

✓ Limpieza y Preparación de Datos caso covid-19 fase 1

Carlos Rodrigo Piñeros, Juan Restrepo Rosero y Joel Doria Atencia

curso gestión de datos, Maestria en ciencia de datos Universidad Javeriana de Cali

Limpieza y preparación de datos

Es el proceso fundamental de transformar datos disponibles en información limpia, consistente y utilizable para el análisis, la toma de decisiones y el aprendizaje automático. Este proceso implica identificar y corregir errores, inconsistencias y valores faltantes en los datos, así como estandarizar formatos y estructuras. Este proceso es importante porque permite:

- Mayor precisión en los resultados: Datos limpios conducen a análisis más precisos y confiables.
- Aumento de la eficiencia: Datos preparados aceleran los procesos de análisis y modelado.
- Prevención de errores: Identificar y corregir errores desde el principio evita problemas a largo plazo.

Durante la limpieza de datos se resaltan las siguientes actividades:

- Detección de valores atípicos: Identificar valores que se desvían significativamente de la norma.
- Corrección de errores: Corregir errores de tipeo, formato y cálculo.
- Tratamiento de valores faltantes: Imputar valores faltantes utilizando métodos estadísticos o de aprendizaje automático.
- Eliminación de duplicados: Eliminar registros duplicados para evitar sesgos en el análisis.
- Normalización: Convertir datos a un formato común y comparable.

En la preparación de datos se resaltan:

- Transformación: Convertir datos a un formato adecuado para el análisis (ej: discretización, binarización).
- Creación de nuevas variables: Derivar nuevas variables a partir de las existentes (ej: ratios, indicadores).
- Selección de variables: Seleccionar las variables más relevantes para el análisis.
- Reducción de dimensionalidad: Reducir el número de variables para mejorar la eficiencia computacional.

La limpieza y preparación de datos es una etapa crucial en cualquier proyecto de análisis de datos. Al invertir tiempo y esfuerzo en esta fase, se garantiza la calidad de los resultados y se tomarán decisiones más acertadas.

casos confirmados covid-19

Este dataset es un repositorio ampliamente utilizado en la comunidad científica para analizar la evolución de la pandemia de COVID-19 a nivel mundial. Este archivo CSV contiene información sobre el número acumulado de casos confirmados de COVID-19 por país y por fecha, desde el inicio de la pandemia.

Para analizar este dataset, vamos a utilizar el lenguaje de programación Python, con las librerías Pandas, Numpy y Matplotlib para responder las siguientes preguntas:

1. ¿En cuál mes se presentó el mayor número de contagios?
2. ¿En ese mismo mes, cuál fue el país que reportó más contagios?
3. ¿Cuál es el país con el menor número de casos reportados hasta la fecha?

```
# Importar librerías Pandas, Numpy y Matplotlib para analizar y limpiar los datos
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
# Utilizar la función read_csv() de Pandas para cargar el archivo CSV
data_origen = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/csse_covid_19_daily_reports.csv')
```

Dado que estamos trabajando con un dataset de casos confirmados de COVID-19, hay algunos aspectos específicos que debemos considerar:

✓ 1. Comprender la Estructura de los Datos:


El dataset presenta la siguiente estructura:

- **Province/State:** Región o provincia. Tipo categórico.
- **Country/Region:** País. Tipo categórico.
- **Lat:** Latitud geográfica. Tipo numérico.

- **Long:** Longitud geográfica. Tipo numérico.
- **Columnas desde 1/22/20:** representan las fechas, a partir de las cuales se registra el número acumulado de casos, comenzando desde el 22 de enero de 2020. Tipo numérico.

El valor numérico en cada celda indica el número total de casos confirmados hasta esa fecha en el país o región correspondiente, como se observa a continuación con los metodos `head()` y `tail()` de pandas:

```
# Para visualizar los primeros registros usamos el método head().
data_origen.head(10)
```



	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	2/2
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	20
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	...	33
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	...	27
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	...	4
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	...	10
5	NaN	Antarctica	-71.94990	23.347000	0	0	0	0	0	0	...	
6	NaN	Antigua and Barbuda	17.06080	-61.796400	0	0	0	0	0	0	...	
7	NaN	Argentina	-38.41610	-63.616700	0	0	0	0	0	0	...	1004
8	NaN	Armenia	40.06910	45.038200	0	0	0	0	0	0	...	44
9	Australian Capital Territory	Australia	-35.47350	149.012400	0	0	0	0	0	0	...	23

```
# Para visualizar los ultimos registros usamos el método tail().
data_origen.tail(10)
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...
279	NaN	Uruguay	-32.522800	-55.765800	0	0	0	0	0	0	...
280	NaN	Uzbekistan	41.377491	64.585262	0	0	0	0	0	0	...
281	NaN	Vanuatu	-15.376700	166.959200	0	0	0	0	0	0	...
282	NaN	Venezuela	6.423800	-66.589700	0	0	0	0	0	0	...
283	NaN	Vietnam	14.058324	108.277199	0	2	2	2	2	2	... 1
284	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	0	...
285	NaN	Winter Olympics 2022	39.904200	116.407400	0	0	0	0	0	0	...
286	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	0	...
287	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	0	...
288	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	0	...

Usamos el método `info()` para obtener información de las columnas del dataset.
`data_origen.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 289 entries, 0 to 288
Columns: 1147 entries, Province/State to 3/9/23
dtypes: float64(2), int64(1143), object(2)
memory usage: 2.5+ MB
```

