**PYTHON cargar\_excel\_duplicado\_si-o-no.py**

SCRIPT (carga reporte de ventas de una gasolinera en archivo Excel)

Este script en Python sirve para leer archivos Excel con información de inventario de estaciones de servicio (gasolineras), y cargar esa información en una base de datos MySQL.

import pandas as pd

import mysql.connector

import os

# Configuración conexión a BD (base y tablas ya creadas)

DB\_CONFIG = {

    'host': 'localhost',

    'user': 'USUARIO',

    'password': 'CLAVE',

    'database': 'gasolinas1',

    'charset': 'utf8mb4',

    'collation': 'utf8mb4\_general\_ci'

}

# Ruta base con archivos Excel

RUTA\_BASE\_EXCEL = r'C:\Users\sistemas\Desktop\gasolinas'

def agregar\_estaciones(estaciones):

    """Inserta estaciones si no existen."""

    try:

        conn = mysql.connector.connect(\*\*DB\_CONFIG)

        cursor = conn.cursor()

        for estacion\_id, nombre, direccion, tipo in estaciones:

            cursor.execute("""

                INSERT IGNORE INTO estacion (id, nombre, direccion, tipoestacion)

                VALUES (%s, %s, %s, %s)

            """, (estacion\_id, nombre, direccion, tipo))

        conn.commit()

        cursor.close()

        conn.close()

        print("✅ Estaciones insertadas (si no existían).")

    except Exception as e:

        print(f"❌ Error insertando estaciones: {e}")

def obtener\_estaciones():

    """Obtiene estaciones (id, nombre) de la BD."""

    try:

        conn = mysql.connector.connect(\*\*DB\_CONFIG)

        cursor = conn.cursor()

        cursor.execute("SELECT id, nombre FROM estacion")

        estaciones = cursor.fetchall()

        cursor.close()

        conn.close()

        return estaciones

    except Exception as e:

        print(f"❌ Error obteniendo estaciones: {e}")

        return []

def cargar\_inventario\_estacion(estacion\_id, nombre\_estacion):

    archivo\_excel = os.path.join(RUTA\_BASE\_EXCEL, f"{estacion\_id}.xlsx")

    if not os.path.exists(archivo\_excel):

        print(f"⚠️ Archivo no encontrado para estación '{nombre\_estacion}': {archivo\_excel}")

        return False

    print(f"📊 Procesando estación: {nombre\_estacion} (ID: {estacion\_id})")

    try:

        conn = mysql.connector.connect(\*\*DB\_CONFIG)

        cursor = conn.cursor()

        xls = pd.ExcelFile(archivo\_excel)

        registros\_procesados = 0

        for nombre\_hoja in xls.sheet\_names:

            try:

                fecha = pd.to\_datetime(nombre\_hoja, dayfirst=True, errors='coerce')

                if pd.isna(fecha):

                    print(f"⚠️ Hoja '{nombre\_hoja}' no contiene una fecha válida.")

                    continue

                fecha = fecha.date()

                # Verificar si ya existen datos para esa estación y fecha

                cursor.execute("""

                    SELECT COUNT(\*) FROM inventario

                    WHERE estacionid = %s AND fecha = %s

                """, (estacion\_id, fecha))

                ya\_existe = cursor.fetchone()[0]

                if ya\_existe > 0:

                    confirmacion = input(f"⚠️ Ya existen registros para la estación '{nombre\_estacion}' en la fecha {fecha}. ¿Deseas sobrescribirlos? (s/n): ").strip().lower()

                    if confirmacion != 's':

                        print("⏭️  Se omitió esta hoja.")

                        continue

                    else:

                        cursor.execute("""

                            DELETE FROM inventario WHERE estacionid = %s AND fecha = %s

                        """, (estacion\_id, fecha))

                        print(f"🗑️ Registros anteriores eliminados para la fecha {fecha}.")

                # Cargar DataFrame

                df = pd.read\_excel(xls, sheet\_name=nombre\_hoja, skiprows=6, nrows=4, header=None)

                df = df.iloc[:, [0, 12, 13, 14]]

                df.columns = ['producto', 'cantidad', 'precio\_venta', 'venta\_soles']

                df = df.dropna(subset=['producto', 'cantidad', 'precio\_venta', 'venta\_soles'])

                df[['cantidad', 'precio\_venta', 'venta\_soles']] = df[['cantidad', 'precio\_venta', 'venta\_soles']].apply(pd.to\_numeric, errors='coerce')

                df = df.dropna()

                for \_, fila in df.iterrows():

                    producto = str(fila['producto']).strip()

                    cantidad = float(fila['cantidad'])

                    precio\_venta = float(fila['precio\_venta'])

                    venta\_soles = float(fila['venta\_soles'])

                    cursor.execute("SELECT id FROM productos WHERE nombre=%s", (producto,))

                    res = cursor.fetchone()

                    if res:

                        producto\_id = res[0]

                    else:

                        cursor.execute("INSERT INTO productos (nombre, descripcion) VALUES (%s, '')", (producto,))

