# E-commerce shopping dataset

August 30, 2024

## 0.1 E-commerce Shopping website Analytics

**Objective:** To conduct a thorough exploratory data analysis (EDA) and hypothesis testing on two comprehensive datasets one containing information on customers visiting the shopping site for purchase and another that has demographic, purchase, and marketing information about the group of people

**Expectations:**

The project expects you to Analyze user behavior across different page categories, engagement time, and other features. Gain insights into factors influencing purchase decisions and identify areas for optimization. Formulate some hypotheses on the dataset and check if they are correct.

[6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Downloading and reading the shopping csv file

[7]:
```
df = pd.read_csv("shopping.csv")
df
```

[7]:
```
       Administrative  Administrative_Duration  Informational  \
0                   0                      0.0              0
1                   0                      0.0              0
2                   0                      0.0              0
3                   0                      0.0              0
4                   0                      0.0              0
…                 …                        …              …
12325               3                    145.0              0
12326               0                      0.0              0
12327               0                      0.0              0
12328               4                     75.0              0
12329               0                      0.0              0

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
0                         0.0               1                 0.000000
1                         0.0               2                64.000000
2                         0.0               1                 0.000000
```

```
3                       0.0              2              2.666667
4                       0.0             10            627.500000
…                     …              …               …
12325                   0.0             53           1783.791667
12326                   0.0              5            465.750000
12327                   0.0              6            184.250000
12328                   0.0             15            346.000000
12329                   0.0              3             21.250000

       BounceRates  ExitRates  PageValues  SpecialDay Month  OperatingSystems  \
0         0.200000   0.200000    0.000000         0.0   Feb                 1
1         0.000000   0.100000    0.000000         0.0   Feb                 2
2         0.200000   0.200000    0.000000         0.0   Feb                 4
3         0.050000   0.140000    0.000000         0.0   Feb                 3
4         0.020000   0.050000    0.000000         0.0   Feb                 3
…              …          …           …         …   …               …
12325     0.007143   0.029031   12.241717         0.0   Dec                 4
12326     0.000000   0.021333    0.000000         0.0   Nov                 3
12327     0.083333   0.086667    0.000000         0.0   Nov                 3
12328     0.000000   0.021053    0.000000         0.0   Nov                 2
12329     0.000000   0.066667    0.000000         0.0   Nov                 3

       Browser  Region  TrafficType        VisitorType  Weekend  Revenue
0            1       1            1  Returning_Visitor    False    False
1            2       1            2  Returning_Visitor    False    False
2            1       9            3  Returning_Visitor    False    False
3            2       2            4  Returning_Visitor    False    False
4            3       1            4  Returning_Visitor     True    False
…          …     …          …                …      …      …
12325        6       1            1  Returning_Visitor     True    False
12326        2       1            8  Returning_Visitor     True    False
12327        2       1           13  Returning_Visitor     True    False
12328        2       3           11  Returning_Visitor    False    False
12329        2       1            2        New_Visitor     True    False

[12330 rows x 18 columns]
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
```

```
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

Unique number of values for specific categorical columns

[9]:
```
columns_list = df[['SpecialDay', 'Month', 'OperatingSystems', 'Browser',␣
 ↪'Region', 'TrafficType', 'VisitorType',
                'Weekend', 'Revenue']]

for columns in columns_list.columns:
  unique_count = columns_list[columns].nunique()
  print(columns, "-", unique_count)
```

```
SpecialDay - 6
Month - 10
OperatingSystems - 8
Browser - 13
Region - 9
TrafficType - 20
VisitorType - 3
Weekend - 2
Revenue - 2
```

Checking for the presence of null values in dataset.

[10]:
```
df.isna().isna().sum()
```

[10]:
```
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
```

```
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

shape of the dataset

[11]:
```
df.shape
```

[11]:
```
(12330, 18)
```

summary statistics of the dataset

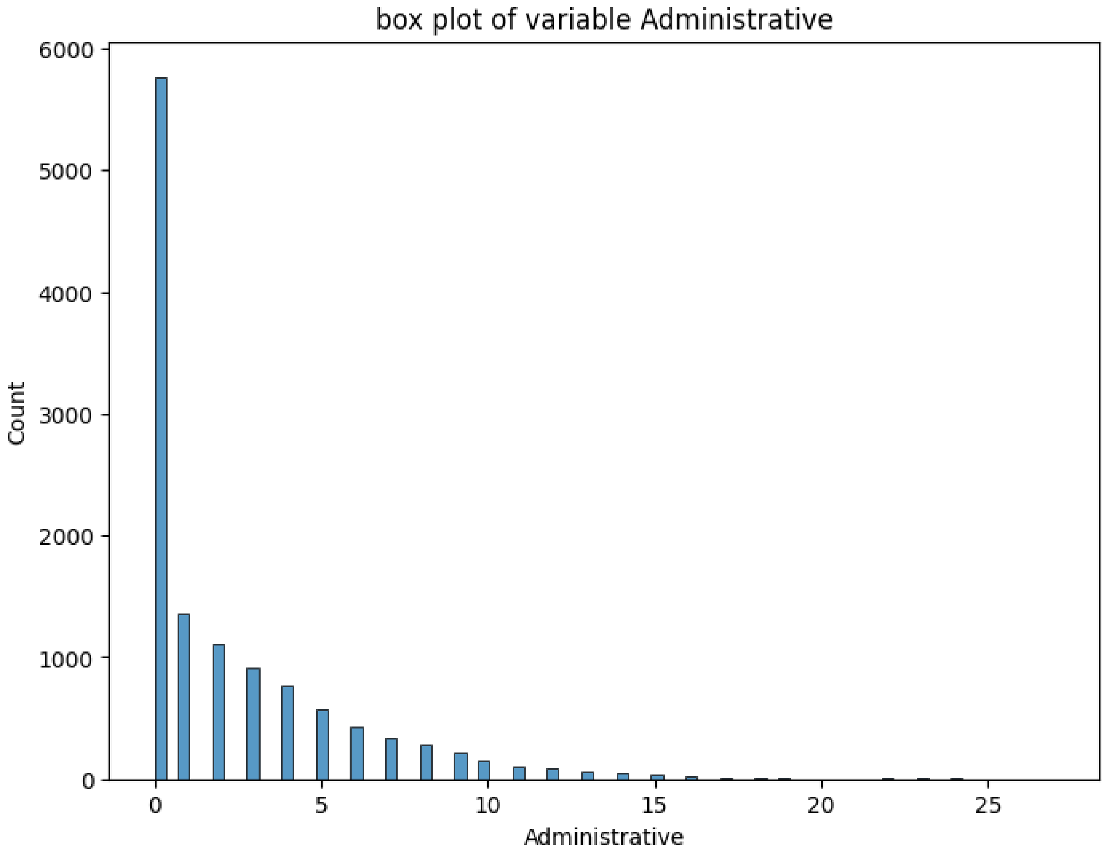
[12]:
```
summary_df = df.describe()
summary_df
```

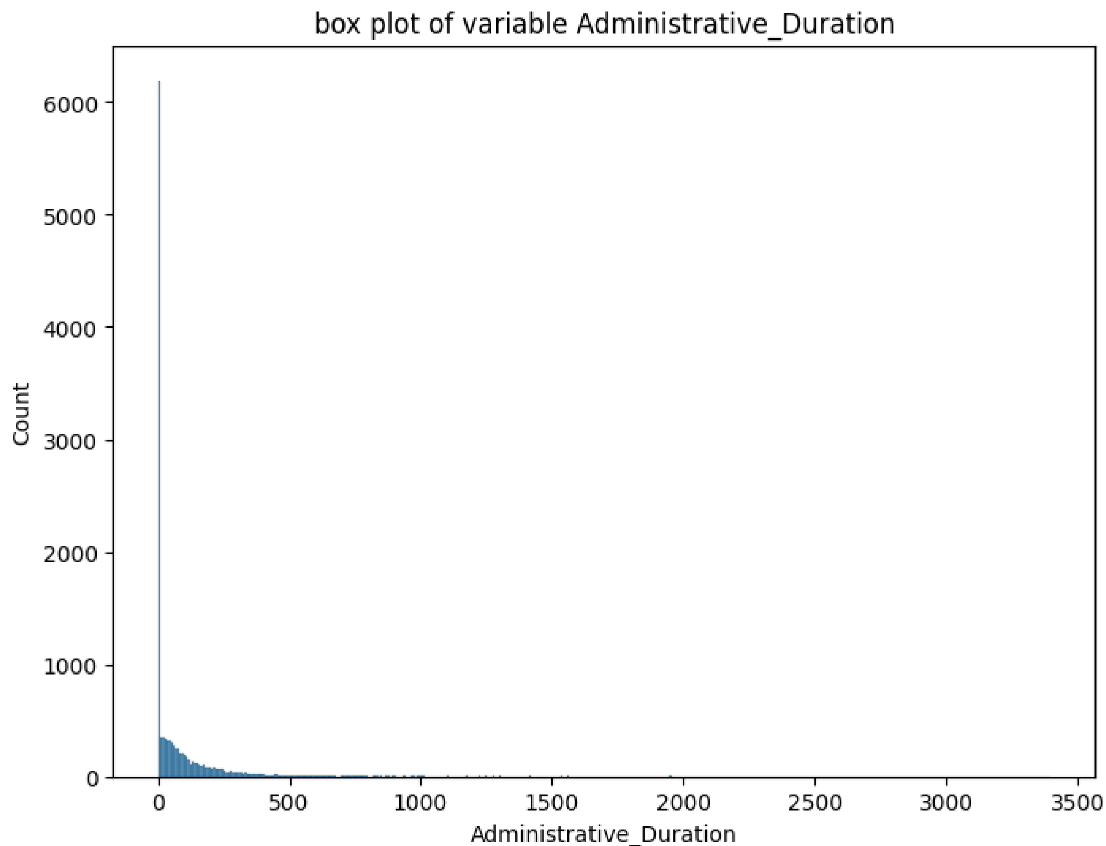
[12]:
```
       Administrative  Administrative_Duration  Informational  \
count    12330.000000             12330.000000   12330.000000
mean         2.315166                80.818611       0.503569
std          3.321784               176.779107       1.270156
min          0.000000                 0.000000       0.000000
25%          0.000000                 0.000000       0.000000
50%          1.000000                 7.500000       0.000000
75%          4.000000                93.256250       0.000000
max         27.000000              3398.750000      24.000000

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
count            12330.000000    12330.000000             12330.000000
mean                34.472398       31.731468              1194.746220
std                140.749294       44.475503              1913.669288
min                  0.000000        0.000000                 0.000000
25%                  0.000000        7.000000               184.137500
50%                  0.000000       18.000000               598.936905
75%                  0.000000       38.000000              1464.157214
max               2549.375000      705.000000             63973.522230

        BounceRates     ExitRates    PageValues    SpecialDay  \
count  12330.000000  12330.000000  12330.000000  12330.000000
mean       0.022191      0.043073      5.889258      0.061427
std        0.048488      0.048597     18.568437      0.198917
```

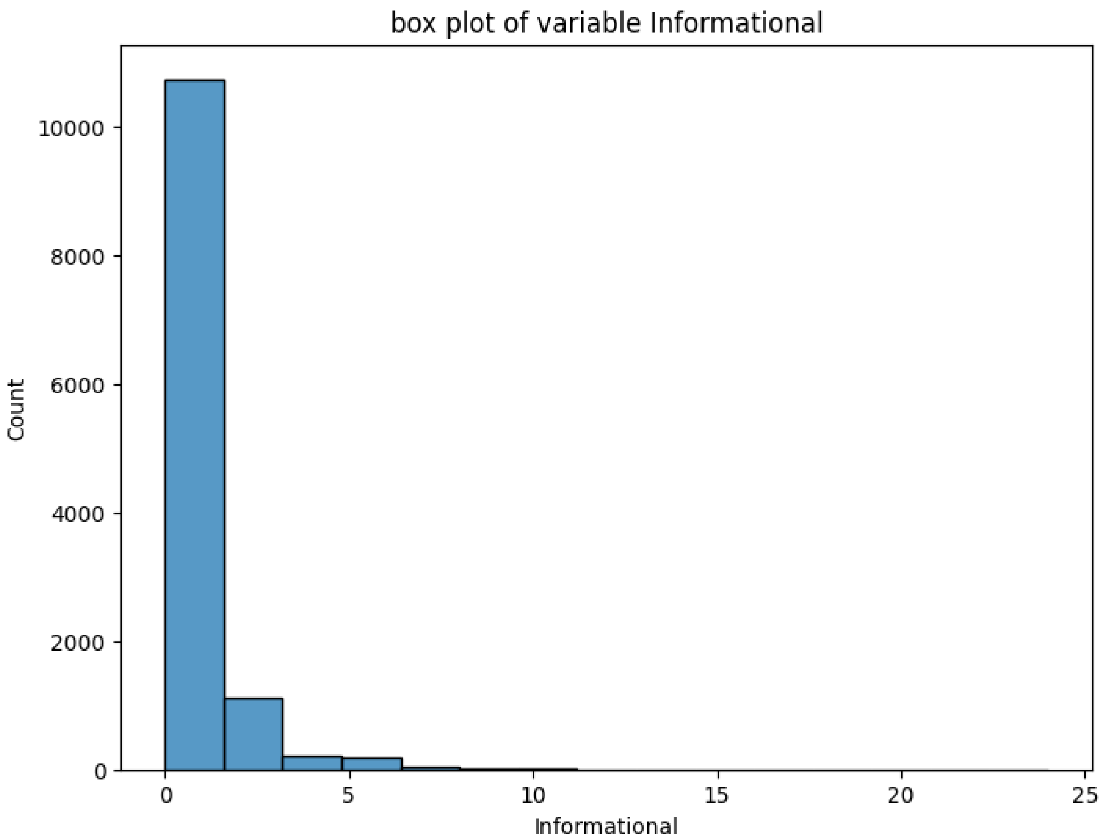
```
min        0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.014286      0.000000      0.000000
50%        0.003112      0.025156      0.000000      0.000000
75%        0.016813      0.050000      0.000000      0.000000
max        0.200000      0.200000    361.763742      1.000000

       OperatingSystems       Browser        Region   TrafficType
count      12330.000000  12330.000000  12330.000000  12330.000000
mean           2.124006      2.357097      3.147364      4.069586
std            0.911325      1.717277      2.401591      4.025169
min            1.000000      1.000000      1.000000      1.000000
25%            2.000000      2.000000      1.000000      2.000000
50%            2.000000      2.000000      3.000000      2.000000
75%            3.000000      2.000000      4.000000      4.000000
max            8.000000     13.000000      9.000000     20.000000
```

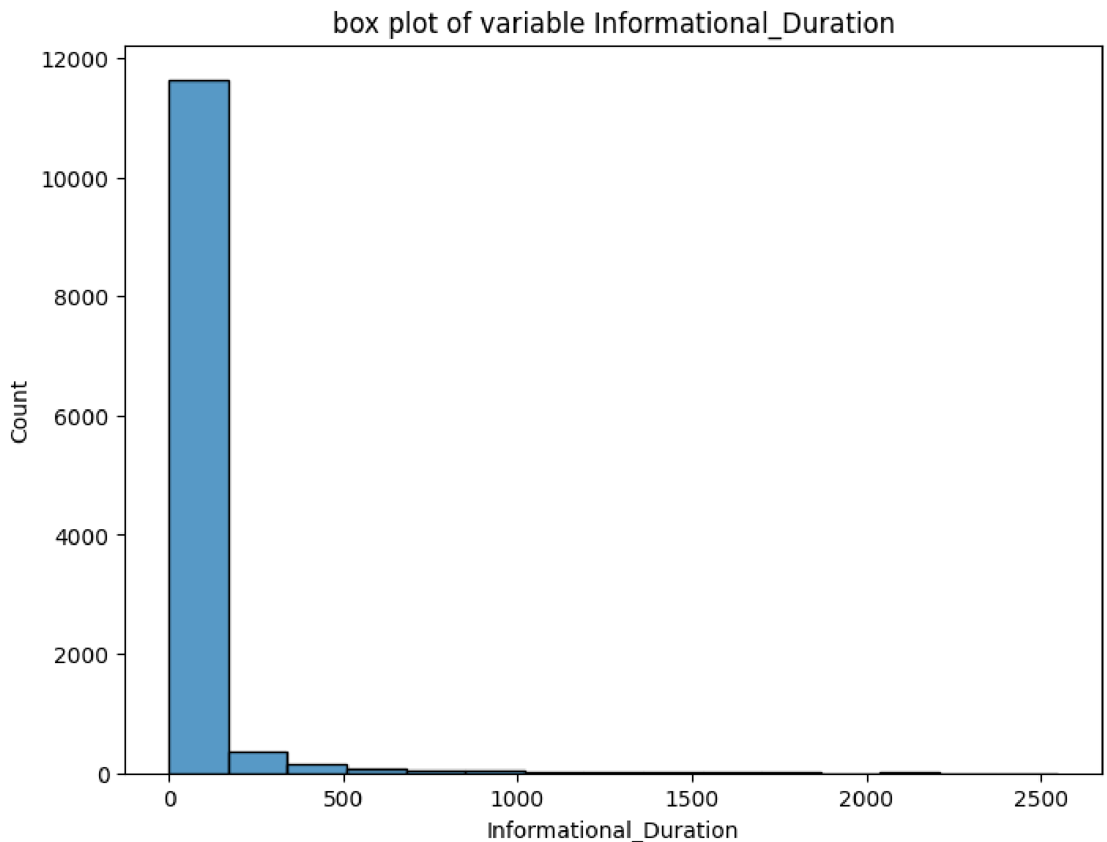
Distribution of the numerical features in the dataset

[13]:
```
numeric_variables = df.select_dtypes(include = np.number).columns
numeric_df = df[numeric_variables]

for variable in numeric_df:
  plt.figure(figsize = (8, 6))
  sns.histplot(data = numeric_df[variable])
  plt.title(f"box plot of variable {variable}")
```

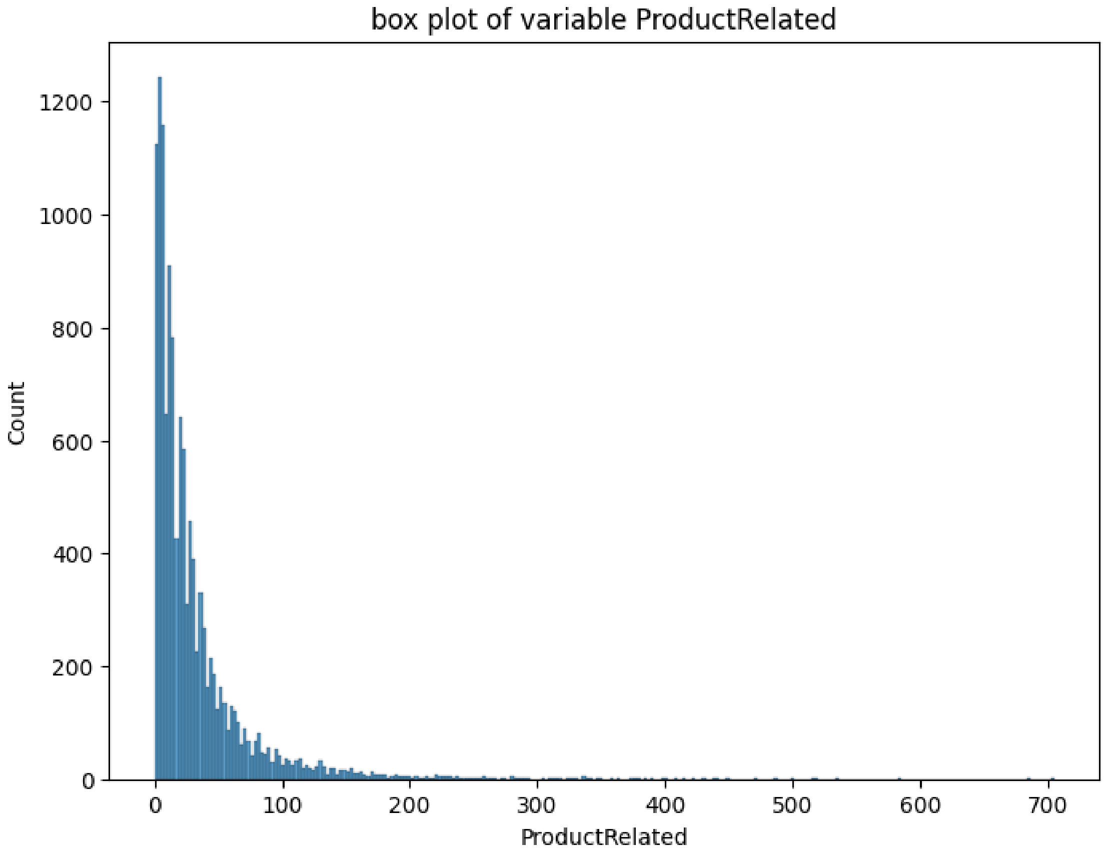

box plot of variable Administrative
Count
Administrative
0
1000
2000
3000
4000
5000
6000
0
5
10
15
20
25

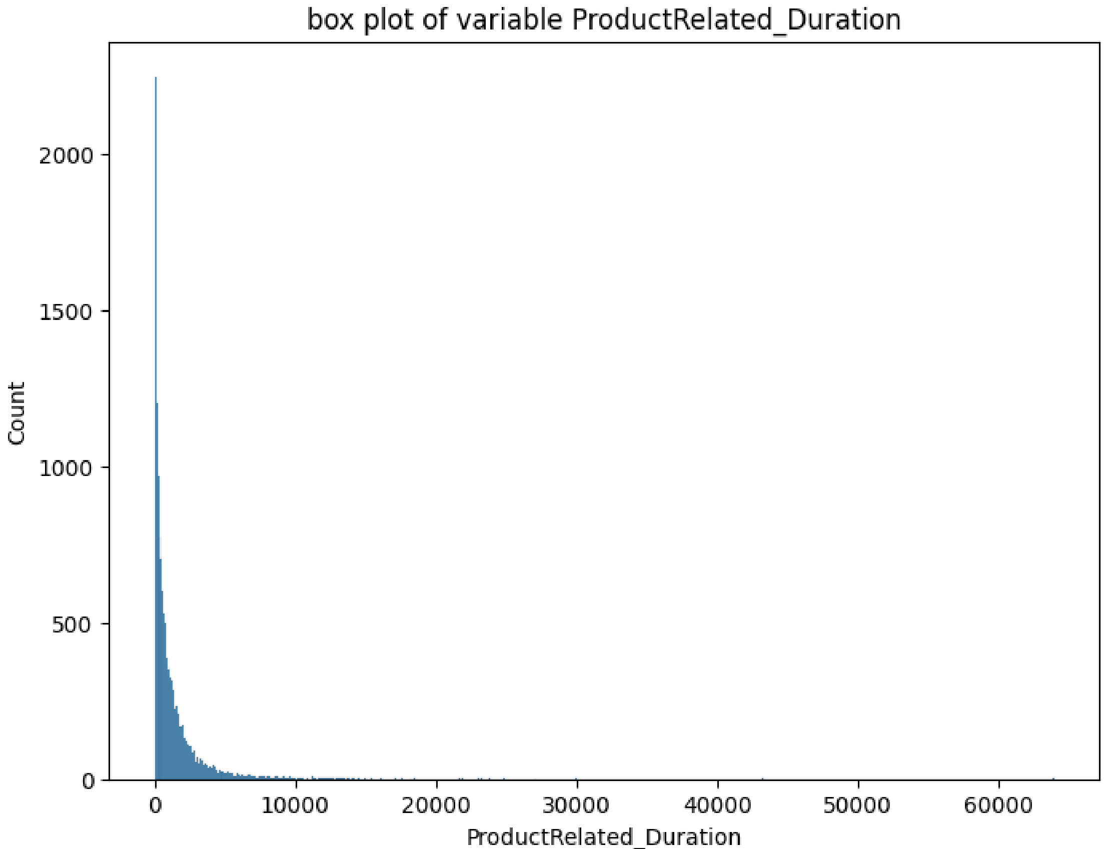

box plot of variable Administrative_Duration
Count
6000
5000
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
3500
Administrative_Duration


box plot of variable Informational
Count
10000
8000
6000
4000
2000
0
0
5
10
15
20
25
Informational

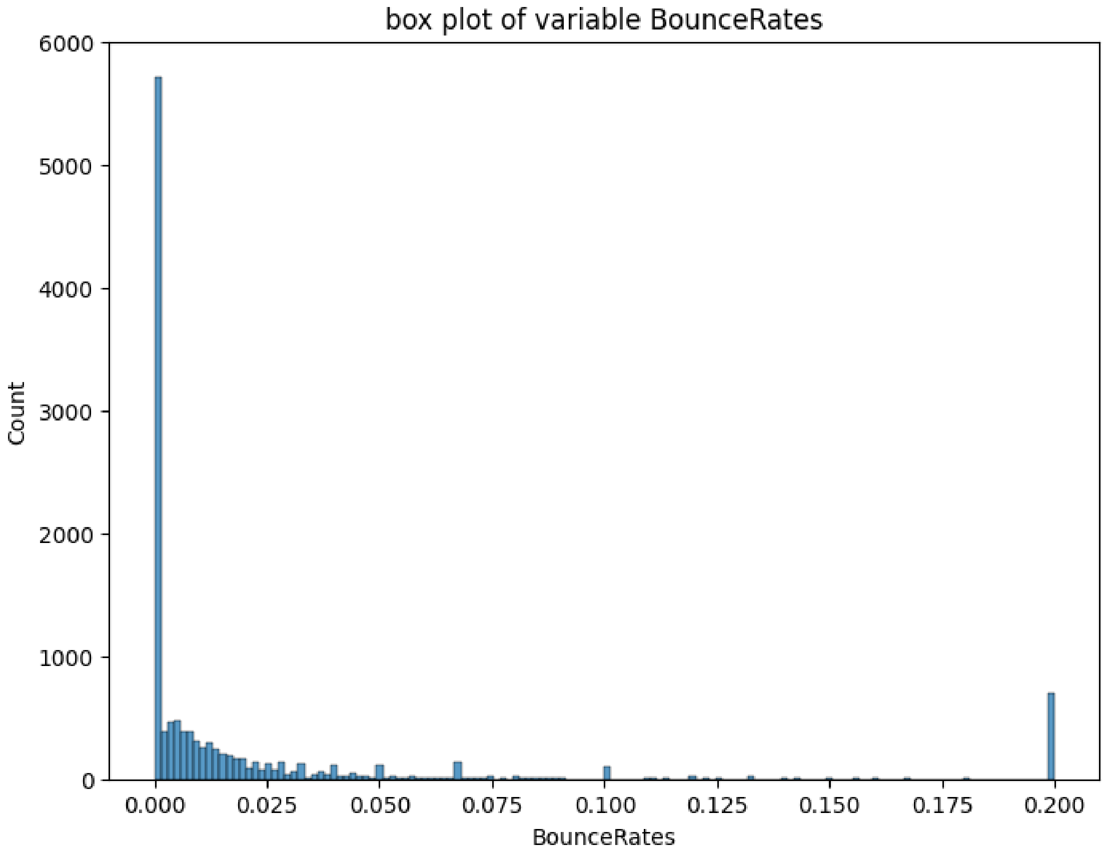


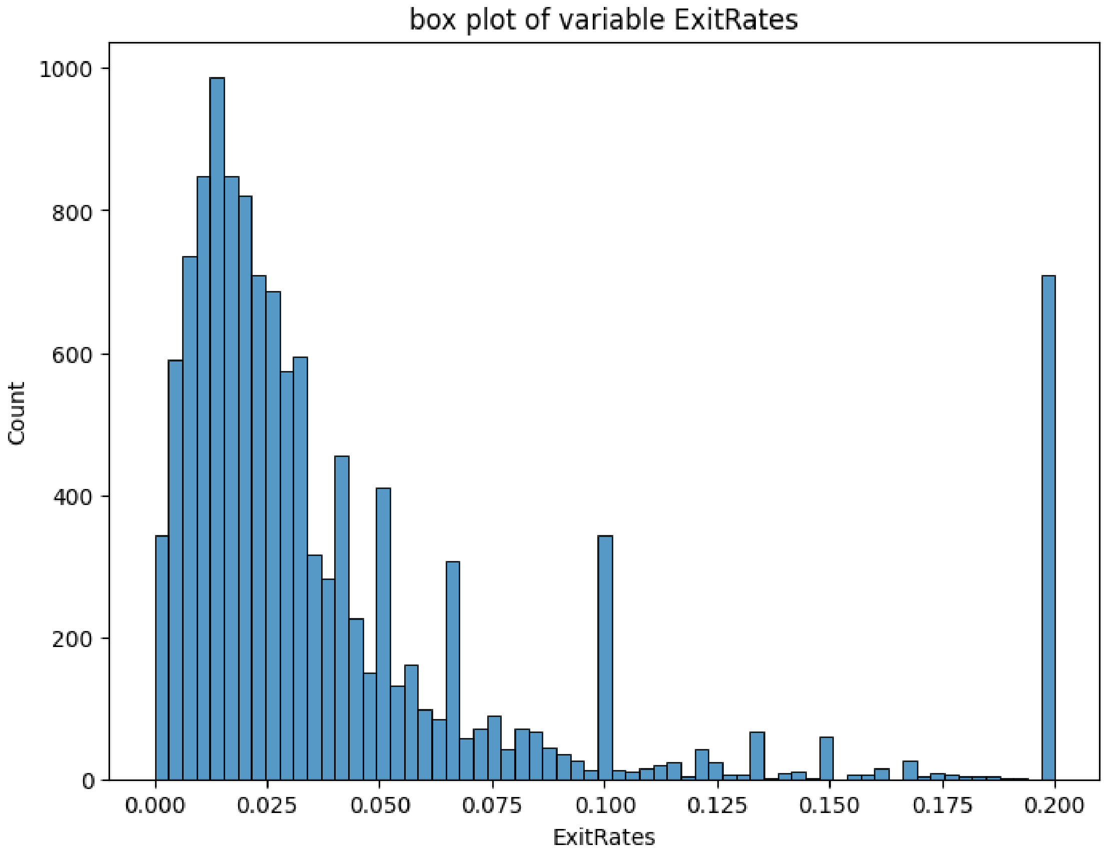
box plot of variable Informational_Duration
Count
12000
10000
8000
6000
4000
2000
0
0
500
1000
1500
2000
2500
Informational_Duration

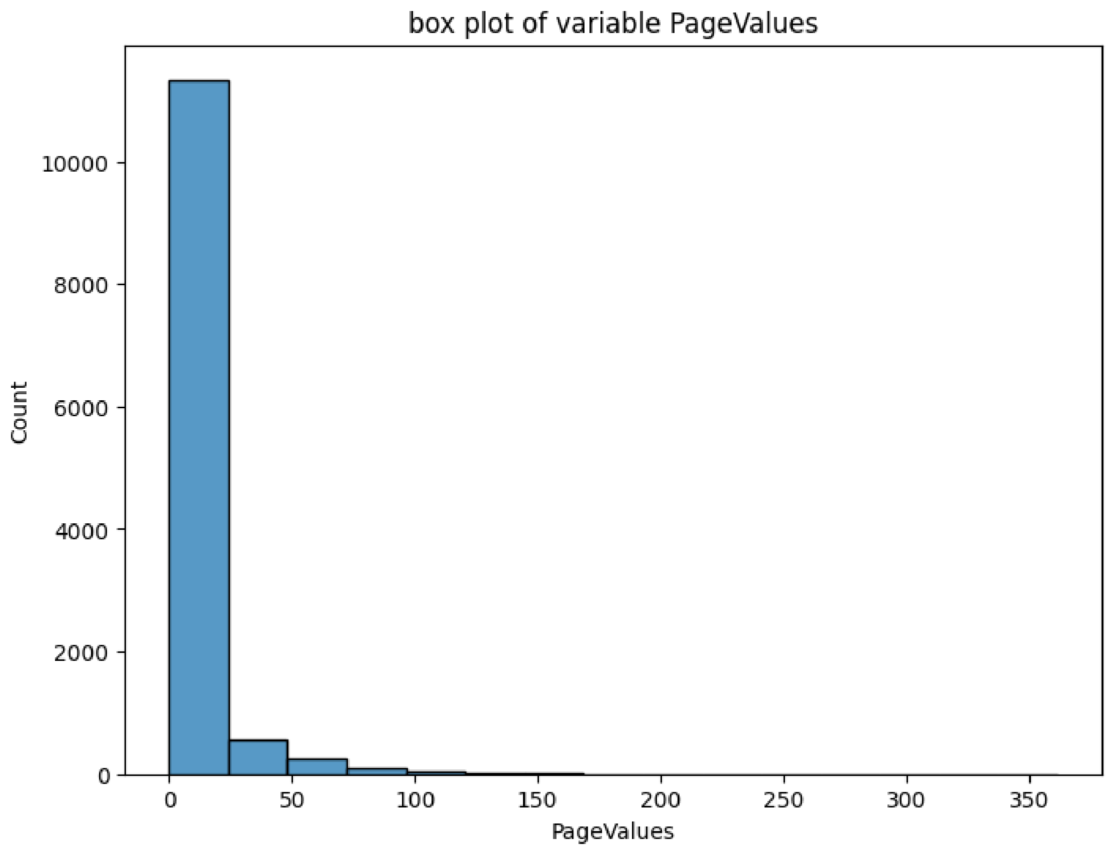

box plot of variable ProductRelated
Count
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
700
ProductRelated

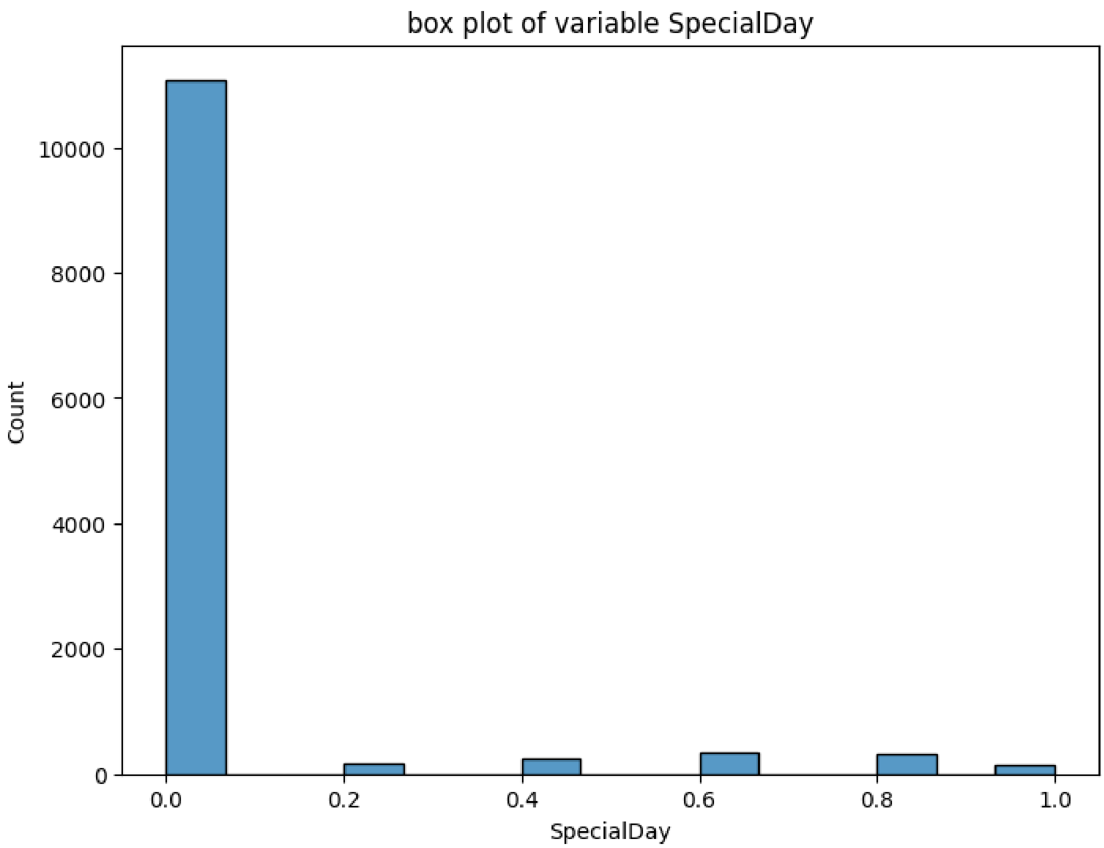


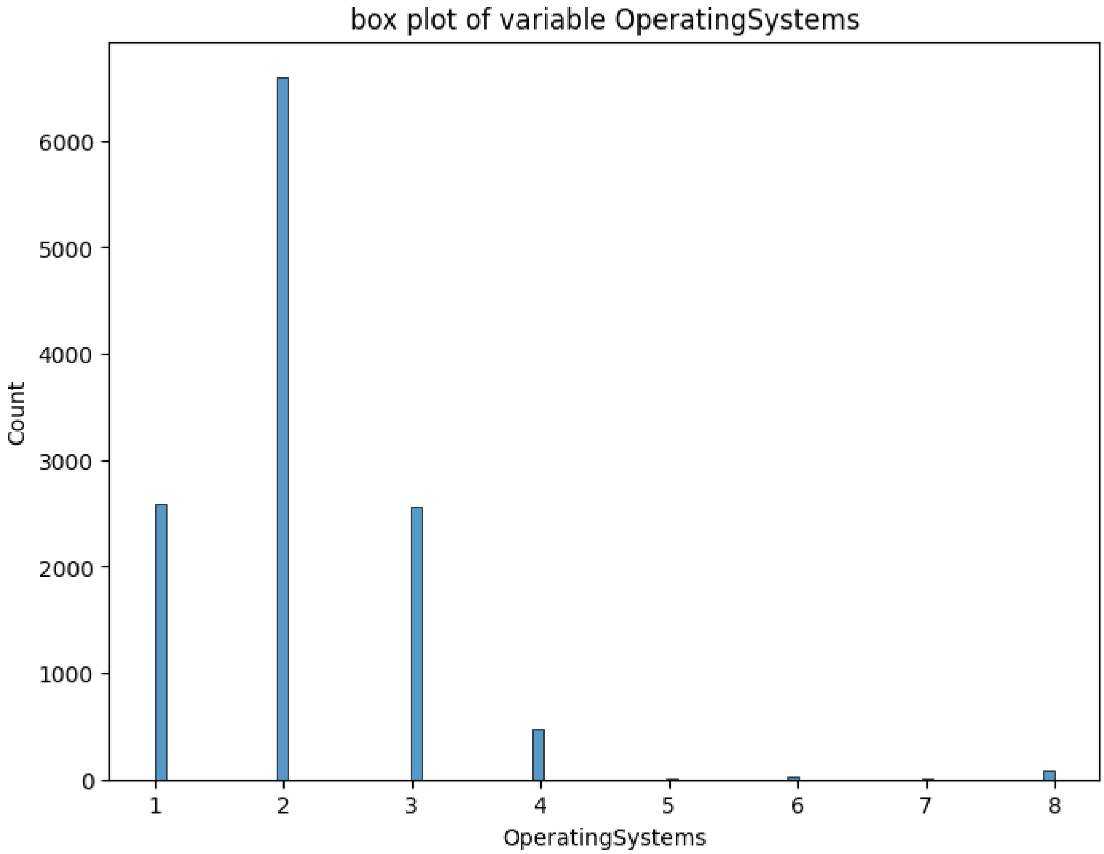
box plot of variable ProductRelated_Duration
Count
ProductRelated_Duration
0
500
1000
1500
2000
0
10000
20000
30000
40000
50000
60000

