bookings-2

November 8, 2024

[68]:
```
from pathlib import Path
import pandas as pd

# Use Path to resolve the file path
file_path = Path(r'C:\Users\divaa\OneDrive\Desktop\pri\Bliend\Bliend␣
 ↪dataset\hotel_bookings.csv')

# Load the dataset
df = pd.read_csv(file_path)
df
df.head()
```

[68]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	\
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	
3	Resort Hotel	0	13	2015	July	
4	Resort Hotel	0	14	2015	July	

	arrival_date_week_number	arrival_date_day_of_month	\
0	27	1	
1	27	1	
2	27	1	
3	27	1	
4	27	1	

	stays_in_weekend_nights	stays_in_week_nights	adults	…	deposit_type	\
0	0	0	2	…	No Deposit	
1	0	0	2	…	No Deposit	
2	0	1	1	…	No Deposit	
3	0	1	1	…	No Deposit	
4	0	2	2	…	No Deposit	

	agent	company	days_in_waiting_list	customer_type	adr	\
0	NaN	NaN	0	Transient	0.0	
1	NaN	NaN	0	Transient	0.0	
2	NaN	NaN	0	Transient	75.0	

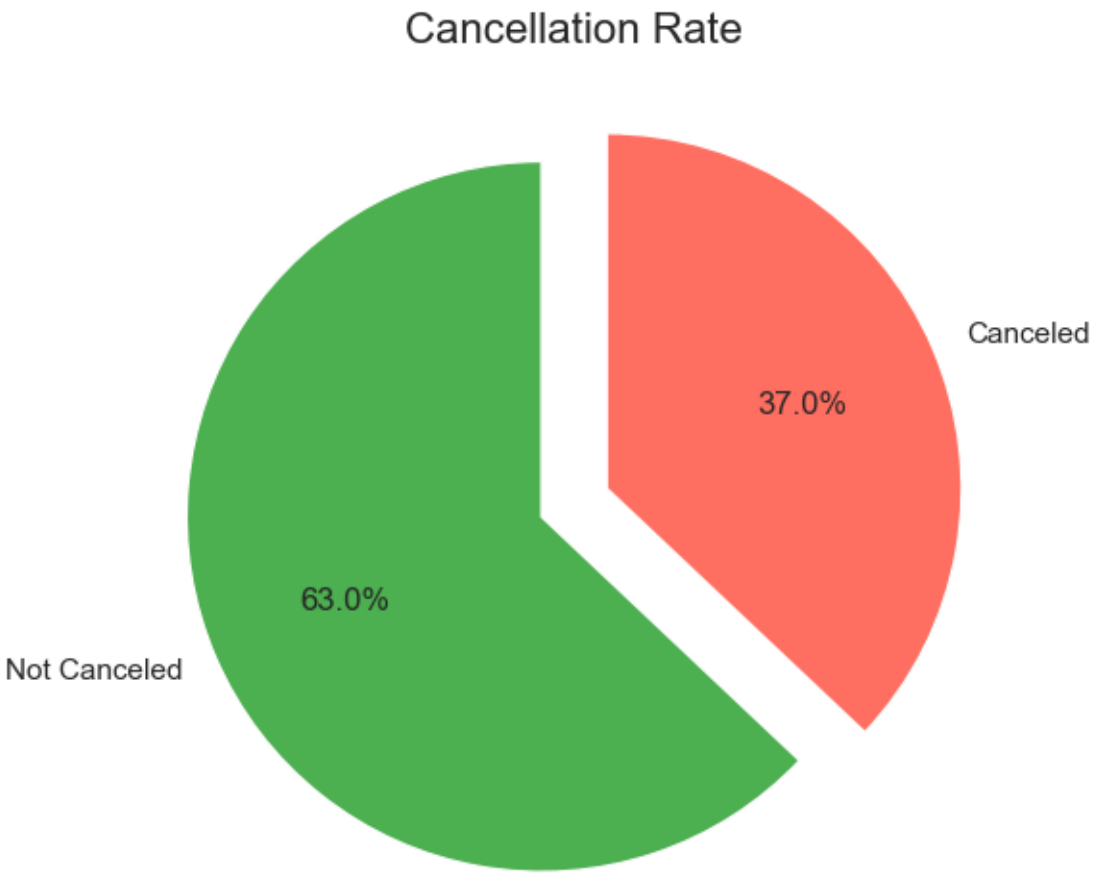
```
3  304.0     NaN                    0     Transient  75.0
4  240.0     NaN                    0     Transient  98.0

   required_car_parking_spaces  total_of_special_requests  reservation_status  \
0                            0                          0           Check-Out
1                            0                          0           Check-Out
2                            0                          0           Check-Out
3                            0                          0           Check-Out
4                            0                          1           Check-Out

  reservation_status_date
0              2015-07-01
1              2015-07-01
2              2015-07-02
3              2015-07-02
4              2015-07-03

[5 rows x 32 columns]
```

[54]:
```python
# Loop through and print each column name
for col in df.columns:
    print(col)
```

```
hotel
is_canceled
lead_time
arrival_date_year
arrival_date_month
arrival_date_week_number
arrival_date_day_of_month
stays_in_weekend_nights
stays_in_week_nights
adults
children
babies
meal
country
market_segment
distribution_channel
is_repeated_guest
previous_cancellations
previous_bookings_not_canceled
reserved_room_type
assigned_room_type
booking_changes
deposit_type
agent
```

```
company
days_in_waiting_list
customer_type
adr
required_car_parking_spaces
total_of_special_requests
reservation_status
reservation_status_date
```

[58]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set a modern style
sns.set(style="whitegrid", palette="muted")

# Cancellation rate
cancellation_counts = df['is_canceled'].value_counts(normalize=True) * 100
labels = ['Not Canceled', 'Canceled']

