**ETL AWS Glue**

Versión 6.0

Subdivisión : **Área de Proyectos - BI**

Autor : Jackson J V da Silva

Revision : Neivam Carvalho/ Mirelys Tamayo

Fecha : 26/10/2018

**CONTENIDO**

[1. Introducción 3](#_Toc528078696)

[1.1 Necesidad del proyecto 3](#_Toc528078697)

[1.2 Objetivo del proyecto 3](#_Toc528078698)

[1.3 Objetivo específico 3](#_Toc528078699)

[2. Arquitectura de la Solución 4](#_Toc528078700)

[2.1 Diagrama de Flujo 4](#_Toc528078701)

[2.2 Envío de Parametros 5](#_Toc528078702)

[2.3 Base de datos 7](#_Toc528078703)

[2.4 Registro cdc y error 8](#_Toc528078704)

[3. Especificaciones Técnicas do Servicio 11](#_Toc528078705)

[3.1 Tipos de Datos 11](#_Toc528078706)

[3.2 Carga de datos 12](#_Toc528078707)

[3.3 Creación de Catálogo 12](#_Toc528078708)

[4. Procedimiento de Ejecución del JOB 14](#_Toc528078709)

[4.1 Creación y ejecución de un job 14](#_Toc528078710)

[4.2 Ejecución de Carga del Datalake 17](#_Toc528078711)

[5. La Solución de Problemas 31](#_Toc528078712)

[5.1 The number of lines between .ctr file and table [nombre de la tabla] doesn't match 32](#_Toc528078713)

[5.2 An error occurred while calling o794.count 33](#_Toc528078714)

[5.3 The ctr file of the table [nombre de la tabla] return with 0 rows 35](#_Toc528078715)

1. Introducción
   1. Necesidad del proyecto

Automatizar el JOB de ETL para realizar la conversión de datos extraídos de los sistemas internos de Cencosud a un formato de Data Lake.

Esta transformación permitirá que la información sea utilizada por las herramientas de Big Data y garantizar un excelente rendimiento en la utilización de esta información por las herramientas del ecosistema Hadoop.

Básicamente este Job es una herramienta totalmente parametrizada que permitirá la conversión de todas las interfaces, realizar procesamiento y cargas incrementales, transformando los archivos en formato CSV al formato Parquet ejecutado por AWS Glue, dependiendo de los parámetros informados en cada archivo.

* 1. Objetivo del proyecto

Construir un Data Lake que tendrá dos tipos de información:

1. Información que será utilizada por el proceso de ETL para realizar la carga de los datos en el EDW, con interfaces predefinidas, y que deben ser mantenidas por los equipos de los sistemas de origen, pues poseen reglas de extracción de datos y que durante el ciclo de vida de los sistemas pueden modificarse.
2. Datos crudos, que se basan simplemente en tablas de los sistemas de origen. Estos datos podrán ser utilizados por los científicos de datos, por personas autorizadas o por cualquier persona que necesite acceso a estos datos. Debido a que estos datos se replican en la nube, pueden ser accedidos sin impactar los sistemas de origen, por herramientas del ecosistema Hadoop.

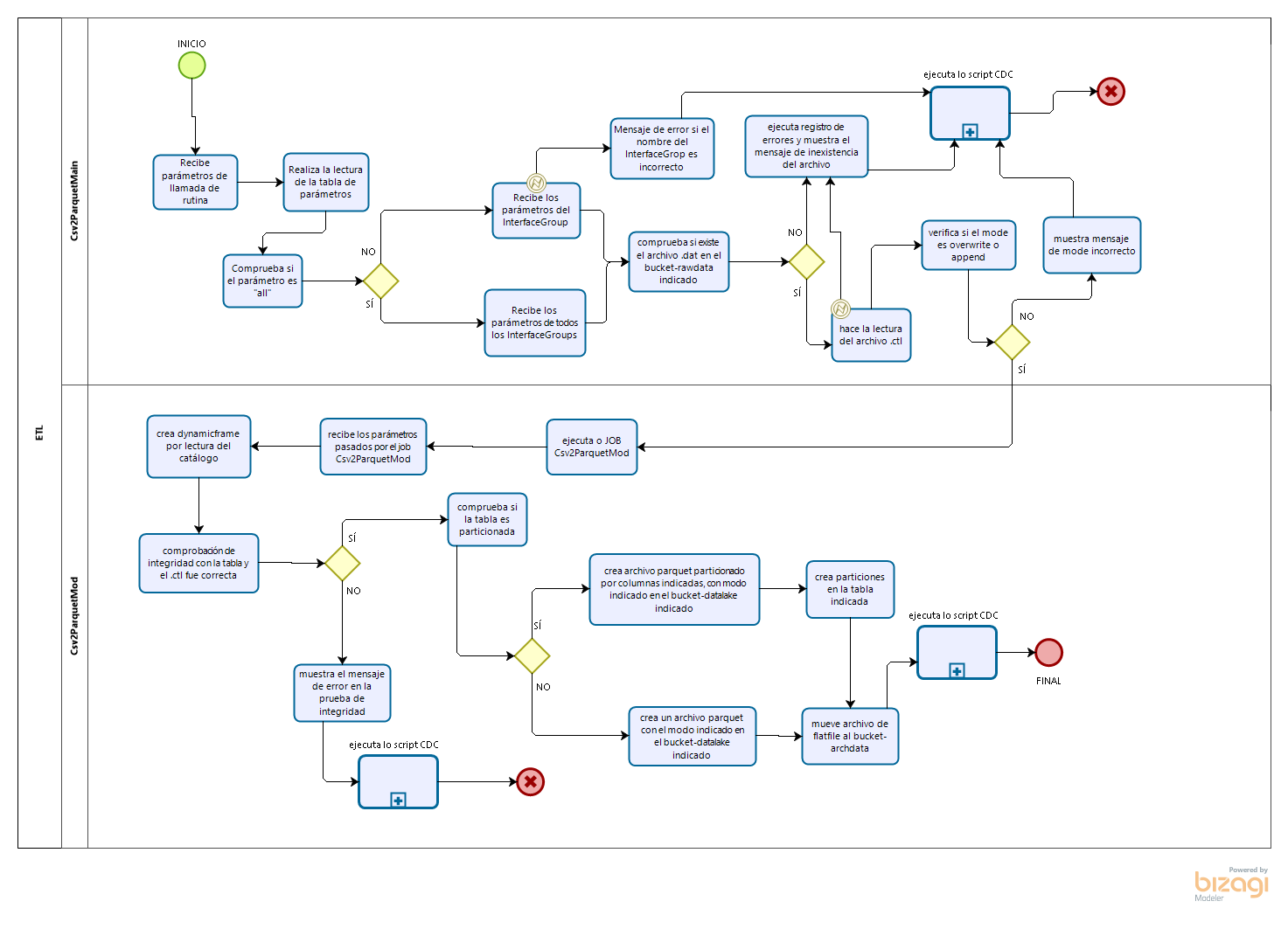
ATENCIÓN: Todos los datos se pueden utilizar en el SandBox.

* 1. Objetivo específico
* Crear una base de datos en el Catálogo AWS Glue.
* Crear una tabla de parámetros que recibe un archivo JSON.
* Lectura de las tablas creadas en Athena. La lectura se hará para que los datos se convierten al formato de columnas
* Ejecuta JOB que lee archivo Flat File (CSV, Texto) por el catalogo y crea archivo PARQUET (de columna) y compactación SNAPPY.
* Mantener registro de carga y error de DataLake.

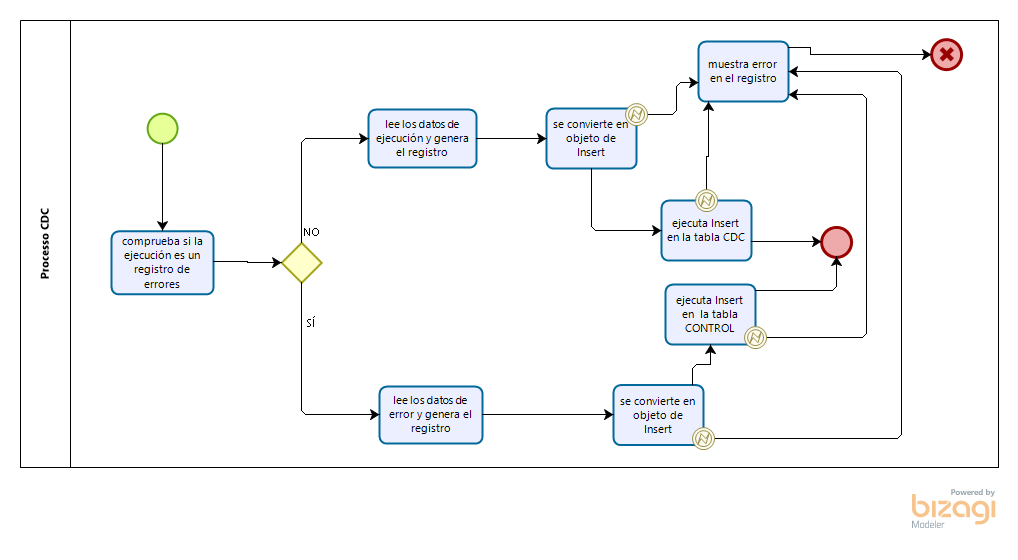
1. Arquitectura de la Solución
   1. Diagrama de Flujo

El "job" al principio tiene la siguiente lógica:

**Procedimiento Csv2ParquetMain y Csv2ParquetMod**



**Procedimiento CDC**



* 1. Envío de Parametros

Estos son requeridos para la ejecución del job, los parámetros se dividen en:

La llamada de rutina, se recibe cuando se inicia el trabajo, estos son:

**--interfacegroup:**

define qué grupo de interfaz se cargará para la ejecución del job

* “all” => Cuando sea necesario peocessar todas las interfaces parametrizadas en el archivo JSON;
* “[Nombre del Grupo de Interfaces]” => Utilizado como filtro cuando es necesario procesar sólo un grupo de interfaces parametrizadas en el archivo JSON.

Ejemplo: cliente, venta, etc ...

**--business:**

tipo de negocio

Ejemplo: sm-> supermercados, tiendas por departamento(parís), mdh (easy)...

**--country:**

país del negocio

Ejemplo: Colombia, Chile, etc ...

**--enviroment:**

Fase de estado del proyecto

Ejemplo: desa1, test1, etc ...

**--schema\_cdc:**

Nombre del esquema que se creó el cdc in RedShift

Ejemplo: col\_super\_stg, etc ...

El job también lee la tabla de parámetros que recibe valores de un archivo JSON, este archivo se debe cargar en: “s3://[cencosud.desa1.scripts.sm.col](https://s3.console.aws.amazon.com/s3/buckets/cencosud.desa.scripts.sm.col/?region=us-east-1&tab=overview)/[parameters](https://s3.console.aws.amazon.com/s3/)/”, en ese contiene:

**InterfaceGroup:**

Es el parámetro utilizado para identificar el "Grupo de Interfaz" de la tabla, se pasa el valor:

Ejemplo: DataLake\_Master\_Data, DataLake\_Sale, DataLake\_Inventory, etc...

**TableDataLake:**

Es el parámetro utilizado para recibir el nombre de la tabla datalake

Ejemplo: vta\_enc, arti\_arti.

**TableFlatFile:**

Es el parámetro utilizado para recibir el nombre de la tabla flatfile (lectura del archivo CSV no baket DataRaw)

Ejemplo: vta\_enc\_flatfile, arti\_arti\_flatfile.

**--Source:**

Sistema o base de datos de origen.

Ejemplo: jan, gen, pos, etc..

**Mode:**

Es el parámetro utilizado para recibir el modo, que son "append" o "overwrite",

Append para agregar los datos en destino

Overwrite sobrescribe los datos en destino

**IsPartition:**

Es el parámetro utilizado para identificar si el archivo está particionado, “y” Para tablas particionadas y sub particiones, “n” Para tablas no particionadas

**PartitionColumn:**

Es el parámetro utilizado para recibir los nombres de las columnas que se han particionado.

Ejemplo: fecha\_contable, centro\_cd, business\_dt, etc..

