Hotel Booking Data Analysis — Jupyter Notebook Overview

This Jupyter Notebook explores a rich Hotel Booking Demand dataset, uncovering patterns behind booking behaviors, cancellations, customer types, and pricing trends. With a wide range of features—from arrival details to special requests—this dataset provides a great opportunity to practice data cleaning, visualization, and predictive insights.

Table: Dataset Description — Hotel Booking Features

Column / Feature	Description
hotel	Type of hotel (City Hotel / Resort Hotel)
is_canceled	Indicates whether the booking was cancelled
lead_time	Number of days between booking date and arrival
arrival_date_year / month / week_number / day_of_month	Detailed components of the guest's arrival date
stays_in_weekend_nights / stays_in_week_nights	Number of weekend and weekday nights stayed
adults, children, babies	Number of guests in each category
meal	Selected meal type
country	Country of origin of the guest
market_segment & distribution_channel	Booking source and channel used
is_repeated_guest	Indicates if the guest has booked before
previous_cancellations / previous_bookings_not_canceled	Guest's past booking and cancellation behavior
reserved_room_type / assigned_room_type	Room type initially reserved vs. type assigned at check-in
booking_changes	Total modifications made to the booking
deposit_type	Whether any deposit was required
agent & company	IDs of the booking agent or company
days_in_waiting_list	Number of days the booking was on the waiting list
customer_type	Type of customer (e.g., transient, group, etc.)
adr	Average Daily Rate (price per room per night)
total_of_special_requests	Number of special requests made by the guest
reservation_status & reservation_status_date	Final booking status and the date of that status

Data Analysis Lifecycle

Stage	Description
S1: Understanding Use Case	Define the business problem, objectives, and expected outcomes. Clearly understand what insights are needed and why the analysis is being performed.
S2: Extract Data	Collect or import the required dataset from sources such as CSV files, databases, APIs, or other data repositories.
S3: Data Cleaning	Prepare the dataset for analysis to ensure accuracy and reliability. Includes: • **Remove duplicate rows** • **Remove irrelevant rows** • **Fix errors** (typos, inconsistent categories) • **Check missing values** and handle them appropriately • **Validate data types** and correct incorrect types • **Deal with outliers** using statistical methods or domain knowledge

S-1 : Understanding the Bussiness Problem

Define the Bussiness Problem :

- ▪ Perform descriptive Analysis
- ▪ Where do the guests come from?

- ▪ Is any difference between assigned and reservedd room type or not?
- ▪ Which Market segments has highest booking ?
- ▪ Analysing Avg.price per night (ADR) of various room-types for all the market segment .
- ▪ Total guest Arrival on each day ?
- ▪ Analysing distribution of guest_arrival

S-2 : Extract Data or Data collection :

- Collect or import the required dataset from sources such as CSV files, databases, APIs,or other data repositories.

In [1]:
```
# import required Library to perform All step
# Data cleaning -  numpy , pandas most common Library to  used to data extraction ,data preprocessing(Data cleaning)
import numpy as np                   # import numpy and aliasing as np
import pandas as pd                  #import pandas and aliasing as pd
# Dta visualization -  matplotlib , seaborn are most common Library to use data visualization in static format
import matplotlib.pyplot as plt      #import matplotlib and aliasing as plt
import seaborn as sns                #import seaborn and Aliasing as sns
# INteractive visualization -  For Interactive Visualization most common libarary plotly
import plotly.express as px      # import plotly   and aliasing as px
```

Note - I Use csv file which I also added in github repository

In [2]:
```
# Read data and store in df variable
df = pd.read_csv('hotel_bookings (1).csv')
```

S3: Data Cleaning: Data preparation for Data Analysis

In [3]:
```
# show only first five row
df.head()
```

Out[3]:

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

5 rows × 32 columns

In [4]:
```
# show total row and column of data
print(f'Total rows of Dataset     :  {df.shape[0]}')
print(f'Total columns of Dataset  :  {df.shape[1]}')
```

```
Total rows of Dataset     :  119390
Total columns of Dataset  :  32
```

In [5]:
```
# Show the name of columns of dataset
print("                    Name of columns")
print("            -----------------------------------")
print(df.columns)
```

```
                    Name of columns
            -----------------------------------
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 [6]:
```
# let,s check the unwanted entries -  if (babies,adults,children are 0 - Thats mean not booking its consider as unwanted Entr
Unwanted_entry= (df['adults']==0) & (df['children']==0) & (df['babies']==0)
```

Note - if adult,children,babies = 0 , That's mean no one come here.

In [7]:
```
# Total number of  Unwanted_entry
print(f'Total number of  Unwanted_entry: {df[Unwanted_entry].shape[0]}')
```

