In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv('../Machine Learning Project/online_shoppers_intention.csv')
```

In [3]:
```
df
```

Out[3]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductI
0	0	0.0	0	0.0	
1	0	0.0	0	0.0	
2	0	0.0	0	0.0	
3	0	0.0	0	0.0	
4	0	0.0	0	0.0	
...	...	...	...	...	
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	

12330 rows × 18 columns

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductI
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.(
mean	2.315166	80.818611	0.503569	34.472398	31.
std	3.321784	176.779107	1.270156	140.749294	44.(
min	0.000000	0.000000	0.000000	0.000000	0.(
25%	0.000000	0.000000	0.000000	0.000000	7.(
50%	1.000000	7.500000	0.000000	0.000000	18.(
75%	4.000000	93.256250	0.000000	0.000000	38.(
max	27.000000	3398.750000	24.000000	2549.375000	705.(

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
df['Revenue'] = df['Revenue'].astype(int) #clean data type: bool to int
```

In [8]:
```
df['Weekend'] = df['Weekend'].astype(int) #clean data type: bool to int
```

In [9]:
```
month = {'Feb':2, 'Mar':3,'May':5, 'June':6, 'Jul':7, 'Aug':8, 'Sep':9, 'Oct'
df['Month'] = df['Month'].map(month)
```

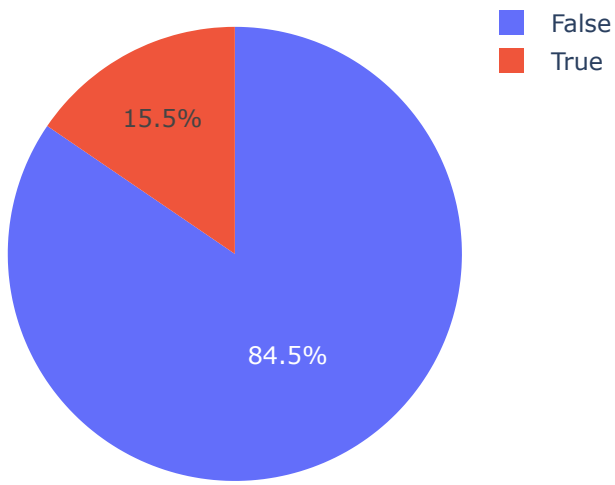
EDA

In [10]:
```
df_purchaser = df[df['Revenue']==1]
```

In [11]:
```
df_visitor = df[df['Revenue']==0]
```

1. Target Variable - revenue

In [12]:
```
#Target Variable
labels = ["False","True"]
values = df['Revenue'].value_counts().tolist()

fig1 = px.pie(df, values=values, names=labels)
fig1.update_layout(
    autosize=False,
    width=400,
    height=400)
```



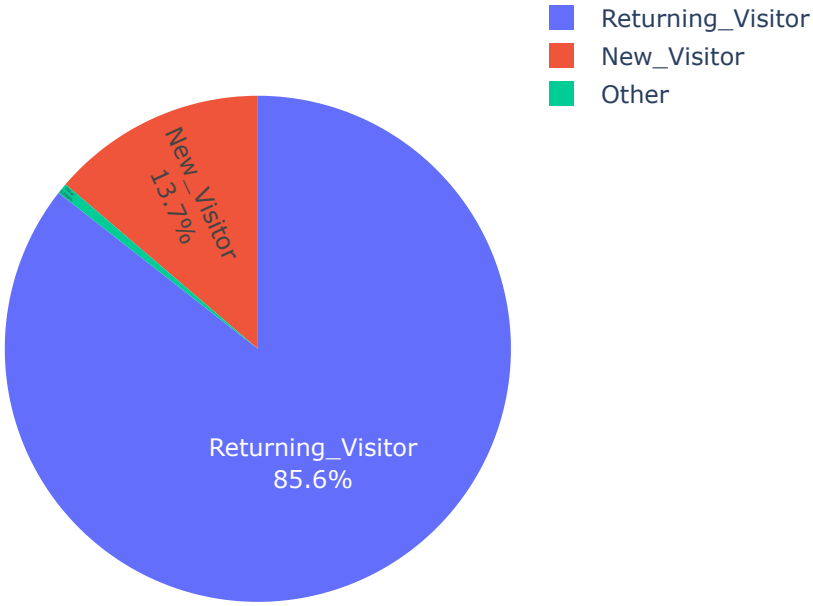
2.1 Feature Variable - visitor type

In [13]:
```
df['VisitorType'].value_counts()*100/df['VisitorType'].count() #Most are retu
```

Out[13]:
```
Returning_Visitor    85.571776
New_Visitor          13.738848
Other                 0.689376
Name: VisitorType, dtype: float64
```

In [14]:
```
#visitor type ratio
labels = ["Returning_Visitor","New_Visitor",'Other']
values = df['VisitorType'].value_counts().tolist()

fig1 = px.pie(df, values=values, names=labels)
fig1.update_layout(
```

