➤ SITUATION : City Hotel and Resort Hotel have experienced a significant increase in booking cancellations, which has adversely affected their financial performance and operational efficiency. This report delves into the underlying reasons for these high cancellation rates and provides recommendations to mitigate their impact on revenue generation.

➤ TASK : The primary objective of this study is to identify the factors contributing to high cancellation rates, leading to reduced revenues and suboptimal hotel room utilization. By addressing these issues, both hotels aim to enhance their efficiency in revenue generation.

➤ ACTION : To ensure the collection of valid data, an exhaustive research methodology was implemented, encompassing the retrieval of booking records, meticulous examination of customer feedback, observation of prevailing market trends, and comprehensive analysis of industry best practices. Subsequently, sophisticated analytical tools were employed to discern and establish patterns and correlations in cancellation behavior, providing valuable insights for the study.

➤ RESULT : By proactively addressing the issue of high cancellation rates through the implementation of the recommended measures, both City Hotel and Resort Hotel significantly enhanced their revenue generation capabilities. By closely monitoring the impact of these measures, the hotels successfully can achieve a targeted reduction in cancellation rates, measured as a percentage decrease in cancellations over a specified period.

1 - Importing Libraries

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

The 2nd line imports the Matplotlib library's pyplot module, allowing to create and customize plots.

Seaborn is a Python data visualization library that is built on top of Matplotlib. It provides a high-level interface for creating beautiful and informative statistical graphics. Seaborn is particularly useful for visualizing complex datasets and drawing attractive statistical plots with minimal code.

warnings allows you to handle warnings in Python, and "warnings.filterwarnings('ignore')" to ignore any warnings that may be raised during the execution of the code.

2 - Loading the Dataset

In [5]:
```
df = pd.read_csv("E:/STUDY/Data Analyst/Projects/Ayushi/Hotel Data Analysis -- Pyth
```

3 - Exploratory Data Analysis & Data Cleaning

In [6]:
```
df.head(10)
```

Out[6]:

	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	Resort Hotel	0	14	2015	July	
6	Resort Hotel	0	0	2015	July	
7	Resort Hotel	0	9	2015	July	
8	Resort Hotel	1	85	2015	July	
9	Resort Hotel	1	75	2015	July	

10 rows × 32 columns

In [7]:
```
df.tail(10)
```

Out[7]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_we
119380	City Hotel	0	44	2017	August	
119381	City Hotel	0	188	2017	August	
119382	City Hotel	0	135	2017	August	
119383	City Hotel	0	164	2017	August	
119384	City Hotel	0	21	2017	August	
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	

10 rows × 32 columns

In [8]:
```
df.shape
```

Out[8]:
```
(119390, 32)
```

In [9]:
```
df.columns                    # need to focus on the relevant columns only
```

Out[9]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

In [10]:
```
df.info()          # checking the data_types and null values
```

```
<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
```

In [11]:
```
df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'], forma

# choosing the correct format for the dates
```

In [12]:
```
df.info()               # checking the changed format
```

```
<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  datetime64[ns]
dtypes: datetime64[ns](1), float64(4), int64(16), object(11)
memory usage: 29.1+ MB
```

In [13]:
```
df.describe(include = 'object')       # to get the total no. of object values in ev
```

Out[13]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channe
count	119390	119390	119390	118902	119390	119390
unique	2	12	5	177	8	5
top	City Hotel	August	BB	PRT	Online TA	TA/TO
freq	79330	13877	92310	48590	56477	97870

The above code is used to generate a summary of the DataFrame 'df' that includes descriptive statistics for columns containing object (string) data types.

In [14]:
```
for col in df.describe(include = 'object').columns:
    print(col)
    print(df[col].unique())
    print('-'*70)
```

```
hotel
['Resort Hotel' 'City Hotel']
----------------------------------------------------------------------
arrival_date_month
['July' 'August' 'September' 'October' 'November' 'December' 'January'
 'February' 'March' 'April' 'May' 'June']
----------------------------------------------------------------------
meal
['BB' 'FB' 'HB' 'SC' 'Undefined']
----------------------------------------------------------------------
country
['PRT' 'GBR' 'USA' 'ESP' 'IRL' 'FRA' nan 'ROU' 'NOR' 'OMN' 'ARG' 'POL'
 'DEU' 'BEL' 'CHE' 'CN' 'GRC' 'ITA' 'NLD' 'DNK' 'RUS' 'SWE' 'AUS' 'EST'
 'CZE' 'BRA' 'FIN' 'MOZ' 'BWA' 'LUX' 'SVN' 'ALB' 'IND' 'CHN' 'MEX' 'MAR'
 'UKR' 'SMR' 'LVA' 'PRI' 'SRB' 'CHL' 'AUT' 'BLR' 'LTU' 'TUR' 'ZAF' 'AGO'
 'ISR' 'CYM' 'ZMB' 'CPV' 'ZWE' 'DZA' 'KOR' 'CRI' 'HUN' 'ARE' 'TUN' 'JAM'
 'HRV' 'HKG' 'IRN' 'GEO' 'AND' 'GIB' 'URY' 'JEY' 'CAF' 'CYP' 'COL' 'GGY'
 'KWT' 'NGA' 'MDV' 'VEN' 'SVK' 'FJI' 'KAZ' 'PAK' 'IDN' 'LBN' 'PHL' 'SEN'
 'SYC' 'AZE' 'BHR' 'NZL' 'THA' 'DOM' 'MKD' 'MYS' 'ARM' 'JPN' 'LKA' 'CUB'
 'CMR' 'BIH' 'MUS' 'COM' 'SUR' 'UGA' 'BGR' 'CIV' 'JOR' 'SYR' 'SGP' 'BDI'
 'SAU' 'VNM' 'PLW' 'QAT' 'EGY' 'PER' 'MLT' 'MWI' 'ECU' 'MDG' 'ISL' 'UZB'
 'NPL' 'BHS' 'MAC' 'TGO' 'TWN' 'DJI' 'STP' 'KNA' 'ETH' 'IRQ' 'HND' 'RWA'
 'KHM' 'MCO' 'BGD' 'IMN' 'TJK' 'NIC' 'BEN' 'VGB' 'TZA' 'GAB' 'GHA' 'TMP'
 'GLP' 'KEN' 'LIE' 'GNB' 'MNE' 'UMI' 'MYT' 'FRO' 'MMR' 'PAN' 'BFA' 'LBY'
 'MLI' 'NAM' 'BOL' 'PRY' 'BRB' 'ABW' 'AIA' 'SLV' 'DMA' 'PYF' 'GUY' 'LCA'
 'ATA' 'GTM' 'ASM' 'MRT' 'NCL' 'KIR' 'SDN' 'ATF' 'SLE' 'LAO']
----------------------------------------------------------------------
market_segment
['Direct' 'Corporate' 'Online TA' 'Offline TA/TO' 'Complementary' 'Groups'
 'Undefined' 'Aviation']
----------------------------------------------------------------------
distribution_channel
['Direct' 'Corporate' 'TA/TO' 'Undefined' 'GDS']
----------------------------------------------------------------------
reserved_room_type
['C' 'A' 'D' 'E' 'G' 'F' 'H' 'L' 'P' 'B']
----------------------------------------------------------------------
assigned_room_type
['C' 'A' 'D' 'E' 'G' 'F' 'I' 'B' 'H' 'P' 'L' 'K']
----------------------------------------------------------------------
deposit_type
['No Deposit' 'Refundable' 'Non Refund']
----------------------------------------------------------------------
customer_type
['Transient' 'Contract' 'Transient-Party' 'Group']
----------------------------------------------------------------------
reservation_status
['Check-Out' 'Canceled' 'No-Show']
----------------------------------------------------------------------
```

The above code is a loop that iterates through each column in the summary statistics generated by df.describe(include='object').

It then prints the column name, the unique values present in that column, and a line of 70 dashes for separation.

In [15]:
```
df.isnull().sum()                    # to calculate the number of missing (null) valu
```

Out[15]:
```
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
```

In [16]:
```
df.drop(['company', 'agent'], axis = 1, inplace = True)
df.dropna(inplace = True)
```

Removing 'agent' column bcoz it has no relevance in calculating the revenue of the hotel.

Removing 'company' column bcoz we can't handle so much of company data

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
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
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 [18]:
```
df.describe()
```

Out[18]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_d
count	118898.000000	118898.000000	118898.000000	118898.000000	
mean	0.371352	104.311435	2016.157656	27.166555	
min	0.000000	0.000000	2015.000000	1.000000	
25%	0.000000	18.000000	2016.000000	16.000000	
50%	0.000000	69.000000	2016.000000	28.000000	
75%	1.000000	161.000000	2017.000000	38.000000	
max	1.000000	737.000000	2017.000000	53.000000	
std	0.483168	106.903309	0.707459	13.589971	

In the above o/p, "adr" (Avg Daily Rate) is -6 and 5400 which is inappropriate for a daily average record, so it should be removed.

