Welcome to Hotel Booking Analysis!

ABC Travel's sales revenue reduced 16% after the customers left due to competitive rate. The management team immediately drafted an action plan, pulling together a team from Commercial, Strategy, and Business Analytics to tackle this issue.

How should ABC Travel protect the existing market share and increase the average daily rate forecast accuracy to increase sales by focusing on three key areas (Customer Strategy, Market Opportunities, and Product Divestment) by the end of this year?

The datasets provided by Kaggle recorded hotel booking data between July 2015 and August 2017 for two type of hotels (city hotel and resort hotel in Portugal.

Step 1: Import Libraries

Import the libraries I'll need for my analysis.

Matplotlib - This is Python's basic plotting library. The pyplot and dates function collections from matplotlib. The line **'%matplotlib inline'** make the graphs are easily included in this notebook.

Modify the matplotlib plot sizes so they're at a comfortable reading size by using the following:

import matplotlib as mpl

mpl.rcParams['figure.figsize'] = (20,5)

Seaborn - This library is to create aesthetically pleasing plots.

Pandas - This library is to view and manipulate the data in a tabular format.

statsmodels.api - This library is to create statistical models.

In [24]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.dates as md
%matplotlib inline
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
```

Step 2: Descriptive Statistics

i. Import the data Use the **header** argument to ensure the columns have meaningful names!

ii. Print descriptive statistics for each of the dataframes using **.describe()** and **.info()**

In [25]:
```
#Load data and print head of data
hotel=pd.read_csv("C:/Users/yingr/Desktop/hotel_bookings.csv", header=0)
```

```
hotel.head()
```

Out[25]:

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

5 rows × 32 columns

In [26]:

```
hotel.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
```

In [27]:
```
hotel.describe()
```

Out[27]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_mo
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000
mean	0.370416	104.011416	2016.156554	27.165173	15.798
std	0.482918	106.863097	0.707476	13.605138	8.780
min	0.000000	0.000000	2015.000000	1.000000	1.000
25%	0.000000	18.000000	2016.000000	16.000000	8.000
50%	0.000000	69.000000	2016.000000	28.000000	16.000
75%	1.000000	160.000000	2017.000000	38.000000	23.000
max	1.000000	737.000000	2017.000000	53.000000	31.000

In [28]:
```
#descibe the categorical data and see basic stats
hotel.describe(include="O")
```

Out[28]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	reserved_ro
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	

In [29]:
```
#check if there's NA
hotel.isnull().sum()
```

Out[29]:
```
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 [30]:

```
hotel.hist(figsize=(20,20))
plt.show()
```

Step 3: Data Cleansing

In [31]:
```
# check percentage of N/A
missing_percentage =round(hotel.isnull().sum()*100/len(hotel),2).reset_index()
missing_percentage.columns = ['column_name','missing_percentage']
missing_percentage = missing_percentage.sort_values('missing_percentage',ascending =Fal
missing_percentage
```

Out[31]:

	column_name	missing_percentage
24	company	94.31
23	agent	13.69
13	country	0.41
0	hotel	0.00
17	previous_cancellations	0.00

In [32]:
```
# drop company, agent columns due to large amount of N/A
import numpy as pd
hotel.drop(["company"], axis = 1, inplace = True)
hotel.drop(["agent"], axis = 1, inplace = True)
```

In [33]:
```
#find our how many bookings with number of children
hotel['children'].value_counts()
```

Out[33]:
```
0.0     110796
1.0       4861
2.0       3652
3.0         76
10.0         1
Name: children, dtype: int64
```

In [34]:
```
#most the bookings with no children, fill children na with 0
hotel["children"] = hotel["children"].fillna(value=0)
hotel['children'].isnull().sum()
```

Out[34]:
```
0
```

In [35]:
```
#forward fill country column
hotel['country'].ffill(axis = 0,inplace=True)
hotel['country'].isnull().sum()
```

Out[35]:
```
0
```

In [36]:
```
#create column name length of stay and drop length of stay is 0
hotel['length of stay'] = hotel['stays_in_weekend_nights'] + hotel['stays_in_week_night
hotel[hotel['length of stay']==0].shape[0]
hotel.drop(hotel[hotel['length of stay']==0].index, inplace=True)
```

In [37]:
```
#create column name sales
```

```
hotel['sales'] = hotel['adr'] * hotel['length of stay']
```

In [38]:
```
# drop adr negative value
hotel.drop(hotel[hotel.adr < 0].index, inplace=True)
#drop outlier more than 5000
hotel.drop(hotel[hotel.adr > 5000].index, inplace=True)
#drop 0 ocupied bookings
hotel.drop(hotel[hotel.adults+hotel.children+hotel.babies == 0].index, inplace=True)

hotel.isnull().sum()
```

Out[38]:
```
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
length of stay                  0
sales                           0
dtype: int64
```

In [39]:
```
#hotel.to_csv (r'C:\Users\yingr\Desktop\hotel_clean.csv', index = False, header=True)
```

Step 4: Explore Data Analysis

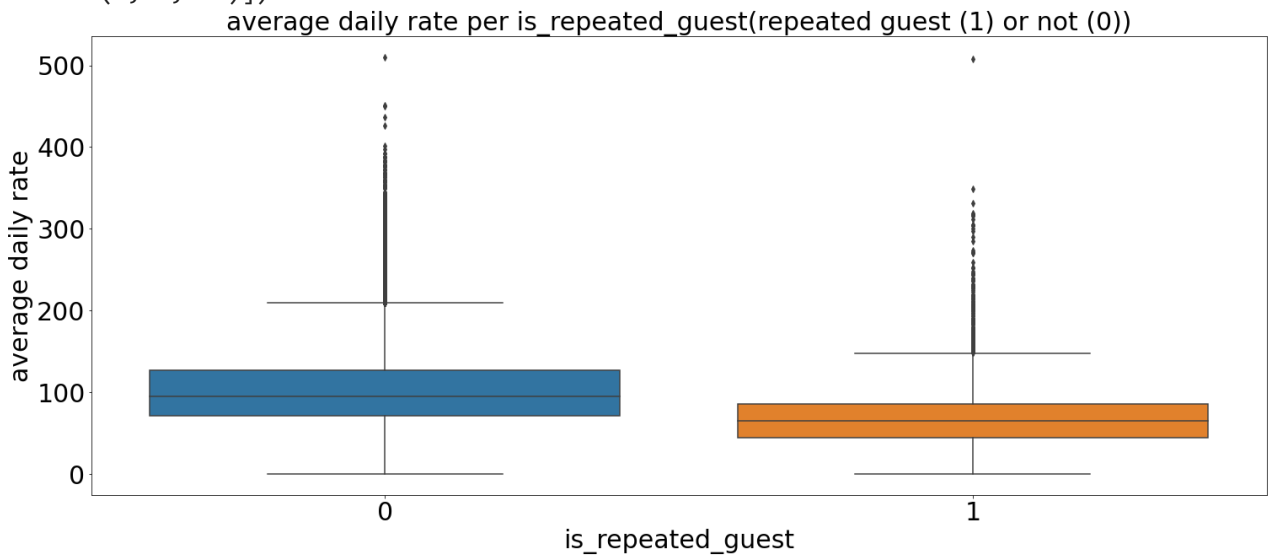
Customer strategy: Average Daily Rate based on customer behaviour

This part of the analysis will help our market department find patterns that unlock customer behaviors' mysteries to increase sales by identifying and discovering potential business.

In [40]:
```
#reate per repeated guest (1) or not (0)

plt.figure(figsize=(25,10))
sns.boxplot(x= "is_repeated_guest", y="adr", data = hotel)
plt.title('average daily rate per is_repeated_guest(repeated guest (1) or not (0))', fo
plt.xlabel('is_repeated_guest', fontsize=30)
plt.ylabel('average daily rate', fontsize=30)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
```

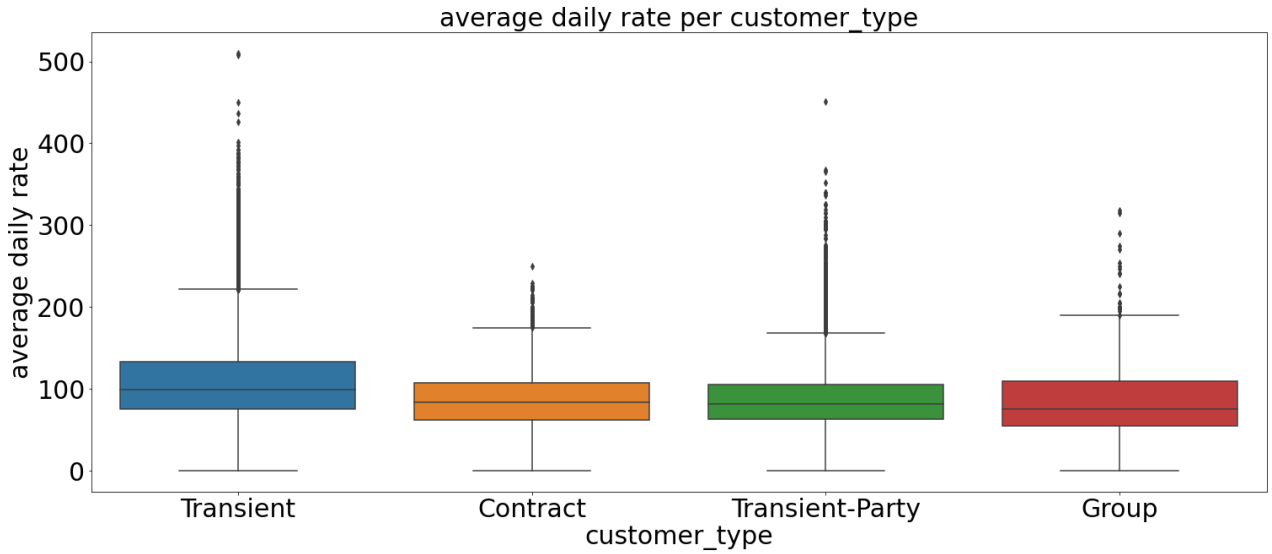
Out[40]:
```
(array([-100.,    0.,  100.,  200.,  300.,  400.,  500.,  600.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



* Is repeated guest based on average daily rate: **Repeated guest paying less than non repeated customer**

In [41]:
```
#rate per customer type
plt.figure(figsize=(25,10))
sns.boxplot(x= "customer_type", y="adr", data = hotel)
plt.title('average daily rate per customer_type', fontsize=30)
plt.xlabel('customer_type', fontsize=30)
plt.ylabel('average daily rate', fontsize=30)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
```

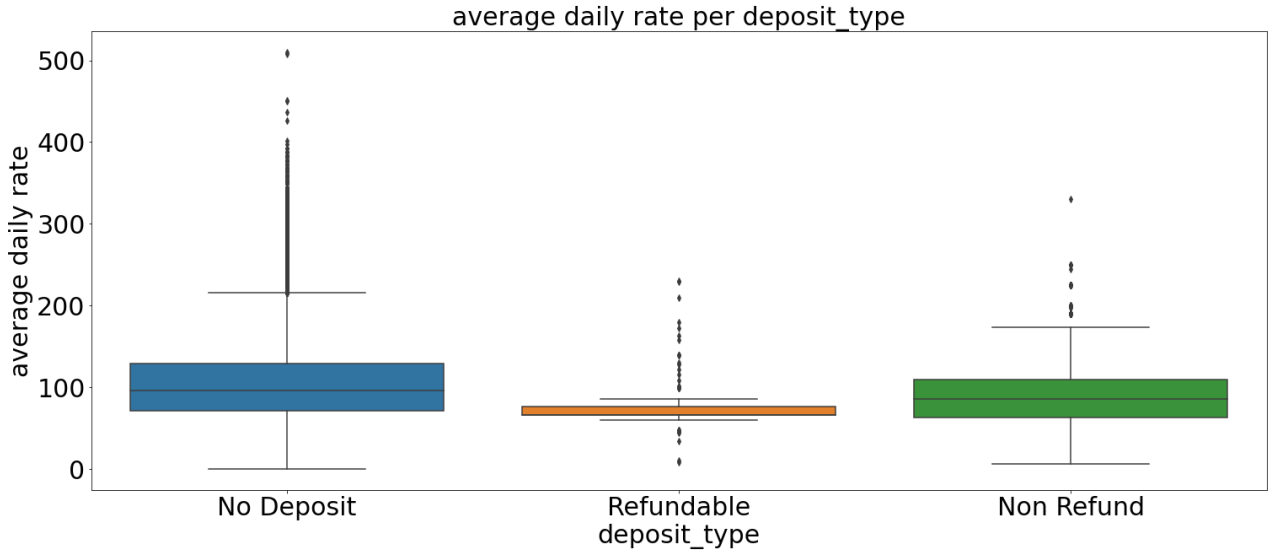
Out[41]:
```
(array([-100.,    0.,  100.,  200.,  300.,  400.,  500.,  600.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



* Customer type based on average daily rate: Median Transient customer (Transient – when the booking is not part of a group or contract and is not associated with other transient bookings) paying more than other types of customers.

