from google.cloud import bigquery

from datetime import datetime

import os

# Establece la variable de entorno para el archivo de credenciales de la cuenta de servicio

# Esto es necesario para autenticar la aplicación con Google Cloud.

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = (

"C:/Users/JoaquínFonsecaVázque/OneDrive/PROYECTOS/BANORTE/banorte\_ksm.json"

)

# Crea un cliente de BigQuery

client = bigquery.Client()

# Configura las variables necesarias

project\_id = "poc-dataai-kd"

dataset\_id = "test\_replicate"

location = "us"

kms\_key\_name = f"projects/{project\_id}/locations/{location}/keyRings/ksm\_banorte/cryptoKeys/bq-dataset\_banorte"

logging\_table\_id = "poc-dataai-kd.banorte\_ksm.tablas\_no\_encriptadas"

# Referencia al dataset

dataset\_ref = client.dataset(dataset\_id, project=project\_id)

try:

# Lista todas las tablas en el dataset

tables = client.list\_tables(dataset\_ref)

# Itera sobre las tablas y ejecuta el comando COPY TABLE para cambiar el cifrado

for table in tables:

table\_id = f"{project\_id}.{dataset\_id}.{table.table\_id}"

print(f"Ejecutando copia para la tabla {table\_id}:")

# Configuración del trabajo de copia

job\_config = bigquery.CopyJobConfig(

write\_disposition=bigquery.WriteDisposition.WRITE\_TRUNCATE, # Sobrescribe la tabla de destino

destination\_encryption\_configuration=bigquery.EncryptionConfiguration(

kms\_key\_name=kms\_key\_name

),

)

try:

# Ejecuta el trabajo de copia de la tabla sobre sí misma para cambiar el cifrado

copy\_job = client.copy\_table(

table\_id,

table\_id, # Copia sobre sí misma

location=location,

job\_config=job\_config,

)

copy\_job.result() # Espera a que el trabajo se complete

print(f"Tabla {table\_id} copiada y encriptada con éxito.")

except Exception as e:

print(f"Error al copiar o encriptar la tabla {table\_id}: {str(e)}")

# Inserta un registro en la tabla de logging si ocurre un error

rows\_to\_insert = [

{

"project": project\_id,

"dataset": dataset\_id,

"table": table.table\_id,

"update": datetime.now().isoformat(),

"comments": str(e),

}

]

errors = client.insert\_rows\_json(logging\_table\_id, rows\_to\_insert)

if errors == []:

print(f"Registro insertado en {logging\_table\_id}")

else:

print(f"Error al insertar el registro en {logging\_table\_id}: {errors}")

except Exception as e:

print(f"Error al listar las tablas en el dataset: {str(e)}")

print("Proceso completado.")