

# 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.

```
root@DESKTOP-IUUM5: ~/NYCity-Taxi-App/data/cleaned
id,vendor_id,pickup_datetime,dropoff_datetime,passenger_count,pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude,store_and_fwd_flag,trip_duration_sec,trip_distance,fare_amount,trip_speed,fare_per_km,pickup_hour,speed_outlier
id2875421,2,2016-03-14 17:24:55,2016-03-14 17:32:50,1,-73.98215484619139,40.76793670654297,-73.96463012695312,40.765602111816406,N,455,1.4985207796458557,7.4,11.85642814664
8529,4,93820312705229,17,False
id2377394,1,2016-06-12 00:43:35,2016-06-12 00:54:38,1,-73.98041534423827,40.738563537597656,-73.99948120117188,40.731151580810554,N,663,1.8055071687965203,9.08,9.8036588358
94227,5.0290578497411165,0,False
id3858529,2,2016-01-19 11:35:24,2016-01-19 12:10:48,1,-73.97902679443358,40.763938903808594,-74.00533294677734,40.710086822509766,N,2124,6.385098495252866,24.47,10.82220083
941164,3.8323606156103507,11,False
id3504673,2,2016-04-06 19:32:31,2016-04-06 19:39:40,1,-74.01004028320312,40.719970703125,-74.01226806640625,40.70671844482422,N,429,1.4854984227709385,7.23,12.4657210302456
37,4.867053299534102,19,False
id2181028,2,2016-03-26 13:30:55,2016-03-26 13:38:10,1,-73.97305297851561,40.79320907592773,-73.9729232788086,40.782520294189446,N,435,1.1885884593338754,6.82,9.836594146211
382,5.737898552222319,13,False
id0801584,2,2016-01-30 22:01:40,2016-01-30 22:09:03,6,-73.98285675048828,40.74219512939453,-73.99208068847656,40.74918365478516,N,443,1.0989424593065542,9.23,8.930457908585
993,8.398983879305419,22,False
id1813257,1,2016-06-17 22:34:59,2016-06-17 22:40:40,4,-73.9690170288086,40.75783920288086,-73.95740509033203,40.76589584350586,N,341,1.3262785770590748,7.98,14.001767968952
109,6.016835480895019,22,False
id1324603,2,2016-05-21 07:54:58,2016-05-21 08:20:49,1,-73.96927642822266,40.79777908325195,-73.92247009277344,40.76055908203125,N,1551,5.714980630789906,20.12,13.2649453712
72509,3.5205718618890716,7,False
id1301050,1,2016-05-27 23:12:23,2016-05-27 23:16:38,1,-73.99948120117188,40.738399505615234,-73.98578643798828,40.73281478881836,N,255,1.3103532828841316,5.95,18.4991051701
20917,4.5407650173096780,23,False
id0012891,2,2016-03-10 21:45:01,2016-03-10 22:05:26,1,-73.98104858398438,40.744338989257805,-73.972999572539,40.78998947143555,N,1225,5.1211615621414746,17.33,15.049944182
619843,3.3839979054972935,21,False
id1436371,2,2016-05-10 22:08:41,2016-05-10 22:29:55,1,-73.98265075683594,40.76383972167969,-74.00222778320312,40.73299026489258,N,1274,3.806139394875776,15.64,10.7551819635
42224,4.109150605744027,22,False
id1299280,2,2016-05-15 11:16:11,2016-05-15 11:34:59,4,-73.9915313720703,40.74943923950195,-73.95654296875,40.7706298828125,N,1128,3.7730959384715126,16.24,12.04179554831333
7,4.3041577168532985,11,False
id1187965,2,2016-02-19 09:52:46,2016-02-19 10:11:20,2,-73.96298217773438,40.7566795349121,-73.98440551757811,40.760719299316406,N,1114,1.8594830202292363,12.29,6.0091013221
05251,6.609363928735898,9,False
id0799785,2,2016-06-01 20:58:29,2016-06-01 21:02:49,1,-73.95630645751953,40.76794052124024,-73.96611022949219,40.76300048828125,N,260,0.9916848505447494,5.5,13.731021007542
685,5.5461167900051676,20,False
id2900608,2,2016-05-27 00:43:36,2016-05-27 01:07:10,1,-73.99219512939453,40.72722625732422,-73.9746551513672,40.783069610595696,N,1414,6.382835756274595,20.32,16.2505012182
38007,3.183537972134813,0,False
id3319787,1,2016-05-16 15:29:02,2016-05-16 15:32:33,1,-73.95551300048828,40.768592834472656,-73.94876098632812,40.77154541015625,N,211,0.6565780261385659,4.72,11.2022791189
51834,7.188787641522257,15,False
id3379579,2,2016-04-11 17:29:50,2016-04-11 18:08:26,1,-73.99116516113281,40.75556182861328,-73.9992904663086,40.7253532409668,N,2316,3.428085961968358,21.15,5.3286310289663
6,6.169623584309411,17,False
id1154431,1,2016-04-14 08:48:26,2016-04-14 09:00:37,1,-73.99425506591797,40.745803833007805,-73.9996566772461,40.723342895507805,N,731,2.538671818029313,10.57,12.5023509506
23157,4.16359449257413,8,False
id3552682,1,2016-06-27 09:55:13,2016-06-27 10:17:10,1,-74.00390254394531,40.7130126953125,-73.97919464111328,40.74992370605469,N,1317,4.605201075572819,17.09,12.58824895372
9001,3.7110211455859939,9,False
id3390316,2,2016-06-05 13:47:23,2016-06-05 13:51:34,1,-73.98388671875,40.73819732660016,-73.99120330810547,40.7278709411621,N,251,1.303271217237264,5.92,18.69233618348267,4
.54241597734682,13,False
```

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.