Avance 1

Importación de librerías

```
#Importamos las librerías necesarias
import pandas as pd
import numpy as np
```

Carga y transformación de datos

Lee el archivo data_latinoamerica.csv con código Python en tu Visual Studio Code

```
df = pd.read_csv('data_latinoamerica.csv')
```

Comprueba que el dataset cargado tiene la cantidad de registros y columnas especificadas

```
#Reviso la cantidad de registros cargados
df.shape
```

```
(12216057, 50)
```

```
#Reviso la información de cada columna
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12216057 entries, 0 to 12216056
Data columns (total 50 columns):
 #   Column                                 Dtype
---  ------                                 -----
 0   location_key                           object
 1   date                                   object
 2   country_code                           object
 3   country_name                           object
 4   new_confirmed                          float64
 5   new_deceased                           float64
 6   cumulative_confirmed                   float64
 7   cumulative_deceased                    float64
 8   cumulative_vaccine_doses_administered  float64
 9   population                             float64
 10  population_male                        float64
 11  population_female                      float64
 12  population_rural                       float64
 13  population_urban                       float64
 14  population_density                     float64
 15  human_development_index                float64
 16  population_age_00_09                   float64
```

```
 17  population_age_10_19                    float64
 18  population_age_20_29                    float64
 19  population_age_30_39                    float64
 20  population_age_40_49                    float64
 21  population_age_50_59                    float64
 22  population_age_60_69                    float64
 23  population_age_70_79                    float64
 24  population_age_80_and_older             float64
 25  gdp_usd                                 float64
 26  gdp_per_capita_usd                      float64
 27  latitude                                float64
 28  longitude                               float64
 29  area_sq_km                              float64
 30  smoking_prevalence                      float64
 31  diabetes_prevalence                     float64
 32  infant_mortality_rate                   float64
 33  nurses_per_1000                         float64
 34  physicians_per_1000                     float64
 35  average_temperature_celsius             float64
 36  minimum_temperature_celsius             float64
 37  maximum_temperature_celsius             float64
 38  rainfall_mm                             float64
 39  relative_humidity                       float64
 40  population_largest_city                 float64
 41  area_rural_sq_km                        float64
 42  area_urban_sq_km                        float64
 43  life_expectancy                         float64
 44  adult_male_mortality_rate               float64
 45  adult_female_mortality_rate             float64
 46  pollution_mortality_rate                float64
 47  comorbidity_mortality_rate              float64
 48  new_recovered                           float64
 49  cumulative_recovered                    float64
dtypes: float64(46), object(4)
memory usage: 4.6+ GB
```

```
#Reviso que la carga se haya realizado correctamente visualizando las primeras 10 filas
df.head(n=10)
```

```
  location_key        date country_code country_name  new_confirmed  \
0           AR  2020-01-01           AR    Argentina            3.0
1           AR  2020-01-02           AR    Argentina           14.0
2           AR  2020-01-03           AR    Argentina            3.0
3           AR  2020-01-04           AR    Argentina            7.0
4           AR  2020-01-05           AR    Argentina            5.0
5           AR  2020-01-06           AR    Argentina            9.0
6           AR  2020-01-07           AR    Argentina            4.0
7           AR  2020-01-08           AR    Argentina            3.0
8           AR  2020-01-09           AR    Argentina            0.0
```

```
9           AR  2020-01-10           AR    Argentina             1.0

   new_deceased  cumulative_confirmed  cumulative_deceased  \
0           0.0                   3.0                  0.0
1           0.0                  17.0                  0.0
2           0.0                  20.0                  0.0
3           0.0                  27.0                  0.0
4           0.0                  32.0                  0.0
5           0.0                  41.0                  0.0
6           0.0                  45.0                  0.0
7           0.0                  48.0                  0.0
8           0.0                  48.0                  0.0
9           0.0                  49.0                  0.0

   cumulative_vaccine_doses_administered  population  ...  \
0                                    NaN  44938712.0  ...
1                                    NaN  44938712.0  ...
2                                    NaN  44938712.0  ...
3                                    NaN  44938712.0  ...
4                                    NaN  44938712.0  ...
5                                    NaN  44938712.0  ...
6                                    NaN  44938712.0  ...
7                                    NaN  44938712.0  ...
8                                    NaN  44938712.0  ...
9                                    NaN  44938712.0  ...

   population_largest_city  area_rural_sq_km  area_urban_sq_km  \
0               15057273.0         2690269.0           55032.0
1               15057273.0         2690269.0           55032.0
2               15057273.0         2690269.0           55032.0
3               15057273.0         2690269.0           55032.0
4               15057273.0         2690269.0           55032.0
5               15057273.0         2690269.0           55032.0
6               15057273.0         2690269.0           55032.0
7               15057273.0         2690269.0           55032.0
8               15057273.0         2690269.0           55032.0
9               15057273.0         2690269.0           55032.0

   life_expectancy  adult_male_mortality_rate
adult_female_mortality_rate  \
0            76.52                   147.086
79.483
1            76.52                   147.086
79.483
2            76.52                   147.086
79.483
3            76.52                   147.086
79.483
4            76.52                   147.086
79.483
```

```
5             76.52                     147.086
79.483
6             76.52                     147.086
79.483
7             76.52                     147.086
79.483
8             76.52                     147.086
79.483
9             76.52                     147.086
79.483

   pollution_mortality_rate  comorbidity_mortality_rate  new_recovered
\
0                      26.6                        15.8            NaN

1                      26.6                        15.8            NaN

2                      26.6                        15.8            NaN

3                      26.6                        15.8            NaN

4                      26.6                        15.8            NaN

5                      26.6                        15.8            NaN

6                      26.6                        15.8            NaN

7                      26.6                        15.8            NaN

8                      26.6                        15.8            NaN

9                      26.6                        15.8            NaN


   cumulative_recovered
0                   NaN
1                   NaN
2                   NaN
3                   NaN
4                   NaN
5                   NaN
6                   NaN
7                   NaN
8                   NaN
9                   NaN

[10 rows x 50 columns]
```

Filtrado de los datos

Selecciona los países donde se expandirán: Colombia, Argentina, Chile, México, Perú y Brasil

```
#Reviso el listado de paises por KEY
df['location_key'].unique()
```

```
array(['AR', 'AR_A', 'AR_A_007', ..., 'VE_X', 'VE_Y', 'VE_Z'],
      shape=(12327,), dtype=object)
```

```
#Analizo la estructura de los códigos de locación
df.groupby(df['location_key'])['population'].max()
```

```
location_key
AR          44938712.0
AR_A         1406584.0
AR_A_007       57411.0
AR_A_014        7315.0
AR_A_021       14850.0
               ...
VE_U          499049.0
VE_V         4323467.0
VE_X               NaN
VE_Y          167676.0
VE_Z         1422000.0
Name: population, Length: 12327, dtype: float64
```

Se observa una dependencia jerárquica en los códigos

```
#Confirmo la dependencia jerárquica entre los códigos, demostrando la
existencia de distintos niveles que duplican la información
df['nivel'] = df['location_key'].str.count('_')
df[['location_key','nivel']].drop_duplicates().sort_values('nivel')
```

```
         location_key  nivel
0                  AR      0
12171462           UY      0
550005             BR      0
6115461            CL      0
6475194            CO      0
...               ...    ...
20811        AR_A_161      2
19820        AR_A_154      2
3964         AR_A_021      2
2973         AR_A_014      2
1982         AR_A_007      2

[12327 rows x 2 columns]
```

Con la dependencia demostrada, decido filtrar por location key

```
#Creo la variable de filtrado con los key de los paises solicitados
paises_a_filtrar=["CO", "AR", "CL", "MX", "PE", "BR"]

#Filtro según los paises solicitados
df_filtrado= df.loc[df['location_key'].isin(paises_a_filtrar)]

#Pido la cantidad total de filas luego del filtrado
df_filtrado.shape
```

```
(5946, 51)
```

Sobre un total de 12.216.057 filas, quedan 5.946 luego del filtrado por país

Filtra los datos en fechas mayores a 2021-01-01

```
#Creo la variable de filtrado con la fecha
fecha_a_filtrar='2021-01-01'

#Filtro según la fecha
df_filtrado= df_filtrado.loc[df_filtrado['date']>=fecha_a_filtrar]

#Pido la cantidad total de filas luego del filtrado
df_filtrado.shape
```

```
(3750, 51)
```

Luego del nuevo filtrado, se obtienen un total de 3.750 filas

Compara a nivel de país para llenar valores faltantes

```
#Creo la variable para contar la cantidad de registros nulos por país
valores_nulos_por_pais =
df_filtrado.groupby('country_name').apply(lambda x: x.isnull().sum())

#Muestro el resultado de la variable creada
valores_nulos_por_pais
```

```
C:\Users\loren\AppData\Local\Temp\ipykernel_29652\254299692.py:2:
FutureWarning: DataFrameGroupBy.apply operated on the grouping
columns. This behavior is deprecated, and in a future version of
pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or
explicitly select the grouping columns after groupby to silence this
warning.
  valores_nulos_por_pais =
df_filtrado.groupby('country_name').apply(lambda x: x.isnull().sum())

              location_key  date  country_code  country_name
new_confirmed  \
country_name
```

```
Argentina                  0     0              0              0
4
Brazil                     0     0              0              0
2
Chile                      0     0              0              0
4
Colombia                   0     0              0              0
4
Mexico                     0     0              0              0
3
Peru                       0     0              0              0
4

               new_deceased  cumulative_confirmed  cumulative_deceased
\
country_name

Argentina                  4                     4                     4

Brazil                     2                     2                     2

Chile                      4                     4                     4

Colombia                   4                     4                     4

Mexico                     3                     3                     3

Peru                       4                     4                     4


               cumulative_vaccine_doses_administered
population  ...  \
country_name                                                        ...

Argentina                                          4           0  ...

Brazil                                            18           0  ...

Chile                                             14           0  ...

Colombia                                         291           0  ...

Mexico                                           218           0  ...

Peru                                              44           0  ...


               area_rural_sq_km  area_urban_sq_km  life_expectancy  \
country_name
Argentina                     0                 0                0
Brazil                        0                 0                0
```

```
Chile                          0                 0                 0
Colombia                       0                 0                 0
Mexico                         0                 0                 0
Peru                           0                 0                 0

              adult_male_mortality_rate
adult_female_mortality_rate  \
country_name

Argentina                                  0                                  0

Brazil                                     0                                  0

Chile                                      0                                  0

Colombia                                   0                                  0

Mexico                                     0                                  0

Peru                                       0                                  0


              pollution_mortality_rate  comorbidity_mortality_rate  \
country_name
Argentina                            0                           0
Brazil                               0                           0
Chile                                0                           0
Colombia                             0                           0
Mexico                               0                           0
Peru                                 0                           0

              new_recovered  cumulative_recovered  nivel
country_name
Argentina               625                   625      0
Brazil                    3                     3      0
Chile                     4                   625      0
Colombia                237                   237      0
Mexico                  625                   625      0
Peru                    625                   625      0

[6 rows x 51 columns]
```

Realiza una limpieza preliminar de los datos, eliminando registros nulos y corrigiendo los tipos de datos donde sea necesario, trata con valores medios, valores anteriores o valores siguientes

Utilizando un bucle For, completo la lista de columnas a imputar con un promedio de los valores para las variables climáticas y con el valor anterior para los acumulados (por location_key)

```
# Defino los grupos de columnas según el tipo de dato
columnas_acumuladas = [
    'cumulative_confirmed',
    'cumulative_deceased',
    'cumulative_recovered',
    'cumulative_vaccine_doses_administered'
]

columnas_climaticas = [
    'rainfall_mm',
    'relative_humidity',
    'average_temperature_celsius',
    'minimum_temperature_celsius',
    'maximum_temperature_celsius'
]

# Itero país por país para imputar valores
for pais in paises_a_filtrar:
    mask = df_filtrado['location_key'] == pais

    # --- 1. Variables acumuladas ---
    # Se completan con el último valor válido hacia adelante. Esto
preserva la naturaleza acumulativa de la serie temporal
    df_filtrado.loc[mask, columnas_acumuladas] = (
        df_filtrado.loc[mask, columnas_acumuladas]
        .ffill()
    )

    # --- 2. Variables climáticas ---
    # Se completan con la media del país. Esto mantiene coherencia en
caso de datos faltantes puntuales
    for col in columnas_climaticas:
        media_pais = df_filtrado.loc[mask, col].mean()
        df_filtrado.loc[mask, col] = df_filtrado.loc[mask,
col].fillna(media_pais)
```

Utilizando un bucle For, completo los valores faltanes en las columnas especificadas, con cero

```
# Imputamos valores faltantes con 0 en columnas específicas
columnas_a_llenar_con_cero = ['new_confirmed', 'new_deceased',
'new_recovered', ]

for columna in columnas_a_llenar_con_cero:
    df_filtrado[columna] = df_filtrado [columna].fillna(0)
```

Verificación del cambio en las columnas imputadas

```
#Verifico la información final del DataFrame
print(df_filtrado.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3750 entries, 366 to 10253876
Data columns (total 51 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   location_key                            3750 non-null   object 
 1   date                                    3750 non-null   object 
 2   country_code                            3750 non-null   object 
 3   country_name                            3750 non-null   object 
 4   new_confirmed                           3750 non-null   float64
 5   new_deceased                            3750 non-null   float64
 6   cumulative_confirmed                    3750 non-null   float64
 7   cumulative_deceased                     3750 non-null   float64
 8   cumulative_vaccine_doses_administered   3642 non-null   float64
 9   population                              3750 non-null   float64
 10  population_male                         3750 non-null   float64
 11  population_female                       3750 non-null   float64
 12  population_rural                        3750 non-null   float64
 13  population_urban                        3750 non-null   float64
 14  population_density                      3750 non-null   float64
 15  human_development_index                 3750 non-null   float64
 16  population_age_00_09                    3750 non-null   float64
 17  population_age_10_19                    3750 non-null   float64
 18  population_age_20_29                    3750 non-null   float64
 19  population_age_30_39                    3750 non-null   float64
 20  population_age_40_49                    3750 non-null   float64
 21  population_age_50_59                    3750 non-null   float64
 22  population_age_60_69                    3750 non-null   float64
 23  population_age_70_79                    3750 non-null   float64
 24  population_age_80_and_older             3750 non-null   float64
 25  gdp_usd                                 3750 non-null   float64
 26  gdp_per_capita_usd                      3750 non-null   float64
 27  latitude                                3750 non-null   float64
 28  longitude                               3750 non-null   float64
 29  area_sq_km                              3750 non-null   float64
 30  smoking_prevalence                      3750 non-null   float64
 31  diabetes_prevalence                     3750 non-null   float64
 32  infant_mortality_rate                   3750 non-null   float64
 33  nurses_per_1000                         3750 non-null   float64
 34  physicians_per_1000                     3750 non-null   float64
 35  average_temperature_celsius             3750 non-null   float64
 36  minimum_temperature_celsius             3750 non-null   float64
 37  maximum_temperature_celsius             3750 non-null   float64
 38  rainfall_mm                             3750 non-null   float64
 39  relative_humidity                       3750 non-null   float64
 40  population_largest_city                 3750 non-null   float64
 41  area_rural_sq_km                        3750 non-null   float64
 42  area_urban_sq_km                        3750 non-null   float64
 43  life_expectancy                         3750 non-null   float64
 44  adult_male_mortality_rate               3750 non-null   float64
```

```
 45  adult_female_mortality_rate             3750 non-null   float64
 46  pollution_mortality_rate                3750 non-null   float64
 47  comorbidity_mortality_rate              3750 non-null   float64
 48  new_recovered                           3750 non-null   float64
 49  cumulative_recovered                    1250 non-null   float64
 50  nivel                                   3750 non-null   int64
dtypes: float64(46), int64(1), object(4)
memory usage: 1.5+ MB
None
```

Guarda los datos filtrados en un archivo con el nombre DatosFinalesFiltrado.csv a fin de poder utilizarlo luego y no tener que repetir el proceso de filtrado y limpieza

Guardado de DataFrame filtrado

```
#Guardo el DataFrame limpio en un nuevo archivo CSV
df_filtrado.to_csv('DatosFinalesFiltrado.csv', index=False)
```

Anáisis a partir del DataSet ya filtrado

Carga del nuevo DataSet ya filtrado

```
df_limpio = pd.read_csv('DatosFinalesFiltrado.csv')

#Verifico las columnas del df limpio
print(df_limpio.columns)
```

```
Index(['location_key', 'date', 'country_code', 'country_name',
'new_confirmed',
       'new_deceased', 'cumulative_confirmed', 'cumulative_deceased',
       'cumulative_vaccine_doses_administered', 'population',
       'population_male', 'population_female', 'population_rural',
       'population_urban', 'population_density',
'human_development_index',
       'population_age_00_09', 'population_age_10_19',
'population_age_20_29',
       'population_age_30_39', 'population_age_40_49',
'population_age_50_59',
       'population_age_60_69', 'population_age_70_79',
       'population_age_80_and_older', 'gdp_usd', 'gdp_per_capita_usd',
       'latitude', 'longitude', 'area_sq_km', 'smoking_prevalence',
       'diabetes_prevalence', 'infant_mortality_rate',
'nurses_per_1000',
       'physicians_per_1000', 'average_temperature_celsius',
       'minimum_temperature_celsius', 'maximum_temperature_celsius',
       'rainfall_mm', 'relative_humidity', 'population_largest_city',
       'area_rural_sq_km', 'area_urban_sq_km', 'life_expectancy',
       'adult_male_mortality_rate', 'adult_female_mortality_rate',
       'pollution_mortality_rate', 'comorbidity_mortality_rate',
```

```
       'new_recovered', 'cumulative_recovered', 'nivel'],
      dtype='object')
```

Estadísticas y Métricas

Aplica bucles for y/o while para el cálculo de estadísticas descriptivas y otras métricas importantes que ofrece pandas por default

```
# Selecciono solo las columnas numéricas de interés
columnas_numericas = [
    'new_confirmed', 'new_deceased', 'cumulative_confirmed',
    'cumulative_deceased', 'cumulative_vaccine_doses_administered',
    'population', 'population_density', 'population_urban',
'population_rural',
    'nurses_per_1000', 'physicians_per_1000', 'life_expectancy',
'diabetes_prevalence',
    'smoking_prevalence'
]

# Con un bucle For, calculo estadísticas descriptivas en cada una de
las columnas seleccionadas
for col in columnas_numericas:
    print(f"--- {col} ---")
    print("Media:", df_limpio[col].mean())
    print("Mediana:", df_limpio[col].median())
    print("Desvío estándar:", df_limpio[col].std())
    print("Mínimo:", df_limpio[col].min())
    print("Máximo:", df_limpio[col].max())
    print()
```

```
--- new_confirmed ---
Media: 13846.620533333333
Mediana: 5102.0
Desvío estándar: 24212.287393216036
Mínimo: -573.0
Máximo: 298408.0

--- new_deceased ---
Media: 275.748
Mediana: 98.0
Desvío estándar: 507.2040080719143
Mínimo: 0.0
Máximo: 11447.0

--- cumulative_confirmed ---
Media: 6786495.4992
Mediana: 3640785.5
Desvío estándar: 8001192.793824149
Mínimo: 971.0
```

```
Máximo: 34568833.0

--- cumulative_deceased ---
Media: 192863.5696
Mediana: 139614.0
Desvío estándar: 182912.08558762528
Mínimo: 1.0
Máximo: 685203.0

--- cumulative_vaccine_doses_administered ---
Media: 83091713.00713894
Mediana: 49955599.0
Desvío estándar: 93853024.13063993
Mínimo: 18.0
Máximo: 347868481.0

--- population ---
Media: 77721474.16666667
Mediana: 47910798.0
Desvío estándar: 67137742.833887
Mínimo: 17574003.0
Máximo: 212559409.0

--- population_density ---
Media: 34.26683333333334
Mediana: 25.7345
Desvío estándar: 16.839470603642653
Mínimo: 16.515
Máximo: 66.325

--- population_urban ---
Media: 68339307.83333333
Mediana: 41083436.5
Desvío estándar: 58295278.79012064
Mínimo: 16610135.0
Máximo: 183241641.0

--- population_rural ---
Media: 12554975.833333334
Mediana: 8316127.5
Desvío estándar: 10080740.043868225
Mínimo: 2341903.0
Máximo: 27807886.0

--- nurses_per_1000 ---
Media: 5.368366666666668
Mediana: 2.5197000000000003
Desvío estándar: 4.605890594633589
Mínimo: 1.3309
Máximo: 13.3248
```

```
--- physicians_per_1000 ---
Media: 2.4363166666666665
Mediana: 2.28375
Desvío estándar: 0.8020514495942896
Mínimo: 1.3048
Máximo: 3.9901

--- life_expectancy ---
Media: 76.8085
Mediana: 76.518
Desvío estándar: 1.5973936149332786
Mínimo: 74.992
Máximo: 80.042

--- diabetes_prevalence ---
Media: 8.733333333333334
Mediana: 8.0
Desvío estándar: 2.578241295224828
Mínimo: 5.9
Máximo: 13.5

--- smoking_prevalence ---
Media: 16.883333333333336
Mediana: 13.95
Desvío estándar: 10.70227074886466
Mínimo: 4.8
Máximo: 37.8
```

Crea una función que permita obtener la mediana, varianza y el rango

```python
#Creo una función que calcula mediana, varianza y rango para una
columna numérica de un DataFrame.
def calcular_estadisticas(df, columna):

    mediana = df[columna].median()
    varianza = df[columna].var()
    rango = df[columna].max() - df[columna].min()

    return {
       print(f"Columna: {columna}"),
       print(f"  Mediana: {mediana:.2f}"),
       print(f"  Varianza: {varianza:.2f}"),
       print(f"  Rango: {rango:.2f}")
    }

# Pruebo la función con la columna 'population':
resultado = calcular_estadisticas(df_limpio, 'population')
```

```
Columna: population
  Mediana: 47910798.00
  Varianza: 4507476512829145.00
  Rango: 194985406.00
```

