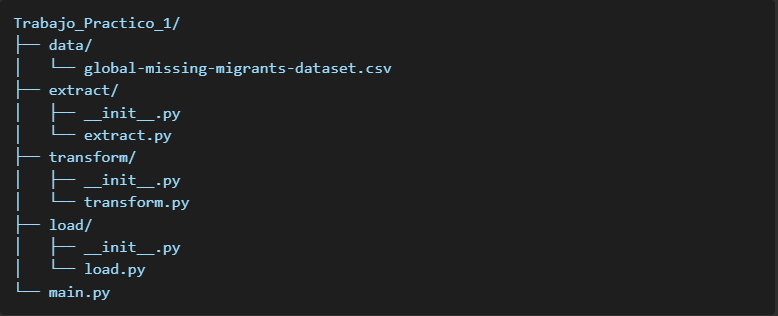
**ETL Modular en Python: Detalle de Cada Módulo**

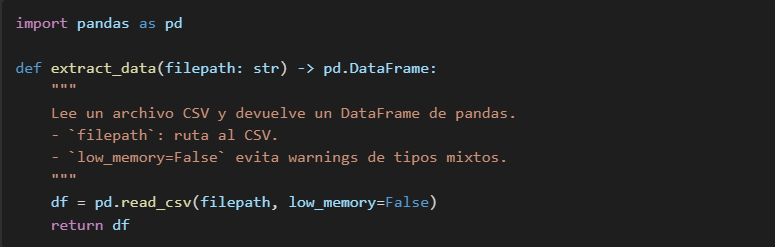
**Este documento describe en detalle cada archivo y función de nuestro pipeline ETL modular, para que puedas consultarlo y comprender cómo trabaja paso a paso.**

**Estructura del proyecto**



* **extract/**: lectura de datos.- **transform/**: limpieza y preparación.- **load/**: generación de resultados (gráficos, archivos).
* **main.py**: orquesta todas las fases.

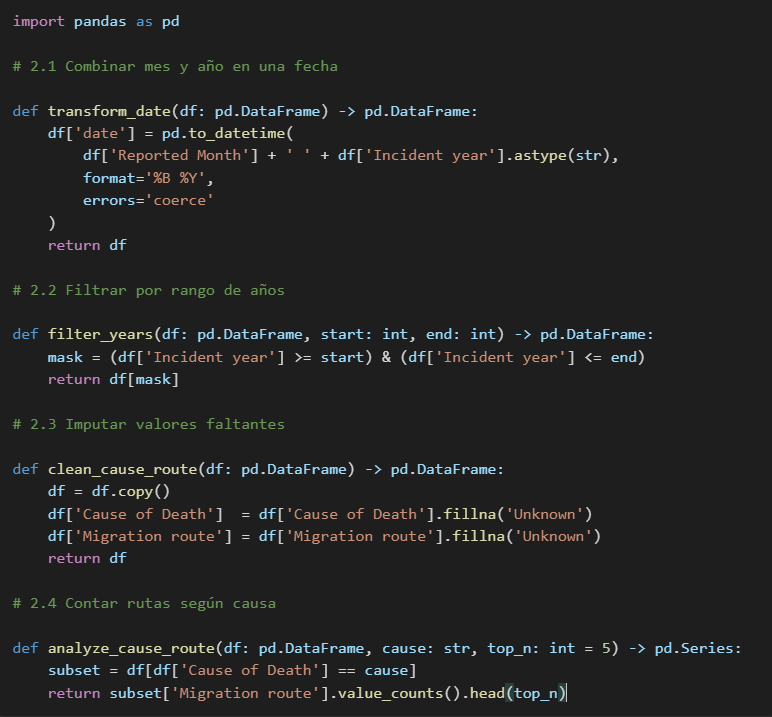
**1. extract/extract.py**



**Explicación:**

* pd.read\_csv carga el CSV completo en un **DataFrame**, la principal estructura tabular de pandas.
* Un **DataFrame** se compone de múltiples **Series** (columnas) etiquetadas.
* El parámetro low\_memory=False mejora la inferencia de tipos.

