```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving shopping.csv to shopping (1).csv
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df=pd.read_csv('shopping.csv')
```

EDA

```
df
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 12330,\n \"fields\": [\n   {\n   \"column\": \"Administrative\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3,\n   \"min\": 0,\n   \"max\": 27,\n   \"num_unique_values\": 27,\n   \"samples\": [\n   5,\n   11,\n   9\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Administrative_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 176.77910747048634,\n   \"min\": 0.0,\n   \"max\": 3398.75,\n   \"num_unique_values\": 3335,\n   \"samples\": [\n   93.6,\n   63.08333333,\n   351.0833333\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Informational\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 24,\n   \"num_unique_values\": 17,\n   \"samples\": [\n   0,\n   1,\n   5\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Informational_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 140.74929442219798,\n   \"min\": 0.0,\n   \"max\": 2549.375,\n   \"num_unique_values\": 1258,\n   \"samples\": [\n   793.8,\n   50.0,\n   57.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ProductRelated\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 44,\n   \"min\": 0,\n   \"max\": 705,\n   \"num_unique_values\": 311,\n   \"samples\": [\n   330,\n   23,\n   54\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ProductRelated_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1913.6692878720035,\n

```
\"min\": 0.0,\n        \"max\": 63973.52223,\n
\"num_unique_values\": 9551,\n        \"samples\": [\n
225.68,\n          232.6666667,\n          2834.280117\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BounceRates\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.048488321806260656,\n        \"min\": 0.0,\n        \"max\": 0.2,\n
\"num_unique_values\": 1872,\n        \"samples\": [\n
0.004778942,\n          0.010793651,\n          0.005073996\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ExitRates\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.048596540551443565,\n        \"min\":
0.0,\n        \"max\": 0.2,\n        \"num_unique_values\": 4777,\n
\"samples\": [\n          0.120833333,\n          0.006857143,\n
0.008914729\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PageValues\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 18.568436607806525,\n        \"min\":
0.0,\n        \"max\": 361.7637419,\n        \"num_unique_values\":
2704,\n        \"samples\": [\n          54.65714872,\n
20.97239726,\n          11.65996434\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SpecialDay\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.19891727315262864,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          0.0,\n
0.4,\n          0.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Month\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 10,\n        \"samples\": [\n
\"Sep\",\n          \"Mar\",\n          \"Jul\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"OperatingSystems\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 1,\n        \"max\": 8,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          2,\n
6,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Browser\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 1,\n        \"min\": 1,\n        \"max\": 13,\n
\"num_unique_values\": 13,\n        \"samples\": [\n          13,\n
9,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Region\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2,\n        \"min\": 1,\n        \"max\": 9,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          7,\n
9,\n          5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

\"TrafficType\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4,\n   \"min\": 1,\n
\"max\": 20,\n   \"num_unique_values\": 20,\n   \"samples\":
[\n   1,\n   16,\n   18\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"VisitorType\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n
\"Returning_Visitor\",\n   \"New_Visitor\",\n
\"Other\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Weekend\",\n   \"properties\": {\n   \"dtype\":
\"boolean\",\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   true,\n   false\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Revenue\",\n   \"properties\":
{\n   \"dtype\": \"boolean\",\n   \"num_unique_values\": 2,\
n   \"samples\": [\n   true,\n   false\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"Administrative\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 4357.421533220783,\n
\"min\": 0.0,\n   \"max\": 12330.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   12330.0,\
n   2.3151662611516626,\n   4.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Administrative_Duration\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
4330.581827976747,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   12330.0,\
n   80.81861053933592,\n   93.25625\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Informational\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
4358.019452262415,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
0.5035685320356853,\n   24.0,\n   1.270156425983391\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Informational_Duration\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
4313.088078405079,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
34.47239792772304,\n   2549.375,\n   140.74929442219798\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"ProductRelated\",\n   \"properties\": {\n   \"dtype\":

\"number\",\n   \"std\": 4323.288444017446,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   31.731467964314678,\n   18.0,\n   12330.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ProductRelated_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 22099.576243757514,\n   \"min\": 0.0,\n   \"max\": 63973.52223,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   1194.7462199688268,\n   598.9369047499999,\n   12330.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"BounceRates\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4359.298628768896,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   12330.0,\n   0.02219138047072182,\n   0.016812558499999998\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ExitRates\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4359.294057418972,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   0.04307279776650446,\n   0.0251564025,\n   12330.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"PageValues\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4341.615343276191,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   5.889257862693592,\n   361.7637419,\n   18.568436607806525\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SpecialDay\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4359.24966237291,\n   \"min\": 0.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.061427412814274135,\n   1.0,\n   0.19891727315262864\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"OperatingSystems\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4358.352459689198,\n   \"min\": 0.911324828710665,\n   \"max\": 12330.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   12330.0,\n   2.124006488240065,\n   3.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Browser\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4358.099131873571,\n   \"min\": 1.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 6,\n   \"samples\": [\n   12330.0,\n   2.357096512570965,\n   13.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Region\",\n   \"properties\":

{\n   \"dtype\": \"number\",\n   \"std\": 4358.124632706003,\n   \"min\": 1.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   12330.0,\n   3.1473641524736413,\n   4.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"TrafficType\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 4357.444019425214,\n   \"min\": 1.0,\n   \"max\": 12330.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   12330.0,\n   4.069586374695864,\n   4.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df.head(5)
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 12330,\n \"fields\": [\n   {\n   \"column\": \"Administrative\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3,\n   \"min\": 0,\n   \"max\": 27,\n   \"num_unique_values\": 27,\n   \"samples\": [\n   5,\n   11,\n   9\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Administrative_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 176.77910747048634,\n   \"min\": 0.0,\n   \"max\": 3398.75,\n   \"num_unique_values\": 3335,\n   \"samples\": [\n   93.6,\n   63.08333333,\n   351.0833333\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Informational\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 24,\n   \"num_unique_values\": 17,\n   \"samples\": [\n   0,\n   1,\n   5\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Informational_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 140.74929442219798,\n   \"min\": 0.0,\n   \"max\": 2549.375,\n   \"num_unique_values\": 1258,\n   \"samples\": [\n   793.8,\n   50.0,\n   57.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ProductRelated\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 44,\n   \"min\": 0,\n   \"max\": 705,\n   \"num_unique_values\": 311,\n   \"samples\": [\n   330,\n   23,\n   54\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ProductRelated_Duration\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1913.6692878720035,\n   \"min\": 0.0,\n   \"max\": 63973.52223,\n   \"num_unique_values\": 9551,\n   \"samples\": [\n   225.68,\n   232.6666667,\n   2834.280117\n   ],\n

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BounceRates\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.04848832180626065 6,\n        \"min\": 0.0,\n        \"max\": 0.2,\n
\"num_unique_values\": 1872,\n        \"samples\": [\n
0.004778942,\n          0.010793651,\n          0.005073996\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ExitRates\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.048596540551443565,\n        \"min\":
0.0,\n        \"max\": 0.2,\n        \"num_unique_values\": 4777,\n
\"samples\": [\n          0.120833333,\n          0.006857143,\n
0.008914729\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PageValues\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 18.568436607806525,\n        \"min\":
0.0,\n        \"max\": 361.7637419,\n        \"num_unique_values\":
2704,\n        \"samples\": [\n          54.65714872,\n
20.97239726,\n          11.65996434\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SpecialDay\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.19891727315262864,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          0.0,\n
0.4,\n          0.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Month\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 10,\n        \"samples\": [\n
\"Sep\",\n          \"Mar\",\n          \"Jul\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"OperatingSystems\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 1,\n        \"max\": 8,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          2,\n
6,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Browser\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 1,\n        \"min\": 1,\n        \"max\": 13,\n
\"num_unique_values\": 13,\n        \"samples\": [\n          13,\n
9,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Region\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2,\n        \"min\": 1,\n        \"max\": 9,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          7,\n
9,\n          5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"TrafficType\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 4,\n        \"min\": 1,\n
\"max\": 20,\n        \"num_unique_values\": 20,\n        \"samples\":
```

```
[\n          1,\n          16,\n          18\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"VisitorType\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Returning_Visitor\",\n          \"New_Visitor\",\n
\"Other\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Weekend\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Revenue\",\n      \"properties\":
{\n        \"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\
n        \"samples\": [\n          true,\n          false\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
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
```

Checking for duplicate values

```
df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
12325    False
12326    False
12327    False
12328    False
12329    False
Length: 12330, dtype: bool
```

```
new_var = df.duplicated()
(new_var == True).any()
```

```
True
```

```
df.duplicated().sum()
```

```
125
```

There are 125 duplicated values

Checking for missing values

```
df.isnull().sum()
```

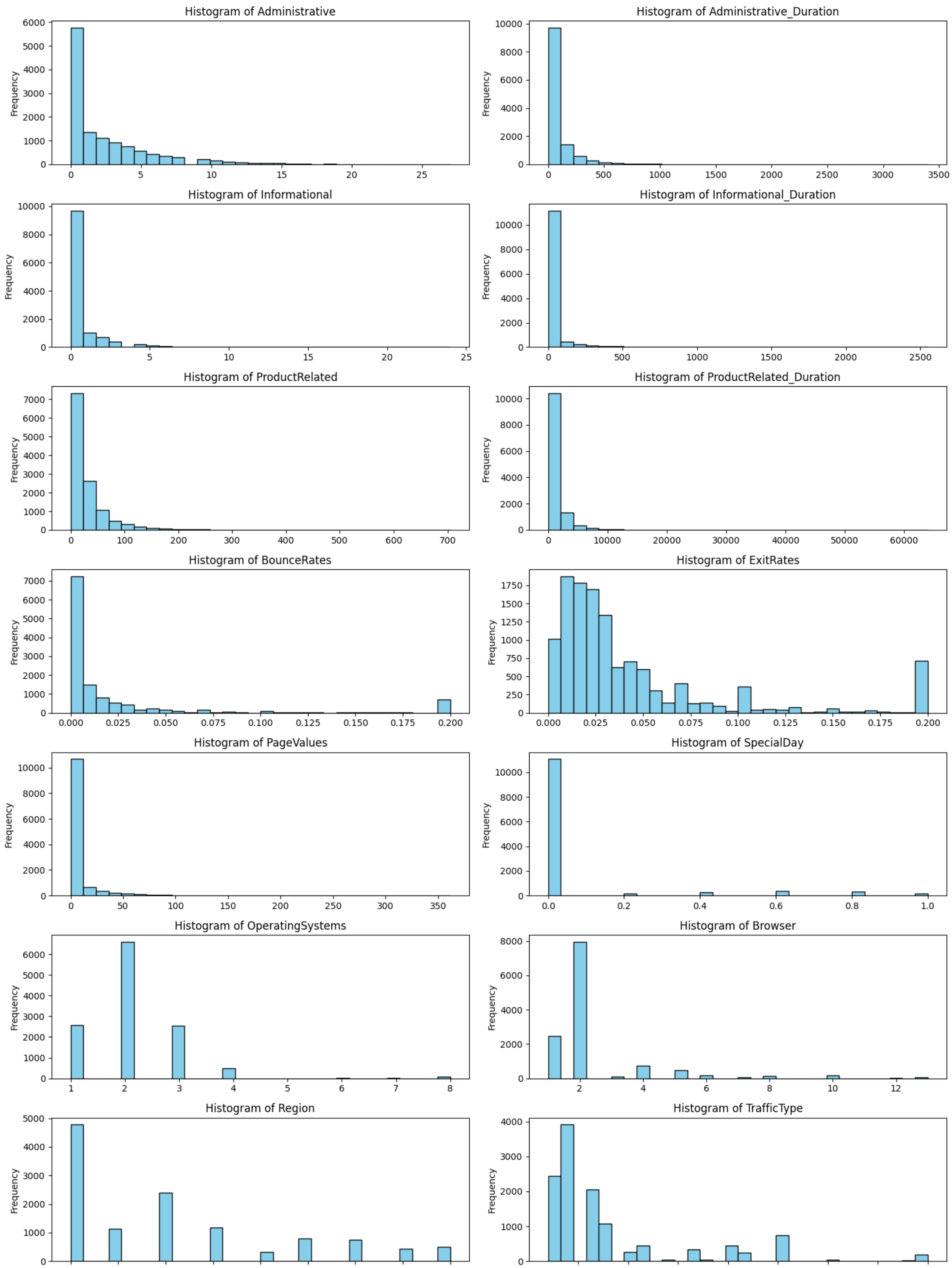
```
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