Avance 2

Importación de librerías

```
import seaborn as sns
import matplotlib.pyplot as plt
import os
```

Estadística

Análisis Estadístico con Pandas y Numpy: Explora las propiedades estadísticas del dataset. Calcula medidas de tendencia central, dispersión y correlaciones entre las variables para entender mejor la situación actual y las necesidades de las áreas en estudio -> recuerda que filtraste por país

```
# Con un Bubcle For recorro las columnas numéricas para obtener el describe de cada una
for columna in df_limpio.columns:
    if df_limpio[columna].dtype in ['int64', 'float64']:
        print(columna)
        print(df_limpio[columna].describe())
        print('_' * 40)
```

```
new_confirmed
count      3750.000000
mean      13846.620533
std       24212.287393
min        -573.000000
25%        1511.000000
50%        5102.000000
75%       14802.500000
max      298408.000000
Name: new_confirmed, dtype: float64
________________________________________
new_deceased
count     3750.000000
mean       275.748000
std        507.204008
min          0.000000
25%         25.000000
50%         98.000000
```

```
75%        311.750000
max      11447.000000
Name: new_deceased, dtype: float64
________________________________________________
cumulative_confirmed
count    3.750000e+03
mean     6.786495e+06
std      8.001193e+06
min      9.710000e+02
25%      2.130490e+06
50%      3.640786e+06
75%      6.247634e+06
max      3.456883e+07
Name: cumulative_confirmed, dtype: float64
________________________________________________
cumulative_deceased
count      3750.000000
mean     192863.569600
std      182912.085588
min           1.000000
25%       59241.250000
50%      139614.000000
75%      215073.000000
max      685203.000000
Name: cumulative_deceased, dtype: float64
________________________________________________
cumulative_vaccine_doses_administered
count    3.642000e+03
mean     8.309171e+07
std      9.385302e+07
min      1.800000e+01
25%      1.876207e+07
50%      4.995560e+07
75%      1.021711e+08
max      3.478685e+08
Name: cumulative_vaccine_doses_administered, dtype: float64
________________________________________________
population
count    3.750000e+03
mean     7.772147e+07
std      6.713774e+07
min      1.757400e+07
25%      2.938188e+07
50%      4.791080e+07
75%      1.109920e+08
max      2.125594e+08
Name: population, dtype: float64
________________________________________________
population_male
```

```
count    3.750000e+03
mean     3.787035e+07
std      3.318493e+07
min      8.972014e+06
25%      1.445076e+07
50%      2.225416e+07
75%      5.485523e+07
max      1.044358e+08
Name: population_male, dtype: float64
_________________________________________________
population_female
count    3.750000e+03
mean     3.927162e+07
std      3.448406e+07
min      8.601989e+06
25%      1.493113e+07
50%      2.324582e+07
75%      5.748131e+07
max      1.081236e+08
Name: population_female, dtype: float64
_________________________________________________
population_rural
count    3.750000e+03
mean     1.255498e+07
std      1.008074e+07
min      2.341903e+06
25%      3.599141e+06
50%      8.316128e+06
75%      2.494867e+07
max      2.780789e+07
Name: population_rural, dtype: float64
_________________________________________________
population_urban
count    3.750000e+03
mean     6.833931e+07
std      5.829528e+07
min      1.661014e+07
25%      2.539034e+07
50%      4.108344e+07
75%      1.026269e+08
max      1.832416e+08
Name: population_urban, dtype: float64
_________________________________________________
population_density
count    3750.000000
mean       34.266833
std        16.839471
min        16.515000
25%        25.431000
```

```
50%        25.734500
75%        45.861000
max        66.325000
Name: population_density, dtype: float64
_________________________________________________
human_development_index
count    3750.000000
mean        0.777333
std         0.036353
min         0.747000
25%         0.750000
50%         0.755000
75%         0.825000
max         0.832000
Name: human_development_index, dtype: float64
_________________________________________________
population_age_00_09
count    3.750000e+03
mean     1.205908e+07
std      9.755702e+06
min      2.428079e+06
25%      5.140624e+06
50%      7.066513e+06
75%      2.157586e+07
max      2.907691e+07
Name: population_age_00_09, dtype: float64
_________________________________________________
population_age_10_19
count    3.750000e+03
mean     1.263694e+07
std      1.034931e+07
min      2.493879e+06
25%      5.035905e+06
50%      7.582694e+06
75%      2.196605e+07
max      3.116045e+07
Name: population_age_10_19, dtype: float64
_________________________________________________
population_age_20_29
count    3.750000e+03
mean     1.265890e+07
std      1.082701e+07
min      2.995538e+06
25%      4.895056e+06
50%      7.638860e+06
75%      1.868045e+07
max      3.410464e+07
Name: population_age_20_29, dtype: float64
_________________________________________________
```

```
population_age_30_39
count    3.750000e+03
mean     1.202553e+07
std      1.098801e+07
min      2.945404e+06
25%      4.379777e+06
50%      6.793724e+06
75%      1.676378e+07
max      3.447676e+07
Name: population_age_30_39, dtype: float64
_________________________________________
population_age_40_49
count    3.750000e+03
mean     9.932776e+06
std      9.359076e+06
min      2.578404e+06
25%      3.660378e+06
50%      5.478956e+06
75%      1.293796e+07
max      2.946201e+07
Name: population_age_40_49, dtype: float64
_________________________________________
population_age_50_59
count    3.750000e+03
mean     7.993603e+06
std      7.666352e+06
min      2.352271e+06
25%      2.772568e+06
50%      4.727962e+06
75%      8.959656e+06
max      2.442120e+07
Name: population_age_50_59, dtype: float64
_________________________________________
population_age_60_69
count    3.750000e+03
mean     5.442399e+06
std      5.270852e+06
min      1.791787e+06
25%      1.846407e+06
50%      3.342804e+06
75%      5.433731e+06
max      1.689686e+07
Name: population_age_60_69, dtype: float64
_________________________________________
population_age_70_79
count    3.750000e+03
mean     2.964913e+06
std      2.702992e+06
min      9.931260e+05
```

```
25%      1.078066e+06
50%      1.898659e+06
75%      3.119417e+06
max      8.801551e+06
Name: population_age_70_79, dtype: float64
_________________________________________
population_age_80_and_older
count    3.750000e+03
mean     1.451953e+06
std      1.252169e+06
min      5.377210e+05
25%      5.731030e+05
50%      9.698185e+05
75%      1.502231e+06
max      4.159027e+06
Name: population_age_80_and_older, dtype: float64
_________________________________________
gdp_usd
count    3.750000e+03
mean     7.301129e+11
std      6.066856e+11
min      2.268481e+11
25%      2.823182e+11
50%      3.867331e+11
75%      1.258287e+12
max      1.839758e+12
Name: gdp_usd, dtype: float64
_________________________________________
gdp_per_capita_usd
count     3750.000000
mean      9481.833333
std       2766.047856
min       6432.000000
25%       6977.000000
50%       9290.000000
75%      10006.000000
max      14896.000000
Name: gdp_per_capita_usd, dtype: float64
_________________________________________
latitude
count    3750.000000
mean      -10.566667
std        20.021133
min       -34.000000
25%       -33.000000
50%       -11.700000
75%         4.000000
max        23.000000
Name: latitude, dtype: float64
```

```
________________________________________________
longitude
count    3750.000000
mean      -73.208333
std        14.917317
min      -102.000000
25%       -76.000000
50%       -72.125000
75%       -64.000000
max       -53.000000
Name: longitude, dtype: float64
________________________________________________
area_sq_km
count    3.750000e+03
mean     2.740702e+06
std      2.663906e+06
min      7.567000e+05
25%      1.141748e+06
50%      1.624798e+06
75%      2.780400e+06
max      8.515770e+06
Name: area_sq_km, dtype: float64
________________________________________________
smoking_prevalence
count    3750.000000
mean       16.883333
std        10.702271
min         4.800000
25%         9.000000
50%        13.950000
75%        21.800000
max        37.800000
Name: smoking_prevalence, dtype: float64
________________________________________________
diabetes_prevalence
count    3750.000000
mean        8.733333
std         2.578241
min         5.900000
25%         6.600000
50%         8.000000
75%        10.400000
max        13.500000
Name: diabetes_prevalence, dtype: float64
________________________________________________
infant_mortality_rate
count    3750.00000
mean       10.35000
std         2.23767
```

```
min          6.20000
25%          8.80000
50%         11.05000
75%         12.20000
max         12.80000
Name: infant_mortality_rate, dtype: float64
_________________________________________
nurses_per_1000
count    3750.000000
mean        5.368367
std         4.605891
min         1.330900
25%         2.396100
50%         2.519700
75%        10.119000
max        13.324800
Name: nurses_per_1000, dtype: float64
_________________________________________
physicians_per_1000
count    3750.000000
mean        2.436317
std         0.802051
min         1.304800
25%         2.164300
50%         2.283750
75%         2.591200
max         3.990100
Name: physicians_per_1000, dtype: float64
_________________________________________
average_temperature_celsius
count    3750.000000
mean       21.025519
std         6.046220
min         3.432099
25%        17.271605
50%        21.590741
75%        25.800000
max        39.138889
Name: average_temperature_celsius, dtype: float64
_________________________________________
minimum_temperature_celsius
count    3750.000000
mean       14.966547
std         6.993140
min        -5.383333
25%         9.584877
50%        15.886111
75%        20.800000
max        33.000000
```

```
Name: minimum_temperature_celsius, dtype: float64
________________________________________
maximum_temperature_celsius
count    3750.000000
mean       27.062621
std         5.229628
min         6.950617
25%        24.000000
50%        27.410416
75%        30.868827
max        41.944444
Name: maximum_temperature_celsius, dtype: float64
________________________________________
rainfall_mm
count    3750.000000
mean        1.457207
std         4.292579
min         0.000000
25%         0.000000
50%         0.000000
75%         0.453277
max        46.736000
Name: rainfall_mm, dtype: float64
________________________________________
relative_humidity
count    3750.000000
mean       62.300888
std        15.803864
min        10.296407
25%        51.959500
50%        65.229235
75%        74.364487
max        94.817706
Name: relative_humidity, dtype: float64
________________________________________
population_largest_city
count    3.750000e+03
mean     1.443888e+07
std      5.710076e+06
min      6.723516e+06
25%      1.055471e+07
50%      1.291832e+07
75%      2.167191e+07
max      2.184651e+07
Name: population_largest_city, dtype: float64
________________________________________
area_rural_sq_km
count    3.750000e+03
mean     2.636580e+06
```

```
std      2.584740e+06
min      7.094180e+05
25%      1.090598e+06
50%      1.543881e+06
75%      2.690269e+06
max      8.241430e+06
Name: area_rural_sq_km, dtype: float64
_________________________________________________
area_urban_sq_km
count      3750.000000
mean      59502.500000
std       45125.310561
min       12027.000000
25%       16425.000000
50%       45582.000000
75%      102418.000000
max      134981.000000
Name: area_urban_sq_km, dtype: float64
_________________________________________________
life_expectancy
count    3750.000000
mean       76.808500
std         1.597394
min        74.992000
25%        75.672000
50%        76.518000
75%        77.109000
max        80.042000
Name: life_expectancy, dtype: float64
_________________________________________________
adult_male_mortality_rate
count    3750.000000
mean      154.274667
std        27.000140
min       107.669000
25%       146.370000
50%       149.351000
75%       184.379000
max       188.528000
Name: adult_male_mortality_rate, dtype: float64
_________________________________________________
adult_female_mortality_rate
count    3750.000000
mean       81.428000
std        11.800621
min        59.035000
25%        77.999000
50%        82.149000
75%        91.421000
```

```
max        95.815000
Name: adult_female_mortality_rate, dtype: float64
_________________________________________
pollution_mortality_rate
count    3750.000000
mean       36.566667
std        13.027566
min        25.300000
25%        26.600000
50%        33.300000
75%        37.000000
max        63.900000
Name: pollution_mortality_rate, dtype: float64
_________________________________________
comorbidity_mortality_rate
count    3750.000000
mean       14.816667
std         1.665972
min        12.400000
25%        12.600000
50%        15.750000
75%        15.800000
max        16.600000
Name: comorbidity_mortality_rate, dtype: float64
_________________________________________
new_recovered
count      3750.000000
mean       8263.965600
std       23037.778773
min      -31119.000000
25%           0.000000
50%           0.000000
75%        1833.500000
max      282957.000000
Name: new_recovered, dtype: float64
_________________________________________
cumulative_recovered
count    1.250000e+03
mean     1.278349e+07
std      1.013277e+07
min      1.461223e+06
25%      4.861089e+06
50%      6.094043e+06
75%      2.109625e+07
max      3.370623e+07
Name: cumulative_recovered, dtype: float64
_________________________________________
nivel
count    3750.0
mean        0.0
```

```
std          0.0
min          0.0
25%          0.0
50%          0.0
75%          0.0
max          0.0
Name: nivel, dtype: float64
```

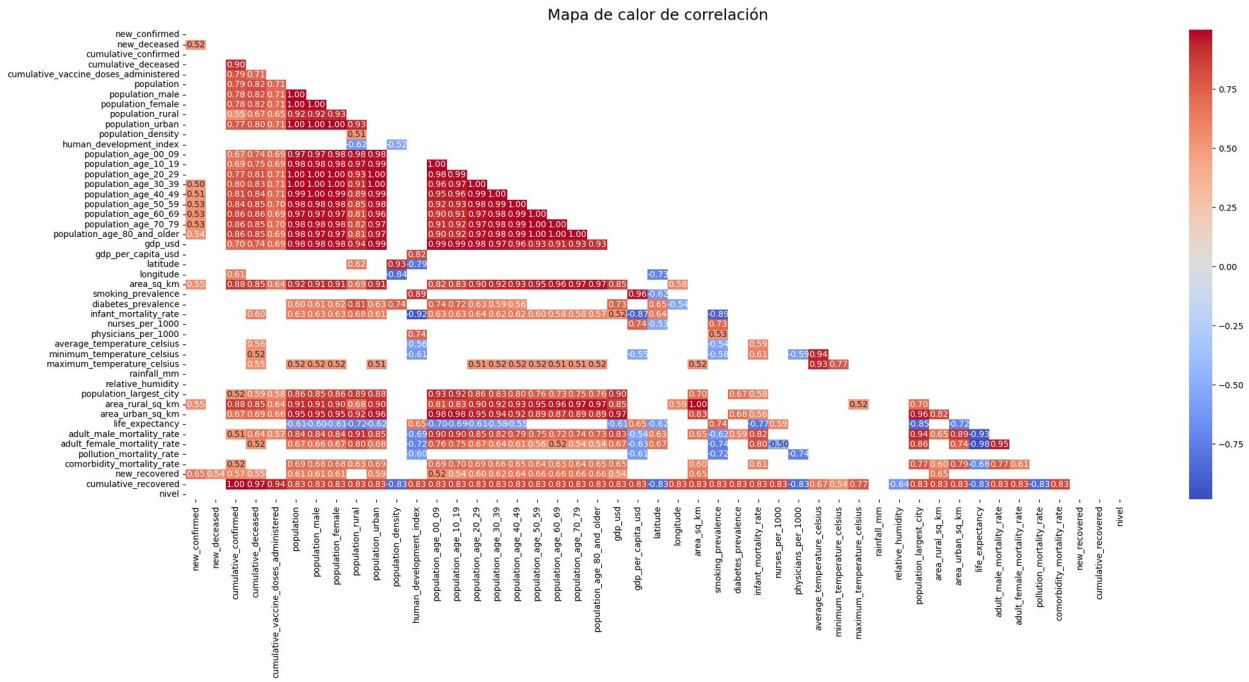
Indexación de la columna Date

```
# Convierto la columna "date" del DataFrame en un objeto de tipo fecha
(datetime64) para poder trabajar con series temporales
df_limpio['date'] =pd.to_datetime(df_limpio['date'])

# Establezco la columna "date" como índice del DataFrame.
df_limpio.set_index('date', inplace=True)
```

Mapa de Correlación

Genero un mapa de calor para visualizar la correlación entre las variables numéricas del DataFrame.

```
# Selecciono sólo las columnas numéricas
columnas_numericas = df_limpio.select_dtypes(include=[np.number])

# Creo la matriz de correlación
correlacion = columnas_numericas.corr()

#Creo máscara para el triángulo superior
mask = np.triu(np.ones_like(correlacion, dtype=bool))

#   Muestro sólo los valores fuertes (entre -0.5 y 0.5)
correlacion_filtrada = correlacion[abs(correlacion) > 0.5]

# Creo la grafica
plt.figure(figsize=(25, 10))
mapadecalor= sns.heatmap(correlacion_filtrada, annot=True, fmt=".2f",
cmap='coolwarm', mask=mask)
mapadecalor.set_title('Mapa de calor de correlación',
fontdict={'fontsize':18}, pad=12)
plt.show()
```



Se observa que la administración de vacunas tiene una alta correlación (0.94) con el acumulado de recuperados

Visualización de Datos con Matplotlib y Seaborn: Representa los hallazgos por cada país o de manera general a través de gráficos y/o visualizaciones. Debes incluir:

Histogramas gráficos de densidad y gráficos de densidad para entender la distribución de la incidencia de COVID-19 y las tasas de vacunación.

Gráficos de barras para comparar diferentes regiones.

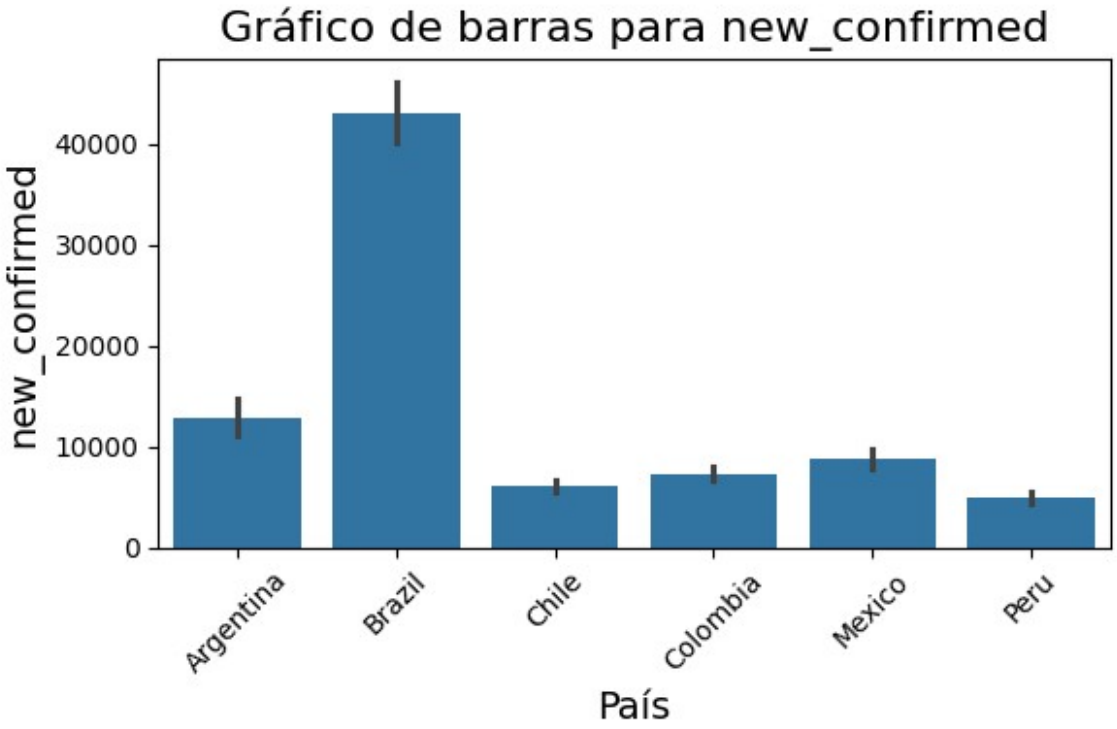
Mapas de calor para identificar correlaciones entre diferentes variables.

Gráficos de dispersión para explorar posibles relaciones entre las variables.

