```
import pandas as pd
import os
import matplotlib.pyplot as plt
import webbrowser
import datetime

def obter_nome_usuario():
    """Solicita o nome do usuário e retorna-o após validação básica."""
    while True:
        nome = input("Por favor, digite seu nome (ao menos 3 caracteres):
").strip()
        nomes = nome.split()
        if len(nomes) >= 2 and all(len(n) >= 1 and n.isalpha() for n in nomes):
            return nome
        elif len(nome) >= 3 and nome.isalpha():
            return nome
        else:
            print("Nome inválido. Por favor, digite um nome com ao menos 3
caracteres alfabéticos.")

def registrar_acao(nome_usuario, acao):
    """Registra a ação do usuário em um arquivo de log."""
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    log_mensagem = f"[{timestamp}] Usuário: {nome_usuario} - Ação: {acao}\n"
    with open("registro_acoes.log", "a") as arquivo_log:
        arquivo_log.write(log_mensagem)

def carregar_dados():
    """Carrega dados de um arquivo CSV ou JSON para um DataFrame do Pandas.

    Pergunta ao usuário o caminho do arquivo e tenta carregá-lo.
    Suporta arquivos nos formatos CSV e JSON.

    Returns
    -------
    pandas.DataFrame or None
        Um DataFrame contendo os dados carregados,
        ou None se ocorrer algum erro.
    """
    while True:
        caminho_arquivo = input("Por favor, digite o caminho do arquivo (CSV ou
JSON): ")
        registrar_acao(nome_usuario, f"Tentou carregar arquivo:
{caminho_arquivo}")
        if not os.path.exists(caminho_arquivo):
            print("Erro: O caminho do arquivo especificado não existe.")
            registrar_acao(nome_usuario, f"Erro ao carregar: arquivo não
encontrado.")
            continue
        try:
            if caminho_arquivo.lower().endswith('.csv'):
                df = pd.read_csv(caminho_arquivo)
                print("Arquivo CSV carregado com sucesso!")
                registrar_acao(nome_usuario, f"Arquivo CSV '{caminho_arquivo}'
```

```
carregado com sucesso.")
                return df
            elif caminho_arquivo.lower().endswith('.json'):
                df = pd.read_json(caminho_arquivo)
                print("Arquivo JSON carregado com sucesso!")
                registrar_acao(nome_usuario, f"Arquivo JSON '{caminho_arquivo}'
carregado com sucesso.")
                return df
            else:
                print("Erro: Formato de arquivo não suportado. Por favor, use um
arquivo CSV ou JSON.")
                registrar_acao(nome_usuario, f"Erro ao carregar: formato de
arquivo não suportado.")
                continue
        except pd.errors.EmptyDataError:
            print("Erro: O arquivo está vazio.")
            registrar_acao(nome_usuario, f"Erro ao carregar: arquivo vazio.")
        except pd.errors.ParserError:
            print("Erro: Falha ao analisar o arquivo. Verifique se o formato
está correto.")
            registrar_acao(nome_usuario, f"Erro ao carregar: falha na análise do
arquivo.")
        except Exception as e:
            print(f"Ocorreu um erro inesperado: {e}")
            registrar_acao(nome_usuario, f"Erro ao carregar: erro inesperado -
{e}")
        return None

def analisar_dados_basico(df):
    """Realiza uma análise básica do DataFrame fornecido.

    Exibe a quantidade total de dados, a distribuição por gênero
    e a quantidade de registros sem informação sobre a educação dos pais.

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame a ser analisado.
    """
    if df is not None:
        total_registros = len(df)
        quantidade_generos = df['Gender'].value_counts()
        registros_sem_educacao_pais =
df['Parent_Education_Level'].isnull().sum()

        print("\n--- Análise Básica dos Dados ---")
        print(f"Quantidade total de registros carregados: {total_registros}")
        print("\nDistribuição por gênero:")
        print(quantidade_generos)
        print(f"\nQuantidade de registros sem informação sobre a educação dos
pais (Parent_Education_Level): {registros_sem_educacao_pais}")
        registrar_acao(nome_usuario, "Realizou análise básica dos dados.")
    else:
        print("Erro: Nenhum dado carregado para analisar.")