# Create a 3D exploded pie chart
plt.figure(figsize=(6, 6))
explode = (0.1, 0.1)  # "explode" the slices
plt.pie(cancellation_counts, labels=labels, autopct='%1.1f%%', startangle=90,␣
 ↪colors=['#4CAF50', '#FF6F61'], explode=explode)
plt.title('Cancellation Rate', fontsize=16)
plt.show()
```



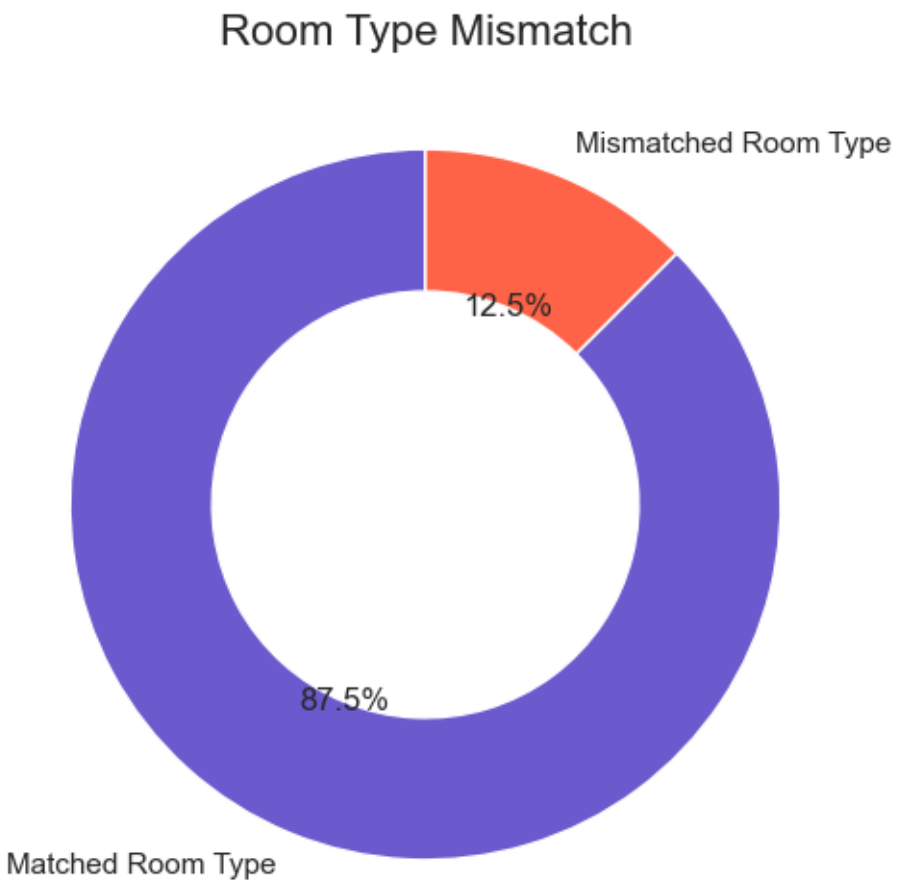
[59]:
```
cancellation_by_hotel = df.groupby('hotel')['is_canceled'].mean() * 100
print(cancellation_by_hotel)
```

```
hotel
City Hotel      41.726963
Resort Hotel    27.763355
Name: is_canceled, dtype: float64
```

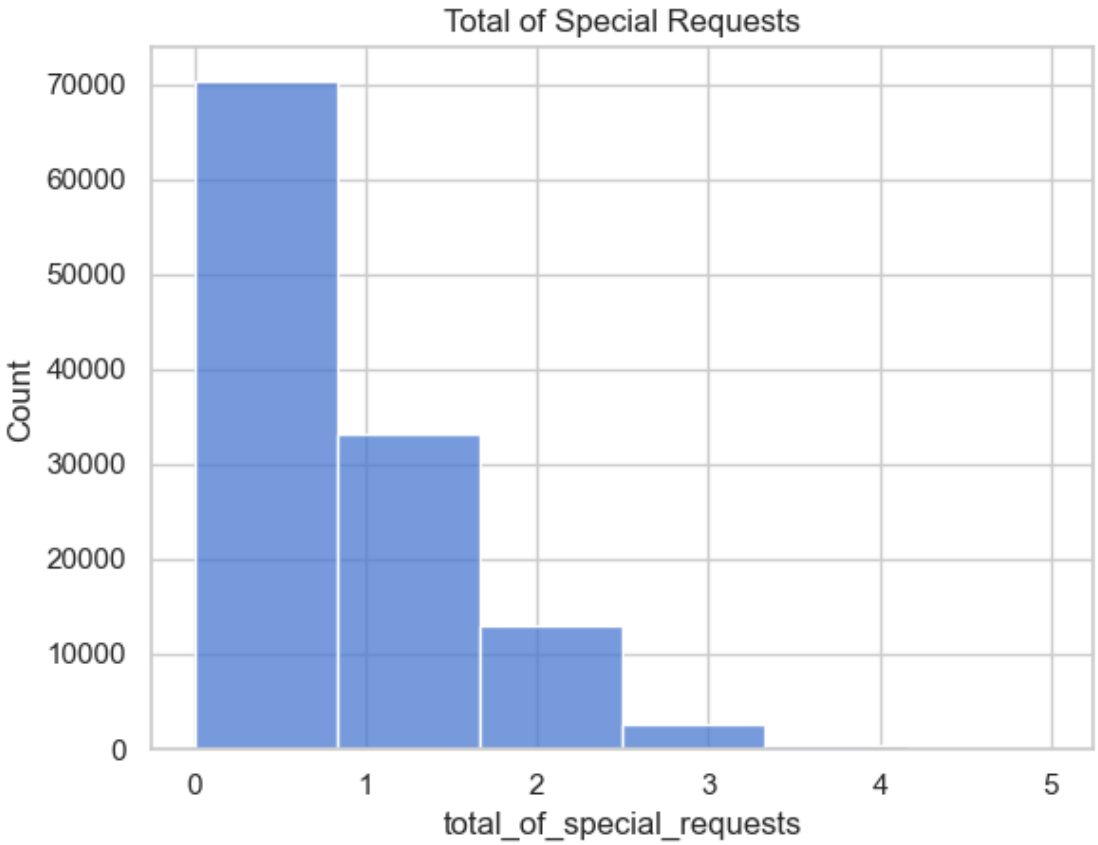
[62]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set a modern style
sns.set(style="whitegrid", palette="muted")

# Ensure the 'hotel' column is treated as a string
df['hotel'] = df['hotel'].astype(str)
```

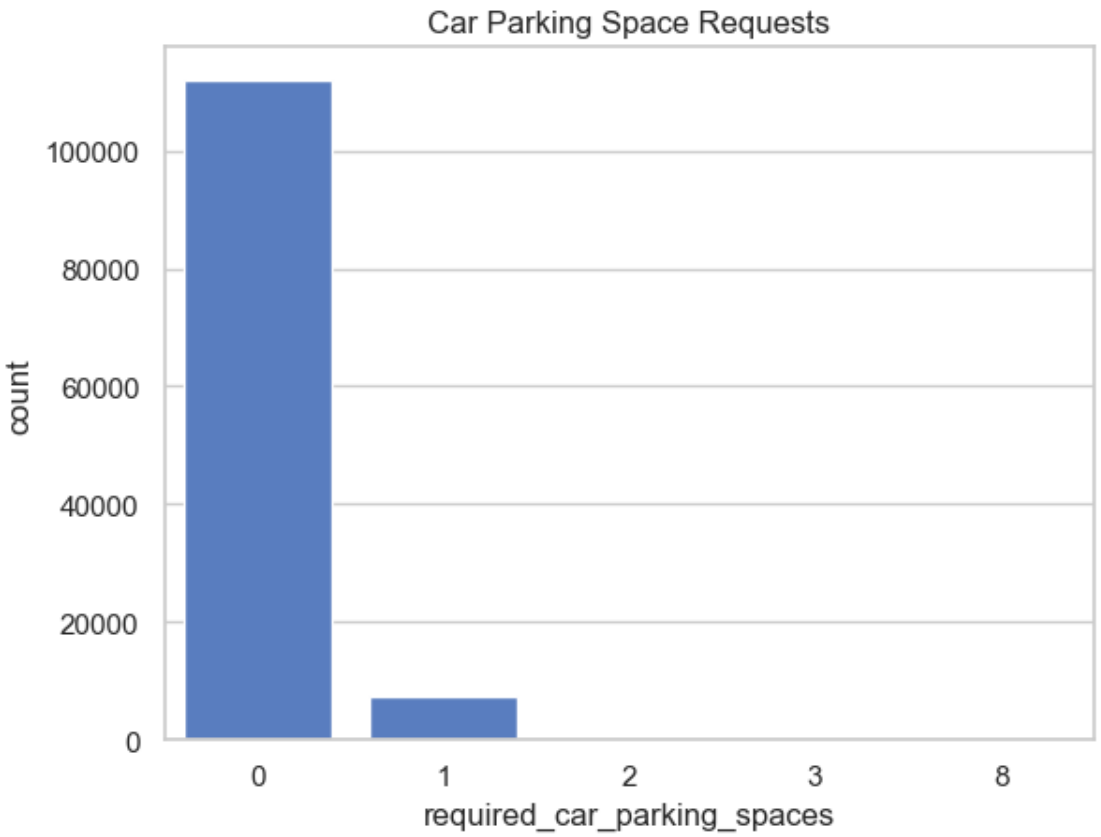
```
# Bar plot for cancellation rate by hotel type with hue
plt.figure(figsize=(8, 6))
sns.barplot(x='hotel', y='is_canceled', data=df, estimator=lambda x: len(x) /␣
 ↪len(df) * 100, hue='is_canceled', palette='coolwarm')

# Title and labels
plt.title('Cancellation Rate by Hotel Type', fontsize=18)
plt.xlabel('Hotel Type', fontsize=14)
plt.ylabel('Cancellation Rate (%)', fontsize=14)
plt.ylim(0, 100)

# Ensure hotel names are correctly shown as labels on the x-axis
plt.xticks(rotation=45, ha='right')

plt.show()
```

[66]:
```
import matplotlib.pyplot as plt

# Set the figure size
plt.figure(figsize=(6, 6))

# Calculate room type mismatch
room_type_mismatch = (df['reserved_room_type'] != df['assigned_room_type']).
 ↪value_counts(normalize=True) * 100
labels = ['Matched Room Type', 'Mismatched Room Type']

# Create a donut chart with unique colors
plt.pie(room_type_mismatch, labels=labels, autopct='%1.1f%%', startangle=90,
        colors=['#6A5ACD', '#FF6347'], wedgeprops={'width': 0.4})  # Unique␣
 ↪colors

# Title and display the plot
plt.title('Room Type Mismatch', fontsize=16)
plt.show()
```



[12]:
```
# Analyze special requests
sns.histplot(df['total_of_special_requests'], bins=6)
plt.title('Total of Special Requests')
plt.show()

# Analyze parking space demand
sns.countplot(x='required_car_parking_spaces', data=df)
plt.title('Car Parking Space Requests')
plt.show()
```

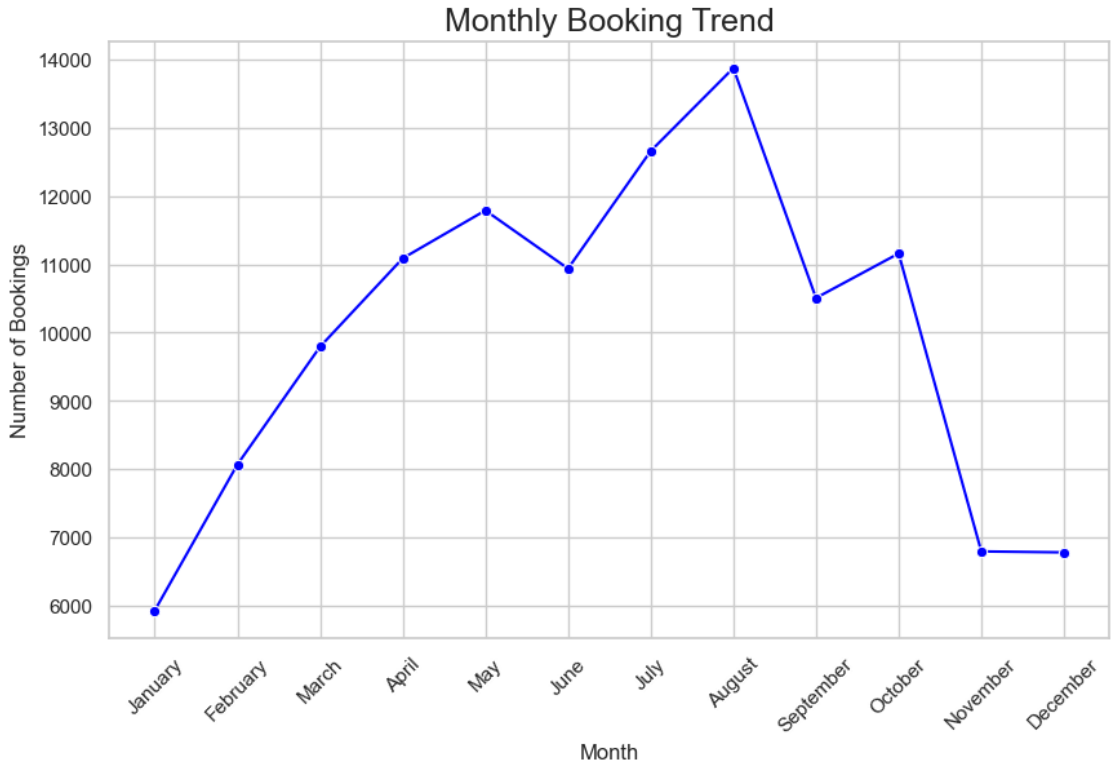


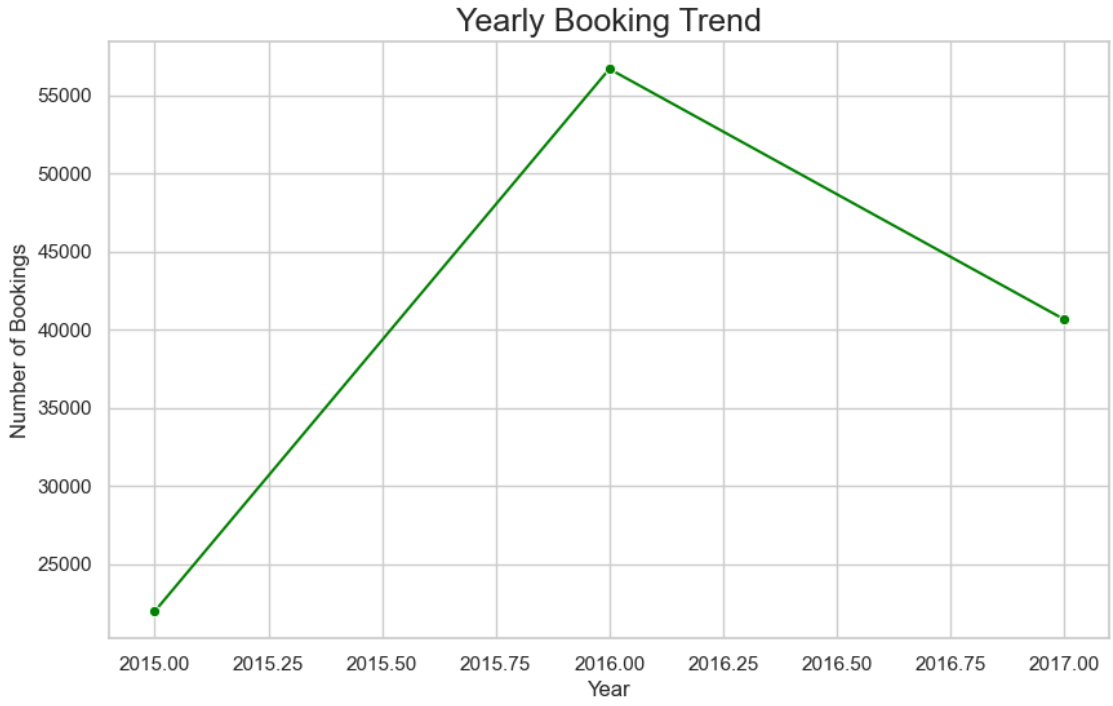


[14]:
```
# Monthly trend analysis
df['arrival_date_month'] = pd.Categorical(df['arrival_date_month'],
    categories=['January', 'February', 'March', 'April', 'May', 'June', 'July',
                'August', 'September', 'October', 'November', 'December'],
    ordered=True)

plt.figure(figsize=(10, 6))
sns.lineplot(data=df.groupby('arrival_date_month').size(), marker='o',␣
 ↪color='blue')
plt.title('Monthly Booking Trend', fontsize=18)
plt.xlabel('Month')
plt.ylabel('Number of Bookings')
plt.xticks(rotation=45)
plt.show()

# Yearly trend analysis
plt.figure(figsize=(10, 6))
sns.lineplot(data=df.groupby('arrival_date_year').size(), marker='o',␣
 ↪color='green')
plt.title('Yearly Booking Trend', fontsize=18)
```

```
plt.xlabel('Year')
plt.ylabel('Number of Bookings')
plt.show()
```

```
C:\Users\divaa\AppData\Local\Temp\ipykernel_1444\3650855569.py:8: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  sns.lineplot(data=df.groupby('arrival_date_month').size(), marker='o',
color='blue')
```



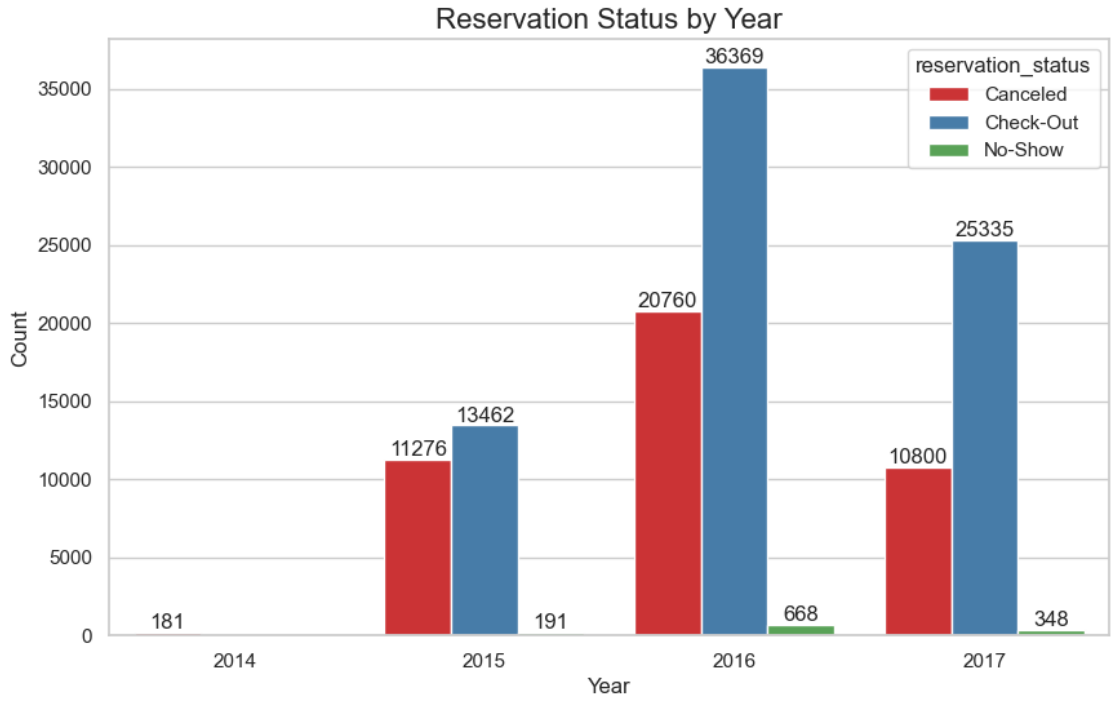


[17]:
```
# Grouping the data by year and reservation status, then counting the␣
 ↪occurrences
status_counts = df.groupby(['status_year', 'reservation_status']).size().
 ↪reset_index(name='counts')

# Display the counts
print(status_counts)
```

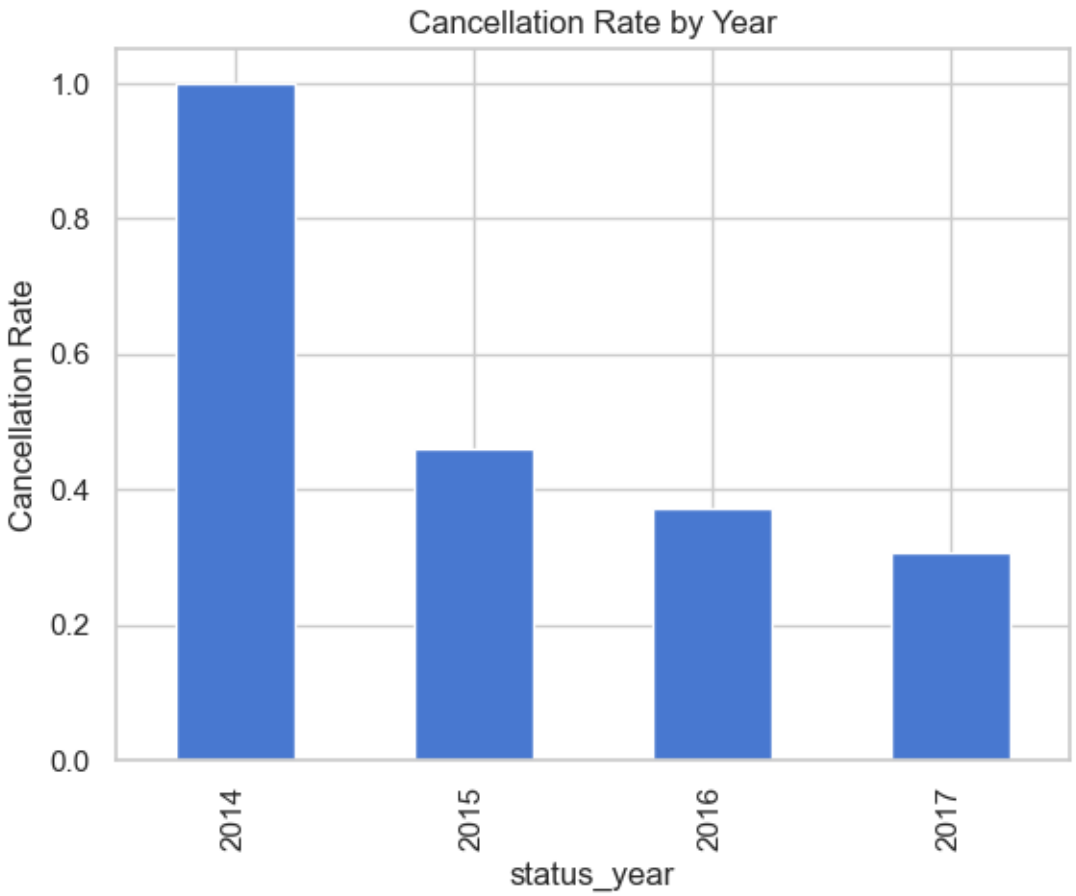
	status_year	reservation_status	counts
0	2014	Canceled	181
1	2015	Canceled	11276
2	2015	Check-Out	13462
3	2015	No-Show	191
4	2016	Canceled	20760
5	2016	Check-Out	36369
6	2016	No-Show	668
7	2017	Canceled	10800
8	2017	Check-Out	25335
9	2017	No-Show	348

[18]:
```
# Analyzing reservation status over time with numbers on bars
plt.figure(figsize=(10, 6))
ax = sns.countplot(data=df, x='status_year', hue='reservation_status',␣
 ↪palette='Set1')
```

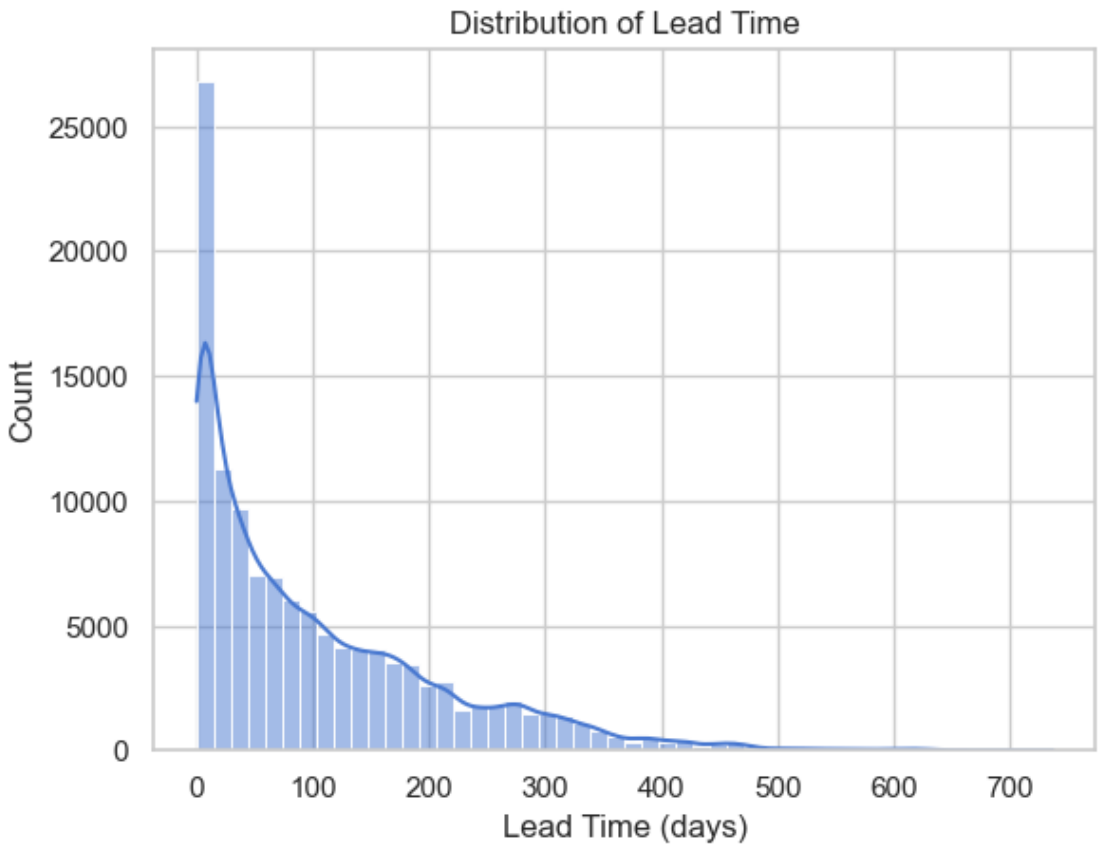
```
# Add counts on top of each bar
for container in ax.containers:
    ax.bar_label(container)

plt.title('Reservation Status by Year', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Count')
plt.show()
```



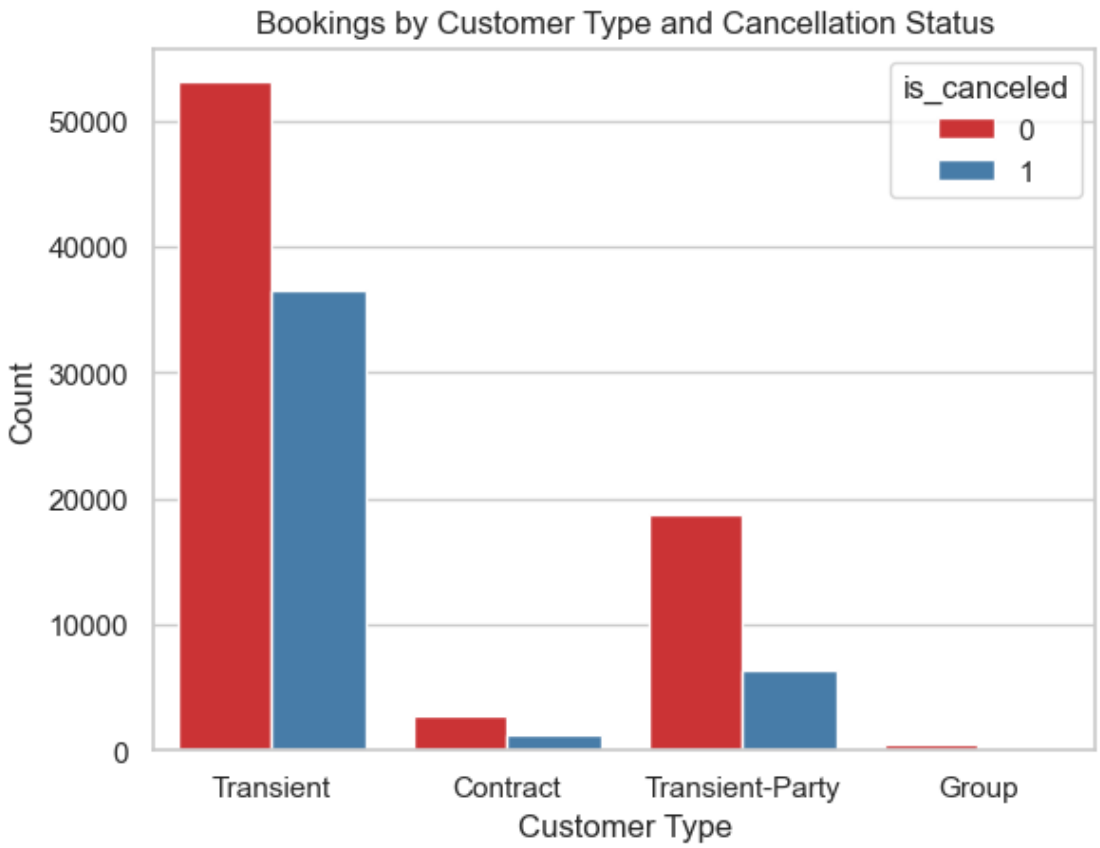
[24]:
```
# Cancellation rate by year
df.groupby('status_year')['is_canceled'].mean().plot(kind='bar')
plt.title('Cancellation Rate by Year')
plt.ylabel('Cancellation Rate')
plt.show()
```



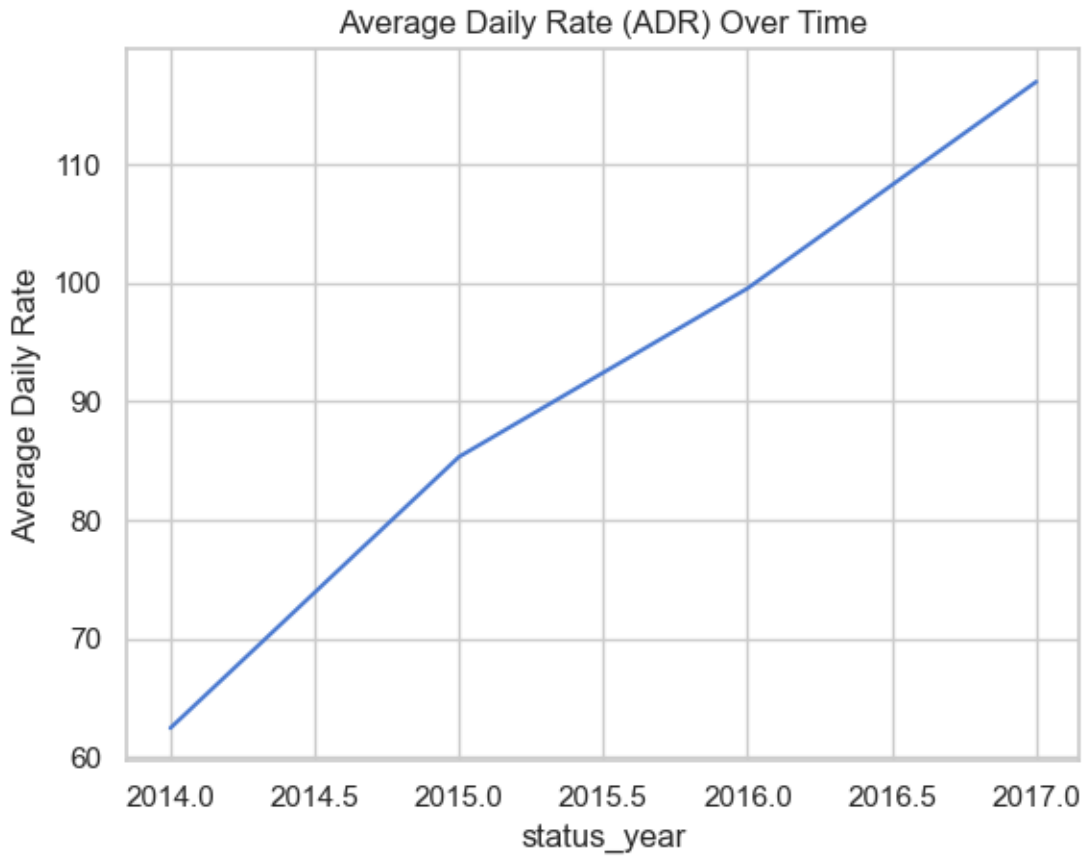
[26]:
```
# Lead time distribution
sns.histplot(df['lead_time'], bins=50, kde=True)
plt.title('Distribution of Lead Time')
plt.xlabel('Lead Time (days)')
plt.show()
```



[29]:
```
# Bookings by customer type
sns.countplot(data=df, x='customer_type', hue='is_canceled', palette='Set1')
plt.title('Bookings by Customer Type and Cancellation Status')
plt.xlabel('Customer Type')
plt.ylabel('Count')
plt.show()
```

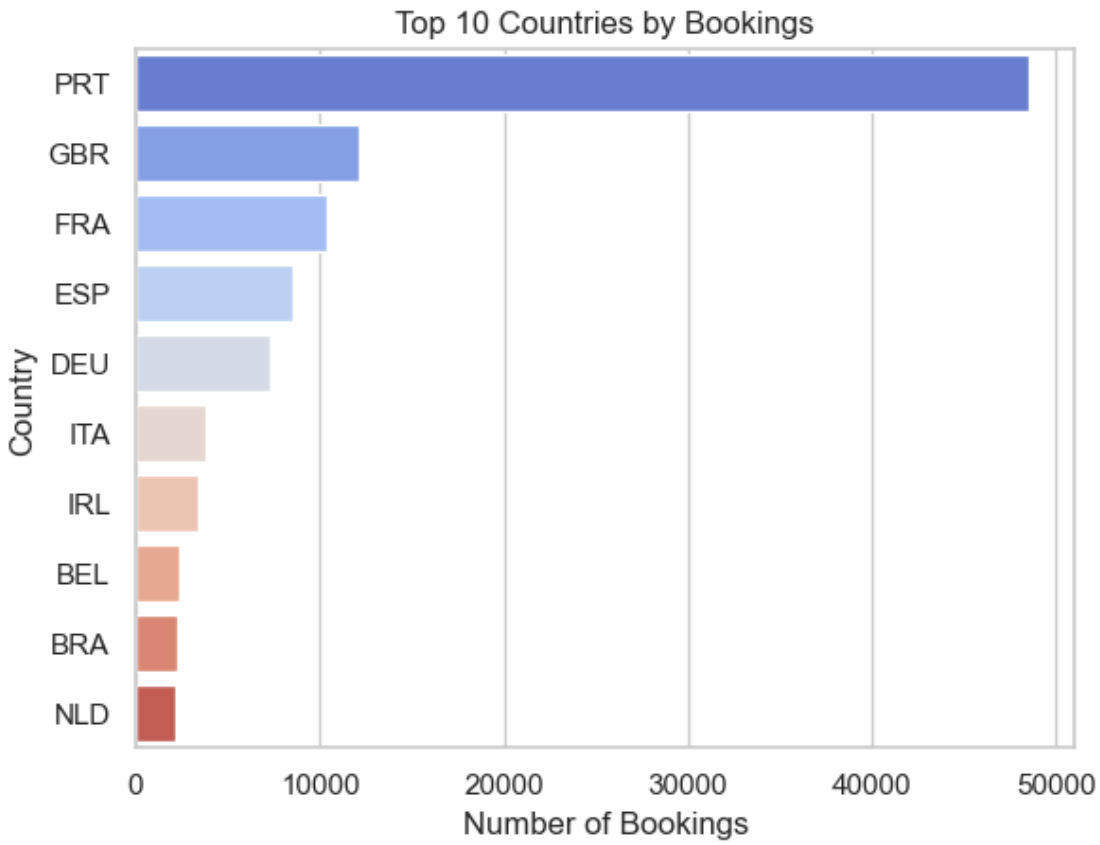


[30]:
```
# ADR over time
df.groupby('status_year')['adr'].mean().plot(kind='line')
plt.title('Average Daily Rate (ADR) Over Time')
plt.ylabel('Average Daily Rate')
plt.show()
```



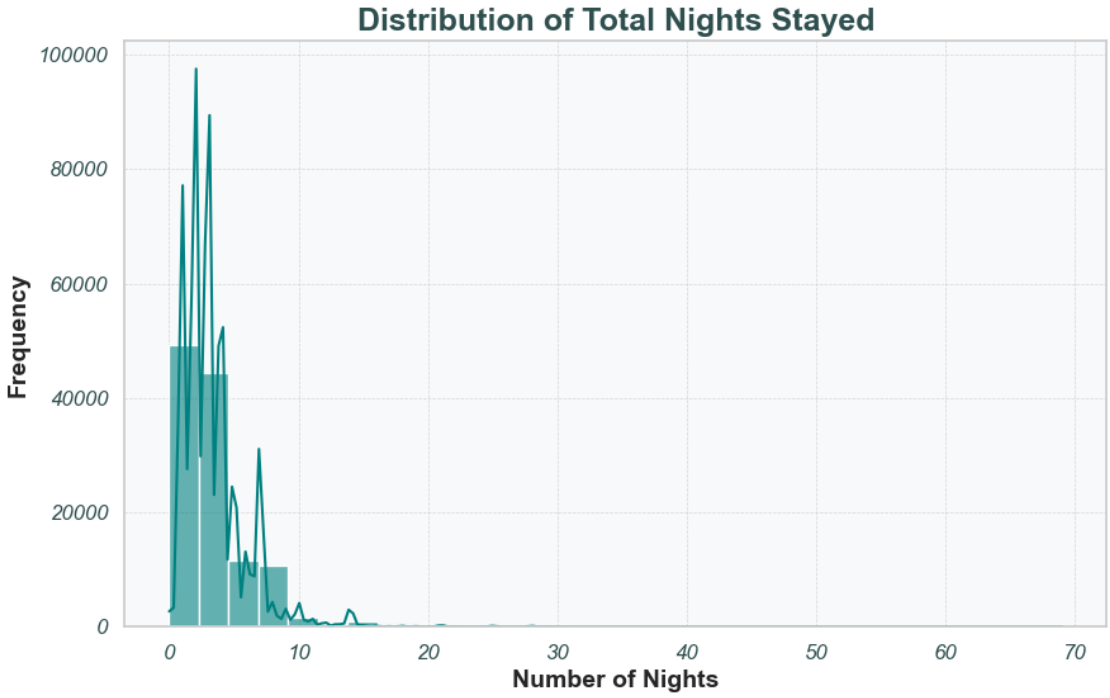
[32]:
```
# Top 10 countries by booking count
top_countries = df['country'].value_counts().head(10)
sns.barplot(y=top_countries.index, x=top_countries.values, palette='coolwarm')
plt.title('Top 10 Countries by Bookings')
plt.xlabel('Number of Bookings')
plt.ylabel('Country')
plt.show()
```

```
C:\Users\divaa\AppData\Local\Temp\ipykernel_1444\2782046020.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=top_countries.index, x=top_countries.values, palette='coolwarm')
```



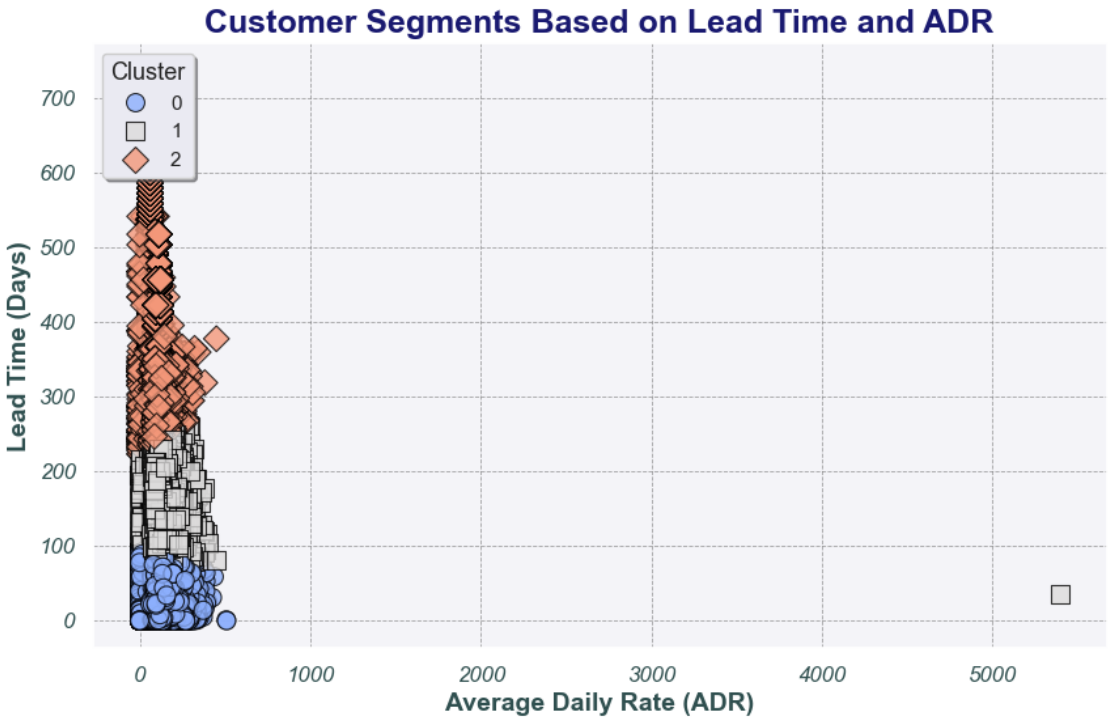
[36]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Set a custom style
sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))

# Set a color palette
palette = sns.color_palette("coolwarm", as_cmap=True)

# Plot the histogram with KDE and custom style
sns.histplot(df['total_nights'], bins=30, kde=True, color='teal', alpha=0.6)

# Add gridlines with custom transparency
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha=0.75)

# Customize the title, labels, and font sizes
plt.title('Distribution of Total Nights Stayed', fontsize=18,␣
 ↪fontweight='bold', color='darkslategray')
plt.xlabel('Number of Nights', fontsize=14, fontweight='bold')
```

```
plt.ylabel('Frequency', fontsize=14, fontweight='bold')

# Use a unique font style for ticks
plt.xticks(fontsize=12, color='darkslategray', fontstyle='italic')
plt.yticks(fontsize=12, color='darkslategray', fontstyle='italic')

# Add a background color to the plot area
plt.gca().set_facecolor('#f8f9fa')

# Show the plot
plt.show()
```



[41]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Perform KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['cluster'] = kmeans.fit_predict(df[['lead_time', 'adr', 'total_nights']])

# Set up the plot with a unique style and figure size
sns.set_style("darkgrid")
plt.figure(figsize=(10, 6))
```

```
# Create a custom color palette with transparency for the clusters
palette = sns.color_palette("coolwarm", n_colors=3)

# Plot the scatter plot with distinct markers for each cluster
sns.scatterplot(data=df, x='adr', y='lead_time', hue='cluster', palette=palette,
                style='cluster', markers=["o", "s", "D"], s=100, alpha=0.8,␣
 ↪edgecolor='black')

# Add gridlines with custom transparency and color
plt.grid(True, linestyle='--', linewidth=0.6, alpha=0.7, color='gray')

# Customize title and labels with unique font styles and color
plt.title('Customer Segments Based on Lead Time and ADR', fontsize=18,␣
 ↪fontweight='bold', color='midnightblue')
plt.xlabel('Average Daily Rate (ADR)', fontsize=14, fontweight='bold',␣
 ↪color='darkslategray')
plt.ylabel('Lead Time (Days)', fontsize=14, fontweight='bold',␣
 ↪color='darkslategray')

# Customize ticks style
plt.xticks(fontsize=12, fontstyle='italic', color='darkslategray')
plt.yticks(fontsize=12, fontstyle='italic', color='darkslategray')

# Set a background color for the plot area
plt.gca().set_facecolor('#f4f4f8')

# Add a legend with a custom location and style
plt.legend(title='Cluster', title_fontsize='13', loc='upper left', fontsize=11,␣
 ↪frameon=True, shadow=True, fancybox=True)

# Display the plot
plt.show()
```



The segmentation indicates clear customer behaviors: Budget-conscious, last-minute bookers (Cluster 0), Moderate spenders with varied planning preferences (Cluster 1), High-spending, advance planners (Cluster 2).