Motor de Calidad de Datos

Guía para la actualización motor de calidad.

Módulo constants

El módulo constants contiene todas las constantes con sus respectivos valores.

Rules

La primera constante es la clase Rules, esta contiene distintas subclases las cuales son las reglas y estas subclases contienen tres atributos; el nombre (name), la característica de regla (property) y el código de la regla (code). Si se desean incluir nuevas reglas se debe considerar la creación de una nueva subclase dentro de la clase Rules y se deberán completar sus atributos respectivos.

```
class NullRule:
    name = "Completitud"
    property = "Completitud de Registro"
    code = "101"
```

Json Parts

La clase Json Parts contiene los nombres de los atributos que se esperan extraer de los JSON a los que se haga lectura. Para añadir nuevos atributos solo se debe añadir la variable con el valor tal cual vendrá escrito en el JSON.

```
class JsonParts:
    Input = "INPUT"
```

Fields

Se creó la clase Fields la cual se inicializa con el Nombre del campo y adicionalmente se tiene el método value, el cual permite asignarle un valor a la columna. A continuación se tienen definidos los campos que se utilizan durante el desarrollo de la librería.

```
CountryId = Field("CODIGO_DE_PAIS")
```

Otras constantes

A continuación se definen todos aquellos valores constantes que se utilizarán en el desarrollo del código.

Módulo Rules

En el módulo rules se encuentran definidas todas aquellas funciones de validación que se utilizarán en el código.
Para la creación de nuevas reglas se puede utilizar como parámetros de entrada todos aquellos que se crean necesarios, sin embargo, como condición para el correcto funcionamiento la función debe devolver una lista y el dataframe que contiene los registros con errores.
La lista debe contener los siguientes elementos:
- Cantidad de registros
- Código de regla
- Nombre de regla
- Característica de regla
- Código único de regla (Concatenación de código/entidad/campo)
- Umbral
- Requerimiento de datos
- Campo
- Ratio de éxito de regla
- Cantidad de Registros Fallidos

```
def validateNull(object:DataFrame,field: str,registersAmount: int,entity:
str,threshold):
    dataRequirement = f"El atributo {entity}.{field} debe ser obligatorio (NOT
NULL)."
    errorDf = object.filter(col(field).isNull())
    nullCount = object.select(field).filter(col(field).isNull()).count()
    notNullCount = registersAmount - nullCount
    ratio = (notNullCount/ registersAmount) * OneHundred
    return
[registersAmount,Rules.NullRule.code,Rules.NullRule.name,Rules.NullRule.property,R
ules.NullRule.code + "/" + entity + "/" +
field,threshold,dataRequirement,field,ratio,nullCount], errorDf
```

Módulo Functions

El módulo functions es el módulo principal de la librería y este contiene todas las funcionalidades adicionales para la ejecución de las reglas.

Start Validation

La función start validation es la función principal de la librería desde la cual se inicia el proceso de validación de reglas de calidad. Esta función debe invocar los procesos de lectura de Json obteniendo los parámetros necesarios y a continuación ejecutar la validación. Finalmente se debe escribir el resultado y devolver el dataframe.

Extract Params From Json

Esta fuinción permite obtener los parámetros del archivo de configuración en formato JSON. Este también realiza la lectura del input sobre el cual se hará la validación. Finalmente se debe asegurar de que se devuelven todas las variables que serán utilizadas más adelante en el código.

Read Df

La función readDf permite realizar la lectura de inputs. Esta se va a utilizar tanto para la lectura del input sobre el cual se ejecutan las reglas como para aquellos datasets necesarios para la ejecución de las mismas (Ej: La regla de integridad referencial enfrente un dataset contra otro). En esta función se han definido distintos métodos de lectura, para añadir un nuevo método es necesario la creación de un nuevo elif donde la condición de entrada será el tipo de lectura que vendrá informada desde el JSON con el atributo TYPE en la sección INPUT. La lectura deberá realizarse por medio de métodos de spark y deberá dar como resultado un DataFrame.

Write Df

La función writeDf permite realiza la escritura de los resultados. Esta se utiliza para la escritura de los resultados finales de la validación de calidad y para la escritura de la data observada. En este método, al igual que en el anterior, se pueden agregar nuevos formatos de escrituro los cuales deberán ir dentro de un nuevo elif y tener como condicionante el atributo TYPE de la sección OUTPUT. Se deberán utilizar métodos de spark para la escritura.

Create Error Data

La función createErrorData nos permite la creación del dataframe de Data Observada sobre el cual se anexa la información de los registros que resulten incorrectos en la validación. Esta función obtiene la lista de nombres de columnas y tipos de datos que vienen del dataframe input, añade las columnas que se agregan a la data observada y finalmente crea un dataframe vacío con el esquema específicado.

Validate Rules

Esta es la función que ejecuta la validación de las reglas. Por medio de un loop for se pasa a través de las distintas reglas definidas en el archivo de configuración JSON. La condición de entrada para la ejecución de una regla deben ser los 3 primeros caracteres del código de regla, esto se debe a que una regla puede aparecer más de una vez dentro del JSON.
Todas las reglas empiezan con la medición del tiempo en el que inicia la regla, posteriormente este se restará con el tiempo de finalización para obtener el tiempo de duración de la ejecución de la regla. Luego se hace la invocación a la regla, pasándole los datos necesarios para su correcto funcionamiento. A continuación se valida que la data observada sea mayor a 0 y se procede a anexar las columnas de data observada y finalmente a unir la data observada al dataframe de data observada inicial. Finalmente se escribe el resultado en el dataframe de data.

```
t = time.time()
data, errorDf = validateNull(object,field,registerAmount,entity,threshold)
errorDesc = "Nulos - " + str(field)
if data[-One] > Zero :
    errorTotal = errorDf.withColumn("error", lit(errorDesc))\
    .withColumn("run_time", lit(runTime))
    if write != False :
        errorData = errorData.union(errorTotal)
rulesData.append(data)
print("regla de nulos: %s segundos" % (time.time() - t))
```