**PI Data Engineer Challenge**

Tomaremos dos formas muy parecidas de resolver el challenge pero con tecnologías diferentes. Primero trataremos de partir de un entorno on premise y luego aplicaremos el caso en cloud.

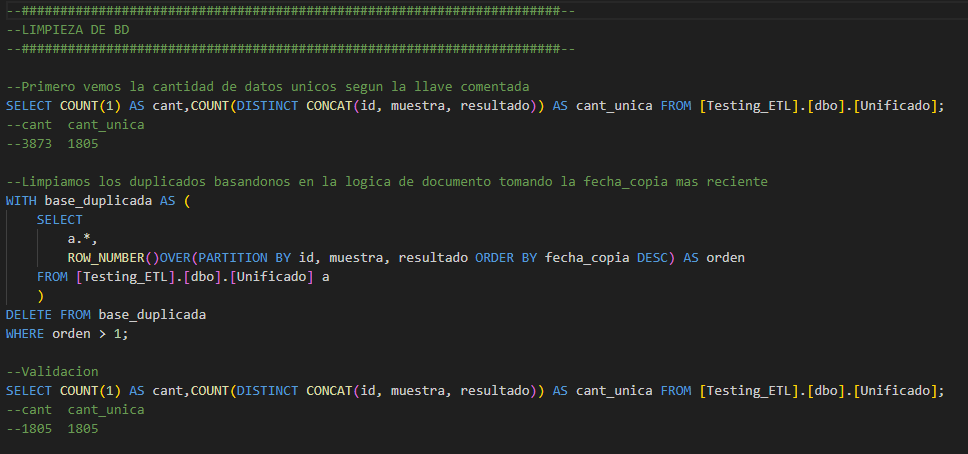
**On Premise**

Para este caso utilizare como lenguaje de programación python y SQL principalmente y para interactuar con la base de datos usare SQL Server Management Studio de manera local.

1. Duplicados en base final

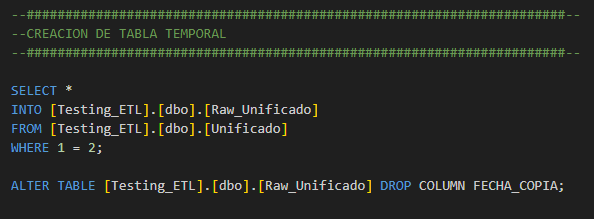
Para este problema primero tendremos que realizar una limpieza de los registros que se encuentran en la BD basándonos en las llaves ID, MUESTRA y RESULTADO y tomando como campo de referencia la FECHA\_COPIA que nos permitirá conocer cuál es el último registro que se copió.

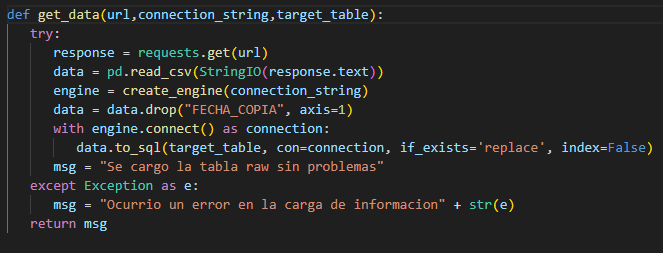
Se valida la cantidad de llaves únicas y se hace una limpieza con la función de row\_number y las CTE para cumplir con esto. El código se puede revisar en el archivo sql/Query\_Preparacion.sql



1. Descarga de información

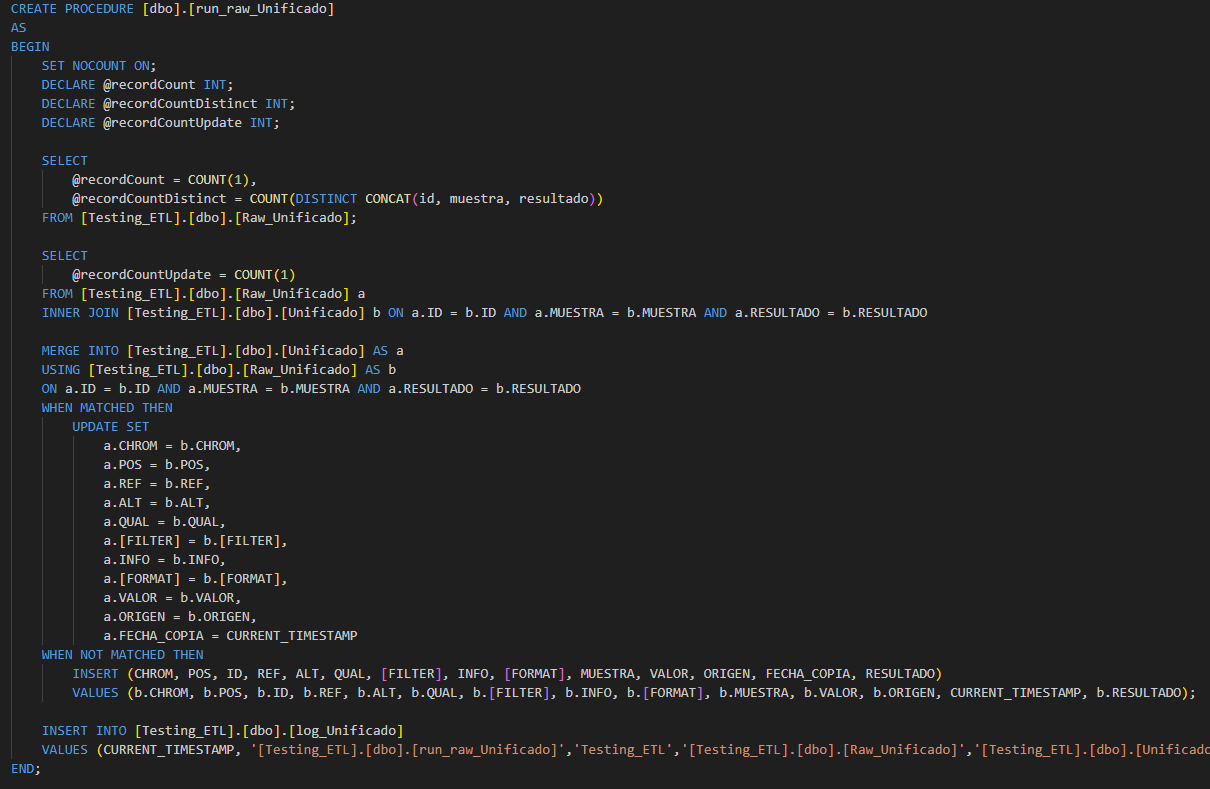
En este caso la data la obtenemos directamente de una URL por lo cual utilizaremos la librería de requests en python para poder capturarlo luego usaremos pandas para estructurar la información y luego mandarla a una tabla previamente creada el SQL Server.





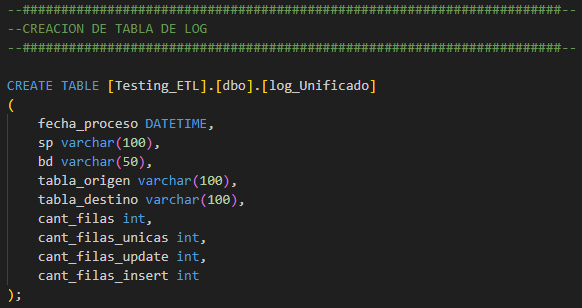
1. Proceso que inserte la informacion del dia a la tabla final

Para la creación del pipeline usare un store procedure donde se guardará la lógica (MERGE) de cruce con la tabla final para evitar la generación de nuevos duplicados así como la captura de metadata que nos servirá luego para registrar en los logs.



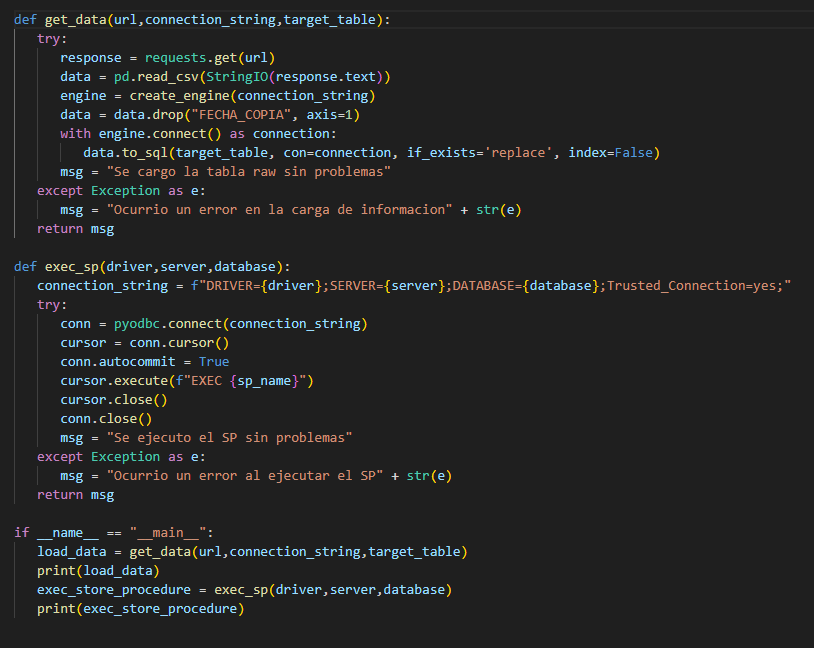
1. Captura de logs

En este caso aprovecharemos el store procedure para capturar estos datos y guardarlos en una tabla de logs previamente creada.



1. Schedular proceso

Todo el proceso (descarga y carga) será orquestado en un solo python el cual llamaremos por medio del Task Scheduler de windows para tener ejecuciones controladas y programadas.





1. Conclusiones

* Se utilizó esta solución debido a que es una de las maneras más rápidas y de bajo costo con la cual puedes obtener el resultado esperado.
* Las principales dificultades encontradas fueron los formatos de conexión y estructura de elementos utilizados en la solución por lo cual se tuvo que apoyar en documentación así como uso de IA para obtener ciertas configuraciones.
* La solución estaba planeada para tomar 2 o 3 horas pero tomó un poco más de 4 horas debido a la preparación del ambiente previo a la ejecución del desarrollo.
* Las páginas visitadas fueron documentación propia de Microsoft así como ChatGPT.

**Azure - Cloud**

En este caso utilizaremos una serie de servicios que nos permitiran completar el objetivo del challenge.

Azure SQL: Como BD

Storage Account: Para guardar la información del endpoint en una capa de landing

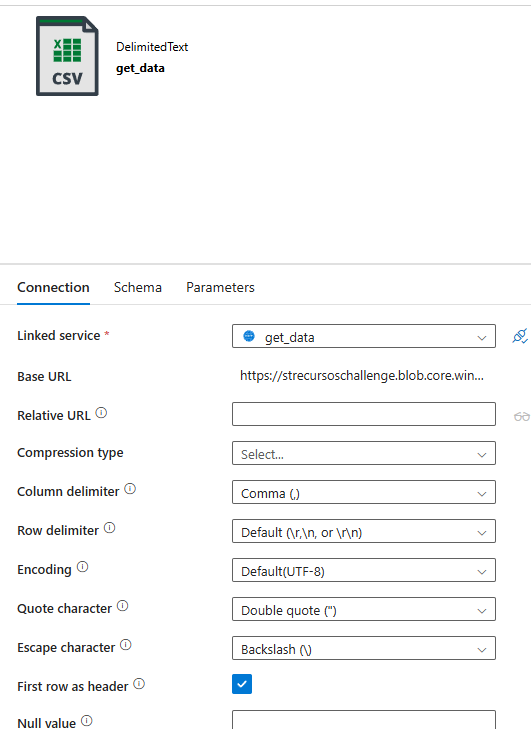
Data Factory: Como orquestador y scheduler

