**Proyecto**

**base\_historicas\_lotus\_txt**

**Objetivo:**

Subir muchos archivos .txt a una base de datos. Debe haber una sola tabla con todos los .txt unidos

**Requisitos:**

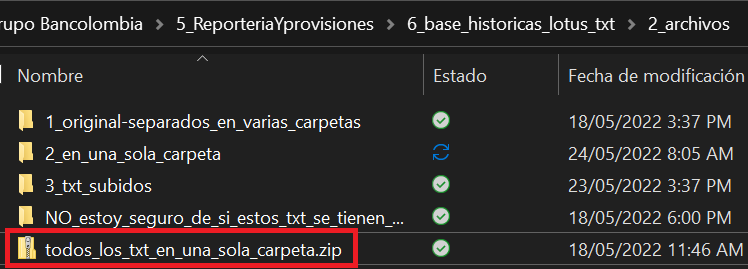
**1)** Personas que me asignaron la tarea:

- Santiago González Gil

- Rubén Alonso Guevara Jaramillo

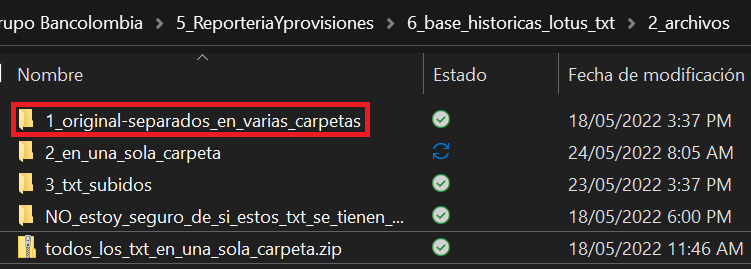
**2)** Hacer el proyecto en Python y ejecutar las consultas SQL desde Python.

**3)** El Python solo se va a ejecutar una sola vez para los archivos .txt que me pasaron en la carpeta comprimida:

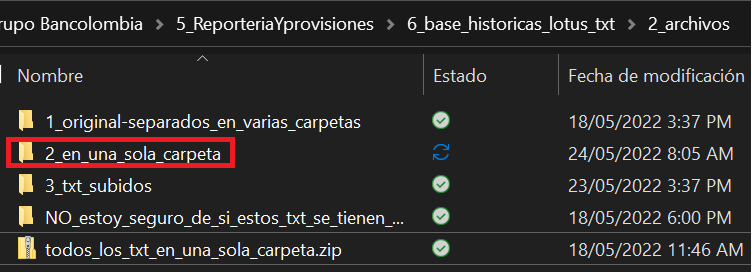


NO se seguirá ejecutando en el futuro. Por lo que no hace falta que el código sea tan “full automático”.

**4)** Pasar MANUALMENTE los archivos que están en muchas carpetas en ...\2\_archivos\1\_original-separados\_en\_varias\_carpetas



a una sola carpeta en ...\2\_archivos\2\_en\_una\_sola\_carpeta



**5)** Los nombres de los archivos .txt son:

- Actual NombreMes.txt = Toda la Base1.txt --> subir solo uno por que son los mismos

- Histórico NombreMes.txt = Toda la Base2.txt --> “ ”

- aval... --> este NO hay q subirlo a LZ

- Consolidado... .xlsx --> “ ”

- comentarios... --> “ ”

- monto... --> “ ”

