Limpieza básica de datos con Pandas

1. Importar la librería Pandas

In [3]:
```
import pandas as pd
```

2. Leer e imprimir el archivo CSV que queremos limpiar

In [5]:
```
df = pd.read_csv('C:Downloads/datos_ejemplo.csv') #Leer
print('DataFrame original:') #Leyenda
df #Imprimir
```

```
DataFrame original:
```

Out[5]:

	ID	Nombre	Edad	Genero	Salario	Departamento
0	1	Ana	25	F	50000.0	Ventas
1	2	Luis	30	M	NaN	Marketing
2	3	Carlos	Treinta y cinco	M	60000.0	Ventas
3	4	Sofia	40	F	55000.0	RRHH
4	5	NaN	45	M	70000.0	TI

Como podemos observar existen algunos errores en nuestro DataFrame como: Valores nulos y errores en los tipos de datos

3. Manejo de valores nulos

In [7]:
```
print('Valores nulos por columna:') # Leyenda
df.isnull().sum()  # Verificar valores nulos en nuestro DataFrame
```

```
Valores nulos por columna:
```

Out[7]:
```
ID              0
Nombre          1
Edad            0
Genero          0
Salario         1
Departamento    0
dtype: int64
```

La función isnull() nos permite indentificar valores nulos en cada serie, al usarla en conjunto con la función sum() podemos ver el total de nulos por serie en nuestro DataFrame

3.1. Eliminación de filas con valores nulos

En este punto podemos tratar nuestro DataFrame de varias formas, en esta primera opción eliminaremos todas las filas que contengan uno o varios valores nulos en alguna de sus filas con la función dropna()

In [9]:
```
df_sin_nulos = df.dropna() #Guardamos en una nueva variable nuestro DataFrame sin valores nul
print('DataFrame sin valores nulos:') #Leyenda
df_sin_nulos #Imprimir
```

```
DataFrame sin valores nulos:
```

Out[9]:

	ID	Nombre	Edad	Genero	Salario	Departamento
0	1	Ana	25	F	50000.0	Ventas
2	3	Carlos	Treinta y cinco	M	60000.0	Ventas
3	4	Sofia	40	F	55000.0	RRHH

3.2. Imputación de datos para valores nulos

Una alternativa a la elimicación de las filas con nulos es la imputación de datos, un método para conservar la mayoría de los datos y la información del conjunto de datos sustituyendo los datos faltantes por un valor diferente, esto se logra con la función fillna() a la cual le especificaremos con que datos rellenar cada serie del DataFrame

In [11]:
```
df_rellenado = df.fillna({'Salario': 0, 'Nombre': 'Desconocido'}) #Nueva variable con valores
print('DataFrame con valores rellenados:') #Leyenda
df_rellenado #Imprimir
```

```
DataFrame con valores rellenados:
```

Out[11]:

	ID	Nombre	Edad	Genero	Salario	Departamento
0	1	Ana	25	F	50000.0	Ventas
1	2	Luis	30	M	0.0	Marketing
2	3	Carlos	Treinta y cinco	M	60000.0	Ventas
3	4	Sofia	40	F	55000.0	RRHH
4	5	Desconocido	45	M	70000.0	TI

3.3. Resumen

Recapitulando hasta el momento tenemos 3 DataFrames:

- df = dataframe original si ningún cambio
- df_sin_nulos = dataframe sin filas con valores nulos
- df_rellenado = dataframe con valores nulos rellenados

In [13]:
```
print("\nDataFrame original:")
print(df)
print("\nDataFrame sin filas con valores nulos:")
print(df_sin_nulos)
print("\nDataFrame con valores nulos rellenados:")
print(df_rellenado)
```

```
DataFrame original:
   ID  Nombre             Edad Genero  Salario Departamento
0   1     Ana               25      F  50000.0       Ventas
1   2    Luis               30      M      NaN    Marketing
2   3  Carlos  Treinta y cinco      M  60000.0       Ventas
3   4   Sofia               40      F  55000.0         RRHH
4   5     NaN               45      M  70000.0           TI

DataFrame sin filas con valores nulos:
   ID  Nombre             Edad Genero  Salario Departamento
0   1     Ana               25      F  50000.0       Ventas
2   3  Carlos  Treinta y cinco      M  60000.0       Ventas
3   4   Sofia               40      F  55000.0         RRHH

DataFrame con valores nulos rellenados:
   ID       Nombre             Edad Genero  Salario Departamento
0   1          Ana               25      F  50000.0       Ventas
1   2         Luis               30      M      0.0    Marketing
2   3       Carlos  Treinta y cinco      M  60000.0       Ventas
3   4        Sofia               40      F  55000.0         RRHH
4   5  Desconocido               45      M  70000.0           TI
```

4. Detección de errores en los tipos de datos

Como podemos observar, en la columna Edad tenemos un dato como cadena de texto, esto quiere decir que esta columna probablemente no sea de tipo numérica ya que acepta este tipo de dato, lo primero será observar los tipos de datos de nuestras columnas

In [15]:
```
df_rellenado.dtypes #la función dtypes nos muestra los tipos de datos de nuestro DataFrame,
```

Out[15]:
```
ID              int64
Nombre         object
Edad           object
Genero         object
Salario       float64
Departamento    object
dtype: object
```

4.1. Corrección de errores en los tipos de datos

- Edad es de tipo object, por lo que pasaremos esta columna a datos numérico
- La función to_numeric() convierte los valores de una columna a tipo numérico. Con el parámetro errors='coerce' si encuentra valores que no pueden convertirse a números los convierte en NaN

In [17]:
```
df_rellenado['Edad'] = pd.to_numeric(df_rellenado['Edad'], errors='coerce') #Pasar columna Ed
print(df_rellenado.dtypes) #Podemos ver que Edad ahora es de tipo float
df_rellenado
```

```
ID              int64
Nombre         object
Edad          float64
Genero         object
Salario       float64
Departamento    object
dtype: object
```

Out[17]:

	ID	Nombre	Edad	Genero	Salario	Departamento
0	1	Ana	25.0	F	50000.0	Ventas
1	2	Luis	30.0	M	0.0	Marketing
2	3	Carlos	NaN	M	60000.0	Ventas
3	4	Sofia	40.0	F	55000.0	RRHH
4	5	Desconocido	45.0	M	70000.0	TI

4.2. Corrección de errores en los valores NaN

Como en casos anterioires tenemos varias opciones, eliminar las filas con valores nules con dropna() o rellenar los valores con fillna(), en el anterior paso imputamos valores fijos, en esta ocación imputaremos los datos con una función estadística como es el promedio, esto quiere decir que estaremos rellenando los valores nulos con un promedio de los demás valores, esto lo logramos con la función mean()

In [19]:
```
df_rellenado['Edad'] = df_rellenado['Edad'].fillna(df_rellenado['Edad'].mean())
print('\nDataFrame Final:')
df_rellenado
```

```
DataFrame Final:
```

Out[19]:

	ID	Nombre	Edad	Genero	Salario	Departamento
0	1	Ana	25.0	F	50000.0	Ventas
1	2	Luis	30.0	M	0.0	Marketing
2	3	Carlos	35.0	M	60000.0	Ventas
3	4	Sofia	40.0	F	55000.0	RRHH
4	5	Desconocido	45.0	M	70000.0	TI

5. Exportación

Finalmente podemos exportar nuestro DataFrame limpio a un archivo CV en la ruta que elijamos, el parámetro index=false nos permite no importar los índices para que sea más facil la lectura de nuestro archivo en otros programas como PowerBI o Tableau

In [ ]:
```
df_rellenado.to_csv('C:Downloads/datos_ejemplo_limpios', index=False)
```

Conclusión

Como pudimos ver existen varias funciones y parámetros configurables que nos pueden ayudar a limpiar nuestro DataFrame, es importante tomar en consideración previo a la limpieza, definir que tratamiento se le dará a los valores nulos y que tipo de datos tenemos en nuestras columnas

Hecho por Sebastián Viera