SESSÃO DE SCRIPTS PYTHON

```
#Importações

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from pandas_profiling import ProfileReport
import sweetviz as sv
import seaborn as sns

#Carregamento_de_dados
df = pd.read_csv("dados.csv")

#Profiling
profile = ProfileReport(df, title="Report")
display(profile)

#Sweetviz
analyze_report = sv.analyze(df)
analyze_report.show_html('report.html', open_browser=True)


#Correlação
plt.figure(figsize=(15, 10))
sns.heatmap(df.corr(), annot=True, cmap='Greens')


#Funções para a análise dos outliers
def excluir_outliers(df, nome_coluna):
    qtde_linhas = df.shape[0]
    lim_inf, lim_sup = limites(df[nome_coluna])
    df = df.loc[(df[nome_coluna] >= lim_inf) & (df[nome_coluna] <=
lim_sup), :]
    linhas_removidas = qtde_linhas - df.shape[0]
    return df,  linhas_removidas

def limites(coluna):
    q1 = coluna.quantile(0.25)
    q3 = coluna.quantile(0.75)
    amplitude = q3 - q1
    return q1 - 1.5 * amplitude, q3 + 1.5 * amplitude

def diagrama_caixa(coluna):
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(15, 5)
    sns.boxplot(x=coluna, ax=ax1)
    ax2.set_xlim(limites(coluna))
    sns.boxplot(x=coluna, ax=ax2)

def histograma(coluna):
    plt.figure(figsize=(15, 5))
    sns.distplot(coluna, hist=True)

def grafico_barra(df,index, coluna):
    plt.figure(figsize=(15, 5))
```

```
    df = df.groupby(index).sum()[coluna].reset_index()
    ax = sns.barplot(x=df[index], y=df[coluna])


# quantidade
diagrama_caixa(df['quantidade'])
histograma(df['quantidade'])
grafico_barra(df,'comércio','quantidade')


# valor calórico
diagrama_caixa(df['valor calórico'])
histograma(df['valor calórico'])
grafico_barra(df,'comércio','valor calórico')


#Dias restantes
diagrama_caixa(df['dias restantes'])
histograma(df['dias restantes'])
grafico_barra(df,'comércio','dias restantes')


#Tipo de alimento
plt.figure(figsize=(15, 5))
grafico = sns.countplot('tipo de alimento', data=df)
grafico.tick_params(axis='x', rotation=90)

#Maiores fornecedores
plt.figure(figsize=(15, 5))
grafico = sns.countplot('comércio', data=df)
grafico.tick_params(axis='x', rotation=90)

#Maiores alimento
plt.figure(figsize=(15, 5))
grafico = sns.countplot('alimento', data=df)
grafico.tick_params(axis='x', rotation=90)


#Análise Exploratória de Dados
df_describe = df.describe().round(2)
display(df_describe)

#Data da doação
#Mes
df_datas = df.copy()
df_datas['data da doação'] = pd.to_datetime(df_datas['data da
doação'])
df_datas = df_datas.sort_values("data da
doação",ascending=True).reset_index(drop=True)
df_datas = df_datas.groupby(["data da
doação"]).sum()["quantidade"].to_frame().reset_index()
df_datas['data da doação']  = df_datas['data da doação'].apply(lambda
x: x.strftime("%m/%Y"))

#ano
df_datas_y = df.copy()
df_datas_y['data da doação'] = pd.to_datetime(df_datas_y['data da
doação'])
```

```
df_datas_y = df_datas_y.sort_values("data da
doação",ascending=True).reset_index(drop=True)
df_datas_y = df_datas_y.groupby(["data da
doação"]).sum()["quantidade"].to_frame().reset_index()
df_datas_y['data da doação']  = df_datas_y['data da
doação'].apply(lambda x: x.strftime("%Y"))

fig, (ax1, ax2) = plt.subplots(2,figsize=(15, 7))
ax1.bar(df_datas['data da doação'], df_datas["quantidade"], color
='maroon',width = 0.4)
ax1.tick_params(axis="x", labelsize=12,labelrotation=90)
ax1.tick_params(axis="y", labelsize=12)
ax1.set_ylabel("Quantidade")
ax1.title.set_text("Quantidade doada por mês e ano")
fig.tight_layout()

ax2.bar(df_datas_y['data da doação'], df_datas_y["quantidade"], color
='maroon',width = 0.4)
ax2.set_xlabel("Período da Doação")
ax2.tick_params(axis="x", labelsize=12)
ax2.tick_params(axis="y", labelsize=12)
ax2.set_ylabel("Quantidade")


#Data da validade

#Mes
df_datas = df.copy()
df_datas['data de validade'] = pd.to_datetime(df_datas['data de
validade'])
df_datas = df_datas.sort_values("data de
validade",ascending=True).reset_index(drop=True)
df_datas = df_datas.groupby(["data de
validade"]).sum()["quantidade"].to_frame().reset_index()
df_datas['data de validade']  = df_datas['data de
validade'].apply(lambda x: x.strftime("%m/%Y"))

#ano
df_datas_y = df.copy()
df_datas_y['data de validade'] = pd.to_datetime(df_datas_y['data de
validade'])
df_datas_y = df_datas_y.sort_values("data de
validade",ascending=True).reset_index(drop=True)
df_datas_y = df_datas_y.groupby(["data de
validade"]).sum()["quantidade"].to_frame().reset_index()
df_datas_y['data de validade']  = df_datas_y['data de
validade'].apply(lambda x: x.strftime("%Y"))

fig, (ax1, ax2) = plt.subplots(2,figsize=(15, 7))
ax1.bar(df_datas['data de validade'], df_datas["quantidade"], color
='maroon',width = 0.4)
ax1.tick_params(axis="x", labelsize=12,labelrotation=90)
ax1.tick_params(axis="y", labelsize=12)
ax1.set_ylabel("Quantidade")
ax1.title.set_text("Quantidade por prazo de validade por mês e ano")
fig.tight_layout()
```

```
ax2.bar(df_datas_y['data de validade'], df_datas_y["quantidade"],
color ='maroon',width = 0.4)
ax2.set_xlabel("Período da validade")
ax2.tick_params(axis="x", labelsize=12)
ax2.tick_params(axis="y", labelsize=12)
ax2.set_ylabel("Quantidade")
```