E-commerce Python EDA: Shopping

In [330…
```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading a Dataset

1. Loading the data
2. Analyzing the data
3. Checking for duplicates
4. Missing Value Calculation

In [331…
```
df = pd.read_csv('/content/shopping - shopping.csv')
```

In [332…
```
df.shape
```

Out[332]:
```
(12330, 18)
```

In [333…
```
df.head()
```

Out[333]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated
0	0	0.0	0	0.0	1
1	0	0.0	0	0.0	2
2	0	0.0	0	0.0	1
3	0	0.0	0	0.0	2
4	0	0.0	0	0.0	10

In [334…
```
df.tail()
```

Out[334]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRela
12325	3	145.0	0	0.0	
12326	0	0.0	0	0.0	
12327	0	0.0	0	0.0	
12328	4	75.0	0	0.0	
12329	0	0.0	0	0.0	

In [335…
```
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
```

In [336…

```
df.nunique()
```

Out[336]:

	0
Administrative	27
Administrative_Duration	3335
Informational	17
Informational_Duration	1258
ProductRelated	311
ProductRelated_Duration	9551
BounceRates	1872
ExitRates	4777
PageValues	2704
SpecialDay	6
Month	10
OperatingSystems	8
Browser	13
Region	9
TrafficType	20
VisitorType	3
Weekend	2
Revenue	2

dtype: int64

In [337…
```
df.duplicated().sum()
```

Out[337]:
```
125
```

In [338…
```
df.drop_duplicates(inplace=True)
```

In [339…
```
df.isnull().sum()
```

Out[339]:

	0
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

In [340…
```
df.describe().T
```

Out[340]:

	count	mean	std	min	25%	50%	7
Administrative	12205.0	2.338878	3.330436	0.0	0.000000	1.000000	4.000
Administrative_Duration	12205.0	81.646331	177.491845	0.0	0.000000	9.000000	94.700
Informational	12205.0	0.508726	1.275617	0.0	0.000000	0.000000	0.000
Informational_Duration	12205.0	34.825454	141.424807	0.0	0.000000	0.000000	0.000
ProductRelated	12205.0	32.045637	44.593649	0.0	8.000000	18.000000	38.000
ProductRelated_Duration	12205.0	1206.982457	1919.601400	0.0	193.000000	608.942857	1477.154
BounceRates	12205.0	0.020370	0.045255	0.0	0.000000	0.002899	0.016
ExitRates	12205.0	0.041466	0.046163	0.0	0.014231	0.025000	0.048
PageValues	12205.0	5.949574	18.653671	0.0	0.000000	0.000000	0.000
SpecialDay	12205.0	0.061942	0.199666	0.0	0.000000	0.000000	0.000
OperatingSystems	12205.0	2.124211	0.906823	1.0	2.000000	2.000000	3.000
Browser	12205.0	2.357804	1.710114	1.0	2.000000	2.000000	2.000
Region	12205.0	3.153298	2.402340	1.0	1.000000	3.000000	4.000
TrafficType	12205.0	4.073904	4.016654	1.0	2.000000	2.000000	4.000

In [341…
```
df.describe(include=['object', 'bool']).T
```

Out[341]:

	count	unique	top	freq
Month	12205	10	May	3329
VisitorType	12205	3	Returning_Visitor	10431
Weekend	12205	2	False	9346
Revenue	12205	2	False	10297

In [342…
```
# Converting columns to appropriate data types
cat_cols = ['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType',
                    'VisitorType', 'Weekend', 'SpecialDay']

for cat in cat_cols:
  df[cat] = df[cat].astype('object')

df['Revenue'] = df['Revenue'].astype('int')
```

In [343…
```
num_cols = df.dtypes != 'object'
num_cols = list(num_cols[num_cols].index)
num_cols.remove('Revenue')

cat_cols = df.dtypes == 'object'
cat_cols = list(cat_cols[cat_cols].index)

target = 'Revenue'
```

In [344…
```
num_cols
```

Out[344]:
```
['Administrative',
 'Administrative_Duration',
 'Informational',
 'Informational_Duration',
 'ProductRelated',
 'ProductRelated_Duration',
 'BounceRates',
 'ExitRates',
 'PageValues']
```

Non-Graphical Analysis

Helper Functions

In [345…
```
# Function to print basic useful details for a given column
def get_column_details(df,column):
    print("Details of",column,"column")

    #DataType of column
    print("\nDataType: ",df[column].dtype)

    #Check if null values are present
    count_null = df[column].isnull().sum()
    if count_null==0:
        print("\nThere are no null values")
    elif count_null>0:
        print("\nThere are ",count_null," null values")

    #Get Number of Unique Values
    print("\nNumber of Unique Values: ",df[column].nunique())

    #Get Distribution of Column
    print("\nDistribution of column:\n")
    print(df[column].value_counts())
```

In [346…
```
# Function to calculate outlier percentage for each numerical column
def calculate_outlier_percentage(df):
    outlier_percentage = {}

    for col in df.select_dtypes(include=['float64', 'int64']).columns:
        # Calculate Q1 (25th percentile) and Q3 (75th percentile)
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)

        # Calculate IQR
        IQR = Q3 - Q1

        # Determine outlier boundaries
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Count outliers
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
        outlier_count = outliers.shape[0]

        # Calculate percentage of outliers
        outlier_percentage[col] = (outlier_count / df.shape[0]) * 100

    return outlier_percentage

# Calculate and print outlier percentages
outlier_percentages = calculate_outlier_percentage(df)
```

```
print("Outlier Percentages for Each Numerical Column:")
print(outlier_percentages)
```

```
Outlier Percentages for Each Numerical Column:
{'Administrative': 3.3101188037689475, 'Administrative_Duration': 9.41417451863990
2, 'Informational': 21.556739041376485, 'Informational_Duration': 19.7050389184760
35, 'ProductRelated': 8.250716919295371, 'ProductRelated_Duration': 7.791888570258
091, 'BounceRates': 11.700122900450635, 'ExitRates': 10.856206472757068, 'PageValu
es': 22.36788201556739, 'Revenue': 15.632937320770177}
```

In [347…

```
df.columns
```

Out[347]:

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

In [348…

```
for i in df.columns:
  get_column_details(df,i)
```

```
Details of Administrative column

DataType:  int64

There are no null values

Number of Unique Values:  27

Distribution of column:

Administrative
0     5643
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
24       4
22       4
23       3
21       2
20       2
27       1
26       1
Name: count, dtype: int64
Details of Administrative_Duration column

DataType:  float64

There are no null values

Number of Unique Values:  3335

Distribution of column:

Administrative_Duration
0.000000      5778
4.000000        56
5.000000        53
7.000000        45
11.000000       42
               ...
68.014286        1
362.300000       1
90.700000        1
760.900000       1
150.357143       1
Name: count, Length: 3335, dtype: int64
Details of Informational column
```

```
DataType:  int64

There are no null values

Number of Unique Values:  17

Distribution of column:

Informational
0     9574
1     1041
2      728
3      380
4      222
5       99
6       78
7       36
9       15
8       14
10       7
12       5
14       2
16       1
11       1
24       1
13       1
Name: count, dtype: int64
Details of Informational_Duration column

DataType:  float64

There are no null values

Number of Unique Values:  1258

Distribution of column:

Informational_Duration
0.00      9800
9.00        33
7.00        26
10.00       26
6.00        26
           ...
246.80       1
274.00       1
13.40        1
223.15       1
211.25       1
Name: count, Length: 1258, dtype: int64
Details of ProductRelated column

DataType:  int64

There are no null values

Number of Unique Values:  311

Distribution of column:

ProductRelated
1      504
3      458
2      458
```

```
4      404
6      396
      ...
243      1
409      1
262      1
414      1
192      1
Name: count, Length: 311, dtype: int64
Details of ProductRelated_Duration column

DataType:  float64

There are no null values

Number of Unique Values:  9551

Distribution of column:

ProductRelated_Duration
0.000000      630
17.000000      21
11.000000      17
8.000000       17
15.000000      16
              ...
964.070513      1
593.507143      1
831.388889      1
922.208333      1
346.000000      1
Name: count, Length: 9551, dtype: int64
Details of BounceRates column

DataType:  float64

There are no null values

Number of Unique Values:  1872

Distribution of column:

BounceRates
0.000000    5518
0.200000     575
0.066667     134
0.028571     115
0.050000     113
            ...
0.079279       1
0.006723       1
0.013527       1
0.074419       1
0.011149       1
Name: count, Length: 1872, dtype: int64
Details of ExitRates column

DataType:  float64

There are no null values

Number of Unique Values:  4777

Distribution of column:
```

```
ExitRates
0.200000    585
0.100000    338
0.050000    329
0.033333    291
0.066667    267
           ...
0.021816      1
0.015787      1
0.010302      1
0.014534      1
0.029031      1
Name: count, Length: 4777, dtype: int64
Details of PageValues column

DataType:  float64

There are no null values

Number of Unique Values:  2704

Distribution of column:

PageValues
0.000000     9475
53.988000       6
42.293068       3
59.988000       2
16.158558       2
              ...
6.673696        1
6.094324        1
28.253955       1
16.090650       1
12.241717       1
Name: count, Length: 2704, dtype: int64
Details of SpecialDay column

DataType:  object

There are no null values

Number of Unique Values:  6

Distribution of column:

SpecialDay
0.0    10956
0.6      350
0.8      324
0.4      243
0.2      178
1.0      154
Name: count, dtype: int64
Details of Month column

DataType:  object

There are no null values

Number of Unique Values:  10

Distribution of column:
```

```
Month
May     3329
Nov     2982
Mar     1860
Dec     1706
Oct      549
Sep      448
Aug      433
Jul      432
June     285
Feb      181
Name: count, dtype: int64
Details of OperatingSystems column

DataType:  object

There are no null values

Number of Unique Values:  8

Distribution of column:

OperatingSystems
2    6541
1    2549
3    2530
4     478
8      75
6      19
7       7
5       6
Name: count, dtype: int64
Details of Browser column

DataType:  object

There are no null values

Number of Unique Values:  13

Distribution of column:

Browser
2     7883
1     2427
4      731
5      465
6      174
10     163
8      135
3      105
13      56
7       49
12      10
11       6
9        1
Name: count, dtype: int64
Details of Region column

DataType:  object

There are no null values
```

```
Number of Unique Values:  9

Distribution of column:

Region
1    4714
3    2379
4    1171
2    1128
6     801
7     758
9     505
8     431
5     318
Name: count, dtype: int64
Details of TrafficType column

DataType:  object

There are no null values

Number of Unique Values:  20

Distribution of column:

TrafficType
2     3911
1     2388
3     2013
4     1066
13     728
10     450
6      443
8      343
5      260
11     247
20     193
9       41
7       40
15      37
19      17
14      13
18      10
16       3
12       1
17       1
Name: count, dtype: int64
Details of VisitorType column

DataType:  object

There are no null values

Number of Unique Values:  3

Distribution of column:

VisitorType
Returning_Visitor    10431
New_Visitor           1693
Other                   81
Name: count, dtype: int64
Details of Weekend column
```

```
DataType:  object

There are no null values

Number of Unique Values:  2

Distribution of column:

Weekend
False    9346
True     2859
Name: count, dtype: int64
Details of Revenue column

DataType:  int64

There are no null values

Number of Unique Values:  2

Distribution of column:

Revenue
0    10297
1     1908
Name: count, dtype: int64
```

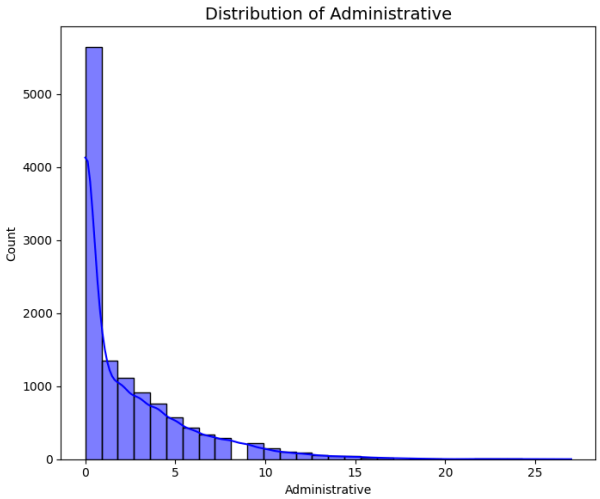
EDA

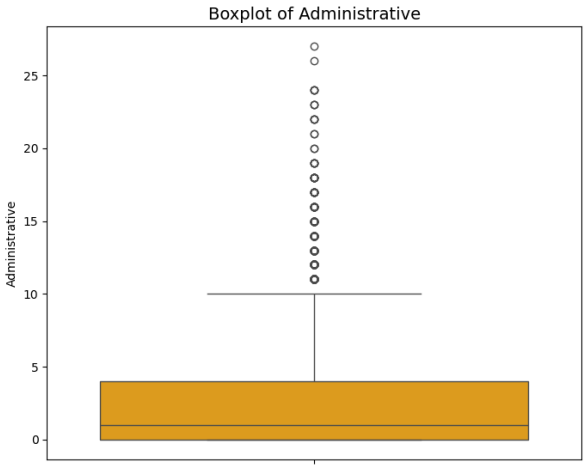
1. Uni-variate Analysis
2. Bi-Variate Analysis
3. Multi-variate Analysis

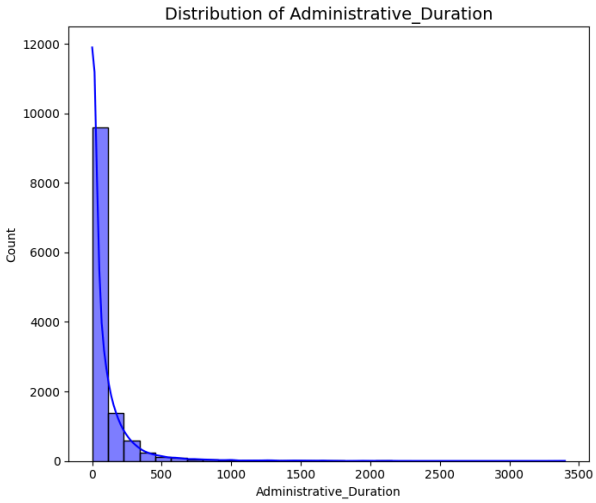
Uni-Variate Analysis

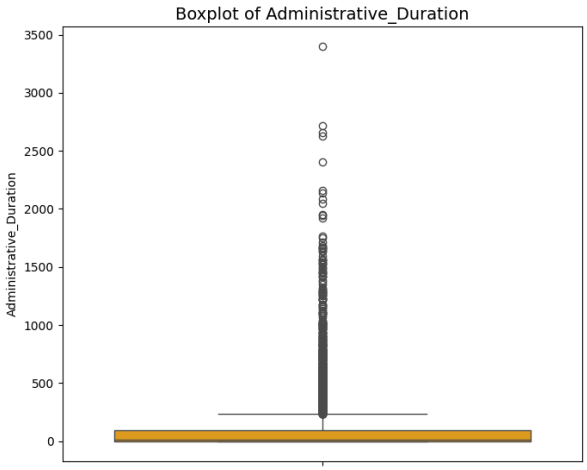
In [349…

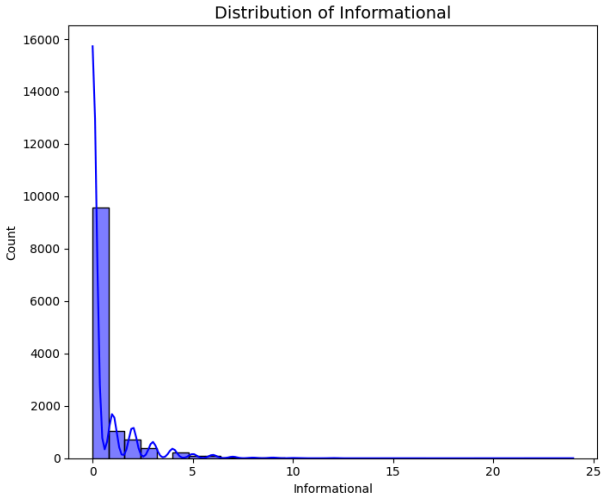
```
# Univariate analysis for each numerical column
for col in num_cols:
    plt.figure(figsize=(14, 6))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(df[col], kde=True, bins=30, color='blue')
    plt.title(f'Distribution of {col}', fontsize=14)

    # Boxplot"
    plt.subplot(1, 2, 2)
    sns.boxplot(y=df[col], color='orange')
    plt.title(f'Boxplot of {col}', fontsize=14)

