Aprofundando em filtros com Pandas

Filtrando dados com condições

Já vimos anteriormente como filtrar dados com condições através da propriedade de indexação **loc[]**!! E agora vamos dar uma reforçada como fazemos filtros com **N condições** diferentes!

```
filtro1 = df.loc[(df['Rooms'] == 3) & (df['Method'] == 'S')]
filtro1
```

✓ 0.0s

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017
9	Albert Park	18 Mills St	3	h	1925000.0	S	Cayzer	1/04/2017
...	...	...	...	...	...	...	...	...
63008	Yarraville	4/247 Williamstown Rd	3	t	NaN	S	Jas	30/12/2017
63017	Preston	229 Murray Rd	3	h	808000.0	S	RW	31/03/2018
63018	Roxburgh Park	3 Carr Pl	3	h	566000.0	S	Raine	31/03/2018
63019	Roxburgh Park	9 Parker Ct	3	h	500000.0	S	Raine	31/03/2018
63020	Roxburgh Park	5 Parkinson Wy	3	h	545000.0	S	Raine	31/03/2018

15871 rows × 13 columns

Veja como é simples!! Basta separar cada condição do filtro, que seria cada listagem do objeto **Series Boolean (máscara booleana)**, através de **()**.

Filtrando com a função contains()

Em alguns momentos pode ser que precisamos realizar algum filtro referente a busca de uma **parte do texto**, como por exemplo filtrar dados que contém a palavra **streat (st)** ou **rua**.

E para isso no momento do filtro, primeiramente chamamos a propriedade **str**, para que cada dado do nosso objeto **series** seja convertido em **string**, e logo em seguida utilizamos a função **contains()**, como podemos ver a seguir:

```
filtro2 = df.loc[df['Address'].str.contains('St')]
filtro2
```

✓ 0.0s

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Post
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	

Utilizando regular expression

Vimos que anteriormente conseguimos realizar filtro referente a busca de uma parte do texto utilizado a função **contains()**. Porém da forma como está, caso o texto que estivemos filtrando tiver em **uper case** ou em **lower case** diferente do texto informado do filtro, logo o filtro não será realizado!

E para resolver podemos utilizar o recurso de **regular expression (regex)**, pois com o **regex** conseguimos descrever padrões de texto! Muito utilizado para realizar **validações dentro de um texto** ou até mesmo **extrair parte de um texto**!

E para usar o **regular expression** nesse caso, na própria função do **contains()**, temos um parâmetro chamado **flags**, onde podemos definir nossa expressão regular nele!

```
import pandas as pd
import regex as re
```
✓ 0.0s

```
filtro3 = df.loc[df['Address'].str.contains('ST', flags=re.I)]
filtro3
```
✓ 0.0s

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017

Veja que primeiramente foi importado a biblioteca do **regex**, e após importar a biblioteca utilizamos a propriedade **I** da biblioteca no parâmetro **flags** do **contains()**, onde estamos definindo que seja ignorado letras **maiúsculas** e **minúscula** no filtro!

Modificando dados no DataFrame

Já vimos anteriormente como podemos atualizar dados de um **DataFrame**, ou de um objeto **Series** pertencente a um **DataFrame**, como podemos ver a seguir:

```
df['SellerG'] = 'Fernando'
df
```
✓ 0.0s

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Fernando	1/04/2017
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Fernando	1/04/2017
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Fernando	1/04/2017
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Fernando	1/04/2017

Veja que alteramos todos os valores da coluna (**Series**) '**SellerG**' para '**Fernando**'! Porém não queremos alterar todos os dados dessa coluna para **Fernando**!!

Dito isso, queremos alterar somente o nome de **Nelson** para **Fernando**, logo precisaremos realizar um filtro primeiro para pegar todos os registros referente ao **SellerG** de **Nelson**, só para depois realizar a atualização em si para **Fernando**!

```
df_update = df.loc[df['SellerG'] == 'Nelson']
df_update['SellerG'] = 'Fernando'
df_update
```

✓ 0.0s

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Fernando	1/04/2017	3067	Norther
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Fernando	1/04/2017	3042	Wester
7	Airport West	1/26 Highridge Cr	3	h	715000.0	SP	Fernando	1/04/2017	3042	Wester
50	Bellfield	1/15 Toohey St	3	u	675000.0	SP	Fernando	1/04/2017	3081	Easter
94	Brunswick	53 Amelia St	2	h	740000.0	PI	Fernando	1/04/2017	3056	Norther