There are no missing values

Univariate analysis

Histogram

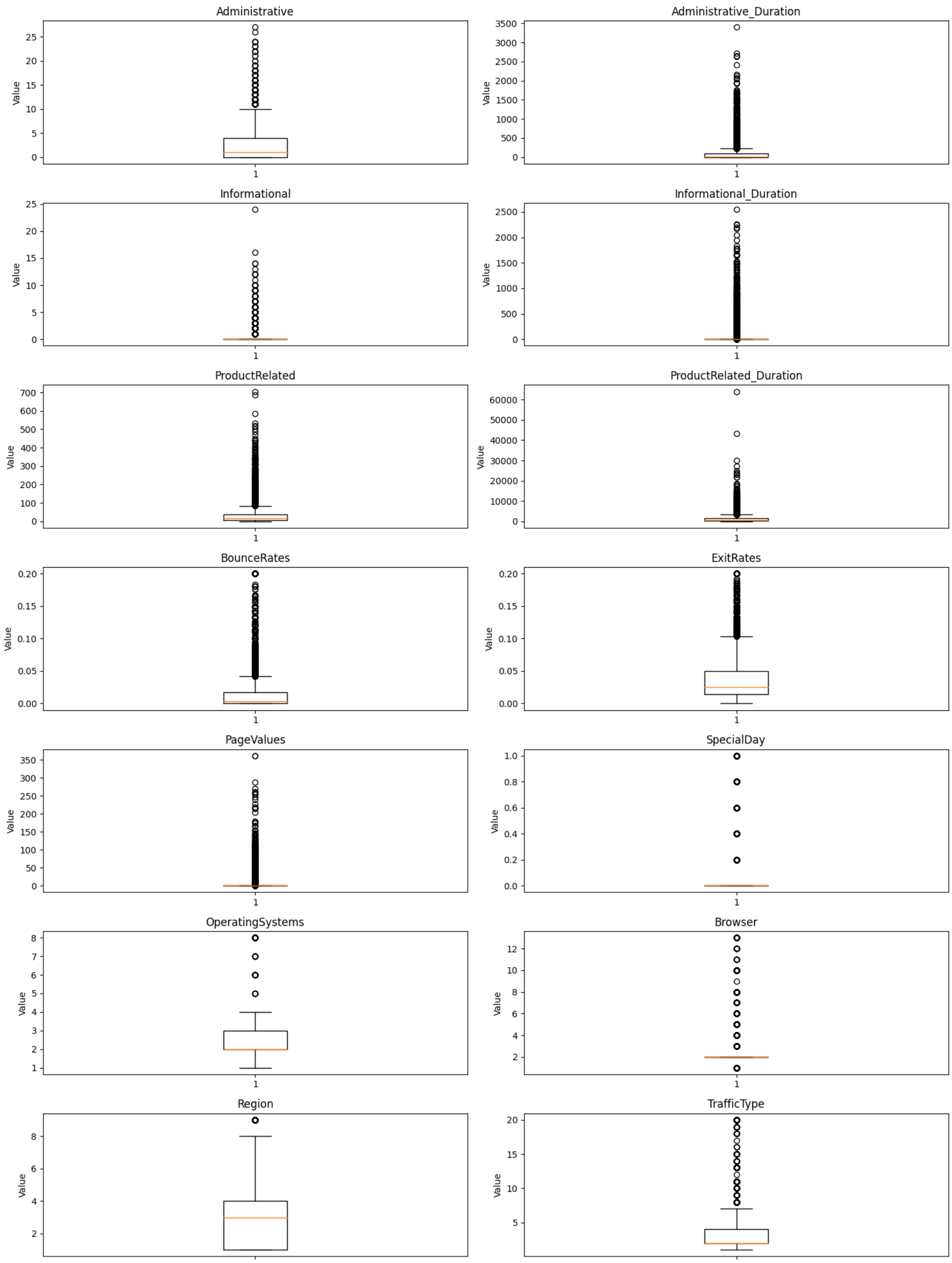
```
fig, axes = plt.subplots(nrows=7, ncols=2, figsize=(15, 20))

axes = axes.flatten()
numerical_columns = df.select_dtypes(include=['int', 'float']).columns

for i, column in enumerate(numerical_columns):
    ax = axes[i]
    ax.hist(df[column], bins=30, color='skyblue', edgecolor='black')
    ax.set_title(f'Histogram of {column}')
    ax.set_ylabel('Frequency')

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


Histogram of Administrative
Histogram of Administrative_Duration
Histogram of Informational
Histogram of Informational_Duration
Histogram of ProductRelated
Histogram of ProductRelated_Duration
Histogram of BounceRates
Histogram of ExitRates
Histogram of PageValues
Histogram of SpecialDay
Histogram of OperatingSystems
Histogram of Browser
Histogram of Region
Histogram of TrafficType
Frequency

The presence of right-skewed distributions in numerical columns of a shopping dataset indicates that a few high-spending customers significantly contribute to overall revenue.

Outliers Detections

```
fig, axes = plt.subplots(nrows=7, ncols=2, figsize=(15, 20))

axes = axes.flatten()

for i, column in enumerate(numerical_columns):
    ax = axes[i]
    ax.boxplot(df[column])
    ax.set_title(column)
    ax.set_ylabel('Value')

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


Administrative
Administrative_Duration
Informational
Informational_Duration
ProductRelated
ProductRelated_Duration
BounceRates
ExitRates
PageValues
SpecialDay
OperatingSystems
Browser
Region
TrafficType
Value
1

```
df.shape
```

```
(12330, 18)
```

```
outliers_count = {}

for column in numerical_columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] >
upper_bound)]
    outliers_count[column] = outliers.shape[0]

for column, count in outliers_count.items():
    print(f"Number of outliers in {column}: {count}")
```

```
Number of outliers in Administrative: 404
Number of outliers in Administrative_Duration: 1172
Number of outliers in Informational: 2631
Number of outliers in Informational_Duration: 2405
Number of outliers in ProductRelated: 987
Number of outliers in ProductRelated_Duration: 961
Number of outliers in BounceRates: 1551
Number of outliers in ExitRates: 1099
Number of outliers in PageValues: 2730
Number of outliers in SpecialDay: 1251
Number of outliers in OperatingSystems: 111
Number of outliers in Browser: 4369
Number of outliers in Region: 511
Number of outliers in TrafficType: 2101
```

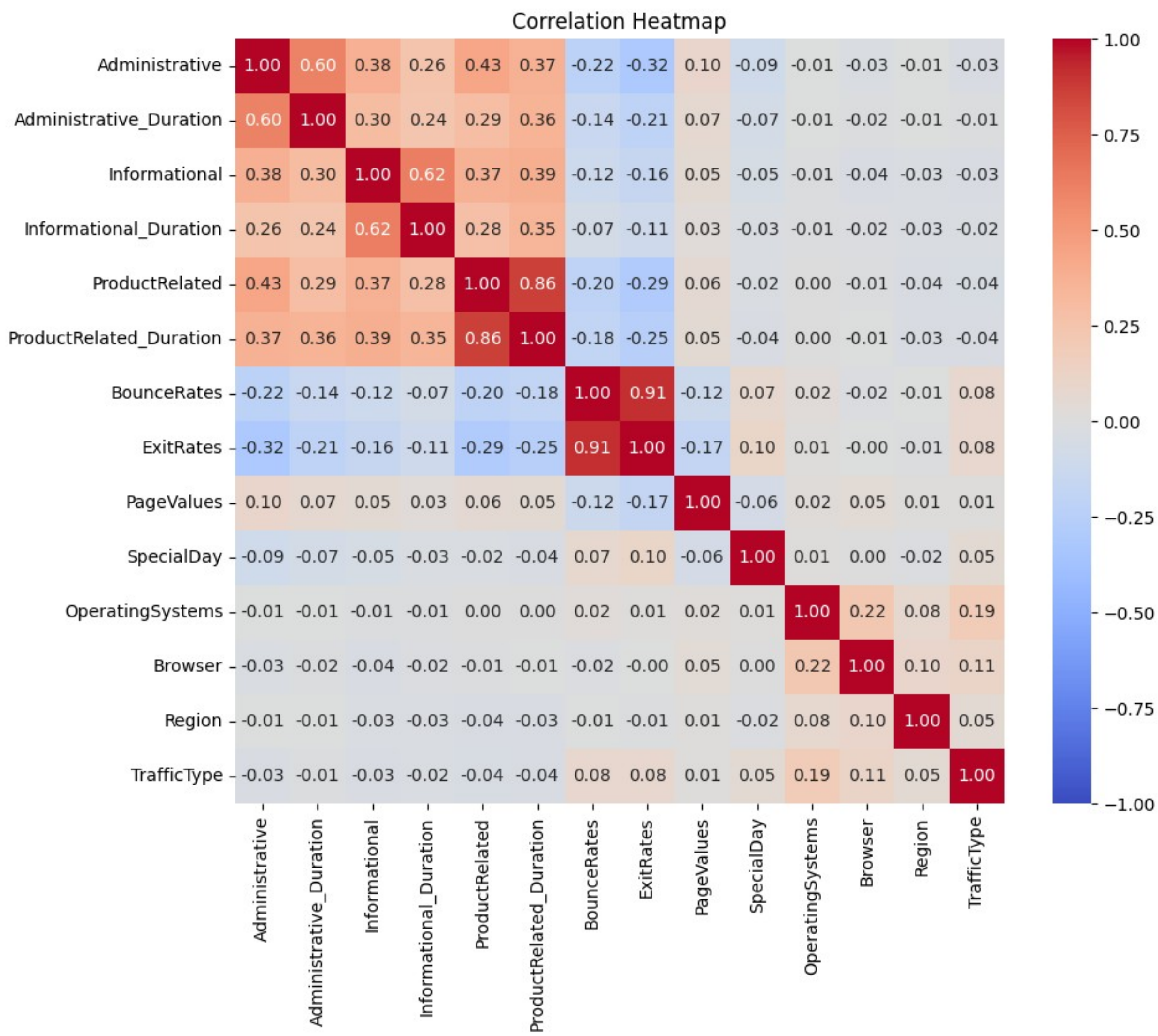
Insights

- Number of outliers in Administrative: 404
- Number of outliers in Administrative_Duration: 1172
- Number of outliers in Informational: 2631
- Number of outliers in Informational_Duration: 2405
- Number of outliers in ProductRelated: 987
- Number of outliers in ProductRelated_Duration: 961
- Number of outliers in BounceRates: 1551
- Number of outliers in ExitRates: 1099
- Number of outliers in PageValues: 2730
- Number of outliers in SpecialDay: 1251
- Number of outliers in OperatingSystems: 111

- Number of outliers in Browser: 4369
- Number of outliers in Region: 511
- Number of outliers in TrafficType: 2101

Correlation Matrix

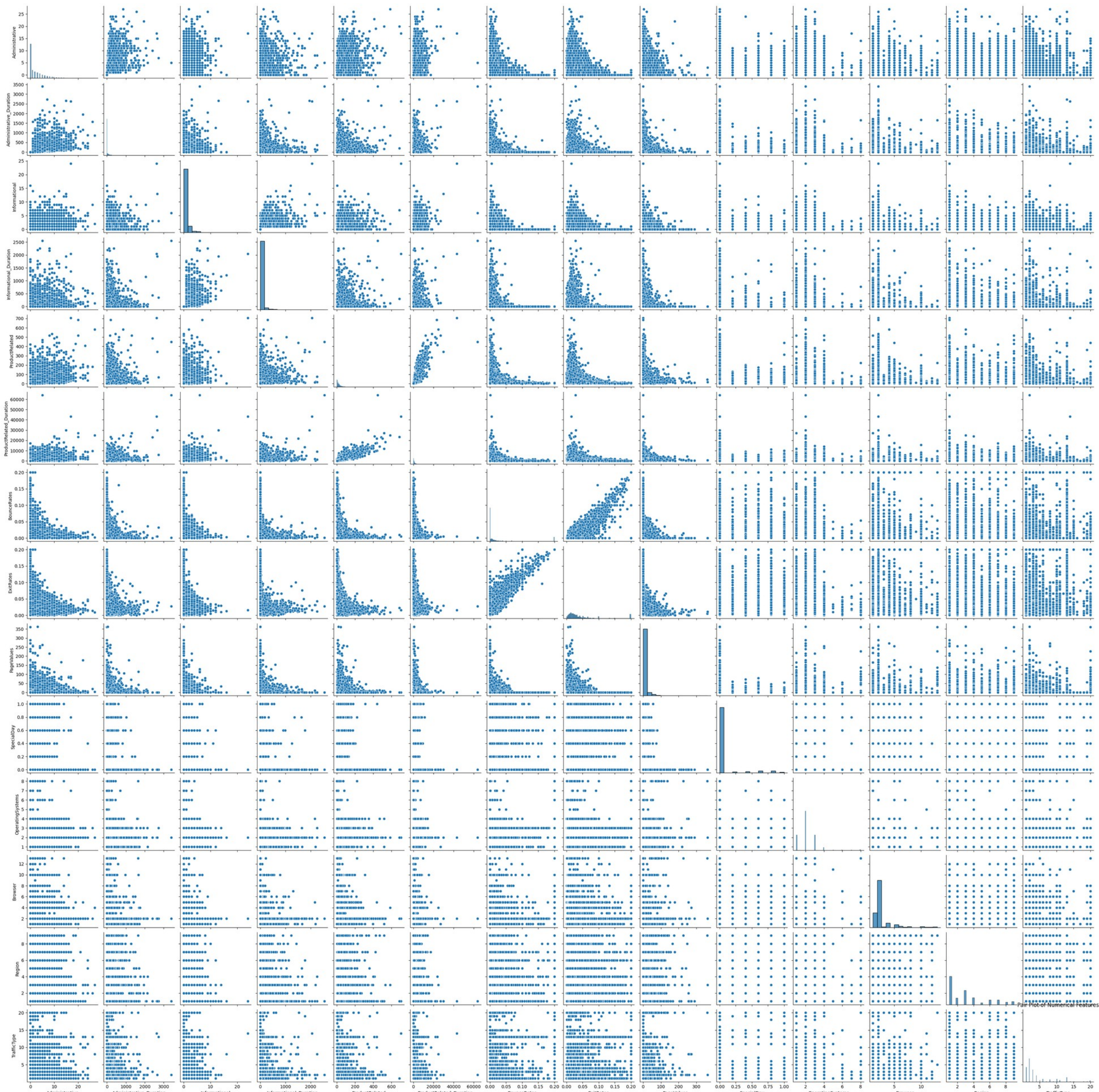
```
numerical_columns = df.select_dtypes(include=['int', 'float']).columns

correlation_matrix = df[numerical_columns].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f", vmin=-1, vmax=1)
plt.title("Correlation Heatmap")
plt.show()
```



Pairplot

```
plt.figure(figsize=(12, 10))
sns.pairplot(df[numerical_columns])
plt.title("Pair Plot of Numerical Features")
plt.show()
```

```
<Figure size 1200x1000 with 0 Axes>
```



Insights

Exit Rate and Bounce Rate (0.91):

- The strong positive correlation of 0.91 between Exit Rate and Bounce Rate indicates that pages with higher bounce rates also tend to have higher exit rates, and vice versa.
- This suggests that users who bounce from a page (i.e., leave without interacting further) are likely to exit the website from that same page.

Product Related Duration and Product Related Pages (0.86):

- The strong positive correlation of 0.86 between Product Related Duration and the number of Product Related Pages suggests that users who spend more time on product-related pages tend to view more of them.
- This implies that users engage more deeply with product-related content when they spend longer durations on these pages.

