

2. Análisis Inicial de la Base de Datos

Antes de comenzar a limpiar la base de datos, debe realizar un análisis preliminar para comprender la naturaleza y distribución de los errores. Para ello, siga los siguientes pasos:

- Mostrar un resumen estadístico de los datos.

```
resumen_estadístico = df.describe(include='all')
print("Resumen estadístico:\n", resumen_estadístico)
```

Resumen estadístico:

	country	location_name	latitude	longitude	timezone	\
count	33099	33115	33083	33076	33116	
unique	211	248	235	240	191	
top	Bulgaria	aaaaa	40.4	aaaaa	Asia/Bangkok	
freq	369	326	343	382	604	
mean	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	

	last_updated_epoch	last_updated	temperature_celsius	\
count	33045	33080	33117	
unique	183	3776	476	
top	aaaaa	aaaaa	28.0	
freq	352	322	592	
mean	NaN	NaN	NaN	
std	NaN	NaN	NaN	
min	NaN	NaN	NaN	
25%	NaN	NaN	NaN	
50%	NaN	NaN	NaN	
75%	NaN	NaN	NaN	
max	NaN	NaN	NaN	

	temperature_fahrenheit	condition_text	...	air_quality_PM2.5	\
count	33084	33084	...	33075	
unique	794	30	...	1818	
top	84.2	Partly cloudy	...	0.5	
freq	535	11302	...	1277	
mean	NaN	NaN	...	NaN	
std	NaN	NaN	...	NaN	
min	NaN	NaN	...	NaN	
25%	NaN	NaN	...	NaN	
50%	NaN	NaN	...	NaN	
75%	NaN	NaN	...	NaN	
max	NaN	NaN	...	NaN	

Resumen estadístico:

- Se incluyen descripciones básicas como el número de valores, valores únicos, y el valor más frecuente (moda) para cada columna.
- La columna “country” tiene 211 valores únicos, mientras que “location_name” tiene 248.
- Algunas columnas como “latitude”, “longitude”, y “last_updated_epoch” contienen valores poco consistentes, como “aaaaa”, lo que sugiere errores en los datos.

	air_quality_PM10	air_quality_us-epa-index	air_quality_gb-defra-index	\
count	33100	33078	33093	
unique	2717	7	11	
top	0.6	1.0	1.0	
freq	360	24084	21874	
mean	NaN	NaN	NaN	
std	NaN	NaN	NaN	
min	NaN	NaN	NaN	
25%	NaN	NaN	NaN	
50%	NaN	NaN	NaN	
75%	NaN	NaN	NaN	
max	NaN	NaN	NaN	

	sunrise	sunset	moonrise	moonset	moon_phase	moon_illumination
count	33085	33104	0.0	33105	33071	33094
unique	309	408	NaN	1442	9	78
top	05:56 AM	aaaaa	NaN	No moonset	Waxing Gibbous	0.0
freq	428	340	NaN	1005	8137	1438
mean	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

[11 rows x 41 columns]

- Calcular el porcentaje de valores faltantes por columna.

```
# Calcular el porcentaje de valores faltantes por columna
porcentaje_faltantes = df.isnull().mean() * 100
print("\nPorcentaje de valores faltantes por columna:\n", porcentaje_faltantes)
```

```
Porcentaje de valores faltantes por columna:
country                0.951611
location_name          0.903732
latitude               0.999491
longitude              1.020439
timezone               0.900739
last_updated_epoch     1.113206
last_updated           1.008469
temperature_celsius    0.897747
temperature_fahrenheit 0.996499
condition_text         0.996499
wind_mph               0.975551
wind_kph               0.912709
wind_degree            0.921687
wind_direction         0.936649
pressure_mb            0.981536
pressure_in            0.948619
precip_mm              0.972559
precip_in              0.972559
humidity               0.942634
cloud                 1.077296
feels_like_celsius     0.894754
feels_like_fahrenheit  0.924679
visibility_km           1.050364
visibility_miles        0.930664
uv_index               0.960589
gust_mph               0.945626
gust_kph               1.068319
air_quality_Carbon_Monoxide 1.035401
air_quality_Ozone       0.948619
air_quality_Nitrogen_dioxide 1.128168
air_quality_Sulphur_dioxide 1.017446
air_quality_PM2.5       1.023431
air_quality_PM10        0.948619
air_quality_us-epa-index 1.014454
air_quality_gb-defra-index 0.969566
sunrise                0.993506
sunset                 0.936649
moonrise                100.000000
moonset                0.933657
moon_phase              1.035401
moon_illumination       0.966574
dtype: float64
```

Porcentaje de valores faltantes por columna:

- Hay algunas columnas con un porcentaje considerable de valores faltantes.

Por ejemplo:

- moonrise: 100% valores faltantes.
- sunrise: 99.35% valores faltantes.
- moon_phase : 1.03% valores faltantes.

- Identificar si hay filas duplicadas.

```
# Identificar filas duplicadas
duplicados = df.duplicated().sum()
print(f"\nNúmero de filas duplicadas: {duplicados}")
```

```
Número de filas duplicadas: 1504
```

- **Filas duplicadas:**

Se han identificado 1,504 filas duplicadas en el conjunto de datos.

- Analizar los tipos de datos de las columnas y si son consistentes con el contenido esperado.

```
# Tipos de datos de las columnas
tipos_datos = df.dtypes
print("\nTipos de datos de las columnas:\n", tipos_datos)
```

Tipos de datos de las columnas:

country	object
location_name	object
latitude	object
longitude	object
timezone	object
last_updated_epoch	object
last_updated	object
temperature_celsius	object
temperature_fahrenheit	object
condition_text	object
wind_mph	object
wind_kph	object
wind_degree	object
wind_direction	object
pressure_mb	object
pressure_in	object
precip_mm	object
precip_in	object
humidity	object
cloud	object
feels_like_celsius	object
feels_like_fahrenheit	object
visibility_km	object
visibility_miles	object
uv_index	object
gust_mph	object
gust_kph	object
air_quality_Carbon_Monoxide	object
air_quality_Ozone	object
air_quality_Nitrogen_dioxide	object
air_quality_Sulphur_dioxide	object
air_quality_PM2.5	object
air_quality_PM10	object
air_quality_us-epa-index	object
air_quality_gb-defra-index	object
sunrise	object
sunset	object
moonrise	float64
moonset	object
moon_phase	object
moon_illumination	object
dtype:	object

Tipos de datos por columna:

Muchas columnas que deberían ser numéricas están registradas como texto (object).

Por ejemplo, columnas como “latitude”, “longitude”, “temperature_celsius”, y “humidity” están categorizadas como texto en lugar de números, lo cual indica inconsistencias en los tipos de datos.

3. Limpieza de Datos

Debe realizar las siguientes tareas de limpieza en la base de datos:

- Eliminación o imputación de valores faltantes: Justificar si decide eliminar filas/columnas con NaNs o utilizar técnicas de imputación (relleno de valores).

```
[18] # 1. Eliminación o Imputación de Valores Faltantes

# Revisar porcentaje de valores faltantes
porcentaje_faltantes = df.isnull().mean() * 100

# Decidir si eliminar o imputar en base a los valores faltantes
threshold = 50 # Si más del 50% de los valores faltan, eliminamos la columna
columnas_a_eliminar = porcentaje_faltantes[porcentaje_faltantes > threshold].index
df_limpio = df.drop(columns=columnas_a_eliminar)

# Para las columnas restantes, imputamos usando la mediana (opcional)
df_limpio.fillna(df_limpio.median(numeric_only=True), inplace=True)
df_limpio
```

• **Eliminación de columnas con muchos NaNs:** Si más del 50% de los datos en una columna están ausentes, es mejor eliminar la columna, ya que su presencia no aporta información suficiente.

• **Imputación de valores faltantes:** Para las columnas clave con pocos valores faltantes, imputamos usando la mediana. También se puede usar la media o un valor específico si es más adecuado.

- Eliminación de duplicados: Identificar filas duplicadas y eliminarlas.

```
# 2. Eliminación de Duplicados
df_limpio = df_limpio.drop_duplicates()
df_limpio.head()
```

- Corrección de tipos de datos: Asegurarse de que las columnas tengan tipos de datos adecuados (por ejemplo, números como int o float, fechas como datetime, textos como str).

```
# 3. Corrección de Tipos de Datos
# Listado de columnas que deberían ser numéricas y están como texto
columnas_a_convertir = ['latitude', 'longitude', 'temperature_celsius', 'temperature_fahrenheit',
                        'wind_mph', 'wind_kph', 'humidity', 'pressure_mb', 'precip_mm',
                        'feels_like_celsius', 'feels_like_fahrenheit', 'visibility_km', 'uv_index']

# Convertir a float
for columna in columnas_a_convertir:
    df_limpio[columna] = pd.to_numeric(df_limpio[columna], errors='coerce')

# Convertir las fechas
df_limpio['last_updated'] = pd.to_datetime(df_limpio['last_updated'], errors='coerce')
df_limpio.head()
```