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

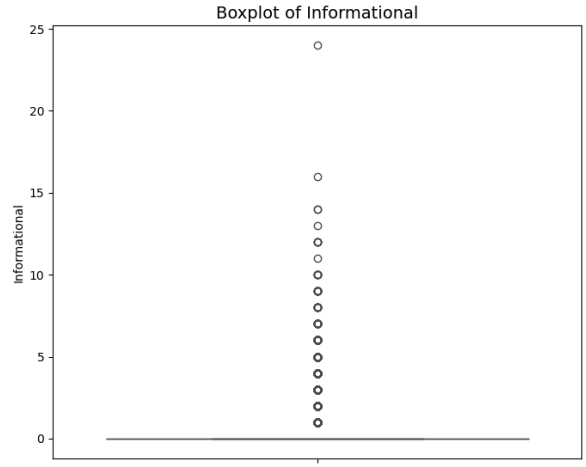

Distribution of Administrative
Count
Administrative

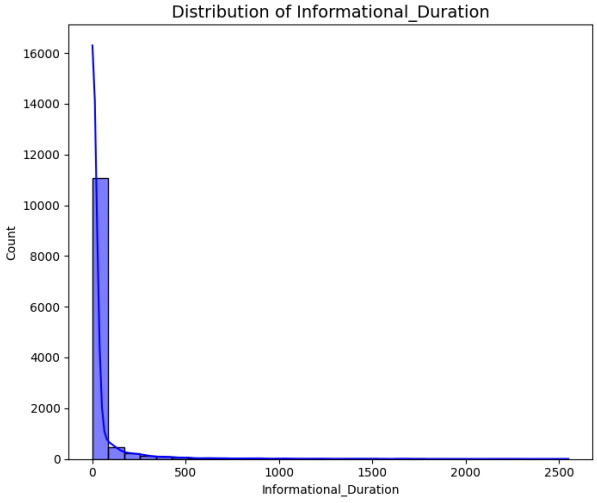

Boxplot of Administrative
Administrative

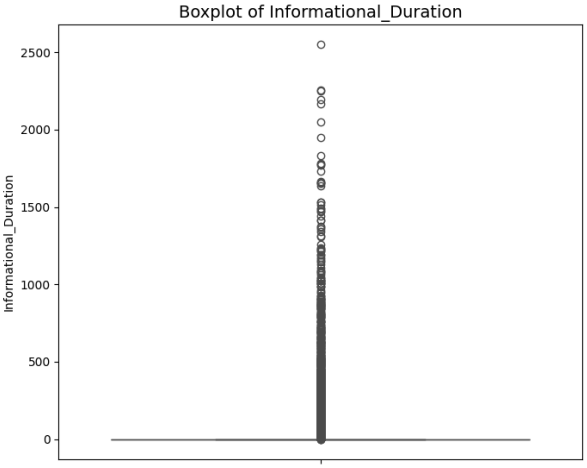

Distribution of Administrative_Duration
Count
Administrative_Duration

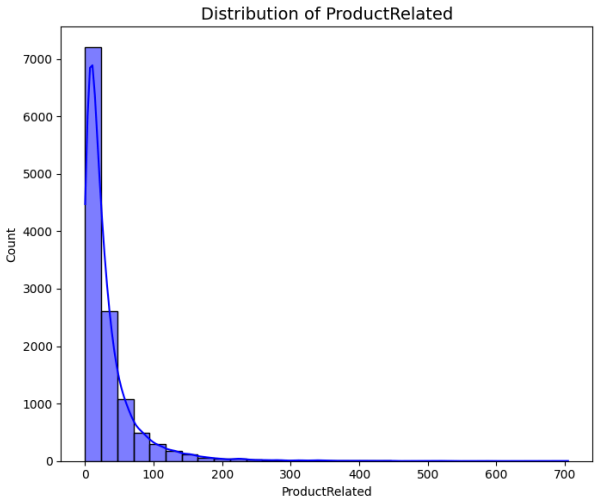

Boxplot of Administrative_Duration
Administrative_Duration

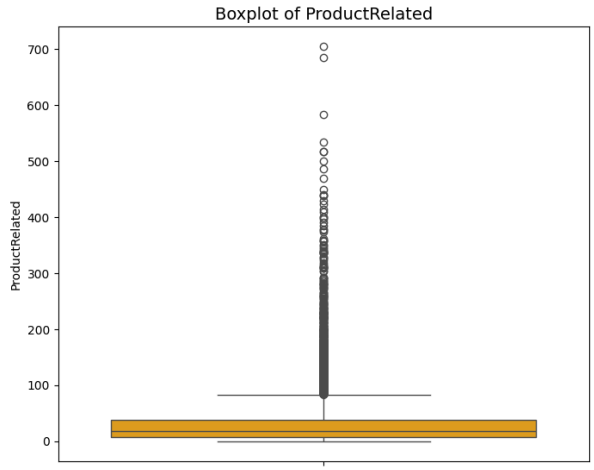

Distribution of Informational
Count
Informational

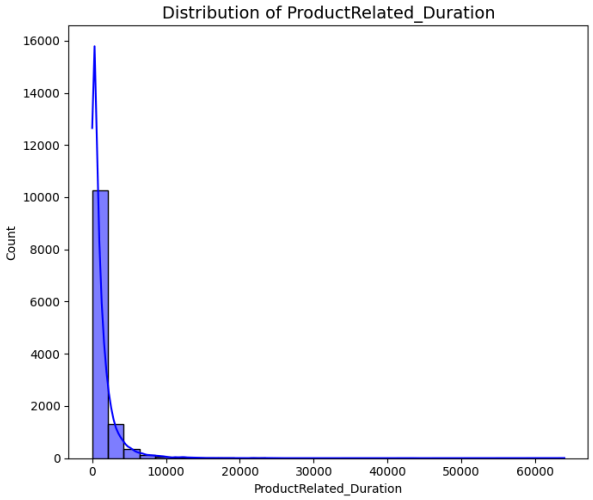

Boxplot of Informational
Informational

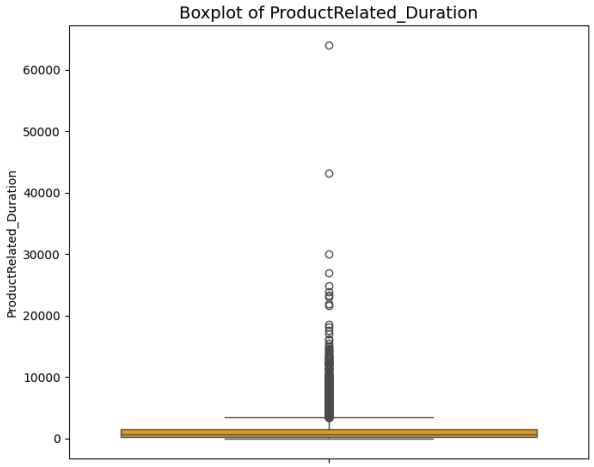

Distribution of Informational_Duration
Count
Informational_Duration

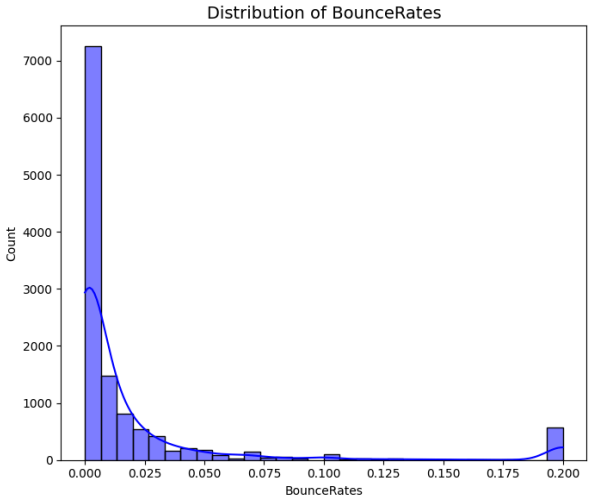

Boxplot of Informational_Duration
Informational_Duration

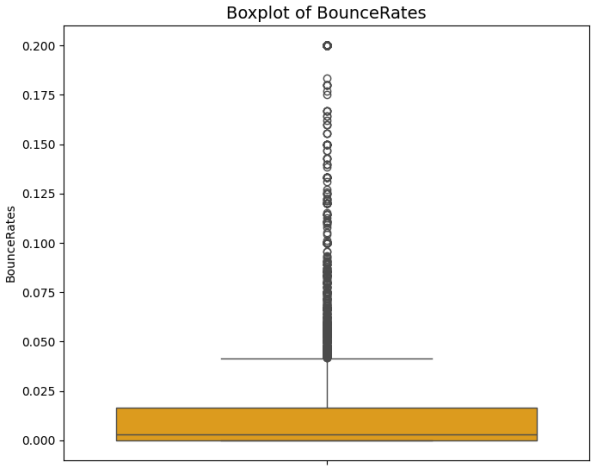

Distribution of ProductRelated
Count
ProductRelated

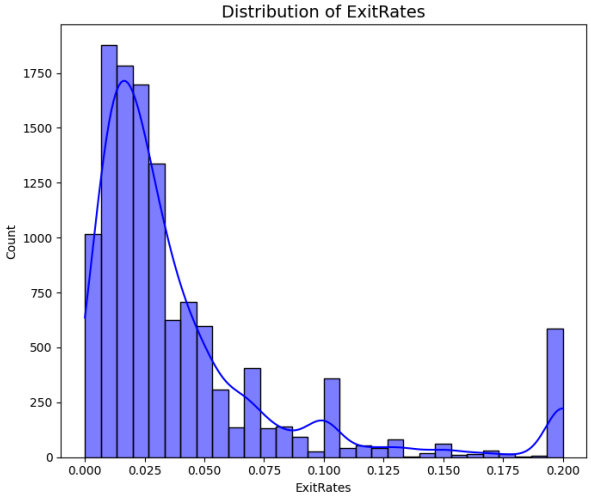

Boxplot of ProductRelated
ProductRelated

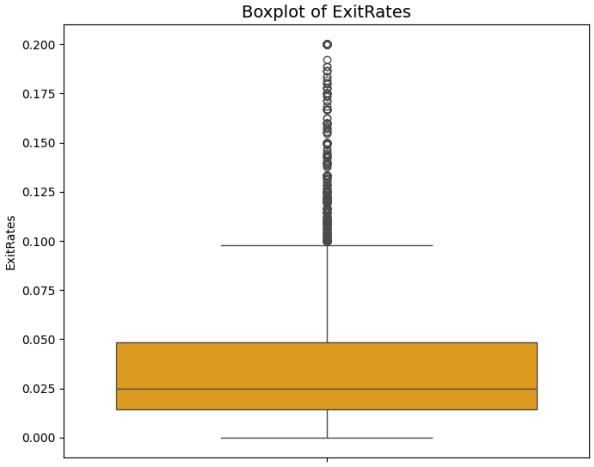

Distribution of ProductRelated_Duration
Count
ProductRelated_Duration

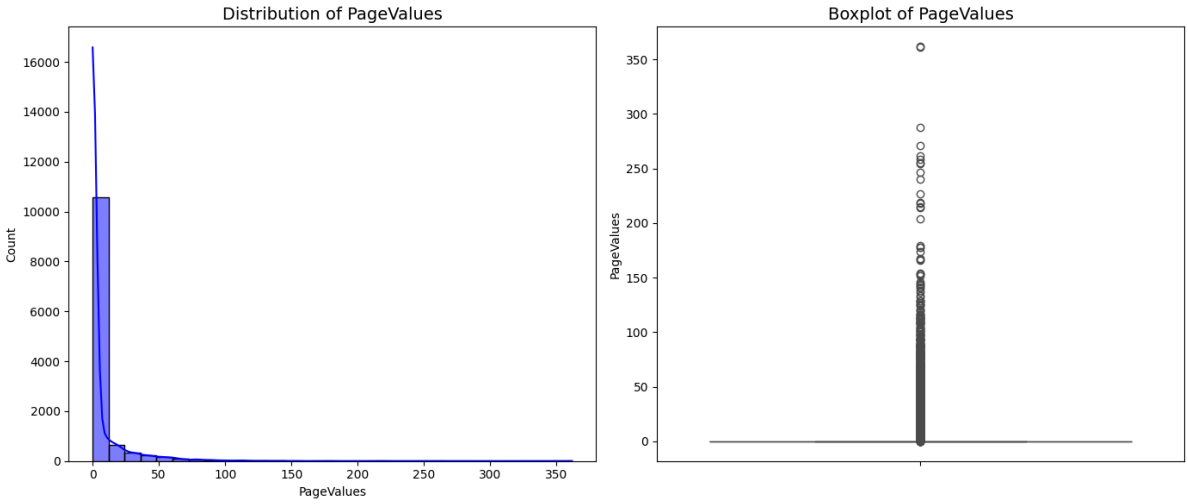

Boxplot of ProductRelated_Duration
ProductRelated_Duration


Distribution of BounceRates
Count
BounceRates


Boxplot of BounceRates
BounceRates


Distribution of ExitRates
Count
ExitRates


Boxplot of ExitRates
ExitRates



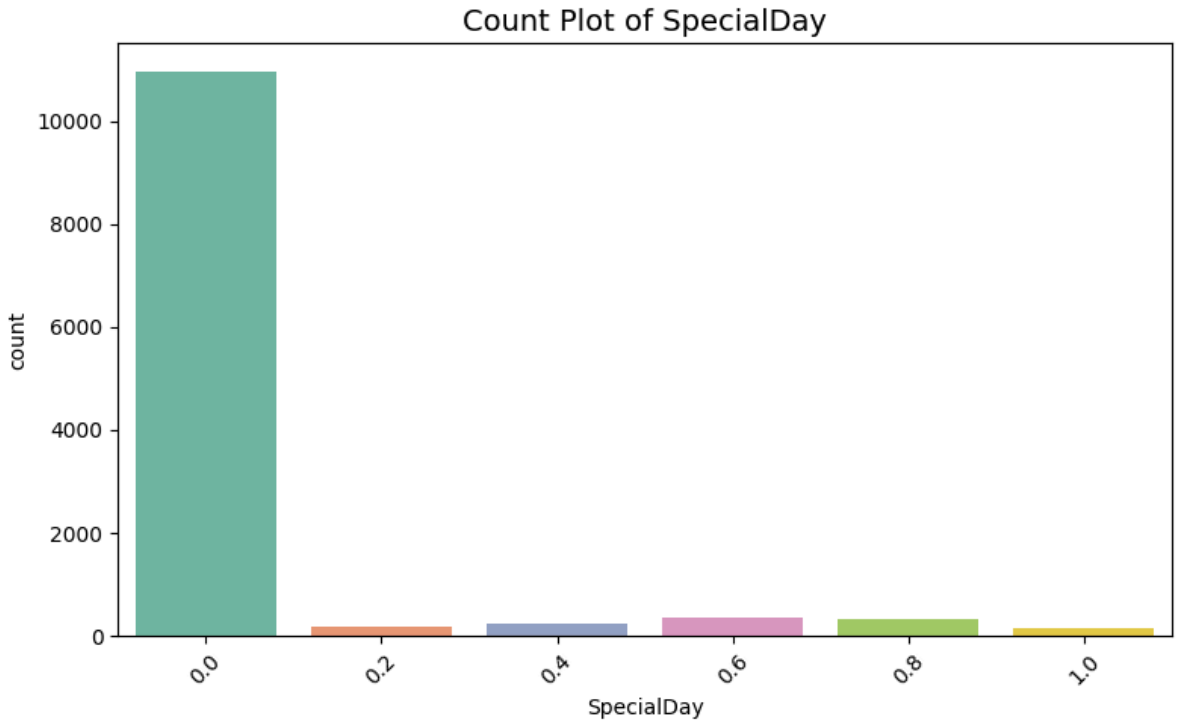
Not Handling Outliers:

- If your goal is to predict something like whether a user will generate revenue or whether a session will convert, having lots of zeros might actually be informative, especially if they correlate with the target variable (Revenue).

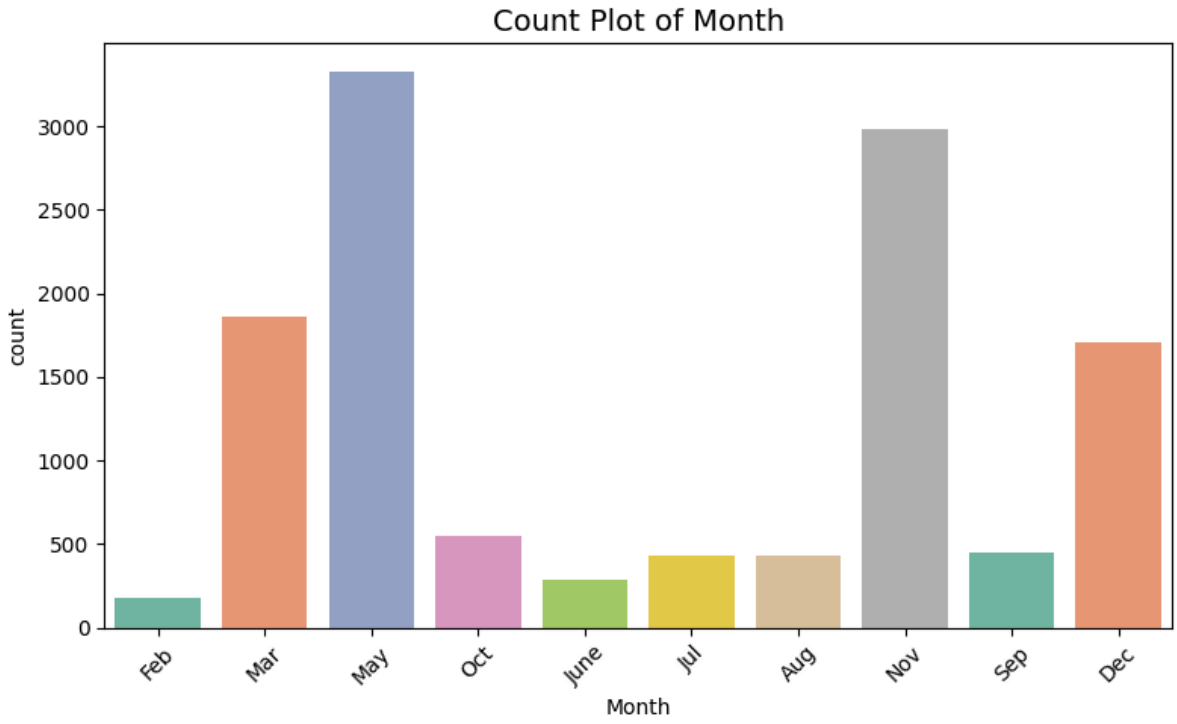
In [350…

```
# Univariate analysis for each categorical column
for col in cat_cols:
    plt.figure(figsize=(8, 5))

    # Count plot for each categorical column
    sns.countplot(x=df[col], palette='Set2')

    plt.title(f'Count Plot of {col}', fontsize=14)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

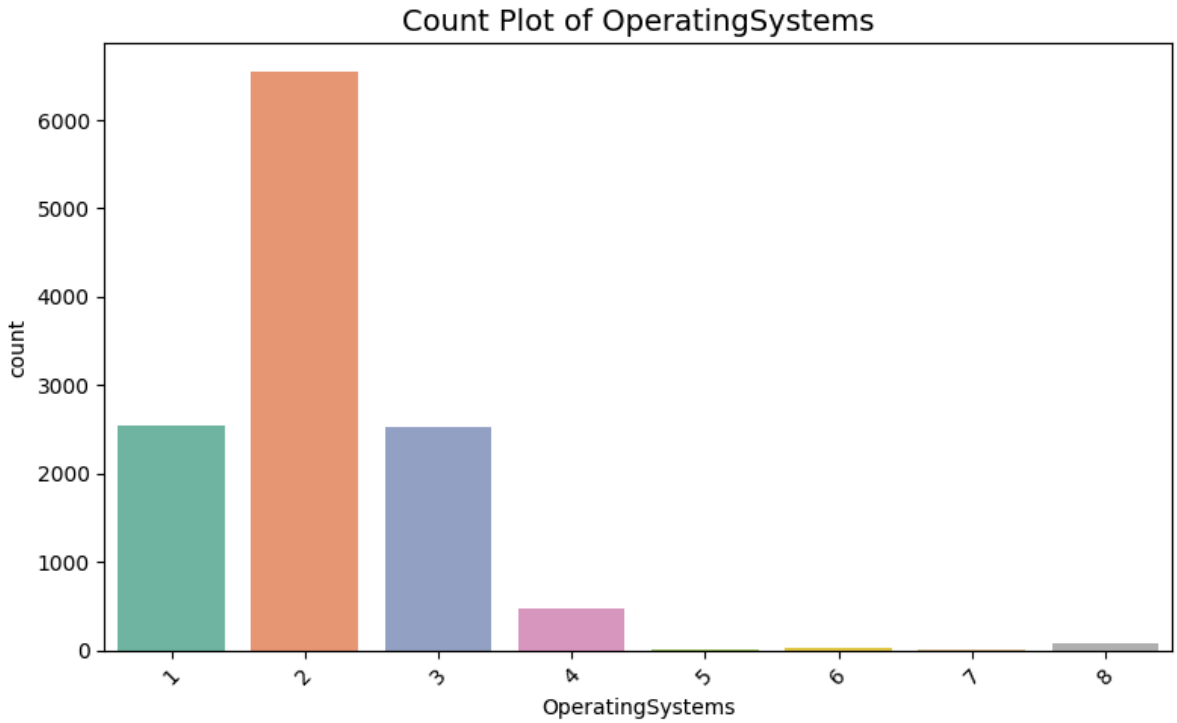
```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```



