Combinar `DataFrame`s usando `merge`

Muy a menudo, tenemos distintos `DataFrame` s que contienen información de distintas fuentes y queremos combinarlos en un único `DataFrame` .

```
# Preliminares
import pandas as pd
import numpy as np
```

Consideramos los dos `DataFrame` s:

```
clientes= pd.DataFrame(
    {'dni': ['12345678', '23456789', '34567890', '01234567'],
     'nombre': ['José', 'Pedro', 'María', 'Blanca'],
     'apellido1': ['Pérez', 'Martínez', 'Sánchez', 'Ruiz'],
     'apellido2': ['Martínez', 'Moreno', 'Meseguer', 'Torres']
    }
)
clientes
```

	dni	nombre	apellido1	apellido2
0	12345678	José	Pérez	Martínez
1	23456789	Pedro	Martínez	Moreno
2	34567890	María	Sánchez	Meseguer
3	01234567	Blanca	Ruiz	Torres

```
pedidos= pd.DataFrame(
    {'id': [10, 12, 21, 22, 24, 25, 28],
     'dni': np.repeat(['23456789','12345678', '34567890', '87654321'], repeats=[2, 3, 1, 1]),
     'id_producto': ['AAA123', 'SOX433', 'QWE000', 'SOX433', 'PII342', 'ZXY099','PII342']})
pedidos
```

	id	dni	id_producto
0	10	23456789	AAA123
1	12	23456789	SOX433
2	21	12345678	QWE000
3	22	12345678	SOX433
4	24	12345678	PII342
5	25	34567890	ZXY099
6	28	87654321	PII342

Queremos construir una única tabla, `DataFrame` que contenga para cada cliente los distintos pedidos que ha realizado.

```
clientes.merge(pedidos)
```

	dni	nombre	apellido1	apellido2	id	id_producto
0	12345678	José	Pérez	Martínez	21	QWE000
1	12345678	José	Pérez	Martínez	22	SOX433
2	12345678	José	Pérez	Martínez	24	PII342
3	23456789	Pedro	Martínez	Moreno	10	AAA123
4	23456789	Pedro	Martínez	Moreno	12	SOX433
5	34567890	María	Sánchez	Meseguer	25	ZXY099

> *`merge` ha utilizado las columnas comunes para realizar la combinación. En este caso ha usado los valores de 'dni' para casar filas de ambos `DataFrame`s.*

Por defecto, `merge` sólo preserva las filas que aparecen en ambas tablas.

Si un DNI que sí aparece en el `DataFrame` de la izquierda no aparece en el `DataFrame` de la derecha, está incluido del resultado, y viceversa.

> *Este comportamiento se controla con el argumento `how` que, por defecto, toma el valor 'inner'.*

El argumento `how` puede tomar el valor 'left'.

En este caso, todos los DNI que aparecen en el `DataFrame` de la izquierda, están incluidos en el resultado, aunque no aparezca en el `DataFrame` de la dereche, completando, en caso necesario, con `NaN`.

```
clientes.merge(pedidos, how='left')
```

	dni	nombre	apellido1	apellido2	id	id_producto
0	12345678	José	Pérez	Martínez	21.0	QWE000
1	12345678	José	Pérez	Martínez	22.0	SOX433
2	12345678	José	Pérez	Martínez	24.0	PII342
3	23456789	Pedro	Martínez	Moreno	10.0	AAA123
4	23456789	Pedro	Martínez	Moreno	12.0	SOX433
5	34567890	María	Sánchez	Meseguer	25.0	ZXY099
6	01234567	Blanca	Ruiz	Torres	NaN	NaN

> *En cambio, si `how='left'`, si un DNI aparece en el `DataFrame` de la derecha pero no en el de la izquierda, está descartado del resultado.*

Otro valor posible de `how`: 'right'

En este caso, todos los DNI que aparecen en el `DataFrame` de la derecha, están incluidos en el resultado, aunque no aparezca en el `DataFrame` de la izquierda, completando, en caso necesario, con `NaN`.

```
clientes.merge(pedidos, how='right')
```

	dni	nombre	apellido1	apellido2	id	id_producto
0	23456789	Pedro	Martínez	Moreno	10	AAA123
1	23456789	Pedro	Martínez	Moreno	12	SOX433
2	12345678	José	Pérez	Martínez	21	QWE000
3	12345678	José	Pérez	Martínez	22	SOX433
4	12345678	José	Pérez	Martínez	24	PII342
5	34567890	María	Sánchez	Meseguer	25	ZXY099
6	87654321	NaN	NaN	NaN	28	PII342

El dni 87654321 está en el `DataFrame` de la derecha pero no en el de la izquierda.