Análisis de casos y cuerpo médico por paises

```
# Diagramas de barras por país

# Selecciono las columnas que voy a incluir en los gráficos
columnas_seleccionadas = ['new_confirmed', 'new_deceased',
'population', 'population_density']

columnas_a_graficar = [col for col in df_limpio.columns if col in
columnas_seleccionadas]

# Recorro las columnas a graficar
for columna in columnas_a_graficar:
    plt.figure(figsize=(6, 4))
    sns.barplot(x='country_name', y=columna, data=df_limpio)
    plt.title(f'Gráfico de barras para {columna}', fontsize=16)
```

```
    plt.xticks(rotation=45)
    plt.xlabel('País', fontsize=14)
    plt.ylabel(columna, fontsize=14)
    plt.tight_layout()
    plt.show()
```



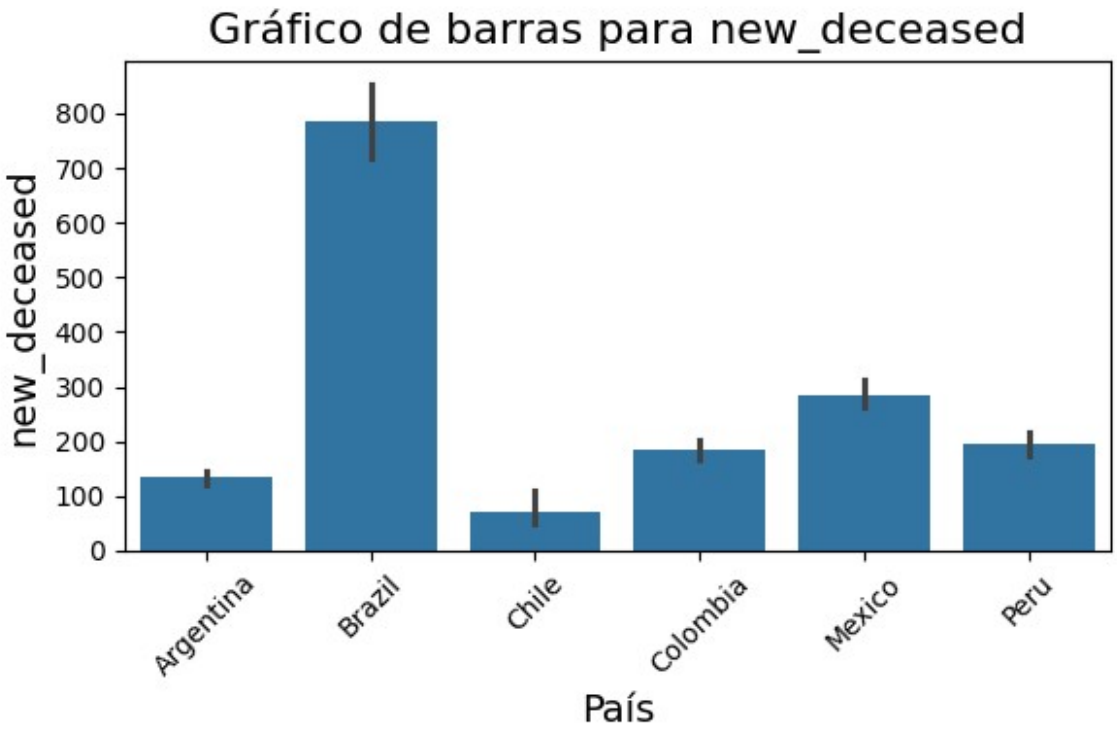


Gráfico de barras para new_deceased
new_deceased
800
700
600
500
400
300
200
100
0
Argentina
Brazil
Chile
Colombia
Mexico
Peru
País

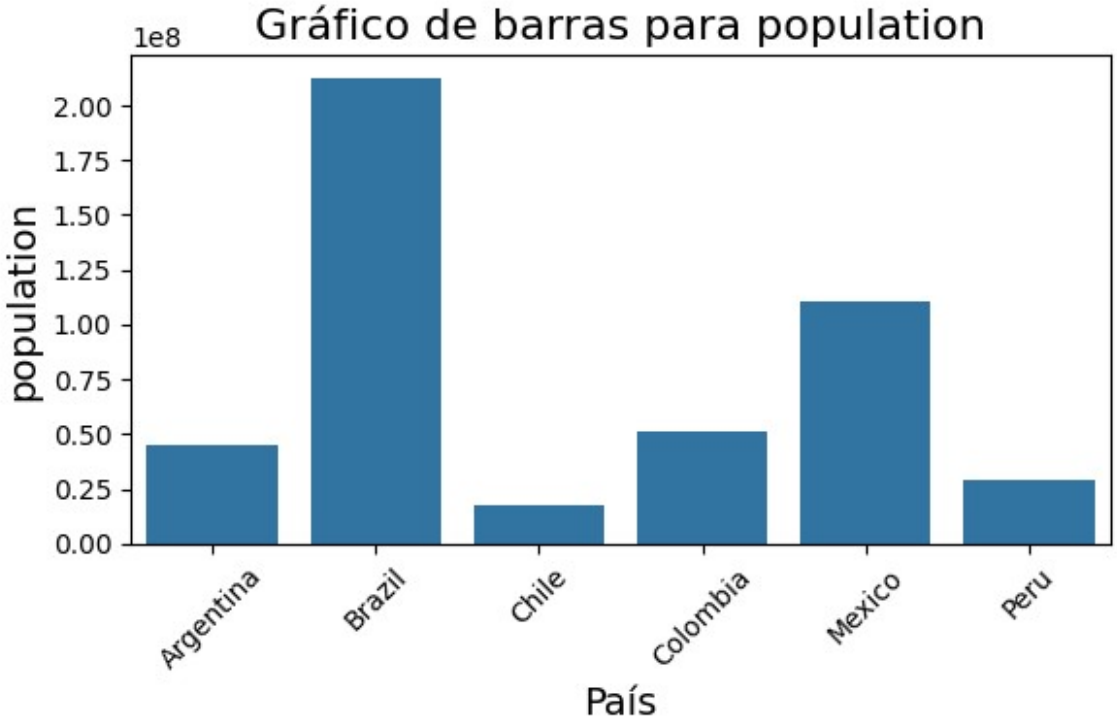
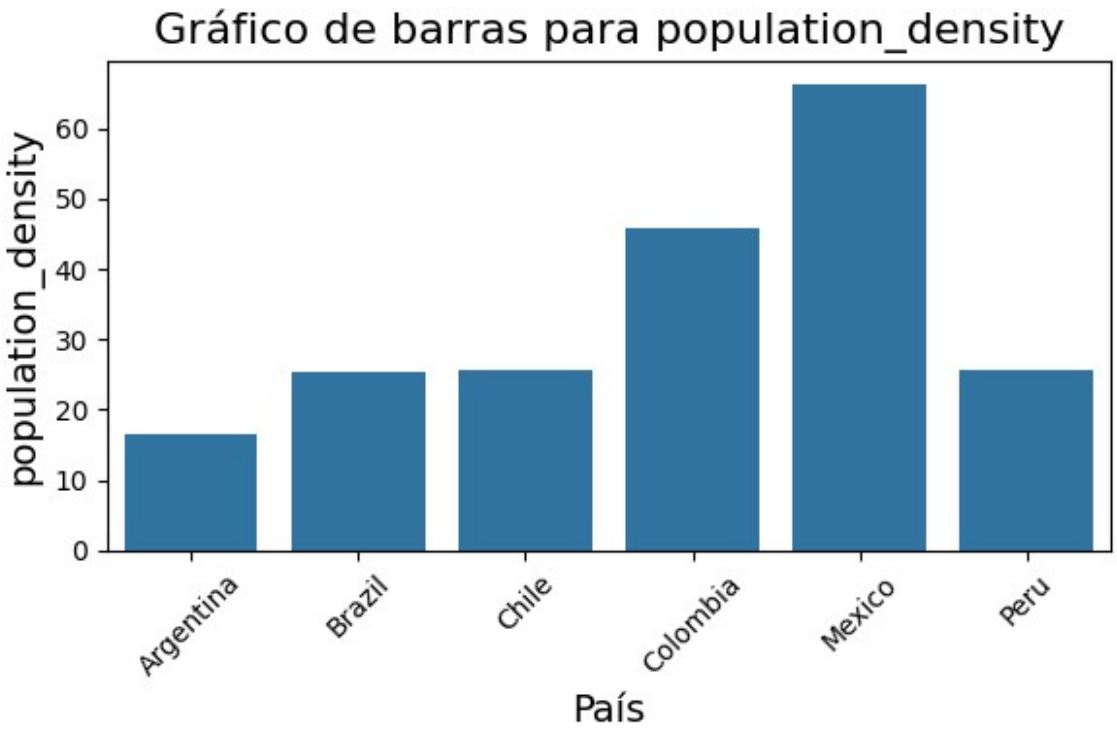
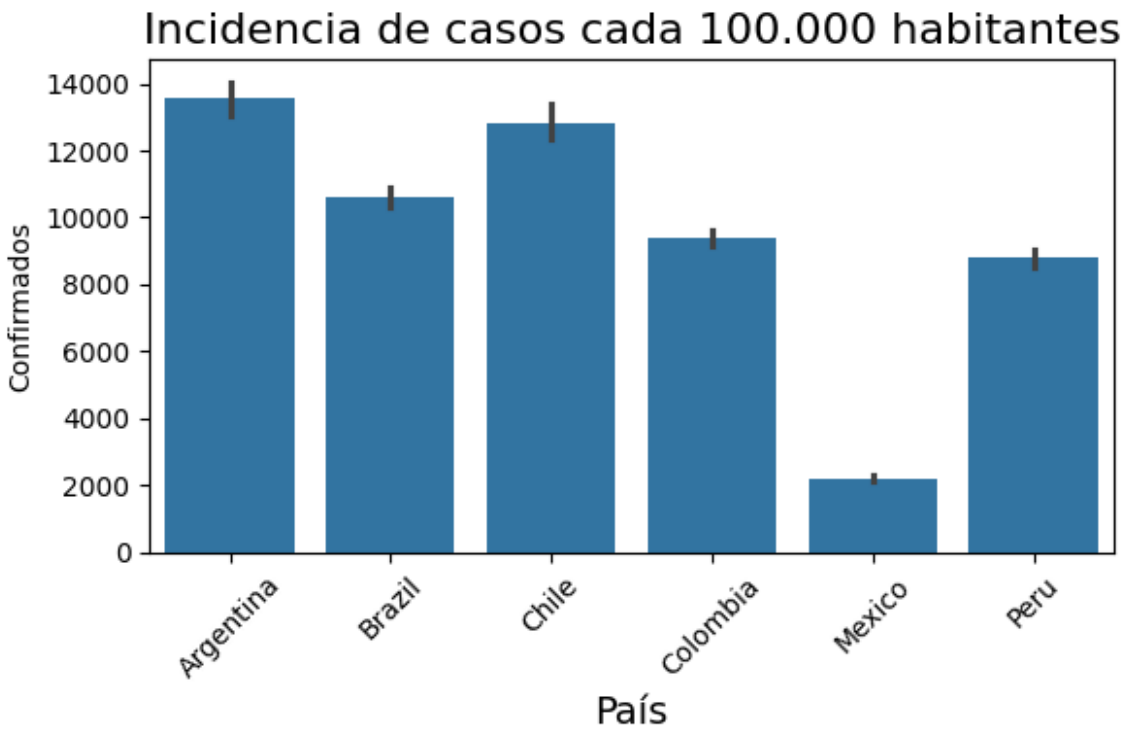


Gráfico de barras para population
1e8
population
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Argentina
Brazil
Chile
Colombia
Mexico
Peru
País



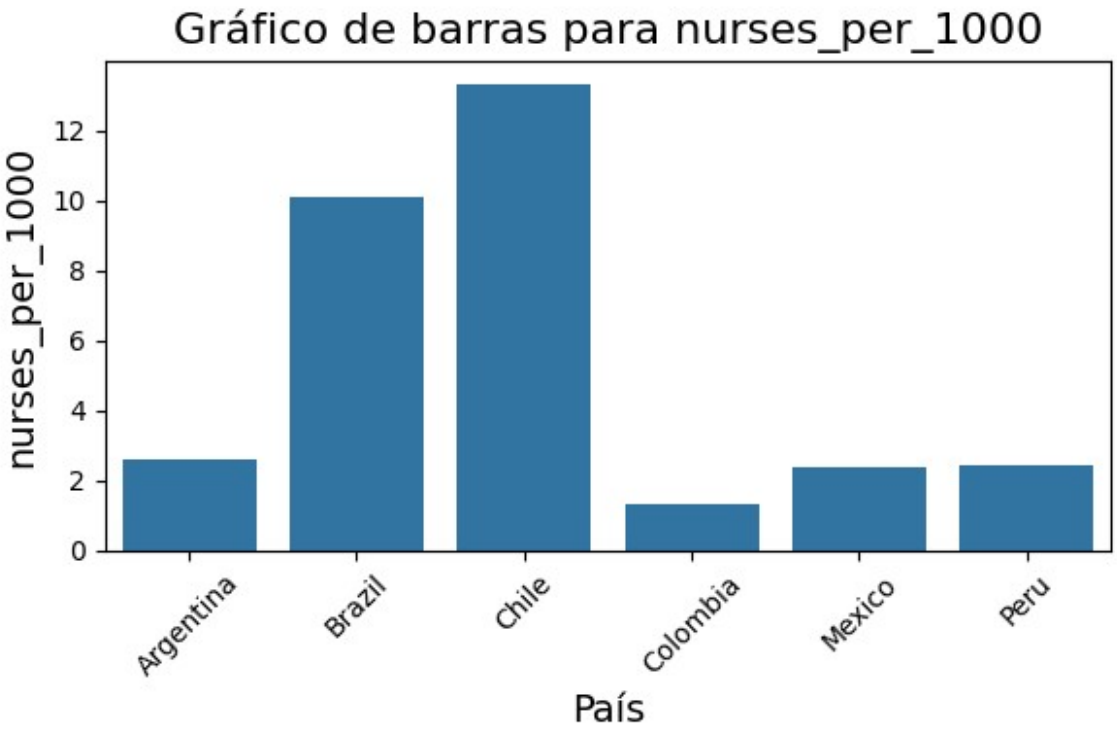
Dada la diferencia considerable en la población entre paises, no se puede establecer a partir de los gráficos, una comparativa entre los mismos. A continuación calculo una tasa de incidencia per cápita

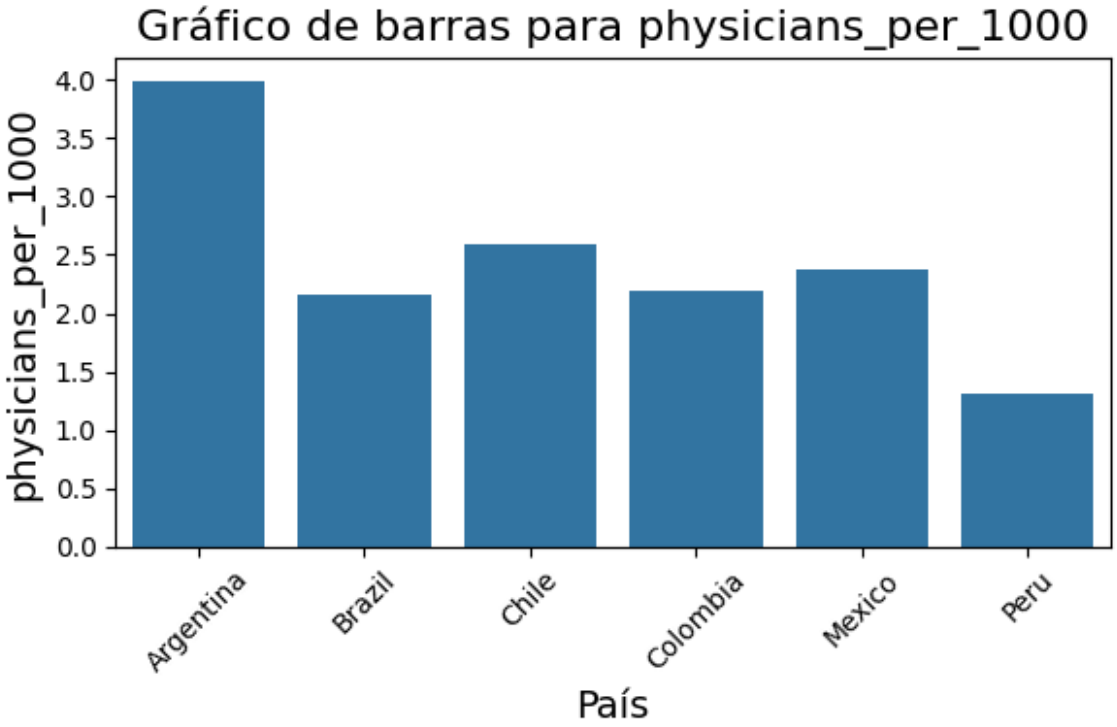
```
# Calculo la proporción de casos confirmados por cada 100,000
habitantes
incidencia_confirmados = (df_limpio['cumulative_confirmed'] /
df_limpio['population']) * 100000

# Grafico con la nueva variable
plt.figure(figsize=(6, 4))
sns.barplot(x='country_name', y=incidencia_confirmados,
data=df_limpio)
plt.title(f'Incidencia de casos cada 100.000 habitantes', fontsize=16)
plt.xticks(rotation=45)
plt.xlabel('País', fontsize=14)
plt.ylabel('Confirmados', fontsize=10)
plt.tight_layout()
plt.show()
```



Se observa mayor incidencia de confirmados en Argentica y Chile cada 100.000 habitantes. Quedando Brasil en tercer lugar.

```
# Repito la operación para ver nos enfermeros y médicos de cada país
# Diagramas de barras por país

# Selecciono las columnas que voy a incluir en los gráficos
columnas_seleccionadas = ['nurses_per_1000', 'physicians_per_1000']

columnas_a_graficar = [col for col in df_limpio.columns if col in
columnas_seleccionadas]

# Recorro las columnas a graficar
for columna in columnas_a_graficar:
    plt.figure(figsize=(6, 4))
    sns.barplot(x='country_name', y=columna, data=df_limpio)
    plt.title(f'Gráfico de barras para {columna}', fontsize=16)
    plt.xticks(rotation=45)
    plt.xlabel('País', fontsize=14)
    plt.ylabel(columna, fontsize=14)
    plt.tight_layout()
    plt.show()
```

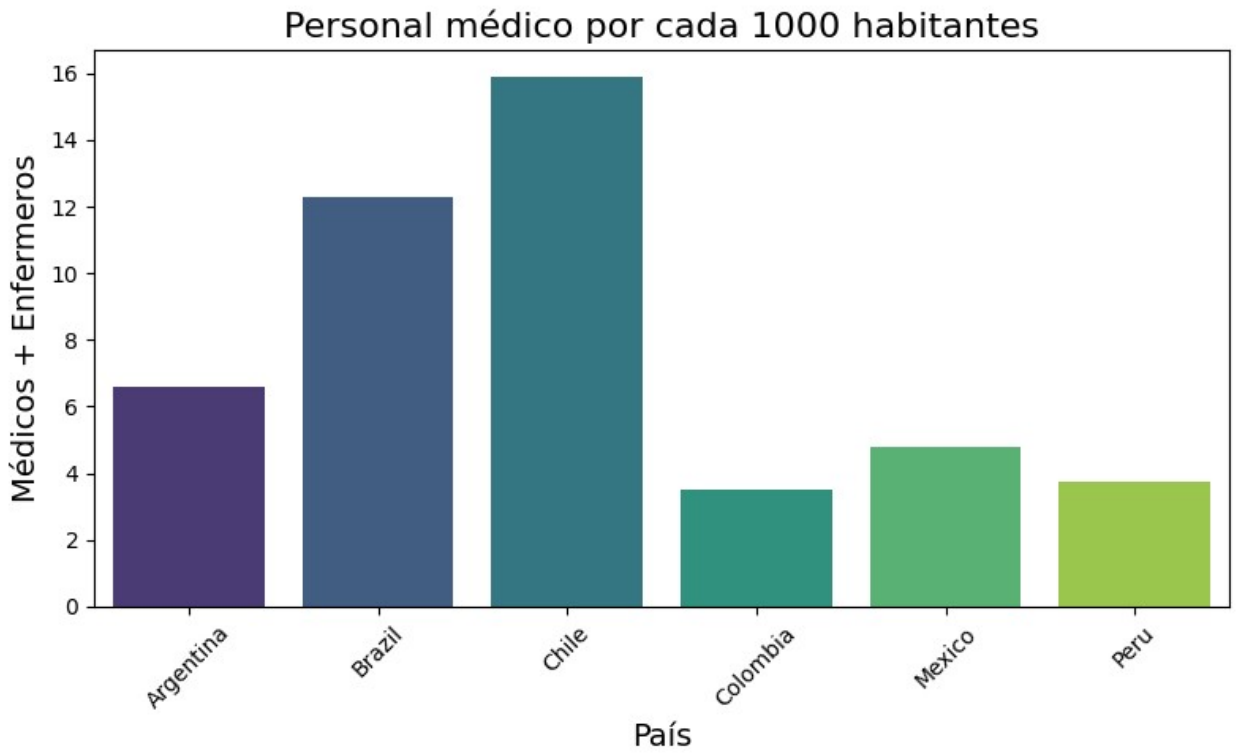




Mientras que Brasil y Chile cuentan con una alta cobertura de enfermeros, Argentina es más sólida en cantidad de Médicos.

```
# Calcular personal médico total por país
df_limpio['personal_salud_total'] = df_limpio['nurses_per_1000'] +
df_limpio['physicians_per_1000']

personal_por_pais = df_limpio.groupby('country_name')
['personal_salud_total'].mean().reset_index()
# uso mean() porque son "por cada 1000 habitantes" (no tiene sentido
sumarlos en el tiempo)

# Gráfico
plt.figure(figsize=(8, 5))
sns.barplot(x='country_name', y='personal_salud_total',
data=personal_por_pais, palette='viridis')
plt.title('Personal médico por cada 1000 habitantes', fontsize=16)
plt.xticks(rotation=45)
plt.xlabel('País', fontsize=14)
plt.ylabel('Médicos + Enfermeros', fontsize=14)
plt.tight_layout()
plt.show()
```

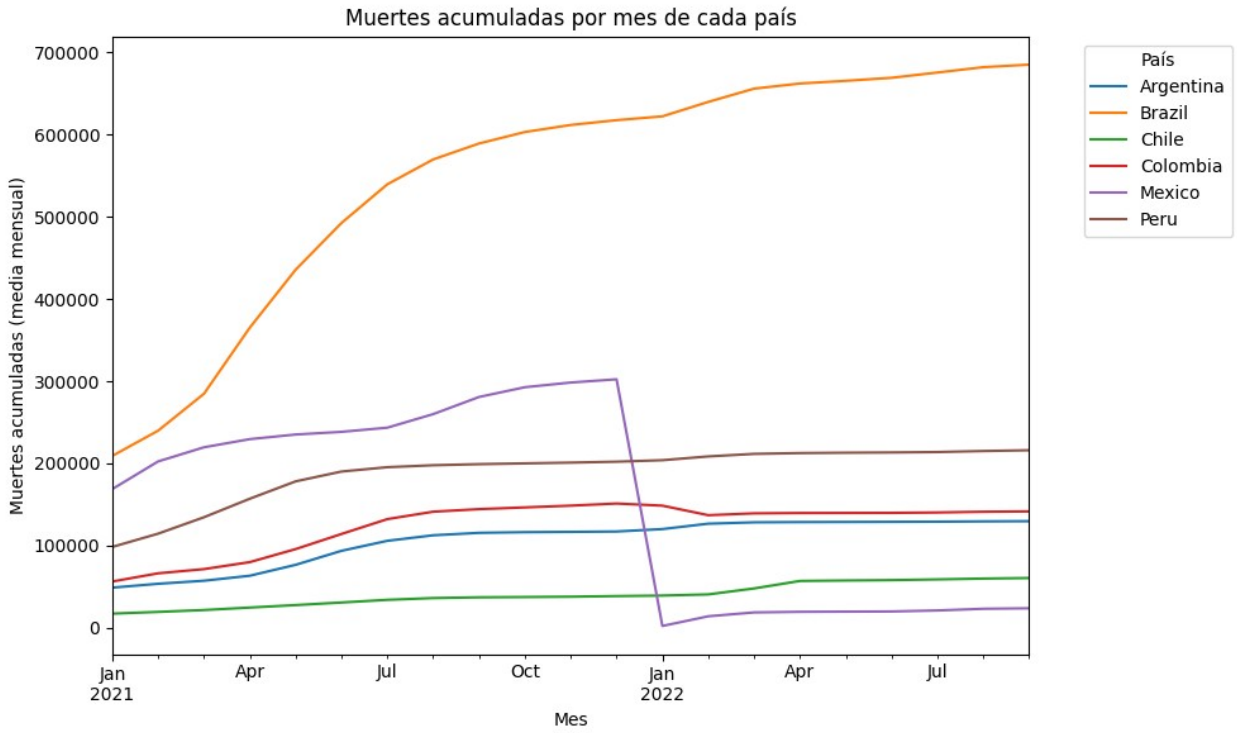
```
C:\Users\loren\AppData\Local\Temp\ipykernel_29652\3913663649.py:9:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='country_name', y='personal_salud_total',
data=personal_por_pais, palette='viridis')
```



Sin embargo en la sumatoria total del cuerpo médico, Chile alcanza los valores más altos y Argentina queda por detrás de Brasil.

A continuación analizo las muertes de cada país

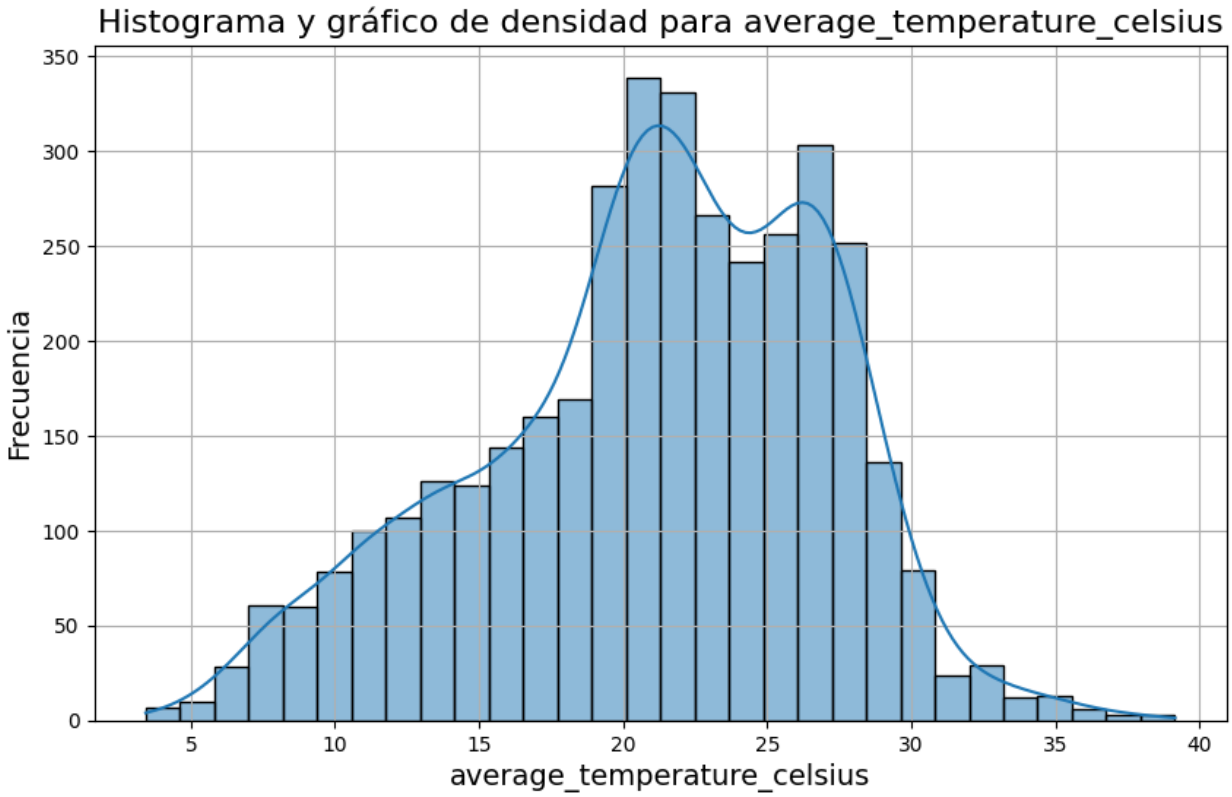
```
# Gráfica de la evolución mensual de muertes acumuladas por país
fig, ax = plt.subplots (figsize=(10, 6))

# Iterar sobre cada país y graficar la media mensual de muertes
acumuladas
for pais in  df_limpio['country_name'].unique():

    # Filtrar datos por país y hacer resample mensual (fin de mes)
    datos_pais = df_limpio[df_limpio['country_name'] ==
pais].resample('ME').mean(numeric_only=True)
    # Graficar la serie temporal de muertes acumuladas
    datos_pais['cumulative_deceased'].plot(ax=ax, label=pais)
# Configurar el título y las etiquetas
plt.title('Muertes acumuladas por mes de cada país')
plt.xlabel('Mes')
plt.ylabel('Muertes acumuladas (media mensual)')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', title='País')
plt.tight_layout()
plt.show()
```

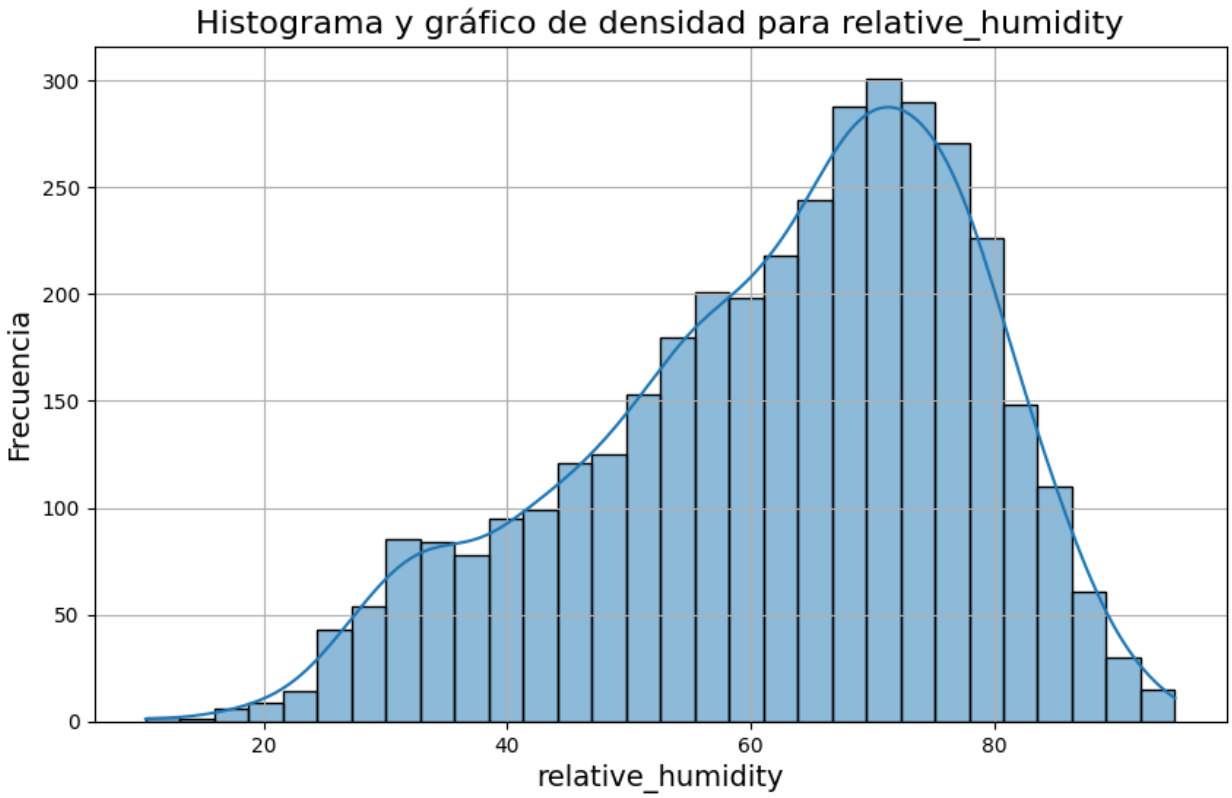


Brasil tiene una alta curva de muertes mientras que en el resto de los paises se logra controlar. México muetra un comportamiento llamativo en Junio del 2022. Podría deberse a errores de información. Tal vez un cambio en la dimensión de registro de los valores.

Analisis de variables ambientales

```
# Lista de columnas para analizar la incidencia de variables
ambientales y de salud
columnas_analizar =
['average_temperature_celsius','relative_humidity']
# Recorro las columnas a graficar
for columna in columnas_analizar:
    plt.figure(figsize=(10, 6))
    sns.histplot(df_limpio[columna], kde=True, bins=30)
    plt.title(f'Histograma y gráfico de densidad para {columna}',
fontsize=16)
    plt.xlabel(columna, fontsize=14)
    plt.ylabel('Frecuencia', fontsize=14)
    plt.grid(True)
    plt.show()
```

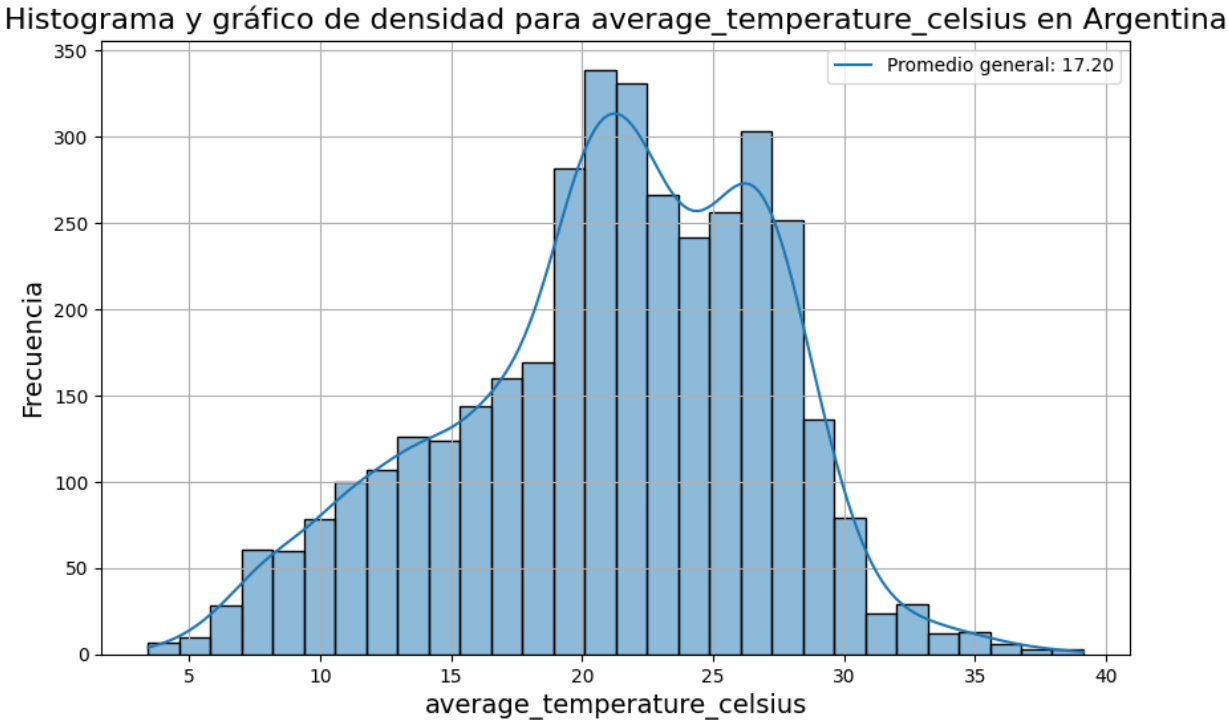

Histograma y gráfico de densidad para average_temperature_celsius
Frecuencia
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40
average_temperature_celsius

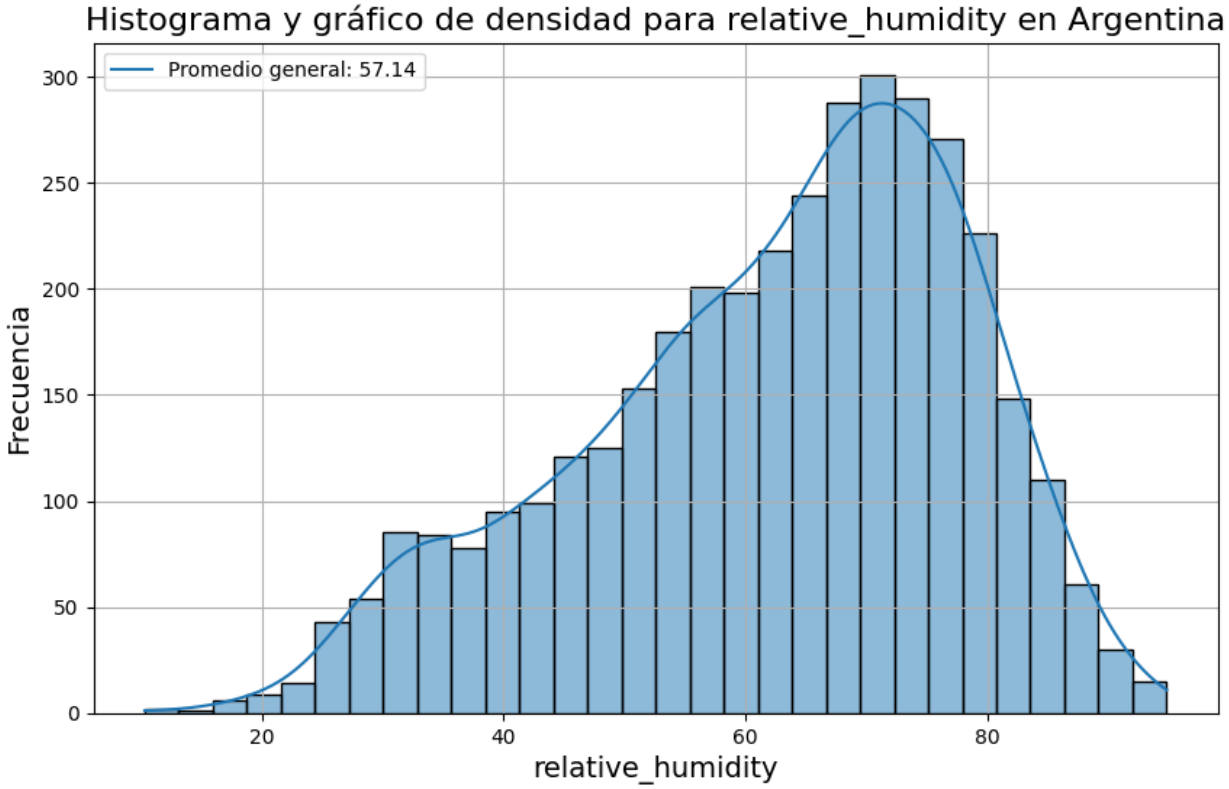

Histograma y gráfico de densidad para relative_humidity
Frecuencia
0
50
100
150
200
250
300
20
40
60
80
relative_humidity

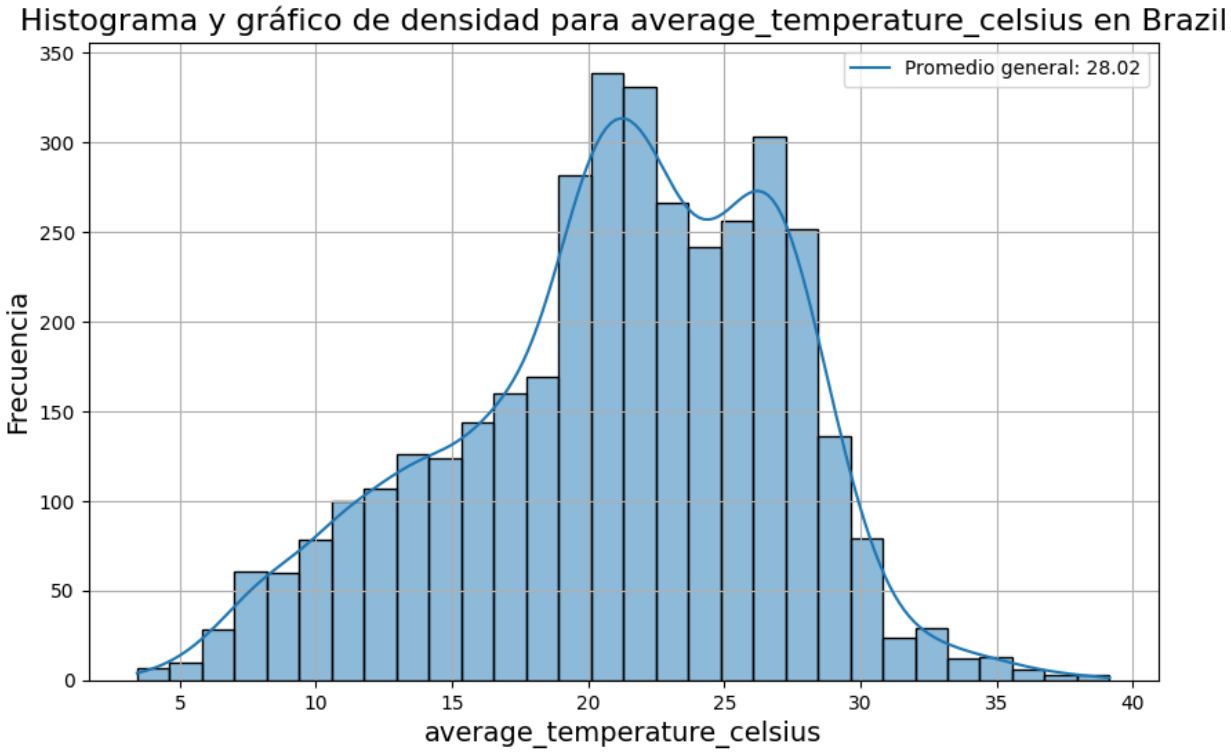
La mayoría de los casos parecen estar agrupados en temperaturas que oscilan entre los 20 y 30 grados con una humedad relativa entre 70 y 80%. A priori creería que es la temperatura y humedad habitual en estos paises.

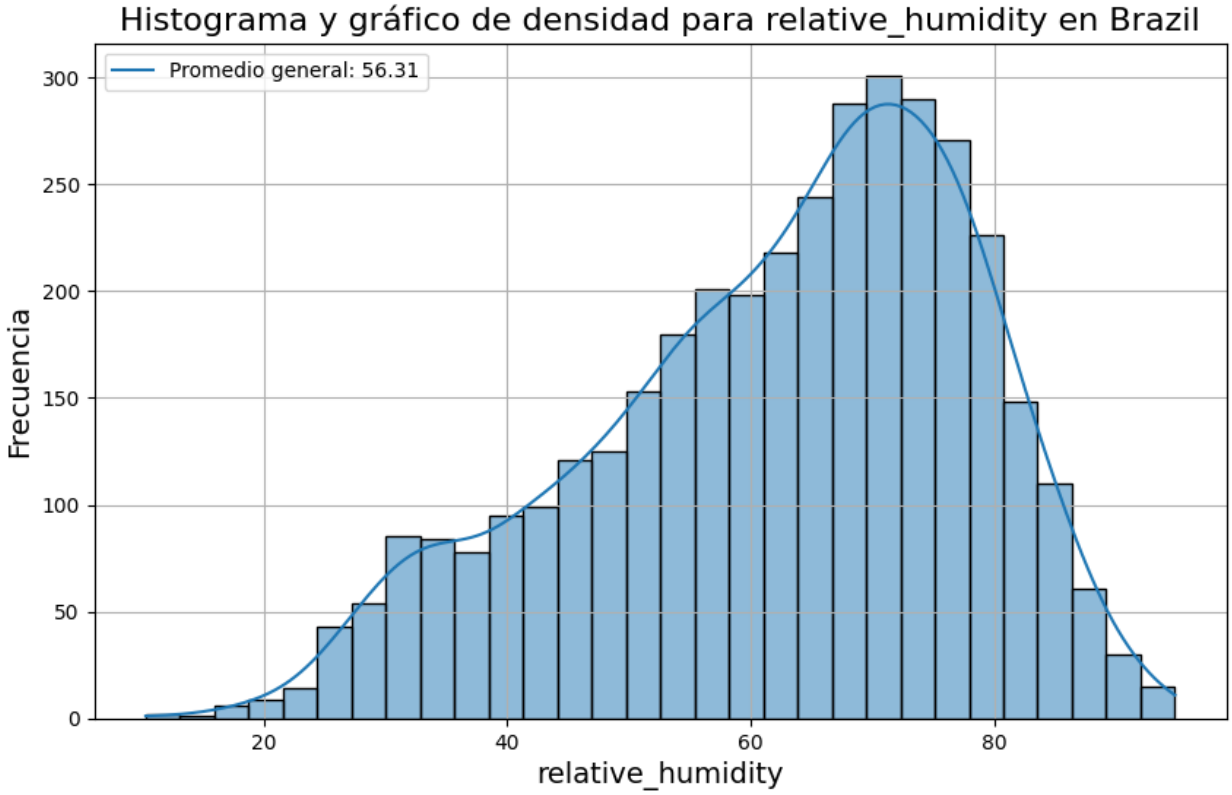
Reviso las condiciones ambientales por pais

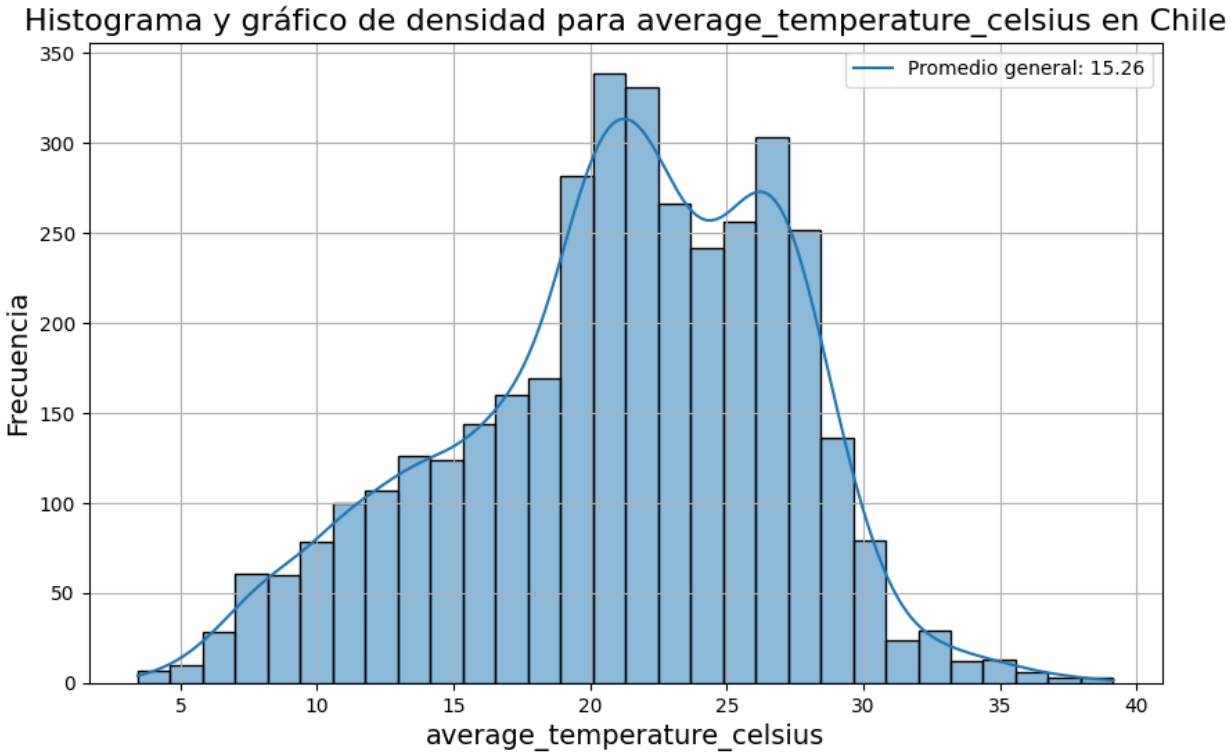
```
# Lista de columnas para analizar la incidencia de variables
ambientales y de salud
columnas_analizar =
['average_temperature_celsius','relative_humidity']


# Iterar por país
for country in df_limpio['country_name'].unique():
    df_country = df_limpio[df_limpio['country_name'] == country]
    promedio_temperatura =
df_country['average_temperature_celsius'].mean()
    promedio_humedad = df_country['relative_humidity'].mean()
    # Recorro las columnas a graficar
    for columna in columnas_analizar:
        plt.figure(figsize=(10, 6))
        sns.histplot(df_limpio[columna], kde=True, bins=30)
        plt.title(f'Histograma y gráfico de densidad para {columna} en
{country}' , fontsize=16)
        plt.xlabel(columna, fontsize=14)
        plt.ylabel('Frecuencia', fontsize=14)
        plt.legend([f'Promedio general: {promedio_temperatura:.2f}' if
columna == 'average_temperature_celsius' else f'Promedio general:
{promedio_humedad:.2f}'])
        plt.grid(True)
        plt.show()
```

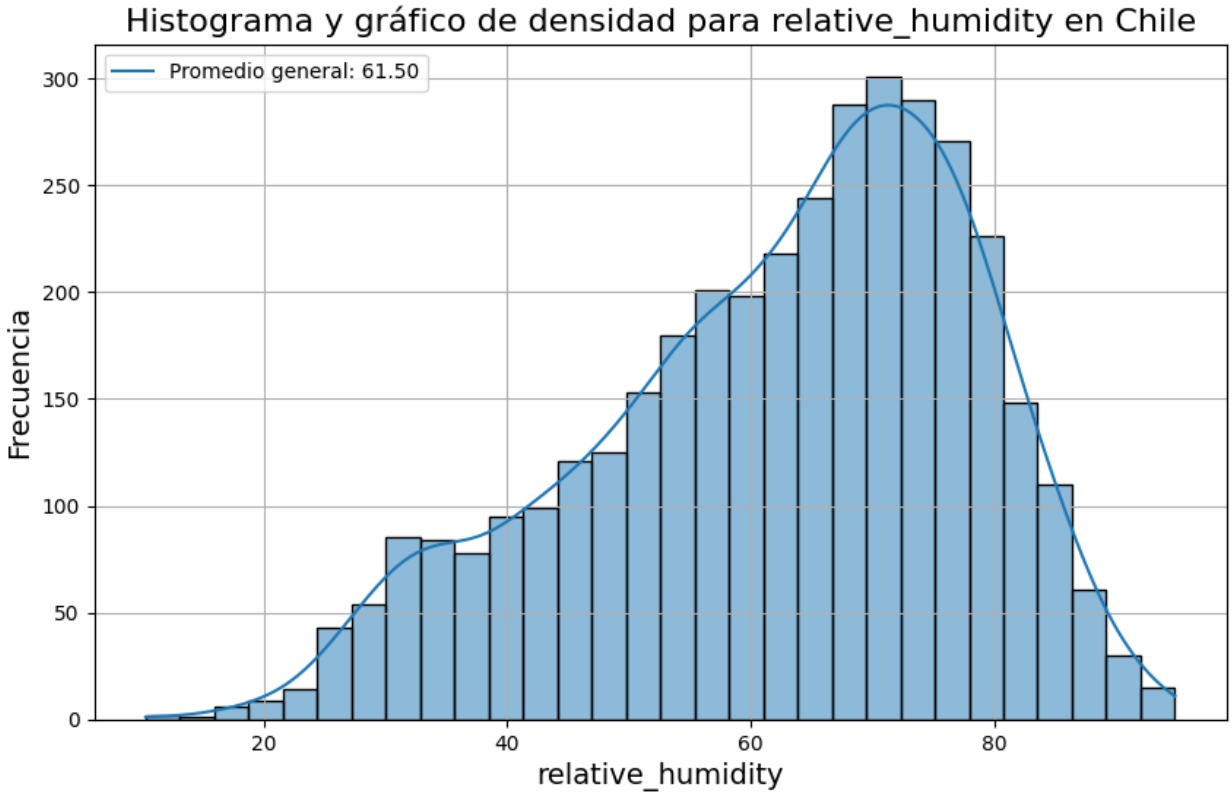

Histograma y gráfico de densidad para average_temperature_celsius en Argentina
Promedio general: 17.20
Frecuencia
average_temperature_celsius
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40

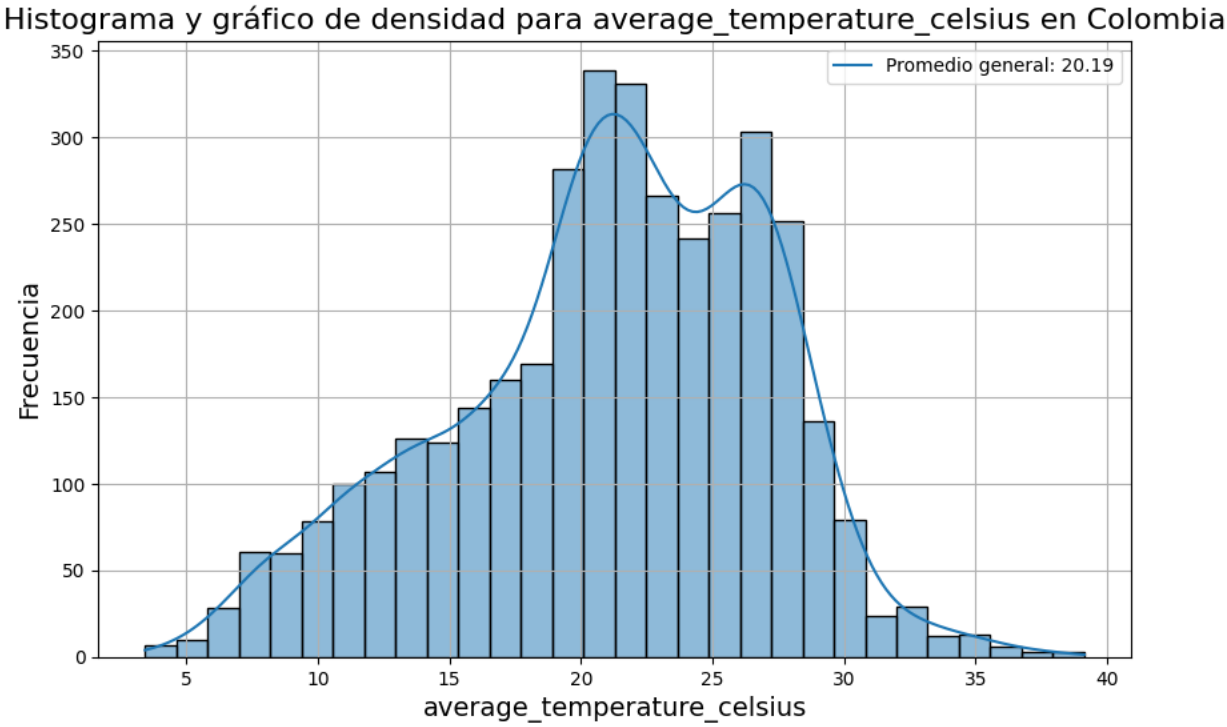

Histograma y gráfico de densidad para relative_humidity en Argentina
Promedio general: 57.14
Frecuencia
relative_humidity
0
50
100
150
200
250
300
20
40
60
80

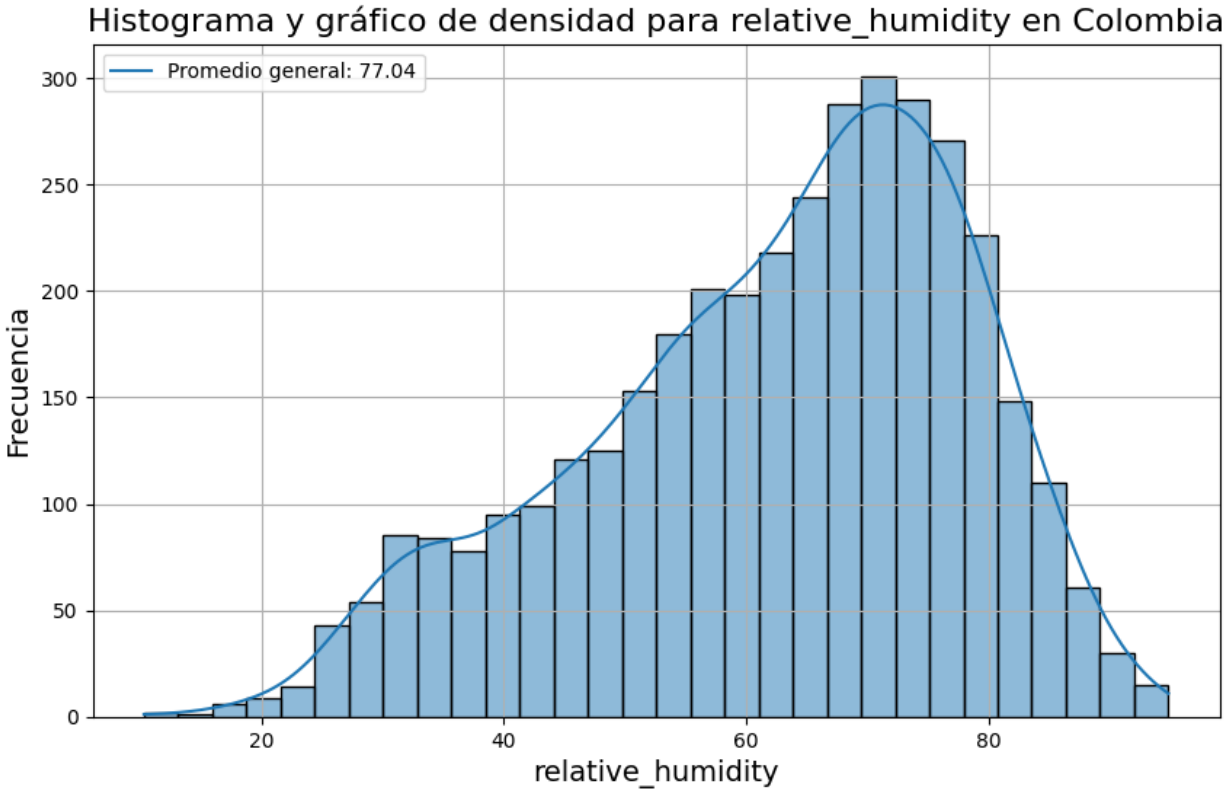

Histograma y gráfico de densidad para average_temperature_celsius en Brazil
Promedio general: 28.02
Frecuencia
average_temperature_celsius
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40

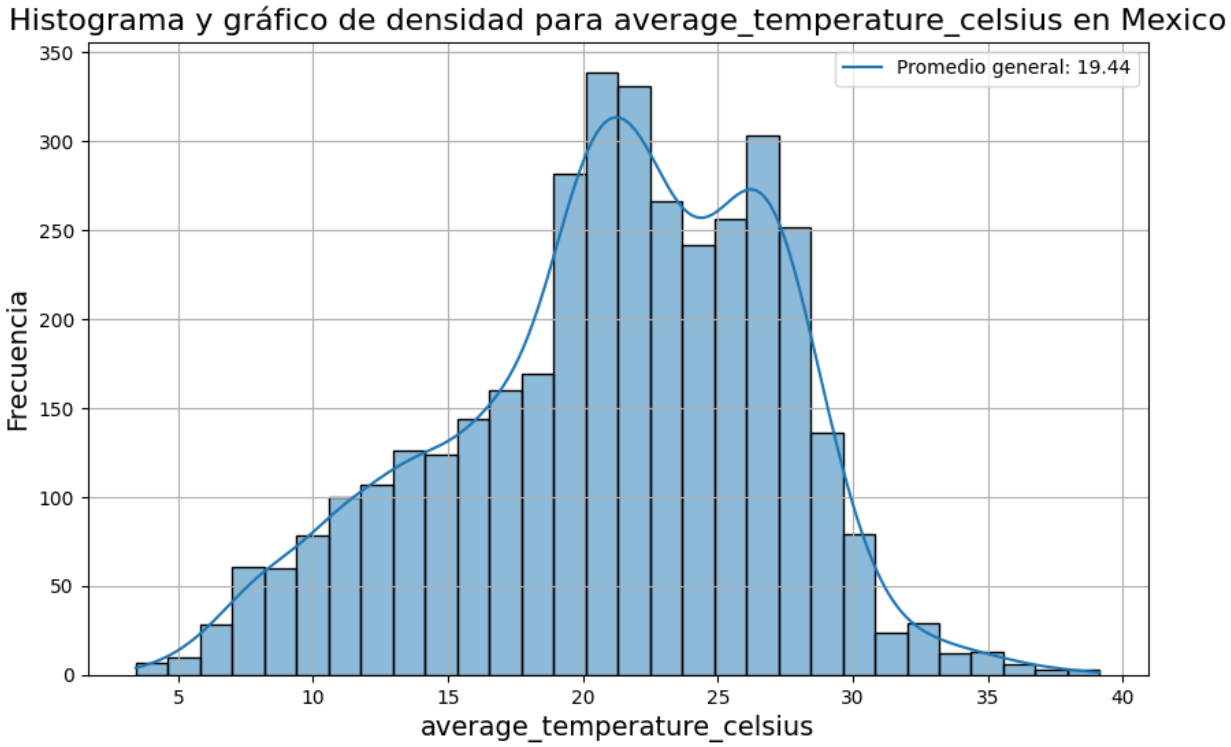

Histograma y gráfico de densidad para relative_humidity en Brazil
Promedio general: 56.31
Frecuencia
relative_humidity
0
50
100
150
200
250
300
20
40
60
80

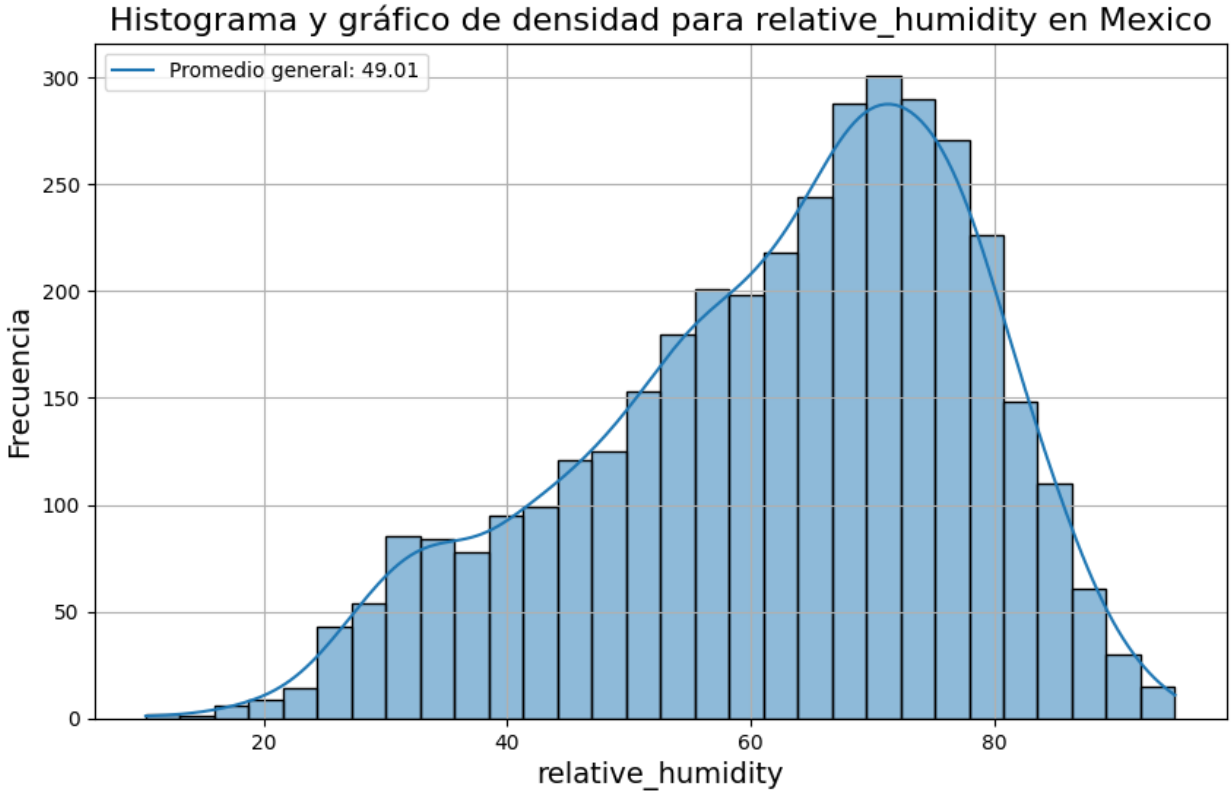

Histograma y gráfico de densidad para average_temperature_celsius en Chile
Promedio general: 15.26
Frecuencia
average_temperature_celsius
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40

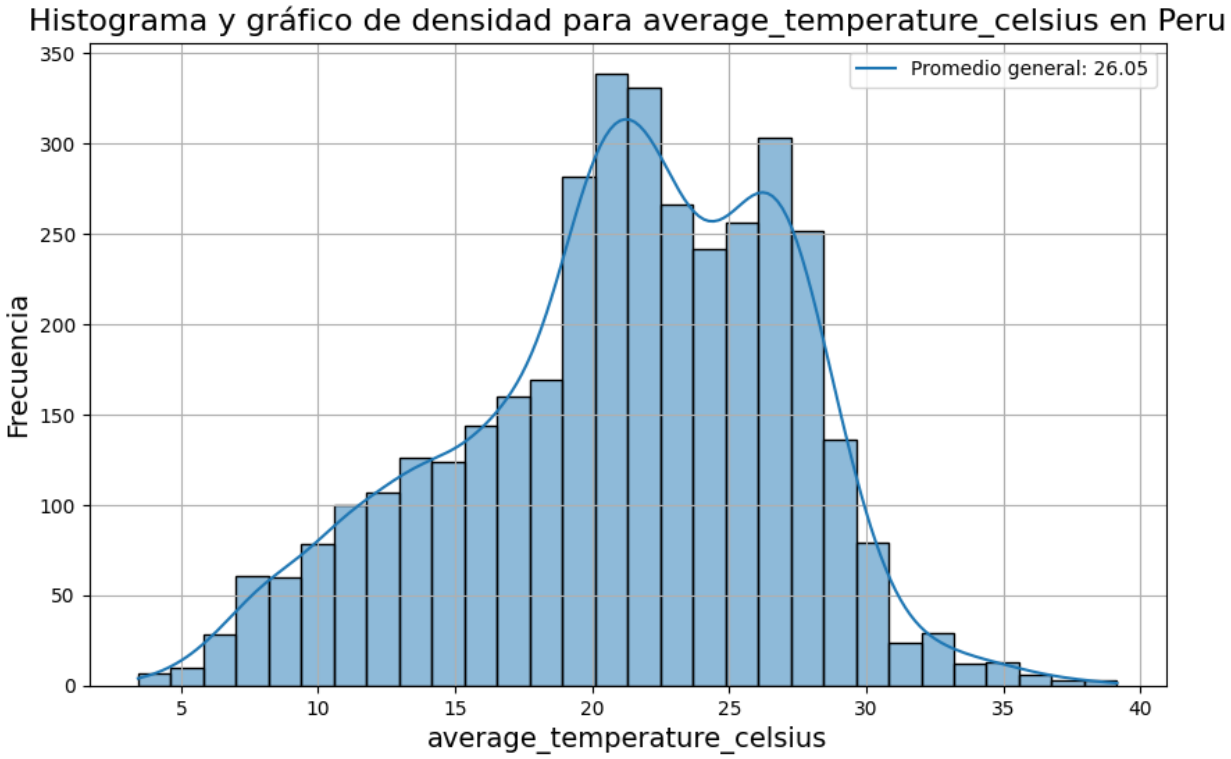

Histograma y gráfico de densidad para relative_humidity en Chile
Promedio general: 61.50
Frecuencia
relative_humidity
0
50
100
150
200
250
300
20
40
60
80

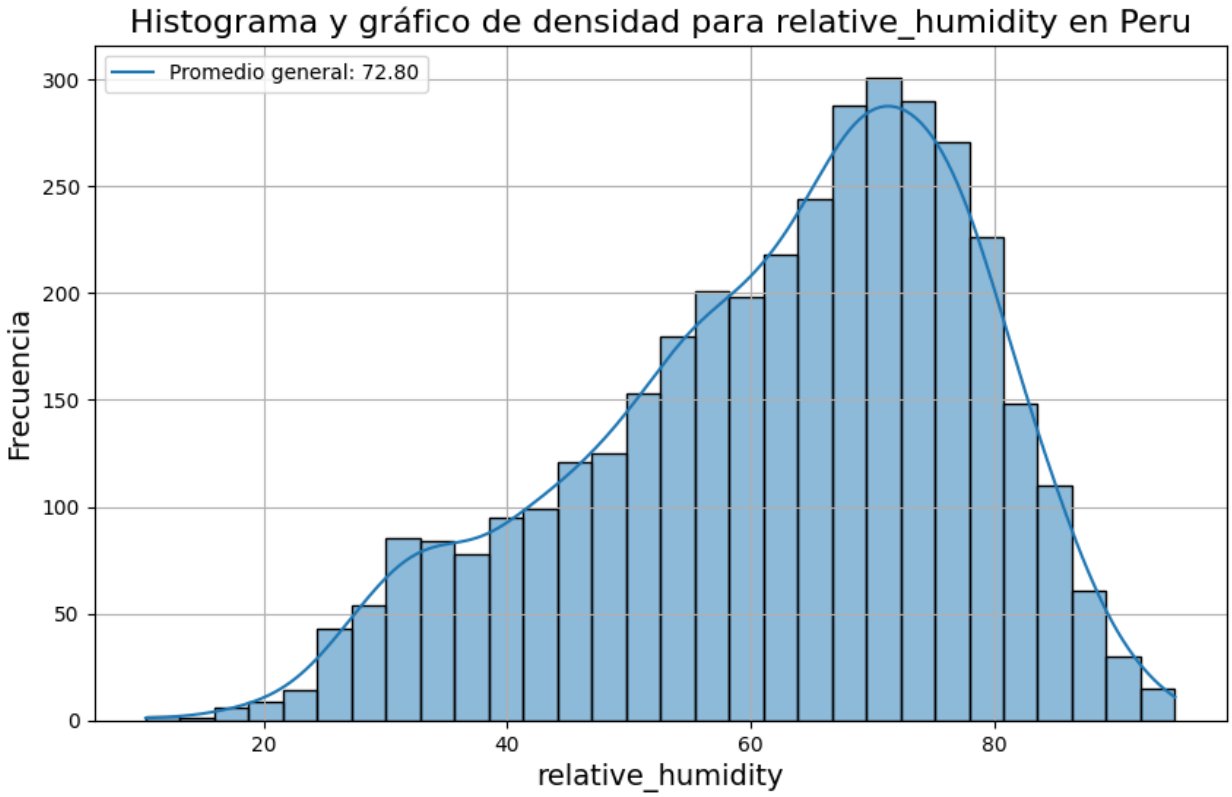

Histograma y gráfico de densidad para average_temperature_celsius en Colombia
Promedio general: 20.19
Frecuencia
average_temperature_celsius
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40


Histograma y gráfico de densidad para relative_humidity en Colombia
Promedio general: 77.04
Frecuencia
relative_humidity
0
50
100
150
200
250
300
20
40
60
80


Histograma y gráfico de densidad para average_temperature_celsius en Mexico
Promedio general: 19.44
Frecuencia
average_temperature_celsius
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40

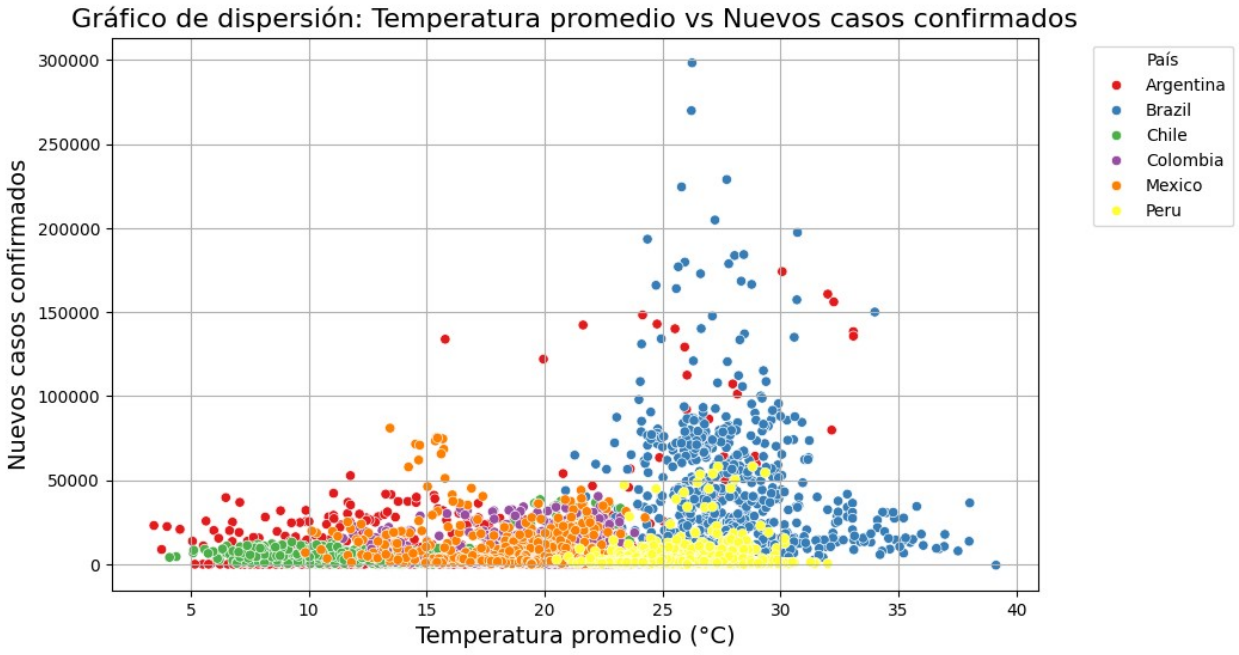

Histograma y gráfico de densidad para relative_humidity en Mexico
Promedio general: 49.01
Frecuencia
relative_humidity
0
50
100
150
200
250
300
20
40
60
80


Histograma y gráfico de densidad para average_temperature_celsius en Peru
Promedio general: 26.05
Frecuencia
average_temperature_celsius
0
50
100
150
200
250
300
350
5
10
15
20
25
30
35
40


Histograma y gráfico de densidad para relative_humidity en Peru
Promedio general: 72.80
Frecuencia
relative_humidity
0
50
100
150
200
250
300
20
40
60
80

Los casos en cada país aparecen reflejados entre los mismo valores de la celda anterior, idependientemente de sus propios promedios. Confirmando el rango de temperaturas y humedad de mayor propagación de casos.

```
# Graficos de dispersión para analizar la relación entre el promedio de temperatura de cada país identificados por colores
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_limpio, x='average_temperature_celsius', y='new_confirmed', hue='country_name', palette='Set1')
plt.title('Gráfico de dispersión: Temperatura promedio vs Nuevos casos confirmados', fontsize=16)
plt.xlabel('Temperatura promedio (°C)', fontsize=14)
plt.ylabel('Nuevos casos confirmados', fontsize=14)
plt.legend(title='País', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
```



Avance 3

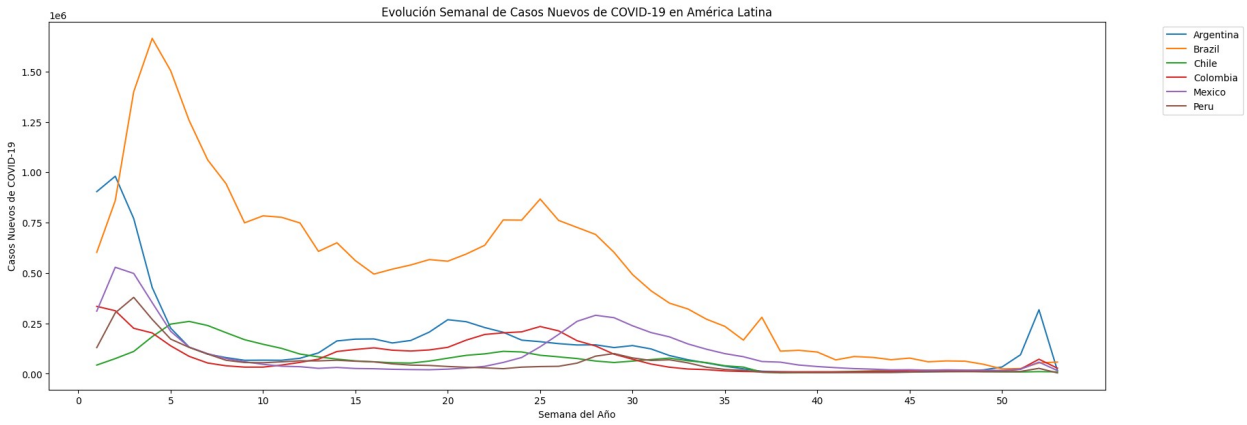
EDA con Numpy y Pandas

Realiza un análisis exploratorio detallado utilizando técnicas avanzadas de Pandas y Numpy, centrándote en el análisis de series temporales para comprender la evolución de elementos específicos del conjunto de datos. Para ello, puedes empezar por identificar tendencias, estacionalidad y patrones temporales relevantes, análisis de autocorrelación y descomposición de series temporales.

Análisis temporal de casos y vacunas

Analizo la evolución de casos por semana

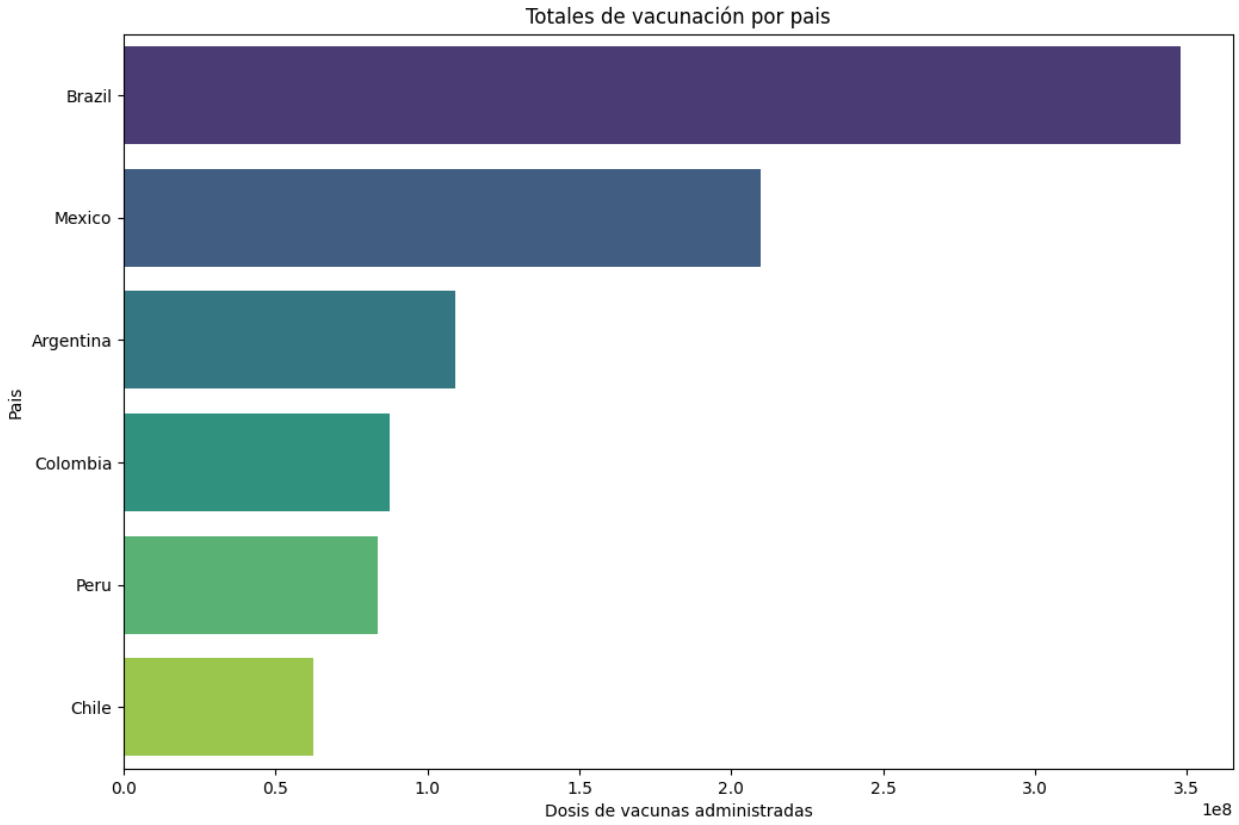
```
# Analizo cómo evolucionan los casos de COVID-19 en América Latina a
lo largo del tiempo por semana
df_limpio['week'] = df_limpio.index.isocalendar().week

casos_por_semana = df_limpio.groupby(['country_name',
                                      'week'])
['new_confirmed'].sum().reset_index()
plt.figure(figsize=(20, 7))
for country in casos_por_semana['country_name'].unique():
    casos_por_semana_por_pais = casos_por_semana [casos_por_semana
['country_name'] == country]
    plt.plot(casos_por_semana_por_pais['week'],
casos_por_semana_por_pais['new_confirmed'], label=country)

plt.xlabel('Semana del Año')
plt.ylabel('Casos Nuevos de COVID-19')
plt.title('Evolución Semanal de Casos Nuevos de COVID-19 en América
Latina')

# Definio ticks de 5 en 5 para aumentar la legibilidad
semanas_min = casos_por_semana['week'].min()
semanas_max = casos_por_semana['week'].max()
plt.xticks(np.arange(semanas_min-1, semanas_max, 5))

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



Mientras a mayoría de los paises muestran olas hacia abajo, Brasil obtuvo un pico alrededor de la semana 5. Argentina presenta un rebrote luego de la semana 50.

Evaluación de las dosis administradas de vacunación

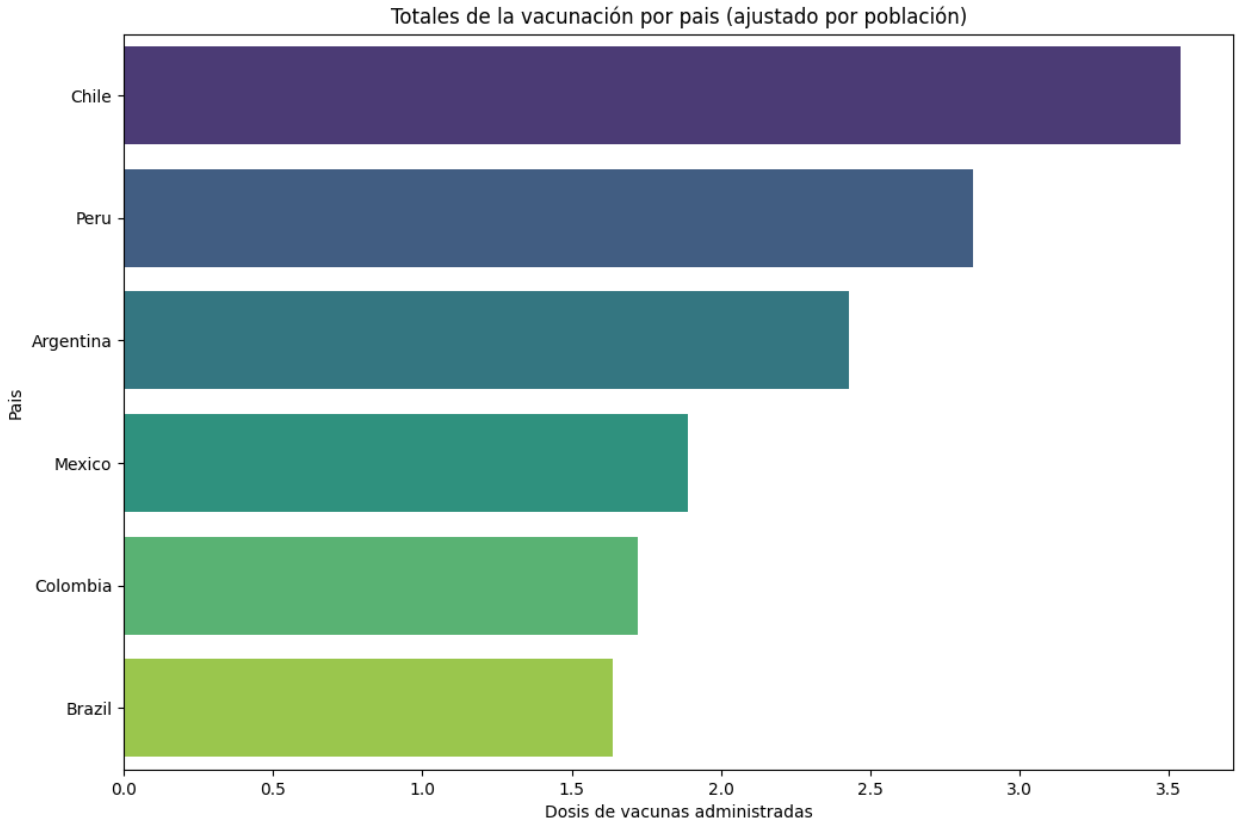
```
# Valor máximo de dosis de vacunas administradas por país
datos_de_vacunacion = df_limpio.groupby("country_name")
['cumulative_vaccine_doses_administered'].max().sort_values(ascending=
False)
plt.figure(figsize=(12, 8))
sns.barplot(x=datos_de_vacunacion.values, y=datos_de_vacunacion.index,
hue=datos_de_vacunacion.index, palette='viridis')

plt.xlabel('Dosis de vacunas administradas')
plt.ylabel('Pais')
plt.title('Totales de vacunación por pais')
plt.show()
```



Nuevamente, al ser disimil la cantidad de habitantes, es necesario realizar una exploración basada en tasas per cápita.

```
#Calculo la cantidad de dosis de vacuna por habitante hay registradas
estrategia_de_vacunacion = df_limpio.groupby('country_name')
['cumulative_vaccine_doses_administered'].max()/df_limpio.groupby('cou
ntry_name') ['population'].mean()
estrategia_de_vacunacion=estrategia_de_vacunacion.sort_values
(ascending=False)
```

```
plt.figure(figsize=(12, 8))
sns.barplot(x=estrategia_de_vacunacion.values,
y=estrategia_de_vacunacion.index, hue=estrategia_de_vacunacion.index,
palette='viridis')

plt.xlabel('Dosis de vacunas administradas')
plt.ylabel('Pais')
plt.title('Totales de la vacunación por pais (ajustado por
población)')
plt.show()
```



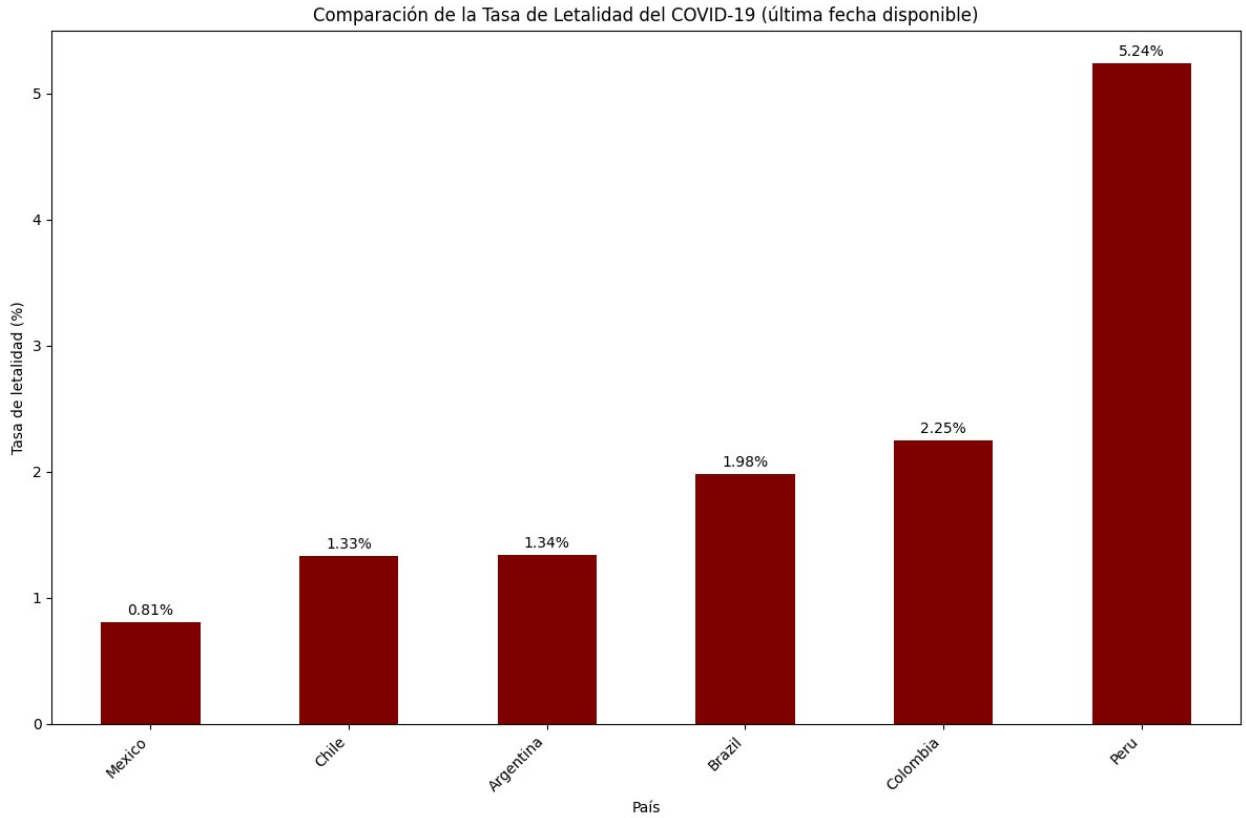
Chile y Perú dominan en la cantidad de vacunados según su población, mientras que Brasil queda en inferioridad.

Analizo que ocurre con la tasa de letalidad en cada país

```
# Calculo la tasa de letalidad como el cociente entre los casos
acumulados y los fallecidos acumulados
df_limpio['tasa_casos_letalidad'] = df_limpio['cumulative_deceased'] /
df_limpio['cumulative_confirmed']

tasa_letalidad = (
    df_limpio.groupby('country_name')
      .apply(lambda g: g.loc[g.index.max(), 'tasa_casos_letalidad'])
```

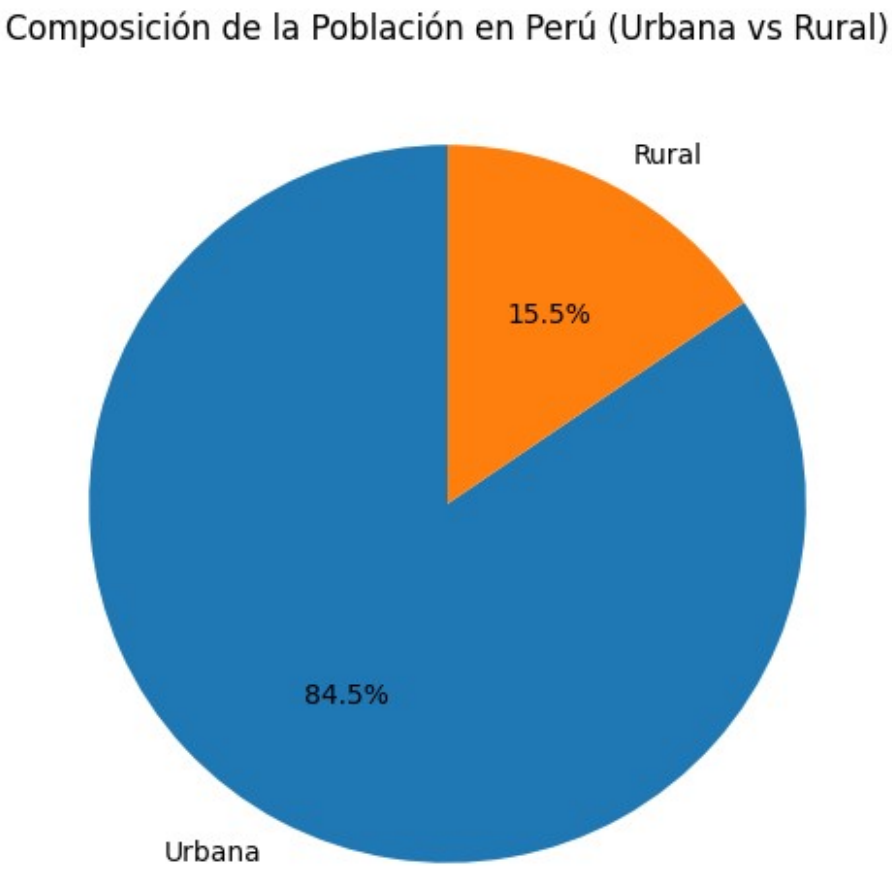
```
      .sort_values()
) * 100

ax = tasa_letalidad.plot(kind='bar', color='maroon', figsize=(12, 8))
ax.set_xlabel('País')
ax.set_ylabel('Tasa de letalidad (%)')
ax.set_title('Comparación de la Tasa de Letalidad del COVID-19 (última
fecha disponible)')

# Etiquetas con el valor exacto
for container in ax.containers:
    ax.bar_label(container, fmt='%.2f%%', padding=3)

plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
C:\Users\loren\AppData\Local\Temp\ipykernel_29652\2554126437.py:6:
FutureWarning: DataFrameGroupBy.apply operated on the grouping
columns. This behavior is deprecated, and in a future version of
pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or
explicitly select the grouping columns after groupby to silence this
warning.
  .apply(lambda g: g.loc[g.index.max(), 'tasa_casos_letalidad'])
```



Pese a que Perú tiene una alta dosis de vacunas suministradas, la tasa de letalidad en este país es muy alta. Aún así la cantidad de casos fue disminuyendo a lo largo de las semanas. Según los gráficos desarrollados anteriormente, Perú no muestra una alta densidad poblacional. Sin embargo su cobertura total del cuerpo médico es de las más bajas. Aún así la cantidad de casos fue disminuyendo a lo largo de las semanas.

```
# Composición de la población urbana y rural en Perú
poblacion = {
    'Urbana': df_limpio['population_urban'].mean(),
    'Rural': df_limpio['population_rural'].mean()
}

# Gráfico de torta
plt.figure(figsize=(6,6))
plt.pie(poblacion.values(),
        labels=poblacion.keys(),
        autopct='%1.1f%%',  # mostrar porcentajes
        startangle=90,      # comienza desde arriba
        colors=['#1f77b4', '#ff7f0e'])
plt.title('Composición de la Población en Perú (Urbana vs Rural)')
plt.show()
```

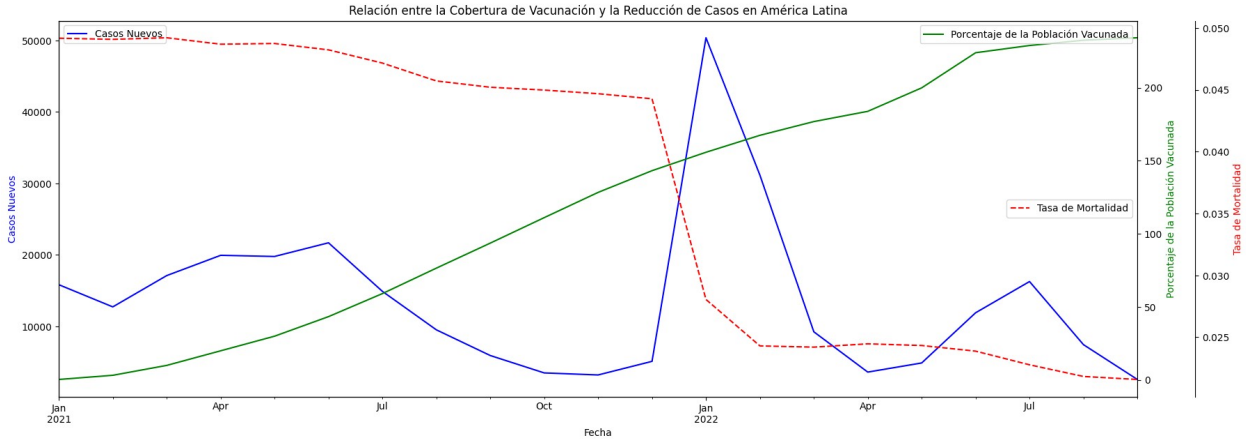


La mayoría de la población de Perú es urbana, no podría inferirse que haya problemas de administración de dosis o inconvenientes de traslado de los pacientes hacia los centros de atención

Empieza a ser necesaria buscar una relación entre las vacunas aplicadas y los resultados en cuanto a disminución de casos y muertes. Para establecer una medida pareja para todos los paises, trabajaré con tasas respecto de la publación total.

```
# Grafico con 3 ejes para analizar la relación entre la vacunación,
los casos nuevos y la tasa de mortalidad. Cada uno con sus escalas
plt.figure(figsize=(20, 7))
ax1 = plt.gca()
ax2 = ax1.twinx()
ax3 = ax1.twinx()

#creo porcentajes de vacunacion y tasa de mortalidad
df_limpio['percentage_vaccinated'] =
(df_limpio['cumulative_vaccine_doses_administered'] /
df_limpio['population'] * 100)
df_limpio['mortality_rate'] = df_limpio['cumulative_deceased']
```

```
/df_limpio['cumulative_confirmed']

df_limpio.resample('ME').mean(numeric_only=True)
['new_confirmed'].plot(ax=ax1, color='blue', label='Casos Nuevos')
df_limpio.resample('ME').mean(numeric_only=True)
['percentage_vaccinated'].plot(ax=ax2, color='green',
label='Porcentaje de la Población Vacunada')
df_limpio.resample('ME').mean(numeric_only=True)
['mortality_rate'].plot(ax=ax3, color='red', label='Tasa de
Mortalidad', linestyle='--')


ax1.set_xlabel('Fecha')
ax1.set_ylabel('Casos Nuevos', color='blue')
ax2.set_ylabel('Porcentaje de la Población Vacunada', color='green')
ax3.set_ylabel('Tasa de Mortalidad', color='red')
ax1.set_title('Relación entre la Cobertura de Vacunación y la
Reducción de Casos en América Latina')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
ax3.legend(loc='center right')
ax3.spines['right'].set_position(('outward', 60))

plt.show()
```

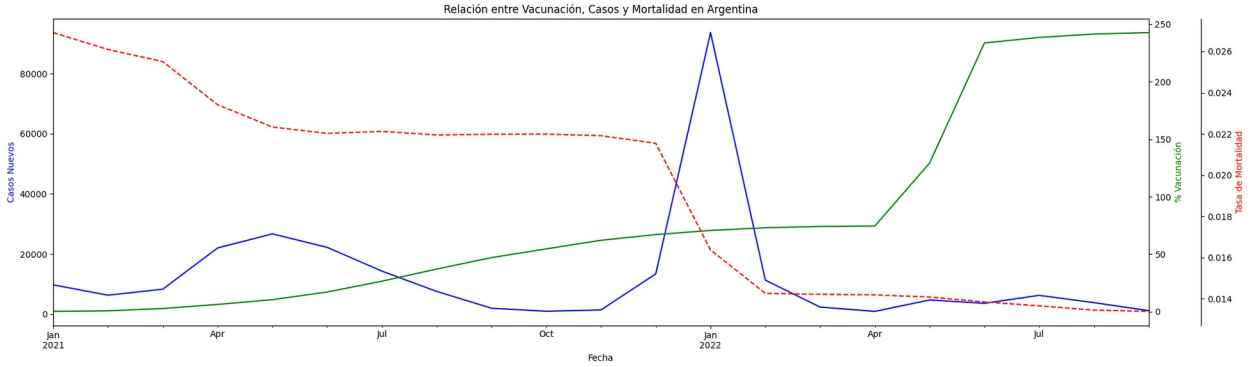


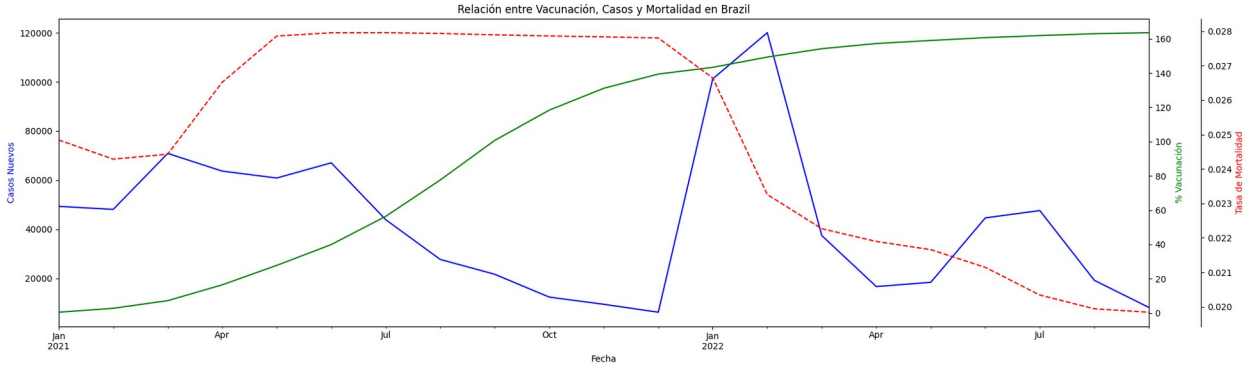
En la totalidad de los paises parece haber un pico de casos hacia Junio 2022, no obstante, el marcado ascenso de dosis de vacunas administradas, refleja una disminución significativa en la cantidad de muertes.

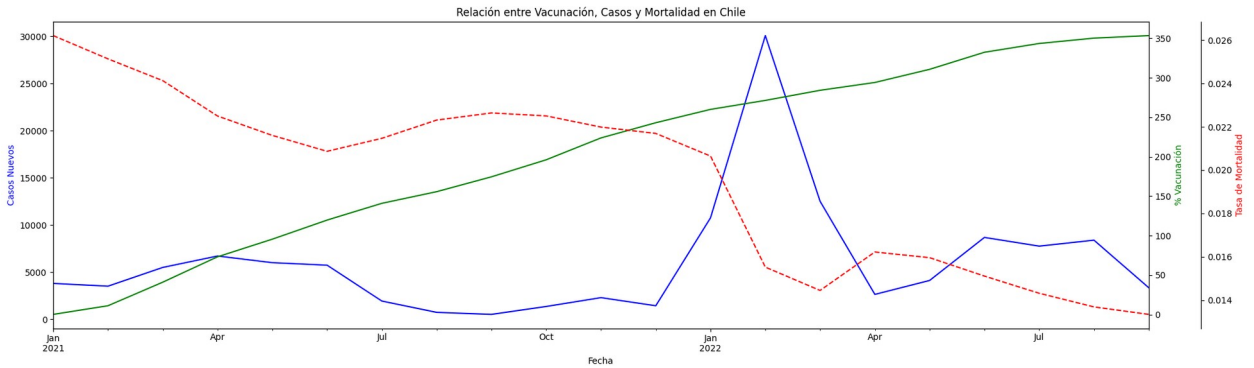
Analizaré las mismas curvas pero por país

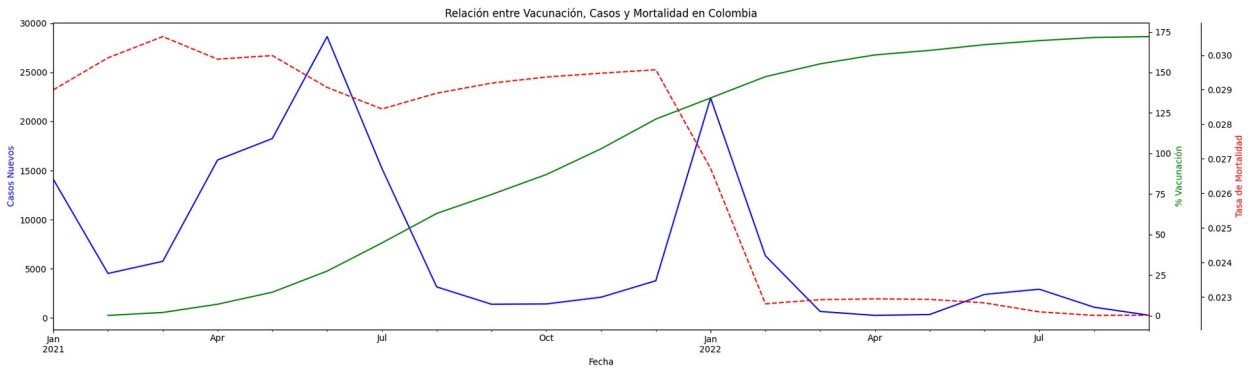
```
# Aseguro columnas derivadas
df_limpio['percentage_vaccinated'] = (
    df_limpio['cumulative_vaccine_doses_administered'] /
df_limpio['population'] * 100
)
```

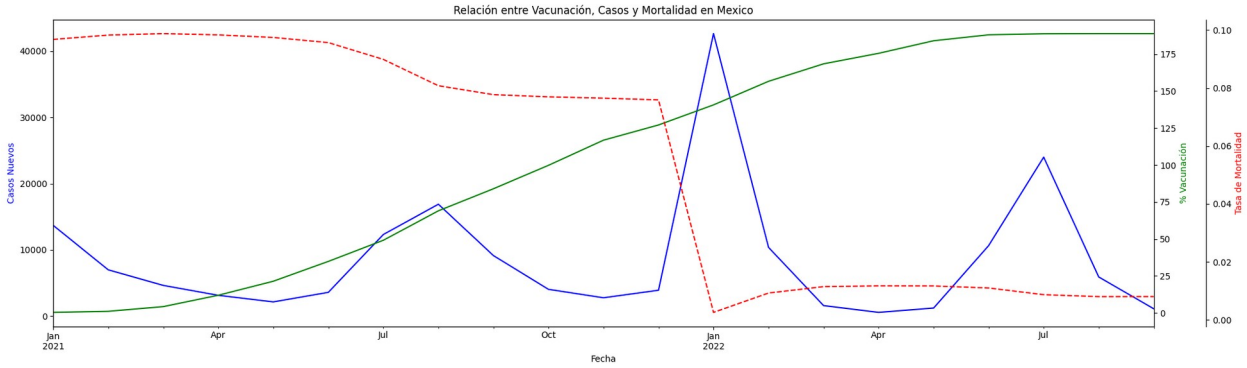
```
df_limpio['mortality_rate'] = (
    df_limpio['cumulative_deceased'] /
df_limpio['cumulative_confirmed']
)

# Iterar por país
for country in df_limpio['country_name'].unique():
    df_country = df_limpio[df_limpio['country_name'] == country]

    # resample mensual (ME = end of month)
    df_monthly = df_country.resample('ME').mean(numeric_only=True)

    # Crear figura y ejes
    fig, ax1 = plt.subplots(figsize=(20, 6))
    ax2 = ax1.twinx()
    ax3 = ax1.twinx()



    # Plots
    df_monthly['new_confirmed'].plot(ax=ax1, color='blue',
label='Casos Nuevos')
    df_monthly['percentage_vaccinated'].plot(ax=ax2, color='green',
label='% Vacunación')
    df_monthly['mortality_rate'].plot(ax=ax3, color='red',
linestyle='--', label='Tasa de Mortalidad')

    # Etiquetas
    ax1.set_xlabel('Fecha')
    ax1.set_ylabel('Casos Nuevos', color='blue')
    ax2.set_ylabel('% Vacunación', color='green')
    ax3.set_ylabel('Tasa de Mortalidad', color='red')
    # Para que el tercer eje no se superponga con el segundo:
    ax3.spines['right'].set_position(('outward', 60))
    # Título y leyendas
    ax1.set_title(f'Relación entre Vacunación, Casos y Mortalidad en
{country}')

    plt.tight_layout()
    plt.show()
```

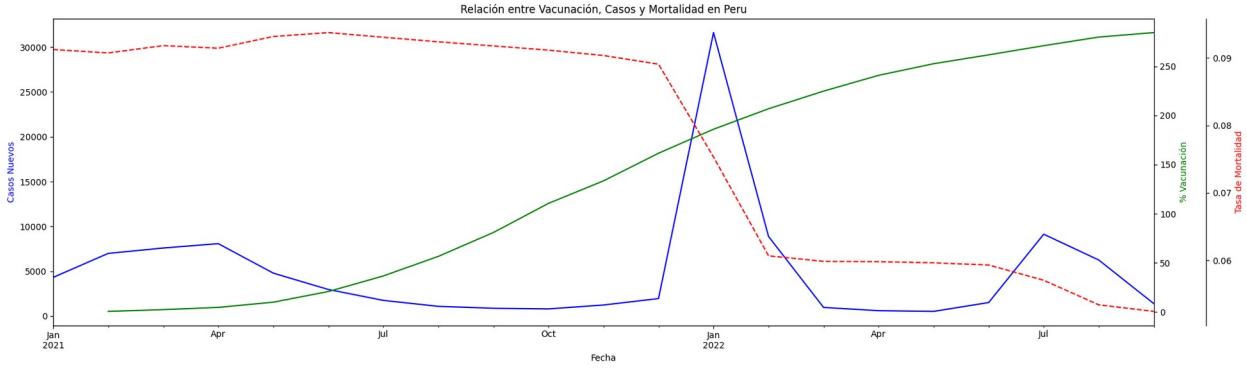

Relación entre Vacunación, Casos y Mortalidad en Argentina
Casos Nuevos
% Vacunación
Tasa de Mortalidad
Fecha
Jan 2021
Apr
Jul
Oct
Jan 2022
Apr
Jul


Relación entre Vacunación, Casos y Mortalidad en Brazil
Casos Nuevos
% Vacunación
Tasa de Mortalidad
Fecha
Jan 2021
Apr
Jul
Oct
Jan 2022
Apr
Jul


Relación entre Vacunación, Casos y Mortalidad en Chile
Casos Nuevos
% Vacunación
Tasa de Mortalidad
Fecha
Jan 2021
Apr
Jul
Oct
Jan 2022
Apr
Jul


Relación entre Vacunación, Casos y Mortalidad en Colombia
Casos Nuevos
% Vacunación
Tasa de Mortalidad
Fecha
Jan 2021
Apr
Jul
Oct
Jan 2022
Apr
Jul



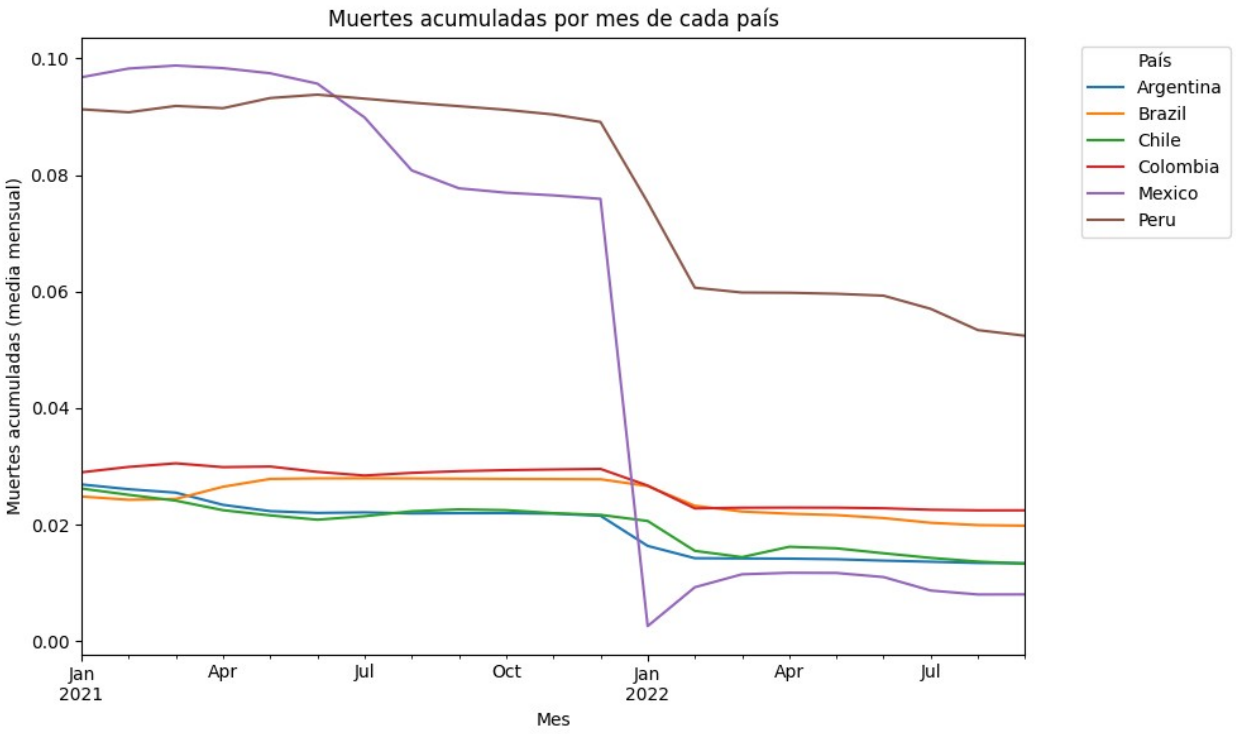


A Perú le constó controlar los casos de muerte en toda la fase incial. Parece haber mayor números de casos también frente a los otros paises.

Demuestro en el gráfico siguiente este comportamiento

```
df_limpio['mortality_rate'] = (
    df_limpio['cumulative_deceased'] /
df_limpio['cumulative_confirmed']
)

# Gráfica de la evolución mensual de muertes acumuladas por país
fig, ax = plt.subplots (figsize=(10, 6))

# Iterar sobre cada país y graficar la media mensual de muertes
acumuladas
for pais in  df_limpio['country_name'].unique():

    # Filtrar datos por país y hacer resample mensual (fin de mes)
    datos_pais = df_limpio[df_limpio['country_name'] ==
pais].resample('ME').mean(numeric_only=True)
    # Graficar la serie temporal de muertes acumuladas
    datos_pais['mortality_rate'].plot(ax=ax, label=pais)
# Configurar el título y las etiquetas
plt.title('Muertes acumuladas por mes de cada país')
plt.xlabel('Mes')
plt.ylabel('Muertes acumuladas (media mensual)')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', title='País')
```

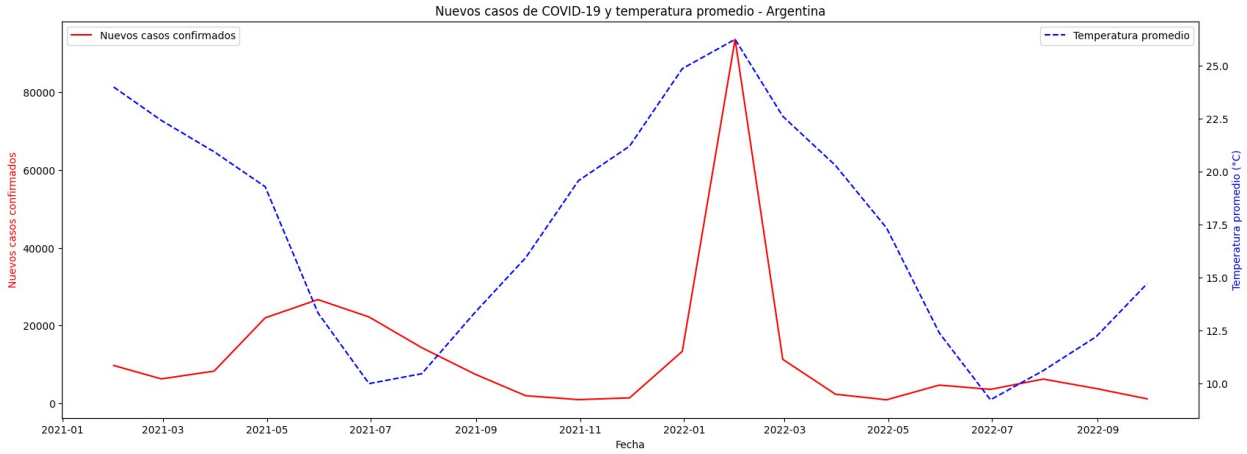
```
plt.tight_layout()
plt.show()
```

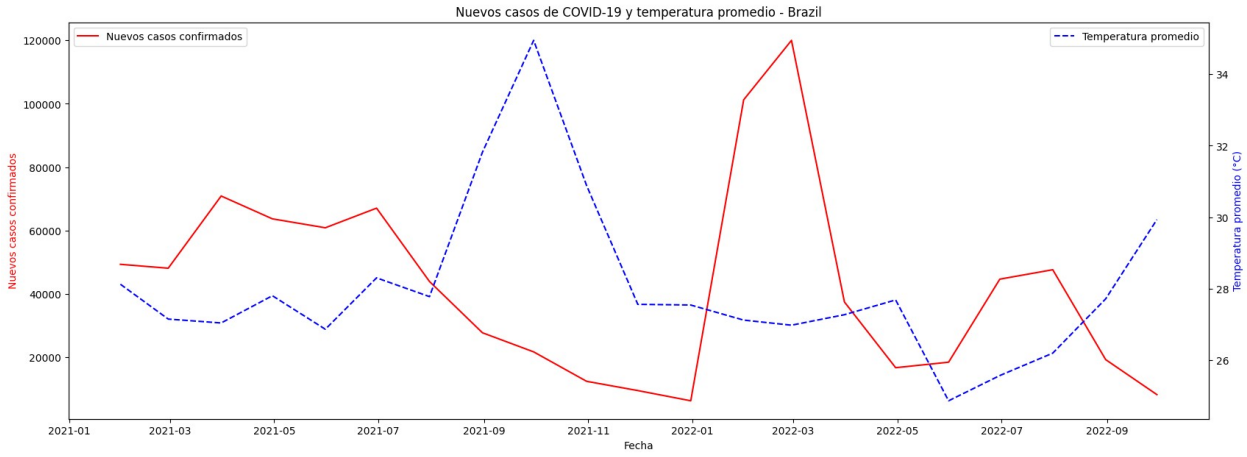


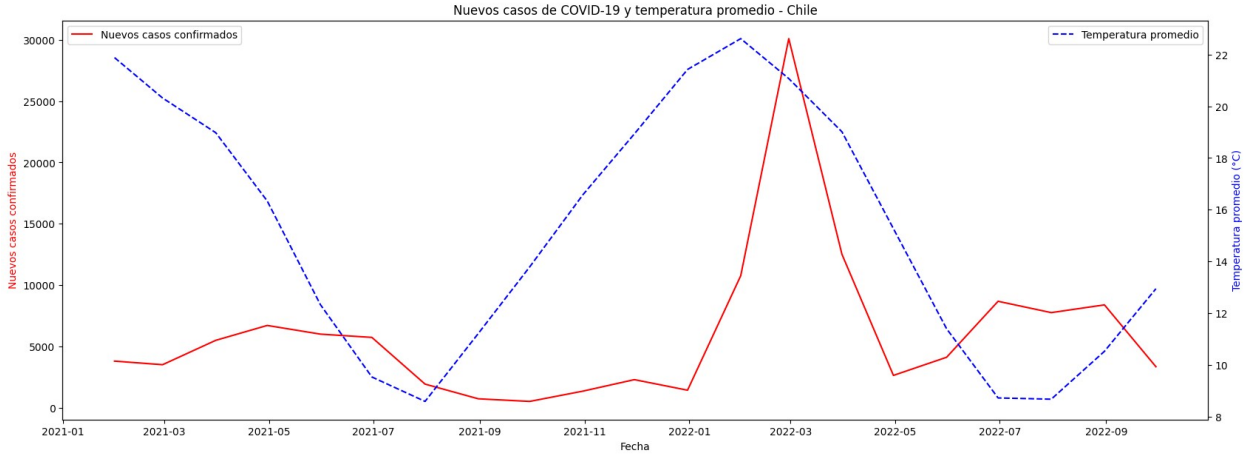
Análisis de variables ambientales

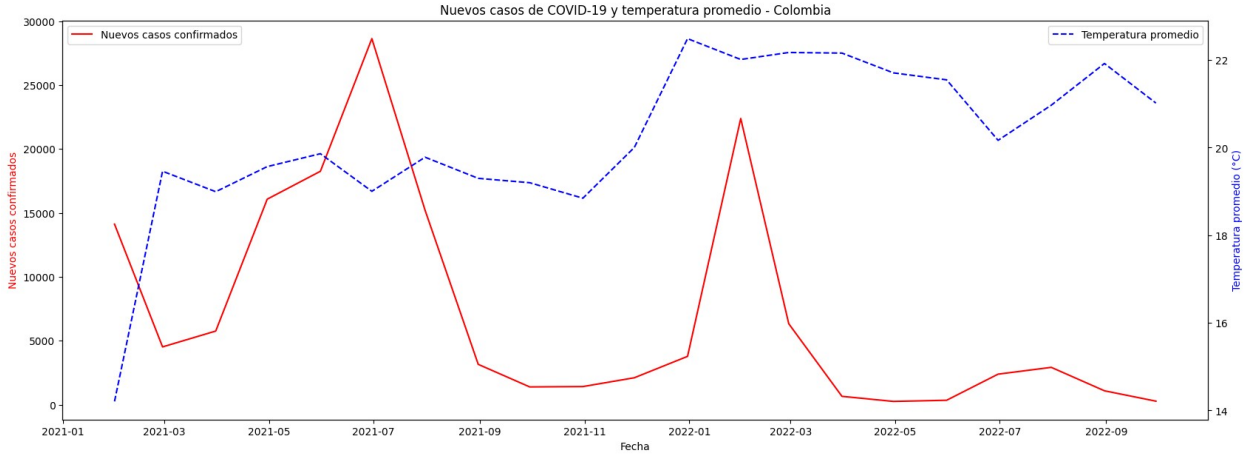
```
# Analizo la relación entre la temperatura promedio y los nuevos casos confirmados de COVID-19 por cada país
for pais in df_limpio['country_name'].unique():
    df_pais = df_limpio[df_limpio['country_name'] == pais]

    # Resampleo mensual por país
    promedio_mensual = df_pais.resample('ME').mean(numeric_only=True)

    # Crear gráfico
    plt.figure(figsize=(20, 7))
    ax1 = plt.gca()
    ax2 = ax1.twinx()

    ax1.plot(promedio_mensual.index,
promedio_mensual['new_confirmed'],
             color='red', label='Nuevos casos confirmados')
    ax2.plot(promedio_mensual.index,
promedio_mensual['average_temperature_celsius'],
             color='blue', linestyle='--', label='Temperatura
promedio')
```

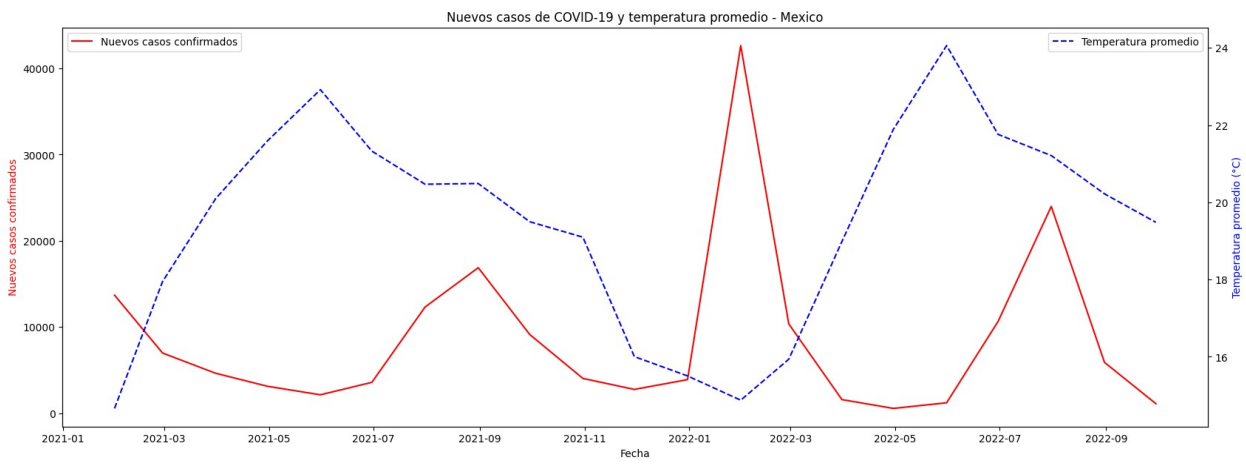
```
    ax1.set_xlabel('Fecha')
    ax1.set_ylabel('Nuevos casos confirmados', color='red')
    ax2.set_ylabel('Temperatura promedio (°C)', color="blue")
    plt.title(f'Nuevos casos de COVID-19 y temperatura promedio -
{pais}')

    ax1.legend(loc='upper left')
    ax2.legend(loc='upper right')

    plt.show()
```

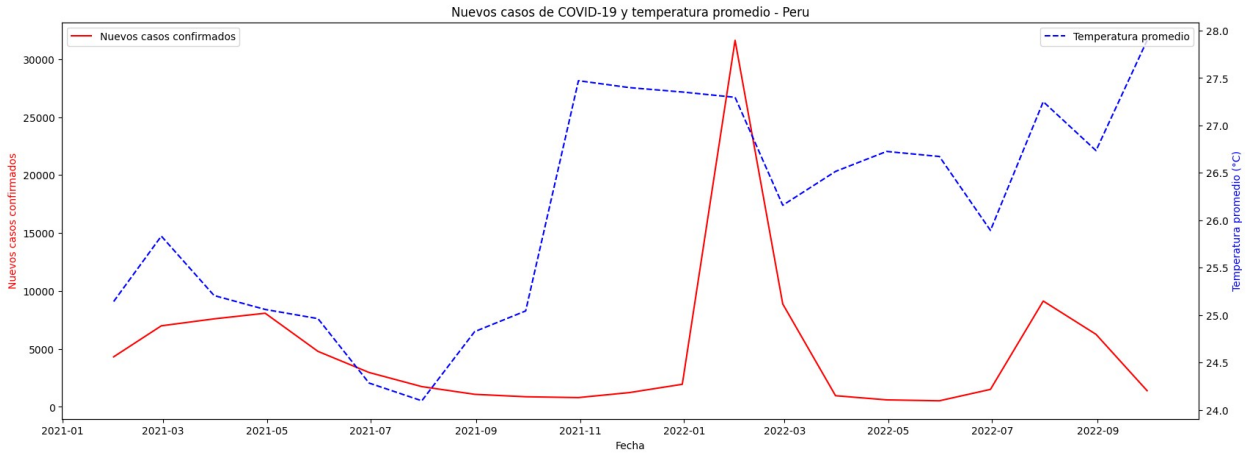





Nuevos casos de COVID-19 y temperatura promedio - Chile
Nuevos casos confirmados
Temperatura promedio
Nuevos casos confirmados
Temperatura promedio (°C)
Fecha


Nuevos casos de COVID-19 y temperatura promedio - Colombia
Nuevos casos confirmados
Temperatura promedio
Nuevos casos confirmados
Temperatura promedio (°C)
Fecha

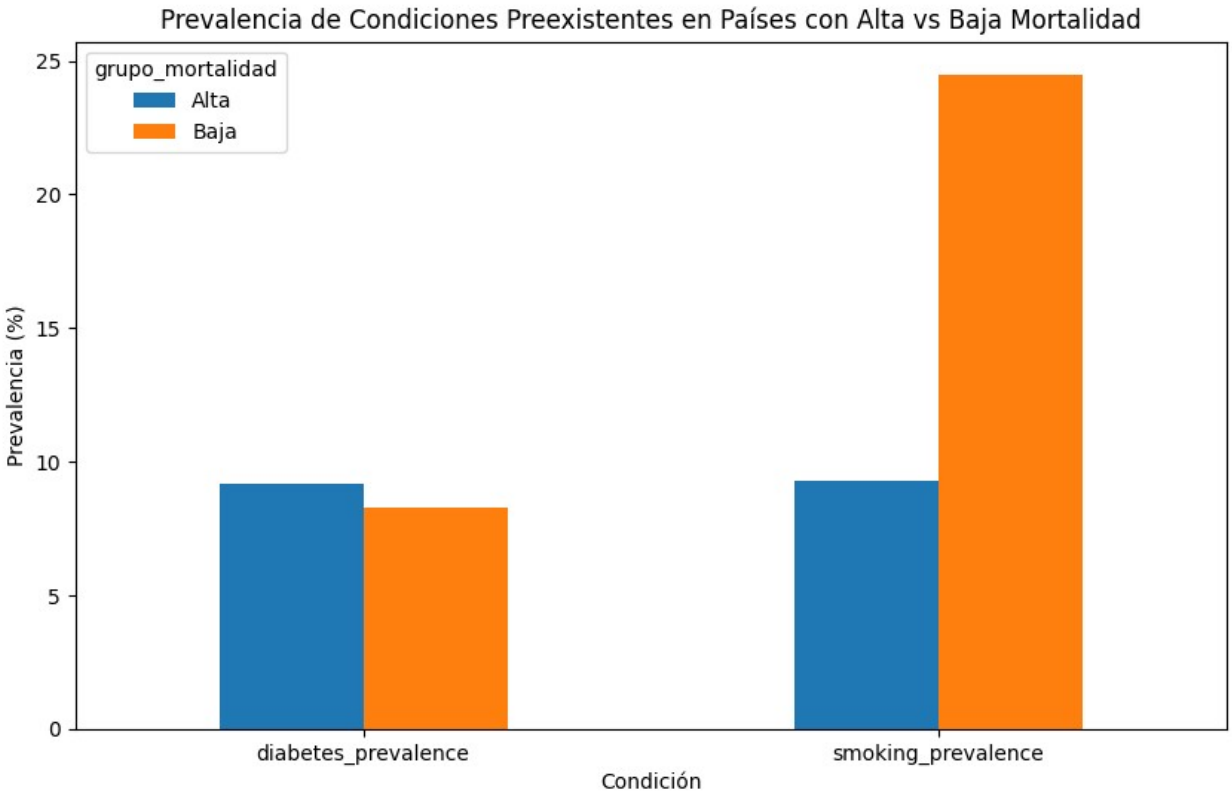

Nuevos casos de COVID-19 y temperatura promedio - Mexico
Nuevos casos confirmados
Temperatura promedio
Nuevos casos confirmados
Temperatura promedio (°C)
Fecha



Sólo en Argentina parece coincidir el pico de temperaturas con el pico de casos nuevos, en los demás países no parece haber una relación clara entre ambas variables.

Análisis de Prevalencia de Condiciones Preexistentes en Países con Altas y Bajas Tasas de Mortalidad

```
# Análisis de Prevalencia de Condiciones Preexistentes en Países con
Altas y Bajas Tasas de Mortalidad
# Calculo las medidas
df_mortalidad = df_limpio.groupby("country_name").agg({
    "cumulative_deceased":"max",
    "cumulative_confirmed":"max",
    "diabetes_prevalence":"mean",
    "smoking_prevalence":"mean"
}).reset_index()

df_mortalidad["mortality_rate"] = df_mortalidad["cumulative_deceased"]
/ df_mortalidad["cumulative_confirmed"]
mediana = df_mortalidad["mortality_rate"].median()
df_mortalidad["grupo_mortalidad"] =
df_mortalidad["mortality_rate"].apply(lambda x: "Alta" if x > mediana
else "Baja")

# defino las variables a analizar
condiciones = ["diabetes_prevalence","smoking_prevalence"]
df_condiciones = df_mortalidad.groupby("grupo_mortalidad")
[condiciones].mean().T

df_condiciones.plot(kind="bar", figsize=(10,6))
plt.title("Prevalencia de Condiciones Preexistentes en Países con Alta
vs Baja Mortalidad")
plt.ylabel("Prevalencia (%)")
plt.xlabel("Condición")
plt.xticks(rotation=0)
plt.show()
```



No parece haber una relación entre las condiciones pre existentes y la tasa de mortalidad.