Traffic Accidents ETL Documentation

Version: 2.0

Last Updated: March 2025

Project Repository: GitHub - Traffic Accidents ETL

Authors:

- Isabella Pérez C. - 2230603
- Luz Angela Carabalí M. - 2230652
- Nicolás Peña Irurita - 2232049

1. Introduction

The **Traffic Accidents ETL** project is a data pipeline solution designed to extract, transform, and load (ETL) traffic accident data for analysis and visualization. Initially developed as a Python-based ETL pipeline, the project has been enhanced in its second iteration to incorporate **Apache Airflow** for workflow orchestration, improving scalability, scheduling, and monitoring capabilities. This documentation provides a comprehensive overview of the updated solution, including its architecture, components, installation instructions, and usage guidelines.

The project leverages open data sources, such as a CSV dataset from Kaggle (migrated to a PostgreSQL database) and geospatial data from OpenStreetMap, to enrich traffic accident datasets with location-based features. This enables advanced analysis of accident patterns, infrastructure correlations, and risk zones. The final output is a dimensional model stored in a database, which is visualized through an interactive dashboard built with Power BI for actionable insights.

---

2. Objectives

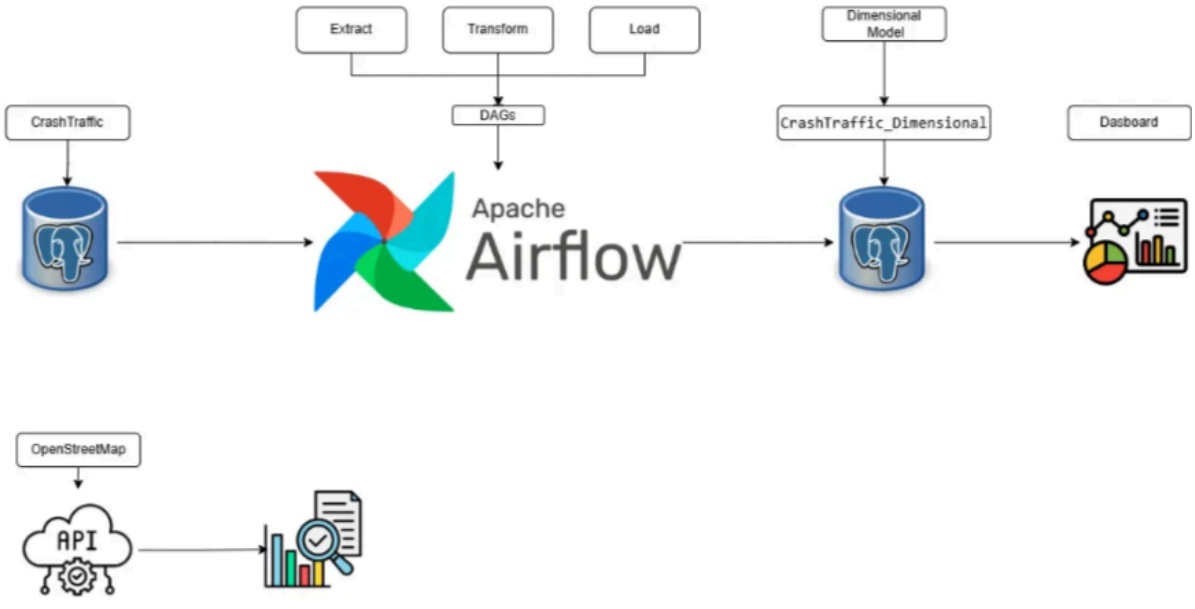
The primary objectives of the Traffic Accidents ETL project are:

- **Data Extraction**: Collect traffic accident data from a CSV file sourced from Kaggle, which is then migrated to a PostgreSQL database. Additionally, retrieve complementary geospatial data from the OpenStreetMap (OSM) API.
- **Data Transformation**: Clean, enrich, and transform raw data into a structured dimensional model suitable for analysis.
- **Data Loading**: Store the transformed data into a new database for downstream analytics.
- **Workflow Orchestration**: Use Apache Airflow to automate and manage the ETL pipeline, ensuring reliability and scalability.
- **Visualization**: Provide an interactive dashboard to visualize key insights, such as accident hotspots and infrastructure-related trends.

---

3. System Architecture

The updated architecture of the Traffic Accidents ETL pipeline is depicted in the following diagram:



3.1. Workflow Overview

1. **Dataset Extraction**: Raw traffic accident data is extracted from a PostgreSQL database (`CrashTraffic` table).
2. **Apache Airflow Orchestration**: Airflow manages the ETL pipeline through Directed Acyclic Graphs (DAGs), scheduling and executing tasks.
3. **Dimensional Model Creation**: The raw dataset is transformed into a dimensional model for efficient querying and analysis.
4. **Dashboard Visualization**: The transformed data is loaded into a structured database (`CrashTraffic_Dimensional`) and visualized using a Power BI dashboard.
5. **Exploratory Data Analysis (EDA)**: EDA is performed on the data obtained from the OpenStreetMap API to evaluate its quality, consistency, and relevance, identifying opportunities for integration with the dimensional model.

---

4. Project Structure

The repository structure for the Traffic Accidents ETL project is as follows:

```
TRAFFICACCIDENTS/
├── API/
│   ├── data/
│   ├── notebooks/
│   │   ├── API_EDA.ipynb
│   │   └── git_attributes
├── config/
│   └── conexion_db.py
├── dags/
│   └── etl_crash_traffic.py
├── Dashboard/
├── data/
│   ├── CrashTraffic_clean.csv
│   └── CrashTraffic.csv
├─ logs/
├── notebooks/
│   ├── 001_extract.ipynb
│   ├── 002_EDA_csv.ipynb
```

```
├── pdf/
├── venv/
├── .env
├── .gitignore
├── docker-compose.yaml
├── README.md
└── requirements.txt
```

4.1. Key Directories and Files

- `API/`: Contains resources related to data retrieval and analysis from the OpenStreetMap API.
  - `data/`: Stores data downloaded or generated from the API.
  - `notebooks/`: Jupyter notebooks for exploring and validating geospatial data.
    - `API_EDA.ipynb`: Exploratory data analysis of the data obtained from OSM.
    - `git_attributes`: Git configuration file related to version control in notebooks.
- `config/`:
  - `conexion_db.py`: Script to manage the connection with the PostgreSQL database.
- `dags/`:
  - `etl_crash_traffic.py`: Airflow ETL pipeline definition for automating traffic accident data processing.
- `Dashboard/`: Folder for resources required to build and display dashboards.
- `data/`:
  - `CrashTraffic.csv`: Original dataset downloaded from Kaggle.
  - `CrashTraffic_clean.csv`: Transformed and cleaned version of the dataset, ready for analysis.
- `logs/`: Stores logs from the ETL pipeline, useful for monitoring and debugging.
- `notebooks/`: Notebooks for traffic accident data analysis and preparation.

    - `001_extract.ipynb`: Initial extraction and loading of data from the CSV.
    - `002_EDA_csv.ipynb`: Exploratory data analysis of the traffic accident dataset.
- **`pdf/`**: Space to store reports, presentations, or documentation exported in PDF format.
- **`venv/`**: Virtual environment containing all project dependencies.
- **`.env`**: Environment variables file, such as credentials and sensitive paths, not shared publicly.
- **`.gitignore`**: Defines which files or folders should be ignored by Git (e.g., `venv/`, `__pycache__/`, etc.).
- **`docker-compose.yaml`**: Configuration file to spin up project services (such as Airflow, PostgreSQL) using Docker containers.
- **`README.md`**: Document containing an overview of the project, usage instructions, and installation guide.
- **`requirements.txt`**: List of dependencies needed to run the project in Python.

