

# 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	<code>taxi_raw_records</code>
Silver	Suspicious ride filtering & weekly aggregation	<code>flagged_rides</code> , <code>weekly_stats</code>
Gold	Top 3 highest fare rides for business analysis	<code>top_n</code>

# 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 shows the Databricks workspace interface. On the left, the sidebar includes sections for Microsoft Azure, Workspace, Recents, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main area displays the 'DLT\_NYC\_Taxi' pipeline. The 'Pipeline graph' tab is active, showing a flow from a 'Streaming table: taxi\_raw\_records' to a 'Streaming table: flagged\_rides' (with 19 output records), which then feeds into a 'Materialized view: weekly\_stats' (with 10 output records). Finally, the 'weekly\_stats' view feeds into a 'Materialized view: top\_n' (with 3 output records). Below the graph, the 'Tables' and 'Performance' tabs are visible, showing detailed information for each table. The bottom status bar indicates the current time as Nov 23, 2025, 04:37 PM, a refresh interval of 18s, and a 'Query performance' link.

## Bronze

**Pipeline configuration:**

- Catalog: siddarthasamisetty
- Schema: default

**Data Preview:**

	tpep_pickup_datetime	tpep_dropoff_datetime	l2_trip_distance	l2_fare_amount	pickup_zip	dropoff_zip
1	2016-02-13T18:29:00.000	2016-02-13T21:57:15.000	1.4	8	10103	
2	2016-02-13T18:29:00.000	2016-02-13T18:37:23.000	1.31	7.5	10023	
3	2016-02-06T19:40:50.000	2016-02-06T19:52:32.000	1.8	9.5	10001	
4	2016-02-12T19:06:43.000	2016-02-12T19:20:45.000	2.3	11.5	10044	
5	2016-02-23T10:27:56.000	2016-02-23T10:58:33.000	2.6	18.5	10199	
6	2016-02-13T00:41:43.000	2016-02-13T00:46:52.000	1.4	6.5	10023	
7	2016-02-18T23:49:53.000	2016-02-19T00:12:53.000	10.4	31	11371	
8	2016-02-19T02:19:05.000	2016-02-18T28:38:23.000	10.15	28.5	11371	
9	2016-02-19T10:47:50.000	2016-02-19T11:07:06.000	3.27	15	10014	
10	2016-02-19T01:26:39.000	2016-02-19T01:40:01.000	4.42	15	10003	
11	2016-02-12T00:19:38.000	2016-02-12T00:34:59.000	3.5	13.5	10012	
12	2016-02-18T02:32:31.000	2016-02-18T02:37:16.000	1.1	6	10009	
13	2016-02-24T13:52:82.000	2016-02-24T14:13:02.000	1.1	10	10119	
14	2016-02-29T11:36:24.000	2016-02-29T11:47:16.000	0.93	8	10065	
15	2016-02-12T15:55:09.000	2016-02-12T16:05:16.000	1.63	9	10021	
16	2016-02-11T18:39:53.000	2016-02-11T18:44:26.000	0.71	5.5	10021	

## Silver-1

**Pipeline configuration:**

- Catalog: siddarthasamisetty
- Schema: default

**Data Preview:**

	week	zip	l2_fare_amount	l2_trip_distance
4	2016-02-08T00:00:00.000	11422	\$2	0.2
5	2016-01-11T00:00:00.000	11422	\$2	8.7
6	2015-12-28T00:00:00.000	10023	\$2	0.3
7	2016-01-18T00:00:00.000	10020	\$2	0.1
8	2016-02-27T00:00:00.000	10115	\$5	0.16
9	2016-02-27T00:00:00.000	11371	\$2	4.02
10	2016-02-27T00:00:00.000	10017	\$2	0.12
11	2016-02-27T00:00:00.000	11101	\$2	2.88
12	2016-02-22T00:00:00.000	11109	\$2	3.34
13	2016-01-11T00:00:00.000	11106	\$2	2.94
14	2016-01-11T00:00:00.000	10003	\$2	0.03
15	2015-12-28T00:00:00.000	10020	\$2	15.3
16	2016-01-04T00:00:00.000	10035	\$2	4.7
17	2016-02-27T00:00:00.000	11422	\$6	0.92
18	2016-01-11T00:00:00.000	11109	\$2	2.39
19	2016-01-04T00:00:00.000	10009	\$5	5.2

## Silver-Weak stats

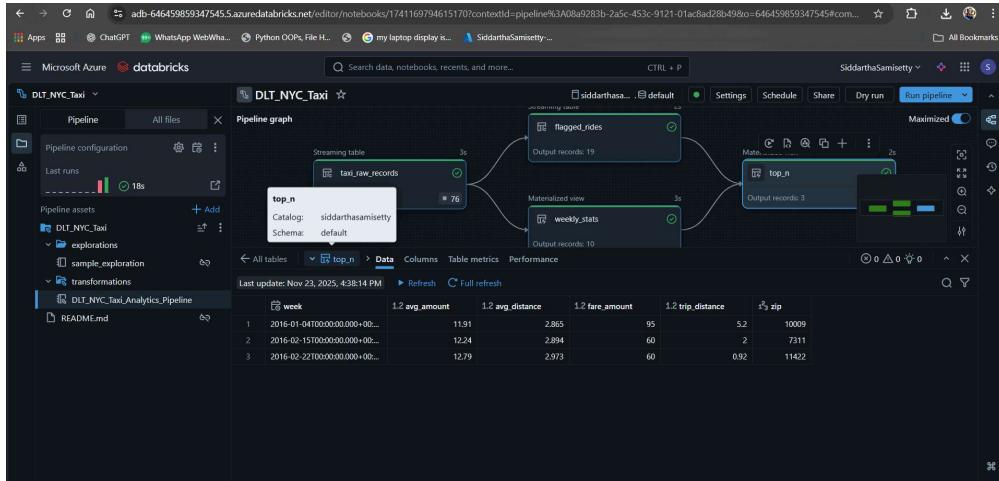
**Pipeline configuration:**

- Catalog: siddarthasamisetty
- Schema: default

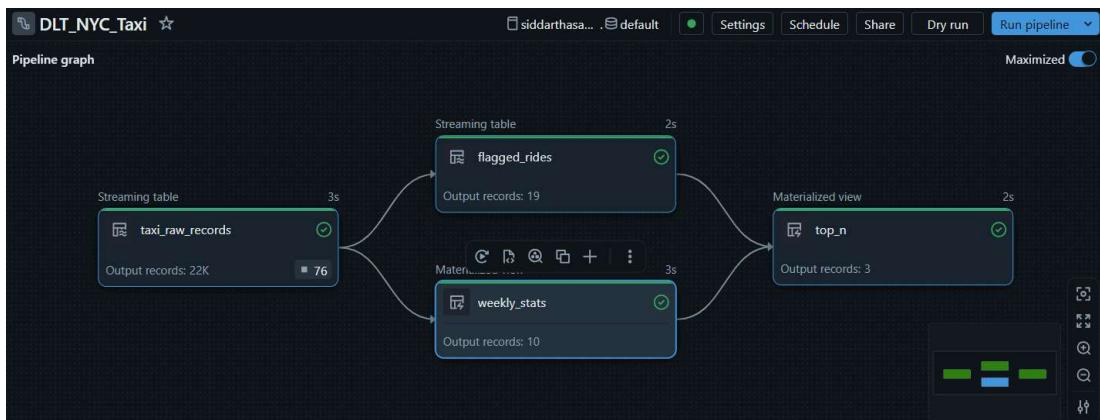
**Data Preview:**

	week	l2_avg_amount	l2_avg_distance
1	2016-02-01T00:00:00.000	11900.3911671924	2.742367507886796
2	2016-01-18T00:00:00.000	11956739403573064	2.742175947182723
3	2016-02-15T00:00:00.000	12244146522870956	2.8944923763488025
4	2016-01-04T00:00:00.000	1190700676824396	2.86400562630116
5	2016-02-08T00:00:00.000	122065136236968	2.75108174321874
6	2015-12-28T00:00:00.000	12178038319530918	3.10406183686974
7	2016-02-22T00:00:00.000	127921401184006	2.97347278536483
8	2016-02-29T00:00:00.000	1260960960960961	2.97326336363634
9	2016-01-11T00:00:00.000	12332039911308204	2.9312638580931316
10	2016-01-25T00:00:00.000	1298136142656077	2.8746961102106963

