BLOG

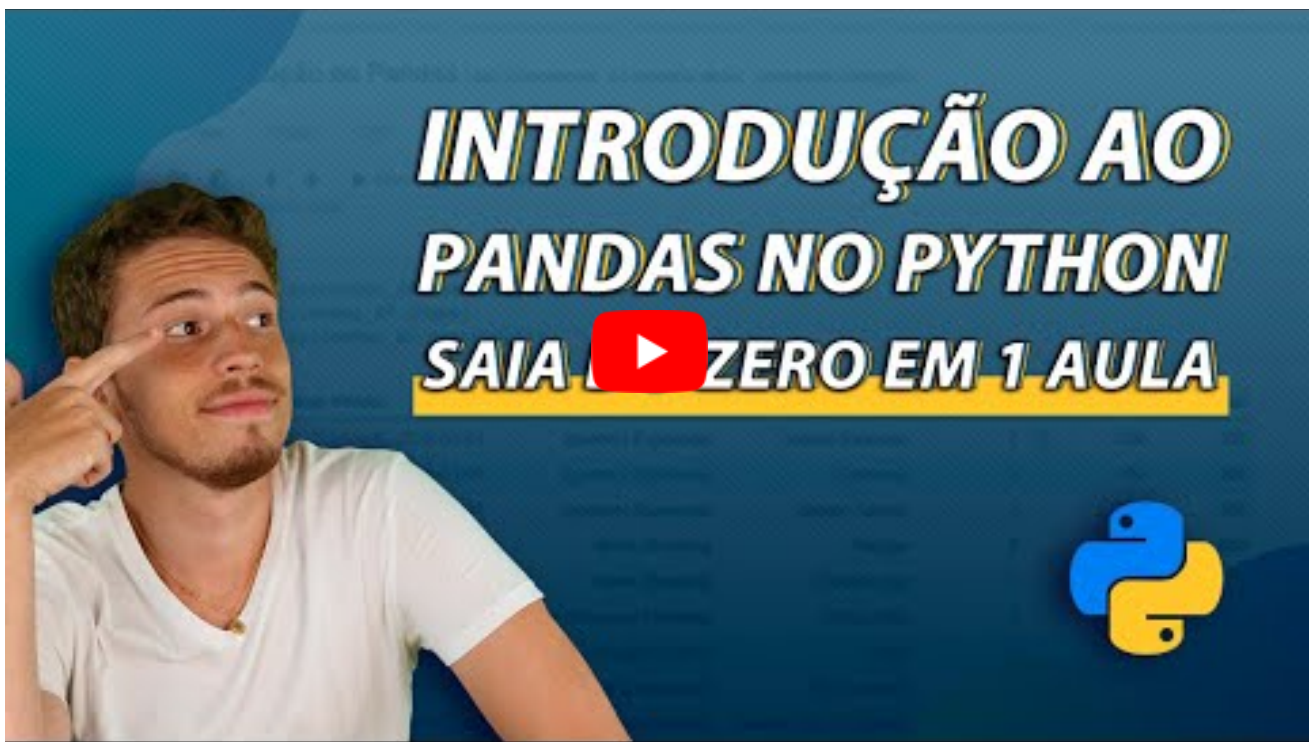
Postado em**Python**, em 19 de abril de 2021

INTRODUÇÃO AO PANDAS – SAIA DO ZERO EM 1 AULA

Automação de Programa ou Sistema

Nessa aula vamos te mostrar a introdução ao Pandas e tudo o que precisa saber nesse início para trabalhar com dados dentro do Python!

Caso prefira esse conteúdo no formato de vídeo-aula, assista ao vídeo abaixo ou acesse o nosso canal do YouTube!



Para receber a planilha que usamos na aula no seu e-mail, preencha:

Seu melhor e-mail

Enviar

Introdução ao Pandas

Você que quer começar a programar em Python já deve ter se perguntado o que é esse tal de Pandas?

Não se preocupe que hoje nós vamos te mostrar a introdução ao Pandas, que é uma biblioteca do Python e vamos abordar os principais comandos do panda para que você possa utilizá-lo de forma eficiente.