```
Total number of  Unwanted_entry: 180
```

In [8]:
```
# Total number of  rows except unwanted data
print(f'Total number of  _entry: {df[~Unwanted_entry].shape[0]}')
```

```
Total number of  _entry: 119210
```

In [9]:
```
df = df[~Unwanted_entry] #  (~ exclude ) -   show data exclude Unwanted entry . and store in df dataframe
```

In [10]:
```
#chaeck duplicate rows
print(f'Total no of duplicate record : {df.duplicated().sum()}')
```

```
Total no of duplicate record : 31980
```

In [11]:
```
# remove duplicated data
df = df.drop_duplicates()
```

In [12]:
```
#check again duplicated value for confiramtion
print(f'Total Duplicates value:    {df.duplicated().sum()}')
```

```
Total Duplicates value:    0
```

Bussiness Problem : - performing a descriptive Analysis

In [13]:
```
# descriptive Analysis
df.describe().T
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max
is_canceled	87230.0	0.275238	0.446637	0.00	0.00	0.0	1.0	1.0
lead_time	87230.0	79.971019	86.058683	0.00	11.00	49.0	125.0	737.0
arrival_date_year	87230.0	2016.210352	0.686064	2015.00	2016.00	2016.0	2017.0	2017.0
arrival_date_week_number	87230.0	26.835091	13.669216	1.00	16.00	27.0	37.0	53.0
arrival_date_day_of_month	87230.0	15.815832	8.835545	1.00	8.00	16.0	23.0	31.0
stays_in_weekend_nights	87230.0	1.004609	1.027408	0.00	0.00	1.0	2.0	19.0
stays_in_week_nights	87230.0	2.623925	2.039830	0.00	1.00	2.0	4.0	50.0
adults	87230.0	1.879365	0.621724	0.00	2.00	2.0	2.0	55.0
children	87226.0	0.138904	0.456274	0.00	0.00	0.0	0.0	10.0
babies	87230.0	0.010845	0.113704	0.00	0.00	0.0	0.0	10.0
is_repeated_guest	87230.0	0.038565	0.192556	0.00	0.00	0.0	0.0	1.0
previous_cancellations	87230.0	0.030402	0.369344	0.00	0.00	0.0	0.0	26.0
previous_bookings_not_canceled	87230.0	0.184054	1.733033	0.00	0.00	0.0	0.0	72.0
booking_changes	87230.0	0.268497	0.710633	0.00	0.00	0.0	0.0	18.0
agent	75089.0	94.200429	113.205767	1.00	9.00	14.0	240.0	535.0
company	5237.0	182.970594	130.486644	6.00	47.00	169.0	263.0	543.0
days_in_waiting_list	87230.0	0.746291	10.001001	0.00	0.00	0.0	0.0	391.0
adr	87230.0	106.518031	54.891227	-6.38	72.25	98.2	134.1	5400.0
required_car_parking_spaces	87230.0	0.084306	0.281659	0.00	0.00	0.0	0.0	8.0
total_of_special_requests	87230.0	0.698934	0.832051	0.00	0.00	0.0	1.0	5.0

Insight - 1 - MOst of The data collection (year) - 2017 2 - Avg of total data collection (year) - 2015 3 - Standard deviation — most bookings are close to 2 adults, with relatively low variation. 4 - Most of the hotel booking via Adult (2 adults)

In [14]:

```
# Show Information about Data column datatype and null value , distribution of Datatype ,  Memory usage -   show approx not Ex
df.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
```

Insight - memory usage: 22.0+ MB , that means approx not excatly . if you want exact memory usage ` run df.info(memory_usage='deep')`

In [15]:

```
df.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: 69.2 MB
```

Summary of data -

- Total number of column (dtype - Object) : 12
- Total number of column (dtype - float) : 4
- Total number of column (dtype - int) : 16
- Exact memory usage : 69.2 MB

Problem Statement :- where do the Guests come from ?

step to solve the problem

- - filter data who not canceled the hotel booking.
- - count separate data value of each country using groupby fuction and store in a variable.
- - To show on map use plotly library also plot top 10 country guest come from

1 - filter data who not canceled the hotel booking

In [16]:
```python
# filter data who is not_canceled  booking and store in  not_cancelled_hotel
not_cancelled_hotel = df[df['is_canceled']==0]
```

2- count data value for each country

In [17]:
```python
# count value of each country and convert it dataframe using reset_index function
country_wise_data  = not_cancelled_hotel['country'].value_counts().reset_index()
```

In [18]:
```python
# this is dataframe
country_wise_data.info()
```

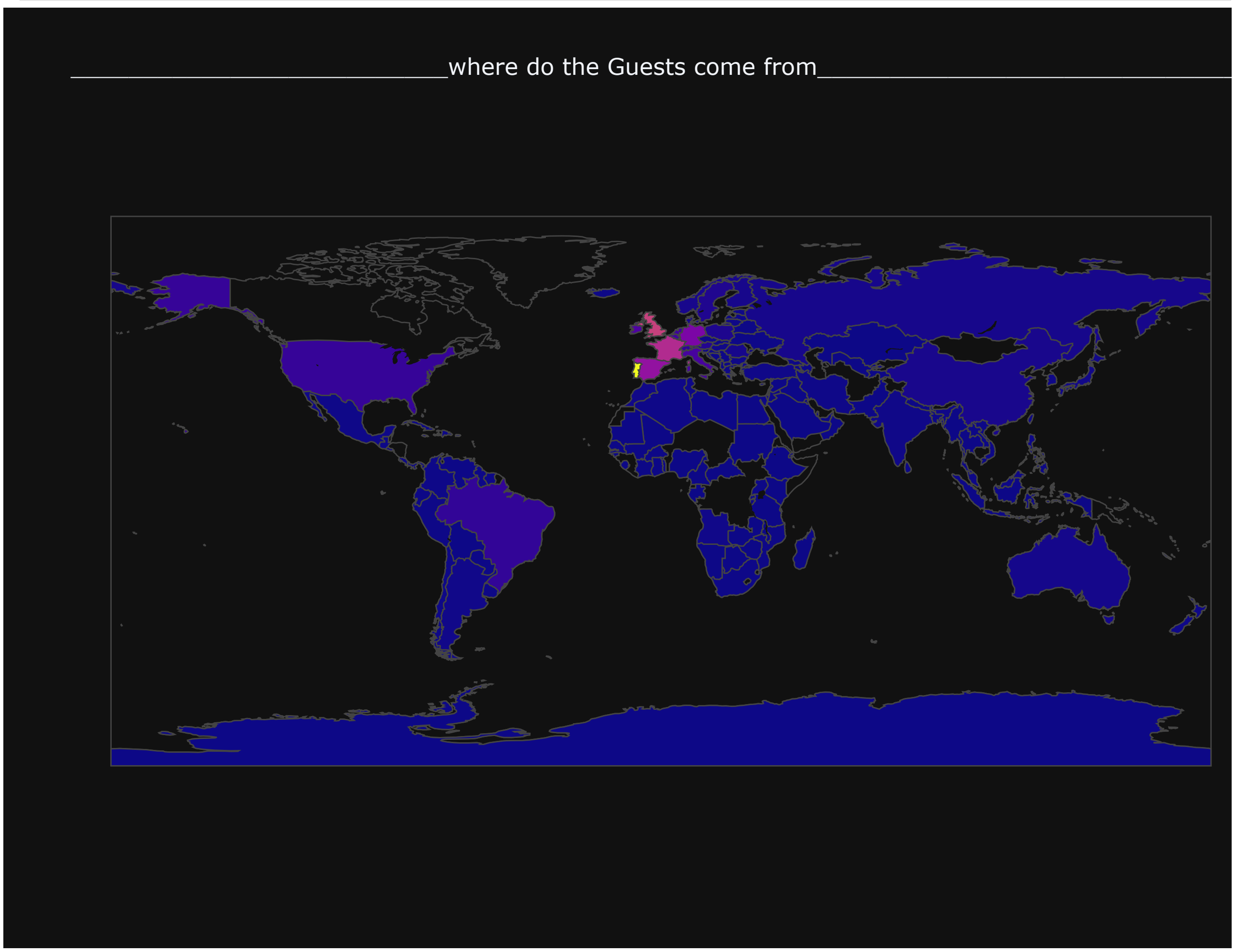
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165 entries, 0 to 164
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   country  165 non-null    object
 1   count    165 non-null    int64
dtypes: int64(1), object(1)
memory usage: 2.7+ KB
```

To show on map use plotly library also plot top 10 country guest come from ?

In [19]:
```
#!pip install chart-studio
```

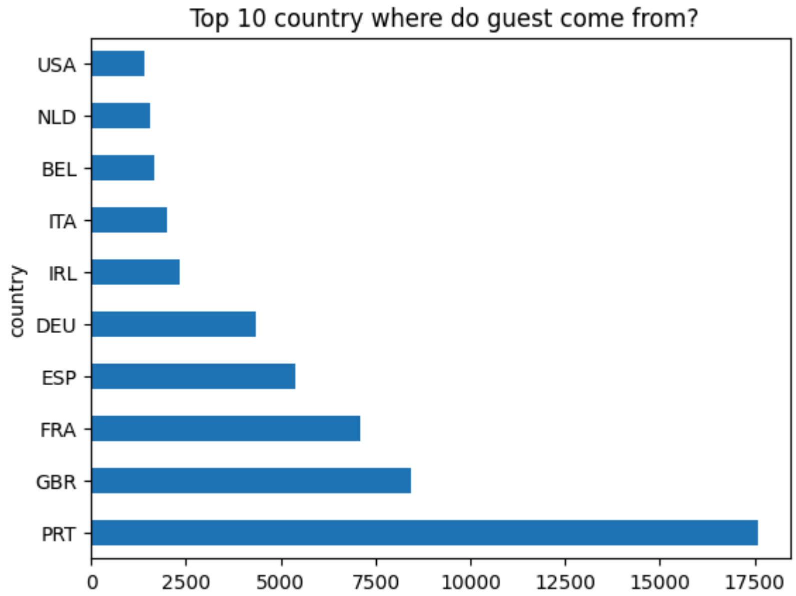
In [20]:
```
# import some required library for ploting a map using plotly
import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs ,init_notebook_mode , plot ,iplot
init_notebook_mode(connected=True)
# Make sure when run it stable Internet connection -  becouse it use PLotly API
```

In [21]:
```
# plot a map plot function name -choropleth ,  and store in map_guest variable
map_guest =  px.choropleth(data_frame= country_wise_data,locations= country_wise_data['country']
              ,color=country_wise_data['count'],
              title="__________________________where do the Guests come from______________________________",template='plotly_(
map_guest.update_layout(width=1000, height=700)# update layout
map_guest.show()# to show a plot
```



In [22]:
```
# Top 10 country - Not Interactive plot  -
plt.title(" Top 10 country where do guest come from?")
not_cancelled_hotel['country'].value_counts().head(10).plot(kind='barh')
```

Out[22]:
```
<Axes: title={'center': ' Top 10 country where do guest come from?'}, ylabel='country'>
```



Problem Statement : Is any difference between assigned and reserved room type or not?

step to solve problem : we use crosstab and create a table for better understaning we normalize dataset , then round off 2 decimal and for convert percentage -multiply 100 create a pie chart for visualization

In [23]:
```
#In this code only use Normalize but not round off value with 2 decimal places , Its Interpretable but not more Understadable
pd.crosstab(index=df['reserved_room_type'],columns=df['assigned_room_type'],margins=True,normalize='index')
```

Out[23]:

assigned_room_type	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**K**	**L**
reserved_room_type											
A	0.812425	0.015806	0.022202	0.113438	0.018322	0.006910	0.003119	0.001666	0.003632	0.002481	0.000000
B	0.106426	0.875502	0.000000	0.005020	0.002008	0.002008	0.008032	0.000000	0.000000	0.001004	0.000000
C	0.005470	0.002188	0.947484	0.006565	0.004376	0.002188	0.010941	0.009847	0.010941	0.000000	0.000000
D	0.016977	0.001554	0.001842	0.919602	0.037811	0.011453	0.004719	0.000518	0.003856	0.001669	0.000000
E	0.002485	0.000331	0.000994	0.003645	0.904241	0.063453	0.016070	0.000663	0.006627	0.001491	0.000000
F	0.002128	0.004965	0.000000	0.001418	0.010993	0.934752	0.040071	0.001064	0.003546	0.001064	0.000000
G	0.002439	0.000488	0.000976	0.000000	0.001951	0.006829	0.975122	0.003415	0.007317	0.001463	0.000000
