INTRODUÇÃO À PANDAS

LEITURA DE BASES

```
import pandas as pd
```

Leitura de arquivos CSV

df = pd.read_csv('nome_do_arquivo.csv', sep = 'caracter que separa os dados', skiprows = número de linhas a ser puladas

Exemplo

df_vendas = pd.read_csv('vendas.csv', sep = ';' , encoding = 'utf-8')

df_vendas:

Loja	Cod. Cliente	Material	Qtd.venda
BEMOL MATRIZ	1	FOGAO	1
BEMOL MATRIZ	1	PANELA	3
BEMOL EDUCANDOS	2	GELADEIRA	1
BEMOL EDUCANDOS	3	CELULAR	2

Leitura de arquivos Excel

df = pd.read_excel('nome_do_arquivo.xlsx', sheet_name = 'nome da aba', skiprows = número de linhas a ser pulada

Exemplo

df_precos = pd.read_excel('tabela_preços.xlsx', sheet_name = 'Planilha1' , skiprows = 1)

df_precos:

material	valor
FOGAO	729,00
GELADEIRA	2.432,00
CELULAR	1.169,00
HEADSET	236,00
XBOX ONE	949,00
COLCHAO	639,00
VENTILADOR	392,00
AR-CONDICIONADO	1.139,00

TRATAMENTO DE DADOS

Renomeando Colunas

Entende-se por boas práticas de programação que todas as colunas possuam letras minúsculas, sem caracteres especiais e sem espaços.

Para renomear uma ou mais colunas:

```
df = df.rename(columns={'coluna1':'novo_nome1', 'coluna2':'novo_nome2'})
```

Para renomear todas as colunas:

```
df.columns = ['novo_nome1', 'novo_nome2']
```

Exemplo

df_vendas:

Loja	Cod. Cliente	Material	Qtd.venda
BEMOL MATRIZ	1	FOGAO	1
BEMOL MATRIZ	1	PANELA	3
BEMOL EDUCANDOS	2	GELADEIRA	1
BEMOL EDUCANDOS	3	CELULAR	2

```
df_vendas = df_vendas.rename(columns = {'Loja' : 'loja', 'Cod. Cliente' : 'cod_cliente', 'Material' : 'material', 'Qtd venda' : 'qtd_venda'})
```

ou

```
df_vendas.columns = ['loja', 'cod_cliente', 'material', 'qtd_venda']
```

df_vendas:

loja	cod_cliente	material	qtd_venda
BEMOL MATRIZ	1	FOGAO	1
BEMOL MATRIZ	1	PANELA	3
BEMOL EDUCANDOS	2	GELADEIRA	1
BEMOL EDUCANDOS	3	CELULAR	2

Verificando e alterando tipo de colunas

Para verificar os tipos de todas as colunas utiliza-se a função dtypes

```
df_precos.dtypes
```

material	object
valor	object

Para verificar o tipo de uma coluna específica utiliza-se dtype

```
df_precos['valor'].dtype
```

dtype('O')

- A coluna valor foi lida como objeto (string) e precisa estar como int ou float
- Isso acontece pois os números não estão no padrão americano e então não foi reconhecido como um valor numérico pelo python
- Para transformá-lo em um valor numérico é necessário primeiro padronizá-lo eliminando os pontos(.) e substituindo as vírgulas(,) por ponto(.)

Para substituir caracteres dentro de uma coluna:

```
df['coluna'] = df['coluna'].str.replace('caracter_atual' , 'novo_caracter')
```

Exemplo

```
df_precos['valor'] = df_precos['valor'].str.replace('.' , '').str.replace(',' , '.')
```

df_precos:

material	valor
FOGAO	729.00
GELADEIRA	2432.00
CELULAR	1169.00
HEADSET	236.00
XBOX ONE	949.00
COLCHAO	639.00
VENTILADOR	392.00
AR-CONDICIONADO	1139.00

Agora pode-ser alterar o tipo da coluna de objeto para float ou int

```
df['coluna'] = df['coluna'].astype(float)
```

Exemplo

df_precos['valor'] = df_precos['valor'].astype(int)

df_precos:

material	valor
FOGAO	729
GELADEIRA	2532
CELULAR	1169
HEADSET	236
XBOX ONE	949
COLCHAO	639
VENTILADOR	392
AR-CONDICIONADO	1139

Removendo Colunas

Para deletar colunas utiliza-se a função drop

```
df = df.drop(['coluna1'], axis = 'columns')
```

Exemplo

df_vendas = df_vendas.drop(['loja', 'cod_cliente'], axis = 'columns')

df_vendas:

material	qtd_venda
FOGAO	1
PANELA	3
GELADEIRA	1
CELULAR	2

Obs 1:
Também pode-se deletar linhas utilizando a função drop

```
df = df.drop([linha], axis = 'index')
```

df_vendas.drop([1,3], axis = 'index')

material	qtd_venda
FOGAO	1
GELADEIRA	1

Obs 2:

Para reorganizar ou omitir colunas pode ser utilizar o seguinte formato passando os nomes das colunas que deseja que apareça e na ordem desejada

```
df = df[['coluna1', 'coluna2']].copy()
```

Exemplo

```
df_vendas[['qtd_venda', 'material']].copy()
```

qtd_venda	material
1	FOGAO
3	PANELA
1	GELADEIRA
2	CELULAR

Concatenando Tabelas

Quando deseja-se unir duas ou mais tabelas que possuem os mesmos conjuntos de dados utilizar a função concat:

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

Exemplo

df_vendas:

loja	material	qtd_venda
BEMOL MATRIZ	FOGAO	1
BEMOL MATRIZ	PANELA	3
BEMOL EDUCANDOS	GELADEIRA	1
BEMOL EDUCANDOS	CELULAR	2

df_vendas2:

Loja	Material	Qtd.venda
BEMOL ITACOATIARA	MOUSE	2
BEMOL ITACOATIARA	FOGAO	1
BEMOL ITACOATIARA	XBOX ONE	1
BEMOL MANACAPURU	CELULAR	4

```
df_vendas = pd.concat([df_vendas, df_vendas2])
```

df_vendas:

loja	material	qtd_venda
BEMOL MATRIZ	FOGAO	1
BEMOL MATRIZ	PANELA	3
BEMOL EDUCANDOS	GELADEIRA	1
BEMOL EDUCANDOS	CELULAR	2
BEMOL ITACOATIARA	MOUSE	2
BEMOL ITACOATIARA	FOGAO	1
BEMOL ITACOATIARA	XBOX ONE	1
BEMOL MANACAPURU	CELULAR	4

AGRUPAMENTO DE DADOS

Agrupando itens de um DataFrame

O agrupamento consiste em juntar itens iguais de uma coluna e sempre vem acompanhado de outra função aggregate que vai ditar o que acontecerá com as outras colunas

```
df  =df.groupby(['coluna1','coluna2'].agg({'coluna3' : 'metodo', 'coluna4' : 'metodo'}).reset_index()
```

Obs.: existem diversos métodos no agg, os mais utilizados são: mean (média), sum (soma), max (maior valor), min (menor valor), first (primeiro valor), last (último valor)

Exemplo

df_vendas:

loja	material	qtd_venda
BEMOL MATRIZ	FOGAO	1
BEMOL MATRIZ	PANELA	3
BEMOL EDUCANDOS	GELADEIRA	1
BEMOL EDUCANDOS	CELULAR	2
BEMOL ITACOATIARA	MOUSE	2
BEMOL ITACOATIARA	FOGAO	1
BEMOL ITACOATIARA	XBOX ONE	1
BEMOL MANACAPURU	CELULAR	4

Utilizando a tabela de vendas, podemos agrupar os valores iguais da coluna loja, assim podemos somar as quantidades e mostrar o primeiro valor da coluna material:

```
df_vendas.groupby(['loja']).agg({'qtd_venda' : 'sum', 'material' : 'first}).reset_index()
```

loja	qtd_venda	material
BEMOL MATRIZ	4	FOGAO
BEMOL EDUCANDOS	3	GELADEIRA
BEMOL ITACOATIARA	4	MOUSE
BEMOL MANACAPURU	4	CELULAR

Podemos agrupar também a tabela de vendas original por material e pedindo pra somar as quantidades de vendas dele:

```
df_vendas = df_vendas.groupby(['material']).agg({'qtd_venda' : 'sum'}).reset_index()
```

df_vendas:

material	qtd_venda
FOGAO	2
PANELA	3
GELADEIRA	1
CELULAR	6
MOUSE	2
XBOX ONE	1

CORRELACIONANDO TABELAS

XP Lab bemol

- Utilizado para unir duas tabelas através de uma ou mais colunas em comum

```
df = pd.merge(df1, df2, on = ['colunas em comum'], how = 'método de priorização')
```

Os métodos de priorização podem ser: left, right, inner, outer

Exemplo

'LEFT'

df_vendas:

material	qtd_venda
FOGAO	2
PANELA	3
GELADEIRA	1
CELULAR	6
MOUSE	2
XBOX ONE	1

df_precos:

material	valor
FOGAO	729
GELADEIRA	2532
CELULAR	1169
HEADSET	236
XBOX ONE	949
COLCHAO	639
VENTILADOR	392
AR-CONDICIONADO	1139

'left' pd.merge(df_vendas, df_precos, on = ['material'], how = 'left')

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	NaN
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	NaN
XBOX ONE	1	949

Nesse métordo primeira tabela que é passada dentro da função é mantida e então é realizada a correlação com os dados da segunda tabela de acordo com a coluna em comum

Obs: caso um valor da coluna em comum entre as duas tabelas só exista na primeira tabela (nesse caso PANELA e MOUSE) o valor é preenchido por NaN, o qual é um valor nulo

2 'RIGHT'

'right' pd.merge(df_vendas, df_precos, on = ['material'], how = 'right')

material	valor	qtd_venda
FOGAO	729	2
GELADEIRA	2532	1
CELULAR	1169	6
HEADSET	236	NaN
XBOX ONE	949	1
COLCHAO	639	NaN
VENTILADOR	392	NaN
AR-CONDICIONADO	1139	NaN

Nesse método segunda tabela que é passada dentro da função é mantida e então é realizada a correlação com os dados da primeira tabela de acordo com a coluna em comum

Obs: caso um valor da coluna em comum entre as duas tabelas só exista na segunda tabela (por exemplo: HEADSET e COLCHAO) o valor é preenchido por NaN, o qual é um valor nulo

3 'INNER'

'inner' pd.merge(df_vendas, df_precos, on = ['material'], how = 'inner')

material	qtd_venda	valor
FOGAO	2	729
GELADEIRA	1	2532
CELULAR	6	1169
XBOX ONE	1	949

Nesse método é feita a correlação apenas dos itens contidos na coluna em comum que existam em ambas as tabelas

4 'OUTER'

df_vendas = pd.merge(df_vendas, df_precos, on = ['material'], how = 'outer')

df_vendas:

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	NaN
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	NaN
XBOX ONE	1	949
HEADSET	NaN	236
COLCHAO	NaN	639
VENTILADOR	NaN	392
AR-CONDICIONADO	NaN	1139

Nesse método todos os itens da coluna em comum entre as tabelas são mantidos e realizado as devidas correlações, preenchendo com vazio (NaN) os valores não encontrados

PREENCHENDO ESPAÇOS VAZIOS (NAN)

Quando surgem valores vazios (NaN) nas tabelas as vezes torna-se necessário substituí-lo por algum valor, para realizara essa substituição utiliza-se a função **fillna**:

Para subistituir os NaN de uma coluna em específico:

```
df['coluna'] = df['coluna'].fillna(novo_valor)
```

Para subistituir os NaN da tabela inteira:

```
df = df.fillna(novo_valor)
```

Exemplo

df_vendas:

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	NaN
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	NaN
XBOX ONE	1	949
HEADSET	NaN	236
COLCHAO	NaN	639
VENTILADOR	NaN	392
AR-CONDICIONADO	NaN	1139

df_vendas['valor'].fillna(-)

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	-
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	-
XBOX ONE	1	949
HEADSET	NaN	236
COLCHAO	NaN	639
VENTILADOR	NaN	392
AR-CONDICIONADO	NaN	1139

df_vendas = df_vendas.fillna(0)

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	0
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	0
XBOX ONE	1	949
HEADSET	0	236
COLCHAO	0	639
VENTILADOR	0	392
AR-CONDICIONADO	0	1139

FILTRO

Utilizado para selecionar linhas de uma tabela de acordo com um ou mais parâmetros

```
df = df[ df[‘coluna1’] == valor1 ]
```

```
df = df[ (df[‘coluna1’] > valor1) & (df[‘coluna1) < valor2) ]
```

Exemplo

df_vendas:

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	0
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	0
XBOX ONE	1	949
HEADSET	0	236
COLCHAO	0	639
VENTILADOR	0	392
AR-CONDICIONADO	0	1139

Filtrar a tabela para apenas valores não nulos

df_vendas[df_vendas[‘valor’] != 0]

material	qtd_venda	valor
FOGAO	2	729
GELADEIRA	1	2532
CELULAR	6	1169
XBOX ONE	1	949
HEADSET	0	236
COLCHAO	0	639
VENTILADOR	0	392
AR-CONDICIONADO	0	1139

df_vendas[(df_vendas[‘valor’] != 0) & (df_vendas[‘qtd_venda’] > 1)]

material	qtd_venda	valor
FOGAO	2	729
CELULAR	6	1169

Filtrar apenas material fogão

df_vendas[df_vendas[‘material’] == ‘FOGAO’]

material	qtd_venda	valor
FOGAO	2	729

ORDENANDO VALORES

Utilizado para ordenar valores de forma crescente ou decrescente a partir de uma ou mais colunas

```
df = df.sort_values([‘coluna(s) a ser ordenada], ascending = [valor_booleano])
```

True = ordem crescente

False = ordem decrescente

Exemplo

df_vendas

material	qtd_venda	valor
FOGAO	2	729
PANELA	3	0
GELADEIRA	1	2532
CELULAR	6	1169
MOUSE	2	0
XBOX ONE	1	949
HEADSET	0	236
COLCHAO	0	639
VENTILADOR	0	392
AR-CONDICIONADO	0	1139

Ordenando por valor decrescente:

```
df_vendas = df_vendas.sort_values([‘valor’], ascending = False)
```

material	qtd_venda	valor
GELADEIRA	1	2532
CELULAR	6	1169
AR-CONDICIONADO	0	1139
XBOX ONE	1	949
FOGAO	2	729
COLCHAO	0	639
VENTILADOR	0	392
HEADSET	0	236
PANELA	3	0
MOUSE	2	0

Ordenando material de forma crescente:

```
df_vendas.sort_values([‘material’], ascending = True)
```

material	qtd_venda	valor
AR-CONDICIONADO	0	1139
CELULAR	6	1169
COLCHAO	0	639
FOGAO	2	729
GELADEIRA	1	2532
HEADSET	0	236
MOUSE	2	0
PANELA	3	0
VENTILADOR	0	392
XBOX ONE	1	949

OPERAÇÕES ENTRE COLUNAS

Sem Condicional

É possível realizar operações aritméticas simples entre colunas e constantes

Exemplo

df_vendas:

material	qtd_venda	valor
GELADEIRA	1	2532
CELULAR	6	1169
AR-CONDICIONADO	0	1139
XBOX ONE	1	949
FOGAO	2	729
COLCHAO	0	639
VENTILADOR	0	392
HEADSET	0	236
PANELA	3	0
MOUSE	2	0

```
df_vendas['valor_total'] = df_vendas['qtd_vendas'] * df_vendas['valor']
```

df_vendas:

material	qtd_venda	valor	valor_total
GELADEIRA	1	2532	2532
CELULAR	6	1169	7014
AR-CONDICIONADO	0	1139	0
XBOX ONE	1	949	949
FOGAO	2	729	1458
COLCHAO	0	639	0
VENTILADOR	0	392	0
HEADSET	0	236	0
PANELA	3	0	0
MOUSE	2	0	0

Com Condicional

Caso exista alguma condição para determinando cálculo na tabela torna-se necessário criar uma função para poder ser aplicado de forma mais rápida a condição em cada linha da tabela

Exemplo

Para calcular a divisão da coluna valor da tabela de vendas pelo valor total é necessário existir a condição onde o valor total precisa ser diferente de zero

df_vendas:

material	qtd_venda	valor	valor_total
GELADEIRA	1	2532	2532
CELULAR	6	1169	7014
AR-CONDICIONADO	0	1139	0
XBOX ONE	1	949	949
FOGAO	2	729	1458
COLCHAO	0	639	0
VENTILADOR	0	392	0
HEADSET	0	236	0
PANELA	3	0	0
MOUSE	2	0	0

```
def dividir_valores(valor, total):
  if (total != 0):
    return valor/total
  else:
    return 0
```

Para aplicar a função em todas as linhas da tabela é necessário utilizar a função apply:

```
df['nova_coluna'] = df.apply( lambda linha: funcao(linha['coluna1'), axis = 1)
```

Aplicando para a tabela de vendas:

```
df_vendas['divisao'] = df_vendas.apply( lambda linha: dividir_valores(linha['valor'], linha['valor_total']), axis = 1)
```

material	qtd_venda	valor	valor_total	divisao
GELADEIRA	1	2532	2532	1
CELULAR	6	1169	7014	0
AR-CONDICIONADO	0	1139	0	0
XBOX ONE	1	949	949	1
FOGAO	2	729	1458	1
COLCHAO	0	639	0	0
VENTILADOR	0	392	0	0
HEADSET	0	236	0	0
PANELA	3	0	0	0
MOUSE	2	0	0	0

EXPORTANDO DADOS

Exportando para o excel

Primeiro deve-se criar um arquivo em excel em branco utilizando o seguinte comando:

```
arquivo = pd.ExcelWriter('nome_do_arquivo.xlsx', engine='xlsxwriter')
```

Após o arquivo ser criado podemos preenchê-lo com as tabelas criadas seguindo o formato:

```
df.to_excel(arquivo, sheet_name = 'nome da aba onde o arquivo será inserido', index = False,
skiprows = número da linha onde se deseja iniciar a tabela, skipcolumns = número da coluna onde se deseja iniciar a tabela)
```

Por último, gerando o arquivo já montado:

```
arquivo.save()
```

Exemplo

```
arquivo = pd.ExcelWriter('vendas.xlsx', engine='xlsxwriter')
vendas.to_excel(arquivo, sheet_name = 'total de vendas', index = False)
arquivo.save()
```

Exportando para CSV

Para exportar para um arquivo com extensão .csv apenas seguir o formato:

```
df.to_csv('nome_do_arquivo.csv', sep = 'caracter que separa os dados', index=False)
```

Exemplo

```
df_vendas.to_csv('vendas.csv', sep = ';', index = False)
```