**Para saber mais: entendendo o groupby**

O método groupby do Pandas permite agrupar e resumir dados de um DataFrame, com base em um ou mais critérios. Esses critérios podem ser variáveis numéricas ou categóricas, como colunas ou índices do DataFrame.

A ideia por trás do groupby é dividir os dados em grupos com base nos critérios selecionados e, em seguida, aplicar uma operação a esses grupos. Essa operação pode ser uma função de agregação, como soma, média, contagem, desvio padrão, entre outras, ou mesmo uma operação personalizada definida pela pessoa usuária.

Esse método possui diversos parâmetros, alguns deles são:

* **by**: esse é o parâmetro mais comum e é usado para especificar a coluna ou colunas que queremos agrupar. Como argumento dele, podemos passar o nome de uma coluna ou uma lista de nomes de colunas;
* **axis**: utilizamos esse parâmetro para especificar o eixo ao longo do qual queremos agrupar. O valor padrão dele é 0, o que significa que as **linhas** serão agrupadas. Se quisermos agrupar as **colunas**, devemos definir esse parâmetro como 1;
* **sort**: esse parâmetro é um booleano (True ou False) que indica se os grupos devem ser ordenados pelo valor da coluna de agrupamento. O valor padrão é True;
* **dropna**: utilizamos esse parâmetro para controlar se os valores ausentes (NaN) serão excluídos ou não durante o processo de agrupamento. O valor padrão é True.

Vamos analisar um exemplo simples utilizando o groupby. Suponha que temos o seguinte DataFrame:

Código do DataFrame:

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

df = pd.DataFrame({

'Animal': ['Cachorro', 'Gato', 'Elefante', 'Cachorro', 'Gato', 'Elefante'],

'Cor': ['Preto', 'Branco', 'Cinza', 'Marrom', 'Preto', 'Marrom'],

'Quantidade': [2, 3, 1, 4, 2, 2]

})

df

DataFrame:

|  | **Animal** | **Cor** | **Quantidade** |
| --- | --- | --- | --- |
| 0 | Cachorro | Preto | 2 |
| 1 | Gato | Branco | 3 |
| 2 | Elefante | Cinza | 1 |
| 3 | Cachorro | Marrom | 4 |
| 4 | Gato | Preto | 2 |
| 5 | Elefante | Marrom | 2 |

Aqui nós temos um DataFrame com dados referentes a alguns animais. Vamos utilizar o groupby para agrupar os dados pelo tipo de animal e realizarmos a soma da coluna "Quantidade" para conferirmos o total de cada tipo de animal:

df.groupby('Animal').sum(numeric\_only=**True**)

Resultado:

| **Animal** | **Quantidade** |
| --- | --- |
| Cachorro | 6 |
| Elefante | 3 |
| Gato | 5 |

Podemos também usar o método groupby para agrupar os dados por múltiplas colunas. Por exemplo, vamos agrupar pelo tipo do animal e também pela cor e realizar a soma da coluna "Quantidade":

df.groupby(['Animal', 'Cor'])[['Quantidade']].sum()

Resultado:



Assim, podemos observar que o groupby é um método muito útil para situações em que precisamos dividir os dados em grupos para analisar o comportamento desses grupos. Utilizando esse método, nós podemos responder perguntas específicas sobre nossos conjuntos de dados.

Caso queira aprender mais sobre esse método, deixo a sugestão de dois artigos:

* [Pandas GroupBy: Your Guide to Grouping Data in Python](https://realpython.com/pandas-groupby/);
* [Pandas’ groupby explained in detail](https://towardsdatascience.com/pandas-groupby-aggregate-transform-filter-c95ba3444bbb).

Ambos os artigos estão em inglês, mas podem ser facilmente traduzidos utilizando a extensão de tradução do Google Chrome.

**Para saber mais: alterando os nomes de colunas**

Quando trabalhamos com bases de dados, em alguns momentos, pode ser necessário fazermos alterações nos nomes das colunas. Por exemplo, quando criamos o seguinte DataFrame no nosso projeto:

df['Tipo'].value\_counts(normalize=True).to\_frame().sort\_values('Tipo')

Tivemos o seguinte resultado:

|  | **Tipo** |
| --- | --- |
| Studio | 0.000173078 |
| Loft | 0.00220674 |
| Casa de Vila | 0.0107741 |
| Flat | 0.0205963 |
| Quitinete | 0.0361733 |
| Casa | 0.0418415 |
| Casa de Condomínio | 0.0430964 |
| Apartamento | 0.845139 |

Podemos notar que a coluna com os percentuais, ficou nomeada como "Tipo". E se quiséssemos mudar o nome dela para "Percentuais"?

Nesse caso, nós podemos utilizar o método rename() para mudar o nome dessa coluna. Esse método nos permite especificar um dicionário que associa o nome antigo da coluna ao novo nome que queremos atribuir. Então, vamos fazer isso:

# Salvando o DataFrame em uma variável

df\_exemplo = df['Tipo'].value\_counts(normalize=**True**).to\_frame().sort\_values('Tipo')

# Alterando o nome da coluna "Proportion" para "Percentuais"

df\_exemplo.rename(columns={'proportion': 'Percentuais'}, inplace=**True**)

# Visualizando o DataFrame

df\_exemplo

Nós utilizamos o parâmetro inplace=True para que essa alteração seja definitivamente aplicada no nosso DataFrame df\_exemplo. Sendo assim, ao visualizar o DataFrame, teremos o seguinte resultado:

|  | **Percentuais** |
| --- | --- |
| Studio | 0.000173078 |
| Loft | 0.00220674 |
| Casa de Vila | 0.0107741 |
| Flat | 0.0205963 |
| Quitinete | 0.0361733 |
| Casa | 0.0418415 |
| Casa de Condomínio | 0.0430964 |
| Apartamento | 0.845139 |

E pronto! Agora nós também sabemos como alterar nomes de colunas em um DataFrame.