1. Duplicados en base final

Sigue la misma lógica que el on premise

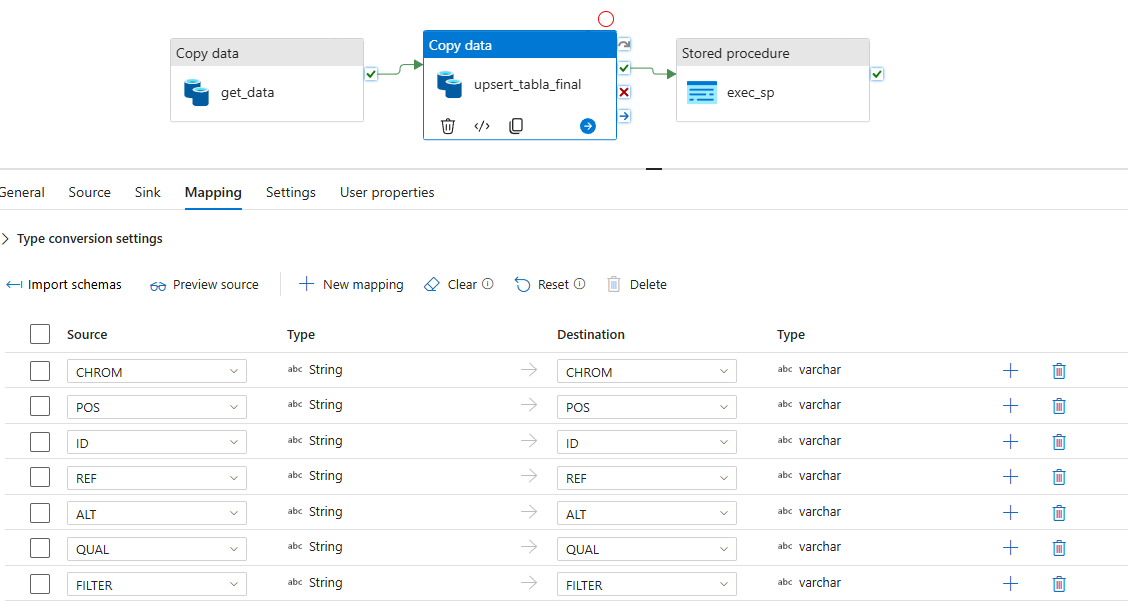
1. Descarga de información

En este caso nos valemos del componente de Copy Data en ADF, creando un linked service de HTTPS que nos permitira descargar el archivo en un container.



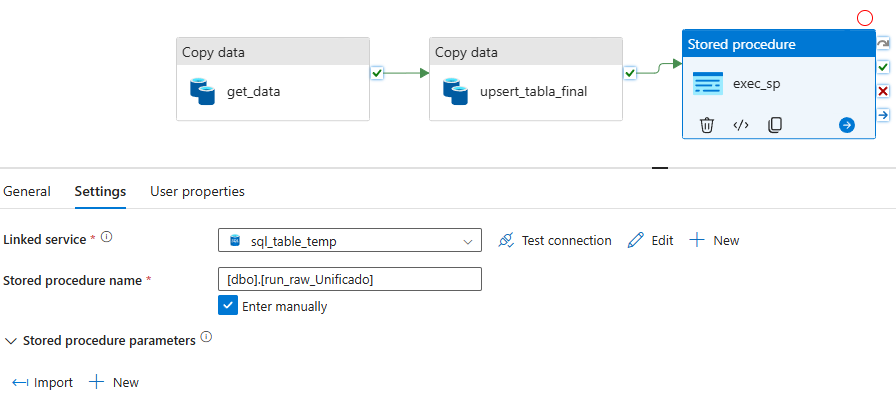
1. Carga de archivo a tabla temporal

Igualmente utilizaremos un Copy Data para realizar la carga a la tabla temporal teniendo en cuenta el campo FECHA\_COPIA no la tendremos en el mapping.



1. Proceso que inserte la informacion del dia a la tabla final y logs

En este caso optamos por la ejecución del mismo stored procedure que creamos en el on premise ya que nos permite tener un mayor control sobre la metadata generada a su vez que hace más sencillo la aplicación.



1. Conclusiones

* Si bien esta solución es un poco más cara, nos permite tener los beneficios de la nube.
* En este caso la dificultad se presentó en el momento de definir cómo guardar los metadatos, lo cual era una necesidad dentro del challenge. Por comodidad se decidió utilizar el stored procedure que permite hacer update sobre la tabla final y guardar metadata.
* Esta segunda parte tomó un poco menos de lo esperado debido a que gran parte de la solución on premise fue reutilizada acá.
* Las páginas visitadas fueron documentación propia de Azure.