El método `info()` proporciona un resumen conciso del Dataset, incluyendo el número de filas y columnas, los tipos de datos de cada columna y el uso de memoria:

- Dimensiones: El DataFrame tiene 289 filas (registros) y 1147 columnas (que corresponden a las fechas).
- Tipos de Datos:

- **Float64 (2):** Dos columnas contienen números de punto flotante, que corresponden a la latitud y longitud geográficas.
 - **Int64 (1143):** La mayoría de las columnas son de tipo entero, que corresponden a los conteos acumulados en cada fecha.
 - **Object (2):** Estas columnas probablemente contienen texto, que corresponden a las columnas Province/State y Country/Region.
- **Uso de Memoria:** El DataFrame ocupa aproximadamente 2.5 MB en la memoria, lo cual es relativamente bajo considerando el gran número de columnas.

```
# Si usamos el método describe() con el parámetros include='O' podemos ver información adicional de las colu
# que tienen valores categóricos.
data_origen.describe(include='O')
```



	Province/State	Country/Region
count	91	289
unique	91	201
top	Australian Capital Territory	China
freq	1	34


El metodo describe() con el parámetros include='O' nos proporciona información sobre la distribución de los valores dentro de estas dos columnas categóricas.

- **count:** Indica el número total de valores no nulos en cada columna. En este caso, tenemos 91 valores únicos para 'Province/State' y 289 valores para 'Country/Region'. Esto sugiere que hay más de un valor de 'Province/State' por cada valor de 'Country/Region', lo cual es lógico, ya que un país puede tener múltiples provincias o estados.
- **unique:** Muestra el número de valores únicos en cada columna. Esto significa que hay 91 provincias/estados diferentes y 201 países/regiones diferentes en el dataset.
- **top:** Indica el valor más frecuente en cada columna. En este caso, 'Australian Capital Territory' es la provincia/estado que aparece con mayor frecuencia, mientras que 'China' es el país/región que aparece con mayor frecuencia.

- **freq:** Muestra la frecuencia del valor más frecuente. En este caso, 'Australian Capital Territory' aparece una vez, mientras que 'China' aparece 34 veces.

Usamos el método describe para obtener datos estadísticos de las columnas de tipo numérico.

data_origen.describe().T



	count	mean	std	min	25%	50%	75%	max
Lat	287.0	1.971872e+01	2.595661e+01	-71.9499	4.072192	21.512583	4.040178e+01	7.170690e+01
Long	287.0	2.218208e+01	7.787093e+01	-178.1165	-32.823050	20.939400	8.922435e+01	1.780650e+02
1/22/20	289.0	1.927336e+00	2.617366e+01	0.0000	0.000000	0.000000	0.000000e+00	4.440000e+02
1/23/20	289.0	2.273356e+00	2.627019e+01	0.0000	0.000000	0.000000	0.000000e+00	4.440000e+02
1/24/20	289.0	3.266436e+00	3.270727e+01	0.0000	0.000000	0.000000	0.000000e+00	5.490000e+02
...
3/5/23	289.0	2.339187e+06	8.518645e+06	0.0000	14567.000000	103248.000000	1.052664e+06	1.036470e+08
3/6/23	289.0	2.339387e+06	8.519346e+06	0.0000	14567.000000	103248.000000	1.052664e+06	1.036555e+08
3/7/23	289.0	2.339839e+06	8.521641e+06	0.0000	14567.000000	103248.000000	1.052926e+06	1.036909e+08
3/8/23	289.0	2.340460e+06	8.524968e+06	0.0000	14567.000000	103248.000000	1.053068e+06	1.037558e+08
3/9/23	289.0	2.341073e+06	8.527765e+06	0.0000	14567.000000	103248.000000	1.053213e+06	1.038027e+08

1145 rows × 8 columns

Analizar esta salida de describe() nos permite tener una idea inicial de la distribución de los valores en cada columna numérica y detectar posibles valores atípicos. Aquí un análisis más detallado:

Columnas de Latitud (Lat) y Longitud (Long)

Promedio y Desviación Estándar:

- Estos valores no tienen una aplicación practica, dado que corresponden a coordenadas geográficas de los países o provincias.

Rango de Valores:

- Latitudes: Desde -71.95 hasta 71.71.
- Longitudes: Desde -178.12 hasta 178.06.

Estos valores son consistentes con coordenadas geográficas, pero su gran rango y desviación estándar podrían indicar valores atípicos en ubicaciones extremas (como países cerca de los polos).

Columnas de Fechas

Cada una de estas columnas representa la cantidad de casos confirmados acumulados para una fecha específica y muestra las siguientes estadísticas:

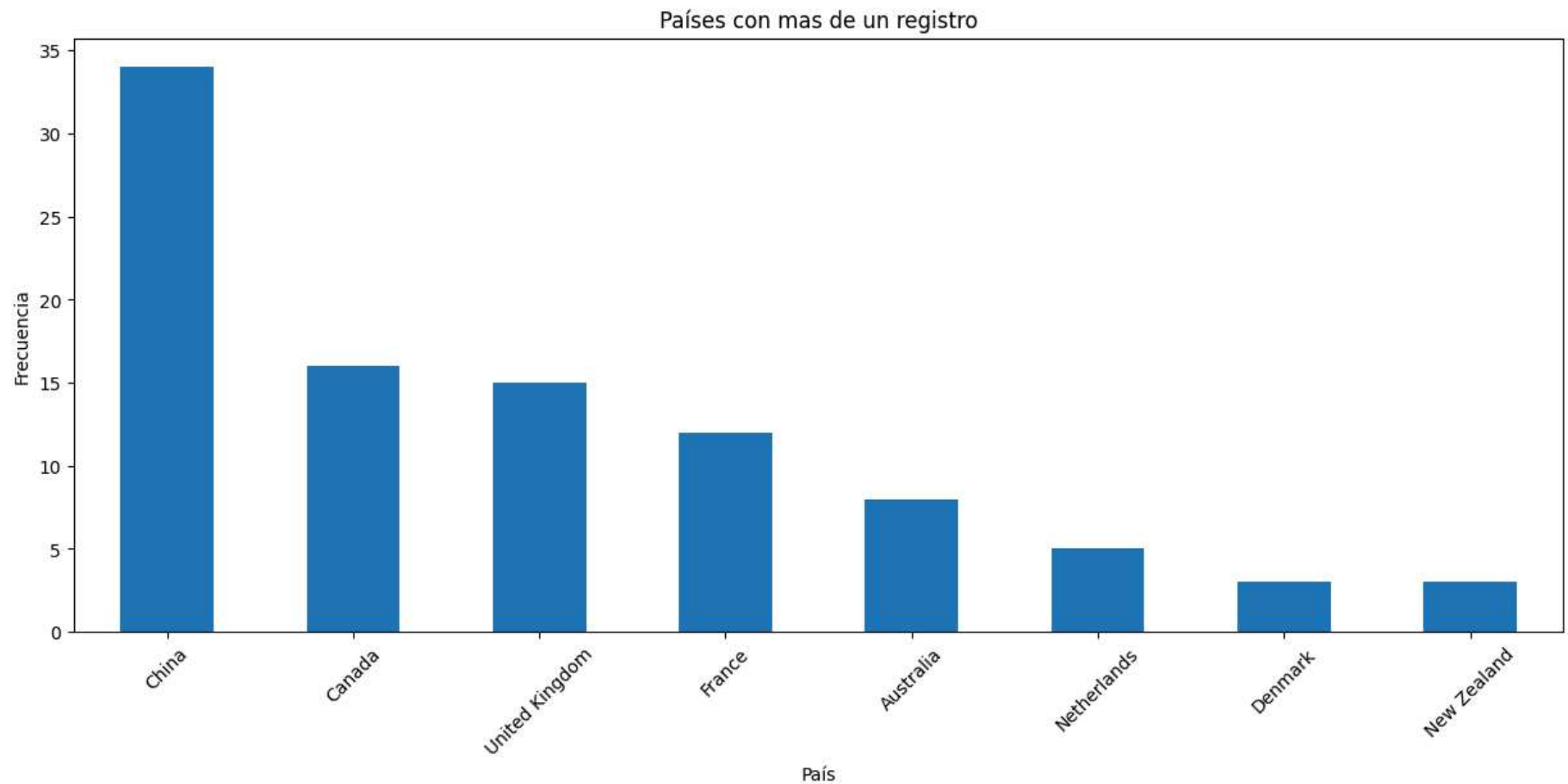
Promedios Altos y Rango de Valores Extremos:

- Los valores promedio incrementan conforme se avanza en las fechas, alcanzando millones en las últimas fechas (3/5/23 a 3/9/23), lo que es consistente con el aumento acumulativo de casos de COVID-19 en el tiempo.
- La desviación estándar también es considerable, lo que sugiere una amplia variabilidad entre países.

Percentiles e Incremento en la Mediana:

- El incremento en el 50% (mediana) y el 75% en los últimos días indica que la mayoría de los países muestran un acumulado de casos, con un incremento continuo y gradual en los casos confirmados.

```
# Contar la frecuencia de cada país
country_counts = data_origen['Country/Region'].value_counts()
# Filtrar los países con frecuencia mayor a 1
country_counts = country_counts[country_counts > 1]
# Crear un gráfico de barras
plt.figure(figsize=(15, 6))
country_counts.plot(kind='bar')
plt.title('Países con mas de un registro')
plt.xlabel('País')
plt.ylabel('Frecuencia')
plt.xticks(rotation=45)
plt.show()
```



En el grafico anterior, se muestran los paises que tienen varios registros en el dataset, que se deben evaluar si es necesario unificarlos. Estos registros se deben a que estos países reportaron información de varias provincias con sus coordenadas geográficas.

A continuación, se evalúan los valores únicos de la columna países, donde se observan algunos registros que no son países en realidad como 'Winter Olympics 2022'

```
# Usamos el método unique() sobre la columna deseada para verificar cuáles son los valores únicos.
data_origen['Country/Region'].unique()
```

```
→ array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
        'Antarctica', 'Antigua and Barbuda', 'Argentina', 'Armenia',
        'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
        'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin',
        'Bhutan', 'Bolivia', 'Bosnia and Herzegovina', 'Botswana',
        'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi',
        'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
        'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia',
        'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Costa Rica',
        "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
        'Diamond Princess', 'Djibouti', 'Dominica', 'Dominican Republic',
        'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
        'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'France',
        'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
        'Grenada', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana',
        'Haiti', 'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
        'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
        'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
        'Korea, North', 'Korea, South', 'Kosovo', 'Kuwait', 'Kyrgyzstan',
        'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya',
        'Liechtenstein', 'Lithuania', 'Luxembourg', 'MS Zaandam',
        'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta',
        'Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico',
        'Micronesia', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
        'Morocco', 'Mozambique', 'Namibia', 'Nauru', 'Nepal',
        'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria',
        'North Macedonia', 'Norway', 'Oman', 'Pakistan', 'Palau', 'Panama',
        'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland',
        'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
        'Saint Kitts and Nevis', 'Saint Lucia',
        'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
        'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
        'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
        'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
```

```
'Spain', 'Sri Lanka', 'Sudan', 'Summer Olympics 2020', 'Suriname',
'Sweden', 'Switzerland', 'Syria', 'Taiwan*', 'Tajikistan',
'Tanzania', 'Thailand', 'Timor-Leste', 'Togo', 'Tonga',
'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Tuvalu', 'US',
'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom',
'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam',
'West Bank and Gaza', 'Winter Olympics 2022', 'Yemen', 'Zambia',
'Zimbabwe'], dtype=object)
```

✓ 2. Limpieza de Datos

Errores de tipeo: No se evidencian errores de tipeo en los nombres de países, regiones o provincias.

Valores atípicos: No se identifican valores atípicos que afecten el análisis para responder las preguntas planteadas inicialmente.

Duplicados: a continuación se procede a transformar el dataset para eliminar filas duplicadas y calcular el número de nuevos casos diarios, para evitar sesgos en el análisis.

Tenemos un dataset con una estructura bastante específica, donde las columnas a partir de una cierta posición representan fechas y los valores correspondientes representan el número de casos confirmados en una determinada fecha. El objetivo es transformar este dataset en uno más estructurado, con columnas claras para:

- Provincia: El nombre de la provincia o región.
- País: El país al que pertenece la provincia.
- Fecha: La fecha específica.
- Casos confirmados: El número de casos confirmados en esa fecha para esa provincia.

```
# Identificar las columnas que queremos mantener como identificadores
id_vars = ['Province/State', 'Country/Region', 'Lat', 'Long']
```

```
# Derretir el DataFrame para obtener una fila por cada combinación de provincia, país, fecha y caso confirmado
df_melted = pd.melt(data_origen,
                    id_vars=id_vars,
                    var_name='Fecha',
                    value_name='Casos confirmados')
```

```
# Convertir la columna 'Fecha' a tipo datetime
df_melted['Fecha'] = pd.to_datetime(df_melted['Fecha'], format='%m/%d/%y')
```

El dataset resultante tendrá una fila por cada combinación de provincia, país y fecha, con las columnas correspondientes a los valores de cada variable.

A continuación, se agrupan los datos por país y fecha para calcular el numero de casos diarios. Adicionalmente, se crean columnas para el año y el mes para realizar el análisis más detallado y así responder a las preguntas planteadas.

```
# Agrupar por 'Country/Region' y 'Fecha' para consolidar los casos diarios de todas las provincias de un país
df_consolidado = df_melted.groupby(['Country/Region', 'Fecha']).agg({'Casos confirmados': 'sum'}).reset_index()

# Calcular los casos diarios (diferencia entre días consecutivos)
df_consolidado = df_consolidado.sort_values(by=['Country/Region', 'Fecha'])
df_consolidado['CasosDiarios'] = df_consolidado.groupby('Country/Region')['Casos confirmados'].diff().fillna(0)

# Crear columnas para el año y el mes para el análisis mensual y anual
df_consolidado['Año'] = df_consolidado['Fecha'].dt.year
df_consolidado['Mes'] = df_consolidado['Fecha'].dt.month
```

Ahora, se procede a responder la pregunta inicial: **¿En cuál mes se presentó el mayor número de contagios?**

```
# 1. Calcular los contagios por mes y encontrar el mes con mayor número de contagios

# Agrupar por año y mes para obtener los casos totales por mes
casos_por_mes_y_ano = df_consolidado.groupby(['Año', 'Mes'])['CasosDiarios'].sum()

# Encontrar el año y mes con el mayor número de casos
max_casos = casos_por_mes_y_ano.max()
```

```
indice_max = casos_por_mes_y_ano.idxmax()
ano_max, mes_max = indice_max

print(f"El año con más casos diarios fue: {ano_max}")
print(f"El mes con más casos diarios fue: {mes_max}")
print(f"Número total de casos en ese mes y año: {max_casos}")
```

```
→ El año con más casos diarios fue: 2022
   El mes con más casos diarios fue: 1
   Número total de casos en ese mes y año: 90483564.0
```

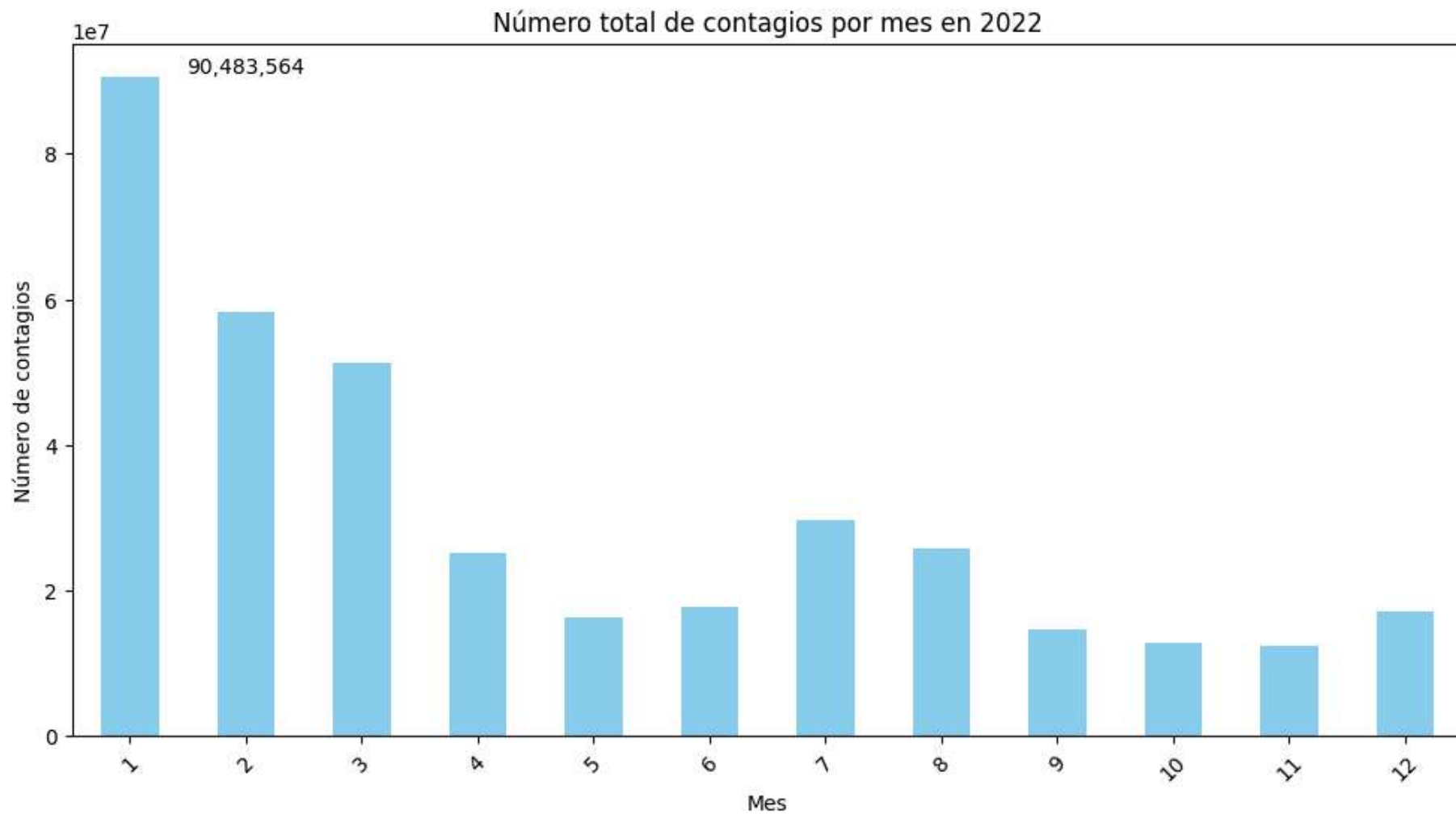
Adicionalmente, los siguientes gráficos muestran los meses con más contagios en el año 2022 y también desde el inicio de la pandemia:

```
# Gráfico de barras de casos por mes en el año pico
plt.figure(figsize=(12, 6))
ax = casos_por_mes_y_ano.loc[ano_max].plot(kind='bar', color='skyblue')
plt.title(f'Número total de contagios por mes en {ano_max}')
plt.xlabel('Mes')
plt.ylabel('Número de contagios')
plt.xticks(rotation=45)

# Identificar el mes con el mayor número de contagios
mes_max_indice = casos_por_mes_y_ano.loc[ano_max].idxmax() # Mes con más contagios en el año máximo
valor_max = casos_por_mes_y_ano.loc[ano_max][mes_max_indice]

# Mostrar el valor en la barra correspondiente
ax.text(mes_max_indice, valor_max + 5000, f'{valor_max:,.0f}', ha='center', va='bottom', fontsize=10)

plt.show()
```