H	0.000000	0.000000	0.000000	0.001678	0.000000	0.000000	0.016779	0.971477	0.010067	0.000000	0.000000
L	0.166667	0.166667	0.166667	0.000000	0.000000	0.166667	0.000000	0.166667	0.000000	0.000000	0.166667
All	0.530586	0.020761	0.024762	0.257010	0.082426	0.041580	0.028603	0.008094	0.004047	0.002121	0.000011

In [24]:
```
#In this code snippet we use normalize to better decision making and Easily Interpretable
pd.crosstab(index=df['reserved_room_type'],columns=df['assigned_room_type'],margins=True,normalize='index').round(2)*100
```

Out[24]:

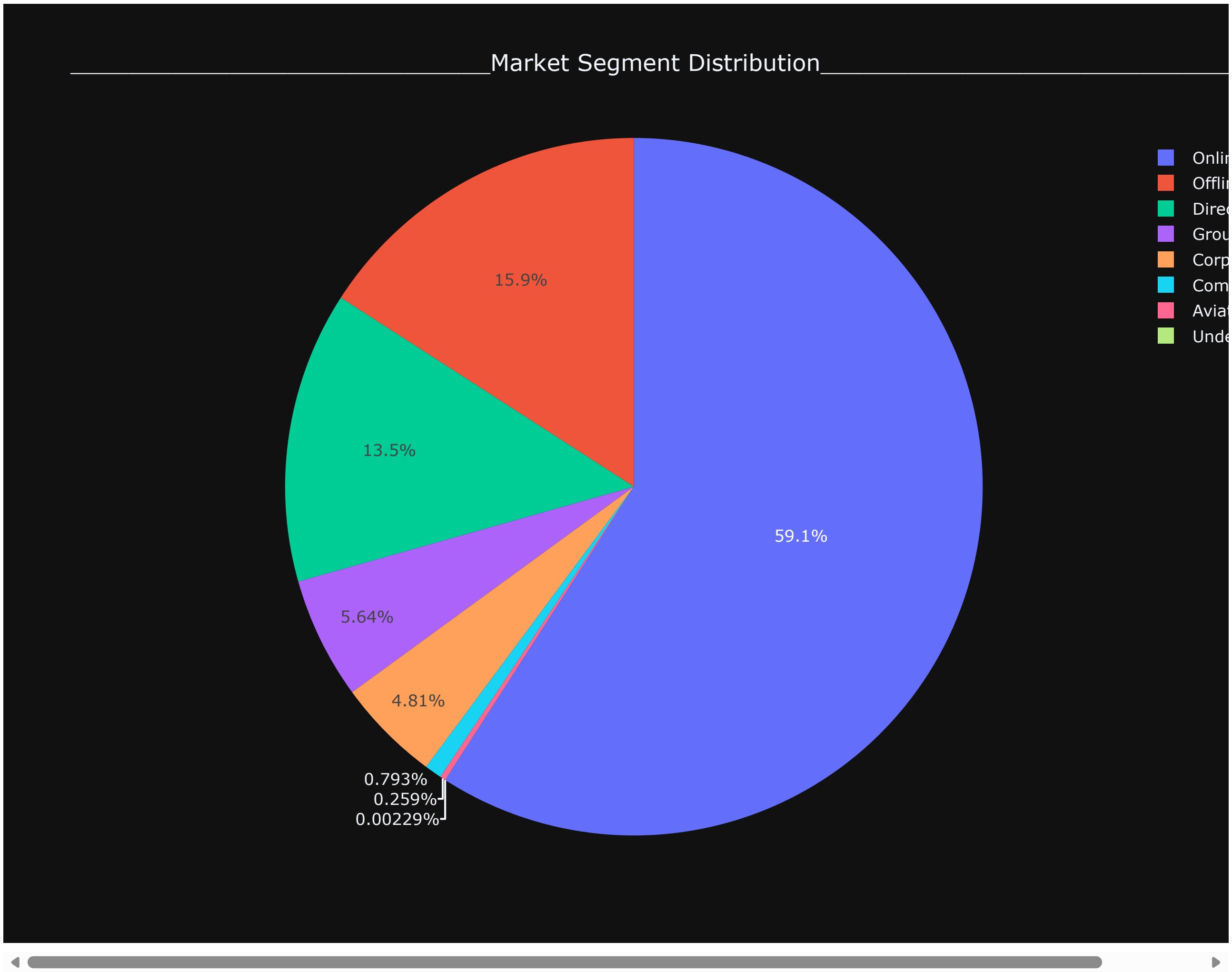
assigned_room_type reserved_room_type	A	B	C	D	E	F	G	H	I	K	L
A	81.0	2.0	2.0	11.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0
B	11.0	88.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
C	1.0	0.0	95.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
D	2.0	0.0	0.0	92.0	4.0	1.0	0.0	0.0	0.0	0.0	0.0
E	0.0	0.0	0.0	0.0	90.0	6.0	2.0	0.0	1.0	0.0	0.0
F	0.0	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

Which Market segments has highest booking ?

In [25]:
```
df['market_segment'].value_counts()
```

Out[25]:
```
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 [26]:
```
fig = px.pie(
    names=df['market_segment'],
    title="_____________________________Market Segment Distribution__________________________________",template='plotly_dark'
fig.update_layout(width=1000, height=700)
fig.show()
```



Insighgt - Most of Payment approx 59% from Online TA

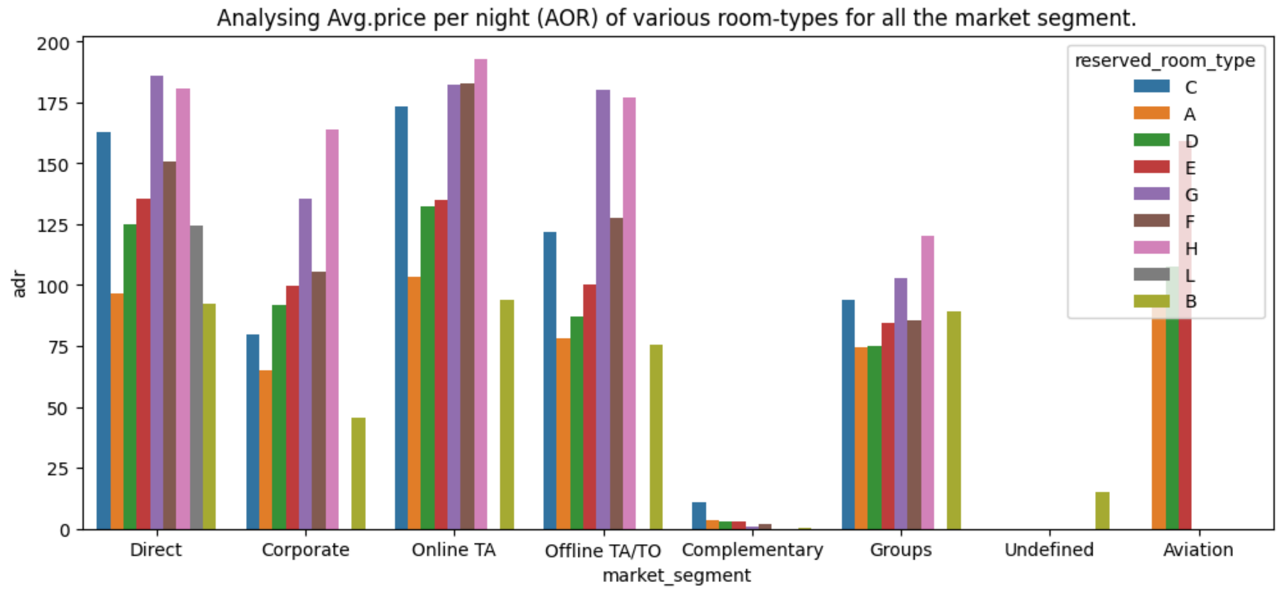
Analysing Avg.price per night (ADR) of various room-types for all the market segment .

In [27]:
```
df["adr"].value_counts()
```

Out[27]:
```
adr
0.00      1643
75.00     1320
65.00     1260
48.00      878
85.00      858
          ...
174.20       1
78.61        1
99.72        1
174.16       1
194.60       1
Name: count, Length: 8866, dtype: int64
```

In [28]:
```
# Avg daily rate - adr
plt.figure(figsize=(12,5))
fig = sns.barplot(data=df,x="market_segment",y="adr",hue='reserved_room_type',errorbar=None)
fig.set_title("Analysing Avg.price per night (AOR) of various room-types for all the market segment.")
```

Out[28]:
```
Text(0.5, 1.0, 'Analysing Avg.price per night (AOR) of various room-types for all the market segment.')
```



Insight : Market Segments: Includes Direct, Corporate, Online TA, Offline TA/TO, Complementary, Groups, Undefined, and Aviation. Room Types: Represented by labels like C, A, D, E, G, F, H, L, B — each shown in different colors. Price Range: ADR values span from 0 to 200, indicating significant variation in pricing. 0.00 → 1643 bookings Means 1,643 records had an average daily rate of 0. This could indicate complimentary stays, errors, or special cases (like promotional bookings). 75.00 → 1320 bookings 1,320 bookings had a nightly rate of exactly 75. 65.00 → 1260 bookings 1,260 bookings had a nightly rate of 65. 48.00 → 878 bookings 878 bookings had a nightly rate of 48. 85.00 → 858 bookings 858 bookings had a nightly rate of 85. Rare values (like 174.20, 78.61, 99.72, 194.60)

Problem statements : Total guest Arrival on each day ?

Step to solve problem : 1 - assign month as month number like - January:1 , February: 2. 2 - Add year,month , date like : 2017-07-22 3 - create a new column name total guest where i add adults,children and babies . 4 - Filtered Data who not cancelled booking. 5 - Groped arrival_date based on total of guest each date. 6 - plot a line plot to show trend

1 - assign month as month number like - January:1 , February: 2.

In [29]:
```
# find  all month
df['arrival_date_month'].unique()
```

Out[29]:
```
array(['July', 'August', 'September', 'October', 'November', 'December',
       'January', 'February', 'March', 'April', 'May', 'June'],
      dtype=object)
```

In [30]:
```
# create a dictionary and assign month number to month name
dict_month = {'July':7, 'August':8, 'September':9, 'October':10, 'November':11, 'December':12,
       'January':1, 'February':2, 'March':3, 'April':4, 'May':5, 'June':6}
print("code excuted Secessfully")
```

```
code excuted Secessfully
```

In [31]:
```
# creating a new column name is  arrival_date_month_index , map month number
df['arrival_date_month_index'] = df['arrival_date_month'].map(dict_month)
print("code excuted Secessfully")
```

```
code excuted Secessfully
```

2 - create a new column in df dataframe name - Arrival_date . for Add year,month , date like : 2017-07-22

In [32]:
```
# Adding year , month and data , But not work date format it works only str , so I convert str datatype
df['arrival_date'] = df['arrival_date_year'].astype(str) + '-' + df['arrival_date_month_index'].astype(str) + '-' + df['arriv
print("code excuted Secessfully")
```

```
code excuted Secessfully
```

3 - create a new column name total guest where i add adults,children and babies

In [33]:
```
df['total_guest'] = df['adults'] + df['children'] +df['babies']
print("code excuted Secessfully")
```

```
code excuted Secessfully
```

4 - Filtered Data who not cancelled booking

In [34]:
```
data_not_cancelled = df[df['is_canceled']==0]
print("code excuted Secessfully")
```

```
code excuted Secessfully
```

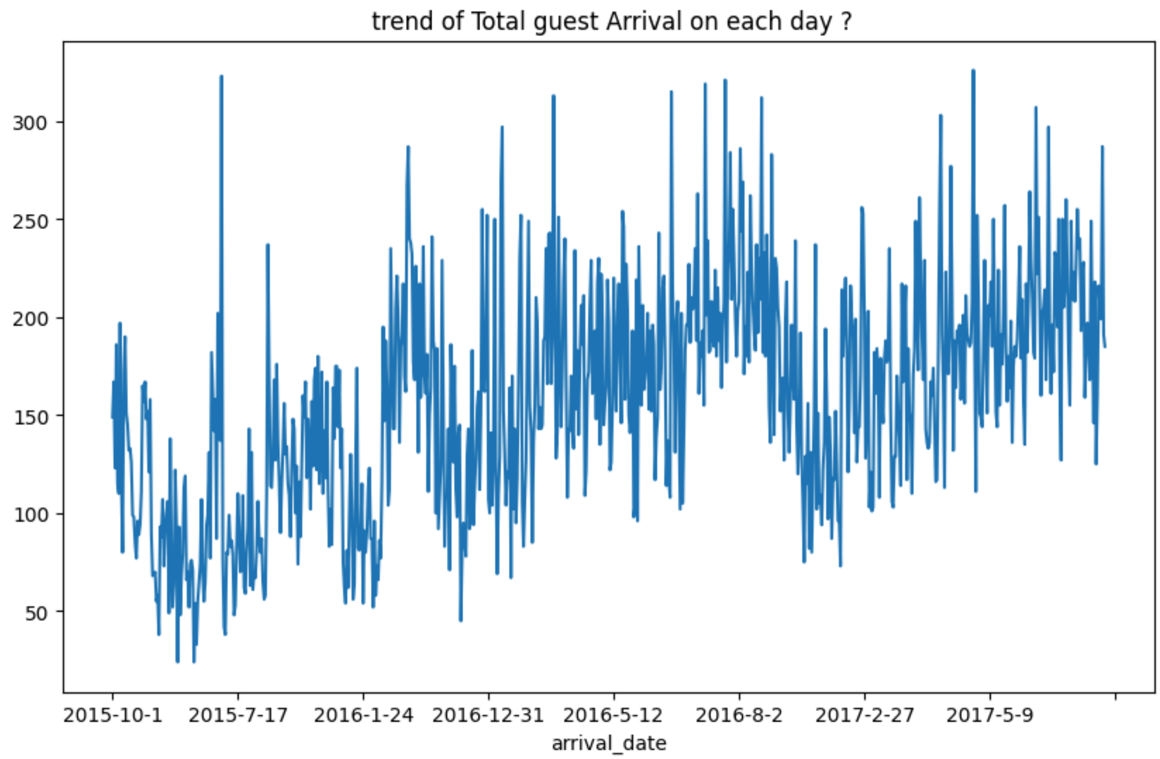
5- Groped arrival_date based on total of guest each date

In [35]:
```
guest_arrival = data_not_cancelled.groupby(['arrival_date'])['total_guest'].sum()
```

6 - plot a line plot to show trend

In [36]:
```
plt.title(" trend of Total guest Arrival on each day ?")
guest_arrival.plot(figsize=(10,6))
```

Out[36]:
```
<Axes: title={'center': ' trend of Total guest Arrival on each day ?'}, xlabel='arrival_date'>
```



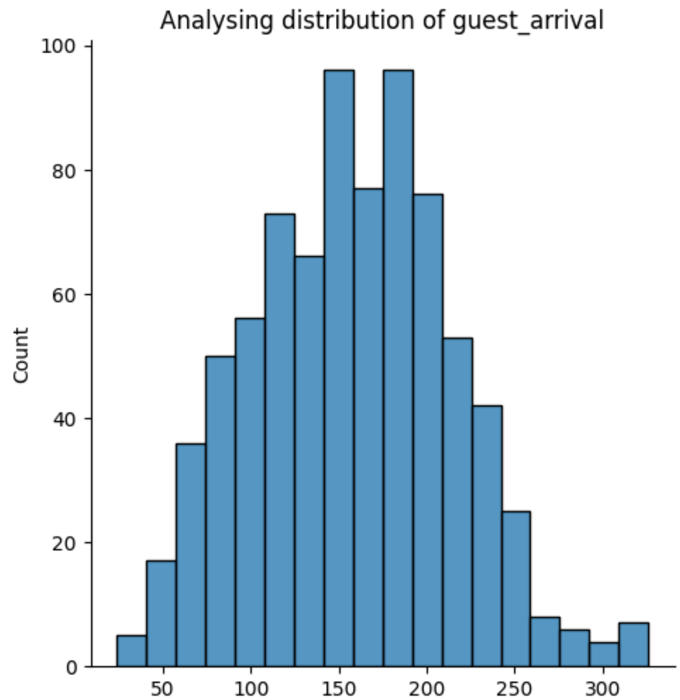
Insight - The graph shows regular ups and downs, indicating seasonal patterns in guest arrivals. Peaks may correspond to holiday seasons, festivals, or tourism cycles, while dips suggest off-peak periods. Some days show spikes above 300 guests, which could be due to: Special events or conferences Group bookings Weekend or holiday surges

In [ ]:
```
```

Problem statement :Analysing distribution of guest_arrival

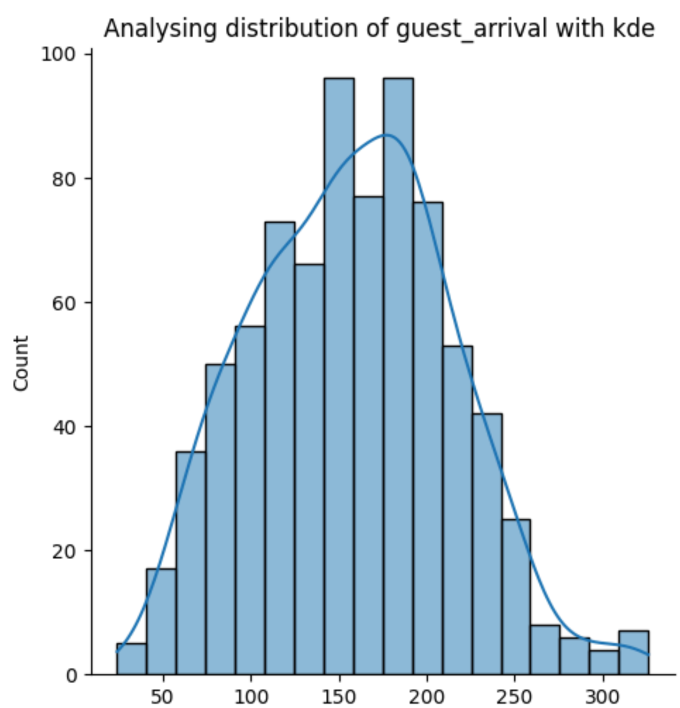
In [37]:
```
sns.displot(guest_arrival.values)
plt.title("Analysing distribution of guest_arrival")
```

Out[37]:
```
Text(0.5, 1.0, 'Analysing distribution of guest_arrival')
```



In [39]:
```
sns.displot(guest_arrival.values,kde=True)
plt.title("Analysing distribution of guest_arrival with kde ")
```

Out[39]: `Text(0.5, 1.0, 'Analysing distribution of guest_arrival with kde ')`



*Thankyou for reading my notebook I hope this notebook will help you.

if any suggestion please comment on comment section