Hotel Booking Cancellations Analysis

Introduction

This notebook analyzes a hotel booking dataset to understand patterns and factors influencing cancellations. The dataset includes various features related to bookings, customer demographics, and stay details. The objective is to extract insights and build predictive models to minimize cancellations and optimize hotel operations.

Dataset Description

The dataset contains the following columns:

- hotel: The type of hotel (city hotel or resort hotel).
- is_canceled: Indicates if the booking was canceled (1) or not (0).
- lead_time: The number of days between the booking date and the arrival date.
- arrival_date_year: The year of arrival date.
- arrival_date_month: The month of arrival date.
- arrival_date_week_number: The week number of the arrival date.
- arrival_date_day_of_month: The day of the month of the arrival date.
- stays_in_weekend_nights: Number of weekend nights (Saturday and Sunday) the guest stayed or booked to stay at the hotel.
- stays_in_week_nights: Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel.
- adults: Number of adults.
- children: Number of children.
- babies: Number of babies.
- meal: Type of meal booked (e.g., BB - Bed & Breakfast, HB - Half Board, FB - Full Board).
- country: Country of origin of the guest.
- market_segment: Market segment designation (e.g., Direct, Corporate, Online TA).
- distribution_channel: Booking distribution channel (e.g., Direct, TA/TO).
- is_repeated_guest: Indicates if the guest is a repeated guest (1) or not (0).
- previous_cancellations: Number of previous bookings that were canceled by the customer prior to the current booking.
- previous_bookings_not_canceled: Number of previous bookings not canceled by the customer prior to the current booking.
- reserved_room_type: Code of room type reserved.

assigned_room_type: Code of room type assigned.

- booking_changes: Number of changes/amendments made to the booking.
- deposit_type: Type of deposit made (e.g., No Deposit, Non Refund, Refundable).
- agent: ID of the travel agent who made the booking.

- company: ID of the company/entity that made the booking or responsible for the booking.
- days_in_waiting_list: Number of days the booking was in the waiting list before it was confirmed to the customer.
- customer_type: Type of booking (e.g., Contract, Group, Transient).
- adr: Average Daily Rate, as defined by dividing the sum of all lodging transactions by the total number of staying nights.
- required_car_parking_spaces: Number of car parking spaces required by the customer.
- total_of_special_requests: Total number of special requests made by the customer (e.g., high floor, crib, etc.).
- reservation_status: Reservation last status (e.g., Canceled, Check-Out, No-Show).
- reservation_status_date: Date at which the last status was set.

To extract data from an API, you'll typically follow these steps:

Understand the API Documentation: Review the API documentation to understand the endpoints, request methods, required parameters, and authentication mechanisms.

Set Up Your Environment: Ensure you have the necessary libraries installed. For Python, common libraries include requests for making HTTP requests and json for handling JSON data.

Make the API Request: Use the requests library to make a GET or POST request to the API endpoint.

Handle the Response: Check the response status code to ensure the request was successful. If successful, parse the JSON data from the response.

Process the Data: Depending on your needs, you can process, analyze, or store the extracted data.

In [87]:
```
# Import necessary libraries
import requests
import pandas as pd
import json
```

In [ ]:
```
# Define the API endpoint and any necessary headers (e.g., for authentication)
api_url = "https://api.hotel.com/v1/bookings"
headers = {
    "Authorization": "0524sdfw_45723dfrty00ty0w"
}

# Make the API request
response = requests.get(api_url, headers=headers)

# Check if the request was successful
if response.status_code == 200:
    # Parse the response JSON into a dictionary
    data = response.json()

    # If the data is nested, you may need to adjust this step to flatten it appropriatel
    bookings = data['bookings']

    # Convert the dictionary to a Pandas DataFrame
    df = pd.DataFrame(bookings)
```

```
    # Display the DataFrame
    print(df.head())
else:
    print(f"Failed to retrieve data: {response.status_code}")

# Save the DataFrame to a CSV file
df.to_csv('hotel_bookings.csv', index=False)
```

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:

```
hotel_df = pd.read_csv(r'hotel_bookings.csv')
```

In [6]:

```
hotel_df.shape
```

Out[6]:

```
(119390, 32)
```

In [7]:

```
hotel_df.dtypes
```

Out[7]:

```
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
```

In [8]:

```
hotel_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
```

Types of Data

1. Categorical data refers to a data type that can be stored

into groups/categories/labels.

- Examples of categorical variables are :

age group, educational level,blood type etc.

2. Numerical data refers to the data that is in the form of numbers,

- Examples of numerical data are height, weight, age etc.

Numerical data has two categories: discrete data and continuous data

- Discrete data : It basically takes countable numbers like 1, 2, 3, 4, 5, and so on. In case of infinity, these numbers will keep going on. Age of a fly : 8 , 9 day etc..
- Continuous data : which is continuous in nature amount of sugar , 11.2 kg, temp of a cit, your bank balance !

For example, salary levels and performance classifications are discrete variables, whereas height and weight are continuous variables.

- Categorical data has : Object & bool data-types
- Numerical data have : Integer & Float data-type

Variations of int are : ('int64','int32','int16') in numpy library.

- Int16 is a 16 bit signed integer , it means it can store both positive & negative values

-- int16 has has a range of (2^15 − 1) to -2^15 -- int16 has a length of 16 bits (2 bytes).. ie Int16 uses 16 bits

- Int32 is a 32 bit signed integer , it means it stores both positive & negative values

-- int32 has has a range of (2^{31} − 1) to -2^31 -- int32 has a length of 32 bits (4 bytes),, ie Int32 uses 32 bits

- Int64 is a 64 bit signed integer , it means it can store both positive & negative values

-- int64 has has a range of (2^63 − 1) to -2^63 -- int64 has a length of 64 bits (8 bytes) , ie Int64 uses 64 bits.

The only difference is that int64 has max range of storing numbers , then comes int32 , then 16 , then int8

That means that Int64's take up twice as much memory-and doing operations on them may be a lot slower in some machine architectures.

However, Int64's can represent numbers much more accurately than 32 bit floats.They also allow much larger numbers to be stored..

Data Cleaning

In [9]:
```
hotel_df.head(6)
```

Out[9]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arri
0	Resort Hotel	0	342	2015	July	27	

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arri
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	
3	Resort Hotel	0	13	2015	July	27	
4	Resort Hotel	0	14	2015	July	27	
5	Resort Hotel	0	14	2015	July	27	

6 rows × 32 columns

In [10]:

```
hotel_df.columns
```

Out[10]:

```
Index(['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'],
      dtype='object')
```

In [12]:

```
# Adults,babies & children cant be zero at a same time because booking couldn't be possi
filter1 = (hotel_df['children']==0) & (hotel_df['adults']==0) & (hotel_df['babies']==0)
```

In [13]:

```
filter1
```

Out[13]:

```
0         False
1         False
2         False
3         False
4         False
          ...
119385    False
119386    False
119387    False
119388    False
119389    False
Length: 119390, dtype: bool
```

In [14]:

```
hotel_df[filter1]
```

Out[14]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number
2224	Resort Hotel	0	1	2015	October	41
2409	Resort Hotel	0	0	2015	October	42
3181	Resort Hotel	0	36	2015	November	47
3684	Resort Hotel	0	165	2015	December	53
3708	Resort Hotel	0	165	2015	December	53
...	...	...	...	...	...	...
115029	City Hotel	0	107	2017	June	26
115091	City Hotel	0	1	2017	June	26
116251	City Hotel	0	44	2017	July	28
116534	City Hotel	0	2	2017	July	28
117087	City Hotel	0	170	2017	July	30

180 rows × 32 columns

In [15]:
```
hotel_df.shape
```

Out[15]:
```
(119390, 32)
```

In [16]:
```
hotel_df[filter1].shape
```

Out[16]:
```
(180, 32)
```

In [17]:
```
hotel_df2 = hotel_df[~filter1]
```

In [18]:
```
hotel_df2.shape
```

Out[18]:
```
(119210, 32)
```

In [19]:
```
len(hotel_df2[(hotel_df2['children'] == 1) & (hotel_df2['adults'] == 0) & (hotel_df2['ba
```

