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

Big Data Engineering Project

Joan Sánchez Verdú **María López Hernández**
Fernando Blanco Membrives
Link to Code Repository:
https://github.com/joanstudyai-code/MUCEIM-BDET-Project.git

December 18, 2025

Abstract

Abstract. This paper presents the design and implementation of a scalable 3-tier Data Lakehouse architecture tailored for the analysis of large-scale mobility data in Spain. Leveraging open data from the Spanish Ministry of Transport (MITMA) and the National Statistics Institute (INE), the system integrates high-volume Origin-Destination (OD) matrices with demographic and economic indicators. The infrastructure utilizes DuckDB for efficient analytical processing and DuckLake for ACID-compliant storage, orchestrated via Apache Airflow. We demonstrate the system's utility through two key use cases: characterizing typical daily mobility patterns and identifying transport infrastructure gaps using gravity models. The result is a robust, cost-effective platform enabling transport experts to derive data-driven insights for urban planning.

1 Introduction

Mobility analysis is critical for modern urban planning...

1.1 Objectives

The primary objective of this work was to construct a robust data infrastructure...

- Build a scalable Lakehouse for mobility and economic data.
- Implement ELT processes across Bronze, Silver, and Gold tiers.
- Demonstrate analytical value through specific mobility use cases.

2 Data Sources

The data lakehouse architecture is designed to ingest and integrate public domain data exclusively, ensuring reproducibility and open access. The following subsections detail the specific datasets and sources selected to build the information system.

2.1 MITMA Open Data

The core mobility dataset is sourced 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 the daily movements of residents derived from mobile network big data. From the MITMA the following files have been used:

- ***(202301-202312)_ Viajes_ municipios.tar*:** These files, located in the MITMA web at *estudios_ basicos/por-municipios/viajes/meses-completos/*, contain the information of the mobility from an origin to a destination municipality, per day, per hour, for the year 2023.
- ***nombres_ municipios.csv*:** This file, located in the MITMA web at *zonificacion/zonificacion_ municipios/*, contains the name of each municipality.
- ***relacion_ ine_ zonificacionMitma.csv*:** This file, located in the MITMA web at *zonificacion/*, cointains the mapping between the MITMA codes and the INE codes for the municipalities.
- ***poblacion.csv*:** This file, located in the MITMA web at *zonificacion/*, cointains the population of each municipality per year.

2.2 INE Demographics and Economics

To contextualize mobility flows and enable the gravity model analysis, mobility data is enriched with official statistics from the Spanish National Statistics Institute (INE [2]). The statistics used from INE are the following:

- ***ine_ rent_ municipalities.csv*:** This file, located in the INE web at `https://www.ine.es/jaxiT3/files/t/es/csv_bd/`, contains the rent information per year and municipality.

2.3 CNIG and Geographic Data

To perform spatial analysis (such as calculating distances between zones for the gravity model), the Lakehouse requires geospatial vector data. This data is sourced effectively from the National Center for Geographic Information (CNIG [3]).

- ***pyspainmobility* (Python Library):** This package serves as the extraction interface for the official municipal boundaries. Specifically, the `Zones(zones="municipios", version=2)` module is used to fetch the geometry of Spanish municipalities.

2.4 Open Data

To categorize mobility patterns by day type (e.g., distinguishing standard workdays from holidays), specific calendar data is integrated from Open Data portals.

- ***calendario_ laboral.csv*:** This file, sourced from the Madrid City Council Open Data Portal (*Datos Abiertos Madrid* [4]) at `https://datos.madrid.es/`, contains the official working calendar. It provides the classification of dates (working days, weekends, and holidays) from 2013 to 2026 (included).

3 Lakehouse Architecture

The system follows the Medallion Architecture pattern, decoupled into storage and compute layers to ensure scalability.

3.1 Technology Stack

The platform is built upon a modular, open-source stack designed to decouple compute from storage, allowing for independent scaling and cost efficiency. The architecture leverages the following core components:

- **Compute Engine (DuckDB [5]):** Selected for its high-performance, vectorized execution engine optimized for OLAP workloads. Unlike distributed systems (e.g., Apache Spark), DuckDB runs in-process, significantly reducing infrastructure overhead while maintaining the ability to process datasets larger than RAM efficiently.
- **Lakehouse Management (DuckLake [6]):** A specialized extension that brings Data Lakehouse capabilities to the ecosystem. It provides ACID transaction support (Atomicity, Consistency, Isolation, Durability) and snapshot isolation. This ensures that long-running ingestion tasks do not block analytical queries and prevents data corruption during partial write failures.
- **Storage - Cloud Infrastructure (Neon + S3):** The system adopts a cloud-native architecture that physically separates data from metadata:
  - **Data Plane (AWS S3):** Stores the actual physical files (Parquet and CSV). S3 provides durability and scalable object storage for the Bronze, Silver, and Gold layers.
  - **Control Plane (Neon Postgres):** A serverless PostgreSQL instance acts as the centralized catalog and metadata store. It manages table definitions, schemas, and transaction locking, enabling safe concurrent writes to S3 from multiple Airflow workers.
- **Orchestration (Apache Airflow [7]):** Manages the end-to-end lifecycle of data pipelines. Airflow allows for the definition of Directed Acyclic Graphs (DAGs) to handle complex task dependencies, backfilling strategies for historical data, and granular retries in case of network or source failures.

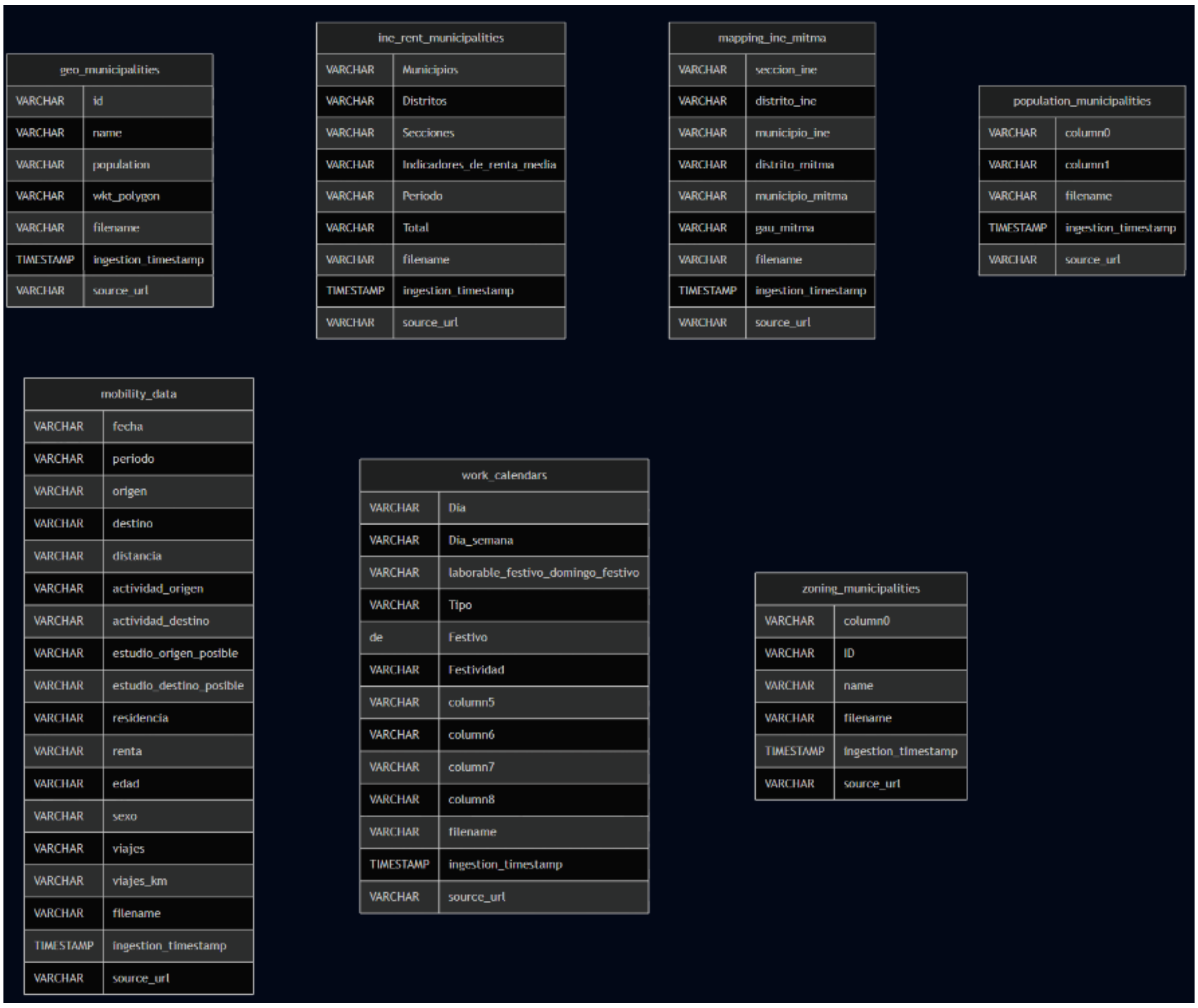
3.2 Data Layering Strategy

The data pipeline is structured following the standard "Medallion" architecture pattern (Multi-hop), which organizes data quality into progressive stages. This design ensures that raw data is preserved for auditability while downstream layers provide cleaned and enriched datasets optimized for analytical consumption. The architecture consists of three distinct zones, each with a specific purpose and schema design.

3.2.1 Bronze Layer (Raw)

The Bronze layer acts as the landing zone (Staging Area) for all external data. Its primary function is to capture data from source systems in its native format with minimal modification, ensuring an immutable record of the original input.

In this implementation (Figure 1), a distinct table is instantiated for each source entity to maintain data isolation. To ensure pipeline robustness and prevent ingestion failures due to type mismatches, all columns are initially cast as `VARCHAR`. Additionally, the ingestion process is configured to ignore malformed records (e.g., data quality anomalies such as invalid dates like '20231035'), ensuring that isolated errors do not halt the entire pipeline. Furthermore, to enhance data governance and auditability, three metadata columns are appended to every record during ingestion: `ingestion_timestamp`, `filename`, and `source_url`. Finally, to optimize storage and performance, the high-volume mobility data is physically partitioned by date (`fecha`), enabling efficient batch processing and parallelized reads in downstream layers.



geo_municipalities	
VARCHAR	id
VARCHAR	name
VARCHAR	population
VARCHAR	wkt_polygon
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

ine_rent_municipalities	
VARCHAR	Municipios
VARCHAR	Distritos
VARCHAR	Secciones
VARCHAR	Indicadores_de_renta_media
VARCHAR	Periodo
VARCHAR	Total
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

mapping_ine_mitma	
VARCHAR	seccion_ine
VARCHAR	distrito_ine
VARCHAR	municipio_ine
VARCHAR	distrito_mitma
VARCHAR	municipio_mitma
VARCHAR	gau_mitma
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

population_municipalities	
VARCHAR	column0
VARCHAR	column1
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

mobility_data	
VARCHAR	fecha
VARCHAR	periodo
VARCHAR	origen
VARCHAR	destino
VARCHAR	distancia
VARCHAR	actividad_origen
VARCHAR	actividad_destino
VARCHAR	estudio_origen_posible
VARCHAR	estudio_destino_posible
VARCHAR	residencia
VARCHAR	renta
VARCHAR	edad
VARCHAR	sexo
VARCHAR	viajes
VARCHAR	viajes_km
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

work_calendars	
VARCHAR	Dia
VARCHAR	Dia_semana
VARCHAR	laborable_festivo_domingo_festivo
VARCHAR	Tipo
de	Festivo
VARCHAR	Festividad
VARCHAR	column5
VARCHAR	column6
VARCHAR	column7
VARCHAR	column8
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

zoning_municipalities	
VARCHAR	column0
VARCHAR	ID
VARCHAR	name
VARCHAR	filename
TIMESTAMP	ingestion_timestamp
VARCHAR	source_url

Figure 1: Entity Diagram of the Bronze Layer. One table for each file.

3.2.2 Silver Layer (Cleaned & Integrated)

The Silver layer represents the refined, enterprise-wide view of the data. In this stage, data undergoes validation, cleaning, standardization, and enrichment processes to transform raw inputs into structured, trustworthy tables. The schema design (Figure 2) follows a Star Schema approach, establishing a central fact table for mobility flows linked to surrounding dimension tables for spatial and socio-economic context. All tables include a `TIMESTAMP WITH TIME ZONE` column `processed_at` for auditability.

Dimension Modeling and Spatial Integration (dim_zones)

The foundation of the Silver layer is the creation of a unified spatial dimension, `dim_zones`. This table integrates the descriptive zoning metadata (MITMA codes, INE codes, and municipality names) with the geospatial vector data ingested in the Bronze layer.

- **Geometry Reconstruction:** The Well-Known Text (WKT) strings stored in the Bronze layer are parsed into native `Geometry` objects using DuckDB's `ST_GeomFromText` function from the `spatial` module.
- **Surrogate Keys:** A unique integer identifier (`zone_id`) is generated for each municipality. This decouples downstream analytics from changes in external string codes (MITMA/INE IDs) and improves join performance.

Socio-Economic Metrics and Filtering

Complementary metric tables are generated by cleaning raw CSVs. For the `metric_population`, data is cast to `BIGINT` while removing metadata headers.

For the `metric_ine_rent` table, specific business logic is applied to the INE source file to resolve granularity mismatches. The pipeline filters the dataset to isolate the "Average Net Income per Person" indicator and explicitly excludes district-level or census-section entries (filtering out rows where `Distritos` or `Secciones` are populated). This ensures that the resulting economic metrics align perfectly with the municipal granularity of the `dim_zones` table.

Temporal Context (Holidays)

To support temporal analysis, a `dim_zone_holidays` table is constructed. This process involves filtering the raw working calendar for events classified as "National Holidays" and cross-referencing them with the zones. This dimension allows the Gold layer to accurately distinguish mobility patterns between standard working days and holidays.

Mobility Fact Table Construction

The core mobility data is transformed into the `fact_mobility` table. This process converts the raw, text-based daily CSVs into a strongly typed, time-series optimized format. Key transformations include:

1. **Temporal Standardization:** The raw separate date and hour fields are combined into a single `TIMESTAMP WITH TIME ZONE` column, explicitly localized to 'Europe/Madrid'.
2. **Spatial Lookup:** Origin and Destination codes are replaced by their corresponding `zone_id` through double joins against the `dim_zones` table, ensuring referential integrity.
3. **Type Casting:** The "trips" column is cleaned (handling European decimal formats) and cast to `DOUBLE` precision to support fractional trip expansion factors.

Data Observability and Quality Logging

To ensure the reliability of the pipeline, a dedicated table named `data_quality_logs` has been implemented within the Silver layer. Unlike the business data tables, this table serves as a metadata registry to persist the results of automated audit checks performed during ingestion (e.g., null rate calculations, row count validations, or statistical anomalies). This allows for longitudinal tracking of data health.

The schema is designed to be generic enough to store heterogeneous metrics:

- **`check_timestamp`:** The exact time when the audit was executed.
- **`table_name`:** The target table being audited (e.g., *fact_ mobility*).
- **`metric_name`:** A descriptive identifier for the KPI (e.g., *avg_ income_ per_ capita*, *null_ zone_ rate*).
- **`metric_value`:** A numeric field (`DOUBLE`) storing the result of the check.
- **`notes`:** Textual field for context, such as the specific batch date range or error warnings associated with the metric.

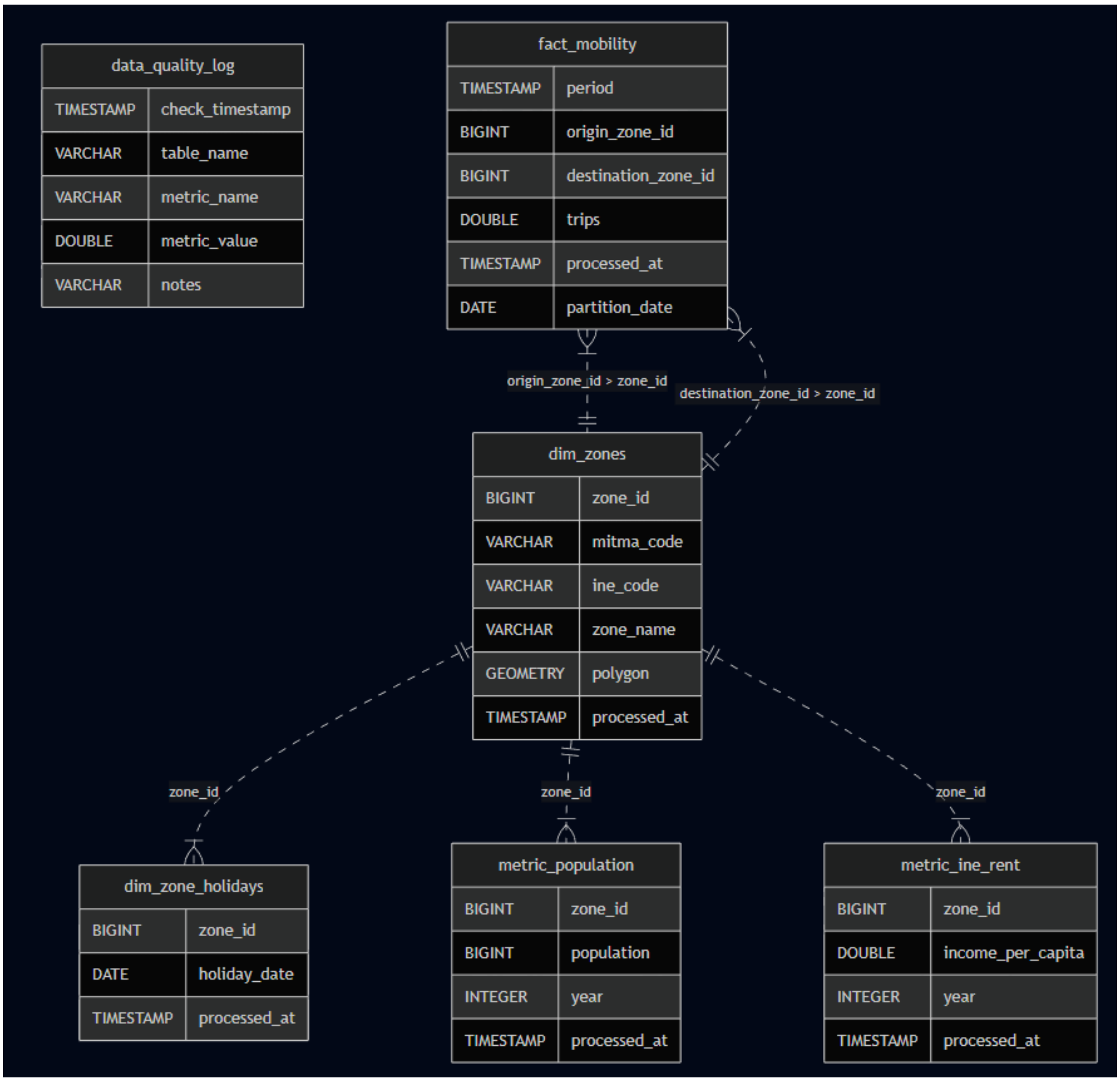


Figure 2: Entity-Relationship Diagram (ERD) of the Silver Layer. The schema follows a Star Schema design, centering on the `fact_mobility` table linked to spatial (`dim_zones`) and temporal dimensions.

3.2.3 Gold Layer (Mart)

[TODO]

4 Implementation Methodology

The operational logic of the Data Lakehouse is implemented through a unified, automated workflow orchestrated by Apache Airflow [7]. 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 and modular task isolation.

4.1 Orchestration Strategy

To manage the dependencies between the Bronze, Silver, and Gold layers, the system utilizes a monolithic Directed Acyclic Graph (DAG) named `mobility_unified_pipeline`. This pipeline is architected using Airflow's TaskFlow API, allowing for the Pythonic definition of dependencies and data passing.

As illustrated in Figure 3, the workflow is designed as a linear dependency chain of grouped tasks, ensuring that foundational data (Dimensions) is fully consistent before high-volume processing (Facts) begins.

Figure 3: Visual representation of the Airflow DAG. The workflow progresses from static infrastructure setup (left) to parallelized daily processing (center), culminating in analytical modeling (right).

The specific responsibilities of the key tasks depicted in the diagram are as follows:

- `create_schemas`: Initializes the DuckLake schemas (*bronze, silver, gold*) and the persistence layer for data quality logs.
- `ingest_[geo|static]`: A parallel group of tasks that extract reference data (Shapefiles, INE CSVs) from external web sources.
- `build_silver_dimensions`: Performs SQL transformations to clean reference data and generate surrogate keys (e.g., `dim_zones`).
- `ensure_fact_tables_exist`: A singleton task that prepares the destination tables for the mobility data, preventing race conditions during parallel writes.
- `process_single_day`: The dynamic mapped task. Each instance handles the full ELT cycle (Download → Bronze → Silver) for a specific date within a transaction block.
- `audit_[dims|batch]`: Quality control gates that calculate metrics (e.g., null rates) and log them to the metadata registry.
- `create_gold_[cluster|gaps]`: The final analytical steps that aggregate the fully processed Silver data into business-ready insights.

4.2 Parallelized Ingestion and Processing

A critical requirement of the project was handling the high cardinality of daily mobility files (one file per day per year). Instead of a sequential loop, the implementation leverages Airflow's Dynamic Task Mapping.

The `process_single_day` task is designed as an atomic worker. The DAG dynamically generates a list of dates based on the input parameters (`start_date`, `end_date`) and expands the worker task across this list using the `.expand()` method. This allows the system to scale horizontally; identifying 365 days results in 365 mapped task instances. Concurrency is managed via the `max_active_tis_per_dag` parameter (configured to 8 in the cloud environment) to optimize throughput without overwhelming the MITMA source servers or the DuckDB memory limits.

4.3 Transaction Management and ACID Compliance

To adhere to the Robustness objective, the pipeline implements strict transaction controls within the DuckDB connection context. Given the potential for network failures during the streaming of large compressed CSVs, avoiding partial data writes is paramount.

The implementation wraps the Bronze ingestion and Silver transformation logic within a single database transaction block:

- **Atomicity:** The worker task explicitly initiates a transaction using `con.begin()`. Both the Bronze insertion (raw data) and Silver transformation (fact table loading) must succeed together.
- **Rollback Mechanism:** The logic is encapsulated in a `try...except` block. If any error occurs (e.g., a connection timeout or parsing error), `con.rollback()` is triggered immediately. This ensures that a failed run leaves no ghost data in the Silver layer, maintaining the lakehouse in a consistent state.

4.4 Automated Analytical Modeling (Gold Layer)

Unlike standard ETL pipelines that stop at data cleaning, this implementation integrates advanced analytics directly into the workflow. Once the batch processing is complete and validated via the `audit_batch_results` task, the Gold layer construction begins.

This phase is implemented via two distinct tasks:

- **Clustering Analysis:** The `create_gold_clustering` task extracts normalized hourly profiles from the Silver layer into memory. It utilizes the *Scikit-learn* library to perform K-Means clustering, determining typical mobility patterns (e.g., Weekday vs. Weekend). The results are materialized back into DuckDB tables for reporting.
- **Gap Analysis:** The `create_gold_gaps` task performs a full-scan aggregation of the processed period, joining mobility flows with socio-economic metrics to calculate the Gravity Model mismatch ratios.

4.5 Decoupled Data Serving

While the unified pipeline handles the heavy lifting of construction and modeling, the architecture is designed to support a decoupled reporting pattern for data consumption.

[TODO]

5 Results and Use Cases

5.1 Use Case 1: Typical Mobility Patterns

To characterize the "typical day," we aggregated hourly flows and applied K-Means clustering...

Figure 4: Hourly mobility profiles identified (Weekdays vs. Weekends).

5.2 Use Case 2: Infrastructure Gap Analysis

We compared actual MITMA demand against theoretical demand derived from a gravity model:

$$T_{ij} = k \cdot \frac{P_i \cdot E_j}{d_{ij}^2} \tag{1}$$

The mismatch ratio calculated in the Gold layer highlights zones where actual transport usage significantly lags behind potential demand...

6 Discussion

The implemented lakehouse allows experts to query billions of rows using standard SQL without managing infrastructure...

7 Conclusions and Limitations

The project successfully established a scalable 3-tier architecture. However, limitations exist regarding data anomalies in late 2023...

References

[1] Spanish Ministry of Transport (MITMA). Open Data Mobility. `https://www.transportes.gob.es/ministerio/proyectos-singulares/estudios-de-movilidad-con-big-data/opendata-movilidad`

[2] Spanish National Statistics Institute (INE). `https://www.ine.es/`

[3] Centro Nacional de Información Geográfica (CNIG). `https://www.ign.es/`

[4] Ayuntamiento de Madrid. Portal de Datos Abiertos. `https://datos.madrid.es/`

[5] DuckDB Documentation. `https://duckdb.org/`

[6] DuckLake. *DuckLake Documentation.* `https://ducklake.select/docs/stable/`

[7] Apache Software Foundation. *Apache Airflow Documentation.* `https://airflow.apache.org/`