Exercise 1: Python for Data **Analysis with Google Colab**

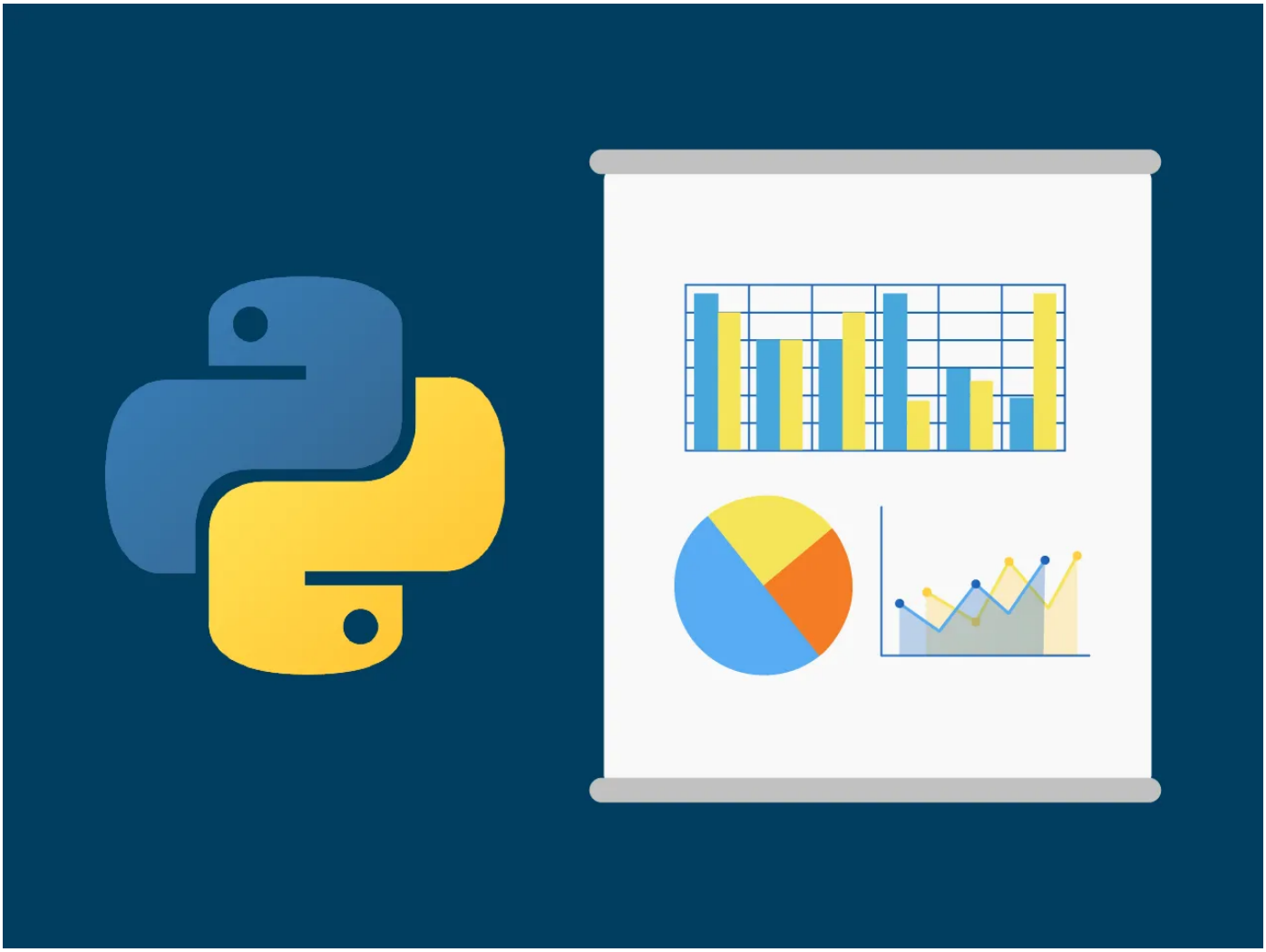


Image from Cognitive Class

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted in many fields.

Python has become a staple in data science, allowing data analyst and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyze data, and complete other data-related tasks.

Colaboratory (Colab) is a notebook (like a Jupyter Notebook) where we can run Python code in our Google Drive. We can write text, write code, run that code, and see the output all in line in the same notebook. You can check here to know how to add Colab to your Google Drive.

Image from Hwlibre

The first thing we do is importing several Python libraries that we need to do the exercise. There are:

- **Pandas library (pd).** It is used for working with data sets and has functions for analyzing, cleaning, exploring, and manipulating data. Pandas stands for Python Data Analysis. Click here for more

information.

- **NumPy library (np).** It is used for working with arrays and has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy stands for Numerical Python. Click here for more information.
- **Matplotlib libary.** It is used to create graphs and plots by using Python scripts. It has a module named **pyplot (plt)** which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. Click here for more infromation.
- **Seaborn library (sns).** It is a Python data visualization library based on matplotlib and provides a high-level interface for drawing

- **Regular Expression module (re).** It is a sequence characters that forms a search pattern and can be used to check if a string contains the specified search pattern. Click here for more information.

