NYC Taxi Spark Project

DATA PROCESSING AND ANALYTICS

Thomas Ernest CERESA

Analyzing New York City Taxi Data

Spatial and temporal exploration, enrichment and analysis of taxi trips in New York using Apache Spark

Date: October 22, 2025

Academic Year: 2025-2026

Contents

Introduction

This project analyses New York City taxi trip data using Apache Spark, focusing on the spatial and temporal aspects of taxi operations. The main objective is to understand taxi utilisation patterns through geospatial data enrichment and temporal analysis.

I adopt the following approach: first, exploratory data analysis to understand the structure and quality of the raw data; second, systematic enrichment and quality assessment to prepare the dataset for data cleaning, particularly by flagging anomalies. Then, data cleaning is done on several datasets. And utimatly, the queries are applied.

1 Exploration Data

1.1 Dataset Structure

The NYC taxi dataset consists of 99,999 trip records, each representing a single taxi ride. Each record contains:

- **Identifiers:** Medallion (vehicle ID) and hack license (driver ID), both hashed for privacy
- **Temporal information:** Pickup and dropoff datetimes in the format `dd-MM-yy HH:mm`
- **Spatial information:** Pickup and dropoff coordinates as (latitude, longitude) pairs
- **Operational data:** Passenger count, vendor ID, rate code, and store-and-forward flag

The GeoJSON file contains 104 polygonal features representing the geographical boundaries of NYC's five boroughs: Manhattan (code 1), Bronx (code 2), Brooklyn (code 3), Queens (code 4), and Staten Island (code 5). These boundaries enable spatial classification of trips.

1.2 Data Loading with Spark

We employ Apache Spark's RDD and DataFrame APIs for data ingestion. The choice between RDDs and DataFrames is strategic:

- **RDDs** provide low-level control and are useful for exploratory tasks where schema is not yet defined
- **DataFrames** offer columnar structure and benefit from Catalyst query optimization, making them preferable for subsequent analytical operations

My final choice is to keep out the benefits of DataFrames – SQL Spark uses, column names etc.

1.3 Geospatial Data Representation

The GeoJSON data is enriched with Shapely polygon objects:

$$\text{Polygon}_i = \text{shape}(\text{GeoJSON}_i[\text{geometry}]) \tag{1}$$

Each polygon's area is computed to enable optimization by sorting. Borough codes (1-5) and polygon areas (in square degrees) provide a natural ordering heuristic: features are sorted first by borough code, then by area in descending order. This makes sure that queries encounter the largest, a reducing average time.

1.4 Data Quality Assessment

Initial exploration reveals potential anomalies:

- **Borough distribution:** Most trips originate and terminate in Manhattan (90,099 of 99,999 pickups)
- **Unknown borough classifications:** 1,940 pickups and 2,357 dropoffs cannot be mapped to known boroughs

2 Enrichement and Analyse

2.1 Enrichment Strategy

Data enrichment augments raw records with derived attributes that facilitate analysis. I enrich records through two dimensions: spatial and temporal.

2.2 Spatial

2.2.1 Point-in-Polygon Classification

For each trip, we classify pickup and dropoff locations by determining which borough polygon contains the corresponding point. This is formalized as:

$$\text{borough}(\text{lat}, \text{lon}) = \begin{cases} \texttt{borough}_i & \text{if } P(\text{lon}, \text{lat}) \in \text{Polygon}_i \\ \text{“Unknown”} & \text{otherwise} \end{cases} \tag{2}$$

where $P(\text{lon}, \text{lat})$ is the spatial point and Polygon_i is the geometric polygon for borough i. This operation is computationally expensive due to the need to test point containment against multiple polygons. To optimize, we broadcast the sorted GeoJSON data to all Spark workers, ensuring a single copy per worker in memory.