In [19]:
```
df = df[(df['adr'] < 5000) & (df['adr'] > 0)]          # to keep only the rows where
```

In [20]:
```
df.describe()
```

Out[20]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_d
count	116958.000000	116958.000000	116958.000000	116958.000000	
mean	0.375767	105.376879	2016.161443	27.138109	
min	0.000000	0.000000	2015.000000	1.000000	
25%	0.000000	19.000000	2016.000000	16.000000	
50%	0.000000	71.000000	2016.000000	27.000000	
75%	1.000000	162.000000	2017.000000	38.000000	
max	1.000000	709.000000	2017.000000	53.000000	
std	0.484322	106.944356	0.706509	13.561162	

4 - Data Analysis and Visualizations

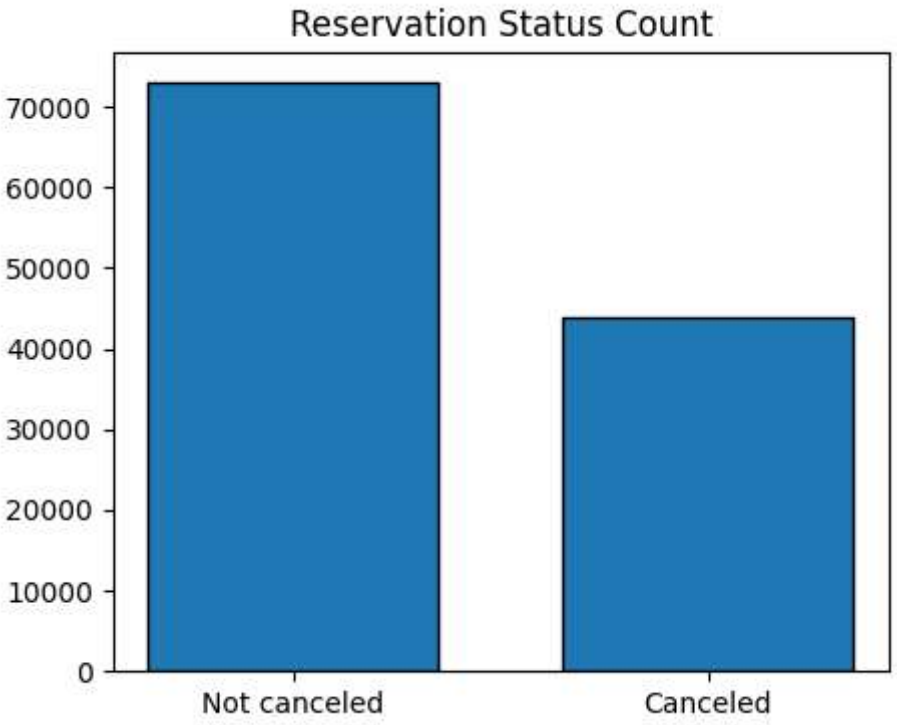
In [21]:
```
cancelled_perc = df['is_canceled'].value_counts(normalize = True)
print(cancelled_perc)
```

```
is_canceled
0    0.624233
1    0.375767
Name: proportion, dtype: float64
```

We can see that 62% of reservations are not cancelled and around 38% are cancelled.

Although 38% is less in comparison to 62%, but it can also be oberved that the cancellation is more than half of successful bookings

In [22]:
```
plt.figure(figsize = (5,4))                # Creating a figure-space with a specifi

plt.title('Reservation Status Count')       #  Title of the plot

plt.bar(['Not canceled','Canceled'],df['is_canceled'].value_counts(), edgecolor = '
            # Creating a bar plot with the count of 'is_cancelled' values

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



The first argument is a list of the x-axis labels ('Not canceled' and 'Canceled'), the second argument is the count of 'is_canceled' values obtained from df['is_canceled'].value_counts().

And the edgecolor and width parameters are used for customizing the appearance of the bars.

In [23]:
```
plt.figure(figsize = (8,4))

ax1= sns.countplot(x = 'hotel', hue = 'is_canceled', data = df, palette = 'Blues')
            # Creating a count plot using Seaborn's countplot() function

legend_labels = ax1. get_legend_handles_labels()
            # Getting the legend handles and labels from the countplot

ax1.legend(bbox_to_anchor=(1,1))       # Placing the legend outside the plot at the
                                       # The bbox_to_anchor parameter specifies the

plt.title('Reservation status in different hotels', size = 20)

plt.xlabel('Hotel')                          # Setting the x-axis label

plt.ylabel('Total Reservations')         # Setting the y-axis label

plt.legend(['not canceled', 'canceled'])     # Setting the legend labels manually (

