



Problema

Uma empresa que fornece cestas básicas confere semanalmente os preços dos produtos nos seguintes supermercados: Descontão, KiBarato, Pop, Mercadão de Descontos e SuperPrice.

Os produtos do kit Higiene e Limpeza conferidos são:

Produto	Categoria	Unidade	Quantidade
Sabonete	Higiene	90 g	3
Papel Higiênico	Higiene	Pct 4 unidades	2
Escova Dental	Higiene	Unidade	2
Creme Dental	Higiene	90 g	1
Protetor Solar FPS >= 30	Higiene	100 ml	1
Repelente de insetos	Higiene	300 ml	1
Detergente Líquido	Limpeza	500 ml	1
Lã de Aço	Limpeza	Pct 4 unidades	1
Sabão em Pó	Limpeza	500 g	1
Desinfetante	Limpeza	1000 ml	1

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3,39	2,48	1,97	2,09	2,60
Papel Higiênico	6,75	8,36	7,92	9,43	7,57
Escova Dental	1,69	3,58	1,80	1,67	3,88
Creme Dental	2,69	2,80	2,37	3,35	2,86
Protetor FPS >= 30	16,21	28,23	17,28	27,80	24,37
Repelente	9,24	8,02	11,76	10,81	12,33
Detergente Líquido	1,34	1,87	1,22	1,73	2,10
Lã de Aço	0,68	2,33	2,74	2,90	2,26
Sabão em Pó	6,21	9,73	6,76	7,98	9,35
Desinfetante	3,26	2,72	1,37	2,09	2,34

Construa um programa que exiba:

- para cada produto seu preço médio e qual mercado o vende mais barato e seu preço;
- considerando as quantidades necessárias para montar o kit higiene da cesta básica, qual o melhor mercado.

1



Análise do Problema

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3,39	2,48	1,97	2,09	2,60
Papel Higiênico	6,75	8,36	7,92	9,43	7,57
.....					
Sabão em Pó	6,21	9,73	6,76	7,98	9,35
Desinfetante	3,26	2,72	1,37	2,09	2,34

Visão por Supermercado:

Um supermercado:

- nome
- 10 preços x produtos: Series
index: nome dos produtos
values: preços dos produtos

Todos os supermercados:

Dicionário com
chave = nome do supermercado
valor = Series

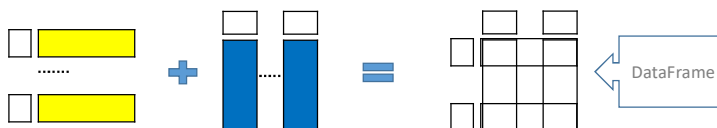
Visão por produto:

Um produto:

- nome
- 5 preços x supermercados: Series
index: nomes dos supermercados
values: preços nos supermercados

Todos os produtos:

Dicionário com
chave = nome do produto
valor = Series



2

DataFrames

DataFrames do Pandas

DataFrames: estrutura bidimensional indexada que armazena valores de qualquer tipo.

Estrutura tabular com linhas e colunas, similar a uma planilha, composta por:

- **valores**: *array* bidimensional (estruturado ou homogêneo), dicionário (que pode conter Series, arrays, constants ou objetos do tipo lista) ou DataFrame
 - **índices**: uma sequência de números ou rótulos (*labels*) quaisquer que identifiquem as linhas
 - **colunas**: uma sequência de números ou rótulos (*labels*) quaisquer que identifiquem as colunas
- ✓ Os índices e as colunas não precisam ser exclusivos. Por padrão, variam de 0 a itens -1

Índice e Coluna Padrões

	0	1	2	4	4
0	216	Huguinho	9.6	3.8	1.0
1	233	Lalá	6.2	6.9	9.2
2	234	Lelé	6.5	2.7	3.0
3	235	Lili	1.3	4.6	6.5
4	230	Luisinho	9.8	3.7	9.3
5	215	Zezinho	6.2	1.3	8.5

Índice e Coluna Dado

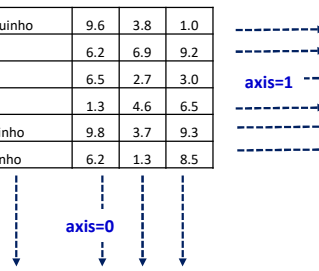
	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5

DataFrames do Pandas

Estrutura tabular com linhas e colunas, similar a uma planilha, composta por:

- ✓ “grupo de Series que compartilham um índice (nome das colunas)” ou
- ✓ “um dicionário de Series”

	Nome	P1	P2	P3
216	Huguinho	9.6	3.8	1.0
233	Lalá	6.2	6.9	9.2
234	Lelé	6.5	2.7	3.0
235	Lili	1.3	4.6	6.5
230	Luisinho	9.8	3.7	9.3
215	Zezinho	6.2	1.3	8.5



 axis=0 axis=1

5

Construindo DataFrames

6



Construindo um DataFrame

sem especificação dos índices

Sintaxe:

`pd.DataFrame(valores)`

a partir de uma lista de listas

```
>>>lNtA = [
    ['Lalá', 133, 6.2, 6.9, 9.2],
    ['Lelé', 131, 6.5, 2.7]]
```

```
>>>dfNtA= pd.DataFrame(lNtA)
```

```
>>>dfNtA
```

	0	1	2	3	4
0	Lalá	133	6.2	6.9	9.2
1	Lelé	131	6.5	2.7	NaN

a partir do dicionário de listas

```
>>>dNtB= {
    'Lalá': [133, 6.2, 6.9, 9.2],
    'Lelé': [131, 6.5, 2.7, None]}
```

```
>>>dfNtB= pd.DataFrame(dNtB)
```

```
>>>dfNtB
```

	Lalá	Lelé
0	133.0	131.0
1	6.2	6.5
2	6.9	2.7
3	9.2	NaN

← ???

7



Índices e Colunas

Índices:

`df.index`

```
dfNtA:   0    1    2    3    4
0  Lalá  133  6.2  6.9  9.2
1  Lelé  131  6.5  2.7  3.0
```

```
>>>dfNtA.index
```

```
RangeIndex(start=0, stop=2, step=1)
```

```
>>>dfNtA.columns
```

```
RangeIndex(start=0, stop=5, step=1)
```

Colunas:

`df.columns`

```
dfNt1: Matr  P1  P2  P3
Lalá    133  6.2  6.9  9.2
Lelé    131  6.5  2.7  3.0
```


```
>>>dfNt1.index
```

```
Index(['Lalá', 'Lelé'],
      dtype='object')
```

```
>>>dfNt1.columns
```

```
Index(['Matr', 'P1', 'P2', 'P3'],
      dtype='object')
```

8



Valores e dimensões

Valores: `df.values`

Dims: `df.shape`

dfNtA:

	0	1	2	3	4
0	Lalá	133	6.2	6.9	9.2
1	Lelé	131	6.5	2.7	3.0

dfNt1:

	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0


```
>>>dfNtA.values
array([[ 'Lalá', 133, 6.2, 6.9, 9.2],
       [ 'Lelé', 131, 6.5, 2.7, 3.0]] ,
      dtype=object)

>>>dfNtA.shape
(2, 5)
```

```
>>>dfNt1.values
array([[ 133, 6.2, 6.9, 9.2],
       [ 131, 6.5, 2.7, 3.0]], dtype=object)

>>>dfNt1.shape
(2, 4)
```

9



DataFrame Transposto

Sintaxe: `df.T`

dfNt1

	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

dfNt2

	Lalá	Lelé
Matr	133	131
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	3.0

```
>>>dfNt1.T
      Lalá    Lelé
Matr    133    131
P1      6.2    6.5
P2      6.9    2.7
P3      9.2    3.0
```

```
>>>dfNt2.T
      Matr    P1    P2    P3
Lalá    133  6.2  6.9  9.2
Lelé    131  6.5  2.7  3.0
```

10



Construindo um DataFrame

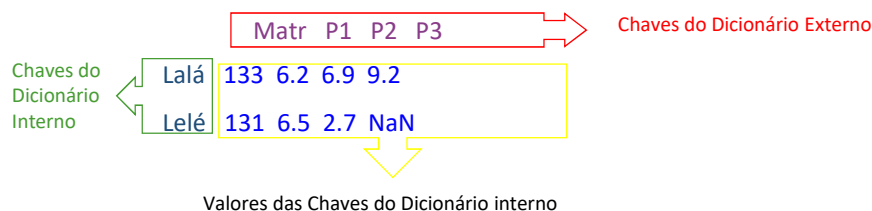
a partir de dicionário de dicionários_{1/2}

a partir de dicionário de dicionários

```
dNtC= {'Matr':{'Lalá': 133, 'Lelé': 131},
      'P1': {'Lalá': 6.2, 'Lelé': 6.5},
      'P2': {'Lelé': 2.7, 'Lalá': 6.9},
      'P3': {'Lalá': 9.2} }
```

```
>>>dfNtC= pd.DataFrame(dNtC)
```

```
>>>dfNtC      #casamento pelas chaves
```



11



Construindo um DataFrame

a partir de dicionário de dicionários_{1/2}

PASSO A PASSO – Casamento pelas chaves

a partir de dicionário de dicionários

```
dNtC= { 'Matr': { 'Lalá': 133, 'Lelé': 131},
      'P1': { 'Lalá': 6.2, 'Lelé': 6.5},
      'P2': { 'Lelé': 2.7, 'Lalá': 6.9},
      'P3': { 'Lalá': 9.2} }
```

```
>>>dfNtC= pd.DataFrame(dNtC)
```

```
>>>dfNtC      #casamento pelas chaves
```

```
      Matr   P1   P2   P3
Lalá  133   6.2   6.9   9.2
Lelé  131   6.5   2.7  NaN
```

12



Construindo um DataFrame

a partir de dicionário de dicionários_{2/2}

a partir de dicionário de dicionários

```
dNtD= { 'Lalá':{'Matr': 133,'P1': 6.2,'P2': 6.9,'P3': 9.2 },
        'Lelé':{'Matr': 131,'P1': 6.5,'P2': 2.7      }
      }
```

```
>>>dfNtD= pd.DataFrame(dNtD)
```

```
>>>dfNtD      #casamento pela chave
```

	Lalá	Lelé
Matr	133.0	131.0
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	NaN

???

13



Construindo um DataFrame

a partir de dicionário de Series_{1/2}

a partir de dicionário de Series, index indefinido

```
sNomes=pd.Series(['Lalá','Lelé'])
sMatrs=pd.Series([133,131])
sP1s=pd.Series([6.2,6.5])
sP2s=pd.Series([6.9,2.7])
sP3s=pd.Series([9.2])
dNtSs={'Nome':sNomes,
      'Matr':sMatrs,
      'P1':sP1s,
      'P2':sP2s,
      'P3':sP3s}
```

Nome:	Matr:	P1:	P2:	P3:
0 Lalá	0 133	0 6.2	0 6.9	0 9.2
1 Lelé	1 131	1 6.5	1 2.7	

```
dfNtSs= pd.DataFrame(dNtSs)
```

```
>>>dfNtSs      #casamento pelo index
```

	Matr	Nome	P1	P2	P3
0	133	Lalá	6.2	6.9	9.2
1	131	Lelé	6.5	2.7	NaN

14



Construindo um DataFrame

a partir de dicionário de Series_{2/2}

a partir de dicionário de Series, index definido

```
I=['Lalá', 'Lelé']
sMatri=pd.Series([133,131],
                 index=I)
sP1i=pd.Series([6.2,6.5],index=I)
sP2i=pd.Series([6.9,2.7],index=I)
sP3i=pd.Series([9.2,3.0],index=I)

dNtSi={'Matr':sMatri,
       'P1':sP1i,
       'P2':sP2i,
       'P3':sP3i}
```

Matr :	P1 :	P2 :	P3 :
Lalá 133	Lalá 6.2	Lalá 6.9	Lalá 9.2
Lelé 131	Lelé 6.5	Lelé 2.7	Lelé 3.0

```
dfNtSi= pd.DataFrame(dNtSi)
```

```
>>>dfNtSi #casamento pelo index
```

	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

15



Mãos na Massa

Crie possíveis estruturas que podem dar origem aos seguintes DataFrames:

A)

	0	1	2	3	4
0	Sabonete	3.39	2.48	1.97	2.09
1	Escova Dental	1.69	3.58	1.80	1.67
2	Creme Dental	2.69	2.80	2.37	3.35

B)

	Descontão	KiBarato	Pop
Escova Dental	1.69	3.58	1.80
Creme Dental	2.69	NaN	2.37

16


```
import pandas as pd
l = [['Sabonete',3.39,2.48,1.97,2.09],
     ['Escova Dental',1.69,3.58,1.80,1.67],
     ['Creme Dental',2.69,2.80,2.37,3.35]]

dfProd1= pd.DataFrame(l)
print(dfProd1)

dic = {'Descontão': {'Escova Dental':1.69,'Creme Dental',2.69}
       'KiBarato': {'Escova Dental':3.58 }
       'Pop':       {'Escova Dental':1.80,'Creme Dental',2.37}
      }
dfProd2= pd.DataFrame(dic)
print(dfProd2)
```

17

Sintaxe:

```
pd.DataFrame(valores, index=array1d, columns=array1d)
```

a partir de uma lista

```
lNtA=[['Lalá',133,6.2,6.9,9.2], ['Lelé',131,6.5,2.7,3.0]]
ind = ['Lalá','Lelé']
cols = ['Nome','Matr','P1','P2','P3']

>>>dfNtAic= pd.DataFrame( lNtA , index=ind, columns=cols)
>>> dfNtAic
      Nome  Matr  P1  P2  P3
Lalá  Lalá   133  6.2  6.9  9.2
Lelé  Lelé   131  6.5  2.7  3.0

>>>dfNtAc= pd.DataFrame( lNtA ,columns=cols)
>>> dfNtAc
      Nome  Matr  P1  P2  P3
0  Lalá   133  6.2  6.9  9.2
1  Lelé   131  6.5  2.7  3.0
```

18



Construindo um DataFrame

com 1ª planilha de um arquivo Excel

Sintaxe:

```
pd.read_excel(caminho, index_col=n, header=n, decimal=',')
```

caminho - localização do arquivo: composto pelo caminho (absoluto/relativo) e nome

index_col = n - O número da coluna do arquivo a ser usada como labels das linhas (índice). O padrão é **None** (o arquivo não possui tal coluna)

header = n - O número da linha para os labels das colunas (padrão é 0) ou **None** quando não há tal linha

decimal = ',' - quando o separador de casas decimais é a vírgula, Padrão: '.'

```
dfNtA=pd.read_excel("als.xlsx", decimal=',')
print(dfNtA.head(2))
```

	Nome	Matr	P1	P2	P3
0	Lalá	133	6.2	6.9	9.2
1	Lelé	131	6.5	2.7	3.0

als.xlsx

	A	B	C	D	E
1	Nome	Matr	P1	P2	P3
2	Lalá	133	6.2	6.9	9.2
3	Lelé	131	6.5	2.7	3.0

Arquivo na mesma pasta do arq .py

19



Construindo um DataFrame

com 1ª planilha de um arquivo Excel

```
dfNt2=pd.read_excel("als.xlsx", index_col=0, decimal=',')
print(dfNt2.head(2))
```

