import os

import pandas as pd

import logging

import pyodbc

# Configuración de logging

logging.basicConfig(filename='ingesta.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Crear conexión con pyodbc

def create\_connection():

    try:

        conn = pyodbc.connect(

            r'DRIVER={ODBC Driver 17 for SQL Server};'

            r'SERVER=DESKTOP-57C01NF\SQLEXPRESS;'

            r'DATABASE=Demo1;'

            r'UID=usr\_Nico;'

            r'PWD=123456'

        )

        logging.info("Conexión exitosa a SQL Server.")

        return conn

    except Exception as e:

        logging.error(f"Error al conectar a SQL Server: {e}")

        return None

# Función para obtener IDs existentes en la tabla

def obtener\_ids\_existentes(conn, tabla\_sql):

    try:

        cursor = conn.cursor()

        query = f"SELECT aircraft\_code FROM {tabla\_sql}"

        cursor.execute(query)

        rows = cursor.fetchall()

        ids\_existentes = set(str(row[0]).strip() for row in rows)

        print(f"🔍 IDs en SQL: {list(ids\_existentes)[:11]}")

        return ids\_existentes

    except Exception as e:

        logging.error(f"Error al obtener IDs existentes: {str(e)}")

        return set()

# Función para detectar archivos CSV nuevos

def obtener\_csv\_nuevos(carpeta\_csv):

    return [f for f in os.listdir(carpeta\_csv) if f.endswith(".csv")]

# Función para insertar registros en la base de datos

def insertar\_datos(conn, df, tabla\_sql):

    try:

        cursor = conn.cursor()

        for \_, row in df.iterrows():

            query = f"INSERT INTO {tabla\_sql} (aircraft\_code, range, model\_en, model\_ru) VALUES (?, ?, ?, ?)" ##Especificar el nombre de todas las columnas

            cursor.execute(query, row['aircraft\_code'], row['range'], row['model\_en'], row['model\_ru']) ##Aqui tambien

        conn.commit()

        logging.info(f"Se insertaron {len(df)} nuevos registros en {tabla\_sql}.")

    except Exception as e:

        logging.error(f"Error al insertar datos en SQL: {str(e)}")

# Función para procesar y cargar archivos

def procesar\_archivo(archivo, conn, tabla\_sql, carpeta\_csv):

    ruta\_completa = os.path.join(carpeta\_csv, archivo)

    try:

        # Leer CSV

        df = pd.read\_csv(ruta\_completa)

        # Verificar columna clave

        if "aircraft\_code" not in df.columns:

            logging.warning(f"El archivo {archivo} no tiene la columna 'aircraft\_code'. Se omite.")

            return

        # Obtener IDs existentes

        ids\_existentes = obtener\_ids\_existentes(conn, tabla\_sql)

        # Filtrar registros nuevos

        df["aircraft\_code"] = df["aircraft\_code"].astype(str).str.strip()

        df\_nuevo = df[~df["aircraft\_code"].isin(ids\_existentes)]

        if not df\_nuevo.empty:

            insertar\_datos(conn, df\_nuevo, tabla\_sql)

            print(f"✅ Se agregaron {len(df\_nuevo)} nuevos registros desde {archivo}.")

        else:

            print(f"⚠️ Todos los registros de {archivo} ya existen en la base de datos.")

    except Exception as e:

        logging.error(f"Error procesando {archivo}: {str(e)}")

# Función principal para ejecutar la carga automática

def ejecutar\_ingesta():

    carpeta\_csv = "C:/Silver/Henry/Proyecto Final/prueba"  # Ajusta la carpeta según corresponda

    tabla\_sql = "aircrafts\_data"  # Nombre de la tabla en SQL

    archivos = obtener\_csv\_nuevos(carpeta\_csv)

    if not archivos:

        print("❌ No hay archivos nuevos para procesar.")

        return

    conn = create\_connection()

    if conn is None:

        print("❌ Error al conectar a la base de datos.")

        return

    for archivo in archivos:

        procesar\_archivo(archivo, conn, tabla\_sql, carpeta\_csv)

    conn.close()  # Cerrar la conexión

    print("🏁 Ingesta completada.")

# Ejecutar la ingesta

ejecutar\_ingesta()