Resumen visual: combinar `DataFrame`s con `merge`

Los dos `DataFrame`s: las columnas con colores representan las claves sobre las que se van a hacer el `merge`.

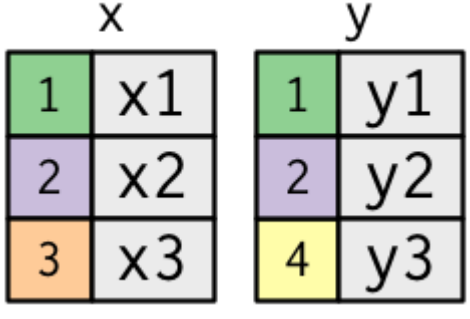
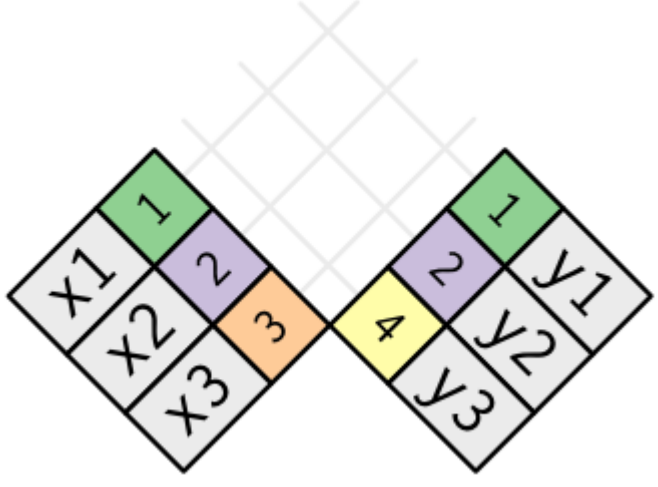


Imagen extraida de "R for Data Science", G. Grolemund & H. Wickham, capítulo 13, https://r4ds.had.co.nz/. Las cuatro imágenes que siguen están extraidas de esa misma fuente.

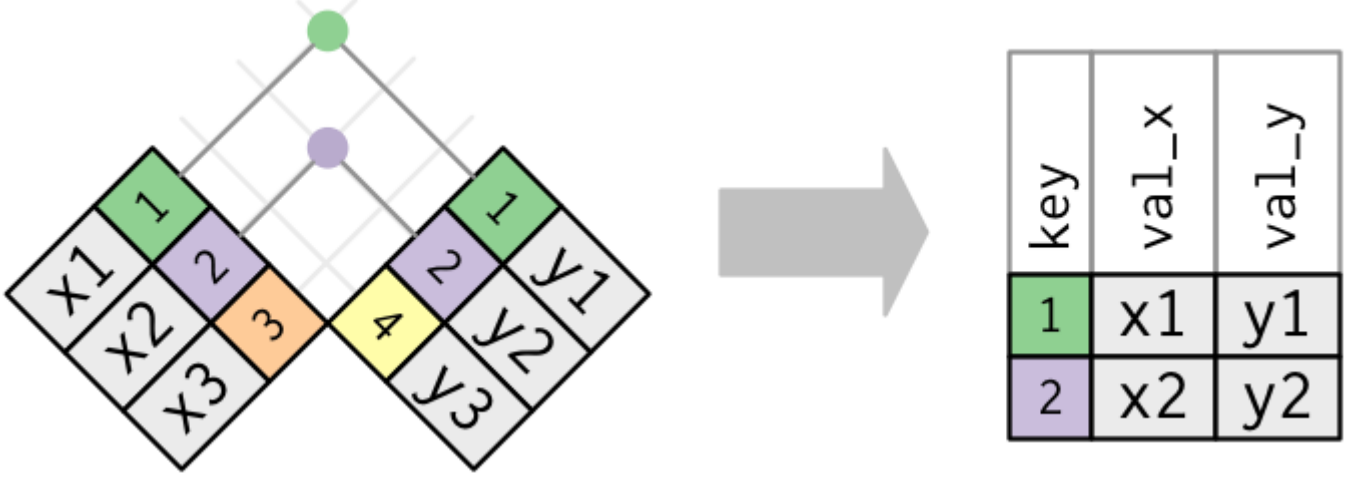
Para casar filas:



> *Imagen extraida de "R for Data Science", G. Grolemund & H. Wickham, capítulo 13, https://r4ds.had.co.nz/*

`how='inner'`

Sólo se preservan las claves que están tanto en el `DataFrame` de la izquierda como en el `DataFrame` de la derecha.