Here is the code we can use to import those libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
sns.set()
```

After that, we import the dataset. For this exercise, we're going to use Hotel Booking Demand Dataset (hotels.csv). For importing the dataset in CSV from the URL link, we can use this code:

```
df_hotels = pd.read_csv('https://raw.githubusercontent.com/rfordatascience/t
```

The dataset doesn't have unique identifier column, so we add the 'id' column as a unique identifier. Here is the code we can use:

```
df_hotels = df_hotels.reset_index().rename(columns={'index':'id'})
```

Now, we're ready to do the exercise!

> *Question 1:*
> *Create a function with one argument formed in DataFrame to check the data type, the number of null values, the percentage of null values and the number of unique values for each column!*

To answer the question above, we can use this code to create function to check the needed values above for each column in DataFrame:

```
def check_values(df):
    data = []
```

```
    for column in df.columns:
      data.append([
                   column, \
                   df[column].dtype, \
                   df[column].isna().sum(), \
                   round(100*(df[column].isna().sum()/len(df)),2), \
                   df[column].nunique()
                   ])
    return pd.DataFrame(columns=['Data_Features','Data_Type','Null','Null_Pe
```

And here we can check the needed values for each column of the DataFrame using this code:

```
check_values(df_hotels)
```

The output of the code:

	Data_Features	Data_Type	Null	Null_Percentage	Unique_Value
0	id	int64	0	0.00	119390
1	hotel	object	0	0.00	2
2	is_canceled	int64	0	0.00	2
3	lead_time	int64	0	0.00	479
4	arrival_date_year	int64	0	0.00	3
5	arrival_date_month	object	0	0.00	12
6	arrival_date_week_number	int64	0	0.00	53
7	arrival_date_day_of_month	int64	0	0.00	31
8	stays_in_weekend_nights	int64	0	0.00	17
9	stays_in_week_nights	int64	0	0.00	35
10	adults	int64	0	0.00	14

> ***Question 2:***
> *How many visitors are there who cancel the reservation and who don't? And from that number draw conclusions about the proportions of each!*

First, we can use the ***value_counts()*** to count value of the column we need. As the question asked is how many visitors who cancel the reservation and who don't, so the column needed is 'is_canceled' column. So, here is the code we can use:

```
df_hotels.is_canceled.value_counts()
```

The one who cancel the reservation is valued as 1 and the one who don't is valued as 0 in 'is_canceled' column. Here is the output of the code:

```
0    75166
1    44224
Name: is_canceled, dtype: int64
```

Secondly, we can use the ***normalize=True*** inside the brackets of ***value_counts()*** to normalize the values so we can see the proportion of each value. So, here is the code we can use:

```
df_hotels.is_canceled.value_counts(normalize=True)
```

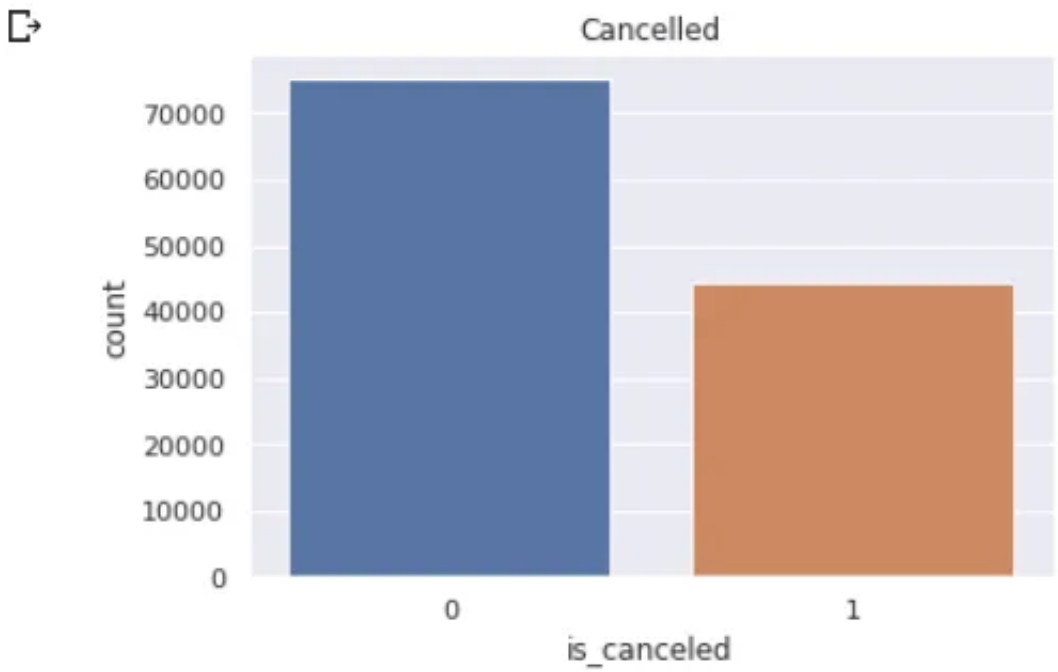
Here is the output of the code:

```
0    0.629584
1    0.370416
Name: is_canceled, dtype: float64
```

At last, for better way to know the proportion of the values, we can visualize the values above with the countplot. Here is the code we can use:

```
sns.countplot(data=df_hotels, x='is_canceled')
plt.title('Cancelled')
plt.show()
```

Here is the output of the code:



> *Question 3:*
> *a) For "City Hotel", what is the percentage of canceled reservations?*
> *b) For "Resort Hotel", what is the percentage of canceled reservations?*
> *c) What type of hotel that has the bigger percentage of canceled reservations?*

For the specific condition above, we can use '==' to determine the value we're looking for. For example, if we want to know the values only for "City Hotel", so the column 'hotel' should be written ***hotel=='City Hotel'***. The other condition is to know the value only for canceled reservations, so the column 'is_canceled' should be written ***is_canceled==1***.

To compute the percentage of canceled reservations, we can use formula ***100*(total canceled reservation/total reservation)***. To make it round, we can add ***round*** in the beginning and then *2* to make it 2 decimals at the end.

a) So, here is the code we can use to answer the question:

```
round(100*(len(df_hotels[(df_hotels.hotel=='City Hotel')&(df_hotels.is_cance
```

Here is the output of the code:

```
41.73
```

b) So, here is the code we can use to answer the question:

```
round(100*(len(df_hotels[(df_hotels.hotel=='Resort Hotel')&(df_hotels.is_can
```

Here is the output of the code:

```
27.76
```

c) Based on the values above, we can conclude that City Hotel has the bigger percentage of canceled reservations, with the value up to 41,73%.

> ***Question 4:***
> *Filter data so that it only displays the visitors who don't cancel the reservation and save the result in df_checkout variable!*

Almost the same as Question 3, for specific condition to only displays the visitors who don't cancel the reservation, we can use ***is_canceled==0*** and defined it as ***df_checkout*** variable that means it is only the visitors who successfully checked out the reservation (not cancel the reservation).

So, here is the code we can use:

```
df_checkout = df_hotels[df_hotels.is_canceled==0]
```

> ***Question 5:***

> *a) Show the number of reservations per month of arrival for each type of hotel!*
> *b) Then in which month there are the most reservations in each type of hotel? Make a conclusion whether the trend is the same in both types of hotels?*
> *c) Do as point B but with the name of the month that has been mapped into months in numbers!*
>
> ***note:** for this and subsequent questions will use the df_checkout dataframe.*

a) To show data based on the number of reservations per month of arrival and per type of hotel, we can use ***groupby()*** and write the grouping of column used inside the brackets, which are ***['hotel']*** and ***['arrival_date_month']***.

Then, to show the number of reservations itself, we can count the unique values of reservation id from column 'id' with ***['id'].nunique()***.

So, here is the code we can use:

```
df_checkout.groupby(['hotel','arrival_date_month'])['id'].nunique()
```

Here is the output of the code:

```
hotel         arrival_date_month
City Hotel    April               4015
              August              5381
              December            2392
              February            3064
              January             2254
              July                4782
              June                4366
              March               4072
              May                 4579
              November            2696
              October             4337
              September           4290
Resort Hotel  April               2550
              August              3257
              December            2017
              February            2308
              January             1868
              July                3137
              June                2038
              March               2573
              May                 2535
              November            1976
              October             2577
              September           2102
Name: id, dtype: int64
```

b) Based on the output above, we can roughly see that the most reservations in each type of hotel were occurring in the same month, which is August.

c) To change the name of the months into months in numbers, we can use this code:

```
import calendar

month_dict = {month: index for index, month in enumerate(calendar.month_name
month_dict
```

```
df_checkout['arrival_date_month_number'] = df_checkout['arrival_date_month']
```

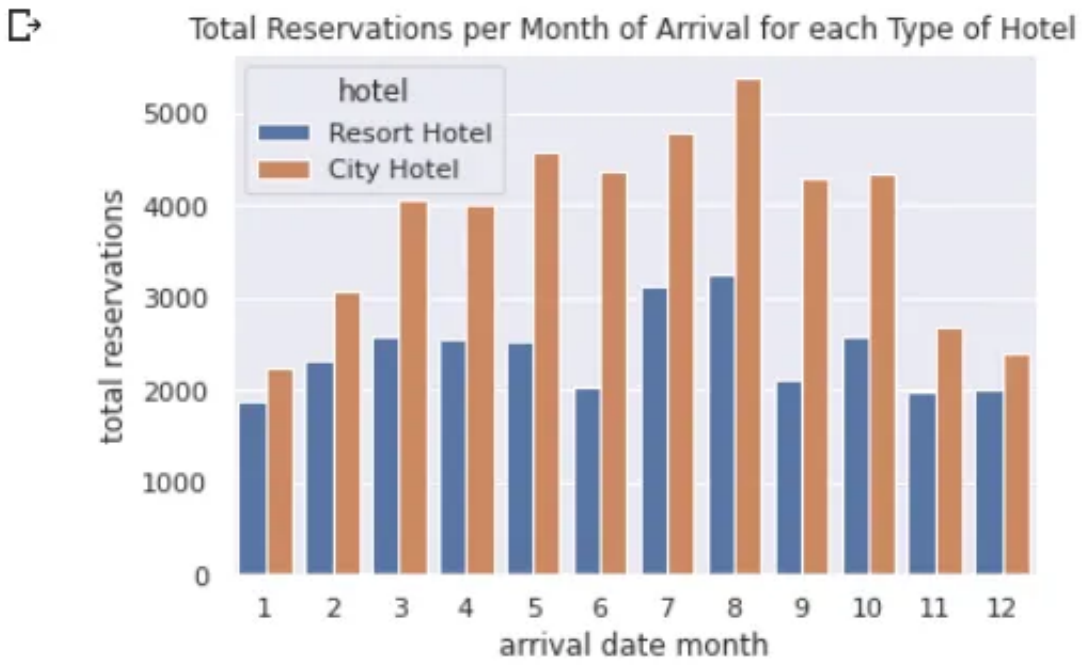
To show the output after we change the name of months into months in numbers, we use the same code as used in 5a. Then, here is the output we get:

```
hotel         arrival_date_month_number
City Hotel    1                           2254
              2                           3064
              3                           4072
              4                           4015
              5                           4579
              6                           4366
              7                           4782
              8                           5381
              9                           4290
              10                          4337
              11                          2696
              12                          2392
Resort Hotel  1                           1868
              2                           2308
              3                           2573
              4                           2550
              5                           2535
              6                           2038
              7                           3137
              8                           3257
              9                           2102
              10                          2577
              11                          1976
              12                          2017
Name: id, dtype: int64
```

For better way to show the number of reservations itself per month of arrival for each type of hotel, we can visualize the values above with the countplot. Here is the code we can use:

```
sns.countplot(data=df_checkout, x='arrival_date_month_number',hue='hotel')
plt.xlabel('arrival date month')
plt.ylabel('total reservations')
plt.title('Total Reservations per Month of Arrival for each Type of Hotel')
plt.show()
```

Here is the output of the code:



Based on graph above, we can strengthen the statement in 5b that both type of hotel (City Hotel and Resort Hotel) has the most reservations in August. They also have the same trend pattern.

> ***Question 6:***
> *a) Create a new column named arrival_date which contains complete information about the year, month, and date of arrival!*
> *b) Change the column to datetime type!*

a) To combine the value of year, month, and date of arrival, we have to change the data type from datetime to string first, so we can add or combine one value to another. We can use ***astype()*** to change the datatype and add ***'str'*** inside the brackets to change datatype into string. We change all the three of year, month, and date of arrival datatype into string.

In month and date of arrival we want to add '0' if there is only single digit so it turns into 2 digits of number. We can use ***str.pad(2,fillchar='0')***.

After that, we can combine all three values of year, month, and date of arrival into one value with adding ' — ' as connector.

Lastly, we defined that combined value in a new column named 'arrival_date'.

So, here is the code we can use:

```
df_checkout['arrival_date'] = \
  df_checkout['arrival_date_year'].astype('str') + '-' +\
  df_checkout.arrival_date_month_number.astype('str').str.pad(2,fillchar='0'
  df_checkout.arrival_date_day_of_month.astype('str').str.pad(2,fillchar='0'
```

The backslash symbol (\ ) at the end of line is used to extend the current logical line over across to the next physical line.

To check the values of 'arrival_date' column we just made, we can use this

code:

```
df_checkout['arrival_date']
```

Here is the output of the code:

```
0         2015-07-01
1         2015-07-01
2         2015-07-01
3         2015-07-01
4         2015-07-01
              ...
119385    2017-08-30
119386    2017-08-31
119387    2017-08-31
119388    2017-08-31
119389    2017-08-29
Name: arrival_date, Length: 75166, dtype: object
```

b) As we know that the datatype of arrival_date is string, so we want to change it back to datetime by using this code:

```
df_checkout['arrival_date'] = pd.to_datetime(df_checkout.arrival_date)
df_checkout['arrival_date']
```

> *Question 7:*
> *Create two dataframe containing:*
> *a) Daily reservation (df_daily_reservation)*

b) Average of daily reservation per week (df_avg_daily_reservation_per_week)

a) First, we define the ***df_daily_reservation*** as the name of dataframe in the beginning of the code. Then, to create dataframe containing daily reservation, we take data from 'df_checkout' and count the reservations using ***size()*** based on how many it appears in each 'arrival_date'. Then we can rename the column name using ***rename()***. The full code is shown below:

```
df_daily_reservation = df_checkout.resample('D', on='arrival_date').size().r
df_daily_reservation
```

Here is the output of the code:

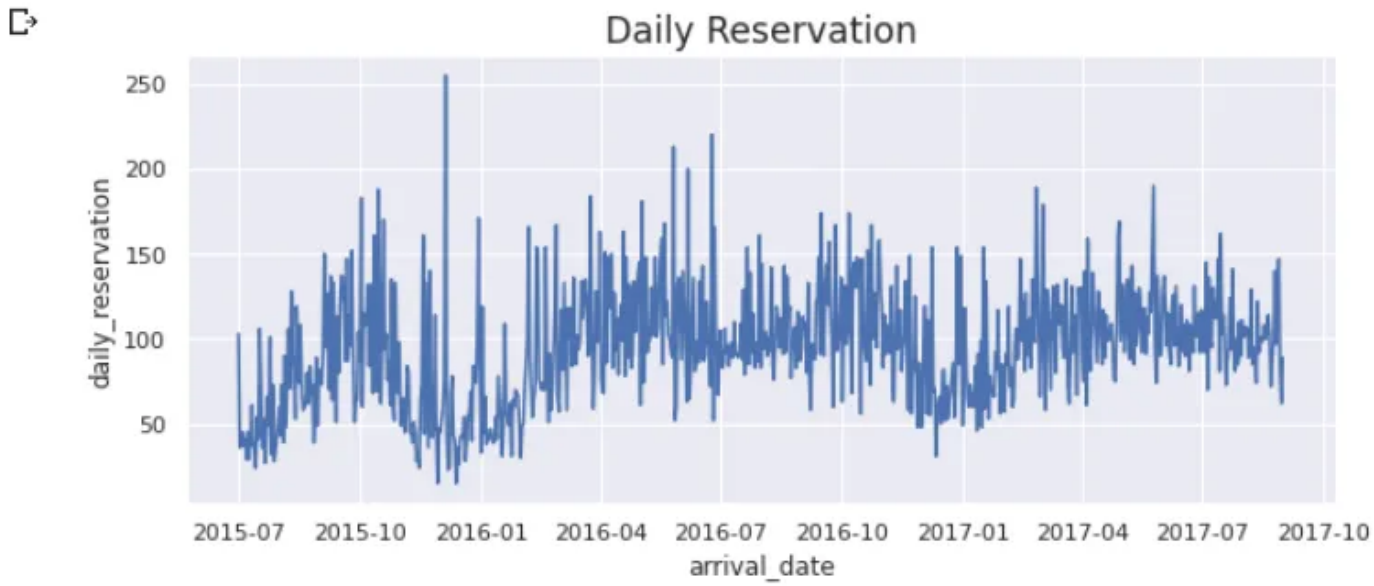
	arrival_date	daily_reservation
0	2015-07-01	103
1	2015-07-02	36
2	2015-07-03	37
3	2015-07-04	45
4	2015-07-05	37
...	...	...
788	2017-08-27	125
789	2017-08-28	147
790	2017-08-29	81
791	2017-08-30	62
792	2017-08-31	89

793 rows × 2 columns

For better way to show the number of daily reservations, we can visualize it with the lineplot. Here is the code we can use:

```
plt.figure(figsize=(10,4))
sns.lineplot(data=df_daily_reservation, x='arrival_date', y='daily_reservati
plt.title('Daily Reservation', fontsize='x-large')
plt.show()
```

Here is the output of the code:



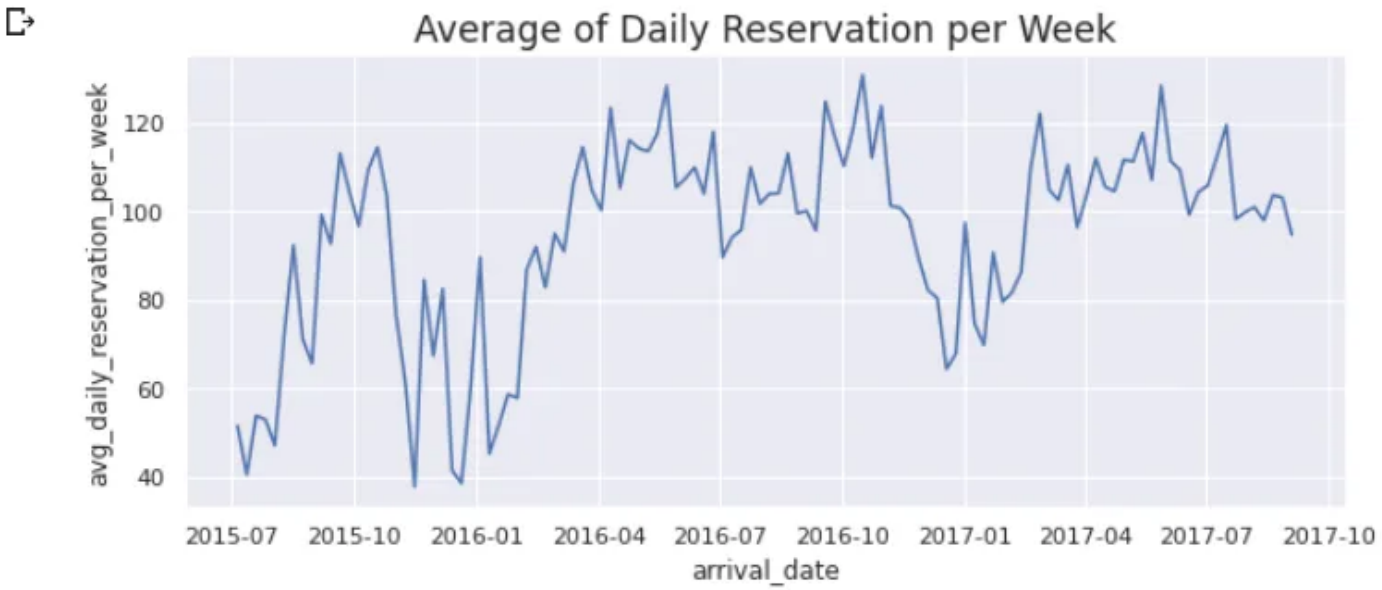
b) First, we define the ***df_avg_daily_reservation_per_week*** as the name of dataframe in the beginning of the code. Then, to create dataframe containing average of daily reservation per week, we take data from 'df_checkout' and count the average using ***mean()*** based on average of how many it appears in each 'arrival_date' per week. We can also add ***round*** in the beginning and then *2* to make it 2 decimals at the end. Then we can rename the column name using ***rename()***. The full code is shown below:

```
df_avg_daily_reservation_per_week = round(df_checkout.resample('D', on='arri
df_avg_daily_reservation_per_week
```

For better way to show the number of average of daily reservations per week, we can visualize it with the lineplot. Here is the code we can use:

```
plt.figure(figsize=(10,4))
sns.lineplot(data=df_avg_daily_reservation_per_week, x='arrival_date', y='avg
plt.title('Average of Daily Reservation per Week', fontsize='x-large')
plt.show()
```

Here is the output of the code:



> ***Question 8:***
> *a) What is the average ADR (Average Daily Rate) based on hotel type and customer type?*
> *b) Which type of customer has the highest the average of ADR in each type of hotel?*

a) Almost same with Question 5, here we use ***group()***. To show data based on the hotel type and customer type, we can use ***groupby()*** and write the grouping of column used inside the brackets, which are ***['hotel']*** and ***['customer_type']***.

Then, to show the mean of ADR, we can use ***mean()*** of the value in 'adr' column written as ***['adr'].mean()***. We can also add ***round*** in the beginning and then *2* to make it 2 decimals at the end.

So, here is the code we can use:

```
round(df_checkout.groupby(['hotel','customer_type'])['adr'].mean(),2)
```

Here is the output of the code:

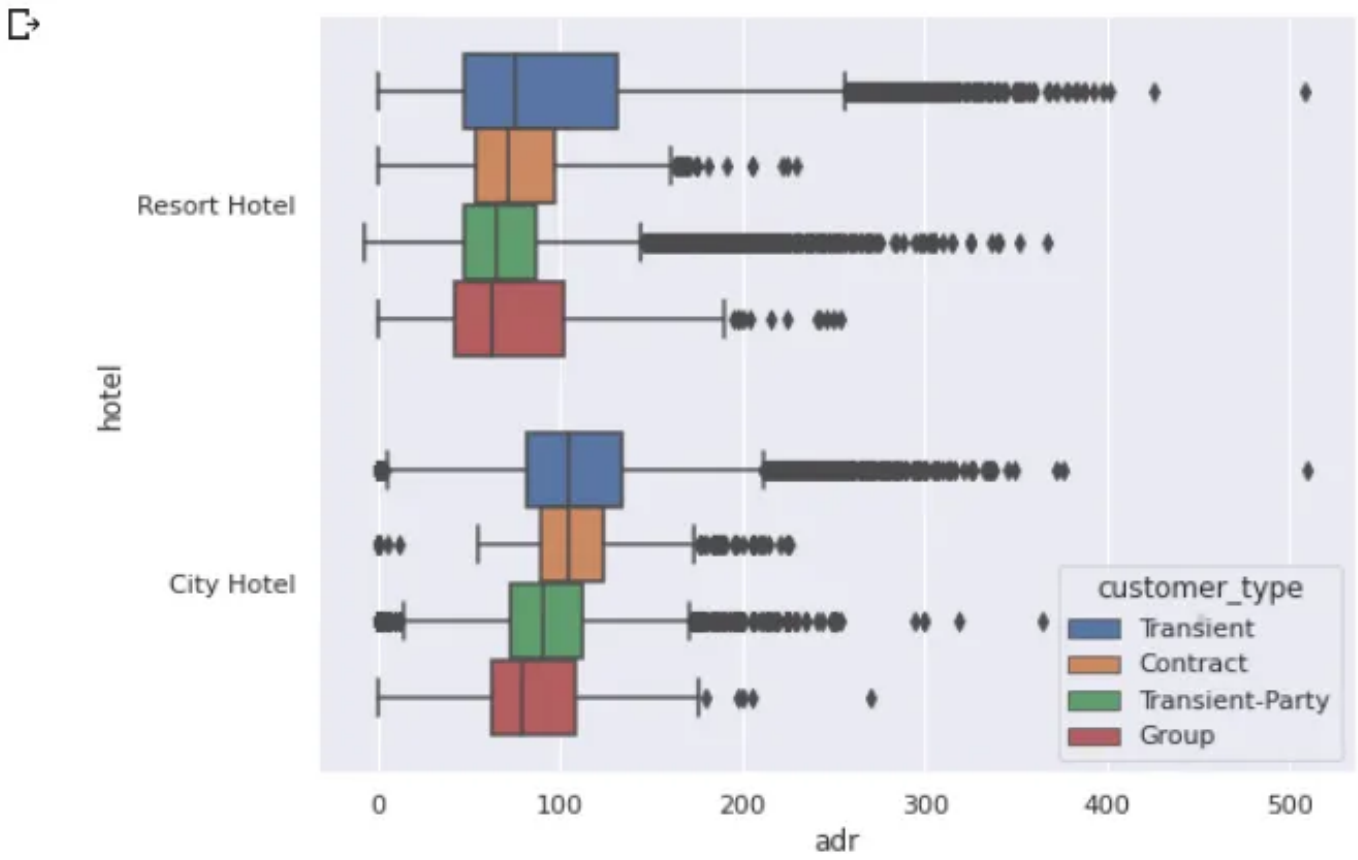
```
hotel         customer_type
City Hotel    Contract           108.93
              Group               87.40
              Transient          110.42
              Transient-Party     93.71
Resort Hotel  Contract            78.58
              Group               77.31
              Transient           96.00
              Transient-Party     77.20
Name: adr, dtype: float64
```

b) Based on mean value above, we can see that the type of customer has the highest average of ADR in each type of hotel is Transient.

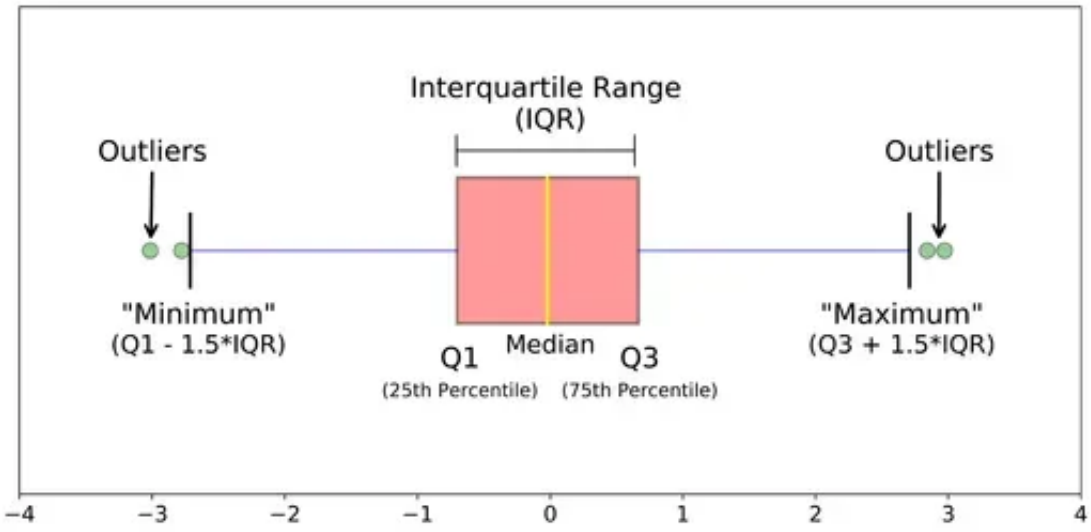
For better way to show the full distribution of the data for each customer type of the two hotel types, exclude mean, we can visualize with boxplot. Here is the code we can use:

```
plt.figure(figsize=(8,6))
sns.boxplot(data=df_checkout, x='adr', y='hotel',hue='customer_type')
plt.show()
```

Here is the output of the code:



For the information of how to read boxplot shown below:



source: https://www.simplypsychology.org/

> *Question 9:*
> *By using the 'df_country' dataframe which contains the country name and country code information, show the 10 countries with the largest number of reservations!*

First, we're going to add Countries Codes and Coordinates.csv Dataset from the URL link using this code:

```
df_country = pd.read_csv('https://gist.githubusercontent.com/tadast/8827699/
```

Then, we adjust the value of three alpha code of country name in the 'df_country' so it will be the same as the value of three alpha code of country name in the 'df_checkout'. We can erase the double quotation mark (" ") of the value in the 'Alpha-3 code' column and then define as 'code' column. Here is the code we can use:

```
df_country['code'] = df_country['Alpha-3 code'].str.replace('"','').str.stri
```

After that, we can merge ‘df_checkout’ with ‘df_country’ and define as ‘df_merged’ variable. We can use this code:

```
df_merged = pd.merge(df_checkout[['id','country']],
                     df_country[['Country','code']],
                     left_on='country',
                     right_on='code',
                     indicator=True,
                     how='left')
df_merged
```

So, ‘df_merged’ looks like this:

	id	country	Country	code	_merge
0	0	PRT	Portugal	PRT	both
1	1	PRT	Portugal	PRT	both
2	2	GBR	United Kingdom	GBR	both
3	3	GBR	United Kingdom	GBR	both
4	4	GBR	United Kingdom	GBR	both
...	...	...	...	...	...
75712	119385	BEL	Belgium	BEL	both
75713	119386	FRA	France	FRA	both
75714	119387	DEU	Germany	DEU	both
75715	119388	GBR	United Kingdom	GBR	both
75716	119389	DEU	Germany	DEU	both

75717 rows × 5 columns

To show the 10 countries with the largest number of reservations, we can use ***value_counts().head(10)*** of the 'Country' column in 'df_merged'. Here is the code we can use:

```
df_merged.Country.value_counts().head(10)
```

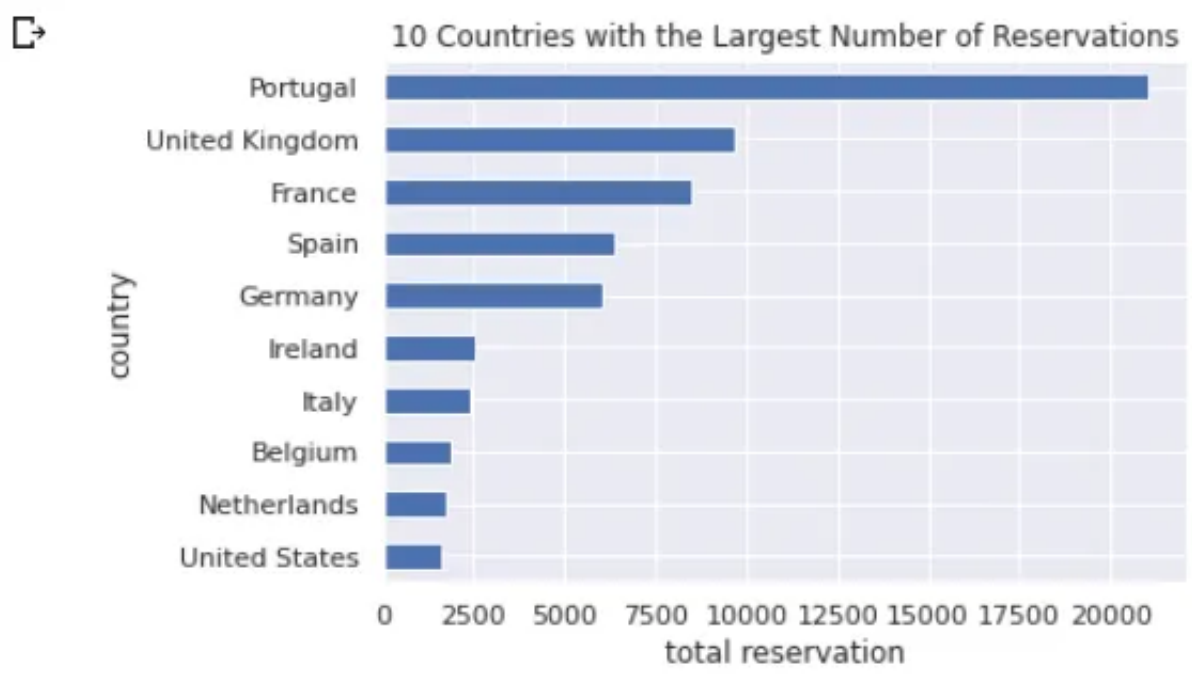
Here is the output of the code:

```
Portugal           21071
United Kingdom      9676
France              8481
Spain               6391
Germany             6069
Ireland             2543
Italy               2433
Belgium             1868
Netherlands         1717
United States       1596
Name: Country, dtype: int64
```

For better way to show top 10 countries with the largest number of reservations, we can visualize it with horizontal bar chart ***plot.barh()***. Here is the code we can use:

```
df_merged.Country.value_counts().head(10).sort_values(ascending=True).plot.ba
plt.xlabel('total reservation')
plt.ylabel('country')
plt.title('10 Countries with the Largest Number of Reservations')
plt.show()
```

Here is the output of the code:



> *Question 10:*
> *a) How many average guests stay for each reservation?*
> *b) Based on the dataset, what is the highest number of guests? Also show the reservation data row that has the highest number of guests.*

a) There is no column yet that shows total guests in each reservation, so we create new column first. Total guests can be counted by summing up the value in column 'adults', 'children' and 'babies'. So, here is the code we can use to create new column named 'total_guest':

```
df_checkout['total_guest'] = df_checkout.adults + df_checkout.children + df_
```

Then, we can count the average guests stay for each reservation using ***mean()***. We can also add ***round*** in the beginning and then ***0*** to make it

round number because the data is number of people. Here is the code we can use:

```
round(df_checkout.total_guest.mean(),0)
```

Here is the output of the code:

```
2.0
```

So, the average guests stay for each reservation is 2 people.

b) To find the highest number of guests stay in a reservation, we can use *max()*. Here is the code we can use:

```
df_checkout.total_guest.max()
```

Here is the output of the code:

```
12.0
```

We can also show the reservation data row which has the highest number

of guests. At the end of code, we use ***.T*** to transpose the form of one data row into one data column for easy reading. Here is the code we can use:

```
df_checkout[df_checkout.total_guest==df_checkout.total_guest.max()].T
```

Here is the output of the code:

id	46619
hotel	City Hotel
is_canceled	0
lead_time	37
arrival_date_year	2016
arrival_date_month	January
arrival_date_week_number	3
arrival_date_day_of_month	12
stays_in_weekend_nights	0
stays_in_week_nights	2
adults	2
children	0.0
babies	10
meal	BB
country	PRT
market_segment	Online TA
distribution_channel	TA/TO
is_repeated_guest	0
previous_cancellations	0
previous_bookings_not_canceled	0
reserved_room_type	D
assigned_room_type	D
booking_changes	1
deposit_type	No Deposit