EDA_v2

May 18, 2021

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

[2]:
```
data = pd.read_csv("data/online_shoppers_intention.csv")
```

Ramka składa się z następujących kolumn:

[3]:
```
data.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
```

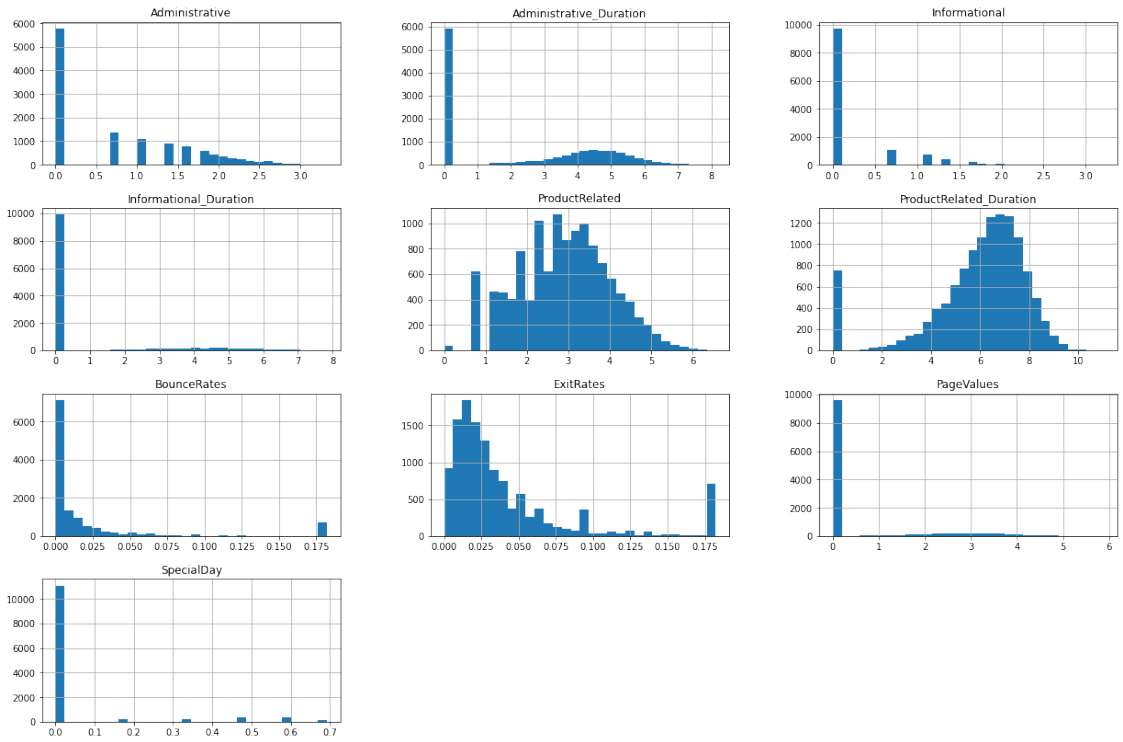
Zmienne: “Administrative”, “Administrative Duration”, “Informational”, “Informational Duration”, “Product Related”, “Product Related Duration”, “Bounce Rate”, “Exit Rate”, “Page Value”

oraz “Special Day” to zmienne numeryczne.

Natomiast “OperatingSystems”, “Browser”, “Region”, “TrafficType”, “VisitorType”, “Weekend” oraz “Revenue” to zmienne kategoryczne.

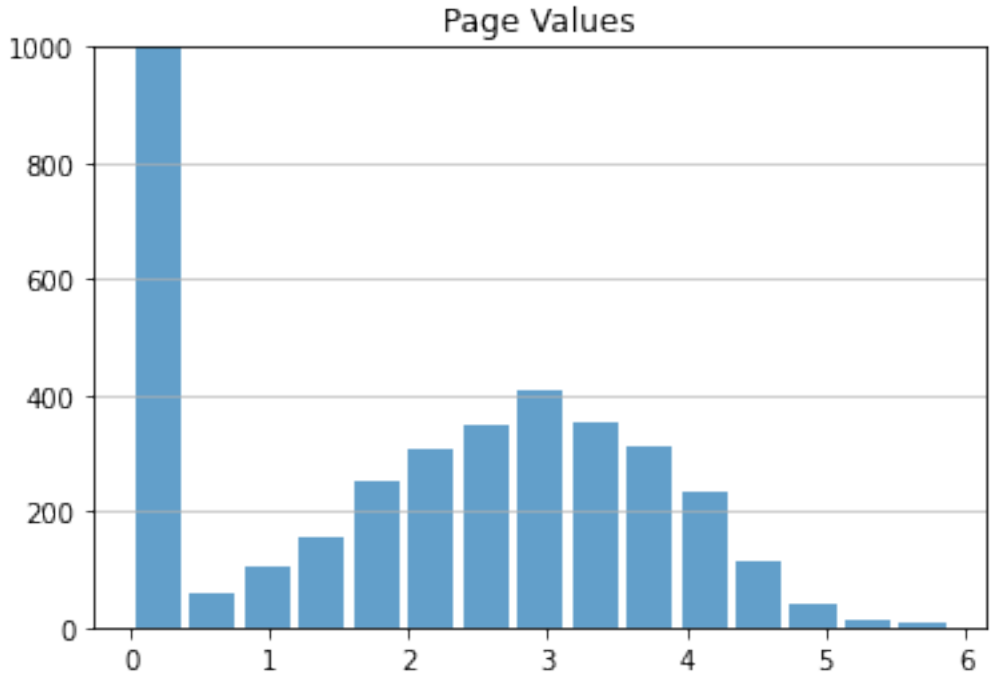
[4]:
```
num_vars = ["Administrative", "Administrative_Duration", "Informational",␣
 ↪"Informational_Duration", "ProductRelated", "ProductRelated_Duration",␣
 ↪"BounceRates", "ExitRates", "PageValues", "SpecialDay"]
cat_vars = ["OperatingSystems", "Browser", "Region", "TrafficType",␣
 ↪"VisitorType", "Weekend","Revenue"]
```

[5]:
```
from sklearn.preprocessing import FunctionTransformer


transformer = FunctionTransformer(np.log1p)

transformer.transform(data[num_vars]).hist(bins=30, figsize=(21,14))
plt.show()
```

