eda-on-hotel-booking-dataset

March 25, 2024

0.1 Importing Important Libraries

[4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

0.2 Adding Data to the Dataframe

[5]:
```
df1= pd.read_csv("hotel_bookings.csv")
```

0.3 Exploring the Dataset

[6]:
```
df1.head(4)
```

[6]:
```
          hotel  is_canceled  lead_time  arrival_date_year arrival_date_month  \
0  Resort Hotel            0        342               2015               July
1  Resort Hotel            0        737               2015               July
2  Resort Hotel            0          7               2015               July
3  Resort Hotel            0         13               2015               July

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

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

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

```
3  304.0     NaN                    0     Transient  75.0

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

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

[4 rows x 32 columns]
```

[7]:
```
df1.describe()
```

[7]:
```
         is_canceled      lead_time  arrival_date_year  \
count  119390.000000  119390.000000      119390.000000
mean        0.370416     104.011416        2016.156554
std         0.482918     106.863097           0.707476
min         0.000000       0.000000        2015.000000
25%         0.000000      18.000000        2016.000000
50%         0.000000      69.000000        2016.000000
75%         1.000000     160.000000        2017.000000
max         1.000000     737.000000        2017.000000

       arrival_date_week_number  arrival_date_day_of_month  \
count             119390.000000              119390.000000
mean                  27.165173                  15.798241
std                   13.605138                   8.780829
min                    1.000000                   1.000000
25%                   16.000000                   8.000000
50%                   28.000000                  16.000000
75%                   38.000000                  23.000000
max                   53.000000                  31.000000

       stays_in_weekend_nights  stays_in_week_nights         adults  \
count            119390.000000         119390.000000  119390.000000
mean                  0.927599              2.500302       1.856403
std                   0.998613              1.908286       0.579261
min                   0.000000              0.000000       0.000000
25%                   0.000000              1.000000       2.000000
50%                   1.000000              2.000000       2.000000
75%                   2.000000              3.000000       2.000000
max                  19.000000             50.000000      55.000000
```

```
            children         babies  is_repeated_guest  \
count  119386.000000  119390.000000      119390.000000
mean        0.103890       0.007949           0.031912
std         0.398561       0.097436           0.175767
min         0.000000       0.000000           0.000000
25%         0.000000       0.000000           0.000000
50%         0.000000       0.000000           0.000000
75%         0.000000       0.000000           0.000000
max        10.000000      10.000000           1.000000

       previous_cancellations  previous_bookings_not_canceled  \
count           119390.000000                   119390.000000
mean                 0.087118                        0.137097
std                  0.844336                        1.497437
min                  0.000000                        0.000000
25%                  0.000000                        0.000000
50%                  0.000000                        0.000000
75%                  0.000000                        0.000000
max                 26.000000                       72.000000

       booking_changes          agent      company  days_in_waiting_list  \
count    119390.000000  103050.000000  6797.000000         119390.000000
mean          0.221124      86.693382   189.266735              2.321149
std           0.652306     110.774548   131.655015             17.594721
min           0.000000       1.000000     6.000000              0.000000
25%           0.000000       9.000000    62.000000              0.000000
50%           0.000000      14.000000   179.000000              0.000000
75%           0.000000     229.000000   270.000000              0.000000
max          21.000000     535.000000   543.000000            391.000000

                 adr  required_car_parking_spaces  total_of_special_requests
count  119390.000000                119390.000000              119390.000000
mean      101.831122                     0.062518                   0.571363
std        50.535790                     0.245291                   0.792798
min        -6.380000                     0.000000                   0.000000
25%        69.290000                     0.000000                   0.000000
50%        94.575000                     0.000000                   0.000000
75%       126.000000                     0.000000                   1.000000
max      5400.000000                     8.000000                   5.000000
```

[8]:
```
df1.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
```

[9]:
```
# copying this dataset to a new dataset for further analysis.

df=df1.copy()
```

[10]:
```
df.columns
```

[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')
```

0.4 Cleaning The Dataset

[11]:
```
# number of duplicate rows

df.duplicated().value_counts()
```

[11]:
```
False    87396
True     31994
dtype: int64
```

0.4.1 Observation: There are 31994 duplicate rows in our data

[13]:
```
# removing the duplicate rows

df=df.drop_duplicates()
```

[14]:
```
df.shape
```

[14]:
```
(87396, 32)
```

[15]:
```
# Number of Null Values

df.isna().sum().sort_values(ascending=False).reset_index().
 ↪rename(columns={'index':'Columns',0:'Number of Null values'})[:8]
```

[15]:

	Columns	Number of Null values
0	company	82137
1	agent	12193
2	country	452
3	children	4
4	reserved_room_type	0
5	assigned_room_type	0
6	booking_changes	0
7	deposit_type	0

There are mainly 4 columns which have null values. These Columns are : - Company - Agent - Country - Children

1. For company and agent columns I am going to replace the missing values with 0

2. For country column I am going to replace the missing values with object "Others"
3. For children column there are only 4 missing values and I am going to replace the missing values with 0

[16]:
```
# Filling/replacing null values with 0.

df["company"].fillna(0,inplace=True)
df["agent"].fillna(0,inplace=True)
df["children"].fillna(0,inplace=True)
```

[17]:
```
df['country'].fillna('others',inplace=True)
```

[18]:
```
# checking the Null Values

df.isna().sum().sort_values(ascending=False).reset_index().
 ↪rename(columns={'index':'Columns',0:'Number of Null values'})[:8]
```

[18]:
```
                      Columns  Number of Null values
0                       hotel                      0
1                 is_canceled                      0
2          reservation_status                      0
3   total_of_special_requests                      0
4 required_car_parking_spaces                      0
5                         adr                      0
6               customer_type                      0
7        days_in_waiting_list                      0
```

[19]:
```
# Total number of rows where addtion of of adlults,children and babies is 0.
#That simply means no bookings were made in these records.

len(df[df['adults']+df['babies']+df['children']==0])
```

[19]: 166

[20]:
```
# Dropping the records where no bookings were made

df.drop(df[df['adults']+df['babies']+df['children']==0].index,inplace=True)
```

0.4.2 Adding new column of Total People and Total Stay

[21]:
```
df['total_people'] = df['adults']+df['babies']+df['children']
df['total_stay'] = df['stays_in_weekend_nights'] + df['stays_in_week_nights']
```

[22]:
```
df.head()
```

[22]:
```
          hotel  is_canceled  lead_time  arrival_date_year arrival_date_month  \
0  Resort Hotel            0        342               2015               July
```

```
1  Resort Hotel            0        737              2015              July
2  Resort Hotel            0          7              2015              July
3  Resort Hotel            0         13              2015              July
4  Resort Hotel            0         14              2015              July

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

   stays_in_weekend_nights  stays_in_week_nights  adults  …  company  \
0                        0                     0       2  …      0.0
1                        0                     0       2  …      0.0
2                        0                     1       1  …      0.0
3                        0                     1       1  …      0.0
4                        0                     2       2  …      0.0

   days_in_waiting_list customer_type   adr required_car_parking_spaces  \
0                     0     Transient   0.0                           0
1                     0     Transient   0.0                           0
2                     0     Transient  75.0                           0
3                     0     Transient  75.0                           0
4                     0     Transient  98.0                           0

  total_of_special_requests  reservation_status  reservation_status_date  \
0                         0           Check-Out               2015-07-01
1                         0           Check-Out               2015-07-01
2                         0           Check-Out               2015-07-02
3                         0           Check-Out               2015-07-02
4                         1           Check-Out               2015-07-03

   total_people total_stay
0           2.0          0
1           2.0          0
2           1.0          1
3           1.0          1
4           2.0          2

[5 rows x 34 columns]
```

[23]:
```
df.shape
```

[23]: (87230, 34)

[ ]:
```

```

1 Exploratory Data Analysis (EDA)

2 Univariate Analysis

2.1 1. What kind of hotels are most popular among visitors?

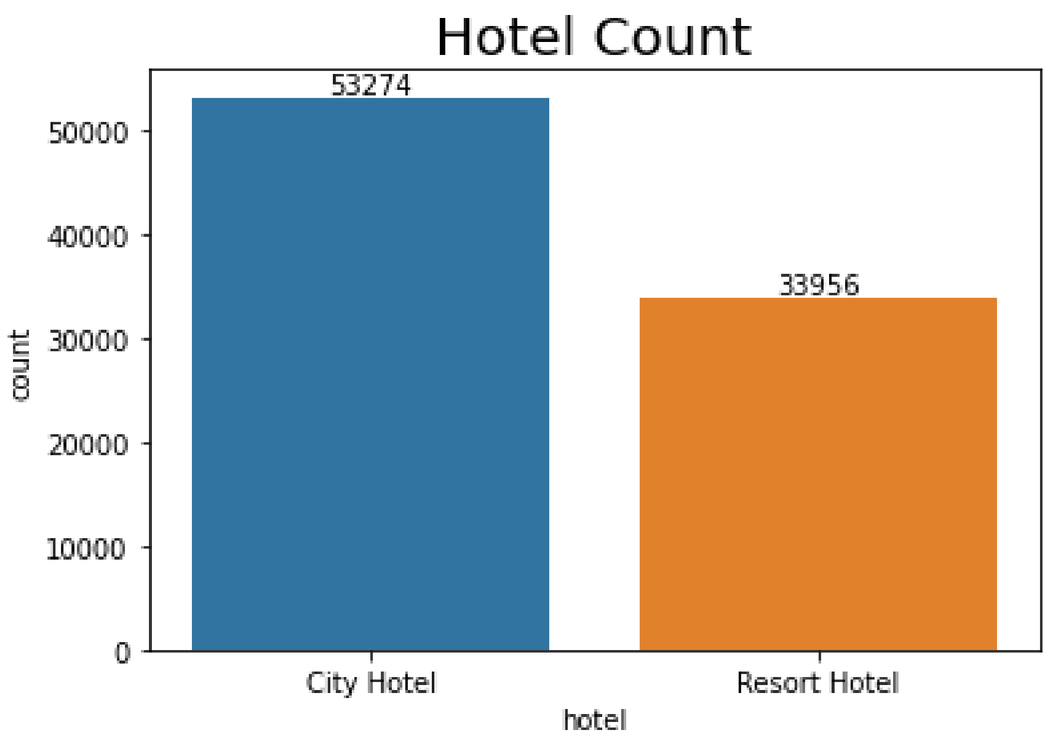
[24]:
```
hotel_count = df.groupby(['hotel'])['hotel'].agg({'count'}).reset_index()
hotel_count
```

[24]:

	hotel	count
0	City Hotel	53274
1	Resort Hotel	33956

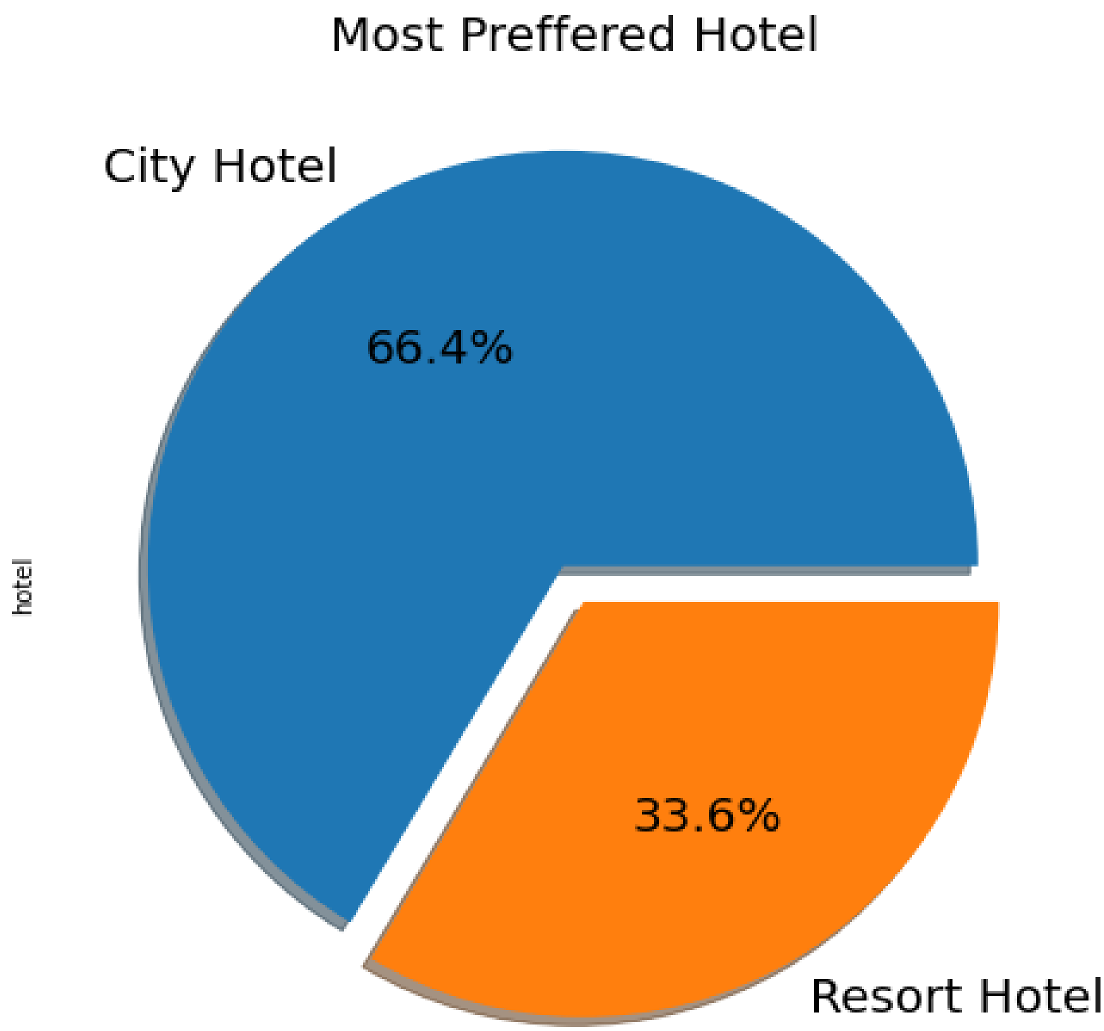
[83]:
```
ax=sns.barplot(x="hotel",y="count",data=hotel_count)
ax.bar_label(ax.containers[0])

plt.title("Hotel Count",fontsize=20)
```

[83]: Text(0.5, 1.0, 'Hotel Count')



[86]:
```
df1['hotel'].value_counts().plot.pie(explode=[0.05, 0.05], autopct='%1.1f%%',␣
↪shadow=True, figsize=(10,8),fontsize=20)
plt.title('Most Reserved Hotel',fontsize=20)
```

[86]: Text(0.5, 1.0, 'Most Preffered Hotel')



2.1.1 Observation: Here we can clearly see that City Hotel is most preferred hotel by the visitors

[ ]:

2.2 2. Which agent secured the most bookings?