**6)** Todas las tablas de los .txt tienen la misma estructura y los siguientes nombres de columnas:

Radicado

Identificación del cliente

Tipo de identificación

Nombre del cliente

Región

Zona

Segmento

Sector

Actividad económica

Código CIIU

Gerente

Código del gerente

Centro de costos

Grupo de riesgo

Código de riesgo

Calificación Superbancaria

Calificación Interna actual

Fecha de creación

Autor

Finalidad del crédito

Fecha de decisión

Nombre de quien aprueba

Código de quien aprueba

Número acta de comité de crédito

LME solicitado

LME PIC

LME aprobado

Flujo Actual

Estado Actual

Responsable actual

Flujo

Estado

Responsable

Fecha de entrada

Hora de entrada

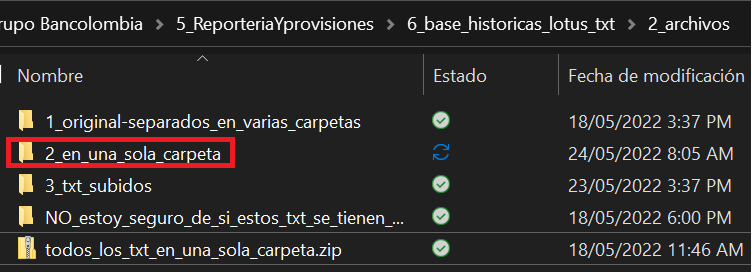
Fecha de salida

Hora de salida

Tiempo

Vigencia LME

**7)** Leer los N .txt q están en ...\2\_archivos\2\_en\_una\_sola\_carpeta así:



<https://www.geeksforgeeks.org/how-to-read-text-files-with-pandas/>

df = pd.read\_csv(f,

encoding='latin-1',

sep='^

)

**8)** En los nombres de las columnas hacer lo siguiente:

**8.1)** Eliminar los espacios en blanco al principio y al final

**8.2)** Reemplazar los espacios por guion bajo \_

**8.3)** Quitar las tildes de las vocales

**9)** Hacer que TODOS los datos de la tabla sean tipo string.

Convertir TODAS las columnas que tengan fecha a tipo string, esto se hace para que las fechas queden tal cual como están en los .txt y no modifique el formato

**10)** Crear una tabla vacia en LZ - HUE (base de datos de Bancolombia) con los nombres de las columnas del paso **6)**

La tabla se llama proceso\_consumidores.base\_historicas\_lotus\_txt

**11)** Juntar todos los .txt (UNION ALL) para que queden en una sola tabla de LZ, esto se hace con SQL:

INSERT INTO NombreTabla.NombreBaseDeDatos

VALUES (valor1, valor2, valor3, ... valorN);

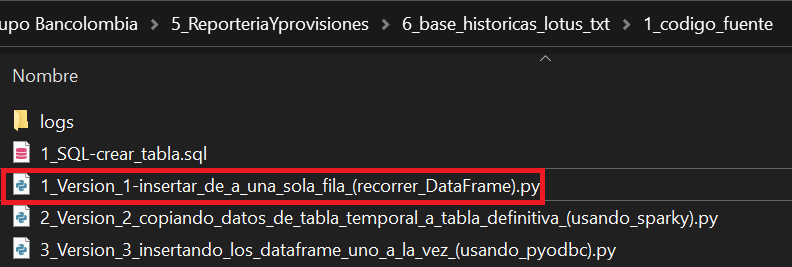
**12)** Descargar en Excel la tabla resultante (final) proceso\_consumidores.base\_historicas\_lotus\_txt

**Formas de Resolver el Problema:**

**Forma 1:**

\* Insertar de a una sola fila (fila por fila) recorriendo las filas del DataFrame

- El .py con esta forma 1 es:



- El problema de esta forma 1 es:

\* Solamente puedo hacer 720 peticiones al servidor, esto significa que solo puedo insertar 720 filas y son demasiadas filas, son mucho mas de 720 por lo que al hacer esto no estaría subiendo todos los datos.

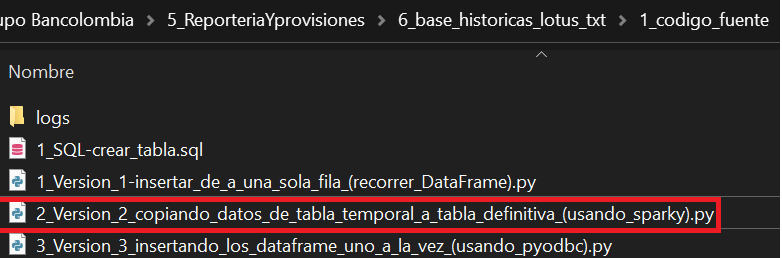
\* Hacer peticiones al servidor hace que el código se demore MUCHO en ejecutarse (se vuelve muy lento), la culpa no es del Python en sí, la culpa es del servidor de la base de datos de Bancolombia (LZ - HUE) que es lento, por lo tanto, cuantas menos peticiones haga al servidor, mejor:

<https://hue.bancolombia.corp:8889/hue/accounts/login?next=/hue/>

**Forma 2:**

\* Copiar los datos (registros) de una tabla temporal a una tabla definitiva.

- El .py con esta forma 2 es:

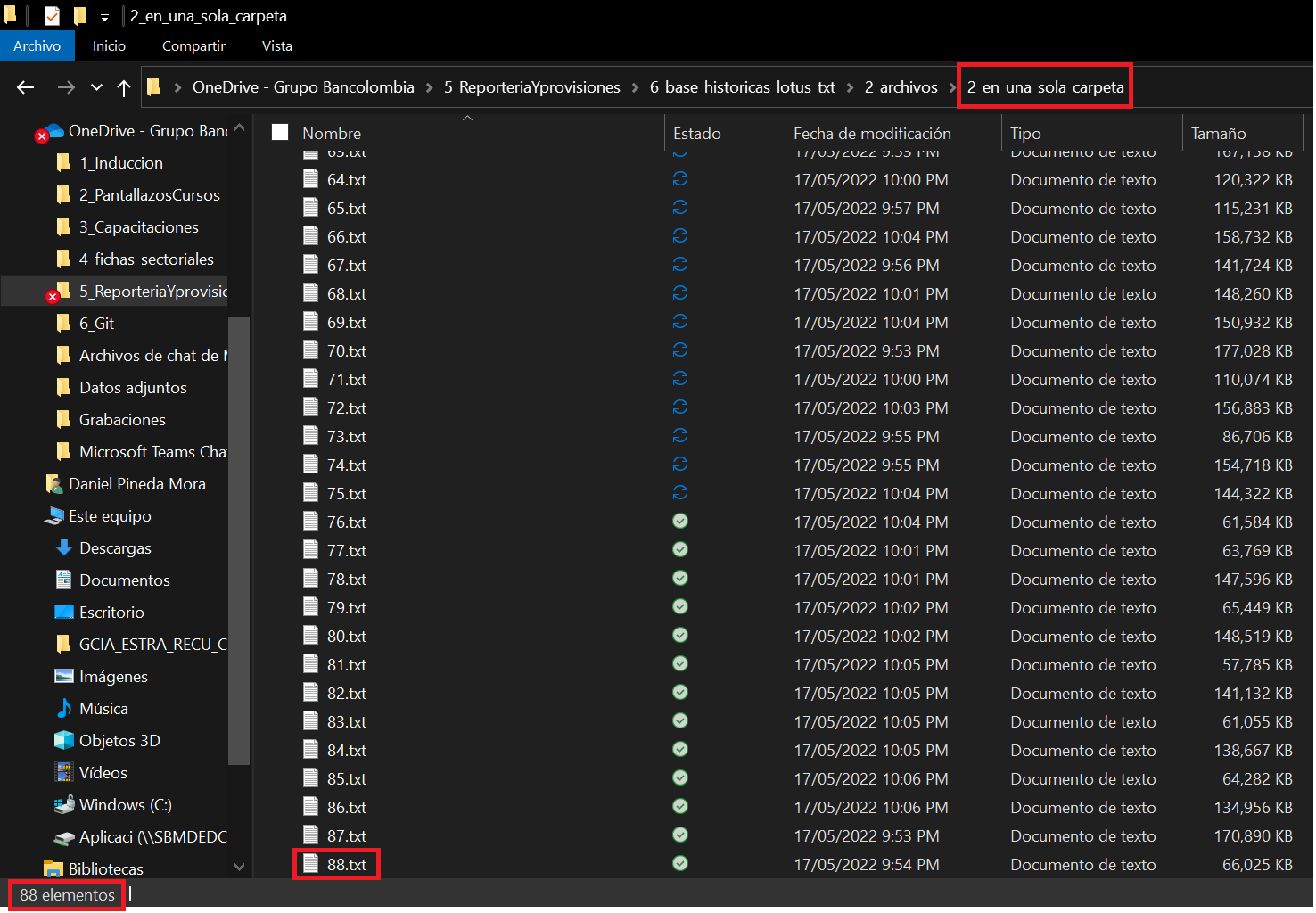


- El problema de esta forma 2 es:

\* Esta es la forma que finalmente fue usada para resolver el problema.

\* Copiar los datos de una tabla temporal a una tabla definitiva es lento porque son muchos datos:

Son 88 .txt (DataFrame)



entonces para calcular cuantas consultas en total se hace lo siguiente:

(88 INSERT a tabla temporal desde el Python) + (88 copiar lo q hay en tabla temporal a definitiva, esto se hace en la base de datos de Bancolombia) = (176 consultas ejecutadas en la base de datos)

- Los nombres de las tablas son:

|  |  |
| --- | --- |
| **Sintaxis** | NombreBaseDeDatos.NombreTabla |
| **Tabla Temporal** | proceso\_consumidores.base\_historicas\_lotus\_txt\_tmp |
