Pró-reitora de Pós-Graduação, Pesquisa e Inovação

Especialização em Ciências de Dados e Analytics

Programação na Prática

Parte 2 – Pandas

Aula 7

Agenda

- Biblioteca Pandas
  - ~~Visão geral e instalação~~
  - ~~*Series*~~
  - ~~*DataFrame*~~
  - ~~*I/O*~~
  - Multiníveis
  - Dados ausentes
  - Agrupamento
  - Concatenação, junção e *mesclagem* de dados

Pandas :: *DataFrame*

- Multiníveis
  - ***df.append***

```
dfPeD
```

	FIN	JP	EUA	CG	LX	IS	SUE	BR
2000	6731	5151	4545	4245	3773	0	0	423
2005	7544	5360	4612	5291	4864	7261	6100	587
2009	7643	5147	0	6149	4810	9117	5046	667

```
df
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	0.0	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
dfT = df.append(dfPeD)
```

```
dfT
```

	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
1990	42.4	1.0	15.0	0.0	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
2010	158.5	8.0	126.9	406.0	145.5	NaN	99.7	NaN	122.0	NaN	NaN	59.8
2000	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

Neste caso, temos apenas uma união dos DF´s anteriores. Mas os dados são diferentes semanticamente.

Pandas :: *DataFrame*

- Multiníveis
  - ***pd.MultiIndex***

```
indiceAnos = list(dfT.index)
```

```
indiceAnos
```

```
['1990', '2000', '2010', '2000', '2005', '2009']
```

```
indiceSemantico = ['Exp', 'Exp', 'Exp', 'P&D', 'P&D', 'P&D']
```

```
indiceSemantico
```

```
['Exp', 'Exp', 'Exp', 'P&D', 'P&D', 'P&D']
```

```
indiceHierarquico = pd.MultiIndex.from_tuples(indiceHierarquico)
```

```
indiceHierarquico
```

```
MultiIndex(levels=[['Exp', 'P&D'], ['1990', '2000', '2005', '2009', '2010']],
           labels=[[0, 0, 0, 1, 1, 1], [0, 1, 4, 1, 2, 3]])
```

```
dfT.set_index(indiceHierarquico)
```

		AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	**1990**	42.4	1.0	15.0	0.0	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
	2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
	2010	158.5	8.0	126.9	406.0	145.5	NaN	99.7	NaN	122.0	NaN	NaN	59.8
P&D	**2000**	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
	2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
	2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

```
eHierarquico = list(zip(indiceSemantico, indiceAnos))
```

```
eHierarquico
```

```
', '1990'),
', '2000'),
', '2010'),
', '2000'),
', '2005'),
', '2009')]
```

Pandas :: *DataFrame*

- Multiníveis
  - ***Rotulando os índices***

```
dfT.index.names = ['Indicador', 'Ano']
```

```
dfT
```

Indicador	Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	1990	42.4	1.0	15.0	0.0	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
	2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
	2010	158.5	8.0	126.9	406.0	145.5	NaN	99.7	NaN	122.0	NaN	NaN	59.8
P&D	2000	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
	2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
	2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

Pandas :: *DataFrame*

- Multiníveis
  - ***df.xs: Acessando dados do DataFrame***

Indicador	Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	1990	42.4	1.0	15.0	0.0	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
	2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
	2010	158.5	8.0	126.9	406.0	145.5	NaN	99.7	NaN	122.0	NaN	NaN	59.8
P&D	2000	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
	2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
	2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

```
dfT.xs('P&D')
```

Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
2000	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

```
dfT.xs('2000', level = 'Ano')
```

Indicador	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
P&D	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN

Exercício

- Construa um novo DataFrame, que agregue tanto informações do DataFrame do exercício anterior, quanto informações de outro indicador a sua escolha.
- O novo DataFrame deverá ter como nível mais externo os 2 indicadores (o anterior, mais o novo que você escolheu), mantendo como nível mais interno o ano

Agenda

- Biblioteca Pandas
  - ~~Visão geral e instalação~~
  - ~~*Series*~~
  - ~~*DataFrame*~~
  - ~~*Multiníveis*~~
  - Dados ausentes
  - Agrupamento
  - Concatenação, junção e *mesclagem* de dados
  - I/O

Pandas :: *DataFrame*

- Dados Ausentes
  - ***Abordagem 1:*** *Eliminando dados com NaN*

Indicador	Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	1990	42.4	1.0	15.0	0.0	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
	2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
										NaN	122.0	NaN	NaN	59.8
										0.0	5151.0	3773.0	0.0	NaN
										51.0	5360.0	4864.0	6100.0	NaN
										17.0	5147.0	4810.0	5046.0	NaN

```
dfT.dropna()
```

Indicador	Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK

```
dfT.dropna(axis=1)
```

Indicador	Ano	BR	CG	EUA	JP
Exp	1990	1.0	15.0	89.5	66.6
	2000	5.9	73.9	197.4	128.9
	2010	8.0	126.9	145.5	122.0
P&D	2000	423.0	4245.0	4545.0	5151.0
	2005	587.0	5291.0	4612.0	5360.0
	2009	667.0	6149.0	0.0	5147.0

```
dfT.dropna
```

```
Signature: dfT.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
Docstring:
Return object with labels on given axis omitted where alternately any
or all of the data are missing
```

Pandas :: *DataFrame*

- Dados Ausentes
  - ***Abordagem 2:*** *Substituindo os dados NaN*

Indicador	Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	1990	42.4	1.0	15.0	0.0	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
	2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
	2010	158.5	8.0	126.9	406.0	145.5	NaN	99.7	NaN	122.0	NaN	NaN	59.8
P&D	2000	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
	2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
	2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

```
dfT.fillna(value=0)
```

Indicador	Ano	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
Exp	1990	42.4	1.0	15.0	0.0	89.5	0.0	26.9	0.0	66.6	0.0	0.0	34.6
	2000	85.5	5.9	73.9	41.7	197.4	0.0	58.7	0.0	128.9	0.0	0.0	70.4
	2010	158.5	8.0	126.9	406.0	145.5	0.0	99.7	0.0	122.0	0.0	0.0	59.8
P&D	2000	0.0	423.0	4245.0	0.0	4545.0	6731.0	0.0	0.0	5151.0	3773.0	0.0	0.0
	2005	0.0	587.0	5291.0	0.0	4612.0	7544.0	0.0	7261.0	5360.0	4864.0	6100.0	0.0
	2009	0.0	667.0	6149.0	0.0	0.0	7643.0	0.0	9117.0	5147.0	4810.0	5046.0	0.0

Pandas :: *DataFrame*

- Dados Ausentes
  - ***Abordagem 2:*** *Substituindo os dados NaN*

```
df
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	NaN	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
df['CH'].fillna(value = df.loc['1990'].mean())
```

```
Ano
1990     39.428571
2000     41.700000
2010    406.000000
Name: CH, dtype: float64
```

```
df['CH'].fillna(value = df['CH'].mean())
```

```
Ano
1990    223.85
2000     41.70
2010    406.00
Name: CH, dtype: float64
```

Pandas :: *DataFrame*

- Dados Ausentes
  - ***Abordagem 2:*** *Substituindo os dados NaN*

```
df
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	NaN	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
df.fillna(method='bfill')
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	41.7	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
df.fillna(method='ffill', axis=1)
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	15.0	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

Exercícios

- Com o DataFrame criado no exercício anterior, crie novos DataFrames, considerando as seguintes opções:
  - Substituindo as informações ausentes de um país pela média dos valores presentes deste país
  - Substituindo as informações ausentes de um país em um determinado ano, pelo valor presente no ano anterior se estiver disponível, ou no ano subsequente se estiver disponível
    - Crie funções auxiliares para isso

Agenda

- Biblioteca Pandas
  - ~~Visão geral e instalação~~
  - ~~*Series*~~
  - ~~*DataFrame*~~
  - ~~*I/O*~~
  - ~~Dados ausentes~~
  - Agrupamento
  - Concatenação, junção e *mesclagem* de dados

Pandas :: *DataFrame*

- Agrupamento
  - ***df.groupby***

```
dados = {'Universidade' : ['UPE','UPE','UPE','UFPE','UFPE','UFPE','UNICAP','UNICAP'],
         'Curso' : ['Comp','Med','Dir','Comp','Med','Dir','Comp','Dir'],
         'Alunos': [40,80,80,200,80,80,40,80]}
```

```
dfUni = pd.DataFrame(dados)
```

```
dfUni
```

	Alunos	Curso	Universidade
0	40	Comp	UPE
1	80	Med	UPE
2	80	Dir	UPE
3	200	Comp	UFPE
4	80	Med	UFPE
5	80	Dir	UFPE
6	40	Comp	UNICAP
7	80	Dir	UNICAP

Pandas :: *DataFrame*

- Agrupamento
  - ***df.groupby***

	Alunos	Curso	Universidade
0	40	Comp	UPE
1	80	Med	UPE
2	80	Dir	UPE
3	200	Comp	UFPE
4	80	Med	UFPE
5	80	Dir	UFPE
6	40	Comp	UNICAP
7	80	Dir	UNICAP

