

# Romina Barrabino

Modulo 5 - Integrador Python

Archivo Editar Ver Ejecutar Ayuda Última edición hace 7 minutos

Ejecutar todo

11:58 (<1 s)1

#1) Extracción de datos

#Importo Pandas

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:

▶ ✓ 11:58 (<1 s) 1

# 2) Transformación de datos:  
  
#Creo un DataFrames desde csv\_Astencias y json\_AstenciasNuevas  
df\_csv=pd.DataFrame(csv\_Astencias)  
df\_json=pd.DataFrame(json\_AstenciasNuevas)  
#Uno los Dataframes  
df=pd.concat([df\_csv,df\_json],ignore\_index=True)  
  
#Instalo e importo Data prep.clean

▶ ✓ 11:10 (3 s) 2

!pip install dataprep  
  
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 0.5.13  
Requirement already satisfied: beautifulsoup4 in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 4.11.1  
Requirement already satisfied: jupyterlab-pygments in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 0.1.2  
Requirement already satisfied: mistune<2,>=0.8.1 in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 0.8.4  
Requirement already satisfied: nbconvert<6.0,>=5.6.0 in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 5.6.0  
Requirement already satisfied: ipywidgets<8.0,>=7.5 in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 7.6.0  
Requirement already satisfied: dataprep in /databricks/python3/lib/python3.9/site-packages (from nbconvert->notebook) 0.5.13

▶ ✓ 11:58 (<1 s) 3

from dataprep.clean import clean\_date

▶ ✓ 11:58 (<1 s) 4

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

▶ ✓ 11:58 (<1 s) 5

#b) Normalización de nombres de columnas:  
#Elimino los "." y "Doña" de los registros de la columna Nombre  
df['Nombre'] = df['Nombre'].apply(lambda nombre: nombre.strip().capitalize().replace('Doña','').replace('.', ''))  
  
print(df) #Verificación

▶ ✓ 11:58 (<1 s)

6

```
#Elimino los duplicados de la columna Id_Numero
data_cleaned=df.drop_duplicates(subset=['Id_Numero'])
print(data_cleaned) #Verificación
```

	Id_Numero	Fecha	Nombre	Texto
0	1	2022/01/03	Susana	Asistio
1	2	2022/02/03	Pablo	Falto
2	3	2023/05/10	Teresa	Licencia
3	4	2024/03/11	susana	Falto
4	5	2024/01/31	Juan	Vacaciones
5	6	2024/07/28	Leonel	Licencia
6	7	2025/01/31	Pablo	Asistio

▶ ✓ 11:59 (<1 s)

7

```
#Elimino los espacios en blanco de la columna Nombre
data_cleaned['Nombre']=data_cleaned['Nombre'].apply(lambda x: str(x).strip().title())
print(data_cleaned) #Verificación
```

▶ ✓ 11:59 (<1 s)

8

```
#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())
print(data_cleaned) #Verificación
```

▶ ✓ 11:59 (<1 s)

9

```
#c)Conversión de tipos de datos:
#Modifico el tipo de datos si no son correcto
data_cleaned = data_cleaned.astype({
    'Id_Numero': 'Int64',
    'Nombre': 'str',
    'Fecha': 'datetime64',
    'Texto': 'str'
})
#Uso unas modificaciones sugeridas int=Int64,Varchar(30)=str,Date=datetime64

#Convierto la columna Fechas al formato YYYY-MM-DD
data_cleaned['Fecha'] = pd.to_datetime(data_cleaned['Fecha'], errors='coerce', format='%Y-%m-%d')

print(data_cleaned) #Verificación
```

### 3) Carga de Datos:

```
▶ ✓ 12:00 (<1 s)

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

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

Datos Cargados (Centralizados):

	Id_Numero	Fecha	Nombre	Texto
0	1	2022-01-03	Susana	Asistio
1	2	2022-02-03	Pablo	Falto
2	3	2023-05-10	Teresa	Licencia
3	4	2024-03-11	Susana	Falto
4	5	2024-01-31	Juan	Vacaciones
5	6	2024-07-28	Leonel	Licencia
6	7	2025-01-31	Pablo	Asistio

## 4) Implementación de Pruebas Automatizadas:

```
11
#4)Implementación de pruebas automaticas

#Importo Great expectations, se verifico y se importo
!pip install great_expectations

Requirement already satisfied: pytz>=2020.1 in /databricks/python3/lib/python3.9/site-packages (from pandas<2.2,>=1.1.3->great_expectations) (2021.3)
Requirement already satisfied: six>=1.5 in /databricks/python3/lib/python3.9/site-packages (from posthog<4,>3->great_expectations) (1.16.0)
Requirement already satisfied: monotonic>=1.5 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-9d561a0c-c321-4c7c-8509-1d3195fc08f4/lib/python3.9/site-packages (from posthog<4,>3->great_expectations) (1.6)
```

```
12
!pip show great_expectations

Name: great-expectations
Version: 1.3.5
Summary: Always know what to expect from your data.
Home-page: https://greatexpectations.io
Author: The Great Expectations Team
Author-email: team@greatexpectations.io
License: Apache-2.0
Location: /local_disk0/.ephemeral_nfs/envs/pythonEnv-9d561a0c-c321-4c7c-8509-1d3195fc08f4/lib/python3.9/site-packages
Requires: python-dateutil, typing-extensions, numpy, tqdm, mistune, scipy, pyparsing, ruamel.yaml, marshmallow, jsonschema, requests, tzlocal, Jinja2, posthog, pandas, packaging, altair, cryptography, pydantic
Required-by:
```

```
13
import great_expectations as ge
```


```
14
#Última ejecución fallida

1 #Convierto el Dataframe df en un objeto Great Expectations
2 df_ge=ge.from_pandas(df)
3
4 # Expectativa1
5 df_ge.expect_column_values_to_not_be_null('Id_Numero')
6 # Expectativa2
7 df_ge.expect_column_values_to_not_be_null('Nombre')
8 # Expectativa3
9 df_ge.expect_column_values_to_be_unique('Id_Numero')
10 # Expectativa4
11 df_ge.expect_column_values_to_be_of_type('Texto','str')
12 # Expectativa5
13 df_ge.expect_column_values_to_match_strftime_format('Fecha','%Y-%m-%d')
14
15 #Ejecución de pruebas automatizadas
16 validation_results=df_ge.validate()
17 print("\nResultados de las Validaciones:")
18 print(validation_results)
19 #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.
```

AttributeError: module 'great\_expectations' has no attribute 'from\_pandas'

## 5) Reporte de Resultados:

 > **AttributeError:** module 'great\_expectations' has no attribute 'from\_pandas'

 15

```
#5)Reporte de resultados:

#Exporto un reporte de las validaciones a HTML
df_ge.save_expectation_suite(discard_failed_expectations=False)
#Genero un reporte de resultados
validation_report=df_ge.validate()
print("\nGeneración de reporte completada.")
```

## 6) Documentación del Proceso:

- Las herramientas utilizadas fueron la página de Databricks, Pandas, Data prep y Great expectations.
- Los tipos de pruebas implementadas están adjuntadas en la sección 4, aunque debido a que no se pudo utilizar Great Expectations con pandas, no pude verificar los resultados de las expectativas. De igual manera, las expectativas se realizaron con el objetivo de que no devuelvan fallas ni errores.