**ESTRUCTURA DEL ARCHIVO JSON:**

{"InterfaceGroup":"DataLake\_Master\_Data","TableDataLake":"vta\_desc","TableFlatFile":"vta\_desc\_flatfile","Source":"jan","Mode":"Overwrite","IsPartition":"y","PartitionColumn":"fecha, centro\_cd"}

Observación: Este bloque está en una sola línea.

* 1. Base de datos

La ETL Data Lake utilizará el S3 de AWS para el almacenamiento de los datos, donde se creará un bucket direccionado para cada ejecución del proyecto (ejemplo: cenconud.etl.datalake, cenconud.cloudformation, etc.).

Para el proyecto del ETL Data Lake se utilizaron los buckets de DataRaw (Ejemplo: cencosud.desa1.dataraw.sm.col.jan) para extraer los datos crudos, el bucket DataArch para archivar los flatfiles cargados (Ejemplo: cencosud.desa1.dataarch.sm.col.jan), y el bucket DataLake (cencosud.desa1.datalake.sm.col) para almacenar los archivos que se han convertido al parquet.

**Nomeclatura de los Buckets:**

Para la nomeclatura de los buckets se utilizan los parámetros de ambiente, país y negocio.

Ejemplo:

[Organización].[ambiente].[Calidad de Datos].[Negocio].[Pais] => cencosud.desa1.datalake.sm.col

Que se especifica en el archivo asignado en Sharepoint:

BI Cencosud Colombia - Documentos\Especificación Técnica\Estándares de nomenclatura Interfaces

En el archivo está marcado los siguientes buckets para ser usados:

|  |  |
| --- | --- |
| Servicio AWS | **BUCKETS S3 OFICIALES** |
| Bucket S3 | cencosud.desa1.dataraw.sm.col.jan |
| cencosud.desa1.dataraw.sm.col.gen |
| cencosud.desa1.dataarch.sm.col.jan |
| cencosud.desa1.dataarch.sm.col.gen |
| cencosud.desa1.scripts.sm.col |
| cencosud.desa1.datalake.sm.col |
| cencosud.desa1.loaddata.sm.col |
| cencosud.desa1.ldm.sm.col |
| cencosud.desa1.datamart.sm.col |

|  |  |
| --- | --- |
| Bucket S3 | cencosud.desa1.dataraw.sm.col.mot |
| cencosud.desa1.dataraw.sm.col.bnk |
| cencosud.desa1.dataraw.sm.col.pos |
| cencosud.desa1.dataraw.sm.col.inf |
| cencosud.desa1.dataarch.sm.col.mot |
| cencosud.desa1.dataarch.sm.col.bnk |
| cencosud.desa1.dataarch.sm.col.pos |
| cencosud.desa1.dataarch.sm.col.inf |

**Seguridad:**

Para almacenamiento y movimiento de archivos por el ambiente Cencosud es necesario subir archivos cifrados en "AES-256", tanto el usuario como la herramienta que utilice el S3 deben estar insertados en una IAM Role que tenga permiso de acceso al bucket en cuestión y hacer la carga de archivos encriptados.

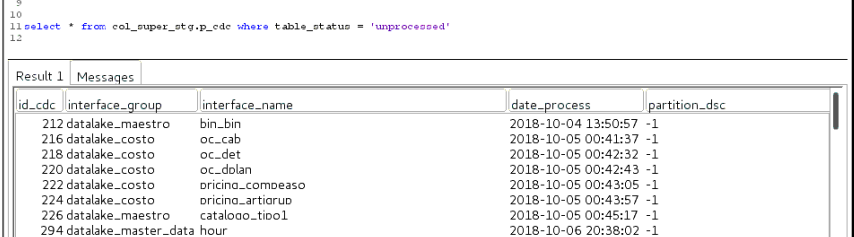
ATENCIÓN: cualquier ejecución que se haga sin seguir las directrices de encriptación, el resultado será error de acceso o encriptación.

* 1. Registro cdc y error

En el proceso de conversión de FLATFILE a PARQUET, al finalizar el job registra el resultado del proceso, sea con éxito o error.

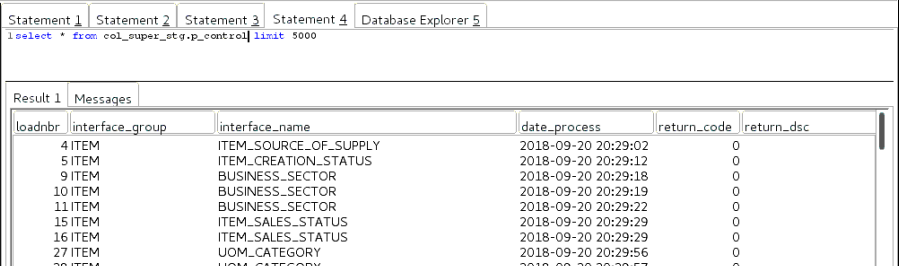
En caso de éxito el job hace un registro de la tabla cargada en la tabla p\_cdc, con los valores:

* id\_cdc => identificación del registro
* grupo de interfaz => interface\_group
* nombre de la tabla => interface\_name
* fecha y hora del registro => date\_proces
* si la tabla está particionada, se realiza un registro para cada partición y el valor se registra en la columna => partition\_dsc (Ejemplo: fecha=2018-08-20); si no, se asigna el valor "-1"
* un indicador si la tabla es una carga completa con "y" (tabla dimensión) o no "n" (tabla fato)=> is\_full\_load
* indica si la tabla se cargó una vez con "n" o es un reprocesamiento "y" =>ind\_reproces
* situación de la tabla (procesado, no procesado) => table\_status
* tipo de interface (ldm, datamart) => interface type



En caso de error el job hace un registro del error en la tabla p\_control, con los valores:

* loadnbr => identificación del registro
* grupo de interfaz => interface\_group
* nombre de la tabla => interface\_name
* fecha y hora del registro => date\_proces
* return\_code => control de respuesta del proceso (0=éxito, 9=error)
* return\_dsc => mensaje de status del proceso

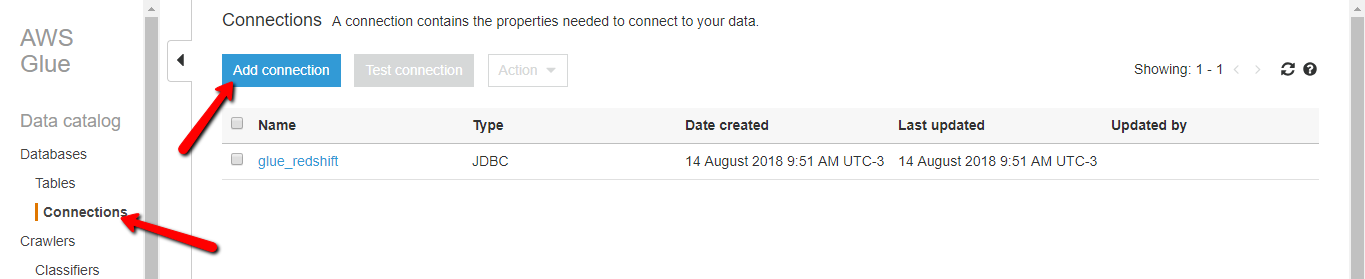


Como requisito esencial de la ejecución del job el ambiente de registro de p\_cdc y p\_control deben estar configurados.

Se crea una conexión con RedShift, que debe haber un cluster, vpc, subned y todas las propiedades necesarias para conectarse al job, como dnc para vpc y directiva de acceso en una IAM Rule al clúster de RedShift, con la siguiente base de datos:

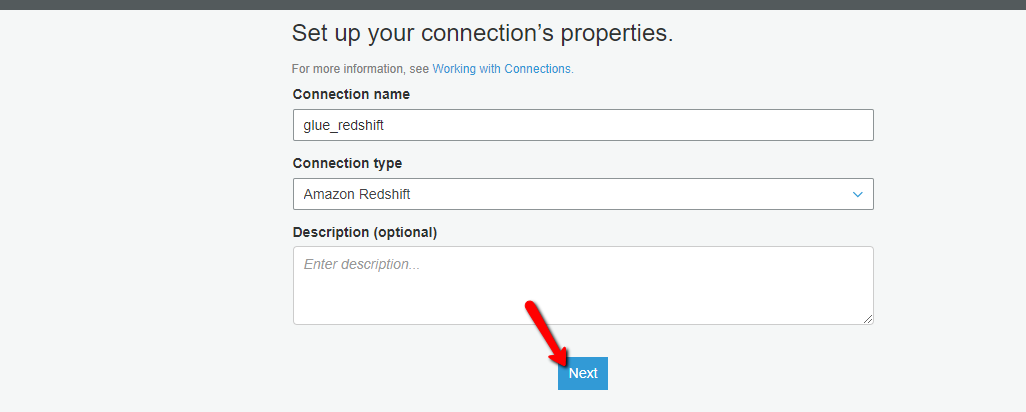


En el campo “connections” en la consola de AWS Glue, se crea la conexión:

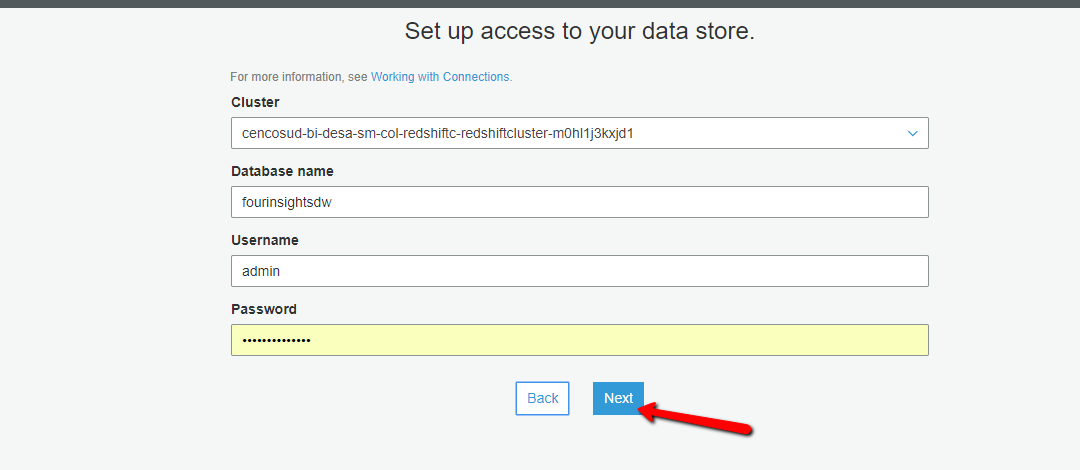


Definir los campos

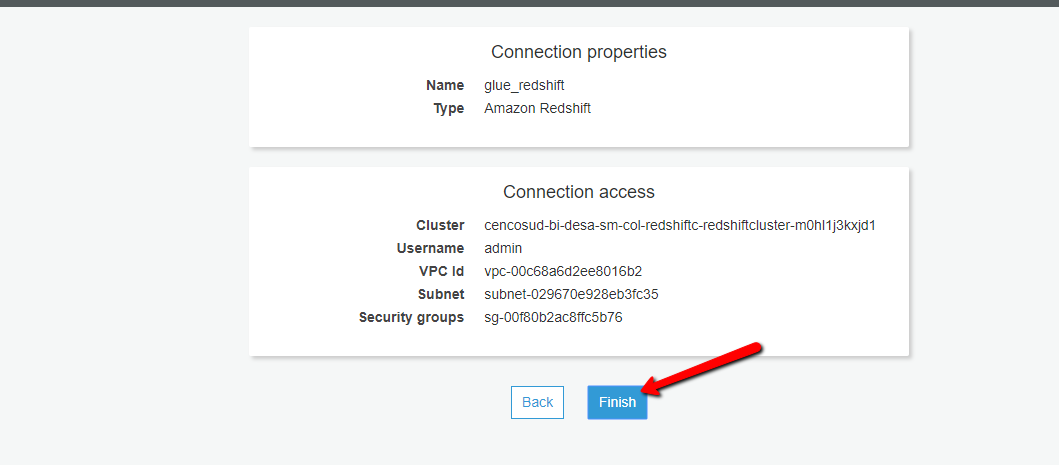
* Connection name => nombre de conexión = establecer como “glue\_redshift”
* Connection type => tipo de conexión = establecer como “Amazon Redshift”

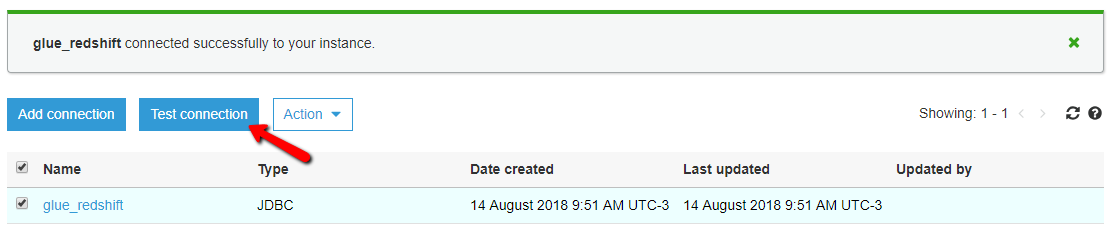


* Cluster => nombre del cluste
* Database name => nombre del database = definir como “fourinsightsdw”
* Username => nombre del usuario root
* Password => contraseña del usuario



Compruebe los datos de conexión antes de finalizar y pruebe la conexión.





