**README - Processamento de DataFrame Banco MCC**

Este código em Python utiliza a biblioteca Tkinter para criar uma interface gráfica que permite o processamento e extração de dados de um DataFrame em formato Excel (.xlsx). O processo inclui várias etapas de manipulação do DataFrame, como renomear colunas, dividir células, repetir valores e muito mais.

**Como Usar**

1. **Carregar DataFrame**:
   * Clique no botão "Carregar DataFrame" para abrir uma janela de seleção de arquivo.
   * Escolha um arquivo Excel (.xlsx) que contenha os dados que deseja processar.
2. **Processamento do DataFrame**:
   * O código lerá o arquivo Excel e aplicará uma série de transformações no DataFrame. Aqui estão algumas das etapas principais realizadas:
     + Descarta as cinco primeiras linhas do DataFrame.
     + Renomeia colunas com nomes que começam com "Total" para evitar duplicatas.
     + Concatena valores em várias colunas em uma única coluna chamada "Total\_Concatenado".
     + Divide valores da coluna "Total\_Concatenado" em linhas separadas.
     + Preenche valores nulos com "N/A".
     + Divide a coluna "Total\_Concatenado" por ', ' e cria uma nova coluna chamada "TOTAL".
     + Define algumas colunas com valores específicos.
3. **Extrair Dados Processados**:
   * Clique no botão "Extrair Dados Processados" para abrir uma janela de seleção de destino.
   * Escolha um local e forneça um nome de arquivo para salvar o DataFrame processado como um arquivo Excel (.xlsx).
4. **Resultado**:
   * O rótulo exibirá uma mensagem informando se o processo foi bem-sucedido ou não.

**Requisitos**

* Certifique-se de que você tenha o Python instalado em seu sistema.
* Você também deve ter as bibliotecas Tkinter, pandas e numpy instaladas em seu ambiente Python. Você pode instalá-las usando pip:
* pip install tkinter pandas numpy

**Observações**

* O código faz uso de uma variável global chamada processed\_df para armazenar o DataFrame processado, que pode ser acessado após a conclusão do processamento.
* Certifique-se de que o arquivo Excel carregado atende aos requisitos esperados pelo código, pois ele realiza uma série de operações específicas no DataFrame. Se o arquivo não estiver em conformidade com o esperado, as operações podem não funcionar corretamente.
* É importante entender que o código foi simplificado para fins de explicação e pode precisar de ajustes para atender a requisitos específicos de entrada de dados.
* A interface gráfica oferece funcionalidade básica para carregar e extrair dados, mas pode ser expandida e aprimorada de acordo com suas necessidades.

Certifique-se de entender o funcionamento do código e personalize-o de acordo com os requisitos do seu projeto.

**Explicação do código mais detalhada**

python

import tkinter as tk

from tkinter import filedialog

import pandas as pd

import numpy as np

# Variável global para armazenar o DataFrame processado

processed\_df = None

* O código começa importando as bibliotecas necessárias: tkinter para criar a interface gráfica, filedialog para lidar com caixas de diálogo de arquivo, pandas para manipular os DataFrames e numpy para operações matemáticas.

python

def process\_dataframe():

global processed\_df

file\_path = filedialog.askopenfilename(filetypes=[("Excel files", "\*.xlsx")])

if file\_path:

df = pd.read\_excel(file\_path)

# Descarte as cinco primeiras linhas

df = df.iloc[2:]

# ...

# Resto da função de processamento do DataFrame

# ...

* A função process\_dataframe é chamada quando o botão "Carregar DataFrame" é pressionado. Ela permite ao usuário selecionar um arquivo Excel (.xlsx) usando uma caixa de diálogo de arquivo.
* Em seguida, lê o arquivo Excel em um DataFrame df e descarta as cinco primeiras linhas do DataFrame.

python

def mover\_colunas\_primeira\_linha(df):

# ...

# Função para mover as colunas da primeira linha e descartar a primeira linha

# ...

* A função mover\_colunas\_primeira\_linha é usada para renomear as colunas do DataFrame com base nos valores da primeira linha. Ela também descarta a primeira linha do DataFrame.

python

def remove\_colunas\_com\_x(dataframe):

# ...

# Função para remover colunas com nomes terminando em 'x'

# ...

* A função remove\_colunas\_com\_x remove as colunas do DataFrame cujos nomes terminam em 'x'.

python

def renomear\_colunas\_com\_total(dataframe):

# ...

# Função para renomear colunas com nomes que começam com 'Total'

# ...

* A função renomear\_colunas\_com\_total renomeia as colunas do DataFrame que começam com 'Total' para evitar duplicatas.

python

def split\_and\_concat\_inplace(dataframe, column\_name):

# ...

# Função para dividir e concatenar valores de uma coluna em várias linhas

# ...

* A função split\_and\_concat\_inplace divide os valores da coluna especificada e os concatena em várias linhas do DataFrame.

python

# Chame a função para mover as colunas da primeira linha

df = mover\_colunas\_primeira\_linha(df)

# Remover a primeira linha

df = df.iloc[1:]

* Aqui, as funções mover\_colunas\_primeira\_linha e a remoção da primeira linha são chamadas para continuar o processamento do DataFrame.

python

# Converter a coluna 'Cod Produto' em string

df['CÓDIGO'] = df['CÓDIGO'].astype(str)

* A coluna 'CÓDIGO' é convertida em uma string.

python

# Arredondar e converter a coluna 'TAXA' em string formatada

df['TAXA'] = df['TAXA'].apply(lambda x: str(round(float(x), 2))

* A coluna 'TAXA' é arredondada e convertida em uma string formatada com duas casas decimais.

python

# Repetir todas as colunas, exceto "Total\_Concatenado", 7 vezes

cols = df.columns.tolist()

cols.remove("Total\_Concatenado")

df = df[cols]

* Todas as colunas do DataFrame, exceto "Total\_Concatenado," são repetidas sete vezes.

python

# Dividir a string em valores individuais e converter em porcentagem

df['TAXA'] = df['TAXA'].apply(lambda x: f"{float(x) \* 100:.0f}%")

* Os valores na coluna 'TAXA' são multiplicados por 100 e formatados como porcentagem.

python

#Criar a nova coluna com a concatenação

df['Tabela/Nome do Produto'] = df['CÓDIGO'] + '-' + df['TIPO TAXA'] + '-' + df['TAXA']

* Uma nova coluna chamada 'Tabela/Nome do Produto' é criada, concatenando os valores de três colunas.

python

# Crie um DataFrame vazio com as mesmas colunas do seu DataFrame atual

linhas\_em\_branco = pd.DataFrame(columns=df.columns)

# Repita o processo para adicionar 1000 linhas em branco

for \_ in range(500):

linhas\_em\_branco = linhas\_em\_branco.append(pd.Series([None] \* len(df.columns), index=df.columns), ignore\_index=True)

* O código cria um DataFrame vazio chamado linhas\_em\_branco com as mesmas colunas do DataFrame atual e adiciona 500 linhas em branco.

python

# Concatene o DataFrame original com as linhas em branco

df = pd.concat([df, linhas\_em\_branco], ignore\_index=True)

df = df.reset\_index(drop=True)

* O DataFrame original é concatenado com as linhas em branco, resultando em um DataFrame mais longo.

python

# Coluna que você deseja repetir

coluna\_a\_repetir = 'Tabela/Nome do Produto'

# Número de repetições desejadas

n\_repeticoes = 7

# Repetir a coluna e criar um novo DataFrame

df\_repetido = pd.DataFrame({coluna\_a\_repetir: [item for item in df[coluna\_a\_repetir] for \_ in range(n\_repeticoes)]})

df\_repetido = df\_repetido.reset\_index(drop=True)

* A coluna 'Tabela/Nome do Produto' é repetida sete vezes e armazenada em um novo DataFrame chamado df\_repetido.

python

# Se você já tem um DataFrame com os índices indesejados

df = df.reset\_index(drop=True)

df['Tabela/Nome do Produto'] = df\_repetido['Tabela/Nome do Produto']

* O código ajusta os índices dos DataFrames e substitui a coluna original pela coluna repetida.

As etapas subsequentes no código continuam o processo de processamento do DataFrame, incluindo a divisão de valores em várias linhas e a manipulação de valores específicos de colunas.

A interface gráfica criada usando Tkinter ofere duas funcionalidades principais:

1. **Carregar DataFrame**: Permite ao usuário carregar um arquivo Excel contendo os dados a serem processados.
2. **Extrair Dados Processados**: Permite ao usuário salvar o DataFrame processado em um novo arquivo Excel.

A variável processed\_df é usada para armazenar o DataFrame após o processamento, e os resultados são exibidos em uma caixa de mensagem na interface gráfica.