UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

POLYTECHNIC UNIVERSITY OF VALENCIA
Higher Polytechnic School of Alcoy

Master's Degree in Computational Engineering & Industrial Mathematics

Big Data Engineering and Technologies

Building a Scalable 3-Tier Data Lakehouse for Mobility Analysis in Spain

Authors
María López Hernández
Joan Sánchez Verdú
Fernando Blanco Membrives

January 17, 2026

Contents

1 Introduction

The availability of high-resolution mobility data provided by the Spanish Ministry of Transport (MITMA) has significantly enhanced the potential for data-driven urban planning. However, the magnitude of these datasets, comprising billions of daily trip records, poses engineering challenges that surpass the limits of conventional analytical software. Transportation planners often lack the accessible, scalable infrastructure necessary to convert this raw big data into strategic insights.

This project bridges this technical gap by implementing a Data Lakehouse architecture. By integrating the cost-effectiveness of data lakes with the structured data management of data warehouses, we propose a solution leveraging **DuckDB** and **DuckLake**. This infrastructure enables the efficient processing of massive mobility matrices and their enrichment with socio-economic indicators, avoiding the high complexity and maintenance overhead of traditional enterprise clusters.

1.1 Objectives

The primary goal of this project is to build a robust, reproducible and cost-efficient data platform. The specific objectives are defined as follows:

- **Scalable Architecture:** Design a Medallion (Bronze, Silver, Gold) Lakehouse architecture that decouples storage (Amazon S3) from compute (AWS Batch) to maximize both cost-efficiency and query performance.
- **Data Integration:** Develop automated ELT pipelines to sanitize and fuse heterogeneous datasets, including mobility flows, census demographics and geospatial vector data.
- **Applied Analytics:** Validate the system's utility through advanced use cases, specifically applying gravity models for infrastructure impact analysis and defining functional zoning classifications.

2 Data Sources

The Data Lakehouse architecture is designed to ingest and integrate public domain data exclusively. This approach ensures the system remains reproducible, cost-effective and fully accessible. The following subsections detail the primary mobility datasets and the secondary enrichment sources selected to build the information system.

2.1 MITMA Open Data

The core mobility dataset is retrieved from the Spanish Ministry of Transport, Mobility and Urban Agenda (MITMA [1]), specifically the *Estudios Básicos de Movilidad*. This dataset provides high-resolution insights into daily population movements, derived from aggregated mobile network signaling data.

To ensure transparency and facilitate reproduction of this study, the specific files utilized from the MITMA repository are detailed in Table 1.

Table 1: MITMA Mobility and Zoning Data Sources

Data Type	**Filename / Pattern**	**Description**
Mobility Matrices	`(202301-202312)_Viajes_municipios.tar`	Daily mobility flows between origin and destination municipalities, aggregated by hour for the year 2023. *Path: `estudios_basicos/por-municipios/`*
Zoning Metadata	`nombres_municipios.csv`	Official nomenclature for each municipality. *Path: `zonificacion/zonificacion_municipios/`*
Crosswalk	`relacion_ine_zonificacionMitma.csv`	Mapping table linking MITMA custom zoning codes to standard INE municipal codes. *Path: `zonificacion/`*
Demographics	`poblacion.csv`	Annual resident population counts per municipality. *Path: `zonificacion/`*
Geometries	`zonificacion_municipios.shp`	ESRI Shapefile containing the geospatial vector polygons for the municipal study zones. *Path: `zonificacion/zonificacion_municipios/`*

2.2 Enrichment Data Sources

To contextualize the raw mobility flows and enable advanced gravity model analysis, the system integrates socioeconomic and temporal data from external official sources. These datasets allow for the characterization of municipalities by economic attractiveness and the classification of dates by business activity.

These enrichment sources are summarized in Table 2.

Table 2: Socioeconomic and Temporal Enrichment Data

Source	**Dataset / Filename**	**Description**
INE [2]	`ine_rent_municipalities.csv`	**Net Income Distribution:** Provides the average net income per year (from 2013 to 2023) and municipality. This metric serves as a key proxy for economic attractiveness in gravity models. *Source:* `https://www.ine.es/jaxiT3/files/t/es/csv_bd/`
Open Data Madrid [4]	`calendario_laboral.csv`	**Working Calendar:** Contains the official calendar from 2013 to 2026, classifying dates into working days, weekends and national holidays to segregate mobility patterns by day type.

3 Lakehouse Architecture

The system adheres to the **Medallion Architecture** pattern, a multi-hop design that progressively improves data quality. Critically, the architecture decouples storage from compute, ensuring that the system can scale to handle billions of records without the cost overhead of permanently running clusters.

3.1 Technology Stack

The platform employs a modular cloud-native stack (Figure 1), leveraging the following core components:

Compute Engine (DuckDB [5]): A high-performance, in-process SQL OLAP engine. DuckDB utilizes vectorized query execution and supports out-of-core processing, enabling the analysis of datasets significantly larger than available RAM on a single node.

Elastic Compute (AWS Batch [6]): Provides an ephemeral, serverless compute layer. It offloads resource-intensive transformation tasks to on-demand container instances, optimizing costs by leveraging Spot market pricing.

Lakehouse Management (DuckLake [7]): Functions as the transaction manager, enforcing ACID properties and snapshot isolation over object storage. This ensures data consistency and prevents race conditions during concurrent write operations.

Cloud Storage & Catalog (Neon + S3): A hybrid storage design where **AWS S3** serves as the physical data layer (storing Parquet/CSV files), while **Neon** (Serverless Postgres) acts as the centralized metadata catalog and locking mechanism.

Orchestration (Apache Airflow [8]): Manages the end-to-end data lifecycle via Directed Acyclic Graphs (DAGs), handling dependency resolution, task scheduling and automated failure recovery.