```
grUni = dfUni.groupby('Universidade')
```

```
grUni.sum()
```

Universidade	Alunos
UFPE	360
UNICAP	120
UPE	200

Note que o *groupby* gera outro ***DataFrame***

```
grCurso = dfUni.groupby('Curso')
```

```
grCurso.mean()
```

Curso	Alunos
Comp	93.333333
Dir	80.000000
Med	80.000000

Pandas :: *DataFrame*

- Agrupamento
  - ***df.describe***

	Alunos	Curso	Universidade
0	40	Comp	UPE
1	80	Med	UPE
2	80	Dir	UPE
3	200	Comp	UFPE
4	80	Med	UFPE
5	80	Dir	UFPE
6	40	Comp	UNICAP
7	80	Dir	UNICAP

```
grUni.describe()
```

	Alunos							
	count	mean	std	min	25%	50%	75%	max
Universidade								
UFPE	3.0	120.000000	69.282032	80.0	80.0	80.0	140.0	200.0
UNICAP	2.0	60.000000	28.284271	40.0	50.0	60.0	70.0	80.0
UPE	3.0	66.666667	23.094011	40.0	60.0	80.0	80.0	80.0

```
grCurso.describe()
```

	Alunos							
	count	mean	std	min	25%	50%	75%	max
Curso								
Comp	3.0	93.333333	92.376043	40.0	40.0	40.0	120.0	200.0
Dir	3.0	80.000000	0.000000	80.0	80.0	80.0	80.0	80.0
Med	2.0	80.000000	0.000000	80.0	80.0	80.0	80.0	80.0

Agenda

- Biblioteca Pandas
  - ~~Visão geral e instalação~~
  - ~~*Series*~~
  - ~~*DataFrame*~~
  - ~~*I/O*~~
  - ~~Dados ausentes~~
  - ~~Agrupamento~~
  - Concatenação, junção e *mesclagem* de dados

Pandas :: *DataFrame*

- Concatenação (União)
  - ***pd.concat***

```
df
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	NaN	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
dfPeD
```

	FIN	JP	EUA	CG	LX	IS	SUE	BR
2000	6731	5151	4545	4245	3773	0	0	423
2005	7544	5360	4612	5291	4864	7261	6100	587
2009	7643	5147	0	6149	4810	9117	5046	667

```
pd.concat([df, dfPeD])
```

	AL	BR	CG	CH	EUA	FIN	FR	IS	JP	LX	SUE	UK
1990	42.4	1.0	15.0	NaN	89.5	NaN	26.9	NaN	66.6	NaN	NaN	34.6
2000	85.5	5.9	73.9	41.7	197.4	NaN	58.7	NaN	128.9	NaN	NaN	70.4
2010	158.5	8.0	126.9	406.0	145.5	NaN	99.7	NaN	122.0	NaN	NaN	59.8
2000	NaN	423.0	4245.0	NaN	4545.0	6731.0	NaN	0.0	5151.0	3773.0	0.0	NaN
2005	NaN	587.0	5291.0	NaN	4612.0	7544.0	NaN	7261.0	5360.0	4864.0	6100.0	NaN
2009	NaN	667.0	6149.0	NaN	0.0	7643.0	NaN	9117.0	5147.0	4810.0	5046.0	NaN

Pandas :: *DataFrame*

- Concatenação (União)
  - ***pd.concat***

```
df
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	NaN	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
dfPeD
```

	FIN	JP	EUA	CG	LX	IS	SUE	BR
2000	6731	5151	4545	4245	3773	0	0	423
2005	7544	5360	4612	5291	4864	7261	6100	587
2009	7643	5147	0	6149	4810	9117	5046	667

```
pd.concat([df, dfPeD], axis=1)
```

	EUA	JP	AL	UK	FR	CG	CH	BR	FIN	JP	EUA	CG	LX	IS	SUE	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9	6731.0	5151.0	4545.0	4245.0	3773.0	0.0	0.0	423.0
2005	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7544.0	5360.0	4612.0	5291.0	4864.0	7261.0	6100.0	587.0
2009	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7643.0	5147.0	0.0	6149.0	4810.0	9117.0	5046.0	667.0
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Pandas :: *DataFrame*

- Junção (Join)
  - ***df.join***

O join só irá funcionar se as colunas dos DataFrames juntados forem diferentes! Ou seja, temos uma chave primaria comum e as demais colunas devem ser diferentes entre si.

```
df
```

Ano	EUA	JP	AL	UK	FR	CG	CH	BR
1990	89.5	66.6	42.4	34.6	26.9	15.0	NaN	1.0
2000	197.4	128.9	85.5	70.4	58.7	73.9	41.7	5.9
2010	145.5	122.0	158.5	59.8	99.7	126.9	406.0	8.0

```
dfPeD
```

	FIN	JP	EUA	CG	LX	IS	SUE	BR
2000	6731	5151	4545	4245	3773	0	0	423
2005	7544	5360	4612	5291	4864	7261	6100	587
2009	7643	5147	0	6149	4810	9117	5046	667

```
df.join(dfPeD)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-220-3130a1a3e0f8> in <module>()
----> 1 df.join(dfPeD)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in join(self, other, on, how, lsuffix, rsuffix, sort)
   5314         # For SparseDataFrame's benefit
   5315         return self._join_compat(other, on=on, how=how, lsuffix=lsuffix,
-> 5316                                  rsuffix=rsuffix, sort=sort)
   5317
```

Pandas :: *DataFrame*

- Junção (Join)
  - ***df.join***

```
dfLeft = pd.DataFrame({'C1': list(np.random.randint(0, 10, size=3)),
                       'C2': list(np.random.randint(0, 10, size=3))},
                      index = ['I1', 'I2', 'I3'])
```

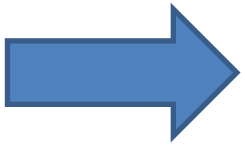
```
dfRight = pd.DataFrame({'D1': list(np.random.randint(0, 10, size=3)),
                        'D2': list(np.random.randint(0, 10, size=3))},
                       index = ['I0', 'I2', 'I3'])
```

```
dfLeft
```

	C1	C2
I1	5	1
I2	8	6
I3	2	2

```
dfRight
```

	D1	D2
I0	2	1
I2	6	8
I3	9	4



```
dfLeft.join(dfRight)
```

	C1	C2	D1	D2
I1	5	1	NaN	NaN
I2	8	6	6.0	8.0
I3	2	2	9.0	4.0

```
dfLeft.join(dfRight, how='outer')
```

	C1	C2	D1	D2
I0	NaN	NaN	2.0	1.0
I1	5.0	1.0	NaN	NaN
I2	8.0	6.0	6.0	8.0
I3	2.0	2.0	9.0	4.0

```
dfLeft.join(dfRight, how='inner')
```

	C1	C2	D1	D2
I2	8	6	6	8
I3	2	2	9	4

Pandas :: *DataFrame*

- Mesclar
  - ***pd.merge:***

Similar a um join de tables em SQL a partir de chave(s)

```
pd.merge(dfUni, dfUniFormandos, how='inner', on=['Universidade', 'Curso'])
```

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
1	80	Med	UPE	75
2	80	Dir	UPE	60
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

Outras opções são: *outer, left, right*

```
dfUniFormandos
```

	Curso	Formandos	Universidade
0	Comp	25	UPE
1	Med	75	UPE
2	Dir	60	UPE
3	Comp	180	UFPE
4	Med	78	UFPE
5	Dir	67	UFPE
6	Comp	35	UNICAP
7	Dir	75	UNICAP

```
dfUni
```

	Alunos	Curso	Universidade
0	40	Comp	UPE
1	80	Med	UPE
2	80	Dir	UPE
3	200	Comp	UFPE
4	80	Med	UFPE
5	80	Dir	UFPE
6	40	Comp	UNICAP
7	80	Dir	UNICAP

Pandas :: *DataFrame*

- Selecionando valores únicos
  - ***pd.unique***

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
1	80	Med	UPE	75
2	80	Dir	UPE	60
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

```
dfUniMerge['Curso'].unique()
```

```
array(['Comp', 'Med', 'Dir'], dtype=object)
```

```
dfUniMerge['Universidade'].unique()
```

```
array(['UPE', 'UFPE', 'UNICAP'], dtype=object)
```

```
dfUniMerge['Universidade'].nunique()
```

```
3
```

```
dfUniMerge['Universidade'].value_counts()
```

```
UFPE       3
UPE        3
UNICAP     2
Name: Universidade, dtype: int64
```

Pandas :: *DataFrame*

- Selecionando valores com operadores booleanos
  - **& |**

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
1	80	Med	UPE	75
2	80	Dir	UPE	60
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

