Content in this Notebook

Assignment 4: Exploratory Data Analysis

REGex Software Services (https://www.linkedin.com/company/regexsoftware/

EDA on European hotel reviews data set from

kaggle

back to top

2. Importing Libraries

In [45]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

plt.rcParams['figure.figsize'] = (16, 8)
plt.style.use('fivethirtyeight')

import warnings
warnings.filterwarnings('ignore')
```

back to top

3. Loading and Reading Data

In [46]:
```
df = pd.read_csv('Europe Hotel Booking Satisfaction Score.csv')
display(df.head())
print('Shape of the Data: ', (df.shape))
```

	id	Gender	Age	purpose_of_travel	Type of Travel	Type Of Booking	Hotel wifi service	Departure/Arrival convenience	Ease of Online booking
0	70172	Male	13	aviation	Personal Travel	Not defined	3	4	3
1	5047	Male	25	tourism	Group Travel	Group bookings	3	2	3
2	110028	Female	26	tourism	Group Travel	Group bookings	2	2	2
3	24026	Female	25	tourism	Group Travel	Group bookings	2	5	5
4	119299	Male	61	aviation	Group Travel	Group bookings	3	3	3

```
Shape of the Data:  (103904, 17)
```

back to top

4. Data Description and initial cleaning

In [47]:
```
df.describe()
```

Out[47]:

	id	Age	Hotel wifi service	Departure/Arrival convenience	Ease of Online booking	Hotel locatic
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103902.00000
mean	64924.210502	39.379706	2.729683	3.060296	2.756901	2.97690
std	37463.812252	15.114964	1.327829	1.525075	1.398929	1.2776
min	1.000000	7.000000	0.000000	0.000000	0.000000	0.00000
25%	32533.750000	27.000000	2.000000	2.000000	2.000000	2.00000
50%	64856.500000	40.000000	3.000000	3.000000	3.000000	3.00000
75%	97368.250000	51.000000	4.000000	4.000000	4.000000	4.00000
max	129880.000000	85.000000	5.000000	5.000000	5.000000	5.00000

back to top

4.1 Cleaning Strings in Column and Values

In [48]:
```
df.columns = df.columns.str.lower().str.replace(" ", "_")
df.head()
```

Out[48]:

	id	gender	age	purpose_of_travel	type_of_travel	type_of_booking	hotel_wifi_service	dep
0	70172	Male	13	aviation	Personal Travel	Not defined	3	
1	5047	Male	25	tourism	Group Travel	Group bookings	3	
2	110028	Female	26	tourism	Group Travel	Group bookings	2	
3	24026	Female	25	tourism	Group Travel	Group bookings	2	
4	119299	Male	61	aviation	Group Travel	Group bookings	3	

back to top

4.2 Making List of Categorical Columns

In [49]:
```
"""
Making List of Categorical Columns
"""
#categorical = list(df.dtypes[df.dtypes == 'object'].index)
categorical = list([col for col in df.columns if df[col].dtype == 'object'])
categorical
# Other methods
# list(df.select_dtypes(include = 'O').columns)
# [col for col in df.columns if df[col].dtype == 'object']
```

Out[49]:
```
['gender',
 'purpose_of_travel',
 'type_of_travel',
 'type_of_booking',
 'satisfaction']
```

back to top

4.3 Cleaning Categorical Data in our data set

In [50]:
```
"""
Cleaning Categorical Data in our data set
"""
for col in categorical:
    df[col] = df[col].str.lower().str.replace(" ", "_")

df.head()
```

Out[50]:

	id	gender	age	purpose_of_travel	type_of_travel	type_of_booking	hotel_wifi_service	dep
0	70172	male	13	aviation	personal_travel	not_defined	3	
1	5047	male	25	tourism	group_travel	group_bookings	3	
2	110028	female	26	tourism	group_travel	group_bookings	2	
3	24026	female	25	tourism	group_travel	group_bookings	2	
4	119299	male	61	aviation	group_travel	group_bookings	3	

Unique values and numbers

In [51]:
```
"""Printing out First 5 Unique Values."""
for col in df.columns:
    print(col)
    print(df[col].unique()[:5])
    print(df[col].nunique())
    print('\n')
```

```
id
[ 70172   5047 110028  24026 119299]
103904


gender
['male' 'female']
2


age
[13 25 26 61 47]
75


purpose_of_travel
['aviation' 'tourism' 'business' 'academic' 'personal']
5


type_of_travel
['personal_travel' 'group_travel']
2


type_of_booking
['not_defined' 'group_bookings' 'individual/couple']
3


hotel_wifi_service
[3 2 4 1 5]
6


departure/arrival__convenience
[4 2 5 3 1]
6


ease_of_online_booking
[3 2 5 4 1]
6


hotel_location
[1. 3. 2. 5. 4.]
6
```

```
food_and_drink
[5 1 2 4 3]
6


stay_comfort
[5. 1. 2. 3. 4.]
6


common_room_entertainment
[5 1 2 3 4]
6


checkin/checkout_service
[4. 1. 3. 5. 2.]
6


other_service
[5 4 3 1 2]
6


cleanliness
[5 1 2 3 4]
6


satisfaction
['neutral_or_dissatisfied' 'satisfied' nan]
2
```

back to top

5. Missing Values

In [52]:
```
print("Number of Missing Values in our data set\n")
missing_df = df.isnull().sum().to_frame().reset_index().rename({"index" : 'Variab
display(missing_df.style.background_gradient('gnuplot2_r'))
print("\n Percentage of Missing Values in our data set")
display((df.isnull().sum() / (len(df.index)) * 100).head(20).to_frame().rename({0
round((df.isnull().sum() / (len(df.index)) * 100) , 2).plot(kind = 'barh',color =

plt.title("Percentage of Missing values");
```

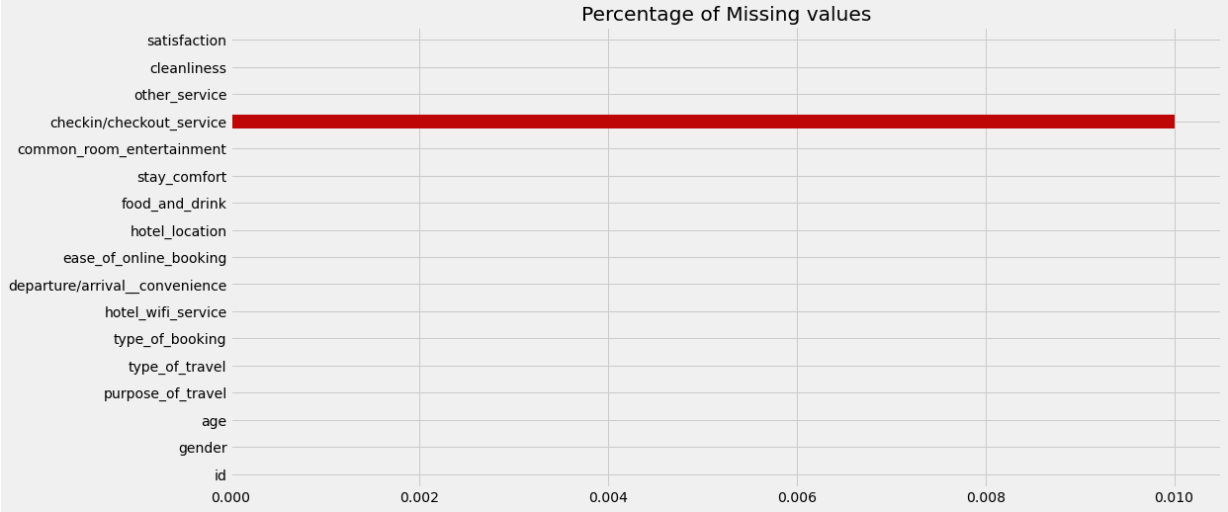
Number of Missing Values in our data set

	Variable	Missing Values
0	id	0
1	gender	0
2	age	0
3	purpose_of_travel	0
4	type_of_travel	0
5	type_of_booking	0
6	hotel_wifi_service	0
7	departure/arrival__convenience	0
8	ease_of_online_booking	0
9	hotel_location	2
10	food_and_drink	0
11	stay_comfort	1
12	common_room_entertainment	0
13	checkin/checkout_service	6
14	other_service	0
15	cleanliness	0
16	satisfaction	2

Percentage of Missing Values in our data set

	Count
id	0.000000
gender	0.000000
age	0.000000
purpose_of_travel	0.000000
type_of_travel	0.000000
type_of_booking	0.000000
hotel_wifi_service	0.000000

	Count
departure/arrival__convenience	0.000000
ease_of_online_booking	0.000000
hotel_location	0.001925
food_and_drink	0.000000
stay_comfort	0.000962
common_room_entertainment	0.000000
checkin/checkout_service	0.005775
other_service	0.000000
cleanliness	0.000000
satisfaction	0.001925



Observarion

- Above tables and plot shows frequency and percentage of missing values in our data set.
- Variable market_category has the highest missing values and engine_fuel_type has the least missing values count.

- I will use mode to compute missing values in categorical columns and mean to compute missing values in numerical columns.

back to top

5.1 Treating Missing Values

In [53]:
```
#We will use Mode to fill up missing values in Categorical columns"""
df['satisfaction'].fillna(df['satisfaction'].mode()[0], inplace = True)


#We will use mean to fill up missing values in Numerical columns"""
#df['engine_hp'].fillna(df['engine_hp'].mean(), inplace = True)

#We will use median to fill up missing values in Ordinal Numerical columns"""
df['hotel_location'].fillna(df['hotel_location'].median(), inplace = True)
df['stay_comfort'].fillna(df['stay_comfort'].median(), inplace = True)
df['checkin/checkout_service'].fillna(df['checkin/checkout_service'].median(), ir

#Checking Missing Values after imputing """
display(df.isnull().sum().to_frame().reset_index().rename({'index' : 'Variables',
```

	Variables	Missing Values
0	id	0
1	gender	0
2	age	0
3	purpose_of_travel	0
4	type_of_travel	0
5	type_of_booking	0
6	hotel_wifi_service	0
7	departure/arrival__convenience	0
8	ease_of_online_booking	0
9	hotel_location	0
10	food_and_drink	0
11	stay_comfort	0
12	common_room_entertainment	0
13	checkin/checkout_service	0
14	other_service	0
15	cleanliness	0
16	satisfaction	0

6 checking duplicates

In [54]:
```
#Creating for duplicates
display("Total number of of Duplicates present in data: %s"%df.duplicated().sum()
```

```
'Total number of of Duplicates present in data: 0'
```

6.1 Dropping the duplicates

In [55]:
```
#Dropping the Duplicates"""
df.drop_duplicates(inplace = True)

#Checking the Duplicates again"""
print("Total number of of Duplicates present in data: %s"%df.duplicated().sum())
```

```
Total number of of Duplicates present in data: 0
```

Observarion

- We had total 0 duplicated values.

back to top

7. Checking Correlation

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

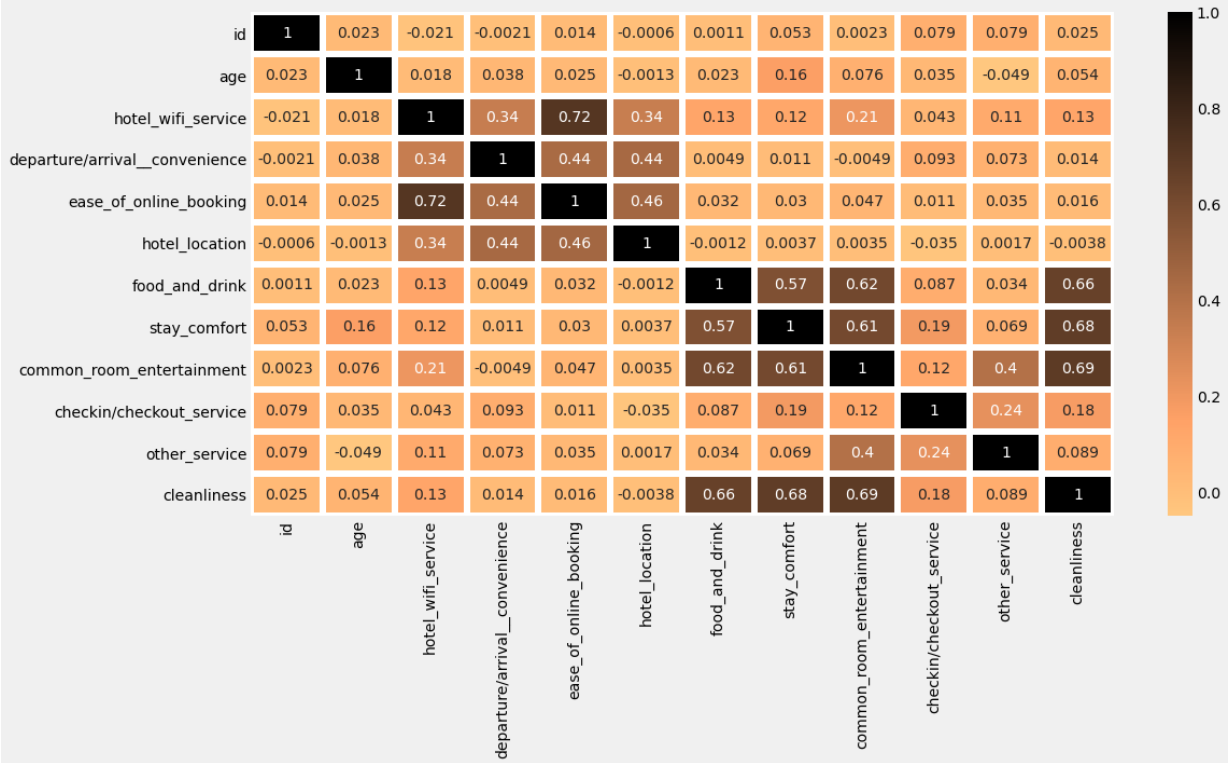
$\bar{y}$ = mean of the values of the y-variable

In [56]:
```
#Correlation Matrix of Data"""
df.corr().style.background_gradient('copper_r')
```

Out[56]:

	id	age	hotel_wifi_service	departure/arrival__convenienc
id	1.000000	0.022857	-0.021276	-0.0021
age	0.022857	1.000000	0.017859	0.03812
hotel_wifi_service	-0.021276	0.017859	1.000000	0.34384
departure/arrival__convenience	-0.002110	0.038125	0.343845	1.00000
ease_of_online_booking	0.014163	0.024842	0.715856	0.43696
hotel_location	-0.000603	-0.001341	0.336244	0.44474
food_and_drink	0.001063	0.023000	0.134718	0.00490
stay_comfort	0.052902	0.160270	0.122659	0.01135
common_room_entertainment	0.002300	0.076444	0.209321	-0.00486
checkin/checkout_service	0.079291	0.035472	0.043191	0.09328
other_service	0.079346	-0.049427	0.110441	0.07331
cleanliness	0.024965	0.053611	0.132698	0.01429

In [57]:
```
sns.heatmap(df.corr(), cmap = 'copper_r', annot = True, lw = 5);
```



Observarion

- stay_comfort is the dependent variable.

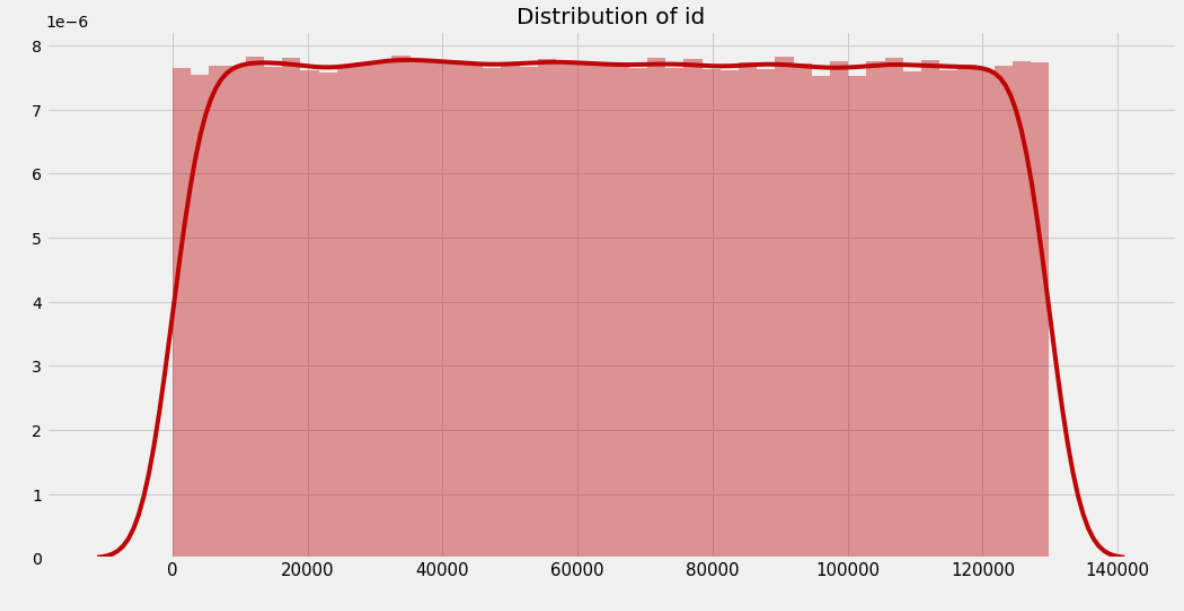
- Variables cleanliness, common_room_entertainment and food_and_drink has the highest correlation with dependent variable.

back to top

8. Checking Relation between all variables

8.1 Distribution and relationship of Numerical Variables with dependent variable

In [58]:
```
numerical = [col for col in df.columns if col not in categorical]
for i in numerical:
    ax = sns.distplot(df[i], color = '#bf0606')
    plt.title("Distribution of %s" %i, fontsize = 20)
    plt.xlabel(" ")
    plt.ylabel(" ")
    plt.xticks(fontsize = 15)
    plt.show();
    print('\n')
```



In [59]:
```
categorical,numerical
```

Out[59]:
```
(['gender',
  'purpose_of_travel',
  'type_of_travel',
  'type_of_booking',
  'satisfaction'],
 ['id',
  'age',
  'hotel_wifi_service',
  'departure/arrival__convenience',
  'ease_of_online_booking',
  'hotel_location',
  'food_and_drink',
  'stay_comfort',
  'common_room_entertainment',
  'checkin/checkout_service',
  'other_service',
  'cleanliness'])
```

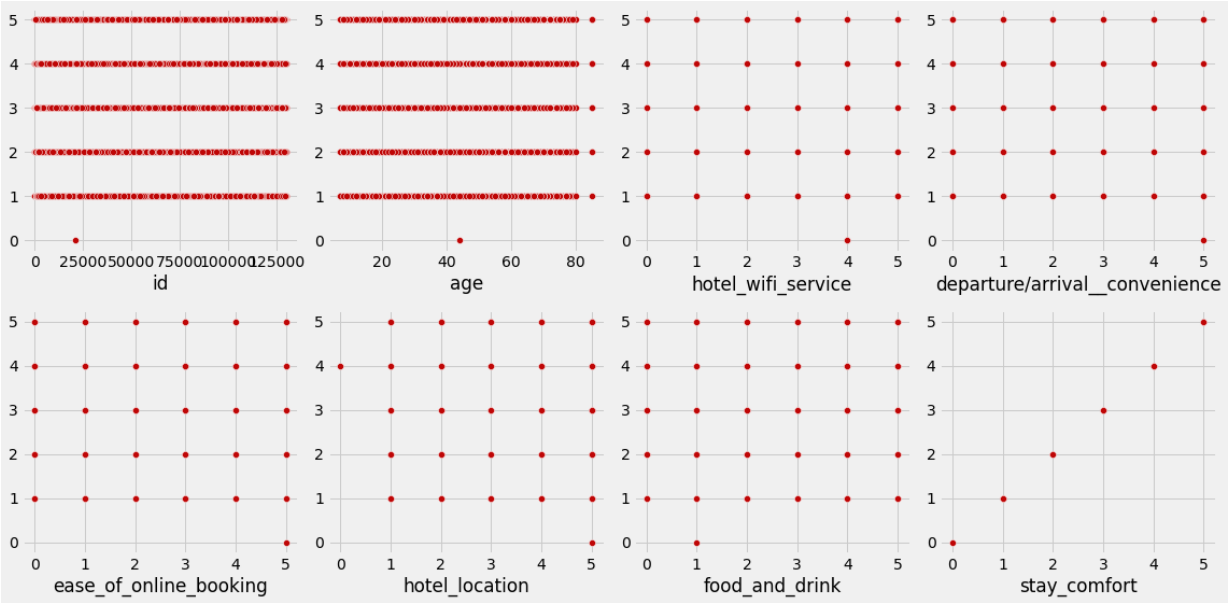
Observarion

- Numerical column distribution shows that some variables has skewed data: msrp, highway_mpg, year. Log transformation of data may be useful before using the data for prediction.
- There are certain numerical variables which are ordinal in nature e.g. number_of_doors and engine_cylinders. They can be converted into categorical columns and then transformed for the prediction by using One Hot Encoding etc.
- Variable popularity shows multimodal distribution.

back to top

8.1.1 Distribution and relationship of Numerical Variables with dependent variable

In [60]:
```
nrows = 2
ncols = 4
i = 0
fig, ax = plt.subplots(nrows, ncols, figsize = (16,8),)
for row in range(nrows):
    for col in range(ncols):
        sns.scatterplot(df[numerical].iloc[:, i],df['stay_comfort'],ax = ax[row,
        plt.tight_layout()
        i += 1
```



Observation: every numerical variable distributed perfectly

back to top

8.2. Checking Relation between all categorical variables and dependent variables (msrp)

In [61]:
```
#Creating a function for ordering the groups in a column as per their frequency"'
def sort_order(column):
    orders = (df.groupby([column]).mean().sort_values(by ='stay_comfort', ascendi
    return orders
```

In [62]:
```
#Looping over categorical variables to check the hospitality rating over differer
for i in categorical:
    if df[i].nunique() < 10:
        sns.barplot(df[i],df['stay_comfort'], order = sort_order(i), palette='cop
        plt.title("Bar Plot of %s" %i, fontsize = 20)
        plt.xticks(fontsize = 12)
        plt.xlabel("%s"%i)
        plt.ylabel("Hospitality Rating in restaurent")
        plt.xticks(fontsize = 15, rotation = 90)
        plt.show();
        print('\n')
```