plt.show()
```

The 'x' parameter specifies the column 'hotel' to be plotted on the x-axis, the 'hue' parameter groups the bars based on the 'is_canceled' column, and the 'data' parameter specifies the DataFrame to use for plotting. The palette='Blues' sets the color palette for the plot.

In [24]:
```
resort_hotel = df[df['hotel'] == 'Resort Hotel']
resort_hotel['is_canceled'].value_counts(normalize = True)
```

Out[24]:
```
is_canceled
0    0.717039
1    0.282961
Name: proportion, dtype: float64
```

We can see that 72% of the bookings are successful while 28% are being cancelled.

The above code will generate a new Dataframe 'resort_hotel' which will contain only the rows from the original DataFrame df where the 'hotel' column has the value 'Resort Hotel'.

Here, 'resort_hotel' is a new DataFrame, and any modifications or operations we perform on it will not affect the original DataFrame 'df'.

In [25]:
```
city_hotel = df[df['hotel'] == 'City Hotel']
city_hotel['is_canceled'].value_counts(normalize = True)
```

Out[25]:
```
is_canceled
0    0.578059
1    0.421941
Name: proportion, dtype: float64
```

We can see that 58% of the bookings are successful while 42% are being cancelled , which is quite high

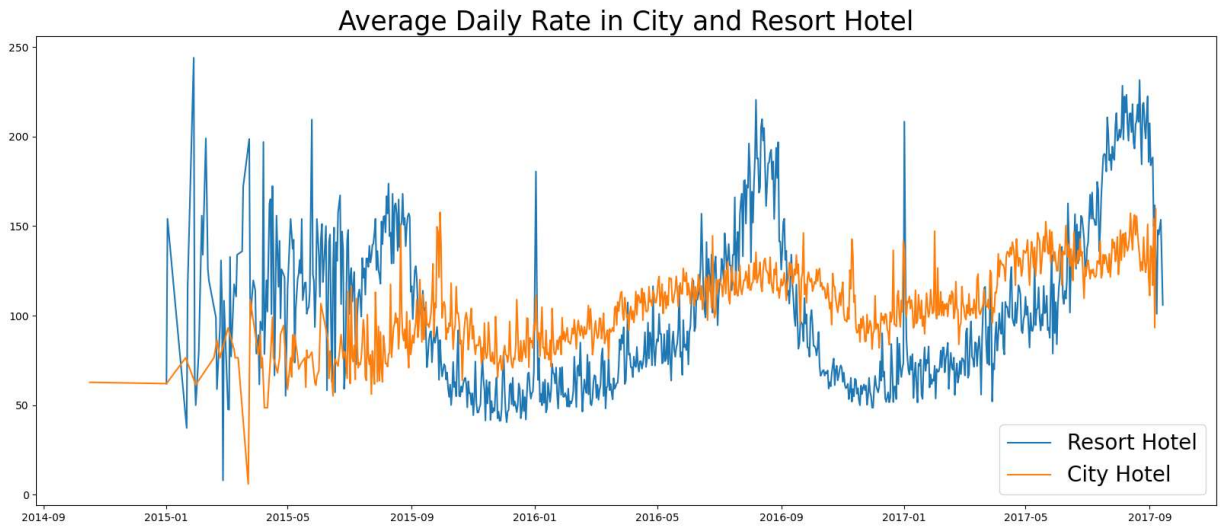
In [26]:
```
resort_hotel = resort_hotel.groupby('reservation_status_date')[['adr']].mean()
city_hotel = city_hotel.groupby('reservation_status_date')[['adr']].mean()
```

In the above code, we are using the groupby() function in combination with the mean() function to calculate the average daily rate (ADR) for the 'Resort Hotel' and 'City Hotel' separately.

We are grouping the data by the 'reservation_status_date' column and then calculating the mean ADR for each date.

The result will be DataFrames with two columns: 'reservation_status_date' and 'adr', where 'adr' represents the average daily rate for each date. The index of the DataFrames will be the unique dates from the 'reservation_status_date' column.

In [27]:
```
plt.figure(figsize = (20,8))

plt.title('Average Daily Rate in City and Resort Hotel', fontsize = 25)

plt.plot(resort_hotel.index, resort_hotel['adr'], label = 'Resort Hotel')     # Plo
plt.plot(city_hotel.index, city_hotel['adr'], label = 'City Hotel')

plt.legend(fontsize = 20)

plt.show()
```

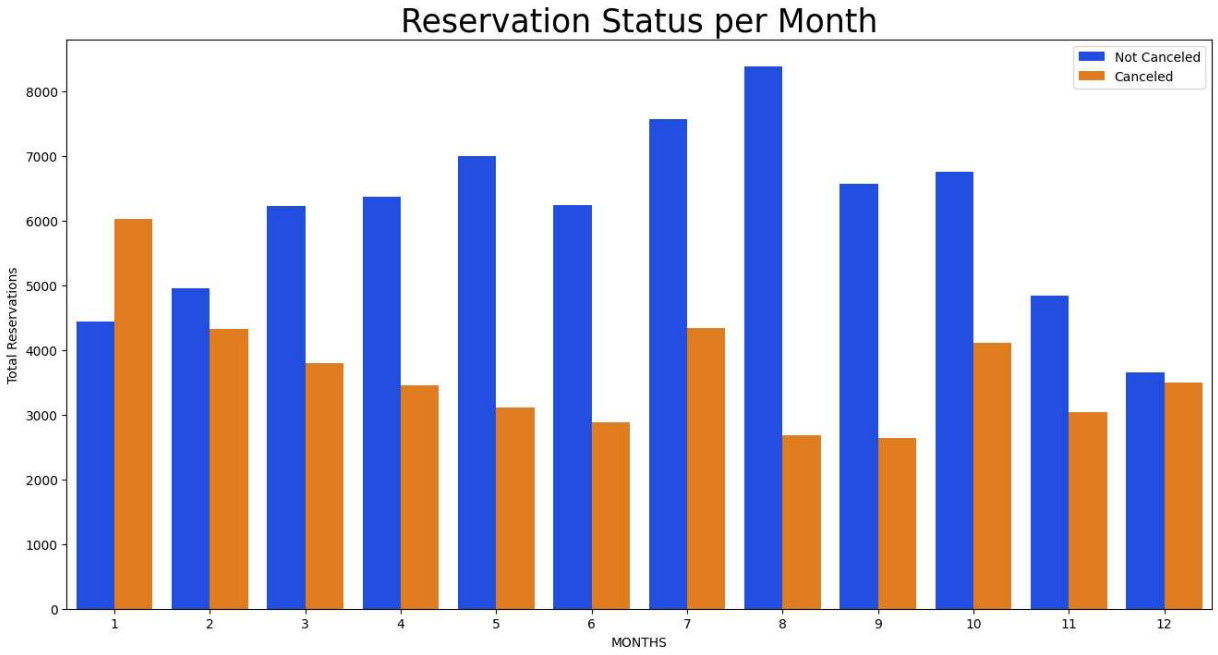


The spikes in the graph can be considered as 'weekends' bookings.

It can be observed that the booking price for 'City Hotel' is less as compared to that of 'Resort Hotel' for some days.

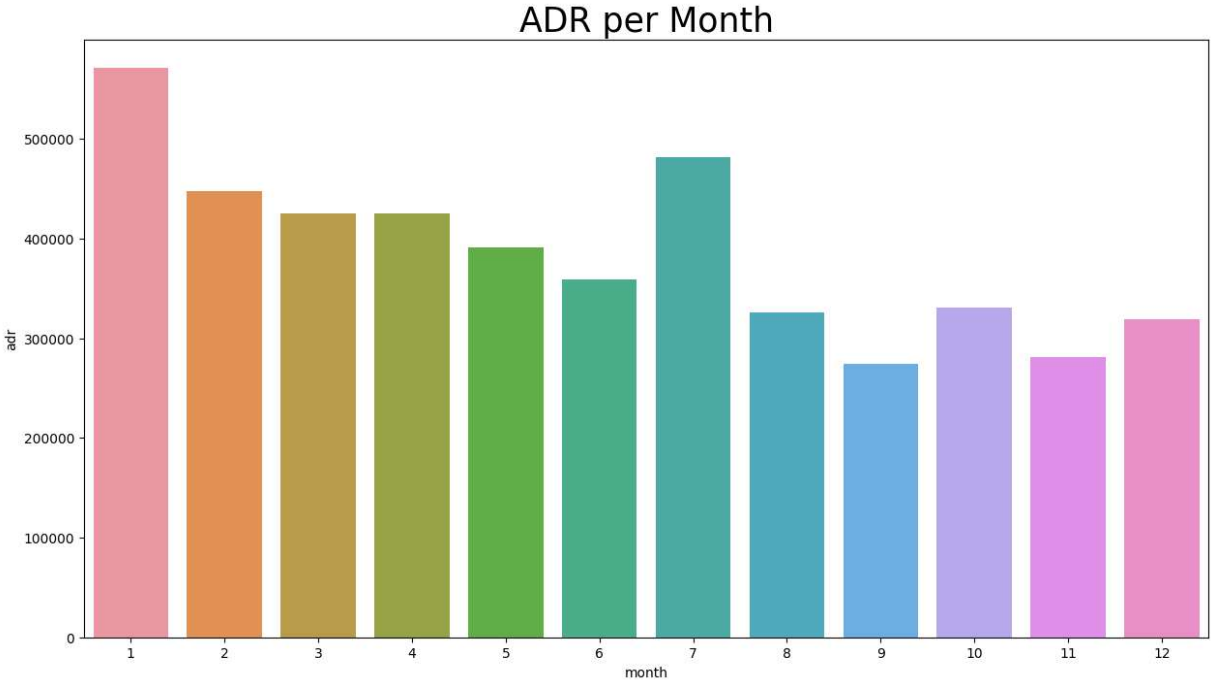
In [28]:
```
df['month'] = df['reservation_status_date'].dt.month          # all the months from

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

ax1 = sns.countplot(x = 'month', hue = 'is_canceled', data = df, palette = 'bright'
```

```
legend_labels,_ = ax1. get_legend_handles_labels()     # Getting the legend handles

ax1.legend(bbox_to_anchor=(1,1))

plt.title('Reservation Status per Month', size = 25)

plt.xlabel('MONTHS')                          # Setting the name for x and y axis
plt.ylabel('Total Reservations')

plt.legend(['Not Canceled', 'Canceled'])   # Setting the name for the legend

plt.show()
```



The resulting plot is showing the bars representing the number of reservations that were not canceled and reservations that were canceled for each month.

In [29]:
```
plt.figure(figsize = (15,8))

plt.title('ADR per Month', fontsize = 25)

cancelled_adr_sum = df[df['is_canceled'] == 1].groupby('month')[['adr']].sum().rese

    # Filter the DataFrame to include only canceled reservations (is_canceled == 1)
    # Group by 'month' and calculate the sum of ADR for each month
    # Reset the index to make 'month' a regular column

sns.barplot(x='month', y='adr', data=cancelled_adr_sum)

plt.show()
```



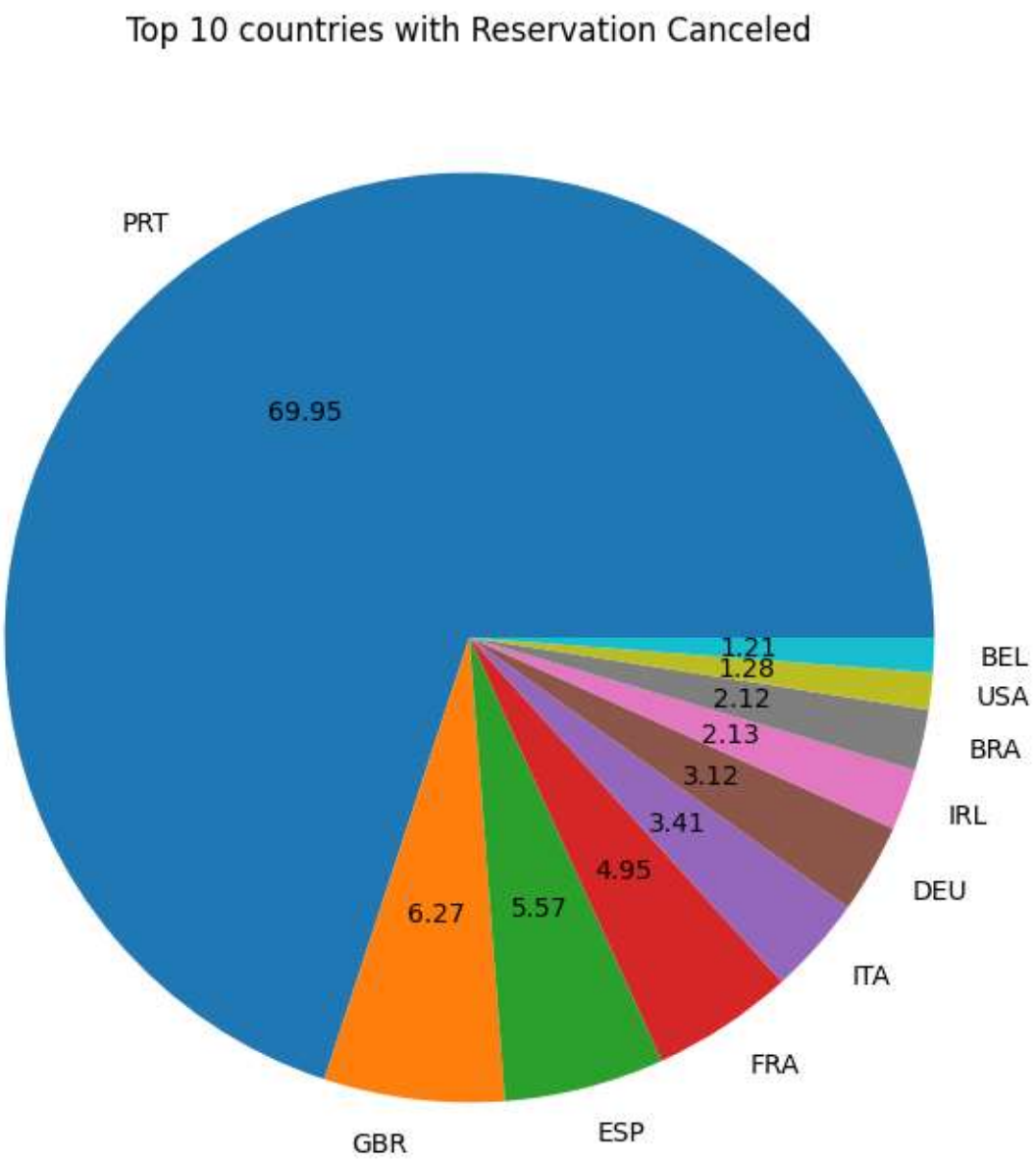
The resulting bar plot is showing the sum of the average daily rate (ADR) for canceled reservations for each month.

Each bar represents the ADR sum for a specific month, and the x-axis labels indicate the corresponding months.

The height of each bar represents the ADR sum value for that month.

This shows that higher bookings will invite higher cancellations as well.

In [30]:
```
cancelled_data = df[df['is_canceled'] == 1]                # Filtering the DataFram

top_10_country = cancelled_data['country'].value_counts()[:10]  # Getting the top 1

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

plt.title('Top 10 countries with Reservation Canceled')

plt.pie(top_10_country, autopct = '%.2f', labels = top_10_country.index)

plt.show()
```



The above pie-chart shows that "Portugal" is the country with highest cancellations. This can be improved by giving seasonal discounts, good advertising, etc.

In [31]:
```
df['market_segment'].value_counts()
```

Out[31]:
```
market_segment
Online TA        56036
Offline TA/TO    23829
Groups           19554
Direct           12210
Corporate         5035
Aviation           231
Complementary       63
Name: count, dtype: int64
```

Online TravelAgents brings the highest bookings.

In [32]:
```
df['market_segment'].value_counts(normalize = True)
```

Out[32]:
```
market_segment
Online TA        0.479112
Offline TA/TO    0.203740
Groups           0.167188
Direct           0.104396
Corporate        0.043050
Aviation         0.001975
Complementary    0.000539
Name: proportion, dtype: float64
```

In [33]:
```
cancelled_data['market_segment'].value_counts(normalize = True)
```

Out[33]:
```
market_segment
Online TA        0.471410
Groups           0.274705
Offline TA/TO    0.187467
Direct           0.042982
Corporate        0.022162
Aviation         0.001160
Complementary    0.000114
Name: proportion, dtype: float64
```

Online TravelAgents also invites highest cancellations.

It can be observed that the booking and cancellation are almost same for Online TravelAgents segment.

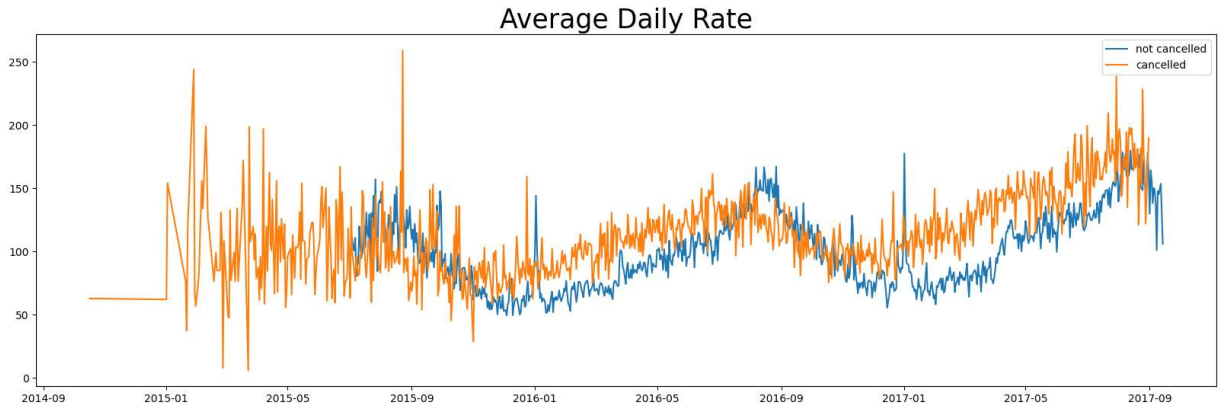
This can be possible due to false information of the hotels on the websites OR due to the surrounding of the hotel, etc.

In [34]:
```
cancelled_df_adr = cancelled_data.groupby('reservation_status_date')[['adr']].mean(
        # Grouping the 'cancelled_data' DataFrame by 'reservation_status_date' and

cancelled_df_adr.reset_index(inplace = True)
        # Reset the index of the DataFrame to make 'reservation_status_date' a regu

cancelled_df_adr.sort_values('reservation_status_date', inplace = True)
        # Sorting the DataFrame by 'reservation_status_date' in ascending order


not_cancelled_data = df[df['is_canceled'] == 0]
      # Filtering the DataFrame to include only non_canceled reservations

not_cancelled_df_adr = not_cancelled_data.groupby('reservation_status_date')[['adr'

not_cancelled_df_adr.reset_index(inplace = True)

not_cancelled_df_adr.sort_values('reservation_status_date', inplace = True)
```

The sorted DataFrame can be used for further analysis or visualization of the ADR trends for canceled reservations over time.

In [35]:
```
plt.figure(figsize = (20,6))

plt.title('Average Daily Rate', fontsize = 25)

plt.plot(not_cancelled_df_adr['reservation_status_date'],not_cancelled_df_adr['adr'
plt.plot(cancelled_df_adr['reservation_status_date'],cancelled_df_adr['adr'], label

       # Plotting the ADR trends for canceled and non_cancelled reservations

plt.legend()
```

Out[35]: <matplotlib.legend.Legend at 0x287d2462b80>



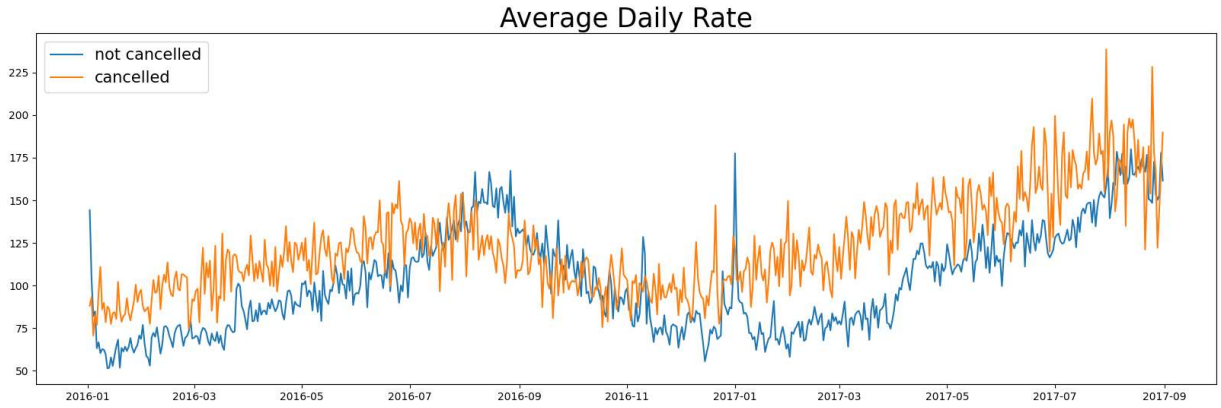
The resultant line plot is showing two lines representing the ADR trends over time for both canceled and not canceled reservations.

The x-axis represents the 'reservation_status_date', and the y-axis represents the ADR value.

The legend indicates which line corresponds to each reservation status.

In [36]:
```
# Filtering 'cancelled_df_adr' and 'not_cancelled_df_adr' to include data between J
# to avoid inconsistency which can be shown in the graph above.

cancelled_df_adr = cancelled_df_adr[(cancelled_df_adr['reservation_status_date']>'2

not_cancelled_df_adr = not_cancelled_df_adr[(not_cancelled_df_adr['reservation_stat
```

In [37]:
```
plt.figure(figsize = (20,6))

plt.title('Average Daily Rate', fontsize = 25)

plt.plot(not_cancelled_df_adr['reservation_status_date'],not_cancelled_df_adr['adr'
plt.plot(cancelled_df_adr['reservation_status_date'],cancelled_df_adr['adr'], label
            # Plotting the ADR trends for not_canceled and canceled reservations

plt.legend(fontsize = 15)

plt.show()
```



The resultant line plot shows two lines representing the ADR trends over time for both canceled and not canceled reservations.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: