MODELOS DE DATOS.

1. Visión general del modelo de datos

El modelo de datos se ha diseñado para seguir la arquitectura del proyecto:

1. **Ingesta** (Python Producer)
    - Consulta las APIs de Open Data Valencia (contaminación y meteorología).
    - Envía cada registro como mensaje JSON a Kafka (`valencia_pollution` y `valencia_weather`).
2. **Streaming** (Kafka)
    - Kafka actúa como búfer de alta velocidad para los mensajes.
3. **Procesamiento y almacenamiento RAW** (Python Consumer + PostgreSQL)
    - El *consumer* lee ambos topics y guarda los mensajes en tablas RAW de PostgreSQL:
        - `raw_pollution`
        - `raw_weather`
4. **Transformación** (dbt)
    - A partir de `raw_pollution` se construye una tabla limpia:
        - `clean_pollution`
5. **Visualización** (Dash / Plotly)
    - El dashboard consulta únicamente `clean_pollution` para generar mapas y gráficos.

2. Tablas RAW en PostgreSQL

2.1. Tabla `raw_pollution`

Propósito
Almacenar de forma íntegra los mensajes de contaminación atmosférica recibidos desde Kafka, preservando el JSON original de la API.

Granularidad
Una fila = un mensaje de contaminación para una estación en un instante de tiempo.

Esquema

- *id*
  - Tipo: SERIAL PRIMARY KEY
  - Definición: identificador técnico interno de la tabla.
- *ingestion_time*
  - Tipo: TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  - Definición: marca de tiempo en la que el *consumer* inserta el registro en PostgreSQL.
  - Uso: permite ordenar cronológicamente las mediciones y recuperar el dato más reciente por estación.
- *data*
  - Tipo: JSON
  - Definición: contenido completo del mensaje JSON recibido desde la API de **estaciones de contaminación**.
  - Campos relevantes dentro de *data* (se usan en etapas posteriores):
    - no2: valor medido de dióxido de nitrógeno (µg/m³) que se utiliza como indicador principal de contaminación en el proyecto.
    - pm10, pm25, o3, so2, co: otros contaminantes monitorizados (no explotados todavía en el modelo limpio, pero disponibles).
    - nombre: nombre de la estación.
    - objectid: identificador de la estación en el dataset original.
    - geo_point_2d: objeto JSON con lat y lon que indica la localización geográfica de la estación.
    - Otros metadatos que la API pueda incluir (fecha de medición, códigos internos, etc.).

Esta tabla es la "copia fiel" de la API de contaminación. No se pierden campos; solo se añade la columna técnica `ingestion_time`.

2.2. Tabla `raw_weather`

Propósito
Almacenar los mensajes JSON con información meteorológica generados a partir del dataset de **estaciones atmosféricas**.

Granularidad
Una fila = un mensaje de meteorología para una estación en un momento dado.

Esquema

- *id*
  - Tipo: SERIAL PRIMARY KEY
  - Definición: identificador técnico del registro.
- *ingestion_time*
  - Tipo: TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  - Definición: instante en que se guarda el mensaje en la base de datos.
- *data*
  - Tipo: JSON
  - Definición: JSON completo devuelto por la API de estaciones atmosféricas.
  - Contiene campos como:
    - nombre: estación meteorológica.
    - geo_point_2d: coordenadas (latitud/longitud).
    - Variables meteorológicas (dirección del viento, temperatura, humedad, etc.), según lo que proporcione la API.

En la versión actual del proyecto, `raw_weather` se conserva como almacenamiento bruto para futuras ampliaciones. La tabla limpia `clean_pollution` se nutre únicamente de `raw_pollution`.

3. Tabla CLEAN en PostgreSQL (dbt)

3.1. Modelo `clean_pollution` (dbt → tabla `clean_pollution`)

Este modelo dbt se materializa como una tabla física y representa los **"datos limpios"** que consume el dashboard.

Propósito
Proporcionar una vista simplificada con las columnas mínimas necesarias para el análisis y la visualización: índice de contaminación por NO_2, estación y ubicación.

Granularidad
Una fila = una medición de NO_2 para una estación, en un instante de ingesta (*ingestion_time*).

Atributos

- *indice_aqi*
  - Tipo: NUMERIC
  - Definición: valor numérico del contaminante NO_2 extraído de data-'no2'.
  - Rol: se utiliza como **proxy de índice de calidad del aire** en el mapa y en las tablas (mayor valor ⇒ peor calidad).
- *nombre_estacion*
  - Tipo: TEXT
  - Definición: nombre de la estación.
    - Si el campo nombre no viene informado, se genera un nombre genérico usando objectid ('Estación ' || objectid).
  - Rol: etiqueta principal en el dashboard (listas de estaciones, tooltips del mapa, etc.).
- *latitud*
  - Tipo: NUMERIC
  - Definición: latitud decimal de la estación, extraída de data-'geo_point_2d'-'lat'.
  - Rol: necesaria para ubicar la estación en el mapa (Geopy + Plotly).
- *longitud*
  - Tipo: NUMERIC
  - Definición: longitud decimal de la estación, extraída de data-'geo_point_2d'-'lon'.
  - Rol: junto con la latitud, permite representar la estación en el mapa y calcular distancias.
- *ingestion_time*
  - Tipo: TIMESTAMP
  - Definición: marca temporal heredada de la tabla RAW; indica cuándo llegó ese dato al sistema.

- Rol: el dashboard usa DISTINCT ON (nombre_estacion) ... ORDER BY ingestion_time DESC para quedarse con el **último dato disponible de cada estación**.

4. Relación con la visualización

El dashboard realiza consultas del tipo:

```
SELECT DISTINCT ON (nombre_estacion)
       nombre_estacion, indice_aqi, latitud, longitud, ingestion_time
FROM clean_pollution
ORDER BY nombre_estacion, ingestion_time DESC;
```

Con esto se consigue:

- Un registro por estación, siempre el **más reciente**.
- Los campos necesarios para:
  - Pintar el **mapa de calor** (*latitud, longitud, índice*).
  - Listar las **estaciones más contaminadas** (ranking por *indice_aqi*).
  - Mostrar al usuario en qué momento (*ingestion_time*) se actualizó la información.