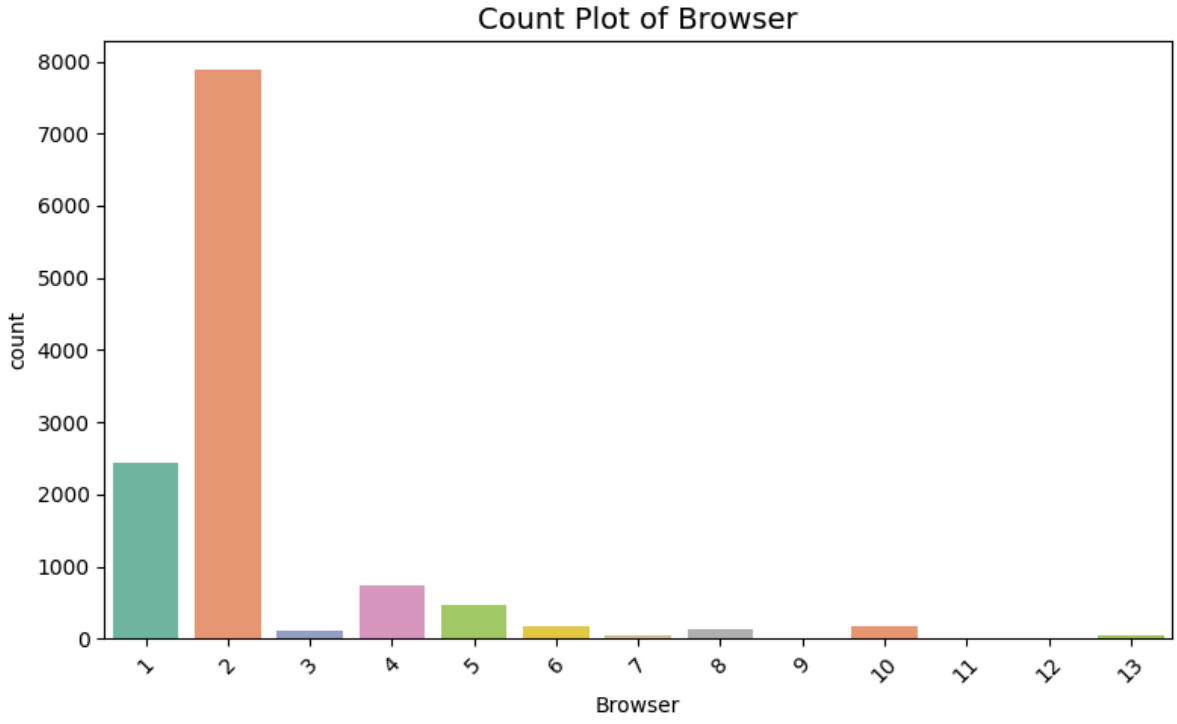

```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```



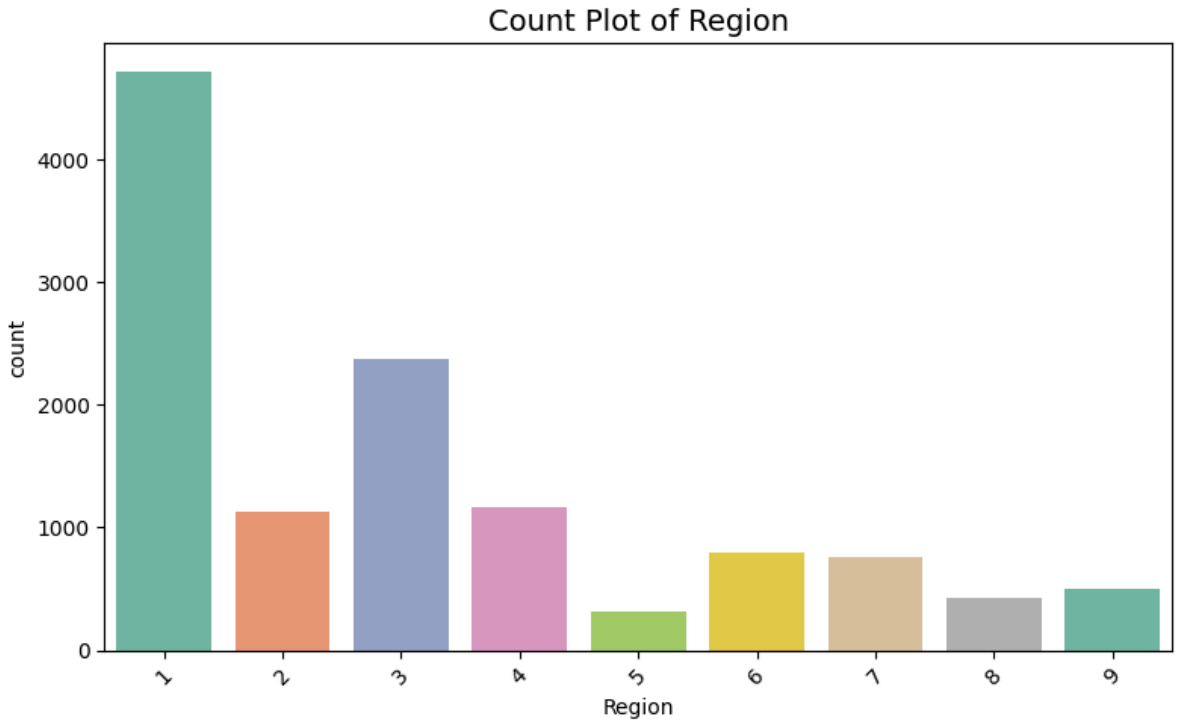

```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```



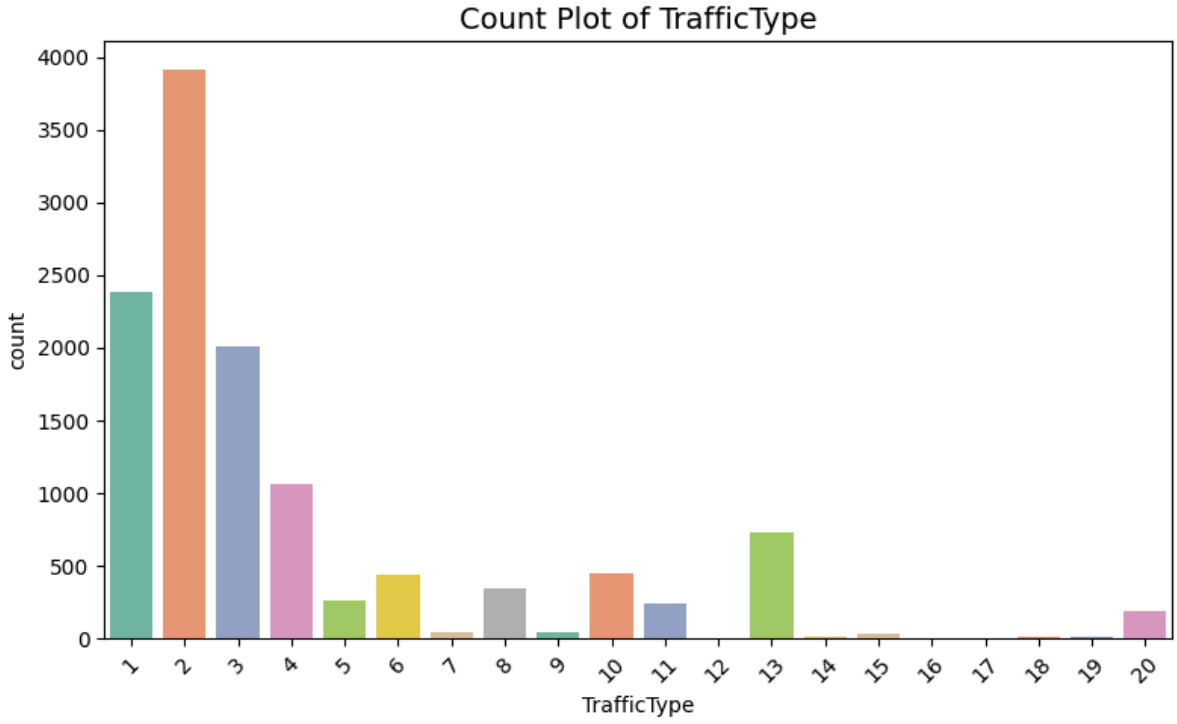

```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```



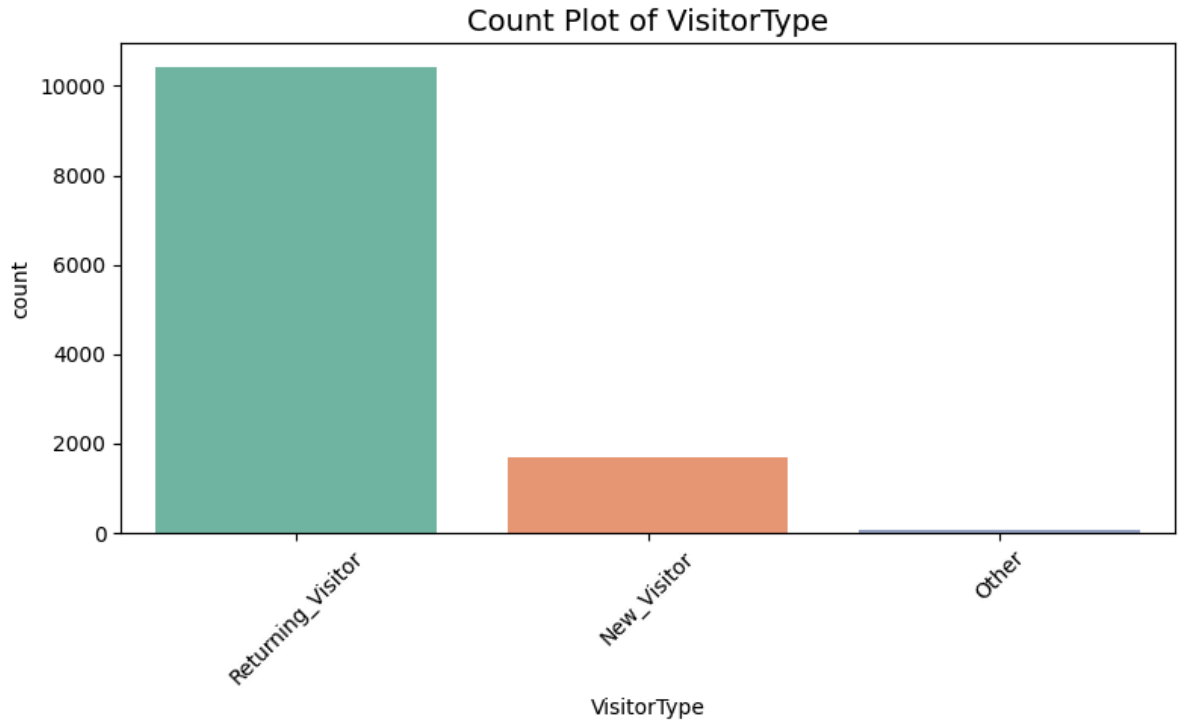

```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```



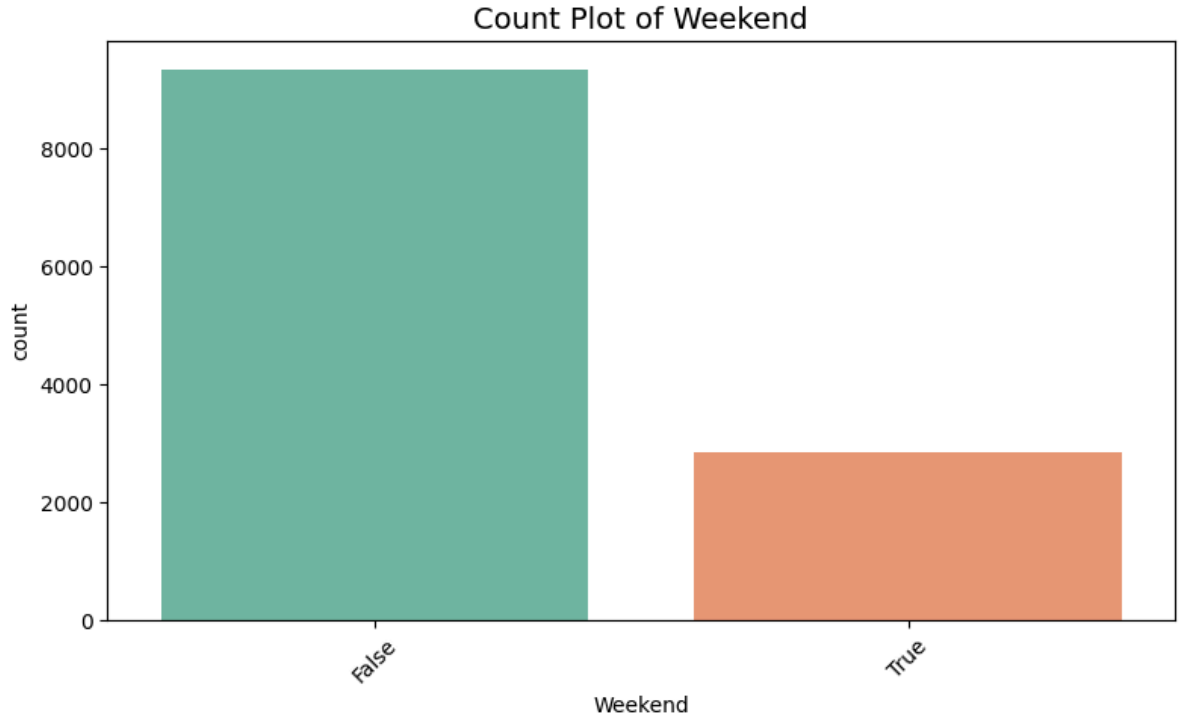

```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```




```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```




```
<ipython-input-350-b84c7c645ff8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=df[col], palette='Set2')
```