## 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

**Default location for data assets**

Default catalog ⓘ siddarthasamisetty

Default schema ⓘ dlt\_nyc\_taxi\_db

**Edit catalog and schema**

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

## 1. Previewing Bronze Layer Output

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

	tpep_pickup_datetime	tpep_dropoff_datetime	l1_trip_distance	l1_fare_amount	l1_pickup_zip	l1_dropoff_zip
1	2016-02-13T21:47:53.000+00:00	2016-02-17T21:57:15.000+00:00	1.4	8	10103	10110
2	2016-02-13T19:29:09.000+00:00	2016-02-17T18:37:23.000+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	2016-02-17T19:20:54.000+00:00	2.3	11.5	10044	10111
5	2016-02-23T19:27:56.000+00:00	2016-02-23T19:58:31.000+00:00	2.6	18.5	10199	10022
6	2016-02-13T04:41:43.000+00:00	2016-02-13T04:46:52.000+00:00	1.4	6.5	10023	10069
7	2016-02-18T21:49:53.000+00:00	2016-02-19T01:25.12.000+00:00	10.4	31	11371	10003
8	2016-02-18T21:21:45.000+00:00	2016-02-18T20:38:21.000+00:00	10.15	28.5	11371	11201
9	2016-02-03T10:47:50.000+00:00	2016-02-03T11:07:06.000+00:00	3.27	15	10014	10023
10	2016-02-19T01:26:39.000+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;
```

```

SELECT *
FROM flagged_rides
ORDER BY fare_amount DESC
LIMIT 10;

```

Table

	week	1.2 zip	1.2 fare_amount	1.2 trip_distance
1	2016-01-04T00:00:00+00:00...	95	5.2	0.92
2	2016-02-15T00:00:00+00:00...	7311	60	2
3	2016-02-23T00:00:00+00:00...	11422	60	0.92
4	2016-03-23T00:00:00+00:00...	10115	55	0.18
5	2016-01-11T00:00:00+00:00...	11422	52	8.7
6	2016-01-25T00:00:00+00:00...	11109	52	3
7	2016-01-18T00:00:00+00:00...	10020	52	0.1
8	2015-12-28T00:00:00+00:00...	10023	52	0.2
9	2016-02-08T00:00:00+00:00...	11422	52	0.2
10	2016-02-23T00:00:00+00:00...	11371	52	4.02

10 rows

### 3. Validating Weekly Aggregates

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

```

SELECT *
FROM weekly_stats
ORDER BY week ASC
LIMIT 10;

```

Table

	week	1.2 avg_amount	1.2 avg_distance
1	2015-12-28T00:00:00+00:00...	12.178038379530918	3.1040618336886974
2	2016-01-14T00:00:00+00:00...	11.907765070682436	2.864603862030116
3	2016-01-21T00:00:00+00:00...	12.332039911028043	2.9316305805031116
4	2016-01-28T00:00:00+00:00...	11.966793403573664	2.7421759047182723
5	2016-02-04T00:00:00+00:00...	12.98136142625077	2.874696102196963
6	2016-02-11T00:00:00+00:00...	11.990339116719243	2.746367507863696
7	2016-02-18T00:00:00+00:00...	12.20651356238698	2.7510813743181674
8	2016-02-25T00:00:00+00:00...	12.244146322670956	2.894923763480825
9	2016-03-01T00:00:00+00:00...	12.792114018409	2.3744727878565483
10	2016-02-29T00:00:00+00:00...	12.6956906960961	2.9736336363364

10 rows

### 4. Validating the Gold Table

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

```

SELECT *
FROM top_n
ORDER BY fare_amount DESC;

```

Table

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

3 rows

This result is stored as `_sql1dF` and can be used in other Python and SQL cells.

