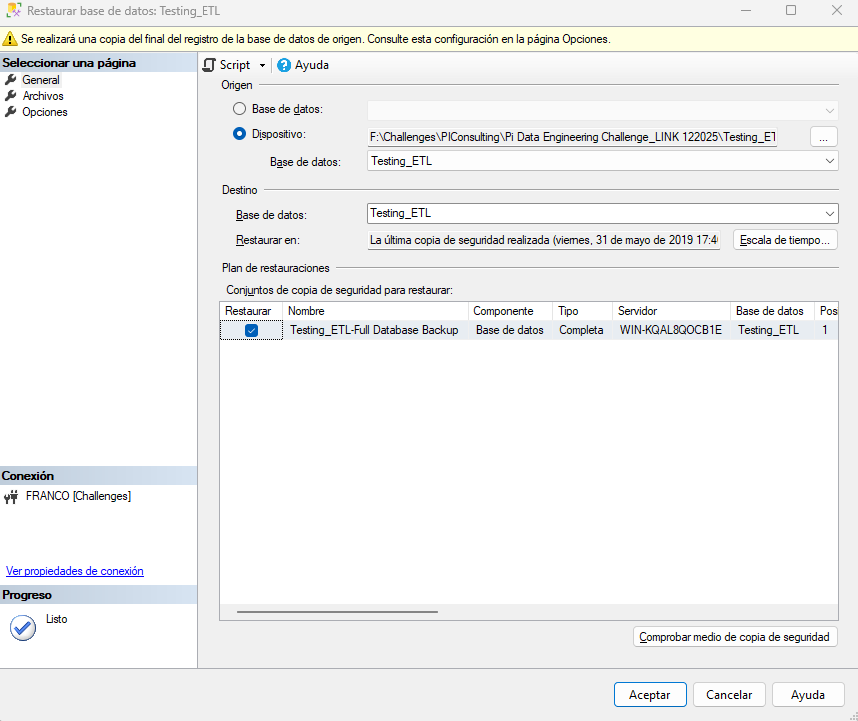
A-Montar el backup de la base de datos (se envía en .bak y .bacpac)

Se realizó el montaje del backup de la siguiente manera:

Bases de datos → Restaurar Base de datos →Dispositivo (buscamos el archivo .bak)→seleccionar la casilla restaurar → Aceptar



B-Descargar programáticamente el archivo csv con el link de abajo. Hacelo teniendo en cuenta que este archivo en el origen cambia semana a semana con datos nuevos para integrar a la base de datos.

Se generó un método para obtener el archivo desde el link el cual realiza la consulta y dependiendo del código de estado que devuelva la consulta se realiza o no la acción.

def obtener\_archivo(url,ruta):

response= requests.get(url)

if response.status\_code == 200:

with open(ruta,"wb") as documento:

documento.write(response.content)

print(f'Archivo descargado con exito y guardado en: {ruta}')

return None

else:

print(f'Error al descargar el archivo, codigo del error : {response.status\_code}')

return None

El siguiente bloque de código se genera para guardar el archivo segun los parametros solicitados, pensando en que el mismo se obtendrá una vez por semana, quedando el archivo en /Documentos/2025/7/Nuevos\_registros\_25-07-02.csv

fecha\_hoy= datetime.now().strftime("%y-%m-%d")

anio= datetime.now().year

mes= datetime.now().month

nombre\_csv= f"Nuevos\_registros\_{fecha\_hoy}.csv"

ruta= os.path.join(os.getcwd(), f'Documentos\\{anio}\\{mes}')

ruta\_final= os.path.join(ruta,nombre\_csv)

os.makedirs(ruta,exist\_ok=True)

obtener\_archivo(url\_csv,ruta\_final)

d. Solucionar el problema de registros duplicados en la tabla final considerando que no vuelva a ocurrir en el futuro. (La tabla final puede ser la misma original o una nueva que incluya la historia si se utilizan tecnologías cloud).

* A fin de poder probar y demostrar la diferencia de registros se creó la tabla [dbo].[Unificado\_Test]

Se generó un script para ejecutar por única vez previo a la carga de los datos

WITH duplicados AS(

SELECT

ID,

MUESTRA,

RESULTADO,

FECHA\_COPIA,

ROW\_NUMBER() OVER (PARTITION BY ID, MUESTRA, RESULTADO ORDER BY FECHA\_COPIA DESC) as cant

FROM [dbo].[Unificado\_Test]

)

DELETE FROM [dbo].[Unificado\_Test]

WHERE EXISTS(

SELECT 1

FROM duplicados

WHERE duplicados.ID = [dbo].[Unificado\_Test].ID

AND duplicados.MUESTRA =[dbo].[Unificado\_Test].MUESTRA

AND duplicados.RESULTADO = [dbo].[Unificado\_Test].RESULTADO

AND duplicados.cant > 1

AND duplicados.FECHA\_COPIA = [dbo].[Unificado\_Test].FECHA\_COPIA

);

Con este script se soluciono el problema de duplicados de la tabla

[dbo].[Unificado] (sin limpieza de duplicados)



[dbo].[Unificado\_Test](post ejecución del script y previo a la carga de los próximos registros)



Para evitar futuros problemas con duplicados en la tabla se generó la siguiente función

La función toma la ruta donde se encuentra el archivo origen por parámetro, luego genera un DataFrame de pandas y cumpliendo con parte de lo solicitado en el punto C (Completar el campo FECHA\_COPIA que viene vacío) agrega la fecha en la que fue cargado el archivo(fecha hoy con formato y-m-d) elimina los duplicados existentes siguiendo la regla de que estos son los que tenga los campos ID,MUESTRA y RESULTADO iguales, finalmente devolvemos el DataFrame limpio sin duplicados y contamos los registros del mismo para utilizarlos más adelante.

def limpieza\_datos(ruta):

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

df\_csv['FECHA\_COPIA'] = datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]

df\_final= df\_csv.drop\_duplicates(subset=['ID','MUESTRA','RESULTADO'], keep='last')

registros=df\_final.shape[0]

return df\_final,registros

**C. Desarrollar un proceso que inserte las filas del archivo .csv en la tabla Unificado. Tener en cuenta que la columna FECHA\_COPIA está vacía en el archivo, y hay que agregarle la fecha en la cual estas insertando las nuevas filas a la base de datos.**

Como se explicó en el punto anterior el paso de completar los datos de la columna FECHA\_ COPIA se realizan en la función

def limpieza\_datos(ruta):

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

df\_csv['FECHA\_COPIA'] = datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]

df\_final= df\_csv.drop\_duplicates(subset=['ID','MUESTRA','RESULTADO'], keep='last')

registros=df\_final.shape[0]

return df\_final,registros

Se creó Conexion\_DB.py

Contiene simplemente la creación de una clase ConexionDB que tiene 2 funciones conectar donde realizamos la conexión a la DB utilizando la libreria pyodbc y cerrar para cerrar la misma.

from dotenv import load\_dotenv

import os

import pyodbc

load\_dotenv()

class ConexionDB:

def \_\_init\_\_(self):

self.server = os.getenv("SQL\_SERVER")

self.db = os.getenv("SQL\_DATABASE")

self.user = os.getenv("SQL\_USER")

self.password = os.getenv("SQL\_PASSWORD")

self.conn = None

def conectar(self):

try:

self.conn = pyodbc.connect(

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

f'SERVER={self.server};'

f'DATABASE={self.db};'

f'UID={self.user};'

f'PWD={self.password}'

)

print("Conexion exitosa")

return self.conn

except pyodbc.Error as e:

print("Error al conectar a la base de datos:", e)

return None

def cerrar(self):

if self.conn:

self.conn.close()

Con el siguiente bloque de código realiza la conexión a la DB y ejecuta las Query de carga.

Se pensó esta de la siguiente manera, para cubrir los siguientes puntos solicitados, realizar una consulta previa a la tabla para ver cuántos registro tiene en ese momento luego crear una tabla temporal para guardar las el tipo de carga que se va a realizar y por último recorrer el Dataframe previamente limpiado de duplicados para realizar la carga final a la tabla, teniendo en cuenta los campos ID, MUESTRA Y RESULTADO realizar la carga o actualización de los registros tomando como premisa que si los 3 coinciden este será un registro a actualizar y si no uno nuevo. Esto también forma parte de lo realizado para evitar futuros problemas con registros duplicados en la tabla final.

conexion = ConexionDB()

conn = conexion.conectar()

cursor = conn.cursor()

cursor.execute("SELECT COUNT(\*) FROM [Testing\_ETL].[dbo].[Unificado\_test];")

total\_registros= cursor.fetchone()[0]

cursor.execute("""

IF NOT EXISTS (SELECT \* FROM tempdb.sys.tables WHERE name = 'Accion')

BEGIN

CREATE TABLE #Accion (

Tipo NVARCHAR(10),

ID NVARCHAR(50),

MUESTRA NVARCHAR(50),

RESULTADO NVARCHAR(50)

);

END;

""")

for index, row in df.iterrows():

set\_values = ', '.join([f"{col} = '{row[col]}'" if isinstance(row[col], str) else f"{col} = {row[col]}" for col in df.columns])

query = f"""

MERGE INTO [dbo].[Unificado\_Test] AS target

USING (SELECT '{row['ID']}' AS ID, '{row['MUESTRA']}' AS MUESTRA, '{row['RESULTADO']}' AS RESULTADO) AS source

ON target.ID = source.ID AND target.MUESTRA = source.MUESTRA AND target.RESULTADO = source.RESULTADO

WHEN MATCHED THEN

UPDATE SET {set\_values}

WHEN NOT MATCHED THEN

INSERT ({', '.join(df.columns)}) VALUES ({', '.join([f"'{x}'" if isinstance(x, str) else str(x) for x in row])})

OUTPUT $action AS Tipo, inserted.ID, inserted.MUESTRA, inserted.RESULTADO INTO #Accion;

"""

cursor.execute(query)

g.Mejorar el proceso para que guarde logs en cada ejecución con la información que creas necesaria (cantidad de filas afectadas, fecha del proceso, instancia de base de datos, etc).

Para cumplir con este punto se fueron mencionando algunas de las cosas que se realizaron para completar este punto.

* Se agregó un contador de registros del archivo csv luego de realizar la limpieza de duplicados del mismo
* Luego se creó la tabla temporal #Acción que almacena el tipo de carga que se realiza ya sea insert o update al momento para su posterior consulta.

Se creó la tabla

CREATE TABLE dbo.logs\_Unificado\_Test(

ID int IDENTITY(1,1) NOT NULL,

Archivo varchar(255) NULL,

Registro\_Tabla int NULL,

Registros\_Archivo int NULL,

Insertados int NULL,

Actualizados int NULL,

Fecha\_Inicio\_Proceso DATETIME NULL,

Fecha\_Fin\_Proceso DATETIME NULL,

Usuario varchar(255) NULL

)

Se realiza la consulta a la tabla #Accion para guardar la cantidad de tipo de grabados que se realizaron

cursor.execute("SELECT COUNT(\*) FROM #Accion WHERE Tipo = 'INSERT';")

insertados = cursor.fetchone()[0]

cursor.execute("SELECT COUNT(\*) FROM #Accion WHERE Tipo = 'UPDATE';")

Actualizados = cursor.fetchone()[0]

y por último cargamos la tabla de logs

querylog = f"""

INSERT INTO [dbo].[logs\_Unificado\_Test] (Archivo, Registro\_Tabla, Registros\_Archivo,Insertados,Actualizados,Fecha\_Inicio\_Proceso,Fecha\_Fin\_Proceso,usuario)

VALUES (?,?,?,?,?,?,?,?)

"""

cursor.execute(querylog, nombre\_csv, total\_registros, Registros\_Archivo,insertados,Actualizados,fecha\_inicio\_Proceso,fecha\_Fin\_Proceso,usuario)

Finalmente la tabla cargada con los nuevos registros

[dbo].[Unificado\_Test]



Un vistazo de la tabla final

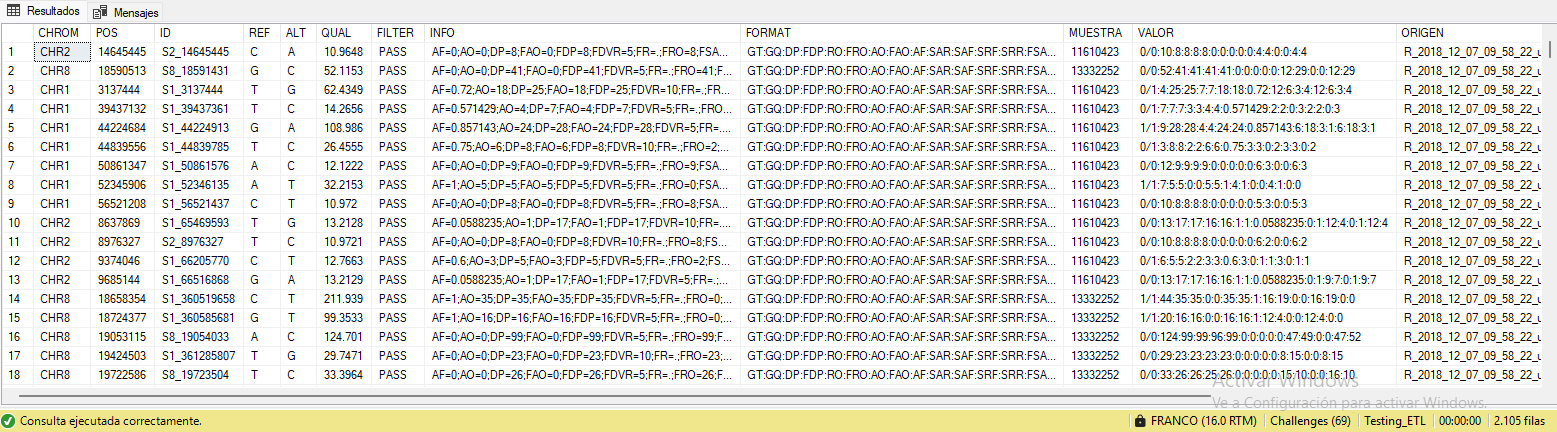
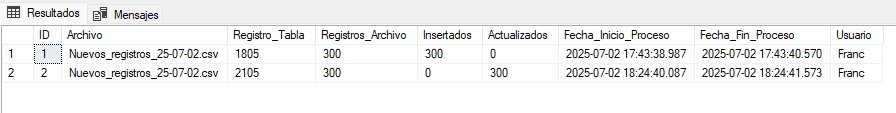


Tabla log

f. Dejar de algún modo programado ese proceso para que se ejecute los lunes de cada semana, a las 5:00 AM.

Se utilizó el programador de tareas de windows para cumplir con este punto