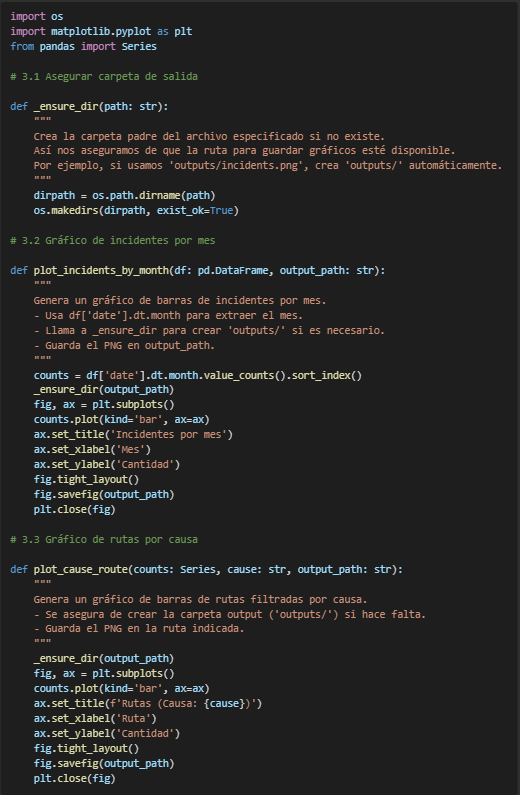
**2. transform/transform.py**



**Explicación de funciones:**

1. **transform\_date**
   * Une la columna de texto del mes (Reported Month) con el año (Incident year).
   * Usa pd.to\_datetime para crear la columna date de tipo datetime64.
   * errors='coerce' convierte cadenas inválidas en NaT.
2. **filter\_years**
   * Crea una máscara booleana comparando la columna Incident year con los límites.
   * Devuelve las filas cuyo año esté en el rango.
3. **clean\_cause\_route**
   * df.copy() evita advertencias de pandas al modificar vistas.
   * .fillna('Unknown') reemplaza valores nulos con "Unknown" en las columnas específicas.
4. **analyze\_cause\_route**
   * Filtra el DataFrame por una causa dada.
   * Usa value\_counts() para obtener una **Series** (índice = rutas, valor = conteo).
   * .head(top\_n) devuelve las rutas más frecuentes.

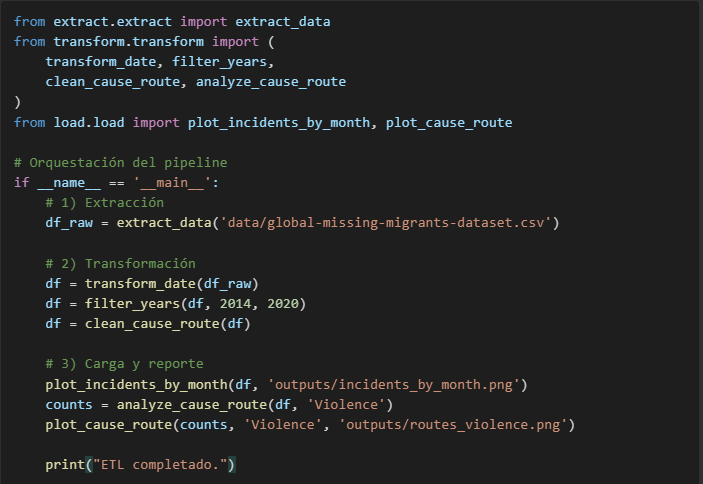
**3. load/load.py**



**Explicación de elementos claves:**

* **os.makedirs(dirpath, exist\_ok=True)**: crea la carpeta de salida si no existe.
* **matplotlib.pyplot**: biblioteca de gráficos; plt.subplots() crea figura y ejes.
* **Series**: estructura unidimensional de pandas para conteos.
* df['date'].dt.month: extrae mes de tipo datetime.

**4. main.py**



**Explicación de main.py:**

* **Imports claros**:
  + extract\_data, transform\_date, filter\_years, clean\_cause\_route, analyze\_cause\_route y funciones de load se importan directamente.
* **if \_\_name\_\_ == '\_\_main\_\_':**:
  + Garantiza que la orquestación solo ocurra al ejecutar el script directamente, no al importar como módulo.
* **Paso 1: Extracción**:
  + df\_raw = extract\_data(...) carga todos los datos crudos en un DataFrame.
* **Paso 2: Transformación**:
  1. transform\_date: combina mes y año en una columna date datetime.
  2. filter\_years: filtra el rango de años deseado.
  3. clean\_cause\_route: imputa valores faltantes en causas y rutas.
* **Paso 3: Carga y reporte**:
  1. plot\_incidents\_by\_month: genera y guarda gráfico de incidentes por mes.
  2. analyze\_cause\_route + plot\_cause\_route: filtra por causa, cuenta rutas y grafica.
* **Mensaje final**:
  + print("ETL completado.") indica al usuario que todos los pasos se completaron sin errores.

**Buenas prácticas y recomendaciones**

* **Modularidad:** cada archivo realiza una sola tarea (Single Responsibility Principle).
* **Nombres claros:** funciones y variables autoexplicativas.
* **Manejo de errores:** errors='coerce', exist\_ok=True para evitar caídas.
* **Pruebas:** escribir tests unitarios para cada función usando pytest.
* **Configuración:** parametrizar rutas y filtros con un archivo config.yaml.

Material desarrollado por Lucas De Rito.