In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Import Data

In [2]:
```
shop=pd.read_csv("shopping.csv")
shop.head()
```

Out[2]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	Pag
0	0	0.0	0	0.0	1	0.000000	0.20	0.20	
1	0	0.0	0	0.0	2	64.000000	0.00	0.10	
2	0	0.0	0	0.0	1	0.000000	0.20	0.20	
3	0	0.0	0	0.0	2	2.666667	0.05	0.14	
4	0	0.0	0	0.0	10	627.500000	0.02	0.05	

Basic Metric

In [3]:
```
shop.shape
```

Out[3]:
```
(12330, 18)
```

In [4]:
```
shop.size
```

Out[4]:
```
221940
```

In [5]:
```
shop.ndim
```

Out[5]: 2

In [6]:
```
shop.dtypes
```

Out[6]:
```
Administrative                int64
Administrative_Duration     float64
Informational                 int64
Informational_Duration      float64
ProductRelated                int64
ProductRelated_Duration     float64
BounceRates                 float64
ExitRates                   float64
PageValues                  float64
SpecialDay                  float64
Month                        object
OperatingSystems              int64
Browser                       int64
Region                        int64
TrafficType                   int64
VisitorType                  object
Weekend                        bool
Revenue                        bool
dtype: object
```

In [7]:
```
shop.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
```

In [8]:

```
shop.describe()
```

Out[8]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRa
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000(
mean	2.315166	80.818611	0.503569	34.472398	31.731468	1194.746220	0.022191	0.043(
std	3.321784	176.779107	1.270156	140.749294	44.475503	1913.669288	0.048488	0.048!
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000(
25%	0.000000	0.000000	0.000000	0.000000	7.000000	184.137500	0.000000	0.014:
50%	1.000000	7.500000	0.000000	0.000000	18.000000	598.936905	0.003112	0.025'
75%	4.000000	93.256250	0.000000	0.000000	38.000000	1464.157214	0.016813	0.050(
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	0.200(

Checking for NULL

In [9]:

```
shop.isna().sum()
```

Out[9]:

```
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
```

- There is no null in data set

Checking for duplicate

In [10]:
```
shop.duplicated().value_counts()
```

Out[10]:
```
False    12205
True       125
Name: count, dtype: int64
```

Non-Graphical Analysis

In [11]:
```
col = ['Administrative','Informational','Month','OperatingSystems',
          'Browser','Region','TrafficType','Weekend','VisitorType','Revenue']
for i in col:
    print(shop.value_counts(i))
    print()
    print("*"*100)
```

```
Administrative
0     5768
1     1354
2     1114
3      915
4      765
5      575
6      432
7      338
8      287
9      225
10     153
11     105
12      86
13      56
14      44
15      38
16      24
17      16
18      12
19       6
22       4
24       4
23       3
20       2
21       2
26       1
27       1
Name: count, dtype: int64

****************************************************************************************************
Informational
0     9699
1     1041
2      728
3      380
4      222
5       99
6       78
7       36
9       15
8       14
10       7
12       5
```

```
14       2
11       1
13       1
16       1
24       1
Name: count, dtype: int64

****************************************************************************************************
Month
May     3364
Nov     2998
Mar     1907
Dec     1727
Oct      549
Sep      448
Aug      433
Jul      432
June     288
Feb      184
Name: count, dtype: int64

****************************************************************************************************
OperatingSystems
2    6601
1    2585
3    2555
4     478
8      79
6      19
7       7
5       6
Name: count, dtype: int64

****************************************************************************************************
Browser
2     7961
1     2462
4      736
5      467
6      174
10     163
8      135
3      105
13      61
```

```
7       49
12      10
11       6
9        1
Name: count, dtype: int64

****************************************************************************************************
Region
1    4780
3    2403
4    1182
2    1136
6     805
7     761
9     511
8     434
5     318
Name: count, dtype: int64

****************************************************************************************************
TrafficType
2     3913
1     2451
3     2052
4     1069
13     738
10     450
6      444
8      343
5      260
11     247
20     198
9       42
7       40
15      38
19      17
14      13
18      10
16       3
12       1
17       1
Name: count, dtype: int64

****************************************************************************************************
```

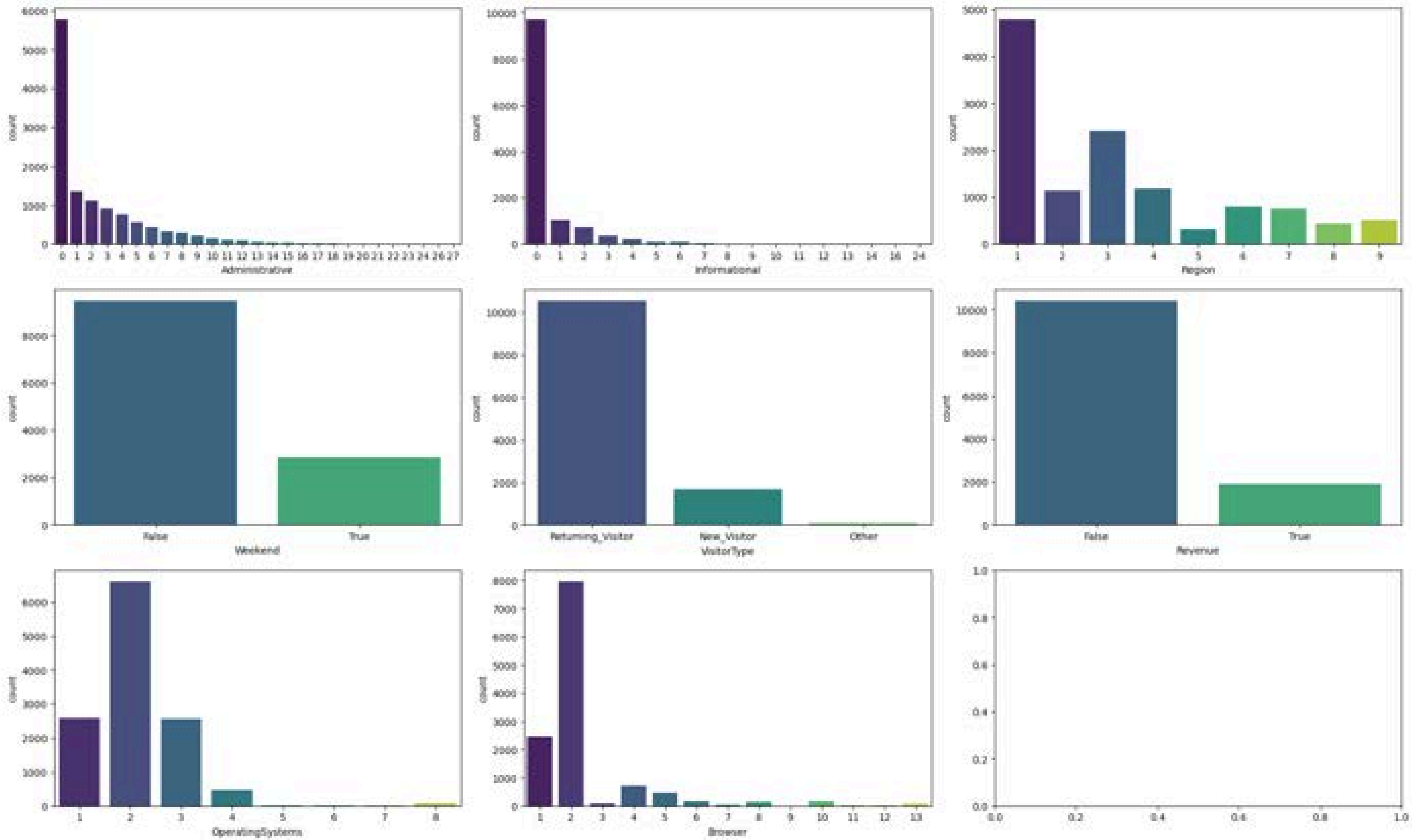
```
Weekend
False    9462
True     2868
Name: count, dtype: int64

****************************************************************************************************
VisitorType
Returning_Visitor    10551
New_Visitor           1694
Other                   85
Name: count, dtype: int64

****************************************************************************************************
Revenue
False    10422
True      1908
Name: count, dtype: int64

****************************************************************************************************
```

Graphical Analysis1

Univariate

In [12]:

```
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 12))
columns = ['Administrative','Informational','Region', 'Weekend', 'VisitorType', 'Revenue', 'OperatingSystems', 'Browser']

for ax, col in zip(axs.flatten(), columns):
    sns.countplot(data=shop, x=col, ax=ax, palette='viridis')

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

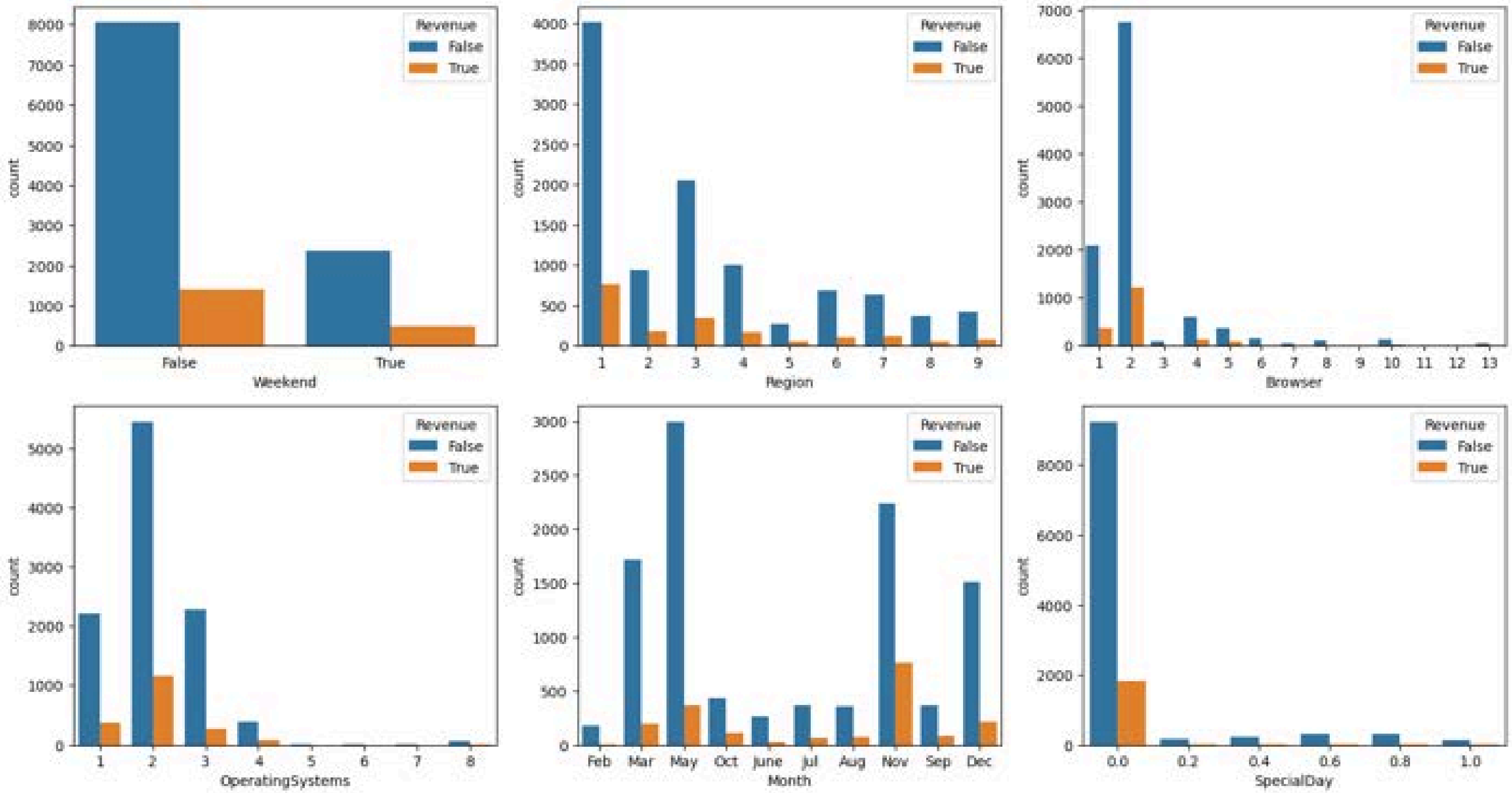


- People are visiting the 0th page more, and the visits gradually decrease towards the last page.
- The same pattern is observed for informal page visits, where the 0th page is visited the most, and it gradually decreases.
- It appears that, in the first region, most customers are from a specific demographic group.
- People are not preferring to shop on weekends.
- Retention is higher than the rate of new customer acquisition.
- Operating systems 1, 2, 3, and 4 are the most commonly used.

Bivarite

In [13]:

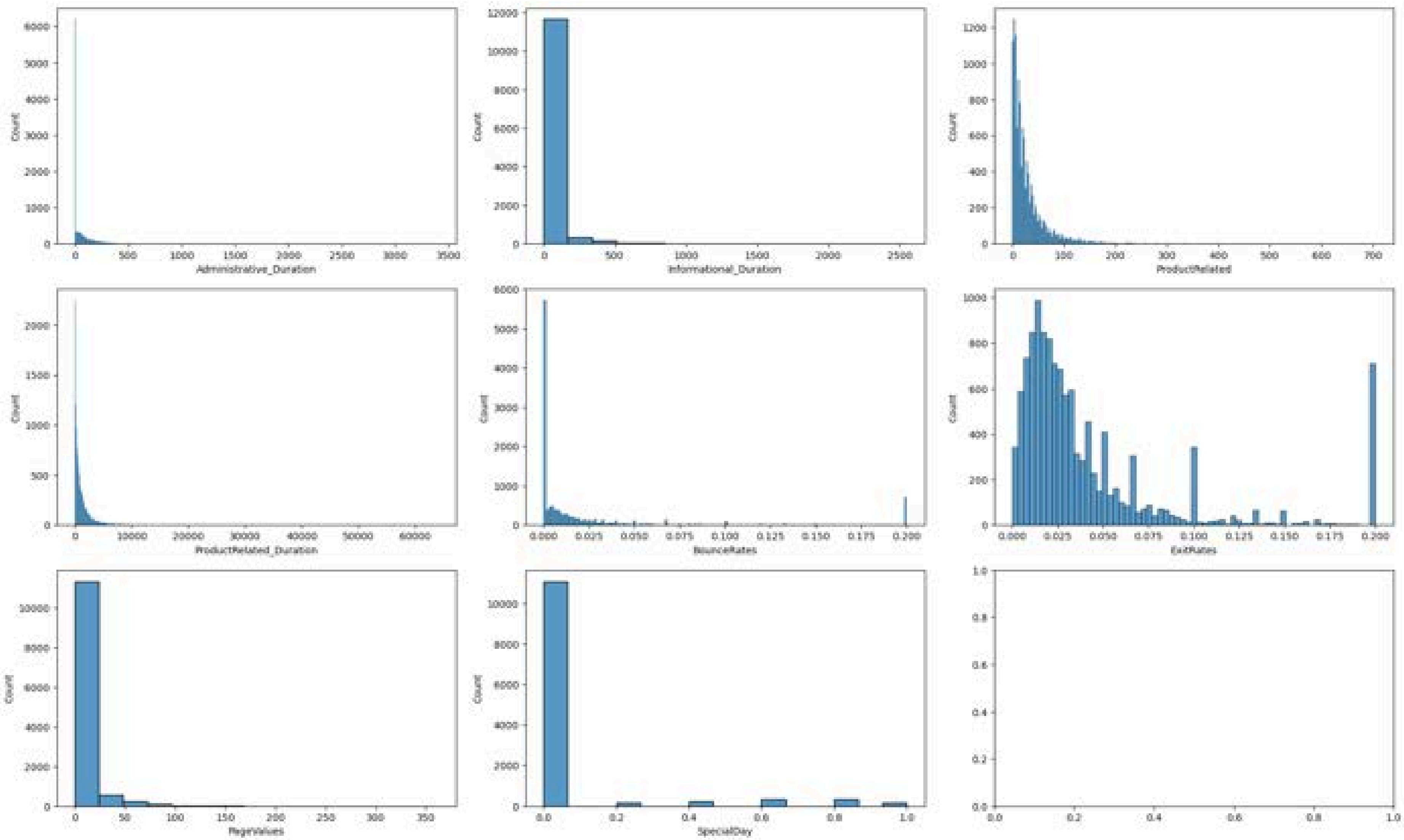
```
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(15, 8))
columns = ['Weekend', 'Region', 'Browser', 'OperatingSystems', 'Month', 'SpecialDay']
positions = [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]

for col, pos in zip(columns, positions):
    sns.countplot(data=shop, x=col, hue='Revenue', ax=axs[pos[0], pos[1]])

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



- There is no significant increase in revenue on weekends.
- Compared to other regions, Region 1 had more customer purchases.
- The second browser appears to have a higher number of users.
- The revenue generated from the second operating system seems to be higher.
- In the current month, the revenue is higher compared to May and December.

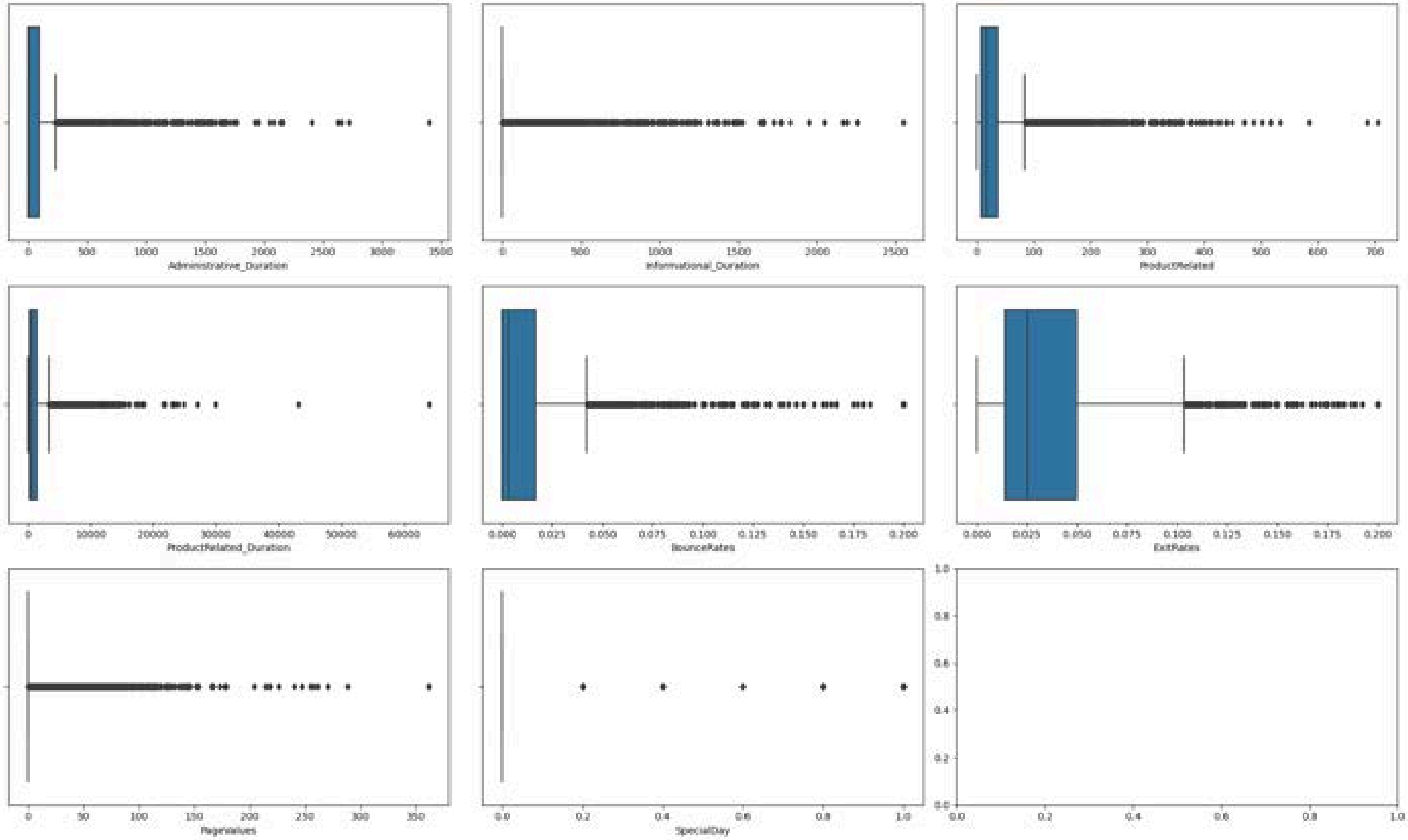
In [15]:
```
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 12))
col_hist=['Administrative_Duration','Informational_Duration','ProductRelated','ProductRelated_Duration',
          'BounceRates','ExitRates','PageValues','SpecialDay']
for ax, col in zip(axs.flatten(), col_hist):
```

```
    sns.histplot(data=shop, x=col, ax=ax)
plt.tight_layout()
plt.show()
```



- Almost all the distributions are right-skewed.

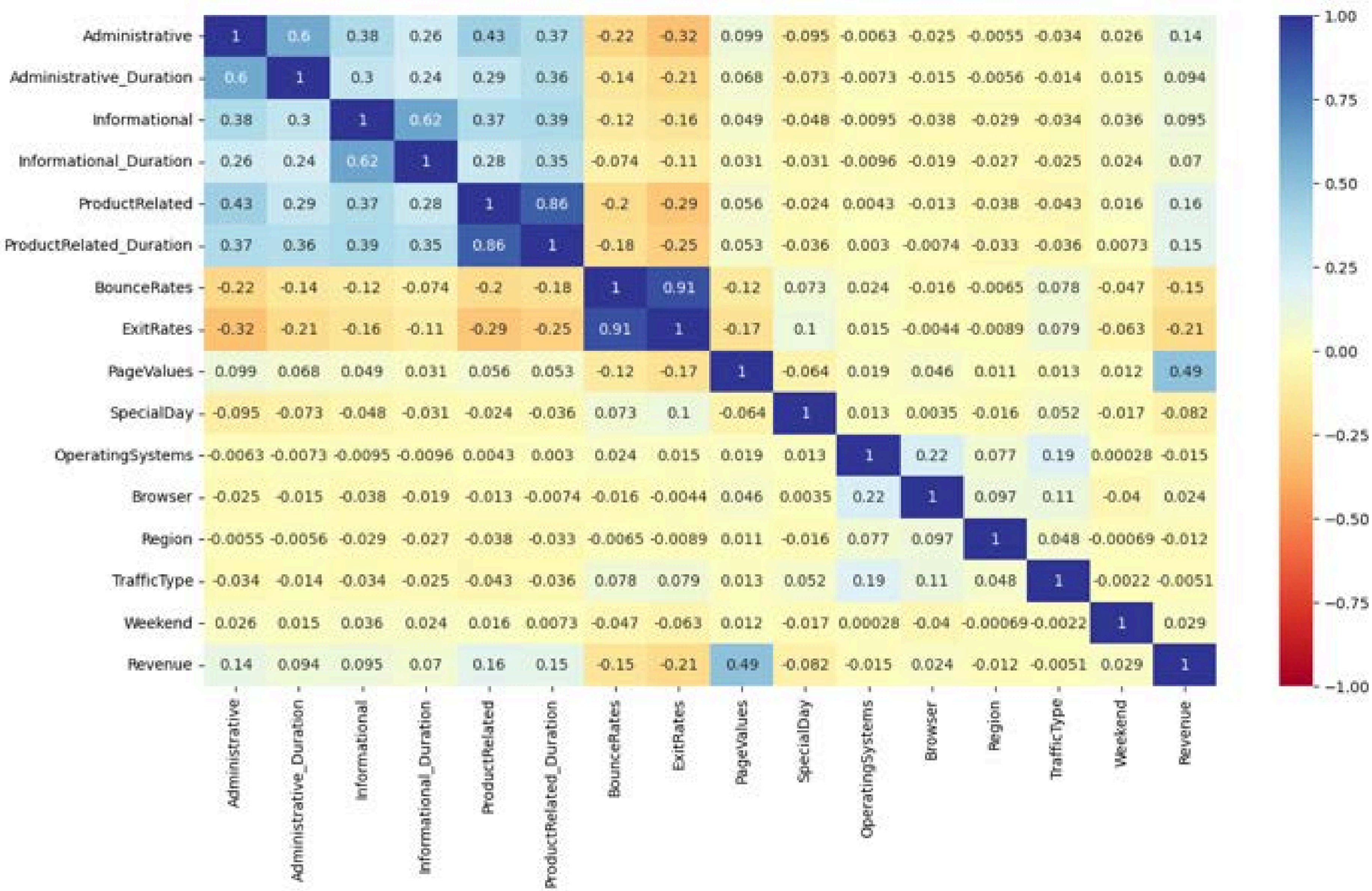
In [16]:
```
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 12))
col_box=['Administrative_Duration','Informational_Duration','ProductRelated','ProductRelated_Duration',
         'BounceRates','ExitRates','PageValues','SpecialDay']
for ax, col in zip(axs.flatten(), col_box):
    sns.boxplot(data=shop, x=col, ax=ax)
plt.tight_layout()
plt.show()
```



Correlation Analysis:

In [17]:
```
corr=shop.corr(numeric_only=True)
```

In [18]:
```
plt.figure(figsize=(15, 8))
sns.heatmap(corr, annot=True, cmap='RdYlBu', vmin=-1, vmax=1)
plt.show()
```

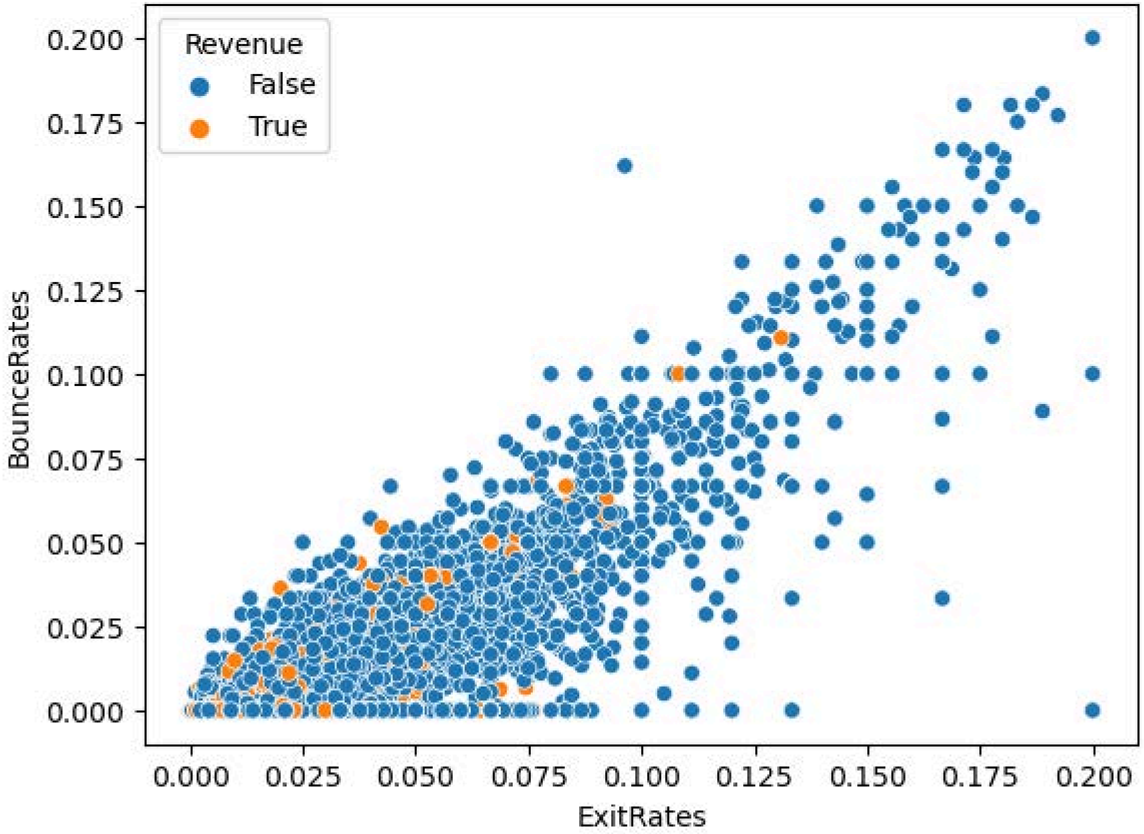


- BounceRates and ExitRates are highly correlated with each other compared to other features.
- The second most highly correlated pair are ProductRelated_Duration and ProductRelated.
- Information and Information_Duration are also 62% correlated with each other.
- PageValue and Revenue are highly correlated with each other.