- Corrección de valores "invalid": Corregir los valores que fueron etiquetados con cadenas incorrectas como 'bbb'.

```
# 4. Corrección de Valores "Invalid"
# Identificar y corregir valores como 'aaa', 'bbb', etc.
valores_invalidos = ['aaa', 'aaaa', 'aaaaa', 'bbb', 'cccc', 'xxxxx'] # Añadir más si es necesario
df_limpio.replace(valores_invalidos, pd.NA, inplace=True)
df_limpio.head()
```

4. Documentación y Reporte

Luego de limpiar los datos, redacte un reporte en el cual documente los siguientes aspectos:

- Análisis inicial:

- Resumen estadístico de la base de datos antes de la limpieza.

```
# Resumen estadístico antes de la limpieza
resumen_estadístico = df.describe(include='all')
print("Resumen estadístico antes de la limpieza:\n", resumen_estadístico)
```

```
resumen_estadístico = df.describe(include='all')
print("Resumen estadístico:\n", resumen_estadístico)
```

Resumen estadístico:

	country	location_name	latitude	longitude	timezone
count	33099	33115	33083	33076	33116
unique	211	248	235	240	191
top	Bulgaria	aaaaa	40.4	aaaaa	Asia/Bangkok
freq	369	326	343	382	604
mean	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN

	last_updated_epoch	last_updated	temperature_celsius
count	33045	33080	33117
unique	183	3776	476
top	aaaaa	aaaaa	28.0
freq	352	322	592
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

	temperature_fahrenheit	condition_text	...	air_quality_PM2.5
count	33084	33084	...	33075
unique	794	30	...	1818
top	84.2	Partly cloudy	...	0.5
freq	535	11302	...	1277
mean	NaN	NaN	...	NaN
std	NaN	NaN	...	NaN
min	NaN	NaN	...	NaN
25%	NaN	NaN	...	NaN
50%	NaN	NaN	...	NaN
75%	NaN	NaN	...	NaN
max	NaN	NaN	...	NaN

	air_quality_PM10	air_quality_us-epa-index	air_quality_gb-defra-index
count	33100	33078	33093
unique	2717	7	11
top	0.6	1.0	1.0
freq	360	24084	21874
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

	sunrise	sunset	moonrise	moonset	moon_phase	moon_illumination
count	33085	33104	0.0	33105	33071	33094
unique	309	408	NaN	1442	9	78
top	05:56 AM	aaaaa	NaN	No moonset	Waxing Gibbous	0.0
freq	428	340	NaN	1005	8137	1438
mean	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

[11 rows x 41 columns]

- Tabla que muestre el porcentaje de valores faltantes por columna.

```
# Porcentaje de valores faltantes por columna antes de la limpieza
porcentaje_faltantes = df.isnull().mean() * 100
print("\nPorcentaje de valores faltantes por columna antes de la limpieza:\n", porcentaje_faltantes)
```

```
Porcentaje de valores faltantes por columna antes de la limpieza:
country                0.951611
location_name          0.903732
latitude               0.999491
longitude              1.020439
timezone               0.900739
last_updated_epoch     1.113206
last_updated           1.008469
temperature_celsius    0.897747
temperature_fahrenheit 0.996499
condition_text         0.996499
wind_mph               0.975551
wind_kph               0.912709
wind_degree            0.921687
wind_direction         0.936649
pressure_mb            0.981536
pressure_in            0.948619
precip_mm              0.972559
precip_in              0.972559
humidity               0.942634
cloud                  1.077296
feels_like_celsius     0.894754
feels_like_fahrenheit  0.924679
visibility_km          1.050364
visibility_miles       0.930664
uv_index               0.960589
gust_mph               0.945626
gust_kph               1.068319
air_quality_Carbon_Monoxide 1.035401
air_quality_Ozone      0.948619
air_quality_Nitrogen_dioxide 1.128168
air_quality_Sulphur_dioxide 1.017446
air_quality_PM2.5      1.023431
air_quality_PM10       0.948619
air_quality_us-epa-index 1.014454
air_quality_gb-defra-index 0.969566
sunrise                0.993506
sunset                 0.936649
moonrise               100.000000
moonset                0.933657
moon_phase             1.035401
moon_illumination      0.966574
dtype: float64
```

- Total de filas duplicadas encontradas.

```
[35] # Número de filas duplicadas antes de la limpieza
duplicados = df.duplicated().sum()
print(f"\nNúmero de filas duplicadas antes de la limpieza: {duplicados}")
```

```
Número de filas duplicadas antes de la limpieza: 1504
```

- Descripción de los tipos de datos originales y los problemas encontrados.

```
# Tipos de datos antes de la limpieza
tipos_datos = df.dtypes
print("\nTipos de datos antes de la limpieza:\n", tipos_datos)
```

```
Tipos de datos antes de la limpieza:
country                object
location_name          object
latitude               object
longitude              object
timezone               object
last_updated_epoch     object
last_updated           object
temperature_celsius    object
temperature_fahrenheit object
condition_text         object
wind_mph               object
wind_kph               object
wind_degree            object
wind_direction         object
pressure_mb            object
pressure_in            object
precip_mm              object
precip_in              object
humidity               object
cloud                  object
```

```
feels_like_celsius     object
feels_like_fahrenheit  object
visibility_km          object
visibility_miles       object
uv_index               object
gust_mph               object
gust_kph               object
air_quality_Carbon_Monoxide object
air_quality_Ozone      object
air_quality_Nitrogen_dioxide object
air_quality_Sulphur_dioxide object
air_quality_PM2.5      object
air_quality_PM10       object
air_quality_us-epa-index object
air_quality_gb-defra-index object
sunrise                object
sunset                 object
moonrise               float64
moonset                object
moon_phase             object
moon_illumination      object
dtype: object
```

Tipos de datos por columna:

Muchas columnas que deberían ser numéricas están registradas como texto (object).

Por ejemplo, columnas como “latitude”, “longitude”, “temperature_celsius”, y “humidity” están categorizadas como texto en lugar de números, lo cual indica inconsistencias en los tipos de datos.

- Proceso de limpieza:

▪ Describir qué métodos utilizaron para limpiar la base de datos (eliminación, imputación, etc.):

- **Eliminación de columnas** con más del 50% de valores faltantes.
- **Imputación de valores faltantes** en las columnas clave usando la mediana.
- **Eliminación de duplicados** para garantizar datos únicos.
- **Corrección de valores inválidos** y tipos de datos.

▪ Mostrar antes y después de cada paso clave (por ejemplo, antes y después de eliminar duplicados).

ANTES:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	last_updated	temperature_celsius	temperature_fahrenheit	condition_text	...	air_quality_PM2.5	air_quality_PM10	air_quality_us- epa-index	
0	Afghanistan	Kabul	NaN	69.18	Asia/Kabul	1715849100.0	2024-05-16 13:15	26.6	79.8	Partly Cloudy	...	8.4	26.6	1.0	
1	Albania	Tirana	41.33	19.82	Europe/Tirane	1715849100.0	2024-05-16 10:45	19.0	66.2	Partly cloudy	...	1.1	2.0	1.0	
2	Algeria	Algiers	36.76	aaaaa	Africa/Algiers	1715849100.0	2024-05-16 09:45	23.0	73.4	Sunny	...	10.4	18.4	1.0	
3	Andorra	Andorra La Vella	42.5	1.52	Europe/Andorra	1715849100.0	2024-05-16 10:45	6.3	43.3	Light drizzle	...	0.7	0.9	1.0	
4	Angola	Luanda	-8.84	13.23	Africa/Luanda	1715849100.0	2024-05-16 09:45	26.0	78.8	Partly cloudy	...	183.4	262.3	5.0	

5 rows x 41 columns

DESPUES de Eliminación o imputación de valores faltantes:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	last_updated	temperature_celsius	temperature_fahrenheit	condition_text	...	air_quality_Sulphur_dioxide	air_quality_PM2.5	air_quality	
0	Afghanistan	Kabul	NaN	69.18	Asia/Kabul	1715849100.0	2024-05-16 13:15	26.6	79.8	Partly Cloudy	...	0.2	8.4		
1	Albania	Tirana	41.33	19.82	Europe/Tirane	1715849100.0	2024-05-16 10:45	19.0	66.2	Partly cloudy	...	0.1	1.1		
2	Algeria	Algiers	36.76	aaaaa	Africa/Algiers	1715849100.0	2024-05-16 09:45	23.0	73.4	Sunny	...	13.4	10.4		
3	Andorra	Andorra La Vella	42.5	1.52	Europe/Andorra	1715849100.0	2024-05-16 10:45	6.3	43.3	Light drizzle	...	0.2	0.7		
4	Angola	Luanda	-8.84	13.23	Africa/Luanda	1715849100.0	2024-05-16 09:45	26.0	78.8	Partly cloudy	...	31.5	183.4		

DESPUES de Eliminación de duplicados:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	last_updated	temperature_celsius	temperature_fahrenheit	condition_text	...	air_quality_Sulphur_dioxide	air_quality_PM2.5	air_quality	
0	Afghanistan	Kabul	NaN	69.18	Asia/Kabul	1715849100.0	2024-05-16 13:15	26.6	79.8	Partly Cloudy	...	0.2	8.4		
1	Albania	Tirana	41.33	19.82	Europe/Tirane	1715849100.0	2024-05-16 10:45	19.0	66.2	Partly cloudy	...	0.1	1.1		
2	Algeria	Algiers	36.76	aaaaa	Africa/Algiers	1715849100.0	2024-05-16 09:45	23.0	73.4	Sunny	...	13.4	10.4		
3	Andorra	Andorra La Vella	42.5	1.52	Europe/Andorra	1715849100.0	2024-05-16 10:45	6.3	43.3	Light drizzle	...	0.2	0.7		
4	Angola	Luanda	-8.84	13.23	Africa/Luanda	1715849100.0	2024-05-16 09:45	26.0	78.8	Partly cloudy	...	31.5	183.4		

DESPUES de Corrección de Tipos de Datos:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	last_updated	temperature_celsius	temperature_fahrenheit	condition_text	...	air_quality_Sulphur_dioxide	air_quality_PM2.5	air_quality	
0	Afghanistan	Kabul	NaN	69.18	Asia/Kabul	1715849100.0	2024-05-16 13:15:00	26.6	79.8	Partly Cloudy	...	0.2	8.4		
1	Albania	Tirana	41.33	19.82	Europe/Tirane	1715849100.0	2024-05-16 10:45:00	19.0	66.2	Partly cloudy	...	0.1	1.1		
2	Algeria	Algiers	36.76	NaN	Africa/Algiers	1715849100.0	2024-05-16 09:45:00	23.0	73.4	Sunny	...	13.4	10.4		
3	Andorra	Andorra La Vella	42.50	1.52	Europe/Andorra	1715849100.0	2024-05-16 10:45:00	6.3	43.3	Light drizzle	...	0.2	0.7		
4	Angola	Luanda	-8.84	13.23	Africa/Luanda	1715849100.0	2024-05-16 09:45:00	26.0	78.8	Partly cloudy	...	31.5	183.4		

5 rows x 40 columns

DESPUES de Correccion de Valores Invalid:

	temperature_celsius	temperature_fahrenheit	condition_text	...	air_quality_Sulphur_dioxide	air_quality_PM2.5	air_quality_PM10	air_quality_us- epa-index	air_quality_gb- defra-index	sunrise	sunset	moonset	moon_phase	moon_illumination	
	26.6	79.8	Partly Cloudy	...	0.2	8.4	26.6	1.0	1.0	04:50 AM	06:50 PM	01:11 AM	Waxing Gibbous	55.0	
	19.0	66.2	Partly cloudy	...	0.1	1.1	2.0	1.0	1.0	05:21 AM	07:54 PM	02:14 AM	Waxing Gibbous	55.0	
	23.0	73.4	Sunny	...	13.4	10.4	18.4	1.0	1.0	05:40 AM	07:50 PM	02:14 AM	Waxing Gibbous	55.0	
	6.3	43.3	Light drizzle	...	0.2	0.7	0.9	1.0	<NA>	06:31 AM	09:11 PM	03:31 AM	Waxing Gibbous	55.0	
	26.0	78.8	Partly cloudy	...	31.5	183.4	262.3	5.0	10.0	06:12 AM	05:55 PM	12:38 AM	Waxing Gibbous	55.0	

- Resultados:

- Resumen final de la base de datos después de la limpieza.

```
# Resumen estadístico después de la limpieza
resumen_final = df_limpio.describe(include='all')
print("Resumen estadístico después de la limpieza:\n", resumen_final)
```

Resumen estadístico después de la limpieza:

	country	location_name	latitude	longitude	timezone
count	32732	32789	32755	32694	32786
unique	210	247	234	239	190
top	Bulgaria	Muscat	40.4	14.51	Asia/Bangkok
freq	369	186	343	336	604

	last_updated_epoch	last_updated	temperature_celsius
count	32693	32758	32733
unique	182	3775	475
top	1721133900.0	2024-07-16 14:45	28.0
freq	263	63	592

	temperature_fahrenheit	condition_text	...	air_quality_Sulphur_dioxide
count	32764	32718	...	32753
unique	793	29	...	838
top	84.2	Partly cloudy	...	0.1
freq	535	11302	...	2451

	air_quality_PM2.5	air_quality_PM10	air_quality_us-epa-index
count	32726	32768	32729
unique	1817	2716	6
top	0.5	0.6	1.0
freq	1277	360	24084

	air_quality_gb-defra-index	sunrise	sunset	moonset
count	32759	32763	32764	32782
unique	10	308	407	1441
top	1.0	05:56 AM	06:28 PM	No moonset
freq	21874	428	338	1005

	moon_phase	moon_illumination
count	32726	32707
unique	8	77
top	Waxing Gibbous	0.0
freq	8137	1438

[4 rows x 40 columns]

- Confirmar que los tipos de datos son correctos.

```
# Tipos de datos después de la limpieza
tipos_datos_despues = df_limpio.dtypes
print("\nTipos de datos después de la limpieza:\n", tipos_datos_despues)
```

Tipos de datos después de la limpieza:

Variable	tipo de dato
country	object
location_name	object
latitude	float64
longitude	float64
timezone	object
last_updated_epoch	object
last_updated	datetime64[ns]
temperature_celsius	float64
temperature_fahrenheit	float64
condition_text	object
wind_mph	float64
wind_kph	float64
wind_degree	object
wind_direction	object
pressure_mb	float64
pressure_in	object
precip_mm	float64
precip_in	object
humidity	float64
cloud	object
feels_like_celsius	float64
feels_like_fahrenheit	float64
visibility_km	float64
visibility_miles	object
uv_index	float64
gust_mph	object
gust_kph	object
air_quality_Carbon_Monoxide	object
air_quality_Ozone	object
air_quality_Nitrogen_dioxide	object
air_quality_Sulphur_dioxide	object
air_quality_PM2.5	object
air_quality_PM10	object
air_quality_us-epa-index	object
air_quality_gb-defra-index	object
sunrise	object
sunset	object
moonset	object
moon_phase	object
moon_illumination	object
dtype:	object

- Tabla que muestre el porcentaje de valores faltantes final por columna.

```
# Porcentaje de valores faltantes por columna después de la limpieza
porcentaje_faltantes_despues = df_limpio.isnull().mean() * 100
print("\nPorcentaje de valores faltantes por columna después de la limpieza:\n", porcentaje_faltantes_despues)
```

Porcentaje de valores faltantes por columna después de la limpieza:

country	2.049855
location_name	1.879283
latitude	1.981028
longitude	2.163569
timezone	1.888260
last_updated_epoch	2.166562
last_updated	1.972050
temperature_celsius	2.046862
temperature_fahrenheit	1.954095
condition_text	2.091750
wind_mph	1.987013
wind_kph	1.879283
wind_degree	2.055840
wind_direction	2.010953
pressure_mb	2.004968
pressure_in	2.046862
precip_mm	1.939133
precip_in	1.987013
humidity	1.957088
cloud	2.199479
feels_like_celsius	1.873298
feels_like_fahrenheit	2.010953
visibility_km	2.061825
visibility_miles	1.969058
uv_index	1.927163
gust_mph	1.954095
gust_kph	2.151599
air_quality_Carbon_Monoxide	2.040877
air_quality_Ozone	2.088757
air_quality_Nitrogen_dioxide	2.220427
air_quality_Sulphur_dioxide	1.987013
air_quality_PM2.5	2.067810
air_quality_PM10	1.942125
air_quality_us-epa-index	2.058832
air_quality_gb-defra-index	1.969058
sunrise	1.957088
sunset	1.954095
moonset	1.900230
moon_phase	2.067810
moon_illumination	2.124667
dtype: float64	

- Comprobación de que no hay duplicados ni valores inválidos

```
[68] # Comprobar si aún existen valores inválidos
valores_invalidos = ['aaa', 'aaa', 'bbb', 'cccc', 'xxxxx'] # Añadir más si es necesario
df_invalidos = df_limpio[df_limpio.isin(valores_invalidos).any(axis=1)]
print(f"\nFilas con valores inválidos después de la limpieza: {df_invalidos.shape[0]}")
```

Filas con valores inválidos después de la limpieza: 0