Importing Liabraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import plotly.graph_objs as go
import plotly.express as px
plt.style.use('seaborn-dark')
plt.style.context('grayscale')
%matplotlib inline
from wordcloud import WordCloud, STOPWORDS
```

```
<ipython-input-2-26edfa6b5209>:8: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-dark')
```

Loading Dataset

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

EDA

```
df.columns = df.columns.str.replace(' ', '_')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15007 entries, 0 to 15006
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   index                        15007 non-null  int64 
 1   Activity_Period              15007 non-null  int64 
 2   Operating_Airline            15007 non-null  object
 3   Operating_Airline_IATA_Code  14953 non-null  object
 4   Published_Airline            15007 non-null  object
 5   Published_Airline_IATA_Code  14953 non-null  object
 6   GEO_Summary                  15007 non-null  object
 7   GEO_Region                   15007 non-null  object
 8   Activity_Type_Code           15007 non-null  object
 9   Price_Category_Code          15007 non-null  object
 10  Terminal                     15007 non-null  object
 11  Boarding_Area                15007 non-null  object
 12  Passenger_Count              15007 non-null  int64 
 13  Adjusted_Activity_Type_Code  15007 non-null  object
 14  Adjusted_Passenger_Count     15007 non-null  int64 
 15  Year                         15007 non-null  int64 
 16  Month                        15007 non-null  object
dtypes: int64(5), object(12)
memory usage: 1.9+ MB
```

```
df.describe()
```

	index	Activity_Period	Passenger_Count	Adjusted_Passenger_Count	Year
count	15007.00000	15007.000000	15007.000000	15007.000000	15007.000000

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

```
index                          0
Activity_Period                0
Operating_Airline              0
Operating_Airline_IATA_Code    54
Published_Airline              0
Published_Airline_IATA_Code    54
GEO_Summary                    0
GEO_Region                     0
Activity_Type_Code             0
Price_Category_Code            0
Terminal                       0
Boarding_Area                  0
Passenger_Count                0
Adjusted_Activity_Type_Code    0
Adjusted_Passenger_Count       0
Year                           0
Month                          0
dtype: int64
```

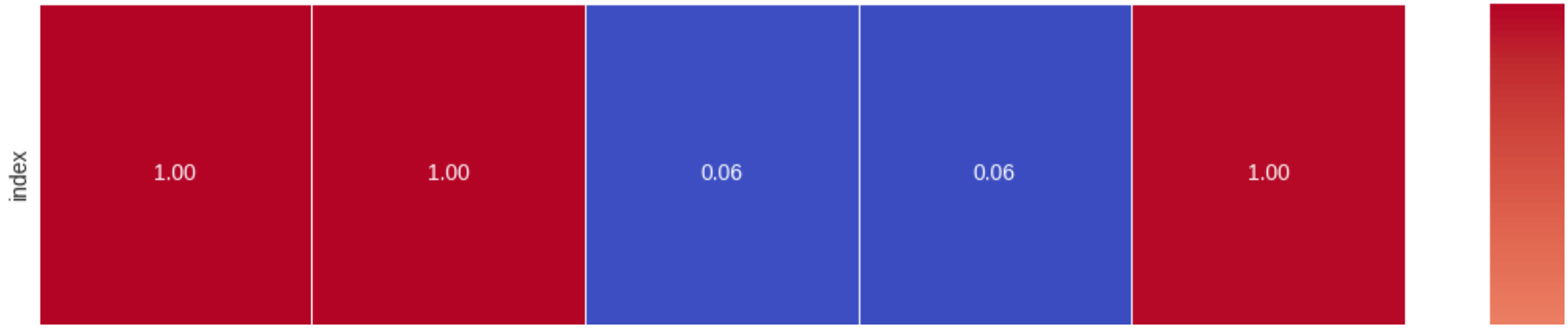
```
df['Operating_Airline_IATA_Code'] = df['Operating_Airline_IATA_Code'].fillna(method='ffill')
df['Published_Airline_IATA_Code'] = df['Published_Airline_IATA_Code'].fillna(method='ffill')
```

Data Visualisation

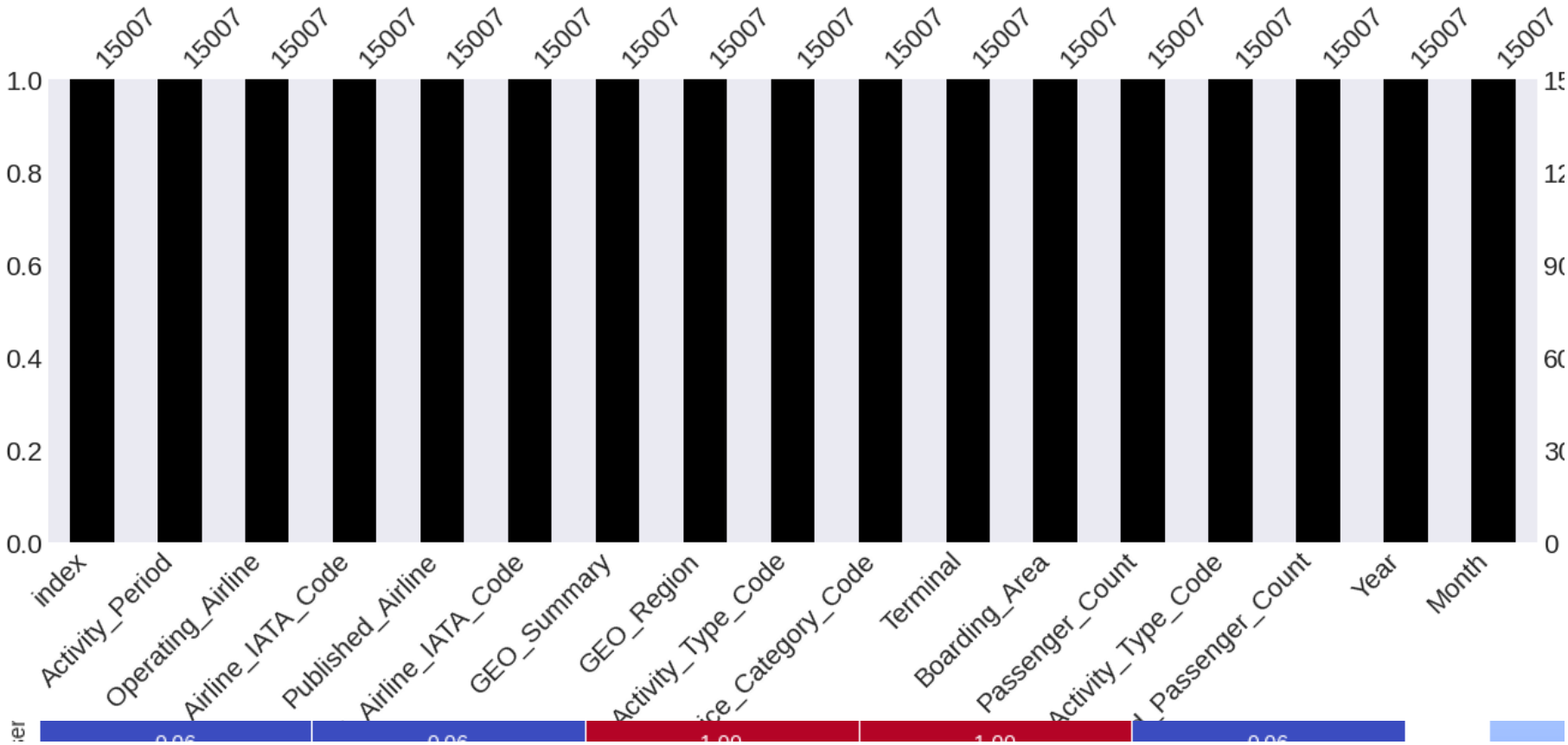
```
plt.figure(figsize=(14,14))
sns.heatmap(df.corr(),annot=True,linewidths=0.7,fmt=".2f",cmap="coolwarm")
```

```
plt.show()
```

```
<ipython-input-9-61898091fe03>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In
  sns.heatmap(df.corr(),annot=True,linewidths=0.7,fmt=".2f",cmap="coolwarm")
```



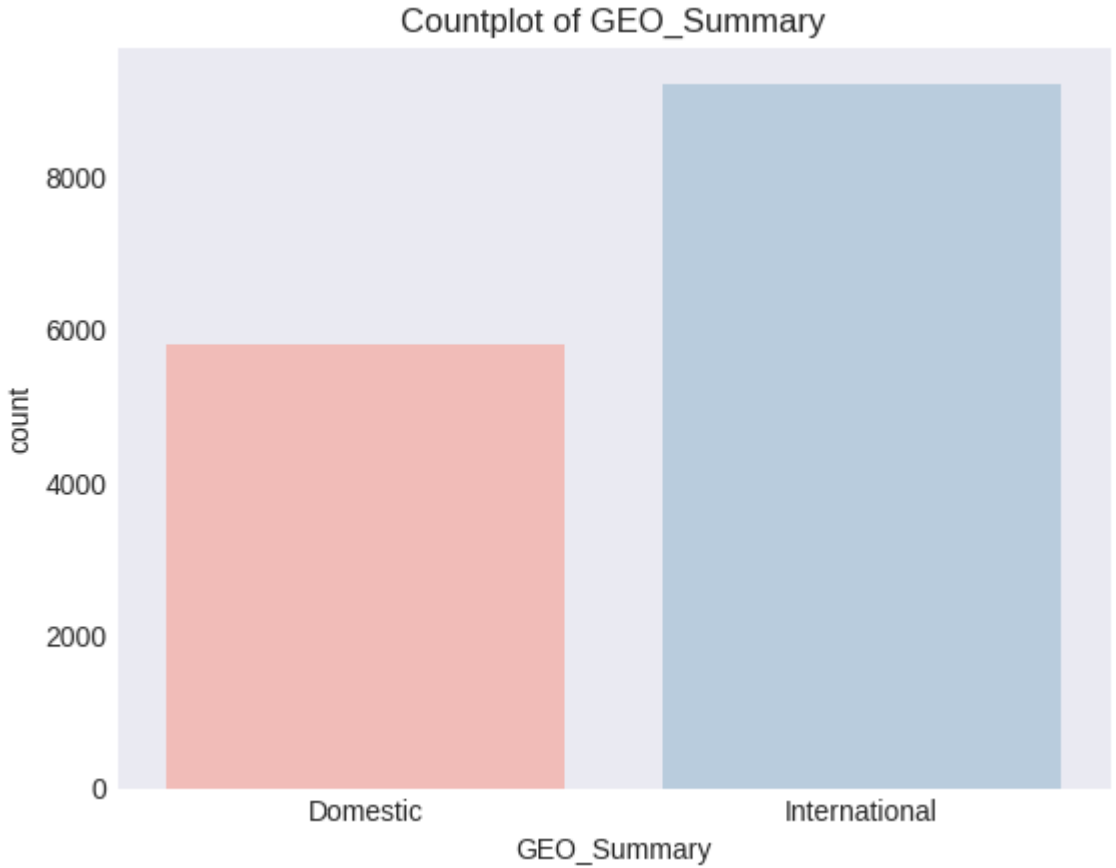

```
msno.bar(df, figsize = (16,5),color = "black")
plt.show()
```



```
df.GEO_Summary.value_counts()
```

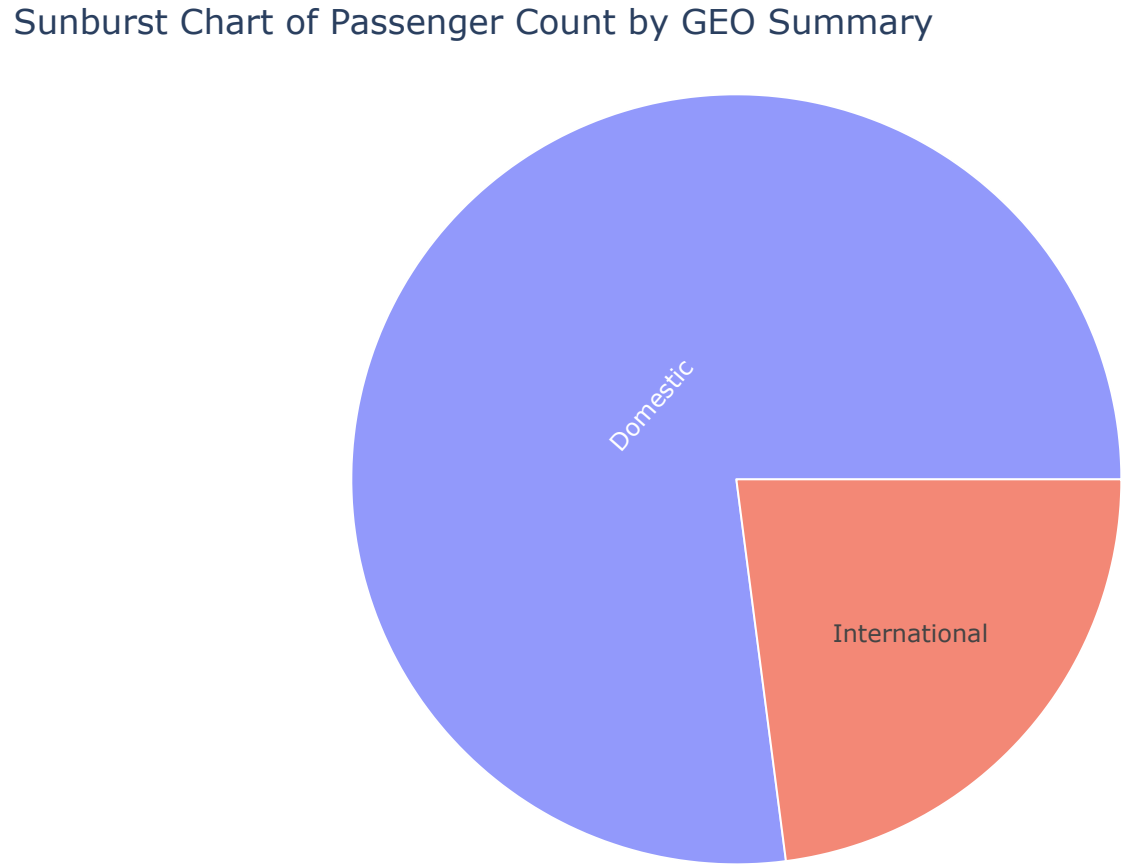
```
International    9210
Domestic         5797
Name: GEO_Summary, dtype: int64
```

```
sns.countplot(data=df,x="GEO_Summary",palette="Pastel1")
plt.title("Countplot of GEO_Summary")
plt.show()
```




```
fig = px.sunburst(df, path=['GEO_Summary'], values='Passenger_Count')
fig.update_layout(title="Sunburst Chart of Passenger Count by GEO Summary")
```

```
fig.show()
```

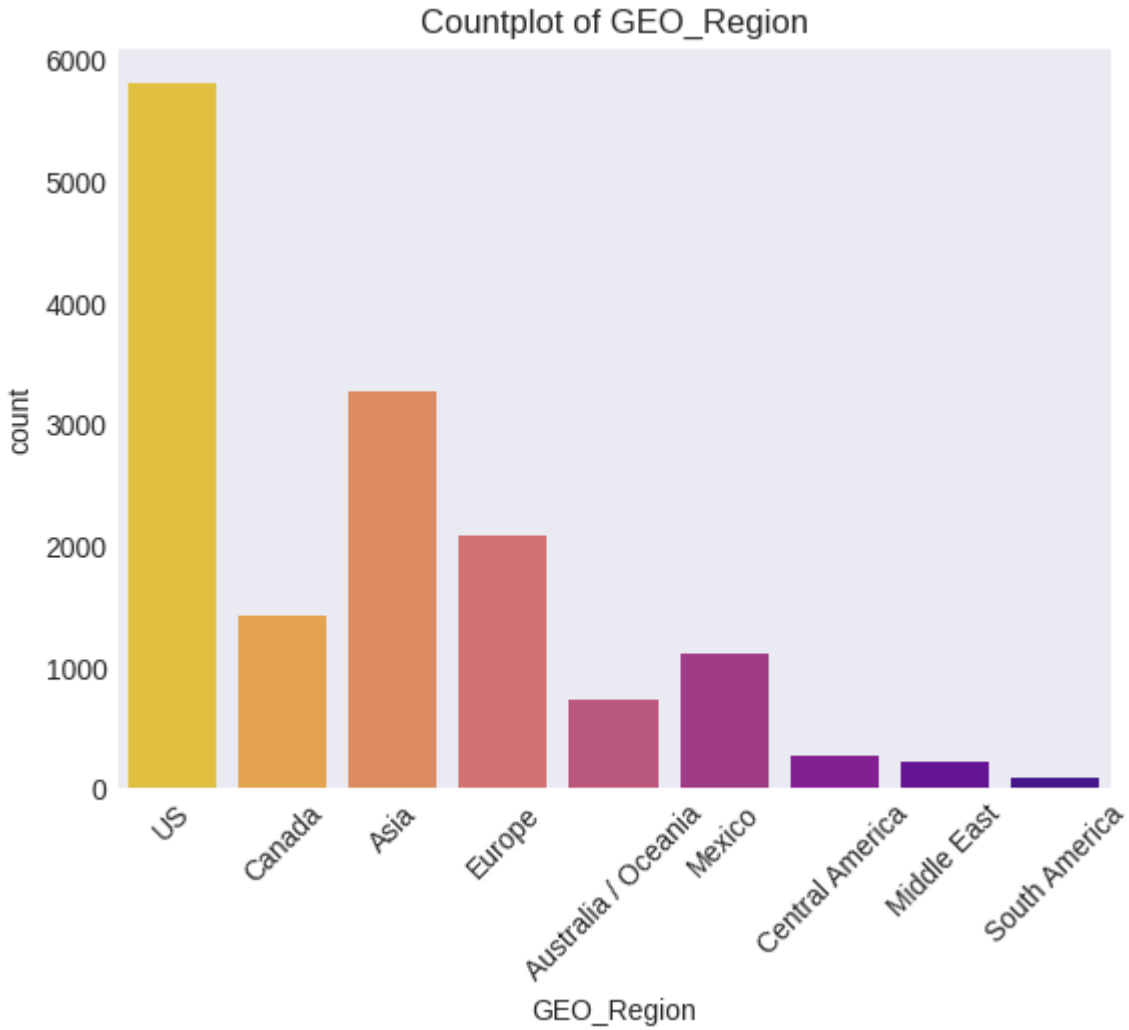



```
df.GEO_Region.value_counts()
```

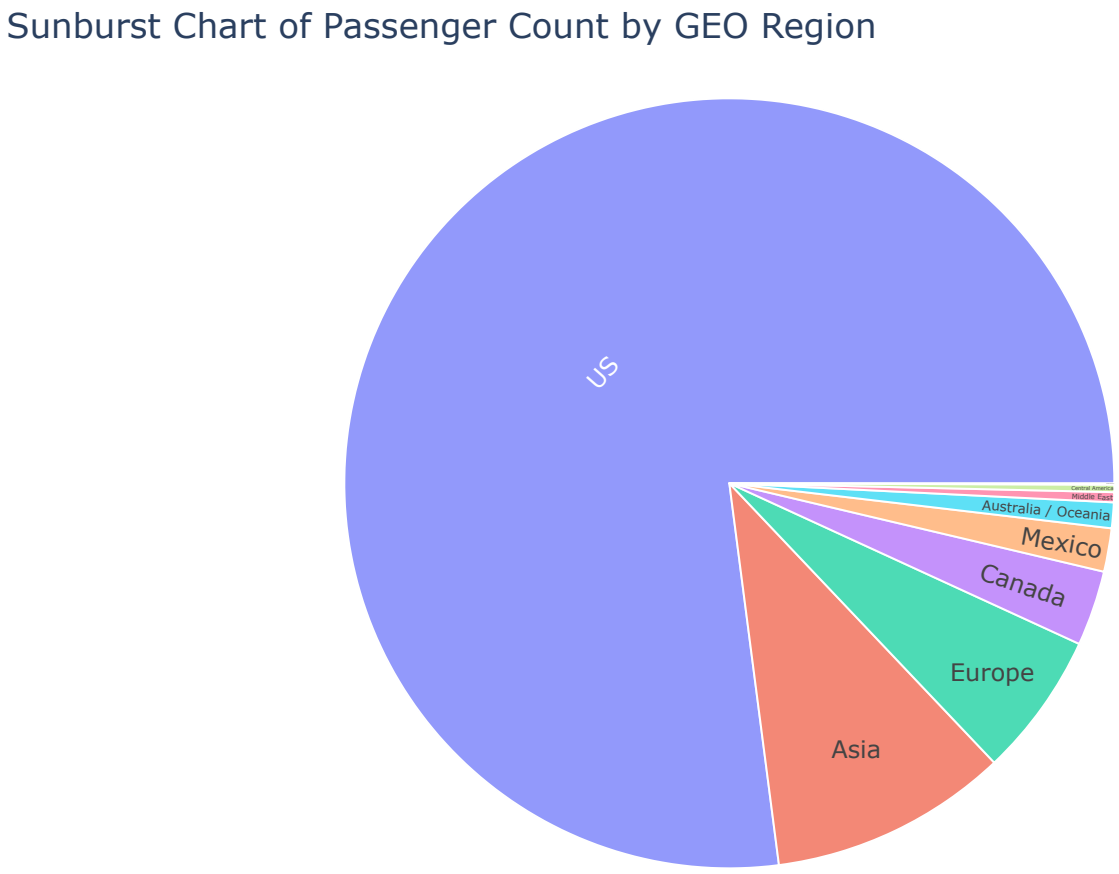
```
US                    5797
Asia                  3273
Europe                2089
Canada                1418
Mexico                1115
Australia / Oceania    737
```

```
Central America         274
Middle East             214
South America            90
Name: GEO_Region, dtype: int64
```

```
sns.countplot(data=df,x="GEO_Region",palette="plasma_r")
plt.title("Countplot of GEO_Region")
plt.xticks(rotation=45)
plt.show()
```



```
fig = px.sunburst(df, path=['GEO_Region'], values='Passenger_Count')
fig.update_layout(title="Sunburst Chart of Passenger Count by GEO Region")
fig.show()
```



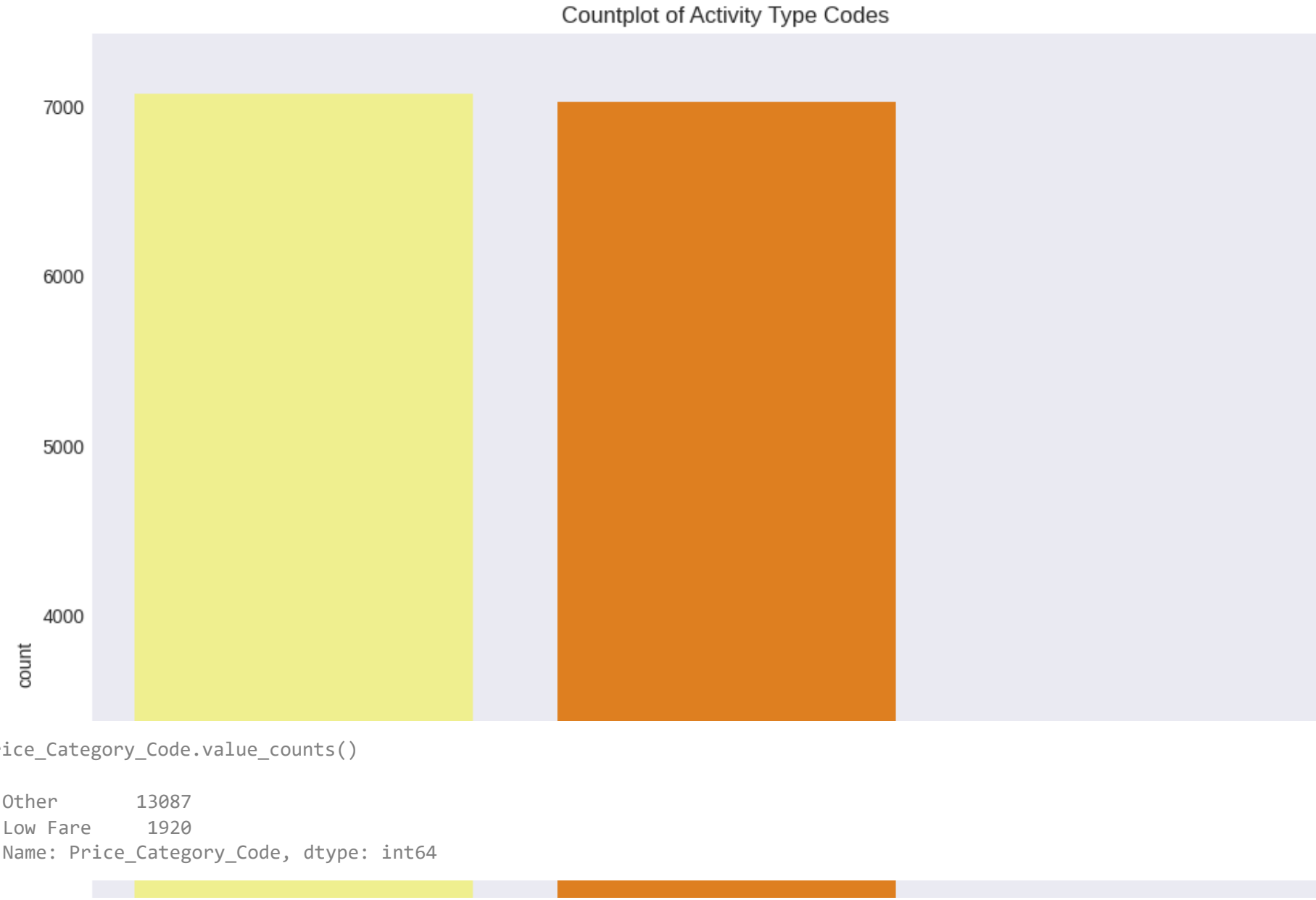

```
from wordcloud import WordCloud, STOPWORDS
text = ' '.join(df['GEO_Region'])
plt.rcParams['figure.figsize'] = (12,12)
wordcloud = WordCloud(background_color = 'white',colormap='vlag', width = 1200,  height = 1200, max_words = 121).generate(te
```

```
plt.imshow(wordcloud)
plt.axis('off')
plt.title("Word Cloud of GEO Regions", fontsize=20)
plt.show()
```


```
df.Activity_Type_Code.value_counts()
```

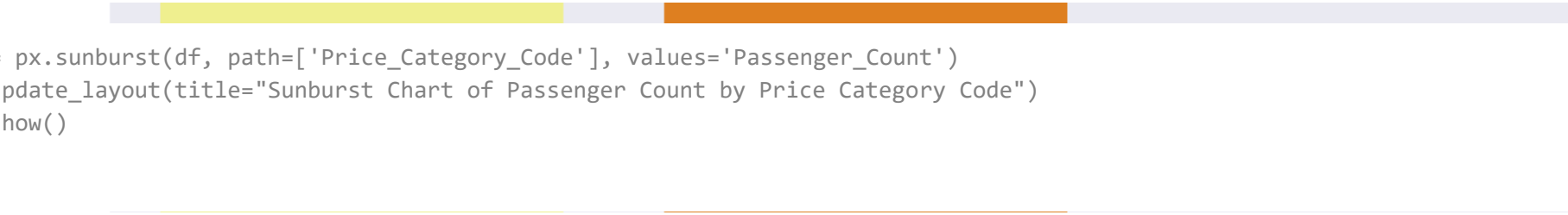
```
Deplaned          7071
Enplaned          7016
Thru / Transit     920
Name: Activity_Type_Code, dtype: int64
```

```
sns.countplot(data=df,x="Activity_Type_Code",palette="afmhot_r")
plt.title("Countplot of Activity Type Codes")
plt.show()
```



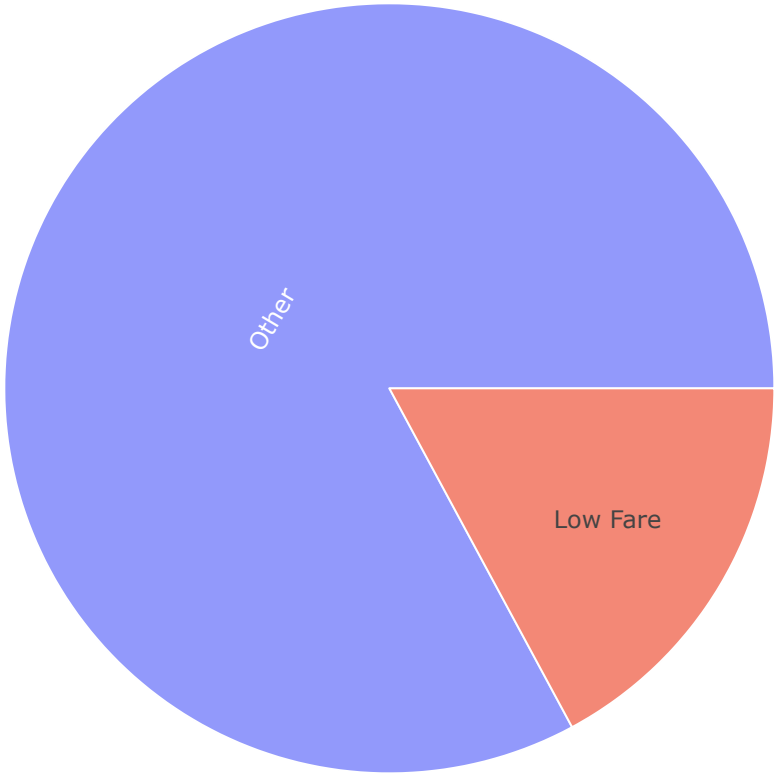

```
df.Price_Category_Code.value_counts()
```

```
Other       13087
Low Fare     1920
Name: Price_Category_Code, dtype: int64
```

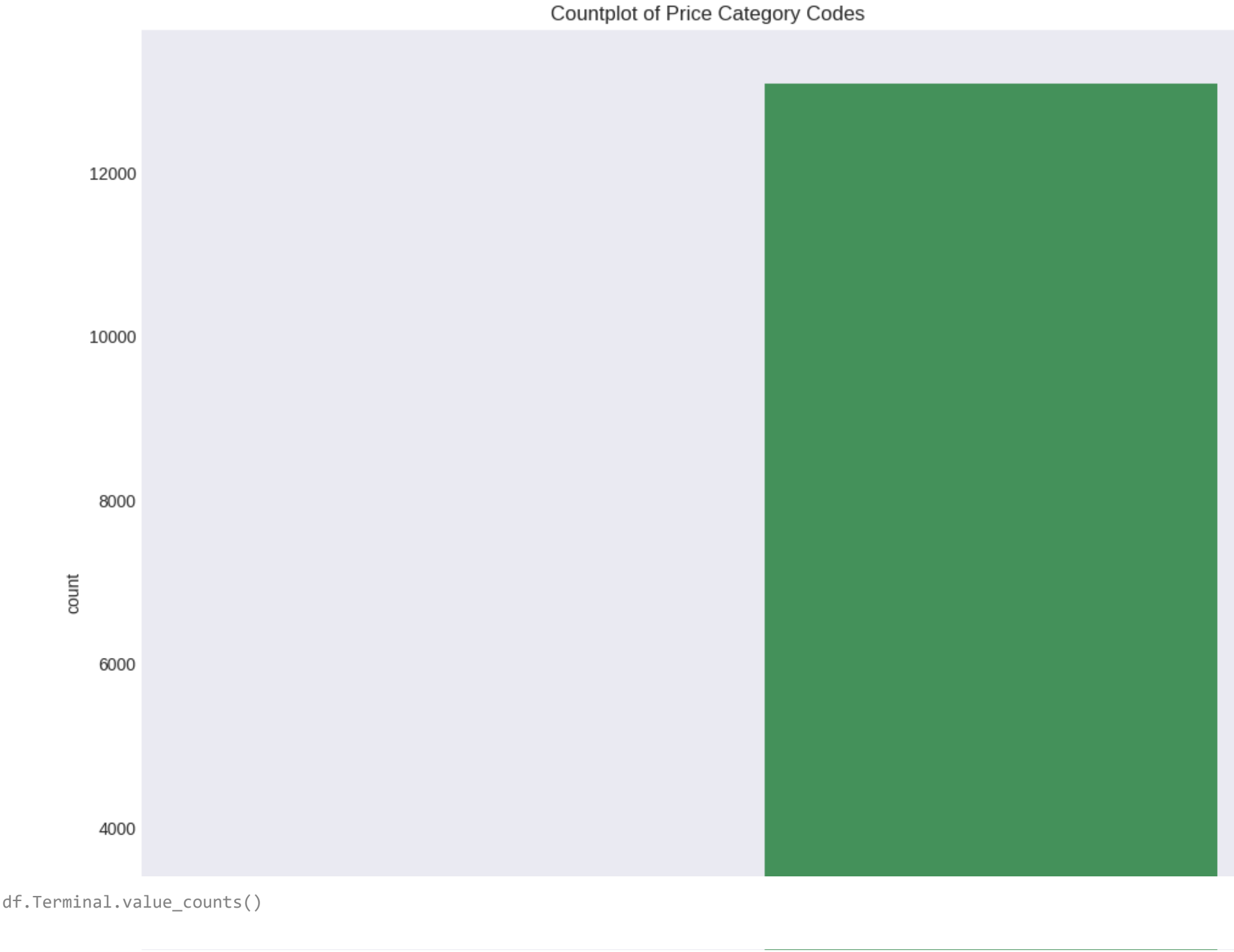


```
fig = px.sunburst(df, path=['Price_Category_Code'], values='Passenger_Count')
fig.update_layout(title="Sunburst Chart of Passenger Count by Price Category Code")
fig.show()
```

Sunburst Chart of Passenger Count by Price Category Code




```
sns.countplot(data=df,x="Price_Category_Code",palette="YlGn")
plt.title("Countplot of Price Category Codes")
plt.show()
```

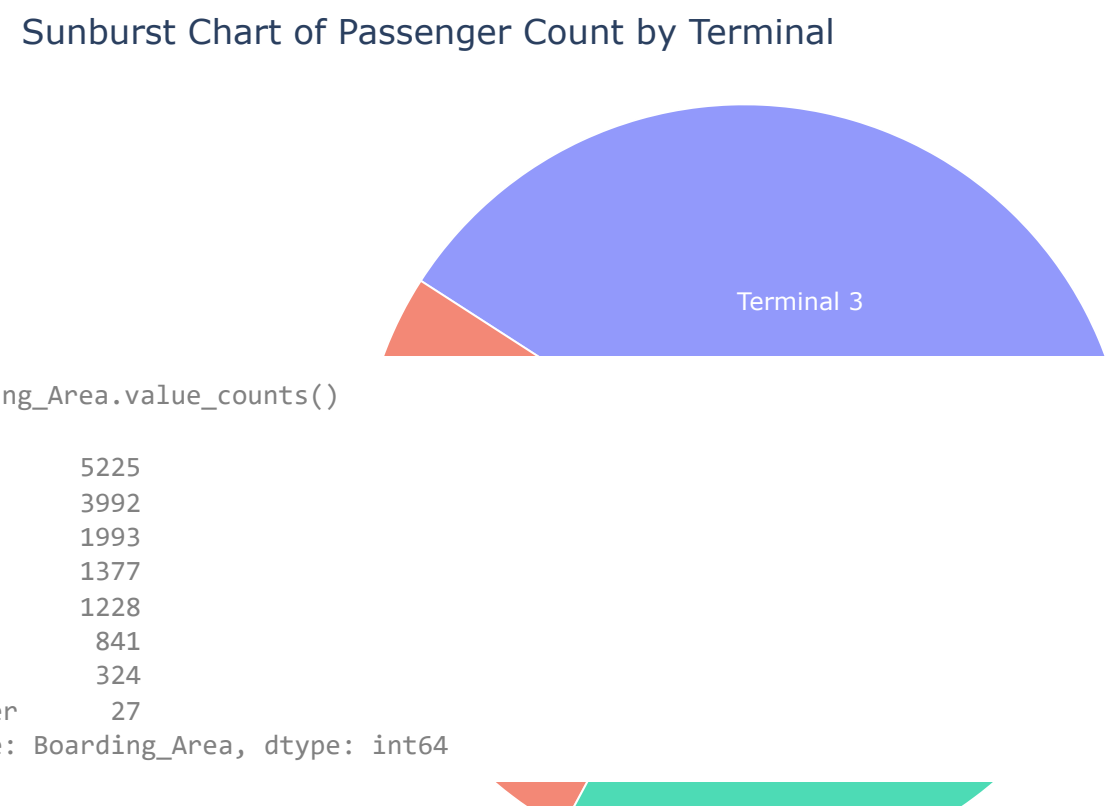



```
df.Terminal.value_counts()
```

```
International    9197
Terminal 1       3241
Terminal 3       2218
Terminal 2        324
Other              27
Name: Terminal, dtype: int64
```

```
sns.countplot(data=df,x="Terminal",palette="magma_r")
plt.title("Countplot of Terminal")
plt.show()
```


```
fig = px.sunburst(df, path=['Terminal'], values='Passenger_Count')
fig.update_layout(title="Sunburst Chart of Passenger Count by Terminal")
fig.show()
```

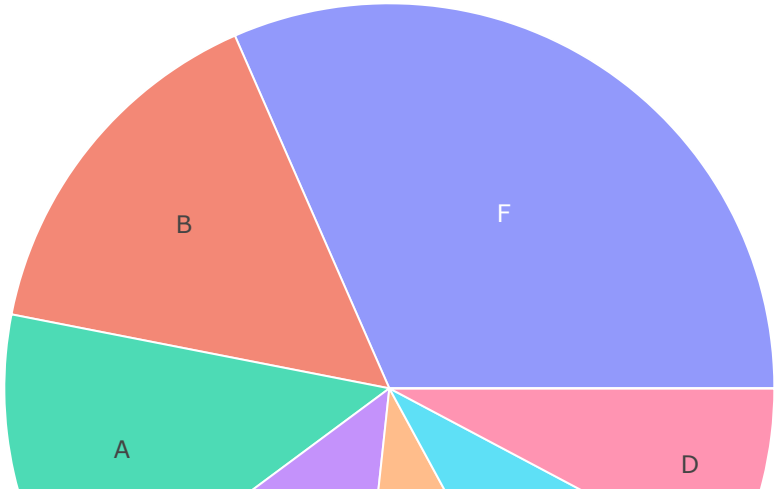



```
df.Boarding_Area.value_counts()
```

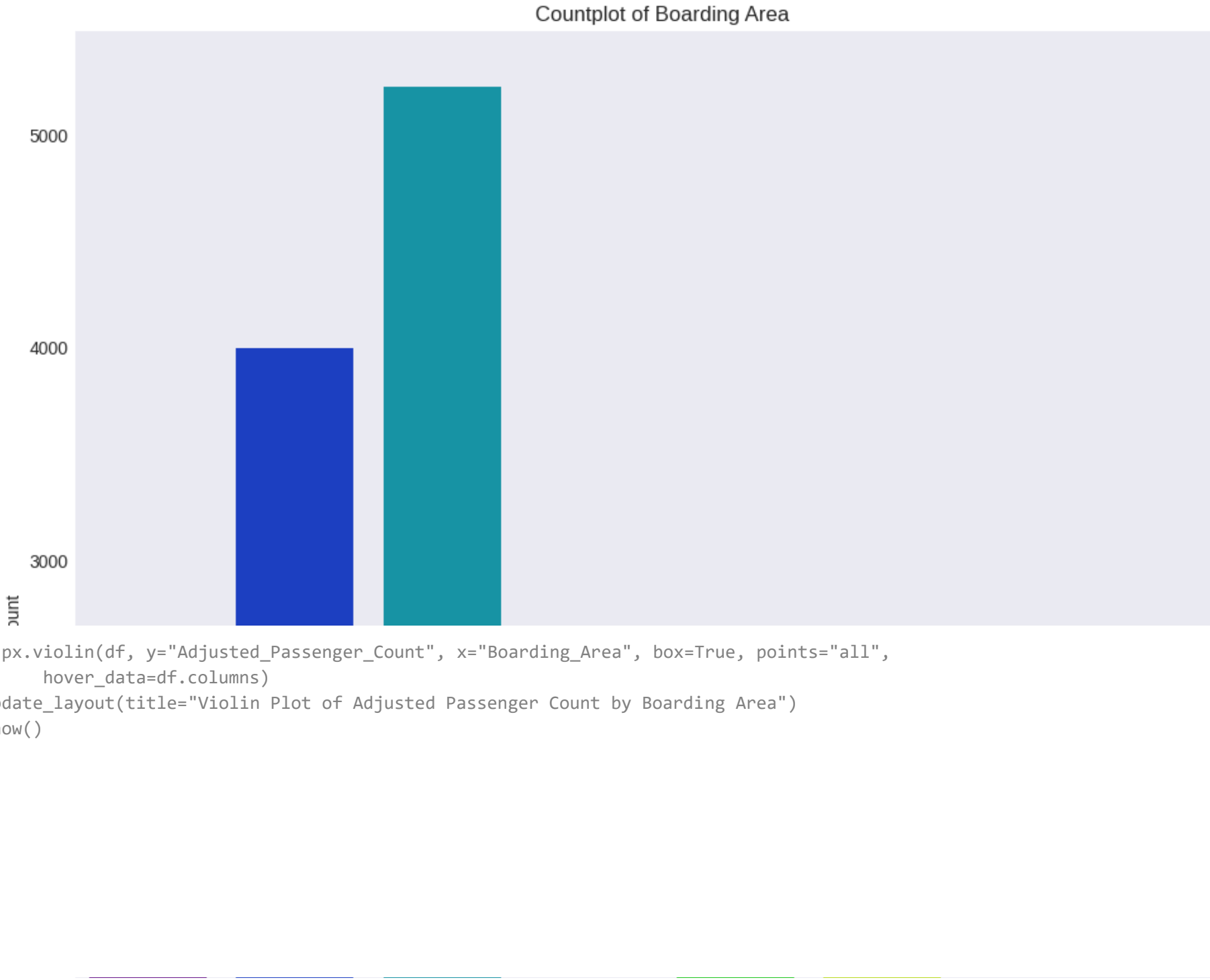
```
A        5225
G        3992
B        1993
F        1377
C        1228
E         841
D         324
Other      27
Name: Boarding_Area, dtype: int64
```

```
fig = px.sunburst(df, path=['Boarding_Area'], values='Passenger_Count')
fig.update_layout(title="Sunburst Chart of Passenger Count by Boarding Area")
fig.show()
```

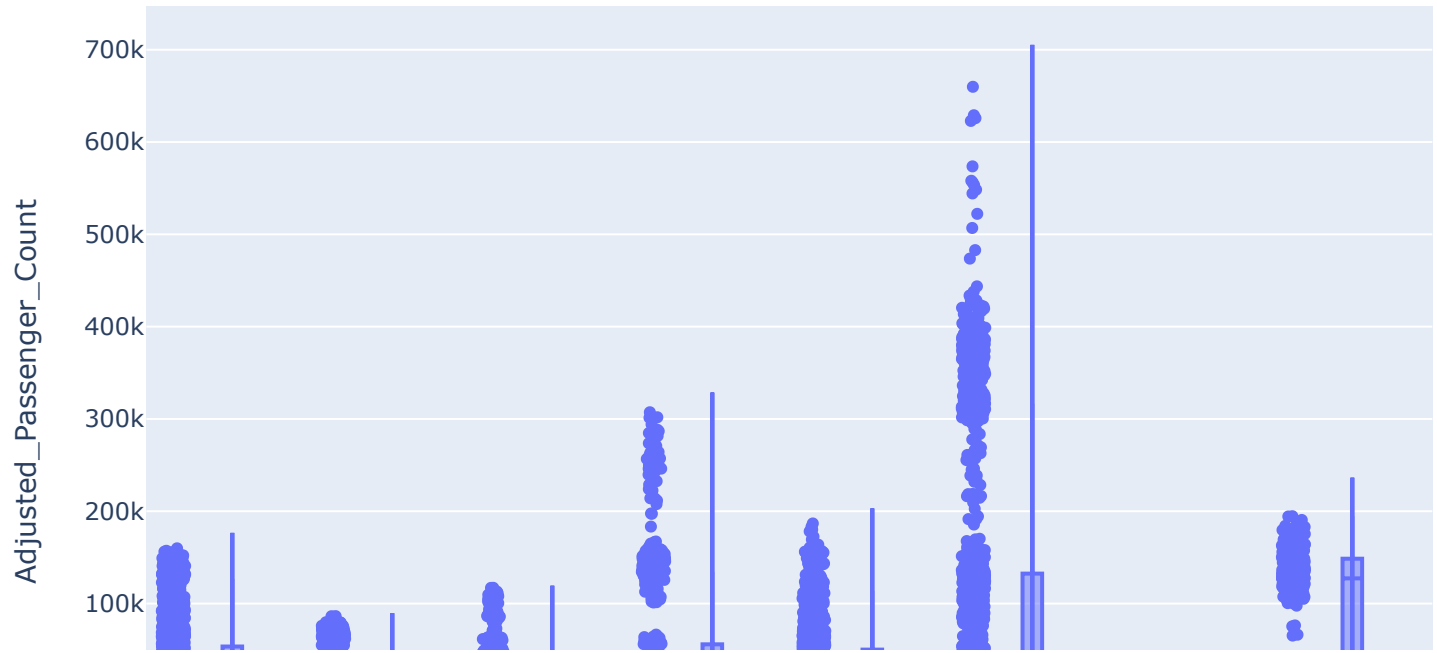
Sunburst Chart of Passenger Count by Boarding Area




```
sns.countplot(data=df,x="Boarding_Area",palette="nipy_spectral")
plt.title("Countplot of Boarding Area")
plt.show()
```




```
fig = px.violin(df, y="Adjusted_Passenger_Count", x="Boarding_Area", box=True, points="all",
          hover_data=df.columns)
fig.update_layout(title="Violin Plot of Adjusted Passenger Count by Boarding Area")
fig.show()
```

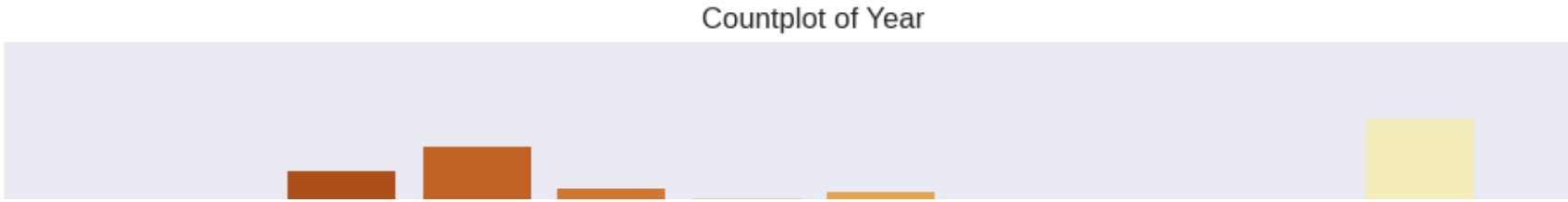



```
df.Year.value_counts()
```

```
2015    1460
2008    1433
2007    1409
2009    1393
2011    1390
2010    1383
2012    1378
2006    1369
2014    1368
2013    1358
2005     695
2016     371
Name: Year, dtype: int64
```

```
sns.countplot(data=df,x="Year",palette="YlOrBr_r")
plt.title("Countplot of Year")
```

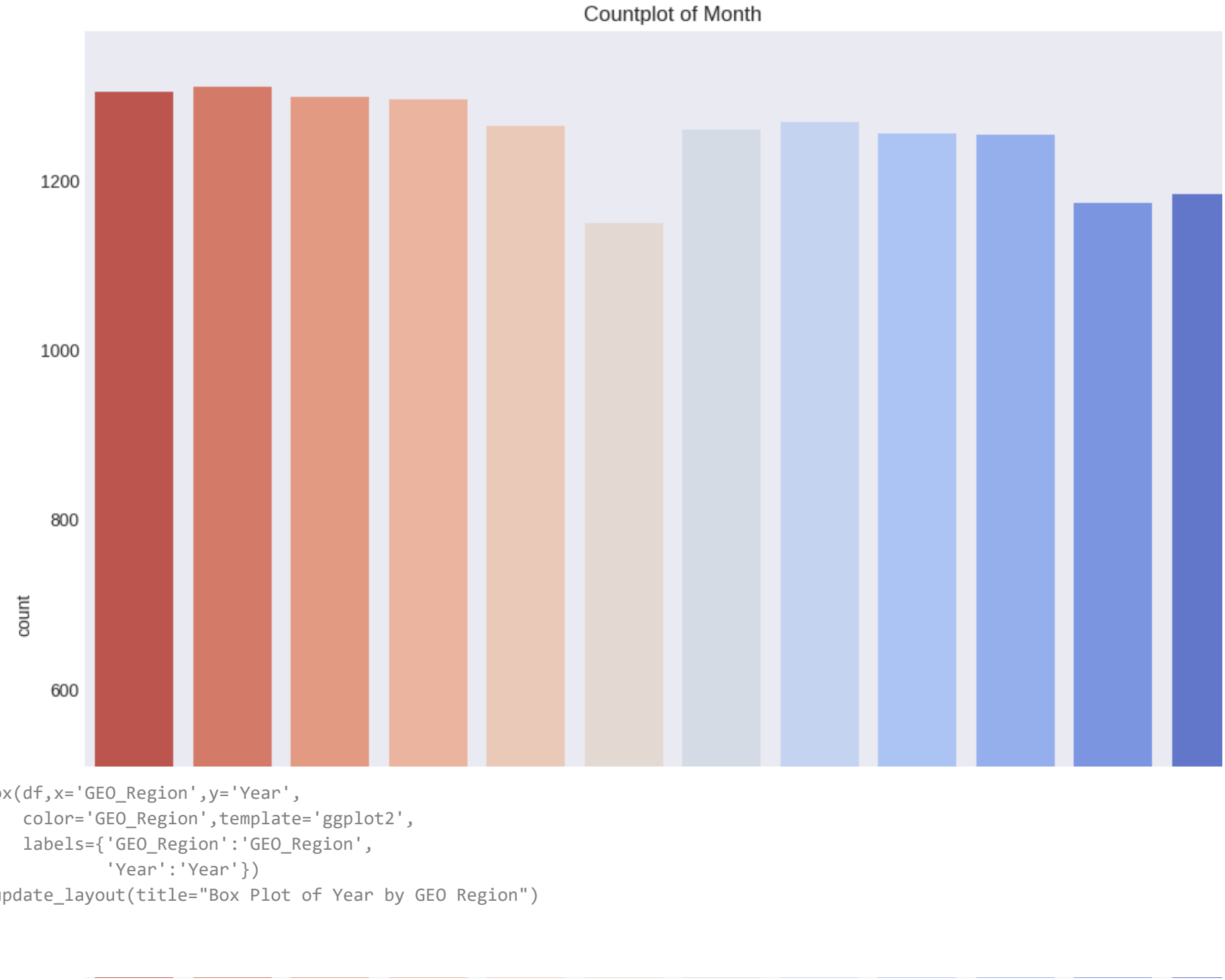
```
plt.show()
```



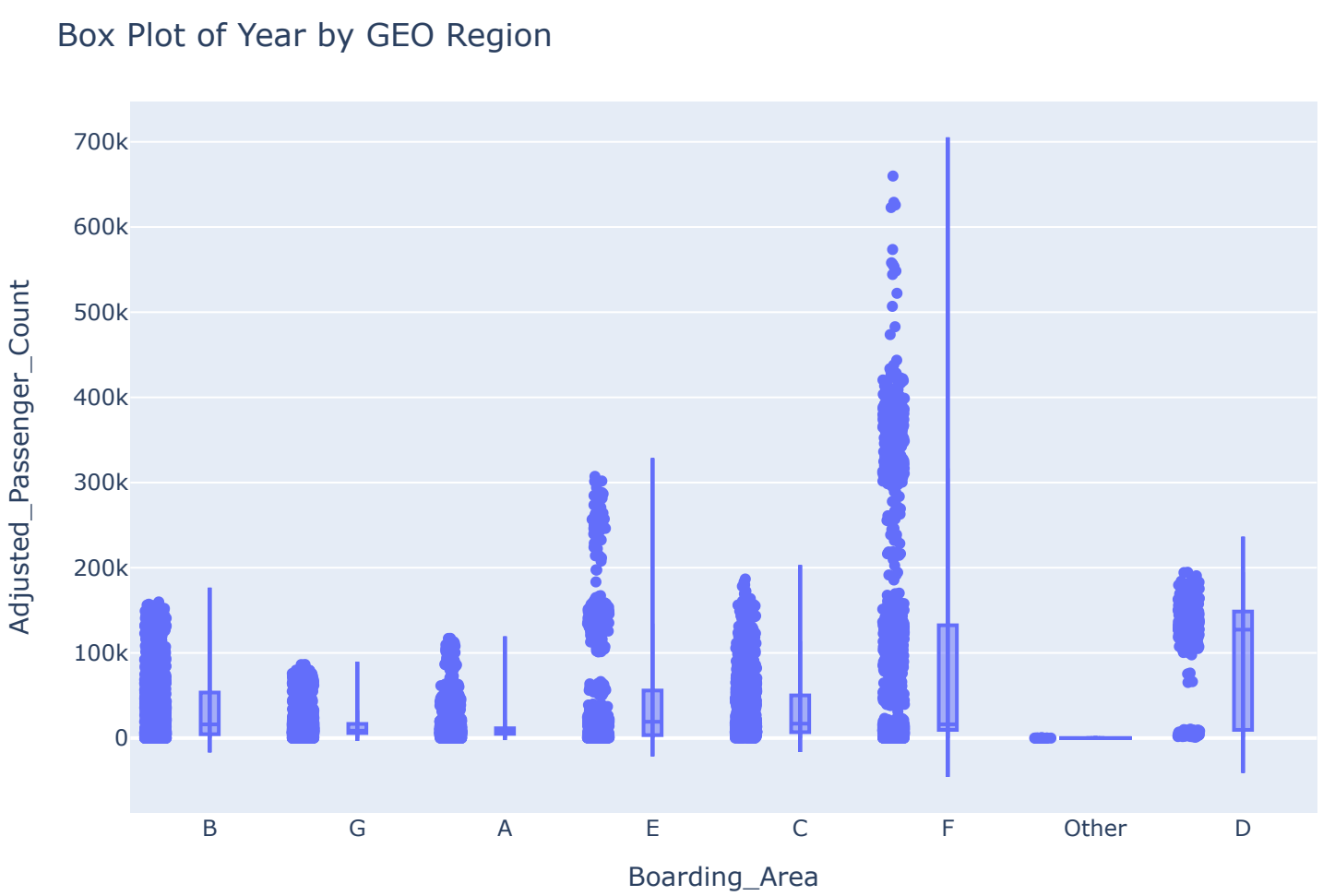

```
df.Month.value_counts()
```

```
August       1310
July         1303
September    1297
October      1295
January      1268
November     1263
December     1259
February     1255
March        1253
June         1183
May          1172
April        1149
Name: Month, dtype: int64
```

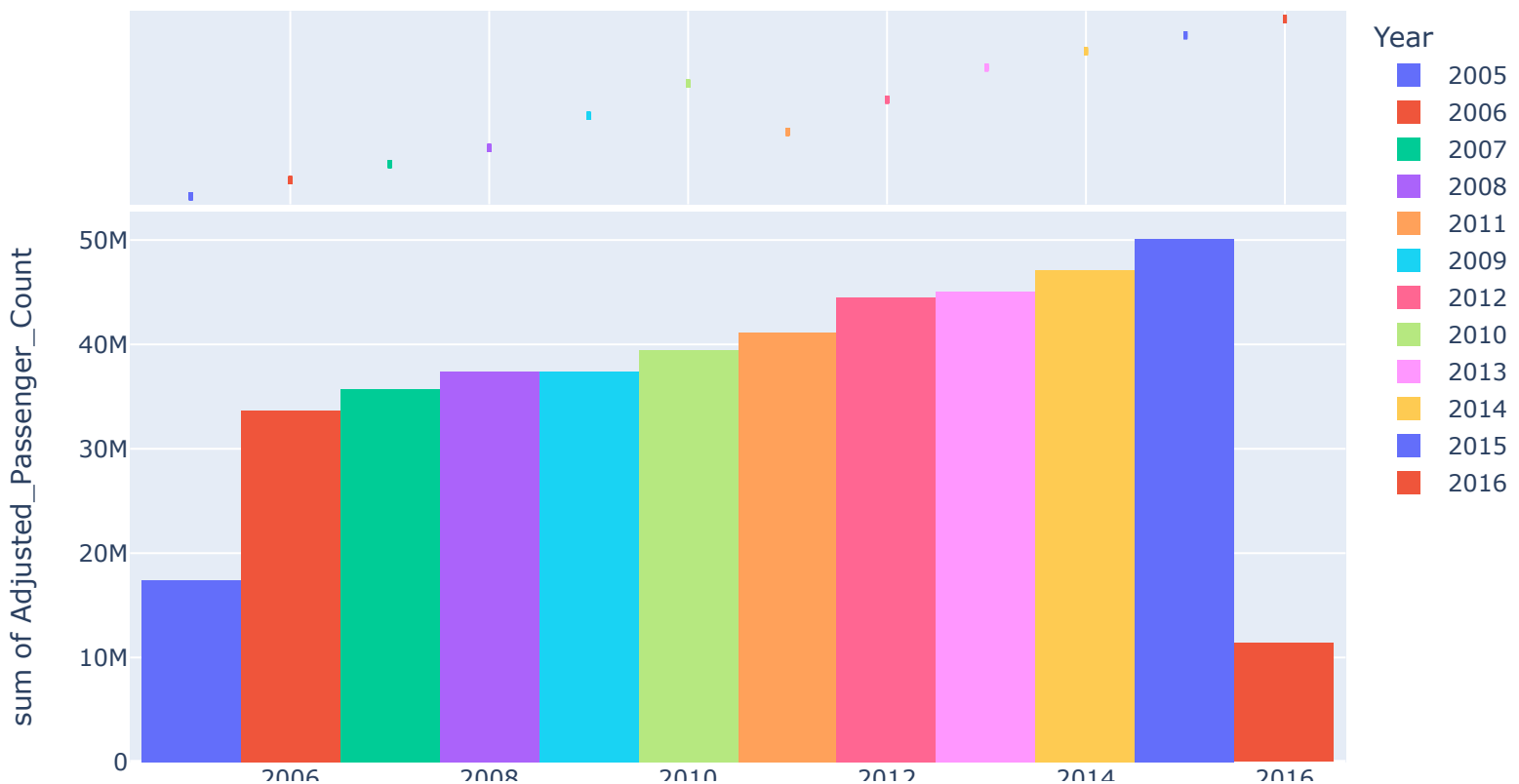
```
sns.countplot(data=df,x="Month",palette="coolwarm_r")
plt.title("Countplot of Month")
plt.show()
```




```
px.box(df,x='GEO_Region',y='Year',
       color='GEO_Region',template='ggplot2',
       labels={'GEO_Region':'GEO_Region',
               'Year':'Year'})
fig.update_layout(title="Box Plot of Year by GEO Region")
```

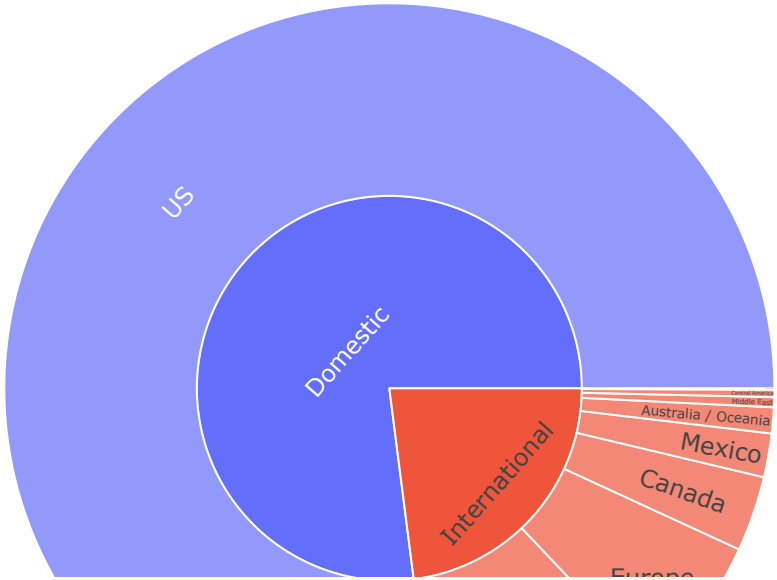



```
fig = px.histogram(df, x="Year", y="Adjusted_Passenger_Count", color="Year",
                   marginal="box", # or violin, rug
                   hover_data=df.columns)
fig.update_layout(title="Histogram Plot of Adjusted Passenger Count by Year")
fig.show()
```

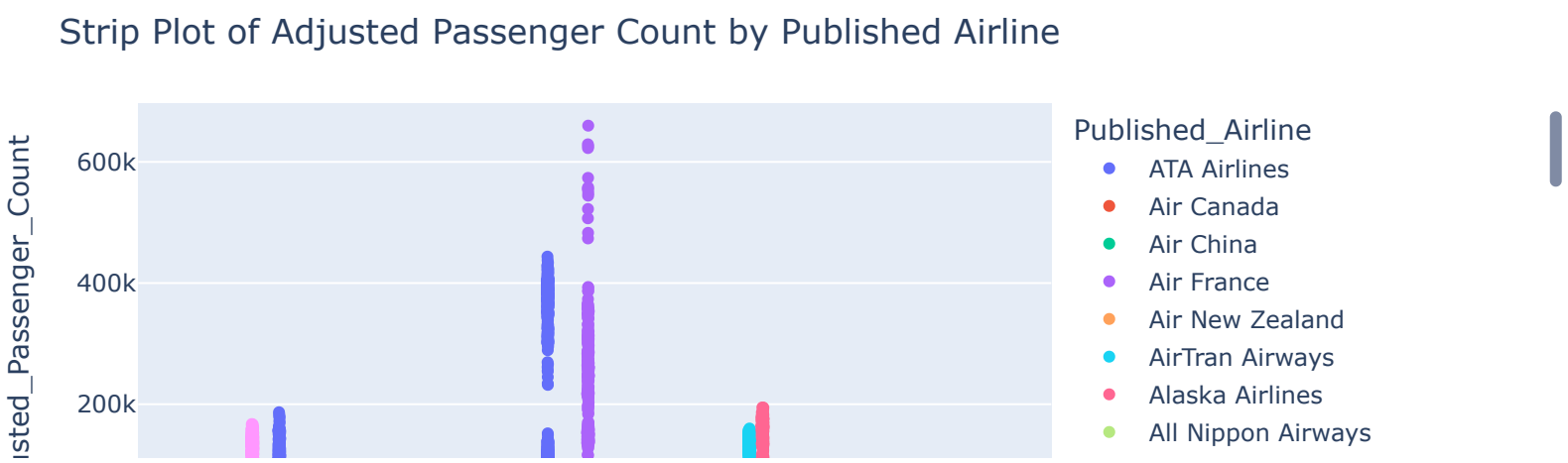



```
fig = px.sunburst(df, path=['GEO_Summary', 'GEO_Region'], values='Adjusted_Passenger_Count')
fig.update_layout(title="Sunburst Chart of Adjusted Passenger Count by GEO Summary and GEO Region")
fig.show()
```

Sunburst Chart of Adjusted Passenger Count by GEO Summary and GEO Region

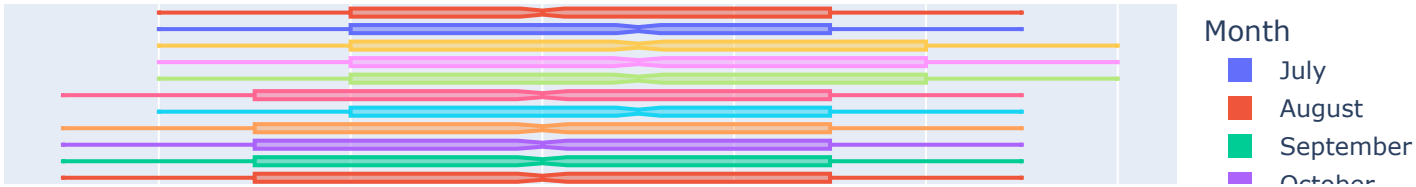



```
fig = px.strip(df, x='Published_Airline', y='Adjusted_Passenger_Count', color="Published_Airline")
fig.update_layout(title="Strip Plot of Adjusted Passenger Count by Published Airline")
fig.show()
```




```
fig = px.histogram(df, x="Year", y="Adjusted_Passenger_Count", color="Month",barmode="relative",
                   marginal="box")
fig.update_layout(title="Histogram Plot of Adjusted Passenger Count by Year and Month")
fig.show()
```

Histogram Plot of Adjusted Passenger Count by Year and Month

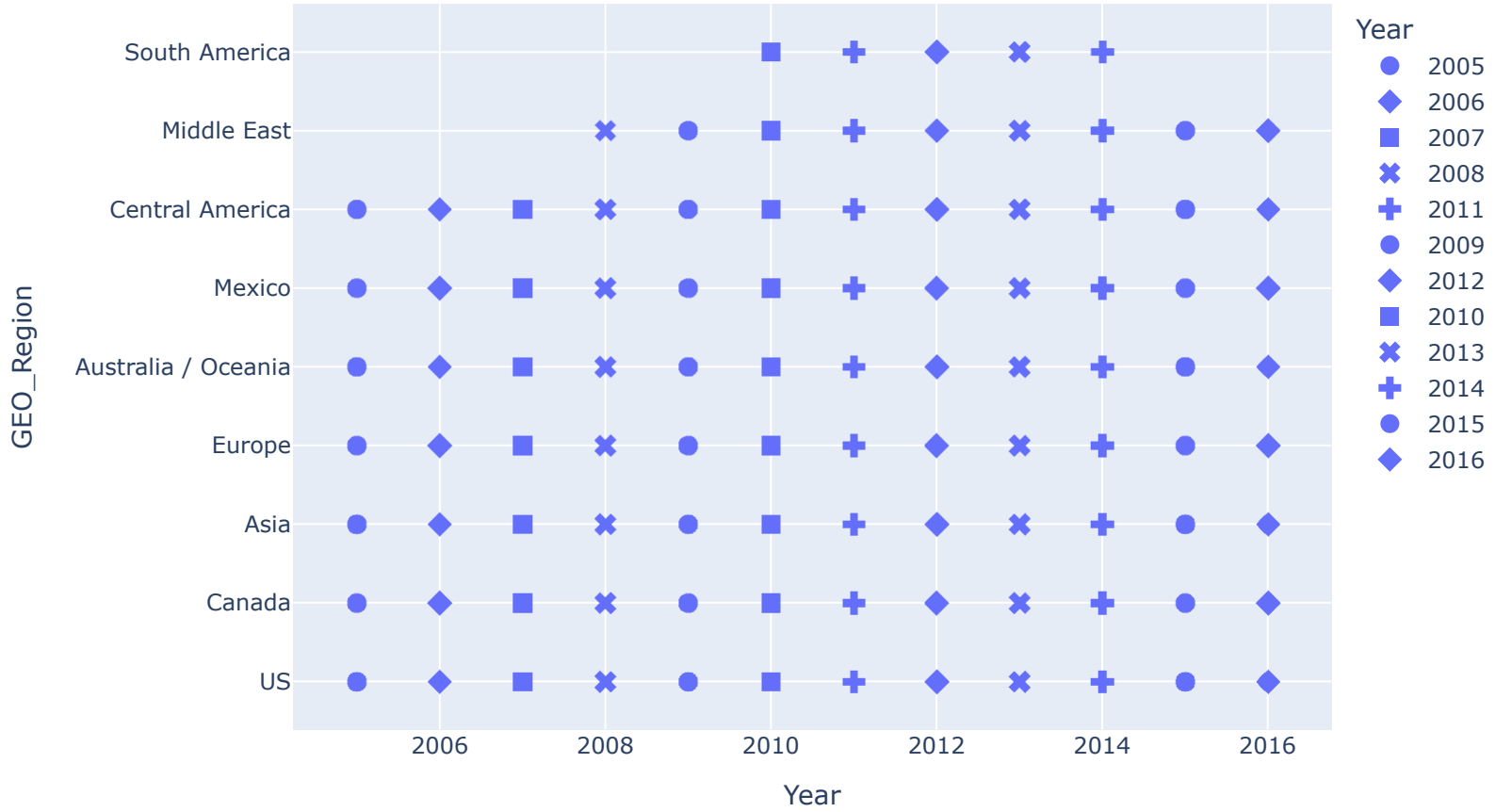



```
fig = px.strip(df, x='GEO_Region', y='Adjusted_Passenger_Count', color="GEO_Region")
fig.update_layout(title="Strip Plot of Adjusted Passenger Count by GEO Region")
fig.show()
```

Strip Plot of Adjusted Passenger Count by GEO Region

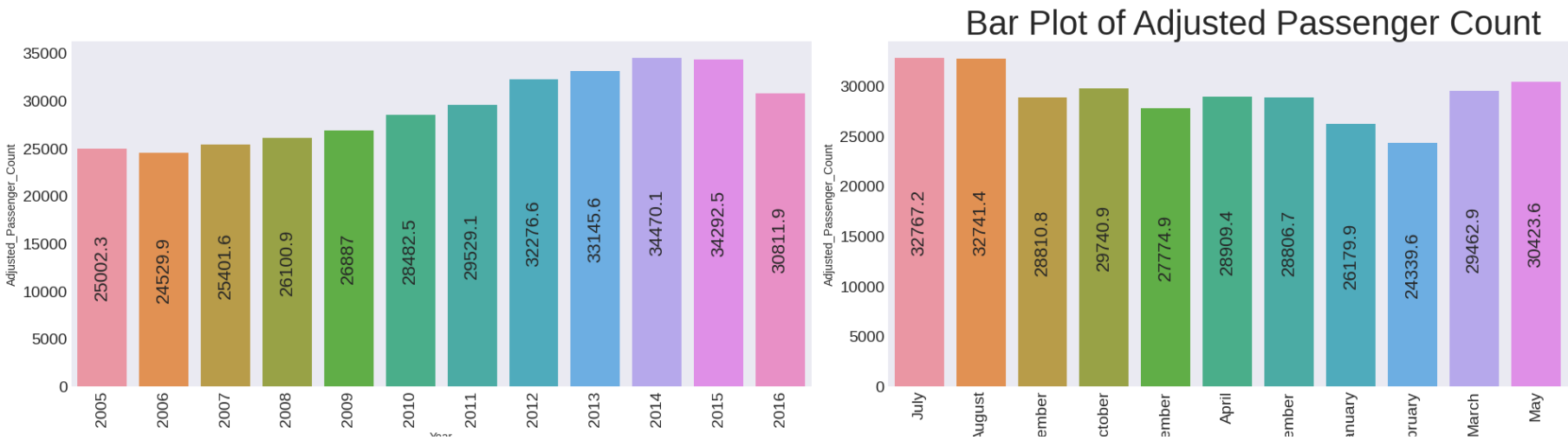
```
fig = px.scatter(df, y="GEO_Region", x="Year", symbol="Year")
fig.update_traces(marker_size=10)
fig.update_layout(title="Scatter Plot of GEO Region by Year")
fig.show()
```

Scatter Plot of GEO Region by Year

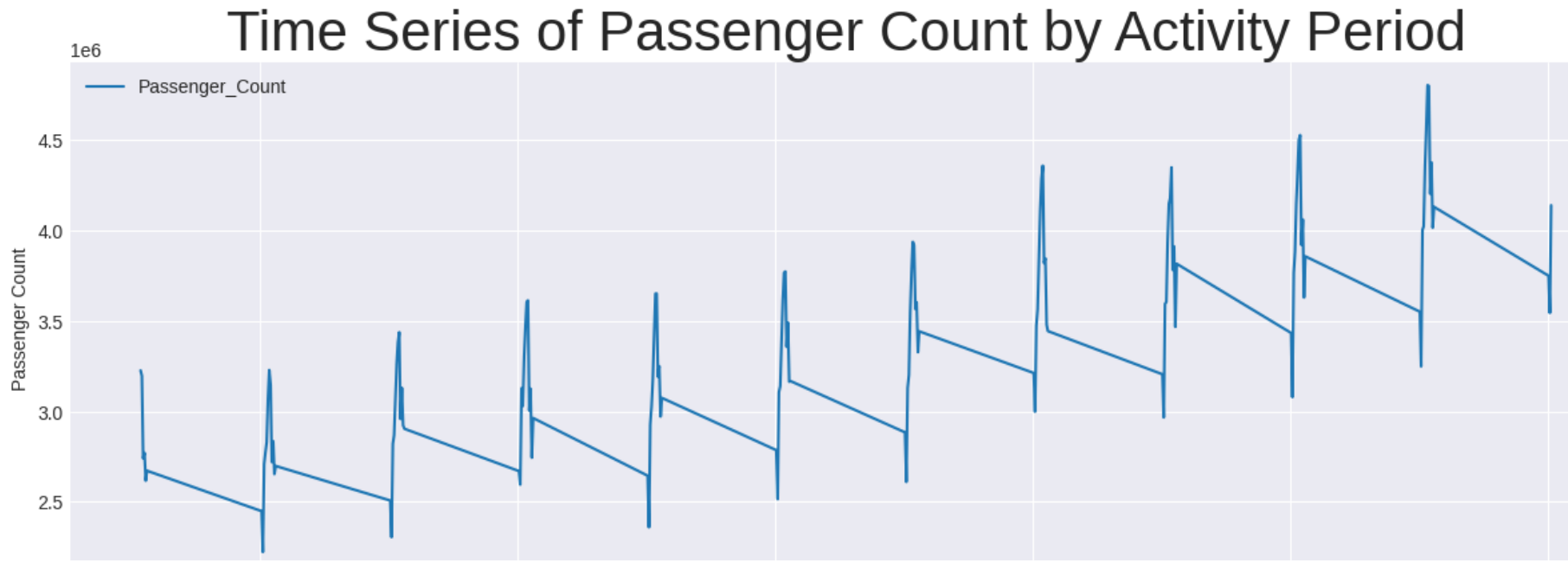


```
colunas = ['Year','Month']
plt.figure(figsize=(20,10))
for i, col in enumerate(colunas):
    axes = plt.subplot(2,2,i + 1)
    sns.barplot(x=df[col], y=df['Adjusted_Passenger_Count'], ci=None)
    plt.xticks(fontsize=14, rotation=90)
    plt.yticks(fontsize=14)
    for n in axes.containers:
        axes.bar_label(n, fontsize=16, rotation=90, label_type='center')
plt.tight_layout()
plt.title('Bar Plot of Adjusted Passenger Count ', fontsize=30)
plt.show()
```

```
<ipython-input-37-e94ebe94547d>:5: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

<ipython-input-37-e94ebe94547d>:5: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.
```



```
TS1 = df.groupby("Activity_Period")["Passenger_Count"].sum().to_frame()

fig, ax = plt.subplots(1,1,figsize=(15,5))
TS1.plot(ax=ax)
ax.set_xlabel("Date")
ax.set_ylabel("Passenger Count")
plt.grid(True)
ax.set_title("Time Series of Passenger Count by Activity Period",fontsize=30)
plt.show()
```



Result

- The number of domestic flights is higher than international flights.

- However, the number of international passengers is higher than that of domestic flights.
- Although there are more domestic flights, there are more people on international flights.
- The number of passengers going to US is very high.
- Usually tickets are not cheap.
- The "International" terminal was used more as a terminal.
- Terminal 3 was used more for the number of people.
- More passengers exit through door F.
- Gate A is mostly used as a gate.
- There are more flights in 2015.
- There were more flights in the US in 2010.
- More flights were made to Canada in 2011.
- More flights were made to Asia in 2010.

- More flights were made to Europa in 2011.
- More flights were made to Austraila in 2010.
- More flights were made to Mexico in 2011.
- More flights were made to Central America in 2011.
- More flights were made to the Middle East in 2013.
- More flights were made to South America in 2012.
- United Airlines ranks first among the countries with the most flights.
- Too many people traveled in 2015.

Forcast

```
import tensorflow as tf

import random
from keras.models import Sequential
from keras.layers import LSTM, Dense, RepeatVector, TimeDistributed, LeakyReLU
from keras.optimizers import Adam
```

```
from sklearn.preprocessing import MinMaxScaler


np.random.seed(123)
random.seed(123)
tf.random.set_seed(1234)


df = df[['Activity_Period', 'Passenger_Count']]
df = df.groupby('Activity_Period').sum()
df.sort_values(by='Activity_Period', inplace=True)

scaler = MinMaxScaler()
scaler.fit(df)
df['Passenger_Count'] = scaler.transform(df)
df.head()
```

Activity_Period	Passenger_Count
200507	0.388944
200508	0.377345
200509	0.200739
200510	0.212438
200511	0.152944

```
X_raw = df.copy()

n_steps_in = 24
n_steps_out = 12

split_idx = 158 - (n_steps_in + n_steps_out + 1)
X_raw_train = X_raw[:split_idx]
X_raw_test = X_raw[split_idx:]
```

```
print(X_raw_train.shape)
print(X_raw_test.shape)
```

```
 (121, 1)
 (8, 1)
```

```
def split_sequence(sequence, n_input: int, n_output: int) -> np.array:
    """Splits a times series sequence into input and output sequences

    Args:
        sequence: a time series to be split
        n_input: number of input steps
        n_output: number of output steps

    Returns:
        two numpy arrays

    """
    x, y = [], []
    for i, _ in enumerate(sequence):
        input_end = i + n_input
        output_end = input_end + n_output - 1
        # stop if we reach end of the sequence
        if output_end > len(sequence):
            break

        seq_x, seq_y = sequence[i: input_end], sequence[input_end-1: output_end]
        x.append(seq_x)
        y.append(seq_y)
    return np.array(x), np.array(y)


test_seq = [1,2,3,4,5,6,7,8,9,10]
a, b = split_sequence(test_seq, 5, 3)
print(a.shape)
print(b.shape)
```

```
print(a[-1])
print(b[-1])
```

```
     (4, 5)
     (4, 3)
     [4 5 6 7 8]
     [ 8  9 10]
```

```
X_train, y_train = split_sequence(X_raw_train.values, n_steps_in, n_steps_out)
X_test, y_test = split_sequence(X_raw_test.values, n_steps_in, n_steps_out)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
     (87, 24, 1)
     (87, 12, 1)
     (0,)
     (0,)
```

```
n_features = 1
```

```
sample = 3

tmp = X_train[sample].flatten()
x_tmp = pd.DataFrame({'date': np.arange(len(tmp)), 'value': tmp}).set_index('date')

tmp = y_train[sample].flatten()
y_tmp = pd.DataFrame({'date': np.arange(n_steps_in, n_steps_in+len(tmp)), 'value': tmp}).set_index('date')

plt.plot(x_tmp, label="input")
plt.plot(y_tmp, label="output")
plt.legend()
plt.show()
```


input
output
0.5
0.4
0.3
0.2
0.1

✓ 2s completed at 12:21 PM