Project of Data Cleaning, K means clustering to identify customers segments(Machine Learning) & PCA component analysis

In [174…

```
#Imports of Libaries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statistics
import sqlite3
import os
from math import ceil
from datetime import datetime
from itertools import product
import warnings
import missingno as mno
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer
from pickle import dump
from sklearn.decomposition import PCA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from yellowbrick.cluster import KElbowVisualizer
from matplotlib import ticker
from yellowbrick.cluster import SilhouetteVisualizer
from yellowbrick.cluster import InterclusterDistance
```

```
%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.decomposition import PCA
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Data Importation and Regulaization

In [ ]:
```
#Importing data from CSV

#df = pd.read_csv('/content/drive/MyDrive/all tutorials/Case1_HotelCustomerSegmentation.csv')
df = pd.read_csv('/content/drive/MyDrive/all tutorials/Case1_HotelCustomerSegmentation.csv')
# Load data
#df = pd.read_csv('/kaggle/input/wali-3/Case1_HotelCustomerSegmentation.csv', sep=";")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 111733 entries, 0 to 111732
Data columns (total 1 columns):
 #   Column
Non-Null Count   Dtype
---  ------
--------------   -----
 0   ID;Nationality;Age;DaysSinceCreation;NameHash;DocIDHash;AverageLeadTime;LodgingRevenue;OtherRevenue;BookingsCanceled;Bookin
gsNoShowed;BookingsCheckedIn;PersonsNights;RoomNights;DistributionChannel;MarketSegment;SRHighFloor;SRLowFloor;SRAccessibleRoom;
SRMediumFloor;SRBathtub;SRShower;SRCrib;SRKingSizeBed;SRTwinBed;SRNearElevator;SRAwayFromElevator;SRNoAlcoholInMiniBar;SRQuietRo
om  111733 non-null  object
dtypes: object(1)
memory usage: 873.0+ KB
```

In [ ]:
```
#Visualisation of the Inital data shape
```

```
df.shape
```

Out[ ]:
```
(111733, 1)
```

In [ ]:
```
#Visualiasation of the first 5 rows, in order to get a sense of the whats up with the data

df.head()
```

Out[ ]:

	ID;Nationality;Age;DaysSinceCreation;NameHash;DocIDHash;AverageLeadTime;LodgingRevenue;OtherRevenue;BookingsCanceled;BookingsNoShowed;Booking
0	
1	
2	
3	
4	

In [ ]:
```
#Fixing columns problem (iregularites using the ";" as a delimiter to separate data into the intended columns)

#file_path = '/content/drive/MyDrive/all tutorials/Case1_HotelCustomerSegmentation.csv'
file_path = '/content/drive/MyDrive/all tutorials/Case1_HotelCustomerSegmentation.csv'  # Replace with your actual file path
try:
    df = pd.read_csv(file_path, sep=";", engine='python', error_bad_lines=False)
    print(df.head())
except pd.errors.ParserError as e:
    print(f"Error parsing file: {e}")
```

```
<ipython-input-11-413b150fe321>:6: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a futu
re version. Use on_bad_lines in the future.


  df = pd.read_csv(file_path, sep=";", engine='python', error_bad_lines=False)
```

```
   ID Nationality   Age  DaysSinceCreation  \
0   1         PRT  52.0                440
1   2         PRT   NaN               1385
2   3         DEU  32.0               1385
3   4         FRA  61.0               1385
4   5         FRA  52.0               1385

                                        NameHash  \
0  0x2C371FD6CE12936774A139FD7430C624F1C4D5109CE6...
1  0x198CDB98BF37B6E23F9548C56A88B00912D65A9AA0D6...
2  0xDA46E62F66936284DF2844EC4FC542D0DAD780C0EE0C...
3  0xC45D4CD22C58FDC5FD0F95315F6EFA5A6E7149187D49...
4  0xD2E3D5BFCA141865669F98D64CDA85AD04DEFF47F8A0...

                                       DocIDHash  AverageLeadTime  \
0  0x434FD3D59469C73AFEA087017FAF8CA2296493AEABDE...               59
1  0xE3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B...               61
2  0x27F5DF762CCDA622C752CCDA45794923BED9F1B66300...                0
3  0x8E59572913BB9B1E6CAA12FA2C8B7BF387B1D1F3432E...               93
4  0x42BDEE0E05A9441C94147076EDDCC47E604DA5447DD4...                0

   LodgingRevenue  OtherRevenue  BookingsCanceled  ...  SRMediumFloor  \
0           292.0          82.3                 1  ...              0
1           280.0          53.0                 0  ...              0
2             0.0           0.0                 0  ...              0
3           240.0          60.0                 0  ...              0
4             0.0           0.0                 0  ...              0

   SRBathtub  SRShower  SRCrib SRKingSizeBed SRTwinBed  SRNearElevator  \
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

   SRAwayFromElevator  SRNoAlcoholInMiniBar  SRQuietRoom
0                   0                     0            0
1                   0                     0            0
2                   0                     0            0
3                   0                     0            0
4                   0                     0            0

[5 rows x 29 columns]
```

In [ ]:
```
#reassign the column names

column_names = [
    "ID", "Nationality", "Age", "DaysSinceCreation", "NameHash", "DocIDHash",
    "AverageLeadTime", "LodgingRevenue", "OtherRevenue", "BookingsCanceled",
    "BookingsNoShowed", "BookingsCheckedIn", "PersonsNights", "RoomNights",
    "DistributionChannel", "MarketSegment", "SRHighFloor", "SRLowFloor",
    "SRAccessibleRoom", "SRMediumFloor", "SRBathtub", "SRShower", "SRCrib",
    "SRKingSizeBed", "SRTwinBed", "SRNearElevator", "SRAwayFromElevator",
    "SRNoAlcoholInMiniBar", "SRQuietRoom"
]
df.columns = column_names
```

In [ ]:
```
#checking the data again to see it has been fixed

df.head()
```

Out[ ]:

	ID	Nationality	Age	DaysSinceCreation	NameHash	DocIDHash
0	1	PRT	52.0	440	0x2C371FD6CE12936774A139FD7430C624F1C4D5109CE6...	0x434FD3D59469C73AFEA087017FAF8CA2296493AEABDE...
1	2	PRT	NaN	1385	0x198CDB98BF37B6E23F9548C56A88B00912D65A9AA0D6...	0xE3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B...
2	3	DEU	32.0	1385	0xDA46E62F66936284DF2844EC4FC542D0DAD780C0EE0C...	0x27F5DF762CCDA622C752CCDA45794923BED9F1B66300...
3	4	FRA	61.0	1385	0xC45D4CD22C58FDC5FD0F95315F6EFA5A6E7149187D49...	0x8E59572913BB9B1E6CAA12FA2C8B7BF387B1D1F3432E...
4	5	FRA	52.0	1385	0xD2E3D5BFCA141865669F98D64CDA85AD04DEFF47F8A0...	0x42BDEE0E05A9441C94147076EDDCC47E604DA5447DD4...

5 rows × 29 columns

Data Preprocessing

Exploration

In [ ]:
```
df.describe
```

Out[ ]:
pandas.core.generic.NDFrame.describe

```
def describe(percentiles=None, include=None, exclude=None, datetime_is_numeric: bool_t=False) -> NDF
rameT

Generate descriptive statistics.

Descriptive statistics include those that summarize the central
tendency, dispersion and shape of a
dataset's distribution, excluding ``NaN`` values.
```

Metric and Non-Metric Features

In [ ]:
```
df.dtypes
```

Out[ ]:
```
ID                        int64
Nationality              object
Age                     float64
DaysSinceCreation         int64
NameHash                 object
DocIDHash                object
AverageLeadTime           int64
LodgingRevenue          float64
OtherRevenue            float64
BookingsCanceled          int64
BookingsNoShowed          int64
BookingsCheckedIn         int64
PersonsNights             int64
RoomNights                int64
DistributionChannel      object
MarketSegment            object
SRHighFloor               int64
SRLowFloor                int64
SRAccessibleRoom          int64
SRMediumFloor             int64
SRBathtub                 int64
SRShower                  int64
SRCrib                    int64
SRKingSizeBed             int64
SRTwinBed                 int64
SRNearElevator            int64
SRAwayFromElevator        int64
SRNoAlcoholInMiniBar      int64
SRQuietRoom               int64
dtype: object
```

In [ ]:
```
#To further explore the data we will divide the data to numeric and non numeric to further explore the data.

#numeric columns
#numeric_df = df.select_dtypes(include=[np.number])

#non-numeric (object in this case) columns
#non_numeric_df = df.select_dtypes(exclude=[np.number])
```

In [ ]:
```
# Metric (Numeric) features
metric_features = [
    "Age", "DaysSinceCreation", "AverageLeadTime", "LodgingRevenue",
    "OtherRevenue", "BookingsCanceled", "BookingsNoShowed", "BookingsCheckedIn",
```

```
    "PersonsNights", "RoomNights"
]

# Non-metric (Non-numeric) features
non_metric_features = [
    "ID", "Nationality", "NameHash", "DocIDHash", "DistributionChannel",
    "MarketSegment", "SRHighFloor", "SRLowFloor", "SRAccessibleRoom",
    "SRMediumFloor", "SRBathtub", "SRShower", "SRCrib", "SRKingSizeBed",
    "SRTwinBed", "SRNearElevator", "SRAwayFromElevator", "SRNoAlcoholInMiniBar",
    "SRQuietRoom"
]

# Now, you can select these features directly from your DataFrame
numeric_df = df[metric_features]
non_numeric_df = df[non_metric_features]
```

In [ ]:
```
numeric_df.head(5)
```

Out[ ]:

	Age	DaysSinceCreation	AverageLeadTime	LodgingRevenue	OtherRevenue	BookingsCanceled	BookingsNoShowed	BookingsCheckedIn	PersonsNight
0	52.0	440	59	292.0	82.3	1	0	2	
1	NaN	1385	61	280.0	53.0	0	0	1	1
2	32.0	1385	0	0.0	0.0	0	0	0	
3	61.0	1385	93	240.0	60.0	0	0	1	1
4	52.0	1385	0	0.0	0.0	0	0	0	

In [ ]:
```
non_numeric_df.head(5)
```

Out[ ]:

	ID	Nationality	NameHash	DocIDHash	DistributionChannel	Ma
0	1	PRT	0x2C371FD6CE12936774A139FD7430C624F1C4D5109CE6...	0x434FD3D59469C73AFEA087017FAF8CA2296493AEABDE...	Corporate	
1	2	PRT	0x198CDB98BF37B6E23F9548C56A88B00912D65A9AA0D6...	0xE3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B...	Travel Agent/Operator	A
2	3	DEU	0xDA46E62F66936284DF2844EC4FC542D0DAD780C0EE0C...	0x27F5DF762CCDA622C752CCDA45794923BED9F1B66300...	Travel Agent/Operator	A
3	4	FRA	0xC45D4CD22C58FDC5FD0F95315F6EFA5A6E7149187D49...	0x8E59572913BB9B1E6CAA12FA2C8B7BF387B1D1F3432E...	Travel Agent/Operator	A
4	5	FRA	0xD2E3D5BFCA141865669F98D64CDA85AD04DEFF47F8A0...	0x42BDEE0E05A9441C94147076EDDCC47E604DA5447DD4...	Travel Agent/Operator	A

Missing Values

In [ ]:
```
#Turning "?" into nan which is what python identifies missing values
Oddmissingvalues = ["?","$","%","!","*","+","_","@","€"," ","{"]
df.replace(Oddmissingvalues, np.nan, inplace = True)
```

In [ ]:
```
#Here we are specifically looking for all the Missing values in the native for of Nan

pd.set_option('display.max_rows', 300)
df.isna().sum()
```

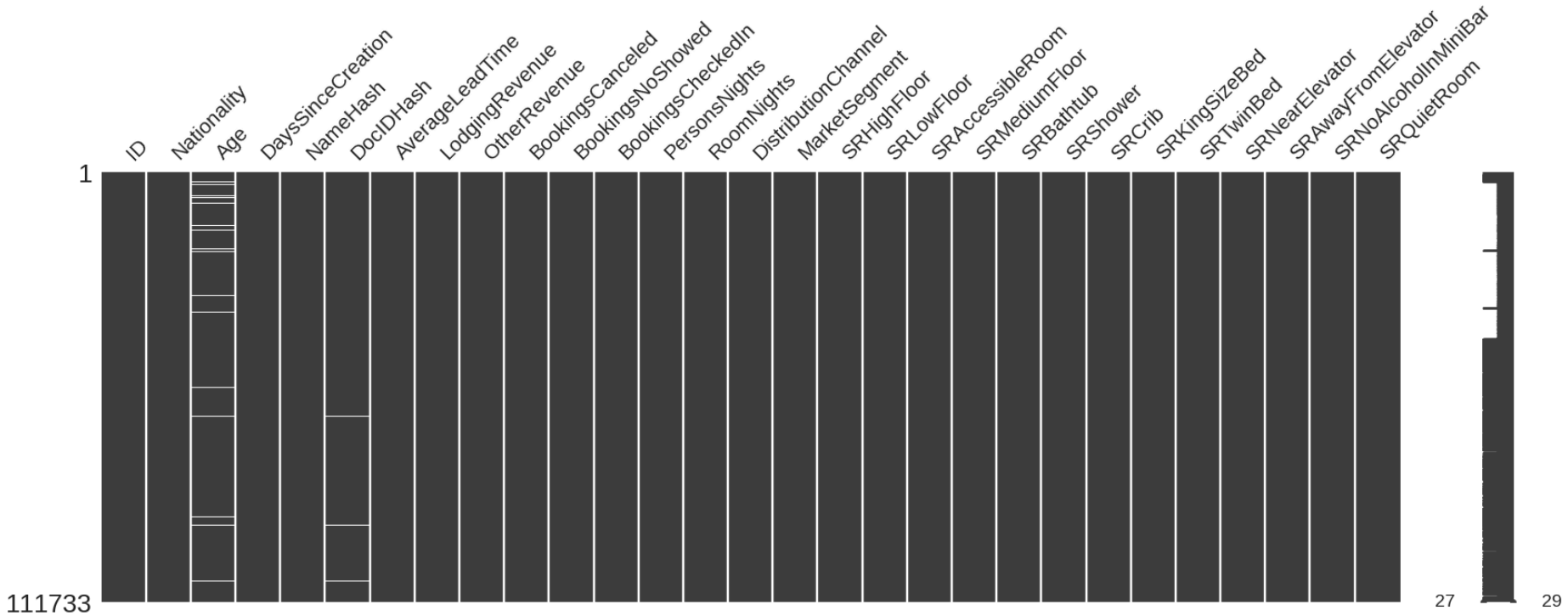
Out[ ]:
```
ID                          0
Nationality                 0
Age                      4172
DaysSinceCreation           0
NameHash                    0
DocIDHash                1001
AverageLeadTime             0
LodgingRevenue              0
OtherRevenue                0
BookingsCanceled            0
BookingsNoShowed            0
BookingsCheckedIn           0
PersonsNights               0
RoomNights                  0
DistributionChannel         0
MarketSegment               0
SRHighFloor                 0
SRLowFloor                  0
SRAccessibleRoom            0
SRMediumFloor               0
SRBathtub                   0
SRShower                    0
SRCrib                      0
SRKingSizeBed               0
SRTwinBed                   0
SRNearElevator              0
SRAwayFromElevator          0
SRNoAlcoholInMiniBar        0
SRQuietRoom                 0
dtype: int64
```

In [ ]:
```
#This is just a percentage of the missing values in Nan for to explore the idea of possible deleting or adding mode and how much

df.isna().sum()/len(df)*100
```

Out[ ]:
```
ID                      0.000000
Nationality             0.000000
Age                     3.733901
DaysSinceCreation       0.000000
NameHash                0.000000
DocIDHash               0.895886
AverageLeadTime         0.000000
LodgingRevenue          0.000000
OtherRevenue            0.000000
BookingsCanceled        0.000000
BookingsNoShowed        0.000000
BookingsCheckedIn       0.000000
PersonsNights           0.000000
RoomNights              0.000000
DistributionChannel     0.000000
MarketSegment           0.000000
SRHighFloor             0.000000
SRLowFloor              0.000000
SRAccessibleRoom        0.000000
SRMediumFloor           0.000000
SRBathtub               0.000000
SRShower                0.000000
SRCrib                  0.000000
SRKingSizeBed           0.000000
SRTwinBed               0.000000
SRNearElevator          0.000000
SRAwayFromElevator      0.000000
SRNoAlcoholInMiniBar    0.000000
SRQuietRoom             0.000000
dtype: float64
```

In [ ]:
```
mno.matrix(df, figsize = (20, 6))
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
# To fill in the NAN we will use the KNN Imputer and for that we first need to convert necessary columns to numeric

# Initialize the KNN Imputer
imputer = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean')

# Fit the imputer and transform your data (this will return a numpy array)
imputed_data = imputer.fit_transform(df.select_dtypes(include=[np.number]))

# Convert the imputed numpy array back to a DataFrame
imputed_df = pd.DataFrame(imputed_data, columns=df.select_dtypes(include=[np.number]).columns)

# If you want to update your original dataframe with the imputed values for the numeric columns
df.update(imputed_df)

# Display the first few rows of the dataframe to check the imputed values
print(df.head())
```

```
    ID Nationality   Age  DaysSinceCreation  \
0  1.0         PRT  52.0              440.0
1  2.0         PRT  53.4             1385.0
2  3.0         DEU  32.0             1385.0
3  4.0         FRA  61.0             1385.0
4  5.0         FRA  52.0             1385.0

                                             NameHash  \
0  0x2C371FD6CE12936774A139FD7430C624F1C4D5109CE6...
1  0x198CDB98BF37B6E23F9548C56A88B00912D65A9AA0D6...
2  0xDA46E62F66936284DF2844EC4FC542D0DAD780C0EE0C...
3  0xC45D4CD22C58FDC5FD0F95315F6EFA5A6E7149187D49...
4  0xD2E3D5BFCA141865669F98D64CDA85AD04DEFF47F8A0...

                                          DocIDHash  AverageLeadTime  \
0  0x434FD3D59469C73AFEA087017FAF8CA2296493AEABDE...             59.0
1  0xE3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B...             61.0
2  0x27F5DF762CCDA622C752CCDA45794923BED9F1B66300...              0.0
3  0x8E59572913BB9B1E6CAA12FA2C8B7BF387B1D1F3432E...             93.0
4  0x42BDEE0E05A9441C94147076EDDCC47E604DA5447DD4...              0.0

   LodgingRevenue  OtherRevenue  BookingsCanceled  ...  SRMediumFloor  \
0           292.0          82.3               1.0  ...            0.0
1           280.0          53.0               0.0  ...            0.0
2             0.0           0.0               0.0  ...            0.0
3           240.0          60.0               0.0  ...            0.0
4             0.0           0.0               0.0  ...            0.0

   SRBathtub  SRShower  SRCrib SRKingSizeBed SRTwinBed  SRNearElevator  \
0        0.0       0.0     0.0           0.0       0.0             0.0
1        0.0       0.0     0.0           0.0       0.0             0.0
2        0.0       0.0     0.0           0.0       0.0             0.0
3        0.0       0.0     0.0           0.0       0.0             0.0
4        0.0       0.0     0.0           0.0       0.0             0.0

   SRAwayFromElevator  SRNoAlcoholInMiniBar  SRQuietRoom
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

[5 rows x 29 columns]
```

In [ ]:
```
#Here we are specifically looking for all the Missing values in the native for of Nan

pd.set_option('display.max_rows', 300)
df.isna().sum()
```

Out[ ]:
```
ID                          0
Nationality                 0
Age                         0
DaysSinceCreation           0
NameHash                    0
DocIDHash                1001
AverageLeadTime             0
LodgingRevenue              0
OtherRevenue                0
BookingsCanceled            0
BookingsNoShowed            0
BookingsCheckedIn           0
PersonsNights               0
RoomNights                  0
DistributionChannel         0
MarketSegment               0
SRHighFloor                 0
SRLowFloor                  0
SRAccessibleRoom            0
SRMediumFloor               0
SRBathtub                   0
SRShower                    0
SRCrib                      0
SRKingSizeBed               0
SRTwinBed                   0
SRNearElevator              0
SRAwayFromElevator          0
SRNoAlcoholInMiniBar        0
SRQuietRoom                 0
dtype: int64
```

In [ ]:
```
# Find duplicate 'NameHash' values
duplicates = df[df.duplicated('NameHash', keep=False)].sort_values('NameHash')

# Displaying the duplicates
print(duplicates.count())
```

```
ID                      7228
Nationality             7228
Age                     7228
DaysSinceCreation       7228
NameHash                7228
DocIDHash               6931
AverageLeadTime         7228
LodgingRevenue          7228
OtherRevenue            7228
BookingsCanceled        7228
BookingsNoShowed        7228
BookingsCheckedIn       7228
PersonsNights           7228
RoomNights              7228
DistributionChannel     7228
MarketSegment           7228
SRHighFloor             7228
SRLowFloor              7228
SRAccessibleRoom        7228
SRMediumFloor           7228
SRBathtub               7228
SRShower                7228
SRCrib                  7228
SRKingSizeBed           7228
SRTwinBed               7228
SRNearElevator          7228
SRAwayFromElevator      7228
SRNoAlcoholInMiniBar    7228
SRQuietRoom             7228
dtype: int64
```

In [ ]:

```
#This take the direct (here it doing with age and average lead time) and turns them into numeric so it can be averaged
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')
df['AverageLeadTime'] = pd.to_numeric(df['AverageLeadTime'], errors='coerce')

# Define the aggregation rules for each column, ensuring 'mean' is only used on numeric columns
aggregation_rules = {
    'ID': 'first',
    'Nationality': 'first',
    'Age': 'mean',
    'DaysSinceCreation': 'first',
    'DocIDHash': 'first',
    'AverageLeadTime': 'mean',
    'LodgingRevenue': 'sum',
```

```
    'OtherRevenue': 'sum',
    'BookingsCanceled': 'sum',
    'BookingsNoShowed': 'sum',
    'BookingsCheckedIn': 'sum',
    'PersonsNights': 'sum',
    'RoomNights': 'sum',
    'DistributionChannel': 'first',
    'MarketSegment': 'first',
    # Ensure special request fields are handled appropriately (e.g., 'max' for binary indicators)
    'SRHighFloor': 'max',
    'SRLowFloor': 'max',
    'SRAccessibleRoom': 'max',
    'SRMediumFloor': 'max',
    'SRBathtub': 'max',
    'SRShower': 'max',
    'SRCrib': 'max',
    'SRKingSizeBed': 'max',
    'SRTwinBed': 'max',
    'SRNearElevator': 'max',
    'SRAwayFromElevator': 'max',
    'SRNoAlcoholInMiniBar': 'max',
    'SRQuietRoom': 'max'
}

# Group by 'NameHash' and apply the aggregation rules
df = df.groupby('NameHash').agg(aggregation_rules).reset_index()

# Display the first few rows of the processed DataFrame to verify the results
print(df.head())
```

```
                                                NameHash       ID Nationality  \
0  0x000093906C9FA7A54C937EF1848D4AA5F79F104D8365...  91332.0         BRA
1  0x0000DA6E0C98F2EA5DEE75FE5541A09A2932A0E85102...  26632.0         CHE
2  0x00025D64CF323C2AA58DBC810B2B08664A7CBC28F989...  43208.0         FRA
3  0x000274B2ED25A149BEB70829F52B3EC0B05A2A2FA0E0...  66941.0         AUT
4  0x00040537EF47B28CFD9E3CA98A75D543FB1A29581449...  49247.0         DEU

    Age  DaysSinceCreation                                        DocIDHash  \
0  51.0              205.0  0xFBB48E8A898A0973E8FC147D2AC89307B16EE4D6B618...
1  25.0              908.0  0x7DE0876FB3F78A6C3EFB6BF57A9C61B086C4C7DC183B...
2  39.0              651.0  0x9035FA078B3BE036893E6CD565F1BEDD5DB562D75B68...
3  57.0              435.0  0xF6AE2E9C37E7C9327ED08C50952B0CB5B52FB9610782...
4  42.0              576.0  0x7EB374B852C8D4583929799B91A6AA9E29213FFCF5D8...

   AverageLeadTime  LodgingRevenue  OtherRevenue  BookingsCanceled  ...  \
0             41.0          416.66         271.5               0.0  ...
1             66.0          384.00          58.5               0.0  ...
2             24.0          234.00          28.0               0.0  ...
3              0.0            0.00           0.0               0.0  ...
4              8.0          432.00          21.0               0.0  ...

   SRMediumFloor  SRBathtub  SRShower  SRCrib SRKingSizeBed SRTwinBed  \
0            0.0        0.0       0.0     0.0           1.0       0.0
1            0.0        0.0       0.0     0.0           0.0       0.0
2            0.0        0.0       0.0     0.0           0.0       1.0
3            0.0        0.0       0.0     0.0           0.0       1.0
4            0.0        0.0       0.0     0.0           0.0       0.0

   SRNearElevator  SRAwayFromElevator  SRNoAlcoholInMiniBar  SRQuietRoom
0             0.0                 0.0                   0.0          0.0
1             0.0                 0.0                   0.0          0.0
2             0.0                 0.0                   0.0          1.0
3             0.0                 0.0                   0.0          1.0
4             0.0                 0.0                   0.0          0.0

[5 rows x 29 columns]
```

In [ ]:
```
#Checking missing values again
pd.set_option('display.max_rows', 300)
df.isna().sum()
```

Out[ ]:
```
NameHash                 0
ID                       0
Nationality              0
Age                      0
DaysSinceCreation        0
DocIDHash              767
AverageLeadTime          0
LodgingRevenue           0
OtherRevenue             0
BookingsCanceled         0
BookingsNoShowed         0
BookingsCheckedIn        0
PersonsNights            0
RoomNights               0
DistributionChannel      0
MarketSegment            0
SRHighFloor              0
SRLowFloor               0
SRAccessibleRoom         0
SRMediumFloor            0
SRBathtub                0
SRShower                 0
SRCrib                   0
SRKingSizeBed            0
SRTwinBed                0
SRNearElevator           0
SRAwayFromElevator       0
SRNoAlcoholInMiniBar     0
SRQuietRoom              0
dtype: int64
```

In [ ]:
```
# To fill in the NAN we will use the KNN Imputer and for that we first need to convert necessary columns to numeric

# Initialize the KNN Imputer
imputer = KNNImputer(n_neighbors=5, weights='uniform', metric='nan_euclidean')

# Fit the imputer and transform your data (this will return a numpy array)
imputed_data = imputer.fit_transform(df.select_dtypes(include=[np.number]))

# Convert the imputed numpy array back to a DataFrame
imputed_df = pd.DataFrame(imputed_data, columns=df.select_dtypes(include=[np.number]).columns)

# If you want to update your original dataframe with the imputed values for the numeric columns
df.update(imputed_df)
```

```
# Display the first few rows of the dataframe to check the imputed values
print(df.head())
```

```
                                        NameHash       ID Nationality  \
0  0x000093906C9FA7A54C937EF1848D4AA5F79F104D8365...  91332.0         BRA
1  0x0000DA6E0C98F2EA5DEE75FE5541A09A2932A0E85102...  26632.0         CHE
2  0x00025D64CF323C2AA58DBC810B2B08664A7CBC28F989...  43208.0         FRA
3  0x000274B2ED25A149BEB70829F52B3EC0B05A2A2FA0E0...  66941.0         AUT
4  0x00040537EF47B28CFD9E3CA98A75D543FB1A29581449...  49247.0         DEU

    Age  DaysSinceCreation                                    DocIDHash  \
0  51.0              205.0  0xFBB48E8A898A0973E8FC147D2AC89307B16EE4D6B618...
1  25.0              908.0  0x7DE0876FB3F78A6C3EFB6BF57A9C61B086C4C7DC183B...
2  39.0              651.0  0x9035FA078B3BE036893E6CD565F1BEDD5DB562D75B68...
3  57.0              435.0  0xF6AE2E9C37E7C9327ED08C50952B0CB5B52FB9610782...
4  42.0              576.0  0x7EB374B852C8D4583929799B91A6AA9E29213FFCF5D8...

   AverageLeadTime  LodgingRevenue  OtherRevenue  BookingsCanceled  ...  \
0             41.0          416.66         271.5               0.0  ...
1             66.0          384.00          58.5               0.0  ...
2             24.0          234.00          28.0               0.0  ...
3              0.0            0.00           0.0               0.0  ...
4              8.0          432.00          21.0               0.0  ...