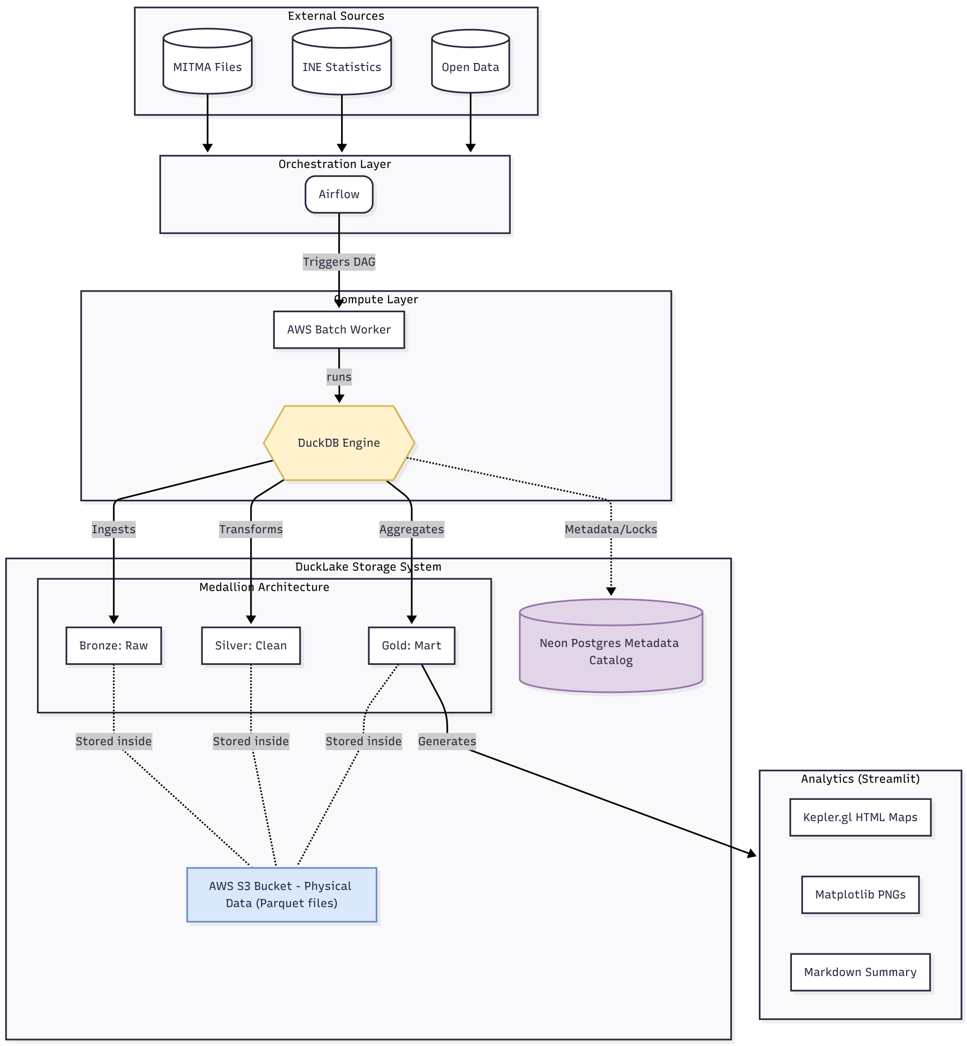


Figure 1: High-level overview of the 3-Tier Data Lakehouse architecture, decoupling compute (DuckDB) from storage (S3).

3.2 Data Layering Strategy

The data pipeline is structured into three distinct zones—Bronze, Silver and Gold—each serving a specific validation and analytical purpose. This design preserves the raw input for auditability while providing refined datasets for downstream consumption.

3.2.1 Bronze Layer (Raw Ingestion)

The Bronze layer serves as the immutable landing zone (Staging Area). Its primary objective is to capture external data in its native format with zero information loss.

In this implementation, a distinct table is instantiated for each source entity to ensure isolation. To prevent ingestion failures caused by schema drift or type mismatches, all tabular columns are initially cast as `VARCHAR` and ingested using `ignore_errors=true`. This configuration allows the pipeline to skip individual rows containing irrecoverable formatting errors (e.g., invalid dates such as '`20230139`') while preserving the continuity of large-scale batch ingestion. A notable exception is the geospatial data: by leveraging DuckDB's `spatial` extension, ESRI Shapefiles are ingested using the `ST_Read` function. This captures municipal boundaries directly as native `GEOMETRY` objects, preserving the original spatial precision.

To enhance governance, the ingestion process appends metadata columns to every record, including `ingestion_timestamp` and `source_url`. Furthermore, the high-volume mobility data is physically partitioned by date (`fecha`) to optimize I/O performance during batch processing.

3.2.2 Silver Layer (Cleaned & Integrated)

The Silver layer functions as the enterprise-wide, trusted data repository. In this stage, data undergoes rigorous validation, standardization and enrichment. The schema follows a **Star Schema** topology, centering on a mobility fact table linked to surrounding dimensions for spatial and socio-economic context.

Dimension Modeling and Spatial Integration (`dim_zones`)
The foundation of the Silver layer is the unified spatial dimension, `dim_zones`. This table integrates descriptive metadata (MITMA and INE codes) with the geospatial vectors ingested in the Bronze layer.

- **Spatial Integration:** By joining municipal geometries with the `mapping_ine_mitma` crosswalk, the system creates a spatially-enabled master dimension. Each zone persists both its full municipal `GEOMETRY (POLYGON)` and its derived `Centroid` geometry to support distinct spatial and network-based analyses.
- **Surrogate Keys:** A unique integer identifier (`zone_id`) is generated for each municipality. This decouples downstream analytics from unstable string-based source codes and improves join performance.

Spatial Connectivity (`dim_zone_distances`)
To support network analysis, a derivative dimension table, `dim_zone_distances`, serves as a pre-computed distance matrix. By calculating the Euclidean distance between the centroids of geometries in `dim_zones`, the system eliminates the need for expensive geospatial operations at query time. The `dist_km` column stores the distance in kilometers as a `DOUBLE` precision value.

Socio-Economic Enrichment
Complementary metric tables are generated to enrich the mobility data.

- `metric_population`: Raw counts are cast to `BIGINT` and stripped of metadata headers.

- `metric_ine_rent`: The pipeline filters the INE dataset to isolate the "Average Net Income per Person," explicitly excluding sub-municipal (district/section) entries to ensure 1:1 alignment with the `dim_zones` table.

Mobility Fact Table Construction (`fact_mobility`)
The core mobility flows are transformed into the `fact_mobility` table. This process converts raw CSVs into a strongly typed, time-series optimized format:

1. **Temporal Standardization:** Separate date and hour fields are merged into a single `TIMESTAMP WITH TIME ZONE` column, localized to 'Europe/Madrid'.
2. **Referential Integrity:** Origin/Destination codes are replaced by their corresponding `zone_id` via double joins against `dim_zones`.
3. **Type Casting:** Trip counts are cleaned and cast to `DOUBLE` to support fractional expansion factors.

Data Observability (`data_quality_logs`)
To ensure pipeline reliability, a dedicated metadata registry, `data_quality_logs`, persists the results of automated audit checks (e.g., null rates, row count anomalies). This allows for longitudinal tracking of data health, storing the `check_timestamp`, `metric_name` and `metric_value` for every batch execution.

3.2.3 Gold Layer (Analytical Data Mart)

The Gold layer is organized around specific business problems rather than data structures. Tables here are denormalized, pre-aggregated and optimized for low-latency querying by Business Intelligence (BI) tools.

The schema consists of three purpose-built analytical tables:

`gold_typical_day_patterns` **(Temporal Analysis):** Reduces billions of raw records into lightweight hourly profiles.

- *Purpose:* Enables instant plotting of demand curves without re-scanning raw data.
- *Structure:* Aggregated by `cluster_id` (e.g., "Weekday", "Holiday") and `hour`, storing the `avg_trips`.

`gold_infrastructure_gaps` **(Gravity Models):** A specialized Origin-Destination (OD) matrix enriched with socio-economic context.

- *Purpose:* Supports the Gravity Model analysis (Use Case 2) by combining flow data with impedance metrics.
- *Structure:* Contains `total_population` (Origin), `rent` (Destination attractiveness) and `dist_km`. The key derivative is the `mismatch_ratio`, quantifying the gap between theoretical potential and observed trips.

`gold_zone_functional_classification` **(Zoning):** A semantic layer that assigns functional roles to municipalities.

- *Purpose:* Simplifies map visualizations by converting raw IDs into human-readable classifications (e.g., "Bedroom Community").
- *Structure:* Persists calculated indices such as `net_flow_ratio` (trade balance) and `retention_rate` (self-sufficiency).

4 Implementation Methodology

The operational workflow of the Data Lakehouse is automated through a modular architecture orchestrated by **Apache Airflow** [8]. The system adopts a decoupled design where distinct Directed Acyclic Graphs (DAGs) manage specific stages of the data lifecycle, separated by their update frequency and functional scope.

This section details the engineering strategies employed to ensure scalable data ingestion, atomic processing of daily batches and the automated generation of analytical models. The implementation prioritizes robustness through strict transaction management, task isolation and the separation of heavy computational ELT processes from on-demand reporting queries.

4.1 Orchestration Strategy

To efficiently manage dependencies between the Bronze, Silver and Gold layers, the system is architected into a multi-DAG ecosystem. This design enables independent scaling for static dimensions versus high-volume mobility facts.

1. **Infrastructure & Dimensions DAG:** Manages the ingestion of low-velocity static data (Demographics, Zoning, Calendars). It establishes the schema foundations and populates the dimension tables in the Bronze and Silver layers.
2. **Mobility Ingestion DAG:** A parameterized, high-throughput worker pipeline. It utilizes dynamic task mapping to ingest, clean and transform daily mobility matrices from Bronze to Silver using atomic, parallelizable tasks.
3. **Analytical & Reporting DAGs:** These pipelines (DAGs 31–33) consolidate business logic into unified flows. Triggered on-demand with custom parameters, they materialize aggregated Gold tables and immediately generate final visual assets (Kepler.gl maps, Matplotlib charts) for future analysis.

4.2 Infrastructure & Dimensions (DAG 1)

The foundational pipeline, `infrastructure_and_dimensions`, is responsible for initializing the Lakehouse schema and ingesting low-velocity reference data. Unlike the daily mobility processing, this DAG is designed for on-demand or annual execution, as zoning definitions and census statistics change infrequently.

Initialization Phase:
Before data movement begins, the pipeline ensures environmental consistency. The `create_schemas` task connects to the Neon catalog to verify the existence of the logical namespaces (*bronze, silver, gold*). Simultaneously, the `create_stats_table` task initializes the `data_quality_log` table, a prerequisite for audit logging in all subsequent pipelines.

Ingestion Phase:
To optimize execution time, external extraction tasks run in parallel. These tasks utilize Airflow's retry mechanism (3 retries, 60s delay) to handle transient network issues. Specialized ingestion logic is applied based on the data format:

- **Spatial Data:** The `br_ingest_geo_data` task leverages DuckDB's `spatial` extension. It performs a pre-flight HTTP HEAD request to validate the remote Shapefiles before executing `ST_Read` to ingest vector geometries directly into the Bronze layer.
- **Statistical Data:** Tasks targeting INE data (e.g., `br_ingest_ine_rent`) enforce `ISO-8859-1` encoding and specific separators to prevent parsing errors common in legacy government datasets.

- **Dictionaries:** Zoning and mapping CSVs are ingested via `read_csv_auto`, with metadata columns (`source_url`, `ingestion_timestamp`) appended dynamically.

Silver Transformation (Funnel Pattern):
The dependency structure in the Silver layer follows a "Funnel" pattern (Figure 2). The system enforces a hard dependency on the Master Dimension before populating satellite metrics.

1. **Convergence Point (`dim_zones`):** This task acts as a synchronization barrier. It awaits the completion of three Bronze tasks (Geometry, Zoning names and INE Mapping), before performing a 3-way join to construct the master `dim_zones` table and generate surrogate `zone_id` keys.
2. **Satellite Expansion:** Once the master dimension is established, downstream tasks (e.g., `metric_population`, `dim_zone_distance_matrix`) trigger in parallel. These tasks join their respective raw sources with `dim_zones` to propagate the correct surrogate keys to the statistical data.

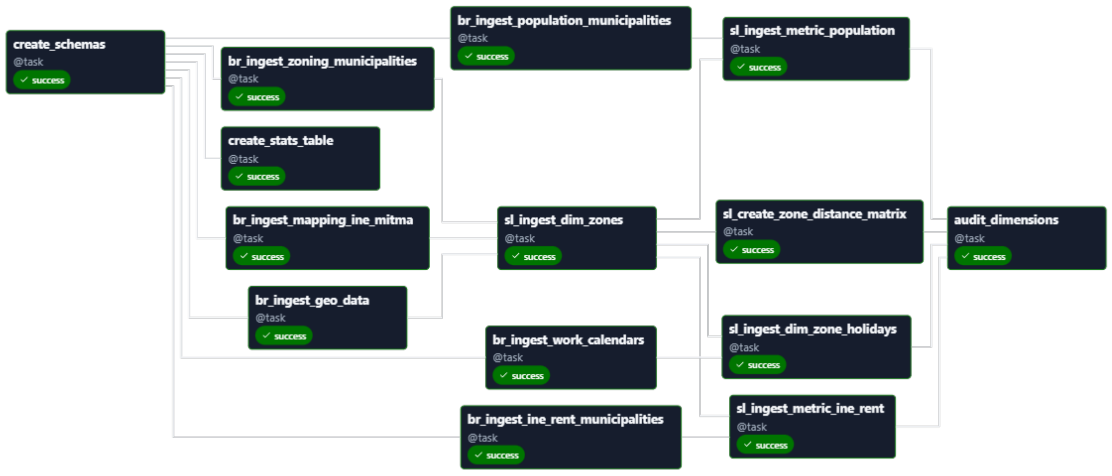

create_schemas
br_ingest_zoning_municipalities
create_stats_table
br_ingest_mapping_ine_mitma
br_ingest_geo_data
br_ingest_population_municipalities
sl_ingest_dim_zones
br_ingest_work_calendars
br_ingest_ine_rent_municipalities
sl_ingest_metric_population
sl_create_zone_distance_matrix
sl_ingest_dim_zone_holidays
sl_ingest_metric_ine_rent
audit_dimensions
@task
success

Figure 2: Dependency graph of the Infrastructure DAG, illustrating the funnel pattern for dimension creation.

4.3 Mobility Ingestion (DAG 2)

The second pipeline, `mobility_ingestion`, constitutes the core processing engine of the Lakehouse. It handles the daily ingestion and transformation of high-volume Origin-Destination matrices. Given that the source data is partitioned by day, this DAG utilizes Airflow Dynamic Task Mapping to scale horizontally based on run-time parameters.

Execution Planning:
The workflow begins with the `generate_date_list` task, which parses the user-supplied `start_date` and `end_date` to produce a list of target partition keys (e.g., `['20230101', '20230102']`). Simultaneously, idempotent initialization tasks (`ensure_br_mobility...`) prepare the partition-

ing structures in the Bronze/Silver layers to prevent race conditions during parallel writes.

Atomic Processing:
The `br_process_single_day` task is mapped over the date list, with each instance handling a specific day. Robustness is ensured through:

- **Pre-flight Validation:** An HTTP `HEAD` check prevents pipeline failure if a specific date file is missing from the source server (a known issue with MITMA late-2023 data).
- **Throttling:** Concurrency is limited (`max_active_tis=2`) to prevent memory saturation (only when local execution), with a strict retry policy for network resilience.
- **Streaming Ingestion:** Valid files are streamed directly into DuckDB memory using `read_csv_auto`, bypassing local disk I/O.

Referential Integrity:
The transformation task (`sl_process_single_day`) mirrors the ingestion mapping. It enforces integrity by joining raw mobility records with the `dim_zones` table (from DAG 1). Records with invalid or unknown zone codes are implicitly filtered, ensuring that the Silver layer contains only clean, linkable data.

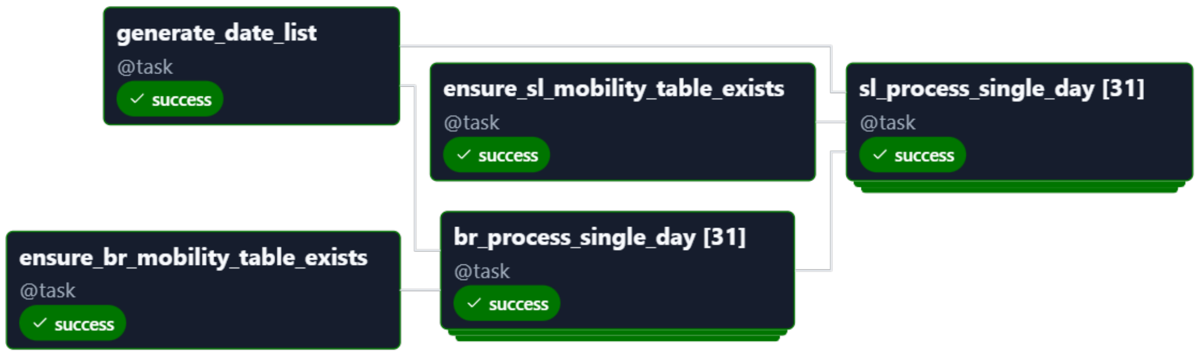

generate_date_list
@task
success
ensure_sl_mobility_table_exists
@task
success
sl_process_single_day [31]
@task
success
ensure_br_mobility_table_exists
@task
success
br_process_single_day [31]
@task
success

Figure 3: The Mobility Ingestion DAG utilizing Dynamic Task Mapping for parallel daily processing.

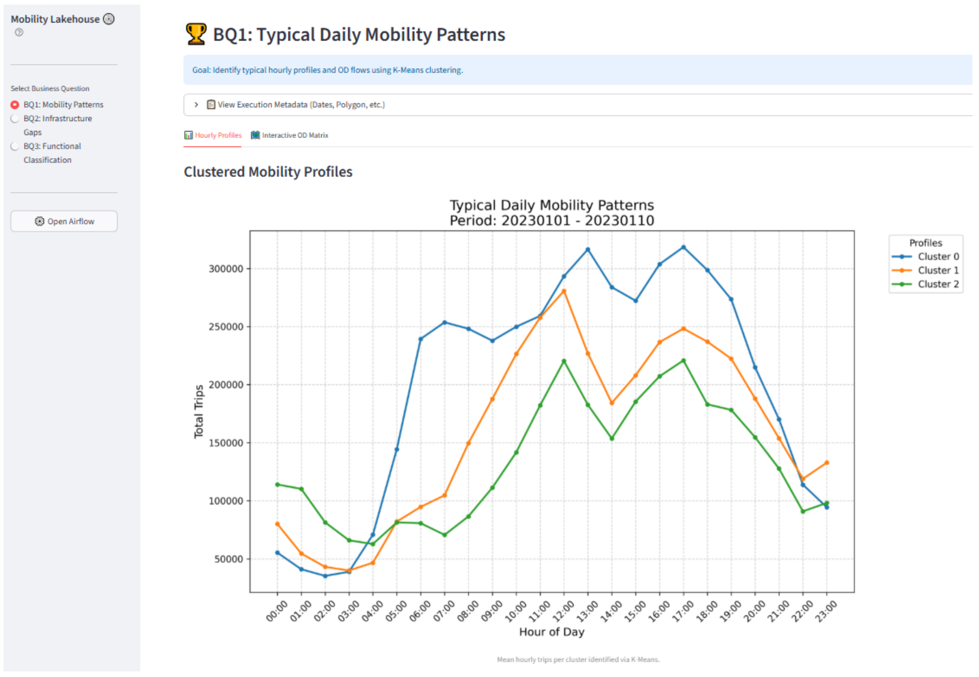
4.4 Analytical Generation and Reporting (DAGs 3)

The analytical layer is materialized through three dedicated pipelines. Each DAG addresses a specific business question (BQ) by transforming Silver data into Gold aggregates and generating visual reports.

DAG 31: Typical Day Clustering (BQ1)

This pipeline characterizes temporal mobility patterns via unsupervised learning.

- **Transformation:** Aggregates mobility data to extract hourly profiles and applies K-Means clustering ($k = 3$) to categorize days.
- **Output:** Generates a static Matplotlib chart for temporal profiles and an interactive Plotly HTML heatmap for OD analysis (Figure 4).



Mobility Lakehouse

Select Business Question

[✓ BQ1: Mobility Patterns]
BQ2: Infrastructure Gaps
BQ3: Functional Classification

Open Airflow

BQ1: Typical Daily Mobility Patterns

Goal: Identify typical hourly profiles and OD flows using K-Means clustering.

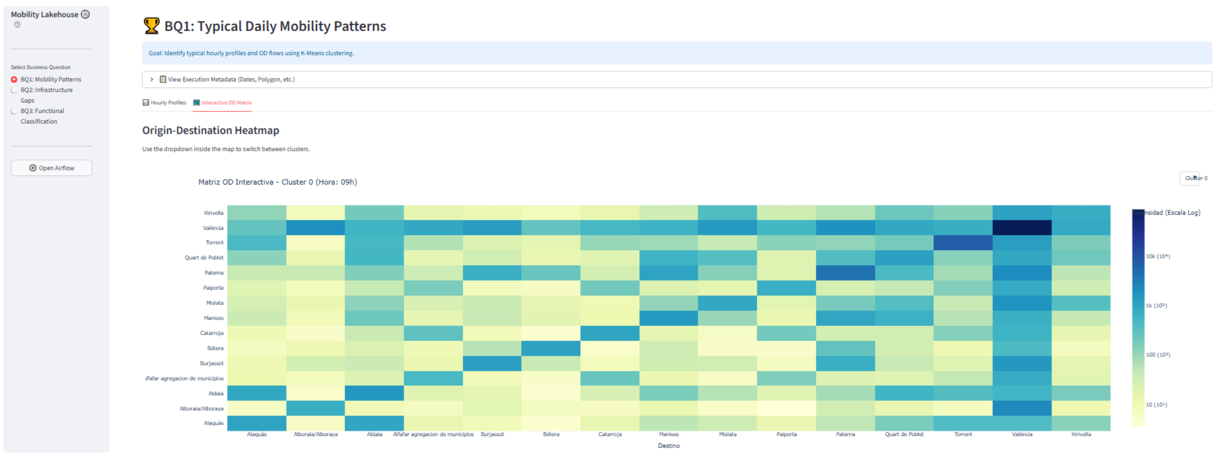
View Execution Metadata (Dates, Polygon, etc.)

Hourly Profiles | Interactive OD Matrix

Clustered Mobility Profiles

Mean hourly trips per cluster identified via K-Means.

(a) Temporal mobility profiles (Clustering results)



Mobility Lakehouse

Select Business Question

[✓ BQ1: Mobility Patterns]
BQ2: Infrastructure Gaps
BQ3: Functional Classification

Open Airflow

BQ1: Typical Daily Mobility Patterns

Goal: Identify typical hourly profiles and OD flows using K-Means clustering.

View Execution Metadata (Dates, Polygon, etc.)

Hourly Profiles | Interactive OD Matrix

Origin-Destination Heatmap

Use the dropdown inside the map to switch between clusters.

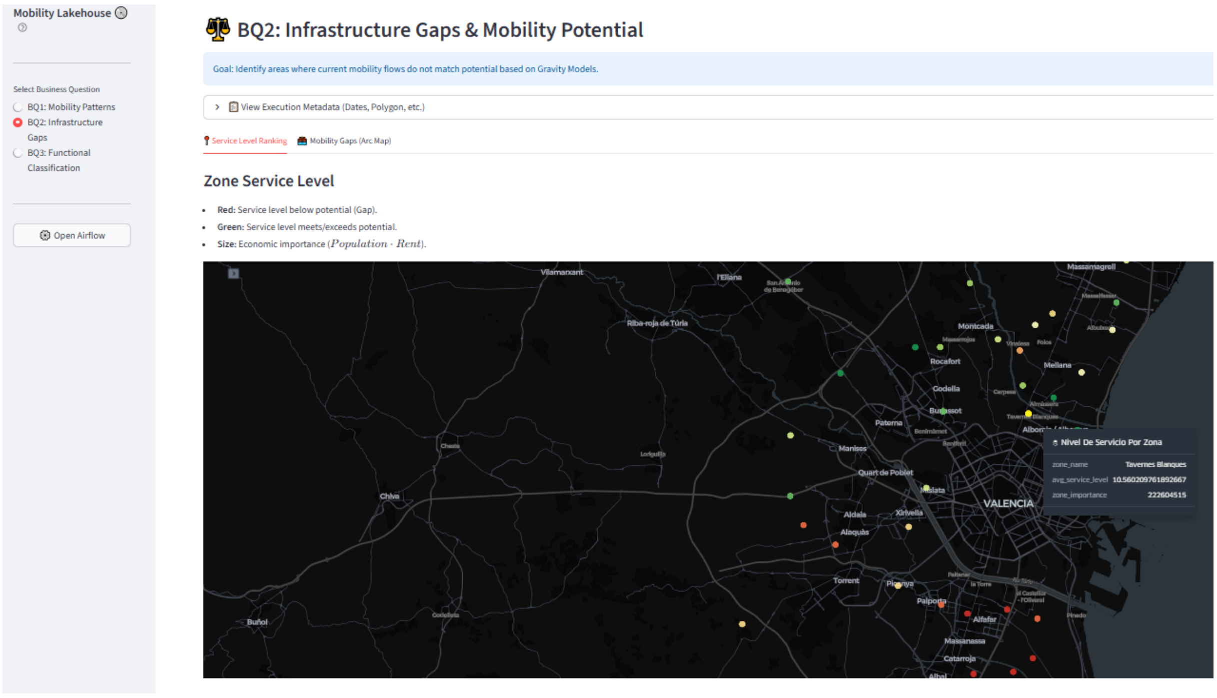
(b) Interactive Origin-Destination Heatmap

Figure 4: Visualization results for BQ1: Typical Daily Mobility Patterns.

DAG 32: Infrastructure Gap Analysis (BQ2)

This pipeline implements a Gravity Model to quantify the disparity between theoretical potential and actual mobility.

- **Transformation:** Integrates population, income and distance metrics to calculate a theoretical "Gravity Score" ($P \cdot E/d^2$). This is compared against observed flows to derive a `mismatch_ratio`.
- **Output:** Uses Kepler.gl to generate a *Service Level Ranking* and a *Mobility Gaps* arc map (Figure 5).



BQ2: Infrastructure Gaps & Mobility Potential

Goal: Identify areas where current mobility flows do not match potential based on Gravity Models.

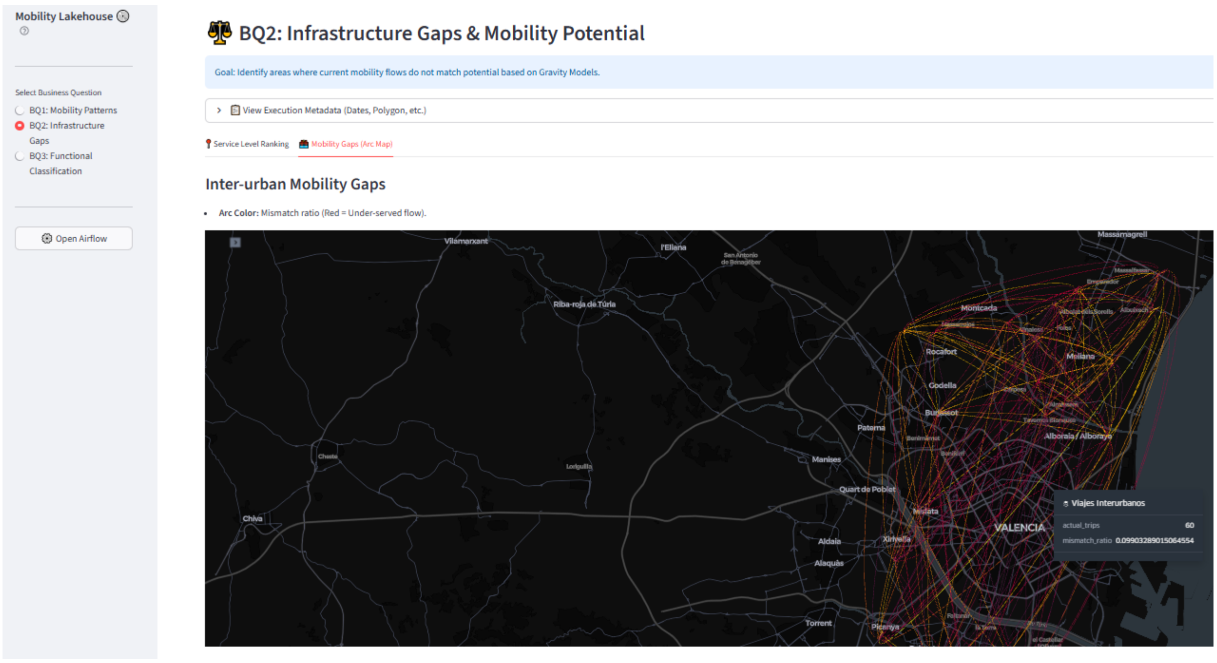
Zone Service Level

- **Red:** Service level below potential (Gap).
- **Green:** Service level meets/exceeds potential.
- **Size:** Economic importance ($Population \cdot Rent$).

Nivel De Servicio Por Zona

```
zone_name                  Tavernes Blanques
avg_service_level   10.560209761892667
zone_importance              222604515
```

(a) Zone Service Level Ranking



BQ2: Infrastructure Gaps & Mobility Potential

Goal: Identify areas where current mobility flows do not match potential based on Gravity Models.

Inter-urban Mobility Gaps

- **Arc Color:** Mismatch ratio (Red = Under-served flow).

Viajes Interurbanos

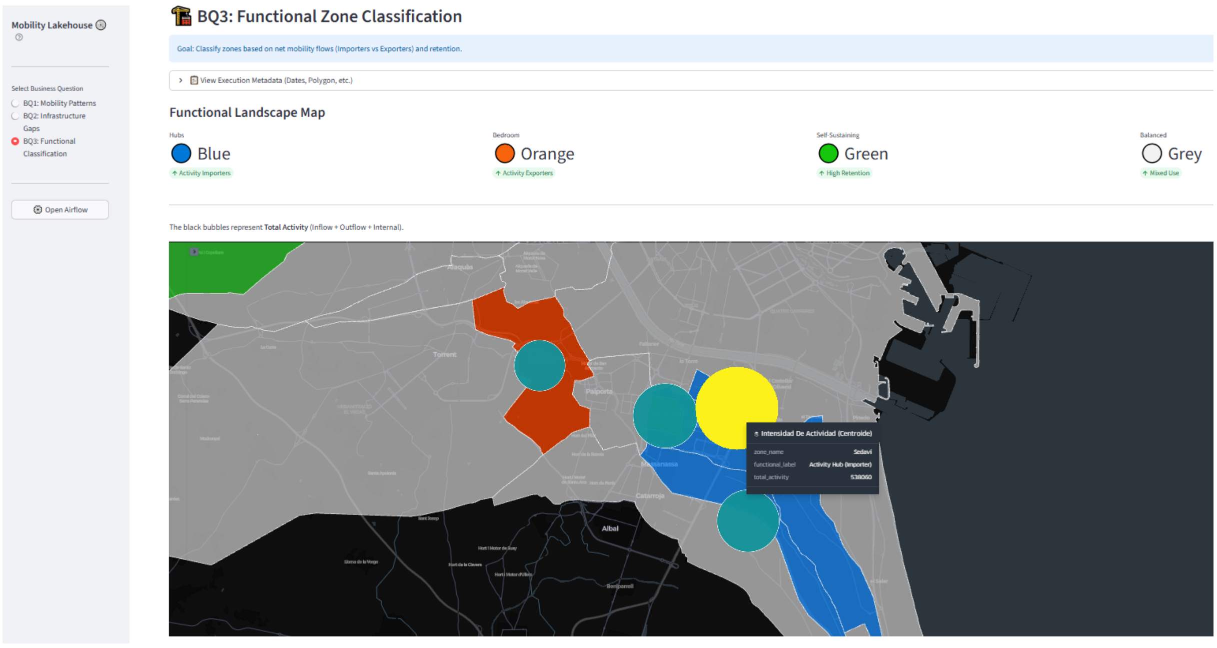
```
actual_trips                       60
mismatch_ratio  0.09903289015064554
```

(b) Inter-urban Mobility Gaps (Arc Map)

Figure 5: Visualization results for BQ2: Infrastructure Gaps and Mobility Potential.

DAG 33: Functional Classification (BQ3)
This pipeline categorizes municipalities based on their structural role in the metropolitan network.

- **Transformation:** Calculates flow metrics (retention rates, net flow ratios) and applies a deterministic decision tree to assign functional labels (e.g., "Bedroom Community").
- **Output:** Produces a Kepler.gl Choropleth map illustrating the region's functional hierarchy (Figure 6).



BQ3: Functional Zone Classification

Goal: Classify zones based on net mobility flows (Importers vs Exporters) and retention.

View Execution Metadata (Dates, Polygon, etc.)

Functional Landscape Map

Hubs	Bedroom	Self-Sustaining	Balanced
🔵 Blue	🟠 Orange	🟢 Green	⚪ Grey
↑ Activity Importers	↑ Activity Exporters	↑ High Retention	↑ Mixed Use

The black bubbles represent **Total Activity** (Inflow + Outflow + Internal).

Intensidad De Actividad (Centroide)

zone_name	Sedavi
functional_label	Activity Hub (Importer)
total_activity	538060

Figure 6: BQ3 Results: Functional Zone Classification based on net flows and retention rates.

5 Results and Use Cases

5.1 Use Case 1: Typical Mobility Patterns and Profiling

The first analytical objective focuses on the identification and characterization of canonical mobility behaviors within the reference year (2023). Traditional transport planning often relies on static calendar rules, assuming for instance, that all Mondays exhibit identical traffic pressure. To overcome this limitation, we implemented an unsupervised machine learning approach in the Gold layer to discover intrinsic daily patterns, distinguishing between standard workdays, holidays and transition days based on the geometry of their hourly flow curves.

Implementation Approach
The solution is orchestrated via the Airflow DAG `31_bq1_clustering`. The pipeline employs a hybrid processing strategy, leveraging DuckDB for high-volume aggregation and Python (Scikit-Learn) for in-memory clustering. The methodology follows a three-stage process:

Feature Extraction (Temporal Profiling): Granular records from `silver.fact_mobility` are first aggregated into daily hourly trip profiles by summing all Origin-Destination flows per day and hour. This temporal aggregation intentionally ignores spatial origin and destination to isolate the *shape* of daily demand curves. To ensure that clustering is driven by temporal structure (e.g., presence and timing of AM/PM peaks) rather than absolute volume, each daily profile is normalized as:

$$V_{d,h} = \frac{T_{d,h}}{\sum_{h=0}^{23} T_{d,h}} \tag{1}$$

where $T_{d,h}$ denotes the total number of trips observed on day d at hour h.

Unsupervised Classification (K-Means): The normalized daily vectors are clustered using the K-Means algorithm ($k = 3$), assigning each calendar date in the analysis period to a distinct behavioral class (e.g., standard workday, leisure-oriented day, or national holiday).

Feature Extraction (Spatial OD Profiling): After cluster labels are assigned, they are projected back onto the original, spatially-explicit mobility records. For each cluster and hour, trips are aggregated by origin and destination zone to compute representative Origin-Destination (OD) matrices. Unlike the temporal features, these spatial features preserve absolute trip magnitudes, as they are intended to support intensity-based spatial comparison rather than shape-based clustering.

Pattern Materialization: The resulting aggregates are persisted in the Gold layer as two complementary analytical tables: `gold.typical_day_patterns` (hourly temporal centroids per cluster) and `gold.typical_od_matrices` (cluster-specific OD matrices), enabling both temporal and spatial analytical products.

Visualization and Reporting

To facilitate validation by transport experts, the pipeline generates two automated reporting assets:

- **Temporal Profiles (Static):** A Matplotlib line chart plotting the centroids of the identified clusters. This visualizes the structural differences between profiles, such as the steep morning commute peak characteristic of workdays versus the bell-shaped curve typical of weekends.
- **Interactive OD Heatmaps:** A Plotly HTML report allowing analysts to filter by cluster and hour (e.g., "Cluster 0 at 08:00 AM"). This uses a logarithmic color scale to visualize the spatial distribution of trips, handling the high variance between urban cores and rural peripheries.

5.2 Use Case 2: Infrastructure Gap Analysis

The second objective addresses the detection of spatial inefficiencies where transport infrastructure fails to meet potential demand. To quantify this, we implemented a **Gravity Model** in the Gold layer. This approach models the theoretical interaction potential between municipalities based on their mass (population/economy) and impedance (distance) and compares it against the actual flows observed in the MITMA dataset.

Implementation Approach

The analysis is encapsulated in the `32_bq2_gaps` DAG and executes in three logical steps:

1. **Theoretical Modeling:** A "Gravity Score" is calculated for every Origin-Destination pair (i, j) using the classic interaction formula:

$$G_{ij} = k \cdot \frac{P_i \cdot E_j}{d_{ij}^2} \tag{2}$$

where P_i represents the population at the origin (supply), E_j represents the economic attractiveness at the destination (using net income per capita as a proxy) and d_{ij} is the Euclidean distance derived from `dim_zone_distances`.

2. **Dynamic Normalization (k-factor):** To render the theoretical score comparable to physical trip counts, the system calculates a global normalization factor k. This is derived by equating the total sum of observed trips in the network to the total sum of gravity scores.

3. **Gap Detection:** The critical metric, `mismatch_ratio`, is derived by dividing the actual observed trips by the normalized potential (G_{ij}). A ratio significantly below 1.0 identifies connections where mobility is suppressed relative to demographic pressure, signaling a potential infrastructure deficit.

Spatial Analytics and Visualization

The pipeline includes a spatial filtering mechanism (accepting WKT Polygons) to restrict analysis to specific regions (e.g., "Comunidad de Madrid"). Results are visualized via Kepler.gl:

- **Service Level Ranking:** A bubble map where zones are sized by theoretical importance ($P \times E$) and colored by their service level ratio.
- **Mobility Arcs:** A geospatial flow map highlighting specific links where the disparity between potential and actual traffic is most severe.

5.3 Use Case 3: Functional Zoning Classification

The third analytical product moves beyond raw traffic counting to characterize the *functional role* of each municipality within the metropolitan system. By analyzing the directionality and retention of flows, this use case automatically classifies zones into semantic categories (e.g., "Bedroom Communities" vs. "Activity Hubs").

Methodology and Classification Logic

The logic, encapsulated in DAG `33_bq3_functional_classification`, aggregates granular records into three flow types per zone: **Inflow** (entering), **Outflow** (leaving) and **Internal** (intra-zonal).

From these aggregates, two defining indices are calculated:

Net Flow Ratio (R_{net}): Measures the balance of trade in trips.

$$R_{\text{net}} = \frac{\text{Inflow} - \text{Outflow}}{\text{Inflow} + \text{Outflow}}$$

Retention Rate (R_{ret}): Measures the zone's self-sufficiency.

$$R_{\text{ret}} = \frac{\text{Internal}}{\text{Outflow} + \text{Internal}}$$

Based on these metrics, a deterministic decision tree assigns a `functional_label` to each municipality:

- **Self-Sustaining Cell ($R_{ret} > 0.20$):** Zones that satisfy a significant portion of their residents' needs locally.
- **Activity Hub / Importer ($R_{net} > 0$):** Zones with a positive flow balance, typically business districts attracting commuters.

- **Bedroom Community / Exporter ($R_{net} < 0$):** Zones with a negative flow balance, characterizing residential areas reliant on external employment centers.
- **Balanced / Transit Zone:** Areas with neutral flow ratios acting as connectors.

Visualization Strategy

The results are materialized in a Kepler.gl dashboard combining two visual layers:

- **Choropleth Layer:** Displays administrative boundaries colored by their `functional_label`, revealing regional clusters (e.g., residential rings surrounding central hubs).
- **Centroid Layer:** Displays bubbles sized by `total_activity` (Sum of In + Out + Internal), distinguishing major metropolitan centers from smaller rural nodes regardless of their functional type.

6 Discussion

The implementation of this 3-tier Data Lakehouse represents a paradigm shift from traditional, file-based workflows in mobility analysis. By transitioning from ad-hoc processing of disparate CSV files to a structured, transactional Lakehouse architecture, the system successfully bridges the gap between raw data availability and actionable urban planning insights.

The results of the implementation validate the architecture's ability to achieve three critical operational milestones:

- **Operational Democratization and Accessibility:** The architecture effectively decouples the complex data engineering lifecycle (ELT) from analytical consumption. Transport experts, who previously faced significant technical barriers when parsing terabytes of raw daily records, can now interact with the Gold layer using standard SQL queries or Business Intelligence (BI) tools. By pre-computing complex derivatives—such as *gravity scores* and *functional zoning labels*—the system reduces analytical latency from days to seconds, enabling rapid hypothesis testing.
- **Cost-Effective Scalability:** The architectural decision to decouple storage (Amazon S3) from compute (DuckDB/AWS Batch) ensures the system can retain years of historical data with negligible holding costs. Unlike tightly coupled Data Warehouses where infrastructure costs accrue continuously, this serverless approach incurs compute costs only during active transformation or querying. This empirically validates that high-performance analytics on national-scale datasets is achievable within a constrained academic or municipal budget.
- **High-Dimensional Contextualization:** The primary analytical value is derived from the integration logic within the Silver layer. By enriching raw mobility flows with socio-economic indicators (INE income/demographics) and geospatial impedance metrics, the system transforms simple trip counts into sophisticated accessibility indicators. As demonstrated in Use Case 2, this fusion reveals infrastructure deficits that would remain invisible if mobility matrices were analyzed in isolation.

Ultimately, this platform demonstrates that open-source tools (DuckDB, Apache Airflow) can replicate the capabilities of enterprise-grade big data systems. It provides a reproducible blueprint for public administrations to internalize the processing of mobility data, reducing reliance on third-party aggregators and ensuring data sovereignty.

7 Conclusions and Limitations

7.1 Conclusions

This project presents the design and successful deployment of a robust Data Lakehouse architecture capable of ingesting, cleaning and analyzing national-scale mobility data. The implementation confirms that modern open-source technologies can effectively democratize access to big data in the urban planning domain. The key conclusions drawn from this work are:

- **Architectural Viability:** The 3-tier Medallion architecture (Bronze, Silver, Gold) proved essential for data governance. By decoupling storage (S3) from compute (AWS Batch), the system achieved a high degree of cost-efficiency and reproducibility, validating the "Lakehouse" paradigm as a superior alternative to monolithic Data Warehouses for intermittent analytical workloads.
- **Analytical Versatility:** The deployment of three distinct analytical pipelines demonstrated the platform's flexibility. The system successfully supported diverse computational patterns, ranging from unsupervised machine learning (Clustering for temporal profiling) to physics-based simulations (Gravity Models) and semantic geospatial classification.
- **Technology Stack Validation:** The integration of **DuckDB** and **DuckLake** provided a performant, ACID-compliant SQL engine without the overhead of enterprise clusters. This confirms that single-node, vectorized processing is sufficient for handling billions of records when leveraging modern columnar formats (Parquet).

7.2 Limitations

While the proposed architecture meets the primary research objectives, several limitations were identified regarding external dependencies and infrastructure constraints. These challenges delineate the boundaries of the current system and suggest areas for future optimization.

- **Source Data Continuity:** The reliability of the system is strictly bound by the upstream provider. As documented in the MITMA technical specifications, the source datasets exhibit significant discontinuities during Q4 2023 due to mobile network signaling incidents. Although the pipeline's ingestion logic successfully handles missing files without failure, downstream analytics for this period require statistical imputation to be statistically representative.
- **Metadata Catalog Constraints (Neon):** A critical bottleneck was identified in the interaction between the compute engine and the metadata catalog. The implementation utilized the generic Free Tier of Neon (Serverless Postgres) to manage DuckLake transactions. It was observed that the strict idle-connection timeouts and auto-suspend policies of the serverless tier are incompatible with long-running bulk ingestion tasks. This necessitated the artificial fragmentation of historical backfills into smaller temporal windows to ensure transaction stability, increasing total processing time.
- **Vertical Scalability Ceiling:** A fundamental structural limitation of DuckDB is its single-node architecture. Unlike distributed systems (e.g., Apache Spark) which scale horizontally by adding nodes to a cluster, DuckDB relies on vertical scaling (Scale-Up). While AWS Batch allows for the provisioning of high-memory instances (e.g., 128GB RAM), the system has a theoretical hard limit. Analytical queries requiring memory exceeding the capacity of the largest available single EC2 instance will degrade significantly due to disk spilling or fail entirely.
- **Latency and Real-Time Applicability:** The architecture is intrinsically designed for $T + 1$ (Daily) batch processing. Due to the inherent latency in provisioning ephemeral AWS

Batch containers and the file-based delivery mechanism of the source data, this platform is not suitable for real-time monitoring. Use cases requiring sub-minute latency—such as immediate accident detection or live traffic signal optimization—would necessitate a Streaming architecture (e.g., Apache Kafka/Flink) rather than the current batch-oriented Lakehouse approach.

References

[1] Spanish Ministry of Transport (MITMA). *Open Data Mobility*. Available at: https://www.transportes.gob.es/ministerio/proyectos-singulares/estudios-de-movilidad-con-big-data/opendata-movilidad (Accessed on 4 January 2026).

[2] Spanish National Statistics Institute (INE). Available at: https://www.ine.es/ (Accessed on 4 January 2026).

[3] Centro Nacional de Información Geográfica (CNIG). Available at: https://www.ign.es/ (Accessed on 4 January 2026).

[4] Ayuntamiento de Madrid. *Portal de Datos Abiertos*. Available at: https://datos.madrid.es/ (Accessed on 4 January 2026).

[5] DuckDB Documentation. Available at: https://duckdb.org/ (Accessed on 4 January 2026).

[6] Amazon Web Service Batch. *AWS Batch Documentation*. Available at: https://aws.amazon.com/es/batch/ (Accessed on 4 January 2026).

[7] DuckLake. *DuckLake Documentation*. Available at: https://ducklake.select/docs/stable/ (Accessed on 4 January 2026).

[8] Apache Software Foundation. *Apache Airflow Documentation*. Available at: https://airflow.apache.org/ (Accessed on 4 January 2026).