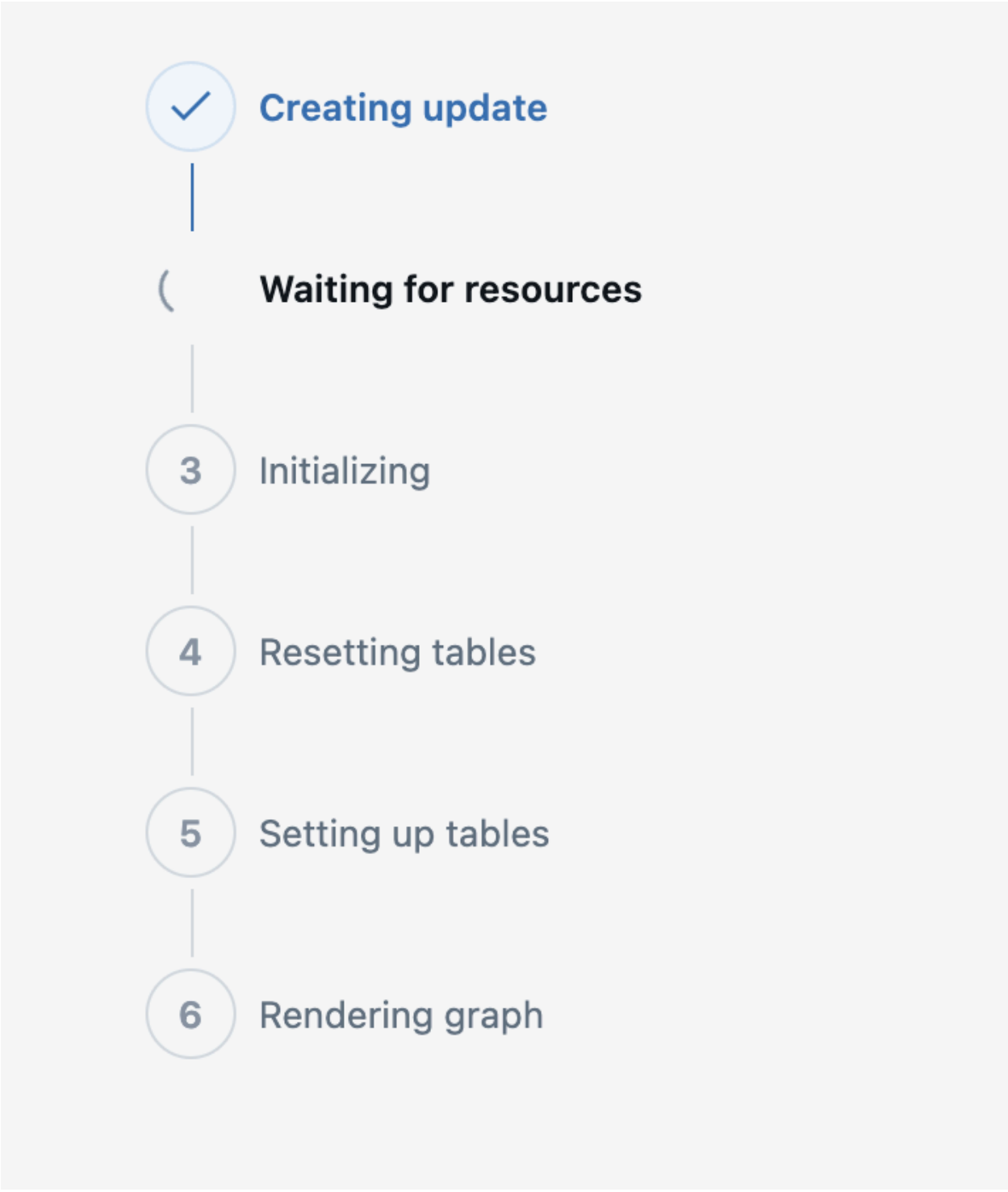
Four constraints occurred. Select the table and then the “Data Quality” tab and you can see what the constraint that was honored was and the net effect.



Go back to your pipeline and select to do a “Full Table Refresh”



Note the step of “resetting tables”, this lets us know a full table refresh is happening.



OPTIONAL: Explain that MVs do not show records written at this time, but that there is a JIRA filed to have this changed

DEMONSTRATE AUTOLOADER

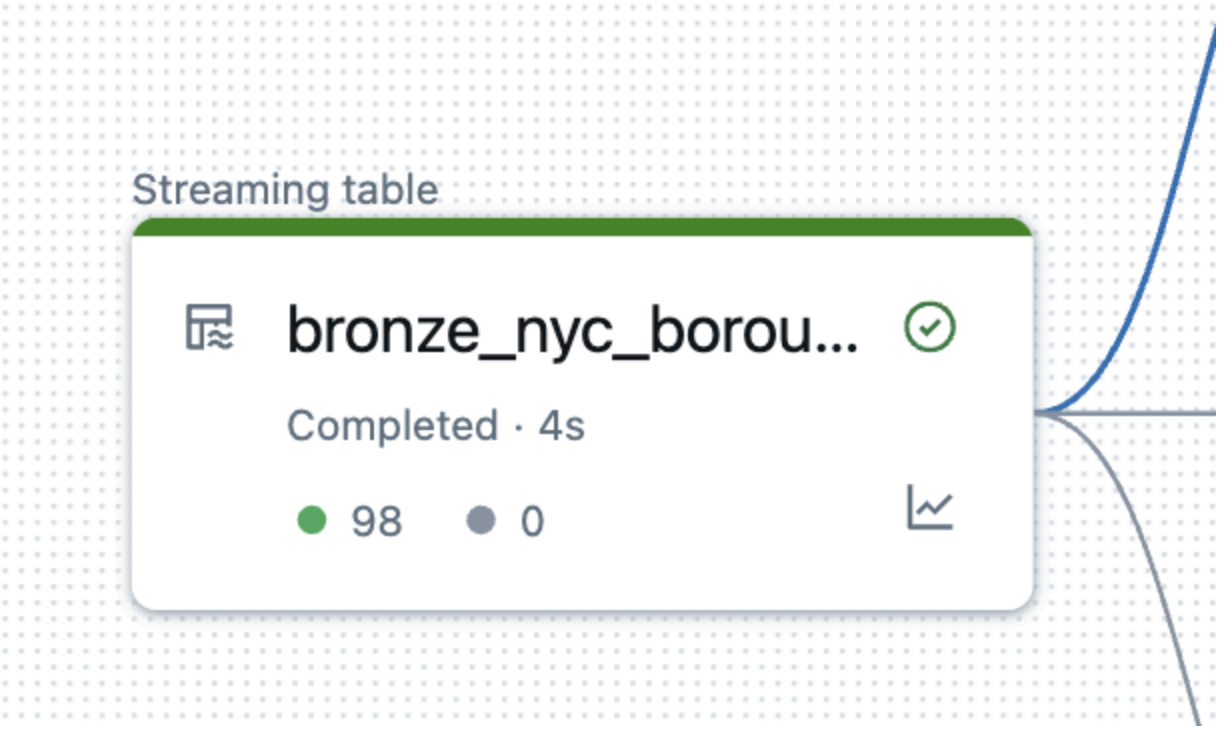
Upload the second file, dlt-pipeline-nyc_boroughs2.csv, to our volume.

