# NYC Taxi Data Analysis

Sanya Garg
*Computer Science and Engineering*
*Netaji Subhas University of Technology*
New Delhi, India
sanya.garg.ug22@nsut.ac.in

***Abstract*—This project presents a complete end-to-end data analytics pipeline leveraging modern cloud infrastructure and open-source tools to analyze New York City taxi data. The raw dataset is first uploaded to Google Cloud Storage and then ingested using Mage AI inside a Python-configured virtual machine. The pipeline applies dimensional modeling principles to transform the flat dataset into normalized dimension and fact tables. These structured tables are then exported to BigQuery using Mage's built-in data exporter module. Analytical SQL queries are executed on BigQuery to derive meaningful insights, which are finally visualized using an interactive dashboard built in Looker Studio. This streamlined pipeline demonstrates how cloud-native tools and modular ETL frameworks can be effectively used for scalable data engineering and analytics workflows**

***Index Terms*—NYC Taxi Data Analysis, GCP Storage, Python, Compute Instance, Mage Data Pipeline Tool, BigQuery, Looker Studio, PowerBI,**

## I. Introduction

In recent years, the volume and complexity of data generated by urban transport systems have grown exponentially. Cities like New York, where millions of taxi rides are recorded every month, generate rich datasets that can be harnessed to uncover patterns in human mobility, identify inefficiencies, and support data-driven policymaking. Among these, the New York City Taxi and Limousine Commission (TLC) trip data stands out as a publicly available, high-granularity dataset that includes timestamps, geolocations, fare details, and trip durations. However, working with such large-scale data poses significant challenges, including the need for efficient storage, fast querying, scalable processing, and intuitive visualization.

To address these challenges, this project adopts a modern data engineering pipeline built on cloud-native and open-source tools. The pipeline begins with data ingestion from Google Cloud Storage, where raw NYC taxi trip data is stored. A transformation layer powered by the Mage AI orchestration tool extracts, cleans, and models the data using dimensional schemas tailored for analytical use. The transformed data is then loaded into Google BigQuery, a highly scalable data warehouse designed for real-time analytics. This setup enables fast querying and complex aggregations over millions of records without compromising on performance or scalability.

The final stage of the pipeline focuses on data consumption through visualization. By connecting BigQuery to Looker Studio (formerly Google Data Studio), the project delivers interactive dashboards that enable users to explore key insights such as peak travel hours, average trip distances, fare distributions, and geographic pickup/dropoff hotspots. The integration of these technologies reflects best practices in modern data architecture and demonstrates how large datasets can be transformed into meaningful insights with minimal infrastructure overhead. This project not only provides a practical solution for NYC taxi data analysis but also serves as a blueprint for similar data analytics workflows in other domains.

This project was carried out as part of an academic research initiative.

## II. Dataset Description

This project utilizes the New York City Taxi and Limousine Commission (TLC) Trip Record Data, which provides detailed information on taxi trips in NYC. Specifically, we focus on yellow taxi data, which includes millions of trip records per month and offers a rich set of attributes such as pickup/dropoff timestamps, geolocations, trip distance, fare details, passenger count, and payment type.

The dataset is updated monthly and publicly available in CSV format, with historical records dating back to 2009. Its high volume and diverse features make it ideal for big data analysis and time-series forecasting. For this project, selected monthly records were stored in Google Cloud Storage and processed using a cloud-based ETL pipeline for further analysis and visualization.

## III. Methodology

The end-to-end data processing pipeline was designed to efficiently transform raw NYC taxi trip data into a structured analytical dataset hosted in Google BigQuery, and ultimately visualized using Looker Studio. The following subsections outline each step in the pipeline, from preprocessing to dashboard development.

### A. Initial Dimensional Modeling using R

The transformation logic for converting raw flat-file trip data into a dimensional model was first prototyped in R. This involved applying principles of dimensional modeling to identify natural dimension and fact splits. The raw dataset was decomposed into several normalized tables:

- **Datetime Dimension:** Captures pickup and dropoff timestamps and derived fields (hour, day, month, weekday).

TABLE I
DATASET DICTIONARY FOR NYC YELLOW TAXI TRIP DATA

| Field Name | Description |
|---|---|
| VendorID | Code for the TPEP provider: 1 = Creative Mobile Technologies, 2 = Curb Mobility, 6 = Myle, 7 = Helix. |
| tpep_pickup_datetime | Timestamp when the trip started (meter engaged). |
| tpep_dropoff_datetime | Timestamp when the trip ended (meter disengaged). |
| passenger_count | Number of passengers in the taxi. |
| trip_distance | Distance traveled in miles, recorded by the meter. |
| RatecodeID | Final rate code used: 1 = Standard, 2 = JFK, 3 = Newark, 4 = Nassau/Westchester, 5 = Negotiated, 6 = Group ride, 99 = Unknown. |
| store_and_fwd_flag | Indicates if trip data was stored before forwarding: Y = Yes, N = No. |
| PULocationID | Pickup location ID (TLC taxi zone). |
| DOLocationID | Dropoff location ID (TLC taxi zone). |
| payment_type | Payment method: 0 = Flex Fare, 1 = Credit card, 2 = Cash, 3 = No charge, 4 = Dispute, 5 = Unknown, 6 = Voided. |
| fare_amount | Fare calculated by meter based on time and distance. |
| extra | Extra charges and surcharges (e.g., rush hour fee). |
| mta_tax | MTA tax automatically added based on rate. |
| tip_amount | Tip amount (credit card tips only; cash tips not included). |
| tolls_amount | Total tolls paid during the trip. |
| improvement_surcharge | Fixed surcharge added at trip start (since 2015). |
| total_amount | Total fare amount charged to the passenger (excluding cash tips). |
| congestion_surcharge | NYS congestion surcharge collected during trip. |
| airport_fee | Fixed fee for pickups at JFK and LaGuardia. |
| cbd_congestion_fee | Per-trip congestion fee in Manhattan (since Jan 5, 2025). |

- **Passenger Count, Trip Distance, Rate Code, Payment Type:** Each treated as individual dimensions with unique keys.
- **Pickup and Dropoff Location Dimensions:** Derived from TLC zone identifiers.
- **Fact Table:** Contains ride-specific metrics such as fare amount, tip amount, total amount, and foreign keys referencing the dimensions.

This initial modeling in R helped validate the data structure and guide the pipeline development process.

### *B. Cloud Infrastructure Setup and Mage Installation*

The next phase involved setting up the infrastructure on Google Cloud Platform (GCP):

- A Virtual Machine (VM) instance was created using Google Compute Engine (GCE), with SSH access enabled.
- The required environment was configured by installing Python and dependencies.
- Mage AI, an open-source data pipeline tool, was installed using command-line instructions to orchestrate the pipeline within the VM.
- The raw dataset, exported as a CSV file, was uploaded to a Google Cloud Storage (GCS) bucket to enable public access.

### *C. Data Ingestion Using Mage's Data Loader*

A Mage pipeline was created to begin with a data ingestion block:

- The `data_loader` block fetched the dataset directly from the public GCS URL using Python's `requests` library.
- The data was parsed and loaded into a Pandas DataFrame, where initial cleaning (e.g., handling missing values, type casting) was performed.

### *D. Transformation and Dimensional Modeling in Mage*

The core transformation logic was implemented in Mage's `transform` block:

- The previously validated dimensional model from R was replicated in Python within Mage.
- The flat DataFrame was decomposed into six normalized dimension tables and one central fact table.
- Foreign keys were generated to maintain referential integrity.
- Duplicate and inconsistent records were filtered out, and data types were normalized.

### *E. Exporting to BigQuery*

Once the dimensional structure was created, a `data_exporter` block was configured to load the transformed data into BigQuery:

- Each dimension and the fact table were written to a custom BigQuery dataset named `taxidataset`.
- The exporter was configured using Mage's YAML-based export settings, which handled schema definition and table creation automatically.

### *F. Analytics Table Creation using SQL*

To facilitate high-level analysis, a custom analytics table named `tbl_analytics` was created within BigQuery:

- SQL scripts were written to join dimension and fact tables and derive meaningful aggregations (e.g., total revenue, trip count by location, average tip by hour).
- These queries also performed data enrichment by adding calculated fields such as average trip speed and fare efficiency.