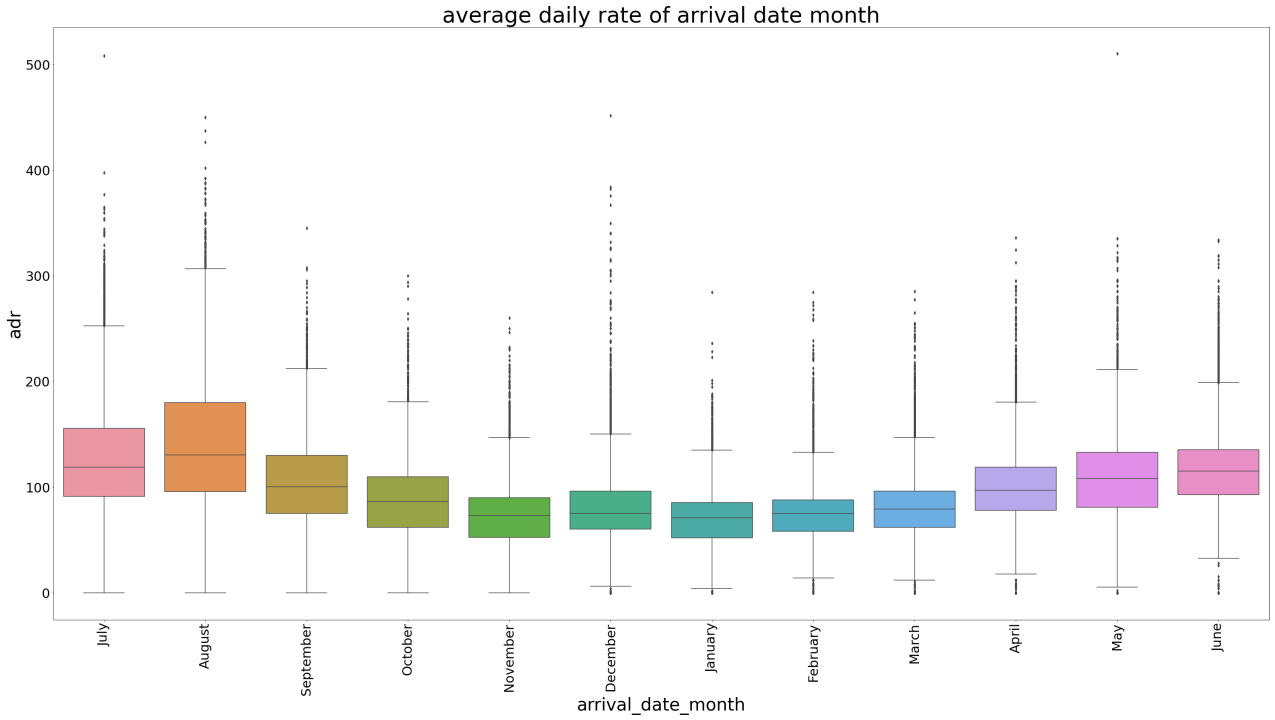
In [42]:
```
#rate per deposit type
plt.figure(figsize=(25,10))
sns.boxplot(x= "deposit_type", y="adr", data = hotel)
plt.title('average daily rate per deposit_type', fontsize=30)
plt.xlabel('deposit_type', fontsize=30)
plt.ylabel('average daily rate', fontsize=30)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
```

Out[42]:
```
(array([-100.,    0.,  100.,  200.,  300.,  400.,  500.,  600.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



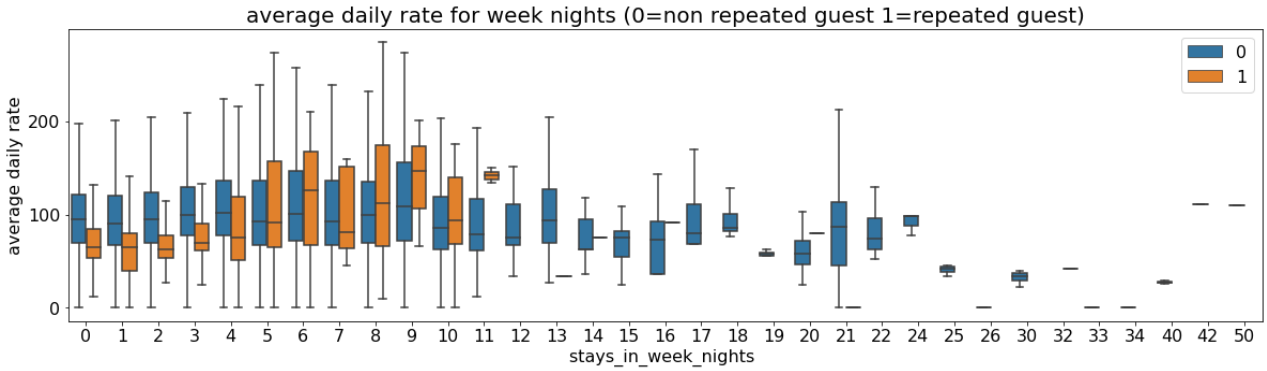
* Deposit type based on average daily rate: The median no deposit bookings pay higher rate than non refundable bookings. Refundable bookings paying the lowest rate.

In [44]:
```
#check seasonality of rate
sns.boxplot(x= "arrival_date_month", y="adr", data = hotel)
plt.title('average daily rate of arrival date month', fontsize=50)
plt.xlabel('arrival_date_month', fontsize=40)
plt.ylabel('adr', fontsize=40)
plt.xticks(rotation=90)
mpl.rcParams['figure.figsize'] = (50,25)
mpl.rcParams.update({'font.size': 30})
plt.xticks(rotation=90)
plt.show()
```



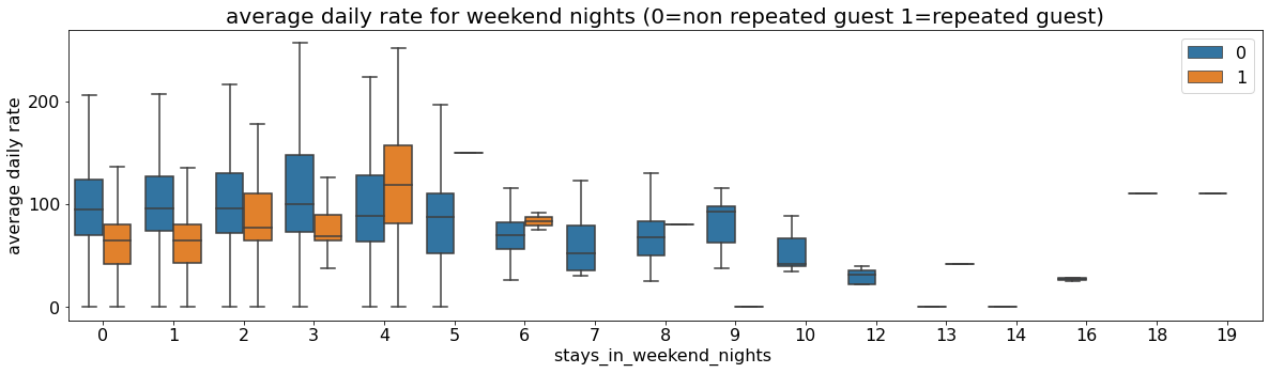
* Arrival date month based on average daily rate: The median rate in August is the highest. It's high in the summer months and low in the winter season.

In [68]:
```
# compare rate of stays on week nights and weekend nights group by if is repeated guest
# stays on week nights
plt.figure(figsize=(20,5))
sns.boxplot(x="stays_in_week_nights", y="adr", hue="is_repeated_guest", data=hotel, sho
plt.title('average daily rate for week nights (0=non repeated guest 1=repeated guest)',
plt.xlabel('stays_in_week_nights ', fontsize=16)
plt.ylabel('average daily rate', fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(loc='upper right', fontsize=16)
plt.show()
```



In [69]:

```
#stays on weekend nights group by if is repeated guest
plt.figure(figsize=(20,5))
sns.boxplot(x="stays_in_weekend_nights", y="adr", hue="is_repeated_guest", data=hotel,
plt.title('average daily rate for weekend nights (0=non repeated guest 1=repeated guest
plt.xlabel('stays_in_weekend_nights ', fontsize=16)
plt.ylabel('average daily rate', fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(loc='upper right', fontsize=16)
plt.show()
```



*** Stays in week and weekend nights group by if is repeated guest based on average daily rate: On weekend nights, repeated guests paying the lowest rate for staying 5 nights or less. Repeated guests rarely stay longer than 5 nights, and their average daily rate is higher than non repeated guests. A similar pattern showed the repeated guests paying the lowest rate for staying 5 nights or less on weeknights. For non repeated guests, the average daily rate is lower when they stay longer. In general, non repeated guest stay longer than repeated guests.**

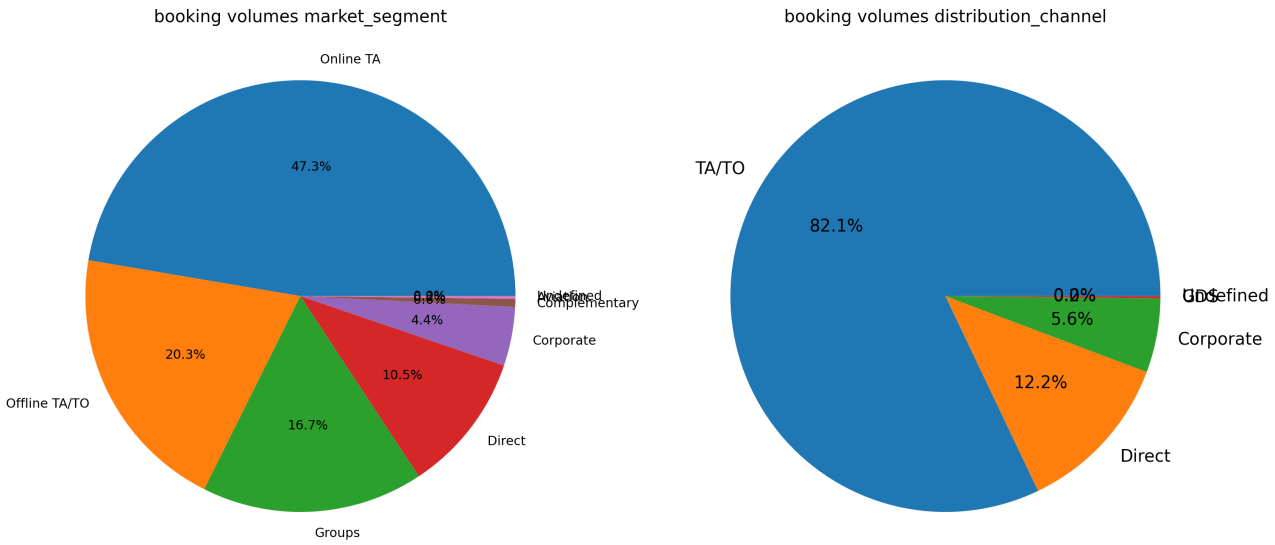
Recommendations: Maintain existing customers with high standard quality products and services, further keeping the current market share. Promoting non repeated Transient new customers based on their individual needs and preferences provides a more personalized travel experience. It is valuable for the customer and has enormous potential value for marketing more effectively.

Market Opportunity: booking volumes

In [70]:
```
fig, ax = plt.subplots(1, 2)
plt.figure(figsize=(30,20))

# booking volumes 7 nighters with meal type
value_meal = hotel['market_segment'].value_counts().values
label_meal=list(hotel['market_segment'].value_counts().index)
ax[0].pie(value_meal, labels=label_meal, autopct='%1.1f%%')
ax[0].set_title('booking volumes market_segment', fontsize=40)
mpl.rcParams.update({'font.size': 40})


# booking volumes 7 nighters with room type
value_room = hotel['distribution_channel'].value_counts().values
label_room=list(hotel['distribution_channel'].value_counts().index)
ax[1].pie(value_room, labels=label_room, autopct='%1.1f%%')
ax[1].set_title('booking volumes distribution_channel', fontsize=40)
mpl.rcParams.update({'font.size': 40})
```



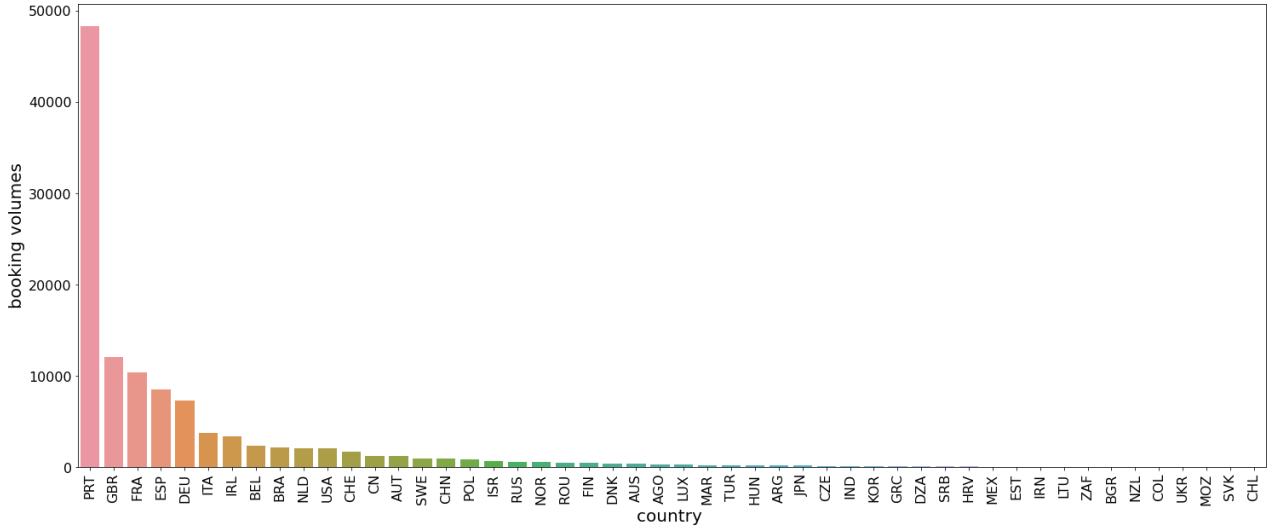