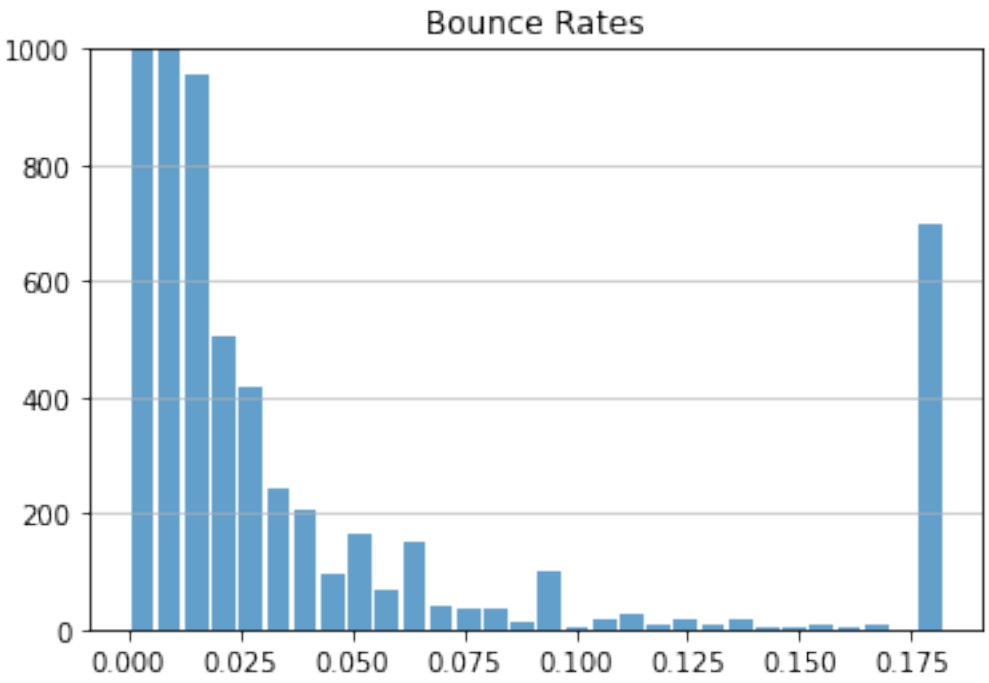


[25]:
```
n, bins, patches = plt.hist(x=transformer.
 ↪transform(data[num_vars])["PageValues"], bins='auto',
                             alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title("Page Values")
```

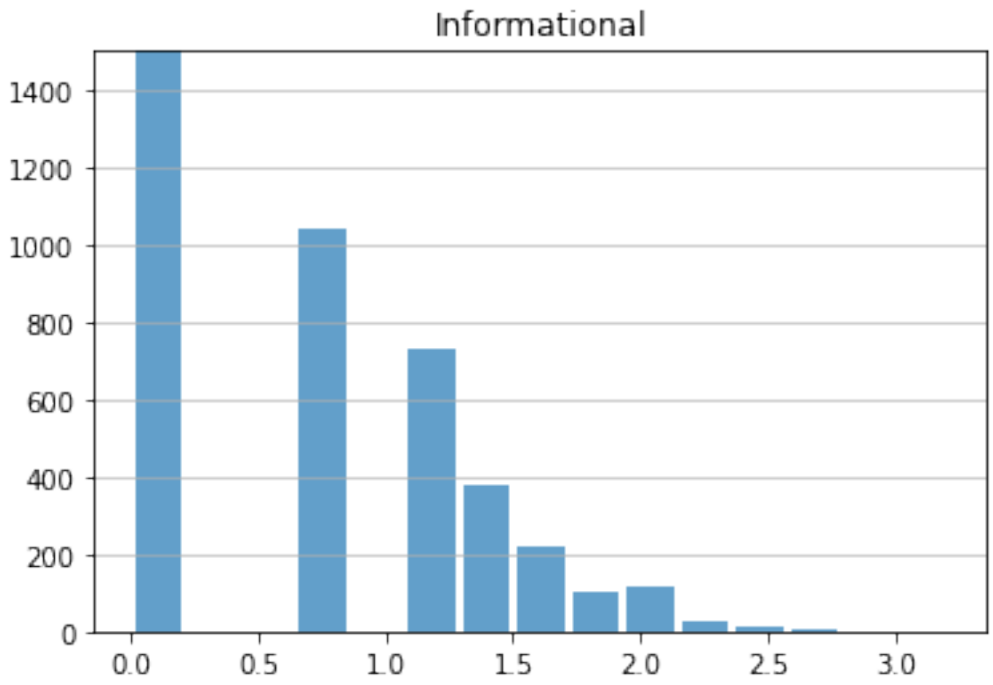
```
plt.ylim(top=1000)
plt.show()
```



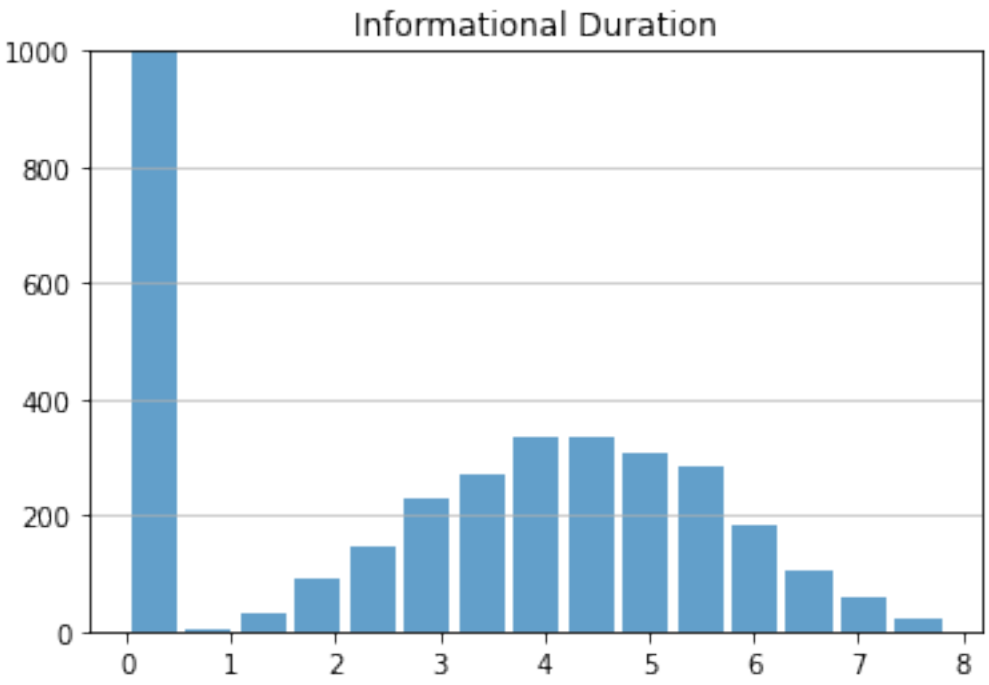
[28]:
```
n, bins, patches = plt.hist(x=transformer.
 ↪transform(data[num_vars])["BounceRates"], bins=30,
                             alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title("Bounce Rates")
plt.ylim(top=1000)
plt.show()
```



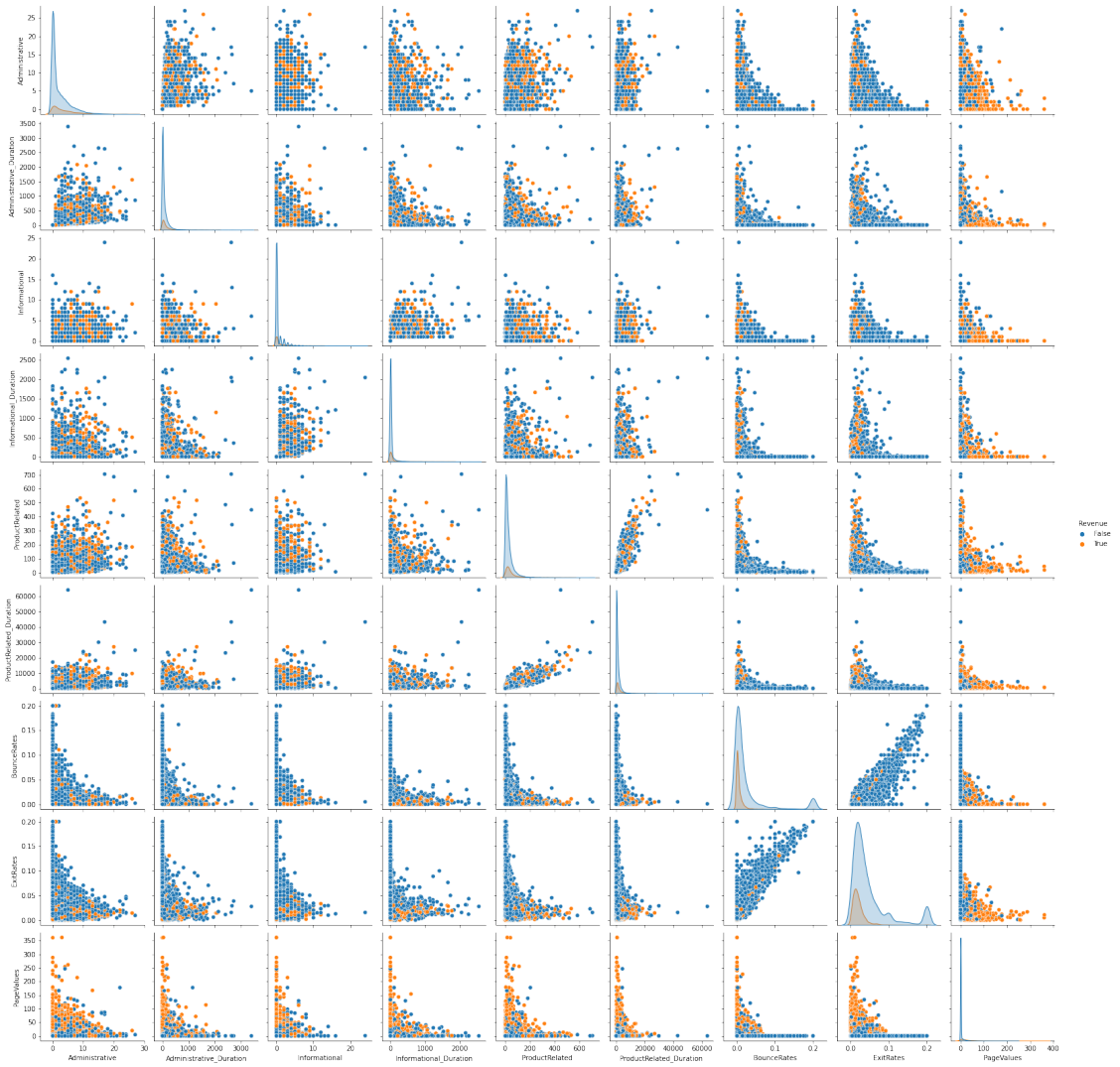
[30]:
```
n, bins, patches = plt.hist(x=transformer.
 ↪transform(data[num_vars])["Informational"], bins='auto',
                             alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title("Informational")
plt.ylim(top=1500)
plt.show()
```



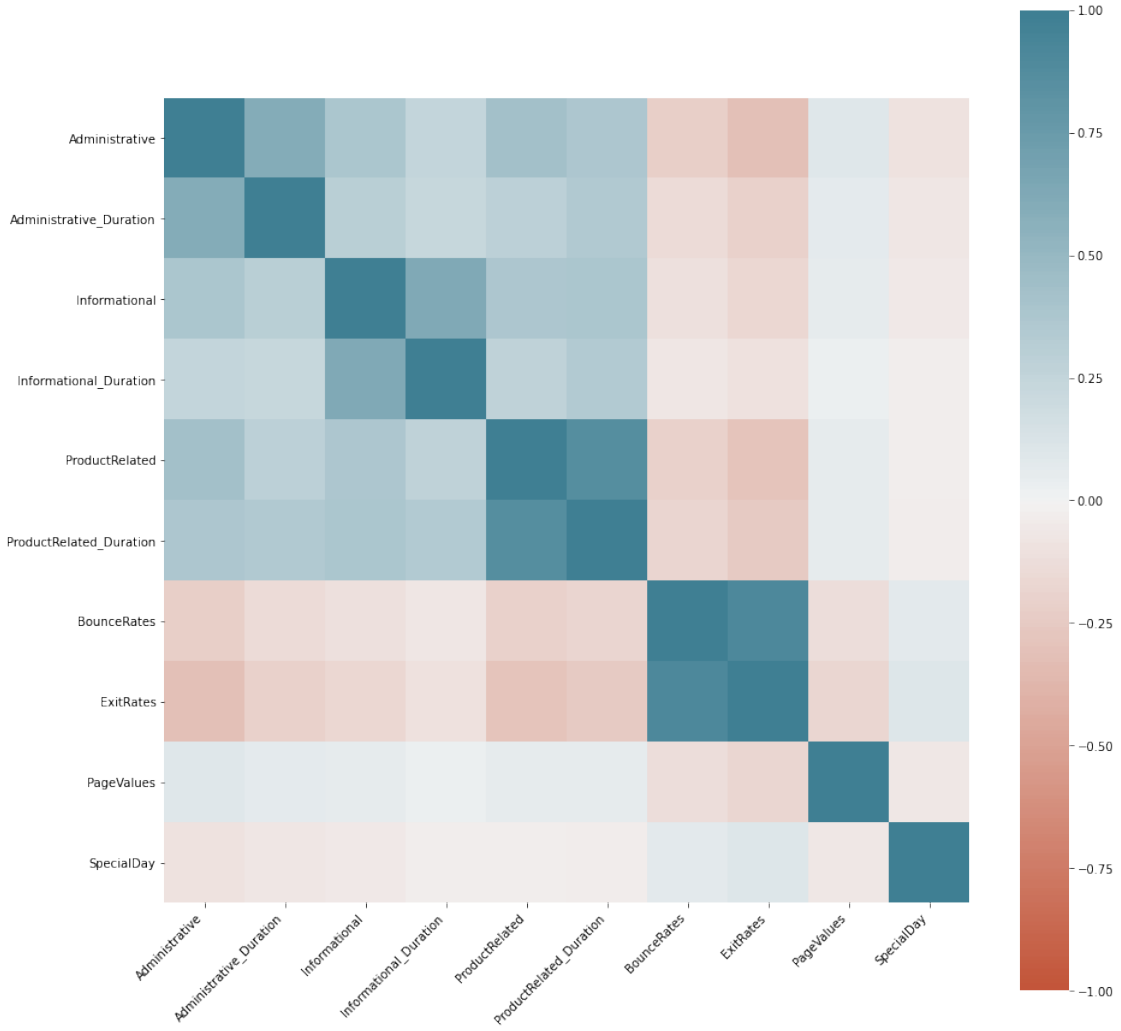
[31]:
```
n, bins, patches = plt.hist(x=transformer.
 ↪transform(data[num_vars])["Informational_Duration"], bins='auto',
                             alpha=0.7, rwidth=0.85)
plt.grid(axis='y', alpha=0.75)
plt.title("Informational Duration")
plt.ylim(top=1000)
plt.show()
```



