Explicando os métodos para tratamento de dados utilizados na tabela do exercício 6. Workshop de Dados dia 11/09/2025.

**Célula 1**

import pandas as pd

df = pd.read\_csv("/content/dados\_nao\_tratados.csv")

display(df.head())

display(df)

Aqui estou apenas importando a biblioteca Panda, utilizada para manipulação de dados. A função pd.read\_csv("/content/dados\_nao\_tratados.csv") serve para carregar o arquivo fornecido. O arquivo CSV (fonte de dados) é lido e a partir disso posso começar as manipulações na tabela. O display (repetido várias vezes durante o código) serve pra mostrara tabela. Na primeira vez que aparece, como display(df.head()), mostra apenas os 5 primeiros elementos.

**Célula 2**

df["idade"] = pd.to\_numeric(df["idade"], errors='coerce')

df["idade"] = df["idade"].astype('Int64')

df["idade"] = df["idade"].fillna(0)

display(df)

A função pd.to\_numeric() é utilizada para tentar converter todos os valores da coluna em números. Valores que não sejam números válidos (textos ou caracteres especiais) são substituídos por NaN (Not a Number), permitindo identificar e tratar esses casos. O parâmetro coerce assegura que os valores sejam convertidos sem gerar erros.

Por conseguinte, o tipo da coluna é alterado para Int64 utilizando o método astype(). O Int64 consegue lidar com valores nulos, o que significa que o DataFrame mantém registros ausentes sem que ocorra perda de informação. Por fim, a função fillna(0) substitui todos os valores Nan por 0, não permitindo a existência de valores ausentes na coluna, o que é útil para análises que necessitem de valores numéricos consistentes. Esse processo assegura que a coluna "idade" tenha dados numéricos válidos e sem lacunas, o que é crucial para análise e modelagem de dados.

**Célula 3**

df["nota"] = pd.to\_numeric(df["nota"], errors='coerce')

df["nota"] = df["nota"].astype('Float64')

df["nota"] = df["nota"].fillna(0)

display(df)

De forma análoga à célula anterior, a função pd.to\_numeric() garante que todos os valores da coluna nota sejam transformados em numéricos, transformandos outros tipos de dados em NaN (Not a Number).

Nesse sentido, o tipo da coluna nota é convertido para Float64, novamente utilizando o método astype(). Esse tipo de dado permite o armazenamento de valores numéricos com casas decimais, bem como suporta valores NaN no caso de registros aussentes.

Ao final, o método fillna(0) substitui os valores NaN por 0, e a função display(df) exibe o DataFrame atualizado.

**Célula 4**

df["ativo"] = df["ativo"].astype(str).str.strip().str.lower().map({

    "sim":True,

    "não":False,

    "true":True,

    "false":False,

    "yes":True,

    "nao":False,

    "e":False, # assumindo que e é um valor inválido e deve ser tratado como falso

    "nan": False, # atribui nulo = falso

    "TRUE":True,

    "FALSE":False

})

display(df)

Aqui, o objetivo é fazer com que as diferentes respostas para ‘ativo’ sejam entendidas como True ou False, ainda que outros tipos de respostas tenham sido enviadas.

Para isso, o método astype(str) garante que todos os valores sejam tratados como texto, independentemente da formatação original (números, booleanos, etc). Em seguida, a função str.strip() remove espaços em branco antes e depois de cada valor, para que não haja problemas com caracteres extras.

Após isso, o método str.lower() é aplicado para transformar todos os valores da coluna para minúsculas, tornando a análise independente de maiúsculas ou minúsculas. A seguir, o método map() é utilizado para mapear valores padronizados para valroes booleanos. Desse modo, valores como “sim”, “yes” e outras variações de escrita ou capitalização são convertidos para True, enquanto valores como “não” ou “false” são atribuídos para False. Importante ressaltar que aqui fiz a escolha de atribuir False para o valor fornecido “E”, que poderia, de outra forma, ser atribuído como nulo.

**Célula 5**

df["data\_inscricao"] = pd.to\_datetime(df["data\_inscricao"], errors='coerce')

df["data\_inscricao"] = df["data\_inscricao"].dt.strftime("%d/%m/%Y")

display(df)

O código dessa célula tem por objetivo limpar e formatar a coluna "data\_inscricao", garantindo que os valores sejam convertidos para um formato de data adequado. O processo é feito em duas etapas principais:

Primeiramente, a função pd.to\_datetime(df["data\_inscricao"], errors='coerce') é utilizada para converter os valores da coluna "data\_inscricao" para o tipo de dado datetime do pandas. Caso algum valor não seja uma data válida ou não possa ser convertido (como texto ou valores errados), ele será substituído por NaN devido ao parâmetro errors='coerce'. Esse parâmetro garante que os valores inválidos sejam tratados sem gerar erro no código.

Em seguida, a coluna convertida para o formato de data é formatada para o padrão dd/mm/aaaa usando o método dt.strftime("%d/%m/%Y"). Essa operação garante que todas as datas na coluna "data\_inscricao" estejam no formato "dia/mês/ano", onde %d representa o dia com dois dígitos, %m o mês com dois dígitos e %Y o ano com quatro dígitos.

**Célula 6**

df=df.fillna(0)

display(df)

O método fillna(0) serve apenas para transformar todos os valores NaN do DataFrame em 0. Nesse caso, foi utilizado aqui apenas para modificar os valores NaN da tabela data\_inscricao, nas demais tabelas esse procedimento foi feito na própria cédula.