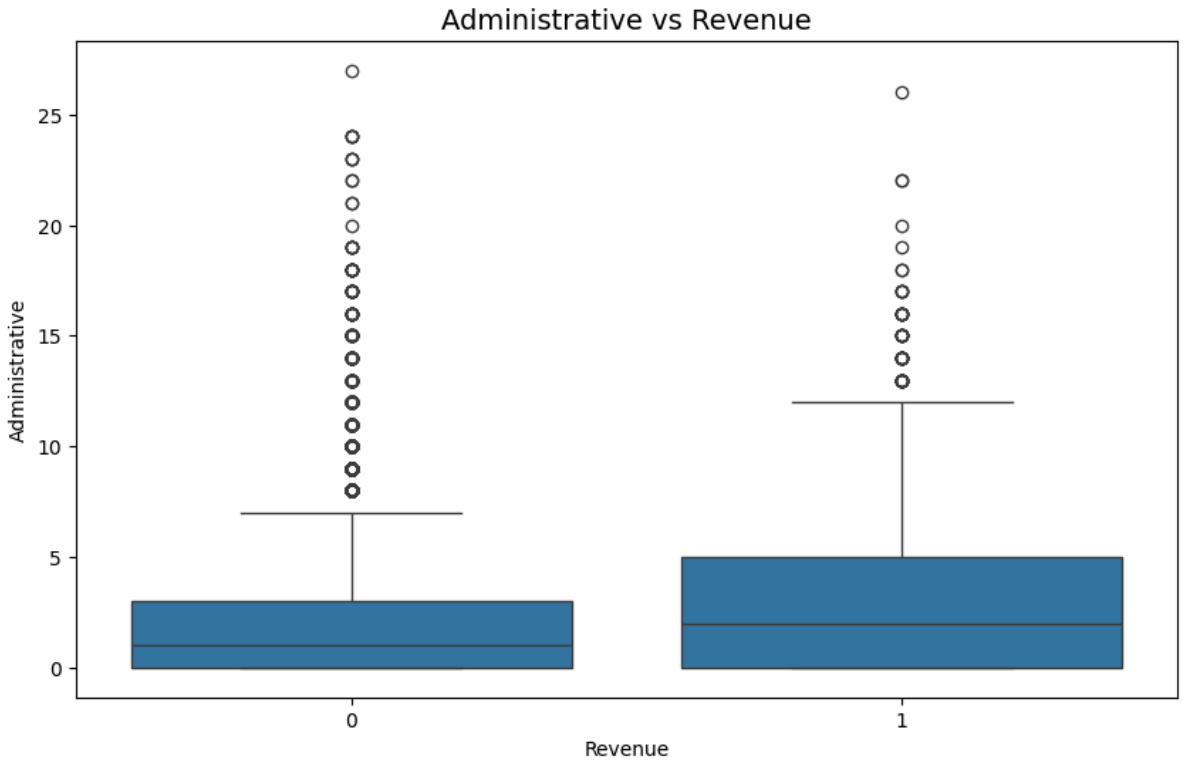


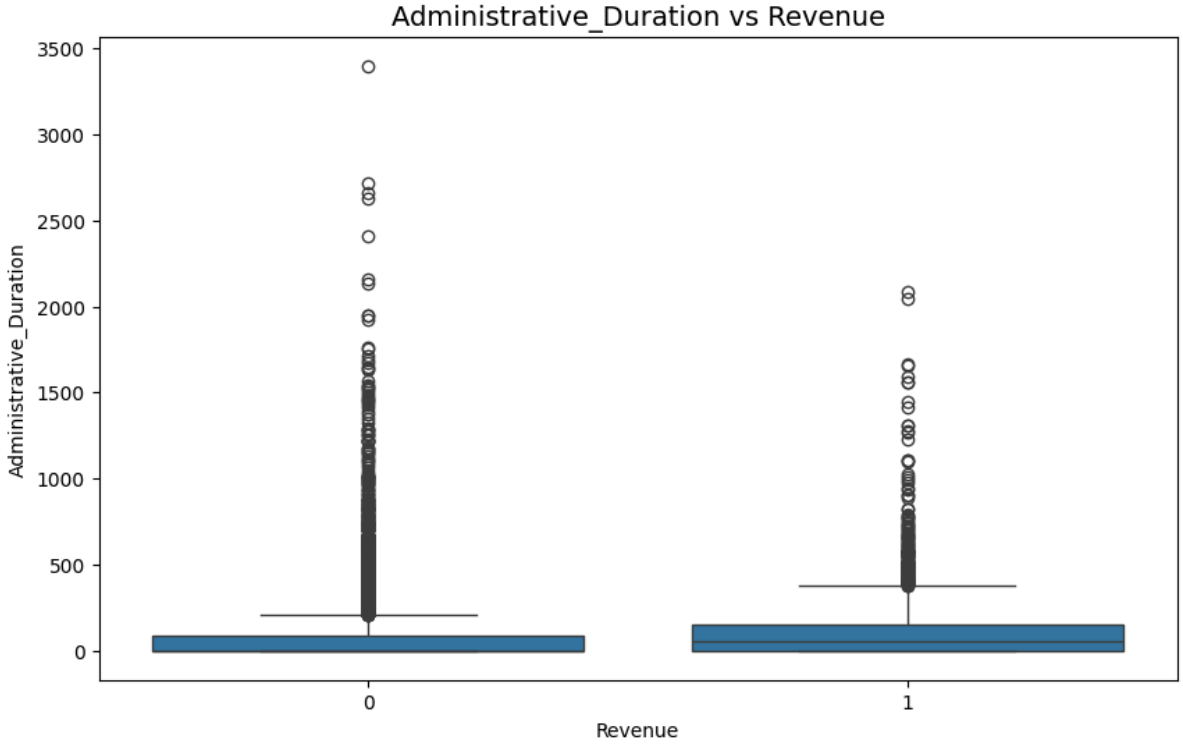
Bi-Variate Analysis

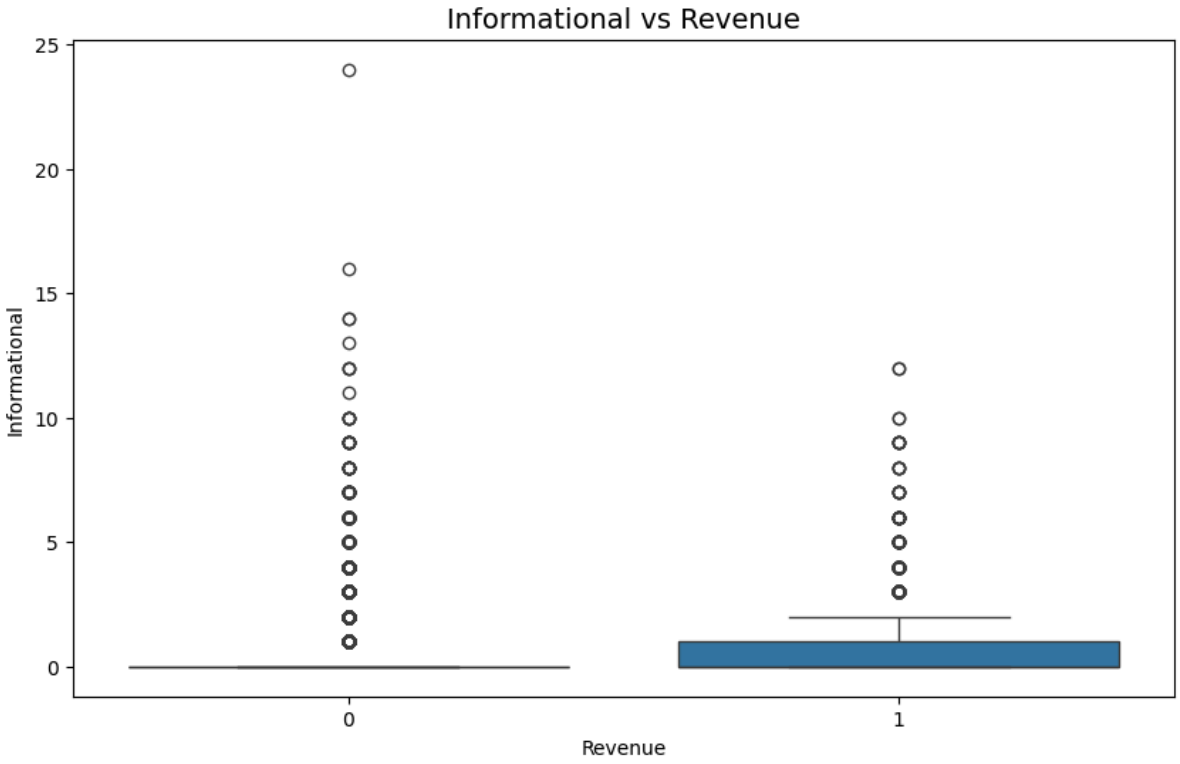
In [351…

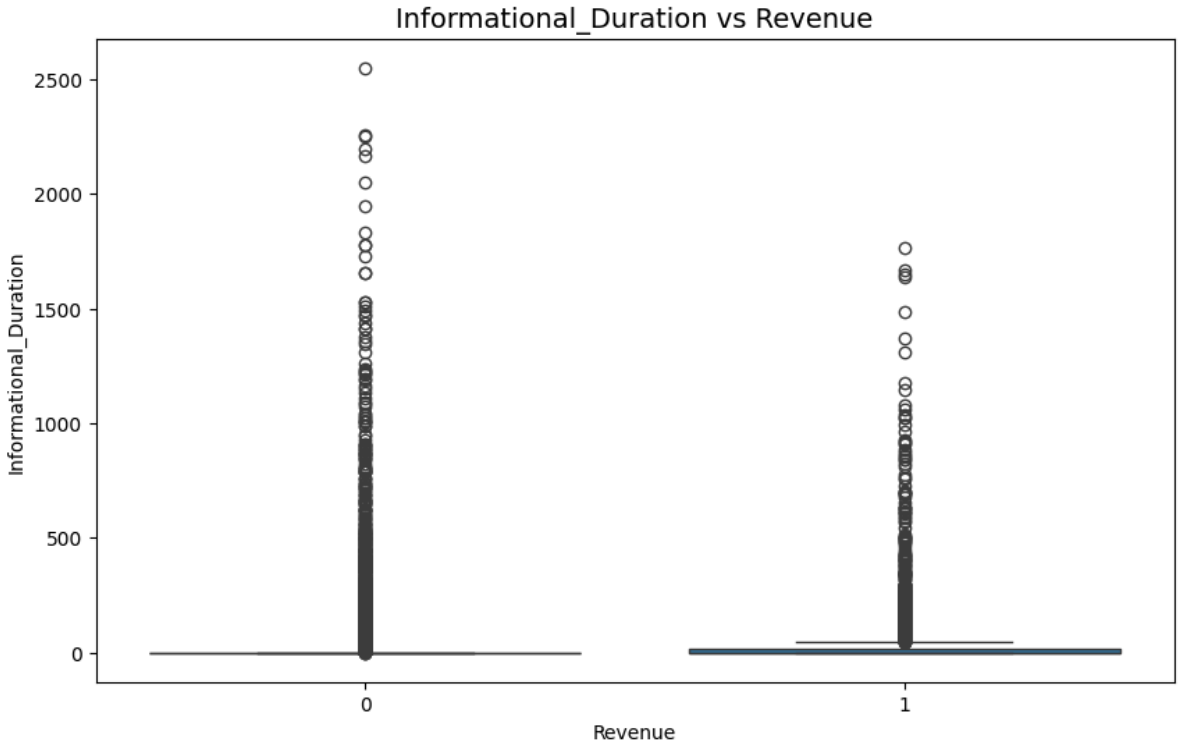
```
# Bi-variate analysis between numerical columns and the target variable (Revenue)
for col in num_cols:
    plt.figure(figsize=(10, 6))

    # Boxplot of numerical features vs target
    sns.boxplot(x='Revenue', y=df[col], data=df)
```

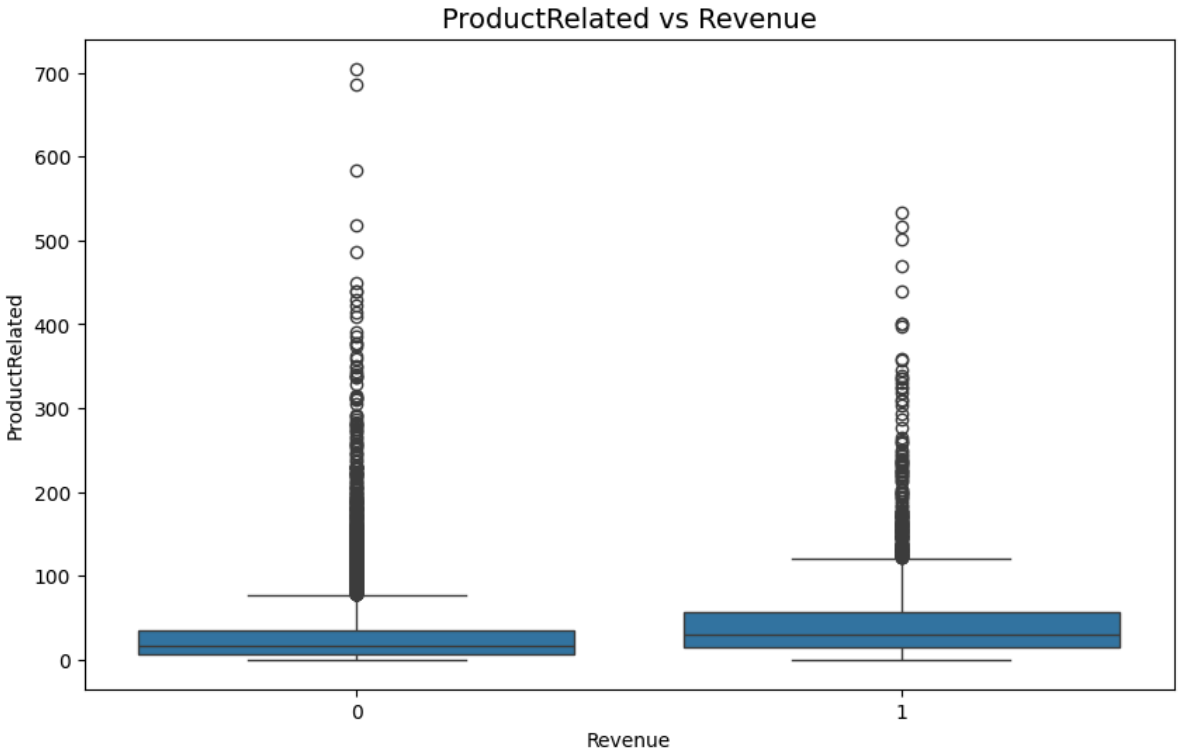
```
    plt.title(f'{col} vs Revenue', fontsize=14)
    plt.show()
```

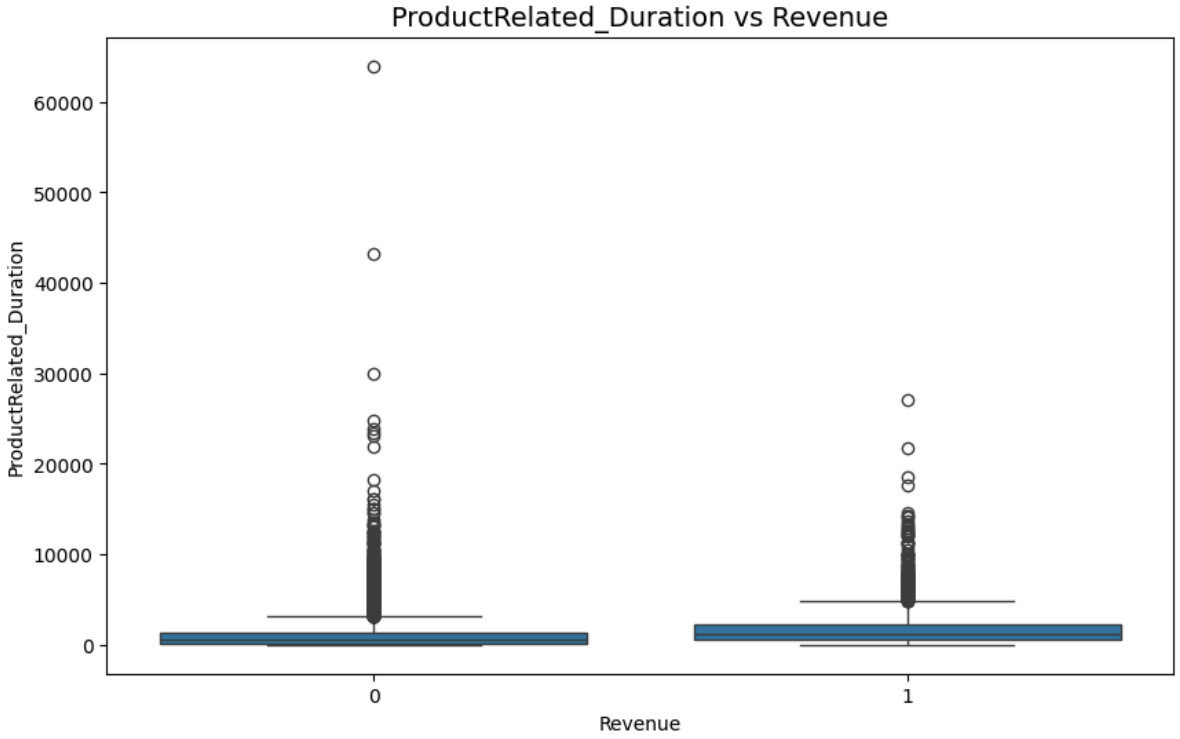


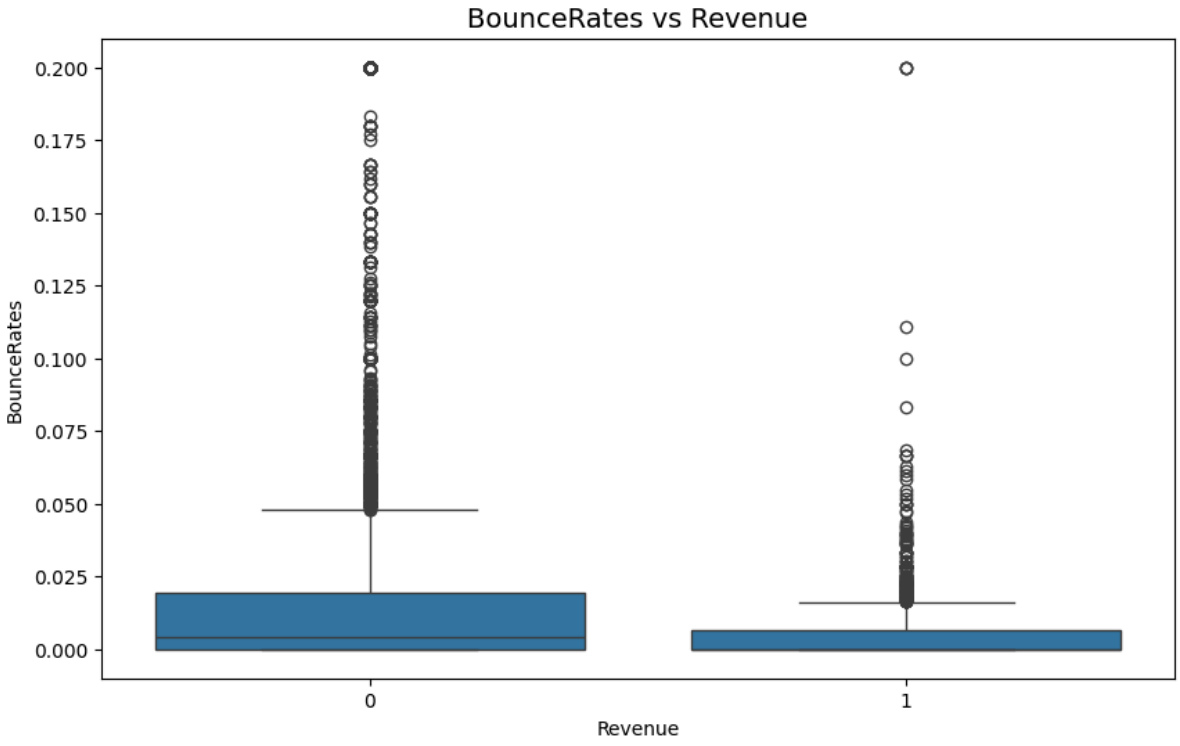


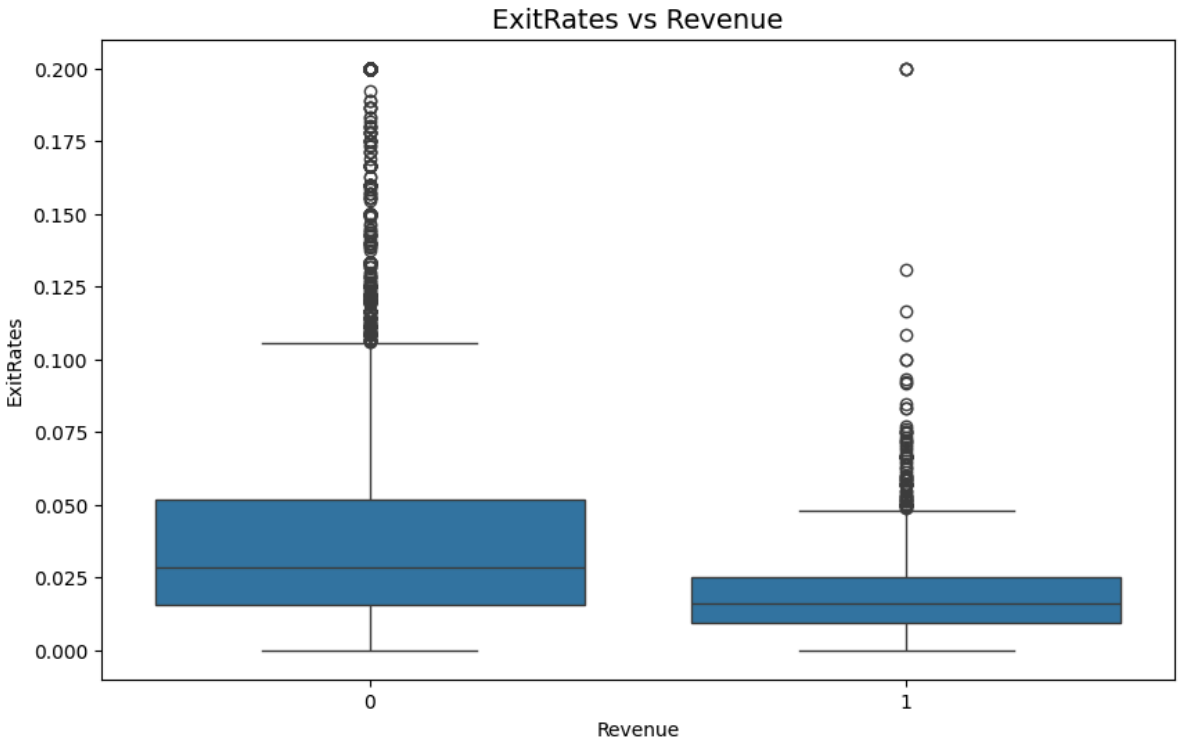

Informational vs Revenue
Informational
25
20
15
10
5
0
0
1
Revenue

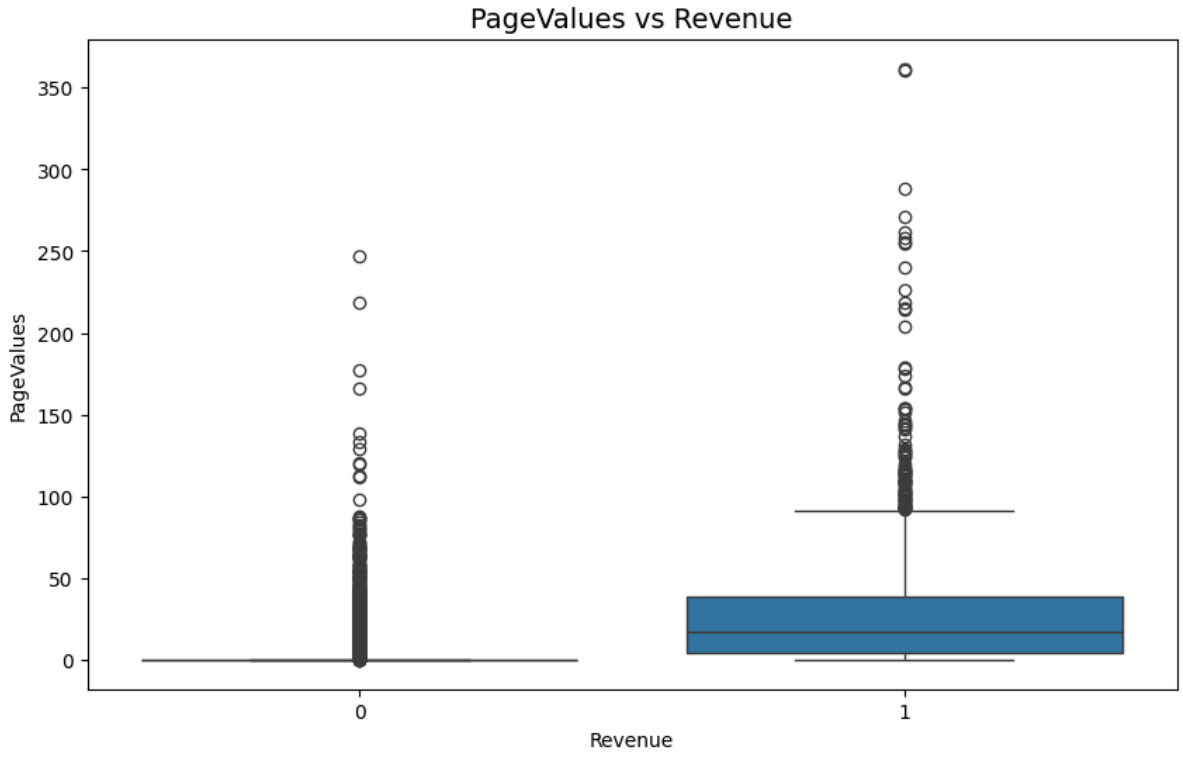

Informational_Duration vs Revenue
Informational_Duration
2500
2000
1500
1000
500
0
0
1
Revenue


