Proceso ETL con Detección Automática de Delimitador y Exportación a Excel

**1. Introducción**

Este documento describe un proceso ETL (Extract, Transform, Load) implementado en Python. El objetivo del proceso es:

* **Extraer:** Leer un archivo CSV, detectando automáticamente el delimitador y el encoding mediante la librería estándar csv.
* **Transformar:** Procesar los datos eliminando filas vacías y, en caso de existir, convertir la columna fecha a tipo datetime.
* **Cargar:** Exportar el DataFrame resultante a un archivo Excel (.xlsx) con autoajuste de ancho de columnas y formato de encabezados, utilizando la librería openpyxl.

**2. Requisitos**

* **Python 3.6+**
* Librerías necesarias:
  + csv (incluida en la biblioteca estándar de Python)
  + pandas
  + openpyxl
* Archivo CSV de entrada que contenga los datos a procesar.

**3. Descripción del Código**

El código se divide en cuatro partes principales, cada una encapsulada en una función:

**3.1. Función detect\_delimiter(file\_path, encodings=None)**

* **Objetivo:**  
  Detectar el delimitador y el encoding del archivo CSV.
* **Descripción:**
  + Intenta leer un fragmento del archivo (4096 caracteres) con distintos encodings: 'utf-8', 'latin1' e 'iso-8859-1'.
  + Utiliza csv.Sniffer().sniff(sample) para detectar el delimitador usado en el archivo.
  + Retorna una tupla con el delimitador detectado y el encoding utilizado, o (None, None) en caso de fallo.

**3.2. Función extract(file\_path)**

* **Objetivo:**  
  Extraer datos del archivo CSV usando el delimitador y encoding detectados (o valores por defecto si no se detectan).
* **Descripción:**
  + Llama a detect\_delimiter para obtener el separador y encoding.
  + Si la detección falla, utiliza valores predeterminados (por ejemplo, ',' y 'utf-8').
  + Lee el CSV usando pandas.read\_csv con parámetros como quotechar='"', skipinitialspace=True y on\_bad\_lines='skip' para mayor robustez.
  + Imprime información sobre el número de filas y columnas leídas.

**3.3. Función transform(df)**

* **Objetivo:**  
  Realizar transformaciones en el DataFrame extraído.
* **Descripción:**
  + Elimina filas completamente vacías utilizando dropna(how='all').
  + Si existe una columna llamada fecha, se convierte a tipo datetime usando pd.to\_datetime con el parámetro errors='coerce' para manejar conversiones fallidas.
  + Imprime mensajes informativos sobre las transformaciones realizadas.

**3.4. Función load\_to\_excel(df, output\_path)**

* **Objetivo:**  
  Exportar el DataFrame transformado a un archivo Excel (.xlsx) de forma presentable.
* **Descripción:**
  + Utiliza pd.ExcelWriter con el engine openpyxl para escribir el DataFrame en una hoja llamada "Datos".
  + Recorre cada columna para determinar la longitud máxima del contenido y ajusta el ancho de la columna en el archivo Excel.
  + Aplica un estilo al encabezado (texto en negrita con color blanco y fondo en color azul).
  + Imprime la ruta del archivo Excel generado.