Filtrando os Dados

```
import pandas as pd
import unicodedata
import os

COLUNAS_DESEJADAS = [
    "data",
    "temp._ins._(c)",
    "temp._max._(c)",
    "temp._min._(c)",
    "temperatura_do_ar___bulbo_seco,_horaria_(c)",
    "temperatura_do_ponto_de_orvalho_(c)",
    "temperatura_maxima_na_hora_ant._(aut)_(c)",
    "temperatura_minima_na_hora_ant._(aut)_(c)",
    "temperatura_orvalho_max._na_hora_ant._(aut)_(c)",
    "temperatura_orvalho_min._na_hora_ant._(aut)_(c)",
    "consumo_kw"
]

def normalizar_colunas(df: pd.DataFrame) -> pd.DataFrame:
    """Normaliza os nomes das colunas (remove acentos, espaços e caracteres especiais)."""
    df.columns = [
        unicodedata.normalize('NFKD', str(c))
        .encode('ascii', 'ignore')
        .decode('ascii')
        .strip()
        .replace(" ", "_")
        .replace("/", "_")
        .replace("-", "_")
        .lower()
        for c in df.columns
    ]
    return df


def converter_coluna_data(df: pd.DataFrame) -> pd.DataFrame:
    """
    Tenta converter a coluna 'data' em datetime, lidando com vários formatos:
    - YYYYMMDD
    - YYYY/MM/DD
    - DD/MM/YYYY
    """
    if 'data' not in df.columns:
        return df
```

```
    # Remove espaços e converte tudo para string
    df['data'] = df['data'].astype(str).str.strip()

    # Detecta datas no formato numérico contínuo (ex: 20110101)
    mask_numeric = df['data'].str.match(r'^\d{8}$')
    if mask_numeric.any():
        df.loc[mask_numeric, 'data'] = pd.to_datetime(df.loc[mask_numeric, 'data'],
format='%Y%m%d', errors='coerce')

    # Tenta os outros formatos comuns
    df['data'] = pd.to_datetime(df['data'], errors='coerce', dayfirst=True)

    # Remove linhas com datas inválidas
    df = df.dropna(subset=['data'])

    # Padroniza para formato YYYY-MM-DD
    df['data'] = df['data'].dt.strftime('%Y-%m-%d')

    return df


def filtrar_colunas(csv_path: str, output_path: str = None, ano_minimo: int = 2024) ->
pd.DataFrame:
    """
    Carrega CSV, mantém apenas as colunas desejadas, filtra por ano mínimo e salva
resultado.

    @param {str} csv_path - Caminho do arquivo CSV original
    @param {str} output_path - Caminho de saída opcional
    @param {int} ano_minimo - Linhas com ano menor que este serão removidas
    @return {pd.DataFrame} DataFrame filtrado
    """
    df = pd.read_csv(csv_path, encoding="utf-8")
    df = normalizar_colunas(df)

    # Mantém apenas colunas relevantes
    colunas_para_manter = [c for c in COLUNAS_DESEJADAS if c in df.columns]
    df_filtrado = df[colunas_para_manter]

    # Converte e padroniza coluna de data
    df_filtrado = converter_coluna_data(df_filtrado)

    #  🔹 Remove linhas com ano anterior ao mínimo
    if 'data' in df_filtrado.columns:
        df_filtrado = df_filtrado[pd.to_datetime(df_filtrado['data']).dt.year >= ano_minimo]

    # Define o nome de saída
    if output_path is None:
```

```
        base, ext = os.path.splitext(os.path.basename(csv_path))
        output_path = f"./dados filtrados/{base}_filtrado{ext}"

    # Cria diretório de saída se não existir
    os.makedirs(os.path.dirname(output_path), exist_ok=True)

    # Salva CSV filtrado
    df_filtrado.to_csv(output_path, index=False, encoding="utf-8")
    print(f"[OK] CSV filtrado salvo em {output_path} | Colunas: {len(colunas_para_manter)} |
Linhas: {len(df_filtrado)}")

    return df_filtrado


if __name__ == "__main__":
    filtrar_colunas("./dados/anuario estatistico de energia eletrica.csv")
    filtrar_colunas("./dados/INMET_BRASILIA_01-01-2024_A_31-12-2024.csv")
    filtrar_colunas("./dados/INMP 15102025-15102025.csv")
```