Université Claude Bernard Lyon 1

Data Processing and Analytics (DPA)

TP1 – Analyzing New York City Taxis Data

Authors: Lovato Edoardo, Markhovski Julia, Piccoli Leonardo Arduino

Academic Year 2025–2026

27 October 2025

1 Introduction

Urban mobility in New York City produces rich spatiotemporal patterns; understanding how ride counts, fares, and tips evolve across time, space, and vendor/provider attributes is key to answering practical questions (demand forecasting, driver earnings, surge windows).
This project analyzes a sample of the New York City taxi dataset using **Apache Spark SQL**. The main goal is to compute several analytical queries on the rides:

- **utilization rate** per taxi/driver pair;
- **average idle time** before the next trip per destination borough;
- number of **intra-borough** trips;
- number of **inter-borough** trips.

The purpose of this document is to explain the decisions that were made, and to present the results. All sections were written and refined cooperatively, with `Python` code (`Pandas` for analysis, `Matplotlib` for visualizations, `PySpark` for processing).

2 Preprocessing

Datasets Import and Schema Inference

This is essential to understand the underlying data and being able to process it accordingly, to obtain accurate results.
The dataset used includes two main sources:

1. a CSV file containing individual taxi trips with fields such as `medallion`, `hack_license`, `pickup/dropoff_datetime`, `passenger_count`, and coordinates;
2. a GeoJSON file defining the boundaries of the five NYC boroughs.

Exploratory Data Analysis (EDA)

The process begins by displaying the **head** of the Spark DataFrame to sanity-check parsing and column names. The sample contains 99,999 rows (pre-cleaning) and the NYC schema with timestamps (`pickup_datetime`, `dropoff_datetime`), trip attributes (`passenger_count`, `rate_code_id`), categorical flags (`vendor_id`, `store_and_fwd_flag`), and location identifiers (`pickup_longitude`, `pickup_latitude`, `dropoff_longitude`,`dropoff_latitude`).
Types do not correspond to the meaning of the value for all cells. For example, pickup and dropoff date time are not coded as timestamps in the dataset but as strings. Nevertheless, there were numeric types for pickup and dropoff coordinates, integers for passenger amount and rating, and strings for flags and IDs.

Next, the percentage of **missing values** per column shown in Figure 1 is computed. `store_and_fwd_flag` has the highest missingness (a legacy field that is uniformly N), while all other columns show negligible or zero missing values in this sample. Since `store_and_fwd_flag` is not used in the core metrics, it will not be imputed.

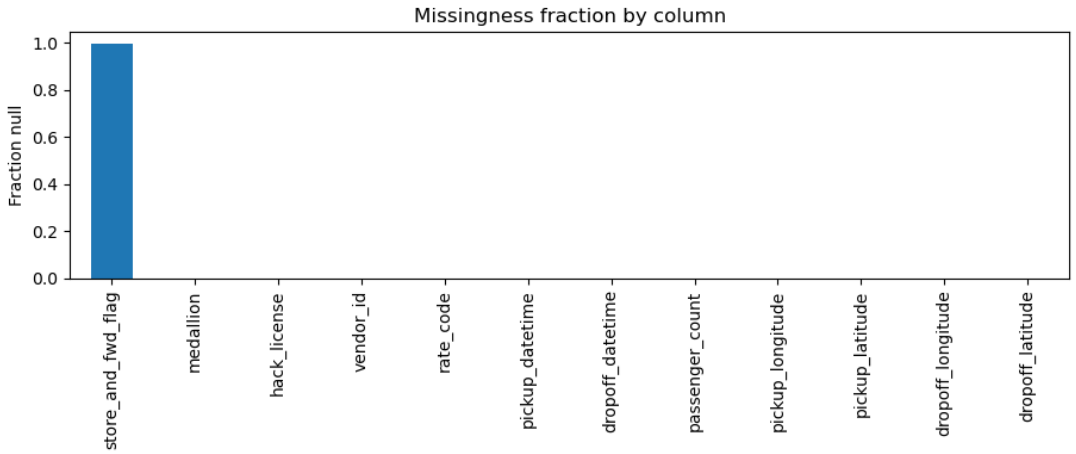

Figure 1: Missingness fraction by column.

Distinct values per categorical column are listed in Figure 2, following by a brief interpretation:

- `dropoff_latitude`: highest number of distinct values (more than 25000), since New York City is a very long city (from north to south), it confirms the expectations that latitude differs more than longitude;
- `pickup_latitude`: around 25000 distinct values, second highest variation;
- `dropoff_longitude`: around 15000 distinct values;
- `pickup_longitude`: as with latitude, pickup had a slightly smaller number of variations between the distinct values;
- `medallion` / `hack_license`: both are not unique alone which one can see in the number of variations of distinct values being only between 10000 and 15000. Hence, a combination of these two IDs are a unique ID is used;
- `pickup_datetime` / `dropoff_datetime`: around 2500 distinct values;
- `passenger_count`, `rate_code_id`, `vendor_id`: mostly standard;
- `store_and_fwd_flag`: primarily N, hence no variation within the values;

ID uniqueness side-note: it was considered to use only one of `medallion` (vehicle) or `hack_license` (driver) as a unique identifier; however, drivers can operate multiple vehicles and vehicles can be driven by multiple drivers. Therefore, the unit of analysis is modeled as the `(medallion, hack_license)` pair to avoid false merges and double counting.

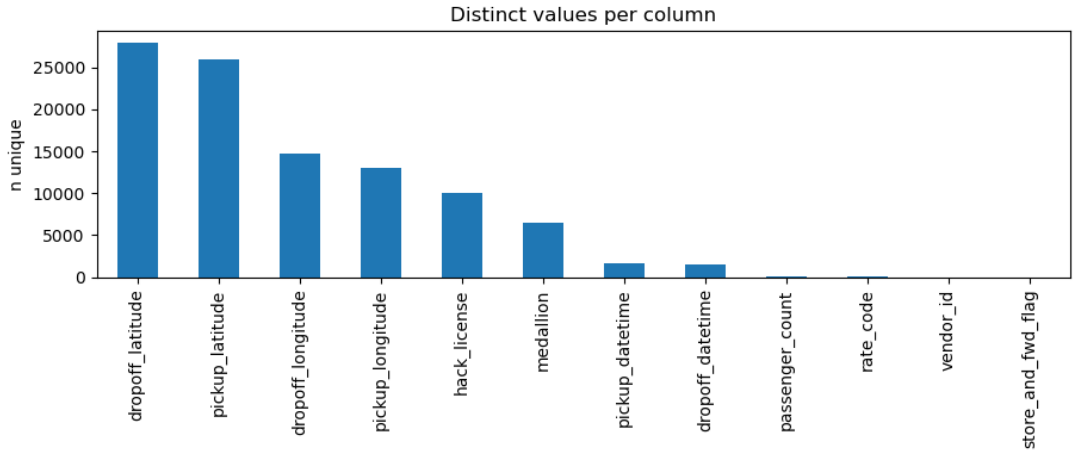

Figure 2: Cardinality by column.

Further, a full describe summary for **numeric fields** was run, particularly reviewing count, mean, std, min, max, zeros, negatives and quartiles.
The table in Figure 3 reports them for numerical columns of the database.

column	count	mean	std	min	max	zeros	negatives	q01	q05	q25	q50	q75	q95	q99
dropoff_latitude	99999	39.922430	14.806523	-3113.788800	652.72314	1901	2	-3113.788800	40.692448	40.736450	40.755222	40.769623	40.792145	652.72314
dropoff_longitude	99999	-72.553158	10.154353	-95.650002	0.00000	1917	98082	-95.650002	-74.005402	-73.991348	-73.980804	-73.963890	-73.870720	0.00000
passenger_count	99999	2.163002	1.739888	0.000000	6.00000	1	0	0.000000	1.000000	1.000000	1.000000	3.000000	6.000000	6.00000
pickup_latitude	99999	40.048643	5.795548	0.000000	646.43829	1749	0	0.000000	40.675812	40.735104	40.755291	40.769539	40.790394	646.43829
pickup_longitude	99999	-72.655734	9.784012	-98.116669	0.00000	1770	98229	-98.116669	-74.005600	-73.991867	-73.980858	-73.964722	-73.870750	0.00000
rate_code	99999	1.042100	0.285498	1.000000	5.00000	0	0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	5.00000

Figure 3: Descriptive Statistics.

Key Observations and Decisions

Latitude and longitude are well-behaved in the bulk but have extreme outliers. The central quantiles line up with NYC:

- Pickup latitude: $q_{25} \approx 40.735$, $median \approx 40.756$, $q_{75} \approx 40.770$;
- Pickup longitude: $q_{25} \approx -73.992$, $median \approx -73.980$, $q_{75} \approx -73.964$.

Dropoff quantiles are virtually identical; this is exactly the expected NYC range.
However, min/max show obvious outlier values (e.g., min `dropoff_latitude` ≈ -3113, max `pickup_latitude` ≈ 646). These are treated as invalid coordinates and removed later: $40.496 \leq$ latitude ≤ 40.9155 and $-74.25559 \leq$ longitude ≤ -73.7. Furthermore, zeros (e.g. $0.0, 0.0$) are also regarded as invalid coordinates.

`passenger_count` is plausible on average but includes edge cases; mean ≈ 2.16, median $= 1$, max $= 6$. There is a single zero reported in the "zeros" column; 1–6 are kept as the valid range.

`rate_code` is concentrated at the standard rate. Mean ≈ 1.04, with $q_{99} = 5$, indicating a small

mass at special fares.

These statistics confirm that the substance of the data is geographically and operationally consistent with NYC taxi trips, and they motivate the rules being applied in the cleaning step.

Histogram Plots

Plotting the colums values emphasize where trips occur and basic trip composition:

- pickup/dropoff latitude histograms: narrow peaks around 40.7–40.8° with tiny spikes at 0 or extreme values. The spikes validate our choice to drop coordinates outside the NYC box and zeros treated as invalid positions;
- pickup/dropoff longitude histograms: mass centered near −73.98° with a long but thin tail; a small pile-up at 0 marks invalid coordinates. Again, the boundary filter addresses this;
- passenger count bar chart: dominant modes at 1 passenger; long tail to 6; vanishingly few zeros. This supports keeping a valid range of [1,6];
- rate code bar chart: heavily concentrated at 1 (standard) with small bars at [2,5];
- pickup/dropoff hour histograms: clear midday peak (around noon one sees the highest number of trips) with secondary shoulder peaks bracketing commuting hours. Dropoff hours mirror pickup hours with a slight right-shift.

To quantify concentration, the Top-30 values are listed for selected fields:

- medallion / hack license: classic long-tail since a few IDs account for many trips, while many IDs appear infrequently. Because neither ID is unique by itself, the `(medallion, hack_license)` pair is treated as the unit of analysis for utilization and sequencing. Aggregations to "per-driver" or "per-vehicle" are done explicitly from this pair;
- pickup / Dropoff hour: noon ranks 1 in both series (most rides), followed by late morning/early afternoon hours; nighttime hours contribute least. This matches the histograms and supports later time-based slicing (e.g., peak vs. off-peak).

Correlation analysis and scatter-matrix

A Pearson correlation matrix was computed over the numeric fields available in this vintage:

- `pickup_latitude`, `pickup_longitude`
- `dropoff_latitude`, `dropoff_longitude`
- `passenger_count`, `rate_code`

A scatter-matrix (pairplot) was also produced, with histograms on the diagonal and 2D scatter plots off-diagonal.

Findings.

- **Strong cross-axis spatial coherence.** High values were observed for corr(`pickup_latitude`, `dropoff_longitude`) and corr(`pickup_longitude`, `dropoff_latitude`). This indicates that the pickup location on one axis is predictive of the dropoff location on the other axis. In the scatter-matrix, the corresponding panels appear as tight, slanted point clouds. NYC flows are not axis-aligned: many trips follow diagonal corridors (Manhattan's NE–SW spine, crosstown connectors, bridge/tunnel approaches). Because latitude and longitude co-vary along these diagonals, a northward pickup (higher latitude) often pairs with an eastward or westward dropoff (longitude shift), and vice versa, yielding strong cross-axis correlations.

- **Near-zero correlations for party size vs. location.** The variable `passenger_count` shows very weak correlation with either latitude or longitude. In the scatter-matrix, this appears as horizontal bands at integer counts (1–6) with no obvious spatial drift, suggesting that while demand is location-dense, party size is largely independent of exact coordinates.

- `rate_code` **shows geographic structure, but Pearson is not ideal.** Since `rate_code` is an encoded category, Pearson correlations with latitude/longitude are modest even though the scatter-matrix reveals clusters: for example, points near JFK coordinates $\sim (40.64, -73.78)$ are enriched for the JFK code, and a thin cluster near Newark longitudes appears for its code. For a single-number association, a categorical test (e.g., ANOVA/Cramér's V) would be more appropriate than Pearson.

Data Cleaning

After having gained an understanding of the underlying and given the results from the EDA, the dataset is trimmed to the variables that are relevant to the assignment: timestamps, geographical coordinates, passenger composition and identifiers, by removing legacy flags that did not inform any of the analyses.

In particular, `vendor_id`, `rate_code`, and `store_and_fwd_flag` were dropped from the working table. The first two are provider / tariff descriptors that do not change how utilization is computed, time-to-next-fare, or within- vs. cross-borough flows; the third is a legacy transmission flag that was largely missing in the sample.
Retaining them would have increased width without adding signal, so they are excluded from subsequent steps and figures.

Timestamps were the first hard constraint. Trip duration is computed as `dropoff_time - pickup_time`, after converting them in Unix timestamp, and delete any row where the drop-off timestamp was less than or equal to the pickup timestamp, i.e., non-positive duration. Also all trips with duration $>$ 4h are removed, as they are considered outliners.
These records are not interpretable for queuing, utilization, or spatial flow, and keeping them would distort both averages and percentiles.

Passenger composition was handled with a simple plausibility rule: because the yellow-cab cabin seats at most six, trips with `passenger_count` in [1, 6] are kept and the rare out-of-range values (zeros or negatives, and anything above six) are removed. This preserves the empirical distribution (modes at 1–2) while preventing invalid counts from leaking into descriptive statistics or groupings. `passenger_count` is not imputed; given the rarity of invalid entries, deletion has negligible impact and avoids inventing information.
Given that the queries require only to know if and when a taxi is occupied, and not by how many peoples, the column `passenger_count` can be deleted from the table after being cleaned.

Finally, basic geographic sanity is enforced on the legacy latitude / longitude fields. Quantiles clustered tightly around NYC's coordinates, but the raw min/max revealed obvious garbage. Therefore, only a conservative NYC bounding box (approximately $40.4 \leq$ latitude ≤ 41.0, $-74.3 \leq$ longitude ≤ -73.6) is kept. This step removes a small number of rows with corrupted GPS while leaving the core distribution intact.

After these filters, the dataset contains only trips with valid time ordering, plausible passenger counts and NYC-consistent coordinates.
All following queries are computed on this cleaned table.

3 GeoJSON Integration and Spatial Enrichment

The GeoJSON file was loaded as a json dataframe with `Spark-SQL`; each line contains the name and the code of a specific borough with a list of **coordinate points** determining its perimeter. To optimize lookups, the polygon with the largest number of points was selected for each borough, assuming it represents the **main region**.
Then the rows with highest cardinality of points (given an unique `boroughID`) were exported from the dataframe and saved in a normal Python list, with each element containing the borough name, code and the list of its points.

The geospatial analysis combines data using the `Shapely` library to identify the shape of each borough to associated with each pickup and drop-off point.
The lists of points (one for each borough) were converted into a `shapely.Polygon` object and a User-Defined Function (UDF) called `findBorough(longitude, latitude)` was implemented. This function determines whether a pickup or drop-off point lies within a given borough polygon and allows to add the columns `pickup_borough` and `dropoff_borough` in the dataframe of taxi trips.

After finding the five boroughs of NYC, the maximum and minimum value for latitude and longitude were computed.
These are needed to delete from the dataframe all rows with coordinate **outliners**.
Some rows with pickup and/or drop-off locations not belonging to NYC were still maintained (but in a short distance from the city) to represent that some trips can start or end outside of the town. These were marked as `"OUTSIDE NY"`.

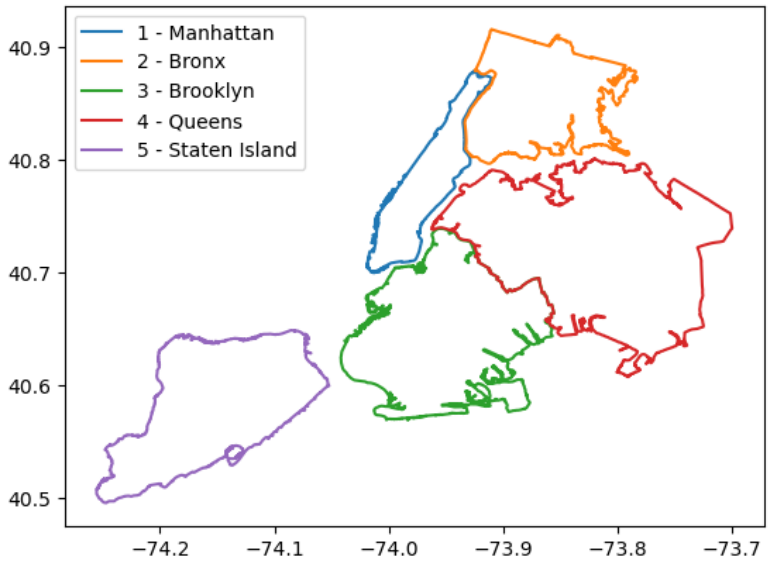

Figure 4: Plot of the NYC boroughs.

4 Queries Implementation and Results

Query 1 – Calculation of Utilization Rate

Key Selection

To uniquely identify each operational unit of the taxi service, the composite key `(medallion, hack_license)` was selected. This combination represents the vehicle and the driver, respectively. It avoids ambiguities when multiple drivers use the same vehicle or a driver operates different vehicles over time, this ensures that temporal measurements are consistent and refer to a single vehicle–driver combination. The dataset contains 10,006 unique combinations of `medallion` and `hack_license`.

Objective

The purpose of this query is to estimate **utilization**, defined as the proportion of time a taxi is occupied with passengers relative to the total observed time. This metric allows evaluation of operational efficiency and vehicle productivity.

Calculation Method

Trips were chronologically ordered by `pickup time` within each taxi–driver group. Using Spark **window functions**, specifically `lag()`, the drop-off time of the previous trip was retrieved to compute the time interval until the next trip. This interval represents the **idle time**.
Intervals were excluded if:

- negative, due to timestamp errors;
- longer than four hours, likely representing shift ends or long breaks.

For each taxi–driver pair, the following were calculated:

- **occupied time**: sum of trip durations (in milliseconds);
- **idle time**: sum of valid intervals between trips.

The utilization is then computed as:

$$\text{utilization} = \frac{\text{occupied time}}{\text{occupied time} + \text{idle time}}$$

Results and Observations

The results show a high variability in utilization values across taxi/driver units. Some units exhibit values close to 1, indicating nearly continuous activity, while others show very low values (below 0.1), reflecting long periods of inactivity or extended operational breaks.
Figure 5 shows the **relative frequency distribution of utilization** among all units analyzed. About 17.5% of units have utilization equal to 1, while a large fraction of taxis/drivers show values between 0.3 and 0.6 (realistic range).

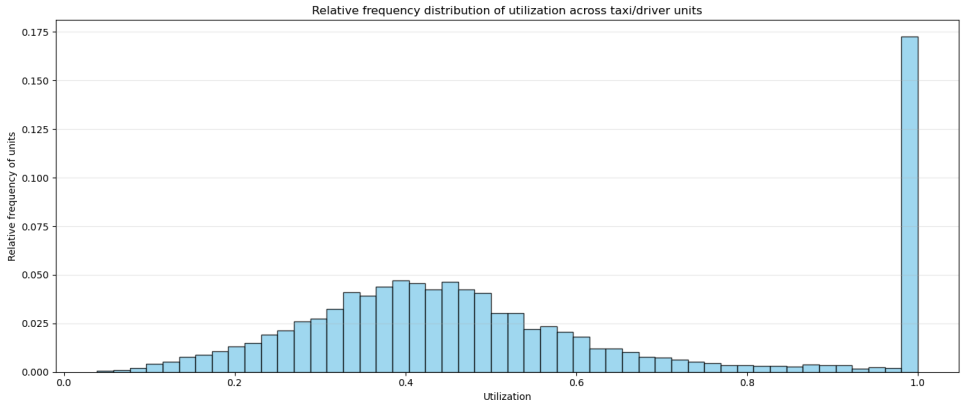

Figure 5: Relative frequency distribution of utilization across all taxi/driver units.

Side-note: caution is required when interpreting these results. In particular, a utilization of 1 does not necessarily indicate sustained continuous activity; it may result from a small number of closely spaced trips, while periods of inactivity exceeding 4 hours are not counted as idle and correspond to actual rest or absence of operational activity.

Figure 6 shows the **timeline of three examples of taxi/driver units**:

- The first unit (bottom) has **low utilization**, with few trips and long inactivity intervals.
- The second unit represents a **median case**, with a balanced alternation of occupied and idle periods ($\leq$ 4h).
- The third unit has **high utilization (=1)**, but represents only 2 trips and periods of inactivity exceeding 4 hours.

Blue segments indicate **time occupied with passengers on board**, while orange segments represent **idle periods of $\leq$ 4 hours**. Periods of inactivity longer than 4 hours are considered true rest or absence of operational activity and do not contribute to utilization.

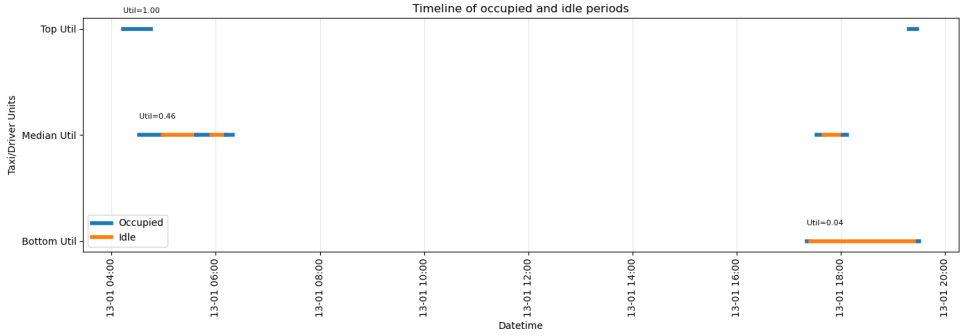

Figure 6: Timeline of occupied (blue) and idle (orange) periods for three instances.

This visualization highlights that the **density of trips over the operational time** is more relevant than the total number of trips: two closely spaced trips can lead to utilization = 1, even if overall trips are few, whereas long inactivity intervals drastically reduce utilization.

General Interpretation of Utilization Values

- **High values (≈ 1)**: indicate no detected idle $\leq$ 4h between trips. This does not always imply high overall productivity, especially if the dataset is partial or trips are few but closely spaced.
- **Low values ($<$0.1)**: reflect long inactivity periods or few recorded trips. Interpretation should consider both the number of trips and their temporal distribution.
- **Realistic range (0.3–0.6)**: represents the majority of units, showing a plausible balance between occupied time and operational breaks.

Observation: to correctly interpret utilization, it is essential to consider both the **number of trips** and the **temporal sequence of trips**, distinguishing between idle periods $\leq$ 4h and actual inactivity $>$ 4h.

Query 2 – Average Waiting Time to Find the Next Passenger

Objective

The aim of this query is to estimate the **average time required for a taxi to find the next passenger** after completing a trip, computed per drop-off borough. This provides insights into operational efficiency in relation to the geographic location of trips.

Calculation Method

Trips are chronologically ordered for each taxi/driver pair (same as the previous query). Using the Spark `lead()` function, the pickup timestamp of the subsequent trip is retrieved to compute the **waiting time**.
Only intervals satisfying all the following conditions are considered valid:

- the next trip exists (`next_pu_ts` not null);
- the interval is positive;
- the interval is less than or equal to 4 hours (maximum idle time considered).

For each drop-off borough, the following are calculated:

- **number of trips** recorded;
- **average waiting time** in milliseconds between drop-off and the next pickup.

Results and Observations

As shown in Table 1 and Figure 7, Manhattan exhibits a significantly lower average waiting time, indicating high passenger demand and short idle periods. In contrast, Staten Island and Queens show much higher average times, but the number of trips in these boroughs is very small (Staten Island only 8 trips), so these values should be interpreted with caution.
Intervals greater than four hours are excluded from the calculation; therefore, the reported averages reflect only the actual operational waiting time between trips.

Borough	Number of Trips	Avg. Waiting Time (ms)	Avg. Waiting Time (min)
Bronx	300	2,203,600	36.7
Brooklyn	2,589	2,084,681	34.7
Manhattan	79,072	906,048	15.1
OUTSIDE NY	339	2,328,142	38.8
Queens	4,102	2,680,166	44.7
Staten Island	8	4,710,000	78.5

Table 1: Average waiting time to find the next passenger, per drop-off borough.

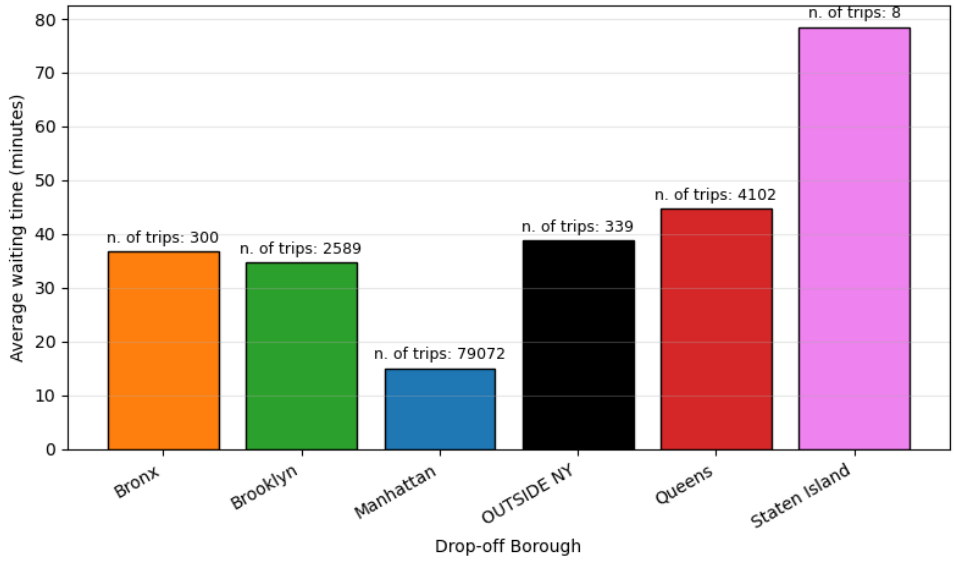

Figure 7: Average waiting time to find the next passenger (minutes), by drop-off borough.

General Interpretation

- **Short waiting times** (e.g., Manhattan ≈ 15 min) indicate high operational efficiency and frequent passenger requests, particularly in central zones.

- **Long waiting times** (e.g., Staten Island ≈ 78 min) may reflect low passenger density, limited dataset, or long periods of true inactivity excluded from the calculation.

This measure is useful to understand the spatial distribution of operational efficiency, but must always be interpreted together with the number of trips to avoid misleading conclusions for drivers with very few recorded trips.

Query 3 – Trips Starting and Ending in the Same Borough

Objective

The objective of this query is to determine how many taxi trips start and end **within the same borough**, and to measure their percentage over the total number of recorded trips. This analysis provides insights into the **local nature of mobility** across New York City, identifying areas where short, intra-borough movements are most common.

Calculation Method

Records with both pickup and drop-off boroughs not null are selected, then only the trips where `pickup_borough = dropoff_borough` are kept.
For each pickup borough, the following are computed:

- **Number of same-borough trips**;
- **Percentage over the total number of trips**.

Results are finally ordered by descending number of same-borough trips.

Results and Observations

Pickup Borough	Same-Borough Trips	% over Total Trips
Manhattan	83,418	85.28%
Queens	1,362	1.39%
Brooklyn	1,062	1.09%
Bronx	49	0.05%
OUTSIDE NY	48	0.05%
Staten Island	1	0.001%

Table 2: Number and percentage of same-borough trips, per pickup borough.

As shown in Table 2 the vast majority of same-borough trips originate from **Manhattan** (over 85% of the total). This clearly reflects the dense concentration of passenger demand and short-distance mobility within the central area of New York City. Queens and Brooklyn follow with minor shares (around 1–1.4%), while other boroughs—such as Bronx and Staten Island—show negligible counts, indicating limited intra-borough taxi activity.

General Interpretation

- A very high proportion of same-borough trips in Manhattan indicates intense **local demand** and **short-distance urban mobility**.
- Low values in outer boroughs suggest that many trips are directed toward other areas, particularly Manhattan.
- This metric helps identify where taxi operations are primarily local versus cross-borough.

Query 4 – Trips Crossing Different Boroughs

Objective

This query aims to identify and quantify the taxi trips that start in one borough and end in another, i.e., **cross-borough trips**. The goal is to capture the **inter-borough mobility patterns** and understand the extent of spatial connectivity between different areas of the city.

Calculation Method

First, the data is restricted to records that have a valid pickup and a valid drop-off borough. From this subset, only the trips where the pickup borough differs from the drop-off borough are retained. Finally, for each pickup borough, the number of such cross-borough trips is computed and expressed as a percentage of the total number of trips.

Results and Observations

Pickup Borough	Cross-Borough Trips	% over Total Trips
Manhattan	6,413	6.56%
Queens	4,450	4.55%
Brooklyn	896	0.92%
OUTSIDE NY	89	0.09%
Bronx	29	0.03%
Staten Island	1	0.001%

Table 3: Number and percentage of cross-borough trips, per pickup borough.

As seen in Table 3 Manhattan again dominates as the origin for most trips, including those directed to other boroughs (over 6.5% of the total). Queens shows a significant share of cross-borough movements (4.5%), likely due to airport connections and inter-zonal mobility. In contrast, other boroughs contribute only marginally to cross-borough traffic.

General Interpretation

- Cross-borough trips reflect longer-distance connections, often toward or from Manhattan and major transport hubs (e.g., JFK, LaGuardia).
- Comparing Query 3 and Query 4 highlights a clear **spatial polarization of taxi demand**: local and dense within Manhattan, more dispersed and directional elsewhere.

To complement these results, Figure 8 compares the distribution of same- and cross-borough trips side by side. This visualization highlights the predominance of intra-borough trips in Manhattan and the more balanced proportions in Queens and Brooklyn.
The Origin–Destination heatmap (Figure 9) visualizes the flow intensity between pickup and drop-off boroughs. Each **row represents the origin** (pickup borough) and each **column represents the destination** (drop-off borough). The diagonal corresponds to same-borough trips, which are particularly high in Manhattan, indicating that most rides start and end within the same area. Off-diagonal cells represent inter-borough flows, mainly from Manhattan to Queens and Brooklyn, confirming Manhattan's central role as the primary origin and destination hub in the NYC taxi network.

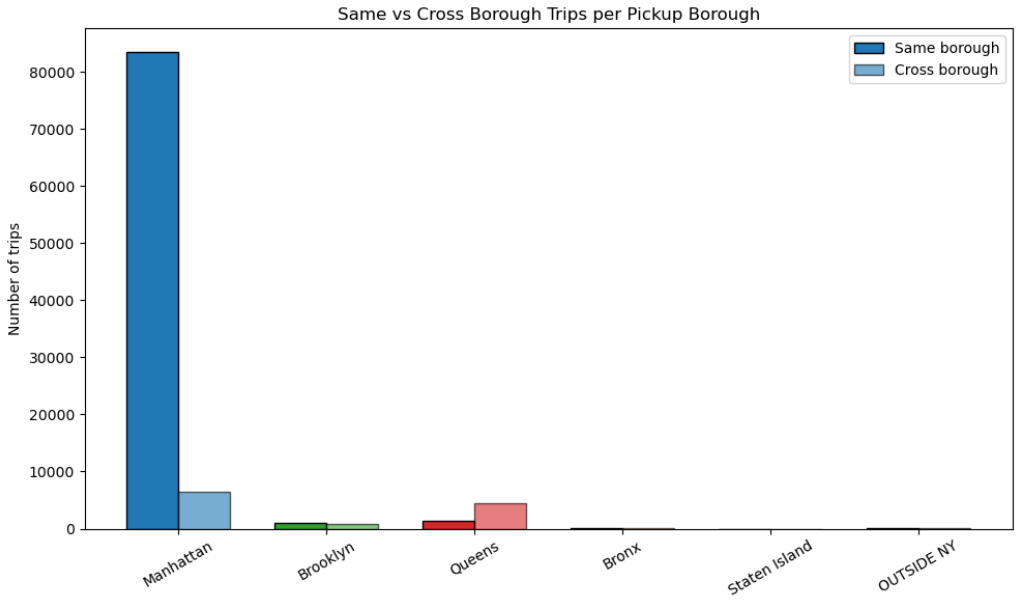

Figure 8: Comparison of same-borough and cross-borough trips by pickup borough.

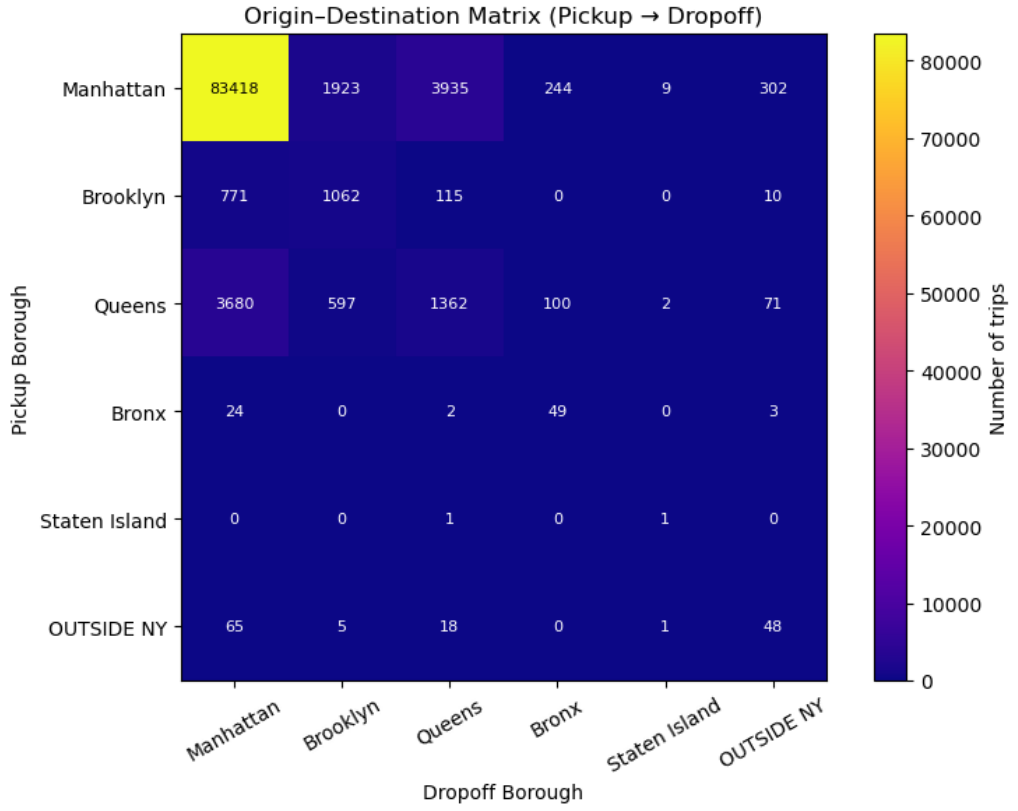

Figure 9: Origin–Destination heatmap of taxi trips by borough.

5 Conclusion

The project demonstrated how to process and analyze large-scale spatio-temporal datasets using Spark. By combining taxi trip data with borough boundaries, metrics of utilization, idle times, and trip flows (same-borough vs. cross-borough) were computed. Quantitative results show that Manhattan dominates taxi activity with minimal idle times and high internal flows, while peripheral boroughs exhibit lower utilization and longer idle periods. The analysis of the Origin–Destination matrices highlights the predominant flows between boroughs.

Limitations and Further Improvements

The pipeline treats each pickup–dropoff record as a separate trip, even when journeys are effectively pooled along the same route. For example, a taxi leaving Penn Station toward Columbus Circle that accepts an additional fare en route (with available seats) will appear as two distinct trips in our data. This overstates trip counts in rare pooled scenarios; a future enhancement could detect overlapping segments (same vehicle, contiguous times, similar paths) and aggregate them into a single shared ride.