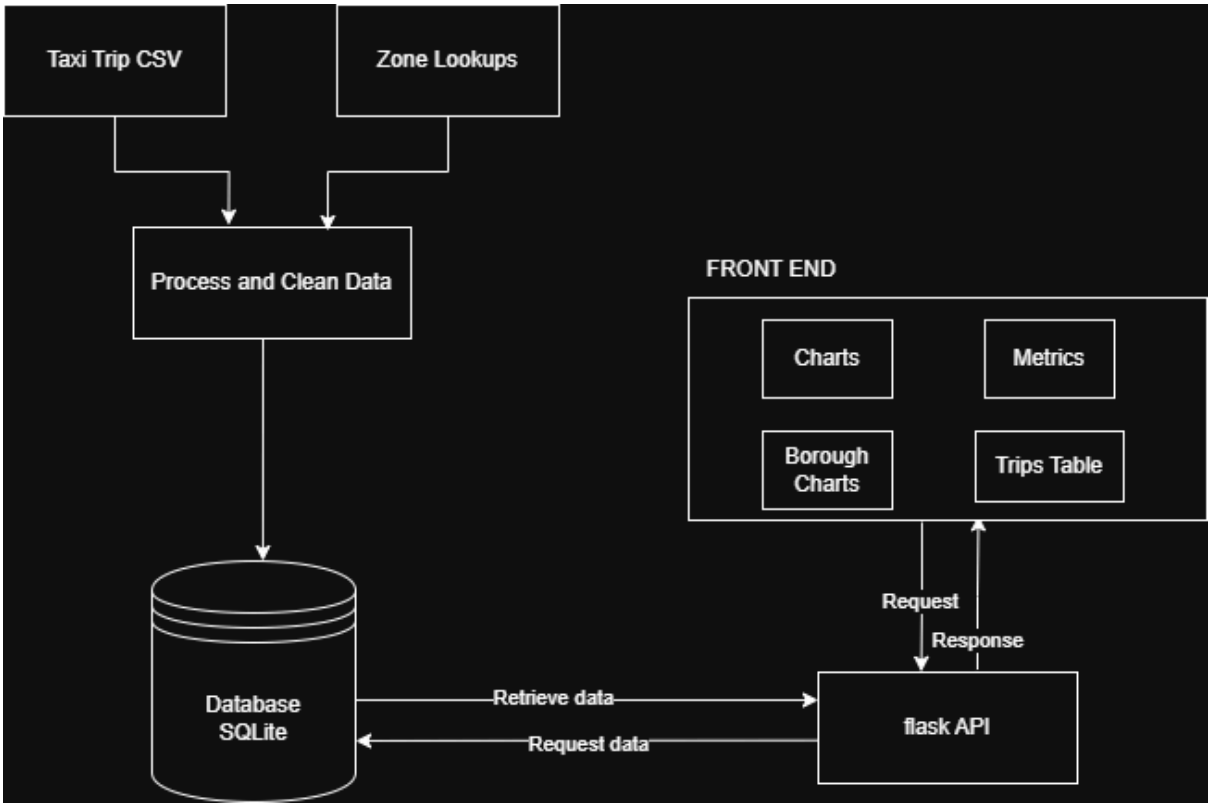
Technical Report

1. Problem Framing and Dataset Analysis

Dataset Context: This project examines the NYC Yellow Taxi dataset for January 2019, which comprises about 7.7 million records. The aim is to develop an interactive dashboard that provides insights into urban mobility trends (volume, revenue, and geography).

Data Challenges:
Scale: Performing JOIN operations on 7.6M records took like 5 minutes.
Latency: Real-time filtering on Borough and Date Range demanded a complete overhaul of database access patterns.

2. System Architecture and Design Decisions



Frontend
HTML
CSS
Vanilla JS

Backend
Flask REST API

SQLite

Design Decisions:
We create a parser that processes, cleans up, and stores the data in the database. So, instead of loading the data once, we load it in chunks of 100000, so that our system will not crash. Then we use the flask api to retrieve the data from the database and display it on the frontend.

3. **Algorithmic Logic:**

We implement an algorithm to manage the pagination of over 150000 pages.

Approach:
The algorithm calculates which page numbers should be visible based on the currentPage. It ensures the first page, last page, and immediate neighbors are always accessible while hiding the middle with ellipses (...).

```
python
# Function: getVisiblePages(current, total, delta=2)
pages = []
for i from 1 to total:
   if i == 1 or i == total or (i >= current - delta and i <= current + delta):
      pages.append(i)
   elif pages[-1] != "...":
      pages.append("...")
return pages
```

4. **Insights and Interpretation**

Summary statistics: They were computed from the pre-aggregated trip_summary table using the GET endpoint /api/stats:

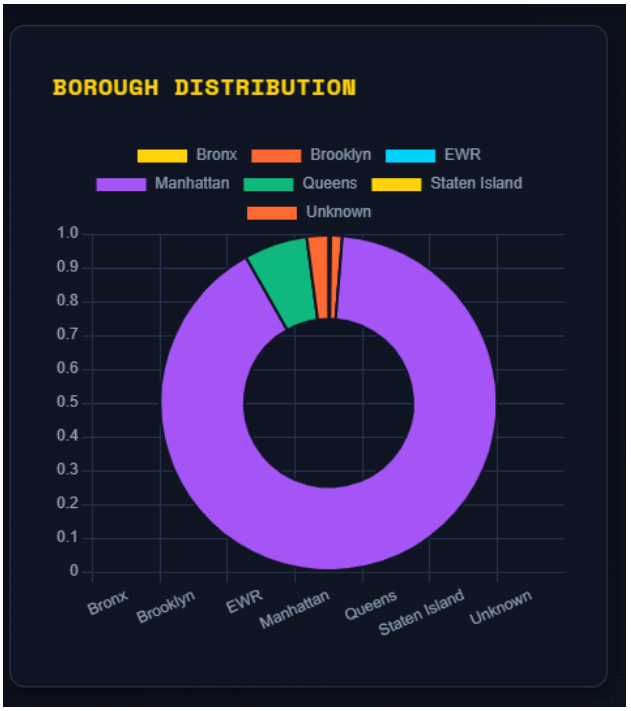
This endpoint:

- Total trip counts
- Total distance
- Total revenue
- Average speed

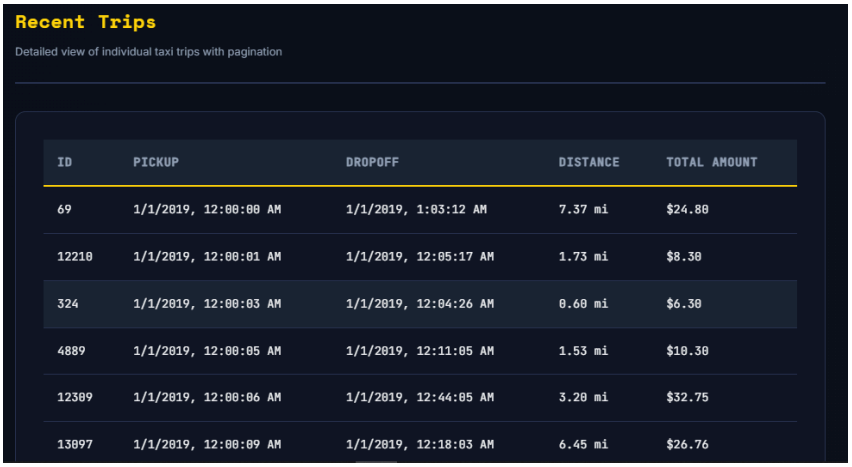
Borough distribution: We calculated it by;

SELECT borough, SUM(trip_count)
FROM trip_summary
GROUP BY borough

Then access the data through this endpoint /api/patterns-borough



Trip Table: Retrieved all the trips from the database and displayed them on a table with the GET: endpoint /api/trips



5. Reflection and Future Work

Technical Challenges: Initially, we used MYSQL and PostgreSQL database but it was taking forever to parse the data and store it in the database. After about an hour, it will terminate. Then we had to switch to SQLite.

Future Work:

To improve the ui design more and add more functionality.