**Relatório de Padronização de Bases de Dados**

**1. Objetivo do Projeto**

O principal objetivo deste projeto foi automatizar a padronização de múltiplas bases de dados de clientes, que estavam em abas diferentes dentro de um único arquivo Excel. O processo de limpeza e unificação foi focado em dois aspectos principais:

* **Padronização de Clientes:** Utilizando uma base de referência (Base\_Referencia), o script identifica e padroniza os nomes de clientes, atribuindo um ID único, a Razão Social e o Grupo Econômico corretos.

**2. Metodologia de Processamento**

A solução foi desenvolvida em Python, utilizando a biblioteca pandas para manipulação de dados e a biblioteca fuzzywuzzy para a lógica de correspondência de texto. O processo seguiu a seguinte sequência:

1. **Carregamento das Bases de Referência:** O script carrega as abas Base\_Referencia e Fases do arquivo Excel. A aba Fases é lida de uma forma especial (header=None) para criar um dicionário de mapeamento de serviços.
2. **Iteração e Processamento:** O código percorre cada uma das outras abas do arquivo Excel. Para cada aba, ele executa as seguintes etapas:
   * Tratamento dos nomes de clientes, removendo caracteres especiais e espaços em branco.
   * Aplicação de **Fuzzy Matching** para encontrar a correspondência mais próxima entre os nomes brutos dos clientes e a Base\_Referencia, com um limite de similaridade de 85%.
   * Mapeamento dos serviços para suas categorias padronizadas usando o dicionário da aba Fases.
3. **Unificação e Salvamento:** Após processar cada aba individualmente, o script concatena todos os resultados em um único DataFrame e salva o arquivo final no formato Excel (.xlsx), no caminho Data/Processed.

**3. Código Final Consolidado**

Aqui está o script completo e funcional que incorpora todas as etapas e correções feitas durante o desenvolvimento.

Python

import pandas as pd

import os

import re

from fuzzywuzzy import fuzz

# Função padronizar\_clientes para que ela possa ser chamada posteriormente.

def padronizar\_clientes(df, df\_referencia):

"""

Usa Fuzzy Matching para encontrar a correspondência mais próxima

dos nomes de clientes na base de referência.

"""

# Prepara as listas para armazenar os resultados

ids\_padronizados = []

razoes\_sociais = []

grupos\_economicos = []

# Loop para cada cliente no DataFrame

for nome\_cliente in df['Nome\_Cliente\_Tratado']:

melhor\_match = None

melhor\_score = 0

# Procura a melhor correspondência entre o nome do cliente e a lista de nomes de referência

for ref\_nome in df\_referencia['Razão Social']:

# A função fuzz.ratio calcula a similaridade entre duas strings

score = fuzz.ratio(nome\_cliente, ref\_nome)

if score > melhor\_score:

melhor\_score = score

melhor\_match = ref\_nome

# Define um limite de score (ex: 85%) para considerar um match válido

if melhor\_score >= 85:

# Pega as informações do cliente padronizado (ID, Razão Social, Grupo)

match\_info = df\_referencia[df\_referencia['Razão Social'] == melhor\_match].iloc[0]

ids\_padronizados.append(match\_info['ID'])

razoes\_sociais.append(match\_info['Razão Social'])

grupos\_economicos.append(match\_info['Empresa / Grupo Economico'])

else:

# Se o score for baixo, o cliente não é padronizado

ids\_padronizados.append(None)

razoes\_sociais.append(None)

grupos\_economicos.append(None)

# Adiciona as novas colunas padronizadas ao DataFrame

df['ID\_Padronizado'] = ids\_padronizados

df['Razao\_Social\_Padronizada'] = razoes\_sociais

df['Grupo\_Economico\_Padronizado'] = grupos\_economicos

print(f" -> Padronização de clientes concluída.")

# A variável 'path' usa um caminho relativo para a pasta 'Data/Raw'.

path = '../Data/Raw/'

nome\_arquivo\_excel = 'Base de Dados Clientes.xlsx'

caminho\_completo\_excel = os.path.join(path, nome\_arquivo\_excel)

try:

# A classe ExcelFile permite ler todas as abas do arquivo.

excel\_file = pd.ExcelFile(caminho\_completo\_excel)

# Carregando as bases de referência que serão usadas para padronizar os dados.

# A base de referência está na aba 'Base\_Referencia'.

df\_referencia = excel\_file.parse('Base\_Referencia')

# A base de fases está na aba 'Fases'. O argumento 'header=None' é crucial aqui,

# pois a primeira linha da aba 'Fases' não é um cabeçalho de dados.

df\_fases = excel\_file.parse('Fases', header=None)

except FileNotFoundError:

print(f"Erro: Verifique se o arquivo '{nome\_arquivo\_excel}' está na pasta '{path}'.")

exit()

# Criando um dicionário para o mapeamento de serviços para categorias.

# O uso de 'str()' garante que o código não falhe se a célula for um número.

mapeamento\_servicos = {}

for \_, row in df\_fases.iterrows():

if pd.notna(row[1]):

categoria = str(row[0])

servico = str(row[1])

mapeamento\_servicos[servico.upper().strip()] = categoria.strip()

# Preparando a base de referência para a busca.

# Padroniza os nomes para facilitar a busca.

df\_referencia['Razão Social'] = df\_referencia['Razão Social'].astype(str).str.upper().str.strip()

df\_referencia['Empresa / Grupo Economico'] = df\_referencia['Empresa / Grupo Economico'].astype(str).str.strip()

# Lista para armazenar todos os DataFrames tratados antes de unificar.

todos\_os\_dfs = []

# Loop principal para processar cada aba do arquivo Excel.

for nome\_aba in excel\_file.sheet\_names:

# Ignora as abas de referência para processar apenas as de dados de clientes.

if nome\_aba not in ['Base\_Referencia', 'Fases']:

print(f"\nProcessando a base: {nome\_aba}")

try:

# Carrega a base de dados.

df = excel\_file.parse(nome\_aba)

# Adiciona uma coluna para identificar a origem dos dados.

df['Base\_Origem'] = nome\_aba

# Renomeia a coluna de clientes para um padrão consistente.

if 'Nome do Cliente' in df.columns:

df = df.rename(columns={'Nome do Cliente': 'Nome\_Cliente\_Bruto'})

else:

print(f"Aviso: Coluna 'Nome do Cliente' não encontrada na aba '{nome\_aba}'. Pulando a padronização de clientes.")

todos\_os\_dfs.append(df)

continue

# Início do tratamento de nomes de clientes para padronização.

df['Nome\_Cliente\_Bruto'] = df['Nome\_Cliente\_Bruto'].astype(str).str.upper().str.strip()

df['Nome\_Cliente\_Tratado'] = df['Nome\_Cliente\_Bruto'].apply(

lambda x: re.sub(r'[^\w\s]', '', x)

)

# Padronização de serviços (sub-categorias) usando o mapeamento 'Fases'.

if 'Fase 2' in df.columns:

df['Servico\_Tratado'] = df['Fase 2'].astype(str).str.upper().str.strip()

df['Categoria\_Padronizada'] = df['Servico\_Tratado'].map(mapeamento\_servicos)

# Padronização de clientes usando Fuzzy Matching para encontrar a melhor correspondência.

# A chamada para a função está aqui.

padronizar\_clientes(df, df\_referencia)

todos\_os\_dfs.append(df)

except Exception as e:

print(f"Erro ao processar a aba '{nome\_aba}': {e}")

continue

# Execução do script

if \_\_name\_\_ == "\_\_main\_\_":

if todos\_os\_dfs:

# Concatena todos os DataFrames tratados em um único.

df\_unificado = pd.concat(todos\_os\_dfs, ignore\_index=True)

print("\n--- Processo de Unificação Concluído ---")

print(f"Total de registros na base unificada: {len(df\_unificado)}")

print(f"Total de clientes únicos padronizados: {df\_unificado['ID\_Padronizado'].nunique()}")

# Salva o resultado final na pasta 'Data/Processed' no formato Excel.

caminho\_salvar = '../Data/Processed/base\_padronizada.xlsx'

os.makedirs('../Data/Processed', exist\_ok=True)

df\_unificado.to\_excel(caminho\_salvar, index=False)

print(f"Base de dados unificada salva em: {caminho\_salvar}")

else:

print("\nNenhum arquivo de dados de clientes foi encontrado ou processado.")

**4. Arquivo de Saída**

O processo resultou no arquivo base\_padronizada.xlsx, salvo na pasta Data/Processed. Este arquivo contém todas as bases de dados unificadas, com as seguintes colunas adicionadas para a sua documentação e análise:

* **Base\_Origem**: Identifica de qual aba do arquivo original cada linha de dados foi extraída.
* **Nome\_Cliente\_Bruto**: Preserva a nomenclatura original do cliente antes da padronização.
* **Nome\_Cliente\_Tratado**: Versão do nome do cliente tratada para facilitar a busca por correspondência.
* **ID\_Padronizado**: O identificador único do cliente, conforme a Base\_Referencia.
* **Razao\_Social\_Padronizada**: A razão social padronizada, também vinda da Base\_Referencia.
* **Grupo\_Economico\_Padronizado**: O grupo econômico associado ao cliente padronizado.

.