### *G. Dashboard Development in Looker Studio*

Finally, the BigQuery dataset was connected to Looker Studio for visualization:

- A fully interactive dashboard was built, enabling dynamic exploration of key metrics.
- Visualizations included time-series trends, geographic trip density, tip patterns, payment method usage, and fare breakdowns.
- Filters were added for time, location, rate code, and payment type, allowing granular analysis by business users.

### *H. Power BI Dashboard on Complete Dataset*

In addition to Looker Studio, a comprehensive dashboard was developed using Power BI to provide alternative analytical perspectives. Unlike the Looker Studio dashboard, which was focused on the transformed dimensional data in BigQuery, the Power BI dashboard worked directly with the full raw dataset.

- The entire CSV dataset was loaded into Power BI, allowing for quick exploration of the raw, unnormalized data.
- Data cleaning and type conversion steps were handled using Power BI's Power Query editor, ensuring consistency in formats such as datetime fields and numeric columns.

- Custom DAX (Data Analysis Expressions) measures were created to derive insights such as average fare per mile, total trips per day, revenue per vendor, and tip percentages.
- Various visuals—such as bar charts, heatmaps, slicers, and time series graphs—were used to highlight patterns in trip volume, revenue generation, passenger count, and pickup/dropoff behavior.
- Interactive filters were added for dimensions like RateCodeID, payment type, and passenger count to enable drill-down analysis.
- The Power BI dashboard served as a validation layer to cross-check findings from the BigQuery-based pipeline and offered additional capabilities like offline usage and custom publishing.

This dual-dashboard approach allowed both real-time cloud-based analysis (via Looker Studio) and more customizable, interactive desktop reporting (via Power BI), giving end-users flexibility in how they interpret the NYC taxi data.

## IV. Results

This project successfully demonstrates the complete development of a modern data engineering pipeline, starting from raw data ingestion to advanced business intelligence reporting. Using Taxi trip data as the foundation, the process began by uploading the dataset to Google Cloud Storage, followed by setting up a custom virtual machine (VM) to run Mage AI locally. Within Mage, the dataset was transformed into a clean and structured dimensional model comprising several well-defined dimension tables and a central fact table. These tables were then exported to Google BigQuery, where a final analytical table (`tbl_analytics`) was created by joining all dimension tables to the fact table. This unified table enabled rich, high-performance analysis and was used as the data source for a fully interactive dashboard built in Looker Studio and PowerBI. The dashboard allows users to explore trends across trip time, distance, location, passenger behavior, and payment types—delivering powerful insights with just a few clicks. This project showcases a seamless integration of data engineering, cloud platforms, and BI tools, using only free-tier resources and open-source technologies, with a focus on scalability, clarity, and real-world business value.

### A. Dimension Tables

These are specialized reference tables, known as dimension tables, that are created by extracting and organizing unique values from specific columns in the original dataset. Their primary purpose is to normalize the data—by storing descriptive information in separate tables—and reduce redundancy in the central fact table. Instead of repeating the same values (like location IDs, payment types, or timestamps) across every row in the fact table, we use lightweight keys (IDs) that link to these dimension tables. This structure not only improves storage efficiency and consistency but also enhances query performance by simplifying joins and enabling more flexible analytics.

