Train Trip Analytics Dashboard

Technical Report & Reflection

Prepared by Yves Niyigena

1. Problem Framing and Dataset Analysis

NYC Taxi Trip dataset contains millions of city movement data records like timestamps, trip distance, trip duration, fare, and GPS location. These are facts that form a rich source of vendor performance, passenger activity, and service reliability.

Data Challenges:
The data set had missing coordinates, incorrect timestamps, and unusually high outliers for fare amount and trip time. There were also numerous duplicates in the data resulting from overlapping collection intervals. Validation checks and deletion rules were used to ensure the integrity of the data.

Assumptions Made:
Shorter trips of less than 1 minute or longer trips of over 3 hours were treated as anomalous and removed. Missing vendor IDs were marked "Unknown Vendor" in order to be consistent.

Unexpected Observation:
We saw that the performance of vendors differ considerably at different times of the day, which is the reason behind our decision to incorporate hourly KPIs and time-based filters in the dashboard.

2. System Architecture and Design Decisions

Our architecture is designed on a three-layer model with a Flask backend, SQLite database, and JavaScript-powered frontend. The reason for choosing Flask is that it has a light-weight framework and is easy to use for RESTful API implementation. SQLite was chosen because of its easy setup process and most efficient local storage management.

Frontend	Backend	Database
HTML, CSS, JS, Chart.js	Flask REST API	SQLite (Normalized Schema)

Stack Choices

Backend: Python Flask REST API for KPI and analytics endpoints with date filtering. Frontend: HTML, CSS, and JavaScript for interactive dashboard. Chart.js to display. Database: SQLite for local storage with relational schema.

3. Algorithmic Logic and Data Structures

Manual filtering algorithm was used to get trip records for a specified date range without relying on filtering libraries.

Pseudo-code Implementation:

function filterTripsByDate(trips, startDate, endDate): result = [] for trip in trips: if trip.date >= startDate and trip.date <= endDate: result.append(trip) return result

Complexity Analysis:

Time Complexity: O(n) — each record is read once. Space Complexity: O(k) — where k is the number of trips within the date range.

4. Insights and Interpretation

The following three key insights were derived from data analysis:

1. Peak Hours:
The majority of the trips were between 5–7 PM and 8–10 AM, which reflects the peak commute times.

2. Vendor Performance:
Vendor A always had higher levels of trips but lower mean fares per trip than Vendor B.

3. Trip Duration:
Mean trip durations were longer on weekends, which reflects possible congestion or recreational travel patterns.

5. Reflection and Future Work

It was an experience to integrate data cleaning, API creation in the backend, and charting in the frontend in one live system. Synchronization of filtered data with updation of charts in real-time was the main challenge. Improvement areas for the future can be hosting a cloud database (PostgreSQL) and integrating machine learning models for providing predictive insights on vendor loyalty and passenger movement.