```

```
        registrar_acao(nome_usuario, "Tentou realizar análise básica, mas não há
dados carregados.")

def limpar_dados(df):
    """Realiza a limpeza dos dados do DataFrame.

    Remove registros com valores nulos na coluna 'Parent_Education_Level'
    e preenche valores nulos na coluna 'Attendance (%)' com a mediana.

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame a ser limpo.

    Returns
    -------
    pandas.DataFrame or None
        O DataFrame com os dados limpos,
        ou None se nenhum dado foi carregado.
    """
    if df is not None:
        registros_antes = len(df)
        df_limpo = df.dropna(subset=['Parent_Education_Level']).copy()
        registros_removidos_educacao = registros_antes - len(df_limpo)
        print(f"\nRegistros removidos devido à falta de informação na educação
dos pais: {registros_removidos_educacao}")
        registrar_acao(nome_usuario, f"Limpou dados: removeu
{registros_removidos_educacao} registros (educação dos pais).")

        mediana_attendance = df_limpo['Attendance (%)'].median()
        nulos_attendance_antes = df_limpo['Attendance (%)'].isnull().sum()
        df_limpo['Attendance (%)'] = df_limpo['Attendance
(%)'].fillna(mediana_attendance)
        nulos_attendance_depois = df_limpo['Attendance (%)'].isnull().sum()
        print(f"Valores nulos na coluna 'Attendance (%)' preenchidos com a
mediana: {mediana_attendance:.2f}%")
        registrar_acao(nome_usuario, f"Limpou dados: preencheu
{nulos_attendance_antes - nulos_attendance_depois} valores nulos em 'Attendance
(%)' com a mediana.")

        soma_attendance = df_limpo['Attendance (%)'].sum()
        print(f"Somatório da coluna 'Attendance (%)': {soma_attendance:.2f}%")
        registrar_acao(nome_usuario, f"Limpou dados: calculou o somatório de
'Attendance (%)'.")

        return df_limpo
    else:
        print("Erro: Nenhum dado carregado para limpar.")
        registrar_acao(nome_usuario, "Tentou limpar dados, mas não há dados
carregados.")
        return None

def consultar_dados_coluna(df):
    """Permite ao usuário consultar estatísticas de uma coluna numérica
```

```
específica do DataFrame
    selecionando por número. Reexibe as opções em caso de erro.

    Lista as colunas numéricas disponíveis com um número correspondente e
solicita
    ao usuário o número da coluna para análise. Calcula e exibe as estatísticas.

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame para consulta.
    """
    if df is not None:
        colunas_numericas = [col for col in df.columns if
pd.api.types.is_numeric_dtype(df[col])]
        if not colunas_numericas:
            print("\nNão há colunas numéricas disponíveis para análise.")
            registrar_acao(nome_usuario, "Tentou consultar dados por coluna, mas
não há colunas numéricas.")
            return

        while True:
            print("\n--- Colunas numéricas disponíveis para análise ---")
            for i, col in enumerate(colunas_numericas):
                print(f"{i + 1}. {col}")
            print("0. Sair")

            opcao = input("\nDigite o número da coluna para análise: ")
            registrar_acao(nome_usuario, f"Selecionou opção '{opcao}' para
consultar dados por coluna.")
            if opcao == '0':
                break
            try:
                indice_coluna = int(opcao) - 1
                if 0 <= indice_coluna < len(colunas_numericas):
                    nome_coluna = colunas_numericas[indice_coluna]
                    media = df[nome_coluna].mean()
                    mediana = df[nome_coluna].median()
                    moda = df[nome_coluna].mode().tolist()
                    desvio_padrao = df[nome_coluna].std()
                    print(f"\n--- Estatísticas da coluna '{nome_coluna}' ---")
                    print(f"Média: {media:.2f}")
                    print(f"Mediana: {mediana:.2f}")
                    print(f"Moda: {moda}")
                    print(f"Desvio Padrão: {desvio_padrao:.2f}")
                    registrar_acao(nome_usuario, f"Consultou estatísticas da
coluna '{nome_coluna}'. Média: {media:.2f}, Mediana: {mediana:.2f}, Moda:
{moda}, Desvio Padrão: {desvio_padrao:.2f}")
                else:
                    print("Erro: Opção inválida. Por favor, digite um número da
lista ou 0 para sair.")
            except ValueError:
                print("Erro: Por favor, digite um número inteiro.")
    else:
```

```
        print("Erro: Nenhum dado carregado para realizar a consulta.")

def gerar_grafico_dispersao(df):
    """Gera um gráfico de dispersão entre as horas de sono e a nota final.

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame contendo os dados.
    """
    if df is not None and 'Sleep_Hours_per_Night' in df.columns and
'Final_Score' in df.columns:
        plt.figure(figsize=(10, 6))
        plt.scatter(df['Sleep_Hours_per_Night'], df['Final_Score'])
        plt.title('Gráfico de Dispersão: Horas de Sono vs. Nota Final')
        plt.xlabel('Horas de Sono por Noite')
        plt.ylabel('Nota Final')
        plt.grid(True)
        plt.show()
        registrar_acao(nome_usuario, "Gerou gráfico de dispersão (Horas de Sono
vs. Nota Final).")
    else:
        print("Erro: Colunas 'Sleep_Hours_per_Night' ou 'Final_Score' não
encontradas para o gráfico de dispersão.")
        registrar_acao(nome_usuario, "Erro ao gerar gráfico de dispersão:
colunas não encontradas.")

def gerar_grafico_barras_idade_media_nota(df):
    """Gera um gráfico de barras da idade versus a média das notas
intermediárias (midterm_Score).

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame contendo os dados.
    """
    if df is not None and 'Age' in df.columns and 'Midterm_Score' in df.columns:
        media_notas_por_idade =
df.groupby('Age')['Midterm_Score'].mean().sort_index()
        plt.figure(figsize=(10, 6))
        plt.bar(media_notas_por_idade.index, media_notas_por_idade.values)
        plt.title('Gráfico de Barras: Idade vs. Média das Notas Intermediárias')
        plt.xlabel('Idade')
        plt.ylabel('Média da Nota Intermediária')
        plt.xticks(media_notas_por_idade.index)
        plt.grid(axis='y', linestyle='--')
        plt.show()
        registrar_acao(nome_usuario, "Gerou gráfico de barras (Idade vs. Média
das Notas Intermediárias).")
    else:
        print("Erro: Colunas 'Age' ou 'Midterm_Score' não encontradas para o
gráfico de barras.")
        registrar_acao(nome_usuario, "Erro ao gerar gráfico de barras: colunas
não encontradas.")
```

```
def gerar_grafico_pizza_idades(df):
    """Gera um gráfico de pizza para a distribuição das idades em grupos.

    Grupos de idade: até 17; 18 a 21; 22 a 24; 25 ou mais.

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame contendo os dados.
    """
    if df is not None and 'Age' in df.columns:
        bins = [0, 18, 22, 25, float('inf')]
        labels = ['Até 17', '18 a 21', '22 a 24', '25 ou mais']
        df['Grupo_Idade'] = pd.cut(df['Age'], bins=bins, labels=labels,
right=False)
        distribuicao_idades = df['Grupo_Idade'].value_counts()
        plt.figure(figsize=(8, 8))
        plt.pie(distribuicao_idades, labels=distribuicao_idades.index,
autopct='%1.1f%%', startangle=140)
        plt.title('Gráfico de Pizza: Distribuição das Idades')
        plt.axis('equal')
        plt.show()
        registrar_acao(nome_usuario, "Gerou gráfico de pizza (Distribuição das
Idades).")
    else:
        print("Erro: Coluna 'Age' não encontrada para o gráfico de pizza.")
        registrar_acao(nome_usuario, "Erro ao gerar gráfico de pizza: coluna não
encontrada.")

def gerar_graficos(df):
    """Apresenta um menu para o usuário escolher qual gráfico gerar.

    Parameters
    ----------
    df : pandas.DataFrame
        O DataFrame contendo os dados.
    """
    if df is not None:
        while True:
            print("\n--- Opções de Gráficos ---")
            print("1. Gráfico de Dispersão: Horas de Sono vs. Nota Final")
            print("2. Gráfico de Barras: Idade vs. Média das Notas
Intermediárias")
            print("3. Gráfico de Pizza: Distribuição das Idades")
            print("0. Voltar ao menu principal")

            opcao = input("Digite o número do gráfico desejado: ")
            registrar_acao(nome_usuario, f"Menu de gráficos: selecionou opção
'{opcao}'.")

            if opcao == '1':
                gerar_grafico_dispersao(df)
            elif opcao == '2':
```

```
                gerar_grafico_barras_idade_media_nota(df)
            elif opcao == '3':
                gerar_grafico_pizza_idades(df)
            elif opcao == '0':
                return  # Retorna ao menu principal
            else:
                print("Opção inválida. Por favor, digite um número da lista.")
    else:
        print("Erro: Nenhum dado carregado para gerar gráficos.")

def abrir_documentacao_html():
    """Abre a página index.html da documentação no navegador Chrome."""
    caminho_documentacao = os.path.abspath("_build/index.html")
    # Registra o Chrome - ajuste o caminho se necessário
    webbrowser.register('chrome', None, webbrowser.BackgroundBrowser("C:/Program
Files/Google/Chrome/Application/chrome.exe"))
    webbrowser.get('chrome').open_new_tab(caminho_documentacao)
    print(f"\nAbrindo documentação em: {caminho_documentacao}")
    registrar_acao(nome_usuario, "Abriu a página de documentação.")

def visualizar_logs():
    """Exibe o conteúdo do arquivo de log."""
    try:
        with open("registro_acoes.log", "r") as arquivo_log:
            conteudo_log = arquivo_log.read()
            print("\n--- Conteúdo do Arquivo de Log ---")
            print(conteudo_log)
    except FileNotFoundError:
        print("\nErro: O arquivo de log 'registro_acoes.log' não foi
encontrado.")
    except Exception as e:
        print(f"\nOcorreu um erro ao ler o arquivo de log: {e}")

if __name__ == '__main__':
    nome_usuario = obter_nome_usuario()
    registrar_acao(nome_usuario, "Iniciou o programa.")
    dados = carregar_dados()
    if dados is not None:
        print("\nPrimeiras linhas dos dados carregados:")
        print(dados.head())
        analisar_dados_basico(dados)  # Exibe a análise básica UMA VEZ
        dados_limpos = limpar_dados(dados.copy()) # Limpa os dados UMA VEZ
        if dados_limpos is not None:
            print("\nPrimeiras linhas dos dados limpos:")
            print(dados_limpos.head())

            # Exibe a distribuição de gênero APÓS a limpeza dos dados
            distribuicao_genero_limpo = dados_limpos['Gender'].value_counts()
            print("\n--- Distribuição de gênero após a limpeza dos dados ---")
            print(distribuicao_genero_limpo)

            while True:
                print("\n--- Menu Principal ---")
                print("1. Consultar dados por coluna")
```

```
                print("2. Gerar gráficos")
                print("3. Abrir documentação")
                print("4. Visualizar Logs")
                print("0. Encerrar o programa")

                opcao_principal = input("Digite o número da opção desejada: ")
                registrar_acao(nome_usuario, f"Menu principal:
{opcao_principal}")

                if opcao_principal == '1':
                    consultar_dados_coluna(dados_limpos)
                elif opcao_principal == '2':
                    gerar_graficos(dados_limpos)
                elif opcao_principal == '3':
                    abrir_documentacao_html()
                elif opcao_principal == '4':
                    visualizar_logs()
                elif opcao_principal == '0':
                    print("Encerrando o programa. Até a próxima!")
                    registrar_acao(nome_usuario, "Encerrou o programa.")
                    break
                else:
                    print("Opção inválida. Por favor, digite um número da
lista.")
```