Name	Size	Last modified
dlt-pipeline-nyc_boroughs1.csv	5.01 KB	18 minutes ago
dlt-pipeline-nyc_boroughs2.csv	5.52 KB	1 second ago

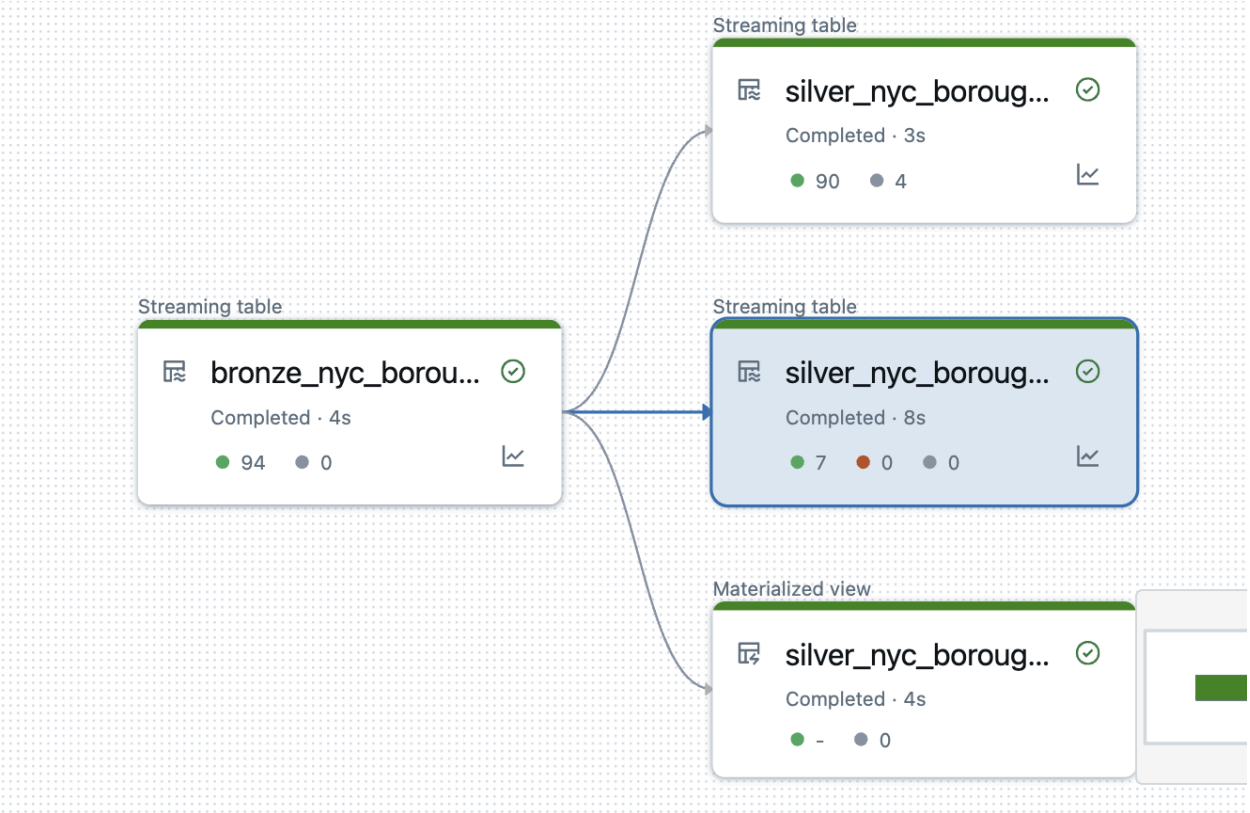
1

This file has a new column which states where a record should be updated, inserted, or deleted.

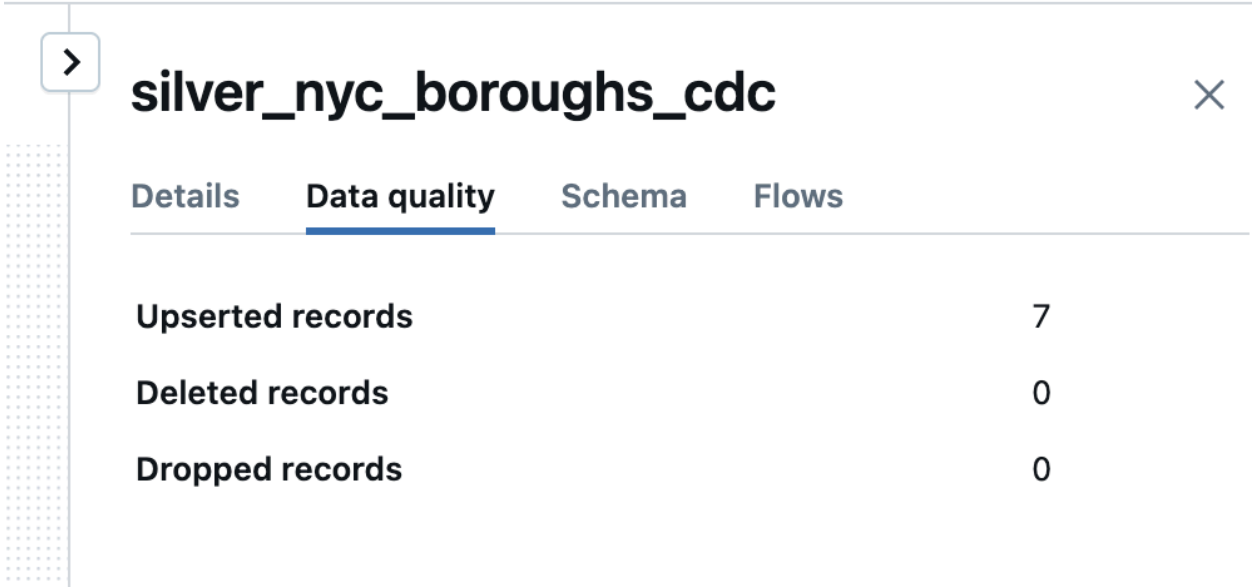
Note that the original pipeline wrote 98 records.



Re-run our DLT Pipeline by clicking the start button. When it completes, let’s look at the CDC (SCD Type 1) table.



If we click that file and choose “Data Quality” we see that 7 records were upserted.



We can further confirm that Autoloader (Cloudfiles) is working as expected. Note that only 94 records were processed this time. This confirms that Autoloader *ONLY* processed new files.

```
chris.crawford@T995G22PW5 workshop-intermediate-concepts % wc -l dlt-pipeline-nyc_boroughs2.csv
      95 dlt-pipeline-nyc_boroughs2.csv
```

(The file has 95 records, but the first line is the header, so only 94 were processed).

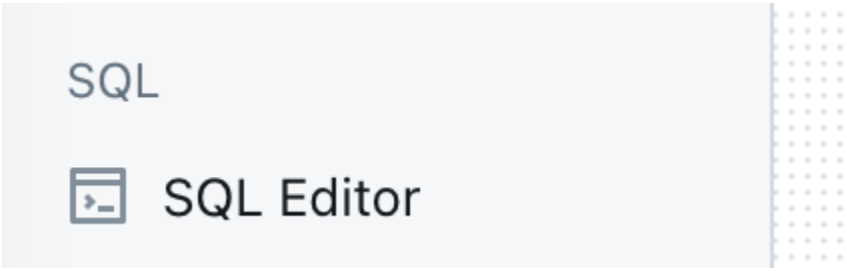
APPLY ROW LEVEL FILTERING AND COLUMN LEVEL MASKING

STOP - If you did not use this workshop as an add-on for the Introduction to Databricks workshop as the necessary tables will not be present. If you want to demonstrate this section, please go to the Introduction to Databricks and run the job to create the Vermont Traffic Stop Silver Table.

You can skip to the Pass Parameter section and Deploy a Databrick's App

On your computer, navigate to the downloads for this workshop and open the file "row-filtering-and-column-masking.txt". Copy the code to your clipboard.

Open SQL Editor and Paste the code



Before you run the code, navigate back to the "silver_traffic_stops_2016_2020" and show that before the filters are applied, you can see all cities and the driver's age.

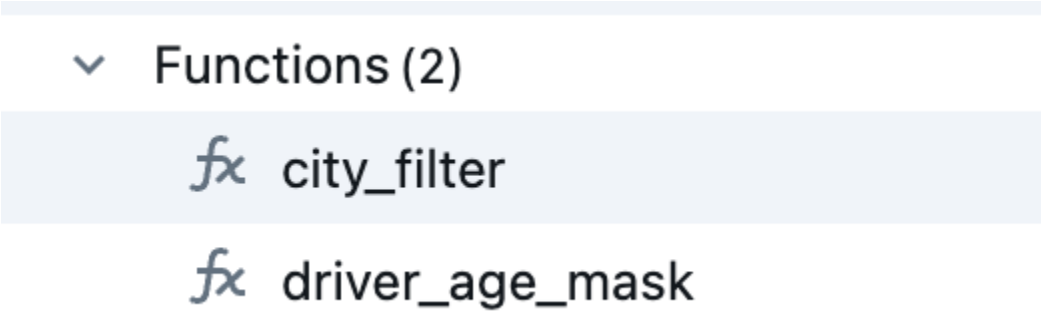
stop_date_and_time	street_address_of_stop	city_of_stop	
2016-02-04T00:39:00.000	I 89 N MM46	WIL	
2016-02-20T18:50:00.000	INTERSTATE 89 N ENTRANCE EXIT; mm47	BER	
2016-03-04T14:09:00.000	VT ROUTE 15;RT 108 N	CAM	
2016-03-14T14:00:00.000	rt14; vast trail	EMO	
2016-03-18T14:04:00.000	I 89 N MM34	RAN	
2016-03-20T18:10:00.000	Vt Route 22A	VRG	
2016-03-20T20:30:00.000	I 91 N MM33	RCK	
2016-06-25T17:40:00.000	TOWNE HILL RD & OLD FARM RD	EMO	
2016-09-28T19:00:00.000	Woodstock Ave @ Irving Oil	RUT	
2016-10-05T17:24:00.000	VT ROUTE 14 & BRIDGE ST	ROY	
2016-10-29T02:00:00.000	US Rt 302;2271 Scott Hwy	GRO	
2016-11-15T12:23:00.000	US Rt 5;Birch Meadow Rd	FAE	
2016-11-24T12:30:00.000	US RT 2; Oregon Rd	LUN	
2016-12-16T10:37:00.000	48 MORSE HILL RD	DOR	

	driver_race_description	driver_age	count	
	White	22	1	nu
	null	32	1	nu
	White	24	1	Wr
	White	50	1	nu
	White	47	1	nu
	White	21	1	nu
	White	26	1	nu
	White	33	1	nu
	Black	26	1	nu
	White	30	1	nu

Navigate back to the SQL Editor and run each Create block on it’s own.

```
▶ Run selected (1000)   ✓ 5 minutes ago (<1s)   uc_demos_chris_crawford . workshop_ccrawford

1  -- Create a Row Level Filter
2
3  CREATE
4  OR REPLACE FUNCTION city_filter(city_param STRING) RETURN city_param like "RCK";
5  SELECT
6    city_filter('RCK'),
7    city_filter('WIL');
8  ALTER TABLE
9    silver_traffic_stops_2016_2020
10 SET
11   ROW FILTER city_filter ON (city_of_stop);
12
13 -- create a SQL function for a simple column mask:
14
15 CREATE OR REPLACE FUNCTION driver_age_mask(driver_age STRING) RETURN IF(
16     is_account_group_member('bu_owner'), driver_age, "****"
17   );
18
19 ALTER TABLE silver_traffic_stops_2016_2020 ALTER COLUMN driver_age SET MASK driver_age_mask;
```

Show that a Function is also a Unity Catalog “object” and has permissions like anything else. This is *unified governance*.



The first filter will only return traffic stops from the city “RCK”

The second Function is an If/Then/Else command.
If the account is in the group “bu_owner”, then return the data, else mask the data with “****”

Go the catalog and show that the filters and masks are applied in the UI.

street_address_of_stop	city_of_stop	stop_reason	
I 91 N MM33	RCK		
991 ROCKINGHAM RD	RCK	M	S
ROCKINGHAM RD & STEARNS RD	RCK	M	N
I 91 N MM33	RCK	M	N
I 91 N MM33	RCK	M	N
ROCKINGHAM RD & LOWER BARTONSVIL	RCK	M	N
ROCKINGHAM RD & UPPER BARTONSVIL	RCK	M	N
I 91 N MM34	RCK	V	N
I 91 N MM34	RCK	M	N
I 91 N MM34	RCK	M	N
I 91 N MM34	RCK	M	N
I 91 N MM34	RCK	M	N

driver_race_description	driver_age	count	
White	****	1	
White	****	1	
White	****	1	
White	****	1	
White	****	1	
White	****	1	
White	****	1	
White	****	1	
White	****	1	
Asian or Pacific Islander	****	1	
Hispanic	****	1	
White	****	1	

PASS PARAMETERS IN A WORKFLOW

Navigate to your workspace for the workshop and create two notebooks:

1. get_tables
2. print_tables

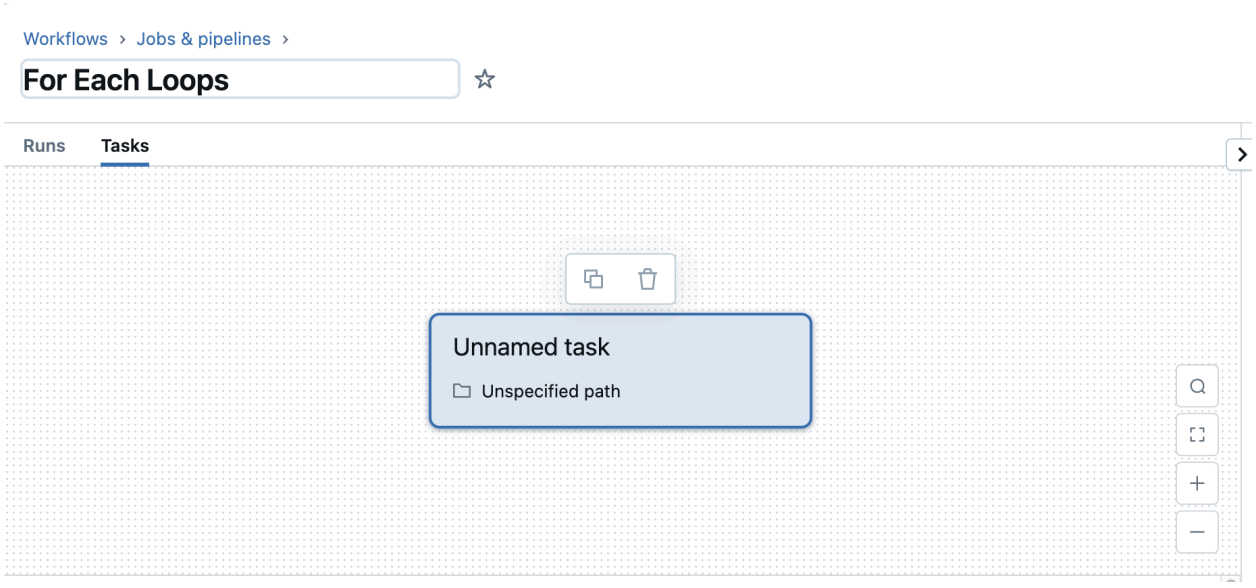
Code for get_tables (also found in the workshop material in jobs-for-each-loop-get_tables.txt)

```
table_list = [table.name for table in
spark.catalog.listTables('ccrawford_workshop.ccrawford_workshop')]


dbutils.jobs.taskValues.set(key = 'table_names', value = table_list)
```

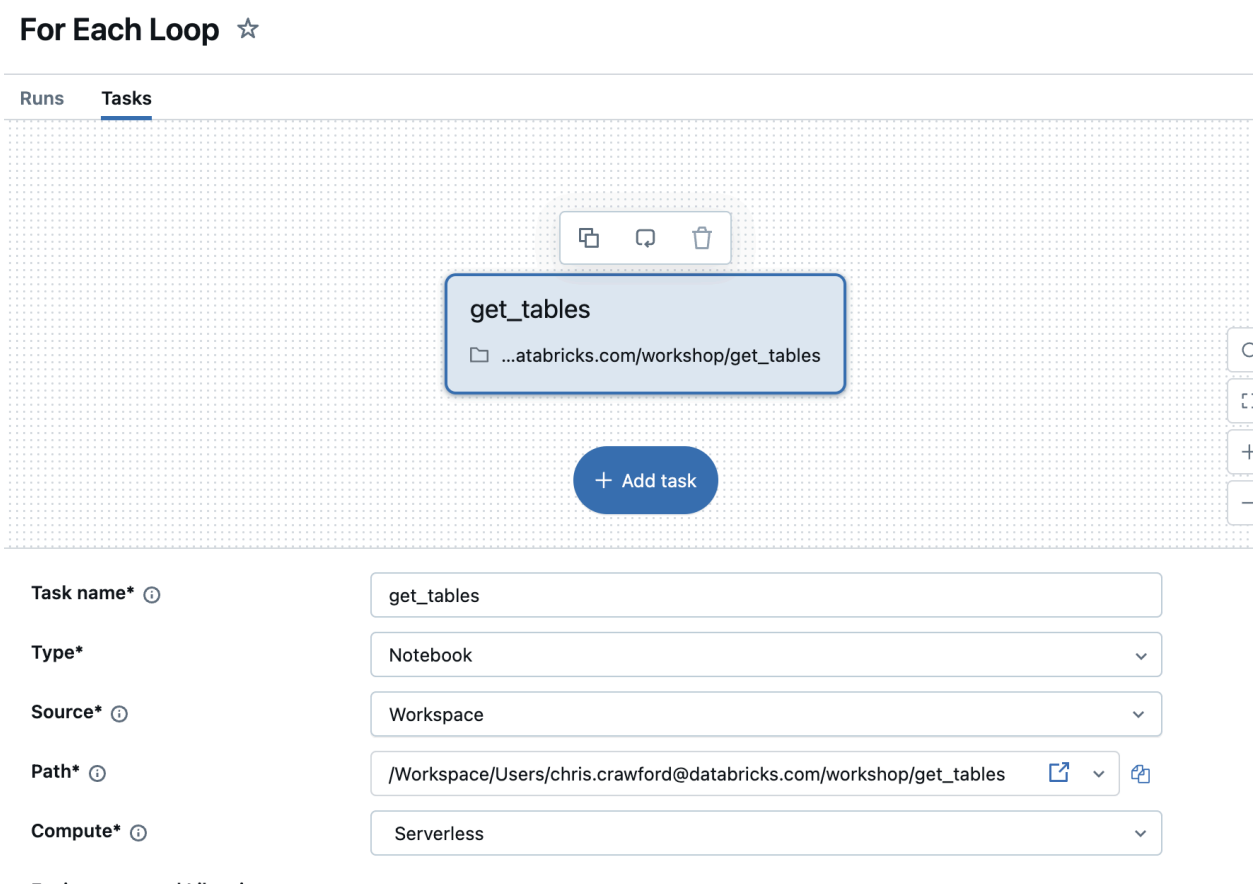
Code for print_tables (also found in the workshop material in jobs-for-each-loop-print_tables.txt)

```
print(dbutils.widgets.get('table_name'))
```

Create a new job:

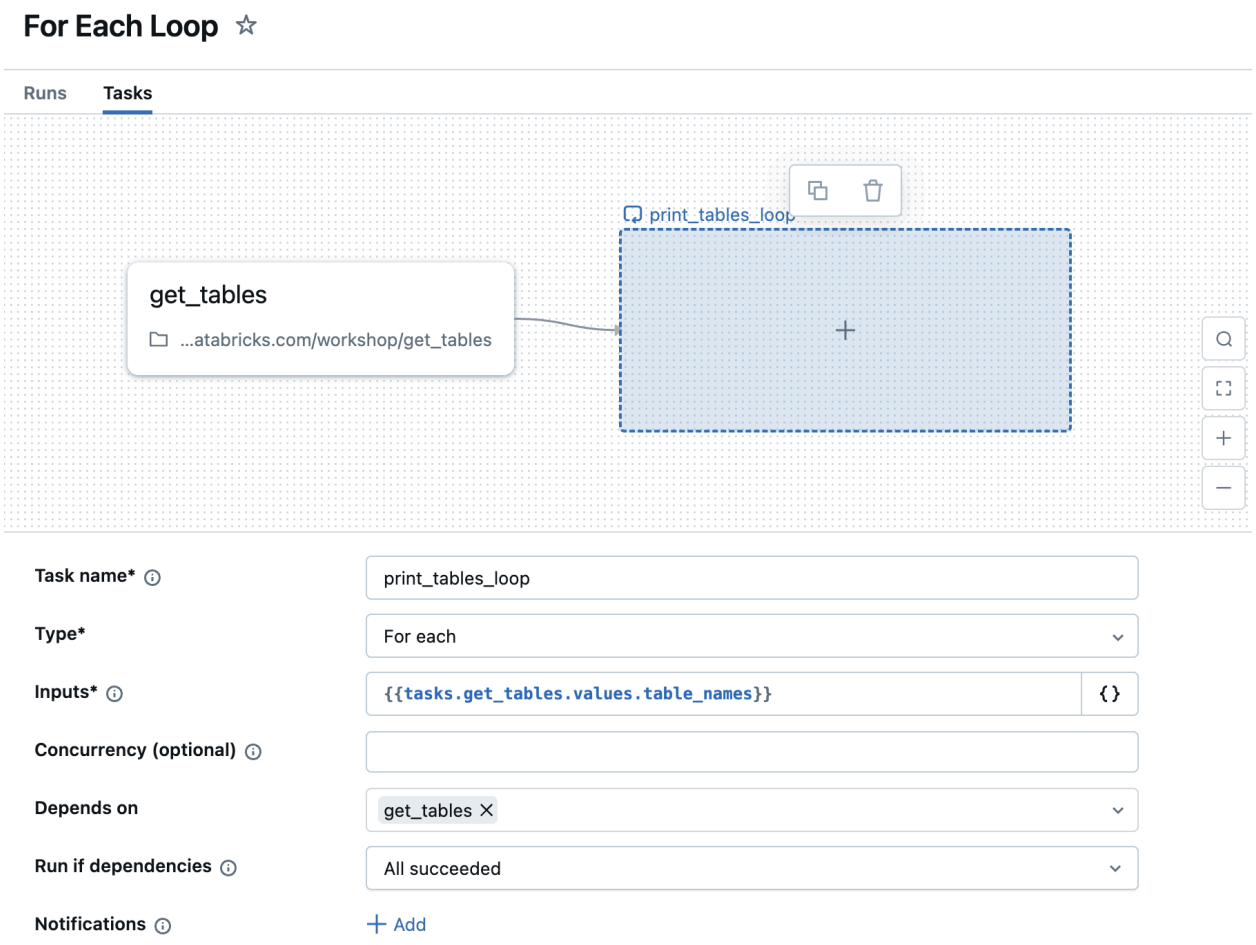


In the first task, select your “get_tables” notebook. And create the task.

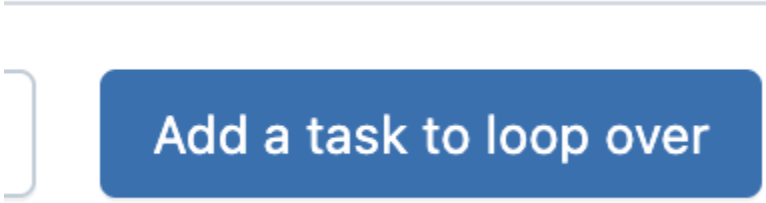


Now choose to “Add Task” create a For Each loop, let’s call this “print_tables_loop”. Change the “Inputs” to:

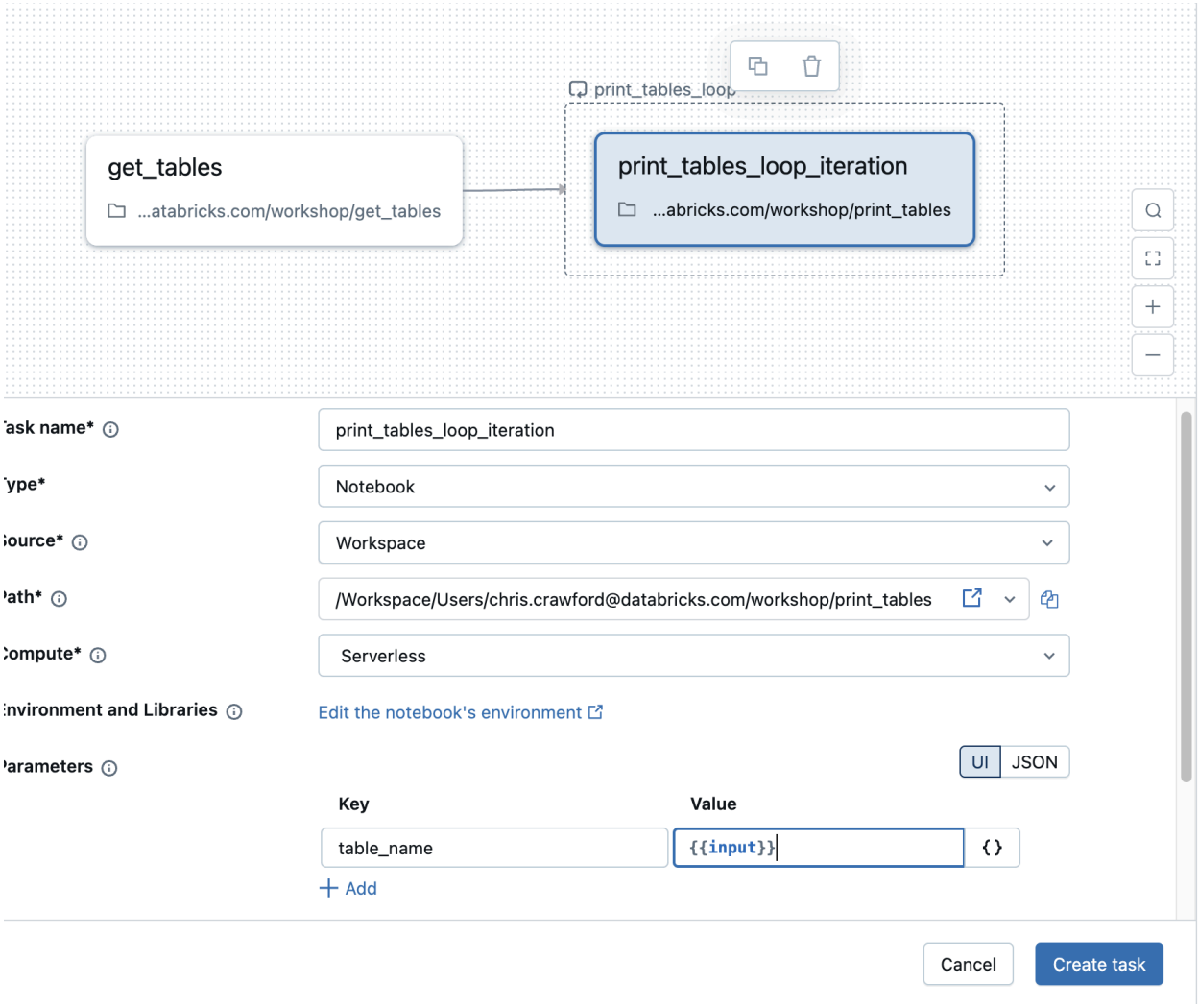
```
{{tasks.get_tables.values.table_names}}
```



Now create a task inside the For Each loop called “print_tables_iteration” and choose your “print_tables” notebook.

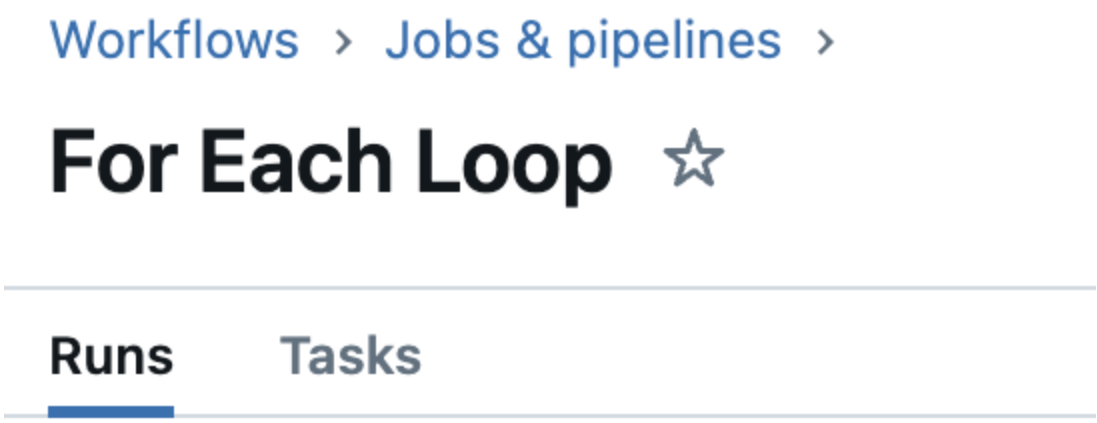


Set the following parameter value:

