# Documentação do Código de Tratamento de Dados

## Visão Geral

Este documento descreve as etapas realizadas no código de tratamento de dados em Python, utilizando a biblioteca Pandas. O objetivo foi corrigir inconsistências e padronizar os dados presentes em uma tabela de usuários.

## 1. Criação do DataFrame

Foi criado um DataFrame a partir de um dicionário contendo as seguintes colunas:

- id: identificador numérico do usuário

- nome: nome do usuário

- idade: valores numéricos e inconsistentes (ex.: 'vinte e três', '?')

- data\_inscricao: datas em múltiplos formatos e com erros

- nota: valores numéricos, nulos e inválidos (ex.: 'dez')

- ativo: diferentes formas de indicar verdadeiro/falso (ex.: 'Sim', 'TRUE', 'nao')

## 2. Etapas de Tratamento

### 2.1 Conversão da Coluna 'idade'

Foi utilizada a função pd.to\_numeric() com errors='coerce' para transformar valores não numéricos em NaN, garantindo que a coluna 'idade' possua apenas valores numéricos.

### 2.2 Conversão da Coluna 'data\_inscricao'

Foi utilizada a função pd.to\_datetime() com errors='coerce' para padronizar datas no formato datetime e invalidar registros incorretos (como '2023-02-30').

### 2.3 Conversão da Coluna 'nota'

A coluna 'nota' foi convertida para valores numéricos, substituindo textos inválidos (como 'dez') por NaN.

### 2.4 Padronização da Coluna 'ativo'

Os valores da coluna 'ativo' foram normalizados para letras minúsculas e mapeados para booleanos (True/False). Por exemplo: 'sim', 'true' e 'yes' foram convertidos para True; enquanto 'não', 'nao' e 'false' foram convertidos para False.

### 2.5 Remoção de Linhas Inválidas

Registros com valores nulos em colunas críticas ('idade', 'data\_inscricao', 'nota') foram removidos para garantir a consistência do conjunto de dados.

### 2.6 Reindexação do DataFrame

Após a remoção de registros inválidos, o índice do DataFrame foi resetado para manter uma sequência contínua.

## 3. Resultados Obtidos

- A tabela final contém apenas registros válidos e consistentes.

- 'idade': somente valores numéricos

- 'data\_inscricao': datas padronizadas em formato datetime

- 'nota': apenas valores numéricos (float)

- 'ativo': representado em valores booleanos (True/False)

- O número de registros foi reduzido devido à exclusão de linhas inválidas.

## 4. Conclusão

O processo garantiu que os dados estivessem padronizados e prontos para análise, seguindo boas práticas de limpeza e tratamento de dados.

# Código de Tratamento de Dados Completo

import pandas as pd  
import numpy as np  
  
# Criar DataFrame a partir dos dados fornecidos  
data = {  
 'id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],  
 'nome': ['Ana', 'Bruno', 'Carla', 'Daniel', 'Érica', 'Francisco', 'Gabriela', 'Henrique', 'Ivana', 'José'],  
 'idade': ['25', 'vinte e três', '30', None, '27', '29', '21', '19', '11', '?'],  
 'data\_inscricao': ['2023-05-12', '2023/13/01', None, '2022-07-15', '2022-08-XX', '15/09/2022',   
 '2023-01-05', '2023-02-30', '2023-03-10', '2022-11-25'],  
 'nota': ['8.5', '9.0', None, '7.2', '8', 'dez', '10', '9.5', None, '6.7'],  
 'ativo': ['Sim', 'TRUE', 'Não', 'sim', 'e', 'FALSE', 'TRUE', 'nao', 'Yes', 'sim']  
}  
  
df = pd.DataFrame(data)  
  
# 1. Remover linhas com valores nulos na coluna 'nome'  
df = df.dropna(subset=['nome'])  
  
# 2. Converter a coluna 'idade' para numérico, tratando valores não numéricos  
idade\_map = {'vinte e três': 23, '?': np.nan}  
df['idade'] = df['idade'].replace(idade\_map)  
df['idade'] = pd.to\_numeric(df['idade'], errors='coerce')  
  
# 3. Converter a coluna 'data\_inscricao' para datetime  
df['data\_inscricao'] = pd.to\_datetime(df['data\_inscricao'], errors='coerce')  
  
# 4. Converter a coluna 'nota' para numérico  
nota\_map = {'dez': 10}  
df['nota'] = df['nota'].replace(nota\_map)  
df['nota'] = pd.to\_numeric(df['nota'], errors='coerce')  
  
# 5. Padronizar a coluna 'ativo' para booleanos  
ativo\_map = {'sim': True,'Sim': True,'TRUE': True,'Yes': True,'não': False,'Não': False,'nao': False,'FALSE': False,'e': np.nan}  
df['ativo'] = df['ativo'].str.strip().str.lower().map(ativo\_map)  
  
# 6. Preencher valores nulos com 0 nas colunas numéricas  
df['idade'] = df['idade'].fillna(0)  
df['nota'] = df['nota'].fillna(0)  
  
# 7. Converter tipos de dados  
df['id'] = df['id'].astype(int)  
df['idade'] = df['idade'].astype(int)  
df['nota'] = df['nota'].astype(float)  
  
# 8. Remover duplicatas  
df = df.drop\_duplicates()  
  
# 9. Renomear colunas para padrão em inglês  
df.rename(columns={'nome':'name','idade':'age','data\_inscricao':'registration\_date','nota':'grade','ativo':'active'}, inplace=True)  
  
# Visualizar resultado  
print("Tabela após tratamento:")  
print(df)  
print("\nInformações do DataFrame:")  
print(df.info())