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

In [2]:
```
pd = pd.read_csv('train_cancellation.csv')
pd.head()
```

Out[2]:

	id	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_type_reser
0	0	2	0	0	2	1	0	
1	1	2	0	1	2	0	0	
2	2	2	0	0	1	0	0	
3	3	1	0	0	2	1	0	
4	4	2	0	1	0	0	0	

In [3]:
```
pd.tail()
```

Out[3]:

	id	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_ty
42095	42095	3	0	0	4	0	0	
42096	42096	2	0	0	3	0	0	
42097	42097	2	0	0	2	2	0	
42098	42098	1	0	0	3	0	0	
42099	42099	2	0	1	1	0	0	

In [4]:
```
pd.isnull().sum()
```

Out[4]:
```
id                                    0
no_of_adults                          0
no_of_children                        0
no_of_weekend_nights                  0
no_of_week_nights                     0
type_of_meal_plan                     0
required_car_parking_space            0
room_type_reserved                    0
lead_time                             0
arrival_year                          0
arrival_month                         0
arrival_date                          0
market_segment_type                   0
repeated_guest                        0
no_of_previous_cancellations          0
no_of_previous_bookings_not_canceled  0
avg_price_per_room                    0
no_of_special_requests                0
booking_status                        0
dtype: int64
```

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

Out[5]:
```
(42100, 19)
```

In [6]:
```
pd.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
pd.columns
```

Out[7]:
```
Index(['id', 'no_of_adults', 'no_of_children', 'no_of_weekend_nights',
       'no_of_week_nights', 'type_of_meal_plan', 'required_car_parking_space',
       'room_type_reserved', 'lead_time', 'arrival_year', 'arrival_month',
       'arrival_date', 'market_segment_type', 'repeated_guest',
       'no_of_previous_cancellations', 'no_of_previous_bookings_not_canceled',
       'avg_price_per_room', 'no_of_special_requests', 'booking_status'],
      dtype='object')
```

In [8]:
```
pd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42100 entries, 0 to 42099
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   id                                    42100 non-null  int64
 1   no_of_adults                          42100 non-null  int64
 2   no_of_children                        42100 non-null  int64
 3   no_of_weekend_nights                  42100 non-null  int64
 4   no_of_week_nights                     42100 non-null  int64
 5   type_of_meal_plan                     42100 non-null  int64
 6   required_car_parking_space            42100 non-null  int64
 7   room_type_reserved                    42100 non-null  int64
 8   lead_time                             42100 non-null  int64
 9   arrival_year                          42100 non-null  int64
 10  arrival_month                         42100 non-null  int64
 11  arrival_date                          42100 non-null  int64
 12  market_segment_type                   42100 non-null  int64
 13  repeated_guest                        42100 non-null  int64
 14  no_of_previous_cancellations          42100 non-null  int64
 15  no_of_previous_bookings_not_canceled  42100 non-null  int64
 16  avg_price_per_room                    42100 non-null  float64
 17  no_of_special_requests                42100 non-null  int64
 18  booking_status                        42100 non-null  int64
dtypes: float64(1), int64(18)
memory usage: 6.1 MB
```

In [9]:
```
pd.describe()
```

Out[9]:

	id	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space
count	42100.000000	42100.000000	42100.000000	42100.000000	42100.000000	42100.000000	42100.000000
mean	21049.500000	1.920713	0.141093	0.884632	2.398005	0.239192	0.025249
std	12153.367503	0.524950	0.450128	0.885693	1.427330	0.587674	0.156884
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	10524.750000	2.000000	0.000000	0.000000	1.000000	0.000000	0.000000
50%	21049.500000	2.000000	0.000000	1.000000	2.000000	0.000000	0.000000
75%	31574.250000	2.000000	0.000000	2.000000	3.000000	0.000000	0.000000
max	42099.000000	4.000000	9.000000	7.000000	17.000000	3.000000	1.000000

In [10]:
```
pd.nunique()
```

Out[10]:

```
id                                    42100
no_of_adults                              5
no_of_children                            6
no_of_weekend_nights                      8
no_of_week_nights                        18
type_of_meal_plan                         4
required_car_parking_space                2
room_type_reserved                        7
lead_time                               338
arrival_year                              2
arrival_month                            12
arrival_date                             31
market_segment_type                       5
repeated_guest                            2
no_of_previous_cancellations             10
no_of_previous_bookings_not_canceled     42
avg_price_per_room                     2286
no_of_special_requests                    6
booking_status                            2
dtype: int64
```

In [11]:
```
df = pd[['no_of_adults', 'no_of_children', 'no_of_weekend_nights',
       'no_of_week_nights', 'type_of_meal_plan', 'required_car_parking_space',
       'room_type_reserved', 'lead_time', 'arrival_year', 'arrival_month',
       'arrival_date', 'market_segment_type', 'repeated_guest',
       'no_of_previous_cancellations', 'no_of_previous_bookings_not_canceled',
       'avg_price_per_room', 'no_of_special_requests', 'booking_status']]
```

In [12]:
```
for i in df.columns:
    print(i)
    print(df[i].unique())
```

```
no_of_adults
[2 1 0 3 4]
no_of_children
[0 2 1 3 4 9]
no_of_weekend_nights
[0 1 2 3 5 6 4 7]
no_of_week_nights
[ 2  1  0  5  3  4  7  6  8 10 11  9 12 15 17 13 16 14]
type_of_meal_plan
[1 0 2 3]
required_car_parking_space
[0 1]
room_type_reserved
[0 2 3 1 4 5 6]
lead_time
[  9 117 315  32 258 215 320 265 189 137   1  69  50 217  58 143 188 155
 109  84   7   0 184  11 224 182 179 111  49  61 317  66  41 139 207 161
  25 147 110 208 148 223  47 275  40 129  34  94 221  51  59  95 133  12
 105 166   3 152  23  36 128  55 280  63  93  56  71 113  42 156   8  57
 160 118  30  10 102 132 168 198 138 107  20 159 150   6 134 172  77  33
  35 119 121 237  28 174 151 252   5 157 131 123 418 100  80 202   2  39
  27 122  67 196  16 245  68   4 335 178 219  14 116 177 153 175  54  75
  97 185  78 124 127 250  99  15 263  65  26  19  70 230  64 130  73 193
  46 183 180  22 164 256 162 141  43 101  37 273  76 200 146  87 181  24
 103 167 126 142 197  52 115 136  48  98  44  86 140 154 349  60 112 206
  53 106 171  91  13  89 259 192  83  45 292 163 203  74 186 377 222 187
 433  72  21  90  92 169 226 253 286 205 149  31 308  81 220  88 170 125
 104 120 244  82 216 194 158  96 190 240 285 291 322  62 268 213 346 108
 191 229 204  29 338  79  85  38 277 144 260 114 271 173 290 443 199 218
 267 209 195 276 247 279 211 309 228 135 145 282 243 302 212  18 254 301
 232 242 255 296 225  17 235 210 227 269 305 359 303 262 274 386 278 298
 281 264 266 238 272 165 304 176 249 214 270 257 289 314 239 287 323 236
 233 299 336 248 325 246 319 231 251 261 332 288 294 201 293 283 307 381
 234 327 313 300 361 331 310 241 330 311 345 326 348 353]
arrival_year
[2018 2017]
arrival_month
[ 1  7 12 10  8  6  9  4 11  5  3  2]
arrival_date
[14 29  2  1 16  6 18 24  8 22 27 12 30 26  3 25 28 21  4 11 15 20  5 10
  7 19  9 23 31 17 13]
market_segment_type
[1 0 2 3 4]
repeated_guest
[1 0]
no_of_previous_cancellations
[11  0  1  4  2  3  8 13  6  5]
no_of_previous_bookings_not_canceled
[ 0  2  3 23 47  7  8 14  4  6 13  1 12  5 33 16 18 10 11 27 17 21 30 43
 44 24 35 20  9 37 41 46 32 25 36 22 26 19 48 28 58 15]
avg_price_per_room
[ 67.5   72.25  52.   ...  93.7  128.36  44.  ]
no_of_special_requests
[0 1 2 3 4 5]
booking_status
[0 1]
```