```
<Figure size 2160x1440 with 0 Axes>
```

In [ ]:
```
```

In [71]:
```
# top 50 countries with the most bookings
top50 = hotel["country"].value_counts().nlargest(50).astype(int)
top50.index
plt.figure(figsize=(25,10))
sns.barplot(x=top50.index, y=top50, data=hotel)
plt.xlabel('country ', fontsize=20)
plt.ylabel('booking volumes', fontsize=20)
plt.xticks(fontsize=16, rotation=90)
plt.yticks(fontsize=16)
```

Out[71]:
```
(array([    0., 10000., 20000., 30000., 40000., 50000., 60000.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
```

```
 Text(0, 0, ''),
 Text(0, 0, '')])
```

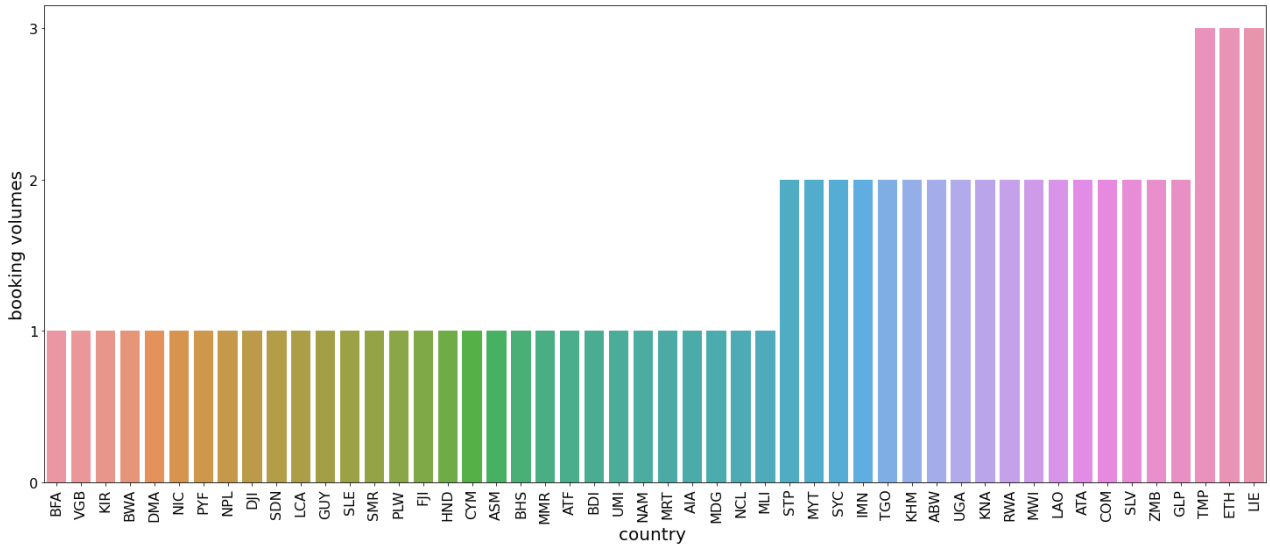


In [72]:

```
# botom 50 countries with the least bookings
botom50 = hotel["country"].value_counts().nsmallest(50).astype(int)
botom50.index
plt.figure(figsize=(25,10))
sns.barplot(x=botom50.index, y=botom50, data=hotel)
plt.xlabel('country ', fontsize=20)
plt.ylabel('booking volumes', fontsize=20)
plt.xticks(fontsize=16, rotation=90)
plt.yticks(fontsize=16)
```

Out[72]:

```
(array([0., 1., 2., 3., 4.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



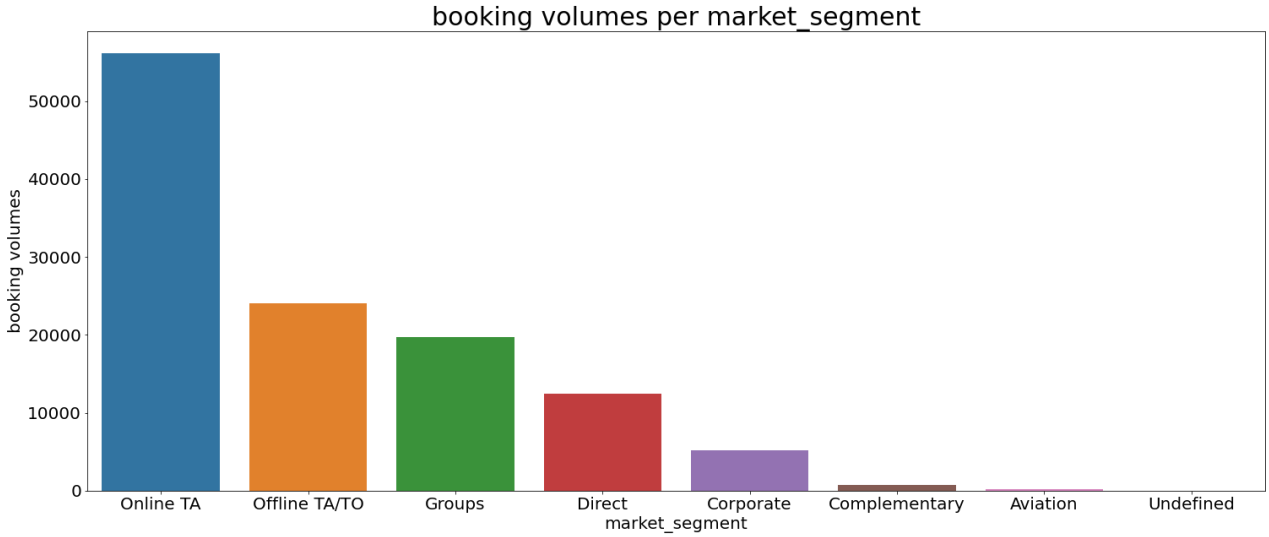
* Country based on booking volumes: Portugal (PRT) made the most bookings identify ABC Travel's majority business are from demestic.

In [73]:

```
#booking volumes per market segment
```

```
plt.figure(figsize=(25,10))
sns.countplot(x= "market_segment", data = hotel, order = hotel['market_segment'].value_
plt.title('booking volumes per market_segment', fontsize=30)
plt.xlabel('market_segment', fontsize=20)
plt.ylabel('booking volumes', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

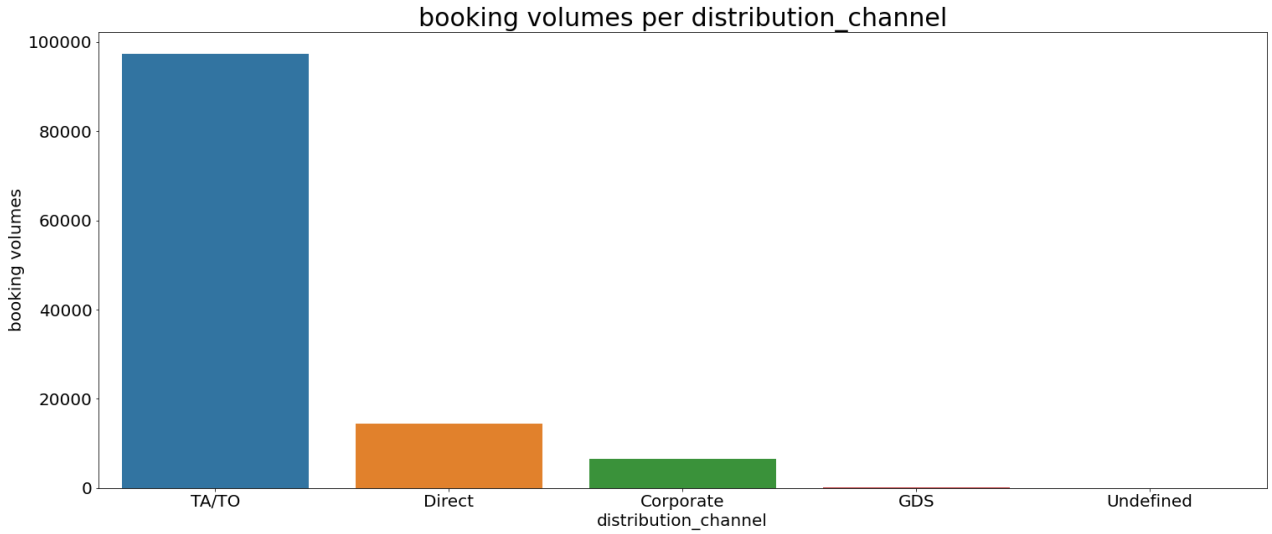
Out[73]:
```
(array([    0., 10000., 20000., 30000., 40000., 50000., 60000.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



* Market segment based on booking volumes: Online TA has the largest mount of bookings, OfflineTA/TO is on the second spot

In [74]:
```
#booking volumes per distribution channel
plt.figure(figsize=(25,10))
sns.countplot(x= "distribution_channel", data = hotel, order = hotel['distribution_chan
plt.title('booking volumes per distribution_channel', fontsize=30)
plt.xlabel('distribution_channel', fontsize=20)
plt.ylabel('booking volumes', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

Out[74]:
```
(array([     0.,  20000.,  40000.,  60000.,  80000., 100000., 120000.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



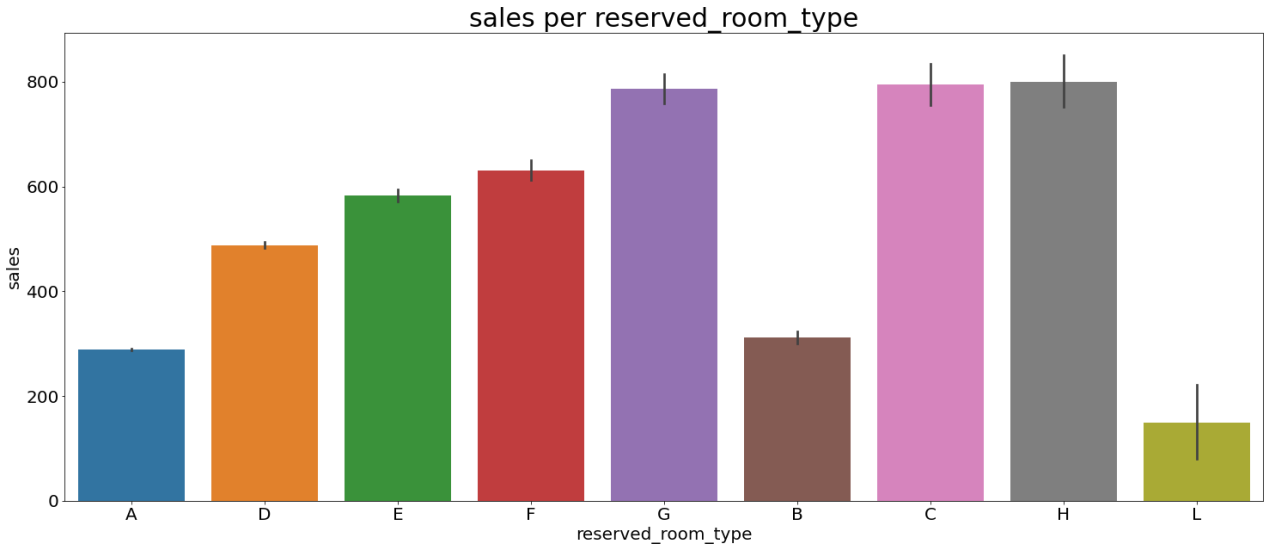
*** Distribution channel based on booking volumes: TA/TO bookings are the majority distribution channels**

Conclusions: Geographically, the majority of bookings are from Portugal through Online Travel Agency and tour operators. The promotional campaigns may target this type of customers. Recommend to continue the promotion in Portugal with a large number of bookings. Expand the market in UK and France booking directly to increase sales.

Product Divestment: Sales

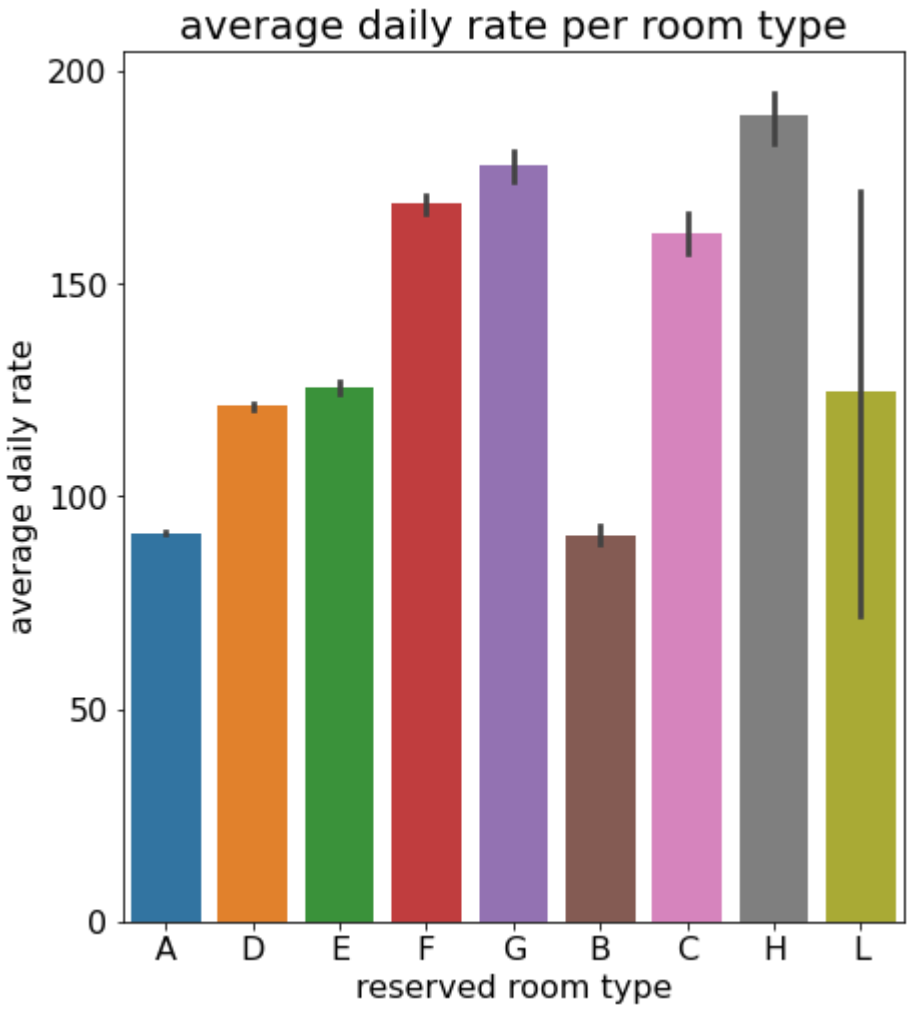
In [75]:
```
#sales per room type
plt.figure(figsize=(25,10))
sns.barplot(x= "reserved_room_type", y="sales", data = hotel, order = hotel['reserved_r
plt.title('sales per reserved_room_type', fontsize=30)
plt.xlabel('reserved_room_type', fontsize=20)
plt.ylabel('sales', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

Out[75]:
```
(array([   0.,  200.,  400.,  600.,  800., 1000.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



In [98]:
```
#average daily rate per room type
plt.figure(figsize=(7,8))
sns.barplot(x= "reserved_room_type", y="adr", data = hotel, order = hotel['reserved_roo
plt.title('average daily rate per room type', fontsize=20)
plt.xlabel('reserved room type', fontsize=16)
plt.ylabel('average daily rate', fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
```

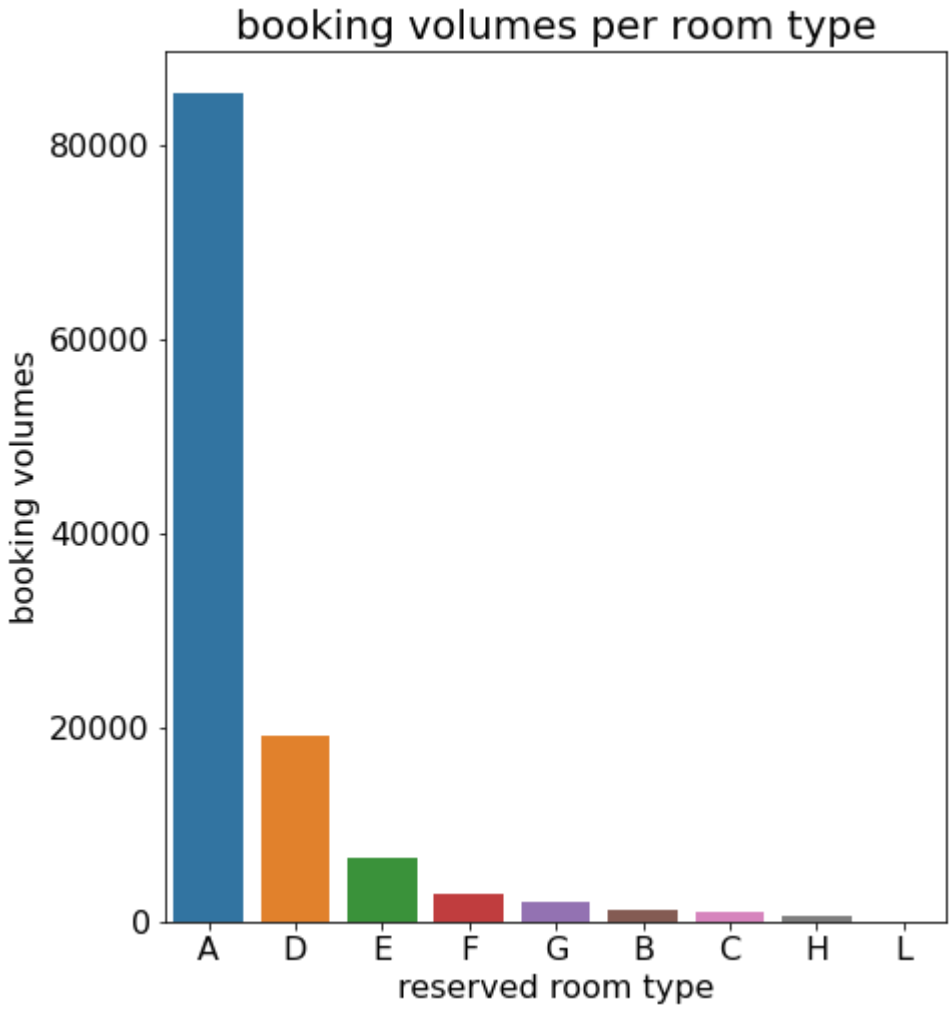
Out[98]:
```
(array([  0.,  50., 100., 150., 200., 250.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



* Room type based on sales: room type G, C, H have the highest sales and room type L has the least sales and bottom third average daily rate

In [99]:
```
#booking volumes per room type
plt.figure(figsize=(7,8))
sns.countplot(x= "reserved_room_type", data = hotel, order = hotel['reserved_room_type'
plt.title('booking volumes per room type', fontsize=20)
plt.xlabel('reserved room type', fontsize=16)
plt.ylabel('booking volumes', fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
```

Out[99]:
```
(array([     0.,  20000.,  40000.,  60000.,  80000., 100000.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



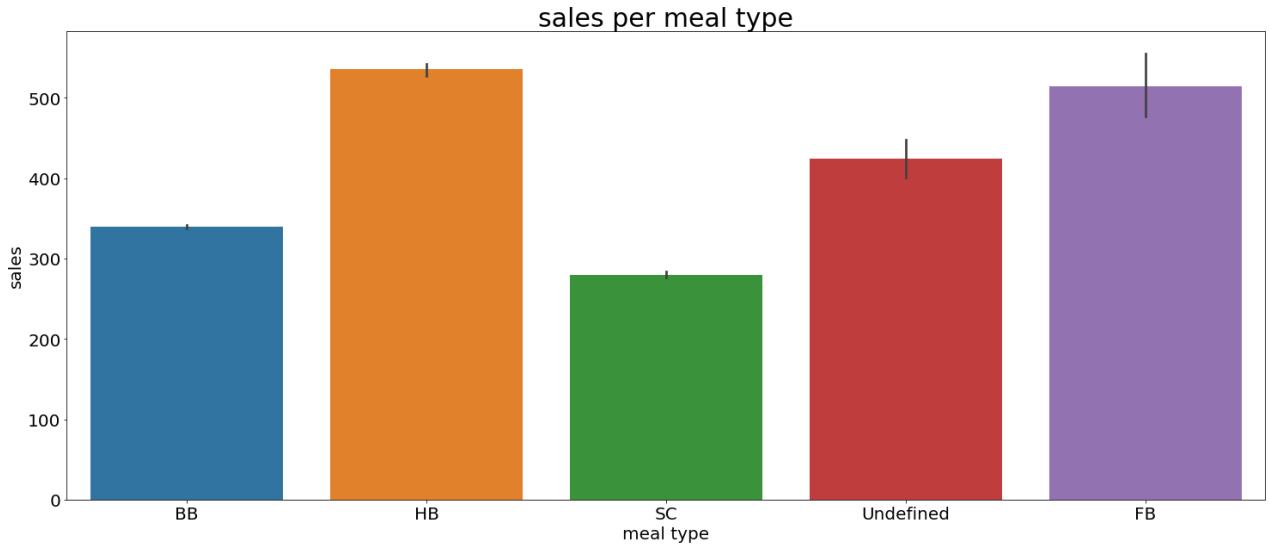
* Room type based on booking volumes: Room type A is the most popular room type. Room type L is the least popular.

In [100…

```
#sales per meal type
plt.figure(figsize=(25,10))
sns.barplot(x= "meal", y="sales", data = hotel, order = hotel['meal'].value_counts().in
plt.title('sales per meal type', fontsize=30)
plt.xlabel('meal type', fontsize=20)
plt.ylabel('sales', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

Out[100…

```
(array([  0., 100., 200., 300., 400., 500., 600.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



* Meal type based on sales: HB – Half board (breakfast and one other meal – usually dinner) with the highest sales, and SC - no meal package with the lowest sales

Recommendations: Room type L generated the least sales with bottom average daily rate for the company. Also, booked the least by the customers which means this room type is not attractive. Consider replacing a different room type. Meal type SC has the least sales, recommend to conduct a survey to investigate the potential products for this type of customers.

Step 5: Booking insights in country of Portugal

Previously, we identify Portugal (PRT) made the most bookings at ABC Travel. Further, we will investigate deeper for the data in the country of Portugal.

From here below, we are going to explore insights into bookings from Portugal.

In [101…

```
#subset data for country equal to PRT and non canceled bookings
hotel_prt = hotel[hotel.country == "PRT"].copy()
hotel_prt_nocxl = hotel_prt[hotel_prt['is_canceled']==0]
hotel_prt_nocxl.describe()
```

Out[101…

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month
count	20803.0	20803.000000	20803.000000	20803.000000	20803.000000
mean	0.0	49.380666	2015.985867	26.987261	16.022353
std	0.0	73.238720	0.728037	14.913192	8.795513
min	0.0	0.000000	2015.000000	1.000000	1.000000
25%	0.0	2.000000	2015.000000	13.000000	9.000000

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month
50%	0.0	13.000000	2016.000000	28.000000	16.000000
75%	0.0	69.000000	2017.000000	39.000000	24.000000
max	0.0	518.000000	2017.000000	53.000000	31.000000

In [102…

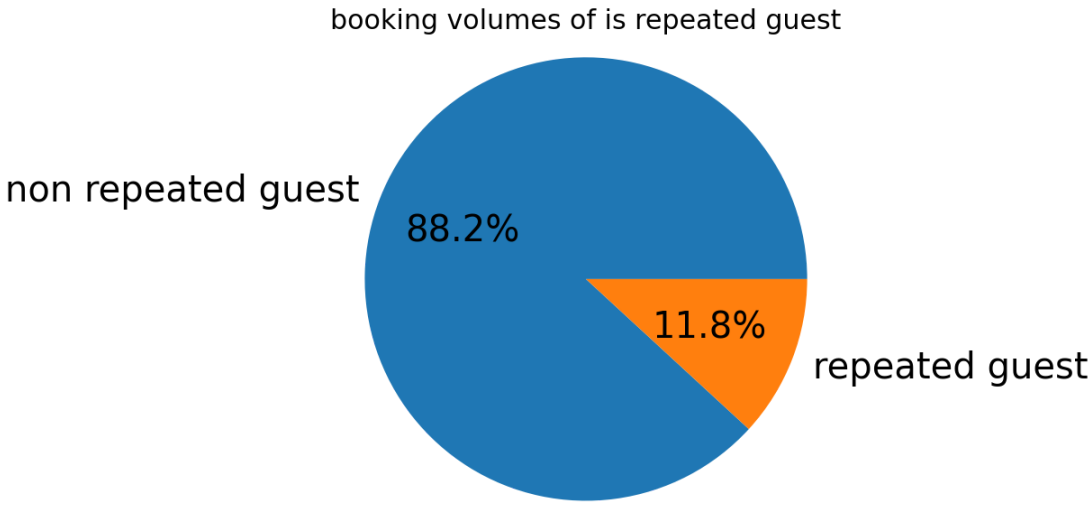
```
hotel_prt_nocxl.describe(include="O")
```

Out[102…

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	reserved_roo
count	20803	20803	20803	20803	20803	20803	
unique	2	12	5	1	7	5	
top	City Hotel	August	BB	PRT	Online TA	TA/TO	
freq	10574	2314	16594	20803	6360	12018	