> *Imagen extraida de "R for Data Science", G. Grolemund & H. Wickham, capítulo 13, https://r4ds.had.co.nz/*

`how='left'`

- Se preservan todas las claves que están en el `DataFrame` de la izquierda, completando las claves del `DataFrame` de la derecha si es necesario.
- Se descartan las claves del `DataFrame` de la derecha que no están en el `DataFrame` de la izquierda.

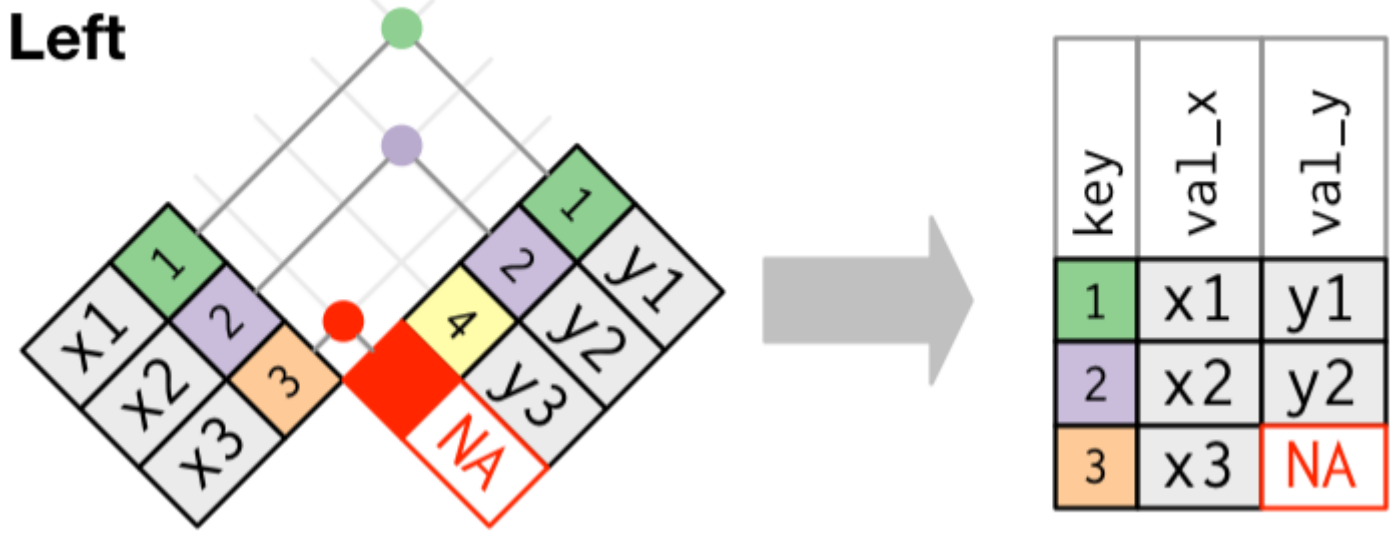


Imagen extraida de "R for Data Science", G. Grolemund & H. Wickham, capítulo 13, https://r4ds.had.co.nz/

`how='right'`

- Se preservan todas las claves que están en el `DataFrame` de la derecha, completando las claves del `DataFrame` de la izquierda si es necesario.
- Se descartan las claves del `DataFrame` de la izquierda que no están en el `DataFrame` de la derecha.