2.2.2 UDFs

To enable distributed enrichment logic, I define three User-Defined Functions (UDFs):

- `find_borough_udf`: Maps (lat, lon) to borough names via point-in-polygon testing
- `is_same_location_udf`: Detects identical pickup/dropoff coordinates ($\epsilon = 0.0001$)
- `get_trip_type_udf`: Classifies trips as intra-borough, inter-borough, or unknown

2.2.3 Quality Flags for Spatial Anomalies

I introduce Boolean flags to categorize unknown borough classifications:

- `flag_coords_zero`: Indicates coordinates are exactly (0.0, 0.0), suggesting missing or null data
- `flag_out_of_nyc`: Indicates coordinates fall outside realistic NYC bounds (latitude $\in [40.4, 41.0]$, longitude $\in [-74.3, -73.7]$)
- `flag_in_nyc_but_unknown`: Coordinates are within NYC bounds but no polygon contains the point, suggesting possible data entry errors or GeoJSON boundary gaps

Analysis reveals:

- 1,932 records (77.5% of unknowns) have zero coordinates
- 2,035 records (81.7% of unknowns) fall outside NYC geographic bounds
- 457 records (18.3% of unknowns) fall within NYC bounds but cannot be classified
- Total unknown records: 2,492 (2.49% of dataset)

The relatively small unknown fraction suggests the dataset is reasonably clean, though outliers warrant investigation.

2.3 Temporal

2.3.1 Timestamp Conversion and Duration Calculation

I convert string datetime fields to Spark timestamp objects using the format `dd-MM-yy HH:mm`. Duration is calculated as the difference in seconds between dropoff and pickup times:

$$\text{duration\_seconds} = \text{dropoff\_ts} - \text{pickup\_ts} \tag{3}$$

2.3.2 Temporal Quality Flags

I introduce flags to detect temporal anomalies:

- `flag_parsing_error`: Timestamp conversion failed, indicating malformed datetime strings
- `flag_negative_duration`: Dropoff time precedes pickup time (logical impossibility)
- `flag_zero_duration`: Pickup and dropoff times are identical (trip length = 0 seconds)
- `flag_excessive_duration`: Trip duration exceeds 4 hours (threshold for outlier detection)
- `flag_too_short`: Trip duration between 1 and 29 seconds (suspiciously brief)

2.3.3 Analysis of Temporal Anomalies

Temporal quality assessment reveals:

- 0 parsing errors: All timestamps parse successfully
- 0 negative durations: No temporal inversions detected
- 450 zero-duration trips (0.45% of dataset)
- 0 excessive durations: No trips exceed 4 hours
- 0 suspiciously short trips: All trips with non-zero duration exceed 29 seconds

The 450 zero-duration trips warrant investigation. Analysis shows:

- 272 occur between unknown-unknown borough pairs (same-location data errors)
- 98 occur between Manhattan-Manhattan pairs
- 289 of the 450 have identical pickup and dropoff coordinates

These likely represent canceled trips, immediate re-pickups, or data entry errors where coordinates were not updated. They should be excluded from downstream analyses of trip duration and utilization.

2.4 Identifier Quality

I verify the integrity of record identifiers (`medallion` for vehicle ID and `hack_license` for driver ID) to ensure proper tracking for utilization analysis. Validation flags detect null, empty, or suspiciously short values (length < 10 characters):

- `flag_invalid_medallion`: Invalid vehicle identifier
- `flag_invalid_hack_license`: Invalid driver identifier

Results: All VALID 99,999 records (100%)

3 Data Cleaning

3.1 Dataset Definitions

Rather than applying a single rigid criterion, I adopt a multi-dataset approach for experimental flexibility. This enables testing analytical robustness under varying quality constraints.

I construct datasets in two categories: **base datasets** that isolate specific quality dimensions, and **composite datasets** that combine valid records across dimensions for broader analysis.

3.1.1 Base Datasets

These datasets isolate specific quality criteria and serve as building blocks:

Dataset	**Criteria**	**Records**	**% Original**
D_{strict}	Spatial VALID $\wedge$ Temporal VALID	97,375	97.38%
$D_{\text{recoverable}}$	IN_NYC_UNKNOWN $\wedge$ Temporal VALID	451	0.45%
D_{spatial}	Spatial VALID $\wedge$ Temporal anomalies	132	0.13%
D_{temporal}	Spatial anomalies $\wedge$ Temporal VALID	2,173	2.17%
$D_{\text{no-zero}}$	Duration > 0	99,549	99.55%
$D_{\text{reasonable}}$	(VALID $\vee$ IN_NYC_UNKNOWN) $\wedge$ $1 \leq$ duration ≤ 240 min	97,827	97.83%

Table 1: Base dataset definitions and record counts

3.1.2 Composite Datasets

These datasets combine valid records across dimensions to maximize coverage while maintaining analytical integrity:

Dataset	**Criteria**	**Records**	**% Original**
$D_{\text{extended\_spatial}}$	$D_{\text{strict}} \cup D_{\text{recoverable}}$	97,826	97.83%
$D_{\text{complete\_temporal}}$	$(D_{\text{strict}} \cup D_{\text{temporal}}) \wedge$ duration > 0	99,5	99.55%
$D_{\text{spatial\_valid}}$	$D_{\text{strict}} \cup D_{\text{spatial}}$	97,507	97.51%

Table 2: Composite dataset definitions and record counts

Rationale for composite datasets:

- $D_{\text{extended\_spatial}}$: Includes borderline spatial cases (IN_NYC_UNKNOWN) for geographic flow analysis where precise borough classification is less critical
- $D_{\text{complete\_temporal}}$: Maximizes temporal coverage for utilization and timing analyses, accepting spatial uncertainty
- $D_{\text{spatial\_valid}}$: All geographically valid trips regardless of temporal quality, useful for pure spatial pattern analysis

3.1.3 Geographic Stratification

For comparative analysis between NYC's core and periphery:

Dataset	**Criteria**	**Records**	**% of $D_{\textbf{strict}}$**
$D_{\text{manhattan}}$	Pickup $\vee$ Dropoff in Manhattan	94,077	96.61%
D_{outer}	Pickup $\wedge$ Dropoff $\notin$ Manhattan	3,298	3.39%

Table 3: Geographic stratification of D_{strict}

3.2 Final Dataset for Queries

For the required queries, I select $D_{\text{final}} := D_{\text{strict}}$.

This dataset ensures high spatial and temporal confidence. I enrich it with the `trip_type` column and select relevant fields: `medallion`, `hack_license`, timestamps, durations, boroughs, `trip_type`, `passenger_count`, and coordinates. The dataset is cached for query optimization.

Data loss: From 99,999 original records, D_{final} retains 97,375 records.

3.3 Optimizations Applied

- **Broadcasting:** Sorted GeoJSON data broadcast to all workers
- **Caching:** D_{final} cached in memory for repeated access
- **Column projection:** Only necessary columns retained
- **Flag precomputation:** Quality flags computed once during enrichment

4 Queries and Results

4.1 Query 1: Utilization Rate

4.1.1 Methodology

We define a driver's work session as a sequence of consecutive trips separated by idle periods shorter than 4 hours. Longer gaps are considered session breaks (e.g., end of shift). For each driver d, utilization is computed as:

$$\text{Utilization}_d = \frac{\sum_i T_{\text{occupied},i}}{\sum_i T_{\text{occupied},i} + \sum_i T_{\text{idle},i}}$$

where $T_{\text{occupied},i}$ is the duration of trip i and $T_{\text{idle},i}$ is the waiting time before the next trip (capped at 4 h).

4.1.2 Implementation

The computation relies on Spark's window functions:

- **Partitioning:** Trips are grouped by `hack_license` to ensure all records for a given driver are processed together.
- **Ordering:** Within each partition, records are ordered by `pickup_ts`.
- **Lag calculation:** The `lag()` function retrieves each driver's previous drop-off timestamp.
- **Idle time computation:** Idle durations are derived from the difference between consecutive trips and capped at 4 hours.
- **Aggregation:** Per-driver utilization is aggregated using `groupBy()` and summarized across the dataset.