1. Especificaciones Técnicas do Servicio
   1. Tipos de Datos

Para la conversión de datos de Flat File a Parquet, se utilizan external tables que se crea utilizando el hive, en el caso del Flat File por no ser tipado puede ser definido por varios tipos de datos (para detalle de qué tipo de datos utilizar, link: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types#LanguageManualTypes-HiveDataTypes), pero el archivo parquet debe tener algunos cambios:

* Para caracteres se utilizará sólo VARCHAR([Número máximo de caracteres]), entonces en el origen todos los tipos CHAR(n), STRING, etc..;
* Para números enteros se utilizan INT y BIGINT;
* Para números decimales se utiliza sólo DOUBLE
* Tipos de fecha el origen se debe establecer como DATE y el archivo parquet será VARCHAR(10)
* El valor con hora no es aceptado por DATE y TIMESTAMP, por lo tanto debe ser definido en el origen como VARCHAR(8)

ATENCIÓN: los valores de datos que contienen "pueden forzar un salto de línea equivocado, por cuenta de cualquier texto que contenga el valor de **"** debe ser removido o sustituido por **‘**.

* 1. Carga de datos

Para cargar los datos que serán convertidos por el job, los datos crudos se deben cargar en los buckets DataRaw "s3://[nombre del bucket]/[NOMBRE DE LA TABLA]" (Ejemplo: s3:// cencosud.desa1.dataraw.sm.col.jan/vta\_desc) los archivos que se deben cargar son el archivo .dat (compactados en .gz), contenido de la tabla, y el .ctr, archivo que contiene el número de líneas del archivo .dat.

ATENCIÓN: el job hace la validación de líneas de la tabla con el .ctr; si el número de líneas de la tabla no es igual al retorno del .ctr, la conversión del archivo se detiene, se genera un registro de error en la comparación de líneas de la tabla, y los archivos se mueven a la carpeta de DataError "s3://[nombre del bucket]/DataError/[ NOMBRE DE LA TABLA].

El archivo .dat y .ctr se deben escribir con encoding utf-8, para evitar errores de lectura del archivo.

Si el archivo .dat se carga sin un archivo .ctr, o lo contrario, o los archivos se cargan con una estantería diferente de .gz y .ctr, el archivo en cuestión se mueve a la carpeta de error, y se genera un registro de error "falta de archivo necesario".

* 1. Creación de Catálogo

El glue hace la lectura de catálogos (creados por athena, por él mismo, redshift, o jdbc) en este proyecto se utilizaron tablas creadas por hive, que hacen la lectura de archivos Flat File compactados (.gz) y almacenados en el S3, pero en los otros casos procedimiento es similar.

El estandard de nombre de la base de datos y de las tablas está en el link para el archivo Sharepoint es:

BI Cencosud Colombia - Documentos\Especificación Técnica\Estándares de nomenclatura Interfaces

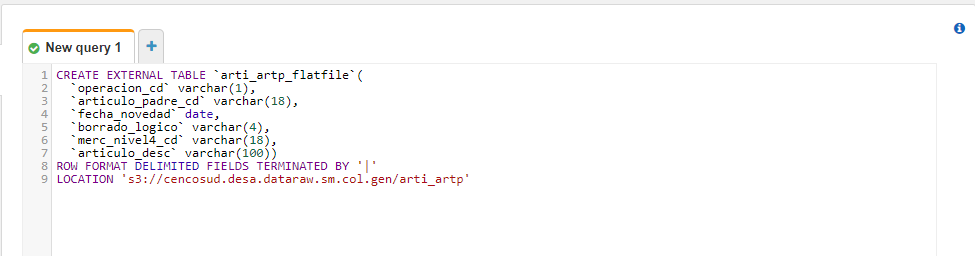
|  |  |
| --- | --- |
| Database Glue | Cencosud\_desa1\_ldm\_sm\_col |
| Cencosud\_desa1\_dataraw\_sm\_col |
| Cencosud\_desa1\_datamart\_sm\_col |
| Cencosud\_desa1\_datalake\_sm\_col |
| Cencosud\_desa1\_sandbox\_sm\_col |

Para crear una external table en el athena si utiliza un DDL que establece:

* Crear base de datos (comando "create database [nombre de la base de datos]")
  + El nombre de la base de datos se compone de “[Organización].[Ambiente].[Calidad de Datos].[Negocio].[Pais]”( Ejemplo: cencosud.desa1.dataraw.sm.col)
* Entrar en la base de datos, en el campo solicitado



* Los nombres de las tablas se definen por:
  + Tabla flatfile => vta\_enc\_flatfile (\_flatfile é fixo)
  + Tabla parquet => jan\_vta\_enc
* Columnas (nombre de columna, y el tipo de datos)
* Tipo de archivo que se lee (Flat File es informado el delimitador de columnas (con el comando = ROW FORMAT delimited fields terminated by "|", y en el caso de archivo parquet STORED AS PARQUET)
* Lugar de donde el archivo se encuentra (Ejemplo: s3:// cencosud.desa1.dataraw.sm.col.gen/arti\_arti)



En la creación de las tablas se debe considerar si esta contiene datos de movimiento (ventas) o registro de datos maestro (cliente, producto).

En caso de que sea de movimiento, debe ser particionada para mejor desempeño de la lectura de los datos de la tabla (por tener muchas líneas agregadas todos los días).

Para crear una tabla particionada: se toma la columna por la que se hará la partición y la inserta en el comando: “partition by ([nombre de columna], [tipo de datos])”.

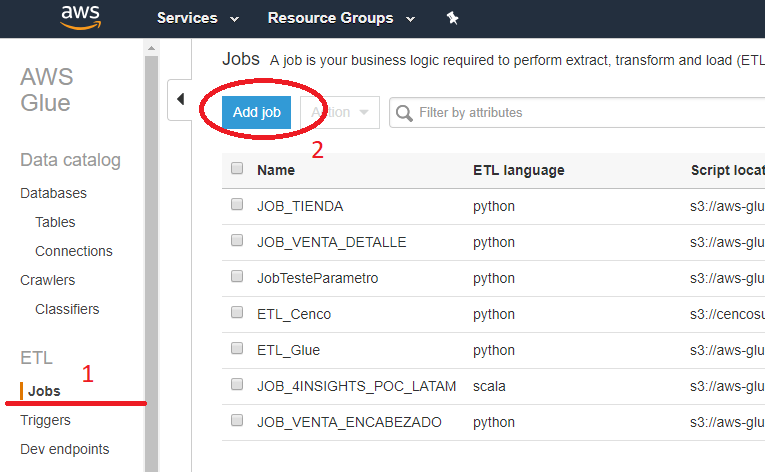
En las tablas que se particionan, se utilizarán las columnas de fecha (fecha) y de tienda (centro\_cd) para el reprocesamiento, esas tablas van a leer archivo que se dividen por grupos "particiones" de fecha (Ejemplo: fecha=2017-02-22, fecha=2017-02-24), y estas particiones se dividir por "sub-particiones" de las tiendas (Ejemplo: centro\_cd=11, centro\_cd=17).

ATENCIÓN: si una tabla es particionada por fecha y tienda, todos los registros de un movimiento completo deben ser generados de una fecha y de una tienda en el proceso de extracción de datos, por el contrario, no será posible mantener el archivo.



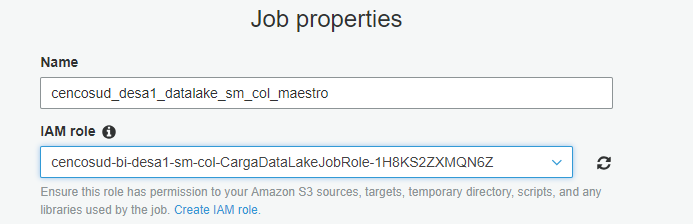
1. Procedimiento de Ejecución del JOB
   1. Creación y ejecución de un job

La creación del trabajo es muy simple, en la consola de AWS Glue, entre en job y Add job:

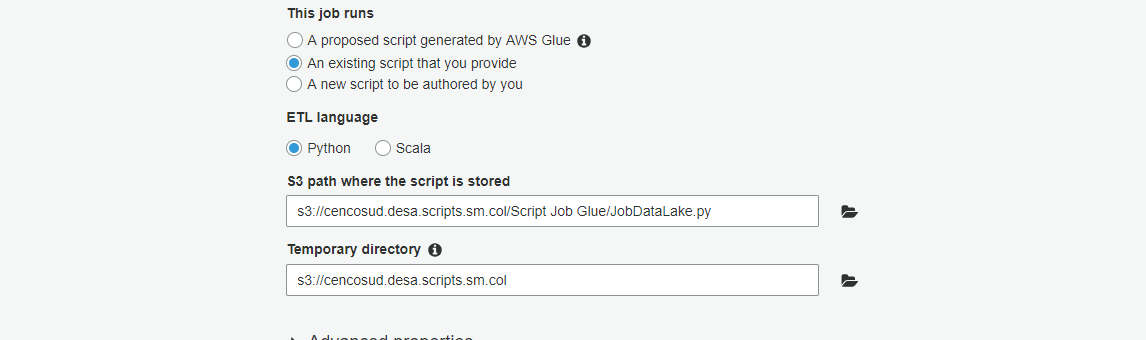


Luego se defina las configuraciones básicas del job:

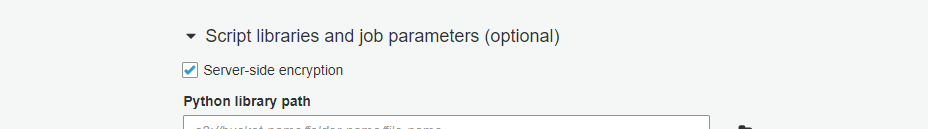
* Nombre del job
* IAM role (regla de acceso y ejecución del usuario aws)



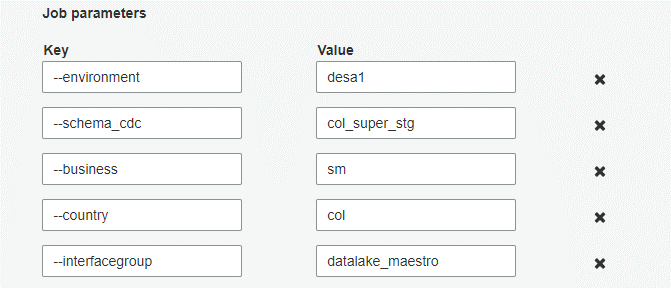
* Definición del script
  + Seleccionar la opción ”An existing script that you provide"
  + In ETL language seleccionar Python
  + in "S3 path where the script is stored", seleccione el archivo python, que se encuentra en el bucket Script (Ejemplo: cencosud.desa1.scripts.sm.col/Script Job Glue/JobDataLake.py)
  + in "Temporary directory", seleccione el bucket Script (Ejemplo: cencosud.desa1.scripts.sm.col/Script Job Glue)



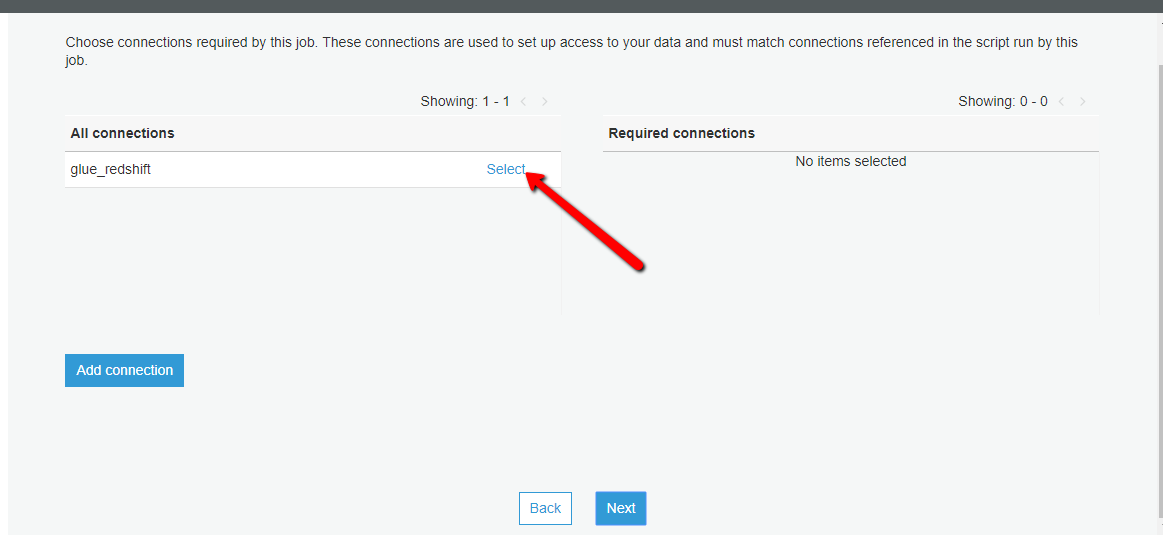
* Abrir las opciones "Script libraries and job parameters (optional)"
  + Marcar el CheckBox "Server-side encryption"



* + Pasar los parámetros de llamada
    - -- interfacegroup (grupo de interface)
    - -- country (país)
    - -- business (negocio)
    - -- enviroment (ambiente)
    - -- schema\_cdc (nombre del esquema que se creó el cdc)

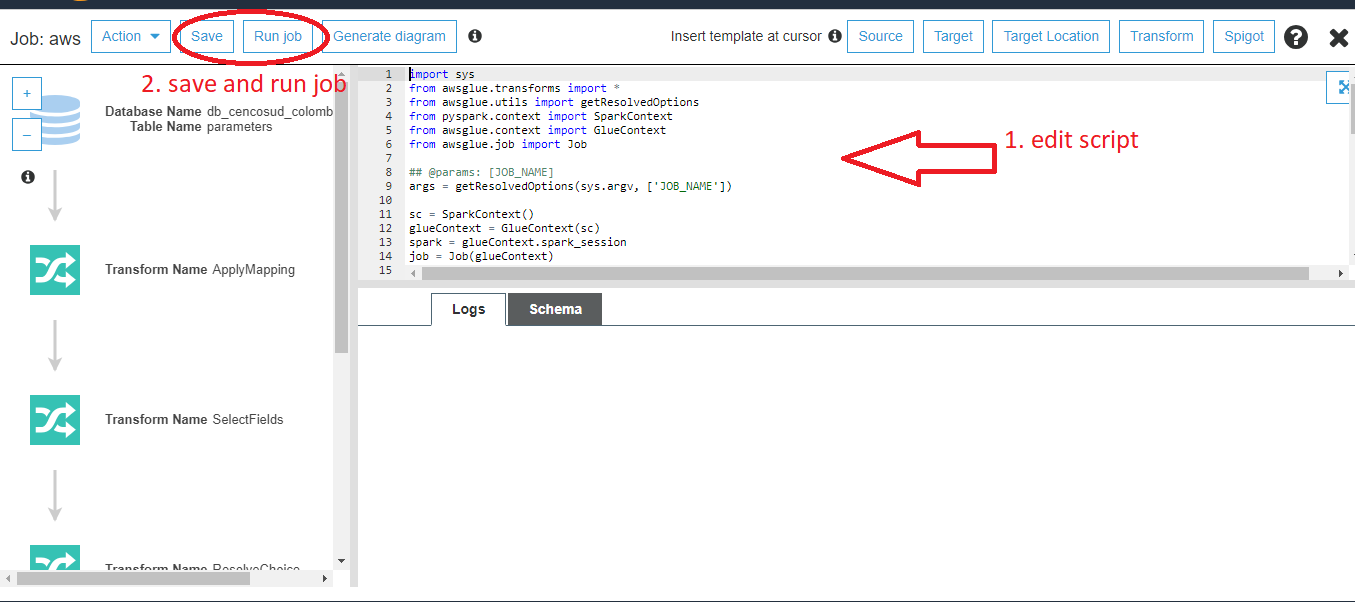


Después de la opción “next”, seleccionar la conexión con el banco de almacenamiento del registro y seleccionar nuevamente la opción “next”.



y seleccionar la opción “Save job and edit script”.

ATENCIÓN: el tiempo de ejecución del trabajo puede variar de acuerdo a la masa de datos y ejecuciones de ese, pero se debe considerar un tiempo entre 5 y 10 min, que es el tiempo que utiliza en la creación de una máquina de ejecución, y luego para eliminarlo, y por el estándar del cargo mínimo del job: s10 minutos.



Este procedimiento se debe realizar para cada paso de parámetro diferente.

Ejemplo: cencosud\_desa1\_datalake\_sm\_col\_maestro, cencosud\_test\_datalake\_sm\_col\_ventas, etc..

* 1. Ejecución de Carga del Datalake

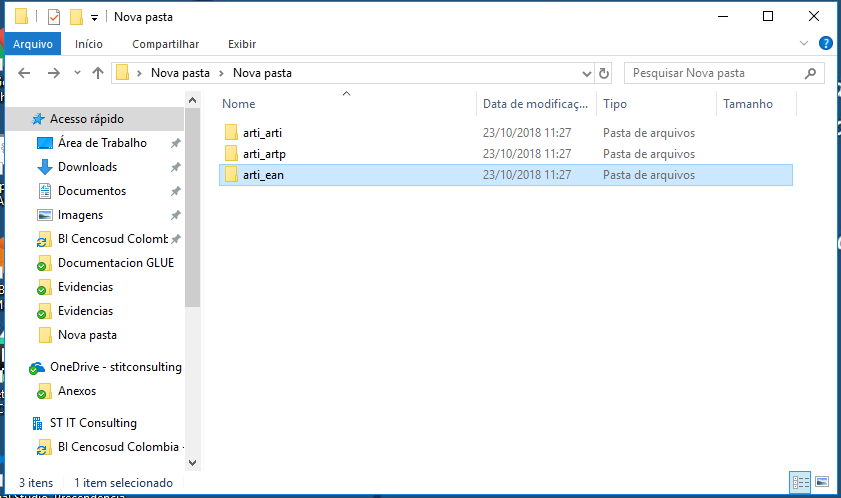
Para la ejecución de una carga en el datalake se deben seguir los pasos ejemplificados por las interfaces:

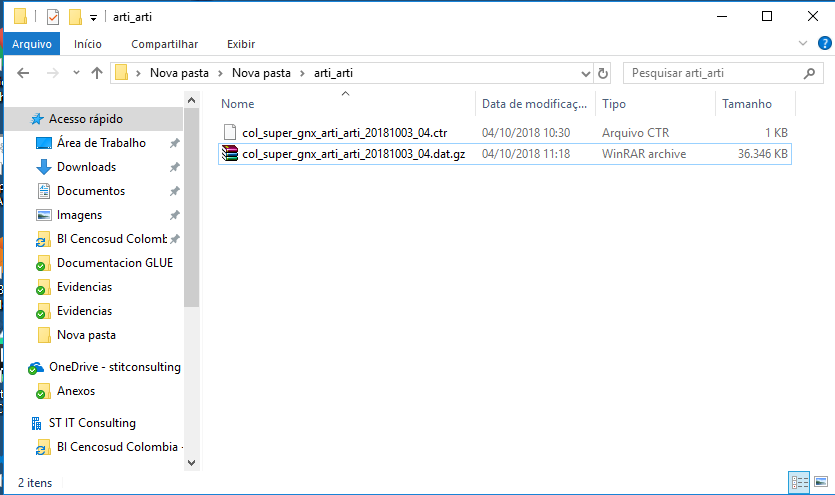
arti\_arti;

arti\_artp;

arti\_ean

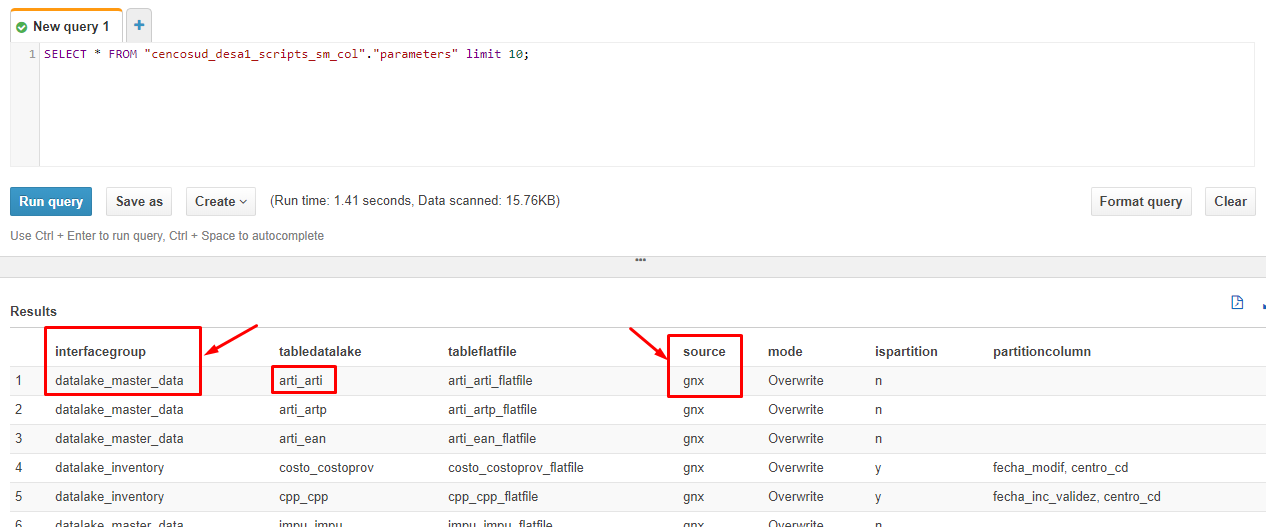
los archivos .dat y .ctr se deben colocar en una carpeta con el nombre de interfaz





Analizar la tabla de parámetros;

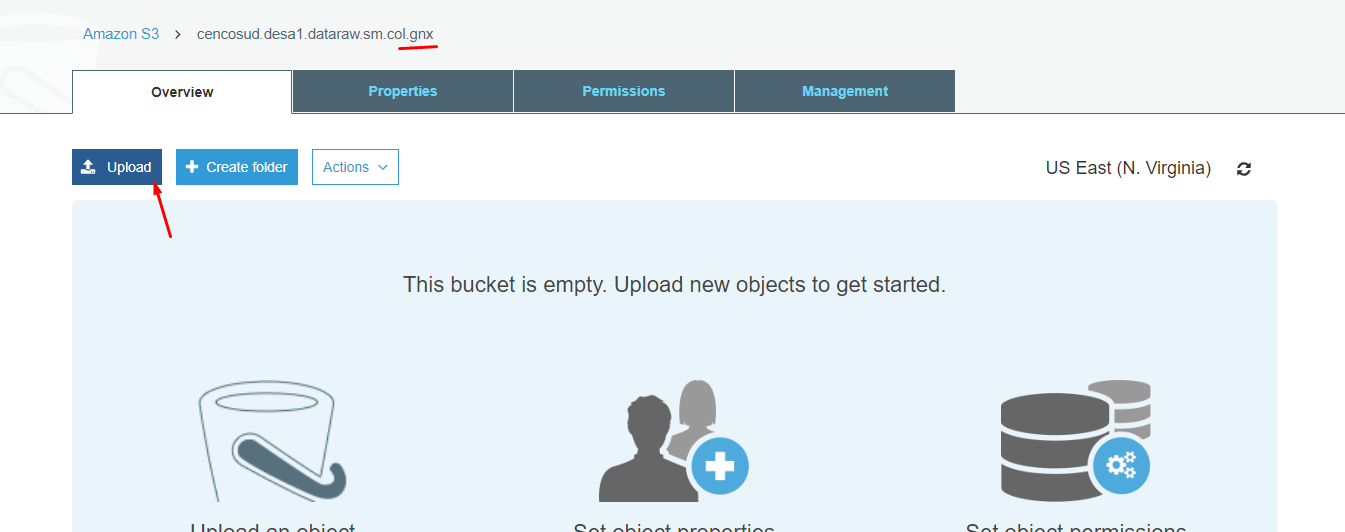
En 'tabledatalake' busque el nombre de la interfaz que desea cargar:



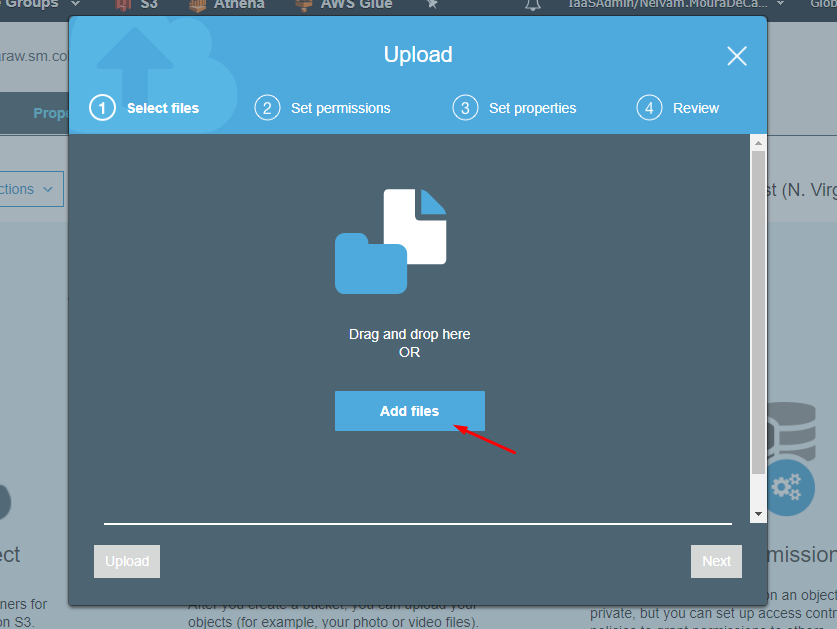
Cuando se encuentra la interfaz se debe considerar los campos 'interfacegroup' y 'source'.

El valor 'gnx' del origen es el sufijo del bucket dataraw donde se debe cargar el archivo:

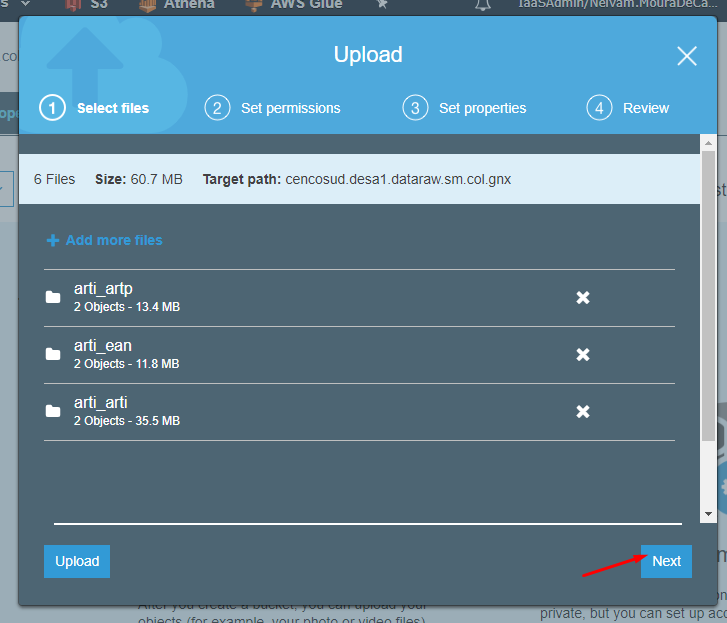
Seleccionar Upload



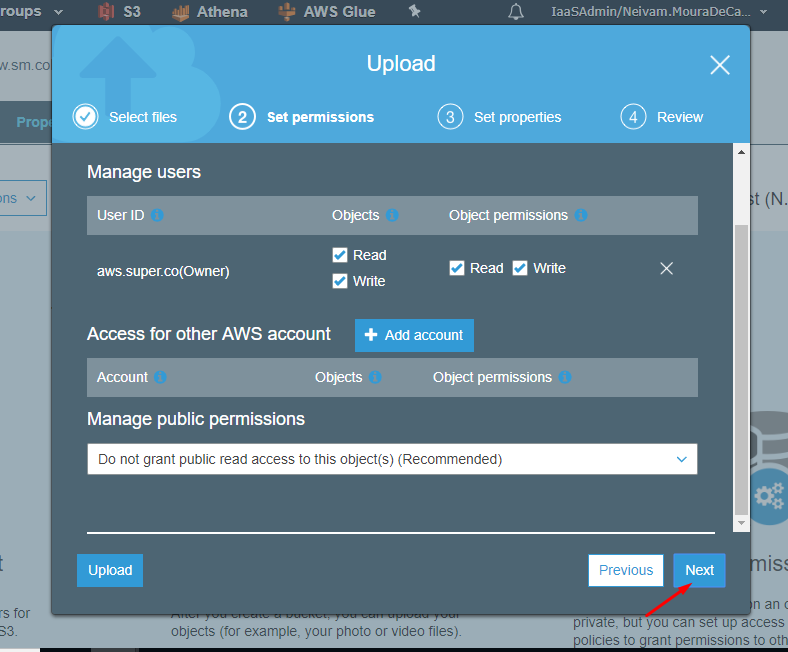
Seleccionar Add Files



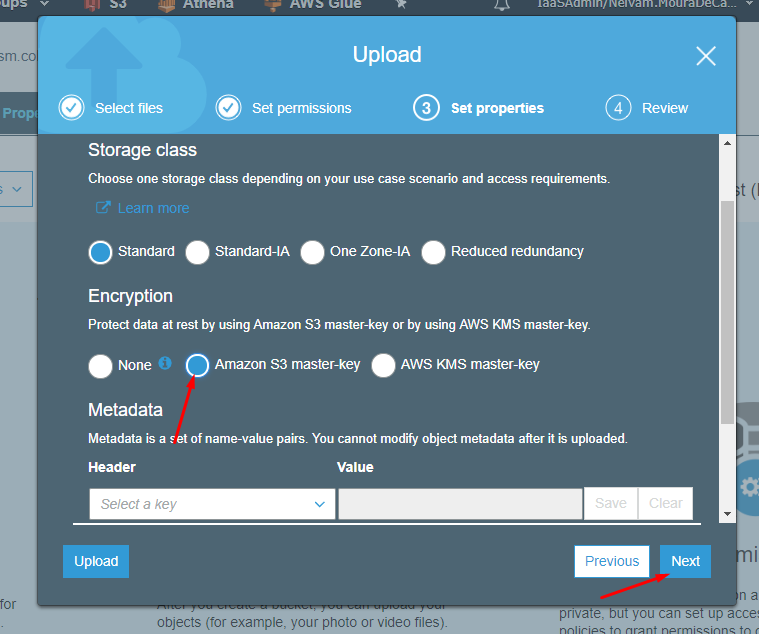
Haga clique en Next



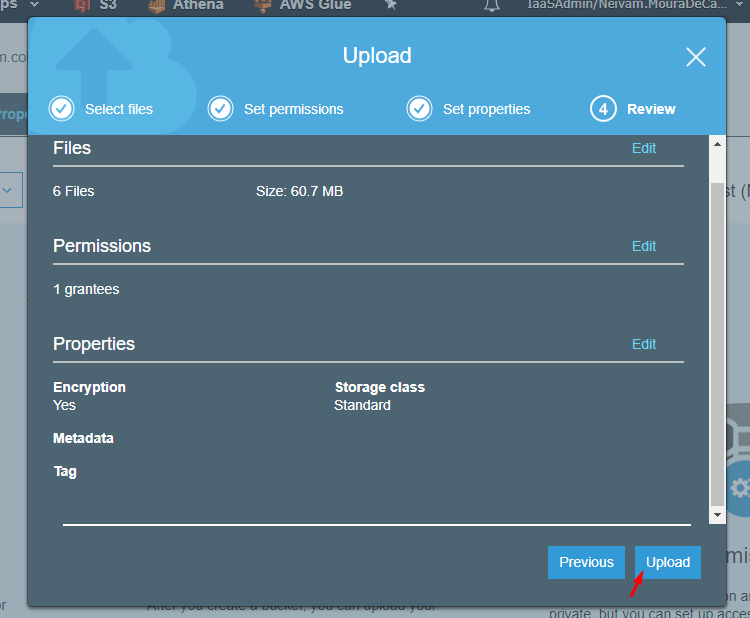
Haga clique en Next

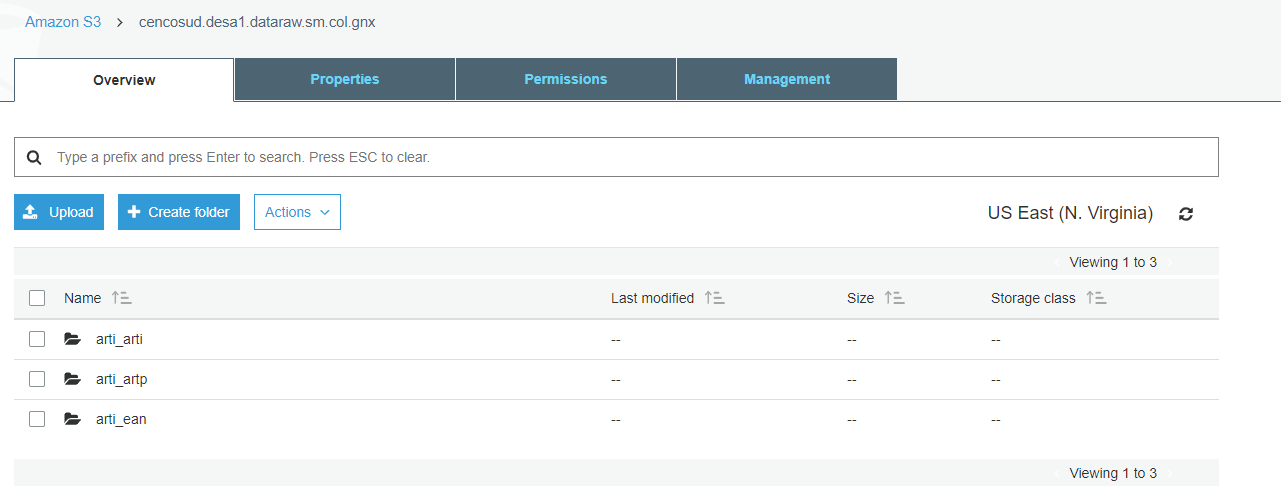


Seleccione Amazon S3 Amazon Key y después Next



Haga clique en Upload Next

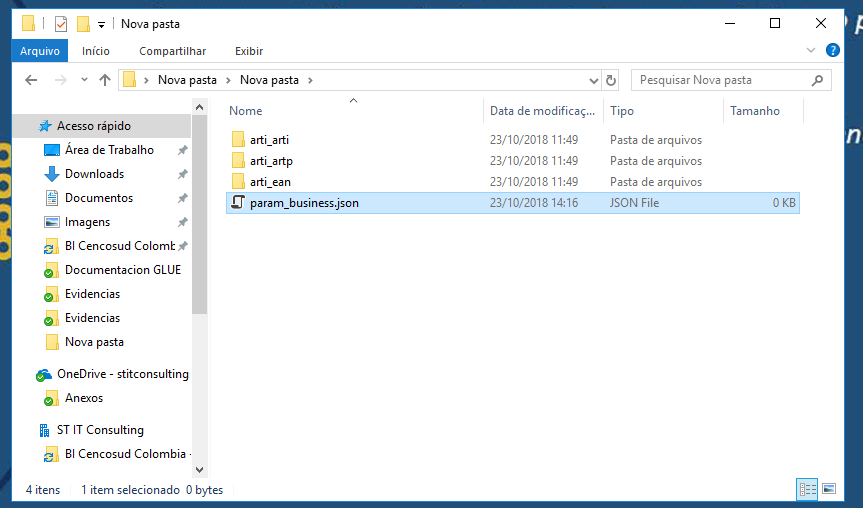




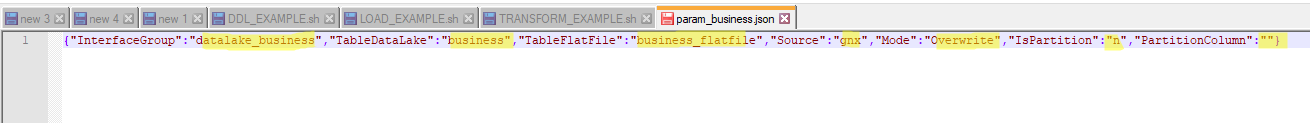
Con esto el archivo está listo para ser cargado.

ATENCIÓN:

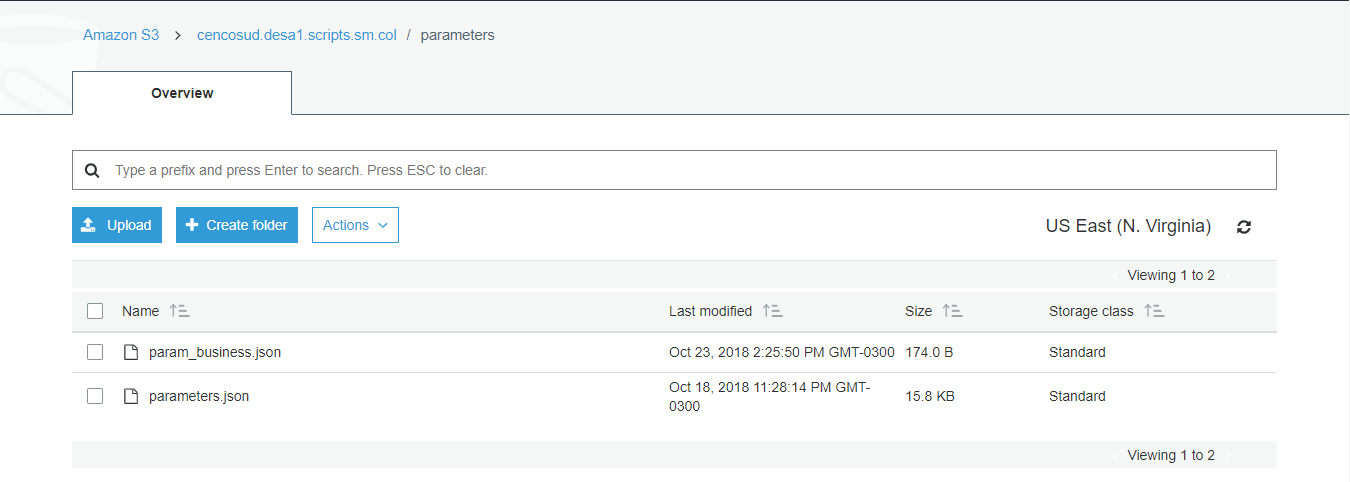
Si la interfaz que se desea cargar no se encuentra en la tabla de parámetros, se debe crear un archivo con estructura en formato json (como se describe en el índice 2.2), con la información de la interfaz para ser cargado en el directorio parameters:

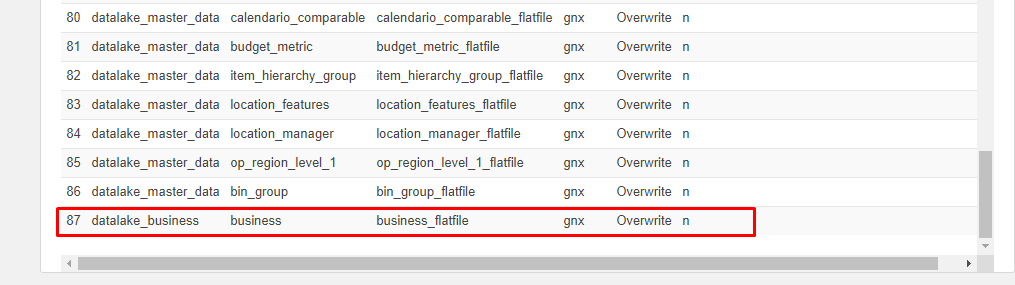


Ejemplo de archivo JSON

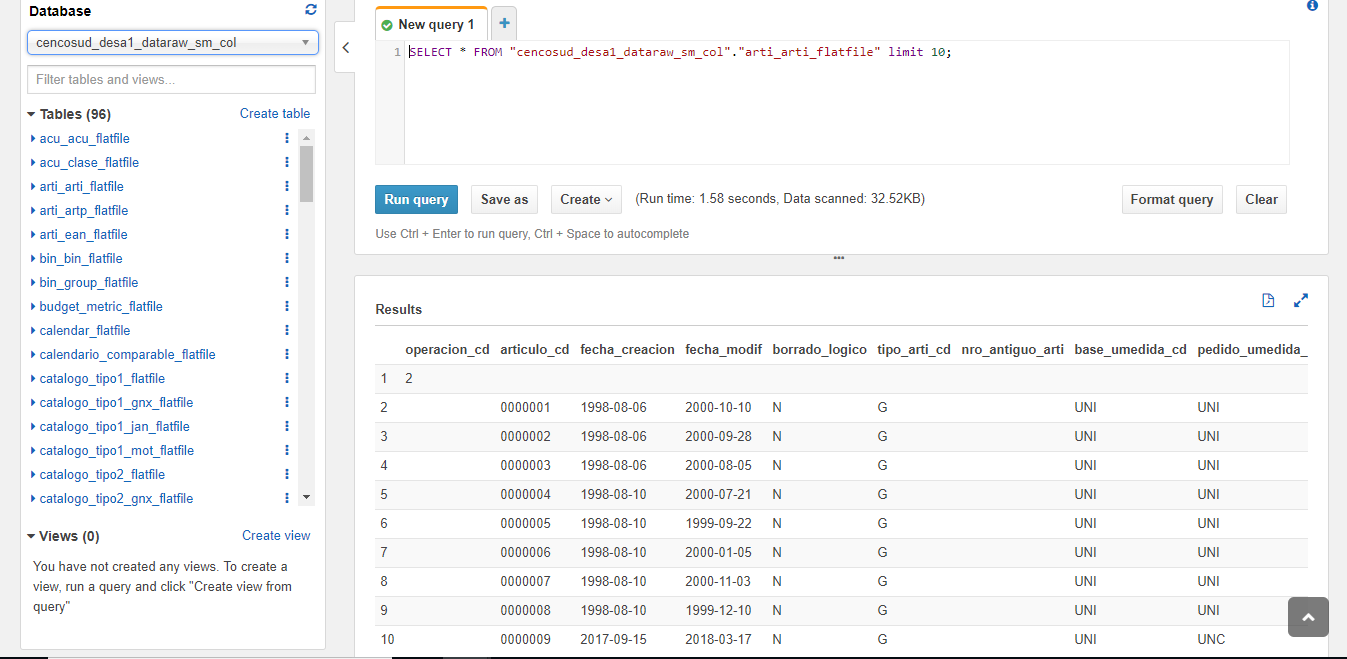


Haga upload del archivo para el bucket de scripts, de acuerdo con la imagen abajo

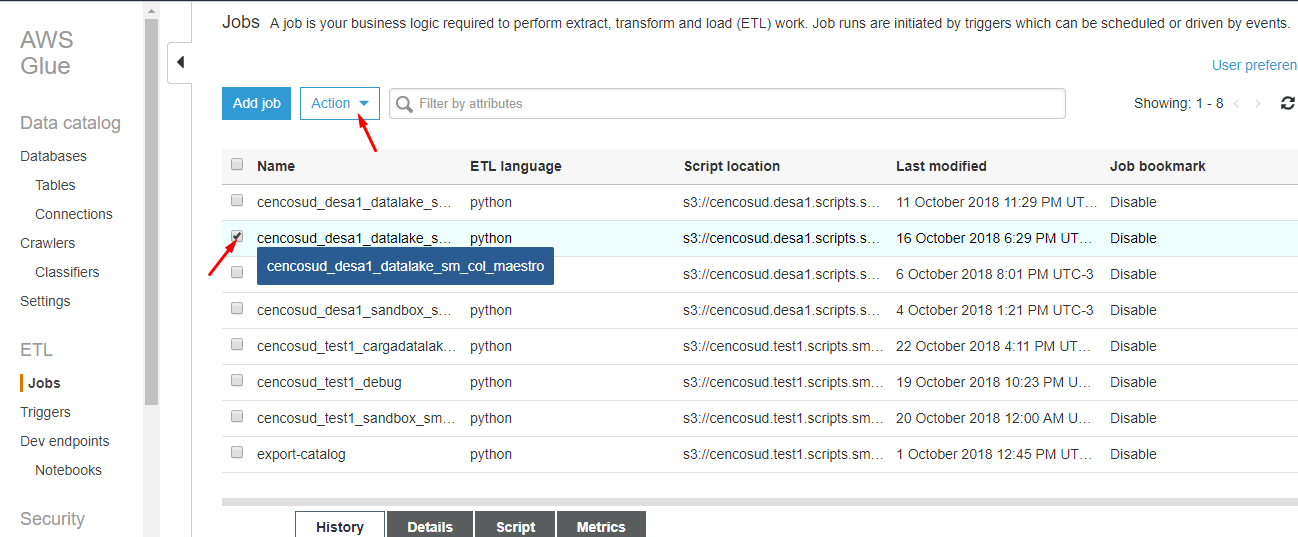


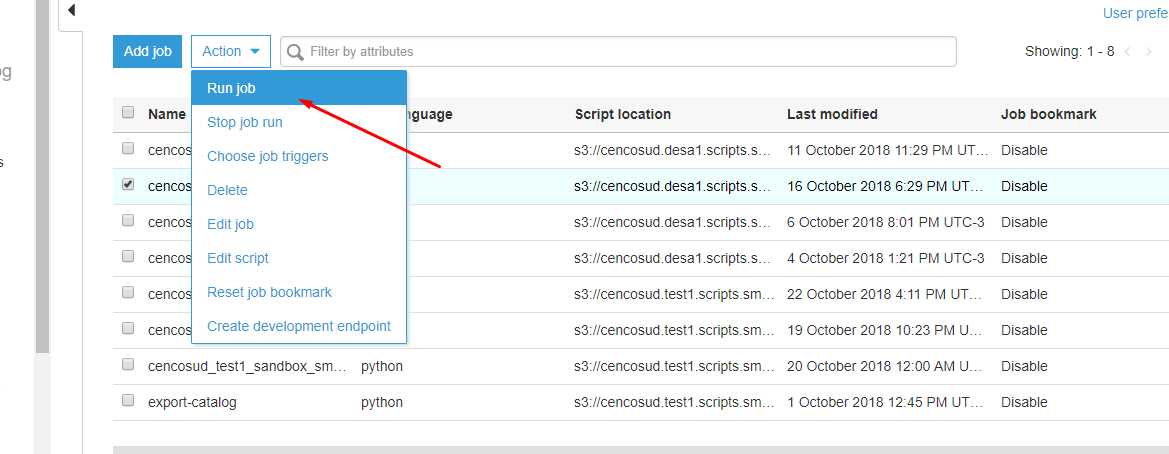


Se debe crear el catálogo que va a leer el archivo cargado en el bucket dataraw (como se describe en el índice 3.3)

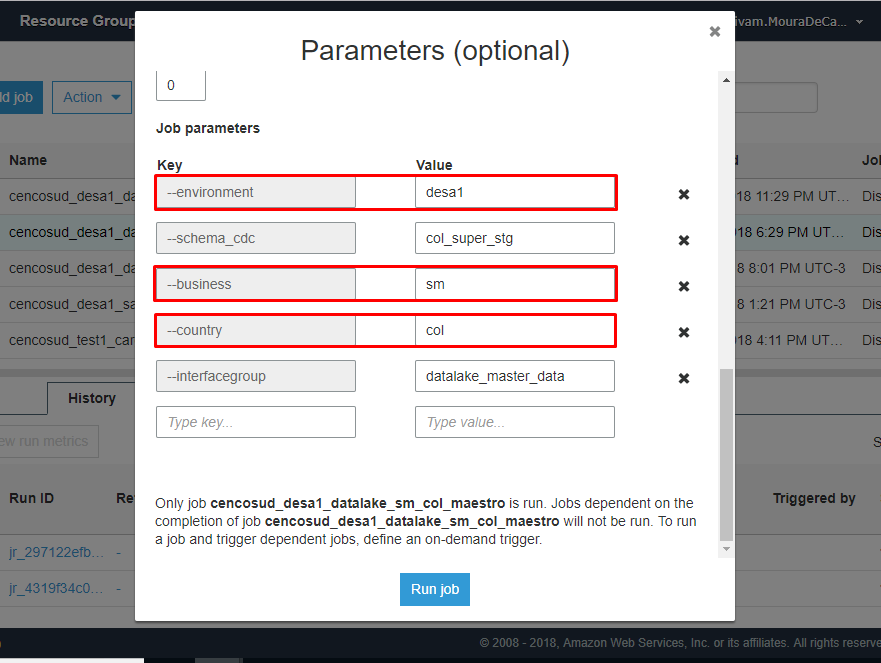


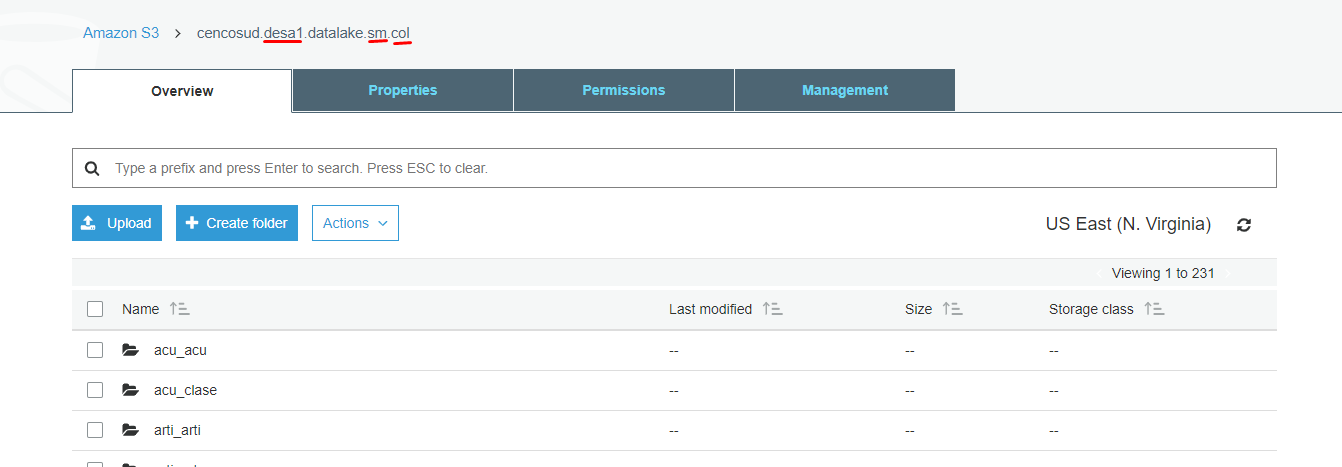
Se debe ejecutar el job de carga del datalake, pero antes se deben ver los parámetros que se están pasando.





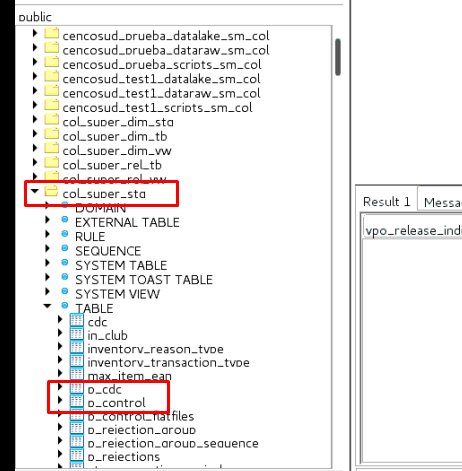
Los parámetros deben componer el nombre del bucket datalake:



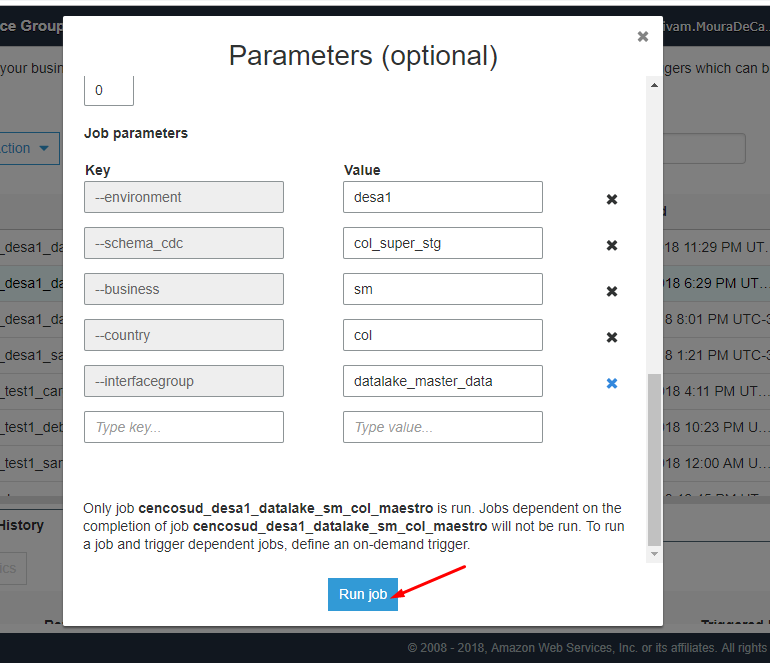


El parámetro '--interfacegroup' debe contener el valor 'all' para cargar todos los interfazes que se encuentran en la tabla de parámetros, o debe contener el valor de ‘InterfaceGroup’ que se encuentra en el registro de grupo de interfaz en la tabla de parámetros.

El valor '--schema\_cdc' debe contener en el nombre del esquema de la base de datos redshift que contiene la tabla p\_cdc y p\_control (como se describe en el índice 2.4)

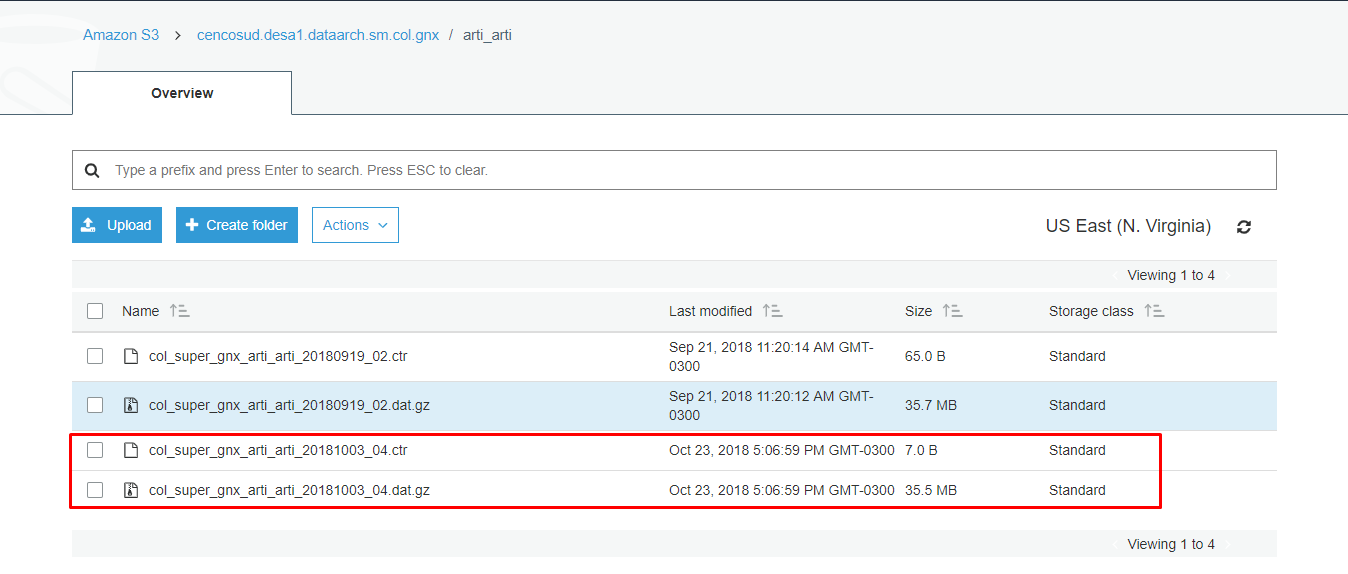


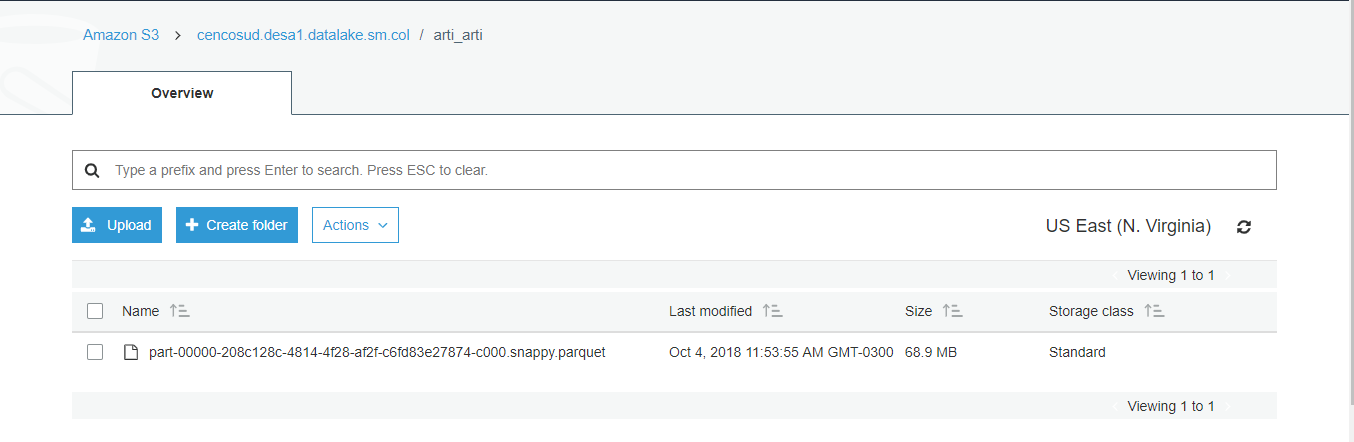
Al final ejecutar y esperar el retorno de la ejecución:



Después del éxito de la ejecución, los archivos que se colocaron en el bucket ‘dataraw‘ se mueven al bucket ‘dataarch’;

se carga el parquet en el bucket ‘datalake’:

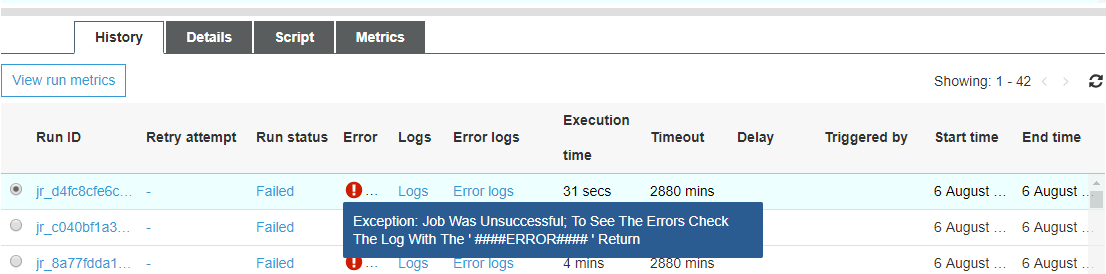




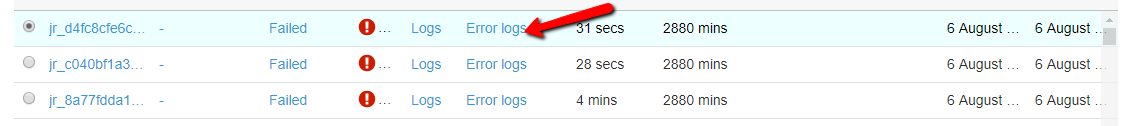
1. La Solución de Problemas

En la ejecución del job glue, al ocurrir errores el procedimiento más eficaz en ese aspecto, se hace de la siguiente manera:

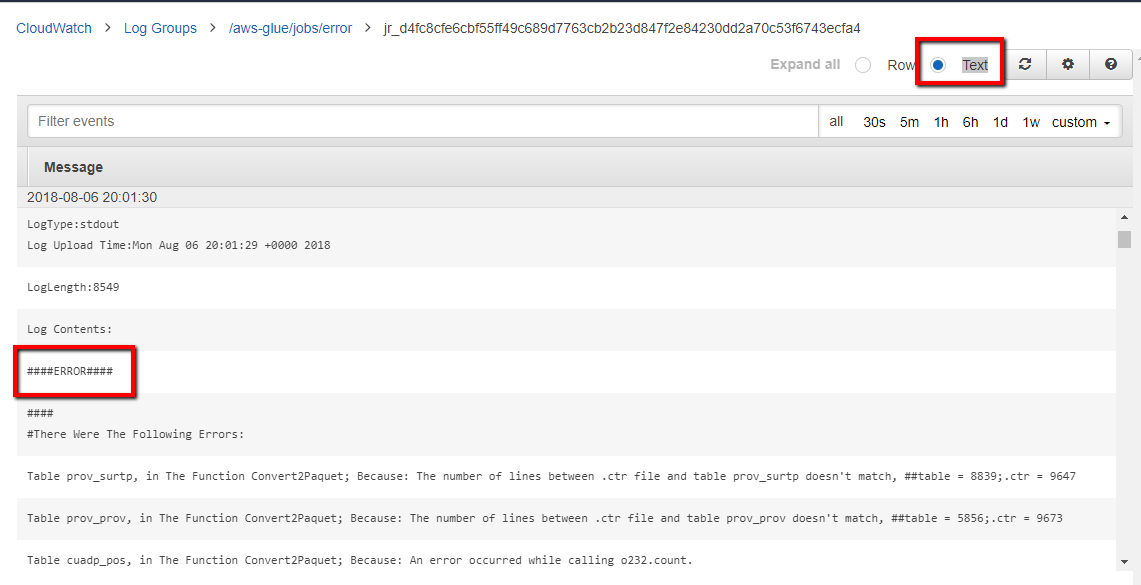
Cuando el job finaliza con error y muestra el siguiente mensaje:



Entre en el link "Error Logs", que mostrará los errores registrados en el CloudWatch:

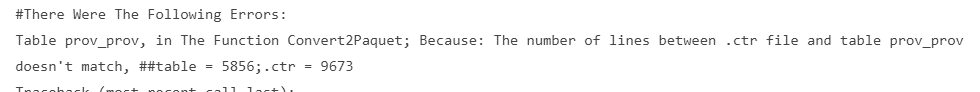


En la consola de CloudWatch seleccionar "Text", y busca el primer registro de "#### ERROR ####", donde encontrará los retornos de error dados por el job:



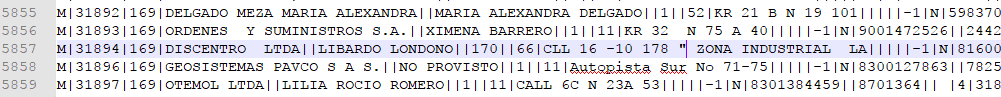
5.1 The number of lines between .ctr file and table [nombre de la tabla] doesn't match

Cuando el job finaliza con error y muestra el siguiente mensaje:



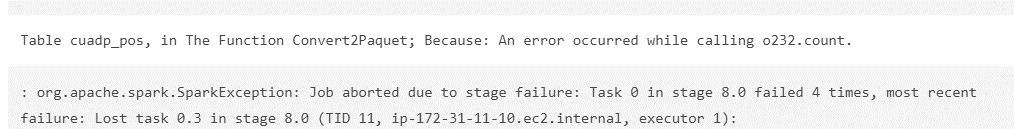
Este error se produce cuando el valor que ha regresado por el .ctr y el número de líneas de la tabla no coinciden, esto es causado por inconsistencia de datos, para resolverlo debe analizarse tanto el .ctr como el .dat correspondiente; este es fácilmente identificado, porque si el problema está en alguna líneas de la tabla, esa recibirá las líneas antes del error.

En este ejemplo en el archivo .dat de la tabla "prob\_prov" en la línea 5857 existe un ", que se debe quitar, que corrompe el separador de columnas (descrito en el índice" 3.1-Tipos de datos ")



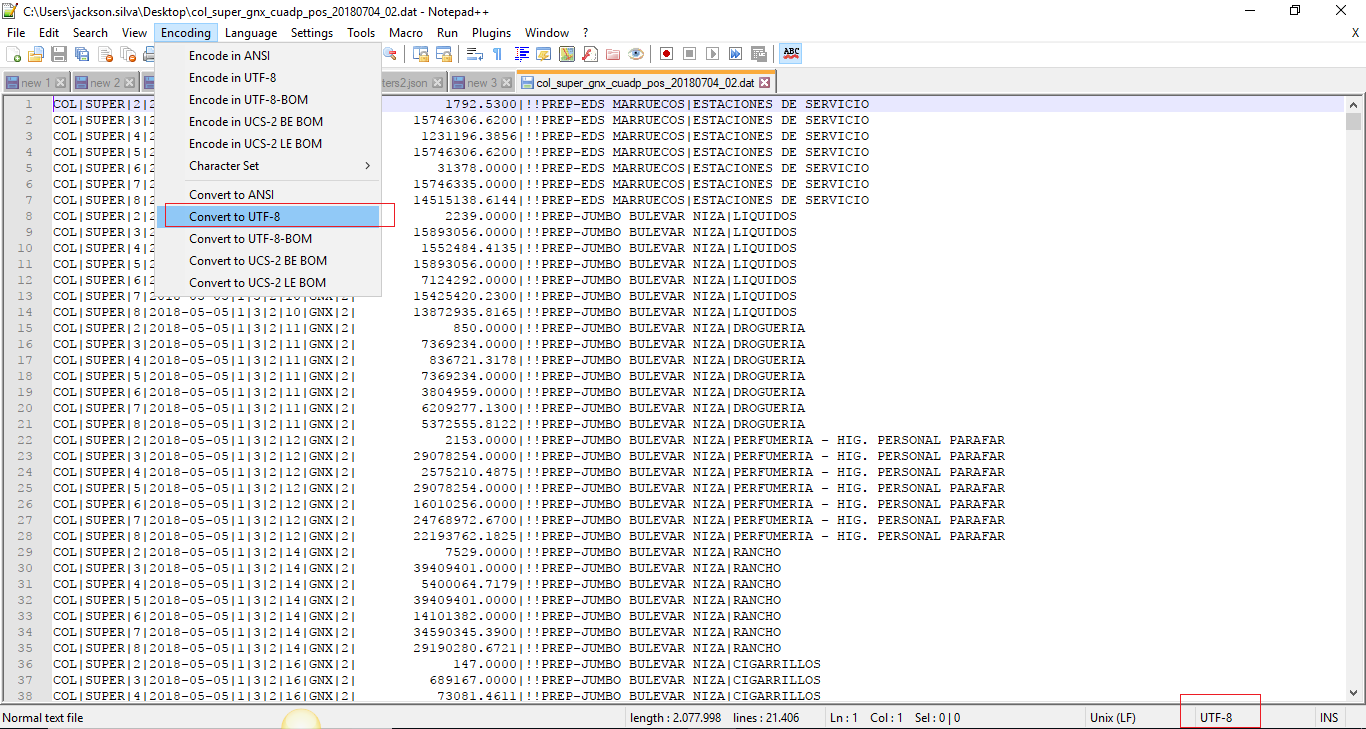
5.2 An error occurred while calling o794.count

Cuando el job finaliza con error y muestra el siguiente mensaje:



Este error es causado porque el job intentó hacer la lectura de un archivo y no ha tenido éxito, que el archivo no fue reconocido o no tiene un encode valido por él.

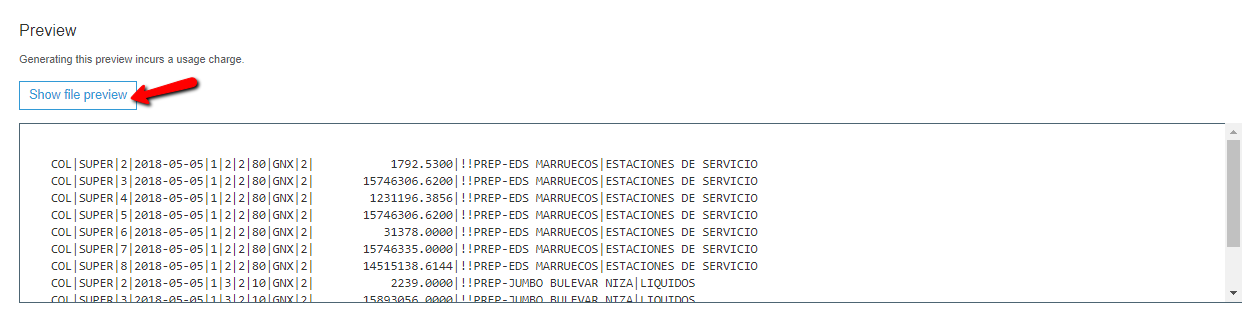
Para solucionar este problema, todos los archivos de lectura .ctr y .dat deben ser convertidos a UTF-8 (descrito en el índice "3.2-Carga de datos")



Para comprobar si no habrá más error, utilice la vista previa de S3 seleccionando el archivo

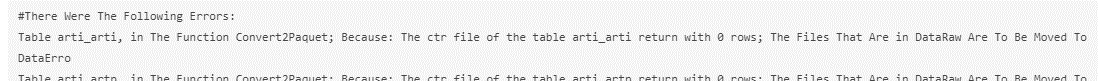


Al presionar sobre la pestaña "Select from", ve al tema de la vista previa, y pulsa el botón "Show file preview" que leerá el archivo.



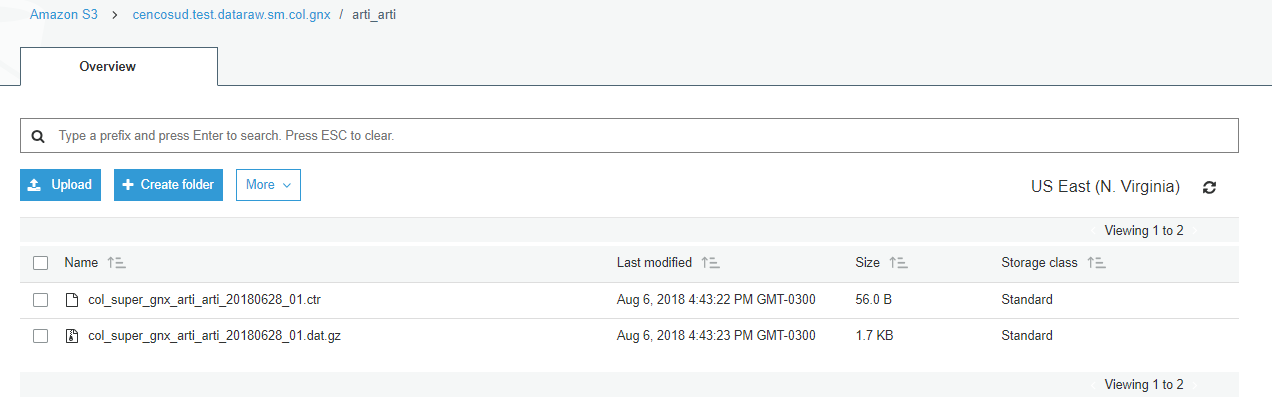
5.3 The ctr file of the table [nombre de la tabla] return with 0 rows

Cuando el job finaliza con error y muestra el siguiente mensaje:



Este problema se debe que la tabla no se cargó porque el archivo .ctr regresó con valor 0.

La solución es comprobar el bucket DataRaw de la tabla que se va a cargar, y ver si hay archivo .dat y un .ctr con el mismo nombre:



El archivo .ctr debe estar con encode UTF-8 y siguiendo uno de los dos ejemplos siguientes:

