MAJORPROJECT

Hotel Booking Insights and Cancellation Prediction Using Data Analytics

Aim:-

This project aims to leverage hotel booking data to optimize pricing strategies, improve customer targeting, and enhance guest satisfaction by analyzing booking trends, guest profiles, and special request patterns.

Importing Libraries:-

```
import pandas as pd numpy as
np import import seaborn as
sns import matplotlib.pyplot
as plt import os
sns.set(style="whitegrid") %matplotlib inline from
sklearn.model_selection import train_test_split from
sklearn.preprocessing import LabelEncoder from sklearn.ensemble
import RandomForestClassifier from sklearn.metrics import
classification_report, confusion_matrix, accuracy_score import
plotly.express as px import folium
```

Load Data Set:-

```
df = pd.read_csv(Hotel Bookings.csv')
```

.	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	...	deposit_type	agent	company	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	reservation_status	reservation_status_date
0	Resort Hotel	0	342	2015	July	27	1	0	0	2	...	No Deposit	NaN	NaN	0	Transient	0.00	0	0	CheckOut	2015-07-01
1	Resort Hotel	0	737	2015	July	27	1	0	0	2	...	No Deposit	NaN	NaN	0	Transient	0.00	0	0	CheckOut	2015-07-01
2	Resort Hotel	0	7	2015	July	27	1	0	1	1	...	No Deposit	NaN	NaN	0	Transient	75.00	0	0	CheckOut	2015-07-02
3	Resort Hotel	0	13	2015	July	27	1	0	1	1	...	No Deposit	304.0	NaN	0	Transient	75.00	0	0	CheckOut	2015-07-02
4	Resort Hotel	0	14	2015	July	27	1	0	2	2	...	No Deposit	240.0	NaN	0	Transient	98.00	0	1	CheckOut	2015-07-03

...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
119385	City Hotel	0	23	2017	August	35	30	2	5	2	...	No Deposit	394.0	NaN	0	Transient	96.14	0	0	CheckOut	2017-09-06
119386	City Hotel	0	102	2017	August	35	31	2	5	3	...	No Deposit	9.0	NaN	0	Transient	225.43	0	2	CheckOut	2017-09-07
119387	City Hotel	0	34	2017	August	35	31	2	5	2	...	No Deposit	9.0	NaN	0	Transient	157.71	0	4	CheckOut	2017-09-07
119388	City Hotel	0	109	2017	August	35	31	2	5	2	...	No Deposit	89.0	NaN	0	Transient	104.40	0	0	CheckOut	2017-09-07
119389	City Hotel	0	205	2017	August	35	29	2	7	2	...	No Deposit	9.0	NaN	0	Transient	151.20	0	2	CheckOut	2017-09-07

119390 rows × 32 columns

A Quick Summary of a Data Frame:-

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 119390 entries, 0 to 119389

Data columns (total 32 columns):

Column Non-Null Count Dtype

--- ------ -------------- -----

0 hotel 119390 non-null object

1 is_canceled 119390 non-null int64

2 lead_time 119390 non-null int64

3 arrival_date_year 119390 non-null int64

4 arrival_date_month 119390 non-null object

5 arrival_date_week_number 119390 non-null int64

6 arrival_date_day_of_month 119390 non-null int64

7 stays_in_weekend_nights 119390 non-null int64

8 stays_in_week_nights 119390 non-null int64

9 adults 119390 non-null int64

10 children 119386 non-null float64

11 babies 119390 non-null int64

12 meal 119390 non-null object

13 country 118902 non-null object

14 market_segment 119390 non-null object

15 distribution_channel 119390 non-null object 16 is_repeated_guest 119390 non-null int64

17 previous_cancellations 119390 non-null int64

18 previous_bookings_not_canceled 119390 non-null int64

19 reserved_room_type 119390 non-null object

20 assigned_room_type 119390 non-null object
21 booking_changes 119390 non-null int64
22 deposit_type 119390 non-null object

23 agent 103050 non-null float64

24 company 6797 non-null float64

25 days_in_waiting_list 119390 non-null int64

26 customer_type 119390 non-null object
27 adr 119390 non-null float64

28 required_car_parking_spaces 119390 non-null int64

29 total_of_special_requests 119390 non-null int64

30 reservation_status 119390 non-null object

31 reservation_status_date 119390 non-null object dtypes: float64(4), int64(16), object(12) memory usage: 29.1+ MB

df.shape (119390,
32)

df.index
RangeIndex(start=0, stop=119390, step=1)
df.dtypes
hotel object
is_canceled int64 lead_time int64
arrival_date_year int64 arrival_date_month
object arrival_date_week_number int64
arrival_date_day_of_month int64
stays_in_weekend_nights int64
stays_in_week_nights int64
adults int64

children float64 babies
int64 meal object country
object
market_segment object
distribution_channel object
is_repeated_guest int64
previous_cancellations int64
previous_bookings_not_canceled
int64 reserved_room_type object
assigned_room_type object
booking_changes int64 deposit_type
object agent float64 company float64
days_in_waiting_list int64 customer_type
object
adr float64
required_car_parking_spaces int64
total_of_special_requests int64
reservation_status object
reservation_status_date object dtype:
object

df.isnull()

	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	...	deposit_type	agent	company	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	reservation_status	reservation_status_date
0	False		False	False	False	False	False	False	False	False	False	...	False	True	True	False	False	False	False	False	False	False
1	False		False	False	False	False	False	False	False	False	False	...	False	True	True	False	False	False	False	False	False	False
2	False		False	False	False	False	False	False	False	False	False	...	False	True	True	False	False	False	False	False	False	False
	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	...	deposit_type	agent	company	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	reservation_status	reservation_status_date
3	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False
4	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False

...	...		...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
119385	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False
119386	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False
119387	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False
119388	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False
119389	False		False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False

119390 rows × 32 columns

df.isnull().sum()

hotel 0

is_canceled 0 lead_time 0

arrival_date_year 0

arrival_date_month 0

arrival_date_week_number 0

arrival_date_day_of_month 0

stays_in_weekend_nights 0

stays_in_week_nights 0

adults 0

children 4

babies 0

meal 0

country 488

market_segment 0

distribution_channel 0

is_repeated_guest 0

previous_cancellations 0

previous_bookings_not_canceled 0

reserved_room_type 0 assigned_room_type 0 booking_changes 0

deposit_type 0 agent 16340 company 112593 days_in_waiting_list 0

customer_type 0

adr 0

required_car_parking_spaces 0

total_of_special_requests 0 reservation_status 0 reservation_status_date 0

dtype: int64

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

129425

df.describe()

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	is_repeated_guest	previous_cancellations	previous_bookings_not_canceled	booking_changes	agent	company	days_in_waiting_list	adr	required_car_parking_spaces	total_of_special_requests
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119386.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	103050.000000	6797.000000	119390.000000	119390.000000	119390.000000	119390.000000
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.7599	2.500302	1.856403	0.103890	0.007949	0.031912	0.087118	0.137097	0.221124	86.693382	189.266735	2.321149	101.831122	0.062518	0.571363
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.998613	1.908286	0.579261	0.398561	0.097436	0.175767	0.844336	1.497437	0.652306	110.774548	131.655015	17.594721	50.535790	0.245291	0.792798
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	6.000000	0.000000	-6.380000	0.000000	0.000000
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	9.000000	62.000000	0.000000	69.290000	0.000000	0.000000
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.000000	2.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	14.000000	179.000000	0.000000	94.575000	0.000000	0.000000
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.000000	3.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	229.000000	270.000000	0.000000	126.000000	0.000000	1.000000
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.000000	50.000000	55.000000	10.000000	10.000000	1.000000	26.000000	72.000000	21.000000	535.000000	543.000000	391.000000	5400.000000	8.000000	5.000000

Clean and Preprocess Data

Data cleaning, also known as data cleansing or data wrangling, is a crucial step in the data analytics process. It involves identifying, correcting, and formatting raw data to ensure its accuracy, consistency, and completeness before analysis.

Here's why data cleaning is essential: Garbage in, garbage out: Unreliable or inaccurate data leads to misleading and unreliable results. Cleaning ensures the foundation of your analysis is solid.

Improves analysis efficiency :

Clean data allows for smoother and faster analysis, saving you time and effort. #Enables better decision-making: Accurate insights derived from clean data empower you to make informed and effective decisions.

What does data cleaning involve? Data cleaning encompasses various tasks, depending on the specific dataset and its quality. Here are some common steps:

Identifying and removing errors: This includes finding and correcting typos, inconsistencies in formatting, and outliers that deviate significantly from the norm.

Handling missing values:

Missing data points can be dealt with by imputation (filling in missing values), deletion, or other techniques depending on the context.

Formatting inconsistencies:

Ensuring consistent formatting across data points, such as date formats, units of measurement, and capitalization, is crucial. Detecting and removing duplicates: Duplicate entries can skew analysis, so identifying and removing them is essential. Standardizing data: Transforming data into a consistent format, like scaling numerical values or converting categorical data into numerical codes, facilitates analysis.

Improved data quality:

Cleaning leads to more reliable and trustworthy data, enhancing the credibility of your analysis.

Enhanced analysis accuracy:

Clean data ensures your analysis reflects the true underlying patterns and relationships within the data.

Efficient data manipulation:

Clean data allows for smoother and faster manipulation and transformation during analysis.

Better decision-making:

Ultimately, clean data empowers you to make informed and effective decisions based on accurate insights. Tools and techniques for data cleaning:

NoProgramming languages:

Python with libraries like Pandas and NumPy is popular for data cleaning tasks.

Spreadsheets:

While suitable for smaller datasets, tools like Microsoft Excel can be used for basic cleaning tasks.

Data cleaning software:

Specialized software offers advanced features and automation for complex cleaning tasks.

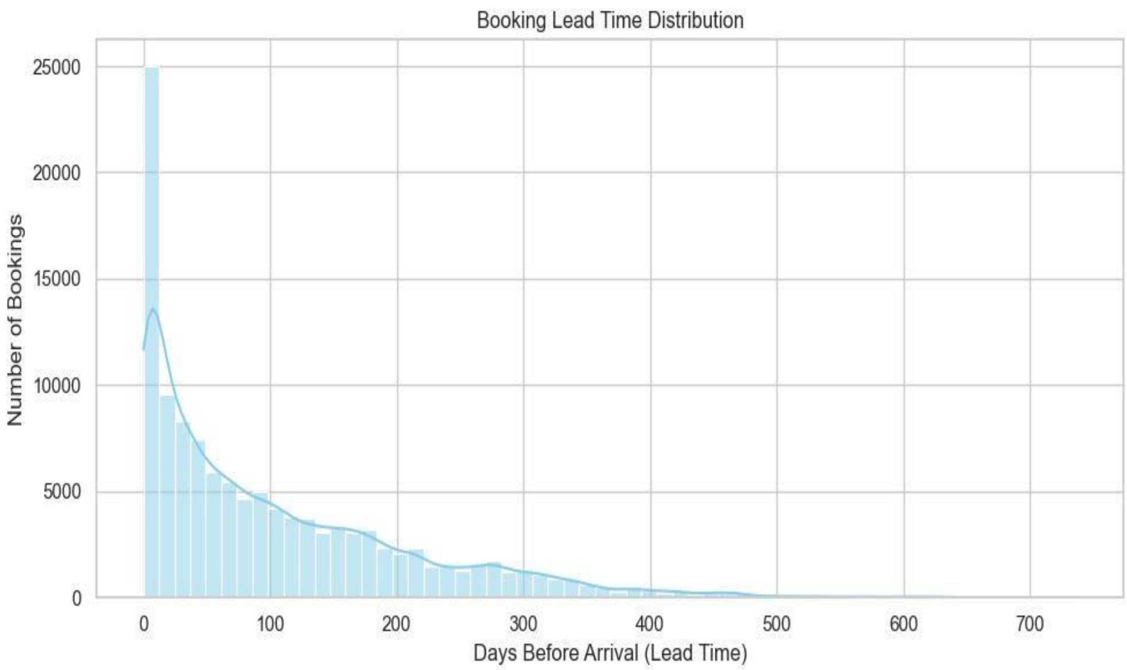
```
# Fill missing values df['children'].fillna(0,
inplace=True) df['country'].fillna('Unknown',
inplace=True) df['agent'].fillna(0,
inplace=True) df['company'].fillna(0,
inplace=True)

# Combine date columns
df['arrival_date'] = pd.to_datetime(
```

df['arrival_date_year'].astype(str) + '-' + df['arrival_date_month'] + '-' + df['arrival_date_day_of_month'].astype(str), format='%Y-%B-%d'
)

Add total nights and total guests columns df['total_nights'] = df['stays_in_weekend_nights'] + df['stays_in_week_nights'] df['total_guests'] = df['adults'] + df['children'] + df['babies']

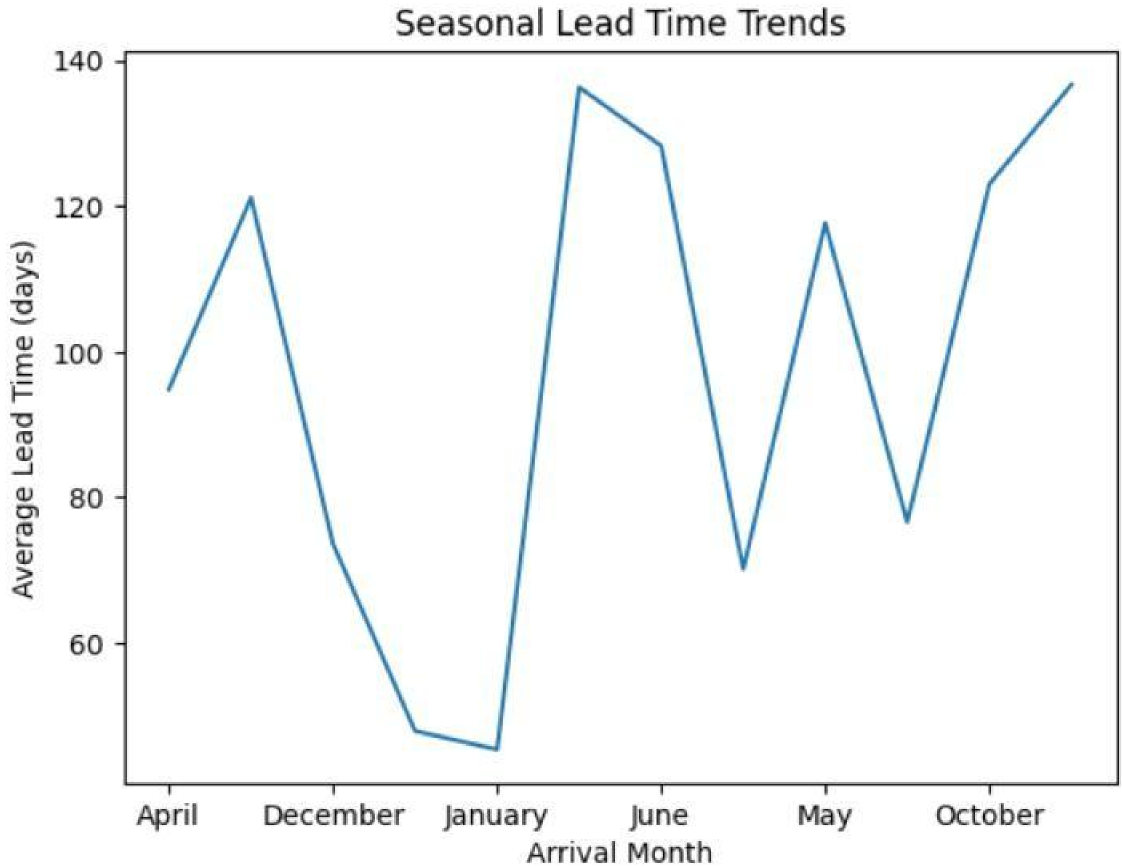
Booking Lead Time Analysis

1.# Lead time distribution plt.figure(figsize=(10, 5)) sns.histplot(df['lead_time'], bins=60, kde=True, color='skyblue') plt.title('Booking Lead Time Distribution') plt.xlabel('Days Before Arrival (Lead Time)') plt.ylabel('Number of Bookings') plt.tight_layout() plt.show()



2.# Seasonal Lead Time Trends lead_time_by_month = df.groupby('arrival_date_month')['lead_time'].mean() lead_time_by_month.plot(kind='line')

```
plt.title('Seasonal Lead Time Trends')
plt.xlabel('Arrival Month')
plt.ylabel('Average Lead Time (days)')
plt.show()
```



Price Vary Per Night Over The Year

```
data_resort = df[(df['hotel'] == 'Resort Hotel') & (df['is_canceled'] ==
0)] data_city = df[(df['hotel'] == 'City Hotel') & (df['is_canceled'] ==
0)] data _resort
```

	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
0	Resort Hotel	0	342	2015	July	27	1	0	0	2
1	Resort Hotel	0	737	2015	July	27	1	0	0	2
2	Resort Hotel	0	7	2015	July	27	1	0	1	1

	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
3	Resort Hotel	0	13	2015	July	27	1	0	1	1
4	Resort Hotel	0	14	2015	July	27	1	0	2	2
...	...	...	...	...	...	...	...	...	...	...
40055	Resort Hotel	0	212	2017	August	35	31	2	8	2
40056	Resort Hotel	0	169	2017	August	35	30	2	9	2
40057	Resort Hotel	0	204	2017	August	35	29	4	10	2
40058	Resort Hotel	0	211	2017	August	35	31	4	10	2
40059	Resort Hotel	0	161	2017	August	35	31	4	10	2

.....

deposit_type	agent	company	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	reservation_status	reservation_status_date
No Deposit	NaN	NaN	0	Transient	0.00	0	0	Check-Out	2015-07-01
No Deposit	NaN	NaN	0	Transient	0.00	0	0	Check-Out	2015-07-01
No Deposit	NaN	NaN	0	Transient	75.00	0	0	Check-Out	2015-07-02
No Deposit	304.0	NaN	0	Transient	75.00	0	0	Check-Out	2015-07-02
No Deposit	240.0	NaN	0	Transient	98.00	0	1	Check-Out	2015-07-03
...	...	...	...	...	...	...	...	...	...
No Deposit	143.0	NaN	0	Transient	89.75	0	0	Check-Out	2017-09-10
No Deposit	250.0	NaN	0	Transient-Party	202.27	0	1	Check-Out	2017-09-10
No Deposit	250.0	NaN	0	Transient	153.57	0	3	Check-Out	2017-09-12

No Deposit	40.0	NaN	0	Contract	112.80	0	1	Check-Out	2017-09-14
deposit_type	**agent**	**company**	**days_in_waiting_list**	**customer_type**	**adr**	**required_car_parking_spaces**	**total_of_special_requests**	**reservation_status**	**reservation_status_date**
No Deposit	69.0	NaN	0	Transient	99.06	0	0	Check-Out	2017-09-14

28938 rows × 32 columns

```
resort_hotel =
data_resort.groupby(['arrival_date_month'])['adr'].mean().reset_index()
resort_hotel
```

	arrival_date_month	adr
0	April	75.867816
1	August	181.205892
2	December	68.322236
3	February	54.147478
4	January	48.708919
5	July	150.122528
6	June	107.921869
7	March	57.012487
8	May	76.657558
9	November	48.681640
10	October	61.727505
11	September	96.416860

```
city_hotel=data_city.groupby(['arrival_date_month'])['adr'].mean().reset_index() city_hotel
```

	arrival_date_month	adr
0	April	111.85682

	arrival_date_month	adr
	August	4
1	December	118.412083
2	February	87.856764
3	January	86.183025
4	July	82.160634
5		115.563810
	arrival_date_month	**adr**
6	June	117.702075
7	March	90.170722
8	May	120.445842
	November	86.500456
9	October	101.745956
10	September	112.598452

```
final_hotel = resort_hotel.merge(city_hotel, on ='arrival_date_month')
final_hotel
```