[26]:
```
##  highest bookings made by agents
```

```
highest_bookings = df.groupby(['agent'])['agent'].agg({'count'}).
  ↪rename(columns={'count': "Most_Bookings" }).
  ↪sort_values(by='Most_Bookings',ascending=False).reset_index()
```

[27]:
```
# As we previously replaced the null values with 0 and indicates no bookings.
# So we have to drop this 0 agent row

highest_bookings.drop(highest_bookings[highest_bookings['agent']==0].
  ↪index,inplace=True)

highest_bookings.head()
```

[27]:

	agent	Most_Bookings
0	9.0	28721
1	240.0	13028
3	14.0	3342
4	7.0	3294
5	250.0	2779

[28]:
```
#top 10 agents

top_ten_highest_bookings=highest_bookings[:10]

top_ten_highest_bookings
```

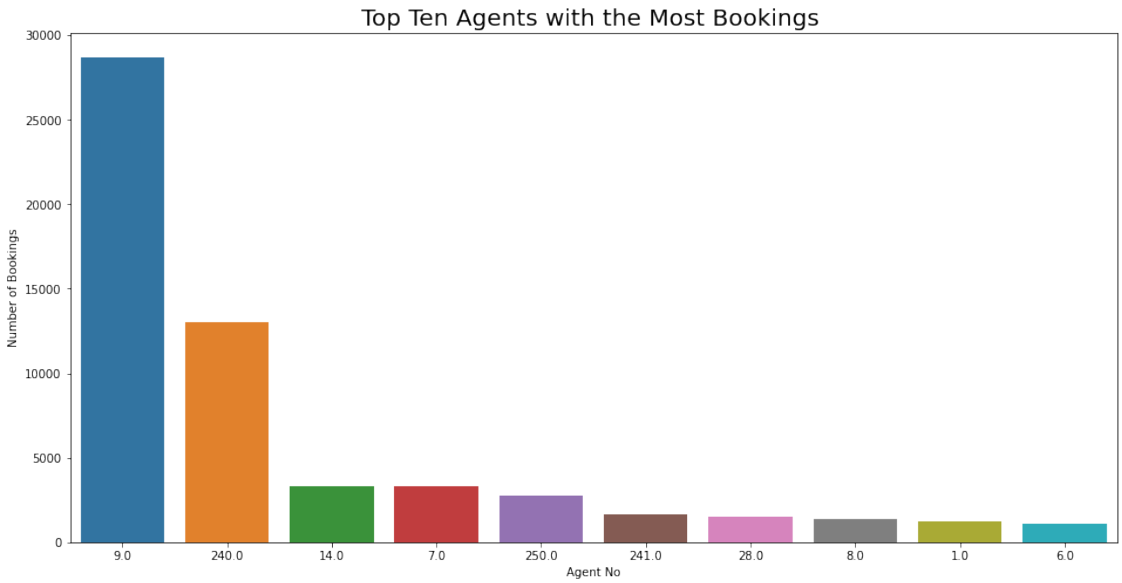
[28]:

	agent	Most_Bookings
0	9.0	28721
1	240.0	13028
3	14.0	3342
4	7.0	3294
5	250.0	2779
6	241.0	1644
7	28.0	1493
8	8.0	1383
9	1.0	1228
10	6.0	1117

[29]:
```
#plotting the graph

plt.figure(figsize=(16,8))

sns.
  ↪barplot(x='agent',y="Most_Bookings",data=top_ten_highest_bookings,order=top_ten_highest_boo

plt.xlabel('Agent No')

plt.ylabel('Number of Bookings')
```

```
plt.title("Top Ten Agents with the Most Bookings ",fontsize=20)
```

[29]: Text(0.5, 1.0, 'Top Ten Agents with the Most Bookings\xa0')



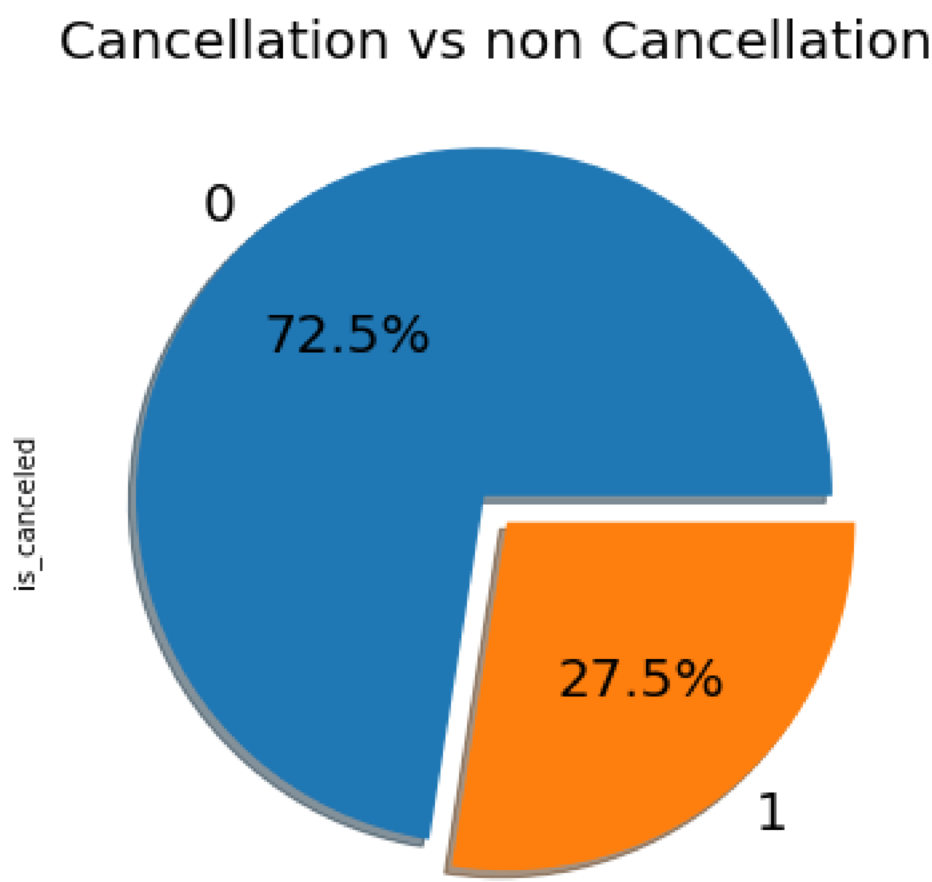
2.2.1 Observation: Top Agent is Agent 9 who has Most Number of Bookings

[ ]:

2.3 3. Cancelation Percentage:

[30]:
```
df['is_canceled'].value_counts().plot.pie(explode=[0.05, 0.05],autopct='%0.
↪1f%%',shadow=True, figsize=(6,6),fontsize=20)

plt.title("Cancellation vs non Cancellation",fontsize=20)
```

[30]: Text(0.5, 1.0, 'Cancellation vs non Cancellation')



** 0=Not cancled 1=Canceled

2.3.1 Observation: 27.5 % of the bookings were cancelled.

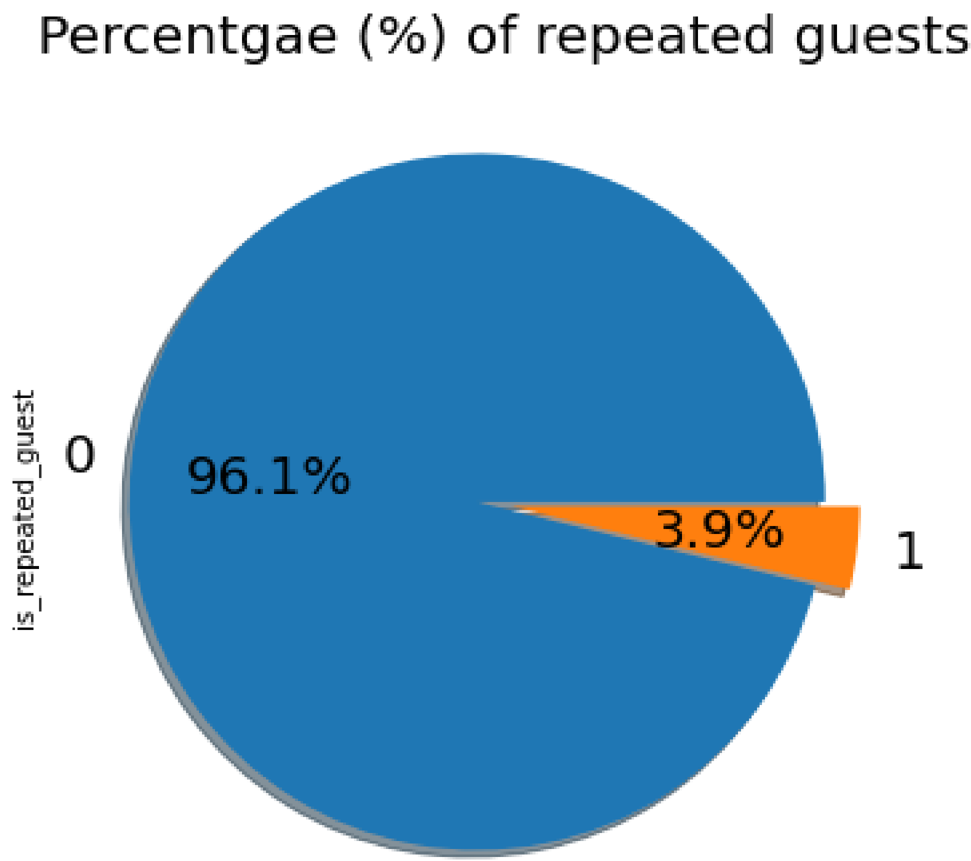
[ ]:

```

```

2.4 4. What is the Percentage of repeated guests?

[31]:
```
df['is_repeated_guest'].value_counts().plot.pie(explode=(0.05,0.05),autopct='%0.
↪1f%%',shadow=True,figsize=(6,6),fontsize=20)

plt.title(" Percentgae (%) of repeated guests",fontsize=20)
```

[31]:
```
Text(0.5, 1.0, ' Percentgae (%) of repeated guests')
```



2.4.1 Observation: Only 3.90% of visitors are repeats, which is quite small.

Management needs to listen to customer feedback in order to improve services and keep customers.

[ ]:

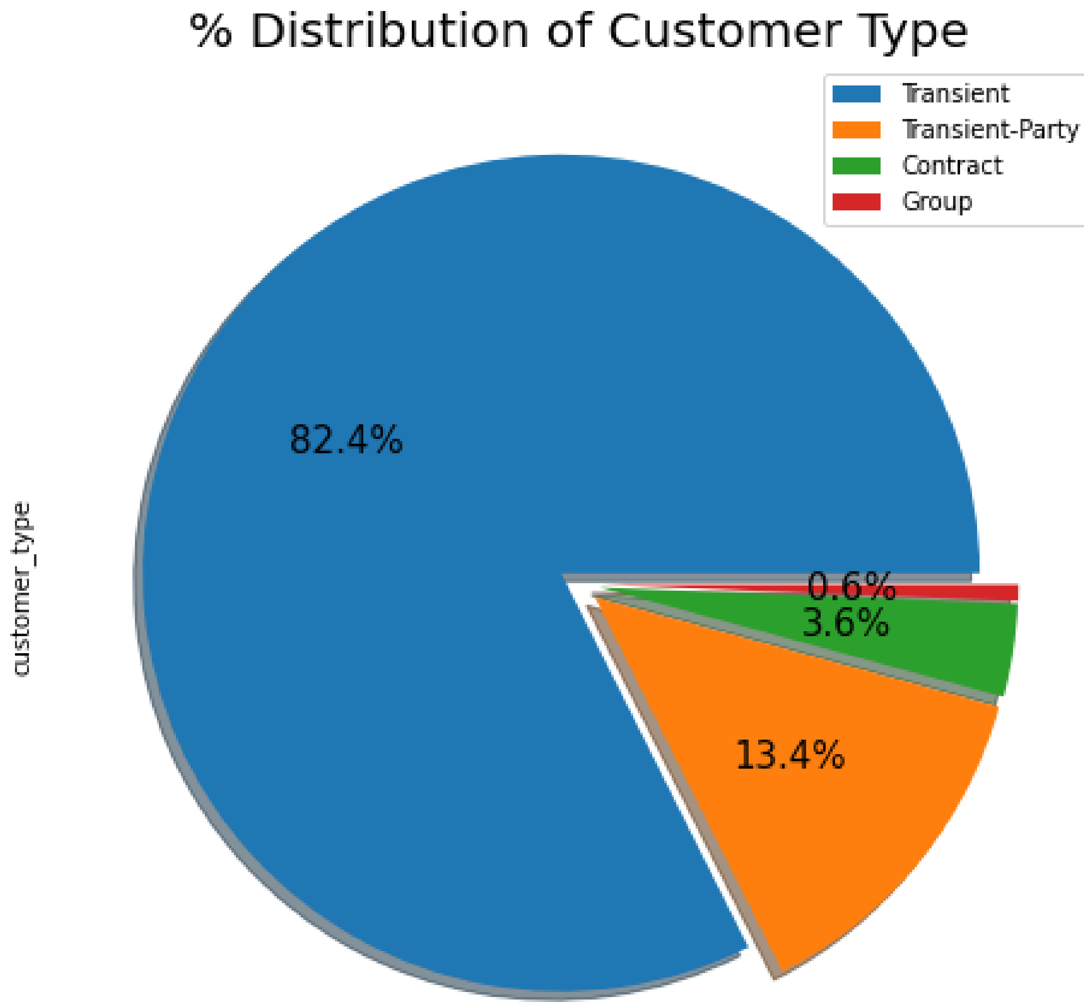
```
```

2.5 5. What is the "Customer Type" distribution in percentages?

[32]:

```
df['customer_type'].value_counts().plot.pie(explode=[0.
↪05]*4,shadow=True,autopct='%1.1f%%',figsize=(8,8),fontsize=15,labels=None)

labels=df['customer_type'].value_counts().index

plt.legend(loc='upper right', labels=labels)

plt.title('% Distribution of Customer Type',fontsize=20)
```

[32]: Text(0.5, 1.0, '% Distribution of Customer Type')



1. Contract : when the booking has an allotment or other type of contract associated to it
2. Group : when the booking is associated to a group
3. Transient : when the booking is not part of a group or contract, and is not associated to other transient booking
4. Transient-party : when the booking is transient, but is associated to at least other transient booking

2.5.1 Observation: The percentage of transient customers is higher at 82.4%. A very small fraction of Bookings are connected to the Group.

[ ]:

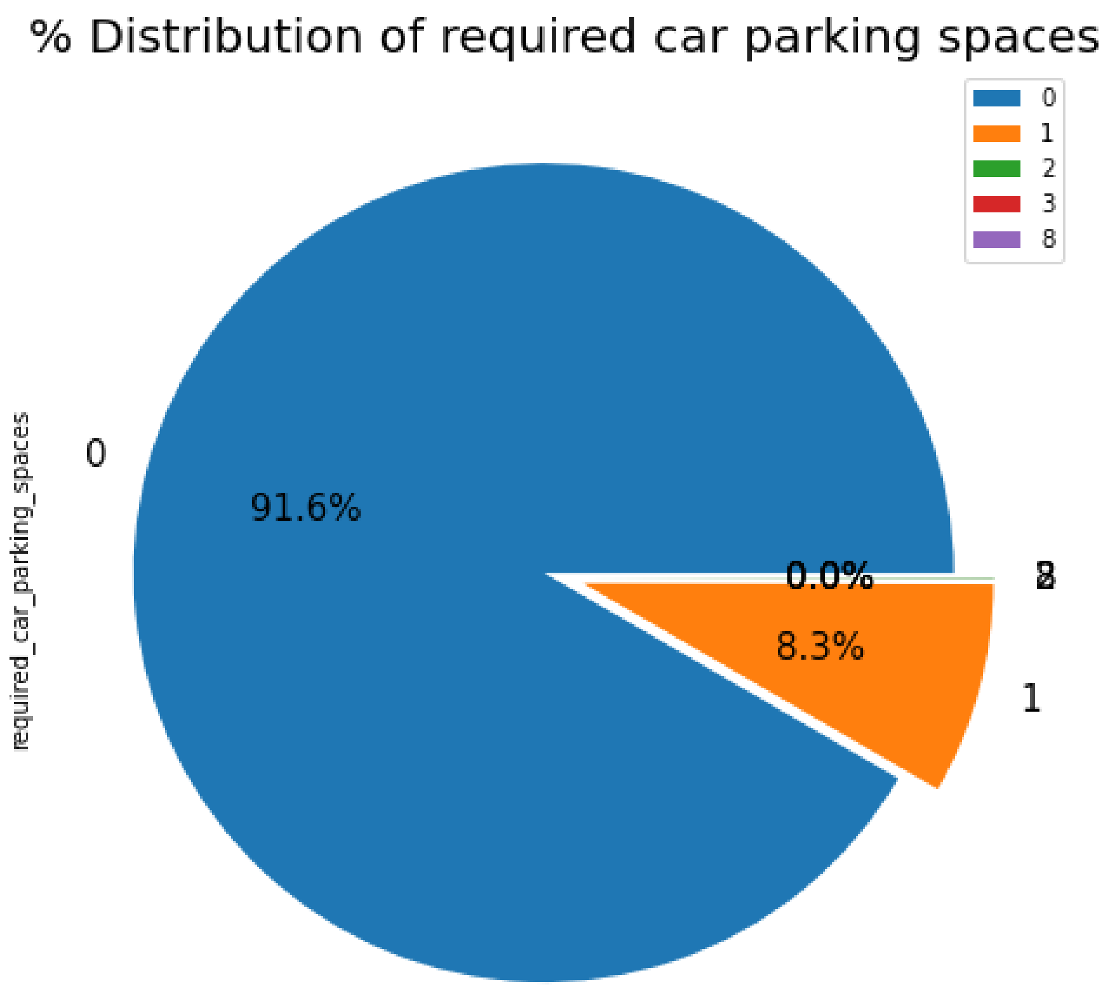
```

```

2.6 6. Percentage distribution of required car parking spaces?