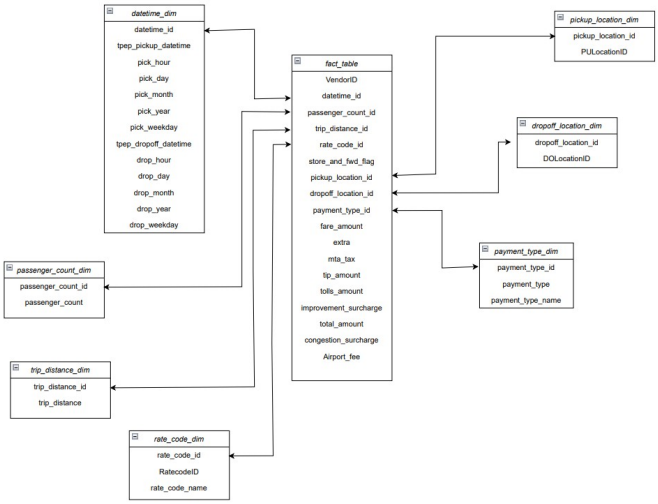



TABLE II
TABLE NAME AND DESCRIPTION

| Table Name | Description |
|---|---|
| `datetime_dim` | Contains unique combinations of pickup and dropoff datetimes, with a generated `datetime_id`. |
| `passenger_count_dim` | Unique passenger counts, each assigned a `passenger_count_id`. |
| `trip_distance_dim` | Unique trip distances with corresponding `trip_distance_id`. |
| `rate_code_dim` | Unique `RatecodeIDs` (1–6), each mapped to a human-readable description like "Standard rate", "JFK", etc. |
| `pickup_location_dim` | Unique pickup location IDs (`PULocationID`), each assigned a `pickup_location_id`. |
| `dropoff_location_dim` | Unique dropoff location IDs (`DOLocationID`), each assigned a `dropoff_location_id`. |
| `payment_type_dim` | Unique payment types mapped to values like "Cash", "Credit card", "Dispute", etc., with `payment_type_id`. |

TABLE III
DATETIME_DIM

| tpep_pickup_datetime | tpep_dropoff_datetime | datetime_id |
|---|---|---|
| 2025-01-01T00:00:37 | 2025-01-01T00:04:46 | 27 |
| 2025-01-01T00:00:03 | 2025-01-01T00:07:46 | 138 |
| 2025-01-01T00:02:57 | 2025-01-01T00:08:05 | 433 |
| 2025-01-01T00:01:44 | 2025-01-01T00:08:38 | 169 |

TABLE IV
PASSENGER_COUNT_DIM

| passenger_count | passenger_count_id |
|---|---|
| 1 | 0 |
| 3 | 1 |
| 2 | 2 |
| 0 | 3 |

TABLE V
TRIP_DISTANCE_DIM

| trip_distance | trip_distance_id |
|---|---|
| 1.6 | 0 |
| 0.5 | 1 |
| 0.6 | 2 |
| 0.52 | 3 |

TABLE VI
PICKUP_LOCATION_DIM

| PULocationID | pickup_location_id |
|---|---|
| 229 | 0 |
| 236 | 1 |
| 141 | 2 |
| 244 | 3 |

TABLE VII
DROPOFF_LOCATION_DIM

| DOLocationID | pickup_location_id |
|---|---|
| 237 | 0 |
| 141 | 1 |
| 244 | 2 |
| 116 | 3 |

TABLE VIII
PAYMENT_TYPE_DIM

| payment_type | payment_type_id | payment_type_name |
|---|---|---|
| 1 | 0 | Credit card |
| 2 | 1 | Cash |
| 4 | 2 | Dispute |
| 3 | 3 | No charge |

### B. Fact Table

The fact_table serves as the central table in the star schema or snowflake schema of the data model. It contains references (foreign keys) to all the associated dimension tables, which provide detailed attributes for the different aspects of each record. This table is crucial because it aggregates key performance metrics and transactional data. In this case, the fact table primarily focuses on the fare_amount as the main metric. However, it also includes other important columns that provide context to the data, such as passenger count, trip distance, payment type, and rate code.

The fact table acts as the central repository for transactional data that is analyzed by grouping through various dimensions. Each row represents a unique trip with associated metrics and foreign keys linking to the relevant dimension tables, such as datetime_dim, passenger_count_dim, trip_distance_dim, rate_code_dim, pickup_location_dim, dropoff_location_dim, and payment_type_dim. This structure allows for detailed analysis of trip-related metrics, such as fare amounts, by specific time periods, passenger counts, locations, and payment types. The fact table's design ensures efficient querying and reporting, enabling analysts to generate insights from the data, such as fare amounts by different time periods, or the average trip distance for a particular location or payment type.

The fact table essentially integrates all of the data, and by joining with the corresponding dimension tables, it supports comprehensive analytical queries. The design and granularity of the fact table are crucial for ensuring that the data is stored in a way that facilitates fast and efficient analysis.