[6]:
```
num_vars_with_rev = num_vars + ["Revenue"]

sns.pairplot(data[num_vars_with_rev].drop("SpecialDay", axis=1), hue="Revenue")
plt.show()
```

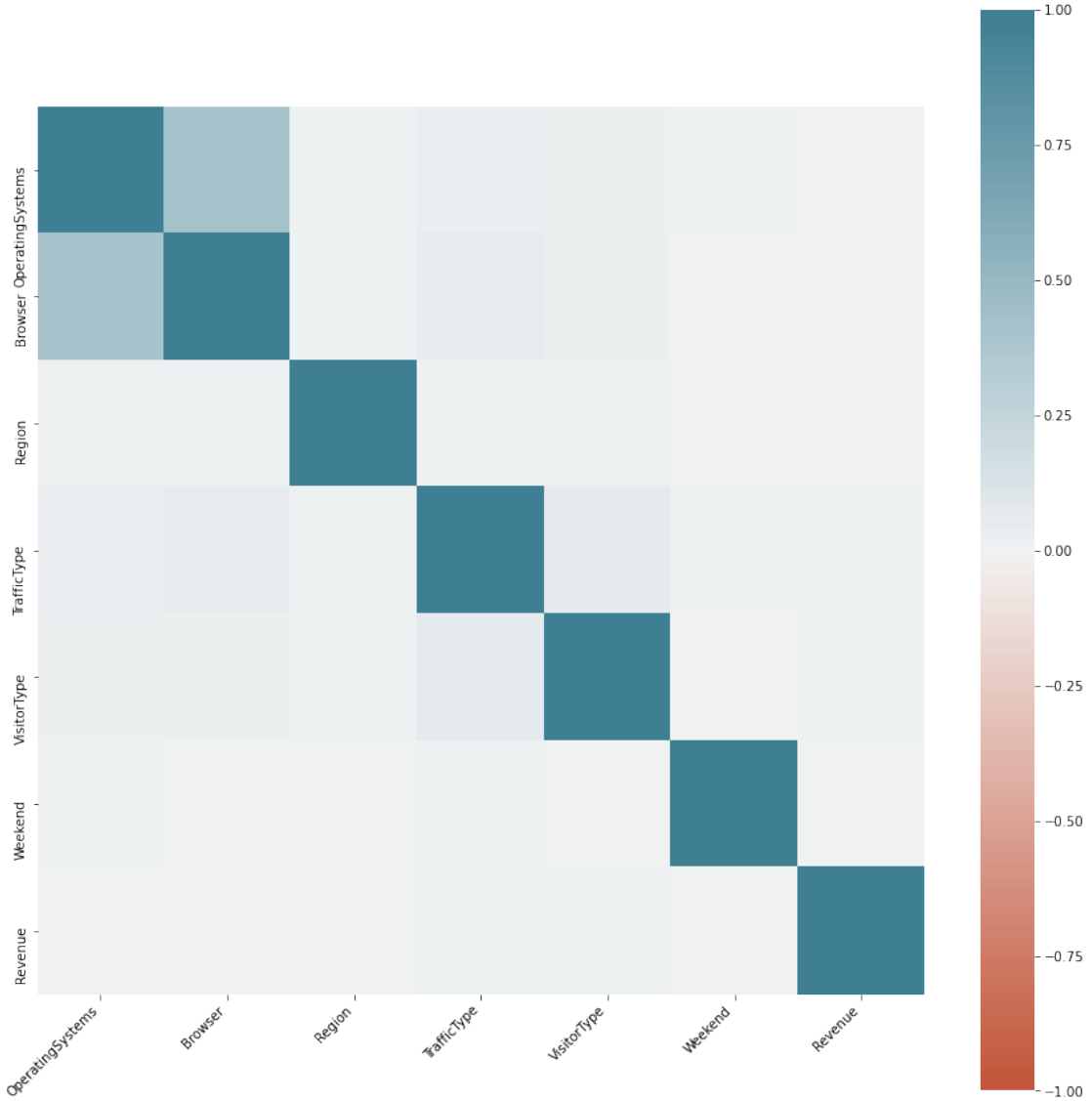


[7]:
```
corr = data[num_vars].corr()
f, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
)
plt.show()
```

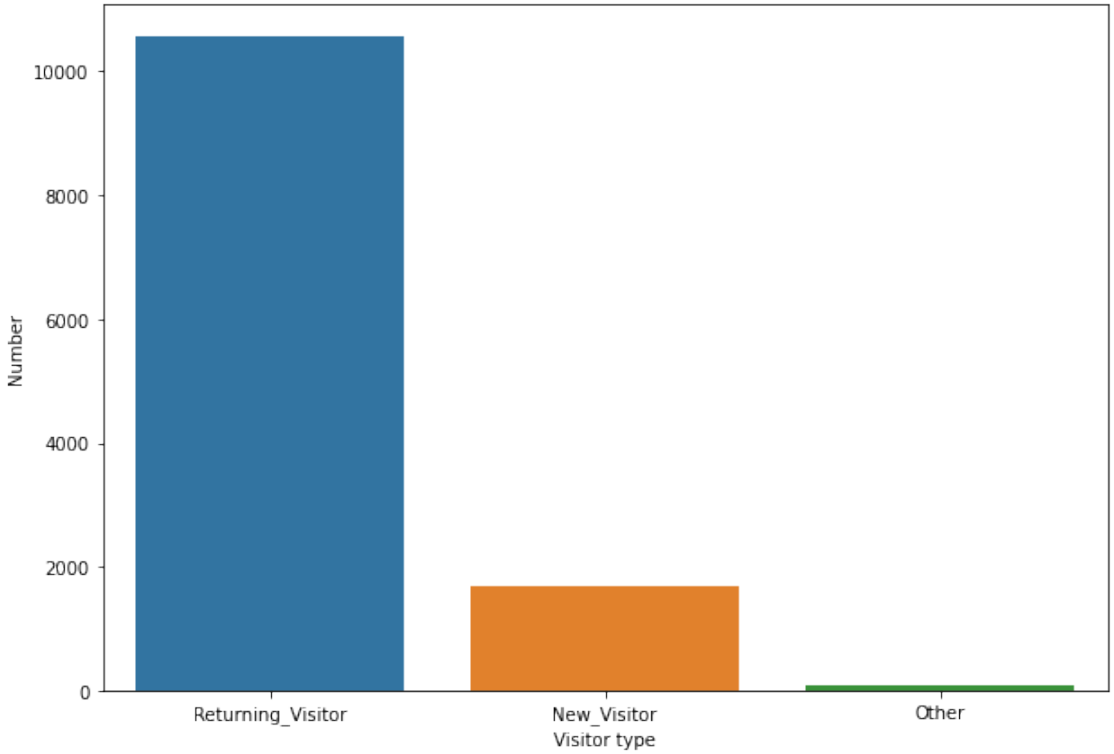


[8]:
```
from sklearn.metrics import normalized_mutual_info_score
mi = []
cat_data = data[cat_vars]


for i in range(len(cat_vars)):
    temp = []
    for j in range(len(cat_vars)):
        temp.append(normalized_mutual_info_score(cat_data.iloc[:,i], cat_data.
 ↪iloc[:,j]))
    mi.append(temp)

midf = pd.DataFrame(mi, columns=cat_vars, index=cat_vars)

f, ax = plt.subplots(figsize=(15, 15))
```

```
ax = sns.heatmap(
    midf,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
)
plt.show()
```

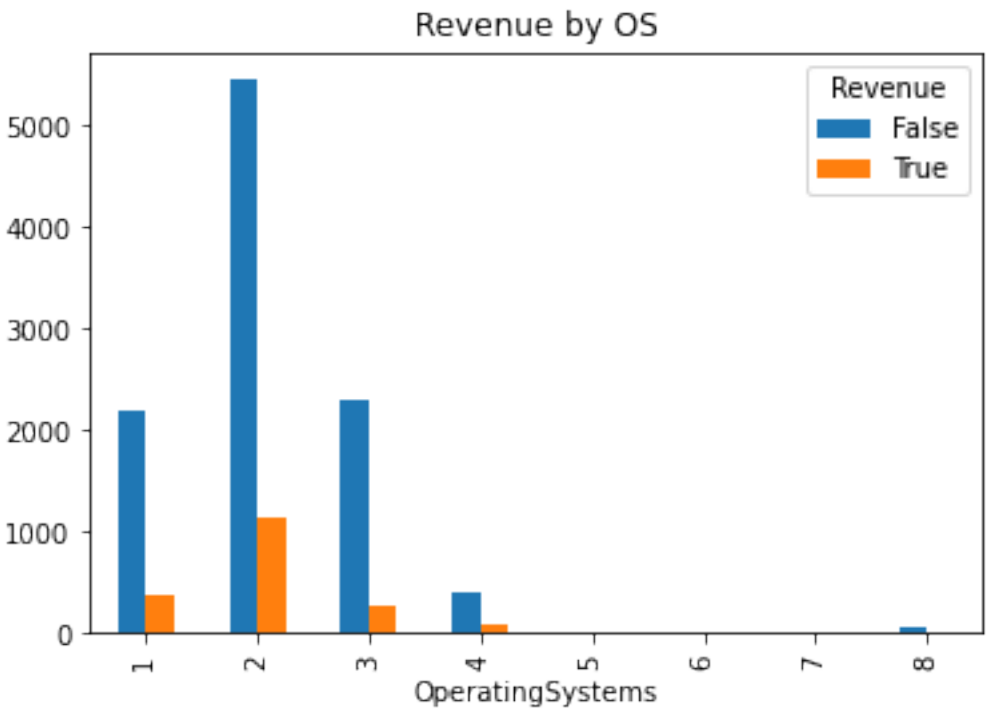


[9]:
```
plt.figure(figsize = (10, 7))
sns.countplot(data = data, x = "VisitorType")
plt.xlabel("Visitor type")
plt.ylabel("Number")
plt.show()
```



[10]:
```
plt.figure(figsize=(20,20))
data.groupby("OperatingSystems")["Revenue"].value_counts().unstack().
 ↪plot(kind="bar")
plt.title("Revenue by OS")
plt.show()

percentage = data.groupby("OperatingSystems")["Revenue"].value_counts().
 ↪unstack()
percentage["percentage"] = percentage[True] / (percentage[True] +␣
 ↪percentage[False] )
percentage
```

```
<Figure size 1440x1440 with 0 Axes>
```

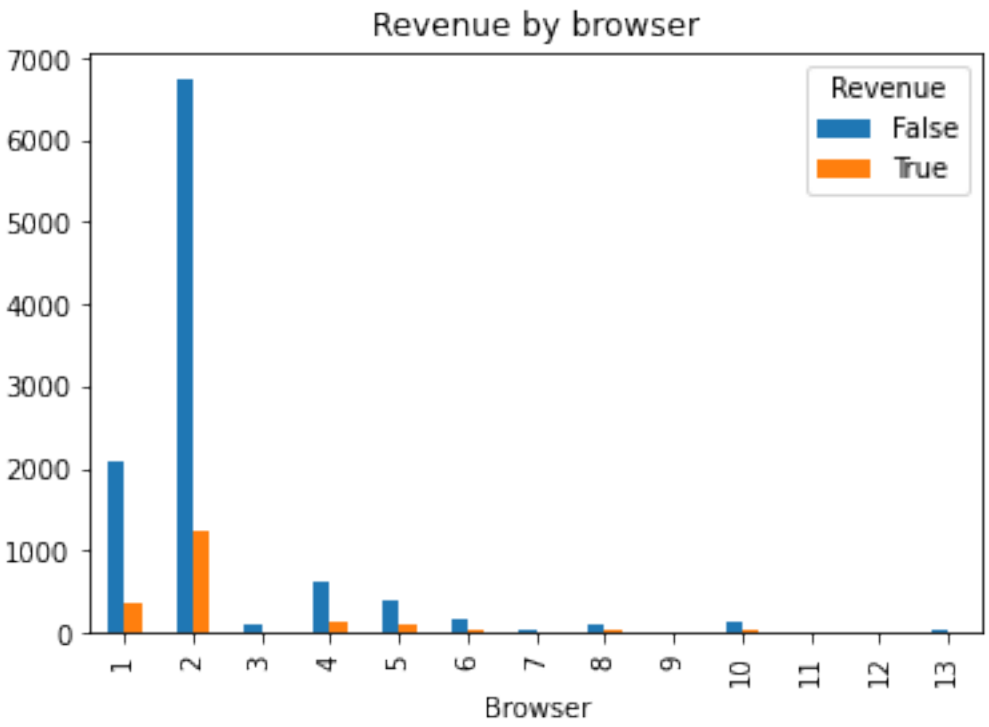


[10]:

Revenue	False	True	percentage
OperatingSystems			
1	2206	379	0.146615
2	5446	1155	0.174973
3	2287	268	0.104892
4	393	85	0.177824
5	5	1	0.166667
6	17	2	0.105263
7	6	1	0.142857
8	62	17	0.215190

[11]:
```
plt.figure(figsize = (10,10))
data.groupby("Browser")["Revenue"].value_counts().unstack().plot(kind="bar")
plt.title("Revenue by browser")
plt.show()

percentage = data.groupby("Browser")["Revenue"].value_counts().unstack()

percentage["percentage"] = percentage[True] / (percentage[True] +␣
 ↪percentage[False] )
percentage
```

```
<Figure size 720x720 with 0 Axes>
```

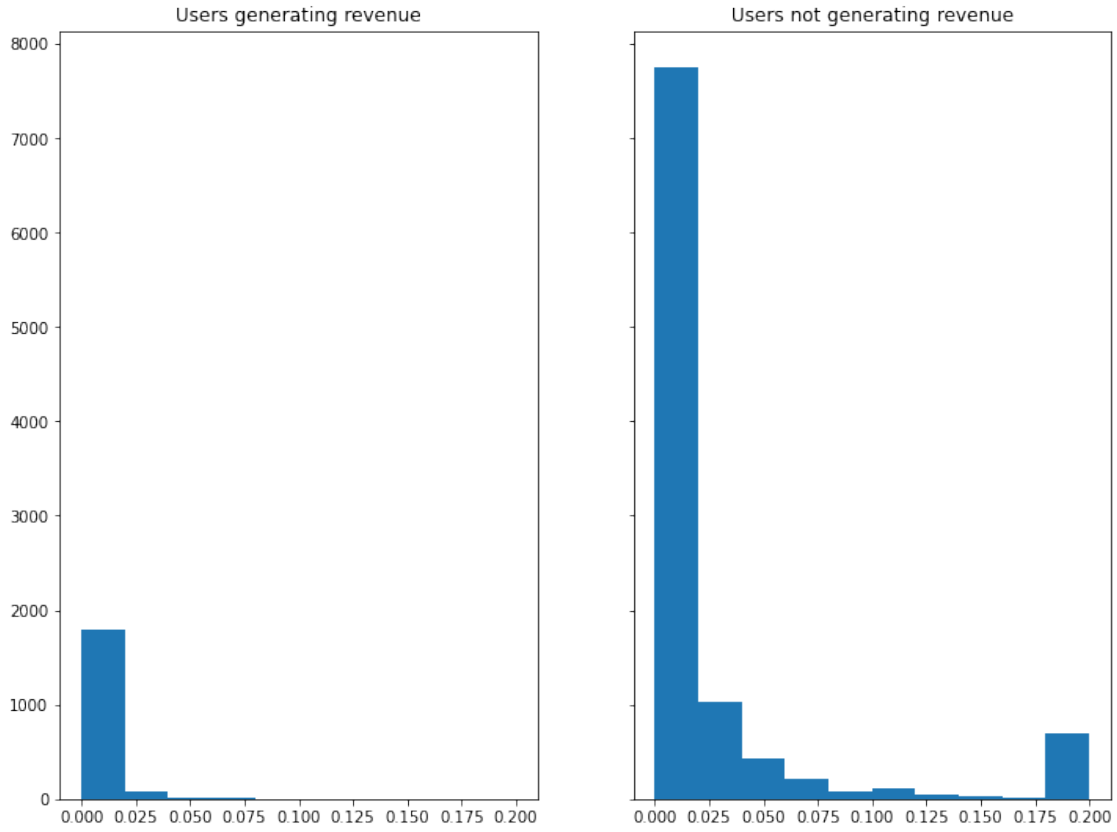


[11]:

Revenue	False	True	percentage
Browser			
1	2097.0	365.0	0.148253
2	6738.0	1223.0	0.153624
3	100.0	5.0	0.047619
4	606.0	130.0	0.176630
5	381.0	86.0	0.184154
6	154.0	20.0	0.114943
7	43.0	6.0	0.122449
8	114.0	21.0	0.155556
9	1.0	NaN	NaN
10	131.0	32.0	0.196319
11	5.0	1.0	0.166667
12	7.0	3.0	0.300000
13	45.0	16.0	0.262295

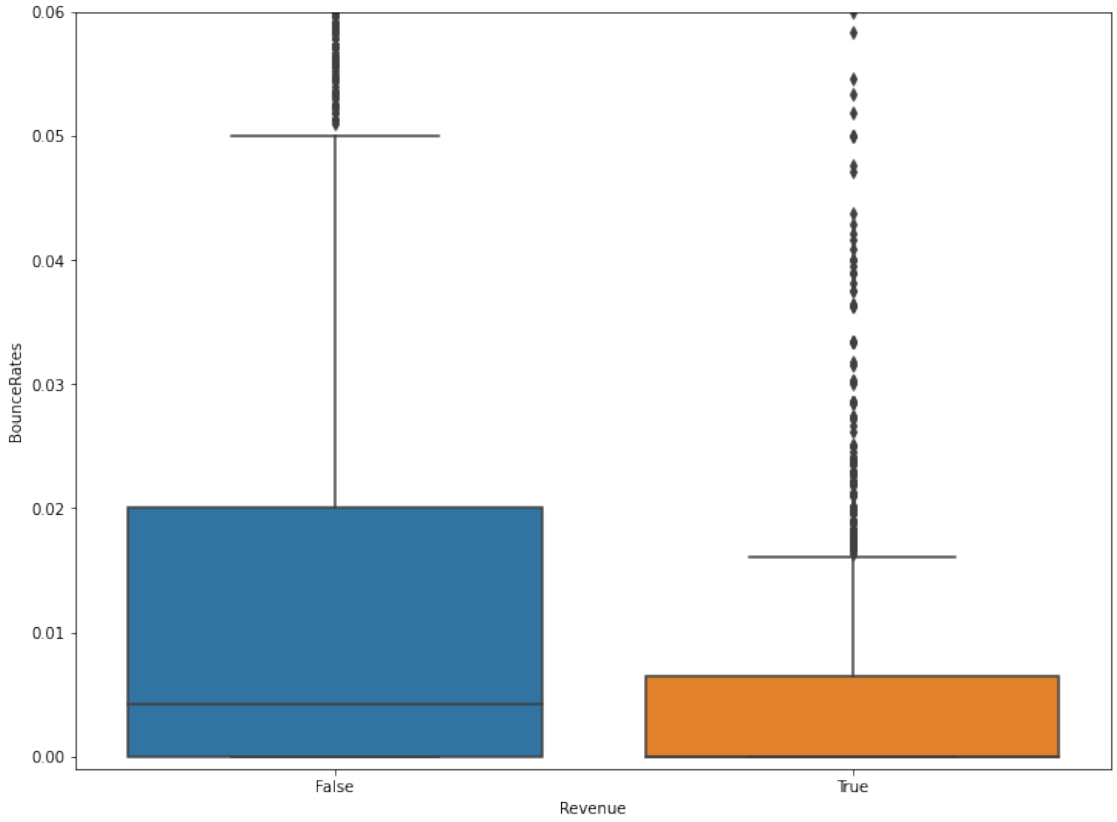
[12]:
```
istrue = data["Revenue"]== True

fig, (ax1, ax2) = plt.subplots(1,2, sharey=True, figsize=(12,9))

ax1.hist(data.loc[istrue, "BounceRates"])
ax1.set_title("Users generating revenue")
ax2.hist(data.loc[istrue==False, "BounceRates"])
```

```
ax2.set_title("Users not generating revenue")
plt.show()
```

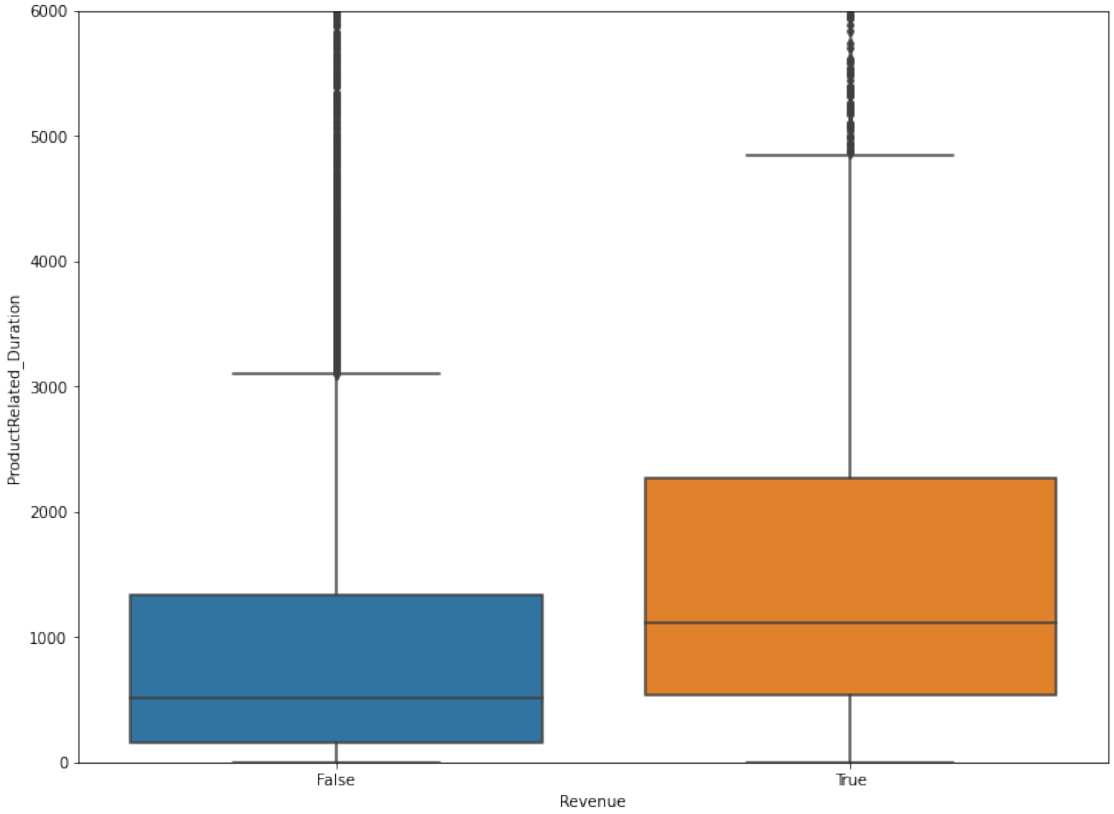


[13]:
```
plt.figure(figsize=(12,9))
sns.boxplot(x=data["Revenue"], y=data["BounceRates"])
plt.ylim([-0.001,0.06])
```

[13]: (-0.001, 0.06)



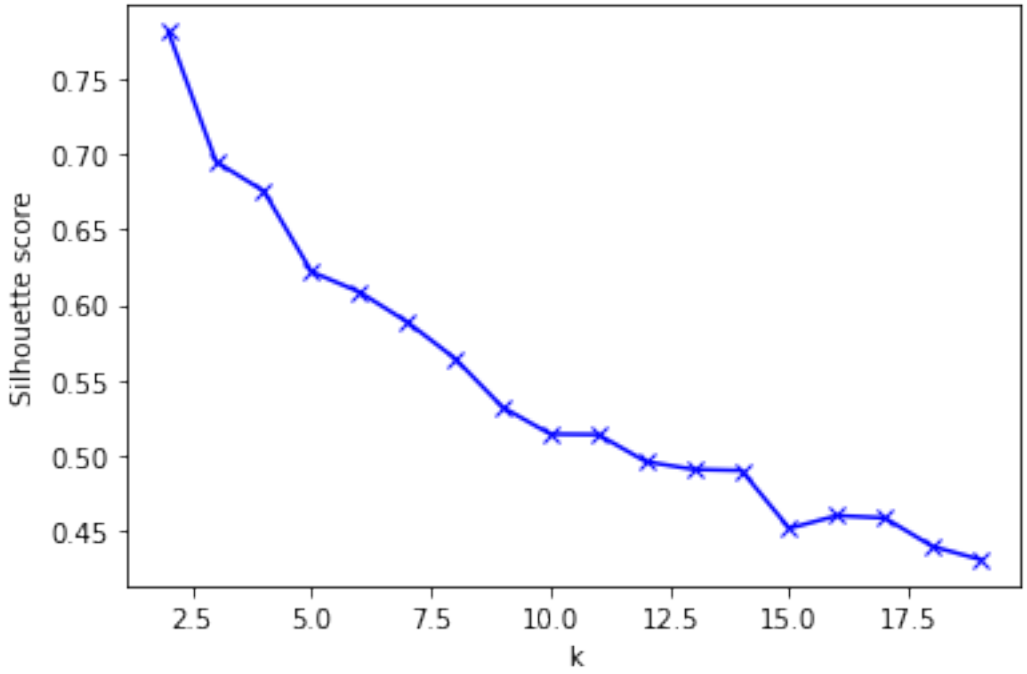
[14]:
```
plt.figure(figsize=(12,9))
sns.boxplot(x=data["Revenue"], y=data["ProductRelated_Duration"])
plt.ylim([0,6000])
plt.show()
```



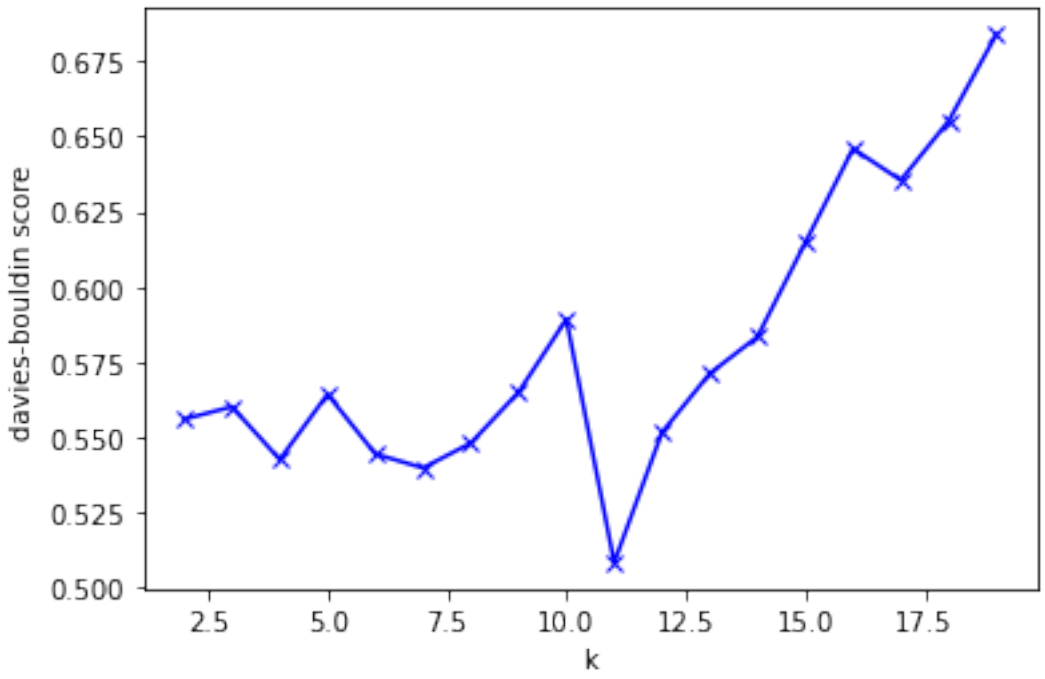
[15]:
```python
def count_clustering_scores(X, cluster_num, model, score_fun):
    if isinstance(cluster_num, int):
        cluster_num_iter = [cluster_num]
    else:
        cluster_num_iter = cluster_num

    scores = []
    for k in cluster_num_iter:
        model_instance = model(n_clusters=k)
        labels = model_instance.fit_predict(X)
        wcss = score_fun(X, labels)
        scores.append(wcss)

    if isinstance(cluster_num, int):
        return scores[0]
    else:
        return scores
```