[33]:

```
df['required_car_parking_spaces'].value_counts().plot.pie(explode=[0.05]*5,␣
↪autopct='%1.1f%%',shadow=False,figsize=(8,8),fontsize=15)

labels=df['required_car_parking_spaces'].value_counts().index

plt.legend(loc='upper right', labels=labels)

plt.title('% Distribution of required car parking spaces',fontsize=20)
```

[33]: Text(0.5, 1.0, '% Distribution of required car parking spaces')



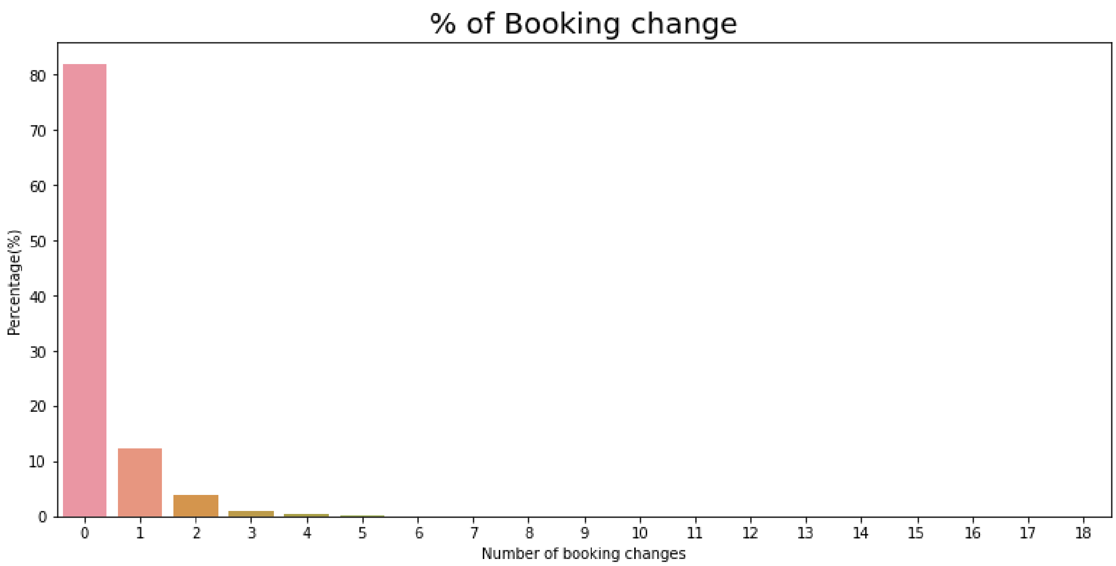
2.6.1 Observation: The parking space was not needed by 91.6% of the visitors. 8.3% of visitors only needed one parking space.

2.7 7. What percentage of customer-made changes to bookings?

[90]:
```
booking_changes=df["booking_changes"].value_counts().reset_index().
 ↪rename(columns={'index': "number_booking_changes",'booking_changes':
 ↪'Counts'})

#Plotting the graph

plt.figure(figsize=(13,6))
```

```
sns.
↪barplot(x=booking_changes['number_booking_changes'],y=booking_changes['Counts']*100/
↪df.shape[0])

plt.title("% of Booking change",fontsize=20)

plt.xlabel('Number of booking changes')

plt.ylabel('Percentage(%)')
```

[90]: Text(0, 0.5, 'Percentage(%)')



** - 0 = 0 changes made in the booking - 1 = 1 changes made in the booking - 2 = 2 changes made in the booking

2.7.1 Observations: Above 80% of the bookings were not changed by guests.

[ ]:

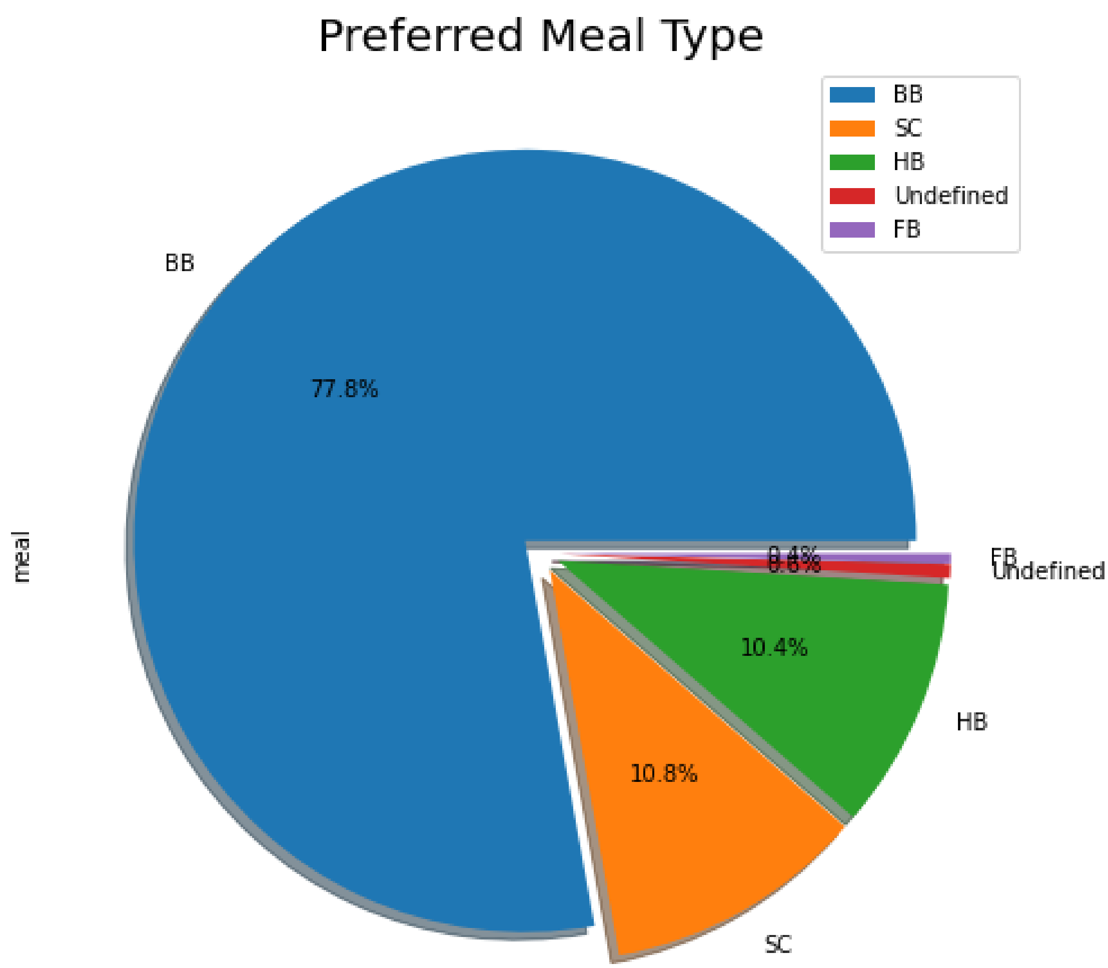
```

```

2.8 8. Which cuisine is most popular among the visitors?

[35]:
```
df['meal'].value_counts().plot.pie(explode=[0.05, 0.05,0.05,0.05,0.05],␣
↪autopct='%1.1f%%', shadow=True, figsize=(8,8),fontsize=10)

labels=df['meal'].value_counts().index

plt.legend(loc='upper right', labels=labels)
```

```
plt.title("Preferred Meal Type",fontsize=20)
```

[35]: Text(0.5, 1.0, 'Preferred Meal Type')



** - BB - (Bed and Breakfast) - SC- (Self Catering) - HB- (Half Board) - FB- (Full Board)

2.8.1 Observations :

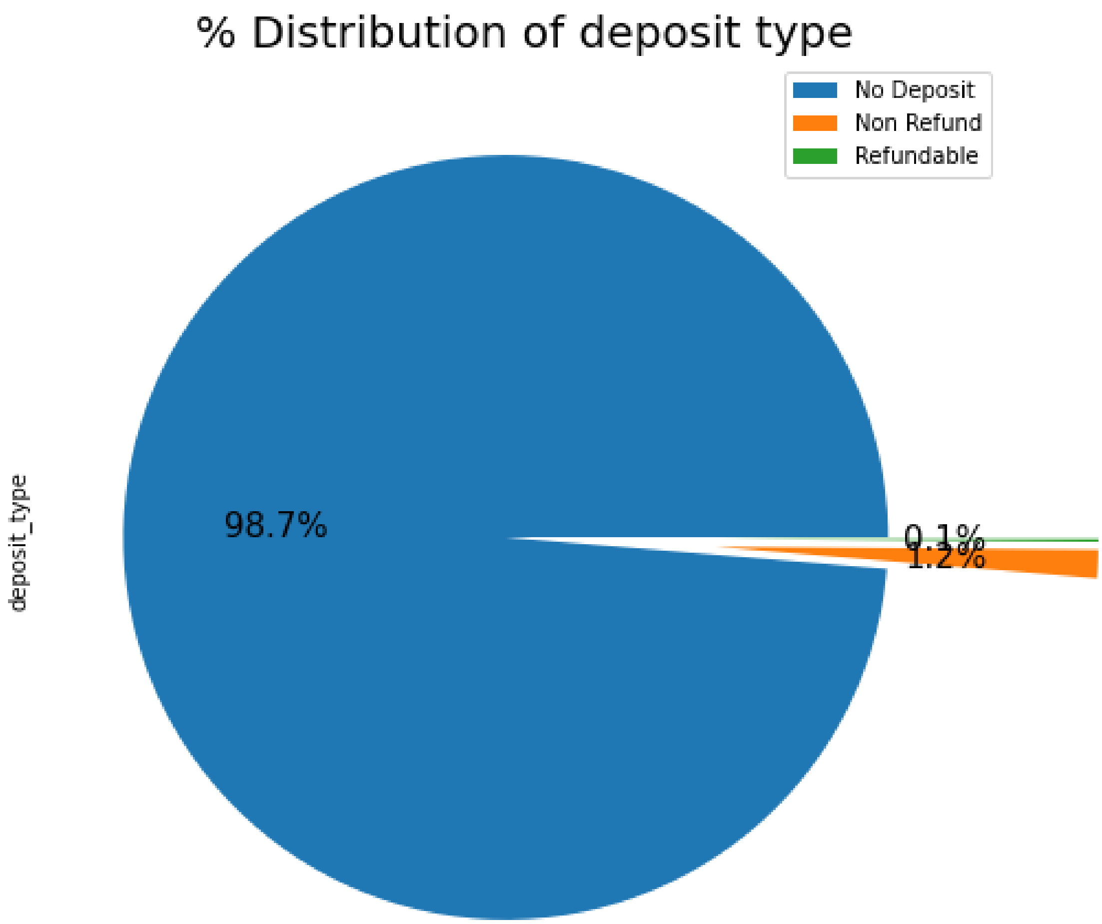
1. Consequently, bed and breakfast(BB) is the most popular sort of meal among the visitors. 2. Almost Equally desirable are HB- (Half Board) and SC- (Self Catering).

[ ]:

2.9 9. What is the Deposite type's percentage distribution?

[36]:
```
df['deposit_type'].value_counts().plot.pie(explode=(0.05,0.5,0.5),autopct='%1.
 ↪1f%%',shadow=False,figsize=(8,8),fontsize=15,labels=None)

plt.title("% Distribution of deposit type",fontsize=20)

labels=df['deposit_type'].value_counts().index

plt.legend(loc='upper right', labels=labels)
```

[36]:
```
<matplotlib.legend.Legend at 0x1b42e87c910>
```



2.9.1 Observation:

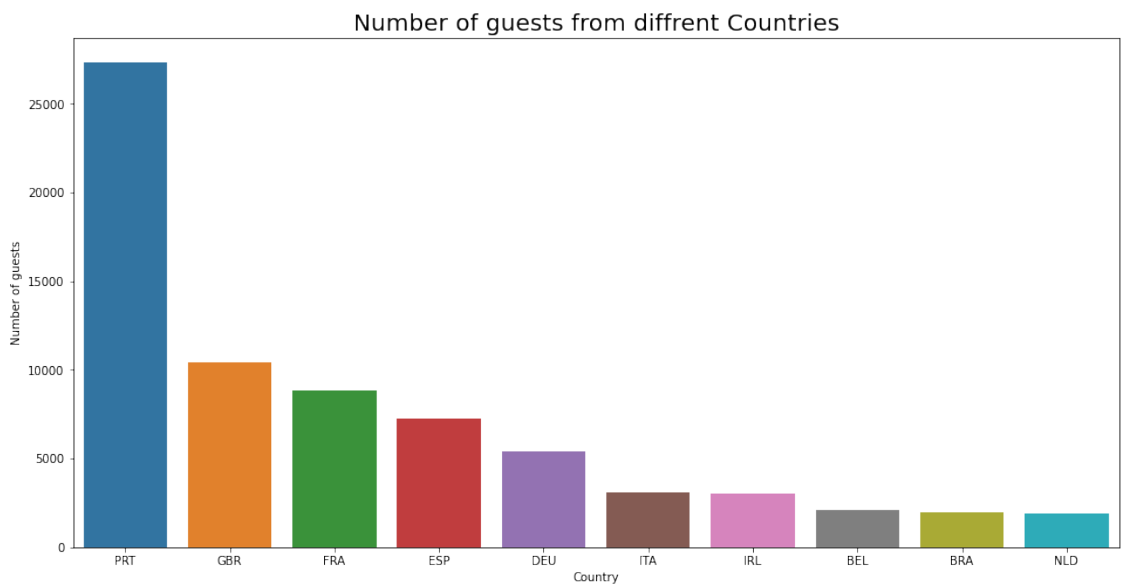
The majority of visitors almost 98.7% prefer "No deposit" types of deposits which means the customer made no deposit to guarantee the booking.

