Módulo 5
Integrador de ETL

Romina Barrabino

1) Extracción de Datos:

```python
#1) Extracción de datos

#Importo Pandas y numpy
import pandas as pd

#Supongo la extracción de datos desde un archivo CSV y JSON

csv_Asistencias={
            'Id_Numero':[1,2,3,4,5],
            'Fecha':['2022/01/03','2022/02/03','2023/05/10','2024/03/11','2024/01/31'],
            'Nombre':['Susana','Pablo','Teresa','Doña Susana','Juan'],
            'Texto':['Asistio','Falto','Licencia','Falto','Vacaciones']
        }

json_AsistenciasNuevas={
            'Id_Numero':[6,7],
            'Fecha':['2024/07/28','2025/01/31'],
            'Nombre':['Le.onel','Pablo']
            'Texto':['Licencia','Asistio']
            }

```

2) Transformación de Datos:

```python
# 2) Trasformación de datos:

#Creo un DataFrames desde csv_Asistencias y json_AsistenciasNuevas
df_csv=pd.DataFrames(csv_Asistencias)
df_json=pd.DataFrame(json_AsistenciasNuevas)
#Uno los Dataframes
df=pd.concat([df_csv,df_json],ignore_indexTrue)

#Instalo e importo Data prep.clean
from dataprep.clean import clean_date

#a)Limpieza de valores nulos:
#Elimino las filas que tengan altos porcentajes de nulos (0.8)
df.dropna(axis=0,inplace=True,thresh=int(df.shape[1]*0.8))

#b) Normalización de nombres de columnas:
#Elimino los duplicados de la columna Id_Numero
data_cleaned=df.drop_duplicates(subset=['Id_Numero'])
#Elimino los espacios en blanco de la columna Nombre
data_cleaned['Nombre']=data_cleaned['Nombre'].apply(lambda x:str(x).strip(x))
#Pongo en mayuscula la primer letra y minuscula en el resto los registros de la columna Nombre
data_cleaned['Nombre']=data_cleaned.['Nombre'].apply(lambda x:str(x).strip().title())

```

```
#b)Conversión de tipos de datos:
#Modifico el tipo de datos si no son correcto
data_cleaned=data_cleaned.astype({
    'Id_Numero':'Int',
    'Nombre':'Varchar(30)',
    'Fecha':'Date',
    'Texto':'Varchar(50)'})
#Convierto la columna Fechas al formato YYYY-MM-DD
data_cleaned['Fecha']=pd.to_datatime(data_cleaned['Fecha'], errors='coerce',format='%Y-%m-%d')

```

3) Carga de Datos:

```
#3) Cargo los datos:
#Llamo a data_cleaned como df
df=data_cleaned

print("\nDatos Cargados (Centralizados):")
print(df)
```

4) Implementación de Pruebas Automatizadas:

```
#4)Implementación de pruebas automaticas

#Importo Great expectations
import great_expectations as ge
```

```
#Convierto el Dataframe df en un objeto Great Expectations
df_ge=ge.from_pandas(data_loaded)

# Expectativa1
df_ge.expect_column_values_to_not_be_null('Id_Numero')
# Expectativa2
df_ge.expect_column_values_to_not_be_null('Nombre')
# Expectativa3
df_ge.expect_column_values_to_be_unique('Id_Numero')
# Expectativa4
df_ge.expect_column_values_to_be_of_type('Texto','varchar(50)')
# Expectativa5
df_ge.expect_column_values_to_match_strftime_format('Fecha','%Y-%m-%d')

#Ejecución de pruebas automatizadas
validation_results=df_ge.validate()
print("\nResultados de las Validaciones:")
print(validation_results)
#Como no pude descargar Great expectations no puedo visualizar los resultados que arroja, pero los detalle con el objetivo de que no resulten en error.

```

5) Reporte de Resultados:

```
85
86  #5)Reporte de resultados:
87
88  #Exporto un reporte de las validaciones a HTML
89  df_ge.save_expectation_suite(discard_failed_expectations=False)
90  #Genero un reporte de resultados
91  validation_report=df_ge.validate()
92  print("\nGeneración de reporte completada.")
93  --------
```

6) **Documentación del Proceso:**

- Las herramientas utilizadas fueron la página de Databricks, pandas y data prep.clean.
- Los tipos de pruebas implementadas están adjuntadas en la sección 4.
- No adjunte ninguna explicación de los resultados, ya que como no pude descargar Great Expectations no pude verificar los resultados. De igual manera, las expectativas se realizaron con el objetivo de que no devuelvan fallas ni errores.