```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import seaborn as sns
import scipy
# Definir o estilo globalmente
plt.style.use('seaborn')
```

Feature : Description

Campaign 1: Accepted if the customer accepted the offer in the 1st campaing, Rejected otherwise;

Campaign 2: Accepted if the customer accepted the offer in the 2nd campaing, Rejected otherwise;

Campaign 3: Accepted if the customer accepted the offer in the 3rd campaing, Rejected otherwise;

Campaign 4: Accepted if the customer accepted the offer in the 4th campaing, Rejected otherwise;

Campaign 5: Accepted if the customer accepted the offer in the 5th campaing, Rejected otherwise;

Campaign 6 : Accepted if the customer accepted the offer in the 6th campaing, Rejected otherwise;

Complain : 1 if the customer complained in the last 2 years;

Customer_Days : Days since customer enrollment with the company;

Education : customer's level of education;

Marital : customer's marital status;

Kidhome : number of small children in customer's household;

Teenhome : number of teenagers children in customer's household;

Income : customer's yearly household income;

MntFishProducts : amount spent on fish products in the last 2 years;

MntMeatProducts : amount spent on meat products in the last 2 years;

MntFruit : amount spent on fruit products in the last 2 years;

MntSweetProducts : amount spent on sweet products in the last 2 years;

MntWines : amount spent on wines products in the last 2 years;

MntGoldProds : amount spent on gold products in the last 2 years;

NumDealsPurchases : number of purchases made with discount;

NumCatalogPurchases : number of purchases made using catalogue;

NumStorePurchases : number of purchases made drectly in stores;

NumWebPurchases : number of purchases made through company's web site;

NumWebVisitsMonth : number of visits to company's web site in the last month;

Recency : number of days since the last purchase;

```
df=pd.read_csv(r"/content/partialdf_ifood.csv")
df
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGold
0	58138.0	0	0	58	635	88	546	172	88	
1	46344.0	1	1	38	11	1	6	2	1	
2	71613.0	0	0	26	426	49	127	111	21	
3	26646.0	1	0	26	11	4	20	10	3	
4	58293.0	1	0	94	173	43	118	46	27	
...	...	...	...	...	...	...	...	...	...	
2200	61223.0	0	1	46	709	43	182	42	118	
2201	64014.0	2	1	56	406	0	30	0	0	
2202	56981.0	0	0	91	908	48	217	32	12	
2203	69245.0	0	1	8	428	30	214	80	30	
2204	52869.0	1	1	40	84	3	61	2	1	

2205 rows × 29 columns

```
colors = ['#FEA500', '#EA0031', '#8A011B', '#FF94C2']
```

```
df['maritalStatus'] = df['maritalStatus'].str.replace('marital_','')
```

```
df['educationLevel'] = df['educationLevel'].str.replace('education_','')
```

Data transformation

```
##original df transformations:

df.replace({
    'AcceptedCmp1': {1: 'Accepted', 0: 'Declined'},
    'AcceptedCmp2': {1: 'Accepted', 0: 'Declined'},
    'AcceptedCmp3': {1: 'Accepted', 0: 'Declined'},
    'AcceptedCmp4': {1: 'Accepted', 0: 'Declined'},
    'AcceptedCmp5': {1: 'Accepted', 0: 'Declined'},
    'Response'    : {1: 'Accepted', 0: 'Declined'},
    'Complain'    : {1: 'True', 0: 'False'}
    },inplace=True)

df.rename(columns={
    'AcceptedCmp1': 'Campaign 1',
    'AcceptedCmp2':'Campaign 2',
    'AcceptedCmp3':'Campaign 3',
    'AcceptedCmp4':'Campaign 4',
    'AcceptedCmp5':'Campaign 5',
    'Response':'Campaign 6',
    },inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Income               2205 non-null   float64
 1   Kidhome              2205 non-null   int64
 2   Teenhome             2205 non-null   int64
 3   Recency              2205 non-null   int64
 4   MntWines             2205 non-null   int64
 5   MntFruits            2205 non-null   int64
 6   MntMeatProducts      2205 non-null   int64
 7   MntFishProducts      2205 non-null   int64
 8   MntSweetProducts     2205 non-null   int64
 9   MntGoldProds         2205 non-null   int64
 10  NumDealsPurchases    2205 non-null   int64
 11  NumWebPurchases      2205 non-null   int64
 12  NumCatalogPurchases  2205 non-null   int64
 13  NumStorePurchases    2205 non-null   int64
 14  NumWebVisitsMonth    2205 non-null   int64
 15  Campaign 3           2205 non-null   object
 16  Campaign 4           2205 non-null   object
 17  Campaign 5           2205 non-null   object
 18  Campaign 1           2205 non-null   object
 19  Campaign 2           2205 non-null   object
```

```
 20  Complain             2205 non-null   object
 21  Campaign 6           2205 non-null   object
 22  Age                  2205 non-null   int64
 23  Customer_Days        2205 non-null   int64
 24  MntTotal             2205 non-null   int64
 25  MntRegularProds      2205 non-null   int64
 26  AcceptedCmpOverall   2205 non-null   int64
 27  maritalStatus        2205 non-null   object
 28  educationLevel       2205 non-null   object
dtypes: float64(1), int64(19), object(9)
memory usage: 499.7+ KB
```

Univariate Analysis

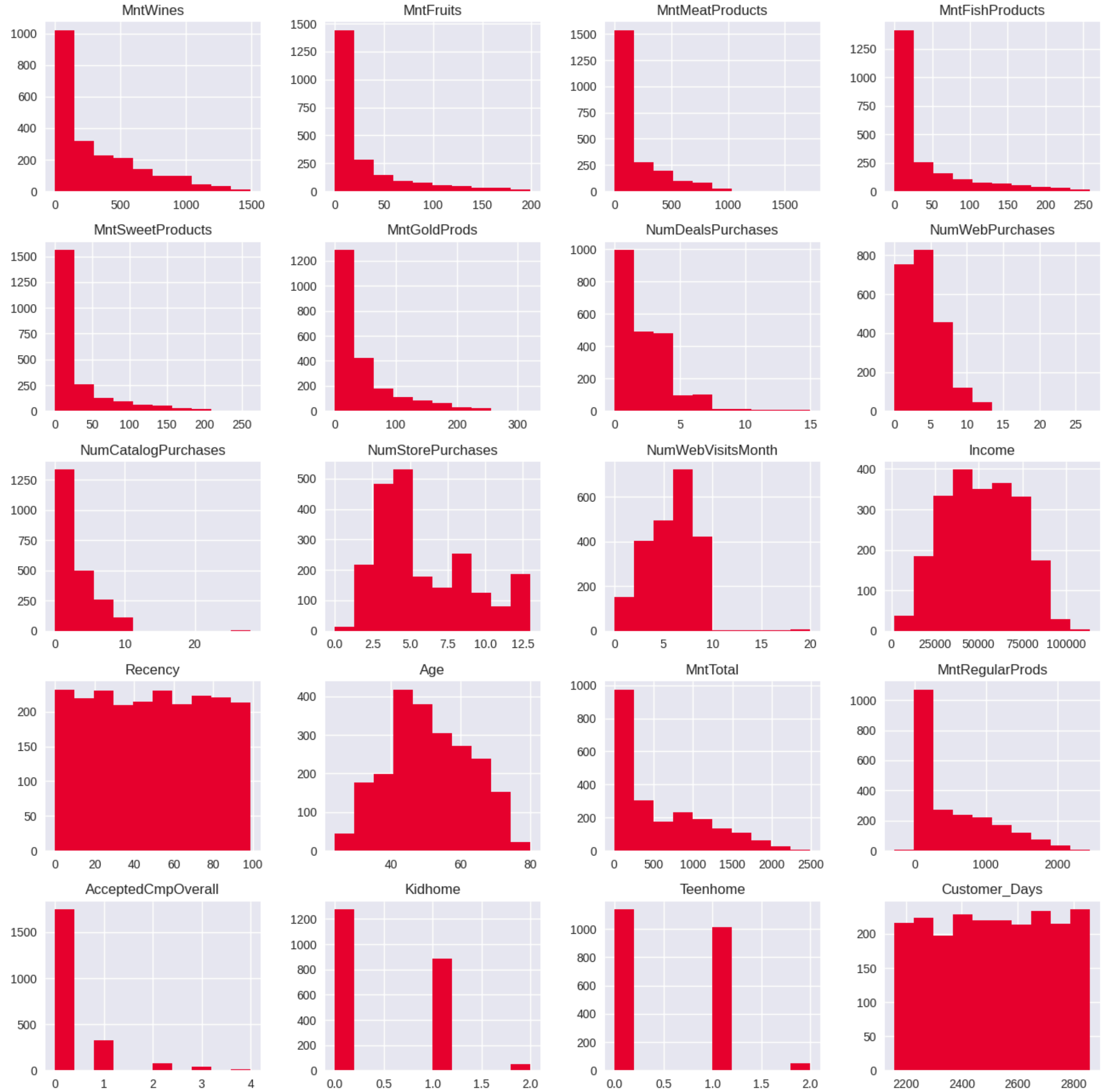
Numerical variables

```
colunas_numericas = ['MntWines', 'MntFruits','MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'Num
                   'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth','Income', 'Recency',
                   'Age', 'MntTotal', 'MntRegularProds', 'AcceptedCmpOverall', 'Kidhome', 'Teenhome', 'Customer_Days']
df_numerical = df[colunas_numericas]
```

Histograms

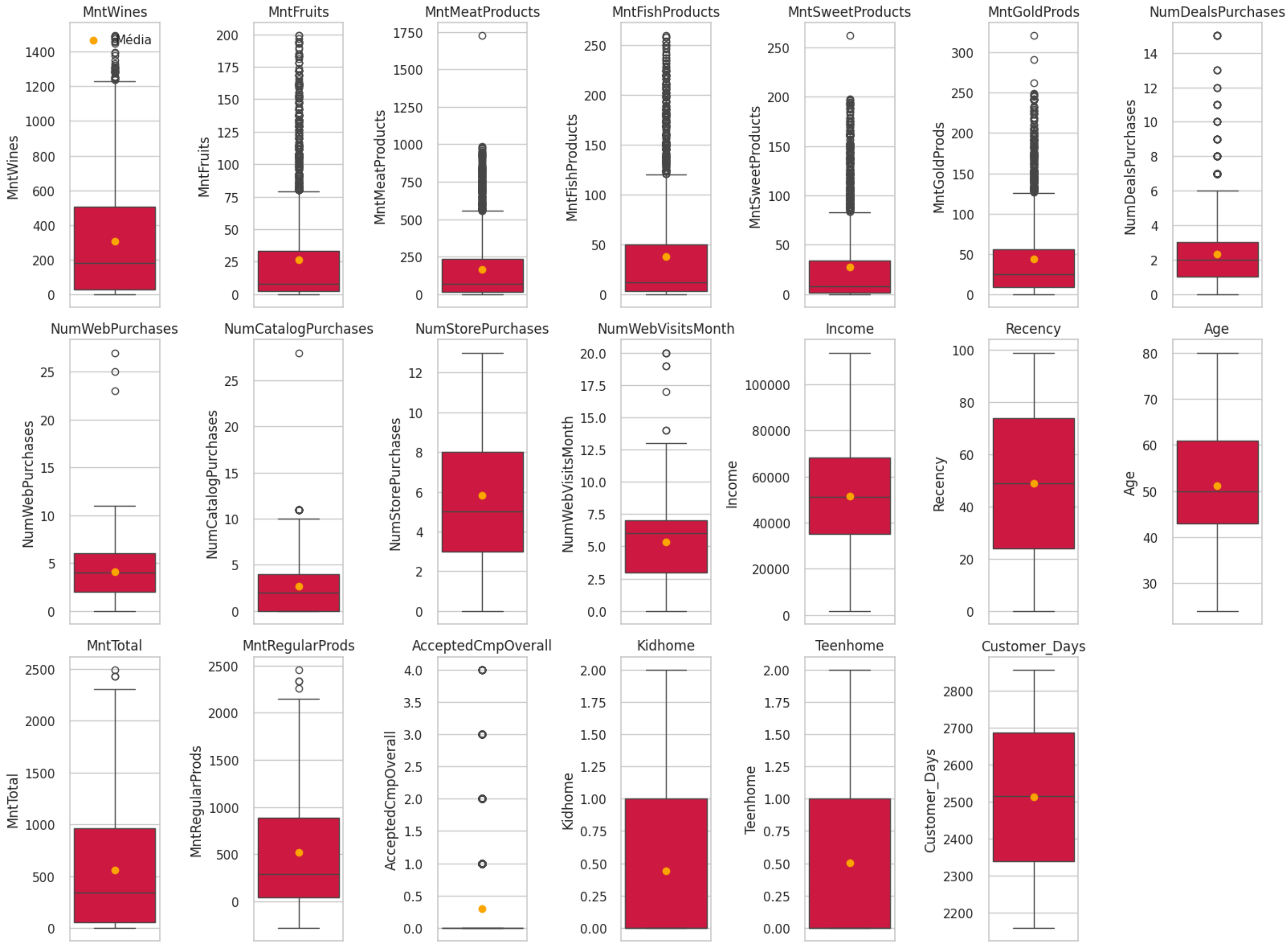
```
df_numerical.hist(figsize=(16,16), color = '#EA0031')
```