	Matr	P1	P2	P3
Nome				
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0

als.xlsx

	A	B	C	D	E
1	Nome	Matr	P1	P2	P3
2	Lalá	133	6.2	6.9	9.2
3	Lelé	131	6.5	2.7	3.0

```
dfNt3=pd.read_excel("als.xlsx", index_col=1, decimal=',')
print(dfNt3.head(2))
```

	Nome	P1	P2	P3
Matr				
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0

20



Construindo um DataFrame

com planilha *nomeada* de um arquivo Excel

```
dfNTesA=pd.read_excel("als.xlsx",sheetname='Teste',index_col=0,decimal=',')
print(dfNTesA.head(2))
```

	Matr	T1	T2	T3
Nome				
Lalá	133	7	7	NaN
Lelé	131	8	8	8.0

```
dfNTesB=pd.read_excel("als.xlsx",sheetname='Teste',index_col=1,decimal=',')
print(dfNTesB.head(2))
```

	Nome	T1	T2	T3
Matr				
133	Lalá	7	7	NaN
131	Lelé	8	8	8.0

21



Obtendo informações do DataFrame

Sintaxe:

```
df.info()
```

dfNTd:	Nome	P1	P2	P3
Matr				
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0
135	Lili	1.3	4.6	6.5

```
dfNTd.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 133 to 135
Data columns (total 4 columns):
Nome      3 non-null object
P1         3 non-null float64
P2         3 non-null float64
P3         3 non-null float64
dtypes: float64(3), object(1)
memory usage: 120.0+ bytes
```

dfNte:	Nome	Matr	P1	P2	P3
0	Lalá	133	6.2	6.9	9.2
1	Lelé	131	6.5	2.7	3.0
2	Lili	135	1.3	4.6	6.5

```
dfNte.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 5 columns):
Nome      3 non-null object
Matr      3 non-null int64
P1         3 non-null float64
P2         3 non-null float64
P3         3 non-null float64
dtypes: float64(3), int64(1), object(1)
memory usage: 200.0+ bytes
```

22



Alterando nome de um eixo (lista de colunas/linhas)

Sintaxe:

```
df.rename_axis(mapper, axis=0, inplace=False)
df.index.name = nome ou df.columns.name = nome
```

Altera o nome do *index* ou *columns*. Retorna um *DataFrame* com o nome da lista de colunas/linhas alterado

mapper = valor que será o nome do eixo (escalar, list-like)

axis = *n/nome* – eixo a renomear. Padrão: 0 (altera o nome do *index*). Pode ser o nome do eixo

inplace = **False/True**. Padrão: **False**. Com *inplace=True*, realiza a operação no *DataFrame*, não cria uma cópia.

```
dfNtd:
Nome
Lalá    133 6.2 6.9 9.2
Lelé    131 6.5 2.7 3.0
```

```
dfNtd.rename_axis("NOME", inplace=True)
# equivalente a:
# dfNtd.rename_axis("NOME", axis="index", inplace=True)
# dfNtd.index.name='NOME'

      Matr   P1   P2   P3
NOME
Lalá    133  6.2  6.9  9.2
Lelé    131  6.5  2.7  3.0
```

```
dfNtd.rename_axis("AL", axis=1, inplace=True)
# equivalente a:
# dfNtd.rename_axis("AL", axis="columns", inplace=True)
# dfNtd.columns.name='AL'
```

```
AL      Matr   P1   P2   P3
NOME
Lalá    133  6.2  6.9  9.2
Lelé    131  6.5  2.7  3.0
```

23



Alterando nome de colunas e/ou linhas

Sintaxe:

```
df.rename(mapper=None, axis= n/str, index=None, columns=None, inplace=False)
```

Altera o nome do *index* ou *columns*. Retorna um *DataFrame* com o nome da lista de colunas/linhas alterado

mapper,index,columns - um dicionário ou função que é aplicado para transformar os valores do eixo. Usar *mapper* e *axis* para especificar o eixo ou *index/columns*

axis = *n/nome* – eixo a renomear. Padrão: 0 (altera o nome do *index*). Pode ser o nome do eixo

inplace = **False/True**. Padrão: **False**. Com *inplace=True*, realiza a operação no *DataFrame*, não cria uma cópia.

```
dfNtf:
Nome
Lalá    133 6.2 6.9 9.2
Lelé    131 6.5 2.7 3.0
```

```
dfNtf.rename(index=str.upper, columns={0:'Matr', 'Pr3':'P3'})

      Matr   P1   P2   P3
Nome
LALÁ    133  6.2  6.9  9.2
LELÉ    131  6.5  2.7  3.0
```

24

Mãos na Massa

O arquivo `PrecosProdutosSuperMercados.xlsx`, possui 3 planilhas:

PrecosProdutosSuperMercados.xlsx

	A	B	C	D

ProdHigieneMerc
ProdLimpezaMerc
Produtos

A planilha `ProdHigieneMerc` contém os preços nos supermercados dos produtos de higiene. A planilha `ProdLimpezaMerc` contém os preços nos supermercados dos produtos de limpeza. A planilha `Produtos` descreve os produtos: nome, tipo (Limpeza ou Higiene), unidade de comercialização e quantidade no kit.

1. Construa o `DataFrame` `dfProdHigMerc` onde em cada linha estão os dados de um produto de Higiene, indexados pelo seu nome.
2. A partir do `DataFrame` `dfProdHigMerc` exibir para cada supermercado a lista de produtos X preços

25

Uma Solução: construir DF a partir do Excel

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
dfPrHigMerc.columns.name='Mercado'
print(dfPrHigMerc)
dfMercPrHig=dfPrHigMerc.T
print(dfMercPrHig)
```

	Mercado	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Produto						
Sabonete		3.39	2.48	1.97	2.09	2.60
Papel Higiênico		6.75	8.36	7.92	9.43	7.57
Escova Dental		1.69	3.58	1.80	1.67	3.88
Creme Dental		2.69	2.80	2.37	3.35	2.86
Protetor FPS >= 30		16.21	28.23	17.28	27.80	24.37
Repelente		9.24	8.02	11.76	10.81	12.33
Produto	Sabonete	Papel Higiênico	Escova Dental	Creme Dental	Protetor FPS >= 30	Repelente
Mercado						
Descontão	3.39	6.75	1.69	2.69	16.21	9.24
KiBarato	2.48	8.36	3.58	2.80	28.23	8.02
Pop	1.97	7.92	1.80	2.37	17.28	11.76
Mercadão	2.09	9.43	1.67	3.35	27.80	10.81
SuperPrice	2.60	7.57	3.88	2.86	24.37	12.33

26

Visualização

27

Traçando gráficos com Pandas

Sintaxe:

```
df.plot(kind='line', xlim=(li,ls), ylim=(li,ls), grid=False, title=None,
        subplot=False, figsize=(x,y), ... )
```

kind= 'line': linha (default) 'bar': barra vertical 'barh': barra horizontal
 'hist': histograma 'box': boxplot 'scatter': dispersão 'pie': pizza
entre outros

[x|y]lim = limites do eixo x ou do eixo y

grid = True/False, mostrar grade de linhas

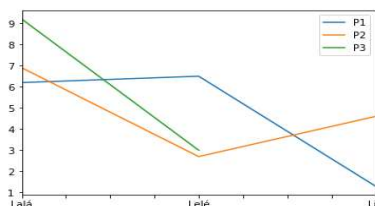
title = título do gráfico

subplots = True/False, criar gráficos separados para cada coluna

figsize = (altura,largura) em polegadas

dfE:	P1	P2	P3
Lalá	6.2	6.9	9.2
Lelé	6.5	2.7	3.0
Lili	1.3	4.6	NaN

dfE.plot()

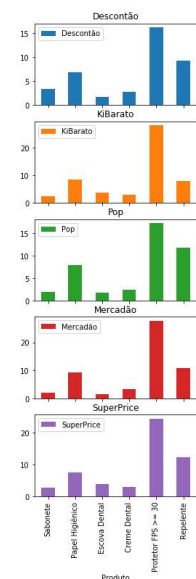


- Valores do eixo x: índices
- Valores do eixo y: colunas

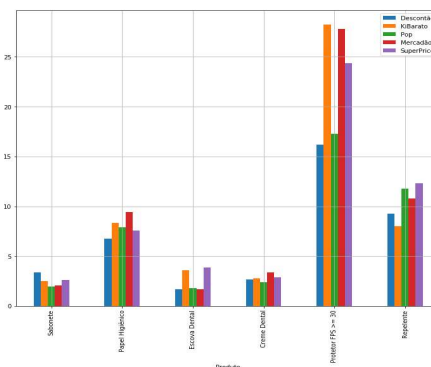
28

Traçando gráficos com Pandas: Exemplos

```
dfPrHigMerc.plot(
    kind='bar',
    figsize=(4,12),
    subplots=True,
    legend=True
)
```



```
dfPrHigMerc.plot(
    kind='bar',
    figsize=(4,12),
    grid=True
)
```



29

Problema: Exibindo de modo Não Tabular

Em DataFrames com muitas colunas ou muitas linhas, a apresentação dos dados em forma tabular geralmente fica pouco legível.

Como exibi-los como nos exemplos abaixo?

SUPERMERCADO: Descontão

Sabonete	: R\$ 3.39
Papel Higiênico	: R\$ 6.75
Escova Dental	: R\$ 1.69
Creme Dental	: R\$ 2.69
Protetor FPS >= 30	: R\$ 16.21
Repelente	: R\$ 9.24

SUPERMERCADO: KiBarato

Sabonete	: R\$ 2.48
Papel Higiênico	: R\$ 8.36
Escova Dental	: R\$ 3.58
Creme Dental	: R\$ 2.80
Protetor FPS >= 30	: R\$ 28.23
Repelente	: R\$ 8.02

SUPERMERCADO: Pop

Sabonete	: R\$ 1.97
Papel Higiênico	: R\$ 7.92
Escova Dental	: R\$ 1.80
Creme Dental	: R\$ 2.37
Protetor FPS >= 30	: R\$ 17.28
Repelente	: R\$ 11.76

PRODUTO: Sabonete

Supermercado Descontão	: R\$ 3.39
Supermercado KiBarato	: R\$ 2.48
Supermercado Pop	: R\$ 1.97
Supermercado Mercado	: R\$ 2.09
Supermercado SuperPrice	: R\$ 2.60

PRODUTO: Papel Higiênico

Supermercado Descontão	: R\$ 6.75
Supermercado KiBarato	: R\$ 8.36
Supermercado Pop	: R\$ 7.92
Supermercado Mercado	: R\$ 9.43
Supermercado SuperPrice	: R\$ 7.57

PRODUTO: Escova Dental

Supermercado Descontão	: R\$ 1.69
Supermercado KiBarato	: R\$ 3.58
Supermercado Pop	: R\$ 1.80
Supermercado Mercado	: R\$ 1.67
Supermercado SuperPrice	: R\$ 3.88

30

Iterando sobre Colunas

Sintaxe: `for (col, Series) in df.iteritems():`
 corpo

Para cada coluna, itera sobre sua Series, isto é, sobre os pares (índice da linha, valor).

```
def exibeColACol(df):
    for (c,s) in df.iteritems(): #(coluna, Series)
        print('Aluno {:8s}'.format(c))
        #Series desta coluna: índice da linha, valor
        for (ind,val) in s.iteritems():
            print(' {} - {}'.format(ind,val))
```

<u>dfNtD</u>	Lalá	Lelé
Matr	133.0	131.0
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	3.0

Aluno Lalá
 Matr - 133.0
 P1 - 6.2
 P2 - 6.9
 P3 - 9.2
 Aluno Lelé
 Matr - 131.0
 P1 - 6.5
 P2 - 2.7
 P3 - 3.0

31

Iterando sobre Linhas

Sintaxe: `for (lin, Series) in df.iterrows():`
 corpo

Para cada linha, itera sobre sua Series, isto é, sobre os pares (índice da coluna, valor).

```
def exibeLinALin(df):
    for (i,s) in df.iterrows(): #(linha, Series)
        print(' {:8s}'.format(i))
        #Series desta linha: índice da coluna, valor
        for (ind,val) in s.iteritems():
            print(' {} - {}'.format(ind,val))
```

<u>dfNtD</u>	Lalá	Lelé
Matr	133.0	131.0
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	3.0

Matr
 Lalá - 133
 Lelé - 131
 P1
 Lalá - 6.2
 Lelé - 6.5
 P2
 Lalá - 6.9
 Lelé - 2.7
 P3
 Lalá - 9.2
 Lelé - 3.0

32



Mãos na Massa

Exibir os dados do *DataFrame* dfPrHigMerc nos seguintes formatos:

a)

```
SUPERMERCADO: Descontão
Sabonete      : R$  3.39
Papel Higiênico : R$  6.75
Escova Dental  : R$  1.69
Creme Dental   : R$  2.69
Protetor FPS >= 30 : R$ 16.21
Repelente      : R$  9.24

SUPERMERCADO: KíBarato
Sabonete      : R$  2.48
Papel Higiênico : R$  8.36
Escova Dental  : R$  3.58
Creme Dental   : R$  2.80
Protetor FPS >= 30 : R$ 28.23
Repelente      : R$  8.02

SUPERMERCADO: Pop
Sabonete      : R$  1.97
Papel Higiênico : R$  7.92
Escova Dental  : R$  1.80
Creme Dental   : R$  2.37
Protetor FPS >= 30 : R$ 17.28
Repelente      : R$ 11.76
```

b)

```
PRODUTO: Sabonete
Supermercado Descontão : R$  3.39
Supermercado KíBarato  : R$  2.48
Supermercado Pop       : R$  1.97
Supermercado Mercadão  : R$  2.89
Supermercado SuperPrice : R$  2.60

PRODUTO: Papel Higiênico
Supermercado Descontão : R$  6.75
Supermercado KíBarato  : R$  8.36
Supermercado Pop       : R$  7.92
Supermercado Mercadão  : R$  9.43
Supermercado SuperPrice : R$  7.57

PRODUTO: Escova Dental
Supermercado Descontão : R$  1.69
Supermercado KíBarato  : R$  3.58
Supermercado Pop       : R$  1.80
Supermercado Mercadão  : R$  1.67
Supermercado SuperPrice : R$  3.88
```

33



Uma Solução: Exibir não tabular_{1/2}

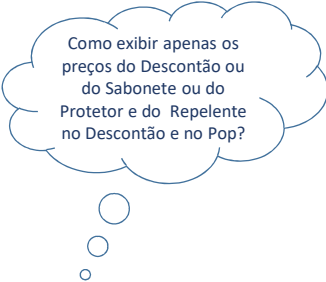
```
def exibeColunaAColuna(df):
    for (sm,v) in df.iteritems():
        print('\nSUPERMERCADO:', sm)
        for (prod,val) in v.iteritems():
            print('\t{:20s}: R$ {:.2f}'.format(prod, val))
    return

def ExibeLinhaALinha(df):
    for (prod,v) in df.iterrows():
        print('\nPRODUTO:', prod)
        for (sm,val) in v.iteritems():
            print('\tSupermercado {:12s}: R$ {:.2f}'.format(sm, val))
    return
```

34

Uma Solução: Exibir não tabular_{2/2}

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
print(dfPrHigMerc)
exibeLinhaALinha(dfPrHigMerc)
exibeColunaAColuna(dfPrHigMerc)
```



Como exibir apenas os
preços do Descontão ou
do Sabonete ou do
Protetor e do Repelente
no Descontão e no Pop?

