Tourism in Ireland

Abstract

Tourism in Ireland offers a captivating experience that combines a rich history, stunning landscapes, and a vibrant culture. With its unique blend of tradition and modernity, the Emerald Isle attracts visitors from all over the world eager to explore its ancient ruins, castles, charming towns, and stunning natural surroundings. In recent years, there has been a notable resurgence in interest in domestic tourism, with an increasing number of Irish people choosing to explore the destinations within their own borders. However, a crucial factor that significantly influences the tourist experience in Ireland is the peculiar climatic conditions of the region.

In this context, this study investigates domestic tourism in Ireland, analysing recent data on travel patterns, tourist behaviours and possible impacts of weather conditions. The research involves detailed data preparations, descriptive analyses, statistical analysis, and machine learning techniques. The results revealed a significant influence of climatic conditions on Irish domestic tourism, as well as a notable increase in tourism in recent years. Through classification and regression machine learning models, it was possible to identify complex patterns and relationships between climate and tourism, allowing future projections based on climate predictions. These findings are of utmost importance as they can aid in the management and promotion of tourism in Ireland by providing valuable insights and guiding policies and strategies that maximise economic and cultural benefits.

Introduction

In Ireland's current scenario, marked by a series of social challenges that demand attention and strategic responses, domestic tourism in Ireland emerges as a vital aspect for the country's economy and culture. As we look at the international scenario, where wide-ranging problems, such as economic, health, environmental and public safety issues, occupy the global agenda, it is essential to recognize the importance of also addressing local challenges, such as tourism and its consequences.

In this context, the present study proposes to analyze domestic tourism in Ireland, considering data from recent years that detail the regions visited, the reasons for the trips and characteristics of the trip, such as the number of nights. In addition, complementary public data on weather conditions in Ireland provided by Met Éireann, the National Meteorological Service of Ireland, which monitors, analyzes and forecasts weather information on the island, will be used, along with data preparation techniques, statistical analysis and modeling, programming and machine learning methods for the following objectives:

- To study the behaviour of Irish domestic tourism in recent years and to identify travel patterns based on periods, reasons for travel and regions visited;
- Identify possible impacts of weather conditions on Irish domestic tourism.

.

DATABASES

Databases from two different sources were used for the study. The first relates to domestic tourism in Ireland, obtained through the website of The Central Statistics Office (CSO), Ireland's national statistics office. The second relates to climatic conditions in Ireland and the data was obtained through the website of the Met Éireann, Ireland's national meteorological service. Below are more details about the databases.

Domestic Travel by Irish Residents Database

The Database **Domestic Travel by Irish Residents** It has information on domestic trips undertaken by Irish residents in the last 20 years and is divided into two files, the first with data from the years 2003 to 2017, and the second with data from the years 2018 to 2023. Below are more details about the data:

- Dataset - Domestic Travel by Irish Residents

```
In [2]: dd1 = pd.read_csv("HTQ10.20240322T190314.csv");dd1.head()
```

Out[2]:

	Statistic Label	Quarter	Reason for Journey	Region Visited	UNIT	VALUE
0	Number of Trips by Irish Residents	2003Q3	All reasons for journey	State	Thousand	NaN
1	Number of Trips by Irish Residents	2003Q3	All reasons for journey	Border, Midland and Western	Thousand	NaN
2	Number of Trips by Irish Residents	2003Q3	All reasons for journey	Border	Thousand	NaN
3	Number of Trips by Irish Residents	2003Q3	All reasons for journey	Midland	Thousand	NaN
4	Number of Trips by Irish Residents	2003Q3	All reasons for journey	West	Thousand	NaN

```
In [3]: dd2 = pd.read_csv("HTQ17.20240322T190314.csv");dd2.head()
```

Out[3]:

	Statistic Label	Quarter	Reason for Journey	Region Visited	UNIT	VALUE
0	Number of Trips by Irish Residents	2018Q1	All reasons for journey	State	Thousand	2118.0
1	Number of Trips by Irish Residents	2018Q1	All reasons for journey	Northern and Western	Thousand	431.0
2	Number of Trips by Irish Residents	2018Q1	All reasons for journey	Border	Thousand	NaN
3	Number of Trips by Irish Residents	2018Q1	All reasons for journey	West	Thousand	NaN
4	Number of Trips by Irish Residents	2018Q1	All reasons for journey	Southern	Thousand	1038.0

Data Dictionary

Column	Description
Statistic Label	Variable names
Quarter	Concatenated Year and Quarter variable
Reason for Journey	Reason for Travel
Region Visited	Region Visited
UNIT	Unit of measurement
VALUE	Variable value

Climatic Conditions in Ireland Database

The data **Climatic Conditions in Ireland** are subdivided into files containing information on climatic conditions of certain locations in Ireland. They are:

- Donegal
- May
- Sligo
- Cork
- Kerry
- Dublin
- Meath
- Westmeath
- Wexford
- Tipperary
- Galway
- Roscommon
- Clare

With these locations we can be sure to capture information on weather conditions in Ireland in general. Below are more details about the data:

- Dataset - Climatic Conditions in Ireland

In [6]:
```
std_name = '*.csv'
i=1
list_dataframes = []
for i in glob.glob(std_name):
    ddtp = pd.read_csv(i)
    if 'year' in ddtp.columns:
        ddtp = ddtp[ddtp['year']>=2003]
    list_dataframes.append(ddtp)
```

In [7]:
```
list_dataframes[1]
```

Out[7]:

	region	year	month	meant	maxtp	mintp	mnmax	mnmin	rain	gmin	wdsp	maxgt	sun
0	Cork	2004	5	11.6	17.6	5.3	14.4	8.8	37.1	2.4	9.7	57	
1	Cork	2004	6	14.2	20	7.2	16.7	11.8		2.8	10.9	36	
2	Cork	2004	8	15.7	20.7	8.4	18.3	13.1	110.9	4.2	10.5	41	
3	Cork	2004	9	14.4	21.3	7.3	16.2	12.6	94	3.6	13.5	51	
4	Cork	2004	10	10.7	14.8	2.8	13	8.4	141.5	0.1	14	55	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
231	Cork	2023	10	12.6	16.8	4.3	14.6	10.6	196.7	0.6	10.7	41	
232	Cork	2023	11	10.2	13.8	2.5	11.9	8.4	115.3	-0.8	15.2	55	
233	Cork	2023	12	9.6	12.8	1.5	11.1	8	123.1	-2.9	17	64	
234	Cork	2024	1	6.7	12.9	-1.9	8.9	4.4	80.9	-8.8	14.3	57	
235	Cork	2024	2	9.2	12.8	4.3	10.9	7.6	144.3	-1.8	14.4	44	

236 rows × 13 columns

Data Dictionary

Column	Description
year	Year
month	Month
rain	Precipitation Amount (mm)
meant	Mean Air Temperature (C)
maxtp	Maximum Air Temperature (C)
mintp	Minimum Air Temperature (C)
mnmax	Mean Maximum Temperature (C)
mnmin	Mean Minimum Temperature (C)
gmin	Grass Minimum Temperature (C)
wdsp	Mean Wind Speed (knot)
mxgt	Highest Gust (knot)
sun	Sunshine duration (hours)
ind	Indicator

DATA PREPARATION

Data Preparation - Domestic Travel by Irish Residents

Below are the steps taken in the preparation of the database **Domestic Travel by Irish Residents**:

1. First, it was necessary to merge the two files, one of them containing data from 2003 to 2017, and the other with data from 2018 to 2023.

- Data Preparation - Domestic Travel by Irish Residents

In [9]:
```
# Concatenating Dataframes

dd = pd.concat([dd1,dd2],ignore_index = True)
dd.head()
```

Out[9]:

	Statistic Label	Quarter	Reason for Journey	Region Visited	UNIT	VALUE
0	Number of Trips by Irish Residents	2003Q3	All reasons for journey	State	Thousand	NaN
1	Number of Trips by Irish Residents	2003Q3	All reasons for journey	Border, Midland and Western	Thousand	NaN
2	Number of Trips by Irish Residents	2003Q3	All reasons for journey	Border	Thousand	NaN
3	Number of Trips by Irish Residents	2003Q3	All reasons for journey	Midland	Thousand	NaN
4	Number of Trips by Irish Residents	2003Q3	All reasons for journey	West	Thousand	NaN

2. The UNITE **variable was removed**, as the variable will not be useful in the analysis.
3. The **Statistic Label** variable was composed of 3 features:
    - Average Length of Stay by Irish Residents
    - Number of Nights by Irish Residents
    - Number of Trips by Irish Residents

    In this way, the variable was transposed, making the features become columns in the database.

```
In [12]: # Transposing the Dataframe
dd = dd.pivot_table(index=['Quarter','Reason for Journey','Region Visited'],columns = 'Statistic Label', values = 'VALUE').reset_
dd.head()
```

Out[12]:

Statistic Label	Quarter	Reason for Journey	Region Visited	Average Length of Stay by Irish Residents	Number of Nights by Irish Residents	Number of Trips by Irish Residents
0	2003Q3	Business	Border	3.7	37.0	10.0
1	2003Q3	Business	Dublin	2.3	93.0	41.0
2	2003Q3	Business	Mid-East	3.6	29.0	8.0
3	2003Q3	Business	Mid-West	3.7	44.0	12.0
4	2003Q3	Business	Midland	3.0	24.0	8.0

4. Separation of the **Quarter** variable into two variables called **Year** and **Quarter**.

5. Renaming of variables to make them easier to manipulate during analysis.

```
In [17]: # Renaming and reordering the collums
dd = dd.rename(columns = {'Reason for Journey':'Reason','Region Visited':'Destiny','Average Length of Stay by Irish Residents':
                          'Number of Nights by Irish Residents':'Total_Nights','Number of Trips by Irish Residents':'Num_Trips'})
dd = dd[['Destiny','Year','Quarter','Reason','Nights_Trip','Total_Nights','Num_Trips']]
dd.columns.name = None
dd.head()
```

Out[17]:

	Destiny	Year	Quarter	Reason	Nights_Trip	Total_Nights	Num_Trips
0	Border	2003	3	Business	3.7	37.0	10.0
1	Dublin	2003	3	Business	2.3	93.0	41.0
2	Mid-East	2003	3	Business	3.6	29.0	8.0
3	Mid-West	2003	3	Business	3.7	44.0	12.0
4	Midland	2003	3	Business	3.0	24.0	8.0

Data Preparation - Climatic Conditions in Ireland

Below are the steps taken in the preparation of the database **Climatic Conditions in Ireland**:

1. Creation of the **Quarter variable** based on the Month variable.
2. Transformation of variable into numeric.
3. Summarization of numerical variables by the average based on quarter, year and region.
4. Concatenation of all the databases of the various locations.

- Data Preparation - Climatic Conditions in Ireland

```
In [18]: ## Creating a function to define Quarter
def quarter_month(month):
    return(month - 1) // 3 + 1
```

```
In [19]: ## Creating variable Quarter
## Transforming variables in numeric
## ## Summarising the variables with mean based in Region, Year and Quarter

list_df_preps = []
for i in list(range(0,len(list_dataframes))):
    df = list_dataframes[i]
    df['quarter'] = df['month'].apply(quarter_month)
    for i in list(range(3,13)):
        df.iloc[:,i] = pd.to_numeric(df.iloc[:,i],errors = 'coerce')
    ## Summarising the variables with mean based in Region, Year and Quarter
    df = df.drop(columns = ['month'])
    df_summar = df.groupby(['region','year','quarter']).mean().reset_index()
    list_df_preps.append(df_summar)
```

```
In [20]: ## Concatenating the dataframes of Climate Conditions
dd_clim = pd.concat(list_df_preps, ignore_index = True)
dd_clim
```

Out[20]:

	region	year	quarter	meant	maxtp	mintp	mnmax	mnmin	rain	gmin	wdsp	maxgt	sun
0	Clare	2003	1	7.266667	14.733333	-1.333333	10.633333	3.833333	48.733333	-8.933333	9.866667	51.333333	104.066667
1	Clare	2003	2	12.833333	23.766667	5.000000	16.700000	8.966667	82.266667	-1.466667	9.666667	39.000000	169.000000
2	Clare	2003	3	16.500000	25.933333	8.400000	20.266667	12.700000	41.866667	1.466667	7.433333	33.000000	137.066667

5. Finally, the name of the towns has been replaced by the name of the region where it is located, respecting the location criteria and summarizing the city/county databases using the average. Below is a breakdown of the cities/counties by region:

Region	Cities / Counties					
State	Galway	Dublin	Cork			
Border, Midland and Western	Donegal	Mayo	Sligo	Roscommon		
Border	Donegal	Sligo				
Midland	Roscommon					
West	Mayo	Galway				
Southern and Eastern	Clare	Cork	Dublin	Kerry	Meath	Wexford
Dublin	Dublin					
Mid-East	Dublin	Meath				
South-East	Wexford					
South-West	Cork	Kerry				
Northern and Western	Donegal	Mayo	Sligo			
Southern	Clare	Cork	Kerry			
Eastern and Midland	Dublin	Meath	Roscommon	Tipperary	Westmeath	

Join the Datasets

After performing data preparation **Climatic Conditions in Ireland**, the data were merged **Climatic Conditions in Ireland** with **Domestic Travel by Irish Residents**, using the variables as a key **Destiny**, **Year** and **Quarter**.

```
## Merging thee datasets
ddt = pd.merge(dd,clim, left_on=['Destiny', 'Year', 'Quarter'], right_on = ['region', 'year', 'quarter'], how = 'left')
ddt
```

	Destiny	Year	Quarter	Reason	Nights_Trip	Total_Nights	Num_Trips	year	quarter	meant	maxtp	mintp	mnmax	mnmin	
0	Border	2003	3	Business	3.7	37.0	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	Dublin	2003	3	Business	2.3	93.0	41.0	2003.0	3.0	15.466667	25.866667	5.900000	20.133333	10.900000	32
2	Mid-East	2003	3	Business	3.6	29.0	8.0	2003.0	3.0	15.466667	25.866667	5.900000	20.133333	10.900000	32
3	Mid-West	2003	3	Business	3.7	44.0	12.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	Midland	2003	3	Business	3.0	24.0	8.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1542	State	2023	3	Other reasons	2.5	654.0	263.0	2023.0	3.0	15.177778	23.433333	7.255556	18.522222	11.844444	134
1543	Eastern and Midland	2023	3	Visiting friends/relatives	2.1	1036.0	491.0	2023.0	3.0	14.953333	24.473333	6.173333	18.973333	10.913333	131
1544	Northern and Western	2023	3	Visiting friends/relatives	2.4	809.0	335.0	2023.0	3.0	14.822222	24.155556	7.411111	18.022222	11.655556	127
1545	Southern	2023	3	Visiting friends/relatives	3.0	2779.0	919.0	2023.0	3.0	15.611111	23.033333	8.766667	18.266667	12.988889	140
1546	State	2023	3	Visiting friends/relatives	2.7	4624.0	1744.0	2023.0	3.0	15.177778	23.433333	7.255556	18.522222	11.844444	134

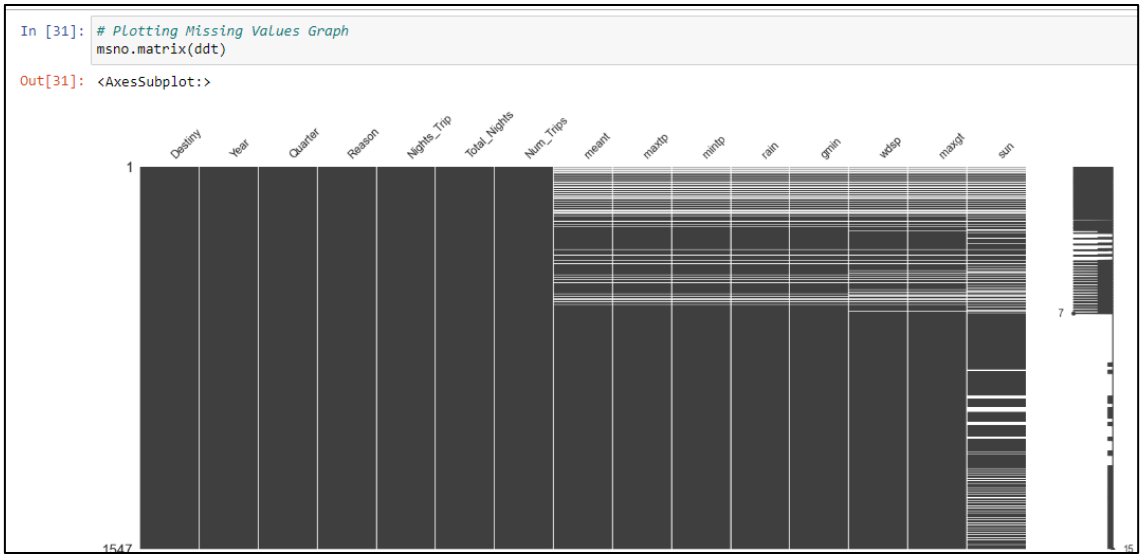
1547 rows × 20 columns

Removing Columns and Treating Missing Values

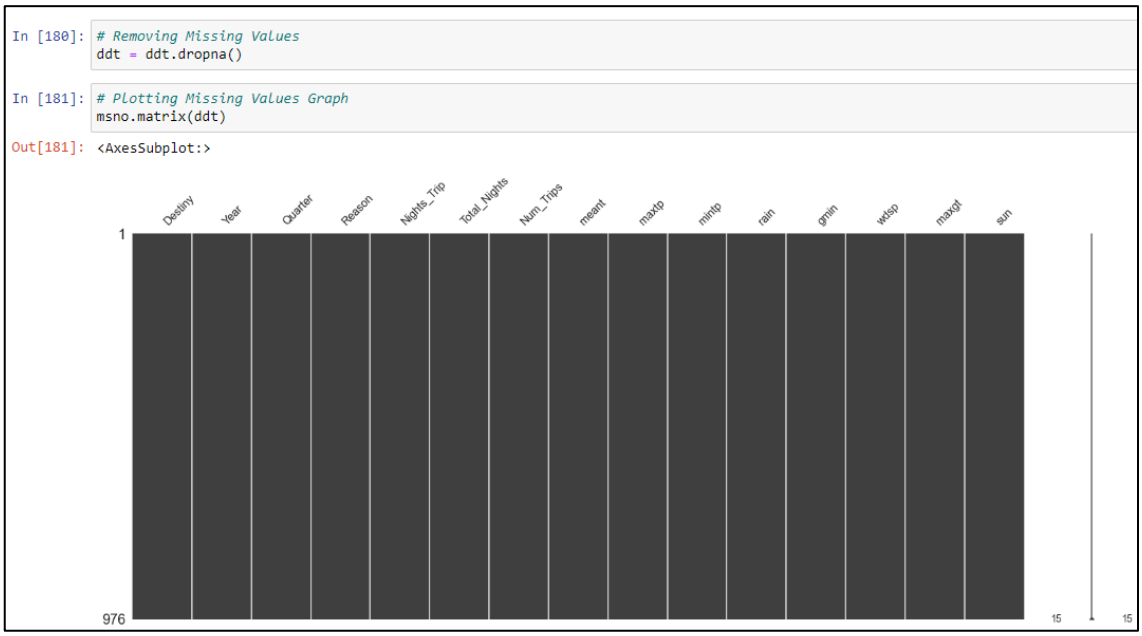
After merging the data, some columns needed to be removed due to duplication or simply because they were not part of the analysis. They are:

- Year (duplicity)
- Quarter (duplicity)
- Region (duplicity)
- Mnmax (outside the context of the analysis)
- Mnmin (outside the context of the analysis)

With the removal of columns, we need to check for the presence of null or missing values. For this, the graph below was plotted:



In this graph we can see a large presence of null values that need to be addressed or removed. For the study, it was decided to completely remove the missing and null values. In this way,



EXPLORATORY DATA ANALYSIS (EDA)

Descriptive Analysis

Starting the exploratory data analysis stage, it is first necessary to describe the variables to understand how they behave, the range of data and some basic statistical metrics. In the first moment, we will pay special attention to the data regarding tourism in Ireland. The other data related to climatic conditions will be used in the following stages of the study. Thus, describing the numerical variables, we have:

```
In [37]: ## Describing numeric variables
         ddt.describe()
```

Out[37]:

	Nights_Trip	Total_Nights	Num_Trips	meant	maxtp	mintp	rain	gmin	wdsp	maxgt	sun
count	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000
mean	2.723583	1463.627500	504.216667	10.468744	18.096316	1.637293	90.026317	-1.967949	9.842314	45.040995	118.296079
std	0.927732	2138.060264	646.707678	3.255728	4.090246	3.613028	30.184562	3.653066	1.644223	7.613627	47.380881
min	1.100000	6.000000	4.000000	4.233333	11.866667	-5.833333	30.833333	-9.400000	5.094444	29.000000	46.366667
25%	2.100000	209.500000	87.000000	7.650000	14.300000	-1.250000	65.944444	-4.811111	8.733333	39.444444	76.033333
50%	2.500000	611.500000	242.000000	10.308333	17.291667	1.188889	85.470833	-2.400000	9.744444	44.555556	104.700000
75%	3.100000	1739.000000	657.250000	13.716667	21.400000	4.666667	109.425000	1.286667	11.055556	50.166667	159.433333
max	8.400000	14777.000000	4518.000000	16.200000	26.786667	9.888889	188.400000	6.300000	14.233333	72.000000	215.066667

With the descriptive analysis of the numerical variables, we can observe that the variables **Total_Nights (number of total nights on trips)** and **Num_Trips (total number of trips)** They have a large range. This becomes clear when looking at the minimum and maximum values of the variables (**Total_Nights – min: 6 / max: 14777) (Num_Trips - min: 4 / max: 4518)**. In addition, we can also note that the standard deviation values are high (**Total_Nights – std: 2138.06) (Num_Trips – std: 645.70),** indicating considerable dispersion in the data. Another point that needs to be highlighted refers to the mean and median values. For both variables (**Total_Nights** and **Num_Trips)**, the mean and median values are far apart, indicating an asymmetric behavior in their distributions.

```
In [39]: ## Describing discrete variables
         ddt.describe(include = object)
```

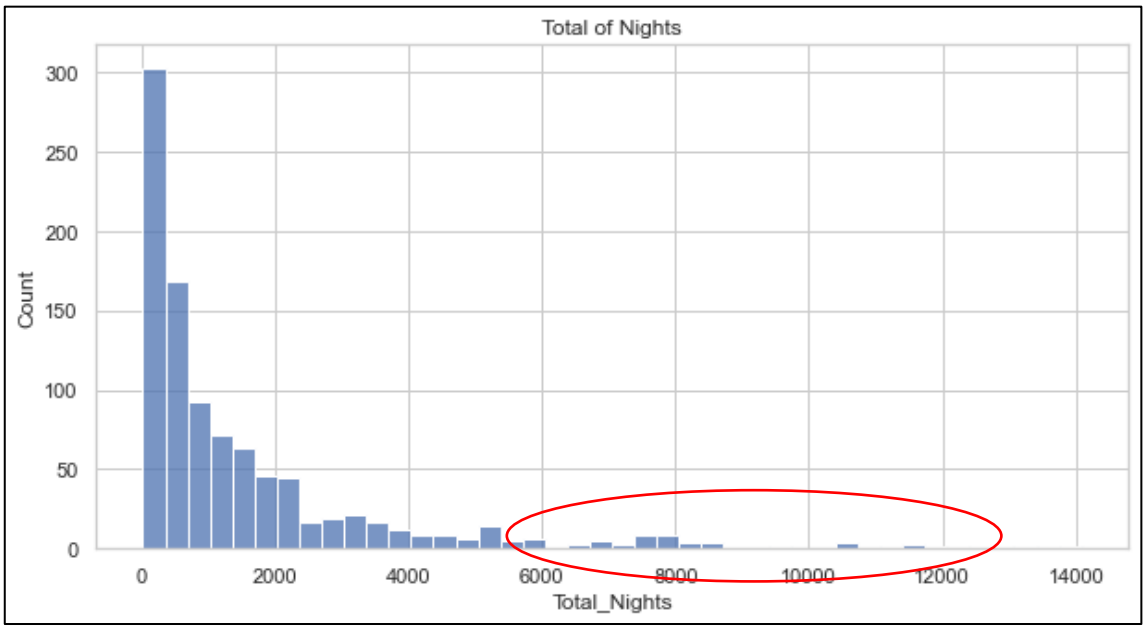
Out[39]:

	Destiny	Year	Quarter	Reason
count	1200	1200	1200	1200
unique	10	19	4	5
top	State	2006	3	Business
freq	355	108	319	243

Regarding the categorical variables, we can see in the image above that in the **top**, the most frequent information in the database is for trips to the region **State (Galway, Dublin, Cork)**, which are considered to be the largest cities in Ireland.

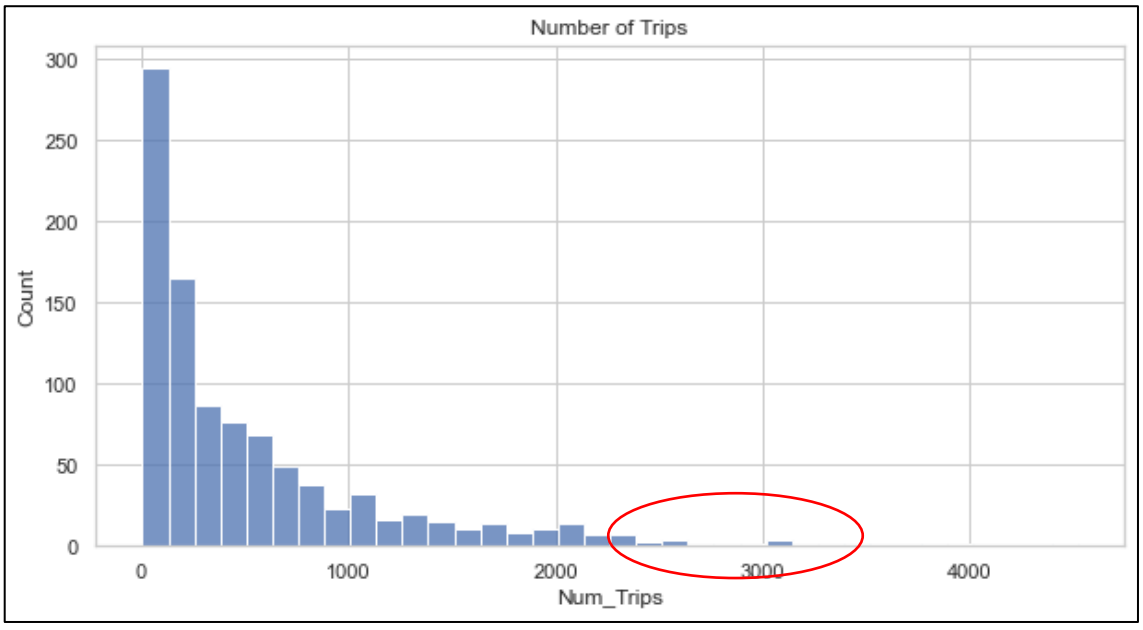
Graphical Analysis

*Histogram of **Total of Nights***



Histogram graph of the variable **Total_Nights (total number of nights on trips)**. The histogram of the variable shows what we could see in the descriptive analysis – high dispersion of the data, with occurrences of outliers to high asymmetry in the distribution.

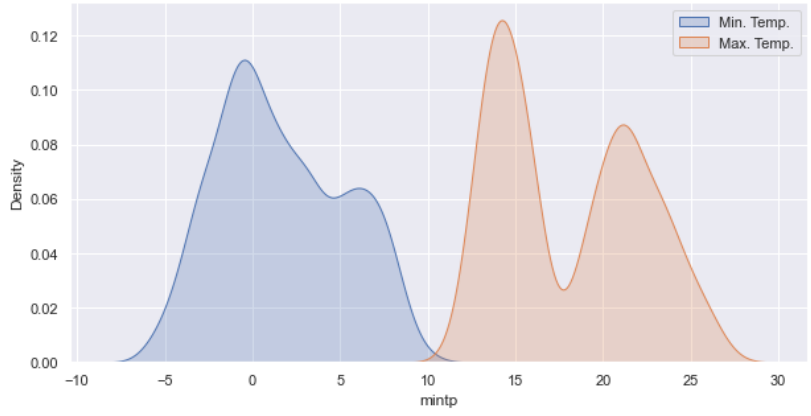
*Histogram of **Number of Trips***



Similar to the previous graph, the variable **Num_Trips (total number of trips)** also has high asymmetry and high dispersion in the data, proving the descriptive analysis previously performed.

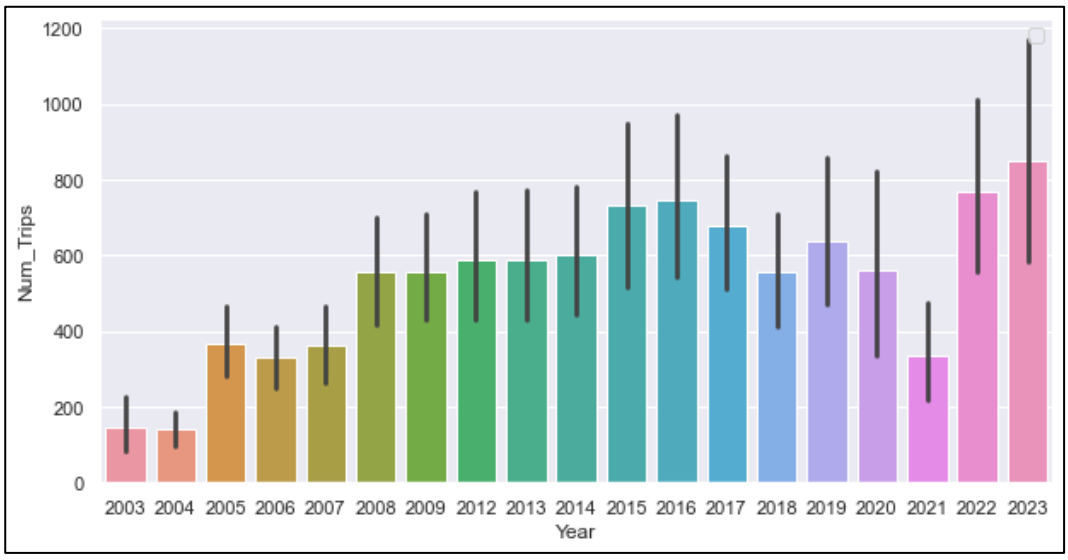
Density of Maximum and Minimum Temperatures

```
In [45]: # Plotting the density of Maximum and Minimum Temperatures
         plt.figure(figsize=(10,5))
         sns.kdeplot(ddt['mintp'],label = 'Min. Temp.',shade = True)
         sns.kdeplot(ddt['maxtp'],label = 'Max. Temp.',shade = True)
         plt.legend()

Out[45]: <matplotlib.legend.Legend at 0x2204ffedca0>
```



In the density graph of **the maxtp (maximum temperature)** and **mintp (minimum temperature) variables,** we can see that the average temperature between the two variables is around 10 degrees. For maximum temperatures, temperatures between 13 and 17 degrees are more frequent. For minimum temperatures, temperatures between -1 and 2 degrees are more frequent.

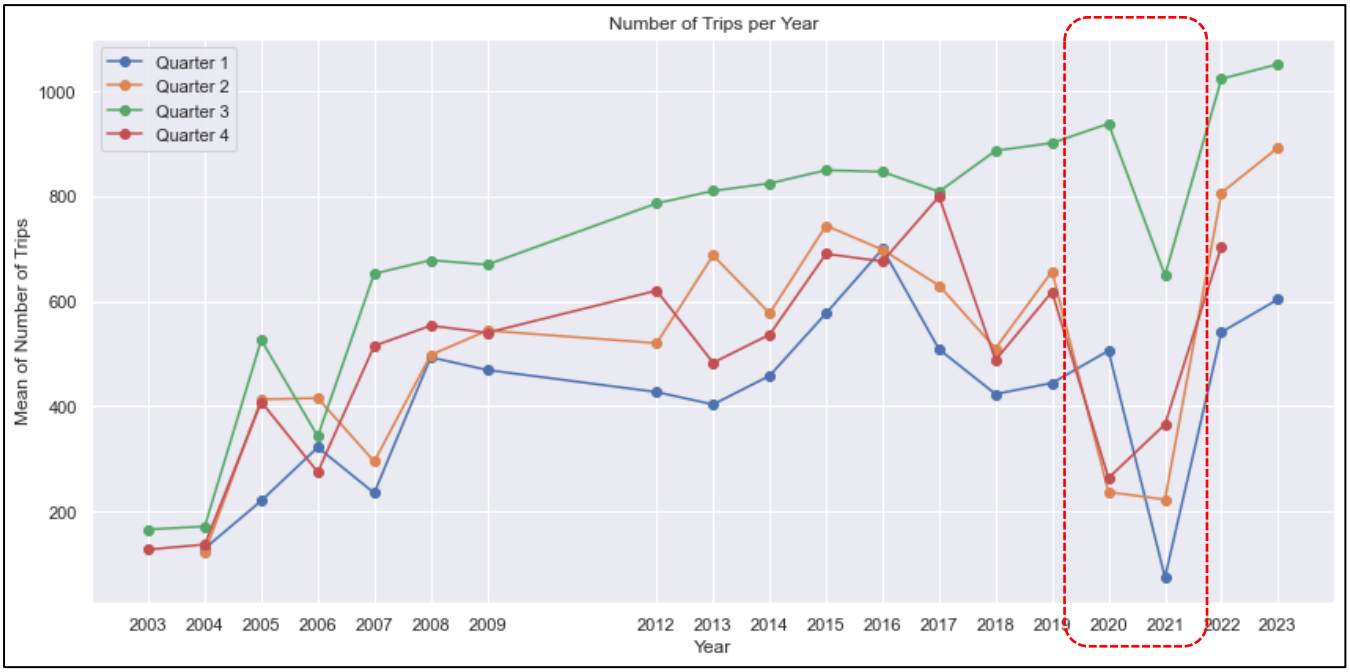
Plotting the Number of Trips per Year



Analyzing the graph with the number of trips distributed per year, we can notice some years out of the standard and understand the reason for this occurrence. Basically, the numbers for the years 2020 and 2021 were affected by the COVID-19 effect, where

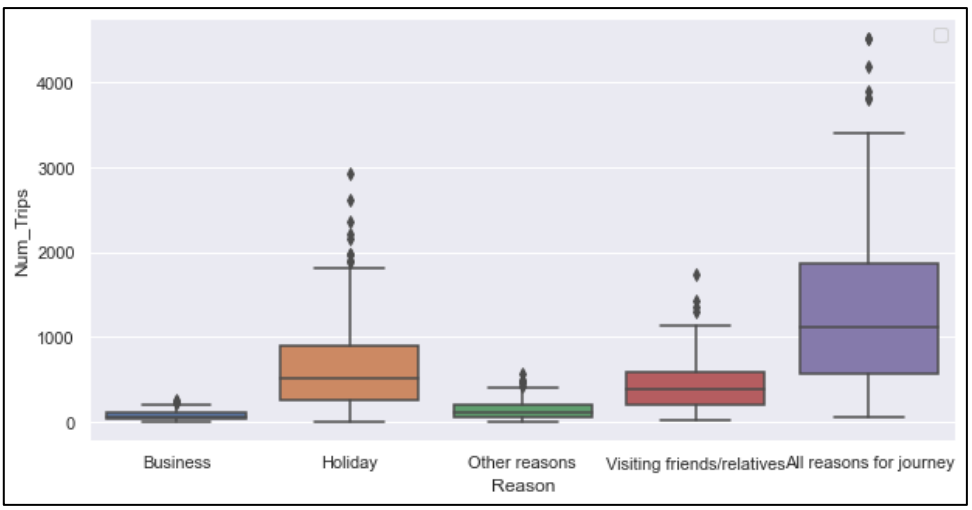
it was expected that the number of trips would decrease due to travel restrictions. The years 2003 and 2004 also show values below the standard, but this is due to the lack of numbers of occurrence of trips in certain blocks. Overall, the chart shows that there is a growth in the number of domestic trips over the years. In addition, the lines at the top of each bar show, in some cases, a large dispersion of the data, with the likely presence of outliers.

To decide whether the removal of data affected by COVID-19 will be by quarters or the full years, we need a complementary analysis where we can look at the numbers by quarters.



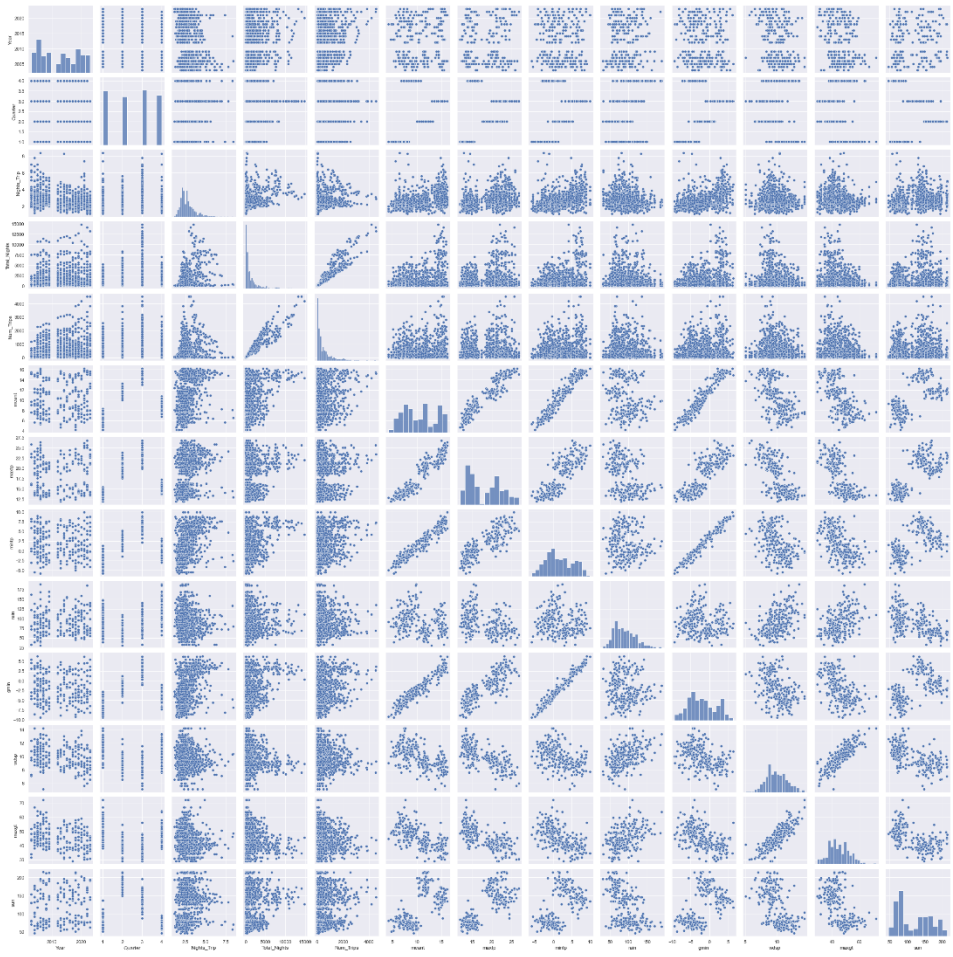
Looking at the chart above, we can see that the effects of travel restrictions impacted travel numbers during 3 quarters of 2020, and during the entire year of 2021. For this reason, we have decided to remove the years 2020 and 2021 in their entirety so that the analysis is not impacted.

Boxplot of the Number of Trips per Reason



Looking at the graph, we can notice a higher frequency in the number of trips when the reason is **Holiday** or **All reasons for Journey**, which is interpreted as being trips with more than one reason, such as **Holiday** and **Visiting friends/relatives.** It is also possible to notice the presence of outliers in the variables that need to be addressed. In general, the distribution of the number of trips is different depending on the reason. This is the case of Holiday trips, where the median exceeds 500 trips, but when the reason is **All reasons for Journey,** this median exceeds 1000 trips.

Plotting the dispersion of all variables with pairplot

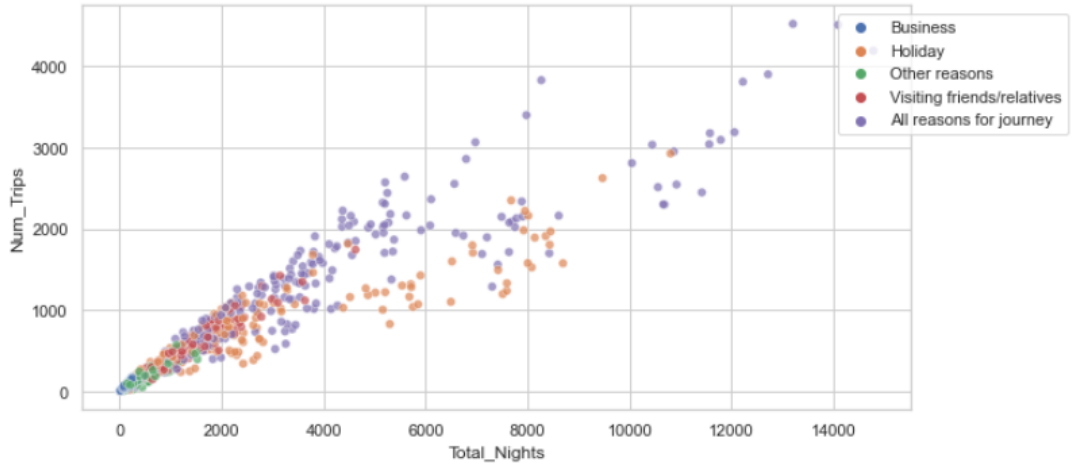


The plot of this graph aims to identify possible linear relationships between the variables. The visualization is not ideal due to the size of the graph, but it is possible to notice some linearity relationships between the variables, which may enable the construction of a linear regression model. However, we need further analysis to make such a decision.

Outliers Analysis

Outliers in variables Number of Trips and Total Nights

In [49]:
```
# Change default style
sns.set_style('whitegrid')
# Plot
plt.figure(figsize=(10, 5))
sns.scatterplot(data=ddt, x='Total_Nights', y='Num_Trips', alpha=0.7, hue='Reason')
plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1));
```



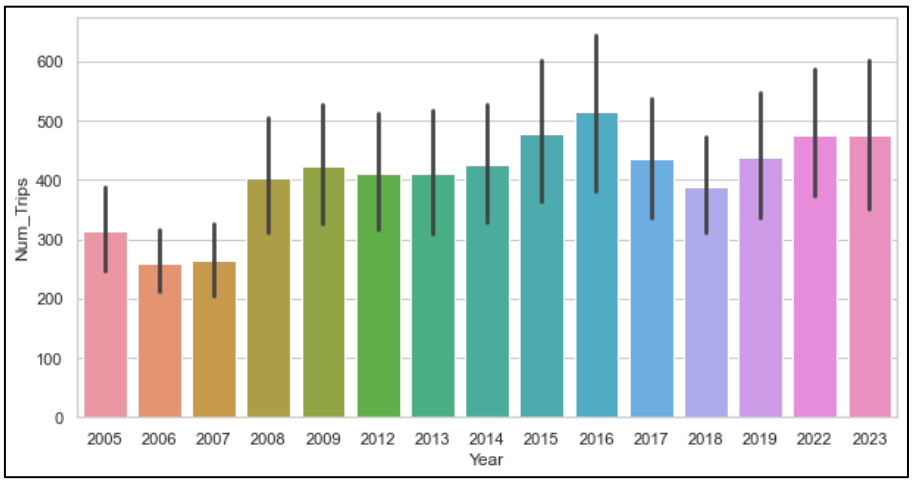
In [50]:
```
# Removing Outliers for Number of Trips
Q1 = ddt.Num_Trips.quantile(0.25)
Q3 = ddt.Num_Trips.quantile(0.75)
IQR = Q3 - Q1
ddt_outliers = ddt[(ddt.Num_Trips >= Q1 - 1.5*IQR) & (ddt.Num_Trips <= Q3 + 1.5*IQR)]
```

Removing COVID-19 Pandemic Effect

In [54]:
```
# Removing Years with Covid-19 Pandemic Effect and other years with outliers values in the trips
ddt_outliers = ddt_outliers.query('Year != 2020 and Year != 2021 and Year != 2003 and Year != 2004' )
ddt_outliers
```

Out[54]:

	Destiny	Year	Quarter	Reason	Nights_Trip	Total_Nights	Num_Trips	meant	maxtp	mintp	rain	gmin	wdsp	max
216	Border, Midland and Western	2005	1	All reasons for journey	3.0	1482.0	501.0	7.050000	14.683333	-1.800000	91.033333	-5.883333	10.866667	53.6666
217	Southern and Eastern	2005	1	All reasons for journey	2.7	2688.0	990.0	7.408333	14.208333	-1.408333	74.266667	-5.033333	11.408333	48.9166
218	State	2005	1	All reasons for journey	2.8	4170.0	1490.0	7.183333	13.400000	-1.466667	51.366667	-4.783333	12.533333	48.8333
220	Border, Midland and Western	2005	1	Business	2.5	162.0	65.0	7.050000	14.683333	-1.800000	91.033333	-5.883333	10.866667	53.6666
221	Dublin	2005	1	Business	2.2	100.0	45.0	6.500000	14.500000	-3.166667	46.300000	-6.133333	11.166667	49.3333



STATISTICAL ANALYSIS

Measures

Mean

- Mean of Number of Trips is: 384.755656
- Mean of Number of Nights per Trip is: 2.615045
- Mean of Total of Nights is: 1046.036199

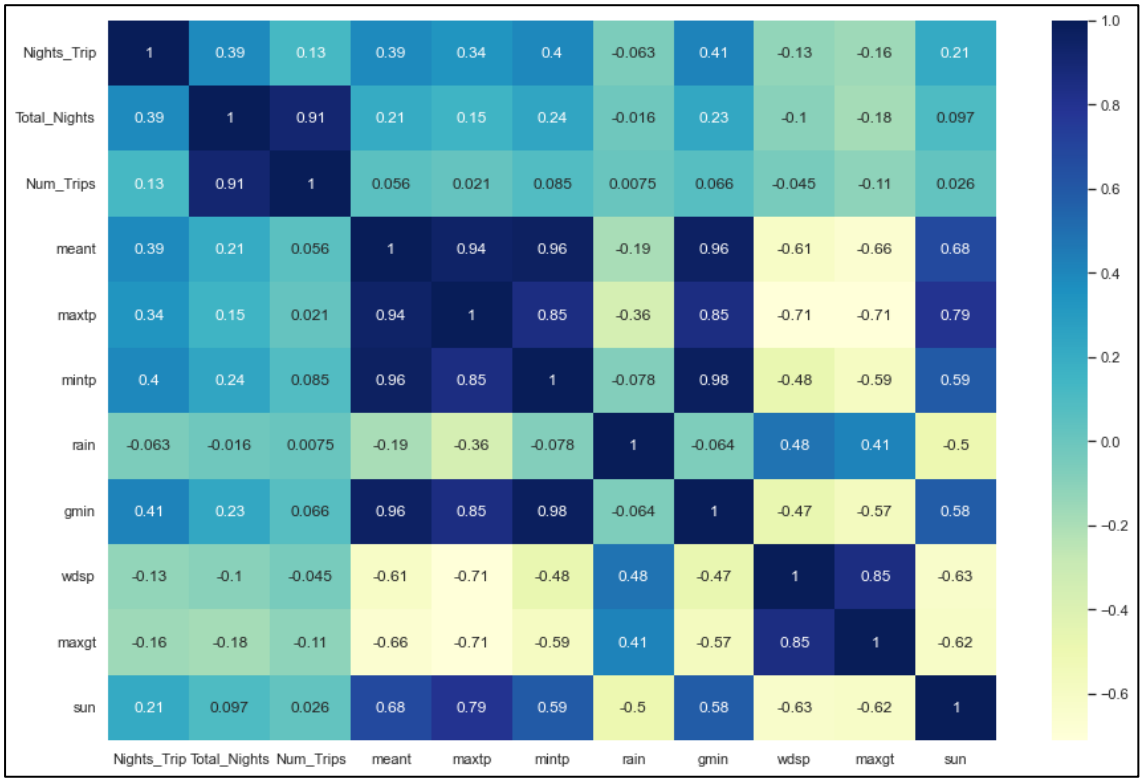
Median

- Median of Number of Trips is: 246.5
- Median of Number of Nights per Trip is: 2.5
- Median of Total of Nights is: 610.5

Trimmed Mean

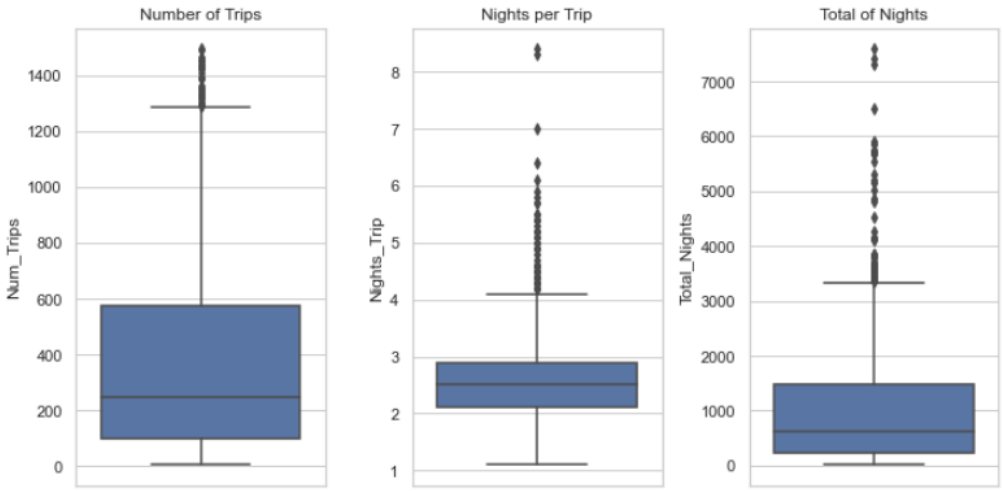
- Trimmed Mean of Number of Trips is: 293.26691729
- Trimmed Mean of Number of Nights per Trip is: 2.46710526
- Trimmed Mean of Total of Nights is: 744.37593985

Correlation Analysis



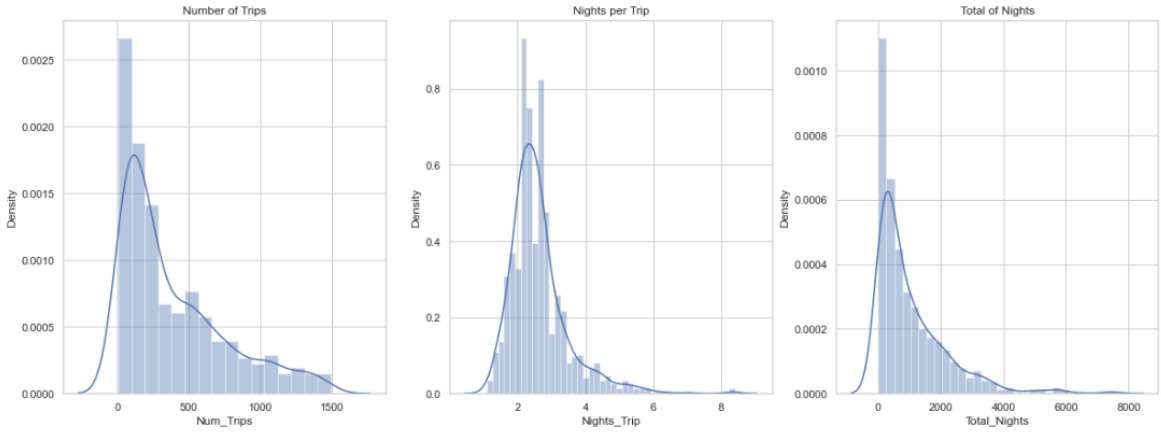
Data Distribution

In [63]:
```
# Plotting boxplots to understand the ocurreency of Outliers in the variables
fig, axs = plt.subplots(1,3,figsize=(10, 5))
sns.boxplot(y='Num_Trips', data=ddt, ax=axs[0])
axs[0].set_title('Number of Trips')
sns.boxplot(y='Nights_Trip', data=ddt, ax=axs[1])
axs[1].set_title('Nights per Trip')
sns.boxplot(y='Total_Nights', data=ddt, ax=axs[2])
axs[2].set_title('Total of Nights')
plt.tight_layout()
plt.show()
```



In [65]:
```
fig, axs = plt.subplots(1,3,figsize=(20, 7))
sns.distplot(ddt['Num_Trips'], hist = True,ax=axs[0])
axs[0].set_title('Number of Trips')
sns.distplot(ddt['Nights_Trip'], hist = True,ax=axs[1])
axs[1].set_title('Nights per Trip')
sns.distplot(ddt['Total_Nights'], hist = True,ax=axs[2])
axs[2].set_title('Total of Nights')
```

Out[65]:
```
Text(0.5, 1.0, 'Total of Nights')
```



Skewness

In [66]:
```
# Function to define the Skewness
def skewness_define(x,name):
    sk_x = x.skew()
    if sk_x < 0:
        if sk_x > -0.5:
            print(f'The Skewness of {name} is Negative and Fairly Symmetrical: {sk_x}')
        elif sk_x > -1:
            print(f'The Skewness of {name} is Negative and Moderately Skewed: {sk_x}')
        elif sk_x < -1:
            print(f'The Skewness of {name} is Negative and Heavily Skewed: {sk_x}')
    elif sk_x > 0:
        if sk_x < 0.5:
            print(f'The Skewness of {name} is Positive and Fairly Symmetrical: {sk_x}')
        elif sk_x < 1:
            print(f'The Skewness of {name} is Positive and Moderately Skewed: {sk_x}')
        elif sk_x > 1:
            print(f'The Skewness of {name} is Positive and Heavily Skewed: {sk_x}')
    else:
        print(f'The variable {name} is not Skew: {sk_x}')
```

- The Skewness of Number of Trips is Positive and Heavily Skewed: 1.1798514936950693
- The Skewness of Number of Nights is Positive and Heavily Skewed: 1.9450659186705377
- The Skewness of Total of Nights is Positive and Heavily Skewed: 2.079659745457055

Kurtosis

- The Kurtosis of Number of Trips is: 0.5334468226441031
- The Kurtosis of Number of Nights per Trip is: 6.721781037345404
- The Kurtosis of Total of Nights is: 5.740111250621173

Poisson Distribution

Considering the variables Number of Trips and Total of Nights, we are going to modelling the variables using the Poisson Distribution.

- **Number of Trips**

Parameters of the distribution

```
In [74]: # Calculing the mean of previous years and defining lambda
         mean_year_lambda_trips = mean_year_trips.mean()
```

Adjusting the distribution

```
In [75]: # Adjusting the Poisson Distribution based on the average of lambdas from previous years
         dist_previous_poisson_trips = poisson(mu = mean_year_lambda_trips)
```

Estimating the Probabilities of the Distribution

1. Estimating the probability of the Number of Trips in 2024 be more than 23k.

```
In [182]: 1 - dist_previous_poisson_trips.cdf(23000)
Out[182]: 0.015463254074295385
```

2. Estimating the probability of the Number of Trips in 2024 be at least 23k.

```
In [183]: dist_previous_poisson_trips.cdf(23000)
Out[183]: 0.9845367459257046
```

3. Estimating the probability of the Number of Trips in 2024 be less than 22.5k.

```
In [189]: dist_previous_poisson_trips.cdf(22500)
Out[189]: 0.12327370978794915
```

- **Total of Nights**

Parameters of the distribution

```
In [79]: # Calculing the mean of previous years and defining lambda
         mean_year_lambda_nights = mean_year_nights.mean()
```

Adjusting the distribution

```
In [80]: # Adjusting the Poisson Distribution based on the average of lambdas from previous years
         dist_previous_poisson_nights = poisson(mu = mean_year_lambda_nights)
```

Estimating the Probabilities of the Distribution

1. Estimating the probability of the Total of Nights of Trips in 2024 be more than 62k.

```
In [191]: 1 - dist_previous_poisson_nights.cdf(62000)
Out[191]: 0.07700909289334601
```

2. Estimating the probability of the Total of Nights of Trips in 2024 be less than 62k.

```
In [192]: dist_previous_poisson_nights.cdf(62000)
Out[192]: 0.922990907106654
```

3. Estimating the probability of the Total of Nights of Trips in 2024 be less than 61k.

```
In [195]: dist_previous_poisson_nights.cdf(61000)
Out[195]: 0.004589478952748032
```

Binomial Distribution

- **Number of Trips**

Parameters of the distribution

```
In [85]: n_trips = mean_year_trips.sum()
         p_trips = mean_year_trips.mean() / n_trips
```

Adjusting the distribution

```
In [86]: binom_trips = stats.binom(n_trips,p_trips)
```

Estimating the Probabilities of the Distribution

1. Estimating the expected number of trips in 2024.

```
In [87]: year = 2024
         trips_2024 = binom_trips.mean()

In [88]: trips_2024

Out[88]: 22674.933333333334
```

- **Total of Nights**

Parameters of the distribution

```
In [89]: n_nights = mean_year_nights.sum()
         p_nights = mean_year_nights.mean() / n_nights
```

Adjusting the distribution

```
In [90]: binom_nights = stats.binom(n_nights,p_nights)
```

Estimating the Probabilities of the Distribution

1. Estimating the expected number of trips in 2024.

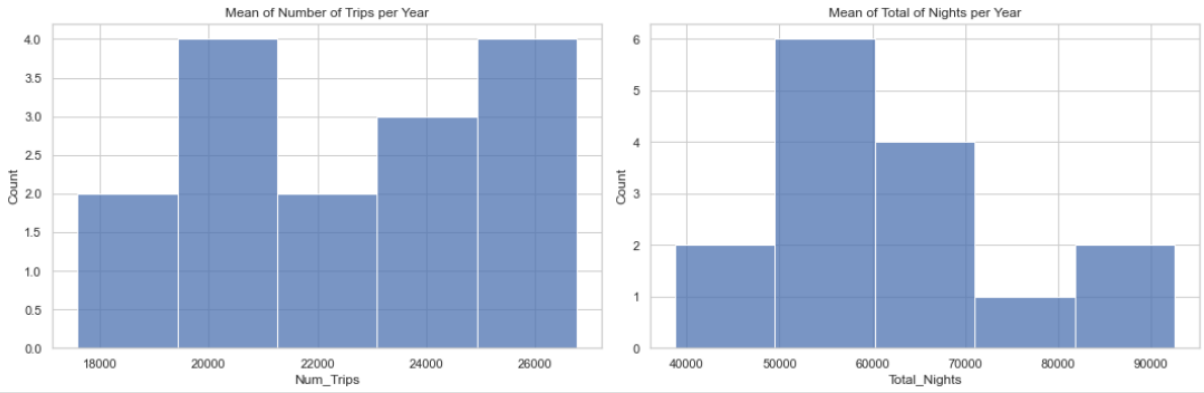
```
In [91]: year = 2024
         nights_2024 = binom_nights.mean()

In [92]: nights_2024
Out[92]: 61646.4
```

The Central Limit Theorem

Considering the variables previously calculated:

- Mean of Number of Trips per Year
- Mean of Total of Nights per Year

```
In [93]: # Plotting the graph of variables
         fig, axes = plt.subplots(1,2,figsize = (15,5))
         sns.histplot(mean_year_trips, ax = axes[0])
         axes[0].set_title('Mean of Number of Trips per Year')
         sns.histplot(mean_year_nights, ax = axes[1])
         axes[1].set_title('Mean of Total of Nights per Year')
         plt.tight_layout()
```



- The parameters of the variable Mean of **NUMBER OF TRIPS** per Year is:

  - **Poisson Distribution:**
    - **Lambda:** 22674.933333333334

  - **Binomial Distribution:**
    - **n:** 340124.0
    - **p:** 0.06666666666666667

- The parameters of the variable Mean of **TOTAL OF NIGHTS** per Year is:

  - **Poisson Distribution:**
    - **Lambda:** 61646.4

  - **Binomial Distribution:**
    - **n:** 924696.0
    - **p:** 0.06666666666666667

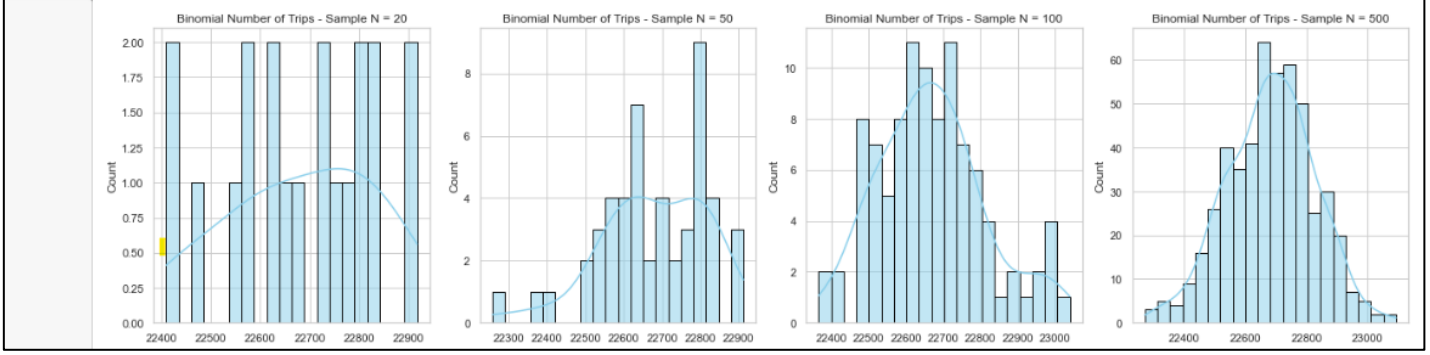
Generating Random Samples for the Variables

Generating samples with different number of observations considering the parameters and the distributions Poisson and Binomial for the variables.

```
In [96]: random.seed(199)

In [97]: # Defining the n of the samples
         n_samples = [20,50,100,500]

In [98]:
         list_trips_binom= []
         list_trips_poisson = []
         list_nights_binom= []
         list_nights_poisson = []
         for i in n_samples:
             samples_trips_binom = np.random.binomial(n_trips,p_trips, size = (1,i))
             samples_trips_pois = np.random.poisson(mean_year_lambda_trips, size = (1,i))
             samples_nights_binom = np.random.binomial(n_nights,p_nights, size = (1,i))
             samples_nights_pois = np.random.poisson(mean_year_lambda_nights, size = (1,i))
             list_trips_binom.append(samples_trips_binom)
             list_trips_poisson.append(samples_trips_pois)
             list_nights_binom.append(samples_nights_binom)
             list_nights_poisson.append(samples_nights_pois)
```

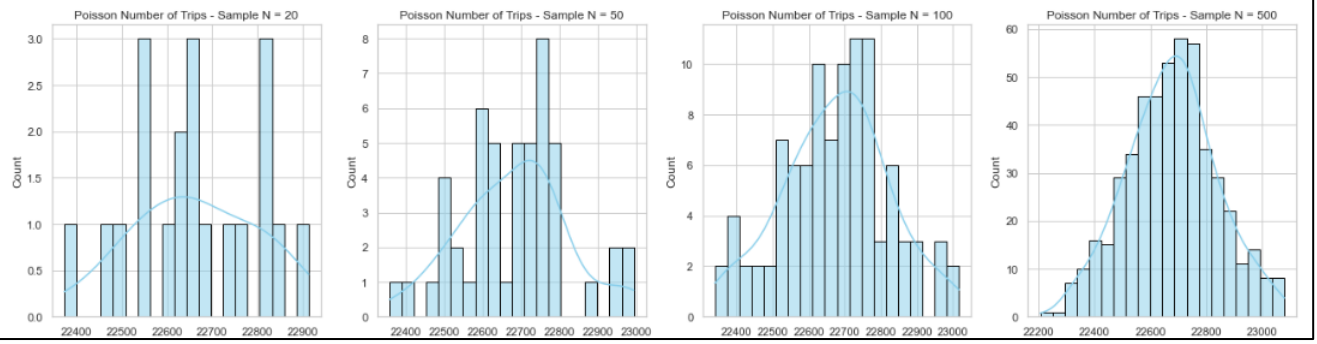
- **Number of Trips - Samples Generated with Binomial Distribution**

```
In [99]: plt.figure(figsize=(18,9))

         for i, sample in enumerate(list_trips_binom):
             plt.subplot(2,len(list_trips_binom),i+1)
             sns.histplot(sample.flatten(), bins=20, color='skyblue', edgecolor='black', kde=True)
             #plt.hist(sample.flatten(),bins = 20, color = 'skyblue',edgecolor = 'black')
             plt.title(f'Binomial Number of Trips - Sample N = {n_samples[i]}')

         plt.tight_layout()
```



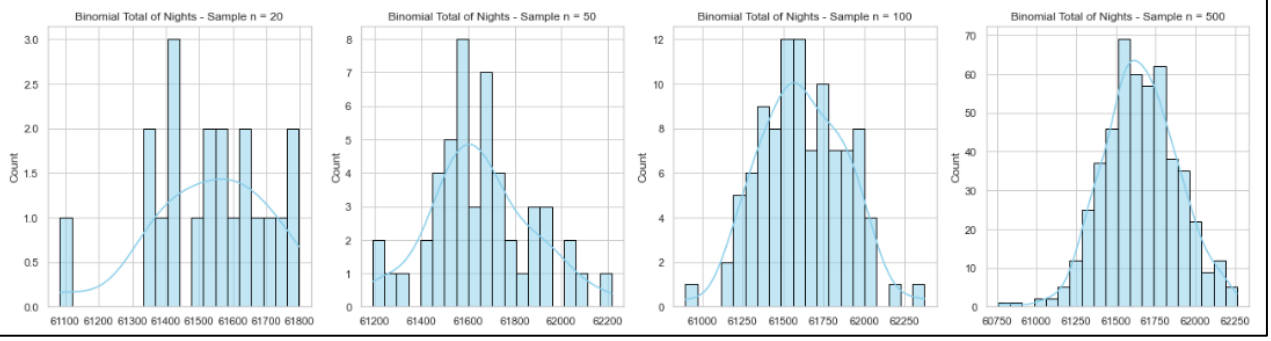
- **Number of Trips - Samples Generated with Poisson Distribution**

In [100]:
```
plt.figure(figsize=(18,9))

for i, sample in enumerate(list_trips_poisson):
    plt.subplot(2,len(list_trips_poisson),i+1)
    sns.histplot(sample.flatten(), bins=20, color='skyblue', edgecolor='black', kde=True)
    #plt.hist(sample.flatten(),bins = 20, color = 'skyblue',edgecolor = 'black')
    plt.title(f'Poisson Number of Trips - Sample N = {n_samples[i]}')

plt.tight_layout()
```



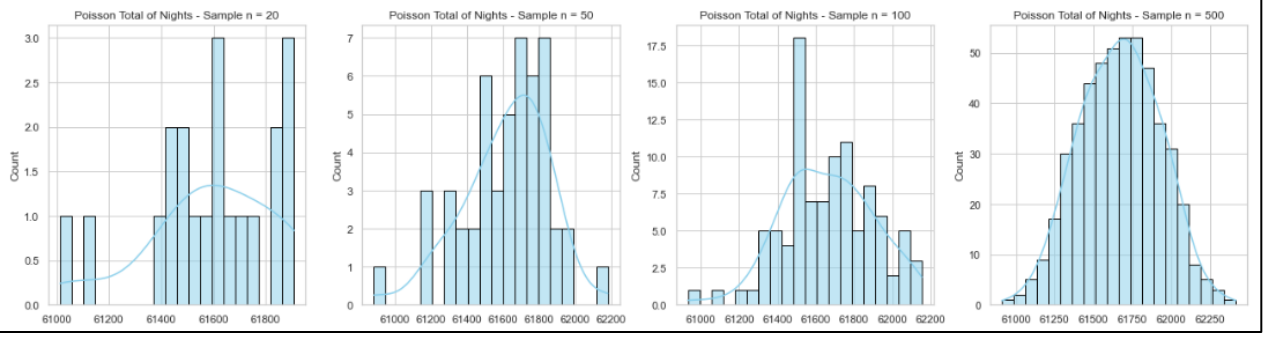
- **Total of Nights - Samples Generated with Binomial Distribution**

In [101]:
```
plt.figure(figsize=(18,9))

for i, sample in enumerate(list_nights_binom):
    plt.subplot(2,len(list_nights_binom),i+1)
    sns.histplot(sample.flatten(), bins=20, color='skyblue', edgecolor='black', kde=True)
    #plt.hist(sample.flatten(),bins = 20, color = 'skyblue',edgecolor = 'black')
    plt.title(f'Binomial Total of Nights - Sample n = {n_samples[i]}')

plt.tight_layout()
```



- **Total of Nights - Samples Generated with Poisson Distribution**

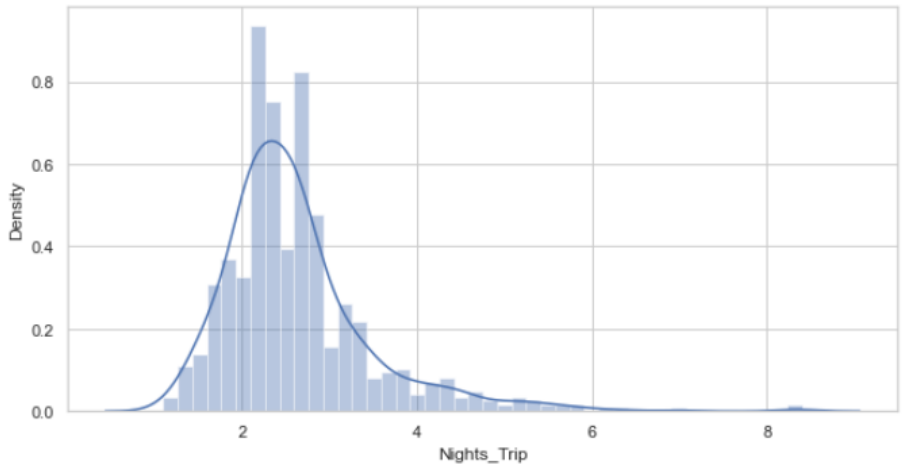
```
In [102]: plt.figure(figsize=(18,9))

for i, sample in enumerate(list_nights_poisson):
    plt.subplot(2,len(list_nights_poisson),i+1)
    sns.histplot(sample.flatten(), bins=20, color='skyblue', edgecolor='black', kde=True)
    #plt.hist(sample.flatten(),bins = 20, color = 'skyblue',edgecolor = 'black')
    plt.title(f'Poisson Total of Nights - Sample n = {n_samples[i]}')

plt.tight_layout()
```



Normal Distribution

Nights per Trip - Checking the Distribution of the Variable

```
In [103]: # Plotting the histogram of the variable Nights per Trip
          plt.figure(figsize=(10,5))
          sns.distplot(ddt['Nights_Trip'])

Out[103]: <AxesSubplot:xlabel='Nights_Trip', ylabel='Density'>
```



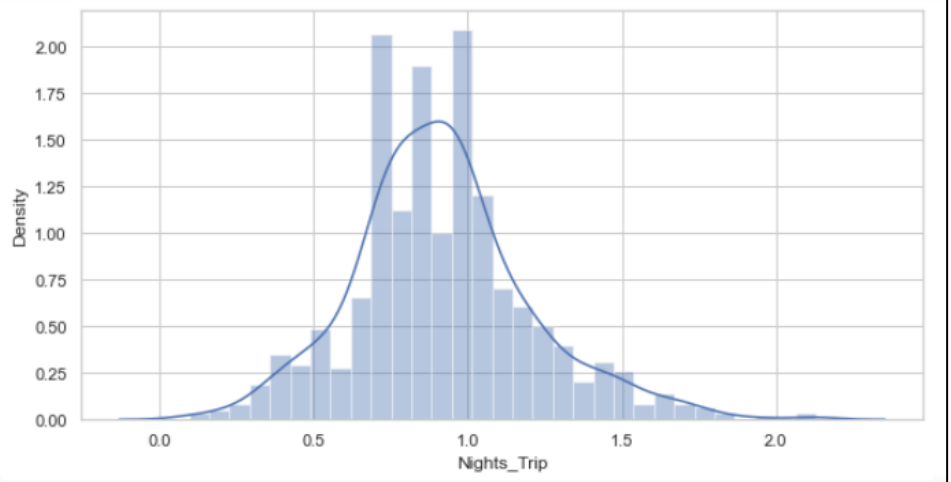
Applying Log-Transformation

Using log transformation to transform in a symmetric variable:

```
In [198]: Nights_Trip_log = np.log(ddt['Nights_Trip'])
```

Plotting and checking the variable Nights per Trip after the log-transformation:

```
In [105]: plt.figure(figsize=(10,5))
          sns.distplot(Nights_Trip_log)
```



Parameters of the distribution

```
In [106]: mean_nights_trip = ddt.groupby('Year')['Nights_Trip'].mean().mean()
          std_nights_trip = ddt.groupby('Year')['Nights_Trip'].std().mean()
```

Adjusting the distribution

```
In [108]: nights_trip_normal = norm(loc = mean_nights_trip, scale = std_nights_trip)
```

Estimating the Probabilities of the Distribution

1. Estimating the probability of the number of nights be more than 3 per trip in 2024.

```
In [109]: ### Using the variable Nights per Trip with log-transformation
          prob_nights_trip_2024 = 1 - nights_trip_normal.cdf(3)
          prob_nights_trip_2024

Out[109]: 0.26162035831838004
```

2. Estimating the probability of the number of nights be more than 4 per trip in 2024.

```
In [110]: ### Using the variable Nights per Trip with log-transformation
          prob_nights_trip_2024 = 1 - nights_trip_normal.cdf(4)
          prob_nights_trip_2024

Out[110]: 0.021729086285877552
```

3. Estimating the probability of the number of nights be less than 2 per trip in 2024.

```
In [111]: ### Using the variable Nights per Trip with log-transformation
          prob_nights_trip_2024 = nights_trip_normal.cdf(2)
          prob_nights_trip_2024

Out[111]: 0.22887293432963463
```

MACHINE LEARNING – FEATURE ENGINEERING

Feature Engineer for Regression

Creating some interactions between the variables

```
In [114]: # Creating interactions between variables to understand the effect
          dd_reg['maxmin'] = dd_reg['maxtp'] * dd_reg['mintp']
          dd_reg['min_wind'] = dd_reg['mintp'] * dd_reg['wdsp']
          dd_reg['rain_wind'] = dd_reg['rain'] * dd_reg['wdsp']
          dd_reg['gmin_temp'] = dd_reg['gmin'] * dd_reg['meant']
```

Variable encoding

```
In [118]: # Encoding the variables Destiny and Reason

          label_encoder = LabelEncoder()
          ddcod['Destiny'] = label_encoder.fit_transform(ddcod['Destiny'])
          ddcod['Reason'] = label_encoder.fit_transform(ddcod['Reason'])
```

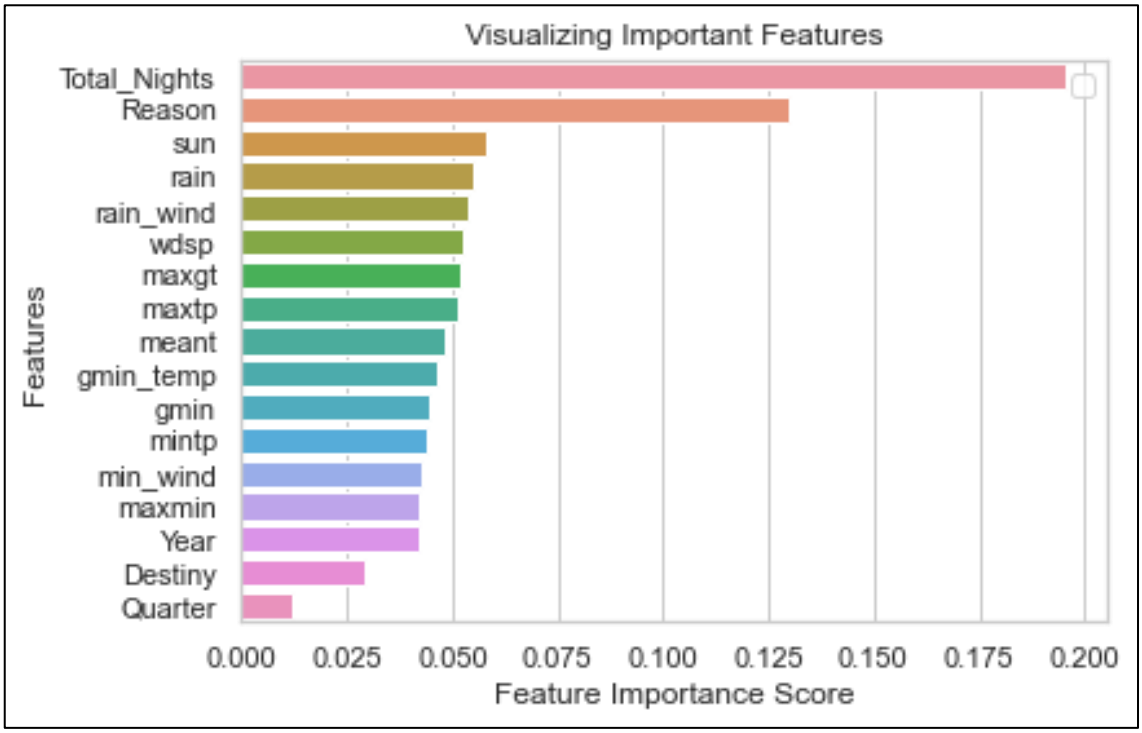
Splitting the Data

```
In [120]: # Defining the variables
          X = dd_lr.iloc[:, :-1]
          y = dd_lr['Num_Trips']
```

```
In [121]: # Split the data into 70% and 30% by using a parameter test_size = 30
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

          X.shape, y.shape, X_train.shape, X_test.shape, y_train.shape, y_test.shape
Out[121]: ((884, 17), (884,), (618, 17), (266, 17), (618,), (266,))
```

Feature Importance for ML

```
In [122]: clf = RandomForestClassifier(n_estimators = 100)

In [123]: #Train the model using the training sets y_pred=clf.predict(X_test)
          clf.fit(X_train,y_train)
Out[123]: ▾ RandomForestClassifier
          RandomForestClassifier()
```



Feature Engineer for Classification

Variable encoding

- **Transforming variables**

```
In [128]: # Encoding the variables Destiny and Reason

label_encoder = LabelEncoder()
dd_class['Destiny'] = label_encoder.fit_transform(dd_class['Destiny'])
dd_class['Reason'] = label_encoder.fit_transform(dd_class['Reason'])
dd_class
```

Out[128]:

	Destiny	Year	Quarter	Reason	Total_Nights	Num_Trips	meant	maxtp	mintp	rain	gmin	wdsp	maxgt	sun
216	0	2005	1	0	1482.0	501.0	7.050000	14.683333	-1.800000	91.033333	-5.883333	10.866667	53.666667	74.600000
217	7	2005	1	0	2688.0	990.0	7.408333	14.208333	-1.408333	74.266667	-5.033333	11.408333	48.916667	67.377778
218	8	2005	1	0	4170.0	1490.0	7.183333	13.400000	-1.466667	51.366667	-4.783333	12.533333	48.833333	60.500000
220	0	2005	1	1	162.0	65.0	7.050000	14.683333	-1.800000	91.033333	-5.883333	10.866667	53.666667	74.600000
221	1	2005	1	1	100.0	45.0	6.500000	14.500000	-3.166667	46.300000	-6.133333	11.166667	49.333333	60.500000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1539	2	2023	3	3	153.0	101.0	14.953333	24.473333	6.173333	131.886667	2.733333	7.286667	37.466667	145.766667
1541	6	2023	3	3	203.0	82.0	15.611111	23.033333	8.766667	140.822222	5.455556	9.844444	45.555556	138.400000
1542	8	2023	3	3	654.0	263.0	15.177778	23.433333	7.255556	134.888889	3.633333	9.288889	43.888889	145.766667
1543	2	2023	3	4	1036.0	491.0	14.953333	24.473333	6.173333	131.886667	2.733333	7.286667	37.466667	145.766667

- **Splitting Number of Trips by classes**

```
In [129]: # Calculing the Quartis
quantil_25 = dd_class['Num_Trips'].quantile(0.25)
quantil_50 = dd_class['Num_Trips'].quantile(0.50)
quantil_75 = dd_class['Num_Trips'].quantile(0.75)
```

```
In [130]: # Defining the intervals and class labels basedin the Quartis
intervals = [0, quantil_25, quantil_50, quantil_75 , float('inf')]
labels = ['Low','Medium', 'High', 'Very High']
```

```
In [200]: # Defining the classes
trip_classes = pd.cut(dd_class["Num_Trips"], bins=intervals, labels=labels)
```

```
In [134]: dd_class['trip_classes'] = label_encoder.fit_transform(dd_class['trip_classes'])
```

Splitting the Data

```
In [136]: # Defining the variables
X_class = dd_class.iloc[:, :-1]
y_class = dd_class['trip_classes']
```

```
In [137]: # Split the data into 70% and 30% by using a parameter test_size = 30
X_train2, X_test2, y_train2, y_test2 = train_test_split(X_class, y_class, test_size = 0.3, random_state = 0)

X_class.shape, y_class.shape, X_train2.shape, X_test2.shape, y_train2.shape, y_test2.shape
```

```
Out[137]: ((884, 13), (884,), (618, 13), (266, 13), (618,), (266,))
```

Normalizing the Data

```
In [138]: # Create an object sc by calling a method StandardScaler
          sc = StandardScaler()

          # Normalizing all the variables,expect those ones we need to interpretate with a specific value: Ex.: Year
          var_norm = ['Total_Nights', 'meant', 'maxtp', 'mintp', 'rain', 'gmin', 'wdsp', 'maxgt', 'sun']
          X_train_normalized = X_train2
          X_test_normalized = X_test2

          # Normalize the data by calling a fit_transform() method
          X_train_normalized[var_norm] = sc.fit_transform(X_train2[var_norm])
          X_test_normalized[var_norm] = sc.transform(X_test2[var_norm])
```

```
In [139]: X_train_normalized = pd.DataFrame(X_train_normalized, columns = X.columns)
          X_train_normalized.head()
```

Out[139]:

	Destiny	Year	Quarter	Reason	Total_Nights	meant	maxtp	mintp	rain	gmin	wdsp	maxgt	sun
1123	8	2018	1	3	-0.417620	-1.591842	-1.343351	-1.534862	0.310858	-1.306177	0.906500	0.656379	-0.728270
381	3	2006	3	0	-0.608059	1.784468	1.953987	1.156982	-0.924834	1.512658	-1.365693	-0.526292	1.028714
981	8	2015	4	2	0.713604	-0.120660	-0.465722	-0.098150	2.132111	0.234807	1.389946	0.627357	-1.221447
685	0	2008	4	4	-0.386027	-0.917991	-0.961291	-0.853140	1.003790	-0.975291	0.467946	0.866793	-1.125118
374	8	2006	2	3	-0.505380	0.371498	0.342694	0.301645	-1.409111	0.321468	-0.029312	-0.591593	1.615506

MACHINE LEARNING – MODELS

Regression

Linear Regression

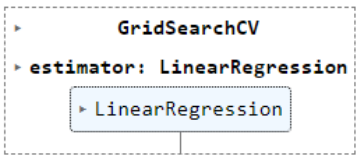
Building the Model

```
In [140]: parameters = {'fit_intercept': [True, False],
                        'n_jobs': [None, -1]}
```

```
In [141]: reg_mod = GridSearchCV(estimator=LinearRegression(),
                                 param_grid=parameters,
                                 scoring=None,
                                 cv=5)
```

```
In [142]: # Training the model
          reg_mod.fit(X_train,y_train)
```

Out[142]:



Evaluating the Model

```
In [143]: print(f"Training set score: {reg_mod.score(X_train, y_train)}")
          print(f"Test set score: {reg_mod.score(X_test, y_test)}")

          Training set score: 0.8858629126786245
          Test set score: 0.8740816797867721
```

Cross Validation

```
In [144]: # Function to calculate Cross Validation with differents CV's
          crosses_scores = []
          def cross_val(x,y,mod,cv,crosses_scores):
              crosses = []
              for i in list(range(0,len(cv))):
                  cross_reg = cross_val_score(mod, x, y, cv=cv[i], scoring= None,verbose=0)
                  crosses.append(cross_reg.mean())
                  print(f'cv: {cv[i]}     score: {cross_reg.mean()}')
              crosses_scores.append(max(crosses))

In [145]: cv = [5,10,20,40]
          cross_val(X,y,reg_mod,cv,crosses_scores)

          cv: 5     score: 0.8609584974783907
          cv: 10     score: 0.8638492796248475
          cv: 20     score: 0.8631369712200918
          cv: 40     score: 0.8601517960448433
```

Ridge Regression

Building the Model

```
In [146]: # list of alphas parameters to tune
          params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
           0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
           4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}

In [147]: ridge = Ridge()

In [148]: folds = 5
          ridge_mod = GridSearchCV(estimator = ridge,
                                   param_grid = params,
                                   scoring= None,
                                   cv = folds,
                                   return_train_score=True,
                                   verbose = 1)
          ridge_mod.fit(X_train, y_train)

          Fitting 5 folds for each of 28 candidates, totalling 140 fits

Out[148]: GridSearchCV
          estimator: Ridge
          Ridge
```

Evaluating the Model

```
In [149]: print(f"Training set score: {ridge_mod.score(X_train, y_train)}")
          print(f"Test set score: {ridge_mod.score(X_test, y_test)}")

          Training set score: 0.8858552319319623
          Test set score: 0.8740632296642642
```

Cross Validation

```
In [150]: cv = [5,10,20,40]
          cross_val(X,y,ridge_mod,cv,crosses_scores)
```

Lasso Regression

Building the Model

```
In [151]: # list of alphas parameters to tune
          params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
           0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
           4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}

In [152]: lasso = Lasso()

In [153]: # cross validation
          lasso_mod = GridSearchCV(estimator = lasso,
                                   param_grid = params,
                                   scoring= None,
                                   cv = folds,
                                   return_train_score=True,
                                   verbose = 1)

In [154]: lasso_mod.fit(X_train, y_train)

          Fitting 5 folds for each of 28 candidates, totalling 140 fits

Out[154]: GridSearchCV
          estimator: Lasso
          Lasso
```

Evaluating the Model

```
In [155]: print(f"Training set score: {lasso_mod.score(X_train, y_train)}")
          print(f"Test set score: {lasso_mod.score(X_test, y_test)}")

          Training set score: 0.8856766445191576
          Test set score: 0.8739518584092244
```

Cross Validation

```
In [156]: cv = [5,10,20,40]
          cross_val(X,y,lasso_mod,cv,crosses_scores)
```

Classification

Random Forest

Building the Model

```
In [163]: # Parameters
          tuned_params = {'n_estimators': [100, 200, 300, 400, 500],
                          'min_samples_split': [2, 5, 10],
                          'min_samples_leaf': [1, 2, 4]}

          # Model with RandomizedSearchCV
          forest_mod = RandomizedSearchCV(RandomForestClassifier(),
                                          tuned_params,
                                          n_iter = 15,
                                          scoring = 'accuracy',
                                          n_jobs  = -1)

In [164]: forest_mod.fit(X_train2, y_train2)

Out[164]: RandomizedSearchCV
          estimator: RandomForestClassifier
          RandomForestClassifier

In [165]: forest_mod.best_estimator_

Out[165]: RandomForestClassifier
          RandomForestClassifier(min_samples_split=10, n_estimators=200)
```

Prediction

```
In [166]: # Predictions with Test data
          forest_mod_pred = pd.Series(forest_mod.predict(X_test2))
```

Evaluating the Model

```
In [167]: print("Accuracy:",metrics.accuracy_score(y_test2, forest_mod_pred))

          Accuracy: 0.8909774436090225

In [168]: confusion_matrix(y_test2, forest_mod_pred)

Out[168]: array([[64,  0,  5,  2],
                 [ 0, 52, 11,  0],
                 [ 2,  4, 62,  0],
                 [ 5,  0,  0, 59]], dtype=int64)
```

Cross Validation

```
In [169]: cv = [5,10,20,40]
          cross_val(X_class,y_class,forest_mod,cv,crosses_scores)

          cv: 5     score: 0.8449922958397534
          cv: 10     score: 0.855209397344229
          cv: 20     score: 0.8678282828282828
          cv: 40     score: 0.8699604743083004
```

SVM

Building the Model

```
In [170]: # Select hyperparameters
          def svc_param_selection(X, y, nfolds):
              Cs = [0.001, 0.01, 0.1, 1]
              gammas = [0.001, 0.01, 0.1, 1]
              param_grid = {'C': Cs, 'gamma' : gammas}
              grid_search = GridSearchCV(SVC(kernel = 'rbf'), param_grid, cv = nfolds)
              grid_search.fit(X_train2, y_train2)
              grid_search.best_params_
              return grid_search.best_params_

          # Aplica a função
          svc_param_selection(X_train2, y_train2, 5)

Out[170]: {'C': 1, 'gamma': 0.01}

In [171]: svm_mod = SVC(C = 1, gamma = 0.01, probability = True)

In [172]: svm_mod.fit(X_train2, y_train2)

Out[172]: SVC
          SVC(C=1, gamma=0.01, probability=True)
```

Prediction

```
In [173]: # Predictions with Test data
          svm_mod_pred = svm_mod.predict(X_test2)
```

Evaluating the Model

```
In [174]: print("Accuracy:",metrics.accuracy_score(y_test2, svm_mod_pred))

          Accuracy: 0.631578947368421

In [175]: confusion_matrix(y_test2, svm_mod_pred)

Out[175]: array([[41, 15, 14,  1],
                 [ 2, 48, 13,  0],
                 [17, 23, 28,  0],
                 [12,  1,  0, 51]], dtype=int64)
```

Cross Validation

```
In [176]: cv = [5,10,20,40]
          cross_val(X_class,y_class,svm_mod,cv,crosses_scores)

          cv: 5     score: 0.45127118644067804
          cv: 10     score: 0.48632533197139943
          cv: 20     score: 0.4976767676767677
          cv: 40     score: 0.5156126482213439
```

Comparing Models

	Name	Algorithm	Type	Metric	Score	Cross Validation
0	Model 01	Linear Regression	Regression	R-Square	0.874082	0.863849
1	Model 02	Ridge Regression	Regression	R-Square	0.874063	0.863691
2	Model 03	Lasso Regression	Regression	R-Square	0.873952	0.862630
3	Model 04	Random Forest	Classification	Accuracy	0.890977	0.869960
4	Model 05	SVM	Classification	Accuracy	0.631579	0.515613