

DLT Mini Project- Submission Report

Delta Live Tables Pipeline – NYC Taxi Analytics (Bronze → Silver → Gold)

— Siddartha Samisetty

Platform: Databricks

Dataset: Built-in samples.nyctaxi.trips

Pipeline Notebook: DLT_NYC_Taxi_Analytics_Pipeline

Validation Notebook: DLT_NYC_Taxi_Validation

Problem Statement

The objective of this project is to create a Delta Live Tables (DLT) pipeline on Databricks that ingests NYC taxi trip data, applies data quality expectations, generates Silver-level analytical tables, and produces a Gold-level business metric table. The pipeline must showcase Bronze → Silver → Gold lineage and include validation queries.

Solution Architecture

Medallion Architecture

Layer	Description	Output Object
Bronze	Raw ingestion from sample dataset	taxi_raw_records
Silver	Suspicious ride filtering & weekly aggregation	flagged_rides, weekly_stats
Gold	Top 3 highest fare rides for business analysis	top_n

DLT Pipeline Implementation (SQL)

Bronze Layer

```
CREATE OR REFRESH STREAMING TABLE taxi_raw_records
(CONSTRAINT valid_distance EXPECT (trip_distance > 0.0) ON VIOLATION
DROP ROW)
AS SELECT * FROM STREAM(samples.nyctaxi.trips);
```

I ingested raw streaming data from the sample dataset into a Bronze DLT table.

I also applied a data quality expectation rule to automatically remove invalid records, specifically any trip with non-positive trip distance.

This ensured only valid raw records were passed to the next stage

Silver – Suspicious Rides

```
CREATE OR REFRESH STREAMING TABLE flagged_rides AS
SELECT date_trunc("week", tpep_pickup_datetime) as week,
pickup_location_id, dropoff_location_id,
fare_amount, trip_distance
FROM STREAM(LIVE.taxi_raw_records)
WHERE ((pickup_location_id = dropoff_location_id AND fare_amount > 50)
OR (trip_distance < 5 AND fare_amount > 50));
```

I created logic to flag unusual taxi trips, such as:

- Very high fare for small distance
- Same pickup and drop location with unexpectedly high charges

This helped identify anomalies in the dataset.

Silver – Weekly Aggregation

I generated weekly average fare and distance metrics to help analyze business trends.

```
CREATE OR REFRESH MATERIALIZED VIEW weekly_stats AS
SELECT date_trunc("week", tpep_pickup_datetime) AS week,
AVG(fare_amount) AS avg_amount, AVG(trip_distance) AS avg_distance
FROM live.taxi_raw_records
GROUP BY week ORDER BY week ASC;
```

Gold – Top N

I combined Silver data to produce a **Gold-level table** that displayed the **Top 3 highest fare rides** for analysis.

This table is intended for **business stakeholders** to understand extreme cost trips and detect potential fraud or premium service usage.

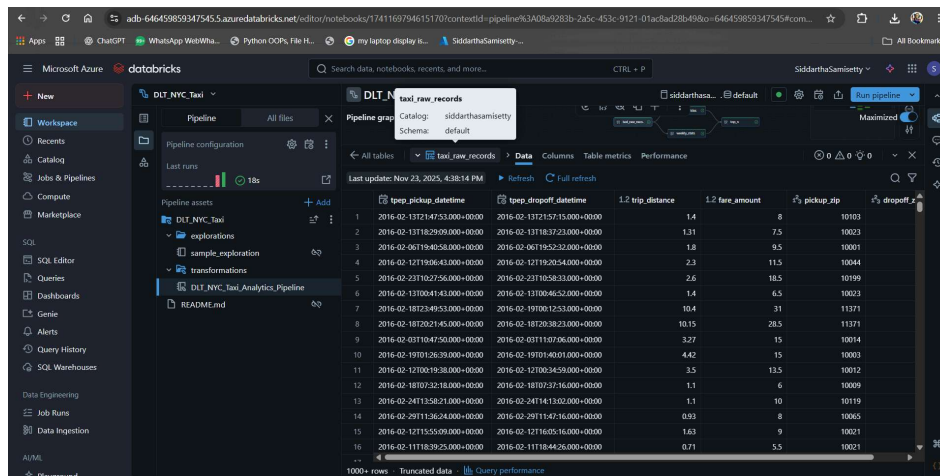
```
CREATE OR REPLACE MATERIALIZED VIEW top_n AS
SELECT weekly_stats.week, ROUND(avg_amount,2) as avg_amount,
ROUND(avg_distance,3) as avg_distance,
fare_amount, trip_distance, pickup_location_id
FROM live.flagged_rides
LEFT JOIN live.weekly_stats ON weekly_stats.week = flagged_rides.week
ORDER BY fare_amount DESC
LIMIT 3;
```

Screenshots:

The screenshot displays the Databricks workspace interface. The left sidebar shows the navigation menu with options like Workspace, Recents, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main area shows a pipeline named 'DLT_NYC_Taxi' with a graph of tables and views. The graph includes a streaming table 'taxi_raw_records' (3s), a streaming table 'flagged_rides' (2s), a materialized view 'weekly_stats' (3s), and a materialized view 'top_n' (2s). The 'top_n' view is the final output of the pipeline. Below the graph, a table performance summary is shown, listing the tables and their performance metrics.

Name	Catalog	Schema	Type	Du...	Ou...	Expecta...	Dr...	Wa...	Fai...	Increte...
flagged_rides	siddarth...	default	Streami...	2s	19	Not define	-	-	-	N/A
taxi_raw_records	siddarth...	default	Streami...	3s	22K	1 unmet	76	0	0	N/A
top_n	siddarth...	default	Material...	2s	3	Not define	-	-	-	Full rec
weekly_stats	siddarth...	default	Material...	3s	10	Not define	-	-	-	Full rec

Bronze



DLT_NYC_Taxi

Pipeline configuration

Last runs

Pipeline assets

DLT_NYC_Taxi

explorations

sample_exploration

transformations

DLT_NYC_Taxi_Analytics_Pipeline

README.md

taxi_raw_records

Catalog: siddharthasamietty

Schema: default

Columns

Table metrics

Performance

Last update: Nov 23, 2025, 4:38:14 PM

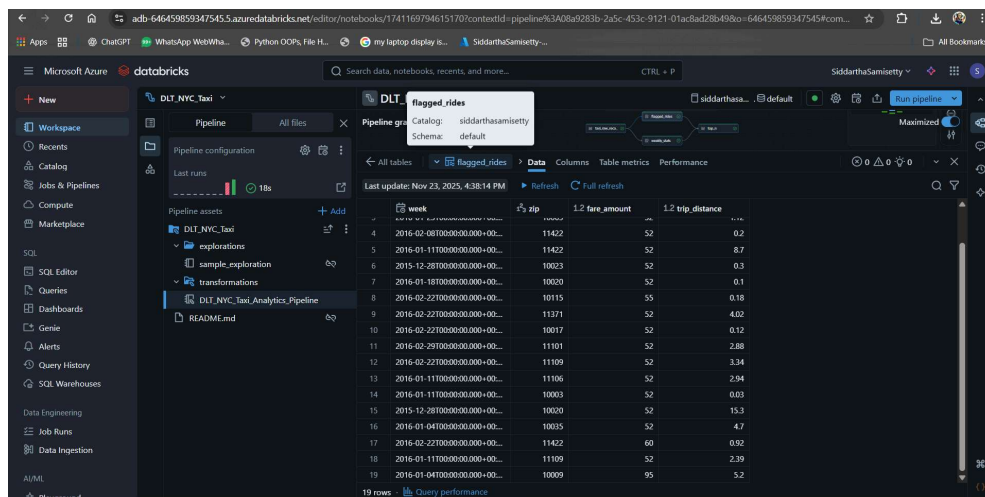
Refresh

Full refresh

	trip_pickup_datetime	trip_dropoff_datetime	1.2 trip_distance	1.2 fare_amount	pickup_zip	dropoff_zip
1	2016-02-13T21:47:53.000+00:00	2016-02-13T21:57:13.000+00:00	1.4	8	10103	
2	2016-02-13T18:29:09.000+00:00	2016-02-13T18:37:23.000+00:00	1.31	7.5	10023	
3	2016-02-06T19:40:58.000+00:00	2016-02-06T19:52:32.000+00:00	1.8	9.5	10001	
4	2016-02-12T19:06:43.000+00:00	2016-02-12T19:20:54.000+00:00	2.3	11.5	10044	
5	2016-02-23T10:27:56.000+00:00	2016-02-23T10:58:33.000+00:00	2.6	18.5	10199	
6	2016-02-13T00:41:43.000+00:00	2016-02-13T00:46:52.000+00:00	1.4	6.5	10023	
7	2016-02-18T23:49:53.000+00:00	2016-02-19T00:14:53.000+00:00	10.4	31	11371	
8	2016-02-18T20:21:45.000+00:00	2016-02-18T20:38:23.000+00:00	10.15	28.5	11371	
9	2016-02-03T10:47:50.000+00:00	2016-02-03T11:07:06.000+00:00	3.27	15	10014	
10	2016-02-19T01:26:39.000+00:00	2016-02-19T01:40:01.000+00:00	4.42	15	10003	
11	2016-02-12T00:19:38.000+00:00	2016-02-12T00:34:59.000+00:00	3.5	13.5	10012	
12	2016-02-18T07:32:18.000+00:00	2016-02-18T07:37:16.000+00:00	1.1	6	10009	
13	2016-02-24T13:58:21.000+00:00	2016-02-24T14:13:02.000+00:00	1.1	10	10119	
14	2016-02-29T11:36:24.000+00:00	2016-02-29T11:47:10.000+00:00	0.93	8	10065	
15	2016-02-12T15:55:09.000+00:00	2016-02-12T16:05:16.000+00:00	1.63	9	10021	
16	2016-02-11T16:39:25.000+00:00	2016-02-11T16:44:26.000+00:00	0.71	5.5	10021	

1000+ rows · Truncated data · Query performance

Silver-1



DLT_NYC_Taxi

Pipeline configuration

Last runs

Pipeline assets

DLT_NYC_Taxi

explorations

sample_exploration

transformations

DLT_NYC_Taxi_Analytics_Pipeline

README.md

flagged_rides

Catalog: siddharthasamietty

Schema: default

Columns

Table metrics

Performance

Last update: Nov 23, 2025, 4:38:14 PM

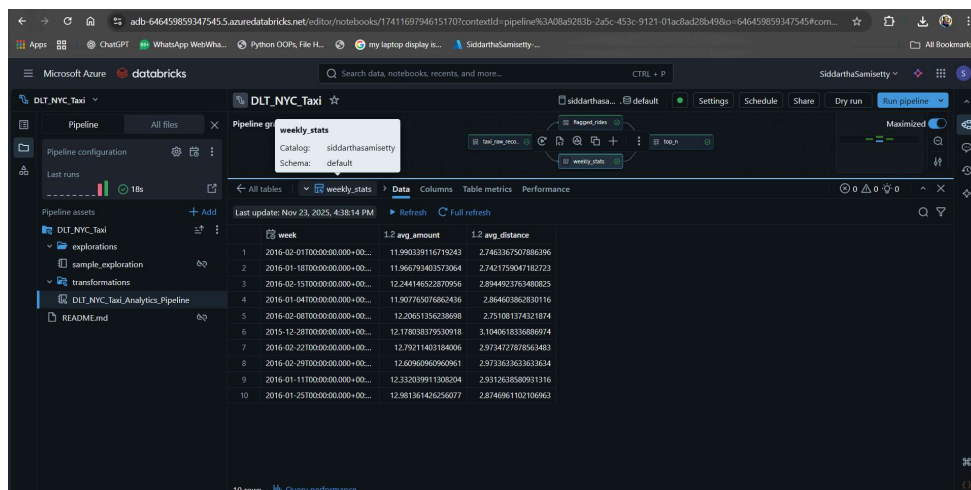
Refresh

Full refresh

	week	zip	1.2 fare_amount	1.2 trip_distance
4	2016-02-08T00:00:00.000+00:00	11422	52	0.2
5	2016-01-11T00:00:00.000+00:00	11422	52	8.7
6	2015-12-28T00:00:00.000+00:00	10023	52	0.3
7	2016-01-18T00:00:00.000+00:00	10020	52	0.1
8	2016-02-22T00:00:00.000+00:00	10115	55	0.18
9	2016-02-22T00:00:00.000+00:00	11371	52	4.02
10	2016-02-22T00:00:00.000+00:00	10017	52	0.12
11	2016-02-29T00:00:00.000+00:00	11101	52	2.88
12	2016-02-22T00:00:00.000+00:00	11109	52	3.34
13	2016-01-11T00:00:00.000+00:00	11106	52	2.94
14	2016-01-11T00:00:00.000+00:00	10003	52	0.03
15	2015-12-28T00:00:00.000+00:00	10020	52	15.3
16	2016-01-04T00:00:00.000+00:00	10035	52	4.7
17	2016-02-22T00:00:00.000+00:00	11422	60	0.92
18	2016-01-11T00:00:00.000+00:00	11109	52	2.39
19	2016-01-04T00:00:00.000+00:00	10009	95	3.2

19 rows · Query performance

silver-Week stats



DLT_NYC_Taxi

Pipeline configuration

Last runs

Pipeline assets

DLT_NYC_Taxi

explorations

sample_exploration

transformations

DLT_NYC_Taxi_Analytics_Pipeline

README.md

weekly_stats

Catalog: siddharthasamietty

Schema: default

Columns

Table metrics

Performance

Last update: Nov 23, 2025, 4:38:14 PM

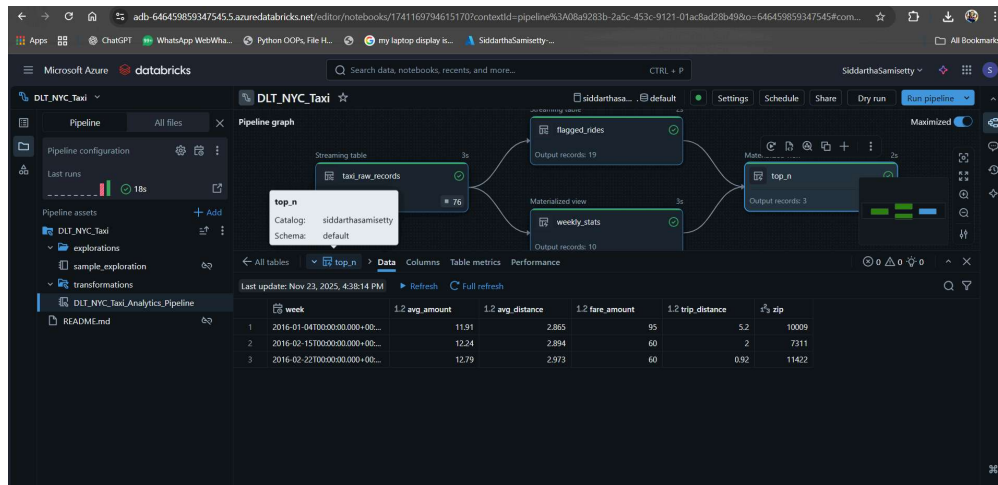
Refresh

Full refresh

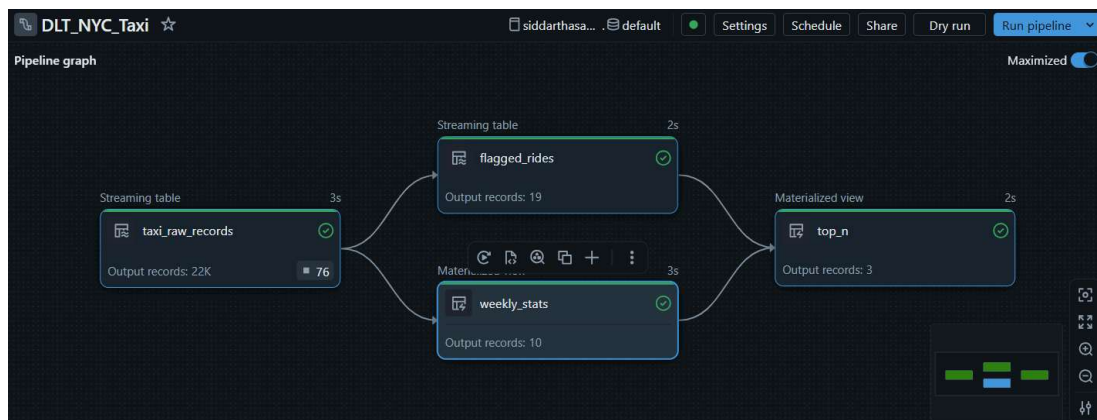
	week	1.2 avg_amount	1.2 avg_distance
1	2016-02-01T00:00:00.000+00:00	11.980339116719243	2.746336750786396
2	2016-01-18T00:00:00.000+00:00	11.966793401573064	2.7421759047182723
3	2016-02-15T00:00:00.000+00:00	12.24414652870956	2.8944923763480823
4	2016-01-04T00:00:00.000+00:00	11.907165076862436	2.864603862810116
5	2016-02-08T00:00:00.000+00:00	12.20651356238698	2.751081174321874
6	2015-12-28T00:00:00.000+00:00	12.17801879539018	3.104061635868674
7	2016-02-22T00:00:00.000+00:00	12.79211401184006	2.973477878563483
8	2016-02-29T00:00:00.000+00:00	12.6908960969061	2.973363363363634
9	2016-01-11T00:00:00.000+00:00	12.33203911308204	2.9312638580931316
10	2016-01-25T00:00:00.000+00:00	12.981361426256077	2.874696110210693

10 rows · Query performance

TopN_GOLD



Pipeline Lineage Graph



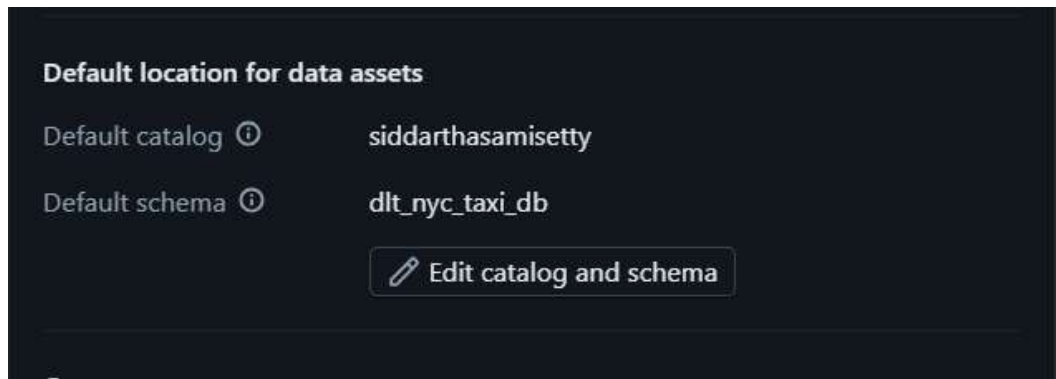
Validation Notebook Queries

1. Connecting to the Correct Catalog & Schema

In the validation notebook, I first selected the catalog and schema where DLT created the tables. This ensures all queries point to the right database, not the default sample environment.

This proved that:

- The DLT pipeline wrote data to the configured storage location
- The output tables are accessible and queryable



```
USE CATALOG siddarthasamisetty;
USE dlt_nyc_taxi_db;
SHOW TABLES;
```

1. Previewing Bronze Layer Output

```
SELECT * FROM taxi_raw_records LIMIT 10;
```

databricks DLT_NYC_Taxi_Validation (SQL) Import notebook

Check Bronze layer sample

3

```
SELECT *
FROM taxi_raw_records
LIMIT 10;
```

pySpark SQL DataFrame Dataframe = [tpep_pickup_datetime: timestamp, tpep_dropoff_datetime: timestamp, ... 4 more fields]

Table	tpep_pickup_datetime	tpep_dropoff_datetime	l2_trip_distance	l2_fare_amount	r5_pickup_zip	r5_dropoff_zip
1	2016-02-13T21:47:53.000+00:00	2016-02-13T21:57:15.000+00:00	1.4	8	10103	10110
2	2016-02-13T18:26:09.000+00:00	2016-02-13T18:37:23.000+00:00	1.31	7.5	10023	10023
3	2016-02-06T19:40:58.000+00:00	2016-02-06T19:52:32.000+00:00	1.8	9.5	10001	10018
4	2016-02-12T19:06:43.000+00:00	2016-02-12T19:20:54.000+00:00	2.3	11.5	10044	10111
5	2016-02-23T10:27:56.000+00:00	2016-02-23T10:58:33.000+00:00	2.6	18.5	10199	10022
6	2016-02-13T00:41:43.000+00:00	2016-02-13T00:46:52.000+00:00	1.4	6.5	10023	10009
7	2016-02-18T23:49:53.000+00:00	2016-02-19T00:12:53.000+00:00	10.4	31	11371	10003
8	2016-02-18T20:14:45.000+00:00	2016-02-18T20:38:23.000+00:00	10.15	28.5	11371	11001
9	2016-02-03T19:47:50.000+00:00	2016-02-03T19:57:06.000+00:00	3.27	15	10014	10023
10	2016-02-19T01:26:30.000+00:00	2016-02-19T01:40:01.000+00:00	4.42	15	10003	11222

10 rows

This result is stored as _sqldf_ and can be used in other Python and SQL cells.

2. Validating Silver Suspicious Rides Table

```
SELECT * FROM flagged_rides ORDER BY fare_amount DESC LIMIT 10;
```

View suspicious flagged rides

```
SELECT *
FROM Flagged_rides
ORDER BY fare_amount DESC
LIMIT 10;
```

__sql__df: pyspark.sql.dataframe.DataFrame = [week: timestamp, zip: integer ... 2 more fields]

	week	zip	1.2 fare_amount	1.2 trip_distance
1	2016-01-04T00:00:00.000+00...	10009	95	5.2
2	2016-02-15T00:00:00.000+00...	7311	60	2
3	2016-02-22T00:00:00.000+00...	11422	60	0.92
4	2016-02-22T00:00:00.000+00...	10115	55	0.18
5	2016-01-11T00:00:00.000+00...	11422	52	8.7
6	2016-01-25T00:00:00.000+00...	11109	52	3
7	2016-01-18T00:00:00.000+00...	10020	52	0.1
8	2015-12-28T00:00:00.000+00...	10023	52	0.3
9	2016-02-08T00:00:00.000+00...	11422	52	0.2
10	2016-02-22T00:00:00.000+00...	11371	52	4.02

10 rows

3. Validating Weekly Aggregates

`SELECT * FROM weekly_stats ORDER BY week ASC LIMIT 10;`

Review weekly aggregated results

```
SELECT *
FROM weekly_stats
ORDER BY week ASC
LIMIT 10;
```

__sql__df: pyspark.sql.dataframe.DataFrame = [week: timestamp, avg_amount: double ... 1 more field]

	week	1.2 avg.am...	1.2 avg_distance
1	2015-12-28T00:00:00.000+00...	12.178038379530918	3.1040618336886974
2	2016-01-04T00:00:00.000+00...	11.907760075662436	2.366460362830116
3	2016-01-11T00:00:00.000+00...	12.33203991338204	2.931363858093116
4	2016-01-18T00:00:00.000+00...	11.966793483573564	2.7421759047183733
5	2016-01-25T00:00:00.000+00...	12.981361424256077	2.8746961103106903
6	2016-02-01T00:00:00.000+00...	11.990339116710243	2.7463367507886396
7	2016-02-08T00:00:00.000+00...	12.20651356238698	2.751081374321874
8	2016-02-15T00:00:00.000+00...	12.244140522870956	2.8944923703480825
9	2016-02-22T00:00:00.000+00...	12.792114031384006	2.9734727878563483
10	2016-02-29T00:00:00.000+00...	12.609060960960961	2.9733633633633634

10 rows

4. Validating the Gold Table

`SELECT * FROM top_n ORDER BY fare_amount DESC;`

Validate Gold output (top N highest rides)

```
SELECT *
FROM top_n
ORDER BY fare_amount DESC;
```

__sql__df: pyspark.sql.dataframe.DataFrame = [week: timestamp, avg_amount: double ... 4 more fields]

	week	1.2 avg_amount	1.2 avg_distance	1.2 fare_amount	1.2 trip_distance	zip
1	2016-01-04T00:00:00.000+00...	11.91	2.865	95	5.2	10009
2	2016-02-15T00:00:00.000+00...	12.24	2.894	60	2	7311
3	2016-02-22T00:00:00.000+00...	12.79	2.973	60	0.92	11422

3 rows

This result is stored as __sql__df and can be used in other Python and SQL cells.

