NYC Taxi Data

Fullstack application project documentation
Github repo: https://github.com/Linda5-umwali/NYCity-Taxi-App.git

1. Problem framing and dataset analysis

The NYC taxi dataset contains detailed trip records which include timestamps, distance, durations, fares, pickup and dropoff locations, as well as other metadata. The main challenge I faced was cleaning and organising the large files, mainly the “train.csv” file that contains the trip records mentioned above. Also, I encountered data issues where there were missing information values, and duplicate entries and outliers in distance and duration. These were later managed using processing scripts through python. The cleaning steps removes duplicates and rows with missing values, and it also drops invalid trips(trip distance < 0.1km, trip duration < 60seconds)
Delivered attributes were:

- Trip distance: is found using haversine formula
- Fare_amount: is estimated using simple pricing model(time, distance, and passenger surcharges)
- Trip speed: calculated using distance covered and time in km/h.
- Fare per kilometer: calculated as fare / distance travelled.
- Pickup hour: extracted from pickup_datetime. Peak taxi demand was between 18:00 - 21:00, which shows strong evening usage patterns.
- Speed_outlier: flag for trips with speed over 80 km/h

2. System architecture and design decisions

The architecture of the project system is fullstack and has three main/ general components:

- Backend: Flask API for data processing and MySQL integration.
- Database: MySQL relational database with cleaned and enriched trip data.
- Frontend: HTML, CSS, and JavaScript dashboard for user interaction and visualization.

The schema was normalized to separate trip data and fare metrics for optimal querying. MySQL was utilized for scalability and indexing.
Flask was utilized for lightweight, rapid API endpoints to feed the processed data to the frontend.

3. Algorithmic Logic and data structures

A proprietary algorithm was utilized to detect speed outliers trips with excessively high or low speeds compared to the mean. The rationale calculates mean and standard deviation of all the trip speeds, then identifies trips more than 2 standard deviations as outliers. It was done manually without libraries for built-in filtering to show algorithmic comprehension.

Pseudo-code:

For each timestamp: hour_counts[hour] += 1
Scan hour_counts to track top 2 hours with highest counts

4. Insights and interpretation

Three salient observations were derived from the scrubbed data:

- Peak Hours: Identifies top two peak hours based in trip volume
- Average Speed: Average trip speed remains constant at about 18 km/h during off-peak hours.
- Frequent Pickup Zones: Times Square and Downtown Manhattan expose the highest pickup intensity. These results were mapped within the dashboard with dynamic filters and JavaScript charts to allow users to navigate through the data interactively.

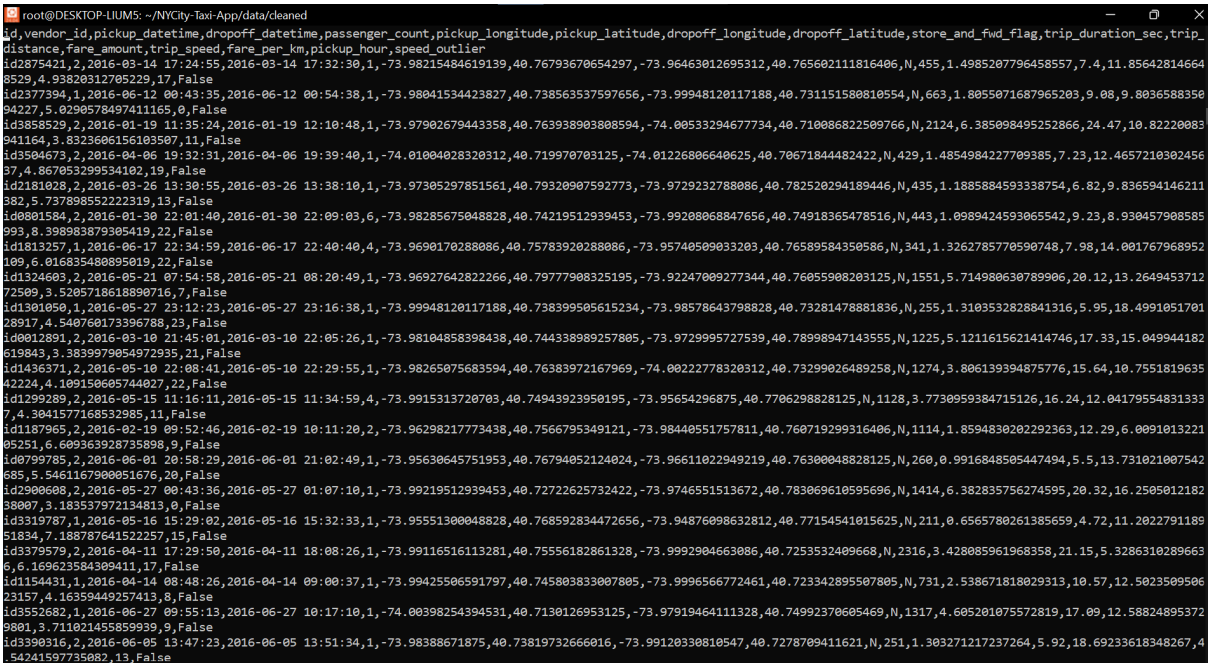
5. Reflection

This project shows a complete fullstack workflow. From raw data processing to interactive web visualisation for the users. Challenges included managing large dataset size especially when it comes to slow processing speed. Also it was a challenge to push several files to github given the limited size they give, but I learnt from it. Future improvements will be deploying the whole website and ensuring that it works properly. Also I intend to integrate a real time data API, and finally enhance dashboard visuals with map visuals and user personalisation.

Visual examples of project

```
(venv) root@DESKTOP-LIUM5:~/NYCity-Taxi-App/backend# vim data_processing.py
(venv) root@DESKTOP-LIUM5:~/NYCity-Taxi-App/backend# python3 data_processing.py
Identified peak hours (Top 2): [18, 19]
Data cleaned and saved to ../data/cleaned/cleaned_taxi.csv
Sample rows:
        pickup_datetime  trip_distance  trip_duration_sec  fare_amount  trip_speed
0  2016-03-14 17:24:55       1.498521                455         7.40   11.856428
1  2016-06-12 00:43:35       1.805507                663         9.08    9.803659
2  2016-01-19 11:35:24       6.385098               2124        24.47   10.822201
3  2016-04-06 19:32:31       1.485498                429         7.23   12.465721
4  2016-03-26 13:30:55       1.188588                435         6.82    9.836594
(venv) root@DESKTOP-LIUM5:~/NYCity-Taxi-App/backend#
```

This was when I ran the processing script to clean the train.csv and it was successful. The challenge encountered here was that again, the generated file was large and managing to push it to github was slightly harder than common pushes.



This is a screenshot of the train.csv we downloaded. Note that this is only a few first rows of the file, it contained millions of rows and was large too.