   SRMediumFloor  SRBathtub  SRShower  SRCrib  SRKingSizeBed  SRTwinBed  \
0            0.0        0.0       0.0     0.0            1.0        0.0
1            0.0        0.0       0.0     0.0            0.0        0.0
2            0.0        0.0       0.0     0.0            0.0        1.0
3            0.0        0.0       0.0     0.0            0.0        1.0
4            0.0        0.0       0.0     0.0            0.0        0.0

   SRNearElevator  SRAwayFromElevator  SRNoAlcoholInMiniBar  SRQuietRoom
0             0.0                 0.0                   0.0          0.0
1             0.0                 0.0                   0.0          0.0
2             0.0                 0.0                   0.0          1.0
3             0.0                 0.0                   0.0          1.0
4             0.0                 0.0                   0.0          0.0

[5 rows x 29 columns]
```

In [ ]:
```
#Checking missing values again
pd.set_option('display.max_rows', 300)
df.isna().sum()
```

Out[ ]:
```
NameHash                 0
ID                       0
Nationality              0
Age                      0
DaysSinceCreation        0
DocIDHash              767
AverageLeadTime          0
LodgingRevenue           0
OtherRevenue             0
BookingsCanceled         0
BookingsNoShowed         0
BookingsCheckedIn        0
PersonsNights            0
RoomNights               0
DistributionChannel      0
MarketSegment            0
SRHighFloor              0
SRLowFloor               0
SRAccessibleRoom         0
SRMediumFloor            0
SRBathtub                0
SRShower                 0
SRCrib                   0
SRKingSizeBed            0
SRTwinBed                0
SRNearElevator           0
SRAwayFromElevator       0
SRNoAlcoholInMiniBar     0
SRQuietRoom              0
dtype: int64
```

In [ ]:
```
#Drop the left over NaN and checking missing values again
df = df.dropna(subset=['DocIDHash'])
pd.set_option('display.max_rows', 300)
df.isna().sum()
```

Out[ ]:
```
NameHash                0
ID                      0
Nationality             0
Age                     0
DaysSinceCreation       0
DocIDHash               0
AverageLeadTime         0
LodgingRevenue          0
OtherRevenue            0
BookingsCanceled        0
BookingsNoShowed        0
BookingsCheckedIn       0
PersonsNights           0
RoomNights              0
DistributionChannel     0
MarketSegment           0
SRHighFloor             0
SRLowFloor              0
SRAccessibleRoom        0
SRMediumFloor           0
SRBathtub               0
SRShower                0
SRCrib                  0
SRKingSizeBed           0
SRTwinBed               0
SRNearElevator          0
SRAwayFromElevator      0
SRNoAlcoholInMiniBar    0
SRQuietRoom             0
dtype: int64
```

Visualization

Outliers

In [ ]:
```
main_colour= 'darkcyan'
```

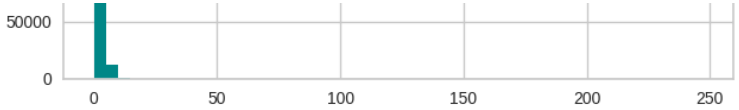
In [ ]:
```
palette= 'BrBG'
```

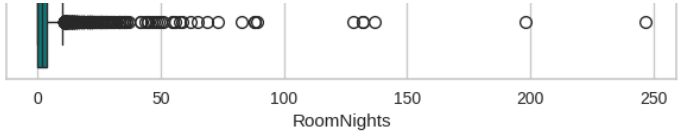
In [ ]:
```
non_metric_features = ['NameHash','DocIDHash',"Nationality","DistributionChannel","MarketSegment","SRHighFloor","SRLowFloor","SRA
metric_features = df.columns.drop(non_metric_features).to_list()
```

In [ ]:
```
def hist_box_maker(df,titl, figx, figy):
    num_of_rows = len(df.columns)
    fig, axes = plt.subplots(num_of_rows, ceil((len(df.columns)*2)/num_of_rows), figsize=(figx, figy))
    temp = (list(df.columns)*2)
    temp.sort()
    # Plot data
    # Iterate across axes objects and associate each histogram:
    i = 0
    for ax, feat in zip(axes.flatten(), temp):
        if i%2 == 0:
            ax.hist(df[feat], bins = 50,color=main_colour)
            ax.set_title(feat,y=0.4,x=2.5)
            pltiswork=feat
        else:
            sns.boxplot(x=df[pltiswork], ax = ax,color=main_colour)
        i+=1
    title = titl
    plt.suptitle(title,y=0.90)
    plt.show()
```

In [ ]:
```
hist_box_maker(df[metric_features],"Outlier Detection",16,16)
```


Outlier Detection
Age
AverageLeadTime
BookingsCanceled
BookingsCheckedIn
BookingsNoShowed
DaysSinceCreation
ID
LodgingRevenue
OtherRevenue
PersonsNights





RoomNights

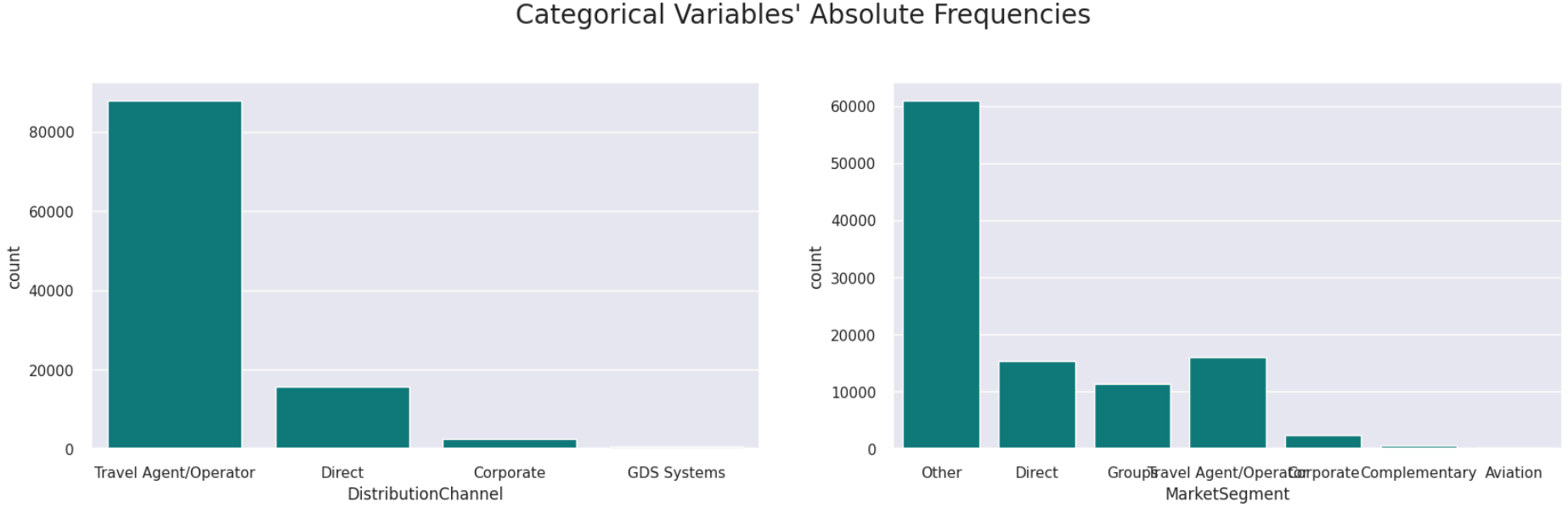
Analysis of the distributions

Analysis of the distributions:

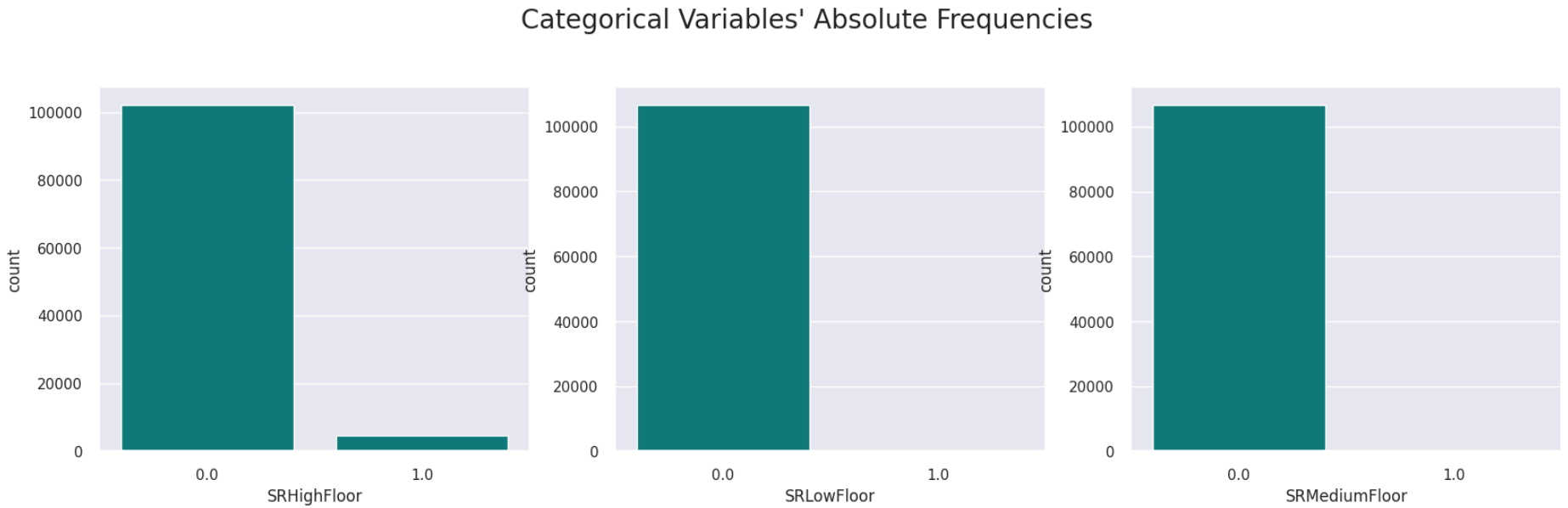
- Age appears to have outliers on the lower and higher side.
- All other variables except the DaysSinceCreation variable seem to have outliers on the higher side

In [ ]:
```
def categ_abs_freq (nr_columns, nr_rows, features):
    sns.set()

    fig, axes = plt.subplots(nr_rows, nr_columns, figsize=(20, 5))


    for ax, feat in zip(axes.flatten(), features):
        sns.countplot(x=df[feat].astype(object), ax=ax, color=main_colour)

    title = "Categorical Variables' Absolute Frequencies"
    plt.suptitle(title, fontsize= '20', y=1.05)

    plt.show()
    return
```

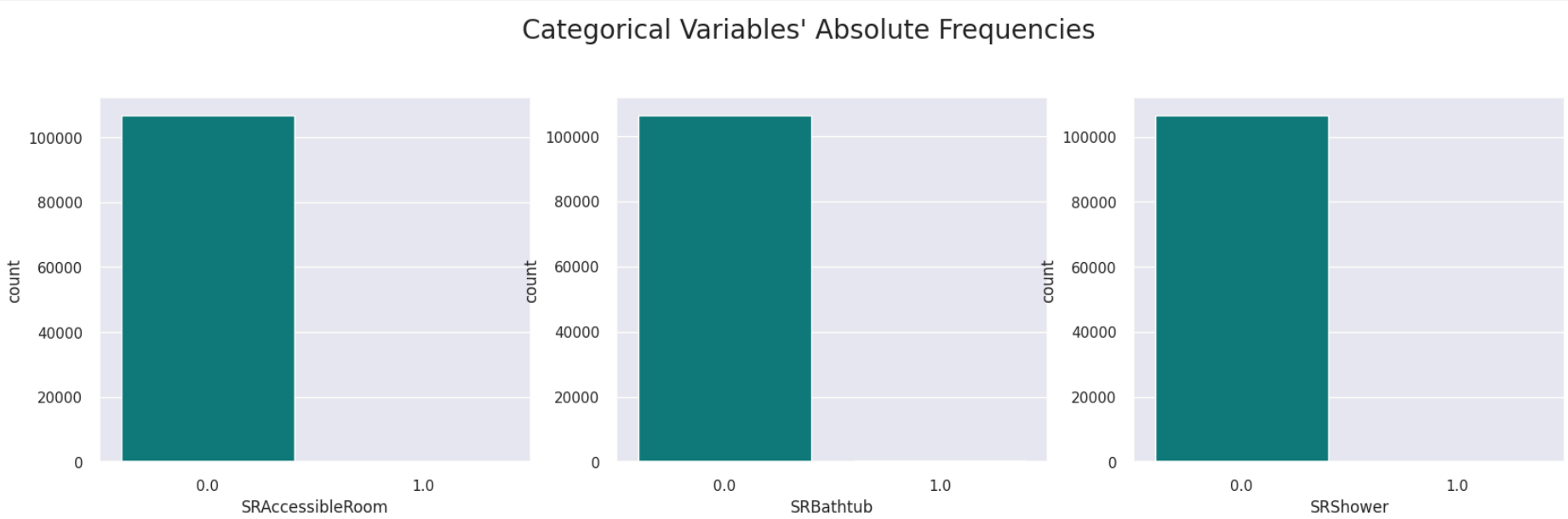
In [ ]:
```
feat_1= ['DistributionChannel','MarketSegment',]
categ_abs_freq(2,1,feat_1)
```



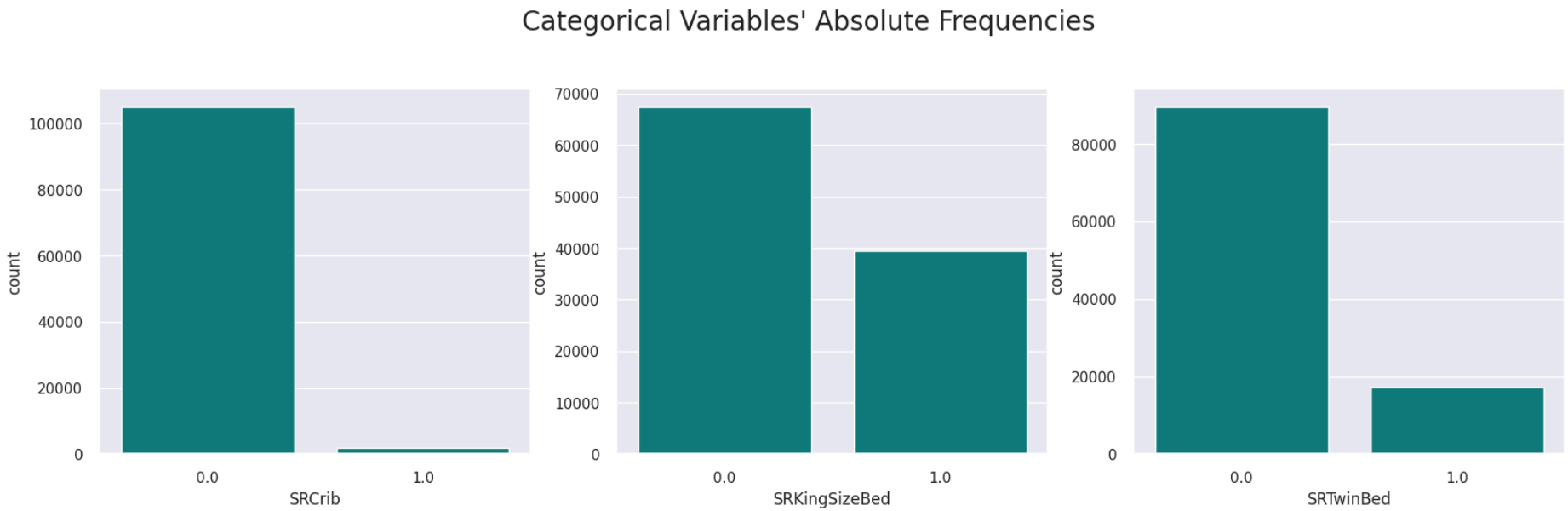
In [ ]:
```
feat_2= ['SRHighFloor','SRLowFloor','SRMediumFloor']
categ_abs_freq(3,1,feat_2)
```



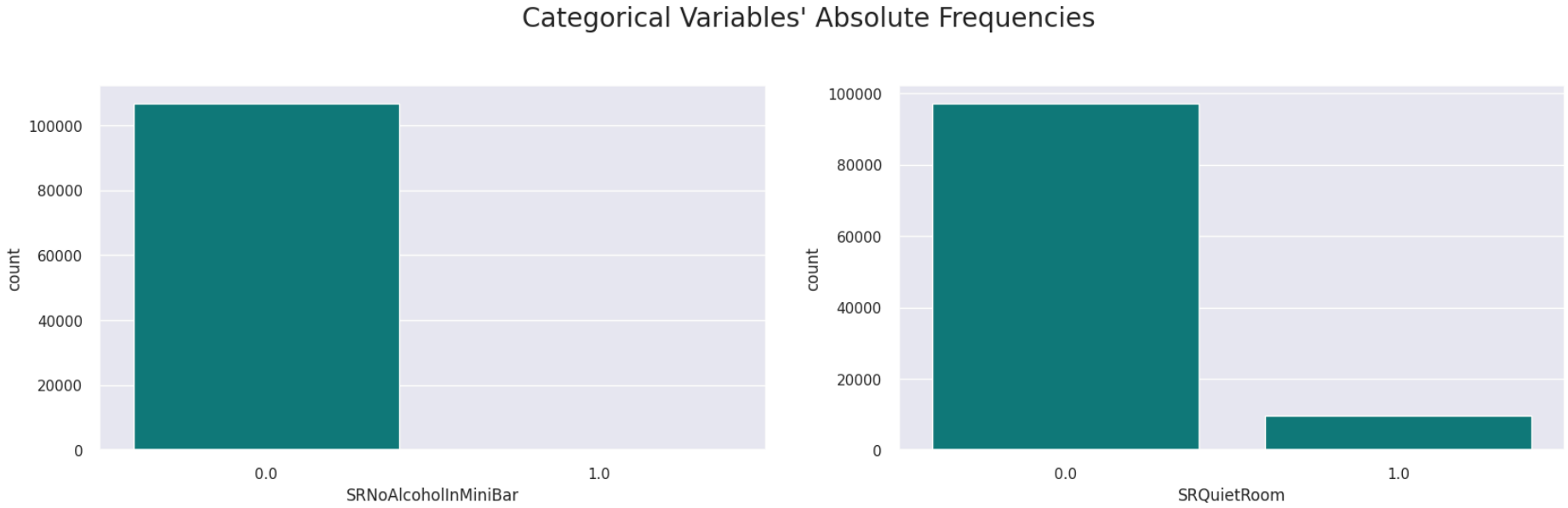
In [ ]:
```
feat_3= ['SRAccessibleRoom','SRBathtub','SRShower']
categ_abs_freq(3,1,feat_3)
```



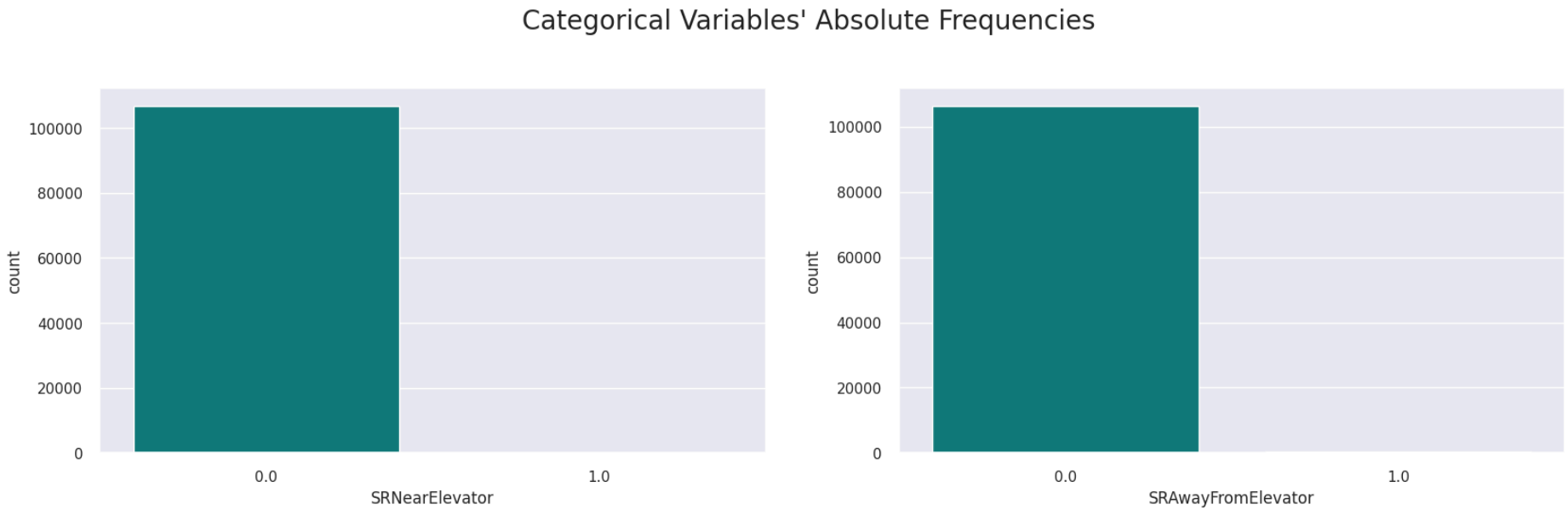
In [ ]:
```
feat_4= ['SRCrib','SRKingSizeBed', 'SRTwinBed']
categ_abs_freq(3,1,feat_4)
```



In [ ]:
```
feat_5= ['SRNoAlcoholInMiniBar','SRQuietRoom']
categ_abs_freq(2,1,feat_5)
```



In [ ]:
```
feat_6= ['SRNearElevator','SRAwayFromElevator']
categ_abs_freq(2,1,feat_6)
```



The main Distribution Channel is distinctly TravelAgent/Operator.

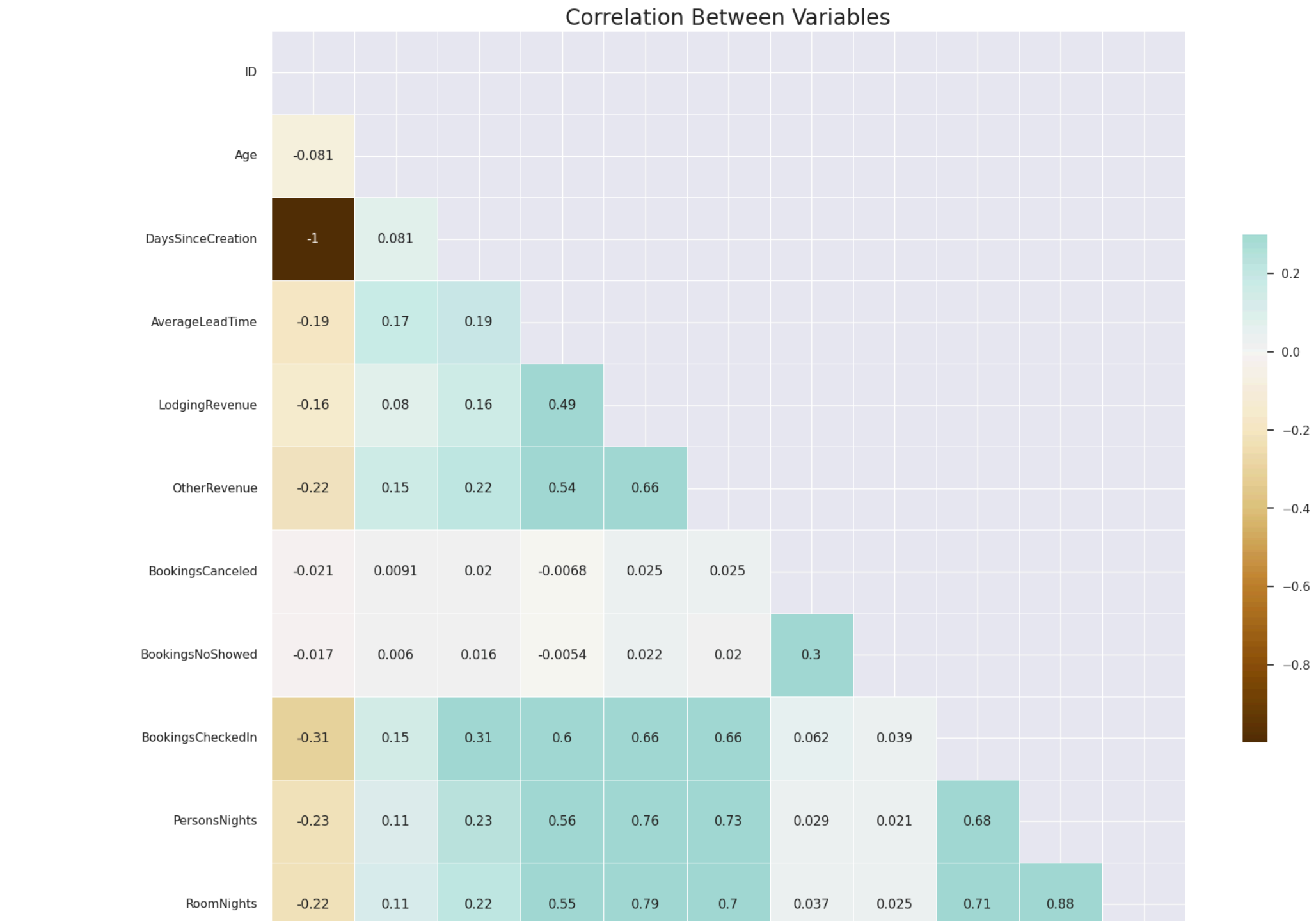
According to the previous Market Segmentation, large part of the customers was classified as Other, followed by Direct and Travel Agent/Operator.

Correlations

In [ ]:
```
#Adapted from https://seaborn.pydata.org/examples/many_pairwise_correlations.html

def correlation_matrix (df, metric_features):

    '''
    Pass a dataframe and the dataframe metric features,
    and plots the correlation matrix (all the correlations
    between the correspondent metric features).

    Arguments:
        df (dataframe): dataframe
```

```
        metric_features (list): df metric features

    '''

    mask = np.triu(np.ones_like(df[metric_features].corr(method='kendall'), dtype=bool))


    # Set up the matplotlib figure
    f, ax = plt.subplots(figsize=(19, 17))


    # Generate a custom diverging colormap
    cmap = sns.diverging_palette(230, 20, as_cmap=True)


    # Draw the heatmap with the mask and correct aspect ratio
    sns.heatmap(df[metric_features].corr(method='kendall'), annot = True, mask=mask, cmap=palette, vmax=.3, center=0,
                square=True, linewidths=.5, cbar_kws={"shrink": .5})

    plt.title('Correlation Between Variables', size=20)

    return
```

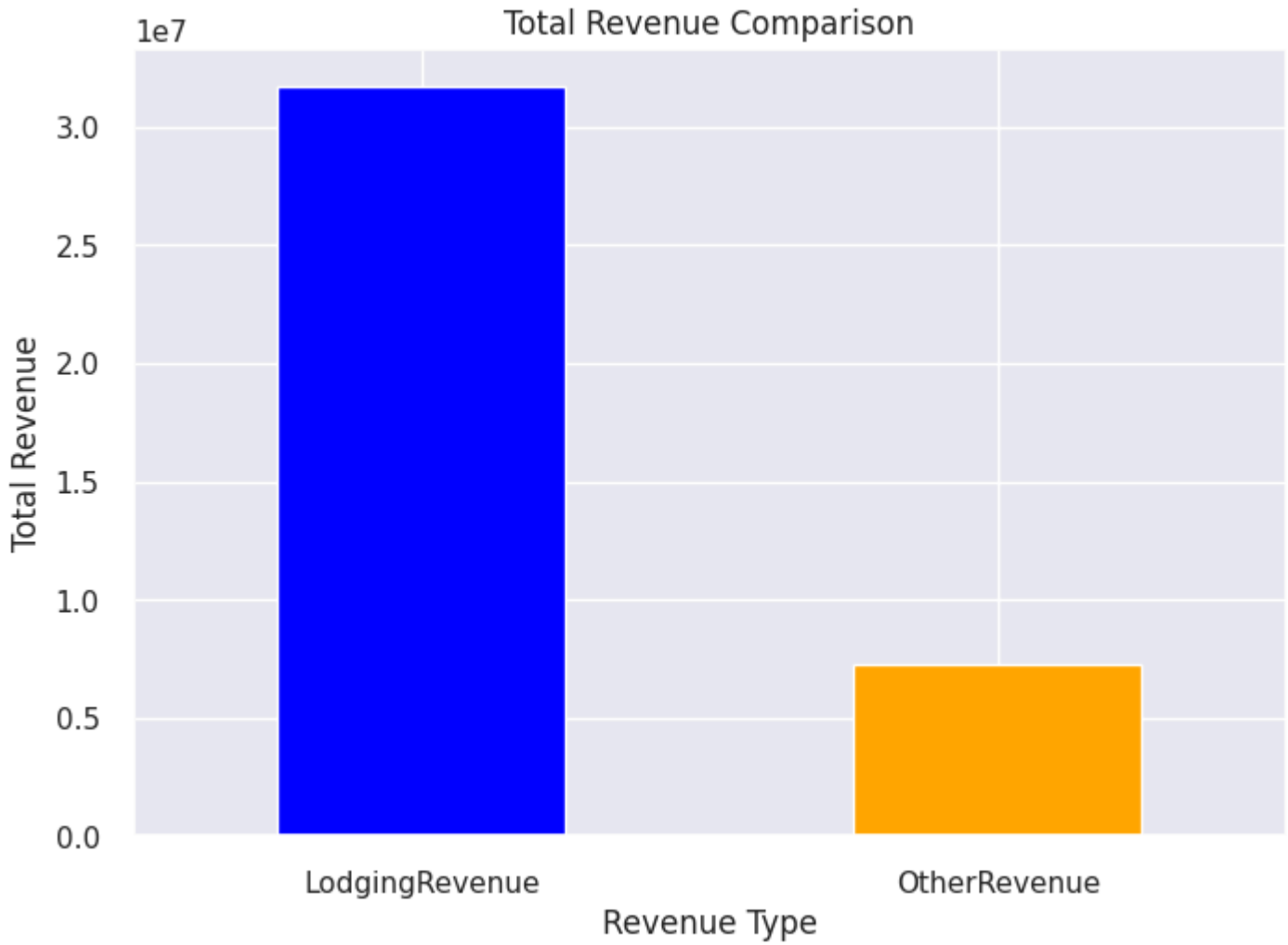
In [ ]:
```
correlation_matrix(df,metric_features)
```


Correlation Between Variables
ID
Age
DaysSinceCreation
AverageLeadTime
LodgingRevenue
OtherRevenue
BookingsCanceled
BookingsNoShowed
BookingsCheckedIn
PersonsNights
RoomNights
-0.081
-1 0.081
-0.19 0.17 0.19
-0.16 0.08 0.16 0.49
-0.22 0.15 0.22 0.54 0.66
-0.021 0.0091 0.02 -0.0068 0.025 0.025
-0.017 0.006 0.016 -0.0054 0.022 0.02 0.3
-0.31 0.15 0.31 0.6 0.66 0.66 0.062 0.039
-0.23 0.11 0.23 0.56 0.76 0.73 0.029 0.021 0.68
-0.22 0.11 0.22 0.55 0.79 0.7 0.037 0.025 0.71 0.88
0.2
0.0
−0.2
−0.4
−0.6
−0.8

In general, customers do not have any preference for the Hotel Floor, regarding room characteristics, bed characteristics, MiniBar and quietness and the distance to the Elevator.

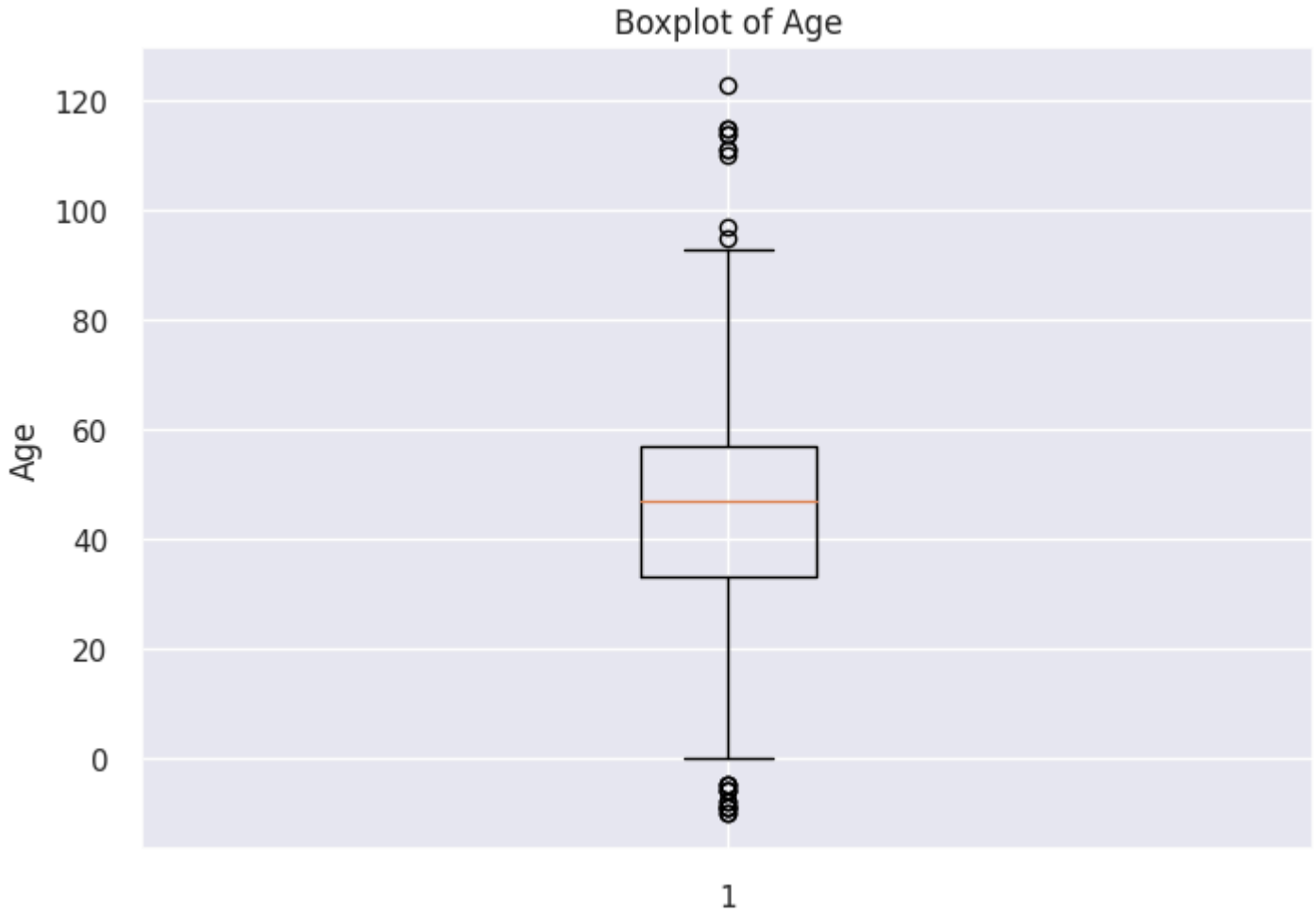
Feature Engineering

In [ ]:
```
#Difference between Ravenue of LodgingRevenue and OtherRevenue

revenue_data = df[['LodgingRevenue', 'OtherRevenue']]

# Plotting the bar chart
revenue_data.sum().plot(kind='bar', color=['blue', 'orange'])
plt.title('Total Revenue Comparison')
plt.xlabel('Revenue Type')
plt.ylabel('Total Revenue')
plt.xticks(rotation=0)  # Rotate x-labels to be horizontal
plt.show()
```

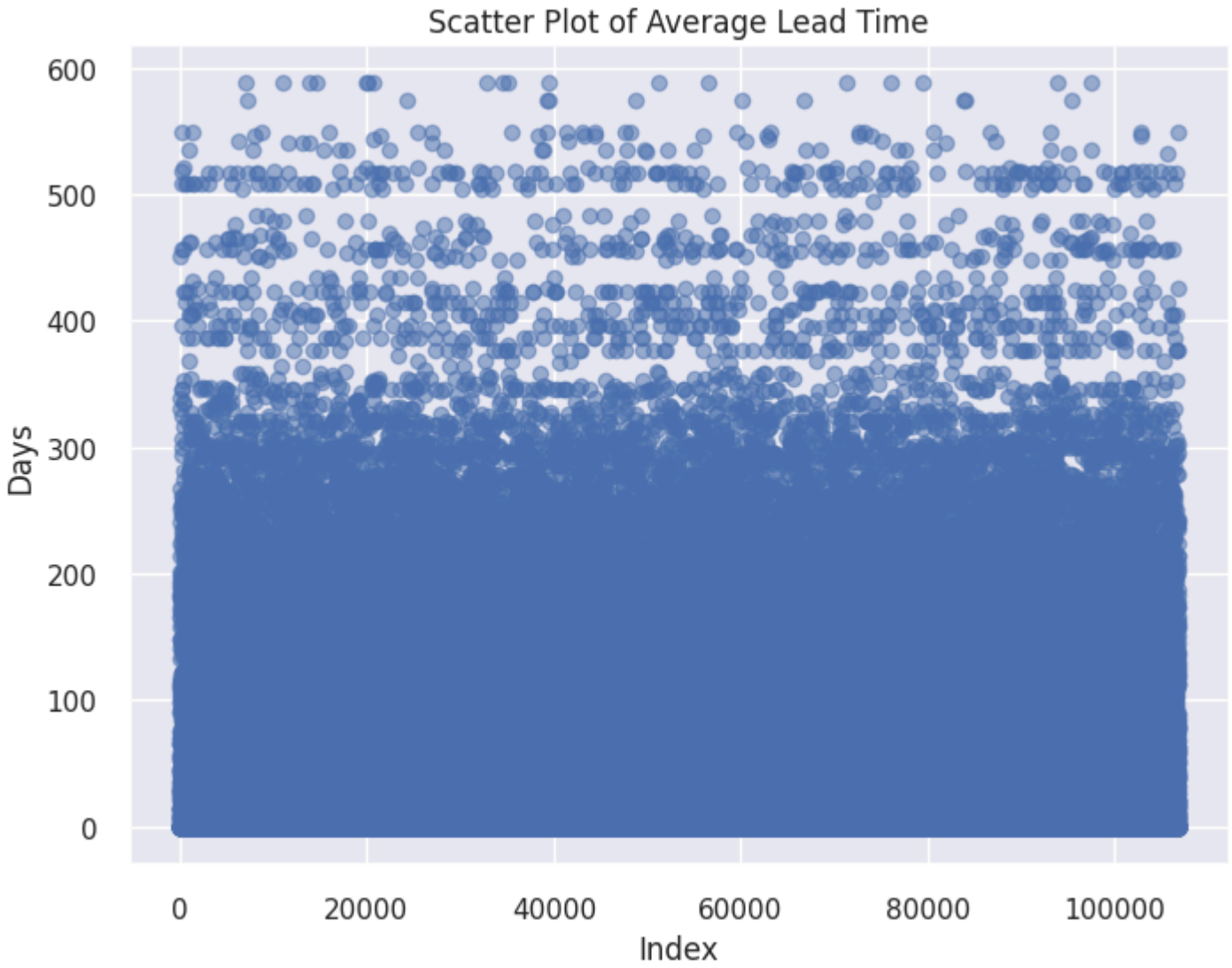


In [ ]:
```
# BOXPLOT For Age column

subPlots_Title_fontSize = 12
subPlots_xAxis_fontSize = 10
subPlots_yAxis_fontSize = 10
subPlots_label_fontSize = 10
heatmaps_text_fontSize = 8

plots_Title_fontSize = 14
plots_Title_textColour = 'black'
```

```
plots_Legend_fontSize = 12
plots_Legend_textColour = 'black'

plots_barTexts_fontSize = 8


plt.boxplot(df['Age'].dropna())  # Remove NaN values and plot the boxplot
plt.title('Boxplot of Age')
plt.ylabel('Age')
plt.show()
```



In [ ]:
```
#AverageLeadTime on Scatter Plot of Average lead time

plt.figure(figsize=(8, 6))  # Adjust the size of the plot if needed
plt.scatter(range(len(df)), df['AverageLeadTime'], alpha=0.5)  # Plotting the scatter plot
plt.title('Scatter Plot of Average Lead Time')
plt.xlabel('Index')
plt.ylabel('Days')
plt.show()
```



In [ ]:
```
selected_columns = df[['BookingsCanceled', 'BookingsNoShowed', 'BookingsCheckedIn']]
print(selected_columns)
```

```
        BookingsCanceled  BookingsNoShowed  BookingsCheckedIn
0                    0.0               0.0                1.0
1                    0.0               0.0                1.0
2                    0.0               0.0                1.0
3                    0.0               0.0                0.0
4                    0.0               0.0                1.0
...                  ...               ...                ...
107579               0.0               0.0                1.0
107580               0.0               0.0                1.0
107581               0.0               0.0                1.0
107582               0.0               0.0                1.0
107583               0.0               0.0                1.0

[106817 rows x 3 columns]
```

In [ ]:
```
# Check the BookingsCheckedin qniqueness
unique_in = df['BookingsCheckedIn'].unique()
print(unique_in)
```

```
[ 1.  0.  2.  3. 10. 13.  9.  4.  5.  6.  7. 51. 20. 11. 16. 15.  8. 33.
 23. 38. 27. 12. 77. 17. 22. 32. 44. 42. 65. 57. 97. 26. 24. 36. 80. 48.
 47. 19. 39. 90.]
```

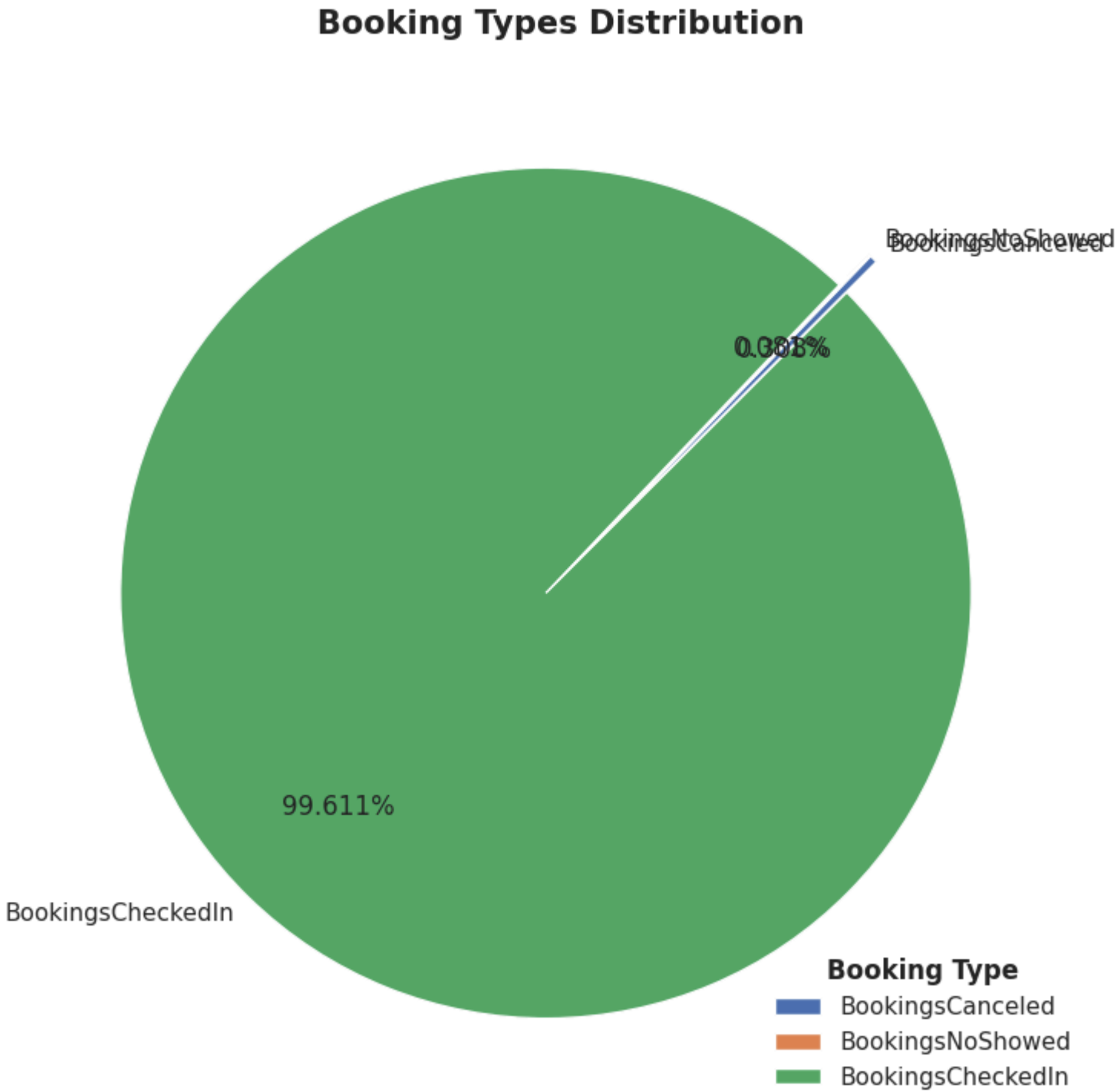
In [ ]:
```
# Check the BookingsCanceled uniqueness
unique_canceled = df['BookingsCanceled'].unique()
print(unique_canceled)
```

```
[ 0.  1. 13.  3.  2. 15.  6.  5.  7. 11.  4.]
```

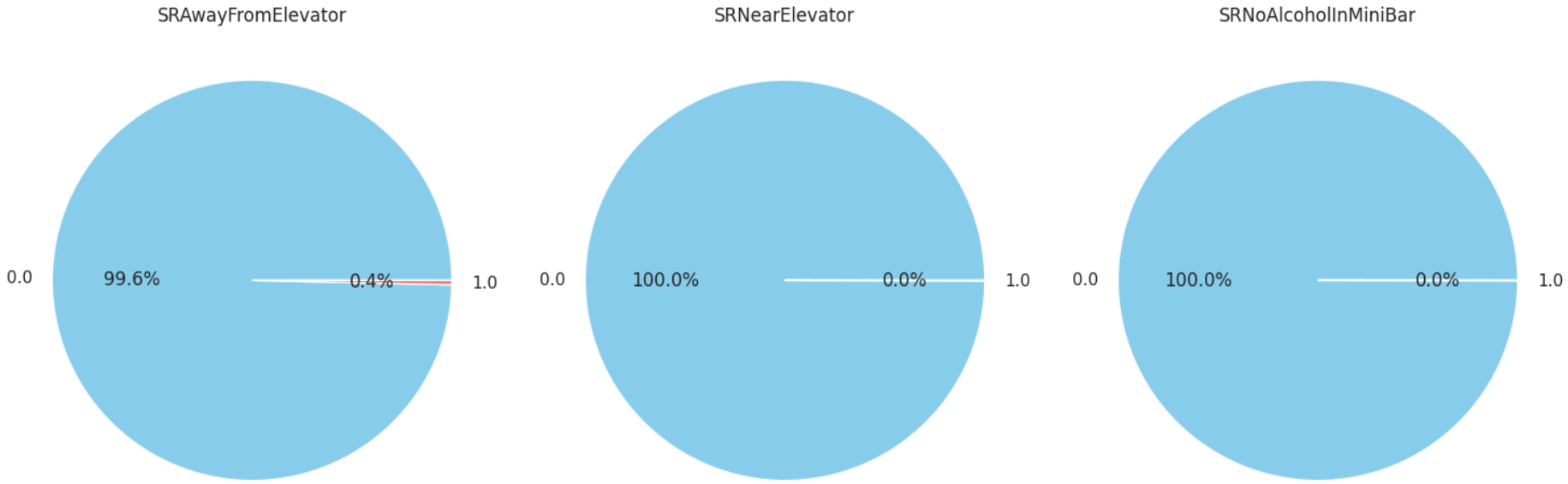
In [ ]:
```
# Check the BookingsNoShowed uniqueness
unique_noshowed = df['BookingsNoShowed'].unique()
print(unique_noshowed)
```

```
[0. 1. 3. 2. 4. 6.]
```

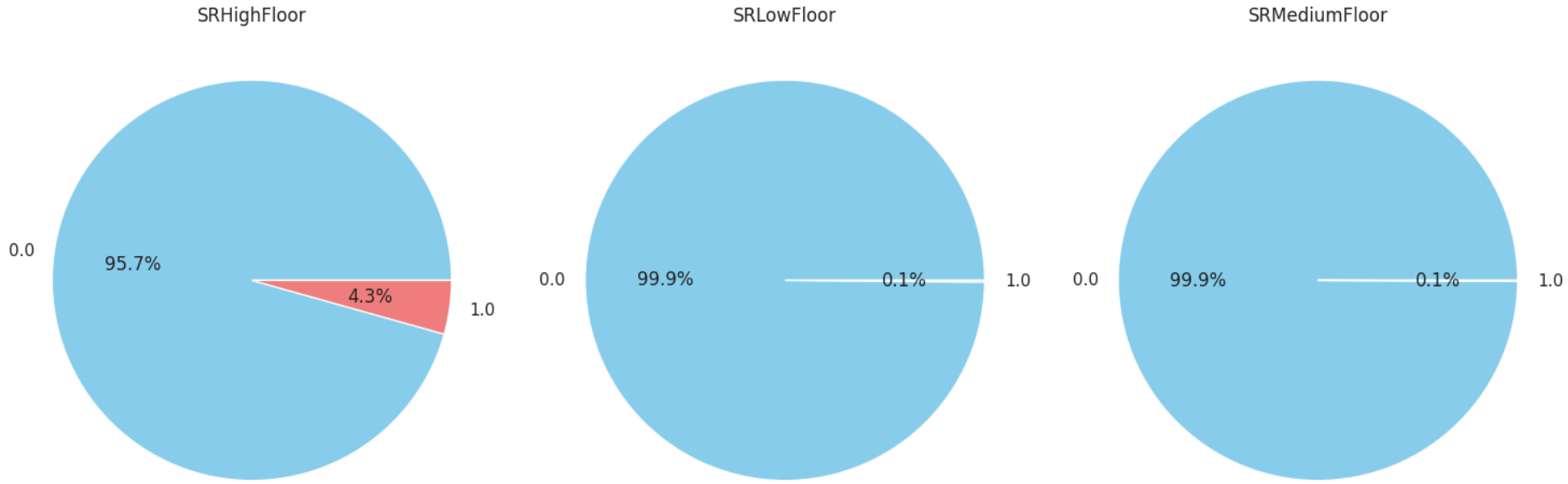
In [ ]:
```
counts = df[['BookingsCanceled', 'BookingsNoShowed', 'BookingsCheckedIn']].sum()

plt.figure(figsize=[9,9])
plt.pie(counts, labels=counts.index, explode=[0.05, 0.05, 0.05], autopct='%0.3f%%', pctdistance=0.7, startangle=45, labeldistance
plt.title('Booking Types Distribution', fontsize=15, fontweight='bold')
legend = plt.legend(loc='lower right', title='Booking Type')
legend.get_title().set_fontweight('bold')
plt.show()
```


Booking Types Distribution
BookingsNoShowed
BookingsCanceled
0.388%
99.611%
BookingsCheckedIn
Booking Type
BookingsCanceled
BookingsNoShowed
BookingsCheckedIn

In [ ]:
```
# Calculate value counts for each column
high_floor_counts = df['SRAwayFromElevator'].value_counts()
low_floor_counts = df['SRNearElevator'].value_counts()
medium_floor_counts = df['SRNoAlcoholInMiniBar'].value_counts()

# Plotting pie chart for 'SRAwayFromElevator'
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.pie(high_floor_counts, labels=high_floor_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title('SRAwayFromElevator')

# Plotting pie chart for 'SRNearElevator'
plt.subplot(1, 3, 2)
plt.pie(low_floor_counts, labels=low_floor_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title('SRNearElevator')

# Plotting pie chart for 'SRNoAlcoholInMiniBar'
plt.subplot(1, 3, 3)
plt.pie(medium_floor_counts, labels=medium_floor_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title('SRNoAlcoholInMiniBar')

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



In [ ]:
```
# Calculate value counts for each column
high_floor_counts = df['SRHighFloor'].value_counts()
low_floor_counts = df['SRLowFloor'].value_counts()
medium_floor_counts = df['SRMediumFloor'].value_counts()

# Plotting pie chart for 'SRHighFloor'
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.pie(high_floor_counts, labels=high_floor_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title('SRHighFloor')

# Plotting pie chart for 'SRLowFloor'
plt.subplot(1, 3, 2)
plt.pie(low_floor_counts, labels=low_floor_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title('SRLowFloor')

# Plotting pie chart for 'SRMediumFloor'
plt.subplot(1, 3, 3)
plt.pie(medium_floor_counts, labels=medium_floor_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral'])
plt.title('SRMediumFloor')

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



In [ ]:
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /usr/local/lib/python3.10/dist-packages (3.1.2)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10/dist-packages (from openpyxl) (1.1.0)
```

In [ ]:
```
df = df[df['Age'] > 0] # Remove negative & null values
df = df[df['AverageLeadTime'] >= 0]

print('The total number of negative values of age is ', len(df[df['Age'] < 0]))
print('The total number of missing values of age column is', df['Age'].isnull().sum())
print('The total number of negative values of averageLeadTime is ', len(df[df['AverageLeadTime'] < 0]))

#df.shape
```

```
The total number of negative values of age is  0
The total number of missing values of age column is 0
The total number of negative values of averageLeadTime is  0
```

In [ ]:
```
# Remove Columns
cols = ['DocIDHash', 'NameHash']
df.drop(cols,axis = 1, inplace = True)

# Change Type
df['ID'] = df['ID'].astype(str)
```

In [ ]:
```
top_15 = df['Nationality'].value_counts().head(15)
fig = px.pie( values = top_15.values,names=top_15.keys(), title='Distributoion of Nationalities')
fig.show()
```

In [ ]:
```
# Remove outliars
df = df[df['Age'] < 100]
# Remove ID column
df.drop('ID', axis = 1, inplace = True)
```

In [ ]:
```
num_of_countries = df['Nationality'].nunique()
print('The number of nationalities before preprocessing is %s' %num_of_countries)
```

```
The number of nationalities before preprocessing is 197
```

In [ ]:
```
# Remove unimportant nationalities
names = df.groupby('Nationality').count().sort_values(by = 'Age', ascending = False)
top_15_nationalities = list(names[names['Age'] < 1000]['Age'].keys())

for i in top_15_nationalities:
    df['Nationality'] = df['Nationality'].str.replace(i,'')

print('The number of nationalities after preprocessing is %s' %df['Nationality'].nunique())
```

```
The number of nationalities after preprocessing is 20
```

In [ ]:
```
# Encoding Variables
transformer = make_column_transformer(
    (OneHotEncoder(sparse=False), ['Nationality', 'MarketSegment', 'DistributionChannel']),
    remainder='passthrough')

transformed = transformer.fit_transform(df)
transformed_df = pd.DataFrame(transformed, columns=transformer.get_feature_names_out())

# Concat the two tables
transformed_df.reset_index(drop=True, inplace=True)
df.reset_index(drop=True, inplace=True)
df2 = pd.concat([transformed_df, df], axis=1)

# Remove old columns
df2.drop(['Nationality', 'MarketSegment', 'DistributionChannel'], axis = 1, inplace = True)
print('The shape after encoding: {}'.format(df2.shape))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning:

`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

The shape after encoding: (106764, 77)
```

In [ ]:
```
df2.head()
```

Out[ ]:

	onehotencoder__Nationality_	onehotencoder__Nationality_AUT	onehotencoder__Nationality_BEL	onehotencoder__Nationality_BRA	onehotencoder__Natio
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	

5 rows × 77 columns

Normalization

In [ ]:
```
# Normalize Data
df_scaled = ((df2 - df2.min()) / (df2.min() + df2.max())) * 9 + 1



#scaler = MinMaxScaler()
#df_scaled = scaler.fit_transform(df2[numeric_df])
#df_scaled
#df_scaler_test = MinMaxScaler()
#df_scaled_feat_test = df_scaler_test.fit_transform(df2[numeric_df])
#df_scaled_feat_test
```

In [ ]:
```
df_scaled.describe()
```

Out[ ]:

	onehotencoder__Nationality_	onehotencoder__Nationality_AUT	onehotencoder__Nationality_BEL	onehotencoder__Nationality_BRA	onehotencoder__N
count	106764.000000	106764.000000	106764.000000	106764.000000	
mean	1.918512	1.159913	1.340227	1.316708	
std	2.724521	1.188975	1.716484	1.658340	
min	1.000000	1.000000	1.000000	1.000000	
25%	1.000000	1.000000	1.000000	1.000000	
50%	1.000000	1.000000	1.000000	1.000000	
75%	1.000000	1.000000	1.000000	1.000000	
max	10.000000	10.000000	10.000000	10.000000	

8 rows × 77 columns

In [ ]:
```
df_scaled = df_scaled.loc[:, ~df_scaled.columns.str.startswith('remainder__')]
df_scaled.dtypes
```

Out[ ]:

onehotencoder__Nationality_	float64
onehotencoder__Nationality_AUT	float64
onehotencoder__Nationality_BEL	float64
onehotencoder__Nationality_BRA	float64
onehotencoder__Nationality_CAN	float64
onehotencoder__Nationality_CHE	float64
onehotencoder__Nationality_CHN	float64
onehotencoder__Nationality_DEU	float64
onehotencoder__Nationality_ESP	float64
onehotencoder__Nationality_FRA	float64
onehotencoder__Nationality_GBR	float64
onehotencoder__Nationality_IRL	float64
onehotencoder__Nationality_ISR	float64
onehotencoder__Nationality_ITA	float64
onehotencoder__Nationality_NLD	float64
onehotencoder__Nationality_NOR	float64
onehotencoder__Nationality_POL	float64
onehotencoder__Nationality_PRT	float64
onehotencoder__Nationality_SWE	float64
onehotencoder__Nationality_USA	float64
onehotencoder__MarketSegment_Aviation	float64
onehotencoder__MarketSegment_Complementary	float64
onehotencoder__MarketSegment_Corporate	float64
onehotencoder__MarketSegment_Direct	float64
onehotencoder__MarketSegment_Groups	float64
onehotencoder__MarketSegment_Other	float64
onehotencoder__MarketSegment_Travel Agent/Operator	float64
onehotencoder__DistributionChannel_Corporate	float64
onehotencoder__DistributionChannel_Direct	float64
onehotencoder__DistributionChannel_GDS Systems	float64
onehotencoder__DistributionChannel_Travel Agent/Operator	float64
Age	float64
DaysSinceCreation	float64
AverageLeadTime	float64
LodgingRevenue	float64
OtherRevenue	float64
BookingsCanceled	float64
BookingsNoShowed	float64
BookingsCheckedIn	float64
PersonsNights	float64
RoomNights	float64
SRHighFloor	float64
SRLowFloor	float64
SRAccessibleRoom	float64

```
SRMediumFloor                float64
SRBathtub                    float64
SRShower                     float64
SRCrib                       float64
SRKingSizeBed                float64
SRTwinBed                    float64
SRNearElevator               float64
SRAwayFromElevator           float64
SRNoAlcoholInMiniBar         float64
SRQuietRoom                  float64
dtype: object
```

Feature Selection

Pearson and Spearman Correlation

In [ ]:
```
enconding_variables =df_scaled.columns[df_scaled.columns.str.startswith('x')]

data_1 = df_scaled.drop(columns=enconding_variables)
data_1
```

Out[ ]:

	onehotencoder__Nationality_	onehotencoder__Nationality_AUT	onehotencoder__Nationality_BEL	onehotencoder__Nationality_BRA	onehotencoder__
0	1.0	1.0	1.0	10.0	
1	1.0	1.0	1.0	1.0	
2	1.0	1.0	1.0	1.0	
3	1.0	10.0	1.0	1.0	
4	1.0	1.0	1.0	1.0	
...	...	...	...	...	
106759	10.0	1.0	1.0	1.0	
106760	1.0	10.0	1.0	1.0	
106761	1.0	1.0	1.0	1.0	
106762	1.0	1.0	1.0	1.0	
106763	1.0	1.0	1.0	1.0	

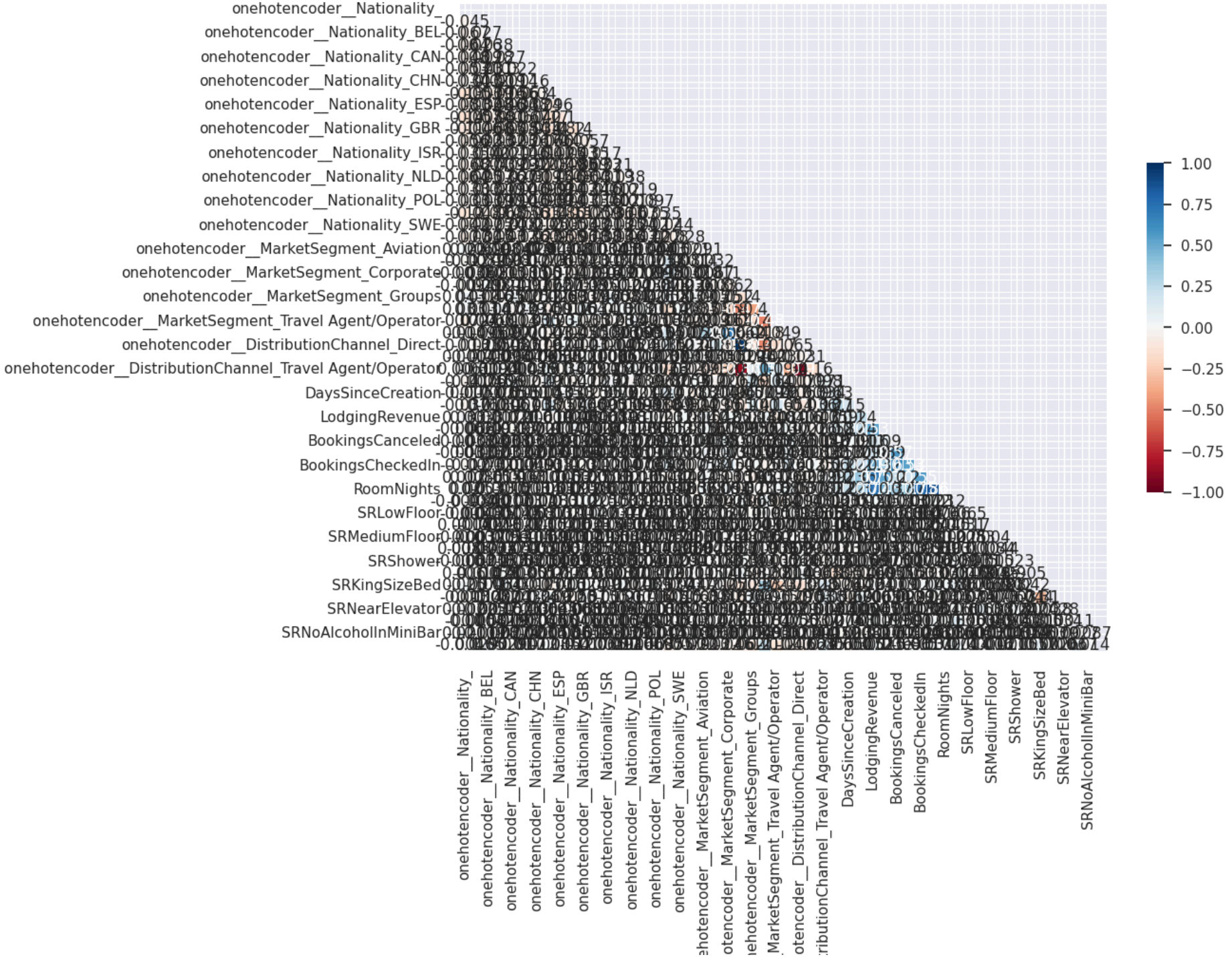
106764 rows × 54 columns

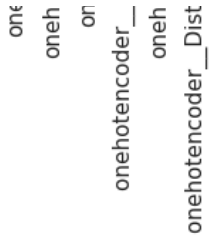
Pearson Correlation

In [ ]:
```
# Compute the correlation matrix - Pearson
corr = data_1.corr(method='pearson')

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(240, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
```

```
sns.heatmap(corr, mask = mask, cmap = 'RdBu', vmax = 1, vmin=-1, center = 0,
            square = True, annot = True, linewidths = .5, cbar_kws = {"shrink": .5})
```

Out[ ]: <Axes: >


onehotencoder__Nationality_
onehotencoder__Nationality_BEL
onehotencoder__Nationality_CAN
onehotencoder__Nationality_CHN
onehotencoder__Nationality_ESP
onehotencoder__Nationality_GBR
onehotencoder__Nationality_ISR
onehotencoder__Nationality_NLD
onehotencoder__Nationality_POL
onehotencoder__Nationality_SWE
onehotencoder__MarketSegment_Aviation
onehotencoder__MarketSegment_Corporate
onehotencoder__MarketSegment_Groups
onehotencoder__MarketSegment_Travel Agent/Operator
onehotencoder__DistributionChannel_Direct
onehotencoder__DistributionChannel_Travel Agent/Operator
DaysSinceCreation
LodgingRevenue
BookingsCanceled
BookingsCheckedIn
RoomNights
SRLowFloor
SRMediumFloor
SRShower
SRKingSizeBed
SRNearElevator
SRNoAlcoholInMiniBar
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



In [ ]:
```
# Identify and print pairs with high correlation
high_correlations = []
for i in range(len(corr)):
    for j in range(i+1, len(corr.columns)):
        # Skip if column names are similar, indicating potential duplicates
        if corr.index[i].replace('remainder__', '') == corr.columns[j].replace('remainder__', ''):
            continue
        # Check if the correlation is above 0.8 or below -0.8
        if corr.iloc[i, j] > 0.8 or corr.iloc[i, j] < -0.8:
            high_correlations.append((corr.index[i], corr.columns[j], corr.iloc[i, j]))

# Print out the high correlations
print("High correlations (Pearson):")
for col_pair in high_correlations:
    print(f"{col_pair[0]} and {col_pair[1]}: {col_pair[2]}")
```

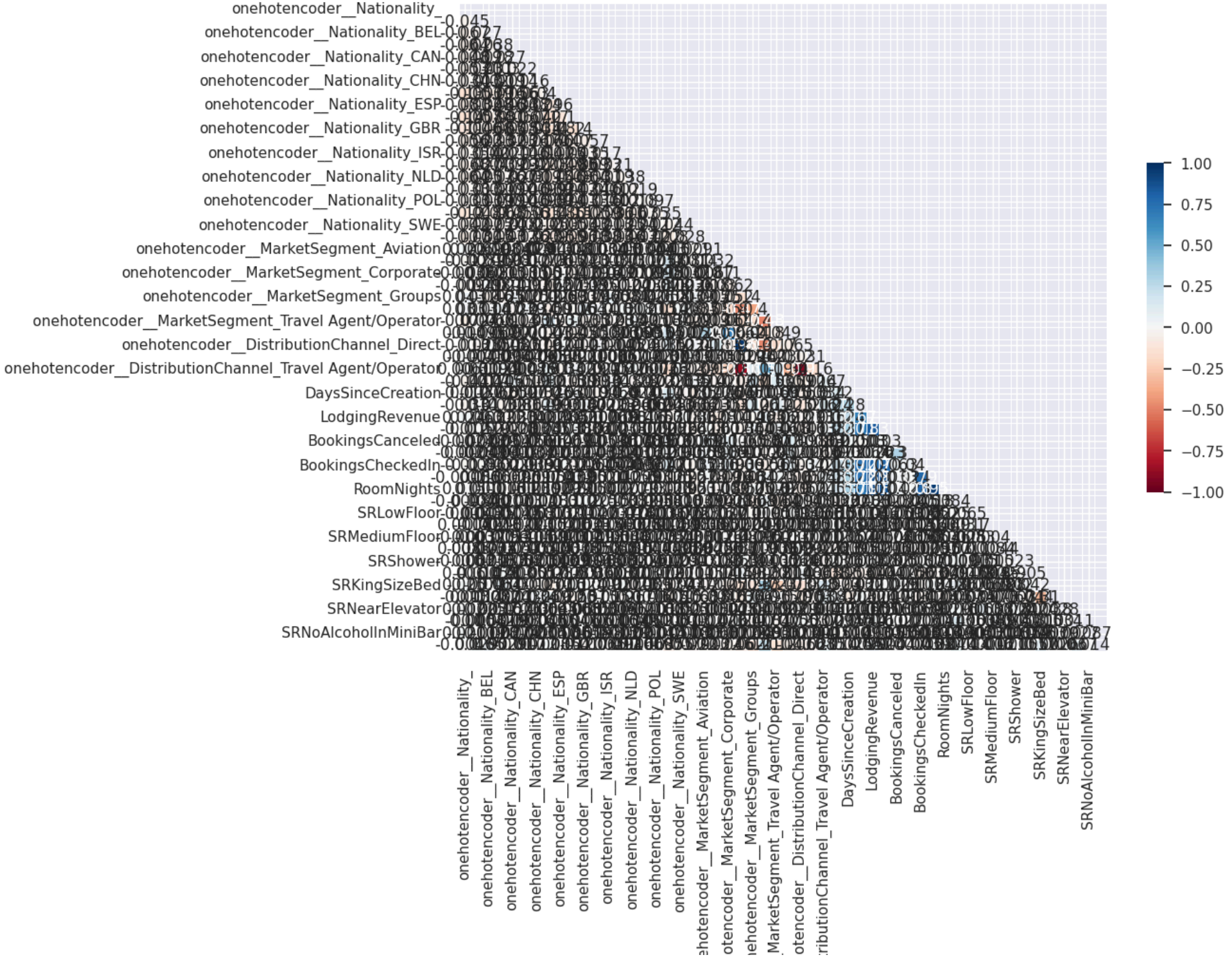
```
High correlations (Pearson):
onehotencoder__MarketSegment_Direct and onehotencoder__DistributionChannel_Direct: 0.9562926411128286
onehotencoder__MarketSegment_Direct and onehotencoder__DistributionChannel_Travel Agent/Operator: -0.8572925535059923
onehotencoder__DistributionChannel_Direct and onehotencoder__DistributionChannel_Travel Agent/Operator: -0.8962964580673204
PersonsNights and RoomNights: 0.8619504127467639
```

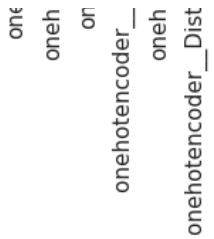
Spearman Correlation

In [ ]:
```
# Compute the correlation matrix - Spearman
corr = data_1.corr(method='spearman')

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
```

```
#cmap = sns.diverging_palette(240, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask = mask, cmap = 'RdBu', vmax = 1, vmin=-1, center = 0,
            square = True, annot = True, linewidths = .5, cbar_kws = {"shrink": .5})
```

Out[ ]: <Axes: >


onehotencoder__Nationality_
onehotencoder__Nationality_BEL
onehotencoder__Nationality_CAN
onehotencoder__Nationality_CHN
onehotencoder__Nationality_ESP
onehotencoder__Nationality_GBR
onehotencoder__Nationality_ISR
onehotencoder__Nationality_NLD
onehotencoder__Nationality_POL
onehotencoder__Nationality_SWE
onehotencoder__MarketSegment_Aviation
onehotencoder__MarketSegment_Corporate
onehotencoder__MarketSegment_Groups
onehotencoder__MarketSegment_Travel Agent/Operator
onehotencoder__DistributionChannel_Direct
onehotencoder__DistributionChannel_Travel Agent/Operator
DaysSinceCreation
LodgingRevenue
BookingsCanceled
BookingsCheckedIn
RoomNights
SRLowFloor
SRMediumFloor
SRShower
SRKingSizeBed
SRNearElevator
SRNoAlcoholInMiniBar
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



In [ ]:
```
corr = data_1.corr(method='spearman')  # Ensure using Spearman's correlation

high_correlations = []
for i in range(len(corr)):
    for j in range(i+1, len(corr.columns)):
        # Skip if column names are similar, indicating potential duplicates
        if corr.index[i].replace('remainder__', '') == corr.columns[j].replace('remainder__', ''):
            continue
        # Check if the correlation is above 0.8 or below -0.8
        if abs(corr.iloc[i, j]) > 0.8:
            high_correlations.append((corr.index[i], corr.columns[j], corr.iloc[i, j]))

# Print out the high correlations
print("High correlations (Spearman):")
for col_pair in high_correlations:
    print(f"{col_pair[0]} and {col_pair[1]}: {col_pair[2]}")
```

```
High correlations (Spearman):
onehotencoder__MarketSegment_Direct and onehotencoder__DistributionChannel_Direct: 0.9562926411128096
onehotencoder__MarketSegment_Direct and onehotencoder__DistributionChannel_Travel Agent/Operator: -0.8572925535059686
onehotencoder__DistributionChannel_Direct and onehotencoder__DistributionChannel_Travel Agent/Operator: -0.896296458067308
LodgingRevenue and OtherRevenue: 0.8313834633286915
LodgingRevenue and PersonsNights: 0.8887755162400476
LodgingRevenue and RoomNights: 0.9060915710918319
OtherRevenue and PersonsNights: 0.8659229645198842
OtherRevenue and RoomNights: 0.8414454546352966
BookingsCheckedIn and RoomNights: 0.8017315523221047
PersonsNights and RoomNights: 0.9504248879272963
```

Chi Squared

In [ ]:
```
from sklearn.feature_selection import chi2
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier

# Creating a binary target variable based on the median of 'LodgingRevenue'
median_revenue = df_scaled['LodgingRevenue'].median()
df_scaled['target'] = (df_scaled['LodgingRevenue'] > median_revenue).astype(int)
df_scaled
```

Out[ ]:

	onehotencoder__Nationality_	onehotencoder__Nationality_AUT	onehotencoder__Nationality_BEL	onehotencoder__Nationality_BRA	onehotencoder__
0	1.0	1.0	1.0	10.0	
1	1.0	1.0	1.0	1.0	
2	1.0	1.0	1.0	1.0	
3	1.0	10.0	1.0	1.0	
4	1.0	1.0	1.0	1.0	
...	...	...	...	...	
106759	10.0	1.0	1.0	1.0	
106760	1.0	10.0	1.0	1.0	
106761	1.0	1.0	1.0	1.0	
106762	1.0	1.0	1.0	1.0	
106763	1.0	1.0	1.0	1.0	

106764 rows × 55 columns

In [ ]:

```
# List of binary features
binary_features = [
    "SRHighFloor", "SRLowFloor", "SRAccessibleRoom",
    "SRMediumFloor", "SRBathtub", "SRShower",
    "SRCrib", "SRKingSizeBed", "SRTwinBed",
    "SRNearElevator", "SRAwayFromElevator", "SRNoAlcoholInMiniBar",
    "SRQuietRoom", "target"
]
binary_features
```

Out[ ]:
```
['SRHighFloor',
 'SRLowFloor',
 'SRAccessibleRoom',
 'SRMediumFloor',
 'SRBathtub',
 'SRShower',
 'SRCrib',
 'SRKingSizeBed',
 'SRTwinBed',
 'SRNearElevator',
 'SRAwayFromElevator',
 'SRNoAlcoholInMiniBar',
 'SRQuietRoom',
 'target']
```

In [ ]:
```
encoded = df_scaled.columns[df_scaled.columns.str.startswith('onehotencoder')].tolist()
encoded
```

Out[ ]:
```
['onehotencoder__Nationality_',
 'onehotencoder__Nationality_AUT',
 'onehotencoder__Nationality_BEL',
 'onehotencoder__Nationality_BRA',
 'onehotencoder__Nationality_CAN',
 'onehotencoder__Nationality_CHE',
 'onehotencoder__Nationality_CHN',
 'onehotencoder__Nationality_DEU',
 'onehotencoder__Nationality_ESP',
 'onehotencoder__Nationality_FRA',
 'onehotencoder__Nationality_GBR',
 'onehotencoder__Nationality_IRL',
 'onehotencoder__Nationality_ISR',
 'onehotencoder__Nationality_ITA',
 'onehotencoder__Nationality_NLD',
 'onehotencoder__Nationality_NOR',
 'onehotencoder__Nationality_POL',
 'onehotencoder__Nationality_PRT',
 'onehotencoder__Nationality_SWE',
 'onehotencoder__Nationality_USA',
 'onehotencoder__MarketSegment_Aviation',
 'onehotencoder__MarketSegment_Complementary',
 'onehotencoder__MarketSegment_Corporate',
 'onehotencoder__MarketSegment_Direct',
 'onehotencoder__MarketSegment_Groups',
 'onehotencoder__MarketSegment_Other',
 'onehotencoder__MarketSegment_Travel Agent/Operator',
 'onehotencoder__DistributionChannel_Corporate',
 'onehotencoder__DistributionChannel_Direct',
 'onehotencoder__DistributionChannel_GDS Systems',
 'onehotencoder__DistributionChannel_Travel Agent/Operator']
```

In [ ]:
```
chi_features = ['SRHighFloor',
 'SRLowFloor',
 'SRAccessibleRoom',
 'SRMediumFloor',
 'SRBathtub',
 'SRShower',
 'SRCrib',
 'SRKingSizeBed',
 'SRTwinBed',
 'SRNearElevator',
 'SRAwayFromElevator',
 #'SRNoAlcoholInMiniBar',
```

```
 'SRQuietRoom','onehotencoder__Nationality_',
 'onehotencoder__Nationality_DEU',
 'onehotencoder__Nationality_FRA',
 'onehotencoder__Nationality_PRT',
 'onehotencoder__MarketSegment_Aviation',
 'onehotencoder__MarketSegment_Complementary',
 'onehotencoder__MarketSegment_Corporate',
 'onehotencoder__MarketSegment_Direct',
 'onehotencoder__MarketSegment_Groups',
 'onehotencoder__MarketSegment_Other',
 'onehotencoder__MarketSegment_Travel Agent/Operator',
 'onehotencoder__DistributionChannel_Corporate',
 'onehotencoder__DistributionChannel_Direct',
 'onehotencoder__DistributionChannel_GDS Systems',
 'onehotencoder__DistributionChannel_Travel Agent/Operator']
chi_features
```

Out[ ]:

```
['SRHighFloor',
 'SRLowFloor',
 'SRAccessibleRoom',
 'SRMediumFloor',
 'SRBathtub',
 'SRShower',
 'SRCrib',
 'SRKingSizeBed',
 'SRTwinBed',
 'SRNearElevator',
 'SRAwayFromElevator',
 'SRQuietRoom',
 'onehotencoder__Nationality_',
 'onehotencoder__Nationality_DEU',
 'onehotencoder__Nationality_FRA',
 'onehotencoder__Nationality_PRT',
 'onehotencoder__MarketSegment_Aviation',
 'onehotencoder__MarketSegment_Complementary',
 'onehotencoder__MarketSegment_Corporate',
 'onehotencoder__MarketSegment_Direct',
 'onehotencoder__MarketSegment_Groups',
 'onehotencoder__MarketSegment_Other',
 'onehotencoder__MarketSegment_Travel Agent/Operator',
 'onehotencoder__DistributionChannel_Corporate',
 'onehotencoder__DistributionChannel_Direct',
 'onehotencoder__DistributionChannel_GDS Systems',
 'onehotencoder__DistributionChannel_Travel Agent/Operator']
```

In [ ]:
```
X = df_scaled[chi_features]
y = df_scaled['target']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

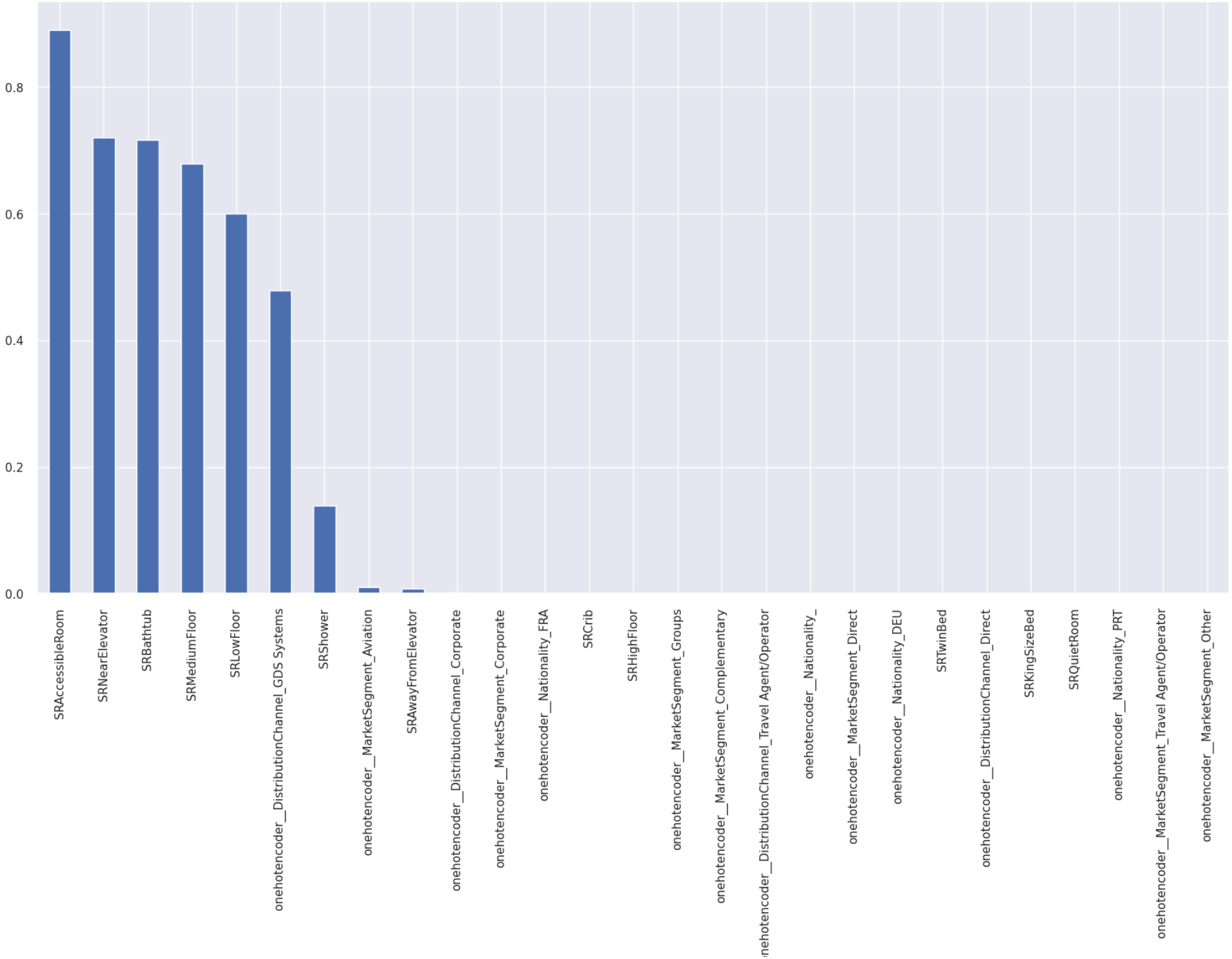
In [ ]:
```
chi_scores = chi2(df_scaled[chi_features], df_scaled['target'])
chi_scores
```

Out[ ]:
```
(array([1.50475141e+02, 2.75143554e-01, 1.91233651e-02, 1.71575976e-01,
        1.32032200e-01, 2.19534114e+00, 1.38152694e+02, 8.52515096e+02,
        5.97068808e+02, 1.28624284e-01, 6.97238797e+00, 1.12634112e+03,
        2.64635549e+02, 3.36031572e+02, 5.46368432e+01, 2.65221066e+03,
        6.62047505e+00, 1.88543792e+02, 1.10288780e+01, 2.69202896e+02,
        1.69556841e+02, 2.37945780e+03, 1.67291962e+03, 1.05690452e+01,
        6.34536089e+02, 5.00663580e-01, 2.17113871e+02]),
 array([1.36491728e-034, 5.99902117e-001, 8.90013397e-001, 6.78714969e-001,
        7.16334221e-001, 1.38428559e-001, 6.74831690e-032, 2.06433439e-187,
        7.26706930e-132, 7.19862715e-001, 8.27769593e-003, 6.22091561e-247,
        1.67537001e-059, 4.66808122e-075, 1.44991762e-013, 0.00000000e+000,
        1.00812940e-002, 6.60991953e-043, 8.97034245e-004, 1.69291612e-060,
        9.24607676e-039, 0.00000000e+000, 0.00000000e+000, 1.14997106e-003,
        5.15547951e-140, 4.79208696e-001, 3.85425500e-049]))
```

In [ ]:
```
p_values = pd.Series(chi_scores[1],index = chi_features)
p_values.sort_values(ascending = False , inplace = True)
```

In [ ]:
```
plt.figure(figsize=(20,10))
p_values.plot.bar()
```

Out[ ]:
```
<Axes: >
```


0.8
0.6
0.4
0.2
0.0
SRAccessibleRoom
SRNearElevator
SRBathtub
SRMediumFloor
SRLowFloor
onehotencoder__DistributionChannel_GDS Systems
SRShower
onehotencoder__MarketSegment_Aviation
SRAwayFromElevator
onehotencoder__DistributionChannel_Corporate
onehotencoder__MarketSegment_Corporate
onehotencoder__Nationality_FRA
SRCrib
SRHighFloor
onehotencoder__MarketSegment_Groups
onehotencoder__MarketSegment_Complementary
onehotencoder__DistributionChannel_Travel Agent/Operator
onehotencoder__Nationality_
onehotencoder__MarketSegment_Direct
onehotencoder__Nationality_DEU
SRTwinBed
onehotencoder__DistributionChannel_Direct
SRKingSizeBed
SRQuietRoom
onehotencoder__Nationality_PRT
onehotencoder__MarketSegment_Travel Agent/Operator
onehotencoder__MarketSegment_Other

In [ ]:

PCA

In [ ]:
```
df_pca = df_scaled.copy()
```

In [ ]:
```
numeric_cols = df_pca.select_dtypes(include=[np.number]).columns
```

In [ ]:
```
pca = PCA()
pca_feat = pca.fit_transform(df_pca[numeric_cols])
pca_feat_names = [f"PC{i}" for i in range(pca.n_components_)]
pca_df = pd.DataFrame(pca_feat, index=df_pca.index, columns=pca_feat_names)
```

In [ ]:
```
pca_df
```

Out[ ]:

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	...	PC45	PC46	PC47
0	-6.065653	-4.321054	1.790015	-1.058040	0.692184	0.499606	0.727668	-0.141117	1.148395	-1.544436	...	0.000822	-0.100599	0.099772
1	13.420813	-4.990012	-2.332093	0.572722	0.932071	-0.920375	0.166776	0.938885	-1.504744	-0.103027	...	0.001912	0.048009	0.022545
2	-3.877055	2.689467	-6.798252	8.165486	-0.616945	3.714969	-2.683104	-2.520999	3.080820	6.671476	...	-0.008150	0.042729	0.049171
3	-3.839501	2.693579	-8.218008	0.473964	0.608832	1.955292	-0.614790	-1.453468	3.555606	7.379031	...	-0.008319	-0.002303	0.004187
4	2.572026	6.449908	2.526239	-5.199795	0.340672	4.108658	-5.638010	-0.040694	-6.267526	-0.312262	...	0.001783	0.032409	-0.004064
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
106759	-6.105898	-4.463506	1.472122	-1.995354	-1.359829	0.574168	0.709201	6.703420	2.565090	-1.417519	...	-0.009733	0.038531	0.006893
106760	-3.938467	0.349177	-3.865632	1.046382	0.179482	-2.847746	-1.436950	1.201189	-2.128825	7.949186	...	-0.003615	0.027405	0.017854
106761	-5.234624	-4.378653	2.141418	-2.703878	-2.273528	-4.419839	-1.972244	-3.389381	3.548636	-1.432021	...	-0.000315	-0.233862	-0.088509
106762	6.830103	2.898345	-4.575232	-1.638325	-2.234157	1.531753	0.078875	-1.565800	3.716292	-0.733164	...	-0.002879	0.019275	-0.081463
106763	13.440908	-4.834302	-2.149688	0.445144	1.063905	-0.514973	0.277310	0.879810	-1.353729	-0.157466	...	0.001232	-0.002072	-0.236438

106764 rows × 55 columns

In [ ]:
```
# Output PCA table
pd.DataFrame(
    {"Eigenvalue": pca.explained_variance_,
     "Difference": np.insert(np.diff(pca.explained_variance_), 0, 0),
     "Proportion": pca.explained_variance_ratio_,
     "Cumulative": np.cumsum(pca.explained_variance_ratio_)},
    index=range(1, pca.n_components_ + 1)
)
```

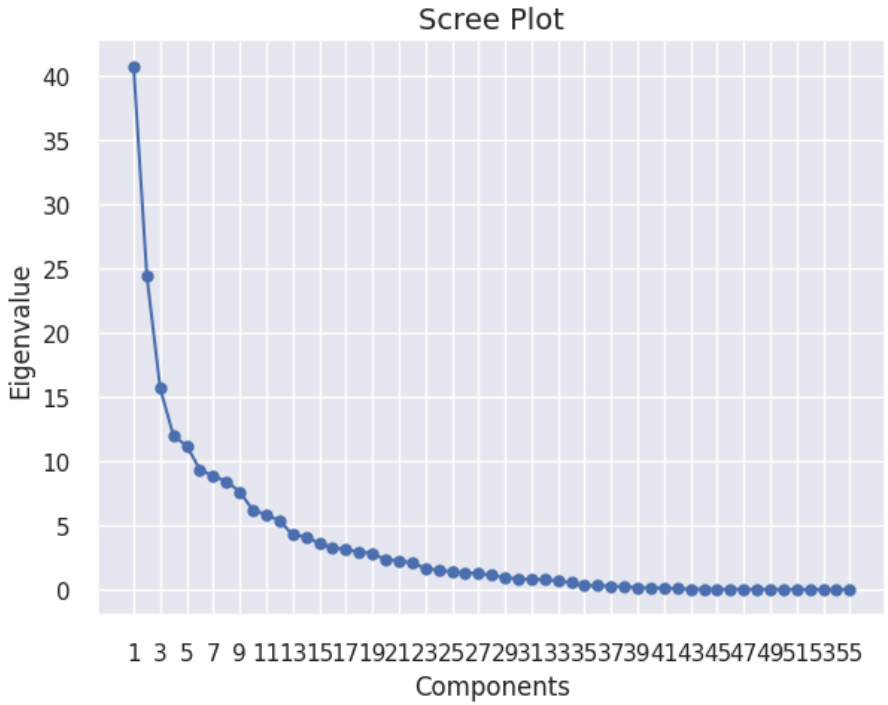
Out[ ]:

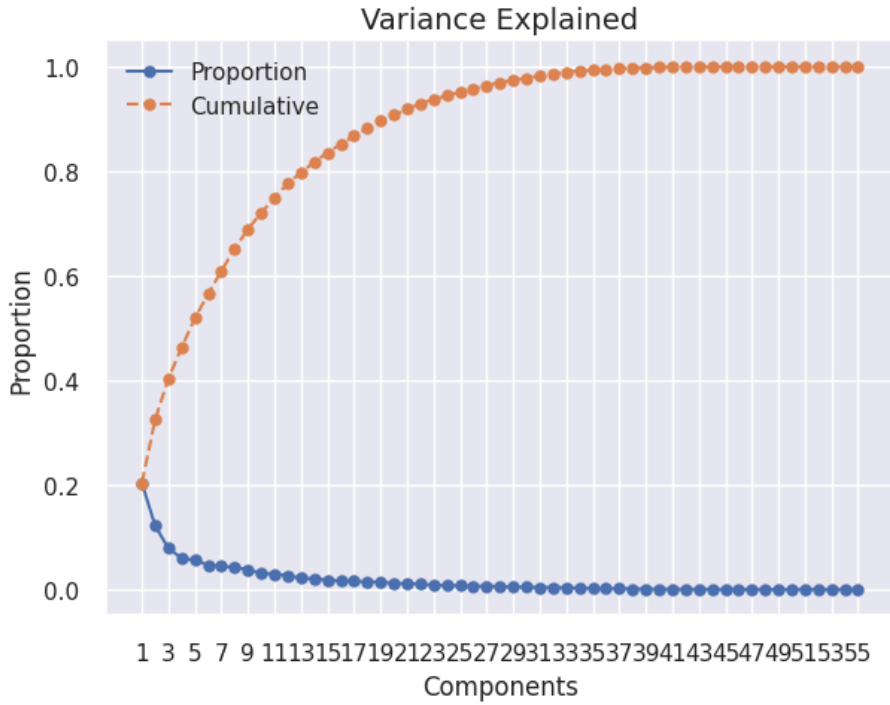
	Eigenvalue	Difference	Proportion	Cumulative
1	4.074457e+01	0.000000e+00	2.032315e-01	0.203232
2	2.440713e+01	-1.633744e+01	1.217413e-01	0.324973
3	1.568955e+01	-8.717574e+00	7.825857e-02	0.403231
4	1.198118e+01	-3.708374e+00	5.976142e-02	0.462993
5	1.117893e+01	-8.022519e-01	5.575983e-02	0.518753
6	9.337047e+00	-1.841879e+00	4.657265e-02	0.565325
7	8.833117e+00	-5.039300e-01	4.405907e-02	0.609384
8	8.411653e+00	-4.214641e-01	4.195684e-02	0.651341
9	7.611045e+00	-8.006073e-01	3.796345e-02	0.689305
10	6.159799e+00	-1.451247e+00	3.072472e-02	0.720029
11	5.834719e+00	-3.250797e-01	2.910324e-02	0.749133
12	5.346160e+00	-4.885586e-01	2.666634e-02	0.775799
13	4.254723e+00	-1.091437e+00	2.122231e-02	0.797021
14	4.103942e+00	-1.507810e-01	2.047023e-02	0.817492
15	3.573895e+00	-5.300476e-01	1.782638e-02	0.835318
16	3.279116e+00	-2.947787e-01	1.635604e-02	0.851674
17	3.093875e+00	-1.852407e-01	1.543207e-02	0.867106
18	2.932299e+00	-1.615763e-01	1.462614e-02	0.881732
19	2.808911e+00	-1.233879e-01	1.401069e-02	0.895743
20	2.317717e+00	-4.911945e-01	1.156064e-02	0.907303
21	2.254371e+00	-6.334590e-02	1.124467e-02	0.918548
22	2.045602e+00	-2.087691e-01	1.020334e-02	0.928752
23	1.655498e+00	-3.901035e-01	8.257529e-03	0.937009
24	1.486931e+00	-1.685668e-01	7.416728e-03	0.944426

	Eigenvalue	Difference	Proportion	Cumulative
25	1.387710e+00	-9.922150e-02	6.921817e-03	0.951348
26	1.286962e+00	-1.007483e-01	6.419290e-03	0.957767
27	1.227739e+00	-5.922224e-02	6.123893e-03	0.963891
28	1.111359e+00	-1.163800e-01	5.543396e-03	0.969434
29	9.201133e-01	-1.912460e-01	4.589472e-03	0.974024
30	8.160580e-01	-1.040553e-01	4.070450e-03	0.978094
31	7.884021e-01	-2.765591e-02	3.932504e-03	0.982027
32	7.718881e-01	-1.651403e-02	3.850133e-03	0.985877
33	6.747870e-01	-9.710107e-02	3.365798e-03	0.989243
34	5.166526e-01	-1.581344e-01	2.577033e-03	0.991820
35	2.945008e-01	-2.221518e-01	1.468953e-03	0.993289
36	2.884940e-01	-6.006707e-03	1.438992e-03	0.994727
37	2.610999e-01	-2.739411e-02	1.302351e-03	0.996030
38	2.219541e-01	-3.914587e-02	1.107094e-03	0.997137
39	1.455946e-01	-7.635948e-02	7.262174e-04	0.997863
40	1.360262e-01	-9.568371e-03	6.784909e-04	0.998542
41	1.080750e-01	-2.795120e-02	5.390719e-04	0.999081
42	6.116767e-02	-4.690735e-02	3.051008e-04	0.999386
43	3.148863e-02	-2.967904e-02	1.570635e-04	0.999543
44	2.844853e-02	-3.040102e-03	1.418996e-04	0.999685
45	1.695546e-02	-1.149307e-02	8.457286e-05	0.999769
46	1.666373e-02	-2.917318e-04	8.311772e-05	0.999852
47	1.127716e-02	-5.386566e-03	5.624984e-05	0.999909
48	7.245271e-03	-4.031893e-03	3.613899e-05	0.999945

	Eigenvalue	Difference	Proportion	Cumulative
49	6.926470e-03	-3.188009e-04	3.454883e-05	0.999979
50	1.752719e-03	-5.173752e-03	8.742459e-06	0.999988
51	1.480087e-03	-2.726316e-04	7.382588e-06	0.999996
52	8.946086e-04	-5.854784e-04	4.462256e-06	1.000000
53	1.093519e-29	-8.946086e-04	5.454409e-32	1.000000
54	4.163915e-30	-6.771274e-30	2.076937e-32	1.000000
55	4.104892e-30	-5.902349e-32	2.047496e-32	1.000000

In [ ]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

# draw plots
ax1.plot(pca.explained_variance_, marker=".", markersize=12)
ax2.plot(pca.explained_variance_ratio_, marker=".", markersize=12, label="Proportion")
ax2.plot(np.cumsum(pca.explained_variance_ratio_), marker=".", markersize=12, linestyle="--", label="Cumulative")

# customizations
ax2.legend()
ax1.set_title("Scree Plot", fontsize=14)
ax2.set_title("Variance Explained", fontsize=14)
ax1.set_ylabel("Eigenvalue")
ax2.set_ylabel("Proportion")
ax1.set_xlabel("Components")
ax2.set_xlabel("Components")
ax1.set_xticks(range(0, pca.n_components_, 2))
ax1.set_xticklabels(range(1, pca.n_components_ + 1, 2))
ax2.set_xticks(range(0, pca.n_components_, 2))
ax2.set_xticklabels(range(1, pca.n_components_ + 1, 2))

plt.show()
```





In [ ]:
```
# Concatenate the original numeric features with PCA components
combined_df = pd.concat([df_pca[numeric_cols], pca_df], axis=1)

# Calculate the correlation between original features and PCA components
loadings = combined_df.corr().loc[numeric_cols, pca_feat_names]
```

In [ ]:
```
def _color_red_or_green(val):
    if val < -0.45:
        color = 'background-color: red'
    elif val > 0.45:
        color = 'background-color: green'
    else:
        color = ''
    return color

# Apply the styling to the loadings DataFrame
styled_loadings = loadings.style.applymap(_color_red_or_green)
display(styled_loadings)
```

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC
onehotencoder__Nationality_	-0.035701	-0.062618	-0.057046	-0.170926	-0.185013	0.133202	0.080512	0.830119	0.236399	-0.00110
onehotencoder__Nationality_AUT	-0.024658	0.018090	-0.008259	-0.015898	-0.015754	-0.009981	0.011078	0.011149	-0.001172	0.04719
onehotencoder__Nationality_BEL	0.000663	-0.022569	-0.034091	-0.003485	0.031317	-0.041623	0.045001	0.024441	0.042957	0.10097
onehotencoder__Nationality_BRA	-0.055542	-0.044270	0.013662	-0.031010	0.010448	-0.002076	0.038508	0.000934	0.057723	-0.00917
onehotencoder__Nationality_CAN	0.007532	-0.013248	0.022375	-0.044989	-0.025801	0.064910	0.015395	-0.008945	0.020053	0.00340
onehotencoder__Nationality_CHE	-0.017348	-0.029523	-0.050061	-0.008659	0.001881	-0.020648	0.009290	0.011942	0.001281	0.01750
onehotencoder__Nationality_CHN	-0.024991	0.007260	-0.049694	-0.002888	-0.005831	-0.006293	0.005641	0.000131	0.018596	0.00654
onehotencoder__Nationality_DEU	-0.001665	0.272146	0.047820	-0.303039	0.717970	0.138783	-0.425577	-0.078067	-0.201368	-0.09066
onehotencoder__Nationality_ESP	0.022804	-0.032585	-0.037308	-0.036358	-0.037428	-0.042805	0.018338	0.046491	-0.062511	0.02629
onehotencoder__Nationality_FRA	-0.011169	0.012948	0.193924	0.897733	-0.123755	0.221270	-0.193816	-0.100908	-0.034082	-0.04160
onehotencoder__Nationality_GBR	-0.062338	-0.037675	-0.111151	-0.123277	-0.036719	0.063818	0.761610	-0.336720	-0.365118	0.05111
onehotencoder__Nationality_IRL	0.002241	-0.045922	-0.064104	-0.003503	0.023694	-0.012878	0.026928	0.010487	0.029455	-0.02246
onehotencoder__Nationality_ISR	-0.005555	0.006359	0.005578	-0.037904	-0.044940	0.048571	-0.008014	-0.009013	-0.006607	0.01280
onehotencoder__Nationality_ITA	-0.023362	-0.036591	-0.015432	-0.028462	-0.017919	-0.033239	0.009634	0.022719	-0.027218	0.03214
onehotencoder__Nationality_NLD	-0.019945	0.027100	-0.029204	0.001972	0.025680	-0.051501	0.061643	0.027109	0.058052	0.03579
onehotencoder__Nationality_NOR	-0.010658	-0.020693	-0.019786	-0.012218	-0.005411	0.002817	0.004806	0.003297	0.000516	-0.01161
onehotencoder__Nationality_POL	-0.010680	-0.014704	-0.013237	-0.010813	-0.007941	-0.006642	0.002990	0.006440	-0.002899	0.01014
onehotencoder__Nationality_PRT	0.183685	-0.053267	0.055896	-0.241021	-0.377626	-0.570764	-0.285822	-0.360768	0.308622	0.00300
onehotencoder__Nationality_SWE	-0.028043	0.002350	-0.020050	-0.009809	0.001288	-0.009221	0.021811	0.004782	0.025379	-0.01249
onehotencoder__Nationality_USA	0.013037	-0.071790	-0.025272	-0.071462	-0.039683	0.099090	0.013226	-0.005045	0.012336	-0.05812
onehotencoder__MarketSegment_Aviation	0.042035	-0.000348	0.008309	-0.014466	-0.035389	-0.033936	-0.013031	0.023164	-0.004278	-0.00205
onehotencoder__MarketSegment_Complementary	0.091761	-0.041397	0.013299	-0.043718	-0.057021	-0.086010	-0.044830	-0.063283	0.077255	-0.01078
onehotencoder__MarketSegment_Corporate	0.130552	0.026439	0.025255	-0.081347	-0.140087	-0.161800	-0.072908	-0.029974	0.035582	-0.02346
onehotencoder__MarketSegment_Direct	0.815377	-0.474818	-0.135883	0.056054	0.136423	0.061445	0.023310	0.002371	0.011108	0.00083

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC
onehotencoder__MarketSegment_Groups	0.125015	0.302461	0.225521	-0.338630	-0.571387	0.475203	-0.235145	-0.055090	-0.283980	0.07079
onehotencoder__MarketSegment_Other	-0.837136	-0.311838	-0.397891	0.090002	0.077312	-0.120507	-0.042625	0.016155	-0.040692	-0.06556
onehotencoder__MarketSegment_Travel Agent/Operator	0.174450	0.634550	0.476008	0.156520	0.325486	-0.213163	0.279798	0.045280	0.259876	0.04130
onehotencoder__DistributionChannel_Corporate	0.161197	0.016214	0.031950	-0.102515	-0.186029	-0.146346	-0.095173	-0.018806	0.011924	-0.01251
onehotencoder__DistributionChannel_Direct	0.828109	-0.481581	-0.138587	0.046148	0.122873	0.044416	0.010692	-0.009968	0.015657	-0.00264
onehotencoder__DistributionChannel_GDS Systems	0.006129	-0.042787	-0.065367	0.000187	0.005888	-0.054931	0.004832	0.022399	-0.058745	-0.03987
onehotencoder__DistributionChannel_Travel Agent/Operator	-0.834874	0.449097	0.128710	-0.001756	-0.040614	0.028262	0.027310	0.012404	-0.007788	0.01530
Age	0.032499	0.203540	0.157847	-0.137143	-0.002536	0.120802	-0.038891	-0.087095	-0.050381	0.10083
DaysSinceCreation	-0.014519	0.074323	0.077155	0.012092	-0.216883	-0.277900	-0.155994	-0.073550	-0.118825	0.04746
AverageLeadTime	-0.017931	0.218704	0.122790	-0.115839	0.079643	0.112330	0.012279	-0.080352	-0.069197	0.06781
LodgingRevenue	-0.014107	-0.060456	-0.037044	-0.001401	-0.014419	0.012470	-0.016529	0.016681	-0.033834	0.05177
OtherRevenue	0.008922	0.009307	0.032234	0.024161	-0.007504	0.014895	-0.015346	-0.024038	-0.009771	0.03382
BookingsCanceled	0.020010	-0.017663	0.019230	-0.020112	-0.023510	-0.026319	-0.016138	-0.023858	0.029040	0.00215
BookingsNoShowed	0.014957	-0.008978	0.012766	-0.015850	-0.018016	-0.019461	-0.011050	-0.019022	0.023981	0.00233
BookingsCheckedIn	0.032284	0.002687	0.060257	-0.056627	-0.085518	-0.093309	-0.056513	-0.060958	0.010703	0.03219
PersonsNights	-0.050023	-0.014697	0.023424	0.035050	0.017706	-0.034789	-0.008383	-0.005986	-0.012396	0.03936
RoomNights	-0.012619	0.013733	0.067091	-0.012686	-0.025972	-0.036116	-0.022555	-0.004627	-0.024690	0.04771
SRHighFloor	-0.060164	-0.113716	-0.040137	-0.001377	0.052527	0.011122	0.012628	-0.038599	0.052046	0.08969
SRLowFloor	-0.000865	-0.017543	-0.012503	-0.005394	-0.000462	-0.001051	0.014928	-0.007354	0.002455	0.01302
SRAccessibleRoom	-0.003187	-0.000138	-0.004839	-0.004995	-0.003215	-0.000430	0.004351	-0.001544	0.003829	-0.00050
SRMediumFloor	0.006527	-0.006850	-0.006268	0.001140	0.001436	-0.003171	-0.003462	-0.003555	0.012888	0.01572
SRBathtub	0.009829	-0.039779	0.005249	0.006523	0.009811	0.009477	0.017172	0.000925	0.017812	0.00898

	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC
SRShower	0.011397	-0.032095	0.001311	0.010159	0.011655	0.010247	0.000499	-0.010282	0.007762	0.01310
SRCrib	0.017148	-0.077306	0.015500	0.019788	0.008794	-0.012210	0.017502	0.016645	0.015993	-0.02306
SRKingSizeBed	-0.455155	-0.638407	0.509818	-0.136797	0.060668	0.198222	0.039460	-0.121255	0.204913	-0.02461
SRTwinBed	0.014379	0.397741	-0.566779	-0.056814	-0.006034	0.416999	0.045086	-0.272709	0.497129	-0.03920
SRNearElevator	0.001307	-0.004787	-0.005861	-0.003232	0.002370	0.001966	0.010850	0.000332	0.000504	0.00431
SRAwayFromElevator	0.003304	-0.035362	-0.011279	0.010039	0.017641	0.006390	-0.000219	-0.013970	0.004046	0.07302
SRNoAlcoholInMiniBar	-0.006236	-0.007072	-0.009791	-0.004238	0.000197	0.008220	0.009852	0.013650	0.011563	0.01013
SRQuietRoom	-0.199191	-0.141565	-0.123266	0.058396	0.147522	0.034507	-0.118753	-0.010956	0.046206	0.93286
target	0.074242	0.040168	0.009466	0.015425	0.013257	0.004802	0.018222	0.019110	0.082842	0.06300

In [ ]:
```
def get_redundant_pairs(df):
    '''Get diagonal and lower triangular pairs of correlation matrix'''
    pairs_to_drop = set()
    cols = df.columns
    for i in range(0, df.shape[1]):
        for j in range(0, i+1):
            pairs_to_drop.add((cols[i], cols[j]))
    return pairs_to_drop

def get_top_abs_correlations(df, n=5):
    au_corr = df.corr().abs().unstack()
    labels_to_drop = get_redundant_pairs(df)
    au_corr = au_corr.drop(labels=labels_to_drop).sort_values(ascending=False)
    return au_corr[0:n]

# Apply the correlation analysis on the PCA components
print("Top Absolute Correlations among PCA Components")
print(get_top_abs_correlations(pca_df, 15))
```

```
Top Absolute Correlations among PCA Components
PC52  PC53    1.114943e-02
      PC54    1.720403e-04
PC53  PC54    7.258429e-05
PC0   PC52    1.639911e-15
PC46  PC50    1.169276e-15
PC0   PC54    1.016840e-15
PC32  PC33    9.666104e-16
PC43  PC44    9.610040e-16
PC31  PC32    9.190318e-16
PC42  PC50    8.905157e-16
PC17  PC21    8.461502e-16
PC24  PC26    8.084145e-16
PC19  PC21    7.792619e-16
PC45  PC47    7.378711e-16
PC5   PC8     7.333245e-16
dtype: float64
```

In [ ]:

Cluster Segmentation

In [ ]:
```
def random_centroids(data, k):
    centroids = []
    for i in range(k):
        centroid = data.apply(lambda x: float(x.sample())) # Gets a random centroid from each column
        centroids.append(centroid)

    return pd.concat(centroids, axis = 1)
```

In [ ]:
```
df
```

Out[ ]:

	Nationality	Age	DaysSinceCreation	AverageLeadTime	LodgingRevenue	OtherRevenue	BookingsCanceled	BookingsNoShowed	BookingsCheck
0	BRA	51.0	205.0	41.0	416.66	271.5	0.0	0.0	
1	CHE	25.0	908.0	66.0	384.00	58.5	0.0	0.0	
2	FRA	39.0	651.0	24.0	234.00	28.0	0.0	0.0	
3	AUT	57.0	435.0	0.0	0.00	0.0	0.0	0.0	
4	DEU	42.0	576.0	8.0	432.00	21.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
106759		35.0	860.0	20.0	416.00	56.0	0.0	0.0	
106760	AUT	46.0	934.0	12.0	366.00	21.0	0.0	0.0	
106761	PRT	29.4	348.0	244.0	1287.90	354.0	0.0	0.0	
106762	ESP	37.0	363.0	20.0	615.00	71.5	0.0	0.0	
106763	SWE	53.0	738.0	8.0	1428.00	50.0	0.0	0.0	

106764 rows × 26 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(106764, 26)
```

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

```
Nationality             0
Age                     0
DaysSinceCreation       0
AverageLeadTime         0
LodgingRevenue          0
OtherRevenue            0
BookingsCanceled        0
BookingsNoShowed        0
BookingsCheckedIn       0
PersonsNights           0
RoomNights              0
DistributionChannel     0
MarketSegment           0
SRHighFloor             0
SRLowFloor              0
SRAccessibleRoom        0
SRMediumFloor           0
SRBathtub               0
SRShower                0
SRCrib                  0
SRKingSizeBed           0
SRTwinBed               0
SRNearElevator          0
SRAwayFromElevator      0
SRNoAlcoholInMiniBar    0
SRQuietRoom             0
dtype: int64
```

In [107…

```
# select Age and LodgingRevenue columns
X = df.iloc[:,[1,4]].values
```

In [108…

```
print(X)
```

```
[[  51.    416.66]
 [  25.    384.  ]
 [  39.    234.  ]
 ...
 [  29.4  1287.9 ]
 [  37.    615.  ]
 [  53.   1428.  ]]
```

In [109…

```
X.shape
```

Out[109]: (106764, 2)

In [110…

```
#Choosing the number of clusters

wscc = []

for i in range(1, 12):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=45)
    kmeans.fit(X)

    wscc.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:
```

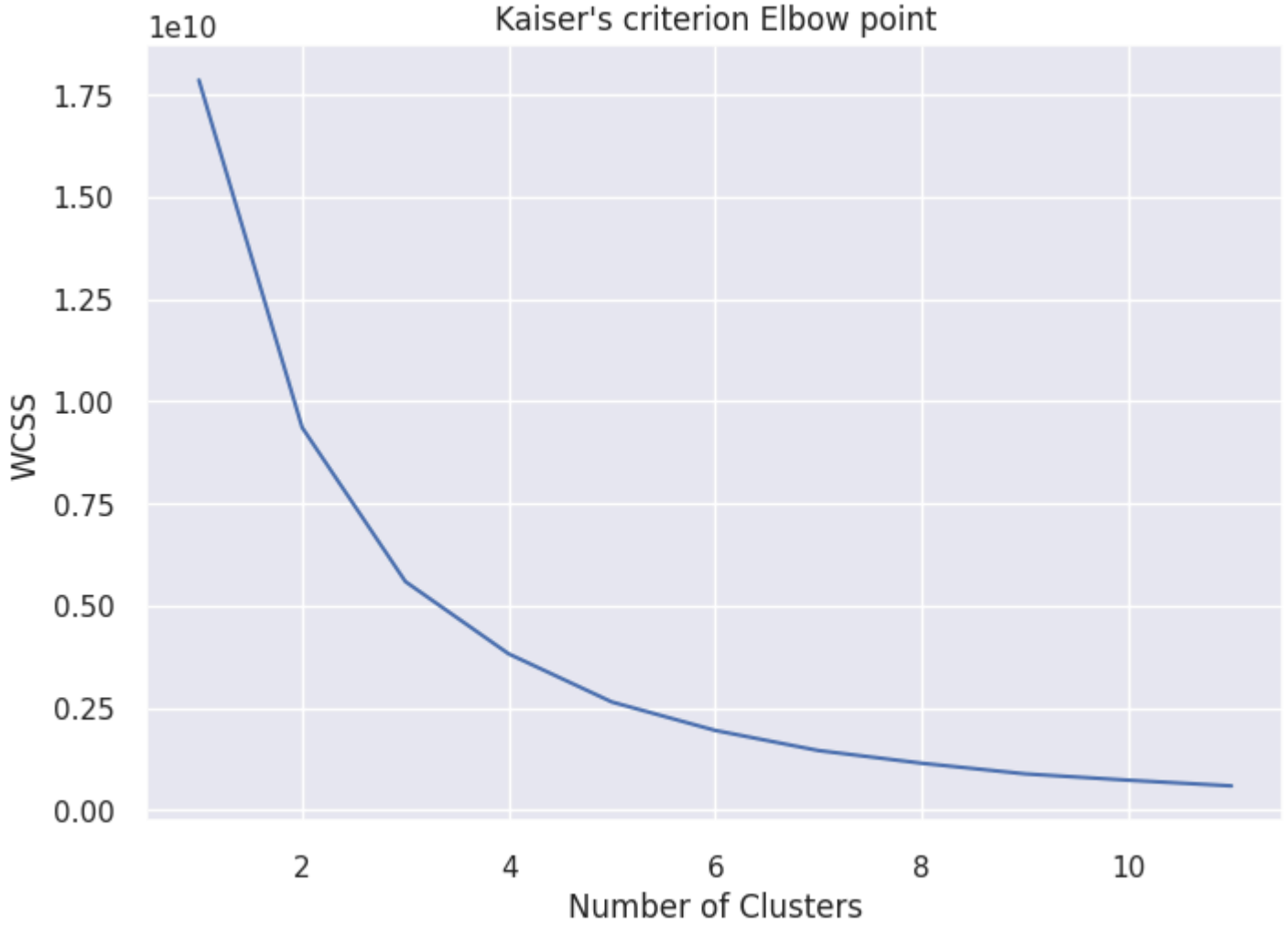
```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

In [111…

```python
# Finding WCSS value for different number of clusters
wcss = []

for i in range(1, 12):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plot the elbow graph
sns.set()
plt.plot(range(1, 12), wcss)
plt.title("Kaiser's criterion Elbow point")
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:
```

```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```



In [112…

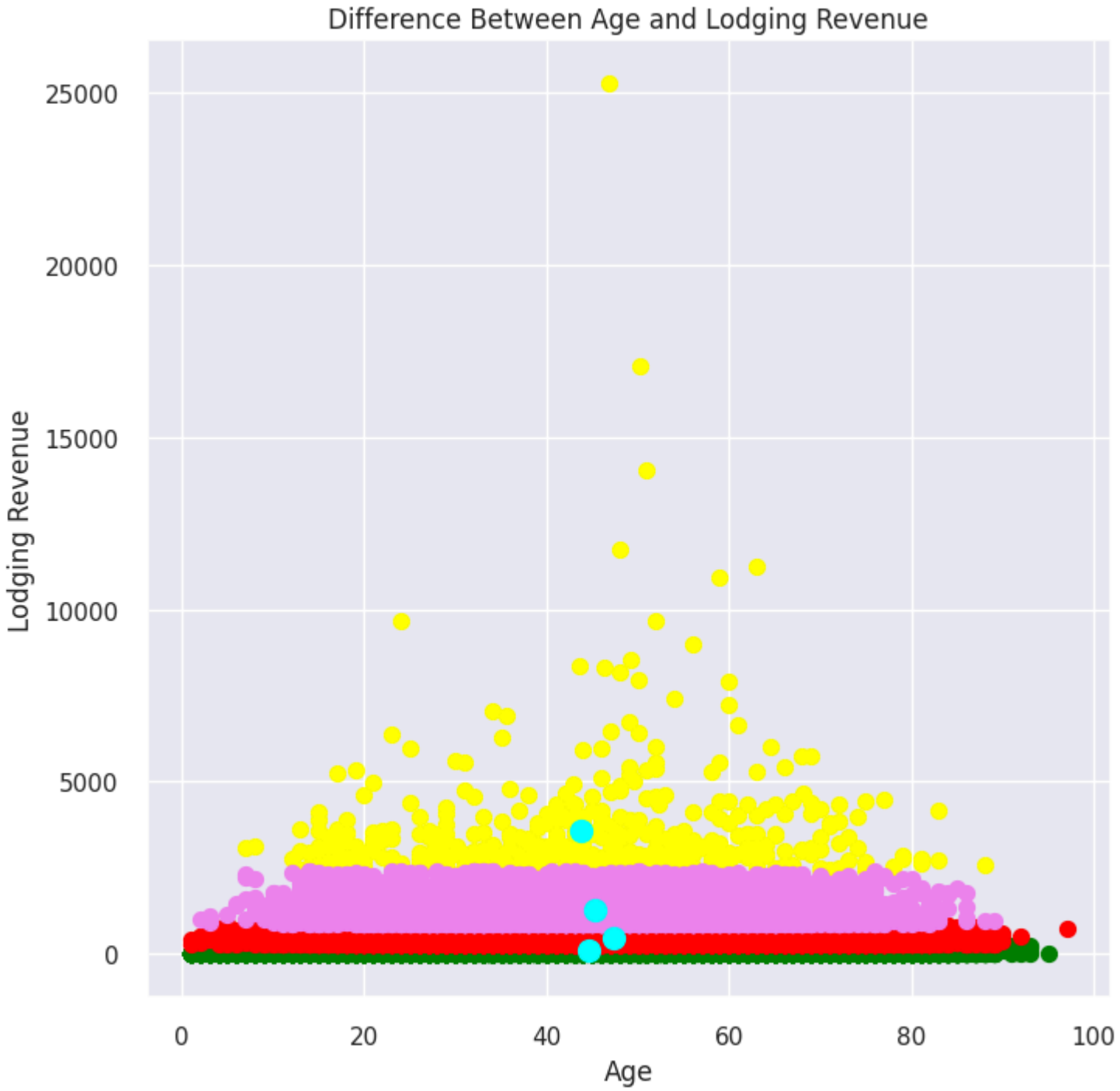
```
kmeans = KMeans(n_clusters=4, init='k-means++',random_state=0)

# return a lebel for each data point based on their cluster

Y = kmeans.fit_predict(X)

print(Y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
[1 1 0 ... 3 1 3]
```

In [113…

```
# plotting all the clusters and their Centroids

plt.figure(figsize=(8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster 3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='violet', label='Cluster 4')


# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')

plt.title('Difference Between Age and Lodging Revenue')
plt.xlabel('Age')
plt.ylabel('Lodging Revenue')
plt.show()
```


Difference Between Age and Lodging Revenue
Lodging Revenue
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
Age

In [114…
```
df.head()
```

Out[114]:

	Nationality	Age	DaysSinceCreation	AverageLeadTime	LodgingRevenue	OtherRevenue	BookingsCanceled	BookingsNoShowed	BookingsCheckedIn
0	BRA	51.0	205.0	41.0	416.66	271.5	0.0	0.0	1.0
1	CHE	25.0	908.0	66.0	384.00	58.5	0.0	0.0	1.0
2	FRA	39.0	651.0	24.0	234.00	28.0	0.0	0.0	1.0
3	AUT	57.0	435.0	0.0	0.00	0.0	0.0	0.0	0.0
4	DEU	42.0	576.0	8.0	432.00	21.0	0.0	0.0	1.0

5 rows × 26 columns

In [115…
```
from sklearn.preprocessing import StandardScaler

# Select only numeric columns for scaling
numeric_columns = df.select_dtypes(include=[np.number]).columns

# Scale the numeric columns
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[numeric_columns])

# Create a DataFrame from the scaled data
scaled_df = pd.DataFrame(scaled_data, columns=numeric_columns)

# Concatenate non-numeric columns with the scaled DataFrame
scaled_df = pd.concat([scaled_df, df[df.columns.difference(numeric_columns)]], axis=1)
```

In [ ]:
```
## CLUSTERING

cluster_nums = [2,3,4,5,6,7]
scores = []
sum_of_squared_distances = []

for cluster_num in cluster_nums:
    kmeans = KMeans(cluster_num, random_state=0)
    kmeans.fit(scaled_data)
    sum_of_squared_distances.append(kmeans.inertia_)
```

```
    clusters = kmeans.predict(scaled_data)
    silhouette = silhouette_score(scaled_data, clusters)
    scores.append(silhouette)
```

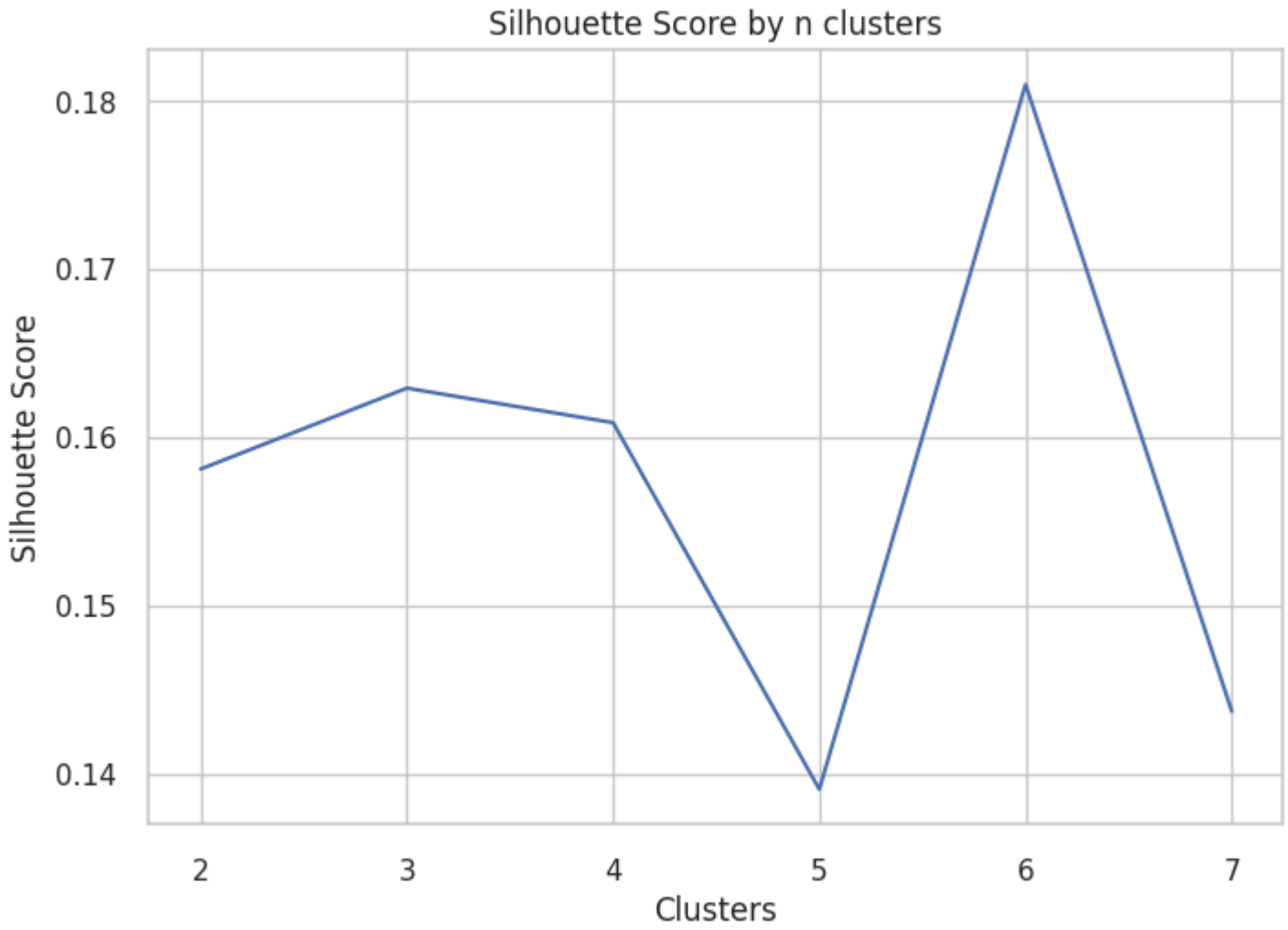
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

In [ ]:

```
# Silhouette score
sns.set_style('whitegrid')
plt.ylabel('Silhouette Score')
plt.xlabel('Clusters')
sns.lineplot(x=cluster_nums,y=scores)
plt.title('Silhouette Score by n clusters')
```

Out[ ]:

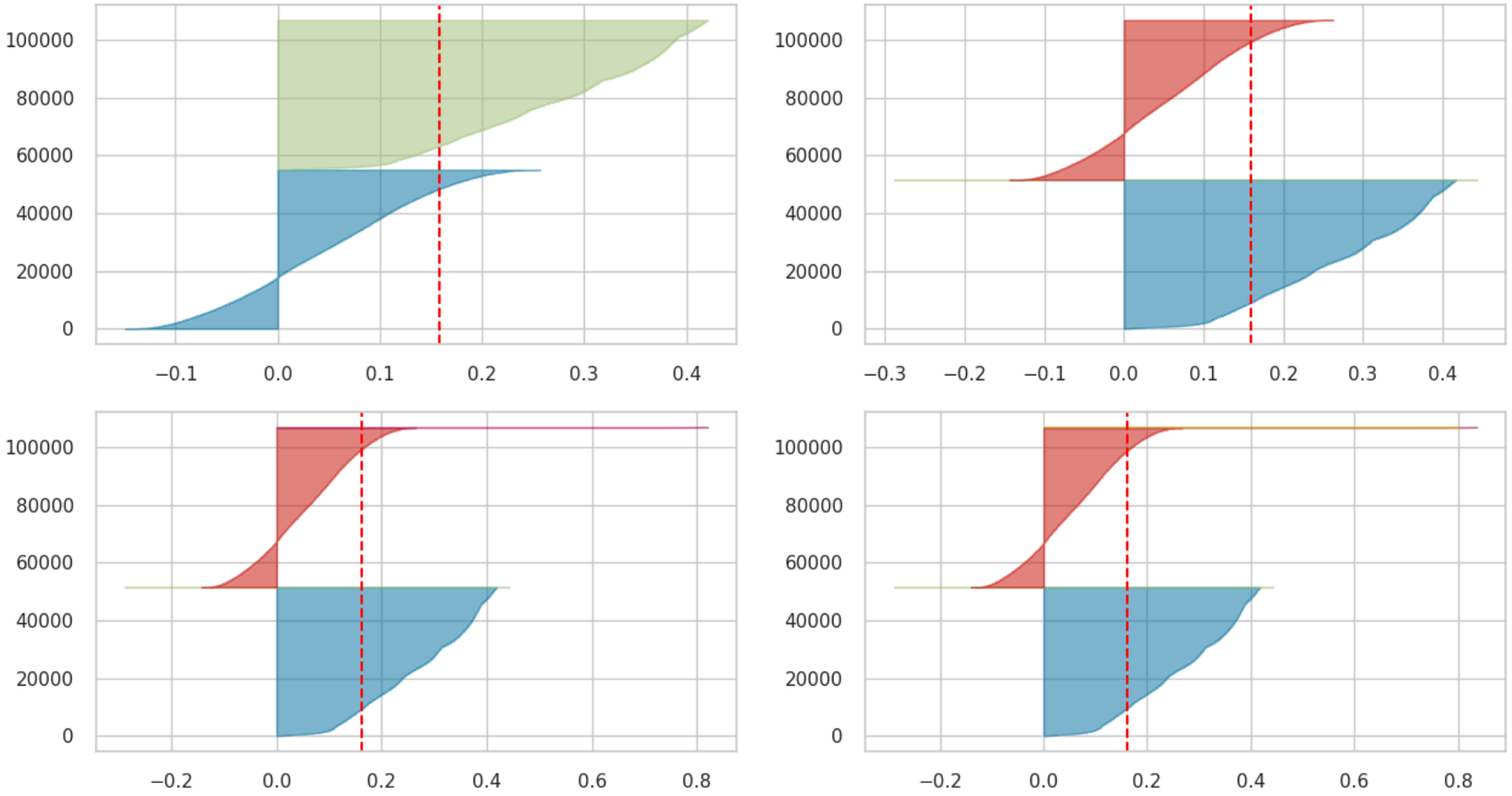
```
Text(0.5, 1.0, 'Silhouette Score by n clusters')
```



In [105…

```
ig, ax = plt.subplots(2, 2, figsize=(15,8))
for i in [2, 3, 4, 5]:
    '''
    Create KMeans instances for different number of clusters
    '''
    km = KMeans(n_clusters=i, init='k-means++', n_init=10, max_iter=100, random_state=42)
    q, mod = divmod(i, 2)
    '''
    Create SilhouetteVisualizer instance with KMeans instance
    Fit the visualizer
```

```
    '''
    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=ax[q-1][mod])
    visualizer.fit(scaled_data)
```



In [118…

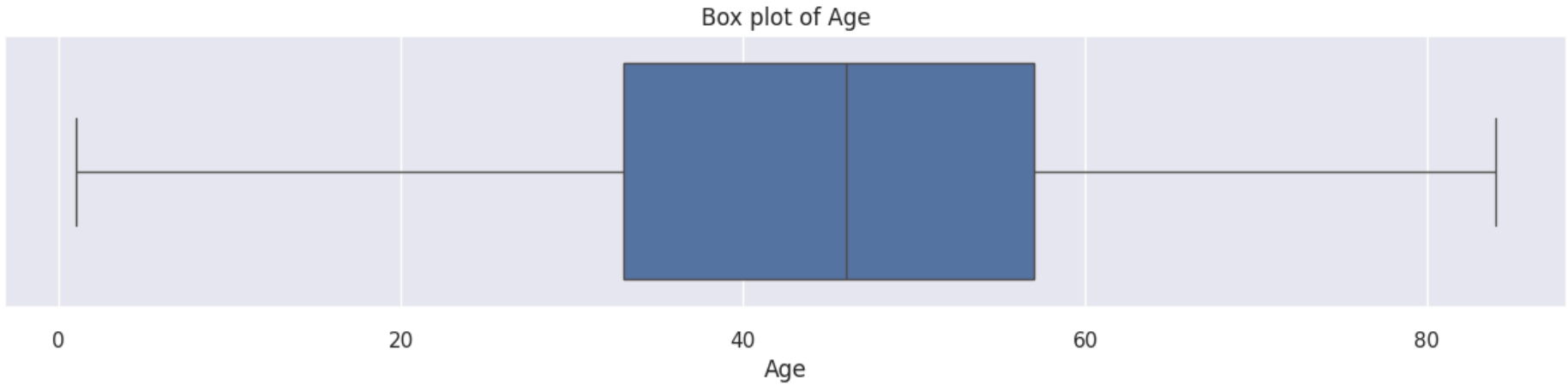
```
#Manual outlier removal
df_outlier=df.loc[ (df['Age'] < 85) & (df['AverageLeadTime'] < 350)& (df['OtherRevenue'] < 1500)  & (df['LodgingRevenue'] < 5000)
print('Percentage of data kept after removing outliers:', np.round(df_outlier.shape[0] / df.shape[0], 4))
```

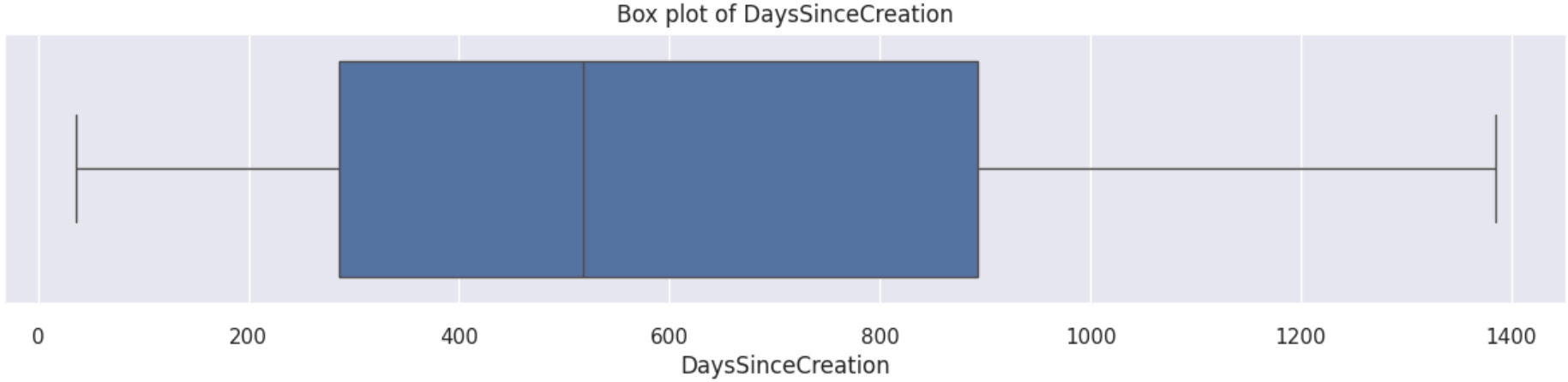
```
Percentage of data kept after removing outliers: 0.9767
```

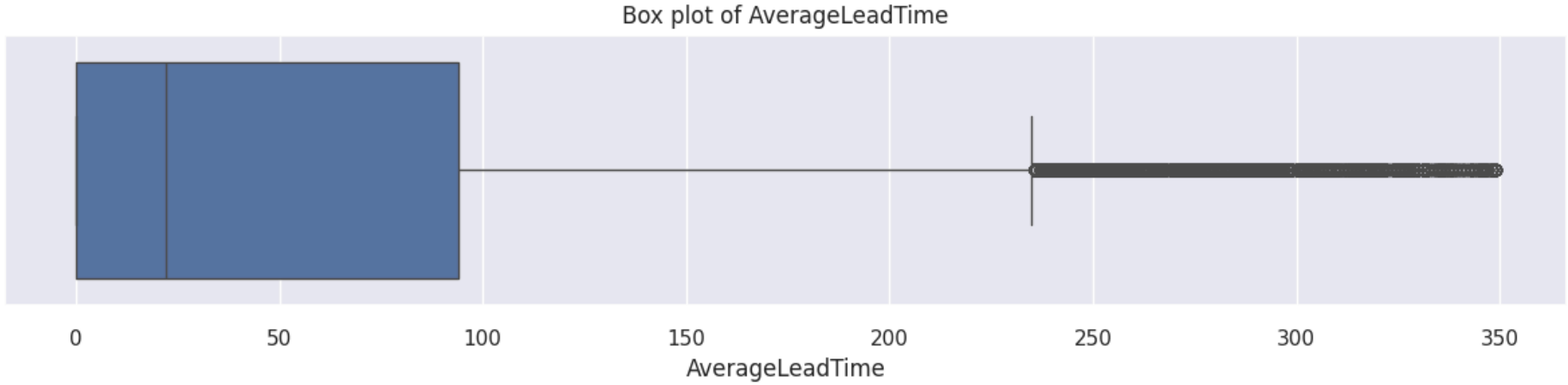
In [119…

```
# Define the numeric features
numeric_features = ['Age', 'DaysSinceCreation', 'AverageLeadTime', 'LodgingRevenue',
                    'OtherRevenue', 'BookingsCanceled', 'BookingsNoShowed',
```

```
                    'BookingsCheckedIn', 'PersonsNights', 'RoomNights']

# Set up the figure and axis with a larger figure size
fig, axes = plt.subplots(nrows=len(numeric_features), figsize=(12, 30))

# Loop through each numeric feature and plot a box plot
for i, feature in enumerate(numeric_features):
    sns.boxplot(x=df_outlier[feature], ax=axes[i])
    axes[i].set_title(f'Box plot of {feature}')
    axes[i].set_xlabel(feature)

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

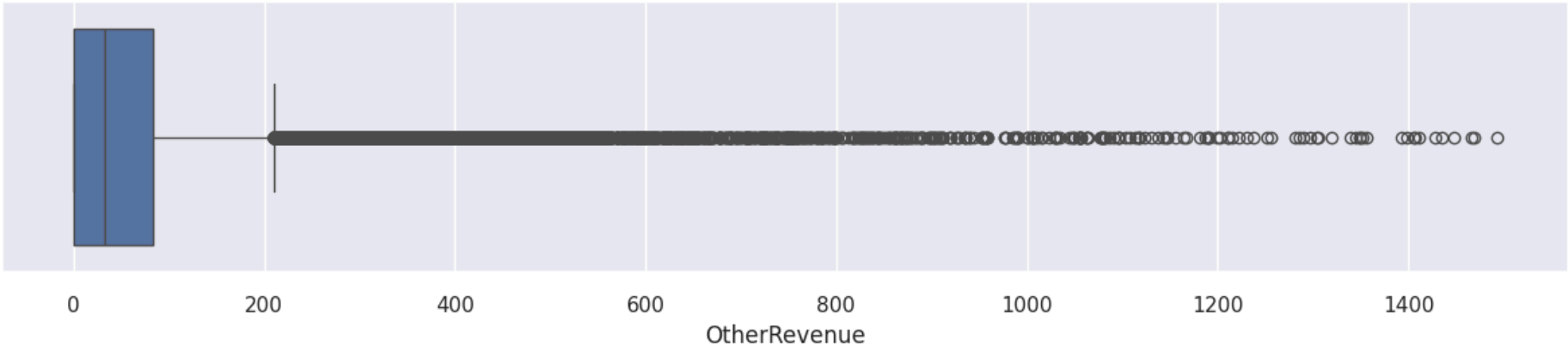




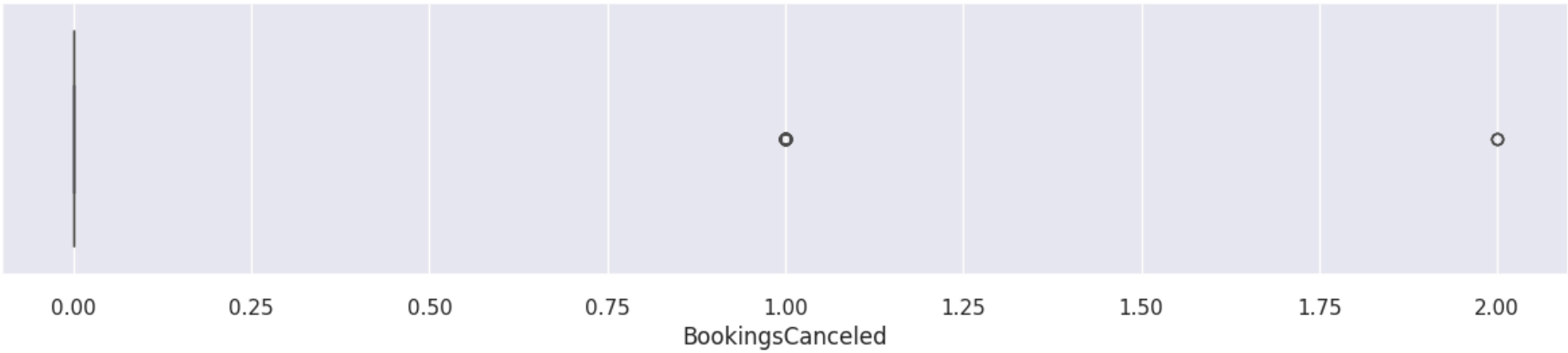


Box plot of LodgingRevenue

Box plot of OtherRevenue



Box plot of BookingsCanceled



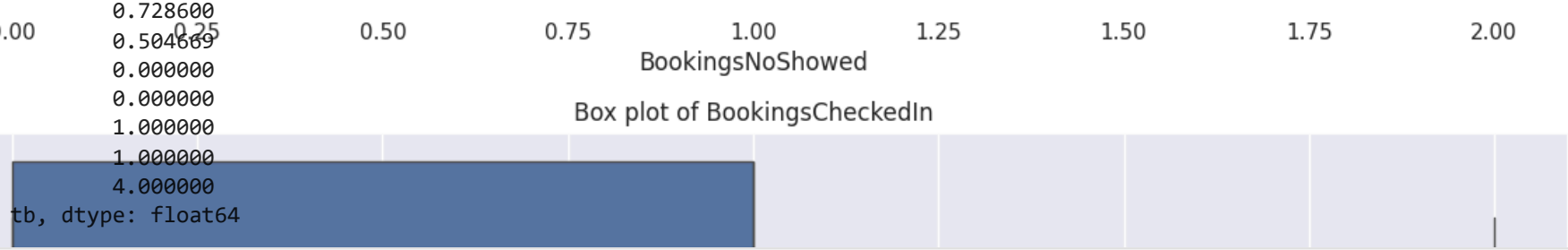
Box plot of BookingsNoShowed

In [120…
```
df=df_outlier.copy()
```

In [121…
```
df['tb']=(df['BookingsCheckedIn']+df['BookingsCanceled']+df['BookingsNoShowed'])
```

In [122…
```
df['tb'].describe()
```

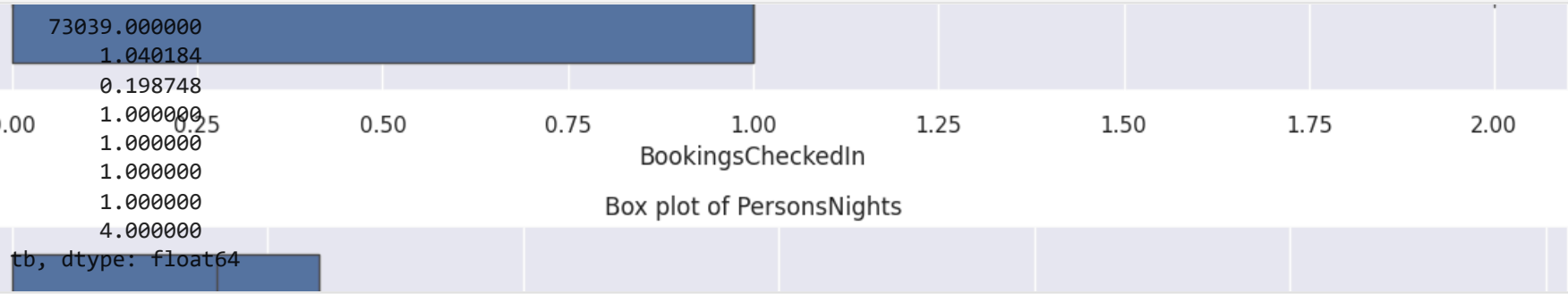
Out[122]:
```
count    104274.000000
mean          0.728600
std           0.504669
min           0.000000
25%           0.000000
50%           1.000000
75%           1.000000
max           4.000000
Name: tb, dtype: float64
```



Box plot of BookingsCheckedIn

In [123…
```
df=df[df['tb']!=0]
df['tb'].describe()
```

Out[123]:
```
count    73039.000000
mean         1.040184
std          0.198748
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          4.000000
Name: tb, dtype: float64
```

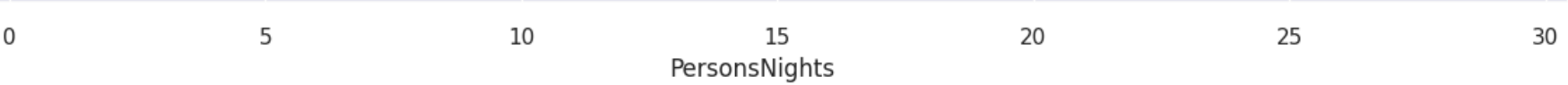


Box plot of PersonsNights

In [124…
```
df.drop('tb',axis=1,inplace=True)
```

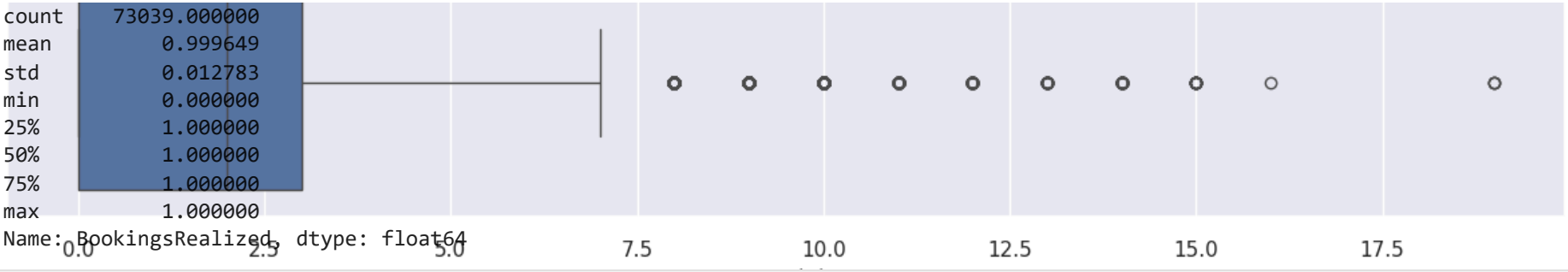
In [125…
```
df['BookingsRealized']=(df['BookingsCheckedIn']/(df['BookingsCheckedIn']+df['BookingsCanceled']+df['BookingsNoShowed']))
```

In [126…
```
df['BookingsRealized'].describe()
```



Box plot of RoomNights

Out[126]:
```
count    73039.000000
mean         0.999649
std          0.012783
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
Name: BookingsRealized, dtype: float64
```



In [127…
```
df['TotalRevenue']=(df['LodgingRevenue']+df['OtherRevenue'])
```

In [128…
```
df[df['TotalRevenue']==0].describe()
```

Out[128]:

	Age	DaysSinceCreation	AverageLeadTime	LodgingRevenue	OtherRevenue	BookingsCanceled	BookingsNoShowed	BookingsCheckedIn	Pe
count	258.000000	258.000000	258.000000	258.0	258.0	258.000000	258.0	258.000000	
mean	42.622093	701.934109	30.511628	0.0	0.0	0.015504	0.0	1.073643	
std	12.359228	388.959541	63.168109	0.0	0.0	0.152003	0.0	0.276166	
min	6.000000	79.000000	0.000000	0.0	0.0	0.000000	0.0	0.000000	
25%	35.000000	321.250000	2.000000	0.0	0.0	0.000000	0.0	1.000000	
50%	42.800000	666.000000	6.000000	0.0	0.0	0.000000	0.0	1.000000	
75%	50.950000	1061.000000	22.000000	0.0	0.0	0.000000	0.0	1.000000	
max	73.000000	1385.000000	315.000000	0.0	0.0	2.000000	0.0	2.000000	

8 rows × 25 columns

In [129…
```
df=df[df['TotalRevenue']!=0]
```

In [130…
```
df.columns
```

Out[130]:
```
Index(['Nationality', 'Age', 'DaysSinceCreation', 'AverageLeadTime',
       'LodgingRevenue', 'OtherRevenue', 'BookingsCanceled',
       'BookingsNoShowed', 'BookingsCheckedIn', 'PersonsNights', 'RoomNights',
       'DistributionChannel', 'MarketSegment', 'SRHighFloor', 'SRLowFloor',
       'SRAccessibleRoom', 'SRMediumFloor', 'SRBathtub', 'SRShower', 'SRCrib',
       'SRKingSizeBed', 'SRTwinBed', 'SRNearElevator', 'SRAwayFromElevator',
       'SRNoAlcoholInMiniBar', 'SRQuietRoom', 'BookingsRealized',
       'TotalRevenue'],
      dtype='object')
```

In [131…
```
df.drop(['NameHash','DocIDHash','MarketSegment'],axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-131-0047f5cca96d> in <cell line: 1>()
----> 1 df.drop(['NameHash','DocIDHash','MarketSegment'],axis=1,inplace=True)

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   5397                 weight  1.0     0.8
   5398         """
-> 5399         return super().drop(
   5400             labels=labels,
   5401             axis=axis,

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in drop(self, labels, axis, index, columns, level, inplace, error
s)
   4503         for axis, labels in axes.items():
   4504             if labels is not None:
-> 4505                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4506
   4507         if inplace:

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in _drop_axis(self, labels, axis, level, errors, only_slice)
   4544                 new_axis = axis.drop(labels, level=level, errors=errors)
   4545             else:
-> 4546                 new_axis = axis.drop(labels, errors=errors)
   4547             indexer = axis.get_indexer(new_axis)
   4548

/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   6932         if mask.any():
```

```
   6933             if errors != "ignore":
-> 6934                 raise KeyError(f"{list(labels[mask])} not found in axis")
   6935             indexer = indexer[~mask]
   6936         return self.delete(indexer)

KeyError: "['NameHash', 'DocIDHash'] not found in axis"
```

In [132…

```
#Remove irrelevant variables
df.drop(['NameHash','DocIDHash','MarketSegment'],axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-132-d4bb8dca9b34> in <cell line: 2>()
      1 #Remove irrelevant variables
----> 2 df.drop(['NameHash','DocIDHash','MarketSegment'],axis=1,inplace=True)

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   5397                 weight  1.0     0.8
   5398         """
-> 5399         return super().drop(
   5400             labels=labels,
   5401             axis=axis,

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in drop(self, labels, axis, index, columns, level, inplace, error
s)
   4503         for axis, labels in axes.items():
   4504             if labels is not None:
-> 4505                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4506
   4507         if inplace:

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in _drop_axis(self, labels, axis, level, errors, only_slice)
   4544                 new_axis = axis.drop(labels, level=level, errors=errors)
   4545             else:
-> 4546                 new_axis = axis.drop(labels, errors=errors)
   4547             indexer = axis.get_indexer(new_axis)
   4548

/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
```

```
   6932         if mask.any():
   6933             if errors != "ignore":
-> 6934                 raise KeyError(f"{list(labels[mask])} not found in axis")
   6935             indexer = indexer[~mask]
   6936         return self.delete(indexer)

KeyError: "['NameHash', 'DocIDHash'] not found in axis"
```

In [133…
```
# Discretize Age column
df['AgeBins'] = pd.cut(x=df['Age'], bins=[0, 19, 29, 39, 49, 59, 99], labels=['<20','20-29','30-39','40-49','50-59','>=60'])
df.drop('Age',axis=1,inplace=True)
```

In [135…
```
iso = pd.read_csv('/content/drive/MyDrive/all tutorials/iso.csv')
```

In [136…
```
iso.drop(['name', 'alpha-2', 'country-code', 'iso_3166-2',
                'intermediate-region', 'region-code',
                'sub-region-code', 'intermediate-region-code'], axis=1, inplace=True)

iso.rename(columns={"alpha-3": "Nationality"}, inplace=True)
iso.rename(columns={"region": "Region"}, inplace=True)
iso.head()
```

Out[136]:

	Nationality	Region	sub-region
0	AFG	Asia	Southern Asia
1	ALA	Europe	Northern Europe
2	ALB	Europe	Southern Europe
3	DZA	Africa	Northern Africa
4	ASM	Oceania	Polynesia

In [137…
```
iso['Region'].value_counts()
```

Out[137]:
```
Africa      60
Americas    57
Asia        51
Europe      51
Oceania     29
Name: Region, dtype: int64
```

In [138…
```
df = pd.merge(df, iso[['Nationality','Region']], on='Nationality')
```

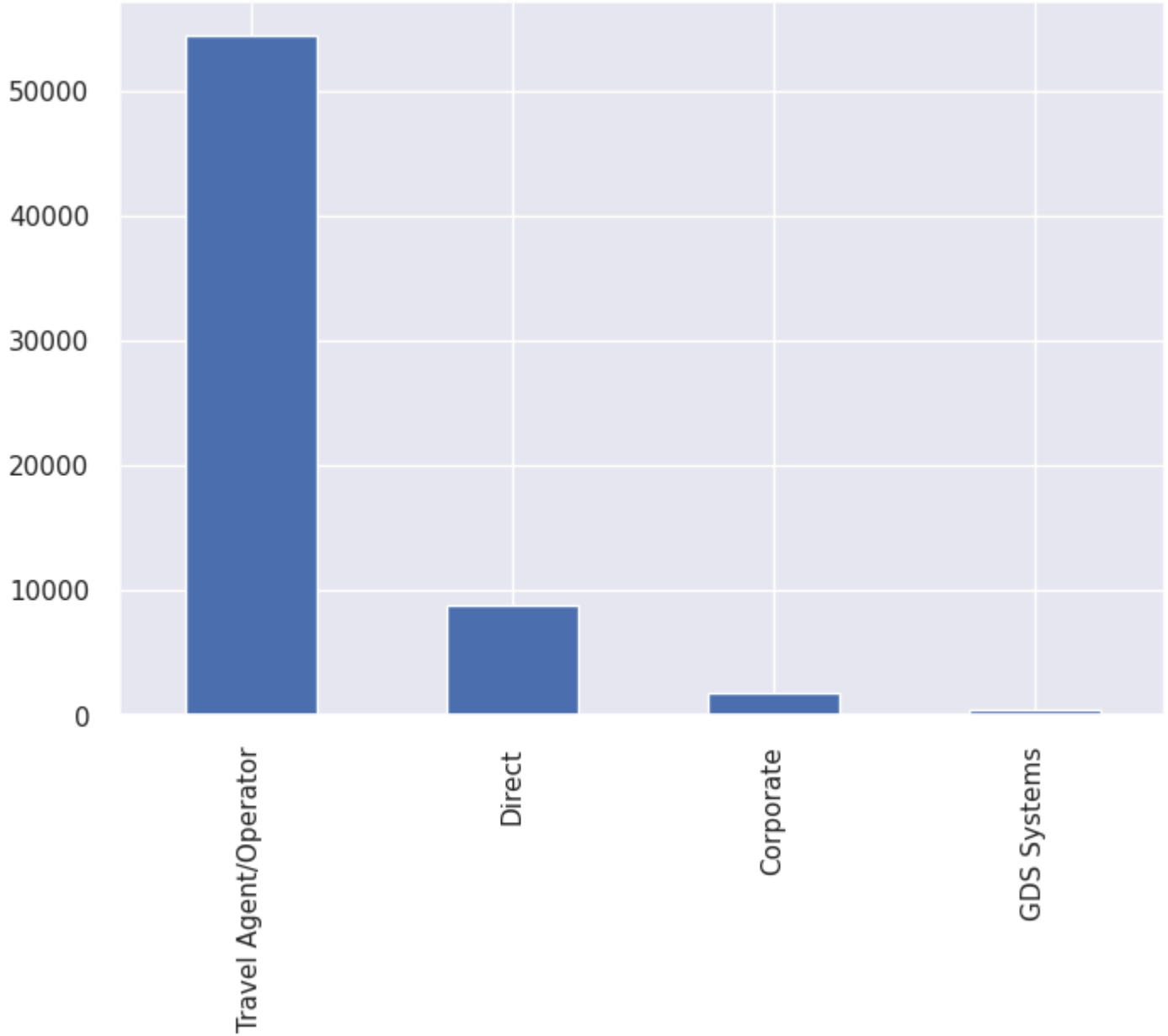
In [139…
```
df = df.drop('Nationality',axis=1)
```

In [140…
```
df.columns
```

Out[140]:
```
Index(['DaysSinceCreation', 'AverageLeadTime', 'LodgingRevenue',
       'OtherRevenue', 'BookingsCanceled', 'BookingsNoShowed',
       'BookingsCheckedIn', 'PersonsNights', 'RoomNights',
       'DistributionChannel', 'MarketSegment', 'SRHighFloor', 'SRLowFloor',
       'SRAccessibleRoom', 'SRMediumFloor', 'SRBathtub', 'SRShower', 'SRCrib',
       'SRKingSizeBed', 'SRTwinBed', 'SRNearElevator', 'SRAwayFromElevator',
       'SRNoAlcoholInMiniBar', 'SRQuietRoom', 'BookingsRealized',
       'TotalRevenue', 'AgeBins', 'Region'],
      dtype='object')
```

In [141…
```
df['DistributionChannel'].value_counts().plot(kind='bar')
```

Out[141]:
```
<Axes: >
```



In [142…
```
dfo=df.copy()
```

In [143…
```
df = df.drop(['LodgingRevenue','OtherRevenue'],axis=1)
```

In [144…
```
df=df.drop(['BookingsCanceled','BookingsNoShowed', 'BookingsCheckedIn'],axis=1)
```

In [145…
```
df.columns
```

Out[145]:
```
Index(['DaysSinceCreation', 'AverageLeadTime', 'PersonsNights', 'RoomNights',
       'DistributionChannel', 'MarketSegment', 'SRHighFloor', 'SRLowFloor',
       'SRAccessibleRoom', 'SRMediumFloor', 'SRBathtub', 'SRShower', 'SRCrib',
       'SRKingSizeBed', 'SRTwinBed', 'SRNearElevator', 'SRAwayFromElevator',
       'SRNoAlcoholInMiniBar', 'SRQuietRoom', 'BookingsRealized',
       'TotalRevenue', 'AgeBins', 'Region'],
      dtype='object')
```

In [146…
```
categorical_features = ['DistributionChannel', 'Region','AgeBins']
metric_features = ['DaysSinceCreation', 'AverageLeadTime', 'BookingsRealized', 'PersonsNights', 'RoomNights','SRHighFloor', 'SRLo
       'SRMediumFloor', 'SRBathtub', 'SRShower', 'SRCrib', 'SRKingSizeBed',
       'SRTwinBed', 'SRNearElevator', 'SRAwayFromElevator',
       'SRNoAlcoholInMiniBar', 'SRQuietRoom', 'TotalRevenue']
```

In [147…
```
data_minmax = df.copy()
scaler = MinMaxScaler()
scaled_feat = scaler.fit_transform(data_minmax[metric_features])
data_minmax[metric_features] = scaled_feat.copy()
data_minmax[metric_features].head()
```

Out[147]:

	DaysSinceCreation	AverageLeadTime	BookingsRealized	PersonsNights	RoomNights	SRHighFloor	SRLowFloor	SRAccessibleRoom	SRMediumFloor	S
0	0.125278	0.117479	1.0	0.344828	0.222222	0.0	0.0	0.0	0.0	
1	0.541883	0.169054	1.0	0.034483	0.000000	0.0	0.0	0.0	0.0	
2	0.434396	0.002865	1.0	0.137931	0.166667	0.0	0.0	0.0	0.0	
3	0.457376	0.472779	1.0	0.413793	0.277778	0.0	0.0	0.0	0.0	
4	0.189770	0.613181	1.0	0.310345	0.111111	0.0	0.0	0.0	0.0	

In [148…
```
data=df.copy()
df=data_minmax.copy()
```

In [149…
```
df=data_minmax.copy()
```

In [150…
```
df.head(5)
```

Out[150]:

	DaysSinceCreation	AverageLeadTime	PersonsNights	RoomNights	DistributionChannel	MarketSegment	SRHighFloor	SRLowFloor	SRAccessibleRoom
0	0.125278	0.117479	0.344828	0.222222	Travel Agent/Operator	Other	0.0	0.0	0.0
1	0.541883	0.169054	0.034483	0.000000	Travel Agent/Operator	Travel Agent/Operator	0.0	0.0	0.0
2	0.434396	0.002865	0.137931	0.166667	Travel Agent/Operator	Other	0.0	0.0	0.0
3	0.457376	0.472779	0.413793	0.277778	Travel Agent/Operator	Groups	0.0	0.0	0.0
4	0.189770	0.613181	0.310345	0.111111	Travel Agent/Operator	Travel Agent/Operator	0.0	0.0	0.0

5 rows × 23 columns

In [155…
```
# Define the numeric features
numeric_features = ['Age', 'DaysSinceCreation', 'AverageLeadTime', 'LodgingRevenue',
                    'OtherRevenue', 'BookingsCanceled', 'BookingsNoShowed',
                    'BookingsCheckedIn', 'PersonsNights', 'RoomNights']
```

In [157…
```
# Get the list of all columns in the dataframe
all_features = df.columns.tolist()

# Identify features that are not in either categorical or numerical features list
features_to_remove = [feature for feature in all_features if feature not in categorical_features and feature not in numeric_featu

# Remove features not in either list from the dataframe
df = df.drop(columns=features_to_remove)

# Display the updated dataframe
df.head()
```

Out[157]:

	DaysSinceCreation	AverageLeadTime	PersonsNights	RoomNights	DistributionChannel	AgeBins	Region
0	0.125278	0.117479	0.344828	0.222222	Travel Agent/Operator	50-59	Americas
1	0.541883	0.169054	0.034483	0.000000	Travel Agent/Operator	>=60	Americas
2	0.434396	0.002865	0.137931	0.166667	Travel Agent/Operator	20-29	Americas
3	0.457376	0.472779	0.413793	0.277778	Travel Agent/Operator	30-39	Americas
4	0.189770	0.613181	0.310345	0.111111	Travel Agent/Operator	>=60	Americas

In [158…
```
df = pd.get_dummies(df, columns=['AgeBins','Region','DistributionChannel'], drop_first=True, dtype=float)
dfo= pd.get_dummies(dfo, columns=['AgeBins','Region','DistributionChannel'], drop_first=True, dtype=float)
```

In [159…
```
df
```

Out[159]:

	DaysSinceCreation	AverageLeadTime	PersonsNights	RoomNights	AgeBins_20-29	AgeBins_30-39	AgeBins_40-49	AgeBins_50-59	AgeBins_>=60	Region_A
0	0.125278	0.117479	0.344828	0.222222	0.0	0.0	0.0	1.0	0.0	
1	0.541883	0.169054	0.034483	0.000000	0.0	0.0	0.0	0.0	1.0	
2	0.434396	0.002865	0.137931	0.166667	1.0	0.0	0.0	0.0	0.0	
3	0.457376	0.472779	0.413793	0.277778	0.0	1.0	0.0	0.0	0.0	
4	0.189770	0.613181	0.310345	0.111111	0.0	0.0	0.0	0.0	1.0	
...	...	...	...	...	...	...	...	...	...	
65498	0.014085	0.057307	0.068966	0.055556	0.0	0.0	1.0	0.0	0.0	
65499	0.160119	0.292264	0.206897	0.111111	0.0	0.0	0.0	0.0	1.0	
65500	0.103781	0.157593	0.275862	0.166667	0.0	0.0	1.0	0.0	0.0	
65501	0.057821	0.080229	0.034483	0.000000	0.0	1.0	0.0	0.0	0.0	
65502	0.190511	0.042980	0.275862	0.166667	0.0	0.0	0.0	1.0	0.0	

65503 rows × 14 columns

In [160…

```
from sklearn.decomposition import PCA
import numpy as np
import pandas as pd

# Fit the PCA algorithm to data
pca = PCA().fit(df,30)

# Show the variance per component
pcaevr = ['{:f}'.format(item) for item in pca.explained_variance_ratio_]
pca_ds = pd.DataFrame({'Component': range(1, len(df.columns)+1),
                       'Variance explained': pcaevr,
                       'Cumulative variance explained': np.cumsum(pca.explained_variance_ratio_)})

pca_ds
```

Out[160]:

	Component	Variance explained	Cumulative variance explained
0	1	0.192423	0.192423
1	2	0.180594	0.373018
2	3	0.162084	0.535101
3	4	0.147290	0.682391
4	5	0.086212	0.768603
5	6	0.083257	0.851861
6	7	0.058391	0.910252
7	8	0.036592	0.946844
8	9	0.018732	0.965576
9	10	0.013485	0.979061
10	11	0.012833	0.991894
11	12	0.003836	0.995730
12	13	0.002967	0.998697
13	14	0.001303	1.000000

In [161…

```
# PCA visualization
pca2 = PCA(n_components = 19)
pca2.fit(df)
X_pca2 = pca2.transform(df)
print(X_pca2.shape)

# Scatter plot of the two Principal Components by the "target" to see if there is any pattern
plt.scatter(X_pca2[:, 0], X_pca2[:, 1], edgecolor='none', alpha=0.5, cmap='viridis')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar();
```

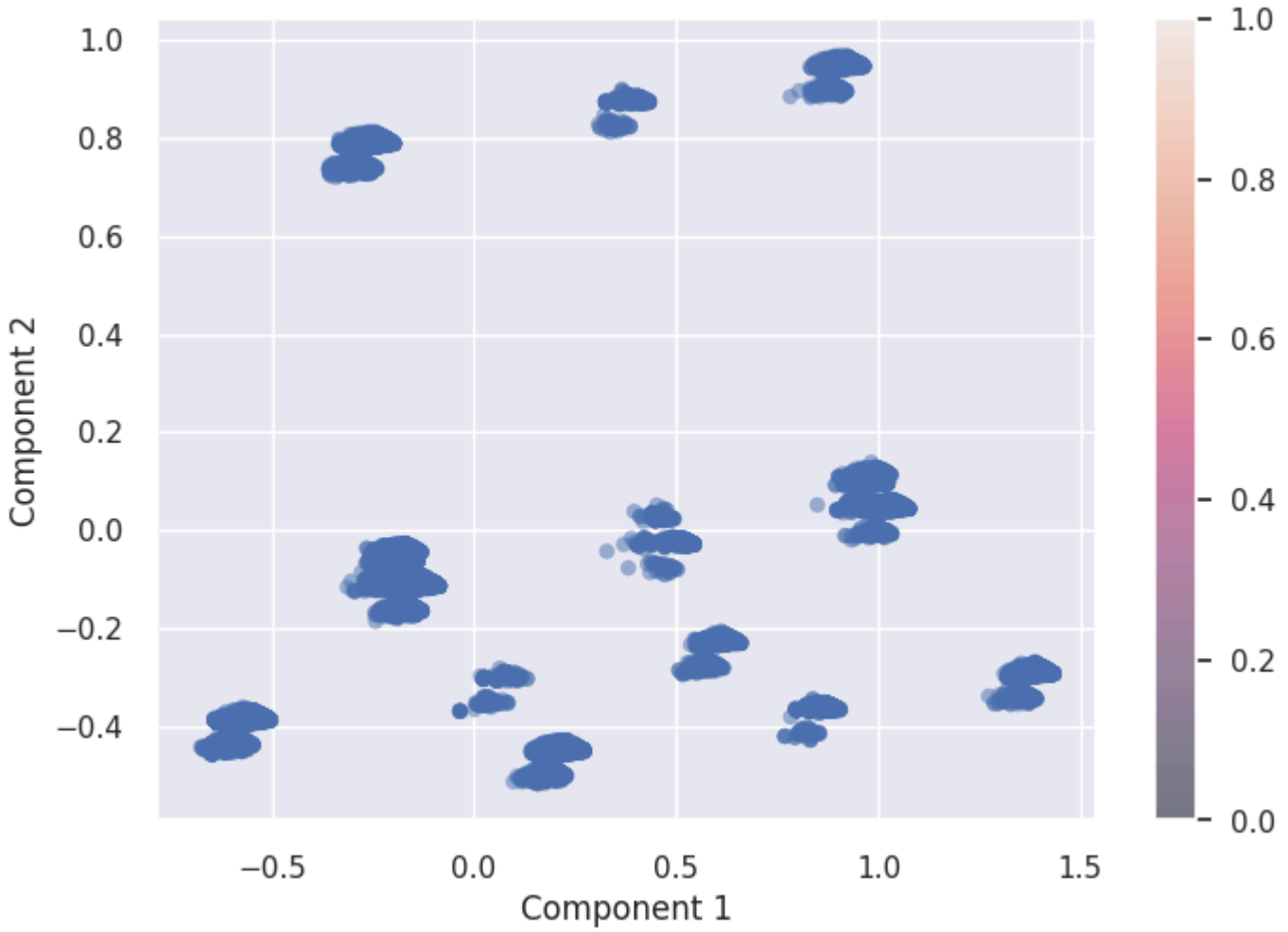
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-161-2d1def8b2d5c> in <cell line: 3>()
      1 # PCA visualization
      2 pca2 = PCA(n_components = 19)
----> 3 pca2.fit(df)
      4 X_pca2 = pca2.transform(df)
      5 print(X_pca2.shape)

/usr/local/lib/python3.10/dist-packages/sklearn/decomposition/_pca.py in fit(self, X, y)
    433         self._validate_params()
    434
--> 435         self._fit(X)
    436         return self
    437

/usr/local/lib/python3.10/dist-packages/sklearn/decomposition/_pca.py in _fit(self, X)
    510         # Call different fits for either full or truncated SVD
    511         if self._fit_svd_solver == "full":
--> 512             return self._fit_full(X, n_components)
    513         elif self._fit_svd_solver in ["arpack", "randomized"]:
    514             return self._fit_truncated(X, n_components, self._fit_svd_solver)

/usr/local/lib/python3.10/dist-packages/sklearn/decomposition/_pca.py in _fit_full(self, X, n_components)
    524                 )
    525         elif not 0 <= n_components <= min(n_samples, n_features):
--> 526             raise ValueError(
    527                 "n_components=%r must be between 0 and "
    528                 "min(n_samples, n_features)=%r with "

ValueError: n_components=19 must be between 0 and min(n_samples, n_features)=14 with svd_solver='full'
```

In [162…

```
# PCA visualization
pca2 = PCA(n_components=min(df.shape[0], df.shape[1]))
pca2.fit(df)
X_pca2 = pca2.transform(df)
print(X_pca2.shape)

# Scatter plot of the two Principal Components by the "target" to see if there is any pattern
plt.scatter(X_pca2[:, 0], X_pca2[:, 1], edgecolor='none', alpha=0.5, cmap='viridis')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar();
```

```
(65503, 14)
```

```
<ipython-input-162-a4fd08c4a41d>:8: UserWarning:

No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
```

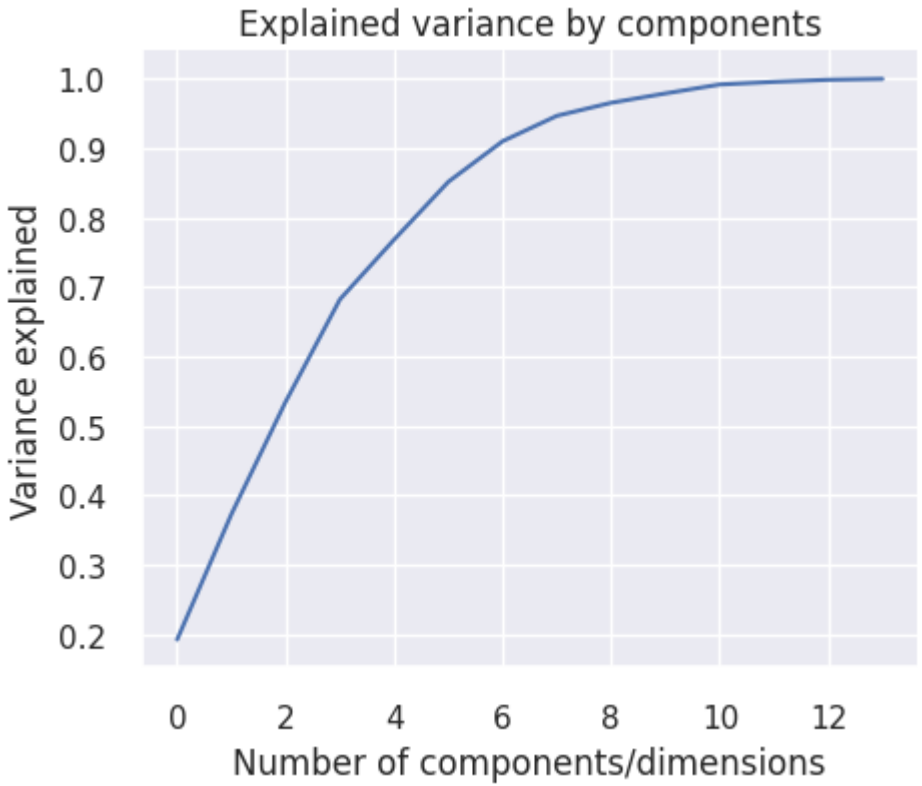


In [163…

```
# Plot the cumulative explained variance
fig , ax = plt.subplots(figsize=(5, 4))
plt.plot(np.cumsum(pca.explained_variance_ratio_))

# Decoration
sns.despine()
```

```
plt.xlabel('Number of components/dimensions')
plt.ylabel('Variance explained')
plt.rc('axes')
plt.title('Explained variance by components')
```

Out[163]:
```
Text(0.5, 1.0, 'Explained variance by components')
```



In [164…
```
top_k = 20
top_q = .50
top_pc = 20

# PCA factor loadings
selected_features = df.columns
df_c = pd.DataFrame(pca.components_, columns=selected_features).T

print("Factor Loadings for the 1. component \n(explains {0:.2f} of the variance)".format(df.iloc[0,0]))
print('='*40,'\n')
print('Top {} highest'.format(top_k))
```

```
print('-'*40)
print(df_c.iloc[:,0].sort_values(ascending=False)[:top_k], '\n')

print('Top {} lowest'.format(top_k))
print('-'*40)
print(df_c.iloc[:,0].sort_values()[:top_k])
```

```
Factor Loadings for the 1. component
(explains 0.13 of the variance)
========================================

Top 20 highest
----------------------------------------
DistributionChannel_Direct                  0.544942
AgeBins_40-49                               0.402189
AgeBins_30-39                               0.043824
Region_Europe                               0.043205
DistributionChannel_GDS Systems             0.017377
AgeBins_20-29                              -0.009045
Region_Asia                                -0.010743
PersonsNights                              -0.012553
RoomNights                                 -0.014664
DaysSinceCreation                          -0.035899
AgeBins_50-59                              -0.068145
AverageLeadTime                            -0.104147
AgeBins_>=60                               -0.372407
DistributionChannel_Travel Agent/Operator  -0.617379
Name: 0, dtype: float64

Top 20 lowest
----------------------------------------
DistributionChannel_Travel Agent/Operator  -0.617379
AgeBins_>=60                               -0.372407
AverageLeadTime                            -0.104147
AgeBins_50-59                              -0.068145
DaysSinceCreation                          -0.035899
RoomNights                                 -0.014664
PersonsNights                              -0.012553
Region_Asia                                -0.010743
AgeBins_20-29                              -0.009045
DistributionChannel_GDS Systems             0.017377
Region_Europe                               0.043205
AgeBins_30-39                               0.043824
AgeBins_40-49                               0.402189
DistributionChannel_Direct                  0.544942
Name: 0, dtype: float64
```

In [165…

```
# Apply the dimensionality reduction through PCA (20 components)
pca_reduced = PCA(n_components = 19)
pca_reduced.fit(df)
X_pca_reduced = pca_reduced.transform(df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-165-3d8813486f2a> in <cell line: 3>()
      1 # Apply the dimensionality reduction through PCA (20 components)
      2 pca_reduced = PCA(n_components = 19)
----> 3 pca_reduced.fit(df)
      4 X_pca_reduced = pca_reduced.transform(df)

/usr/local/lib/python3.10/dist-packages/sklearn/decomposition/_pca.py in fit(self, X, y)
    433         self._validate_params()
    434
--> 435         self._fit(X)
    436         return self
    437

/usr/local/lib/python3.10/dist-packages/sklearn/decomposition/_pca.py in _fit(self, X)
    510         # Call different fits for either full or truncated SVD
    511         if self._fit_svd_solver == "full":
--> 512             return self._fit_full(X, n_components)
    513         elif self._fit_svd_solver in ["arpack", "randomized"]:
    514             return self._fit_truncated(X, n_components, self._fit_svd_solver)

/usr/local/lib/python3.10/dist-packages/sklearn/decomposition/_pca.py in _fit_full(self, X, n_components)
    524                 )
    525         elif not 0 <= n_components <= min(n_samples, n_features):
--> 526             raise ValueError(
    527                 "n_components=%r must be between 0 and "
    528                 "min(n_samples, n_features)=%r with "

ValueError: n_components=19 must be between 0 and min(n_samples, n_features)=14 with svd_solver='full'
```

In [166…

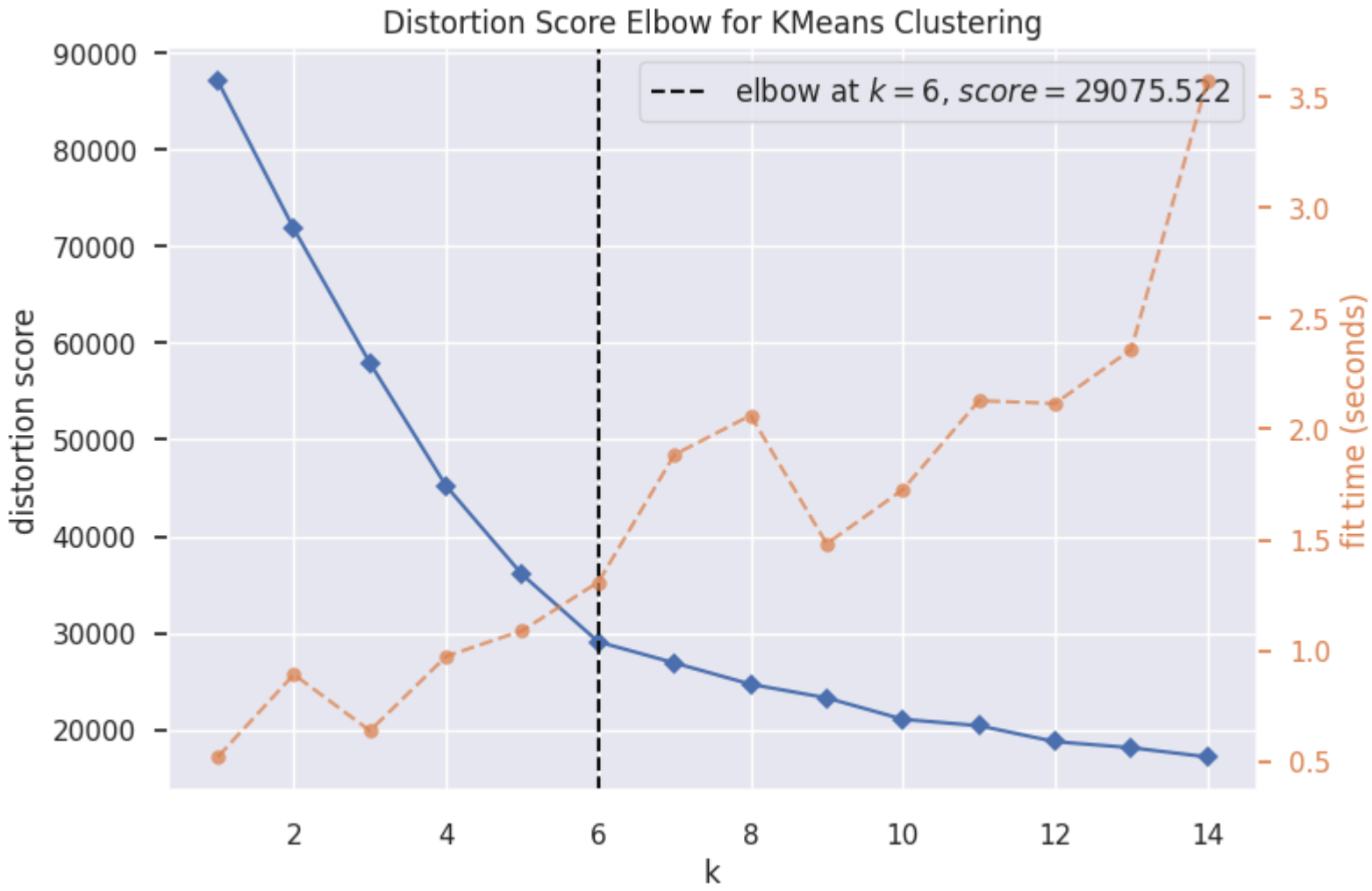
```
# Apply the dimensionality reduction through PCA (19 components)
pca_reduced = PCA(n_components=min(df.shape[0], df.shape[1]))
pca_reduced.fit(df)
X_pca_reduced = pca_reduced.transform(df)
```

In [169…

```
# Apply the Elbow method to select K
km = KMeans()
visualizer = KElbowVisualizer(km, k=(1,15), random_state=123)
visualizer.fit(X_pca_reduced)
visualizer.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```



Out[169]: <Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>

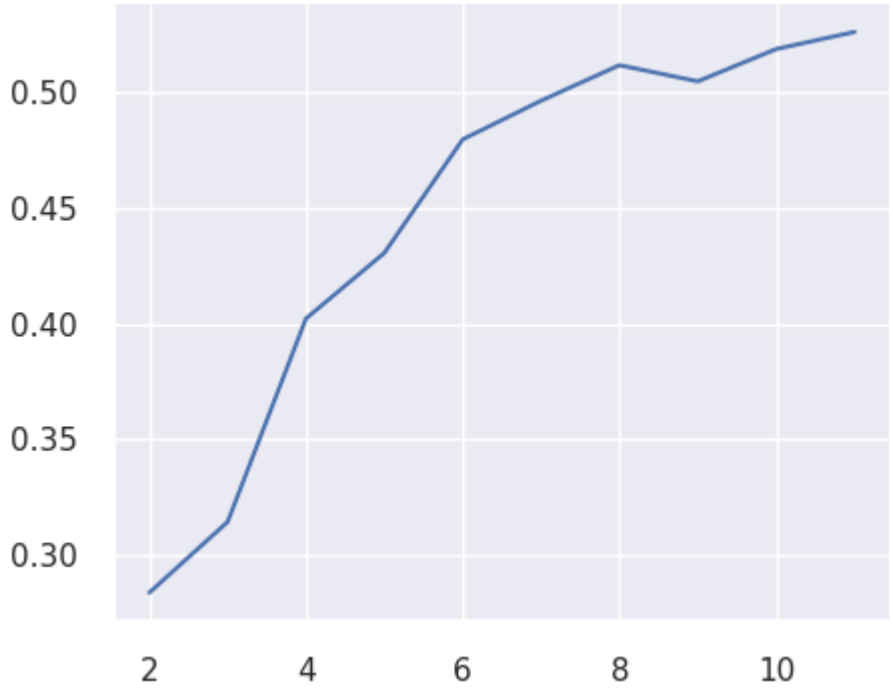
In [170…

```
# Try different setups to find the best silhouette score
sil = []
for k in range(2, 12):
    km = KMeans(n_clusters = k, random_state=42).fit(df)
    labels = km.labels_
    sil.append(silhouette_score(df, labels, metric = 'euclidean'))

# Plot Elbow Curve
fig , ax = plt.subplots(figsize=(5, 4))
```

```
plt.plot(range(2, 12), sil, 'bx-')
sns.despine()
fmt = "{x:,.2f}"
tick = ticker.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)
plt.xlabel('K')
plt.ylabel('Silhouette score')
plt.rc('axes')
ax.xaxis.set_major_locator(ticker.MaxNLocator(integer=True))
plt.title('Silhouette method - Reduced dimensionality')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-170-33600481e2ba> in <cell line: 13>()
     11 sns.despine()
     12 fmt = "{x:,.2f}"
---> 13 tick = ticker.StrMethodFormatter(fmt)
     14 ax.yaxis.set_major_formatter(tick)
     15 plt.xlabel('K')

NameError: name 'ticker' is not defined
```



In [171…

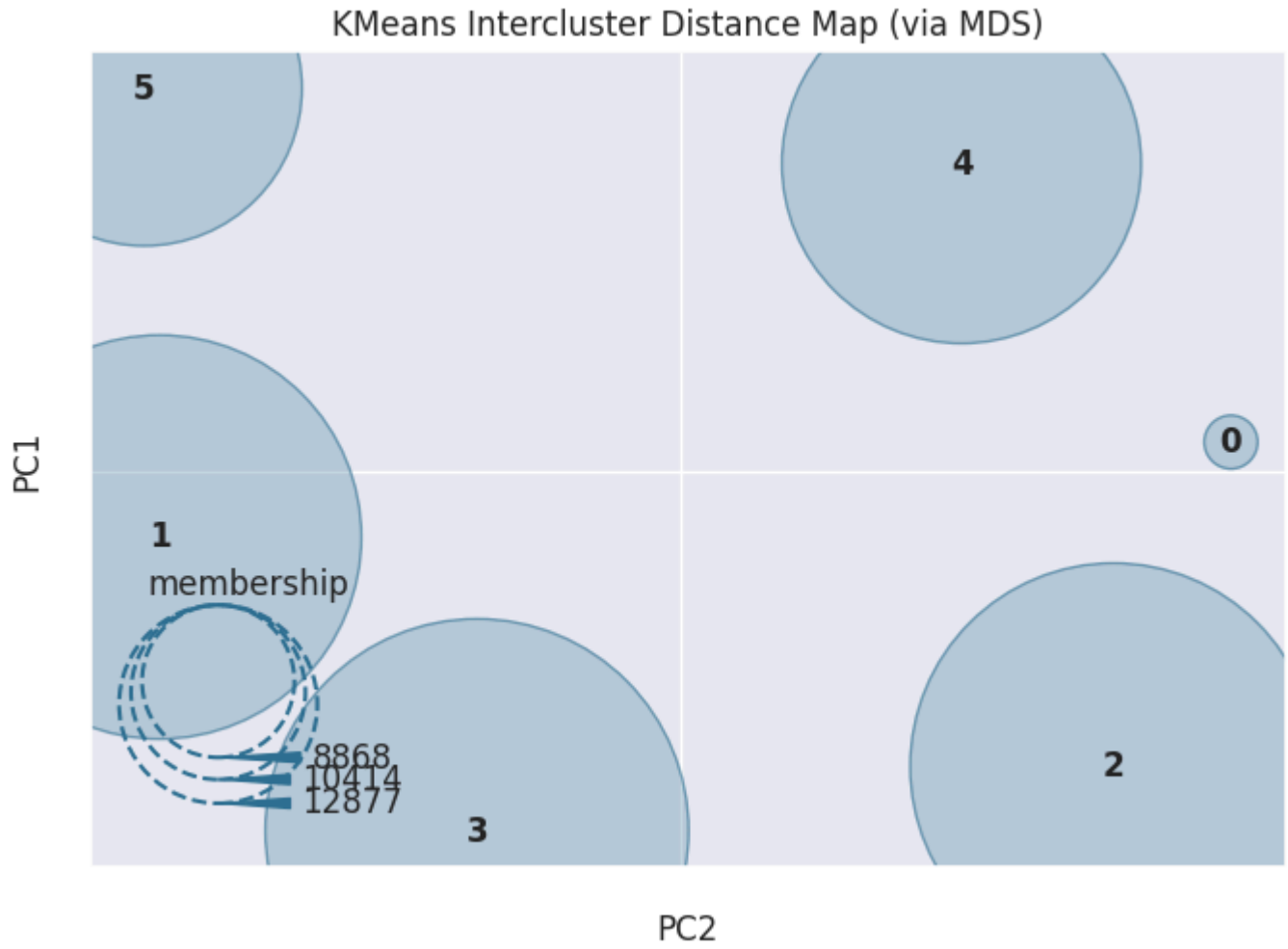
```
K=6
kmeans = KMeans(n_clusters=K, random_state=123)
allDistances = kmeans.fit_transform(X_pca_reduced)
y_kmeans = kmeans.predict(X_pca_reduced)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

In [175…

```
visualizer = InterclusterDistance(kmeans)
visualizer.fit(X_pca_reduced)
visualizer.show()
```

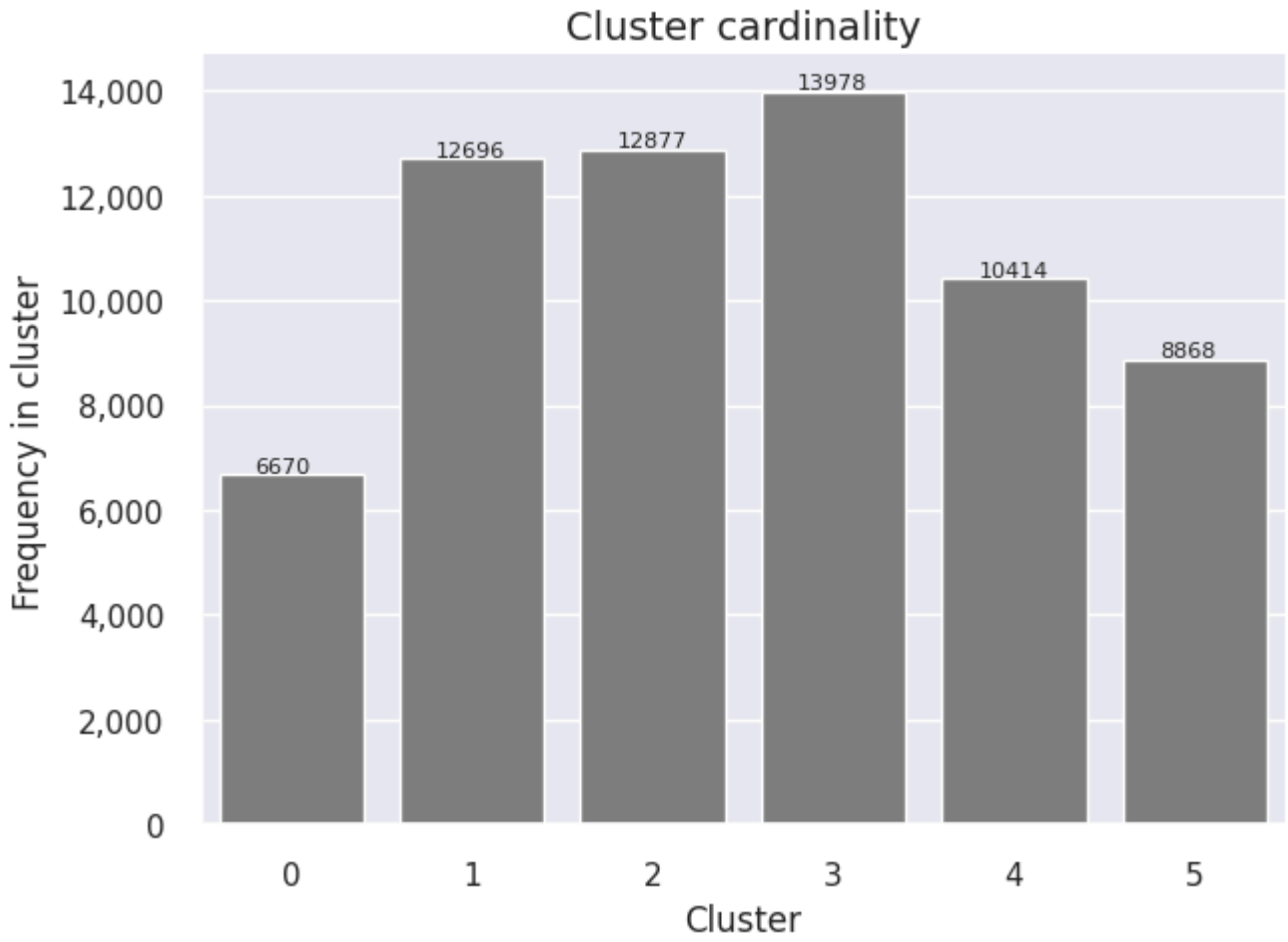
```
/usr/local/lib/python3.10/dist-packages/sklearn/manifold/_mds.py:299: FutureWarning:

The default value of `normalized_stress` will change to `'auto'` in version 1.4. To suppress this warning, manually set the valu
e of `normalized_stress`.
```



Out[175]: <Axes: title={'center': 'KMeans Intercluster Distance Map (via MDS)'}, xlabel='PC2', ylabel='PC1'>

In [176…

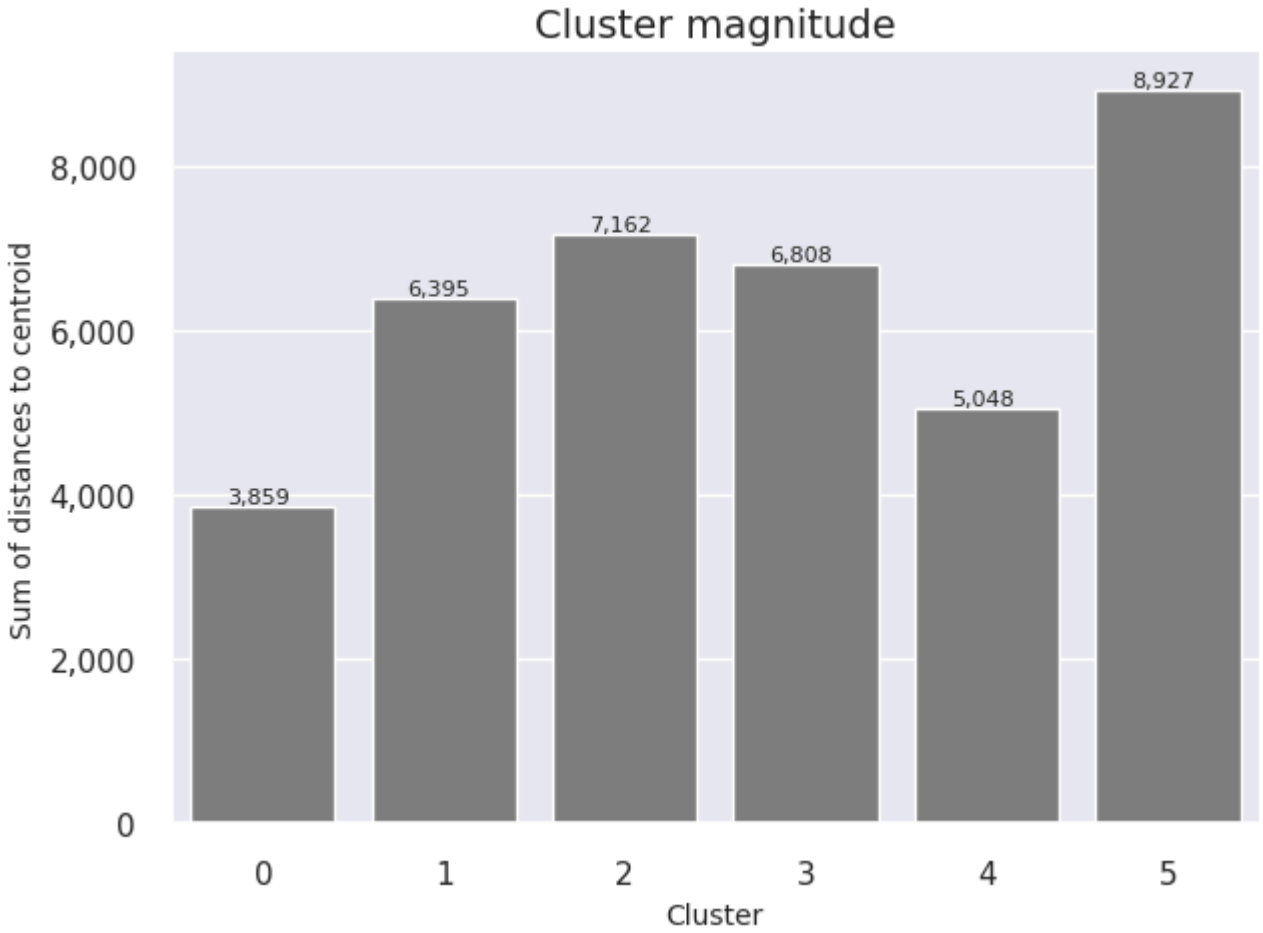
```
# Count observations per cluster
freqByCluster = df.groupby(y_kmeans).size()

# Draw
fig, ax = plt.subplots(figsize=(7,5))
g = sns.countplot(x=y_kmeans, color='grey')

# Decoration
fmt = "{x:,.0f}"
tick = ticker.StrMethodFormatter(fmt)
ax.yaxis.set_major_formatter(tick)
for index,data in enumerate(freqByCluster):
    plt.text(x=index-0.2 , y=data+50 , s=f"{data}" , fontdict=dict(fontsize=plots_barTexts_fontSize))
sns.despine()
plt.title("Cluster cardinality", fontsize=plots_Title_fontSize)
plt.xlabel("Cluster")
plt.ylabel("Frequency in cluster")
plt.rc('axes', labelsize=subPlots_label_fontSize)
```



In [177…

```
# Plot clusters magnitude

df['distanceToCentroid'] = np.min(allDistances,axis=1)
magnitude = df['distanceToCentroid'].groupby(y_kmeans).sum()
df = df.drop(columns=['distanceToCentroid'])

# Draw
fig, ax = plt.subplots(figsize=(7,5))
g = sns.barplot(x=magnitude.index, y=magnitude.values, color='grey')

# Decoration
fmt = "{x:,.0f}"
tick = ticker.StrMethodFormatter(fmt)
```

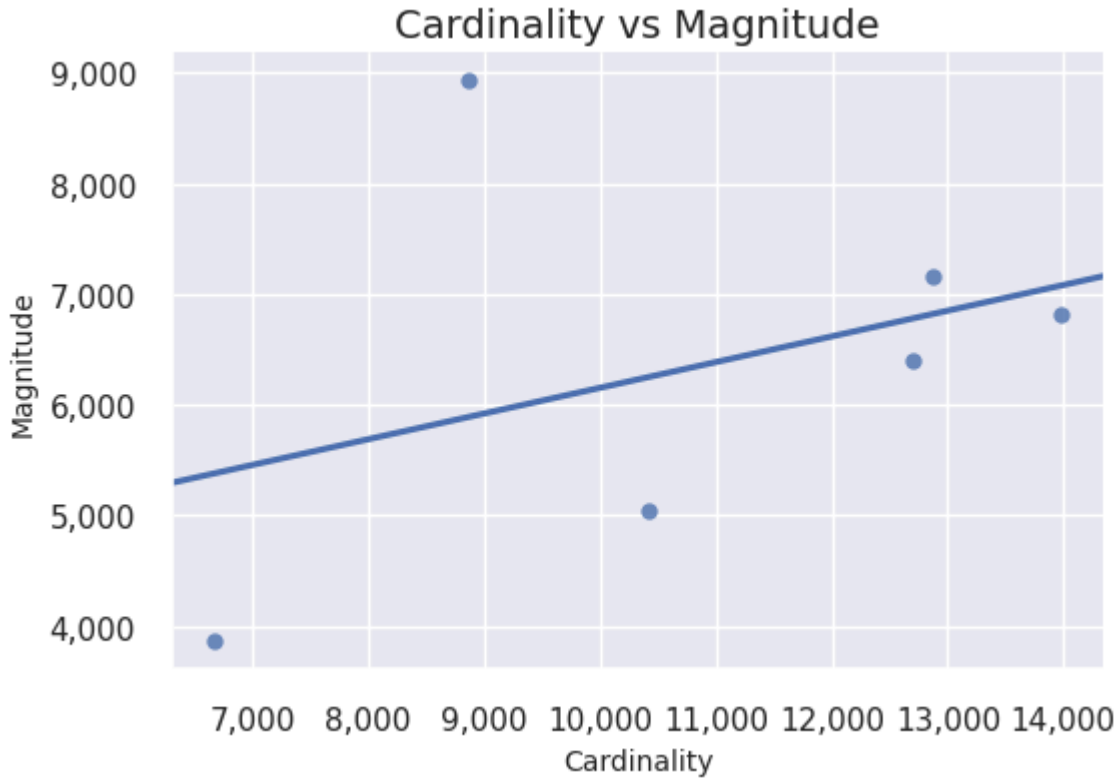
```
ax.yaxis.set_major_formatter(tick)
for index,data in enumerate(magnitude):
    plt.text(x=index-0.2 , y=data+50 , s=f"{data:,.0f}" , fontdict=dict(fontsize=plots_barTexts_fontSize))
sns.despine()
plt.title("Cluster magnitude", fontsize=plots_Title_fontSize)
plt.xlabel("Cluster")
plt.ylabel("Sum of distances to centroid")
plt.rc('axes', labelsize=subPlots_label_fontSize)
```



In [178…

```
# Plot cardinality vs magnitude

# Draw
fig, ax = plt.subplots(figsize=(6,4))
```

```
g = sns.regplot(x=freqByCluster, y=magnitude, scatter=True, seed=123,truncate=False, ci=None)

# Decoration
fmt = "{x:,.0f}"
tick = ticker.StrMethodFormatter(fmt)
ax.xaxis.set_major_formatter(tick)
ax.yaxis.set_major_formatter(tick)
sns.despine()
plt.title("Cardinality vs Magnitude", fontsize=plots_Title_fontSize)
plt.xlabel("Cardinality")
plt.ylabel("Magnitude")
plt.rc('axes', labelsize=subPlots_label_fontSize)
```



In [179…

```
means = pd.DataFrame(dfo.groupby(y_kmeans).mean())
means.transpose()
```

```
<ipython-input-179-2b0b482fc5a2>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to Fals
e. Either specify numeric_only or select only columns which should be valid for the function.
```

Out[179]:

	0	1	2	3	4	5
DaysSinceCreation	634.584558	709.059074	713.529471	729.799614	697.104859	636.048489
AverageLeadTime	79.295040	75.794056	116.539994	88.381618	68.361317	64.959386
LodgingRevenue	466.390094	398.240087	357.333364	409.353290	369.215480	458.847304
OtherRevenue	88.608873	85.477717	103.231610	93.349350	71.106306	101.582515
BookingsCanceled	0.000150	0.001339	0.000155	0.000930	0.000384	0.001015
BookingsNoShowed	0.000000	0.000315	0.000078	0.000143	0.000096	0.000338
BookingsCheckedIn	1.016192	1.041824	1.038984	1.038203	1.032264	1.059653
PersonsNights	6.745277	6.165013	6.126194	6.556804	5.740254	6.281574
RoomNights	3.340480	2.991178	3.192514	3.274646	2.990590	3.012968
SRHighFloor	0.048576	0.042691	0.033315	0.039562	0.050029	0.048602
SRLowFloor	0.000300	0.000630	0.002019	0.001574	0.000960	0.001804
SRAccessibleRoom	0.000150	0.000236	0.000388	0.000358	0.000096	0.000226
SRMediumFloor	0.000600	0.000394	0.001320	0.000501	0.000576	0.001015
SRBathtub	0.004498	0.001812	0.001786	0.001502	0.003073	0.006766
SRShower	0.001049	0.000709	0.002252	0.000858	0.001440	0.004962
SRCrib	0.006297	0.010239	0.000932	0.001789	0.028999	0.020410
SRKingSizeBed	0.382609	0.376260	0.359944	0.374660	0.406280	0.205683
SRTwinBed	0.198651	0.134531	0.208511	0.133424	0.146149	0.085927
SRNearElevator	0.000150	0.000236	0.000699	0.000215	0.000096	0.000564
SRAwayFromElevator	0.002249	0.002993	0.004038	0.004078	0.003553	0.007104
SRNoAlcoholInMiniBar	0.000000	0.000158	0.000000	0.000072	0.000000	0.000113
SRQuietRoom	0.078261	0.095542	0.091326	0.120761	0.083445	0.042174
BookingsRealized	0.999925	0.999317	0.999909	0.999583	0.999808	0.999436
TotalRevenue	554.998967	483.717804	460.564974	502.702640	440.321786	560.429818

	0	1	2	3	4	5
AgeBins_20-29	0.840630	0.000000	0.000000	0.000000	0.000000	0.089874
AgeBins_30-39	0.000000	0.000000	0.000000	0.000000	1.000000	0.196324
AgeBins_40-49	0.000000	1.000000	0.000000	0.000000	0.000000	0.276500
AgeBins_50-59	0.000000	0.000000	0.000000	1.000000	0.000000	0.245602
AgeBins_>=60	0.000000	0.000000	1.000000	0.000000	0.000000	0.161818
Region_Asia	0.028786	0.025520	0.025239	0.021319	0.023814	0.012517
Region_Europe	0.897301	0.893904	0.814708	0.895407	0.882178	0.873027
DistributionChannel_Direct	0.000000	0.000000	0.000000	0.000000	0.000000	0.999211
DistributionChannel_GDS Systems	0.005247	0.013705	0.002097	0.008728	0.012579	0.000113
DistributionChannel_Travel Agent/Operator	0.976612	0.939587	0.979654	0.959722	0.954004	0.000000

In [180…

```
(((means-dfo.mean())/dfo.mean())*100).T
```

```
<ipython-input-180-a7978d587443>:1: FutureWarning:

The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, s
pecifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warn
ing.

<ipython-input-180-a7978d587443>:1: FutureWarning:

The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, s
pecifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warn
ing.
```

Out[180]:

	0	1	2	3	4	5
DaysSinceCreation	-8.692256	2.023574	2.666801	5.007844	0.303532	-8.481617
AverageLeadTime	-5.823404	-9.981429	38.411430	4.968480	-18.809094	-22.849477
LodgingRevenue	15.700804	-1.205667	-11.353697	1.551267	-8.406014	13.829609
OtherRevenue	-2.479694	-5.925751	13.613658	2.737535	-21.742502	11.798713
BookingsCanceled	-78.651001	90.671317	-77.883385	32.434664	-45.305233	44.517415
BookingsNoShowed	-100.000000	87.612419	-53.756169	-14.797279	-42.819107	101.448518
BookingsCheckedIn	-2.174186	0.293359	0.019964	-0.055253	-0.626948	2.009655
PersonsNights	7.953378	-1.333348	-1.954612	4.936988	-8.131316	0.532138
RoomNights	6.735729	-4.425226	2.007888	4.632190	-4.444037	-3.728998
SRHighFloor	13.637674	-0.129889	-22.062628	-7.448544	17.037035	13.698503
SRLowFloor	-76.617763	-50.863414	57.449236	22.732491	-25.120259	40.694203
SRAccessibleRoom	-42.232119	-8.952797	49.612396	37.827931	-63.000599	-13.100639
SRMediumFloor	-18.162169	-46.256859	80.158260	-31.660317	-21.376272	38.495856
SRBathtub	55.881583	-37.214362	-38.096882	-47.931670	6.495631	134.490338
SRShower	-40.738122	-59.970626	27.170537	-51.522590	-18.665109	180.175525
SRCrib	-40.395568	-3.075954	-91.178922	-83.070269	174.501338	93.200385
SRKingSizeBed	7.765813	5.977702	1.382084	5.527029	14.433090	-42.067093
SRTwinBed	31.876104	-10.690632	38.422171	-11.425265	-2.977348	-42.956607
SRNearElevator	-55.361183	-29.645343	108.097242	-36.097959	-71.409554	67.873765
SRAwayFromElevator	-43.775536	-25.169856	0.959815	1.950584	-11.173193	77.613006
SRNoAlcoholInMiniBar	-100.000000	157.967076	-100.000000	17.153742	-100.000000	84.661141
SRQuietRoom	-12.250569	7.125664	2.398186	35.402613	-6.437495	-52.712587
BookingsRealized	0.027881	-0.032907	0.026317	-0.006367	0.016168	-0.021023
TotalRevenue	12.356588	-2.073904	-6.761071	1.769475	-10.859197	13.456035

	0	1	2	3	4	5
AgeBins_20-29	759.833952	-100.000000	-100.000000	-100.000000	-100.000000	-8.073123
AgeBins_30-39	-100.000000	-100.000000	-100.000000	-100.000000	438.897573	5.798452
AgeBins_40-49	-100.000000	332.420121	-100.000000	-100.000000	-100.000000	19.564066
AgeBins_50-59	-100.000000	-100.000000	-100.000000	305.440703	-100.000000	-0.422885
AgeBins_>=60	-100.000000	-100.000000	357.678871	-100.000000	-100.000000	-25.939425
Region_Asia	25.870736	11.590564	10.361614	-6.777530	4.131826	-45.267325
Region_Europe	2.629527	2.240906	-6.817105	2.412867	0.899764	-0.146914
DistributionChannel_Direct	-100.000000	-100.000000	-100.000000	-100.000000	-100.000000	638.644565
DistributionChannel_GDS Systems	-29.853288	83.209270	-71.970576	16.675563	68.158503	-98.492562
DistributionChannel_Travel Agent/Operator	17.576452	13.118999	17.942679	15.543115	14.854688	-100.000000

In [ ]: