

Resumen del capítulo: Filtrado de datos

Filtrado personalizado usando `query()` e `isin()`

Este método `query()` es llamado en un DataFrame y requiere una cadena como entrada. La cadena representa la **consulta** que quieres hacer en tu DataFrame, lo que básicamente significa que le dice a Python qué filas debe filtrar. Filtramos para que solo se seleccionen los juegos cuyas ventas en Japón fueron superiores a un millón de dólares e imprimamos solo las columnas `'name'` y `'jp_sales'`:

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')

print(df.query("jp_sales > 1")[['name', 'jp_sales']])
```

Para filtrar con `query()` basándose en comparaciones de cadenas, es necesario poner comillas alrededor de la cadena. Por ejemplo, seleccionemos solo los juegos publicados por Nintendo:

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')

print(df.query("publisher == 'Nintendo'")[['name', 'publisher']].head())
```

El método que podemos utilizar para filtrar los datos se llama `isin()`. En lugar de utilizar los operadores lógicos conocidos, `isin()` comprueba si los valores de una columna coinciden con alguno de los valores de otra matriz, como una lista o un diccionario.

Podemos utilizar una lista de consolas de videojuegos portátiles para obtener solo las filas de los juegos de una de esas consolas:

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')

handhelds = ['3DS', 'DS', 'GB', 'GBA', 'PSP']
print(df[df['platform'].isin(handhelds)][['name', 'platform']])
```

Uso de estructuras de datos externas para filtrar DataFrames

El método `query()` también ayuda a averiguar si los valores de la columna `'a'` están en la lista `our_list`.

```
our_list = [2, 5, 10]
query("a in @our_list")
```

	a	b	c
0	2	5	X
1	3	4	Y
2	10	3	Y
3	11	2	Y
4	12	1	Z

	a	b	c
0	2	5	X
2	10	3	Y

Este método, a veces en combinación con otros atributos, es aplicable no solo a listas externas, sino también a diccionarios, Series e incluso DataFrames externos:

```
print(df.query("a in @our_dict.values()"))
#comprobará si los valores de la columna a de una estructura de datos externa están presentes en los valores de nuestro diccionario
#fíjate que necesitamos usar el método values() del diccionario, ya que necesitamos buscar entre los valores del diccionario, no las claves

print(df.query("a in @our_series"))
#comprobará si los valores de la columna a de una estructura de datos externa están presentes en los valores de nuestro objeto Series

print(df.query("c in @our_df.index"))
#comprobará si los valores de la columna c de una estructura de datos externa están presentes en los índices de nuestro DataFrame
```

Reemplazo de valores con `where()`

Cuando el procesamiento de nuestros datos implica la modificación de los valores de las columnas, podemos utilizar el método `where()` para filtrar y modificar al mismo tiempo, de modo que solo cambiemos los valores bajo ciertas condiciones.

Supongamos que queremos cambiar todos los valores `'NES'` de la columna `'platform'` por el nombre completo, `'Nintendo Entertainment System'`

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')

df['platform'] = df['platform'].where(df['platform'] != 'NES', 'Nintendo Entertainment System')
print(df.iloc[:, :2].head())
```

Llamamos a `where()` en la columna `'platform'` y le pasamos dos argumentos de posición:

1. La condición lógica: `df['platform'] != 'NES'`. Como sabemos, comprueba todos los valores de la columna `'platform'` y devuelve `True` para las filas en las que `'NES'` **NO** es un valor, y `False` en caso contrario. La salida es una serie de booleanos.
2. Un nuevo valor para reemplazar los valores de `'platform'` para los que la condición lógica es `False`.

El método `where()` comprueba la condición para cada valor de la columna. Si la condición es `True`, `where()` no hace nada; si es `False`, `where()` sustituye el valor actual por el nuevo.