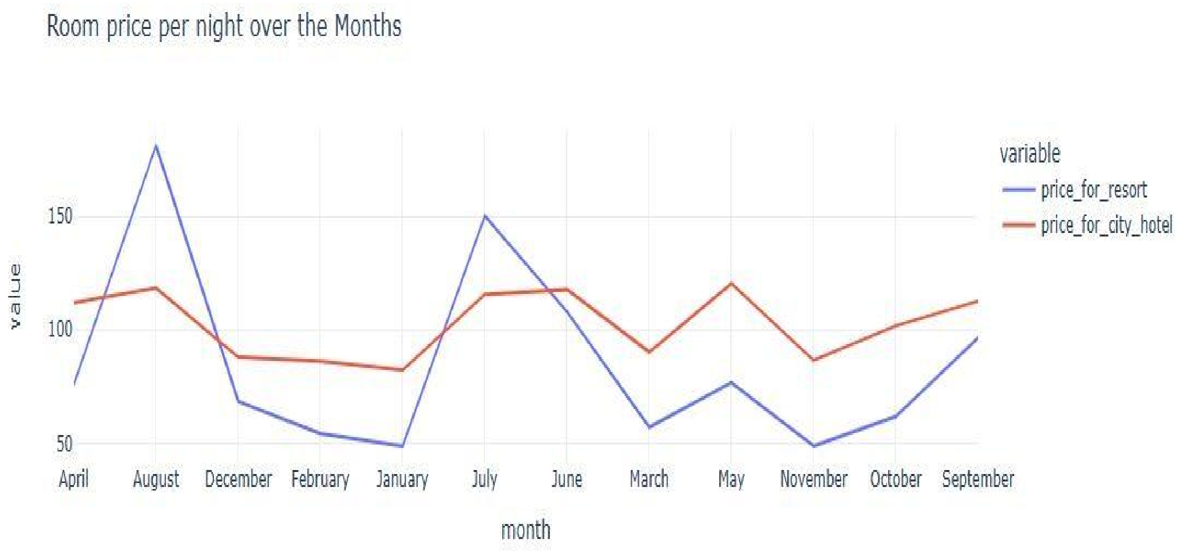
	arrival_date_month	adr_x	adr_y
0	April	75.867816	111.856824
1	August	181.205892	
2	December	118.412083	
3	February	68.322236	87.856764
4	January	54.147478	86.183025
5	July	48.708919	82.160634
6	June	150.122528	
7	March	115.563810	
8	May	107.921869	
9	November	117.702075	

1	October	57.012487	90.170722
0	September	76.657558	120.445842
1		48.681640	86.500456
1		61.727505	101.745956
		96.416860	112.598452

```
final_hotel.columns = ['month', 'price_for_resort','price_for_city_hotel']
final_hotel
```

	month	price_for_resort	price_for_city_hotel
0	April	75.867816	111.856824
1	August	181.205892	118.412083
2	December	68.322236	87.856764
3	February	54.147478	86.183025
4	January	48.708919	82.160634
5	July	150.122528	115.563810
6	June	107.921869	117.702075
7	March	57.012487	90.170722
8	May	76.657558	120.445842
9	November	48.681640	86.500456
10	October	61.727505	101.745956
11	September	96.416860	112.598452

```
plt.figure(figsize = (20,10)) px.line(final_hotel, x = 'month', y = ['price_for_resort','price_for_city_hotel'], title = 'Room price per night over the Months', template = 'plotly_white')
```



<Figure size 2000x1000 with 0 Axes>

final_hotel.columns = ['month', 'price_for_resort', 'price_for_city_hotel']

month_dict = {'January':1,'February':2,'March':3, 'April':4, 'May':5, 'June':6, 'July':7, 'August':8, 'September':9, 'October':10, 'November':11,'December':12}

final_hotel_sorted = final_hotel.sort_values('month', key = lambda x : x.apply (lambda x : month_dict[x]))

print('SORTED DATAFRAME BY MONTHS AS:') final_hotel_sorted

SORTED DATAFRAME BY MONTHS AS:

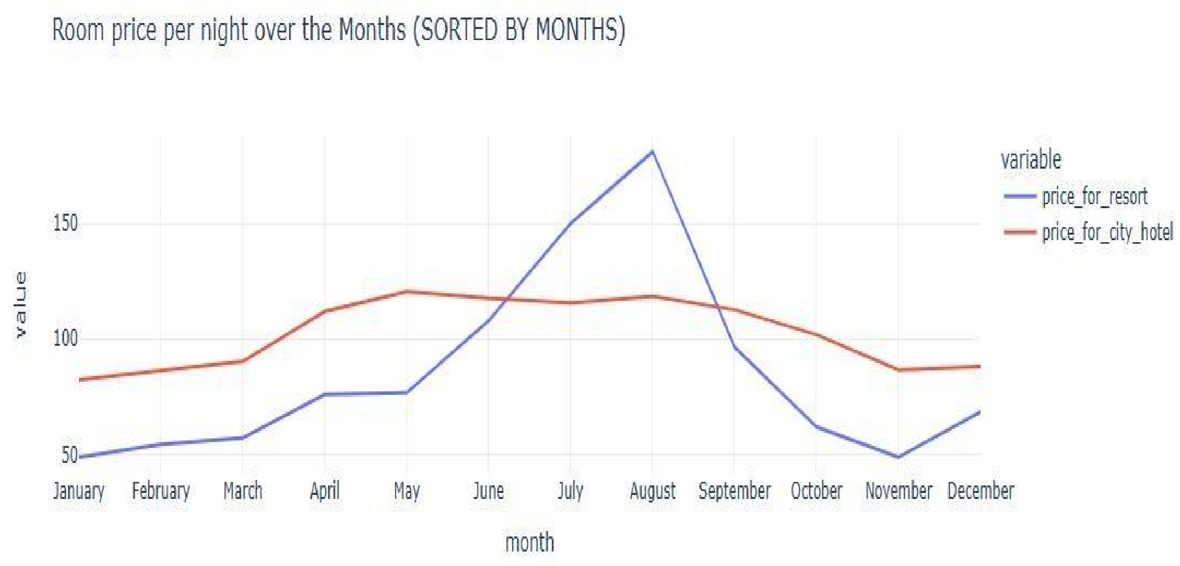
	month	price_for_resort	price_for_city_hotel
4	January	48.708919	82.160634
3	February	54.147478	86.183025
7	March	57.012487	90.170722
0	April	75.867816	111.856824
8	May	76.657558	120.445842
6	June	107.921869	117.702075

5	July	150.122528	115.563810
1	August	181.205892	118.412083
11	September	96.416860	112.598452
10	October	61.727505	101.745956
9	November	48.681640	86.500456
2	December	68.322236	87.856764

plt.figure(figsize = (20,10))

px.line(final_hotel_sorted, x = 'month', y =
['price_for_resort','price_for_city_hotel'],

title = 'Room price per night over the Months (SORTED BY MONTHS)',
template = 'plotly_white')

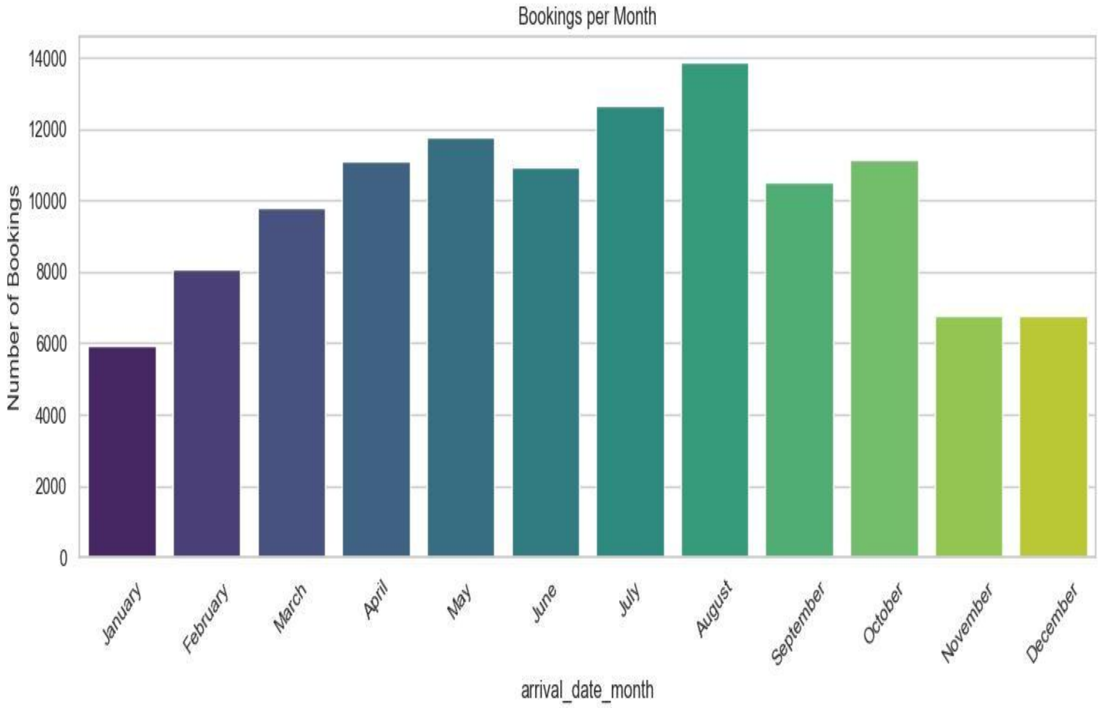


<Figure size 2000x1000 with 0 Axes>

2.# Monthly bookings (ordered)

monthly_order = ['January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December']

```
monthly_bookings =
df['arrival_date_month'].value_counts().reindex(monthly_order)

plt.figure(figsize=(12, 5)) sns.barplot(x=monthly_bookings.index,
y=monthly_bookings.values, palette='viridis') plt.title('Bookings per
Month') plt.ylabel('Number of Bookings') plt.xticks(rotation=45)
plt.tight_layout() plt.show()
```