                        producto\_id = cursor.lastrowid

                        print(f"    ➕ Producto nuevo insertado: {producto}")

                    cursor.execute("""

                        INSERT INTO inventario (productoid, estacionid, fecha, cantidad, precio\_venta, venta\_soles)

                        VALUES (%s, %s, %s, %s, %s, %s)

                    """, (producto\_id, estacion\_id, fecha, cantidad, precio\_venta, venta\_soles))

                    registros\_procesados += 1

            except Exception as e:

                print(f"    ❌ Error procesando hoja '{nombre\_hoja}': {e}")

                continue

        conn.commit()

        cursor.close()

        conn.close()

        print(f"  ✅ Estación procesada. Registros cargados: {registros\_procesados}")

        return True

    except Exception as e:

        print(f"❌ Error procesando estación '{nombre\_estacion}': {e}")

        return False

def procesar\_todas\_las\_estaciones():

    estaciones = obtener\_estaciones()

    if not estaciones:

        print("❌ No hay estaciones para procesar.")

        return

    print(f"📋 Se procesarán {len(estaciones)} estaciones...")

    exitosas = 0

    fallidas = 0

    for estacion\_id, nombre\_estacion in estaciones:

        print("\n"+"="\*40)

        if cargar\_inventario\_estacion(estacion\_id, nombre\_estacion):

            exitosas += 1

        else:

            fallidas += 1

    print("\n"+"="\*40)

    print(f"✅ Procesadas con éxito: {exitosas}")

    print(f"❌ Fallidas: {fallidas}")

if \_\_name\_\_ == "\_\_main\_\_":

    # Si quieres agregar estaciones nuevas, ponlas aquí:

    estaciones\_ejemplo = [

        (1, 'La Rinconada', 'Av. Principal s/n', 'lima'),

        (2, 'America Soler', 'Av. Bolognesi s/n', 'trujillo'),

        (3, 'El Porvenir', 'Av. Porvenir s/n', 'trujillo')

    ]

    agregar\_estaciones(estaciones\_ejemplo)

    procesar\_todas\_las\_estaciones()

¿Qué hace este script?

* **Conecta con una base de datos MySQL** (gasolinas1)
* **Agrega estaciones de servicio** si no existen en la base de datos.
* **Busca archivos Excel por estación** (uno por cada estación, con nombre id.xlsx)
* **Lee cada hoja del archivo como una fecha de inventario**, verifica si ya existen registros para esa fecha, y si el usuario lo permite, los sobrescribe.
* **Inserta los datos leídos en la tabla inventario**, y si un producto no existe, lo agrega a la tabla productos.

Estructura esperada

Carpeta C:\Users\sistemas\Desktop\gasolinas con archivos como:

* 1.xlsx → datos de la estación con ID 1
* 3.xlsx → datos de la estación con ID 3
* 4.xlsx → datos de la estación con ID 4

Cada archivo contiene hojas con nombres de fechas (por ejemplo: "01-02-2025").

Cada hoja tiene datos de productos a partir de la fila 7, por 4 filas.

Tablas esperadas en la base de datos

* Estación

id, nombre, dirección, tipo estación

* Productos

id, nombre , descripción

* Inventario

Id, productoid, estacionid, fecha, cantidad, precio\_venta, venta\_soles

**Explicación de funciones:**

**Agregar\_estaciones(estaciones)**

* Agrega nueva estaciones si no existe (con INSERT IGNORE para evitar duplicados)

**Obtener\_estaciones ()**

* Obtiene de la base de datos los id y nombre de estaciones registrados

**Cargar \_inventario \_estacion (estacion\_id, nombre\_estacion)**

* Procesa un archivo Excel de una estacion especificada
* Verifica si existe el archivo x.xlsx
* Por cada hoja que representa una fecha:
* Valida si ya hay registros en inventario
* Si existen pregunta al usuario si quiere sobrescribir

Lee los productos desde la fila 7 (skiprows=6, nrows=4), y extrae columnas

* Col 0: nombre producto
* Col 12, 13, 14: cantidad, precio, ventas

Si el producto no existe, lo inserta

Inserta los datos en inventario

**Procesar\_toda \_las\_estaciones ()**

Procesa todas las estaciones que están en la base de datos.

* Llama a cargar\_inventario\_estacion () para cada una.
* Muestra resumen de éxito o fallos al final.

**Bloque principal if \_\_name\_\_ == "\_\_main\_\_":**

* Inserta 3 estaciones de ejemplo
* Llama a procesar\_toda\_las\_estaciones () para importar todo el Excel.

**Como se usa**

* Coloca los archivos 1.xlsx, 3.xlsx, etc. En la carpeta indicada
* Asegúrate que las hojas se llamen como fechas (dd-mm-yyyy).
* Ejecuta el script

Python cargar\_excel\_gasolinas.py o el nombre que lo puso

* El script, inserta estaciones, carga los inventarios desde los Excel, pregunta si debe sobrescribirlo si ya existe datos.

**Ventajas del script**

* Automatiza el ingreso de datos de inventario.
* Evita duplicados.
* Permite control manual (confirmación antes de sobrescribir).
* Añade productos automáticamente si no existen.