# 

# Project Overview

This project involved analyzing and visualizing the electric vehicle registration data from the state of Washington, with a focus on its year-over-year evolution by county. The pipeline was built using Python, PostgreSQL, and Power BI, and was structured into clear modules: exploration, cleaning, database loading, and analysis.

During implementation, critical data fields (model\_year, county) were standardized and cleaned to ensure analysis quality. Year-over-year (YoY) changes were accurately calculated using an SQL subquery with LAG(), and null and zero values were handled appropriately based on their analytical context.

Additionally, technical issues such as load errors due to URLs (HTTP 400) were resolved by opting to download the CSV file locally. Data cleaning and record exclusion decisions were thoroughly documented to ensure traceability and reproducibility of the process.

# Explanation

## Step 0

To begin, it's essential to understand the dataset and what each field represents.

This dataset contains information about electric vehicles registered in the state of Washington, USA.

| **Columna** | **Significado** |
| --- | --- |
| **VIN (1-10)** | First 10 characters of the Vehicle Identification Number (unique for each car, similar to a chassis number). Only the first 10 characters are shown for privacy and security reasons. |
| **County** | County where the vehicle is registered |
| **City** | City where the vehicle is registered. |
| **State** | State (all values are “WA” for Washington). |
| **Postal Code** | Postal code of the vehicle registration. This can vary within the same city, especially in larger ones. |
| **Model Year** | Model year of the vehicle. |
| **Make** | Vehicle manufacturer (e.g., Tesla, Nissan, etc.). |
| **Model** | Specific model of the vehicle. |
| **Electric Vehicle Type** | Type of electric vehicle: either **BEV** (fully electric) or **PHEV** (plug-in hybrid). |
| **Clean Alternative Fuel Vehicle (CAFV) Eligibility** | Indicates whether the vehicle qualifies as a “Clean Alternative Fuel Vehicle” under state criteria (eligible vehicles may receive incentives or benefits). |
| **Electric Range** | Electric range in miles (how far the vehicle can travel using electricity only). |
| **Base MSRP** | Manufacturer’s Suggested Retail Price (in USD), excluding taxes, tariffs, or markups. |
| **Legislative District** | State legislative district where the vehicle is registered. |
| **DOL Vehicle ID** | Unique vehicle ID from the Department of Licensing. Used for administrative purposes and allows **unique identification** of each vehicle in the dataset. |
| **Vehicle Location** | Approximate geographic coordinates (latitude and longitude) of the registration. |
| **Electric Utility** | Electric utility provider in the area. |
| **2020 Census Tract** | U.S. Census 2020 tract (used for demographic/geographic analysis). |

## Step 1 – Set Up the Local Environment

Requirements:

* Python 3.8+
* PostgreSQL
* Power BI Desktop
* Git + GitHub

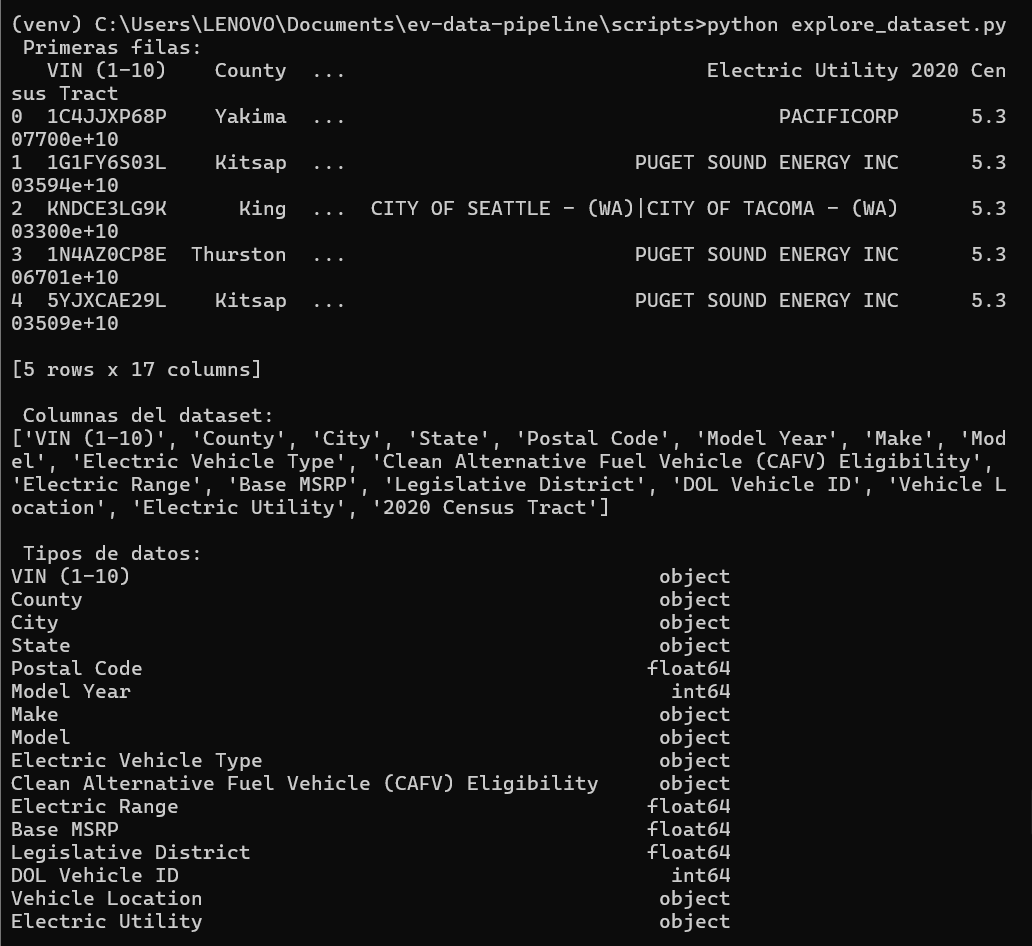
With these requirements installed, we create a virtual environment to work in isolation with the libraries needed for this project, without affecting the rest of your system. This approach makes it easier for anyone to replicate the same environment on another machine using the requirements.txt file.

## Step 2 – Download and Inspect the Dataset

This step is implemented in the ***explore\_dataset.py***file.

We use pandas to download the dataset as a CSV file, and then perform an exploratory analysis to examine:

* Column names
* First few records
* Data types
* Number of nulls per column
* Unique values of key columns (`Electric Vehicle Type` and `Clean Alternative Fuel Vehicle (CAFV) Eligibility`)
* Number of vehicles registered per year



## Step 3 – Cleaning

This step is implemented in the ***clean.py*** file.

**Considerations:**

Data cleaning was limited to the columns necessary to answer the questions posed in the challenge. In the future, this process could be extended to the remaining columns.

1. **Standardize Column Names**

Convert all column names to lowercase.

Replace spaces with underscores.

Remove any character that is not a lowercase letter (a–z), digit (0–9), or underscore (\_).

This helps prevent future errors that can occur due to inconsistent naming, especially when referencing columns in code.

1. **Convert model\_year to Numeric Type**

Convert each value in the model\_year column to a numeric type.

If a value is incorrectly formatted and cannot be converted, it is replaced with NaN (Not a Number).

This ensures the code doesn't break during analysis and allows for filtering or removing invalid rows later. It's especially useful for performing year-based analyses.

1. **Verify That Dates Are Correct**

During the exploratory analysis, a model year of 2026 was found—this is inconsistent with reality. Such records should be removed to maintain data accuracy.

1. **Remove Records with Missing Critical Data**

Drop all rows where either model\_year or county is missing. This is a business-driven decision, as these fields are critical for the intended analysis.

Null values in vehicle\_location are ignored. This field is useful for detailed visualizations (e.g., maps) but is not essential for key analysis metrics.

Whenever decisions are made about handling null or duplicate values, it's important to consider their impact on the analysis goals. In team projects, it's best to validate these choices with stakeholders or business analysts..

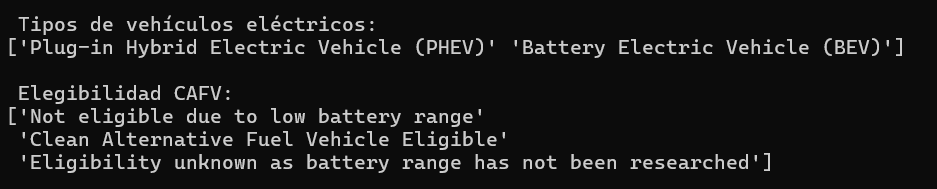
1. **Remove Whitespace**

Removes whitespace (including tabs or line breaks) from the beginning and end of each value.

Invisible spaces at the start or end of a value can break analyses without us realizing it.

1. **Verify Consistency of Columns Electric Vehicle Type and Clean Alternative Fuel Vehicle (CAFV) Eligibility**

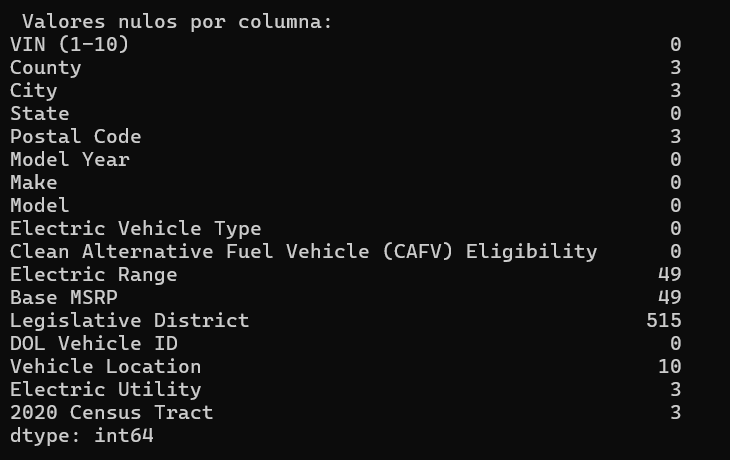
These columns are also key to the analysis. The exploratory analysis reveals the following:



Indicating that these columns have unique and well-defined values, with no unusual duplicates caused by typos.

However, .unique() only shows the present values — it does not include NaN.

To check for nulls, we look at the number of null values per column:



We conclude that these columns are already clean

1. **Save the cleaned data**

Save the cleaned DataFrame to a CSV file at the specified path, excluding the pandas index.

## Step 4 – Create the PostgreSQL Database

Create the database (run in terminal or pgAdmin):

CREATE DATABASE ev\_data;

The table creation and data loading process are handled in the ***load\_to\_postgres.py*** file.

## Step 5 – SQL Queries

The analysis queries are contained in the ***analysis\_queries.sql*** file.

## Step 6 – Power BI

Open Power BI Desktop.

Go to Home > Get Data > PostgreSQL.

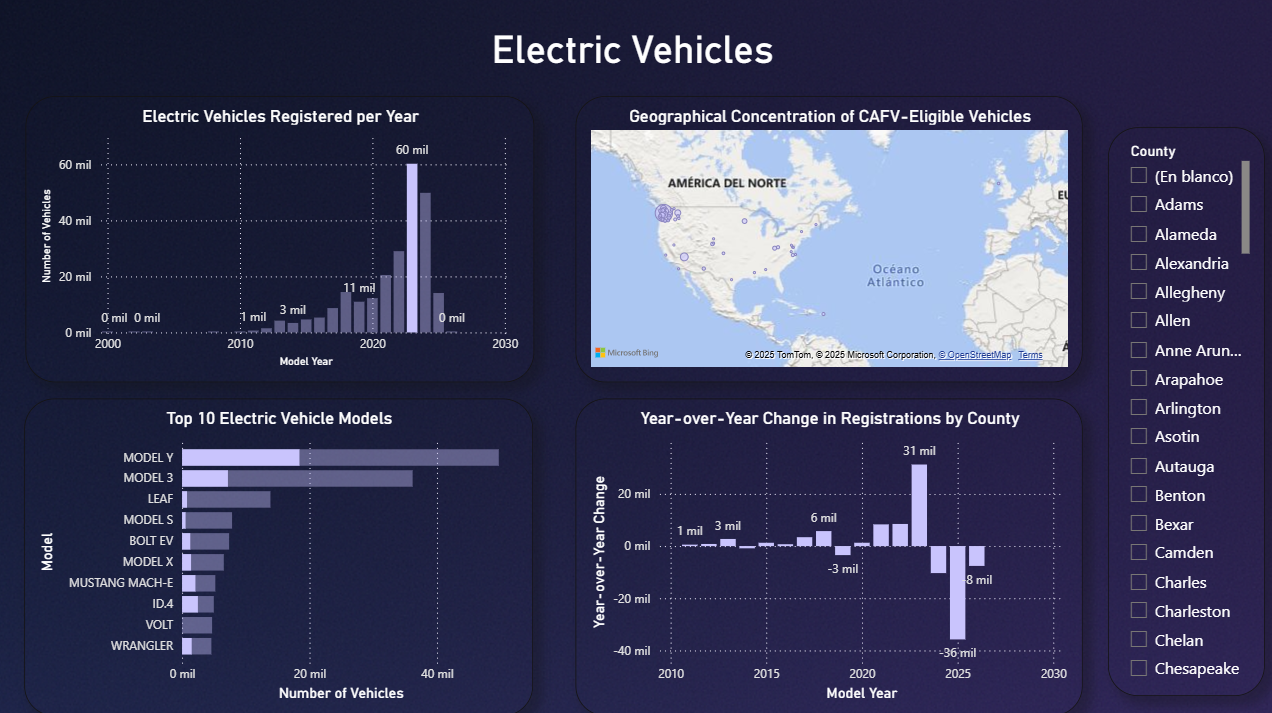
Connect using:

Server: localhost

Database: ev\_data

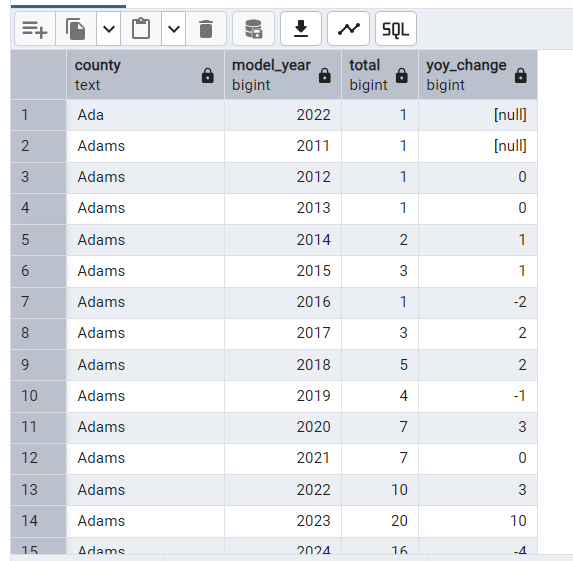
Load the vehicles table.

Dashboard:



## Main challenges encountered

* **Handling Null Values in Year-over-Year (YoY) Calculation**A decision had to be made on how to handle null values that naturally arise in the YoY change calculation—specifically for the first year recorded in each county, where no prior year exists for comparison.
* **Interpreting Zero Values in YoY Change**It was important to recognize that a value of 0 is a valid result—indicating no change from the previous year—and not an error in the data or the calculation.



* **Choosing the Right SQL Strategy for Calculating yoy\_change**

The initial approach attempted to use COUNT(\*) - LAG(COUNT(\*)) directly, which is unsafe and not supported by all SQL engines. A more robust solution was chosen: first calculating yearly totals in a subquery, then applying the LAG() function.

* **Handling Incomplete Data**

Rows with missing values in model\_year or county were removed, as these fields are essential for the analysis. Records missing vehicle\_location were retained, since this field is not critical to the main KPIs.

* **Counties with Limited Data Availability**

Some counties have records for only one or two years, limiting their analytical value. This raises the need to consider how to handle such gaps in visualizations or reports. One option is to include only counties with at least three years of data.

* **HTTP Error 400: Bad Request**

This occurred because the dataset URL could not be accessed directly via pandas.read\_csv(). This may happen due to redirects (e.g., to a temporary link) or protections against simple automated access.

The solution was to manually download the CSV file once and save it locally. This provides greater pipeline stability, avoids interruptions due to network issues or URL changes, and ensures an exact copy of the dataset used in the project is preserved.