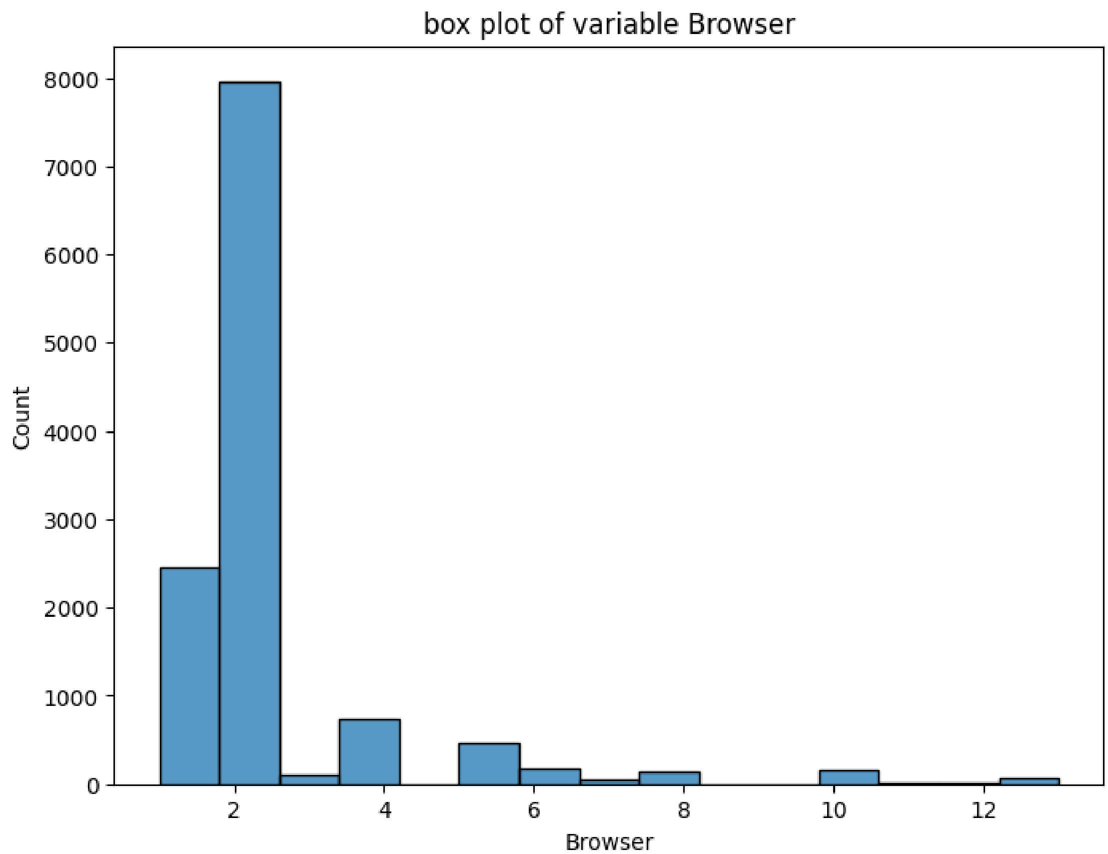

box plot of variable BounceRates
6000
5000
4000
3000
2000
1000
0
Count
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
BounceRates

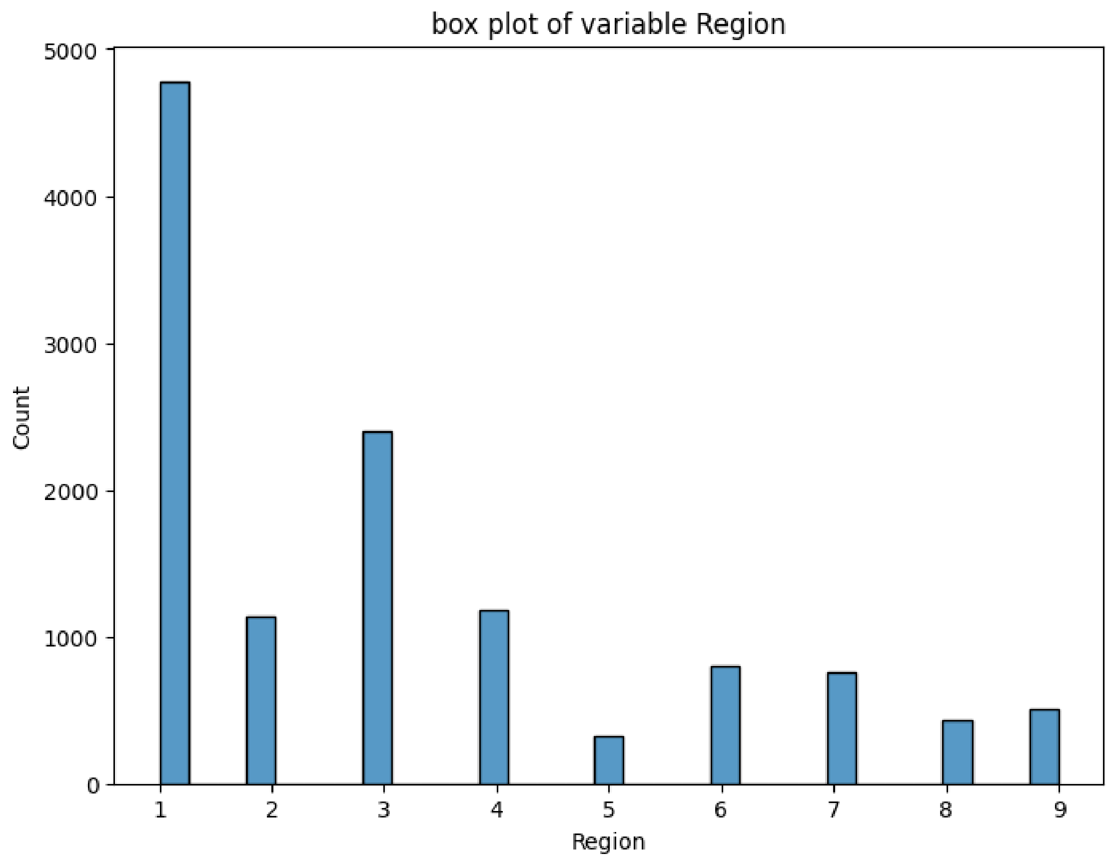

box plot of variable ExitRates
1000
800
600
400
200
0
Count
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
ExitRates

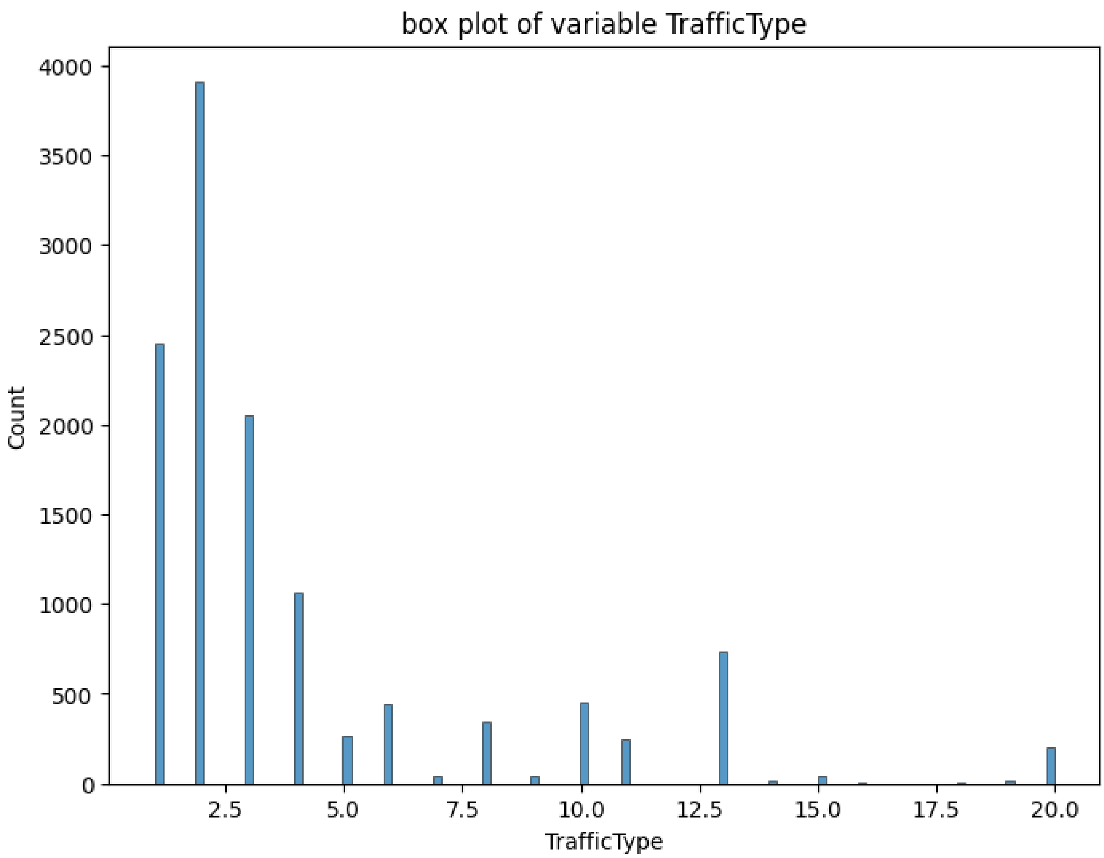

box plot of variable PageValues
Count
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300
350
PageValues


box plot of variable SpecialDay
Count
10000
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
SpecialDay


box plot of variable OperatingSystems
Count
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
OperatingSystems

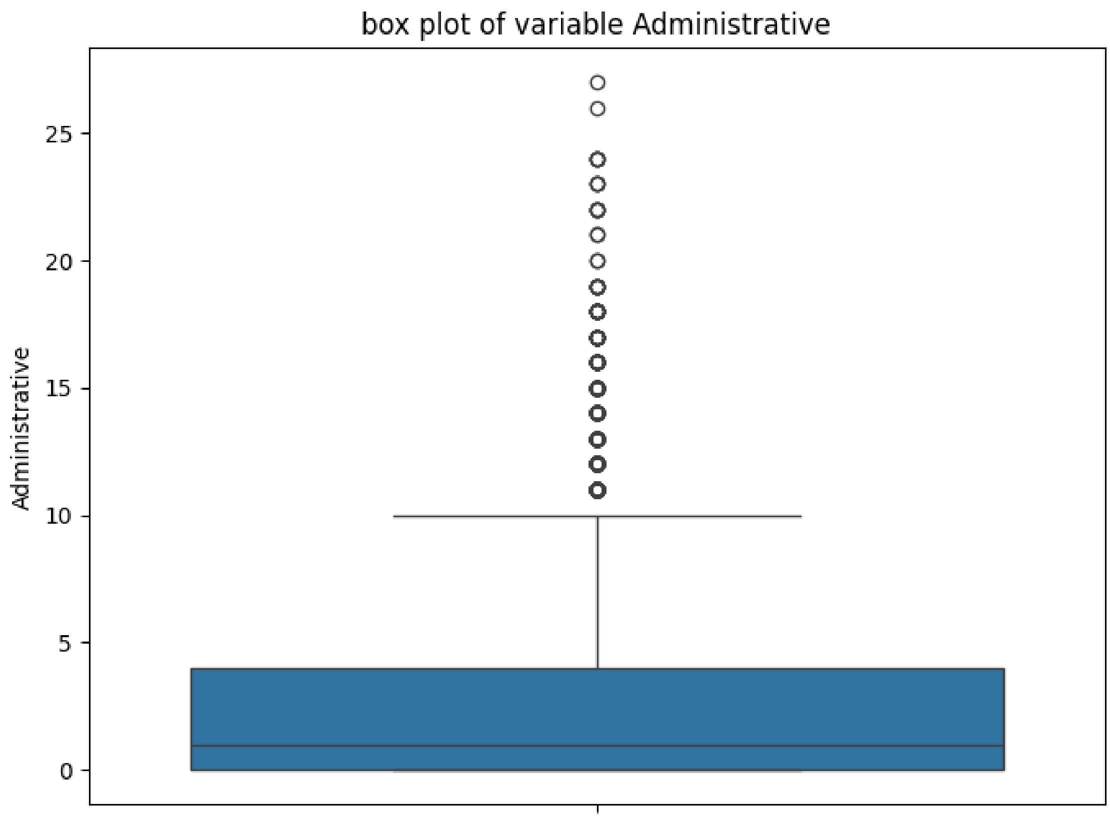

box plot of variable Browser
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
2
4
6
8
10
12
Browser


box plot of variable Region
Count
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
Region

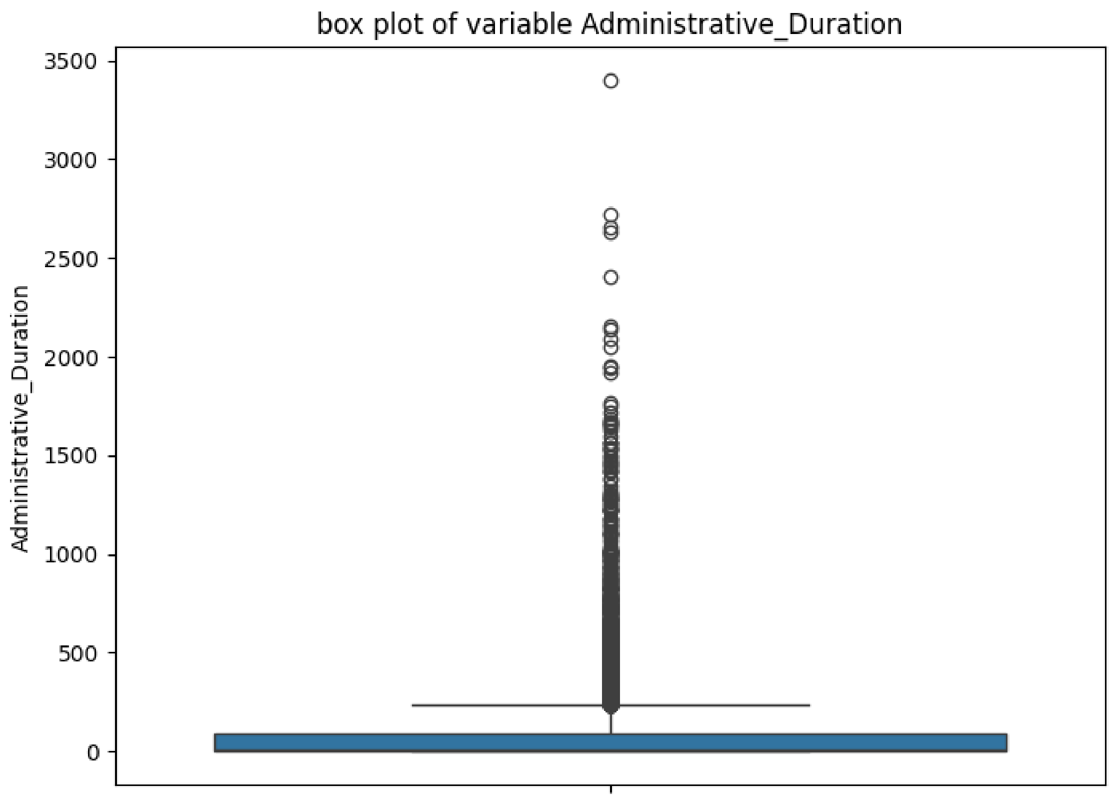


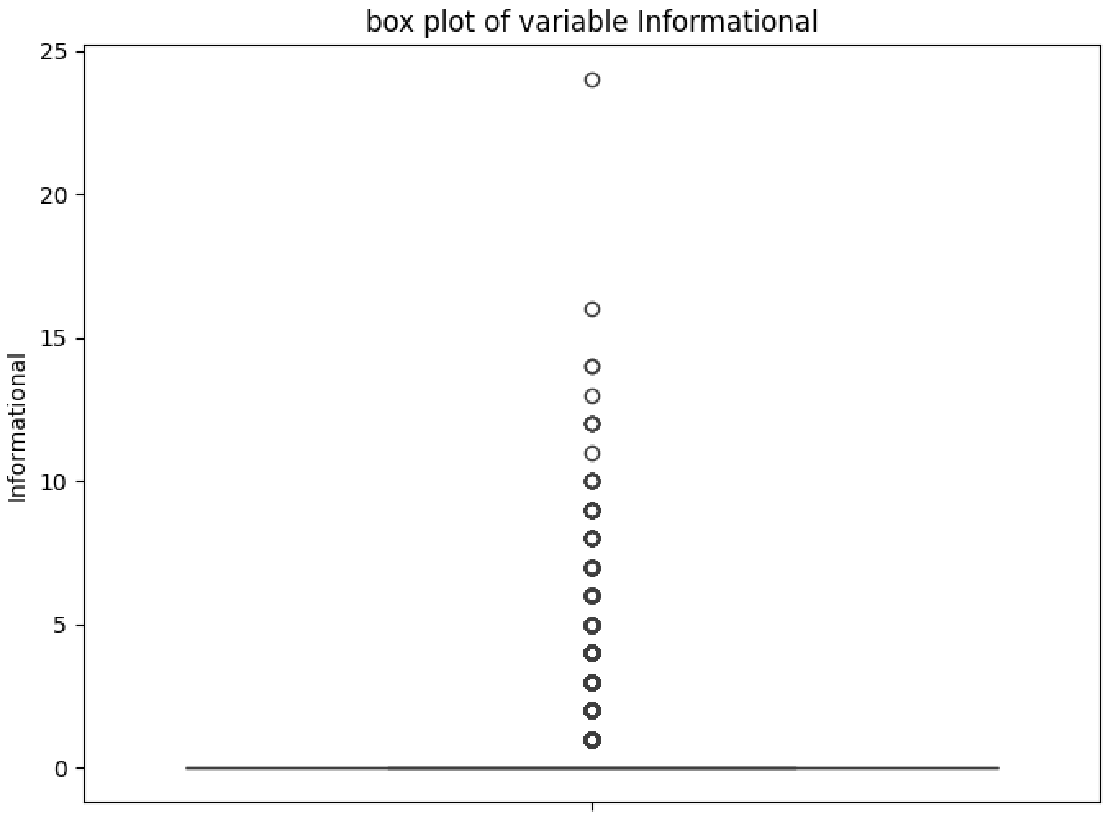

Checking for the presence of outliers in the dataset

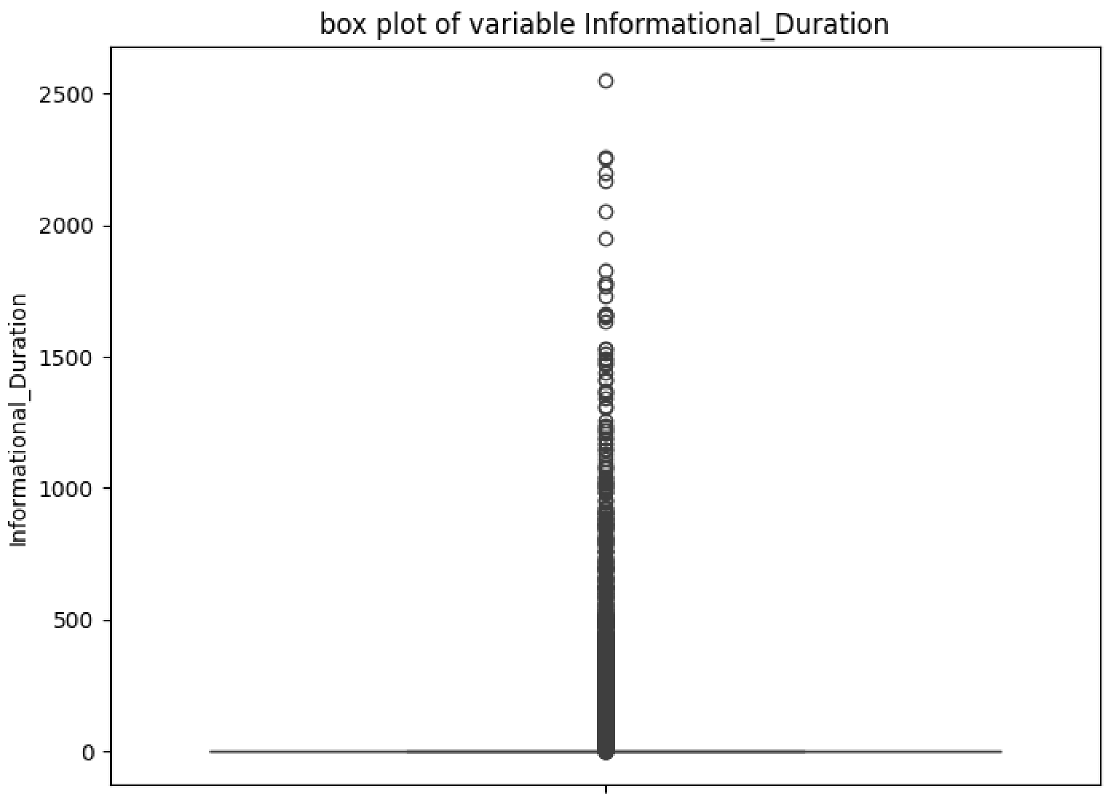
[42]:
```
numeric_variables = df.select_dtypes(include = np.number).columns
numeric_df = df[numeric_variables]

for variable in numeric_df:
  plt.figure(figsize = (8, 6))
  sns.boxplot(data = numeric_df[variable])
  plt.title(f"box plot of variable {variable}")
```

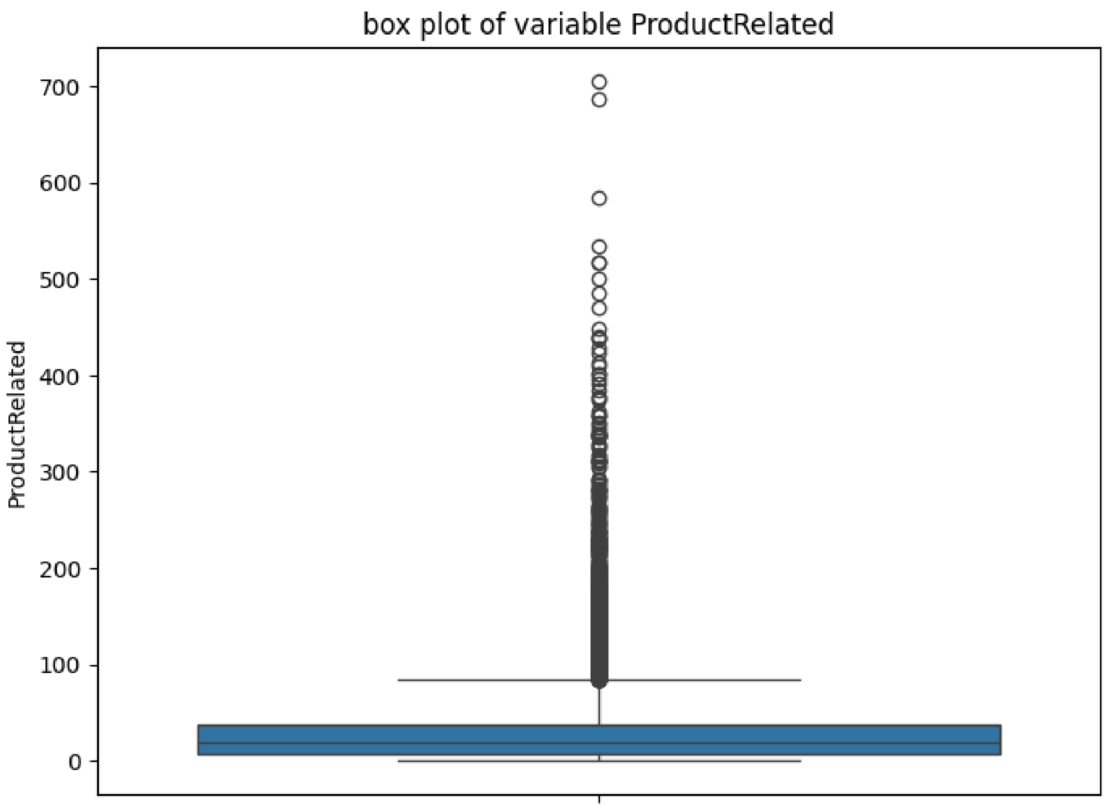


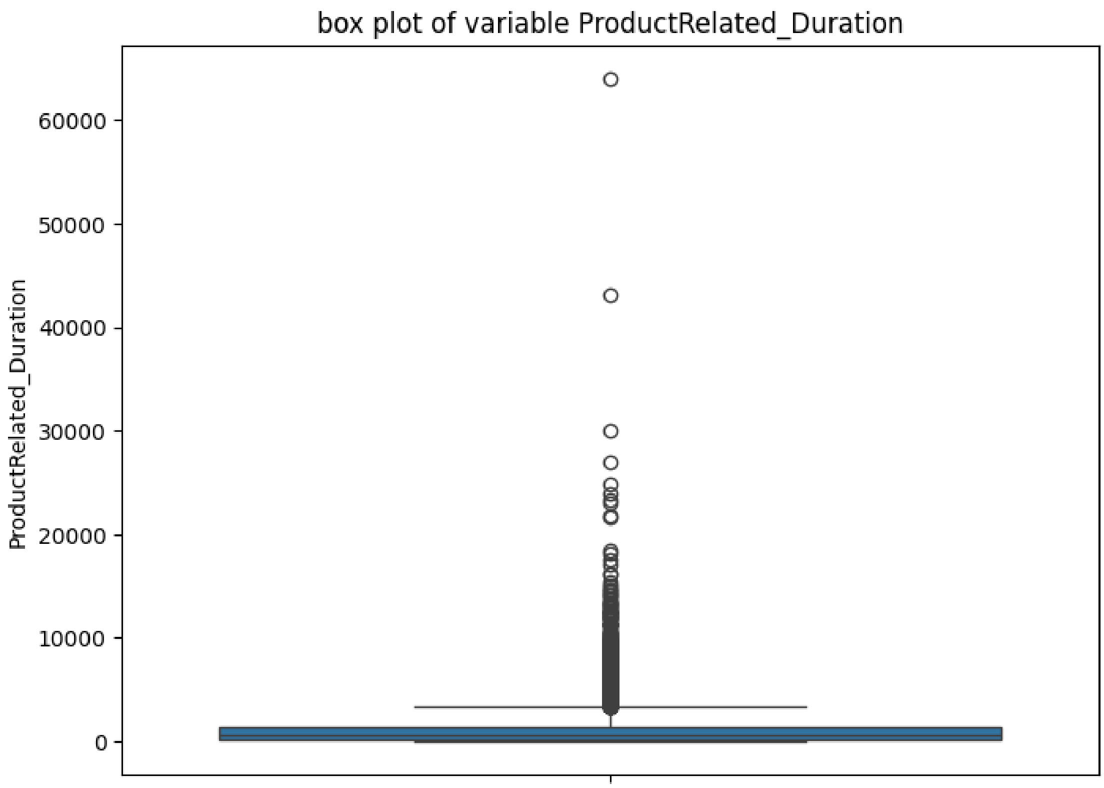
box plot of variable Administrative
Administrative
0
5
10
15
20
25

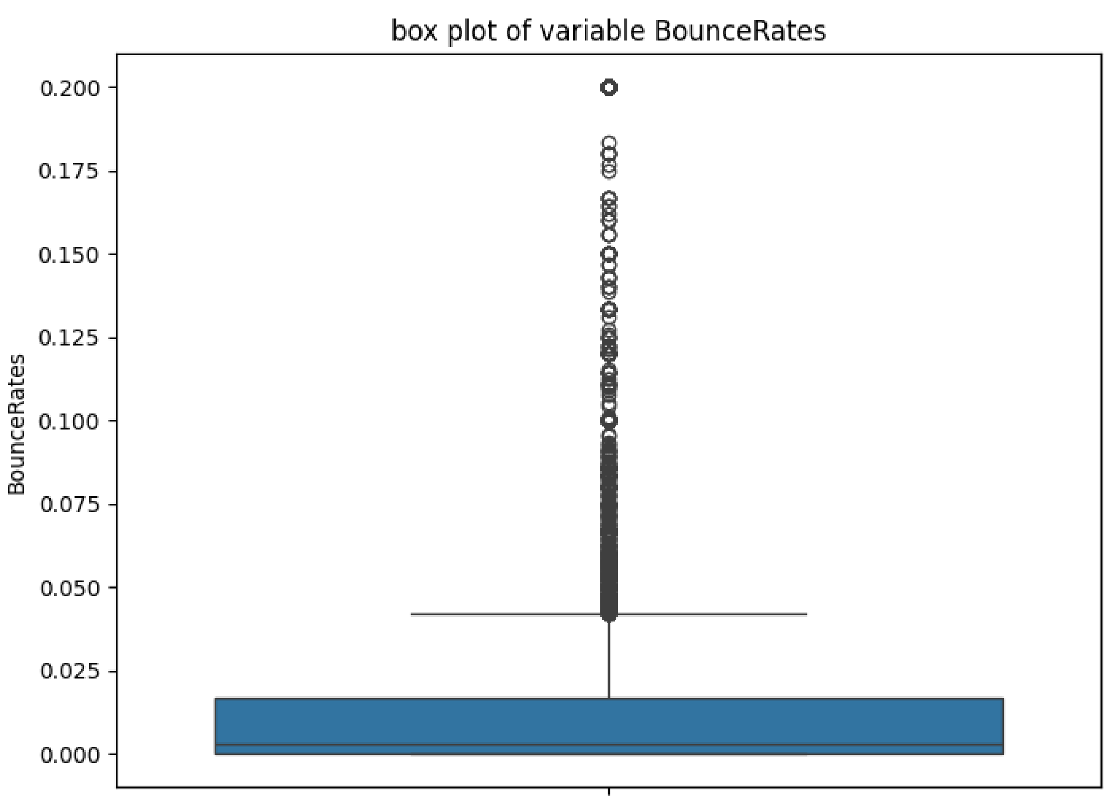

box plot of variable Administrative_Duration
Administrative_Duration
3500
3000
2500
2000
1500
1000
500
0

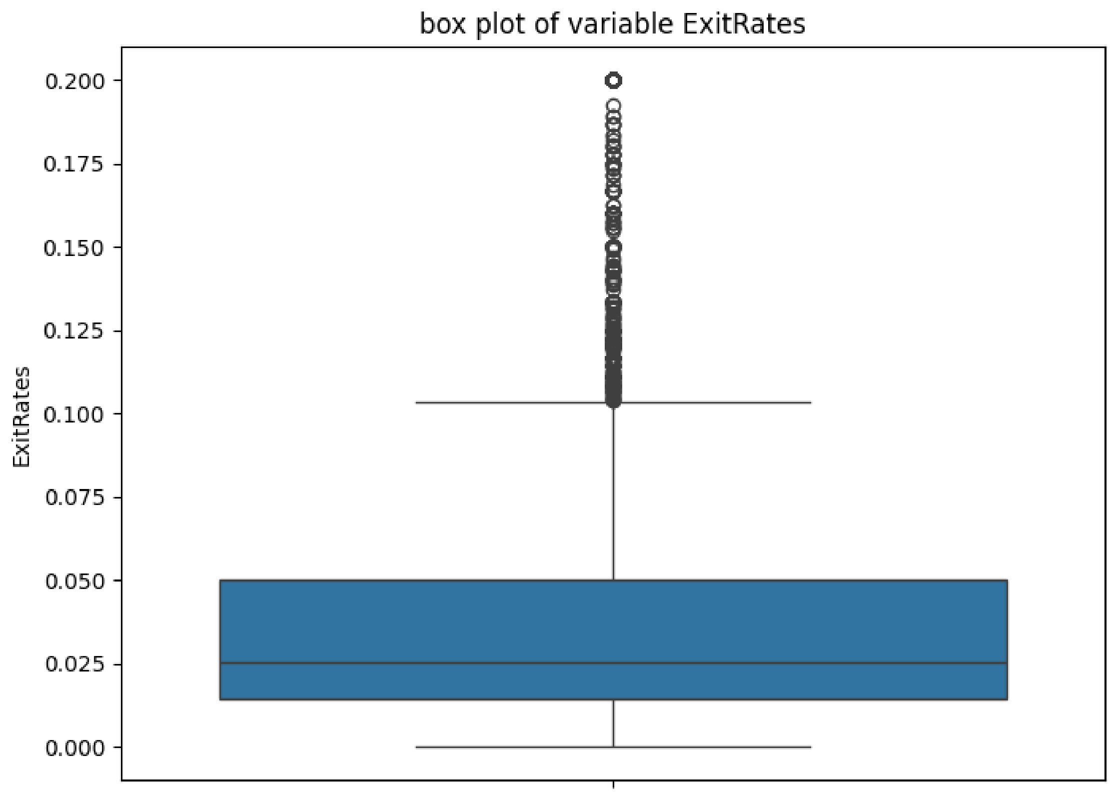

box plot of variable Informational
Informational
25
20
15
10
5
0

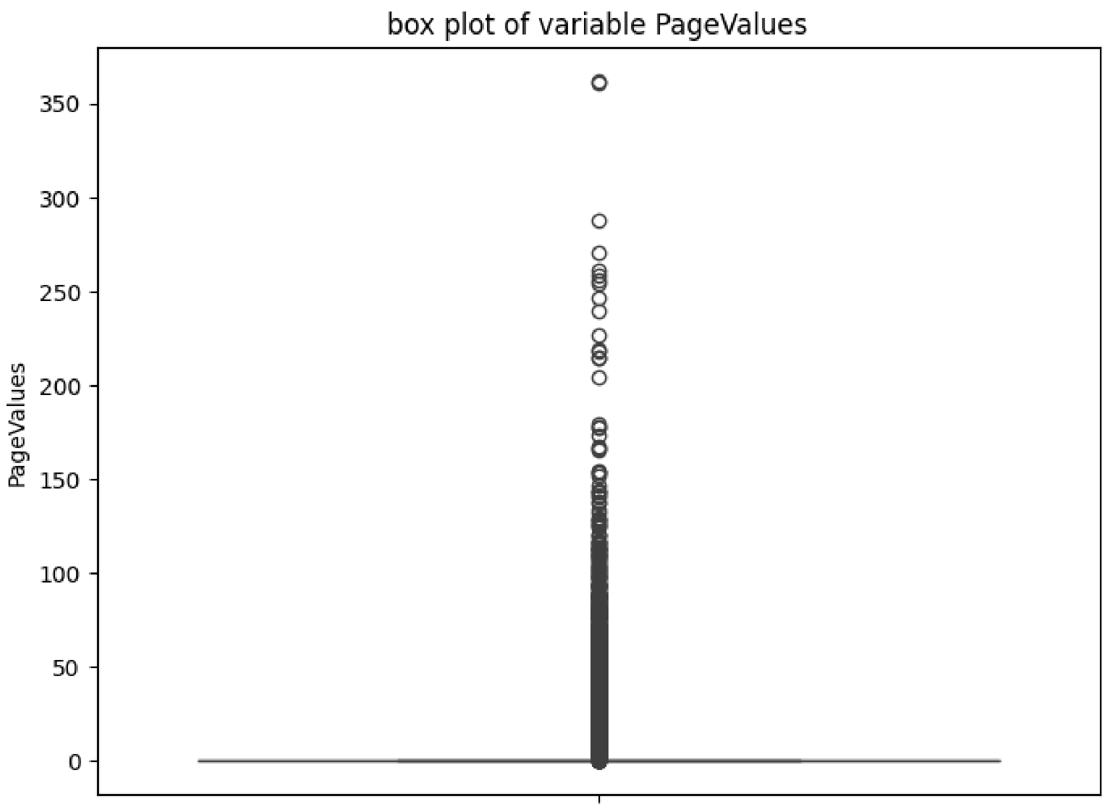

box plot of variable Informational_Duration
Informational_Duration
2500
2000
1500
1000
500
0

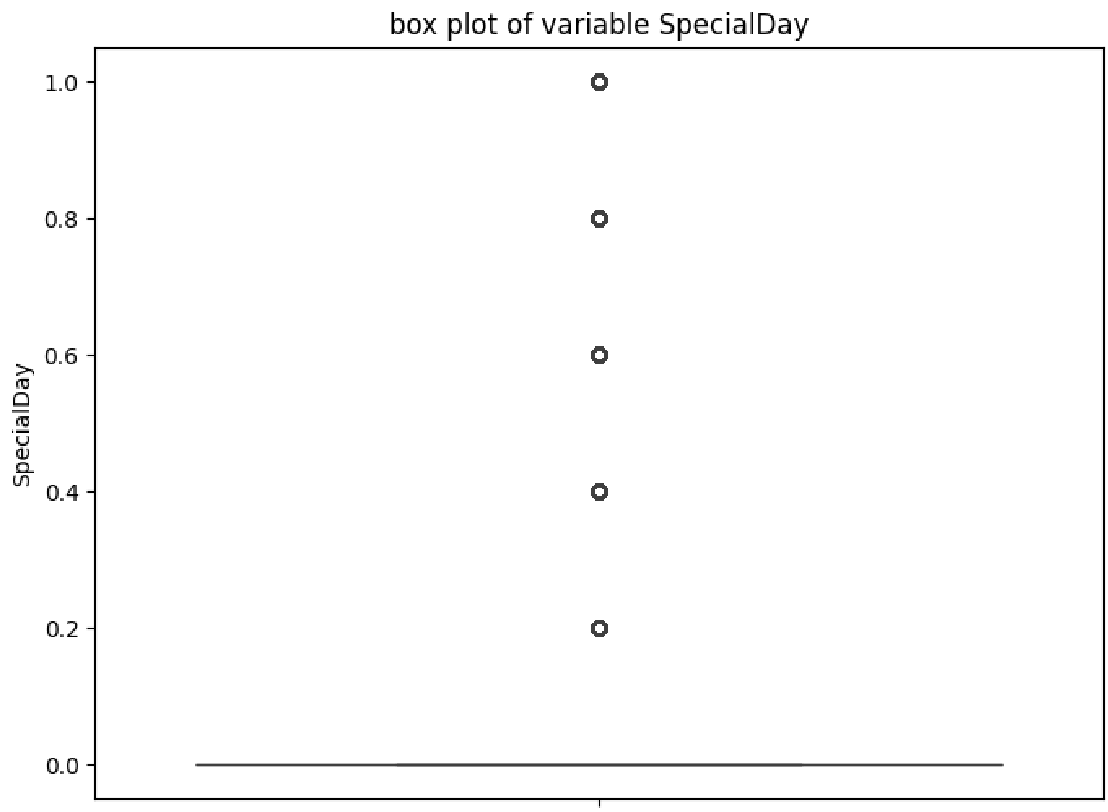

box plot of variable ProductRelated
ProductRelated
700
600
500
400
300
200
100
0

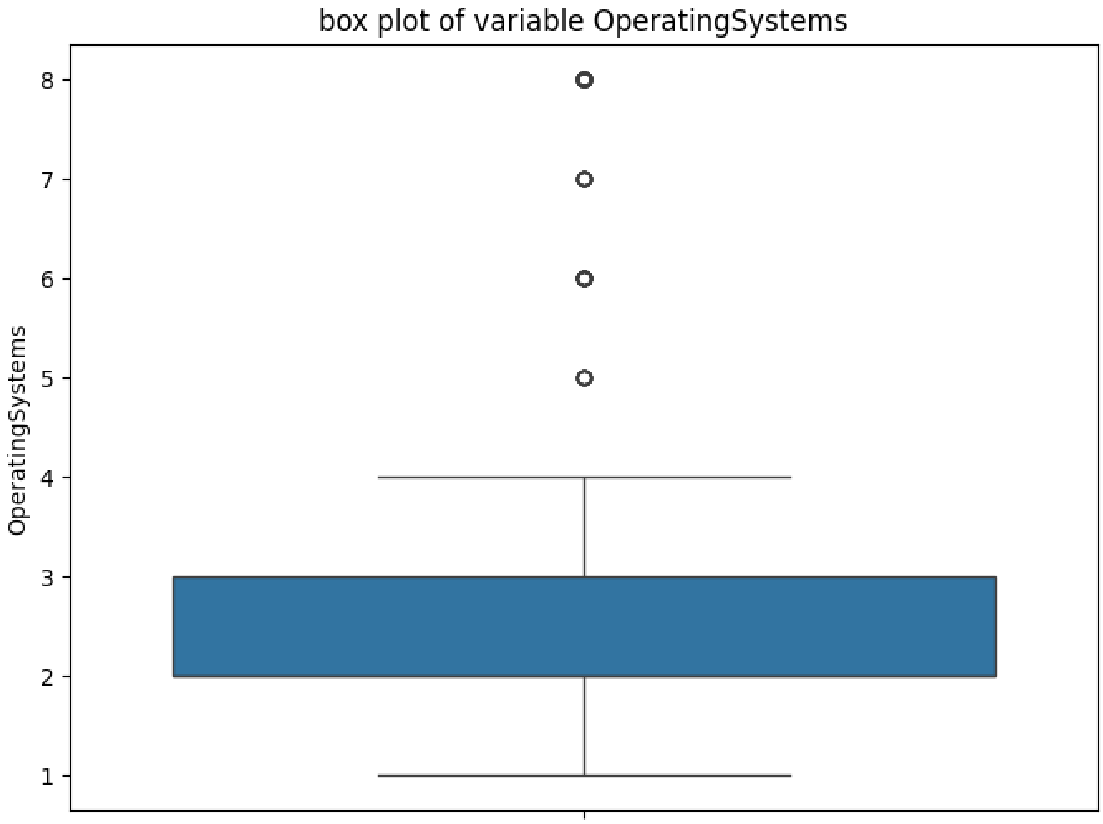

box plot of variable ProductRelated_Duration
ProductRelated_Duration
60000
50000
40000
30000
20000
10000
0

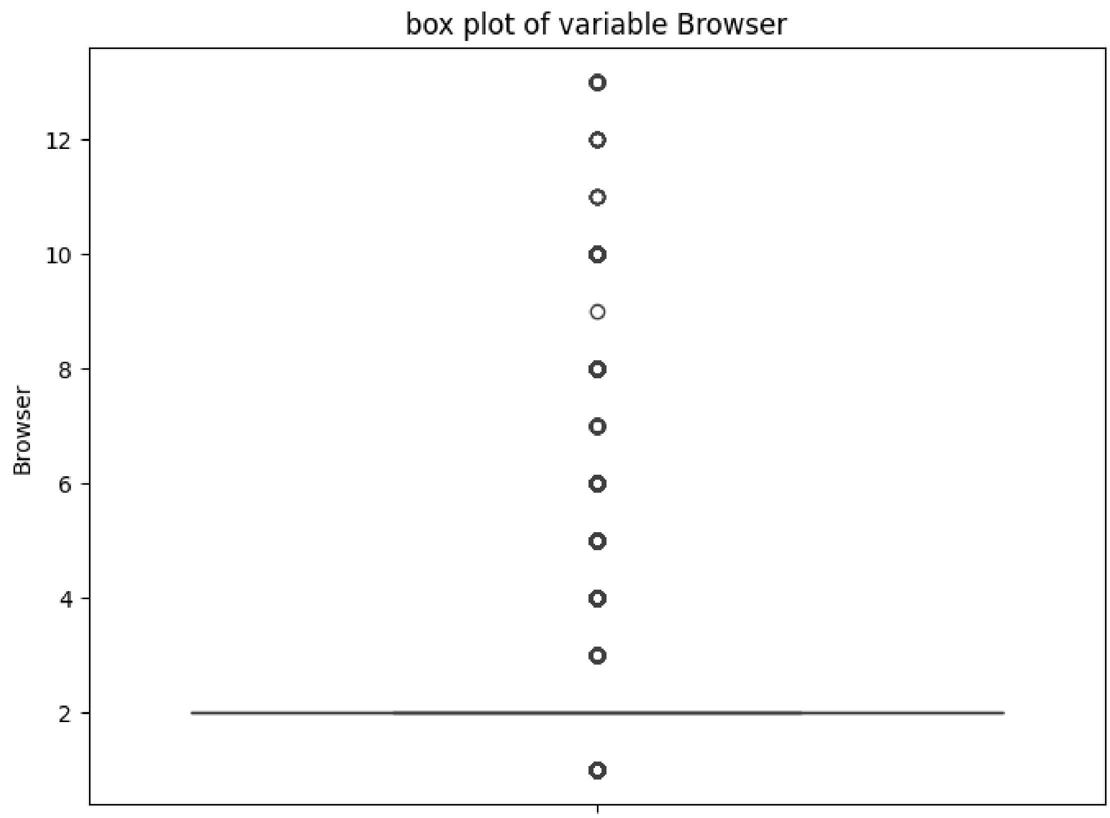

box plot of variable BounceRates
BounceRates
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000


box plot of variable ExitRates
ExitRates
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

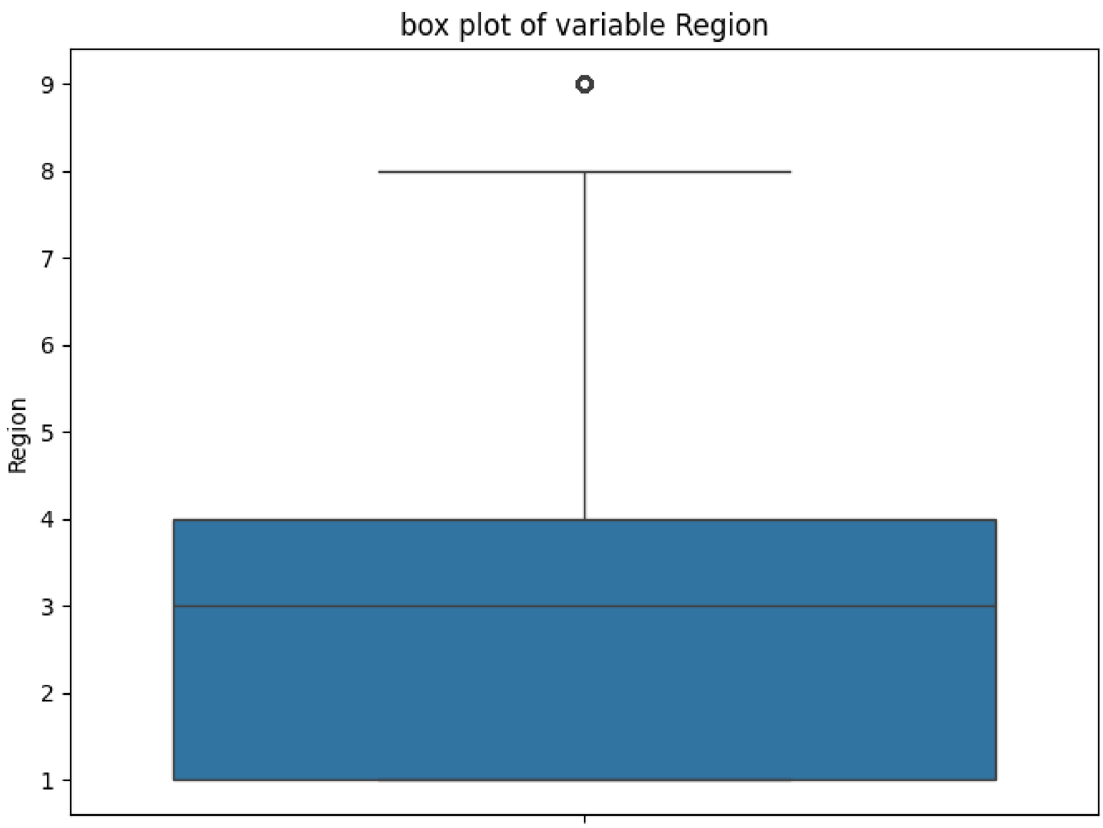

box plot of variable PageValues
PageValues
350
300
250
200
150
100
50
0

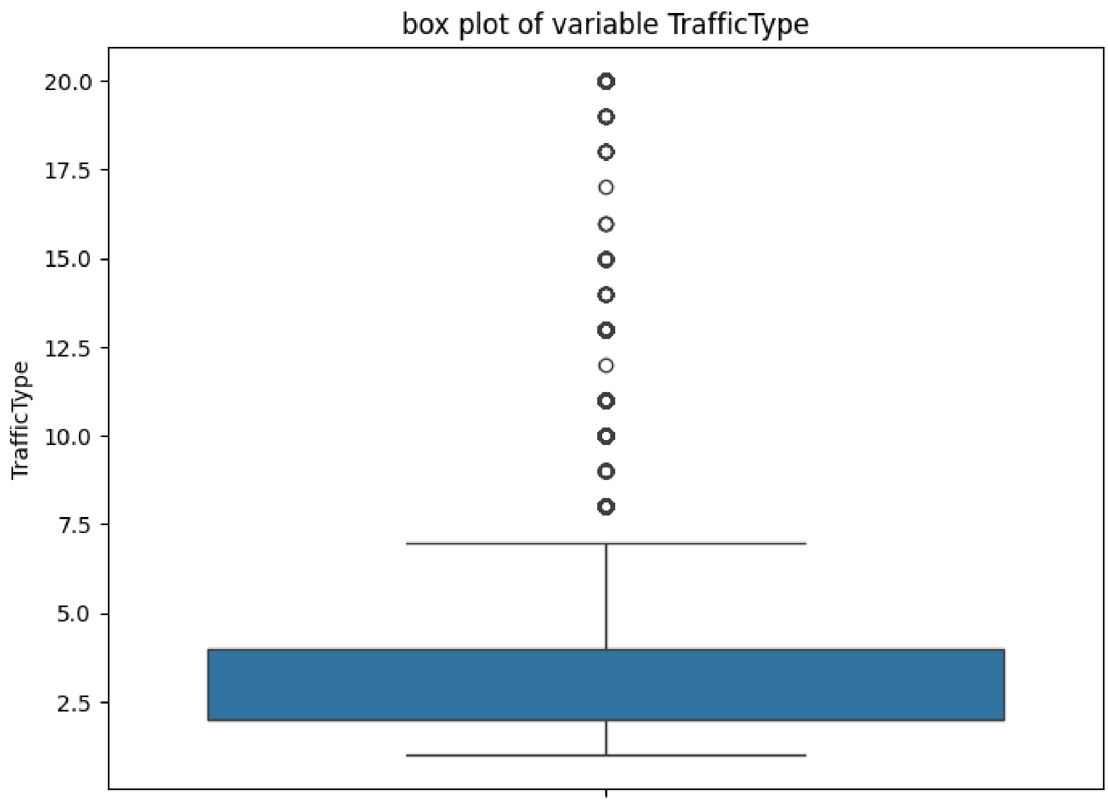

box plot of variable SpecialDay
SpecialDay
1.0
0.8
0.6
0.4
0.2
0.0


box plot of variable OperatingSystems
OperatingSystems
8
7
6
5
4
3
2
1



box plot of variable Browser
Browser
12
10
8
6
4
2


box plot of variable Region
Region
9
8
7
6
5
4
3
2
1




Calculating Total number of outliers

[9]:
```
Q1 = summary_df[numeric_variables].loc["25%"]
Q3 = summary_df[numeric_variables].loc["75%"]

IQR = Q3- Q1
print(IQR)
```

```
Administrative                 4.000000
Administrative_Duration       93.256250
Informational                  0.000000
Informational_Duration         0.000000
ProductRelated                31.000000
ProductRelated_Duration     1280.019714
BounceRates                    0.016813
ExitRates                      0.035714
PageValues                     0.000000
SpecialDay                     0.000000
OperatingSystems               1.000000
Browser                        0.000000
Region                         3.000000
TrafficType                    2.000000
```

```
dtype: float64
```

[10]:
```
lower_bound = Q1- 1.5*IQR
upper_bound = Q3 + 1.5*IQR

bounds_df = pd.DataFrame({"LowerBound" : lower_bound, "UpperBound":␣
 ↪upper_bound})
print(bounds_df)
```

```
                          LowerBound   UpperBound
Administrative             -6.000000    10.000000
Administrative_Duration  -139.884375   233.140625
Informational               0.000000     0.000000
Informational_Duration      0.000000     0.000000
ProductRelated            -39.500000    84.500000
ProductRelated_Duration -1735.892070  3384.186784
BounceRates                -0.025219     0.042031
ExitRates                  -0.039286     0.103571
PageValues                  0.000000     0.000000
SpecialDay                  0.000000     0.000000
OperatingSystems            0.500000     4.500000
Browser                     2.000000     2.000000
Region                     -3.500000     8.500000
TrafficType                -1.000000     7.000000
```

[11]:
```
outliers_lower = (summary_df[numeric_variables] < lower_bound).sum()
outliers_upper = (summary_df[numeric_variables] > upper_bound).sum()
total_outliers = outliers_lower  + outliers_upper

ouliers_count_df = pd.DataFrame({"LowerBound_outliers" :outliers_lower,␣
 ↪"UpperBound_outliers" :outliers_upper, "Total" : total_outliers})
print(ouliers_count_df)
```

```
                         LowerBound_outliers  UpperBound_outliers  Total
Administrative                             0                    2      2
Administrative_Duration                    0                    2      2
Informational                              0                    4      4
Informational_Duration                     0                    4      4
ProductRelated                             0                    2      2
ProductRelated_Duration                    0                    2      2
BounceRates                                0                    3      3
ExitRates                                  0                    2      2
PageValues                                 0                    4      4
SpecialDay                                 0                    4      4
OperatingSystems                           0                    2      2
Browser                                    2                    3      5
Region                                     0                    2      2
TrafficType                                0                    2      2
```

Understanding correaltion between multiple numerical features

[14]:
```
variables_1 = numeric_df[['Administrative', 'Administrative_Duration',␣
↪'Informational', 'Informational_Duration', 'ProductRelated',␣
↪'ProductRelated_Duration', 'BounceRates', 'ExitRates']]
spearman_corr = variables_1.corr(method = 'spearman')
spearman_corr
```

[14]:
```
                         Administrative  Administrative_Duration  \
Administrative                 1.000000                 0.940725
Administrative_Duration        0.940725                 1.000000
Informational                  0.369194                 0.357150
Informational_Duration         0.362861                 0.352060
ProductRelated                 0.460204                 0.430072
ProductRelated_Duration        0.421613                 0.413765
BounceRates                   -0.155219                -0.163609
ExitRates                     -0.434389                -0.437912

                         Informational  Informational_Duration  \
Administrative                0.369194                0.362861
Administrative_Duration       0.357150                0.352060
Informational                 1.000000                0.950958
Informational_Duration        0.950958                1.000000
ProductRelated                0.368673                0.361032
ProductRelated_Duration       0.367522                0.362720
BounceRates                   0.005753               -0.002474
ExitRates                    -0.185691               -0.200056

                         ProductRelated  ProductRelated_Duration  BounceRates  \
Administrative                 0.460204                 0.421613    -0.155219
Administrative_Duration        0.430072                 0.413765    -0.163609
Informational                  0.368673                 0.367522     0.005753
Informational_Duration         0.361032                 0.362720    -0.002474
ProductRelated                 1.000000                 0.882672    -0.052305
ProductRelated_Duration        0.882672                 1.000000    -0.079768
BounceRates                   -0.052305                -0.079768     1.000000
ExitRates                     -0.518920                -0.476935     0.602276

                         ExitRates
Administrative           -0.434389
Administrative_Duration  -0.437912
Informational            -0.185691
Informational_Duration   -0.200056
ProductRelated           -0.518920
ProductRelated_Duration  -0.476935
BounceRates               0.602276
ExitRates                 1.000000
```

Undersatnding correlation between Pages visits and Page values

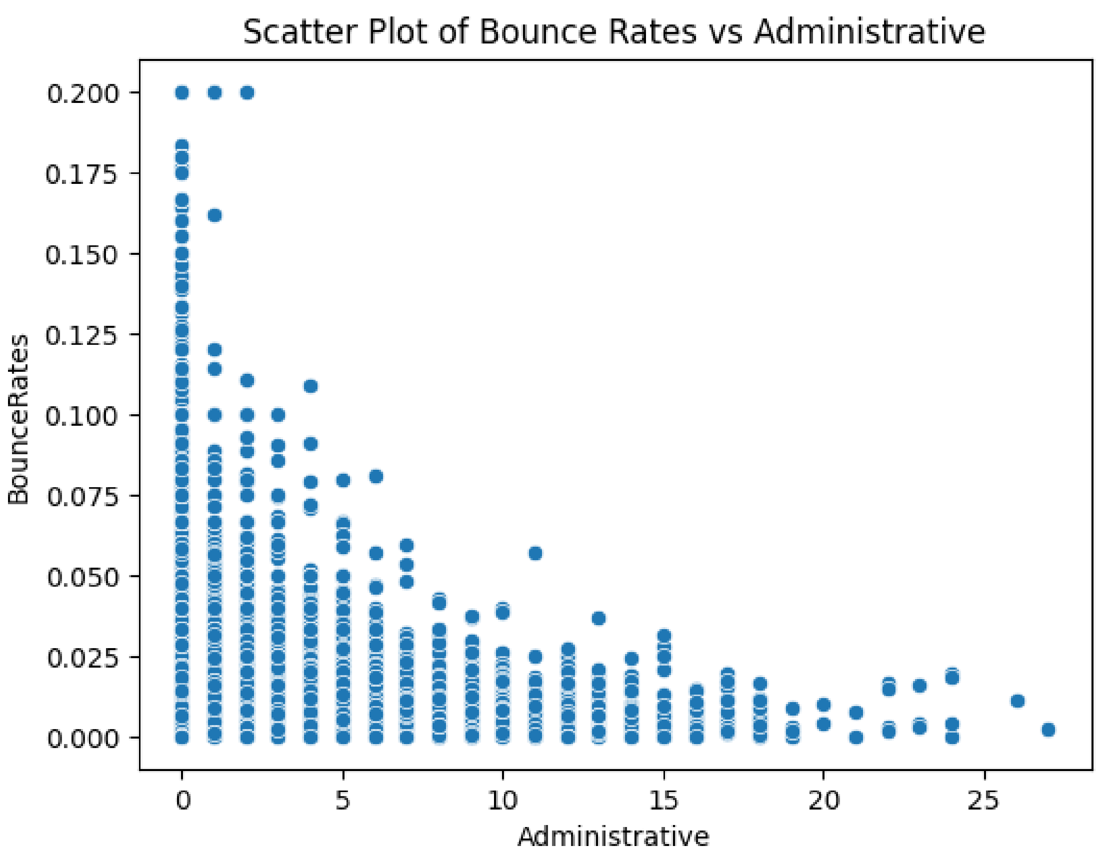
[18]:
```
variables_2 = numeric_df[['Administrative', 'Informational', 'ProductRelated',␣
 ↪'PageValues']]
spearman_corr = variables_2.corr(method = 'spearman')
spearman_corr
```

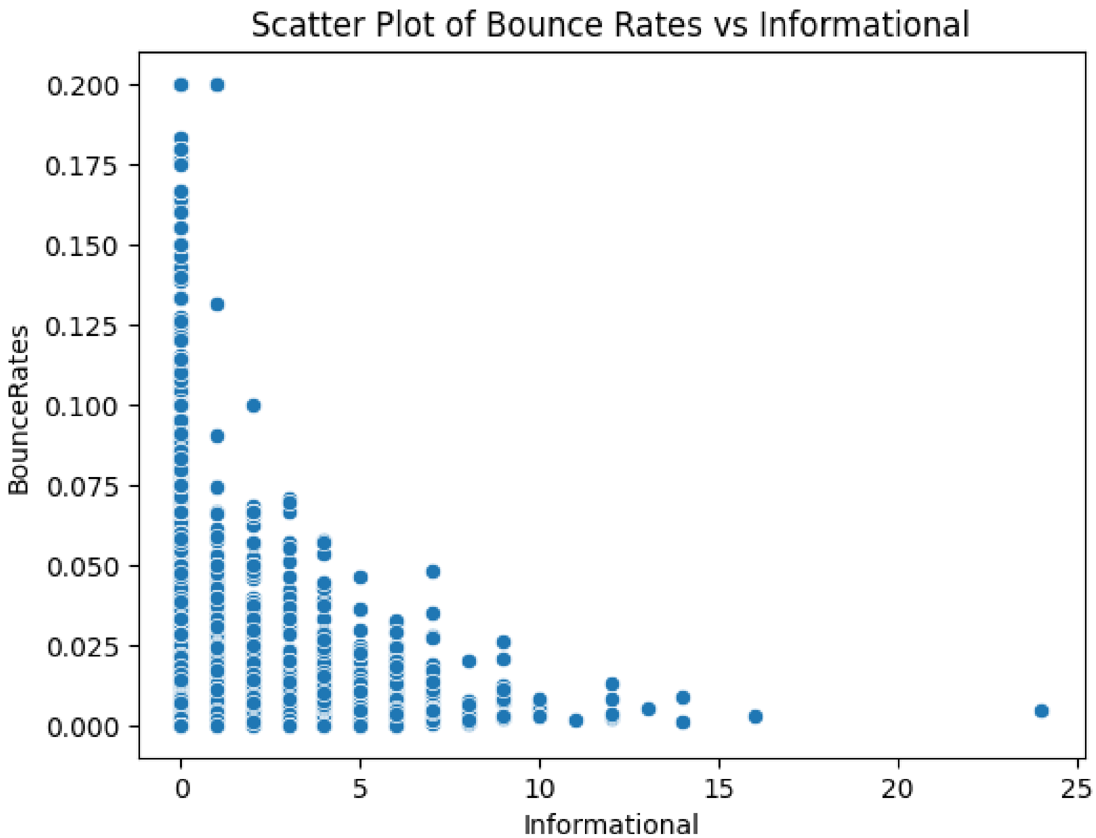
[18]:

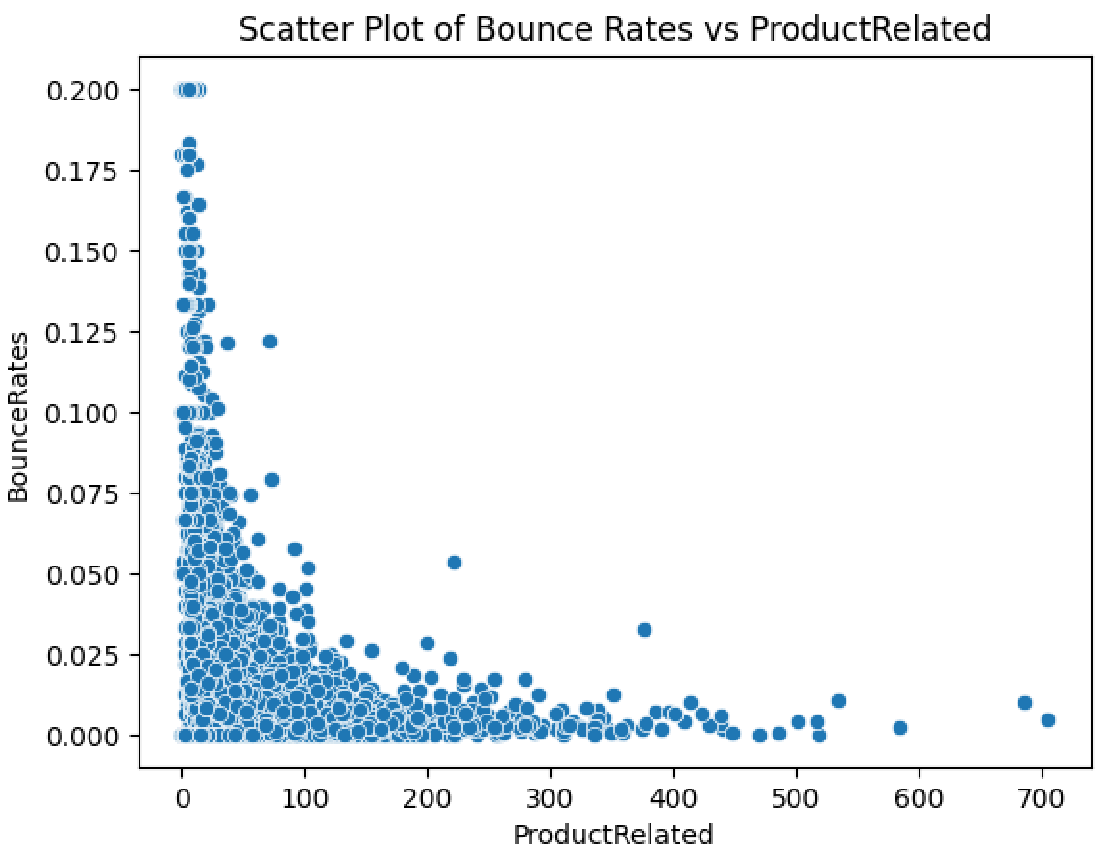
| | Administrative | Informational | ProductRelated | PageValues |
|---|---|---|---|---|
| Administrative | 1.000000 | 0.369194 | 0.460204 | 0.328350 |
| Informational | 0.369194 | 1.000000 | 0.368673 | 0.219471 |
| ProductRelated | 0.460204 | 0.368673 | 1.000000 | 0.341975 |
| PageValues | 0.328350 | 0.219471 | 0.341975 | 1.000000 |

Scatter plot showing relationship between Pages visited and bounce rate

[23]:
```
columns_to_plot = ['Administrative', 'Informational', 'ProductRelated']

for col in columns_to_plot:
  sns.scatterplot(x=df[col], y=df['BounceRates'])
  plt.title(f'Scatter Plot of Bounce Rates vs {col}')
  plt.show()
```

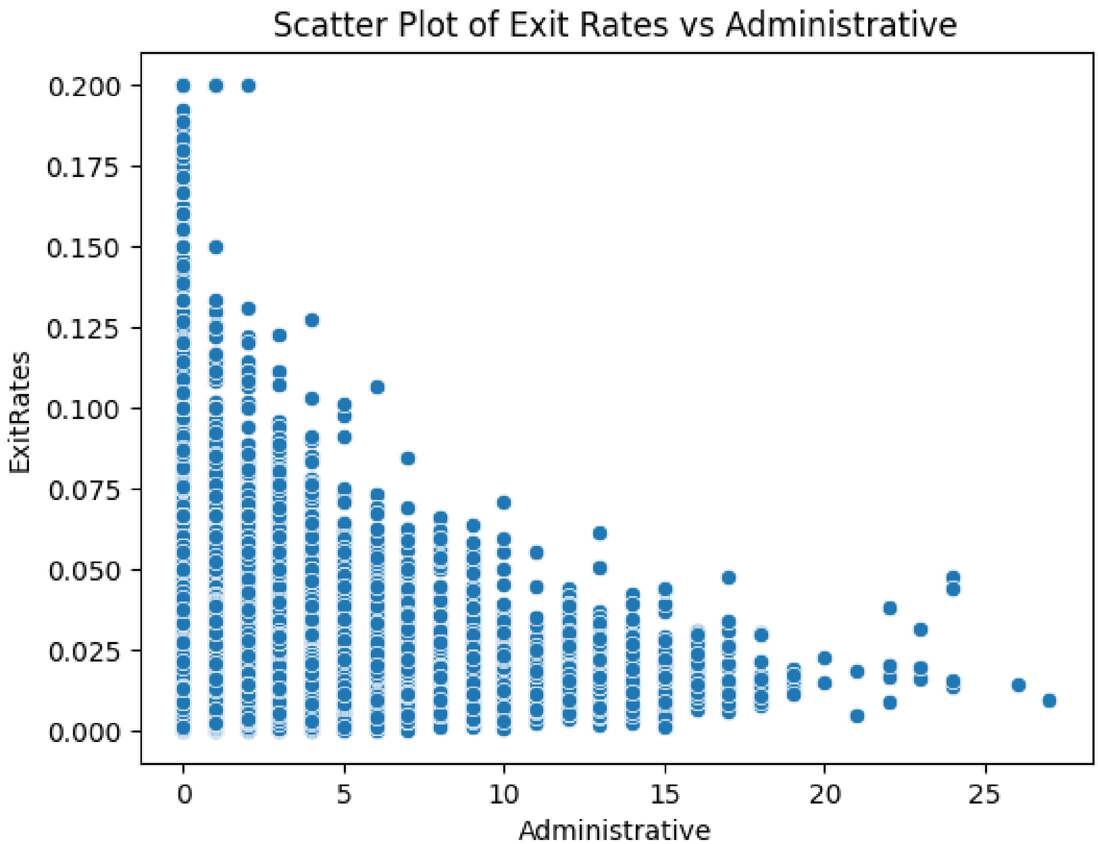


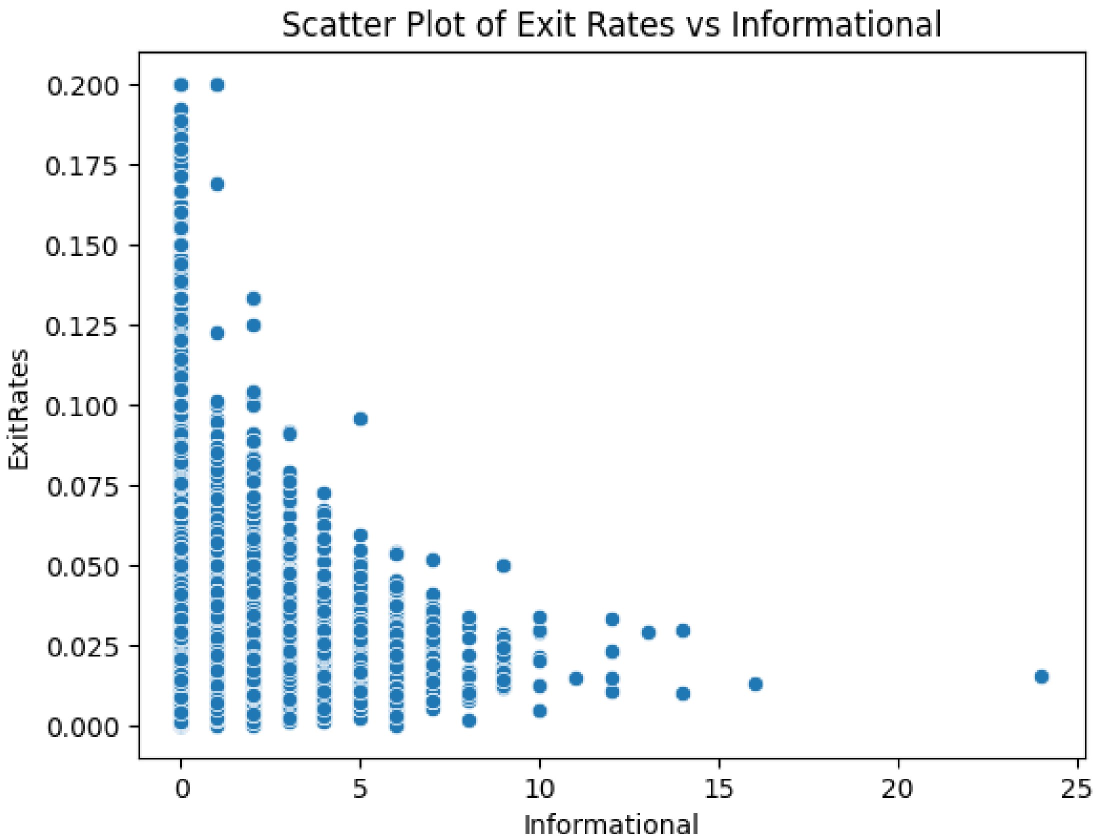

Scatter Plot of Bounce Rates vs Informational
BounceRates
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
5
10
15
20
25
Informational

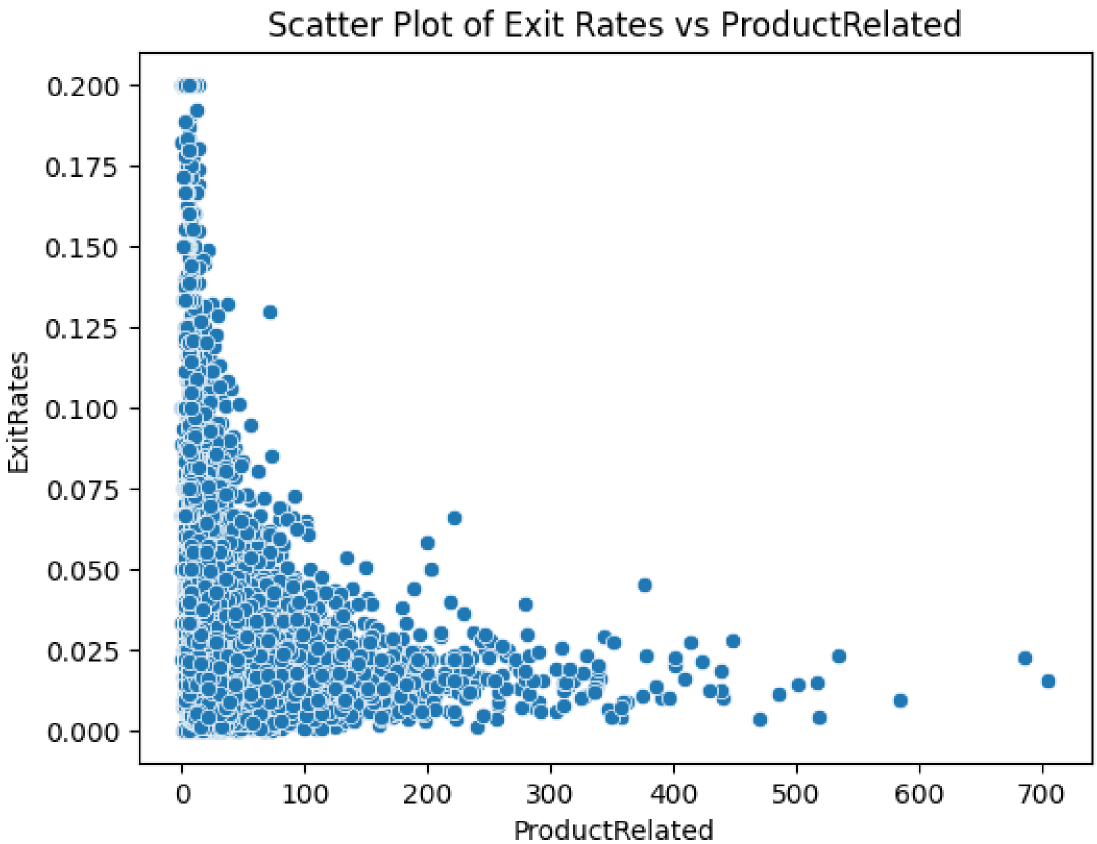



Scatter plot showing relationship between Pages visited and Exit rate

[24]:
```
columns_to_plot = ['Administrative', 'Informational', 'ProductRelated']

for col in columns_to_plot:
  sns.scatterplot(x=df[col], y=df['ExitRates'])
  plt.title(f'Scatter Plot of Exit Rates vs {col}')
  plt.show()
```


Scatter Plot of Exit Rates vs Administrative
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
ExitRates
0
5
10
15
20
25
Administrative


Scatter Plot of Exit Rates vs Informational
ExitRates
Informational
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
5
10
15
20
25




Total pages visited & Individual Revenue contribution for each type of Operating system

[24]:
```
pages_under_OS = df.groupby('OperatingSystems')[['Administrative',␣
  ↪'Informational', 'ProductRelated', 'Revenue']].sum()
pages_under_OS = pages_under_OS.sort_values(by = ['Administrative',␣
  ↪'Informational', 'ProductRelated'], ascending = False).reset_index()
pages_under_OS
```

[24]:

| | OperatingSystems | Administrative | Informational | ProductRelated | Revenue |
|---|---|---|---|---|---|
| 0 | 2 | 15620 | 3329 | 243291 | 1155 |
| 1 | 3 | 5930 | 1402 | 67616 | 268 |
| 2 | 1 | 5756 | 1275 | 61224 | 379 |
| 3 | 4 | 1049 | 171 | 17317 | 85 |
| 4 | 8 | 123 | 17 | 1182 | 17 |
| 5 | 6 | 39 | 10 | 406 | 2 |
| 6 | 7 | 25 | 4 | 140 | 1 |
| 7 | 5 | 4 | 1 | 73 | 1 |

Total pages visited & Individual Revenue contribution for each type of Web browser

```
[29]: pages_under_browser = df.groupby('Browser')[['Administrative', 'Informational',␣
       ↪'ProductRelated', 'Revenue']].sum()
      pages_under_browser = pages_under_browser.sort_values(by = ['Administrative',␣
       ↪'Informational', 'ProductRelated'], ascending = False).reset_index()
      pages_under_browser
```

[29]:

| | Browser | Administrative | Informational | ProductRelated | Revenue |
|---|---|---|---|---|---|
| 0 | 2 | 19345 | 4327 | 276985 | 1223 |
| 1 | 1 | 5376 | 1200 | 60086 | 365 |
| 2 | 4 | 1460 | 268 | 22411 | 130 |
| 3 | 5 | 1056 | 183 | 14633 | 86 |
| 4 | 10 | 376 | 72 | 5357 | 32 |
| 5 | 6 | 334 | 72 | 5198 | 20 |
| 6 | 8 | 189 | 19 | 2433 | 21 |
| 7 | 3 | 141 | 28 | 1724 | 5 |
| 8 | 7 | 137 | 23 | 1259 | 6 |
| 9 | 13 | 107 | 14 | 908 | 16 |
| 10 | 12 | 19 | 2 | 172 | 3 |
| 11 | 11 | 4 | 1 | 73 | 1 |
| 12 | 9 | 2 | 0 | 10 | 0 |

Total pages visited & Individual Revenue contribution for each type of Region

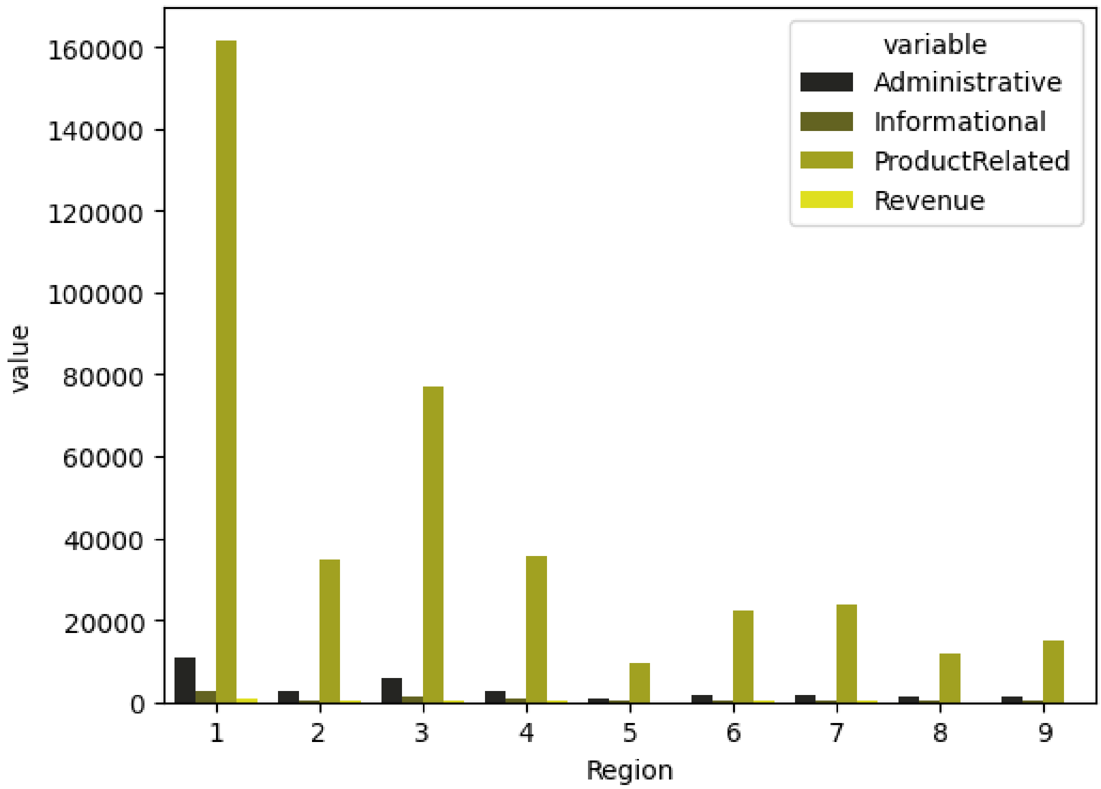
```
[30]: pages_under_Region = df.groupby('Region')[['Administrative', 'Informational',␣
       ↪'ProductRelated', 'Revenue']].sum()
      pages_under_Region = pages_under_Region.sort_values(by = ['Administrative',␣
       ↪'Informational', 'ProductRelated'], ascending = False).reset_index()
      print(pages_under_Region)

      sns.barplot(data = pages_under_Region.melt(id_vars = 'Region'), x = 'Region', y␣
       ↪= 'value', hue = 'variable', color = 'yellow')
```

```
   Region  Administrative  Informational  ProductRelated  Revenue
0       1           10857           2607          161567      771
1       3            5878           1228           77102      349
2       2            2731            506           34566      188
3       4            2722            604           35706      175
4       7            1792            339           23941      119
5       6            1655            408           22410      112
6       8            1077            196           11600       56
7       9            1051            183           14935       86
8       5             783            138            9422       52
```

```
<ipython-input-30-49afc0e1f0f6>:5: FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:yellow'` for the same effect.
```

```
sns.barplot(data = pages_under_Region.melt(id_vars = 'Region'), x = 'Region',
y = 'value', hue = 'variable', color = 'yellow')
```

[30]: <Axes: xlabel='Region', ylabel='value'>




Calculation of weekend and weekdays proportions

[25]:
```
df['Weekend'].value_counts(normalize = True).reset_index()
```

[25]:

| | Weekend | proportion |
|---|---|---|
| 0 | False | 0.767397 |
| 1 | True | 0.232603 |

Checking Total revenue class balance

[27]:
```
df['Revenue'].value_counts(normalize = True).reset_index()
```

[27]:

| | Revenue | proportion |
|---|---|---|
| 0 | False | 0.845255 |
| 1 | True | 0.154745 |

Checking for which type of visitors actually contributed more to revenue

[37]:
```
counts = df.groupby('VisitorType')['Revenue'].value_counts().reset_index()
counts
```

[37]:

| | VisitorType | Revenue | count |
|---|---|---|---|
| 0 | New_Visitor | False | 1272 |
| 1 | New_Visitor | True | 422 |
| 2 | Other | False | 69 |
| 3 | Other | True | 16 |
| 4 | Returning_Visitor | False | 9081 |
| 5 | Returning_Visitor | True | 1470 |

Total pages visited & Individual Revenue contribution for each type of Month

[36]:
```
Monthly_page_views = df.groupby('Month')[['Administrative', 'Informational',␣
↪'ProductRelated', 'Revenue']].sum()
Monthly_page_views = Monthly_page_views.sort_values(by = ['Administrative',␣
↪'Informational', 'ProductRelated'], ascending = False).reset_index()
Monthly_page_views
```

[36]:

| | Month | Administrative | Informational | ProductRelated | Revenue |
|---|---|---|---|---|---|
| 0 | Nov | 7847 | 1938 | 138024 | 760 |
| 1 | May | 6610 | 1426 | 89105 | 365 |
| 2 | Dec | 3793 | 885 | 48347 | 216 |
| 3 | Mar | 3600 | 802 | 37775 | 192 |
| 4 | Oct | 2042 | 268 | 18428 | 115 |
| 5 | Sep | 1494 | 254 | 14831 | 86 |
| 6 | Aug | 1358 | 235 | 16566 | 76 |
| 7 | Jul | 1047 | 223 | 15728 | 66 |
| 8 | June | 655 | 162 | 10387 | 29 |
| 9 | Feb | 100 | 16 | 2058 | 3 |

Calculation of total sessions where a user visited all types of pages

[15]:
```
df['visited_all'] = (df['Administrative'] > 0) & (df['Informational'] > 0) &␣
↪(df['ProductRelated'] > 0)
df['visited_all'].sum()
```

[15]: 2167

Calculation of total sessions where a user visited all types of pages and also contributed to revenue

[16]:
```
df['visited_all&Purchased'] = (df['Administrative'] > 0) & (df['Informational']␣
↪> 0) & (df['ProductRelated'] > 0) & (df['Revenue'] > 0)
df['visited_all&Purchased'].sum()
```

[16]: 526

Finding correlation b/w Special day and revenue

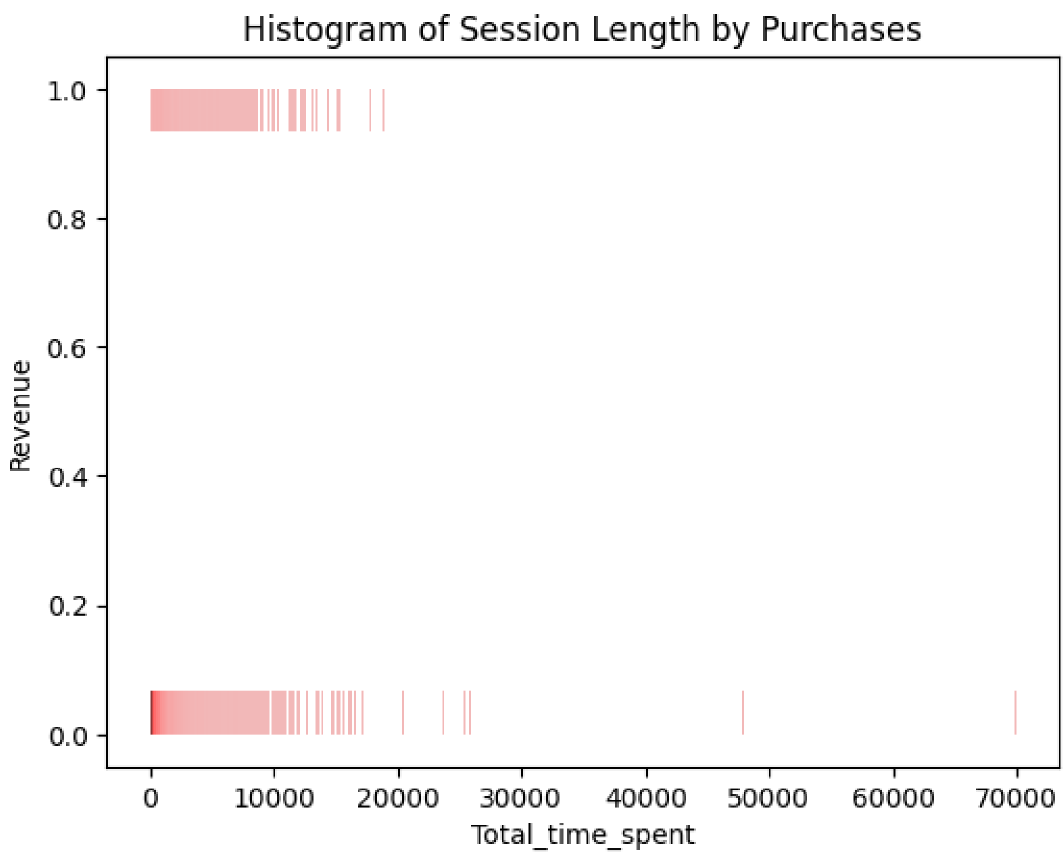
[17]:
```
variables_3 = df[['SpecialDay', 'Revenue']]
spearman_corr = variables_3.corr(method = 'spearman')
spearman_corr
```

[17]:
```
            SpecialDay   Revenue
SpecialDay    1.000000 -0.086878
Revenue      -0.086878  1.000000
```

Investigating user session lengths and their impact on conversion rates.

[18]:
```
df['Total_time_spent'] = df['Administrative_Duration'] +
 ↪df['Informational_Duration'] + df['ProductRelated_Duration']

variables_4 = df[['Total_time_spent', 'Revenue']]
spearman_corr = variables_4.corr(method = 'spearman')
print(spearman_corr)


sns.histplot(data = df, x = 'Total_time_spent', y = 'Revenue', kde = True,
 ↪color = 'red')
plt.title('Histogram of Session Length by Purchases')
plt.show()
```

```
                  Total_time_spent   Revenue
Total_time_spent          1.000000  0.220721
Revenue                   0.220721  1.000000
```

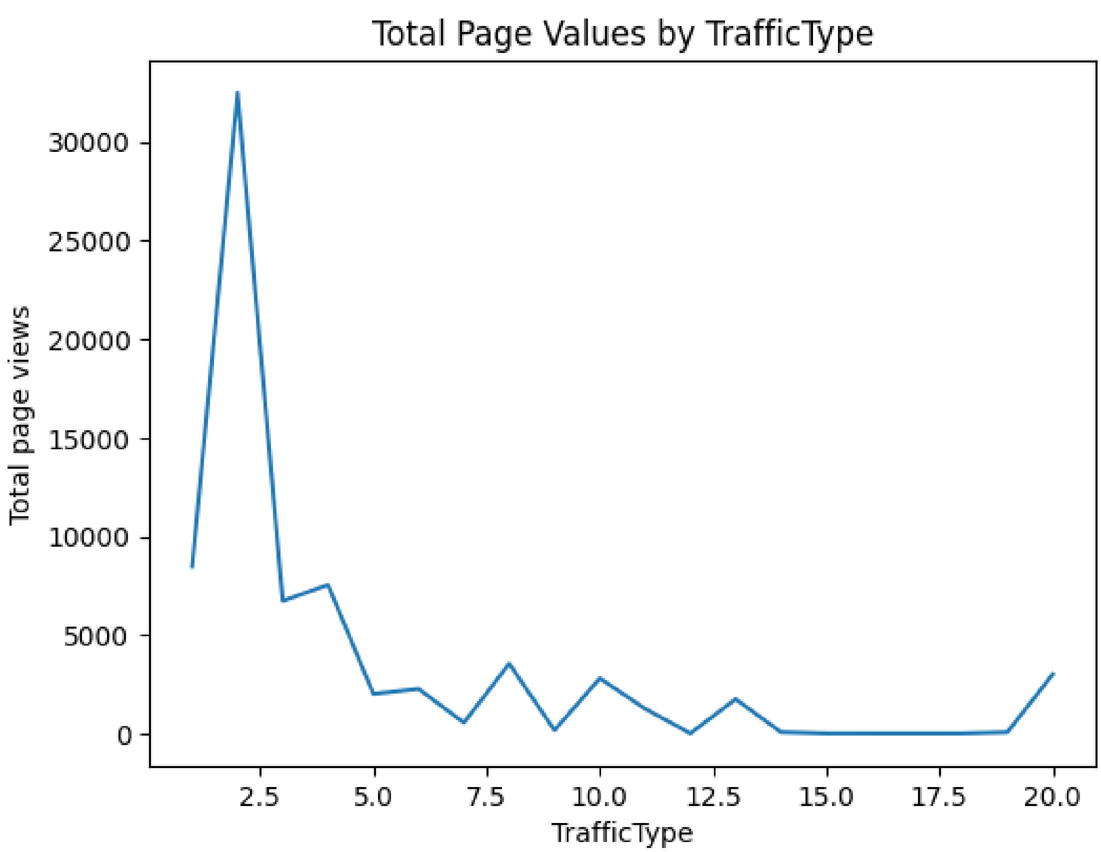



Exploring PageValues distribution and its relationship with TrafficType, VisitorType, and Region.

[19]:
```
variables_5 = df[['TrafficType', 'VisitorType', 'Region']]

for variable in variables_5:
  grouped_data = df.groupby(variable)['PageValues'].sum()
  print(grouped_data)
  print("<------------------------------------------->")

  sns.lineplot(x = grouped_data.index, y = grouped_data.values)
  plt.title(f'Total Page Values by {variable}')
  plt.xlabel(variable)
  plt.ylabel('Total page views')
  plt.show()
```

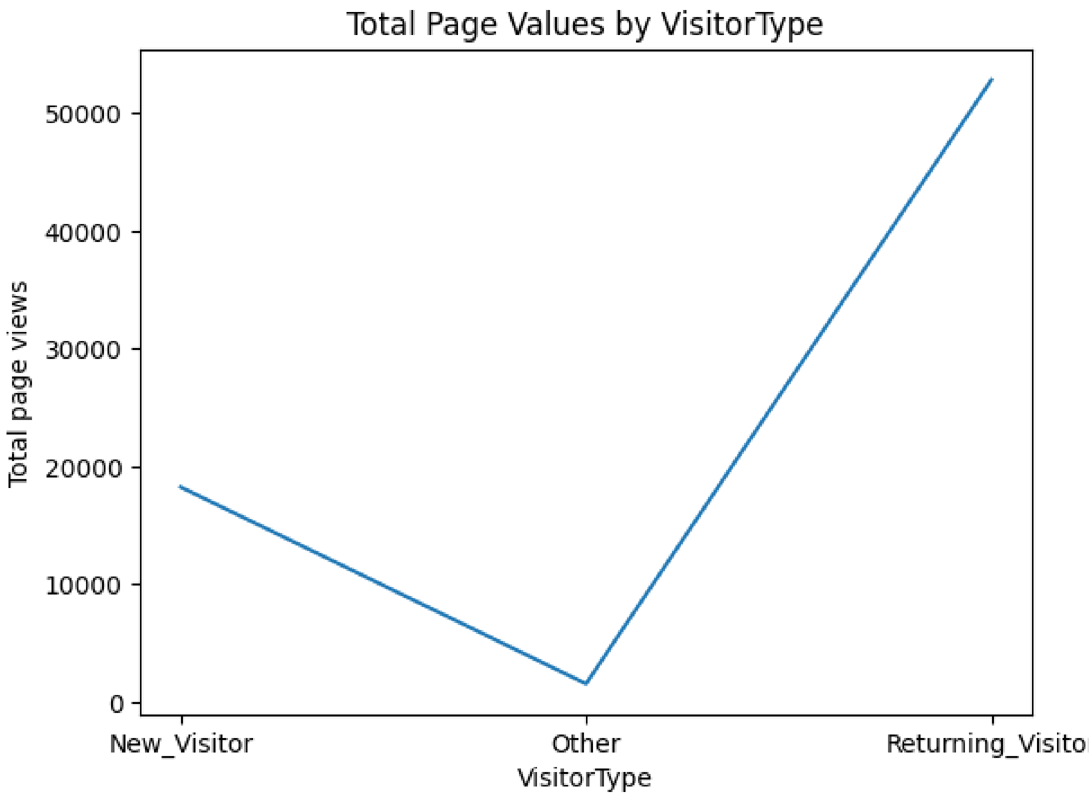
```
TrafficType
1       8468.386672
2      32494.983720
3       6722.420075
4       7529.087303
```

```
5       2005.247088
6       2253.852296
7        542.693810
8       3533.735395
9        160.379441
10      2793.703633
11      1251.954486
12         0.000000
13      1737.684088
14        64.169261
15         1.385792
16         0.000000
17         0.000000
18         0.000000
19        59.457846
20      2995.408541
Name: PageValues, dtype: float64
<------------------------------------------->
```

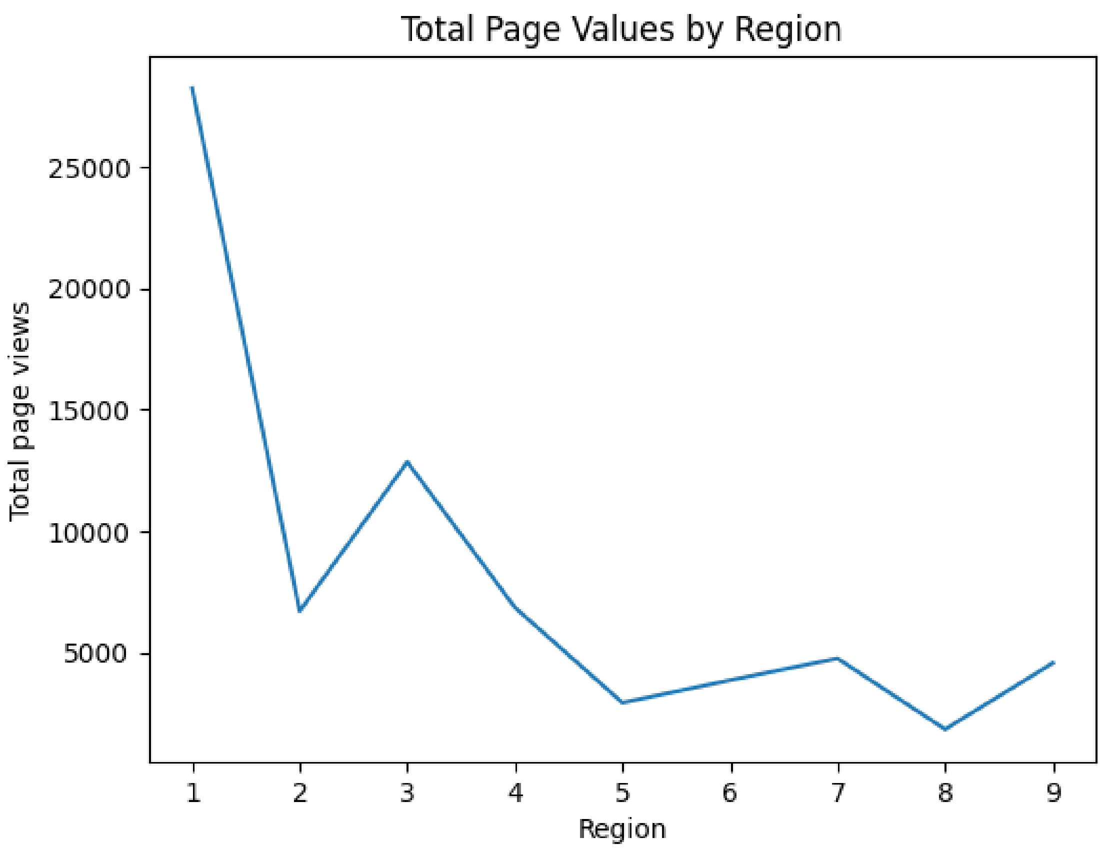



```
VisitorType
New_Visitor            18248.085596
```

```
Other                    1546.304039
Returning_Visitor       52820.159812
Name: PageValues, dtype: float64
<------------------------------------------>
```




```
Region
1    28212.256912
2     6709.893181
3    12842.429647
4     6860.170650
5     2941.753454
6     3866.874528
7     4763.541048
8     1847.940453
9     4569.689575
Name: PageValues, dtype: float64
<------------------------------------------>
```

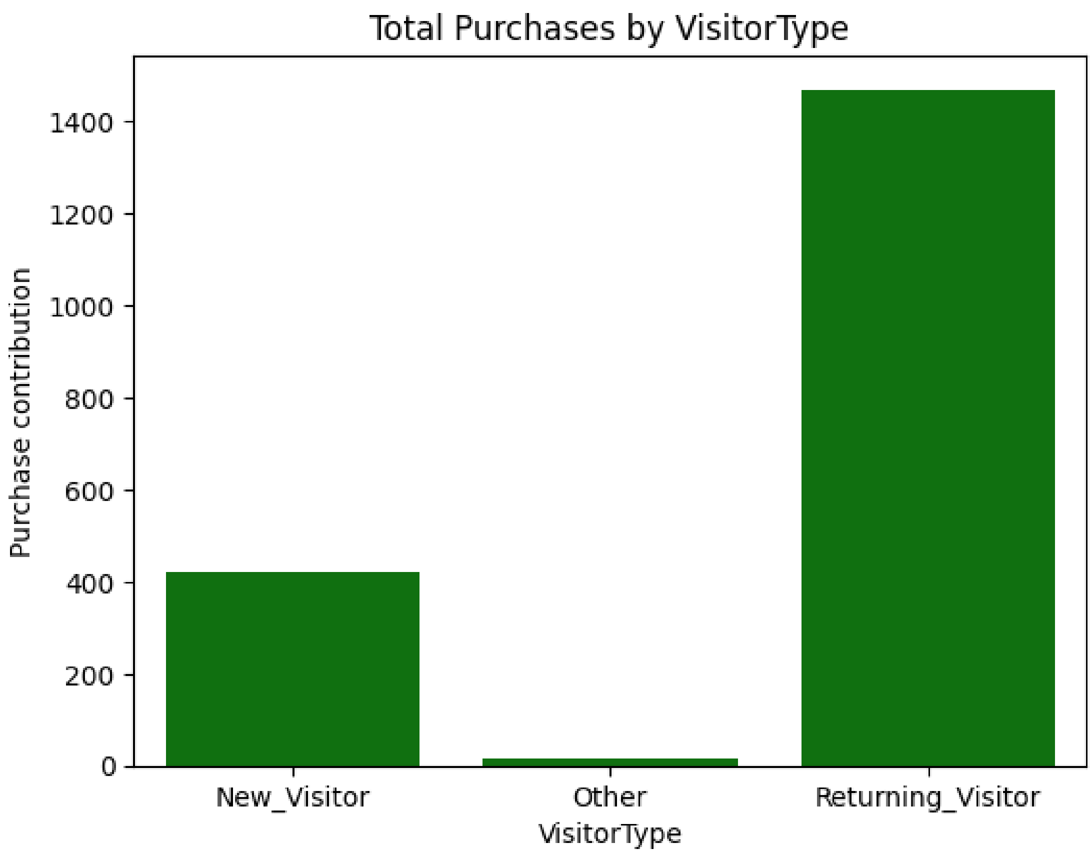



Grouping users based on VisitorType, OperatingSystems, and Region to identify potential differences in behavior and conversion rates

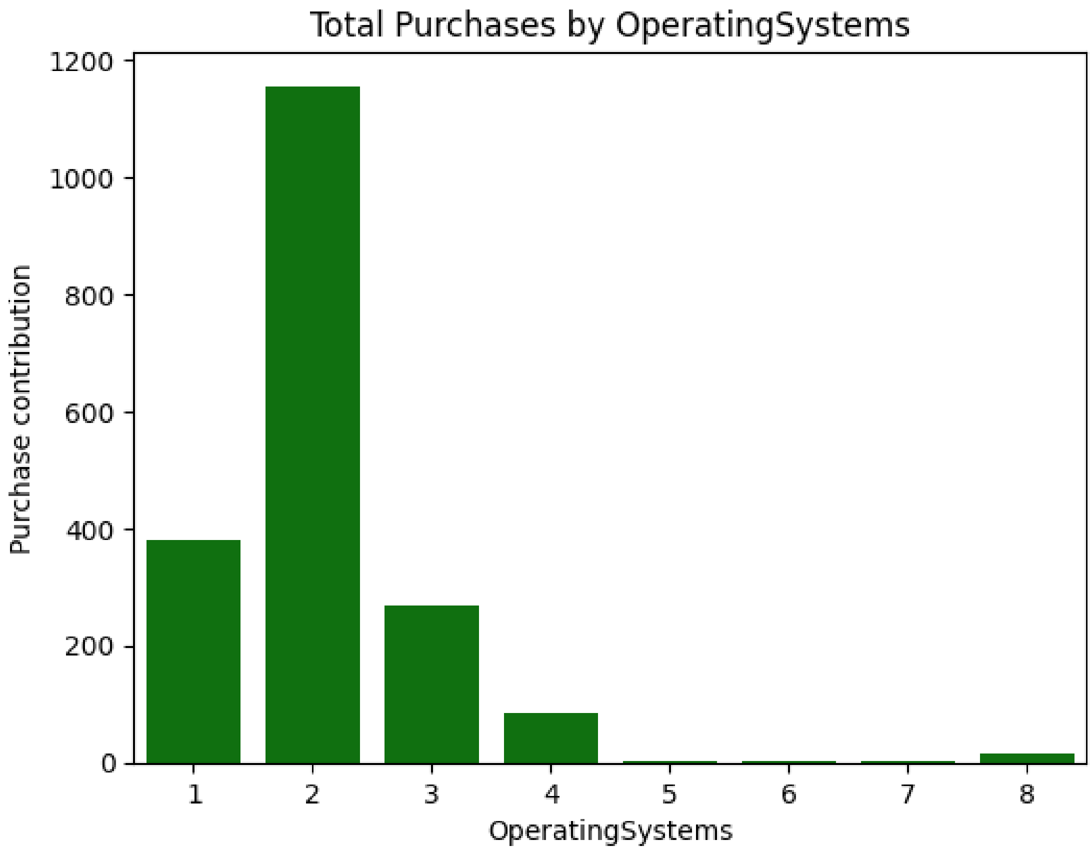
[73]:
```
variables_6 = df[['VisitorType', 'OperatingSystems', 'Region']]

for variable in variables_6:
  grouped_data = df.groupby(variable)['Revenue'].sum()
  print(grouped_data)
  print("<------------------------------------------->")

  sns.barplot(x = grouped_data.index, y = grouped_data.values, color = 'green')
  plt.title(f'Total Purchases by {variable}')
  plt.xlabel(variable)
  plt.ylabel('Purchase contribution')
  plt.show()
```

```
VisitorType
New_Visitor           422
Other                  16
Returning_Visitor    1470
Name: Revenue, dtype: int64
```

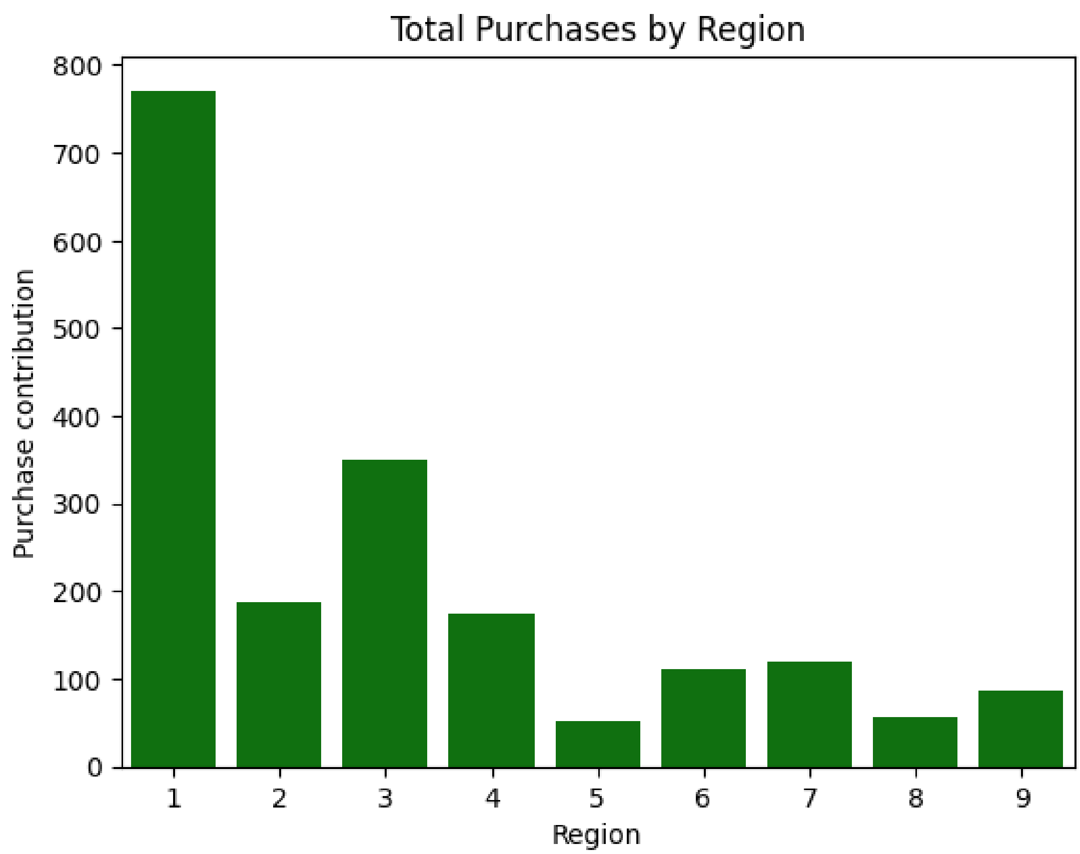
```
<------------------------------------------->
```




```
OperatingSystems
1     379
2    1155
3     268
4      85
5       1
6       2
7       1
8      17
Name: Revenue, dtype: int64
<------------------------------------------->
```




```
Region
1    771
2    188
3    349
4    175
5     52
6    112
7    119
8     56
9     86
Name: Revenue, dtype: int64
<------------------------------------------->
```




Hypothesis testing: Performing One- way and Two way Anova to check relationship between Independent and dependent variable.

[20]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

[21]:
```
df['Revenue'] = df['Revenue'].replace({True : 1, False: 0})
df['visited_all'] = df['visited_all'].replace({True : 1, False : 0})
df['visited_all&Purchased'] = df['visited_all&Purchased'].replace({True: 1,␣
↪False : 0})
df['Weekend'] = df['Weekend'].replace({True : 1, False : 0})
```

[23]:
```
# H0: There is no relationship betweeen dependent and independent variables.
# H1: There is a signifiant relationship between dependent and independent␣
↪variables.

test = ols('Revenue ~ C(OperatingSystems) * C(Browser)', data =df).fit()
anova_table = sm.stats.anova_lm(test,typ= 1)
print(anova_table)
```

```
                                    df       sum_sq   mean_sq          F  \
C(OperatingSystems)                7.0     9.813438  1.401920  10.801121
C(Browser)                        12.0     1.867781  0.155648   1.199197
C(OperatingSystems):C(Browser)    84.0     8.962134  0.106692   0.822011
Residual                       12288.0  1594.907520  0.129794        NaN

                                     PR(>F)
C(OperatingSystems)            1.194180e-13
C(Browser)                     2.766328e-01
C(OperatingSystems):C(Browser)  8.799230e-01
Residual                                NaN
```

[24]:
```
test = ols('Revenue ~ C(visited_all)', data =df).fit()
anova_table = sm.stats.anova_lm(test,typ= 1)
print(anova_table)
```

```
                     df       sum_sq    mean_sq           F        PR(>F)
C(visited_all)      1.0    20.353573  20.353573  157.573355  6.359801e-36
Residual        12328.0  1592.393872   0.129169         NaN           NaN
```

[25]:
```
test = ols('Revenue ~ C(SpecialDay)', data =df).fit()
anova_table = sm.stats.anova_lm(test,typ= 1)
print(anova_table)
```

```
                    df       sum_sq   mean_sq          F        PR(>F)
C(SpecialDay)      5.0    12.566730  2.513346  19.356862  3.003311e-19
Residual       12324.0  1600.180715  0.129843        NaN           NaN
```

[1]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

[ ]:
```
!pip install nbconvert

!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-
packages (6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
```

# Campaign dataset@DhanunjayaReddy

August 30, 2024

## 1 Campaign Dataset

[125]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Downloading and reading the shopping csv file

[126]:
```python
df = pd.read_csv("campaign.csv")
df
```

[126]:
```
         ID  Year_Birth   Education Marital_Status       Income  Kidhome  \
0      1826        1970  Graduation       Divorced  $84,835.00        0
1         1        1961  Graduation         Single  $57,091.00        0
2     10476        1958  Graduation        Married  $67,267.00        0
3      1386        1967  Graduation       Together  $32,474.00        1
4      5371        1989  Graduation         Single  $21,474.00        1
…       …          …           …            …           …          …
2234  10142        1976         PhD       Divorced  $66,476.00        0
2235   5263        1977    2n Cycle        Married  $31,056.00        1
2236     22        1976  Graduation       Divorced  $46,310.00        1
2237    528        1978  Graduation        Married  $65,819.00        0
2238   4070        1969         PhD        Married  $94,871.00        0

      Teenhome Dt_Customer  Recency  MntWines  …  NumCatalogPurchases  \
0            0     6/16/14        0       189  …                    4
1            0     6/15/14        0       464  …                    3
2            1     5/13/14        0       134  …                    2
3            1     5/11/14        0        10  …                    0
4            0      4/8/14        0         6  …                    1
…          …          …        …        …  …                  …
2234         1      3/7/13       99       372  …                    2
2235         0     1/22/13       99         5  …                    0
2236         0     12/3/12       99       185  …                    1
2237         0    11/29/12       99       267  …                    4
2238         2      9/1/12       99       169  …                    5
```

```
      NumStorePurchases  NumWebVisitsMonth  AcceptedCmp3  AcceptedCmp4  \
0                     6                  1             0             0
1                     7                  5             0             0
2                     5                  2             0             0
3                     2                  7             0             0
4                     2                  7             1             0
…                   …                  …             …             …
2234                 11                  4             0             0
2235                  3                  8             0             0
2236                  5                  8             0             0
2237                 10                  3             0             0
2238                  4                  7             0             1

      AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  Country
0                0             0             0         0       SP
1                0             0             1         0       CA
2                0             0             0         0       US
3                0             0             0         0      AUS
4                0             0             0         0       SP
…              …             …             …         …      …
2234             0             0             0         0       US
2235             0             0             0         0       SP
2236             0             0             0         0       SP
2237             0             0             0         0      IND
2238             1             0             0         0       CA

[2239 rows x 27 columns]
```

[88]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2239 entries, 0 to 2238
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2239 non-null   int64
 1   Year_Birth           2239 non-null   int64
 2   Education            2239 non-null   object
 3   Marital_Status       2239 non-null   object
 4   Income               2239 non-null   object
 5   Kidhome              2239 non-null   int64
 6   Teenhome             2239 non-null   int64
 7   Dt_Customer          2239 non-null   object
 8   Recency              2239 non-null   int64
 9   MntWines             2239 non-null   int64
 10  MntFruits            2239 non-null   int64
 11  MntMeatProducts      2239 non-null   int64
```

```
 12  MntFishProducts      2239 non-null   int64
 13  MntSweetProducts     2239 non-null   int64
 14  MntGoldProds         2239 non-null   int64
 15  NumDealsPurchases    2239 non-null   int64
 16  NumWebPurchases      2239 non-null   int64
 17  NumCatalogPurchases  2239 non-null   int64
 18  NumStorePurchases    2239 non-null   int64
 19  NumWebVisitsMonth    2239 non-null   int64
 20  AcceptedCmp3         2239 non-null   int64
 21  AcceptedCmp4         2239 non-null   int64
 22  AcceptedCmp5         2239 non-null   int64
 23  AcceptedCmp1         2239 non-null   int64
 24  AcceptedCmp2         2239 non-null   int64
 25  Complain             2239 non-null   int64
 26  Country              2239 non-null   object
dtypes: int64(22), object(5)
memory usage: 472.4+ KB
```

Unique number of values for specific categorical columns

[128]:
```
columns_list = df[['ID', 'Education', 'Marital_Status', 'Country']]

for columns in columns_list.columns:
  unique_count = columns_list[columns].nunique()
  print(columns, "-", unique_count)
```

```
ID - 2239
Education - 5
Marital_Status - 8
Country - 8
```

Checking for the presence of null values in dataset.

[129]:
```
df.isna().sum()
```

[129]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
```

```
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Country                0
dtype: int64
```

shape of the dataset

[130]:
```
df.shape
```

[130]:
```
(2239, 27)
```

summary statistics of the dataset

[131]:
```
df['Income'] = df['Income'].replace({'\$': '', ',': ''}, regex=True).
 ↪astype(float)
df['Income'] = df['Income'].fillna(0).astype(int)
```

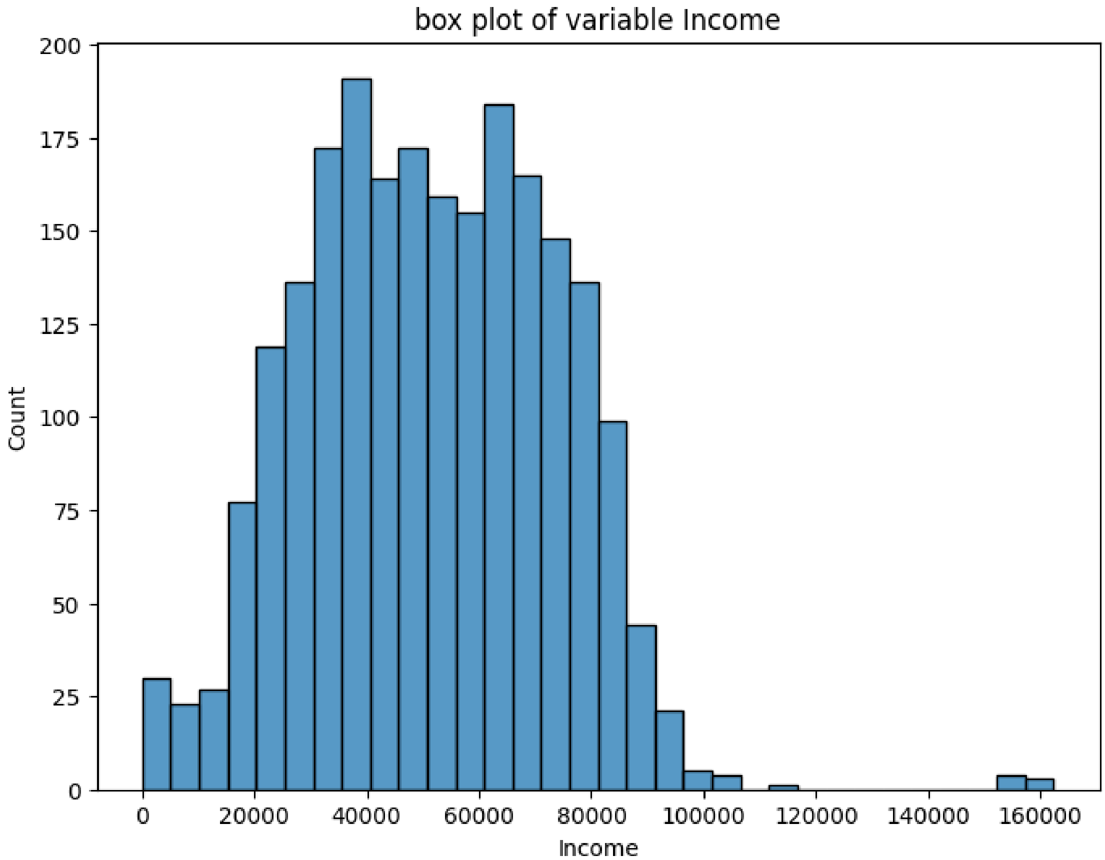
[132]:
```
selected_variables = df[['Income', 'Kidhome',
       'Teenhome', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']]

summary_df = selected_variables.describe()
summary_df
```

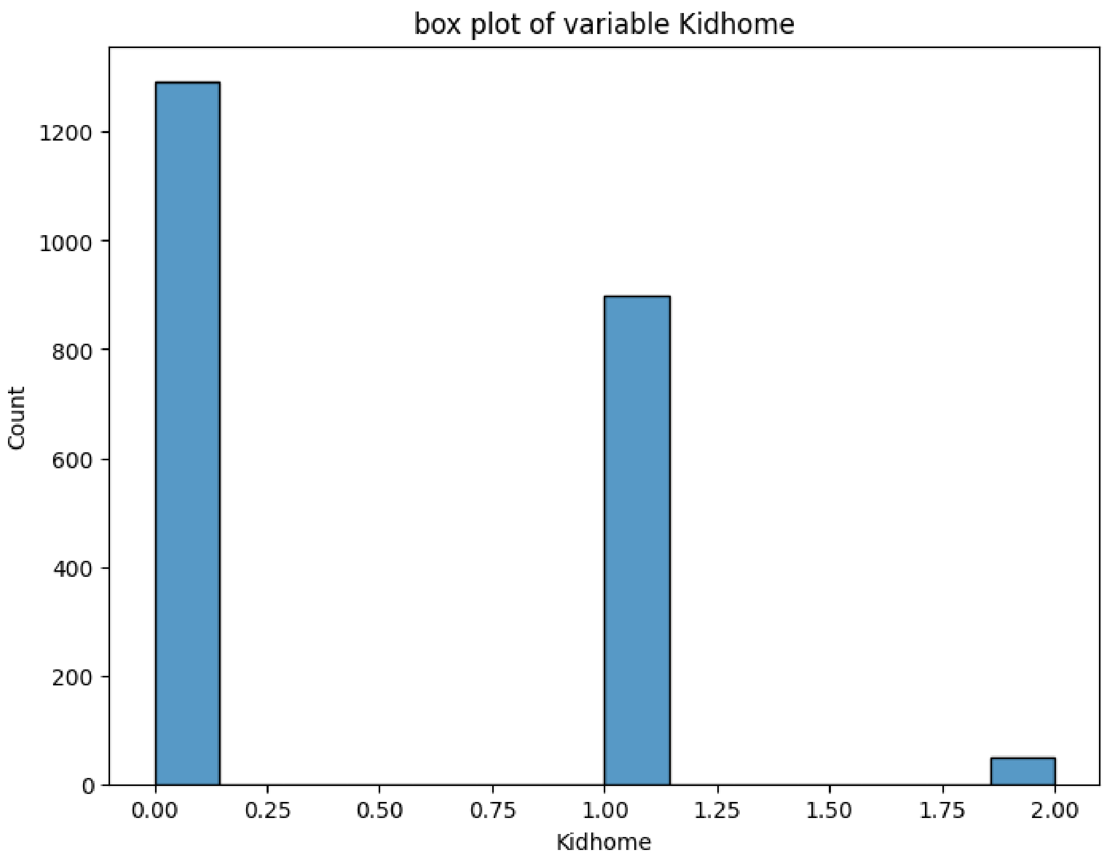
[132]:
```
              Income      Kidhome     Teenhome      Recency     MntWines  \
count    2239.000000  2239.000000  2239.000000  2239.000000  2239.000000
mean    51412.792765     0.443948     0.506476    49.121036   304.067441
std     22069.582225     0.538390     0.544555    28.963662   336.614830
min         0.000000     0.000000     0.000000     0.000000     0.000000
25%     34716.000000     0.000000     0.000000    24.000000    24.000000
50%     51039.000000     0.000000     0.000000    49.000000   174.000000
75%     68277.500000     1.000000     1.000000    74.000000   504.500000
max    162397.000000     2.000000     2.000000    99.000000  1493.000000

         MntFruits  MntMeatProducts  MntFishProducts  MntSweetProducts  \
count  2239.000000      2239.000000      2239.000000       2239.000000
```

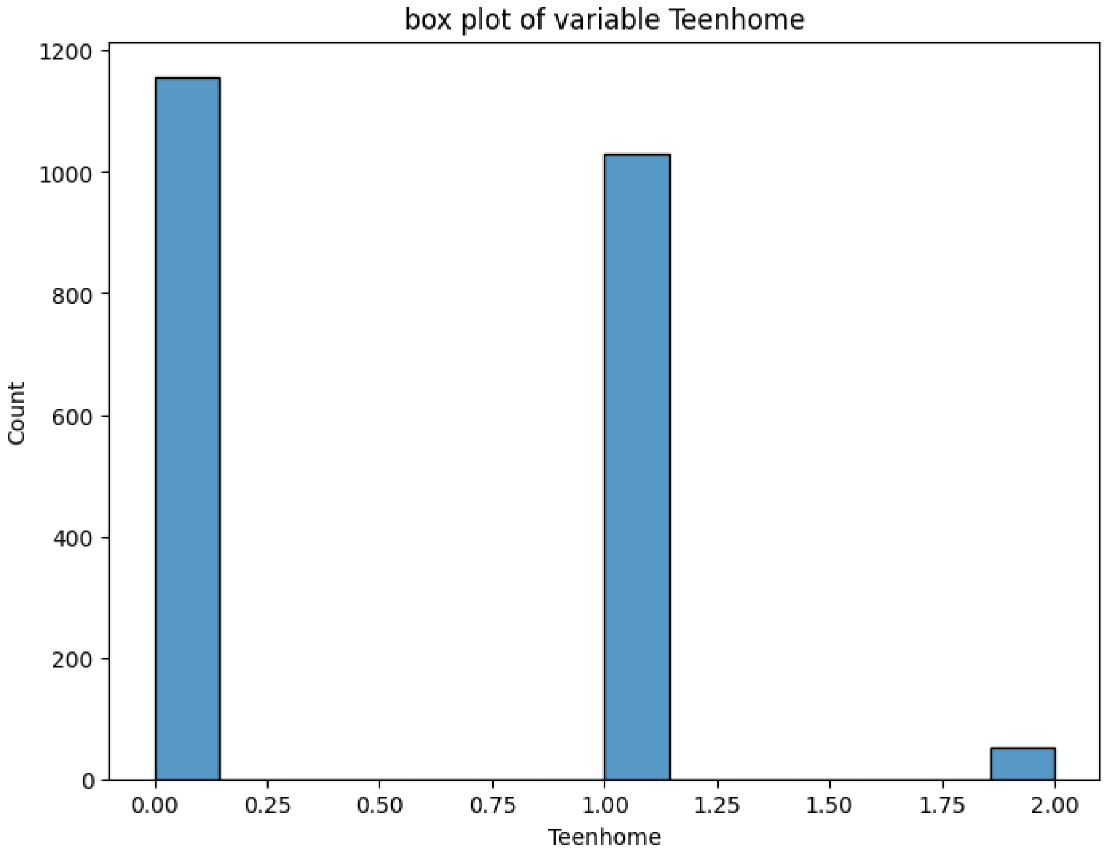
```
mean     26.307727     167.016525     37.538633     27.074587
std      39.781468     225.743829     54.637617     41.286043
min       0.000000       0.000000      0.000000      0.000000
25%       1.000000      16.000000      3.000000      1.000000
50%       8.000000      67.000000     12.000000      8.000000
75%      33.000000     232.000000     50.000000     33.000000
max     199.000000    1725.000000    259.000000    263.000000

       MntGoldProds  NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
count   2239.000000        2239.000000      2239.000000          2239.000000
mean      44.036177           2.324252         4.085306             2.662796
std       52.174700           1.932345         2.779240             2.923542
min        0.000000           0.000000         0.000000             0.000000
25%        9.000000           1.000000         2.000000             0.000000
50%       24.000000           2.000000         4.000000             2.000000
75%       56.000000           3.000000         6.000000             4.000000
max      362.000000          15.000000        27.000000            28.000000

       NumStorePurchases  NumWebVisitsMonth
count        2239.000000        2239.000000
mean            5.791425           5.316213
std             3.251149           2.427144
min             0.000000           0.000000
25%             3.000000           3.000000
50%             5.000000           6.000000
75%             8.000000           7.000000
max            13.000000          20.000000
```

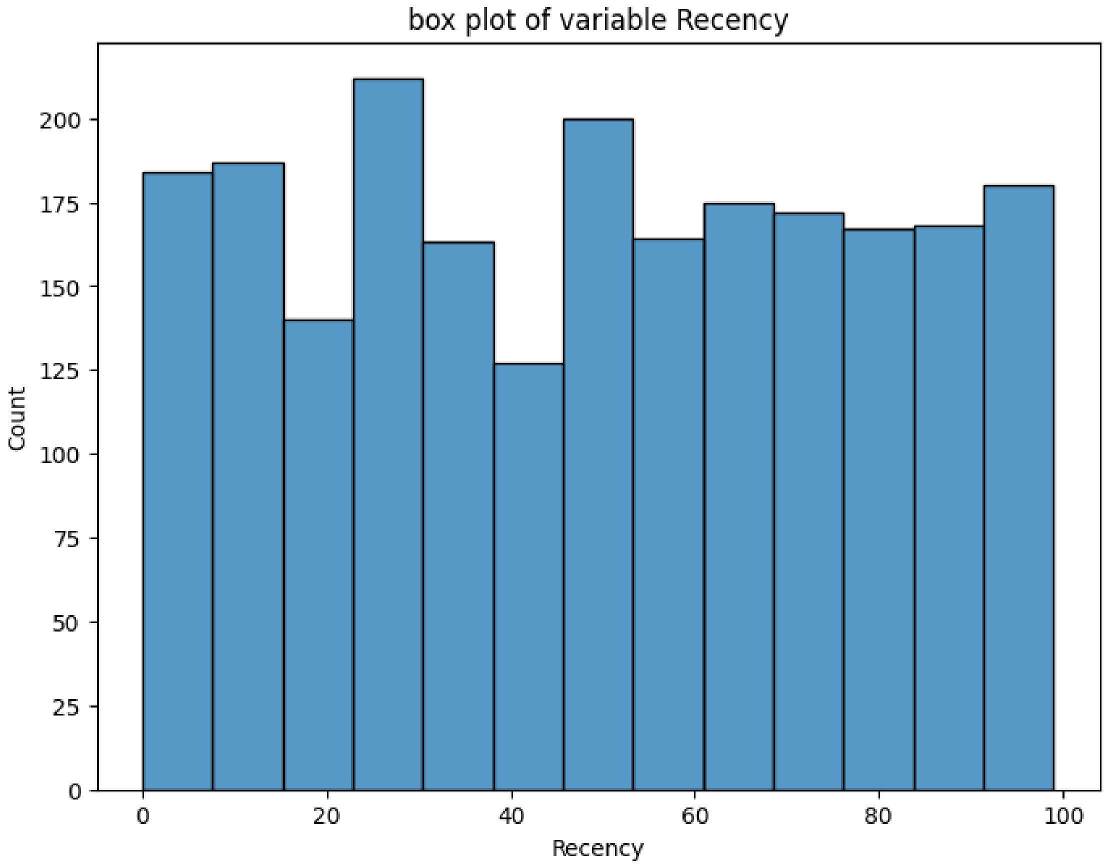
Distribution of the numerical features in the dataset

[133]:
```
for variable in selected_variables:
  plt.figure(figsize = (8, 6))
  sns.histplot(data = selected_variables[variable])
  plt.title(f"box plot of variable {variable}")
```

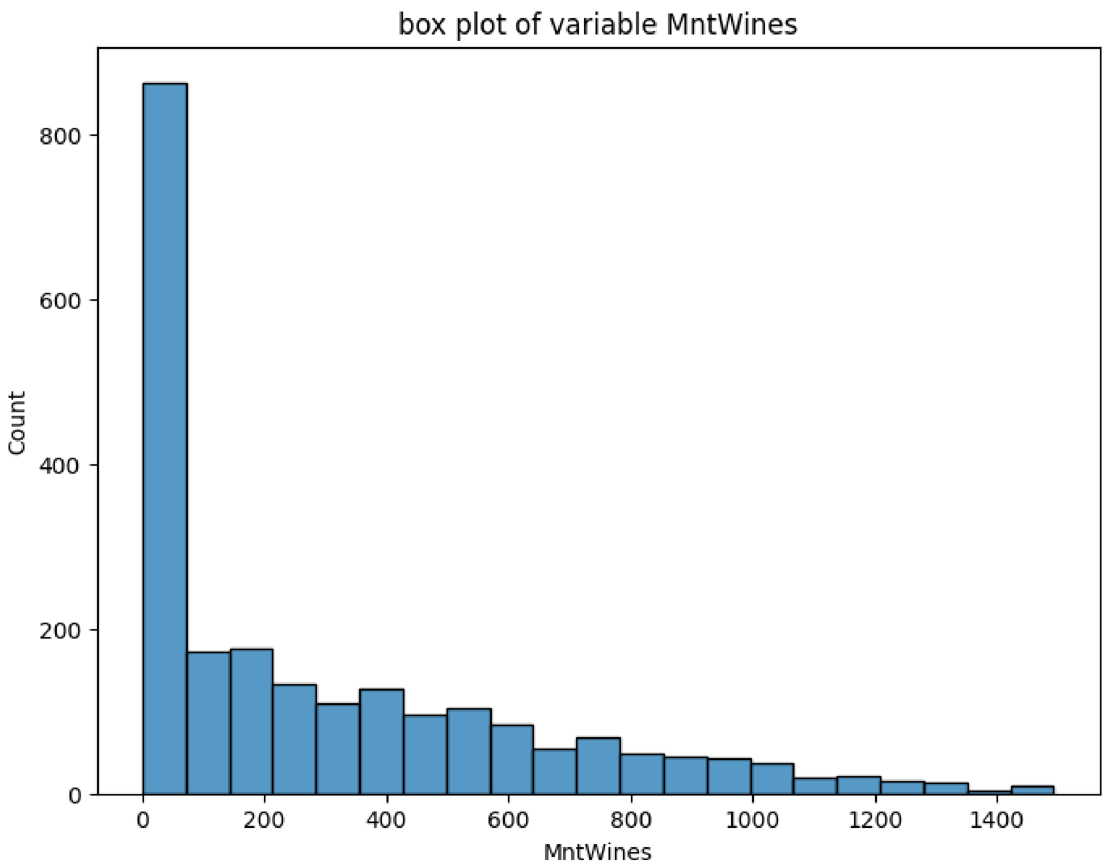

box plot of variable Income
200
175
150
125
100
75
50
25
0
Count
0
20000
40000
60000
80000
100000
120000
140000
160000
Income


box plot of variable Kidhome
Count
1200
1000
800
600
400
200
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Kidhome

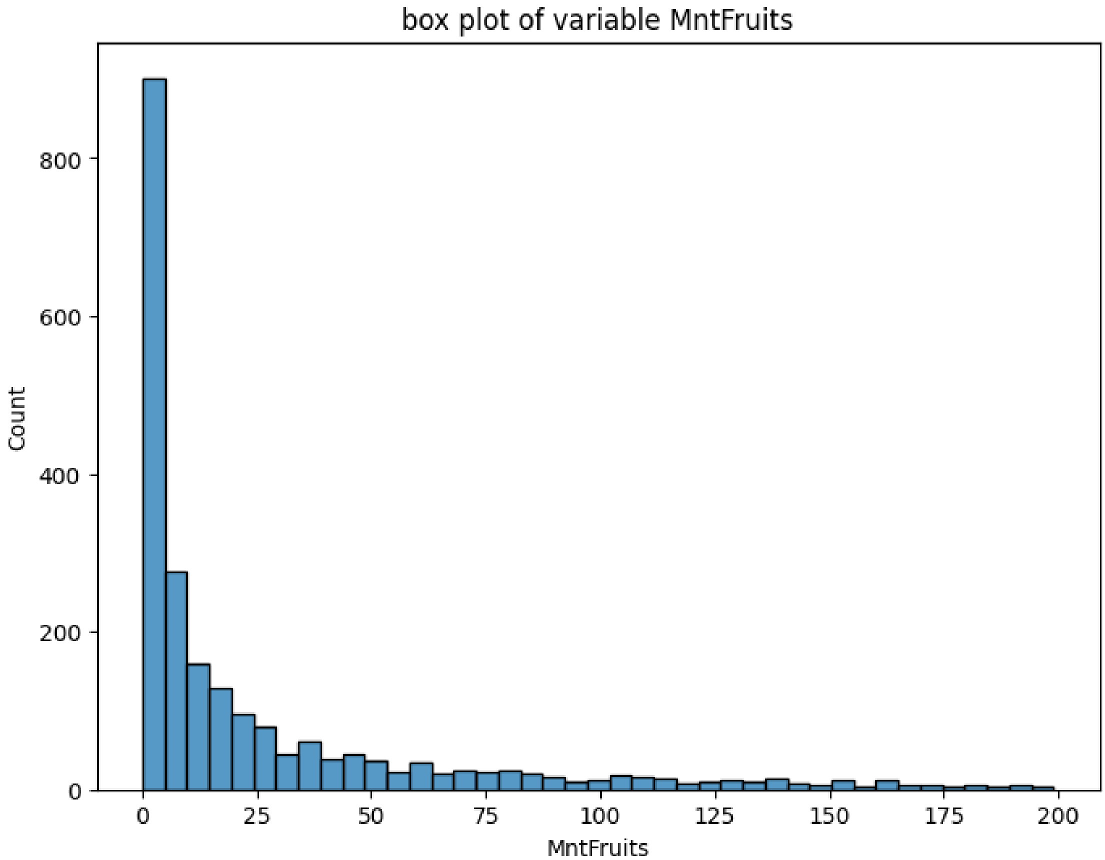

box plot of variable Teenhome
1200
1000
800
600
400
200
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Teenhome

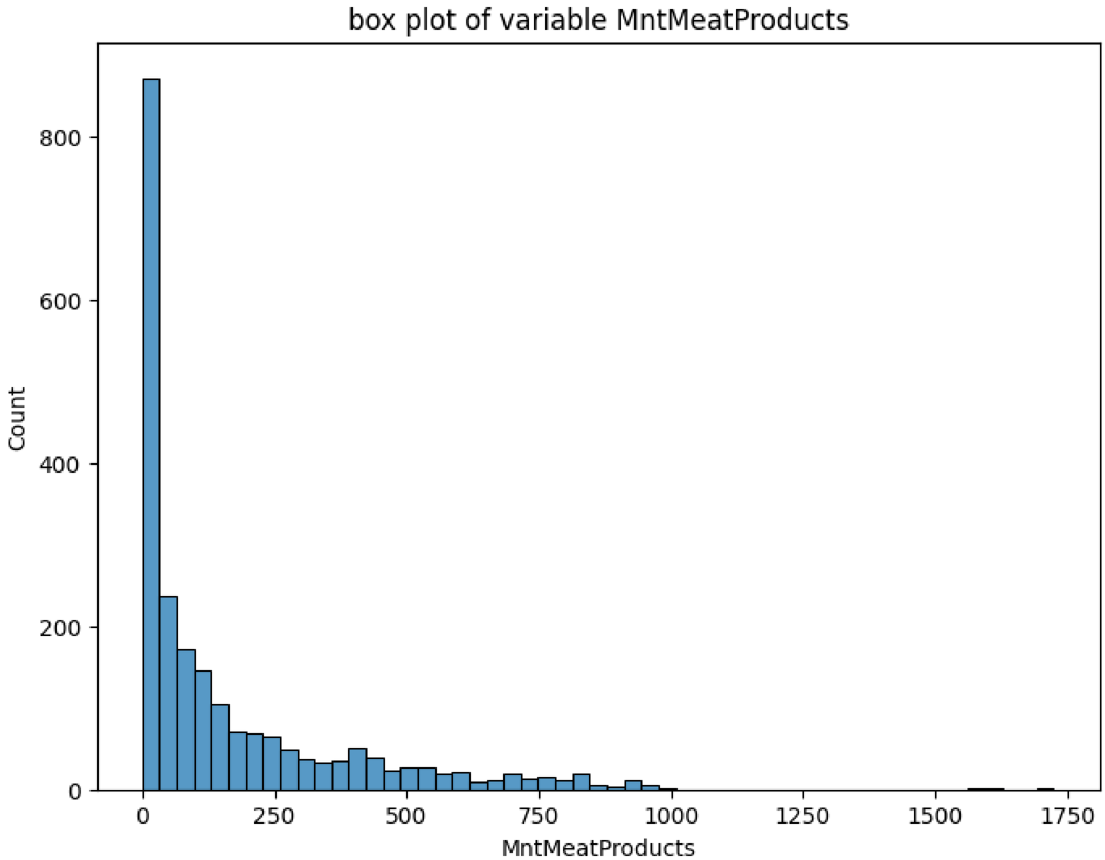


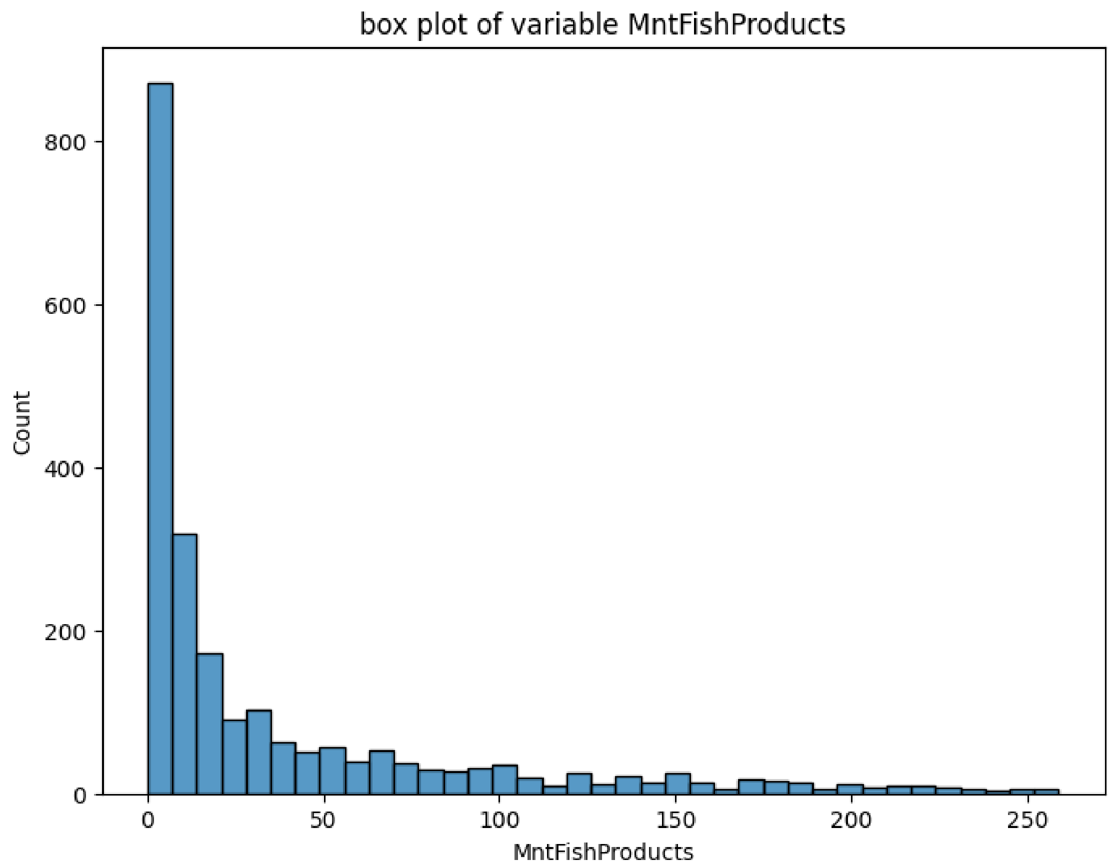
box plot of variable Recency
Count
Recency
0
20
40
60
80
100
25
50
75
125
150
175
200

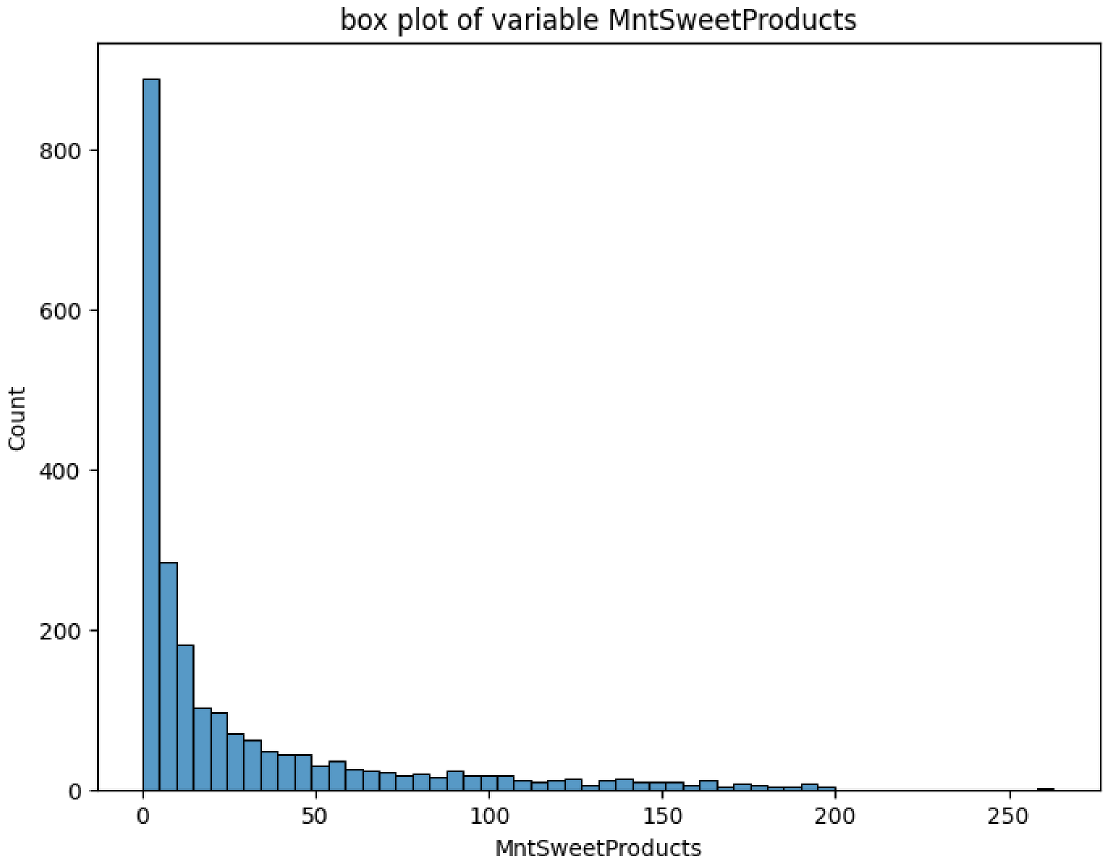


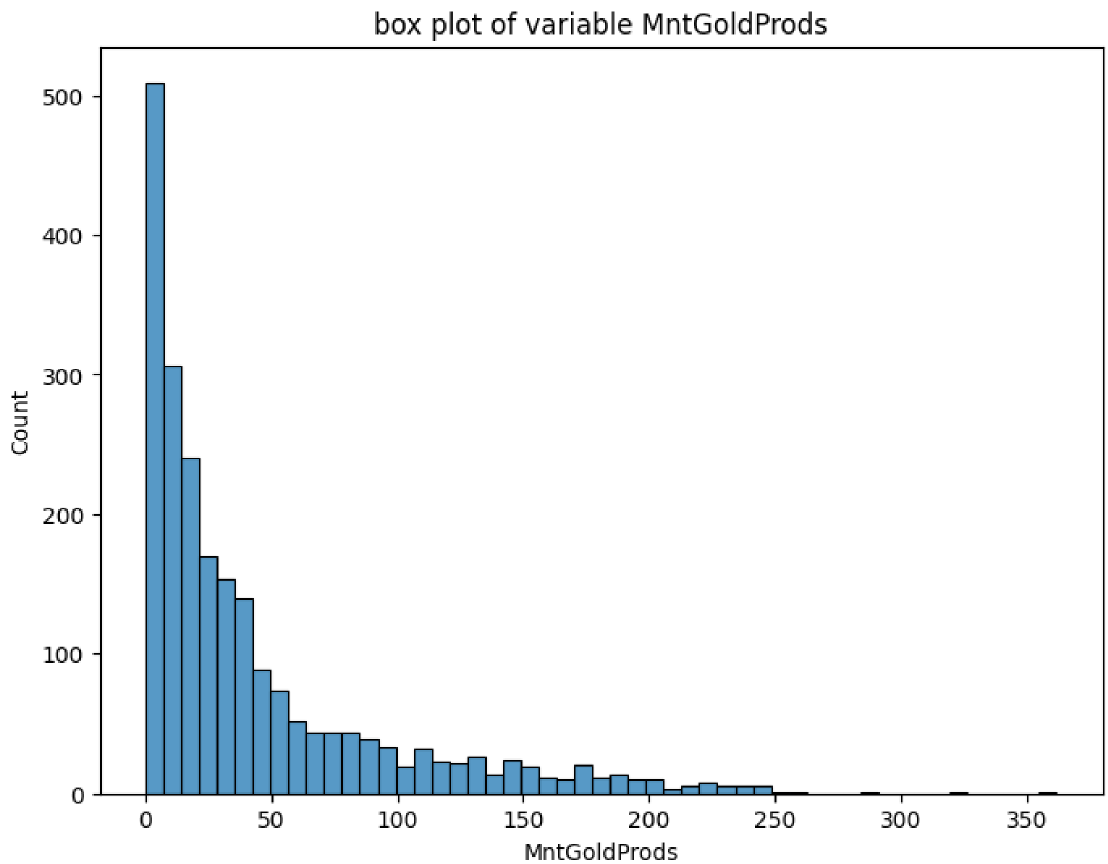
box plot of variable MntWines
Count
MntWines
0
200
400
600
800
1000
1200
1400

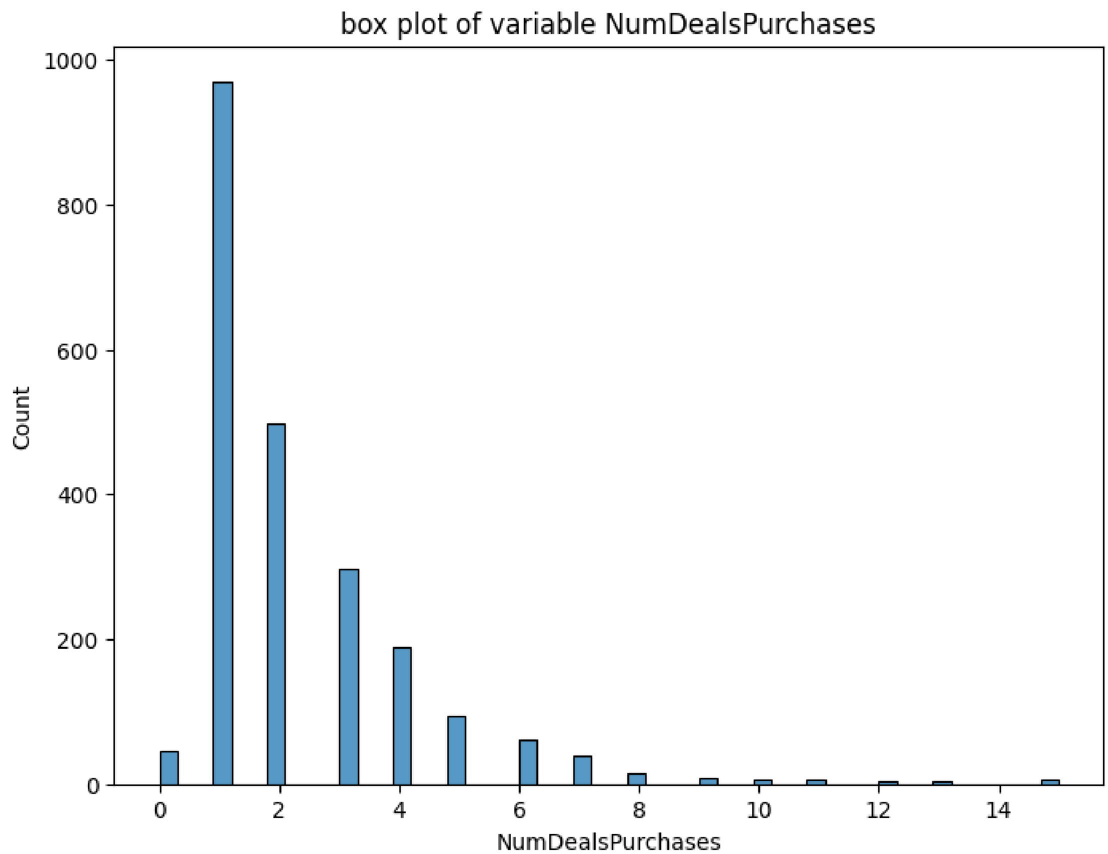

box plot of variable MntFruits
Count
MntFruits
0
200
400
600
800
0
25
50
75
100
125
150
175
200

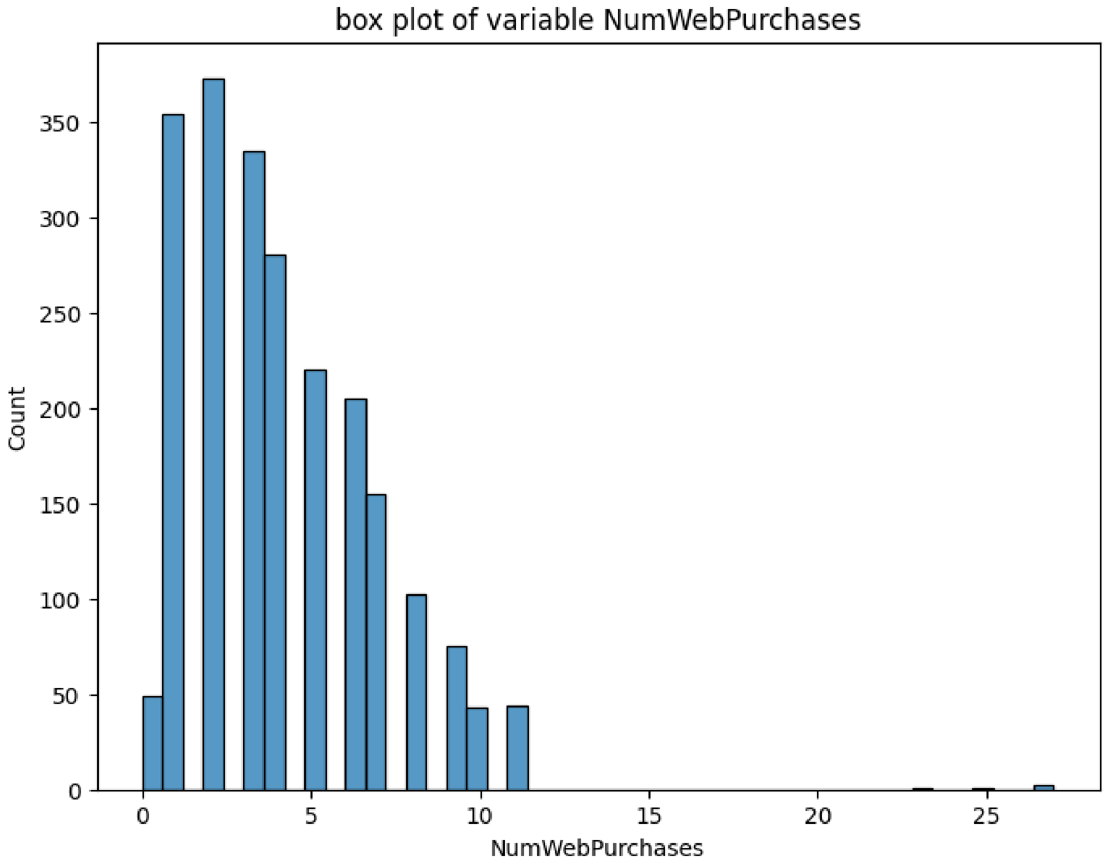

box plot of variable MntMeatProducts
Count
800
600
400
200
0
0
250
500
750
1000
1250
1500
1750
MntMeatProducts

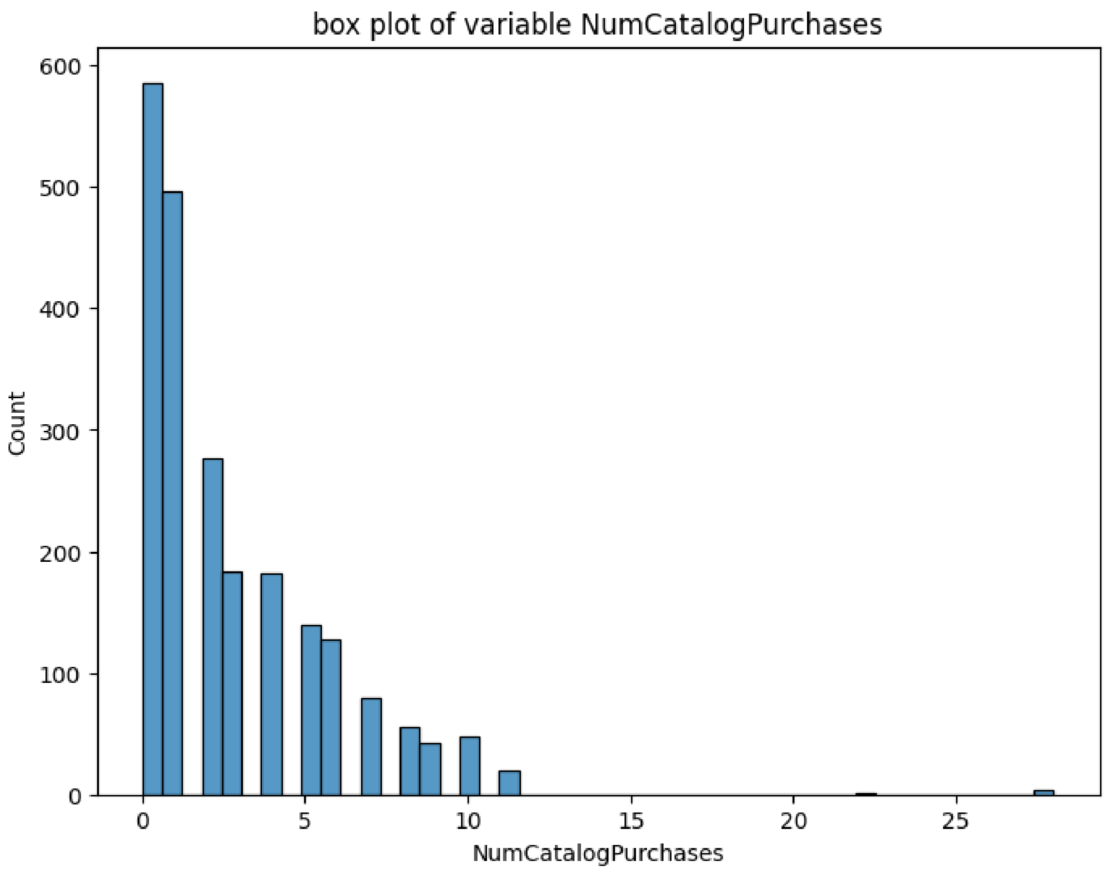

box plot of variable MntFishProducts
Count
800
600
400
200
0
0
50
100
150
200
250
MntFishProducts

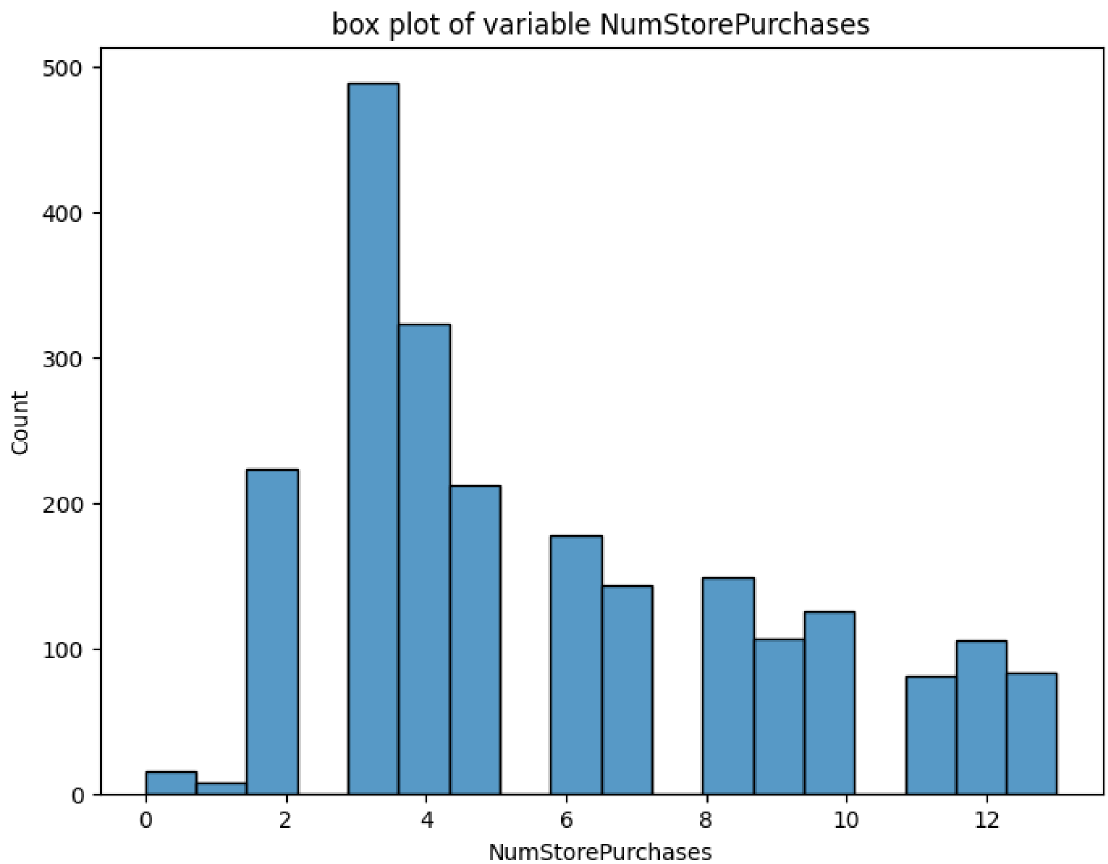

box plot of variable MntSweetProducts
Count
0
200
400
600
800
0
50
100
150
200
250
MntSweetProducts


box plot of variable MntGoldProds
Count
MntGoldProds
0
100
200
300
400
500
0
50
100
150
200
250
300
350

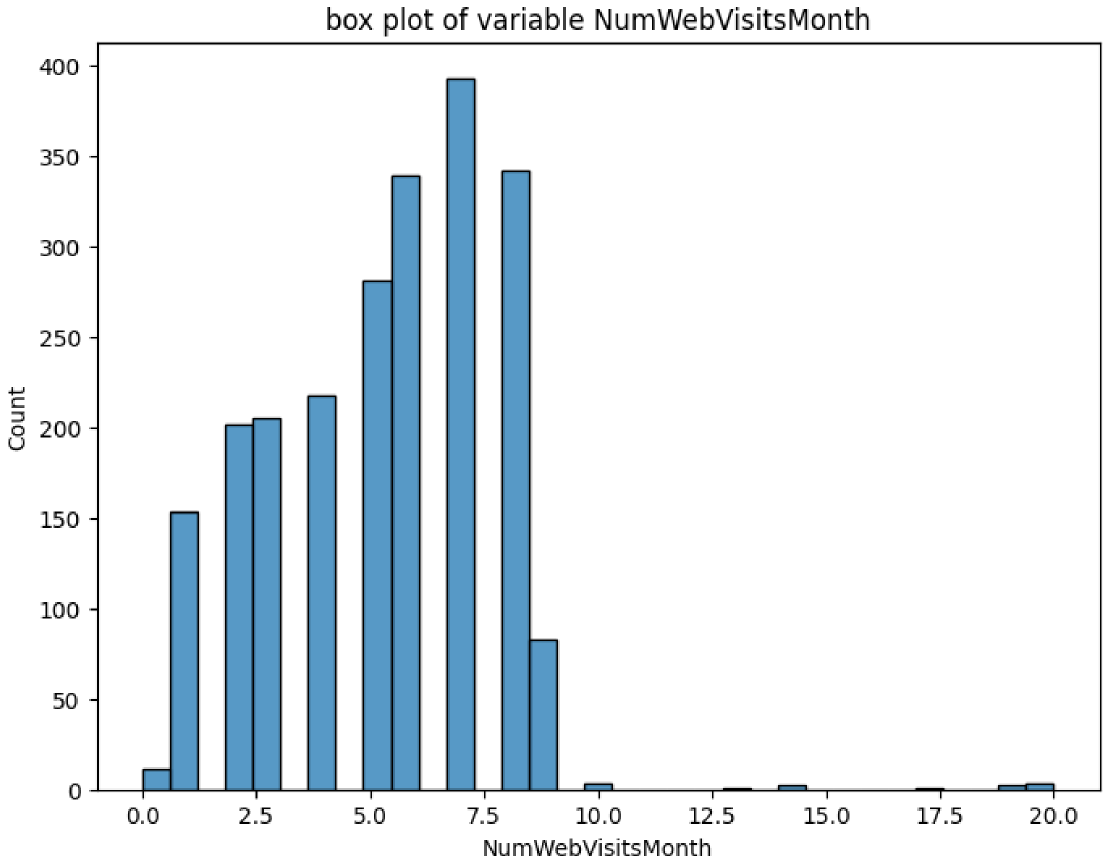

box plot of variable NumDealsPurchases
Count
NumDealsPurchases
0
200
400
600
800
1000
0
2
4
6
8
10
12
14


box plot of variable NumWebPurchases
Count
350
300
250
200
150
100
50
0
0
5
10
15
20
25
NumWebPurchases


box plot of variable NumCatalogPurchases
Count
600
500
400
300
200
100
0
0
5
10
15
20
25
NumCatalogPurchases

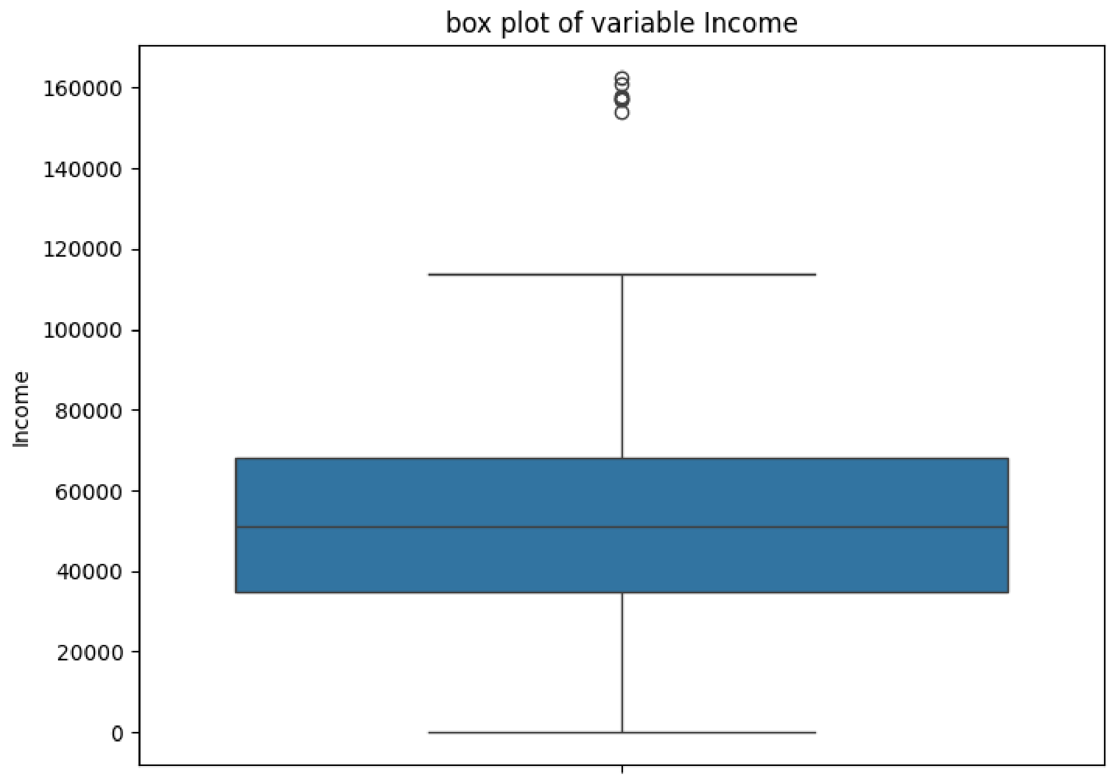


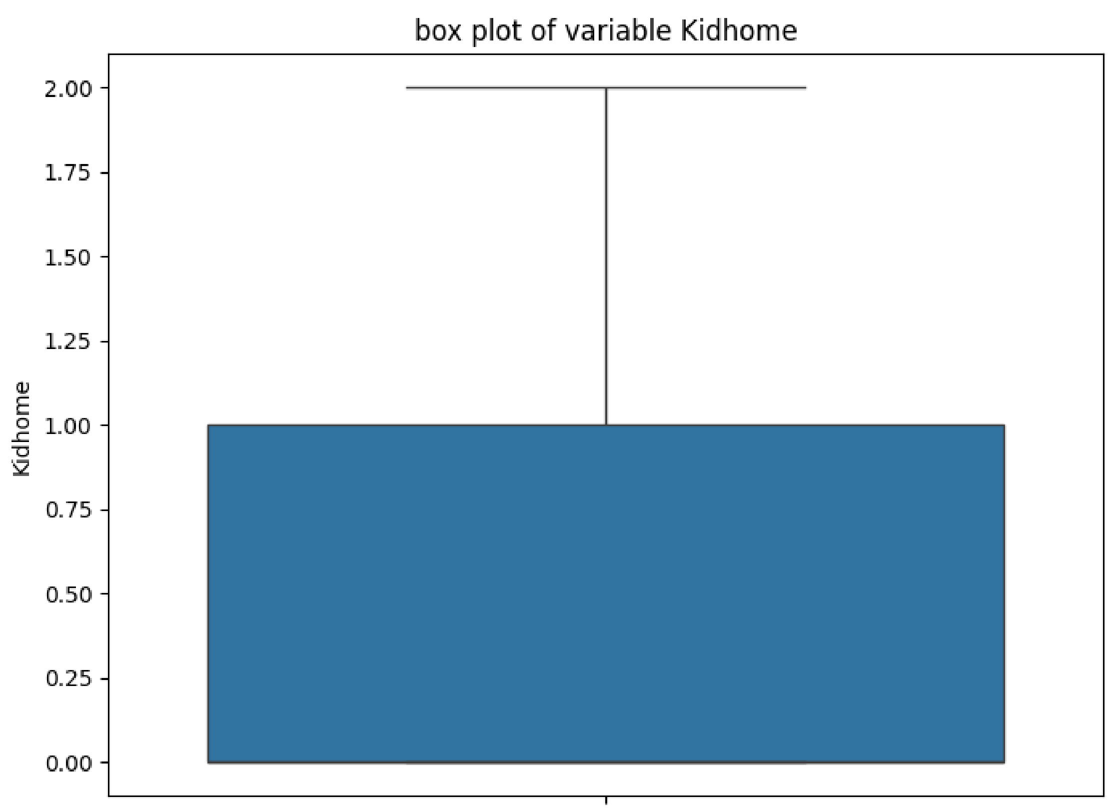
box plot of variable NumStorePurchases
Count
NumStorePurchases
0
100
200
300
400
500
0
2
4
6
8
10
12

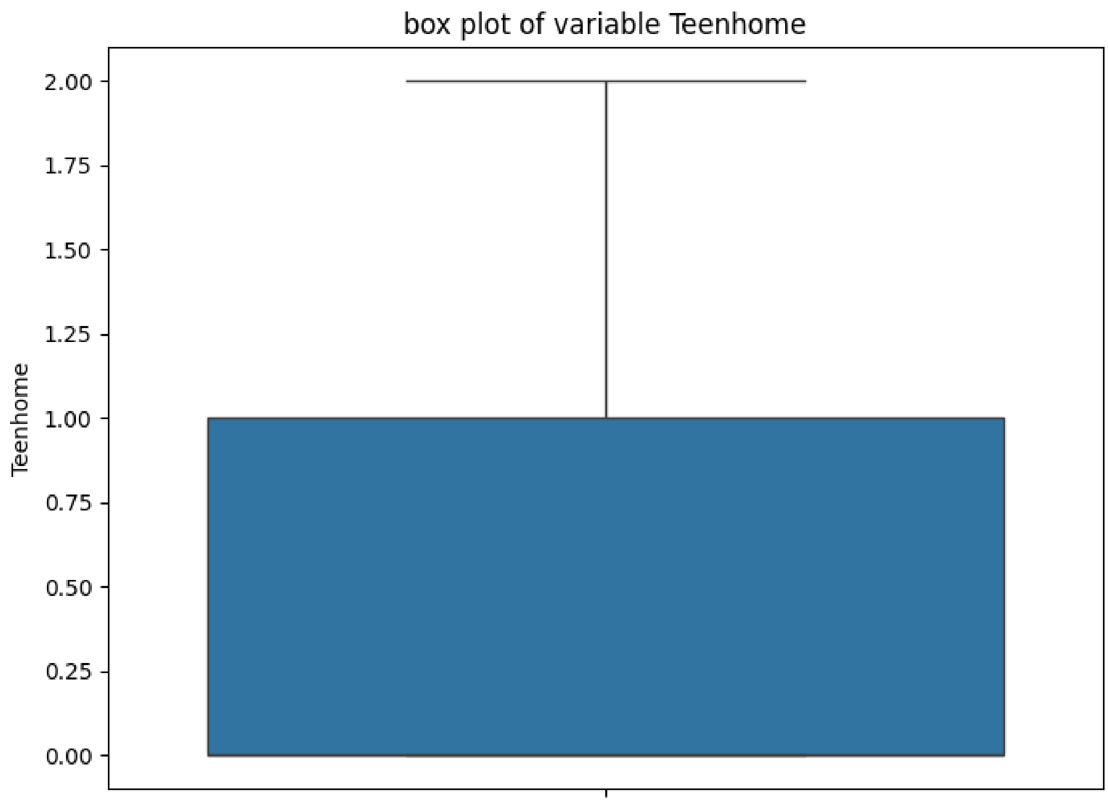


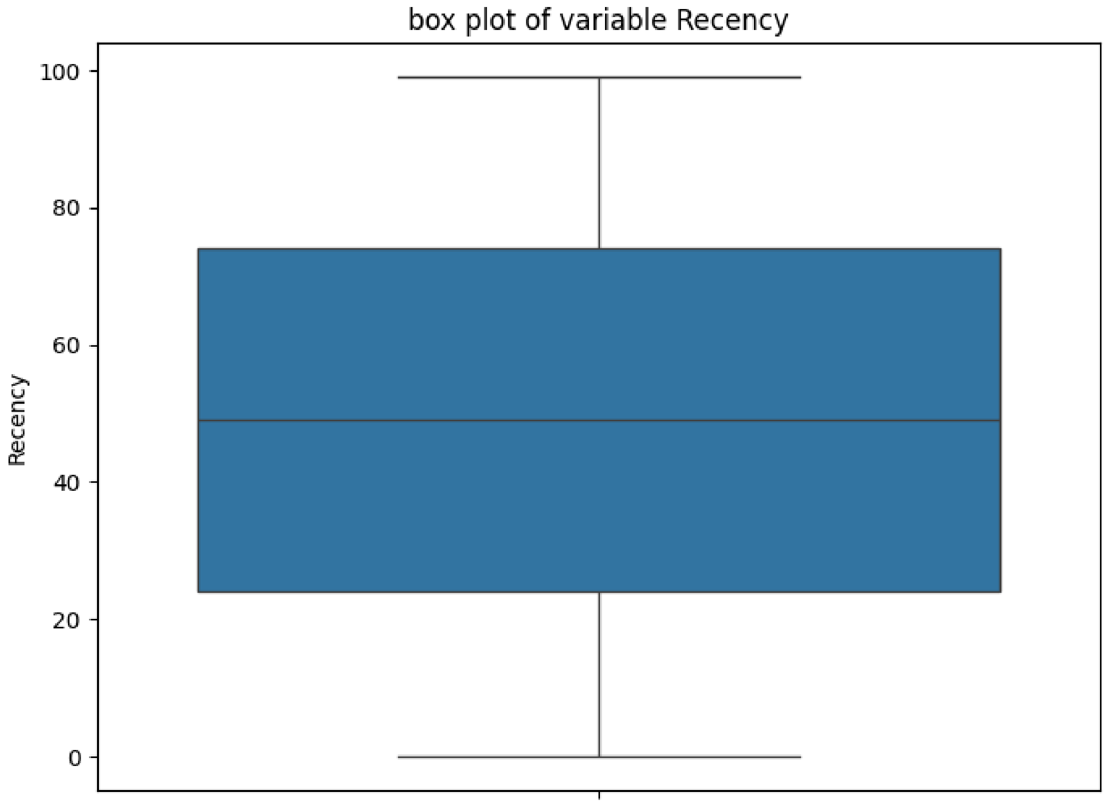

Checking for the presence of outliers in the dataset

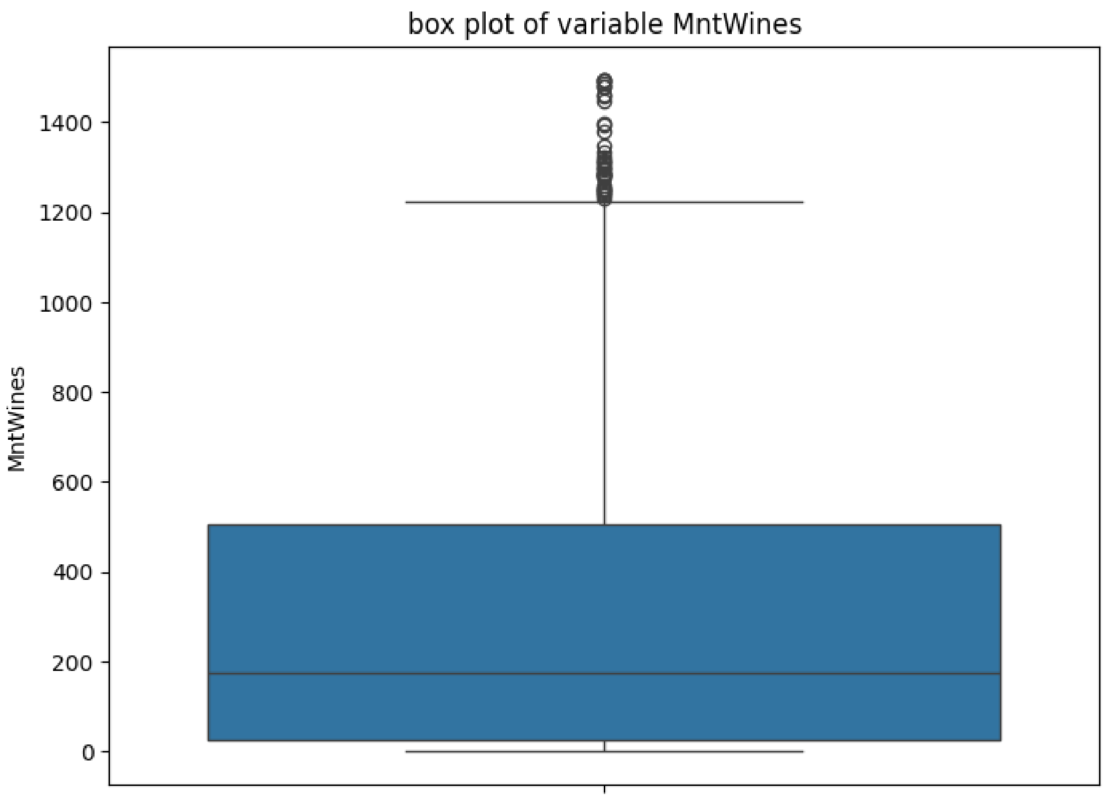
[97]:
```
for variable in selected_variables:
  plt.figure(figsize = (8, 6))
  sns.boxplot(data = selected_variables[variable])
  plt.title(f"box plot of variable {variable}")
```

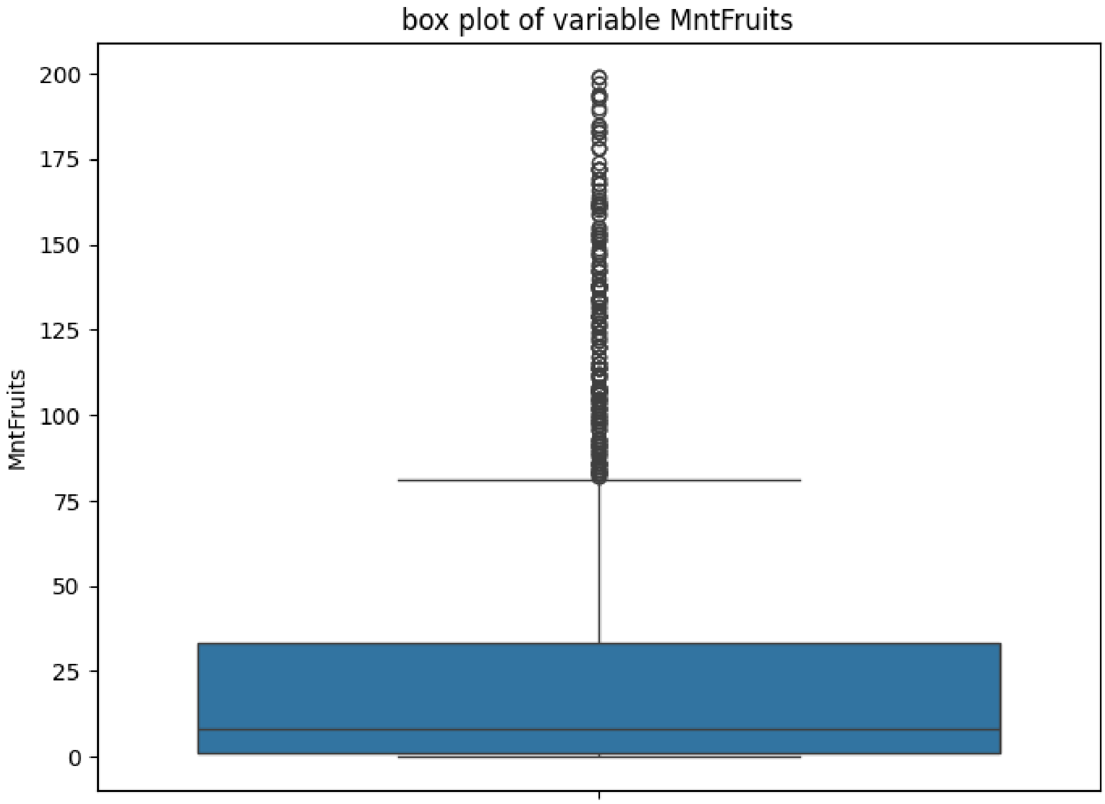

box plot of variable Income
Income
160000
140000
120000
100000
80000
60000
40000
20000
0

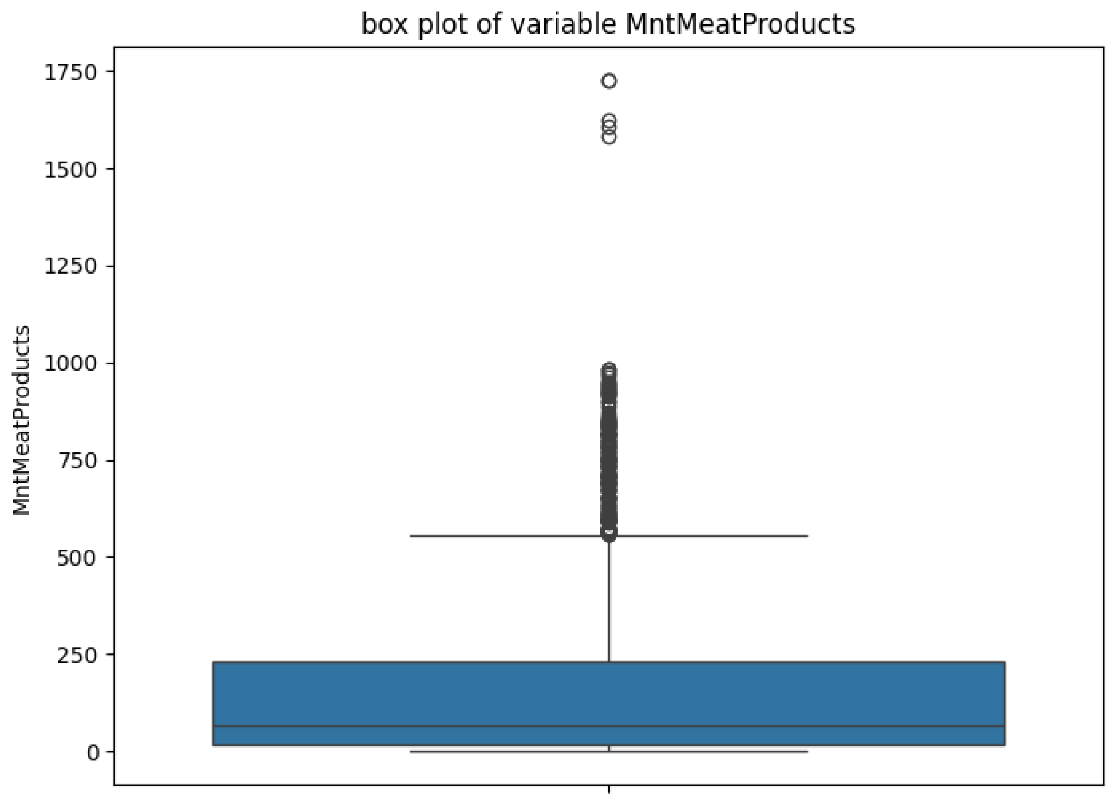

box plot of variable Kidhome
Kidhome
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

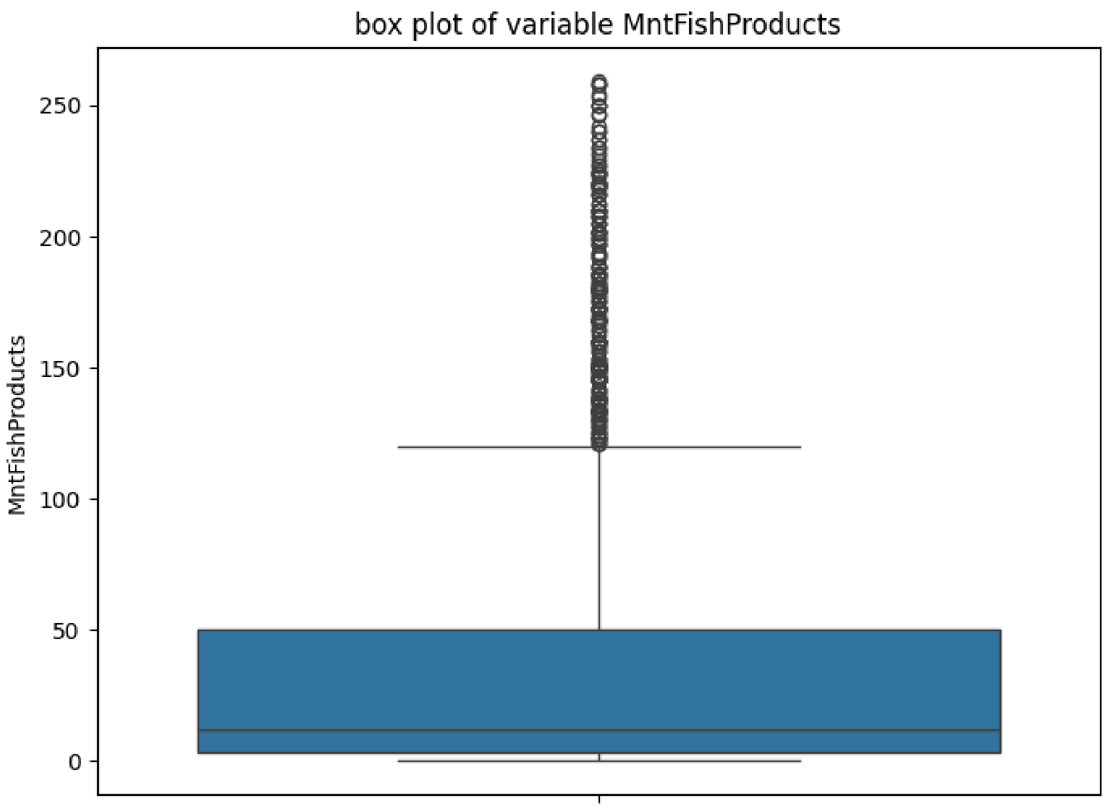

box plot of variable Teenhome
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Teenhome

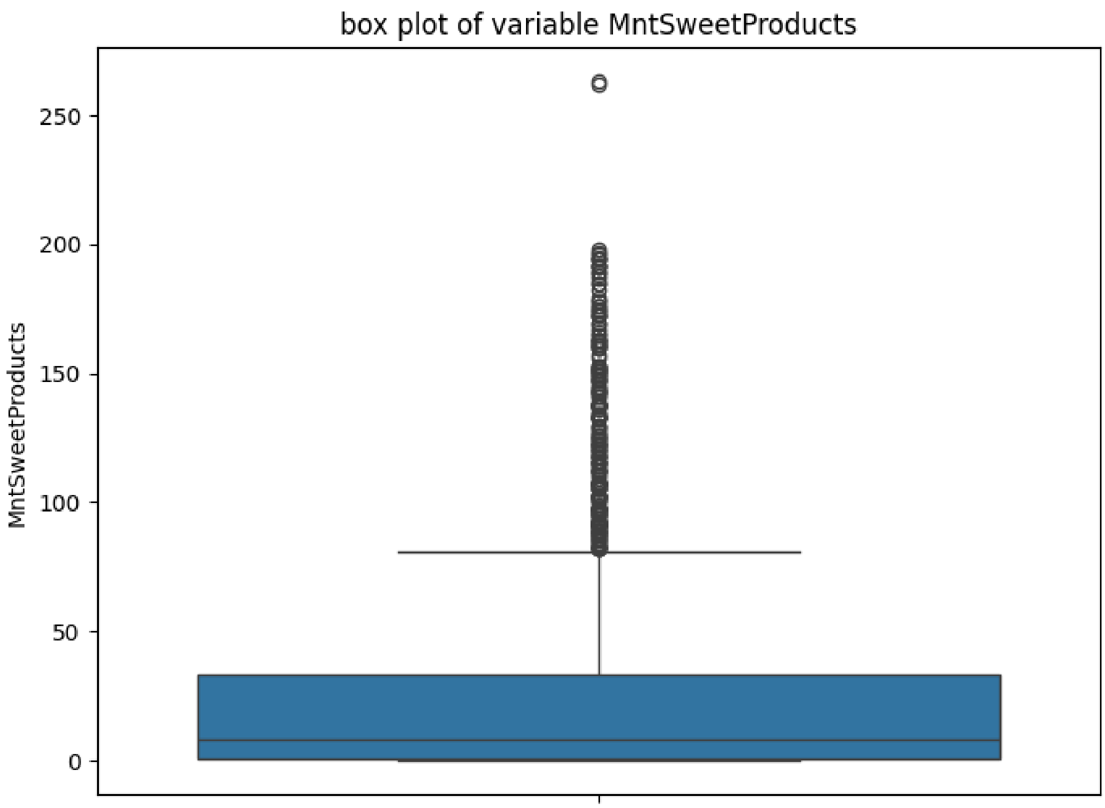

box plot of variable Recency
Recency
100
80
60
40
20
0

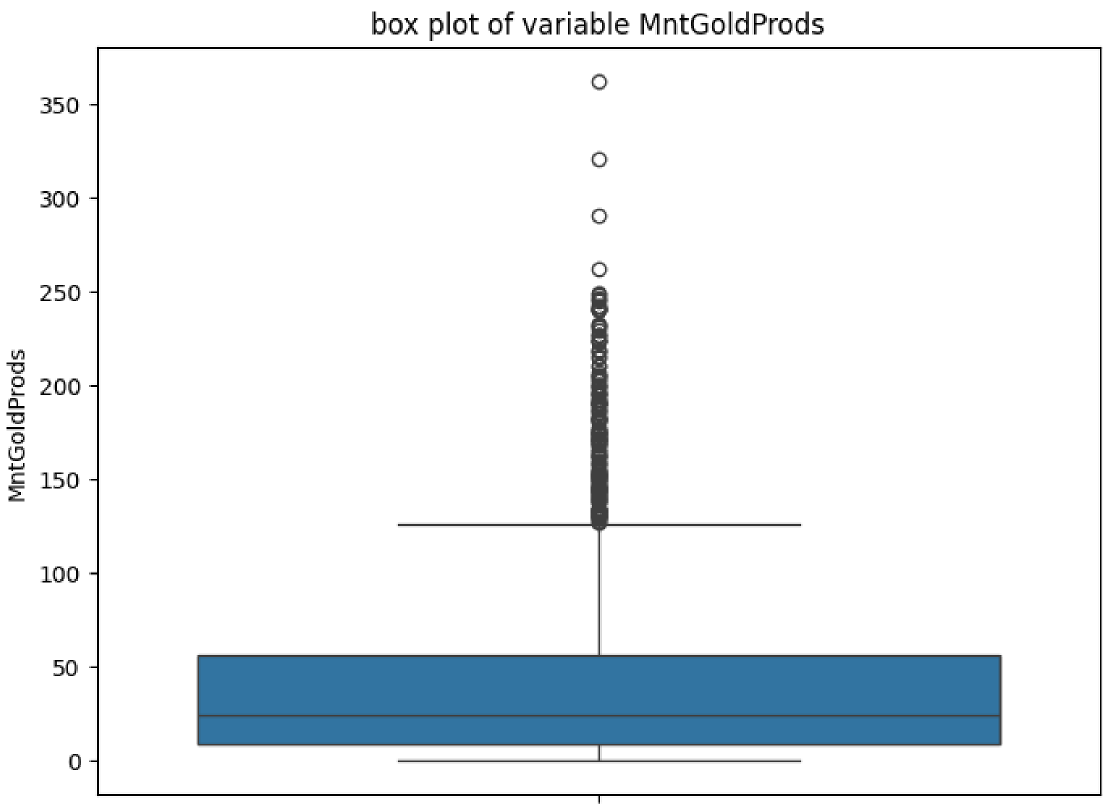

box plot of variable MntWines
MntWines
1400
1200
1000
800
600
400
200
0

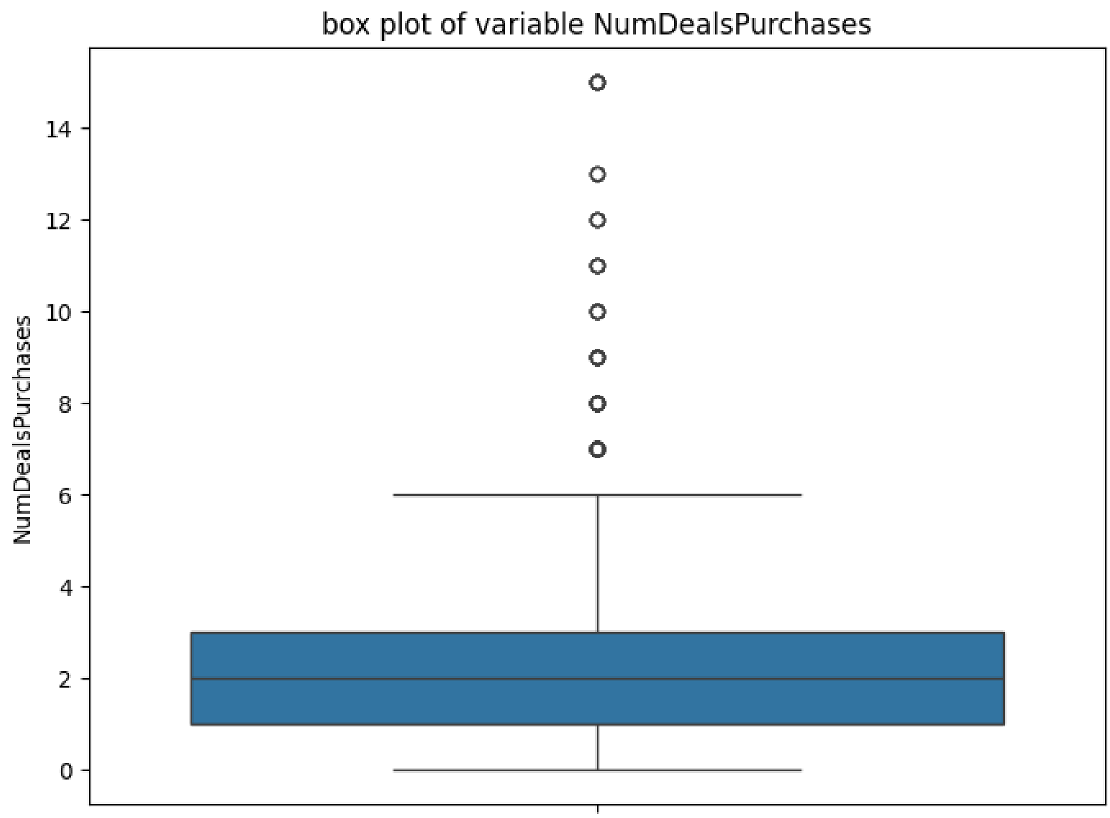

box plot of variable MntFruits
MntFruits
200
175
150
125
100
75
50
25
0

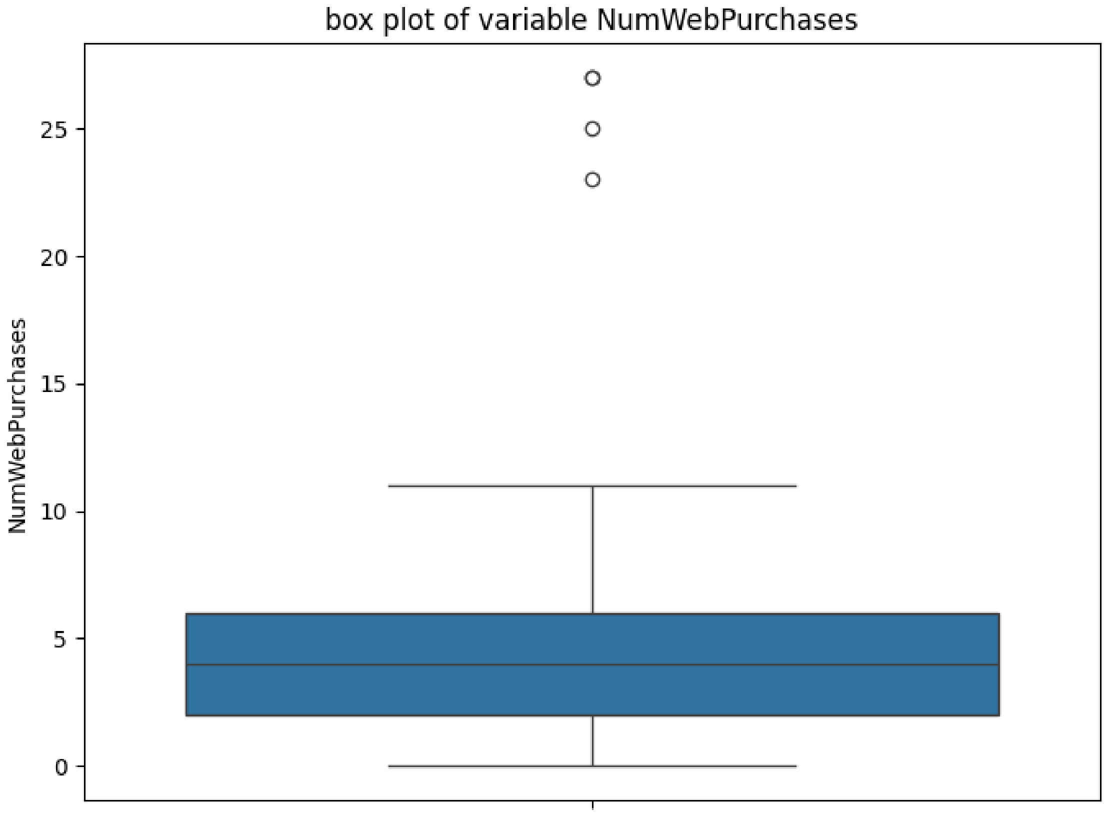

box plot of variable MntMeatProducts
MntMeatProducts
1750
1500
1250
1000
750
500
250
0

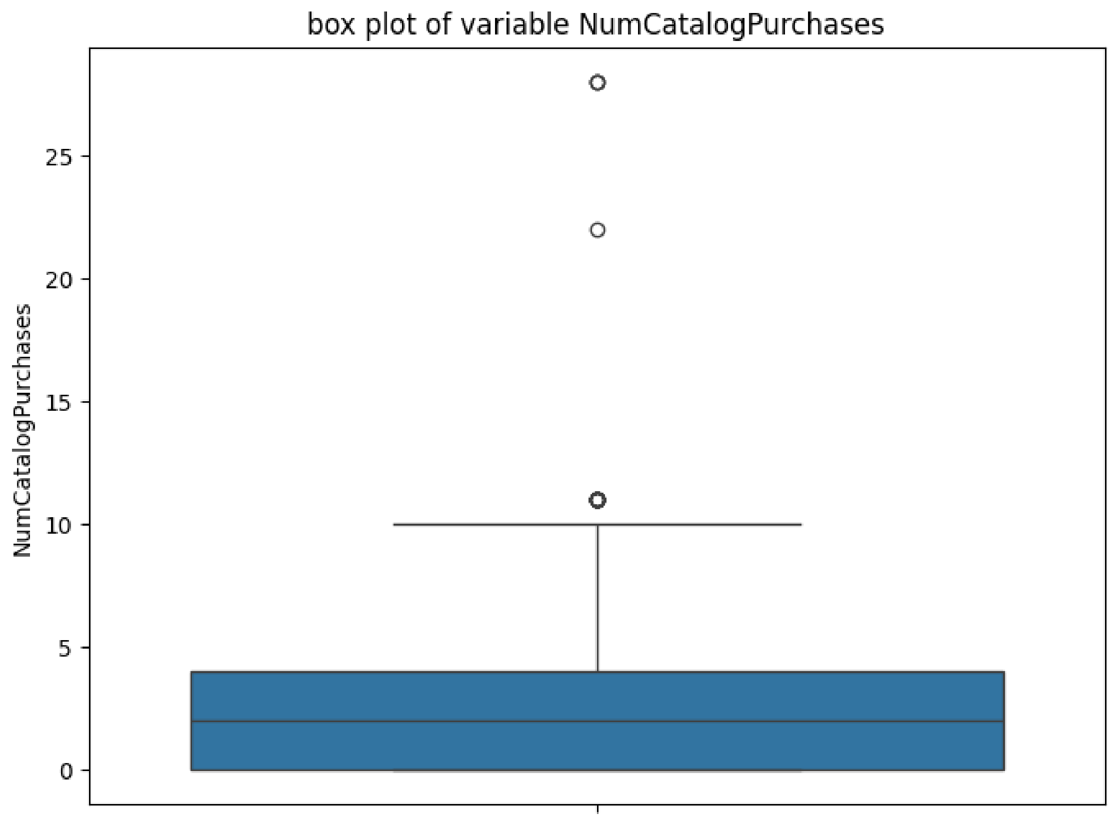

box plot of variable MntFishProducts
MntFishProducts
250
200
150
100
50
0

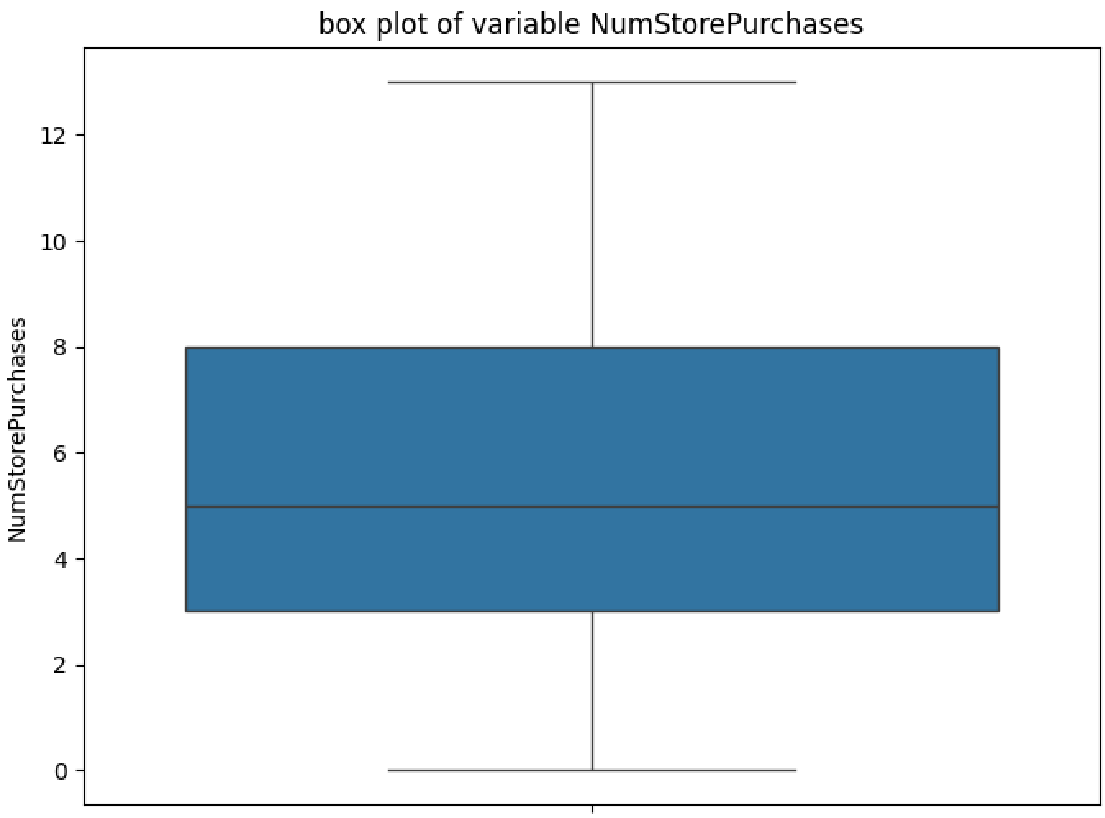

box plot of variable MntSweetProducts
MntSweetProducts
250
200
150
100
50
0

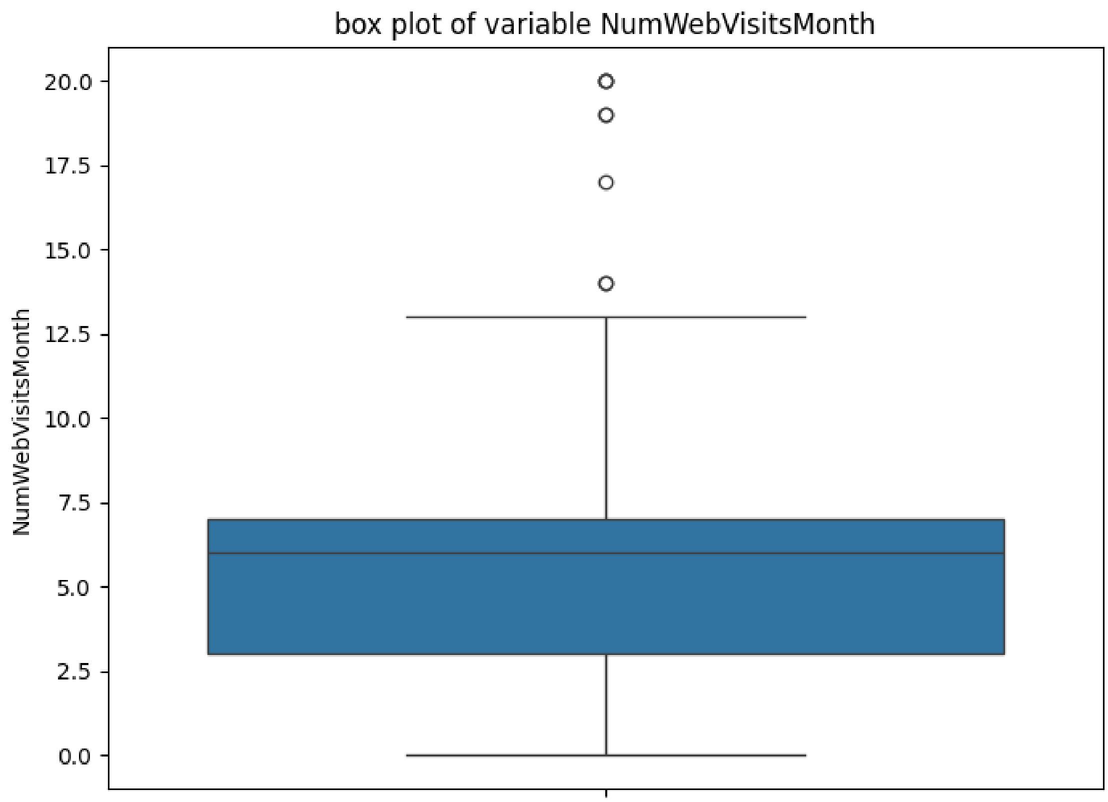

box plot of variable MntGoldProds
MntGoldProds
350
300
250
200
150
100
50
0


box plot of variable NumDealsPurchases
NumDealsPurchases
0
2
4
6
8
10
12
14


box plot of variable NumWebPurchases
NumWebPurchases
0
5
10
15
20
25


box plot of variable NumCatalogPurchases
NumCatalogPurchases
0
5
10
15
20
25


box plot of variable NumStorePurchases
NumStorePurchases
0
2
4
6
8
10
12




Calculating Total number of outliers

[134]:
```
Q1 = summary_df.loc["25%"]
Q3 = summary_df.loc["75%"]

IQR = Q3- Q1
print(IQR)
```

```
Income                 33561.5
Kidhome                    1.0
Teenhome                   1.0
Recency                   50.0
MntWines                 480.5
MntFruits                 32.0
MntMeatProducts          216.0
MntFishProducts           47.0
MntSweetProducts          32.0
MntGoldProds              47.0
NumDealsPurchases          2.0
NumWebPurchases            4.0
NumCatalogPurchases        4.0
NumStorePurchases          5.0
```

```
NumWebVisitsMonth          4.0
dtype: float64
```

[135]:
```
lower_bound = Q1- 1.5*IQR
upper_bound = Q3 + 1.5*IQR

bounds_df = pd.DataFrame({"LowerBound" : lower_bound, "UpperBound":␣
 ↪upper_bound})
print(bounds_df)
```

```
                     LowerBound  UpperBound
Income                -15626.25   118619.75
Kidhome                   -1.50        2.50
Teenhome                  -1.50        2.50
Recency                  -51.00      149.00
MntWines                -696.75     1225.25
MntFruits                -47.00       81.00
MntMeatProducts         -308.00      556.00
MntFishProducts          -67.50      120.50
MntSweetProducts         -47.00       81.00
MntGoldProds             -61.50      126.50
NumDealsPurchases         -2.00        6.00
NumWebPurchases           -4.00       12.00
NumCatalogPurchases       -6.00       10.00
NumStorePurchases         -4.50       15.50
NumWebVisitsMonth         -3.00       13.00
```

[136]:
```
outliers_lower = (summary_df < lower_bound).sum()
outliers_upper = (summary_df > upper_bound).sum()
total_outliers = outliers_lower  + outliers_upper

ouliers_count_df = pd.DataFrame({"LowerBound_outliers" :outliers_lower,␣
 ↪"UpperBound_outliers" :outliers_upper, "Total" : total_outliers})
print(ouliers_count_df)
```

```
                     LowerBound_outliers  UpperBound_outliers  Total
Income                                 0                    1      1
Kidhome                                0                    1      1
Teenhome                               0                    1      1
Recency                                0                    1      1
MntWines                               0                    2      2
MntFruits                              0                    2      2
MntMeatProducts                        0                    2      2
MntFishProducts                        0                    2      2
MntSweetProducts                       0                    2      2
MntGoldProds                           0                    2      2
NumDealsPurchases                      0                    2      2
NumWebPurchases                        0                    2      2
```

```
NumCatalogPurchases                     0                     2       2
NumStorePurchases                       0                     1       1
NumWebVisitsMonth                       0                     2       2
```

Feature engineering

```
[137]: bins = [0, 5000, 25000, 45000, 65000, 85000, 105000, 125000, 145000, 165000]

labels = ['<=5k', '>5k-25k', '>25k-45k', '>45k-65k', '>65k-85k', '>85k-105k',
 ↪'>105k-125k', '>125k-145k', '>145-165k']

df['Income_lables'] = pd.cut(df['Income'], bins = bins, labels = labels)
```

```
[138]: import datetime
df['Age'] = datetime.datetime.now().year - df['Year_Birth']
```

```
[139]: bins = [25, 45, 65, 85, 105, 125, 135]

labels = ['25-45', '>45-65', '>65-85', '>85-105', '>105-125', '>125+']

df['age_labels'] = pd.cut(df['Age'], bins = bins, labels = labels)
```

```
[140]: df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])
df['Customer_period'] =  (datetime.datetime.now().year - df['Dt_Customer'].dt.
 ↪year) * 12 + (datetime.datetime.now().month - df['Dt_Customer'].dt.month)
```

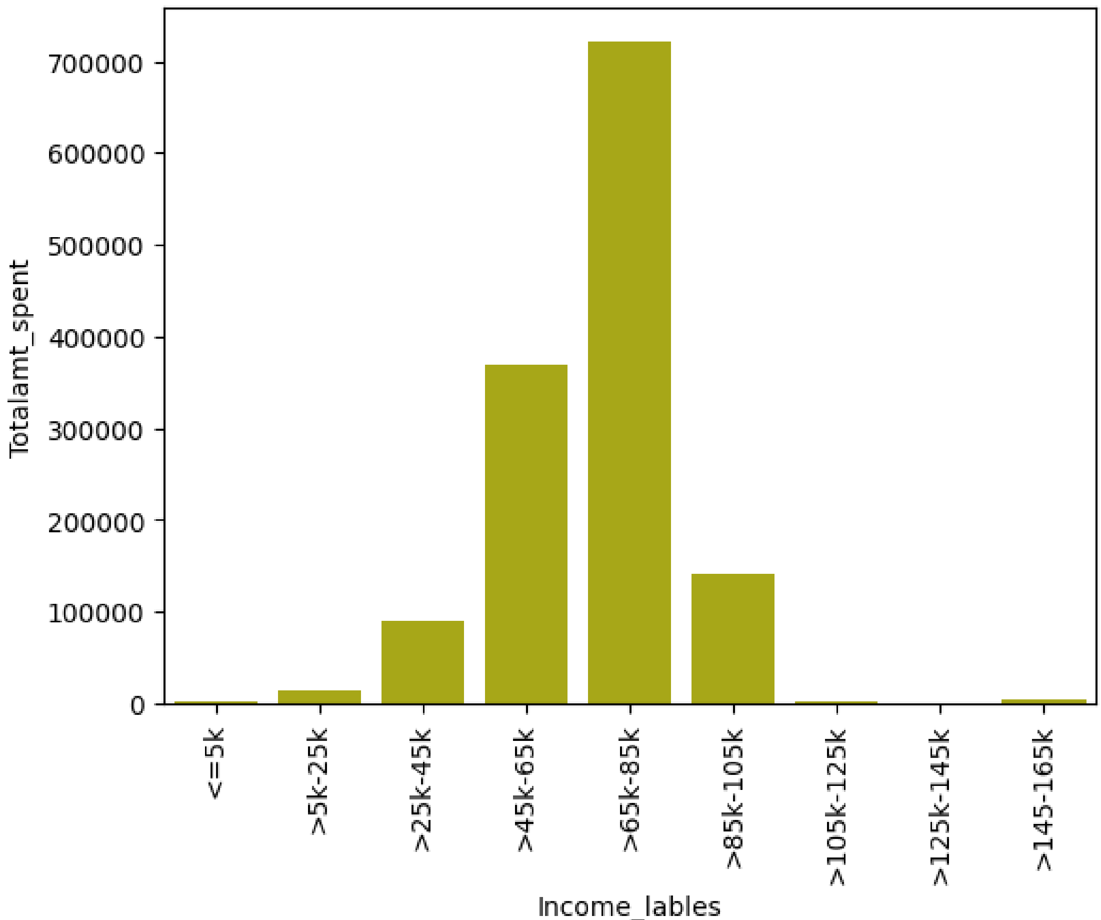
```
[141]: df['Totalamt_spent'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts']
 ↪+ df['MntFishProducts'] + df['MntSweetProducts'] + df['MntGoldProds']

df['Total_purchases'] = df['NumDealsPurchases'] + df['NumWebPurchases'] +
 ↪df['NumCatalogPurchases'] + df['NumStorePurchases']
```

Total amt spent on different products categorized under income labels.

```
[142]: import warnings

warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore')

Income_label_amt_spent = df.groupby('Income_lables')[['MntWines', 'MntFruits',
 ↪'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
 ↪'Totalamt_spent']].sum()
Income_label_amt_spent = Income_label_amt_spent.sort_values(by =
 ↪['Totalamt_spent'], ascending = False).reset_index()
print(Income_label_amt_spent)

sns.barplot(data = Income_label_amt_spent, x = 'Income_lables', y =
 ↪'Totalamt_spent', color = 'y')
```

```python
plt.xticks(rotation = 90)
plt.show()
```

```
  Income_lables  MntWines  MntFruits  MntMeatProducts  MntFishProducts  \
0      >65k-85k    350199      32641           215341            47866
1      >45k-65k    215973      14038            73676            18969
2     >85k-105k     66082       5841            49390             7832
3      >25k-45k     39914       4191            21323             6591
4       >5k-25k      2661       1467             3508             1904
5     >145-165k       203         22             4957               26
6          <=5k        27          8             1743                6
7    >105k-125k      1015        183              107              203
8    >125k-145k         0          0                0                0

   MntSweetProducts  MntGoldProds  Totalamt_spent
0             33792         41565          721404
1             13547         32050          368253
2              6624          5705          141474
3              4109         13304           89432
4              1524          4237           15301
5                 9            18            5235
6                 7           326            2117
7               283           210            2001
8                 0             0               0
```



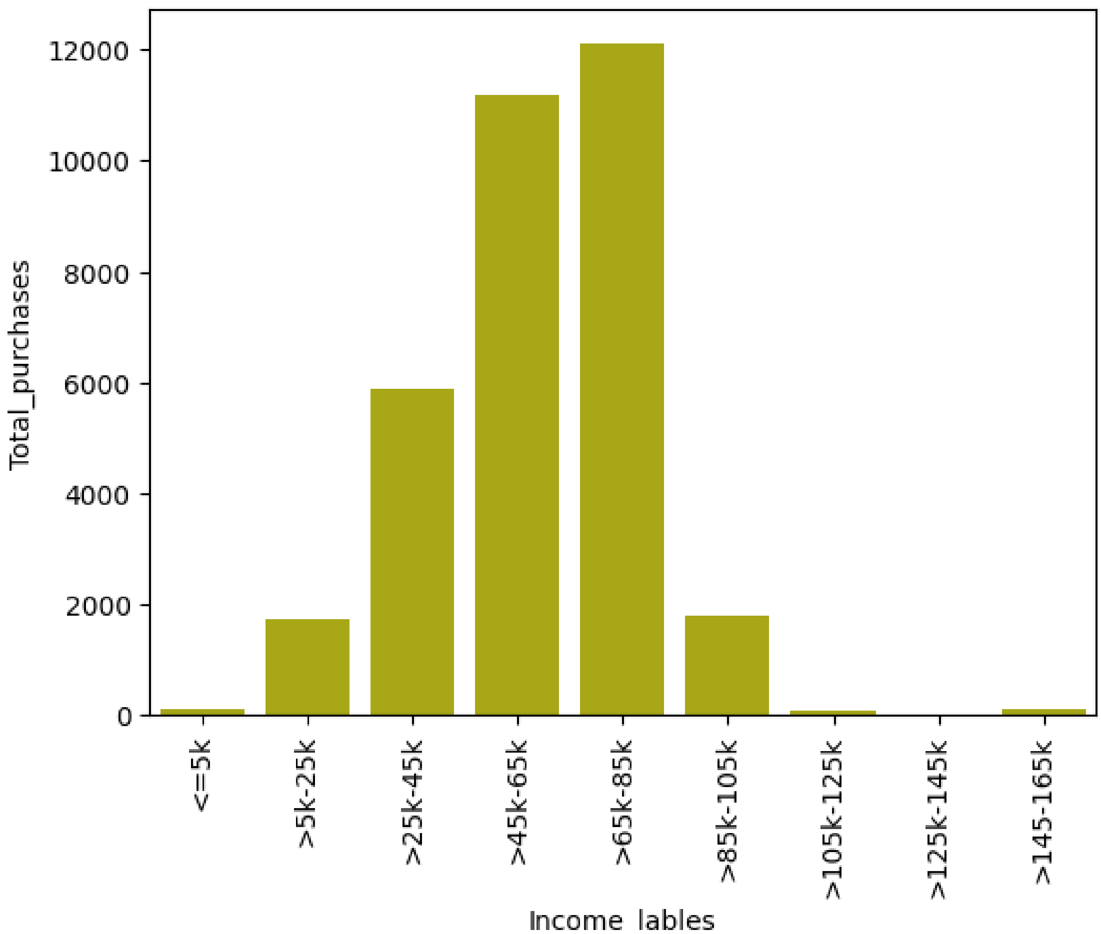

Type of purchases categorized under income labels.

[143]:
```
Income_label_purchases = df.groupby('Income_lables')[['NumDealsPurchases',␣
↪'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',␣
↪'Total_purchases']].sum()
Income_label_purchases = Income_label_purchases.sort_values(by =␣
↪['Total_purchases'], ascending = False).reset_index()
print(Income_label_purchases)

sns.barplot(data = Income_label_purchases, x = 'Income_lables', y =␣
↪'Total_purchases', color = 'y')
plt.xticks(rotation = 90)
plt.show()
```

```
  Income_lables  NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
0      >65k-85k                970             3122                 3072
1      >45k-65k               2024             3271                 1646
2      >25k-45k               1510             1660                  439
```

| | | | | |
|---|---|---|---|---|
| 3 | >85k-105k | 75 | 467 | 548 |
| 4 | >5k-25k | 491 | 468 | 99 |
| 5 | >145-165k | 30 | 1 | 78 |
| 6 | <=5k | 45 | 25 | 28 |
| 7 | >105k-125k | 0 | 36 | 8 |
| 8 | >125k-145k | 0 | 0 | 0 |

| | NumStorePurchases | Total_purchases |
|---|---|---|
| 0 | 4945 | 12109 |
| 1 | 4229 | 11170 |
| 2 | 2292 | 5901 |
| 3 | 708 | 1798 |
| 4 | 662 | 1720 |
| 5 | 3 | 112 |
| 6 | 0 | 98 |
| 7 | 13 | 57 |
| 8 | 0 | 0 |



Purchases categorized for number of teenagers and kids in each household

[144]:
```
df.groupby('Teenhome')[['Totalamt_spent', 'Total_purchases']].sum().
↪reset_index()
```

[144]:
```
   Teenhome  Totalamt_spent  Total_purchases
0         0          802199            16061
1         1          524091            16338
2         2           30636              881
```

[145]:
```
df.groupby('Kidhome')[['Totalamt_spent', 'Total_purchases']].sum().reset_index()
```

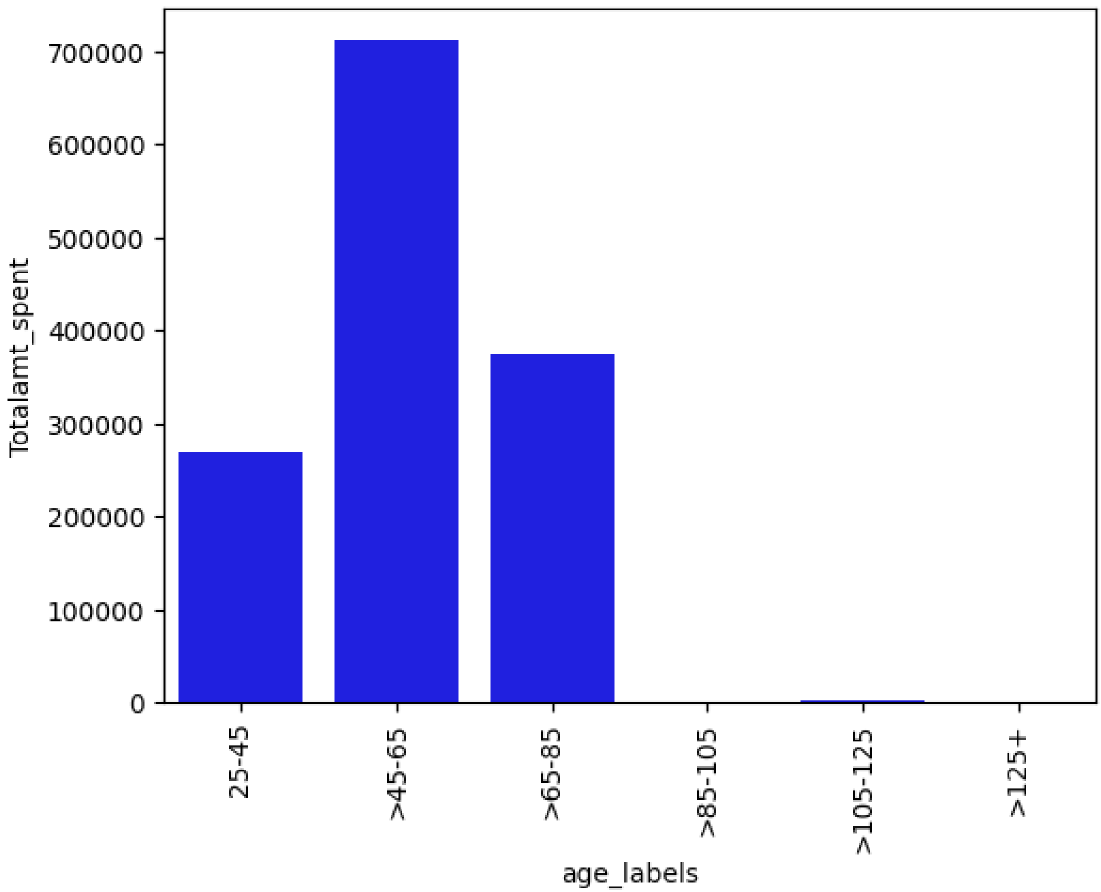
[145]:
```
   Kidhome  Totalamt_spent  Total_purchases
0        0         1165330            23395
1        1          184624             9416
2        2            6972              469
```

Total amt spent on different products categorized under Age labels.

[146]:
```
import warnings

warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore')

Age_label_amt_spent = df.groupby('age_labels')[['MntWines', 'MntFruits',␣
↪'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',␣
↪'Totalamt_spent']].sum()
Age_label_amt_spent = Age_label_amt_spent.sort_values(by = ['Totalamt_spent'],␣
↪ascending = False).reset_index()
print(Age_label_amt_spent)

sns.barplot(data = Age_label_amt_spent, x = 'age_labels', y = 'Totalamt_spent',␣
↪color = 'b')
plt.xticks(rotation = 90)
plt.show()
```

```
  age_labels  MntWines  MntFruits  MntMeatProducts  MntFishProducts  \
0     >45-65    367850      30701           187090            42260
1     >65-85    196310      14039           100228            22594
2      25-45    115869      14013            86057            19077
3   >105-125       770        150              570              111
4      >125+         8          0                5                7
5    >85-105         0          0                0                0

   MntSweetProducts  MntGoldProds  Totalamt_spent
0             31383         51898          711182
1             15432         26147          374750
2             13737         20301          269054
```

```
3                  68             249           1918
4                   0               2             22
5                   0               0              0
```



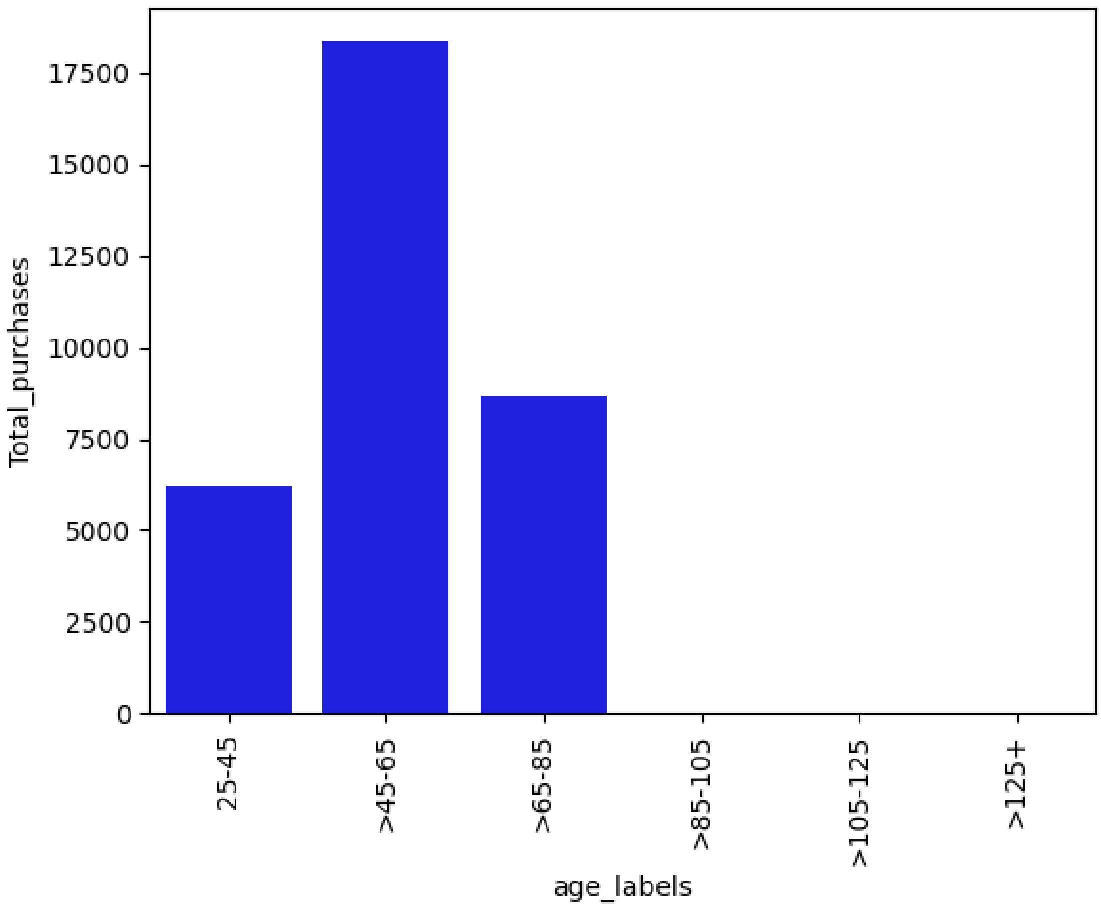

Type of purchases categorized under Age labels.

[147]:
```python
Age_label_purchases = df.groupby('age_labels')[['NumDealsPurchases',
 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
 'Total_purchases']].sum()
Age_label_purchases = Age_label_purchases.sort_values(by = ['Total_purchases'],
 ascending = False).reset_index()
print(Age_label_purchases)

sns.barplot(data = Age_label_purchases, x = 'age_labels', y =
 'Total_purchases', color = 'b')
plt.xticks(rotation = 90)
plt.show()
```

```
  age_labels  NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
```

| | | | | |
|---|---|---|---|---|
| 0 | >45-65 | 3097 | 5068 | 3101 |
| 1 | >65-85 | 1213 | 2395 | 1713 |
| 2 | 25-45 | 891 | 1677 | 1141 |
| 3 | >105-125 | 2 | 6 | 7 |
| 4 | >125+ | 1 | 1 | 0 |
| 5 | >85-105 | 0 | 0 | 0 |

| | NumStorePurchases | Total_purchases |
|---|---|---|
| 0 | 7089 | 18355 |
| 1 | 3331 | 8652 |
| 2 | 2539 | 6248 |
| 3 | 6 | 21 |
| 4 | 2 | 4 |
| 5 | 0 | 0 |

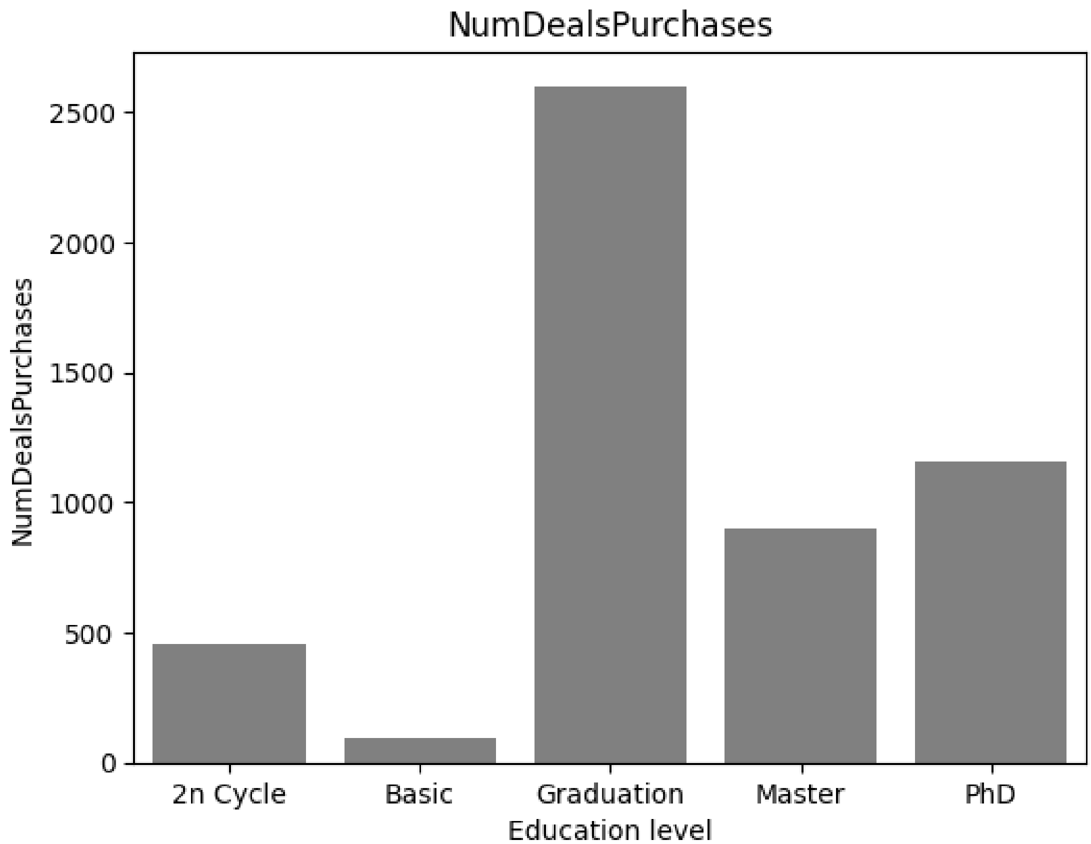



Type of purchases categorized under different Education levels.

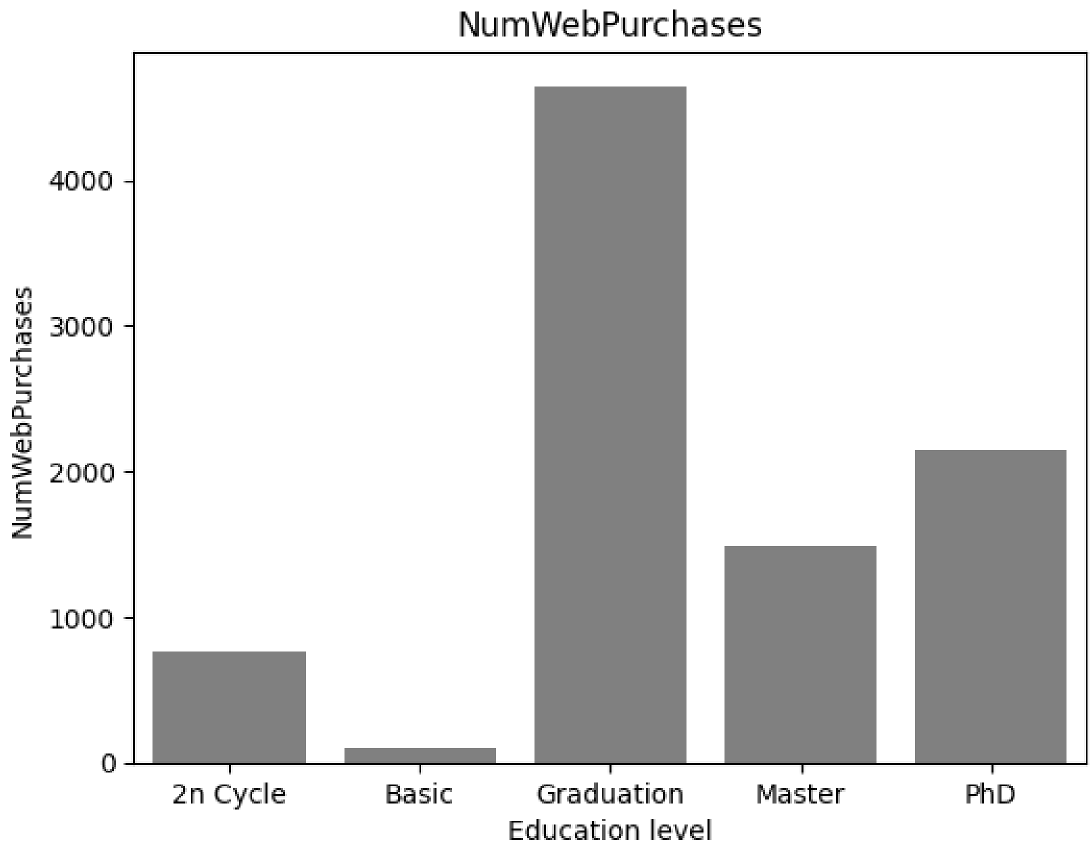
[148]:

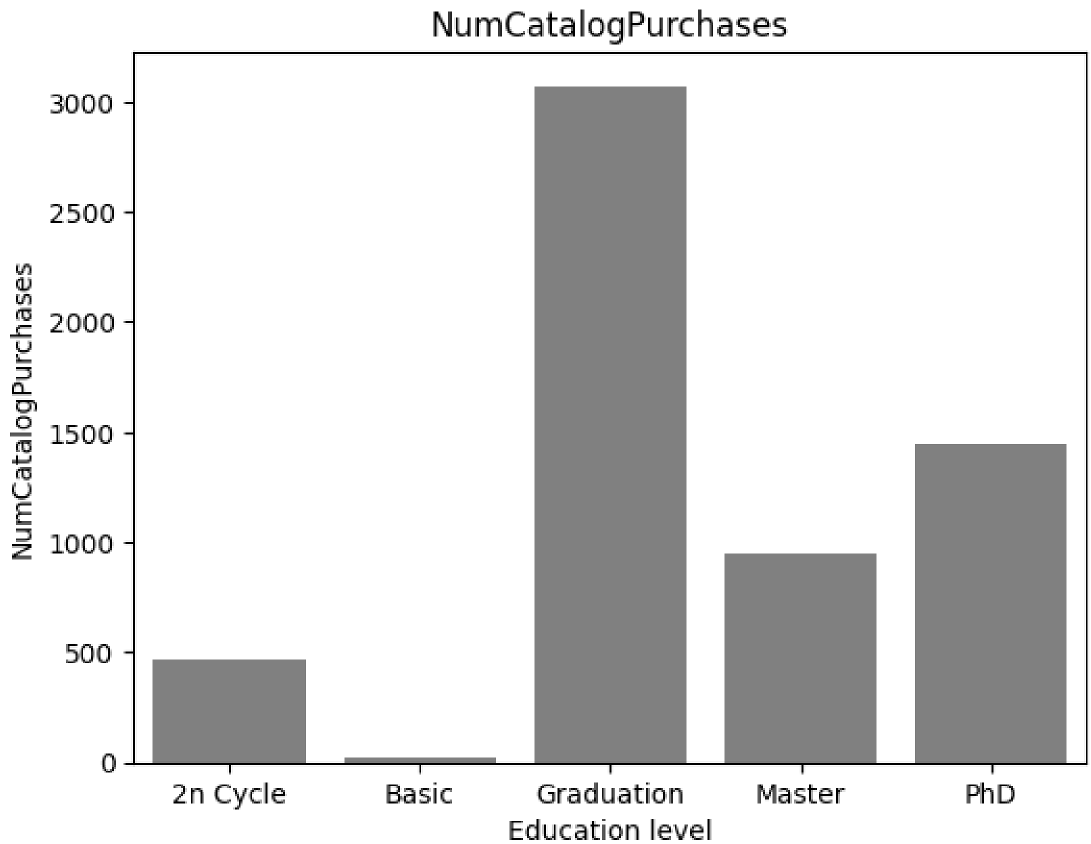
```
Purchases_by_Education_level = df.groupby('Education')[['NumDealsPurchases',␣
↪'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',␣
↪'NumWebVisitsMonth']].sum()
Purchases_by_Education_level = Purchases_by_Education_level.reset_index()
print(Purchases_by_Education_level)

purchase_types = ['NumDealsPurchases', 'NumWebPurchases',␣
↪'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']

for purchase_type in purchase_types:
  sns.barplot(data = Purchases_by_Education_level, x = 'Education', y =␣
↪purchase_type, color = 'grey')
  plt.title(purchase_type)
  plt.xlabel('Education level')
  plt.show()
```

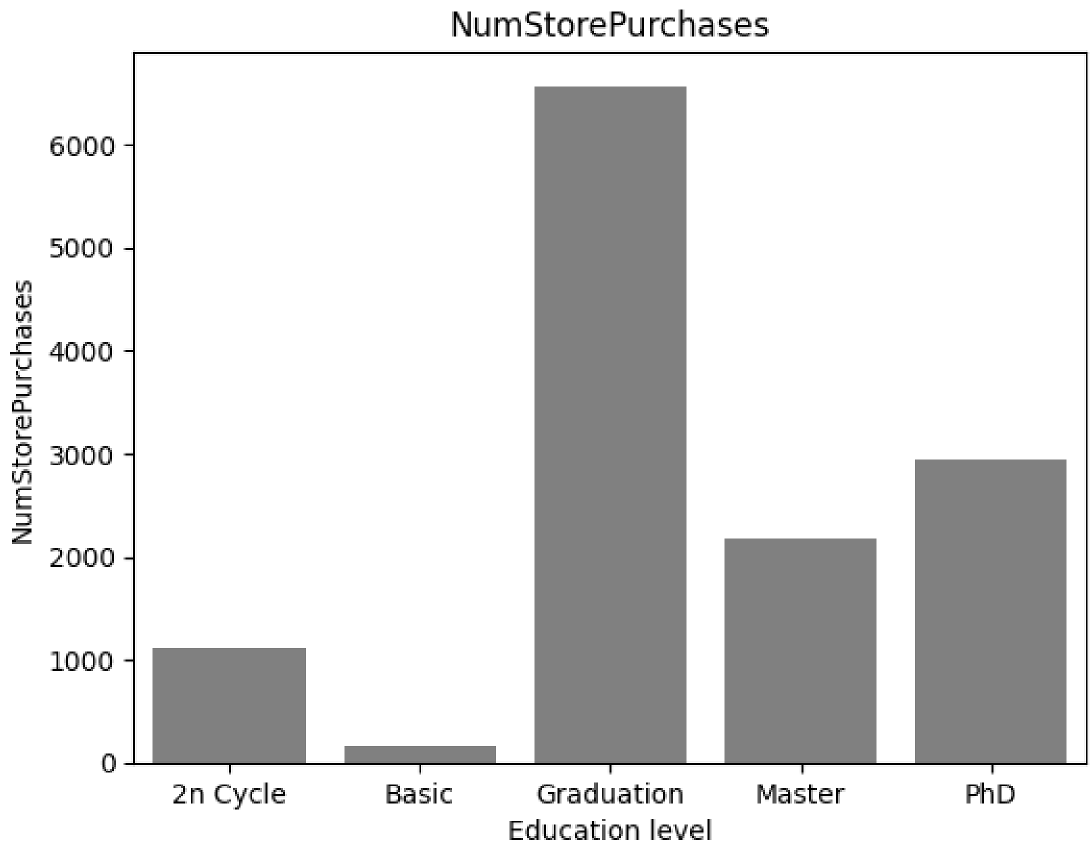
```
    Education  NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
0    2n Cycle                456              757                  471
1       Basic                 97              102                   26
2  Graduation               2599             4646                 3071
3      Master                898             1492                  951
4         PhD               1154             2150                 1443

   NumStorePurchases  NumWebVisitsMonth
0               1118               1107
1                154                371
2               6567               5953
3               2182               1916
4               2946               2556
```

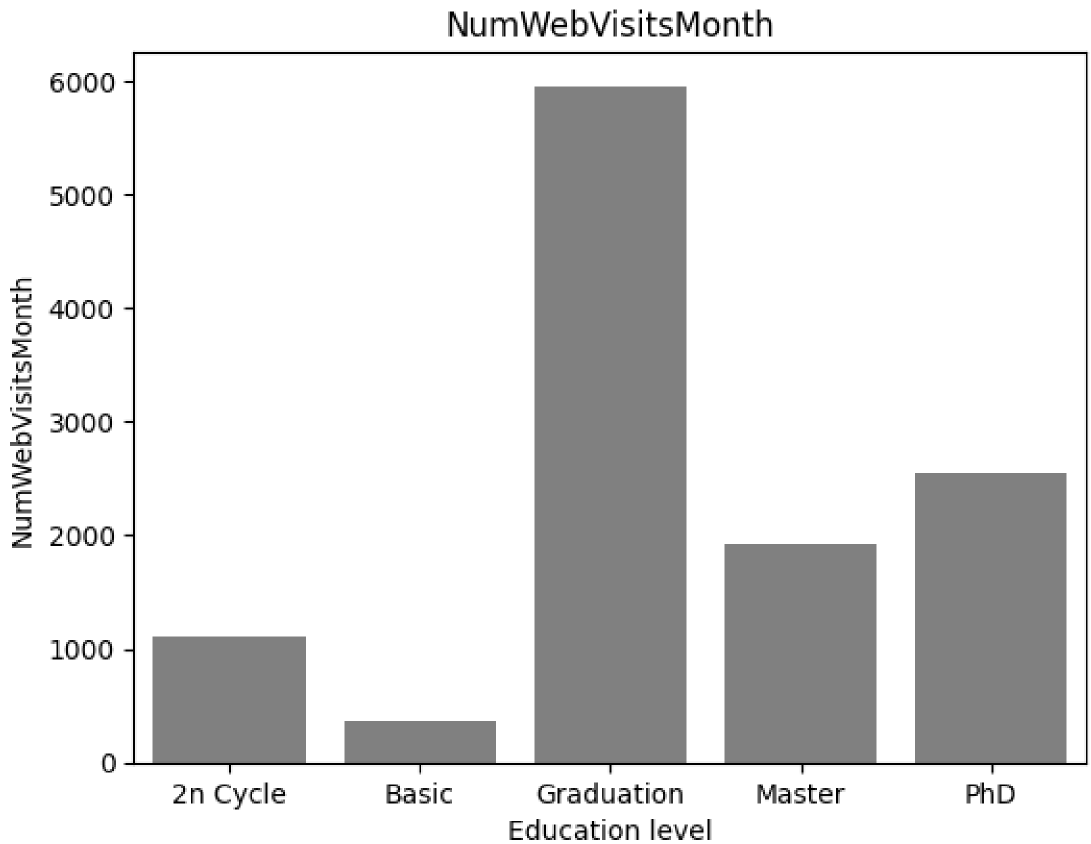


NumDealsPurchases
2500
2000
1500
1000
500
0
NumDealsPurchases
2n Cycle
Basic
Graduation
Master
PhD
Education level


NumWebPurchases
NumWebPurchases
4000
3000
2000
1000
0
2n Cycle
Basic
Graduation
Master
PhD
Education level


NumCatalogPurchases
NumCatalogPurchases
0
500
1000
1500
2000
2500
3000
2n Cycle
Basic
Graduation
Master
PhD
Education level


NumStorePurchases
6000
5000
4000
3000
2000
1000
0
NumStorePurchases
2n Cycle
Basic
Graduation
Master
PhD
Education level




Number of customers attracted for each Different campaign

[149]:
```
count_of_offers = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3',␣
↪'AcceptedCmp4', 'AcceptedCmp5']].sum()
count_of_offers = count_of_offers.reset_index()
count_of_offers.columns = ['Campaign', 'count']
count_of_offers
```

[149]:

| | Campaign | count |
|---|---|---|
| 0 | AcceptedCmp1 | 144 |
| 1 | AcceptedCmp2 | 30 |
| 2 | AcceptedCmp3 | 163 |
| 3 | AcceptedCmp4 | 167 |
| 4 | AcceptedCmp5 | 163 |

Total number of Complaints

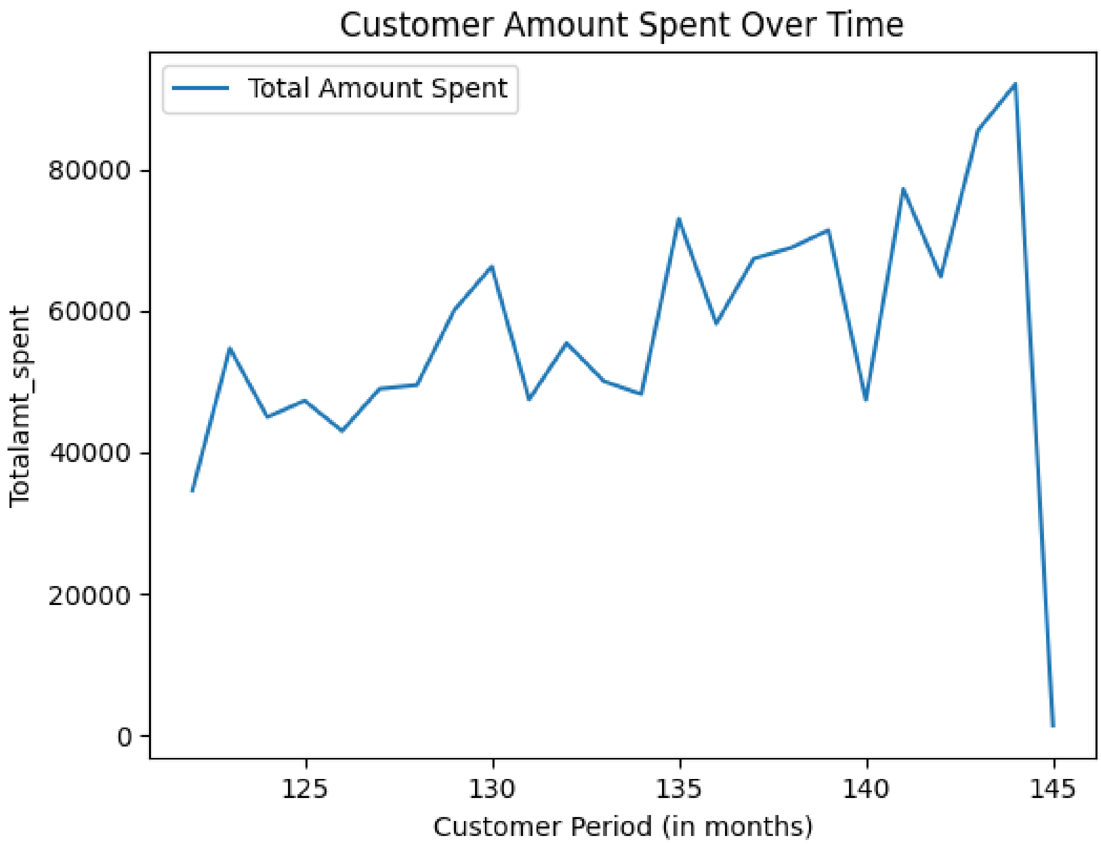
[150]:
```
Total_complaints = df['Complain'].sum()
Total_complaints
```

[150]: 21

Total amt spent withrespect to customer period. (in months)

[151]:
```python
customer_purchases_wtr_period = df.
  ↪groupby('Customer_period')[['Totalamt_spent', 'Total_purchases']].sum()
customer_purchases_wtr_period = customer_purchases_wtr_period.sort_values(by =␣
  ↪['Totalamt_spent'], ascending = False).reset_index()
print(customer_purchases_wtr_period)

sns.lineplot(data=customer_purchases_wtr_period, x='Customer_period',␣
  ↪y='Totalamt_spent', label='Total Amount Spent')
plt.title('Customer Amount Spent Over Time')
plt.xlabel('Customer Period (in months)')
plt.show()
```

```
    Customer_period  Totalamt_spent  Total_purchases
0               144           92046             2026
1               143           85492             1812
2               141           77190             1722
3               135           72954             1593
4               139           71351             1789
5               138           68879             1671
6               137           67322             1667
7               130           66221             1704
8               142           64802             1598
9               129           60064             1394
10              136           58135             1369
11              132           55393             1478
12              123           54648             1475
13              133           49986             1358
14              128           49455             1243
15              127           48932             1306
16              134           48171             1335
17              131           47432             1236
18              140           47382             1116
19              125           47247             1306
20              124           44938             1154
21              126           42970              972
22              122           34559              911
23              145            1357               45
```

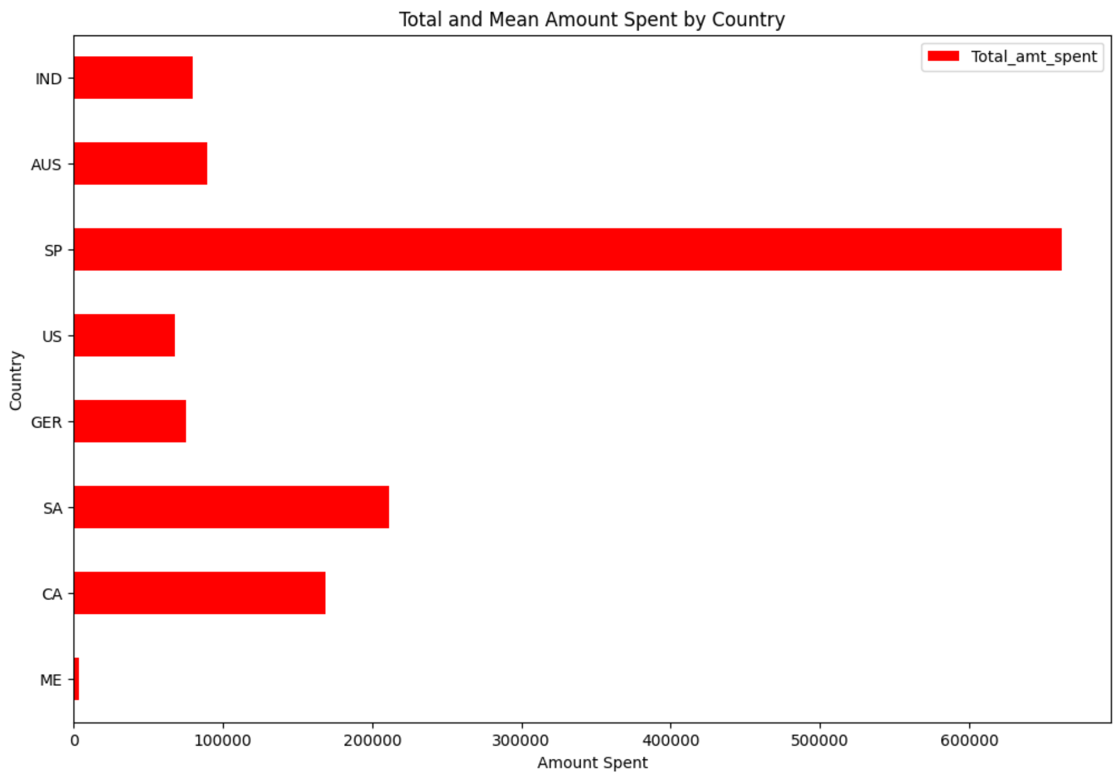



Total amt spent and mean amt spent for each country

[152]:
```
Country_purchases = df.groupby('Country')['Totalamt_spent'].aggregate({'sum',␣
↪'mean'})
Country_purchases = Country_purchases.sort_values(by = ['mean'], ascending =␣
↪False).reset_index()
Country_purchases.columns = ['country', 'Total_amt_spent', 'mean_amt_spent']
print(Country_purchases)

Country_purchases.plot(kind='barh', x='country', y='Total_amt_spent',␣
↪figsize=(12, 8), color = 'r')
plt.title('Total and Mean Amount Spent by Country')
plt.xlabel('Amount Spent')
plt.ylabel('Country')
plt.show()
```

```
  country  Total_amt_spent  mean_amt_spent
0      ME             3122     1040.666667
1      CA           168532      628.850746
2      SA           211009      628.002976
```

```
3     GER        74913      624.275000
4      US        67882      622.770642
5      SP       662220      604.767123
6     AUS        89763      561.018750
7     IND        79485      537.060811
```




Hypothesis testing

Is income of customers dependent on their education

[153]:
```
from scipy.stats import f_oneway

alpha = 0.05
 # H0: The means of Incomes between different education levels would be equal
 # H1: The means of Incomes between different education levels would be not␣
 ↪equal

education_groups = [group['Income'].values for name, group in df.
 ↪groupby('Education')]

f_statistic, p_value = f_oneway(*education_groups)
print(f_statistic, p_value)
print("<------------------->")
```

```
if p_value < 0.05:
    print("Reject the null hypothesis. Income depends on education level.")
else:
    print("Fail to reject null hypothesis : Income doesn't depend on Education␣
↪levels.")
```

```
35.42763066856272 9.87796950058819e-29
<------------------->
Reject the null hypothesis. Income depends on education level.
```

Do higher income people spend more (take in account spending in all categories together)

[154]:
```
import scipy.stats as stats

corr_coefficient, p_value = stats.pearsonr(df['Income'], df['Totalamt_spent'])

print(f"Pearson correlation coefficient: {corr_coefficient}")
print(f"P-value: {p_value}")

print("<---------------->")

if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant linear␣
↪relationship between income and purchases.")
else:
    print("Fail to reject the null hypothesis. No significant linear␣
↪relationship exists between income and purchases.")
```

```
Pearson correlation coefficient: 0.7706290398754154
P-value: 0.0
<---------------->
Reject the null hypothesis. There is a significant linear relationship between
income and purchases.
```

Do couples spend more or less money on wine than people living alone (set 'Married','Together':'In couple' and 'Divorced','Single','Absurd','Widow','YOLO':'Alone')

[155]:
```
df['Living_Status'] = df['Marital_Status'].apply(lambda x: 'Couple' if x in␣
↪['Married', 'Together'] else 'Alone')

wine_spending_couples = df[df['Living_Status'] == 'Couple']['MntWines']
wine_spending_alone = df[df['Living_Status'] == 'Alone']['MntWines']

import scipy.stats as stats

# Perform the t-test
```

```
t_statistic, p_value = stats.ttest_ind(wine_spending_couples,␣
↪wine_spending_alone, equal_var=False)  # Use equal_var=False if variances␣
↪are unequal

print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

print("<-------------------->")

if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in␣
↪wine spending between couples and people living alone.")
else:
    print("Fail to reject the null hypothesis. No significant difference in␣
↪wine spending exists between couples and people living alone.")
```

```
T-statistic: -0.2711337908368919
P-value: 0.7863223090103292
<-------------------->
Fail to reject the null hypothesis. No significant difference in wine spending
exists between couples and people living alone.
```

Are people with lower income are more attracted towards campaign or simply put accept more campaigns. ( create two income brackets one below median , other above median income and create a column which tells if they have ever accepted any campaign)

[156]:
```
median_income = df['Income'].median()

df['Income_Bracket'] = df['Income'].apply(lambda x: 'Below_Median' if x <␣
↪median_income else 'Above_Median')

df['Accepted_Any_Campaign'] = df[['AcceptedCmp1', 'AcceptedCmp2',␣
↪'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1).apply(lambda x:␣
↪1 if x > 0 else 0)



import scipy.stats as stats

# Create a contingency table
contingency_table = pd.crosstab(df['Income_Bracket'],␣
↪df['Accepted_Any_Campaign'])

# Perform the Chi-Square test
chi2, p, dof, expected = stats.chi2_contingency(contingency_table)

print(f"Chi-Square statistic: {chi2}")
print(f"P-value: {p}")
```

```
print("<------------------>")

if p < 0.05:
    print("Reject the null hypothesis. There is a significant association␣
 ↪between income level and campaign acceptance.")
else:
    print("Fail to reject the null hypothesis. No significant association␣
 ↪between income level and campaign acceptance.")
```

```
Chi-Square statistic: 138.8199834041559
P-value: 4.8224046007539564e-32
<------------------>
Reject the null hypothesis. There is a significant association between income
level and campaign acceptance.
```

[157]:

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

[ ]:

```
!pip install nbconvert

!apt-get install texlive texlive-xetex texlive-latex-extra pandoc

!jupyter nbconvert --to pdf "/content/drive/MyDrive/Colab Notebooks/Campaign␣
 ↪dataset@DhanunjayaReddy.ipynb".ipynb
```

```
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-
packages (6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (2.1.5)
```