

Corporación Universitaria Minuto de Dios - UNIMINUTO

Facultad Ingeniería

Programa Ingeniería de sistemas

Arquitectura de datos – NRC 73457

Informe ETL con Python

Autores:

Jeferson Arias Alape – ID: 888243 – Semestre V

Presentado a

Elkin Leandro Velásquez Pesca

Bogotá DC, abril de 2025

1. OBJETIVO

Implementar un proceso ETL (Extract, Transform, Load) que lea datos desde un archivo externo y en Python realice las operaciones de limpieza y transformación en un conjunto de datos desordenados y con problemas comunes como valores faltantes, formatos inconsistentes y datos erróneos. Finalmente, guarde el resultado en un nuevo archivo externo (.csv).

2. CÓDIGO GENERAL

2.1 Librerías utilizadas

```
1 import gender_guesser.detector as gender
2 from text_to_num import text2num as txt
3 import pandas as pd
4 import numpy as np
```

2.1.1 Sustentación del código

La librería *gender_guesser.detector* Se utiliza en esta ocasión, con el propósito de que saber cual es el genero de una persona con base a su nombre.

La librería *text2Num* Se utiliza para convertir textos numéricos a valores numéricos.

La librería *pandas* se estará utilizando para Leer, manipular y analizar datasets.

La librería *numpy* se estará utilizando para manipular datos en arreglos y vectores.

Estas librerías se han implementado con el objetivo de transformar distintos valores importados desde un archivo Excel, ya que proporcionan herramientas clave para llevar a cabo el proceso de transformación de datos.

2.2 Funciones declaradas

2.2.1 limpiar Edad

```
def limpiar_edad(valor):  
    valor = str(valor).strip().lower()  
    try:  
        return pd.to_numeric(valor, errors='raise')  
    except:  
        try:  
            return txt(valor, "es")  
        except:  
            return np.nan
```

Esta función permite transformar datos numéricos escritos en caracteres “treinta” y pasarlos a valores numéricos. En primera instancia, el valor analizado pasara a ser de tipo string, al pasar por el primer control, si este tiene valores numéricos “45” o “32” este retornara dicho valor en formato número, si se presenta una falla, este se pasará a procesar como texto para luego ser convertido en número, encaso de que este falle también, entonces el valor asignado será un valor vacío.

2.2.2 Procesar fechas

```
def procesarFecha(date_str):  
    date_str = str(date_str).strip()  
  
    if len(date_str) >= 4 and date_str[0:4].isdigit():  
        return pd.to_datetime(date_str, errors='coerce', format='%Y-%m-%d')  
    else:  
        return pd.to_datetime(date_str, errors='coerce', format='%d-%m-%Y')
```

Cuando se realiza la lectura de datos por fecha, en algunos momentos son tomados bajo el formato AAAA-MM-DD por defecto, en otros casos se capturaron bajo el formato DD/MM/AAAA, al intentar dar formato para que este sea de tipo fecha, se eliminaban algunos datos que se tenían, mediante esta función, se compara la longitud del primer carácter del registro, si este es mayor o igual a 4, se le dará un formato teniendo en cuenta que dicho dato tiene esa estructura AAAA-MM-DD, de lo contrario, se formateará los datos correspondientes, lo que se estará realizando, mediante panda, definir cual es el día, mes, año, para luego al mostrar o exportar, todos mantengan la misma estructura lógica.

2.2.3 Deducción de género

```
def prediccionGenero(row, detector):  
    if pd.isna(row['Género']) or row['Género'] not in ['M', 'F']:  
        try:  
            nombre = str(row['Nombre']).split()[0]  
            nombre = str(nombre).strip()  
  
            if len(nombre) > 0:  
                genero_predicho = detector.get_gender(nombre)  
  
                if genero_predicho in ['male', 'mostly_male']:  
                    return 'M'  
                elif genero_predicho in ['female', 'mostly_female']:  
                    return 'F'  
  
            return np.nan  
        except Exception as e:  
            print(f"Error procesando el nombre: {row.get('Nombre', 'N/A')}. Error: {e}")  
            return np.nan  
    else:  
        return row['Género']
```

Esta función se utiliza con el fin, de que si el paciente registrado por error se ingreso otro tipo de valor diferente a M o F, entonces, juzgando con el nombre, se le asignara el genero que le corresponda, esto se logra mediante una librería existente, Primero comprueba si el valor esta vacío o es diferente a M o F, con base al primer nombre, es procesado si este es mayor a 0 caracteres, si el resultado al procesase es MALE o MOSTLY_MALE o viceversa, entonces este será asignado al género que le corresponde, en caso de que en la librería no se logro encontrar dicho nombre, entonces este dato será vacío.

2.2.4 Tratamiento a los números telefónicos

```
def limpiar_telefono(telefono):  
    telefono = str(telefono).strip()  
    solo_numeros = ''.join(c for c in telefono if c.isdigit())  
  
    if not solo_numeros:  
        return np.nan  
  
    if len(solo_numeros) == 10:  
        return solo_numeros  
  
    elif len(solo_numeros) == 11 and solo_numeros.startswith('1'):  
        return solo_numeros[1:] # Quitar el código de país  
  
    else:  
        return np.nan
```

En la primera línea, elimina todo lo que se contempla como texto y espacios, en la segunda línea elimina todos los caracteres, luego este será procesado por la estructura de control IF, si este dato está vacío, retornara como valor vacío, Si el dato solo tiene 10 dígitos, entonces este valor será retornado, en caso de que tenga 11 caracteres e inicie con uno, se eliminara este, pues es el indicativo del código del país, en caso de no haber pasado en alguna de estas estructuras de control entonces retornara un valor vacío.

2.2.5 Función Separa valores

```
def separar_presion(presion):  
    if isinstance(presion, str) and '/' in presion:  
        try:  
            sistolica, diastolica = presion.split('/')  
            return pd.Series([int(sistolica), int(diastolica)])  
        except:  
            return pd.Series([np.nan, np.nan])  
    return pd.Series([np.nan, np.nan])
```

Primero se comprueba en que en la columna presión, el dato se encuentre separado por un “/”, de ser afirmativo separa estos dos valores en enteros y los retornara, de lo contrario quedarán como valores vacíos.

2.2.6 Función de clasificar el nivel de presión

```
def clasificar_presion(sis, dia):  
    if pd.isna(sis) or pd.isna(dia):  
        return np.nan  
    if sis < 120 and dia < 80:  
        return 'Normal'  
    elif 120 <= sis < 130 and dia < 80:  
        return 'Elevada'  
    elif 130 <= sis < 140 or 80 <= dia < 90:  
        return 'Hipertensión Grado 1'  
    elif sis >= 140 or dia >= 90:  
        return 'Hipertensión Grado 2'  
    else:  
        return 'Desconocida'
```

Mediante los valores capturados y la estructura de control IF se determina cual será el valor que tendrá cada paciente de acuerdo con la presión, este compara los datos que se tienen almacenados en dos columnas diferentes.

2.3 Función de contadores

```
def inicializar_reporte(df):  
    return {  
        'Edad_invalidas': df['Edad'].isna().sum(), # Antes de limpiar  
        'Edad_transformadas': 0,  
        'Fechas_invalidas': 0,  
        'Fechas_nulas': 0,  
        'Genero_nulo': 0,  
        'Peso_nulo': 0,  
        'Altura_nula': 0,  
        'Telefono_nulo': 0,  
        'Presion_nula': 0,  
        'Nombre_autogenerado': 0,  
        'IMC_nulo': 0,  
        'Clasificacion_Presion_nulo': 0  
    }  
  
def mostrar_reporte(reporte_dict):  
    reporte_df = pd.DataFrame.from_dict(reporte_dict, orient='index', columns=['Cantidad'])  
    print("\n📄 Reporte de Transformaciones de Datos:")  
    print(reporte_df)  
    return reporte_df  
  
def exportar_reporte_csv(reporte_dict, ruta='reporte_transformaciones.csv'):  
    reporte_df = pd.DataFrame.from_dict(reporte_dict, orient='index', columns=['Cantidad'])  
    reporte_df.to_csv(ruta)  
    print(f"\n📄 Reporte exportado a {ruta}")
```

2.4 Código general

```
1  # Importación de librerías
2  from imports import *
3
4  # Importar funciones definidas en otro archivo
5  from funciones import *
6  from reporte_transformaciones import *
7
8  # 2. Extracción de datos
9  df = pd.read_csv('pacientes_sucio.csv', encoding='latin1')
10
11  pd.set_option('display.max_columns', None)
12  print("Primeras filas del dataset original:")
13  print(df.head())
14
15  pd.set_option('display.max_rows', None)
16  print("Se muestran todas los registros del dataset original:")
17  print(df)
18
19  reporte = inicializar_reporte(df)
20
21  # 3. Proceso tratamiento de datos
22  # 3.1 Registros duplicados
23  df = df.drop_duplicates()
24
25  # 3.2 Limpieza columnas
26  # 3.2.1 Limpieza de edad
27  edad_antes = df['Edad'].copy()
28  df['Edad'] = df['Edad'].apply(limpiar_edad)
29  df['Edad'] = df['Edad'].apply(lambda x: abs(x) if x < 120 else np.nan)
30  reporte['Edad_transformadas'] = (edad_antes != df['Edad']).sum()
31  reporte['Edad_invalidas'] = df['Edad'].isna().sum()
```




```
33 # 3.2.2 Estandarizar fechas
34 fechas_antes = df['Fecha_Consulta'].copy()
35 df['Fecha_Consulta'] = df['Fecha_Consulta'].astype(str).str.replace(r"[./]", "-", regex=True)
36 df['Fecha_Consulta'] = df['Fecha_Consulta'].apply(procesarFecha)
37 reporte['Fechas_invalidas'] = (fechas_antes != df['Fecha_Consulta']).sum()
38 reporte['Fechas_nulas'] = df['Fecha_Consulta'].isna().sum()
39
40 # 3.2.3 Columnas numéricas (Peso - altura)
41 df['Peso_kg'] = pd.to_numeric(df['Peso_kg'], errors='coerce')
42 df['Altura_cm'] = pd.to_numeric(df['Altura_cm'], errors='coerce')
43 reporte['Peso_nulo'] = df['Peso_kg'].isna().sum()
44 reporte['Altura_nula'] = df['Altura_cm'].isna().sum()
45
46 # 3.2.4 Limpieza de Género
47 detector = gender.Detector(case_sensitive=False)
48 df['Género'] = df['Género'].str.upper().str.strip()
49 df['Género'] = df.apply(lambda row: prediccionGenero(row, detector), axis=1)
50 reporte['Genero_nulo'] = df['Género'].isna().sum()
51
52 # 3.2.5 Estandarizar Teléfonos
53 df['Teléfono'] = df['Teléfono'].apply(limpiar_telefono)
54 reporte['Telefono_nulo'] = df['Teléfono'].isna().sum()
55
56 # 3.2.6 Separar Presión arterial:
57 df[['Presion_Sistolica', 'Presion_Diastolica']] = df['Presión_Arterial'].apply(separar_presion)
58 reporte['Presion_nula'] = df['Presion_Sistolica'].isna().sum() + df['Presion_Diastolica'].isna().sum()
59
60 # 3.2.7 Manejar nombres faltantes
61 nombres_antes = df['Nombre'].isna().sum()
62 df['Nombre'] = df['Nombre'].fillna('Paciente_' + df['ID_Paciente'].astype(str))
63 reporte['Nombre_autogenerado'] = nombres_antes
64
65 # 3.3 Calculamos campos derivados
66 # 3.3.1 IMC
67 df['IMC'] = df['Peso_kg'] / (df['Altura_cm'] / 100) ** 2
68 reporte['IMC_nulo'] = df['IMC'].isna().sum()
69
70 # 3.3.2 Clasificación Presión Arterial
71 df['Clasificacion_Presion'] = df.apply(lambda x: clasificar_presion(x['Presion_Sistolica'], x['Presion_Diastolica']), axis=1)
72 reporte['Clasificacion_Presion_nulo'] = df['Clasificacion_Presion'].isna().sum()
73
74 # mostrar resumen general
75 mostrar_reporte(reporte)
76 print(pd)
77
78 # Exportar resultados
79 df.to_csv('pacientes_limpio.csv', index=False, encoding='utf-8-sig')
80 exportar_reporte_csv(reporte)
```

2.4.1 Extracción de datos

```
# 2. Extracción de datos
df = pd.read_csv('pacientes_sucio.csv', encoding='latin1')

pd.set_option('display.max_columns', None)
print("Primeras filas del dataset original:")
print(df.head())

pd.set_option('display.max_rows', None)
print("Se muestran todas los registros del dataset original:")
print(df)
```

Se importan los datos que se encuentran almacenados en un archivo csv, como este tiene caracteres especiales, se opta por utilizar latin1, se muestra luego los datos de todas las columnas con sus 5 primeros registros, luego todos los registros.

RESULTADO

Los 5 primeros registros

Primeras filas del dataset original:

ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg \
0	P001 Juan Pérez	35	M	2023-01-15	70.5
1	P002 María González	28	F	15/02/2023	58.2
2	P003 NaN	42	M	2023.03.10	81.7
3	P004 Carlos Ruiz treinta	M	2023-04-22	75.0	
4	P005 Ana López	29	F	23/05/2023	62.3

	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico
0	172	120/80	+1 555-123-4567	juan@email.com	Hipertensión
1	165	110/70	5552345678	maria@email.com	Diabetes
2	178	130/85	(555) 345-6789	carlos@email.com	Asma
3	170	140/90	555.456.7890	cr@email.com	Hipertensión
4	160	115/75	555-567-8901	ana@email.com	Migraña

Todos los registros existentes

Se muestran todas los registros del dataset original:

ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg \
0	P001 Juan Pérez	35	M	2023-01-15	70.5
1	P002 María González	28	F	15/02/2023	58.2
2	P003 NaN	42	M	2023.03.10	81.7
3	P004 Carlos Ruiz treinta	M	2023-04-22	75.0	
4	P005 Ana López	29	F	23/05/2023	62.3
5	P001 Juan Pérez	35	M	2023-01-15	70.5
6	P006 Luisa Fernández	50	F	10-06-2023	68.9
7	P007 Pedro Martínez	33	M	2023/07/18	85.1
8	P008 Sofía Vargas	-5	F	2023-08-30	59.8
9	P009 Jorge Silva	41	X	2023-09-05	72.4

	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico
0	172	120/80	+1 555-123-4567	juan@email.com	Hipertensión
1	165	110/70	5552345678	maria@email.com	Diabetes
2	178	130/85	(555) 345-6789	carlos@email.com	Asma
3	170	140/90	555.456.7890	cr@email.com	Hipertensión
4	160	115/75	555-567-8901	ana@email.com	Migraña
5	172	120/80	+1 555-123-4567	juan@email.com	Hipertensión
6	168	125/82	5556789012	luisa@email.com	Diabetes
7	182	135/88	NaN	pedro@email.com	Artritis
8	163	118/78	5557890123	sofia@email.com	Depresión
9	175	155/102	5558901234	jorge@email.com	Hipertensión

2.4.2 Proceso de tratamiento de datos

2.4.2.1 Limpieza de duplicados

```
df = df.drop_duplicates()
```

Esta línea, se encarga de eliminar todos los duplicados existentes, en caso de encontrar una fila que se repite en otra fila, pues dejara solo una fila, las demás serán eliminadas

RESULTADO

antes										
ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico
0	P001	Juan Pérez	35	M	2023-01-15	70.5	172	120/80	+1 555-123-4567	juan@email.com Hipertensión
1	P002	María González	28	F	15/02/2023	58.2	165	110/70	5552345678	maria@email.com Diabetes
2	P003	NaN	42	M	2023-03-10	81.7	178	130/85	(555) 345-6789	carlos@email.com Asma
3	P004	Carlos Ruiz	treinta	M	2023-04-22	75.0	170	140/90	555.456.7890	cr@email.com Hipertensión
4	P005	Ana López	29	F	23/05/2023	62.3	160	115/75	555-567-8901	ana@email.com Migraña
5	P001	Juan Pérez	35	M	2023-01-15	70.5	172	120/80	+1 555-123-4567	juan@email.com Hipertensión
6	P006	Luisa Fernández	50	F	10-06-2023	68.9	168	125/82	5556789012	luisa@email.com Diabetes
7	P007	Pedro Martínez	33	M	2023/07/18	85.1	182	135/88	NaN	pedro@email.com Artritis
8	P008	Sofía Vargas	-5	F	2023-08-30	59.8	163	118/78	5557890123	sofia@email.com Depresión
9	P009	Jorge Silva	41	X	2023-09-05	72.4	175	155/102	5558901234	jorge@email.com Hipertensión
despues										
ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico
0	P001	Juan Pérez	35	M	2023-01-15	70.5	172	120/80	+1 555-123-4567	juan@email.com Hipertensión
1	P002	María González	28	F	15/02/2023	58.2	165	110/70	5552345678	maria@email.com Diabetes
2	P003	NaN	42	M	2023-03-10	81.7	178	130/85	(555) 345-6789	carlos@email.com Asma
3	P004	Carlos Ruiz	treinta	M	2023-04-22	75.0	170	140/90	555.456.7890	cr@email.com Hipertensión
4	P005	Ana López	29	F	23/05/2023	62.3	160	115/75	555-567-8901	ana@email.com Migraña
6	P006	Luisa Fernández	50	F	10-06-2023	68.9	168	125/82	5556789012	luisa@email.com Diabetes
7	P007	Pedro Martínez	33	M	2023/07/18	85.1	182	135/88	NaN	pedro@email.com Artritis
8	P008	Sofía Vargas	-5	F	2023-08-30	59.8	163	118/78	5557890123	sofia@email.com Depresión
9	P009	Jorge Silva	41	X	2023-09-05	72.4	175	155/102	5558901234	jorge@email.com Hipertensión

2.4.2.2 Limpieza de edad

```
edad_antes = df['Edad'].copy()
df['Edad'] = df['Edad'].apply(limpiar_edad)
df['Edad'] = df['Edad'].apply(lambda x: abs(x) if x < 120 else np.nan)
reporte['Edad_transformadas'] = (edad_antes != df['Edad']).sum()
reporte['Edad_invalidas'] = df['Edad'].isna().sum()
```

Se define una variable, esta comparará la diferencia antes y después, en la segunda línea, mediante la función limpiar_edad todo texto pasa a ser transformado a número si aplica, en la tercera línea, todo número negativo pasa a ser a positivo, y si este es superior a 120, entonces se limpiara el dato, las dos ultimas líneas, corresponde a contadores que serán utilizados más adelante para datos estadísticos.

2.4.2.3 Estandarizar fechas

```
fechas_antes = df['Fecha_Consulta'].copy()
df['Fecha_Consulta'] = df['Fecha_Consulta'].astype(str).str.replace(r"[./]", "-", regex=True)
df['Fecha_Consulta'] = df['Fecha_Consulta'].apply(procesarFecha)
reporte['Fechas_invalidas'] = (fechas_antes != df['Fecha_Consulta']).sum()
reporte['Fechas_nulas'] = df['Fecha_Consulta'].isna().sum()
```

Se define una variable, en la cual almacenara todo lo que se tiene almacenado, para luego comparar, En la segunda línea, todo valor que tenga un “.” se transforma en “-”, mediante la función procesar Fecha, se formatea todas las fechas, con el fin de que todos queden con él la estructura adecuada.

RESULTADOS

```
antes
0    2023-01-15
1    15/02/2023
2    2023.03.10
3    2023-04-22
4    23/05/2023
5    2023-01-15
6    10-06-2023
7    2023/07/18
8    2023-08-30
9    2023-09-05
Name: Fecha_Consulta, dtype: object
despues
0    2023-01-15
1    2023-02-15
2    2023-03-10
3    2023-04-22
4    2023-05-23
5    2023-01-15
6    2023-06-10
7    2023-07-18
8    2023-08-30
9    2023-09-05
Name: Fecha_Consulta, dtype: datetime64[ns]
```

2.4.2.4 Columnas numéricas (Peso - altura)

```
df['Peso_kg'] = pd.to_numeric(df['Peso_kg'], errors='coerce')
df['Altura_cm'] = pd.to_numeric(df['Altura_cm'], errors='coerce')
reporte['Peso_nulo'] = df['Peso_kg'].isna().sum()
reporte['Altura_nula'] = df['Altura_cm'].isna().sum()
```

Los datos que se registren en la columna peso_kg y altura_cm serán tratados como valores numéricos, aunque visualmente se vea las mismas condiciones, estas se encuentran definidas correctamente.

RESULTADO

antes											
ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico	
0	P001	Juan Pérez	35	M	2023-01-15	70.5	172	120/80	+1 555-123-4567	juan@email.com	Hipertensión
1	P002	María González	28	F	15/02/2023	58.2	165	110/70	5552345678	maria@email.com	Diabetes
2	P003	NaN	42	M	2023-03-10	81.7	178	130/85	(555) 345-6789	carlos@email.com	Asma
3	P004	Carlos Ruiz	treinta	M	2023-04-22	75.0	170	140/90	555.456.7890	cr@email.com	Hipertensión
4	P005	Ana López	29	F	23/05/2023	62.3	160	115/75	555-567-8901	ana@email.com	Migraña
6	P006	Luisa Fernández	50	F	10-06-2023	68.9	168	125/82	5556789012	luisa@email.com	Diabetes
7	P007	Pedro Martínez	33	M	2023/07/18	85.1	182	135/88	NaN	pedro@email.com	Artritis
8	P008	Sofía Vargas	-5	F	2023-08-30	59.8	163	118/78	5557890123	sofia@email.com	Depresión
9	P009	Jorge Silva	41	X	2023-09-05	72.4	175	155/102	5558901234	jorge@email.com	Hipertensión
despues											
ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico	
0	P001	Juan Pérez	35	M	2023-01-15	70.5	172	120/80	+1 555-123-4567	juan@email.com	Hipertensión
1	P002	María González	28	F	15/02/2023	58.2	165	110/70	5552345678	maria@email.com	Diabetes
2	P003	NaN	42	M	2023-03-10	81.7	178	130/85	(555) 345-6789	carlos@email.com	Asma
3	P004	Carlos Ruiz	treinta	M	2023-04-22	75.0	170	140/90	555.456.7890	cr@email.com	Hipertensión
4	P005	Ana López	29	F	23/05/2023	62.3	160	115/75	555-567-8901	ana@email.com	Migraña
6	P006	Luisa Fernández	50	F	10-06-2023	68.9	168	125/82	5556789012	luisa@email.com	Diabetes
7	P007	Pedro Martínez	33	M	2023/07/18	85.1	182	135/88	NaN	pedro@email.com	Artritis
8	P008	Sofía Vargas	-5	F	2023-08-30	59.8	163	118/78	5557890123	sofia@email.com	Depresión
9	P009	Jorge Silva	41	X	2023-09-05	72.4	175	155/102	5558901234	jorge@email.com	Hipertensión

2.4.2.5 Limpieza Género

```

detector = gender.Detector(case_sensitive=False)
df['Género'] = df['Género'].str.upper().str.strip()
df['Género'] = df.apply(lambda row: prediccionGenero(row, detector), axis=1)
reporte['Genero_nulo'] = df['Género'].isna().sum()
  
```

Se inicializa una función, en la segunda línea de código se pasa todos los valores a string, en la tercera línea, mediante la función definida, verificara el genero que le corresponde, dependiendo del nombre que tenga, en caso de que el valor almacenado sea vacío o diferente a F o M, en la cuarta línea, se realiza un conteo de todos los valores vacíos.

RESULTADOS

```

antes
      Nombre Género
0   Juan Pérez    M
1  María González  F
2         NaN     M
3   Carlos Ruiz    M
4   Ana López     F
6  Luisa Fernández F
7  Pedro Martínez M
8   Sofía Vargas  F
9   Jorge Silva   X
despues
      Nombre Género
0   Juan Pérez    M
1  María González  F
2         NaN     M
3   Carlos Ruiz    M
4   Ana López     F
6  Luisa Fernández F
7  Pedro Martínez M
8   Sofía Vargas  F
9   Jorge Silva   M
  
```


2.4.2.6 Estandarizar Teléfonos

```
df['Teléfono'] = df['Teléfono'].apply(limpiar_telefono)  
reporte['Telefono_nulo'] = df['Teléfono'].isna().sum()
```

Mediante la función limpiar_telefono, se limpian los datos eliminando texto, caracteres y verificando si es que tienen códigos de identificación del país, para ser eliminados, la segunda línea de código contará todos los datos que estén vacíos.

RESULTADOS

```
antes  
0    +1 555-123-4567  
1      5552345678  
2    (555) 345-6789  
3      555.456.7890  
4      555-567-8901  
6      5556789012  
7           NaN  
8      5557890123  
9      5558901234  
Name: Teléfono, dtype: object  
despues  
0      5551234567  
1      5552345678  
2      5553456789  
3      5554567890  
4      5555678901  
6      5556789012  
7           NaN  
8      5557890123  
9      5558901234  
Name: Teléfono, dtype: object
```


2.4.2.7 Manejar nombres faltantes

```
nombres_antes = df['Nombre'].isna().sum()
df['Nombre'] = df['Nombre'].fillna('Paciente_' + df['ID_Paciente'].astype(str))
reporte['Nombre_autogenerado'] = nombres_antes
```