TABLE IX
FACT_TABLE

| datetime_id | trip_distance_id | pickup_location_id | dropoff_location_id | payment_type_id | fare_amount |
|---|---|---|---|---|---|
| 39 | 0 | 0 | 23 | 4 | 17.0 |
| 51 | 0 | 0 | 26 | 26 | 19.8 |
| 104 | 0 | 0 | 41 | 34 | 19.1 |
| 105 | 0 | 0 | 42 | 44 | 9.3 |
| 159 | 0 | 0 | 46 | 44 | 21.9 |

### C. BigQuery

To finalize the data modeling pipeline, a comprehensive SQL query was written in BigQuery to construct the tbl_analytics table. This table is a denormalized result of joining the central fact_table with all related dimension tables—namely datetime_dim, passenger_count_dim, trip_distance_dim, rate_code_dim, pickup_location_dim, dropoff_location_dim, and payment_type_dim. The query also performs feature engineering by extracting useful time-based attributes such as hour, day, month, year, and weekday from both pickup and dropoff timestamps. These enrichments enable powerful temporal analysis and improve dashboard interactivity. The SQL script ensures that all necessary fields are unified into a single, flat table, making it highly query-efficient and ideal for direct use in BI tools like Looker Studio. By executing this query in BigQuery, the analytics layer of the project becomes robust, scalable, and easily accessible for reporting and visualization.

### D. Analytics Table

The tbl_analytics table is a denormalized, query-optimized table created within BigQuery using SQL. It is derived by joining the central fact_table with all associated dimension tables, which are extracted from both pickup and dropoff datetime fields. These temporal features enable granular time-series analysis and pattern discovery across multiple levels of aggregation.

The main objective of the tbl_analytics table is to serve as the final data layer for reporting, analysis, and visualization. By consolidating all required fields into one flat structure, it eliminates the need for runtime joins or complex transformations. This approach significantly enhances query performance and ensures that business users, analysts, and visualization tools can access clean, consistent, and ready-to-use data with minimal overhead.

Furthermore, the tbl_analytics table acts as the primary source for the interactive dashboard built using Looker Studio, facilitating the presentation of meaningful insights in an accessible and intuitive format.

| Row | datetime_id | tpep_pickup_datetime | pick_hour | pick_day | pick_month | pick_year | pick_weekday | tpep_dropoff_datetime | drop_hour |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 2025-01-01T00:14:47 | 0 | 1 | 1 | 2025 | 3 | 2025-01-01T00:16:15 | 0 |
| 2 | 92 | 2025-01-01T00:11:27 | 0 | 1 | 1 | 2025 | 3 | 2025-01-01T00:16:58 | 0 |
| 3 | 169 | 2025-01-01T00:01:44 | 0 | 1 | 1 | 2025 | 3 | 2025-01-01T00:08:38 | 0 |
| 4 | 396 | 2025-01-01T00:53:43 | 0 | 1 | 1 | 2025 | 3 | 2025-01-01T01:00:47 | 1 |
| 5 | 94 | 2025-01-01T00:33:13 | 0 | 1 | 1 | 2025 | 3 | 2025-01-01T00:40:08 | 0 |
| 6 | 93 | 2025-01-01T00:19:30 | 0 | 1 | 1 | 2025 | 3 | 2025-01-01T00:27:25 | 0 |

| drop_day | drop_month | drop_year | drop_weekday | passenger_count | trip_distance | pickup_location | dropoff_locatio | payment_type_name | fare_amount |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2025 | 3 | 0 | 0.4 | 170 | 170 | Credit card | 4.4 |
| 1 | 1 | 2025 | 3 | 0 | 0.7 | 144 | 211 | Credit card | 7.2 |
| 1 | 1 | 2025 | 3 | 0 | 0.9 | 113 | 148 | Credit card | 7.9 |
| 1 | 1 | 2025 | 3 | 0 | 0.8 | 163 | 161 | Cash | 8.6 |
| 1 | 1 | 2025 | 3 | 0 | 1.2 | 158 | 68 | Credit card | 8.6 |
| 1 | 1 | 2025 | 3 | 0 | 1.0 | 211 | 158 | Credit card | 9.3 |

### *E. Dashboard*

After successfully building the tbl_analytics table, it was seamlessly integrated with Looker Studio (formerly known as Google Data Studio) and PowerBI to create a dynamic, interactive dashboard designed to provide actionable business insights and facilitate effective data storytelling. This dashboard serves as a comprehensive reporting tool that visualizes key metrics, helping stakeholders make informed decisions.

*Looker Dashboard Features*

**Filters:**

- **Pickup Datetime:** Allows users to filter the data based on the pickup time, enabling them to focus on specific time periods.
- **Payment Type:** Provides the ability to filter data based on different payment methods, such as Cash, Credit Card, etc.
- **Dropoff Datetime:** Allows filtering by the dropoff time, helping users analyze trip data within specific time windows.

**Slider:**

- **Trip Distance:** Users can adjust a slider to filter trips based on distance, allowing for analysis of shorter or longer trips specifically.

**Summary:**

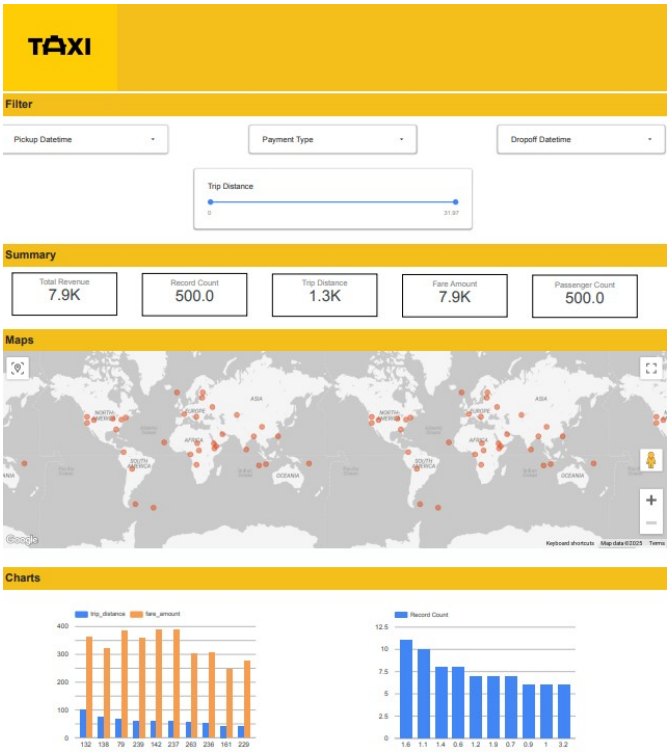
- **Total Revenue:** Displays the total fare revenue generated over the selected timeframe or by the applied filters.
- **Record Count:** Shows the number of records (i.e., trips) within the selected range or filter.
- **Trip Distance:** Displays the total distance traveled in the selected dataset or filter.
- **Fare Amount:** Shows the total amount of fare collected during the selected timeframe or by the applied filters.
- **Passenger Count:** Provides insights into the number of passengers for the selected trips or filter.

**Maps:**

- **Bubble Map:** Visualizes pickup and dropoff locations on a map, with bubble sizes representing metrics such as fare amount or trip distance. This helps in identifying high-demand locations and patterns.

**Charts:**

- **Trip Distance vs Fare Amount:** A scatter plot that visualizes the relationship between trip distance and fare amount, helping to identify pricing trends based on distance.
- **Record Count:** A simple chart displaying the number of records for each filter or selection.




*PowerBI Dashboard Features*

**Key Metrics:**

- **Total Revenue:** Displays the total revenue generated from the taxi trips by aggregating the fare amounts across all trips.
- **Total Trips:** Shows the total number of trips taken within the selected time period, providing a high-level view of trip volume.
- **Total Distance Traveled:** Represents the total distance covered by the taxis, helping assess the overall travel workload.

**Charts:**

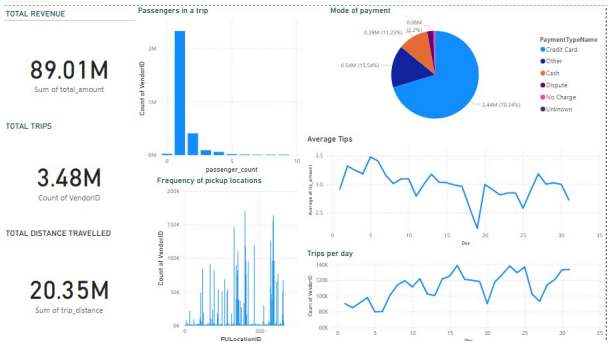
- **Passengers per Trip (Bar Chart):** A bar chart visualizing the number of passengers per trip, helping identify patterns in trip occupancy.
- **Frequency of Pickup Locations (Bar Chart):** A bar chart showing how often trips are picked up from different locations, identifying high-traffic pickup areas.

**Pie Chart:**

- **Mode of Payment:** A pie chart displaying the distribution of payment methods (e.g., cash, credit card, etc.) used by passengers, showing the preferred payment modes.

**Line Chart:**

- **Average Tips:** A line chart showing the average tip amount over time, helping track trends in tips during the selected period.
- **Trips per Day:** A line chart displaying the number of trips per day, which helps identify peak days and trends in taxi trips.




## V. Conclusion

The data engineering and analytics workflow built around the taxi dataset successfully provides valuable insights into taxi rides, customer behavior, and operational patterns. By leveraging Apache Spark, Kubernetes, and BigQuery, the project transitions from an initial cloud-based pipeline to a more robust, scalable architecture, delivering greater flexibility and efficiency.

The `tbl_analytics` table, created through SQL queries in BigQuery, serves as the core of this data pipeline. It consolidates key metrics by joining multiple dimension tables with the central fact table, ensuring that all necessary features, including time-based attributes, are easily accessible for analysis. This denormalized structure greatly enhances querying performance, making it ideal for direct reporting and visualization purposes.

The integration of this data with Looker Studio has resulted in the creation of a dynamic and interactive dashboard that provides business stakeholders with actionable insights. The dashboard's filters, sliders, and visualizations (including bubble maps, bar charts, and summary metrics) empower users to explore the data in a flexible, user-friendly interface. Key insights such as trip patterns, fare distribution, payment preferences, and location hotspots are now easily accessible, helping drive better business decision-making.

In summary, this project demonstrates how a carefully structured data pipeline combined with advanced analytics and visualization tools can effectively transform raw data into meaningful, actionable insights. The architecture is not only scalable but also adaptable, offering long-term value for reporting and decision-making across various use cases.

## VI. Future Work

While the current data pipeline and dashboard provide significant insights into taxi operations and trip patterns, several enhancements can be considered for future development:

- **Incorporation of Weather Data:** Integrating external weather data can help analyze the impact of weather conditions on trip volume, duration, and fare amounts.
- **Real-time Data Processing:** Transitioning from batch processing to real-time streaming analytics using tools like Apache Kafka or Apache Flink can enable live monitoring and faster decision-making.
- **Geospatial Analysis:** Including detailed geospatial analysis using latitude and longitude data can help visualize trip density, optimize routes, and identify high-traffic zones.
- **Predictive Modeling:** Implementing machine learning models to forecast demand, fare prices, or trip durations based on historical patterns and external factors like time, weather, and events.
- **Enhanced User Interface:** Expanding the Power BI dashboard with interactive drill-downs, comparison views, and mobile responsiveness for broader accessibility.
- **Automated Data Refresh:** Scheduling automated ETL workflows and dashboard refreshes to ensure users always access the most recent data without manual intervention.

## VII. References


1) NYC Taxi & Limousine Commission. "TLC Trip Record Data." Available at: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page
2) Mage AI. "Mage: The Modern Replacement for Airflow." Available at: https://www.mage.ai/
3) Google Cloud Storage. "Cloud Storage Documentation." Available at: https://cloud.google.com/storage/docs
4) Google BigQuery. "BigQuery Documentation." Available at: https://cloud.google.com/bigquery/docs
5) Microsoft Power BI. "Business Intelligence like never before." Available at: https://powerbi.microsoft.com/
6) Apache Spark. "Unified Analytics Engine for Big Data." Available at: https://spark.apache.org/
7) Kubernetes. "Production-Grade Container Orchestration." Available at: https://kubernetes.io/
8) R Project. "The R Project for Statistical Computing." Available at: https://www.r-project.org/