Analysis Of Yellow Taxis In New York Big Data III Project

Team Roles:

Alejandro Vilela: Use Power BI for clearer data visualization.
Jaime Municio: Made the documentation and gave insightful visión

Aristeo López: Made data graphs with Python and cleansed the data.
Álvaro Santamarina: Made the presentation and helped with data depuration.

What variables did we encounter? :

VendorID: A code indicating the TPEP provider that provided the record

Tpep_pickup_datetime: The date and time when the meter was engaged.

Passenger_count: The number of passengers in the vehicle. This is a driver-entered value.

Trip_distance: The elapsed trip distance in miles reported by the taximeter.

Pickup_longitude: Longitude where the meter was engaged

Pickup_latitude: Latitude where the meter was engaged.

Store _and_fwd_flag This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “stored

Dropoff_longitude: Longitude where the meter was disengaged.

Dropoff_latitude: Latitude where the meter was disengaged.

Payment_type: A numeric code signifying how the passenger paid for the trip

Fare_amount: The time-and-distance fare calculated by the meter.

Extra: Miscellaneous extras and surcharges. Currently, this only includes. the $0.50 and $1 rush hour and overnight charges

Mta_tax: 0.50 MTA tax that is automatically triggered based on the metered rate in use.

Tip_amount: This field is automatically populated for credit card tips.Cash tips are not included.

Tolls: Total amount of all tolls paid in trip

Total_amount: The total amount charged to passengers. Does not include cash tips.

Phase I : Data Cleaning and transformation

The objective of this phase is to obtain a clean, and ready for analysis dataset, for this we started from the raw New York City yellow taxi trip dataset (yellow_tripdata).

One of the main technical challenges in this project was dealing with the size of the raw taxi data files, which contain **tens of millions of rows** per month. Given the limitations of the available hardware, it was essential to **avoid loading entire datasets into memory** at once.

The volume of the dataset was too much for our computing resources, so we decided to divide it in parts (chunks).

We applied some filters to ensure that the dataset was useful for us

For each chunk, the following filters were applied:

- Removal of null values.
- `trip_distance` > 0
- `fare_amount` ≥ 2.5 (minimum legal fare in NYC)
- Geographic coordinates restricted to Manhattan:
- Latitude between 40.70 and 40.88
- Longitude between -74.03 and -73.90

- Trip duration limited to a maximum of 12 hours (43,200 seconds)
- Valid categories only:
- `payment_type`: 1 (card) or 2 (cash)
- `passenger_count`: between 1 and 4 (We applied this filter because we wanted to choose standarised taxis for our analysis.
- `extra`: between 0 and 1.

We also created some new colums for extracting new insights such as the speed in km/h ; and converted the miles to km

- Distance conversion from miles to kilometers:
  `distance_km = trip_distance * 1.60934`
- Trip duration in hours
- Speed in km/h:
  `speed_kmh = distance_km / duration_hours`
- Also rows with infinite or non-numeric values were removed

We calculated the outliers based on the Q1, Q3 and IQR, and for the columns of `fare_amount`, `trip_distance`, `tip_amount`, and speed_kmh, and we only retained the values that fell within the range `[Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]`

We also converted the pickup and dropoff datetime columns to proper datetime format in order to calculate trip duration accurate

Each cleaned chunk was appended to its corresponding monthly `.csv` file.
We used `mode='a'` to append data and `header=first` chunk to write column names only once which allowed us to save efficiently without overloading memory

Once a chunk was cleaned, it was **immediately written to disk**

We were able to clean full-month datasets without ever holding more than a fraction of the data in memory.

2. Clustering Analysis of Pickups

This phase focused on identifying spatial patterns in taxi pickups using clustering techniques, specifically K-Means. Unlike the heatmap phase, which required only coordinates and timestamps, clustering was applied on data that had passed through a **more extensive cleaning process**, including calculated fields such as trip duration, distance in kilometers, and average speed.

We implemented the clustering pipeline using the fully cleaned monthly datasets generated in the previous phase. To make the process scalable, we read each cleaned CSV in chunks and applied a **1% random sampling per chunk** directly during iteration. This approach ensured geographic diversity while keeping the dataset size manageable for clustering.

Once all sampled chunks were concatenated, we selected only the pickup latitude and longitude and applied **KMeans (k = 15)** using `sklearn`. Each record was assigned a cluster label based on its location.

We visualized the clustered pickup points using `plotly.express.scatter_mapbox`, assigning a different color to each cluster. We configured the map with consistent zoom, opacity, and NYC-centered coordinates, and exported it as an interactive HTML file.

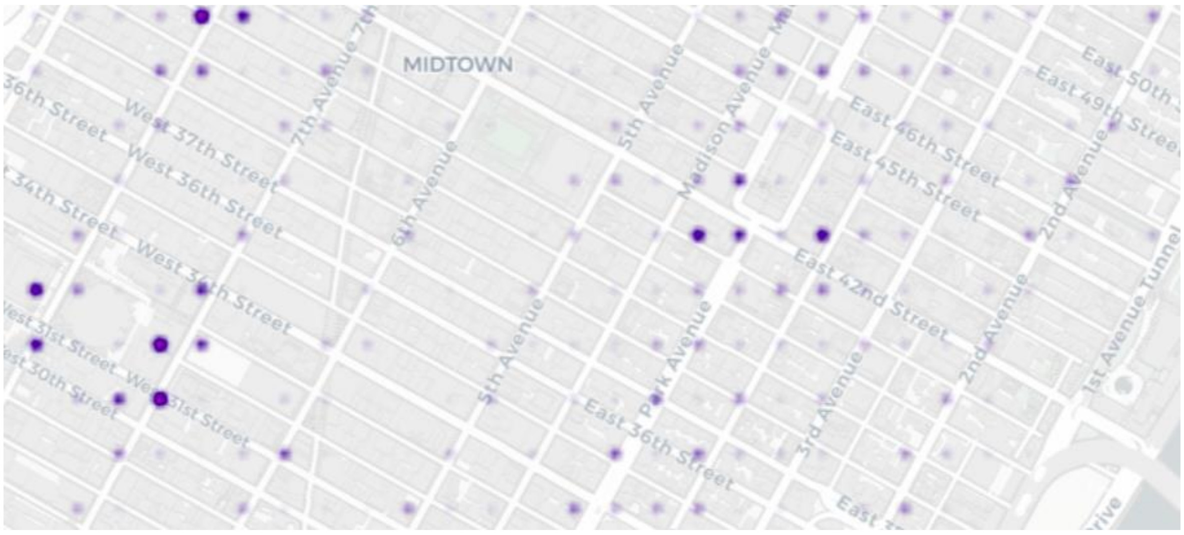
To characterize each cluster, we calculated **mean values** of key variables (`trip_distance`, `duration_min`, `fare_amount`, `tip_amount`, `total_amount`) by grouping the sample by cluster label. These statistics were used to generate a second map showing cluster centroids, with hover labels displaying average metrics.

Finally, we **merged the cluster summaries back into the sampled dataset**, allowing us to enrich the original pickup map with hover data that included the mean values of each cluster. We applied a further sample of 50,000 rows before plotting the final map to ensure smooth rendering.

All visual outputs were saved to the `Graphics` directory. allowing changes to sampling rate, number of clusters, or aggregation logic without affecting the rest of the workflow.

This phase differed from the heatmap workflow not only in method but also in data preparation: clustering required full-feature validation and outlier removal, whereas the heatmap pipeline worked with a lighter, coordinate-only structure.

Heatmaps



This phase focused on generating spatial-temporal visualizations to explore how taxi pickups in New York City vary by location and time. The process was based on cleaned datasets and involved additional transformations to aggregate data on a spatial grid and by time components.

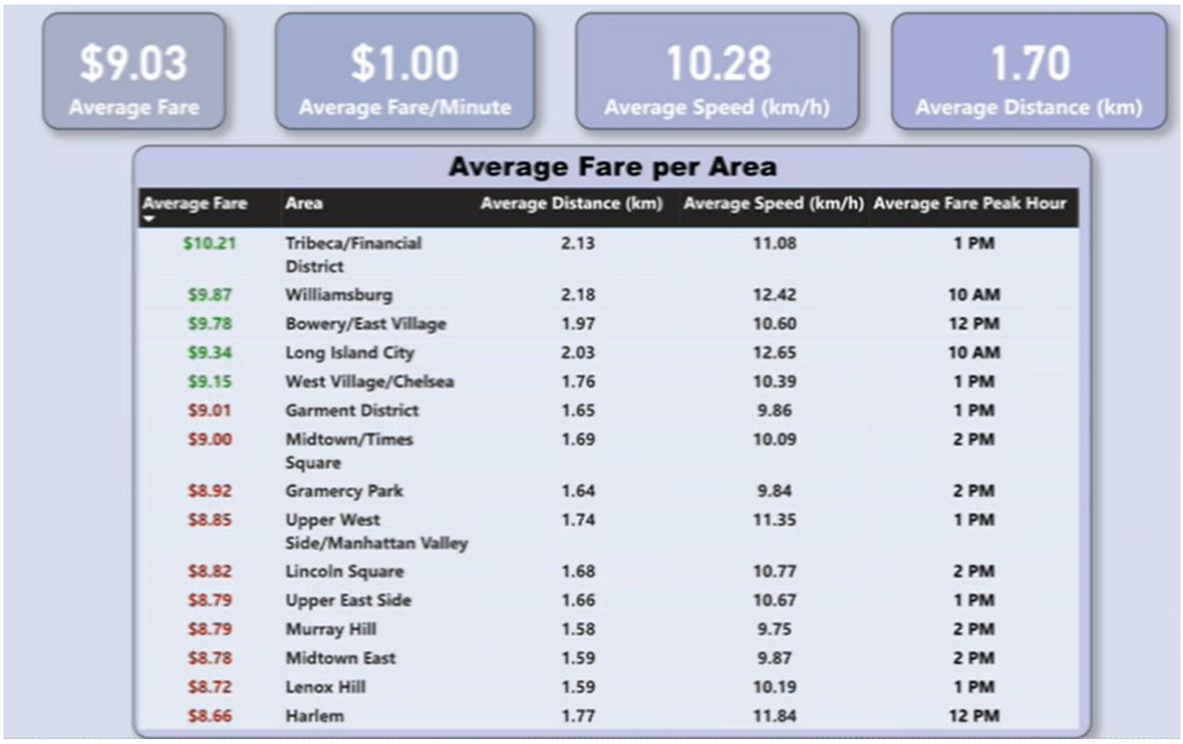
We first read each dataset in chunks and selected only the necessary columns: pickup datetime and pickup coordinates. From the datetime, we extracted the **hour of the day**, and then rounded the coordinates to a fixed grid size (`0.001`) to group trips spatially. We aggregated the number of pickups per grid cell and hour, across all files, and combined the results into a single DataFrame.

This data was used to create an animated **density map**, where each frame represented a specific hour. We used `plotly.express.density_mapbox` to plot pickup intensity, setting consistent parameters such as zoom, center coordinates, and a fixed color scale (`zmin`, `zmax`) to maintain comparability between frames. The output was exported as an interactive HTML map.

We extended this by adding the **weekday** to each record, grouping by both hour and day. This allowed us to explore differences across weekdays and weekends. While our initial attempt to include `facet_col="weekday"` raised a `TypeError` due to Plotly's limitations with `density_mapbox`, we considered alternatives like filtering data per weekday and rendering separate maps or using dropdown controls.

All outputs were saved in the `Graphics` directory for inspection. This phase involved manipulating temporal and spatial features, handling grouped aggregations, and managing visual constraints in Plotly for animated map-based visualizations.

Phase 4: Data Visualization in Power BI



$9.03 Average Fare

$1.00 Average Fare/Minute

10.28 Average Speed (km/h)

1.70 Average Distance (km)

Average Fare per Area

Average Fare	Area	Average Distance (km)	Average Speed (km/h)	Average Fare Peak Hour
$10.21	Tribeca/Financial District	2.13	11.08	1 PM
$9.87	Williamsburg	2.18	12.42	10 AM
$9.78	Bowery/East Village	1.97	10.60	12 PM
$9.34	Long Island City	2.03	12.65	10 AM
$9.15	West Village/Chelsea	1.76	10.39	1 PM
$9.01	Garment District	1.65	9.86	1 PM
$9.00	Midtown/Times Square	1.69	10.09	2 PM
$8.92	Gramercy Park	1.64	9.84	2 PM
$8.85	Upper West Side/Manhattan Valley	1.74	11.35	1 PM
$8.82	Lincoln Square	1.68	10.77	2 PM
$8.79	Upper East Side	1.66	10.67	1 PM
$8.79	Murray Hill	1.58	9.75	2 PM
$8.78	Midtown East	1.59	9.87	2 PM
$8.72	Lenox Hill	1.59	10.19	1 PM
$8.66	Harlem	1.77	11.84	12 PM

In addition to the Python-based visualizations, we used Power BI to create a complementary layer of analysis focused on dashboards and static visual exploration.

This phase allowed us to interact with the data in a different environment, useful for quick metric comparisons and visualizations.

We imported the cleaned CSV files into Power BI, where we built metric cards, bar and line charts, and map visualizations to analyze ride behavior across different zones and time ranges. A significant challenge in this phase was the transformation of variable types, especially datetime and geographic fields. Some columns required reformatting (e.g. converting strings to datetime or splitting lat/lon into separate fields) to ensure proper recognition by Power BI. We also had to manually configure data models and relationships to preserve consistency in aggregations and filters.

Power BI provided a fast, interactive way to summarize some of the patterns identified during the Python-based analysis, while some others were identified during the Power BI analysis.

Conclusion

During the exploratory and visualization phases, we identified several patterns in taxi activity that went beyond the expected concentrations around well-known tourist or commercial areas.

We believe this dataset has significant potential—not only for drivers or ride-hailing companies like Uber or Cabify, looking to optimize where and when to find passengers—but also for demographic analysis. It offers valuable insights into how people move throughout the city, which could be useful for understanding urban behavior and planning targeted advertising strategies for various types of businesses.

Further application of machine learning could reveal hidden patterns in urban mobility, enabling more accurate demand forecasting and dynamic, real-time resource allocation.