Bibliotecas

In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from IPython.display import Image
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from functools import reduce
import pickle
import warnings
warnings.filterwarnings("ignore")
```

Carregamento dos Dados

In [2]:
```
df = pd.read_csv('marketing_campaign.csv', delimiter='\t')
```

In [3]:
```
df
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custome
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-201
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-201
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-201
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-201
4	5324	1981	PhD	Married	58293.0	1	0	19-01-201
...	...	...	...	...	...	...	...	.
2235	10870	1967	Graduation	Married	61223.0	0	1	13-06-201
2236	4001	1946	PhD	Together	64014.0	2	1	10-06-201
2237	7270	1981	Graduation	Divorced	56981.0	0	0	25-01-201
2238	8235	1956	Master	Together	69245.0	0	1	24-01-201
2239	9405	1954	PhD	Married	52869.0	1	1	15-10-201

2240 rows × 29 columns

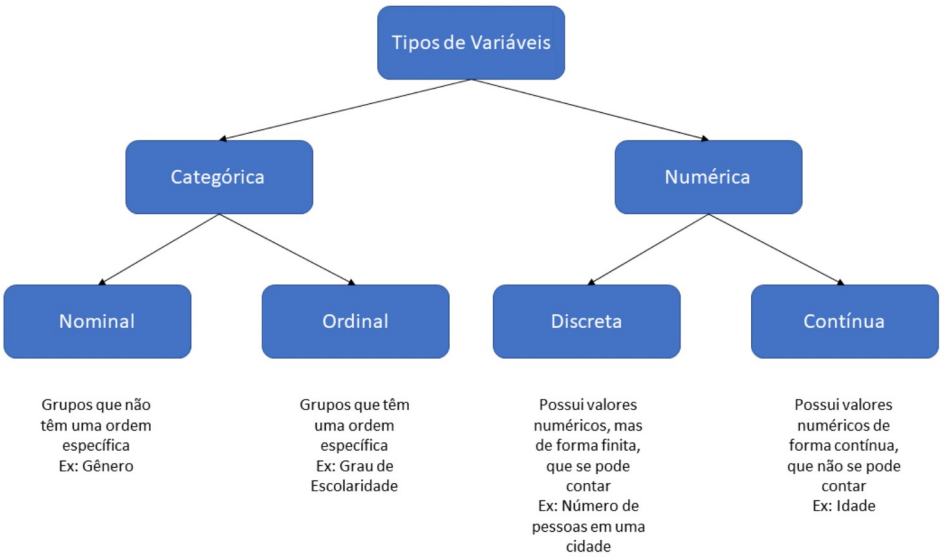
Análise Exploratória e Limpeza

In [4]:
```
df.columns
```

Out[4]:
```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
      dtype='object')
```

In [5]:
```
Image(filename='Tipos de Variáveis.png', width=1000, height=1000)
```

Out[5]:



In [6]:
```
def show_null(df):
    null_columns = (df.isnull().sum(axis=0)/len(df)).sort_values(ascending=False

    null_data = pd.concat([df.isnull().sum(axis=0),
                           (df.isnull().sum(axis=0)/len(df)).sort_values(ascendi
                           df.loc[:, df.columns.isin(list(null_columns))].dtypes
                          axis=1)

    null_data = null_data.rename(columns={0: '#',
                                          1: '% null',
                                          2: 'type'}).sort_values(ascending=Fals
                                                                  by='% null')

    return null_data
```

In [7]:
```
def plot_num(df, feature):
    fig = plt.figure(figsize=(12,8))
    gs = GridSpec(1,2)
    sns.boxplot(y=feature, data=df, color='firebrick', ax=fig.add_subplot(gs[0,0
    plt.ylabel(feature)
    sns.stripplot(y=df[feature], color='darkcyan', ax=fig.add_subplot(gs[0,1]))
    plt.ylabel(None)
    plt.show()
```

In [8]:
```
def plot_cat(df, feature):
    fig = plt.figure(figsize=(12,8))
    ax = sns.countplot(data=df, x=feature)
    for p in ax.patches:

        ax.annotate(f'\n{p.get_height()}',
                    (p.get_x()+0.4, p.get_height()+5),
                    ha='center',
                    color='black')

    plt.xlabel('\n' + feature)
    plt.ylabel('Quantidade de Clientes')
    plt.show()
```

People

ID: Customer's unique identifier Year_Birth: Customer's birth year Education: Customer's education level Marital_Status: Customer's marital status Income: Customer's yearly household income Kidhome: Number of children in customer's household Teenhome: Number of teenagers in customer's household Dt_Customer: Date of customer's enrollment with the company Recency: Number of days since customer's last purchase Complain: 1 if the customer complained in the last 2 years, 0 otherwise

In [9]:
```
people = df[['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidho
            'Recency', 'Complain']]
```

In [10]:
```
people.head()
```

Out[10]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Re
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	

In [11]:
```
show_null(people)
```

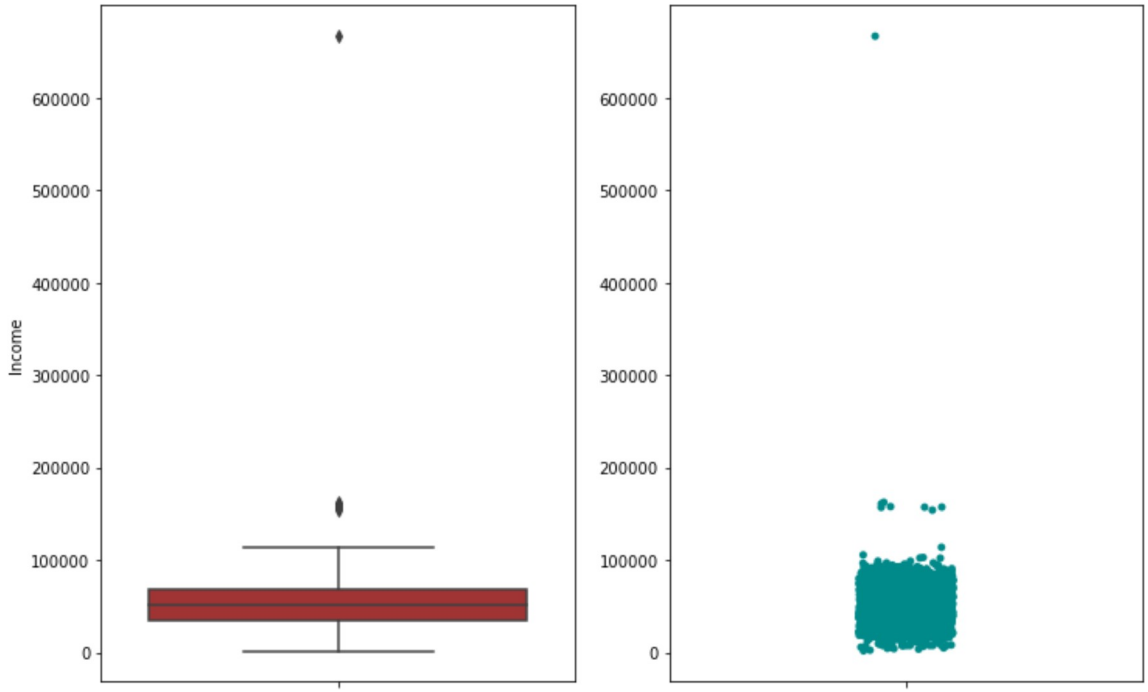
Out[11]:

	#	% null	type
Income	24	0.010714	float64
ID	0	0.000000	int64
Year_Birth	0	0.000000	int64
Education	0	0.000000	object
Marital_Status	0	0.000000	object
Kidhome	0	0.000000	int64
Teenhome	0	0.000000	int64
Dt_Customer	0	0.000000	object
Recency	0	0.000000	int64
Complain	0	0.000000	int64

Income

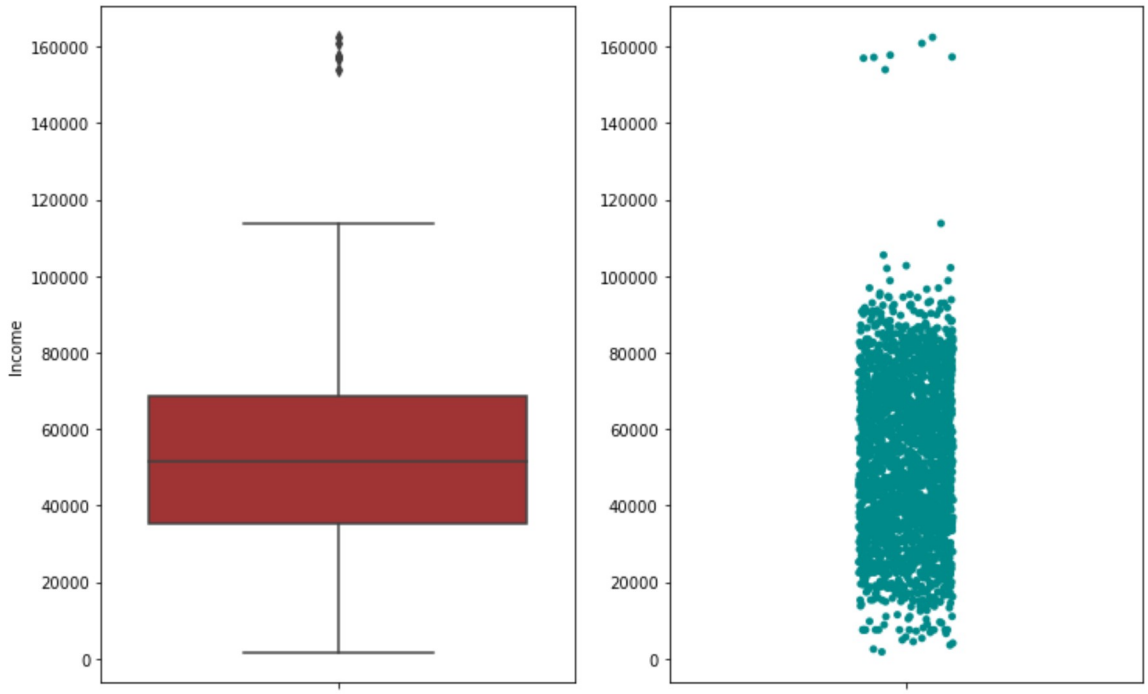
In [12]:
```
people.dropna(inplace=True)
```

In [13]:
```
plot_num(people, 'Income')
```



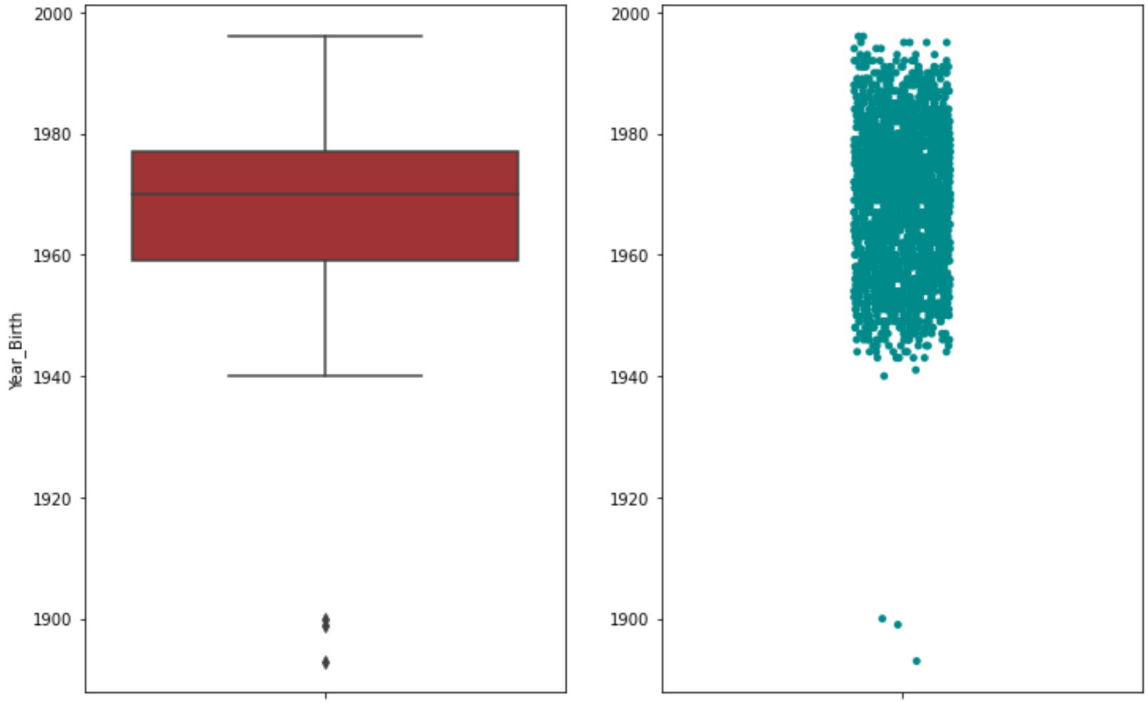
In [14]:
```
people = people[people.Income < 200000]
```

In [15]:
```
plot_num(people, 'Income')
```



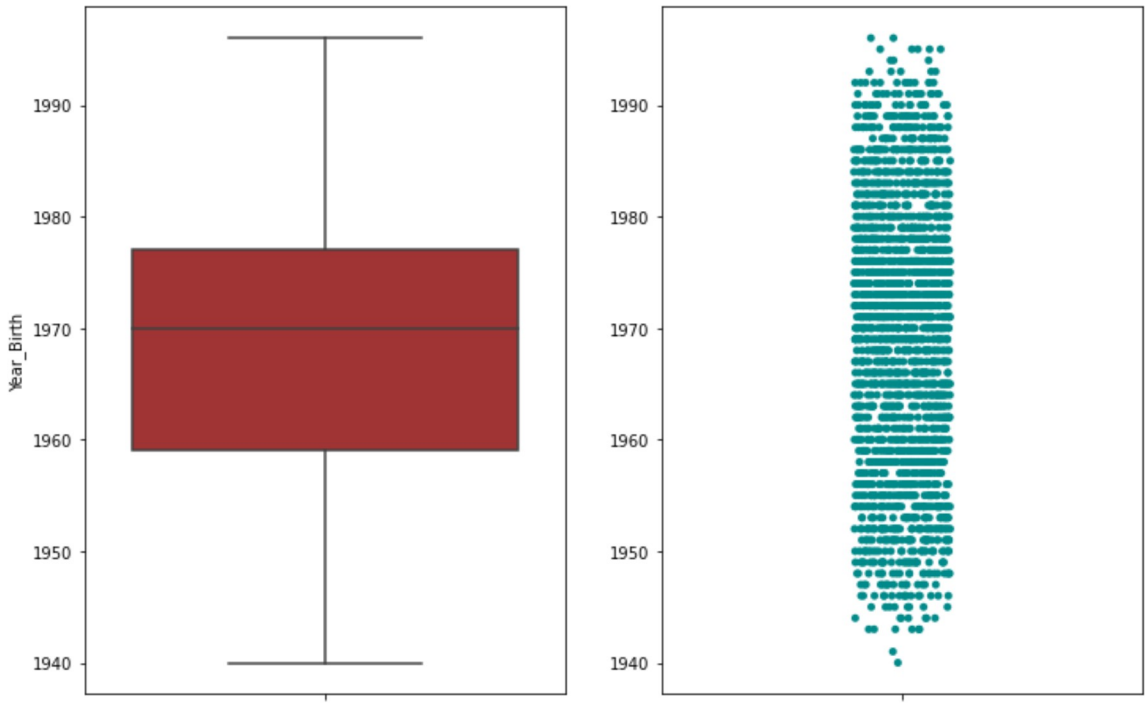
Year_Birth

In [16]:
```
plot_num(people, 'Year_Birth')
```



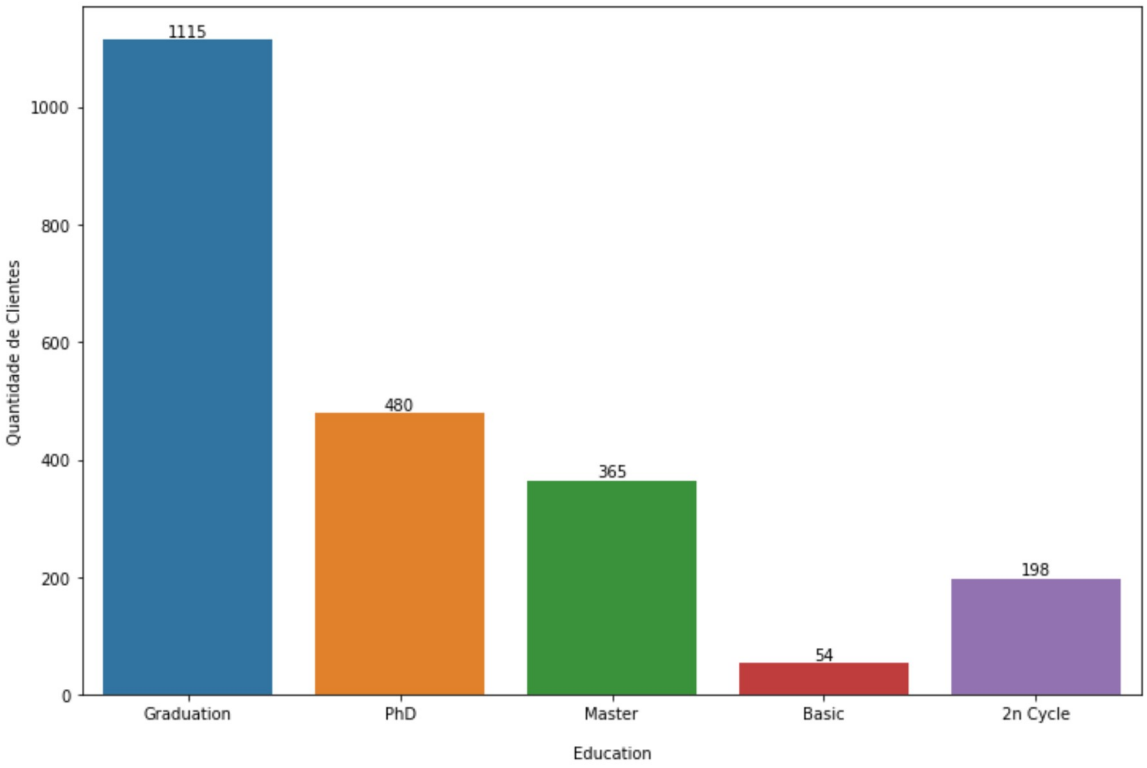
In [17]:
```
people = people[people.Year_Birth > 1930]
```

In [18]:
```
plot_num(people, 'Year_Birth')
```



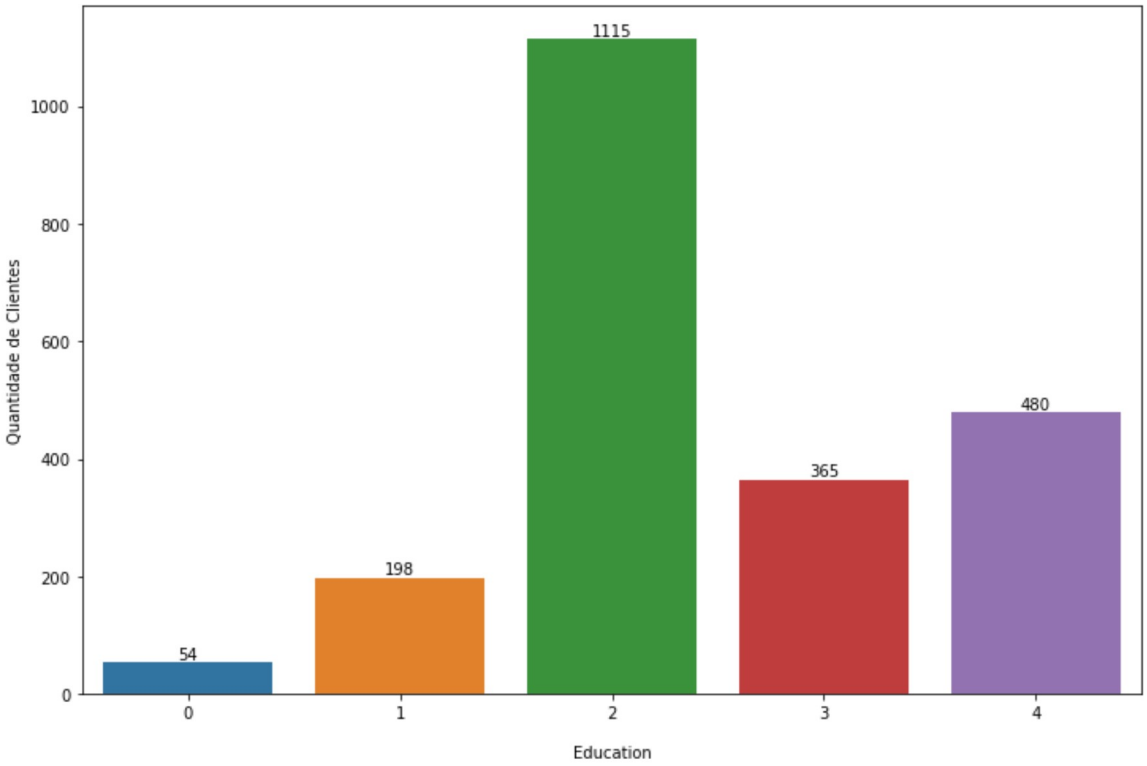
Education

In [19]:
```
plot_cat(people, 'Education')
```



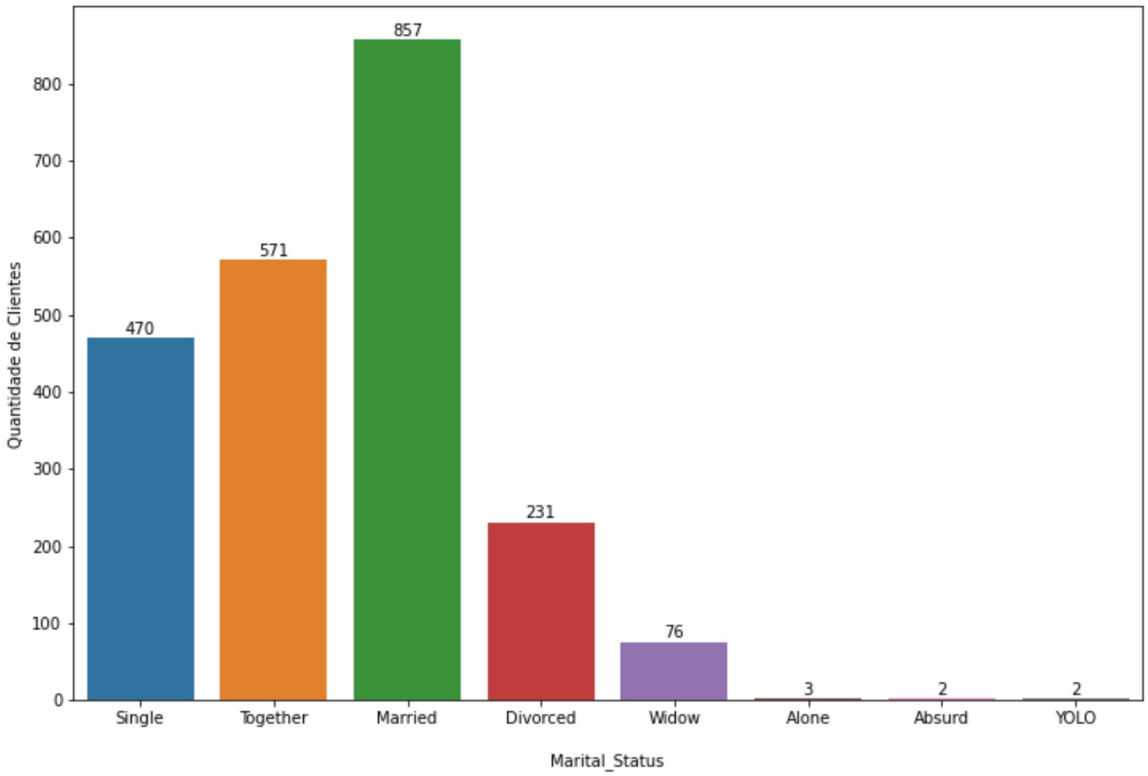
In [20]:
```
people.Education = people.Education.apply(lambda x: 0 if x=='Basic'
                        else(1 if x=='2n Cycle'
                             else(2 if x=='Graduation'
                                  else(3 if x =='Master'
                                       else(4 if x=='PhD' else None)))))
```

In [21]:
```
plot_cat(people, 'Education')
```



Marital_Status

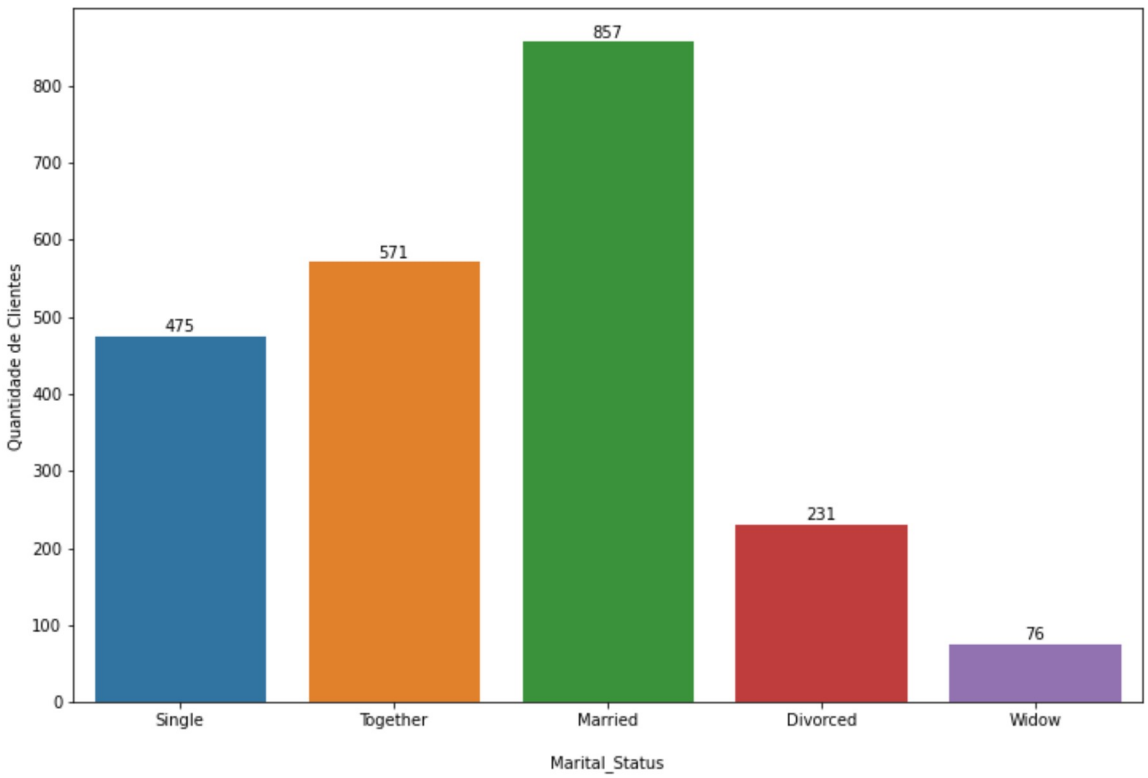
In [22]:
```
plot_cat(people, 'Marital_Status')
```