[16]:
```python
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
```

```
cluster_num_seq = range(2, 20)
silhouette_vec = count_clustering_scores(data.drop(["Month", "VisitorType"],␣
 ↪axis=1), cluster_num_seq, KMeans, silhouette_score)
plt.plot(cluster_num_seq, silhouette_vec, 'bx-')
plt.xlabel('k')
plt.ylabel('Silhouette score')
plt.show()
```



[17]:
```
from sklearn.metrics import davies_bouldin_score
from sklearn.cluster import KMeans

cluster_num_seq = range(2, 20)
davies_vec = count_clustering_scores(data.drop(["Month", "VisitorType"],␣
 ↪axis=1), cluster_num_seq, KMeans, davies_bouldin_score)
plt.plot(cluster_num_seq, davies_vec, 'bx-')
plt.xlabel('k')
plt.ylabel('davies-bouldin score')
plt.show()
```



[18]:
```
from sklearn.manifold import TSNE


tsne = TSNE(perplexity = 60)

X_tsne = pd.DataFrame(tsne.fit_transform(data.drop(["Month", "VisitorType"],␣
↪axis=1)), columns=["tsne1", "tsne2"])

X_tsne.head()
```

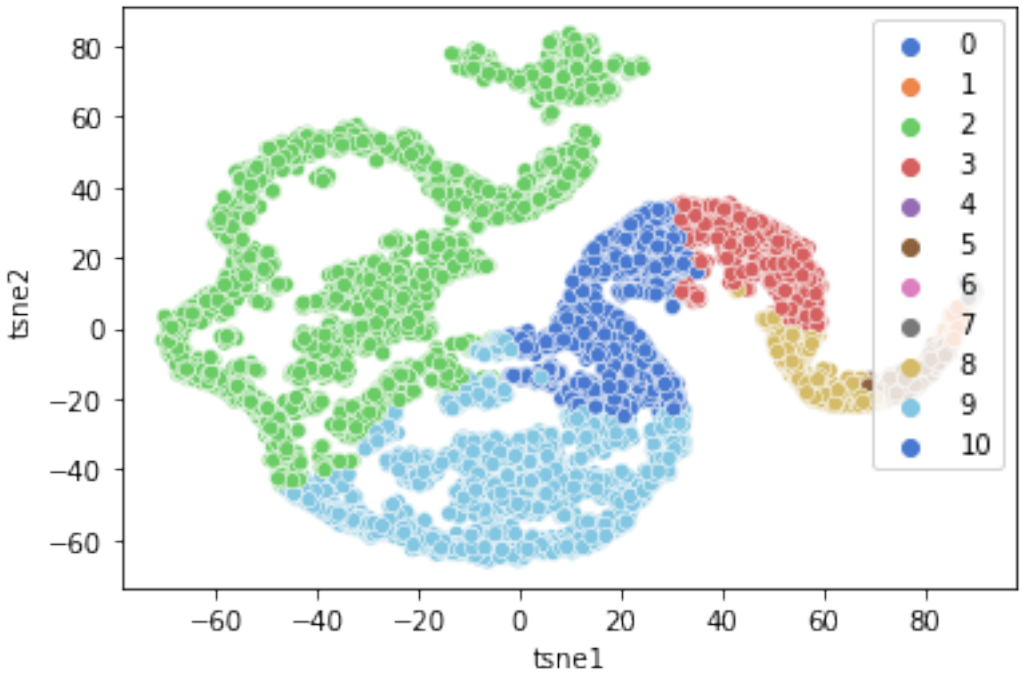
[18]:

	tsne1	tsne2
0	12.185030	65.665810
1	-12.385922	37.943546
2	9.483695	82.318314
3	11.561109	55.786449
4	-19.852266	-59.013126

[19]:
```
km = KMeans(n_clusters=11)

km.fit(data.drop(["Month", "VisitorType"], axis=1))

predictions = km.labels_
```

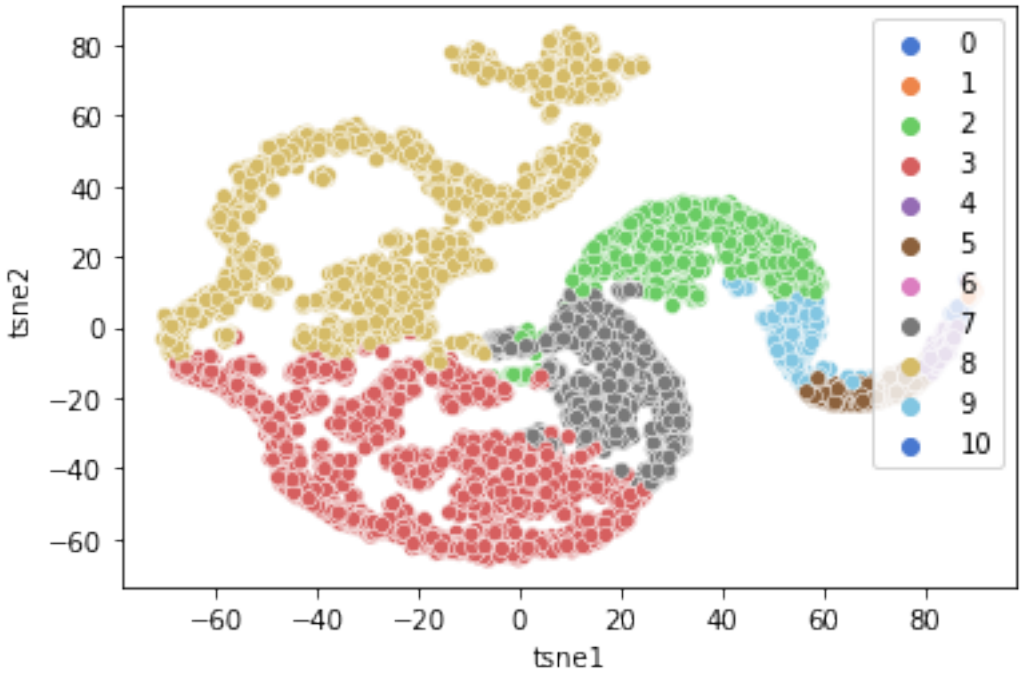
```
sns.scatterplot(x="tsne1", y="tsne2", data=X_tsne, hue=predictions, palette=sns.
↪color_palette("muted", n_colors=11))
```

[19]: <AxesSubplot:xlabel='tsne1', ylabel='tsne2'>



[20]:
```
from sklearn.cluster import AgglomerativeClustering

agg = AgglomerativeClustering(n_clusters=11)

agg.fit(data.drop(["Month", "VisitorType"], axis=1))


predictions = agg.labels_

sns.scatterplot(x="tsne1", y="tsne2", data=X_tsne, hue=predictions, palette=sns.
↪color_palette("muted", n_colors=11))
```

[20]: <AxesSubplot:xlabel='tsne1', ylabel='tsne2'>


tsne2
tsne1
80
60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60
80
0
1
2
3
4
5
6
7
8
9
10