Informational Duration and Informational Pages (0.62):

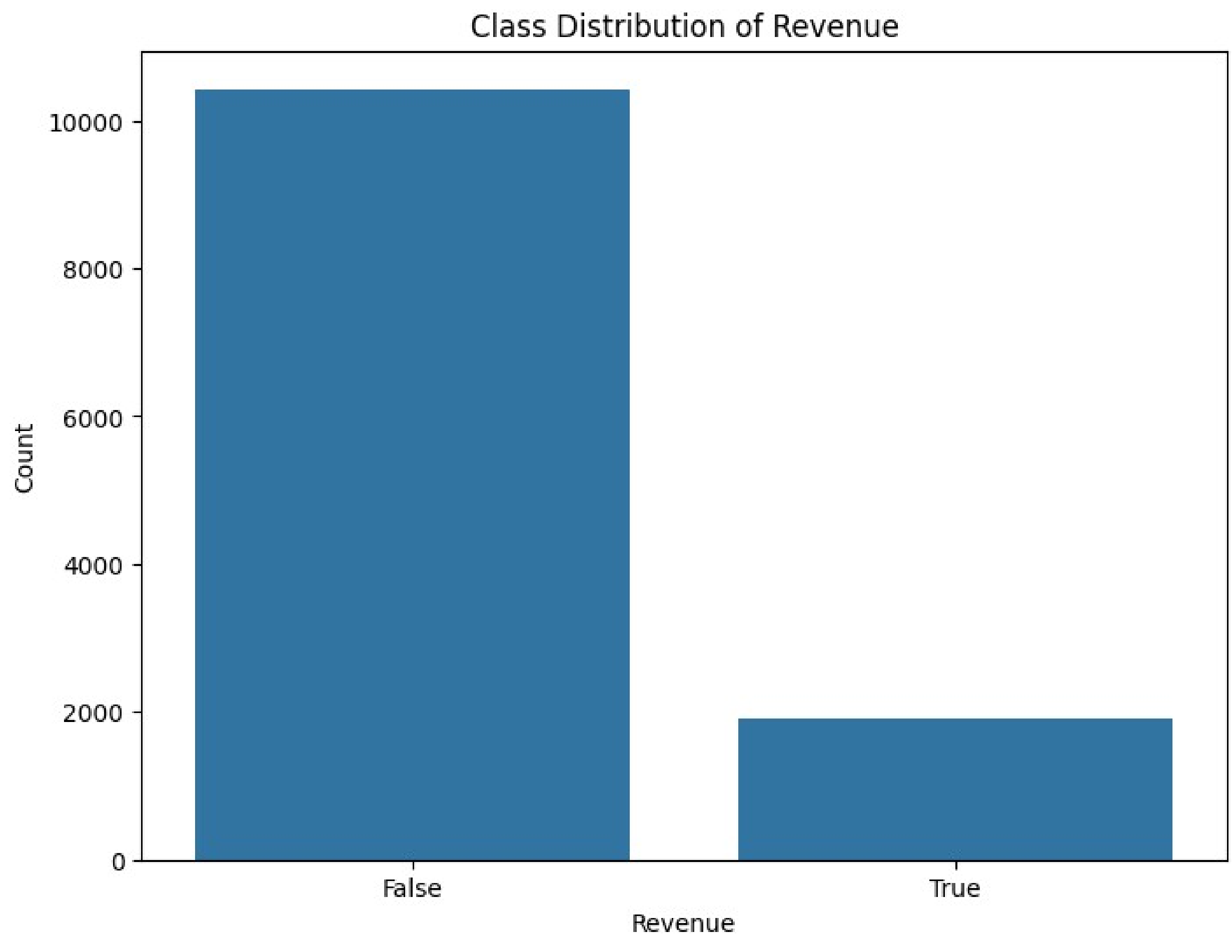
- The moderate positive correlation of 0.62 between Informational Duration and the number of Informational Pages suggests that users who spend more time on informational pages tend to view more of them.
- It indicates a relationship where users who engage longer with informational content explore more informational pages.

Administrative and Exit Rate (-0.32):

- The negative correlation of -0.32 between Administrative Pages and Exit Rate suggests an inverse relationship, where higher administrative page visits are associated with lower exit rates.
- Indicating that users who interact with administrative pages (e.g., account settings, checkout process) are less likely to exit the website immediately afterward.

Class Distribution of Revenue

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Revenue', data=df)
plt.title('Class Distribution of Revenue')
plt.xlabel('Revenue')
plt.ylabel('Count')
plt.show()
```



Insights

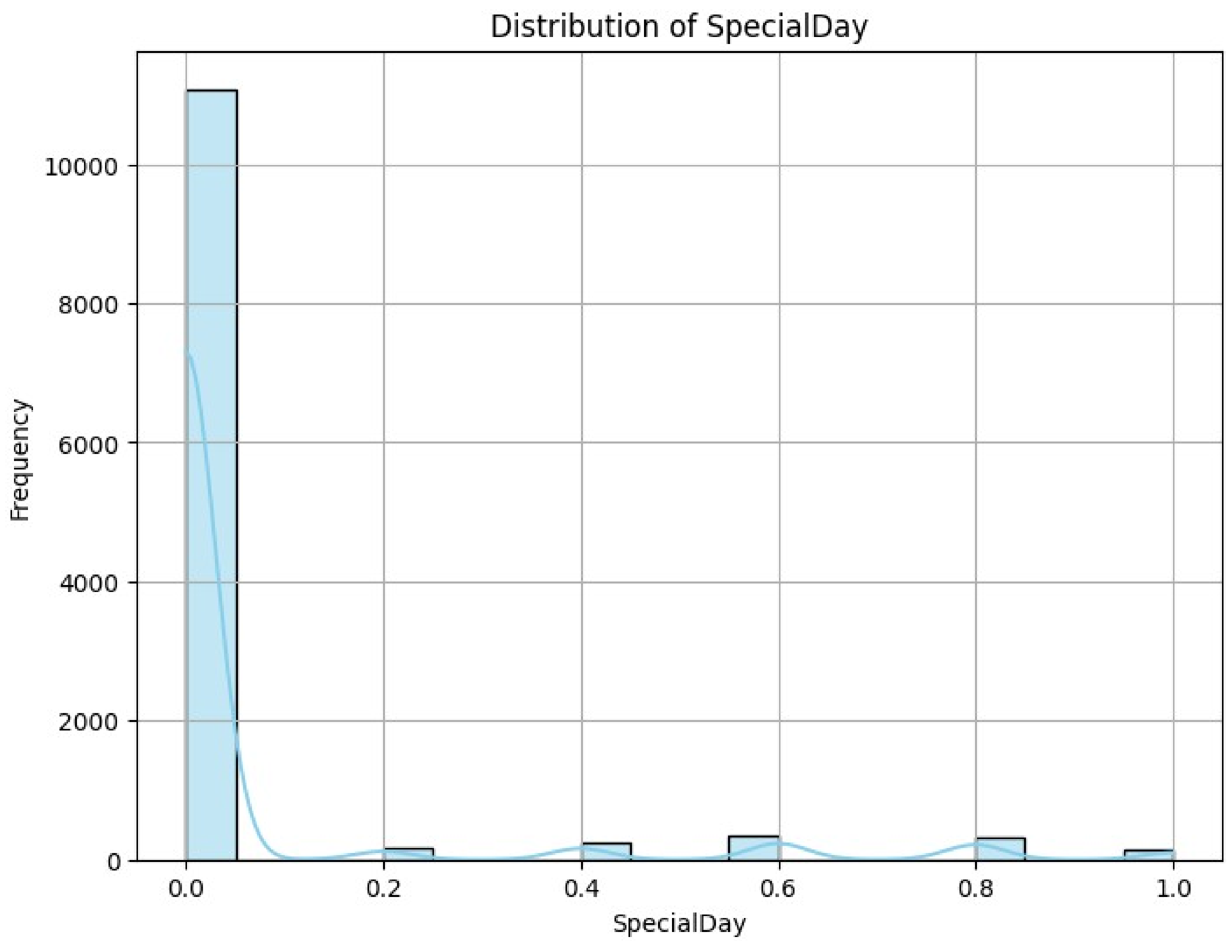
- A class imbalance indicates that the distribution of classes in the dataset is skewed, with one class being much more prevalent than the other. In your case, the 'False' class (indicating no revenue generated) is much more common than the 'True' class (indicating revenue generated).

Summarizing page views, durations, and bounce/exit rates for each page category

```
summary_stats = df.groupby('PageValues').agg(
    page_views=('PageValues', 'count'),
    duration_mean=('Administrative_Duration', 'mean'),
    bounce_rate_mean=('BounceRates', 'mean'),
    exit_rate_mean=('ExitRates', 'mean'))
print("Summary Statistics for Each Page Category:")
print(summary_stats)
```

```
Summary Statistics for Each Page Category:
            page_views  duration_mean  bounce_rate_mean
exit_rate_mean
PageValues

0.000000          9600      59.455400          0.026896
0.049783
0.038035             1     196.250000          0.005995
0.020011
0.067050             1     228.748539          0.000000
0.012144
0.093547             1     251.619048          0.000000
0.007000
0.098621             1      73.233333          0.000039
0.007511
...                ...            ...               ...               ..
.
261.491286           1     172.200000          0.000000
0.010714
270.784693           1       0.000000          0.000000
0.014286
287.953793           1       0.000000          0.000000
0.016667
360.953384           1       0.000000          0.000000
0.004762
361.763742           1      37.500000          0.000000
0.010526

[2704 rows x 4 columns]
```

```python
plt.figure(figsize=(8, 6))
sns.histplot(df['SpecialDay'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of SpecialDay')
plt.xlabel('SpecialDay')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Calculating correlation coefficient between SpecialDay and Revenue
correlation = df['SpecialDay'].corr(df['Revenue'])
print("Correlation between SpecialDay and Revenue:", correlation)
```




```
Correlation between SpecialDay and Revenue: -0.08230459817953266
```

Insights

- The correlation coefficient between 'SpecialDay' and 'Revenue' is approximately -0.082.
- Suggests a weak negative linear relationship between 'SpecialDay' and 'Revenue'. Hence, here is a slight tendency for revenue to decrease slightly as the proximity to special days or holidays increases, but the relationship is not strong.

Generating a binary feature indicating whether the user visited all three page categories.

```
df['Visited_All_Categories'] = (df['Administrative'] > 0) &
(df['Informational'] > 0) & (df['ProductRelated'] > 0)

df['Visited_All_Categories'] =
df['Visited_All_Categories'].astype(int) # Converting boolean values
```

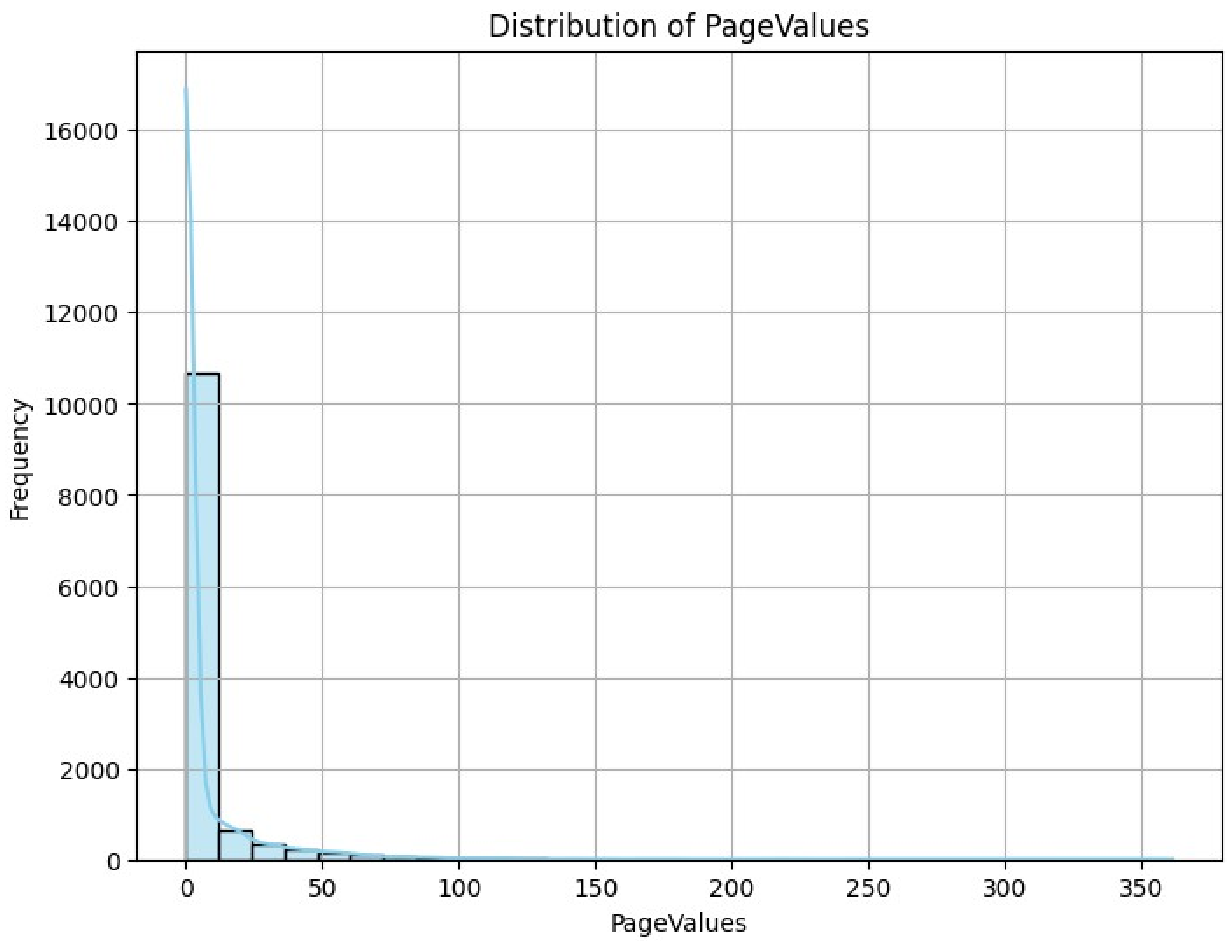
```
to binary (0 or 1)

df['Visited_All_Categories']

0        0
1        0
2        0
3        0
4        0
        ..
12325    0
12326    0
12327    0
12328    0
12329    0
Name: Visited_All_Categories, Length: 12330, dtype: int64

if df['Visited_All_Categories'].any():
    print("There are True values present in the column.")

true_count = df['Visited_All_Categories'].sum() # Counting the number
of True values in Visited_All_Categories column
print("Number of True values in the column:", true_count)

There are True values present in the column.
Number of True values in the column: 2167
```

Insights

- The presence of 2167 True values in the column 'Visited_All_Categories' indicates that a subset of users visited all three page categories: Administrative, Informational, and ProductRelated.

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

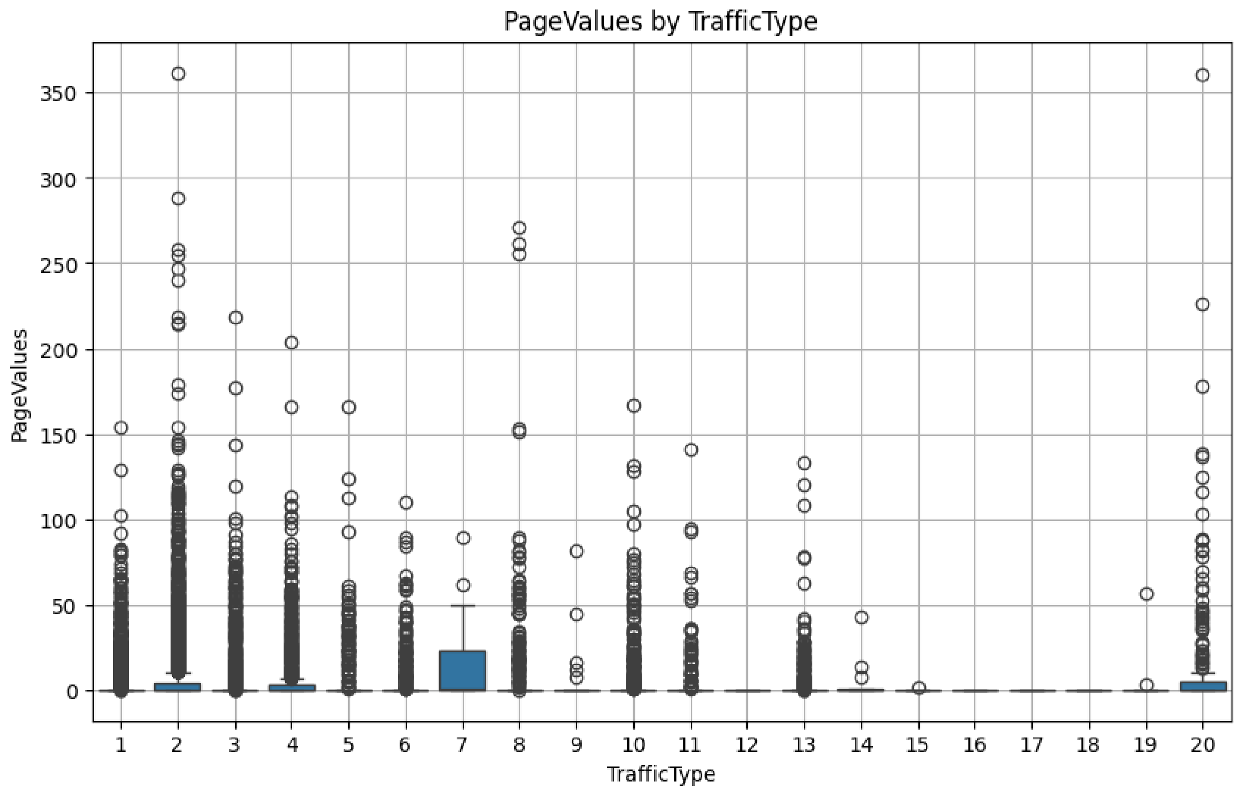
```
plt.figure(figsize=(8, 6))

sns.histplot(df['PageValues'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of PageValues')
plt.xlabel('PageValues')
plt.ylabel('Frequency')
plt.grid(True)

plt.show()
```



Insights

- The left-skewed distribution of PageValues suggests that there are relatively fewer instances with high PageValues compared to lower values.
- This indicates that a majority of page visits may not result in significant value generation for the website or platform.

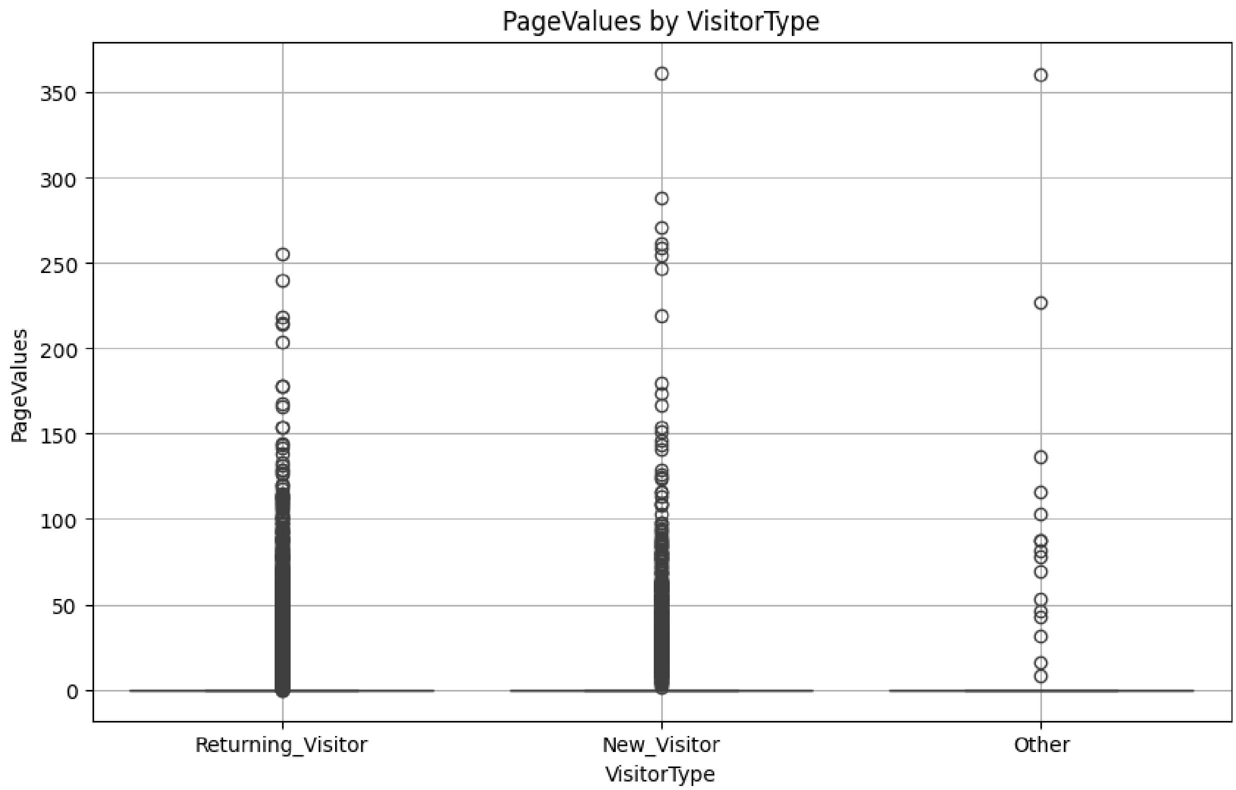
```
plt.figure(figsize=(10, 6)) # Relationship with TrafficType
sns.boxplot(x='TrafficType', y='PageValues', data=df)
plt.title('PageValues by TrafficType')
plt.xlabel('TrafficType')
plt.ylabel('PageValues')
plt.grid(True)
plt.show()
```



Insights

- The presence of maximum outliers in TrafficType 2 and 20 suggests that these traffic types may have a higher variation in PageValues compared to others. Indicating that certain sources of traffic (e.g., specific referral sources or ad campaigns) lead to more valuable interactions on the website.
- The absence of outliers in TrafficType 12, 16, 17, and 18 indicates that these traffic types may exhibit more consistent PageValues and fewer extreme values. Suggesting that these traffic types are less variable in terms of value generation.

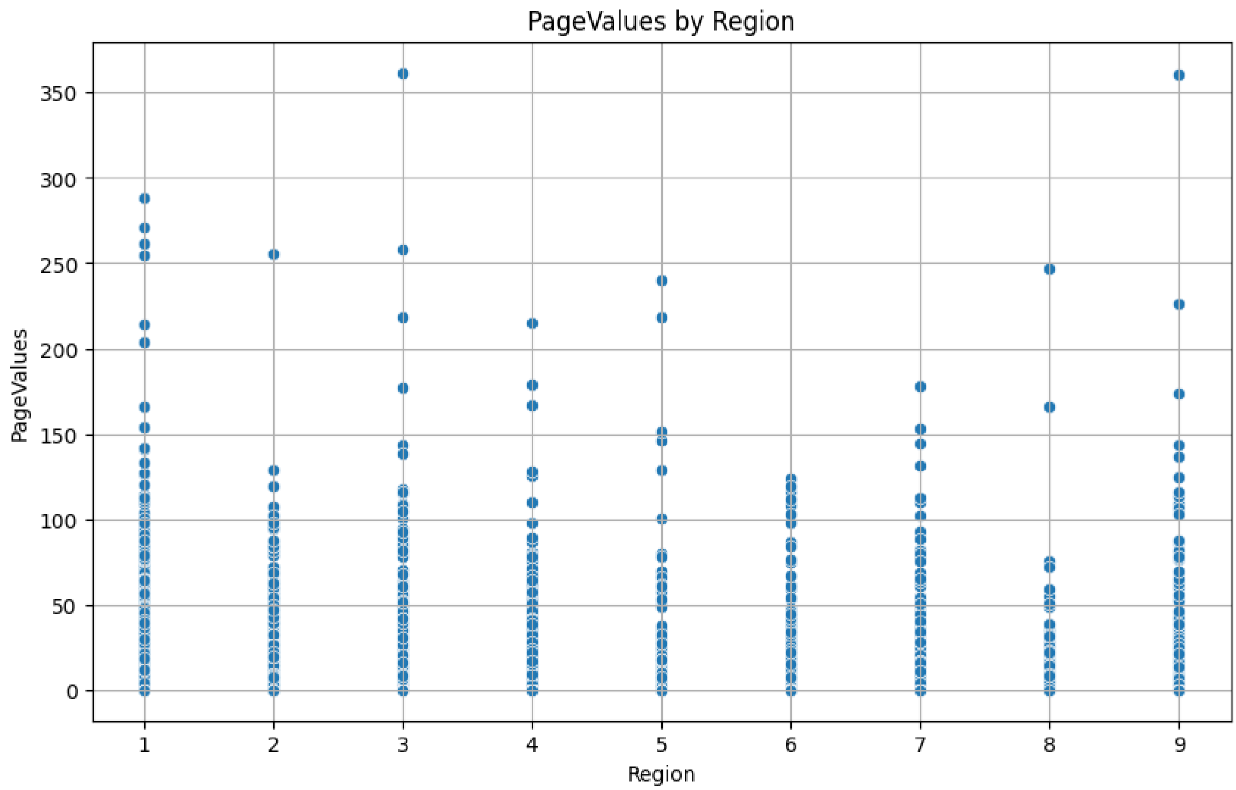
```
plt.figure(figsize=(10, 6)) # Relationship with VisitorType
sns.boxplot(x='VisitorType', y='PageValues', data=df)
plt.title('PageValues by VisitorType')
plt.xlabel('VisitorType')
plt.ylabel('PageValues')
plt.grid(True)
plt.show()
```



Insights

- The presence of maximum outliers in the New_Visitor VisitorType suggests that new visitors to the website may contribute disproportionately to high PageValues. Indicating that attracting and converting new visitors is particularly lucrative for the website.
- The least outliers in the Other VisitorType category suggest that this visitor segment may exhibit more consistent PageValues and fewer extreme values compared to New_Visitors and Returning_Visitors.

```
plt.figure(figsize=(10, 6)) # Relationship with Region
sns.scatterplot(x='Region', y='PageValues', data=df)
plt.title('PageValues by Region')
plt.xlabel('Region')
plt.ylabel('PageValues')
plt.grid(True)
plt.show()
```




```
correlation = df['PageValues'].corr(df['Region'])
print("Correlation between PageValues and Region:", correlation)

Correlation between PageValues and Region: 0.011315299468006896
```

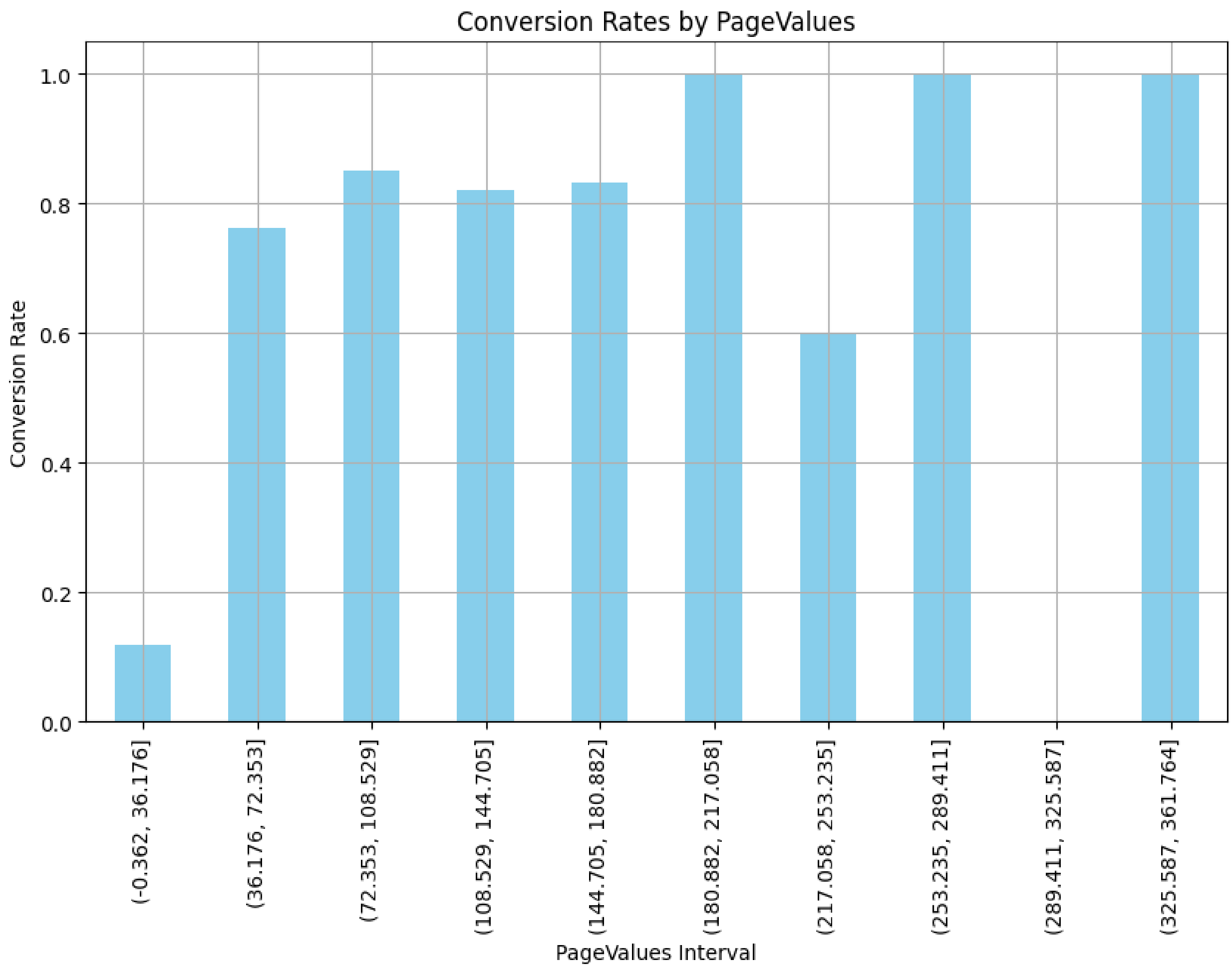
Insights

- It suggests that there is a very weak positive linear relationship between PageValues and Region.

Investigating user session lengths and their impact on conversion rates.