Essa é a melhor biblioteca do Python para dados, então você vai utilizar bastante ela e os comandos contidos nela para fazer suas análises.

Tratamento de Dados no Python

O primeiro passo para a introdução ao Pandas é fazer a importação dessa biblioteca para dentro do Python.

OBS: Lembrando que utilizaremos o Jupyter nesta aula, então se ainda não tem no seu computador temos um vídeo explicando como fazer a instalação completa dele, basta **clicar aqui** para acessar o conteúdo!

Vamos fazer dessa maneira, pois é uma forma de facilitar a escrita dos códigos utilizando essa biblioteca, pois por padrão temos que escrever pandas.(comando desejado). Para introdução ao Pandas e até mesmo para quem já utiliza a algum tempo é muito mais cômodo dessa forma.

Então ao invés de sempre escrever pandas, vamos poder escrever pd.(comando desejado) diminuindo o tamanho da escrita e facilitando na hora de programar.

Outro ponto importante é entender que o pandas funciona com DataFrames, que nada mais são do que tabelas dentro do Python.

Agora nós vamos te mostrar como criar um dateframe a partir de um dicionário (que seria uma descrição do que aconteceu).

Criando um dataframe a partir de um dicionário

```
# dataframe = pd.DataFrame() # Cria um dataframe vazio!
venda = {'data': ['15/02/2021', '16/02/2021'],
         'valor': [500, 300],
         'produto': ['feijao', 'arroz'],
         'qtde': [50, 70],
        }
vendas_df = pd.DataFrame(venda)
```

Criando um dataframe

Neste caso temos a utilização do **pd.DataFrame()** logo no início, mas isso é para criamos um dateframe vazio o que não é muito usual.

Por isso na linha logo abaixo estamos criando um dicionário de vendas, então temos algumas informações de venda como: data, valor, produto e quantidade.

OBS: É importante verificar como é a estrutura desse dicionário para que os dados sejam armazenados de forma correta.

Na última linha vamos criar uma variável para poder atribuir o nosso dataframe com o código pd.DateFrame(venda). Então estamos atribuindo uma tabela a essa variável vendas_df.

OBS: Esse **df** que foi colocado na variável é apenas um indicativo para facilitar, para sabermos que essa variável é um dataframe. Você pode colocar **tabela_vendas** por exemplo!

A criação de dataframe é muito importante para a visualização dos dados, pois se o usuário simplesmente colocar um print(vendas) terá apenas o dicionário sendo mostrado na tela.

Visualização dos Dados

```
- print
- display
```

```
print(vendas_df)
display(vendas_df)
```

```
         data  valor produto  qtde
0  15/02/2021    500   feijao    50
1  16/02/2021    300    arroz    70
```

	data	valor	produto	qtde
0	15/02/2021	500	feijao	50
1	16/02/2021	300	arroz	70

Visualização dos dados

Aqui nós vamos verificar a diferença entre a visualização de dados no Python com **print** e **display**.

Essas duas opções vão te dar o mesmo resultado, no entanto com o print, nós temos um aspecto mais de bloco de notas (mas ainda organizado).

Já com a utilização do display, nós temos algo bem mais visual e mais fácil de visualizar os dados, então quando vamos mostrar algum resultado é importante também verificar se as informações estão sendo mostrados de uma forma fácil de entender.

IMPORTANTE: Para esse exemplo vamos abrir um arquivo em Excel, no entanto esse arquivo está no mesmo local onde temos o arquivo do nosso código.

Importando arquivos e bases de dados

```
vendas_df = pd.read_excel("Vendas.xlsx")
display(vendas_df)
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71
...	...	...	...	...	...	...	...
93905	65012	2019-11-30	Shopping Vila Velha	Mochila Xadrez	2	283	566
93906	65013	2019-11-30	Ribeirão Shopping	Pulseira Listrado	2	79	158
93907	65013	2019-11-30	Ribeirão Shopping	Cueca Listrado	3	67	201
93908	65014	2019-11-30	Shopping Morumbi	Pulseira Linho	5	114	570
93909	65014	2019-11-30	Shopping Morumbi	Casaco Xadrez	4	259	1036

93910 rows × 7 columns

Importando arquivos e base de dados

Caso você queira puxar o arquivo de outro local terá que colocar o caminho completo do arquivo onde escrevemos o nome dele.

É algo que dá mais trabalho, mas funciona tranquilamente, é que fica mais cômodo e fácil escrever apenas o nome do arquivo.

Veja que temos a planilha sendo mostrada normalmente no formato de dataframe já com o visual mais adequado com a utilização do display.

IMPORTANTE: É bom salientar que esse código ao ser executado pode demorar no seu computador, pois é uma base de dados com 90 mil linhas, então de fato tem uma quantidade considerável de dados.

Outro ponto que é bom levar em conta é que o Python mostrou apenas algumas linhas do início e do fim da tabela para não ter que mostrar todos os dados e deixar o usuário perdido.

Mas dessa forma é possível com que o usuário veja a estrutura da tabela para que possa trabalhar com essas informações de forma adequada.

Agora vamos ver a parte de resumos de visualização de dados simples e úteis. O que isso quer dizer? Que temos alguns métodos para facilitar a visualização.

- describe

```
display(vendas_df.head(10))
print(vendas_df.shape)
display(vendas_df.describe())
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71
5	3	2019-01-02	Rio Mar Shopping Fortaleza	Cinto Linho	1	248	248
6	5	2019-01-02	Shopping Barra	Calça	1	170	170
7	6	2019-01-02	Shopping Ibirapuera	Polo Listrado	4	149	596
8	7	2019-01-02	Norte Shopping	Camisa Gola V Listrado	1	116	116
9	7	2019-01-02	Norte Shopping	Camisa Liso	1	105	105

```
(93910, 7)
```

	Código Venda	Quantidade	Valor Unitário	Valor Final
count	93910.000000	93910.000000	93910.000000	93910.000000
mean	32464.762155	2.166553	191.725886	414.862656
std	18809.007093	1.258732	145.215519	434.846228
min	1.000000	1.000000	30.000000	30.000000
25%	16204.000000	1.000000	100.000000	156.000000
50%	32367.000000	2.000000	155.000000	274.000000
75%	48793.750000	3.000000	248.000000	524.000000
max	65014.000000	5.000000	750.000000	3750.000000

Resumo de Visualizações de Dados Simples e Úteis

Aqui inicialmente estamos utilizar o **.head()** que é para que o usuário escolha quantas linhas deseja visualizar dessa base de dados.

Por padrão o Python coloca apenas as 5 primeiras linhas, mas neste exemplo colocamos as 10 primeiras linhas dessa base de dados.

Esse método é importante para que você possa verificar se os dados estão corretos e se a estrutura da tabela também está certa.

No segundo exemplo temos o método **.shape**, que vai nos mostrar quantas linhas e quantas colunas essa base de dados possui.

Por fim vamos verificar o método **.describe** que é muito útil e interessante. Ele vai te dar um resumo das informações numéricos que temos na nossa base de dados.

Então você terá uma visão geral desses itens e um resumo para poder facilitar certas análises sem que seja necessário fazer qualquer tratamento na tabela.

Agora nós vamos passar para os **métodos de edição do dataframe**.

IMPORTANTE: É muito importante ressaltar que sempre que tivermos pd.series quer dizer que temos uma série do pandas, o que é isso?

Nada mais é do que uma única coluna ou uma única linha do seu dataframe. É importante falar isso, pois o próximo método que vamos utilizar é para pegar colunas específicas.

E se você pegar apenas uma coluna, vai ver que mesmo com o método display, ela não vai aparecer toda formatada e bonita.

Então quando utilizarmos dessa forma: **produtos = vendas_df['Produto']** teremos apenas uma única coluna não formatada.

Agora para mais colunas já podemos colocar outros 2 colchetes e trazer normalmente, já formatado.

```
display(produtos)
```

	Produto	ID Loja
0	Sapato Estampa	Iguatemi Esplanada
1	Camiseta	Iguatemi Esplanada
2	Sapato Xadrez	Iguatemi Esplanada
3	Relógio	Norte Shopping
4	Chinelo Liso	Norte Shopping
...	...	...
93905	Mochila Xadrez	Shopping Vila Velha
93906	Pulseira Listrado	Ribeirão Shopping
93907	Cueca Listrado	Ribeirão Shopping
93908	Pulseira Linho	Shopping Morumbi
93909	Casaco Xadrez	Shopping Morumbi

93910 rows × 2 columns

Pegar 1 Coluna

Esse método é para pegarmos apenas colunas, mas e se você quiser pegar uma linha, ou linhas, ou até mesmo um valor específico?

Para isso vamos utilizar o método **.loc[]** para poder fazer essa parte mais específica.

```
# pegar uma linha
display(vendas_df.loc[1:5])

# pegar linhas que correspondem a uma condição
vendas_norteshopping_df = vendas_df.loc[vendas_df['ID Loja'] == 'Norte Shopping']

# pegar várias linhas e colunas usando o loc
vendas_norteshopping_df = vendas_df.loc[vendas_df['ID Loja'] == 'Norte Shopping', ["ID Loja", "Produto", "Quantidade"]]
display(vendas_norteshopping_df)

# pegar 1 valor específico
print(vendas_df.loc[1, 'Produto'])
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71
5	3	2019-01-02	Rio Mar Shopping Fortaleza	Cinto Linho	1	248	248

	ID Loja	Produto	Quantidade
3	Norte Shopping	Relógio	3
4	Norte Shopping	Chinelo Liso	1
8	Norte Shopping	Camisa Gola V Listrado	1
9	Norte Shopping	Camisa Liso	1
100	Norte Shopping	Cueca Xadrez	5
...	...	...	...
93716	Norte Shopping	Polo Xadrez	1
93754	Norte Shopping	Calça Liso	1
93755	Norte Shopping	Casaco Estampa	2
93756	Norte Shopping	Gorro	2
93852	Norte Shopping	Bermuda Linho	1

3924 rows × 3 columns

```
Camiseta
```

Pegar várias linhas e/ou colunas

IMPORTANTE: No primeiro método estamos pegando da linha 1 até a linha 5, no entanto o pandas vai considerar os números que fica a esquerda, que ele mesmo atribui. Então é muito importante lembrar que ele começa no zero para não perder a primeira informação.

No primeiro exemplo estamos apenas pegando as linhas de 1 a 5 da nossa tabela.

No segundo exemplo estamos pegando todas as informações, na qual a coluna ID Loja é igual a Norte Shopping, ou seja, estamos limitando a nossa busca essa informação apenas.

No terceiro exemplo vamos repetir o que fizemos no segundo, mas vamos escolher as colunas que vamos armazenar com esses dados, isso é importante quando não precisa ou não quer mostrar todas as colunas da tabela.

```
# display(vendas_df)

# criar uma coluna com valor padrão
vendas_df.loc[:, "Imposto"] = 0
display(vendas_df)
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final	Comissão	Imposto
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358	17.90	0
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360	18.00	0
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368	18.40	0
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600	30.00	0
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71	3.55	0
...	...	...	...	...	...	...	...	...	...
93905	65012	2019-11-30	Shopping Vila Velha	Mochila Xadrez	2	283	566	28.30	0
93906	65013	2019-11-30	Ribeirão Shopping	Pulseira Listrado	2	79	158	7.90	0
93907	65013	2019-11-30	Ribeirão Shopping	Cueca Listrado	3	67	201	10.05	0
93908	65014	2019-11-30	Shopping Morumbi	Pulseira Linho	5	114	570	28.50	0
93909	65014	2019-11-30	Shopping Morumbi	Casaco Xadrez	4	259	1036	51.80	0

93910 rows × 9 columns

Adicionar 1 coluna

Agora nós vamos ver como podemos criar ou adicionar uma coluna dentro da nossa tabela.

Existem duas maneiras, a primeira que é utilizando uma coluna já existente para compor a nova, ou atribuindo um valor padrão a todas as informações dessa coluna.

OBS: Lembrando que quando utilizamos os **:** (dois pontos) dentro do loc isso significa que estamos querendo selecionar todas as linhas ou colunas (depende de onde colocou).

Agora que aprendemos a inserir colunas, vamos também aprender como podemos inserir linhas, ou seja, como podemos inserir novos dados no nosso dataframe.

Adicionar 1 linha

```
- Linhas de um complemento da base de dados
```

```
vendas_dez_df = pd.read_excel("Vendas - Dez.xlsx")
#display(vendas_dez_df)

vendas_df = vendas_df.append(vendas_dez_df)
display(vendas_df)
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final	Comissão	Imposto
0	1	2019-01-01	Iguatemi Esplanada	Sapato Estampa	1	358	358	17.90	0.0
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360	18.00	0.0
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368	18.40	0.0
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600	30.00	0.0
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71	3.55	0.0
...	...	...	...	...	...	...	...	...	...
7084	69996	2019-12-26	Center Shopping Uberlândia	Short Listrado	2	102	204	NaN	NaN
7085	69996	2019-12-26	Center Shopping Uberlândia	Mochila	4	270	1080	NaN	NaN
7086	69996	2019-12-26	Center Shopping Uberlândia	Pulseira Estampa	1	87	87	NaN	NaN
7087	69997	2019-12-26	Ribeirão Shopping	Camisa Listrado	1	108	108	NaN	NaN
7088	69997	2019-12-26	Ribeirão Shopping	Short Linho	2	133	266	NaN	NaN

100999 rows × 9 columns

Adicionar 1 linha

Neste caso estamos importando novamente para o Python a base de vendas que contém todas as vendas de dezembro.

Em seguida vamos fazer a junção desses dados para que nossa base de dados fique completa com os dados que temos, mais os dados de dezembro.

Para isso vamos utilizar o método **.append()** para indicar que queremos inserir ao vendas_df as informações da base vendas_dez_df.

Bem, agora que aprendemos a inserir linhas e colunas, vamos aprender como fazer para poder excluir linhas e colunas.

```
vendas_df = vendas_df.drop(0, axis=0)
display(vendas_df)
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final	Comissão
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360	18.00
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368	18.40
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600	30.00
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71	3.55
5	3	2019-01-02	Rio Mar Shopping Fortaleza	Cinto Linho	1	248	248	12.40
...	...	...	...	...	...	...	...	...
7084	69996	2019-12-26	Center Shopping Uberlândia	Short Listrado	2	102	204	NaN
7085	69996	2019-12-26	Center Shopping Uberlândia	Mochila	4	270	1080	NaN
7086	69996	2019-12-26	Center Shopping Uberlândia	Pulseira Estampa	1	87	87	NaN
7087	69997	2019-12-26	Ribeirão Shopping	Camisa Listrado	1	108	108	NaN
7088	69997	2019-12-26	Ribeirão Shopping	Short Linho	2	133	266	NaN

100997 rows × 8 columns

Excluir linhas e colunas

Nesse caso é importante verificar os argumentos do método **.drop()**, pois no primeiro argumento vamos precisar do número da linha ou nome da coluna.

E no segundo argumento temos que ter o eixo que essa ação vai acontecer, então se o **eixo for igual a 0** estaremos no **eixo das linhas**, caso o **eixo seja igual a 1** estaremos no **eixo das colunas**.

Até esse momento você já aprendeu os comandos básicos do pandas, no entanto sempre que for fazer análise de dados ou tratamento de dados temos alguns comandos importantes.

Então já vamos aproveitar e passar isso nessa parte extra!

EXTRA – Para Tratamento e Análise de Dados

Os primeiros comandos que vamos apresentar são os comandos para tratar valores vazios, ou seja, aqueles valores que você viu na nossa tabela que estavam como **NaN**.

Valores Vazios

```
- Deletar linhas/colunas vazias
- Deletar linhas que possuem valores vazios
- Preencher valores vazios (média e último valor)
```

```
# deletar linhas e colunas completamente vazias
vendas_df = vendas_df.dropna(how='all', axis=1)

# deletar linhas que possuem pelo menos 1 valor vazio
vendas_df = vendas_df.dropna()

# preencher valores vazios
# preencher com a média da coluna
vendas_df['Comissão'] = vendas_df['Comissão'].fillna(vendas_df['Comissão'].mean())
display(vendas_df)

# preencher com o último valor
vendas_df = vendas_df.ffill()
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final	Comissão
1	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360	18.000000
2	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368	18.400000
3	2	2019-01-02	Norte Shopping	Relógio	3	200	600	30.000000
4	2	2019-01-02	Norte Shopping	Chinelo Liso	1	71	71	3.550000
5	3	2019-01-02	Rio Mar Shopping Fortaleza	Cinto Linho	1	248	248	12.400000
...	...	...	...	...	...	...	...	...
7084	69996	2019-12-26	Center Shopping Uberlândia	Short Listrado	2	102	204	20.743163
7085	69996	2019-12-26	Center Shopping Uberlândia	Mochila	4	270	1080	20.743163
7086	69996	2019-12-26	Center Shopping Uberlândia	Pulseira Estampa	1	87	87	20.743163
7087	69997	2019-12-26	Ribeirão Shopping	Camisa Listrado	1	108	108	20.743163
7088	69997	2019-12-26	Ribeirão Shopping	Short Linho	2	133	266	20.743163

100997 rows × 8 columns

Valores vazios

No primeiro exemplo temos algo parecido com o método para excluir linhas e colunas, só que neste caso nós vamos passar o argumento **how** (como) sendo igual a **all** (todos) para utilizar o método **.dropna()** corretamente.

No segundo exemplo nós vamos utilizar quando queremos excluir uma linha inteira se ao menos um dos valores for vazio.

No terceiro exemplo vamos preencher valores vazios com a média dos valores que já temos nessa coluna. Então vamos utilizar o método **.fillna()** que é para preencher, juntamente com o **.mean()** que é de fato a média.

No quarto exemplo temos outra maneira de preencher que é utilizando o valor que está logo acima dele.

Isso é muito utilizado quando temos uma base de dados onde não querem repetir os itens, nomes, produtos... então colocam somente uma vez, então para preencher sempre com o valor que está acima, vamos utilizar o método **ffill()**.

Agora vamos para uma parte muito interessante e muito utilizada na análise de dados, que é a parte de como calcular os indicadores.

Isso quer dizer, qual o valor total, qual o faturamento por loja e assim por diante.

Calcular Indicadores

- Groupby
- Value Counts

```
# value counts
transacoes_loja = vendas_df['ID Loja'].value_counts()
display(transacoes_loja)

# group by
faturamento_produto = vendas_df[['Produto', 'Valor Final']].groupby('Produto').sum()
display(faturamento_produto)
```

```
Shopping Vila Velha                 4234
Palladium Shopping Curitiba         4210
Norte Shopping                      4179
Rio Mar Shopping Fortaleza          4118
Bourbon Shopping SP                 4116
Iguatemi Campinas                   4108
Rio Mar Recife                      4099
Shopping Center Leste Aricanduva    4093
Shopping SP Market                  4080
Shopping Ibirapuera                 4051
Novo Shopping Ribeirão Preto        4049
Ribeirão Shopping                   4048
Salvador Shopping                   4030
Shopping Iguatemi Fortaleza         4021
Shopping Center Interlagos          4021
Center Shopping Uberlândia          4013
Shopping Eldorado                   4002
Passei das Águas Shopping           4000
Shopping União de Osasco            3995
Iguatemi Esplanada                  3979
Shopping Barra                      3962
Shopping Morumbi                    3959
Parque Dom Pedro Shopping           3902
Shopping Recife                     3891
Shopping Midway Mall                3837
Name: ID Loja, dtype: int64
```

Produto	Valor Final
Bermuda	272250
Bermuda Estampa	291694
Bermuda Linho	394680
Bermuda Liso	275692
Bermuda Listrado	293237
...	...
Tênis Estampa	457728
Tênis Linho	538608
Tênis Liso	474544
Tênis Listrado	481032
Tênis Xadrez	459725

120 rows × 1 columns

Cálculo de indicadores

No primeiro exemplo temos o método **.value_counts()** que serve para contar os valores que temos dentro de uma coluna.

Neste caso estamos contando a quantidade de transações que foi feita por loja, então teremos um resumo de quantas transações cada loja fez de forma fácil e rápida.

Neste caso estamos mostrando somente duas colunas, pois algumas vezes não queremos mostrar toda a tabela, então é importante ocultar alguns detalhes quando necessário.

Por fim estamos agrupando por produtos, ou seja, vamos ter todos os produtos e vamos somar o valor final de cada um deles, desta forma vamos saber qual o valor total de cada um dos produtos dessa loja.

Agora vamos para o último método que vamos explicar nessa aula, que é o método para mesclar 2 dataframes, ou seja, nós vamos conseguir procurar informações de uma dataframe no outro.

Isso quer dizer que vamos conseguir fazer uma busca entre duas tabelas diferentes.

IMPORTANTE: É necessário que essas duas tabelas tenham uma coluna com informações em comum para que a busca possa ser feita.

Mesclar 2 dataframes (Procurar informações de um dataframe em outro)

```
gerentes_df = pd.read_excel("Gerentes.xlsx")
# display(gerentes_df)

vendas_df = vendas_df.merge(gerentes_df)
display(vendas_df)
```

	Código Venda	Data	ID Loja	Produto	Quantidade	Valor Unitário	Valor Final	Comissão	Gerente
0	1	2019-01-01	Iguatemi Esplanada	Camiseta	2	180	360	18.000000	Salvador
1	1	2019-01-01	Iguatemi Esplanada	Sapato Xadrez	1	368	368	18.400000	Salvador
2	21	2019-01-02	Iguatemi Esplanada	Camisa Gola V Listrado	2	116	232	11.600000	Salvador
3	34	2019-01-02	Iguatemi Esplanada	Sapato Listrado	1	363	363	18.150000	Salvador
4	34	2019-01-02	Iguatemi Esplanada	Camisa Gola V Liso	2	118	236	11.800000	Salvador
...	...	...	...	...	...	...	...	...	...
100992	69846	2019-12-25	Salvador Shopping	Short Estampa	2	96	192	20.743163	Mateus
100993	69846	2019-12-25	Salvador Shopping	Tênis Estampa	5	256	1280	20.743163	Mateus
100994	69850	2019-12-25	Salvador Shopping	Calça Estampa	4	177	708	20.743163	Mateus
100995	69972	2019-12-26	Salvador Shopping	Terno Liso	3	720	2160	20.743163	Mateus
100996	69972	2019-12-26	Salvador Shopping	Sapato Estampa	5	358	1790	20.743163	Mateus

100997 rows × 9 columns

Mesclar 2 dataframes

Primeiramente vamos importar o arquivo, utilizando novamente o método **.read_excel()**, em seguida vamos poder fazer a mescla utilizando o método **.merge()**.

Como já temos uma coluna com o mesmo nome o pandas já vai fazer essa busca e já vai retornar as informações da tabela que vamos mesclar.

Isso quer dizer que vamos inserir as informações da tabela gerentes dentro da base de dados que já temos.

Aqui finalizamos a nossa introdução ao Pandas, gostou de tudo que aprendeu?

Conclusão da Introdução ao Pandas

Nessa aula conseguimos passar uma introdução ao Pandas para que você possa iniciar seus trabalhos com ciência de dados de forma eficiente.

Então já vai conseguir tratar dados, juntar informações, criar dataframes, entre outras ações.

Essa biblioteca é muito útil e muito utilizada, por isso é considerada a melhor biblioteca para análise de dados.

Então agora que já teve uma introdução dos diversos métodos basta colocar em prática para poder fixar tudo o que ensinamos e já começar com sua análise de dados!

Hashtag Treinamentos

Para acessar outras publicações de Python, clique aqui!

Quer aprender mais sobre Python com um minicurso gratuito?

Preencha com o seu e-mail*

QUERO ME CADASTRAR

Fale Conosco

Cursos Online e Presenciais

Amanda Egler
listavip@hashtagtreinamentos.com
(61) 9331-4870 (telefone e WhatsApp)

Endereços

Av. Rio Branco, 124 - 12º andar - Centro, Rio de Janeiro - RJ, 20040-001

Newsletter

Seu primeiro nome*

Seu primeiro nome

Seu melhor e-mail*

Seu melhor e-mail

ENVIAR