Hotel Booking EDA

Below are the libraries we used :

We are importing pandas for importing data and pandas has functions for analyzing, cleaning, exploring, and manipulating data.

NumPy is a Python library used for working with arrays.

Seaborn is a Python data visualization library based on matplotlib.

matplotlib.pyplot makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

We are importing warnings to ignore the warnings

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

We are using Hotel Bookings data.

We downloaded the data from kaggle link : https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand (https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand) ¶

We stored data in df data frame.

We used pd.read_csv and read the hotel bookings file.

In [2]:

```
df = pd.read_csv('hotel_bookings.csv')
```

df.head is used to to get top 5 rows from the data.

We used head function to check whether data is properly loaded or not.

We get almost columns from this function

In [3]:

```
1 df.head()
```

Out[3]:

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

5 rows × 32 columns

df.tail is used to to get last 5 rows from the data.

We used last function to check whether data is properly loaded or not.

In [4]:

```
1 df.tail()
```

Out[4]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_wee
119385	City Hotel	0	23	2017	August	
119386	City Hotel	0	102	2017	August	
119387	City Hotel	0	34	2017	August	
119388	City Hotel	0	109	2017	August	
119389	City Hotel	0	205	2017	August	

5 rows × 32 columns

We are using df.shape here to tell how many rows and columns are there in

the dataset

In [5]:

```
1 df.shape
```

Out[5]:

```
(119390, 32)
```

We used info to know what type of data is there.

We can see that there are 16 integer type, 12 object type and 4 float type columns.

We can see that country, agent, children and company have null values too

In [6]:

```
1 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
```

Now we will see how many null values are present in each column

df.isnull.sum() tells about the total number of null values in the column

Country - 488 , agent - 16340 , company - 112593 are the null values present in the dataset

In [7]:

```
1 df.isnull().sum()
```

Out[7]:

```
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
```

Country is object type dataset.

country has 488 null values.

we will find out the mod of country column

In [8]:

```
1 df['country'].mode()
```

Out[8]:

```
0    PRT
Name: country, dtype: object
```

We can see that country has PRT as most repeated country in the data.

PRT is represented as Portugal

In [9]:

```
df['country'] = df['country'].fillna("PRT")
```

Agent is Integer type data.

We will see is there are any outliers in agent column

We will calculate the IQR and Q1 and Q3 then find the outlier

If there are no outlier in the agent column then we will replace the null values with mean

If there are outliers present in the agent column then we will replace null values with median

In [10]:

```
Q1 = df['agent'].quantile(.25)
Q3 = df['agent'].quantile(.75)

IQR = Q3 - Q1
IQR
```

Out[10]:

```
220.0
```

We will now see left value (VL) and and right valu (VR)

In [11]:

```
VR = Q1-(1.5*IQR)
VL = Q3 + (1.5*IQR)

VR,VL
```

Out[11]:

```
(-321.0, 559.0)
```

We will now see descriptive stats of the agent column

In [12]:

```
1 df['agent'].describe()
```

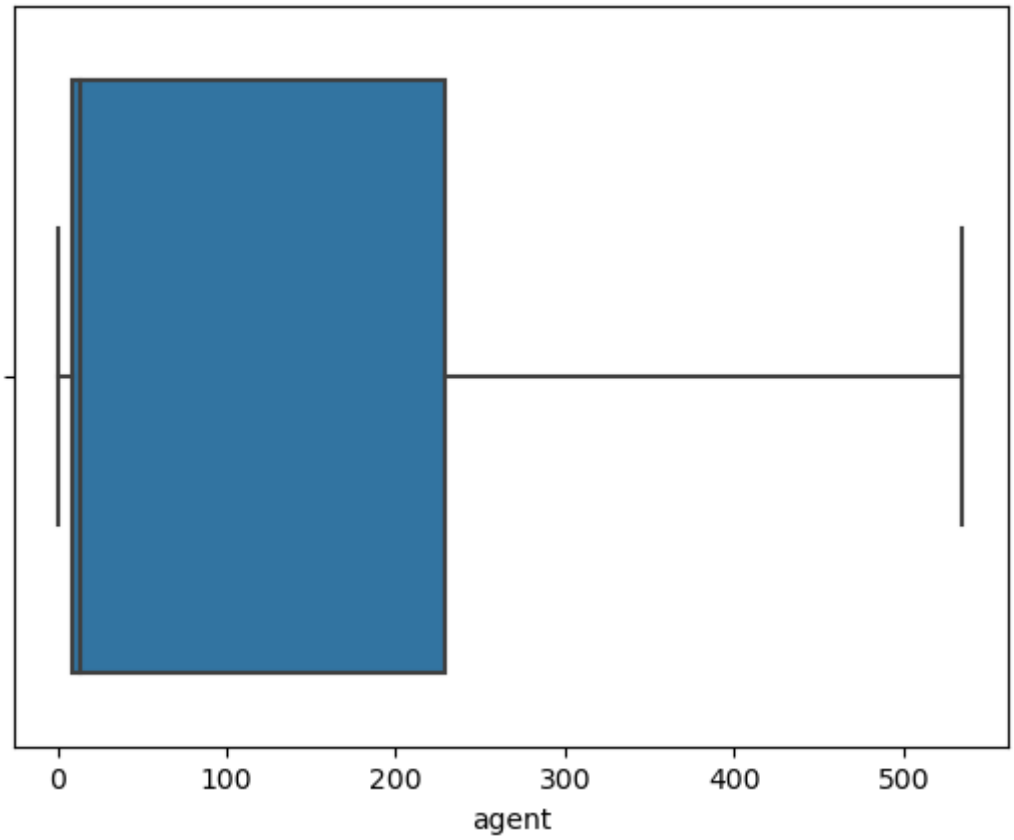
Out[12]:

```
count    103050.000000
mean         86.693382
std         110.774548
min           1.000000
25%           9.000000
50%          14.000000
75%         229.000000
max         535.000000
Name: agent, dtype: float64
```

We are gonna boxplot the agent column data below to see the outliers

In [13]:

```
1 sns.boxplot(df['agent'])
2 plt.show()
```



We are gonna see the number of values which are greater than 535 in the below cells

If there are no outliers which we can see in boxplot and below cell too then we will replace it by mean

In [14]:

```
df1 = df['agent']> 535
df1.value_counts()
```

Out[14]:

```
False    119390
Name: agent, dtype: int64
```

As there are no outliers present here we will replace the null values with mean

In [15]:

```
df['agent'] = df['agent'].replace(np.nan,df['agent'].mean())
```

Now we will work on Company coulmn

Country column has 112593 null values

So if we even replace the null values and change values accordingly biasness will increase

It is better to drop the company.

In [16]:

```
df = df.drop('company',axis = 1)
```

Now we will work on children column

We will see is there are any outliers in agent column

We will calculate the IQR and Q1 and Q3 then find the outlier

If there are no outlier in the agent column then we will replace the null values with mean

If there are outliers present in the agent column then we will replace null values with median

In [17]:

```
Q11 = df['children'].quantile(.25)
Q33 = df['children'].quantile(.75)

IQR = Q3 - Q1
IQR
```

Out[17]:

```
220.0
```

In [18]:

```
VR = Q11-(1.5*IQR)
VL = Q33 + (1.5*IQR)

VR,VL
```

Out[18]:

```
(-330.0, 330.0)
```

In [19]:

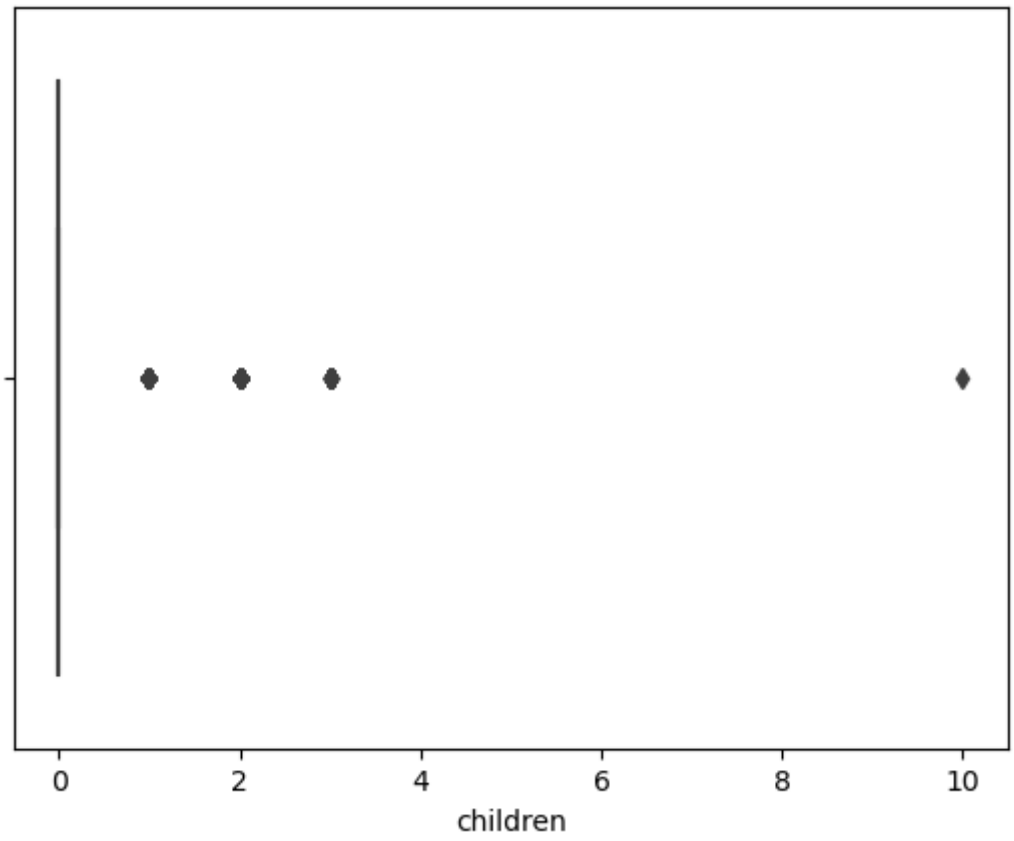
```
df['children'].describe()
```

Out[19]:

```
count    119386.000000
mean          0.103890
std           0.398561
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          10.000000
Name: children, dtype: float64
```

In [20]:

```
sns.boxplot(df['children'])
plt.show()
```



No outliers in this children column

So we will replace the null value with mean

In [21]:

```
df['children'] = df['children'].replace(np.nan,df['children'].mean())
```

Now we will again see the null values in the dataset

In [22]:

```
1 df.isnull().sum()
```

Out[22]:

```
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                        0
babies                          0
meal                            0
country                         0
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                           0
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
```

In [23]:

```
1 df['total_stay'] = df['stays_in_weekend_nights'] + df['stays_in_week_nights']
```

In [24]:

```
df['total_stay'].value_counts(sort = True)
```

Out[24]:

```
2     27643
3     27076
1     21020
4     17383
7      8655
5      7784
6      3857
8      1161
10     1139
14      916
9       841
0       715
11      396
12      223
13      142
15       75
21       71
16       40
25       37
18       35
28       35
19       22
17       20
29       14
20       14
22       14
30       13
23        8
24        6
26        6
27        5
35        5
42        4
33        3
56        2
34        1
57        1
49        1
48        1
69        1
38        1
45        1
60        1
46        1
43        1
Name: total_stay, dtype: int64
```

In [25]:

```
1 df['is_canceled'].value_counts()
```

Out[25]:

```
0    75166
1    44224
Name: is_canceled, dtype: int64
```

In [26]:

```
1 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                        119390 non-null  float64
 11  babies                          119390 non-null  int64  
 12  meal                            119390 non-null  object 
 13  country                         119390 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                           119390 non-null  float64
 24  days_in_waiting_list            119390 non-null  int64  
 25  customer_type                   119390 non-null  object 
 26  adr                             119390 non-null  float64
 27  required_car_parking_spaces     119390 non-null  int64  
 28  total_of_special_requests       119390 non-null  int64  
 29  reservation_status              119390 non-null  object 
 30  reservation_status_date         119390 non-null  object 
 31  total_stay                      119390 non-null  int64  
dtypes: float64(3), int64(17), object(12)
memory usage: 29.1+ MB
```

In [27]:

```
dff = df[['hotel', 'total_of_special_requests']]

total_special_requests = dff.groupby('hotel')['total_of_special_requests'].sum()
```

In [ ]:

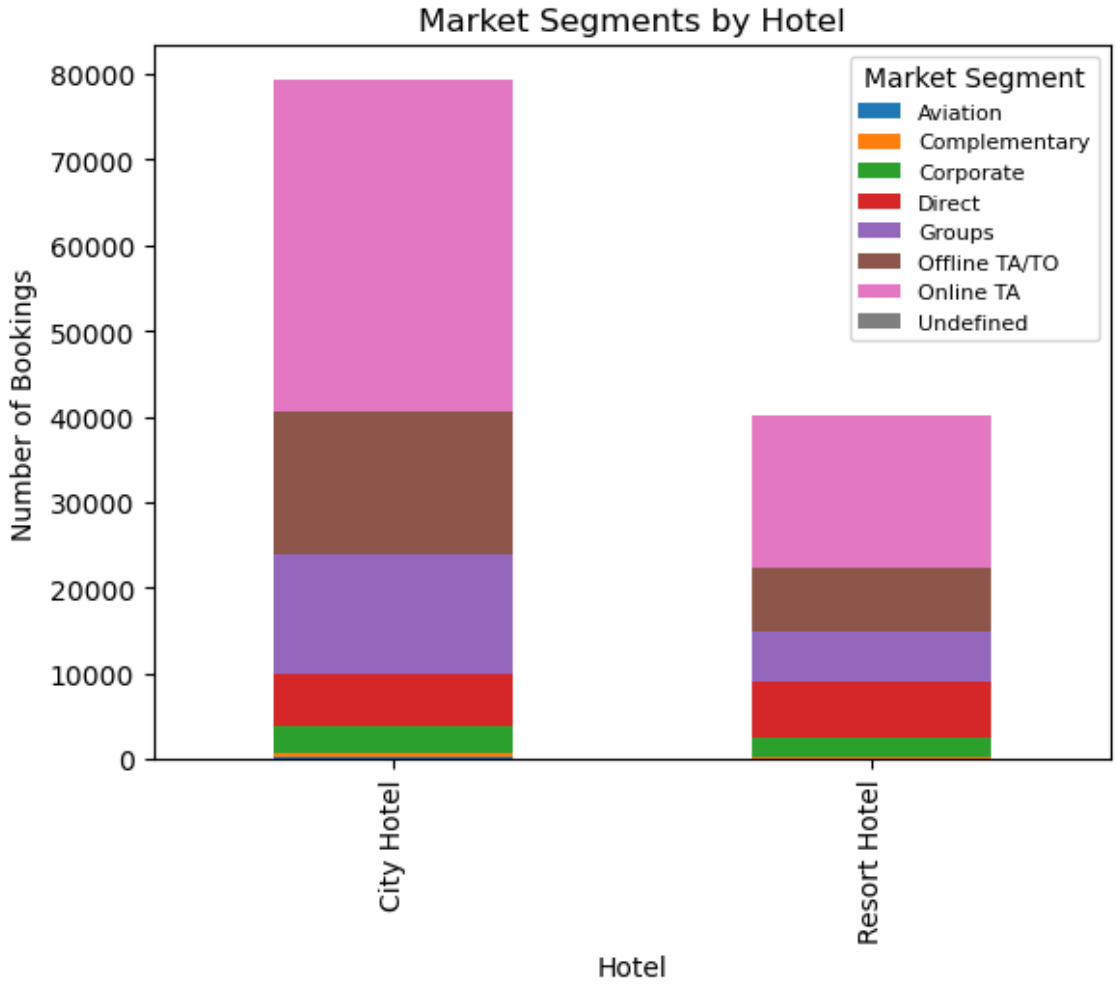
```

```

In [28]:

```
total_special_requests = df.groupby('hotel')['total_of_special_requests'].sum().res

# Visualize the results
sns.barplot(x='hotel', y='total_of_special_requests', data=total_special_requests)
plt.title('Total Special Requests per Hotel')
plt.xlabel('Hotel')
plt.ylabel('Total Special Requests')

# Add labels to the bars
for index, row in total_special_requests.iterrows():
    plt.text(row.name, row.total_of_special_requests, row.total_of_special_requests

plt.show()
```

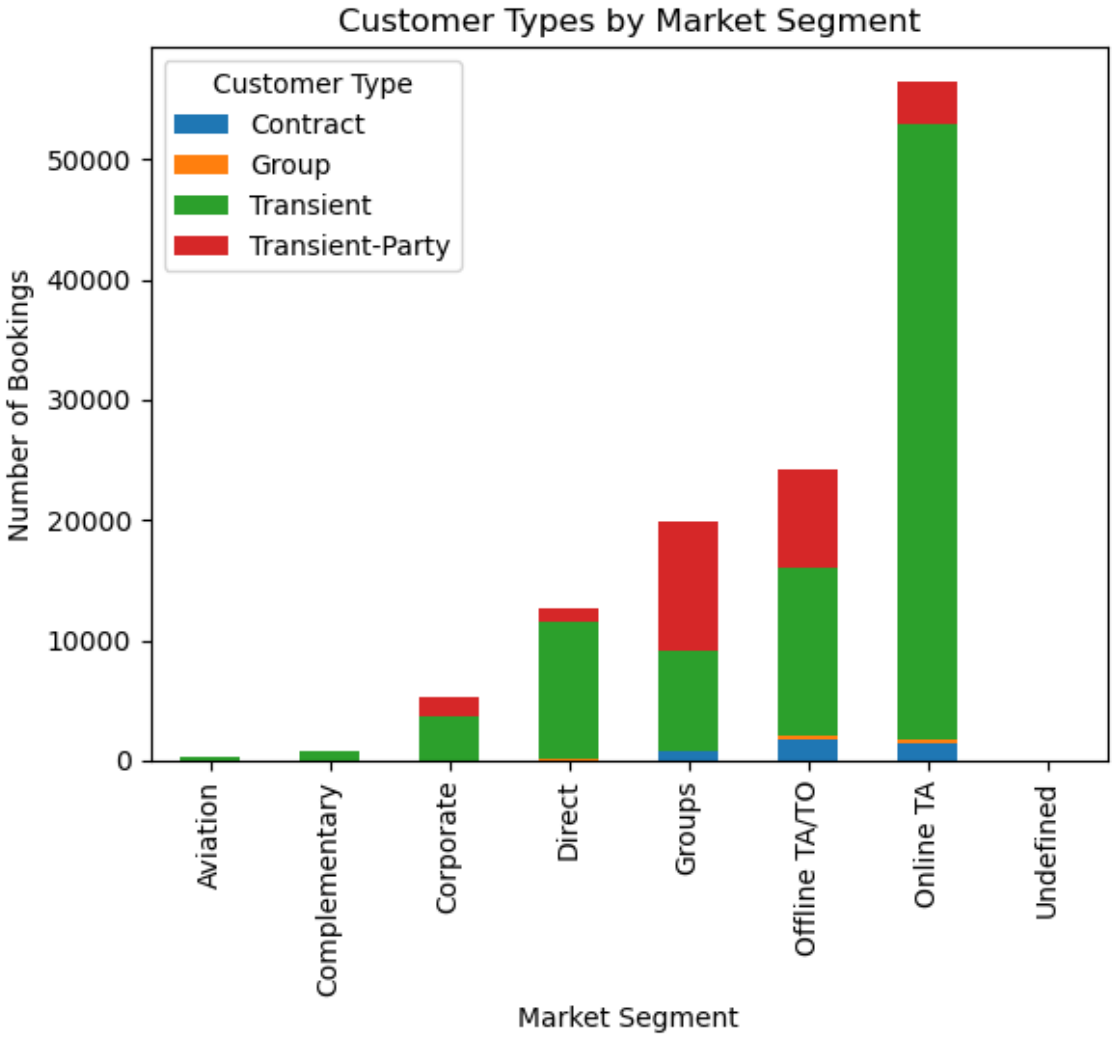
In [29]:

```
# Filter for relevant columns
dfc = df[['hotel', 'market_segment']]

# Group the data by hotel and market segment
grouped = df.groupby(['hotel', 'market_segment']).size().reset_index(name='count')

# Pivot the table to show market segment counts by hotel
pivoted = pd.pivot_table(grouped, values='count', index='hotel', columns='market_se

# Create a stacked bar plot
pivoted.plot(kind='bar', stacked=True)

# Add labels and legend
plt.title('Market Segments by Hotel')
plt.xlabel('Hotel')
plt.ylabel('Number of Bookings')
plt.legend(title='Market Segment', prop={'size': 8})

plt.show()
```



In [30]:

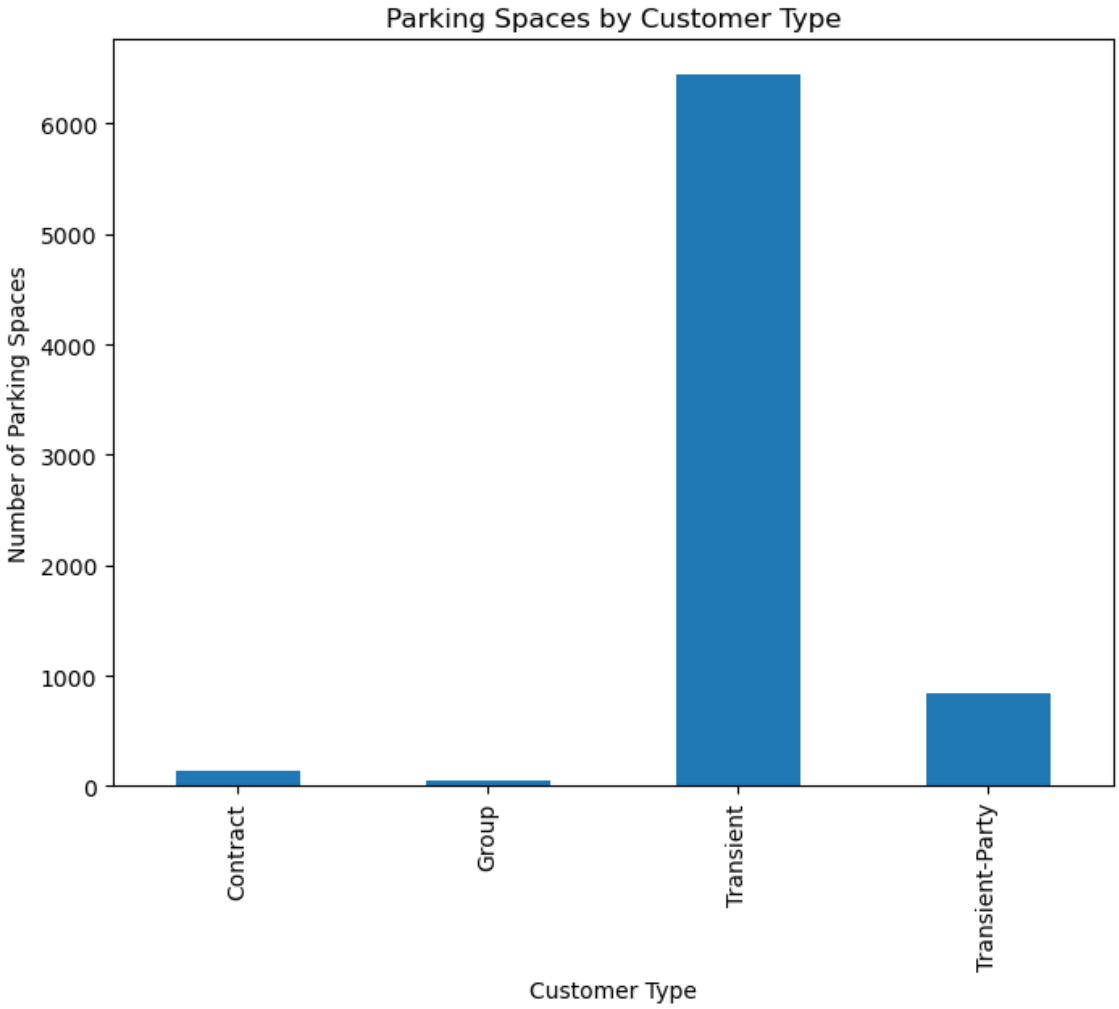
```
# Filter for relevant columns
dfc = df[['market_segment', 'customer_type']]

# Group the data by market segment and customer type
grouped = dfc.groupby(['market_segment', 'customer_type']).size().reset_index(name=

# Pivot the table to show customer type counts by market segment
pivoted = pd.pivot_table(grouped, values='count', index='market_segment', columns='

# Create a stacked bar plot
pivoted.plot(kind='bar', stacked=True)

# Add labels and legend
plt.title('Customer Types by Market Segment')
plt.xlabel('Market Segment')
plt.ylabel('Number of Bookings')
plt.legend(title='Customer Type')

plt.show()
```



In [31]:

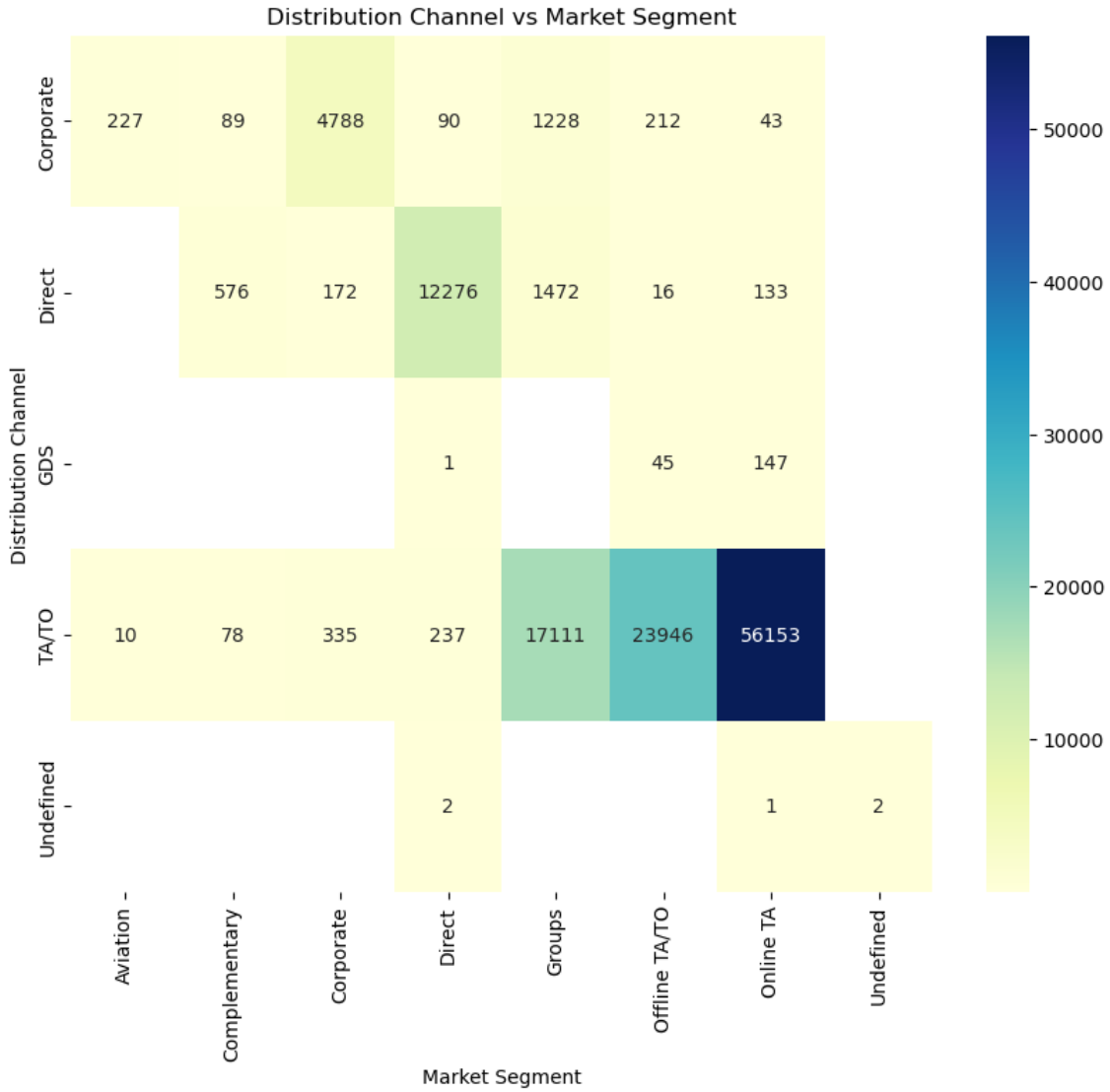
```
# Filter for relevant columns and remove undefined customer type
dfc = df[['customer_type', 'required_car_parking_spaces']]

# Rename the column to 'parking'
dfc = dfc.rename(columns={'required_car_parking_spaces': 'parking'})

# Group by customer type and sum the required parking spaces
parking_by_customer = dfc.groupby('customer_type')['parking'].sum()

# Find the customer type with the highest parking space requests
most_parking_customer = parking_by_customer.idxmax()
most_parking_spaces = parking_by_customer.max()

print(f"The customer type with the most requested parking spaces is {most_parking_c

# Create a bar plot of parking spaces by customer type
plt.figure(figsize=(8, 6))
parking_by_customer.plot(kind='bar')
plt.title('Parking Spaces by Customer Type')
plt.xlabel('Customer Type')
plt.ylabel('Number of Parking Spaces')

plt.show()
```

```
The customer type with the most requested parking spaces is Transient wit
h 6440 parking spaces.
```