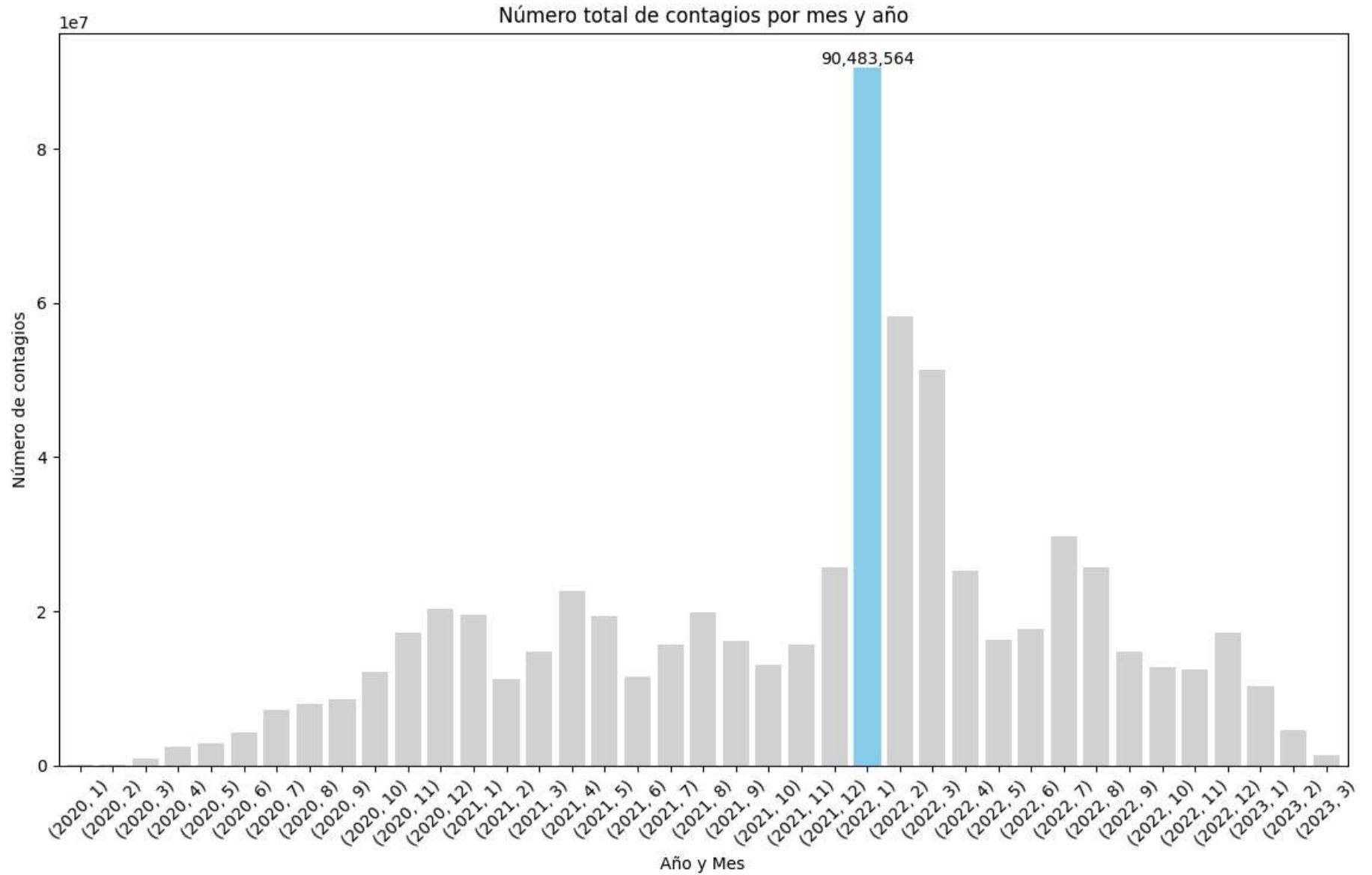
```
# Gráfico de todos los meses y años
plt.figure(figsize=(14, 8))
ax = casos_por_mes_y_ano.plot(kind='bar', color='lightgray', width=0.8) # Color neutro para todas las barras
plt.title('Número total de contagios por mes y año')
plt.xlabel('Año y Mes')
plt.ylabel('Número de contagios')
```

```
plt.xticks(rotation=45)

# Resaltar la barra del mes con el mayor número de contagios
indice_max_barra = casos_por_mes_y_ano.index.get_loc(indice_max) # Índice de la barra del mes máximo
ax.patches[indice_max_barra].set_color('skyblue') # Cambiar el color de la barra específica

# Mostrar el valor en la barra del mes máximo
ax.text(indice_max_barra, max_casos + 5000, f'{max_casos:,.0f}', ha='center', va='bottom', fontsize=10, color='black')

plt.show()
```



¿En ese mismo mes, cuál fue el país que reportó más contagios?

```
# 2. Encontrar el país con más contagios en el mes con mayor número de contagios
df_mes_max = df_consolidado[(df_consolidado['Año'] == ano_max) & (df_consolidado['Mes'] == mes_max)]

# Agrupar por país y sumar los casos diarios para obtener el total de contagios en el mes pico
contagios_por_pais_mes_max = df_mes_max.groupby('Country/Region')['CasosDiarios'].sum()

# Identificar el país con el mayor número de contagios en ese mes
pais_max_contagios = contagios_por_pais_mes_max.idxmax()
print(f"En {ano_max}-{mes_max}, el país con más contagios fue {pais_max_contagios}, con {contagios_por_pais_mes_max[pais_max_contagios].sum()} casos.")
```

➡ En 2022-1, el país con más contagios fue US, con 20,336,435.0 casos.

En el siguiente grafico se muestran los diez países con más casos en el mes con mayor número de contagios

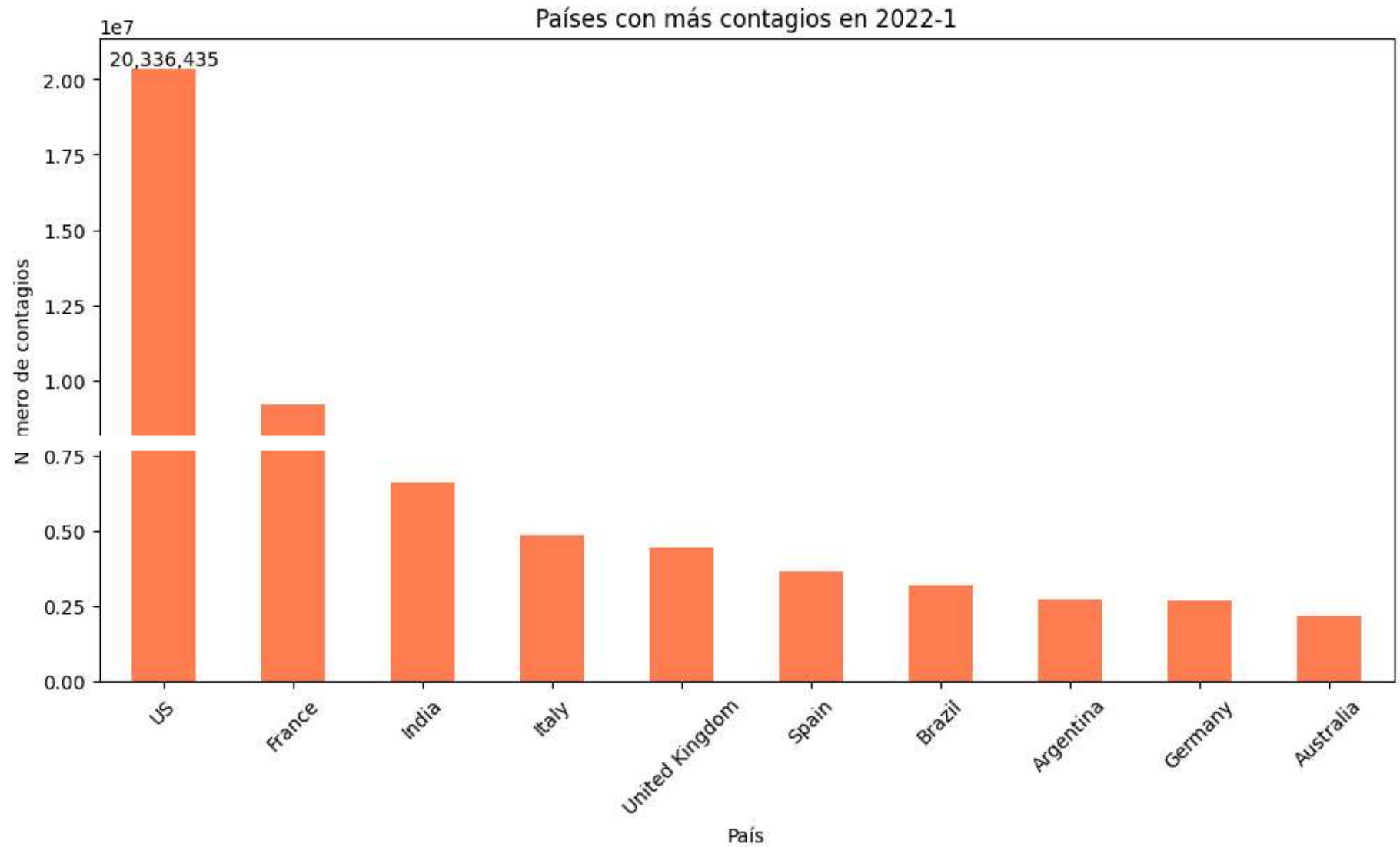
```
# Gráfico de los países con más casos en el mes pico
plt.figure(figsize=(12, 6))
axMax = contagios_por_pais_mes_max.sort_values(ascending=False).head(10).plot(kind='bar', color='coral')
plt.title(f'Países con más contagios en {ano_max}-{mes_max}')
plt.xlabel('País')
plt.ylabel('Número de contagios')
plt.xticks(rotation=45)

# Añadir el valor solo en la barra del país con más contagios
pais_max_indice = contagios_por_pais_mes_max.sort_values(ascending=False).head(10).index.get_loc(pais_max_contagios)
valor_pais_max = contagios_por_pais_mes_max[pais_max_contagios].sum()

# Mostrar el valor en la barra correspondiente
axMax.text(pais_max_indice, valor_pais_max + 1000, f'{valor_pais_max:,.0f}', ha='center', va='bottom', fontweight='bold')
```



```
plt.show()
```



¿Cuál es el país con el menor número de casos reportados hasta la fecha?

```
# 3. Encontrar el país con el menor número de casos reportados hasta la fecha
total_casos_por_pais = df_consolidado.groupby('Country/Region')['Casos confirmados'].max()

# Identificar el país con el menor número de casos reportados
pais_min_casos = total_casos_por_pais.idxmin()
print(f"El país con el menor número de casos reportados hasta la fecha es {pais_min_casos}, con {total_casos_por_pais[pais_min_casos]}
```