```
df.columns

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

```
conversion_rates = df.groupby(pd.cut(df['PageValues'], bins=10))
['Revenue'].mean()

plt.figure(figsize=(10, 6))
conversion_rates.plot(kind='bar', color='skyblue')
plt.title('Conversion Rates by PageValues')
plt.xlabel('PageValues Interval')
plt.ylabel('Conversion Rate')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



Insights

- PageValues intervals of (180-217), (253-289), and (325-361) have a conversion rate of 1, indicating that users with 'PageValues' within these intervals are highly likely to complete a conversion, Suggesting that users who generate higher 'PageValues' within these ranges are more inclined to make a purchase or complete a desired action on the website.

- The PageValues interval of (-0.36 - 36) has the lowest conversion rate of 0.1, indicating that users with 'PageValues' within this range are less likely to convert. This suggests that users with lower 'PageValues' may exhibit less purchase intent or engagement with the website's offerings.

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

```python
df['OperatingSystems'].unique()
```

```
array([1, 2, 4, 3, 7, 6, 8, 5])
```

```python
df['VisitorType'].unique()
```

```
array(['Returning_Visitor', 'New_Visitor', 'Other'], dtype=object)
```

```python
df['Region'].unique()
```

```
array([1, 9, 2, 3, 4, 5, 6, 7, 8])
```

```python
grouped_data = df.groupby(['VisitorType'])
```

```python
conversion_rates = grouped_data['Revenue'].mean().reset_index()
total_conversion = grouped_data['Revenue'].sum().reset_index()

print(conversion_rates)
print(" ")
print(total_conversion)
```

```
         VisitorType   Revenue
0        New_Visitor  0.249115
1              Other  0.188235
2  Returning_Visitor  0.139323

         VisitorType  Revenue
0        New_Visitor      422
1              Other       16
2  Returning_Visitor     1470
```

Insights

- New Visitors may have a higher conversion rate, but they represent a smaller proportion of the overall visitor base. As a result, while their conversion rate is higher, their contribution to the total number of conversions is lower compared to Returning Visitors.

- Returning Visitors contribute more conversions in absolute numbers due to their larger presence on the website, despite their lower conversion rate. New Visitors, on the other hand, have a higher conversion rate but contribute fewer conversions due to their smaller volume.

```
grouped_data = df.groupby(['OperatingSystems'])

conversion_rates = grouped_data['Revenue'].mean().reset_index()
total_conversion = grouped_data['Revenue'].sum().reset_index()

print(conversion_rates)
print(" ")
print(total_conversion)
```

	OperatingSystems	Revenue
0	1	0.146615
1	2	0.174973
2	3	0.104892
3	4	0.177824
4	5	0.166667
5	6	0.105263
6	7	0.142857
7	8	0.215190

	OperatingSystems	Revenue
0	1	379
1	2	1155
2	3	268
3	4	85
4	5	1
5	6	2
6	7	1
7	8	17

Insights

- Operating System 8 has the highest conversion rate of approximately 21.52%.
- Operating System 2 also has a relatively high conversion rate of approximately 17.50%.
- Operating System 3 and Operating System 6 have the lowest conversion rates of approximately 10.49% and 10.53% respectively.
- Conversion Rate: Operating System 8 stands out with the highest conversion rate, indicating that users on this operating system are more likely to complete a conversion. This suggests potential optimization opportunities or targeted marketing strategies tailored to users on this operating system.

- Conversion Volume: Although Operating System 8 has the highest conversion rate, its total number of conversions (17 conversions) is relatively low compared to Operating System 2 (1155 conversions). This indicates that while users on Operating System 8 are more likely to convert, they represent a smaller proportion of the overall visitor base.

```
grouped_data = df.groupby(['Region'])

conversion_rates = grouped_data['Revenue'].mean().reset_index()

print(conversion_rates)
```

	Region	Revenue
0	1	0.161297
1	2	0.165493
2	3	0.145235
3	4	0.148054
4	5	0.163522
5	6	0.139130
6	7	0.156373
7	8	0.129032
8	9	0.168297

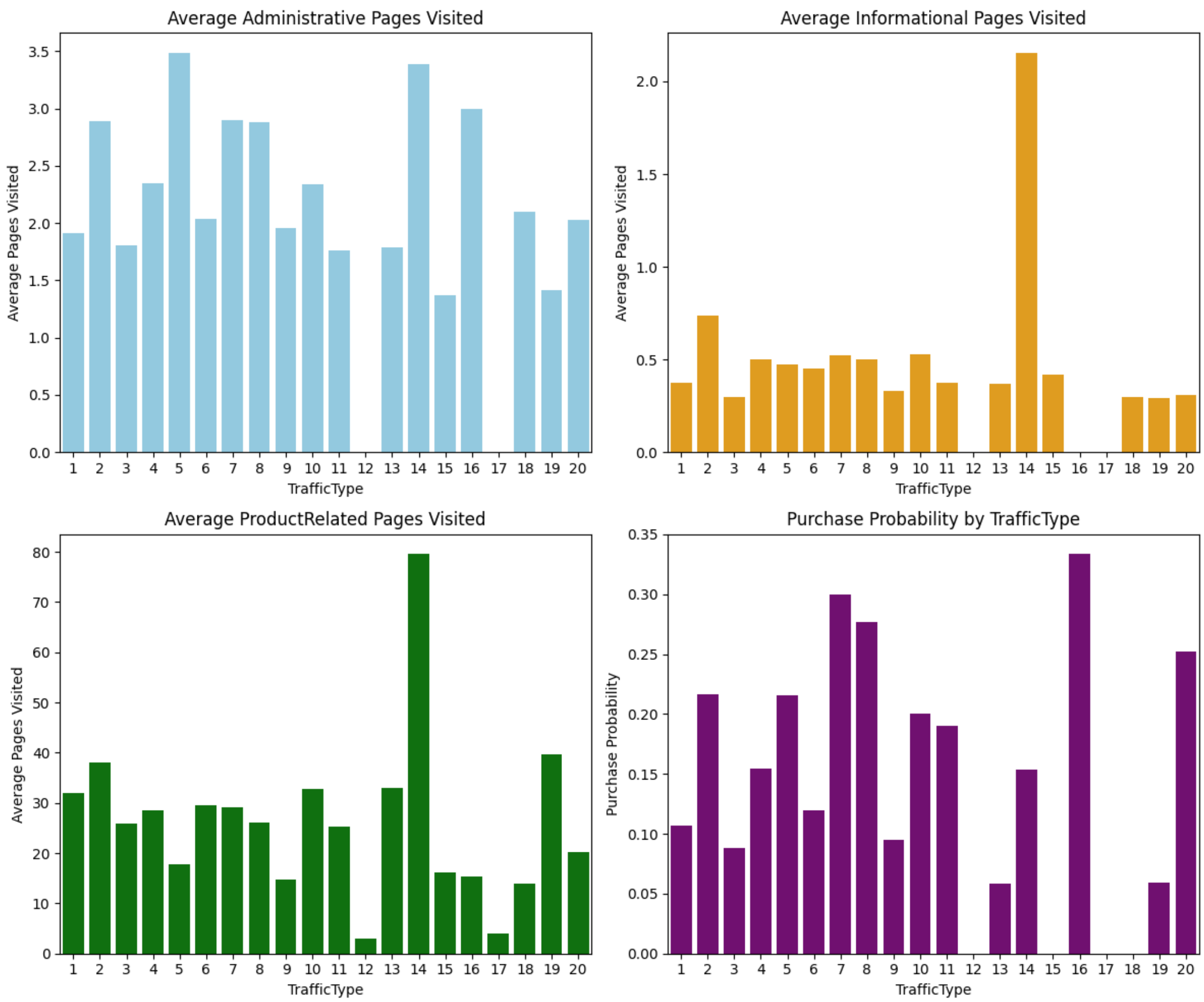
Insights

- Region 9 has the highest conversion rate of approximately 16.83%.
- Region 2 also has a relatively high conversion rate of approximately 16.55%.
- Region 8 has the lowest conversion rate of approximately 12.90%.
- Regions 9 and 2 stand out with relatively high conversion rates, suggesting that users from these regions may be more likely to complete a conversion.
- Regions with lower conversion rates, such as Region 8, may present opportunities for optimization efforts. Analyzing user behavior, preferences, and potential barriers to conversion in these regions can help identify strategies to improve conversion rates and overall performance.

Segmenting users based on TrafficType and analyzing their engagement patterns and purchase probability.

```
traffic_segments = df.groupby('TrafficType')

engagement_purchase_analysis = traffic_segments.agg({
    'Administrative': 'mean',
    'Informational': 'mean',
    'ProductRelated': 'mean',
    'Revenue': 'mean'
}).reset_index()
```

```python
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

sns.barplot(x='TrafficType', y='Administrative',
data=engagement_purchase_analysis, color='skyblue', ax=axes[0, 0])
axes[0, 0].set_title('Average Administrative Pages Visited')
axes[0, 0].set_xlabel('TrafficType')
axes[0, 0].set_ylabel('Average Pages Visited')

sns.barplot(x='TrafficType', y='Informational',
data=engagement_purchase_analysis, color='orange', ax=axes[0, 1])
axes[0, 1].set_title('Average Informational Pages Visited')
axes[0, 1].set_xlabel('TrafficType')
axes[0, 1].set_ylabel('Average Pages Visited')

sns.barplot(x='TrafficType', y='ProductRelated',
data=engagement_purchase_analysis, color='green', ax=axes[1, 0])
axes[1, 0].set_title('Average ProductRelated Pages Visited')
axes[1, 0].set_xlabel('TrafficType')
axes[1, 0].set_ylabel('Average Pages Visited')

sns.barplot(x='TrafficType', y='Revenue',
data=engagement_purchase_analysis, color='purple', ax=axes[1, 1])
axes[1, 1].set_title('Purchase Probability by TrafficType')
axes[1, 1].set_xlabel('TrafficType')
axes[1, 1].set_ylabel('Purchase Probability')

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



Insights

- TrafficType 4 has the highest average administrative pages visited, indicating that users arriving from this traffic source tend to explore administrative pages more extensively. This could imply that users from TrafficType 4 may be more interested in account management, settings, or other administrative tasks.
- TrafficType 14 has the highest average informational pages visited, suggesting that users from this traffic source are more inclined towards seeking information or browsing informational content on the website.
- TrafficType 3 has the lowest average informational pages visited, indicating that users from this traffic source may have a lower interest in informational content compared to other sources.

- TrafficType 14 also has the highest average product-related pages visited, indicating that users from this traffic source exhibit a higher level of engagement with product-related content on the website.
- TrafficType 12 has the lowest average product-related pages visited, suggesting that users from this traffic source may be less interested in product-related content or may have different browsing preferences.
- TrafficType 16 has the highest purchase probability, indicating that users arriving from this traffic source are more likely to make a purchase or complete a conversion. Suggesting that TrafficType 16 may represent a valuable source of high-intent traffic with a strong potential for generating revenue.
- TrafficType 13 has the lowest purchase probability, suggesting that users from this traffic source are less likely to convert or make a purchase.

Recommendations

- Pages with high bounce rates may need optimization to retain users and reduce exits, potentially by improving content relevance, load times, or navigation ease.
- Enhancing the quality, relevance, and presentation of product-related content may encourage users to explore more products, potentially leading to increased conversions.
- Improving the depth and quality of informational content could increase user engagement and promote further exploration of relevant information.
- Streamlining administrative processes, improving navigation to these pages, or providing clear calls-to-action may reduce exit rates and improve overall user experience.
- The presence of 2167 True values in the column 'Visited_All_Categories' indicates that a subset of users visited all three page categories: Administrative, Informational, and ProductRelated. ** Comprehensive Engagement: Users visiting all categories show thorough engagement. ** Behavior Patterns: Analyzing these users reveals valuable behavior patterns. ** Conversion Potential: They may have higher conversion potential. ** Optimization Focus: Insights guide content and site optimization efforts. ** Targeted Strategies: Segmentation enables tailored marketing strategies.
- Pages with higher 'PageValues' (e.g., 180-217, 253-289, 325-361) likely offer more value or engagement opportunities to users, leading to higher conversion rates. Understanding the characteristics of pages within these high-conversion intervals can inform content optimization strategies and identify factors driving user engagement and conversions.

- Operating System 2 has both a high conversion rate and a substantial number of conversions, making it a prime candidate for optimization efforts.

Areas of Improvement

- Pages with lower 'PageValues' may require optimization efforts to enhance user experience, increase engagement, and improve conversion rates.
- Understanding regional differences in conversion rates can inform targeted marketing strategies and campaigns tailored to the preferences and behaviors of users in specific regions. By focusing resources on regions with higher conversion rates, businesses can optimize marketing ROI and drive growth.
- Efforts can be made to improve conversion rates for users with lower 'PageValues' through targeted interventions and optimizations.
- Operating Systems 3, 5, 6, and 7 have lower conversion rates and contribute fewer conversions, suggesting potential areas for improvement or further investigation.

```
from google.colab import files
uploaded = files.upload()

<IPython.core.display.HTML object>

Saving campaign - campaign.csv to campaign - campaign (1).csv

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime, timedelta

df=pd.read_csv('campaign - campaign.csv')
```

EDA

```
df.head()
```

{"type":"dataframe","variable_name":"df"}

```
df.shape

(2239, 27)

df.info()

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

```
 19  NumWebVisitsMonth    2239 non-null   int64 
 20  AcceptedCmp3         2239 non-null   int64 
 21  AcceptedCmp4         2239 non-null   int64 
 22  AcceptedCmp5         2239 non-null   int64 
 23  AcceptedCmp1         2239 non-null   int64 
 24  AcceptedCmp2         2239 non-null   int64 
 25  Complain             2239 non-null   int64 
 26  Country              2239 non-null   object
dtypes: int64(22), object(5)
memory usage: 472.4+ KB
```

```
df.describe()
```

{"type":"dataframe"}

Removing the dollar sign in the Income column

```
# Remove dollar sign ($) and commas from 'Income' column
df['Income'] = df['Income'].str.replace('$', '').str.replace(',', '')

# Convert 'Income' column to numeric data type with handling for 
errors
df['Income'] = pd.to_numeric(df['Income'], errors='coerce')

# Display the first few rows to verify the changes
print(df['Income'].head())
```

```
0    84835.0
1    57091.0
2    67267.0
3    32474.0
4    21474.0
Name: Income, dtype: float64
```

Checking for Duplicate Values

```
duplicate_rows = df.duplicated()

num_duplicates = duplicate_rows.sum()

print(num_duplicates)
```

```
0
```

- There are no duplicate values

Checking for Missing Values

```
missing_values = df.isnull()
```

```
num_missing_values = missing_values.sum()

print("Number of missing values in each column:")
print(num_missing_values)
```

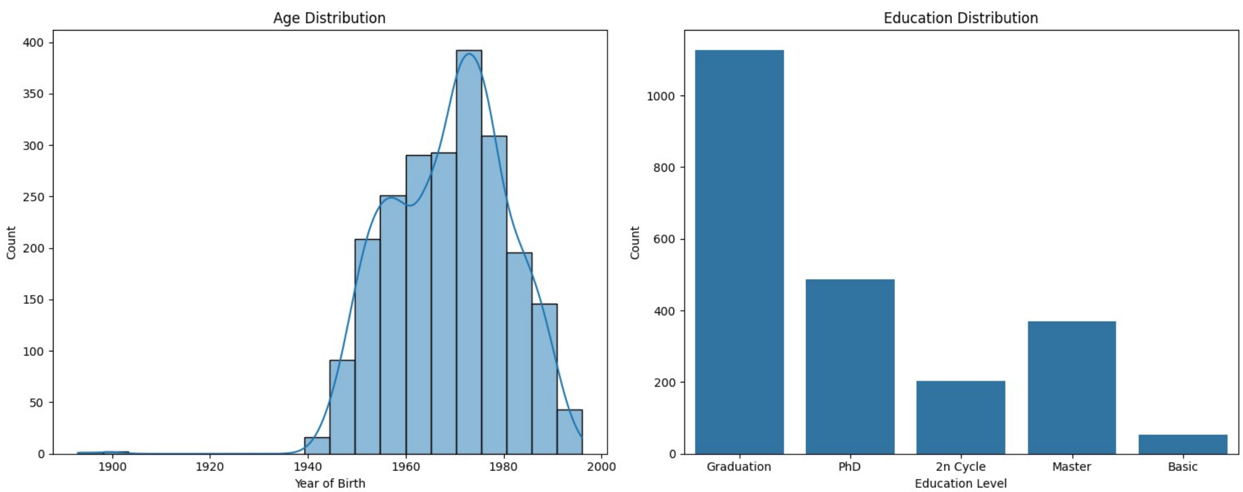
```
Number of missing values in each column:
ID                              0
Year_Birth                      0
Education                       0
Marital_Status                  0
Income                          0
Kidhome                         0
Teenhome                        0
Dt_Customer                     0
Recency                         0
MntWines                        0
MntFruits                       0
MntMeatProducts                 0
MntFishProducts                 0
MntSweetProducts                0
MntGoldProds                    0
NumDealsPurchases               0
NumWebPurchases                 0
NumCatalogPurchases             0
NumStorePurchases               0
NumWebVisitsMonth               0
AcceptedCmp3                    0
AcceptedCmp4                    0
AcceptedCmp5                    0
AcceptedCmp1                    0
AcceptedCmp2                    0
Complain                        0
Country                         0
Age                             0
Age_Group                       0
Total_Amount_Spent              0
Total_Purchases                 0
Total_Campaign_Acceptance       0
Total_Dependents                0
Days_Since_Last_Purchase        0
IncomeBracket                   0
EverAcceptedCampaign            0
dtype: int64
```

- There are no missing values in the Dataset.

Univariate Analysis

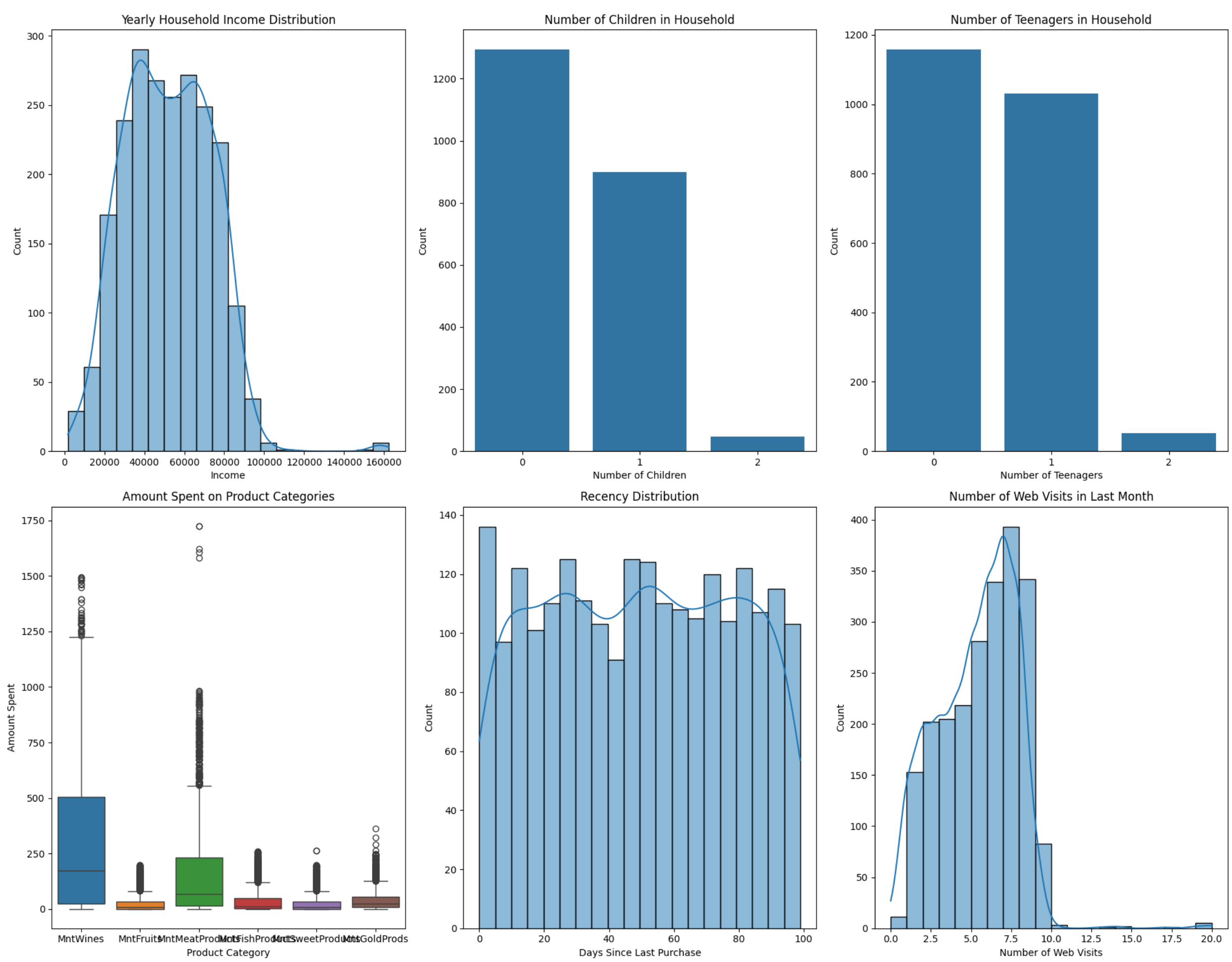
```
fig, axes = plt.subplots(1, 2, figsize=(15, 6))
```

```
sns.histplot(data=df, x='Year_Birth', bins=20, kde=True, ax=axes[0])
axes[0].set_title('Age Distribution')
axes[0].set_xlabel('Year of Birth')
axes[0].set_ylabel('Count')

sns.countplot(data=df, x='Education', ax=axes[1])
axes[1].set_title('Education Distribution')
axes[1].set_xlabel('Education Level')
axes[1].set_ylabel('Count')

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




```
fig, axes = plt.subplots(2, 3, figsize=(18, 14))


sns.histplot(data=df, x='Income', bins=20, kde=True, ax=axes[0, 0])
axes[0, 0].set_title('Yearly Household Income Distribution')
axes[0, 0].set_xlabel('Income')
axes[0, 0].set_ylabel('Count')

sns.countplot(data=df, x='Kidhome', ax=axes[0, 1])
axes[0, 1].set_title('Number of Children in Household')
axes[0, 1].set_xlabel('Number of Children')
axes[0, 1].set_ylabel('Count')

sns.countplot(data=df, x='Teenhome', ax=axes[0, 2])
axes[0, 2].set_title('Number of Teenagers in Household')
axes[0, 2].set_xlabel('Number of Teenagers')
axes[0, 2].set_ylabel('Count')

sns.boxplot(data=df[['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']], ax=axes[1,
0])
```

```
axes[1, 0].set_title('Amount Spent on Product Categories')
axes[1, 0].set_xlabel('Product Category')
axes[1, 0].set_ylabel('Amount Spent')

sns.histplot(data=df, x='Recency', bins=20, kde=True, ax=axes[1, 1])
axes[1, 1].set_title('Recency Distribution')
axes[1, 1].set_xlabel('Days Since Last Purchase')
axes[1, 1].set_ylabel('Count')

sns.histplot(data=df, x='NumWebVisitsMonth', bins=20, kde=True,
ax=axes[1, 2])
axes[1, 2].set_title('Number of Web Visits in Last Month')
axes[1, 2].set_xlabel('Number of Web Visits')
axes[1, 2].set_ylabel('Count')

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



Feature Enginneering

```
#Age
df['Age'] = 2021 - df['Year_Birth']

# Total Amount Spent
df['Total_Amount_Spent'] = df[['MntWines', 'MntFruits',
'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
'MntGoldProds']].sum(axis=1)

# Total Purchases
df['Total_Purchases'] = df[['NumWebPurchases', 'NumCatalogPurchases',
'NumStorePurchases']].sum(axis=1)

# Total Campaign Acceptance
df['Total_Campaign_Acceptance'] = df[['AcceptedCmp1', 'AcceptedCmp2',
'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1)

# Total Children and Teenagers
df['Total_Dependents'] = df['Kidhome'] + df['Teenhome']

df['Marital_Status'].replace({'Married': 'In couple', 'Together': 'In
couple','Divorced': 'Alone', 'Single': 'Alone','Absurd': 'Alone',
'Widow': 'Alone', 'YOLO': 'Alone'},inplace=True)

missing_values = df.isnull()

num_missing_values = missing_values.sum()

print("Number of missing values in each column:")
print(num_missing_values)
```

```
Number of missing values in each column:
ID                             0
Year_Birth                     0
Education                      0
Marital_Status                 0
Income                         0
Kidhome                        0
Teenhome                       0
Dt_Customer                    0
Recency                        0
MntWines                       0
MntFruits                      0
MntMeatProducts                0
MntFishProducts                0
MntSweetProducts               0
MntGoldProds                   0
NumDealsPurchases              0
NumWebPurchases                0
NumCatalogPurchases            0
```

```
NumStorePurchases              0
NumWebVisitsMonth              0
AcceptedCmp3                   0
AcceptedCmp4                   0
AcceptedCmp5                   0
AcceptedCmp1                   0
AcceptedCmp2                   0
Complain                       0
Country                        0
Age                            0
Age_Group                      0
Total_Amount_Spent             0
Total_Purchases                0
Total_Campaign_Acceptance      0
Total_Dependents               0
Days_Since_Last_Purchase       0
IncomeBracket                  0
EverAcceptedCampaign           0
dtype: int64
```

After changing the datatype of 'Income' column, we se that there are missing values in the Income and Age_Group column

```
# Replace missing values with the mean of the 'Income' column
mean_income = df['Income'].mean()
df['Income'].fillna(mean_income, inplace=True)

df['Marital_Status'].unique()

array(['Alone', 'In couple'], dtype=object)
```

Statistical analysis no.of days between current and last purchase date

```
df[['Total_Amount_Spent','Income','Age']].describe()
```


```
{"summary":"{\n  \"name\":
\"df[['Total_Amount_Spent','Income','Age']]\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"Total_Amount_Spent\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
953.5741677600665,\n        \"min\": 5.0,\n        \"max\": 2525.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
606.0410897722197,\n          396.0,\n          2239.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Income\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
51559.85414840027,\n        \"min\": 1730.0,\n        \"max\":
162397.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
51969.86139954853,\n          51717.0,\n          2239.0\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 773.4697671856804,\n
\"min\": 11.985494466990987,\n        \"max\": 2239.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
52.19785618579723,\n          51.0,\n          2239.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

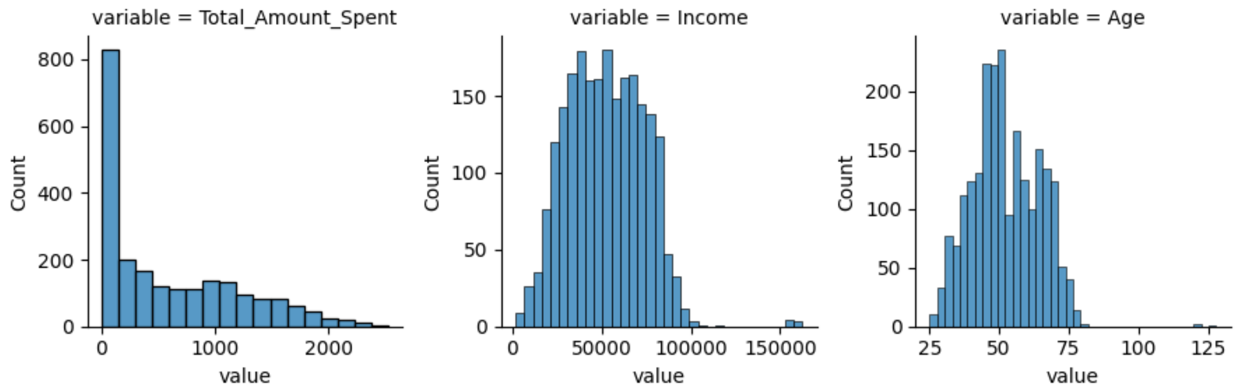

Insights

- Total_Amount_Spent ** The mean amount spent by customers is approximately is 602 dollars, with a median of 396. This indicates there are outliers in the Total_Amount_Spent column. ** The minimum amount spent is 5 dollars , while the maximum is 2525 dollars. The range of spending is quite wide, suggesting varying levels of engagement with the company's products.