---

5. Components and Implementation

5.1. Data Extraction

The pipeline begins with extracting data from a PostgreSQL database (`CrashTraffic` table), which was populated from a CSV file sourced from Kaggle. Additionally, geospatial data is retrieved from OpenStreetMap to enrich the dataset with location-based features, such as proximity to hospitals, schools, or intersections

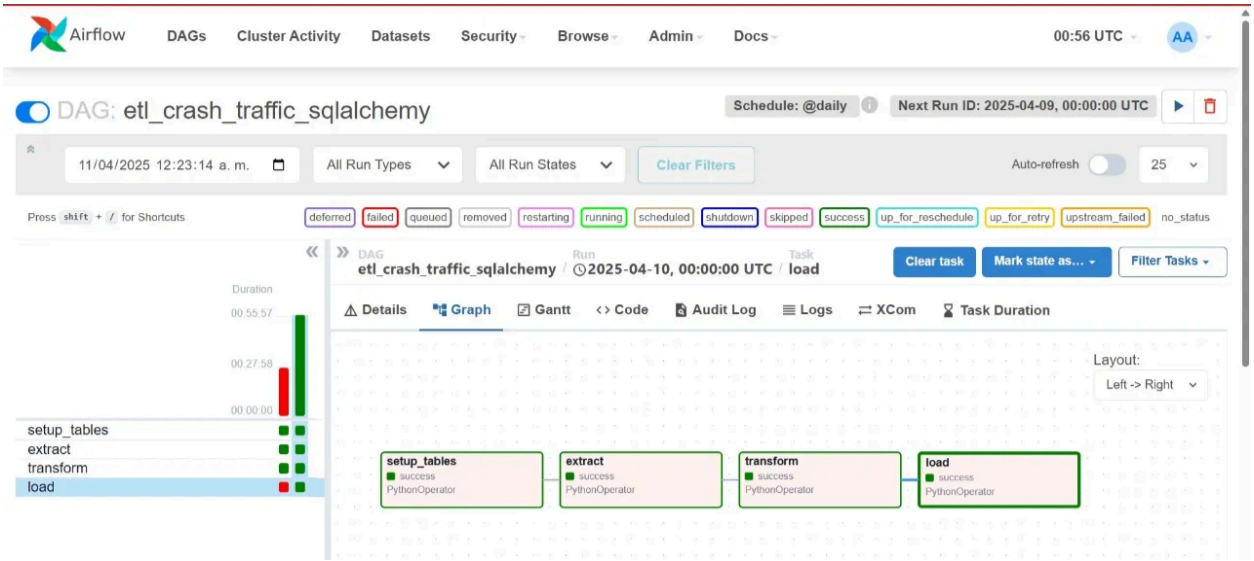
5.2. Apache Airflow DAGs

Apache Airflow is used to orchestrate the ETL workflow. The DAG `etl_crash_traffic.py` defines the sequence of tasks:

- **setup_tables**: Creates tables in the new database (`CrashTraffic_Dimensional`), structured according to the defined dimensional model.
- **Extract**: Retrieves data from the PostgreSQL database (`CrashTraffic`).
- **Transform**: Cleans and enriches the data, creating a dimensional model.

- **Load**: Stores the transformed data into the `CrashTraffic_Dimensional` database.

Airflow's web interface provides monitoring capabilities, as shown in the presentation slides (e.j., Airflow DAGs view)



Airflow DAGs view

5.3. Data Transformation

The transformation step involves:

- Cleaning the raw dataset from the PostgreSQL database (`CrashTraffic`)
- Structuring the data into a dimensional model for efficient querying
- Loading the structured data into the `CrashTraffic_Dimensional` database

5.4. Data Loading

The transformed data is loaded into the `CrashTraffic_Dimensional` database, which serves as the backend for the dashboard. The database schema follows a star schema design, optimized for analytical queries

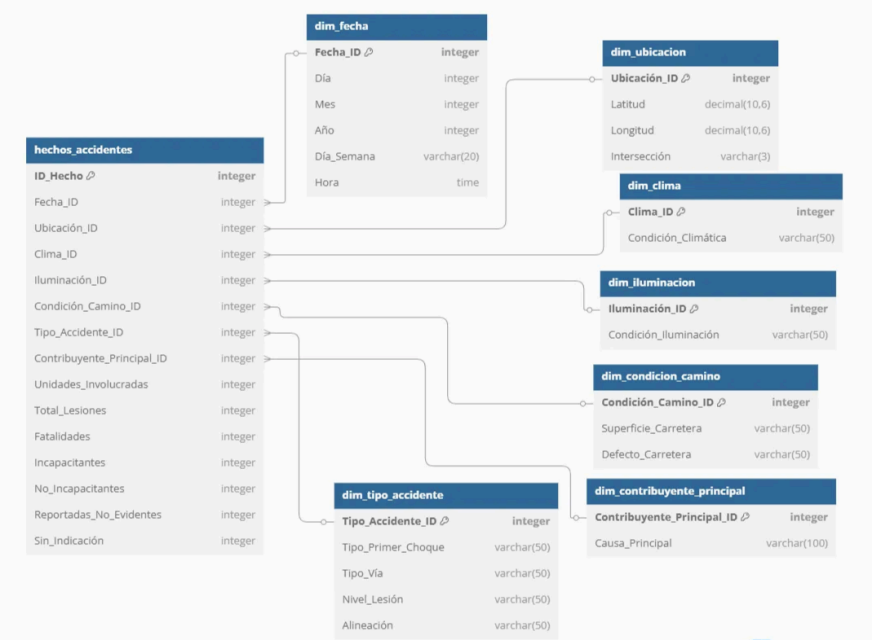
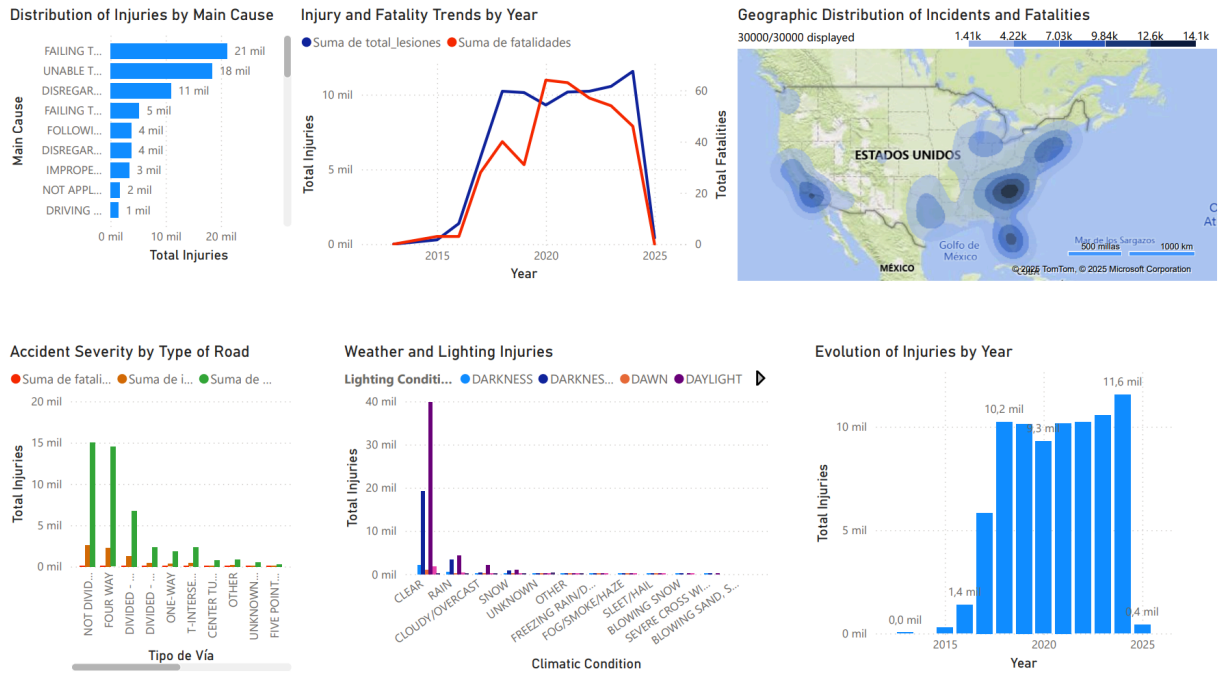


