**EXPLICACIÓN DEL SCRIPT**

El script hace **5 grandes tareas**:

**1. Configuración y Librerías**

import os

import pandas as pd

from sqlalchemy import create\_engine

from azure.storage.blob import BlobServiceClient

import logging

from datetime import datetime

Estas librerías permiten:

pandas: leer y transformar datos.

sqlalchemy: conectar con la base de datos SQL Anywhere.

azure.storage.blob: subir archivos a Azure Data Lake (Blob Storage).

logging: registrar eventos en un archivo de log.

**2. Parámetros del entorno**

SQL\_USER = 'DB\_USER'

SQL\_PASSWORD = 'DB\_PASS'

SQL\_HOST = 'localhost'

SQL\_PORT = '2638'

SQL\_DB = 'database\_name'

AZURE\_CONNECTION\_STRING = "DefaultEndpointsProtocol=https;AccountName=...;AccountKey=...;"

RAW\_CONTAINER = "raw"

CLEANED\_CONTAINER = "cleaned"

EXCEL\_FOLDER = "./excels"

Aquí se define:

Credenciales para conectar con **SQL Anywhere**.

La **cadena de conexión de Azure** (para subir los archivos).

Los **contenedores** en los que subiremos datos crudos y limpios.

La carpeta donde están los **archivos Excel**.

**3. Funciones auxiliares**

**a. Logs personalizados**

def log\_info(msg): logging.info(msg); print(f"ℹ️ {msg}")

def log\_error(msg): logging.error(msg); print(f"❌ {msg}")

Imprimen y guardan mensajes informativos o de error.

**b. Conexión con Azure Blob Storage**

def connect\_blob\_container(container\_name):

service = BlobServiceClient.from\_connection\_string(AZURE\_CONNECTION\_STRING)

return service.get\_container\_client(container\_name)

Se conecta a un contenedor de Azure (por ejemplo, raw o cleaned).

**c. Conexión SQL Anywhere**

def sqlanywhere\_engine():

return create\_engine(f'sqlalchemy\_sqlany://{SQL\_USER}:{SQL\_PASSWORD}@{SQL\_HOST}:{SQL\_PORT}/{SQL\_DB}')

se usa SQLAlchemy para generar una conexión al motor SQL Anywhere.

**4. Extracción, limpieza y transformación de datos**

**a. Desde SQL**

def extract\_sql\_data(query, nombre\_tabla):

...

df.to\_parquet(raw\_path, index=False)

df\_clean = df.dropna()

df\_clean.to\_parquet(clean\_path, index=False)

Ejecuta una consulta SQL.

Genera un archivo .parquet con los datos sin procesar.

Genera otro .parquet eliminando filas con valores nulos.

**b. Desde Excel**

def extract\_excel\_files(folder):

for file in os.listdir(folder):

...

df.to\_parquet(raw\_path, index=False)

df\_clean = df.dropna()

df\_clean.to\_parquet(clean\_path, index=False)

Recorre cada archivo Excel en la carpeta, y hace lo mismo: genera archivos .parquet crudos y limpios.

**5. Subida a Azure Blob Storage**

def upload\_to\_blob(local\_path, blob\_name, container):

with open(local\_path, "rb") as data:

container.upload\_blob(name=blob\_name, data=data, overwrite=True)

Abre el archivo local y lo sube al contenedor que se le indique.

**6. Función principal**

def main():

...

sql\_raw, sql\_clean = extract\_sql\_data("SELECT \* FROM tu\_tabla", "sql\_table")

...

raw\_excels, clean\_excels = extract\_excel\_files(EXCEL\_FOLDER)

Orquesta todo el proceso: se conecta, extrae datos, los transforma, y los sube al Data Lake.

**PROGRAMAR EJECUCIÓN AUTOMÁTICA**

En Windows (Tareas programadas)

Abre Tareas Programadas (Task Scheduler).

Crea una nueva tarea.

En *Acciones*, selecciona:

Programa: python

Argumentos: C:\ruta\al\script\script\_proceso.py

En *Desencadenadores*, selecciona "Diariamente" a la hora que desees.

**En Linux/macOS (cron)**

Abre terminal y ejecuta:

bash

CopiarEditar

crontab -e

Agrega esto al final para ejecutarlo todos los días a las 2 AM:

bash

0 2 \* \* \* /usr/bin/python3 /ruta/absoluta/script\_proceso.py >> /ruta/absoluta/cron\_output.log 2>&1