```
dfUniMerge[(dfUniMerge['Formandos'] > dfUniMerge['Alunos'] * 0.8)]
```

	Alunos	Curso	Universidade	Formandos
1	80	Med	UPE	75
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

```
dfUniMerge[(dfUniMerge['Formandos'] > dfUniMerge['Alunos'] * 0.8)
           & (dfUniMerge['Curso'] == 'Comp')]
```

	Alunos	Curso	Universidade	Formandos
3	200	Comp	UFPE	180
6	40	Comp	UNICAP	35

Pandas :: *DataFrame*

- Executando funções no DF
  - ***df.apply***

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
1	80	Med	UPE	75
2	80	Dir	UPE	60
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

```
dfUniMerge['Curso'].apply(len)
```

```
0    4
1    3
2    3
3    4
4    3
5    3
6    4
7    3
Name: Curso, dtype: int64
```

```
taxa = 20
dfUniMerge['Formandos20'] =
  dfUniMerge['Formandos'].apply(lambda x : x * (1 + taxa/100))
```

```
dfUniMerge
```

	Alunos	Curso	Universidade	Formandos	Formandos20
0	40	Comp	UPE	25	30.0
1	80	Med	UPE	75	90.0
2	80	Dir	UPE	60	72.0
3	200	Comp	UFPE	180	216.0
4	80	Med	UFPE	78	93.6
5	80	Dir	UFPE	67	80.4
6	40	Comp	UNICAP	35	42.0
7	80	Dir	UNICAP	75	90.0

Pandas :: *DataFrame*

- Eliminando colunas
  - ***del***

	Alunos	Curso	Universidade	Formandos	Formandos20
0	40	Comp	UPE	25	30.0
1	80	Med	UPE	75	90.0
2	80	Dir	UPE	60	72.0
3	200	Comp	UFPE	180	216.0
4	80	Med	UFPE	78	93.6
5	80	Dir	UFPE	67	80.4
6	40	Comp	UNICAP	35	42.0
7	80	Dir	UNICAP	75	90.0

```
dfUniMerge[(dfUniMerge['Formandos20'] > dfUniMerge['Alunos'])]
```

	Alunos	Curso	Universidade	Formandos	Formandos20
1	80	Med	UPE	75	90.0
3	200	Comp	UFPE	180	216.0
4	80	Med	UFPE	78	93.6
5	80	Dir	UFPE	67	80.4
6	40	Comp	UNICAP	35	42.0
7	80	Dir	UNICAP	75	90.0

```
del dfUniMerge['Formandos20']
```

```
dfUniMerge
```

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
1	80	Med	UPE	75
2	80	Dir	UPE	60
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

```
dfUniMerge.columns
```

```
Index(['Alunos', 'Curso', 'Universidade', 'Formandos'], dtype='object')
```

Pandas :: *DataFrame*

```
dfUniMerge.sort_values(by='Alunos')
```

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
6	40	Comp	UNICAP	35
1	80	Med	UPE	75
2	80	Dir	UPE	60
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
7	80	Dir	UNICAP	75
3	200	Comp	UFPE	180

- Ordenando as linhas do DF
  - ***df.sort_values***

```
dfUniMerge.sort_values
```

```
Signature: dfUniMerge.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
Docstring:
Sort by the values along either axis
```

Pandas :: *DataFrame*

- Reorganizando o DF
  - ***df.pivot_table***

	Alunos	Curso	Universidade	Formandos
0	40	Comp	UPE	25
1	80	Med	UPE	75
2	80	Dir	UPE	60
3	200	Comp	UFPE	180
4	80	Med	UFPE	78
5	80	Dir	UFPE	67
6	40	Comp	UNICAP	35
7	80	Dir	UNICAP	75

```
dfUniMerge.pivot_table(values='Alunos',
                       index=['Universidade'], columns=['Curso'])
```

Curso	Comp	Dir	Med
Universidade			
UFPE	200.0	80.0	80.0
UNICAP	40.0	80.0	NaN
UPE	40.0	80.0	80.0

```
dfUniMerge.pivot_table(values=['Alunos', 'Formandos'],
                       index=['Universidade', 'Curso'])
```

		Alunos	Formandos
Universidade	Curso		
UFPE	Comp	200	180
	Dir	80	67
	Med	80	78
UNICAP	Comp	40	35
	Dir	80	75
UPE	Comp	40	25
	Dir	80	60
	Med	80	75