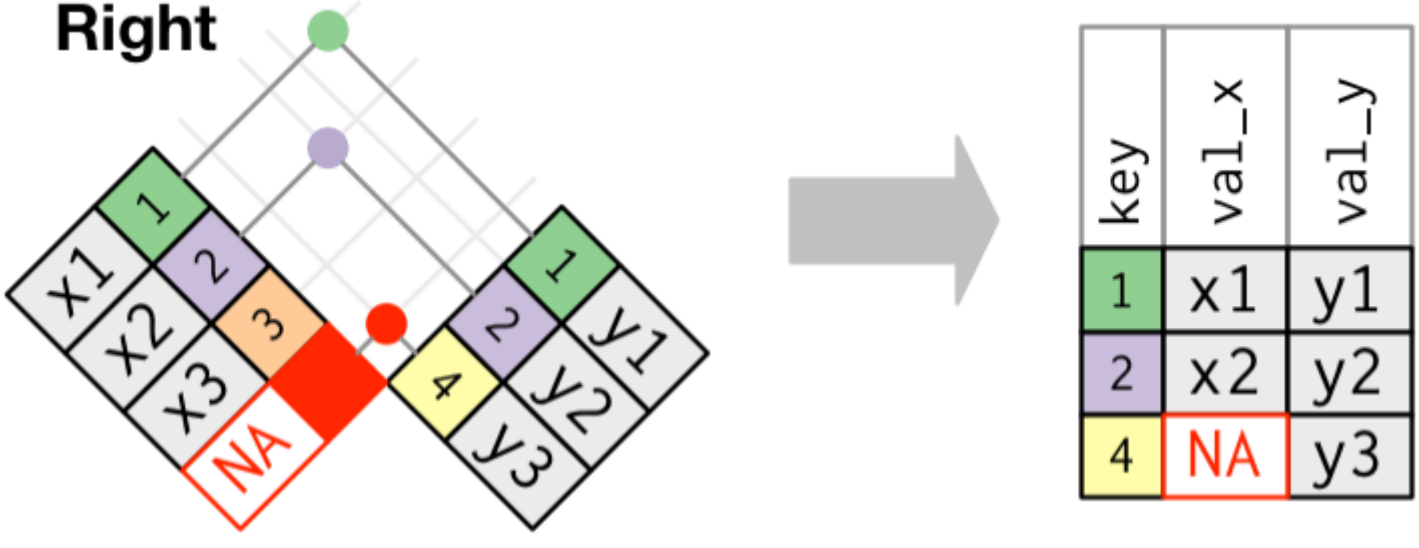


Imagen extraida de "R for Data Science", G. Grolemund & H. Wickham, capítulo 13, https://r4ds.had.co.nz/

`how='outer'`

- Se preservan todas las claves que están en el `DataFrame` de la izquierda y todas las claves del `DataFrame` de la derecha, completando con `NaN` si es necesario.