ProductRelated vs Revenue
ProductRelated
700
600
500
400
300
200
100
0
0
1
Revenue


ProductRelated_Duration vs Revenue
ProductRelated_Duration
60000
50000
40000
30000
20000
10000
0
0
1
Revenue

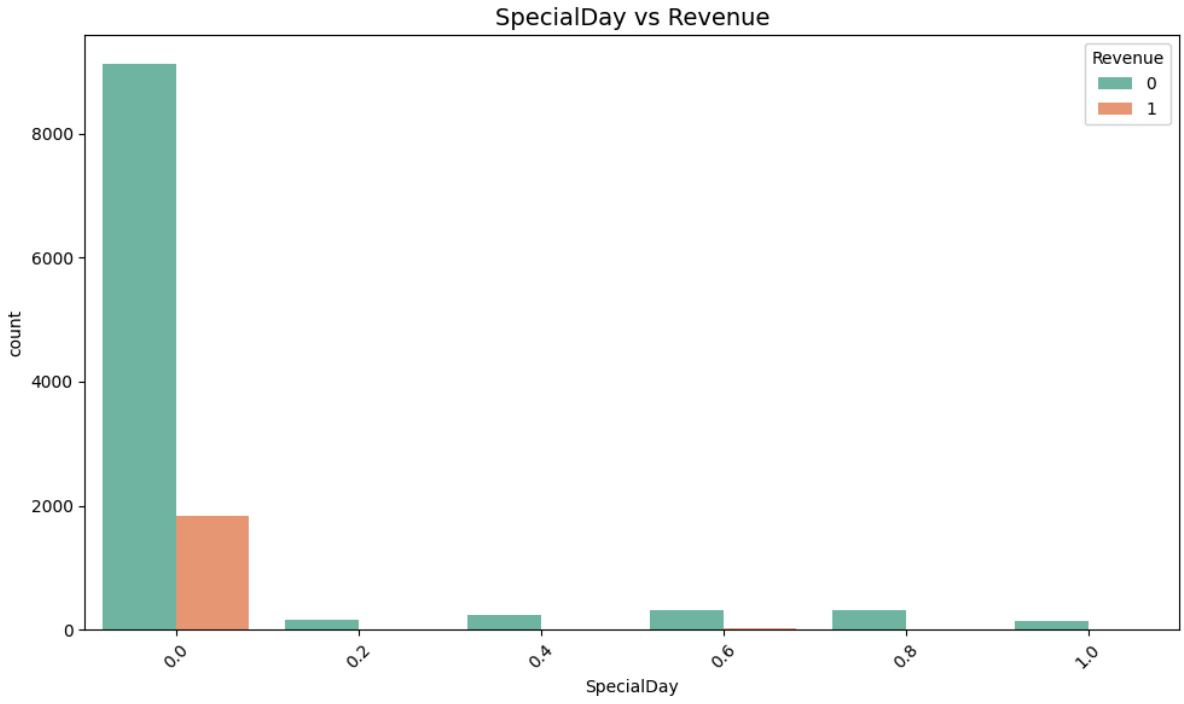

BounceRates vs Revenue
BounceRates
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
1
Revenue

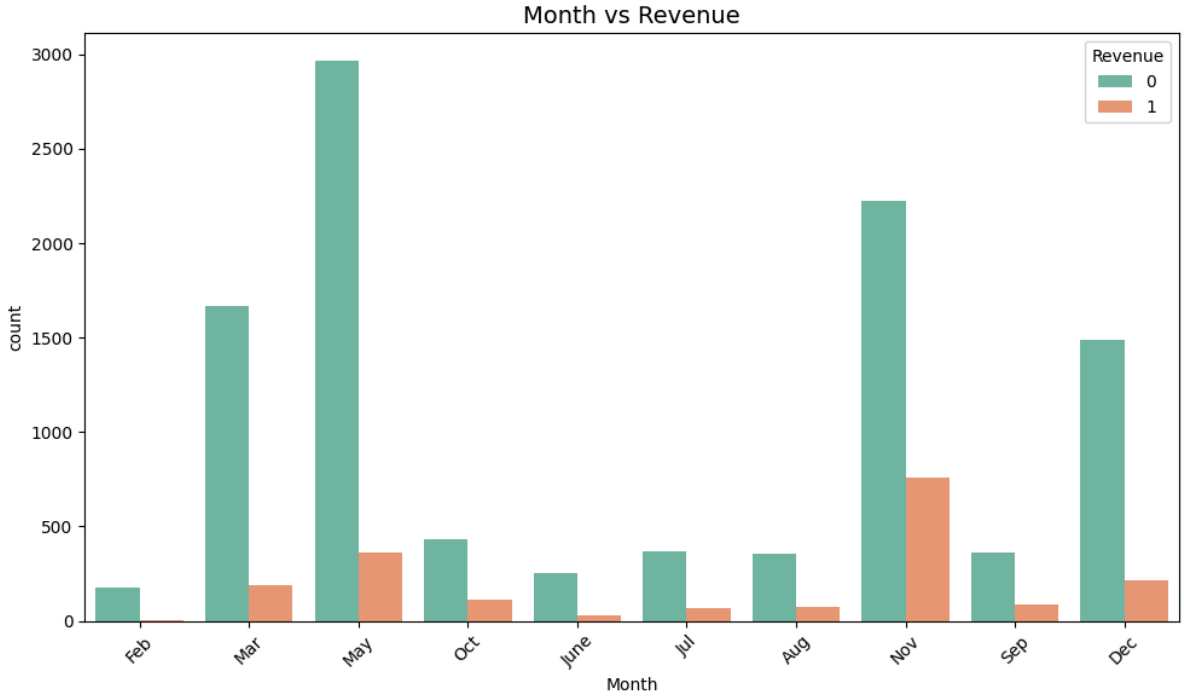

ExitRates vs Revenue
ExitRates
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
1
Revenue

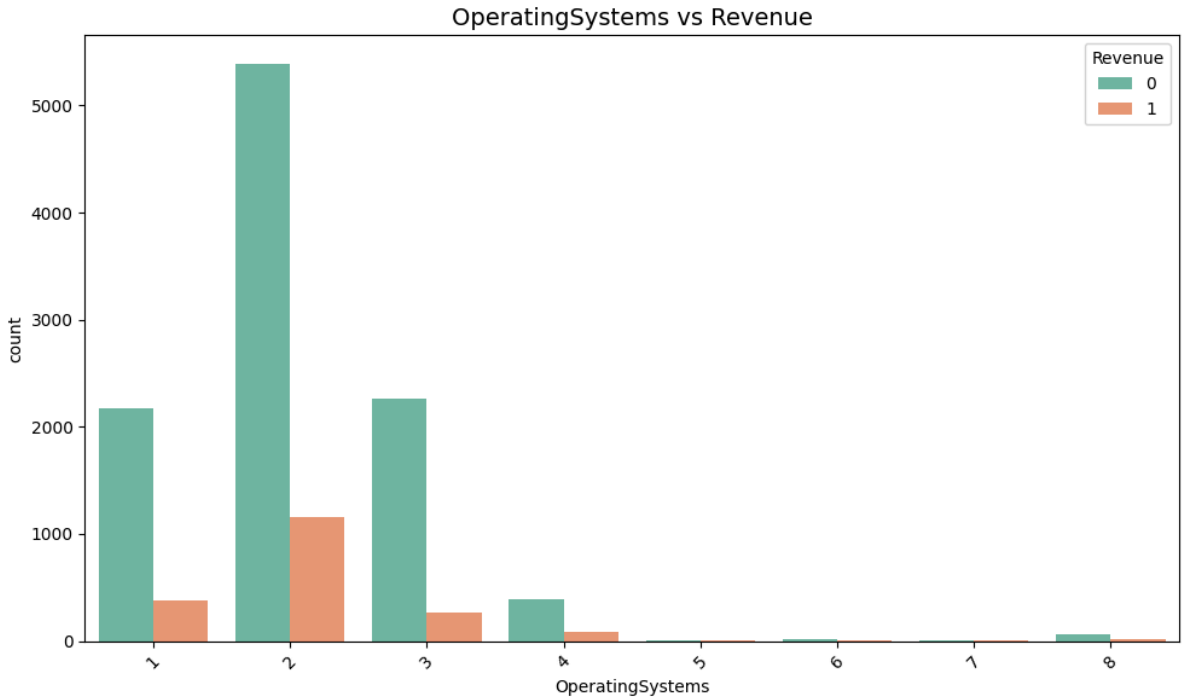


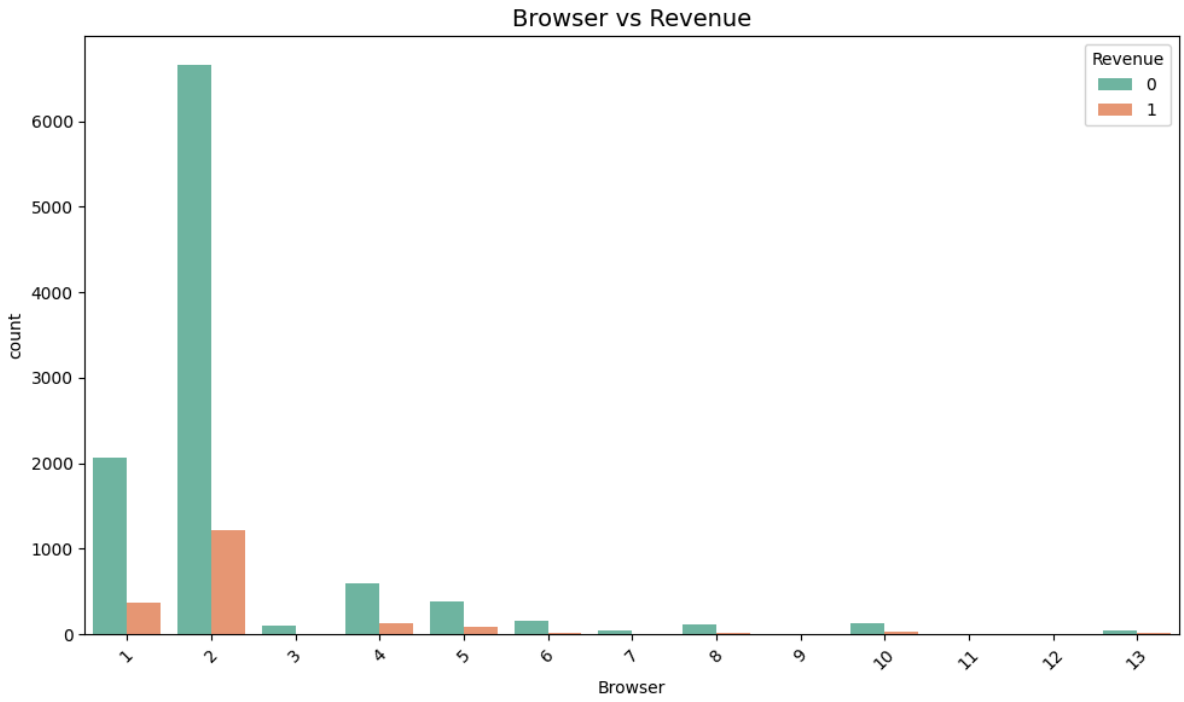
In [352…

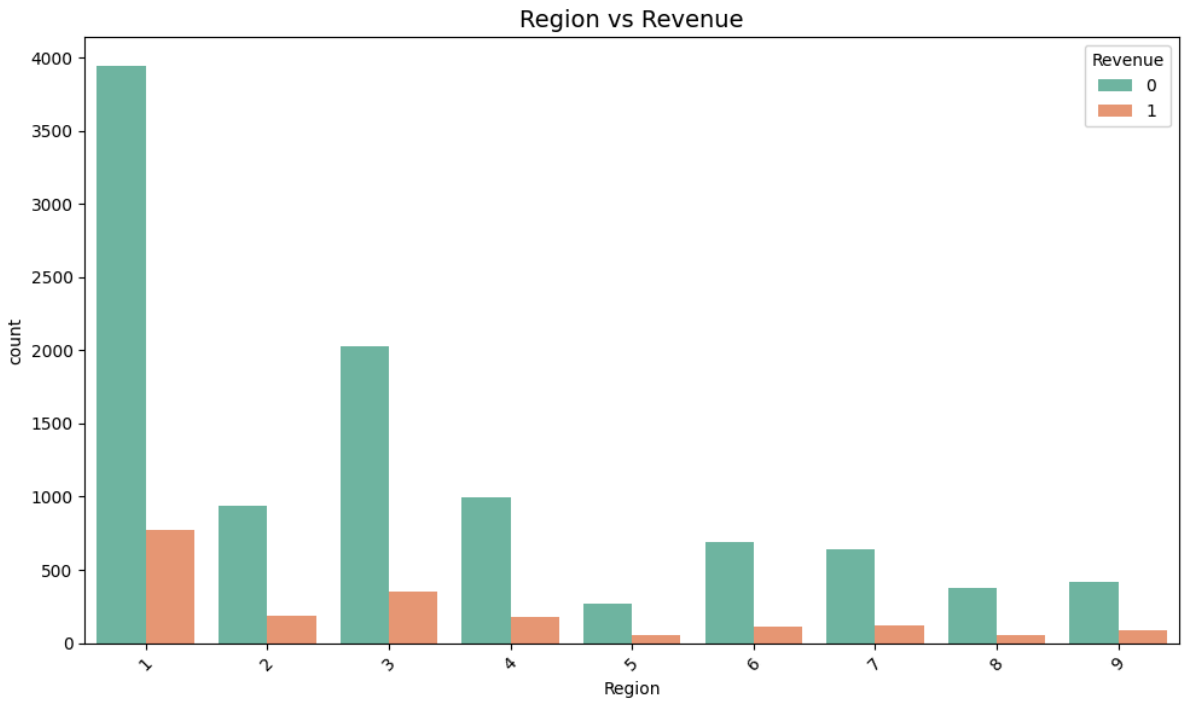
```
# Count plot for categorical variables vs target (Revenue)
for col in cat_cols:
    plt.figure(figsize=(10, 6))

    # Count plot of categorical features vs target
    sns.countplot(x=col, hue='Revenue', data=df, palette='Set2')
    plt.title(f'{col} vs Revenue', fontsize=14)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

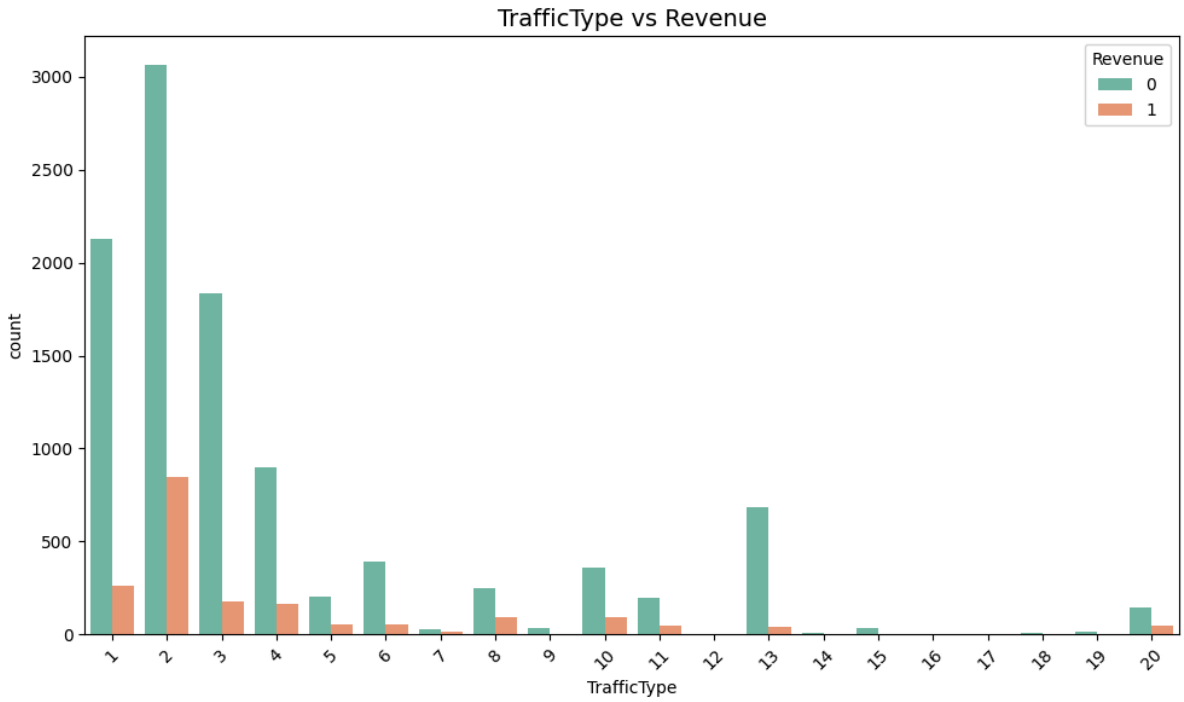


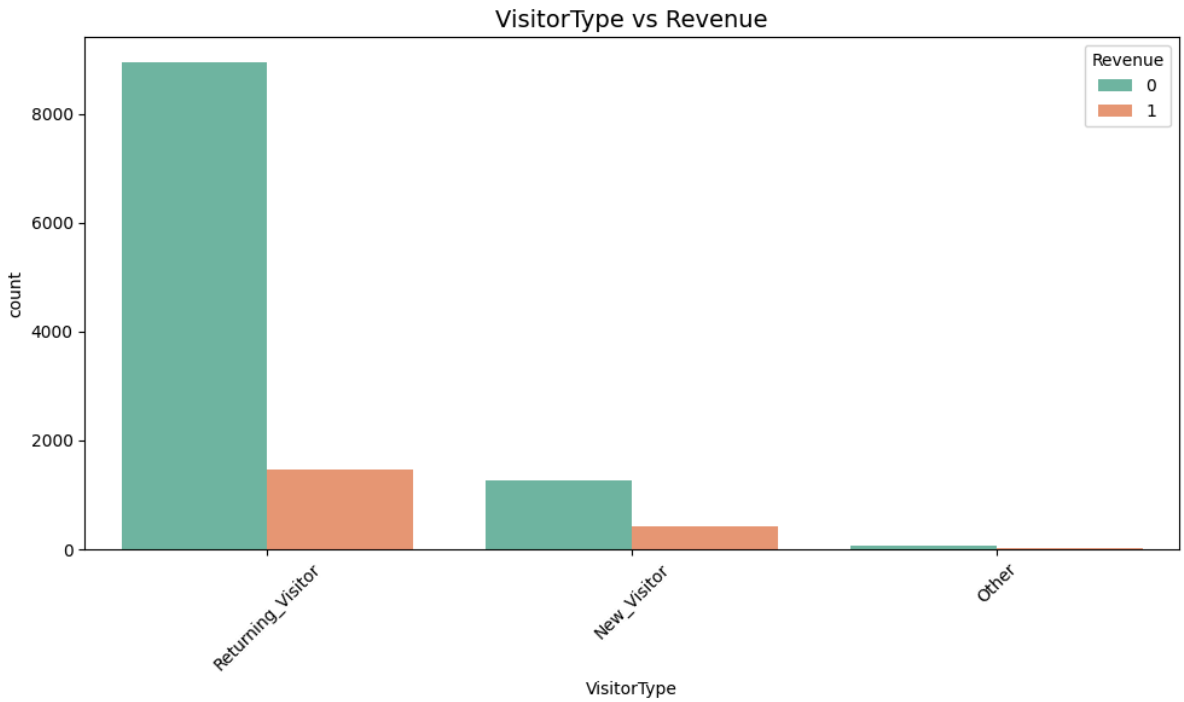

Month vs Revenue
Revenue
0
1
count
3000
2500
2000
1500
1000
500
0
Feb
Mar
May
Oct
June
Jul
Aug
Nov
Sep
Dec
Month

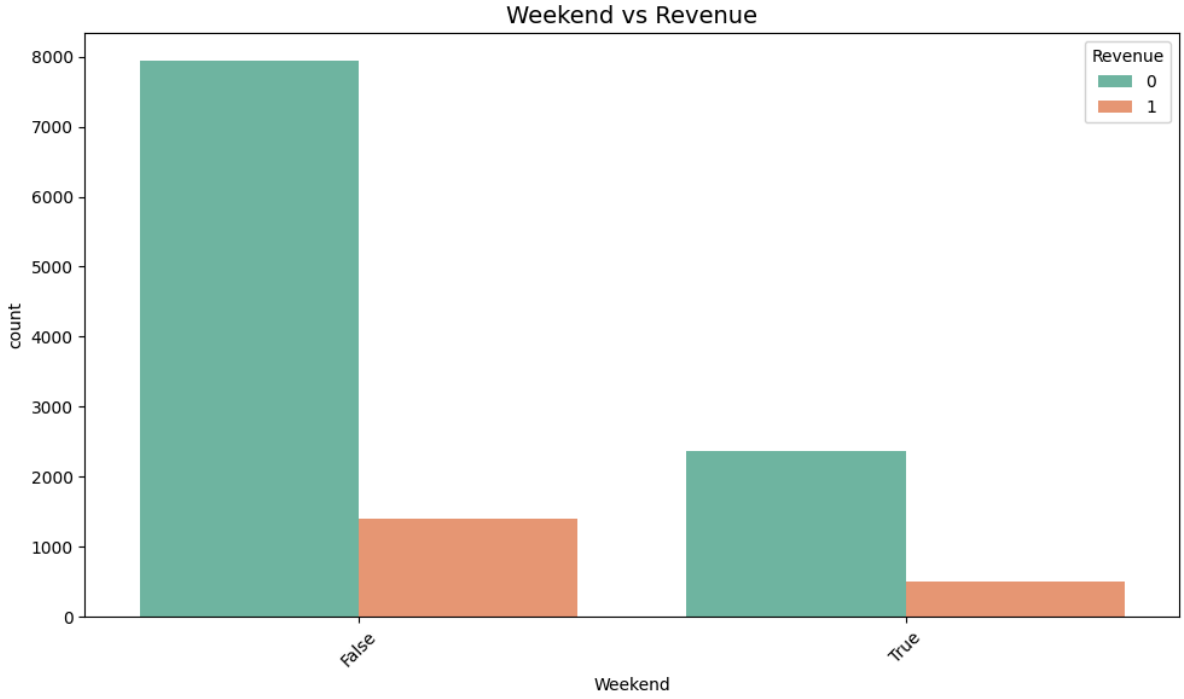

OperatingSystems vs Revenue
Revenue
0
1
count
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
OperatingSystems


Browser vs Revenue
Revenue
0
1
count
Browser
0
1000
2000
3000
4000
5000
6000
1
2
3
4
5
6
7
8
9
10
11
12
13


Region vs Revenue
Revenue
0
1
count
Region
0
500
1000
1500
2000
2500
3000
3500
4000
1
2
3
4
5
6
7
8
9

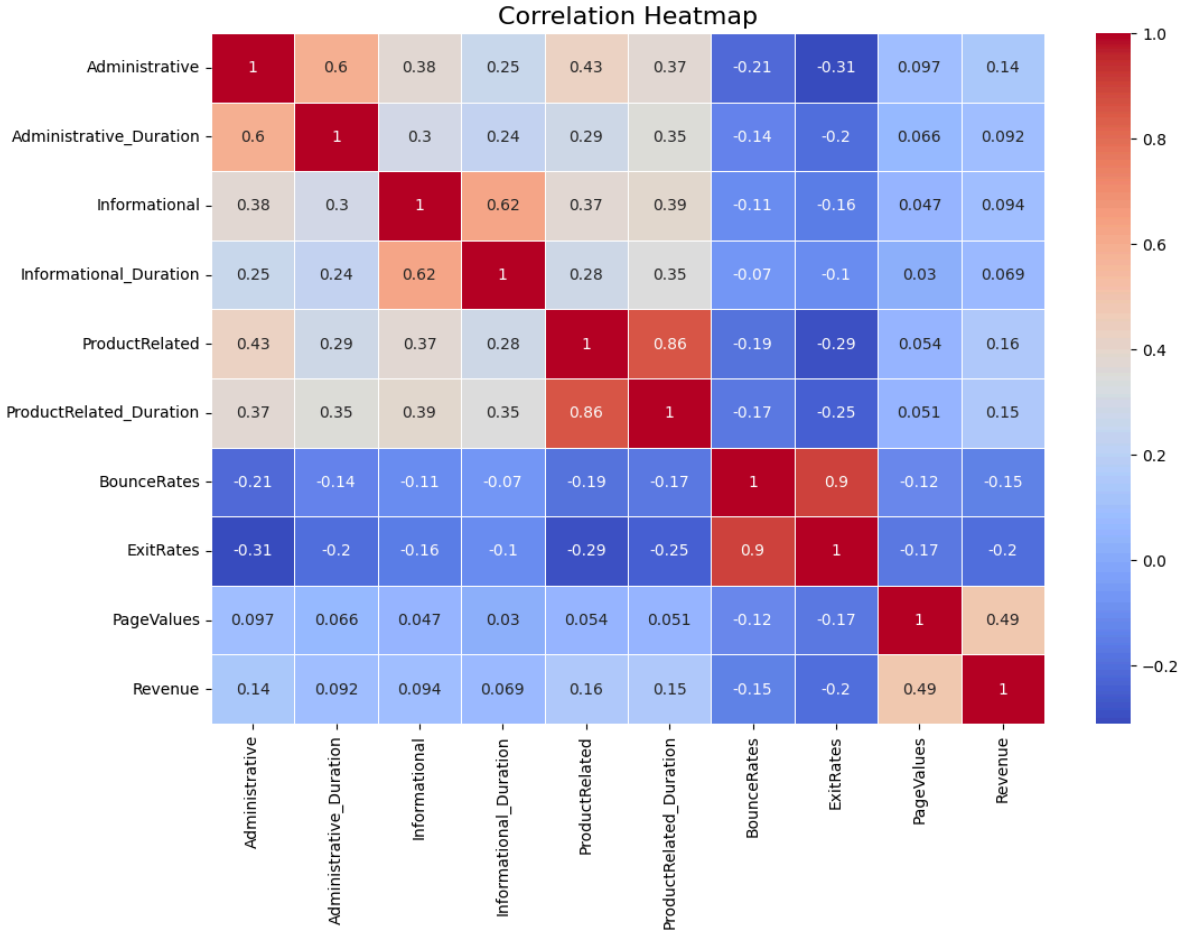

TrafficType vs Revenue
Revenue
0
1
count
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
TrafficType


VisitorType vs Revenue
Revenue
0
1
count
8000
6000
4000
2000
0
Returning_Visitor
New_Visitor
Other
VisitorType



In [353…

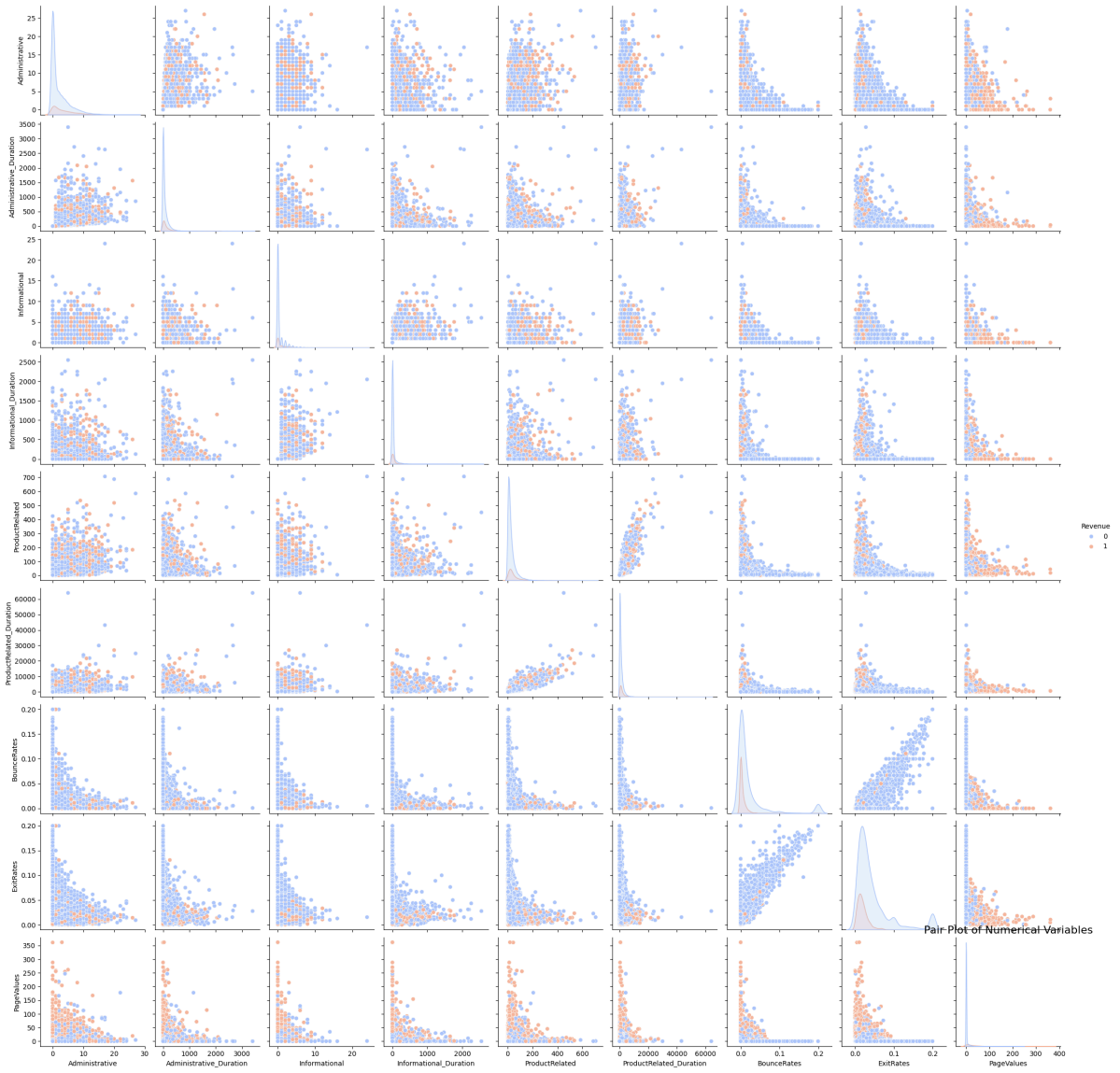
```
# Correlation heatmap between numerical variables and the target (Revenue)
plt.figure(figsize=(12, 8))
corr_matrix = df[num_cols + ['Revenue']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap', fontsize=16)
plt.show()
```



In [354…

```
# Pair plot for visualizing pairwise relationships between numerical columns
sns.pairplot(df[num_cols + ['Revenue']], hue='Revenue', palette='coolwarm')
```

```
plt.title('Pair Plot of Numerical Variables', fontsize=16)
plt.show()
```



In [355…

```
# Count plot for the target variable (Revenue)
plt.figure(figsize=(6, 4))
sns.countplot(x='Revenue', data=df, palette='Set2')
plt.title('Class Distribution of Revenue')
plt.show()

# Check the proportion of each class
print(df['Revenue'].value_counts(normalize=True))
```

```
<ipython-input-355-ecf7263dbfd3>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Revenue', data=df, palette='Set2')
```


```
Revenue
0    0.843671
1    0.156329
Name: proportion, dtype: float64
```

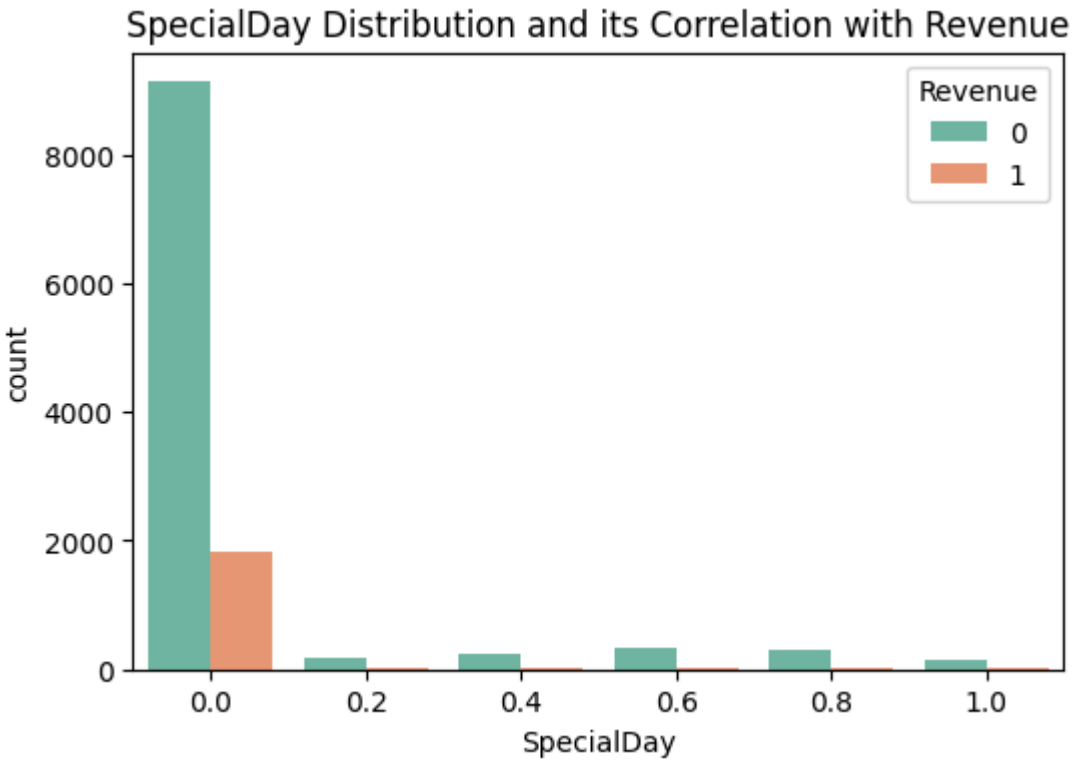
In [356…

```
# Summary of page views and durations by page category
page_cols = ['Administrative', 'Administrative_Duration', 'Informational', 'Informa
print(df[page_cols].describe())

# Summary of bounce and exit rates
bounce_exit_cols = ['BounceRates', 'ExitRates']
print(df[bounce_exit_cols].describe())
```

```
       Administrative  Administrative_Duration  Informational  \
count    12205.000000             12205.000000   12205.000000
mean         2.338878                81.646331       0.508726
std          3.330436               177.491845       1.275617
min          0.000000                 0.000000       0.000000
25%          0.000000                 0.000000       0.000000
50%          1.000000                 9.000000       0.000000
75%          4.000000                94.700000       0.000000
max         27.000000              3398.750000      24.000000

       Informational_Duration  ProductRelated  ProductRelated_Duration
count            12205.000000    12205.000000             12205.000000
mean                34.825454       32.045637              1206.982457
std                141.424807       44.593649              1919.601400
min                  0.000000        0.000000                 0.000000
25%                  0.000000        8.000000               193.000000
50%                  0.000000       18.000000               608.942857
75%                  0.000000       38.000000              1477.154762
max               2549.375000      705.000000             63973.522230
        BounceRates     ExitRates
count  12205.000000  12205.000000
mean       0.020370      0.041466
std        0.045255      0.046163
min        0.000000      0.000000
25%        0.000000      0.014231
50%        0.002899      0.025000
75%        0.016667      0.048529
max        0.200000      0.200000
```

In [357…

```
# Count plot for SpecialDay distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='SpecialDay', hue='Revenue', data=df, palette='Set2')
plt.title('SpecialDay Distribution and its Correlation with Revenue')
plt.show()