| **Tabla Definitiva** | proceso\_consumidores.base\_historicas\_lotus\_txt\_definitivo |

- El paso a paso para hacer esta forma 2 es:

**1.** Eliminar tabla definitivo, la tabla definitivo se elimina una sola vez:

DROP TABLE IF EXISTS proceso\_consumidores.base\_historicas\_lotus\_txt\_definitivo PURGE

;

**2.** Crear tabla definitivo, la tabla definitivo se crea una sola vez, todas las columnas son tipo string, esto se hace así para no tener errores con los tipos de datos y que la fecha conserve el formato que está en los .txt:

CREATE TABLE IF NOT EXISTS proceso\_consumidores.base\_historicas\_lotus\_txt\_definitivo (

columna\_1 STRING,

columna\_2 STRING,

columna\_3 STRING,

...

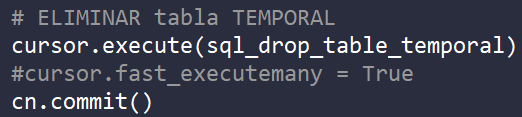
columna\_n STRING

)

;

**3.** Leer los N .txt

**4.** Eliminar tabla temporal usando cn

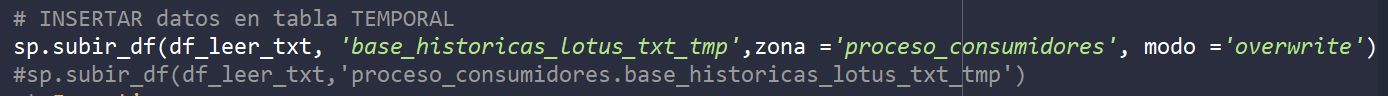


DROP TABLE IF EXISTS proceso\_consumidores.base\_historicas\_lotus\_txt\_tmp PURGE

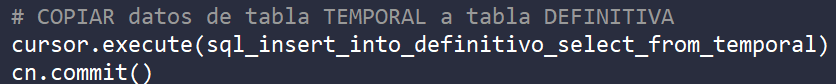
;

**5.** Crear tabla temporal CREATE TABLE... temporal

**6.** Usando sp\_subir\_df(...) subir datos a tabla temporal:



**7.** Copiar datos de temporal a definitivo usando cn



Esto se hace con un INSERT INTO **+** SELECT y se tiene que escribir CAST(columna\_N AS STRING) para no tener problemas con los tipos de datos:

<https://www.w3schools.com/sql/sql_insert_into_select.asp>

INSERT INTO **definitivo** (columna\_1,

columna\_2,

...

columna\_N

)

SELECT CAST(columna\_1 AS STRING)

CAST(columna\_2 AS STRING)

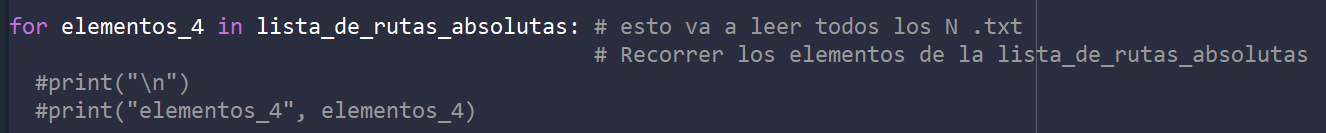
CAST(columna\_3 AS STRING)

...

CAST(columna\_N AS STRING)

FROM **temporal**

**8.** Los pasos 3 hasta 7 se hacen dentro de un for

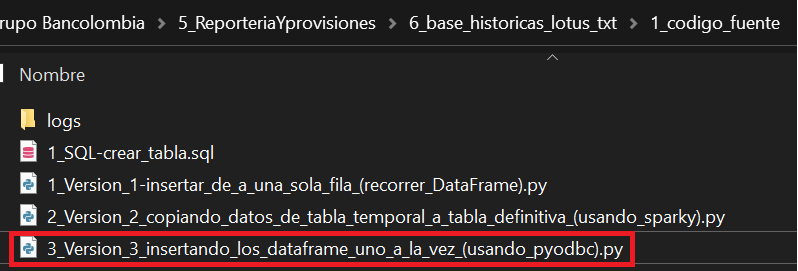


porque se repiten para cada .txt (DataFrame)

**Forma 3:**

\* Insertando los DataFrame uno por uno (uno solo a la vez) en una tabla.

- El .py con esta forma 3 es:



En esta pagina web se explica cómo hacer esto:

<https://towardsdatascience.com/how-i-made-inserts-into-sql-server-100x-faster-with-pyodbc-5a0b5afdba5>

- El problema de las formas 3 y 4 es el mismo:

\* Esto se hace con la librería pyodbc la cual no permite subir DataFrame muy pesados y como los .txt que hay que subir son demasiado pesados entonces cuando intentes hacerlo de esta forma te va a dar el siguiente error:

<https://stackoverflow.com/questions/43641122/odbc-teradata-driver-hy001-memory-allocation-error-what-is-the-meaning>



**Forma 4:**

\* Para cada .txt subir un solo DataFrame, ejemplo: Si son 10 .txt entonces se suben 10 tablas y después en la base de datos de Bancolombia crear una tabla final haciendo UNION ALL de todas las tablas:

<https://www.w3schools.com/sql/sql_union.asp>

- Esta forma 4 es similar a la forma 3 lo único que cambia es que en vez de subir cada uno de los .txt a una sola tabla, estoy creando varias tablas para cada uno de los .txt y después creo una tabla final haciendo UNION ALL

- El problema de las formas 3 y 4 es el mismo sin embargo tal vez puedas encontrar otra librería que permita subir archivos muy pesados.

En este caso se usó from sparky\_bc import Sparky aunque el problema es que esta librería solamente sirve para la base de datos de Bancolombia:

<https://hue.bancolombia.corp:8889/hue/accounts/login?next=/hue/>

Estas son otras alternativas:

- DataFrame.to\_sql(...)

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_sql.html>

- pyodbc

<https://pypi.org/project/pyodbc/>

<https://github.com/mkleehammer/pyodbc/wiki>

- PySparkSQL

<https://spark.apache.org/>

<https://spark.apache.org/docs/2.4.0/api/python/pyspark.sql.html>