002_2019_clean

May 20, 2025

[1]:
```
import pandas as pd
import numpy as np
import logging
import os

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler("002_2019_clean.log"), # Actualizado para 2019
        logging.StreamHandler()
    ]
)

logging.info("Inicio del notebook de limpieza y transformación para 2019.csv␣
 ↪(002_2019_clean.ipynb).")

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 100)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

```
2025-05-20 17:47:59,611 - INFO - Inicio del notebook de limpieza y
transformación para 2019.csv (002_2019_clean.ipynb).
```

[2]:
```
# --- Definición de la ruta al archivo ---
file_path_2019_raw = "/home/nicolas/Escritorio/workshops ETL/workshop_3/data/
 ↪raw/2019.csv" # Actualizado
logging.info(f"Ruta del archivo raw a cargar: {file_path_2019_raw}")

df_2019_raw = None
df_2019_clean = None

try:
    logging.info(f"Intentando cargar el archivo CSV: {os.path.
 ↪basename(file_path_2019_raw)}")
    # Si el CSV de 2019 tuviera algún valor no estándar para NaN (ej. 'N/A'),␣
 ↪añadir na_values=['N/A']
```

```
    df_2019_raw = pd.read_csv(file_path_2019_raw)
    logging.info(f"Archivo {os.path.basename(file_path_2019_raw)} cargado␣
 ↪exitosamente.")
    logging.info(f"El DataFrame df_2019_raw tiene {df_2019_raw.shape[0]} filas␣
 ↪y {df_2019_raw.shape[1]} columnas.")
    df_2019_clean = df_2019_raw.copy()
    logging.info("Copia del DataFrame creada como df_2019_clean para␣
 ↪transformaciones.")

except Exception as e:
    logging.error(f"Ocurrió un error al cargar {os.path.
 ↪basename(file_path_2019_raw)}: {e}")
```

```
2025-05-20 17:47:59,618 - INFO - Ruta del archivo raw a cargar:
/home/nicolas/Escritorio/workshops ETL/workshop_3/data/raw/2019.csv
2025-05-20 17:47:59,619 - INFO - Intentando cargar el archivo CSV: 2019.csv
2025-05-20 17:47:59,622 - INFO - Archivo 2019.csv cargado exitosamente.
2025-05-20 17:47:59,622 - INFO - El DataFrame df_2019_raw tiene 156 filas y 9
columnas.
2025-05-20 17:47:59,623 - INFO - Copia del DataFrame creada como df_2019_clean
para transformaciones.
2025-05-20 17:47:59,619 - INFO - Intentando cargar el archivo CSV: 2019.csv
2025-05-20 17:47:59,622 - INFO - Archivo 2019.csv cargado exitosamente.
2025-05-20 17:47:59,622 - INFO - El DataFrame df_2019_raw tiene 156 filas y 9
columnas.
2025-05-20 17:47:59,623 - INFO - Copia del DataFrame creada como df_2019_clean
para transformaciones.
```

[3]:
```
if df_2019_clean is not None:
    print("DataFrame 2019 cargado y copiado para limpieza:")
    print(df_2019_clean.head().to_markdown(index=False))
else:
    print("Error al cargar df_2019_raw. No se puede continuar con la limpieza.")
    logging.error("Deteniendo el proceso de limpieza debido a un error en la␣
 ↪carga de datos.")
```

```
DataFrame 2019 cargado y copiado para limpieza:
|   Overall rank | Country or region   |   Score |   GDP per capita |   Social
support |   Healthy life expectancy |   Freedom to make life choices |
Generosity |   Perceptions of corruption |
|---------------:|:--------------------|--------:|-----------------:|-----------
------:|--------------------------:|-------------------------------:|-----------
--:|----------------------------:|
|              1 | Finland             |   7.769 |            1.34  |
1.587 |                     0.986 |                          0.596 |
0.153 |                       0.393 |
|              2 | Denmark             |   7.6   |            1.383 |
```

```
1.573 |                    0.996 |                          0.592 |
0.252 |                      0.41  |
|             3 | Norway              |   7.554 |           1.488 |
1.582 |                    1.028 |                          0.603 |
0.271 |                      0.341 |
|             4 | Iceland             |   7.494 |           1.38  |
1.624 |                    1.026 |                          0.591 |
0.354 |                      0.118 |
|             5 | Netherlands         |   7.488 |           1.396 |
1.522 |                    0.999 |                          0.557 |
0.322 |                      0.298 |
```

[4]:
```python
# --- Estandarización de Nombres de Columnas para df_2019_clean ---
if df_2019_clean is not None:
    logging.info("Iniciando estandarización de nombres de columnas para df_2019.
 ↪")

    column_mapping_2019_to_standard = {
        'Country or region': 'country',
        'Overall rank': 'happiness_rank',
        'Score': 'happiness_score',
        'GDP per capita': 'economy_gdp_per_capita',
        'Social support': 'social_support',
        'Healthy life expectancy': 'health_life_expectancy',
        'Freedom to make life choices': 'freedom_to_make_life_choices',
        'Perceptions of corruption': 'perceptions_of_corruption',
        'Generosity': 'generosity'
    }

    df_2019_clean.rename(columns=column_mapping_2019_to_standard, inplace=True)

    logging.info(f"Columnas después del renombrado (df_2019_clean):␣
 ↪{df_2019_clean.columns.tolist()}")
    print("\nColumnas después del renombrado (df_2019_clean):")
    print(df_2019_clean.head(2).to_markdown(index=False))
else:
    logging.error("df_2019_clean no está cargado. No se puede renombrar␣
 ↪columnas.")
```

```
2025-05-20 17:47:59,655 - INFO - Iniciando estandarización de nombres de
columnas para df_2019.
2025-05-20 17:47:59,657 - INFO - Columnas después del renombrado
(df_2019_clean): ['happiness_rank', 'country', 'happiness_score',
'economy_gdp_per_capita', 'social_support', 'health_life_expectancy',
'freedom_to_make_life_choices', 'generosity', 'perceptions_of_corruption']


Columnas después del renombrado (df_2019_clean):
```

```
|   happiness_rank | country    |   happiness_score |   economy_gdp_per_capita |
social_support |   health_life_expectancy |   freedom_to_make_life_choices |
generosity |   perceptions_of_corruption |
|-----------------:|:-----------|------------------:|-------------------------:|-
---------------:|-------------------------:|-------------------------------:|--
-----------:|----------------------------:|
|                1 | Finland    |             7.769 |                     1.34  |
1.587 |                    0.986 |                          0.596 |        0.153
|                      0.393 |
|                2 | Denmark    |             7.6   |                     1.383 |
1.573 |                    0.996 |                          0.592 |        0.252
|                      0.41  |
```

[5]:
```
# --- Creación de la Columna 'region' para df_2019_clean ---
if df_2019_clean is not None:
    logging.info("Iniciando creación de la columna 'region' para df_2019.")

    # Mapeo actualizado para 2019 incluyendo 'Gambia'
    country_to_region_map_2019 = {
        "Western Europe": ['Finland', 'Denmark', 'Norway', 'Iceland',␣
 ↪'Netherlands', 'Switzerland',
                           'Sweden', 'Austria', 'Luxembourg', 'United␣
 ↪Kingdom', 'Ireland', 'Germany',
                           'Belgium', 'France', 'Spain', 'Portugal', 'Italy',␣
 ↪'Northern Cyprus', 'Cyprus', 'Malta', 'Greece'],
        "North America": ['Canada', 'United States'],
        "Australia and New Zealand": ['Australia', 'New Zealand'],
        "Middle East and Northern Africa": ['United Arab Emirates', 'Israel',␣
 ↪'Saudi Arabia', 'Qatar',
                                            'Bahrain', 'Kuwait', 'Jordan',␣
 ↪'Lebanon', 'Egypt', 'Libya',
                                            'Tunisia', 'Morocco', 'Algeria',␣
 ↪'Sudan', 'Palestinian Territories',
                                            'Syria', 'Armenia', 'Azerbaijan',␣
 ↪'Georgia', 'Iran', 'Iraq', 'Yemen', 'Turkey'],
        "Latin America and Caribbean": ['Chile', 'Guatemala', 'Costa Rica',␣
 ↪'Uruguay', 'Panama', 'Mexico',
                                        'Brazil', 'Argentina', 'El Salvador',␣
 ↪'Nicaragua', 'Colombia',
                                        'Ecuador', 'Peru', 'Bolivia',␣
 ↪'Paraguay', 'Venezuela',
                                        'Belize', 'Honduras', 'Jamaica',␣
 ↪'Dominican Republic', 'Trinidad & Tobago', 'Haiti'],
        "Southeastern Asia": ['Singapore', 'Thailand', 'Malaysia', 'Indonesia',␣
 ↪'Philippines', 'Vietnam',
                              'Cambodia', 'Myanmar', 'Laos'],
```

```
      "Central and Eastern Europe": ['Czech Republic', 'Slovakia',␣
↪'Slovenia', 'Poland', 'Hungary',
                                     'Croatia', 'Bosnia and Herzegovina',␣
↪'Serbia', 'Romania', 'Bulgaria',
                                     'Estonia', 'Latvia', 'Lithuania',␣
↪'Belarus', 'Moldova', 'Kosovo',
                                     'North Macedonia', 'Montenegro',␣
↪'Russia', 'Ukraine', 'Albania'],
      "Eastern Asia": ['China', 'Hong Kong', 'Japan', 'South Korea',␣
↪'Taiwan', 'Mongolia'],
      "Sub-Saharan Africa": ['Nigeria', 'South Africa', 'Kenya', 'Ethiopia',␣
↪'Uganda', 'Tanzania',
                             'Ghana', 'Senegal', 'Cameroon', 'Congo␣
↪(Kinshasa)', 'Congo (Brazzaville)',
                             'Angola', 'Benin', 'Burkina Faso', 'Rwanda',␣
↪'Zimbabwe', 'Zambia', 'Mozambique',
                             'Namibia', 'Madagascar', 'Botswana', 'Malawi',␣
↪'Niger', 'Mali', 'Chad',
                             'Central African Republic', 'South Sudan',␣
↪'Somalia', 'Sierra Leone',
                             'Liberia', 'Guinea', 'Ivory Coast', 'Mauritius',␣
↪'Gabon',
                             'Mauritania', 'Lesotho', 'Togo', 'Burundi',␣
↪'Comoros', 'Swaziland',
                             'Gambia'], # Añadido Gambia
      "Southern Asia": ['India', 'Pakistan', 'Bangladesh', 'Sri Lanka',␣
↪'Nepal', 'Bhutan', 'Afghanistan',
                        'Kazakhstan', 'Kyrgyzstan', 'Tajikistan',␣
↪'Turkmenistan', 'Uzbekistan']
  }

  region_lookup_2019 = {}
  for region, countries in country_to_region_map_2019.items():
      for country_name in countries:
          region_lookup_2019[country_name.strip()] = region

  # Estandarizar nombres de país en el DataFrame ANTES de mapear.
  # Revisa los nombres exactos en df_2019_clean['country'] y ajusta si es␣
↪necesario.
  country_name_replacements_2019 = {
      "Trinidad & Tobago": "Trinidad & Tobago", # Ejemplo
      # "Gambia, The": "Gambia", # Si el df usara "Gambia, The"
  }
  df_2019_clean['country_standardized_for_map'] = df_2019_clean['country'].
↪str.strip().replace(country_name_replacements_2019)
```

```
    df_2019_clean['region'] = df_2019_clean['country_standardized_for_map'].
 ↪map(region_lookup_2019)
    df_2019_clean.drop(columns=['country_standardized_for_map'], inplace=True)

    unmapped_countries_2019 = df_2019_clean[df_2019_clean['region'].
 ↪isnull()]['country'].unique()
    if len(unmapped_countries_2019) > 0:
        logging.warning(f"Países NO mapeados a región en df_2019:␣
 ↪{list(unmapped_countries_2019)}. Revisa el mapeo␣
 ↪'country_to_region_map_2019' y los nombres de país en el DataFrame␣
 ↪(especialmente después de 'country_name_replacements_2019').")
        print(f"\nADVERTENCIA: Países no mapeados a región en df_2019:␣
 ↪{list(unmapped_countries_2019)}.")
    else:
        logging.info("Todos los países de df_2019 fueron mapeados a una región␣
 ↪exitosamente.")

    logging.info("Columna 'region' creada y poblada en df_2019_clean.")
else:
    logging.error("df_2019_clean no está cargado. No se puede crear la columna␣
 ↪'region'.")
```

```
2025-05-20 17:47:59,671 - INFO - Iniciando creación de la columna 'region' para
df_2019.
2025-05-20 17:47:59,675 - INFO - Todos los países de df_2019 fueron mapeados a
una región exitosamente.
2025-05-20 17:47:59,676 - INFO - Columna 'region' creada y poblada en
df_2019_clean.
2025-05-20 17:47:59,675 - INFO - Todos los países de df_2019 fueron mapeados a
una región exitosamente.
2025-05-20 17:47:59,676 - INFO - Columna 'region' creada y poblada en
df_2019_clean.
```

[6]:
```
# --- Verificación de la Columna 'region' Creada (df_2019_clean) ---
if df_2019_clean is not None and 'region' in df_2019_clean.columns:
    logging.info("Verificando la columna 'region' creada en df_2019_clean.")

    unique_regions_2019 = df_2019_clean['region'].unique()
    print("\nValores únicos en la columna 'region' (df_2019_clean):")
    print(sorted([str(r) for r in unique_regions_2019]))
    logging.info(f"Valores únicos en 'region' (df_2019): {sorted([str(r) for r␣
 ↪in unique_regions_2019])}")

    print("\nConteo de países por región (df_2019_clean):")
    region_value_counts_2019 = df_2019_clean['region'].
 ↪value_counts(dropna=False)
    print(region_value_counts_2019.to_markdown())
```

```
    logging.info(f"Conteo de valores por región (df_2019):␣
 ↪{region_value_counts_2019.to_dict()}")

    print("\nEjemplo de países y sus regiones asignadas (df_2019_clean):")
    print(df_2019_clean[['country', 'region']].sample(5).
 ↪to_markdown(index=False))
else:
    if df_2019_clean is None: logging.error("df_2019_clean no está cargado.")
    else: logging.error("La columna 'region' no fue creada o no se encuentra en␣
 ↪df_2019_clean.")
    print("\nNo se puede verificar la columna 'region'.")
```

```
2025-05-20 17:47:59,685 - INFO - Verificando la columna 'region' creada en
df_2019_clean.
2025-05-20 17:47:59,686 - INFO - Valores únicos en 'region' (df_2019):
['Australia and New Zealand', 'Central and Eastern Europe', 'Eastern Asia',
'Latin America and Caribbean', 'Middle East and Northern Africa', 'North
America', 'Southeastern Asia', 'Southern Asia', 'Sub-Saharan Africa', 'Western
Europe']
2025-05-20 17:47:59,689 - INFO - Conteo de valores por región (df_2019): {'Sub-
Saharan Africa': 40, 'Middle East and Northern Africa': 22, 'Central and Eastern
Europe': 21, 'Western Europe': 21, 'Latin America and Caribbean': 21, 'Southern
Asia': 12, 'Southeastern Asia': 9, 'Eastern Asia': 6, 'Australia and New
Zealand': 2, 'North America': 2}


Valores únicos en la columna 'region' (df_2019_clean):
['Australia and New Zealand', 'Central and Eastern Europe', 'Eastern Asia',
'Latin America and Caribbean', 'Middle East and Northern Africa', 'North
America', 'Southeastern Asia', 'Southern Asia', 'Sub-Saharan Africa', 'Western
Europe']

Conteo de países por región (df_2019_clean):
| region                          |   count |
|:--------------------------------|--------:|
| Sub-Saharan Africa              |      40 |
| Middle East and Northern Africa |      22 |
| Central and Eastern Europe      |      21 |
| Western Europe                  |      21 |
| Latin America and Caribbean     |      21 |
| Southern Asia                   |      12 |
| Southeastern Asia               |       9 |
| Eastern Asia                    |       6 |
| Australia and New Zealand       |       2 |
| North America                   |       2 |

Ejemplo de países y sus regiones asignadas (df_2019_clean):
| country   | region                          |
```

```
|:----------|:------------------------------|
| Slovakia  | Central and Eastern Europe    |
| Libya     | Middle East and Northern Africa |
| Lesotho   | Sub-Saharan Africa            |
| Austria   | Western Europe                |
| Slovenia  | Central and Eastern Europe    |
```

[7]:
```
# --- Eliminación de Columnas no Deseadas de df_2019_clean ---
if df_2019_clean is not None:
    logging.info("No se eliminan columnas del DataFrame df_2019_clean según el␣
 ↪plan.")
    print("\nNo se eliminaron columnas de df_2019_clean.")
else:
    logging.error("df_2019_clean no está cargado.")
```

```
2025-05-20 17:47:59,698 - INFO - No se eliminan columnas del DataFrame
df_2019_clean según el plan.


No se eliminaron columnas de df_2019_clean.
```

[8]:
```
# --- Manejo de Valores Nulos en df_2019_clean ---
if df_2019_clean is not None:
    nulos_totales = df_2019_clean.isnull().sum().sum()
    if nulos_totales > 0:
        logging.warning(f"Se encontraron {nulos_totales} valores nulos en␣
 ↪df_2019_clean. Columnas afectadas:")
        print(f"\nColumnas con valores nulos en df_2019_clean:\n{df_2019_clean.
 ↪isnull().sum()[df_2019_clean.isnull().sum() > 0].to_markdown()}")
        # Aquí iría la lógica de imputación si fuera necesaria.
        # Por ahora, solo se reportan.
    else:
        logging.info("No se encontraron valores nulos para imputar en␣
 ↪df_2019_clean.")
        print("\nNo se encontraron valores nulos para imputar en df_2019_clean.
 ↪")
else:
    logging.error("df_2019_clean no está cargado. No se puede manejar nulos.")
```

```
2025-05-20 17:47:59,705 - INFO - No se encontraron valores nulos para imputar en
df_2019_clean.


No se encontraron valores nulos para imputar en df_2019_clean.
```

[9]:
```
# --- Añadir Columna 'year' ---
if df_2019_clean is not None:
    df_2019_clean['year'] = 2019 # Actualizado
```

```
    logging.info("Columna 'year' con valor 2019 añadida a df_2019_clean.")
else:
    logging.error("df_2019_clean no está cargado.")
```

```
2025-05-20 17:47:59,712 - INFO - Columna 'year' con valor 2019 añadida a
df_2019_clean.
```

[10]:
```
# --- Limpieza de Valores en Columna 'country' (Ejemplo) ---
if df_2019_clean is not None:
    if 'country' in df_2019_clean.columns:
        df_2019_clean['country'] = df_2019_clean['country'].str.strip()
        logging.info("Espacios extra eliminados de la columna 'country' en␣
 ↪df_2019_clean.")
else:
    logging.error("df_2019_clean no está cargado.")
```

```
2025-05-20 17:47:59,720 - INFO - Espacios extra eliminados de la columna
'country' en df_2019_clean.
```

[11]:
```
# --- Reordenamiento de Columnas para df_2019_clean ---
if df_2019_clean is not None:
    logging.info("Iniciando reordenamiento de columnas para df_2019_clean.")

    desired_column_order = [
        'year', 'region', 'country', 'happiness_rank', 'happiness_score',
        'social_support', 'health_life_expectancy', 'generosity',
        'freedom_to_make_life_choices', 'economy_gdp_per_capita',
        'perceptions_of_corruption'
    ]

    existing_columns_in_df = df_2019_clean.columns.tolist()
    final_column_order_2019 = [col for col in desired_column_order if col in␣
 ↪existing_columns_in_df]

    # ... (lógica completa de verificación de columnas faltantes/extras y␣
 ↪reordenamiento)
    missing_desired_cols = [col for col in desired_column_order if col not in␣
 ↪final_column_order_2019]
    if missing_desired_cols: logging.warning(f"Columnas deseadas no encontradas␣
 ↪en df_2019_clean: {missing_desired_cols}")
    extra_cols_in_df = [col for col in existing_columns_in_df if col not in␣
 ↪final_column_order_2019]
    if extra_cols_in_df: logging.warning(f"Columnas extra en df_2019_clean que␣
 ↪serán eliminadas: {extra_cols_in_df}")

    try:
        df_2019_clean = df_2019_clean[final_column_order_2019]
```

```
        logging.info(f"Columnas reordenadas para df_2019_clean. Nuevo orden:␣
 ↪{df_2019_clean.columns.tolist()}")
        print("\nDataFrame df_2019_clean después de reordenar columnas:")
        print(df_2019_clean.head(2).to_markdown(index=False))
    except KeyError as e:
        logging.error(f"Error al reordenar columnas en df_2019_clean: {e}")
else:
    logging.error("df_2019_clean no está cargado.")
```

```
2025-05-20 17:47:59,727 - INFO - Iniciando reordenamiento de columnas para
df_2019_clean.
2025-05-20 17:47:59,729 - INFO - Columnas reordenadas para df_2019_clean. Nuevo
orden: ['year', 'region', 'country', 'happiness_rank', 'happiness_score',
'social_support', 'health_life_expectancy', 'generosity',
'freedom_to_make_life_choices', 'economy_gdp_per_capita',
'perceptions_of_corruption']


DataFrame df_2019_clean después de reordenar columnas:
|   year | region         | country   |   happiness_rank |   happiness_score |
social_support |   health_life_expectancy |   generosity |
freedom_to_make_life_choices |   economy_gdp_per_capita |
perceptions_of_corruption |
|-------:|:---------------|:----------|-----------------:|------------------:|--
--------------:|-------------------------:|------------:|---------------------
----------:|-------------------------:|---------------------------:|
|   2019 | Western Europe | Finland   |                1 |             7.769 |
1.587 |                    0.986 |       0.153 |                           0.596
|                    1.34  |                       0.393 |
|   2019 | Western Europe | Denmark   |                2 |             7.6   |
1.573 |                    0.996 |       0.252 |                           0.592
|                    1.383 |                       0.41  |
```

[12]:
```
# --- Verificación de Tipos de Datos Final (df_2019_clean) ---
if df_2019_clean is not None:
    logging.info("Mostrando información final de df_2019_clean (tipos de datos).
 ↪")
    print("\nInformación final del DataFrame df_2019_clean:")
    df_2019_clean.info()
else:
    logging.error("df_2019_clean no está cargado.")
```

```
2025-05-20 17:47:59,737 - INFO - Mostrando información final de df_2019_clean
(tipos de datos).


Información final del DataFrame df_2019_clean:
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 156 entries, 0 to 155
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   year                            156 non-null    int64
 1   region                          156 non-null    object
 2   country                         156 non-null    object
 3   happiness_rank                  156 non-null    int64
 4   happiness_score                 156 non-null    float64
 5   social_support                  156 non-null    float64
 6   health_life_expectancy          156 non-null    float64
 7   generosity                      156 non-null    float64
 8   freedom_to_make_life_choices    156 non-null    float64
 9   economy_gdp_per_capita          156 non-null    float64
 10  perceptions_of_corruption       156 non-null    float64
dtypes: float64(7), int64(2), object(2)
memory usage: 13.5+ KB
```

[13]:
```python
# --- Guardar el DataFrame Limpio df_2019_clean (Opcional) ---
if df_2019_clean is not None:
    output_path_2019 = "/home/nicolas/Escritorio/workshops ETL/workshop_3/data/processed/2019_cleaned.csv" # Actualizado
    try:
        df_2019_clean.to_csv(output_path_2019, index=False)
        logging.info(f"DataFrame limpio df_2019_clean guardado en: {output_path_2019}")
        print(f"\nDataFrame limpio df_2019_clean guardado en: {output_path_2019}")
    except Exception as e:
        logging.error(f"Error al guardar el DataFrame limpio df_2019_clean: {e}")
else:
    logging.error("df_2019_clean no está cargado.")
```

```
2025-05-20 17:47:59,753 - INFO - DataFrame limpio df_2019_clean guardado en:
/home/nicolas/Escritorio/workshops
ETL/workshop_3/data/processed/2019_cleaned.csv


DataFrame limpio df_2019_clean guardado en: /home/nicolas/Escritorio/workshops
ETL/workshop_3/data/processed/2019_cleaned.csv
```