35

Seleção de Itens

36

Problema: Selecionando Colunas

I. Como exibir apenas os preços do Descontão?

	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Produto					
Sabonete	3.39	2.48	1.97	2.09	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.80	1.67	3.88
Creme Dental	2.69	2.80	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Selecionar a Series da Coluna *Descontão*

37

Problema: Selecionando Linhas

II. Como exibir apenas preço do Sabonete ?

	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Produto					
Sabonete	3.39	2.48	1.97	2.09	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.80	1.67	3.88
Creme Dental	2.69	2.80	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Selecionar a Series da Linha *Sabonete*

38



Problema: Seleccionando Elementos

I. Como exibir apenas preço do Protetor e do Repelente no Descontão e no Pop?

	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Produto					
Sabonete	3.39	2.48	1.97	2.09	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.80	1.67	3.88
Creme Dental	2.69	2.80	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Selecionar as linhas *Protetor FPS >= 30* e *Repelente* das colunas *Descontão* e *Pop* (*dice*)

39



Seleção de Colunas

Sintaxe:

```
df[coluna] ou df.coluna
df[lista de colunas]
```

Retorna uma *Series* com os valores da *coluna* ou um *DataFrame* com os elementos da *lista de colunas*

```
dfNte:
  Nome  Matr  P1  P2  P3
0  Lalá   133  6.2  6.9  9.2
1  Lelé   131  6.5  2.7  3.0
2  Lili   135  1.3  4.6  6.5
```

```
>>>dfNte.P1
0    6.2
1    6.5
2    1.3
Name: P1, dtype: float64
```

```
>>>dfNte.p1
AttributeError: 'DataFrame' object has
no attribute 'p1'
```

```
>>>dfNte['P1']
0    6.2
1    6.5
2    1.3
Name: P1, dtype: float64
```

```
>>>dfNte[['Nome', 'P3']]
   Nome  P3
0  Lalá  9.2
1  Lelé  3.0
2  Lili  6.5
```

40

Aplicar um formato em uma Coluna

```
dfNte:   Nome  Matr  P1  P2  P3
0   Lalá   133  6.2  6.9  9.2
1   Lelé   131  6.5  2.7  3.0
2    Lili   135  1.3  4.6  6.5
```

```
dfNte['Nome']=dfNte['Nome'].apply('{:*^12s}'.format)
print(dfNte)
```

```

      Nome  Matr  P1  P2  P3
0  ****Lalá****   133  6.2  6.9  9.2
1  ****Lelé****   131  6.5  2.7  3.0
2  ****Lili****   135  1.3  4.6  6.5
```

41

Seleção de Linhas

Sintaxe: `df.loc[índice]` ou `df.loc[lista de índices]`
`df.iloc[posição]` ou `df.iloc[lista de posições]`

Retorna uma Serie com os valores da linha indexada por *índice* ou um *DataFrame* com os elementos da *lista de índices*. `.loc` para os índices criados, `.iloc` para a posição no índice

```
dfNtd:   Nome  P1  P2  P3
Matr
133   Lalá  6.2  6.9  9.2
131   Lelé  6.5  2.7  3.0
135    Lili  1.3  4.6  6.5
```

```
>>>dfNtd.loc[133]
Nome    Lalá
P1      6.2
P2      6.9
P3      9.2
Name: 133, dtype: object
```

```
>>>dfNtd.iloc[1]
Nome    Lelé
P1      6.5
P2      2.7
P3      3
Name: 131, dtype: object

>>> dfNtd.iloc[2:4]
      Nome  P1  P2  P3
Matr
135   Lili  1.3  4.6  6.5
```

42

Aplicar um formato em uma Linha

dfNtX

	Lalá	Lelé
Matr	133.0	131.0
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	3

← ???

```
dfNtX.loc['Matr']=dfNtX.loc['Matr'].apply('{:.0f}'.format)
print(dfNtX)
```

	Lalá	Lelé
Matr	133	131
P1	6.2	6.5
P2	6.9	2.7
P3	9.2	NaN

Observação: Os valores não são mais numéricos

43

Seleção de Elementos

Sintaxe: `df.loc[índice][coluna]` ou `df.loc[lista de índices][lista de colunas]*`
`df[coluna].loc[índice]` ou `df[lista de colunas].loc[lista de índices]*`

Retorna o valor do elemento indexado por *índice*, *coluna* ou uma nova Series com os elementos da *lista de índices/colunas*.

* Para posição no índice deve ser usado `.iloc`

dfNtd:	Nome	P1	P2	P3
Matr				
133	Lalá	6.2	6.9	9.2
131	Lelé	6.5	2.7	3.0
135	Lili	1.3	4.6	6.5

```
>>>dfNtd.loc[131]['P1']
6.5
>>>dfNtd.iloc[2]['P1']
1.3
>>>dfNtd.iloc[1:3]['P1']
Matr
131    6.5
135    1.3
Name: P1, dtype: float64
```

```
>>>dfNtd.loc[[133,135]][['P1','Nome']]
```

	P1	Nome
Matr		
133	6.2	Lalá
135	1.3	Lili

```
>>>dfNtd.iloc[1:3][['P1','Nome']]
```

	P1	Nome
Matr		
131	6.2	Lelé
135	1.3	Lili

44



Mãos na Massa

Considere o Dataframe G:

	Matr	P1	P2	P3
Lalá	133	6.2	6.9	9.2
Lelé	131	6.5	2.7	3.0
Lili	135	1.3	4.6	6.5

Qual o resultado das seguintes seleções?

- 1) `G[['P1', 'P3']].iloc[2]`
- 2) `G['Lalá']`
- 3) `G.T['Lalá']`
- 4) `G.T[['Lalá', 'Lili']].iloc[0:2].T`

- 1)

```
P1    1.3
P3    6.5
Name: Lili, dtype: float64
```

 Series
- 2)

```
KeyError: 'Lalá'
```
- 3)

```
Matr    133
P1       6.2
P2       6.9
P3       9.2
Name: Lalá, dtype: object
```

 Series
- 4)

```
      Matr    P1
Lalá   133    6.2
Lili   135    1.3
```

 DataFrame

45



Mãos na Massa: Seleção de Itens

Exibir:

- a) os preços dos produtos do supermercado Descontão
- b) O preço do sabonete nos diferentes supermercados
- c) O preço do repelente e do protetor solar nos supermercados Pop e Descontão

46



Uma Solução: Seleção de Itens

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
print(dfPrHigMerc['Descontão'])
print(dfPrHigMerc.loc['Sabonete'])
print(dfPrHigMerc.loc[['Protetor FPS >= 30','Repelente']][['Descontão','Pop']])
```

Produto	
Sabonete	3.39
Papel Higiênico	6.75
Escova Dental	1.69
Creme Dental	2.69
Protetor FPS >= 30	16.21
Repelente	9.24
Name: Descontão, dtype: float64	

Descontão	3.39
KiBarato	2.48
Pop	1.97
Mercadão	2.09
SuperPrice	2.60
Name: Sabonete, dtype: float64	

	Descontão	Pop
Produto		
Protetor FPS >= 30	16.21	17.28
Repelente	9.24	11.76

47



Problema: Alterando Linha

- I. Quando um produto não tem seu preço conferido em uma semana é mantido no arquivo o preço da última semana. Como nesta semana o preço do Repelente não foi conferido, altere o DataFrame para anular seu valor.

	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Produto					
Sabonete	3.39	2.48	1.97	2.09	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.80	1.67	3.88
Creme Dental	2.69	2.80	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

→ NaN

48



Problema: Inclusão de Linha/Coluna

- I. Nesta semana, pesquisaram, também, o preço do fio dental que está no arquivo FioDental.xlsx e um novo supermercado: Baratão, cujos preços estão no arquivo Baratao.xlsx. Como incluí-los no DataFrame?

	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Baratão
Produto						
Sabonete	3.39	2.48	1.97	2.09	2.60	2.60
Papel Higiénico	6.75	8.36	7.92	9.43	7.57	7.20
Escova Dental	1.69	3.58	1.80	1.67	3.88	1.50
Creme Dental	2.69	2.80	2.37	3.35	2.86	2.50
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37	15.30
Repelente	9.24	8.02	11.76	10.81	12.33	10.20
Fio Dental	3.50	2.80	2.90	2.85	3.10	3.80

Do arquivo Baratao.xlsx

Do arquivo FioDental.xlsx

49



Alterando ou Incluindo Linhas

Sintaxe:

`df.loc[índice]= valor ou Series`

Altera o valor/valores do elemento(s) indexado(s) por *índice/lista de índices*. Se o índice não existe, é incluído.

```
dfNtd:  Nome    P1    P2    P3
Matr
133    Lalá    6.2    6.9    9.2
131    Lelé    6.5    2.7    3.0
135    Lili    1.3    4.6    6.5
```

```
>>>s=pd.Series(['Loló',9.0,8,7],
               index=['Nome','P1','P2','P3'])
```

```
>>> s
Nome    Loló
P1      9.0
P2      8.0
P3      7.0
dtype: float64
```

```
>>>dfNtd.loc[222]=s; dfNtd
      Nome    P1    P2    P3
Matr
133    Lalá    6.2    6.9    9.2
131    Lelé    6.5    2.7    3.0
135    Lili    1.3    4.6    6.5
222    Loló    9.0    8.0    7.0
```

```
>>>dfNtd.loc[222]=['?',0,0,0]; dfNtd
      Nome    P1    P2    P3
Matr
133    Lalá    6.2    6.9    9.2
131    Lelé    6.5    2.7    3.0
135    Lili    1.3    4.6    6.5
222     ?     0.0    0.0    0.0
```

50



Excluindo Linhas

Sintaxe: `df.drop(índice)` ou `df.drop(lista de índices) *`
`df.dropna() *`

* com `inplace=True`, realiza a operação na Series, não cria uma cópia

`drop`: retorna uma cópia do DataFrame sem a linha/linhas especificadas

`dropna`: retorna uma cópia do DataFrame sem as linhas que tenham colunas com NaN. Com parâmetro `how='all'`, só remove as linhas em que todas as colunas são NaN

```
dfNtd: Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
131 NaN 6.5 2.7 3.0
135 Lili 1.3 NaN 6.5
```

```
>>>dfNtd.dropna()
      Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
>>>dfNtd.dropna(how='all')
      Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
131 NaN 6.5 2.7 3.0
135 Lili 1.3 NaN 6.5
```

```
>>>dfNtd.drop(135)
```

```
      Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
131 NaN 6.5 2.7 3.0
```

```
>>>dfNtd.drop([135,133])
```

```
      Nome P1 P2 P3
Matr
131 NaN 6.5 2.7 3.0
```

51



Alterando ou Incluindo Colunas

Sintaxe: `df[coluna]= valor ou Series`

Altera o valor/valores do(s) elemento(s) indexado(s) por *coluna/lista de colunas*. Se a coluna não existe, é incluída.

```
dfNtd: Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
131 Lelé 6.5 2.7 3.0
135 Lili 1.3 4.6 6.5
```

```
sT:
133 1.5
135 0.3
dtype: float64
```

```
>>>>>dfNtd['T1']=sT
      Nome P1 P2 P3 T1
Matr
133 Lalá 6.2 6.9 9.2 1.5
131 Lelé 6.5 2.7 3.0 NaN
135 Lili 1.3 4.6 6.5 0.3
```

```
dfNtd['T1']=dfNtd['T1'].fillna(0)
dfNtd['G1']=dfNtd['P1'] + dfNtd['T1']
>>>dfNtd
```

```
      Nome P1 P2 P3 T1 G1
Matr
133 Lalá 6.2 6.9 9.2 1.5 7.7
131 Lelé 6.5 2.7 3.0 0.0 6.5
135 Lili 1.3 4.6 6.5 0.3 1.6
```

É possível criar novas colunas a partir de outras colunas!

52



Excluindo Colunas

Sintaxe: `df.drop(coluna,axis=1)` ou `df.drop(lista de colunas,axis=1)*`
`df.dropna(axis=1)*`

* com `inplace=True`, realiza a operação na Series, não cria uma cópia

`drop`: retorna uma cópia do DataFrame sem a coluna/lcolunas especificadas

`dropna`: retorna uma cópia do DataFrame sem as colunas que tenham linhas com NaN.

Com parâmetro `how='all'`, só remove as linhas em que todas as colunas são NaN

```
dfNtd: Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
131 NaN 6.5 2.7 3.0
135 Lili 1.3 NaN 6.5
```

```
>>>dfNtd.dropna(axis=1)
```

```
      P1 P3
Matr
133  6.2 9.2
131  6.5 3.0
135  1.3 6.5
```

```
>>>dfNtd.drop('P3',axis=1)
```

```
      Nome P1 P2
Matr
133  Lalá 6.2 6.9
131  NaN 6.5 2.7
135  Lili 1.3 NaN
```

```
>>>dfNtd.drop(['P2','P3'],axis=1)
```

```
      Nome P1
Matr
133  Lalá 6.2
131  NaN 6.5
135  Lili 1.3
```

53



Ordenar DataFrame Pelos Valores: sort_values

Sintaxe: `df.sort_values(by,axis=0,ascending=True,na_position='last')*`

Retorna uma cópia do DataFrame ordenado pelos valores

by = nomes ou lista de nomes para ordenar. Se `axis=0` os nomes se referem a colunas. Se `axis=1` os nomes se referem a linhas

na_position = 'last' / 'first', coloca NaN no final/ início

* com `inplace=True`, realiza a operação na Series, não cria uma cópia

```
dfNtd: Nome P1 P2 P3
Matr
133 Lalá 6.2 6.9 9.2
131 Lelé 6.5 2.7 3.0
135 Lili 1.3 4.6 6.5
```

```
>>>dfNtdOrd=dfNtd.sort_values('P3')
>>>dfNtdOrd
```

```
      Nome P1 P2 P3
Matr
131  Lelé 6.5 2.7 3.0
135  Lili 1.3 4.6 5.0
133  Lalá 6.2 6.9 9.2
```

```
>>>dfNtdOrd=dfNtd.sort_values('P3',ascending=False)
>>>dfNtdOrd
```

```
      Nome P1 P2 P3
Matr
133  Lalá 6.2 6.9 9.2
135  Lili 1.3 4.6 5.0
131  Lelé 6.5 2.7 3.0
```

54



Ordenar DataFrame Pelos Índices: sort_index

Sintaxe: `df.sort_index(axis=0, level=None, ascending=True, na_position='last', sort_remaining=True, by=None) *`

Retorna uma cópia do DataFrame ordenado pelos labels do índice

* com `inplace=True`, realiza a operação no DataFrame, não cria uma cópia

```
dfNtd: Nome  P1  P2  P3
Matr
133  Lalá  6.2  6.9  9.2
131  Lelé  6.5  2.7  3.0
135  Lili  1.3  4.6  6.5
```

```
>>>dfNtOrd=dfNtd.sort_index()
>>>dfNtOrd
```

```
Nome  P1  P2  P3
Matr
131  Lelé  6.5  2.7  3.0
133  Lalá  6.2  6.9  9.2
135  Lili  1.3  4.6  5.0
```

```
>>>dfNtOrd=dfNtd.sort_index(ascending=False)
```

```
>>>dfNtOrd
Nome  P1  P2  P3
Matr
135  Lili  1.3  4.6  5.0
133  Lalá  6.2  6.9  9.2
131  Lelé  6.5  2.7  3.0
```

```
>>>dfNtOrd=dfNtd.sort_index(axis=1,ascending=False)
```

```
>>>dfNtOrd
P3  P2  P1  Nome
Matr
133  9.2  6.9  6.2  Lalá
131  3.0  2.7  6.5  Lelé
135  5.0  4.6  1.3  Lili
```

55



DataFrame: fillna

Sintaxe: `df.fillna(value=None, axis=None, ascending=True) *`

Retorna uma cópia do DataFrame substituindo valores Nan

value = scalar, dict, Series, or DataFrame

alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). (values not in the dict/Series/DataFrame will not be filled).

axis = 0 ou 1

* com `inplace=True`, realiza a operação no DataFrame, não cria uma cópia

```
dfNt:
Nome  P1  P2  P3
Matr
133  Lalá  6.2  6.9  9.2
131  Lelé  6.5  2.7  3.0
135  Lili  1.3  4.6  5.0
134  Lolo  NaN  NaN  6.0
```

```
>>>dfFill=dfNt.fillna(0)
```

```
>>>dfFill
Nome  P1  P2  P3
Matr
133  Lalá  6.2  6.9  9.2
131  Lelé  6.5  2.7  3.0
135  Lili  1.3  4.6  5.0
134  Lolo  0.0  0.0  6.0
```

```
>>>dfF=dfNt.fillna({'P1':0,'P2':2})
```

```
>>>dfF
Nome  P1  P2  P3
Matr
133  Lalá  6.2  6.9  9.2
131  Lelé  6.5  2.7  3.0
135  Lili  1.3  4.6  5.0
134  Lolo  0.0  2.0  6.0
```

56



Mãos na Massa: Alteração/Inclusão de itens

- I. Quando um produto não tem seu preço conferido em uma semana é mantido no arquivo o preço da última semana. Como nesta semana o preço do Repelente não foi conferido, altere o DataFrame `dfPrHigMerc` para anular seu valor.
- II. Nesta semana, pesquisaram um novo supermercado: Barato, cujos preços estão no arquivo `Barato.xlsx` e também um novo produto: o fio dental, cujo preço nos mercados está no arquivo `FioDental.xlsx`. Incluir este novo supermercado e produto no DataFrame `dfPrHigMerc`

Observação: Desenvolva os itens de forma independente

57



Uma Solução: Anulando uma linha

'''Quando um produto não tem seu preço conferido em uma semana é mantido no arquivo o preço da última semana. Como nesta semana o preço do Repelente não foi conferido, altere o DataFrame para anular seu valor.'''

'''

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
dfPrHigMerc.loc['Repelente']= None
print(dfPrHigMerc)
```

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.80	1.67	3.88
Creme Dental	2.69	2.80	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37
Repelente	NaN	NaN	NaN	NaN	NaN

Qual o resultado de
.dropna (axis=1) neste
DF?

Qual o resultado de
.dropna() neste DF?

58



Uma Solução: Anulando uma coluna

''' Quando um supermercado não tem seu preço conferido em uma semana é mantido no arquivo o preço da última semana. Como nesta semana o supermercado Descontão não foi conferido, altere o DataFrame dfProd para anular seu valor.

'''

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
dfPrHigMerc['Descontão']= None
print(dfPrHigMerc)
```

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	None	2.48	1.97	2.09	2.6
Papel Higiénico	None	8.36	7.92	9.43	7.57
Escova Dental	None	3.58	1.8	1.67	3.88
Creme Dental	None	2.8	2.37	3.35	2.86
Protetor FPS >= 30	None	28.23	17.28	27.8	24.37
Repelente	None	8.02	11.76	10.81	12.33

Qual o resultado de
.dropna(axis=1) neste
DF?

Qual o resultado de
.dropna() neste DF?

59



Uma Solução: Incluindo novo produto e novo supermercado

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
sBar=pd.read_excel("Baratao.xlsx",header=None,squeeze=True, decimal=',',index_col=0)
sFio=pd.read_excel("FioDental.xlsx",header=None,squeeze=True, decimal=',',index_col=0)
dfPrHigMerc['Baratão']=sBar
dfPrHigMerc.loc['FioDental']=sFio
print(dfProd)
```

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Baratão
Sabonete	3.39	2.48	1.97	2.09	2.60	2.60
Papel Higiénico	6.75	8.36	7.92	9.43	7.57	7.20
Escova Dental	1.69	3.58	1.80	1.67	3.88	1.50
Creme Dental	2.69	2.80	2.37	3.35	2.86	2.50
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37	15.30
Repelente	9.24	8.02	11.76	10.81	12.33	10.20
Fio Dental	3.50	2.80	2.90	2.85	3.10	3.80

Gera o mesmo
DF se incluir a
linha antes da
coluna?

Como incluir
neste DF os
produtos de
Limpeza?

60

Unindo DataFrames

.concat/.append/.merge/.join

<https://pandas.pydata.org/pandas-docs/stable/merging.html>

61

Problema: Unindo DataFrames

- I. Como incluir os produtos de Limpeza que estão na planilha ProdMercLimpeza do arquivo PrecosProdutosSuperMercados.xlsx, construindo um único DataFrame com todos os produtos do kit?

Planilha ProdHigieneMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Planilha ProdLimpezaMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

62

Método .concat()

A aplicação do método **.concat** sobre DataFrames, pode realizar tanto a união como a interseção sobre um dos eixos (linha/coluna)

pandas.concat: concatena uma lista ou dicionário de objetos de tipos similares de acordo com o que for estabelecido pelos parâmetros a ser feito com os outros índices

A concatenação é realizada por um dos eixos e as seguintes formas de lidar com o outro eixo estão disponíveis:

- a) `join='outer'`: é o padrão e cria a união, classificada, de todos os itens
- b) `join='inner'`: cria a interseção de todos os itens
- c) `join_axes = índice`: usa o índice especificado

63

União - Eixo coluna (axis=0)

Sintaxe:

```
pd.concat([df1,...dfn])
```

```
ldf = [df1, df2, df3]
result = pd.concat(ldf)
```

df1				
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2				
	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3				
	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

result				
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

64

União - Eixo linha (axis=1)

Sintaxe:

```
pd.concat([df1,...dfn],axis=1)
```

```
ldf = [df1, df4]
result = pd.concat(ldf,axis=1)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Os índices das linhas são unidos e ordenados

result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

65

União - Eixo coluna com índices (keys=[...])

Sintaxe:

```
pd.concat([df1,...dfn],keys=[v11,...,v1n]) ou
pd.concat({chv1:df1,...,chvn:dfn})
```

```
ldf = [df1, df2, df3]
result1 = pd.concat(ldf,keys=['x','y','z'])
result2 = pd.concat({'x':df1,'y':df2,'z':df3})
print(result1.loc['z'])
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

result1

		A	B	C	D
x	0	A0	B0	C0	D0
x	1	A1	B1	C1	D1
x	2	A2	B2	C2	D2
x	3	A3	B3	C3	D3
y	4	A4	B4	C4	D4
y	5	A5	B5	C5	D5
y	6	A6	B6	C6	D6
y	7	A7	B7	C7	D7
z	8	A8	B8	C8	D8
z	9	A9	B9	C9	D9
z	10	A10	B10	C10	D10
z	11	A11	B11	C11	D11

result2

		A	B	C	D
x	0	A0	B0	C0	D0
x	1	A1	B1	C1	D1
x	2	A2	B2	C2	D2
x	3	A3	B3	C3	D3
y	4	A4	B4	C4	D4
y	5	A5	B5	C5	D5
y	6	A6	B6	C6	D6
y	7	A7	B7	C7	D7
z	8	A8	B8	C8	D8
z	9	A9	B9	C9	D9
z	10	A10	B10	C10	D10
z	11	A11	B11	C11	D11

66



União - Eixo linha com índices

(axis=1, keys=[...])

Sintaxe: `pd.concat([df1,...dfn], keys=[v11,...v1n], axis=1)` ou
`pd.concat({chv1:df1,...,chvn:dfn}, axis=1)`

```
ldf = [df1, df4]
result1 = pd.concat(ldf, keys=['x', 'y'], axis=1)
result2 = pd.concat({'x':df1, 'y':df4}, axis=1)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

result1

	x	x	x	x	y	y	y
	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

result2

	x	x	x	x	y	y	y
	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

67



Interseção - Eixo coluna

(join='inner')

Sintaxe: `pd.concat([df1,...dfn], join='inner')`

```
ldf = [df1, df4]
result = pd.concat(ldf, join='inner')
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

result

	B	D
0	B0	D0
1	B1	D1
2	B2	D2
3	B3	D3
2	B2	D2
3	B3	D3
6	B6	D6
7	B7	D7

68



Interseção - Eixo linha (axis=1, join='inner')

Sintaxe:

```
pd.concat([df1,...dfn],axis=1, join='inner')
```

```
ldf = [df1, df4]
result = pd.concat(ldf,axis=1,join='inner')
```

	df1			
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

	df4		
	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

	result1						
	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

69



Pelo índice de um DF - Eixo Coluna (join_axes=df.indice)

Sintaxe:

```
pd.concat([df1,...dfn],join_axes=[df.index/columns])
```

```
ldf = [df1, df4]
result = pd.concat(ldf,join_axes=[df1.columns])
```

	df1			
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

	df4		
	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

	result			
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
2	NaN	B2	NaN	D2
3	NaN	B3	NaN	D3
6	NaN	B6	NaN	D6
7	NaN	B7	NaN	D7

70



Unir Series como Colunas de DF

Sintaxe:

```
pd.concat([df, s1, ..., sn], axis=1, ignore_index=True)
```

As *Series* são transformadas em *DataFrames* com o nome da *Series* como nome da coluna, se a opção `ignore_index=True` não estiver presente

```
s1 = pd.Series(['X0', 'X1', 'X2', 'X3'], name='X')
s2 = pd.Series(['_0', '_1', '_2', '_3'])
result1 = pd.concat([df1, s1], axis=1)
result2 = pd.concat([df1, s2, s2], axis=1, ignore_index=True)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2

s1

	X
0	X0
1	X1
2	X2

s2

0	_0
1	_1
2	_2

result1

	A	B	C	D	X
0	A0	B0	C0	D0	X0
1	A1	B1	C1	D1	X1
2	A2	B2	C2	D2	X2

result2

	0	1	2	3	4	5
0	A0	B0	C0	D0	_0	_0
1	A1	B1	C1	D1	_1	_1
2	A2	B2	C2	D2	_2	_2

71



Pelo índice de um DF - Eixo Linha (axis=1, join_axis=df.index)

Sintaxe:

```
pd.concat([df1, ..., dfn], axis=1, join_axes=[df.index/columns])
```

```
ldf = [df1, df4]
result = pd.concat(ldf, axis=1, join_axes=df1.index)
```

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

72

Mãos na Massa: Unindo DataFrames

- Construir um DataFrame com os produtos de Higiene que estão na planilha ProdHigieneMerc e com os produtos de Limpeza que estão na planilha ProdLimpezaMerc, ambas no arquivo PrecosProdutosSuperMercados.xlsx

Planilha ProdHigieneMerc

Produto	Descontão	KiBarato	Pop	Mercado	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Crema Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Planilha ProdLimpezaMerc

Produto	Descontão	KiBarato	Pop	Mercado	SuperPrice
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

Produto	Descontão	KiBarato	Pop	Mercado	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Crema Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

73

Uma Solução: Unindo DataFrames

```
import pandas as pd
dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',
    sheetname='ProdHigieneMerc',index_col=0)
dfPrLimMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",
    sheetname='ProdLimpezaMerc', decimal=',',index_col=0)
dfPrMerc=pd.concat([dfPrHigMerc,dfPrLimMerc])
print(dfPrMerc)
```

Produto	Descontão	KiBarato	Pop	Mercado	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Crema Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33
Detergente Liquido	1.34	1.87	1.22	1.73	2.1
Lã de Aço	0.68	2.33	2.74	2.9	2.26
Sabão em Pó	6.21	9.73	6.76	7.98	9.35
Desinfetante	3.26	2.72	1.37	2.09	2.34

Como exibir o preço médio de cada produto ou do mercado, ou o menor preço para cada produto?

74

Descrição e Sumarização

75

Descrição e Sumarização

Medidas de Tendência Central

Média:

`df.mean()` ^[1]

```
>>>df.mean()
x    13.333333
y    20.000000
z    16.666667

>>>df.mean(axis=1)
p1    20.0
p2    10.0
p3    20.0
```

Mediana:

`df.median()` ^[1]

```
>>>df.median()
x    10.0
y    20.0
z    10.0
```

Moda:

`df.mode()` ^[1]


```
>>>df.mode()
      x  y  z
0  10.0 10 10.0
1   NaN 20  NaN
2   NaN 30  NaN
```

df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

1- Com axis=1, operação por linha
2 - Operação aceita nos atributos index e columns

76



Descrição e Sumarização

Medidas de Tendência Central

<p>Máximo: <code>df.max()</code> ^{[1][2]}</p> <p>Mínimo: <code>df.min()</code> ^{[1][2]}</p> <p>Índice 1º Mínimo: <code>df.idxmin()</code></p> <p>Índice 1º Máximo: <code>df.idxmax()</code></p>	<pre>>>>df.max(axis=1) p1 30 p2 10 p3 30 dtype: int64 >>>df.min() x 10 y 10 z 10 dtype: int64</pre>
<p>Quantil:</p> <p><code>df.quantile(q=%)</code> ^[1]</p> <p style="padding-left: 40px;">padrão q=0.5</p>	<pre>>>>df.quantile(axis=1) p1 20.0 p2 10.0 p3 20.0</pre>


df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

1- Com axis=1, operação por linha

2 - Operação aceita nos atributos index e columns

77



Descrição e Sumarização

Medidas de Dispersão

<p>Variância:</p> <p><code>df.var()</code> ^[1]</p>	<pre>>>>df.var() x 33.333333 y 100.000000 z 133.333333 dtype: float64</pre>
<p>Desvio Padrão:</p> <p><code>df.std()</code> ^[1]</p>	<pre>>>>df.std() x 5.773503 y 10.000000 z 11.547005 dtype: float64</pre>


df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

1- Com axis=1, operação por linha

2 - Operação aceita nos atributos index e columns

78




Descrição e Sumarização

Totalizações

<p>Soma:</p> <pre>df.sum() [1]</pre>	<pre>>>>df.sum() x 40 y 60 z 50 >>>df.sum(axis=1) p1 60 p2 30 p3 60</pre>	<pre>df: x y z p1 10 20 30 p2 10 10 10 p3 20 30 10</pre>
<p>Quantidade:</p> <pre>df.count() [1]</pre>	<pre>>>> df.count() x 3 y 3 z 3 >>>df.count(axis=1) p1 3 p2 3 p3 3</pre>	
<p>Contagem de valores exclusivos:</p> <pre>df.index.value_counts() df.columns.value_counts()</pre>	<pre>>>>df.index.value_counts() p2 1 p3 1 p1 1 >>>df.columns.value_counts() z 1 x 1 y 1</pre>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> 1- Com axis=1, operação por linha </div>

79



Descrição e Sumarização

Resumo

<p>Resumo:</p> <pre>df.describe()</pre>	<pre>>>>df.describe() count x y z mean 3.000000 20.0 16.666667 std 5.773503 10.0 11.547005 min 10.000000 10.0 10.000000 25% 10.000000 15.0 10.000000 50% 10.000000 20.0 10.000000 75% 15.000000 25.0 20.000000 max 20.000000 30.0 30.000000</pre>	<pre>df: x y z p1 10 20 30 p2 10 10 10 p3 20 30 10</pre>
--	---	---

80



Problema: Sumarização por Linha

- I. Exibir o preço médio de cada produto.



	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Baratão	
Produto							
Sabonete	3.39	2.48	1.97	2.09	2.60	2.60	⇒ média
Papel Higiênico	6.75	8.36	7.92	9.43	7.57	7.20	⇒ média
Escova Dental	1.69	3.58	1.80	1.67	3.88	1.50	⇒ média
Creme Dental	2.69	2.80	2.37	3.35	2.86	2.50	⇒ média
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37	15.30	⇒ média
Repelente	9.24	8.02	11.76	10.81	12.33	10.20	⇒ média
Fio Dental	3.50	2.80	2.90	2.85	3.10	3.80	⇒ média

- II. Exibir o preço médio dos produtos do Descontão e do Baratão formatados com duas casas decimais
- III. Exibir de forma tabular o preço médio, mínimo, máximo, a variância e o desvio padrão do preço dos produtos

81



Uma Solução: Sumarização por Linha

```
def resumosProdutos(df):
    print("\n\nResumos Produtos\n")
    print("\nPreço Médio: ", df.mean(axis=1))
    print("\nPreço Médio no Descontão e Baratão\n",
          dfProd[['Descontão','Baratão']].mean(axis=1).map('{:.2f}'.format))
    sMedia=df.mean(axis=1)
    sStd =df.std(axis=1)
    sVar = df.var(axis=1)
    sMin=df.min(axis=1)
    sMax=df.max(axis=1)
    dfRes=pd.DataFrame([sMedia,sVar,sStd,sMin,sMax],
                       index=['Média','Variância','Desvio','Mínimo','Máximo'])
    print(dfRes.T)
    return
```

82



Problema: Sumarização por Coluna

- I. Exibir os preços individuais dos produtos e o preço médio de cada mercado.

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Baratão
Sabonete	3.39	2.48	1.97	2.09	2.60	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57	7.20
Escova Dental	1.69	3.58	1.80	1.67	3.88	1.50
Creme Dental	2.69	2.80	2.37	3.35	2.86	2.50
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37	15.30
Repelente	9.24	8.02	11.76	10.81	12.33	10.20
Fio Dental	3.50	2.80	2.90	2.85	3.10	3.80
	média	média	média	média	média	média

83



Uma Solução: Sumarização por Coluna

```
def resumosMercados(df):
    sMed= df.mean().map('{:.2f}'.format)
    sMed.rename(index='Média', inplace=True) #dá nome a Series
    dfMed=pd.concat([df.T, sMed], axis=1) # concatena a coluna Média
    print('\nPreço Médio de cada supermercado\n: ', dfMed)
    return
```

Preço Médio de cada supermercado

Mercado	Sabonete	Papel Higiênico	Escova Dental	Creme Dental	Protetor FPS >= 30	Repelente	Média
Descontão	3.39	6.75	1.69	2.69	16.21	9.24	6.66
KiBarato	2.48	8.36	3.58	2.8	28.23	8.02	8.91
Pop	1.97	7.92	1.8	2.37	17.28	11.76	7.18
Mercadão	2.09	9.43	1.67	3.35	27.8	10.81	9.19
SuperPrice	2.6	7.57	3.88	2.86	24.37	12.33	8.94

84



Problema: Totalizações

- II. Exibir o preço individual de cada produto e o total a pagar pelo kit higiene da cesta básica em cada um dos supermercados, considerando apenas uma unidade de cada produto
- III. Indicar qual supermercado tem menor preço total (considerando uma unidade de cada item).

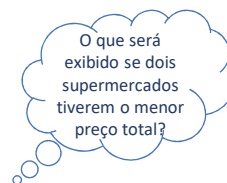
Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Baratão
Sabonete	3.39	2.48	1.97	2.09	2.60	2.60
Papel Higiênico	6.75	8.36	7.92	9.43	7.57	7.20
Escova Dental	1.69	3.58	1.80	1.67	3.88	1.50
Creme Dental	2.69	2.80	2.37	3.35	2.86	2.50
Protetor FPS >= 30	16.21	28.23	17.28	27.80	24.37	15.30
Repelente	9.24	8.02	11.76	10.81	12.33	10.20
Fio Dental	3.50	2.80	2.90	2.85	3.10	3.80
	soma	soma	soma	soma	soma	soma

85



Uma Solução: Totais e menor preço

```
def resumosMercados(df):
    sTot= df.sum()
    sTot.rename(index='Total',inplace=True) #dá nome a Series
    dfTot=pd.concat([df.T,sTot.apply('{:.2f}'.format)],axis=1)
    print(dfTot)
    print("\nMenor Preço Total R${} no supermercado{}\n".format(sTot.min(),sTot.idxmin()))
    return
```



Mercado	Sabonete	Papel Higiênico	Escova Dental	Creme Dental	Protetor FPS >= 30	Repelente	Total
Descontão	3.39	6.75	1.69	2.69	16.21	9.24	39.97
KiBarato	2.48	8.36	3.58	2.8	28.23	8.02	53.47
Pop	1.97	7.92	1.8	2.37	17.28	11.76	43.10
Mercadão	2.09	9.43	1.67	3.35	27.8	10.81	55.15
SuperPrice	2.6	7.57	3.88	2.86	24.37	12.33	53.61

Menor Preço Total R\$ 39.97 no Supermercado Descontão

86

Filtros

87

Problema: Seleção Condicional

Produto	Desconto	KiBarato	Pop	Mercado	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiénico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

- I. Exibir quais produtos tem preço inferior a R\$ 3,70 em todos os mercados

Filtro: Preço menor que R\$ 3,70

Retorna: Como o filtro é aplicado sobre as linhas, o DF resultante possui NaN nos elementos onde preço >= 3,70

Solução 1: Eliminar linhas que tenham valores NaN

Solução 2: Selecionar as linhas que tenham todos os elementos ≠ NaN

88



Problema: Seleção Condicional

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

I. Exibir os produtos que no Descontão são mais baratos que no Pop

Filtro: Preço do Descontão < preço do Pop

Retorna: Series com True onde o Descontão tem preço menor que o Pop e False caso contrário

Solução 1: Filtrar novamente o dfProd, com o resultado da Series

Retorna um DF com todas as colunas e apenas as linhas onde a Series é True

Solução 2: Filtrar novamente o dfProd com o resultado da Series e selecionar do DF resultante apenas as colunas dos dois supermercados

Retorna um DF com as colunas desejadas e as linhas onde a Series é True

89



Seleção condicional em DataFrames

Sintaxe:

`df operador_lógico condição`

Conectivos Lógicos:
e: & ou: | não: ~

Retorna um novo *DataFrame/Series* com valores booleanos True/False.

Pode-se usar o *DataFrame/Series* de booleanos para filtrar os itens selecionados (com valor True).

Normalmente utilizado sobre uma coluna ou linha do *DataFrame*

```
df:
  x  y  z
p1 10 20 30
p2 10 10 10
p3 20 30 10
```

```
>>>dfB = df > 10
```

```
>>>dfB
```

```

      x      y      z
p1 False  True  True
p2 False False False
p3  True  True False
```

```
>>>df[dfB]
      x      y      z
p1  NaN  20.0  30.0
p2  NaN   NaN   NaN
p3  20.0  30.0   NaN
```

```
>>> df[df['x']==10]
```

```

      x      y      z
p1  10     20     30
p2  10     10     10
```

```
p1 True
p2 True
p3 False
Name: x, dtype:bool
```

90



Filtro: método .isin

Sintaxe:

```
df.isin(lista de valores)
df.índice.isin(lista de índices)
```

Retorna um *DataFrame* com True onde o elemento do *DataFrame* é lista de valores.

Para o *index* retorna um vetor booleano, considerando os *labels* de linha e para *columns*, considerando os labels das colunas.

df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

```
>>>df.isin([10,20])
```

	x	y	z
p1	True	True	False
p2	True	True	True
p3	True	False	True

```
>>>df[df.isin([10,20])]
```

	x	y	z
p1	10	20.0	NaN
p2	10	10.0	10.0
p3	20	NaN	10.0

```
>>>df[df.index.isin(['p1','p34'])]
```

	x	y	z
p1	10	20	30

```
>>>sF = df.columns.isin(['x','y','k'])
```

```
>>>df[sF]
```

	x	y	z
p1	10	20	30
p2	10	10	10

[True, True, False]

```
>>>df.T[sF].T
```

	x	y
p1	10	10
p2	10	10
p3	20	30

91



Simplificando os filtros com .query

Sintaxe:

```
df.query('expressão')
```

Conectivos Lógicos:
e: & /and ou: |/or não: ~/not

Seleciona as linhas com os valores que satisfazem a expressão, retornando um *DataFrame*

➤ O *index* e o *columns* podem ser utilizados na expressão. A expressão utiliza as colunas do DF

df:

	x	y	z
p1	10	20	30
p2	10	10	10
p3	20	30	10

Selecionando as linhas onde x <=y e y<=z

```
>>> df[(df['x'] <= df['y'] &
        (df['y'] <= df['z']))]
```

	x	y	z
p1	10	20	30
p2	10	10	10

com filtro

```
>>> df.query('x <= y <= z')
```

	x	y	z
p1	10	20	30
p2	10	10	10

com query

92



Simplificando os filtros com .query

```
df:
  x  y  z
p1 10 20 30
p2 10 10 10
p3 20 30 10
```

Selecionando as linhas onde o valor de x ∈ à coluna z

```
>>> df[df['x'].isin(df['z'])]
```

```
  x  y  z
p1 10 20 30
p2 10 10 10
```

com filtro

```
>>> df.query('x in z')
```

```
  x  y  z
p1 10 20 30
p2 10 10 10
```

com query

Selecionando as linhas onde o valor de x ∈ z e y < z

```
>>>df[(df['x'].isin(df['z']) &
      (df['y'] < df['z']))]
```

```
  x  y  z
p1 10 20 30
```

com filtro

```
>>>df.query('x in z and y < z')
```

```
  x  y  z
p1 10 20 30
```

com query

93



Alteração com Filtro

Sintaxe: `df[critério] = valor ou lista de valores`

critério: produz um *DataFrame/Series* booleano

Altera o(s) valor(es) do(s) elemento(s) indexado(s) por posições onde o valor é True

```
df:
  x  y  z
p1 10 20 30
p2 10 10 10
p3 20 30 10
```

```
>>>df[df['z']>10]=15
```

```
>>>df
```

```
  x  y  z
p1 15 15 15
p2 10 10 10
p3 20 30 10
```

```
p1 True
p2 False
p3 False
```

```
>>>df[df.isin([15,30])]=18; df
```

```
  x  y  z
p1 18 18 18
p2 10 10 10
p3 20 18 10
```

```
  x  y  z
p1 True True True
p2 False False False
p3 False True False
```

```
>>>F=df.T['p3']<=10; df.T[F]
```

```
  p1  p2  p3
z  18  10  10
```

```
F:
x False
y False
z True
Name: p3,
dtype: bool
```

```
>>>df.T[F]-=3; df
```

```
  x  y  z
p1 18 18 15
p2 10 10 7
p3 20 18 7
```

94



Solução 1: Produtos com preço < 3,70 em todos os mercados

```
import pandas as pd
dfProd=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
print('\nPRODUTOS COM PREÇO INFERIOR A 3,70 EM TODOS OS SM \n: ')
print(dfProd[dfProd<3.70].dropna())
```

Produto	Desconto KiBarato		Pop Mercado		SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Creme Dental	2.69	2.8	2.37	3.35	2.86

95



Solução 2: Produtos com preço < 3,70 em todos os mercados

```
import pandas as pd
dfProd=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
print('\nPRODUTOS COM PREÇO INFERIOR A 3,70 EM TODOS OS SM \n: ')
dfTodos=dfProd.loc[dfProd[dfProd<3.70].notnull().all(axis=1)]
print(dfTodos)
```

Produto	Desconto KiBarato		Pop Mercado		SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Creme Dental	2.69	2.8	2.37	3.35	2.86

96



Uma Solução: Produtos onde o Descontão é mais barato que o Pop

```
# Solução 1
import pandas as pd
dfProd=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
print('\n\nPRODUTOS ONDE DESCONTAO É MAIS BARATO QUE O POP\n "\n: ')
print(dfProd[dfProd['Descontão']<dfProd['Pop']])
```

Produto	Descontão KiBarato		Pop Mercadão SuperPrice	
Papel Higiênico	6.75	8.36	7.92	9.43
Escova Dental	1.69	3.58	1.80	1.67
Protetor FPS >= 30	16.21	28.23	17.28	27.8
Repelente	9.24	8.02	11.76	10.81

97



Uma Solução: Produtos onde o Descontão é mais barato que o Pop

```
# Solução 2
import pandas as pd
dfProd=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',',index_col=0)
print('\n\nPRODUTOS ONDE DESCONTAO É MAIS BARATO QUE O POP\n "\n: ')
df = dfProd[dfProd['Descontão']<dfProd['Pop']]
print(df[['Descontão','Pop']])
```

Produto	Descontão	Pop
Papel Higiênico	6.75	7.92
Escova Dental	1.69	1.80
Protetor FPS >= 30	16.21	17.28
Repelente	9.24	11.76

98

Operações com DataFrames

99

Problema: Operações Aritméticas

- I. Exibir o preço total a pagar pelo kit higiene da cesta básica em cada um dos supermercados, considerando a quantidade exigida de cada produto no kit limpeza e indicar qual supermercado tem menor preço total

Planilha ProdHigieneMerc

Produto	Desconto	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Planilha Produtos

Produto	Categoria	Unidade	Quantidade
Sabonete	Higiene	90 g	3
Papel Higiênico	Higiene	Pct 4 unidades	2
Escova Dental	Higiene	Unidade	2
Creme Dental	Higiene	90 g	1
Protetor Solar FPS >= 30	Higiene	100 ml	1
Repelente de insetos	Higiene	300 ml	1
Detergente Liquido	Limpeza	500 ml	1
Lã de Aço	Limpeza	Pct 4 unidades	1
Sabão em Pó	Limpeza	500 g	1
Desinfetante	Limpeza	1000 ml	1

Produto	Quantidade	Desconto	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3	3.39	2.48	1.97	2.09	2.6
Papel Higiênico	2	6.75	8.36	7.92	9.43	7.57
Escova Dental	2	1.69	3.58	1.8	1.67	3.88
Creme Dental	1	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	1	16.21	28.23	17.28	27.8	24.37
Repelente	1	9.24	8.02	11.76	10.81	12.33

100



Operações Aritméticas

Sintaxe: `df.metodoOperação(obj, fill_value=valor)`

obj pode ser um *DataFrame* ou uma *Series* ou um escalar. Os dados são alinhados pelas colunas e pelos índices. Retorna um *DataFrame* com a união das colunas e dos *labels* das linhas. Se o argumento *fill_value* está presente e não há sobreposição nos índices, utiliza *valor* para o cálculo, senão o valor é NaN

`df1.add(df2, fill_value=0)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

dfR	a	b	c	d
A	11	22	33	100
B	44	55	66	200
C	77	88	99	300

`df1.sub(df2, fill_value=0)`

df1	a	b	c	d
A	1	2	3	
B	4	5	6	
C	7	8	9	

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

dfR	a	b	c	d
A	-9	-18	-27	-100
B	-36	-45	-54	-200
C	-63	-72	-81	-300

`df1.mul(df2)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

dfR	a	b	c	d
A	10	40	90	NaN
B	160	250	360	NaN
C	490	650	810	NaN

101



Operações Aritméticas

`df1.div(df2, fill_value=1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

dfR	a	b	c	d
A	0.1	0.1	0.1	0.01
B	0.1	0.1	0.1	0.005
C	0.1	0.1	0.1	0.003

`df1.floordiv(df2, fill_value=1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

dfR	a	b	c	d
A	0	0	0	0.0
B	0	0	0	0.0
C	0	0	0	0.0

`df1.mod(df2, fill_value=0)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

df2	a	b	c	d
A	10	20	30	100
B	40	50	60	200
C	70	80	90	300

dfR	a	b	c	d
A	1	2	3	0.0
B	4	5	6	0.0
C	7	8	9	0.0

102

Métodos Aritméticos DF e Series

`df1.add(s1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

 $+$

s1	
a	10
b	20
c	30
d	40

 $=$

dfR	a	b	c	c
A	11	22	33	NaN
B	14	25	36	NaN
C	17	28	39	NaN

`df1.sub(s1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

 $-$

s1	
a	10
b	20
c	30
d	40

 $=$

dfR	a	b	c	c
A	-9	18	-27	NaN
B	-6	-15	-24	NaN
C	-3	-12	-21	NaN

`df1.mul(s1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

 $*$

s1	
a	10
b	20
c	30
d	40

 $=$

dfR	a	b	c	c
A	10	40	90	NaN
B	40	100	180	NaN
C	70	160	270	NaN

103

Métodos Aritméticos DF e Series

`df1.div(s1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

 $/$

s1	
a	10
b	20
c	30
d	40

 $=$

dfR	a	b	c	c
A	0.1	0.10	0.1	NaN
B	0.4	0.25	0.2	NaN
C	0.7	0.40	0.3	NaN

`df1.floordiv(s1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

 $//$

s1	
a	10
b	20
c	30
d	40

 $=$

dfR	a	b	c	c
A	0	0	0	NaN
B	0	0	0	NaN
C	0	0	0	NaN

`df1.mod(s1)`

df1	a	b	c
A	1	2	3
B	4	5	6
C	7	8	9

 $\%$

s1	
a	10
b	20
c	30
d	40

 $=$

dfR	a	b	c	c
A	1	2	3	NaN
B	4	5	6	NaN
C	7	8	9	NaN

104



Solução: Preço total do Kit em cada supermercado

- I. Exibir o preço a pagar por cada produto do kit higiene em cada um dos supermercados, considerando a quantidade determinada na cesta básica

Planilha ProdHigieneMerc

Produto	Desconto	KiBarato	Pop	Mercado	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiénico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Crema Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Planilha Produtos

Produto	Categoria	Unidade	Quantidade
Sabonete	Higiene	90 g	3
Papel Higiénico	Higiene	Pct 4 unidades	2
Escova Dental	Higiene	Unidade	2
Crema Dental	Higiene	90 g	1
Protetor Solar FPS >= 30	Higiene	100 ml	1
Repelente de insetos	Higiene	300 ml	1
Detergente Líquido	Limpeza	500ml	1
Lã de Aço	Limpeza	Pct 4 unidades	1
Sabão em Pó	Limpeza	500 g	1
Desinfetante	Limpeza	1000 ml	1

Produto	Qt	Desconto	KiBarato	Pop	Mercado	SuperPrice	Tot Desconto	Tot KiBarato	Tot Pop	Tot Mercado	Tot SuperPrice
Sabonete	3	3.39	2.48	1.97	2.09	2.6	10.17	7.44	5.91	6.27	7.8
Papel Higiénico	2	6.75	8.36	7.92	9.43	7.57	13.5	16.72	15.84	18.86	15.14
Escova Dental	2	1.69	3.58	1.8	1.67	3.88	3.38	7.16	3.6	3.34	7.76
Crema Dental	1	2.69	2.8	2.37	3.35	2.86	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	1	16.21	28.23	17.28	27.8	24.37	16.21	28.23	17.28	27.8	24.37
Repelente	1	9.24	8.02	11.76	10.81	12.33	9.24	8.02	11.76	10.81	12.33

105



Desenvolvendo a Solução

- i. Selecionar a coluna quantidade dos produtos de Higiene

Planilha Produtos

Produto	Categoria	Unidade	Quantidade
Sabonete	Higiene	90 g	3
Papel Higiénico	Higiene	Pct 4 unidades	2
Escova Dental	Higiene	Unidade	2
Crema Dental	Higiene	90 g	1
Protetor Solar FPS >= 30	Higiene	100 ml	1
Repelente de insetos	Higiene	300 ml	1
Detergente Líquido	Limpeza	500 ml	1
Lã de Aço	Limpeza	Pct 4 unidades	1
Sabão em Pó	Limpeza	500 g	1
Desinfetante	Limpeza	1000 ml	1

Series Qt

Produto	Qt
Sabonete	3
Papel Higiénico	2
Escova Dental	2
Crema Dental	1
Protetor FPS >= 30	1
Repelente	1

- a) Filtrar linhas da Categoria == 'Higiene'
- b) Selecionar a coluna quantidade (gera uma Series)
- c) Renomear a Series de 'Quantidade' para 'Qt'

106

Desenvolvendo a Solução

'''

i. Selecionar a coluna quantidade dos produtos de Higiene

```

a) Filtrar linhas da Categoria == 'Higiene'
b) Selecionar a coluna quantidade (gera uma Series)
c) Renomear a Series de 'Quantidade' para 'Qt'
'''

```

```
def montaSeriesQtProd():
```

```
    dfPr=pd.read_excel("PrecosProdutosSuperMercados.xlsx",
                      sheetname='Produtos',decimal=',',index_col=0)
```

```
    sPrHig=dfPr.query('Categoria=="Higiene")')['Quantidade']
```

```
    sPrHig.name='Qt'
```

```
    return sPrHig
```

DF com linhas
cuja categoria é
Higiene

Coluna Quantidade
desse DF: Series

107

Desenvolvendo a Solução

ii. Multiplicar o preço individual pela quantidade

sPrHigQt

Produto	Qt
Sabonete	3
Papel Higiénico	2
Escova Dental	2
Crema Dental	1
Protetor FPS >= 30	1
Repelente	1



dfPrHigMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.35	2.48	1.97	2.05	2.6
Papel Higiénico	6.75	8.34	7.92	9.45	7.57
Escova Dental	1.65	3.58	1.8	1.67	3.88
Crema Dental	2.65	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33



dfTot

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	10.17	7.44	5.91	6.27	7.8
Papel Higiénico	13.5	16.72	15.84	18.86	15.14
Escova Dental	3.38	7.16	3.6	3.34	7.76
Crema Dental	2.65	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Produto	Descontão Tot	KiBarato Tot	Pop Tot	Mercadão Tot	SuperPrice Tot
Sabonete	10.17	7.44	5.91	6.27	7.8
Papel Higiénico	13.5	16.72	15.84	18.86	15.14
Escova Dental	3.38	7.16	3.6	3.34	7.76
Crema Dental	2.65	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

- Multiplicar Series por DF cujo alinhamento é pelo index → DF
- Renomear as colunas do DF resultante: *associar nome velho à nome novo → dicionário*

108

Desenvolvendo a Solução

'''

ii. Multiplicar o preço individual pela quantidade

'''

a) Multiplicar Series por DF cujo alinhamento é pelo index → DF

b) Renomear as colunas do DF resultante: *associar nome velho à nome novo* → dicionário

```
def renomearColunasDF(df):
    l=list(dfTot.columns)
    dNomes=dict()
    for el in l:
        dNomes[el]=el+ " Tot"
    dfTot.rename(columns=dNomes,inplace=True)
    return

dfTot=dfPrHigMerc.mul(sPrHig,axis='index')
renomearColunasDF(dfTot)
```

109

Desenvolvendo a Solução

iii. Combinar DataFrames e Series

sPrHigQt

Produto	Qt
Sabonete	3
Papel Higiénico	2
Escova Dental	2
Creme Dental	1
Protetor FPS >= 30	1
Repelente	1

dfPrHigMerc

Produto	Descontão	KiBarato	Pop	Mercadão	SuperPrice
Sabonete	3.39	2.48	1.97	2.09	2.6
Papel Higiénico	6.75	8.36	7.92	9.43	7.57
Escova Dental	1.69	3.58	1.8	1.67	3.88
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

dfTot

Produto	Descontão Tot	KiBarato Tot	Pop Tot	Mercadão Tot	SuperPrice Tot
Sabonete	10.17	7.44	5.91	6.27	7.8
Papel Higiénico	13.5	16.72	15.84	18.86	15.14
Escova Dental	3.38	7.16	3.6	3.34	7.76
Creme Dental	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	16.21	28.23	17.28	27.8	24.37
Repelente	9.24	8.02	11.76	10.81	12.33

Produto	Qt	Descontão	KiBarato	Pop	Mercadão	SuperPrice	Descontão Tot	KiBarato Tot	Pop Tot	Mercadão Tot	SuperPrice Tot
Sabonete	3	3.39	2.48	1.97	2.09	2.6	10.17	7.44	5.91	6.27	7.8
Papel Higiénico	2	6.75	8.36	7.92	9.43	7.57	13.5	16.72	15.84	18.86	15.14
Escova Dental	2	1.69	3.58	1.8	1.67	3.88	3.38	7.16	3.6	3.34	7.76
Creme Dental	1	2.69	2.8	2.37	3.35	2.86	2.69	2.8	2.37	3.35	2.86
Protetor FPS >= 30	1	16.21	28.23	17.28	27.8	24.37	16.21	28.23	17.28	27.8	24.37
Repelente	1	9.24	8.02	11.76	10.81	12.33	9.24	8.02	11.76	10.81	12.33

a) Concatenar com alinhamento pelo *index* (eixo linha)

110

Uma Solução: Completa

```
def montaSeriesQtProd():
    dfPr=pd.read_excel("PrecosProdutosSuperMercados.xlsx",
                      sheetname='Produtos',decimal=',',index_col=0)
    sPrHig=dfPr.query('Categoria=="Higiene")['Quantidade']
    sPrHig.name='Qt'
    return sPrHig

def renomearColunasDF(df):
    l=list(dfTot.columns)
    dNomes=dict()
    for el in l:
        dNomes[el]=el+ " Tot"
    dfTot.rename(columns=dNomes,inplace=True)
    return

dfPrHigMerc=pd.read_excel("PrecosProdutosSuperMercados.xlsx",decimal=',', index_col=0)
sPrHig = montaSeriesQtProd()
dfTot=dfPrHigMerc.mul(sPrHig,axis='index')
renomearColunasDF(dfTot)
dfUnido=pd.concat([dfPrHigMerc,sPrHig,dfTot],axis=1)
```

111

Método Úteis

Outros Métodos para União, Substituição e Atualização de DataFrame:

Sintaxe:

`df.append(df)` - Cria uma cópia com os elementos do DataFrame recebido incluídos no final, alinhados pelo columns

`df.replace(to_replace=valor, value=novo)` - Cria uma cópia, substituindo todas as ocorrências de valor por novo

`df.update(df)` - Altera atuais valores pelos valores recebidos, alinhando pelo índice

112



Método Útil: crosstab

Sintaxe: `pandas.crosstab(index, columns, margins=False, margins_name='All', dropna=True)`

Relaciona duas ou mais sequência de valores. Retorna um **DataFrame** com a tabela de frequência dos valores das sequências, a menos que uma função de agregação e um array de valores sejam especificados.

index, columns - array, Series ou lista de arrays/Series que definem os valores a agrupar nas linhas/colunas

margins, margins_name - quando margins = True é adicionado uma linha/coluna com subtotais e margins_name define o nome da linha/coluna que armazenará os totais.

dfF:

```

0  O C M
1  O L A
2  A P A
3  A R A
4  A L V
5  B C V
6  O L V
7  A C C
8  AB L C
9  B L C
10 O R M
11 AB C C

```

```
pd.crosstab(index=dfF['TpSg'],
            columns=dfF['Cab'],
            margins=True)
```

```

Cab C L P R All
Tipo
A    1 1 1 1 4
AB   1 1 0 0 2
B    1 1 0 0 2
O    1 2 0 1 4
All  4 5 1 2 12

```

```
pd.crosstab(index=dfF['Tipo'],
            columns=[dfF['Cab'], dfF['Olh']])
```

```

Cab C L P R
Olh C M V A C V A A M
Tipo
A    1 0 0 0 0 1 1 1 0
AB   1 0 0 0 1 0 0 0 0
B    0 0 1 0 1 0 0 0 0
O    0 1 0 1 0 1 0 0 1

```

```
pd.crosstab(index=dfF['Tipo'],
            columns=[dfF['Cab'], dfF['Olh']],
            margins=True)
```

```

Cab C L P R All
Olh C M V A C V A A M
Tipo
A    1 0 0 0 0 1 1 1 0 4
AB   1 0 0 0 1 0 0 0 0 2
B    0 0 1 0 1 0 0 0 0 2
O    0 1 0 1 0 1 0 0 1 4
All  2 1 1 1 2 2 1 1 1 12

```



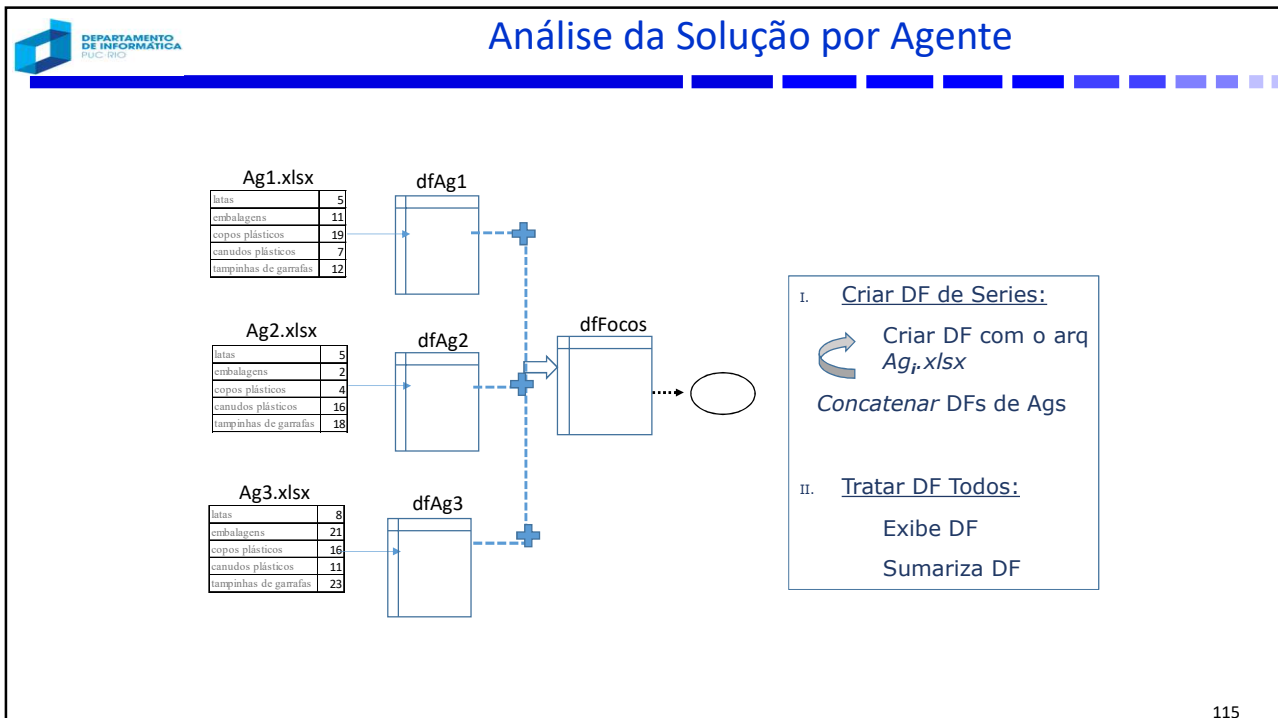
Mãos na Massa

Diariamente, 3 agentes da Vigilância Sanitária visitam uma região para detectar focos de larvas do mosquito *Aedes aegypti* nos seguintes locais:

latas, embalagens, copos e canudos plásticos, garrafas, tampinhas de garrafas, vasos de plantas, jarros de flores, bromélias, caixas d'água, tambores, latões, cisternas, calhas, piscinas, vasos sanitários, pneus velhos, sacos plásticos, lixeiras, bueiros, ralos, lonas e lajes.

A quantidade de focos encontrados por localização são registradas em arquivos Excel denominados Ag_n onde *n* varia de 1 a 3. Caso não seja encontrado focos em alguma localização, ela não consta no arquivo.

Construa um script que mostre para cada agente, a quantidade de focos encontrada por tipo de localização, a localização com maior incidência, quantidade total e a visualização gráfica das quantidades por localização.



DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Uma Solução: Focos por Agente

```
import pandas as pd

def geraNome(n):
    return "AG"+str(n+1)

def montaDataFrame(qt):
    #Cria o nome do arquivo do agente
    ldfs=[]
    for i in range(0,qt):
        #Monta o nome do arquivo do agente
        arq=geraNome(i)+".xlsx"
        #Cria um df a partir do arquivo, renomeando a coluna
        df= pd.read_excel(arq,header=None,index_col=0)
        df.rename(columns={1:'Ag'+str(i+1)},inplace=True)
        #Adiciona o DF Transposto à lista
        ldfs.append(df.T)
    dfFocos=pd.concat(ldfs)
    #Como há focos inexistentes em alguns agentes, substitui NaN por 0
    dfFocos.fillna(value=0,inplace=True)
    return dfFocos
```

116

Uma Solução: Focos por Agente

```
def trataAg(df):
    #Sumarização por agente
    total=df.sum(axis=1)
    maior=df.max(axis=1)
    local=df.idxmax(axis=1)
    df['Total de Focos']=total
    df['Maior Incidência']=maior
    df['Local Maior Incidência']=local
    return

#Cria o DF com dados dos agentes
dfFocos= montaDataFrame(3)
print(dfFocos)
#Sumarização por agente
trataAg(dfFocos):
```

117

Análise da Solução: Totais Gerais

Como saber:

	Ag1	Ag2	Ag3
latas	125	13	1
embalagens	11	8	2
copos plásticos	4	18	22
canudos plásticos	10	23	15
...
lajes	21	6	3

- I. O total de focos encontrados na região por localização?
somar as Linhas
- II. Qual a localização com mais focos na região?
Encontrar o maior valor e o local da soma das linhas
- III. A distribuição gráfica dos focos por localização na região?
Construir um gráfico de barras com o total de focos encontrados por localização

118



Uma Solução

```
def resumoTotais(df):
    total=df.sum(axis=1)
    maior=total.max()
    local=total.idxmax()
    dfTot=pd.DataFrame(total, columns=['Total'])
    dfTot.index.name='Local'
    print(dfTot)
    print('\n\nLocal de Maior Localização: {} com {:.2f} focos'.format(local,maior))

    return

#Cria o DF com dados dos agentes
dfFocos= montaDataFrame(3)
print(dfFocos)
#Sumarização por agente
trataAg(dfFocos):
#Sumarização geral
resumoTotais(dfFocos):
print(dfFocos)
```

119



Mãos na Massa

A planilha NotasAlunosFinal.xlsx armazena as notas dos alunos de uma turma no final do semestre.

Matr	Nome	P1	T1	P2	T2	P3	T3
253	Cascão	1,1	0,7	6,3	8,8	2,1	9,2
232	Cebolinha	7,9	8,0	3,6	2,2	4,2	5,7
216	Huguinho	7,9	5,9	0,7	6,1	9,7	0,5
233	Lalá	6,7	7,4	6,1	1,9	0,0	8,2
234	Lelé	3,7	2,6	0,5	5,5	6,5	2,2
235	Lili	1,0	2,9	7,1	5,9	8,7	7,7
230	Luisinho	5,8	4,3	3,7	2,2	9,4	2,7
251	Magali	5,6	6,2	2,3	7,1	7,9	7,4
252	Mônica	0,8	7,5	7,3	8,5	5,0	9,6
212	Narizinho	3,4	3,8	2,3	8,4	0,2	8,0
211	Pedrinho	1,4	8,2	5,9	0,2	6,9	6,7
215	Zezinho	1,0	2,4	9,6	6,9	2,9	5,6

- Calcule os graus G1,G2, G3. $G_i = P_i * 0.8 + T_i * 0.2$
- Calcule a média aritmética de cada aluno
- Calcule e mostre a maior nota, a menor nota, a nota média, a nota mediana de cada avaliação e de cada grau
- Mostre os alunos cuja média ≥ 5
- Considerando que a média para aprovação é 5 e nenhum Grau pode ser inferior a 3, mostre os dados dos alunos já aprovados
- Calcule e mostre a maior nota, a menor nota a nota média e a nota mediana da turma

120

Uma Solução

```
dfNotas=pd.read_excel('NotasAlunosFinal.xlsx')
print(dfNotas)
dfNotas['G1']=dfNotas['P1']*0.8 + dfNotas['T1']*0.2
dfNotas['G2']=dfNotas['P2']*0.8 + dfNotas['T2']*0.2
dfNotas['G3']=dfNotas['P3']*0.8 + dfNotas['T3']*0.2

dfNotas['Media']=dfNotas[['G1','G2','G3']].mean(axis=1)
lColunas = list(dfNotas.columns[2:]) #elimina a matr e o nome
dfNotas.loc['Média']=dfNotas[lColunas].mean()
dfNotas.loc['Mediana']=dfNotas[lColunas].median()
dfNotas.loc['Mínima']=dfNotas[lColunas].min()
dfNotas.loc['Máxima']=dfNotas[lColunas].max()
dfNotas.fillna(' ',inplace=True) # " " nos resumos de matr e nome
print(dfNotas)
print('\n Média>=5\n',dfNotas.iloc[:-4].query('Media>=5'))
print('\n Aprovados\n',dfNotas.iloc[:-4].query('G1>=3 and G2>=3 and G3>= 3 and Media>=5'))
```

121

Resultado

	Matr	Nome	P1	T1	P2	T2	P3	T3	G1	G2	G3	Media
0	253	Cascão	1,10	0,70	6,30	8,80	2,10	9,20	1,02	6,80	3,52	3,78
1	232	Cebolinha	7,90	8,00	3,60	2,20	4,20	5,70	7,92	3,32	4,50	5,25
2	216	Huguinho	7,90	5,90	0,70	6,10	9,70	0,50	7,50	1,78	7,86	5,71
3	233	Lalá	6,70	7,40	6,10	1,90	0,00	8,20	6,84	5,26	1,64	4,58
4	234	Lele	3,70	2,60	0,50	5,50	6,50	2,20	3,48	1,50	5,64	3,54
5	235	Lili	1,00	2,90	7,10	5,90	8,70	7,70	1,38	6,86	8,50	5,58
6	230	Luisinho	5,80	4,30	3,70	2,20	9,40	2,70	5,50	3,40	8,06	5,65
7	251	Magali	5,60	6,20	2,30	7,10	7,90	7,40	5,72	3,26	7,80	5,59
8	252	Mônica	0,80	7,50	7,30	8,50	5,00	9,60	2,14	7,54	5,92	5,20
9	212	Narizinho	3,40	3,80	2,30	8,40	0,20	8,00	3,48	3,52	1,76	2,92
10	211	Pedrinho	1,40	8,20	5,90	0,20	6,90	6,70	2,76	4,76	6,86	4,79
11	215	Zezinho	1,00	2,40	9,60	6,90	2,90	5,60	1,28	9,06	3,44	4,59
		Média	3,86	4,99	4,62	5,31	5,29	6,13	4,09	4,76	5,46	4,77
		Mediana	3,70	4,99	4,62	5,90	5,29	6,70	3,48	4,76	5,64	4,79
		Mínima	0,80	0,70	0,50	0,20	0,00	0,50	1,02	1,50	1,64	2,92
		Máxima	7,90	8,20	9,60	8,80	9,70	9,60	7,92	9,06	8,50	5,71

Quantos
alunos tem
Teste>7 e
Prova<4

Qual a média
dos alunos
com G3<3 e
dos com
G3>=3

Média > 5												
	Matr	Nome	P1	T1	P2	T2	P3	T3	G1	G2	G3	Media
1	232	Cebolinha	7,90	8,00	3,60	2,20	4,20	5,70	7,92	3,32	4,50	5,25
2	216	Huguinho	7,90	5,90	0,70	6,10	9,70	0,50	7,50	1,78	7,86	5,71
5	235	Lili	1,00	2,90	7,10	5,90	8,70	7,70	1,38	6,86	8,50	5,58
6	230	Luisinho	5,80	4,30	3,70	2,20	9,40	2,70	5,50	3,40	8,06	5,65
7	251	Magali	5,60	6,20	2,30	7,10	7,90	7,40	5,72	3,26	7,80	5,59
8	252	Mônica	0,80	7,50	7,30	8,50	5,00	9,60	2,14	7,54	5,92	5,20

Aprovados												
	Matr	Nome	P1	T1	P2	T2	P3	T3	G1	G2	G3	Media
1	232	Cebolinha	7,90	8,00	3,60	2,20	4,20	5,70	7,92	3,32	4,50	5,25
6	230	Luisinho	5,80	4,30	3,70	2,20	9,40	2,70	5,50	3,40	8,06	5,65
7	251	Magali	5,60	6,20	2,30	7,10	7,90	7,40	5,72	3,26	7,80	5,59

Qual a
nota
necessária
para ser
aprovado?

122

Agrupamento e Agregação

123

Problema: Agrupamentos e Agregação

- I. Deseja-se saber o impacto na G1 caso haja apenas 3 graus como notas dos testes T1 (0- Insuficiente - não fez ou está abaixo de 5 ; 5- parcialmente correto, notas entre 5 e 8.9 e 10 – correto para notas ≥ 9). Portanto, após modificar os valores do T1 de acordo com o critério, mostrar a média de G1, G2, G3 e Média de cada grupo em ambos os casos.
- II. Quantos alunos tem Testes > 7 e Provas < 4 ? Qual a média final deste grupo de alunos?
- III. Quantos e qual a média dos alunos com $G3 < 3$ e dos alunos com $G3 \geq 3$? Qual a relação percentual entre eles?
- IV. Qual a nota necessária na G4 para que os alunos ainda não aprovados sejam aprovados?
- V. Quantos alunos com média final nas faixas E- 0 a 2.9, D-3.0 a 4.9, C- 5.0 a 6.9, B – 7.0 a 8.9, A- 9.0 a 10.0?

124

Agrupamentos em DataFrames

O método `groupby` agrupa os elementos do DataFrame de acordo com um critério definido e retorna um objeto `GroupBy`.

A divisão pode ser por qualquer um de seus eixos. (padrão: 0)

Formas básicas para criar um objeto `GroupBy` a partir de um DataFrame:

`grupo1 = df.groupby(chave)`

	c1	c2	c3	c4
11				
12				
13				
14				
15				

	c1	c2	c3	c4
11				
13				
15				

	c1	c2	c3	c4
12				
14				

`grupo2 = df.groupby(chave,axis=1)`

	c1	c2	c3	c4
11				
12				
13				
14				
15				

	c1	c4
11		
12		
13		
14		
15		

	c2	c3
11		
12		
13		
14		
15		

onde, a **chave** é o critério a ser usado para agrupar e, entre outros, pode ser :

- uma função Python
- o nome de uma coluna a ser usada
- o nível do índice a ser usado
- uma lista de qualquer um dos itens acima

125

Agrupamento (axis=0)

```
df=pd.read_excel("exemploagrup.xlsx",
                 header=None)
df.columns=['Tipo','Resp','Men','Mai','Men','Mai']
df.index=[0,0,0,1,1,1,0,1]
gTipo=df.groupby('Tipo')
for (t,g) in gTipo:
    print("\nTipo:",t)
    print(g)
```

	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	B	S2	4	7	4	8
0	A	S2	0	6	0	5
1	B	S2	4	9	0	5
1	A	S1	2	7	1	8
1	B	S1	0	6	1	8
0	A	S1	2	10	2	6
1	B	S2	4	7	0	6

	Tipo	Resp	Men	Mai	Men	Mai
0	A					
0	B					
0	A					
1	B					
1	A					
1	B					
0	A					
1	B					

```
Tipo: A
Tipo Resp Men Mai Men Mai
0 A S1 1 7 4 10
0 A S2 0 6 0 5
1 A S1 2 7 1 8
0 A S1 2 10 2 6

Tipo: B
Tipo Resp Men Mai Men Mai
0 B S2 4 7 4 8
1 B S2 4 9 0 5
1 B S1 0 6 1 8
1 B S2 4 7 0 6
```

126



Agrupamento (axis=0)

```
df=pd.read_excel("exemploagrup.xlsx",
                 header=None)
df.columns=['Tipo','Resp','Men','Mai','Men','Mai']
df.index=[0,0,0,1,1,1,0,1]
gTipo=df.groupby(level=0) #agrupa pelo índice
for (t,g) in gTipo:
    print("\nLevel:",t)
    print(g)
```

	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	B	S2	4	7	4	8
0	A	S2	0	6	0	5
1	B	S2	4	9	0	5
1	A	S1	2	7	1	8
1	B	S1	0	6	1	8
0	A	S1	2	10	2	6
1	B	S2	4	7	0	6

	Tipo	Resp	Men	Mai	Men	Mai
0						
0						
0						
1						
1						
1						
0						
1						

```
Level: 0
Tipo Resp Men Mai Men Mai
0 A S1 1 7 4 10
0 B S2 4 7 4 8
0 A S2 0 6 0 5
0 A S1 2 10 2 6

Level: 1
Tipo Resp Men Mai Men Mai
1 B S2 4 9 0 5
1 A S1 2 7 1 8
1 B S1 0 6 1 8
1 B S2 4 7 0 6
```

127



Agrupamento (axis=1)_{1/2}

```
def getMedida(nome):
    if nome == 'Men':
        return 'Minima'
    elif nome == 'Mai':
        return 'Máxima'
    else:
        return 'Outros'

df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men','Mai','Men','Mai']
df.index=[0,0,0,1,1,1,0,1]
gVal=df.groupby(getMedida,axis=1)
for (t,g) in gVal:
    print("\nMedida:",t)
    print(g)
```

	Tipo	Resp	Men	Mai	Men	Mai
0						
0						
0						
1						
1						
1						
0						
1						

```
Medida: Minima
Men Men
0 1 4
0 4 4
0 0 0
1 4 0
1 2 1
1 0 1
0 2 2
1 4 0

Medida: Máxima
Mai Mai
0 7 10
0 7 8
0 6 5
1 9 5
1 7 8
1 6 8
0 10 6
1 7 6

Medida: Outros
Tipo Resp
0 A S1
0 B S2
0 A S2
1 B S2
1 A S1
1 B S1
0 A S1
1 B S2
```

128



Agrupamento (axis=1)

```
df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men','Mai','Men','Mai']
df.index=[0,0,0,1,1,1,0,1]
gVall=df.groupby(level=0,axis=1)
for (t,g) in gVall:
    print("\nMedida:",t)
    print(g)
```

	Tipo	Resp	Men	Mai	Men	Mai
0						
0						
0						
1						
1						
1						
0						
1						


```
Medida: Mai
Men
0 7 10
0 7 8
0 6 5
1 9 5
1 7 8
1 6 8
0 10 6
1 7 6

Medida: Men
Men
0 1 4
0 4 4
0 0 0
1 4 0
1 2 1
1 0 1
0 2 2
1 4 0

Medida: Resp
Resp
0 S1
0 S2
1 S2
1 S1
1 S1
0 S1
1 S2

Medida: Tipo
Tipo
0 A
0 B
0 A
1 B
1 B
0 A
1 B
```

129



Agrupando por mais de uma chave simultaneamente

Sintaxe:

```
df.groupby([lista de chaves], as_index = True)
```

Retorna um novo objeto *Groupby* com os grupos determinados pela lista de chaves hierarquicamente organizados (*multi-index*). Quando *as_index=False*, mantém o índice original.

as_index = True/False - hierarquiza (padrão) ou não os índices dos grupo de acordo com o critério

```
df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men','Mai','Men','Mai']
df.index=[0,0,0,1,1,1,0,1]
gLevelTipo=df.groupby(['Tipo','Resp'])
for (t,g) in gLevelTipo:
    print("\nGrupo:",t)
    print(g)
```

	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	B	S2	4	7	4	8
0	A	S2	0	6	0	5
1	B	S2	4	9	0	5
1	A	S1	2	7	1	8
1	B	S1	0	6	1	8
0	A	S1	2	10	2	6
1	B	S2	4	7	0	6


```
MGrupo: ('A', 'S1')
Tipo Resp Men Mai Men Mai
0 A S1 1 7 4 10
1 A S1 2 7 1 8
0 A S1 2 10 2 6

MGrupo: ('A', 'S2')
Tipo Resp Men Mai Men Mai
0 A S2 0 6 0 5

MGrupo: ('B', 'S1')
Tipo Resp Men Mai Men Mai
1 B S1 0 6 1 8

MGrupo: ('B', 'S2')
Tipo Resp Men Mai Men Mai
0 B S2 4 7 4 8
1 B S2 4 9 0 5
1 B S2 4 7 0 6
```

130



Problema: Agrupamentos

- I. Deseja-se saber o impacto na G1 caso haja apenas 3 graus como notas dos testes T1 (0 - Insuficiente - não fez ou está abaixo de 5; 5- parcialmente correto, notas entre 5 e 8.9 e 10 – correto para notas ≥ 9). Portanto, após modificar os valores do T1 de acordo com o critério, mostrar a média de G1, G2, G3 e Média de cada grupo em ambos os casos.
 - i. Critério de agrupamento: nota do Teste1
 - ii. Número de grupos gerados: 3 grupos (0,5 10)
 - iii. O que fazer com cada grupo: calcular a média

DESAFIO: Analisar o impacto nos grupos: T1 no G1, T2 no G2 e T3 no G3

131



Solução 1

```
def alteraGrau(x):
    if x<5:
        return 0
    elif x>=10:
        return 10
    else:
        return 5
def defGrupo(s):
    if s['T1']<5:
        return 0
    elif s['T1']>=10:
        return 10
    else:
        return 5
def calculaG(df):
    df['G1']=df['P1']*0.8 + df['T1']*0.2
    df['G2']=df['P2']*0.8 + df['T2']*0.2
    df['G3']=df['P3']*0.8 + df['T3']*0.2
    df['Media']=df[['G1','G2','G3']].mean(axis=1)
    return
```

132

Solução 1

```
dfNotas=pd.read_excel('NotasAlunosFinal.xlsx')
dfNotasM=dfNotas.copy()

dfNotasM['T1']=dfNotas['T1'].apply(alteraGrau)
calculaG(dfNotasM)
calculaG(dfNotas)

gTesteModif=dfNotasM.groupby('T1')
gTesteNormal=dfNotas.groupby(dfNotas.apply(defGrupo,axis=1))

lColunas=['G1','G2','G3','Media']
dfResModif= pd.DataFrame(gTesteModif[lColunas].mean())
dfResNormal= pd.DataFrame(gTesteNormal[lColunas].mean())
dfFinal=pd.concat([dfResModif,dfResNormal],axis=1)
dfFinal.columns=['G1 Modif','G2 Modif','G3 Modif','Media Modif','G1 Normal','G2 Normal','G3 Normal','Media Normal']
print(dfFinal)
```

133

Problema: Agrupamentos

- II. Quantos alunos tem Testes>7 e Provas<4? Qual a média final deste grupo de alunos?
 - I. Critério de agrupamento: Testes>7 e Provas<4
 - II. Número de grupos gerados: 2 grupos: os que satisfazem o critério e os que não satisfazem
 - III. O que fazer com os que satisfazem o critério: contar e calcular a média final
- III. Quantos e qual a média dos alunos com $G3 < 3$ e dos alunos com $G3 \geq 3$? Qual a relação percentual entre eles?
- IV. Qual a nota necessária na G4 para que os alunos ainda não aprovados sejam aprovados?
- V. Quantos alunos com média final nas faixas E- 0 a 2.9, D-3.0 a 4.9, C- 5.0 a 6.9, B- 7.0 a 8.9, A- 9.0 a 10.0?

134

Uma Solução

```
import pandas as pd

def defGrupo(s):
    if (s['T1']>7 and s['P1']<4) or (s['T2']>7 and s['P2']<4) or (s['T3']>7 and s['P3']<4):
        return 'Teste>7 e Prova<4'
    else:
        return 'Teste<=7 ou Prova>=4'

def calculaG(df):
    df['G1']=df['P1']*0.8 + df['T1']*0.2
    df['G2']=df['P2']*0.8 + df['T2']*0.2
    df['G3']=df['P3']*0.8 + df['T3']*0.2
    df['Media']=df[['G1','G2','G3']].mean(axis=1)
    return

dfNotas=pd.read_excel('NotasAlunosFinal.xlsx')
calculaG(dfNotas)

g=dfNotas.groupby(dfNotas.apply(defGrupo,axis=1))
g=g[['G1','G2','G3','Media']].agg(['mean','median'])
print(g)
```

135

Resultado

	G1		G2		G3		Media	
	mean	median	mean	median	mean	median	mean	median
Teste<= 7 ou Prova>=4	4,51	4,49	4,32	3,36	6,333333	6,75	5,054444	5,413333
Teste>7 e Prova < 4	3,66	3,12	5,19	5,01	4,583333	4,72	4,477778	4,686667

Se apenas os dados do grupo Teste>7 e Prova < 4 fosse desejado:

```
g=g.get_group('Teste>7 e Prova<4')
g=g[['G1','G2','G3','Media']].agg(['mean','median'])
#qt=df.count()
print(g)
```

```

      G1  G2  G3  Media
mean  3.66  5.19  4.583333  4.477778
median 3.12  5.01  4.720000  4.686667
```

136

Agrupando DataFrames

Sintaxe: `GroupBy.groups`
`GroupBy.indices`

`GroupBy.groups` dict {nome do grupo -> labels dos índices do grupo}
`GroupBy.indices` dict {nome do grupo -> array com a posição dos índices do grupo}

Sintaxe: `GroupBy.get_group(nome)`

Constrói um DataFrame com os elementos do grupo cujo nome foi fornecido

Sintaxe: `GroupBy.size()`

Retorna a quantidade de elementos de cada grupo

137

Agregação

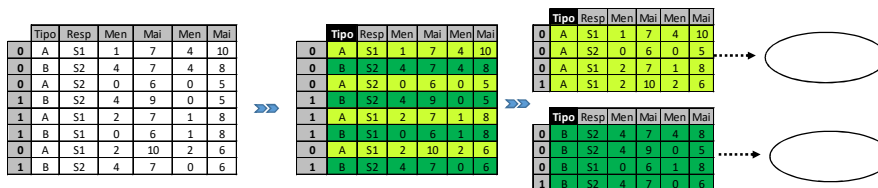
138

Funções de Agregação

Sintaxe: `GroupBy.agg(função pré-definida ou nome de método)`
`GroupBy.agg(lista de funções/nome de método)`

As funções fornecidas para agregação reduzem a dimensão do objeto fornecido. São aplicadas sobre os valores do grupo e retornam um resultado para o conjunto.

As mais comuns são *mean*, *sum*, *size*, *count*, *std*, *var*, *describe*, *first*, *last*, *nth*, *min*, *max*.



	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	B	S2	4	7	4	8
0	A	S2	0	6	0	5
1	B	S2	4	9	0	5
1	A	S1	2	7	1	8
1	B	S1	0	6	1	8
0	A	S1	2	10	2	6
1	B	S2	4	7	0	6

	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	B	S2	4	7	4	8
0	A	S2	0	6	0	5
1	B	S2	4	9	0	5
1	A	S1	2	7	1	8
1	B	S1	0	6	1	8
0	A	S1	2	10	2	6
1	B	S2	4	7	0	6

	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	A	S2	0	6	0	5
0	A	S1	2	7	1	8
1	A	S1	2	10	2	6
1	B	S1	0	6	1	8
1	B	S2	4	7	0	6

Os nomes dos grupos tornam-se o novo índice ao longo do eixo do agrupamento.

No caso de várias chaves, o resultado é um *multi_index* por padrão, mas pode ser alterado usando *as_index=False*

139

Forma Básica

```
df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men1','Mai1','Men2','Mai2']
df.index=[0,0,0,1,1,1,0,1]
gTipoA=df.groupby('Tipo')
dfResAggTipo = gTipoA.agg(['max','min','sum','mean','median','count'])
print(dfResAggTipo)
```

gTipoA

Tipo: A						
	Tipo	Resp	Men	Mai	Men	Mai
0	A	S1	1	7	4	10
0	A	S2	0	6	0	5
1	A	S1	2	7	1	8
0	A	S1	2	10	2	6

Tipo: B						
	Tipo	Resp	Men	Mai	Men	Mai
0	B	S2	4	7	4	8
1	B	S2	4	9	0	5
1	B	S1	0	6	1	8
1	B	S2	4	7	0	6

Men1							Mai1					Men2		\	
Tipo	max	min	sum	mean	median	count	max	min	sum	mean	...	sum	mean		
A	2	0	5	1.25	1.5	4	10	6	30	7.50	...	7	1.75		
B	4	0	12	3.00	4.0	4	9	6	29	7.25	...	5	1.25		
Mai2															
Tipo	median	count	max	min	sum	mean	median	count							
A	1.5	4	10	5	29	7.25	7	4							
B	0.5	4	8	5	27	6.75	7	4							
[2 rows x 24 columns]															

140

Forma Básica renomeando as colunas

```
df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men1','Mai1','Men2','Mai2']
df.index=[0,0,0,1,1,1,0,1]
gTipoA=df.groupby('Tipo')
dfResAggTipo = gTipoA.agg(['max','min','sum','mean','median','count'])
dfResAggTipo.rename(columns={'sum':'Soma','mean':'Média','max':'Maior',
                             'min':'Menor','median':'Mediana','count':'Qt'},
                    inplace=True)
print(dfResAggTipo)
```

Tipo	Men1						Mai1						Men2					
	Maior	Menor	Soma	Média	Mediana	Qt	Maior	Menor	Soma	Média	...	Soma	Média	...	Soma	Média	...	Soma
A	2	0	5	1.25	1.5	4	10	6	30	7.50	...	7	1.75	...	5	1.25	...	5
B	4	0	12	3.00	4.0	4	9	6	29	7.25	...	5	1.25	...	5	1.25	...	5

Tipo	Mai2						Men2					
	Maior	Menor	Soma	Média	Mediana	Qt	Maior	Menor	Soma	Média	Mediana	Qt
A	1.5	4	10	5	29	7.25	7	4
B	0.5	4	8	5	27	6.75	7	4

[2 rows x 24 columns]

141

Forma Básica com funções por Colunas

```
df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men1','Mai1','Men2','Mai2']
df.index=[0,0,0,1,1,1,0,1]
gTipoA=df.groupby('Tipo')
dfResAggTipo = gTipoA.agg({'Men1':['min','sum'],
                             'Men2':['min','count'],
                             'Mai1':['max','mean'],
                             'Mai2':['max','median']})
print(dfResAggTipo)
```

gTipoA

Tipo: A						
Tipo	Resp	Men	Mai	Men	Mai	
0	A	S1	1	7	4	10
0	A	S2	0	6	0	5
1	A	S1	2	7	1	8
0	A	S1	2	10	2	6

Tipo: B						
Tipo	Resp	Men	Mai	Men	Mai	
0	B	S2	4	7	4	8
1	B	S2	4	9	0	5
1	B	S1	0	6	1	8
1	B	S2	4	7	0	6

Tipo	Men1		Men2		Mai1		Mai2	
	min	sum	min	count	max	mean	max	mediano
A	0	5	0	4	10	7.50	10	7
B	0	12	0	4	9	7.25	8	7

Como
renomear as
colunas?

142



Forma Básica com funções por colunas renomeadas

```
df=pd.read_excel("exemploagrup.xlsx",header=None)
df.columns=['Tipo','Resp','Men1','Mai1','Men2','Mai2']
df.index=[0,0,0,1,1,1,0,1]
gTipoA=df.groupby('Tipo')
dicConv={'sum':'Soma','mean':'Média','max':'Maior',
         'min':'Menor','median':'Mediana','count':'Qt'}
dfResAggTipo = gTipoA.agg({'Men1':['min','sum'],
                           'Men2':['min','count'],
                           'Mai1':['max','mean'],
                           'Mai2':['max','median']})
dfResAggTipo = dfResAggTipo.rename(columns=dicConv)
print(dfResAggTipo)
```

Tipo	Men1		Men2		Mai1		Mai2	
	Menor	Soma	Menor	Qt	Maior	Média	Maior	Mediana
A	0	5	0	4	10	7.50	10	7
B	0	12	0	4	9	7.25	8	7