EDA

June 16, 2021

1 Projekt 2 - EDA

Online shoppers intention zawierają informację o aktywności użytkowników w sesji i czy w trakcie sesji użytkownik dokonał jakiegoś zakupu

[189]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import pandas_profiling
import copy
from sklearn.decomposition import PCA
import sklearn.metrics
from sklearn import manifold
```

1.1 Podstawowe informacje

W celu pozyskania podstawowych informacji użyjemy narzędzia do zautomatyzowanej eksploracji danych **pandas_profiling**.

[190]:
```
df=pd.read_csv("online_shoppers_intention.csv")
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
```

Nie ma braków danych Zmienna Month i VisitorsType (Returning, New, Other) typu object Zmienna Weekend i Revenue typu bool.

Zmienne `Administrative`, `Informational`, `ProductRelated` przedstawiają ile stron danego typu odwiedził użytkownik

`Administrative_Duration`, `Informational_Duration`, `ProductRelated_Duration` ile czasu w sumie użytkownik spędził na stronie danego typu

`BounceRates` - procent stron które użytkownik odwiedził bez żadnej dalszej interakcji

`SpecialDay` - jak blisko specjalnego dnia była sesja

[191]:
```
df=pd.read_csv("online_shoppers_intention.csv")
df=df.dropna()
```

[192]:
```
df.profile_report()
```

```
Summarize dataset:   0%|          | 0/31 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

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

[192]:

1.1.1 Wnioski:

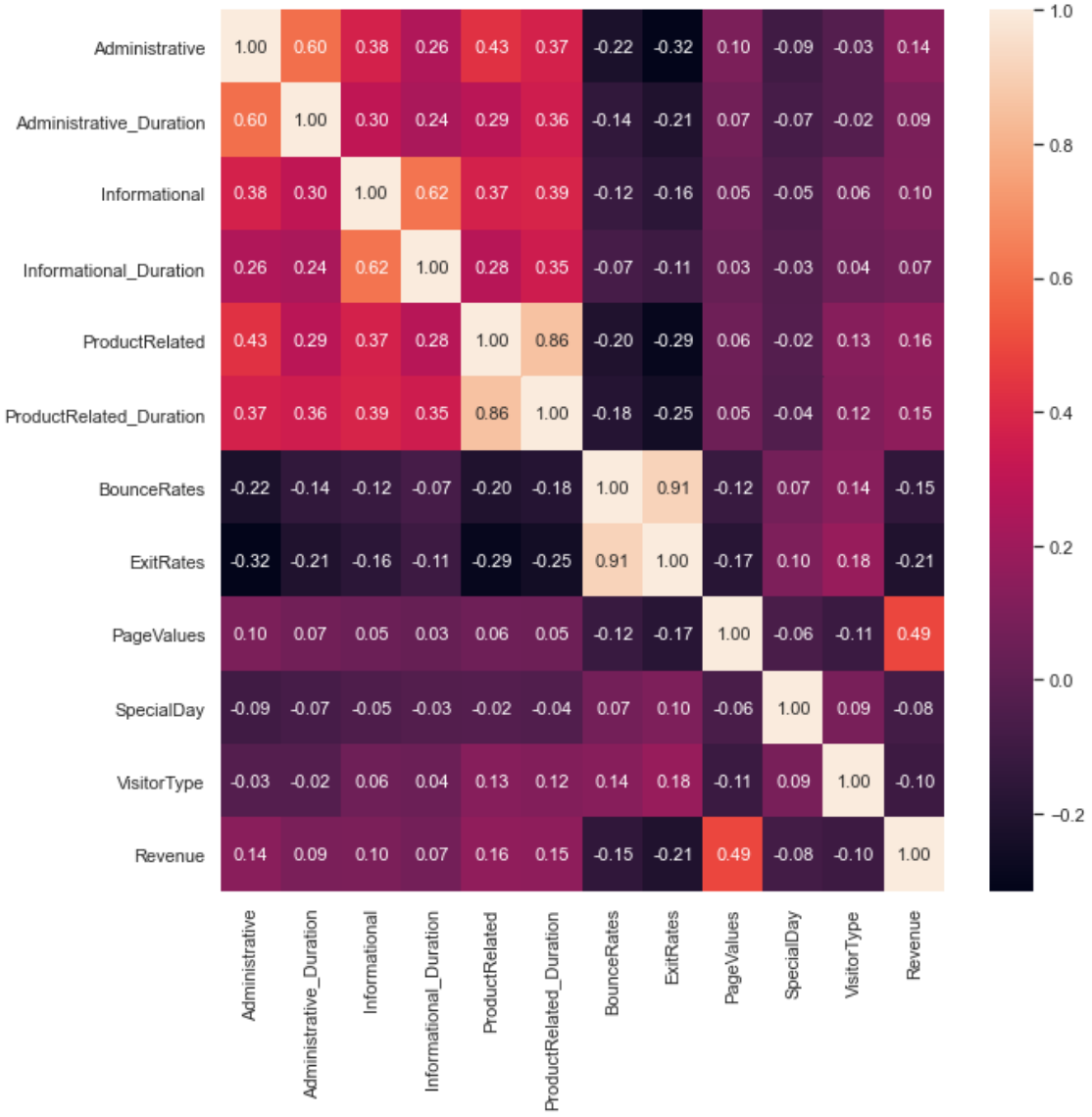
- dużo zer w zmiennych opisujących pobyt na konkretnych typach stron
- Większość wizyt nie kończy się transakcją
- Przewaga przeglądarki 1. i 2.
- Czy jedna sesja = jedna witryna? Czym są zmienne opisujące strony

1.2 Macierz korelacji

[193]:
```
nums=["Administrative", "Administrative_Duration",
      "Informational", "Informational_Duration",
      "ProductRelated", "ProductRelated_Duration",
     "BounceRates", "ExitRates", "PageValues",
```

```
    "SpecialDay", "VisitorType", "Revenue"]
df_hm=df[nums]
df_hm=df_hm.replace(False, 0)
df_hm=df_hm.replace(True, 1)
df_hm=df_hm.replace("New_Visitor", 0)
df_hm=df_hm.replace("Returning_Visitor", 1)
df_hm=df_hm.replace("Other", 0.5)   #jest ich stosunkowo mało więc to bez␣
↪większego znaczenia
df_hm=df_hm.replace("")
pd.to_numeric(df_hm["VisitorType"])
df_hm=df_hm
plt.figure(figsize=(10, 10))
sns.heatmap(df_hm.corr(), annot=True, annot_kws={'size': 11}, fmt='.2f')
```

[193]: <AxesSubplot:>



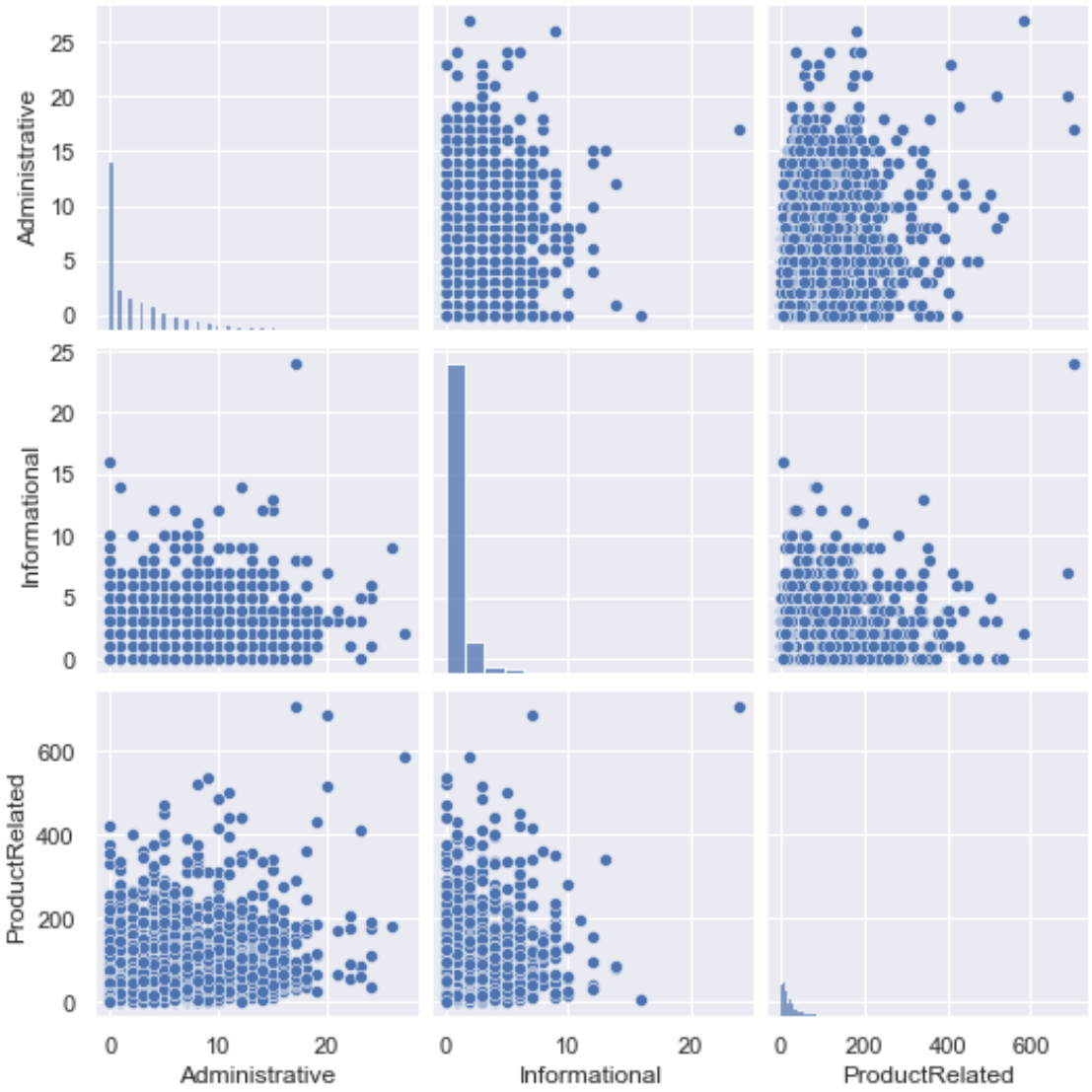
1.2.1 Wnioski/pytania do macierzy korelacji

- Ujemna korelacjia pomiędzy `SpecialDay` i wszystkimi zmiennymi dot. czasu spędzonego na stronach oraz zmiennej `Revenue` - zaskakujące
- Ujemna korelacja pomiędzy `ExitRates` i `Revenue` - też nieintuicyjne
- Czym tak naprawdę jest `PageValues`
- Bardzo duża pomiędzy `ExitRates` i `BounceRates` - czy to ludzie którzy po odpaleniu strony od razu ją zamknęli?

[194]:
```
sns.set()
cols = ['Administrative', 'Informational',
       "ProductRelated"]
```

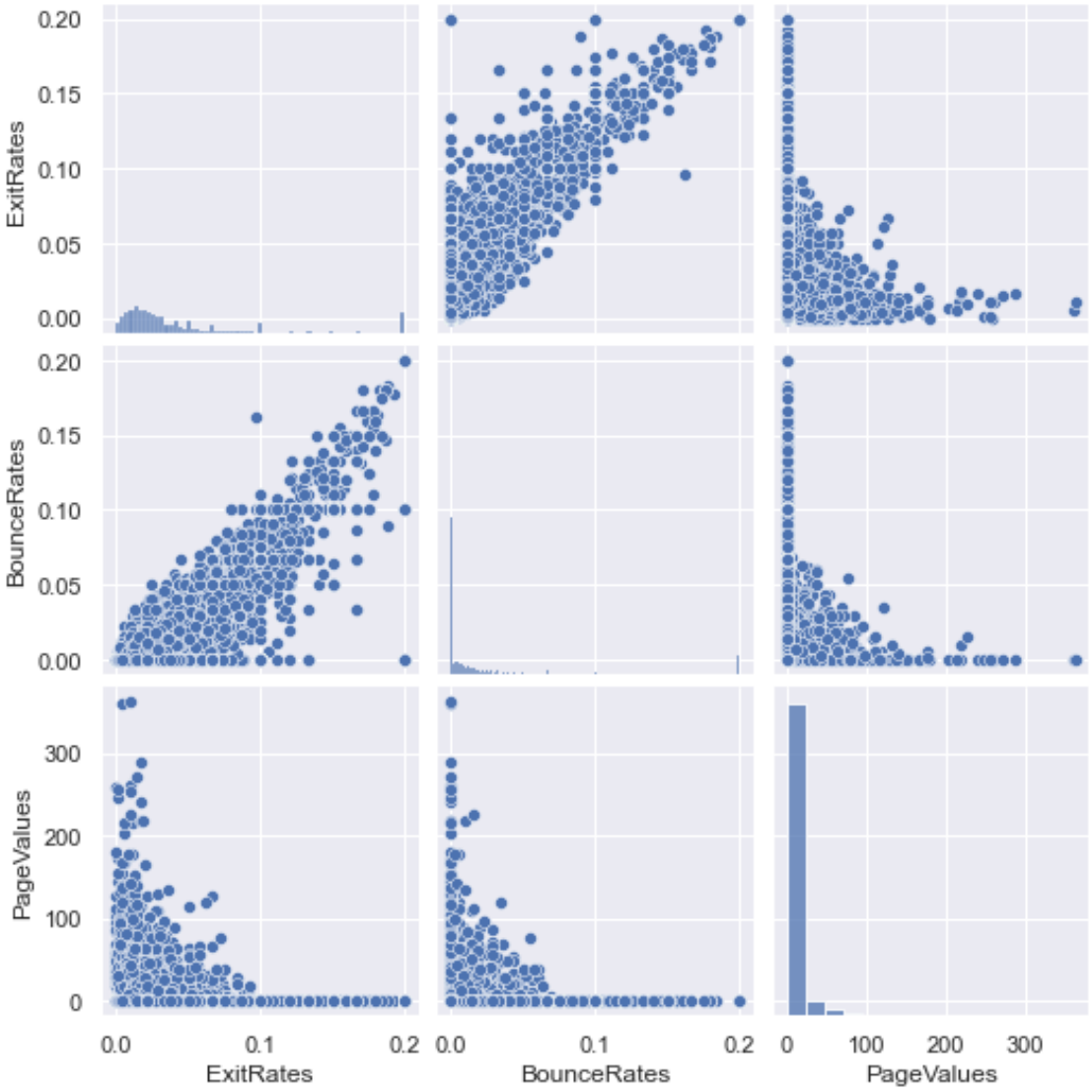
```
sns.pairplot(df[cols], size = 2.5)
plt.show();
```

```
D:\Anaconda\lib\site-packages\seaborn\axisgrid.py:1969: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



[195]:
```
sns.set()
cols = ['ExitRates', 'BounceRates',
       "PageValues"]
sns.pairplot(df[cols], size = 2.5)
plt.show();
```

```
D:\Anaconda\lib\site-packages\seaborn\axisgrid.py:1969: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



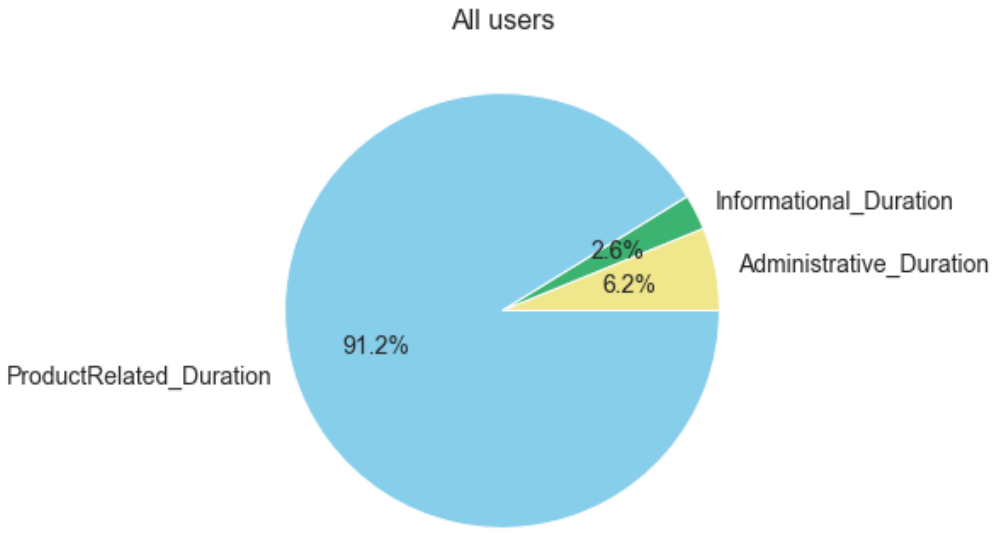
Dla ExitRates/BounceRates jedna obserwacja znajduje się w innej częsci trójkąta

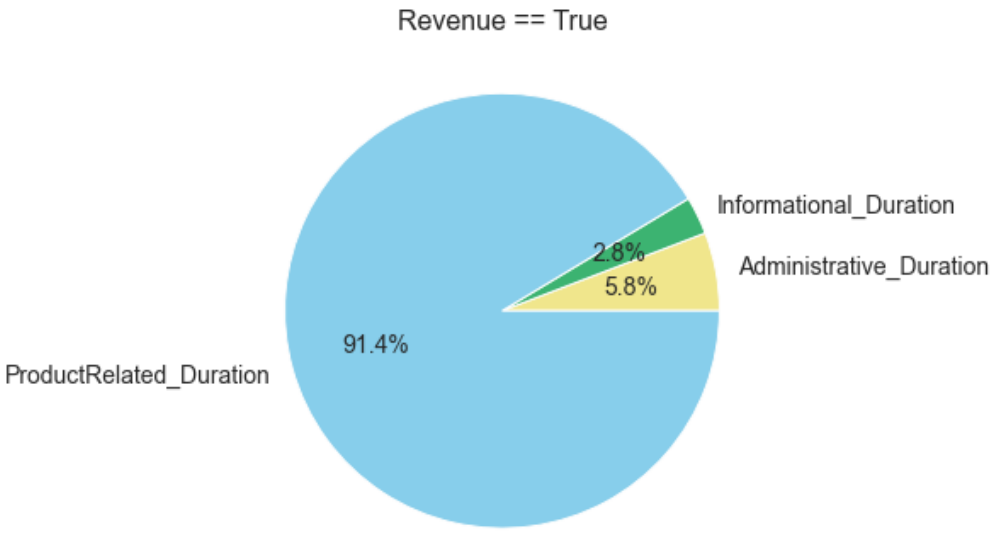
1.3 Proporcje w spędzonym czasie w zależności od zakupu

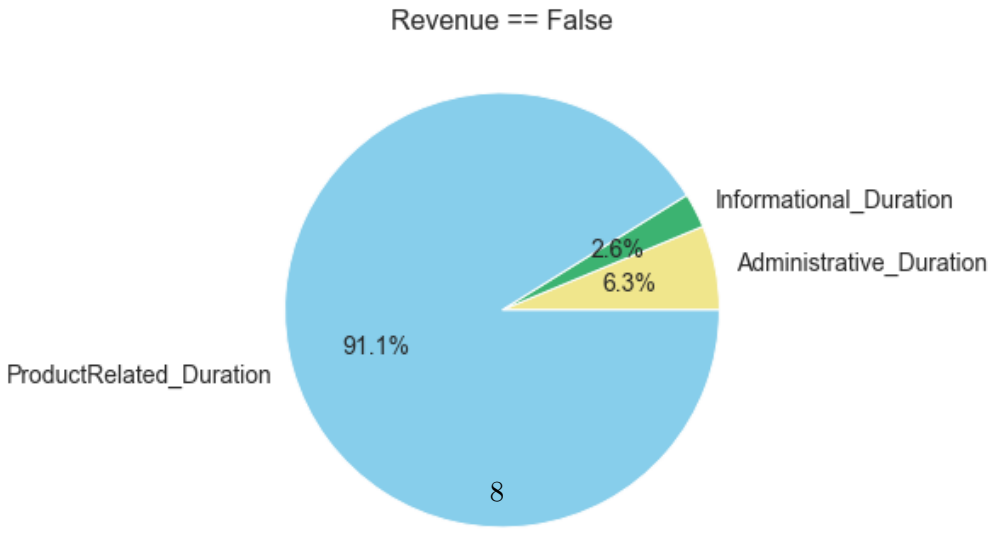
[196]:
```
inf_dur=df["Informational_Duration"].sum()
prod_dur=df["ProductRelated_Duration"].sum()
adm_dur=df["Administrative_Duration"].sum()
labels = ['Administrative_Duration', 'Informational_Duration',␣
 ↪'ProductRelated_Duration']
```

```
sizes = [adm_dur, inf_dur, prod_dur]
colors=["khaki", "mediumseagreen", "skyblue"]
fig, axs=plt.subplots(ncols=1, nrows=3, figsize=(20,20))
axs[0].pie(sizes, labels=labels,  autopct='%1.1f%%', textprops={'fontsize':␣
↪14}, colors=colors)
axs[0].set_title("All users", fontsize=16)
X=df[df["Revenue"]==True]
inf_dur=X["Informational_Duration"].sum()
prod_dur=X["ProductRelated_Duration"].sum()
adm_dur=X["Administrative_Duration"].sum()
labels = ['Administrative_Duration', 'Informational_Duration',␣
↪'ProductRelated_Duration']
sizes = [adm_dur, inf_dur, prod_dur]
colors=["khaki", "mediumseagreen", "skyblue"]
axs[1].pie(sizes, labels=labels,  autopct='%1.1f%%', textprops={'fontsize':␣
↪14}, colors=colors)
axs[1].set_title("Revenue == True", fontsize=16)
X=df[df["Revenue"]==False]
inf_dur=X["Informational_Duration"].sum()
prod_dur=X["ProductRelated_Duration"].sum()
adm_dur=X["Administrative_Duration"].sum()
labels = ['Administrative_Duration', 'Informational_Duration',␣
↪'ProductRelated_Duration']
sizes = [adm_dur, inf_dur, prod_dur]
colors=["khaki", "mediumseagreen", "skyblue"]
axs[2].pie(sizes, labels=labels,  autopct='%1.1f%%', textprops={'fontsize':␣
↪14}, colors=colors)
axs[2].set_title("Revenue == False", fontsize=16)
```

[196]: Text(0.5, 1.0, 'Revenue == False')


All users
Informational_Duration
2.6%
Administrative_Duration
6.2%
91.2%
ProductRelated_Duration

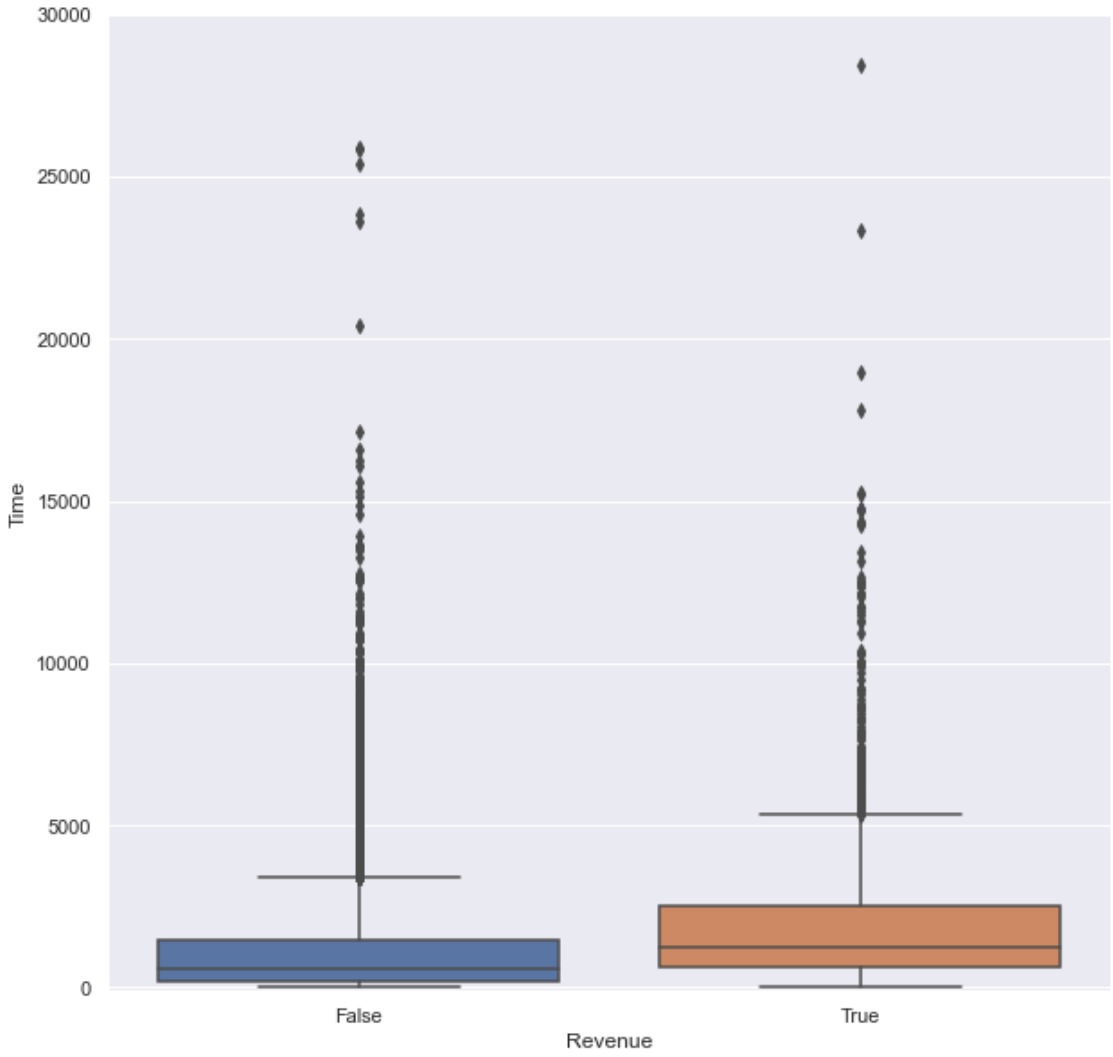

Revenue == True
Informational_Duration
2.8%
Administrative_Duration
5.8%
91.4%
ProductRelated_Duration


Revenue == False
Informational_Duration
2.6%
Administrative_Duration
6.3%
91.1%
ProductRelated_Duration

1.4 Długość pobytu na stronie a zakup

[197]:
```
X=df.copy()
plt.figure(figsize=(10, 10))
X["Time"]=df["Informational_Duration"]+df["ProductRelated_Duration"]+df["Administrative_Durati
sns.boxplot(data=X, x="Revenue", y="Time")
plt.ylim(-100, 30000)
```

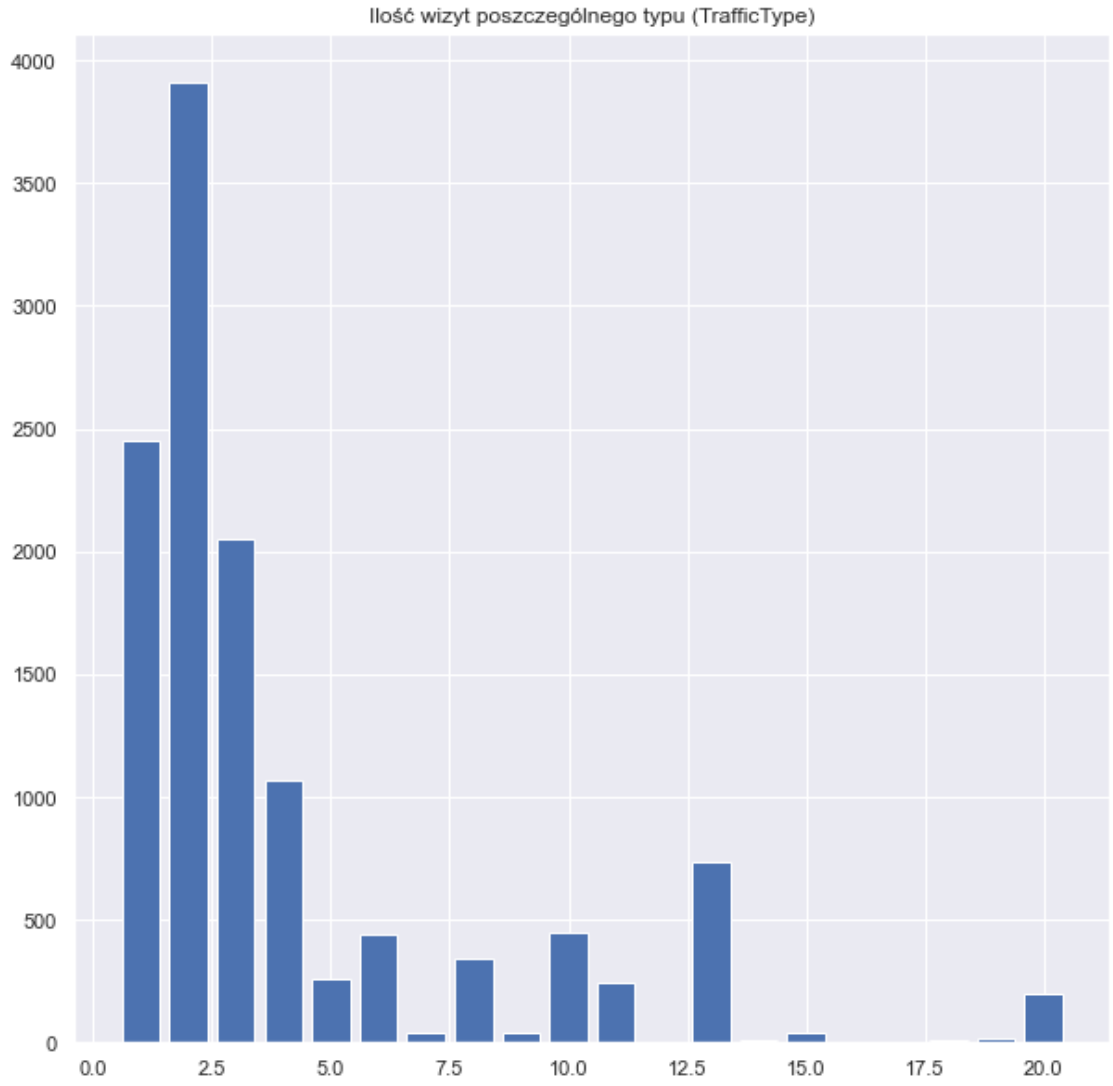
[197]: (-100.0, 30000.0)



1.5 Rodzaje ruchu

[198]:
```
plt.figure(figsize=(10, 10))
X = pd.crosstab(df['TrafficType'], df['Revenue'])
X.reset_index(inplace=True)
plt.bar(X["TrafficType"], X[True]+X[False])
plt.title("Ilość wizyt poszczególnego typu (TrafficType)")
```

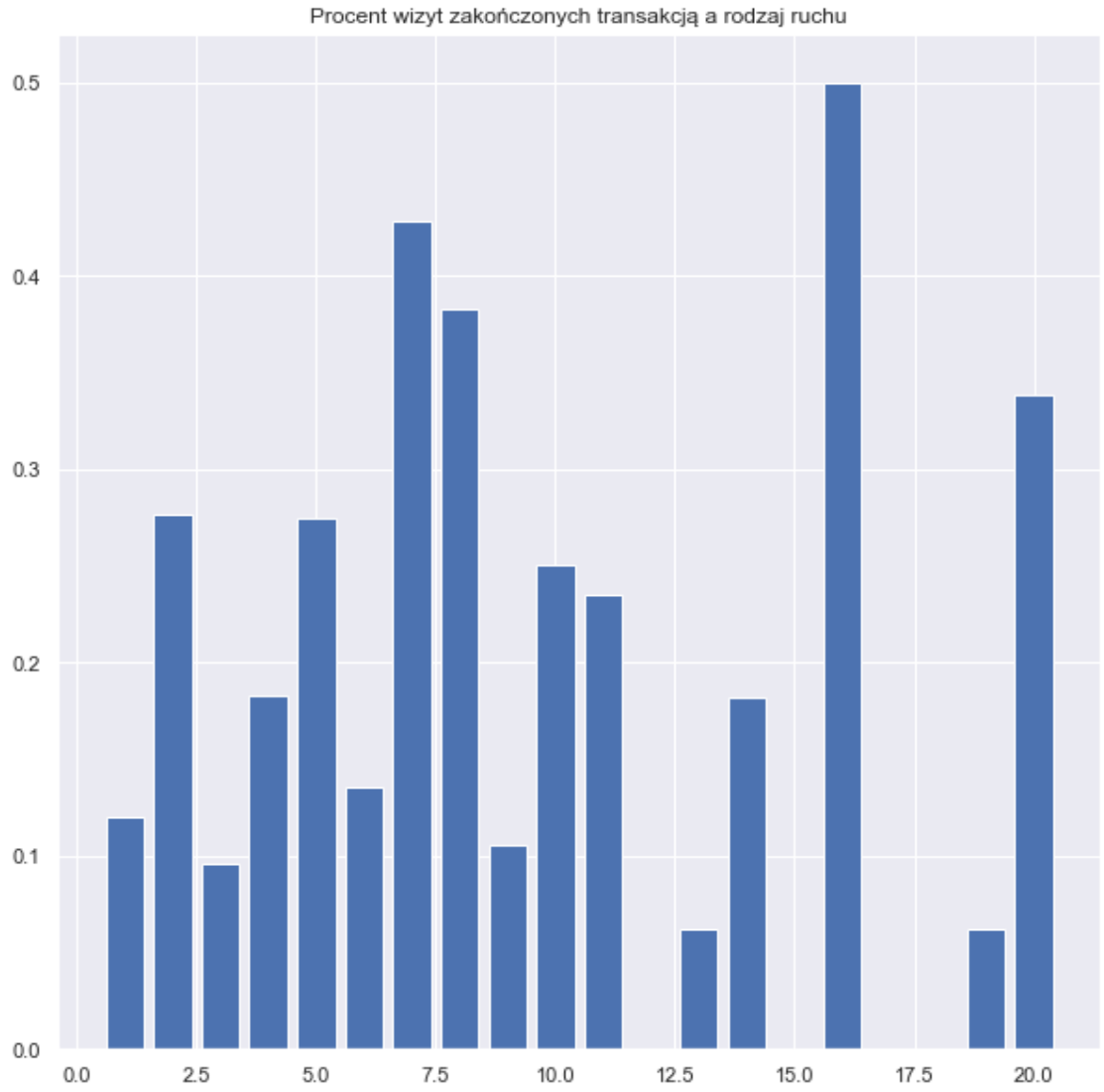
[198]:
```
Text(0.5, 1.0, 'Ilość wizyt poszczególnego typu (TrafficType)')
```



1.6 Rodzaj ruchu a zakup

[199]:
```
plt.figure(figsize=(10, 10))
X = pd.crosstab(df['TrafficType'], df['Revenue'])
X.reset_index(inplace=True)
plt.bar(X["TrafficType"], X[True]/X[False])
plt.title("Procent wizyt zakończonych transakcją a rodzaj ruchu")
```

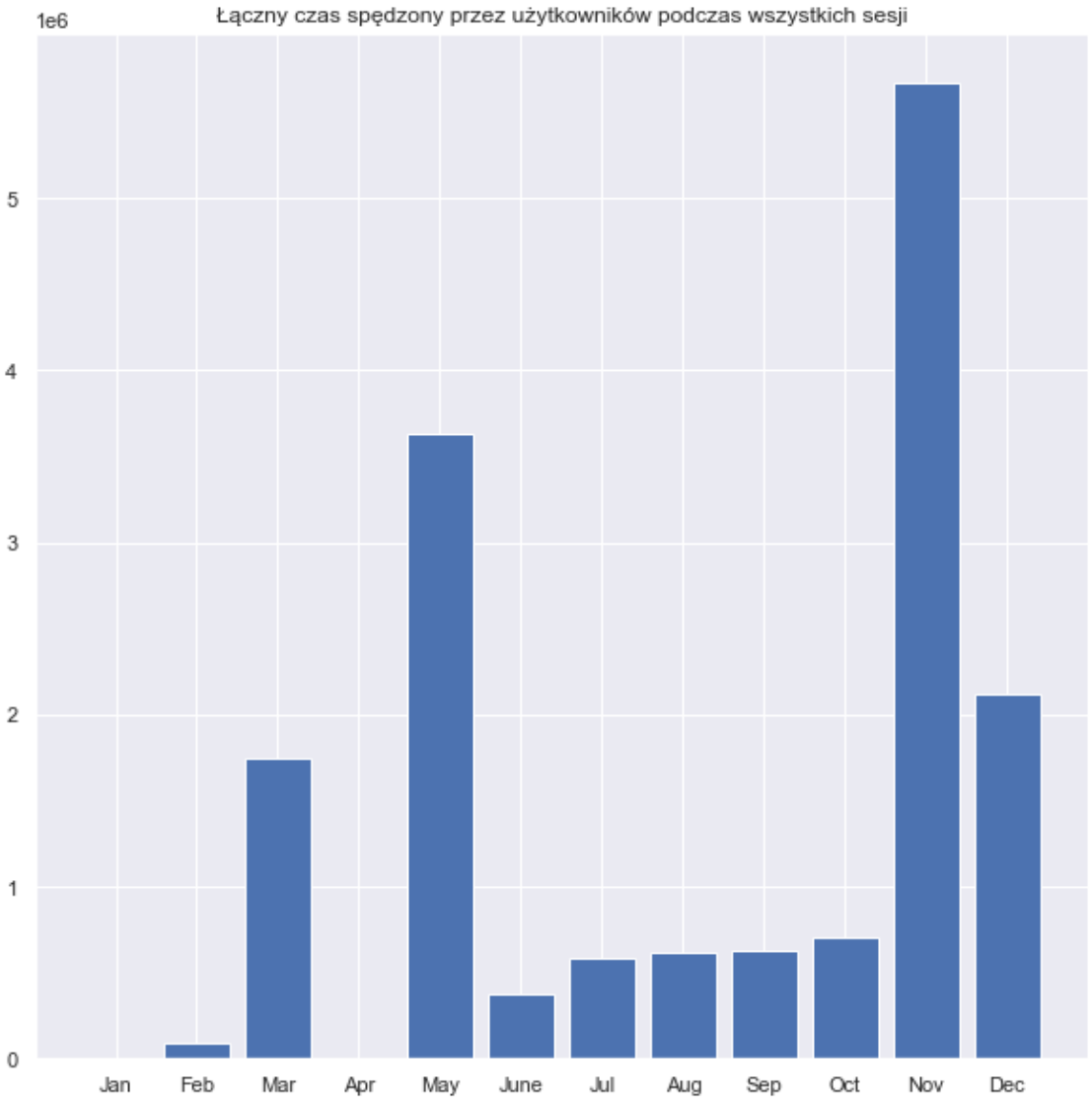
[199]:
```
Text(0.5, 1.0, 'Procent wizyt zakończonych transakcją a rodzaj ruchu')
```



1.7 Miesiące

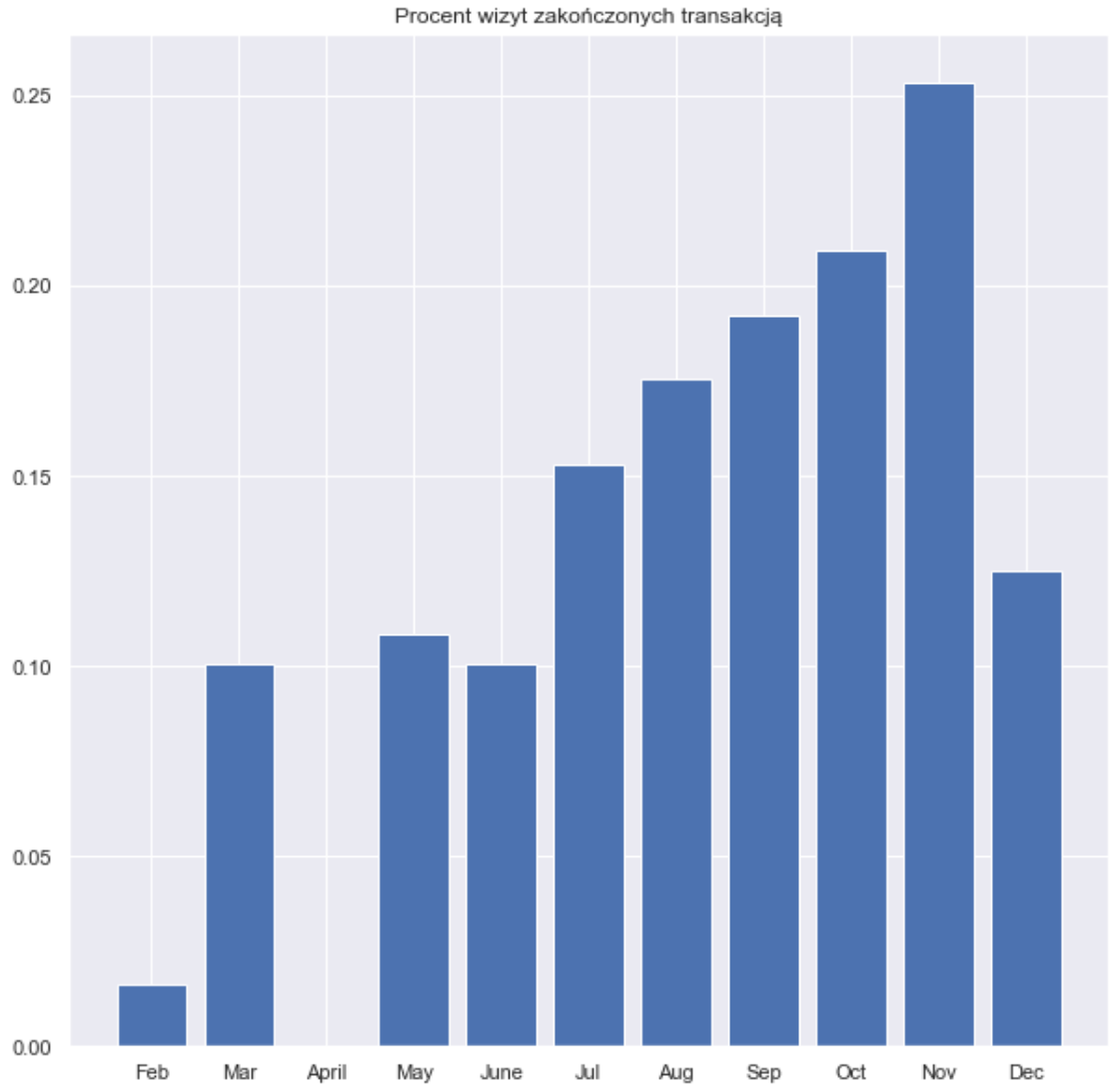
[200]:
```
X=df.copy()
months = ["Jan", "Feb", "Mar", "Apr", "May", "June",
          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
X["Time"]=df["Informational_Duration"]+df["ProductRelated_Duration"]+df["Administrative_Durati
X['Month'] = pd.Categorical(X['Month'], categories=months, ordered=True)
X=X.groupby(["Month"]).agg({'Time':'sum'}).reset_index()
X = X.sort_values(by="Month")
plt.figure(figsize=(10, 10))
plt.bar(X["Month"], X["Time"])
plt.title("Łączny czas spędzony przez użytkowników podczas wszystkich sesji")
```

[200]:
```
Text(0.5, 1.0, 'Łączny czas spędzony przez użytkowników podczas wszystkich
sesji')
```



[201]:
```
X=df.copy()
X=X.replace(True, 1)
X=X.replace(False, 0)
months = ["Jan", "Feb", "Mar", "April", "May", "June",
          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
X['Month'] = pd.Categorical(X['Month'], categories=months, ordered=True)
X=X.groupby(["Month"]).agg({'Revenue':['sum', "count"]}).reset_index()
X = X.sort_values(by="Month")
X.columns=["Month", "sum", "count"]
plt.figure(figsize=(10, 10))
plt.bar(X["Month"], X["sum"]/X["count"])
plt.title("Procent wizyt zakończonych transakcją")
```

[201]:
```
Text(0.5, 1.0, 'Procent wizyt zakończonych transakcją')
```



[202]:
```
df[(df["Month"]=="Feb") & (df["Revenue"]==True)]
```

[202]:

	Administrative	Administrative_Duration	Informational	\
65	3	87.833333	0	
76	10	1005.666667	0	
101	4	61.000000	0	

	Informational_Duration	ProductRelated	ProductRelated_Duration	\
65	0.0	27	798.333333	
76	0.0	36	2111.341667	
101	0.0	19	607.000000	

	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	\
65	0.000000	0.012644	22.916036	0.8	Feb	2	
76	0.004348	0.014493	11.439412	0.0	Feb	2	
101	0.000000	0.026984	17.535959	1.0	Feb	1	

	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
65	2	3	1	Returning_Visitor	False	True
76	6	1	2	Returning_Visitor	False	True
101	1	7	4	Returning_Visitor	True	True

1.8 PCA

[203]:
```
nums=["Administrative_Duration",
      "Informational_Duration",
    "ProductRelated_Duration", "SpecialDay"]  #bierzemy zmienne opisujące ␣
 ↪użytkowników a nie strony
X=df[nums]
X=(X-X.mean())/X.std()
pca=PCA(2)
pca.fit(X)
plt.figure(figsize=(10, 10))
X_pca=pca.transform(X)
sns.scatterplot(X_pca[:, 0], X_pca[:,1], hue=df["Revenue"])
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

[203]: <AxesSubplot:>

1.8.1 Wnioski:

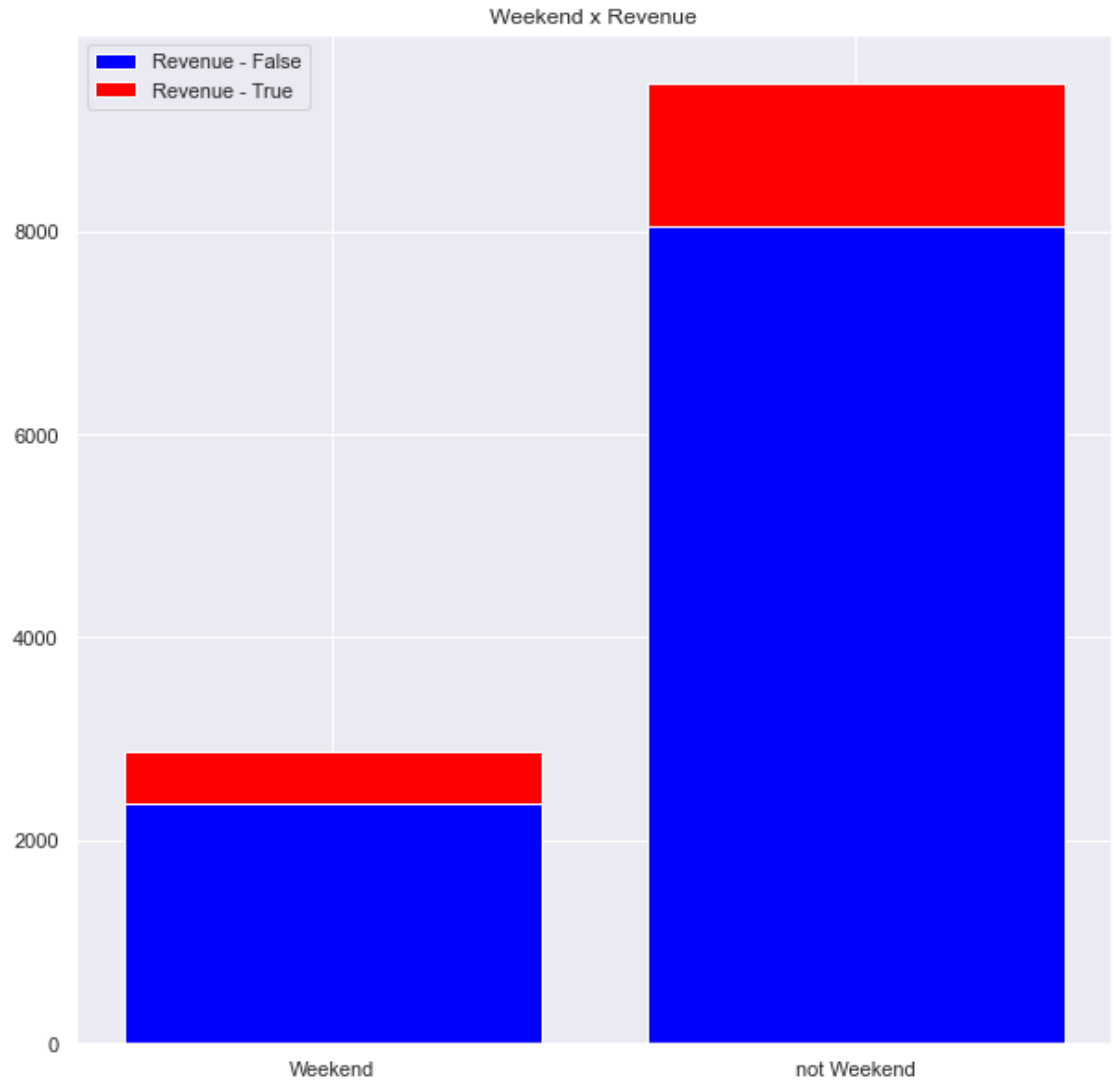
Z powodu dużej ilości zer w naszych danych, PCA nie daje szczególnie dobrych rezultatów. Widoczne na wykresie grupki wynikają z wartości zmiennej "SpecialDay", która przyjmuje w końcu ograniczoną ilość wartości - zależną od dnia, nie użytkownika.

1.9 Weekend

[204]:
```
df.Weekend.value_counts()
```

[204]:
```
False    9462
True     2868
Name: Weekend, dtype: int64
```

[205]:
```
plt.figure(figsize=(10, 10))
tmp=df[["Weekend", "Revenue"]].groupby(["Revenue", "Weekend"]).size().tolist()
plt.bar(["Weekend","not Weekend"], list(reversed(tmp[0:2])), label='Revenue -␣
↪False', color="blue")
plt.bar(["Weekend","not Weekend"], list(reversed(tmp[2:4])),␣
↪bottom=list(reversed(tmp[0:2])),
      label='Revenue - True', color="red")
plt.legend()
plt.title("Weekend x Revenue")
plt.show()
```

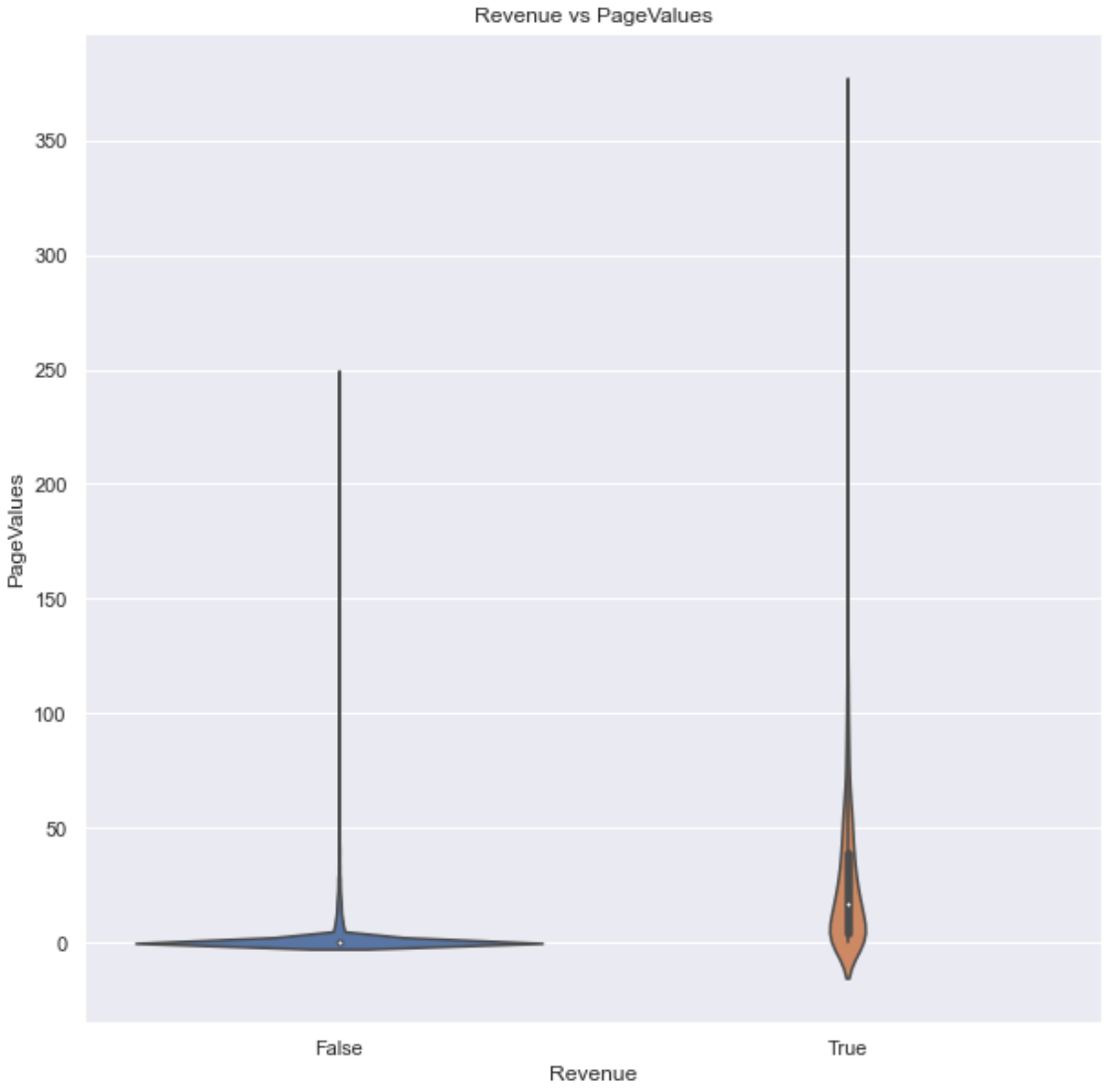


- Wiecej sesji odbyło się w ciagu tygodnia
- Revenue w zależnosci od weekendu rozkłada sie podobnie w obu przypadkach, większość stanowi Revenue = False

1.10 Revenue vs PageValues/ ExitRates/ BoinceRates

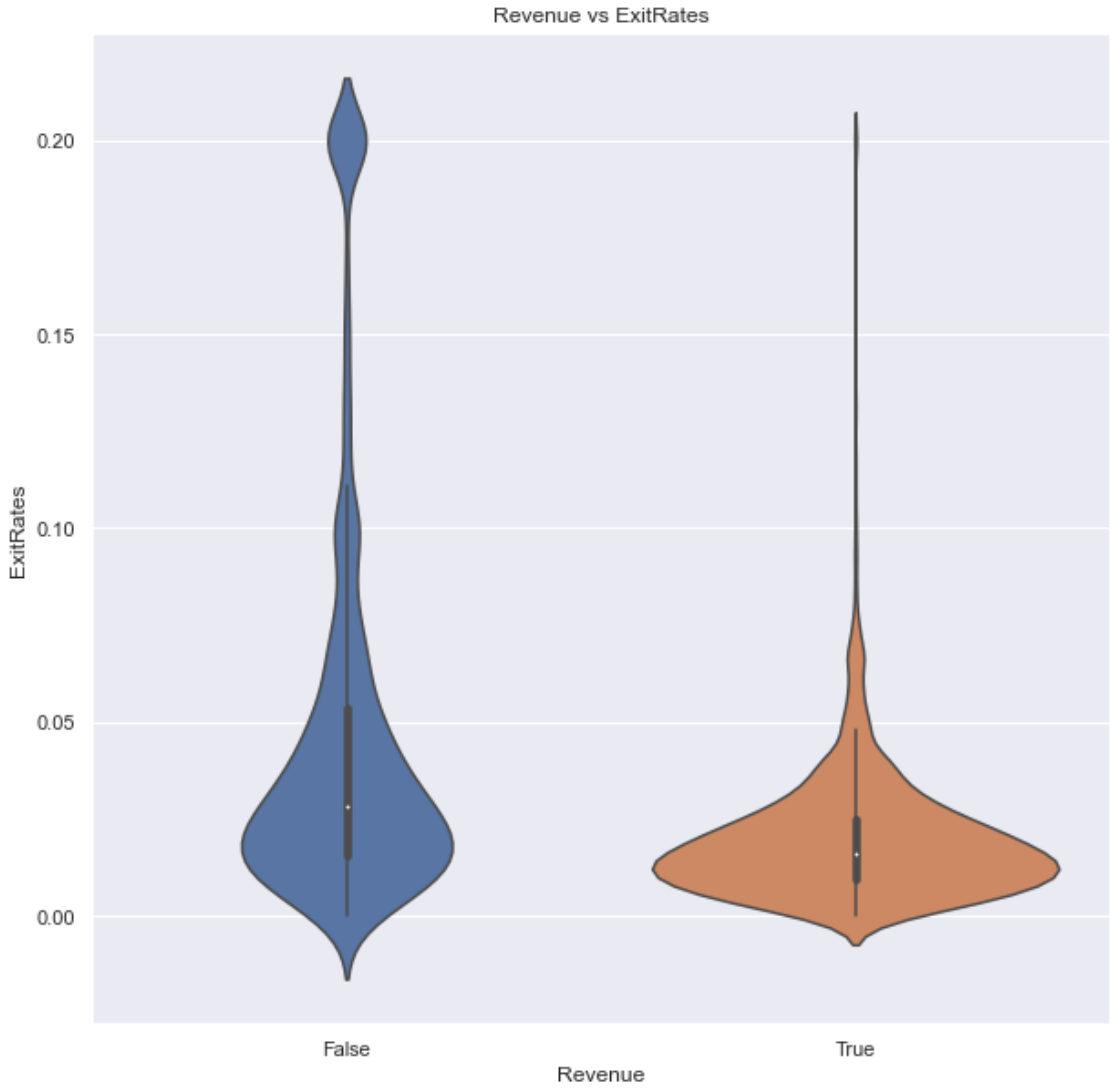
[206]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['Revenue'], y = df['PageValues'])
plt.title('Revenue vs PageValues')
```

[206]: Text(0.5, 1.0, 'Revenue vs PageValues')



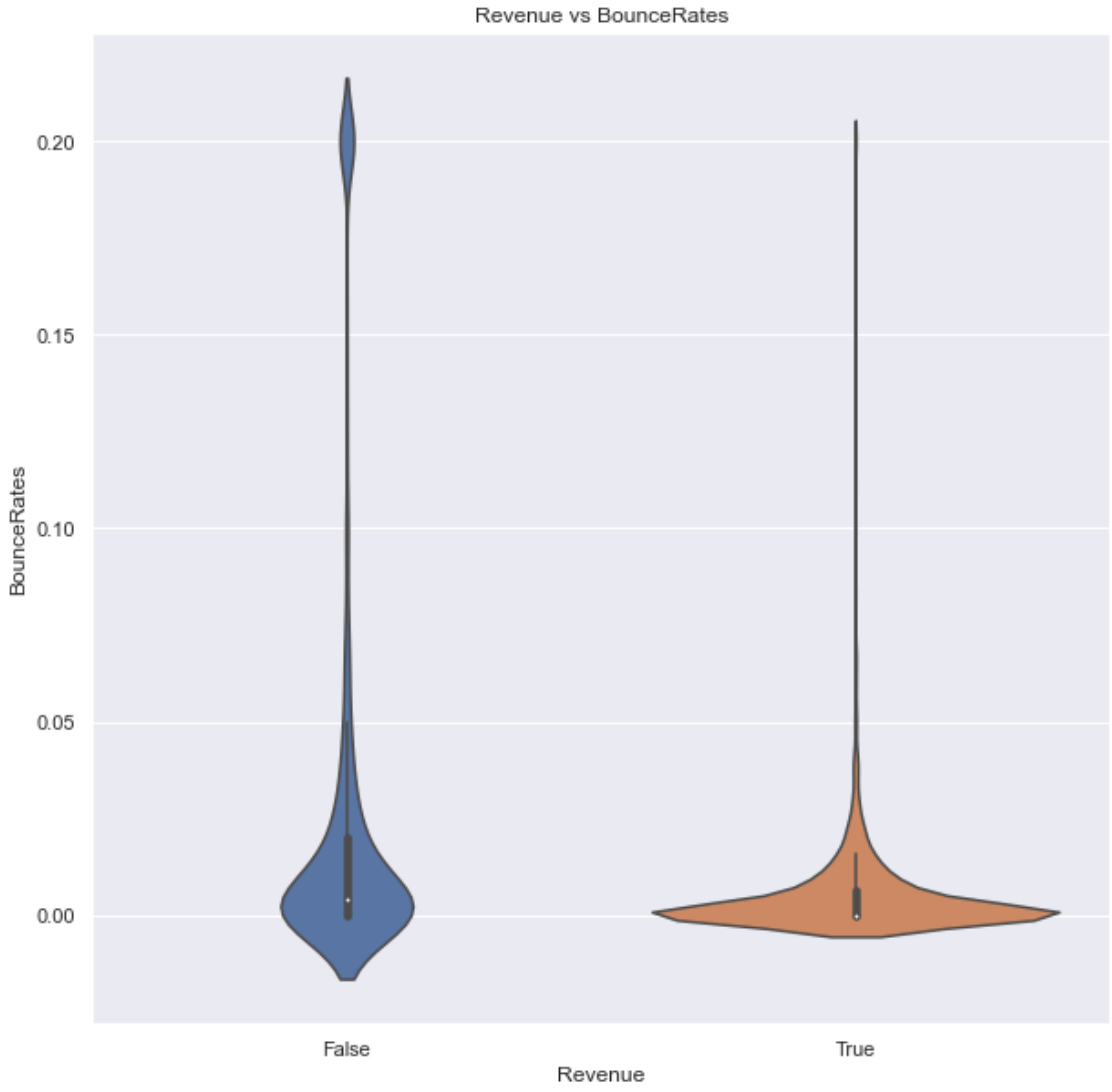
[207]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['Revenue'], y = df['ExitRates'])
plt.title('Revenue vs ExitRates')
```

[207]: Text(0.5, 1.0, 'Revenue vs ExitRates')



[208]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['Revenue'], y = df['BounceRates'])
plt.title('Revenue vs BounceRates')
```

[208]: Text(0.5, 1.0, 'Revenue vs BounceRates')

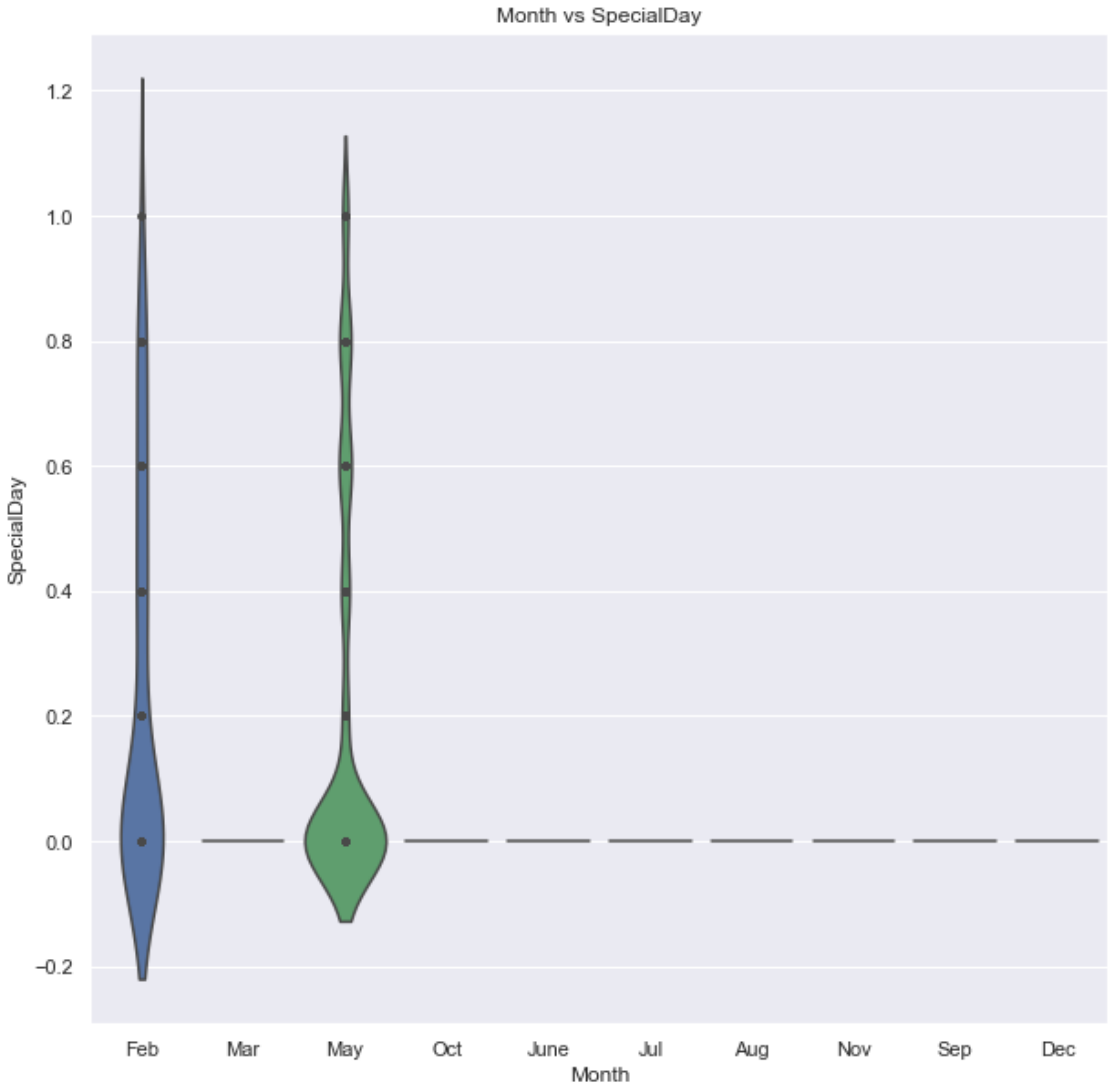


Dla ExitRates i BounceRates dla Revenue = False widać duża liczbę obserwacji o okolicy 2 i zaóważaliny wzrost w okolicy 0.20

1.11 Month vs SpecialDay

[209]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['Month'], y = df['SpecialDay'], inner = 'points')
plt.title('Month vs SpecialDay')
```

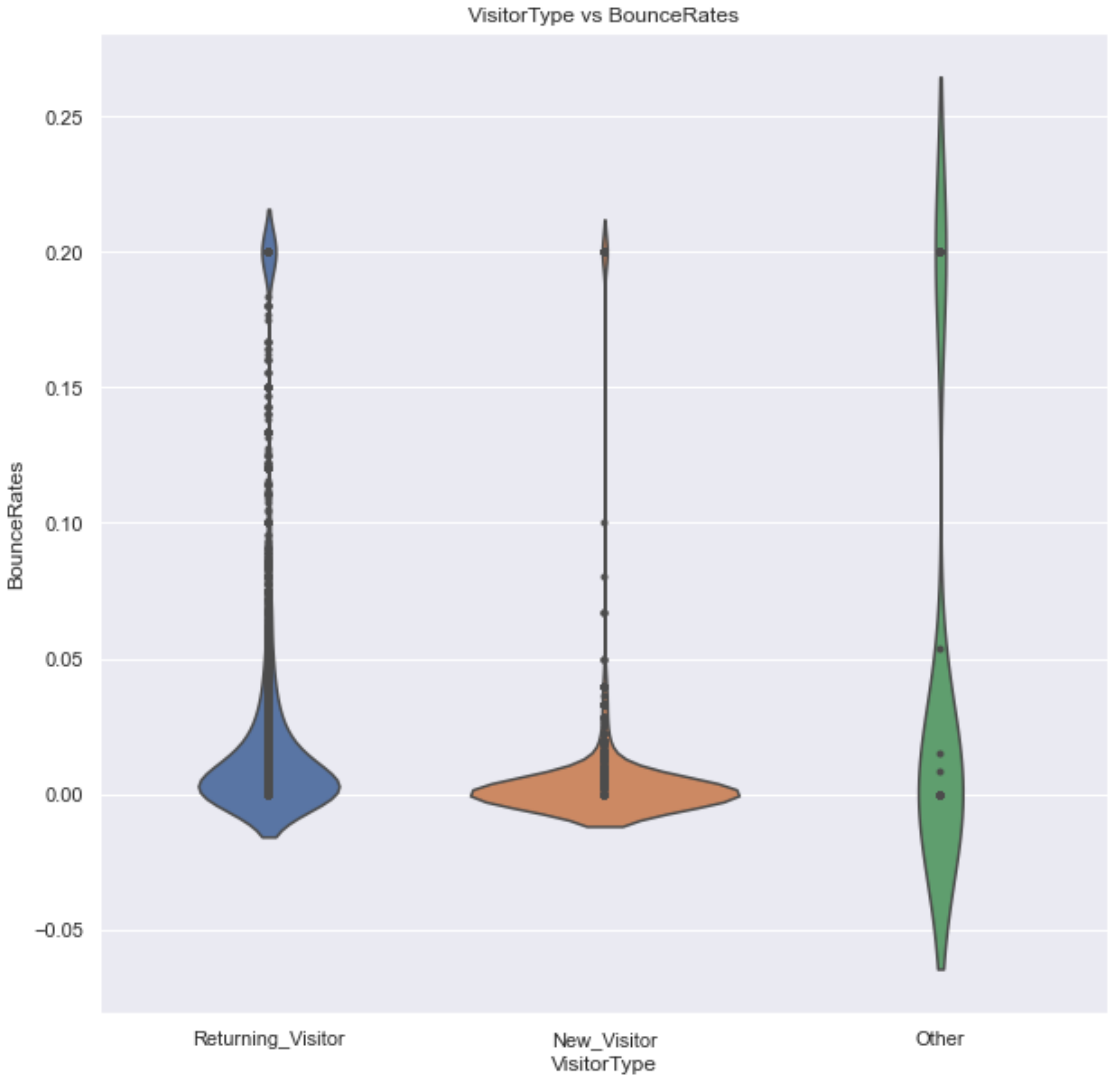
[209]:
```
Text(0.5, 1.0, 'Month vs SpecialDay')
```



1.12 VisitorType

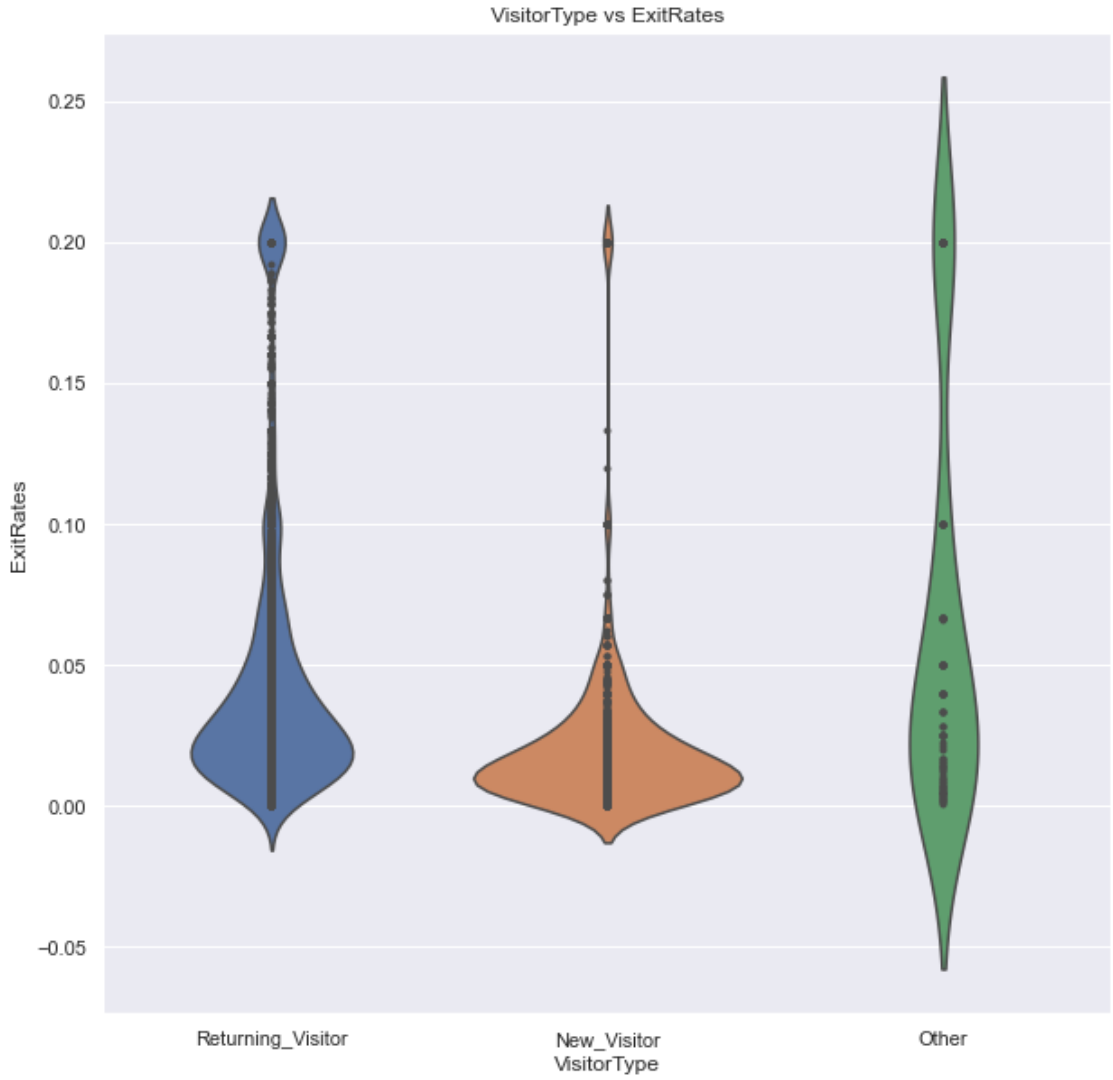
[210]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['VisitorType'], y = df['BounceRates'], inner = 'points')
plt.title('VisitorType vs BounceRates')
```

[210]: Text(0.5, 1.0, 'VisitorType vs BounceRates')



[211]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['VisitorType'], y = df['ExitRates'], inner = 'points')
plt.title('VisitorType vs ExitRates')
```

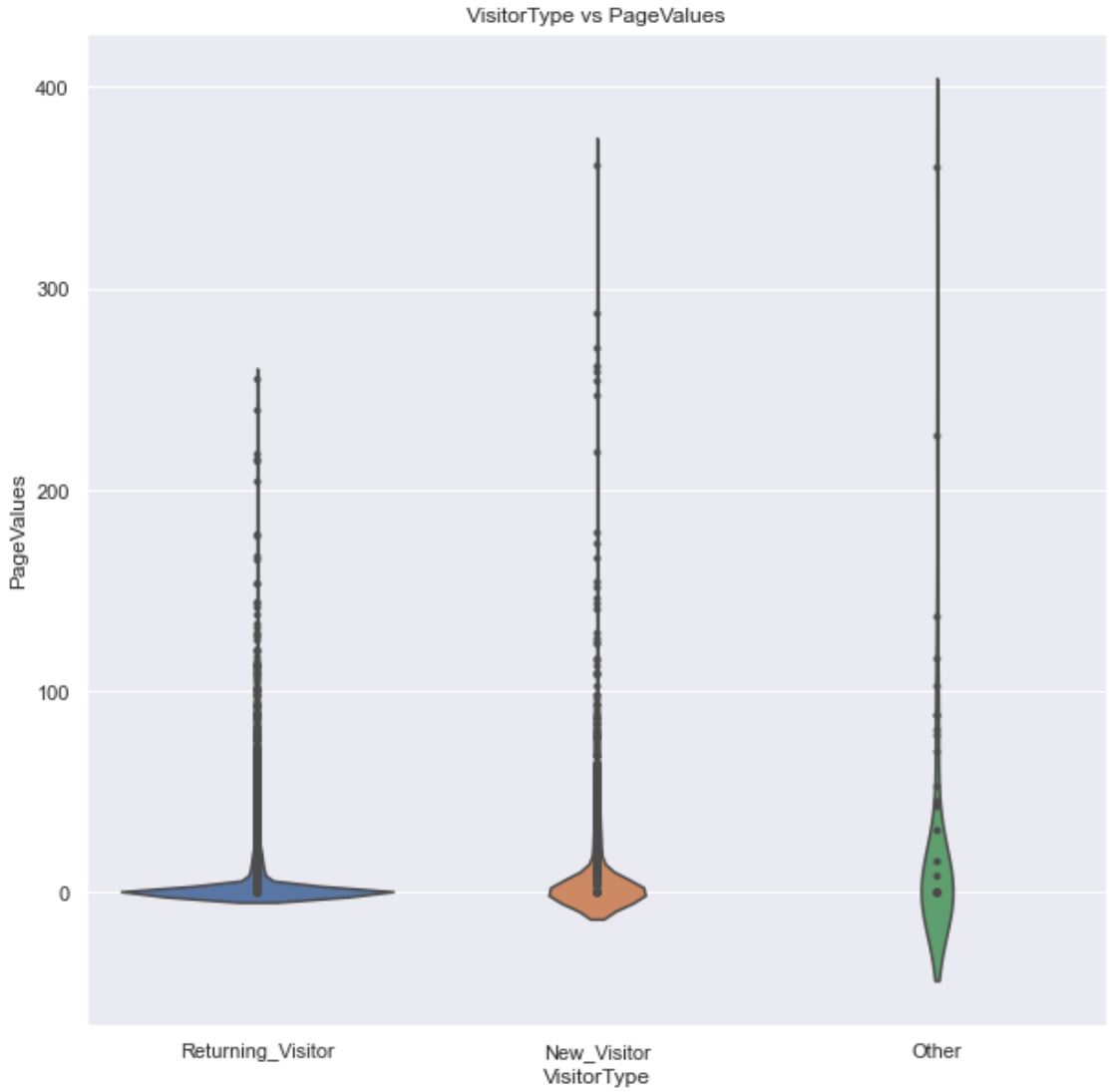
[211]: Text(0.5, 1.0, 'VisitorType vs ExitRates')



New_Visitor dla ExitRates i BounceRates ma dużo obserwacji w okolicy 0. Dla Returning_Visitors instnieje wiele obcerwacji z wartościami pomiędzy 0 a 0.20.

[212]:
```
plt.figure(figsize=(10, 10))
sns.violinplot(x = df['VisitorType'], y = df['PageValues'], inner = 'points')
plt.title('VisitorType vs PageValues')
```

[212]:
```
Text(0.5, 1.0, 'VisitorType vs PageValues')
```



Najwyższe PageValues jest dla New_Visitor. tym razem dużo obserwacji bliskich 0 przypada Returning_Visitors