[ ]:

2.10 10. Top ten countries from which the most visitors arrive

[37]:
```
top_10_country=df['country'].value_counts().reset_index().
 ↪rename(columns={'index': 'country','country': 'count of guests'})[:10]

#plotting the graph
plt.figure(figsize=(16,8))

sns.barplot(x='country',y='count of guests',data=top_10_country)

plt.xlabel('Country')

plt.ylabel('Number of guests')

plt.title("Number of guests from diffrent Countries",fontsize=20)
```

[37]:
```
Text(0.5, 1.0, 'Number of guests from diffrent Countries')
```



[38]:
```
# plotting on map
import folium
import plotly.express as px
```

[39]:
```
basemap = folium.Map()
guests_map = px.choropleth(top_10_country, locations =␣
 ↪top_10_country['country'],color = top_10_country['count of guests'],␣
 ↪hover_name = top_10_country['country'])
guests_map.show()
```

Top 10 Countries are :

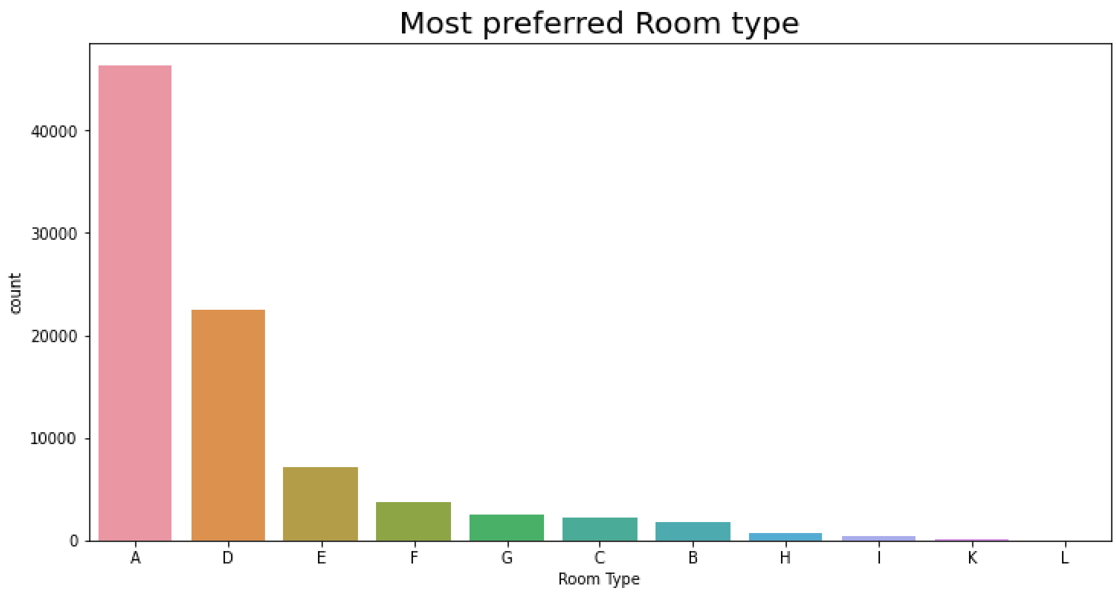
```
PRT- Portugal
GBR- United Kingdom
FRA- France
ESP- Spain
DEU - Germany
ITA -Itlay
IRL - Ireland
BEL -Belgium
BRA -Brazil
NLD-Netherlands
```

2.10.1 Observation: Most of the visitors are coming from portugal

2.11 11. Which kind of room is most popular among customers?

[40]:
```
plt.figure(figsize=(12,6))

sns.countplot(x='assigned_room_type',data=df,order=df['assigned_room_type'].
 ↪value_counts().index)

plt.xlabel('Room Type')

plt.title("Most preferred Room type",fontsize=20)
```

[40]:
```
Text(0.5, 1.0, 'Most preferred Room type')
```



2.11.1 Observation: The most preferred Room type is "A".

[ ]:

```

```

2.12 12. Which month received the most reservations?

[41]:

```
#groupby arrival_date_month and taking the hotel count
bookings_by_months=df.groupby(['arrival_date_month'])['hotel'].count().
↪reset_index().rename(columns={'hotel':"Counts"})

# Create list of months in order
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
↪'August', 'September', 'October', 'November', 'December']

# developing a df that will maintain the above months list's order without␣
↪altering its values.
bookings_by_months['arrival_date_month']=pd.
↪Categorical(bookings_by_months['arrival_date_month'],categories=months,ordered=True)

#sorting by arrival_date_month
bookings_by_months=bookings_by_months.sort_values('arrival_date_month')

bookings_by_months
```

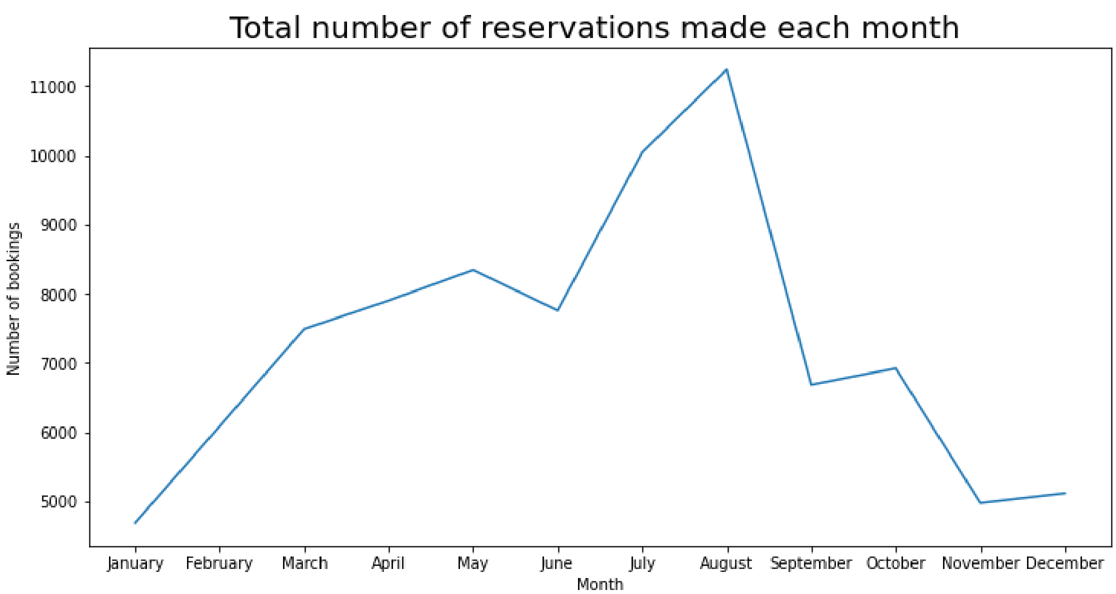
[41]:

	arrival_date_month	Counts
4	January	4685
3	February	6083

```
7              March     7489
0              April     7900
8                May     8344
6               June     7756
5               July    10043
1             August    11242
11         September     6682
10           October     6921
9           November     4973
2           December     5112
```

[42]:
```python
# Plotting the graph
plt.figure(figsize=(12,6))

sns.lineplot(x="arrival_date_month",y="Counts",data=bookings_by_months)

plt.title('Total number of reservations made each month ',fontsize=20)

plt.xlabel('Month')

plt.ylabel('Number of bookings')
```

[42]:
```
Text(0, 0.5, 'Number of bookings')
```



2.12.1 Observations:

The months with the most bookings were July and August. Bookings may have been made in anticipation of summer vacations.

[ ]:

```

```

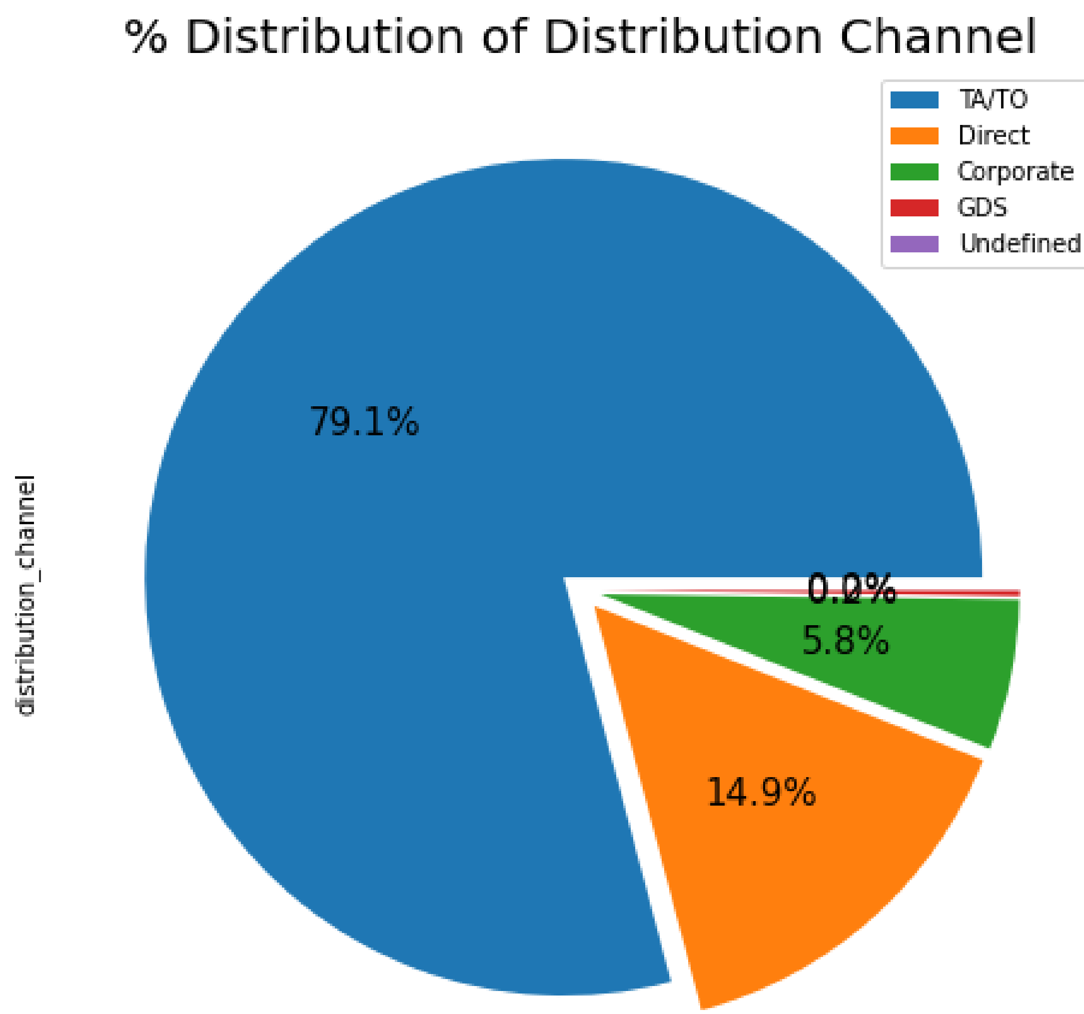
2.13 13. Which distribution method is most popular for hotel reservations?

[100]:

```
df['distribution_channel'].value_counts().plot.pie(explode=(0.05,0.05,0.05,0.
 ↪05,0.05),autopct='%1.
 ↪1f%%',shadow=False,figsize=(8,8),fontsize=15,labels=None)



plt.title("% Distribution of Distribution Channel",fontsize=20)

labels=df['distribution_channel'].value_counts().index

plt.legend(loc='upper right', labels=labels)
```

[100]:

```
<matplotlib.legend.Legend at 0x1b441954520>
```



- Corporate- These are corporate hotel booing companies which makes bookings possible.
- GDS-A GDS is a worldwide conduit between travel bookers and suppliers, such as hotels and other accommodation providers. It communicates live product, price and availability data to travel agents and online booking engines, and allows for automated transactions.
- Direct- means that bookings are directly made with the respective hotels
- TA/TO- means that booings are made through travel agents or travel operators.
- Undefined- Bookings are undefined. may be customers made their bookings on arrival.

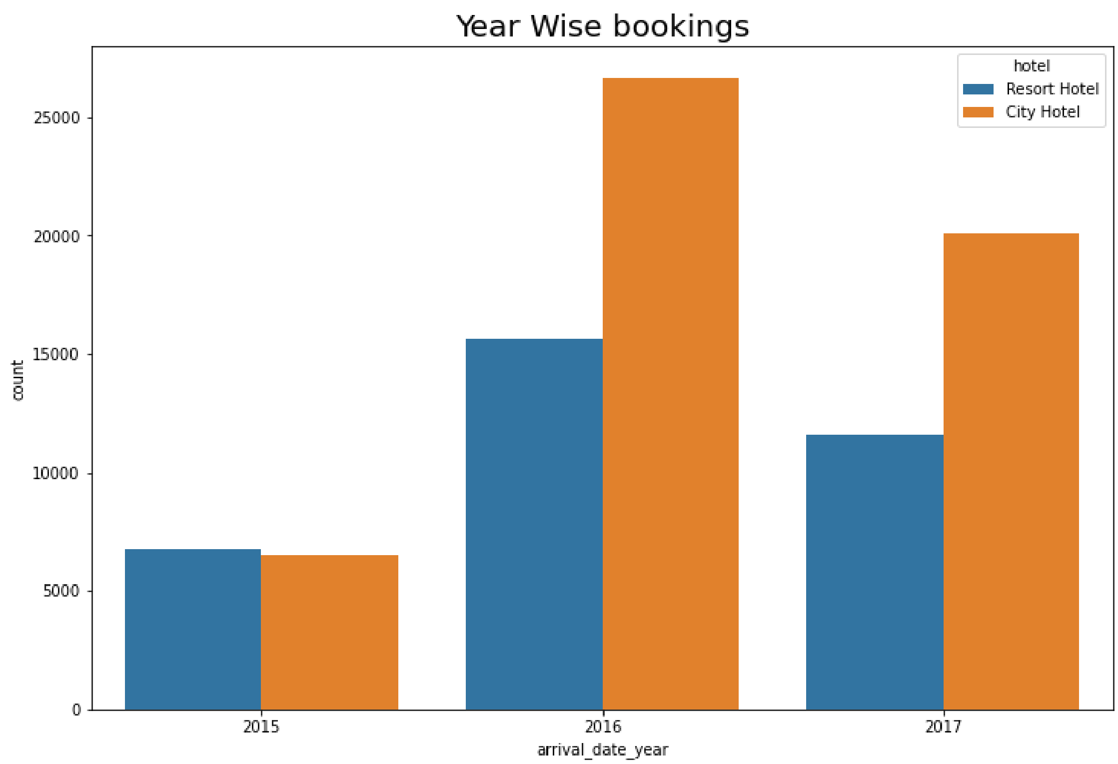
2.13.1 Observations: 'TA/TO' is mostly used for booking hoetls.

[ ]:

2.14 14. In which year were the most reservations made?

[44]:
```
plt.figure(figsize=(12,8))

sns.countplot(x=df['arrival_date_year'],hue=df['hotel'])

plt.title("Year Wise bookings",fontsize="20")
```

[44]: Text(0.5, 1.0, 'Year Wise bookings')



2.14.1 Observations :

1. City hotels had the most of the bookings 2. 2016 had the highest bookings and 2015 had the lowest bookings

[ ]:

3 15. How many guests are not alloted with the same room type which was reserved by them?

[45]:
```
#creating a function If the allocated room type and the reserved room type are␣
↪the same.

def same_room_allotment(x):
    if x['reserved_room_type'] != x['assigned_room_type']:
        return 1
    else:
        return 0
```

[46]:
```
# adding new column

df['Same_room_alloted_or_not']=df.apply(lambda x: same_room_allotment(x),axis=1)
```

[47]:
```
df['Same_room_alloted_or_not'].value_counts()
```

[47]:
```
0    74240
1    12990
Name: Same_room_alloted_or_not, dtype: int64
```

3.0.1 Observations : The majority of the time, guests receive the exact accommodation they have reserved.

[ ]:

[ ]:

[ ]:

3.1 1. What kind of hotel has the highest ADR?

[49]:
```
#grouping by hotel adr
highest_adr=df.groupby(['hotel'])['adr'].mean().reset_index()
highest_adr
```

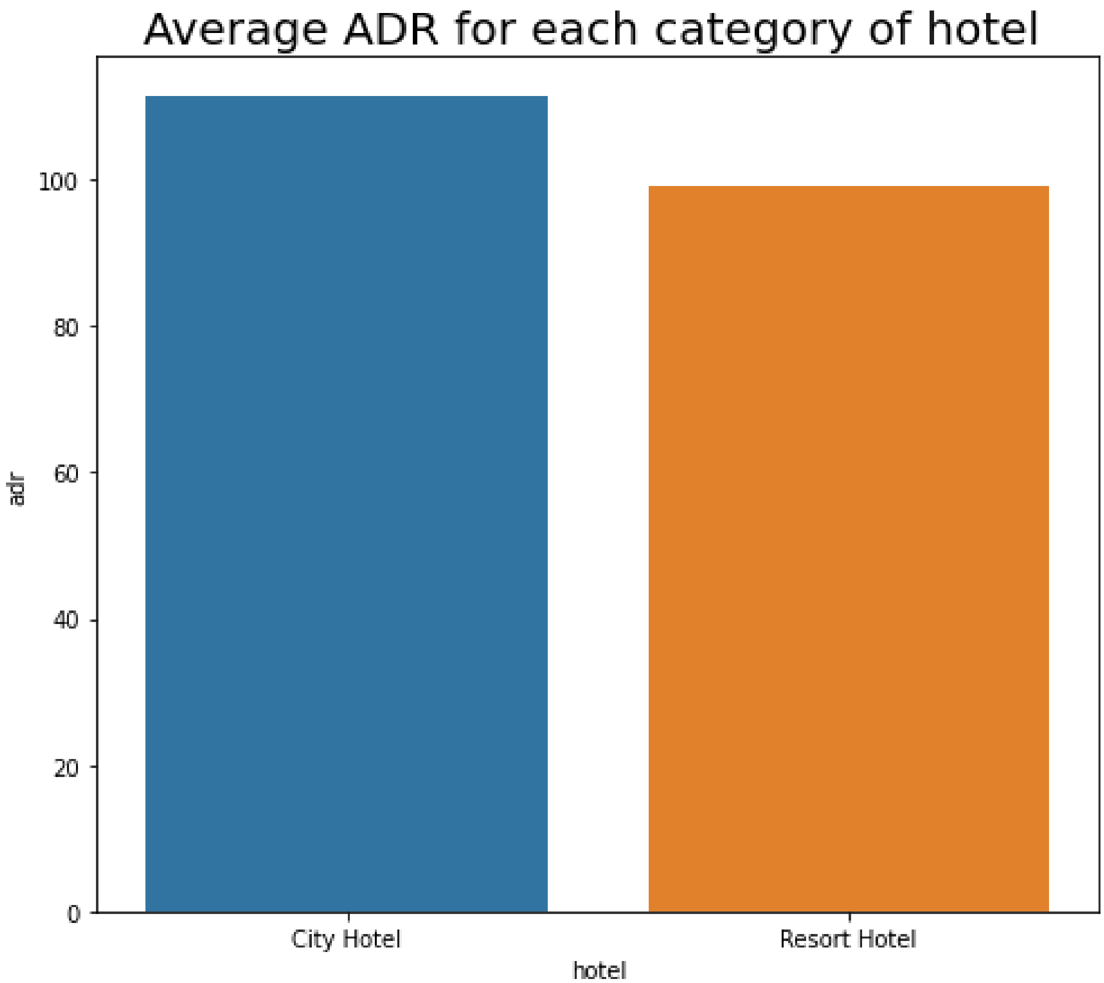
[49]:

	hotel	adr
0	City Hotel	111.271969
1	Resort Hotel	99.059517

[50]:
```
# plotting the graph

plt.figure(figsize=(8,7))

sns.barplot(x=highest_adr['hotel'],y=highest_adr['adr'])
```

```
plt.title("Average ADR for each Hotel Type ",fontsize=20)
```

[50]: Text(0.5, 1.0, 'Average ADR for each category of hotel ')



3.1.1 Observation :

The highest ADR is at the City Hotel. This indicates that city hotels make more money than resort hotels.

[ ]:

3.2 2. Which hotel category has the more average lead time?

[51]:
```
#group by hotel

group_by_hotel=df.groupby('hotel')
```

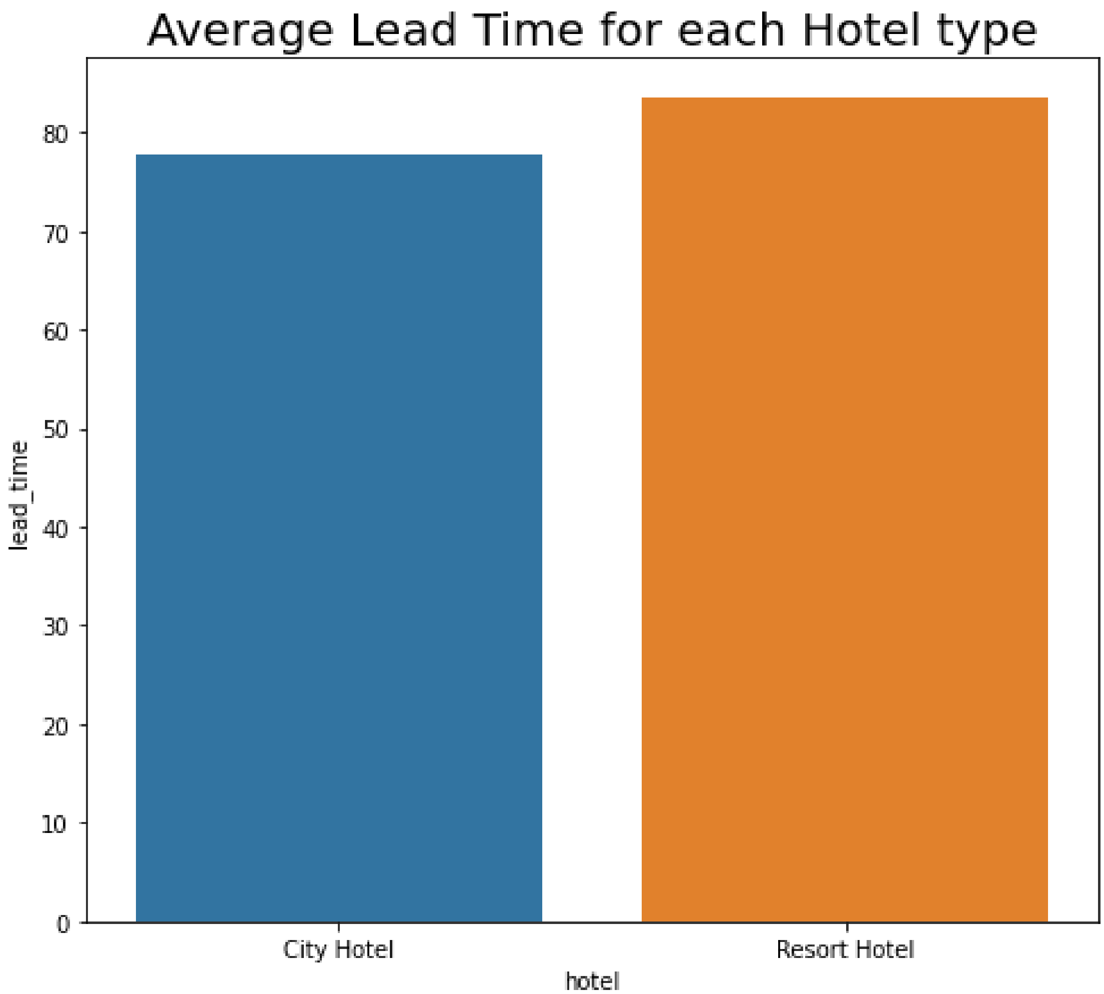
[52]:
```
avg_lead_time=group_by_hotel['lead_time'].mean().reset_index()
avg_lead_time
```

[52]:

	hotel	lead_time
0	City Hotel	77.793257
1	Resort Hotel	83.387737

[53]:
```
#plotting the graph
plt.figure(figsize=(8,7))

sns.barplot(x=avg_lead_time['hotel'],y=avg_lead_time['lead_time'])

plt.title("Average Lead Time for each Hotel type",fontsize=20)
```

[53]:
```
Text(0.5, 1.0, 'Average Lead Time for each Hotel type')
```



3.2.1 Observations :

Hotel resorts have a slightly longer average lead time. Customers must make very early travel plans as a result.

3.3 3. Which hotel experiences the highest rate of cancellations?

booking canceled=1 booking not canceled= 0

[54]:
```
# creating new DataFrame where bookings are cancelled.

canceled_df=df[df['is_canceled']==1]

# Grouping by hotel
canceled_df=canceled_df.groupby('hotel').size().reset_index().rename(columns={0:
 ↪ "no_of_cancelled_bookings"})
```

```
# adding 'total booking column for calculating the percentage.
canceled_df['total_booikngs']=df.groupby('hotel').size().reset_index().
 ↪rename(columns={0:"total_bookings"}).drop('hotel',axis=1)
canceled_df
```

[54]:

	hotel	no_of_cancelled_bookings	total_booikngs
0	City Hotel	16035	53274
1	Resort Hotel	7974	33956

[102]:

```
#plotting the barchat
plt.figure(figsize=(8,7))
sns.
 ↪barplot(x=canceled_df['hotel'],y=100*canceled_df['no_of_cancelled_bookings']/
 ↪canceled_df['total_booikngs'])

plt.xlabel('Hotel type')
plt.ylabel('Percentage(%)')
plt.title("Percentage of booking cancellation for each Hotel type",fontsize=20)
```

[102]: Text(0.5, 1.0, 'Percentage of booking cancellation for each Hotel type')

3.3.1 Observation :

City hotel experiences the highest rate of cancellations.

[ ]:

```

```

3.4 4. Which hotel has longer waiting time?

[56]:

```
#groupping by hoetl and takin avg of days in waiting list
waiting_time_df=df.groupby('hotel')['days_in_waiting_list'].mean().reset_index()

waiting_time_df
```

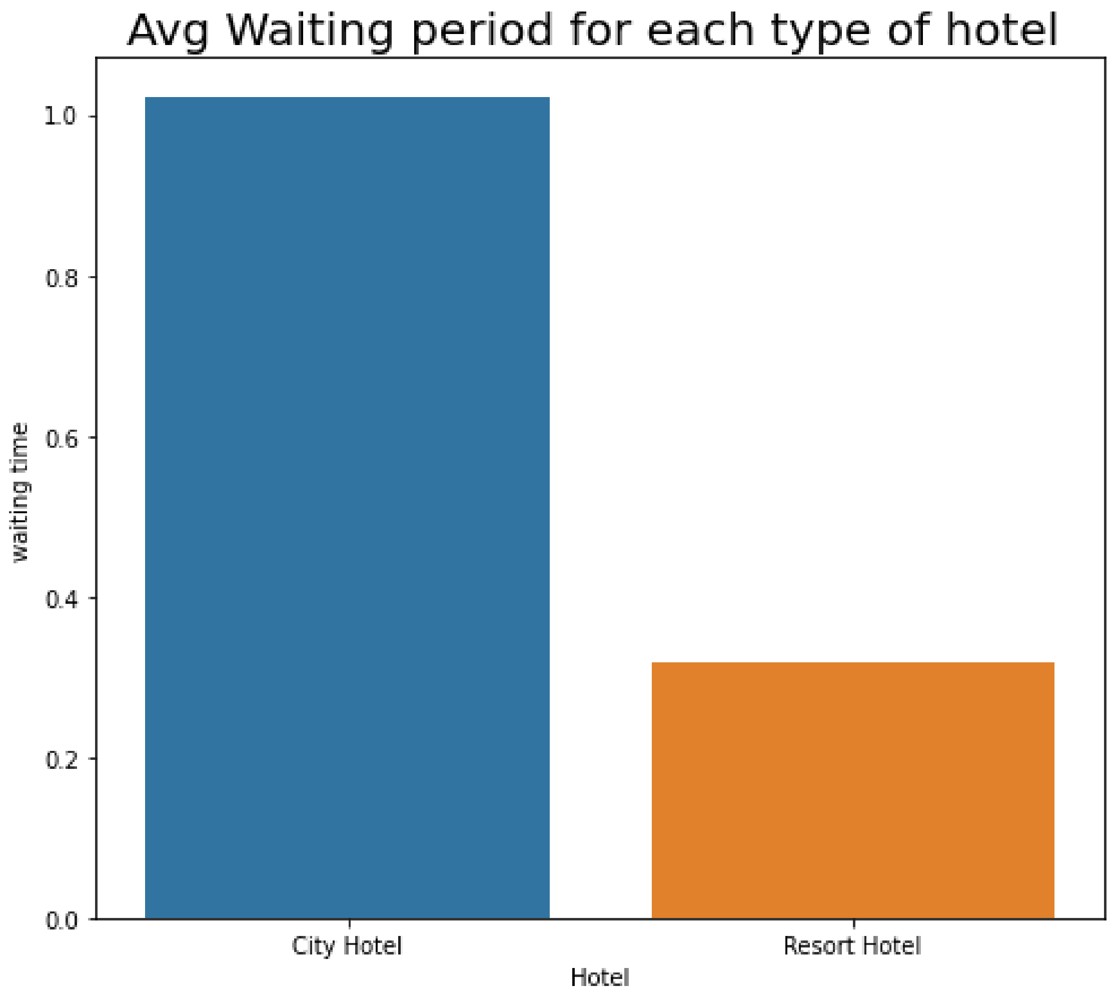
[56]:

	hotel	days_in_waiting_list
0	City Hotel	1.020066
1	Resort Hotel	0.316763

[57]:

```
#plotting the graaph
plt.figure(figsize=(8,7))

sns.
 ↪barplot(x=waiting_time_df['hotel'],y=waiting_time_df['days_in_waiting_list'])


plt.xlabel('Hotel')
plt.ylabel('waiting time')
plt.title("Avg Waiting period for each type of hotel ",fontsize=20)
```

[57]: Text(0.5, 1.0, 'Avg Waiting period for each type of hotel ')



3.4.1 Observation :

Therefore, there is a lengthier wait time at city hotels than at resort hotels. As a result, we can conclude that city hotels are significantly busier than resort hotels.

