desafios-pandas

September 1, 2025

1 Desafios Pandas

[1]:
```
# Importando bibliotecas
import pandas as pd
import numpy as np
```

1.1 DESAFIO 1

Dada uma tabela de funcionários e uma de diretoria, selecione as diretorias que possuam profissionais que ganham mais do que 5 mil reais em salário e adicione uma coluna com o percentual que esses funcionários representam de sua diretoria e uma com a quantidade absoluta deles. Exemplo:

Entrada:

tabela de funcionários

Colunas Tipo

id INTEGER

primeiro_nome VARCHAR

ultimo_nome VARCHAR

salario INTEGER

diretoria_id INTEGER

tabela de diretorias

Colunas Tipo

id INTEGER

nome VARCHAR

Saída:

Coluna Tipo

diretoria_nome VARCHAR

percentual_acima_5k INTEGER

total_funcionarios_acima_5k INTEGER

[2]:
```
# Analisando os DataFrames
diretoria = pd.read_csv('desafio1-arquivos/diretorias.csv')
funcionario = pd.read_csv('desafio1-arquivos/funcionarios.csv')

print(diretoria.head(5))
print(funcionario.head(5))
print(diretoria.shape)
print(funcionario.shape)
```

```
   Unnamed: 0  id              nome
0           0   1        Tecnologia
1           1   2  Recursos Humanos
2           2   3        Financeiro
3           3   4         Marketing
   Unnamed: 0  id primeiro_nome ultimo_nome  salario  diretoria_id
0           0   1           Ana       Silva     7000             1
1           1   2         Bruno    Oliveira     4500             1
2           2   3        Carlos      Santos     8000             2
3           3   4         Diana       Costa     3500             2
4           4   5       Eduardo     Pereira     9000             3
(4, 3)
(10, 6)
```

[7]:
```
'''
Selecione as diretorias que possuam profissionais que ganham mais do que 5 mil␣
 ↪reais em salário
e adicione uma coluna com o percentual que esses funcionários representam de␣
 ↪sua diretoria
e uma com a quantidade absoluta deles
'''

# Juntando os dois datasets
desafio1 = pd.merge(funcionario, diretoria, left_on='diretoria_id',␣
 ↪right_on='id', suffixes=('_func', '_dir'))
desafio1
```

[7]:

	Unnamed: 0_func	id_func	primeiro_nome	ultimo_nome	salario	diretoria_id	\
0	0	1	Ana	Silva	7000	1	
1	1	2	Bruno	Oliveira	4500	1	
2	2	3	Carlos	Santos	8000	2	
3	3	4	Diana	Costa	3500	2	
4	4	5	Eduardo	Pereira	9000	3	
5	5	6	Fernanda	Souza	2000	3	
6	6	7	Gabriel	Lima	5000	3	
7	7	8	Helena	Fernandes	7500	3	
8	8	9	Igor	Carvalho	3000	4	
9	9	10	Juliana	Rodrigues	6500	4	

	Unnamed: 0_dir	id_dir	nome
0	0	1	Tecnologia
1	0	1	Tecnologia
2	1	2	Recursos Humanos
3	1	2	Recursos Humanos
4	2	3	Financeiro
5	2	3	Financeiro
6	2	3	Financeiro
7	2	3	Financeiro
8	3	4	Marketing
9	3	4	Marketing

[22]:
```
# Selecionando apenas quem tem salario acima de 5k
func_acima_5k = desafio1[desafio1['salario'] > 5000]

# Contando por diretoria
func_acima5k_por_diretoria = func_acima_5k.groupby('nome')['id_func'].count().
 ↪reset_index(name='func_acima5k_por_diretoria')
# Contando total por diretoria
total_por_diretoria = desafio1.groupby('nome')['id_func'].count().
 ↪reset_index(name='total_por_diretoria')

# Juntando as duas variaveis
diretoria = pd.merge(func_acima5k_por_diretoria, total_por_diretoria, on='nome')
# Adicionando a porcentagem
diretoria['perc_por_diretoria'] = (diretoria['func_acima5k_por_diretoria'] /␣
 ↪diretoria['total_por_diretoria']) * 100

# Mostrando o resultado
print(diretoria[['nome', 'perc_por_diretoria', 'func_acima5k_por_diretoria']])
```

	nome	perc_por_diretoria	func_acima5k_por_diretoria
0	Financeiro	50.0	2
1	Marketing	50.0	1
2	Recursos Humanos	50.0	1

```
3       Tecnologia              50.0                    1
```

1.2 DESAFIO 2

Os alunos de uma escola participam de um exame abrangente cada bimestre, conhecido como "Provão", que cobre todas as disciplinas. As notas deste exame variam de 0 a 100. A partir de um dataset contendo as notas de todos os "Provões", o objetivo é criar um novo dataset. Este novo dataset deve conter as notas de cada aluno em cada um dos "Provões" como colunas separadas e uma coluna adicional indicando se o aluno foi aprovado, considerando que a aprovação requer uma média superior a 50 em todos os exames. Além disso, queremos identificar qual turma tem o maior número de alunos reprovados no "Provão".

Entrada:

Coluna Tipo

id_aluno INTEGER

turma_id VARCHAR

primeiro_nome VARCHAR

ultimo_nome VARCHAR

id_provao INTEGER

nota_provao INTEGER

Saída:

Coluna Tipo

id_aluno INTEGER

turma_id VARCHAR

nota_provao_1 INTEGER

nota_provao_2 INTEGER

nota_provao_3 INTEGER

nota_provao_4 INTEGER

aprovado BOOLEAN (TRUE, se aprovado)

[31]:
```
# Importando os datasets e analisando
provao = pd.read_csv('desafio2-arquivos/provao_notas.csv')
print(provao.head(1))
print(provao.describe())
print(provao.columns)
```

```
   Unnamed: 0  id_aluno turma_id primeiro_nome ultimo_nome  id_provao  \
0           0         3        B       Juliana   Rodrigues          2

   nota_provao
0           10
       Unnamed: 0   id_aluno  id_provao  nota_provao
count   40.000000  40.000000  40.000000    40.000000
mean    19.500000   5.500000   2.500000    57.475000
std     11.690452   2.908872   1.132277    25.892269
min      0.000000   1.000000   1.000000     2.000000
25%      9.750000   3.000000   1.750000    46.500000
50%     19.500000   5.500000   2.500000    60.000000
75%     29.250000   8.000000   3.250000    72.000000
max     39.000000  10.000000   4.000000    97.000000
Index(['Unnamed: 0', 'id_aluno', 'turma_id', 'primeiro_nome', 'ultimo_nome',
       'id_provao', 'nota_provao'],
      dtype='object')
```

[40]:
```
"""
A partir de um dataset contendo as notas de todos os "Provões", o objetivo é␣
 ↪criar um novo dataset.
Este novo dataset deve conter as notas de cada aluno em cada um dos "Provões"␣
 ↪como colunas separadas e uma coluna adicional indicando
se o aluno foi aprovado, considerando que a aprovação requer uma média superior␣
 ↪a 50 em todos os exames.
Além disso, queremos identificar qual turma tem o maior número de alunos␣
 ↪reprovados no "Provão".
"""

# Criando um novo dataset com colunas separadas por "provões"
provao_pivot = provao.pivot_table(index=['id_aluno', 'turma_id'],
                                  columns='id_provao',
                                  values='nota_provao').reset_index()
# Renomeando as colunas para corresponder ao formato desejado
provao_pivot.columns = ['id_aluno', 'turma_id', 'nota_provao_1',␣
 ↪'nota_provao_2', 'nota_provao_3', 'nota_provao_4']

# Adicionando a coluna aprovado
provao_pivot['aprovado'] = provao_pivot[['nota_provao_1', 'nota_provao_2',␣
 ↪'nota_provao_3', 'nota_provao_4']].mean(axis=1) > 50
```

```
# Contando o número de reprovados por turma
contagem_reprovados = provao_pivot.groupby('turma_id')['aprovado'].apply(lambda
↪x: (x == False).sum())

# Identificando a turma com o maior número de reprovados
turma_mais_reprovados = contagem_reprovados.idxmax()

# Turma com mais reprovados
turma_mais_reprovados
```

[40]: 'B'

1.2.1 Alguns aprendizados

- **.mean(axis=1):** Calcula a média das 4 notas para cada aluno (axis=1 = por linha)
- **idxmax():** Retorna o índice (turma_id) do valor máximo na série

1.3 DESAFIO 3

A sua loja possui os registros das compras de todos os clientes numa tabela com as seguintes informações:

```
id_cliente
estado
data_da_compra
valor_compra
```

A partir dessa tabela, precisamos que você:

- Crie uma nova tabela contendo a informação de id_cliente, qtd_compras (número de compras), valor_total_compras (valor total gasto), valor_medio_compras (valor médio das compras), data_ultima_compra.
- Identifique o estado que mais vendeu em termos de receita e o que mais vendeu em unidades.

[41]:
```
# Importando o dataset e analisando
loja = pd.read_csv('desafio3-arquivos/compras.csv')
print(loja.head())
print(loja.describe())
print(loja.shape)
```

```
   Unnamed: 0  id_cliente estado data_da_compra  valor_compra
0           0           4     RJ     2023-01-01           184
1           1           5     RJ     2023-01-02            70
2           2           3     SP     2023-01-03           378
3           3           5     SP     2023-01-04           216
4           4           5     SP     2023-01-05           323
       Unnamed: 0  id_cliente  valor_compra
count    20.00000   20.000000     20.000000
```

```
mean       9.50000    3.600000    297.000000
std        5.91608    1.231174    139.054703
min        0.00000    1.000000     63.000000
25%        4.75000    3.000000    173.250000
50%        9.50000    4.000000    318.500000
75%       14.25000    5.000000    400.250000
max       19.00000    5.000000    493.000000
(20, 5)
```

[54]:
```
"""
- Crie uma nova tabela contendo a informação de id_cliente, qtd_compras (número␣
 ↪de compras),
valor_total_compras (valor total gasto), valor_medio_compras (valor médio das␣
 ↪compras), data_ultima_compra.
- Identifique o estado que mais vendeu em termos de receita e o que mais vendeu␣
 ↪em unidades.
"""

# Pegando a quantidade de compras por id
compras = loja.groupby('id_cliente').agg(
    qte_compras=('id_cliente', 'count'),
    valor_total_compras=('valor_compra', 'sum'),
    valor_medio_compras=('valor_compra', 'mean'),
    data_ultima_compra=('data_da_compra', 'max')
).reset_index()

# Estado que mais vendeu em termos de receita
estado_maior_receita = loja.groupby('estado')['valor_compra'].sum().idxmax()
print(f'O estado com maior receita é {estado_maior_receita}')
# Estado com maior venda em unidades
estado_maior_venda = loja.groupby('estado')['valor_compra'].count().idxmax()
print(f'O estado com maior venda em unidades é {estado_maior_venda}')
```

```
O estado com maior receita é MG
O estado com maior venda em unidades é MG
```