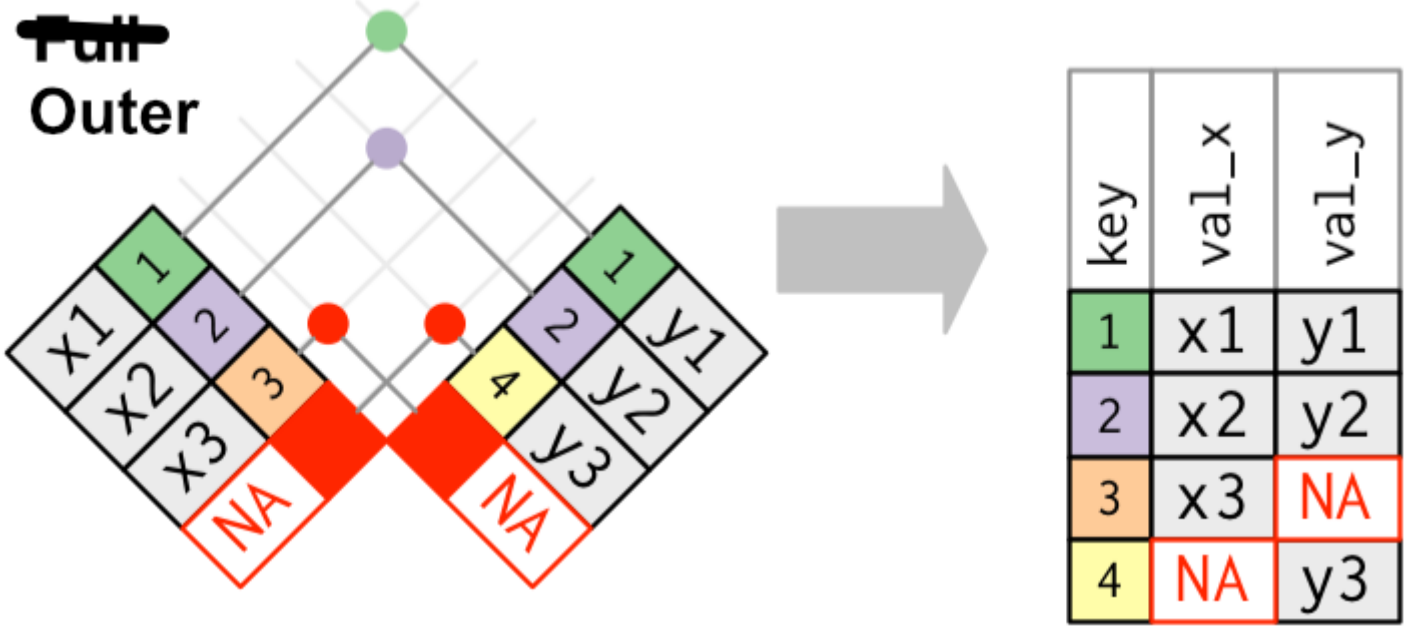


Imagen extraida de "R for Data Science", G. Grolemund & H. Wickham, capítulo 13, https://r4ds.had.co.nz/

NOTA: para todas las modalidades de `merge`, `inner`, `left`, `right` y `outer`

> *Se hace un producto cartesiano: si un valor de la clave se repite en un `DataFrame`, se incluyen combinaciones de las filas.*

Es lo que observamos con `clientes` y `pedidos`, el DNI 123456798 se repite tres veces en el `DataFrame` de pedidos, por lo tanto se combinan tres veces los datos del cliente 12345678 con sus pedidos.

```
clientes.merge(pedidos)
```

	dni	nombre	apellido1	apellido2	id	id_producto
0	12345678	José	Pérez	Martínez	21	QWE000
1	12345678	José	Pérez	Martínez	22	SOX433
2	12345678	José	Pérez	Martínez	24	PII342
3	23456789	Pedro	Martínez	Moreno	10	AAA123
4	23456789	Pedro	Martínez	Moreno	12	SOX433
5	34567890	María	Sánchez	Meseguer	25	ZXY099

Usar el parámetro `on`, `left_on` y `right_on` en `merge` para especificar qué columnas se deben usar para casar filas al combinar los dos `DataFrames`

Vimos que, por defecto, `merge` usa las columnas comunes en ambos `DataFrame`s para casar las filas

Consideramos los dos `DataFrame` s:

```
clientes= pd.DataFrame(
    {'dni': ['12345678', '23456789', '34567890', '01234567'],
     'nombre': ['José', 'Pedro', 'María', 'Blanca'],
     'apellido1': ['Pérez', 'Martínez', 'Sánchez', 'Ruiz'],
     'apellido2': ['Martínez', 'Moreno', 'Meseguer', 'Torres']
    }
)
clientes
```

	dni	nombre	apellido1	apellido2
0	12345678	José	Pérez	Martínez
1	23456789	Pedro	Martínez	Moreno
2	34567890	María	Sánchez	Meseguer
3	01234567	Blanca	Ruiz	Torres

```
pedidos= pd.DataFrame(
    {'id': [10, 12, 21, 22, 24, 25, 28],
     'dni': np.repeat(['23456789','12345678', '34567890', '87654321'], repeats=[2, 3, 1, 1]),
     'id_producto': ['AAA123', 'SOX433', 'QWE000', 'SOX433', 'PII342', 'ZXY099','PII342']})
pedidos
```

	id	dni	id_producto
0	10	23456789	AAA123
1	12	23456789	SOX433
2	21	12345678	QWE000
3	22	12345678	SOX433
4	24	12345678	PII342
5	25	34567890	ZXY099
6	28	87654321	PII342

Consideramos ahora un `DataFrame` adicional

```
productos = pd.DataFrame({
        'id': ['AAA123', 'SOX433', 'QWE000', 'PII342', 'ZXY099'],
        'nombre': ['Pila', 'Bombilla', 'Interruptor', 'Enchufe', 'Toma']})
productos
```

	id	nombre
0	AAA123	Pila
1	SOX433	Bombilla
2	QWE000	Interruptor
3	PII342	Enchufe
4	ZXY099	Toma

> *Queremos añadir la información de cada producto al `DataFrame` de los pedidos de cada cliente.*

Por defecto, `merge` usa las columnas en común

Creemos el `DataFrame` de clientes_pedidos

```
clientes_pedidos = clientes.merge(pedidos, how='left')
clientes_pedidos
```

	dni	nombre	apellido1	apellido2	id	id_producto
0	12345678	José	Pérez	Martínez	21.0	QWE000
1	12345678	José	Pérez	Martínez	22.0	SOX433
2	12345678	José	Pérez	Martínez	24.0	PII342
3	23456789	Pedro	Martínez	Moreno	10.0	AAA123
4	23456789	Pedro	Martínez	Moreno	12.0	SOX433
5	34567890	María	Sánchez	Meseguer	25.0	ZXY099
6	01234567	Blanca	Ruiz	Torres	NaN	NaN

Intentamos ahora combinar `clientes_pedidos` con la información de productos

> *En este caso, intentará casar las columnas `nombre` y `id`*

```
clientes_pedidos.merge(productos)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/dg/266f71_10dzb9tqnr41_jhk00000gn/T/ipykernel_20868/2597250559.py in <module>
----> 1 clientes_pedidos.merge(productos)

/opt/homebrew/lib/python3.9/site-packages/pandas/core/frame.py in merge(self, right, how, on, left_on, right_on, left_index, right_index, sort, suffixes, copy, indicator, validate)
   9188         from pandas.core.reshape.merge import merge
   9189
```

```
-> 9190         return merge(
   9191             self,
   9192             right,

/opt/homebrew/lib/python3.9/site-packages/pandas/core/reshape/merge.py in merge(left, right, how, on, left_on, right_on, left_index, right_index, sort, suffixes, copy, indicator, validate)
    104     validate: str | None = None,
    105 ) -> DataFrame:
--> 106     op = _MergeOperation(
    107         left,
    108         right,

/opt/homebrew/lib/python3.9/site-packages/pandas/core/reshape/merge.py in __init__(self, left, right, how, on, left_on, right_on, axis, left_index, right_index, sort, suffixes, copy, indicator, validate)
    701         # validate the merge keys dtypes. We may need to coerce
    702         # to avoid incompatible dtypes
--> 703         self._maybe_coerce_merge_keys()
    704
    705         # If argument passed to validate,

/opt/homebrew/lib/python3.9/site-packages/pandas/core/reshape/merge.py in _maybe_coerce_merge_keys(self)
   1254                     inferred_right in string_types and
```

```
 inferred_left not in string_types
   1255                 ):
-> 1256                     raise ValueError(msg)
   1257
   1258             # datetimelikes must match exactly

ValueError: You are trying to merge on float64 and object colum
ns. If you wish to proceed you should use pd.concat
```

El error que devuelve es porque la columna `id` en los dos `DataFrame`s son de diferentes tipos. No se pueden unir usando dos columnas que no sean del mismo tipo.

```
# Comprobamos tipos
print(clientes_pedidos.dtypes)
print(productos.dtypes)
```

```
dni              object
nombre           object
apellido1        object
apellido2        object
id              float64
id_producto      object
dtype: object
id        object
nombre    object
dtype: object
```

> *`id` es de tipo `float64` en `clientes_pedidos` y de tipo `str` (string) en `productos`.*

Cambiamos el tipo de `id` en `clientes_pedidos`:

```
clientes_pedidos['id'] = clientes_pedidos['id'].astype(str)
```

Volvemos a intentar el `merge`

```
clientes_pedidos.merge(productos)
```

dni	nombre	apellido1	apellido2	id	id_producto

> *Como era esperado, obtenemos ahora un `DataFrame` vacío porque las columnas comunes `id` y `nombre` no contienen información sobre los mismos elementos y no tienen valores en común.*

Para arreglar este problema:

Para empezar, vamos a cambiar los nombres de las columnas conflictivas de `productos`

```
productos.rename(columns={'nombre': 'nombre_producto'}, inplace=True)
productos
```

	id	nombre_producto
0	AAA123	Pila
1	SOX433	Bombilla
2	QWE000	Interruptor
3	PII342	Enchufe
4	ZXY099	Toma

