PROBLEM STATEMENT:

Have you ever wondered when the best time of year to book a hotel room is? Or the optimal length of stay in order to get the best daily rate? What if you wanted to predict whether or not a hotel was likely to receive a disproportionately high number of special requests? This hotel booking dataset can help you explore those questions! This data set contains booking information for a city hotel and a resort hotel, and includes information such as when the booking was made, length of stay, the number of adults, children, and/or babies, and the number of available parking spaces, among other things. All personally identifying information has been removed from the data. Explore and analyze the data to discover important factors that govern the bookings.

EDA on Hotel Booking

Importing Libraries

In [1]:
```
import pandas as pd  #pandas for handling analysis of the dataset
import matplotlib.pyplot as plt  #matplotlib for visulazing the analysis
import numpy as np   #numpy to handle numerical array based calculation
import seaborn as sns
%matplotlib inline
```

Reading csv file

In [3]:
```
df = pd.read_csv("Hotel Bookings.csv")
```

Exploring The Dataset

First Five Columns

In [4]:
```
df.head().T
```

Out[4]:

	0	1	2	3	4
hotel	Resort Hotel	Resort Hotel	Resort Hotel	Resort Hotel	Resort Hotel
is_canceled	0	0	0	0	0
lead_time	342	737	7	13	14
arrival_date_year	2015	2015	2015	2015	2015
arrival_date_month	July	July	July	July	July
arrival_date_week_number	27	27	27	27	27
arrival_date_day_of_month	1	1	1	1	1
stays_in_weekend_nights	0	0	0	0	0
stays_in_week_nights	0	0	1	1	2
adults	2	2	1	1	2
children	0	0	0	0	0
babies	0	0	0	0	0
meal	BB	BB	BB	BB	BB
country	PRT	PRT	GBR	GBR	GBR
market_segment	Direct	Direct	Direct	Corporate	Online TA
distribution_channel	Direct	Direct	Direct	Corporate	TA/TO
is_repeated_guest	0	0	0	0	0
previous_cancellations	0	0	0	0	0
previous_bookings_not_canceled	0	0	0	0	0
reserved_room_type	C	C	A	A	A
assigned_room_type	C	C	C	A	A
booking_changes	3	4	0	0	0
deposit_type	No Deposit	No Deposit	No Deposit	No Deposit	No Deposit
agent	NaN	NaN	NaN	304	240
company	NaN	NaN	NaN	NaN	NaN
days_in_waiting_list	0	0	0	0	0
customer_type	Transient	Transient	Transient	Transient	Transient
adr	0	0	75	75	98
required_car_parking_spaces	0	0	0	0	0
total_of_special_requests	0	0	0	0	1
reservation_status	Check-Out	Check-Out	Check-Out	Check-Out	Check-Out
reservation_status_date	2015-07-01	2015-07-01	2015-07-02	2015-07-02	2015-07-03

Looking into shape of data to find out number of rows and columns

In [5]:
```
df.shape
```

Out[5]:
```
(119390, 32)
```

Information about Dataset

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 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
```

Description about dataset

In [7]:
```
df.describe()
```

Out[7]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	0.927599
std	0.482918	106.863097	0.707476	13.605138	8.780829	0.998613
min	0.000000	0.000000	2015.000000	1.000000	1.000000	0.000000
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	0.000000
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	1.000000
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	2.000000
max	1.000000	737.000000	2017.000000	53.000000	31.000000	19.000000

Conclusion: There are few columns with many null values. So we are dropping those col

Dropping The Columns ["Company","agent"]

In [8]:
```
df.drop(columns=["company","agent"],inplace=True)
```

Verfying it

In [9]:
```
df.columns
```

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

Childern column has few null values

In [10]:
```
df.children.isna().sum()
```

Out[10]:
```
4
```

We have four 4 values. So we can fill that using mode

In [11]:
```
df.children.fillna(df.children.mode,inplace=True)
```

In [12]:
```
df.children.isna().sum()
```

Out[12]:
```
0
```

Country column has few null values

In [13]:
```
df.country.isna().sum()
```

Out[13]:
```
488
```

We have 488 null values. As it is categorical data type We can fill using "Mode"

In [14]:
```
df.country.fillna(df.country.mode,inplace=True)
```

In [15]:
```
df.country.isna().sum()
```

Out[15]:
```
0
```

Now our Dataset is cleaned

Let's find insights from the Dataset

Types of Hotels

In [60]:
```
df['hotel'].value_counts()
```

Out[60]:
```
City Hotel      79330
Resort Hotel    40060
Name: hotel, dtype: int64
```

In [61]:
```
df['hotel'].value_counts().plot(kind='pie')
```

Out[61]:
```
<AxesSubplot:ylabel='hotel'>
```

Majority of the booked Hotel is City Hotel than Resort Hotel

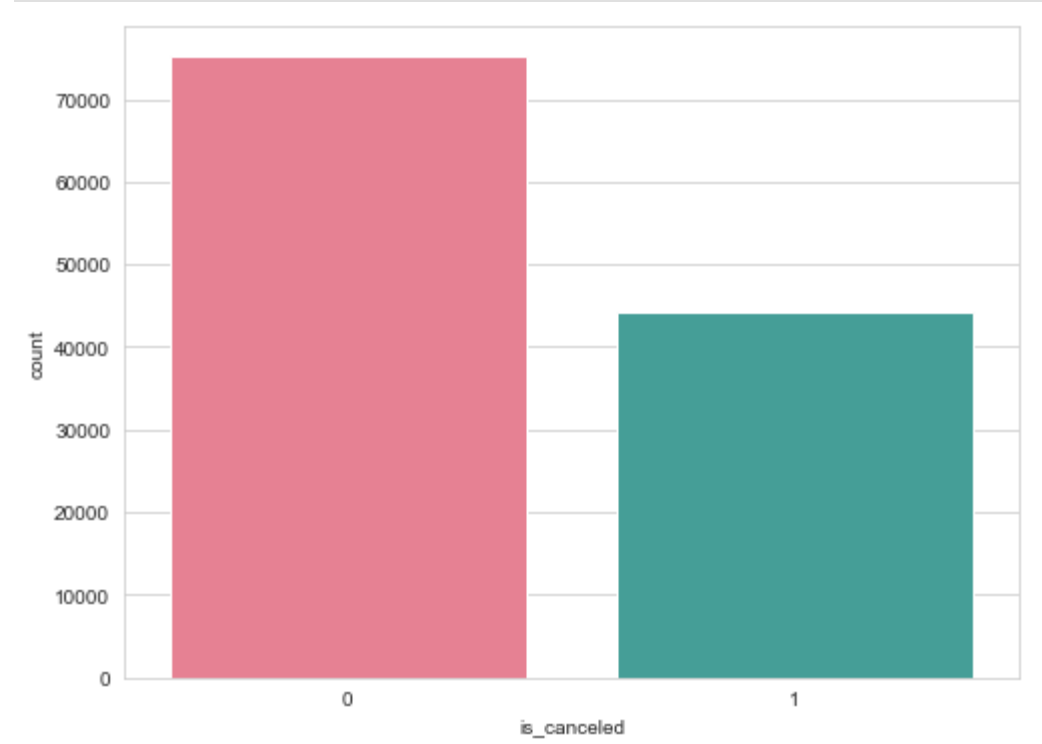
HOW MANY BOOKING WERE CANCELLED?

In [16]:
```
print('Total Bookings cancelled')
print(df.is_canceled.value_counts())
print('Cancelation percentage')
print(df.is_canceled.value_counts(normalize=True)*100)
```

```
Total Bookings cancelled
0    75166
1    44224
Name: is_canceled, dtype: int64
Cancelation percentage
0    62.958372
1    37.041628
Name: is_canceled, dtype: float64
```

Conclusion: During the year we have 37% of cancelations.

In [18]:
```
plt.figure(figsize=(8,6))
sns.set_style('whitegrid')
sns.countplot(x=df['is_canceled'],palette='husl')
plt.show()
# 0 is canceled and 1 is not canceled
```



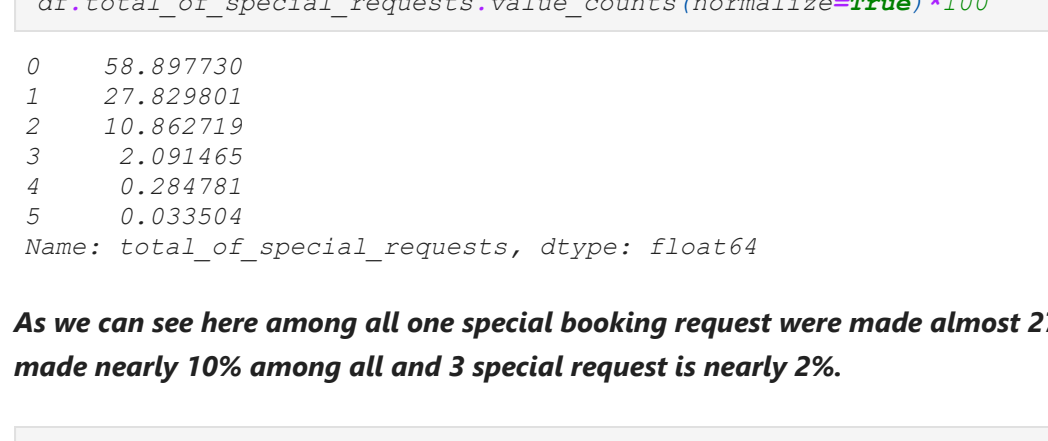
SPECIAL REQUESTS

In [19]:
```
df.total_of_special_requests.value_counts(normalize=True)*100
```

Out[19]:
```
0    58.897730
1    27.829801
2    10.862719
3     2.091465
4     0.284781
5     0.033504
Name: total_of_special_requests, dtype: float64
```

As we can see here among all one special booking request were made almost 27% of total bookings.Two special request were made nearly 10% among all and 3 special request is nearly 2%.

In [21]:
```
plt.figure(figsize=(8,6))
sns.set_style('whitegrid')
sns.countplot(x='total_of_special_requests',data=df)
plt.show()
```

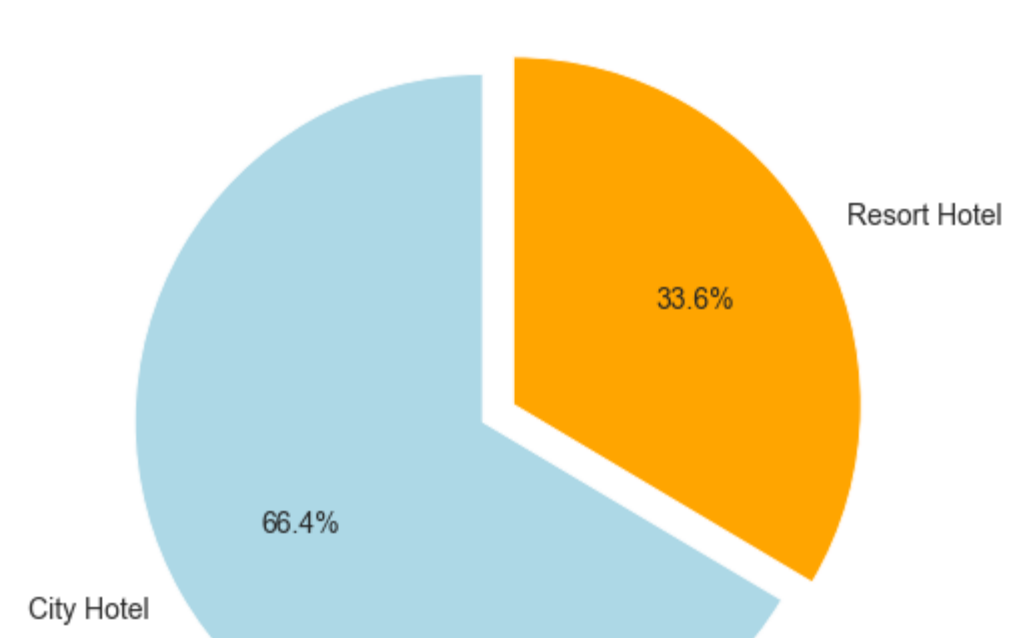


BOOKING RATIO BETWEEN RESORT HOTEL AND CITY HOTEL

In [22]:
```
plt.rcParams['figure.figsize'] = 8,8
labels = df['hotel'].value_counts().index.tolist()
sizes = df['hotel'].value_counts().tolist()

