Documentación

Resumen

En este proyecto desarrollé un pipeline de datos automatizado para ingestar, limpiar, transformar y almacenar datos de registros de vehículos eléctricos. A continuación, el resumen de mi enfoque:

1. **Automatización de descarga**: para ello usé Beautiful Soup en Python para hacer web scraping del CSV público con datos de Electric Vehicle Population Data.
2. **Limpieza de datos**: con Pandas leí el CSV, corregí tipos de datos (fechas, numéricos), eliminé registros nulos o inconsistentes, y normalicé formatos de texto (por ejemplo, nombres de condados).
   Resultado: un DataFrame limpio listo para análisis.
3. **Generación de archivo Parquet**: Exporté el DataFrame limpio a formato Parquet. Esto es porque Airflow no me permite pasar datos tan pesador por xcom.
4. **Modelado y creación de tablas**: Diseñé un esquema en estrella (star schema) con:
   - **Dimensiones**: dim_vehicle, dim_date, dim_location, dim_electric_type, dim_policy
   - **Tabla de hechos**: fact_registration
5. **Carga de datos en PostgreSQL**: Mediante SQLAlchemy y COPY en Postgres, cargué los datos del Parquet en sus tablas correspondientes. Las dimensiones se llenan evitando duplicados (usando slow change dimension), y el hecho enlaza a los IDs.

DAGS y Scripts

- **dags/electric_vehicles_etl.py**: define un DAG con 5 tareas: descargar CSV, limpiar, convertir a Parquet, crear esquema en Postgres, cargar datos.
- **Lógica en etl/**:
  - data_reader.py: descarga y lee CSV.
  - data_cleaner.py: aplica transformaciones Pandas.
  - schema_creator.py: ejecuta sentencias DDL para tablas.
  - load_data.py: realiza la carga eficiente del Parquet con SQLAlchemy y COPY.
  - db_connection.py: configuración de la conexión a Postgres.

Decisiones de diseño

- **Parquet vs CSV**: Parquet ofrece compresión y tipos nativos, acelerando cargas.

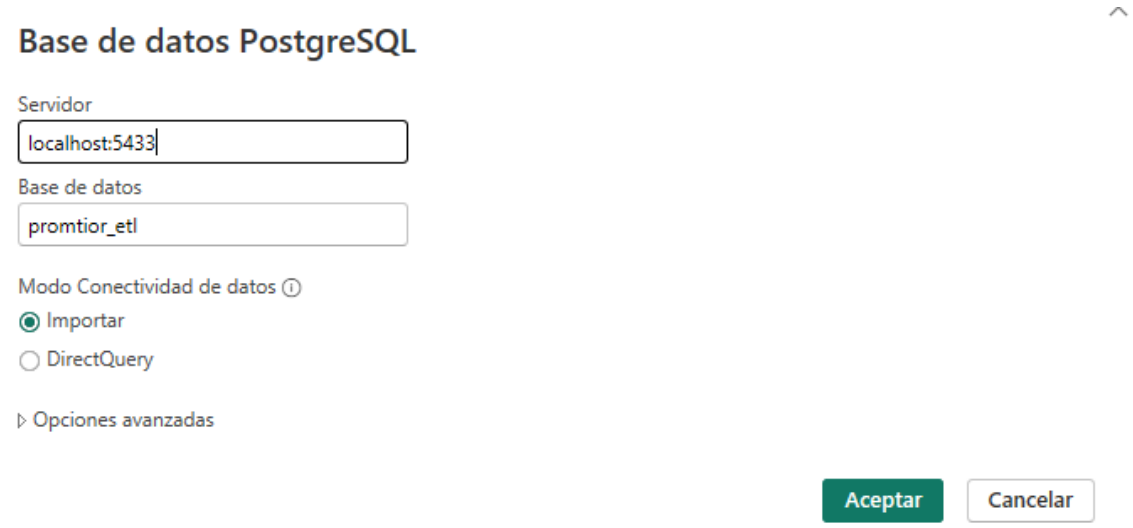
- **Star Schema**: mejora rendimiento de consultas analíticas y facilita visualizaciones.
- **Airflow**: permite programar re-ejecuciones, recuperaciones en fallos y monitorización de logs.

Transformaciones de datos

1. **Fechas**: básicamente, se creó desde 0 la tabla dim_date.
2. **Normalización de texto**: todas las columnas poseen nombres en snake_case
3. **Extracción de coordenadas**: se obtuvo tanto la latitud como la longitud a partir de la columna POINT (...)
4. **Rellenado de valores**: algunos valores faltantes (como valores numéricos) fueron rellenados con 0

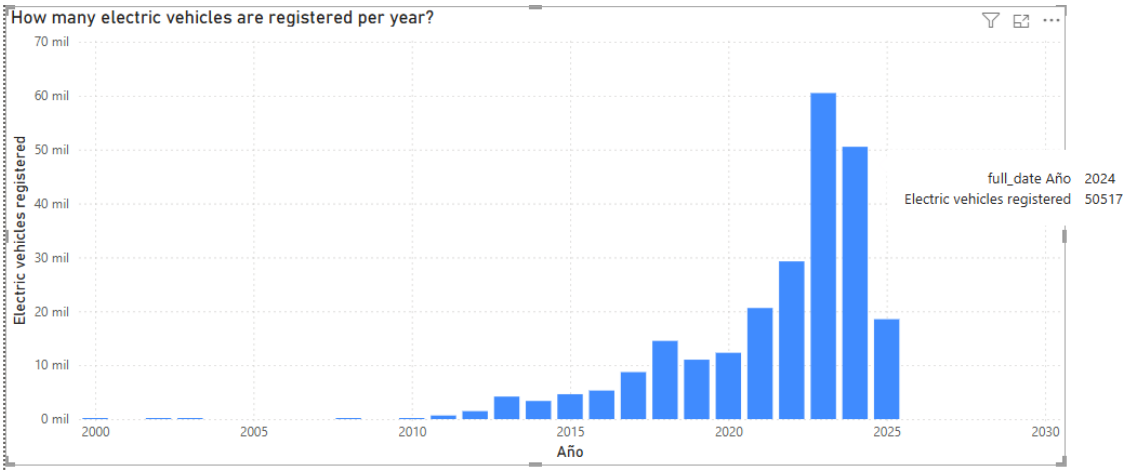
Análisis de datos

Se configuró Docker para que ya traiga una base de datos integrada, hosteada en el puerto 5433 por lo que solo se tuvo que conectar a dicho host y luego seleccionar las tablas creadas en el proyecto.



<u>- How many electric vehicles are registered per year?</u>

Para obtener este gráfico se creó la medida Total Registros que lo único que hace es contar las filas de la tabla de hechos

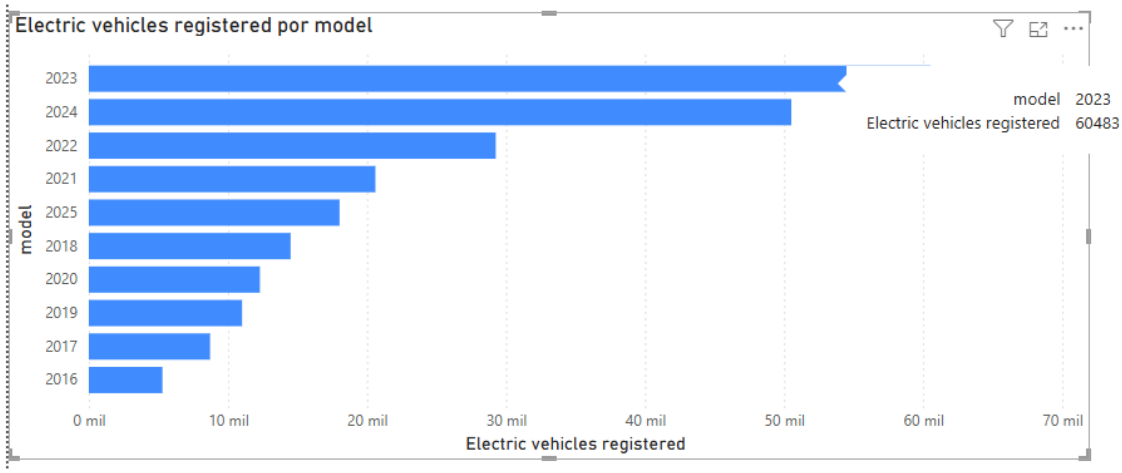


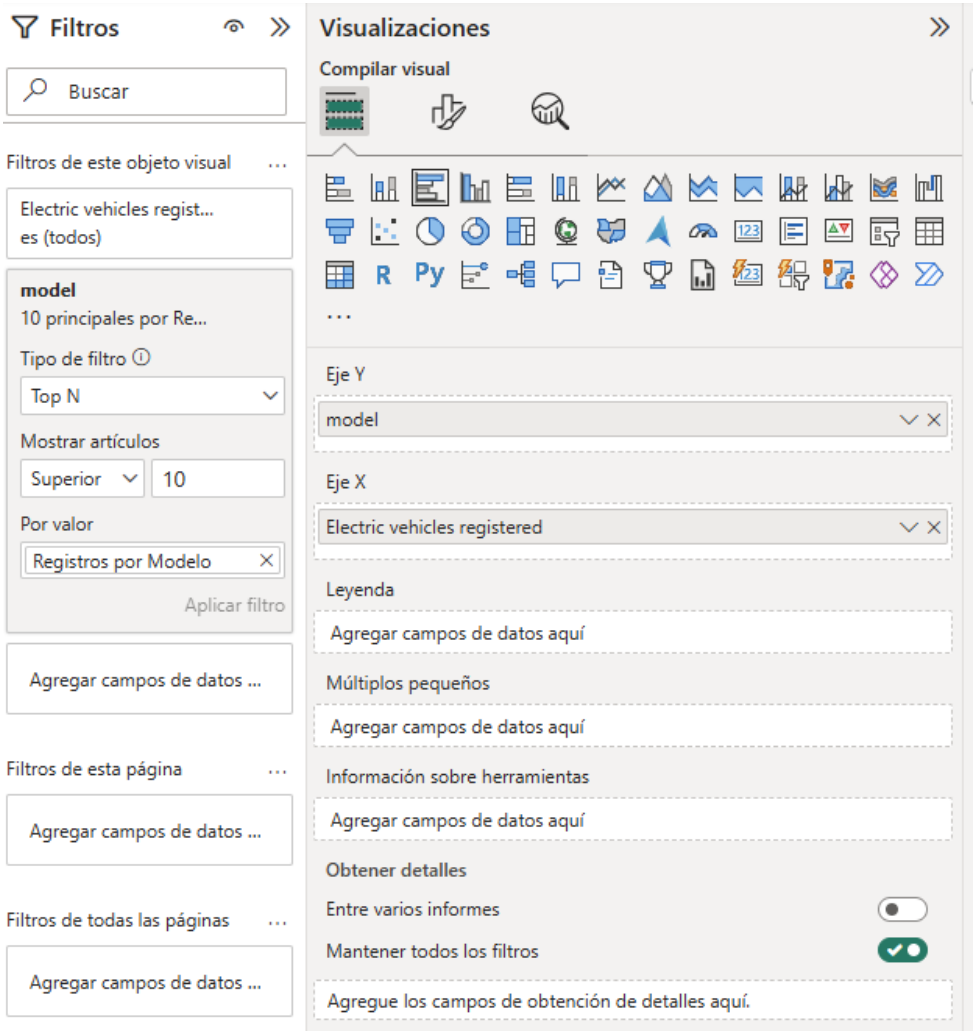

```
Total Registros =
COUNTROWS('public fact_registration')
```

- What are the top 10 electric vehicle models by registration count?

Para obtener este gráfico se creó la medida Registros por Modelo que quita todos los filtros de la tabla excepto los que se aplican al modelo.

Además, se filtró por el top 10 de modelos según Registros por Modelo



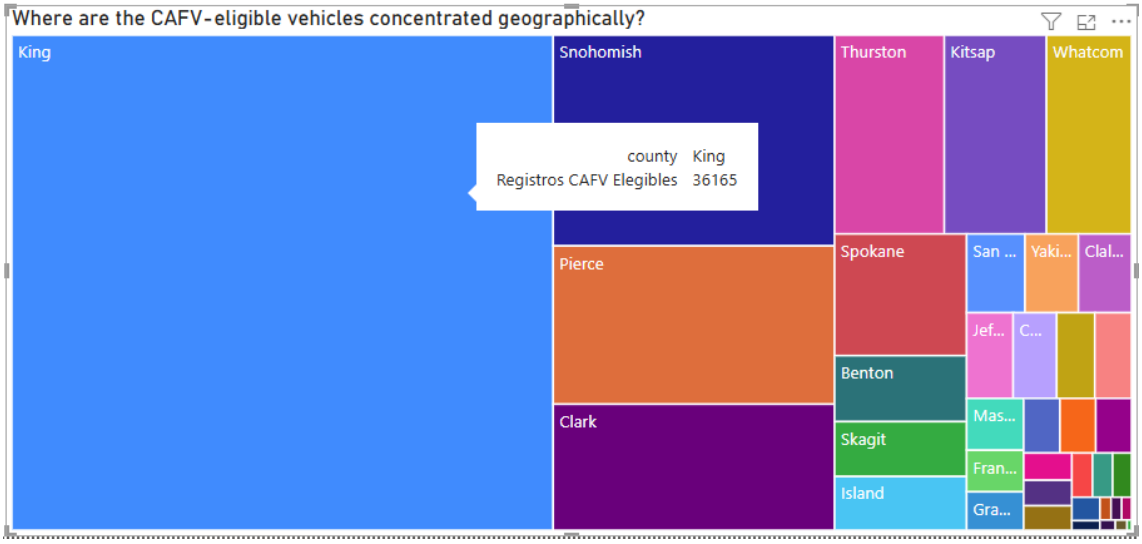



```
1 Registros por Modelo =
2 CALCULATE(
3     COUNTROWS('public fact_registration'),
4     ALLEXCEPT('public dim_vehicle', 'public dim_vehicle'[model])
5 )
```

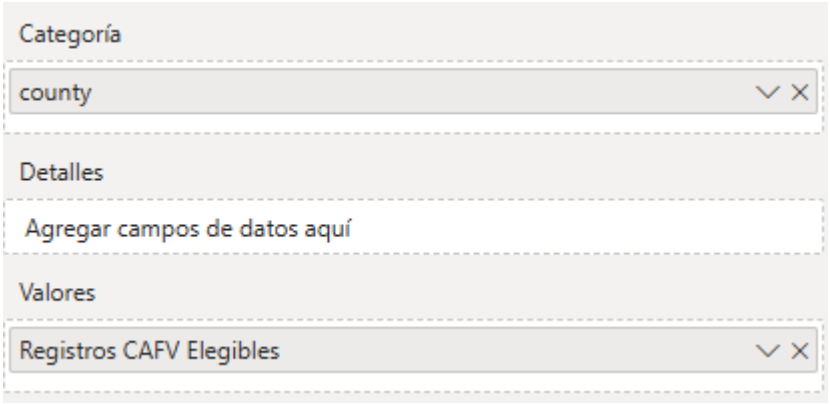
- Where are the CAFV-eligible vehicles concentrated geographically?

Para obtener este gráfico se creó la medida Registros CAFV Elegibles cuenta todas las columnas de dim_policy que contengan un cafv_eligibility = ‘Clean Alternative Fuel Vehicle Eligible’

Y se mostró por esos valores por condado




```
1  Registros CAFV Elegibles =
2  CALCULATE(
3      COUNTROWS('public fact_registration'),
4      'public dim_policy'[cafv_eligibility] = "Clean Alternative Fuel Vehicle Eligible"
5  )
```

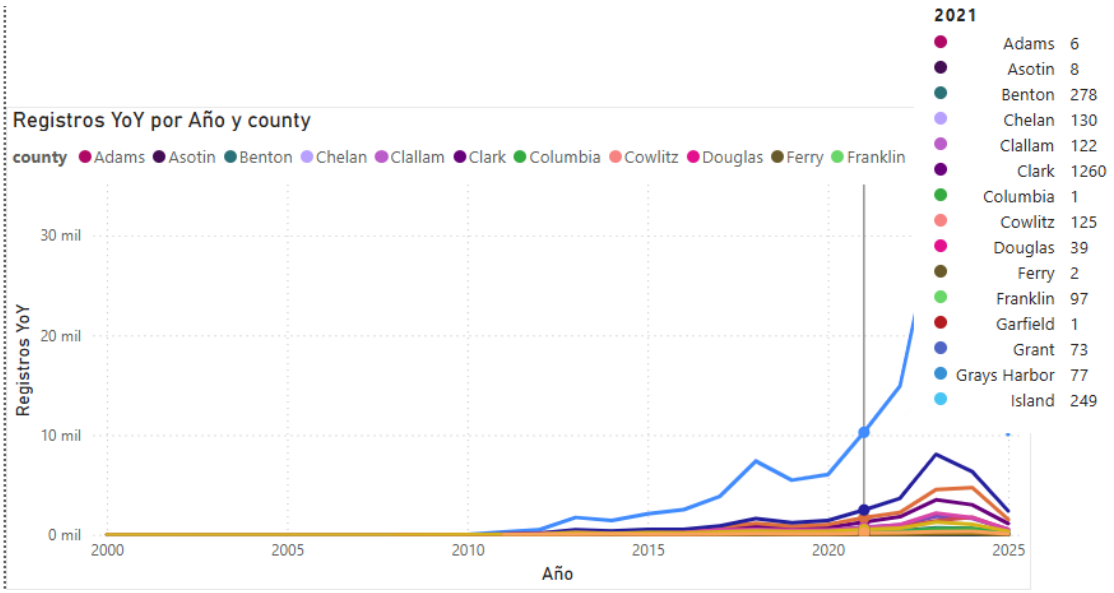


- What is the year-over-year change in electric vehicle registrations by county?

Para obtener este grafico se creó la medida Registros YoY que consta de 3 partes:

1) TotalAnioActual: cuenta todas filas de la tabla de hechos
2) TotalAnioAnterior: cuenta todas las filas de la tabla hechos pero de un año anterior
3) DiferenciaYoY: resta TotalAnioActual menos TotalAnioAnterior dando como resultado, para un año en especifico, la diferencia de registros que había entre ese año y el anterior.

Se usa como leyenda a los condados para verlos como líneas y se divide por años




```
Registros YoY =
VAR TotalAnioActual = COUNTROWS('public fact_registration')
VAR TotalAnioAnterior =
    CALCULATE(
        TotalAnioActual,
        SAMEPERIODLASTYEAR('public dim_date'[full_date])
    )
VAR DiferenciaYoY =
    TotalAnioActual - TotalAnioAnterior
RETURN
    TotalAnioActual
```