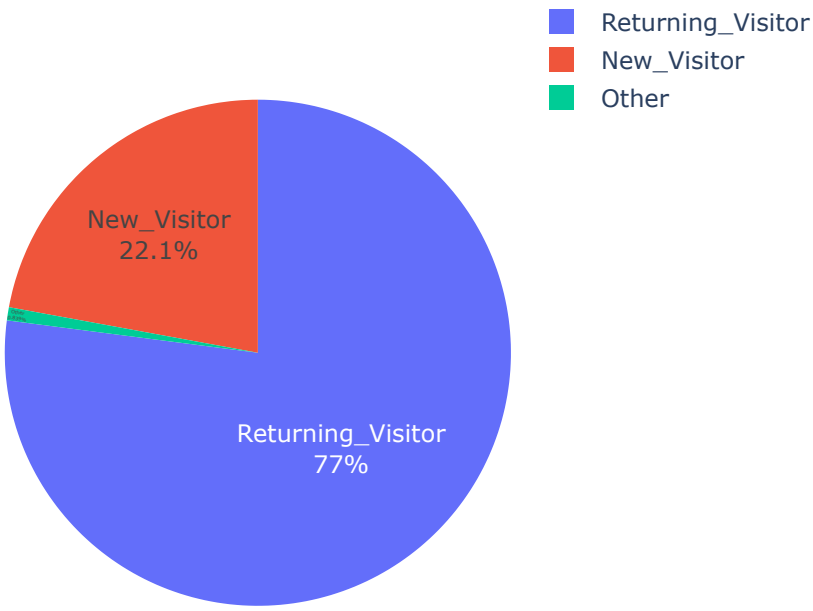
```
    autosize=False,
    width=500,
    height=500)
fig1.update_traces(textposition='inside', textinfo='percent+label')
```



In [15]:
```
df.groupby('VisitorType')['Revenue'].value_counts()
```

Out[15]:
```
VisitorType        Revenue
New_Visitor        0          1272
                   1           422
Other              0            69
                   1            16
Returning_Visitor  0          9081
                   1          1470
Name: Revenue, dtype: int64
```

In [16]:
```
#visitor type ratio - those who purchased only
labels = ["Returning_Visitor","New_Visitor",'Other']
values = df_purchaser['VisitorType'].value_counts().tolist()

fig1 = px.pie(df_purchaser, values=values, names=labels)
fig1.update_layout(
    autosize=False,
    width=500,
    height=500)
fig1.update_traces(textposition='inside', textinfo='percent+label')
```



In [17]:
```
df_purchaser.groupby('VisitorType')['Revenue'].value_counts()/df_purchaser['R
```

Out[17]:
```
VisitorType        Revenue
New_Visitor        1          0.221174
Other              1          0.008386
Returning_Visitor  1          0.770440
Name: Revenue, dtype: float64
```

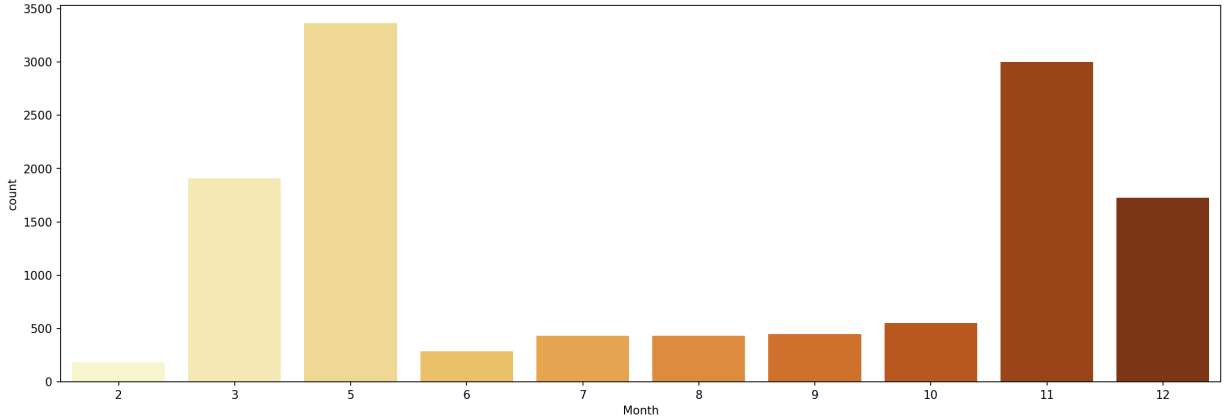
In [18]:
```
df_visitor.groupby('VisitorType')['Revenue'].value_counts()/df_visitor['Reven
```

Out[18]:
```
VisitorType        Revenue
New_Visitor        0          0.122050
Other              0          0.006621
Returning_Visitor  0          0.871330
Name: Revenue, dtype: float64
```

2.2 Feature Variable - seasonality

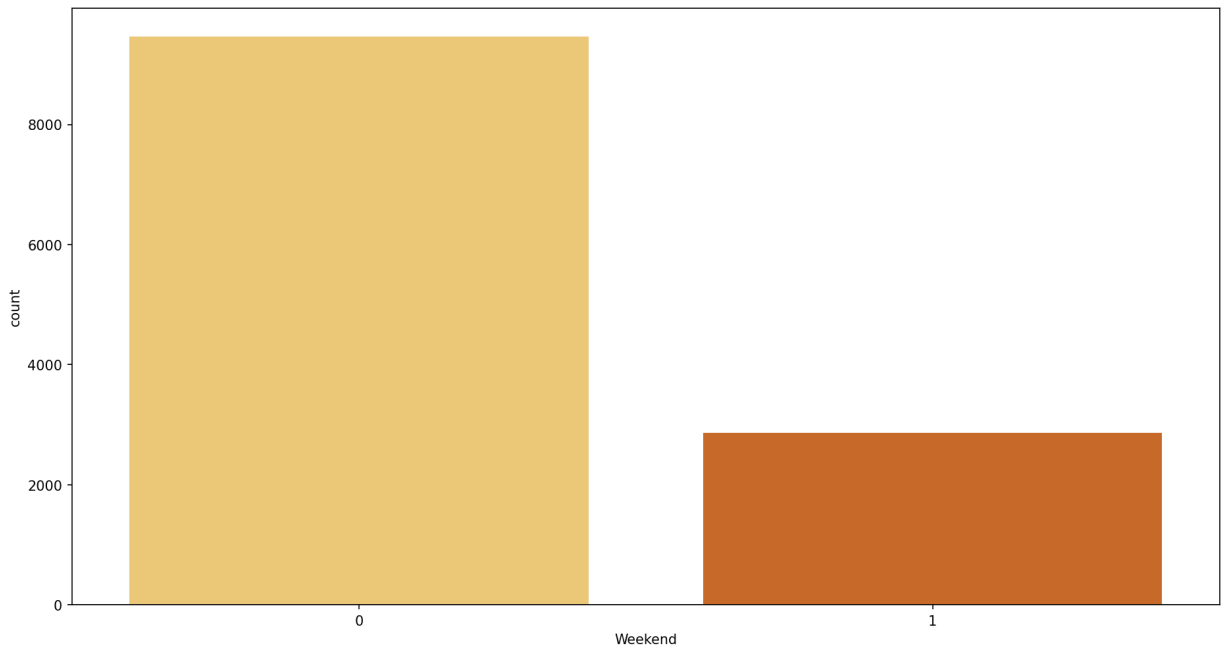
In [19]:
```
from matplotlib.pyplot import figure
figure(figsize=(18,6), dpi=150)
sns.countplot(x ='Month',data = df, palette="YlOrBr")
```

Out[19]:
```
<AxesSubplot:xlabel='Month', ylabel='count'>
```



In [20]:
```
figure(figsize=(15,8), dpi=150)

sns.countplot(x ='Weekend',data = df, palette="YlOrBr")
```

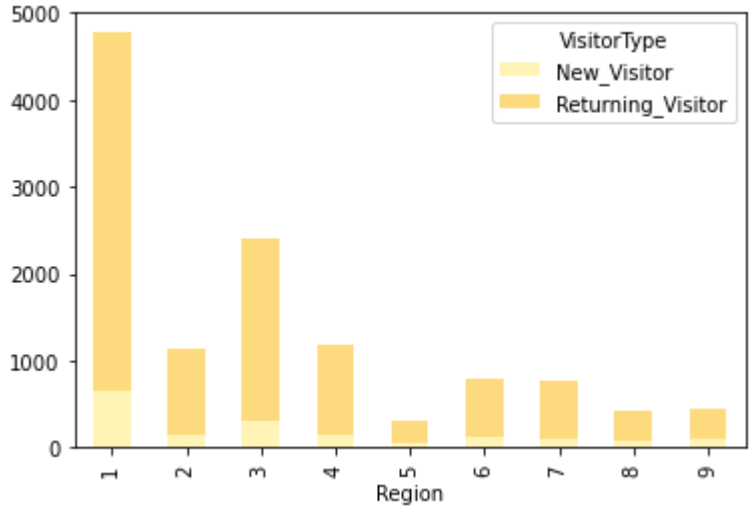
Out[20]:
```
<AxesSubplot:xlabel='Weekend', ylabel='count'>
```



2.3 Feature Variable - region

In [21]:
```
counts = df.groupby(['Region','VisitorType']).count()
Region = df.Region.unique()
counts = counts.unstack(level=1)
counts.columns = counts.columns.droplevel(level=0)
counts=counts.iloc[:, 0:3]
counts
color=sns.color_palette("YlOrBr")
counts[["New_Visitor", "Returning_Visitor"]].plot(kind="bar", stacked=True,co
```

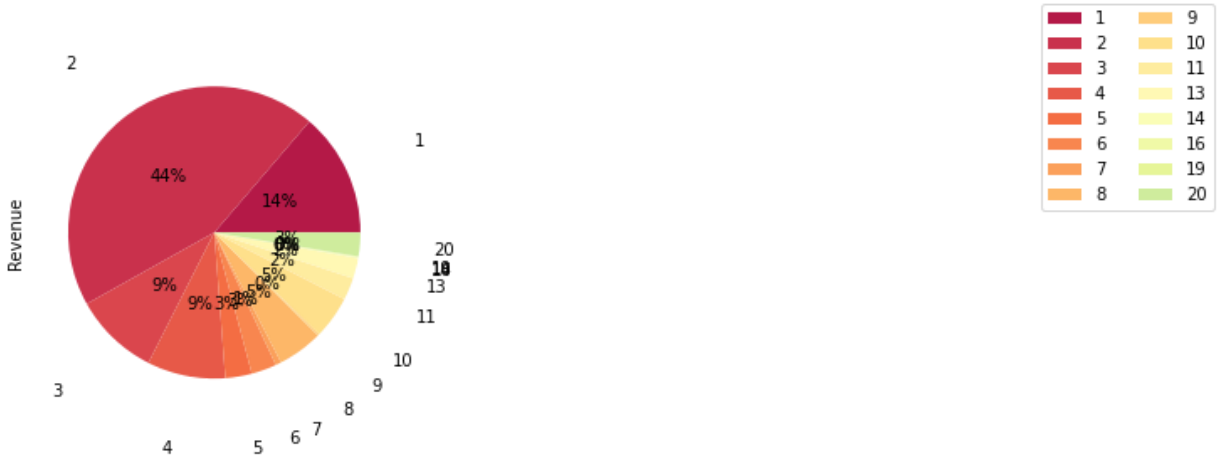
Out[21]:
```
<AxesSubplot:xlabel='Region'>
```



2.4 Feature Variable - Traffic Type & Browsers

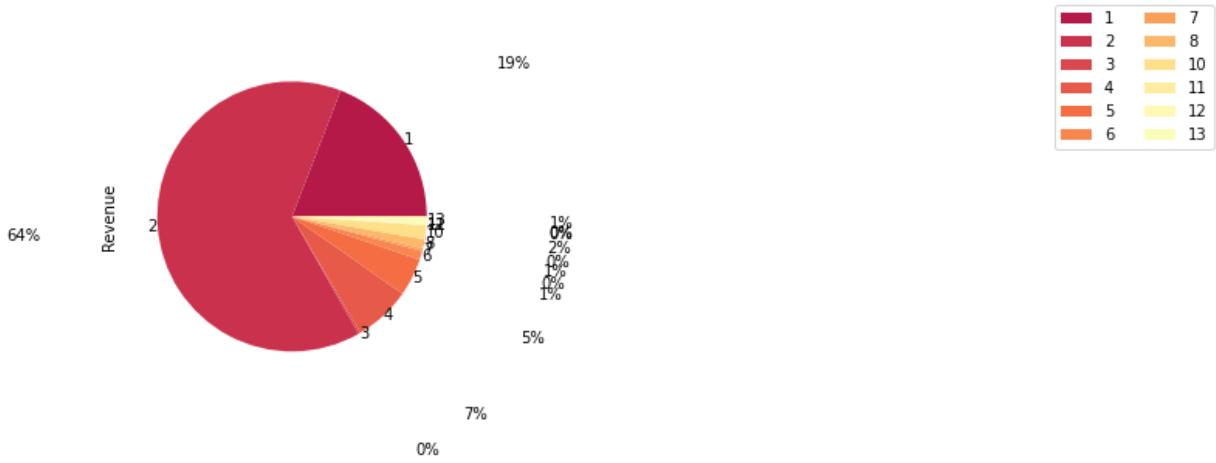
In [22]:
```
#converted traffic type
figure(figsize=(40,40), dpi=150)
colors=sns.color_palette("Spectral",24)
df_purchaser.groupby(['TrafficType']).sum().plot(kind='pie', y='Revenue', aut
plt.legend(loc="upper center", bbox_to_anchor=(3, 1.15), ncol=2)
```

Out[22]:
```
<matplotlib.legend.Legend at 0x7f92a8d8db80>

<Figure size 6000x6000 with 0 Axes>
```

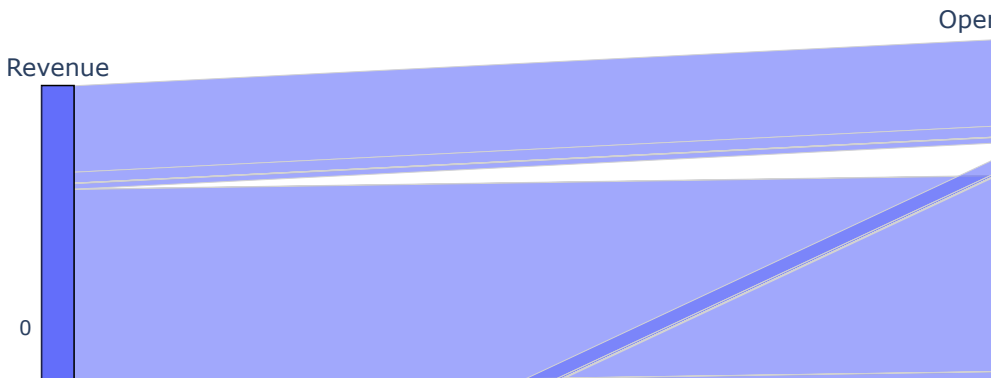


In [23]:
```
#converted brwoser type
figure(figsize=(40,40), dpi=150)
colors=sns.color_palette("Spectral",24)
df.groupby(['Browser']).sum().plot(kind='pie', y='Revenue', autopct='%1.0f%%'
plt.legend(loc="upper center", bbox_to_anchor=(3, 1.15), ncol=2)
```

Out[23]:
```
<matplotlib.legend.Legend at 0x7f92b8eb3580>

<Figure size 6000x6000 with 0 Axes>
```



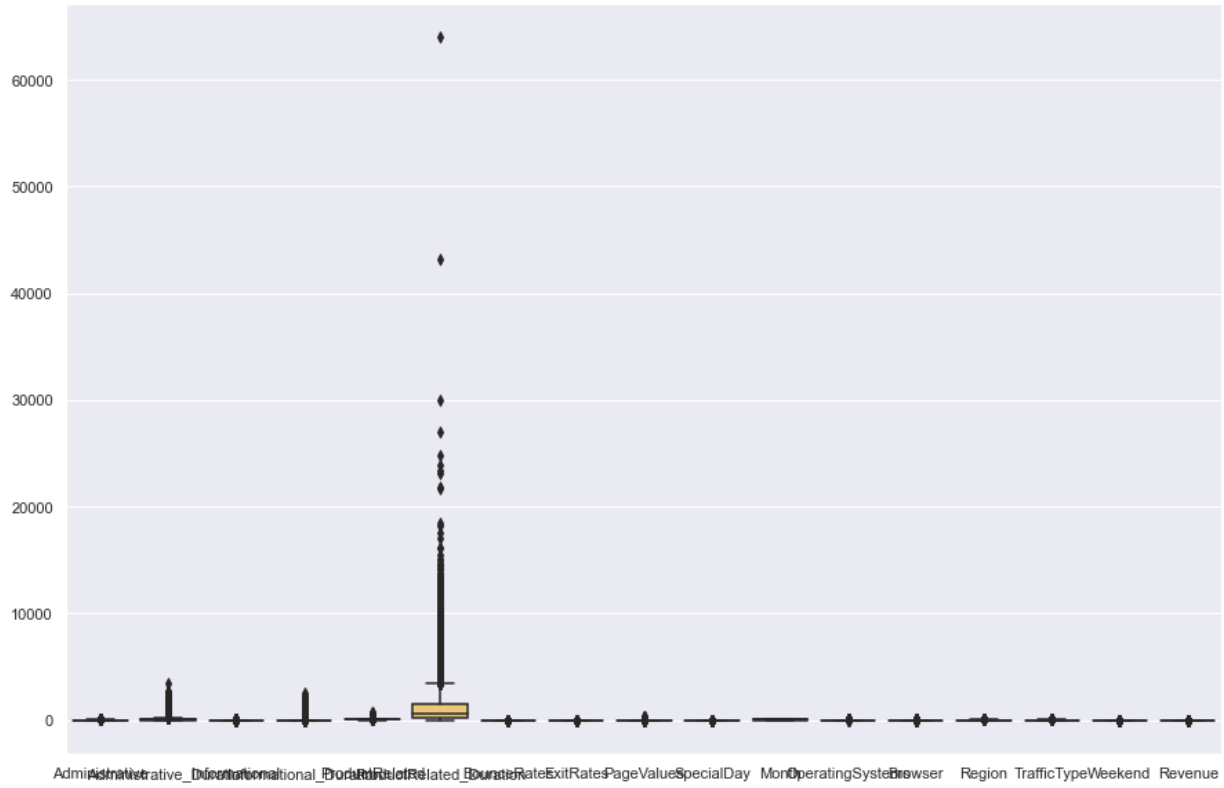
In [24]:
```
#Revenue vs OS vs Brwoser relationship
px.parallel_categories(df[['Revenue','OperatingSystems','Browser']])
```



2.3 Feature Variable - Web

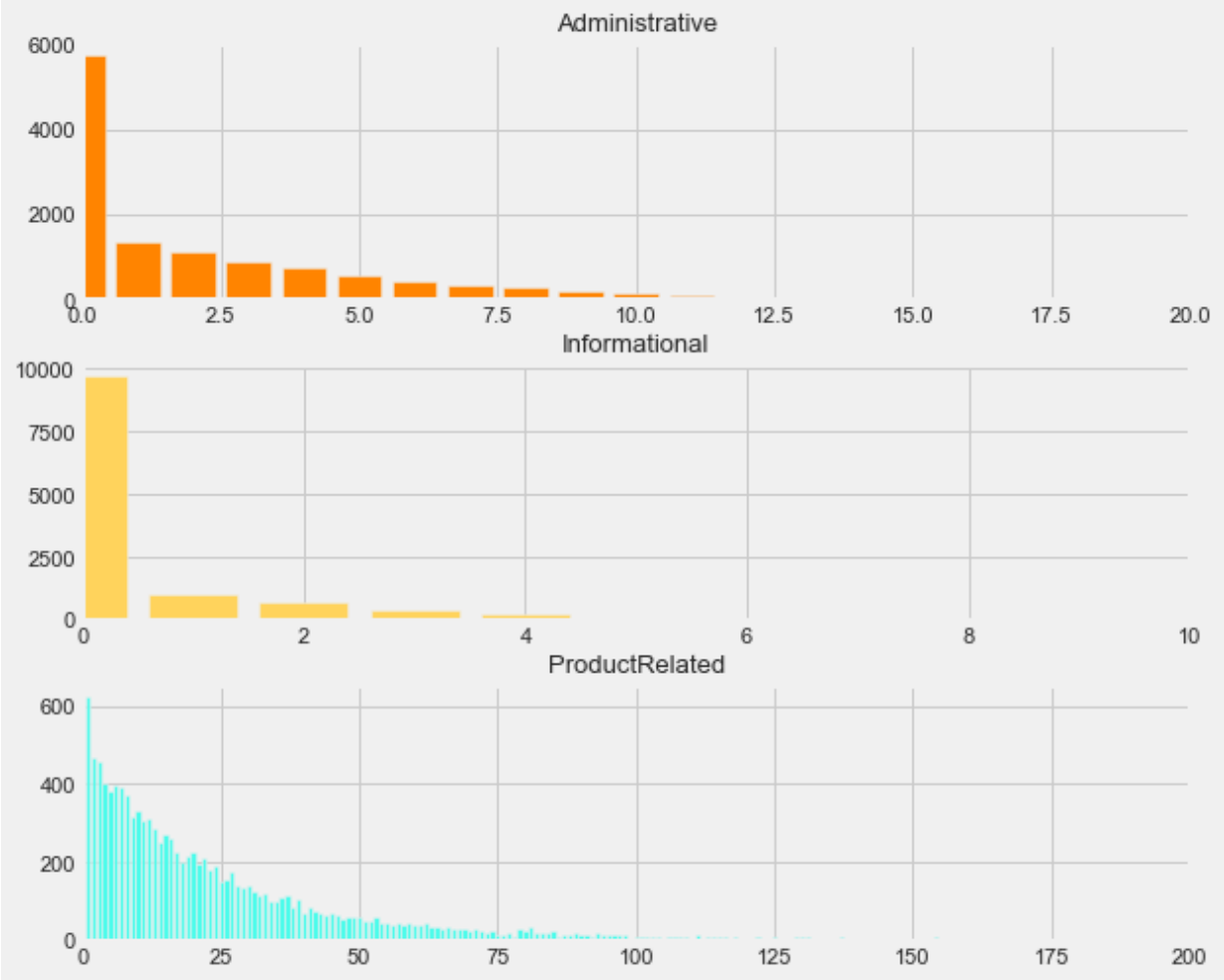
In [25]:
```
#outliners on product related duration
sns.set(rc={'figure.figsize':(15,10)})
sns.boxplot(data=df, palette="YlOrBr")
```

Out[25]:
```
<AxesSubplot:>
```



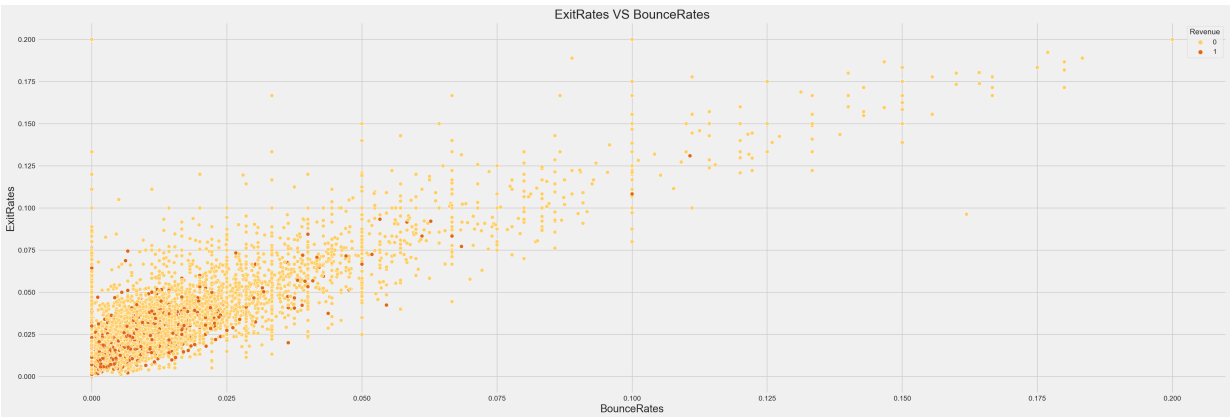
In [26]:

```
# Count of 3 types of page
plt.style.use('fivethirtyeight')
fig,ax = plt.subplots(nrows = 3, ncols = 2,figsize = (17,7))
fig.tight_layout()


ax[0,0].bar(df['Administrative'].value_counts().index,df['Administrative'].va
ax[0,0].set_title('Administrative',size=13)
ax[0,0].set_xlim(0,20)

ax[1,0].bar(df['Informational'].value_counts().index,df['Informational'].valu
ax[1,0].set_title('Informational',size=13)
ax[1,0].set_xlim(0,10)

ax[2,0].bar(df['ProductRelated'].value_counts().index,df['ProductRelated'].va
ax[2,0].set_title('ProductRelated',size=13)
ax[2,0].set_xlim(0,200)


fig.delaxes(ax[0,1])
fig.delaxes(ax[1,1])
fig.delaxes(ax[2,1])
```

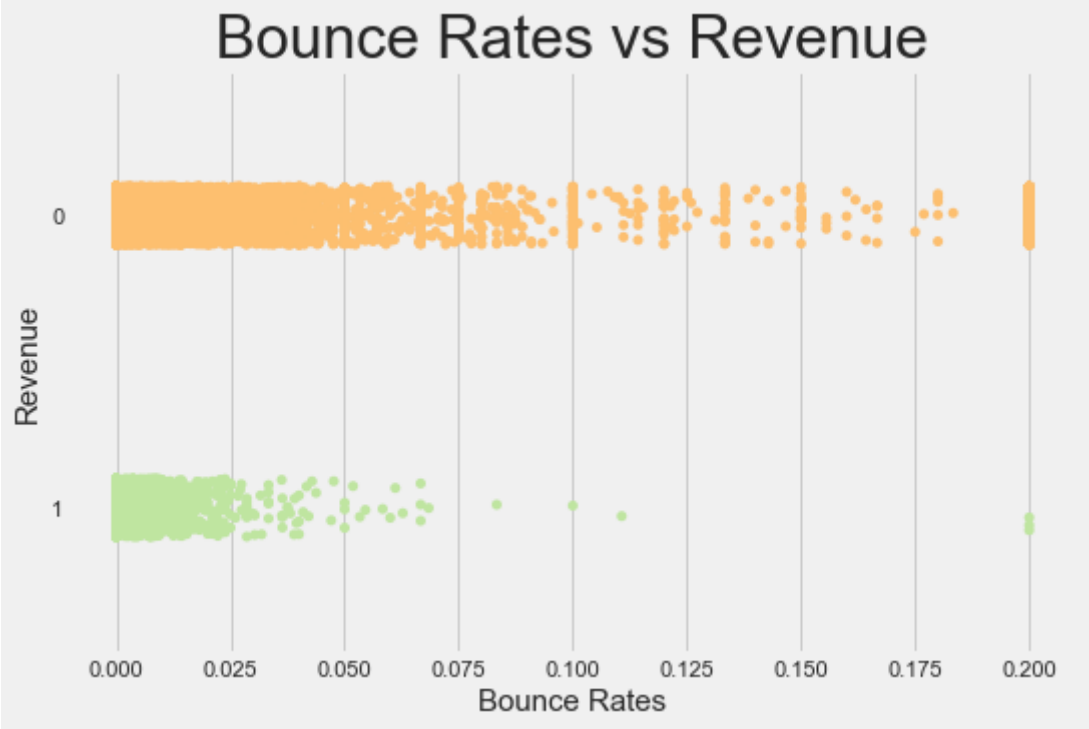


In [27]:
```
#Exit Rate vs Bounce Rate
figure(figsize=(30,10), dpi=150)
sns.scatterplot(df['BounceRates'],df['ExitRates'],hue = df['Revenue'],palette
```

Out[27]:
```
Text(0.5, 1.0, 'ExitRates VS BounceRates')
```

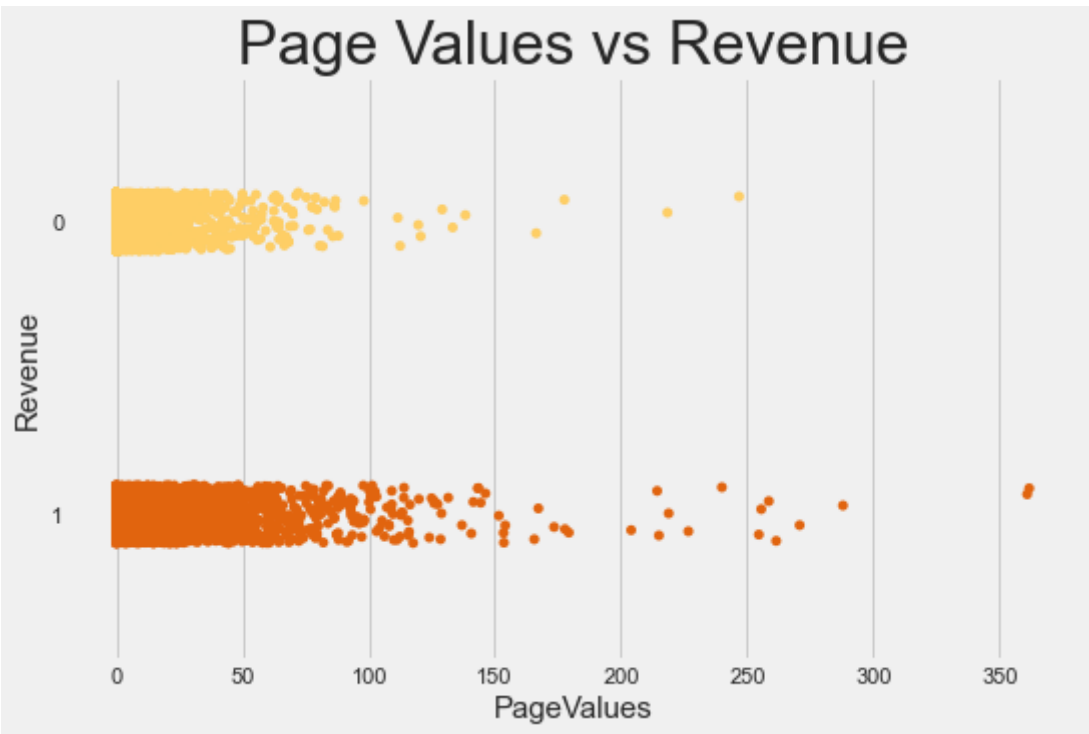


In [28]:
```
#Page value vs Revenue
plt.rcParams['figure.figsize'] = (8, 5)

sns.stripplot(df['BounceRates'], df['Revenue'], palette = 'Spectral', orient =
plt.title('Bounce Rates vs Revenue', fontsize = 30)
plt.xlabel('Bounce Rates', fontsize = 15)
plt.ylabel('Revenue', fontsize = 15)
plt.show()
```



In [29]:

```
#Page values vs Revenue
plt.rcParams['figure.figsize'] = (8, 5)

sns.stripplot(df['PageValues'], df['Revenue'], palette="YlOrBr", orient = 'h'
plt.title('Page Values vs Revenue', fontsize = 30)
plt.xlabel('PageValues', fontsize = 15)
plt.ylabel('Revenue', fontsize = 15)
plt.show()
```

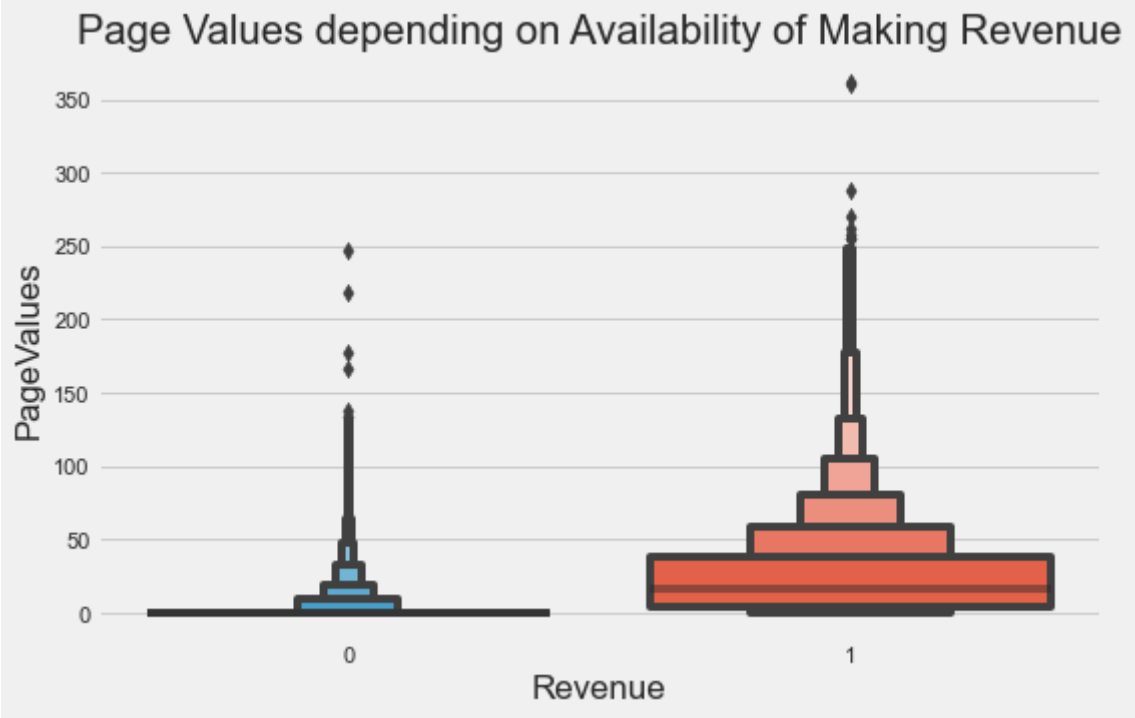


In [30]:

```
#Page value vs revenue
sns.boxenplot(x=df['Revenue'],y=df['PageValues']).set_title('Page Values depe
```

Out[30]:

```
Text(0.5, 1.0, 'Page Values depending on Availability of Making Revenue')
```


Page Values depending on Availability of Making Revenue
PageValues
350
300
250
200
150
100
50
0
0
1
Revenue