AID 2025/26

Assignment Report

Group:

- Diogo Santos (up202108747)
- Manuel Alves (up201906910)
- Rodrigo Esteves (up202403070)

Subject

This project develops a data warehouse for NYC Yellow Taxi operations, using a public dataset from the Taxi and Limousine Commission (TLC). The main goal is to transform raw trip data into a multidimensional schema that assures online analytical processing (OLAP) of demand, revenue, operational intensity, and behavioural patterns across temporal and geographic dimensions.

The data warehouse is built using July 2025 Yellow Taxi trip records in Parquet format, complemented by the TLC taxi zone lookup table in CSV. The data is cleaned, and coded attributes are standardized, such as vendor, rate code, and payment type. Additional metrics are also derived, including trip duration, average speed, and surcharge totals. Then the processed dataset is loaded into a staging table and transformed into conformed dimension and fact tables.

The scope of the project will address questions such as: how do revenue and trip volume evolve across days and hours, which zones contribute the most to overall activity, and how does payment behaviour change across locations and time.

Planning

Planning begins by identifying the facts and their grains. This model has two fact tables, one analyzing each trip, and the other one a daily aggregated report. Below, **Fig. 17** summarizes which dimensions and facts participate in each star schema.

Dimensional bus matrix

Data mart	Star / Dimension	Date	TimeOfDay	Vendor	Location	RateCode	PaymentType	TripCharacteristics	PassengerGroup
NYC Taxi Analytics	Trip	X	X	X	X	X	X	X	X
	Daily_ZoneVendor	X		X	X				

Figure 17: Dimensional Bus Matrix

Dimension dictionary

The dimension dictionary defines the meaning of each dimension, the granularity, structure, and attributes. Temporal dimensions are generated as references table (type 0), while lookup dimensions are treated as updated references (type 1) within the scope of the project (Month July 2025). Below, there are tables describing each of the dimensions, facts, and their attributes.

Dim_Date

Name	***Description***	***SCD***	***Version***	1.0	***Date***	02/02/26
Date	Calendar date of the taxi trips.	type 0	***Hierarchy***	Date < Month < Year		
Attribute	***Description***	***Level***	***Key***	***Type***	***Size***	***Precision***
DateKey	Surrogate key of the date.	Date	PK	ID	8	0
FullDate	Full calendar date (YYYY-MM-DD).	Date		Date		
Day	Day of the month (1–31).	Date		Integer	2	0
Month	Month number (1–12).	Date		Integer	2	0
MonthName	Month name (January, …).	Date		Varchar	15	
Quarter	Quarter of the year (1–4).	Date		Integer	1	0
Year	Calendar year (e.g., 2025).	Date		Integer	4	0
DayOfWeekNumber	Day of week (1–7).	Date		Integer	1	0
DayOfWeekName	Name of day (Monday, …).	Date		Varchar	10	
IsWeekend	Weekend flag (Y/N).	Date		Bool	1	

Table 1: Dimension Date

Dim_TimeOfDay

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
TimeOfDay	Time of day of trip pickup/dropoff.	type 0	*Hierarchy*	TimeOfDay < TimeBucket		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
TimeKey	Surrogate key of the time.	TimeOfDay	PK	ID	4	0
Hour	Hour of day (0–23).	TimeOfDay		Integer	2	0
Minute	Minute (0–59).	TimeOfDay		Integer	2	0
TimeLabel	Human-readable label (13:00–13:59).	TimeOfDay		Varchar	20	
TimeBucket	Time segment (Night, Morning, …).	TimeOfDay		Varchar	20	

Table 2: Dimension TimeOfDay

Dim_Vendor

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
Vendor	Taxi service provider/vendor organization.	type 1	*Hierarchy*	Vendor		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
VendorKey	Surrogate key of the vendor.	Vendor	PK	ID		
VendorCode	Original vendor ID/code.	Vendor	UK	Integer	3	0
VendorName	Name of the vendor.	Vendor		Varchar	100	

Table 3: Dimension Vendor

Dim_Location

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
Location	TLC geographical zone (borough + zone + service zone).	type 1	*Hierarchy*	ServiceZone < Borough < Zone		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
LocationKey	Surrogate key of the location.	Location	PK	ID		
LocationID	Identifier to represent zones	Location	UK	Integer	4	0
Borough	NYC borough (Manhattan, Brooklyn, …).	Location		Varchar	30	
Zone	TLC zone name.	Location		Varchar	30	
ServiceZone	TLC service zone (Yellow Zone, Boro Zone, Airport, …).	Location		Varchar	30	
IsAirport	Airport trip flag (Y/N).	Location		Bool		
IsCBD	Inside Congestion Relief Zone flag (Y/N).	Location		Bool		

Table 4: Dimension Location

Dim_RateCode

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
RateCode	Tariff / rate category applied to the taxi trip.	type 1	*Hierarchy*	Rate Code		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
RateCodeKey	Surrogate key of the rate code.	RateCode	PK	ID		
RateCodeDesc	Description (Standard, JFK, …).	RateCode		Varchar	50	

Table 5: Dimension RateCode

Dim_PaymentType

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
PaymentType	Method used to pay for the taxi trip.	type 1	*Hierarchy*	PaymentType		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
PaymentTypeKey	Surrogate key of the payment type.	PaymentType	PK	ID		
PaymentTypeDesc	Description (Credit card, Cash, …).	PaymentType		Varchar	60	

Table 6: Dimension PaymentType

Dim_PassengerGroup

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
PassengerGroup	Passenger count grouped into usage categories.	type 0	*Hierarchy*	PassengerGroup		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
PassengerGroupKey	Surrogate key of passenger group.	PassengerGroup	PK	ID		
PassengerGroupCode	Code (P1, P2, P3_4, P5plus).	PassengerGroup	UK	Varchar	10	
PassengerGroupDesc	Description (Solo, Couple, …).	PassengerGroup		Varchar	50	
MinPassengers	Minimum passengers in the group.	PassengerGroup		Integer	2	0
MaxPassengers	Maximum passengers in the group (null = +∞).	PassengerGroup		Integer	2	0

Table 7: Dimension PassengerGroup

Dim_TripCharacteristics

Name	*Description*	*SCD*	*Version*	1.0	*Date*	02/01/26
TripCharacteristics	Technical/behavioral flags characterizing each trip.	type 0	*Hierarchy*	TripCharacteristics		
Attribute	*Description*	*Level*	*Key*	*Type*	*Size*	*Precision*
TripCharacteristicsKey	Surrogate key of the trip characteristics.	TripCharacteristics	PK	ID		
StoreAndForwardBool	Store-and-forward flag from TLC (Y/N).	TripCharacteristics		Bool		
IsAirportTrip	Derived flag (Y/N) if airport fee or airport zone.	TripCharacteristics		Bool		
IsCBDTrip	Derived flag (Y/N) if CBD fee or CBD zone.	TripCharacteristics		Bool		
IsCongestionSurcharge	Flag (Y/N) if congestion_surcharge > 0.	TripCharacteristics		Bool		

Table 8: Dimension TripCharacteristics

Fact dictionary

The fact dictionary specifies the granularity of each fact table, its dimensional keys, and the measures with additivity properties.

Fact_Trip

Star	**Trip**	*Version*	1.0	*Date*	02/01/26
Granularity	One row per completed yellow taxi trip.				
Dimension	*Role in Fact Table*				
Date	PickupDate / DropoffDate (role-playing)				
TimeOfDay	PickupTime / DropoffTime (role-playing)				
Vendor	Vendor				
Location	PickupLocation / DropoffLocation (role-playing)				
RateCode	RateCode				
PaymentType	PaymentType				
TripCharacteristics	TripCharacteristics				
FactTripKey	FactTripKey				

PassengerGroup	PassengerGroup	
Measure	***Description***	***Additivity***
TripDistance	Distance of the trip in miles.	Additive
FareAmount	Metered fare (time & distance).	Additive
Extra	Extras and surcharges.	Additive
MtaTax	MTA tax applied to the trip.	Additive
TipAmount	Tip amount (card tips only).	Additive
TollsAmount	Total tolls paid.	Additive
ImprovementSurcharge	Improvement surcharge at flag drop.	Additive
TotalAmount	Total amount charged to passenger (no cash tips).	Additive
CongestionSurcharge	NYS congestion surcharge amount.	Additive
AirportFee	Airport fee for JFK/LGA pickups.	Additive
CbdCongestionFee	CBD congestion relief fee.	Additive
TripDurationMinutes	Duration from pickup to dropoff, in minutes (derived).	Additive
AverageSpeedMph	trip_distance / (duration_hours) (derived).	Non-additive
TotalSurcharges	Sum of all surcharges and taxes (derived).	Additive
NetAmountExclTips	total_amount − tip_amount (derived).	Additive

Table 9: Fact Trip

Fact_Daily_ZoneVendor

Star	Daily_ZoneVendor	*Version*	1.0	*Date*	02/01/26
Granularity	One row per day, per vendor, per pickup location				
Dimension Name	*Role in Fact Table*				
Date	Date				
Vendor	Vendor				
Location	PickupLocation				
Measure	*Description*			*Additivity*	
TripsCount	Number of trips in that day/vendor/location.			Additive	
TotalTripDistance	Sum of trip_distance.			Additive	
TotalFareAmount	Sum of fare_amount.			Additive	
TotalTotalAmount	Sum of total_amount.			Additive	
TotalTipAmount	Sum of tip_amount.			Additive	
TotalTollsAmount	Sum of tolls_amount.			Additive	
TotalDurationMinutes	Sum of trip_duration_minutes.			Additive	
MaxSimultaneousTrips	Max Simultaneous Trips during day D, for vendor V at location L			Semi-Additive	
OpenTripsAtPeakHourMorning	Open Trips At 09h during day D, for vendor V at location L			Semi-Additive	
OpenTripsAtPeakHourNight	Open Trips At 18h during day D, for vendor V at location L			Semi-Additive	

Table 10: Fact Daily_ZoneVendor

Dimensional Data Model

The dimensional model follows a start schema approach, as measures are stored in fact tables at a defined granularity level, while descriptive context is stored in reusable dimensions that are connected. This structure supports OLAP operations such as slice, dice, roll-up, and drill-down, while keeping analytical queries concise and clear.

Two related stars were implemented. Fact_Trip preserves detail about each trip and supports behavioural and analysis for each day (hourly patterns, payment behaviour, characteristics of the trip, comparisons between pickup and drop-off zones). Fact_Daily_ZoneVendor is a derived aggregate that stores daily totals by vendor and pick up

location in order to speed up trend and ranking queries, avoiding repeatedly scanning the detailed fact table.

This second fact table also introduces semi-additive operational measures. Semi-additive measures are those that can be meaningfully aggregated along some dimensions, but not all. For example, metrics such as OpenTripsAtPeakHourMorning, OpenTripsAtPeakHourNight, and MaxSimultaneousTrips behave like daily snapshots: they can be compared and aggregated across vendors and locations for a given day, but summing them across multiple days does not yield a meaningful result.

Temporal analysis is also enabled through two temporal dimensions: Dim_Date (calendar) and Dim_TimeOfDay (minute-level reference with time buckets). Geographic analysis is enabled by Dim_Location, which stores the zones and can be joined in both pickup and drop-off roles. Operational and behavioural slicing is supported by Dim_TripCharacteristics and Dim_PassengerGroup, reducing repetition of multiple boolean fields in the fact tables. The dimensions Dim_Vendor, Dim_RateCode and Dim_PaymentType also provide crucial information for the operations, as they describe the provider, tariffs applied and the type of payment, respectively.

Dimensions such as Dim_Date, Dim_TimeOfDay and Dim_Location are reused in multiple roles. Fact_Trip stores separate keys for both pickup and drop-off in order to preserve directionality while still maintaining a single conformed dimension table for each domain. Dim_Vendor is conformed across both stars, enabling consistent vendor comparisons at both trip and daily granularities.

Most of the monetary and distance measures are fully additive across all dimensions. Derived rates like AverageSpeedMph are non-additive and should be aggregated using averages. For snapshot measures in Fact_Daily_ZoneVendor, such as the maximum number of simultaneous trips, these are semi-additive: they can be aggregated for a single day and across vendors, but it is not meaningful to sum these measures across multiple days. This distinction is important for correct aggregation in reports.

Fig.1 presents the dimensional data model of the taxi data warehouse. The design contains two star schemas, according to the two fact tables that were generated. The main schema is centred on Fact_Trip, at the grain of one row per taxi trip, and links to role-playing dimensions for pickup and dropoff date, pickup and dropoff time-of-day, and pickup and dropoff location, as well as to Vendor, RateCode, PaymentType, PassengerGroup, and TripCharacteristics.

The second schema, Fact_Daily_ZoneVendor, is a daily snapshot at the grain (Date, Vendor, Pickup Location), implemented with a composite primary key over these foreign keys. It supports aggregated reporting through additive totals and includes semi-additive measures, such as the maximum number of simultaneous trips and the number of open trips at peak hours.

ETL

Data Preparation and Staging

The data used in this project originates from publicly available datasets provided by the NYC Taxi and Limousine Commission (TLC). The primary source is the Yellow Taxi trip records for July 2025, distributed in Parquet format and containing on the order of four million trip-level observations, complemented by a taxi zone lookup table in CSV format that maps location identifiers to NYC zones, boroughs, and service areas. As illustrated in **Fig. 2**, the pipeline starts with a Python-based pre-processing stage that first loads the full monthly Parquet dataset and then applies uniform random sampling to cap the working set at 15,000 trips. It subsequently removes records with missing passenger count values, which usually reduces the sampled dataset to slightly above 10,000 trips. The remaining records are standardized by mapping coded attributes into descriptive values for *RatecodeID*, *payment_type*, and *VendorID*, and by normalizing the *store_and_fwd_flag* by assigning “N” to missing values. In parallel, the zone lookup table is loaded and corrected for special cases by setting *Zone* and *service_zone* to “Unknown” when *Borough* is “Unknown”, and by setting *Borough* and *service_zone* to “Outside of NYC” when *Zone* is “Outside of NYC”. The trip records are then enriched through left outer joins that attach pickup and drop-off zone metadata by matching *PULocationID* and *DOLocationID*, after which these location identifier columns are removed. Finally, the dataset is exported as a processed CSV file whose filename includes a newly generated short UUID, ensuring that each execution produces a distinct output reflecting a different random sample. The processed dataset includes vendor information in *VendorID* as a descriptive string, pickup and drop-off timestamps in *tpep_pickup_datetime* and *tpep_dropoff_datetime*, core trip attributes in *passenger_count* and *trip_distance*, mapped categorical fields in *RatecodeID* and *payment_type*, and a normalized *store_and_fwd_flag*. It also contains the main fare components and totals in *fare_amount*, *extra*, *mta_tax*, *tip_amount*, *tolls_amount*, *improvement_surcharge*, *total_amount*, *congestion_surcharge*, *Airport_fee*, and *cbd_congestion_fee*, along with the enriched zone fields *PU_Borough*, *PU_Zone*, *PU_Service_zone*, *DO_Borough*, *DO_Zone*, and *DO_Service_zone*.

After the Python pre-processing stage, the processed CSV file is loaded into the data warehouse through a staging table *Stg_Trip*, as illustrated in **Fig. 3**. The table is created with a fixed schema that consolidates timestamps, numeric measures, categorical descriptors, and location metadata in a single relational structure. Data is ingested using *LOAD DATA LOCAL INFILE* in append mode, and most conversions are applied directly during loading via the *SET* clause. Pickup and drop-off timestamps are parsed into datetime values, and date and time keys are derived from them to support later joins with time dimensions. Passenger count is cast to an integer when present and is also mapped into a passenger group code, while distance and monetary fields are coerced into numeric form with default values when required. The ETL also derives indicator flags for airport-related trips, CBD-fee trips, and congestion-surcharge trips, computed directly from the presence of positive values in the corresponding fee fields and, for airport trips, reinforced by pickup zone metadata. Finally, additional measures such as trip duration, average speed, total surcharges, and net amount excluding tips are computed.

Dimension Tables

Before constructing the temporal dimensions, a basic validity gate is applied to *Stg_Trip* to remove rows with missing timestamps and trips with inconsistent temporal order, namely cases where *PickupDT* is null, *DropoffDT* is null, or *PickupDT* is greater than or equal to *DropoffDT*, as illustrated in **Fig. 4**. After this filtering step, the date dimension *Dim_Date* is built from the remaining valid trip intervals by first extracting the minimum pickup date and the maximum drop-off date from *Stg_Trip*. A continuous date range is then generated between these bounds, producing one row per calendar day covered by the dataset. For each date, a surrogate-friendly *DateKey* is derived in YYYYMMDD format and standard calendar attributes are computed, including day, month, month name, quarter, year, day-of-week number and name, and a weekend flag. The resulting rows are inserted into *Dim_Date*, providing a complete July calendar dimension to support time-based filtering and aggregation in later queries.

The time-of-day dimension *Dim_TimeOfDay* is generated as a complete enumeration of all minutes in a day, as illustrated in **Fig. 5**. The ETL creates the dimension by cross joining the set of hours from 0 to 23 with the set of minutes from 0 to 59, producing 1440 rows that cover the full daily cycle. For each hour and minute combination, a *TimeKey* is computed as minute-of-day using the expression hour multiplied by 60 plus minute, matching the time keys derived earlier in *Stg_Trip*. A formatted label is also created in HH:MM format to support readable reporting, and each minute is assigned to a coarse-grained *TimeBucket* based on the hour of day, separating LateNight, Morning, Afternoon, Evening, and Night periods. The resulting rows populate *Dim_TimeOfDay*, yielding a deterministic, gap-free minute-level reference for later analysis.

The vendor dimension *Dim_Vendor* is derived from the standardized vendor names stored in *Stg_Trip*, as illustrated in **Fig. 6**. The dimension is defined with a surrogate primary key *VendorKey* and includes both a numeric *VendorCode* and a descriptive *VendorName*, with uniqueness enforced on each of these attributes. During population, distinct vendor names are selected from *Stg_Trip* and a canonical vendor code is assigned using a simple mapping that translates the known TLC vendor strings into stable numeric identifiers. These distinct rows populate *Dim_Vendor*, creating a compact and consistent vendor reference for later fact-table joins.

The location dimension *Dim_Location* is built from the pickup and drop-off zone attributes stored in *Stg_Trip*, as illustrated in **Fig. 7**. To capture all relevant geographic categories, the ETL extracts location descriptors from both sides of each trip by selecting the pickup fields *PUBorough*, *PUZone*, and *PUServiceZone* and the corresponding drop-off fields *DOBorough*, *DOZone*, and *DOServiceZone*, and then combines them using a *UNION ALL*. For each candidate location, two binary attributes are derived, *IsAirport* and *IsCBD*. The airport flag is set when an airport fee is present or when the service zone equals Airports or the zone name contains the string Airport, while the CBD flag is set when the CBD congestion fee is positive. The combined set is then grouped by borough, zone, and service zone, using the maximum of each flag to ensure that a location is marked as airport-related or CBD-related whenever it appears in that context in any trip record. A stable numeric *LocationID* is generated deterministically as a CRC32 hash of the concatenated borough, zone, and service zone values, and the resulting distinct locations are inserted into

Dim_Location with uniqueness constraints to prevent duplicate geographic entries and to provide a consistent location lookup for downstream joins.

The rate code dimension *Dim_RateCode* is populated directly from the descriptive rate code values stored in *Stg_Trip*, as illustrated in **Fig. 8**. The dimension consists of a surrogate key *RateCodeKey* and a single descriptive attribute *RateCodeDesc*, with a uniqueness constraint to prevent duplicates. During loading, the ETL selects the distinct set of *RateCodeDesc* values present in *Stg_Trip* and inserts them into *Dim_RateCode*, so that rate code categories are represented once and can be referenced consistently whenever the fact tables and queries require them.

The payment type dimension *Dim_PaymentType* is populated from the descriptive payment category values stored in *Stg_Trip*, as illustrated in **Fig. 9**. The table is defined with a surrogate key *PaymentTypeKey* and a single descriptive attribute *PaymentTypeDesc*, with a uniqueness constraint to guarantee one row per payment category. During loading, the ETL extracts the distinct set of *PaymentTypeDesc* values present in *Stg_Trip* and inserts them into *Dim_PaymentType*, resulting in a small, stable reference table that can be reused consistently across the analytical model.

The passenger group dimension *Dim_PassengerGroup* is a small generated reference table that defines the discrete passenger group codes used in *Stg_Trip*, as illustrated in **Fig. 10**. The table is created with a surrogate key *PassengerGroupKey* and stores a stable *PassengerGroupCode* alongside a human-readable description and the corresponding passenger count range through *MinPassengers* and *MaxPassengers*. Rather than being derived from the data, the dimension is populated explicitly with a fixed set of categories, covering solo trips, couples, small groups, large groups, and an unknown bucket. This provides a clear and consistent grouping layer that downstream joins and reports can rely on when analysing trips by passenger capacity.

The trip characteristics dimension *Dim_TripCharacteristics* captures common operational flags as a single categorical reference, as illustrated in **Fig. 11**. The table is defined with a surrogate key *TripCharacteristicsKey* and stores the combination of *StoreAndForwardFlagBool*, *IsAirportTrip*, *IsCBDTrip*, and *IsCongestionSurcharge*, with a composite uniqueness constraint to ensure that each distinct flag pattern appears only once. The ETL populates the dimension by selecting the distinct combinations of these four indicators from *Stg_Trip* and inserting them into *Dim_TripCharacteristics*, allowing the fact tables to reference trip-level characteristics through a single key rather than repeating multiple boolean columns.

Fact Tables

The first fact table, *Fact_Trip*, materialises the trip-level analytical dataset by joining *Stg_Trip* with all previously constructed dimensions, as illustrated in **Fig. 12**. Its grain is one row per taxi trip, preserving the original trip measures while replacing descriptive attributes with surrogate keys to support a star-schema layout. Each record links to the temporal dimensions through *PickupDateKey*, *DropoffDateKey*, *PickupTimeKey*, and *DropoffTimeKey*, and to the remaining dimensions through *VendorKey*, pickup and drop-off *LocationKey*

values, *RateCodeKey*, *PaymentTypeKey*, *TripCharacteristicsKey*, and *PassengerGroupKey*. The fact measures include distance, fare components, totals, and derived metrics such as trip duration, average speed, total surcharges, and net amount excluding tips. Population is performed through inner joins that match staging values to their corresponding dimension rows, ensuring that only trips with complete dimensional references are loaded into the fact table. Finally, foreign key constraints are added to explicitly enforce referential integrity between *Fact_Trip* and all associated dimensions, anchoring downstream analysis on consistent, validated joins across the model.

The second fact table, *Fact_Daily_ZoneVendor*, is a derived aggregation built on top of *Fact_Trip* to support fast daily reporting by geography and vendor, as illustrated in **Fig. 13**. Its grain is one row per day, per vendor, and per pickup location, enforced through a composite primary key on *DateKey*, *VendorKey*, and *PickupLocationKey*. The table stores daily totals and counts, including the number of trips, total distance, fare and total amounts, tips, tolls, and total trip duration minutes, which are computed through a grouped aggregation over *Fact_Trip* using the pickup date key. In addition to these additive measures, the ETL derives operational intensity indicators that are harder to obtain on the fly. Two “open trips” metrics count how many trips are active at fixed peak times, using time-window conditions that check whether a trip started before the target minute and ended after it on the same day. A separate computation estimates the daily maximum number of simultaneous trips by constructing pickup and drop-off events with positive and negative deltas, applying a running sum window over time to obtain concurrent trip counts, and then taking the maximum per group. The resulting aggregates are combined through left joins and inserted into *Fact_Daily_ZoneVendor*, providing a compact summary table that enables high-level trend analysis without scanning the full trip-level fact table.

Querying and Data Analysis

After constructing and populating the dimensional model, the data warehouse is explored through a set of analytical queries aimed at validating the model and demonstrating core OLAP operations. The analysis is conducted from multiple perspectives: temporal evolution, geographic distribution, and operational efficiency. The results are visualised in a Power BI dashboard organised into three pages: Overview, Analysis & Metrics, and Vendor Insights.

The analysis begins with the Overview page (Fig. 14), which establishes the baseline activity. First, the Key Performance Indicators (KPIs) reveal the overall size of the dataset (approximately 11k trips in the sampled set), while the map visualisation highlights the spatial density of pickups. The daily revenue trend line serves as a multi-level analytical tool supported by a temporal hierarchy. By leveraging the drill-down features, users can navigate from a macro daily view down to finer granularities, specifically from Day to TimeBucket, and finally to the specific Hour. This hierarchical capability allows analysts to investigate not just which days were most profitable, but to drill deeper to understand when within those days the revenue was generated (e.g., distinguishing Morning Rush peaks from Late Night surges) without leaving the Overview dashboard. Meanwhile, comparisons like the "Revenue Change (H2 vs H1)" chart confirm that Brooklyn saw the highest relative growth later in the month.

The Analysis & Metrics page (Fig. 15) focuses on behavioural patterns and is anchored by a Borough slicer at the top right, which serves as a control for geographic drill-down. This slicer allows the user to isolate data for specific districts (e.g., Manhattan vs. Queens), revealing that traffic patterns are highly dependent on location. For instance, the "Trips Heatmap" currently displays the aggregate city-wide view, a scarcity of trips in the early morning (03:00–05:00), followed by a surge during weekday evenings (18:00–21:00) and late-night peaks on weekends. By using the slicer, analysts can verify if outer boroughs follow this commuter trend or exhibit different demand cycles.

To understand revenue distribution, a Pareto analysis ("Revenue Concentration") was performed on the pickup zones. The chart combines total revenue bars with a cumulative percentage line, visually demonstrating the "80/20 rule". The results display that a small subset of zones, specifically Times Square, JFK Airport, and Midtown Centre, dominate the market. As shown by the yellow cumulative line, the top 20 zones alone (out of over 260) account for the vast majority of income, confirming that there is a highly concentrated market structure around transport hubs and tourist centres. Finally, the "Trips by Payment Method" doughnut chart shows that Credit Card transactions account for approximately 82% of the volume. This overwhelming dominance over cash suggests a user base consisting largely of business travellers and tourists who prefer digital payment methods, a trend that is likely even more pronounced in the high-revenue zones identified in the Pareto chart.

Finally, the Operational KPIs page (Fig. 16) leverages the aggregated table Fact_Daily_ZoneVendor to analyse operational performance. The dashboard is empowered by two critical interactive controls at the top right: a Date Range slider and a Vendor selector. These filters allow analysts to drill down into specific time windows, isolating, for exemple an holiday week, or to toggle between providers to compare market presence. The "Daily Fleet Load" area chart visualises the capacity stress and reveals a clear market disparity, with Curb Mobility consistently displaying significantly higher simultaneous trip volumes than its competitor, Creative Mobile.

The analysis of "Rush Hour" (9:00 AM vs. 6:00 PM) provides a view of commuter flows. The data highlights a strong temporal skew, particularly in the borough of Manhattan, where the demand at 6:00 PM is nearly double that of the morning rush, suggesting a heavy reliance on taxis for post-work commuting and evening leisure. Furthermore, by identifying the "Top Zones by Peak Traffic", we observe that operational bottlenecks are geographically localised. JFK Airport leads this ranking with the highest peak intensity, followed by Midtown hubs, indicating that these specific locations require targeted fleet management strategies to handle simultaneous demand surges.

Comparison with Operational Databases

This chapter compares the analytical approach adopted in the data warehouse with the characteristics of operational databases, clarifying why the analyses presented earlier are better suited to a dedicated analytical environment. While operational systems are designed to support day-to-day transaction processing, the queries explored in this work focus on understanding patterns, trends, and aggregated behaviour over time and space. The

comparison helps contextualise the design choices made and highlights the complementary roles of operational databases and data warehouses within a data-driven architecture.

A key difference starts with how data is modelled and prepared for analysis. In the data warehouse developed in this project, the multidimensional model structures data around fact tables and shared dimensions, allowing analytical questions to be expressed directly in terms of time, location, vendor, and trip characteristics. The separation between Fact_Trip, which preserves trip-level detail, and Fact_Daily_ZoneVendor, which stores pre-aggregated daily measures, reflects a deliberate design choice to support different analytical needs efficiently. By contrast, operational databases typically follow highly normalised schemas optimised for update consistency, where analytical queries must reconstruct this multidimensional context through multiple joins, making complex analysis more cumbersome and less transparent.

Building on this design, operational and analytical workloads also differ in the way data is accessed and in the structure of the queries themselves. In this project, analytical queries frequently scan and aggregate large portions of the warehouse to compute roll-ups, rankings, cumulative shares, and complex semi-additive measures. A prime example is the analysis of "Daily Fleet Load" (Max Simultaneous Trips) presented in the Vendor Insights dashboard. Calculating the maximum number of simultaneous trips in a purely operational database would require expensive self-joins to check time overlaps for every single trip record. In our data warehouse, this heavy computation is pre-calculated during the ETL process and stored in Fact_Daily_ZoneVendor, allowing the dashboard to render these trends instantly.

This read-intensive and computation-heavy access pattern is characteristic of OLAP workloads. In contrast, executing the same analyses directly over an operational schema would require repeatedly aggregating detailed records and reconstructing analytical context, which is inefficient and poorly aligned with transactional performance goals. Overall, this comparison illustrates that the data warehouse does not replace operational databases, but rather complements them by providing an environment specifically designed for flexible exploration across multiple dimensions, supporting analyses that would be impractical in a purely operational setting.

Conclusion

This project implemented a comprehensive Data Warehouse for NYC Taxi and Limousine Commission data, converting raw transactional records into actionable business intelligence. Using a structured methodology from ETL design to dimensional modelling and final visualisation, the system supports analytical needs related to demand, revenue, and operational pressure.

A Star Schema architecture was the main element. The main star schema with Fact_Trip enabled detailed analysis of rider behaviour, including payment preferences and hourly demand patterns. A second aggregated star schema with Fact_Daily_ZoneVendor showed the value of derived fact tables by enabling fast analysis of semi-additive measures such as

simultaneous fleet load and rush hour comparisons, which would be too costly to compute directly from raw transactions.

The final Power BI dashboard functions as a decision support system, helping stakeholders explore revenue trends and drill down into operational bottlenecks across boroughs. The results indicate that the NYC taxi market is concentrated in a small number of zones and follows consistent temporal patterns, offering useful insights for improving management and identifying consistent patterns among passengers.

Annexes

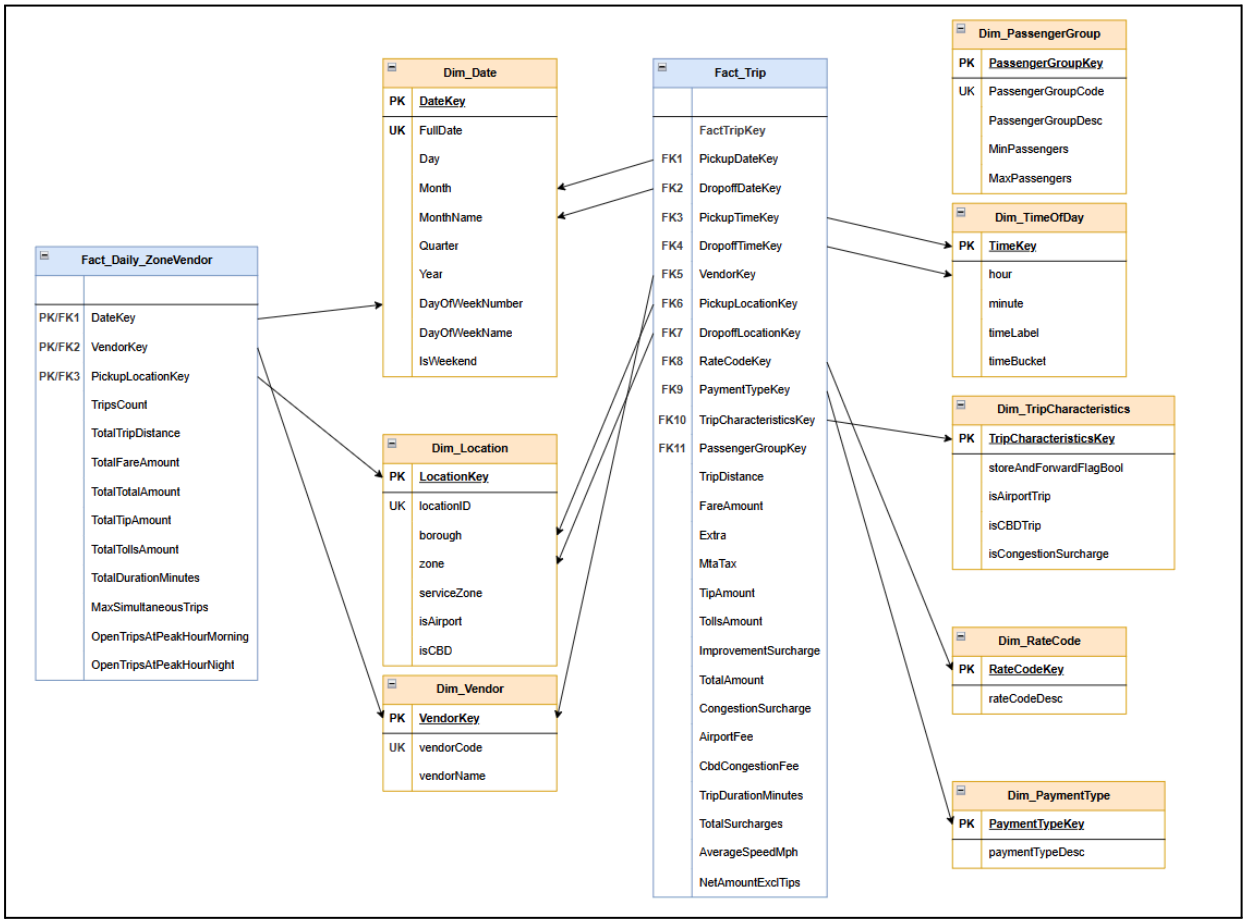

Figure 1: Dimensional data model

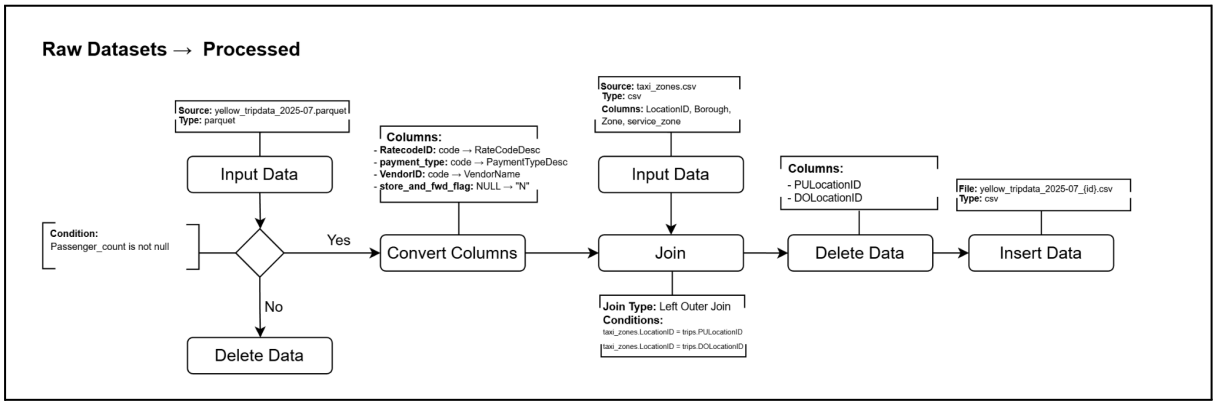

Figure 2: Pre-processing of raw datasets into processed files

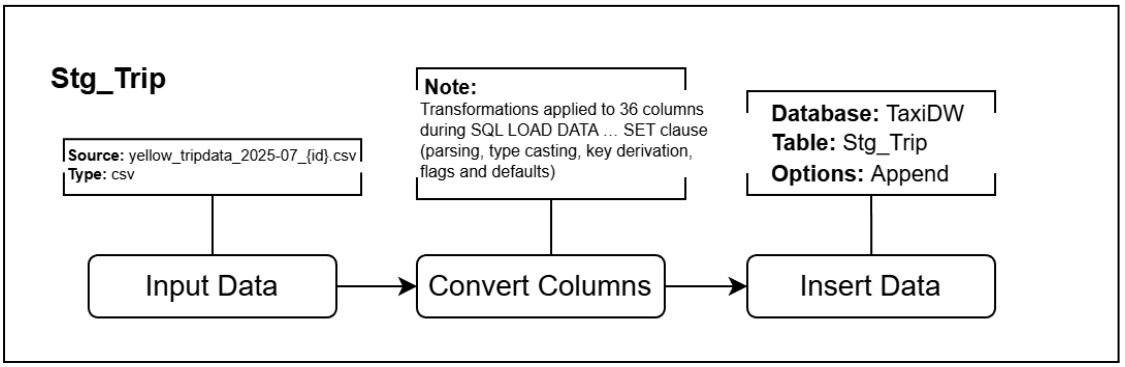

Figure 3: Loading process of the Stg_Trip table

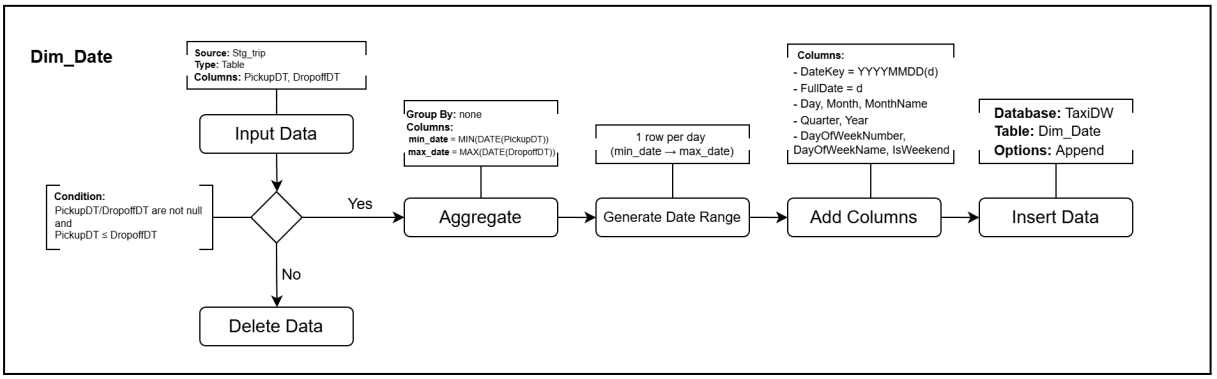

Figure 4: Construction of the Dim_Date dimension

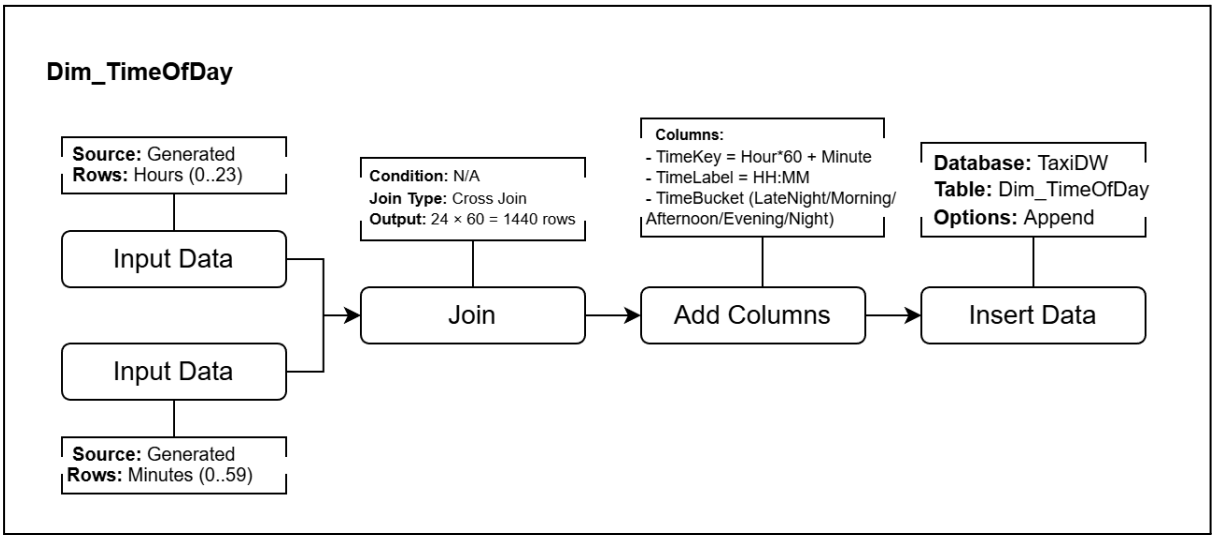

Figure 5: Construction of the Dim_TimeOfDay dimension

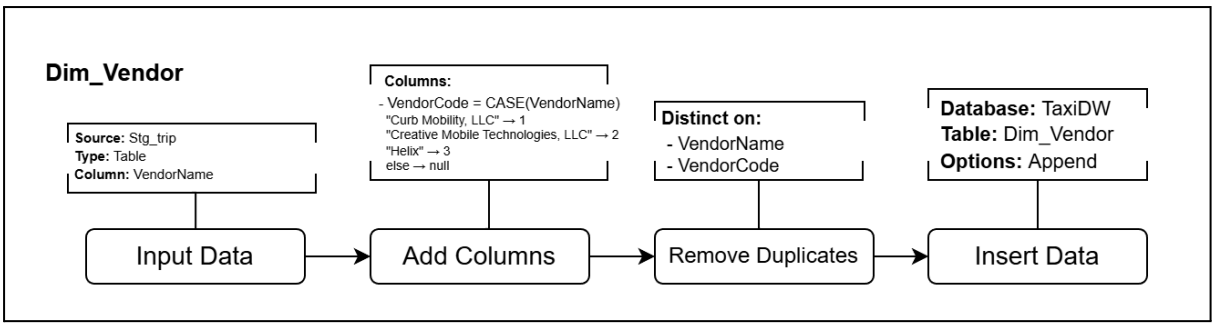

Figure 6: Construction of the Dim_Vendor dimension

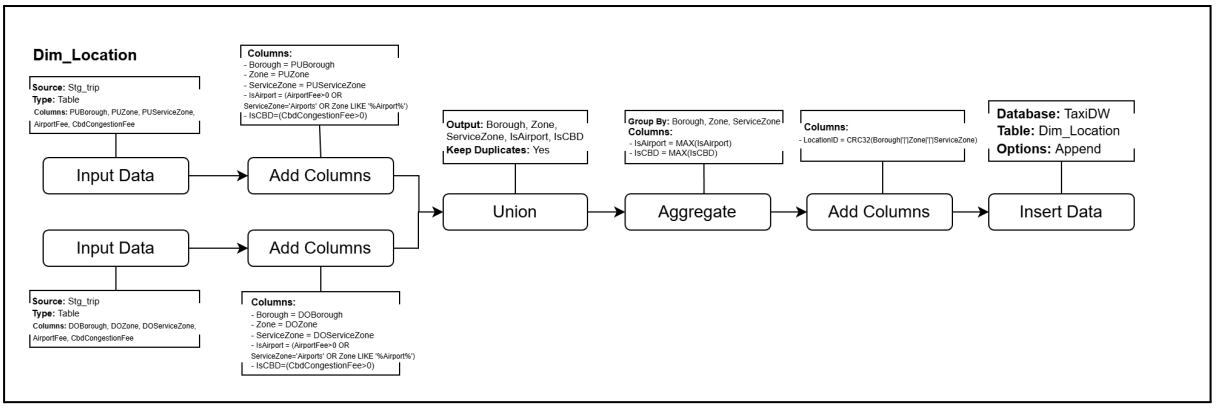

Figure 7: Construction of the Dim_Location dimension

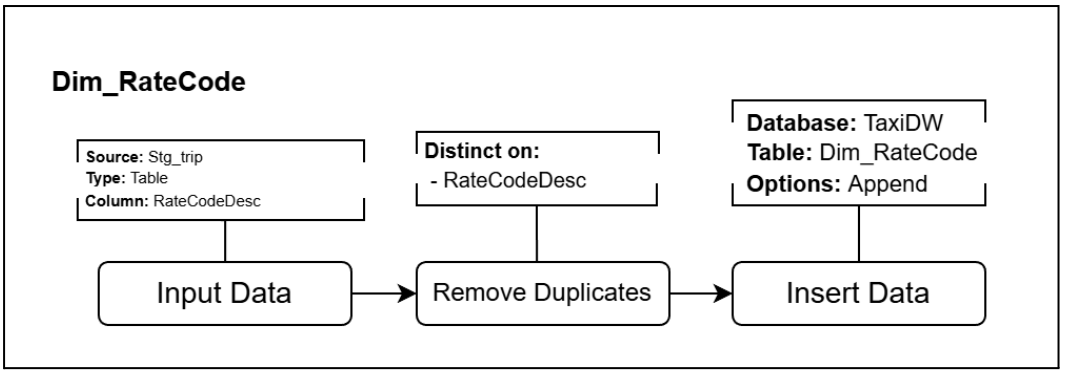

Figure 8: Construction of the Dim_RateCode dimension

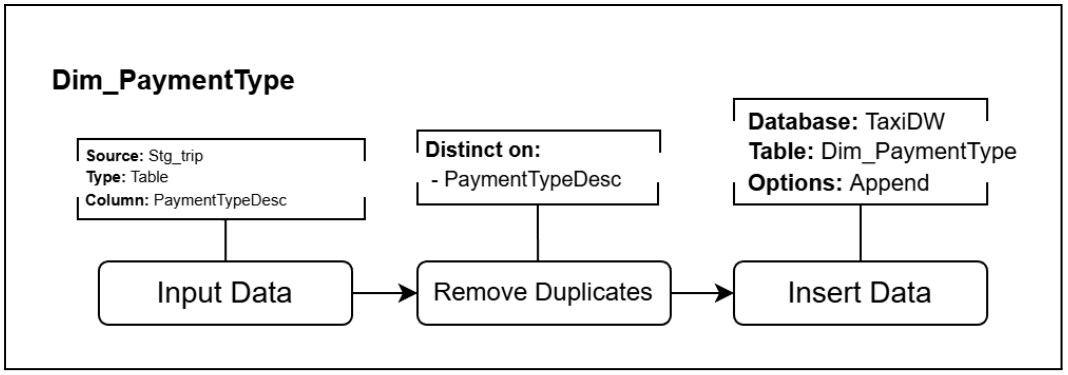

Figure 9: Construction of the Dim_PaymentType dimension

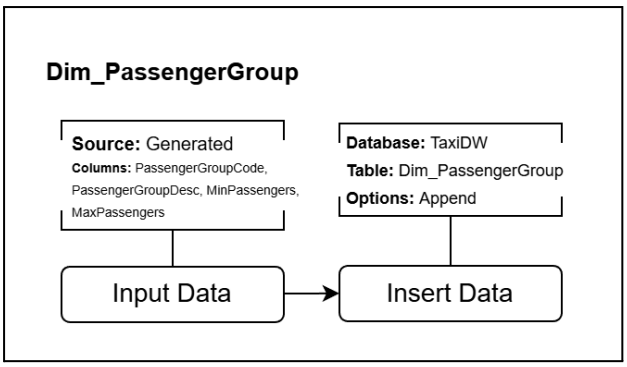

Figure 10: Construction of the Dim_PassengerGroup dimension

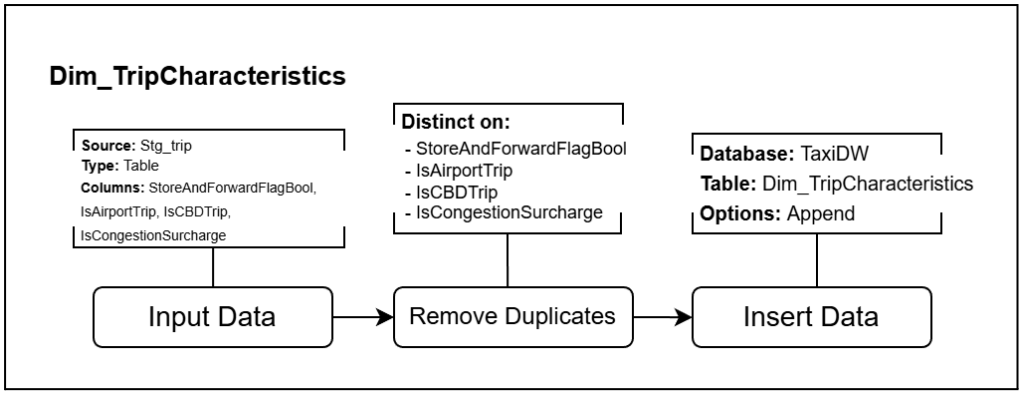

Figure 11: Construction of the Dim_TripCharacteristics dimension

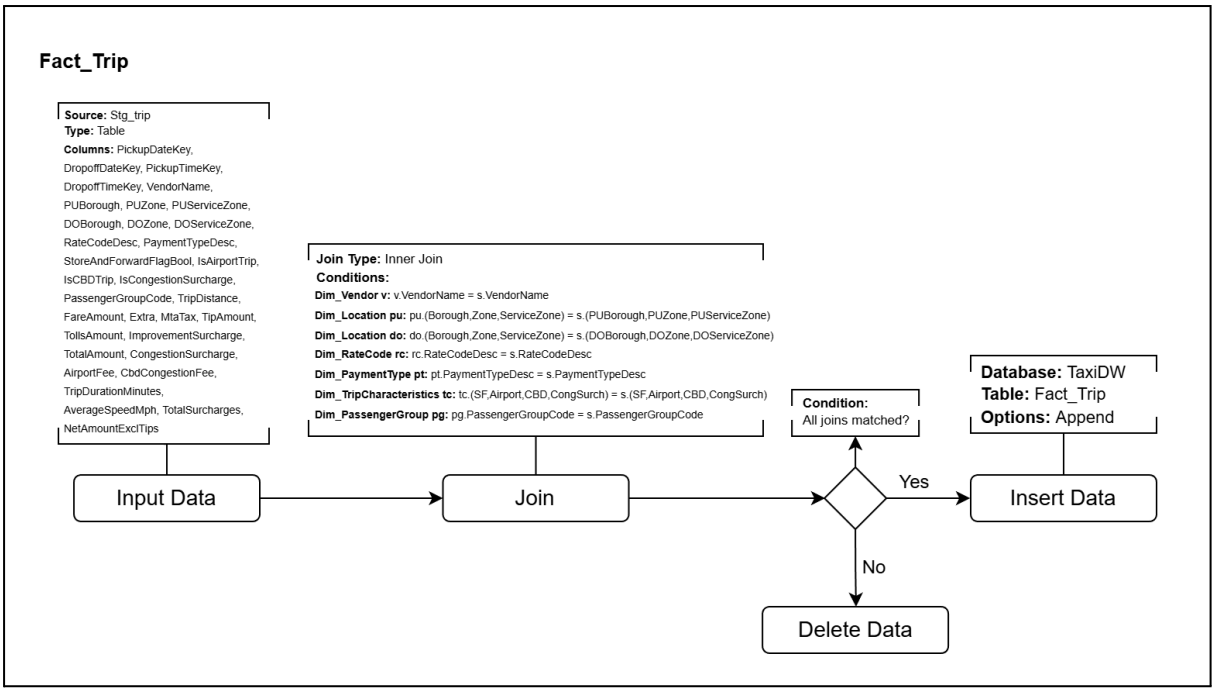

Figure 12: Loading of the Fact_Trip table

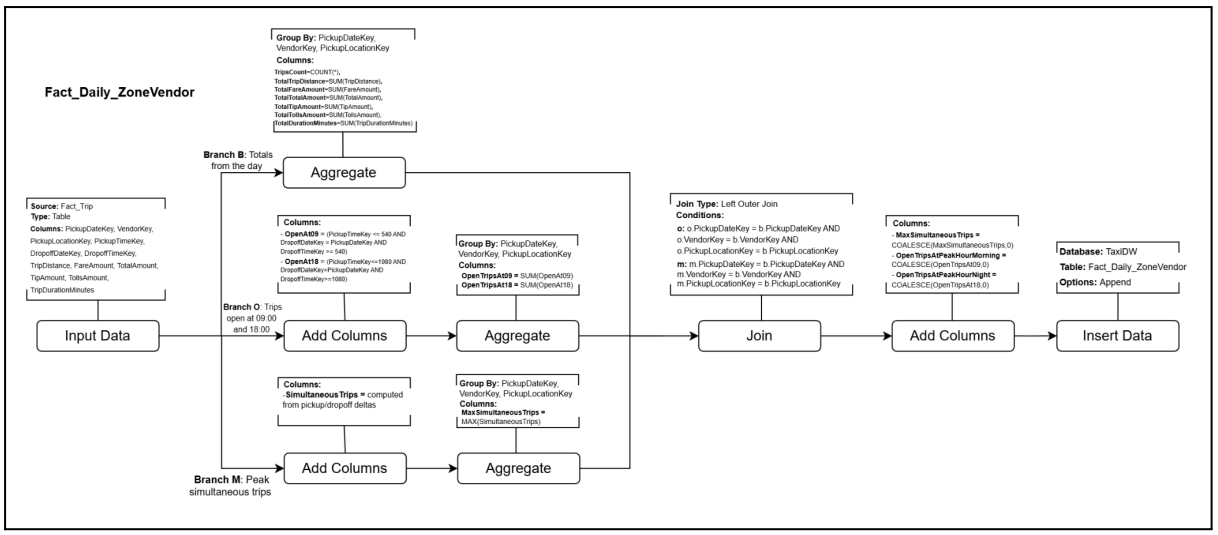

Figure 13: Construction of the Fact_Daily_ZoneVendor table

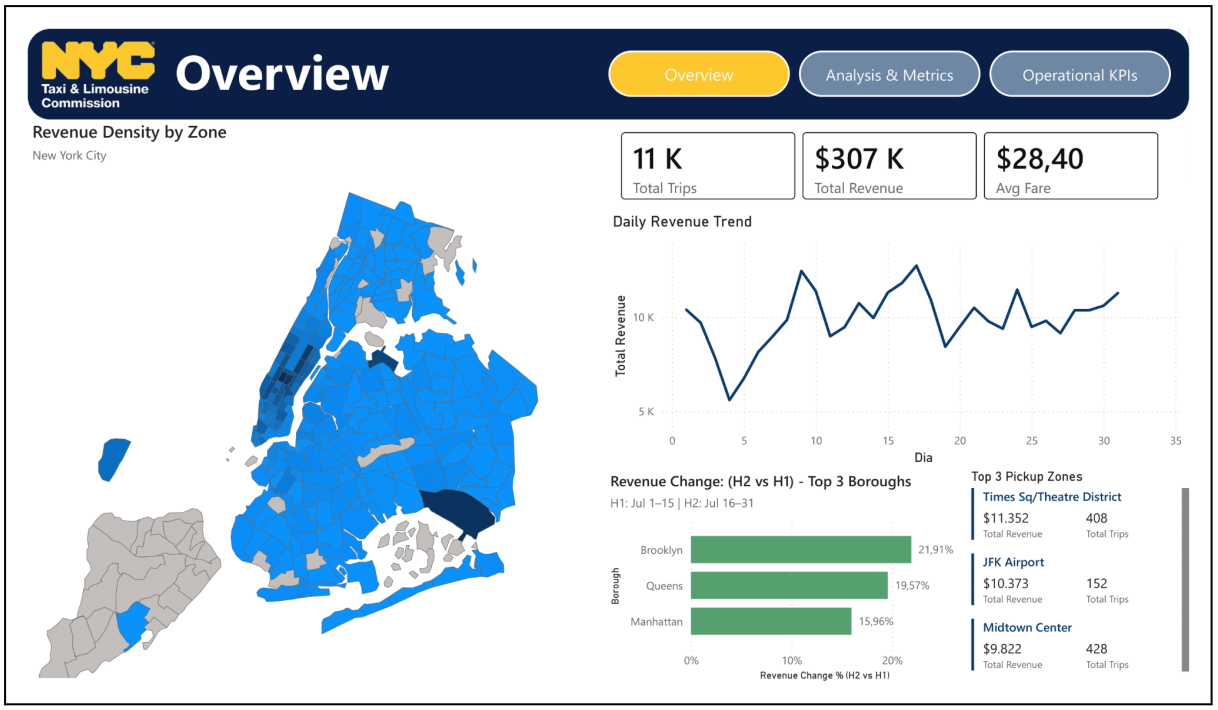

Figure 14: Dashboard: Revenue Overview

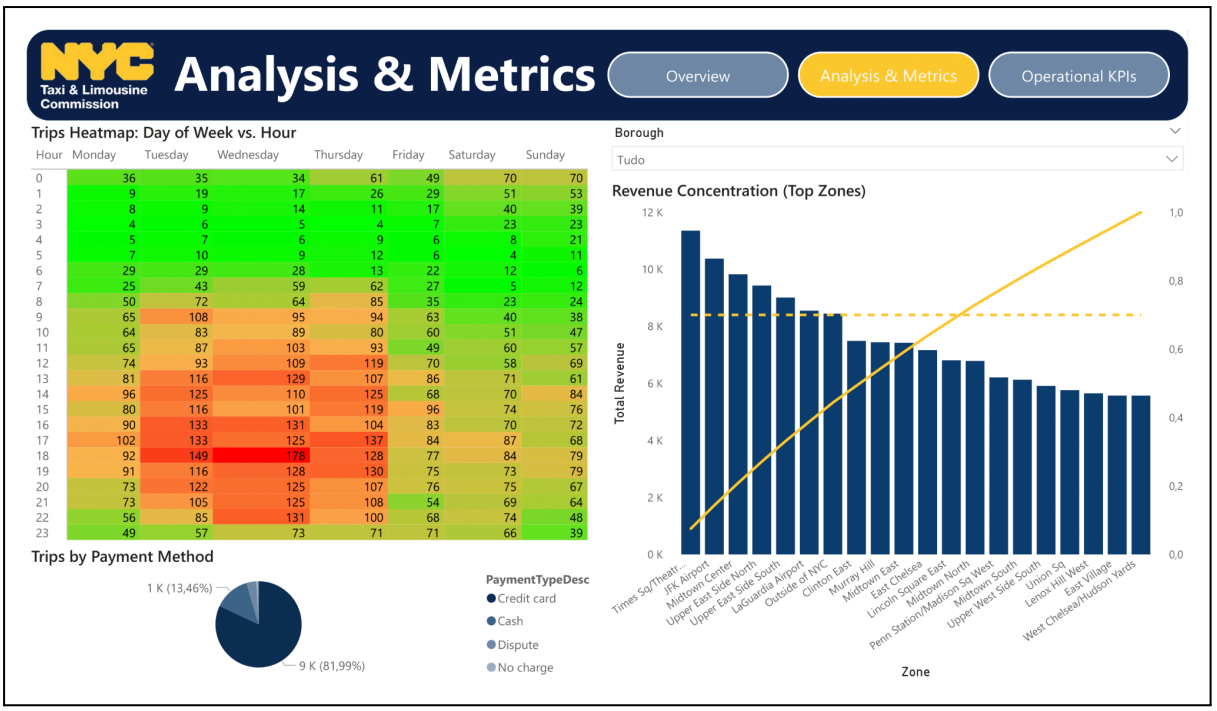

Figure 15: Dashboard: Metrics & Analysis

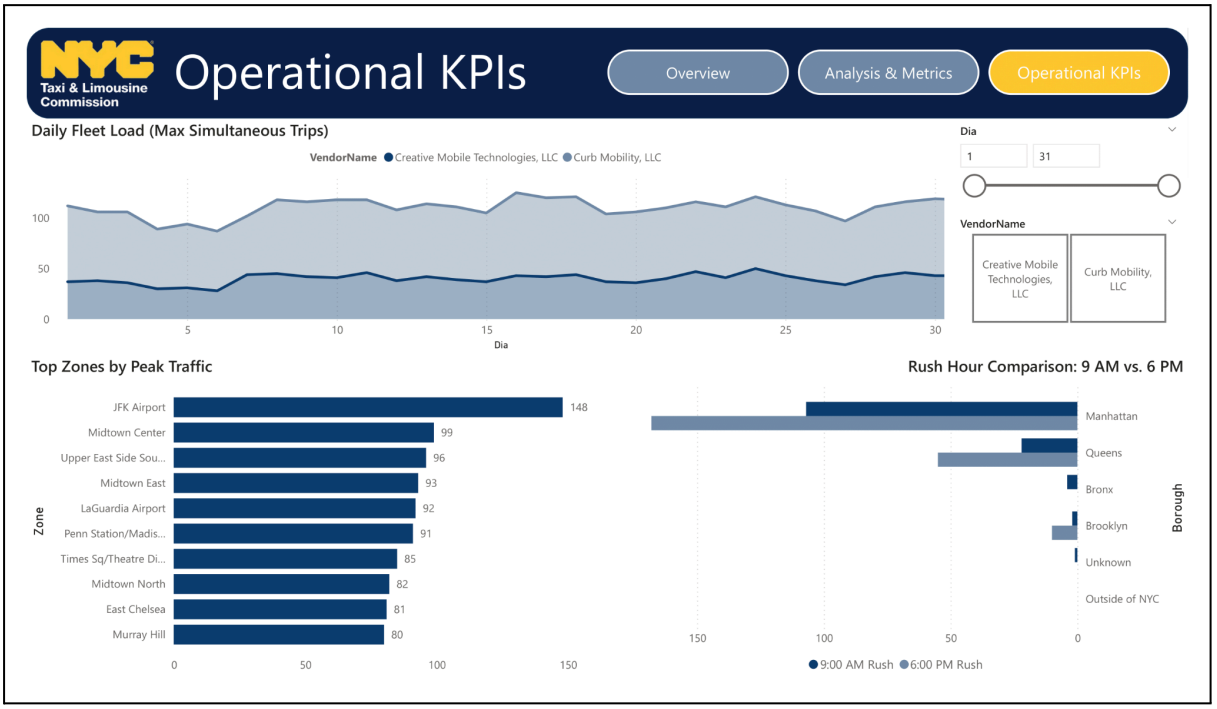

Figure 16: Dashboard: Operational KPIs