Out[19]:
```
4
```

Removing Duplicated data

In [20]:

```
hotel_df2.duplicated()
```

Out[20]:

```
0         False
1         False
2         False
3         False
4         False
          ...
119385    False
119386    False
119387    False
119388    False
119389    False
Length: 119210, dtype: bool
```

In [21]:

```
hotel_df2.duplicated().sum()
```

Out[21]:

```
31980
```

In [22]:

```
hotel_data = hotel_df2.drop_duplicates()
```

In [23]:

```
hotel_data.shape
```

Out[23]:

```
(87230, 32)
```

In [24]:

```
hotel_data.head(7)
```

Out[24]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arri
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	
3	Resort Hotel	0	13	2015	July	27	
4	Resort Hotel	0	14	2015	July	27	
6	Resort Hotel	0	0	2015	July	27	
7	Resort Hotel	0	9	2015	July	27	

7 rows × 32 columns

3. Performing descriptive analysis

In [26]:

```
hotel_data.columns
```

Out[26]:

```
Index(['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'],
      dtype='object')
```

In [28]:

```
# getting (mean, median , std , percentile) of above features
hotel_data.describe()
```

Out[28]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_mo
count	87230.000000	87230.000000	87230.000000	87230.000000	87230.000
mean	0.275238	79.971019	2016.210352	26.835091	15.815
std	0.446637	86.058683	0.686064	13.669216	8.835
min	0.000000	0.000000	2015.000000	1.000000	1.000
25%	0.000000	11.000000	2016.000000	16.000000	8.000
50%	0.000000	49.000000	2016.000000	27.000000	16.000
75%	1.000000	125.000000	2017.000000	37.000000	23.000
max	1.000000	737.000000	2017.000000	53.000000	31.000

In [29]:

```
hotel_data[['lead_time' , 'total_of_special_requests' , 'adr']].describe()
```

Out[29]:

	lead_time	total_of_special_requests	adr
count	87230.000000	87230.000000	87230.000000
mean	79.971019	0.698934	106.518031
std	86.058683	0.832051	54.891227
min	0.000000	0.000000	-6.380000
25%	11.000000	0.000000	72.250000
50%	49.000000	0.000000	98.200000
75%	125.000000	1.000000	134.100000

	lead_time	total_of_special_requests	adr
max	737.000000	5.000000	5400.000000

In [30]:

```
hotel_data[['lead_time' , 'total_of_special_requests' , 'adr']].describe().T
```

Out[30]:

	count	mean	std	min	25%	50%	75%	max
lead_time	87230.0	79.971019	86.058683	0.00	11.00	49.0	125.0	737.0
total_of_special_requests	87230.0	0.698934	0.832051	0.00	0.00	0.0	1.0	5.0
adr	87230.0	106.518031	54.891227	-6.38	72.25	98.2	134.1	5400.0

In [31]:

```
hotel_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 87230 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   hotel                           87230 non-null  object 
 1   is_canceled                     87230 non-null  int64  
 2   lead_time                       87230 non-null  int64  
 3   arrival_date_year               87230 non-null  int64  
 4   arrival_date_month              87230 non-null  object 
 5   arrival_date_week_number        87230 non-null  int64  
 6   arrival_date_day_of_month       87230 non-null  int64  
 7   stays_in_weekend_nights         87230 non-null  int64  
 8   stays_in_week_nights            87230 non-null  int64  
 9   adults                          87230 non-null  int64  
 10  children                        87226 non-null  float64
 11  babies                          87230 non-null  int64  
 12  meal                            87230 non-null  object 
 13  country                         86783 non-null  object 
 14  market_segment                  87230 non-null  object 
 15  distribution_channel            87230 non-null  object 
 16  is_repeated_guest               87230 non-null  int64  
 17  previous_cancellations          87230 non-null  int64  
 18  previous_bookings_not_canceled  87230 non-null  int64  
 19  reserved_room_type              87230 non-null  object 
 20  assigned_room_type              87230 non-null  object 
 21  booking_changes                 87230 non-null  int64  
 22  deposit_type                    87230 non-null  object 
 23  agent                           75089 non-null  float64
 24  company                         5237 non-null   float64
 25  days_in_waiting_list            87230 non-null  int64  
 26  customer_type                   87230 non-null  object 
 27  adr                             87230 non-null  float64
 28  required_car_parking_spaces     87230 non-null  int64  
 29  total_of_special_requests       87230 non-null  int64  
 30  reservation_status              87230 non-null  object 
 31  reservation_status_date         87230 non-null  object 
dtypes: float64(4), int64(16), object(12)
memory usage: 22.0+ MB
```

12 features belong to object data-type , ie.. in context to Python , they belong to string data-type

16 features belong to int64 nature

4 features belong to float64 nature. The memory usage of a DataFrame (including the index) is shown when calling the info().

The + symbol indicates that the true memory usage could be higher, because pandas does not count the memory used by values in columns with dtype=object

Passing memory_usage='deep' will enable a more accurate memory usage report .

In [32]:

```
hotel_data.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 87230 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   hotel                           87230 non-null  object
 1   is_canceled                     87230 non-null  int64
 2   lead_time                       87230 non-null  int64
 3   arrival_date_year               87230 non-null  int64
 4   arrival_date_month              87230 non-null  object
 5   arrival_date_week_number        87230 non-null  int64
 6   arrival_date_day_of_month       87230 non-null  int64
 7   stays_in_weekend_nights         87230 non-null  int64
 8   stays_in_week_nights            87230 non-null  int64
 9   adults                          87230 non-null  int64
 10  children                        87226 non-null  float64
 11  babies                          87230 non-null  int64
 12  meal                            87230 non-null  object
 13  country                         86783 non-null  object
 14  market_segment                  87230 non-null  object
 15  distribution_channel            87230 non-null  object
 16  is_repeated_guest               87230 non-null  int64
 17  previous_cancellations          87230 non-null  int64
 18  previous_bookings_not_canceled  87230 non-null  int64
 19  reserved_room_type              87230 non-null  object
 20  assigned_room_type              87230 non-null  object
 21  booking_changes                 87230 non-null  int64
 22  deposit_type                    87230 non-null  object
 23  agent                           75089 non-null  float64
 24  company                         5237 non-null   float64
 25  days_in_waiting_list            87230 non-null  int64
 26  customer_type                   87230 non-null  object
 27  adr                             87230 non-null  float64
 28  required_car_parking_spaces     87230 non-null  int64
 29  total_of_special_requests       87230 non-null  int64
 30  reservation_status              87230 non-null  object
 31  reservation_status_date         87230 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 77.1 MB
```

Perform Basic Analysis

In [33]:
```
not_cancelled = hotel_data[hotel_data['is_canceled']==0]
```

In [34]:
```
not_cancelled.head(3)
```

Out[34]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arri
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	

3 rows × 32 columns

In [35]:
```
country_wise_data = not_cancelled['country'].value_counts().reset_index()
```

In [36]:
```
country_wise_data
```

Out[36]:

	country	count
0	PRT	17573
1	GBR	8440
2	FRA	7091
3	ESP	5382
4	DEU	4332
...	...	...
160	ZMB	1
161	SYC	1
162	MDG	1
163	SMR	1
164	FRO	1

165 rows × 2 columns

In [40]:
```
country_wise_data.columns = ['country' , 'No of guests']
```

In [41]:
```
country_wise_data
```

Out[41]:

	country	No of guests
0	PRT	17573
1	GBR	8440
2	FRA	7091
3	ESP	5382
4	DEU	4332
...	...	...
160	ZMB	1
161	SYC	1
162	MDG	1
163	SMR	1
164	FRO	1