Porém se observar no resultado, temos um alerta que o **Jupyter** nos apresenta, dizendo que estamos tentando modificar uma parte (**slice**) de um **DataFrame**, porém o **Pandas** não sabe se estamos tentando modificar a lista original (**df**) ou uma cópia que foi feito através do **slice**, quando realizamos o filtro e armazenamos esse **slice** através da variável (**df_update**)!!

```
C:\Users\fernando.spelling\AppData\Local\Temp\ipykernel_14928\1266044858.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

E para resolver isso é muito simples!! Basta no momento do filtro que usamos através da propriedade **loc[]**, definirmos o valor a ser modificado no mesmo comando, assim o **Pandas** saberá que a modificação deve ser feita no **DataFrame** original (**df**)!!

```
df.loc[df['SellerG'] == 'Nelson', ['SellerG']] = 'Fernando'
df
```

✓ 0.0s

	Suburb	Address	Rooms	Type	Price	Method	SellerG	
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Fernando	1/
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Fernando	1/
...	...	...	...	...	...	...	...	
63018	Roxburgh Park	3 Carr Pl	3	h	566000.0	S	Raine	31

<u>Obs: Note que precisamos definir qual coluna(s) queremos modificar, pois da forma padrão o filtro retornara o DataFrame filtrado com todas as colunas, e nesse caso não queremos modificar o DataFrame inteiro e sim apenas uma coluna apenas (SellerG)!</u>

```
df.loc[df['SellerG'] == 'Nelson', ['SellerG']] = 'Fernando'
```

Agrupando dados com GroupBy

Um recurso bem importante que podemos utilizar também, seria a utilização do **GroupBy**!! Onde basicamente irá agrupar todos os nossos dados através de uma coluna de referência!

Como por exemplo, vamos agrupar os nossos dados calculando a soma dos valores referente a cada vendedor (**SallerG**)!! Dito isso, usaremos a função **groupby()**, passando o **SallerG** como parâmetro, e consequentemente chamando também a função **sum()** para realizar o cálculo da somatória dos valores de cada vendedor!

```
filtro5 = df.groupby(['SellerG']).sum()
filtro5
```

0.2s Open 'filtro5' in Data Wrangler

SellerG	Suburb	Address	Rooms	Type	Price
@Realty	CraigieburnCraigieburnBroadmeadowsCraigieburnR...	59 Balyang Wy66 Northern Cr83 Waranga Cr2 Bens...	98	hhhhhhhhhthuhhhtthhhhhhhhh	15727500.0
A	DallasDallasGreenvaleGlenroyMeadow HeightsHadf...	8 Corinella Cr5 Leigh Ct27 Langton Wy3/24 Gran...	27	hhhhhhhhh	4359000.0
AIME	Ashwood	2/11 Salisbury Rd	4	t	980000.0
ASL	TemplestoweDoncasterRingwoodCroydonReservoirCo...	2/61 Wood St48 Winston Dr62 Ringwood St16 Dona...	34	thhhhhhhhh	11037500.0

Veja que foi listado corretamente, **somando todos os valores agrupados** para cada vendedor (**SellerG**)!!

Porém por padrão, está sendo ordenado referente aos vendedores (**SellerG**) e queremos que seja ordenado de forma decrescente os preços (**Price**), ou seja, que seja ordenado os vendedores que tiveram as maiores vendas!

```
filtro5 = df.groupby(['SellerG']).sum().sort_values('Price', ascending=False)
filtro5
```

0.2s Open 'filtro5' in Data Wrangler

SellerG	Suburb	Address	Rooms	Type	Price
Jellis	AbbotsfordAirport WestAlphingtonArmadaleArmada...	49 Lithgow St4/32 Earl St41 Toolangi Rd2/23 As...	16948	hthuhhthhhhhhhhhtthhhthhuhtthhuuhuhhhuhuhhuuhu...	5.301902e+09
Nelson	AbbotsfordAirport WestAirport WestBellfieldBru...	119B Yarra St92 Clydesdale Rd1/26 Highridge Cr...	13808	hhhuhhhhthhuhuhhhhhhhuhhhhuhhhhhhhuhhhhhuhhhhhh...	4.089097e+09
Barry	AberfeldieAirport WestAltona MeadowsAttwoodBla...	68 Vida St3/74 Hawker St57 Tatman Dr46 Threadn...	22919	huhhhthhthhhhhhhhhhhhhhhhhhhhhhhuhhhhhhthhhhtu...	4.022899e+09