**2.- Transforme los datos aplicando reglas como:**

* + **Cambiar el formato de los nombres a mayúsculas.**
  + **Modiﬁcar los correos para usar el dominio example.com**
  + **Agregar un código único por usuario.**

**Cambiar el formato de los nombres a mayúsculas.**

Para transformar los datos según las reglas que has mencionado, puedes usar Python junto con la biblioteca pandas, que es muy útil para la manipulación de datos tabulares.

A continuación, te muestro cómo puedes implementar estas transformaciones en un DataFrame que podría ser una representación de tus datos extraídos de la tabla users.

Código de Ejemplo en Python

Primero, asegúrate de tener instalada la biblioteca pandas. Puedes instalarla usando pip:

bash

pip install pandas

Luego, aquí tienes un ejemplo de cómo se podría implementar:

python

import pandas as pd

import uuid

# Suponiendo que has extraído los datos de la tabla users a un DataFrame

# Aquí te muestro un ejemplo de cómo podría ser ese DataFrame

data = {

'nombre': ['Juan Pérez', 'Ana García', 'Luis Smith'],

'correo': ['juan.perez@correo.com', 'ana.garcia@correo.com', 'luis.smith@correo.com'],

}

# Crear un DataFrame a partir de los datos

df = pd.DataFrame(data)

**# Aplicar las transformaciones**

**# 1. Cambiar el formato de los nombres a mayúsculas**

df['nombre'] = df['nombre'].str.upper()

**# 2. Modificar los correos para usar el dominio example.com**

df['correo'] = df['correo'].apply(lambda x: f"{x.split('@')[0]}@example.com")

**# 3. Agregar un código único por usuario**

df['codigo\_unico'] = [str(uuid.uuid4()) for \_ in range(len(df))]

# Mostrar el DataFrame transformado

print(df)

Explicación de las Transformaciones:

Cambiar el formato de los nombres a mayúsculas :

Se usa str.upper() para convertir todos los caracteres de la columna nombre a mayúsculas.

Modificar los correos para usar el dominio example.com :

La función apply se utiliza para cambiar el dominio de cada correo.

Se divide el correo en el símbolo @ y se conserva solo la parte anterior para luego unirla con @example.com.

Agregar un código único por usuario :

Se utiliza uuid.uuid4() para generar un UUID único para cada usuario en la columna codigo\_unico.

Salida

La salida del print(df) mostrará el DataFrame transformado con los nombres en mayúsculas, los correos con el nuevo dominio y un código único para cada usuario.

Notas Adicionales

Asegúrate de ajustar las columnas y tipos de datos según los datos reales que estés utilizando.

Si estás trabajando directamente con datos extraídos de PostgreSQL, podrías incorporar estas transformaciones después de haber cargado los datos en un DataFrame utilizando pandas.

1. **Cargue los datos transformados en una nueva colección llamada users\_transformed en MongoDB.**

Para cargar los datos transformados en una nueva colección de MongoDB llamada users\_transformed, puedes usar la biblioteca pymongo en Python. A continuación, te muestro cómo hacerlo. Este ejemplo asume que ya tienes el DataFrame resultante de las transformaciones que hiciste en el paso anterior.

Paso 1: Instalar PyMongo

Si no lo tienes instalado, puedes instalar pymongo utilizando pip:

bash

pip install pymongo

Paso 2: Código para Cargar los Datos en MongoDB

Aquí hay un ejemplo completo que incluye la conexión a MongoDB y la inserción de los datos transformados:

python

import pandas as pd

import uuid

from pymongo import MongoClient

# Supongamos que ya tienes el DataFrame 'df' transformado

data = {

'nombre': ['Juan Pérez', 'Ana García', 'Luis Smith'],

'correo': ['juan.perez@example.com', 'ana.garcia@example.com', 'luis.smith@example.com'],

}

df = pd.DataFrame(data)

# Agregar un código único

df['codigo\_unico'] = [str(uuid.uuid4()) for \_ in range(len(df))]

# Conexión a la base de datos MongoDB

client = MongoClient('mongodb://localhost:27017/') # Cambia la URI según tu configuración

db = client['tu\_basededatos'] # Cambia el nombre de la base de datos

collection = db['users\_transformed'] # Nombre de la colección

# Convertir DataFrame a diccionarios para insertar en MongoDB

data\_to\_insert = df.to\_dict(orient='records')

# Insertar los datos transformados en la colección

result = collection.insert\_many(data\_to\_insert)

# Confirmar la inserción

print(f'Documentos insertados: {len(result.inserted\_ids)}')

Explicación del Código:

Conexión a MongoDB :

Se establece una conexión a través de MongoClient. Asegúrate de que la URI sea correcta y que el servidor de MongoDB esté activo.

Seleccionar la Base de Datos y la Colección :

Cambia 'tu\_basededatos' por el nombre de tu base de datos en MongoDB. Se crea o se accede a la colección llamada users\_transformed.

Convertir el DataFrame a Diccionarios :

Utiliza df.to\_dict(orient='records') para convertir el DataFrame a una lista de diccionarios, donde cada diccionario representa un documento en MongoDB.

Insertar Documentos en MongoDB :

Se insertan los datos en la colección mediante el método insert\_many.

Confirmar la Inserción :

Se imprime el número de documentos insertados para confirmar que la operación se realizó con éxito.