SESIÓN AGRUPAMIENTO, PIVOTEO Y COMBINACIÓN DE DATOS

CONTENIDOS:

- Agrupamiento con indexación jerárquica.
- Agrupamiento de datos con la función groupby.
- Pivoteo de dataframes.
- Despivoteo de dataframes (método melt).
- Combinación y merge de datos:
    - Concatenación de dataframes.
    - Merge de dataframes

AGRUPAMIENTO CON INDEXACIÓN JERÁRQUICA

La indexación jerárquica (o MultiIndex) permite tener múltiples niveles de índices en un DataFrame, lo que facilita la organización y manipulación de datos complejos, especialmente cuando se trabaja con datos que tienen una estructura multinivel o multidimensional.

Creación de un DataFrame con Indexación Jerárquica

Ejemplo:

```python
import pandas as pd

# Crear un DataFrame con indexación jerárquica
data = {
    'Ciudad': ['Madrid', 'Madrid', 'Barcelona', 'Barcelona'],
    'Año': [2020, 2021, 2020, 2021],
    'Ventas': [1000, 1500, 1200, 1300]
}

df = pd.DataFrame(data)

# Establecer el índice jerárquico
df.set_index(['Ciudad', 'Año'], inplace=True)

print(df)
```

Ilustración 1 Ejemplo dataframe con indexación jerárquica

Aquí se crea un diccionario data con tres columnas:

- "Ciudad": Indica la ciudad donde se registraron las ventas.
- "Año": Representa el año de las ventas.
- "Ventas": Contiene el monto de las ventas en cada ciudad y año.

Luego, el diccionario se convierte en un DataFrame llamado df.

Si imprimamos df en este punto, se vería así:

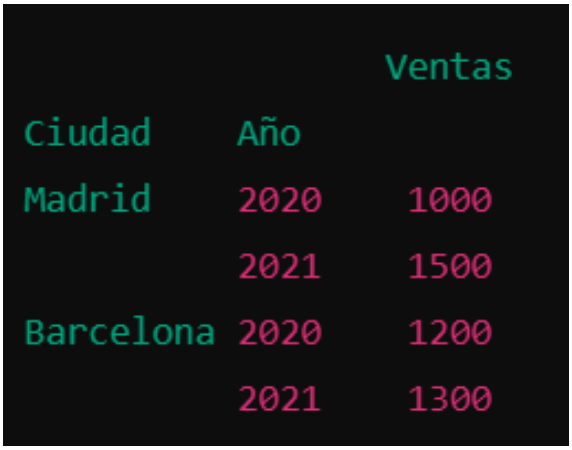
	Ciudad	**Año**	**Ventas**
0	Madrid	2020	1000
1	Madrid	2021	1500
2	Barcelona	2020	1200
3	Barcelona	2021	1300

Establecer un Índice Jerárquico

Aquí se usa .set_index(['Ciudad', 'Año']) para convertir las columnas "Ciudad" y "Año" en un índice jerárquico o multiíndice.

La opción inplace=True hace que la modificación se aplique directamente sobre el DataFrame sin necesidad de reasignarlo.

Después de esta transformación, df se verá así:



		Ventas
Ciudad	Año	
Madrid	2020	1000
	2021	1500
Barcelona	2020	1200
	2021	1300

Ilustración 2 Salida después de la transformación

Aquí:

- "Ciudad" es el nivel superior del índice.
- "Año" es el segundo nivel del índice.
- "Ventas" sigue siendo la única columna de valores.

Acceso a Datos con Indexación Jerárquica

Ejemplo:

```
# Acceder a los datos de Barcelona en 2021
print(df.loc['Barcelona', 2021])
```

Ilustración 3 Ejemplo acceso a datos con indexación jerárquica

Con este código, accedemos a los datos específicos de Barcelona en el año 2021.

AGRUPAMIENTO DE DATOS CON LA FUNCIÓN GROUPBY

La función .groupby() proviene de la biblioteca Pandas. En Pandas, groupby se basa en el concepto de dividir-aplicar-combinar (split-apply-combine), lo que permite agrupar datos según una o más columnas y aplicar funciones de agregación a esos grupos.

Agrupamiento y Agregación

Ejemplo:

```
# Agrupar por 'Ciudad' y calcular la suma de 'Ventas'
df_grouped = df.groupby('Ciudad')['Ventas'].sum()

print(df_grouped)
```

Ilustración 4 Ejemplo agrupamiento y agregación

En este ejemplo, agrupamos los datos por la columna Ciudad y luego calculamos la suma de las ventas para cada ciudad.

Funciones de Agregación Comunes

- sum(): Suma de valores.
- mean(): Media de valores.
- count(): Conteo de valores.
- max(), min(): Valor máximo y mínimo.

Ejemplo con varias funciones de agregación:

```python
# Agrupar por 'Ciudad' y calcular varias agregaciones
df_grouped = df.groupby('Ciudad')['Ventas'].agg([sum, mean, max, min])

print(df_grouped)
```

Ilustración 5 Ejemplo con varias funciones de agregación

PIVOTEO DE DATAFRAMES

El pivoteo es una técnica que transforma los datos de un formato largo a un formato ancho, reorganizando los valores en una tabla. Es útil cuando necesitamos reorganizar los datos para facilitar el análisis.

Uso de pivot_table

Ejemplo:

```python
# Crear un DataFrame de ejemplo
data = {
    'Ciudad': ['Madrid', 'Madrid', 'Barcelona', 'Barcelona'],
    'Año': [2020, 2021, 2020, 2021],
    'Ventas': [1000, 1500, 1200, 1300]
}

df = pd.DataFrame(data)

# Pivoteo con pivot_table
df_pivot = df.pivot_table(values='Ventas', index='Ciudad', columns='Año', aggfunc='sum')

print(df_pivot)
```

Ilustración 6 Ejemplo de pivot_table

Este código pivotea el DataFrame para mostrar las ventas por ciudad y año.

DESPIVOTEO DE DATAFRAMES (MÉTODO MELT)

El despivoteo transforma los datos de un formato ancho a un formato largo, lo que es útil para análisis posteriores, especialmente cuando se quiere trabajar con series temporales o hacer análisis más detallados.

Uso de melt

Ejemplo:

```python
# Crear un DataFrame de ejemplo
data = {
    'Ciudad': ['Madrid', 'Barcelona'],
    '2020': [1000, 1200],
    '2021': [1500, 1300]
}

df = pd.DataFrame(data)

# Despivoteo con melt
df_melted = pd.melt(df, id_vars=['Ciudad'], value_vars=['2020', '2021'], var_name='Año', value_name='Ventas')

print(df_melted)
```

Ilustración 7 Ejemplo uso de melt

En este ejemplo, convertimos el formato ancho en un formato largo, donde cada fila representa las ventas de una ciudad para un año determinado.

COMBINACIÓN Y MERGE DE DATOS

En el análisis de datos, a menudo es necesario combinar información de múltiples fuentes. Pandas ofrece herramientas como concat() y merge() para realizar estas operaciones de manera eficiente, dependiendo de la estructura y relación de los datos.

Concatenación de DataFrames

La función **pd.concat()** permite unir múltiples DataFrames a lo largo de un eje:

axis=0 → Concatena los DataFrames verticalmente (apilando filas).

axis=1 → Concatena los DataFrames horizontalmente (uniendo columnas).

Ejemplo de concatenación vertical (agregar filas):

```python
# Crear dos DataFrames
df1 = pd.DataFrame({'ID': [1, 2], 'Ventas': [100, 200]})
df2 = pd.DataFrame({'ID': [3, 4], 'Ventas': [300, 400]})

# Concatenar los DataFrames
df_concat = pd.concat([df1, df2], axis=0)

print(df_concat)
```

Ilustración 8 Ejemplo de concatenación vertical (agregar filas)

Salida

	ID	Ventas
0	1	100
1	2	200
2	3	300
0	4	400

Ilustración 9 Salida

Nota: Los índices originales se conservan. Si quieres restablecerlos, usa df_concat.reset_index(drop=True).

Ejemplo de concatenación horizontal (unir columnas):

```
df3 = pd.DataFrame({'ID': [1, 2, 3], 'Descuento': [10, 15, 20]})
df_hconcat = pd.concat([df1, df3], axis=1)
print(df_hconcat)
```

Ilustración 10 Ejemplo de concatenación horizontal (unir columnas)

Salida

	ID	Ventas	ID	Descuento
0	1	100	1	10
1	2	200	2	15
2	3	300	3	20

Ilustración 11 Salida

Nota: En este caso, los DataFrames deben tener el mismo número de filas para una combinación lógica.

Cuándo usar pd.concat()

Usa concat() cuando:

- Tienes estructuras de datos similares y quieres unirlas.
- No necesitas combinar los datos en base a una clave específica.
- Solo deseas apilar filas (agregar más datos de la misma estructura) o unir columnas (si ambos DataFrames tienen el mismo número de filas).

Merge de DataFrames

La función pd.merge() permite combinar DataFrames basándose en una clave común, similar a una unión (JOIN) en SQL.

```python
# Crear dos DataFrames
df1 = pd.DataFrame({'ID': [1, 2, 3], 'Ventas': [100, 200, 300]})
df2 = pd.DataFrame({'ID': [2, 3, 4], 'Año': [2020, 2021, 2022]})

# Realizar el merge en base a la columna 'ID'
df_merged = pd.merge(df1, df2, on='ID')

print(df_merged)
```

Ilustración 12 Ejemplo de merge por columna "ID"

Tipos de Merge disponibles:

- inner (intersección) → Solo valores comunes.
- left (izquierda) → Conserva todos los valores del primer DataFrame y agrega los del segundo si hay coincidencia.
- right (derecha) → Conserva todos los valores del segundo DataFrame.
- outer (unión) → Conserva todos los valores de ambos DataFrames, rellenando con NaN si no hay coincidencia.

Nota: Si no se declara el parámetro how en pd.merge(), Pandas por defecto usa how='inner'.

Cuándo usar pd.merge()

Usa merge() cuando:

- Necesitas combinar datos de diferentes fuentes en base a una clave común (similar a una unión en SQL).
- Los DataFrames contienen información relacionada pero distribuida en tablas separadas.
- Quieres realizar diferentes tipos de uniones (inner, left, right, outer).

Método	Cuándo usarlo	Descripción
pd.concat()	Para unir filas o columnas sin depender de claves comunes.	Une DataFrames por filas (axis=0) o columnas (axis=1).
pd.merge()	Para combinar datos con una clave común, como en SQL.	Combina DataFrames según una clave (inner, left, right, outer).

¿Cómo elegir entre concat() y merge()?

- Si los DataFrames tienen la misma estructura y solo necesitas unirlos, usa concat().
- Si necesitas combinar información basada en una clave común, usa merge().