# Correlation of SpecialDay with Revenue
special_day_corr = df[['SpecialDay', 'Revenue']].corr()
print(special_day_corr)
```

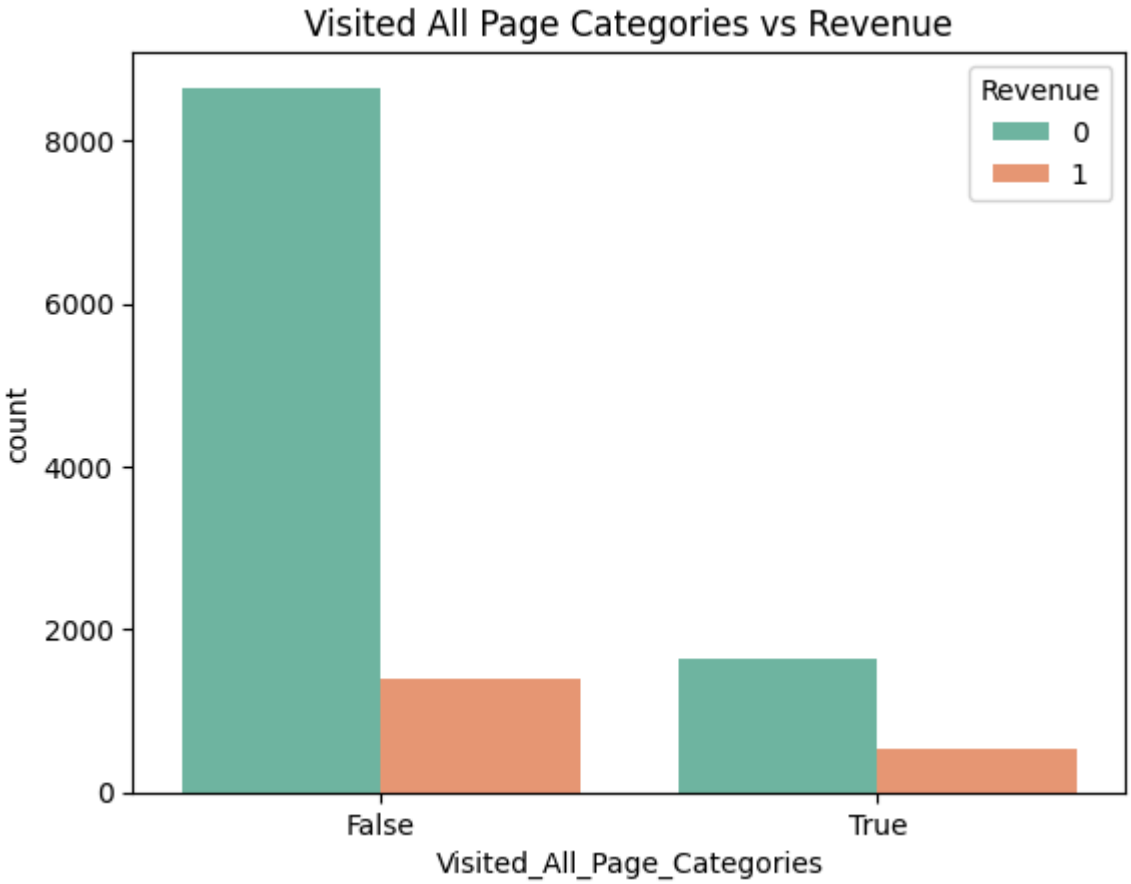


```
             SpecialDay   Revenue
SpecialDay     1.000000 -0.083601
Revenue       -0.083601  1.000000
```

In [358…

```
# Create a binary feature indicating if the user visited all three page categories
df['Visited_All_Page_Categories'] = (df['Administrative'] > 0) & (df['Informational
print(df[['Visited_All_Page_Categories', 'Revenue']].head())

# Count plot to analyze the new feature
sns.countplot(x='Visited_All_Page_Categories', hue='Revenue', data=df, palette='Set
plt.title('Visited All Page Categories vs Revenue')
plt.show()
```

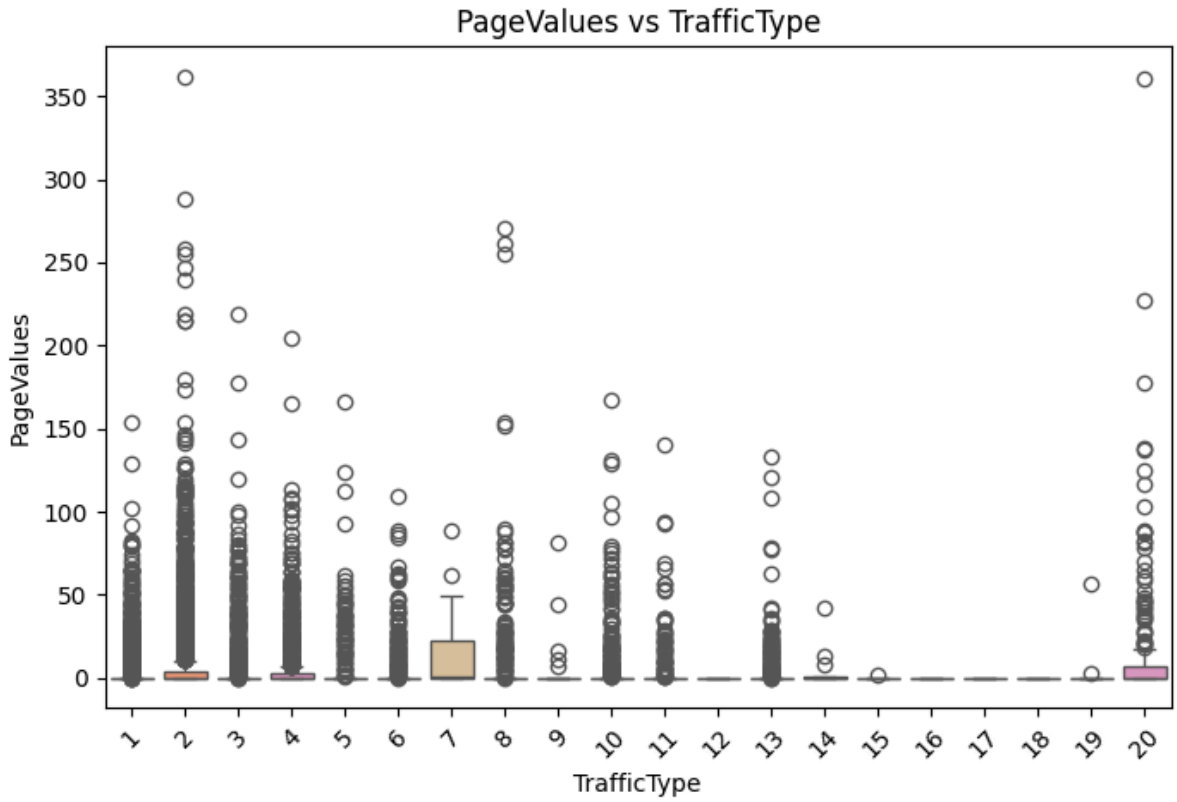
```
   Visited_All_Page_Categories  Revenue
0                        False        0
1                        False        0
2                        False        0
3                        False        0
4                        False        0
```



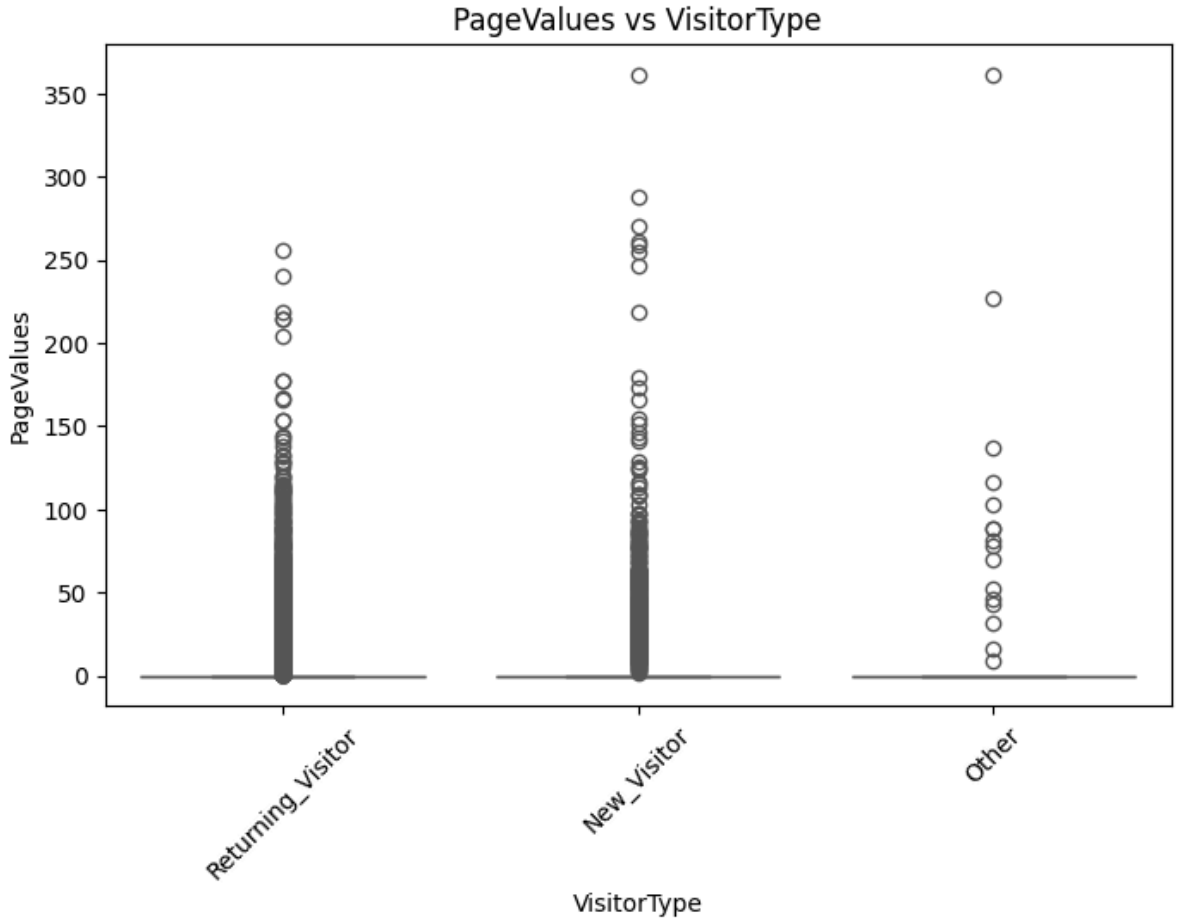
In [359…

```
# Boxplot of PageValues with different categories
for col in ['TrafficType', 'VisitorType', 'Region']:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x=col, y='PageValues', data=df, palette='Set2')
    plt.title(f'PageValues vs {col}')
    plt.xticks(rotation=45)
    plt.show()
```

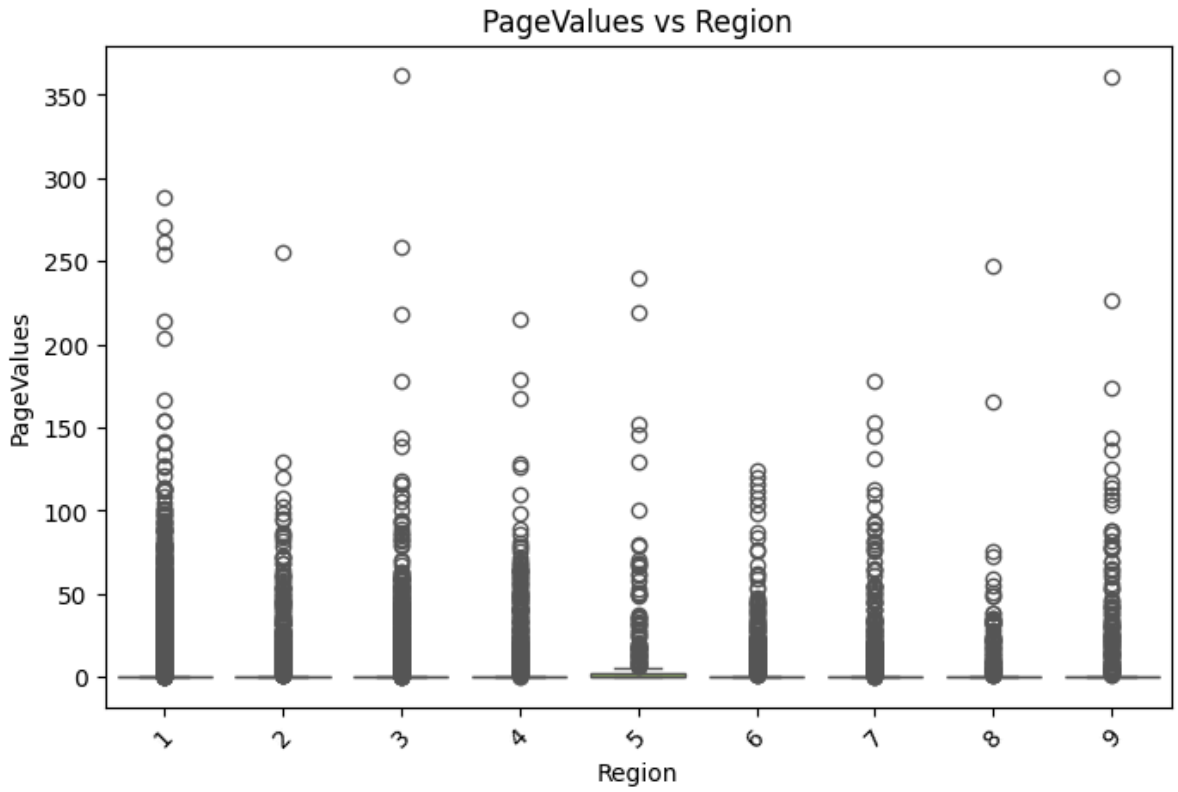
```
<ipython-input-359-9492d60f3739>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=col, y='PageValues', data=df, palette='Set2')
```




```
<ipython-input-359-9492d60f3739>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=col, y='PageValues', data=df, palette='Set2')
```



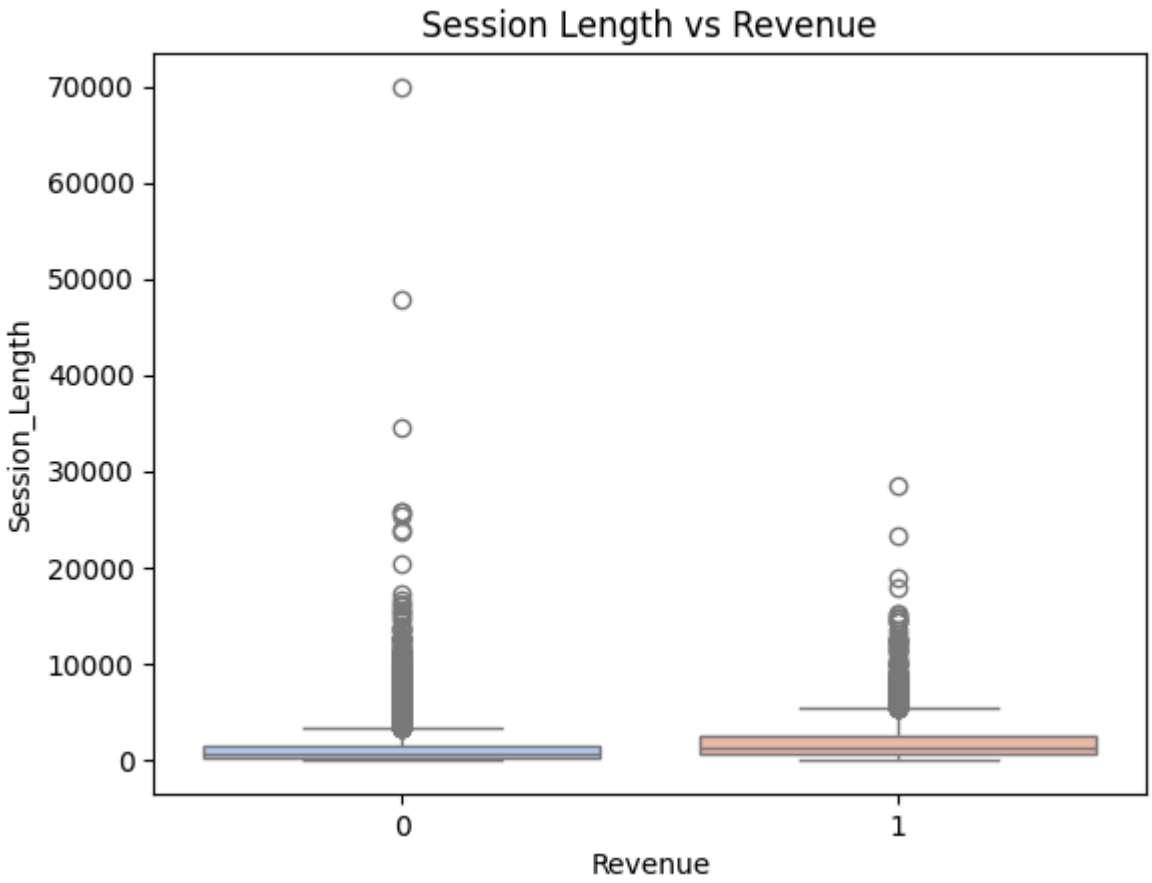
```
<ipython-input-359-9492d60f3739>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=col, y='PageValues', data=df, palette='Set2')
```



In [360…

```
# Calculate session length as the sum of all durations
df['Session_Length'] = df['Administrative_Duration'] + df['Informational_Duration']

# Boxplot for session lengths
sns.boxplot(x='Revenue', y='Session_Length', data=df, palette='coolwarm')
plt.title('Session Length vs Revenue')
plt.show()
```

```
<ipython-input-360-059e869dfa70>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='Revenue', y='Session_Length', data=df, palette='coolwarm')
```



In [361…

```
### 1. Group by VisitorType, OperatingSystems, and Region using agg() ###
grouped_data = df.groupby(['VisitorType', 'OperatingSystems', 'Region']).agg({
    'PageValues': ['mean', 'median', 'count'],
    'BounceRates': ['mean', 'median'],
    'ExitRates': ['mean', 'median'],
    'Revenue': ['mean', 'sum', 'count']  # mean -> purchase probability, sum -> tot
})

# Flatten the MultiIndex columns after agg()
grouped_data.columns = ['_'.join(col).strip() for col in grouped_data.columns.value
print("Grouped by VisitorType, OperatingSystems, and Region:")
print(grouped_data)

# Visualization: VisitorType vs Revenue by Operating Systems
plt.figure(figsize=(12, 6))
sns.boxplot(x='VisitorType', y='Revenue', data=df, hue='OperatingSystems', palette=
plt.title('VisitorType vs Revenue by Operating Systems')
plt.show()