In [23]:
```
people = people[people.Marital_Status != 'Absurd']
```

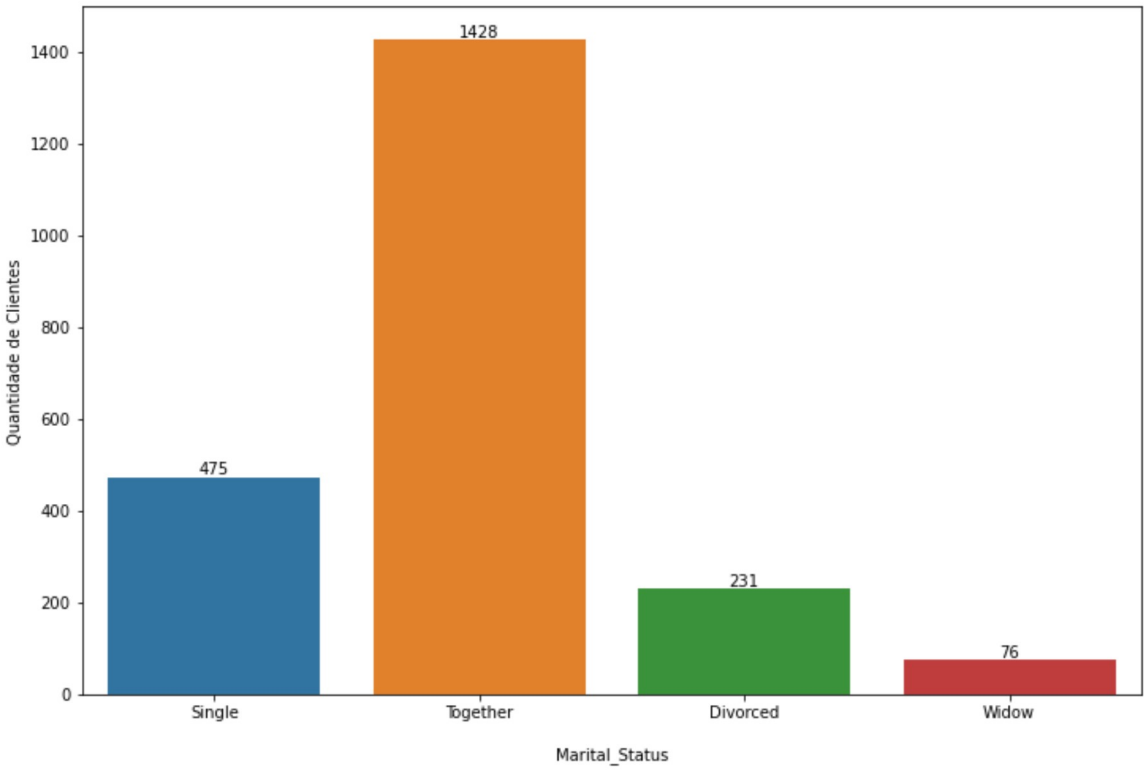
In [24]:
```
people.Marital_Status = people.Marital_Status.apply(lambda x: 'Single' if x=='Al
                                                    else('Single' if x =='YOLO'
```

In [25]:
```
plot_cat(people, 'Marital_Status')
```



In [26]:
```
people.Marital_Status = people.Marital_Status.apply(lambda x: 'Together' if x=='
```

In [27]:
```
plot_cat(people, 'Marital_Status')
```



In [28]:
```
people.dtypes
```

Out[28]:
```
ID                   int64
Year_Birth           int64
Education            int64
Marital_Status      object
Income             float64
Kidhome              int64
Teenhome             int64
Dt_Customer         object
Recency              int64
Complain             int64
dtype: object
```

In [29]:
```
people.Dt_Customer = pd.to_datetime(people.Dt_Customer, dayfirst=True)
```

In [30]:
```
people = pd.get_dummies(people)
```

In [31]:
```
people
```

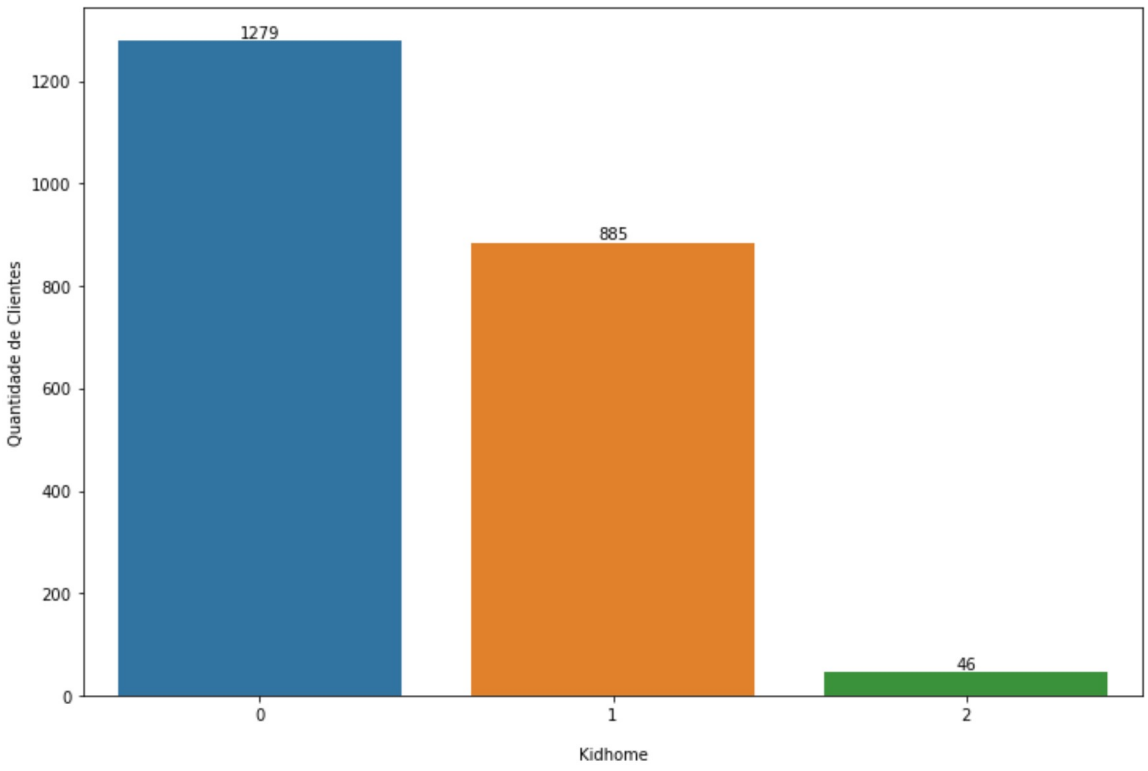
Out[31]:

	ID	Year_Birth	Education	Income	Kidhome	Teenhome	Dt_Customer	Recency	Con
0	5524	1957	2	58138.0	0	0	2012-09-04	58	
1	2174	1954	2	46344.0	1	1	2014-03-08	38	
2	4141	1965	2	71613.0	0	0	2013-08-21	26	
3	6182	1984	2	26646.0	1	0	2014-02-10	26	
4	5324	1981	4	58293.0	1	0	2014-01-19	94	
...	...	...	...	...	...	...	...	...	
2235	10870	1967	2	61223.0	0	1	2013-06-13	46	
2236	4001	1946	4	64014.0	2	1	2014-06-10	56	
2237	7270	1981	2	56981.0	0	0	2014-01-25	91	
2238	8235	1956	3	69245.0	0	1	2014-01-24	8	
2239	9405	1954	4	52869.0	1	1	2012-10-15	40	

2210 rows × 13 columns

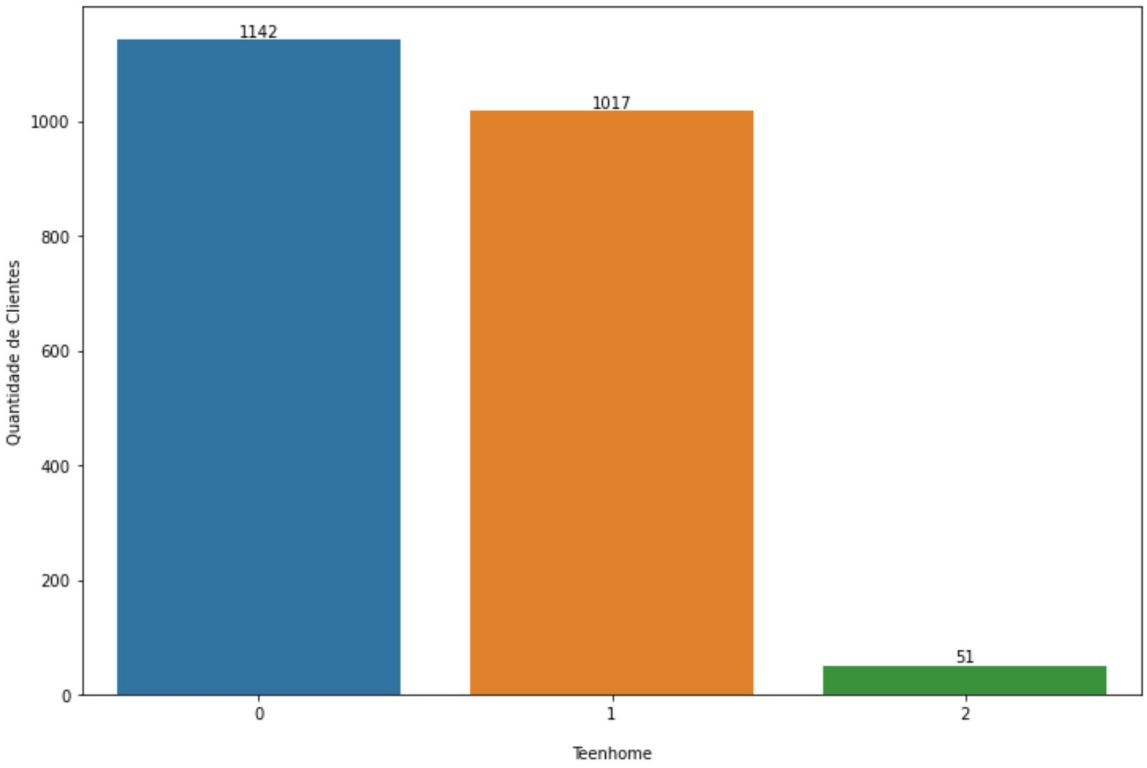
Kidhome

In [32]:
```
plot_cat(people, 'Kidhome')
```



TeenHome

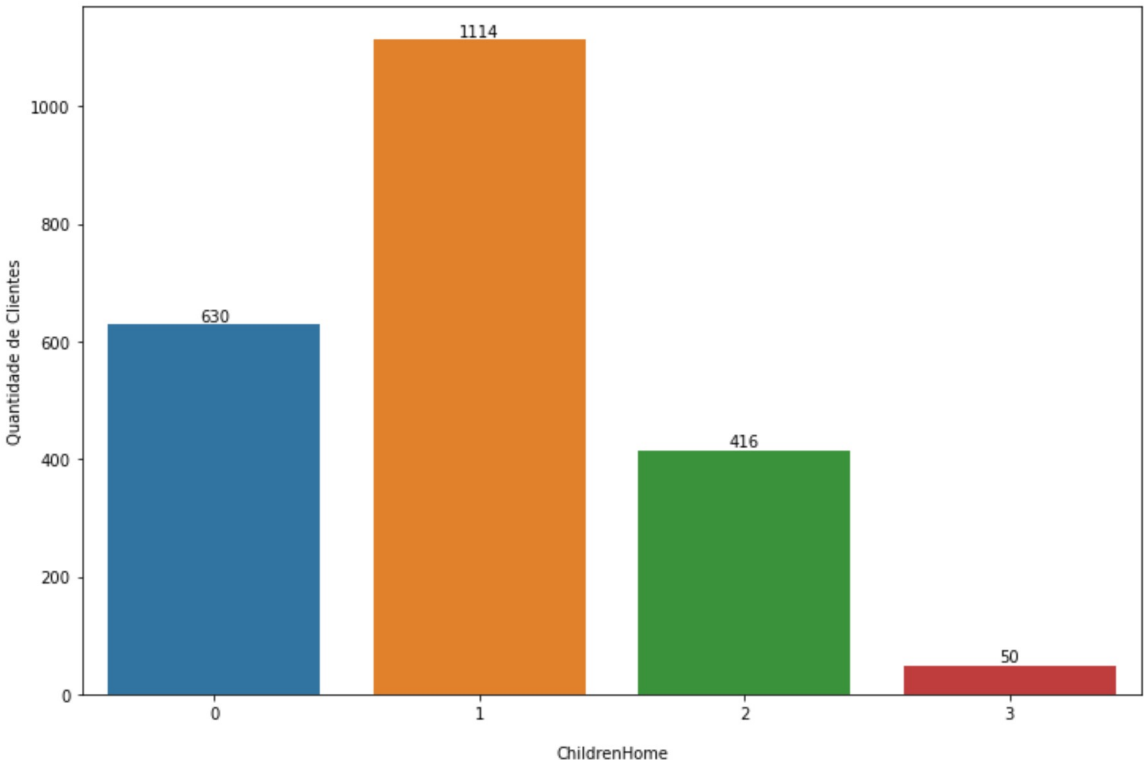
In [33]:
```
plot_cat(people, 'Teenhome')
```



ChildrenHome

In [34]:
```
people['ChildrenHome'] = people.Kidhome + people.Teenhome
```

In [35]:
```
plot_cat(people, 'ChildrenHome')
```



In [36]:
```
people.drop(columns=['Kidhome', 'Teenhome'], inplace=True)
```

In [37]:
```
people.head()
```

Out[37]:

	ID	Year_Birth	Education	Income	Dt_Customer	Recency	Complain	Marital_Status_Divor
0	5524	1957	2	58138.0	2012-09-04	58	0	
1	2174	1954	2	46344.0	2014-03-08	38	0	
2	4141	1965	2	71613.0	2013-08-21	26	0	
3	6182	1984	2	26646.0	2014-02-10	26	0	
4	5324	1981	4	58293.0	2014-01-19	94	0	

Dt_Customer

In [38]:
```
people.Dt_Customer.min()
```

Out[38]:
```
Timestamp('2012-07-30 00:00:00')
```

In [39]:
```
people.Dt_Customer.max()
```

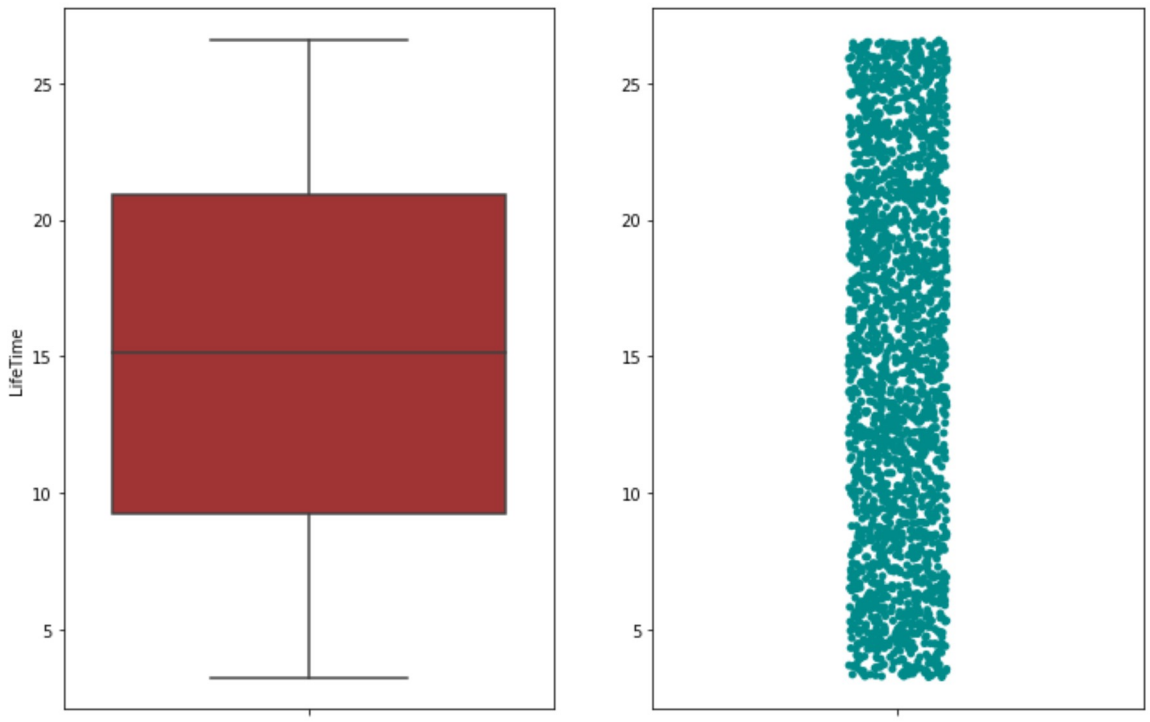
Out[39]:
```
Timestamp('2014-06-29 00:00:00')
```

In [40]:
```
(people['Dt_Customer'] + pd.to_timedelta(people['Recency'], unit='d')).max()
```

Out[40]:
```
Timestamp('2014-10-04 00:00:00')
```

In [41]:
```
people['LifeTime'] = (pd.to_datetime('2014-10-05') - people['Dt_Customer']).dt.d
```

In [42]:
```
plot_num(people, 'LifeTime')
```

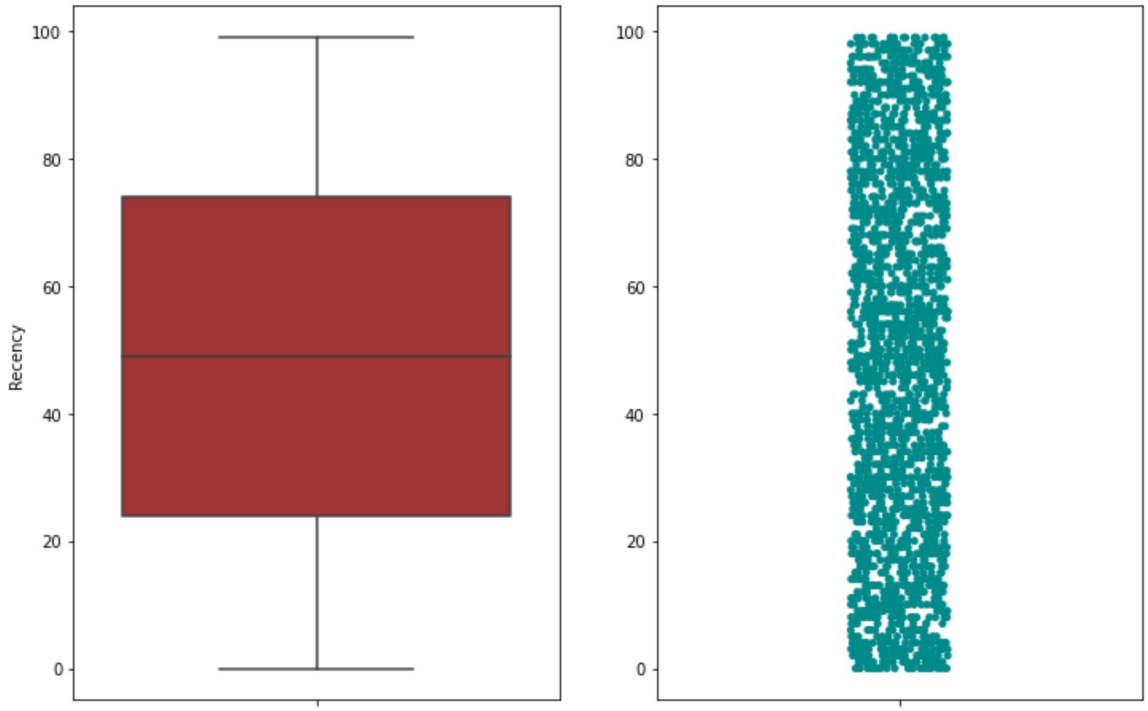


In [43]:
```
people.columns
```

Out[43]:
```
Index(['ID', 'Year_Birth', 'Education', 'Income', 'Dt_Customer', 'Recency',
       'Complain', 'Marital_Status_Divorced', 'Marital_Status_Single',
       'Marital_Status_Together', 'Marital_Status_Widow', 'ChildrenHome',
       'LifeTime'],
      dtype='object')
```

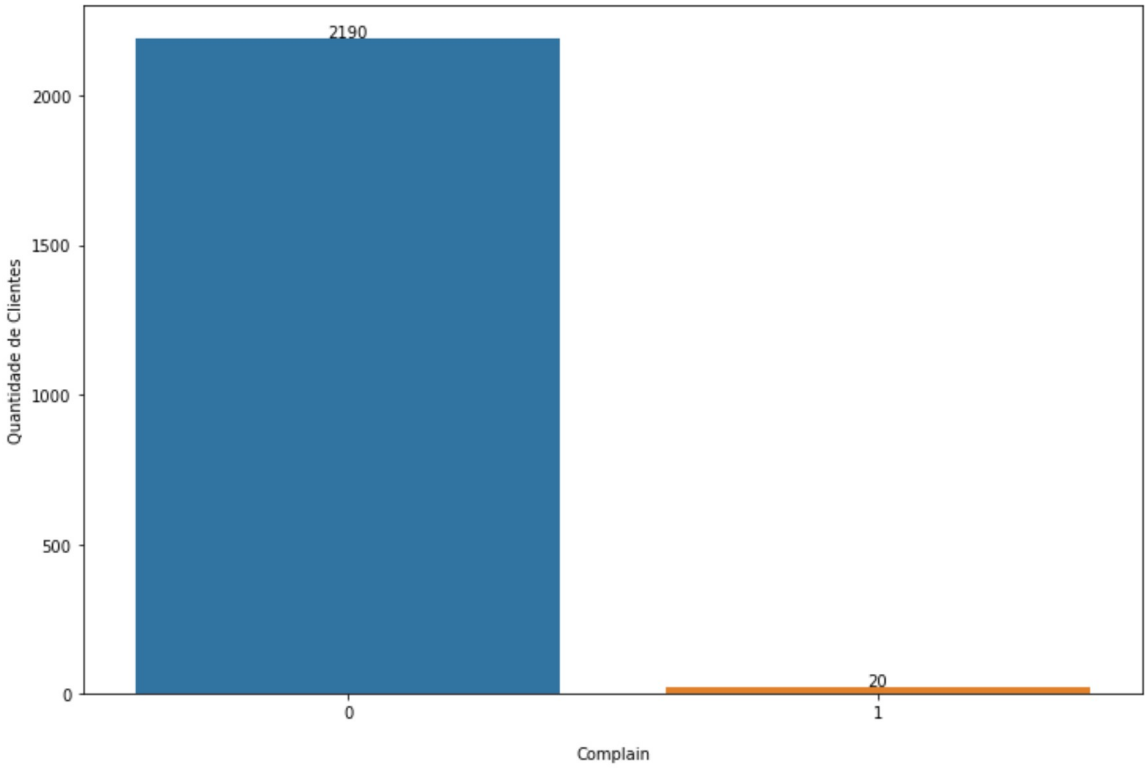
Recency

In [44]:
```
plot_num(people, 'Recency')
```



Complain

In [45]:
```
plot_cat(people, 'Complain')
```



In [46]:
```
people.drop(columns=['Complain'], inplace=True)
```

In [47]:
```
people.head()
```

Out[47]:

	ID	Year_Birth	Education	Income	Dt_Customer	Recency	Marital_Status_Divorced	Marita
0	5524	1957	2	58138.0	2012-09-04	58	0	
1	2174	1954	2	46344.0	2014-03-08	38	0	
2	4141	1965	2	71613.0	2013-08-21	26	0	
3	6182	1984	2	26646.0	2014-02-10	26	0	
4	5324	1981	4	58293.0	2014-01-19	94	0	

In [48]:
```
people.drop(columns=['Dt_Customer'], inplace=True)
```

In [49]:
```
people.columns
```

Out[49]:
```
Index(['ID', 'Year_Birth', 'Education', 'Income', 'Recency',
       'Marital_Status_Divorced', 'Marital_Status_Single',
       'Marital_Status_Together', 'Marital_Status_Widow', 'ChildrenHome',
       'LifeTime'],
      dtype='object')
```

Products

MntWines: Amount spent on wine in last 2 years MntFruits: Amount spent on fruits in last 2 years MntMeatProducts: Amount spent on meat in last 2 years MntFishProducts: Amount spent on fish in last 2 years MntSweetProducts: Amount spent on sweets in last 2 years MntGoldProds: Amount spent on gold in last 2 years

In [50]:
```
products = df[['ID','MntWines','MntFruits', 'MntMeatProducts', 'MntFishProducts'
```

In [51]:
```
products.head()
```

Out[51]:

	ID	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGc
0	5524	635	88	546	172	88	
1	2174	11	1	6	2	1	
2	4141	426	49	127	111	21	
3	6182	11	4	20	10	3	
4	5324	173	43	118	46	27	

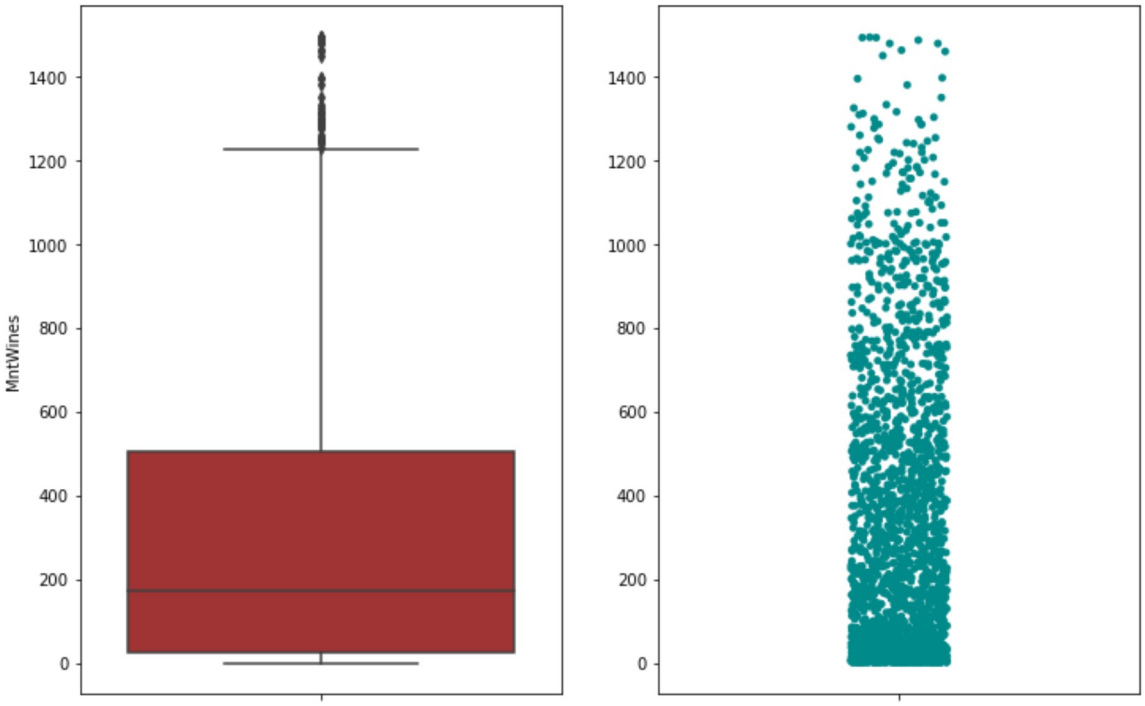
In [52]:
```
show_null(products)
```

Out[52]:

	#	% null	type
ID	0	0.0	int64
MntWines	0	0.0	int64
MntFruits	0	0.0	int64
MntMeatProducts	0	0.0	int64
MntFishProducts	0	0.0	int64
MntSweetProducts	0	0.0	int64
MntGoldProds	0	0.0	int64

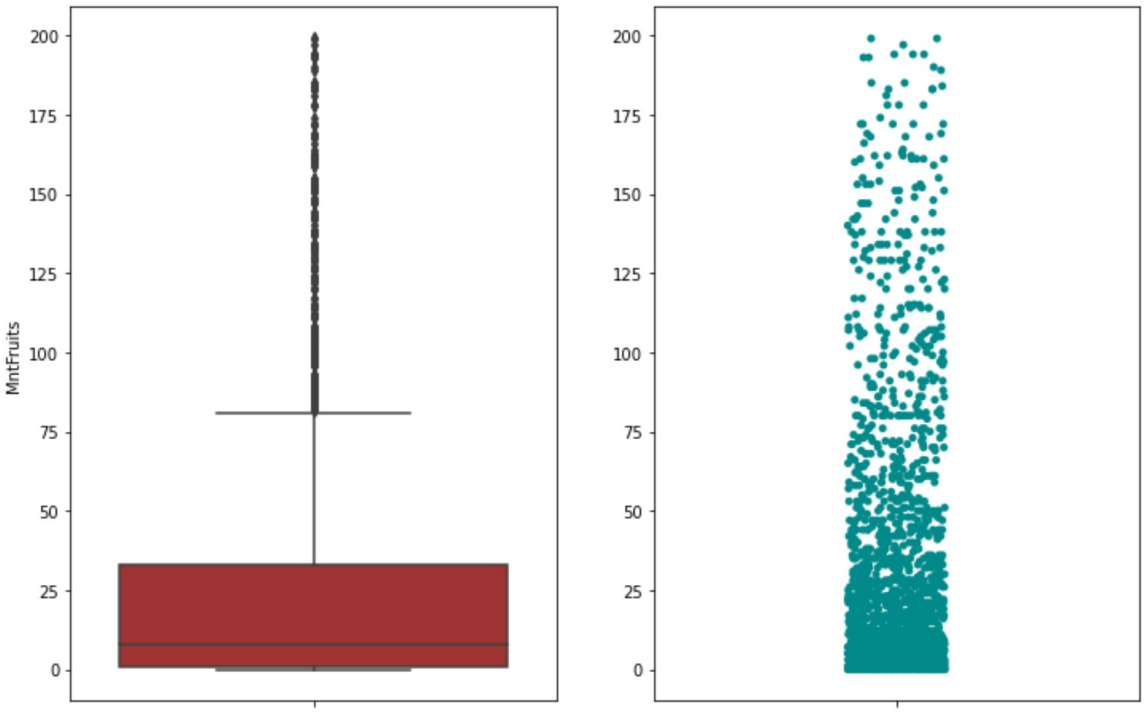
MntWines

In [53]:
```
plot_num(products, 'MntWines')
```



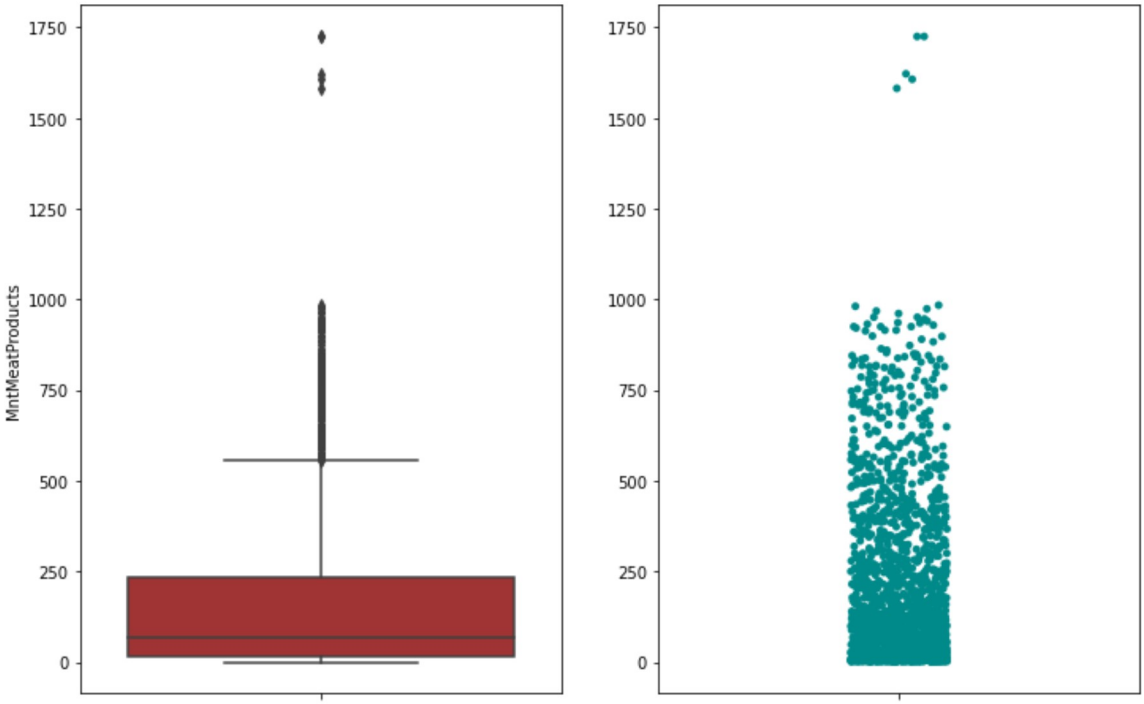
MntFruits

In [54]:
```
plot_num(products, 'MntFruits')
```



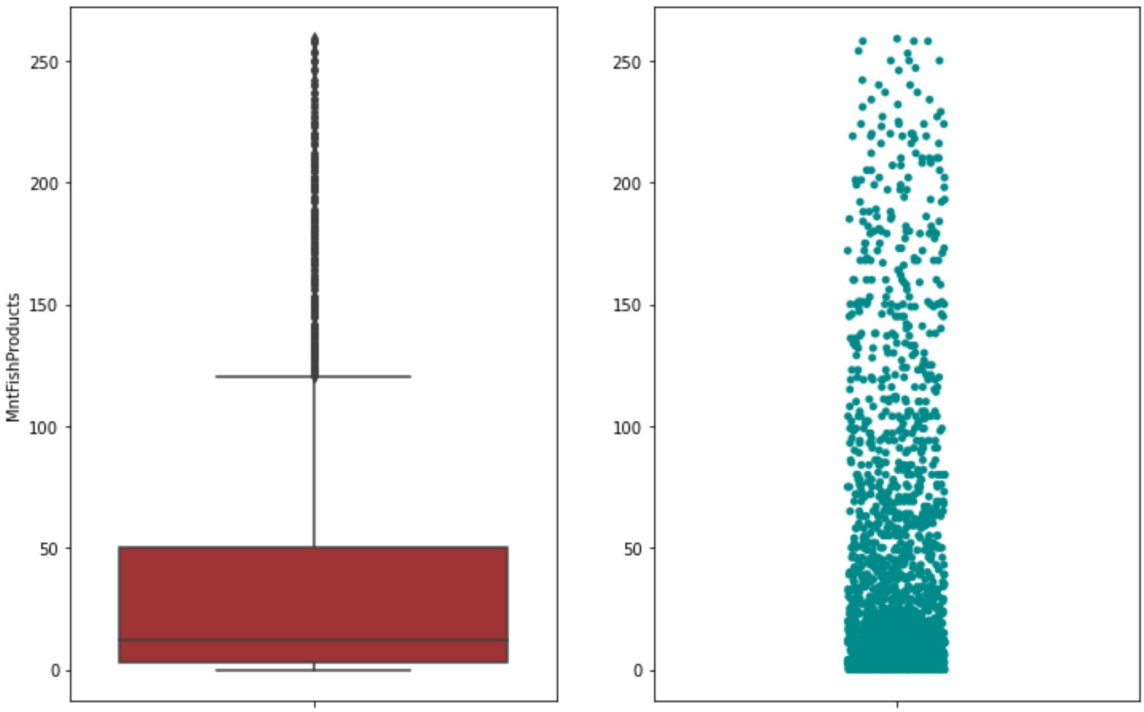
MntMeatProducts

In [55]:
```
plot_num(products, 'MntMeatProducts')
```



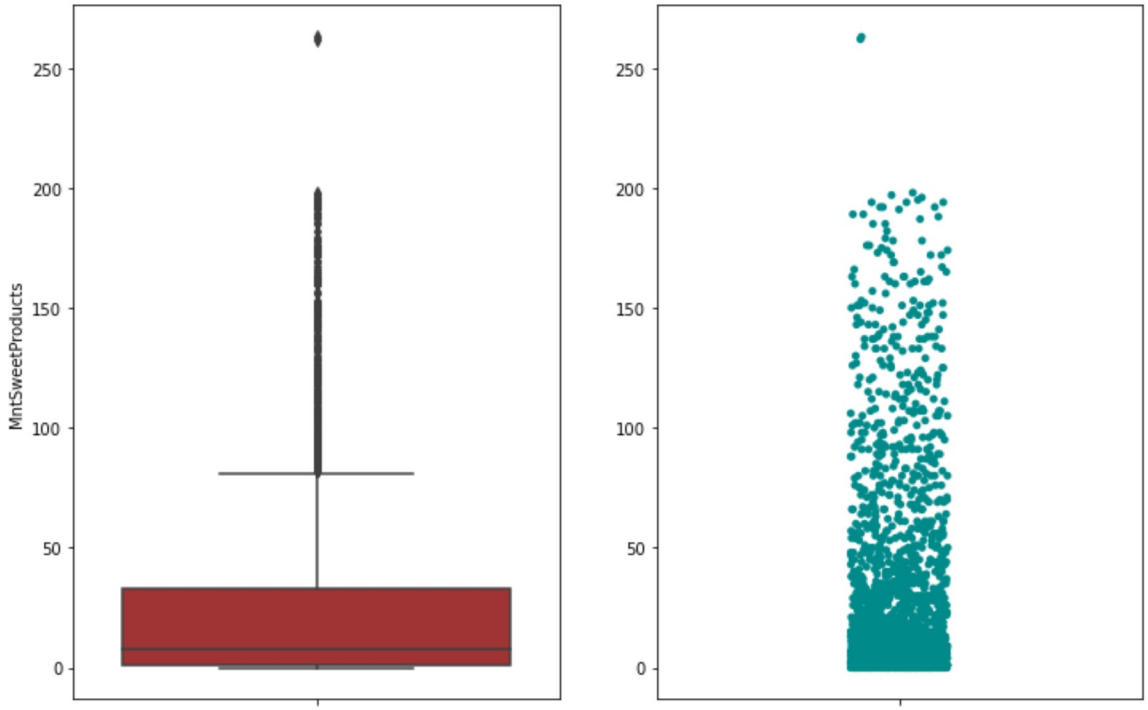
MntFishProducts

In [56]:
```
plot_num(products, 'MntFishProducts')
```



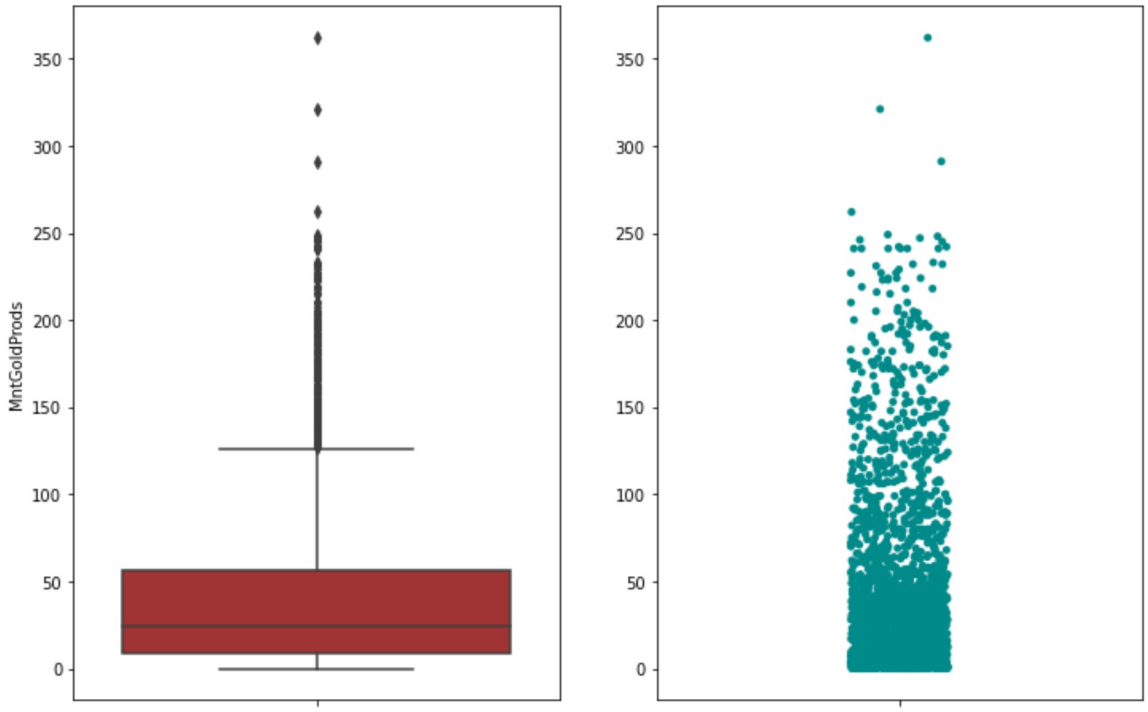
MntSweetProducts

In [57]:
```
plot_num(products, 'MntSweetProducts')
```



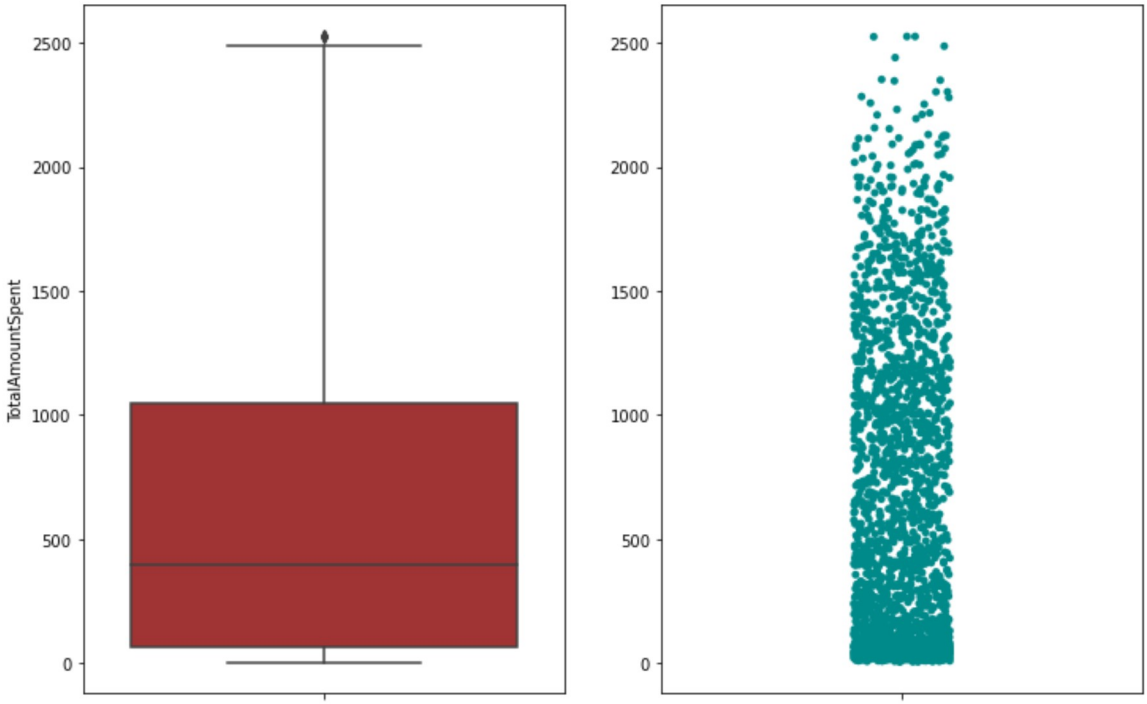
MntGoldProds

In [58]:
```
plot_num(products, 'MntGoldProds')
```



In [59]:
```
products['TotalAmountSpent'] = products.MntFishProducts + products.MntFruits + p
+ products.MntMeatProducts + products.MntSweetProducts + products.MntWines
```

In [60]:
```
plot_num(products, 'TotalAmountSpent')
```



In [61]:
```
products.MntFishProducts = products.MntFishProducts/products['TotalAmountSpent']
products.MntFruits = products.MntFruits/products['TotalAmountSpent']
products.MntGoldProds = products.MntGoldProds/products['TotalAmountSpent']
products.MntMeatProducts = products.MntMeatProducts/products['TotalAmountSpent']
products.MntSweetProducts = products.MntSweetProducts/products['TotalAmountSpent
products.MntWines = products.MntWines/products['TotalAmountSpent']
```

In [62]:
```
products.describe()
```

Out[62]:

	ID	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetF
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	224(
mean	5592.159821	0.458481	0.049505	0.249508	0.071532	(
std	3246.662198	0.228411	0.055867	0.126633	0.077909	(
min	0.000000	0.000000	0.000000	0.000000	0.000000	(
25%	2828.250000	0.289506	0.008924	0.156250	0.012571	(
50%	5458.500000	0.457482	0.029840	0.233447	0.048193	(
75%	8427.750000	0.639143	0.070237	0.328227	0.104703	(
max	11191.000000	0.963303	0.445545	0.997110	0.590909	(

In [63]:
```
products.head()
```

Out[63]:

	ID	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGc
0	5524	0.392703	0.054422	0.337662	0.106370	0.054422	(
1	2174	0.407407	0.037037	0.222222	0.074074	0.037037	(
2	4141	0.548969	0.063144	0.163660	0.143041	0.027062	(
3	6182	0.207547	0.075472	0.377358	0.188679	0.056604	(
4	5324	0.409953	0.101896	0.279621	0.109005	0.063981	(

Promotion

AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

In [64]:
```
promotion = df[['ID','AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCm
                'AcceptedCmp5', 'Response']]
```

In [65]:
```
promotion.head()
```

Out[65]:

	ID	AcceptedCmp1	AcceptedCmp2	AcceptedCmp3	AcceptedCmp4	AcceptedCmp5	Resp
0	5524	0	0	0	0	0	
1	2174	0	0	0	0	0	
2	4141	0	0	0	0	0	
3	6182	0	0	0	0	0	
4	5324	0	0	0	0	0	

In [66]:
```
show_null(promotion)
```

Out[66]:

	#	% null	type
ID	0	0.0	int64
AcceptedCmp1	0	0.0	int64
AcceptedCmp2	0	0.0	int64
AcceptedCmp3	0	0.0	int64
AcceptedCmp4	0	0.0	int64
AcceptedCmp5	0	0.0	int64
Response	0	0.0	int64

In [67]:
```
promotion.describe()
```

Out[67]:

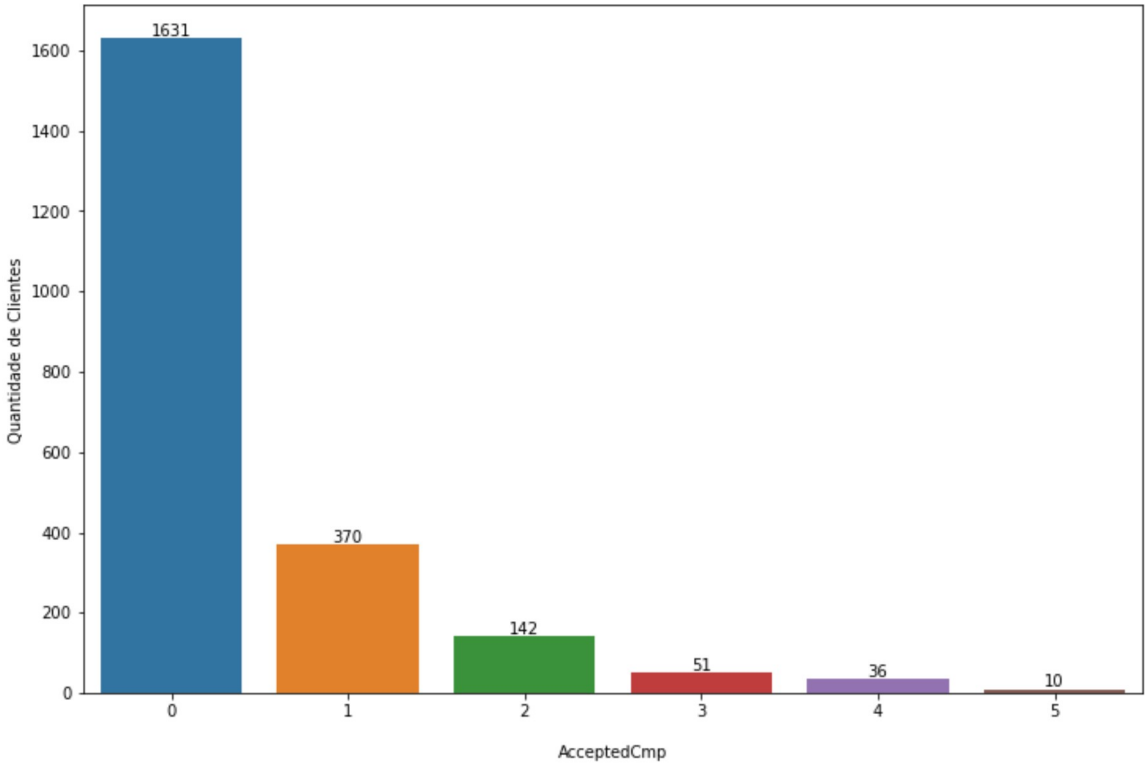
	ID	AcceptedCmp1	AcceptedCmp2	AcceptedCmp3	AcceptedCmp4	Accepted
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.0
mean	5592.159821	0.064286	0.013393	0.072768	0.074554	0.0
std	3246.662198	0.245316	0.114976	0.259813	0.262728	0.2
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	2828.250000	0.000000	0.000000	0.000000	0.000000	0.0
50%	5458.500000	0.000000	0.000000	0.000000	0.000000	0.0
75%	8427.750000	0.000000	0.000000	0.000000	0.000000	0.0
max	11191.000000	1.000000	1.000000	1.000000	1.000000	1.0

In [68]:
```
promotion['AcceptedCmp'] = promotion.AcceptedCmp1 + promotion.AcceptedCmp2 + pro
promotion.AcceptedCmp5 + promotion.Response
```

In [69]:
```
promotion['AcceptedCmp'].describe()
```

Out[69]:
```
count    2240.000000
mean        0.446875
std         0.890543
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         5.000000
Name: AcceptedCmp, dtype: float64
```

In [70]:
```
plot_cat(promotion, 'AcceptedCmp')
```



In [71]:
```
promotion = promotion[['ID', 'AcceptedCmp']]
```

In [72]:
```
promotion.head()
```

Out[72]:

	ID	AcceptedCmp
0	5524	1
1	2174	0
2	4141	0
3	6182	0
4	5324	0

Place

NumWebPurchases: Number of purchases made through the company's website NumCatalogPurchases: Number of purchases made using a catalogue NumStorePurchases: Number of purchases made directly in

stores NumDealsPurchases: Number of purchases made with a discount NumWebVisitsMonth: Number of visits to company's website in the last month

In [73]:
```
place = df[['ID','NumWebPurchases','NumCatalogPurchases', 'NumStorePurchases', '
```

In [74]:
```
place.head()
```

Out[74]:

	ID	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumDealsPurchases	I
0	5524	8	10	4	3	
1	2174	1	1	2	2	
2	4141	8	2	10	1	
3	6182	2	0	4	2	
4	5324	5	3	6	5	

In [75]:
```
show_null(place)
```

Out[75]:

	#	% null	type
ID	0	0.0	int64
NumWebPurchases	0	0.0	int64
NumCatalogPurchases	0	0.0	int64
NumStorePurchases	0	0.0	int64
NumDealsPurchases	0	0.0	int64
NumWebVisitsMonth	0	0.0	int64

In [76]:
```
place['TotalPurchases'] = place.NumWebPurchases + place.NumCatalogPurchases + pl
```

In [77]:
```
place.describe()
```

Out[77]:

	ID	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumDeals
count	2240.000000	2240.000000	2240.000000	2240.000000	22
mean	5592.159821	4.084821	2.662054	5.790179	
std	3246.662198	2.778714	2.923101	3.250958	
min	0.000000	0.000000	0.000000	0.000000	
25%	2828.250000	2.000000	0.000000	3.000000	
50%	5458.500000	4.000000	2.000000	5.000000	
75%	8427.750000	6.000000	4.000000	8.000000	
max	11191.000000	27.000000	28.000000	13.000000	

In [78]:
```
place[place.TotalPurchases == 0]
```

Out[78]:

	ID	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumDealsPurchas
655	5555	0	0	0	
981	3955	0	0	0	
1245	6862	0	0	0	
1524	11110	0	0	0	
1846	9931	0	0	0	
2132	11181	0	0	0	

In [79]:
```
place = place[place['TotalPurchases']>0]
```

In [80]:
```
place['NumWebPurchases'] = place['NumWebPurchases']/place['TotalPurchases']
place['NumCatalogPurchases'] = place['NumCatalogPurchases']/place['TotalPurchase
place['NumStorePurchases'] = place['NumStorePurchases']/place['TotalPurchases']
place['NumDealsPurchases'] = place['NumDealsPurchases']/place['TotalPurchases']
```

In [81]:
```
place.describe()
```

Out[81]:

	ID	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumDeals
count	2234.000000	2234.000000	2234.000000	2234.000000	22
mean	5585.427037	0.329874	0.164809	0.505317	
std	3245.168172	0.121858	0.140668	0.149485	
min	0.000000	0.000000	0.000000	0.000000	
25%	2817.750000	0.250000	0.000000	0.401852	
50%	5453.500000	0.333333	0.150000	0.500000	
75%	8419.500000	0.400000	0.250000	0.600000	
max	11191.000000	1.000000	1.000000	1.000000	

In [82]:
```
place[place.NumDealsPurchases > 1]
```

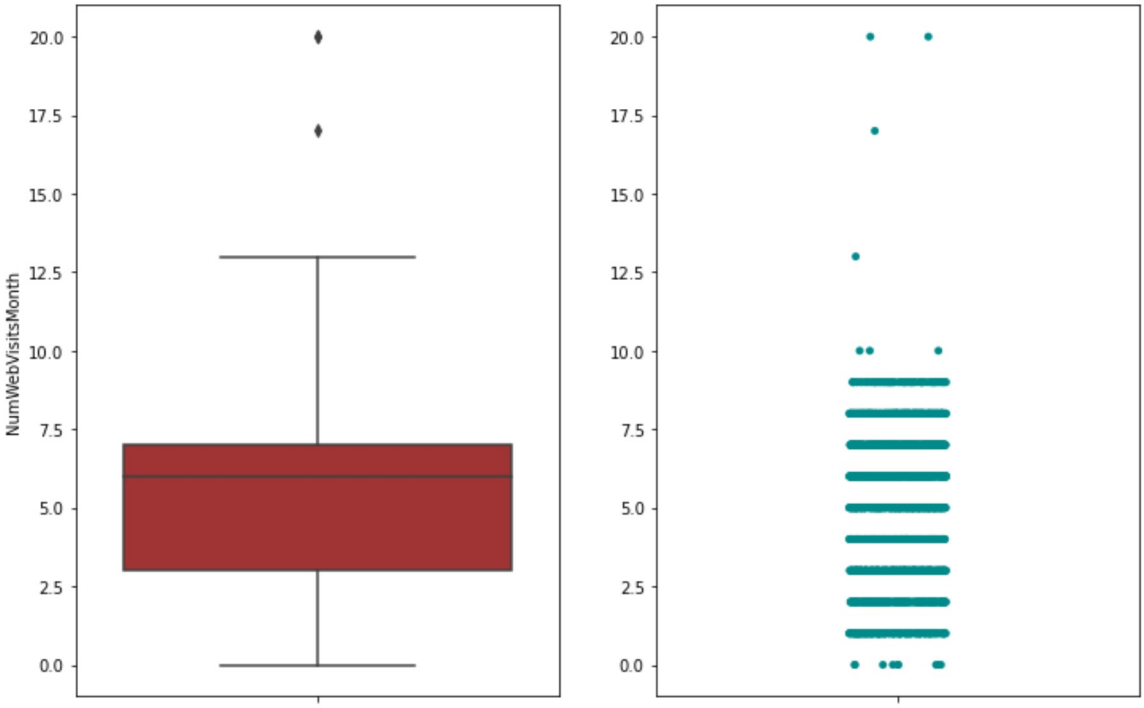
Out[82]:

	ID	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumDealsPurchas
1042	10749	0.0	1.0	0.0	15

In [83]:
```
place = place[place['NumDealsPurchases']<=1]
```

NumWebPurchases

In [84]:
```
plot_num(place, 'NumWebVisitsMonth')
```



Juntando Dados

In [85]:
```
df_lista = [people, products, promotion, place]
```

In [86]:
```
df_final = reduce(lambda df1, df2: pd.merge(df1, df2, on='ID'), df_lista)
```

In [87]:
```
df_final.shape
```

Out[87]:
```
(2203, 25)
```

In [88]:
```
df_final.columns
```

Out[88]:
```
Index(['ID', 'Year_Birth', 'Education', 'Income', 'Recency',
       'Marital_Status_Divorced', 'Marital_Status_Single',
       'Marital_Status_Together', 'Marital_Status_Widow', 'ChildrenHome',
       'LifeTime', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'TotalAmountSpent', 'AcceptedCmp', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumDealsPurchases',
       'NumWebVisitsMonth', 'TotalPurchases'],
      dtype='object')
```

In [89]:
```
df_final.drop(columns=['ID'], inplace=True)
```

Normalização

In [90]:
```
def scaler(df):
    df2 = df.copy()
    for column in df2:
        minimo = df2[column].min()
        maximo = df2[column].max()
        df2[column] = (df[column] - minimo)/(maximo - minimo)
    return df2
```

In [91]:
```
scaled_df = scaler(df_final)
```

In [92]:
```
scaled_df.describe()
```

Out[92]:

	Year_Birth	Education	Income	Recency	Marital_Status_Divorced	Marital_Sta
count	2203.000000	2203.000000	2203.000000	2203.000000	2203.000000	22
mean	0.516179	0.615070	0.309517	0.494858	0.103495	
std	0.208790	0.251205	0.132629	0.292626	0.304674	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.339286	0.500000	0.205796	0.242424	0.000000	
50%	0.535714	0.500000	0.305883	0.494949	0.000000	
75%	0.660714	0.750000	0.412801	0.747475	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

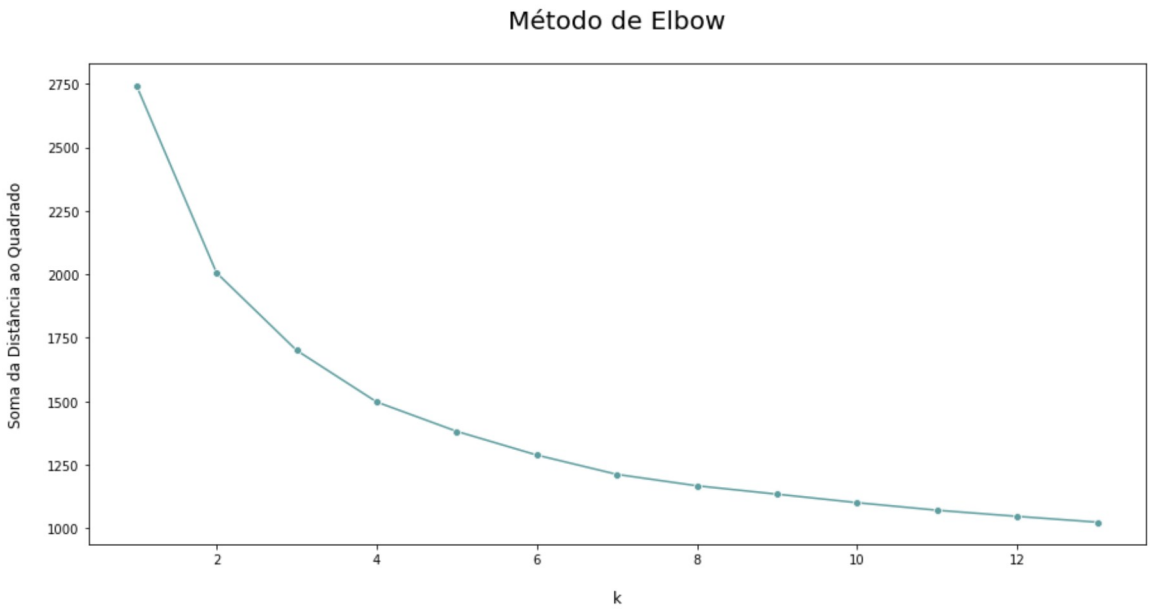
8 rows × 24 columns

Método de Elbow

In [93]:
```
inertia = []
num_clusters = range(1, 14)

for k in num_clusters:
    Model = KMeans(n_clusters=k)
    Model.fit(scaled_df)
    inertia.append(Model.inertia_)
```

In [95]:
```
plt.figure(figsize=(15,7))
sns.lineplot(x=num_clusters, y=inertia, color='cadetblue', marker='o')
plt.xlabel('\nk', size=12)
plt.ylabel('Soma da Distância ao Quadrado\n', size=12)
plt.title('Método de Elbow\n', size=20)
plt.show()
```



Alocação de Pesos

In [96]:
```
scaled_df.columns
```

Out[96]:
```
Index(['Year_Birth', 'Education', 'Income', 'Recency',
       'Marital_Status_Divorced', 'Marital_Status_Single',
       'Marital_Status_Together', 'Marital_Status_Widow', 'ChildrenHome',
       'LifeTime', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'TotalAmountSpent', 'AcceptedCmp', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumDealsPurchases',
       'NumWebVisitsMonth', 'TotalPurchases'],
      dtype='object')
```

In [97]:
```
scaled_df.TotalAmountSpent = 8*scaled_df.TotalAmountSpent
scaled_df.TotalPurchases = 6*scaled_df.TotalPurchases
scaled_df.Recency = 4*scaled_df.Recency
scaled_df.NumWebVisitsMonth = 4*scaled_df.NumWebVisitsMonth
scaled_df.Income = 4*scaled_df.Income
scaled_df.LifeTime = 4*scaled_df.LifeTime
scaled_df.Year_Birth = 2*scaled_df.Year_Birth
scaled_df.ChildrenHome = 2*scaled_df.ChildrenHome
scaled_df.Education = 2*scaled_df.Education
```

In [98]:
```
scaled_df.describe()
```

Out[98]:

	Year_Birth	Education	Income	Recency	Marital_Status_Divorced	Marital_Sta
count	2203.000000	2203.000000	2203.000000	2203.000000	2203.000000	22
mean	1.032358	1.230141	1.238068	1.979431	0.103495	
std	0.417580	0.502410	0.530515	1.170504	0.304674	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.678571	1.000000	0.823182	0.969697	0.000000	
50%	1.071429	1.000000	1.223532	1.979798	0.000000	
75%	1.321429	1.500000	1.651204	2.989899	0.000000	
max	2.000000	2.000000	4.000000	4.000000	1.000000	

8 rows × 24 columns

Kmeans

In [99]:
```
Model = KMeans(n_clusters=4)
```

In [100…
```
clusters = Model.fit_predict(scaled_df)
```

In [101…
```
clusters
```

Out[101]:
```
array([0, 2, 3, ..., 3, 3, 2])
```

In [102…
```
df_final['cluster'] = clusters
```

In [103…
```
df_final
```

Out[103]:

	Year_Birth	Education	Income	Recency	Marital_Status_Divorced	Marital_Status_Single	I
0	1957	2	58138.0	58	0	1	
1	1954	2	46344.0	38	0	1	
2	1965	2	71613.0	26	0	0	
3	1984	2	26646.0	26	0	0	
4	1981	4	58293.0	94	0	0	
...	...	...	...	...	...	...	
2198	1967	2	61223.0	46	0	0	
2199	1946	4	64014.0	56	0	0	
2200	1981	2	56981.0	91	1	0	
2201	1956	3	69245.0	8	0	0	
2202	1954	4	52869.0	40	0	0	

2203 rows × 25 columns

Salvando o modelo

In [104…

```
# save the model to disk
filename = 'modelo_finalizado.sav'
pickle.dump(Model, open(filename, 'wb'))
```

In [ ]:

```
# load the model from disk
Model = pickle.load(open(filename, 'rb'))
```

Análise dos Clusters

In [105…

```
df_final.cluster.value_counts()
```

Out[105]:

```
3    629
2    580
1    575
0    419
Name: cluster, dtype: int64
```

In [106…

```
df_final.groupby('cluster').mean()
```

Out[106]:

cluster	Year_Birth	Education	Income	Recency	Marital_Status_Divorced	Marital_Statu
0	1968.761337	2.596659	77120.233890	50.840095	0.102625	
1	1970.019130	2.314783	36496.231304	74.389565	0.104348	
2	1971.329310	2.394828	36569.187931	23.355172	0.094828	
3	1965.750397	2.562798	63507.672496	48.179650	0.111288	

4 rows × 24 columns

In [107…
```
pd.set_option('display.max_columns', None)
```

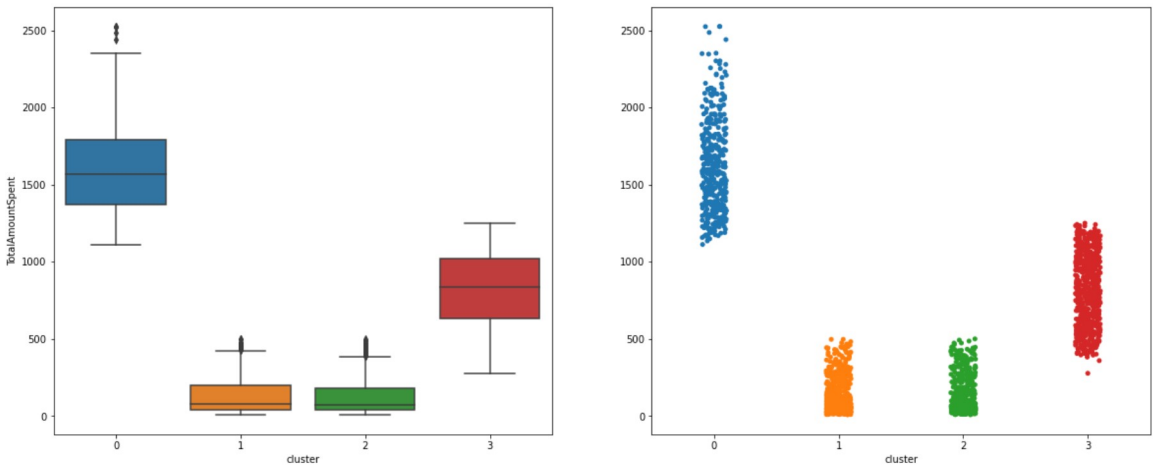
In [109…
```
df_final.groupby('cluster').mean()
```

Out[109]:

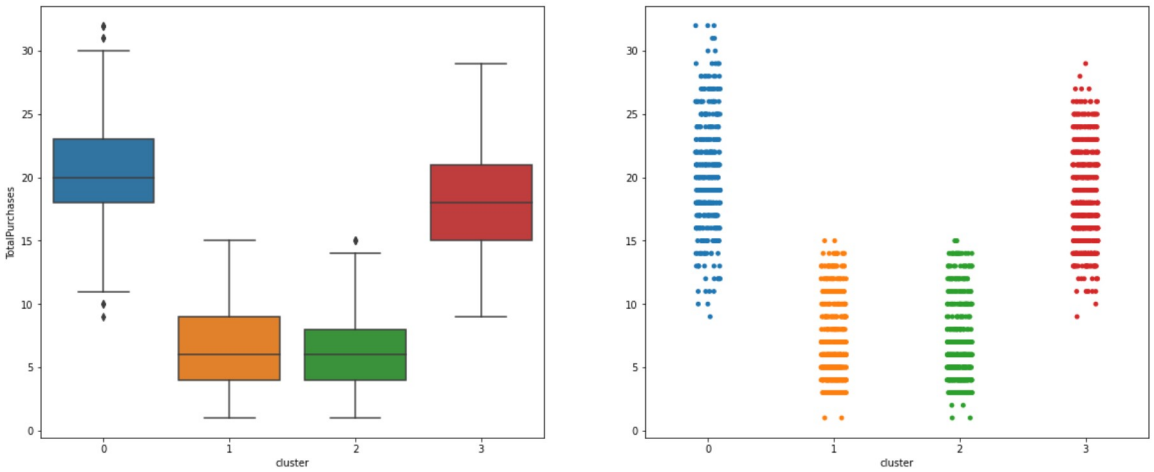
cluster	Year_Birth	Education	Income	Recency	Marital_Status_Divorced	Marital_Statu
0	1968.761337	2.596659	77120.233890	50.840095	0.102625	(
1	1970.019130	2.314783	36496.231304	74.389565	0.104348	(
2	1971.329310	2.394828	36569.187931	23.355172	0.094828	(
3	1965.750397	2.562798	63507.672496	48.179650	0.111288	(

Cluster 0: Grupo de mais alto valor, com menor número de clientes. Além de terem uma alta renda e maior nível de educação, são os que mais compram, com ticket médio mais alto entre todos os grupos. Não costumam visitar tanto o website da empresa, nem comprar sempre através de promoções, pois preferem as compras por catálogo e diretamente na loja. As categorias preferidas são a de carnes e a de vinhos. A maioria não tem filhos em casa e são os clientes cadastrados há mais tempo.Cluster 3: É um grupo que realiza muitas compras, porém com ticket médio bem menor que o grupo anterior, mas ainda assim trazendo bastante receita para a empresa. Os clientes têm um salário anual relativamente alto, são mais velhos e geralmente têm 1 filho em casa. É o grupo que mais aprecia a categoria de vinhos e que tem maior número de clientes.Cluster 2: É o cluster com clientes mais novos, com menor poder aquisitivo, porém são os mais ativos dentro da empresa. É o grupo que se cadastrou há menos tempo. Os clientes gostam de fazer compras em promoções. Gostam de fazer compras pela loja física. Gostam de visitar o website, provavelmente para buscar promoções. Gostam mais de gold prods do que os outros clusters de maior valor.Cluster 1: É um grupo jovem, com menor poder aquisitivo, que é o menos ativo entre todos eles. A empresa precisa tomar certo cuidado, pois estão churnando e deixando de comprar como antes. É o grupo que menos aceita as campanhas de promoção. É uma oportunidade importante para a empresa fazer campanhas de produtos mais destinados ao público infantil. Grupo que mais faz compras em promoções (%). Gostam de fazer compras pela loja física. São os que mais visitam o website. Gostam mais de gold prods do que os outros clusters de maior valor.

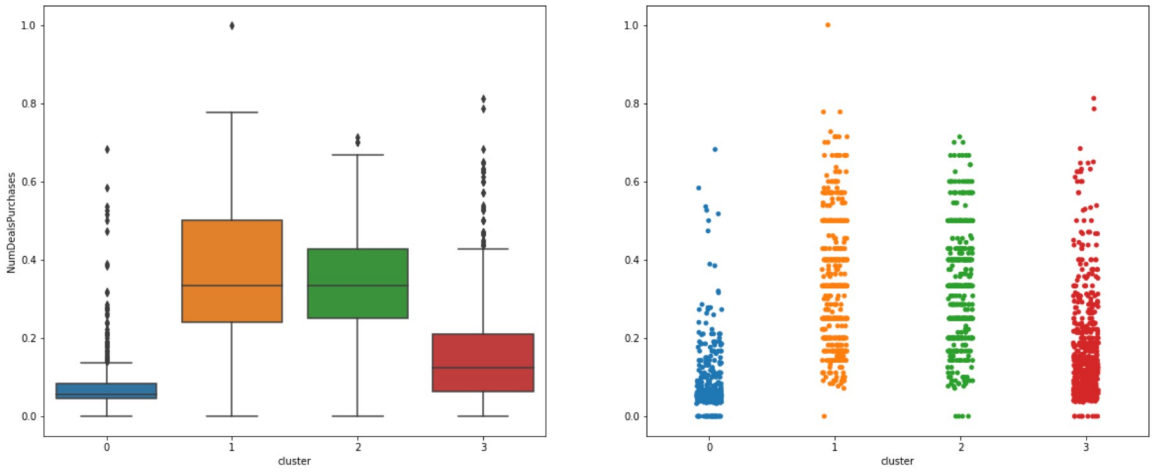
In [110…
```
fig = plt.figure(figsize=(20,8))
gs = GridSpec(1,2)
sns.boxplot(x='cluster', y='TotalAmountSpent', data=df_final, ax=fig.add_subplot
plt.ylabel('TotalAmountSpent')
plt.xlabel('cluster')
sns.stripplot(x='cluster', y='TotalAmountSpent', data=df_final, ax=fig.add_subpl
plt.ylabel(None)
plt.xlabel('cluster')
plt.show()
```



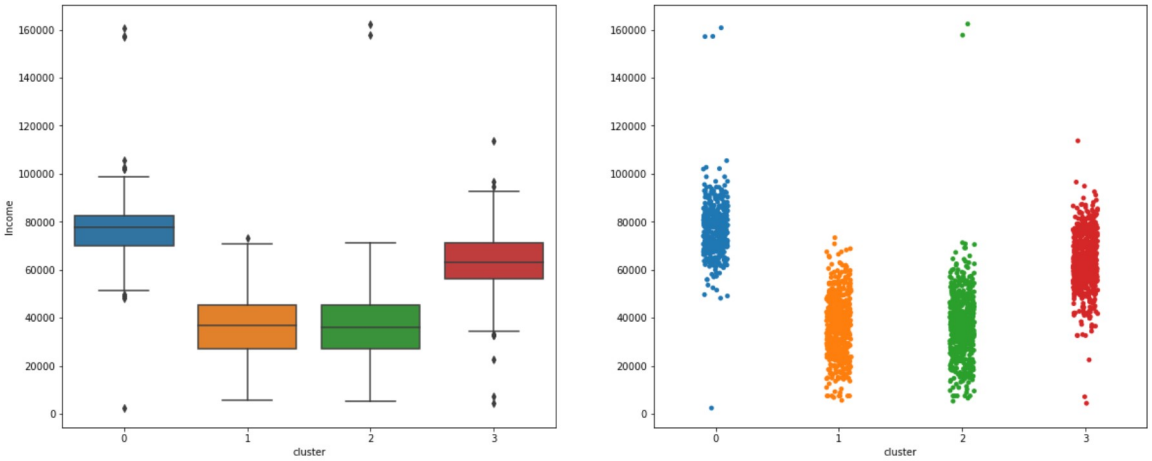
In [111…
```
fig = plt.figure(figsize=(20,8))
gs = GridSpec(1,2)
sns.boxplot(x='cluster', y='TotalPurchases', data=df_final, ax=fig.add_subplot(g
plt.ylabel('TotalPurchases')
plt.xlabel('cluster')
sns.stripplot(x='cluster', y='TotalPurchases', data=df_final, ax=fig.add_subplot
plt.ylabel(None)
plt.xlabel('cluster')
plt.show()
```



In [112…
```
fig = plt.figure(figsize=(20,8))
gs = GridSpec(1,2)
sns.boxplot(x='cluster', y='NumDealsPurchases', data=df_final, ax=fig.add_subplo
plt.ylabel('NumDealsPurchases')
plt.xlabel('cluster')
sns.stripplot(x='cluster', y='NumDealsPurchases', data=df_final, ax=fig.add_subp
plt.ylabel(None)
plt.xlabel('cluster')
plt.show()
```

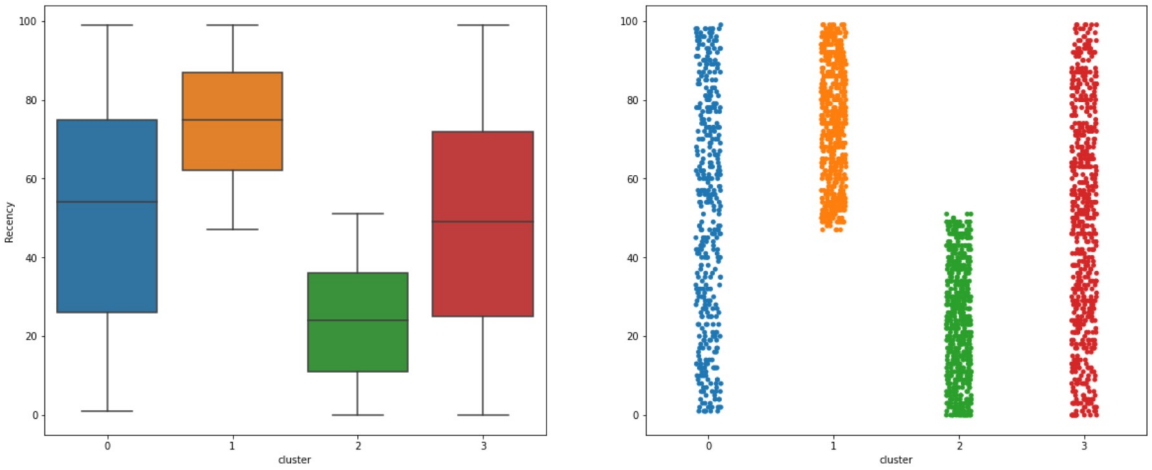


In [113…
```
fig = plt.figure(figsize=(20,8))
gs = GridSpec(1,2)
sns.boxplot(x='cluster', y='Income', data=df_final, ax=fig.add_subplot(gs[0,0]))
plt.ylabel('Income')
plt.xlabel('cluster')
sns.stripplot(x='cluster', y='Income', data=df_final, ax=fig.add_subplot(gs[0,1]
plt.ylabel(None)
plt.xlabel('cluster')
plt.show()
```



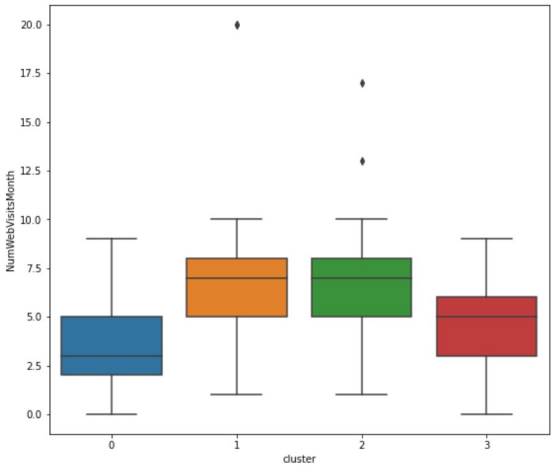
In [114…

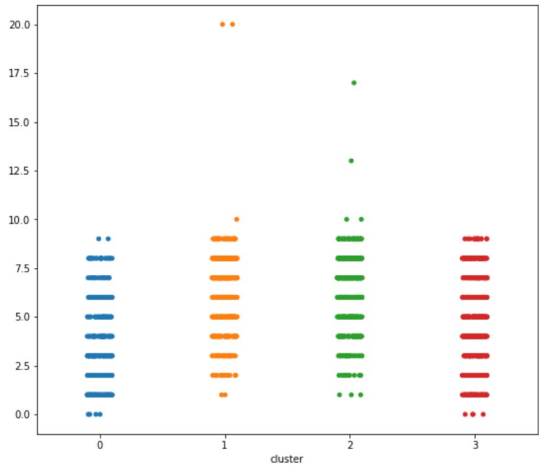
```
fig = plt.figure(figsize=(20,8))
gs = GridSpec(1,2)
sns.boxplot(x='cluster', y='Recency', data=df_final, ax=fig.add_subplot(gs[0,0])
plt.ylabel('Recency')
plt.xlabel('cluster')
sns.stripplot(x='cluster', y='Recency', data=df_final, ax=fig.add_subplot(gs[0,1
plt.ylabel(None)
plt.xlabel('cluster')
plt.show()
```



In [115…

```
fig = plt.figure(figsize=(20,8))
gs = GridSpec(1,2)
sns.boxplot(x='cluster', y='NumWebVisitsMonth', data=df_final, ax=fig.add_subplo
plt.ylabel('NumWebVisitsMonth')
plt.xlabel('cluster')
sns.stripplot(x='cluster', y='NumWebVisitsMonth', data=df_final, ax=fig.add_subp
plt.ylabel(None)
plt.xlabel('cluster')
plt.show()
```


NumWebVisitsMonth
cluster
0
1
2
3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


cluster
0
1
2
3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0