Ahora vamos a usar los argumentos `left_on` y `right_on` para especificar sobre qué columna del `DataFrame` de la izquierda y qué columna del `DataFrame` de la derecha nos vamos a basar para casar filas

```
clientes_pedidos.merge(productos, left_on='id_producto', right_on='id')
```

	dni	nombre	apellido1	apellido2	id_x	id_producto	id_y	nombre_producto
0	12345678	José	Pérez	Martínez	21.0	QWE000	QWE000	Interruptor
1	12345678	José	Pérez	Martínez	22.0	SOX433	SOX433	Bombilla
2	23456789	Pedro	Martínez	Moreno	12.0	SOX433	SOX433	Bombilla
3	12345678	José	Pérez	Martínez	24.0	PII342	PII342	Enchufe
4	23456789	Pedro	Martínez	Moreno	10.0	AAA123	AAA123	Pila
5	34567890	María	Sánchez	Meseguer	25.0	ZXY099	ZXY099	Toma

`merge` ha añadido sufijos a los nombres de las columnas comunes para diferenciarlas, podríamos haber especificado nuestros propios sufijos con el parámetro `suffixes`

```
clientes_pedidos.merge(productos, left_on='id_producto', right_on='id', suffixes=['_pedido','_producto'])
```

	dni	nombre	apellido1	apellido2	id_pedido	id_producto	id_producto	nombre_producto
0	12345678	José	Pérez	Martínez	21.0	QWE000	QWE000	Interruptor
1	12345678	José	Pérez	Martínez	22.0	SOX433	SOX433	Bombilla
2	23456789	Pedro	Martínez	Moreno	12.0	SOX433	SOX433	Bombilla
3	12345678	José	Pérez	Martínez	24.0	PII342	PII342	Enchufe
4	23456789	Pedro	Martínez	Moreno	10.0	AAA123	AAA123	Pila
5	34567890	María	Sánchez	Meseguer	25.0	ZXY099	ZXY099	Toma

Usar el parámetro `indicator` para añadir una columna con la procedencia de cada fila

Consideramos los dos `DataFrame` s:

```
clientes= pd.DataFrame(
    {'dni': ['12345678', '23456789', '34567890', '01234567'],
     'nombre': ['José', 'Pedro', 'María', 'Blanca'],
     'apellido1': ['Pérez', 'Martínez', 'Sánchez', 'Ruiz'],
     'apellido2': ['Martínez', 'Moreno', 'Meseguer', 'Torres']
    }
)
clientes
```

	dni	nombre	apellido1	apellido2
0	12345678	José	Pérez	Martínez
1	23456789	Pedro	Martínez	Moreno
2	34567890	María	Sánchez	Meseguer
3	01234567	Blanca	Ruiz	Torres

```
pedidos= pd.DataFrame(
    {'id': [10, 12, 21, 22, 24, 25, 28],
     'dni': np.repeat(['23456789','12345678', '34567890', '87654321'], repeats=[2, 3, 1, 1]),
     'id_producto': ['AAA123', 'SOX433', 'QWE000', 'SOX433', 'PII342', 'ZXY099','PII342']})
pedidos
```

	id	dni	id_producto
0	10	23456789	AAA123
1	12	23456789	SOX433
2	21	12345678	QWE000
3	22	12345678	SOX433
4	24	12345678	PII342
5	25	34567890	ZXY099
6	28	87654321	PII342

Al usar `indicator=True`, `merge` añadirá una columna llamada `_merge` al resultado, con tres valores posibles

- "both": en el caso en el valor de la clave está en ambos `DataFrame`s
- "left_only": si el valor de la clave aparece en el `DataFrame` de la izquierda pero no en el de la derecha
- "right_only": si el valor de la clave aparece en el `DataFrame` de la derecha pero

```
clientes.merge(pedidos, how='outer', indicator=True)
```

	dni	nombre	apellido1	apellido2	id	id_producto	_merge
0	12345678	José	Pérez	Martínez	21.0	QWE000	both
1	12345678	José	Pérez	Martínez	22.0	SOX433	both
2	12345678	José	Pérez	Martínez	24.0	PII342	both
3	23456789	Pedro	Martínez	Moreno	10.0	AAA123	both
4	23456789	Pedro	Martínez	Moreno	12.0	SOX433	both
5	34567890	María	Sánchez	Meseguer	25.0	ZXY099	both
6	01234567	Blanca	Ruiz	Torres	NaN	NaN	left_only
7	87654321	NaN	NaN	NaN	28.0	PII342	right_only

Podríamos hacer el recuento de esta columna para obtener información sobre qué ha pasado en el merge

```
clientes.merge(pedidos, how='outer', indicator=True)['_merge'].value_counts()
```

```
both          6
left_only     1
right_only    1
Name: _merge, dtype: int64
```