**Cómo lidiar con datos duplicados en Pandas**

Es muy usual que los registros de una base de datos aparezcan más de una vez, así que veamos cómo pandas puede ayudarnos a lidiar con estos casos. Para comenzar, importemos pandas y creemos un DataFrame con dos columnas y algunos datos repetidos.

**import** pandas **as** pd

df = pd.DataFrame({'a': ['w'] \* 4 + ['x'] \* 3 + ['y'] \* 2 + ['z']+['v'],

'b': [1, 1, 1, 1, 2, 2, 2, 3, 3, 4,5]})

df

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 0 | w | 1 |
| 1 | w | 1 |
| 2 | w | 1 |
| 3 | w | 1 |
| 4 | x | 2 |
| 5 | x | 2 |
| 6 | x | 2 |
| 7 | y | 3 |
| 8 | y | 3 |
| 9 | z | 4 |
| 10 | v | 5 |

Para encontrar los registros duplicados usamos duplicated , que marca con True aquellos casos de filas duplicadas:

df.duplicated()

|  |  |
| --- | --- |
|  |  |
| 0 | False |
| 1 | True |
| 2 | True |
| 3 | True |
| 4 | False |
| 5 | True |
| 6 | True |
| 7 | False |
| 8 | True |
| 9 | False |
| 10 | False |

Podemos usar keep='first' para marcar solo la primera ocurrencia o keep='last' para marcar la última:

df.duplicated(keep='first')

|  |  |
| --- | --- |
|  |  |
| 0 | False |
| 1 | True |
| 2 | True |
| 3 | True |
| 4 | False |
| 5 | True |
| 6 | True |
| 7 | False |
| 8 | True |
| 9 | False |
| 10 | False |

df.duplicated(keep='last')

|  |  |
| --- | --- |
|  |  |
| 0 | True |
| 1 | True |
| 2 | True |
| 3 | False |
| 4 | True |
| 5 | True |
| 6 | False |
| 7 | True |
| 8 | False |
| 9 | False |
| 10 | False |

Identificados los casos duplicados, podemos usar este resultado para filtrar y seleccionar aquellos que no tienen un registro duplicado:

df[~ df.duplicated()]

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 0 | w | 1 |
| 4 | x | 2 |
| 7 | y | 3 |
| 9 | z | 4 |
| 10 | v | 5 |

Si quisieras dejar el primer registro de los duplicados o el último, recuerda usar keep='first' o keep='last'. Remarco el hecho de que usé negación '~' para ver los registros no duplicados.

Y si me interesara ver cuáles son los registros duplicados, podemos usar keep=False:

df.duplicated(keep=**False**)

|  |  |
| --- | --- |
|  | 0 |
| 0 | True |
| 1 | True |
| 2 | True |
| 3 | True |
| 4 | True |
| 5 | True |
| 6 | True |
| 7 | True |
| 8 | True |
| 9 | False |
| 10 | False |

df[df.duplicated(keep=**False**)]

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 0 | w | 1 |
| 1 | w | 1 |
| 2 | w | 1 |
| 3 | w | 1 |
| 4 | x | 2 |
| 5 | x | 2 |
| 6 | x | 2 |
| 7 | y | 3 |
| 8 | y | 3 |

Por último, puedes usar el comando 'drop\_duplicates' para eliminar los duplicados. Por defecto, la función guarda el primer resultado keep='first':

df.drop\_duplicates()

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 0 | w | 1 |
| 4 | x | 2 |
| 7 | y | 3 |
| 9 | z | 4 |
| 10 | v | 5 |

Y si quieres solo borrar duplicados teniendo en cuenta una sola columna, lo puedes hacer mediante una lista nombrando las columnas donde vas a eliminar los duplicados, en este caso, ['a']:

df.drop\_duplicates(['a'],keep='last')

|  |  |  |
| --- | --- | --- |
|  | a | b |
| 3 | w | 1 |
| 6 | x | 2 |
| 8 | y | 3 |
| 9 | z | 4 |
| 10 | v | 5 |