Diagram of the dimensional model

5.5. Dashboard



The dashboard, built using Power BI, visualizes key metrics, including:

- Accident severity by type of road

- Weather and lighting conditions affecting injuries
- Evolution of injuries by year
- Fatality of accidents by location
- Correlation between accidents and infrastructure (e.j., lack of traffic signals)
- Trends in the number of injuries over time

5.6. Exploratory Data Analysis (EDA)

EDA is performed at two stages:

- **API EDA** (`API_EDA.ipynb`): Analyzes raw data from the OpenStreetMap API.
- **Dataset EDA** (`002_EDA_csv.ipynb`): Validates the cleaned dataset and identifies trends before loading it into the `CrashTraffic` database for the ETL process.

---

6. Setup and Installation

6.1. Prerequisites

- Python 3.8 or higher
- Docker and Docker Compose (for running Airflow on Windows)
- PostgreSQL or another compatible database
- Git

6.2. Installation Steps

1. **Clone the Repository**:

```
git clone <https://github.com/isabellaperezcav/Traffic-Accidents-ETL.git>
cd Traffic-Accidents-ETL
```

2. **Set Up a Virtual Environment**:

```
python -m venv venv
source venv/bin/activate  # En Windows: venv\\Scripts\\activate
```

```
pip install -r requirements.txt
```

3. **Configure Environment Variables**:
   - Create a `.env` file in the `venv/` directory
   - Add the following variables:

```
AIRFLOW__CORE__EXECUTOR=SequentialExecutor
AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg
2://usuario:contraseña@localhost:5432/airflow
DB_HOST=localhost
DB_PORT=5432
DB_USER=tu_usuario
DB_NAME=tu_base_de_datos
DB_PASS=tu_contraseña
```

4. **Set Up Airflow with Docker**:
   - Start the Airflow services using Docker Compose:

```
docker-compose up -d
```

   - Access the Airflow web interface at `http://localhost:8080`
5. **Initialize the Database**:
   - Configure the database connection in `config/conexion_db.py`
   - Run any necessary migrations or schema setup scripts

6.3. Running the Pipeline

1. Iniciar el programador y el servidor web de Airflow (si no se usa Docker)
2. Habilitar el DAG `etl_crash_traffic` en la interfaz de Airflow
3. Monitor the pipeline execution and logs via the Airflow interface, as shown in the Airflow DAGs view (Section 5.2)

---

7. Usage

7.1. Running the ETL Pipeline

- The pipeline is scheduled via Airflow, but you can trigger it manually:
  - In the Airflow UI, navigate to the DAGs section
  - Toggle the `etl_crash_traffic` DAG to "On" and click "Trigger DAG"

7.2. Viewing the Dashboard

- Once the pipeline completes, the data in the `CrashTraffic_Dimensional` database is updated
- Access the dashboard in Power BI to explore visualizations

7.3. Performing EDA

- Run the EDA notebooks to analyze the data:

```
API/notebooks/API_EDA.ipynb
/notebooks/002_EDA_csv.ipynb
```

---

8. Maintenance and Troubleshooting

8.1. Monitoring

- Use the Airflow web interface to monitor DAG executions, task states, and logs (`logs/` directory)
- Check the `logs/` directory for detailed error messages if a task fails

8.2. Common Issues

- **Database Connection Errors**: Verify the credentials in `.env` and `conexion_db.py`
- **Airflow Task Failures**: Ensure all dependencies are installed (`requirements.txt`) and that the DAG file is syntactically correct

- **API Rate Limits**: Handle rate limits by implementing retry logic in the extraction stage

8.3. Updates

- To update the pipeline, modify the `etl_crash_traffic.py` DAG and redeploy through Airflow
- Add new functionalities to the dashboard by updating the data processing logic in the `Dashboard/` directory

---

9. References

- Apache Airflow Documentation
- OpenStreetMap
- Project Repository
- Power BI

---

10. Glossary

- **ETL**: Extract, Transform, Load—a process for data integration
- **DAG**: Directed Acyclic Graph, used by Airflow to define workflows
- **Dimensional Model**: A database schema optimized for analytical queries, often using a star or snowflake design