[ ]:
```

```

3.5 5.Which hotels receive the most return visitors?

repeated guest=1

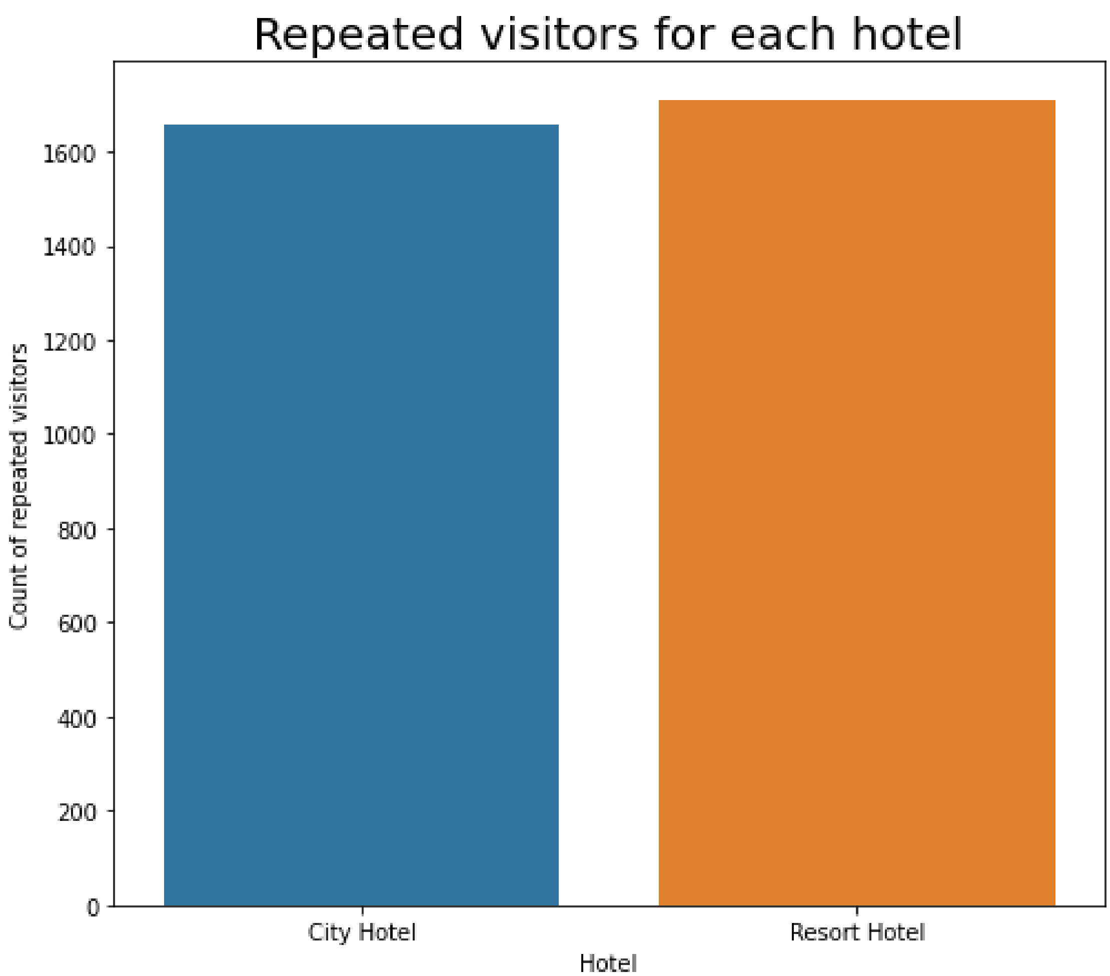
not repeated guest=0

[58]:
```
#groupby hotel
repeated_guests_df=df[df['is_repeated_guest']==1].groupby('hotel').size().
 ↪reset_index().rename(columns={0:'number_of_repated_guests'})
repeated_guests_df
```

[58]:
```
          hotel  number_of_repated_guests
0    City Hotel                      1657
1  Resort Hotel                      1707
```

[101]:
```
#plotting bar plot

plt.figure(figsize=(8,7))
sns.
 ↪barplot(x=repeated_guests_df['hotel'],y=repeated_guests_df['number_of_repated_guests'])

# set labels
plt.xlabel('Hotel')
plt.ylabel('Count of repeated visitors')
plt.title("Repeated visitors for each hotel",fontsize=20)
```

[101]:
```
Text(0.5, 1.0, 'Repeated visitors for each hotel')
```



3.5.1 Observation : Compared to City Hotels, Resort Hotel has a little bit more repeat visitors.

[ ]:

```

```

3.6 6. ADR compared between several months.

[60]:

```
# Group by month and hotel type

bookings_by_months_df=df.groupby(['arrival_date_month','hotel'])['adr'].mean().
↪reset_index()


#create month list

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

#order of the month list in the df along with values

bookings_by_months_df['arrival_date_month']=pd.
↪Categorical(bookings_by_months_df['arrival_date_month'],categories=months,ordered=True)

#sorting the value based on month
bookings_by_months_df=bookings_by_months_df.sort_values('arrival_date_month')
bookings_by_months_df
```

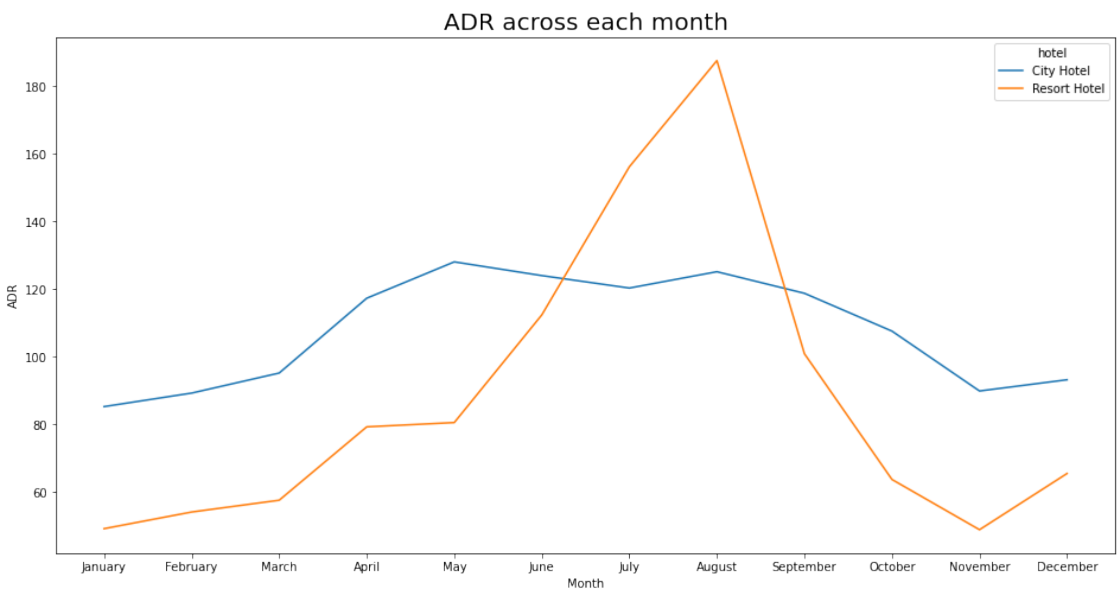
[60]:

	arrival_date_month	hotel	adr
8	January	City Hotel	85.269875
9	January	Resort Hotel	49.181693
6	February	City Hotel	89.266427
7	February	Resort Hotel	54.102809
15	March	Resort Hotel	57.590889
14	March	City Hotel	95.193911
0	April	City Hotel	117.314134
1	April	Resort Hotel	79.283805
17	May	Resort Hotel	80.551101
16	May	City Hotel	128.055724
13	June	Resort Hotel	112.380859
12	June	City Hotel	123.996416
11	July	Resort Hotel	156.166914
10	July	City Hotel	120.318314
3	August	Resort Hotel	187.566659
2	August	City Hotel	125.148662
22	September	City Hotel	118.764693
23	September	Resort Hotel	100.892331
20	October	City Hotel	107.585401

```
21            October  Resort Hotel   63.723065
18           November    City Hotel   89.882912
19           November  Resort Hotel   48.871043
5            December  Resort Hotel   65.488671
4            December    City Hotel   93.204767
```

[61]:
```
# plotting line graph

plt.figure(figsize=(16,8))
sns.
↪lineplot(x=bookings_by_months_df['arrival_date_month'],y=bookings_by_months_df['adr'],hue=b

# set lables
plt.title('ADR across each month',fontsize=20)
plt.xlabel('Month')
plt.ylabel('ADR')
```

[61]:
```
Text(0, 0.5, 'ADR')
```



3.6.1 Observation:

1. In comparison to City Hotels, the ADR for Resrot Hotel is higher in the months of July and August.Perhaps clients/people wish to vacation in resort hotels this summer. 2. January, February, March, April, October, November, and December are the ideal months for visitors to resort or city hotels because of the low average daily rate throughout these months.

[ ]:

3.7 7. ADR across Distribution Channel

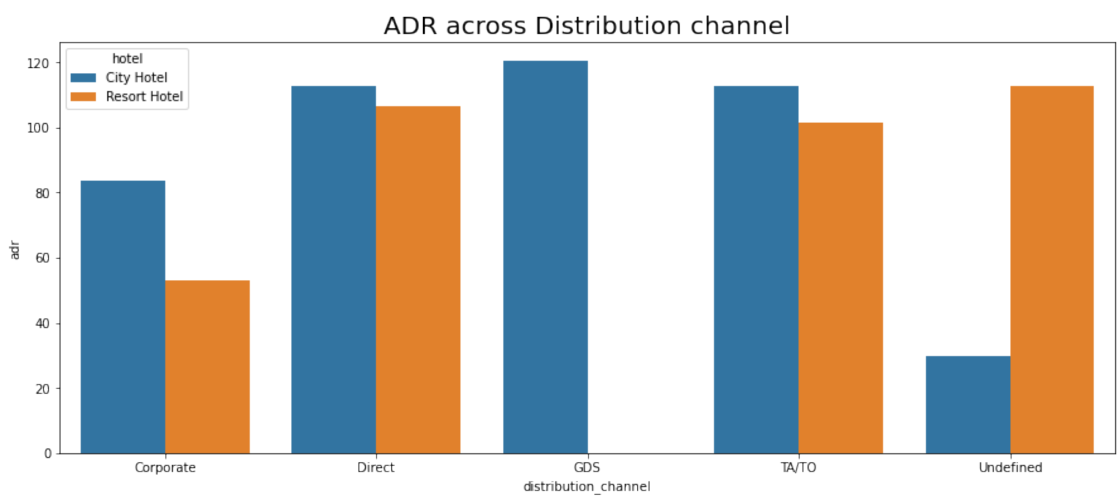
[62]:
```
# group by distribution channel and hotel

distribution_channel_adr=df.groupby(['distribution_channel','hotel'])['adr'].
↪mean().reset_index()

distribution_channel_adr
```

[62]:

	distribution_channel	hotel	adr
0	Corporate	City Hotel	83.777368
1	Corporate	Resort Hotel	53.036835
2	Direct	City Hotel	112.606688
3	Direct	Resort Hotel	106.566215
4	GDS	City Hotel	120.317845
5	TA/TO	City Hotel	112.663552
6	TA/TO	Resort Hotel	101.578317
7	Undefined	City Hotel	29.625000
8	Undefined	Resort Hotel	112.700000

[63]:
```
#plotting the graph

plt.figure(figsize=(15,6))
sns.barplot(x='distribution_channel', y='adr', data=distribution_channel_adr,␣
↪hue='hotel')
plt.title('ADR across Distribution channel',fontsize=20)
```

[63]: Text(0.5, 1.0, 'ADR across Distribution channel')



3.7.1 Observations:

1. "Direct" and "TA/TO" has almost equal ADR in both type of Hotel which is high among other channels 2. GDS scores highly in the "City Hotel" category. Bookings for Resort Hotels must rise at GDS. 3. This indicates that "Direct" and "TA/TO" are outperforming the other channels in terms of revenue generation.

[ ]:

```

```

3.8 8. ADR across Different Market Segment

[64]:

```
#Groupby market segment and hotel
market_seg_adr = df.groupby(['market_segment','hotel'])['adr'].mean().
 ↪reset_index()
market_seg_adr
```

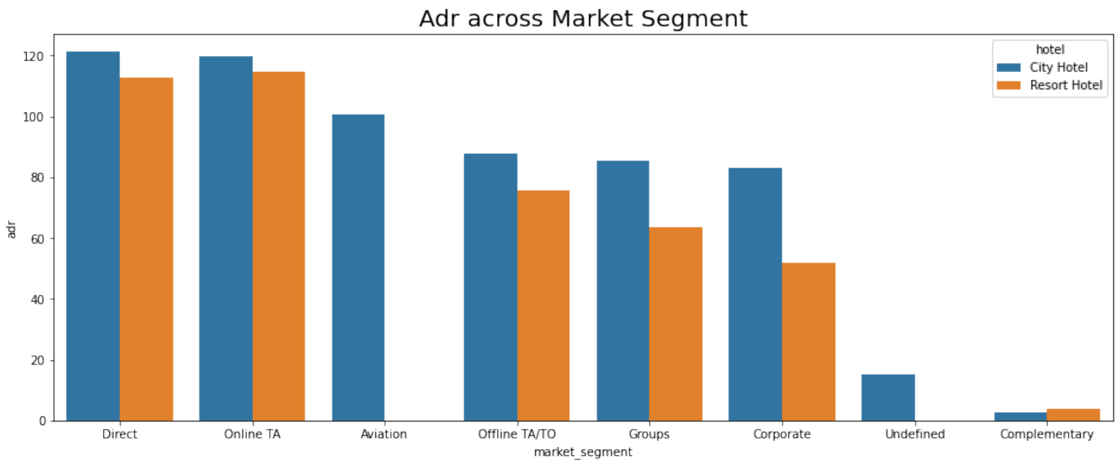
[64]:

	market_segment	hotel	adr
0	Aviation	City Hotel	100.613628
1	Complementary	City Hotel	2.802048
2	Complementary	Resort Hotel	3.868466
3	Corporate	City Hotel	83.020234
4	Corporate	Resort Hotel	51.920873
5	Direct	City Hotel	121.243682
6	Direct	Resort Hotel	112.827406
7	Groups	City Hotel	85.262047
8	Groups	Resort Hotel	63.688498
9	Offline TA/TO	City Hotel	87.632267
10	Offline TA/TO	Resort Hotel	75.730349
11	Online TA	City Hotel	119.971001
12	Online TA	Resort Hotel	114.776912
13	Undefined	City Hotel	15.000000

[65]:

```
# plotting barchart

plt.figure(figsize=(16,6))
sns.barplot(x='market_segment',y='adr',hue='hotel',data=market_seg_adr.
 ↪sort_values(by='adr',ascending=False))
plt.title('Adr across Market Segment',fontsize=20)
```

[65]: Text(0.5, 1.0, 'Adr across Market Segment')



3.8.1 Observation: In both types of hotels, "Direct" and "Online TA" are making the most contributions.

[ ]:

```

```

3.9 9. The relationship between ADR and Length of stay

[66]:

```
# Groupby adr,total,stay
adr_vs_stay = df.groupby(['total_stay'])["adr"].mean().reset_index()
adr_vs_stay.head()
```

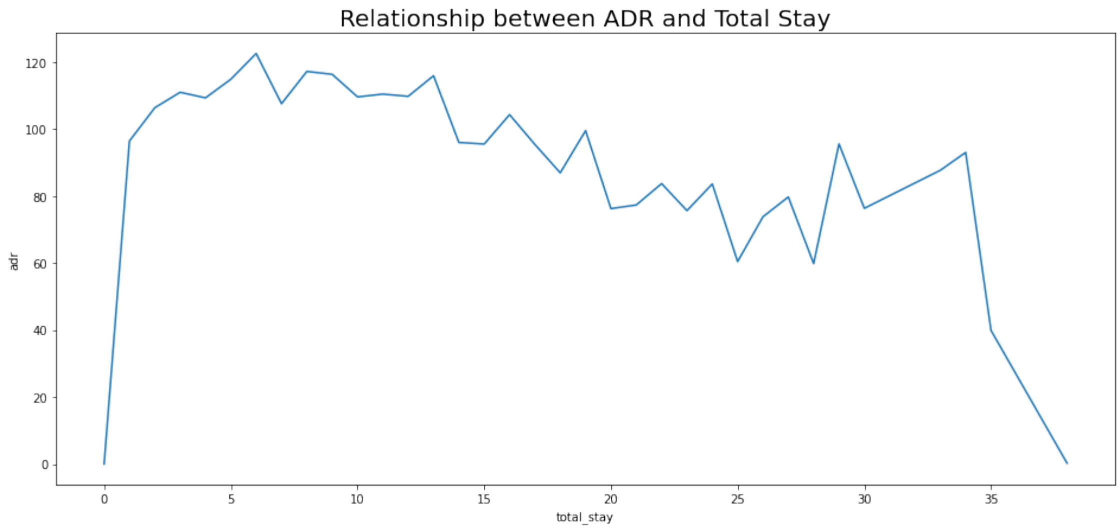
[66]:

	total_stay	adr
0	0	0.000000
1	1	96.405030
2	2	106.325130
3	3	110.941067
4	4	109.269494

[67]:

```
plt.figure(figsize=(16,7))
sns.lineplot(x='total_stay',y='adr',data=adr_vs_stay[0:35])
plt.title('Relationship between ADR and Total Stay',fontsize=20)
```

[67]: Text(0.5, 1.0, 'Relationship between ADR and Total Stay')



3.9.1 Observation : The average daily rate rises along with the length of stay. However, the ADR decreases if a guest stays for a longer time.

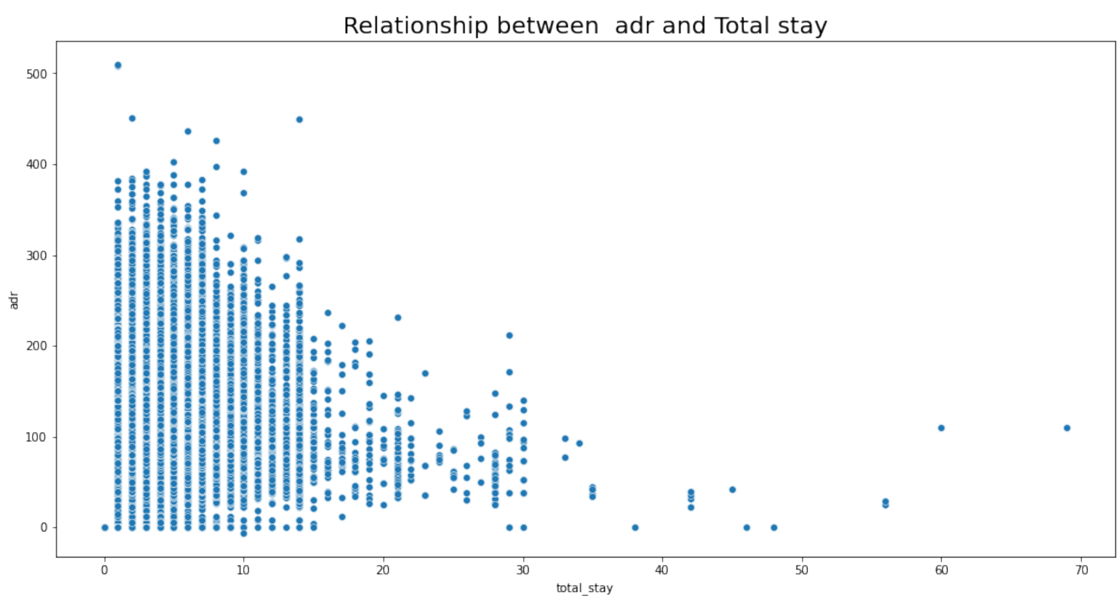
[ ]:

```
```

[68]:

```
df3=df.drop(df[df['adr'] > 1000].index)

plt.figure(figsize=(16,8))
sns.scatterplot(x='total_stay',y='adr',data=df3)
plt.title('Relationship between  adr and Total stay',fontsize=20)
```

[68]: Text(0.5, 1.0, 'Relationship between adr and Total stay')



3.9.2 Observation: We can infer from the aforementioned dispersion that adr is decreasing as stay rises. Therefore, customers can get good adr for longer stays.

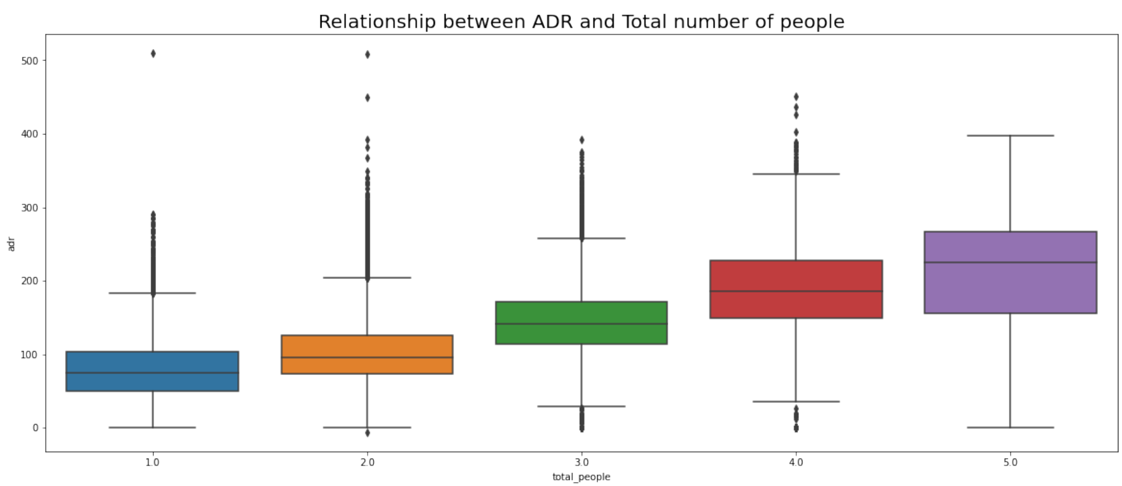
[ ]:

3.10 10. Relationship between ADR and Total number of people

[69]:
```
#creating new dataframe where "Total number of people is less than 6"

df4 = df3[df3['total_people'] < 6]
```

[70]:
```
#plotting the graph
plt.figure(figsize=(20,8))
sns.boxplot(x='total_people',y='adr',data=df4)

plt.title('Relationship between ADR and Total number of people',fontsize=20)
```

[70]:
```
Text(0.5, 1.0, 'Relationship between ADR and Total number of people')
```



3.10.1 Observation : ADR also rises in proportion to the Total number of people

[ ]:

3.11 11. Which type of Distribution Channel has the highest cancellation count?

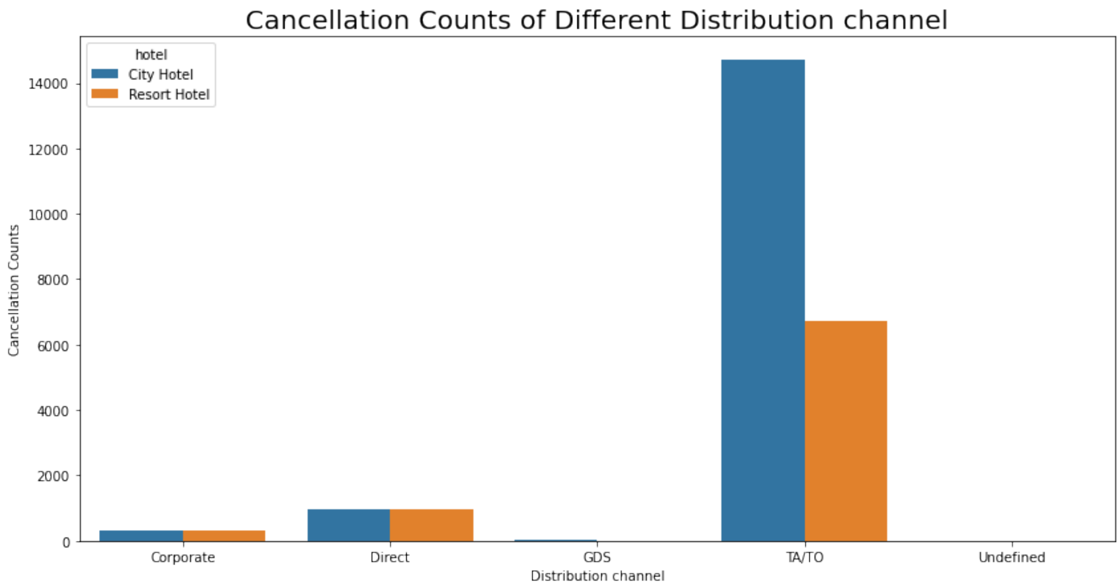
[71]:
```
# creating new DataFrame where bookings are cancelled.

canceled_df_DC = df[df['is_canceled']==1]

#group by distribution channel

canceled_df_DC=canceled_df_DC.groupby(['distribution_channel','hotel']).size().
    ↪reset_index().rename(columns={0:'Counts'})

canceled_df_DC
```

[71]:

	distribution_channel	hotel	Counts
0	Corporate	City Hotel	330
1	Corporate	Resort Hotel	316
2	Direct	City Hotel	971
3	Direct	Resort Hotel	952
4	GDS	City Hotel	36
5	TA/TO	City Hotel	14694
6	TA/TO	Resort Hotel	6706
7	Undefined	City Hotel	4

[72]:
```
# plotting the graph

plt.figure(figsize=(14,7))
sns.barplot(x='distribution_channel',y='Counts',hue="hotel",data =␣
    ↪canceled_df_DC)

plt.xlabel('Distribution channel')
plt.ylabel('Cancellation Counts')
plt.title('Cancellation Counts of Different Distribution channel',fontsize=20)
```

[72]:
```
Text(0.5, 1.0, 'Cancellation Counts of Different Distribution channel')
```



3.11.1 Observation:

In "TA/TO", City hotels has the high cancellation rate compared to resort hotels.

[ ]:

3.12 12. Which type of market segment's has the highest Cancellation Count ?

[73]:
```python
# creating new dataframe where bookings are canceled

market_segment_df=df[df['is_canceled']==1]

#group by market segment and hotel

market_segment_df=market_segment_df.groupby(['market_segment','hotel']).size().
    ↪reset_index().rename(columns={0:'counts'})

market_segment_df
```

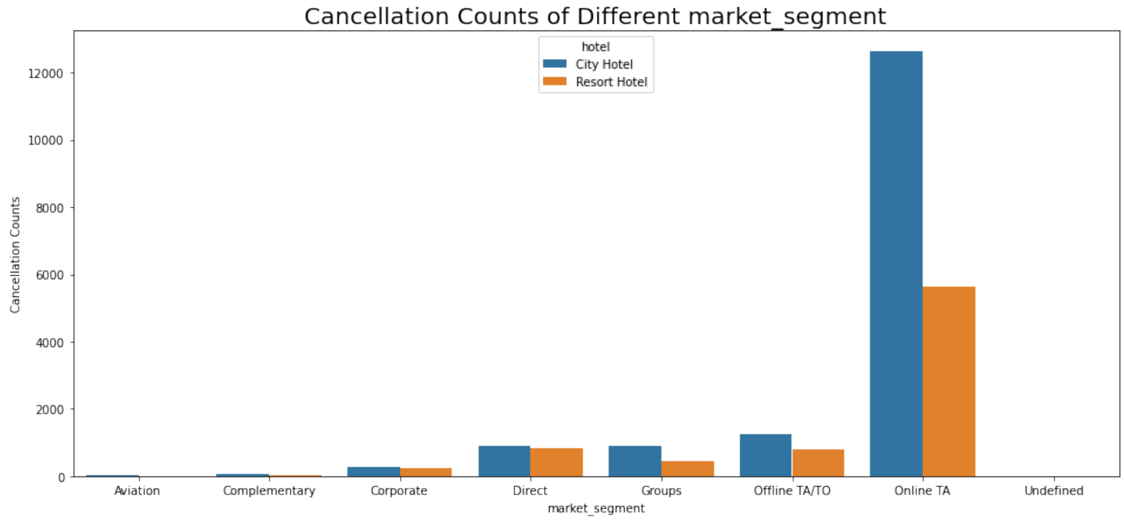
[73]:

	market_segment	hotel	counts
0	Aviation	City Hotel	45
1	Complementary	City Hotel	54
2	Complementary	Resort Hotel	31
3	Corporate	City Hotel	263
4	Corporate	Resort Hotel	246
5	Direct	City Hotel	912
6	Direct	Resort Hotel	825

```
7           Groups    City Hotel     887
8           Groups  Resort Hotel     445
9     Offline TA/TO    City Hotel    1257
10    Offline TA/TO  Resort Hotel     800
11        Online TA    City Hotel   12615
12        Online TA  Resort Hotel    5627
13        Undefined    City Hotel       2
```

[103]:
```
# plotting the graph

plt.figure(figsize=(16,7))
sns.barplot(x='market_segment',y='counts',hue="hotel",data= market_segment_df)

plt.xlabel('market_segment')
plt.ylabel('Cancellation Counts')
plt.title('Cancellation Counts of Different market_segment',fontsize=20)
```

[103]:
```
Text(0.5, 1.0, 'Cancellation Counts of Different market_segment')
```



3.12.1 Observations : Online T/A has the highest cancellation in both type of Hotels

[ ]:

3.13 13. What is the Total stay length in each Hotel type ?

[75]:
```
#Group by Total stay and hotel type

day_count = df.groupby(['total_stay', 'hotel']).agg('count').reset_index().
↪iloc[:, :3].rename(columns={'is_canceled':'Number of stays'})
```

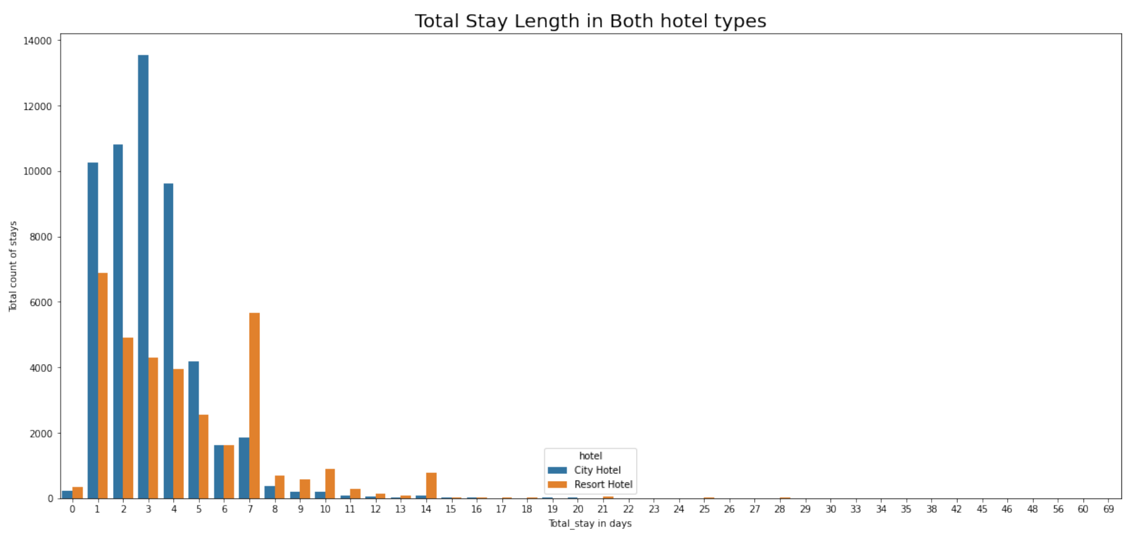
```python
day_count.head(10)
```

[75]:

	total_stay	hotel	Number of stays
0	0	City Hotel	231
1	0	Resort Hotel	360
2	1	City Hotel	10270
3	1	Resort Hotel	6899
4	2	City Hotel	10813
5	2	Resort Hotel	4921
6	3	City Hotel	13542
7	3	Resort Hotel	4285
8	4	City Hotel	9610
9	4	Resort Hotel	3955

[104]:
```python
# plotting the graph

plt.figure(figsize=(20,9))
sns.barplot(x='total_stay',y='Number of stays',hue='hotel',data=day_count)

plt.xlabel('Total_stay in days')
plt.ylabel('Total count of stays')
plt.title('Total Stay Length in Both hotel types',fontsize=20)
```

[104]: Text(0.5, 1.0, 'Total Stay Length in Both hotel types')



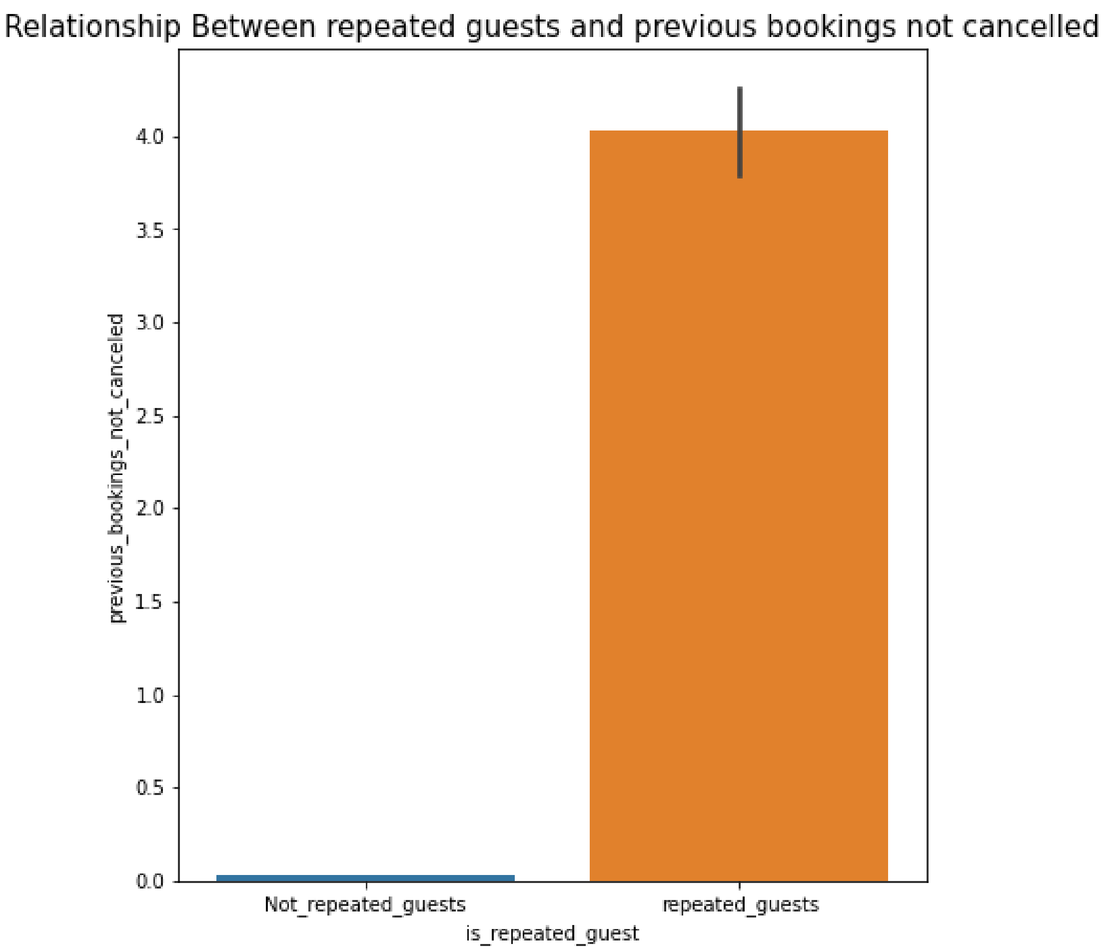
3.13.1 Observation: The ideal stay in either sort of hotel is under seven days.

[ ]:

3.14 14. Relationship between returning visitors and cancelled reservations.

[77]:
```
plt.figure(figsize=(7,8))
sns.barplot(x='is_repeated_guest',y= 'previous_bookings_not_canceled',data=df)
plt.xticks([0,1],['Not_repeated_guests','repeated_guests'],fontsize=10)
plt.title('Relationship Between repeated guests and previous bookings not␣
↪cancelled',fontsize=15)
```

[77]:
```
Text(0.5, 1.0, 'Relationship Between repeated guests and previous bookings not
cancelled')
```



** previous_bookings_not_canceled : Number of previous bookings not cancelled by the customer prior to the current booking

3.14.1 Observation : Booking cancellations are more common among non-repeat visitors.

[ ]:

3.15 15. The correlation between booking cancellation and the percentage of guests who are not assigned to the same room that they reserved.

[78]:
```
# Group by is_canceled

grp_by_canceled=df.groupby('is_canceled')
```

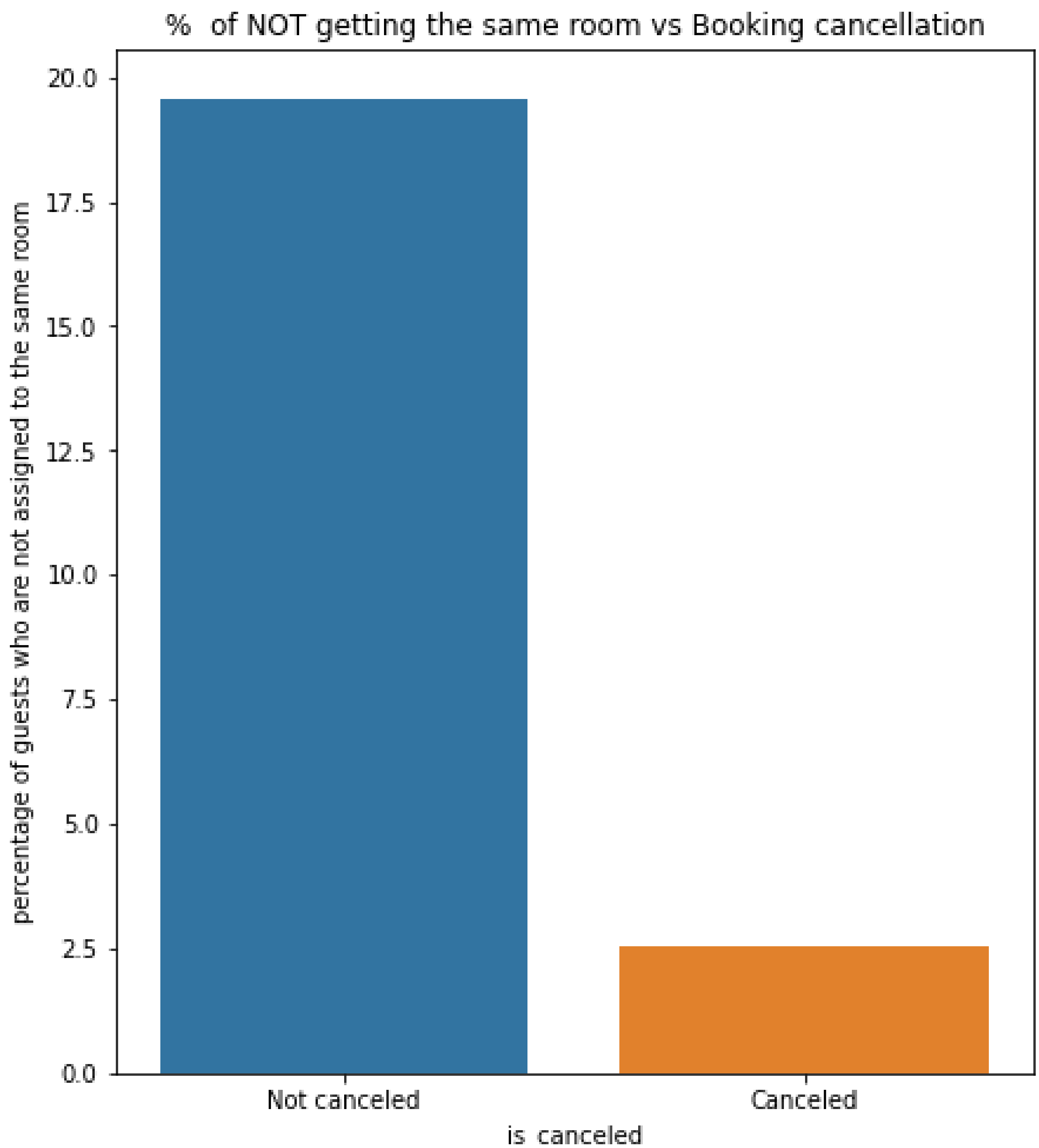
[79]:
```
# create DF and calculate  percentage of guests who are not assigned to the␣
↪same room that they reserved.

DF=pd.DataFrame(grp_by_canceled['Same_room_alloted_or_not'].sum()*100/
↪grp_by_canceled.size()).rename(columns={0:"percentage"})
DF
```

[79]:

is_canceled	percentage
0	19.572610
1	2.565705

[105]:
```
plt.figure(figsize=(7,8))
sns.barplot(x=DF.index,y=DF['percentage'])


plt.title('%  of NOT getting the same room vs Booking cancellation')
plt.xlabel('is_canceled')
plt.ylabel('percentage of guests who are not assigned to the same room')
plt.xticks([0,1],['Not canceled','Canceled'])
```

[105]:
```
([<matplotlib.axis.XTick at 0x1b43f8f5760>,
  <matplotlib.axis.XTick at 0x1b43f8f5940>],
 [Text(0, 0, 'Not canceled'), Text(1, 0, 'Canceled')])
```



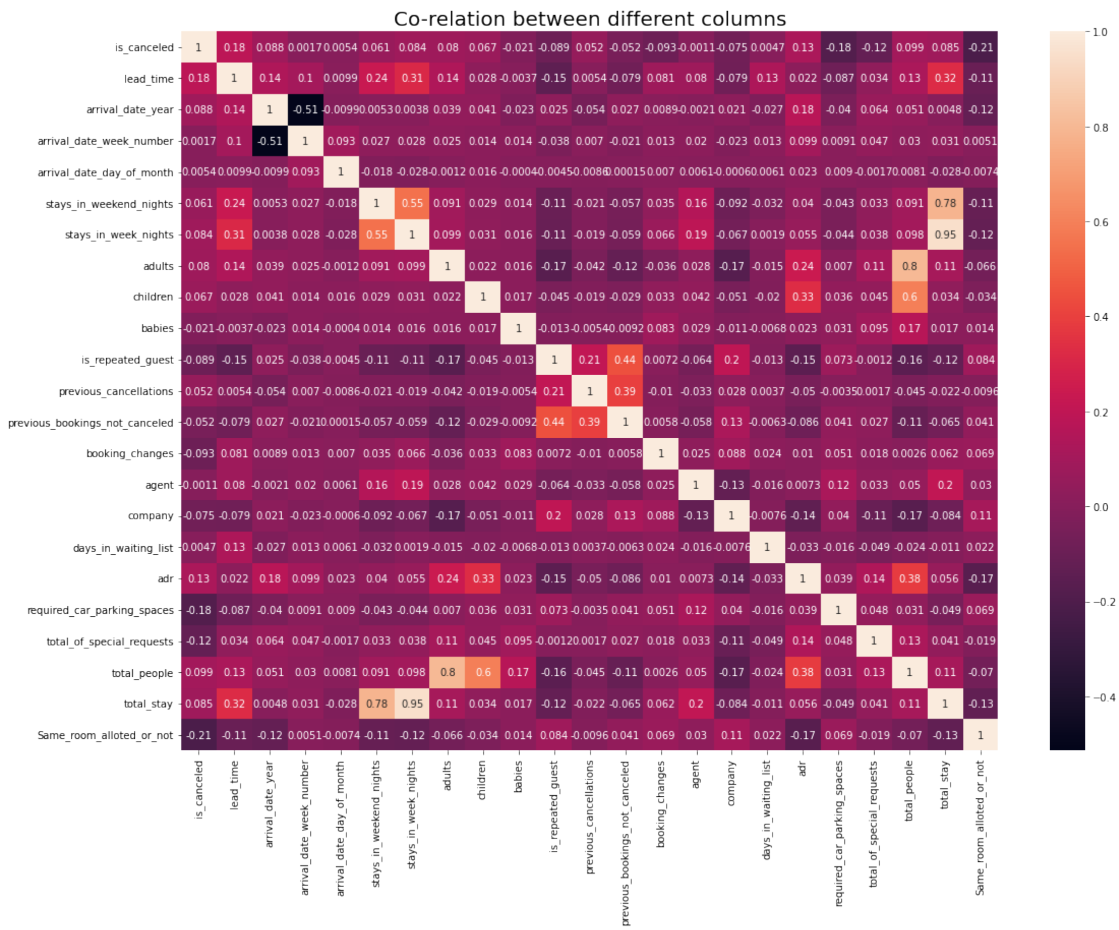
3.15.1 Observations : It is evident that even if customers are not given the rooms they requested throughout the booking process, there is no appreciable impact on cancellations of reservations.

```
[ ]:
```

3.16 16. Correlation between different columns

[106]:
```
plt.figure(figsize=(18,13))

sns.heatmap(df.corr(),annot=True)
plt.title('Co-relation between different columns',fontsize=20)
```

[106]: `Text(0.5, 1.0, 'Co-relation between different columns')`



3.16.1 Observations :

1. 'is_canceled' and 'same_room_alloted_or_not' are negatively corelated. So, even if he doesn't get the exact accommodation he reserved, the customer is unlikely to change his reservations. It is already proven, to the last analysis.
2. Total people and adr are positively Correlated that means more number of people increases the adr.
3. "is_repeted guest" and "previous_bookings_not_canceled" have a stong positive correlation.
4. Lead time and total stay have a positive relationship. This implies that the lead time will increase as the customer stays longer.

```
[ ]:
```

4 Findings :

1. Here we can clearly see that City Hotel is most preferred hotel by the visitors.
2. 'Agent 9' who has Most Number of Bookings
3. 27.5 % of the bookings were cancelled.
4. 3.9% of visitors went back to the hotels. 96.1 percent of the visitors were first-timers. The retention rate is consequently low.
5. The percentage of transient customers is higher at 82.4%. A very small fraction of Bookings is connected to the Group.
6. The parking space was not needed by 91.6% of the visitors. 8.3% of visitors only needed one parking space.
7. Above 80% of the bookings were not changed by guests.
8. Preferred Meal Type
   - Consequently, bed and breakfast (BB) is the most popular sort of meal among the visitors.
   - Almost Equally desirable are HB- (Half Board) and SC- (Self Catering).
9. The majority of visitors almost 98.7% prefer "No deposit" types of deposits which means the customer made no deposit to guarantee the booking.
10. Top ten countries from which the most visitors arrive
    - PRT- Portugal
    - GBR- United Kingdom
    - FRA- France
    - ESP- Spain
    - DEU - Germany
    - ITA - Itlay
    - IRL - Ireland
    - BEL -Belgium
    - BRA -Brazil
    - NLD-Netherlands
11. The most preferred Room type is "A".
12. The months with the most bookings were July and August. Bookings may have been made in anticipation of summer vacations.
13. 'TA/TO' distribution method is most popular for hotel reservations
14. In 2016, the majority of reservations for resort hotels and city hotels were made.
15. The majority of the time, guests receive the exact accommodation they have reserved.

16. The highest ADR is at the City Hotel. This indicates that city hotels make more money than resort hotels.
17. City hotel experiences the highest rate of cancellations.
18. Resorts Hotels have a slightly longer average lead time. Customers must make very early travel plans as a result.
19. There is a lengthier wait time at city hotels than at resort hotels. As a result, we can conclude that city hotels are significantly busier than resort hotels.
20. Compared to City Hotels, Resort Hotel has a little bit more repeat visitor.
21. ADR compared between several months.
    - In comparison to City Hotels, the ADR for Resort Hotel is higher in the months of July and August. Perhaps clients/people wish to vacation in resort hotels this summer.
    - January, February, March, April, October, November, and December are the ideal months for visitors to resort or city hotels because of the low average daily rate throughout these months.
22. ADR across Distribution Channel
    - "Direct" and "TA/TO" has almost equal ADR in both type of Hotel which is high among other channels.
    - GDS scores highly in the "City Hotel" category. Bookings for Resort Hotels must rise at GDS.
    - This indicates that "Direct" and "TA/TO" are outperforming the other channels in terms of revenue generation.
23. If we compare ADR across Different Market Segment we can see that In both types of hotels, "Direct" and "Online TA" are making the most contributions.
24. The ADR rises along with the length of stay. However, the ADR decreases if a guest stays for a longer time.
25. ADR also rises in proportion to the Total number of people.
26. In "TA/TO" type of Distribution Channel has the highest cancellation count.
27. If we compare the Cancellation Count among different market segment then we can find that Online T/A has the highest cancellation in both type of Hotels.
28. The ideal stay in either sort of hotel is under seven days.
29. Booking cancellations are more common among non-repeat visitors.
30. Even if customers are not given the rooms they requested throughout the booking process, there is no appreciable impact on cancellations of reservations.

[ ]:

```
```

[ ]:

```
```