4.1.3 Results

The analysis involves 9,846 unique drivers with valid trip sequences.

Metric	**Value**
Mean utilization	53.0%
Standard deviation	25.6%
Minimum utilization	3.0%
Maximum utilization	100.0%
Average trips per driver	9.9

Table 4: Global utilization statistics

To illustrate, only drivers with at least 10 trips were considered for ranking. The table below illustrates examples of top and bottom performers.

Driver ID (masked)	**Trips**	**Utilization (%)**
d41f9a...	32	92.1
84a2b9...	28	87.4
71c4ab...	26	85.9
32b8c4...	12	10.2
94f7a1...	11	5.8

Table 5: Examples of top and bottom driver utilization (<= 10 trips)

The distribution (Table 4) shows a large group of drivers around 40–60% utilization and a smaller cluster above 80%. This suggests heterogeneous habits: some drivers operate continuously throughout a shift, while others have sporadic trips with long idle intervals.

4.2 Query 2: Average Time to Next Fare by Borough

4.2.1 Methodology

This query computes the average waiting time for drivers to acquire their next passenger after completing a dropoff, stratified by dropoff borough. The metric provides insights into demand density across NYC's five boroughs.

For each valid dropoff at time t_{dropoff} in borough B, we identify the subsequent pickup at time $t_{\text{next\_pickup}}$ and compute:

$$\Delta t = t_{\text{next\_pickup}} - t_{\text{dropoff}} \tag{4}$$

subject to constraints $0 < \Delta t \leq 4$ hours (to exclude inter-session gaps).

4.2.2 Implementation

The computation employs both `lag()` and `lead()` window functions:

- **Lead computation:** For each trip, retrieve the timestamp of the driver's next pickup using `lead('pickup_ts')`
- **Wait time calculation:** Compute Δt as the difference between current dropoff and next pickup
- **Filtering:** Exclude invalid waits (negative, zero, or exceeding 4 hours)
- **Aggregation:** Group by `dropoff_borough` and compute summary statistics

4.2.3 Results

Borough	**Observations**	**Mean (min)**	**Median (sec)**	**Std Dev (sec)**
Manhattan	77,892	15.46	420	1,447.37
Brooklyn	2,569	35.25	1,380	2,237.67
Bronx	298	37.02	1,560	2,071.05
Queens	4,081	45.20	2,040	2,350.28
Staten Island	7	88.14	4,800	3,821.43

Table 6: Average time to next fare by dropoff borough

Key Observations:

Manhattan has by far the largest number of trips ($\approx$ 92% of valid records) and the shortest average waiting time (15 minutes), which is consistent with its high demand and central role in NYC. In contrast, outer boroughs such as Queens, Bronx, and Brooklyn show waiting times 2–3 times longer, indicating lower passenger density or longer repositioning times. Staten Island has very few records, with unusually long waits, likely due to limited taxi activity.

4.3 Query 3-4: Intra-Borough vs Inter-Borough Trips

4.3.1 Classification

Trips are classified according to the equality of pickup and drop-off boroughs (implemented and mentioned udf in section **2.2.2**):

$$\text{Trip Type} = \begin{cases} \text{Intra-borough} & \text{if } B_{\text{pickup}} = B_{\text{dropoff}} \\ \text{Inter-borough} & \text{if } B_{\text{pickup}} \neq B_{\text{dropoff}} \\ \text{Unknown} & \text{if } B_{\text{pickup}} = \text{Unknown} \vee B_{\text{dropoff}} = \text{Unknown} \end{cases} \tag{5}$$

4.3.2 Overall Distribution

Trip Type	**Count**	**Percentage**
Intra-borough	85,944	88.26%
Inter-borough	11,431	11.74%
Total	97,375	100.00%

Table 7: Distribution of trip types in D_{final}

4.3.3 Flow Analysis

The spatial distribution reveals asymmetric travel patterns:

Origin	**Dest.**	**Trips**	**%**
Manhattan	Queens	3,943	34.49
Queens	Manhattan	3,697	32.34
Manhattan	Brooklyn	1,923	16.82
Brooklyn	Manhattan	773	6.76
Queens	Brooklyn	597	5.22

Table 8: Top inter-borough flows

Borough	**Trips**	**%**
Manhattan	83,463	97.11
Queens	1,369	1.59
Brooklyn	1,062	1.24
Bronx	49	0.06
Staten Island	1	<0.01

Table 9: Intra-borough distribution

4.3.4 Key Observations

- **Dominant local usage:** Nearly 90% of trips remain within boroughs, indicating taxis serve primarily short-distance, localized needs.
- **Manhattan centrality:** Manhattan dominates with 97% of intra-borough trips and involvement in 89.5% of inter-borough travel.
- **Airport corridor:** The Manhattan-Queens connection is nearly symmetric (3,943 vs 3,697 trips), likely driven by JFK/LaGuardia airport traffic.
- **Asymmetric flows:** Manhattan receives 2.5× more trips from Brooklyn than it sends, reinforcing its role as a primary destination hub.

4.4 Overall Results

4.4.1 Utilization Rate (Query 1)

Dataset	Drivers	Mean (%)	Std Dev (%)	Trips/Driver
D_{strict}	9,846	53.03	25.55	9.89
$D_{\text{reasonable}}$	9,861	53.10	25.48	9.92
$D_{\text{complete\_temporal}}$	9,974	53.08	25.41	9.98
$D_{\text{manhattan}}$	9,660	53.77	25.84	9.74
D_{outer}	2,241	**88.60**	23.56	**1.47**

Table 10: Utilization stability across dataset configurations

Key Finding: Utilization metrics remain remarkably stable ($\approx 53\%$) across datasets. However, D_{outer} shows dramatically higher utilization (88.6%) but with far fewer trips per driver (1.47), suggesting outer-borough drivers operate in sparse, point-to-point mode rather than continuous urban cruising.

4.4.2 Time to Next Fare (Query 2)

Dataset	Borough	Observations	Mean (min)	Median (sec)
3*D_{strict}	Manhattan	77,892	15.46	420
	Brooklyn	2,569	35.25	1,380
	Queens	4,081	45.20	2,040
3*$D_{\text{extended\_spatial}}$	Manhattan	77,994	15.33	420
	Brooklyn	2,576	34.88	1,380
	Queens	4,097	44.88	2,040
3*D_{outer}	Brooklyn	450	**34.83**	**660**
	Queens	432	**41.76**	1,440

Table 11: Wait time consistency across spatial validation levels

Key Finding: Manhattan wait times are invariant (≈ 15 min) regardless of dataset selection, confirming demand density is unaffected by borderline spatial cases. Notably, D_{outer} shows *lower* wait times in Brooklyn and Queens when Manhattan trips are excluded, suggesting outer-borough drivers optimize routes differently when not competing for Manhattan fares.

4.4.3 Trip Flow Patterns (Queries 3-4)

Inter-borough flow proportions remain stable across all datasets:

- Manhattan-Queens corridor: 34.5% $\pm$ 0.2% across all configurations
- Queens-Manhattan return: 32.3% $\pm$ 0.1%

- Manhattan-Brooklyn: 16.8% ± 0.3%

The only significant deviation occurs in $D_{\text{manhattan}}$, where outer-borough internal flows are mechanically excluded, yielding Manhattan-Queens as 37.2% of *Manhattan-involved* inter-borough trips.

Conclusion

This project has established a data processing and analysis workflow. Through systematic enrichment, validation, and optimization (broadcasting, caching, and flag precomputation), I produced datasets suitable for large-scale geospatial and temporal analysis.

The results are consistent across data quality thresholds: the mean driver utilization remains close to 53%, with high stability across dataset variants. Manhattan clearly dominates taxi activity, combining the shortest average waiting times and the densest trip flows. Outer boroughs, by contrast, show longer idle times but higher utilization for fewer, more targeted trips—likely airport runs or scheduled transfers.