# Visualization: Region vs Revenue by VisitorType
plt.figure(figsize=(12, 6))
sns.boxplot(x='Region', y='Revenue', data=df, hue='VisitorType', palette='coolwarm'
plt.title('Region vs Revenue by VisitorType')
plt.show()
```

Grouped by VisitorType, OperatingSystems, and Region:

VisitorType	OperatingSystems	Region	PageValues_mean	PageValues_median
New_Visitor	1	1	10.803223	0.0
		2	8.827076	0.0
		3	4.819538	0.0
		4	15.363550	0.0
		5	47.873485	0.0
...			...	...
Returning_Visitor	8	4	0.000000	0.0
		5	0.000000	0.0
		6	0.000000	0.0
		7	0.000000	0.0
		9	0.000000	0.0

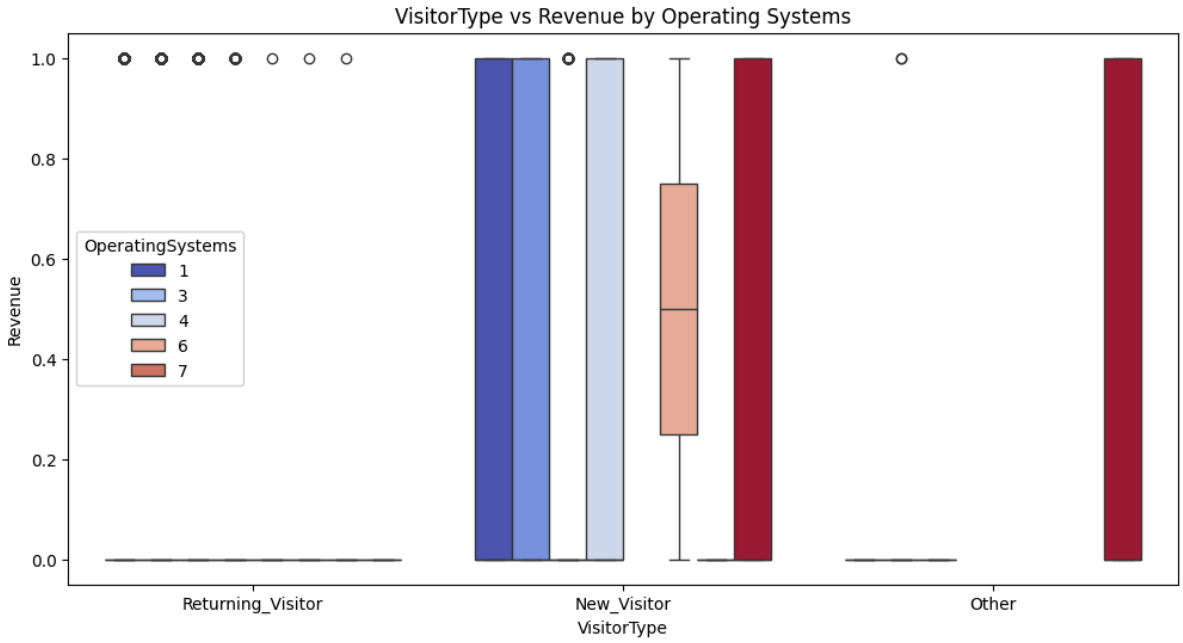
VisitorType	OperatingSystems	Region	PageValues_count	BounceRates_mean
New_Visitor	1	1	172	0.008729
		2	41	0.007214
		3	87	0.007632
		4	37	0.007870
		5	6	0.034848
...			...	...
Returning_Visitor	8	4	2	0.000000
		5	1	0.000000
		6	1	0.200000
		7	1	0.000000
		9	1	0.200000

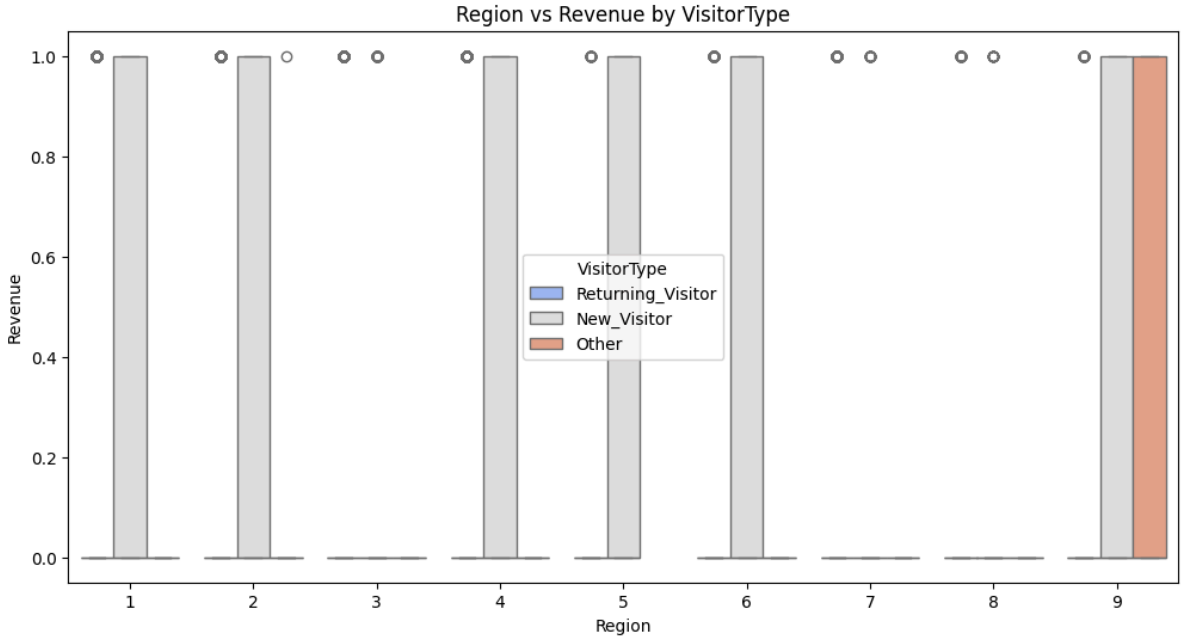
VisitorType	OperatingSystems	Region	BounceRates_median	ExitRates_mean
New_Visitor	1	1	0.0	0.025466
		2	0.0	0.030134
		3	0.0	0.022825
		4	0.0	0.025108
		5	0.0	0.045864
...			...	...
Returning_Visitor	8	4	0.0	0.075000
		5	0.0	0.003175
		6	0.2	0.200000
		7	0.0	0.100000
		9	0.2	0.200000

VisitorType	OperatingSystems	Region	ExitRates_median	Revenue_mean
New_Visitor	1	1	0.015385	0.267442
		2	0.022222	0.341463
		3	0.014706	0.149425
		4	0.016667	0.297297
		5	0.016234	0.333333
...			...	...
Returning_Visitor	8	4	0.075000	0.000000
		5	0.003175	0.000000
		6	0.200000	0.000000
		7	0.100000	0.000000
		9	0.200000	0.000000

VisitorType	OperatingSystems	Region	Revenue_sum	Revenue_count
New_Visitor	1	1	46	172
		2	14	41
		3	13	87
		4	11	37
		5	2	6

```
...                                    ...          ...
Returning_Visitor 8                    4              0            2
                                       5              0            1
                                       6              0            1
                                       7              0            1
                                       9              0            1

[115 rows x 10 columns]
```





In [362…

```python
### 2. Segment by TrafficType using agg() ###
grouped_by_traffic = df.groupby('TrafficType').agg({
    'PageValues': ['mean', 'median', 'count'],
    'BounceRates': ['mean', 'median'],
    'ExitRates': ['mean', 'median'],
    'Revenue': ['mean', 'sum', 'count']  # mean -> purchase probability, sum -> tot
})

# Flatten the MultiIndex columns after agg()
grouped_by_traffic.columns = ['_'.join(col).strip() for col in grouped_by_traffic.c
print("Grouped by TrafficType:")
print(grouped_by_traffic)

# Visualization: TrafficType vs PageValues
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='TrafficType', y='PageValues', data=df, palette='coolwarm')
plt.title('TrafficType vs PageValues')
plt.show()

# Visualization: TrafficType vs Revenue (purchase probability)
plt.figure(figsize=(10, 6))
sns.barplot(x='TrafficType', y='Revenue', data=df, palette='coolwarm')
plt.title('TrafficType vs Purchase Probability')
plt.show()
```

Grouped by TrafficType:

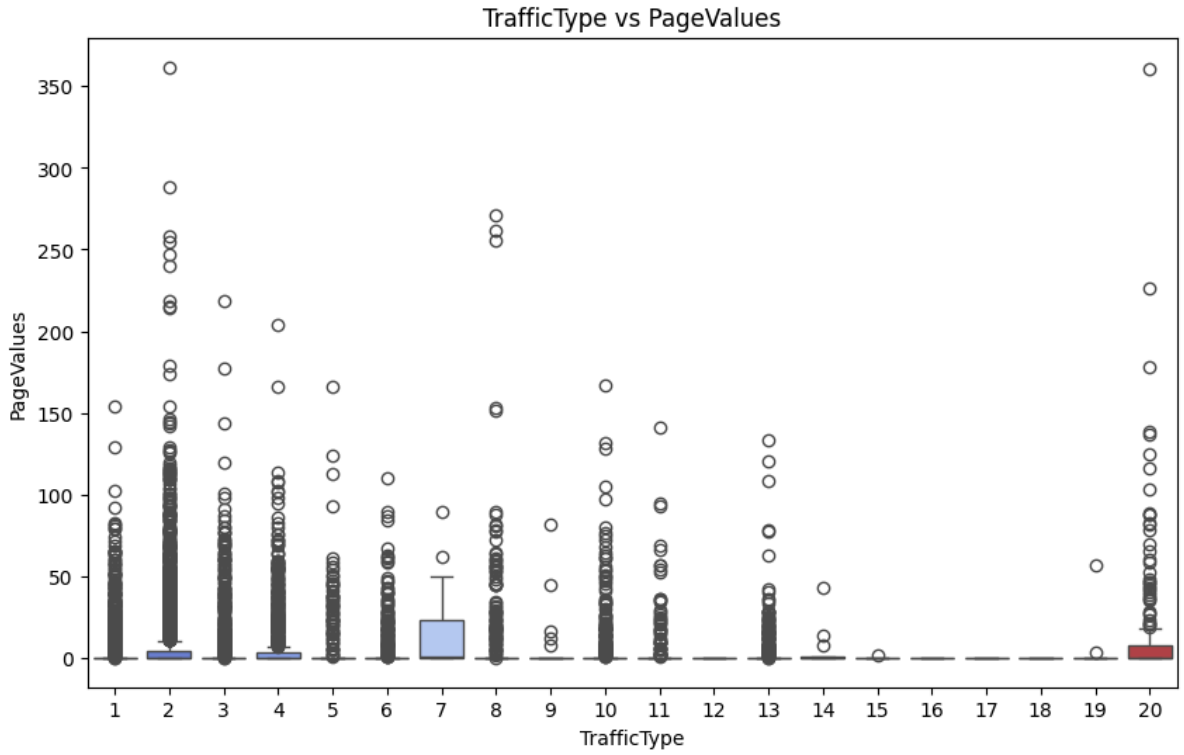
TrafficType	PageValues_mean	PageValues_median	PageValues_count
1	3.546226	0.000000	2388
2	8.308613	0.000000	3911
3	3.339503	0.000000	2013
4	7.062934	0.000000	1066
5	7.712489	0.000000	260
6	5.087703	0.000000	443
7	13.567345	0.423669	40
8	10.302436	0.000000	343
9	3.911694	0.000000	41
10	6.208230	0.000000	450
11	5.068642	0.000000	247
12	0.000000	0.000000	1
13	2.386929	0.000000	728
14	4.936097	0.000000	13
15	0.037454	0.000000	37
16	0.000000	0.000000	3
17	0.000000	0.000000	1
18	0.000000	0.000000	10
19	3.497520	0.000000	17
20	15.520252	0.000000	193

TrafficType	BounceRates_mean	BounceRates_median	ExitRates_mean
1	0.027923	0.007692	0.051901
2	0.008357	0.000000	0.026303
3	0.030085	0.006667	0.054349
4	0.015744	0.001388	0.035694
5	0.009451	0.000000	0.029679
6	0.021702	0.003636	0.045063
7	0.007822	0.001490	0.024818
8	0.011499	0.000000	0.029639
9	0.022106	0.000000	0.041595
10	0.016633	0.001956	0.037910
11	0.022210	0.002020	0.043753
12	0.000000	0.000000	0.066667
13	0.046630	0.021132	0.068552
14	0.002434	0.000000	0.024006
15	0.066406	0.028571	0.086776
16	0.000000	0.000000	0.017998
17	0.050000	0.050000	0.075000
18	0.033128	0.008824	0.053055
19	0.025751	0.004310	0.049762
20	0.023066	0.000000	0.043599

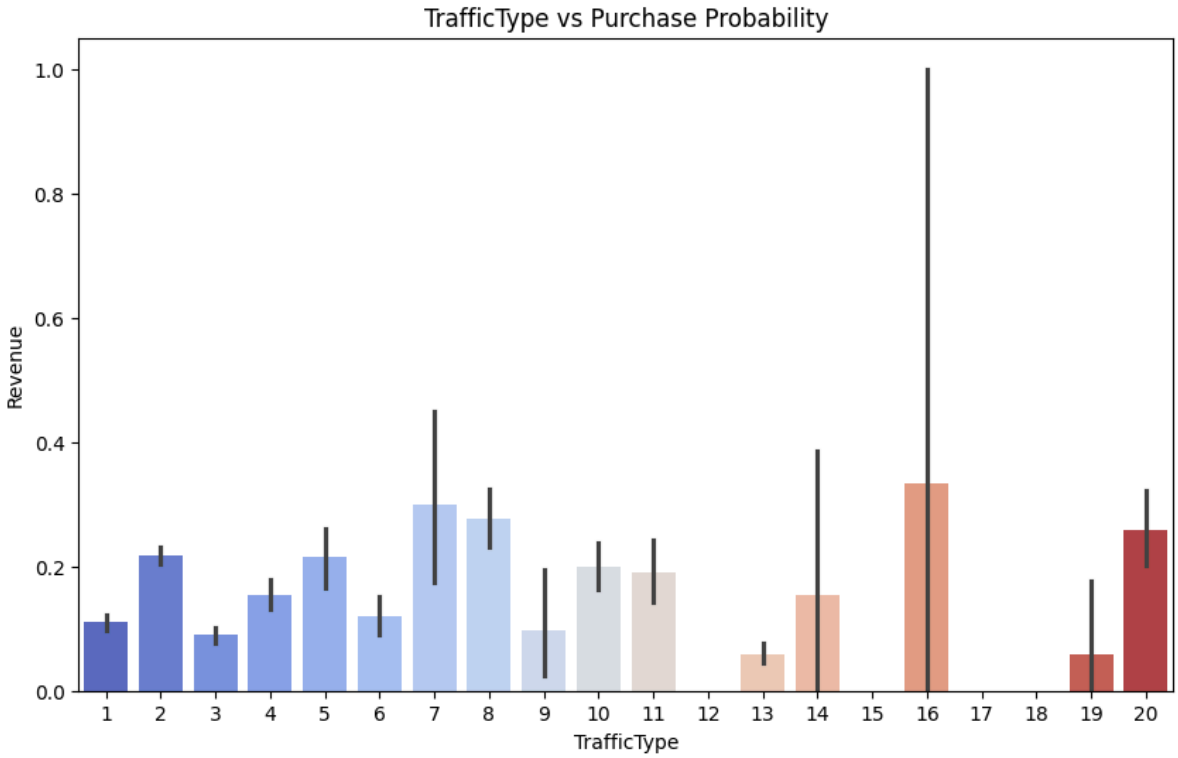
TrafficType	ExitRates_median	Revenue_mean	Revenue_sum	Revenue_count
1	0.033333	0.109715	262	2388
2	0.018750	0.216569	847	3911
3	0.033333	0.089419	180	2013
4	0.022222	0.154784	165	1066
5	0.018182	0.215385	56	260
6	0.030172	0.119639	53	443
7	0.023443	0.300000	12	40
8	0.019048	0.276968	95	343
9	0.020667	0.097561	4	41
10	0.023171	0.200000	90	450
11	0.026667	0.190283	47	247
12	0.066667	0.000000	0	1
13	0.046652	0.059066	43	728
14	0.016087	0.153846	2	13
15	0.066667	0.000000	0	37

16	0.010526	0.333333	1	3
17	0.075000	0.000000	0	1
18	0.032323	0.000000	0	10
19	0.044444	0.058824	1	17
20	0.026667	0.259067	50	193

```
<ipython-input-362-7fa74df8ec4f>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='TrafficType', y='PageValues', data=df, palette='coolwarm')
```




```
<ipython-input-362-7fa74df8ec4f>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='TrafficType', y='Revenue', data=df, palette='coolwarm')
```



Insights

1. **Visitor Engagement**: Visitors spend the most time on Product-related pages (average: 1195 seconds), followed by Administrative (81 seconds) and Informational pages (34 seconds). They visit Product pages more frequently (average: 32 visits) compared to Administrative (2) and Informational (0.5).

2. **Page Value**: The average Page Value is 18, with notable outliers. Regions 5 and 9 have higher Page Values (average: 9), while Traffic Type 20 has the highest (average: 16).

3. **Peak Months**: May (27%) and November (24%) have the highest visitor numbers, with most purchases occurring in November. Visitors spend more time and encounter fewer outliers when they buy.

4. **Visitor Demographics**: 54% of visitors use OS 2, 65% use Browser 2, and 39% come from Region 1, which also leads in purchases, followed by Region 3.

5. **Traffic Types**: Traffic Type 2 accounts for 32% of visits and leads in purchases.

6. **Returning Visitors**: 86% of visitors are returning, primarily making purchases on weekdays, and they exhibit high exit and bounce rates.

7. **Weekend Traffic**: 23% of visits occur on weekends, and 16% result in purchases.

8. **Special Days Impact**: Special Days have a weak correlation with revenue, but visits increase around these times.

9. **Page Interaction**: There's a strong connection between page types and the time spent on them, with higher bounce rates on fewer pages.

10. **Cross-Category Visits**: 18% of visitors explore all categories, with a link between Informational pages and overall engagement.

Recommendations

1. **Optimize Product Pages**: Focus on enhancing Product-related pages since visitors spend the most time here. Improve details, visuals, and overall user experience.
2. **Revamp Administrative and Informational Pages**: Simplify these pages for quicker navigation and easy access to key information like checkout and account details.
3. **Promote During Special Days**: Launch targeted campaigns around special days to take advantage of increased visitor traffic.
4. **Investigate High Page Values**: Look into why Regions 5 and 9, as well as Traffic Type 20, have higher Page Values and apply those strategies elsewhere.
5. **Engage Returning Visitors**: Develop loyalty programs and personalized recommendations to encourage repeat purchases from returning visitors.
6. **Plan Promotions for Weekdays**: Schedule major promotions and email campaigns on weekdays when purchases peak.
7. **Ensure Compatibility with Popular Platforms**: Since most visitors use OS 2 and Browser 2, make sure your website performs well on these platforms.
8. **Reduce Bounce Rates**: Work on improving landing pages to make better first impressions and reduce high bounce rates.
9. **Encourage Category Exploration**: Promote content that encourages visitors to check out multiple categories, as only 18% currently do.
10. **Leverage May and November**: Organize major sales or special offers during May and November to maximize revenue when traffic is high.