165 rows × 2 columns

In [37]:

```
!pip install chart-studio
!pip install plotly
```

```
Collecting chart-studio
  Downloading chart_studio-1.1.0-py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: plotly in c:\users\kanis\anaconda3\lib\site-packages (fro
m chart-studio) (5.9.0)
Requirement already satisfied: requests in c:\users\kanis\anaconda3\lib\site-packages (f
rom chart-studio) (2.31.0)
Collecting retrying>=1.3.3 (from chart-studio)
  Downloading retrying-1.3.4-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: six in c:\users\kanis\anaconda3\lib\site-packages (from c
hart-studio) (1.16.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\kanis\anaconda3\lib\site-pack
ages (from plotly->chart-studio) (8.2.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\kanis\anaconda3\lib
\site-packages (from requests->chart-studio) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\kanis\anaconda3\lib\site-package
s (from requests->chart-studio) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\kanis\anaconda3\lib\site-p
ackages (from requests->chart-studio) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\kanis\anaconda3\lib\site-p
ackages (from requests->chart-studio) (2024.6.2)
Downloading chart_studio-1.1.0-py3-none-any.whl (64 kB)
   ---------------------------------------- 0.0/64.4 kB ? eta -:--:--
   ---------------------------------------- 0.0/64.4 kB ? eta -:--:--
   ------------------------ -------------- 41.0/64.4 kB 653.6 kB/s eta 0:00:01
   ---------------------------------------- 64.4/64.4 kB 694.6 kB/s eta 0:00:00
Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Installing collected packages: retrying, chart-studio
Successfully installed chart-studio-1.1.0 retrying-1.3.4
Requirement already satisfied: plotly in c:\users\kanis\anaconda3\lib\site-packages (5.
9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\kanis\anaconda3\lib\site-pack
ages (from plotly) (8.2.2)
```

In [38]:

```
### establishing the entire set-up of Plotly..

import chart_studio.plotly as py
## chart_studio provides a web-service for hosting graphs!

import plotly.graph_objs as go
import plotly.express as px

from plotly.offline import download_plotlyjs , init_notebook_mode , plot , iplot

## iplot() when working in a Jupyter Notebook to
## display the plot in the Ipython notebook.

init_notebook_mode(connected=True)
```

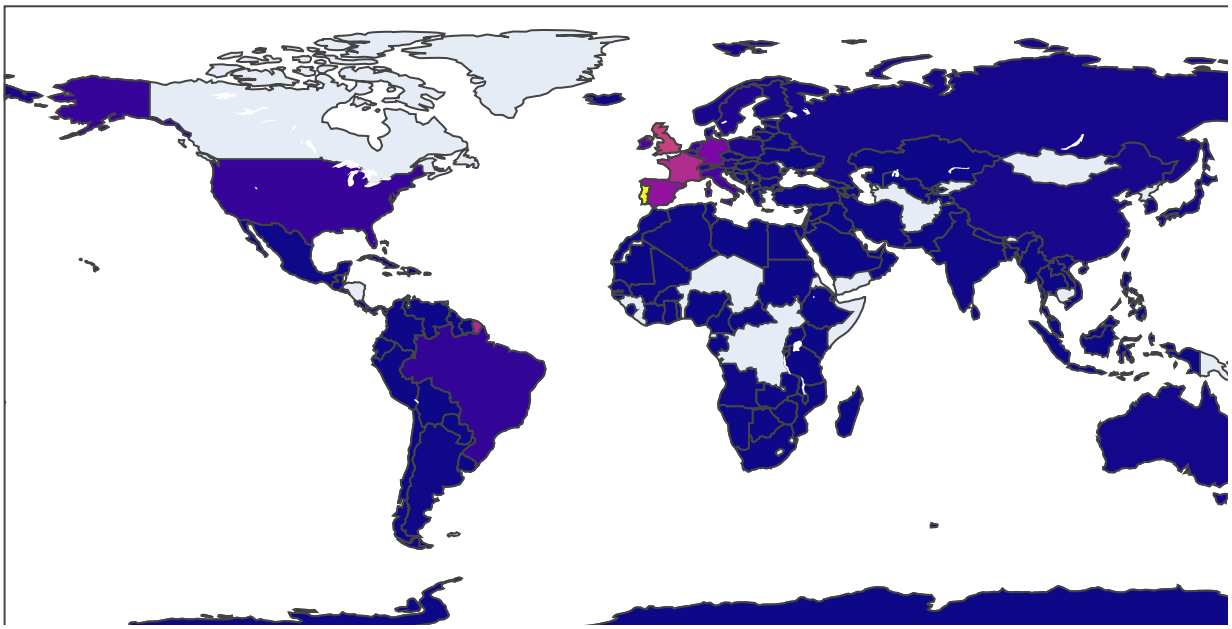
In [42]:

```
# show on map

map_guest = px.choropleth(data_frame = country_wise_data ,
              locations= country_wise_data['country'] ,
              color=country_wise_data['No of guests'] ,
              hover_name=country_wise_data['country'] ,
              title= "Native country of Guests")
```

In [43]:

```
map_guest.show()
```

Native country of Guests



Most guests are from Portugal and other countries in Europe

5. Is any difference between assigned and reserved room types or not ?

In [44]:

```
pivot = pd.crosstab(index = hotel_data['reserved_room_type'] , columns=hotel_data['assig
```

In [45]:

```
pivot
```

Out[45]:

assigned_room_type	A	B	C	D	E	F	G	H	I	K	L	All
reserved_room_type												
A	45850	892	1253	6402	1034	390	176	94	205	140	0	56436
B	106	872	0	5	2	2	8	0	0	1	0	996
C	5	2	866	6	4	2	10	9	10	0	0	914
D	295	27	32	15979	657	199	82	9	67	29	0	17376
E	15	2	6	22	5458	383	97	4	40	9	0	6036
F	6	14	0	4	31	2636	113	3	10	3	0	2820

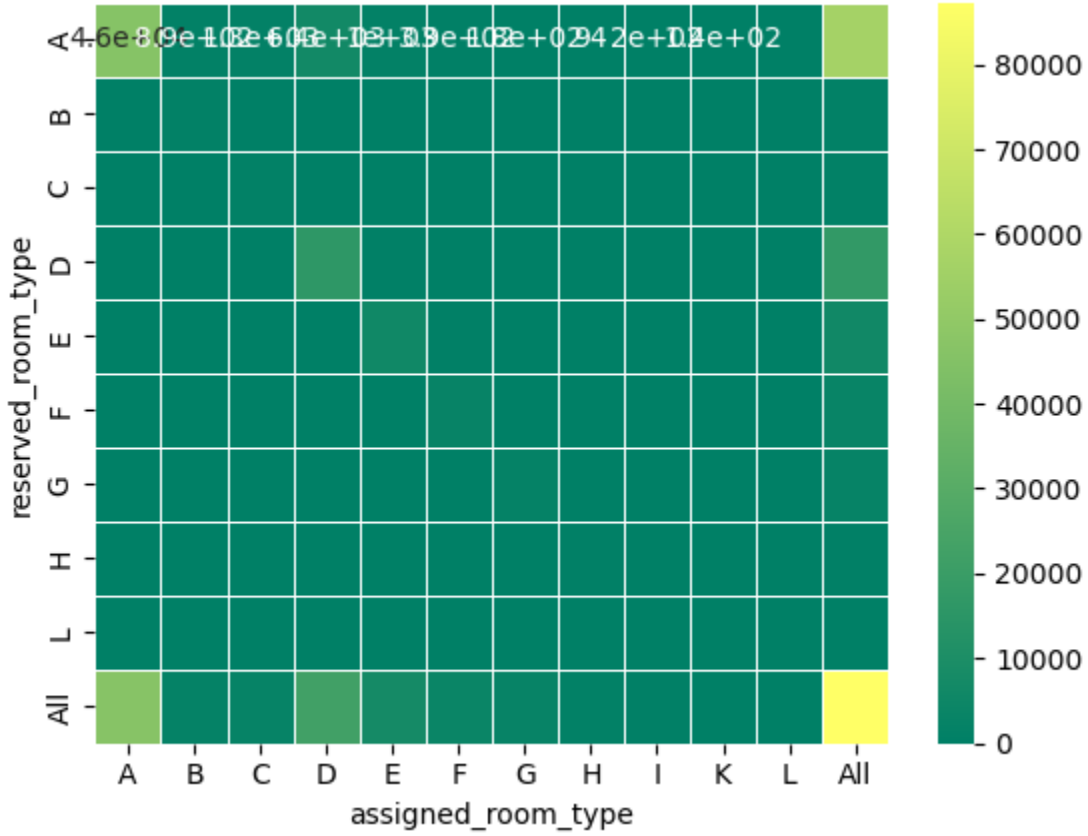
assigned_room_type	A	B	C	D	E	F	G	H	I	K	L	All
reserved_room_type												
G	5	1	2	0	4	14	1999	7	15	3	0	2050
H	0	0	0	1	0	0	10	579	6	0	0	596
L	1	1	1	0	0	1	0	1	0	0	1	6
All	46283	1811	2160	22419	7190	3627	2495	706	353	185	1	87230

