Chapter 1
Introduction

I landed a great job with the Ritz-Jager Hotel operator as a data scientist. This hotel operator wants to improve their business efficiency by utilizing their historical data and they want to find out what happened in their previous bookings, knowing their customer better, and optimizing the promo timing. Your team of engineer have to **analyze the data** that they have based on the pre-defined questions that your CEO gave.

Questions:
1. Where do the guests come from?
2. How much do guests pay for a room per night?
3. How does the price per night vary over the year?
4. Which are the busiest months?
5. How long do people stay at the hotels?
6. Bookings by market segment
7. How many bookings were cancelled?
8. Which month has the highest number of cancellations?

Chapter 2
Progress Report

Day/Date	**Task**	**Level** (easy/medium/hard)	**Comments**
02/05/2020	Implement explanatory data analysis in Ritz Jager dataset	Easy	

Chapter 3
Task Report

The first step is to import the required libraries, which are Pandas, Numpy, MatplotLib, Seaborn and Plotly. Next, import the Ritz Jager dataset using pandas. The data has 119390 rows and 32 columns.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly.express as px

In [88]: df=pd.read_csv("E:/Ritz_Jager_Data.csv")

In [89]: df.shape
Out[89]: (119390, 32)

In [90]: df.head()
Out[90]:
```

	hotel_type	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	sta
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	

5 rows × 32 columns

The next step is to preprocessing data. First check the missing value for each variable. There are 4 variables that have missing values, namely children, country origin, agent and company variables. Here are the details:

```
In [6]: df.isnull().sum()
```

```
Out[6]: hotel_type                             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_type                              0
        country_origin                       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
```

The four variables are imputed to eliminate the missing value. Missing values for the children variable are imputed with a median value of 0, which means there are no children's guests on the order. Missing value in the country origin variable is imputed with the value "unknown" because we do not know the country of origin of the customer. Missing values for the agent variable are imputed with a value of 0, where we assume that if the agent ID is not listed or given, ordering is most likely done without an agent. Furthermore, the missing value on the company variable is also imputed with a value of 0, which we assume if the company ID is not listed or given, most likely the order is made in private.

Furthermore, in the meal type variable there is an observation with the value "Underfined" where the value is the same as the "SC" class. Therefore, we replace the value "Underfined" with the value "SC". Furthermore, there are some observations in which the number of guests both adults, children and babies all have a value of 0, which means there are no guests in the booking. Therefore, we delete or drop the observation with the number of guests equal to 0. Furthermore, the reservation status date variable is initially the object data type which is changed to the date time data type.

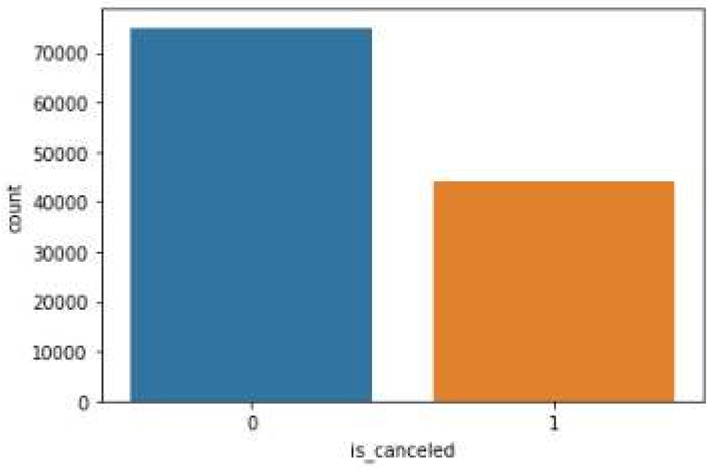
```
In [105]: df["children"].median()
Out[105]: 0.0
```

```
In [100]: df = df.fillna({"children:": 0.0,"country_origin": "Unknown", "agent": 0,
                          "company": 0})
          df["meal_type"]=df["meal_type"].replace("Undefined", "SC")
          zero= list(df[df["adults"]
                        + df["children"]
                        + df["babies"]==0].index)
          df=df.drop(df.index[zero])
```

```
In [92]: df["reservation_status_date"]=pd.to_datetime(df["reservation_status_date"],format="%d/%m/%Y")
```

The next step, we check the comparison between canceled and non-canceled orders. There are 75001 orders that were not canceled and 44199 orders that were canceled. The total number of orders canceled is more than half of the un-canceled orders. Therefore to simplify the analysis process, we divide the data into 6 parts, namely total booking data, booking data not canceled, total hotel resort booking data, hotel resort booking data not canceled, city hotel total booking data and city hotel booking data not canceled.

```
In [98]: sns.countplot(df["is_canceled"])
```




```
In [99]: df_total=df
         df_noncancel=df[df["is_canceled"] == 0]
         resort_total= df[df["hotel_type"] == "Resort Hotel"]
         city_total= df[df["hotel_type"] == "City Hotel"]
         resort_noncancel= df[(df["hotel_type"] == "Resort Hotel") &
                              (df["is_canceled"] == 0)]
         city_noncancel= df[(df["hotel_type"] == "City Hotel") &
                            (df["is_canceled"] == 0)]
```

1. Where do the guests come from?

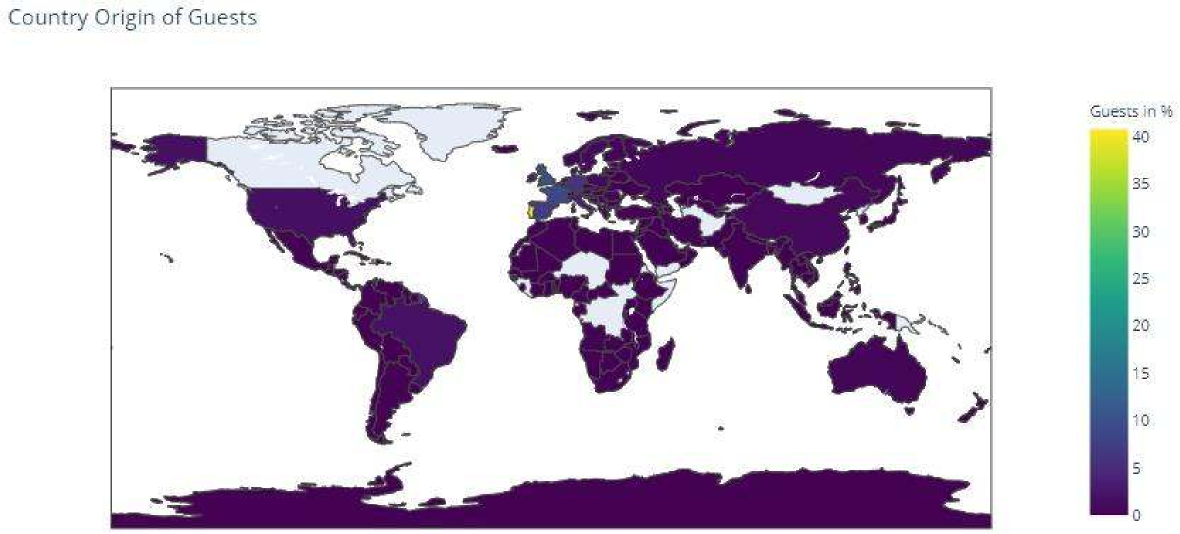
a. All guests including those canceled

```
In [100]: df_total["country_origin"].value_counts().head()
Out[100]: PRT    48483
          GBR    12120
          FRA    10401
          ESP     8560
          DEU     7285
          Name: country_origin, dtype: int64
```

Ritz Jager hotel guests as a whole when viewed from all reservations both divested and not most are from countries in Europe, namely Portugal, United Kingdom, France, Spain and

Germany. Furthermore, to make it easier to see the country of origin of the guests, visualization in the form of a map will be used. Here are the results of the visualization:

```
In [63]: country = pd.DataFrame(df_total["country_origin"].value_counts())
         country=country.rename(columns={"country_origin": "Number of Guests"})
         total_guests = country["Number of Guests"].sum()
         country["Guests in %"] = round(country["Number of Guests"] / total_guests
                                        * 100, 3)
         country["country_origin"] = country.index
         guest_map = px.choropleth(country,
                             locations=country.index,
                             color=country["Guests in %"],
                             hover_name=country.index,
                             color_continuous_scale="Viridis",
                             title="Country Origin of Guests")
         guest_map.show()
```



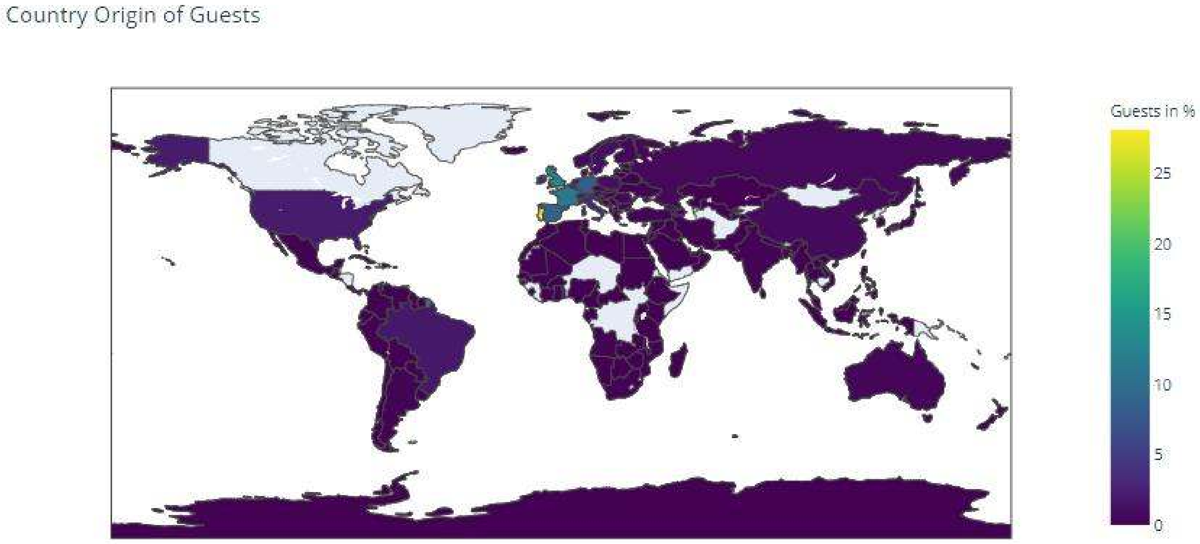
Guests of the Ritz Jager hotel spread almost all over the world from the continents of Europe, Asia, Africa, South America, North America to Australia. However, countries in Europe, especially Western Europe, Southern Europe and the United Kingdom are the most guests at the Ritz Jager hotel.

b. All guests do not include canceled

```
In [64]: df_noncancel["country_origin"].value_counts().head()

Out[64]: PRT    20977
         GBR     9668
         FRA     8468
         ESP     6383
         DEU     6067
         Name: country_origin, dtype: int64
```

Likewise with hotel customers who were not canceled, most Ritz hotel guests were from countries in Europe, namely Portugal, United Kingdom, France, Spain and Germany. Furthermore, to make it easier to see the country of origin of the guests, visualization in the form of a map will be used. Here are the results of the visualization:

```
In [14]: country = pd.DataFrame(df_noncancel["country_origin"].value_counts())
         country=country.rename(columns={"country_origin": "Number of Guests"})
         total_guests = country["Number of Guests"].sum()
         country["Guests in %"] = round(country["Number of Guests"] / total_guests
                                        * 100, 3)
         country["country_origin"] = country.index
         guest_map = px.choropleth(country,
                             locations=country.index,
                             color=country["Guests in %"],
                             hover_name=country.index,
                             color_continuous_scale="Viridis",
                             title="Country Origin of Guests")
         guest_map.show()
```



Guests of the Ritz Jager hotel spread almost all over the world from the continents of Europe, Asia, Africa, South America, North America to Australia. However, countries in Europe, especially Western Europe, Southern Europe and the United Kingdom are the most guests at the Ritz Jager hotel.

c. All Resort Hotel guests are not included as canceled

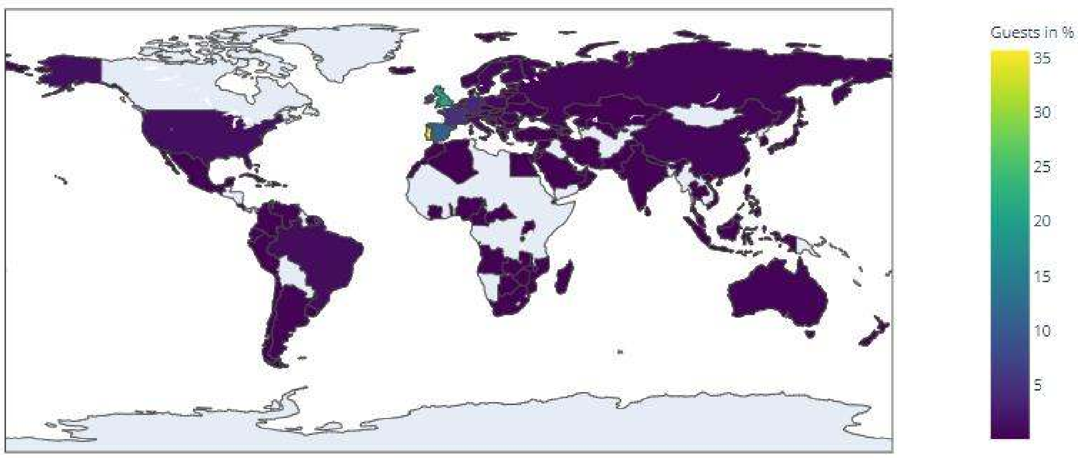
```
In [85]: resort_noncancel["country_origin"].value_counts().head()

Out[85]: PRT    10184
         GBR     5922
         ESP     3105
         IRL     1734
         FRA     1399
         Name: country_origin, dtype: int64
```

Next to the type of resort hotel, countries in the European region such as Portugal, United Kingdom, Spain, Ireland and France are the countries with the most number of guests at the Ritz Jager hotel. However, Germany is not in the top 5 while Ireland is in the top 5, which is ranked

4. The people of Ireland and the United Kingdom prefer to choose hotels with resort types.

```
In [16]: country = pd.DataFrame(resort_noncancel["country_origin"].value_counts())
         country=country.rename(columns={"country_origin": "Number of Guests"})
         total_guests = country["Number of Guests"].sum()
         country["Guests in %"] = round(country["Number of Guests"] / total_guests
                                        * 100, 3)
         country["country_origin"] = country.index
         guest_map = px.choropleth(country,
                                   locations=country.index,
                                   color=country["Guests in %"],
                                   hover_name=country.index,
                                   color_continuous_scale="Viridis",
                                   title="Country Origin of Guests")
         guest_map.show()
```

Country Origin of Guests



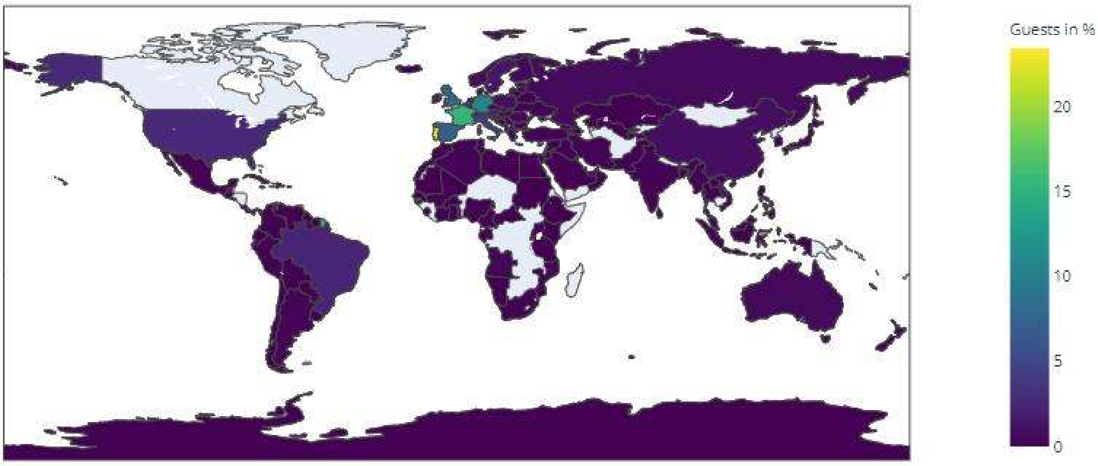
d. All City Hotel guests are not included as canceled

```
In [86]: city_noncancel["country_origin"].value_counts().head()

Out[86]: PRT    10793
         FRA     7069
         DEU     5010
         GBR     3746
         ESP     3278
         Name: country_origin, dtype: int64
```

Furthermore, for City hotel types, countries in the European region such as Portugal, France, Germany, United Kingdom, and Spain are the countries with the most number of guests at the Ritz Jager hotel. However, Ireland is not in the top 5, while Germany is in the top 5, which is ranked 3. The German and French people prefer to choose hotels of the City type.

```
In [18]: country = pd.DataFrame(city_noncancel["country_origin"].value_counts())
         country=country.rename(columns={"country_origin": "Number of Guests"})
         total_guests = country["Number of Guests"].sum()
         country["Guests in %"] = round(country["Number of Guests"] / total_guests
                                        * 100, 3)
         country["country_origin"] = country.index
         guest_map = px.choropleth(country,
                                   locations=country.index,
                                   color=country["Guests in %"],
                                   hover_name=country.index,
                                   color_continuous_scale="Viridis",
                                   title="Country Origin of Guests")
         guest_map.show()
```

Country Origin of Guests



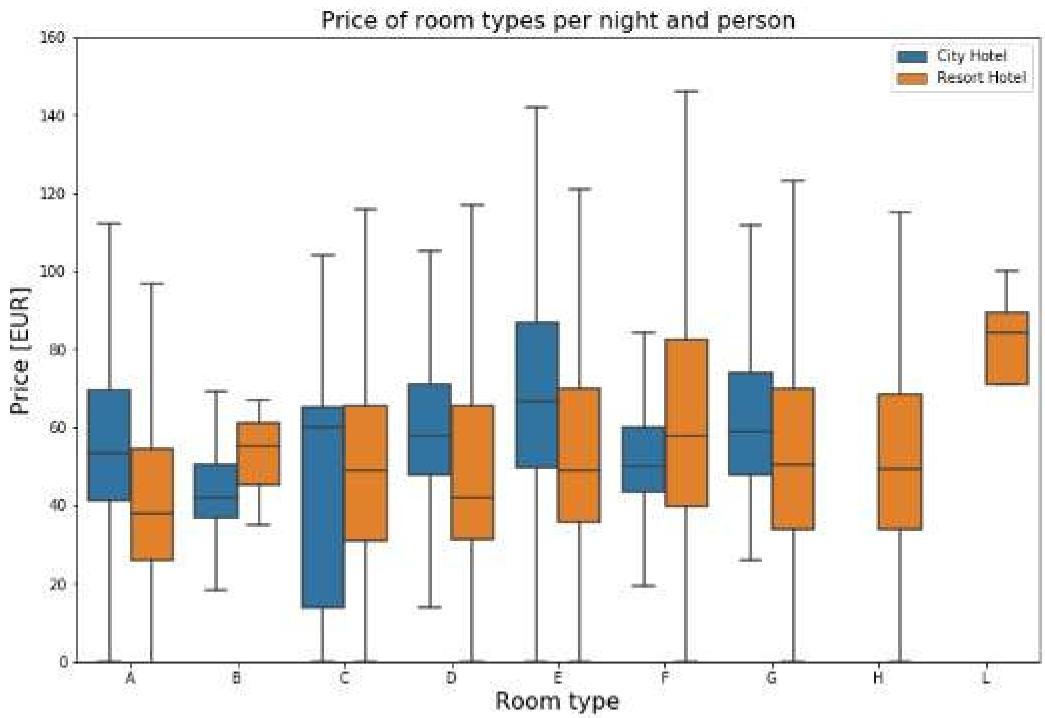
2. How much do guests pay for a room per night?

Both hotels have different room types and different meal arrangements. Seasonal factors are also important. So the prices vary a lot.

```
In [137]: resort_noncancel["adr_pp"] = resort_noncancel["adr"]/(resort_noncancel["adults"] + resort_noncancel["children"])
          city_noncancel["adr_pp"] = city_noncancel["adr"]/(city_noncancel["adults"] + city_noncancel["children"])

          print("""From all non-canceled bookings, across all room types and meals, the average prices are:
          Resort hotel: {:.2f} € per night and person.
          City hotel: {:.2f} € per night and person."""
                .format(resort_noncancel["adr_pp"].mean(),city_noncancel["adr_pp"].mean()))
```

From all non canceled bookings, across all room types and meals, the average prices are:
Resort Hotel : 47.49 € per night and person.
City Hotel : 59.27 € per night and person.

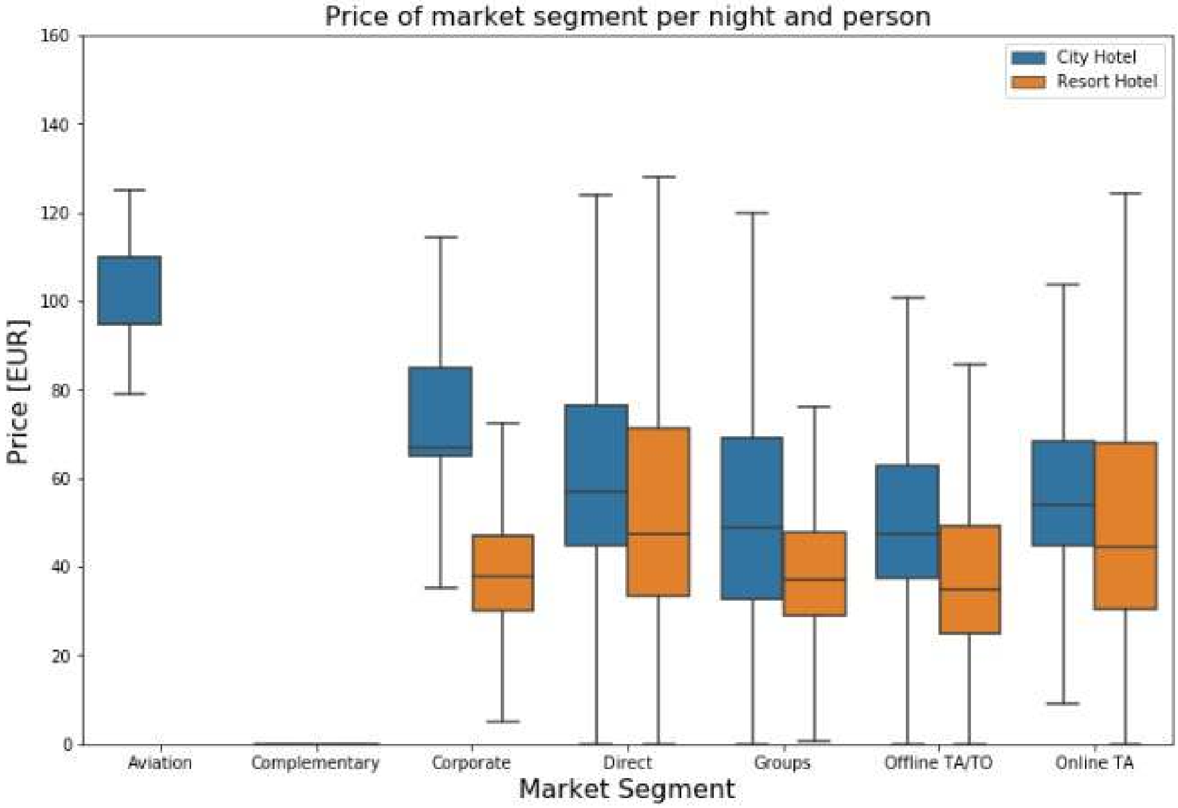
```python
In [20]: df_noncancel["adr_pp"] = df_noncancel["adr"] / (df_noncancel["adults"] +
                                                           df_noncancel["children"])
         room_prices = df_noncancel[["hotel_type", "reserved_room_type",
                                    "adr_pp"]].sort_values("reserved_room_type")

         # boxplot:
         plt.figure(figsize=(12, 8))
         sns.boxplot(x="reserved_room_type",
                     y="adr_pp",
                     hue="hotel_type",
                     data=room_prices,
                     hue_order=["City Hotel", "Resort Hotel"],
                     fliersize=0)
         plt.title("Price of room types per night and person", fontsize=16)
         plt.xlabel("Room type", fontsize=16)
         plt.ylabel("Price [EUR]", fontsize=16)
         plt.legend(loc="upper right")
         plt.ylim(0, 160)
         plt.show()
```



Furthermore, if the price is seen from the room type:

1. City hotel: room type E is the room type with the highest middle price, while room type B is the room type with the lowest middle price.
2. Resort Hotel: Room type L is the type of room with the highest middle price, while room type A is the room type with the lowest middle price.

```
In [11]: df_noncancel["adr_pp"] = df_noncancel["adr"] / (df_noncancel["adults"] +
                                                    df_noncancel["children"])
         room_prices = df_noncancel[["hotel_type", "market_segment",
                                     "adr_pp"]].sort_values("market_segment")

         # boxplot:
         plt.figure(figsize=(12, 8))
         sns.boxplot(x="market_segment",
                     y="adr_pp",
                     hue="hotel_type",
                     data=room_prices,
                     hue_order=["City Hotel", "Resort Hotel"],
                     fliersize=0)
         plt.title("Price of market segment per night and person", fontsize=16)
         plt.xlabel("Market Segment", fontsize=16)
         plt.ylabel("Price [EUR]", fontsize=16)
         plt.legend(loc="upper right")
         plt.ylim(0, 160)
         plt.show()
```



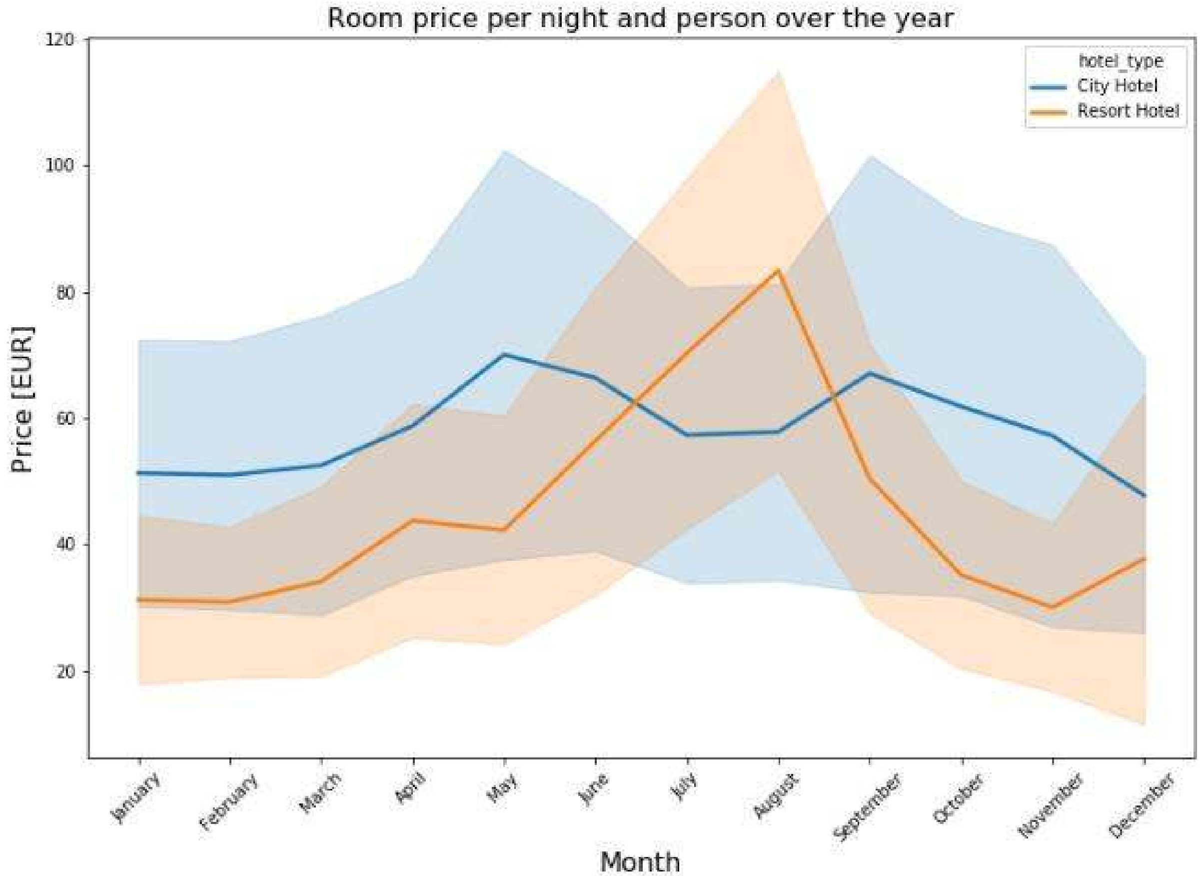
Furthermore, if the price is seen from the market segment:

1. City hotel: Market aviation segment is the market segment with the highest middle price value, while the offline TA / TO market segment is the market segment with the lowest middle price value.
2. Resort Hotels: Direct market segment is the market segment with the highest middle price, while the offline TA / TO market segment is the market segment with the lowest middle price.

3. How does the price per night vary over the year?

In this question, to measure variations in hotel rental prices, I use the average price per night and per person, regardless of room type and meals.

```
In [23]: room_prices_mothly = df_noncancel[["hotel_type", "arrival_date_month", "adr_pp"]].sort_values("arrival_date_month")

         # order by month:
         ordered_months = ["January", "February", "March", "April", "May", "June",
                   "July", "August", "September", "October", "November", "December"]
         room_prices_mothly["arrival_date_month"] = pd.Categorical(room_prices_mothly["arrival_date_month"],
                                                              categories=ordered_months, ordered=True)

         # barplot with standard deviation:
         plt.figure(figsize=(12, 8))
         sns.lineplot(x = "arrival_date_month", y="adr_pp", hue="hotel_type", data=room_prices_mothly,
                     hue_order = ["City Hotel", "Resort Hotel"], ci="sd", size="hotel_type", sizes=(2.5, 2.5))
         plt.title("Room price per night and person over the year", fontsize=16)
         plt.xlabel("Month", fontsize=16)
         plt.xticks(rotation=45)
         plt.ylabel("Price [EUR]", fontsize=16)
         plt.show()
```



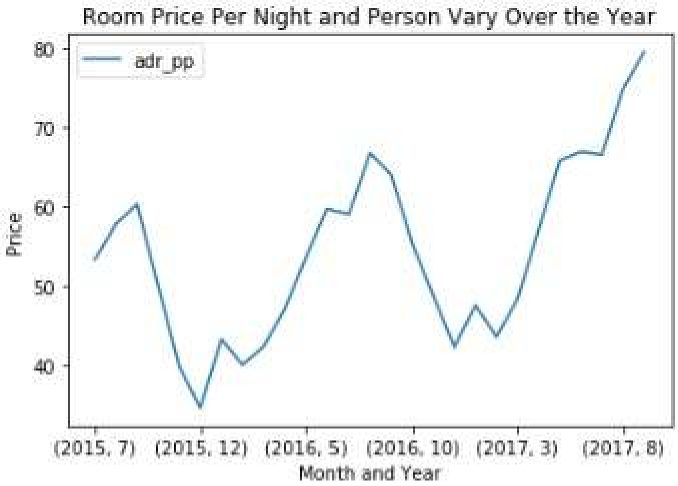
Based on the above plot it can be shown that prices at Resort hotels are much higher during summer in Europe ie June to September. As for City hotels the price varies less and is most expensive during spring in Europe, which is April to June and autumn in Europe, September to October.

```
In [143]: over_time=df_noncancel.groupby([df_noncancel["reservation_status_date"].dt.year,
                                          df_noncancel["reservation_status_date"].dt.month]).agg({"adr_pp":"mean"})
          over_time
```

Out[143]:

reservation_status_date	reservation_status_date	adr_pp
2015	7	53.345439
	8	57.798528
	9	60.306470
	10	50.067665
	11	39.838142
	12	34.545492
2016	1	43.277637
	2	40.022822
	3	42.314265
	4	47.052320
	5	53.542732
	6	59.708150
	7	59.003203
	8	66.773613
	9	64.045491
	10	55.411306
	11	48.783655
	12	42.278126
2017	1	47.565549
	2	43.563497
	3	48.376643
	4	57.119372
	5	65.820227
	6	66.938421
	7	68.564596
	8	74.875745
	9	79.573823

In [152]:
```
over_time.plot()
plt.title('Room Price Per Night and Person Vary Over the Year')
plt.xlabel('Month and Year')
plt.ylabel('Price')
```

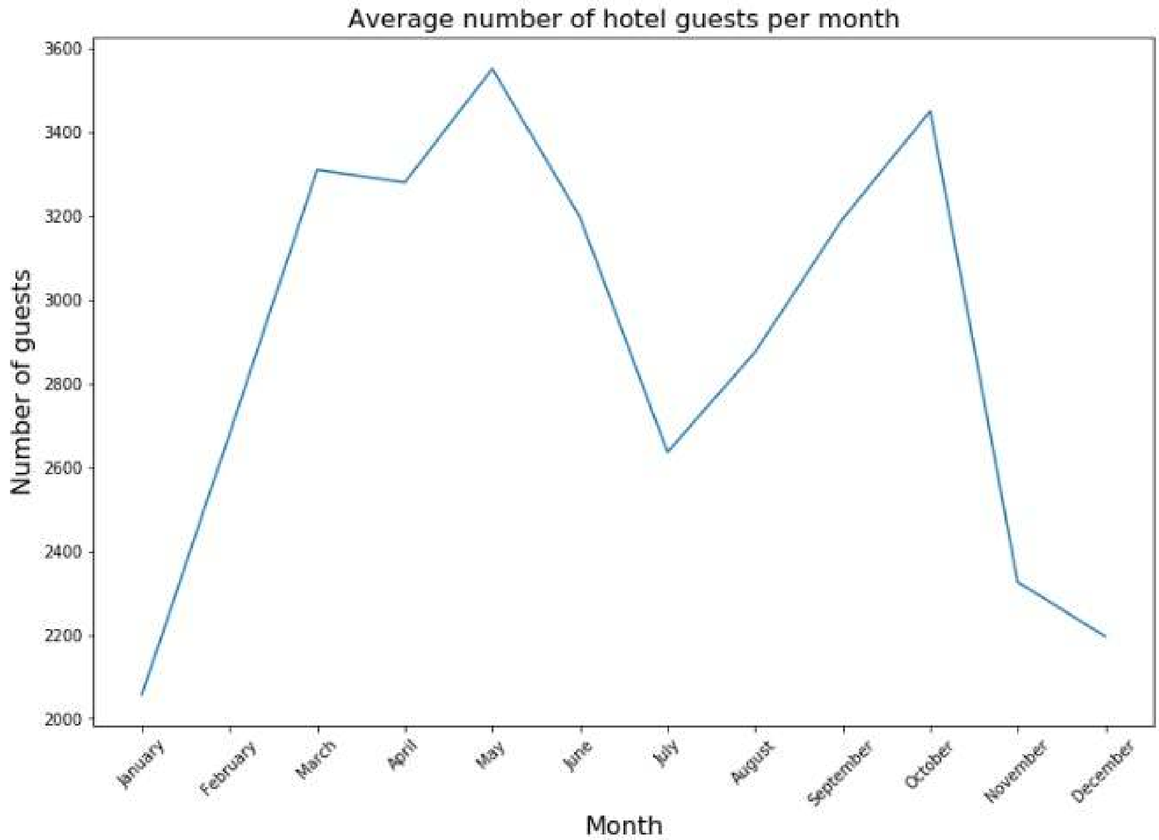


In addition, based on the plot above it can be seen that the price of room per night and people continues to increase throughout the year. Prices in 2017 are higher than prices in 2016, while prices in 2016 are higher than in 2015.

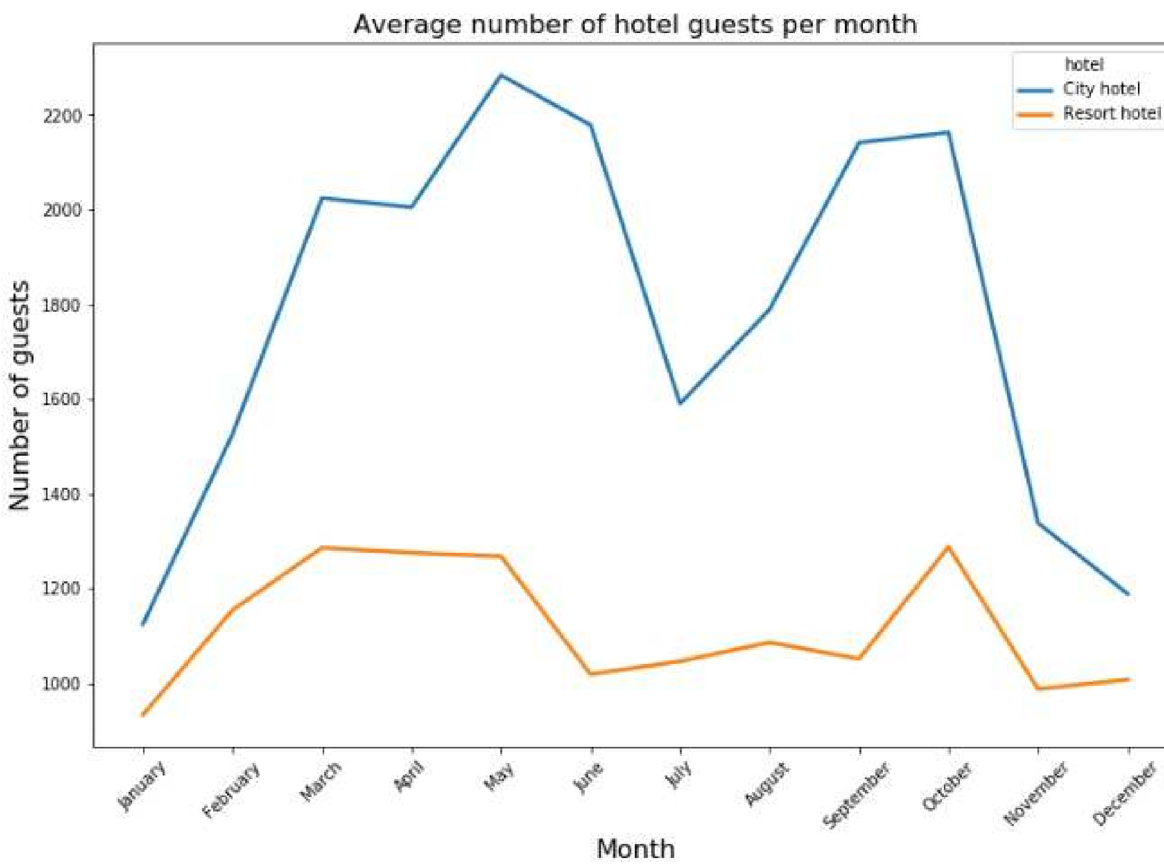
In [149]:
```
avgmonth=over_time['adr_pp'].pct_change()
plt.figure(figsize=(12,3))
plt.title('The Precentage Change of Price Per Night and Person Vary Over the Year')
plt.xlabel('Month and Year')
plt.ylabel('Percentage change (%)')
avgmonth.plot()
```

4. Which are the busiest months?

In [24]:
```
total_guests_monthly = df_noncancel.groupby("arrival_date_month")["hotel_type"].count()

total_guest_data = pd.DataFrame({"month": list(total_guests_monthly.index),
                                 "guests": list(total_guests_monthly.values)})

# order by month:
ordered_months = ["January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"]
total_guest_data["month"] = pd.Categorical(total_guest_data["month"], categories=ordered_months, ordered=True)

# Dataset contains July and August date from 3 years, the other month from 2 years. Normalize data:
total_guest_data.loc[(total_guest_data["month"] == "July") | (total_guest_data["month"] == "August"),
                     "guests"] /= 3
total_guest_data.loc[~((total_guest_data["month"] == "July") | (total_guest_data["month"] == "August")),
                     "guests"] /= 2

#show figure:
plt.figure(figsize=(12, 8))
sns.lineplot(x = "month", y="guests", data=total_guest_data, sizes=(2.5, 2.5))
plt.title("Average number of hotel guests per month", fontsize=16)
plt.xlabel("Month", fontsize=16)
plt.xticks(rotation=45)
plt.ylabel("Number of guests", fontsize=16)
plt.show()
```



```
In [25]: resort_guests_monthly = resort_noncancel.groupby("arrival_date_month")["hotel_type"].count()
city_guests_monthly = city_noncancel.groupby("arrival_date_month")["hotel_type"].count()

resort_guest_data = pd.DataFrame({"month": list(resort_guests_monthly.index),
                    "hotel": "Resort hotel",
                    "guests": list(resort_guests_monthly.values)})

city_guest_data = pd.DataFrame({"month": list(city_guests_monthly.index),
                    "hotel": "City hotel",
                    "guests": list(city_guests_monthly.values)})

full_guest_data = pd.concat([resort_guest_data,city_guest_data], ignore_index=True)

# order by month:
ordered_months = ["January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"]
full_guest_data["month"] = pd.Categorical(full_guest_data["month"], categories=ordered_months, ordered=True)

# Dataset contains July and August date from 3 years, the other month from 2 years. Normalize data:
full_guest_data.loc[(full_guest_data["month"] == "July") | (full_guest_data["month"] == "August"),
                    "guests"] /= 3
full_guest_data.loc[~((full_guest_data["month"] == "July") | (full_guest_data["month"] == "August")),
                    "guests"] /= 2

#show figure:
plt.figure(figsize=(12, 8))
sns.lineplot(x = "month", y="guests", hue="hotel", data=full_guest_data,
             hue_order = ["City hotel", "Resort hotel"], size="hotel", sizes=(2.5, 2.5))
plt.title("Average number of hotel guests per month", fontsize=16)
plt.xlabel("Month", fontsize=16)
plt.xticks(rotation=45)
plt.ylabel("Number of guests", fontsize=16)
plt.show()
```

The Ritz jager dataset was taken from 1 July 2015 to 31 August 2017. Logically the number of guests in July and August was the highest, this is because the data for July and August were recorded 3 times, namely 2015, 2016 and 2017. While other months were only recorded twice. Therefore, to find out which is the busiest month or the month in which the most number of guests are used, the average value of the number of guests will be used. City Hotels have more guests during the European spring, April to June and autumn in Europe, September to October, although during that season prices are also highest. The peak month with the most number of guests for City hotels is in May. Whereas in January, February, July, November and December there are fewer visitors, even though prices are lower. In addition, the number of guests for Resort hotels decreased from June to September, which is also when prices are highest. The peak month with the highest number of guests for Resort hotels is in October. Both hotels have the fewest guests during the winter (November - January).

5. How long do people stay at the hotels?

In [77]:
```
# Create a DateFrame with the relevant data:
resort_noncancel["total_nights"] = resort_noncancel["stays_in_weekend_nights"] + resort_noncancel["stays_in_week_nights"]
city_noncancel["total_nights"] = city_noncancel["stays_in_weekend_nights"] + city_noncancel["stays_in_week_nights"]

num_nights_res = list(resort_noncancel["total_nights"].value_counts().index)
num_bookings_res = list(resort_noncancel["total_nights"].value_counts())
rel_bookings_res = resort_noncancel["total_nights"].value_counts() / sum(num_bookings_res) * 100 # convert to percent

num_nights_cty = list(city_noncancel["total_nights"].value_counts().index)
num_bookings_cty = list(city_noncancel["total_nights"].value_counts())
rel_bookings_cty = city_noncancel["total_nights"].value_counts() / sum(num_bookings_cty) * 100 # convert to percent

res_nights = pd.DataFrame({"hotel": "Resort hotel",
                           "num_nights": num_nights_res,
                           "num_bookings":num_bookings_res,
                           "rel_num_bookings": rel_bookings_res})

cty_nights = pd.DataFrame({"hotel": "City hotel",
                           "num_nights": num_nights_cty,
                           "num_bookings":num_bookings_cty,
                           "rel_num_bookings": rel_bookings_cty})
```

In [91]:
```
res_nights.head()
```

Out[91]:

	hotel	num_nights	num_bookings	rel_num_bookings
1	Resort hotel	1	6579	22.743458
2	Resort hotel	2	4488	15.514917
7	Resort hotel	7	4434	15.328240
3	Resort hotel	3	3828	13.233311
4	Resort hotel	4	3321	11.480624

In [89]:
```
cty_nights.head()
```

Out[89]:

	hotel	num_nights	num_bookings	rel_num_bookings
3	City hotel	3	11889	25.798542
2	City hotel	2	10983	23.832567
1	City hotel	1	9155	19.865897
4	City hotel	4	7694	16.695599
5	City hotel	5	3210	6.965541

For city hotels there is a clear preference for 1-4 nights staying in hotels. As for resort hotels, 1-4 nights are also often booked, but 7 nights are also very popular.

In [97]:
```
avg_nights_res = sum(list((res_nights["num_nights"] * (res_nights["rel_num_bookings"]/100)).values))
avg_nights_cty = sum(list((cty_nights["num_nights"] * (cty_nights["rel_num_bookings"]/100)).values))
print(f"On average, guests of the City hotel stay {avg_nights_cty:.2f} nights, and {cty_nights['num_nights'].max()} at maximum.")
print(f"On average, guests of the Resort hotel stay {avg_nights_res:.2f} nights, and {res_nights['num_nights'].max()} at maximum.
```

```
On average, guests of the City hotel stay 2.92 nights, and 48 at maximum.
On average, guests of the Resort hotel stay 4.14 nights, and 69 at maximum.
```

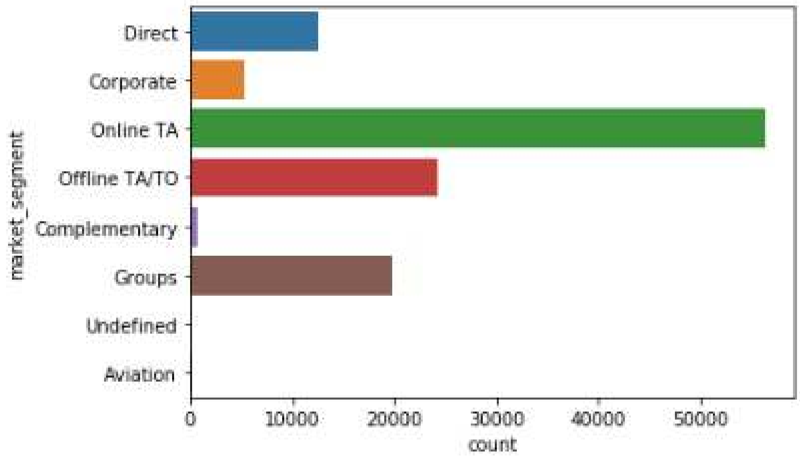
As for the average length of stay, city hotels have an average of 3 nights while resort hotels have an average of 4 nights.

6. Bookings by market segment

a. All of Bookings

```
In [28]: df["market_segment"].value_counts()

Out[28]: Online TA        56408
         Offline TA/TO    24182
         Groups           19791
         Direct           12582
         Corporate         5282
         Complementary      728
         Aviation           235
         Undefined            2
         Name: market_segment, dtype: int64
```

```
In [29]: sns.countplot(y=df["market_segment"],orient="h")
```

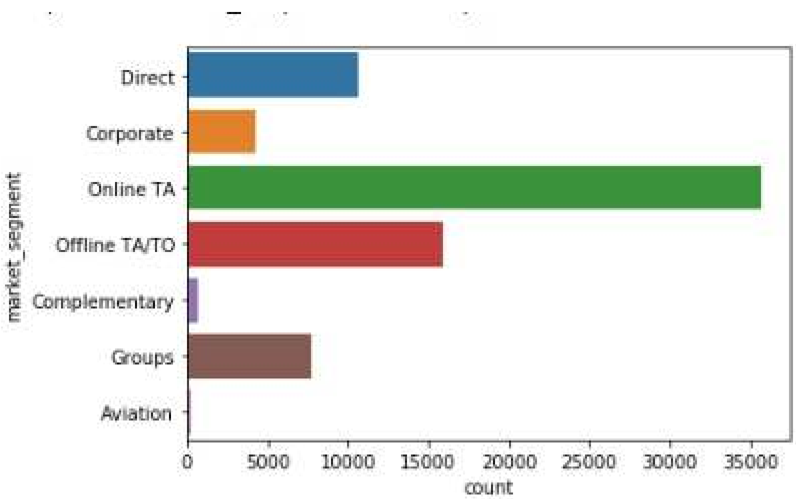


Based on all existing and unrelated bookings, online TA is the most effective market segment because there are more than 50000 bookings from that market segment. While rated 2 and 3, there are offline market segments TA / TO and Groups. In addition, aviation is the least effective market segment because it only produces 235 bookings.

b. All of Bookings are not canceled

```
In [30]: df_noncancel["market_segment"].value_counts()

Out[30]: Online TA        35673
         Offline TA/TO    15880
         Direct           10648
         Groups            7697
         Corporate         4291
         Complementary      639
         Aviation           183
         Name: market_segment, dtype: int64
```

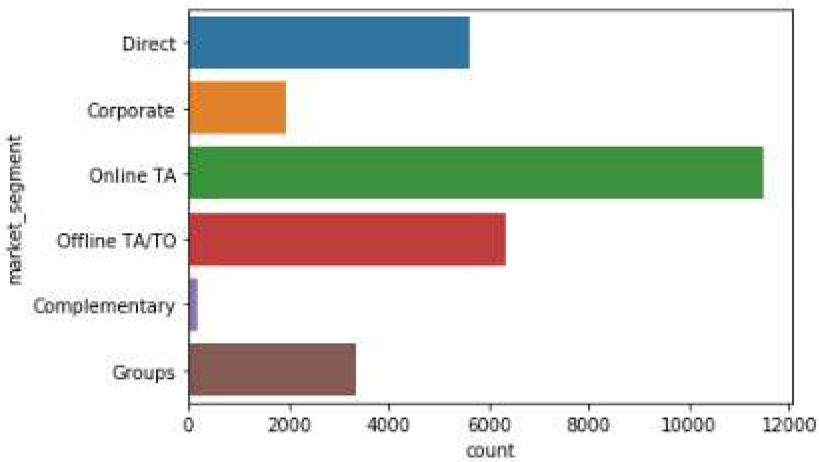
```
In [31]: sns.countplot(y=df_noncancel["market_segment"],orient="h")
```



Based on non-canceling orders, TA online is the most effective market segment because there are more than 35,000 orders from that market segment. While rated 2 and 3, there are offline market segments TA / TO and Direct. In addition, aviation is the least effective market segment because it only produces 183 orders.

c. All of Bookings are not canceled at the Resort Hotel

```
In [32]: resort_noncancel["market_segment"].value_counts()

Out[32]: Online TA        11481
         Offline TA/TO     6334
         Direct            5632
         Groups            3358
         Corporate         1954
         Complementary      168
         Name: market_segment, dtype: int64

In [33]: sns.countplot(y=resort_noncancel["market_segment"],orient="h")
```

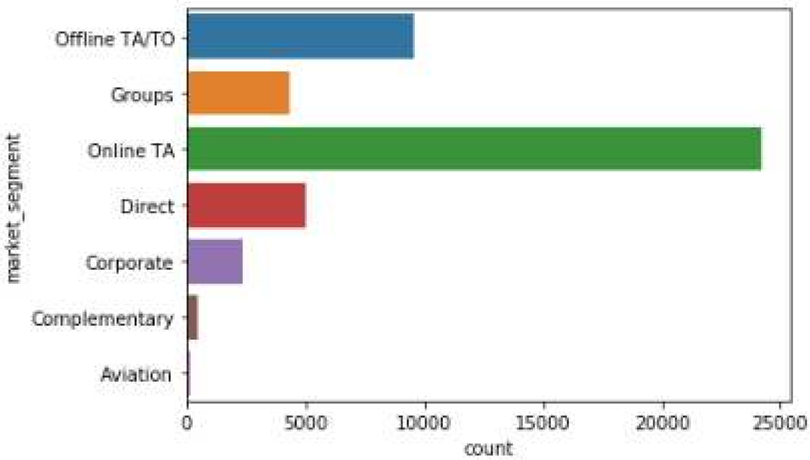


Based on non-canceling bookings at resort hotels, TA online is the most effective market segment because there are more than 11481 bookings from the market segment. While rated 2 and 3, there are offline market segments TA / TO and Direct. In addition, complementary is the least effective market segment because it only produces 168 orders.

d. All of Bookings are not canceled at the City Hotel

```
In [34]: city_noncancel["market_segment"].value_counts()

Out[34]: Online TA        24192
         Offline TA/TO     9546
         Direct            5016
         Groups            4339
         Corporate         2337
         Complementary      471
         Aviation           183
         Name: market_segment, dtype: int64
```

```
In [35]: sns.countplot(y=city_noncancel["market_segment"],orient="h")
```



Based on non-canceling bookings at city hotels, TA online is the most effective market segment because there are more than 24192 bookings from that market segment. While rated 2 and 3, there are offline market segments TA / TO and Direct. In addition, complementary is the least effective market segment because it only produces 183 orders.

7. How many bookings were cancelled?

```
In [36]: cancel1=df[df["is_canceled"]==1]["is_canceled"].count()
         percent1=cancel1/len(df)*100
         print("The total canceled orders is equal to "+str(cancel1)+" or "+str(round(percent1,2))+"%")

         The total canceled orders is equal to 44199 or 37.08%
```

```
In [37]: cancel2=df[(df["is_canceled"]==1) & (df["hotel_type"]=="Resort Hotel")]["is_canceled"].count()
         percent2=cancel2/len(df[df["hotel_type"]=="Resort Hotel"])*100
         print("The total canceled orders for Resort Hotel is equal to "+str(cancel2)+" or "+str(round(percent2,2))+"%")

         The total canceled orders for Resort Hotel is equal to 11120 or 27.77%
```

```
In [38]: cancel3=df[(df["is_canceled"]==1) & (df["hotel_type"]=="City Hotel")]["is_canceled"].count()
         percent3=cancel3/len(df[df["hotel_type"]=="City Hotel"])*100
         print("The total canceled orders for City Hotel is equal to "+str(cancel3)+" or "+str(round(percent3,2))+"%")

         The total canceled orders for City Hotel is equal to 33079 or 41.79%
```

Total bookings canceled is 44199 (37%);
Resort hotel bookings canceled is 11120 (28%);
City hotel bookings canceled is 33079 (42%).

8. Which month has the highest number of cancellations?

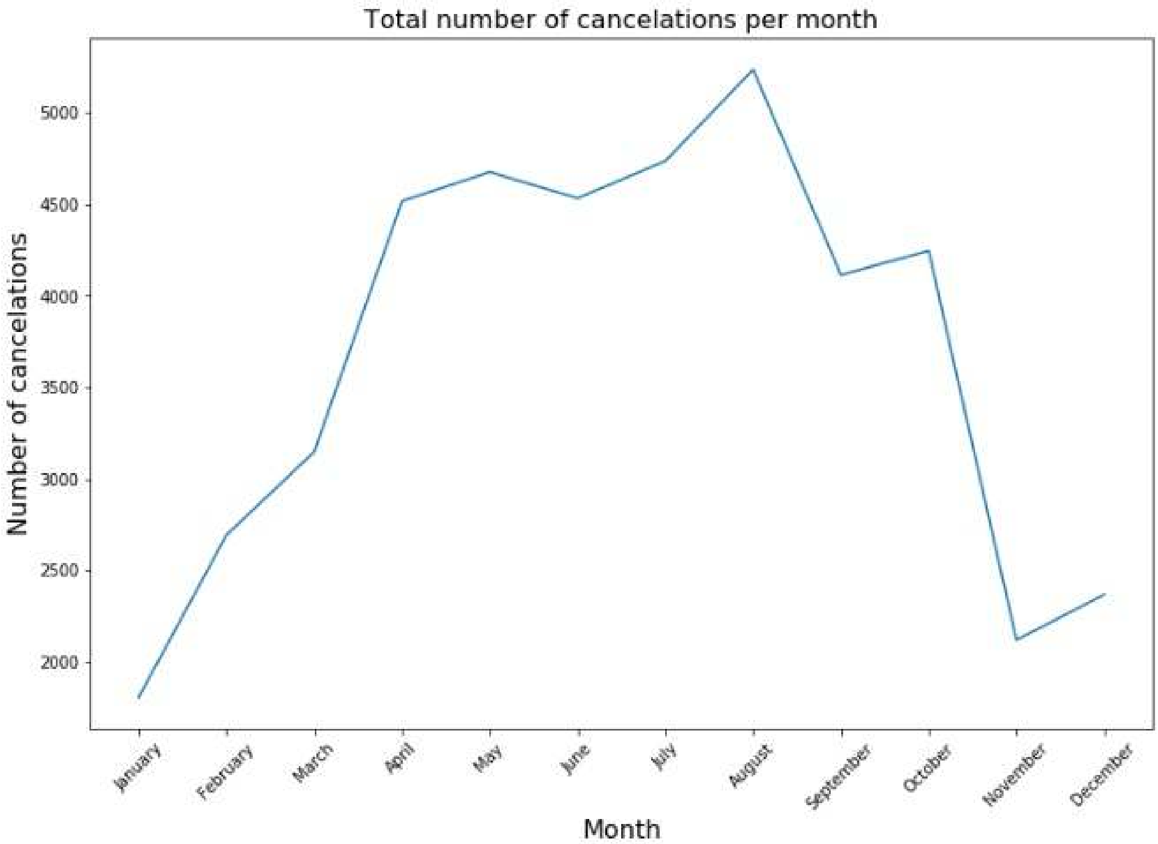
a. All of hotel types

In [39]:
```
df_group3=df.groupby(["arrival_date_month"]).agg({"is_canceled":"sum","arrival_date_month":"count"})
df_group3=df_group3.rename(columns={"is_canceled":"Number of Cancellations","arrival_date_month":"Number of Bookings"})
df_group3["Percent"]=df_group3["Number of Cancellations"]/df_group3["Number of Bookings"]
df_group3.sort_values("Percent",ascending=False)
```

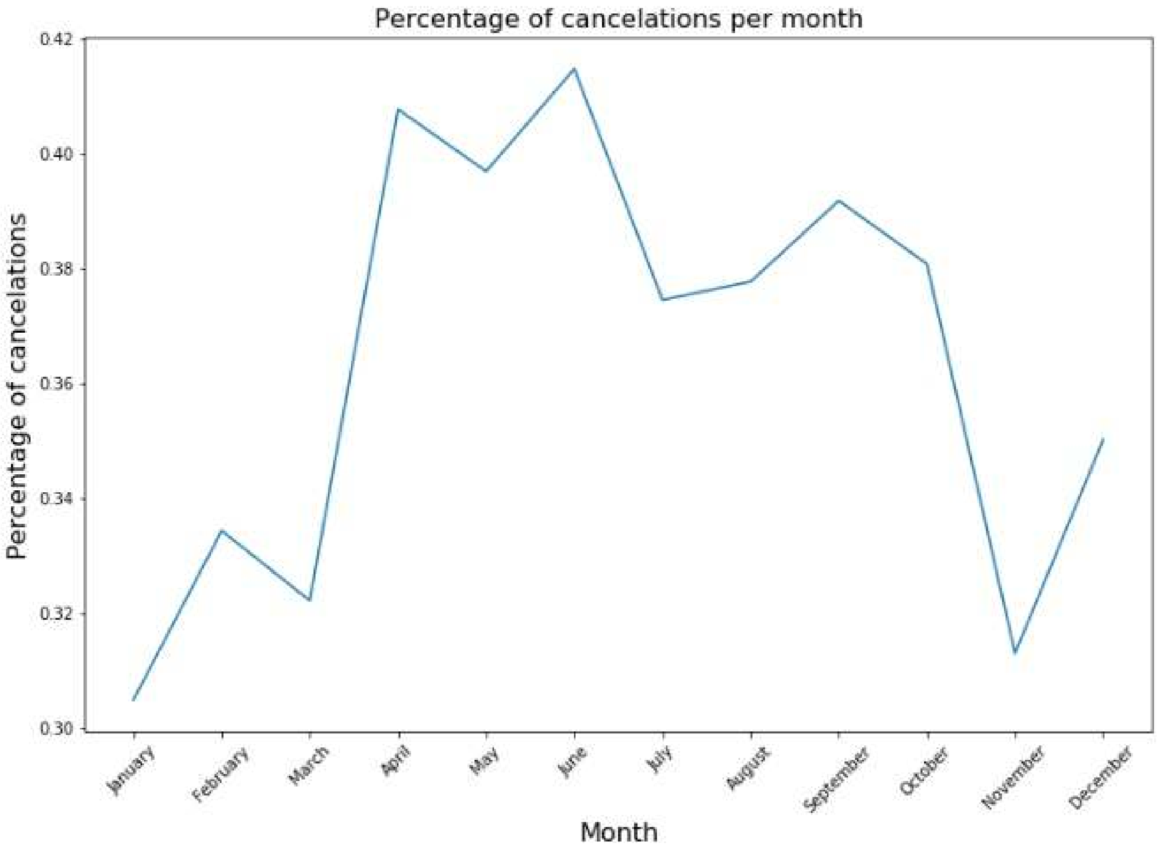
Out[39]:

arrival_date_month	Number of Cancellations	Number of Bookings	Percent
June	4534	10929	0.414860
April	4518	11078	0.407835
May	4677	11780	0.397029
September	4115	10500	0.391905
October	4246	11147	0.380910
August	5237	13861	0.377823
July	4737	12644	0.374644
December	2368	6759	0.350348
February	2693	8052	0.334451
March	3148	9768	0.322277
November	2120	6771	0.313100
January	1806	5921	0.305016

In [40]:
```
df_group3=df_group3.reset_index()
ordered_months = ["January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"]
df_group3["arrival_date_month"] = pd.Categorical(df_group3["arrival_date_month"], categories=ordered_months, ordered=True)

#show figure:
plt.figure(figsize=(12, 8))
sns.lineplot(x = "arrival_date_month", y="Number of Cancellations", data=df_group3)
plt.title("Total number of cancelations per month", fontsize=16)
plt.xlabel("Month", fontsize=16)
plt.xticks(rotation=45)
plt.ylabel("Number of cancelations", fontsize=16)
plt.show()
```



```
In [41]: df_group3=df_group3.reset_index()
df_group3["arrival_date_month"] = pd.Categorical(df_group3["arrival_date_month"], categories=ordered_months, ordered=True)

#show figure:
plt.figure(figsize=(12, 8))
sns.lineplot(x = "arrival_date_month", y="Percent", data=df_group3)
plt.title("Percentage of cancelations per month", fontsize=16)
plt.xlabel("Month", fontsize=16)
plt.xticks(rotation=45)
plt.ylabel("Percentage of cancelations", fontsize=16)
plt.show()
```



The highest number of cancelations occurred in August, which was 5237 cancelations. Then in July there were 4737 cancellations. And the third is 4677 cancelations in May. July and August have the highest cancelation rate because the data were taken from July 1 2015 to August 31 2017, so that July and August were recorded 3 times while other months were only recorded 2 times. Therefore the percentage of cancelation value will be used which is sought from the number of cancelation in that month divided by the number of orders in that month. When viewed from the percentage cancelation rate, June is the month with the highest percentage of cancelation followed by April and May in the second and third ranks.

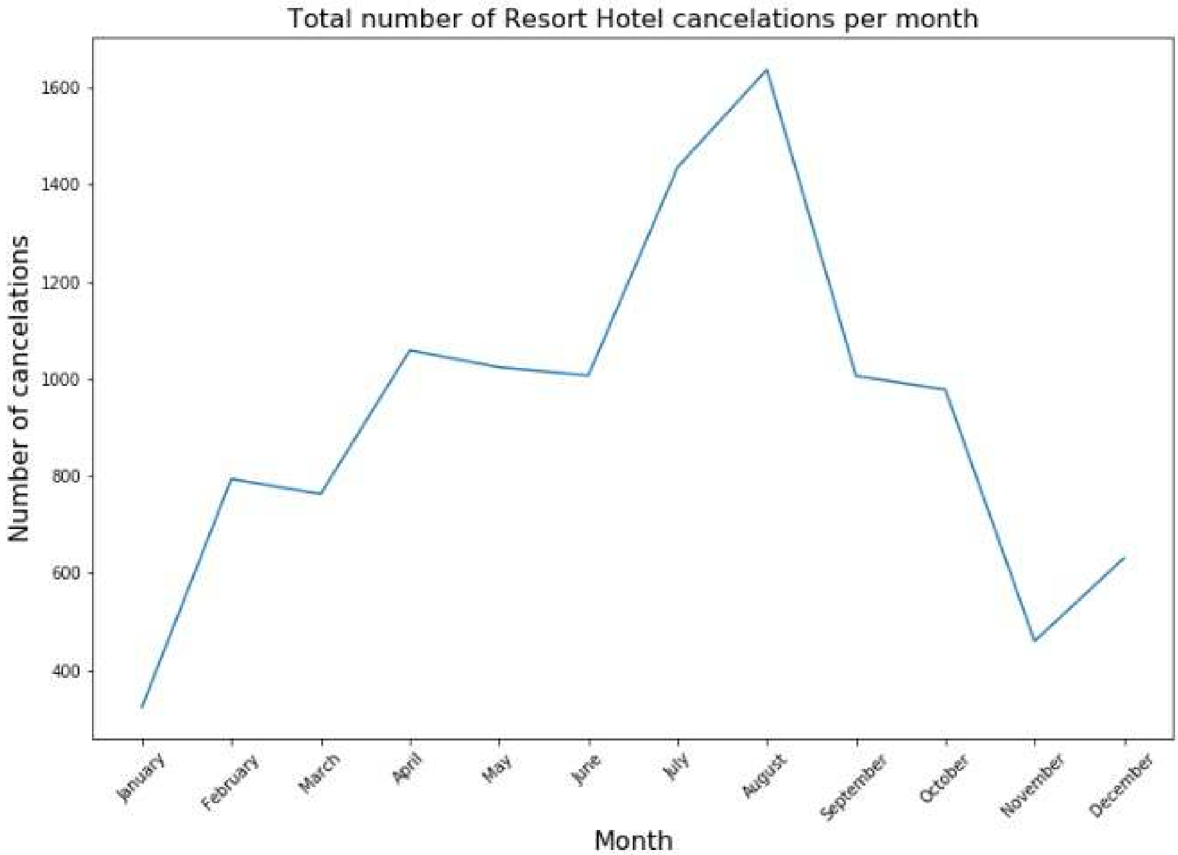
b. Resort Hotel

In [98]:
```
df_group4=df[df["hotel_type"]=="Resort Hotel"].groupby(["arrival_date_month"]).agg({"is_canceled":"sum","arrival_date_month":
                                                                                                                    "count"})
df_group4=df_group4.rename(columns={"is_canceled":"Number of Cancellations","arrival_date_month":"Number of Bookings"})
df_group4["Percent"]=df_group4["Number of Cancellations"]/df_group4["Number of Bookings"]
df_group4.sort_values("Percent",ascending=False)
```

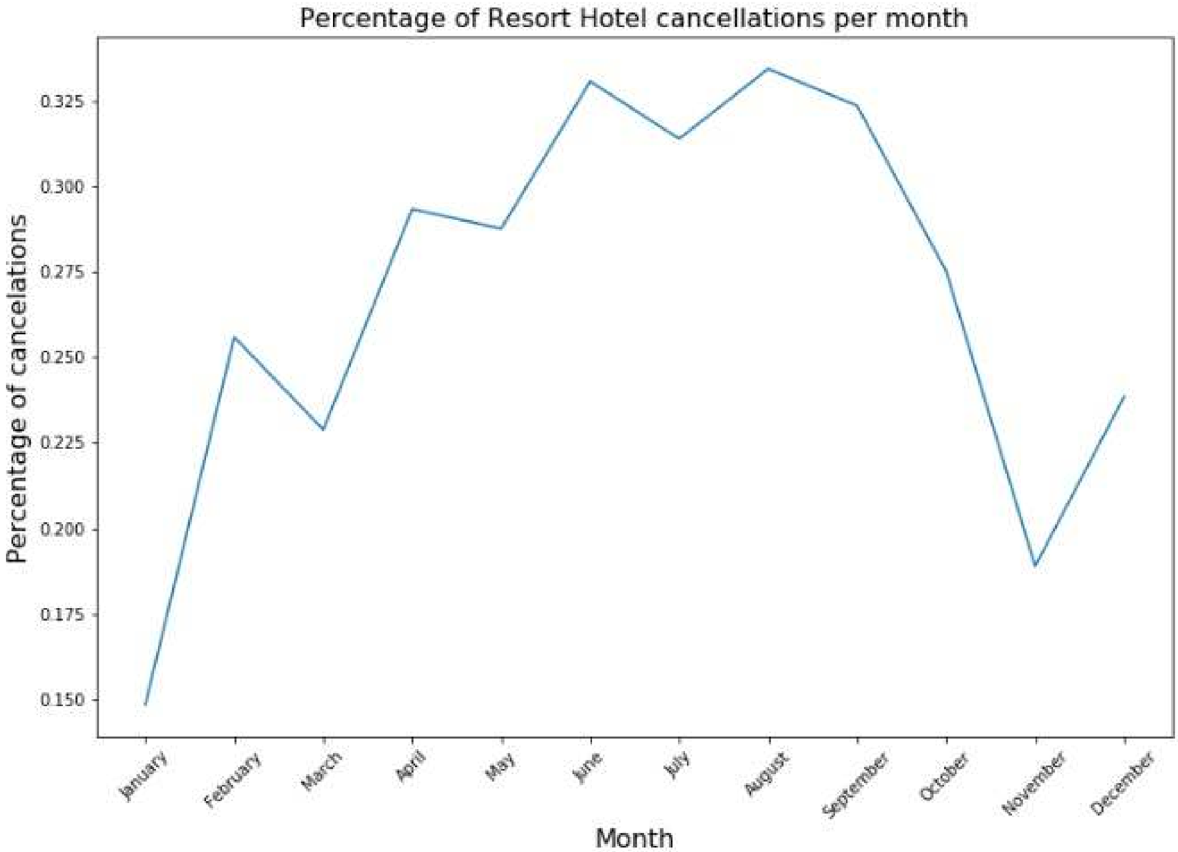
Out[98]:

arrival_date_month	Number of Cancellations	Number of Bookings	Percent
August	1637	4894	0.334491
June	1007	3044	0.330815
September	1006	3108	0.323681
July	1436	4573	0.314017
April	1059	3609	0.293433
May	1024	3559	0.287721
October	978	3553	0.275260
February	794	3102	0.255964
December	631	2645	0.238563
March	763	3334	0.228854
November	460	2435	0.188912
January	325	2191	0.148334

In [43]:
```
df_group4=df_group4.reset_index()
df_group4["arrival_date_month"] = pd.Categorical(df_group4["arrival_date_month"], categories=ordered_months, ordered=True)

#show figure:
plt.figure(figsize=(12, 8))
sns.lineplot(x = "arrival_date_month", y="Number of Cancellations", data=df_group4)
plt.title("Total number of Resort Hotel cancelations per month", fontsize=16)
plt.xlabel("Month", fontsize=16)
plt.xticks(rotation=45)
plt.ylabel("Number of cancelations", fontsize=16)
plt.show()
```




```
In [44]: df_group4=df_group4.reset_index()
         df_group4["arrival_date_month"] = pd.Categorical(df_group4["arrival_date_month"], categories=ordered_months, ordered=True)

         #show figure:
         plt.figure(figsize=(12, 8))
         sns.lineplot(x = "arrival_date_month", y="Percent", data=df_group4)
         plt.title("Percentage of Resort Hotel cancellations per month", fontsize=16)
         plt.xlabel("Month", fontsize=16)
         plt.xticks(rotation=45)
         plt.ylabel("Percentage of cancelations", fontsize=16)
         plt.show()
```



The highest number of cancelations at Resort hotels occurred in August with 1637 cancelations. Then in July there were 1436 cancelations. And the third is in April, namely as many as 1059 cancelations. July and August have the highest cancelation rate because the data were taken from July 1 2015 to August 31 2017, so that July and August were recorded 3 times while other months were only recorded 2 times. Therefore the percentage of cancelation value will be used which is sought from the number of cancelation in that month divided by the number of orders in that month. When viewed from the percentage of cancelation, August is summer is the month with the highest percentage of cancelation which is 33.4% and is followed by June and September in the second and third ranks.

c. City Hotel

In [99]:
```
df_group5=df[df["hotel_type"]=="City Hotel"].groupby(["arrival_date_month"]).agg({"is_canceled":"sum","arrival_date_month":
                                                                                  "count"})
df_group5=df_group5.rename(columns={"is_canceled":"Number of Cancellations","arrival_date_month":"Number of Bookings"})
df_group5["Percent"]=df_group5["Number of Cancellations"]/df_group5["Number of Bookings"]
df_group5.sort_values("Percent",ascending=False)
```

Out[99]:

arrival_date_month	Number of Cancellations	Number of Bookings	Percent
April	3459	7469	0.463114
June	3527	7885	0.447305
May	3653	8221	0.444350
October	3268	7594	0.430340
December	1737	4114	0.422217
September	3109	7392	0.420590
July	3301	8071	0.408995
August	3600	8967	0.401472
January	1481	3730	0.397051
February	1899	4950	0.383636
November	1660	4336	0.382841
March	2385	6434	0.370687

In [46]:
```
df_group5=df_group5.reset_index()
df_group5["arrival_date_month"] = pd.Categorical(df_group5["arrival_date_month"], categories=ordered_months, ordered=True)

#show figure:
plt.figure(figsize=(12, 8))
sns.lineplot(x = "arrival_date_month", y="Number of Cancellations", data=df_group5)
plt.title("Total number of City Hotel cancelations per month", fontsize=16)
plt.xlabel("Month", fontsize=16)
plt.xticks(rotation=45)
plt.ylabel("Number of cancelations", fontsize=16)
plt.show()
```