In [103…

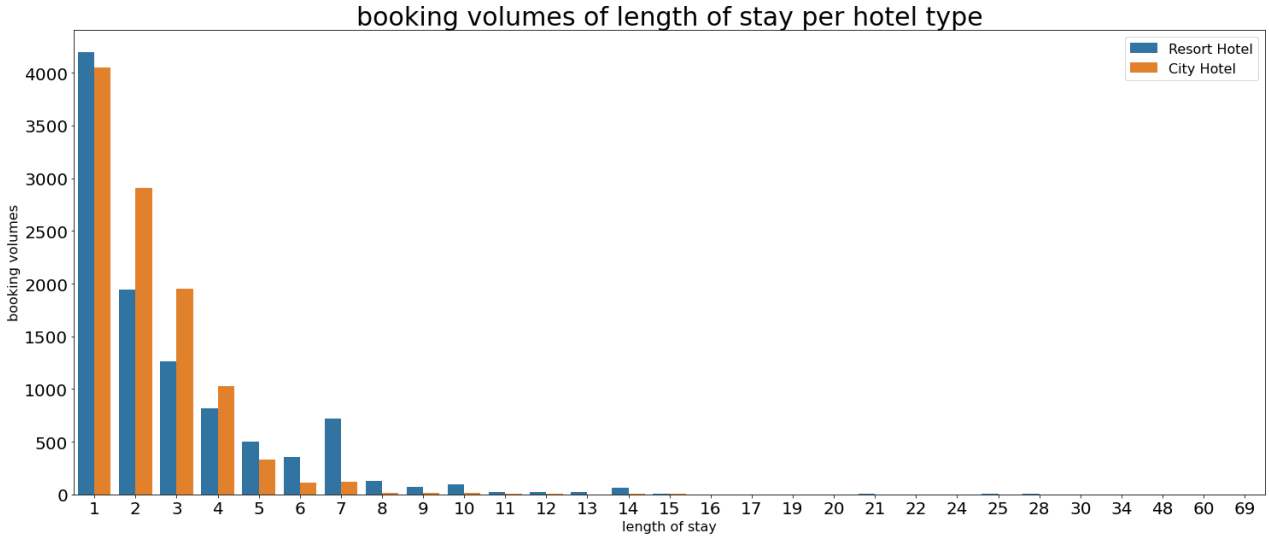
```
# booking volumes length of stay in PRT with repeated guest pct
value = hotel_prt_nocxl['is_repeated_guest'].value_counts().values
label=['non repeated guest', 'repeated guest']
plt.figure(figsize=(25,10))
plt.pie(value, labels=label, autopct='%1.1f%%')
plt.title('booking volumes of is repeated guest', fontsize=30)
plt.axis('equal')
mpl.rcParams.update({'font.size': 20})
```



88.2% of Portugal customers are not repeated guests, which means ABC Travel heavily relies on tourists and new customers. Recommend to develop a new program or membership to increase the share of repeated guests.

In [104…
```
# booking volumes length of stay in PRT by hotel type
plt.figure(figsize=(25,10))
sns.countplot(x= "length of stay", hue='hotel', data = hotel_prt_nocxl)
plt.title('booking volumes of length of stay per hotel type', fontsize=30)
plt.xlabel('length of stay', fontsize=16)
plt.ylabel('booking volumes', fontsize=16)
plt.legend(loc='upper right', fontsize=16)
```

Out[104… <matplotlib.legend.Legend at 0x147196b4460>



Portugal customers mostly like to stay for 1 or 2 nights at the city hotel. The resort hotel's booking volumes gradually decrease from 1-6 nights and sharply increase at 7 nights stay. We will investigate the 7 nights resort hotel peak further.

We will dig deeper for the 7 nighters' stay at the resort hotel from here below.

In [105…
```
#subset data for country equal to PRT and non canceled bookings with 7 nights at the re
hotel_prt_nocxl_7 = hotel_prt_nocxl[hotel_prt_nocxl['length of stay']==7]
hotel_prt_nocxl_7_resort=hotel_prt_nocxl_7[hotel_prt_nocxl_7['hotel']=='Resort Hotel']
hotel_prt_nocxl_7_resort.describe()
```

Out[105…

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	s
count	716.0	716.000000	716.000000	716.000000	716.000000	
mean	0.0	121.530726	2015.960894	30.004190	15.104749	
std	0.0	82.971546	0.816701	7.783521	8.585004	
min	0.0	0.000000	2015.000000	2.000000	1.000000	
25%	0.0	55.750000	2015.000000	28.000000	8.000000	
50%	0.0	111.500000	2016.000000	32.000000	16.000000	
75%	0.0	174.000000	2017.000000	34.000000	22.000000	

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	s
max	0.0	351.000000	2017.000000	52.000000	31.000000	

In [106…

```
hotel_prt_nocxl_7_resort.describe(include='O')
```

Out[106…

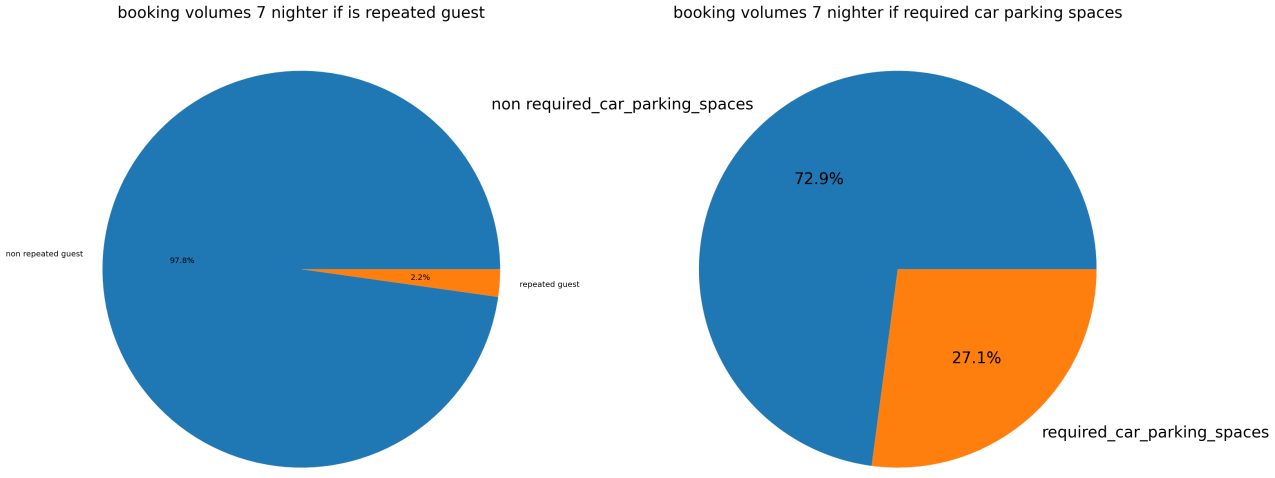
	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	reserved_roon
count	716	716	716	716	716	716	
unique	1	12	5	1	5	3	
top	Resort Hotel	August	BB	PRT	Online TA	TA/TO	
freq	716	316	318	716	238	494	

In [107…

```
#plot booking volumes pct w repeated guest and car space required
fig, ax = plt.subplots(1, 2)
plt.figure(figsize=(30,20))

# booking volumes 7 nighters with repeated guest pct
value_repeat = hotel_prt_nocxl_7_resort['is_repeated_guest'].value_counts().values
label_repeat=['non repeated guest', 'repeated guest']
ax[0].pie(value_repeat, labels=label_repeat, autopct='%1.1f%%')
ax[0].set_title('booking volumes 7 nighter if is repeated guest', fontsize=40)
mpl.rcParams.update({'font.size': 40})


# booking volumes 7 nighters with car spaces required
value_car = hotel_prt_nocxl_7_resort['required_car_parking_spaces'].value_counts().valu
label_car=['non required_car_parking_spaces', 'required_car_parking_spaces']
ax[1].pie(value_car, labels=label_car, autopct='%1.1f%%')
ax[1].set_title('booking volumes 7 nighter if required car parking spaces', fontsize=40
mpl.rcParams.update({'font.size': 40})
```




```
<Figure size 2160x1440 with 0 Axes>
```

The 7 nighters' stay at the resort hotel 98% are new customers. 27% require car parking spaces, and 73% of customers will need on-site transportations. It is an excellent opportunity to expand transportation related service products.

In [108…

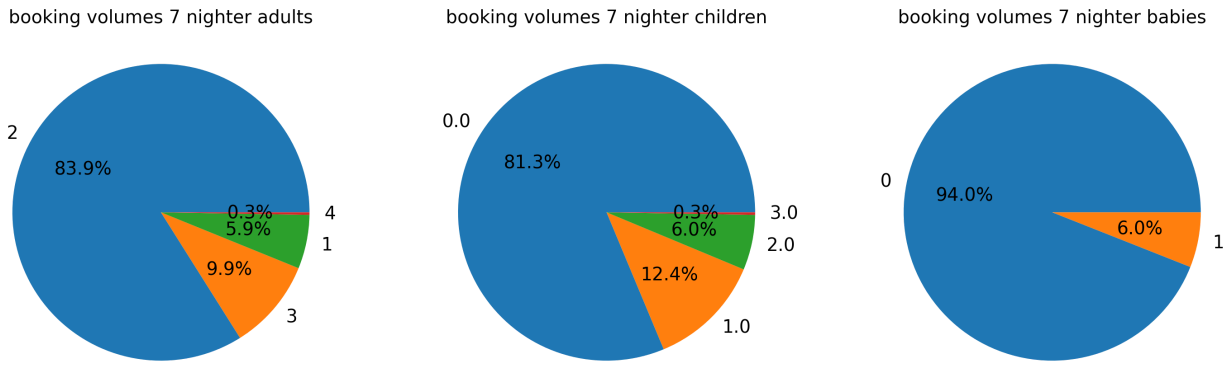
```
# booking volumes 7 nighters occupy customers
fig, ax = plt.subplots(1, 3)
plt.figure(figsize=(30,20))

#adults
value_adults = hotel_prt_nocxl_7_resort['adults'].value_counts().values
label_adults=hotel_prt_nocxl_7_resort['adults'].value_counts().index
ax[0].pie(value_adults, labels=label_adults, autopct='%1.1f%%')
ax[0].set_title('booking volumes 7 nighter adults', fontsize=40)
mpl.rcParams.update({'font.size': 40})

#children
value_children = hotel_prt_nocxl_7_resort['children'].value_counts().values
label_children=hotel_prt_nocxl_7_resort['children'].value_counts().index
plt.figure(figsize=(25,10))
ax[1].pie(value_children, labels=label_children, autopct='%1.1f%%')
ax[1].set_title('booking volumes 7 nighter children', fontsize=40)
mpl.rcParams.update({'font.size': 40})


#babies
value_babies = hotel_prt_nocxl_7_resort['babies'].value_counts().values
label_babies=hotel_prt_nocxl_7_resort['babies'].value_counts().index
plt.figure(figsize=(25,10))
ax[2].pie(value_babies, labels=label_babies, autopct='%1.1f%%')
ax[2].set_title('booking volumes 7 nighter babies', fontsize=40)
mpl.rcParams.update({'font.size': 40})


plt.tight_layout()
```




```
<Figure size 2160x1440 with 0 Axes>
<Figure size 1800x720 with 0 Axes>
<Figure size 1800x720 with 0 Axes>
```

The 7 nighters' stay at the resort hotel more than 81% occupied by 2 adults

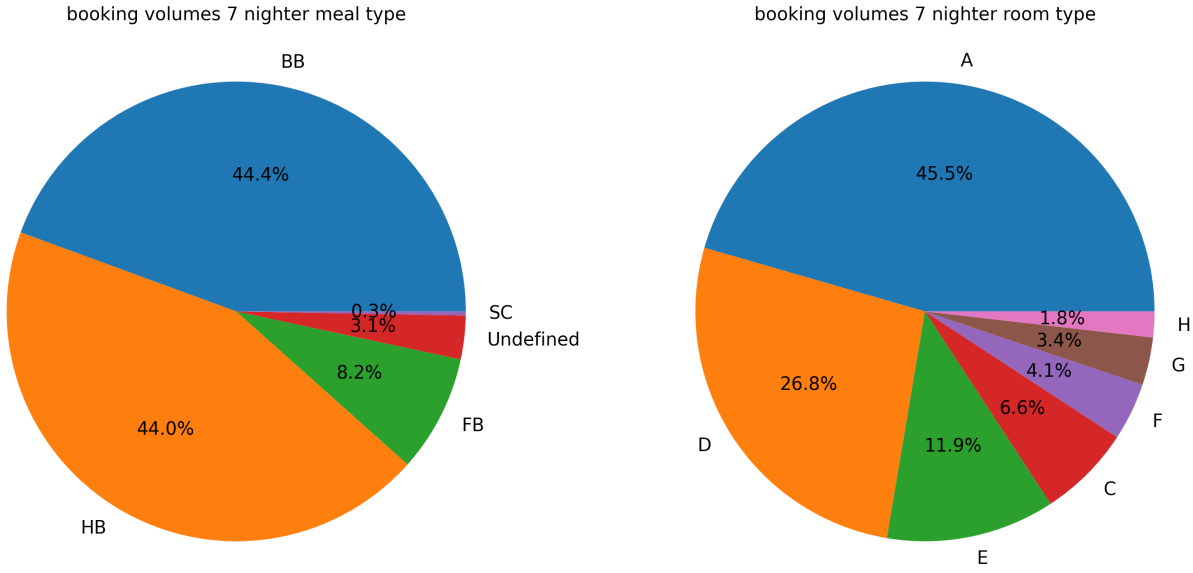
In [109…

```
#booking volumes pct of meal type and room type
```

```
fig, ax = plt.subplots(1, 2)
plt.figure(figsize=(30,20))

# booking volumes 7 nighters with meal type
value_meal = hotel_prt_nocxl_7_resort['meal'].value_counts().values
label_meal=list(hotel_prt_nocxl_7_resort['meal'].value_counts().index)
ax[0].pie(value_meal, labels=label_meal, autopct='%1.1f%%')
ax[0].set_title('booking volumes 7 nighter meal type', fontsize=40)
mpl.rcParams.update({'font.size': 40})


# booking volumes 7 nighters with room type
value_room = hotel_prt_nocxl_7_resort['reserved_room_type'].value_counts().values
label_room=list(hotel_prt_nocxl_7_resort['reserved_room_type'].value_counts().index)
ax[1].pie(value_room, labels=label_room, autopct='%1.1f%%')
ax[1].set_title('booking volumes 7 nighter room type', fontsize=40)
mpl.rcParams.update({'font.size': 40})
```




```
<Figure size 2160x1440 with 0 Axes>
```

The resort hotel's 7 nighters' stay booked either Bed & Breakfast or Half Board meal packages and reserve room type A or D.

In [110…

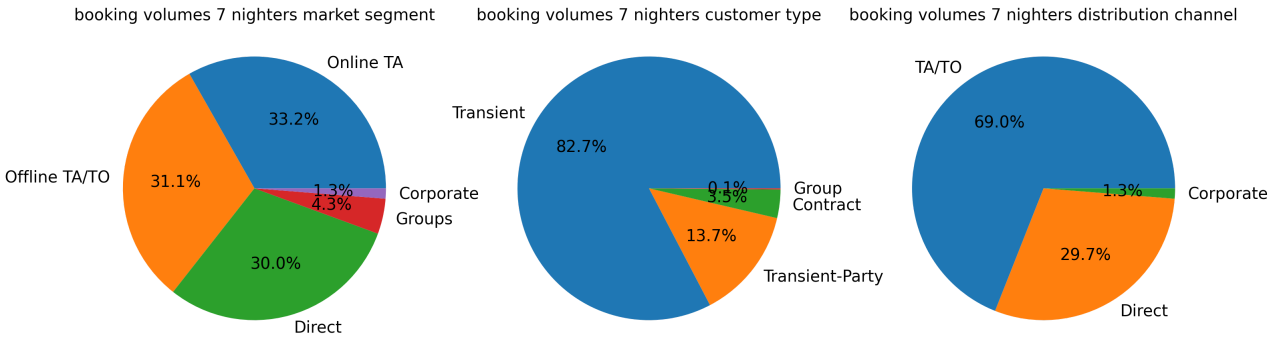
```
# booking volumes pct marekt segment/customer type/distributin channel
fig, ax = plt.subplots(1, 3)
plt.figure(figsize=(30,20))

# booking volumes 7 nighters market_segment
value_market = hotel_prt_nocxl_7_resort['market_segment'].value_counts().values
label_market=list(hotel_prt_nocxl_7_resort['market_segment'].value_counts().index)
ax[0].pie(value_market, labels=label_market, autopct='%1.1f%%')
ax[0].set_title('booking volumes 7 nighters market segment', fontsize=40)
mpl.rcParams.update({'font.size': 40})


# booking volumes 7 nighters customer_type
value_customer = hotel_prt_nocxl_7_resort['customer_type'].value_counts().values
label_customer=list(hotel_prt_nocxl_7_resort['customer_type'].value_counts().index)
ax[1].pie(value_customer, labels=label_customer, autopct='%1.1f%%')
```

```
ax[1].set_title('booking volumes 7 nighters customer type', fontsize=40)
mpl.rcParams.update({'font.size': 40})


# booking volumes 7 nighters distribution channel
value_channel = hotel_prt_nocxl_7_resort['distribution_channel'].value_counts().values
label_channel=list(hotel_prt_nocxl_7_resort['distribution_channel'].value_counts().inde
ax[2].pie(value_channel, labels=label_channel, autopct='%1.1f%%')
ax[2].set_title('booking volumes 7 nighters distribution channel', fontsize=40)
mpl.rcParams.update({'font.size': 40})
```



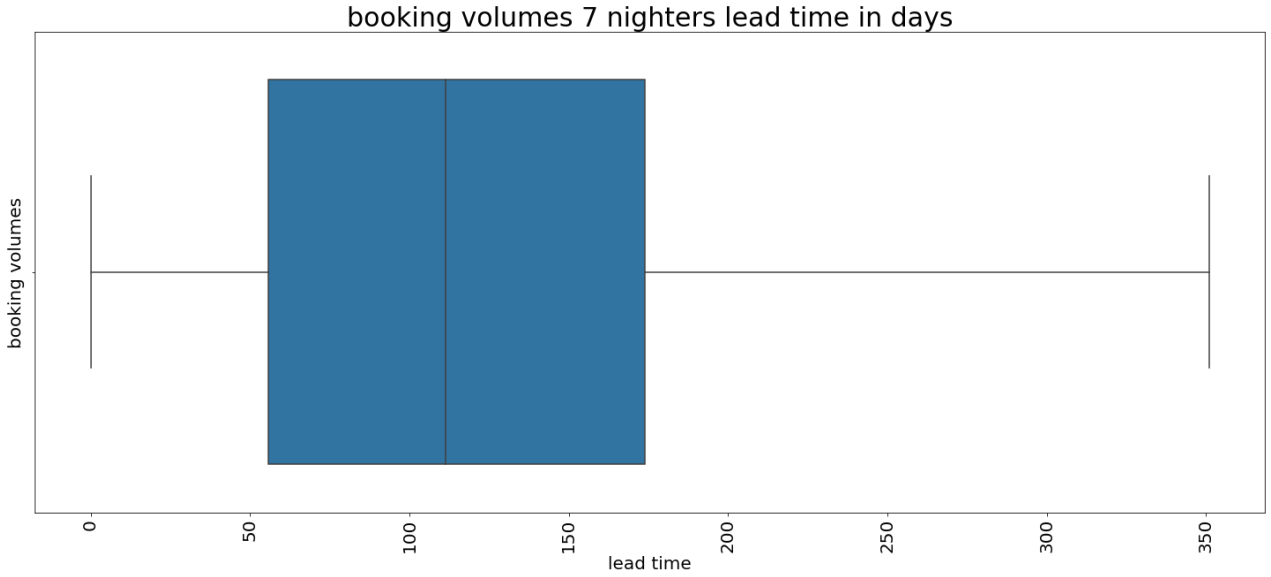

```
<Figure size 2160x1440 with 0 Axes>
```

The 7 nighters' stay at the resort hotel like to make bookings through an online Travel agent or offline Travel agent/tour operators, or direct. 82.7% are Transient new customers.

In [111…

```
#booking volumes per lead time
plt.figure(figsize=(25,10))
sns.boxplot(x= "lead_time", data = hotel_prt_nocxl_7_resort)
plt.title('booking volumes 7 nighters lead time in days', fontsize=30)
plt.xlabel('lead time', fontsize=20)
plt.ylabel('booking volumes', fontsize=20)
plt.xticks(fontsize=20, rotation=90)
plt.yticks(fontsize=20)
```

Out[111… `(array([0]), [Text(0, 0, '')])`



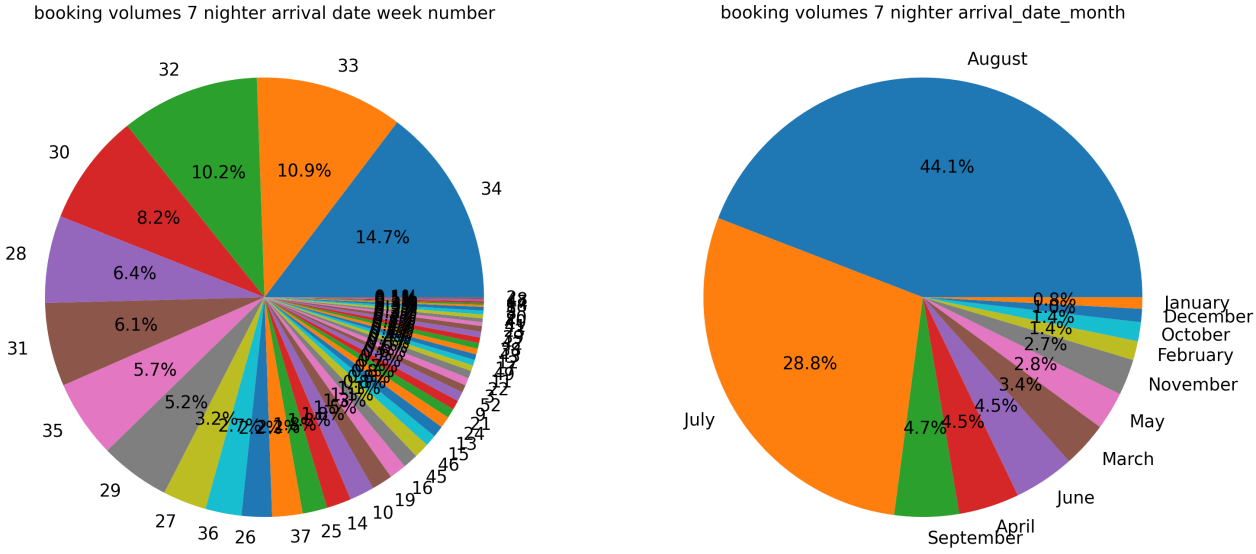
The reservation of the 7 nighters' stay at the resort hotel booked 60-175 days before traveling means the customers are well planned.

In [112…

```
# booking volumes travel weeks and month
fig, ax = plt.subplots(1, 2)
plt.figure(figsize=(30,20))

# booking volumes 7 nighters with arrival_date_week_number
value_week = hotel_prt_nocxl_7_resort['arrival_date_week_number'].value_counts().values
label_week=list(hotel_prt_nocxl_7_resort['arrival_date_week_number'].value_counts().ind
ax[0].pie(value_week, labels=label_week, autopct='%1.1f%%')
ax[0].set_title('booking volumes 7 nighter arrival date week number', fontsize=40)
mpl.rcParams.update({'font.size': 40})


# booking volumes 7 nighters with arrivel_date_month
value_month = hotel_prt_nocxl_7_resort['arrival_date_month'].value_counts().values
label_month=list(hotel_prt_nocxl_7_resort['arrival_date_month'].value_counts().index)
ax[1].pie(value_month, labels=label_month, autopct='%1.1f%%')
ax[1].set_title('booking volumes 7 nighter arrival_date_month', fontsize=40)
mpl.rcParams.update({'font.size': 40})
```



<Figure size 2160x1440 with 0 Axes>

The 7 nighters' stay at resort hotel 73% of customers arrive in August or July, especially in the second or third weeks of August.

In summary: Portugal customers who booked the 7 nights package at the resort hotel most likely are the new transient vocational customers. They are not local. They will need onsite transportation and occupied by 2 adults. They book room types A or D with the BB or HB meal plan through TA/TO or direct around 110 days ahead of time.

Step 6: Modelling

Encoder: convert categorical variables to numeric dummy variables

In [113…

```
hotel=hotel.drop(['sales','length of stay'], axis = 1)
```

In [114…

```
from sklearn.preprocessing import LabelEncoder
#hotel
le = LabelEncoder()
le.fit(hotel.hotel.drop_duplicates())
hotel.hotel = le.transform(hotel.hotel)
# arrival_date_month
le.fit(hotel.arrival_date_month.drop_duplicates())
hotel.arrival_date_month = le.transform(hotel.arrival_date_month)
#meal
le.fit(hotel.meal.drop_duplicates())
hotel.meal = le.transform(hotel.meal)
#country
le.fit(hotel.country.drop_duplicates())
hotel.country = le.transform(hotel.country)
#market_segment
le.fit(hotel.market_segment .drop_duplicates())
hotel.market_segment  = le.transform(hotel.market_segment )
#distribution_channel
le.fit(hotel.distribution_channel.drop_duplicates())
hotel.distribution_channel = le.transform(hotel.distribution_channel)
#reserved_room_type
le.fit(hotel.reserved_room_type.drop_duplicates())
hotel.reserved_room_type = le.transform(hotel.reserved_room_type)
#assigned_room_type
le.fit(hotel.assigned_room_type.drop_duplicates())
hotel.assigned_room_type= le.transform(hotel.assigned_room_type)
#deposit_type
le.fit(hotel.deposit_type.drop_duplicates())
hotel.deposit_type = le.transform(hotel.deposit_type)
#customer_type
le.fit(hotel.customer_type .drop_duplicates())
hotel.customer_type = le.transform(hotel.customer_type )
#reservation_status
le.fit(hotel.reservation_status.drop_duplicates())
hotel.reservation_status = le.transform(hotel.reservation_status)
#reservation_status_date
le.fit(hotel.reservation_status_date.drop_duplicates())
hotel.reservation_status_date = le.transform(hotel.reservation_status_date)
print(hotel.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 118563 entries, 2 to 119389
Data columns (total 30 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           118563 non-null  int32
 1   is_canceled                     118563 non-null  int64
 2   lead_time                       118563 non-null  int64
 3   arrival_date_year               118563 non-null  int64
 4   arrival_date_month              118563 non-null  int32
 5   arrival_date_week_number        118563 non-null  int64
 6   arrival_date_day_of_month       118563 non-null  int64
 7   stays_in_weekend_nights         118563 non-null  int64
 8   stays_in_week_nights            118563 non-null  int64
 9   adults                          118563 non-null  int64
 10  children                        118563 non-null  float64
 11  babies                          118563 non-null  int64
```

```
 12  meal                            118563 non-null  int32
 13  country                         118563 non-null  int32
 14  market_segment                  118563 non-null  int32
 15  distribution_channel            118563 non-null  int32
 16  is_repeated_guest               118563 non-null  int64
 17  previous_cancellations          118563 non-null  int64
 18  previous_bookings_not_canceled  118563 non-null  int64
 19  reserved_room_type              118563 non-null  int32
 20  assigned_room_type              118563 non-null  int32
 21  booking_changes                 118563 non-null  int64
 22  deposit_type                    118563 non-null  int32
 23  days_in_waiting_list            118563 non-null  int64
 24  customer_type                   118563 non-null  int32
 25  adr                             118563 non-null  float64
 26  required_car_parking_spaces     118563 non-null  int64
 27  total_of_special_requests       118563 non-null  int64
 28  reservation_status              118563 non-null  int32
 29  reservation_status_date         118563 non-null  int32
dtypes: float64(2), int32(12), int64(16)
memory usage: 27.6 MB
None
```

In [115…

```
#get correlation of all the variables
hotel.corr()['adr'].sort_values()
```

Out[115…

```
is_repeated_guest               -0.118314
arrival_date_month              -0.116425
country                         -0.115275
deposit_type                    -0.103710
hotel                           -0.098088
customer_type                   -0.083509
previous_bookings_not_canceled  -0.076978
lead_time                       -0.076636
previous_cancellations          -0.071152
days_in_waiting_list            -0.044756
reservation_status              -0.043164
booking_changes                  0.028772
babies                           0.030988
arrival_date_day_of_month        0.031056
is_canceled                      0.040142
stays_in_weekend_nights          0.044008
stays_in_week_nights             0.056701
required_car_parking_spaces      0.058567
meal                             0.066199
arrival_date_week_number         0.083330
distribution_channel             0.092047
total_of_special_requests        0.183051
arrival_date_year                0.208104
adults                           0.235666
market_segment                   0.245137
reservation_status_date          0.279922
assigned_room_type               0.304810
children                         0.345607
reserved_room_type               0.421187
adr                              1.000000
Name: adr, dtype: float64
```

In [120…

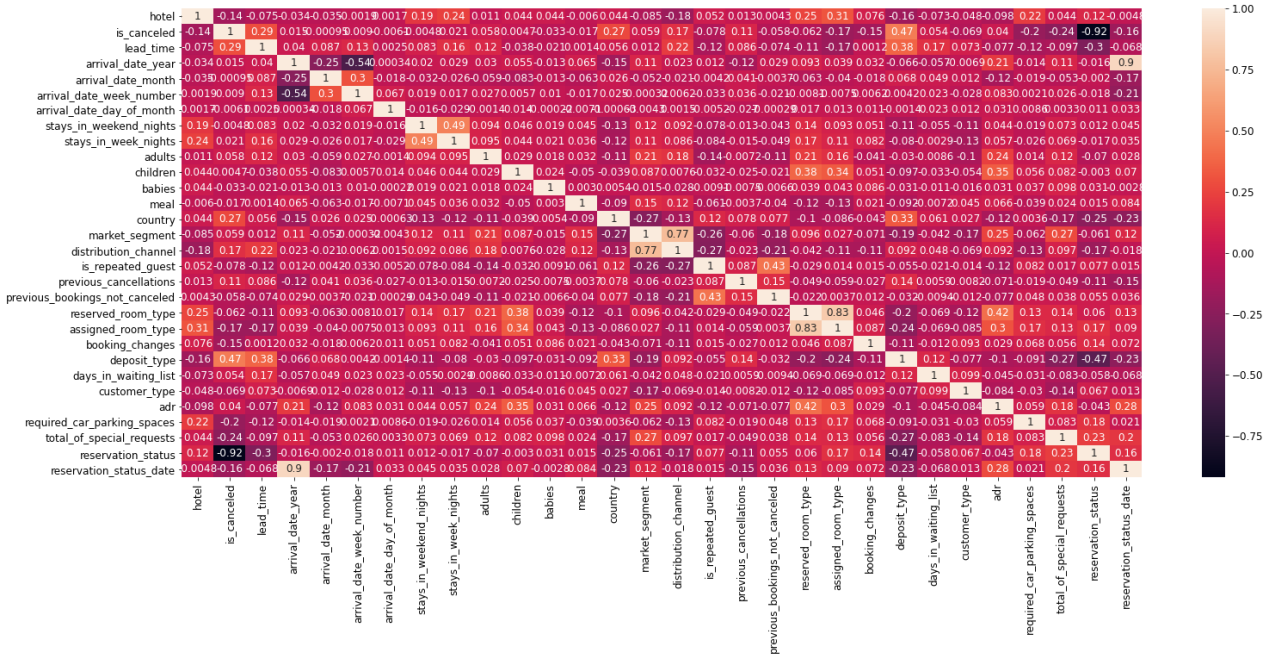
```
import seaborn as sns
```

In [121…

```
cov_matrix=hotel.corr()
```

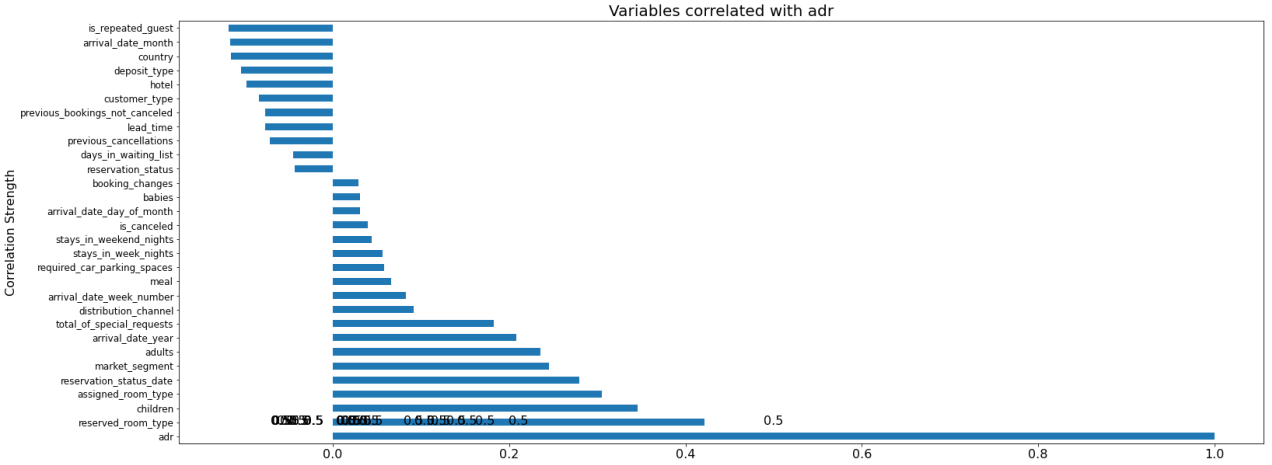
In [122…

```
plt.figure(figsize=(25,10))
sns.heatmap(hotel.corr(),annot=True)
mpl.rcParams.update({'font.size': 12})
```



In [123…

```
#created sorted data based on previous step correlation matrix
a=cov_matrix.sort_values("adr", ascending=False)
#plot bar chart
plt.figure(figsize=(25,10))
b=a['adr'].plot(kind='barh')
plt.title("Variables correlated with adr", fontsize=20)
plt.ylabel("Correlation Strength", fontsize=16)
plt.xticks(rotation=0, fontsize=16)
#add label on the chart
for i in b.patches:
    b.annotate(format(i.get_height(), '.1f'),
                   (i.get_x() + i.get_width() / 2., i.get_height()),
                   ha = 'center', va = 'center',
                   xytext = (0, 10), fontsize=16,
                   textcoords = 'offset points')
mpl.rcParams.update({'font.size': 12})
plt.show()
```



Choose the best model

In [124…

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.inspection import permutation_importance
```

In [125…

```
X = hotel.drop(['adr'], axis = 1)
y = hotel['adr']
```

In [126…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0

# model1 linear regression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
model1_y_pred = regressor.predict(X_test)

print('MAE:', metrics.mean_absolute_error(y_test, model1_y_pred).round(3))
print('MSE:', metrics.mean_squared_error(y_test, model1_y_pred).round(3))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, model1_y_pred)).round(3))
print('RSquared:', r2_score(y_test, model1_y_pred).round(3))
```

```
MAE: 27.026
MSE: 1369.087
RMSE: 37.001
RSquared: 0.393
```

In [127…

```
#model2 ridge regression
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)

model2_y_pred = ridge.predict(X_test)

print('MAE:', metrics.mean_absolute_error(y_test, model2_y_pred).round(3))
print('MSE:', metrics.mean_squared_error(y_test, model2_y_pred).round(3))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, model2_y_pred)).round(3))
print('RSquared:', r2_score(y_test, model2_y_pred).round(3))
```

```
MAE: 27.026
MSE: 1369.086
RMSE: 37.001
RSquared: 0.393
```

In [128…

```
## model 3 Lasso Regression
lasso = Lasso(alpha=0.1)
```

```
lasso.fit(X_train, y_train)

model3_y_pred = lasso.predict(X_test)

print('MAE:', metrics.mean_absolute_error(y_test, model3_y_pred ).round(3))
print('MSE:', metrics.mean_squared_error(y_test, model3_y_pred ).round(3))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, model3_y_pred )).round(3))
print('RSquared:', r2_score(y_test, model3_y_pred ).round(3))
```

```
MAE: 27.032
MSE: 1370.715
RMSE: 37.023
RSquared: 0.392
```

In [129…

```
#model 4 random forest regression
rfr = RandomForestRegressor(n_estimators = 100, random_state = 42)
# fit the regressor with x and y data
rfr.fit(X_train, y_train)
#y_pred
model4_y_pred = rfr.predict(X_test)
print('MAE:', metrics.mean_absolute_error(y_test, model4_y_pred).round(3))
print('MSE:', metrics.mean_squared_error(y_test, model4_y_pred).round(3))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, model4_y_pred)).round(3))
print('RSquared:', r2_score(y_test, model4_y_pred).round(3))
```

```
MAE: 7.869
MSE: 251.984
RMSE: 15.874
RSquared: 0.888
```

In [130…

```
import pandas as pd
#find out importance
rfr_fea = pd.DataFrame(rfr.feature_importances_)
rfr_fea["Feature"] = list(X_train)
rfr_fea.sort_values(by=0, ascending=False).head()
```

Out[130…

	0	Feature
19	0.168382	reserved_room_type
5	0.167688	arrival_date_week_number
28	0.121508	reservation_status_date
0	0.080094	hotel
4	0.068062	arrival_date_month

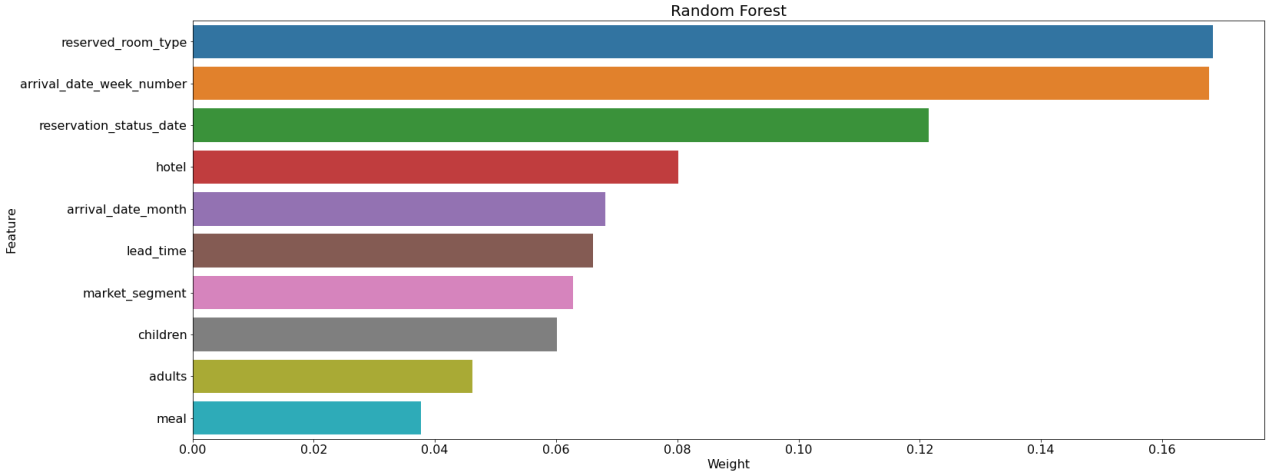
In [131…

```
#plot importance
plt.figure(figsize=(25,10))
b = sns.barplot(0,"Feature",data = rfr_fea.sort_values(by=0, ascending=False)[0:10],ori
b.set_xlabel("Weight", fontsize=16)
b = b.set_title("Random Forest", fontsize=20)
plt.ylabel('Feature', fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
```

Out[131…

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'reserved_room_type'),
```

```
Text(0, 1, 'arrival_date_week_number'),
Text(0, 2, 'reservation_status_date'),
Text(0, 3, 'hotel'),
Text(0, 4, 'arrival_date_month'),
Text(0, 5, 'lead_time'),
Text(0, 6, 'market_segment'),
Text(0, 7, 'children'),
Text(0, 8, 'adults'),
Text(0, 9, 'meal')])
```



Step 7: Validation

In [132…

```
#get the cross validation score
rfr_score = cross_val_score(rfr, X_train, y_train, cv=10)
print(rfr_score)
```

```
[0.88542135 0.88777241 0.89058961 0.8699835  0.88785317 0.89410397
 0.89020442 0.89592782 0.89545241 0.88722248]
```

In [144…

```
# get the mean score
rfr_score.mean()
```

Out[144… 0.8884531148051517

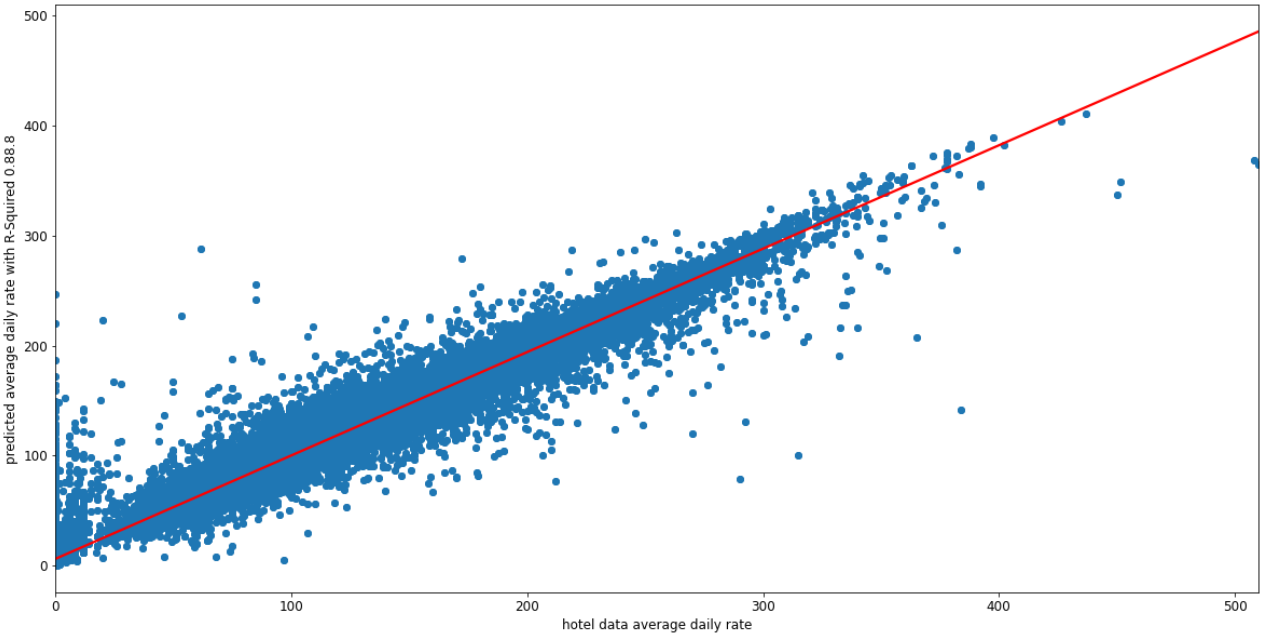
In [146…

```
#predict Y given by X (all variables except company and agent)
m=rfr.predict(X)

#extract real data column adr
n=hotel.adr

plt.scatter(n, m)
mpl.rcParams['figure.figsize'] = (20,10)
sns.regplot(n, m, ci=None, line_kws = {'color': 'red'})
#ax.lines[0].set_linestyle("--")
plt.xlabel('hotel data average daily rate')
plt.ylabel('predicted average daily rate with R-Squired 0.88.8')
```

Out[146… Text(0, 0.5, 'predicted average daily rate with R-Squired 0.88.8')



Summary: Getting a good forecast and understanding of the average daily rate is crucial for hotel industry. We sucessfully build a model to predict average daily rate with average accuracy rate of 88.8% by using Random Forest regression model. This results will help the Marketing and Sales department getting an early indication of how the sales for budgeting purpose.

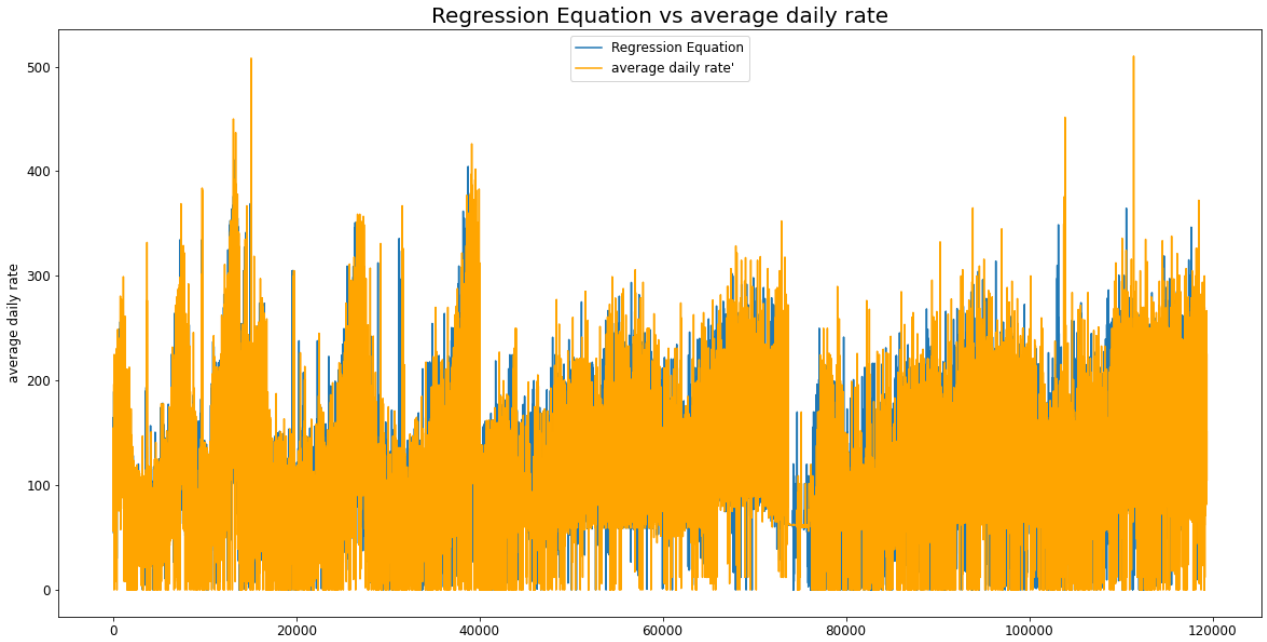
In [141…

```
#predict Y given by X
a=rfr.predict(X)
print(max(a))

#extract real data column adr
b=hotel['adr']
print(max(b))

plt.plot(a, label = 'Regression Equation')
lns2=plt.plot(b, 'orange', label="average daily rate'")
plt.xlabel('')
# Set the y axis label of the current axis.
plt.ylabel('average daily rate')
# Set a title of the current axes.
plt.title('Regression Equation vs average daily rate', fontsize=20)
# show a legend on the plot
plt.legend(loc="upper center")
mpl.rcParams['figure.figsize'] = (20,10)
# Display a figure.
plt.show()
```

```
410.72049999999996
510.0
```



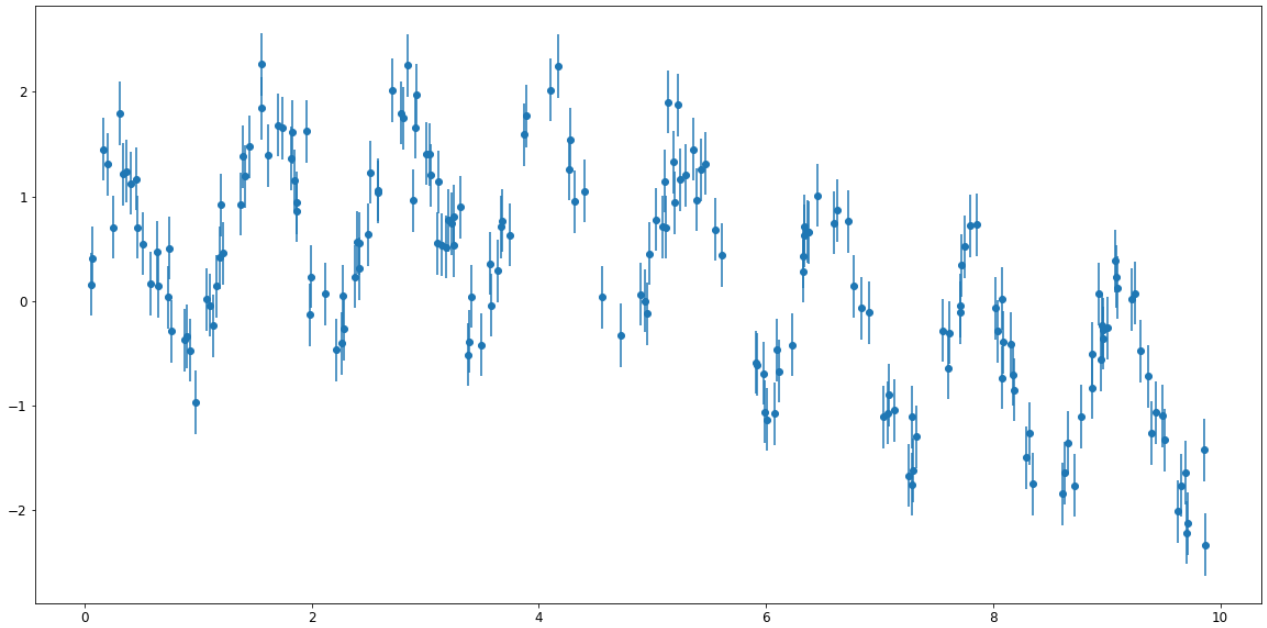
In [139…

```
rng = np.random.RandomState(42)
x = 10 * rng.rand(200)

def model(x, sigma=0.3):
    fast_oscillation = np.sin(5 * x)
    slow_oscillation = np.sin(0.5 * x)
    noise = sigma * rng.randn(len(x))

    return slow_oscillation + fast_oscillation + noise

y = model(x)
plt.errorbar(x, y, 0.3, fmt='o');
```

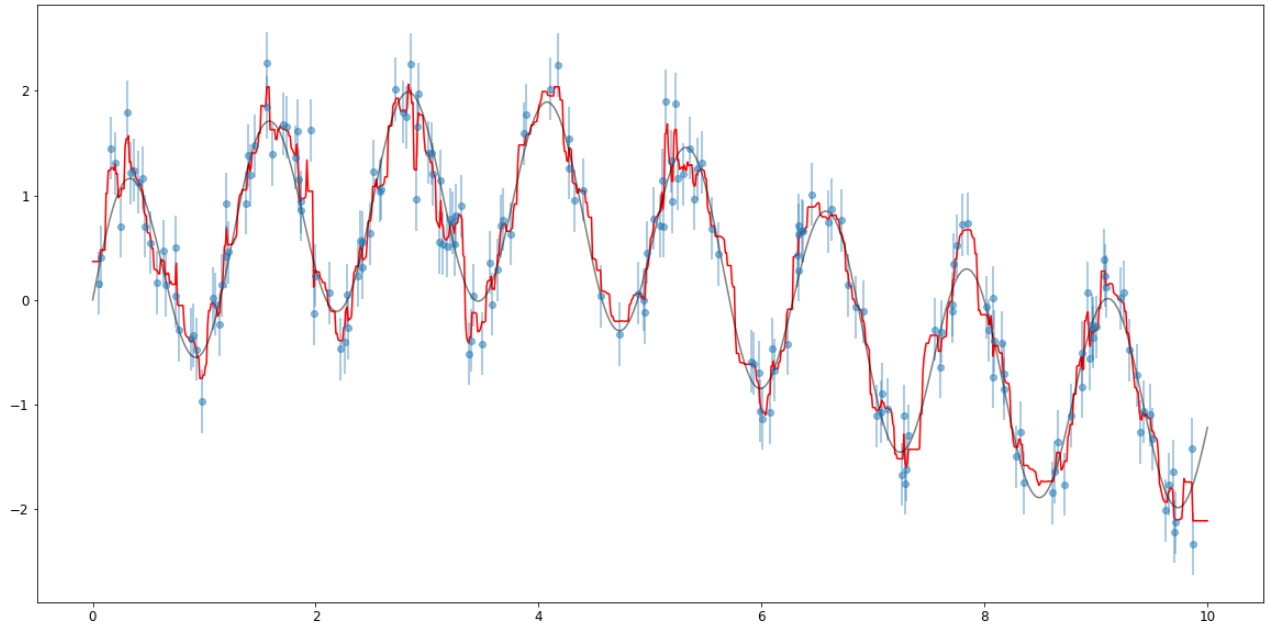


In [140…

```
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor(200)
forest.fit(x[:, None], y)
```

```
xfit = np.linspace(0, 10, 1000)
yfit = forest.predict(xfit[:, None])
ytrue = model(xfit, sigma=0)

plt.errorbar(x, y, 0.3, fmt='o', alpha=0.5)
plt.plot(xfit, yfit, '-r');
plt.plot(xfit, ytrue, '-k', alpha=0.5);
```



In [ ]: