In [1]:
```
from sqlalchemy import create_engine
import urllib
import pyodbc
import pandas as pd
import os
import io
import time
```

In [ ]:
```
```

In [ ]:
```
#Valores a omitir en el read_csv y posterioremente convertidos a None
valores_nulos = ['','0','SIN LÍNEA']

#Columnas a descargar y tratar en SQL
columnas_utiles = [
    'anho',
    'mes',
    'nivel',
    'descripcion_nivel',
    'entidad',
    'descripcion_entidad',
    'oee',
    'descripcion_oee',
    'documento',
    'nombres',
    'apellidos',
    'funcion',
    'estado',
    'carga_horaria',
    'anho_ingreso',
    'sexo',
    'discapacidad',
    'tipo_discapacidad',
    'concepto',
    'linea',
    'cargo',
    'presupuestado',
    'devengado',
    'movimiento',
    'lugar',
    'fecha_nacimiento',
    'fec_ult_modif',
    'fecha_acto',
    'correo',
    'profesion',
    'motivo_movimiento'
]

# Tipo de datos de las columnas utiles (deben de coincidir con la variable de colum
data_types={
    'anho': int, #int
    'mes': int, #int
    'nivel': str, #int
    'descripcion_nivel': str,
    'entidad': str, #int
    'descripcion_entidad': str,
    'oee': str, #int
    'descripcion_oee': str,
    'documento': str,
    'nombres': str,
    'apellidos': str,
```

```
    'funcion': str,
    'estado': str,
    'carga_horaria': str,
    'anho_ingreso': str, #int
    'sexo': str,
    'discapacidad': str,
    'tipo_discapacidad': str,
    'fuente_financiamiento': str, #int
    'objeto_gasto': str, #int
    'concepto': str,
    'linea': str,
    'categoria': str,
    'cargo': str,
    'presupuestado': str, #int
    'devengado': str, #int
    'movimiento': str,
    'lugar': str,
    'fecha_nacimiento': str,
    'fec_ult_modif': str,
    'uri': str,
    'fecha_acto': str,
    'correo': str,
    'profesion': str,
    'motivo_movimiento': str}

#Variables para conexiones de sql
quoted = urllib.parse.quote_plus('DRIVER={SQL Server Native Client 11.0};'+
                                 'SERVER=LAPTOP-0O524DL1;'+
                                 'DATABASE=Python_Projects;'+
                                 'fast_executemany=True;'+
                                 'Trusted_Connection=yes')
engine = create_engine('mssql+pyodbc:///?odbc_connect={}'.format(quoted))

conn = pyodbc.connect('DRIVER={SQL Server Native Client 11.0};' +
                      'SERVER=LAPTOP-0O524DL1;' +
                      'DATABASE=Python_Projects;' +
                      'fast_executemany=True;'+
                      'Trusted_Connection=yes;')

#Archivos a ser exportados
ruta_destino = 'C:\\Users\\HUAWEI\\Desktop\\Python_Projects\\WebScrapping_data\\Sec
archivos_csv = [archivo for archivo in os.listdir(ruta_destino) if archivo.endswith
```

In [3]:

```
def load_csv_insert_sql(chunksize):
    cursor = conn.cursor()
    for archivo_csv in archivos_csv:

        vuelta = 0
        tiempo_vuelta_for = 0
        nombre_tabla = os.path.splitext(archivo_csv)[0]

        with open(os.path.join(ruta_destino, archivo_csv), errors='ignore') as file
            out_file = io.StringIO()
            out_file.write(file.read().replace('"', '$'))
            out_file.seek(0)
            for df in pd.read_csv(out_file,
                                  usecols=columnas_utiles,
                                  dtype=data_types,
                                  encoding= 'utf-8',
                                  delimiter=',',
                                  quotechar='$',
                                  low_memory=False,
                                  chunksize=chunksize,
                                  index_col=False,
```

```
                            keep_default_na=False,
                            na_values=valores_nulos):

                inicio_vuelta_for = time.time()
                df = df.astype(str)
                df.replace('nan', None, inplace=True)

                nombre_columnas = ', '.join(df.columns)
                question_marks = ', '.join(['?']*len(df.columns))

                query = (f'INSERT INTO {nombre_tabla} ({nombre_columnas}) values({q
                values = [tuple(row) for row in df.itertuples(index=False)]

                cursor.executemany(query, values)
                conn.commit()

                fin_vuelta_for = time.time()
                tiempo_vuelta_for = fin_vuelta_for - inicio_vuelta_for
                vuelta +=1
                print(f'Vuelta n° {vuelta} duró {tiempo_vuelta_for} segundos. Archi
                print(f'Memoria del df (chunk) insertado: {(df.memory_usage().sum()
        os.remove(os.path.join(ruta_destino, archivo_csv))
    cursor.close()
```

In [4]:

```
def create_tables_csv_in_sql():
    cursor = conn.cursor()
    for archivo_csv in archivos_csv:

        nombre_tabla = os.path.splitext(archivo_csv)[0]

        with open(os.path.join(ruta_destino, archivo_csv), errors='ignore') as file
            out_file = io.StringIO()
            out_file.write(file.read().replace('"', '$')) # Bloque with implementad
            out_file.seek(0)
            df = pd.read_csv(out_file,
                             nrows=1,
                             usecols=columnas_utiles,
                             dtype=data_types,
                             encoding= 'utf-8',
                             delimiter=',',
                             quotechar='$',
                             low_memory=True,
                             index_col=False)

            columnas = []
            for col_name, dtype in zip(df.columns, df.dtypes):
                if dtype == 'int64':
                    sql_type = "INT"
                elif dtype == 'float64':
                    sql_type = "FLOAT"
                else: sql_type = "VARCHAR(MAX)" #Actualmente todas las columnas ser

                columnas.append(f"{col_name} {sql_type}")

            cursor = conn.cursor()
            cursor.execute(f"DROP TABLE IF EXISTS {nombre_tabla}; CREATE TABLE {nom
            conn.commit()
            print(f'La tabla {nombre_tabla} fue creada exitosamente.')
    cursor.close()
```

---------- UTILIZACION DE TO_SQL EN VEZ DE INSERT (PERFORMANCE MAS LENTA SEGUN PRUEBAS, APROX 5 MINUTOS POR ITERACION DE 50K FILAS) ----------

```
---------- def load_csv_to_sql(chunksize): ruta_destino = 'C:\\Users\\HUAWEI\\Desktop\\Python_Projects\\WebScrapping_data\\Secretaria_Funcion_Publica_PY\\' out_file = io.StringIO() quoted = urllib.parse.quote_plus('DRIVER={SQL Server Native Client 11.0};'+ #DRIVER={SQL Server Native Client 11.0}; 'SERVER=LAPTOP-0O524DL1;'+ 'DATABASE=Python_Projects;'+ 'fast_executemany=True;'+ 'Trusted_Connection=yes') engine = create_engine('mssql+pyodbc:///?odbc_connect={}'.format(quoted)) archivos_csv = [archivo for archivo in os.listdir(ruta_destino) if archivo.endswith('.csv')] vuelta = 0 tiempo_vuelta_for = 0 for archivo_csv in archivos_csv: nombre_tabla = os.path.splitext(archivo_csv)[0] with open(os.path.join(ruta_destino, archivo_csv), errors='ignore') as file: out_file.write(file.read().replace('"', '$')) out_file.seek(0) for df in pd.read_csv(out_file, encoding= 'utf-8', low_memory=True, delimiter=',', quotechar='$', chunksize=chunksize, index_col=False, dtype={ 'anho': str, #int 'mes': str, #int 'nivel': str, #int 'descripcion_nivel': str, 'entidad': str, #int 'descripcion_entidad': str, 'oee': str, #int 'descripcion_oee': str, 'documento': str, 'nombres': str, 'apellidos': str, 'funcion': str, 'estado': str, 'carga_horaria': str, 'anho_ingreso': str, #int 'sexo': str, 'discapacidad': str, 'tipo_discapacidad': str, 'fuente_financiamiento': str, #int 'objeto_gasto': str, #int 'concepto': str, 'linea': str, 'categoria': str, 'cargo': str, 'presupuestado': str, #int 'devengado': str, #int 'movimiento': str, 'lugar': str, 'fecha_nacimiento': str, 'fec_ult_modif': str, 'uri': str, 'fecha_acto': str, 'correo': str, 'profesion': str, 'motivo_movimiento': str}): inicio_vuelta_for = time.time() df = df.astype(str) #print(f'Memoria total del chunk del df {(df.memory_usage().sum())/1000000} megabytes.') df.to_sql(nombre_tabla, schema='dbo', con = engine, if_exists='replace', index=False) fin_vuelta_for = time.time() tiempo_vuelta_for = fin_vuelta_for - inicio_vuelta_for vuelta +=1 print(f'Vuelta n° {vuelta} duró {tiempo_vuelta_for} segundos. Archivo insertado: {nombre_tabla}. Chunksize {chunksize}.')
```