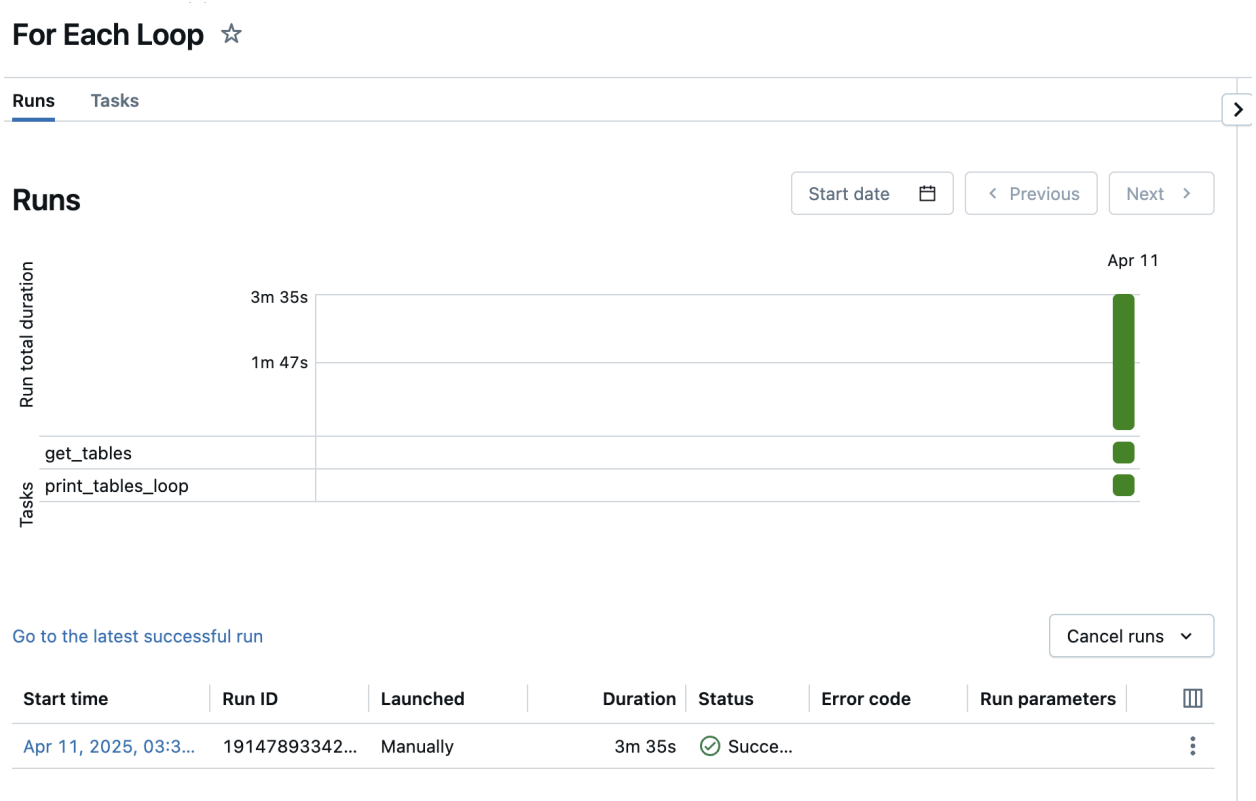


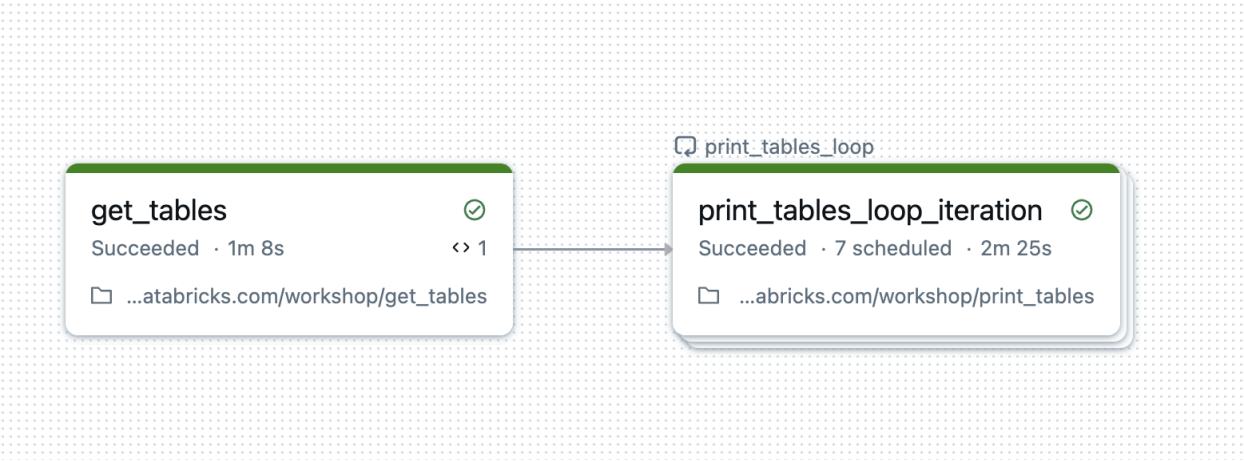
Create the task and run the job.

You can monitor the run by selecting the run tab.

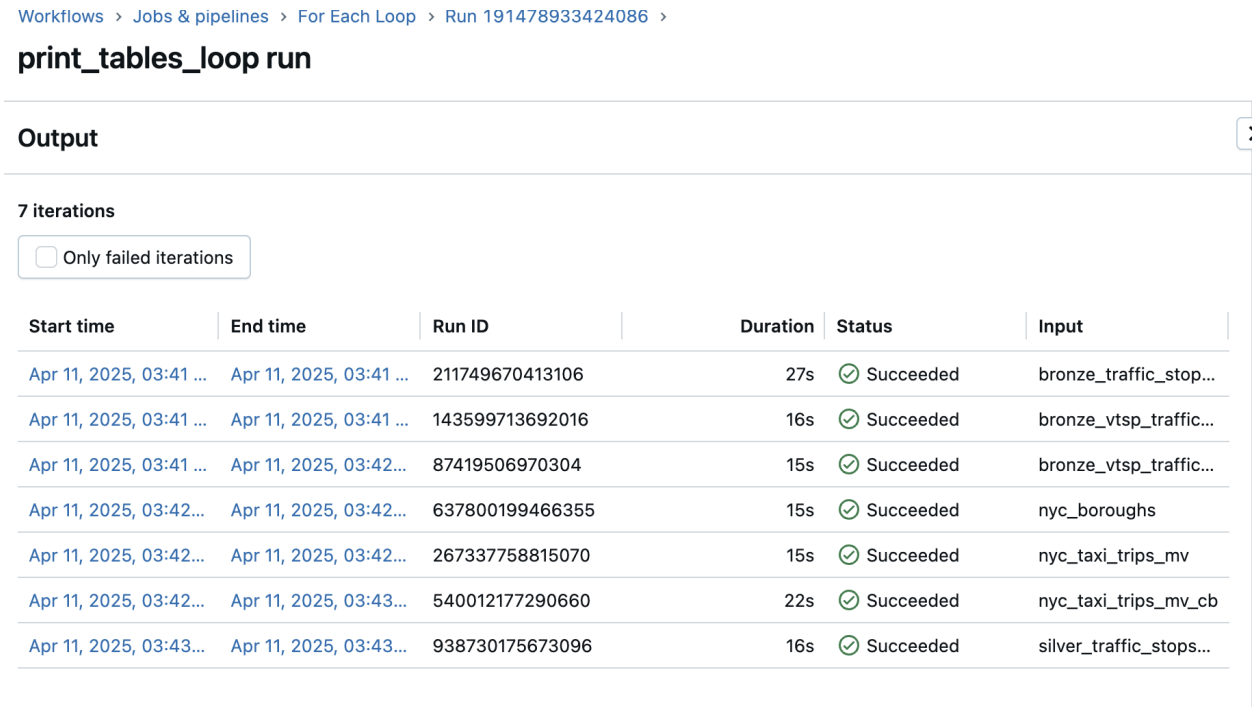


When the jobs ends, you can click the start time to see the details of the run.





Click the print_tables_loop_iteration task. We can see the job ran successfully and parameters were passed between the jobs. In short, we created a list of the tables in our workshop and then looped over that list and printed the names of the tables.



Start time	End time	Run ID	Duration	Status	Input
Apr 11, 2025, 03:41 ...	Apr 11, 2025, 03:41 ...	211749670413106	27s	Succeeded	bronze_traffic_stop...
Apr 11, 2025, 03:41 ...	Apr 11, 2025, 03:41 ...	143599713692016	16s	Succeeded	bronze_vtsp_traffic...
Apr 11, 2025, 03:41 ...	Apr 11, 2025, 03:42...	87419506970304	15s	Succeeded	bronze_vtsp_traffic...
Apr 11, 2025, 03:42...	Apr 11, 2025, 03:42...	637800199466355	15s	Succeeded	nyc_boroughs
Apr 11, 2025, 03:42...	Apr 11, 2025, 03:42...	267337758815070	15s	Succeeded	nyc_taxi_trips_mv
Apr 11, 2025, 03:42...	Apr 11, 2025, 03:43...	540012177290660	22s	Succeeded	nyc_taxi_trips_mv_cb
Apr 11, 2025, 03:43...	Apr 11, 2025, 03:43...	938730175673096	16s	Succeeded	silver_traffic_stops...

(REVAMPING) DEPLOY A DATABRICKS APP

The following workshop was derived from: https://github.com/pbv0/databricks-apps-cookbook

Recommend starting here: https://apps-cookbook.dev/docs/intro