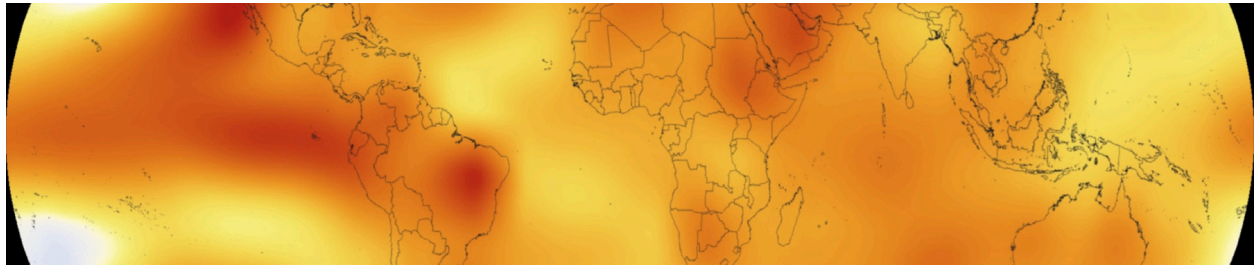


INFORME PROYECTO BEDU



Proyecto a cargo de:

Hannia Estefani Chable Gomez	hannia.chablegom@gmail.com
Frida Alejandra Perez Xequib	fridaperez3004@gmail.com
Gemma Isela Castañeda Hernández	gemma.castaneda9048@alumnos.udg.mx
Gemma Melanie Sánchez Herrera	gemma.sanchez0611@gmail.com
Gretel Dairen Zavaleta Moctezuma	greteldairen@gmail.com
Gladis Lucero Rodríguez Sánchez	gladisrosa02@hotmail.com

Descripción del problema

El cambio climático representa uno de los desafíos globales más críticos en la era tecnológica actual, siendo el incremento histórico de la temperatura global la evidencia más contundente de este fenómeno. Este proyecto se enfoca en analizar la disparidad potencial en el impacto del calentamiento global.

Relevancia del problema

Equidad y Justicia Climática: La pregunta central aborda directamente un tema de justicia climática. Determinar si existe una disparidad en la tasa de calentamiento entre naciones ricas y pobres es crucial, ya que los países de bajos ingresos a menudo tienen menor capacidad financiera e infraestructural para enfrentar los impactos adversos del clima.

Evaluación de Riesgos para el Sector Privado: Empresas en sectores vulnerables (seguros, energía, agricultura) requieren esta información para realizar una evaluación de riesgos informada, planificar operaciones a largo plazo y desarrollar estrategias de sostenibilidad corporativa.

Concienciación Pública: La traducción de estos datos complejos permite a ONGs y Medios de Comunicación elevar la conciencia pública sobre la urgencia del problema y la necesidad de una acción global coordinada.

Objetivos

Objetivo principal

- Evaluar y cuantificar la diferencia en la tendencia de calentamiento entre países de altos y bajos ingresos desde el año 2000 hasta el presente, utilizando métricas de aceleración de la temperatura.

Objetivo específicos

- Identificar Extremos: Determinar cuáles son los 10 países que han registrado la mayor frecuencia de eventos de anomalía de temperatura extrema (superiores a $+2.0^{\circ}\text{C}$) en la última década (2014-2023).
- Cuantificar la Volatilidad: Usar la desviación estándar para identificar qué regiones del mundo exhiben la mayor variabilidad climática en sus temperaturas anuales, además del calentamiento promedio.
- Realizar un Análisis Comparativo Regional: Contrastar la curva de calentamiento de los países más pobres contra los más ricos incluidos en el dataset para entender las dinámicas climáticas entre bloques económicos y geográficos.

Hipótesis

Existe una disparidad positiva; es decir, la tasa de aceleración del calentamiento en el siglo XXI es más alta en las naciones de bajos ingresos en comparación con las de altos ingresos.

Preguntas reflexivas sobre el proyecto

¿Por qué les gustó este proyecto?

El cambio climático es uno de los problemas más importantes que enfrenta el mundo en esta era tecnológica. La mejor prueba de ello es el cambio histórico de temperatura. Este conjunto de dataset provienen de FAOSTAT, que a su vez utiliza la base de datos GISTEMP de la NASA. Esto garantiza que estaremos trabajando con información precisa, estandarizada y reconocida a nivel mundial, lo cual es fundamental para cualquier estudio sobre el clima.

¿A quién iría dirigido este proyecto, o a quién le sería de interés?

Puede interesar a una audiencia muy variada:

- **Organismos Gubernamentales y Legisladores:** para entender el impacto del cambio climático a nivel nacional y regional, y así poder diseñar políticas públicas de mitigación y adaptación (ej. en agricultura, gestión del agua, salud pública).
- **Comunidad Científica y Académica:** investigadores y estudiantes de ciencias ambientales, geografía, economía y ciencias de datos pueden usarlo como base para estudios más profundos, validación de modelos climáticos o proyectos de investigación.
- **Sector Privado:** empresas en sectores como agricultura, seguros, energía y turismo están directamente afectadas por el cambio climático. Un análisis de estos datos puede ayudarles a evaluar riesgos, planificar operaciones y desarrollar estrategias de sostenibilidad.
- **Organizaciones No Gubernamentales (ONGs):** Grupos ambientalistas y de desarrollo pueden usar los resultados para sus campañas de concientización, informes de impacto y para presionar por acciones políticas.
- **Público General y Medios de Comunicación:** A través de visualizaciones de datos claros y atractivos (mapas de calor y/o gráficos de tendencias), se puede comunicar

la urgencia y la escala del problema del calentamiento global de una manera muy efectiva.

¿Qué pueden obtener de su base de datos?

- **Visualizaciones de Impacto:** datos y mapas que muestren qué países se están calentando más rápido, o gráficos que representen la evolución de la temperatura global desde 1961.
- **Identificación de "Puntos Calientes":** localizar geográficamente las regiones del mundo que están experimentando los cambios de temperatura más extremos y acelerados.
- **Análisis de tendencias temporales:** demostrar de forma cuantitativa si la tasa de calentamiento se ha acelerado en las últimas décadas en comparación con el siglo XX.
- **Modelos predictivos:** proyectar la tendencia de la temperatura para los próximos años, tanto a nivel mundial como por país.
- **Informes comparativos:** generar reportes que comparen la evolución del clima entre diferentes regiones económicas o geográficas (ej. países del G7 vs. países en desarrollo, hemisferio norte vs. hemisferio sur).

Preguntas Adicionales para Explorar el Dataset:

- **Análisis de Extremos:** ¿Qué 10 países han experimentado la mayor cantidad de meses con anomalías de temperatura superiores a +2.0 °C en la última década (2014-2023)?
- **Análisis Estacional:** ¿El calentamiento global es uniforme a lo largo del año? Comparando los cambios de temperatura promedio en las estaciones de verano e invierno, ¿cuál de las dos muestra una tendencia de calentamiento más pronunciada a nivel mundial?
- **Análisis de Volatilidad:** Utilizando la desviación estándar que provee el dataset, ¿qué regiones del mundo muestran no sólo un mayor calentamiento, sino también una mayor variabilidad climática en sus temperaturas anuales?

-
- **Análisis Comparativo Regional:** ¿Cómo se compara la curva de calentamiento de los países más pobres contra los más ricos incluidos en el dataset? (requiere agrupar países por situación económica).
 - **Punto de Inflexión:** ¿Es posible identificar una década específica en la que la tasa de aumento de la temperatura global ("World") mostró una aceleración estadísticamente significativa?

Dataset utilizado para el proyecto

Link del dataset de Kaggle: [Temperature change
https://www.kaggle.com/datasets/sevgisarac/temperature-change](https://www.kaggle.com/datasets/sevgisarac/temperature-change)

Análisis del Dataset

¿El conjunto de datos que tengo realmente me sirve para responder algunas de las preguntas que me planteé? Sí.

¿Qué tamaño tiene mi conjunto de datos?

- Número de Columnas (Variables): 9,657
- Número de Filas (Registros/Observaciones): 66

¿Serán datos suficientes? sí.

¿Qué columnas tengo y qué información tengo en cada una de esas columnas?

- Area Code (Código de Área): Código numérico que identifica de manera única el país o la región (ej. "2" para Afganistán, "5873" para OECD).
- Area (Área): Nombre del país o la región geográfica a la que pertenece la medición (ej. "Afghanistan", "World", "OECD").
- Months Code (Código de los meses): Código numérico que identifica el período de tiempo de la medición (ej. "7001" para Enero, "7020" para el Año Meteorológico).

-
- Months (Meses): Nombre del período de tiempo de la medición (ej. "January", "Meteorological year", "Annual"). Esta clave nos puede servir para el análisis estacional.
 - Element Code (Código del elemento): Código numérico que identifica el tipo de fenómeno o variable que se está midiendo.
 - Element (Elemento): Descripción de la variable medida (ej. "Temperature change" cambio de temperatura o "Standard Deviation" desviación estándar).
 - Unit (Unidad): Unidad de medida para los valores de temperatura, que es "°C" (Grados Celsius).
 - Y1961 hasta Y2019: Año de cada valor, inicia desde 1961 hasta el 2019. Valores numéricos que representan la anomalía de temperatura (en °C) para el año correspondiente. Es la diferencia entre la temperatura registrada en un mes o año específico y la temperatura promedio a largo plazo para esa misma ubicación y período. Significado de los valores:
 - Un valor positivo (ej. 0.5) indica que la temperatura fue 0.5°C más cálida que el promedio histórico.
 - Un valor negativo (ej. -0.2) indica que la temperatura fue 0.2°C más fría que el promedio histórico.

Los nombres que tienen mis columnas, ¿son el nombre más apropiado?

No, habría que traducirlos a nuestro idioma y quitar por ejemplo la "Y" en cada uno de los años.

¿Qué tipos de datos tengo en cada columna? ¿Parece ser el tipo correcto de datos? ¿O es un tipo de datos "incorrecto"?

Tipos de datos:

- Area Code (Código de Área): Entero (Integer)
- Area (Área): Texto (String)

-
- Months Code (Código de los meses): Entero (Integer)
 - Months (Meses): Texto (String)
 - Element Code (Código del elemento): Entero (Integer)
 - Element (Elemento): Texto (String)
 - Unit (Unidad): Texto (String)
 - Y1961 hasta Y2019: Decimal/Flotante (Float)

En general, la mayoría de los tipos de datos son correctos y apropiados para las tareas de análisis de datos que se quieren realizar. Sin embargo, hay un punto clave que requerirá una transformación simple:

- Area Code, Months Code, Element Code: Son números, pero se usan como identificadores (IDs), no para hacer cálculos. Cambiarlos a Texto/Categórico para evitar que sean incluidos accidentalmente en cálculos numéricos.
- Area, Months, Element, Unit: Correcto. Son campos descriptivos y categóricos. Mantener como Texto/Categórico. La columna Element se puede descartar si solo contiene "Temperature change".
- Y1961 hasta Y2019: puede que inicialmente se carguen como Texto (String), lo que impediría hacer cálculos (promedios, regresión, etc.). Convertir explícitamente a Flotante (Float). Para poder realizar el análisis de tendencias.

Si selecciono algunas filas al azar y las observo, ¿estoy obteniendo los datos que debería? ¿o hay datos que parecen estar "sucios" o "incorrectos"?

Aunque los datos no están intrínsecamente "incorrectos", hay dos problemas de formato y estructura que los convierten en "sucios" para el software de análisis:

- El Tipo de Dato (El Punto Decimal): El software (Python) los verá como una palabra y no como un número, impidiendo hacer cálculos como promedios o la regresión lineal.

-
- Acción de Limpieza Requerida: Debemos convertir explícitamente todas las columnas de año (Y1961 a Y2019) al tipo de dato Decimal o Flotante (Float).
 - Valores Faltantes (Ausencia de Datos): Hay países con datos faltantes y al cargar el dataset completo, es muy probable que haya celdas vacías o marcadas con códigos especiales como: "NaN" (Not a Number), "...", o celdas simplemente en blanco. Si intentamos promediar una columna con valores faltantes, el resultado será incorrecto o producirá un error.
 - Acción de Limpieza Requerida: Tendremos que decidir una estrategia para manejar estos valores. Las dos estrategias más comunes para datos faltantes son: 1. Reemplazar el valor faltante con un valor calculado (ej. la media o mediana de la columna). 2. Descartar la fila completa si el valor faltante es crítico.

Código del proyecto


Primeramente se importaron todas las librerías necesarias, **pandas** es una librería que nos sirve para manipulación y análisis de datos tabulares en este caso nuestro Dataset el cual es un archivo .csv, mientras que **numpy** nos ayuda a realizar cálculos numéricos eficiente y manejar grandes volúmenes de datos. Con ayuda del **as** le proporcionamos un *alias* a las librerías lo que nos sirve para simplificar el código y mejorar su legibilidad.

```
#primero vamos a importar las bibliotecas necesarias
import pandas as pd
import numpy as np
```

Para poder trabajar con el dataset fue necesario descargarlo de Kaggle y subir el archivo .csv a un drive. Para poder conectar el drive al entorno de Colab es necesario el **from...import drive** se utiliza para importar un módulo específico de un paquete en este caso el submódulo *drive*, el **google.colab** se refiere al paquete de colab que tiene las funciones y herramientas para el manejo de archivos, autenticación, entre otras cosas. El método **.mount()** realiza la acción de montar (conectar) cuando se ejecuta, **'/content/drive'** es la ruta (path) en el sistema de archivos virtual de Colab donde se

montará el drive. El resultado en la parte de abajo significa que el montaje (conexión) se completó con éxito.

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

La variable **df** (dataframe) va almacenar el contenido de nuestro archivo csv en memoria como un objeto dataframe, **pd** es el alias antes mencionado de pandas mientras **.read_csv()** es el método que abre el archivo e interpreta las columnas y filas convirtiéndolo en un dataframe **'/content/drive/MyDrive/tecnolochicas_bedu/Environment_Temperature_change_E_All_Data_NOFLAG.csv'** es la ruta (path) donde se encuentra el archivo, **encoding='latin-1'** evita errores de decodificación si el csv contiene caracteres especiales.


```
df = pd.read_csv('/content/drive/MyDrive/BEDU/Environment_Temperature_change_E_All_Data_NOFLAG.csv', encoding='latin-1')
```

La variable **df_copia** almacena la copia, esta copia es generada por **df.copy()** se creará pero cualquier cambio no afectará al dataframe original, es práctica estándar para mantener el código organizado y legible.

```
df_copia = df.copy()
```


El atributo **.shape** obtiene las dimensiones del Dataframe, es decir, el número de filas y columnas que lo conforman. Es clave mencionar que esto no modifica nada, si no que solo devuelve información básica.

```
df.shape #tamaño total de datos, filas y columnas
```

 (9656, 66)


El atributo **.ndim** devuelve un entero que indica el número de dimensiones del dataframe.

```
df.ndim #dimensiones de la matriz
```

 2

El método **.info()** genera y muestra un resumen informativo del dataframe sin modificarlo. La respuesta será el tipo de índice, el número de filas, una tabla detallada de cada columna, un resumen de los datos agrupados y el uso aproximado de memoria.

```
df.info() #Tipos de datos
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9656 entries, 0 to 9655
Data columns (total 66 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Area Code   9656 non-null  int64
1   Area        9656 non-null  object
2   Months Code 9656 non-null  int64
3   Months      9656 non-null  object
4   Element Code 9656 non-null  int64
5   Element     9656 non-null  object
6   Unit        9656 non-null  object
7   Y1961       8287 non-null  float64
8   Y1962       8322 non-null  float64
9   Y1963       8294 non-null  float64
10  Y1964       8252 non-null  float64
60  Y2014       8377 non-null  float64
61  Y2015       8361 non-null  float64
62  Y2016       8348 non-null  float64
63  Y2017       8366 non-null  float64
64  Y2018       8349 non-null  float64
65  Y2019       8365 non-null  float64
dtypes: float64(59), int64(3), object(4)
memory usage: 4.9+ MB
```

La primera parte del código genera una lista llamada **columnas_de_codigo** que contiene nombres de las columnas del dataframe. La otra parte del código **.astype(str)** convierte el tipo de datos de las columnas de int64(entero) a str(texto).

```
# Definir de la lista de columnas de código que queremos convertir de entero (int64) a texto (str u object en Pandas)
columnas_de_codigo = ['Area Code', 'Months Code', 'Element Code']

# Conversión a tipo 'str' (texto) en el DataFrame de análisis
df_copia[columnas_de_codigo] = df_copia[columnas_de_codigo].astype(str)
```

Se crea una lista **columnas_datos_vacios** que contiene los nombres de las columnas del dataframe **df_copia.columns[7:]** a partir de la columna 7 hasta la última, **.replace(to_replace=["", ' '], value=np.nan)** reemplaza todas las celdas vacías (") o celdas con espacios (' ') por np.nan y **regex=False** indica que no se usarán expresiones regulares para el reemplazo, lo que hace que la operación sea más rápida y simple.

```
# Definir las columnas que tienen celdas vacías (desde la 7 hasta el final)
columnas_datos_vacios = df_copia.columns[7:]

# Sustituir en la copia las celdas vacías por np.nan
df_copia[columnas_datos_vacios] = df_copia[columnas_datos_vacios].replace(to_replace=["", ' '], value=np.nan, regex=False)
```

El **df_copia.isna()** genera una matriz booleana donde True indica celdas con valores NaN y False indica celdas con valores válidos, **.sum()** suma los valores True a lo largo de cada columna, dando el total de NaN por columna, **print(f"NaNs por columna: \n{nan_por_columna}\n")** muestra el resultado con un mensaje formateado, donde \n añade saltos de línea para mejor legibilidad, mientras el **.sum(axis=1)** suma los valores True a lo largo de cada fila (el argumento axis=1 indica que la suma se realiza horizontalmente por filas).

```
# Contar NaN en cada columna
nan_por_columna = df_copia.isna().sum()
print(f'NaNs por columna: \n{nan_por_columna}\n')

print("-"*20)

# Contar NaN en cada fila
nan_por_fila = df_copia.isna().sum(axis=1)
print(f'NaNs por fila: \n{nan_por_fila}\n')
```

NaNs por columna:

Area Code	0
Area	0
Months Code	0
Months	0
Element Code	0
...	
Y2015	1295
Y2016	1308
Y2017	1290
Y2018	1307
Y2019	1291

Length: 66, dtype: int64

NaNs por fila:

0	0
1	0
2	0
3	0
4	0
..	
9651	0
9652	0
9653	0
9654	0
9655	0

Length: 9656, dtype: int64

El **df_copia.isna()** genera una matriz booleana donde True indica celdas con valores NaN y False indica celdas con valores válidos, **.sum()** suma todos los valores True por columna, **/ len(df_copia) * 100** divide la suma de NaN por el número total de filas y multiplica por 100 para obtener el porcentaje de NaN por columna. Mientras que el **print(f'% NaNs por columna: \n{porcen_nan_por_columna}\n')** imprime el porcentaje de NaN por columna con un mensaje formateado.

```
# Porcentaje de NaN en cada columna
porcen_nan_por_columna = df_copia.isna().sum() / len(df_copia) * 100
print(f'% NaNs por columna: \n{porcen_nan_por_columna}\n')
```

```
% NaNs por columna:
Area Code      0.000000
Area           0.000000
Months Code    0.000000
Months         0.000000
Element Code    0.000000
...
Y2015          13.411350
Y2016          13.545982
Y2017          13.359569
Y2018          13.535626
Y2019          13.369925
Length: 66, dtype: float64
```

En la primera parte del código se crea un nuevo dataframe llamado **df_clean** el cual es una copia independiente de **df_copia**. El método **.copy()** asegura que los cambios realizados en **df_clean** no afecten el dataframe original, evitando modificaciones no deseadas por referencia. Y en la otra parte el método **.head()** muestra las primeras 5 filas del dataframe **df_clean** por defecto. Esto es útil para inspeccionar rápidamente los datos y verificar que la copia se haya realizado correctamente.

```
# Copia del dataframe para manipulaciones seguras
df_clean = df_copia.copy()
df_clean.head()
```

	Area Code	Area	Months Code	Months	Element Code	Element	Unit	Y1961	Y1962	Y1963	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y2019
0	2	Afghanistan	7001	January	7271	Temperature change	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
1	2	Afghanistan	7001	January	6078	Standard Deviation	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
2	2	Afghanistan	7002	February	7271	Temperature change	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
3	2	Afghanistan	7002	February	6078	Standard Deviation	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
4	2	Afghanistan	7003	March	7271	Temperature change	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234

5 rows x 66 columns

El método **.dropna()** elimina filas o columnas que contienen valores NaN según los parámetros especificados, **axis=0** indica que la operación se aplicará a lo largo de las filas, **how='any'** especifica que se elimina una fila si contiene al menos un valor NaN.

```
# Eliminar filas donde cualquier valor es NaN
df_clean.dropna(axis=0, how='any')
```

	Area Code	Area	Months Code	Months	Element Code	Element	Unit	Y1961	Y1962	Y1963	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y2019
0	2	Afghanistan	7001	January	7271	Temperature change	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
1	2	Afghanistan	7001	January	6078	Standard Deviation	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
2	2	Afghanistan	7002	February	7271	Temperature change	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
3	2	Afghanistan	7002	February	6078	Standard Deviation	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
4	2	Afghanistan	7003	March	7271	Temperature change	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234
...
9651	5873	OECD	7018	JunJulAug	6078	Standard Deviation	°C	0.247	0.247	0.247	...	0.247	0.247	0.247	0.247	0.247	0.247	0.247	0.247	0.247	0.247
9652	5873	OECD	7019	SepOctNov	7271	Temperature change	°C	0.036	0.461	0.665	...	0.958	1.106	0.885	1.041	0.999	1.670	1.535	1.194	0.581	1.233
9653	5873	OECD	7019	SepOctNov	6078	Standard Deviation	°C	0.378	0.378	0.378	...	0.378	0.378	0.378	0.378	0.378	0.378	0.378	0.378	0.378	0.378
9654	5873	OECD	7020	Meteorological year	7271	Temperature change	°C	0.165	-0.009	0.134	...	1.246	0.805	1.274	0.991	0.811	1.282	1.850	1.349	1.088	1.297
9655	5873	OECD	7020	Meteorological year	6078	Standard Deviation	°C	0.260	0.260	0.260	...	0.260	0.260	0.260	0.260	0.260	0.260	0.260	0.260	0.260	0.260

El método **rename()** se usa para renombrar las columnas del dataframe, **{col: col[1:] if col.startswith("Y") else col for col in df_clean.columns}** verifica si la columna empieza con "Y" se renombra todos los caracteres excepto el primero, si no permanece igual, **inplace=True** indica que los cambios se aplican directamente a df_clean. El **head()** muestra las primeras filas del dataframe.

```
#Eliminar la "Y" en los años
df_clean.rename(
    columns={col: col[1:] if col.startswith("Y") else col for col in df_clean.columns},
    inplace=True
)
df_clean.head()
```

	Area Code	Area	Months Code	Months	Element Code	Element	Unit	1961	1962	1963	...	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
0	2	Afghanistan	7001	January	7271	Temperature change	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
1	2	Afghanistan	7001	January	6078	Standard Deviation	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
2	2	Afghanistan	7002	February	7271	Temperature change	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
3	2	Afghanistan	7002	February	6078	Standard Deviation	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
4	2	Afghanistan	7003	March	7271	Temperature change	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234

La variable **traduccion_encabezados** define un diccionario que mapea los nombres originales de las columnas en inglés a sus equivalentes en español, **df.rename** se usa para renombrar según la condición, **columns=traduccion_encabezados** aplica el diccionario para renombrar las columnas que coinciden con las claves del diccionario, **inplace=True**

modifica el dataframe, **df.head()** muestra las primeras 5 filas del dataframe para verificar que los nombres de las columnas se hayan cambiado según el diccionario.

```
#Traducir al español los encabezados de las columnas
traduccion_encabezados = {
    "Area Code": "Código de área",
    "Area": "Área",
    "Months Code": "Código de mes",
    "Months": "Mes",
    "Element Code": "Código de elemento",
    "Element": "Elemento",
    "Unit": "Unidad"
}

df.rename(columns=traduccion_encabezados, inplace=True)
df.head()
```

	Código de área	Área	Código de mes	Mes	Código de elemento	Elemento	Unidad	Y1961	Y1962	Y1963	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y2019
0	2	Afghanistan	7001	NaN	7271	Cambio de temperatura	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
1	2	Afghanistan	7001	NaN	6078	Desviación Estándar	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
2	2	Afghanistan	7002	NaN	7271	Cambio de temperatura	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
3	2	Afghanistan	7002	NaN	6078	Desviación Estándar	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
4	2	Afghanistan	7003	NaN	7271	Cambio de temperatura	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234

La variable **traduccion_elemento** define un diccionario que mapea valores en inglés a sus equivalentes en español, **df['Elemento']** selecciona la columna Elemento del dataframe **.replace(traduccion_elemento)** reemplaza los valores en la columna Elemento que coincidan con las claves del diccionario traduccion_elemento por sus valores correspondientes y **df.head()** muestra las primeras 5 filas del dataframe **df** para verificar que los reemplazos se hayan aplicado correctamente.

```
traduccion_elemento = {"temperature change": "Cambio de temperatura", "Standard Deviation": "Desviación Estándar"}
df["Elemento"] = df["Elemento"].replace(traduccion_elemento)
df.head()
```

	Código de área	Área	Código de mes	Mes	Código de elemento	Elemento	Unidad	Y1961	Y1962	Y1963	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y2019
0	2	Afghanistan	7001	January	7271	Cambio de temperatura	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
1	2	Afghanistan	7001	January	6078	Desviación Estándar	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
2	2	Afghanistan	7002	February	7271	Cambio de temperatura	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
3	2	Afghanistan	7002	February	6078	Desviación Estándar	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
4	2	Afghanistan	7003	March	7271	Cambio de temperatura	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234

La variable **traduccion_meses** define un diccionario que mapea los nombres de los meses en inglés a sus equivalentes en español, **df['Mes']** selecciona la columna Mes del dataframe **.replace(traduccion_meses)** reemplaza los valores en la columna Mes que coincidan con las claves del diccionario traduccion_meses por sus valores correspondientes en español y

df.head() muestra las primeras 5 filas del dataframe *df* para verificar que los reemplazos se hayan aplicado correctamente.

```
traduccion_meses = {"January": "Enero", "February": "Febrero", "March": "Marzo", "April": "Abril",
                    "May": "Mayo", "June": "Junio", "July": "Julio", "August": "Agosto",
                    "September": "Septiembre", "October": "Octubre", "November": "Noviembre", "December": "Diciembre"}
```

```
df["Mes"] = df["Mes"].replace(traduccion_meses)
df.head()
```

	Código de área	Área	Código de mes	Mes	Código de elemento	Elemento	Unidad	Y1961	Y1962	Y1963	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y2019
0	2	Afghanistan	7001	Enero	7271	Cambio de temperatura	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
1	2	Afghanistan	7001	Enero	6078	Desviación Estándar	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
2	2	Afghanistan	7002	Febrero	7271	Cambio de temperatura	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
3	2	Afghanistan	7002	Febrero	6078	Desviación Estándar	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
4	2	Afghanistan	7003	Marzo	7271	Cambio de temperatura	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234

La variable **temp_df** crea una copia del DataFrame para trabajar de manera segura, sobre esta copia primero se filtran las filas donde el código de área se encuentre entre 2 y 5873, dando como resultado un DataFrame más pequeño. Una vez filtrado se itera sobre todas las columnas de *df_copia* donde el nombre comienza con Y (aquí se refiere a las columnas de los años) y para cada columna de año crea una nueva con el mismo nombre pero terminada en **_F** (Fahrenheit), gracias a la función lambda se aplica la conversión a cada valor de la columna y finalmente imprime los resultados de 15 filas aleatorias.

```
# Convertir cambios de temperatura de Centígrados a Fahrenheit por año
temp_df = df_copia.copy()
temp_df = temp_df[(temp_df['Area Code'] >= 2) & (temp_df['Area Code'] <= 5873)].copy()
for col in [col for col in df_copia.columns if col.startswith('Y')]:
    temp_df[f'{col}_F'] = temp_df[col].apply(lambda c: c * 9/5) + 32
print(temp_df[['Area', 'Area Code', 'Months'] + [col for col in temp_df.columns if col.endswith('_F')]].sample(15))
```

9610	Non-Annex I countries	5849	December
7645	Turkmenistan	213	Jun@Jul@Aug
4674	Mauritius	137	September
6555	Saudi Arabia	194	Mar@Apr@May
1999	Cyprus	50	Mar@Apr@May

	Y1961_F	Y1962_F	Y1963_F	Y1964_F	Y1965_F	Y1966_F	Y1967_F	...	\
2597	33.2132	33.2132	33.2132	33.2132	33.2132	33.2132	33.2132	...	
562	30.2126	32.3402	33.5444	27.8978	34.6190	33.4796	33.8144	...	
3774	32.5400	32.0180	31.8506	32.0720	30.5420	32.7758	31.6364	...	
9576	31.6922	31.8200	30.5582	30.2756	33.6812	31.7948	32.3384	...	
9547	32.5364	32.5364	32.5364	32.5364	32.5364	32.5364	32.5364	...	
1327	33.0494	33.0494	33.0494	33.0494	33.0494	33.0494	33.0494	...	
234	31.7084	32.2952	32.6966	32.0180	31.9910	31.9280	31.9424	...	
1391	32.8856	32.8856	32.8856	32.8856	32.8856	32.8856	32.8856	...	
3864	32.8064	34.3886	32.0612	32.5256	33.2258	33.9710	32.8028	...	
2178	33.2528	30.8948	32.8604	32.3168	30.0830	32.1764	32.3276	...	
9610	31.5356	32.4734	32.3456	31.2530	31.6688	31.4672	31.1162	...	
7645	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
4674	32.0252	32.8280	31.0694	31.4510	32.1818	33.2438	31.9406	...	
6555	33.2438	33.2438	33.2438	33.2438	33.2438	33.2438	33.2438	...	
1999	33.1466	33.1466	33.1466	33.1466	33.1466	33.1466	33.1466	...	

Este código identifica los países o regiones que registraron un cambio de temperatura mayor a 6°C en 2019. Además, mediante una función lambda, determina en qué otros años cada país superó este mismo umbral, guardando esos años en una nueva columna llamada **High_Change**. Finalmente, muestra un extracto con las columnas clave para visualizar los resultados del filtrado.

```
# Filtrado países con cambio de temperatura mayor a 6°C en 2019 usando una función lambda
temp_changes = df_clean[df_clean['2019'] > 6].copy()
temp_changes['High_Change'] = temp_changes.apply(lambda row: [col for col in df_clean.columns if col.isdigit() and row[col] > 6], axis=1)
print(temp_changes[['Área', 'Mes', 'Elemento', '2019']])
```

	Área	Mes	Elemento	2019
648	Belarus	Febrero	Cambio de temperatura	6.584
2450	Estonia	Febrero	Cambio de temperatura	6.487
4082	Latvia	Febrero	Cambio de temperatura	6.413
4286	Lithuania	Febrero	Cambio de temperatura	6.465
7180	Svalbard and Jan Mayen Islands	Abril	Cambio de temperatura	7.215

Este código analiza los datos de temperatura de cada país en la última década (2010-2019). Calcula el promedio de esos 10 años y clasifica cada país como “extremo” (SI) si la media supera los 1.0 °C, o “NO” en caso contrario. El resultado se almacena en una nueva columna llamada **Decada_Reciente_Extrema**, facilitando la identificación de regiones con incrementos de temperatura particularmente elevados.

```
# Usamos 'apply' que es la versión de Pandas de 'map'.
# Función Lambda: verifica si la media de los 10 años más recientes (2010-2019) es mayor a 1.0 °C y crea una nueva columna de SI o NO. Recibe la fila (x) y calcula el promedio de las columnas de 2010 a 2019 fila por fila.
df_anual['Decada_Reciente_Extrema'] = df_anual.apply(lambda x: 'SI' if x[['2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019']].mean() > 1.0 else 'NO', axis=1)
print("\n--- Clasificación de Países Extremos Creada ---")
print(df_anual[['Área', 'Decada_Reciente_Extrema']].head(10))
```

```
--- Clasificación de Países Extremos Creada ---
Área Decada_Reciente_Extrema
32 Afghanistan SI
33 Afghanistan NO
66 Albania SI
67 Albania NO
100 Algeria SI
101 Algeria NO
134 American Samoa SI
135 American Samoa NO
168 Andorra SI
169 Andorra NO
```

Este código procesa los datos de cambio de temperatura por área y año. Primero convierte los códigos de área a valores numéricos y filtra únicamente las filas correspondientes al elemento “Cambio de temperatura” dentro de un rango válido de códigos. Luego, identifica las columnas de años (1961-2019) y calcula, para cada fila (área-mes), la suma total de cambios de temperatura registrados. Finalmente, agrupa la información por área y obtiene el cambio total acumulado por cada región, generando un resumen consolidado en el DataFrame **area_totals**.

```

df_clean['Código de área'] = pd.to_numeric(df_clean['Código de área'], errors='coerce')

# Filtrar por Cambio de temperatura en todas las áreas (Usando como referencia los Código de área)
temp_df = df_clean[df_clean['Elemento'] == 'Cambio de temperatura'].copy()
temp_df = temp_df[(temp_df['Código de área'] >= 2) & (temp_df['Código de área'] <= 5873)].copy()

# Identificar columnas de años
year_columns = [col for col in temp_df.columns if col.isdigit() and 1960 < int(col) <= 2019]

# Función para sumar valores de las columnas de años por fila (mes y área)
temp_df['Cambio_Total_por_Mes'] = temp_df.apply(
    lambda fila: reduce(lambda x, y: x + (y if pd.notna(y) else 0), [fila[col] for col in year_columns]),
    axis=1
)

# Sumar el Cambio_Total_por_Mes a lo largo de todos los años por área
def sum_reduce(series):
    return reduce(lambda x, y: x + (y if pd.notna(y) else 0), series, 0)

area_totals = temp_df.groupby(['Área', 'Código de área'])['Cambio_Total_por_Mes'].apply(sum_reduce).reset_index()
area_totals.rename(columns={'Cambio_Total_por_Mes': 'Cambio_Total_por_Área'}, inplace=True)

print(area_totals.head(20))

```

	Área	Código de área	Cambio_Total_por_Área
0	Afghanistan	2	436.238
1	Africa	5100	493.411
2	Albania	3	486.623
3	Algeria	4	716.772
4	American Samoa	5	395.732
5	Americas	5200	485.926
6	Andorra	6	699.507
7	Angola	7	415.385
8	Anguilla	258	258.730
9	Annex I countries	5848	605.546
10	Antarctica	30	159.057
11	Antigua and Barbuda	8	250.776
12	Argentina	9	266.328
13	Aruba	22	390.277
14	Asia	5300	475.665
15	Australia	10	438.000
16	Australia and New Zealand	5501	432.348