In [48]:

```
sns.heatmap(pivot,annot=True,linewidth=0.5, cmap='summer')
```

Out[48]:

```
<Axes: xlabel='assigned_room_type', ylabel='reserved_room_type'>
```



Meaningful insight from this :

- For A category room , 56436 folks have reserved "A" & 45850 folks get assigned_room as "A".. & rest are unable to get !
- For B category room , 996 folks have reserved "B" & 872 folks get assigned_room as "B".. & rest are unable to get !

In [46]:

```
# we will say just normalize over row , hence we need to pass normalize = 'index'

pivot_normalize = pd.crosstab(index = hotel_data['reserved_room_type'] , columns=hotel_d
```

In [47]:

```
pivot_normalize
```

Out[47]:

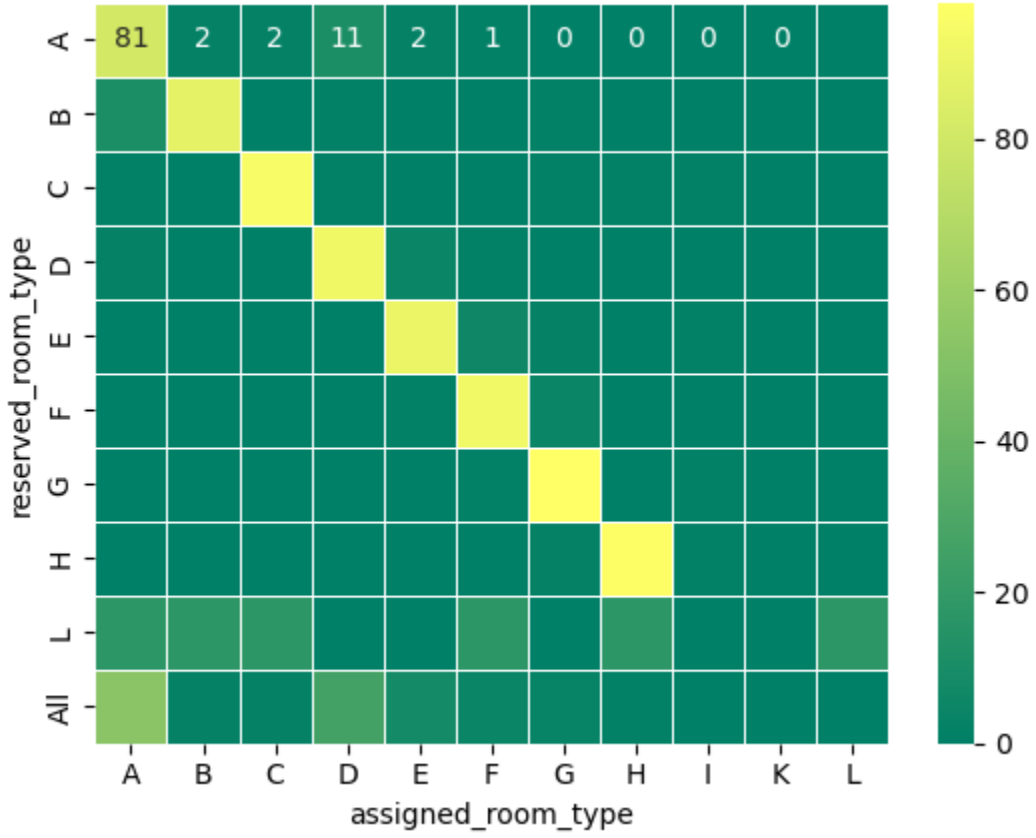
assigned_room_type	A	B	C	D	E	F	G	H	I	K	L
reserved_room_type											
A	81.0	2.0	2.0	11.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0
B	11.0	88.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
C	1.0	0.0	95.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
D	2.0	0.0	0.0	92.0	4.0	1.0	0.0	0.0	0.0	0.0	0.0
E	0.0	0.0	0.0	0.0	90.0	6.0	2.0	0.0	1.0	0.0	0.0
F	0.0	0.0	0.0	0.0	1.0	93.0	4.0	0.0	0.0	0.0	0.0
G	0.0	0.0	0.0	0.0	0.0	1.0	98.0	0.0	1.0	0.0	0.0
H	0.0	0.0	0.0	0.0	0.0	0.0	2.0	97.0	1.0	0.0	0.0
L	17.0	17.0	17.0	0.0	0.0	17.0	0.0	17.0	0.0	0.0	17.0
All	53.0	2.0	2.0	26.0	8.0	4.0	3.0	1.0	0.0	0.0	0.0

In [49]:

```
sns.heatmap(pivot_normalize,annot=True,linewidth=0.5, cmap='summer')
```

Out[49]:

```
<Axes: xlabel='assigned_room_type', ylabel='reserved_room_type'>
```



Is any difference between assigned and reserved room type ? Yes

6. Bookings by market segment

In [50]:

```
hotel_data['market_segment'].value_counts()
```

Out[50]:

```
market_segment
Online TA        51553
Offline TA/TO    13855
Direct           11780
Groups            4922
Corporate         4200
Complementary      692
Aviation           226
Undefined            2
Name: count, dtype: int64
```

In [51]:

```
hotel_data['market_segment'].value_counts().values
```

Out[51]:

```
array([51553, 13855, 11780,  4922,  4200,   692,   226,     2],
      dtype=int64)
```

In [52]:

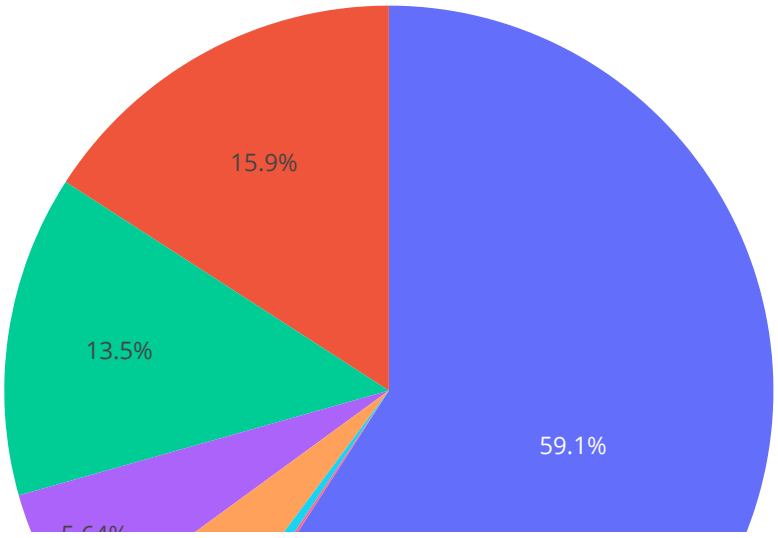
```
hotel_data['market_segment'].value_counts().index
```

