Urban Mobility Data Explorer – Technical Report

Team Name: Group 4

Overview

This project presents a full-stack system for exploring New York City taxi trips from 2019. Raw trip data is processed through a structured data pipeline, stored in a relational database, and exposed through a backend API and interactive frontend dashboard. Custom algorithms demonstrate manual sorting and filtering logic. The system transforms urban mobility data into actionable insights through careful data engineering and system design.

---

1. Problem Framing and Dataset Analysis

1.1 Dataset Context

The dataset contains millions of taxi trip records, including:

- Pickup/dropoff timestamps
- Trip distance, fares, and payment details
- Passenger count and geographic zone identifiers

Additional data includes a taxi zone lookup table and GeoJSON files for zone boundaries.

The goal is to design a system for analyzing urban mobility patterns and economic activity via a database-driven backend and interactive dashboard. Due to dataset size and heterogeneity (CSV, Parquet, GeoJSON), a structured preprocessing pipeline was necessary.

1.2 Data Challenges

Key challenges included:

- Missing essential values (passenger count, fares, timestamps)
- Invalid records (negative or zero distances, fares, illogical timestamps)

- Duplicate trips
- Invalid pickup/dropoff zones
- Mixed timestamp formats
- Large CSV files causing slow row-by-row parsing

Solution: CSV files were converted to **Parquet** for faster columnar access, reduced memory usage, and scalable processing.

1.3 Cleaning and Feature Engineering

The pipeline consists of: **data loading → cleaning → feature engineering → exclusion tracking**.

Cleaning rules:

- Remove records with missing essential values
- Remove invalid trips and duplicates
- Join zone metadata; remove unmatched zones
- Standardize categorical fields

Feature engineering:

- Trip duration (minutes)
- Average speed (km/h)
- Revenue per km
- Pickup hour/day and peak-hour indicators

Exclusion tracking: All removed records are stored in `excluded_records.csv` with reasons and pipeline stage, ensuring transparency and reproducibility.

1.4 Unexpected Observations

Some trips had short durations but unusually high fares. This led to the creation of **revenue per km**, enhancing anomaly detection and economic interpretation. Derived features such as average speed and peak-hour indicators link trips to real-world commuter behavior.

---

2. System Architecture and Design Decisions

2.1 Architecture Overview

The system follows a layered architecture:

1. **Raw Data Layer:** Original CSV/Parquet and GeoJSON data
2. **Data Processing Pipeline:** Cleaning, normalization, and feature engineering
3. **Database Layer:** SQLite relational schema with indexing for efficient queries
4. **Backend API Layer:** Flask REST API for dynamic SQL queries and custom algorithms
5. **Frontend Dashboard:** Interactive visualization and spatial exploration

This design ensures modularity, traceability, and separation of concerns.

2.2 Technology Choices

- **Python & Flask:** Lightweight backend and REST API
- **SQLite:** Embedded relational database for portability
- **Pandas:** Data cleaning and feature engineering
- **JavaScript & GeoJSON:** Frontend interactivity and spatial visualization

2.3 Database Schema

Normalized schema (3NF) with **fact table (trips)** and **dimension table (zones)**:

Zones Table: zone_id (PK), borough, zone_name, service_zone
Trips Table: trip_id (PK), pickup/dropoff timestamps, distance, fare, passenger_count, PULocationID/DOLocationID (FK → zones), trip_duration, speed, revenue per km, pickup hour/day

Indexes: pickup_datetime, fare_amount, trip_distance, PULocationID, DOLocationID

Design justification: SQLite is lightweight, serverless, compatible with Python/Flask, supports indexing and constraints, and simplifies deployment while demonstrating academic algorithmic understanding.

---

3. Algorithmic Logic and Data Structures

Objective: Rank and filter taxi trips manually.

- **Custom Bubble Sort:** Orders trips by fare amount ($O(n^2)$ time, $O(1)$ space)
- **Manual Distance Filter:** Selects trips within user-defined distance range ($O(n)$ time, $O(n)$ space)

Backend applies these algorithms dynamically when enabled, ensuring algorithmic transparency and assignment compliance.

---

4. Insights and Interpretation

1. **Peak Travel Hours:**
    - Morning: 7–9 AM, Evening: 5–7 PM
    - Reflects commuter demand cycles
2. **Revenue per Distance:**
    - Central Manhattan trips yield higher revenue per km
    - Dense traffic and high demand increase economic efficiency
3. **Spatial Hotspots:**
    - Midtown Manhattan and JFK Airport as major pickup zones
    - Key transportation and economic hubs

Summary:
This project demonstrates the complete lifecycle of urban mobility data—from raw ingestion, cleaning, and feature engineering to relational storage, backend processing, custom algorithms, and interactive visualization—transforming transactional trip data into meaningful insights for analysis.