```
array([[<Axes: title={'center': 'MntWines'}>,
        <Axes: title={'center': 'MntFruits'}>,
        <Axes: title={'center': 'MntMeatProducts'}>,
        <Axes: title={'center': 'MntFishProducts'}>],
       [<Axes: title={'center': 'MntSweetProducts'}>,
        <Axes: title={'center': 'MntGoldProds'}>,
        <Axes: title={'center': 'NumDealsPurchases'}>,
        <Axes: title={'center': 'NumWebPurchases'}>],
       [<Axes: title={'center': 'NumCatalogPurchases'}>,
        <Axes: title={'center': 'NumStorePurchases'}>,
        <Axes: title={'center': 'NumWebVisitsMonth'}>,
        <Axes: title={'center': 'Income'}>],
       [<Axes: title={'center': 'Recency'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'MntTotal'}>,
        <Axes: title={'center': 'MntRegularProds'}>],
       [<Axes: title={'center': 'AcceptedCmpOverall'}>,
        <Axes: title={'center': 'Kidhome'}>,
        <Axes: title={'center': 'Teenhome'}>,
        <Axes: title={'center': 'Customer_Days'}>]], dtype=object)
```



Box-Plots

```
plt.figure(figsize=(16, 12))  # Ajuste o tamanho conforme necessário
sns.set(style="whitegrid")

# Calcula a média de todas as colunas numéricas uma única vez
means = df[colunas_numericas].mean()

for i, coluna in enumerate(colunas_numericas, 1):
    plt.subplot(3, 7, i)  # Ajuste para a quantidade correta de subplots, 3 linhas e até 7 colunas aqui
    sns.boxplot(y=df[coluna], color='#EA0031')

    # Plotagem da média para a coluna atual
    # plt.scatter espera coordenadas x, y. Usamos '0' para x porque temos apenas uma categoria.
    plt.scatter(0, means[coluna], color='#FEA500', label='Média', zorder=5)
    plt.title(coluna)

    if i == 1:
        plt.legend(loc="upper left")
    plt.title(coluna)

plt.tight_layout()

plt.show()
```


MntWines
Média
MntFruits
MntMeatProducts
MntFishProducts
MntSweetProducts
MntGoldProds
NumDealsPurchases
NumWebPurchases
NumCatalogPurchases
NumStorePurchases
NumWebVisitsMonth
Income
Recency
Age
MntTotal
MntRegularProds
AcceptedCmpOverall
Kidhome
Teenhome
Customer_Days

Statistics

```
def calc_statistics(df):

    # Central tendency and position statistics
    means = round(df.mean(), 3)  # Mean
    trimmed_means = round(df.apply(lambda x: stats.trim_mean(x, 0.05)), 3)  # Calculates trimmed mean, excluding the bottom
    modes = df.mode().iloc[0]  # Mode. Since a column can have multiple modes, we take the first one.
    first_quartile = df.quantile(0.25)  # First quartile
    medians = df.median()  # Second quartile // median
    third_quartile = df.quantile(0.75)  # Third quartile

    # Dispersion statistics
    iqr = third_quartile - first_quartile  # Interquartile Range
    variance = round(df.var(), 3)  # Variance
    standard_deviation = round(df.std(), 3)  # Standard Deviation
    coefficient_variation = round((standard_deviation / means) * 100, 3) # Coefficient of Variation

    # Combines all the results into a new DataFrame
    statistics_df = pd.DataFrame({
        'Mean': means,
        'Trimmed Mean': trimmed_means,
        'Mode': modes,
        '1st Quartile': first_quartile,
        'Median': medians,
        '3rd Quartile': third_quartile,
        'IQR': iqr,
        'Variance': variance,
        'Standard Deviation': standard_deviation,
        'Coefficient of Variation (%)': coefficient_variation
    })

    return statistics_df


df_estatisticas = calc_statistics(df_numerical)

df_estatisticas
```

	Mean	Trimmed Mean	Mode	1st Quartile	Median	3rd Quartile	IQR	Variance	Standard Deviation	Coefficient of Variation (%)
MntWines	306.165	274.828	2.0	24.0	178.0	507.0	483.0	1.139021e+05	337.494	110.233
MntFruits	26.403	20.809	0.0	2.0	8.0	33.0	31.0	1.582805e+03	39.784	150.680
MntMeatProducts	165.312	138.323	7.0	16.0	68.0	232.0	216.0	4.743009e+04	217.785	131.742
MntFishProducts	37.756	30.523	0.0	3.0	12.0	50.0	47.0	3.005741e+03	54.825	145.209
MntSweetProducts	27.128	21.385	0.0	1.0	8.0	34.0	33.0	1.691715e+03	41.130	151.615
MntGoldProds	44.057	37.787	3.0	9.0	25.0	56.0	47.0	2.676636e+03	51.736	117.430
NumDealsPurchases	2.318	2.096	1.0	1.0	2.0	3.0	2.0	3.557000e+00	1.886	81.363
NumWebPurchases	4.101	3.932	2.0	2.0	4.0	6.0	4.0	7.493000e+00	2.737	66.740
NumCatalogPurchases	2.645	2.388	0.0	0.0	2.0	4.0	4.0	7.832000e+00	2.799	105.822
NumStorePurchases	5.824	5.663	3.0	3.0	5.0	8.0	5.0	1.050900e+01	3.242	55.666
NumWebVisitsMonth	5.337	5.346	7.0	3.0	6.0	7.0	4.0	5.825000e+00	2.414	45.231
Income	51622.095	51630.889	7500.0	35196.0	51287.0	68281.0	33085.0	4.290310e+08	20713.064	40.124
Recency	49.009	49.001	56.0	24.0	49.0	74.0	50.0	8.370670e+02	28.932	59.034
Age	51.096	51.071	44.0	43.0	50.0	61.0	18.0	1.370260e+02	11.706	22.910
MntTotal	562.765	517.364	39.0	56.0	343.0	964.0	908.0	3.317033e+05	575.937	102.341
MntRegularProds	518.707	472.892	16.0	42.0	288.0	884.0	842.0	3.067468e+05	553.847	106.775
AcceptedCmpOverall	0.299	0.188	0.0	0.0	0.0	0.0	0.0	4.630000e-01	0.680	227.425
Kidhome	0.442	0.413	0.0	0.0	0.0	1.0	1.0	2.890000e-01	0.537	121.493

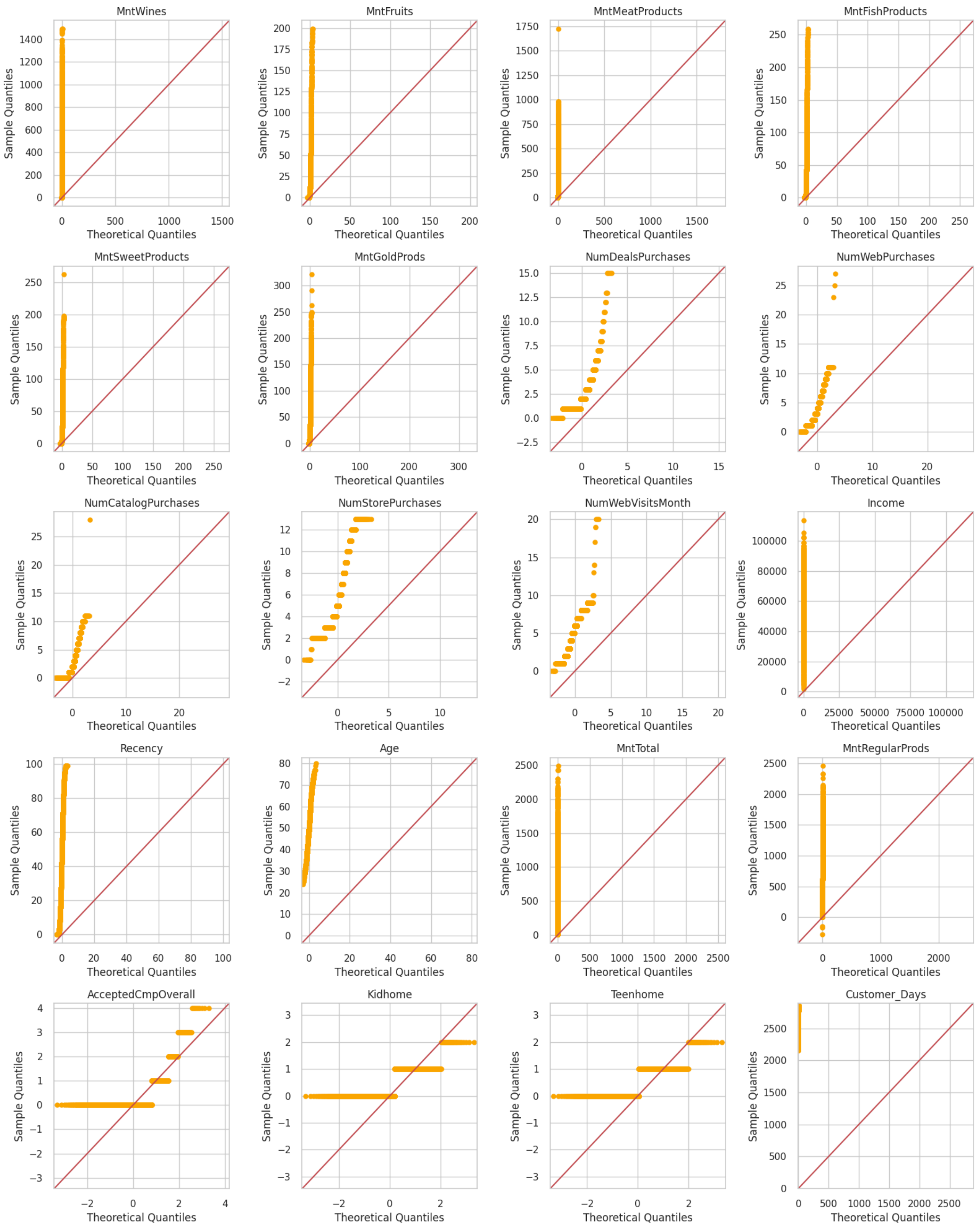
Next steps: View recommended plots

```
    # Criar uma figura. Ajuste o tamanho conforme necessário
    fig, ax = plt.subplots(figsize=(10, 4))  # Largura e altura da figura em polegadas

    # Esconder os eixos
    ax.axis('tight')
    ax.axis('off')

    # Preparar dados com índice incluído
    data_with_index = df_estatisticas.reset_index().values  # Reset index para converter o índice em uma coluna
    column_labels = ['Index'] + list(df_estatisticas.columns)  # Adiciona 'Index' à lista de rótulos de coluna

    # Criar a tabela no gráfico
    table = ax.table(cellText=data_with_index, colLabels=column_labels, loc='center')

    # Ajustar o tamanho da fonte e largura das colunas conforme necessário
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.auto_set_column_width(col=list(range(len(column_labels))))  # Ajusta a largura das colunas

    # Salvar a imagem
    plt.savefig("df_estatisticas_with_index.png", dpi=300, bbox_inches='tight')  # Ajuste a resolução conforme necessário

    # Mostrar a imagem
    plt.show()
```

Index	Mean	Trimmed Mean	Mode	1st Quartile	Median	3rd Quartile	IQR	Variance	Standard Deviation	Coefficient of Variation (%)
MntWines	306.165	274.828	2.0	24.0	178.0	507.0	483.0	113902.091	337.494	110.233
MntFruits	26.403	20.809	0.0	2.0	8.0	33.0	31.0	1582.805	39.784	150.68
MntMeatProducts	165.312	138.323	7.0	16.0	68.0	232.0	216.0	47430.091	217.785	131.742
MntFishProducts	37.756	30.523	0.0	3.0	12.0	50.0	47.0	3005.741	54.825	145.209
MntSweetProducts	27.128	21.385	0.0	1.0	8.0	34.0	33.0	1691.715	41.13	151.615
MntGoldProds	44.057	37.787	3.0	9.0	25.0	56.0	47.0	2676.636	51.736	117.43
NumDealsPurchases	2.318	2.096	1.0	1.0	2.0	3.0	2.0	3.557	1.886	81.363
NumWebPurchases	4.101	3.932	2.0	2.0	4.0	6.0	4.0	7.493	2.737	66.74
NumCatalogPurchases	2.645	2.388	0.0	0.0	2.0	4.0	4.0	7.832	2.799	105.822
NumStorePurchases	5.824	5.663	3.0	3.0	5.0	8.0	5.0	10.509	3.242	55.666
NumWebVisitsMonth	5.337	5.346	7.0	3.0	6.0	7.0	4.0	5.825	2.414	45.231
Income	51622.095	51630.889	7500.0	35196.0	51287.0	68281.0	33085.0	429031013.055	20713.064	40.124
Recency	49.009	49.001	56.0	24.0	49.0	74.0	50.0	837.067	28.932	59.034
Age	51.096	51.071	44.0	43.0	50.0	61.0	18.0	137.026	11.706	22.91
MntTotal	562.765	517.364	39.0	56.0	343.0	964.0	908.0	331703.325	575.937	102.341
MntRegularProds	518.707	472.892	16.0	42.0	288.0	884.0	842.0	306746.774	553.847	106.775
AcceptedCmpOverall	0.299	0.188	0.0	0.0	0.0	0.0	0.0	0.463	0.68	227.425
Kidhome	0.442	0.413	0.0	0.0	0.0	1.0	1.0	0.289	0.537	121.493
Teenhome	0.507	0.482	0.0	0.0	0.0	1.0	1.0	0.296	0.544	107.298
Customer_Days	2512.718	2513.052	2826.0	2339.0	2515.0	2688.0	349.0	41032.031	202.564	8.062

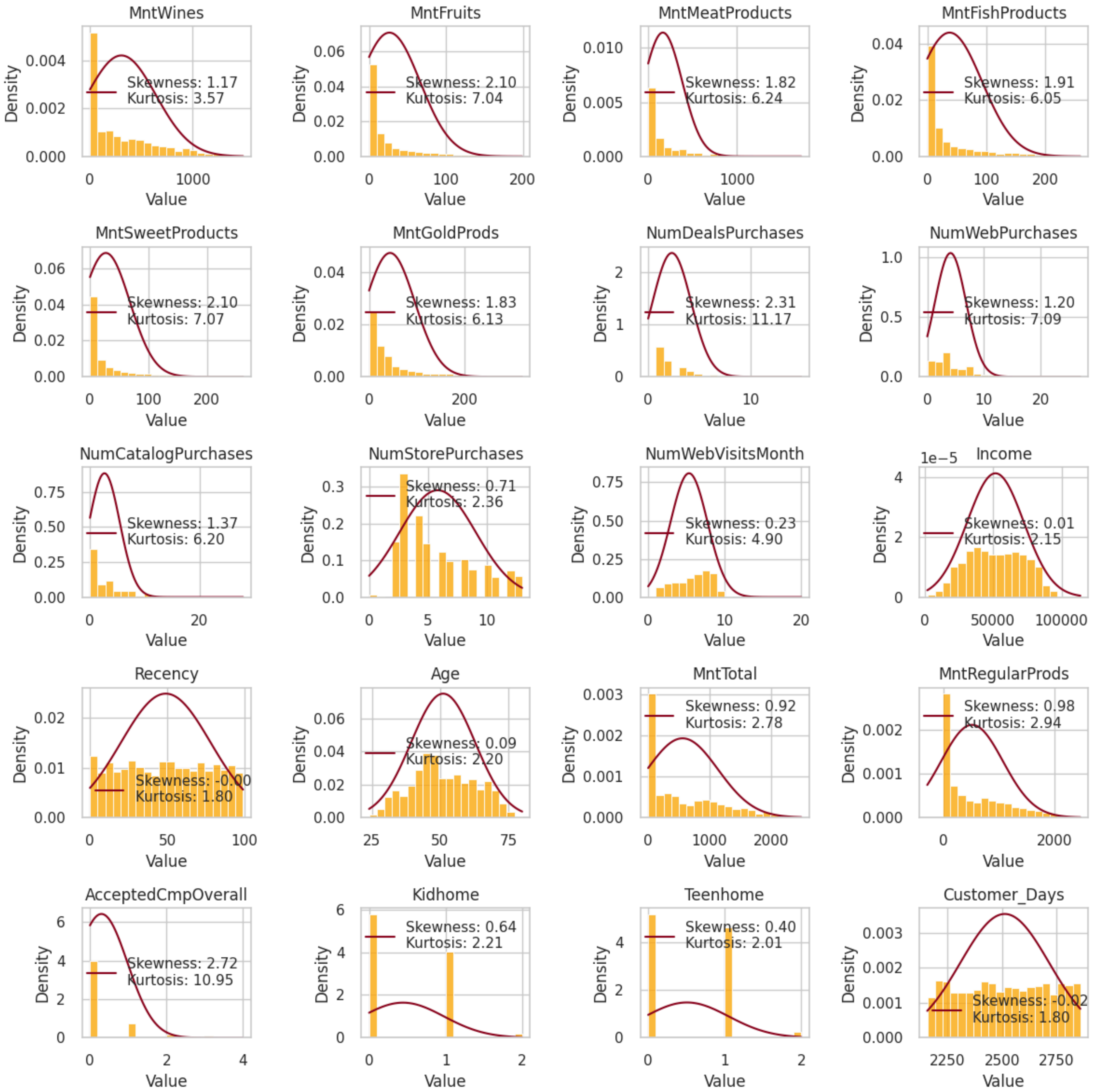
QQ plot

```
import statsmodels.api as sm
import matplotlib.pyplot as plt


# Calcular o número de subplots necessários
num_cols = len(df_numerical.columns)
num_rows = (num_cols + 3) // 4

# Configurar o layout dos subplots
fig, axes = plt.subplots(num_rows, 4, figsize=(16, num_rows * 4))

axes = axes.flatten() if num_rows > 1 else [axes]

# Plotar QQ plots para as variáveis
for i, column in enumerate(df_numerical.columns):
    ax = axes[i]

    # Plotar QQ plot
    sm.qqplot(df_numerical[column], line ='45', ax=ax, marker='o', markeredgecolor='#FEA500', markerfacecolor='#FEA500')

    ax.set_title(column)
    ax.set_xlabel('Theoretical Quantiles')
    ax.set_ylabel('Sample Quantiles')
    ax.grid(True)

# Remover eixos desnecessários
for i in range(num_cols, num_rows * 4):
    fig.delaxes(axes[i])

# Ajustar layout
plt.tight_layout()
plt.show()
```

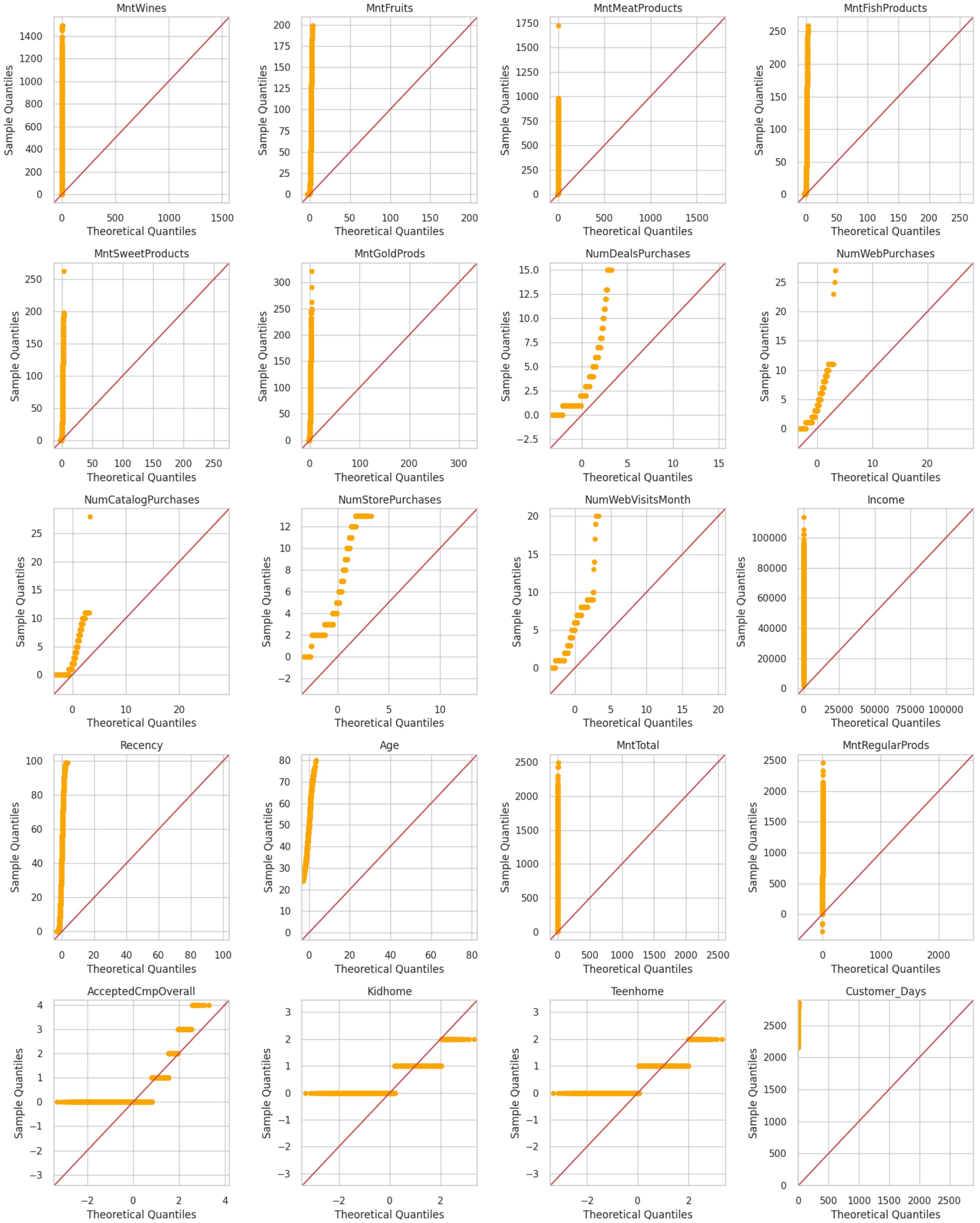

MntWines
MntFruits
MntMeatProducts
MntFishProducts
MntSweetProducts
MntGoldProds
NumDealsPurchases
NumWebPurchases
NumCatalogPurchases
NumStorePurchases
NumWebVisitsMonth
Income
Recency
Age
MntTotal
MntRegularProds
AcceptedCmpOverall
Kidhome
Teenhome
Customer_Days
Sample Quantiles
Theoretical Quantiles

Skewness and Kurtosis

```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, kurtosis


# Calcular o número de subplots necessários
#num_cols = len(df_numerical.columns)
#num_rows = (num_cols + 3) // 4  # Arredondamento para cima

# Configurar o layout dos subplots
fig, axes = plt.subplots(5, 4, figsize=(12,12))

# Flattening the axes array if num_rows = 1
axes = axes.flatten() if num_rows > 1 else [axes]

# Plotar histogramas com curva sinoidal e valores de skewness e kurtosis
for i, column in enumerate(df_numerical.columns):
    ax = axes[i]

    # Plotar histograma
    sns.histplot(df_numerical[column], kde=False, color='#FEA500', stat='density', bins=20, ax=ax)

    # Calcular skewness e kurtosis
    skew = df_numerical[column].skew()
    kurt = kurtosis(df_numerical[column], fisher=False)

    # Gerar dados para a curva sinoidal
    x = np.linspace(df_numerical[column].min(), df_numerical[column].max(), 100)
    y = norm.pdf(x, df_numerical[column].mean(), df_numerical[column].std()) * kurt

    # Plotar a curva sinoidal
    ax.plot(x, y, label=f'Skewness: {skew:.2f}\nKurtosis: {kurt:.2f}', color='#8A011B')

    ax.set_title(column)
    ax.set_xlabel('Value')
    ax.set_ylabel('Density')
    ax.legend()
    ax.grid(True)



#plt.delaxes(plt.subplot(3, 3, 7))

# Ajustar layout
plt.tight_layout()
plt.show()
```


MntWines
Skewness: 1.17
Kurtosis: 3.57
MntFruits
Skewness: 2.10
Kurtosis: 7.04
MntMeatProducts
Skewness: 1.82
Kurtosis: 6.24
MntFishProducts
Skewness: 1.91
Kurtosis: 6.05
MntSweetProducts
Skewness: 2.10
Kurtosis: 7.07
MntGoldProds
Skewness: 1.83
Kurtosis: 6.13
NumDealsPurchases
Skewness: 2.31
Kurtosis: 11.17
NumWebPurchases
Skewness: 1.20
Kurtosis: 7.09
NumCatalogPurchases
Skewness: 1.37
Kurtosis: 6.20
NumStorePurchases
Skewness: 0.71
Kurtosis: 2.36
NumWebVisitsMonth
Skewness: 0.23
Kurtosis: 4.90
Income
1e−5
Skewness: 0.01
Kurtosis: 2.15
Recency
Skewness: -0.00
Kurtosis: 1.80
Age
Skewness: 0.09
Kurtosis: 2.20
MntTotal
Skewness: 0.92
Kurtosis: 2.78
MntRegularProds
Skewness: 0.98
Kurtosis: 2.94
AcceptedCmpOverall
Skewness: 2.72
Kurtosis: 10.95
Kidhome
Skewness: 0.64
Kurtosis: 2.21
Teenhome
Skewness: 0.40
Kurtosis: 2.01
Customer_Days
Skewness: -0.02
Kurtosis: 1.80
Density
Value

```
import statsmodels.api as sm
import matplotlib.pyplot as plt


# Calcular o número de subplots necessários
num_cols = len(df_numerical.columns)
num_rows = (num_cols + 3) // 4

# Configurar o layout dos subplots
fig, axes = plt.subplots(num_rows, 4, figsize=(16, num_rows * 4))

axes = axes.flatten() if num_rows > 1 else [axes]

# Plotar QQ plots para as variáveis
for i, column in enumerate(df_numerical.columns):
    ax = axes[i]

    # Plotar QQ plot
    sm.qqplot(df_numerical[column], line ='45', ax=ax, marker='o', markeredgecolor='#FEA500', markerfacecolor='#FEA500')

    ax.set_title(column)
    ax.set_xlabel('Theoretical Quantiles')
    ax.set_ylabel('Sample Quantiles')
    ax.grid(True)

# Remover eixos desnecessários
for i in range(num_cols, num_rows * 4):
    fig.delaxes(axes[i])

# Ajustar layout
plt.tight_layout()
plt.show()
```


MntWines
MntFruits
MntMeatProducts
MntFishProducts
MntSweetProducts
MntGoldProds
NumDealsPurchases
NumWebPurchases
NumCatalogPurchases
NumStorePurchases
NumWebVisitsMonth
Income
Recency
Age
MntTotal
MntRegularProds
AcceptedCmpOverall
Kidhome
Teenhome
Customer_Days
Sample Quantiles
Theoretical Quantiles

```
from scipy.stats import shapiro

# Empty list to store the results
results = []

# Perform Shapiro's normality test for each numerical variable
for column in df_numerical.columns:
    stat, p_value = shapiro(df_numerical[column])

    # Determine if the variable is normal or not based on the p-value
    is_normal = 'Normality confirmed' if p_value > 0.05 else 'Normality not confirmed'

    # Add the results to the list
    results.append({
        'Variable': column,
        'P-value': p_value,
        'Normality': is_normal
    })

# Create a DataFrame from the list of results
normality_results = pd.DataFrame(results)

# Print or save the DataFrame with the results
normality_results
```

	Variable	P-value	Normality
0	MntWines	1.537224e-42	Normality not confirmed
1	MntFruits	0.000000e+00	Normality not confirmed
2	MntMeatProducts	0.000000e+00	Normality not confirmed
3	MntFishProducts	0.000000e+00	Normality not confirmed
4	MntSweetProducts	0.000000e+00	Normality not confirmed
5	MntGoldProds	0.000000e+00	Normality not confirmed
6	NumDealsPurchases	0.000000e+00	Normality not confirmed
7	NumWebPurchases	1.306171e-34	Normality not confirmed
8	NumCatalogPurchases	2.631639e-42	Normality not confirmed
9	NumStorePurchases	1.674303e-35	Normality not confirmed
10	NumWebVisitsMonth	3.505789e-31	Normality not confirmed
11	Income	9.199109e-15	Normality not confirmed
12	Recency	1.051384e-25	Normality not confirmed
13	Age	2.239517e-15	Normality not confirmed
14	MntTotal	8.060129e-41	Normality not confirmed
15	MntRegularProds	1.333616e-41	Normality not confirmed
16	AcceptedCmpOverall	0.000000e+00	Normality not confirmed
17	Kidhome	0.000000e+00	Normality not confirmed
18	Teenhome	0.000000e+00	Normality not confirmed
19	Customer_Days	7.723260e-26	Normality not confirmed

Next steps: View recommended plots

Categorical Variables

Mode

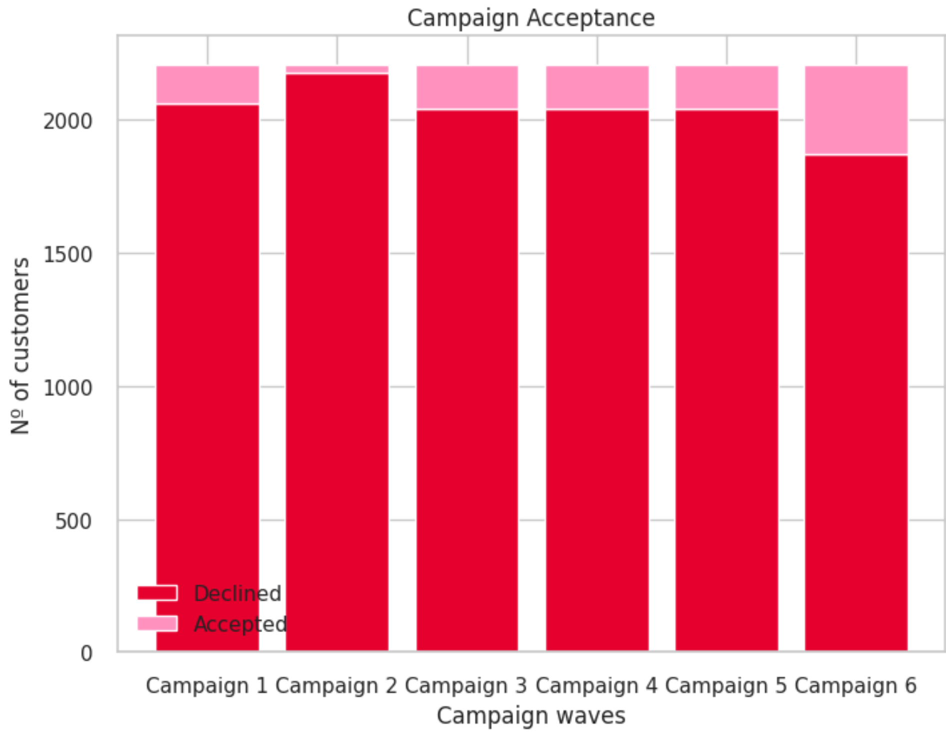
```
df_categorical = df[[ 'Campaign 1','Campaign 2','Campaign 3', 'Campaign 4',
                      'Campaign 5','Campaign 6','maritalStatus', 'educationLevel','Complain']]
cmp_categorical = df[[ 'Campaign 1','Campaign 2','Campaign 3', 'Campaign 4',
                      'Campaign 5','Campaign 6']]
df_categorical.mode()
```

	Campaign 1	Campaign 2	Campaign 3	Campaign 4	Campaign 5	Campaign 6	maritalStatus	educationLevel	Complain
0	Declined	Declined	Declined	Declined	Declined	Declined	Married	Graduation	False

Stacked Bar Plots

Campaign Acceptance

```
# Calcular a contagem de ocorrências de cada valor em cada coluna categórica
contagem_por_valor_cmp = pd.DataFrame({col: cmp_categorical[col].value_counts() for col in cmp_categorical.columns})

# Transpor o DataFrame para ter as variáveis categóricas como índices
contagem_por_valor = contagem_por_valor_cmp.transpose()

# Plotting
plt.figure(figsize=(8, 6))
plt.bar(contagem_por_valor.index, contagem_por_valor['Declined'], label='Declined',color='#EA0031')
plt.bar(contagem_por_valor.index, contagem_por_valor['Accepted'], bottom=contagem_por_valor['Declined'], label='Accepted',c

# Adding labels and title
plt.xlabel('Campaign waves')
plt.ylabel('Nº of customers')
plt.title('Campaign Acceptance')
plt.legend()

# Display the plot
plt.show()
```




```
count_complain = pd.DataFrame(df_categorical['Complain'].value_counts())

# Transpor o DataFrame para ter as variáveis categóricas como índices
count_complain = count_complain
count_complain
```

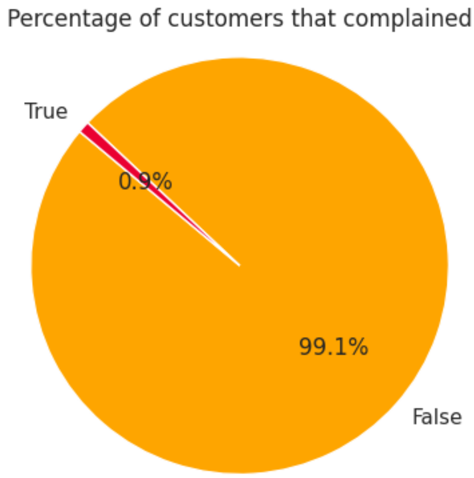
	count
Complain	
False	2185
True	20

Next steps: View recommended plots

Pie Charts

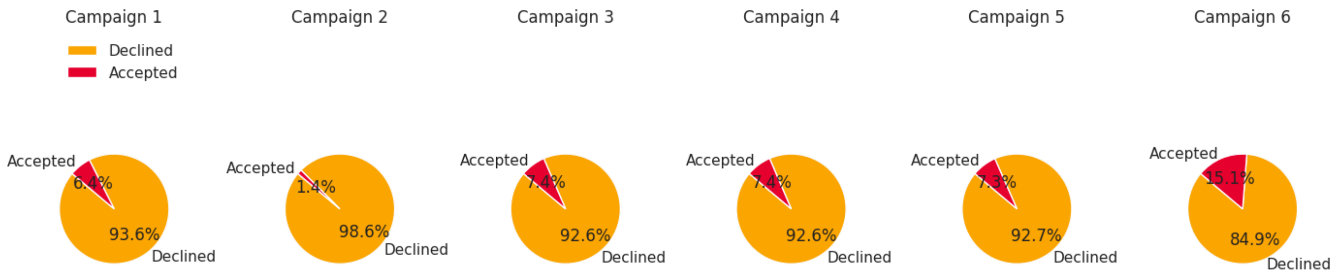
Customers Complain

```
count_complain = pd.DataFrame(df_categorical['Complain'].value_counts())


plt.figure(figsize=(4, 4))  # Ajuste o tamanho conforme necessário
sns.set(style="whitegrid")

# plt.scatter espera coordenadas x, y. Usamos '0' para x porque temos apenas uma categoria.
plt.pie(count_complain['count'], labels=count_complain.index, autopct='%1.1f%%', startangle=140, colors=colors)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
if i == 1:
  plt.legend(loc="upper left")
plt.title('Percentage of customers that complained')

plt.tight_layout()
```



Campaign Acceptance

```
plt.figure(figsize=(16, 12))  # Ajuste o tamanho conforme necessário
sns.set(style="whitegrid")

pie_chart_camp_acctpce = contagem_por_valor.transpose()

for i, coluna in enumerate(pie_chart_camp_acctpce, 1):
    plt.subplot(3, 7, i)  # Ajuste para a quantidade correta de subplots, 3 linhas e até 7 colunas aqui

    # plt.scatter espera coordenadas x, y. Usamos '0' para x porque temos apenas uma categoria.
    plt.pie(pie_chart_camp_acctpce[coluna], labels=pie_chart_camp_acctpce.index, autopct='%1.1f%%', startangle=140, colors=
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    if i == 1:
        plt.legend(loc="upper left")
    plt.title(coluna)

plt.tight_layout()
```



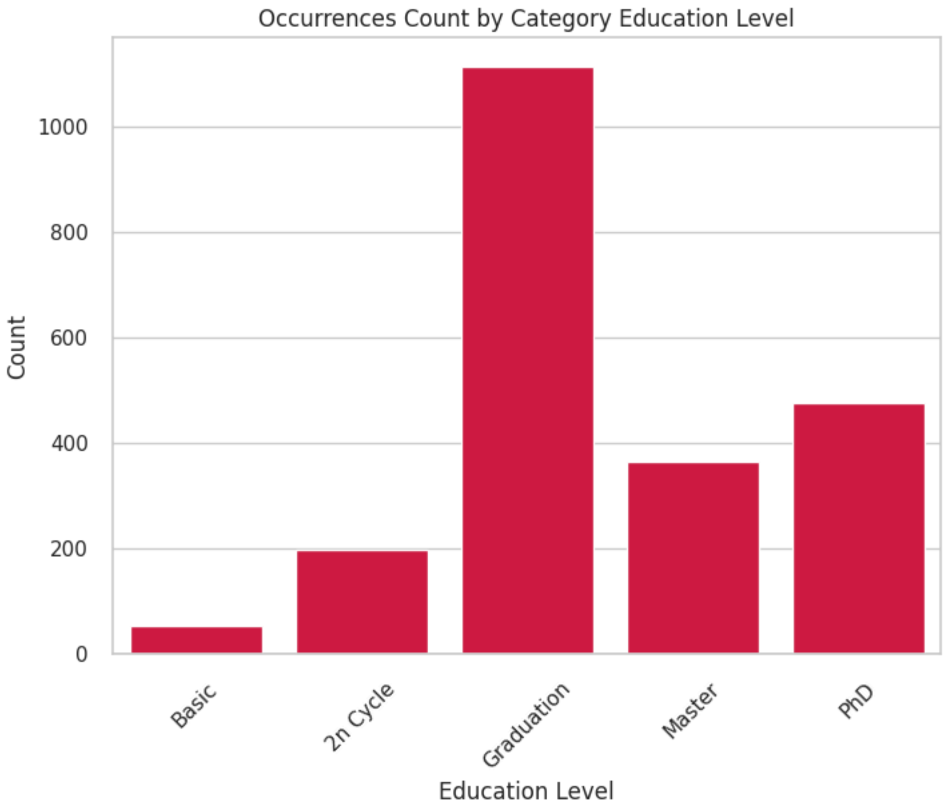
```
contagem_por_valor_edu = pd.DataFrame(df_categorical['educationLevel'].value_counts())
contagem_por_valor_edu.transpose()
```

educationLevel	Graduation	PhD	Master	2n Cycle	Basic
count	1113	476	364	198	54

Bar Plots

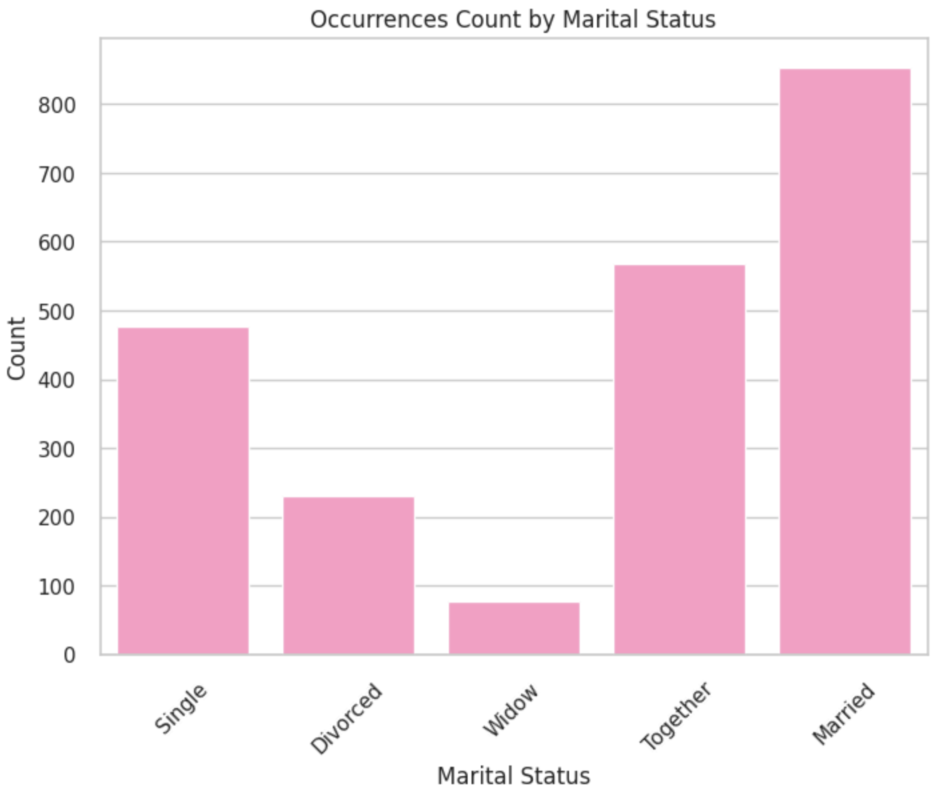
Education Level

```
# Calcular a contagem de ocorrências de cada valor em cada coluna categórica
contagem_por_valor_edu = pd.DataFrame(df_categorical['educationLevel'].value_counts())

contagem_por_valor_edu = contagem_por_valor_edu.reindex(['Basic', '2n Cycle', 'Graduation', 'Master', 'PhD'])
# Plotar gráficos de barras para cada coluna categórica
plt.figure(figsize=(8, 6))
sns.barplot(x=contagem_por_valor_edu.index, y=contagem_por_valor_edu['count'], data=contagem_por_valor_edu, color='#EA0031'
plt.title(f'Occurrences Count by Category Education Level')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



Marital Status

```
# Calcular a contagem de ocorrências de cada valor em cada coluna categórica
contagem_por_valor_marit = pd.DataFrame(df_categorical['maritalStatus'].value_counts())

contagem_por_valor_marit = contagem_por_valor_marit.reindex(['Single', 'Divorced', 'Widow', 'Together', 'Married'])
# Plotar gráficos de barras para cada coluna categórica
plt.figure(figsize=(8, 6))
sns.barplot(x=contagem_por_valor_marit.index, y=contagem_por_valor_marit['count'], data=contagem_por_valor_marit, color='#F
plt.title(f'Occurrences Count by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



Bivariate Analysis

Possiveis combinações para analisar:

- Cat x Cat;
- Cat x Num;
- Num x Num;

--- Dadas as variáveis do dataset, podemos considerar como importante para o negocio perceber qual o perfil do publico e a eficiencia de campanhas publicitárias.

Cat x Cat:

- AcceptedN + Response x maritalStatus;
- AcceptedN + Response x educationLevel;
- Complain x maritalStatus;
- Complain x educationLevel

Cat x Num:

- 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'MntTotal' x maritalStatus;
- 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'MntRegularProds' x educationLevel;
- 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'MntRegularProds' x Complain;
- 'NumDealsPurchases','NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases','NumWebVisitsMonth', 'MntTotal' x maritalStatus;
- 'NumDealsPurchases','NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases','NumWebVisitsMonth' x educationLevel;
- 'NumDealsPurchases','NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases','NumWebVisitsMonth' x Complain;
- 'Income' x maritalStatus; - makes no sense for business logic
- 'Income' x educationLevel; - makes no sense for business logic
- 'Recency' x maritalStatus;
- 'Recency' x educationLevel;
- 'Recency' x Complain;
- 'Age' x maritalStatus; - makes no sense for business logic
- 'Age' x educationLevel; - makes no sense for business logic
- 'AcceptedCmpOverall' x maritalStatus;
- 'AcceptedCmpOverall' x educationLevel;
- 'AcceptedCmpOverall' x Complain;

Num x Num:

- Numerical variables - Correlation Matrix - Correlation analysis
    - Extract some relevant correlations and make scatter-plots

Categorical x Categorical

Campaign x maritalStatus/educationLevel

Contigency Table | Campaign x MaritalStatus

```
def generate_cont_table_cat1_dfcat2(cat1,df_cat_2):
  list_dfs_cont_tables = []
  for col in df_cat_2.columns:
    col_df_cat1_x_cat2 = pd.crosstab(index=df_categorical[cat1], columns=[cmp_categorical[col]], rownames=[cat1],margins=Tr
    list_dfs_cont_tables.append(col_df_cat1_x_cat2)
    print(col_df_cat1_x_cat2)

  return list_dfs_cont_tables
```

```
list_cmp_x_mart_status = generate_cont_table_cat1_dfcat2('maritalStatus',cmp_categorical)
```

```
Campaign 1     Accepted  Declined  Total
maritalStatus
Divorced             12       218    230
Married              62       792    854
Single               31       446    477
Together             32       536    568
Widow                 5        71     76
Total               142      2063   2205
Campaign 2     Accepted  Declined  Total
maritalStatus
Divorced              5       225    230
Married               7       847    854
Single                5       472    477
Together             12       556    568
Widow                 1        75     76
Total                30      2175   2205
Campaign 3     Accepted  Declined  Total
maritalStatus
Divorced             20       210    230
Married              63       791    854
Single               39       438    477
Together             37       531    568
Widow                 4        72     76
Total               163      2042   2205
Campaign 4     Accepted  Declined  Total
maritalStatus
Divorced             18       212    230
Married              62       792    854
Single               32       445    477
Together             42       526    568
Widow                10        66     76
Total               164      2041   2205
Campaign 5     Accepted  Declined  Total
maritalStatus
Divorced             13       217    230
Married              66       788    854
Single               32       445    477
Together             43       525    568
Widow                 7        69     76
Total               161      2044   2205
Campaign 6     Accepted  Declined  Total
maritalStatus
Divorced             48       182    230
Married              98       756    854
Single              109       368    477
Together             60       508    568
Widow                18        58     76
Total               333      1872   2205
```

ChiSquare Test | Campaign x MaritalStatus

```
from scipy.stats import chi2_contingency
```

```
campaign_names = ["Campaign 1", "Campaign 2", "Campaign 3", "Campaign 4", "Campaign 5", "Campaign 6"]
results_marital = pd.DataFrame(columns=campaign_names, index=["Chi2 Statistic", "P-value"])

# Calcular o qui-quadrado e o p-valor para cada tabela de contingência
for campaign, table in zip(campaign_names, list_cmp_x_mart_status):
    chi2, p, dof, expected = chi2_contingency(table.iloc[:-1, :-1])  # Exclui a linha e coluna 'Total'
    results_marital[campaign] = [chi2, p]

# Mostra os resultados
print("Chi-Squared Test of Independence: Marital Status vs. Marketing Campaigns\n")
results_marital
```

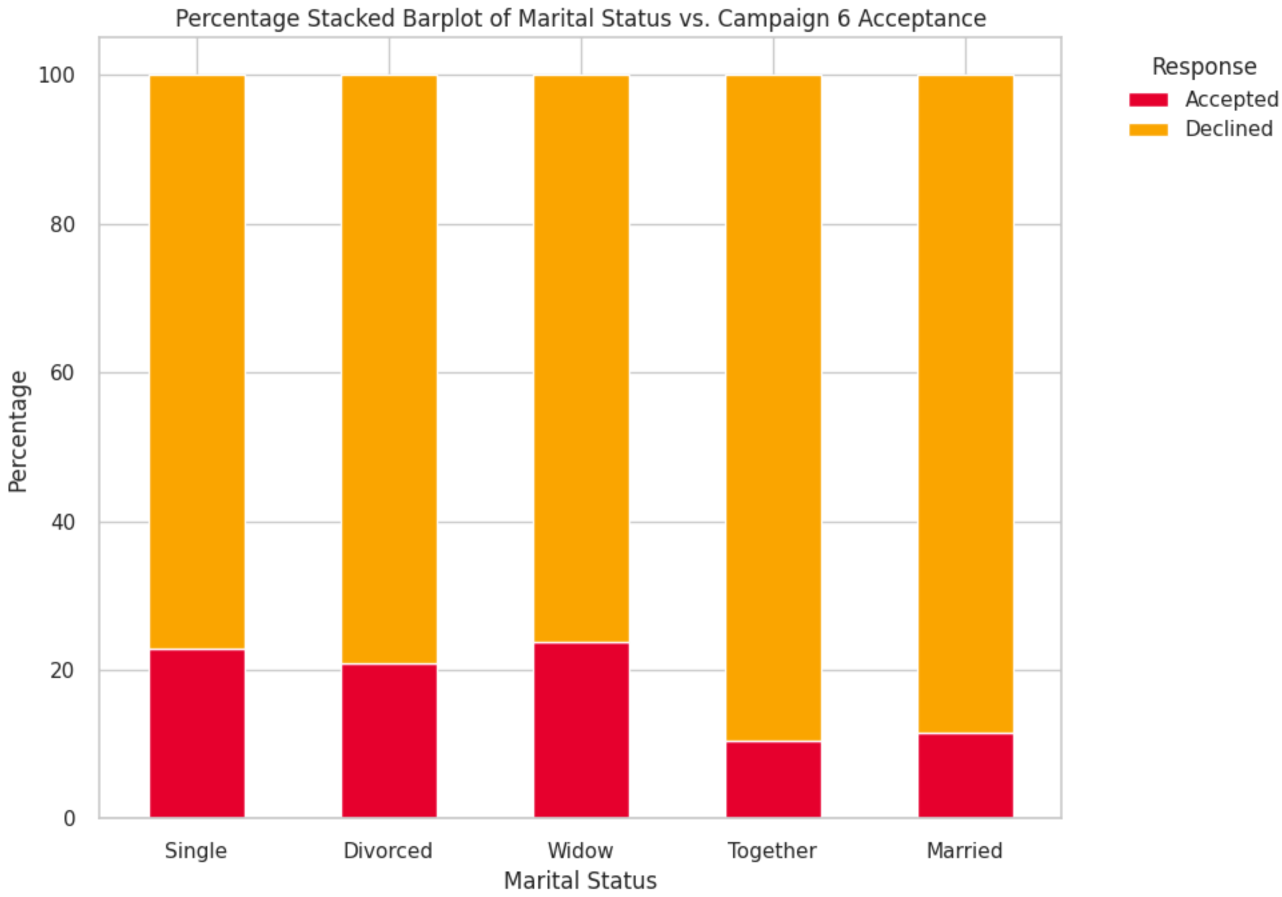
```
Chi-Squared Test of Independence: Marital Status vs. Marketing Campaigns
```

	Campaign 1	Campaign 2	Campaign 3	Campaign 4	Campaign 5	Campaign 6
Chi2 Statistic	2.141445	5.737514	2.142255	4.071607	1.871900	5.055983e+01
P-value	0.709762	0.219627	0.709614	0.396402	0.759305	2.758648e-10

Next steps: View recommended plots

Stacked Bar Plot | Campaign x MaritalStatus

```
# Agrupar os dados por "maritalStatus" e calcular as contagens para cada resposta da ultima campanha
df_grouped = df.groupby('maritalStatus')['Campaign 6'].value_counts().unstack().fillna(0)
df_grouped = df_grouped.reindex(['Single', 'Divorced', 'Widow', 'Together', 'Married'])
df_percent = df_grouped.div(df_grouped.sum(axis=1), axis=0) * 100

# Criar um plot empilhado
ax = df_percent.plot(kind='bar', stacked=True, figsize=(10, 7), color=['#EA0031', '#FEA500'])
ax.set_ylabel('Percentage')
ax.set_xlabel('Marital Status')
ax.set_title('Percentage Stacked Barplot of Marital Status vs. Campaign 6 Acceptance')
plt.xticks(rotation=0)  # Mantém os rótulos na horizontal
plt.legend(title='Response', labels=['Accepted', 'Declined'], bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



Contigency Table | Campaign x EducationLevel

```
list_cmp_x_edu_level = generate_cont_table_cat1_dfcat2('educationLevel', cmp_categorical)
```

```
Campaign 1      Accepted  Declined  Total
educationLevel
2n Cycle              14       184    198
Basic                  0        54     54
Graduation            80      1033   1113
Master                18       346    364
PhD                   30       446    476
Total                142      2063   2205
Campaign 2      Accepted  Declined  Total
educationLevel
2n Cycle               2       196    198
Basic                  0        54     54
Graduation            16      1097   1113
Master                 2       362    364
PhD                   10       466    476
Total                 30      2175   2205
Campaign 3      Accepted  Declined  Total
educationLevel
2n Cycle              15       183    198
Basic                  6        48     54
Graduation            78      1035   1113
Master                24       340    364
PhD                   40       436    476
Total                163      2042   2205
Campaign 4      Accepted  Declined  Total
educationLevel
2n Cycle               9       189    198
Basic                  0        54     54
Graduation            79      1034   1113
Master                31       333    364
PhD                   45       431    476
Total                164      2041   2205
Campaign 5      Accepted  Declined  Total
educationLevel
2n Cycle              10       188    198
Basic                  0        54     54
Graduation            86      1027   1113
Master                27       337    364
PhD                   38       438    476
Total                161      2044   2205
Campaign 6      Accepted  Declined  Total
educationLevel
2n Cycle              22       176    198
Basic                  2        52     54
Graduation           152       961   1113
Master                56       308    364
PhD                  101       375    476
Total                333      1872   2205
```

ChiSquare Test | Campaign X EducationLevel

```
results_educacional = pd.DataFrame(columns=campaign_names, index=["Chi2 Statistic", "P-value"])

# Calcular o qui-quadrado e o p-valor para cada tabela de contingência
for campaign, table in zip(campaign_names, list_cmp_x_edu_level):
    chi2, p, dof, expected = chi2_contingency(table.iloc[:-1, :-1])  # Exclui a linha e coluna 'Total'
    results_educacional[campaign] = [chi2, p]

# Mostra os resultados
print("Relationship between Education Level and Marketing Campaign Effectiveness:Chi-Squared Statistics and P-values.\n")
results_educacional
```

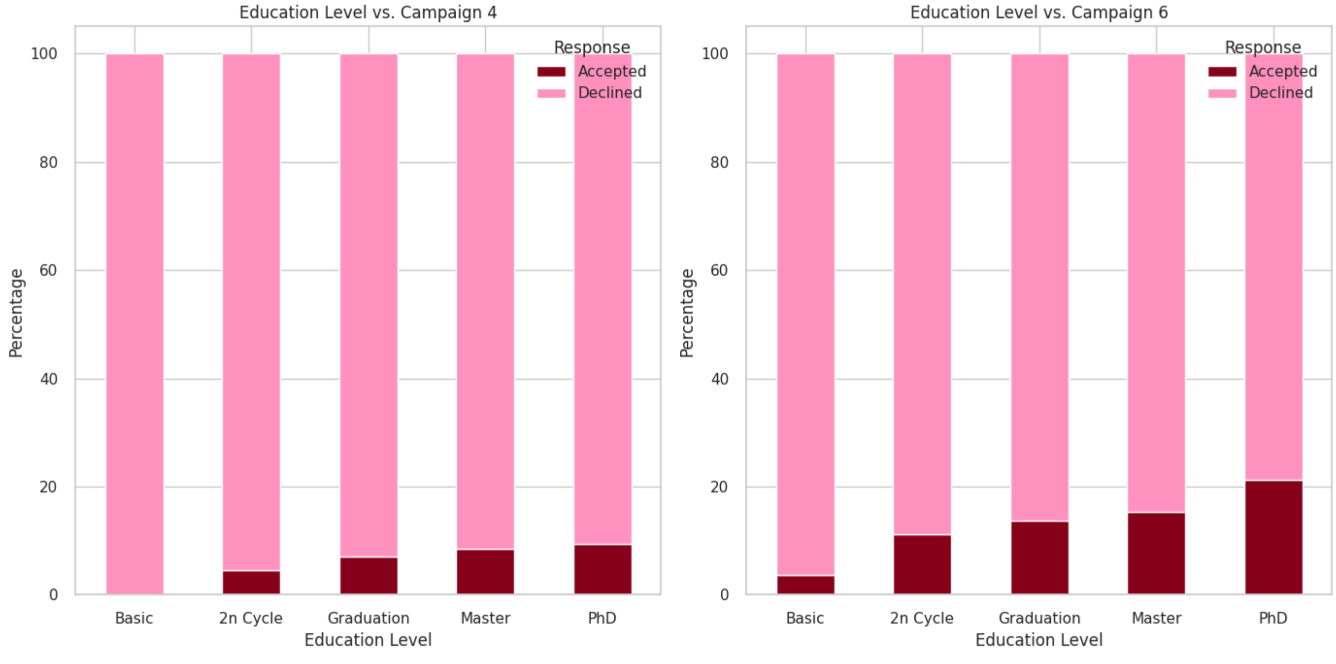
Relationship between Education Level and Marketing Campaign Effectiveness:Chi-Squared Statistics and P-values.

	Campaign 1	Campaign 2	Campaign 3	Campaign 4	Campaign 5	Campaign 6
Chi2 Statistic	6.245760	4.703369	2.390748	10.357209	6.367169	23.656607
P-value	0.181531	0.319109	0.664300	0.034822	0.173355	0.000094

Next steps: View recommended plots

Stacked Bar Plot | Campaign x EducationLevel

```
# Agrupar os dados por "educationLevel" e calcular as contagens para cada resposta da "Proposta 4"
df_grouped_proposal = df.groupby('educationLevel')['Campaign 4'].value_counts().unstack().fillna(0)
df_grouped_proposal = df_grouped_proposal.reindex(['Basic', '2n Cycle', 'Graduation', 'Master', 'PhD'])

# Calcular porcentagens
df_percent_proposal = df_grouped_proposal.div(df_grouped_proposal.sum(axis=1), axis=0) * 100

# Agrupar os dados por "educationLevel" e calcular as contagens para cada resposta da "Response"
df_grouped_response = df.groupby('educationLevel')['Campaign 6'].value_counts().unstack().fillna(0)
df_grouped_response = df_grouped_response.reindex(['Basic', '2n Cycle', 'Graduation', 'Master', 'PhD'])

# Calcular porcentagens
df_percent_response = df_grouped_response.div(df_grouped_response.sum(axis=1), axis=0) * 100


# Criar figura e eixos para os subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 7))

# Gráfico "educationLevel" vs. "Proposal 4"
ax_proposal = df_percent_proposal.plot(kind='bar', stacked=True, ax=axes[0], color=['#8A011B', '#FF94C2'])
ax_proposal.set_ylabel('Percentage')
ax_proposal.set_xlabel('Education Level')
ax_proposal.set_title('Education Level vs. Campaign 4')
ax_proposal.legend(title='Response', labels=['Accepted', 'Declined'])
ax_proposal.set_xticklabels(ax_proposal.get_xticklabels(), rotation=0)

# Gráfico "educationLevel" vs. "Response"
ax_response = df_percent_response.plot(kind='bar', stacked=True, ax=axes[1], color=['#8A011B', '#FF94C2'])
ax_response.set_ylabel('Percentage')
ax_response.set_xlabel('Education Level')
ax_response.set_title('Education Level vs. Campaign 6')
ax_response.legend(title='Response', labels=['Accepted', 'Declined'])
ax_response.set_xticklabels(ax_response.get_xticklabels(), rotation=0)

# Ajustar o layout
plt.tight_layout()
plt.show()
```




```
marit_status_x_edu_level = pd.crosstab(index=df_categorical['educationLevel'], columns=[df_categorical['maritalStatus']], r
marit_status_x_edu_level
```

maritalStatus	Divorced	Married	Single	Together	Widow	Total
educationLevel						
2n Cycle	22	80	35	56	5	198
Basic	1	20	18	14	1	54
Graduation	118	429	248	283	35	1113
Master	37	138	77	101	11	364
PhD	52	187	99	114	24	476
Total	230	854	477	568	76	2205

Next steps: View recommended plots

```
res = chi2_contingency(marit_status_x_edu_level)
print("Qui-Quadrado:", res.statistic)
print("p-value:", res.pvalue)
```

```
Qui-Quadrado: 16.205244635440557
p-value: 0.9085792991588382
```

Complain x maritStatus/educationLevel

```
count_complain_x_marital_status = pd.crosstab(index=df_categorical['Complain'], columns=[df_categorical['maritalStatus']],
count_complain_x_marital_status
```

maritalStatus	Divorced	Married	Single	Together	Widow	Total
Complain						
False	229	846	471	563	76	2185
True	1	8	6	5	0	20
Total	230	854	477	568	76	2205

Next steps: View recommended plots

```
res = chi2_contingency(count_complain_x_marital_status)
print("Qui-Quadrado:", res.statistic)
print("p-value:", res.pvalue)
```

```
Qui-Quadrado: 1.9324785924349863
p-value: 0.9968319335103694
```

```
count_complain_x_edu_lvl = pd.crosstab(index=df_categorical['Complain'], columns=[df_categorical['educationLevel']], rownam
count_complain_x_edu_lvl
```

educationLevel	2n Cycle	Basic	Graduation	Master	PhD	Total
Complain						
False	195	54	1099	362	475	2185
True	3	0	14	2	1	20
Total	198	54	1113	364	476	2205

Next steps: View recommended plots

```
res = chi2_contingency(count_complain_x_edu_lvl)
print("Qui-Quadrado:", res.statistic)
print("p-value:", res.pvalue)
```

```
Qui-Quadrado: 5.923340529293711
p-value: 0.8216621735632803
```

Auto combinações Campaign x Campaign

```
from itertools import combinations
list_contingency_tables = []

column_combinations = list(combinations(cmp_categorical.columns, 2))

# Calcular tabelas de contingência para cada combinação de colunas
for col1, col2 in column_combinations:
    contingency_table = pd.crosstab(index=cmp_categorical[col1], columns=cmp_categorical[col2], margins=True, margins_name=
    list_contingency_tables.append(contingency_table)

list_contingency_tables
```

```
 Campaign 1
 Accepted          45        97    142
 Declined         119      1944   2063
 Total            164      2041   2205,
 Campaign 5  Accepted  Declined  Total
 Campaign 1
 Accepted          68        74    142
 Declined          93      1970   2063
 Total            161      2044   2205,
 Campaign 6  Accepted  Declined  Total
 Campaign 1
 Accepted          79        63    142
 Declined         254      1809   2063
 Total            333      1872   2205,
 Campaign 3  Accepted  Declined  Total
 Campaign 2
 Accepted           7        23     30
 Declined         156      2019   2175
 Total            163      2042   2205,
 Campaign 4  Accepted  Declined  Total
 Campaign 2
 Accepted          22         8     30
 Declined         142      2033   2175
 Total            164      2041   2205,
 Campaign 5  Accepted  Declined  Total
 Campaign 2
 Accepted          17        13     30
 Declined         144      2031   2175
 Total            161      2044   2205,
 Campaign 6  Accepted  Declined  Total
 Campaign 2
 Accepted          20        10     30
 Declined         313      1862   2175
 Total            333      1872   2205,
 Campaign 4  Accepted  Declined  Total
 Campaign 3
 Accepted           0       163    163
 Declined         164      1878   2042
 Total            164      2041   2205,
 Campaign 5  Accepted  Declined  Total
 Campaign 3
 Accepted          24       139    163
 Declined         137      1905   2042
 Total            161      2044   2205,
 Campaign 6  Accepted  Declined  Total
 Campaign 3
 Accepted          77        86    163
 Declined         256      1786   2042
 Total            333      1872   2205,
 Campaign 5  Accepted  Declined  Total
 Campaign 4
 Accepted          59       105    164
 Declined         102      1939   2041
 Total            161      2044   2205,
 Campaign 6  Accepted  Declined  Total
 Campaign 4
 Accepted          62       102    164
 Declined         271      1770   2041
 Total            333      1872   2205,
```

```
# Sua lista de tabelas de contingência
contingency_tables = [
    {'Campaigns': '1_vs_2', 'Table': [[13, 129], [17, 2046]]},
    {'Campaigns': '1_vs_3', 'Table': [[24, 118], [139, 1924]]},
    {'Campaigns': '1_vs_4', 'Table': [[45, 97], [119, 1944]]},
    {'Campaigns': '1_vs_5', 'Table': [[68, 74], [93, 1970]]},
    {'Campaigns': '1_vs_6', 'Table': [[79, 63], [254, 1809]]},
    {'Campaigns': '2_vs_3', 'Table': [[7, 23], [156, 2019]]},
    {'Campaigns': '2_vs_4', 'Table': [[22, 8], [142, 2033]]},
    {'Campaigns': '2_vs_5', 'Table': [[17, 13], [144, 2031]]},
    {'Campaigns': '2_vs_6', 'Table': [[20, 10], [313, 1862]]},
    {'Campaigns': '3_vs_4', 'Table': [[0, 163], [164, 1878]]},
    {'Campaigns': '3_vs_5', 'Table': [[24, 139], [137, 1905]]},
    {'Campaigns': '3_vs_6', 'Table': [[77, 86], [256, 1786]]},
    {'Campaigns': '4_vs_5', 'Table': [[59, 105], [102, 1939]]},
    {'Campaigns': '4_vs_6', 'Table': [[62, 102], [271, 1770]]},
    {'Campaigns': '5_vs_6', 'Table': [[91, 70], [242, 1802]]}
]

results = []

for entry in contingency_tables:
    # Convertendo a lista para uma matriz numpy
    table_np = np.array(entry['Table'])
    # Calculando o teste qui-quadrado e o valor p
    chi2, p_value, _, _ = chi2_contingency(table_np)
    # Adicionando a coluna 'Correlation'
    correlation = 'Correlated' if p_value < 0.05 else 'Not Correlated'
    results.append({'Campaigns': entry['Campaigns'], 'Chi-Square': chi2, 'P-value': p_value, 'Correlation': correlation})

results_df = pd.DataFrame(results)

results_df
```

	Campaigns	Chi-Square	P-value	Correlation
0	1_vs_2	62.639104	2.482747e-15	Correlated
1	1_vs_3	18.589962	1.620717e-05	Correlated
2	1_vs_4	125.932519	3.181188e-29	Correlated
3	1_vs_5	362.983125	6.309928e-81	Correlated
4	1_vs_6	191.107224	1.822433e-43	Correlated
5	2_vs_3	9.052296	2.623652e-03	Correlated
6	2_vs_4	182.248340	1.565067e-41	Correlated
7	2_vs_5	102.232472	4.937391e-24	Correlated
8	2_vs_6	59.061336	1.528330e-14	Correlated
9	3_vs_4	13.000458	3.114147e-04	Correlated
10	3_vs_5	13.166574	2.849873e-04	Correlated
11	3_vs_6	139.089324	4.210775e-32	Correlated
12	4_vs_5	210.674675	9.787502e-48	Correlated
13	4_vs_6	69.325569	8.348030e-17	Correlated
14	5_vs_6	228.927232	1.021640e-51	Correlated

Next steps: View recommended plots

Categorical x Numerical

Analizing the amount spent in products in the last 2 years and its distribution across the categorical variables maritalStatus / educationLevel / Complain:

Categorical K>2 : maritalStatus, educationLevel

Kruskal-Wallis test (K>2) | MaritalStatus and Education Level x Numerical Variables

```
mnt_spnt_prds = df[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'Mnt
```

```
import pandas as pd
import scipy
from itertools import product
from scipy.stats import kruskal

def kruskal_wallis_test(df, categorical_vars, numerical_vars):
    results = []

    for cat_var, num_var in product(categorical_vars, numerical_vars):
        groups = []
        for group, data in df.groupby(cat_var):
            groups.append(data[num_var])

        # Perform Kruskal-Wallis test
        stat, p_value = scipy.stats.kruskal(*groups)
        # Add a column indicating association based on p-value
        association = 'Associated' if p_value < 0.05 else 'Not associated'
        results.append({'Categorical Variable': cat_var, 'Numerical Variable': num_var, 'Statistic': stat, 'P-value': p_val

    test_results = pd.DataFrame(results)
    test_results = test_results.sort_values(by='P-value')

    return test_results

kover2_cat = df_categorical[['maritalStatus','educationLevel']]

krusk_df = kruskal_wallis_test(df, kover2_cat, df_numerical)
krusk_df
```

	Categorical Variable	Numerical Variable	Statistic	P-value	Is Group Associated?
20	educationLevel	MntWines	205.144383	2.942432e-43	Associated
31	educationLevel	Income	140.002639	2.818900e-29	Associated
35	educationLevel	MntRegularProds	119.664537	6.299556e-25	Associated
13	maritalStatus	Age	99.025820	1.585692e-20	Associated
22	educationLevel	MntMeatProducts	97.498064	3.352347e-20	Associated
33	educationLevel	Age	89.548469	1.642170e-18	Associated
34	educationLevel	MntTotal	87.181501	5.224231e-18	Associated
24	educationLevel	MntSweetProducts	74.762165	2.237275e-15	Associated
23	educationLevel	MntFishProducts	73.648564	3.847585e-15	Associated
29	educationLevel	NumStorePurchases	71.777040	9.565442e-15	Associated
21	educationLevel	MntFruits	69.398399	3.040798e-14	Associated
25	educationLevel	MntGoldProds	65.532434	1.987445e-13	Associated
27	educationLevel	NumWebPurchases	60.513955	2.262084e-12	Associated
28	educationLevel	NumCatalogPurchases	55.231919	2.905023e-11	Associated
38	educationLevel	Teenhome	53.907004	5.504007e-11	Associated
18	maritalStatus	Teenhome	32.373905	1.604334e-06	Associated
30	educationLevel	NumWebVisitsMonth	28.410744	1.029633e-05	Associated
37	educationLevel	Kidhome	13.112661	1.073823e-02	Associated
39	educationLevel	Customer_Days	12.947385	1.153576e-02	Associated
17	maritalStatus	Kidhome	12.643445	1.315601e-02	Associated
5	maritalStatus	MntGoldProds	9.035275	6.022381e-02	Not associated
6	maritalStatus	NumDealsPurchases	7.620047	1.065308e-01	Not associated
7	maritalStatus	NumWebPurchases	7.615049	1.067418e-01	Not associated
8	maritalStatus	NumCatalogPurchases	7.456202	1.136562e-01	Not associated
0	maritalStatus	MntWines	7.145067	1.284178e-01	Not associated
36	educationLevel	AcceptedCmpOverall	6.852647	1.438831e-01	Not associated
11	maritalStatus	Income	5.983661	2.003718e-01	Not associated
15	maritalStatus	MntRegularProds	5.581177	2.326856e-01	Not associated
14	maritalStatus	MntTotal	5.220184	2.654420e-01	Not associated
9	maritalStatus	NumStorePurchases	4.260477	3.719004e-01	Not associated
3	maritalStatus	MntFishProducts	4.254075	3.727112e-01	Not associated
26	educationLevel	NumDealsPurchases	4.085517	3.945560e-01	Not associated
10	maritalStatus	NumWebVisitsMonth	3.784389	4.359712e-01	Not associated
1	maritalStatus	MntFruits	3.223624	5.211238e-01	Not associated
2	maritalStatus	MntMeatProducts	3.154049	5.323846e-01	Not associated
32	educationLevel	Recency	2.553368	6.351056e-01	Not associated
12	maritalStatus	Recency	1.556823	8.165305e-01	Not associated
19	maritalStatus	Customer_Days	0.896018	9.251314e-01	Not associated
16	maritalStatus	AcceptedCmpOverall	0.839064	9.331356e-01	Not associated
4	maritalStatus	MntSweetProducts	0.734835	9.469646e-01	Not associated

Next steps: View recommended plots

Numerical variables that has association with educationLevel

```
list_num_edu_lvl_associated = krusk_df[(krusk_df['Categorical Variable'] == 'educationLevel') & (krusk_df['Is Group Associa
list_num_edu_lvl_associated
```

```
['MntWines',
 'Income',
 'MntRegularProds',
 'MntMeatProducts',
 'Age',
```

```
       'MntTotal',
       'MntSweetProducts',
       'MntFishProducts',
       'NumStorePurchases',
       'MntFruits',
       'MntGoldProds',
       'NumWebPurchases',
       'NumCatalogPurchases',
       'Teenhome',
       'NumWebVisitsMonth',
       'Kidhome',
       'Customer_Days']
```

Numerical variables that has association with maritalStatus

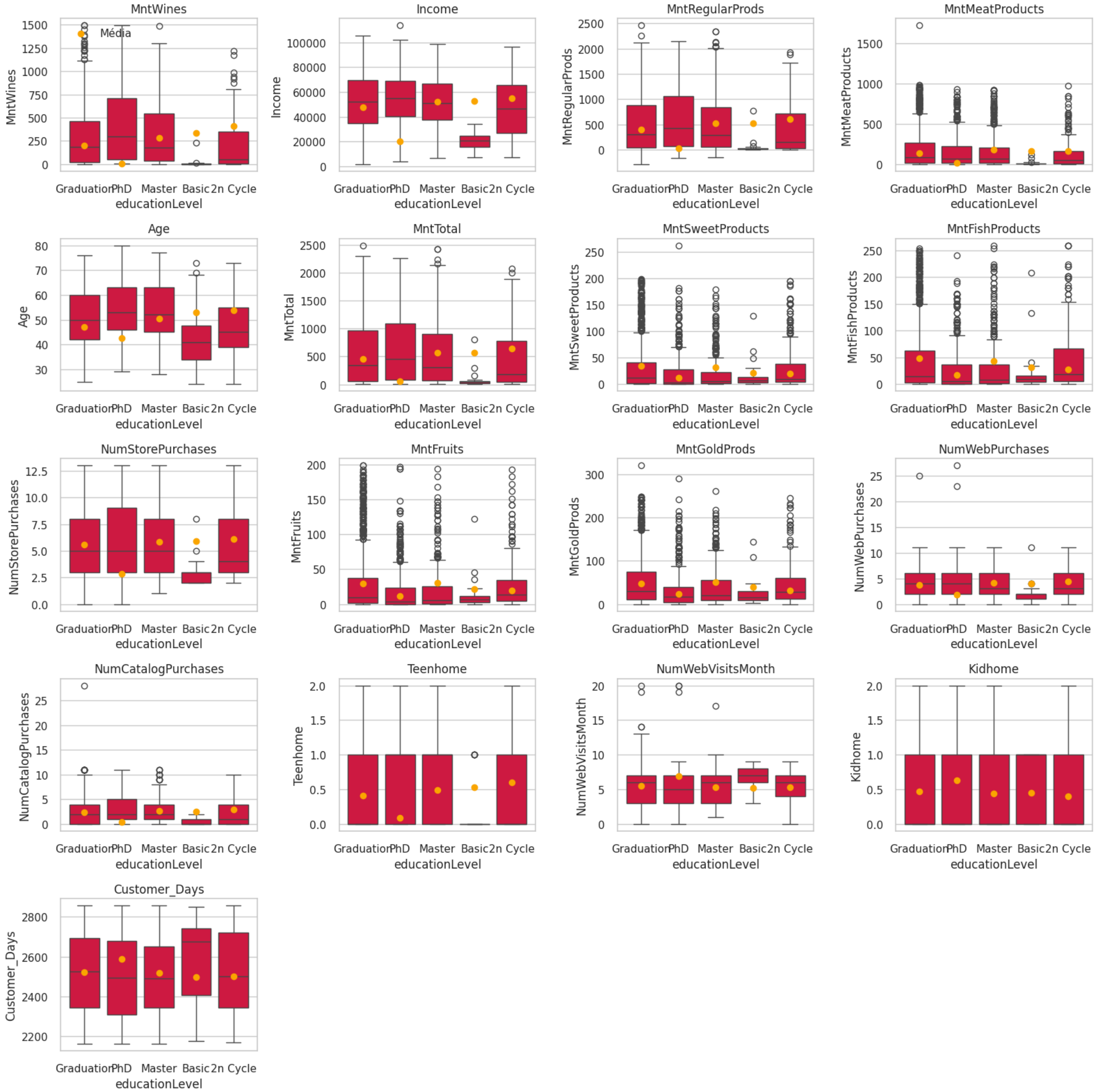
```
list_num_marit_status_associated = krusk_df[(krusk_df['Categorical Variable'] == 'maritalStatus') & (krusk_df['Is Group Ass
list_num_marit_status_associated
```

```
['Age', 'Teenhome', 'Kidhome']
```

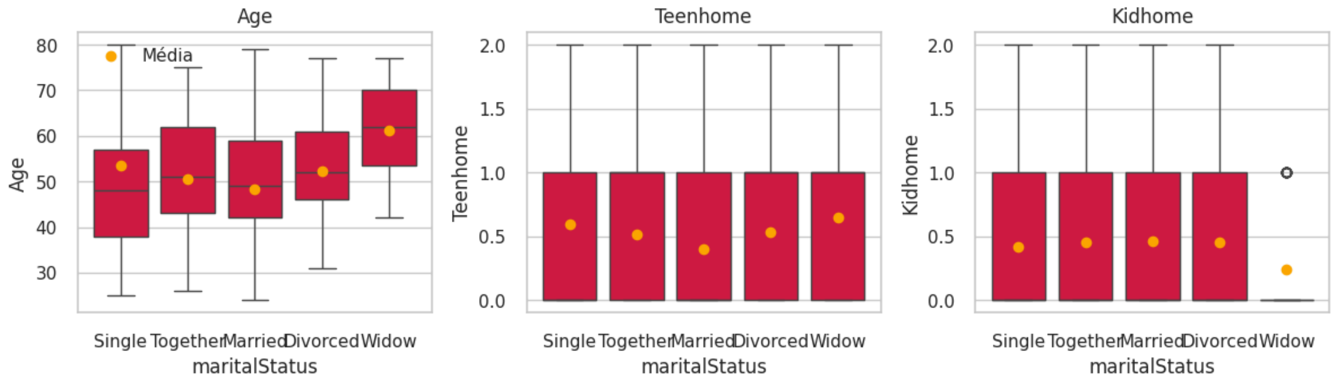
```
def box_plot_list_num_x_categorical(cat_var, list_of_numerical):
  plt.figure(figsize=(16, 16))  # Ajuste o tamanho conforme necessário
  sns.set(style="whitegrid")

  num_values = df[list_of_numerical]
  cat_values = df[cat_var]


  for i, product in enumerate(num_values, 1):
      plt.subplot(5, 4, i)  # Ajuste para a quantidade correta de subplots, 3 linhas e até 7 colunas aqui
      sns.boxplot(x=cat_values,y=num_values[product], color='#EA0031')

      # Calculate mean for each category
      means = num_values.groupby(cat_values)[product].mean()

      # Plot mean for each category
      x_values = range(len(means))
      plt.scatter(x_values, means, color='#FEA500', label='Média', zorder=5)


      if i == 1:
          plt.legend(loc="upper left")
      plt.title(product)

  plt.tight_layout()
  plt.show()
```

Box-Plot K>2 and Associated | Education Level x Num

```
box_plot_list_num_x_categorical('educationLevel',list_num_edu_lvl_associated)
```


MntWines
Média
Income
MntRegularProds
MntMeatProducts
Age
MntTotal
MntSweetProducts
MntFishProducts
NumStorePurchases
MntFruits
MntGoldProds
NumWebPurchases
NumCatalogPurchases
Teenhome
NumWebVisitsMonth
Kidhome
Customer_Days
Graduation PhD Master Basic 2n Cycle
educationLevel

```
box_plot_list_num_x_categorical('maritalStatus',list_num_marit_status_associated)
```



Categorical K=2 : Campaign N, Complain

Mann-Whitney U test (K=2) | Campaign N, Complain x Numerical Variables

```
import pandas as pd
import scipy.stats as stats
from itertools import product

def mann_whitney_u_test(df, categorical_vars, numerical_vars):
    results = []

    for cat_var, num_var in product(categorical_vars, numerical_vars):
        groups = []
        for group, data in df.groupby(cat_var):
            groups.append(data[num_var])

        # Perform Mann-Whitney U test
        stat, p_value = stats.mannwhitneyu(*groups)

        # Add a column indicating association based on p-value
        association = 'Associated' if p_value < 0.05 else 'Not associated'

        results.append({'Categorical Variable': cat_var, 'Numerical Variable': num_var, 'Statistic': stat, 'P-value': p_val

    test_results = pd.DataFrame(results)
    test_results = test_results.sort_values(by='P-value')

    return test_results

categorical_vars = ['Campaign 1', 'Campaign 2', 'Campaign 3', 'Campaign 4', 'Campaign 5', 'Campaign 6', 'Complain']  # List
num_without_cmp_accpt = df_numerical.drop('AcceptedCmpOverall', axis=1,)
numerical_vars = num_without_cmp_accpt.columns.tolist()  # List of numerical variables from df_numerical


mannwhitneyu_results = mann_whitney_u_test(df, categorical_vars, numerical_vars)
mannwhitneyu_results
```

	Categorical Variable	Numerical Variable	Statistic	P-value	Is Group Associated?
87	Campaign 5	Income	310193.5	3.031723e-78	Associated
90	Campaign 5	MntTotal	304735.0	1.251735e-72	Associated
91	Campaign 5	MntRegularProds	304146.5	4.902812e-72	Associated
76	Campaign 5	MntWines	298173.0	3.682504e-66	Associated
11	Campaign 1	Income	261680.5	1.535550e-55	Associated
...	...	...	...	...	...
31	Campaign 2	Recency	32377.5	9.431420e-01	Not associated
23	Campaign 2	MntSweetProducts	32412.5	9.510126e-01	Not associated
127	Complain	Age	21730.0	9.663588e-01	Not associated
88	Campaign 5	Recency	164856.5	9.677959e-01	Not associated
22	Campaign 2	MntFishProducts	32509.0	9.733182e-01	Not associated

133 rows × 5 columns

Next steps: View recommended plots

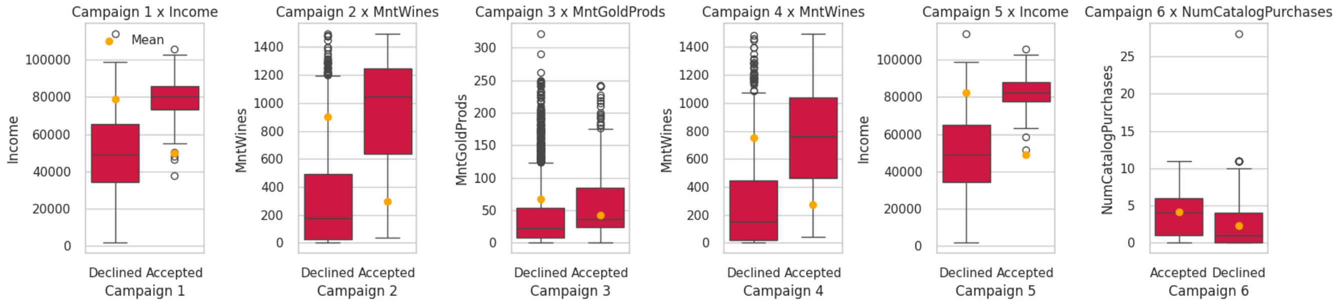
```
highest_p_value_rows = mannwhitneyu_results.loc[mannwhitneyu_results.groupby('Categorical Variable')['P-value'].idxmin()]
highest_p_value_rows_associated = highest_p_value_rows[highest_p_value_rows['Is Group Associated?'] == 'Associated']
highest_p_value_rows_associated
```

	Categorical Variable	Numerical Variable	Statistic	P-value	Is Group Associated?
11	Campaign 1	Income	261680.5	1.535550e-55	Associated
19	Campaign 2	MntWines	55205.0	7.049267e-11	Associated
43	Campaign 3	MntGoldProds	217426.5	6.975071e-11	Associated
57	Campaign 4	MntWines	281912.0	2.681020e-48	Associated
87	Campaign 5	Income	310193.5	3.031723e-78	Associated
103	Campaign 6	NumCatalogPurchases	426970.0	7.035959e-28	Associated

Next steps: View recommended plots

Box-Plot K=2 and Associated | Campaing N x Most associated numerica varaibles

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(16, 4))  # Adjust the size as needed
sns.set(style="whitegrid")

# Define the number of rows and columns for subplots
num_plots = len(highest_p_value_rows_associated)
num_cols = 6  # Assuming you want 6 columns
num_rows = -(-num_plots // num_cols)  # Ceiling division to calculate the number of rows

for i, (_, row) in enumerate(highest_p_value_rows_associated.iterrows(), 1):
    plt.subplot(num_rows, num_cols, i)  # Adjust for the correct number of subplots
    num_value = df[row['Numerical Variable']]
    cat_value = df[row['Categorical Variable']]

    # Plot box plot
    sns.boxplot(x=cat_value, y=num_value, color='#EA0031')

    # Calculate mean for each category
    means = num_value.groupby(cat_value).mean()

    # Plot mean for each category
    x_values = range(len(means))
    plt.scatter(x_values, means, color='#FEA500', label='Mean', zorder=5)

    plt.title(row['Categorical Variable'] + " x " + row['Numerical Variable'])
    if i == 1:
        plt.legend(loc="upper left")

plt.tight_layout()
plt.show()
```




```
list_num_cmp1_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Campaign 1') & (mannwhitn
list_num_cmp1_associated
```

```
['Income',
 'MntTotal',
 'MntRegularProds',
 'MntWines',
 'NumCatalogPurchases',
 'MntMeatProducts',
 'MntSweetProducts',
 'MntFishProducts',
 'NumWebVisitsMonth',
 'NumStorePurchases',
 'NumWebPurchases',
 'Kidhome',
 'NumDealsPurchases',
 'MntGoldProds',
 'MntFruits',
 'Teenhome']
```

```
list_num_cmp2_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Campaign 2') & (mannwhitn
list_num_cmp2_associated
```

```
['MntWines',
 'MntTotal',
 'MntRegularProds',
 'Income',
 'NumCatalogPurchases',
 'Kidhome',
 'NumStorePurchases',
 'MntGoldProds',
 'MntMeatProducts',
 'NumDealsPurchases']
```

```
list_num_cmp3_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Campaign 3') & (mannwhitn
list_num_cmp3_associated
```

```
['MntGoldProds',
 'NumCatalogPurchases',
 'NumStorePurchases',
 'Age',
 'NumWebVisitsMonth',
 'Teenhome']
```

```
list_num_cmp4_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Campaign 4') & (mannwhitn
list_num_cmp4_associated
```

```
['MntWines',
 'MntRegularProds',
 'MntTotal',
 'Income',
 'NumStorePurchases',
 'NumCatalogPurchases',
 'NumWebPurchases',
 'Kidhome',
 'MntMeatProducts',
 'Age',
 'MntGoldProds']
```

```
list_num_cmp5_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Campaign 5') & (mannwhitn
list_num_cmp5_associated
```

```
['Income',
 'MntTotal',
 'MntRegularProds',
 'MntWines',
 'MntMeatProducts',
 'NumCatalogPurchases',
 'NumWebVisitsMonth',
 'NumDealsPurchases',
 'MntSweetProducts',
 'MntFruits',
 'NumStorePurchases',
 'MntFishProducts',
 'Kidhome',
 'Teenhome',
 'MntGoldProds',
 'NumWebPurchases']
```

```
list_num_cmp6_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Campaign 6') & (mannwhitn
list_num_cmp6_associated
```

```
['NumCatalogPurchases',
 'MntTotal',
 'MntMeatProducts',
 'MntRegularProds',
 'Recency',
 'MntWines',
 'Customer_Days',
 'MntGoldProds',
 'NumWebPurchases',
 'Income',
 'Teenhome',
 'MntFruits',
 'MntSweetProducts',
 'MntFishProducts',
 'Kidhome',
 'NumStorePurchases']
```

```
list_num_complain_associated = mannwhitneyu_results[(mannwhitneyu_results['Categorical Variable'] == 'Complain') & (mannwhi
list_num_complain_associated
```

```
[]
```

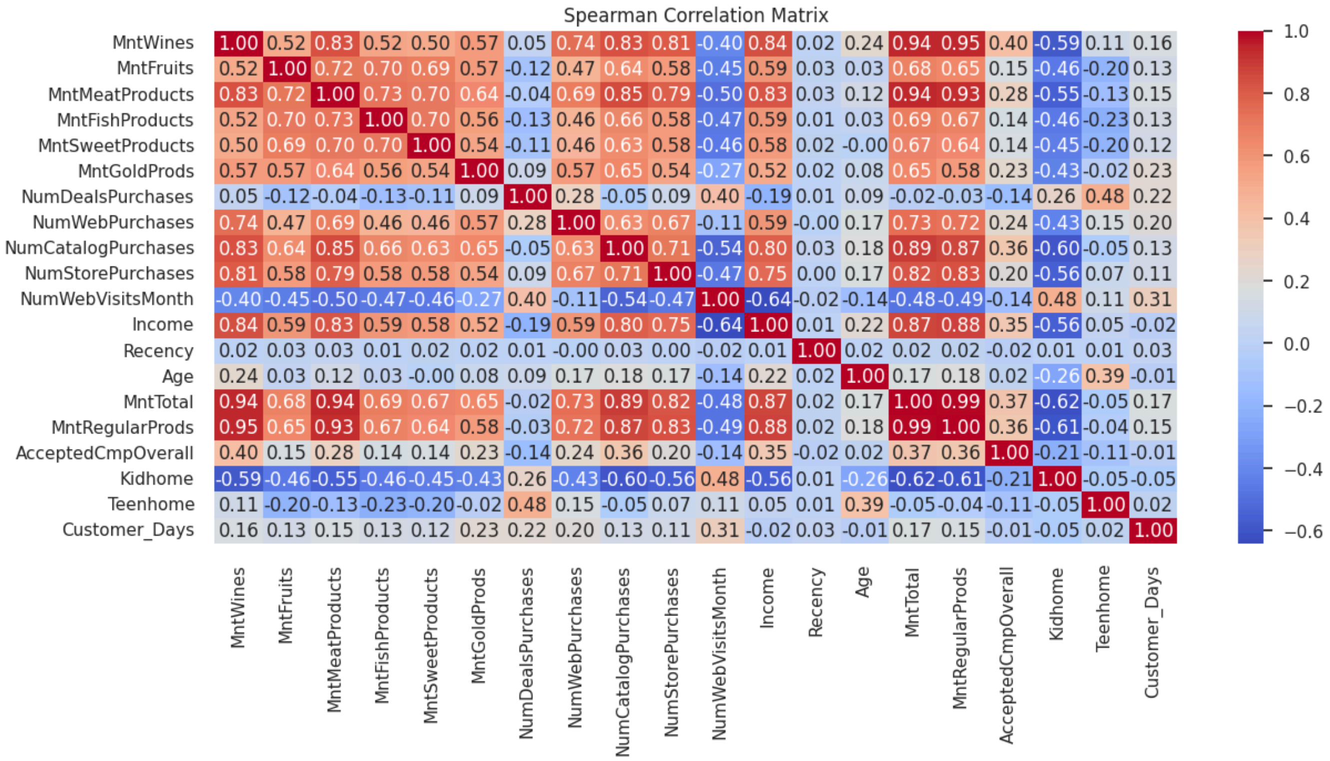
Analizing the purchases in the last 2 years and its distribution across the categorical variables maritalStatus / educationLevel:

Numerical x Numerical

Correlation Matrix

Defining key correlations in business perspectives:

```
spearman_corr = df_numerical.corr(method='spearman')

# Criar o heatmap
plt.figure(figsize=(14, 6))
sns.heatmap(spearman_corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Spearman Correlation Matrix')
plt.show()
```



```
df_numerical.columns
```

```
Index(['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth', 'Income', 'Recency', 'Age', 'MntTotal',
       'MntRegularProds', 'AcceptedCmpOverall', 'Kidhome', 'Teenhome',
       'Customer_Days'],
      dtype='object')
```

Correlation Evaluation