Out[52]:

```
Index(['Online TA', 'Offline TA/TO', 'Direct', 'Groups', 'Corporate',
       'Complementary', 'Aviation', 'Undefined'],
      dtype='object', name='market_segment')
```

In [54]:

```
# pie plot

fig = px.pie(hotel_data ,
      values = hotel_data['market_segment'].value_counts().values ,
      names = hotel_data['market_segment'].value_counts().index)
fig.show()
```



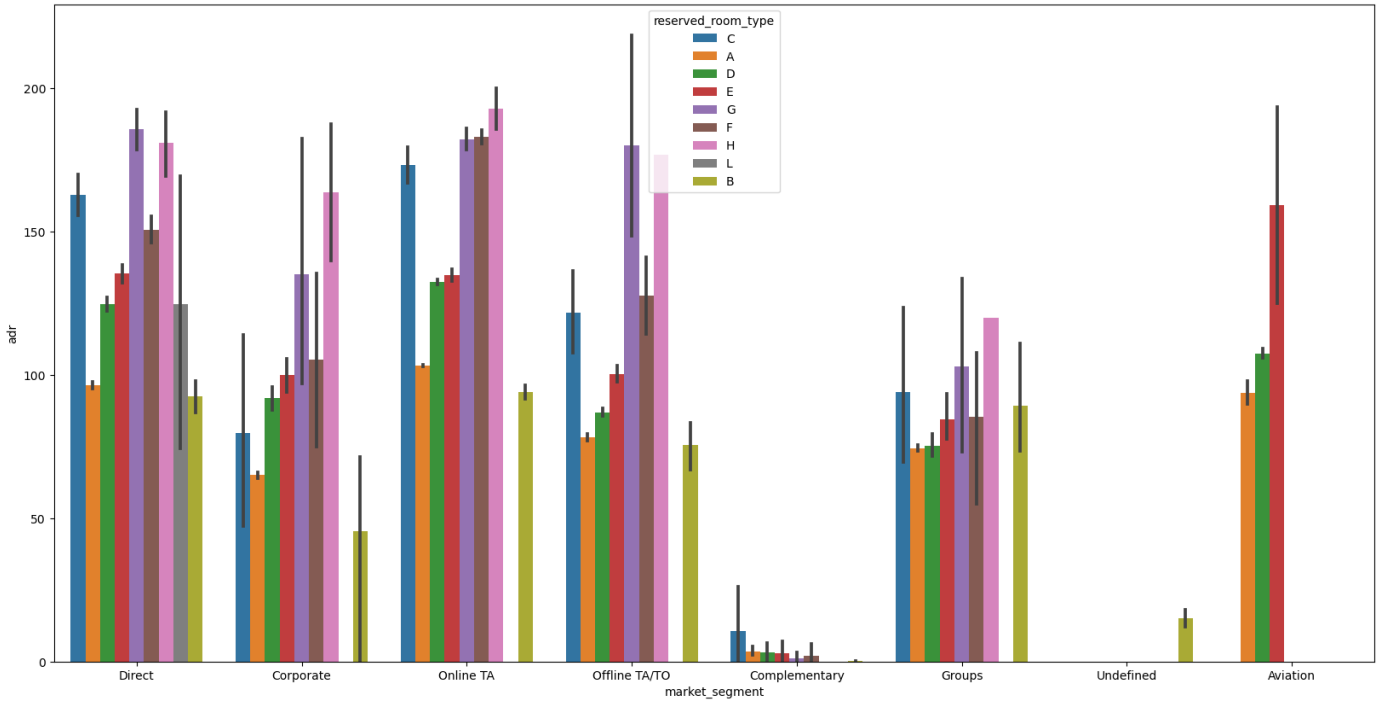
Most of the bookings have been done in Online mode

In [57]:

```
# this plot tells avg adr of various room-types for all the market segment.
plt.figure(figsize=(20,10))
sns.barplot(x="market_segment" , y="adr" , hue="reserved_room_type" , data=hotel_data)
```

Out[57]:

```
<Axes: xlabel='market_segment', ylabel='adr'>
```



7.Total guests arrival on each day:

Is there any pattern in guests arrival , ie whether Guests number have increased or not ?

In [59]:

```
hotel_data['arrival_date_month'].unique()
```

Out[59]:

```
array(['July', 'August', 'September', 'October', 'November', 'December',
       'January', 'February', 'March', 'April', 'May', 'June'],
      dtype=object)
```

In [60]:

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

In [61]:

```
hotel_data['arrival_date_month_index'] = hotel_data['arrival_date_month'].map(dict_month
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_4324\2077173483.py:1: SettingWithCopyWarnin
g:


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/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

In [62]:

```
hotel_data['arrival_date_month'].unique()
```

Out[62]:

```
array(['July', 'August', 'September', 'October', 'November', 'December',
       'January', 'February', 'March', 'April', 'May', 'June'],
      dtype=object)
```

In [63]:

```
hotel_data['arrival_date_month_index']
```

Out[63]:

```
0         7
1         7
2         7
3         7
4         7
         ..
119385    8
119386    8
119387    8
119388    8
119389    8
Name: arrival_date_month_index, Length: 87230, dtype: int64
```

In [64]:

```
hotel_data.columns
```

Out[64]:

```
Index(['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',
       'arrival_date_month_index'],
      dtype='object')
```

In [65]:

```
hotel_data[['arrival_date_year','arrival_date_month_index','arrival_date_day_of_month']]
```

Out[65]:

	arrival_date_year	arrival_date_month_index	arrival_date_day_of_month
0	2015	7	1
1	2015	7	1
2	2015	7	1
3	2015	7	1
4	2015	7	1
...	...	...	...
119385	2017	8	30
119386	2017	8	31
119387	2017	8	31
119388	2017	8	31
119389	2017	8	29

87230 rows × 3 columns

In [66]:

```
## we need to use .astype(str) to convert int values to string,otherwise we are unable t
hotel_data['arrival_date'] = hotel_data['arrival_date_year'].astype(str) + '-' + hotel_d
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_4324\515530105.py:2: 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/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

In [67]:

```
hotel_data['arrival_date']
```

Out[67]:

```
0          2015-7-1
1          2015-7-1
2          2015-7-1
3          2015-7-1
4          2015-7-1
            ...
119385    2017-8-30
119386    2017-8-31
119387    2017-8-31
119388    2017-8-31
119389    2017-8-29
Name: arrival_date, Length: 87230, dtype: object
```

Guests

In [68]:

```
hotel_data[['adults', 'children', 'babies']]
```

Out[68]:

	adults	children	babies
0	2	0.0	0
1	2	0.0	0
2	1	0.0	0
3	1	0.0	0
4	2	0.0	0
...	...	...	...
119385	2	0.0	0
119386	3	0.0	0
119387	2	0.0	0
119388	2	0.0	0
119389	2	0.0	0

87230 rows × 3 columns

In [69]:

```
hotel_data['Total_guests'] = hotel_data['adults'] + hotel_data['children'] + hotel_data[
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_4324\4038671766.py:1: SettingWithCopyWarnin
g:


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/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

In [70]:

```
hotel_data['Total_guests']
```

Out[70]:

```
0         2.0
1         2.0
2         1.0
3         1.0
4         2.0
         ...
119385    2.0
119386    3.0
119387    2.0
119388    2.0
119389    2.0
Name: Total_guests, Length: 87230, dtype: float64
```

In [71]:

```
hotel_data.columns
```

Out[71]:

```
Index(['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',
       'arrival_date_month_index', 'arrival_date', 'Total_guests'],
      dtype='object')
```

In [72]:

```
hotel_data.shape
```

Out[72]:

```
(87230, 35)
```

In [73]:

```
hotel_data[['arrival_date', 'Total_guests']]
```

Out[73]:

	arrival_date	Total_guests
0	2015-7-1	2.0
1	2015-7-1	2.0
2	2015-7-1	1.0
3	2015-7-1	1.0
4	2015-7-1	2.0
...	...	...
119385	2017-8-30	2.0
119386	2017-8-31	3.0
119387	2017-8-31	2.0
119388	2017-8-31	2.0
119389	2017-8-29	2.0

87230 rows × 2 columns

In [74]:

```
dataNoCancel = hotel_data[hotel_data['is_canceled']==0]
```

In [75]:

```
dataNoCancel
```

Out[75]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number
0	Resort Hotel	0	342	2015	July	27
1	Resort Hotel	0	737	2015	July	27
2	Resort Hotel	0	7	2015	July	27
3	Resort Hotel	0	13	2015	July	27
4	Resort Hotel	0	14	2015	July	27
...	...	...	...	...	...	...
119385	City Hotel	0	23	2017	August	35
119386	City Hotel	0	102	2017	August	35
119387	City Hotel	0	34	2017	August	35
119388	City Hotel	0	109	2017	August	35
119389	City Hotel	0	205	2017	August	35

63221 rows × 35 columns

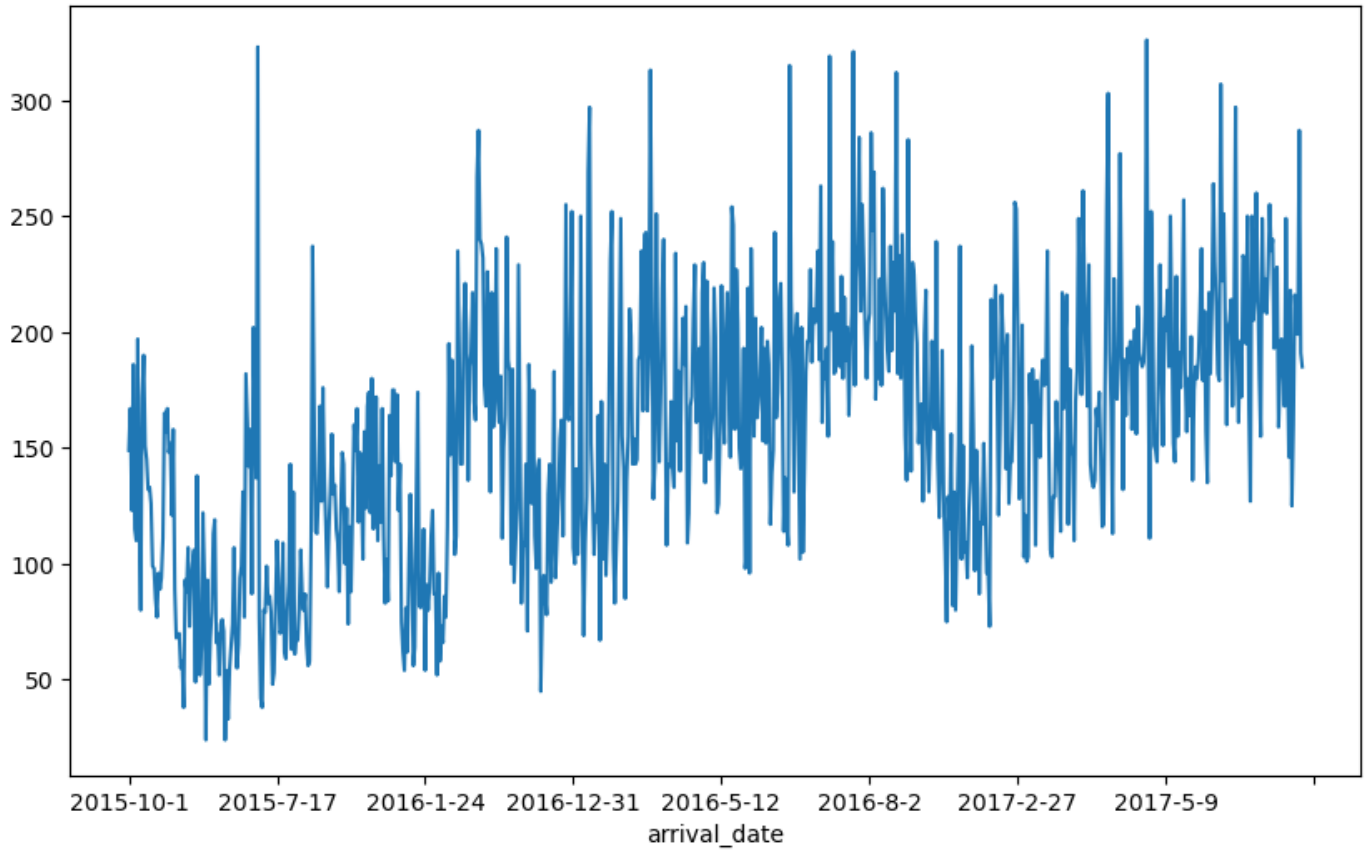
In [76]:
```
guest_arrival_series = dataNoCancel.groupby(['arrival_date'])['Total_guests'].sum()
```

In [77]:
```
guest_arrival_series
```

Out[77]:
```
arrival_date
2015-10-1     149.0
2015-10-10    167.0
2015-10-11    123.0
2015-10-12    186.0
2015-10-13    115.0
              ...
2017-8-5      205.0
2017-8-6      199.0
2017-8-7      287.0
2017-8-8      191.0
2017-8-9      185.0
Name: Total_guests, Length: 793, dtype: float64
```

In [78]:
```
guest_arrival_series.plot(figsize=(10,6))
```

Out[78]:
```
<Axes: xlabel='arrival_date'>
```



Is there any pattern ? No , there is no visible pattern in guests arrival in this line-plot as we have some un-even trend .

8.Analysing distribution of "guests arrival"

In [79]:

```
guest_arrival_series
```

Out[79]:

```
arrival_date
2015-10-1     149.0
2015-10-10    167.0
2015-10-11    123.0
2015-10-12    186.0
2015-10-13    115.0
              ...
2017-8-5      205.0
2017-8-6      199.0
2017-8-7      287.0
2017-8-8      191.0
2017-8-9      185.0
Name: Total_guests, Length: 793, dtype: float64
```

In [80]:

```
guest_arrival_series.values

## lets obtain array representation of Series so that it is easy to get our distribution
```

Out[80]:

```
array([149., 167., 123., 186., 115., 110., 197., 118.,  80., 172., 190.,
       151., 145., 132., 133., 126.,  99.,  98.,  85.,  77.,  96.,  89.,
        94., 111., 165., 157., 167., 148., 152., 121., 158.,  89.,  68.,
```

```
 69.,  70.,  55.,  58.,  38.,  93.,  88., 107.,  73.,  92., 100.,
106.,  49., 138.,  81.,  52.,  64., 122.,  95.,  24.,  93.,  48.,
 68.,  79., 113., 119.,  66.,  70.,  52.,  74.,  76.,  71.,  24.,
 54.,  33.,  55.,  64.,  74., 107.,  68.,  55.,  65.,  94.,  99.,
131.,  77., 182., 161., 142., 158.,  87., 202., 174., 137., 323.,
 77.,  42.,  38.,  80.,  79.,  99.,  83.,  86.,  79.,  48.,  53.,
 80., 110.,  93.,  70.,  71., 109.,  62.,  59.,  80.,  92., 143.,
 63., 131.,  61.,  70.,  67.,  81., 106.,  87.,  80.,  87.,  65.,
 56.,  58., 111., 237., 171., 114., 113., 131., 168., 127., 176.,
137., 116.,  90., 114., 132., 156., 130., 134., 115., 109.,  88.,
122., 148., 143., 100., 124.,  74., 116.,  88., 125., 160., 149.,
167., 118., 148., 135., 102., 157., 124., 168., 174., 122., 180.,
115., 132., 172., 110., 142., 118., 167., 112.,  83., 102.,  84.,
164., 138., 175., 175., 144., 173., 123., 143.,  75.,  62.,  54.,
 81.,  62.,  99., 130., 101.,  56.,  64., 126., 174.,  82.,  81.,
 87., 115.,  54.,  91.,  80.,  90., 112., 123.,  87.,  87.,  52.,
 96.,  58.,  73.,  66.,  86.,  77., 116., 195., 147., 188., 165.,
104., 112., 235., 189., 143., 143., 204., 221., 199., 136., 185.,
191., 217., 170., 162., 267., 287., 240., 238., 232., 177., 168.,
226., 197., 131., 217., 159., 178., 236., 170., 161., 181., 111.,
150., 164., 241., 188., 183., 100., 184.,  92., 109., 124., 229.,
156.,  83., 109., 108., 143.,  71., 186., 159., 126., 175., 112.,
 98., 141., 145.,  45.,  80.,  95.,  86.,  78., 130., 143.,  92.,
117., 183.,  94., 113., 128., 154., 162., 112., 152., 255., 164.,
162., 197., 252., 107., 100., 141., 104., 131., 250., 122.,  69.,
109., 129., 269., 297., 152., 130., 104., 121., 118., 164.,  67.,
170., 102., 143.,  95., 118., 167., 231., 252., 106.,  83., 107.,
127., 190., 249., 155., 141.,  85., 136., 155., 210., 198., 143.,
154., 143., 145., 188., 190., 235., 166., 242., 243., 166., 190.,
313., 215., 128., 142., 251., 179., 144., 174., 219., 240., 169.,
108., 143., 142., 170., 142., 133., 234., 153., 183., 140., 177.,
206., 186., 211., 109., 122., 168., 172., 206., 229., 161., 174.,
193., 148., 217., 230., 135., 222., 198., 145., 156., 164., 219.,
198., 122., 126., 155., 220., 170., 152., 201., 217., 179., 146.,
254., 246., 158., 227., 204., 150., 141., 150., 193.,  98., 104.,
219.,  96., 236., 192., 155., 206., 163., 170., 180., 202., 153.,
193., 152., 196., 186., 117., 139., 149., 243., 163., 174., 215.,
221., 183., 114., 137., 117., 108., 315., 214., 188., 131., 198.,
208., 153., 102., 202., 105., 141., 183., 196., 196., 227., 187.,
210., 204., 210., 235., 188., 263., 161., 187., 180., 193., 155.,
319., 201., 239., 182., 183., 208., 190., 185., 224., 180., 215.,
188., 202., 164., 193., 208., 321., 177., 223., 238., 284., 209.,
255., 231., 201., 180., 203., 208., 286., 244., 269., 171., 195.,
180., 223., 177., 262., 227., 207., 192., 183., 237., 192., 230.,
209., 312., 182., 233., 180., 242., 213., 155., 136., 283., 195.,
140., 230., 225., 205., 195., 152., 168., 169., 127., 199., 218.,
152., 131., 157., 196., 170., 158., 239., 145., 120., 149., 192.,
128., 105.,  75., 129., 115., 156.,  82., 131.,  80., 118., 152.,
237., 102., 151., 105., 109.,  94., 123., 136., 194., 157.,  97.,
149., 132.,  87., 118., 117., 152., 126.,  96.,  96.,  73., 214.,
180., 205., 220., 185., 121., 156., 216., 199., 190., 141., 199.,
126., 144., 144., 171., 256., 253., 203., 128., 153., 203., 103.,
121., 101., 103., 182., 161., 184., 172., 108., 179., 158., 146.,
179., 188., 177., 182., 235., 151., 106., 103., 129., 129., 170.,
152., 138., 114., 217., 167., 198., 216., 117., 184., 152., 146.,
110., 170., 183., 249., 201., 173., 261., 221., 190., 168., 229.,
143., 136., 133., 136., 167., 160., 174., 147., 116., 117., 181.,
253., 303., 192., 167., 113., 223., 176., 171., 191., 277., 212.,
132., 188., 164., 193., 187., 196., 158., 170., 201., 156., 211.,
```

```
       191., 188., 185., 187., 208., 326., 191., 111., 252., 212., 151.,
       149., 144., 187., 229., 183., 151., 206., 201., 218., 185., 250.,
       198., 152., 144., 224., 155., 191., 176., 203., 257., 184., 157.,
       180., 164., 198., 136., 175., 185., 180., 189., 210., 236., 179.,
       209., 157., 135., 217., 182., 211., 264., 230., 211., 182., 179.,
       307., 222., 251., 208., 160., 202., 205., 214., 168., 206., 297.,
       189., 161., 196., 172., 233., 231., 195., 250., 167., 127., 250.,
       205., 215., 260., 222., 181., 155., 249., 209., 223., 208., 227.,
       255., 235., 240., 193., 225., 228., 159., 180., 197., 189., 168.,
       249., 189., 146., 218., 125., 160., 216., 205., 199., 287., 191.,
       185.])
```

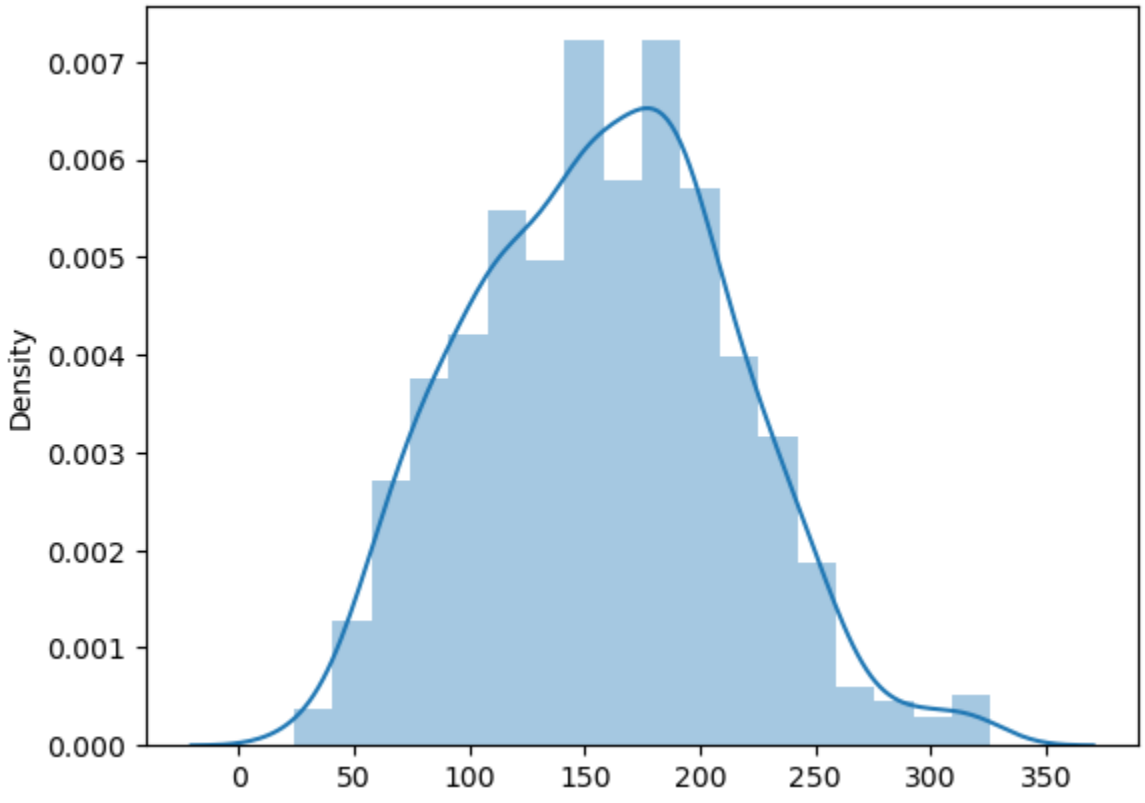
In [81]:

```
sns.distplot(guest_arrival_series.values )
```

```
C:\Users\kanis\AppData\Local\Temp\ipykernel_4324\3465509047.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


C:\Users\kanis\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
```

Out[81]:

```
<Axes: ylabel='Density'>
```



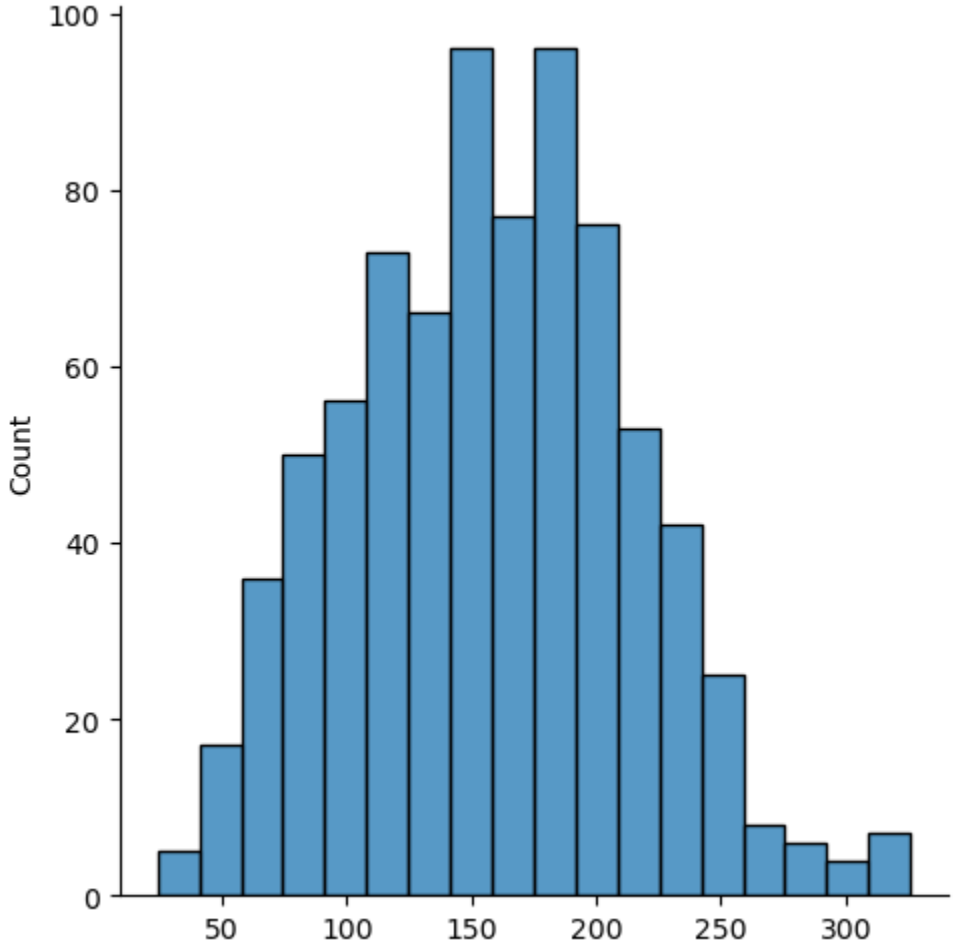
In [82]:

```
sns.displot(guest_arrival_series.values )
```

```
C:\Users\kanis\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
```

Out[82]:

```
<seaborn.axisgrid.FacetGrid at 0x18e1dc74790>
```



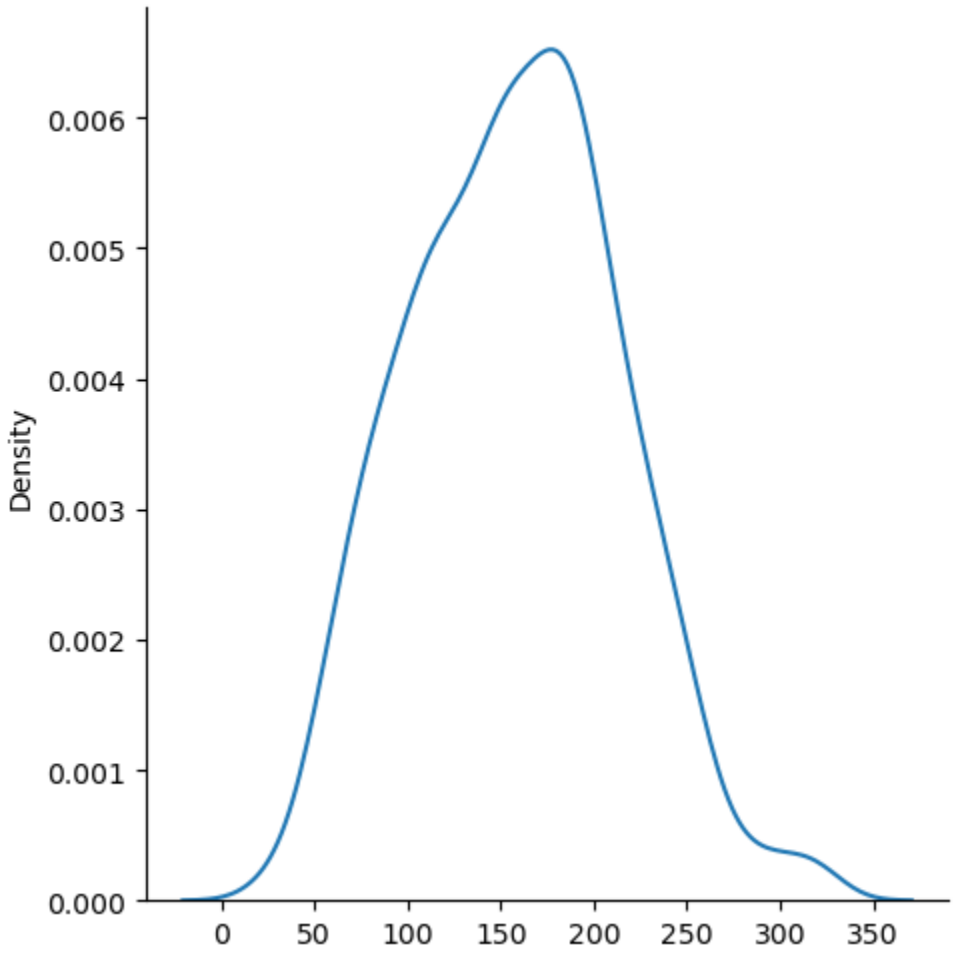
In [83]:

```
sns.displot(guest_arrival_series.values , kind='kde' )
```

```
C:\Users\kanis\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf
values to NaN before operating instead.
```

Out[83]:

```
<seaborn.axisgrid.FacetGrid at 0x18e1ecd4790>
```



In [84]:

```
np.mean(guest_arrival_series.values)
```

Out[84]:

```
157.92559899117276
```

In [85]:

```
np.median(guest_arrival_series.values)
```

Out[85]:

```
158.0
```

In [86]:

```
np.std(guest_arrival_series.values)
```

Out[86]:

```
56.48263702610786
```

The histogram with a KDE plot shows the distribution of guest arrivals, suggesting a dataset that includes arrival_date and Total_guests.

- Distribution Shape: The data appears to follow a roughly normal distribution, with a single peak and symmetric tails. This suggests that most guest arrival numbers are centered around a mean value, with fewer occurrences of extremely high or low guest numbers.
- Skewness: The distribution does not appear to be significantly skewed, suggesting a balance in the frequency of guest arrivals on either side of the mean.

- Kurtosis: The height and sharpness of the peak, along with the tails of the distribution, suggest moderate kurtosis. This indicates that the data has a pronounced peak but is not excessively peaked or flat.