explode = (0, 0.1)
colors = ['lightblue','ORANGE']
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',startangle=90, textprops={'font
```

Out[22]:
```
([<matplotlib.patches.Wedge at 0x2666f06ba60>,
  <matplotlib.patches.Wedge at 0x2666f079190>],
 [Text(-0.956416133381298, -0.5433858480011854, 'City Hotel'),
  Text(1.0433630545977801, 0.592784561455838, 'Resort Hotel')],
 [Text(-0.5216815272988897, -0.29639228072791923, '66.4%'),
  Text(0.6086284485153717, 0.3457909941825721, '33.6%')])
```



What do we see here?
It seems that a huge proportion of hotels was city hotel. Resort hotel tend to be on the expensive side and most people will just stick with city hotel. Also, resort hotels tend to be appropriate for larger group of people.

Let's have an overview of the number of people who booked the hotel.

In [23]:
```
# Looking into adults.
# Using groupby to group according to hotel types only.
df['adults'].groupby(df['hotel']).describe()
```

Out[23]:

hotel	count	mean	std	min	25%	50%	75%	max
City Hotel	79330.0	1.850977	0.509292	0.0	2.0	2.0	2.0	4.0
Resort Hotel	40060.0	1.867149	0.697285	0.0	2.0	2.0	2.0	55.0

In [24]:
```
# Looking into children.
# Using groupby to group according to hotel types only.
df['children'].groupby(df['hotel']).describe()
```

Out[24]:

hotel	count	unique	top	freq
City Hotel	79330.0	5.0	0.0	74220.0
Resort Hotel	40060.0	5.0	0.0	36576.0

It seems that mean values for adults and children are higher. This means that resort hotels are better choice for large families.

FROM WHICH COUNTRY MOST GUEST COME?

In [25]:
```
df.country.value_counts(normalize=True)*100
```

Out[25]:
```
PRT    40.698551
GBR    10.159142
FRA     8.723511
ESP     7.176480
DEU     6.103526
          ...
SDN     0.000838
MRT     0.000838
NAM     0.000838
ASM     0.000838
HND     0.000838
Name: country, Length: 178, dtype: float64
```

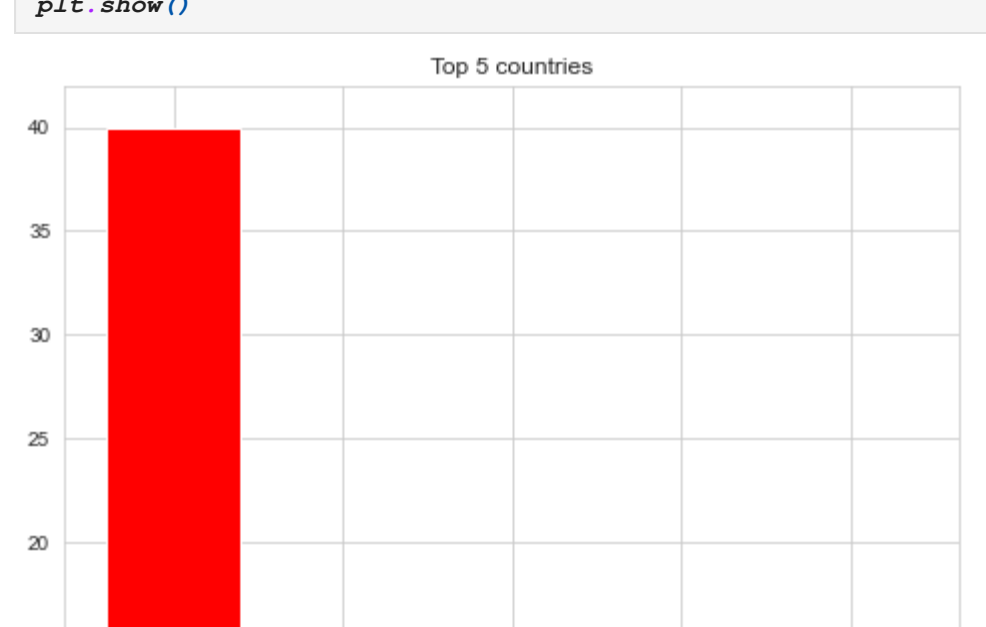
As we can see, Portugal tops the list with 40.69% of the guest, followed by

---

Great Britain with 10.15%, France, Spain and Germany.

Majority of the guests are from European Countries.

In [29]:
```
x=['PRT','GBR','FRA','ESP','DEU']
h=[40,10,0.8,0.7,0.6]
plt.bar(x,h,color=["red","orange","blue","blue","blue"])
plt.xlabel('Country')
plt.title("Top 5 countries")
plt.show()
```

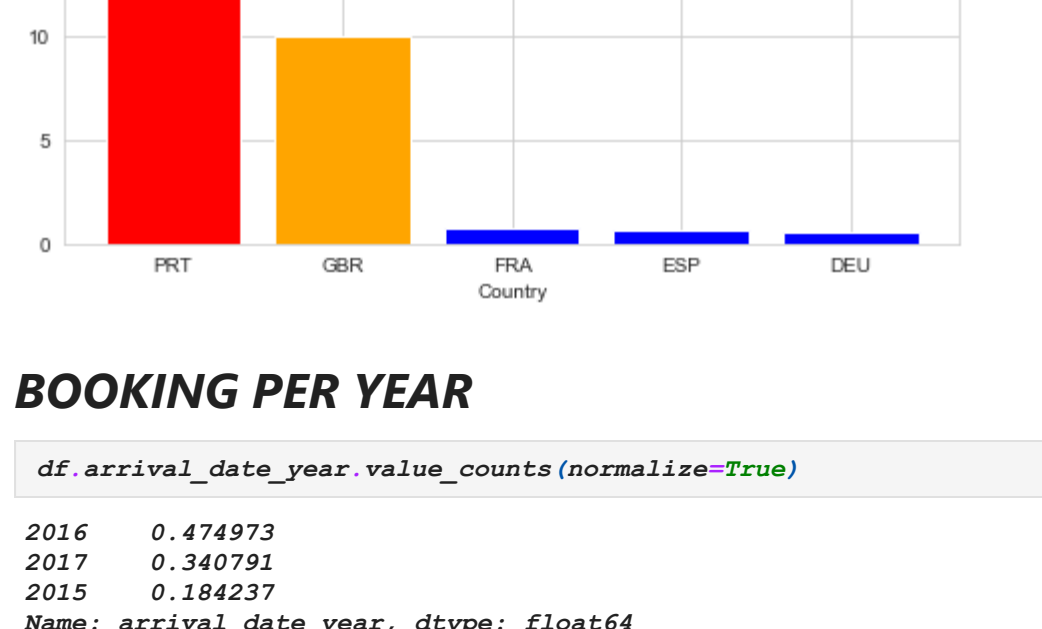


BOOKING PER YEAR

In [30]:
```
df.arrival_date_year.value_counts(normalize=True)
```

Out[30]:
```
2016    0.474973
2017    0.340791
2015    0.184237
Name: arrival_date_year, dtype: float64
```

In [32]:
```
plt.figure(figsize=(8,8))
sns.countplot(x=df['arrival_date_year'], palette='husl')
plt.show()
```



Conclusion:</i>
1.There has been many arrivals in the year 2016 than the remaning years
2. We can also say that there has been increase in the arrivals as years passes.

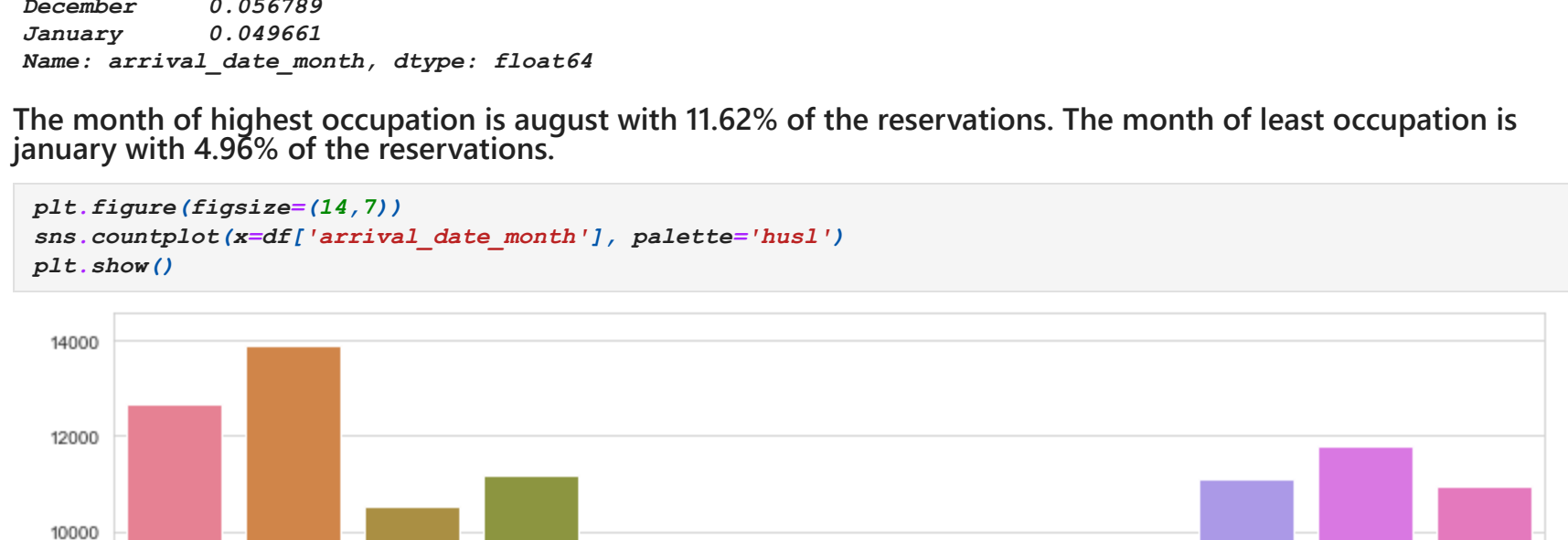
BUSIEST MONTH FOR HOTELS

In [34]:
```
df.arrival_date_month.value_counts(normalize=True)
```

Out[34]:
```
August       0.116233
July         0.106047
May          0.098760
October      0.093475
April        0.092880
June         0.091624
September    0.088014
March        0.082034
February     0.067577
November     0.056906
December     0.056789
January      0.049661
Name: arrival_date_month, dtype: float64
```

The month of highest occupation is august with 11.62% of the reservations. The month of least occupation is january with 4.96% of the reservations.

In [37]:
```
plt.figure(figsize=(14,7))
sns.countplot(x=df['arrival_date_month'], palette='husl')
plt.show()
```



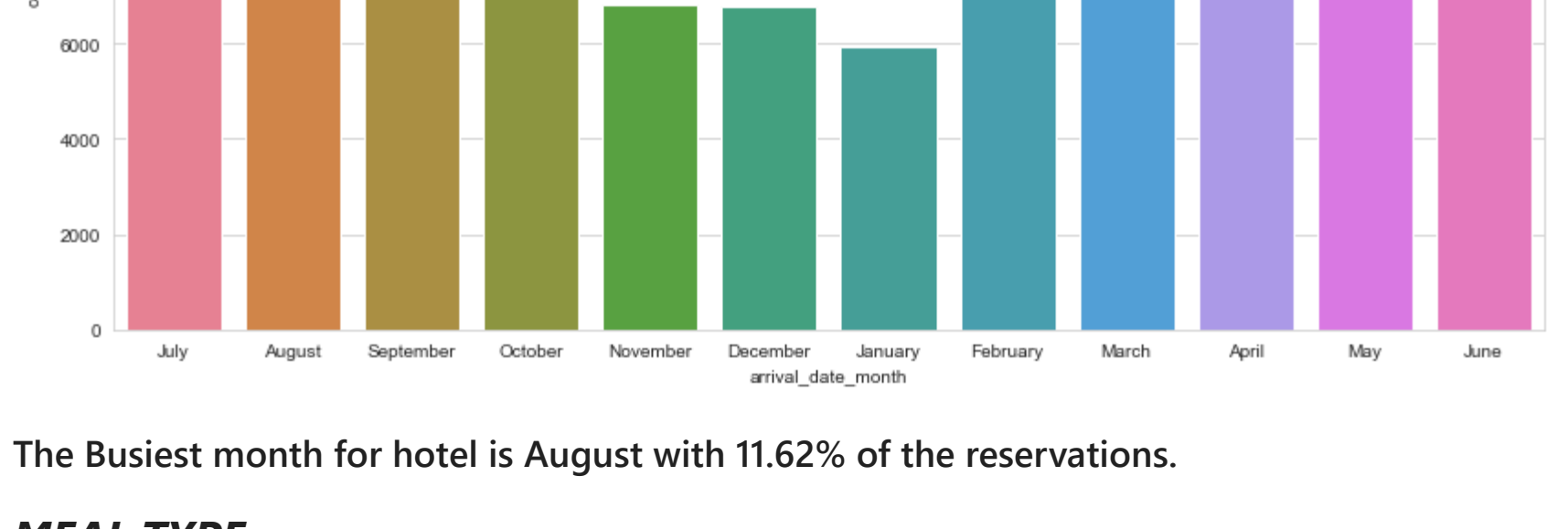
The Busiest month for hotel is August with 11.62% of the reservations.

MEAL TYPE

In [41]:
```
df.meal.value_counts(normalize=True)
```

Out[41]:
```
BB           0.773180
HB           0.121141
SC           0.089203
Undefined    0.009791
FB           0.006684
Name: meal, dtype: float64
```

In [43]:
```
plt.figure(figsize=(14,7))
sns.countplot(x=df['meal'], palette='husl')
plt.show()
```

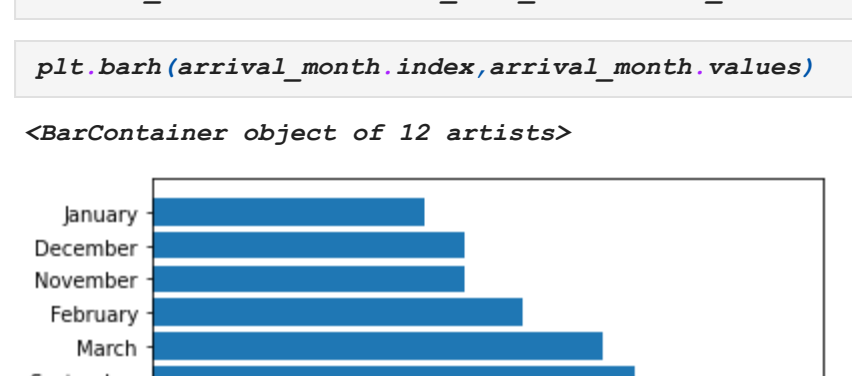


No of travellers in Various Months

In [21]:
```
arrival_month = df.arrival_date_month.value_counts()
```

In [27]:
```
plt.barh(arrival_month.index,arrival_month.values)
```

Out[27]:
```
<BarContainer object of 12 artists>
```



So, We can say the peak months are : August, July, May(Summer Time)

Accomodation room types

In [69]:
```
df.reserved_room_type.value_counts()
```

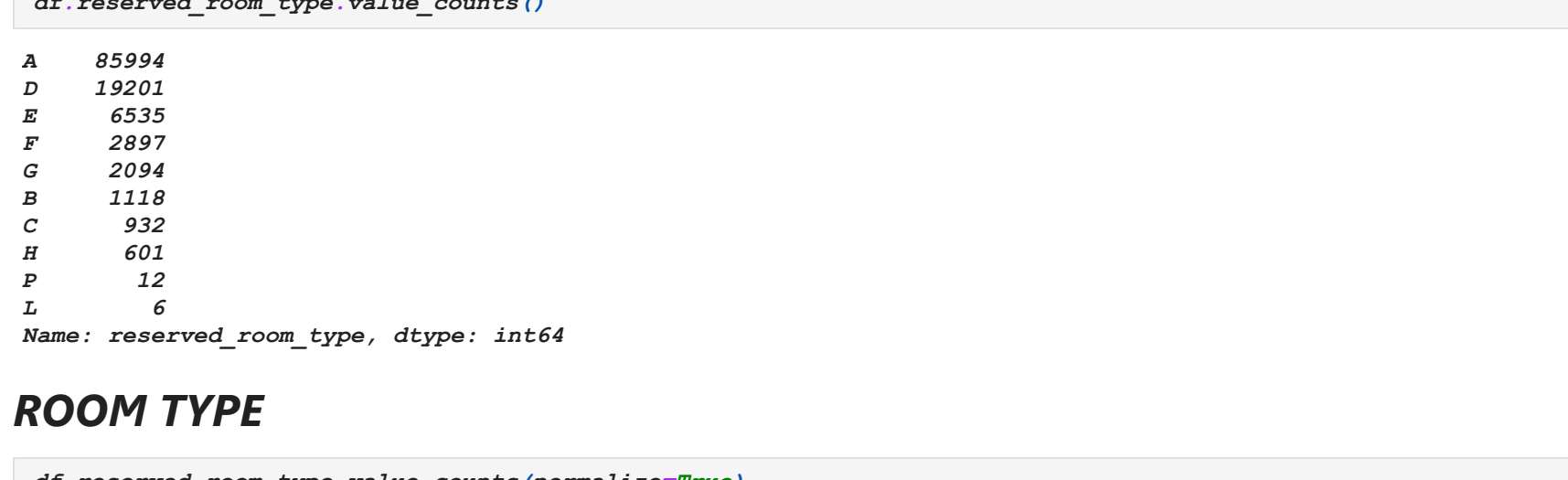
Out[69]:
```
A    85994
D    19201
E     6535
F     2897
G     2094
B     1118
C      932
H      601
P       12
L        6
Name: reserved_room_type, dtype: int64
```

ROOM TYPE

In [46]:
```
df.reserved_room_type.value_counts(normalize=True)
```

Out[46]:
```
A    0.720278
D    0.160826
E    0.054737
F    0.024265
G    0.017539
B    0.009364
C    0.007806
H    0.005034
P    0.000101
L    0.000050
Name: reserved_room_type, dtype: float64
```

In [47]:
```
plt.figure(figsize=(14,7))
sns.countplot(x=df['reserved_room_type'], palette='husl')
plt.show()
```



REPEATED GUEST

In [48]:
```
df.is_repeated_guest.value_counts(normalize=True)
```

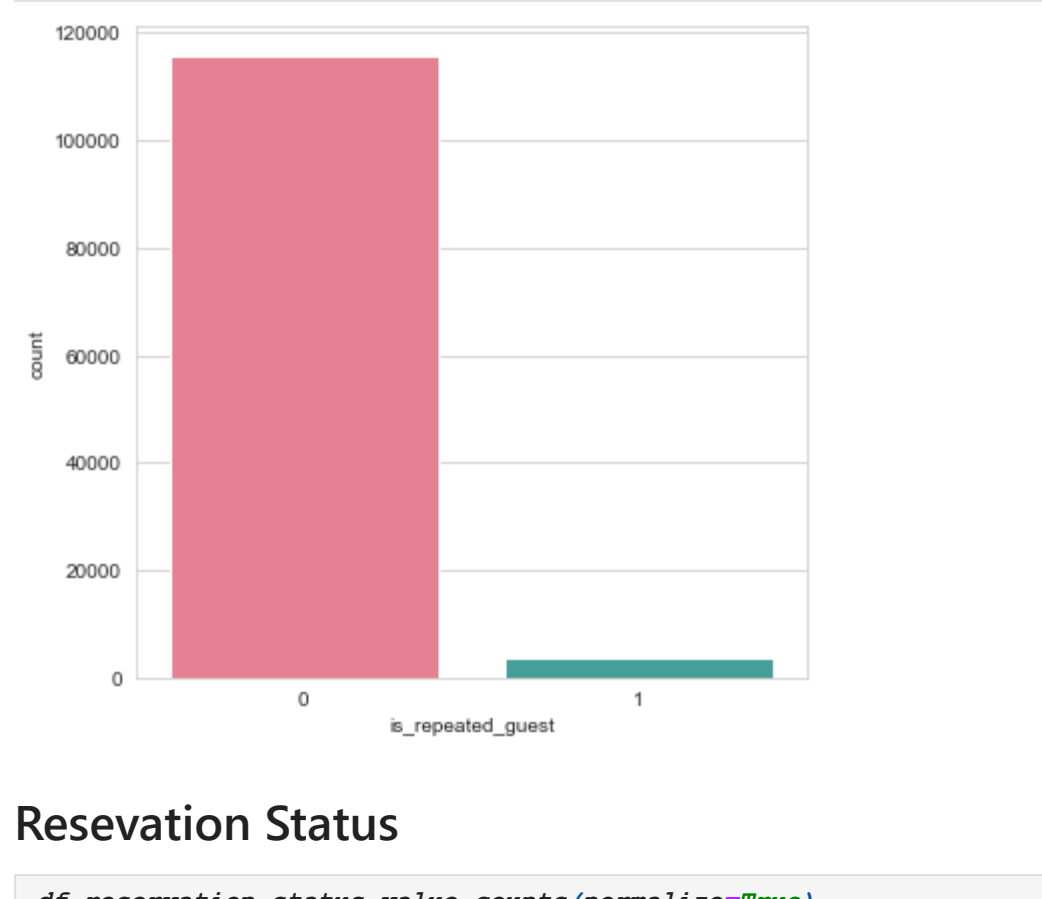
Out[48]:
```
0    0.968088
1    0.031912
Name: is_repeated_guest, dtype: float64
```

In [49]:
```
plt.figure(figsize=(6,6))
sns.countplot(x=df['is_repeated_guest'], palette='husl')
plt.show()
```

Resevation Status

In [51]:
```
df.reservation_status.value_counts(normalize=True)
sns.countplot(x=df['reservation_status'],palette='husl')
```

Out[51]:
```
<AxesSubplot:xlabel='reservation_status', ylabel='count'>
```



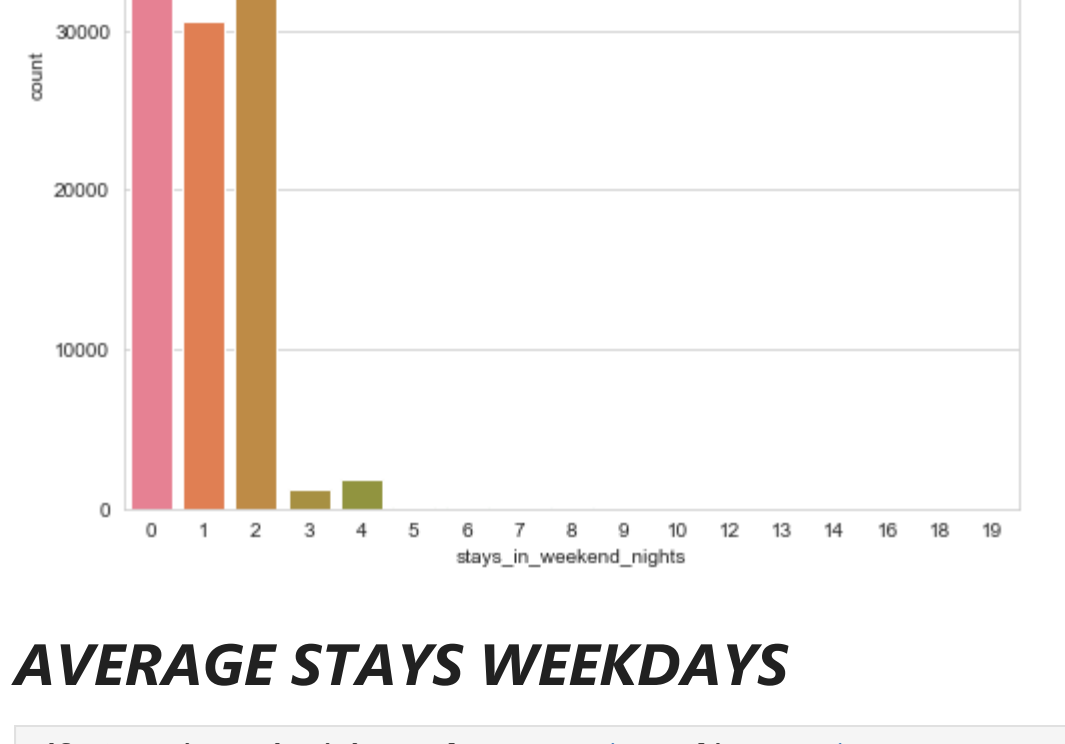
AVERAGE STAY WEEKENDS VS WEEKDAYS

PEOPLE GENERALLY PREFER LONG STAYS ON WEEKDAYS RATHER THAN WEEKENDS

AVERAGE STAYS WEEKENDS

In [54]:
```
df.stays_in_weekend_nights.value_counts(normalize=True)
sns.countplot(x=df['stays_in_weekend_nights'],palette='husl')
print(df.stays_in_weekend_nights.mean()/2)
```

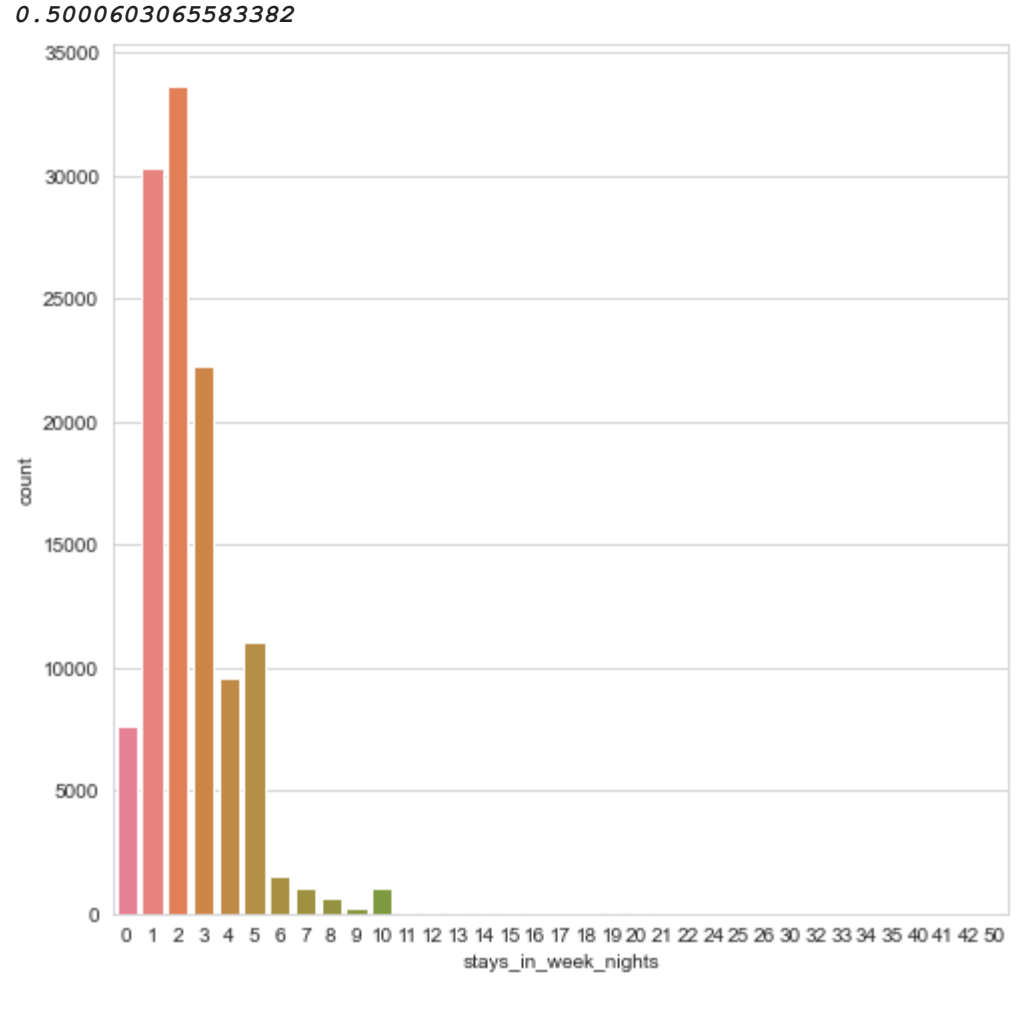
```
0.4637993131753078
```



AVERAGE STAYS WEEKDAYS

In [56]:
```
df.stays_in_week_nights.value_counts(normalize=True)
sns.countplot(x=df['stays_in_week_nights'],palette='husl')
print(df.stays_in_week_nights.mean()/5)
```

```
0.5000603065583382
```



Checking the travellers without children

```
In [75]: df.children[df.children!=0]
```

```
Out[75]: 13        1
         45        2
         55        2
         65        2
         87        1
                  ..
         119270    1
         119287    1
         119293    2
         119318    1
         119319    2
         Name: children, Length: 8594, dtype: object
```

Checking the travellers without babies

```
In [71]: df.babies.value_counts()
```

```
Out[71]: 0     118473
         1        900
         2         15
         10         1
         9          1
         Name: babies, dtype: int64
```

Checking the adult travellers count

```
In [90]: df.adults.value_counts()
```

```
Out[90]: 2     89680
         1     23027
         3      6202
         0       403
         4        62
         26        5
         27        2
         20        2
         5         2
         55        1
         50        1
         40        1
         10        1
         6         1
         Name: adults, dtype: int64
```

```
In [85]: single = df[(df['children']==0) & (df['babies']==0)]
```

```
In [89]: actual_single = len(df[df['adults']==1])
```

```
In [91]: actual_single
```

```
Out[91]: 23027
```

There are around 23,027 rooms booked as single room

```
In [94]: actual_couple  = len(df[df['adults']==2])
```

```
In [95]: actual_couple
```

```
Out[95]: 89680
```

There are around 89,680 rooms booked as couples room

```
In [96]: actual_family = len(df) - actual_couple - actual_single
```

```
In [97]: actual_family
```

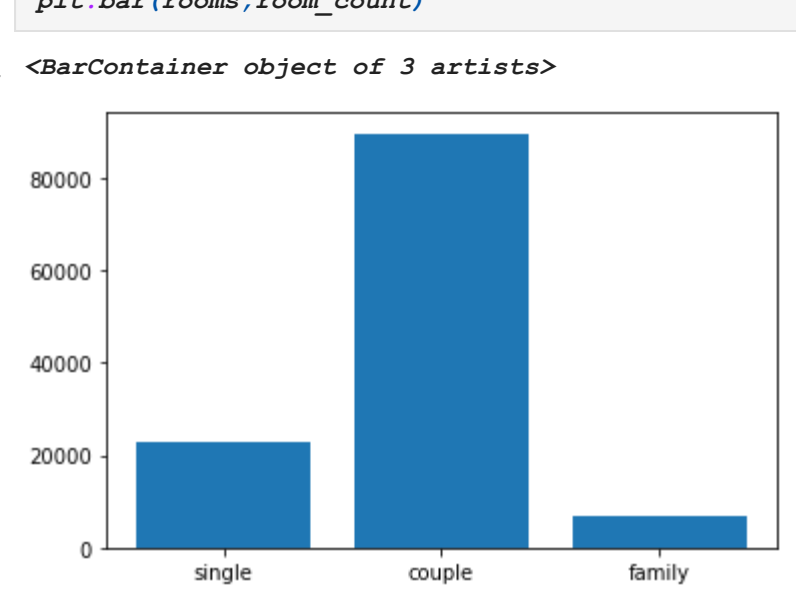
```
Out[97]: 6683
```

There around 6683 rooms booked as family room

Room Types

```
In [100…: rooms = ['single','couple','family']
          room_count = [actual_single,actual_couple,actual_family]
          plt.bar(rooms,room_count)
```

```
Out[100…: <BarContainer object of 3 artists>
```



Clearly Majority of the rooms booked are couple rooms

Online Booking vs Offline Booking(Travel Agency,Direct)

```
In [106…: modes = df.market_segment.value_counts()
```

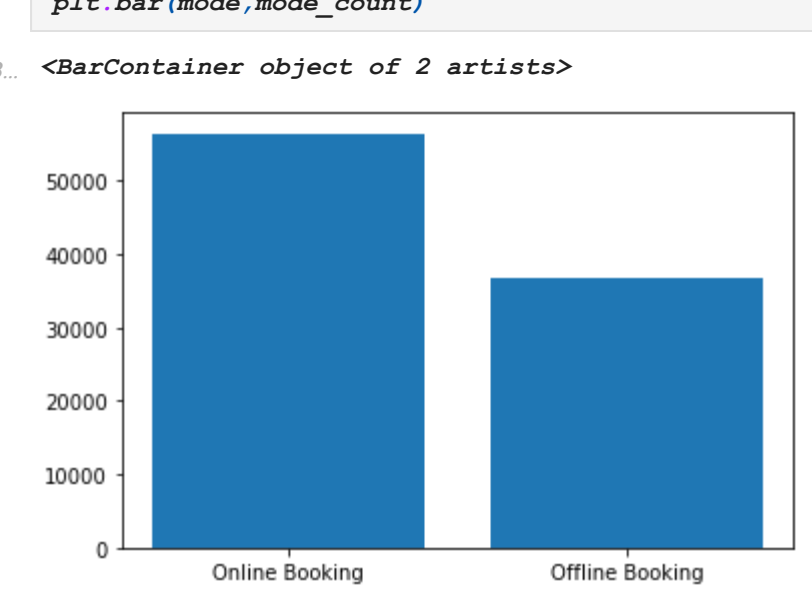
```
In [112…: modes
```

```
Out[112…: Online TA        56477
          Offline TA/TO    24219
          Groups           19811
          Direct           12606
          Corporate         5295
          Complementary      743
          Aviation           237
          Undefined            2
          Name: market_segment, dtype: int64
```

```
In [104…: onlineTa = df[df.market_segment=='Online TA']
```

```
In [113…: mode = ["Online Booking", "Offline Booking"]
          mode_count = [modes["Online TA"],modes["Offline TA/TO"]+modes["Direct"]]
          plt.bar(mode,mode_count)
```

```
Out[113…: <BarContainer object of 2 artists>
```



So, Majority of the customers book through online rather than offline booking

```
In [150…: count_cancel=len(df[(df["market_segment"]=="Online TA") & df["is_canceled"]!=0])
```

```
In [152…: total_online = len(df[(df["market_segment"]=="Online TA")])
```

```
In [156…: print("Possibility of Not Cancelling is:",round(100-(count_cancel*100/total_online),2))
          Possibility of Not Cancelling is: 63.28
```

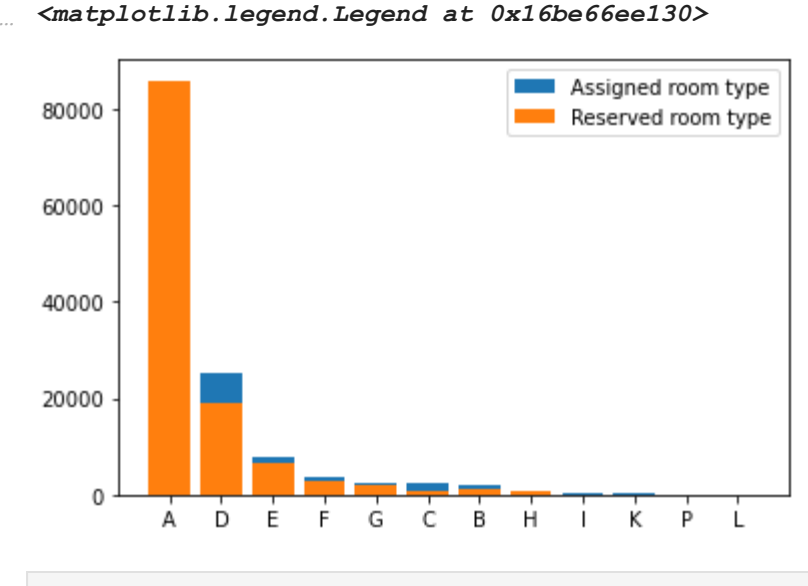
Satisfactory of customers

```
In [146…: reserv = df.reserved_room_type.value_counts()
```

```
In [144…: assign = df.assigned_room_type.value_counts()
```

```
In [148…: plt.bar(assign.index,assign.values,label="Assigned room type")
          plt.bar(reserv.index,reserv.values,label="Reserved room type")
          plt.legend()
```

```
Out[148…: <matplotlib.legend.Legend at 0x16be66ee130>
```



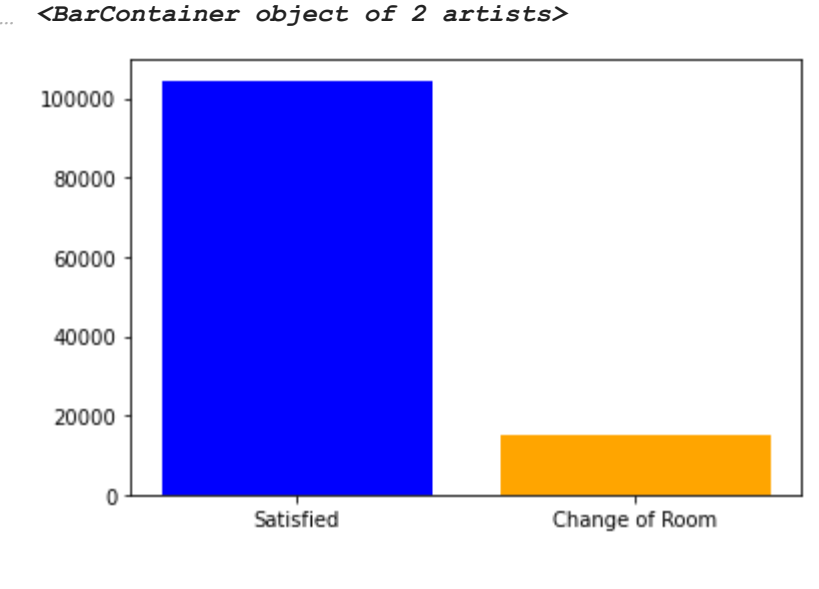

```
In [140…: def fun(x,y):
              if x==y:
                  return True
              else:
                  return False
```

```
In [141…: df["prefer"] = df.apply(lambda x: fun(x.reserved_room_type,x.assigned_room_type),axis=1)
```

```
In [142…: sat_count = df["prefer"].value_counts()
```

```
In [143…: model = ["Satisfied","Change of Room"]
          data_sat = [sat_count[True],sat_count[False]]
          plt.bar(model,data_sat,color = ['blue','orange'])
```

```
Out[143…: <BarContainer object of 2 artists>
```



Inference: Majority of the customers has got the room type they have booked

---

Based on the exploration of Data we can say that:

1. During Summer Season the hotels are expected to get more no of bookings than any other seasons. So, if You want to enjoy privacy You can book in remaining seasons.

2. Majority of the Bookings happen through online as it is very easy and efficient way.

3. We can clearly see the average stay is between 1-3 days we must try to extend the duration as there is high chance to get attrcative deals from the hotels.