Lab 3 Report: Spark-Based Data Lake

Group L3-T04

Marvin Ernst, Oriol Gelabert, Alex Malo

June 24, 2025

Overview of Pipeline Architecture

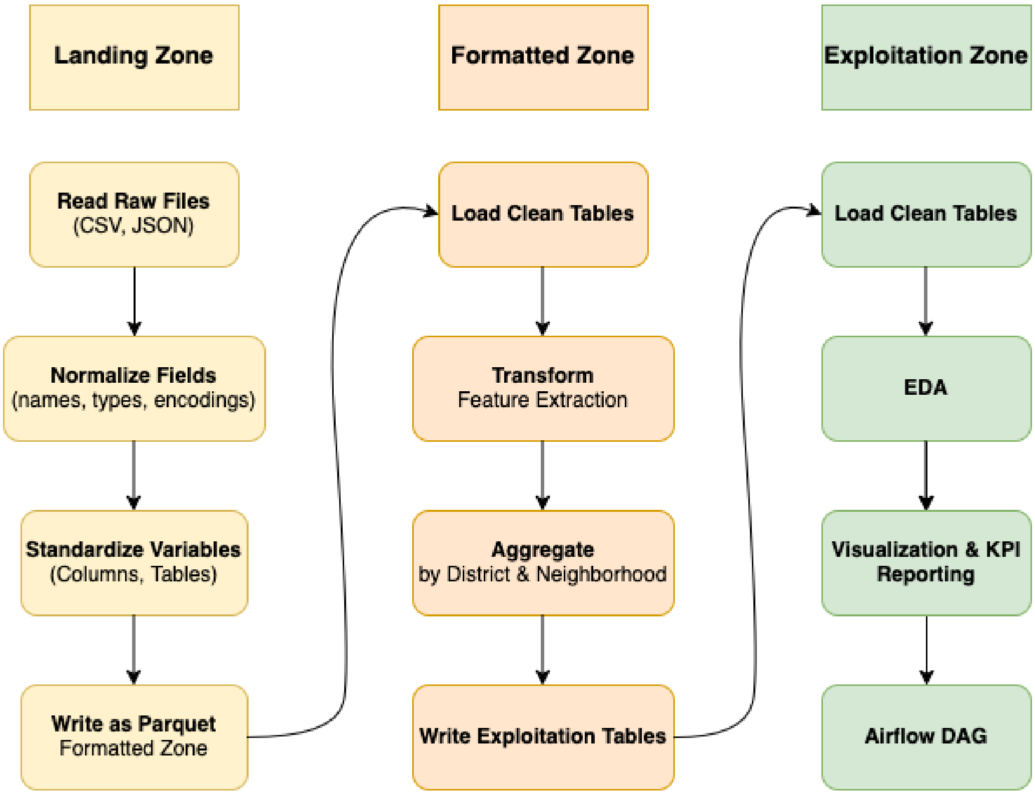
Landing Zone
Formatted Zone
Exploitation Zone
Read Raw Files (CSV, JSON)
Normalize Fields (names, types, encodings)
Standardize Variables (Columns, Tables)
Write as Parquet Formatted Zone
Load Clean Tables
Transform Feature Extraction
Aggregate by District & Neighborhood
Write Exploitation Tables
Load Clean Tables
EDA
Visualization & KPI Reporting
Airflow DAG

Figure 1: High-level abstraction of the three main pipelines

- **Pipeline 1: Data Formatting** (Raw → Cleaned Parquet)
  - Input: CSV/JSON files from `landing_zone/`
  - Tasks: unzip, harmonize schema, type casting, renaming, spatial normalization
  - Output: Cleaned and harmonized Parquet files in `formatted_zone/`
- **Pipeline 2: Exploitation** (Cleaned → Aggregated)
  - Input: Parquet files from `formatted_zone/`
  - Tasks: aggregations at multiple spatial levels (district, neighborhood, census section), KPI computation
  - Output: Indicator datasets in `exploitation_zone/`

- **Pipeline 3: Analysis** (Aggregated → Dashboards)
  - Input: Aggregated indicator files
  - Tasks: descriptive analysis and plots using Seaborn and Matplotlib
  - Output: Visual dashboards to reveal trends and spatial patterns

Selected Datasets and Assumptions

We selected the following three datasets, all located in `landing_zone/`:

1. **Tourist Housing**: quarterly CSV files with apartment registration data
2. **Commercial Premises**: yearly CSV files on ground-floor businesses
3. **Household Size**: JSON files with household structure (number of people per dwelling)

Key assumptions made:

- **District/Neighborhood Mapping**: Many datasets lacked district and neighborhood codes but included names. We extracted reliable mappings from the 2023 tourist housing files and built mapping dictionaries.
- **Harmonization**: Yearly and quarterly datasets had inconsistent schemas. We renamed and typecast all relevant columns for cross-year integration.
- **Boolean Fields**: Strings like `"Sí"`/`"No"` or other variations were normalized to binary indicators for commercial premises.
- **Date Inference**: Registration dates were inferred from string IDs (e.g., `N_EXPEDIENT` in housing).
- **Missing Data**: Null values were preserved unless necessary to replace (e.g., for joins or aggregations).

Pipeline Specific Design Justifications

Data Formatting Pipeline

- Used regex and encoding detection to process CSV and JSON files robustly
- Applied batch extraction, delimiter correction, and header normalization to ensure Spark compatibility
- Used UDFs to normalize accent marks and string cases for consistent joins
- Final outputs stored in columnar Parquet format for downstream efficiency

Exploitation Pipeline

- Aggregated indicators by district, neighborhood, and section to support flexible spatial analysis
- Computed KPIs like share of coworking/nightlife premises, average household size, and total tourist licenses
- Calculated relative proportions (% of premises or households) instead of raw counts to allow comparability
- Stored all outputs in `exploitation_zone/` with clear file naming convention for traceability

Analysis Pipeline

- Combined Spark (ETL) with Pandas for final summarization and plotting (performance trade-off)
- Produced plots for:
  - Household size distributions
  - Commercial indicators by district
  - Tourist housing trends over time
  - Correlation between tourist activity and commercial premises
- Constructed a KPI summary table merging all indicators

Airflow Orchestration

To ensure reproducibility and automation, we implemented a DAG in Apache Airflow that sequentially runs:

1. `01_data_formatting_pipeline.py`
2. `02_exploitation_pipeline.py`
3. `03_analysis_pipeline.py`

Each script is executable and reads paths from environment variables to avoid hardcoding. Airflow provides a centralized orchestration mechanism ensuring repeatable workflows.