Hace el conteo de todos los nombres que su valor son vacíos, en la siguiente línea, todos los valores vacíos, se autocompletan con paciente_ y el id que este tiene, en la tercera línea declara la cantidad de datos afectados,

RESULTADOS

```
antes
0      Juan Pérez
1    María González
2           NaN
3     Carlos Ruiz
4     Ana López
6    Luisa Fernández
7    Pedro Martínez
8     Sofía Vargas
9     Jorge Silva
Name: Nombre, dtype: object
despues
0      Juan Pérez
1    María González
2    Paciente_P003
3     Carlos Ruiz
4     Ana López
6    Luisa Fernández
7    Pedro Martínez
8     Sofía Vargas
9     Jorge Silva
Name: Nombre, dtype: object
```

2.4.2.8 IMC

```
df['IMC'] = df['Peso_kg'] / (df['Altura_cm'] / 100) ** 2
reporte['IMC_nulo'] = df['IMC'].isna().sum()
```

Se crea una nueva columna llamada IMC, el cual almacenara el resultado de la operación que se encuentra al lado, teniendo en cuenta que la altura se pasa a metros y que este está al cuadrado.

RESULTADOS

```
antes
despues
0    23.830449
1    21.377410
2    25.785886
3    25.951557
4    24.335937
6    24.411848
7    25.691342
8    22.507433
9    23.640816
Name: IMC, dtype: float64
PS C:\Users\Jefer\Repositorios\Laboratorio-ETL>
```

2.4.2.9 Clasificación presión arterial

```
df['Clasificacion_Presion'] = df.apply(lambda x: clasificar_presion(x['Presion_Sistolica'], x['Presion_Diastolica']),axis=1)
reporte['Clasificacion_Presion_nulo'] = df['Clasificacion_Presion'].isna().sum()
```

En la primera línea se encuentra la asignación de una nueva columna, dependiendo del resultado que arroje la estructura de control, se mostrara la categoría a la que pertenece el paciente.

RESULTADO

```
antes
  Presion_Sistolica  Presion_Diastolica
0             120             80
1             110             70
2             130             85
3             140             90
4             115             75
6             125             82
7             135             88
8             118             78
9             155            102

despues
  Presion_Sistolica  Presion_Diastolica  Clasificacion_Presion
0             120             80  Hipertensión Grado 1
1             110             70             Normal
2             130             85  Hipertensión Grado 1
3             140             90  Hipertensión Grado 2
4             115             75             Normal
6             125             82  Hipertensión Grado 1
7             135             88  Hipertensión Grado 1
8             118             78             Normal
9             155            102  Hipertensión Grado 2
PS C:\Users\Jefer\Repositorios\Laboratorio-ETL>
```

2.4.2.10 Mostrar resumen general

```
mostrar_reporte(reporte)
print(df)
```

Todos los contadores anteriormente utilizados, se muestran en este reporte, el cual almacena todos los datos capturados en su momento, también se muestra por consola los datos tratados durante el proceso

Reporte de Transformaciones de Datos:											
											Cantidad
Edad_invalidas											0
Edad_transformadas											9
Fechas_invalidas											1
Fechas_nulas											0
Genero_nulo											0
Peso_nulo											0
Altura_nula											0
Telefono_nulo											1
Presion_nula											0
Nombre_autogenerado											1
IMC_nulo											0
Clasificacion_Presion_nulo											0
ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	...	Diagnóstico	Presion_Sistolica	Presion_Diastolica	IMC	Clasificacion_Presion	
0	P001	Juan Pérez	35	M	2023-01-15	...	Hipertensión	120	80	23.830449	Hipertensión Grado 1
1	P002	María González	28	F	2023-02-15	...	Diabetes	110	70	21.377410	Normal
2	P003	Paciente_P003	42	M	2023-03-10	...	Asma	130	85	25.785886	Hipertensión Grado 1
3	P004	Carlos Ruiz	30	M	2023-04-22	...	Hipertensión	140	90	25.951557	Hipertensión Grado 2
4	P005	Ana López	29	F	2023-05-23	...	Migraña	115	75	24.335937	Normal
6	P006	Luisa Fernández	50	F	2023-06-10	...	Diabetes	125	82	24.411848	Hipertensión Grado 1
7	P007	Pedro Martínez	33	M	2023-07-18	...	Artritis	135	88	25.691342	Hipertensión Grado 1
8	P008	Sofía Vargas	5	F	2023-08-30	...	Depresión	118	78	22.507433	Normal
9	P009	Jorge Silva	41	M	2023-09-05	...	Hipertensión	155	102	23.640816	Hipertensión Grado 2

2.4.2.11 Exportar resultados

```
df.to_csv('pacientes_limpio.csv', index=False, encoding='utf-8-sig')
exportar_reporte_csv(reporte)
```

Se exportan dos archivos Excel, en donde se encuentra todos los datos almacenados.

RESULTADO

	Cantidad
Edad_invalidas	0
Edad_transformadas	9
Fechas_invalidas	1
Fechas_nulas	0
Genero_nulo	0
Peso_nulo	0
Altura_nula	0
Telefono_nulo	1
Presion_nula	0
Nombre_autogenerado	1
IMC_nulo	0
Clasificacion_Presion_nulo	0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ID_Paciente	Nombre	Edad	Género	Fecha_Consulta	Peso_kg	Altura_cm	Presión_Arterial	Teléfono	Email	Diagnóstico	Presion_Sistolica	Presion_Diastolica	IMC	Clasificación_Presion
2	P001	Juan Pérez	35	M	15/01/2023	70.5	172	120/80	5551234567	juan@email.com	Hipertensión	120		23.830448891292594	Hipertensión Grado 1
3	P002	Maria González	28	F	15/02/2023	58.2	165	110/70	5552345678	maria@email.com	Diabetes	110		21.377410468319564	Normal
4	P003	Paciente P003	42	M	10/03/2023	81.7	178	130/85	5553456789	carlos@email.com	Asma	130		25.785885620502462	Hipertensión Grado 1
5	P004	Carlos Ruiz	30	M	22/04/2023	75	170	140/90	5554567890	cr@email.com	Hipertensión	140		25.95155709342561	Hipertensión Grado 2
6	P005	Ana López	29	F	23/05/2023	62.3	160	115/75	5555678901	ana@email.com	Migraña	115		24.335937499999993	Normal
7	P006	Luisa Fernández	50	F	10/06/2023	68.9	168	125/82	5556789012	luisa@email.com	Diabetes	125		24.411848072562364	Hipertensión Grado 1
8	P007	Pedro Martínez	33	M	18/07/2023	85.1	182	135/88		pedro@email.com	Artritis	135		25.691341625407556	Hipertensión Grado 1
9	P008	Sofía Vargas	5	F	30/08/2023	59.8	163	118/78	5557890123	sofia@email.com	Depresión	118		22.507433475102562	Normal
10	P009	Jorge Silva	41	M	05/09/2023	72.4	175	155/102	5558901234	jorge@email.com	Hipertensión	155		23.640816326530615	Hipertensión Grado 2

3. CONCLUSIONES

Si bien la biblioteca panda ofrece una amplia gama de herramientas eficientes para la manipulación y análisis de datos, en ciertos casos es necesario complementar su funcionalidad con herramientas externas especializadas. Estas herramientas permiten abordar problemas específicos con mayor precisión, como la detección de género, la conversión de texto a números o el tratamiento de valores atípicos. La integración de estas soluciones externas en el flujo ETL enriquece el proceso de transformación de datos, mejorando su precisión, calidad y versatilidad.

Adicionalmente, la implementación de funciones personalizadas dentro del proceso facilita la incorporación de estructuras de control que permiten tomar decisiones a nivel de cada registro. Esto significa que se puede analizar campo por campo, aplicar condiciones, validar formatos o corregir errores de forma dinámica, dependiendo del tipo de dato que se esté procesando. Estas funciones no solo automatizan tareas repetitivas, sino que también aportan flexibilidad al momento de enfrentar datos inconsistentes o con formatos variados.

Finalmente, una vez completado el proceso de transformación, es posible evidenciar un cambio significativo en el tratamiento de los datos. La información, que inicialmente se presentaba desordenada y difícil de interpretar, ahora adquiere una estructura clara, coherente y lista para su análisis. Este resultado no solo mejora la calidad de los datos, sino que también brinda mayor confianza y claridad en la toma de decisiones basadas en la información depurada.

En el siguiente enlace se encuentra el repositorio del Laboratorio ETL, el cual contiene todo el desarrollo del ejercicio realizado para dicha práctica.

<https://github.com/Jeferson-Arias/Laboratorio-ETL.git>