- Administrative and Product_Related are also 43% correlated.

ExitRates vs BounceRates

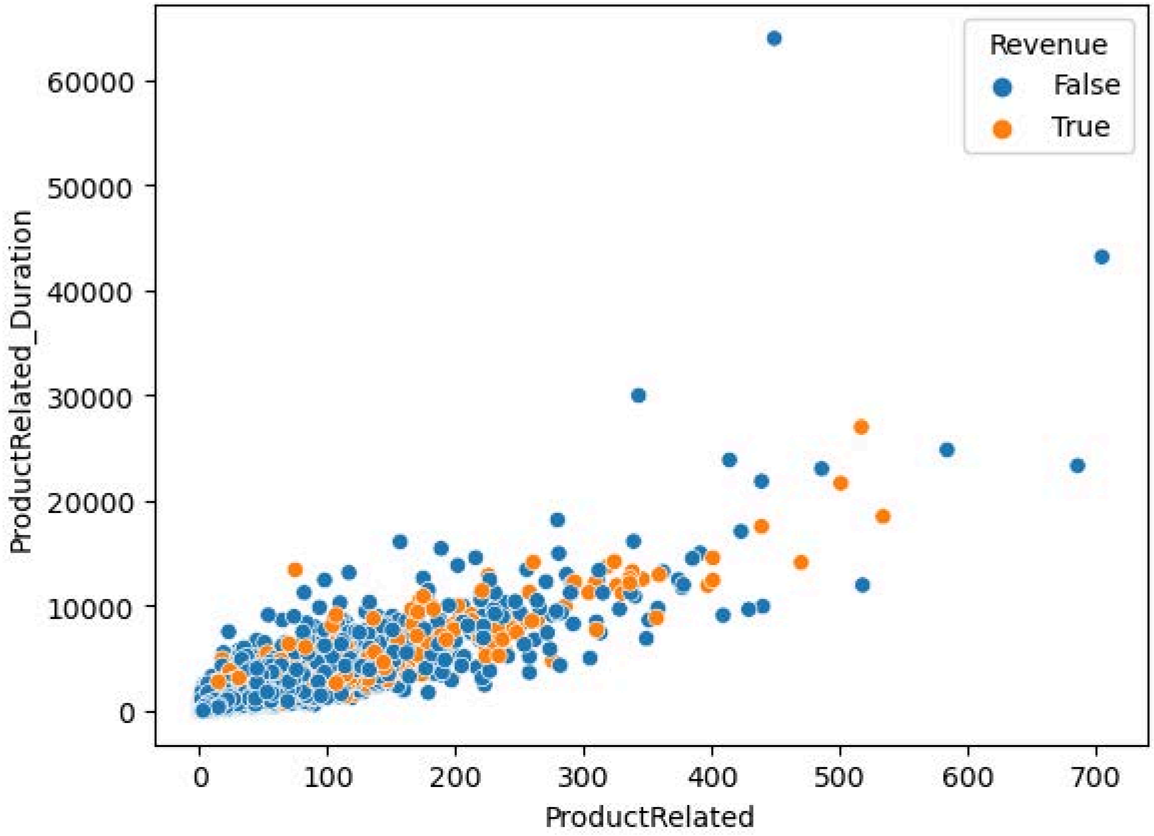
In [19]:
```
sns.scatterplot(data=shop,x='ExitRates',y='BounceRates',hue = 'Revenue')
plt.show()
```



- There is a 91% correlation between the two features

ProductRelated vs ProductRelated_Duration

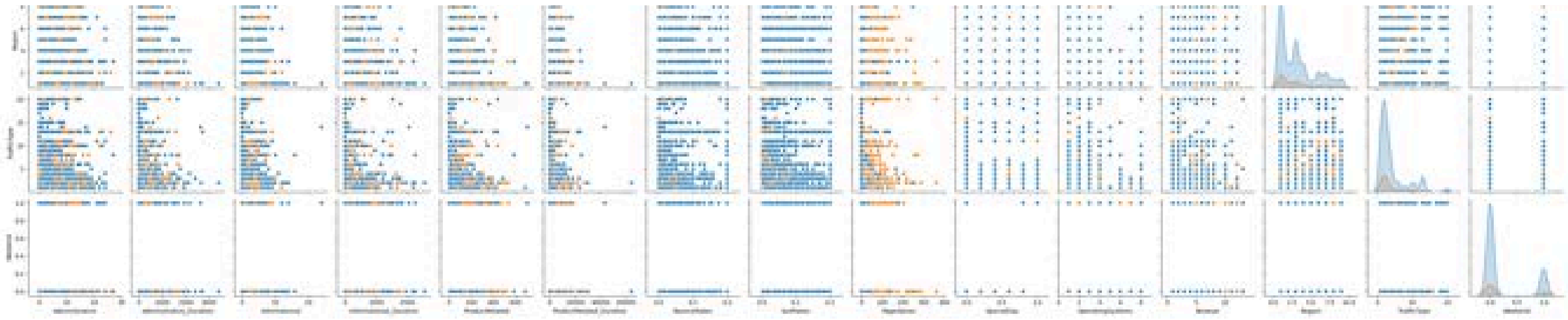
In [20]:
```
sns.scatterplot(data=shop,x='ProductRelated',y='ProductRelated_Duration',hue = 'Revenue')
plt.show()
```



- There is a 86% correlation between the two features

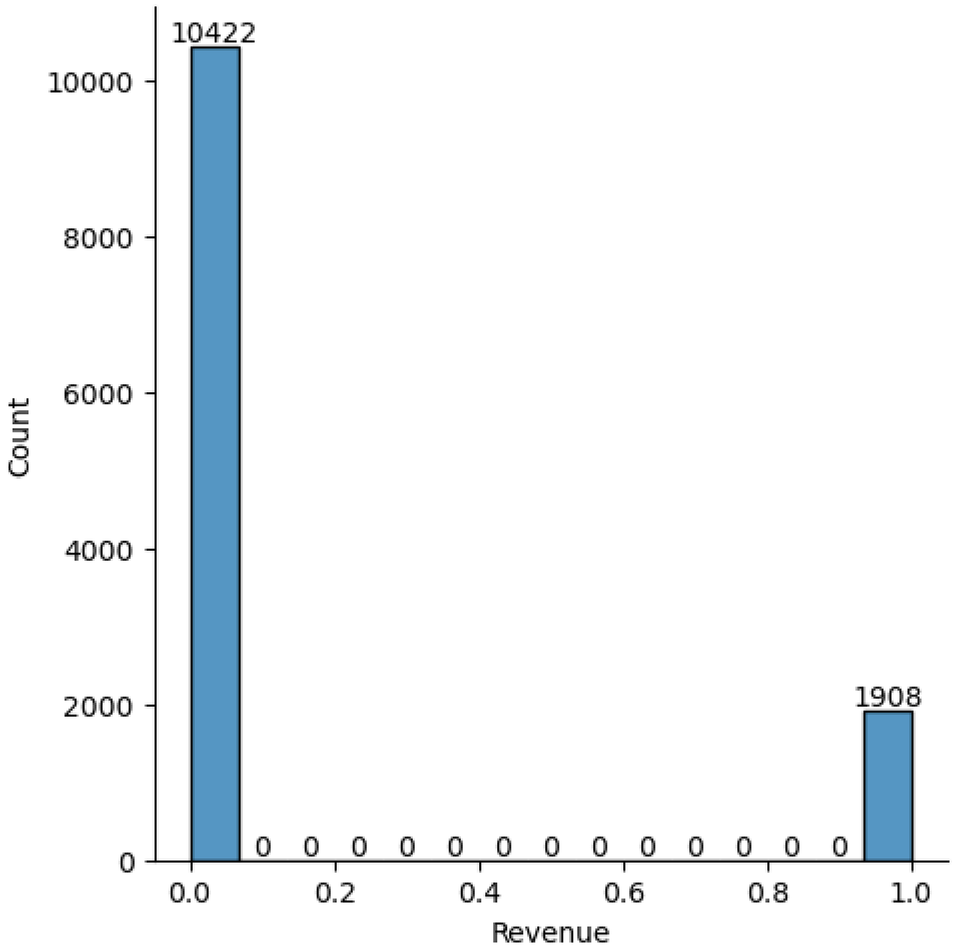
Pair Plot

In [21]:
```
sns.pairplot(shop,hue = 'Revenue')
plt.show()
```



Class Distribution

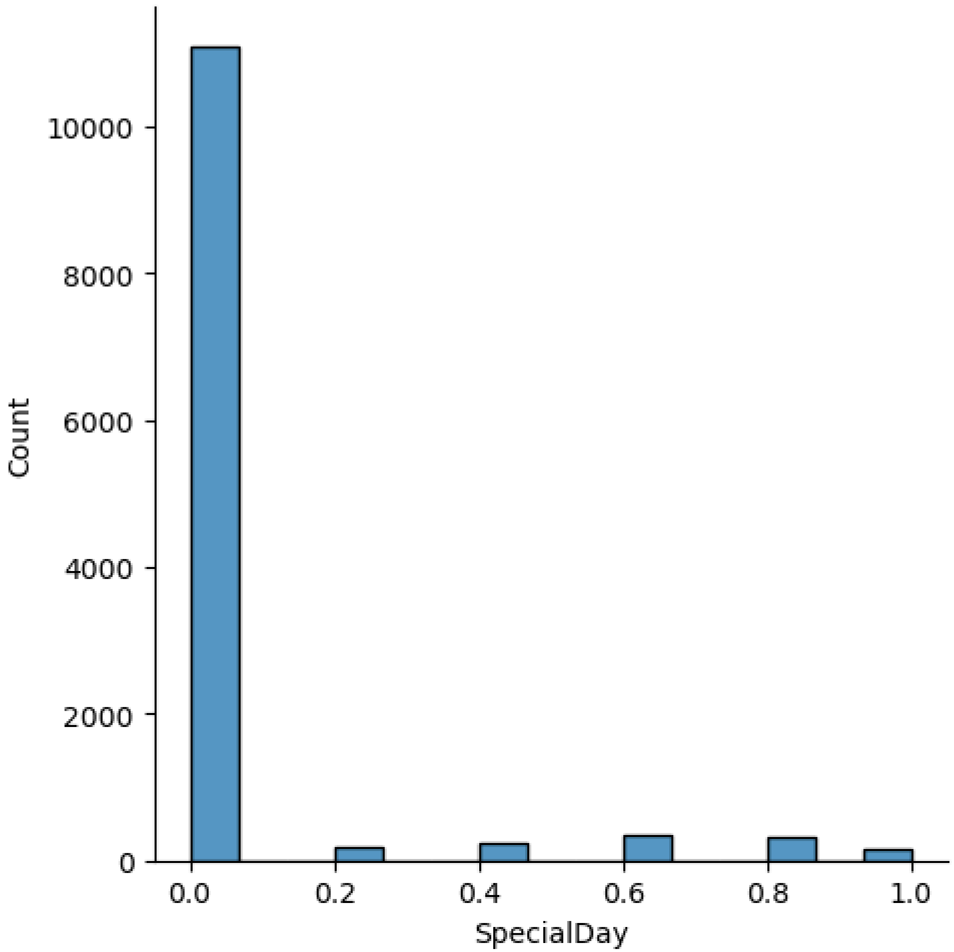
In [22]:
```
sns.displot(data=shop,x='Revenue')
ax=plt.gca()
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



- The distribution is not balanced, with approximately 90% of the values being 0.0 and only 10% being 1.0.

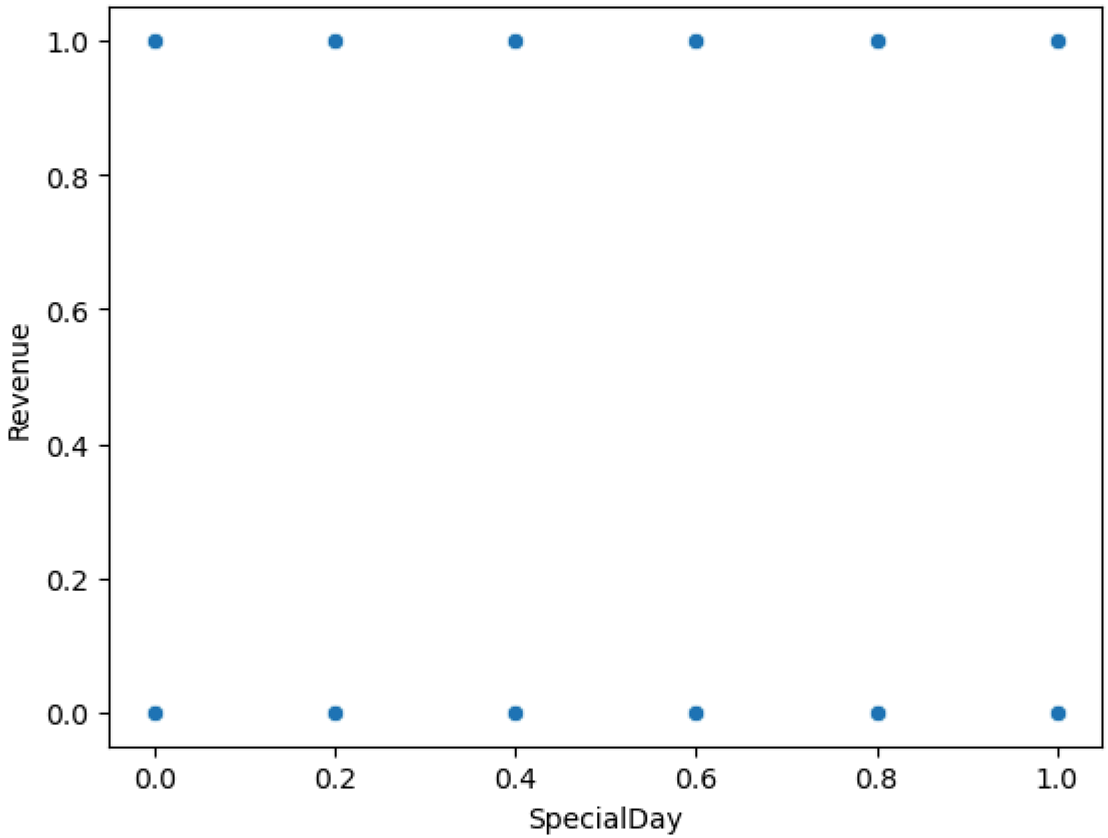
Analyze SpecialDay distribution and its correlation with Revenue

In [23]:
```
sns.displot(data=shop,x='SpecialDay')
plt.show()
```



In [24]:
```python
sns.scatterplot(shop,x='SpecialDay',y='Revenue')
```

Out[24]:
```
<Axes: xlabel='SpecialDay', ylabel='Revenue'>
```



- There is no correlation between Revenue and Special Day.

visited all three page categories.

In [25]:
```
shop['visited_all_3_pages'] = (shop['Administrative'] > 0) & (shop['Informational'] > 0) & (shop['ProductRelated'] > 0)
shop['visited_all_3_pages']
```

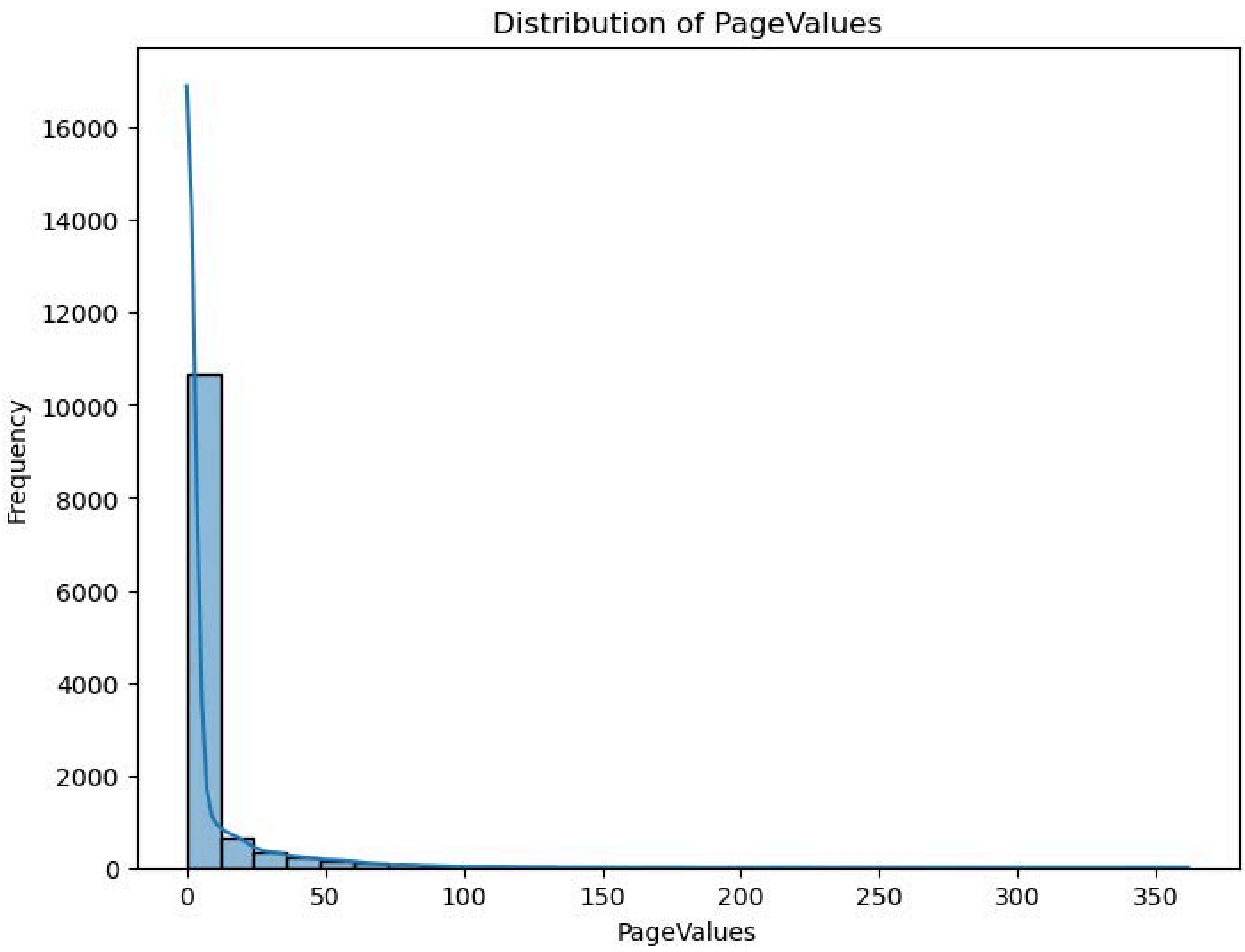
Out[25]:
```
0        False
1        False
2        False
3        False
4        False
         ...
12325    False
12326    False
12327    False
12328    False
12329    False
Name: visited_all_3_pages, Length: 12330, dtype: bool
```

In [26]:
```
shop.visited_all_3_pages.value_counts()
```

Out[26]:
```
visited_all_3_pages
False    10163
True      2167
Name: count, dtype: int64
```

PageValues Vs TrafficType, VisitorType, and Region

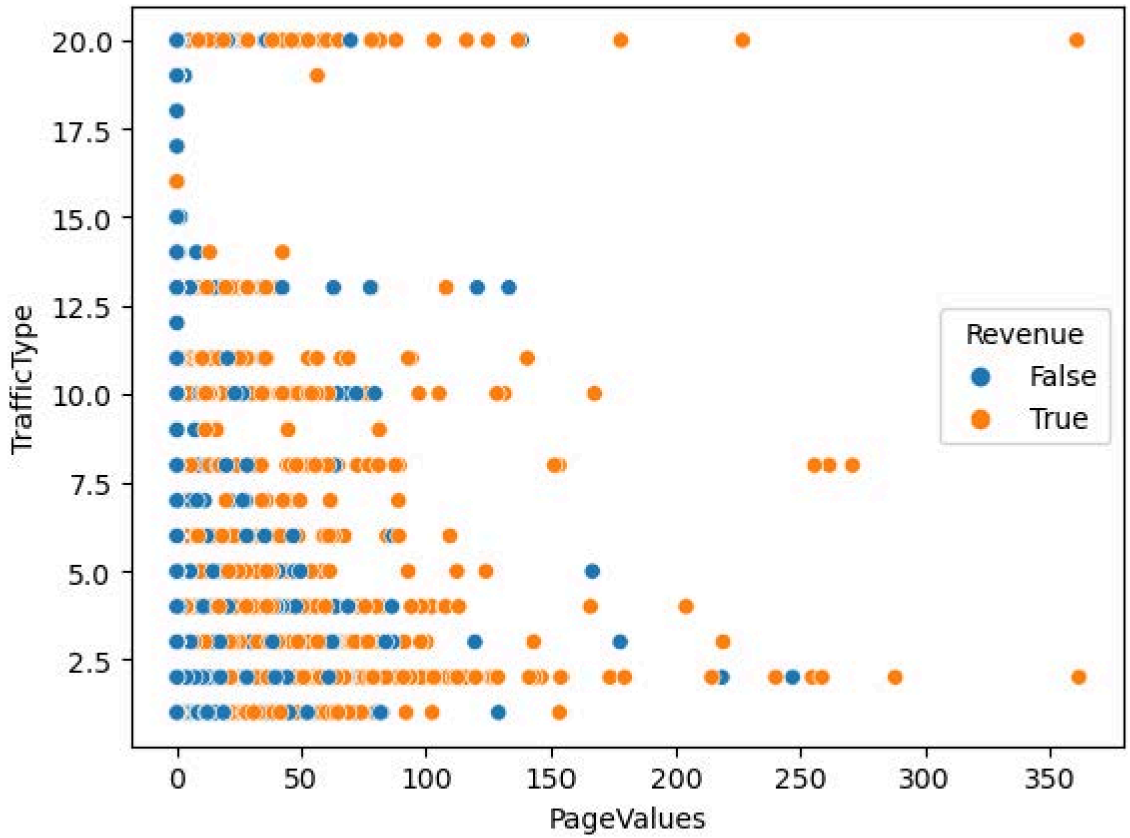
In [27]:
```
plt.figure(figsize=(8, 6))
sns.histplot(shop['PageValues'], bins=30, kde=True)
plt.title('Distribution of PageValues')
plt.xlabel('PageValues')
plt.ylabel('Frequency')
plt.show()
```



- Almost values are between 0 to 25

PageValues Vs TrafficType

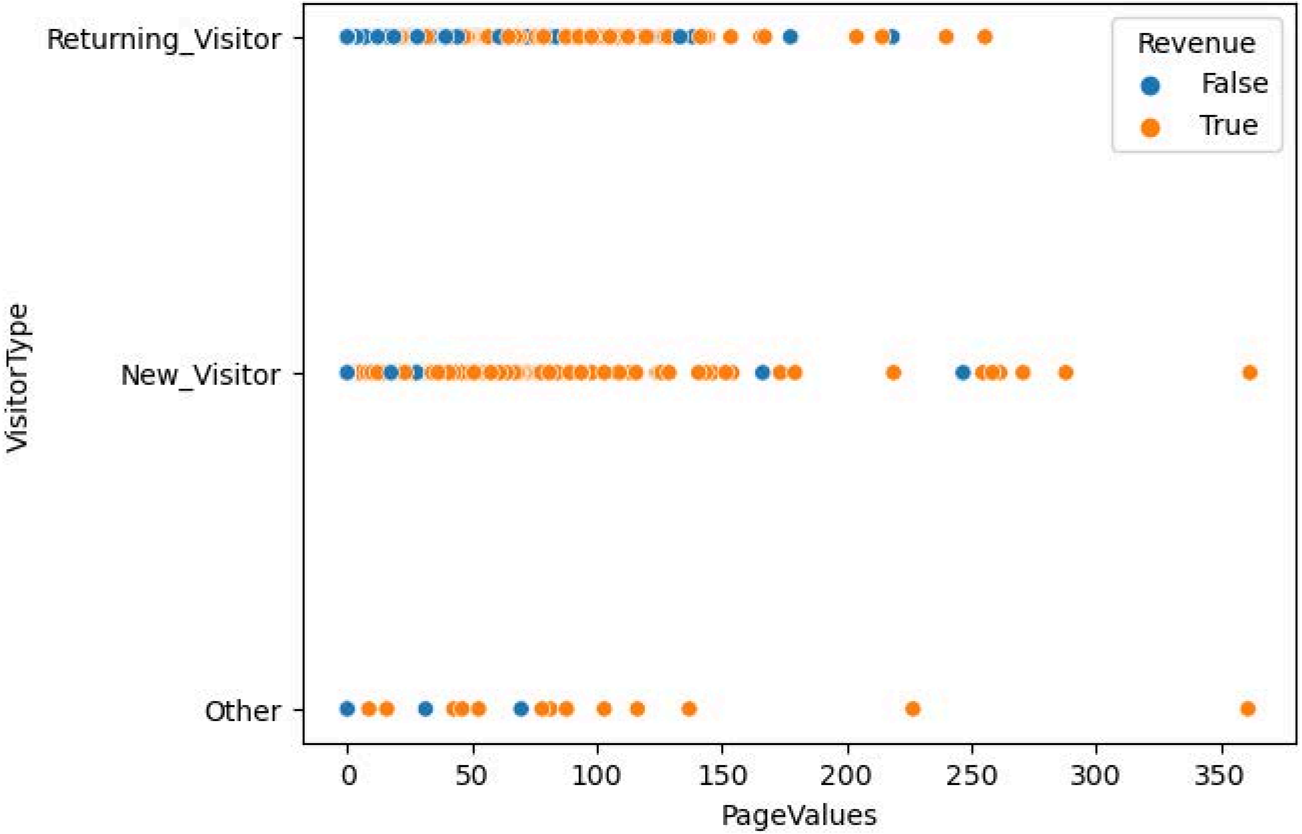
In [28]:
```
sns.scatterplot(data=shop,x='PageValues',y='TrafficType',hue = 'Revenue')
plt.show()
```



PageValues Vs VisitorType

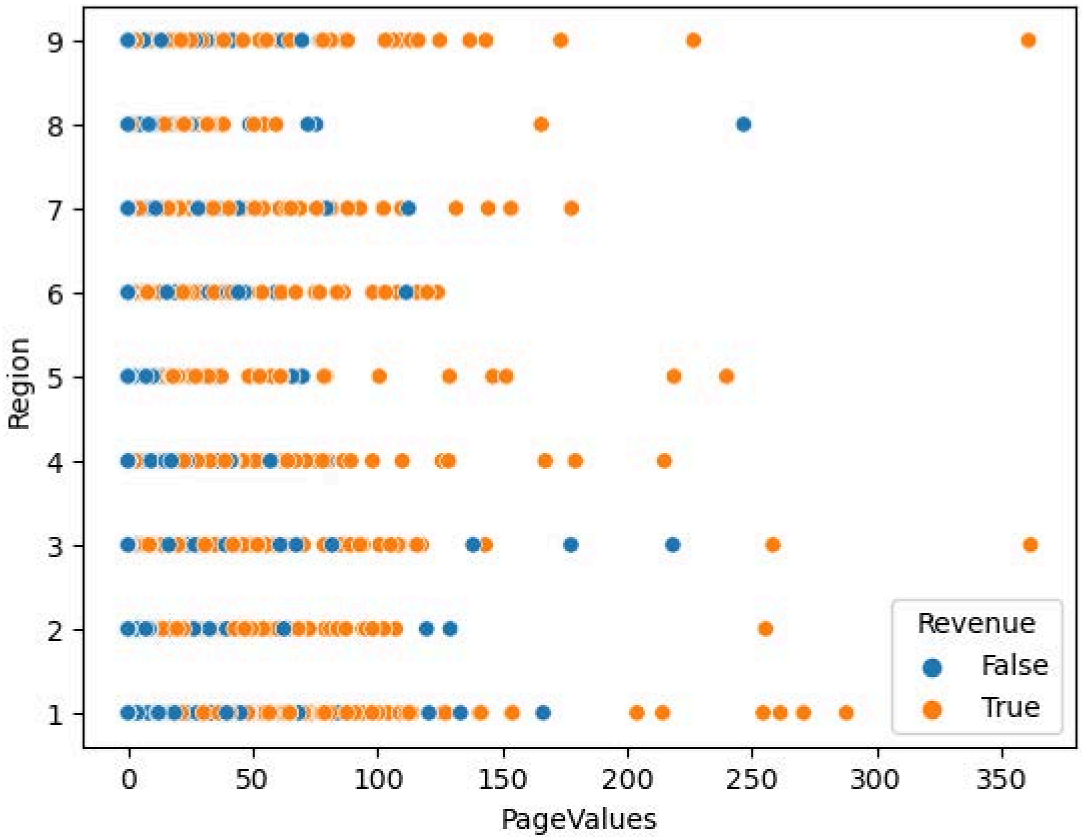
In [29]:

```
sns.scatterplot(data=shop,x='PageValues',y='VisitorType',hue = 'Revenue')
plt.show()
```



PageValues Vs Region

In [30]:
```
sns.scatterplot(data=shop,x='PageValues',y='Region',hue = 'Revenue')
plt.show()
```



In [31]:

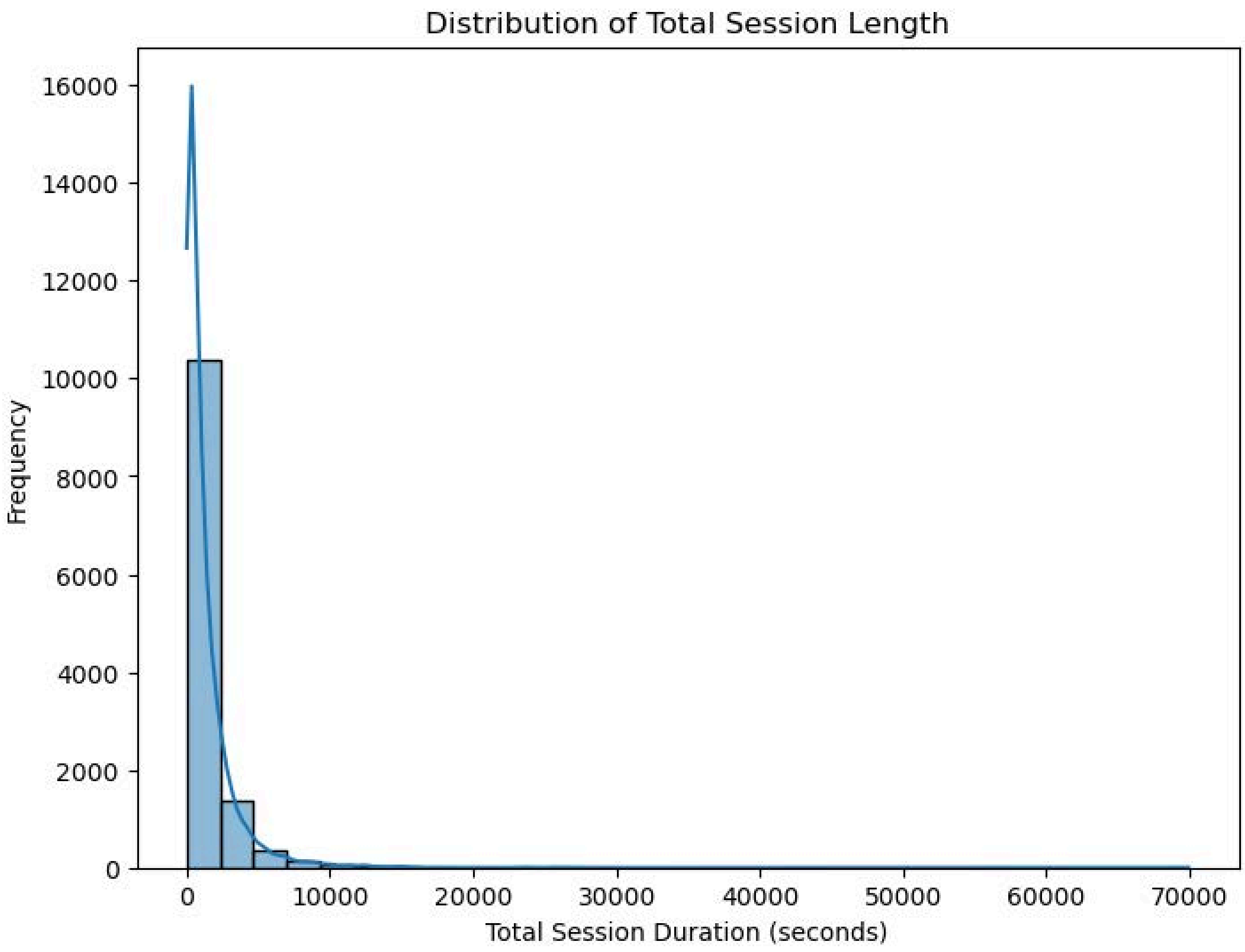
```
grouped = shop.groupby(['TrafficType', 'VisitorType'])['PageValues'].mean().unstack()
plt.figure(figsize=(12, 8))
sns.heatmap(grouped, cmap='YlGnBu', annot=True, fmt=".2f")
plt.title('Average PageValues by TrafficType and VisitorType')
plt.xlabel('VisitorType')
plt.ylabel('TrafficType')
plt.show()
```


Average PageValues by TrafficType and VisitorType
TrafficType
VisitorType
New_Visitor
Other
Returning_Visitor
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
6.56
0.00
3.42
11.74
2.85
7.24
10.28
2.26
2.77
15.54
6.18
8.78
0.00
6.34
4.64
0.00
5.13
12.32
13.75
10.52
0.00
10.14
0.00
4.86
8.20
0.00
6.15
3.54
0.00
5.49
0.00
0.00
0.00
2.38
0.00
5.35
0.00
0.04
0.00
0.00
0.00
0.00
0.00
0.00
0.00
3.72
15.87
28.83
9.51
0
5
10
15
20
25

Investigating user session lengths and their impact on conversion rates

In [32]:
```
#Total session duration
shop['Total_Duration'] = shop['Administrative_Duration'] + shop['Informational_Duration'] + shop['ProductRelated_Duration']
```

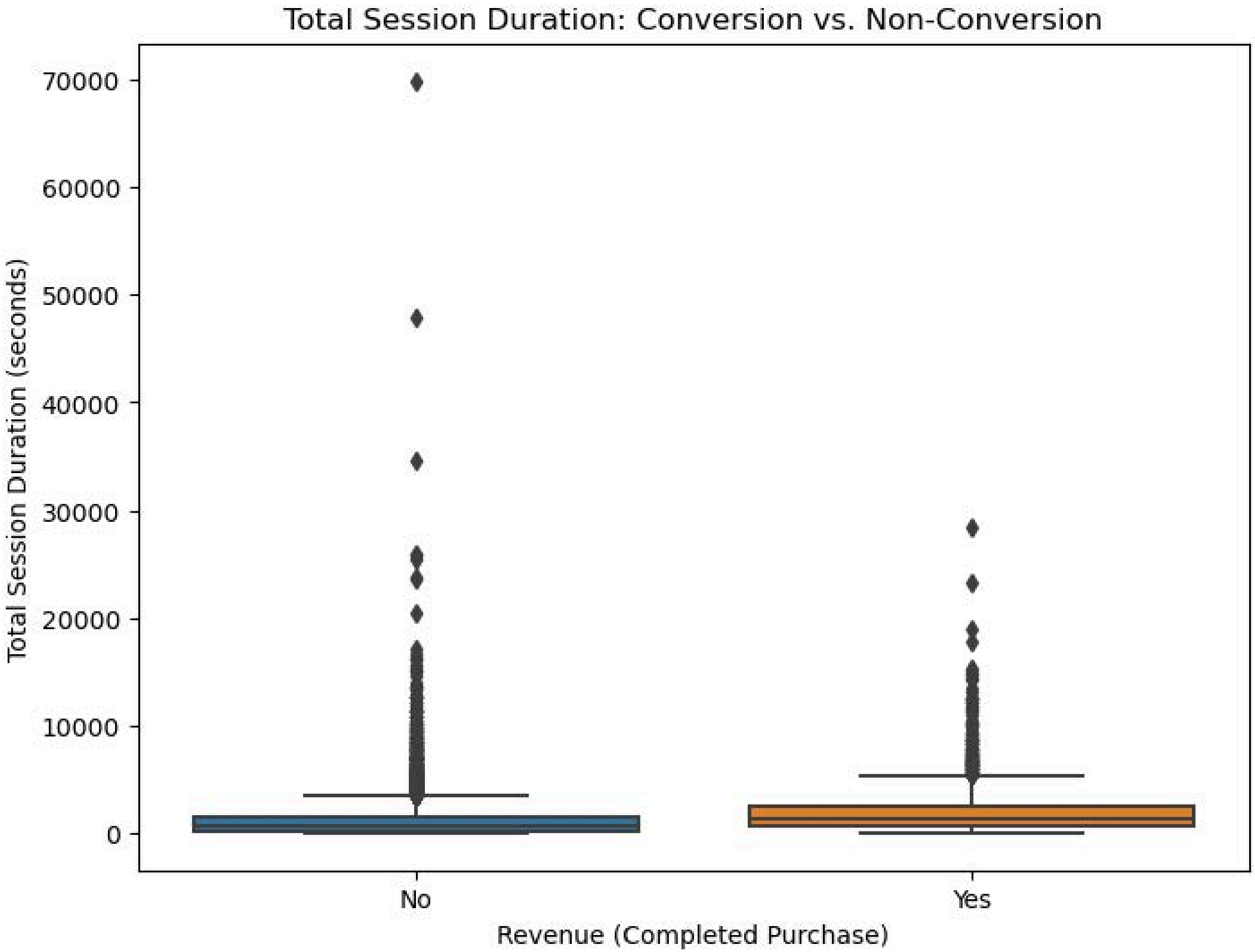
In [33]:
```
plt.figure(figsize=(8, 6))
sns.histplot(shop['Total_Duration'], bins=30, kde=True)
plt.title('Distribution of Total Session Length')
plt.xlabel('Total Session Duration (seconds)')
plt.ylabel('Frequency')
plt.show()
```



Compare Session Who Completed a Purchase vs Those Who Did Not

In [34]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(data=shop, x='Revenue', y='Total_Duration')
plt.title('Total Session Duration: Conversion vs. Non-Conversion')
plt.xlabel('Revenue (Completed Purchase)')
plt.ylabel('Total Session Duration (seconds)')
```

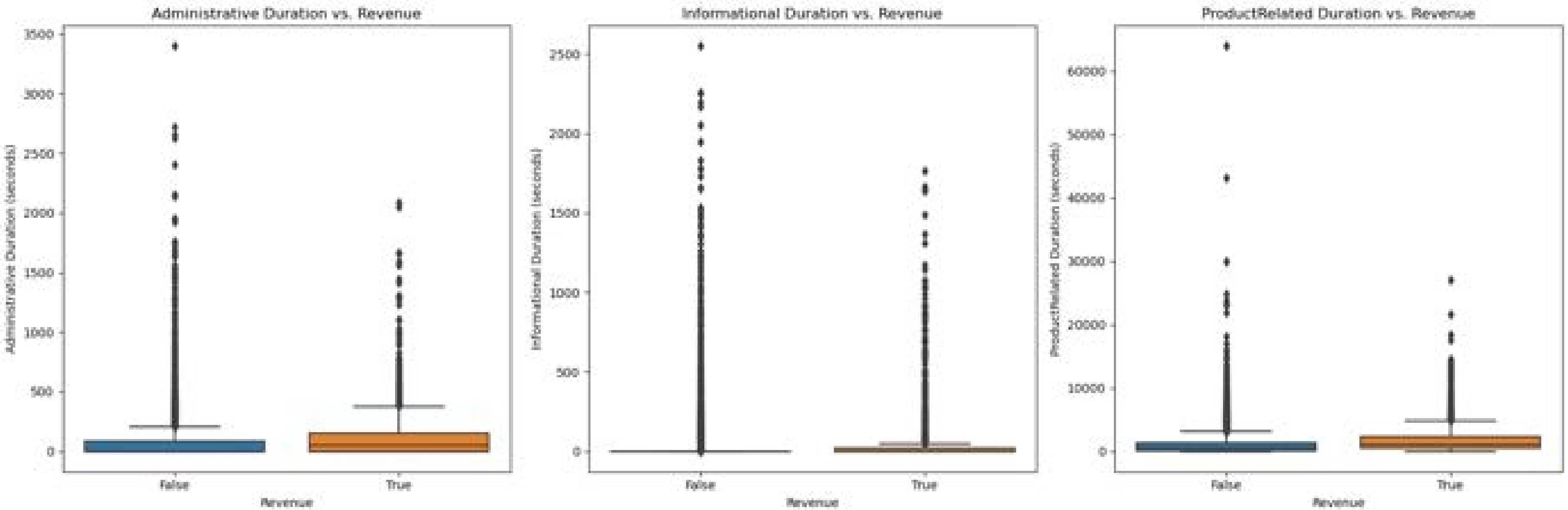
```python
plt.xticks([0, 1], ['No', 'Yes'])
plt.show()
```



- The longer people spend time on the site, the higher the conversion rate tends to be.

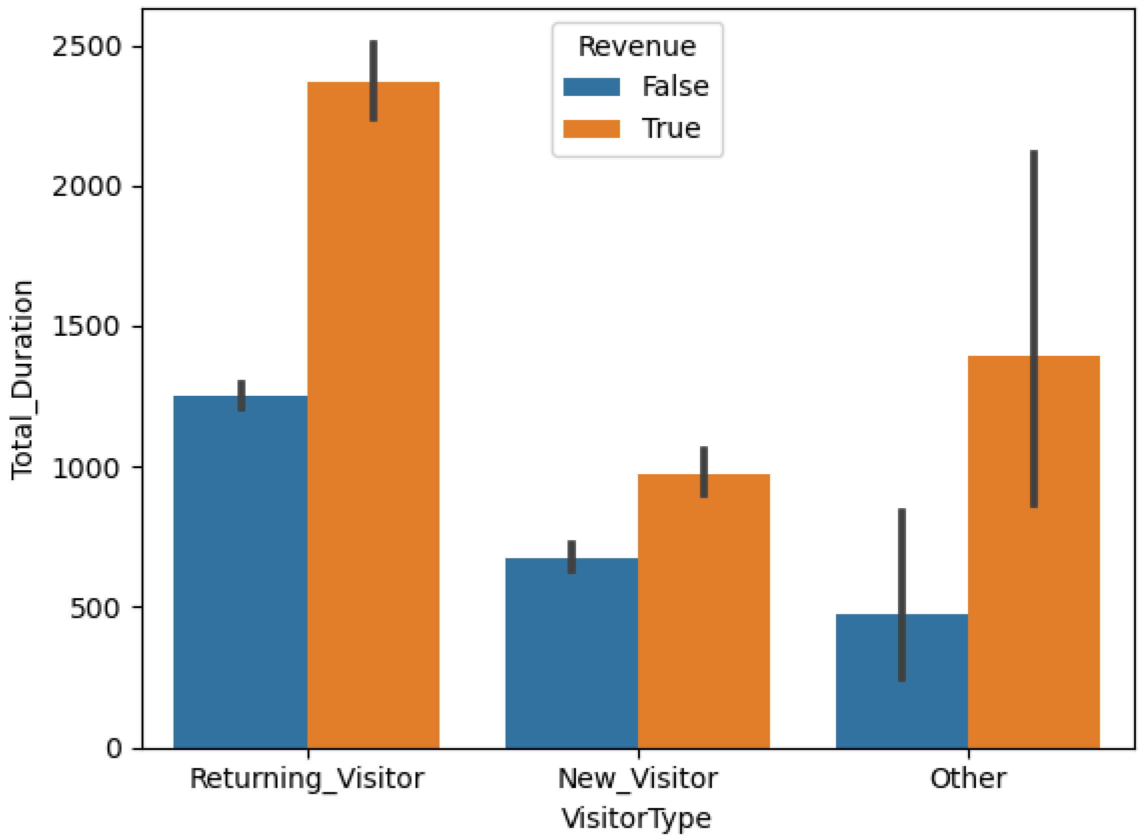
In [35]:
```python
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sns.boxplot(data=shop, x='Revenue', y='Administrative_Duration', ax=axes[0])
```

```
axes[0].set_title('Administrative Duration vs. Revenue')
axes[0].set_xlabel('Revenue')
axes[0].set_ylabel('Administrative Duration (seconds)')

sns.boxplot(data=shop, x='Revenue', y='Informational_Duration', ax=axes[1])
axes[1].set_title('Informational Duration vs. Revenue')
axes[1].set_xlabel('Revenue')
axes[1].set_ylabel('Informational Duration (seconds)')

sns.boxplot(data=shop, x='Revenue', y='ProductRelated_Duration', ax=axes[2])
axes[2].set_title('ProductRelated Duration vs. Revenue')
axes[2].set_xlabel('Revenue')
axes[2].set_ylabel('ProductRelated Duration (seconds)')

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

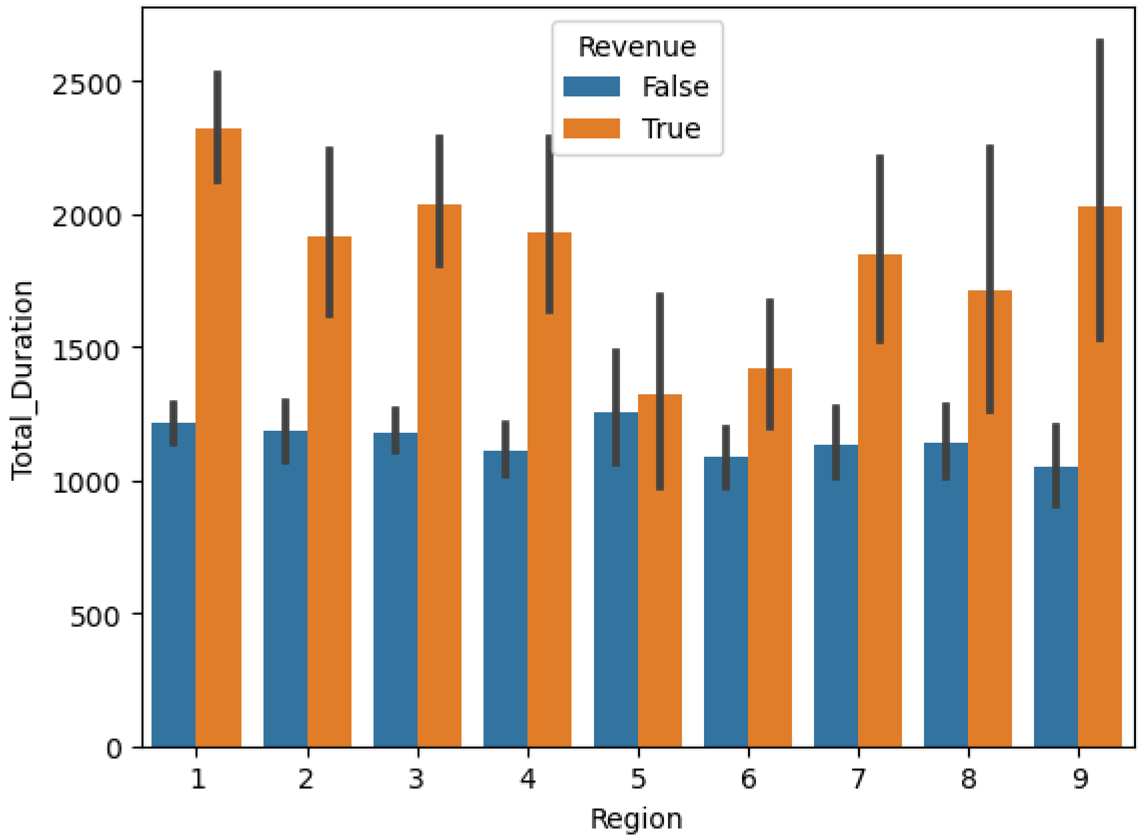


- Here, we can see that the more time customers spend on administrative and product-related pages, the higher the chances of conversion.

Differences in behavior and conversion rates.

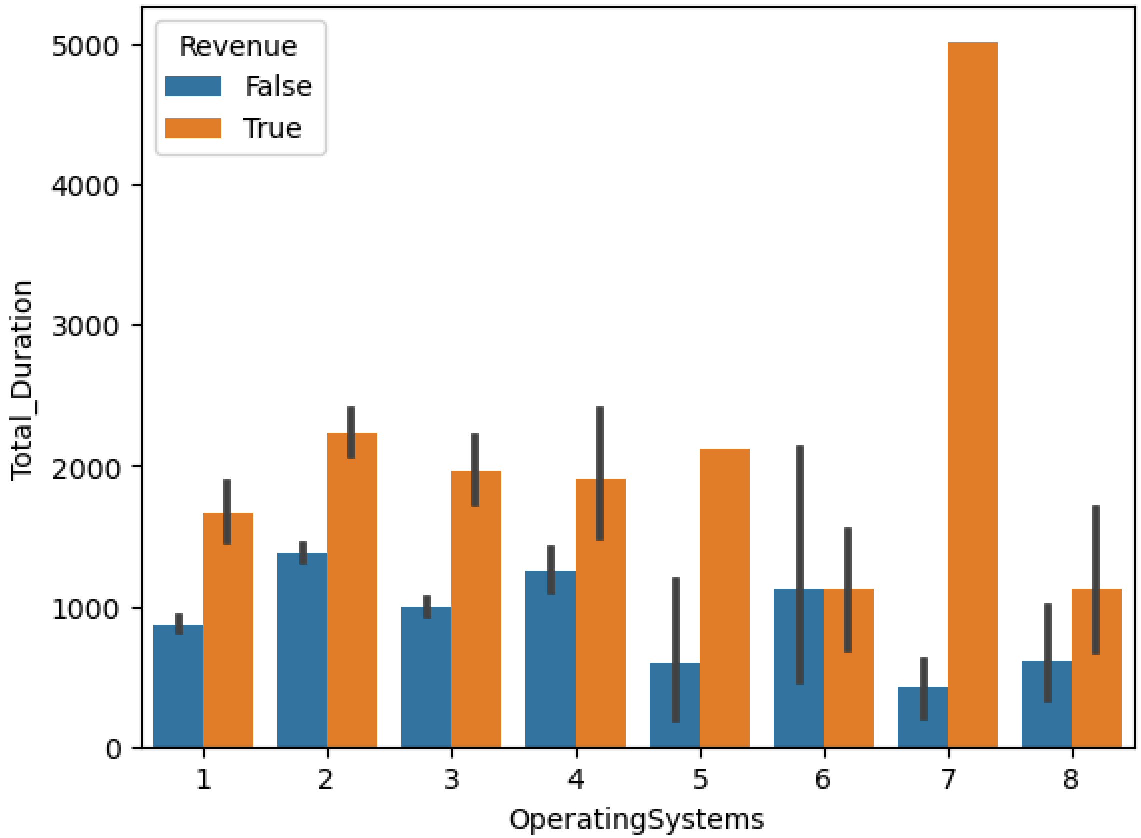
In [36]:
```
sns.barplot(data=shop,x='VisitorType',y='Total_Duration',hue='Revenue')
```
Out[36]:
```
<Axes: xlabel='VisitorType', ylabel='Total_Duration'>
```


In [37]:
```
sns.barplot(data=shop,x='Region',y='Total_Duration',hue='Revenue')
```
Out[37]:
```
<Axes: xlabel='Region', ylabel='Total_Duration'>
```



In [38]:
```
sns.barplot(data=shop,x='OperatingSystems',y='Total_Duration',hue='Revenue')
```

Out[38]:
```
<Axes: xlabel='OperatingSystems', ylabel='Total_Duration'>
```



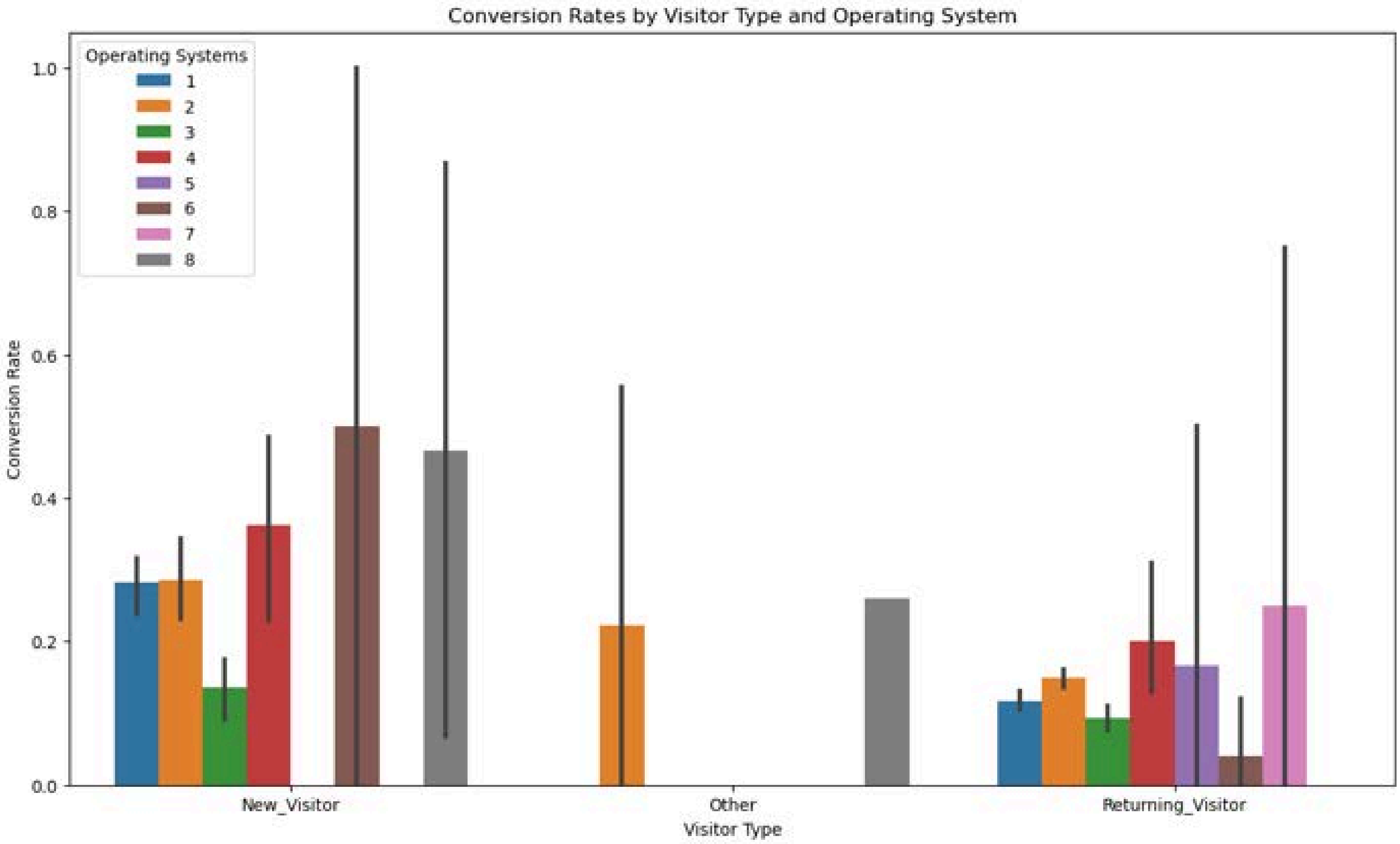
In [39]:

```
grouped_data = shop.groupby(['VisitorType', 'OperatingSystems', 'Region']).agg(
    Total_Sessions=('Revenue', 'count'),
    Conversion_Rate=('Revenue', 'mean'),
    Avg_Admin_Duration=('Administrative_Duration', 'mean'),
    Avg_Info_Duration=('Informational_Duration', 'mean'),
    Avg_Product_Duration=('ProductRelated_Duration', 'mean')
).reset_index()

print(grouped_data)
```

```
            VisitorType  OperatingSystems  Region  Total_Sessions  \
0           New_Visitor                 1       1             173   
1           New_Visitor                 1       2              41   
2           New_Visitor                 1       3              87   
3           New_Visitor                 1       4              37   
4           New_Visitor                 1       5               6   
..                  ...               ...     ...             ...   
110   Returning_Visitor                 8       4               2   
111   Returning_Visitor                 8       5               1   
112   Returning_Visitor                 8       6               1   
113   Returning_Visitor                 8       7               1   
114   Returning_Visitor                 8       9               1   

     Conversion_Rate  Avg_Admin_Duration  Avg_Info_Duration  \
0           0.265896           83.730131           9.577360   
1           0.341463           62.360569          12.685366   
2           0.149425          114.667813          20.024521   
3           0.297297          103.591892           3.378378   
4           0.333333           68.516667           0.000000   
..               ...                 ...                ...   
110         0.000000            0.000000           0.000000   
111         0.000000            0.000000           0.000000   
112         0.000000            0.000000           0.000000   
113         0.000000            0.000000           0.000000   
114         0.000000            0.000000           0.000000   

     Avg_Product_Duration  
0              458.403005  
1              508.990765  
2              486.277128  
3              465.826441  
4              354.758333  
..                    ...  
110            955.750000  
111             87.916667  
112              0.000000  
113             17.000000  
114              0.000000  

[115 rows x 8 columns]
```

In [40]:
```
plt.figure(figsize=(14, 8))
sns.barplot(data=grouped_data, x='VisitorType', y='Conversion_Rate', hue='OperatingSystems')
plt.title('Conversion Rates by Visitor Type and Operating System')
```

```
plt.xlabel('Visitor Type')
plt.ylabel('Conversion Rate')
plt.legend(title='Operating Systems')
plt.show()
```



- New visitors have a higher conversion rate compared to returning customers.
- Users on operating systems 6 and 8 have the highest conversion rates.

- Returning visitors have a nearly identical conversion rate.

Segment users based on TrafficType and analyze their engagement patterns and purchase probability.

In [41]:

```
traffic_analysis = shop.groupby('TrafficType').agg(
    Total_Sessions=('Revenue', 'count'),  # Cnt of sessions
    Conversion_Rate=('Revenue', 'mean'),  # Avg conversion rate
    Total_Admin_Page_Views=('Administrative', 'sum'),  # Total administrative page views
    Total_Info_Page_Views=('Informational', 'sum'),  # Total informational page views
    Total_Product_Page_Views=('ProductRelated', 'sum'),  # Total product-related page views
    Avg_Admin_Duration=('Administrative_Duration', 'mean'),  # Avg duration on administrative pages
    Avg_Info_Duration=('Informational_Duration', 'mean'),  # Avg duration on informational pages
    Avg_Product_Duration=('ProductRelated_Duration', 'mean')  # Avg duration on product-related pages
).reset_index()

# Calculate total page views after aggregation
traffic_analysis['Total_Page_Views'] = (
    traffic_analysis['Total_Admin_Page_Views'] +
    traffic_analysis['Total_Info_Page_Views'] +
    traffic_analysis['Total_Product_Page_Views']
)
traffic_analysis=pd.DataFrame(traffic_analysis)
traffic_analysis
```

Out[41]:

	TrafficType	Total_Sessions	Conversion_Rate	Total_Admin_Page_Views	Total_Info_Page_Views	Total_Product_Page_Views	Avg_Admin_Duration	Avg_Inf
0	1	2451	0.106895	4682	921	78232	65.726273	
1	2	3913	0.216458	11307	2893	149184	105.948043	
2	3	2052	0.087719	3698	616	52953	56.276246	
3	4	1069	0.154350	2511	533	30494	75.264396	
4	5	260	0.215385	906	123	4650	106.858500	
5	6	444	0.119369	903	201	13146	69.740664	
6	7	40	0.300000	116	21	1167	85.339598	
7	8	343	0.276968	987	171	8960	103.662376	
8	9	42	0.095238	82	14	619	67.793651	
9	10	450	0.200000	1051	237	14800	76.523634	
10	11	247	0.190283	435	92	6227	62.513994	
11	12	1	0.000000	0	0	3	0.000000	
12	13	738	0.058266	1317	274	24315	67.844474	
13	14	13	0.153846	44	28	1034	277.147741	
14	15	38	0.000000	52	16	613	70.160088	
15	16	3	0.333333	9	0	46	295.922222	
16	17	1	0.000000	0	0	4	0.000000	
17	18	10	0.000000	21	3	139	72.140000	
18	19	17	0.058824	24	5	674	40.172460	
19	20	198	0.252525	401	61	3989	79.874198	

Insights:

- Page Visits: Initial pages (0th page) receive more visits, with a gradual decrease toward the last page. This pattern holds for both formal and informal page visits.
- Shopping Behavior: Customers prefer not to shop on weekends, as no significant increase in revenue is observed.
- Customer Retention: Retention rates are higher than new customer acquisition rates.
- Geographical Trends: Region 1 shows a higher volume of purchases compared to other regions.
- Browser & OS Usage: The second browser and the second operating system have higher user counts and revenue, respectively.
- Monthly Revenue: Revenue is higher in the current month compared to May and December. #### Feature Correlation:
- BounceRates and ExitRates are strongly correlated.
- ProductRelated_Duration and ProductRelated are the second most correlated.
- Information and Information_Duration show a moderate correlation (62%).
- PageValue and Revenue have a high correlation.
- Administrative and Product_Related have a moderate correlation (43%).
- No correlation observed between Revenue and Special Day.

Recommendations:

- Enhance Page Navigation: Focus on improving engagement on later pages to reduce drop-offs. Consider adding recommendations or incentives on these pages.
- Weekend Promotions: Introduce targeted promotions on weekends to boost sales, as customers currently show lower activity during this period.
- Retention Strategies: Continue to prioritize retention programs, as retaining existing customers has proven more effective than acquiring new ones.
- Geographic Focus: Strengthen marketing efforts in Region 1 and replicate successful strategies in other regions.
- Browser & OS Compatibility: Optimize user experience on the second browser and operating system, ensuring smooth performance to capitalize on high usage.

Campaign Analysis

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import f_oneway,kruskal
```

Import Data set

In [2]:
```
camp=pd.read_csv("campaign.csv")
camp.head()
```

Out[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumCatalogPurchases	NumStor
0	1826	1970	Graduation	Divorced	$84,835.00	0	0	6/16/14	0	189	...	4	
1	1	1961	Graduation	Single	$57,091.00	0	0	6/15/14	0	464	...	3	
2	10476	1958	Graduation	Married	$67,267.00	0	1	5/13/14	0	134	...	2	
3	1386	1967	Graduation	Together	$32,474.00	1	1	5/11/14	0	10	...	0	
4	5371	1989	Graduation	Single	$21,474.00	1	0	4/8/14	0	6	...	1	

5 rows × 27 columns

Basic Metric

In [3]:
```
camp.shape
```

Out[3]:
```
(2239, 27)
```

In [4]:
```
camp.size
```

Out[4]: 60453

In [5]:
```
camp.ndim
```

Out[5]: 2

In [6]:
```
camp.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
```

In [7]:
```
camp.dtypes
```

Out[7]:
```
ID                      int64
Year_Birth              int64
Education              object
Marital_Status         object
Income                 object
Kidhome                 int64
Teenhome                int64
Dt_Customer            object
Recency                 int64
MntWines                int64
MntFruits               int64
MntMeatProducts         int64
MntFishProducts         int64
MntSweetProducts        int64
MntGoldProds            int64
NumDealsPurchases       int64
NumWebPurchases         int64
NumCatalogPurchases     int64
NumStorePurchases       int64
NumWebVisitsMonth       int64
AcceptedCmp3            int64
AcceptedCmp4            int64
AcceptedCmp5            int64
AcceptedCmp1            int64
AcceptedCmp2            int64
Complain                int64
Country                object
dtype: object
```

In [8]:
```
camp.describe()
```

Out[8]:

	ID	Year_Birth	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProdu
count	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000
mean	5590.444841	1968.802144	0.443948	0.506476	49.121036	304.067441	26.307727	167.016525	37.538633	27.0745
std	3246.372471	11.985494	0.538390	0.544555	28.963662	336.614830	39.781468	225.743829	54.637617	41.2860
min	0.000000	1893.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	2827.500000	1959.000000	0.000000	0.000000	24.000000	24.000000	1.000000	16.000000	3.000000	1.0000
50%	5455.000000	1970.000000	0.000000	0.000000	49.000000	174.000000	8.000000	67.000000	12.000000	8.0000
75%	8423.500000	1977.000000	1.000000	1.000000	74.000000	504.500000	33.000000	232.000000	50.000000	33.0000
max	11191.000000	1996.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	263.0000

8 rows × 22 columns

Feature engineering

In [9]:

```
# # SPliting income
camp['Income'] = camp['Income'].astype(str).str.replace('$', '', regex=False).str.replace(',', '', regex=False)
camp['Income'] = camp['Income'].astype(float)
```

Checking for NULL

In [10]:

```
camp.isna().sum()
```

Out[10]:
```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                 24
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
```

In [11]:
```
camp['Income']
```

Out[11]:
```
0       84835.0
1       57091.0
2       67267.0
3       32474.0
4       21474.0
          ...
2234    66476.0
2235    31056.0
2236    46310.0
2237    65819.0
2238    94871.0
Name: Income, Length: 2239, dtype: float64
```

Checking for Duplicate

In [12]:
```
camp.duplicated().value_counts()
```

Out[12]:
```
False    2239
Name: count, dtype: int64
```

- There is no Duplicates

Non Graphical Analysis

In [13]:
```
col=["Education","Marital_Status","Kidhome","NumDealsPurchases","NumWebPurchases",
     "NumWebVisitsMonth","NumCatalogPurchases","AcceptedCmp1","AcceptedCmp3",
     "AcceptedCmp3","AcceptedCmp4","AcceptedCmp5","Complain","Country"]

for i in col:
    print(camp.value_counts(i))
    print()
    print("*"*100)
```

```
Education
Graduation    1126
PhD            486
Master         370
2n Cycle       203
Basic           54
Name: count, dtype: int64

****************************************************************************************************
Marital_Status
Married     864
Together    579
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: count, dtype: int64

****************************************************************************************************
Kidhome
0    1293
1     898
2      48
Name: count, dtype: int64

****************************************************************************************************
NumDealsPurchases
1     970
2     497
3     297
4     188
5      94
6      61
0      46
7      40
8      14
9       8
15      7
10      5
11      5
12      4
13      3
```

```
Name: count, dtype: int64

****************************************************************************************************
NumWebPurchases
2     373
1     354
3     335
4     280
5     220
6     205
7     155
8     102
9      75
0      49
11     44
10     43
27      2
23      1
25      1
Name: count, dtype: int64

****************************************************************************************************
NumWebVisitsMonth
7     393
8     342
6     339
5     281
4     218
3     205
2     202
1     153
9      83
0      11
10      3
20      3
14      2
19      2
13      1
17      1
Name: count, dtype: int64

****************************************************************************************************
NumCatalogPurchases
0     586
```

```
1     496
2     276
3     184
4     182
5     140
6     128
7      79
8      55
10     48
9      42
11     19
28      3
22      1
Name: count, dtype: int64

****************************************************************************************************
AcceptedCmp1
0    2095
1     144
Name: count, dtype: int64

****************************************************************************************************
AcceptedCmp3
0    2076
1     163
Name: count, dtype: int64

****************************************************************************************************
AcceptedCmp3
0    2076
1     163
Name: count, dtype: int64

****************************************************************************************************
AcceptedCmp4
0    2072
1     167
Name: count, dtype: int64

****************************************************************************************************
AcceptedCmp5
0    2076
1     163
Name: count, dtype: int64
```

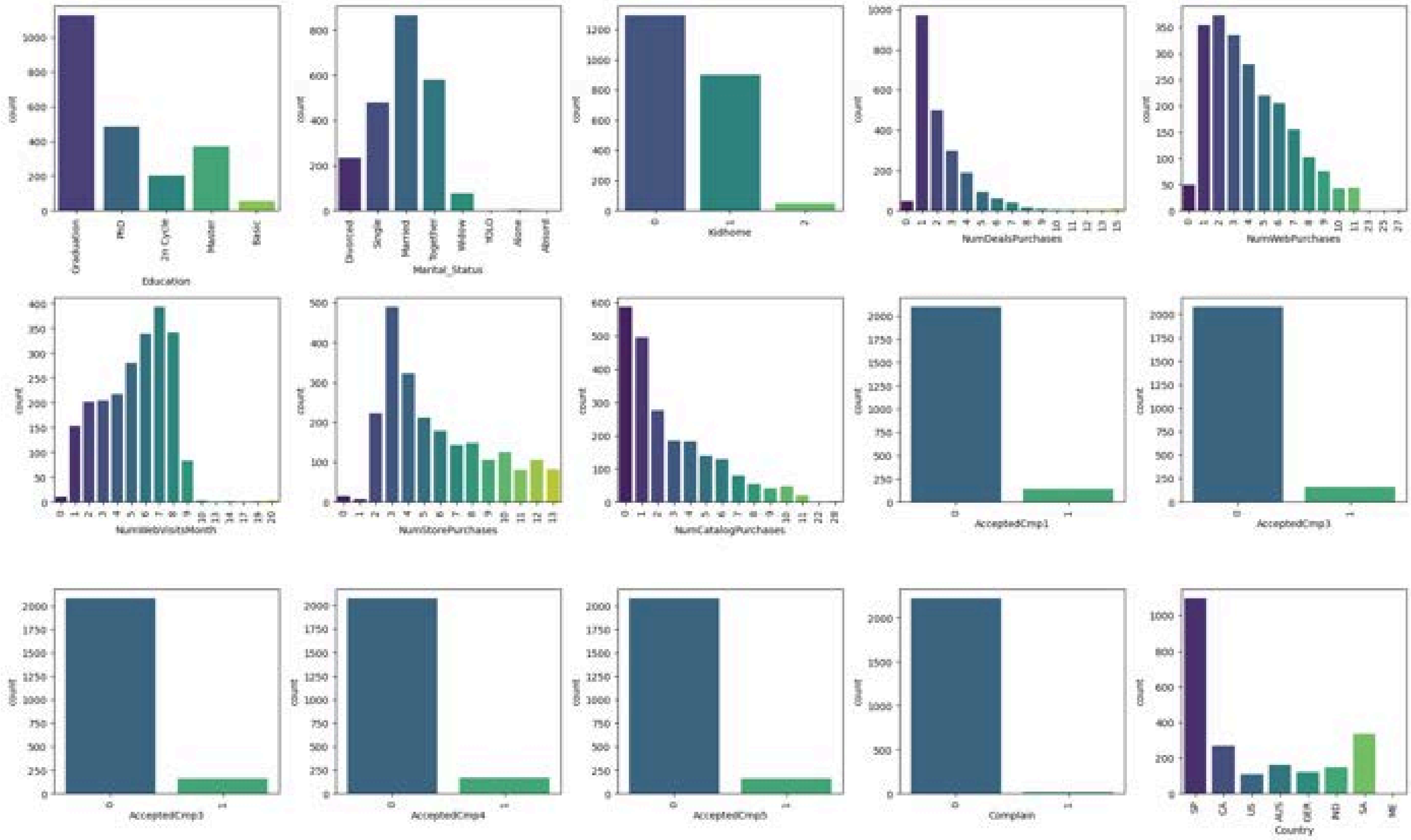
```
****************************************************************************************************
Complain
0    2218
1      21
Name: count, dtype: int64

****************************************************************************************************
Country
SP     1095
SA      336
CA      268
AUS     160
IND     148
GER     120
US      109
ME        3
Name: count, dtype: int64

****************************************************************************************************
```

Graphical Analysis

Univariate

In [14]:

```python
fig, axs = plt.subplots(nrows=3, ncols=5, figsize=(20, 12))

columns = ["Education", "Marital_Status", "Kidhome", "NumDealsPurchases", "NumWebPurchases",
           "NumWebVisitsMonth",'NumStorePurchases', "NumCatalogPurchases", "AcceptedCmp1", "AcceptedCmp3",
           "AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "Complain", "Country"]

for ax, col in zip(axs.flatten(), columns):
    sns.countplot(data=camp, x=col, ax=ax, palette='viridis')
    ax.tick_params(axis='x', rotation=90)  # Rotate x-axis labels

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

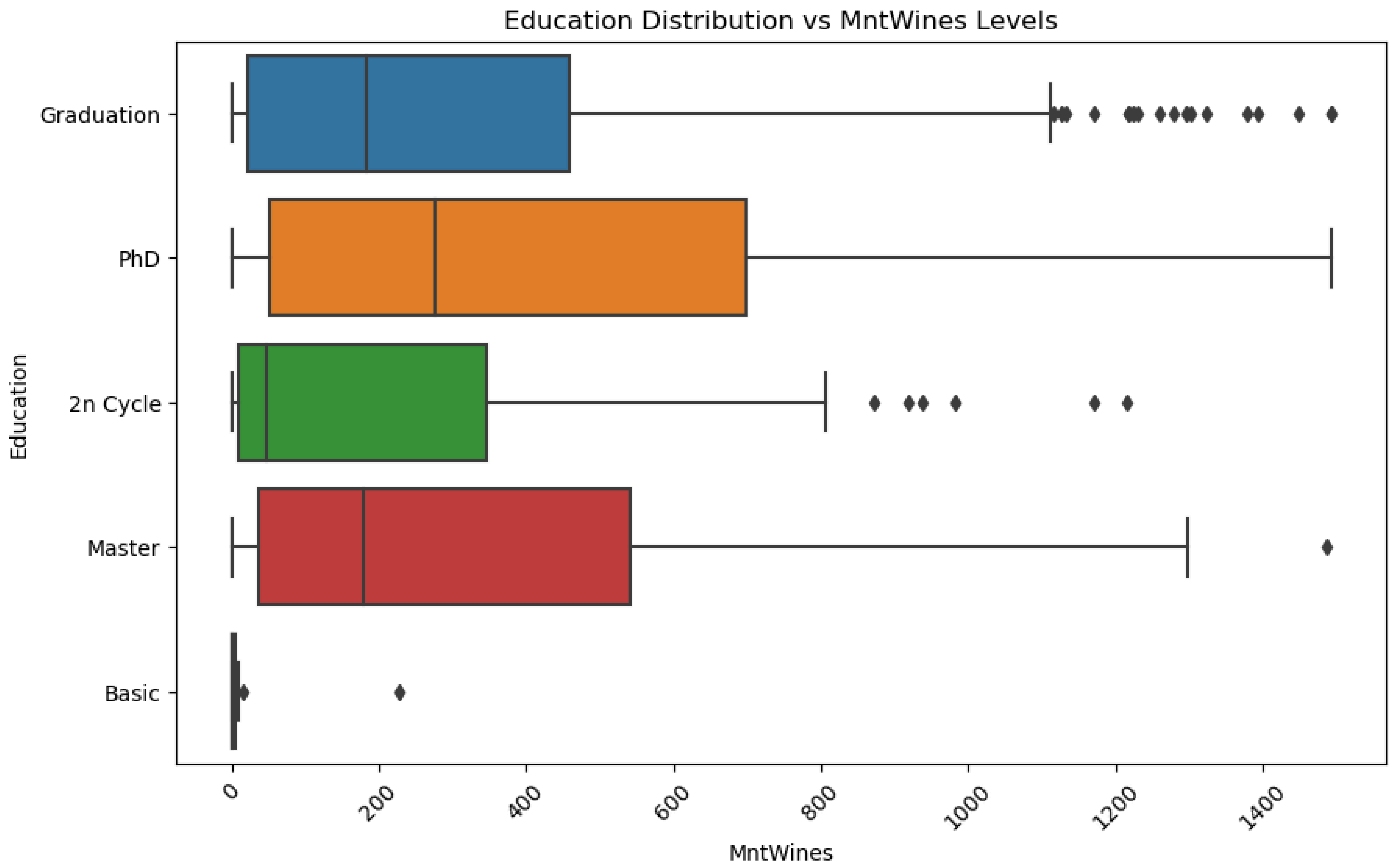


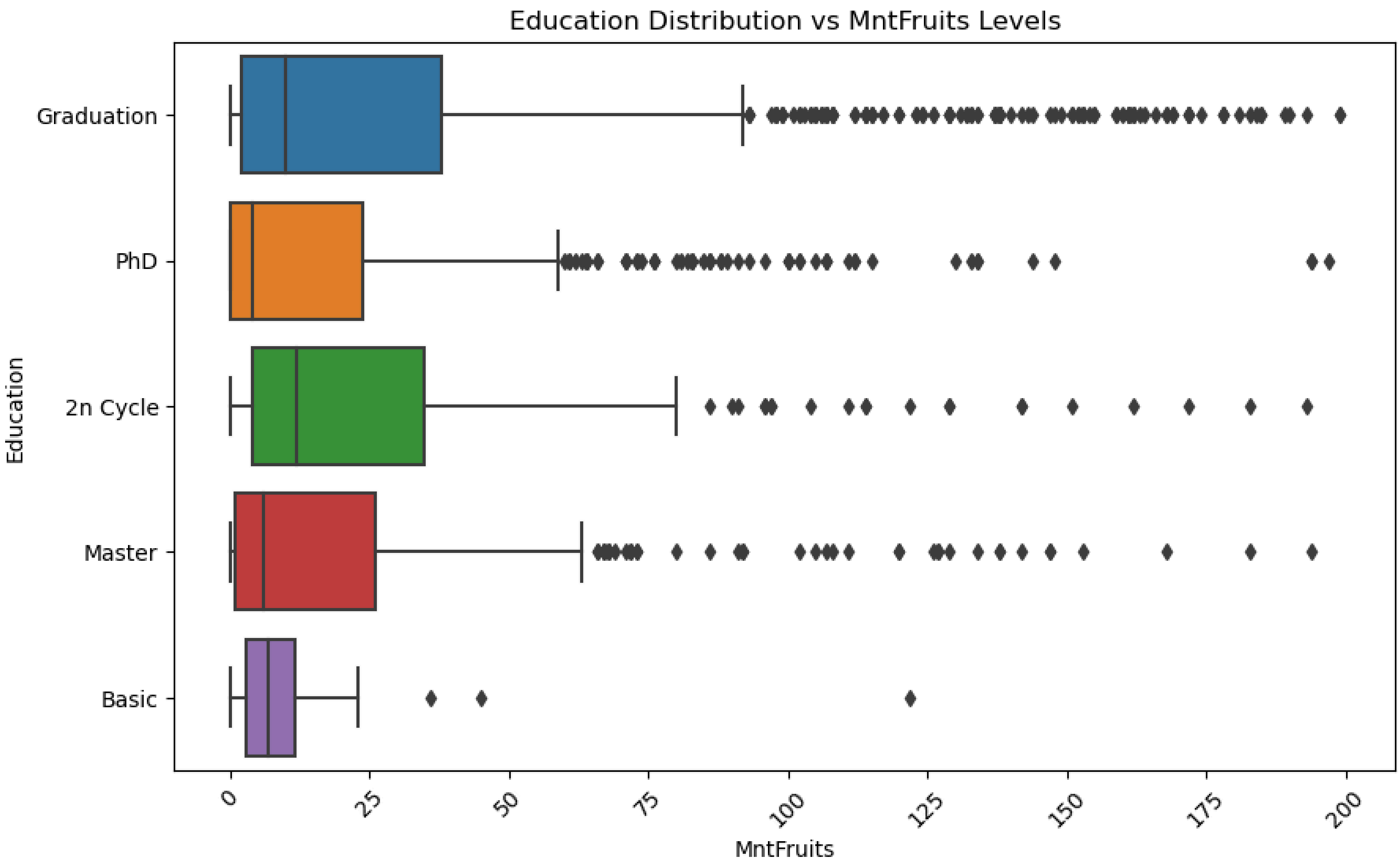
- The majority of customers are graduates.
- Most customers are married and living together.
- Most people have one child.
- Customers prefer buying one product, indicating a right-skewed distribution.
- People mostly buy two products from the website, with a slight difference between those buying one and three products.
- Customers tend to buy three products directly from the store.

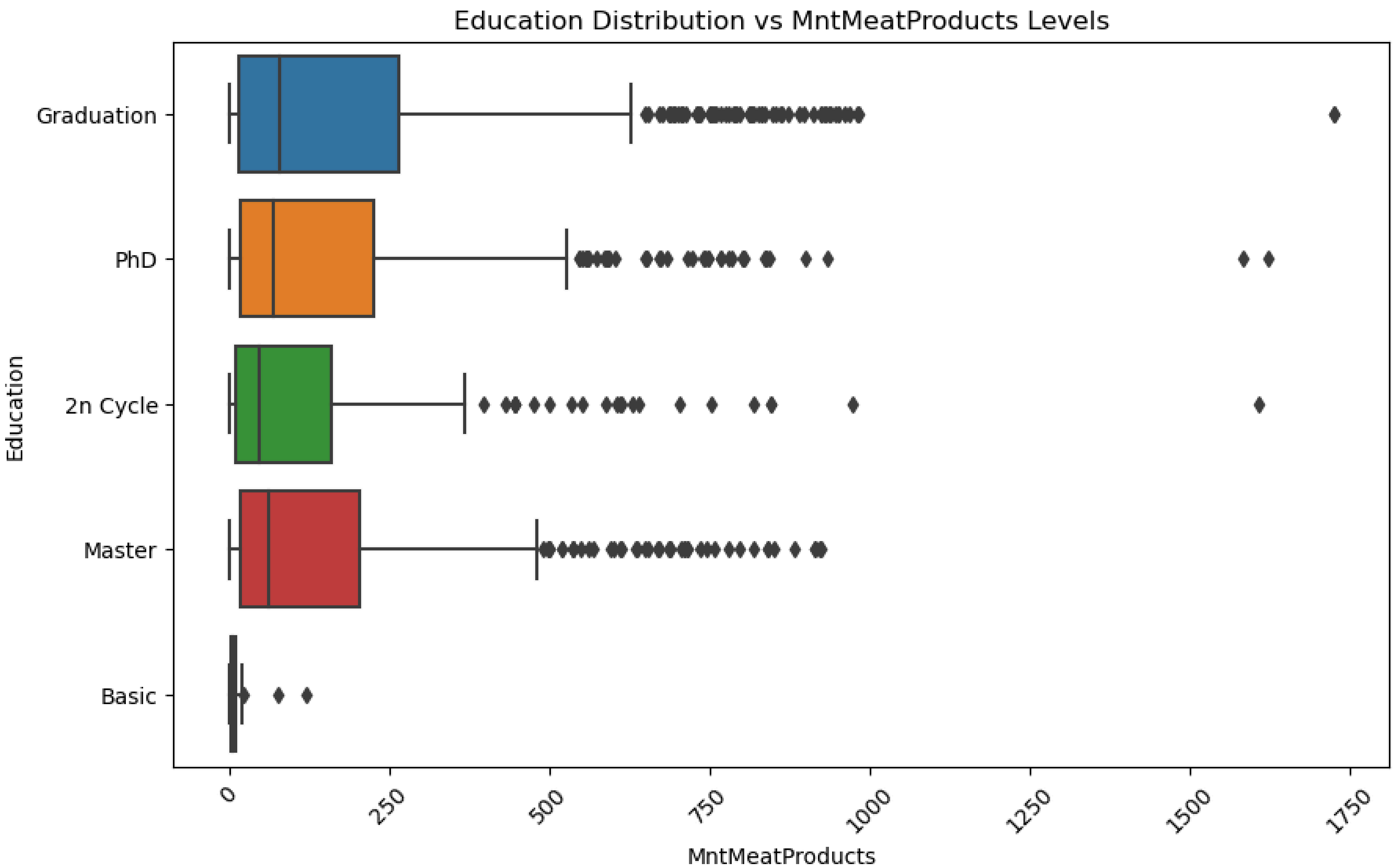
- The majority of people do not prefer making purchases using a catalog.
- Customer acceptance of offers remains almost consistent until Campaign 4, but there is a sudden drop in Campaign 5.
- There are very few complaints, almost around 1-2%.
- Most purchases are made by customers from Spain.

Bivariate

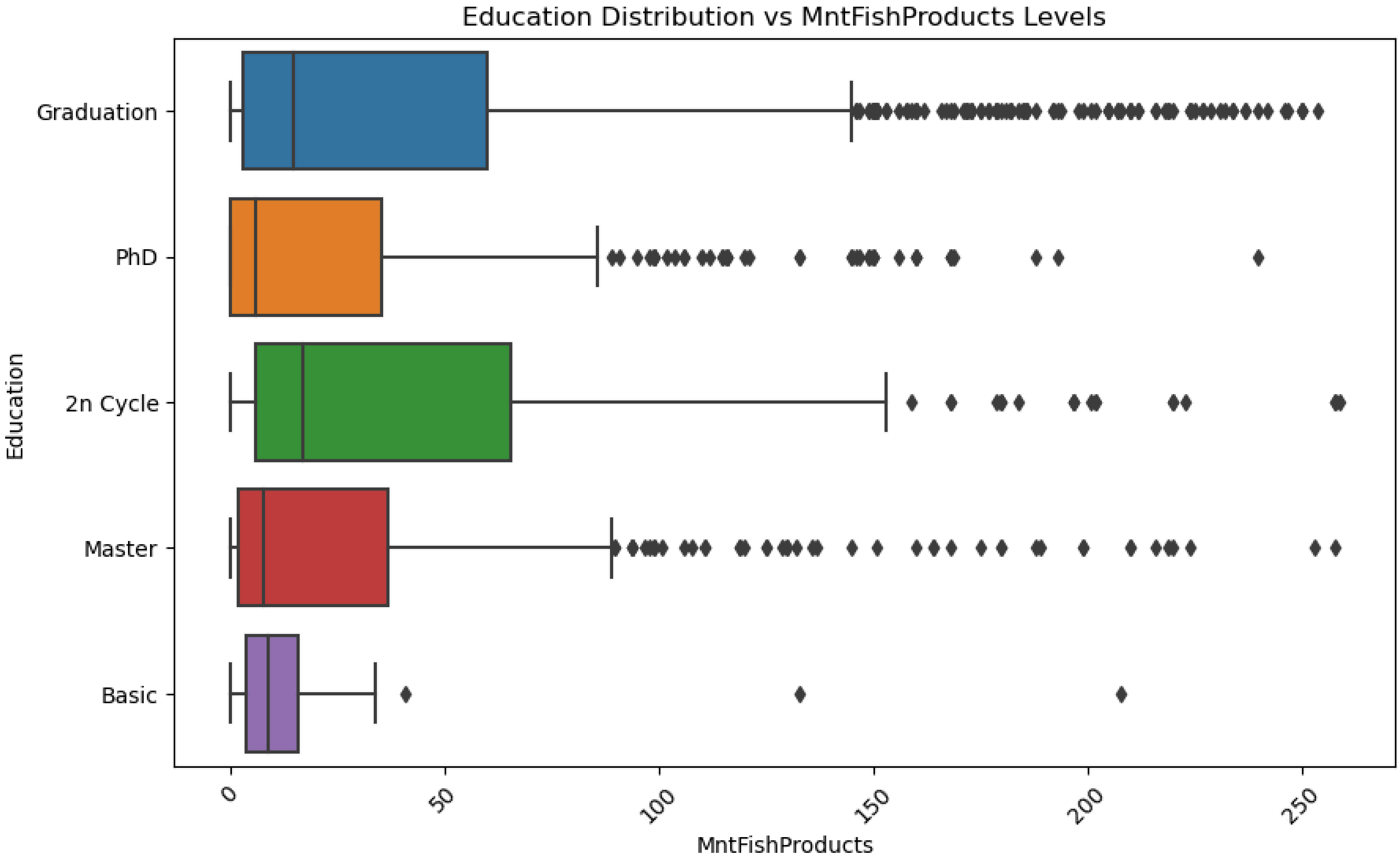
In [15]:
```
#Education vs Amount spend
col=['MntWines', 'MntFruits','MntMeatProducts', 'MntFishProducts', 'MntSweetProducts','MntGoldProds']
for i in col:
    plt.figure(figsize=(10, 6))
    sns.boxplot( x=i,y='Education', data=camp)
    plt.title(f'Education Distribution vs {i} Levels')
    plt.xticks(rotation=45)
    plt.show()
```

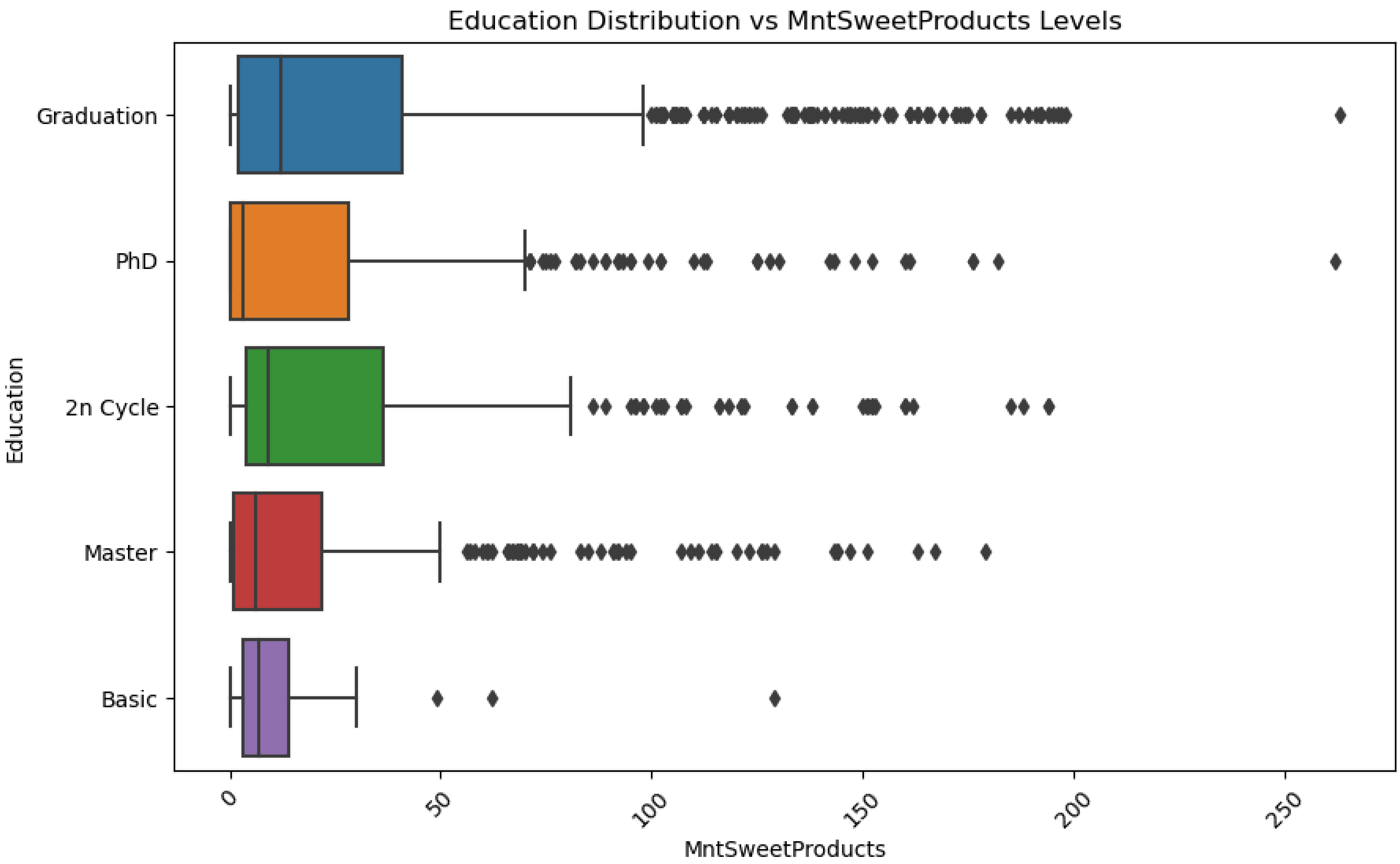

Education Distribution vs MntWines Levels
Education
Graduation
PhD
2n Cycle
Master
Basic
MntWines
0
200
400
600
800
1000
1200
1400


Education Distribution vs MntFruits Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
25
50
75
100
125
150
175
200
MntFruits

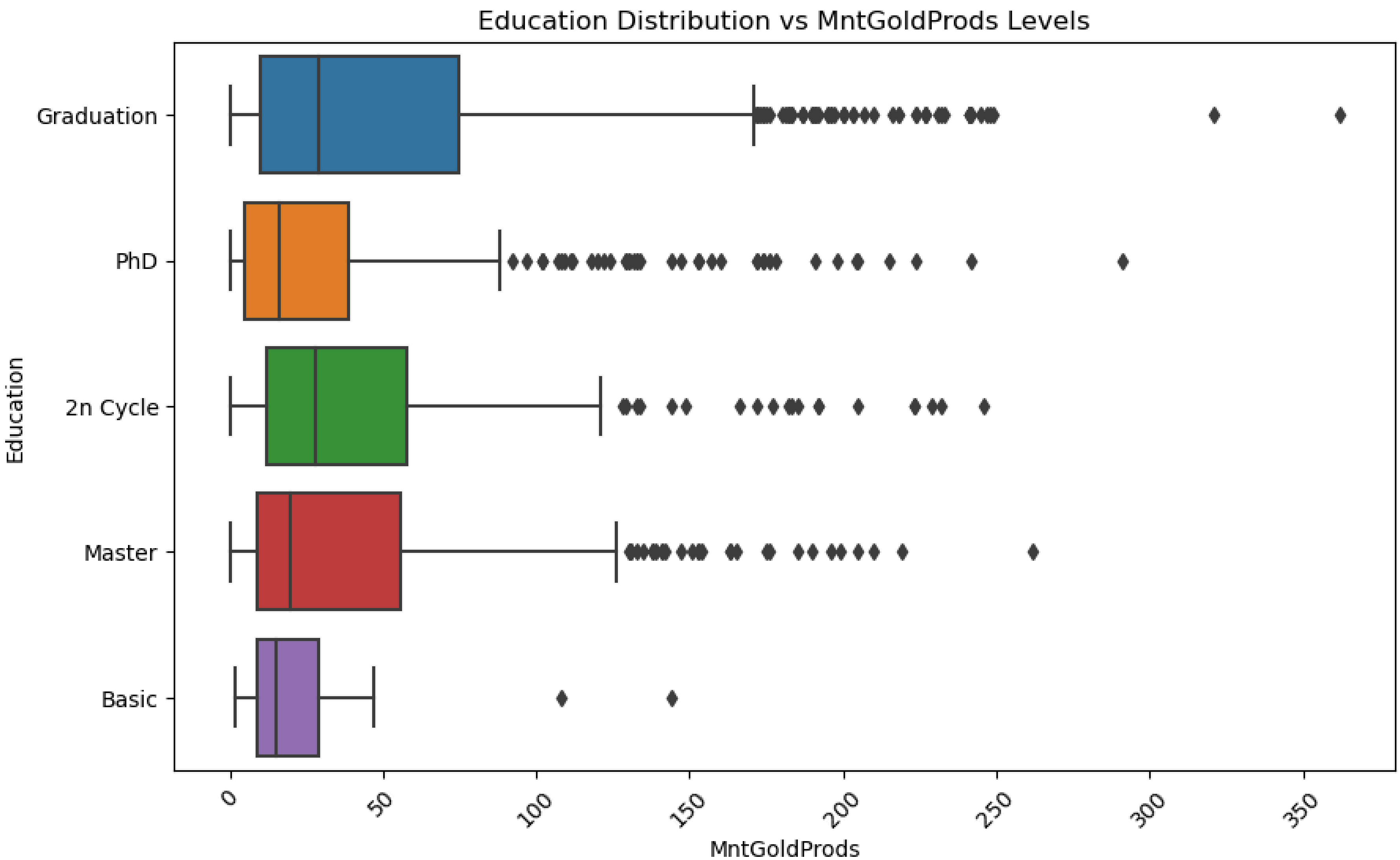


Education Distribution vs MntMeatProducts Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
250
500
750
1000
1250
1500
1750
MntMeatProducts


Education Distribution vs MntFishProducts Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
50
100
150
200
250
MntFishProducts



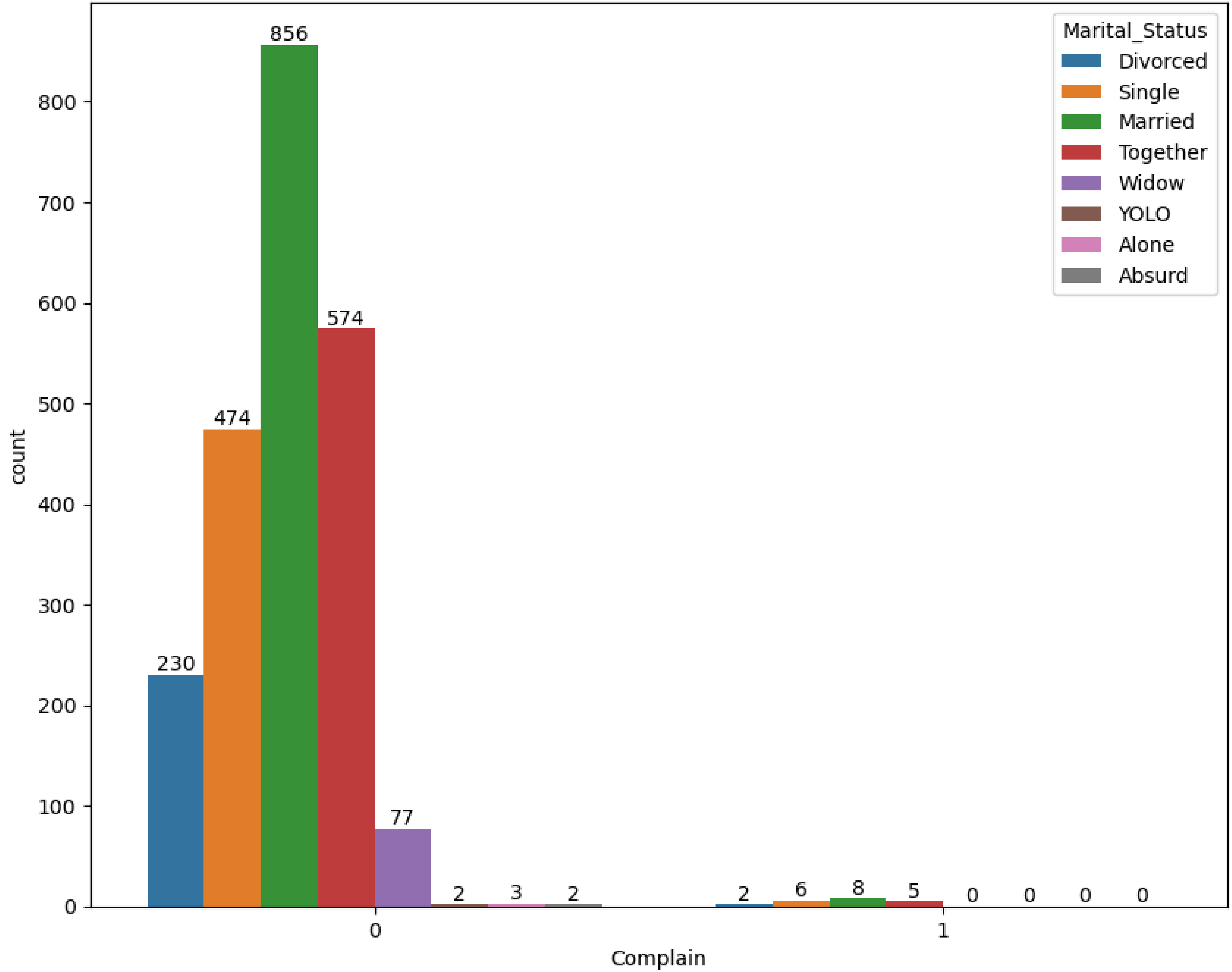
Education Distribution vs MntSweetProducts Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
50
100
150
200
250
MntSweetProducts



- People with a PhD degree spend most of their money on wine.
- Except for those with only basic education, the rest of the educated individuals tend to spend money on meat.
- It appears that customers with a graduation or second-cycle degree spend the most on fish products.
- Graduates primarily spend their money on gold.

In [16]:

```
# Marital_Status vs Complain
plt.figure(figsize=(10,8))
sns.countplot(data=camp,x='Complain',hue='Marital_Status')
ax=plt.gca()
for i in ax.containers:
    ax.bar_label(i)
plt.show()
```

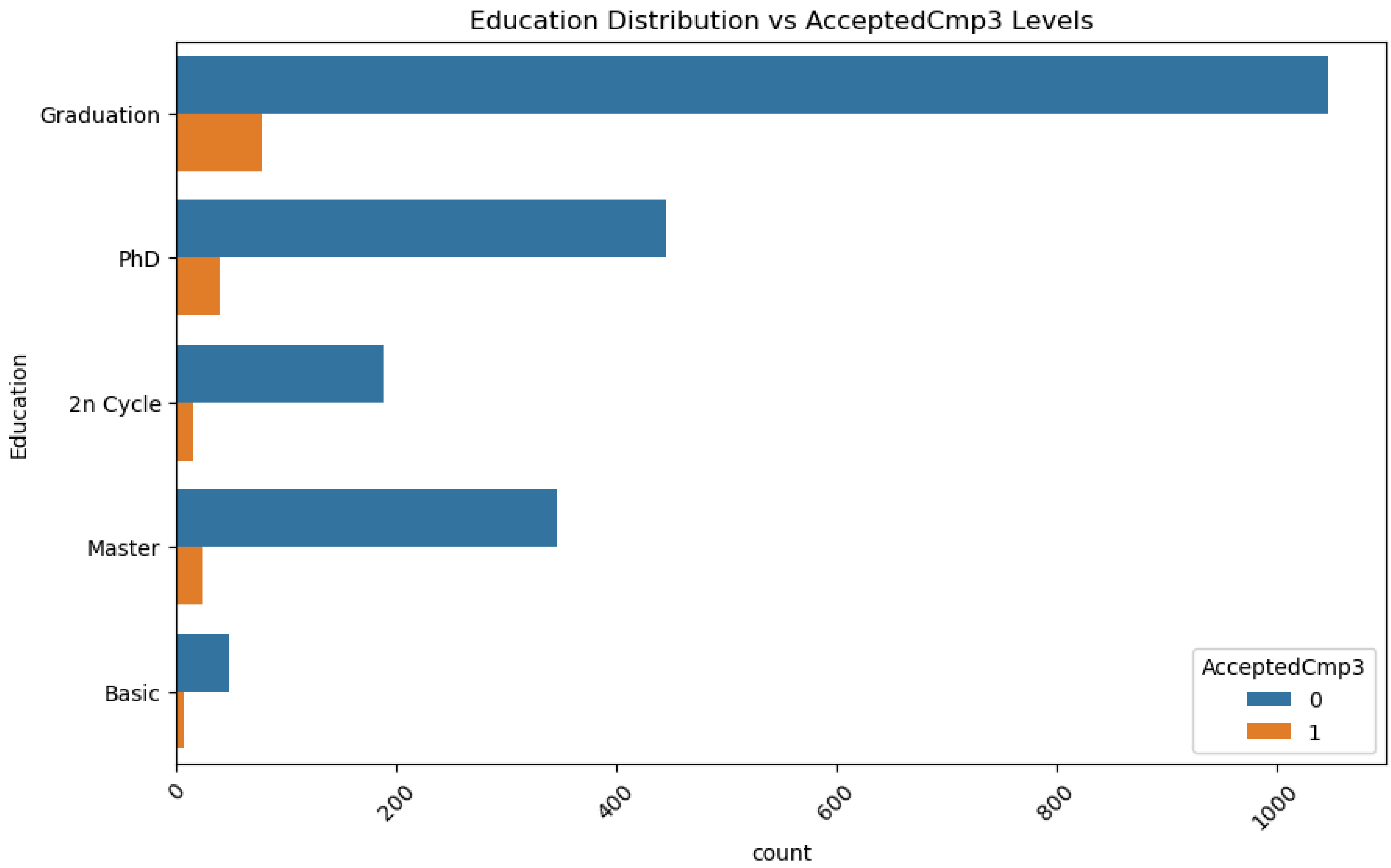


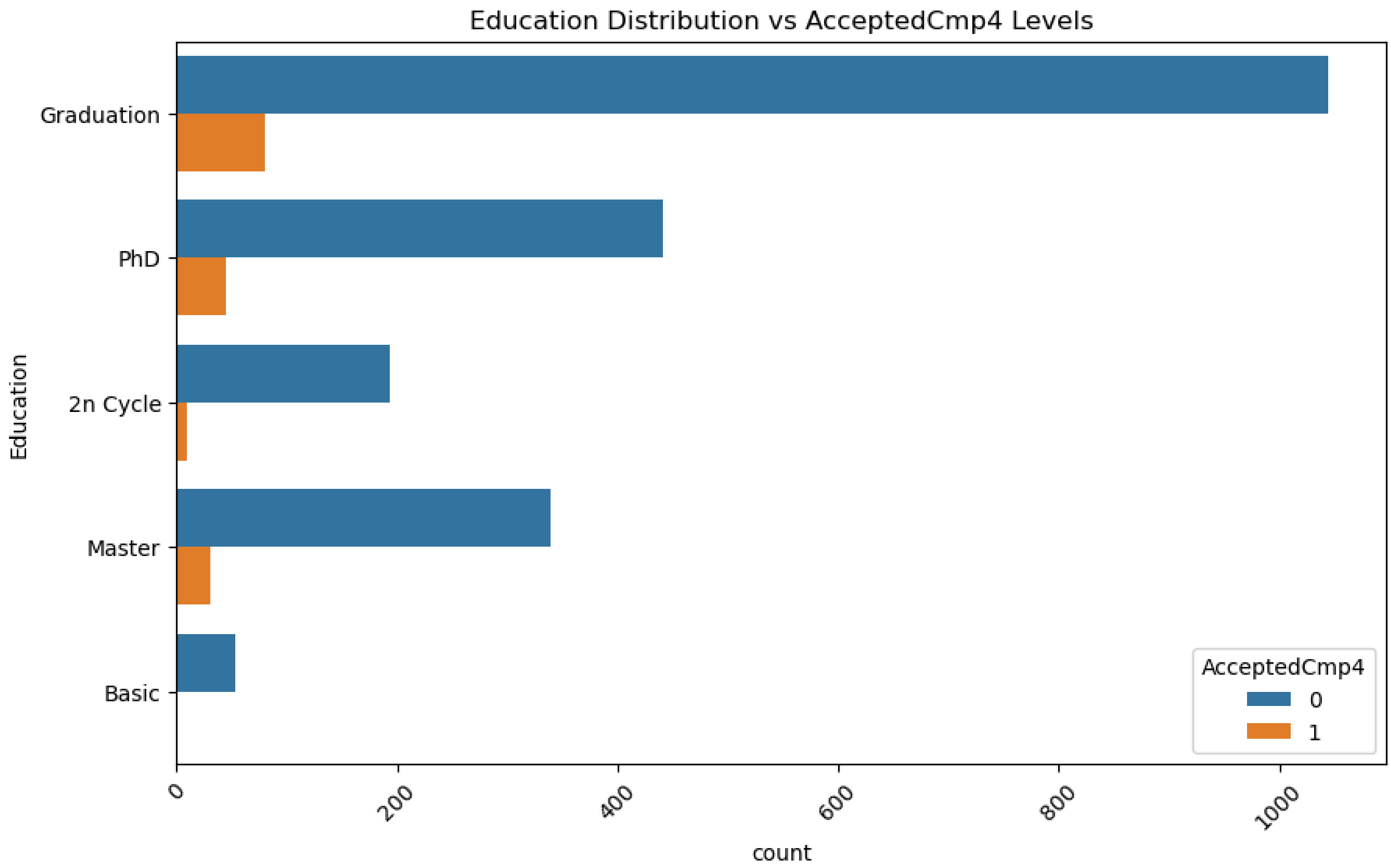
Marital_Status
Divorced
Single
Married
Together
Widow
YOLO
Alone
Absurd
count
Complain
0
100
200
300
400
500
600
700
800
1
856
474
574
230
77
2
3
2
2
6
8
5
0
0
0
0

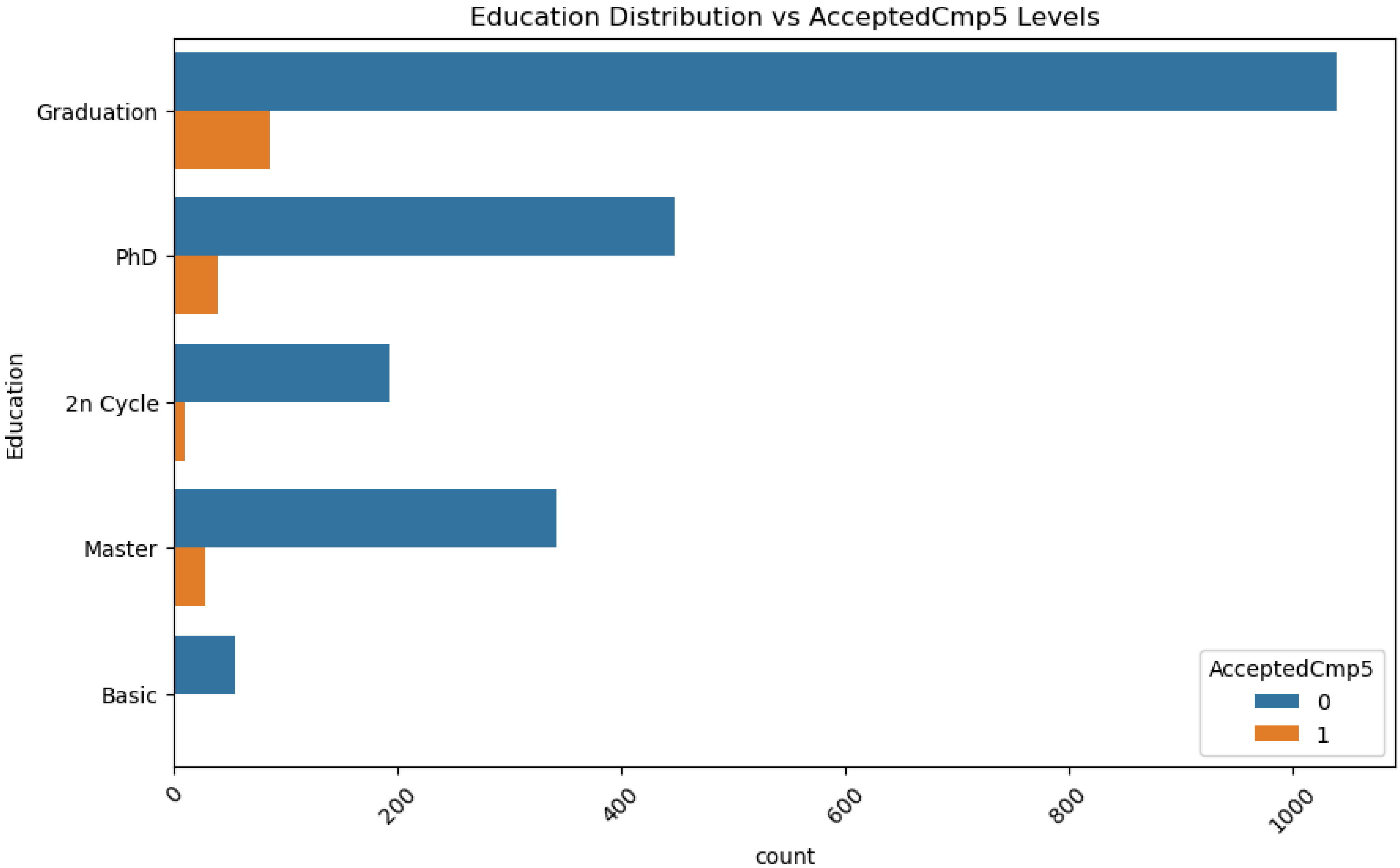
- Here we can see there not much compalin

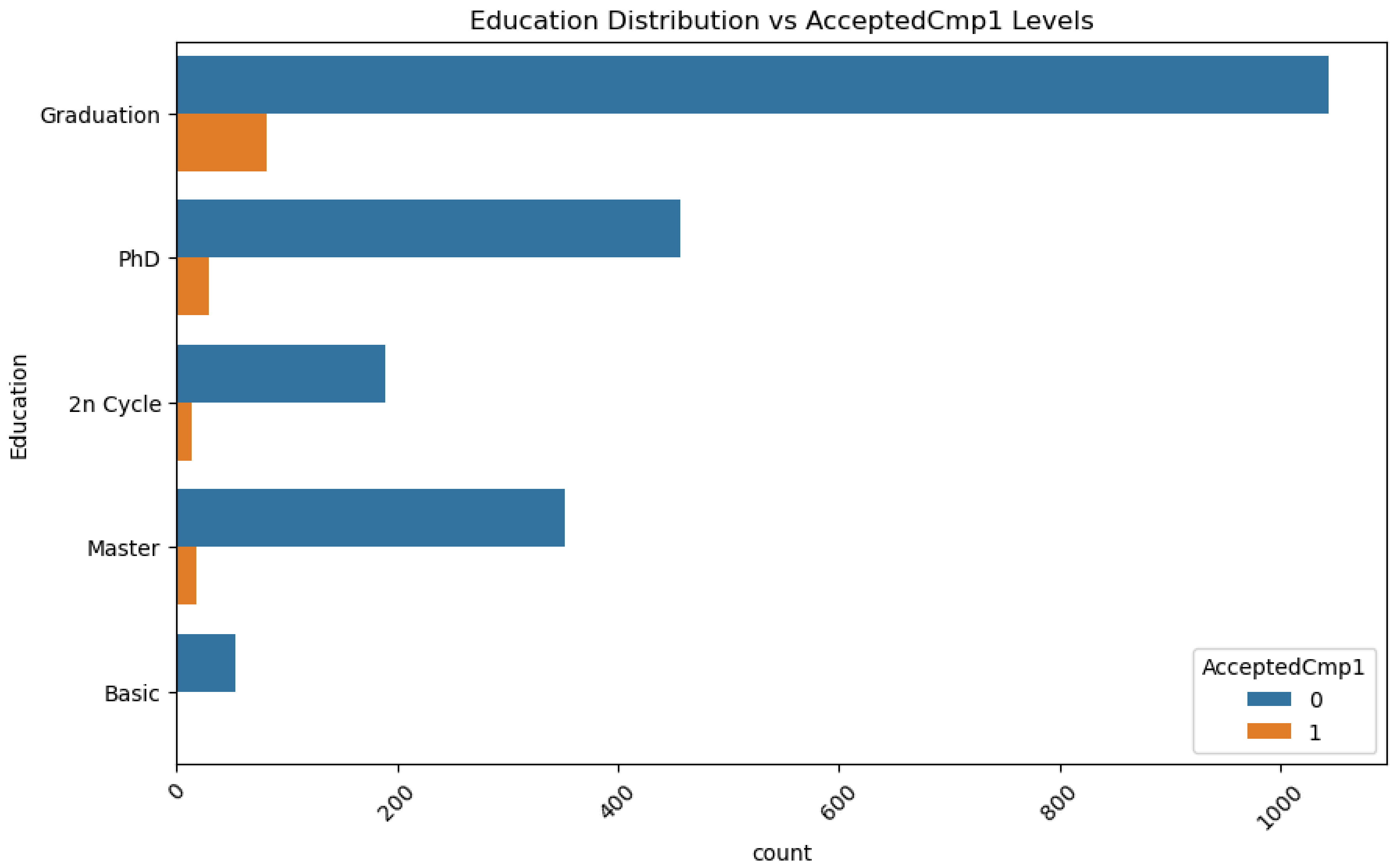
In [17]:
```
# Education vs Camp
```

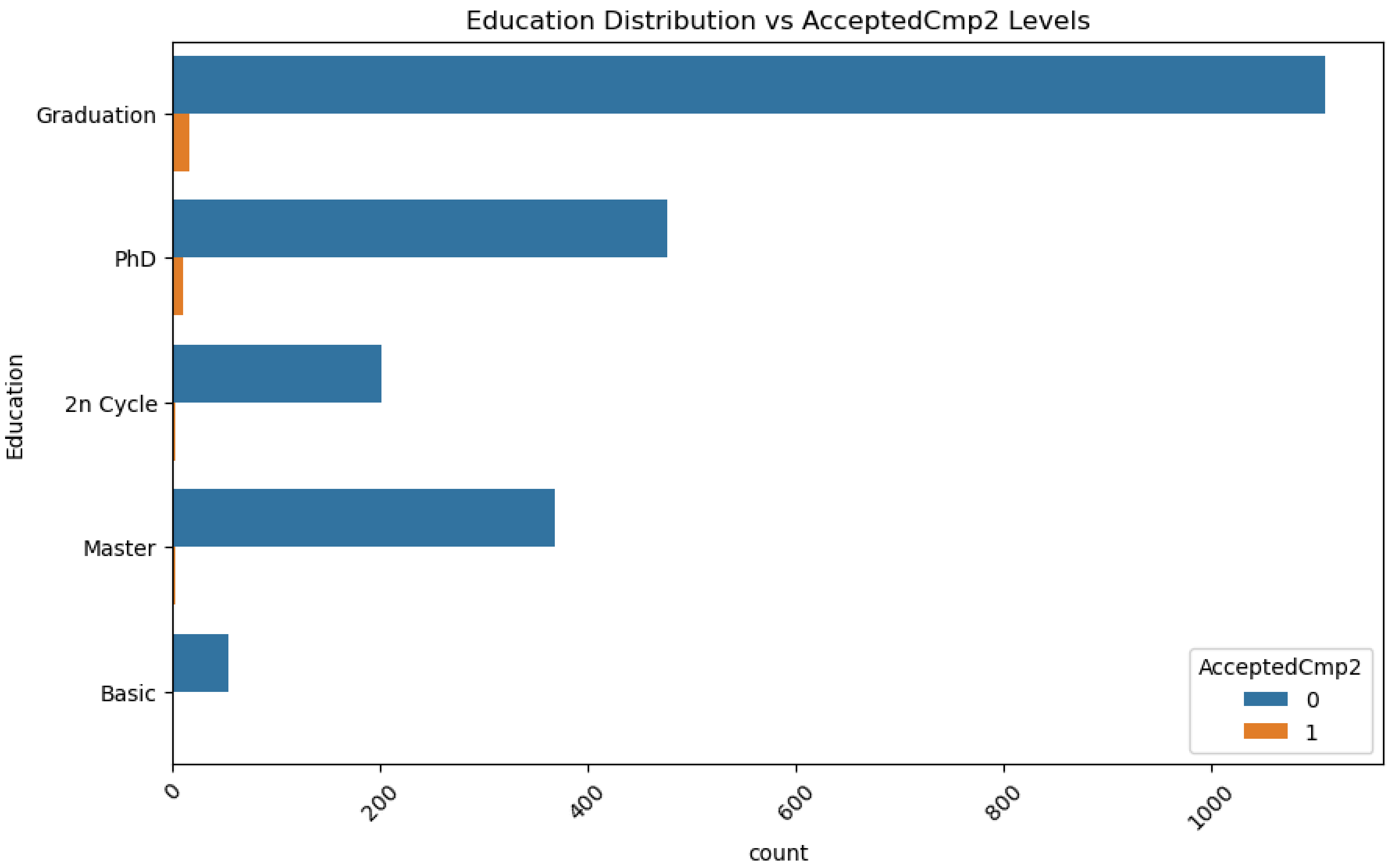
In [18]:
```
#Education vs Amount spend
col=['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2']
for i in col:
    plt.figure(figsize=(10, 6))
    sns.countplot( hue=i,y='Education', data=camp)
    plt.title(f'Education Distribution vs {i} Levels')
    plt.xticks(rotation=45)
    plt.show()
```


Education Distribution vs AcceptedCmp3 Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
200
400
600
800
1000
count
AcceptedCmp3
0
1


Education Distribution vs AcceptedCmp4 Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
200
400
600
800
1000
count
AcceptedCmp4
0
1


Education Distribution vs AcceptedCmp5 Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
200
400
600
800
1000
count
AcceptedCmp5
0
1


Education Distribution vs AcceptedCmp1 Levels
Graduation
PhD
2n Cycle
Master
Basic
Education
0
200
400
600
800
1000
count
AcceptedCmp1
0
1

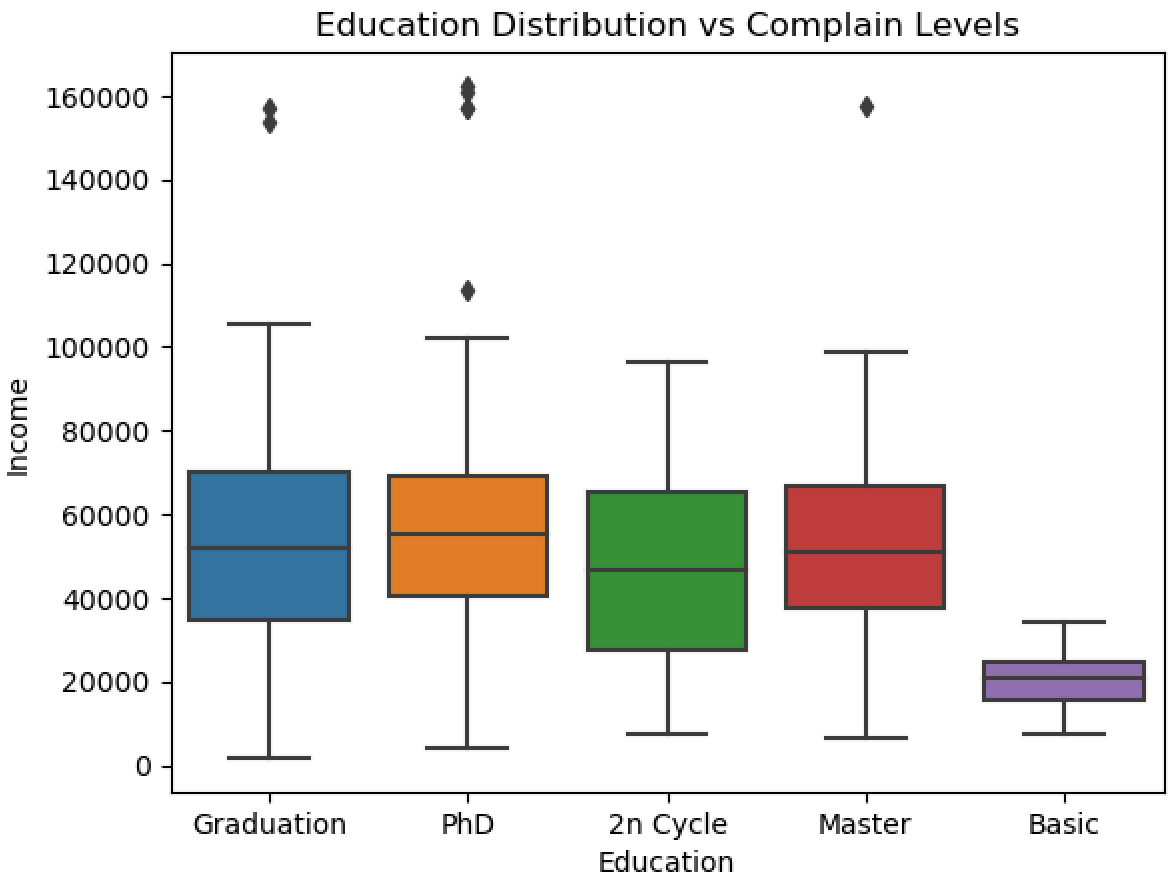


- Most of the customers who accepted the offer have a graduation background.

In [19]:
```
# Income vs Education
```

In [26]:
```
# plt.figure(figsize=(10, 6))
sns.boxplot( x='Education',y='Income', data=camp)
```

```
plt.title(f'Education Distribution vs {i} Levels')
ax=plt.gca()
for i in ax.containers:
    ax.bar_label(i)
# plt.show()
```

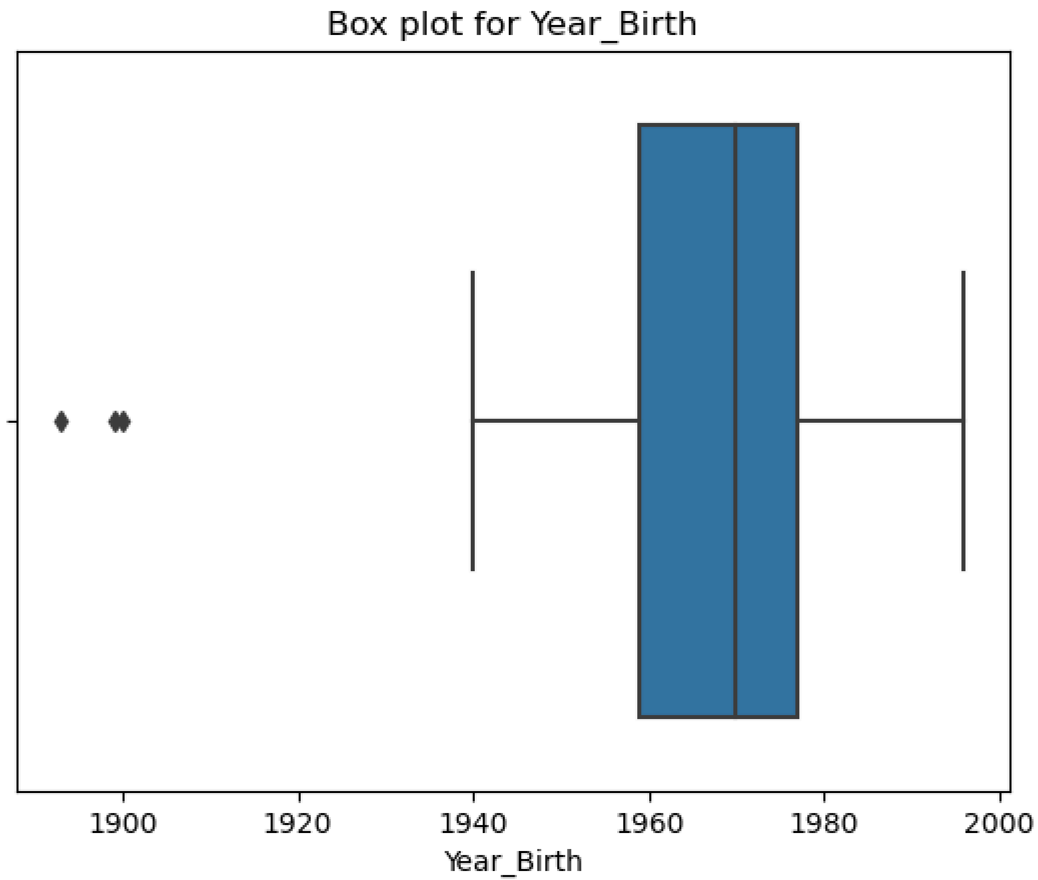


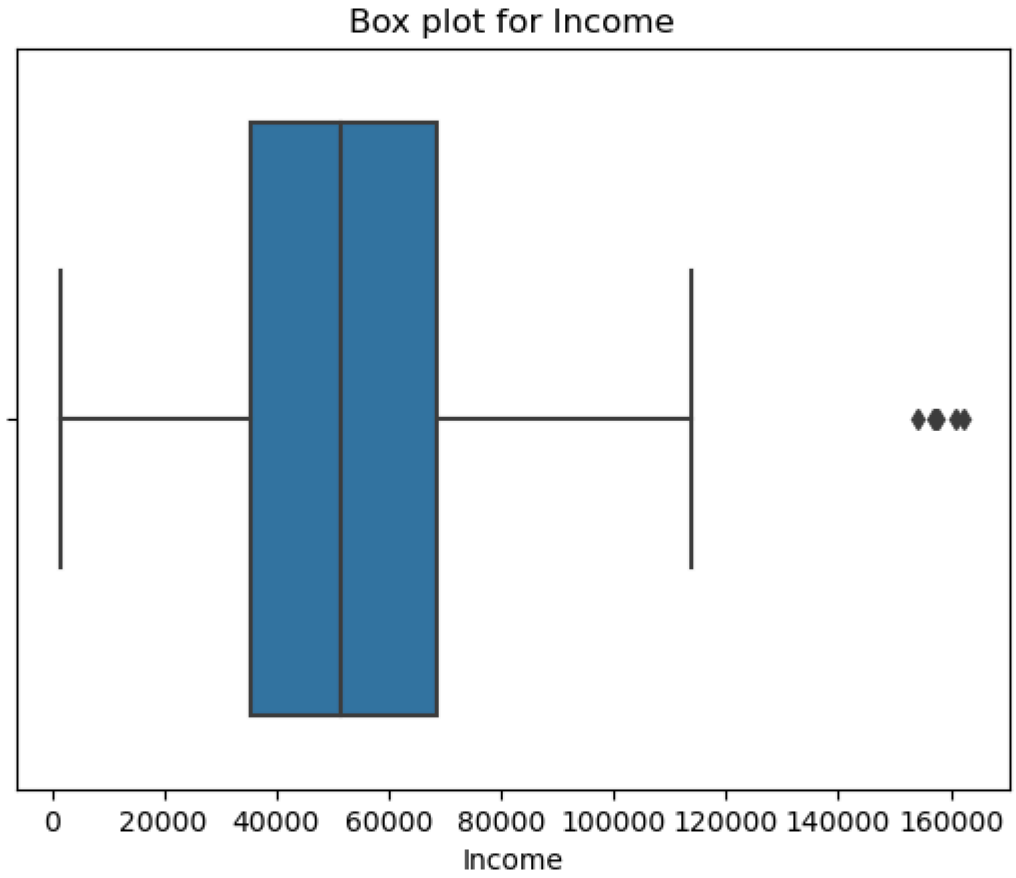
- Except for those with a basic education, customers with other degrees have an income greater than 30,000

Outlier Detection Using Boxplot

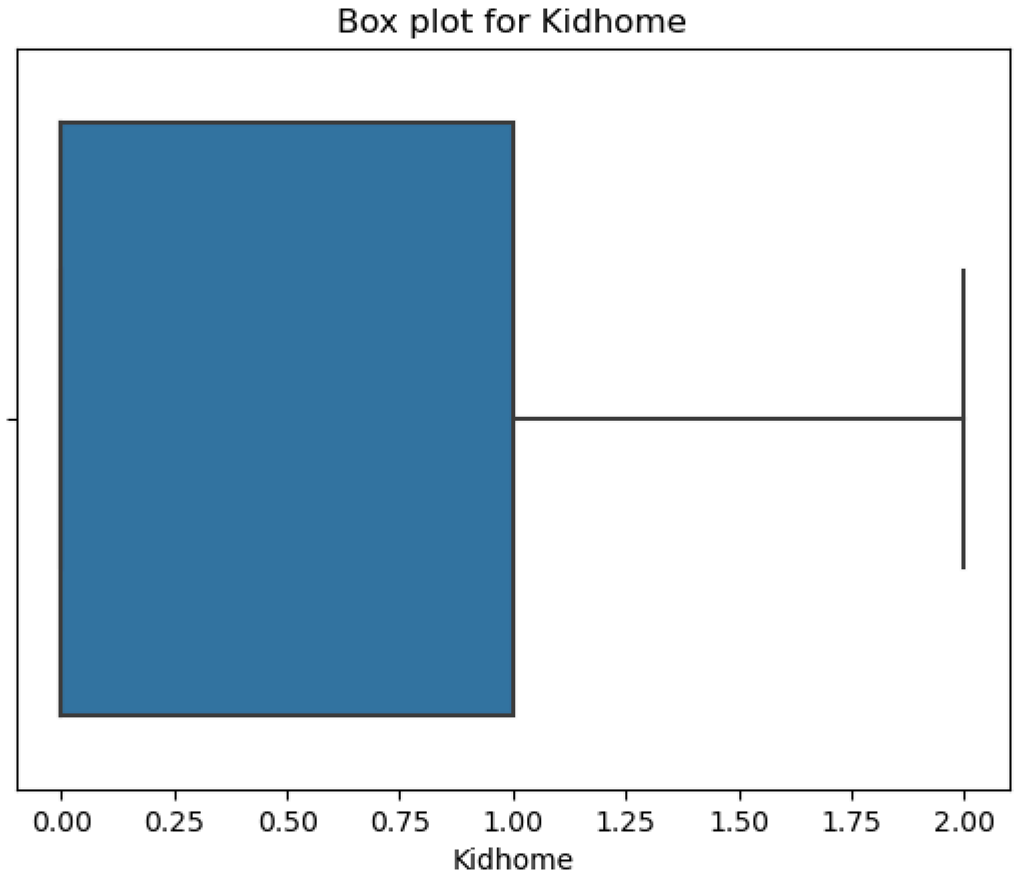
In [22]:

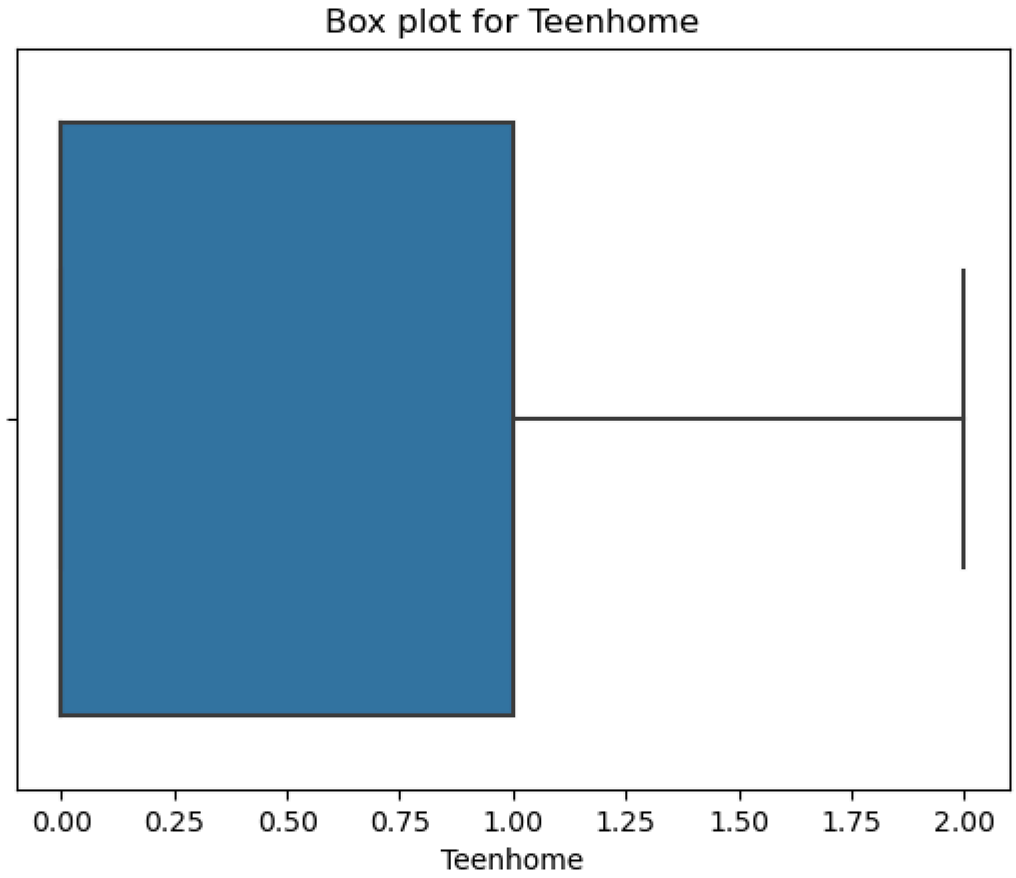
```
col=['Year_Birth','Income', 'Kidhome',
       'Teenhome', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain']
for i in col:
    sns.boxplot(data=camp,x=i)
    plt.title(f"Box plot for {i}")
    plt.show()
```

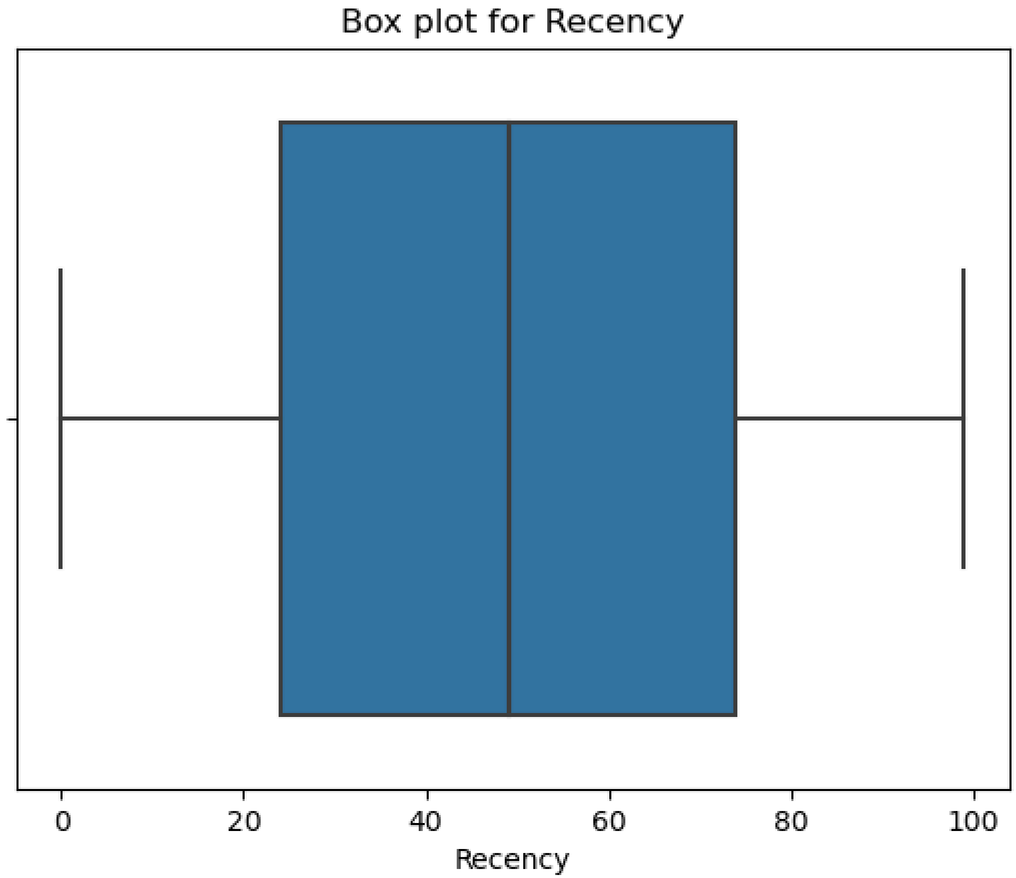


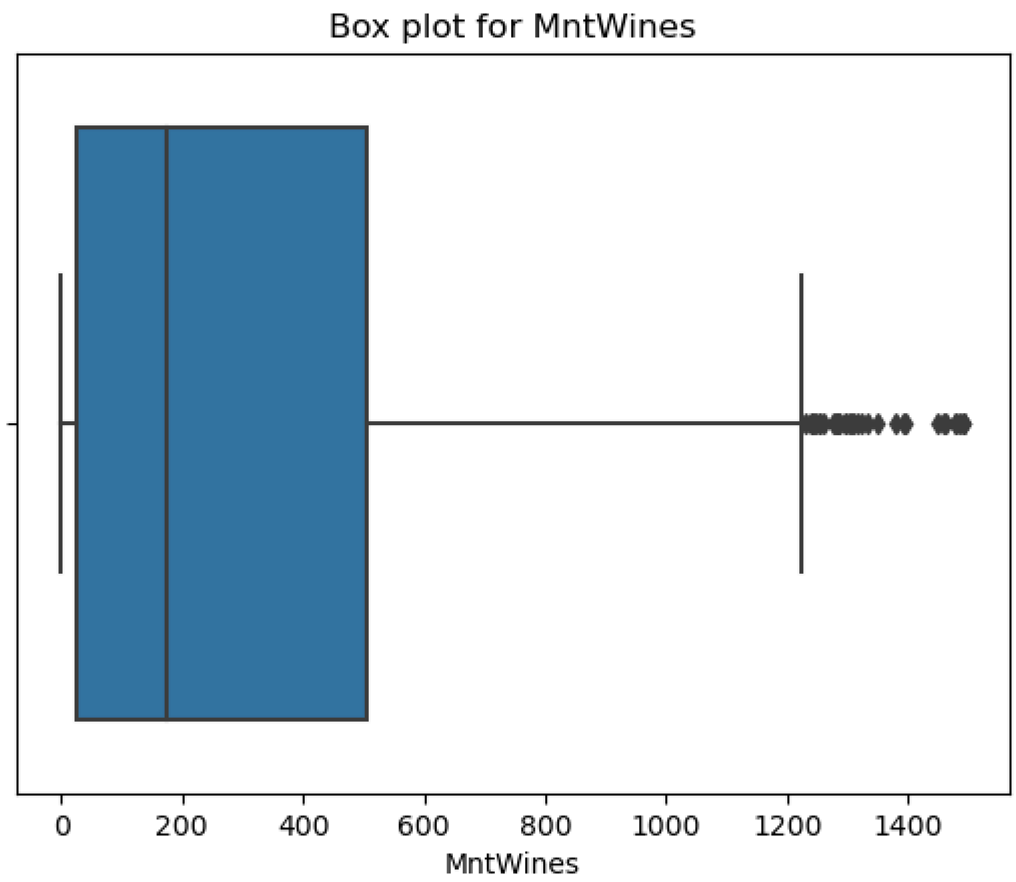


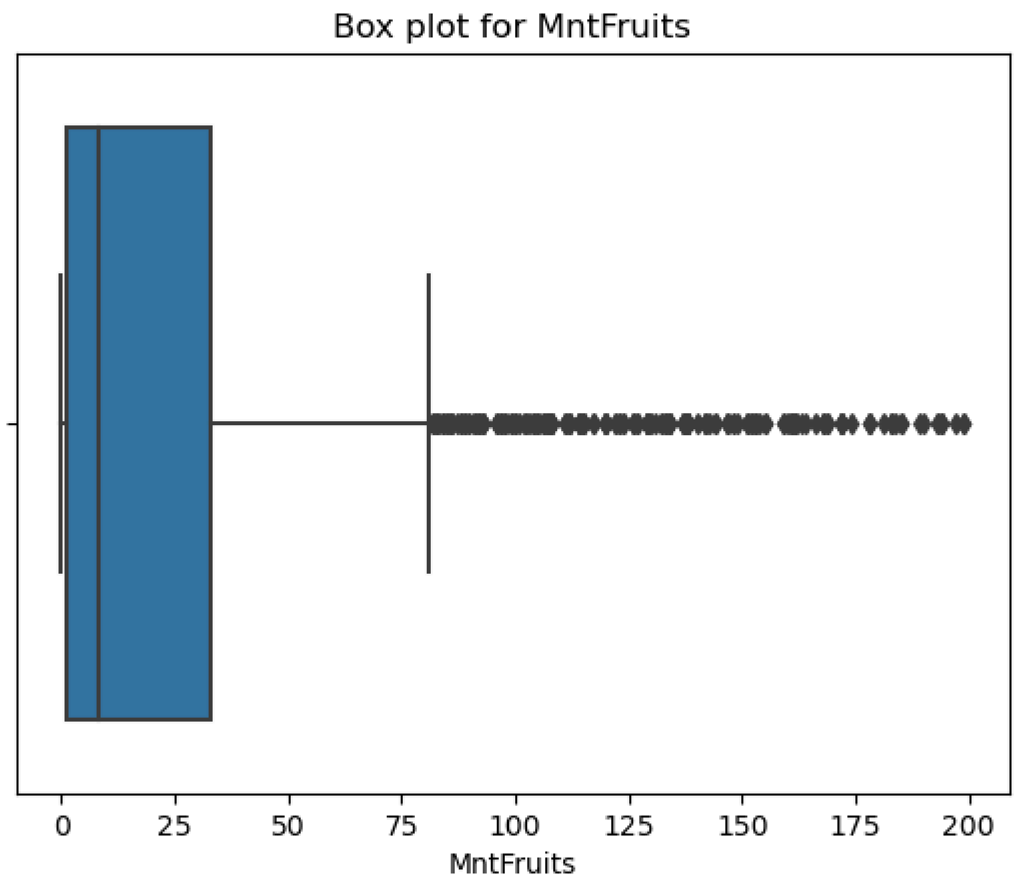
Box plot for Income
0
20000
40000
60000
80000
100000
120000
140000
160000
Income


Box plot for Kidhome
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Kidhome

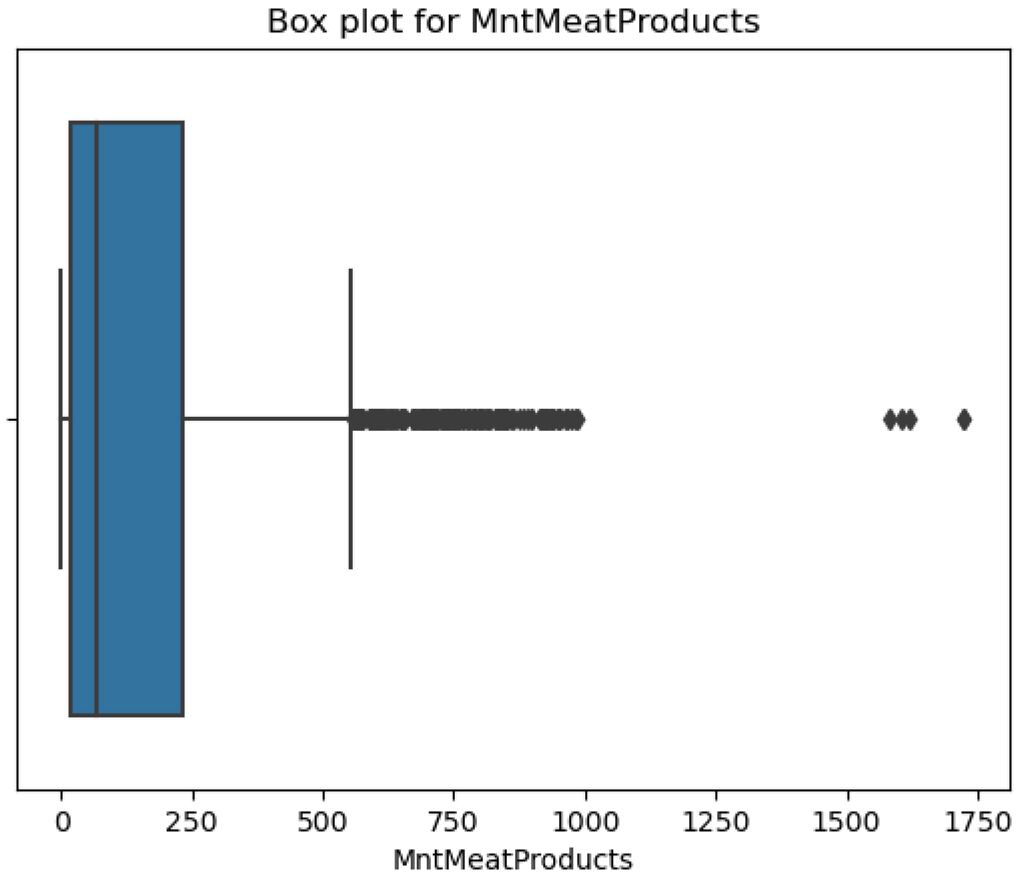

Box plot for Teenhome
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Teenhome

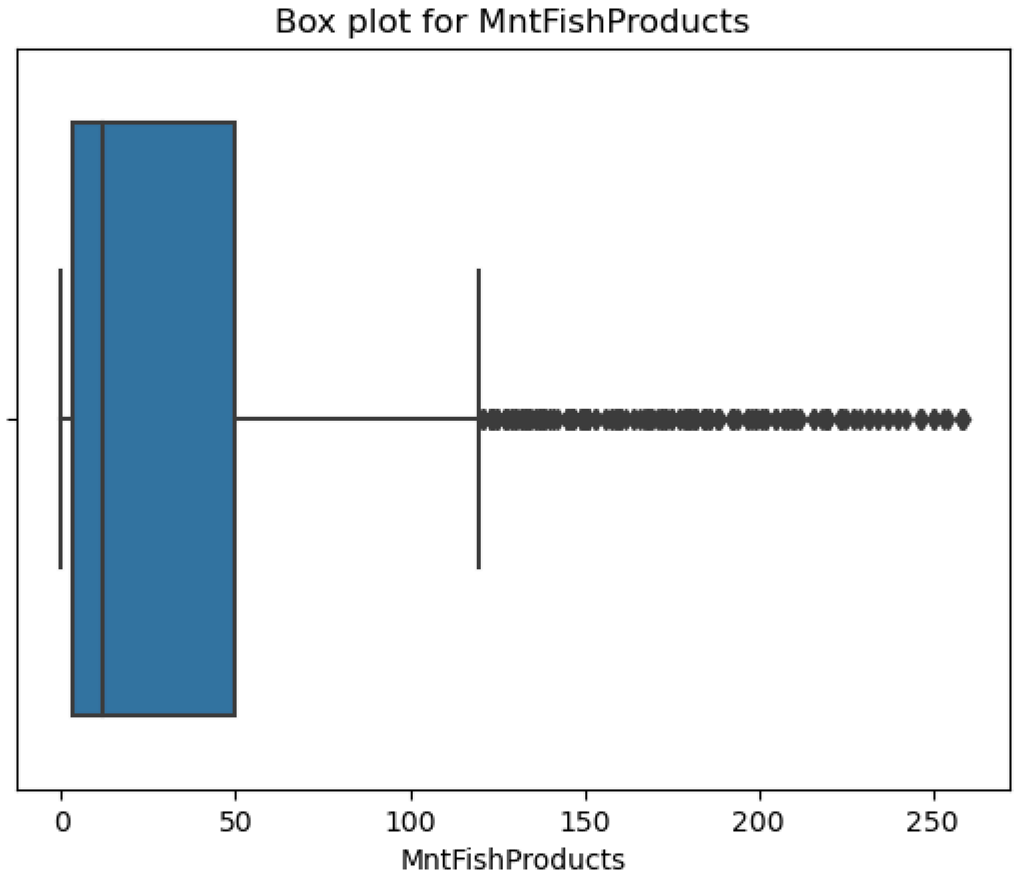

Box plot for Recency
0
20
40
60
80
100
Recency

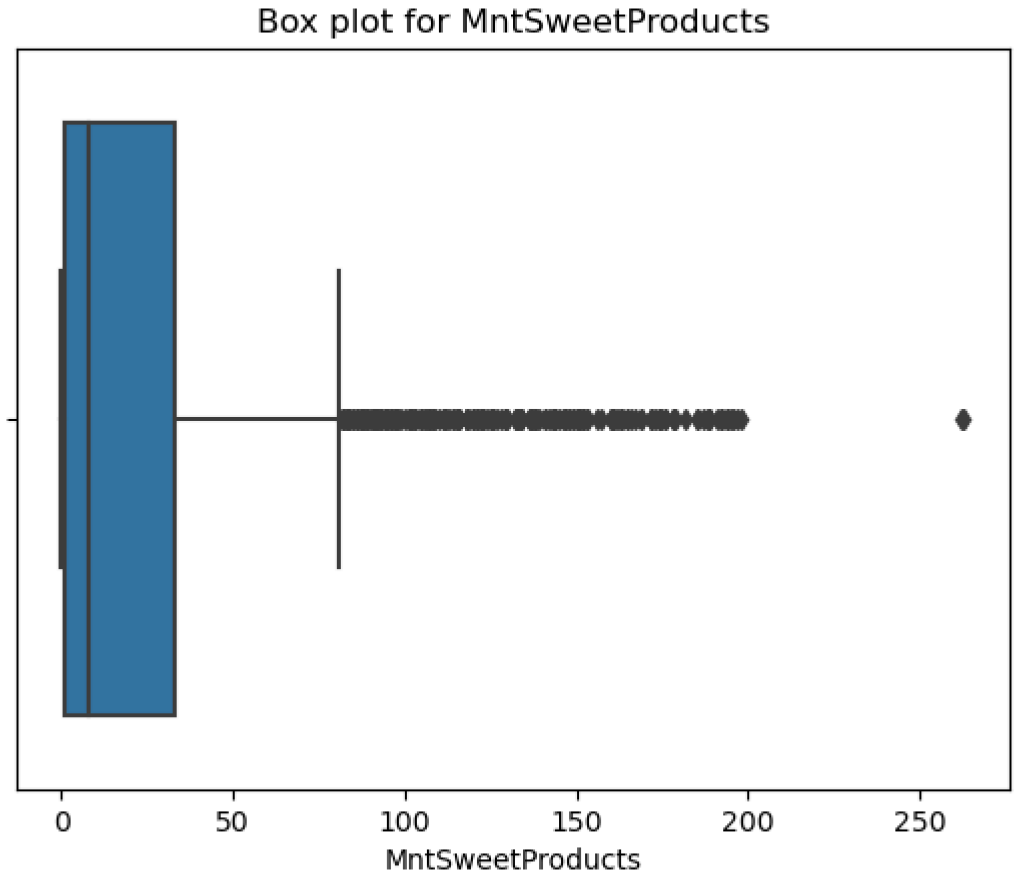

Box plot for MntWines
0
200
400
600
800
1000
1200
1400
MntWines

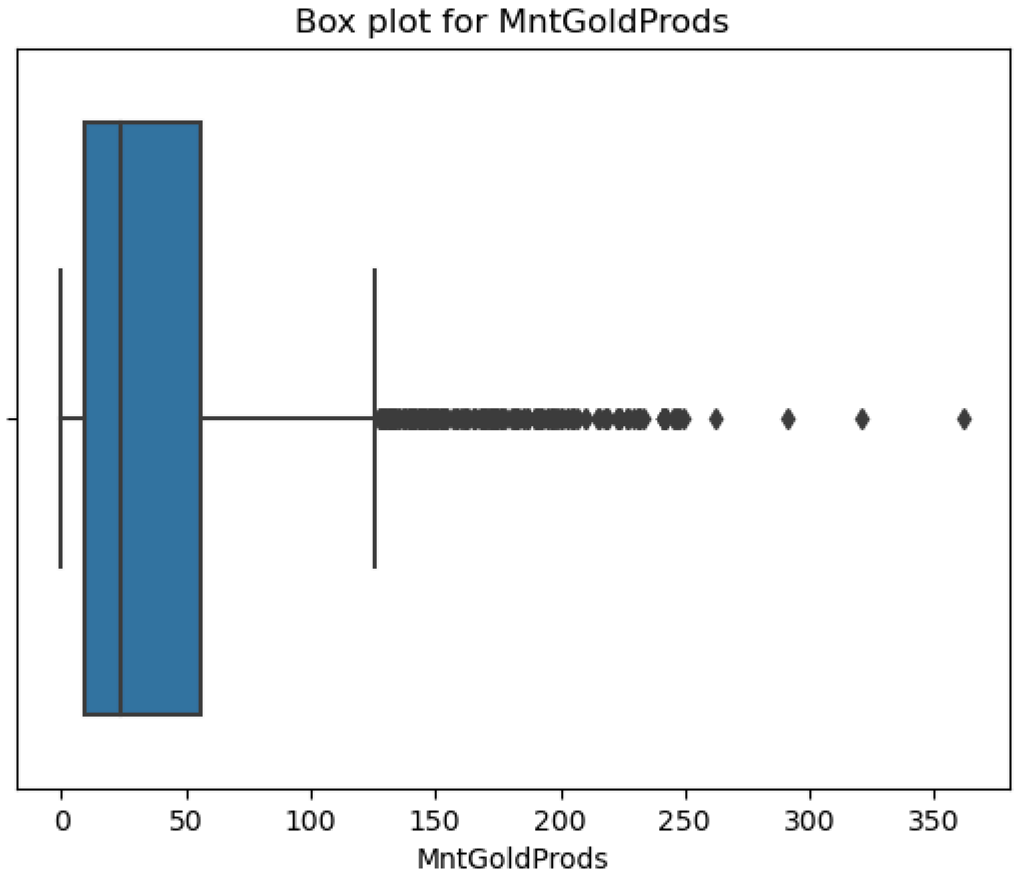


Box plot for MntFruits
0
25
50
75
100
125
150
175
200
MntFruits

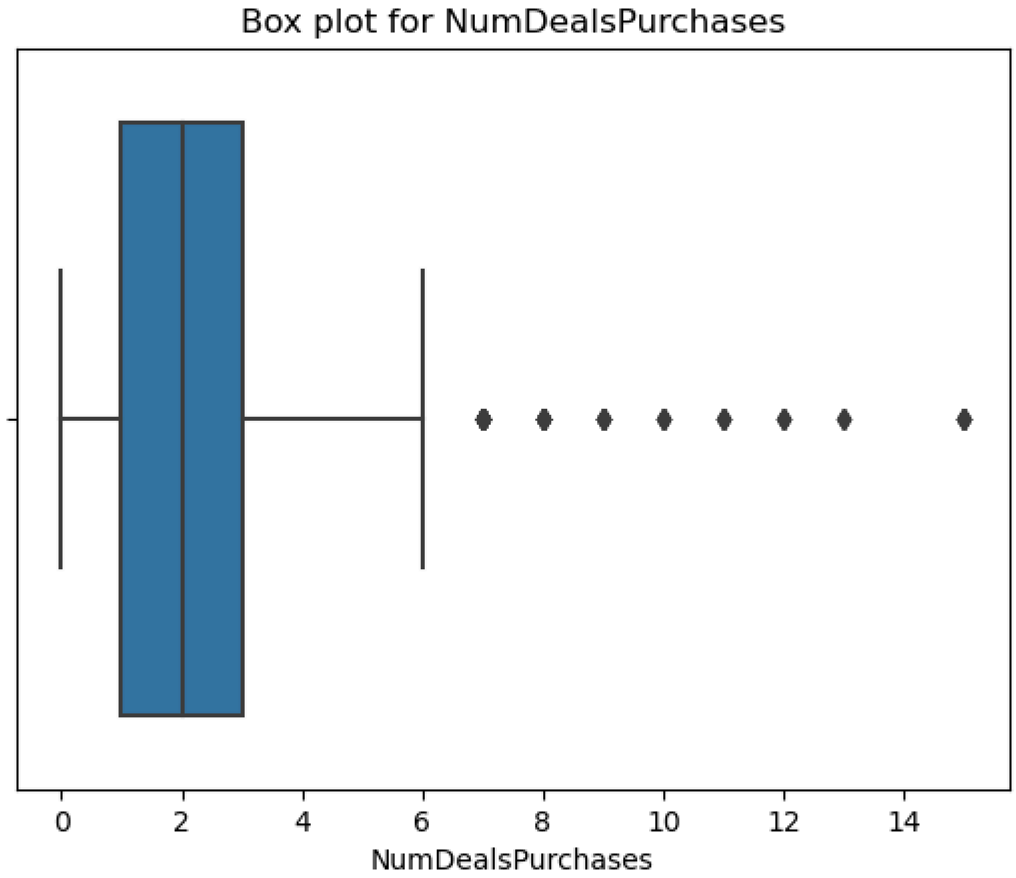

Box plot for MntMeatProducts
0
250
500
750
1000
1250
1500
1750
MntMeatProducts

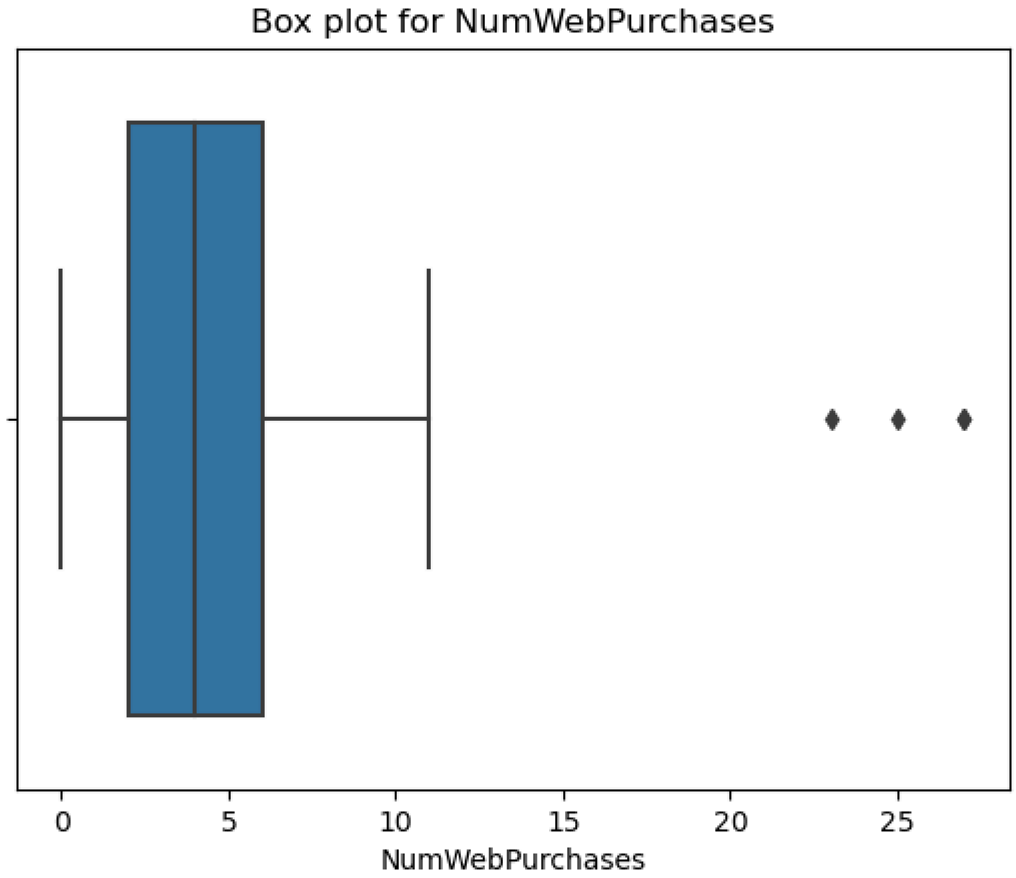

Box plot for MntFishProducts
0
50
100
150
200
250
MntFishProducts


Box plot for MntSweetProducts
0
50
100
150
200
250
MntSweetProducts

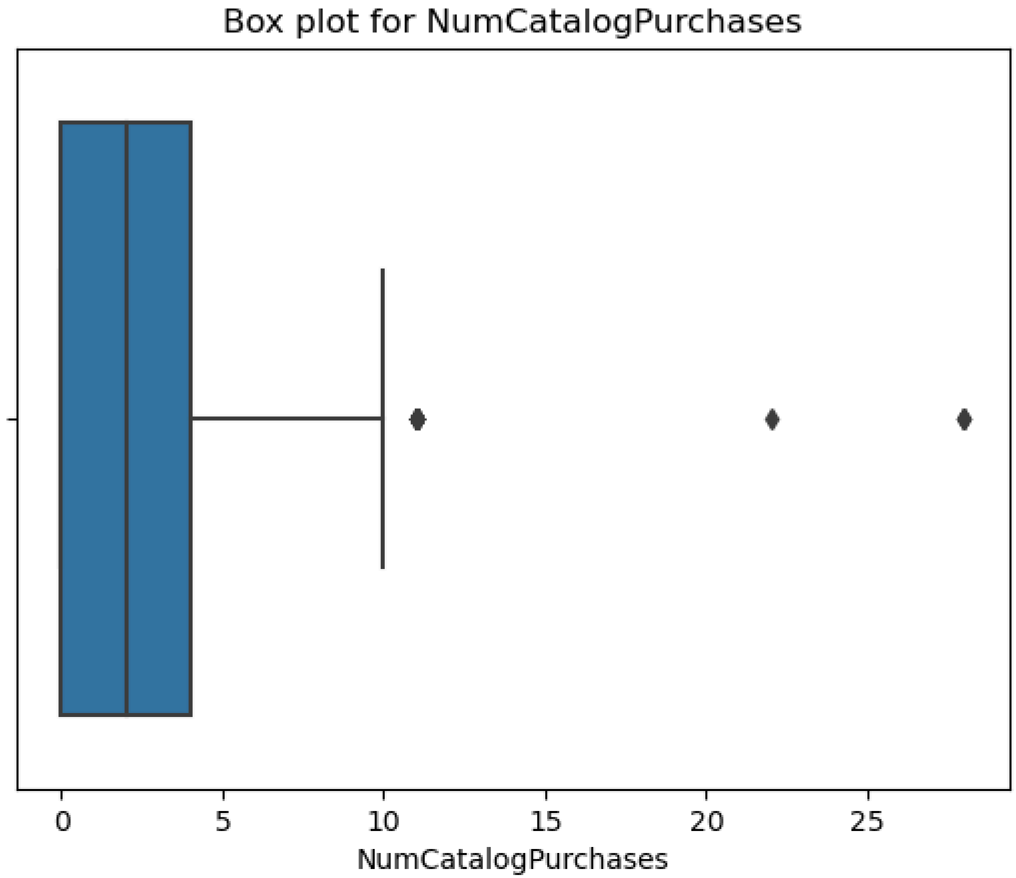


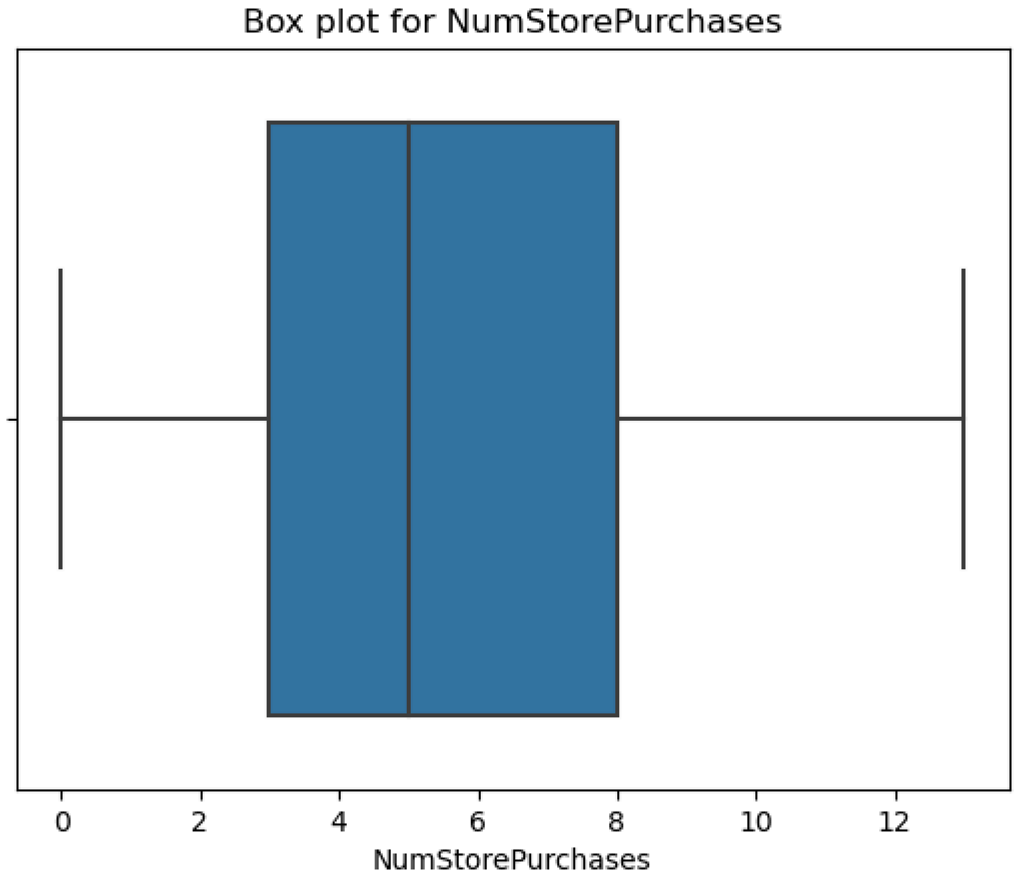
Box plot for MntGoldProds
0
50
100
150
200
250
300
350
MntGoldProds


Box plot for NumDealsPurchases
0
2
4
6
8
10
12
14
NumDealsPurchases

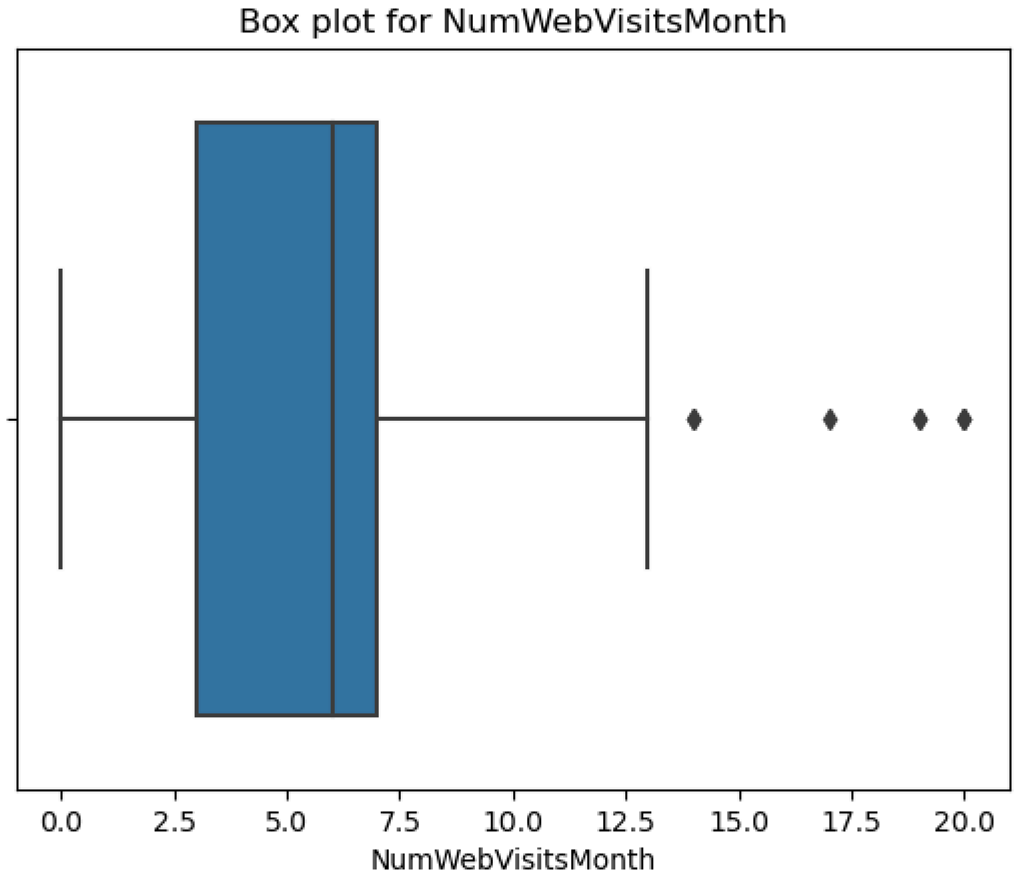


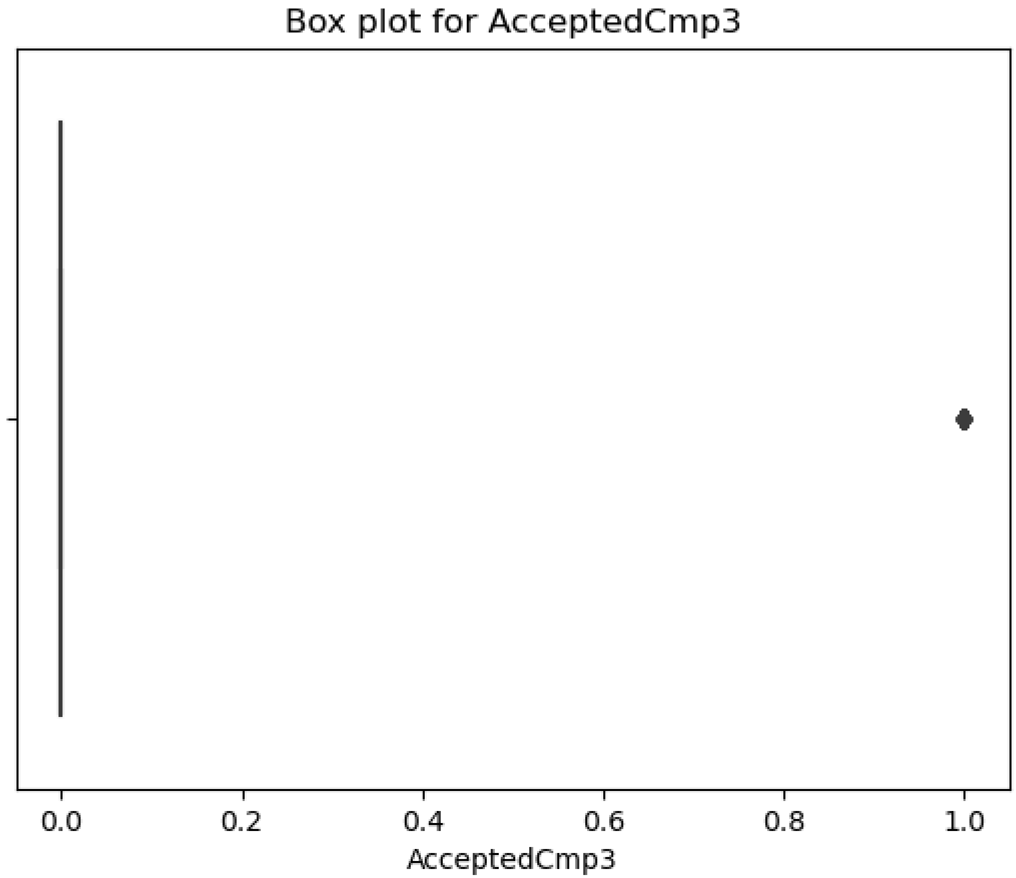
Box plot for NumWebPurchases
0
5
10
15
20
25
NumWebPurchases

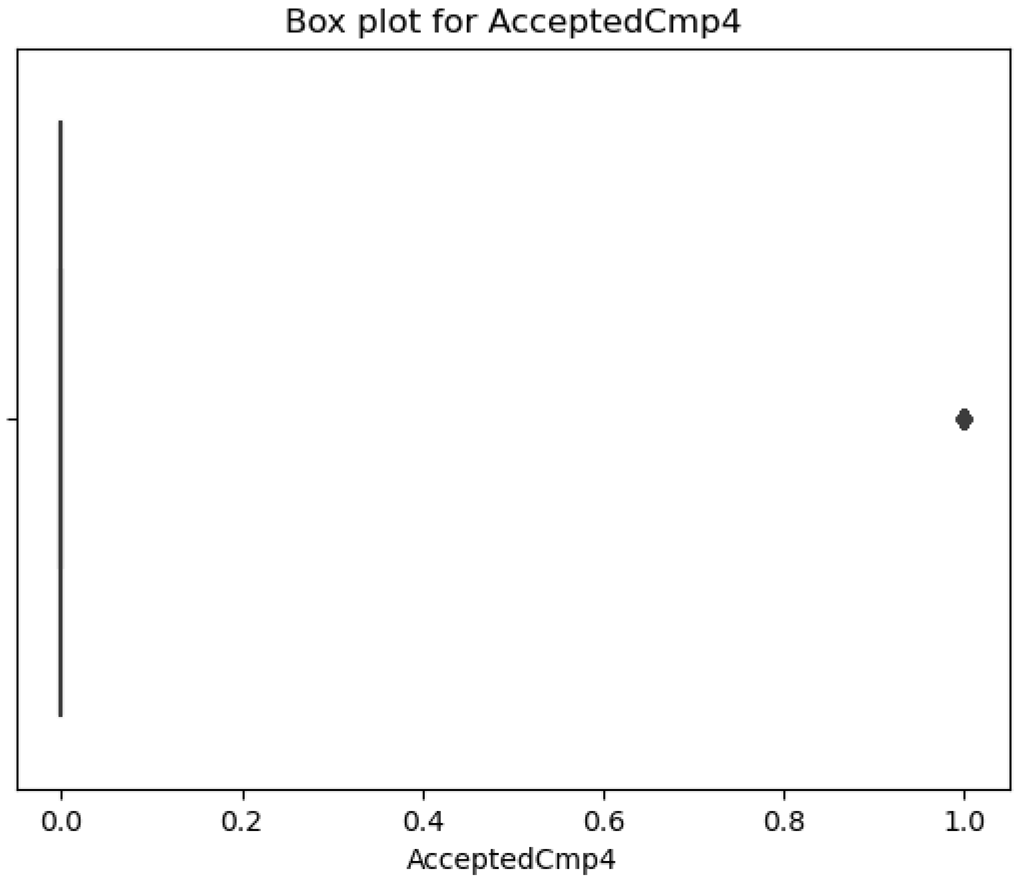

Box plot for NumCatalogPurchases
0
5
10
15
20
25
NumCatalogPurchases

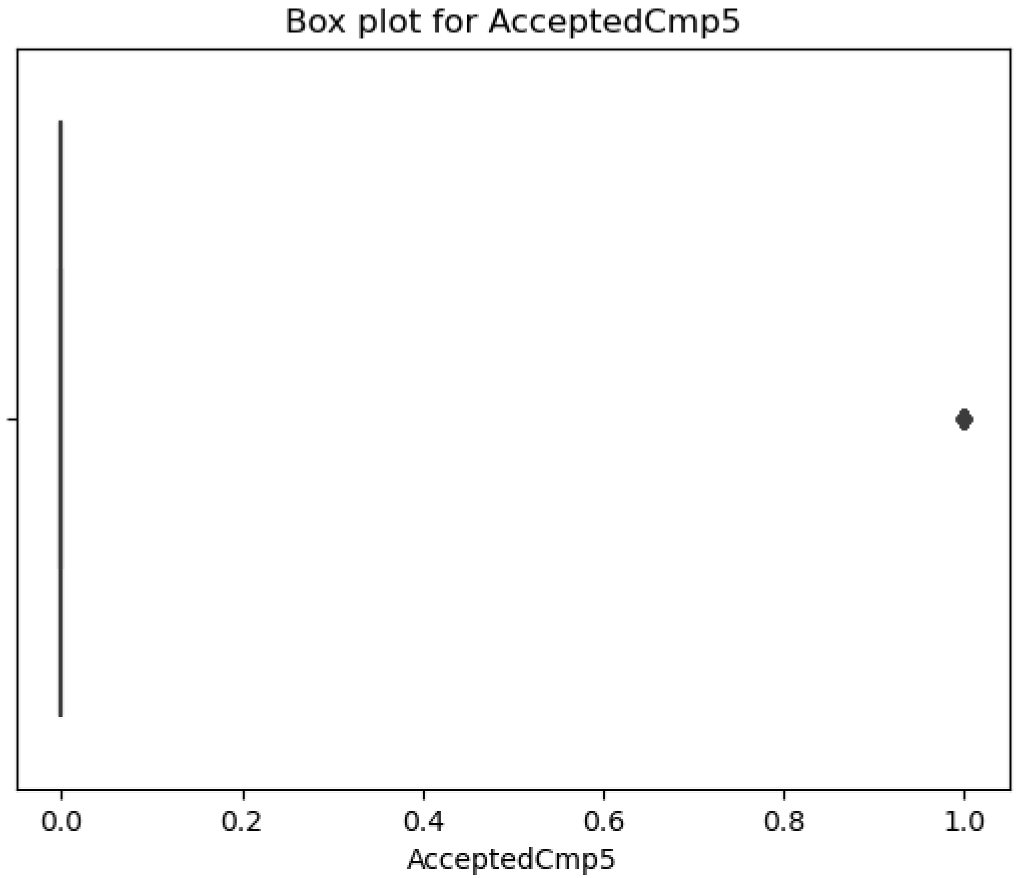


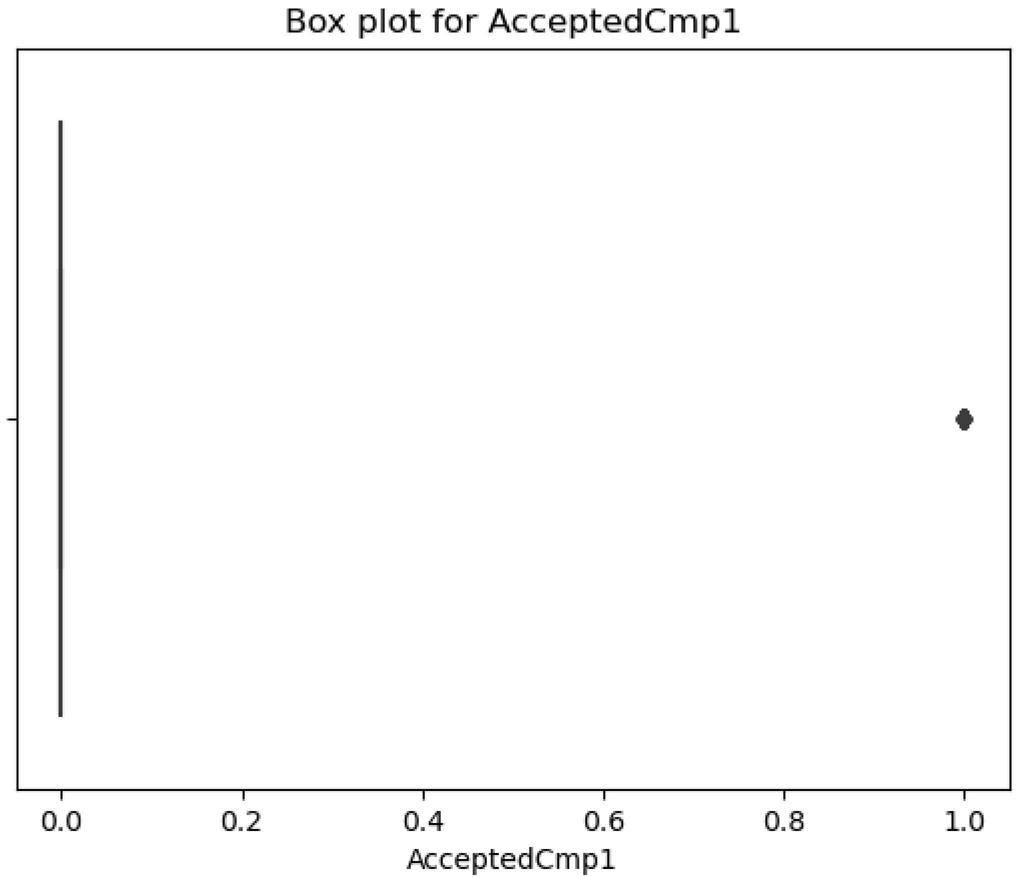
Box plot for NumStorePurchases
0
2
4
6
8
10
12
NumStorePurchases

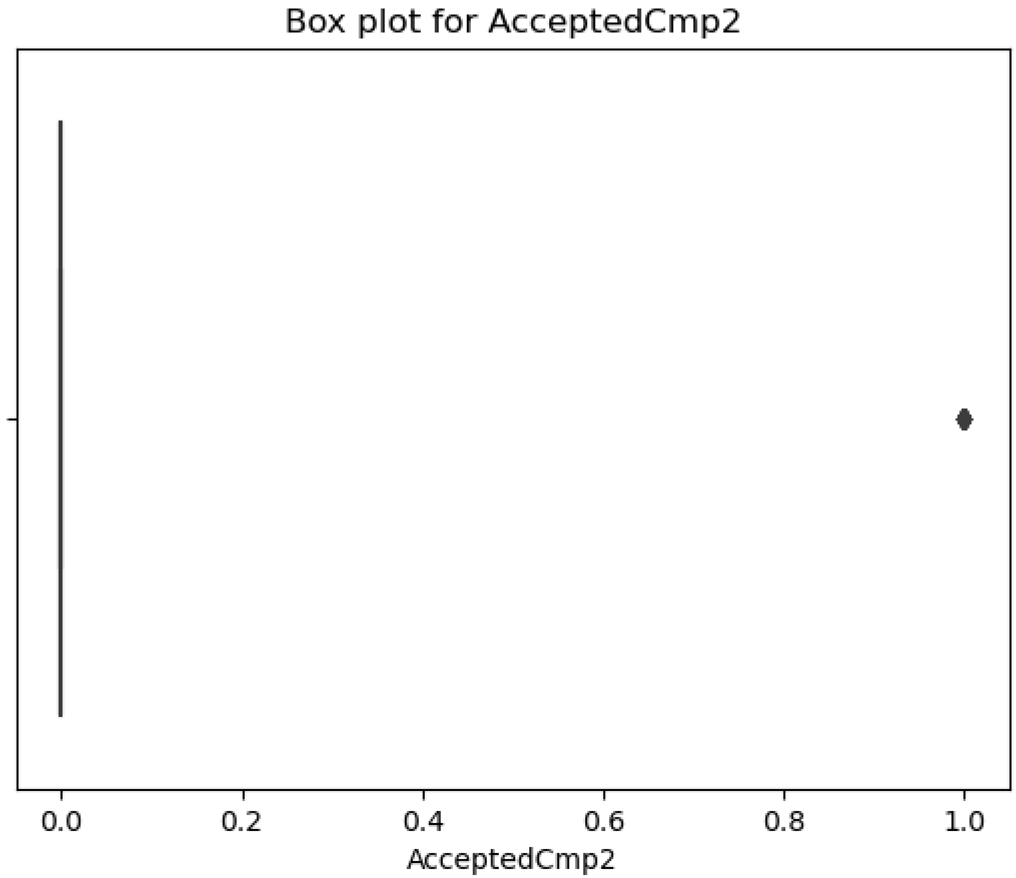

Box plot for NumWebVisitsMonth
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
NumWebVisitsMonth

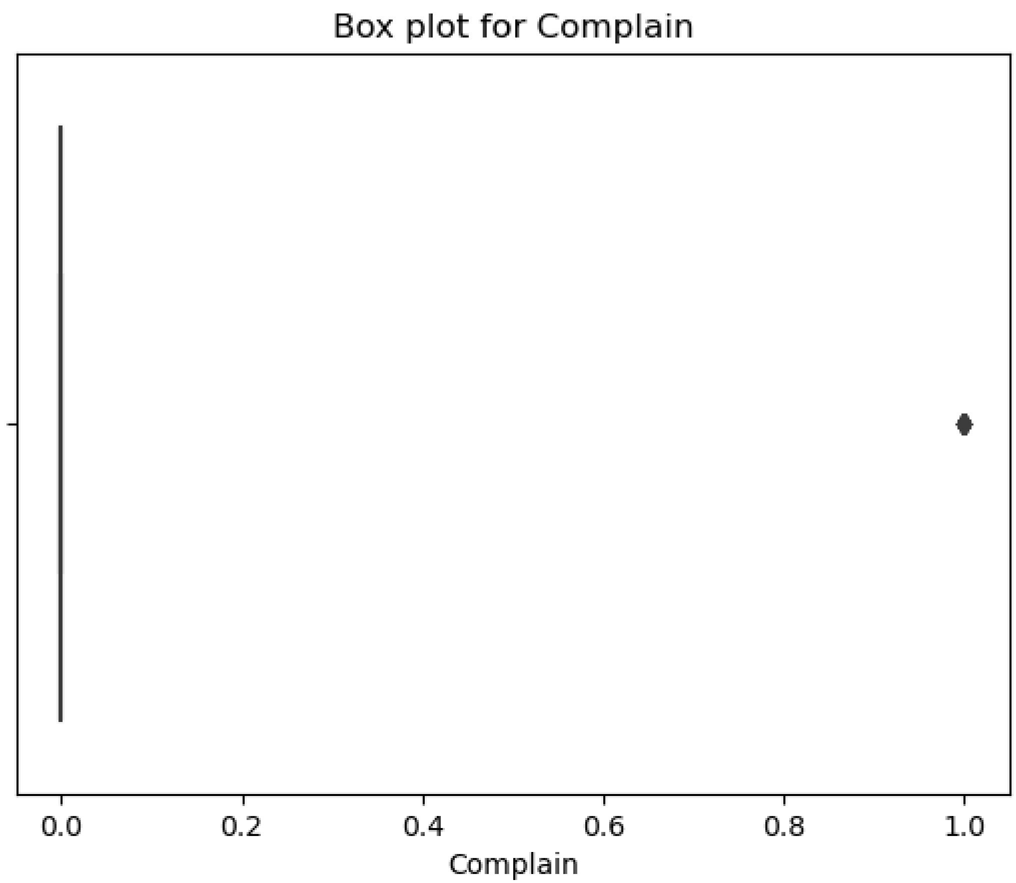

Box plot for AcceptedCmp3
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp3


Box plot for AcceptedCmp4
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp4


Box plot for AcceptedCmp5
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp5


Box plot for AcceptedCmp1
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp1


Box plot for AcceptedCmp2
0.0
0.2
0.4
0.6
0.8
1.0
AcceptedCmp2



Hypothesis Testing

Hypothesis:

- Null Hypothesis (H_0): Income is independent of education level.
- Alternative Hypothesis (H_1): Income is dependent on education level.

In [23]:
```
s,p = stats.f_oneway(*[camp[camp['Education'] == edu]['Income'].dropna() for edu in camp['Education'].unique()])
print(f"stat value : {s} p_value : {p}")
if p < 0.05:
    print("Reject null")
```

```
else:
    print("Falied to reject")
```

```
stat value : 38.39294760713925 p_value : 4.188444786493969e-31
Reject null
```

Hypothesis:

- Null Hypothesis (H_0): Couples and singles spend the same amount on wine.
- Alternative Hypothesis (H_1): There is a difference in spending on wine between couples and singles.

In [24]:
```
camp['Marital_Status_Group'] = camp['Marital_Status'].replace(
    {'Married': 'Couple', 'Together': 'Couple', 'Divorced': 'Alone', 'Single': 'Alone', 'Absurd': 'Alone', 'Widow': 'Alone', 'YOL
wine_couples = camp[camp['Marital_Status_Group'] == 'Couple']['MntWines']
wine_singles = camp[camp['Marital_Status_Group'] == 'Alone']['MntWines']
s,p = stats.ttest_ind(wine_couples, wine_singles, equal_var=False)

print(f"stat value : {s} p_value : {p}")

if p < 0.05:
    print("Reject null")
else:
    print("Falied to reject")
```

```
stat value : -0.2711337908368919 p_value : 0.7863223090103292
Falied to reject
```

Hypothesis:

- Null Hypothesis (H_0): Customers with lower income are not more likely to accept campaigns.
- Alternative Hypothesis (H_1): Customers with lower income are more likely to accept campaigns.

In [25]:
```
median_income = camp['Income'].median()
camp['Income_Bracket'] = camp['Income'].apply(lambda x: 'Below Median' if x < median_income else 'Above Median')
camp['Ever_Accepted_Campaign'] = (camp[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis

contingency_table = pd.crosstab(camp['Income_Bracket'], camp['Ever_Accepted_Campaign'])
chi,p,_,_ = stats.chi2_contingency(contingency_table)
print(f'Chi-square value: {chi}, p-value: {p}')
alpha = 0.05
if p<alpha:
    print("Reject Ho")
```

```python
else:
    print("Failed to reject Ho")
```

```
Chi-square value: 140.11555527497433, p-value: 2.5115657237830455e-32
Reject Ho
```

Insights:

Education & Spending:

- Graduates and PhD holders are key spenders, with PhDs spending more on wine and graduates on gold.Except for basic-educated customers, others prefer spending on meat and fish products.

Marital Status & Buying Behavior:

- Couples and singles show no significant difference in spending on wine.
- Most customers are married or living together, indicating a family-oriented customer base.

Product Preferences:

- Majority buy one or two products, suggesting a skewed distribution toward fewer purchases.
- Customers purchasing three products tend to buy directly from stores.

Income & Campaign Acceptance:

- Income is dependent on education level.
- Lower-income customers are more likely to accept campaigns, indicating price sensitivity.

Geographic & Offer Insights:

- Most purchases come from Spanish customers.
- Catalog purchases are less preferred, while direct offers seem effective initially but drop significantly by Campaign 5.
- Complaints are minimal, showing general satisfaction.

Recommendations:

Targeted Campaigns:

- Focus campaigns on lower-income groups and highlight budget-friendly products to increase acceptance.
- Design promotions for graduates and PhD holders, leveraging their spending habits (wine and gold).

Product Strategy:

- Diversify product offerings to encourage multi-product purchases.
- Highlight meat, fish, and wine in marketing campaigns, especially toward educated and family-oriented segments.

Revisit Campaigns:

- Evaluate why acceptance drops by Campaign 5 and adjust the strategy to maintain engagement.
- Reduce emphasis on catalog sales and promote digital or store offers.

Localization:

- Tailor marketing specifically for the Spanish demographic to capitalize on the existing customer base.

In [ ]:

```

```

In [ ]:

```

```