In [32]:

```
# Filter for relevant columns
dfc = df[['distribution_channel', 'market_segment']]

# Group by distribution channel and market segment and count the number of occurren
channel_market_count = dfc.groupby(['distribution_channel', 'market_segment']).size

# Create a pivot table with distribution channel as the rows, market segment as the
channel_market_pivot = channel_market_count.pivot(index='distribution_channel', col

# Create a heatmap to visualize the relationship between distribution channel and m
plt.figure(figsize=(10, 8))
sns.heatmap(channel_market_pivot, cmap='YlGnBu', annot=True, fmt='g', cbar= True)
plt.title('Distribution Channel vs Market Segment')
plt.xlabel('Market Segment')
plt.ylabel('Distribution Channel')

plt.show()
```



In [33]:

```
1 df.head()
```

Out[33]:

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

5 rows × 32 columns

In [34]:

```
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                        119390 non-null  float64
 11  babies                          119390 non-null  int64  
 12  meal                            119390 non-null  object 
 13  country                         119390 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                           119390 non-null  float64
 24  days_in_waiting_list            119390 non-null  int64  
 25  customer_type                   119390 non-null  object 
 26  adr                             119390 non-null  float64
 27  required_car_parking_spaces     119390 non-null  int64  
 28  total_of_special_requests       119390 non-null  int64  
 29  reservation_status              119390 non-null  object 
 30  reservation_status_date         119390 non-null  object 
 31  total_stay                      119390 non-null  int64  
dtypes: float64(3), int64(17), object(12)
memory usage: 29.1+ MB
```

Creating dataframe with all numeric variables - df_num

Creating dataframe with all categorical variables - df_cat

In [35]:

```
df_num = df.select_dtypes(exclude='object')
df_cat = df.select_dtypes(include='object')
```

In [36]:

```
df_num.head(15)
```

Out[36]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_
0	0	342	2015	27	
1	0	737	2015	27	
2	0	7	2015	27	
3	0	13	2015	27	
4	0	14	2015	27	
5	0	14	2015	27	
6	0	0	2015	27	
7	0	9	2015	27	
8	1	85	2015	27	
9	1	75	2015	27	
10	1	23	2015	27	
11	0	35	2015	27	
12	0	68	2015	27	
13	0	18	2015	27	
14	0	37	2015	27	

Melting all numerical data together with the help of pd.melt function

In [37]:

```
melted_data = pd.melt(df_num)
melted_data.head()
```

Out[37]:

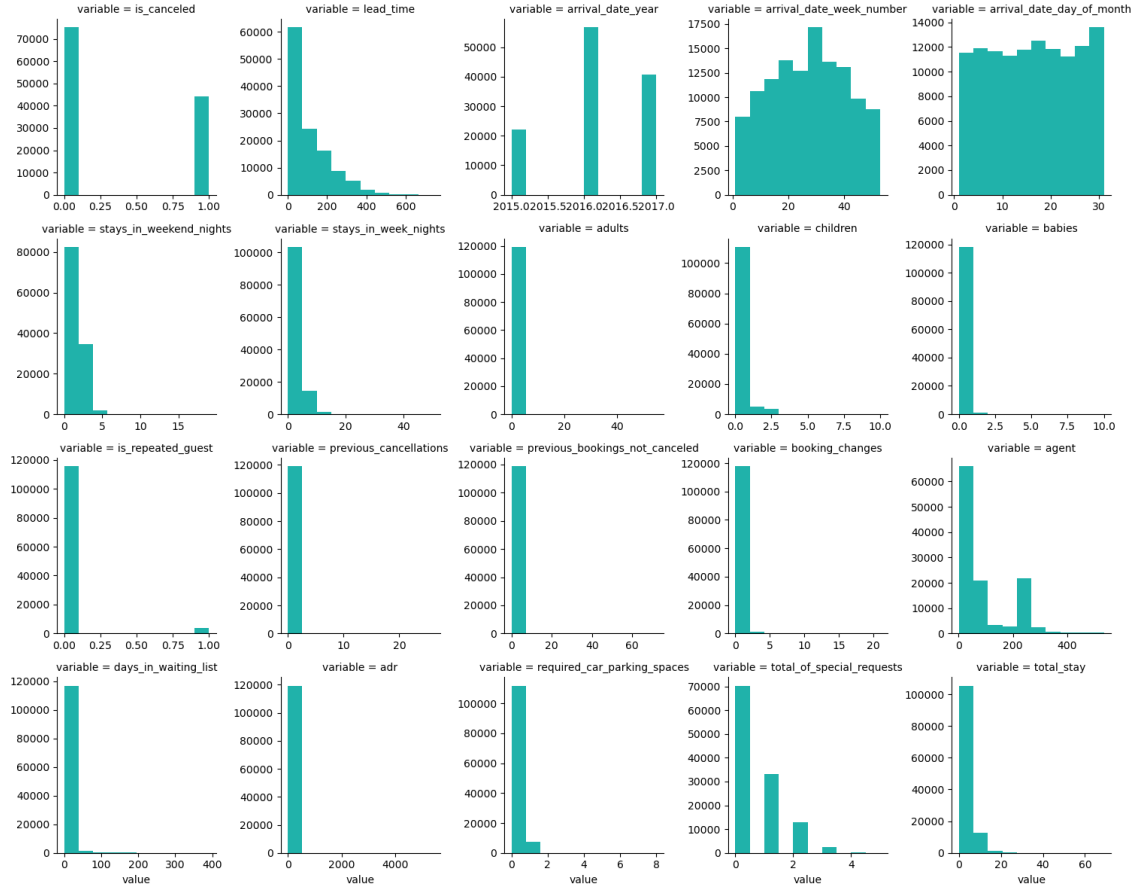
	variable	value
0	is_canceled	0.0
1	is_canceled	0.0
2	is_canceled	0.0
3	is_canceled	0.0
4	is_canceled	0.0

creating histograms for all melted data

Here Sharex and Sharey are kept false as there are no constant parameter to check upon

In [38]:

```
grid=sns.FacetGrid(melted_data, col="variable", col_wrap=5,sharex=False, sharey=Fal
grid=grid.map(plt.hist, "value", color='lightseagreen')
```



1. Around 45000 of the customers cancelled their booking. (0 indicates no and 1 indicates yes)

2. The graph is left skewed and shows that majority of the customers have lead time ranging from 0-200

3. Majority of the customers arrived in the year 2016.

4. Number of customers arriving at week 30 was the highest.

5. The number of customers arriving towards the end of the month are the highest.

6. Higher number of customers stay for 0-4 weekend nights.

7. Higher number of customers stay for 0-10 weekday nights.

8. For all customers, number fo adults are less than 5.

9. For majority customers,number of children range from 0-1.

10. For all customers, number of children range from 0-1.

11. A handful of customers are repeat customers, majority are not.

12. No customers cancelled their previous bookings.

13. The value of previous bookings not cancelled for all customers is zero.

14. Majority of the customers did not make any booking changes post making the booking.

15. Around 20000 customers used an agent to make the booking but majority did not.

16. Majority of the customers did not have to be in the waiting list to make their booking.

17. The average daily rate (ADR) ranges between 0-1000 for most customers.

18. Majority of the customers required no parking spaces, however a few requested one parking space.

19. Majority of the customers, around 65000 did not make any special requests

In [39]:

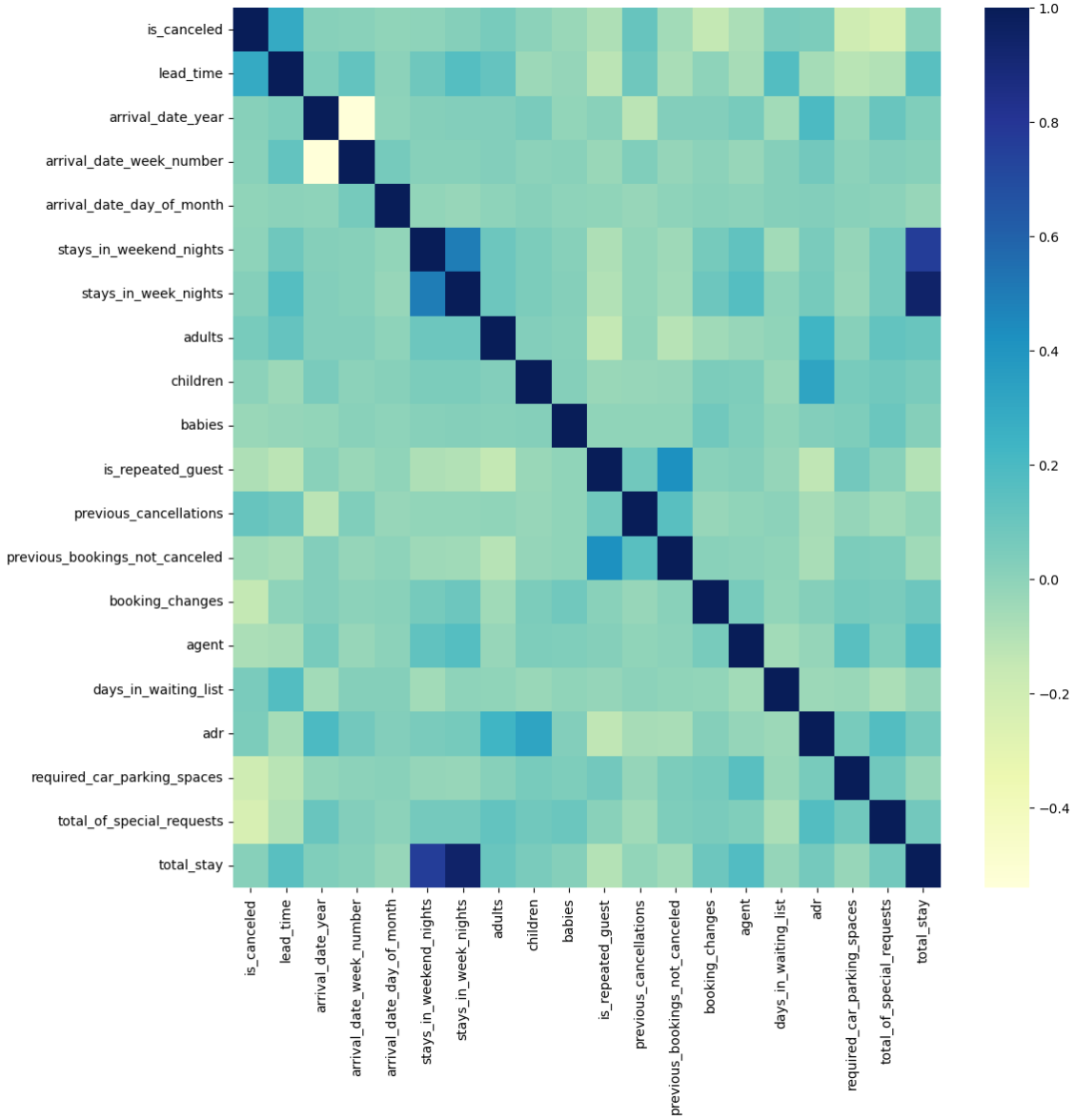
```
correlation = df_num.corr()
correlation
```

Out[39]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week
is_canceled	1.000000	0.293123	0.016660	
lead_time	0.293123	1.000000	0.040142	
arrival_date_year	0.016660	0.040142	1.000000	-
arrival_date_week_number	0.008148	0.126871	-0.540561	
arrival_date_day_of_month	-0.006130	0.002268	-0.000221	
stays_in_weekend_nights	-0.001791	0.085671	0.021497	
stays_in_week_nights	0.024765	0.165799	0.030883	
adults	0.060017	0.119519	0.029635	
children	0.005048	-0.037621	0.054622	
babies	-0.032491	-0.020915	-0.013192	
is_repeated_guest	-0.084793	-0.124410	0.010341	-
previous_cancellations	0.110133	0.086042	-0.119822	
previous_bookings_not_canceled	-0.057358	-0.073548	0.029218	-
booking_changes	-0.144381	0.000149	0.030872	
agent	-0.077992	-0.065283	0.058851	-
days_in_waiting_list	0.054186	0.170084	-0.056497	
adr	0.047557	-0.063077	0.197580	
required_car_parking_spaces	-0.195498	-0.116451	-0.013684	
total_of_special_requests	-0.234658	-0.095712	0.108531	
total_stay	0.017779	0.157167	0.031438	

In [40]:

```
plt.figure(figsize=(12,12))
sns.heatmap(correlation, cmap = "YlGnBu")
plt.show()
```



In [41]:

```
df.columns
```

Out[41]:

```
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',
       'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'total_stay'],
      dtype='object')
```

In [42]:

```
1 df.head()
```

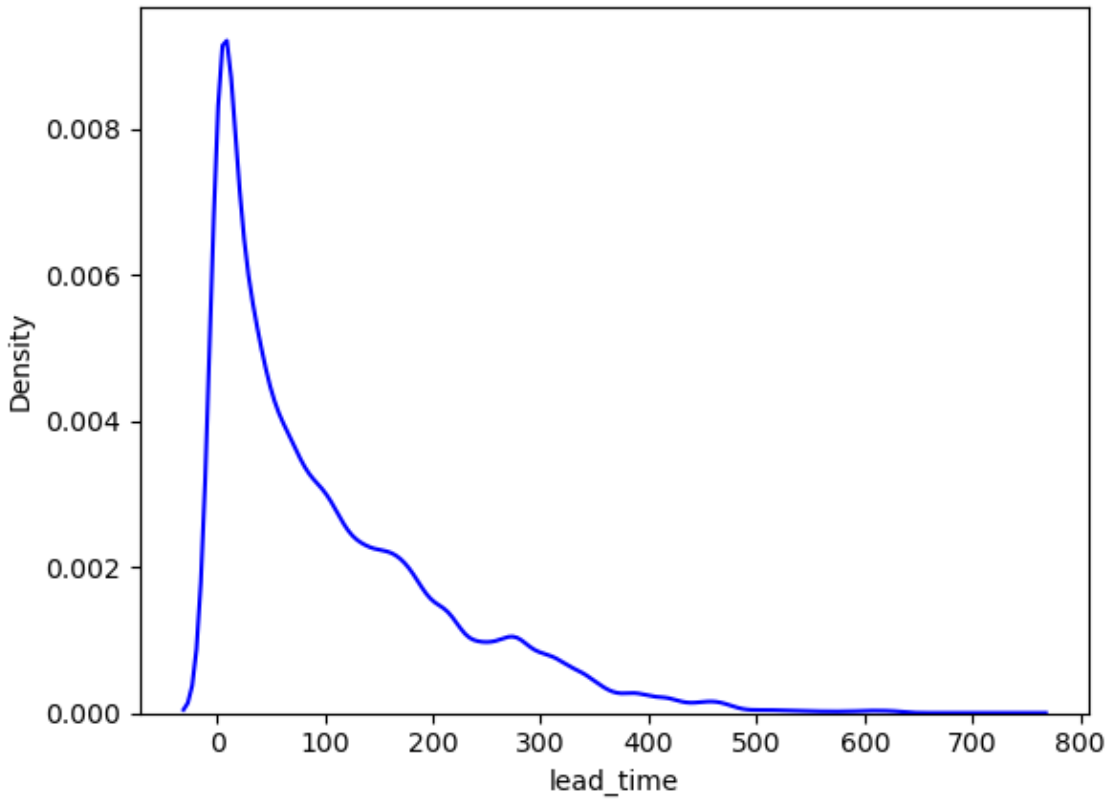
Out[42]:

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

5 rows × 32 columns

In [43]:

```
1 sns.kdeplot(data=df, x='lead_time', color='blue')
2 plt.show()
```



What is the average daily rate (ADR) for each room type? How does this vary by month?

In [44]:

```
df_roomtype = df.groupby('assigned_room_type')
df_roomtype['adr'].mean()
```

Out[44]:

```
assigned_room_type
A     93.142347
B     94.450264
C    113.423583
D    107.453633
E    117.704224
F    151.889931
G    166.530309
H    171.380772
I     40.843774
K     53.698889
L      8.000000
P      0.000000
Name: adr, dtype: float64
```

In [45]:

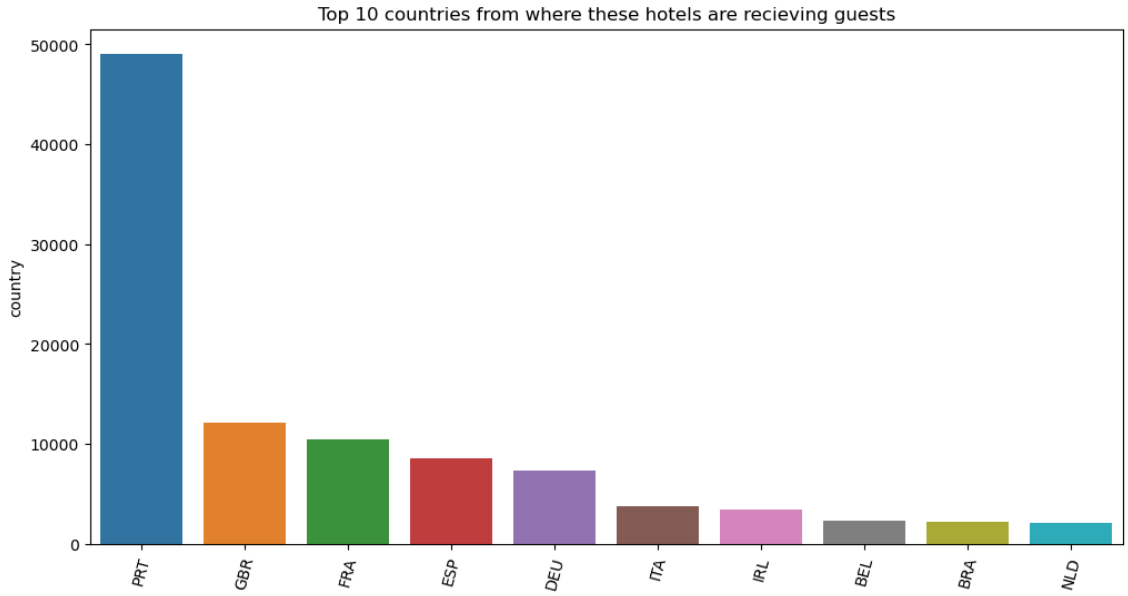
```
df_topcountries = df['country'].value_counts().head(10)
df_topcountries
```

Out[45]:

```
PRT    49078
GBR    12129
FRA    10415
ESP     8568
DEU     7287
ITA     3766
IRL     3375
BEL     2342
BRA     2224
NLD     2104
Name: country, dtype: int64
```

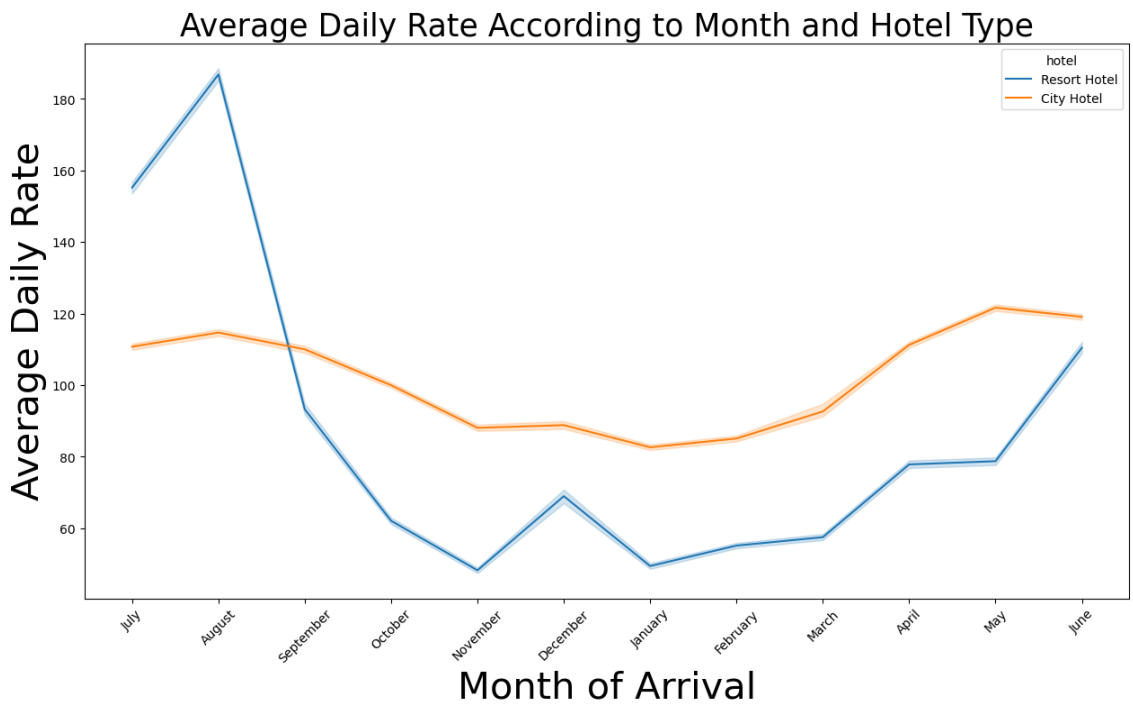
In [46]:

```
plt.figure(figsize=(12,6))
plt.xticks(rotation=75)
plt.title('Top 10 countries from where these hotels are recieving guests')
sns.barplot(x=df_topcountries.index, y=df_topcountries);
```



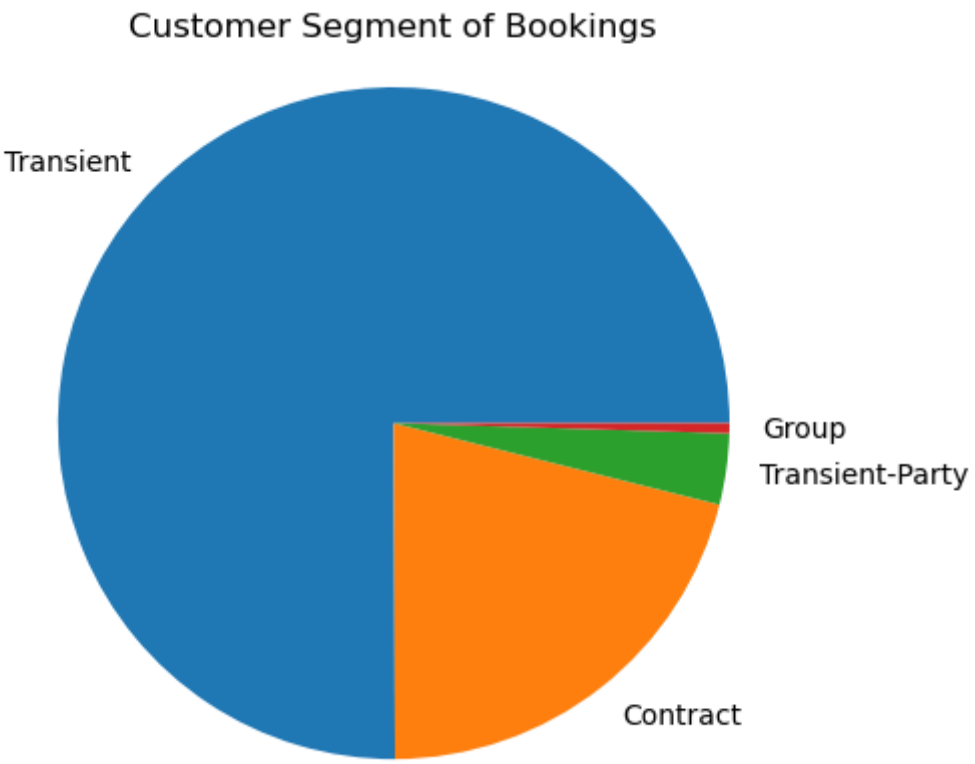
In [47]:

```
plt.figure(figsize = (15,8))
fig = sns.lineplot(data = df, x = 'arrival_date_month',y = 'adr',hue = 'hotel')
plt.xlabel('Month of Arrival', fontsize = 30)
plt.xticks(rotation=45)
plt.ylabel('Average Daily Rate', fontsize = 30)
plt.title('Average Daily Rate According to Month and Hotel Type', fontsize = 25);
```



In [48]:

```
sizes = df['customer_type'].value_counts()
labels = df['customer_type'].unique()

plt.pie(sizes, labels=labels)
plt.title('Customer Segment of Bookings')
plt.axis('equal')

plt.show()

```



In [49]:

```
plt.figure(figsize=(15,10))   #size of the chart
sns.countplot(x="arrival_date_month", data=df, order = df['arrival_date_month'].val
plt.title('BOOKED TYPE',fontsize=25)     # set title of this chart
plt.ylabel('COUNT',fontsize=25)                  # set y axis column name
plt.xlabel('BOOKED MONTH',fontsize=25)         # set x axis coumn name
plt.show()
```

How many booking were cancelled?

To find how many bookings were cancelled, using value_counts functions
The value_counts will return the frequency of unique values in descending order
The 0 value represent that the particular booking is not cancelled and 1 represent that the particular booking is cancelled

In [50]:

```
df['is_canceled'].value_counts()
```

Out[50]:

```
0    75166
1    44224
Name: is_canceled, dtype: int64
```

Plotting Bar Plot of the above question for better visuals using matplotlib

In [51]:

```
# storing categories in the cat variable
cat = df['is_canceled'].unique()

# storing values corresponding to each category in the vals variable
vals = df['is_canceled'].value_counts('1')

#plotting bar chart
colors = ['tan', 'indianred']

# plotting bar plot and giving aruguement to select categories as strings, values c
plt.bar(cat.astype(str), vals, color= colors)

# Naming x-label
plt.xlabel("Booking Cancelled(No = 0, Yes = 1)")

# Naming y-label
plt.ylabel("Percentage%")

# Naming the title of the bar plot
plt.title("Percentage of booking cancelled")

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

INSIGHT :

The booking Non-cancellation to cancellation ratio is approximately 2:1 (60%:30%). There are 30% of the total times where the booking got cancelled and 60% of the time it did not.

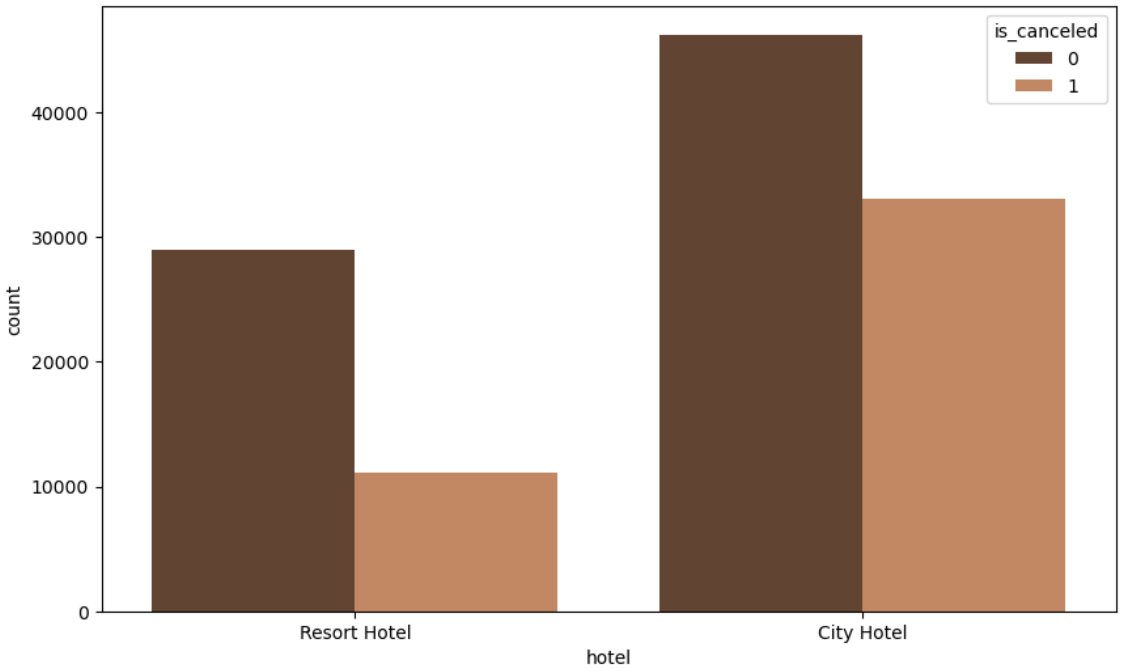
It means that out of three times there might be one time where bookings can get cancelled.

Which Hotel has More Cancelled Booking?

Plotting countplot using function sns.countplot
Countplot Show the counts of observations in each categorical bin using bars
This function contains name of dataset as df, x axis as the category, hue is a categorical variable(dont need to filter the dataset), palette is the color for plot

In [52]:

```
# to set the size of the canvas
plt.figure(figsize=(10,6))

#Using countplot
sns.countplot(x="hotel", data=df, hue="is_canceled", palette="copper")
plt.show()
```



INSIGHT :
City Hotel has highest number of cancelled bookings. Resort Hotel has the least number of cancelled bookings.
It means that Resort Hotel bookings does not cancelled as frequently as City Hotels. It might suggest that there can be a high demand of Resort Hotel owing to less cancellations.

CANCELLATION TRENDS OVER YEARS ON THE BASIS OF HOTEL

Here we are tying to find how many cancellations have taken place in which type of hotel over the years

Crosstab is function provided by pandas that computes a frequency table of the factors passed as an arguements Here 'is_canceled' (cancellation status) and 'hotel' (type of hotel) is passed as a list for index arguement and another factor 'arrival_date_year' (year) is passed as column arguement

The margins arguement is set to true to check the total sum of all categories created

In [53]:

```
pd.crosstab([df['is_canceled'],df['hotel']], df['arrival_date_year'],margins=True)
```

Out[53]:

	arrival_date_year	2015	2016	2017	All
is_canceled	hotel				
0	City Hotel	7678	22733	15817	46228
	Resort Hotel	6176	13637	9125	28938
1	City Hotel	6004	15407	11691	33102
	Resort Hotel	2138	4930	4054	11122
All		21996	56707	40687	119390

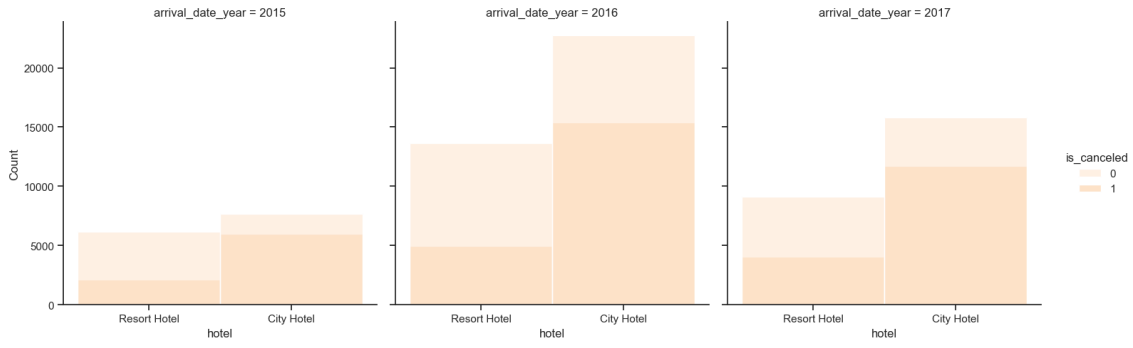
To visualize the above information found through pd.crosstabs, using seasborn displot

sns.set_theme sets aspects of the visual theme for all matplotlib and seaborn plots. Style sets the theme to given arguement, here it is ticks. It can be whitegrid, darkgrid, dark etc. Pallete is a set of colors used to represent data in plots created with the Seaborn

Distribution plots are used here to visualize the data using sns.displot() which is used to create a FacetGrid of histograms for the 'hotel' column in the DataFrame df, with each plot colored according to the is_canceled column, and separated into columns according to the arrival_date_year column.

In [54]:

```
sns.set_theme(style='ticks', palette='Oranges')
sns.displot(data = df, x='hotel', hue ='is_canceled', col='arrival_date_year')
plt.show()
```



INSIGHT :
In all the three years (2015,2016,2017) , City hotel have more cancellation than Resort hotel.
Meanwhile in year 2016 the count of booking cancellation is highest in all the three years

What is the booking ratio between Resort Hotel and City Hotel?

Here we are trying to find the how many hotels booked in Resort Hotel and City Hotel respectively and what is the ratio between the same

Creating a new dataframe named as df_not_cancelled which stores all the values where the cancellation status is No(or 0) i.e. the particular hotel have been book or its booking status is poistive. Then in the next line checking the first five rows of the dataframe to validate if the dataframe has correct values or not.

In [55]:

```
df_not_cancelled = df[df['is_canceled']==0]
df_not_cancelled.head()
```

Out[55]:

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

5 rows × 32 columns

Plotting bar plot to visualize the values. There are two desired categories and their percentage is also calculated in the vals variable so using a bar plot.

In [56]:

```
# storing categories in the cat variable
cat = df_not_cancelled['hotel'].unique()

# storing values in percentage corresponding to each category in the vals variable
vals = df_not_cancelled['hotel'].value_counts() *100 / df_not_cancelled.shape[0]

#plotting bar chart
colors = ['lightsalmon', 'sienna']

# plotting bar plot and giving aruguement to select categories as strings, values c
plt.bar(cat, vals, color= colors)

# Naming x-label
plt.xlabel("Type of hotel")

# Naming y-label
plt.ylabel("Percentage of bookings")

# Naming the title of the bar plot
plt.title("Percentage of booking made by type of hotel")

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

INSIGHT :

There are more bookings in the Resort hotel i.e.around 60% of the total bookings than the City hotel i.e. around 40% of the total bookings.

The booking ratio of the Resort to City hotel is 60:30 or 2:1

What is the percentage of booking for each year?

Here we are trying to find the year wise comparison of the number of bookings in both the hotels and also we categorise this into type of hotels i.e. Year wise booking in City and Resort Hotel.

Using the same dataframe that we made earlier named as df_not_cancelled. Also using value_counts function to calculate the total number of bookings made per year and dividing it by total rows to find the percentage. Storing the calculation in a variable named x for further use.

In [57]:

```
x = df_not_cancelled['arrival_date_year'].value_counts() / df_not_cancelled.shape[0
x
```

Out[57]:

```
2016    0.483862
2017    0.331826
2015    0.184312
Name: arrival_date_year, dtype: float64
```

Plotting bar plot to visualize the values. There are desired year as categories and their percentage is also calculated in the vals variable so using a bar plot.

In [58]:

```
# storing categories in the cat variable
cat = x.keys()

# storing values in percentage corresponding to each category in the vals variable
vals = x.values

# plotting bar plot and giving aruguement to select categories as strings, values c
sns.barplot(x=cat, y=vals, palette="Reds")

# Naming x-label
plt.xlabel("Year of booking")

# Naming y-label
plt.ylabel("Percentage of bookings")

# Naming the title of the bar plot
plt.title("Year wise percentage of booking")

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

INSIGHT :

The number of booking made in 2016 were almost double the number of bookings in 2015, and the number of booking were less than by 15% in 2017.

What is the duration of stay of people in the hotel?

Here we are trying to find the length of duration of people stay in the hotel or for how long people stay in the hotel.

The total variable contains total number of nights people stayed as it calculates the number of occurrences of each possible stays on week nights and stays on weekend nights, and stores the resulting value counts in the x variable.
This next line calculates the value counts of the total Series, which gives you a Pandas Series where the index contains each unique value of total, and the values contain the number of occurrences of each value

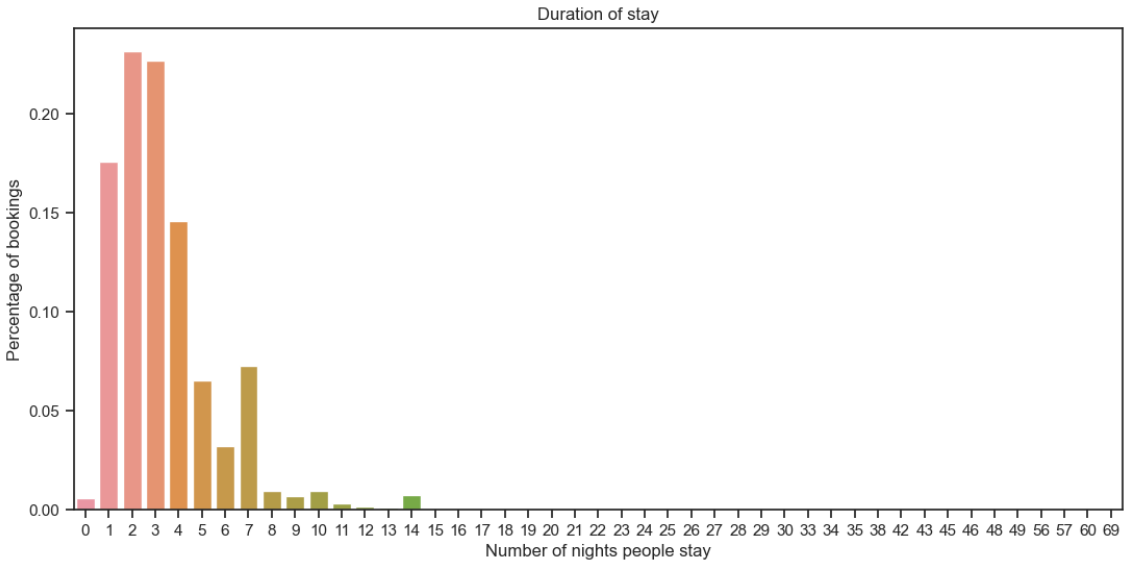
In [59]:

```
total = df['stays_in_week_nights'] + df['stays_in_weekend_nights']

x = total.value_counts()
```

Plotting bar plot to visualize the values. There are desired number of nights for stay as categories and their percentage is also calculated in the vals variable so using a bar plot.

In [60]:

```
# storing categories in the cat variable
cat = x.keys()

# storing values in percentage corresponding to each category in the vals variable
vals = x.values/df.shape[0]

# To set the size of the bar plot
plt.figure(figsize=(13,6))

# plotting bar plot and giving aruguement to select categories as strings, values c
sns.barplot(x=cat, y=vals)

# Naming x-label
plt.xlabel("Number of nights people stay")

# Naming y-label
plt.ylabel("Percentage of bookings")

# Naming the title of the bar plot
plt.title("Duration of stay")

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



INSIGHT :

The lengthiest duration of people stay is one, two, three, four nights. As the number of days increase more than six or seven, the duration of people's stay decreases.

Popular types of Hotels among guests

In [61]:

```
# Fisrt we defined the varible (cat) which contain the unique values of the "Hotel"
# After that we found out there values counts using .value_counts().
# We also plot the bar chart for this variable to show case the different types of
```

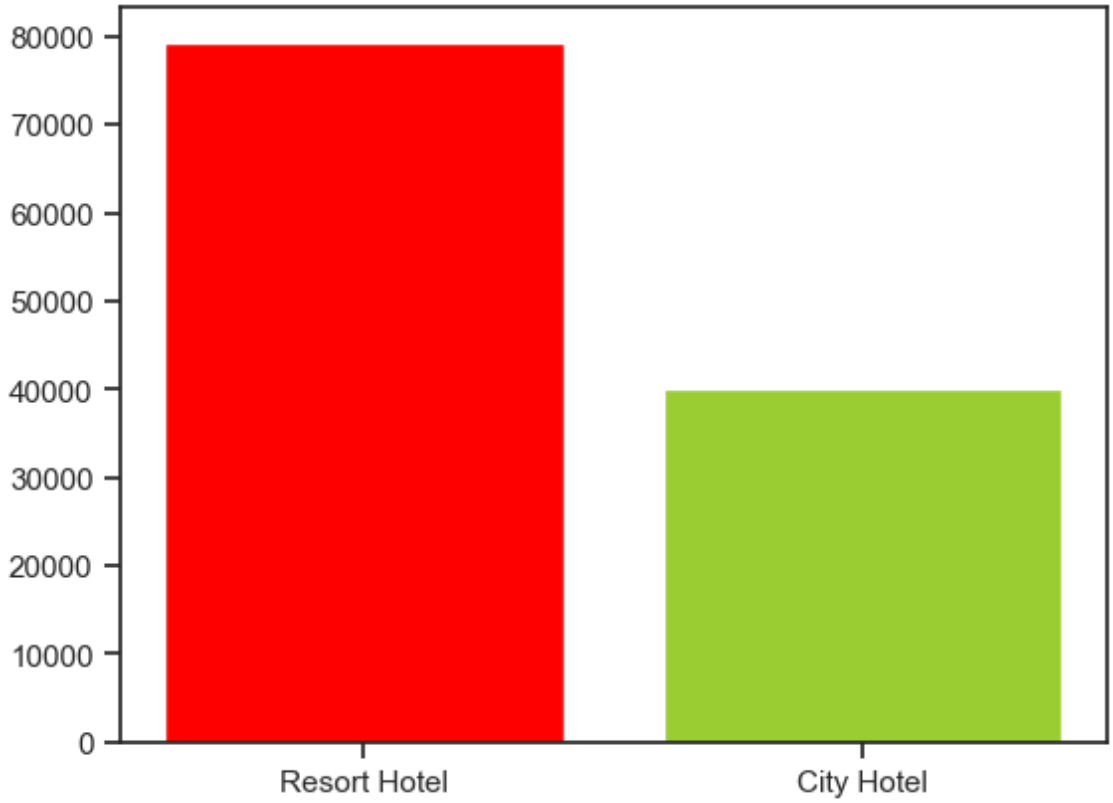
In [62]:

```
cat=df['hotel'].unique()
vals=df['hotel'].value_counts()
print(cat,vals)
```

```
['Resort Hotel' 'City Hotel'] City Hotel      79330
Resort Hotel    40060
Name: hotel, dtype: int64
```

In [63]:

```
colors=['red','yellowgreen','gold','seagreen']
plt.bar(cat,vals,color=colors)
plt.show()
```



In [64]:

```
# It can be seen both from the bar plot and the number of counts that "Resort Hotel
# According to our data set total 40060 number of guests prefer to stay in "Resort
```

Which country have the highest number of hotel bookings in a given period?

In [65]:

```
# First we filter out the country that appears most of the times
# We used .loc to filteration
```

In [66]:

```
df_highest=df.loc[df['country']==df['country'].max(),'hotel':'country']
df_highest
```

Out[66]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_wee
2905	Resort Hotel	0	15	2015	November	
33604	Resort Hotel	0	25	2017	February	
73145	City Hotel	1	215	2017	August	
73460	City Hotel	1	77	2017	August	

It is observed that country ZWE has the highest number of hotels booking.

What are the busiest months for hotel bookings? Are there any seasonal patterns?

In [67]:

```
# We used the pd.crosstab to find find out counts of number of the hotels booked in

pd.crosstab([df['arrival_date_year'],df['arrival_date_month']],df['hotel'],margins=
```

Out[67]:

	hotel	City Hotel	Resort Hotel	All
arrival_date_year	arrival_date_month			
2015	August	2480	1409	3889
	December	1654	1266	2920
	July	1398	1378	2776
	November	1235	1105	2340
	October	3386	1571	4957
	September	3529	1585	5114
2016	April	3561	1867	5428
	August	3378	1685	5063
	December	2478	1382	3860
	February	2371	1520	3891
	January	1364	884	2248
	July	3131	1441	4572
	June	3923	1369	5292
	March	3046	1778	4824
	May	3676	1802	5478
	November	3122	1332	4454
	October	4219	1984	6203
	September	3871	1523	5394
2017	April	3919	1742	5661
	August	3125	1800	4925
	February	2594	1583	4177
	January	2372	1309	3681
	July	3559	1754	5313
	June	3971	1676	5647
	March	3412	1558	4970
	May	4556	1757	6313
All		79330	40060	119390

In [68]:

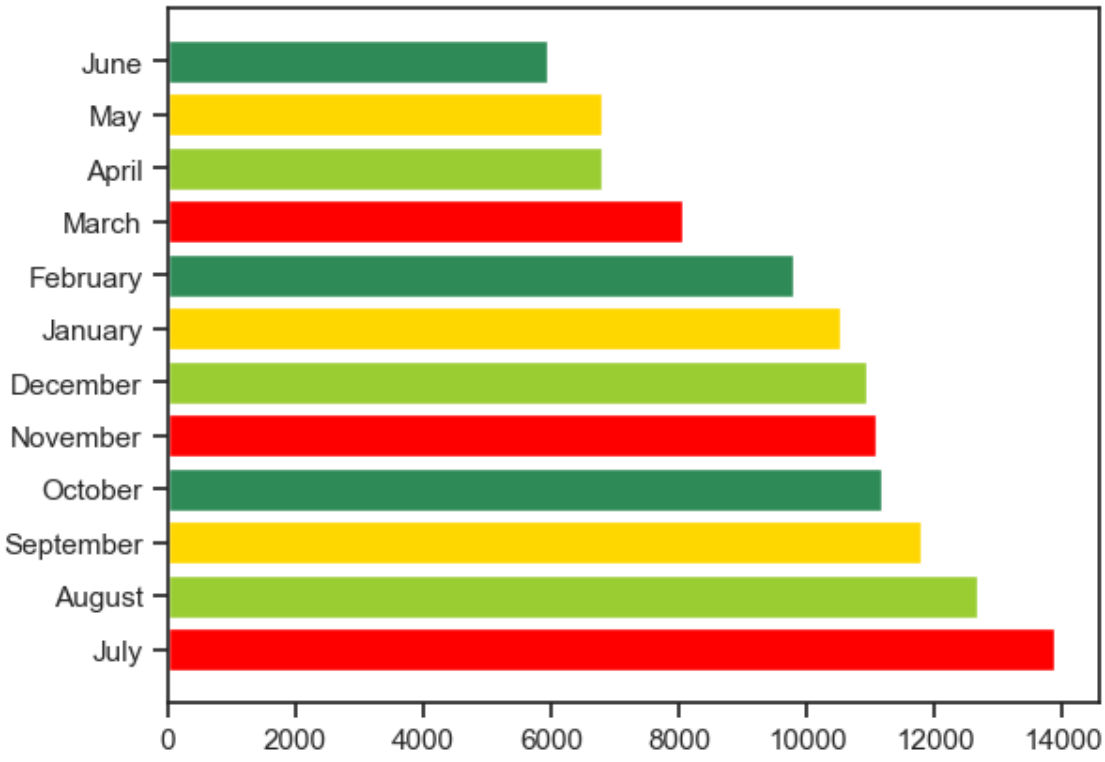
```
# To show the how many hotles are booked in which months and to observe the pattern
# To plot the categorical varible first we defined the other two variables which co
# After that using "plt.barh()" we plotted horizontal bar plot.
cat1=df['arrival_date_month'].unique()
vals2=df['arrival_date_month'].value_counts()
```

It can be observed that mostly the guests prefer to travel between july to November, as most of the booking for different types of hotel happens during these months. It can also be seen from the corss table.

What is the distribution of booking lead time (i.e. the time between booking and check-in date) for different types of hotels?

In [69]:

```
colors=['red','yellowgreen','gold','seagreen']
plt.barh(cat1,vals2,color=colors)
plt.show()
```



It can be observed that mostly the guests prefer to travel between july to November, as most of the booking for different types of hotel happens during these months. It can also be seen from the corss table.

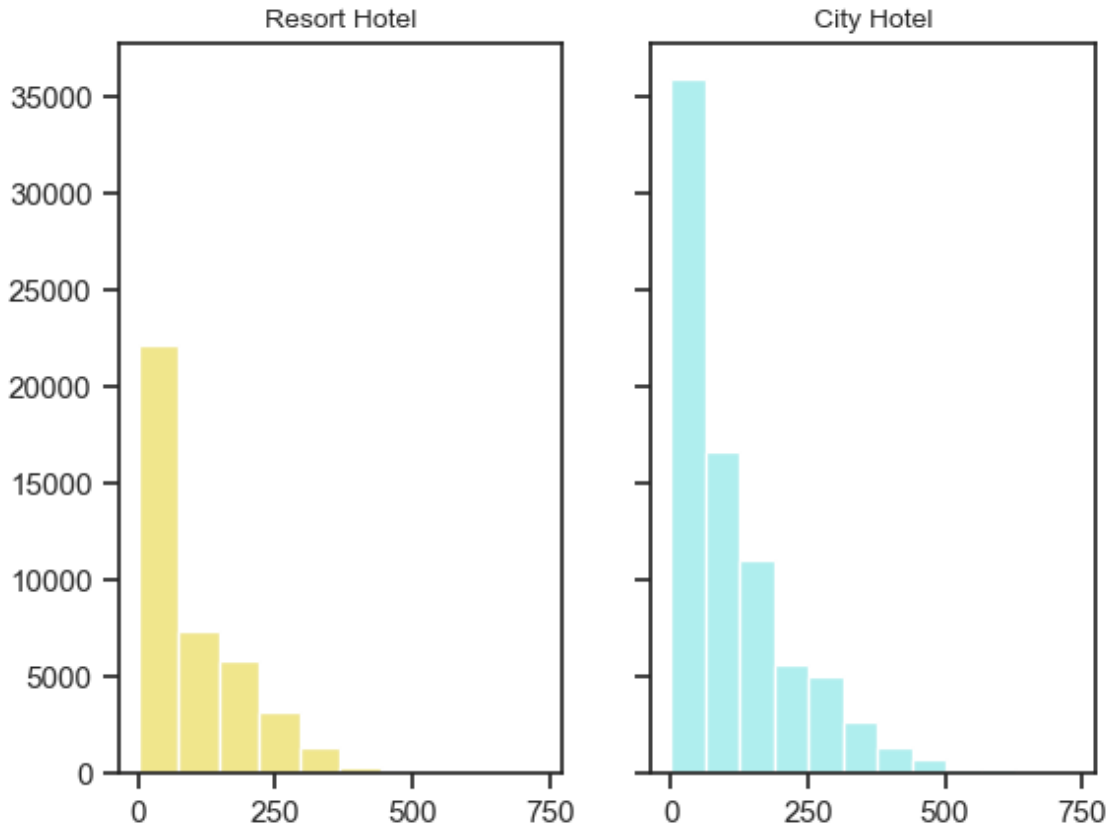
What is the distribution of booking lead time (i.e. the time between booking and check-in date) for different types of hotels?

In [70]:

```
# We filter out the data for the different types of hotel
# To know the distribution of booking lead time for the different types of hotel we

df_resort=df.loc[df['hotel']=='Resort Hotel',]
df_city=df.loc[df['hotel']=='City Hotel',]
```

In [71]:

```
fig,(ax1,ax2) =plt.subplots(1,2,sharex=True,sharey=True)

ax1.hist(df_resort['lead_time'],color='khaki')
ax2.hist(df_city['lead_time'],color='paleturquoise')

ax1.set_title("Resort Hotel",fontsize=10)
ax2.set_title("City Hotel",fontsize=10)
plt.show()
```



It can be observe from the graph that for both for the city hotel and resort hotel the data is not normally distibuted , It has the both skewness and kortosis.

The reason for the data to be not normally distributed is it have some outliers.

What is the average length of stay for guests?

In [72]:

```
# First we will add the another column which will contain length of the stay.

df['Total_Stay']=df['stays_in_weekend_nights']+df['stays_in_week_nights']
df['Total_Stay'].value_counts()
```

Out[72]:

```
2     27643
3     27076
1     21020
4     17383
7      8655
5      7784
6      3857
8      1161
10     1139
14      916
9       841
0       715
11      396
12      223
13      142
15       75
21       71
16       40
25       37
18       35
28       35
19       22
17       20
29       14
20       14
22       14
30       13
23        8
24        6
26        6
27        5
35        5
42        4
33        3
56        2
34        1
57        1
49        1
48        1
69        1
38        1
45        1
60        1
46        1
43        1
Name: Total_Stay, dtype: int64
```

In [73]:

```
1 df['Total_Stay'].mean()
```

Out[73]:

```
3.4279001591423066
```

The average length of stay for all the type of hotels is 3.42 which is about 4 days

How many booking were cancelled?

To find how many bookings were cancelled, using value_counts functions
The value_counts will return the frequency of unique values in descending order
The 0 value represent that the particular booking is not cancelled and 1 represent that the particular booking is cancelled

In [74]:

```
1 df['is_canceled'].value_counts()
```

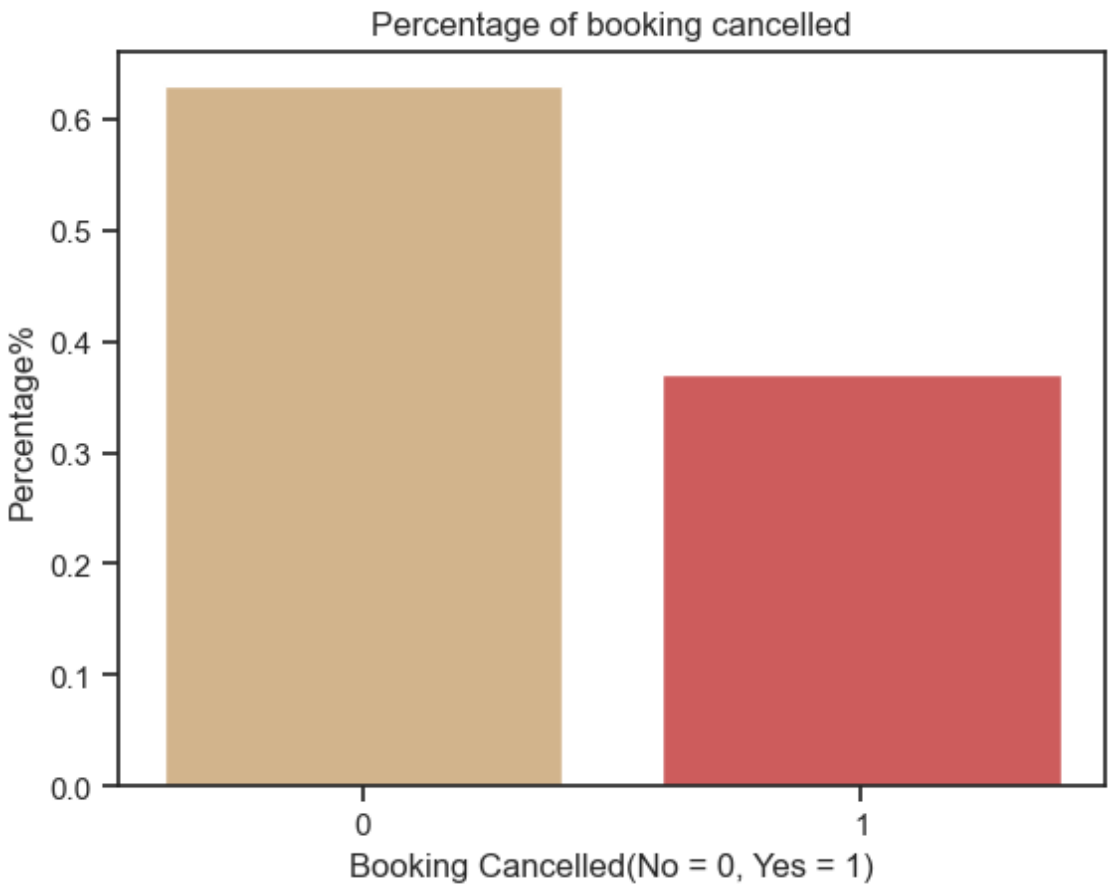
Out[74]:

```
0    75166
1    44224
Name: is_canceled, dtype: int64
```

Plotting Bar Plot of the above question for better visuals using matplotlib

In [75]:

```
# storing categories in the cat variable
cat = df['is_canceled'].unique()

# storing values corresponding to each category in the vals variable
vals = df['is_canceled'].value_counts('1')

#plotting bar chart
colors = ['tan', 'indianred']

# plotting bar plot and giving aruguement to select categories as strings, values c
plt.bar(cat.astype(str), vals, color= colors)

# Naming x-label
plt.xlabel("Booking Cancelled(No = 0, Yes = 1)")

# Naming y-label
plt.ylabel("Percentage%")

# Naming the title of the bar plot
plt.title("Percentage of booking cancelled")

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



INSIGHT :

The booking Non-cancellation to cancellation ratio is approximately 2:1 (60%:30%). There are 30% of the total times where the booking got cancelled and 60% of the time it did not.

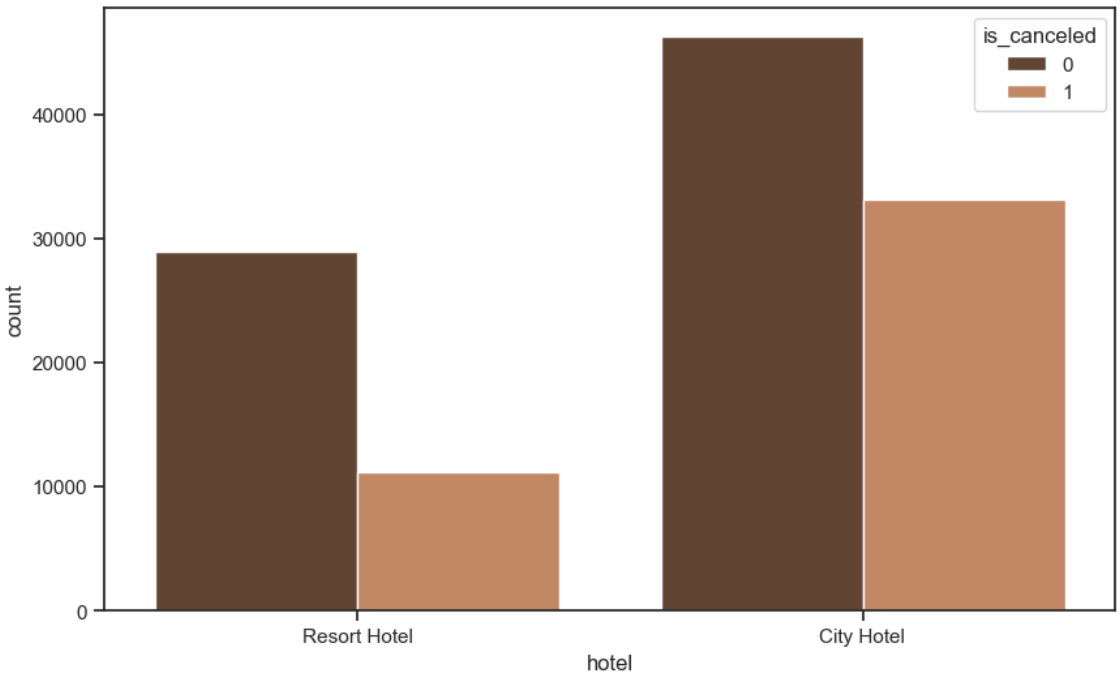
It means that out of three times there might be one time where bookings can get cancelled.

Which Hotel has More Cancelled Booking?

Plotting countplot using function sns.countplot
Countplot Show the counts of observations in each categorical bin using bars
This function contains name of dataset as df, x axis as the category, hue is a categorical variable(dont need to filter the dataset), palette is the color for plot

In [76]:

```
# to set the size of the canvas
plt.figure(figsize=(10,6))

#Using countplot
sns.countplot(x="hotel", data=df, hue="is_canceled", palette="copper")
plt.show()
```



INSIGHT :
City Hotel has highest number of cancelled bookings. Resort Hotel has the least number of cancelled bookings.
It means that Resort Hotel bookings does not cancelled as frequently as City Hotels. It might suggest that there can be a high demand of Resort Hotel owing to less cancellations.

CANCELLATION TRENDS OVER YEARS ON THE BASIS OF HOTEL

Here we are tying to find how many cancellations have taken place in which type of hotel over the years

Crosstab is function provided by pandas that computes a frequency table of the factors passed as an arguements Here 'is_canceled' (cancellation status) and 'hotel' (type of hotel) is passed as a list for index arguement and another factor 'arrival_date_year' (year) is passed as column arguement

The margins arguement is set to true to check the total sum of all categories created

In [77]:

```
pd.crosstab([df['is_canceled'],df['hotel']], df['arrival_date_year'],margins=True)
```

Out[77]:

	arrival_date_year	2015	2016	2017	All
is_canceled	hotel				
0	City Hotel	7678	22733	15817	46228
	Resort Hotel	6176	13637	9125	28938
1	City Hotel	6004	15407	11691	33102
	Resort Hotel	2138	4930	4054	11122
All		21996	56707	40687	119390

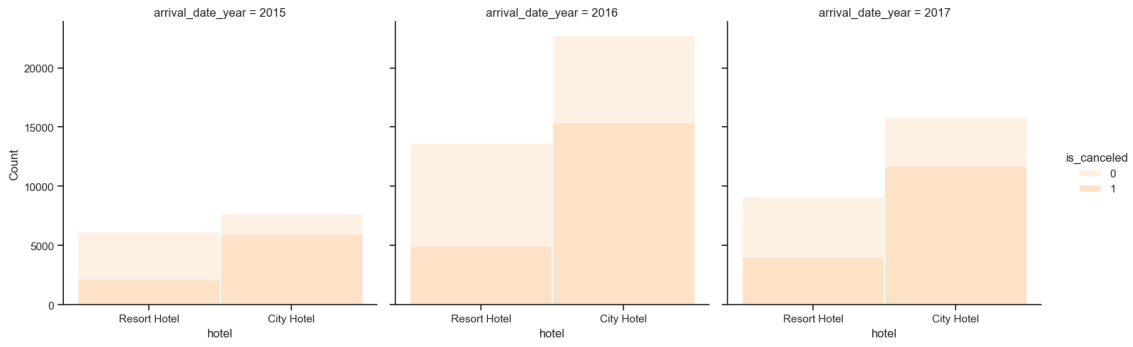
To visualize the above information found through pd.crosstabs, using seasborn displot

sns.set_theme sets aspects of the visual theme for all matplotlib and seaborn plots. Style sets the theme to given arguement, here it is ticks. It can be whitegrid, darkgrid, dark etc. Pallete is a set of colors used to represent data in plots created with the Seaborn

Distribution plots are used here to visualize the data using sns.displot() which is used to create a FacetGrid of histograms for the 'hotel' column in the DataFrame df, with each plot colored according to the is_canceled column, and separated into columns according to the arrival_date_year column.

In [78]:

```
sns.set_theme(style='ticks', palette='Oranges')
sns.displot(data = df, x='hotel', hue ='is_canceled', col='arrival_date_year')
plt.show()
```



INSIGHT :
In all the three years (2015,2016,2017) , City hotel have more cancellation than Resort hotel.
Meanwhile in year 2016 the count of booking cancellation is highest in all the three years

What is the booking ratio between Resort Hotel and City Hotel?

Here we are trying to find the how many hotels booked in Resort Hotel and City Hotel respectively and what is the ratio between the same

Creating a new dataframe named as df_not_cancelled which stores all the values where the cancellation status is No(or 0) i.e. the particular hotel have been book or its booking status is poistive. Then in the next line checking the first five rows of the dataframe to validate if the dataframe has correct values or not.

In [79]:

```
df_not_cancelled = df[df['is_canceled']==0]
df_not_cancelled.head()
```

Out[79]:

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

5 rows × 33 columns

Plotting bar plot to visualize the values. There are two desired categories and their percentage is also calculated in the vals variable so using a bar plot.

In [80]:

```
# storing categories in the cat variable
cat = df_not_cancelled['hotel'].unique()

# storing values in percentage corresponding to each category in the vals variable
vals = df_not_cancelled['hotel'].value_counts() *100 / df_not_cancelled.shape[0]

#plotting bar chart
colors = ['lightsalmon', 'sienna']

# plotting bar plot and giving aruguement to select categories as strings, values c
plt.bar(cat, vals, color= colors)

# Naming x-label
plt.xlabel("Type of hotel")

# Naming y-label
plt.ylabel("Percentage of bookings")

# Naming the title of the bar plot
plt.title("Percentage of booking made by type of hotel")

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

INSIGHT :

There are more bookings in the Resort hotel i.e.around 60% of the total bookings than the City hotel i.e. around 40% of the total bookings.

The booking ratio of the Resort to City hotel is 60:30 or 2:1

What is the percentage of booking for each year?

Here we are trying to find the year wise comparison of the number of bookings in both the hotels and also we categorise this into type of hotels i.e. Year wise booking in City and Resort Hotel.

Using the same dataframe that we made earlier named as df_not_cancelled. Also using value_counts function to calculate the total number of bookings made per year and dividing it by total rows to find the percentage. Storing the calculation in a variable named x for further use.

In [81]:

```
x = df_not_cancelled['arrival_date_year'].value_counts() / df_not_cancelled.shape[0
x
```

Out[81]:

```
2016    0.483862
2017    0.331826
2015    0.184312
Name: arrival_date_year, dtype: float64
```

Plotting bar plot to visualize the values. There are desired year as categories and their percentage is also calculated in the vals variable so using a bar plot.

In [82]:

```
# storing categories in the cat variable
cat = x.keys()

# storing values in percentage corresponding to each category in the vals variable
vals = x.values

# plotting bar plot and giving aruguement to select categories as strings, values c
sns.barplot(x=cat, y=vals, palette="Reds")

# Naming x-label
plt.xlabel("Year of booking")

# Naming y-label
plt.ylabel("Percentage of bookings")

# Naming the title of the bar plot
plt.title("Year wise percentage of booking")

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

INSIGHT :

The number of booking made in 2016 were almost double the number of bookings in 2015, and the number of booking were less than by 15% in 2017.

What is the duration of stay of people in the hotel?

Here we are trying to find the length of duration of people stay in the hotel or for how long people stay in the hotel.

The total variable contains total number of nights people stayed as it calculates the number of occurrences of each possible stays on week nights and stays on weekend nights, and stores the resulting value counts in the x variable.
This next line calculates the value counts of the total Series, which gives you a Pandas Series where the index contains each unique value of total, and the values contain the number of occurrences of each value

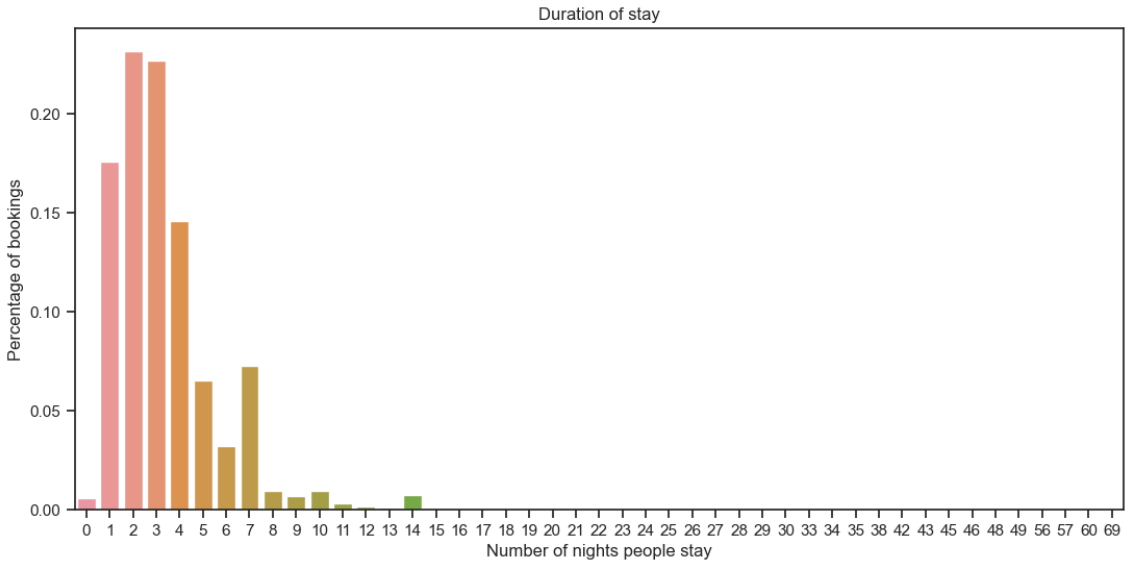
In [83]:

```
total = df['stays_in_week_nights'] + df['stays_in_weekend_nights']

x = total.value_counts()
```

Plotting bar plot to visualize the values. There are desired number of nights for stay as categories and their percentage is also calculated in the vals variable so using a bar plot.

In [84]:

```
# storing categories in the cat variable
cat = x.keys()

# storing values in percentage corresponding to each category in the vals variable
vals = x.values/df.shape[0]

# To set the size of the bar plot
plt.figure(figsize=(13,6))

# plotting bar plot and giving aruguement to select categories as strings, values c
sns.barplot(x=cat, y=vals)

# Naming x-label
plt.xlabel("Number of nights people stay")

# Naming y-label
plt.ylabel("Percentage of bookings")

# Naming the title of the bar plot
plt.title("Duration of stay")

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



INSIGHT :

The lengthiest duration of people stay is one, two, three, four nights. As the number of days increase more than six or seven, the duration of people's stay decreases.

Calculating which week of the year had the maximum and the minimum number of guests using arrival_date_week_number column

This will help us find the most busy and the least busy week of the year

value_counts will count the the number of time unique values are present in the column.

stay_week_number_count is the object storing counts of unique variables from column arrival_date_week_number

In [85]:

```
1
2 stay_week_number_count = df['arrival_date_week_number'].value_counts()
3 busy_week = stay_week_number_count[stay_week_number_count == stay_week_number_count
4 least_busy_week = stay_week_number_count[stay_week_number_count == stay_week_number_
5
6 print("Least busy week with the number of bookings it had = " , least_busy_week)
7 print("Most busy week with the number of bookings it had = " , busy_week)
8
```

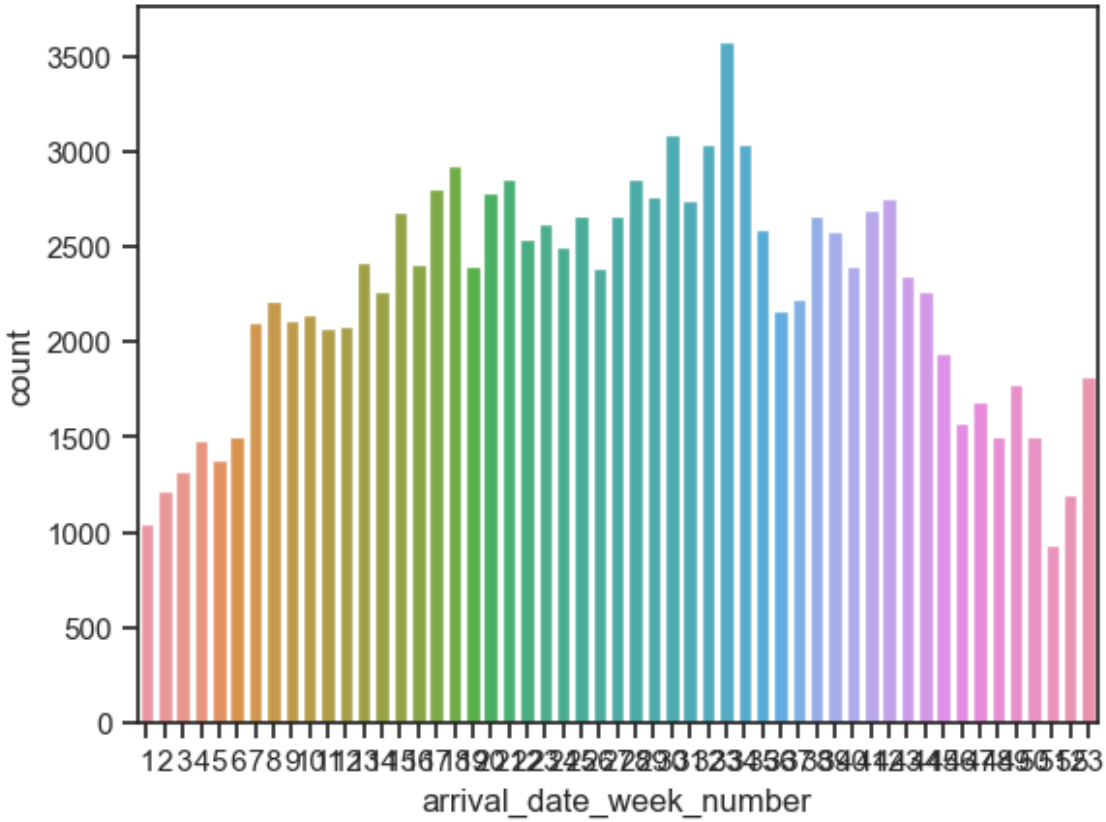
```
Least busy week with the number of bookings it had =  51    933
Name: arrival_date_week_number, dtype: int64
Most busy week with the number of bookings it had =  33    3580
Name: arrival_date_week_number, dtype: int64
```

This told us that 33rd week of the year is the busiest for the hotels and is the with the highest bookings done.

Whereas week with the least number of booking is 51st.

In [86]:

```
fig, axes = plt.subplots(1)
sns.countplot(x=df['arrival_date_week_number'], ax=axes)

plt.show()
```



Calculating most preferred hotel by each of the market segment

Using crosstab function of pandas library

In [87]:

```
#Using pandas library cross tabulation of two columns is done
pd.crosstab(df['market_segment'],df['hotel'],margins = True)
```

Out[87]:

hotel market_segment	City Hotel	Resort Hotel	All
Aviation	237	0	237
Complementary	542	201	743
Corporate	2986	2309	5295
Direct	6093	6513	12606
Groups	13975	5836	19811
Offline TA/TO	16747	7472	24219
Online TA	38748	17729	56477
Undefined	2	0	2
All	79330	40060	119390

The preference of different market segments can be observed in the above frequency tableand you can also see the highest number of bookings are done by which category in each of the hotel type.

Aviation segment goes only for city hotel as per the output.

For every segment, City Hotel got the highest number of bookings so it is the prefferred hotel for each of the segment.

For both of the hotels maximum number of bookings come from online trsvel agent segment.

Analysing the most booked room type by various types of customers

It will be done using cross tab with normalize attribute to provide percentages

In [88]:

```
1 pd.crosstab(df['reserved_room_type'],df['customer_type'], margins = True, normalize
```

Out[88]:

customer_type reserved_room_type	Contract	Group	Transient	Transient-Party	All
A	0.024014	0.003057	0.510495	0.182712	0.720278
B	0.000628	0.000050	0.005335	0.003350	0.009364
C	0.000084	0.000042	0.006935	0.000745	0.007806
D	0.007061	0.001198	0.137532	0.015035	0.160826
E	0.001483	0.000276	0.046645	0.006332	0.054737
F	0.000854	0.000084	0.022305	0.001022	0.024265
G	0.000008	0.000101	0.016392	0.001039	0.017539
H	0.000008	0.000017	0.004808	0.000201	0.005034
L	0.000000	0.000000	0.000050	0.000000	0.000050
P	0.000000	0.000008	0.000092	0.000000	0.000101
All	0.034140	0.004833	0.750591	0.210436	1.000000

Around 72.02% of all the customers go A for room_type and the least go for L room_type

A large percentage of each customer type prefers going for room type A. Contract type customers do not have any data for L and P room type and as per data given their least booked room is G and H.

Group type customers do not have any data for L room and the room booked by them least number of times is the P room.

Transient type customers least bookes room type is L and for the Transient-Party type customers is H and they do not have any data for L and P room type.

The correlation between Number of days the booking was in the waiting list before it was confirmed to the customer and the times when the booking got cancelled

In [89]:

```
1 df['days_in_waiting_list'].corr(df['is_canceled'])
```

Out[89]:

```
0.05418582411778179
```

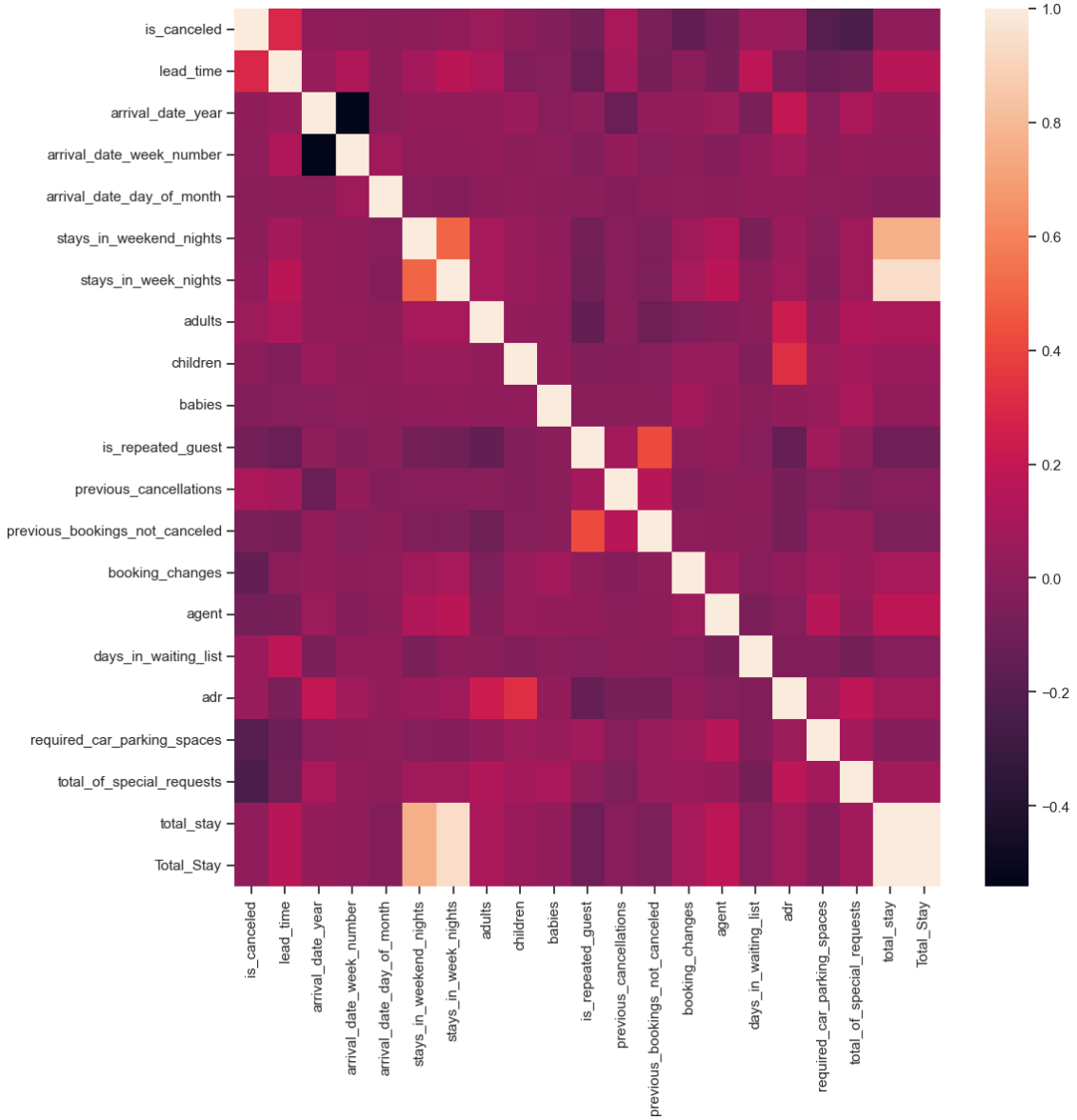
The correlation between the number of days took to confirm booking and the number times the booking got cancelled is positive 0.5.

They are moderately correlated to each other i.e. there is moderate increase in the number of bookings cancelled with the increase in thw waiting days.

Showing Correlation between every data type

In [90]:

```
df_num = df.select_dtypes(exclude="object")

correlation = df_num.corr()

plt.figure(figsize=(12,12))
sns.heatmap(correlation)
plt.show()
```



Showing the correlation between the total stay and the lead time to see in how much advance the guests book their room

In [91]:

```
stay_dur = df['stays_in_week_nights'] + df['stays_in_weekend_nights']
df['lead_time'].corr(stay_dur)
```

Out[91]:

```
0.15716697154821002
```

There is a weak positive correlation showing not much advance bookings done by the customer

In [ ]:

```

```