- Income ** The mean income of customers is approximately 51,970, with a standard deviation of 21,410. This indicates variability in income levels among customers. ** The minimum income is $1,730, while the maximum is 162,397. The income range is quite wide, reflecting the diversity of customers' financial situations.

- Age ** The mean age of customers is approximately 52 years, with a standard deviation of approximately 12 years. This indicates variability in the ages of customers. ** The minimum age is 25 years, while the maximum age is 128 years. The age range also varies widely, suggesting a diverse customer base in terms of age demographics.

```
df2 = pd.DataFrame(data=df,
columns=['Total_Amount_Spent','Income','Age'])
nd = pd.melt(df2, value_vars = df2)
n1 = sns.FacetGrid(nd, col='variable',col_wrap=4, sharex= False,
sharey=False)
n1 = n1.map(sns.histplot,'value')
n1

<seaborn.axisgrid.FacetGrid at 0x7f201d1d49d0>
```




```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])

current_date = datetime.now()
last_purchase_dates = current_date - pd.to_timedelta(df['Recency'],
unit='d')

df['Days_Since_Last_Purchase'] = (current_date -
last_purchase_dates).dt.days

df['Days_Since_Last_Purchase'].describe()

count    2239.000000
mean       49.121036
std        28.963662
min         0.000000
25%        24.000000
50%        49.000000
75%        74.000000
max        99.000000
Name: Days_Since_Last_Purchase, dtype: float64
```

Insights

- Mean: On average, customers make a purchase approximately 49 days after their last purchase. This indicates the typical frequency of repeat purchases.
- Standard Deviation: The standard deviation of approximately 29 days suggests that there is some variability in the time interval between purchases across customers. Some customers make purchases more frequently than others.
- Minimum: The minimum value of 0 days indicates that some customers made a purchase on the same day as their last purchase. This could indicate frequent purchasing behavior or multiple purchases in a short time span.
- 25th, 50th (median), and 75th Percentiles: These percentiles provide insights into the distribution of the time interval between purchases. For example, 25% of customers make a purchase within 24 days of their last purchase, while 50% of customers make a purchase within 49 days.

- Maximum: The maximum value of 99 days suggests that some customers have a longer gap between purchases, possibly indicating less frequent purchasing behavior or periods of inactivity.

Comparing statistics across Educations

```
segment_stats = df.groupby('Education')
['Days_Since_Last_Purchase'].describe()
print(segment_stats)

             count       mean        std  min   25%   50%    75%   max
Education
2n Cycle     201.0  48.293532  30.129693  0.0  24.0  45.0  77.00  99.0
Basic         54.0  48.444444  26.649129  2.0  29.0  48.0  68.50  94.0
Graduation  1126.0  50.059503  28.831357  0.0  26.0  50.0  75.00  99.0
Master       370.0  47.586486  29.344036  0.0  22.0  49.0  73.75  98.0
PhD          485.0  48.509278  28.741050  0.0  23.0  49.0  72.00  99.0
```

Insights

- Mean: The average number of days since the last purchase varies slightly across education levels. Customers with a Master's degree have the lowest mean days since the last purchase, while customers with a PhD have the highest mean.
- Standard Deviation: The variability in the number of days since the last purchase also varies across education levels. Customers with a Basic education level have the lowest standard deviation, indicating less variability in purchasing behavior compared to other education levels.
- Minimum: The minimum values show the shortest time interval between purchases for each education level. Customers across all education levels have made purchases on the same day as their last purchase, indicating instances of frequent purchasing behavior.
- 25th, 50th (median), and 75th Percentiles: These percentiles provide insights into the distribution of the time interval between purchases for each education level. For example, customers with a Master's degree have a higher median number of days since the last purchase compared to customers with a Basic education level.
- Maximum: The maximum values show the longest time interval between purchases for each education level. Customers with a PhD have the highest maximum number of days since the last purchase, suggesting longer gaps between purchases compared to other education levels.

Comparing statistics across effectiveness of campaigns

```
avg_recency_before_campaign = df.groupby('Total_Campaign_Acceptance')
['Days_Since_Last_Purchase'].mean()
print(avg_recency_before_campaign)

Total_Campaign_Acceptance
0    49.342165
1    48.280864
```

```
2    46.710843
3    52.590909
4    41.545455
Name: Days_Since_Last_Purchase, dtype: float64
```

Insights

- The mean number of days since the last purchase varies across different campaigns. Campaign-4 has the highest mean number of days since the last purchase, while Campaign-5 has the lowest. This suggests that customers who accepted Campaign-4 tend to make purchases less frequently compared to customers who accepted other campaigns.

Recommendations

- Targeting Strategies: Understanding the recency of purchases for customers who accepted each campaign can help in refining targeting strategies. For example, if Campaign-5 has a lower mean number of days since the last purchase, it may indicate that this campaign effectively targets customers who are more likely to make repeat purchases in a shorter time frame.
- Campaign Optimization: Analyzing the recency of purchases in relation to campaign acceptance rates can provide insights into the effectiveness of marketing strategies. For instance, if Campaign-4 has a higher mean number of days since the last purchase but a lower acceptance rate compared to other campaigns, it may indicate that adjustments are needed to improve the targeting or messaging of this campaign.

Hypothesise Testing

```python
from scipy.stats import normaltest
from scipy.stats import chi2_contingency
from scipy.stats import chi2
import scipy.stats as stats
```

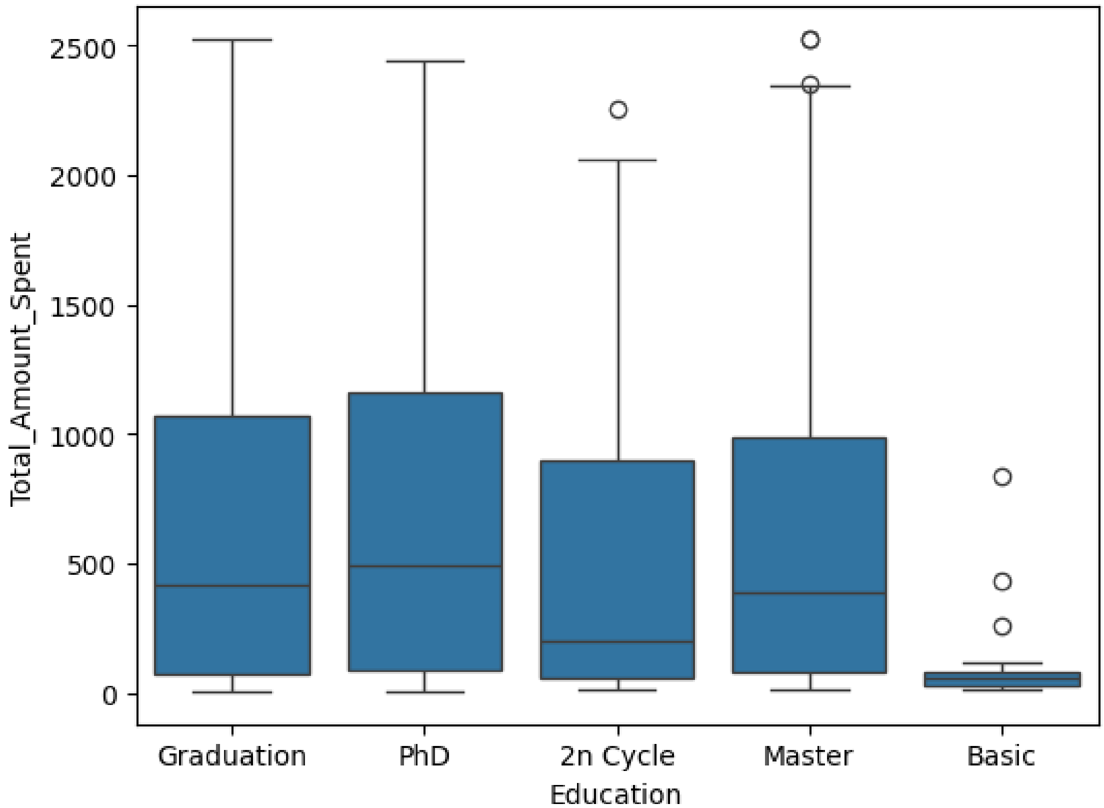
Is income of customers dependent on their education?

Performing One-way analysis of variance (ANOVA) test

- Null Hypothesis (Ho): The mean income of customers is the same across all levels of education.
- Alternative Hypothesis (Ha): The mean income of customers differs across at least one level of education.

```python
sns.boxplot(x='Education', y='Total_Amount_Spent', data=df)
```

```
<Axes: xlabel='Education', ylabel='Total_Amount_Spent'>
```




```
Graduation = df[df["Education"]=="Graduation"]['Total_Amount_Spent']
Phd = df[df["Education"]=="PhD"]['Total_Amount_Spent']
cycle_2n = df[df["Education"]=="2n Cycle"]['Total_Amount_Spent']
Basic = df[df["Education"]=="Basic"]['Total_Amount_Spent']
Master = df[df["Education"]=="Master"]['Total_Amount_Spent']

df.groupby('Education')['Total_Amount_Spent'].describe()
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 5,\n \"fields\": [\n
{\n   \"column\": \"Education\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 5,\n
\"samples\": [\n   \"Basic\",\n   \"PhD\",\n
\"Graduation\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 413.0595598700023,\n   \"min\": 54.0,\n   \"max\":
1126.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n
54.0,\n   486.0,\n   1126.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"mean\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 240.60486195332513,\n
\"min\": 81.79629629629629,\n   \"max\": 672.4094650205761,\n

```
\"num_unique_values\": 5,\n        \"samples\": [\n
81.79629629629629,\n          672.4094650205761,\n
620.3943161634103\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 213.5235064069152,\n        \"min\": 123.22726046566488,\n
\"max\": 623.3931567040069,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          123.22726046566488,\n
616.1191301267693,\n          599.561425579749\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3.6742346141747673,\n
\"min\": 5.0,\n        \"max\": 14.0,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          14.0,\n          8.0,\n
5.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 22.885175332516024,\n        \"min\": 29.75,\n
\"max\": 88.5,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          29.75,\n          88.5,\n          70.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 178.16319765877577,\
n        \"min\": 57.0,\n        \"max\": 493.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          57.0,\n
493.0,\n          414.5\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"75%\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 435.3386612741855,\n        \"min\":
80.0,\n        \"max\": 1157.5,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          80.0,\n          1157.5,\n          1073.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 722.7250514545625,\n
\"min\": 839.0,\n        \"max\": 2525.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          839.0,\n
2440.0,\n          2524.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```



```
f_stats, p_value = stats.f_oneway(Graduation, Phd, Basic, Master,
cycle_2n)

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value> 0.05:
      print("There is not enough evidence to conclude that income of
customers depends on their education.")
else:
```

```
    print("There is significant evidence to conclude that income of customers depends on their education.")
```

```
test statistic: 13.861692515208729
p_value: 3.587879728604343e-11
There is significant evidence to conclude that income of customers depends on their education.
```

Insights

- There is a statistically significant association between the education level of customers and their income.

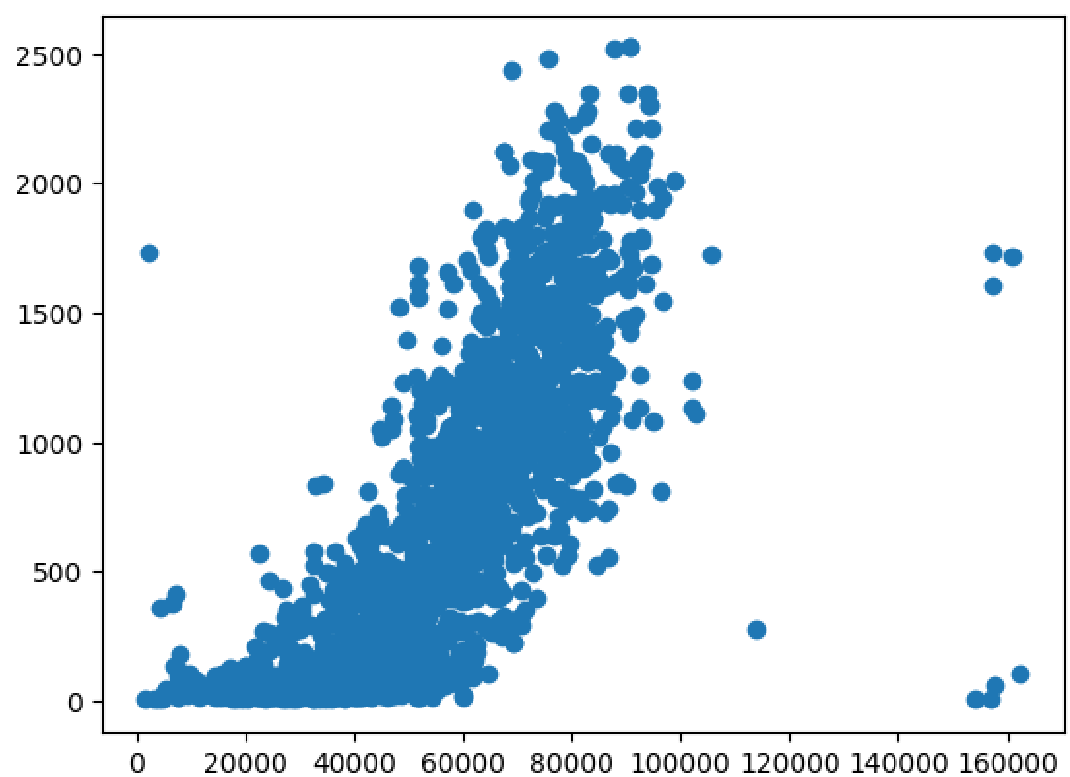
Do higher income people spend more?

Performing Correlation Test (Pearson's Correlation Coefficient / Spearman's Rank Correlation)

- Null Hypothesis (Ho): Income and Amount spend are Independent of each other.
- Alternative Hypothesis (Ha): There is a dependency of Income and Amount spend .

```
plt.scatter(df['Income'], df['Total_Amount_Spent'])
```

```
<matplotlib.collections.PathCollection at 0x7f20227e0eb0>
```



```python
df[['Income','Total_Amount_Spent']].describe()
```


```
{"summary":"{\n  \"name\": \"df[['Income','Total_Amount_Spent']]\",\n
\"rows\": 8,\n  \"fields\": [\n    {\n      \"column\": \"Income\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
51559.85414840027,\n        \"min\": 1730.0,\n        \"max\":
162397.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
51969.86139954853,\n          51717.0,\n          2239.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Total_Amount_Spent\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
953.5741677600665,\n        \"min\": 5.0,\n        \"max\": 2525.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
606.0410897722197,\n          396.0,\n          2239.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```python
df=df.dropna()
```

```python
if df["Income"].skew() > 2 or df['Total_Amount_Spent'].skew() > 2:
    corr, p = stats.spearmanr(df['Income'], df["Spending"])
    print("Spearman's correlation:", corr)
    print("p-value:", p)
    if p < 0.05:
        print("Reject H_0: There is a monotonic association between
income and spending.")
    else:
        print("Fail to reject H_0: No monotonic association found.")
else:
    corr, p = stats.pearsonr(df['Income'], df['Total_Amount_Spent'])
    print("Pearson's correlation:", corr)
    print("p-value:", p)
    if p < 0.05:
        print("Reject H_0: There is a linear association between
income and spending.")
    else:
        print("Fail to reject H_0: No linear association found.")
```

```
Pearson's correlation: 0.7893200552447133
p-value: 0.0
Reject H_0: There is a linear association between income and spending.
```

Insights

- The Pearson correlation coefficient between income and spending is approximately 0.789, which indicates a strong positive linear relationship between income and spending.

- The p-value is 0.0, which is significantly less than the significance level of 0.05. Therefore, we reject the null hypothesis (H0) and conclude that there is a statistically significant linear association between income and spending.

Recommendations

- Higher income people tend to spend more. This finding aligns with the intuitive expectation that individuals with higher incomes have greater purchasing power and are likely to spend more on goods and services.

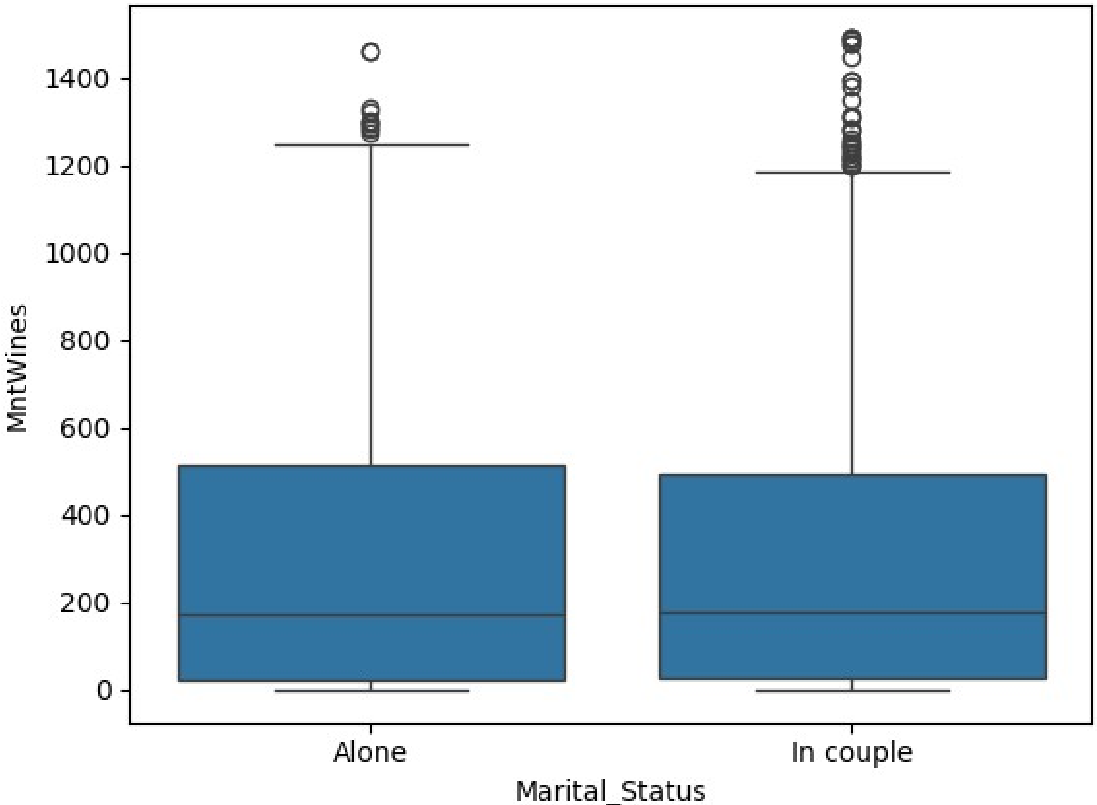
Do couples spend more or less money on wine that people living alone?

Performing Independent two-sample T-test

- Null Hypothesis(Ho): The means of the samples of In couples spending money on wine and Alone people spending on wine are equal.
- Alternnate Hypothesis(Ha): The means of the samples of In couples spending money on wine and Alone people spending on wine are not equal.

```
sns.boxplot(x='Marital_Status', y='MntWines', data=df)
```

```
<Axes: xlabel='Marital_Status', ylabel='MntWines'>
```




```
df.groupby('Marital_Status')['MntWines'].describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 2,\n \"fields\": [\n   {\n   \"column\": \"Marital_Status\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"In couple\",\n   \"Alone\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 458.2051942088828,\n   \"min\": 794.0,\n   \"max\": 1442.0,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1442.0,\n   794.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"mean\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3.5984180078080725,\n   \"min\": 302.3203883495146,\n   \"max\": 307.40931989924434,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   302.3203883495146,\n   307.40931989924434\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"std\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.4160434026117199,\n   \"min\": 336.4493479881944,\n   \"max\": 337.03772221070375,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   336.4493479881944,\n   337.03772221070375\n   ],\n

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.0,\n        \"min\": 0.0,\n
\"max\": 0.0,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.7071067811865476,\n        \"min\": 23.0,\n        \"max\":
24.0,\n        \"num_unique_values\": 2,\n        \"samples\": [\n
24.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"50%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2.1213203435596424,\n        \"min\": 172.5,\n
\"max\": 175.5,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          175.5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 16.263455967290593,\n
\"min\": 491.75,\n        \"max\": 514.75,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          491.75\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 21.920310216782973,\
n        \"min\": 1462.0,\n        \"max\": 1493.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1493.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```python
Alone = df[df['Marital_Status']=="Alone"]["MntWines"].sample(700)

Couple = df[df['Marital_Status']=="In couple"]["MntWines"].sample(700)

t_stat, p_value = stats.ttest_ind(Alone, Couple, 
alternative='greater')
print("p_value:",p_value)

if p_value> 0.05:
      print("There is not enough evidence to conclude that couples 
spend more or less money on wine than people living alone.")
else:
      print("There is significant evidence to conclude that couples 
spend more or less money on wine than people living alone.")
```

```
p_value: 0.1450667527328664
There is not enough evidence to conclude that couples spend more or 
less money on wine than people living alone.
```

Insights

- Given the obtained p-value of approximately 0.388, which is greater than the significance level of 0.05, we fail to reject the null hypothesis. Therefore, there is not enough evidence to conclude that there is a significant difference in the mean amounts spent on wine between individuals living alone and couples.

Are people with lower income are more attracted towards campaign or simply put accept more campaigns?

Performing Chi-Squared Test

- Null Hypothesis(Ho):There is no association between income bracket and campaign acceptance.
- Alternate Hypothesis(Ha):There is a significant association between income bracket and campaign acceptance.

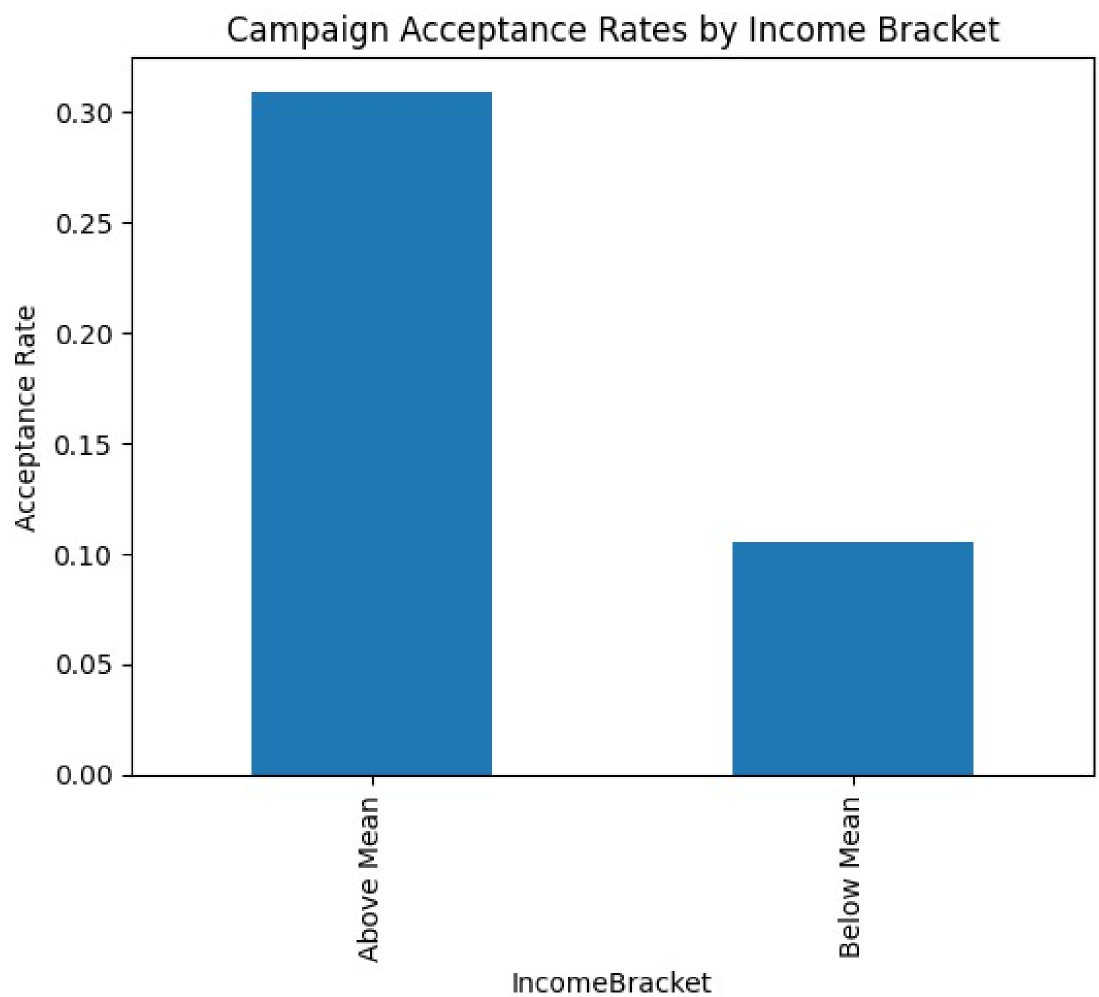
```
mean_income = df['Income'].mean()

df['IncomeBracket'] = 'Above Mean'
df.loc[df['Income'] < mean_income, 'IncomeBracket'] = 'Below Mean'

df['EverAcceptedCampaign'] = (df[['AcceptedCmp1', 'AcceptedCmp2',
'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']] == 1).any(axis=1)


df[['Income', 'IncomeBracket', 'EverAcceptedCampaign']].head(5)
```


```
{"summary":"{\n  \"name\": \"df[['Income', 'IncomeBracket',
'EverAcceptedCampaign']]\",\n  \"rows\": 5,\n  \"fields\": [\n    {\n
\"column\": \"Income\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 25730.647926159963,\n        \"min\":
21474.0,\n        \"max\": 84835.0,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          57091.0,\n          21474.0,\n
67267.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"IncomeBracket\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Below Mean\",\n          \"Above Mean\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"EverAcceptedCampaign\",\n
\"properties\": {\n        \"dtype\": \"boolean\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          true,\n
false\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
contingency_table = pd.crosstab(df['IncomeBracket'],
df['EverAcceptedCampaign'])
print(contingency_table)
```

```
EverAcceptedCampaign  False  True
IncomeBracket
```

```
Above Mean                  769    344
Below Mean                 1005    118

chi2, p, dof, expected = chi2_contingency(contingency_table)

print('Chi-square test statistic:', chi2)
print('p-value:', p)

if p> 0.05:
      print("There is not enough evidence to conclude that there is an
association between income bracket and campaign acceptance.")
else:
      print("There is a significant association between income bracket
and campaign acceptance.")


Chi-square test statistic: 140.66622705956848
p-value: 1.903389496887248e-32
There is a significant association between income bracket and campaign
acceptance.

acceptance_rates = df.groupby('IncomeBracket')
['EverAcceptedCampaign'].mean()

acceptance_rates.plot(kind='bar')
plt.title('Campaign Acceptance Rates by Income Bracket')
plt.ylabel('Acceptance Rate')
plt.show()
```



Insights

- The interpretation of this result is that there is strong evidence to suggest that income bracket and campaign acceptance are dependent of each other. There is a relationship between a customer's income bracket and their likelihood of accepting a campaign offer.

Recommendations

- It suggests that income level may influence how customers respond to campaign offers, and marketers may need to tailor their campaigns differently for customers in different income brackets to optimize their effectiveness.