



4.4 Resumen


Resumen de los métodos aprendidos.

Método	Descripción	Ejemplo en Python
<code>concat</code>	Combina DataFrames a lo largo de un eje específico.	<pre>result = pd.concat([df1, df2])</pre>
<code>merge</code>	Combina DataFrames utilizando columnas comunes.	<pre>result = pd.merge(df1, df2, on='clave')</pre>
<code>join</code>	Combina DataFrames utilizando índices o columnas.	<pre>result = df1.join(df2, on='clave')</pre>
<code>rename</code>	Cambia los nombres de columnas o índices.	<pre>df.rename(columns= {'anterior': 'nuevo'}, inplace=True)</pre>
<code>set_index</code>	Establece una columna como índice del DataFrame.	<pre>df.set_index('columna')</pre>
<code>drop</code>	Elimina filas o columnas del DataFrame.	<pre>df.drop(columns= ['columna'], inplace=True)</pre>
<code>str.lower</code>	Convierte el contenido de una columna a minúsculas.	<pre>df['columna'].str.lower() ()</pre>
<code>str.split</code>	Divide una cadena en columnas separadas.	<pre>df['columna'].str.split (',',')</pre>
<code>str.replace</code>	Reemplaza una cadena o patrón en una columna.	<pre>df['columna'].str.repla ce('viejo', 'nuevo')</pre>
<code>isin</code>	Comprueba si los	<pre>df['columna'].isin(['va</pre>

<code>isin</code>	elementos están en una lista.	<code>lor1', 'valor2']])</code>
<code>between</code>	Filtra filas basadas en valores en un rango.	<code>df[df['columna'].between(5, 10)]</code>
<code>str.contains</code>	Comprueba si una columna contiene un patrón.	<code>df['columna'].str.contains('patrón')</code>

Unión de Datos

- Para unir todo en un único `csv` podemos usar
 - `pd.concat()`: es el más fácil, los nombres de las columnas se tienen que llamar igual. Los ejes:
 - `axis = 0` filas DEBAJO
 - `axis = 1` filas a la DERECHA
 - `pd.merge()`: es el más el común para añadir nuevas columnas. Podemos incluir:
 - `how`: especificamos como unimos los dataframes, puede ser `inner`, `right`, `left`...
 - `on` y `right_on/left_on`:
 - `on`: si los nombres de las columnas por las que queremos unir son iguales
 - `right_on/left_on`: si los nombres de las columnas por las que queremos unir son distintas
 - `pd.join()`: es como un merge, pero UNA DE LAS COLUMNAS POR LAS QUE QUEREMOS UNIR TIENE QUE SER UN ÍNDICE.
 -  `rsuffix` y `lsuffix`: cuando tenemos columnas que se llaman igual, pero no son las columnas que usaremos para unir los dataframes, podemos usar estos parámetros para especificar el nombre "nuevo".
 - `.reset_index()`: cuando unimos tablas que tienen los mismos índices es aconsejable hacer un `reset_index()` para que nuestros vayan del 0-n sin que se repita ninguno. 

 **NOTA** Para el concat y el `axis = 1`, cuidado que une con índices.

Filtrado de datos

- Formas de filtrar datos

- Filtrar por condición

- `df["Column"] == "Nevada"` : una serie de True y False
 - `df[df["Column"] == "Nevada"]` : filtro mi DataFrame por las filas que sean True

- Combinar condiciones con & (and), | (or), ~ (not)

- `(df["Provincia"] == "Madrid") | (df["Provincia"] == "Barcelona")` : Una Serie de True y False
 - `df[(df["Provincia"] == "Madrid") | (df["Provincia"] == "Barcelona")]` : un DataFrame filtrado
 - `df[~(df["Provincia"] == "Madrid") | (df["Provincia"] == "Barcelona"))]` : filtrado al revés

- Podemos guardar en variables los filtros:

- `filtro = (df["Provincia"] == "Madrid") | (df["Provincia"] == "Barcelona")`
 - `filtro_inverso = df[~filtro]`
 - `df["Column"].isin(values)`
 - `df["Provincia"].isin(["Madrid", "Barcelona"])`
 - `str.contains()` : filtrar con un patron de regex

- Guardar el dataset filtrado

Tenemos que hacer la asignación de variable a un nuevo DataFrame

- `df_nevada = df2[df2["Column"] == "Nevada"]`