Duration of people stay in the hotel

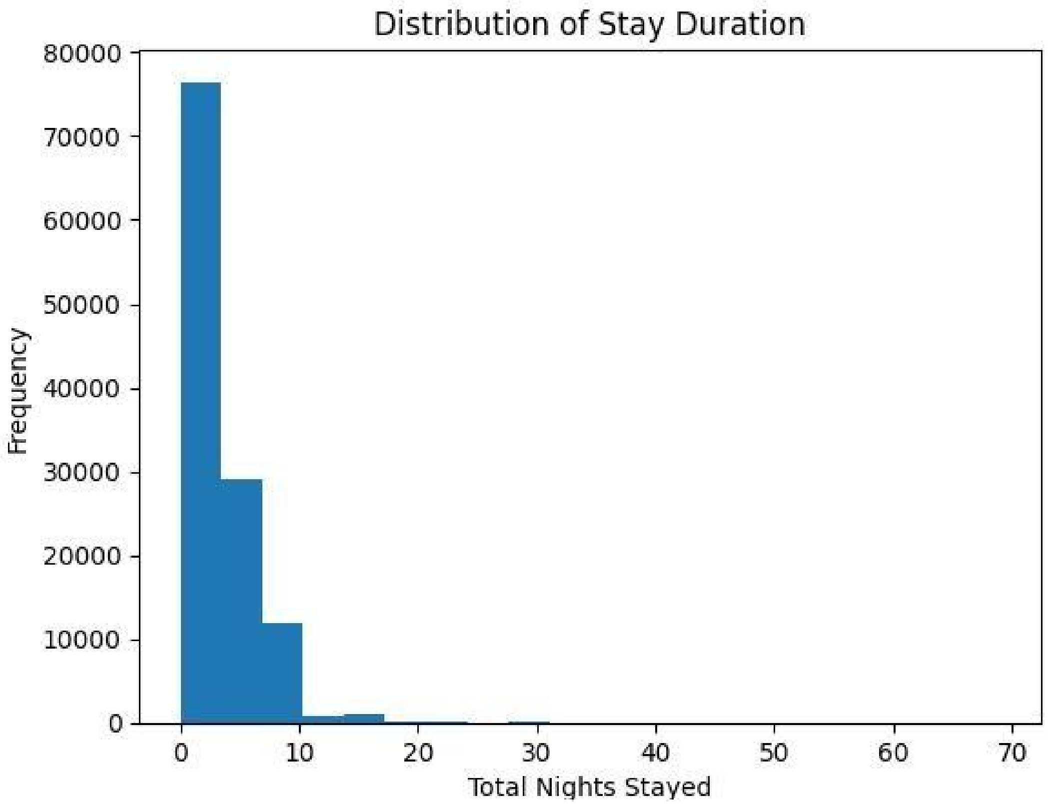
```
data['total_nights'] = data['stays_in_week_nights'] +
data['stays_in_weekend_nights'] print(data['total_nights'].describe())
```

```
count    119390.000000
mean          3.427900
std           2.557439
min           0.000000
25%           2.000000
50%           3.000000
75%           4.000000
max          69.000000
```

Name: total_nights, dtype: float64

#barplot: distribution of stay duration and frequency

plt.hist(data['total_nights'], bins=20) plt.xlabel('Total Nights Stayed') plt.ylabel('Frequency') plt.title('Distribution of Stay Duration') plt.show()



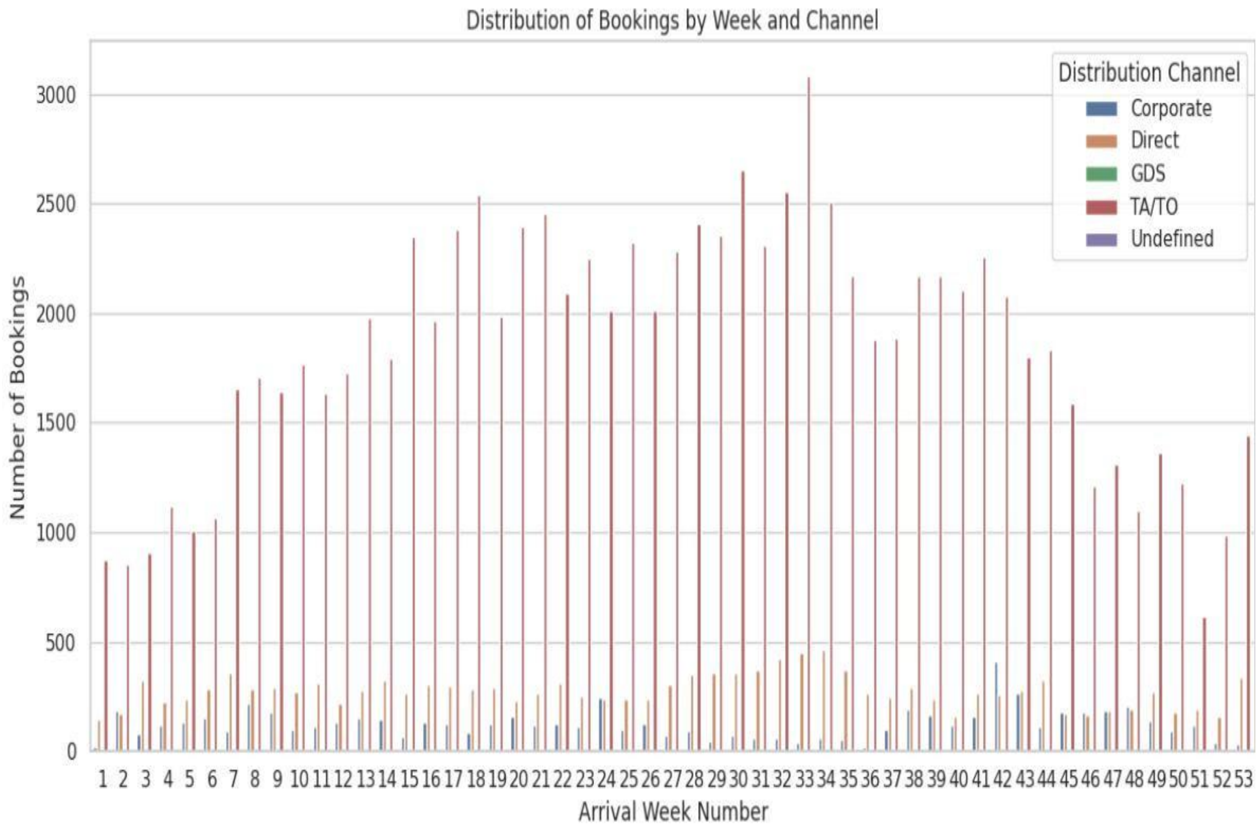
Distribution by Arrival Week and Distribution Channel

Assuming you have already generated the 'headings' list as in the previous code

Create a DataFrame from the headings df_headings =

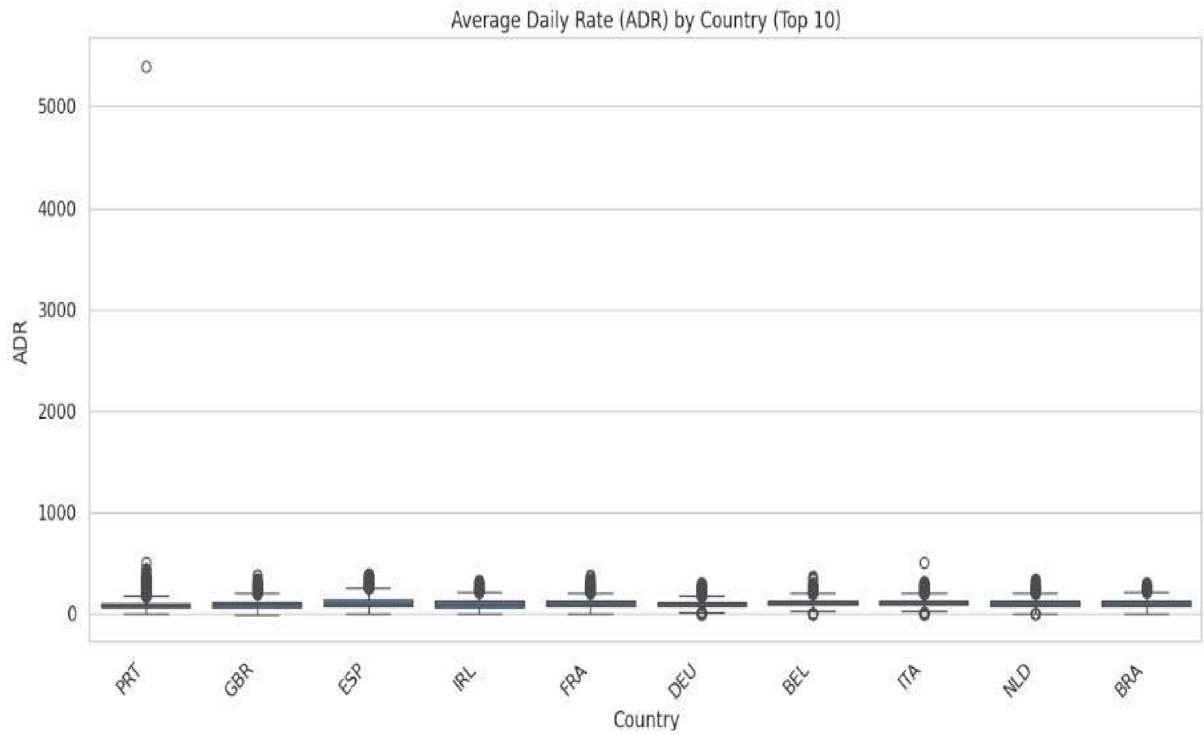
pd.DataFrame(headings, columns=['Heading'])

```
# Extract week number and distribution channel from the headings

df_headings['arrival_date_week_number'] =

df_headings['Heading'].str.extract(r'Week (\d+)').astype(int)

df_headings['distribution_channel'] = df_headings['Heading'].str.extract(r'- (.*)')

# Group by week number and distribution channel and count occurrences

df_grouped = df_headings.groupby(['arrival_date_week_number',
'distribution_channel'])['Heading'].count().reset_index(name='count')


# Create the bar chart using seaborn

plt.figure(figsize=(12, 6))

sns.barplot(x='arrival_date_week_number', y='count',

hue='distribution_channel', data=df_grouped)

plt.title('Distribution of Bookings by Week and Channel')

plt.xlabel('Arrival Week Number') plt.ylabel('Number of

Bookings') plt.legend(title='Distribution Channel')

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



Insights into ADR Variations: Top 10 Booking Countries

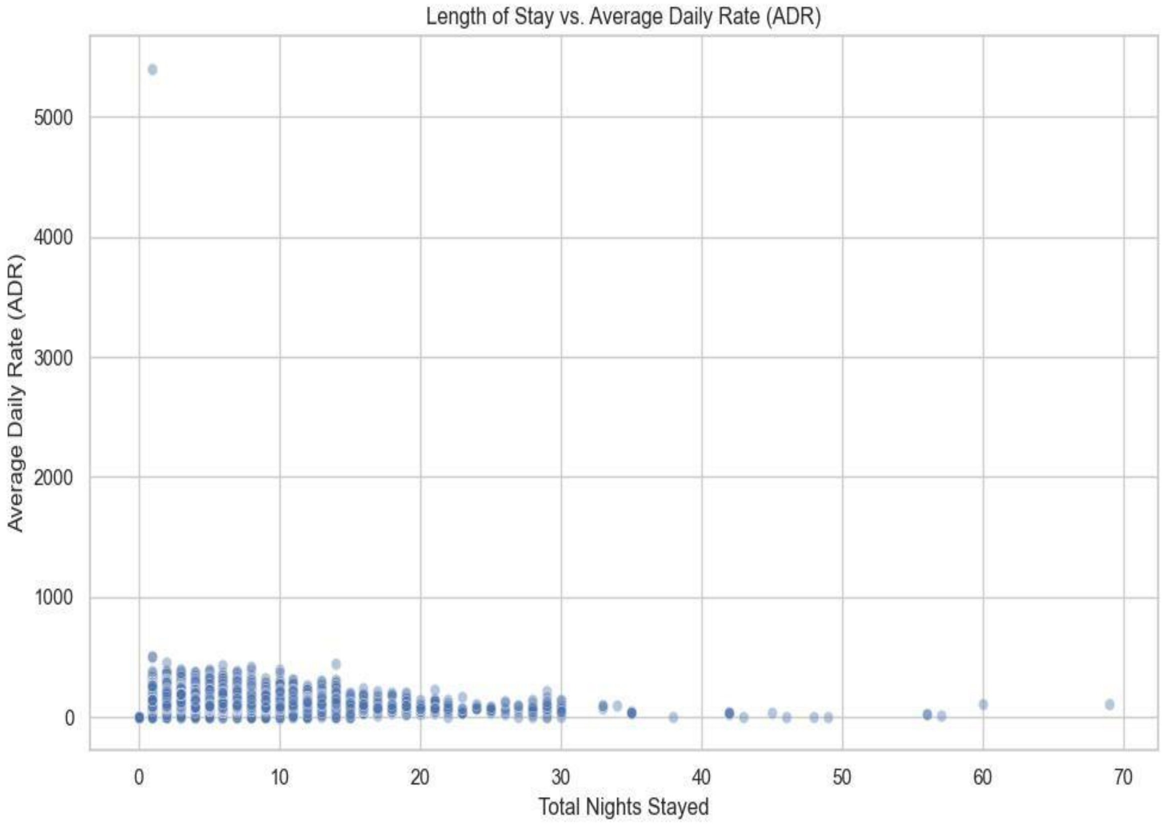
```
# Select top N countries by booking frequency top_n_countries =

df['country'].value_counts().nlargest(10).index filtered_df =

df[df['country'].isin(top_n_countries)]


plt.figure(figsize=(12, 6)) # Adjust figure size as needed

sns.boxplot(x='country', y='adr', data=filtered_df) plt.title('Average Daily

Rate (ADR) by Country (Top 10)') plt.xlabel('Country') plt.ylabel('ADR')

plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better readability

plt.tight_layout()

plt.show()
```



Length of Stay vs. ADR (Price) Analysis

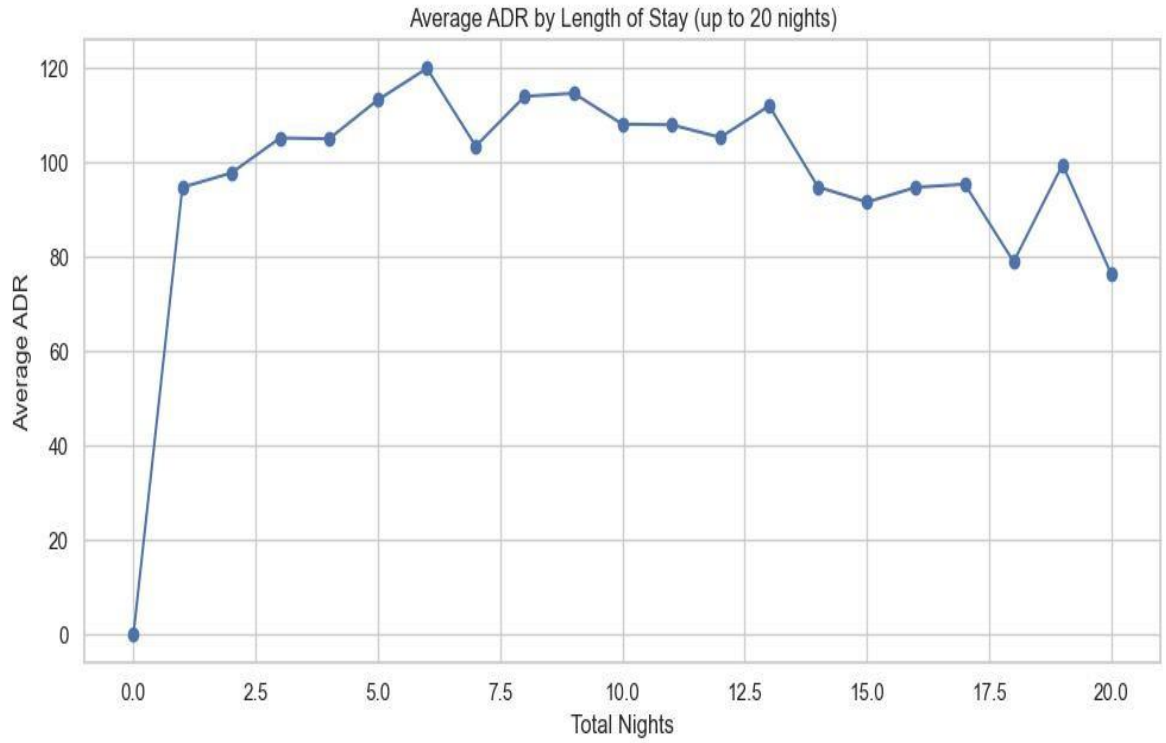
1. Scatter Plot - Total Nights vs ADR

plt.figure(figsize=(10, 6)) sns.scatterplot(data=df, x='total_nights', y='adr', alpha=0.4) plt.title('Length of Stay vs. Average Daily Rate (ADR)') plt.xlabel('Total Nights Stayed') plt.ylabel('Average Daily Rate (ADR)') plt.tight_layout() plt.show()



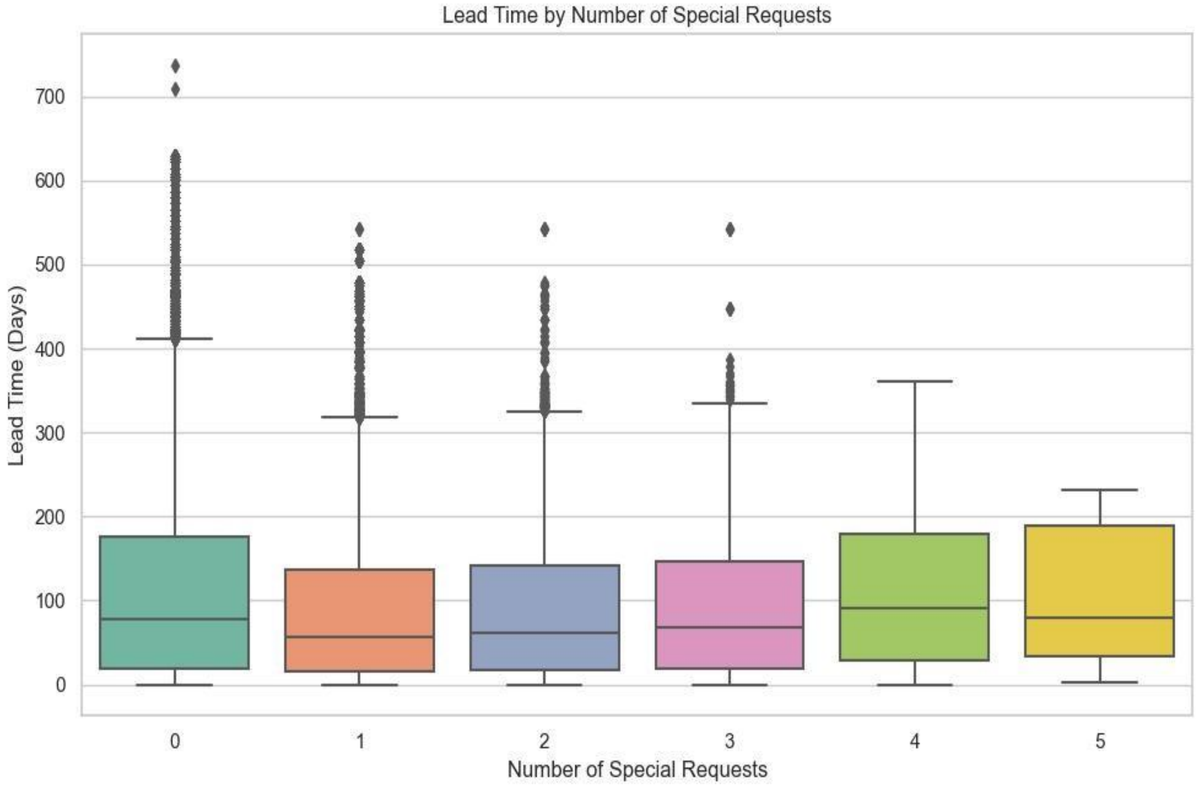
In [24]:

```
# 2. Average ADR by stay duration (for first 20 nights) stay_vs_adr =
df[df['total_nights'] <= 20].groupby('total_nights')['adr'].mean()

plt.figure(figsize=(10, 5)) stay_vs_adr.plot(marker='o')
plt.title('Average ADR by Length of Stay (up to 20 nights)')
plt.xlabel('Total Nights') plt.ylabel('Average ADR')
plt.grid(True) plt.tight_layout() plt.show()
```

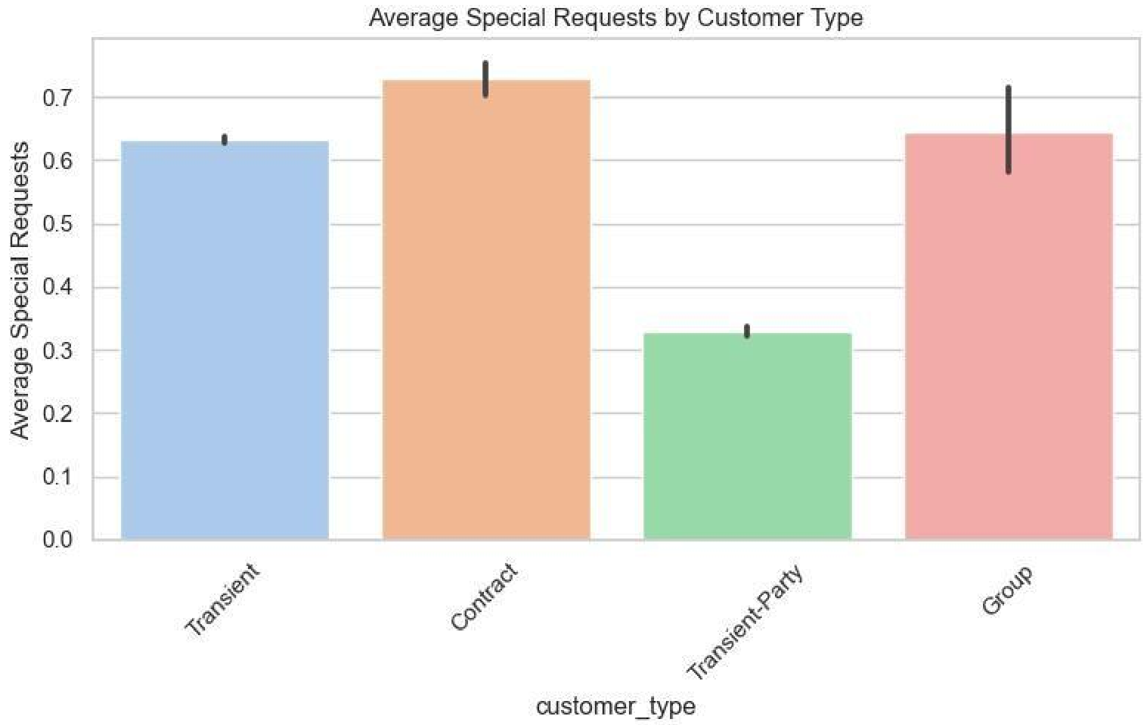


In [25]:

```
# Boxplot: Special Requests vs. Lead Time plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='total_of_special_requests', y='lead_time',
palette='Set2') plt.title('Lead Time by Number of Special
Requests') plt.xlabel('Number of Special Requests')
plt.ylabel('Lead Time (Days)') plt.tight_layout() plt.show()
```

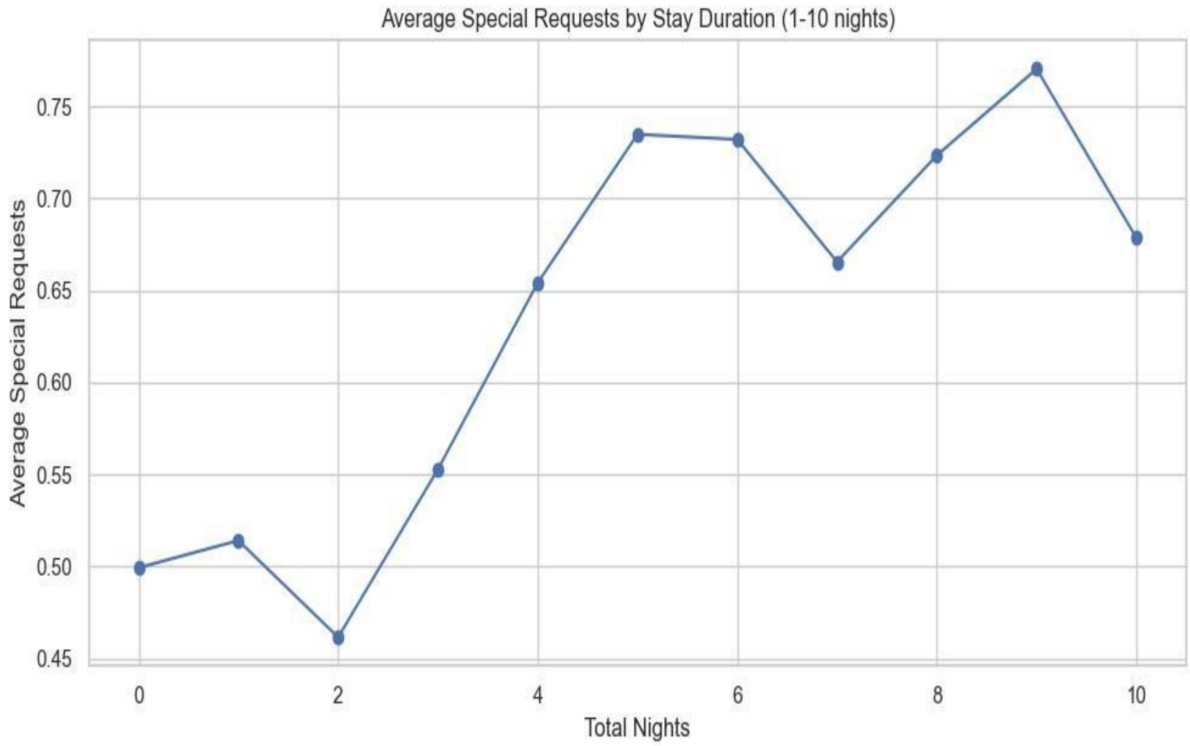



```
# Barplot: Avg Special Requests by Customer Type
if 'customer_type' in df.columns:
    plt.figure(figsize=(8, 5)) sns.barplot(data=df, x='customer_type',
    y='total_of_special_requests',
palette='pastel')
    plt.title('Average Special Requests by Customer Type')
    plt.ylabel('Average Special Requests')
    plt.xticks(rotation=45) plt.tight_layout() plt.show()
```



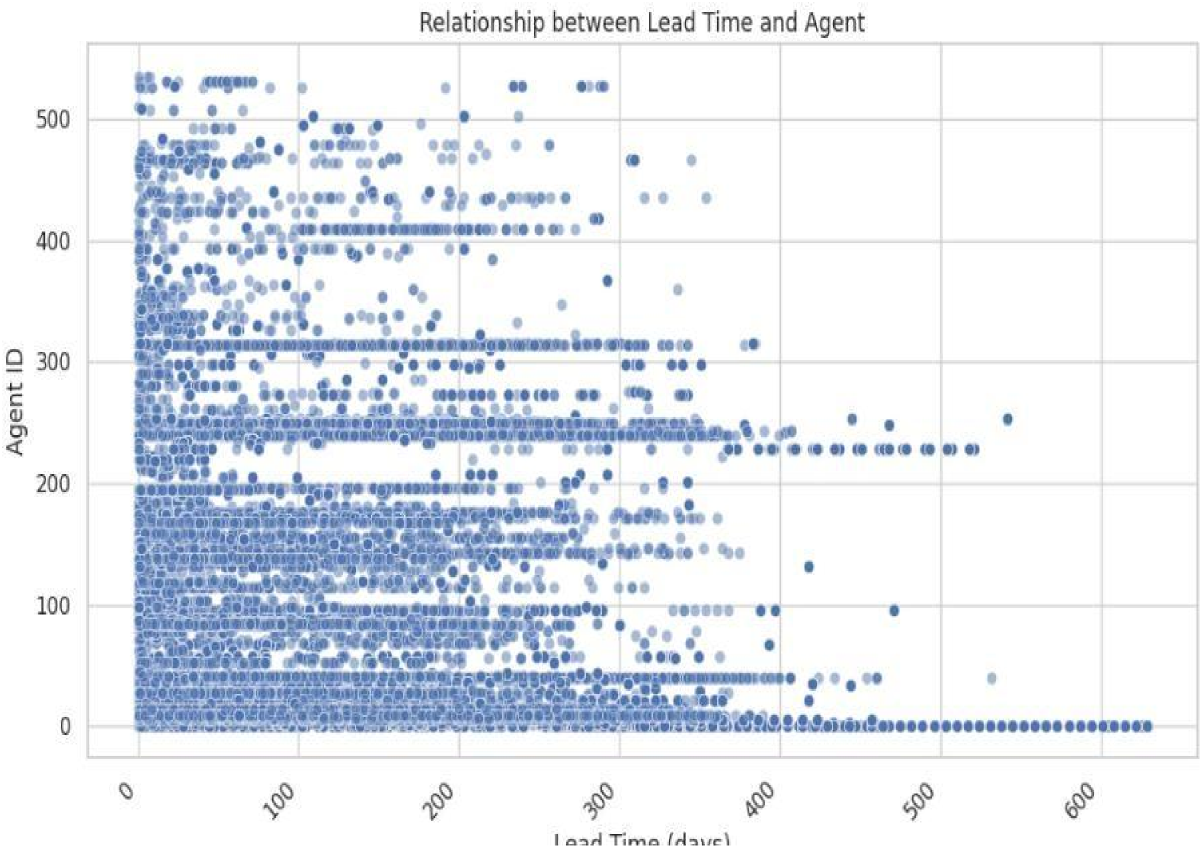
In [27]:

```
# Barplot: Avg Special Requests by Total Nights (limit to 1-10 nights)
requests_vs_nights = df[df['total_nights'] <=
10].groupby('total_nights')['total_of_special_requests'].mean()

plt.figure(figsize=(10, 5)) requests_vs_nights.plot(marker='o')
plt.title('Average Special Requests by Stay Duration (1-10 nights)')
plt.xlabel('Total Nights') plt.ylabel('Average Special Requests')
plt.grid(True) plt.tight_layout() plt.show()
```



Exploring the Relationship between Lead Time and Agent in Hotel Bookings

```
# Filter out rows with agent = NULL
df_filtered = df[df['agent'].notna()]

plt.figure(figsize=(10, 6))
sns.scatterplot(x='lead_time', y='agent', data=df_filtered, alpha=0.5) # Alpha
for transparency
plt.title('Relationship between Lead Time and Agent')
plt.xlabel('Lead Time (days)')
plt.ylabel('Agent ID')
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels if needed
plt.tight_layout() plt.show()
```



Demographic & Country Analysis

We'll explore:

Top countries by number of bookings

Distribution of market segments

Repeated guests vs new guests

!pip install plotly !pip

install folium import

plotly.express as px import

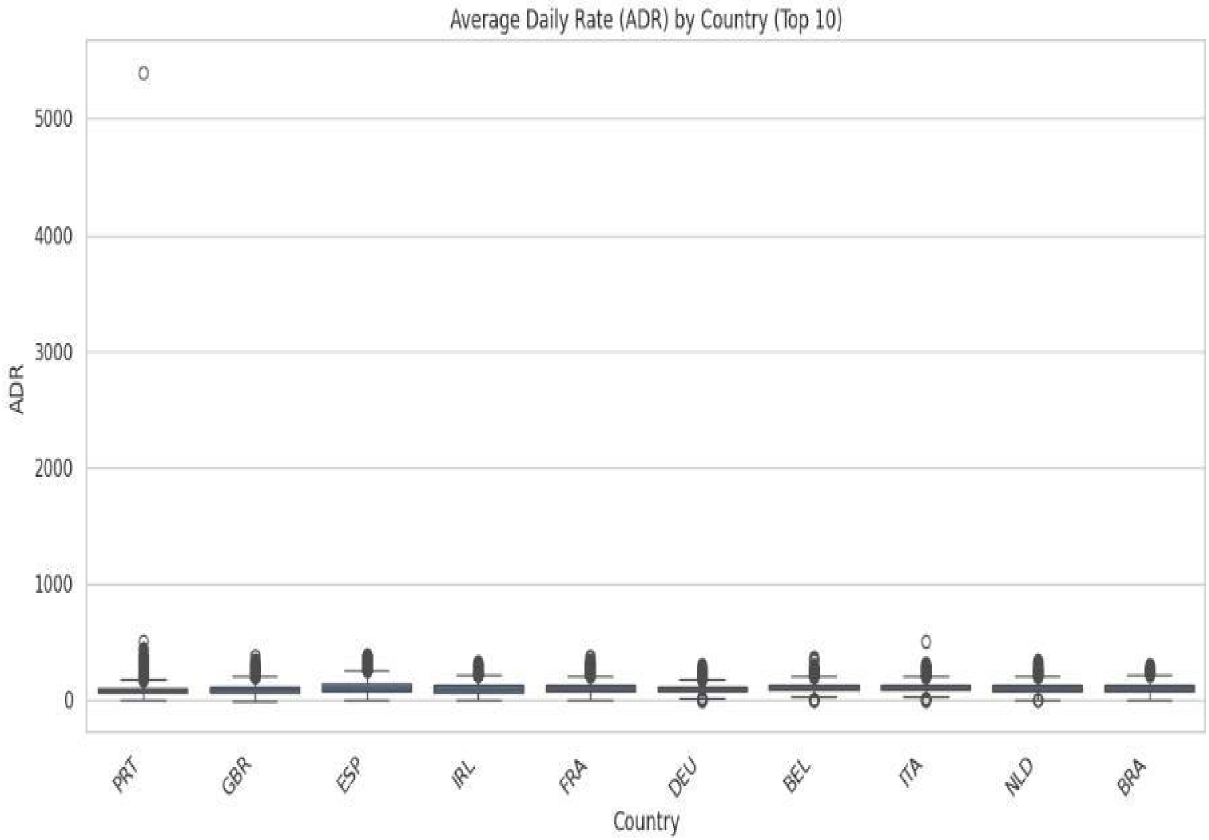
folium

Assuming you have a DataFrame called 'df' with hotel booking data

```
# Create a DataFrame summarizing guests by country country_wise_guests =
df.groupby('country')['hotel'].count().reset_index()
country_wise_guests.columns = ['country', 'No of guests']

# Create the choropleth map using plotly.express
guests_map = px.choropleth(
    country_wise_guests,
    locations='country',
    color='No of guests',
    hover_name='country',
    color_continuous_scale='Viridis', # Optional: Choose a color scale
    title='Distribution of Hotel Guests by Country'
)

# Display the map guests_map.show()

# Create a basemap using folium (optional)
basemap = folium.Map(location=[0, 0], zoom_start=2) # Centered on the world

# You can add markers or other layers to the folium basemap if needed
# ...

# Display the folium basemap (if you added layers)
Basemap
```

Insights into ADR Variations: Top 10 Booking Countries

```
# Select top N countries by booking frequency top_n_countries =

df['country'].value_counts().nlargest(10).index


filtered_df = df[df['country'].isin(top_n_countries)]


plt.figure(figsize=(12, 6)) # Adjust figure size as needed

sns.boxplot(x='country', y='adr', data=filtered_df) plt.title('Average Daily

Rate (ADR) by Country (Top 10)') plt.xlabel('Country') plt.ylabel('ADR')

plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better readability

plt.tight_layout()

plt.show()
```



Cancellation Analysis:

#1.Overall Cancellation Rate

cancellation_with_requests = df[df['total_of_special_requests'] > 0]['is_canceled'].mean()
cancellation_without_requests = df[df['total_of_special_requests'] == 0]['is_canceled'].mean()

print(f"Cancellation rate WITH special requests: {cancellation_with_requests*100:.2f}%")
print(f"Cancellation rate WITHOUT special requests: {cancellation_without_requests*100:.2f}%")

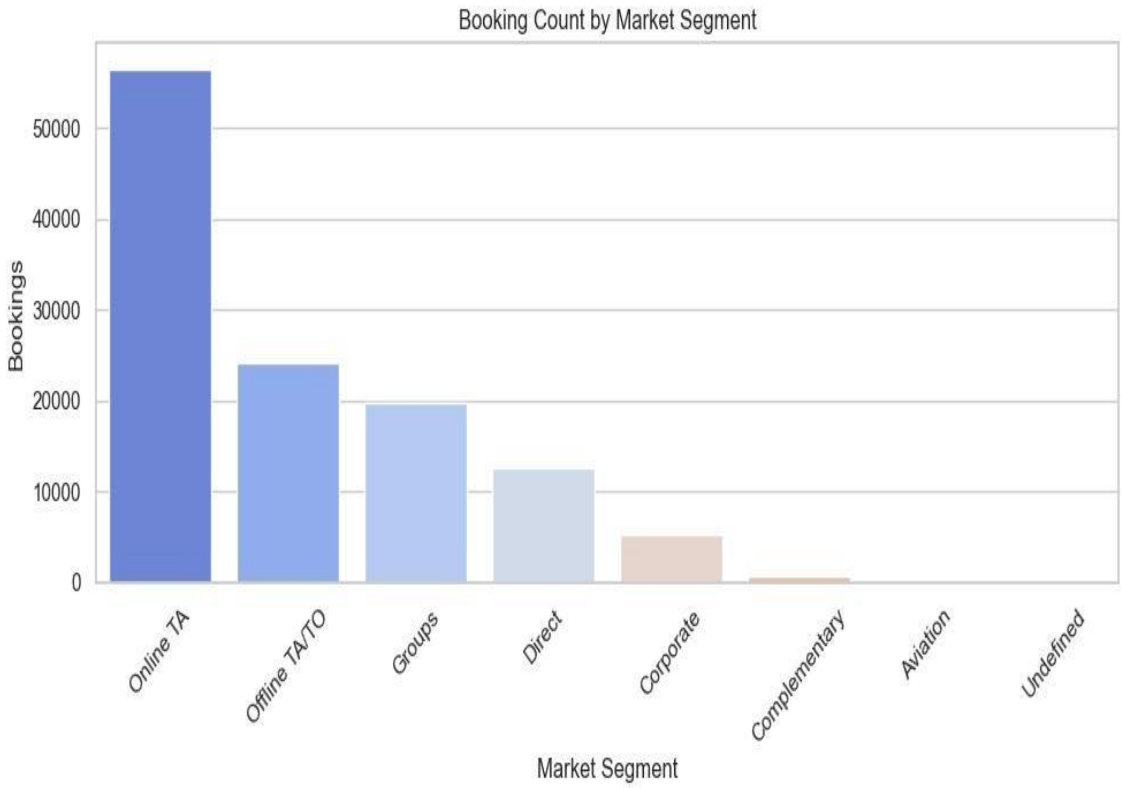
Cancellation rate WITH special requests: 21.74%
Cancellation rate WITHOUT special requests: 47.72%

#2.Factors Influencing Cancellation

```
import pandas as pd

# Analyze cancellation rates based on different factors (e.g., lead time, booking
 channel, deposit type)
# Example: Cancellation rate by lead time
cancellation_by_lead_time = df.groupby('lead_time')['is_canceled'].mean()
print(cancellation_by_lead_time) lead_time

0
0.067770 1
0.092775 2
0.102948 3
0.100220
4      0.102624
       ...
622    1.000000
626    1.000000
629    1.000000
709    0.000000
737    0.000000
Name: is_canceled, Length: 479, dtype: float64
```

Market Segment Analysis

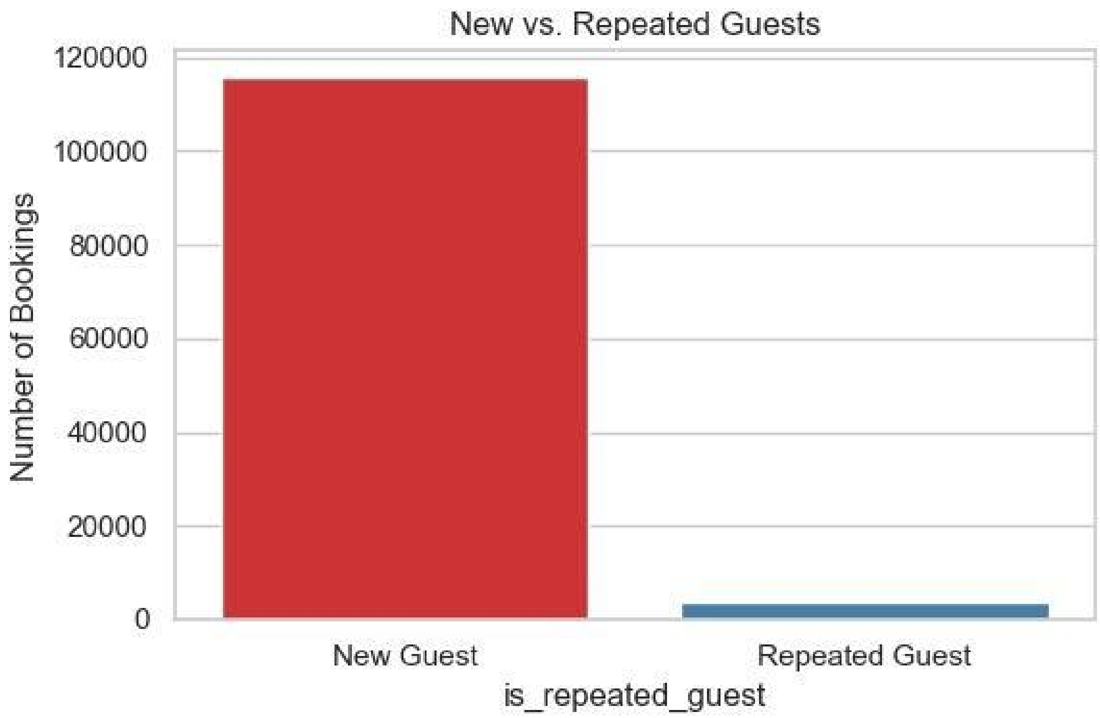
```
# Market segment distribution plt.figure(figsize=(10, 5))
sns.countplot(data=df, x='market_segment',
order=df['market_segment'].value_counts().index, palette='coolwarm')
plt.title('Booking Count by Market Segment') plt.xlabel('Market
Segment') plt.ylabel('Bookings') plt.xticks(rotation=45) plt.tight_layout()
plt.show()
```



Repeated Guest Ratio

Count of repeated guests repeated_guest_counts = df['is_repeated_guest'].value_counts()

plt.figure(figsize=(6, 4)) sns.barplot(x=repeated_guest_counts.index, y=repeated_guest_counts.values, palette='Set1') plt.xticks([0, 1], ['New Guest', 'Repeated Guest']) plt.title('New vs. Repeated Guests') plt.ylabel('Number of Bookings') plt.tight_layout() plt.show()



Correlation Matrix (Find Relationships Between Key Variables)

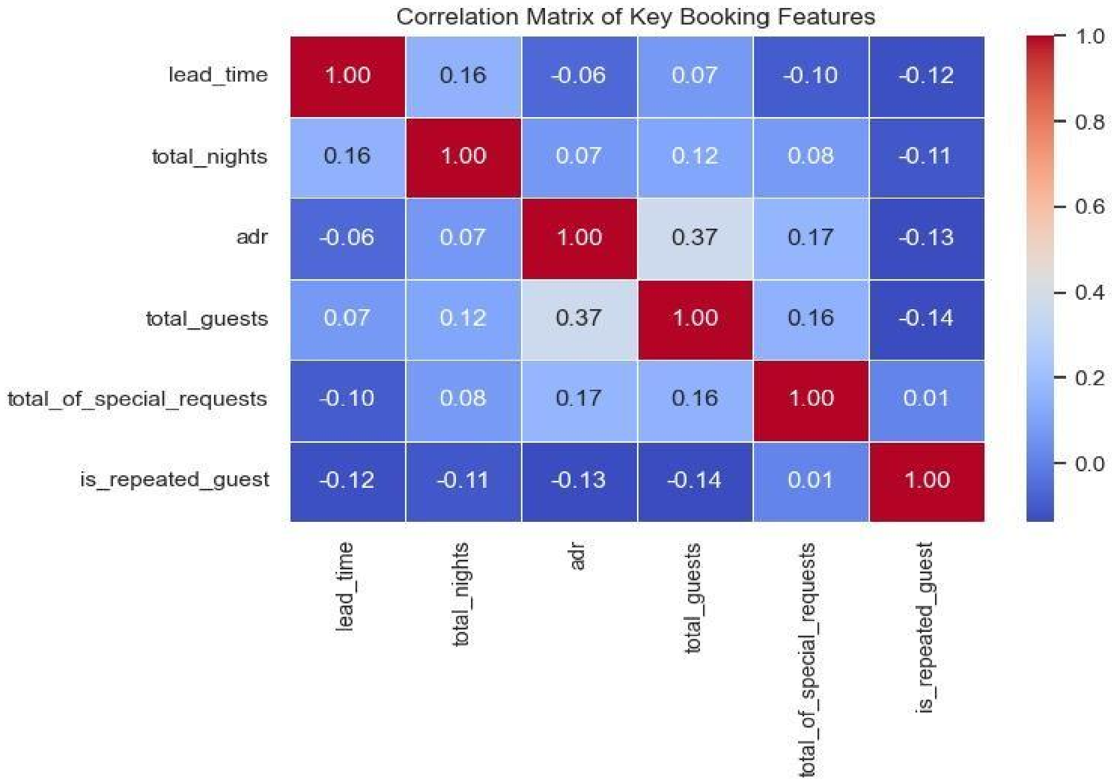
This helps us understand:

How features like lead_time, adr, total_nights, and special_requests are related

Spot patterns for prediction or further insights

```
# Select relevant numerical features corr_features =
['lead_time', 'total_nights', 'adr', 'total_guests',
'total_of_special_requests', 'is_repeated_guest']

# Calculate correlation matrix
corr_matrix = df[corr_features].corr()

# Plot heatmap
plt.figure(figsize=(8, 6)) sns.heatmap(corr_matrix, annot=True,
cmap='coolwarm', fmt=".2f", linewidths=0.5)
```

```
plt.title('Correlation Matrix of Key Booking Features')
plt.tight_layout() plt.show()
```



Predicting Cancellations with Machine Learning

Predict whether a booking will be canceled (is_canceled = 1) using other features

```
# Step 1: Select Features features = ['lead_time',
'total_nights', 'adr', 'total_guests',
        'deposit_type', 'customer_type', 'previous_cancellations',
'booking_changes']
X = df[features] y =
df['is_canceled']

# Step 2: Encode categorical features for
col in ['deposit_type', 'customer_type']:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
```

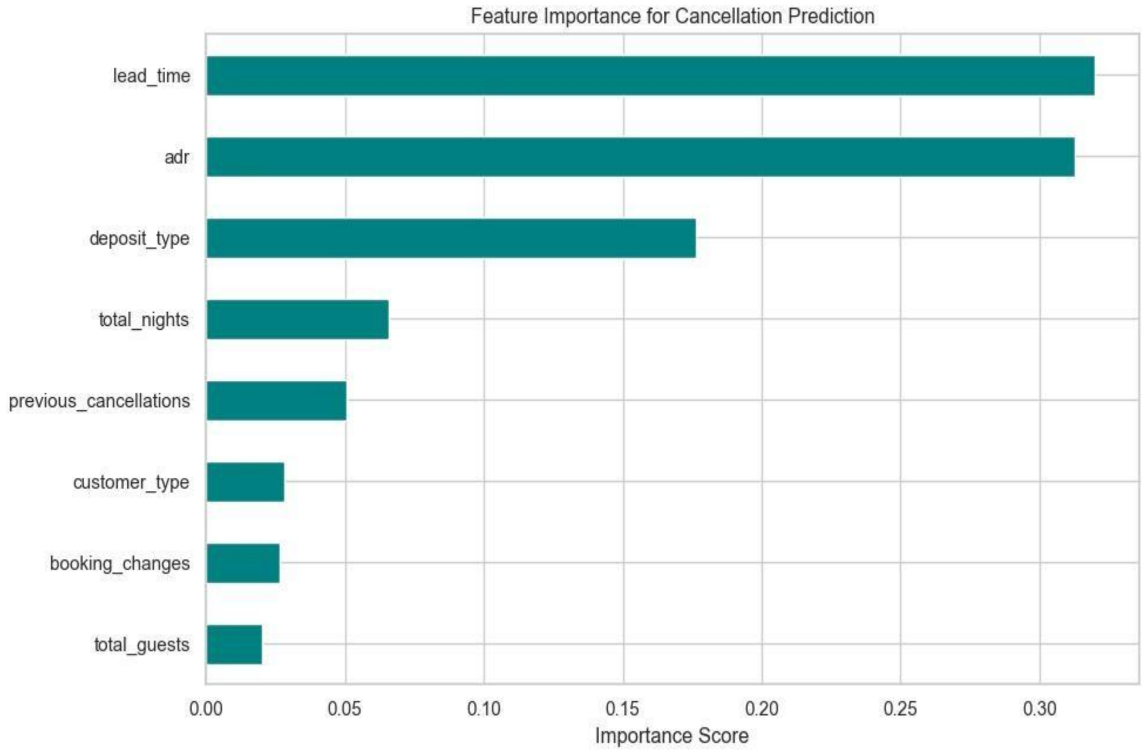
```
# Step 3: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 4: Train the model model =
RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 5: Evaluate y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
C:\Users\varda\AppData\Local\Temp\ipykernel_20676\3347513064.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandasdocs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy X[col] = le.fit_transform(X[col])
C:\Users\varda\AppData\Local\Temp\ipykernel_20676\3347513064.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandasdocs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy X[col] = le.fit_transform(X[col])
Accuracy: 0.8079403635145322
[[13154 1753]
[ 2833 6138]] precision recall f1-score
          support
```

0	0.82	0.88	0.85	14907
1	0.78	0.68	0.73	8971
accuracy			0.81	23878
macro avg	0.80	0.78	0.79	23878
weighted avg	0.81	0.81	0.81	23878

Feature Importance Visualization

feature_importances = pd.Series(model.feature_importances_, index=X.columns) feature_importances = feature_importances.sort_values(ascending=True)

plt.figure(figsize=(10, 6))
feature_importances.plot(kind='barh', color='teal')
plt.title('Feature Importance for Cancellation Prediction')
plt.xlabel('Importance Score') plt.tight_layout()
plt.show()s



Conclusion

This project provided actionable insights into guest behavior and pricing strategies. The cancellation prediction model helps hotels:

Improve resource planning

Identify risky bookings

Reduce no-shows and revenue loss