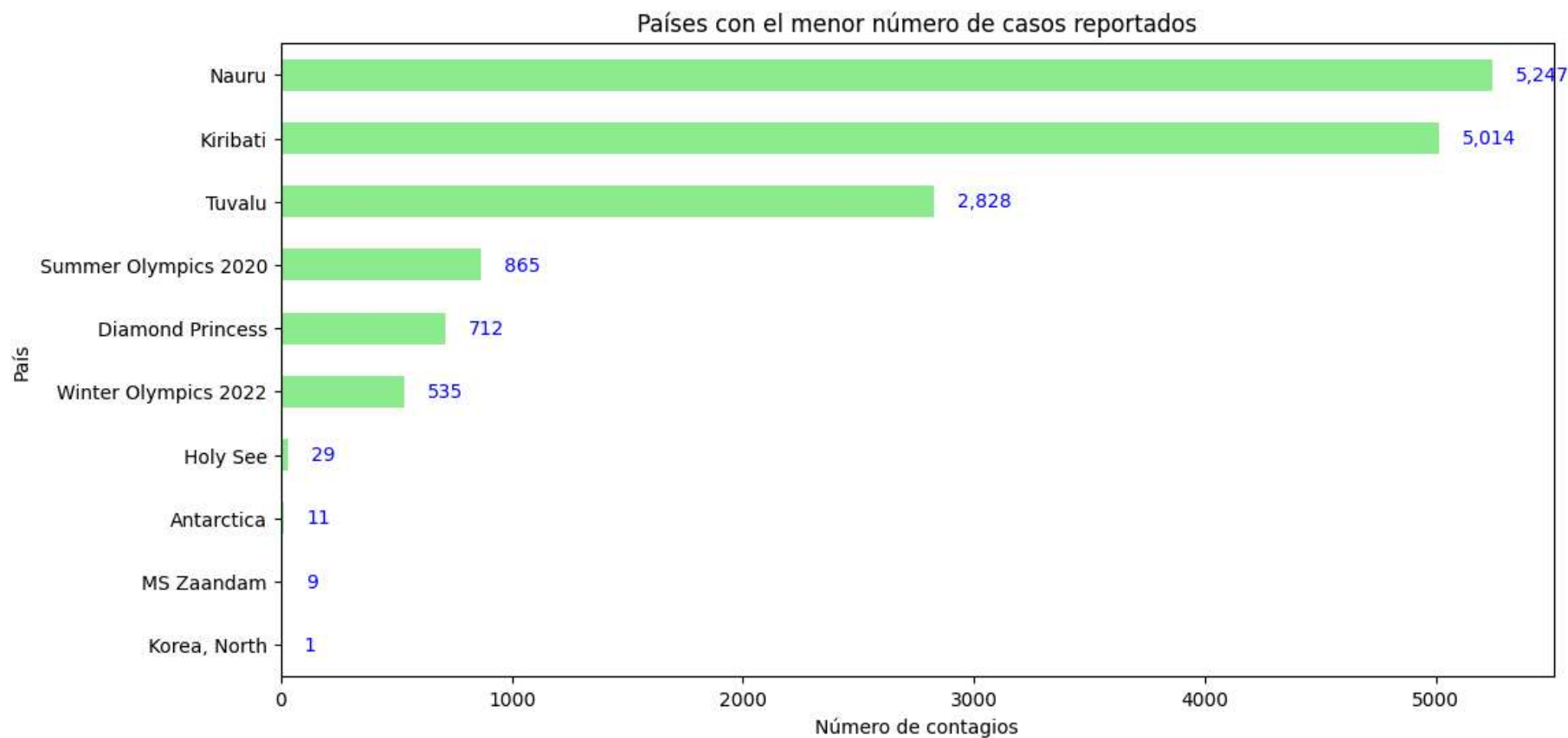
➡ El país con el menor número de casos reportados hasta la fecha es Korea, North, con 1 casos.

Adicionalmente, el gráfico de los países con menos casos acumulados:

```
# Gráfico de los países con menos casos acumulados
plt.figure(figsize=(12, 6))
ax = total_casos_por_pais.nsmallest(10).plot(kind='barh', color='lightgreen')
plt.title('Países con el menor número de casos reportados')
plt.xlabel('Número de contagios')
plt.ylabel('País')

# Añadir los valores en cada barra
for i in range(len(total_casos_por_pais.nsmallest(10))):
    valor = total_casos_por_pais.nsmallest(10).iloc[i]
    ax.text(valor + 100, i, f'{valor:,.0f}', va='center', fontsize=10, color='blue')

plt.show()
```



Conclusión

La limpieza y preparación de datos es una tarea esencial en cualquier proyecto de análisis. Python, junto con sus librerías, proporciona un entorno poderoso y flexible para realizar estas tareas. Al aplicar estos pasos y utilizar las herramientas adecuadas, se pueden obtener datos altos en calidad y realizar análisis más precisos y confiables.

Pandas es una herramienta invaluable para trabajar con datos en un Data Lake, que es un repositorio de datos de bajo costo que almacena datos sin procesar, es decir en su formato original, dichos datos pueden estar estructurados o no y no requieren esquema de lectura, lo cual les permite flexibilidad a los científicos de datos a la hora de implementar proyectos de Machine Learning.

Pandas al proporcionar un conjunto completo de funcionalidades para la manipulación, análisis y visualización de datos, simplifica el proceso de extracción de valor de los datos almacenados en un Data Lake.

Pandas ayuda a:

- Cargar y guardar datos de forma eficiente.
- Limpiar y transformar los datos para análisis.
- Realizar análisis exploratorio de datos.
- Preparar los datos para modelos de machine learning.

Al combinar la potencia de Pandas con la capacidad de almacenamiento de un Data Lake, se pueden construir soluciones de análisis de datos robustas y escalables.

Ventajas de usar Pandas con Data Lakes:

- Escalabilidad: Pandas puede manejar grandes conjuntos de datos almacenados en Data Lakes.
- Flexibilidad: Ofrece una amplia gama de herramientas para manipular y analizar datos.
- Integración: Se integra fácilmente con otras herramientas de análisis de datos y machine learning.
- Eficiencia: Pandas está optimizado para realizar operaciones en DataFrames de forma eficiente.

En resumen, Pandas proporciona un conjunto de herramientas poderosas para limpiar y transformar datos antes de realizar análisis