In [13]:
```
for i in df.columns:
    print(i)
    print(df[i].value_counts())
```

```
no_of_adults
2    30771
1     7089
3     4061
0      167
4       12
Name: no_of_adults, dtype: int64
no_of_children
0    37786
1     2729
2     1561
3       17
4        5
9        2
Name: no_of_children, dtype: int64
no_of_weekend_nights
0    18137
```

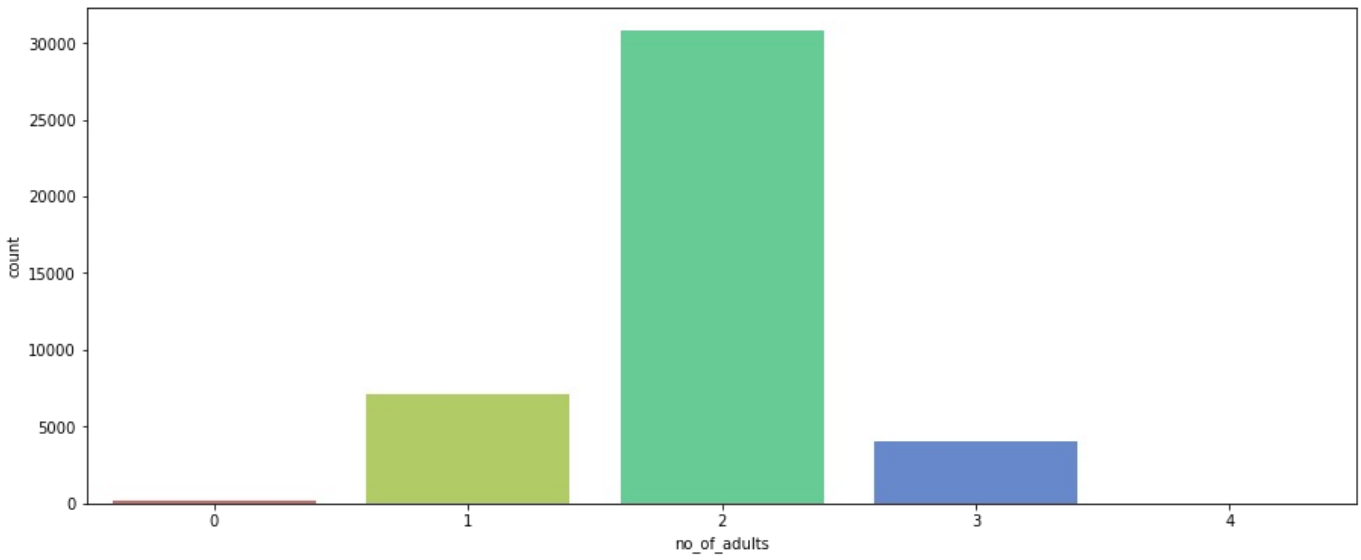
```
2    12143
1    11389
3      239
4      128
5       46
6       17
7        1
Name: no_of_weekend_nights, dtype: int64
no_of_week_nights
2     12905
3     10286
1      9558
4      4175
5      2573
0      1937
6       237
7       160
8        84
10       82
9        53
11       16
15       11
12        9
13        5
16        4
14        3
17        2
Name: no_of_week_nights, dtype: int64
type_of_meal_plan
0    35463
2     3421
1     3210
3        6
Name: type_of_meal_plan, dtype: int64
required_car_parking_space
0    41037
1     1063
Name: required_car_parking_space, dtype: int64
room_type_reserved
0    29853
1     9135
3     1465
2     1131
4      320
5      190
6        6
Name: room_type_reserved, dtype: int64
lead_time
1      846
0      827
2      502
6      498
3      480
      ...
326      1
353      1
300      1
325      1
348      1
Name: lead_time, Length: 338, dtype: int64
arrival_year
2018    36050
2017     6050
Name: arrival_year, dtype: int64
arrival_month
10    6453
8     5763
9     5148
7     4681
12    3391
6     3356
5     3292
4     3006
11    2803
3     2201
2     1201
1      805
Name: arrival_month, dtype: int64
arrival_date
27    1633
16    1600
30    1538
13    1479
```

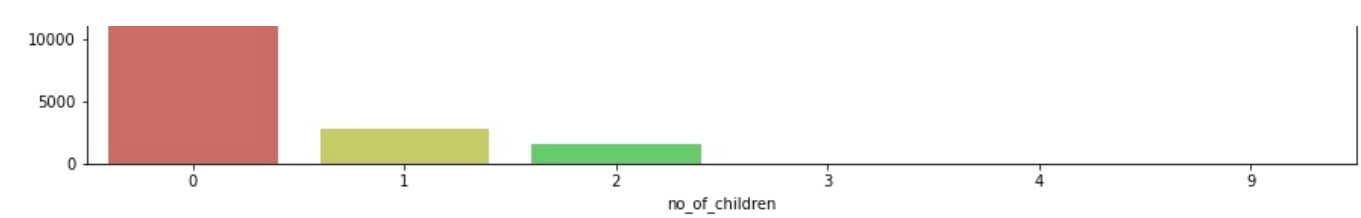
```
26    1460
25    1447
4     1438
20    1437
1     1428
6     1426
17    1426
9     1424
8     1412
2     1408
28    1398
29    1392
15    1376
5     1367
23    1342
12    1339
19    1333
21    1326
18    1291
3     1280
11    1278
22    1274
10    1253
24    1218
7     1206
14    1164
31     707
Name: arrival_date, dtype: int64
market_segment_type
1    25885
0    14306
2     1384
4      442
3       83
Name: market_segment_type, dtype: int64
repeated_guest
0    40871
1     1229
Name: repeated_guest, dtype: int64
no_of_previous_cancellations
0     41755
1       195
2        53
3        51
11       22
4        15
13        3
5         3
6         2
8         1
Name: no_of_previous_cancellations, dtype: int64
no_of_previous_bookings_not_canceled
0     40993
1       262
2       147
3       112
5       102
4        87
8        52
6        49
7        44
11       36
10       34
12       23
18       21
9        20
16       14
14       11
20       10
13       10
23        6
32        5
35        5
43        4
33        4
30        4
24        4
27        4
17        4
46        4
21        4
36        3
47        3
```

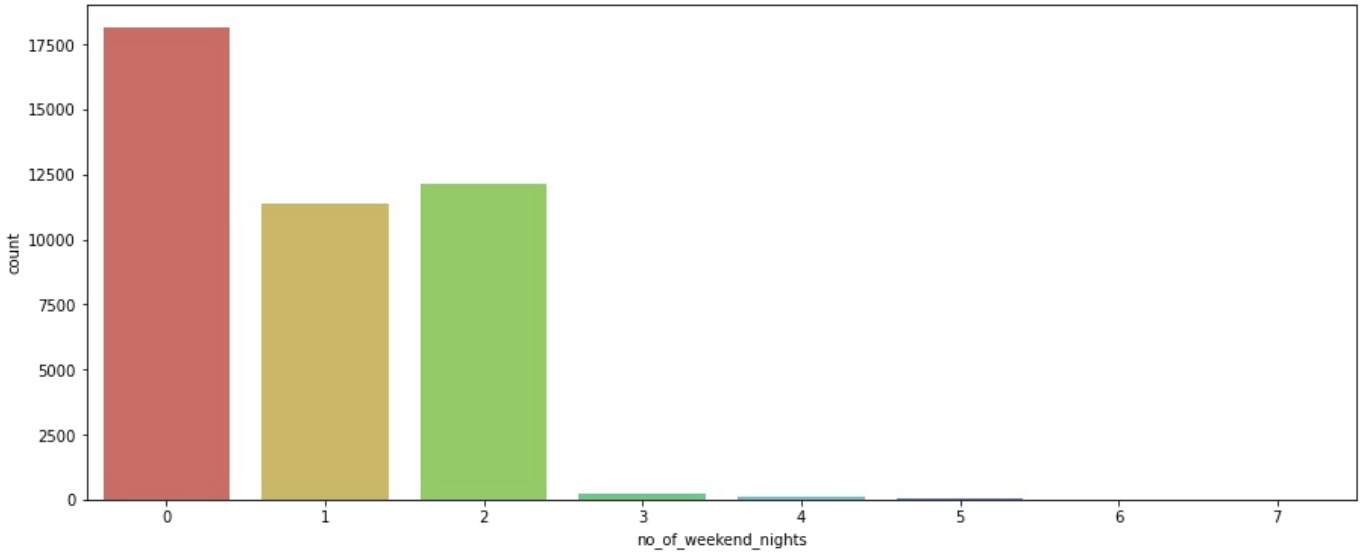
```
48        3
58        2
26        2
25        2
44        2
19        2
37        2
22        1
15        1
41        1
28        1
Name: no_of_previous_bookings_not_canceled, dtype: int64
avg_price_per_room
75.00     1393
65.00     1008
80.75      926
90.00      887
90.95      752
          ...
51.30        1
86.28        1
104.39       1
81.09        1
127.20       1
Name: avg_price_per_room, Length: 2286, dtype: int64
no_of_special_requests
0    24554
1    11885
2     4900
3      660
4      100
5        1
Name: no_of_special_requests, dtype: int64
booking_status
0    25596
1    16504
Name: booking_status, dtype: int64
```

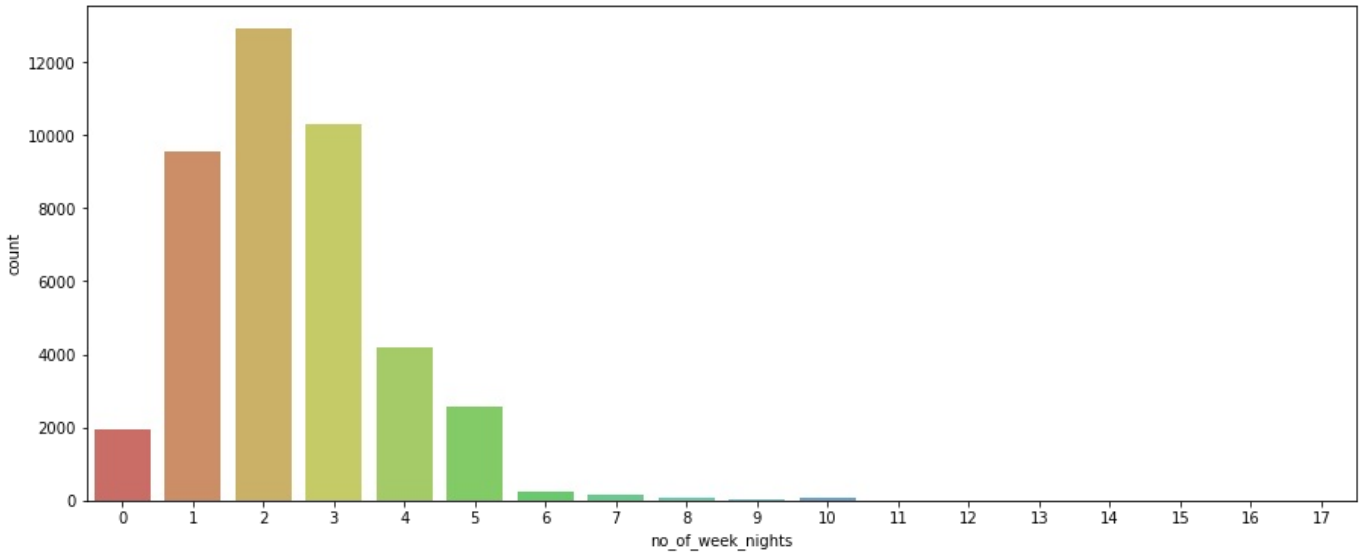
In [14]:
```
import warnings
warnings.filterwarnings('ignore')
```

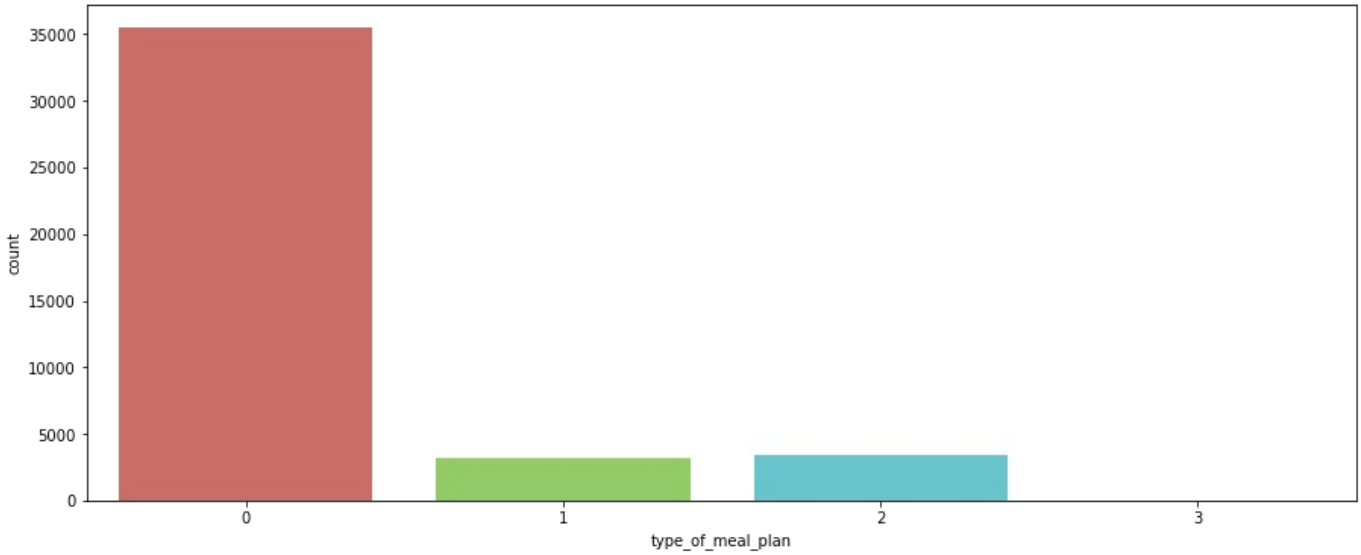
In [15]:
```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.countplot(data=df,  x=df[i], palette = 'hls')
    plt.show()
```



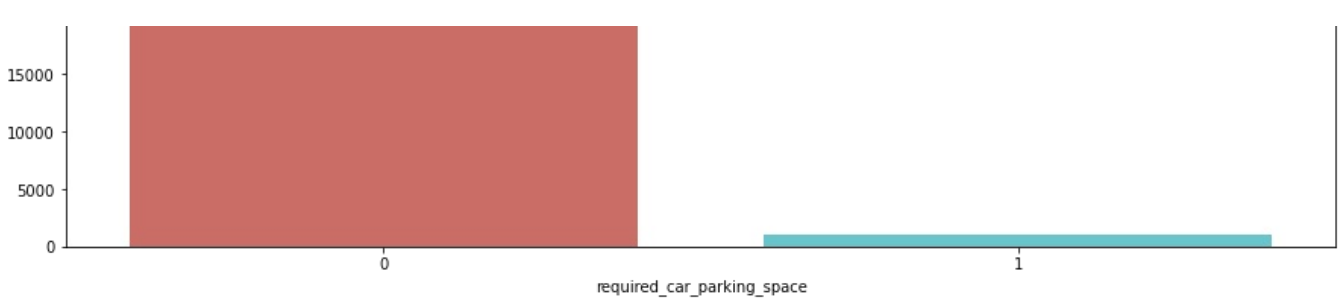

10000
5000
0
0
1
2
3
4
9
no_of_children

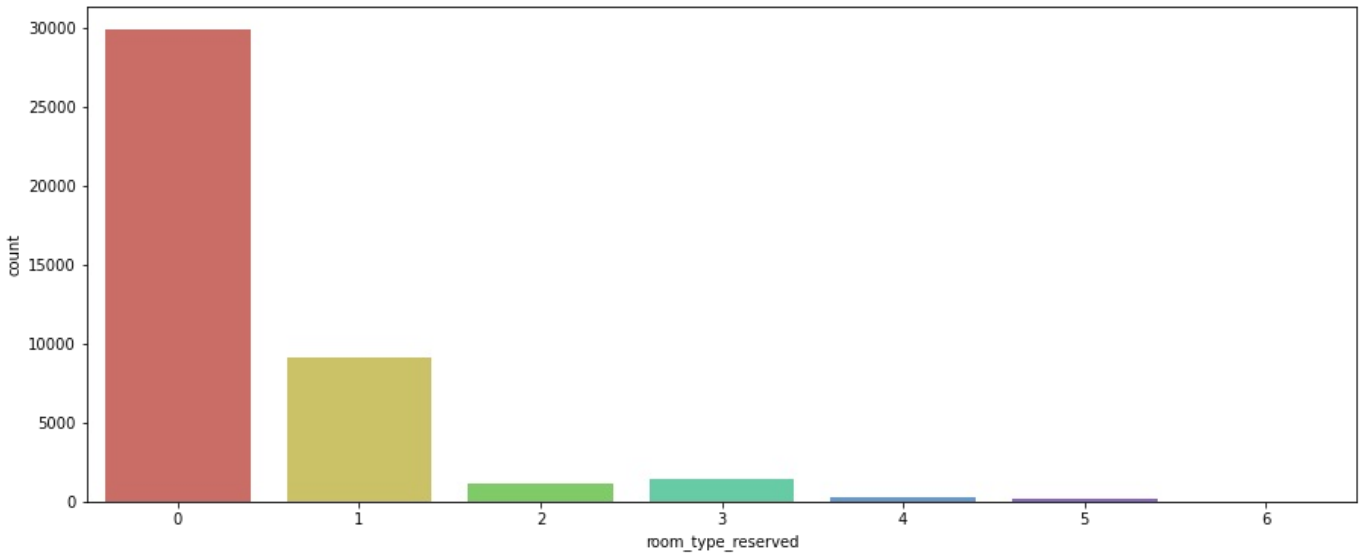

17500
15000
12500
10000
7500
5000
2500
0
count
0
1
2
3
4
5
6
7
no_of_weekend_nights

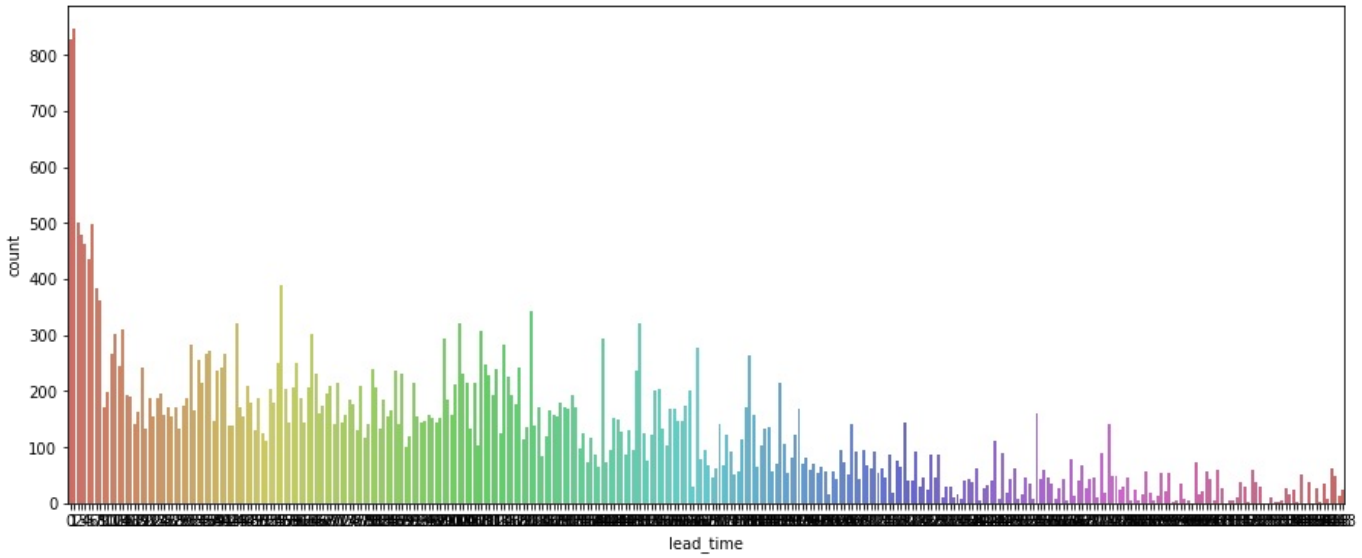

12000
10000
8000
6000
4000
2000
0
count
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
no_of_week_nights

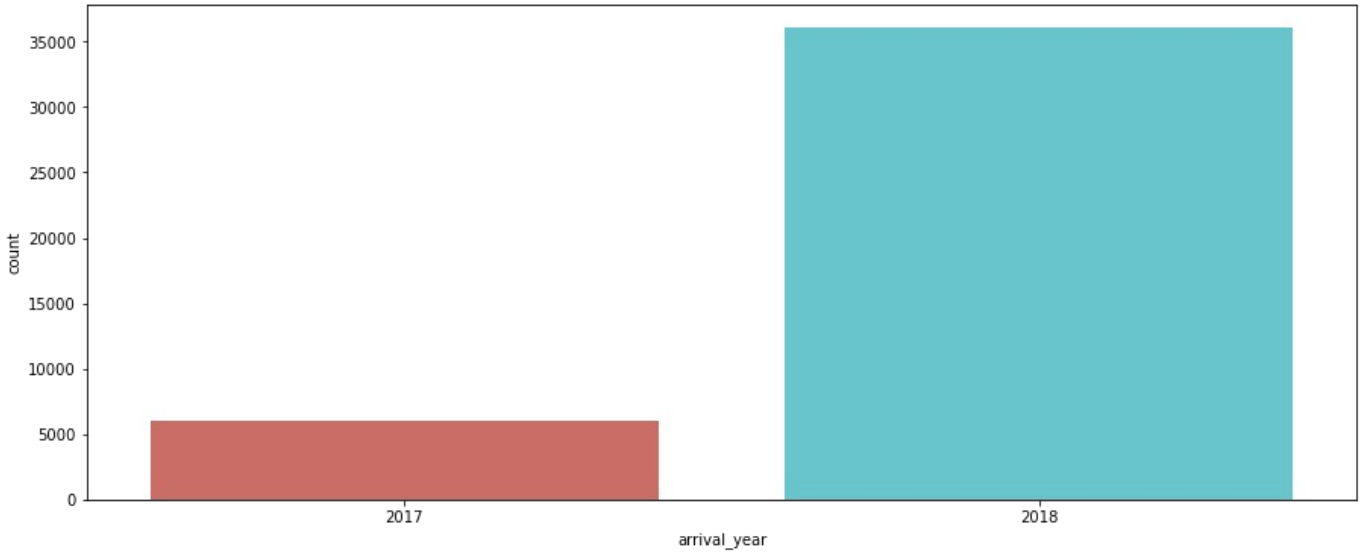

35000
30000
25000
20000
15000
10000
5000
0
count
0
1
2
3
type_of_meal_plan

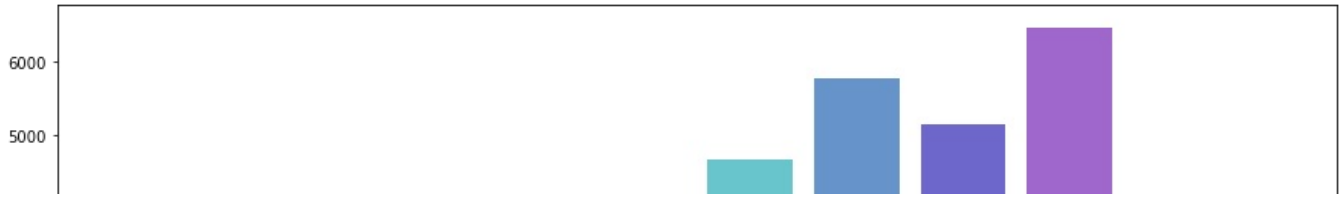

40000
35000
30000
25000
20000
count

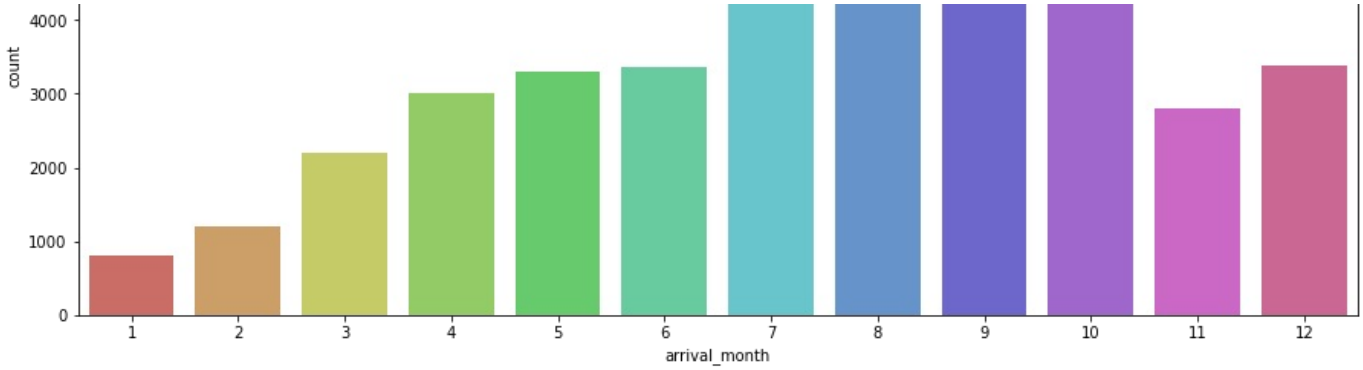

15000
10000
5000
0
0
1
required_car_parking_space

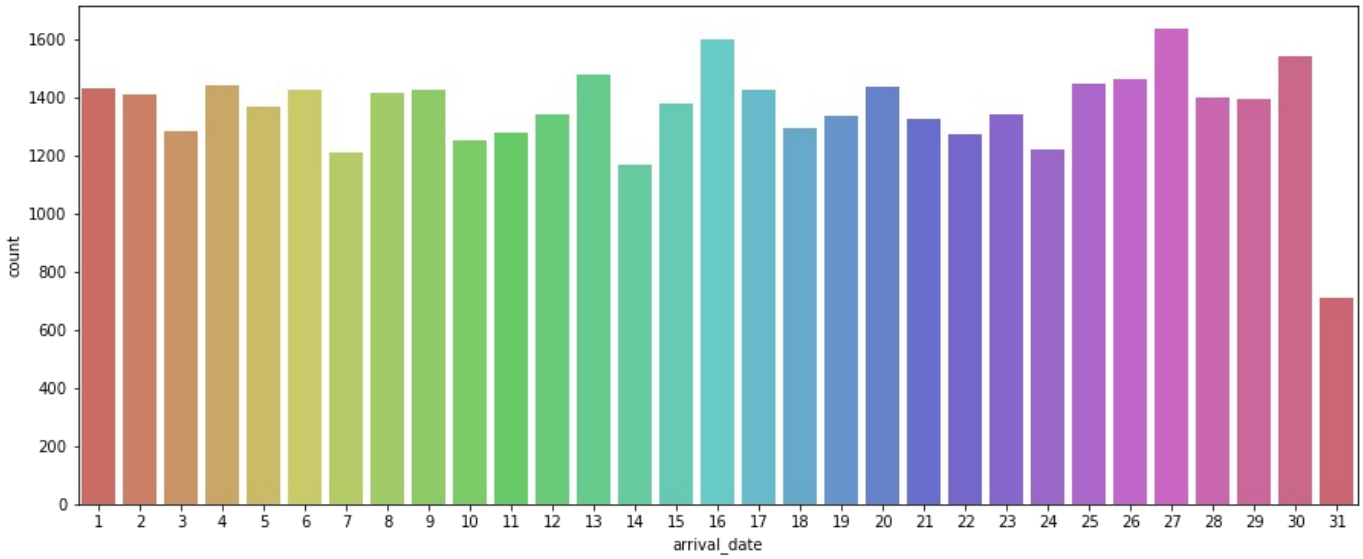

30000
25000
20000
15000
10000
5000
0
count
0
1
2
3
4
5
6
room_type_reserved

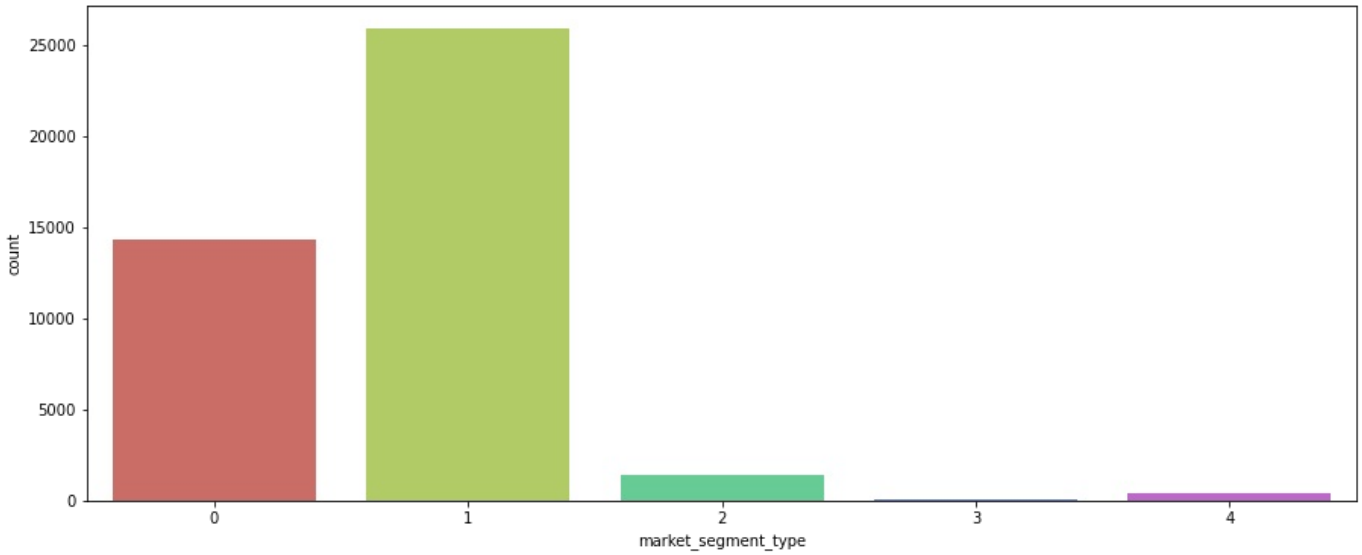

800
700
600
500
400
300
200
100
0
count
lead_time

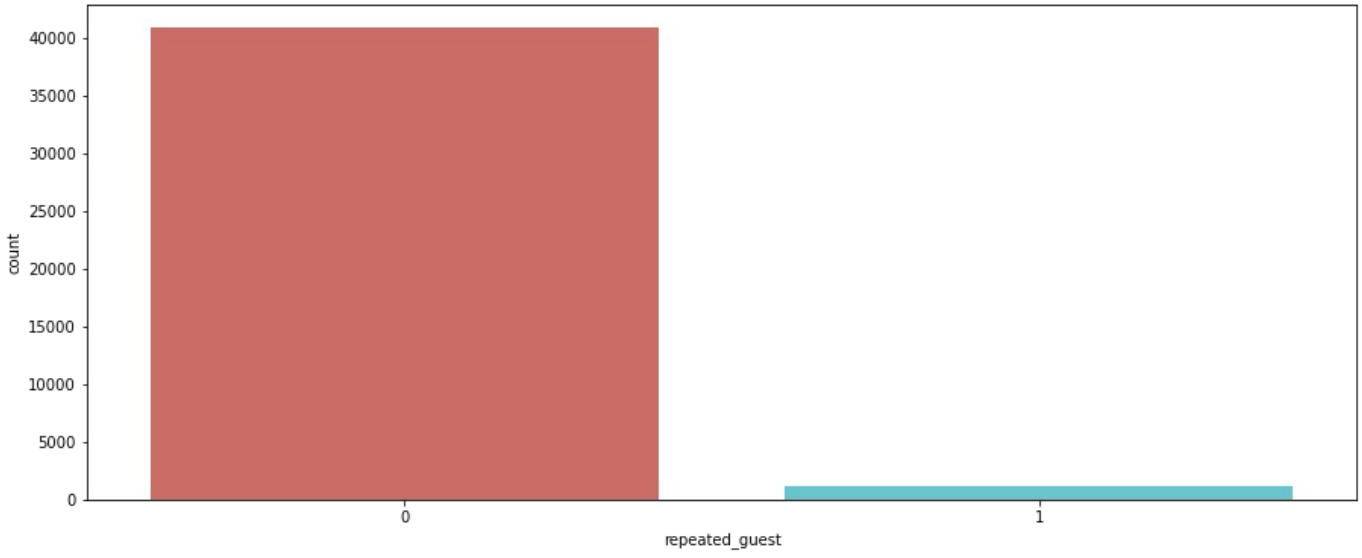

35000
30000
25000
20000
15000
10000
5000
0
count
2017
2018
arrival_year

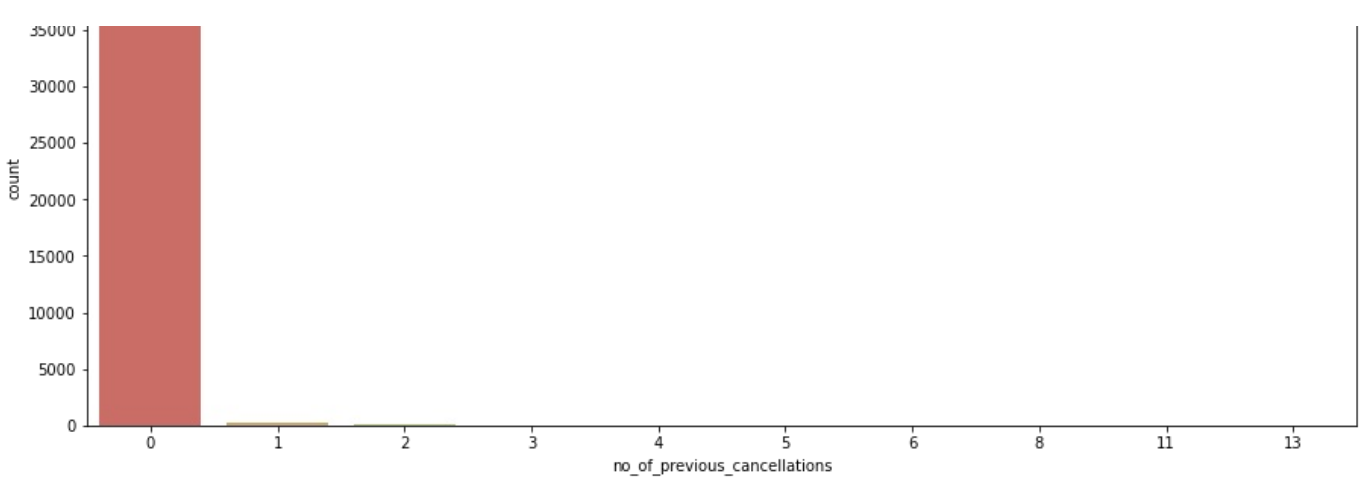

6000
5000

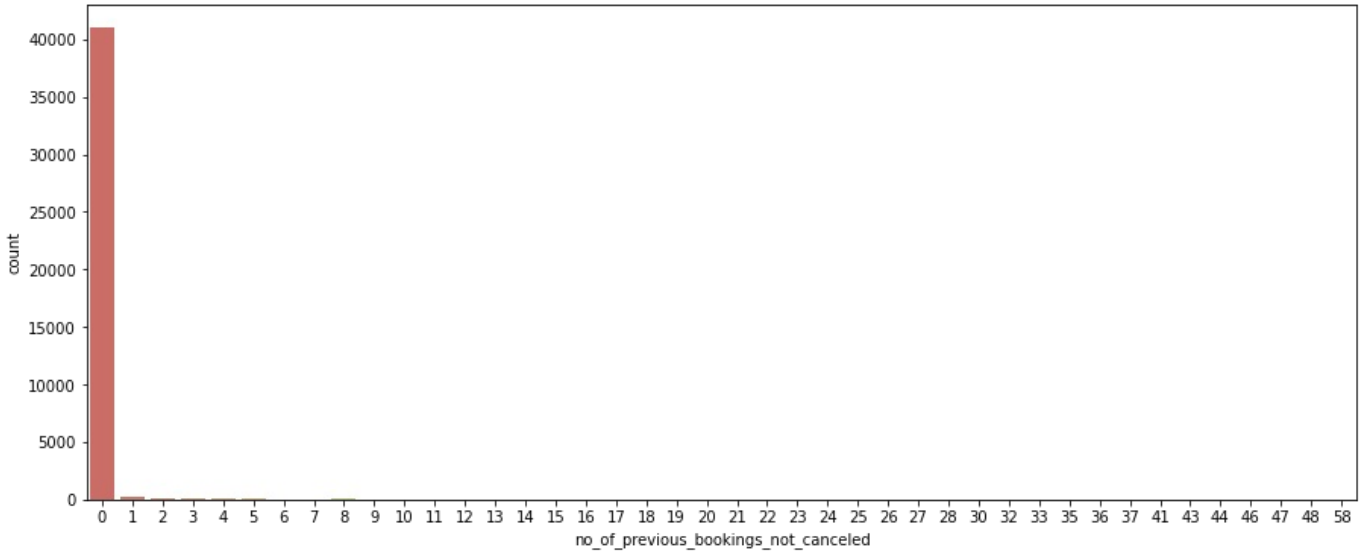

count
arrival_month

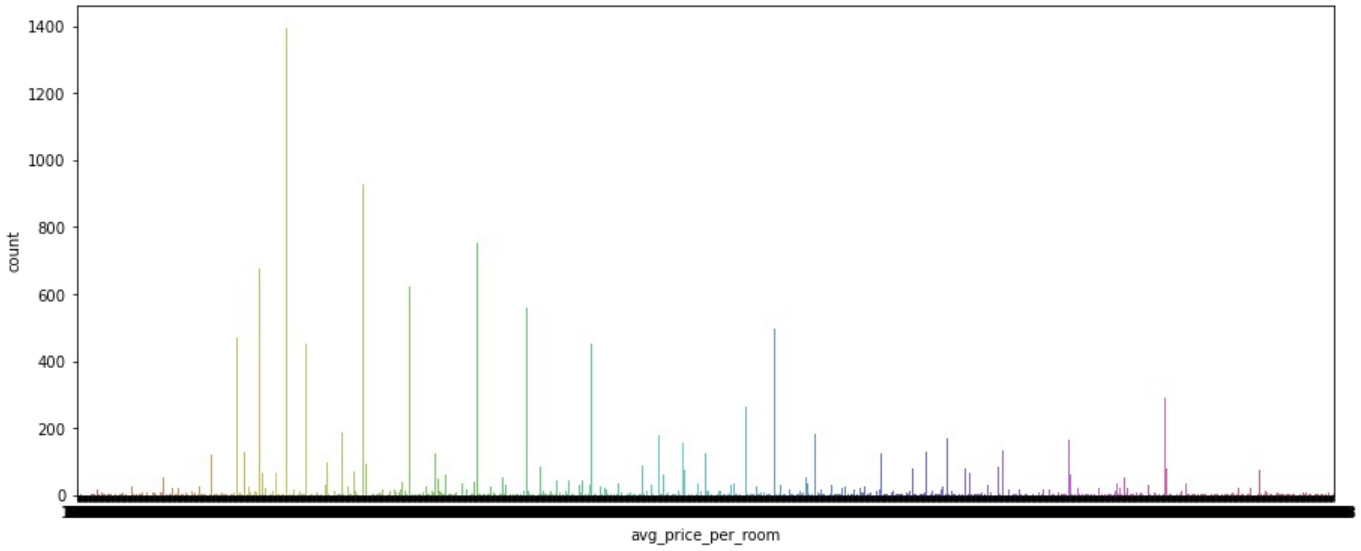

count
arrival_date

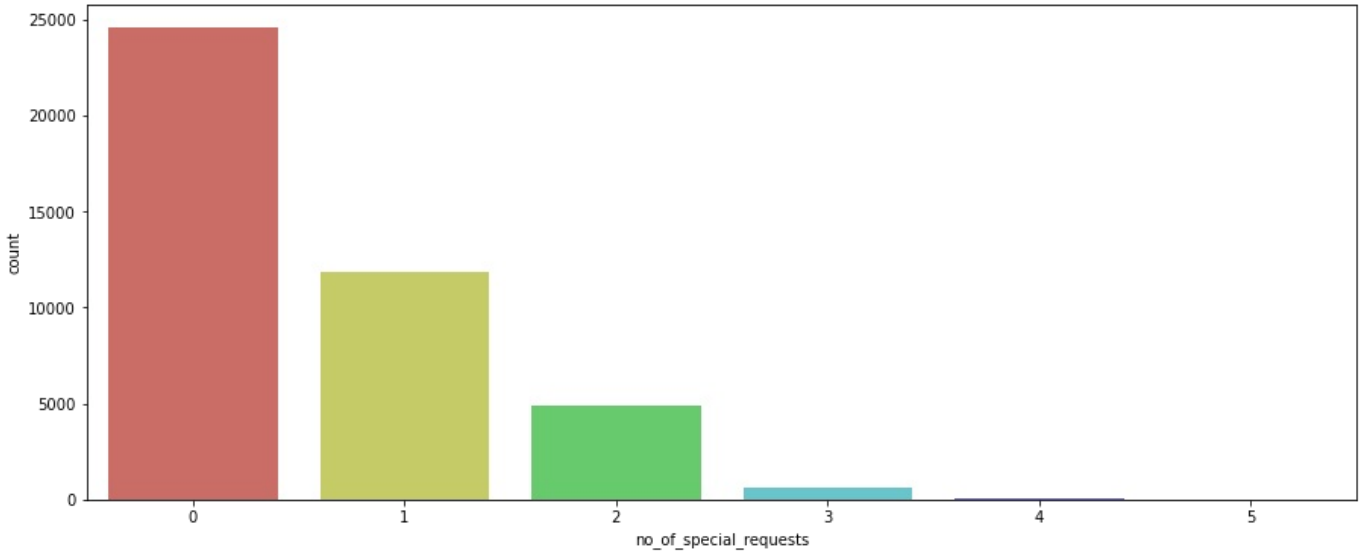

count
market_segment_type

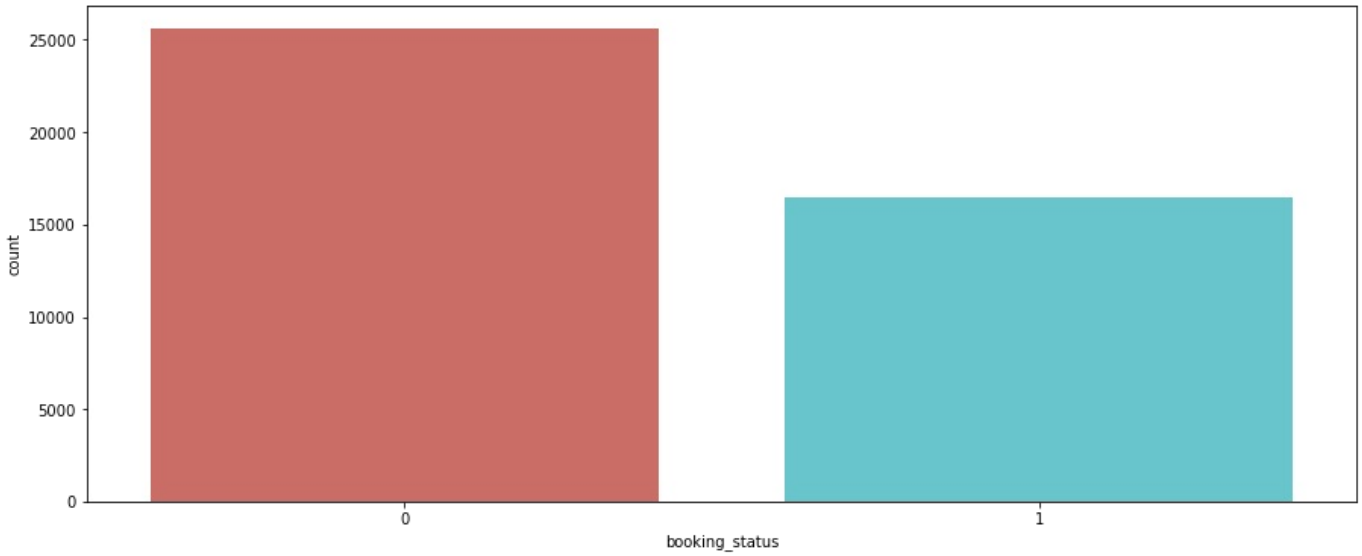

count
repeated_guest


count
no_of_previous_cancellations

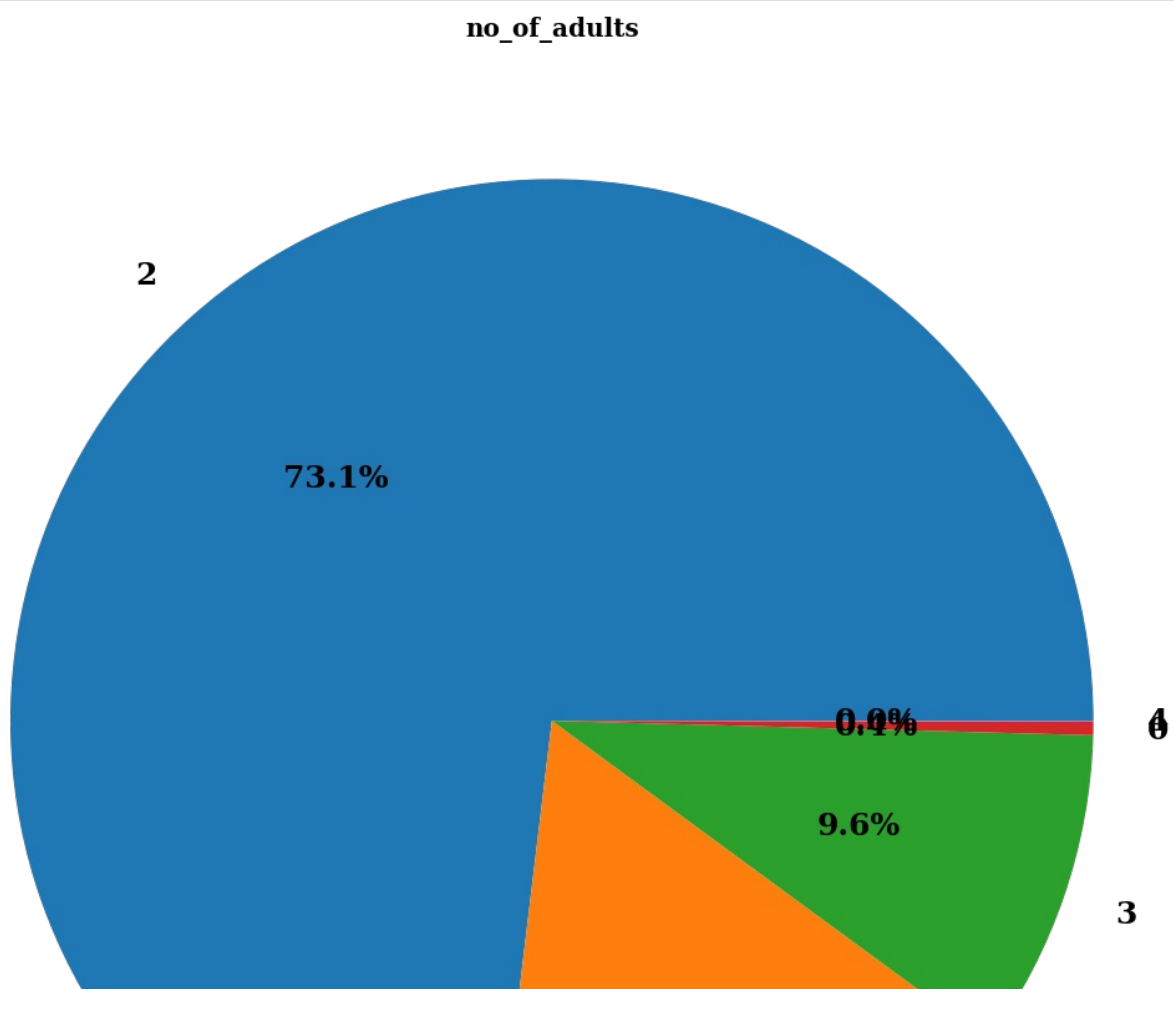

count
no_of_previous_bookings_not_canceled

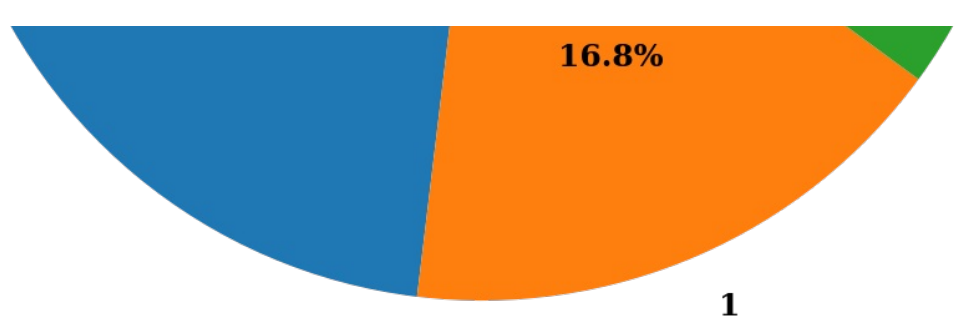

count
avg_price_per_room


count
no_of_special_requests

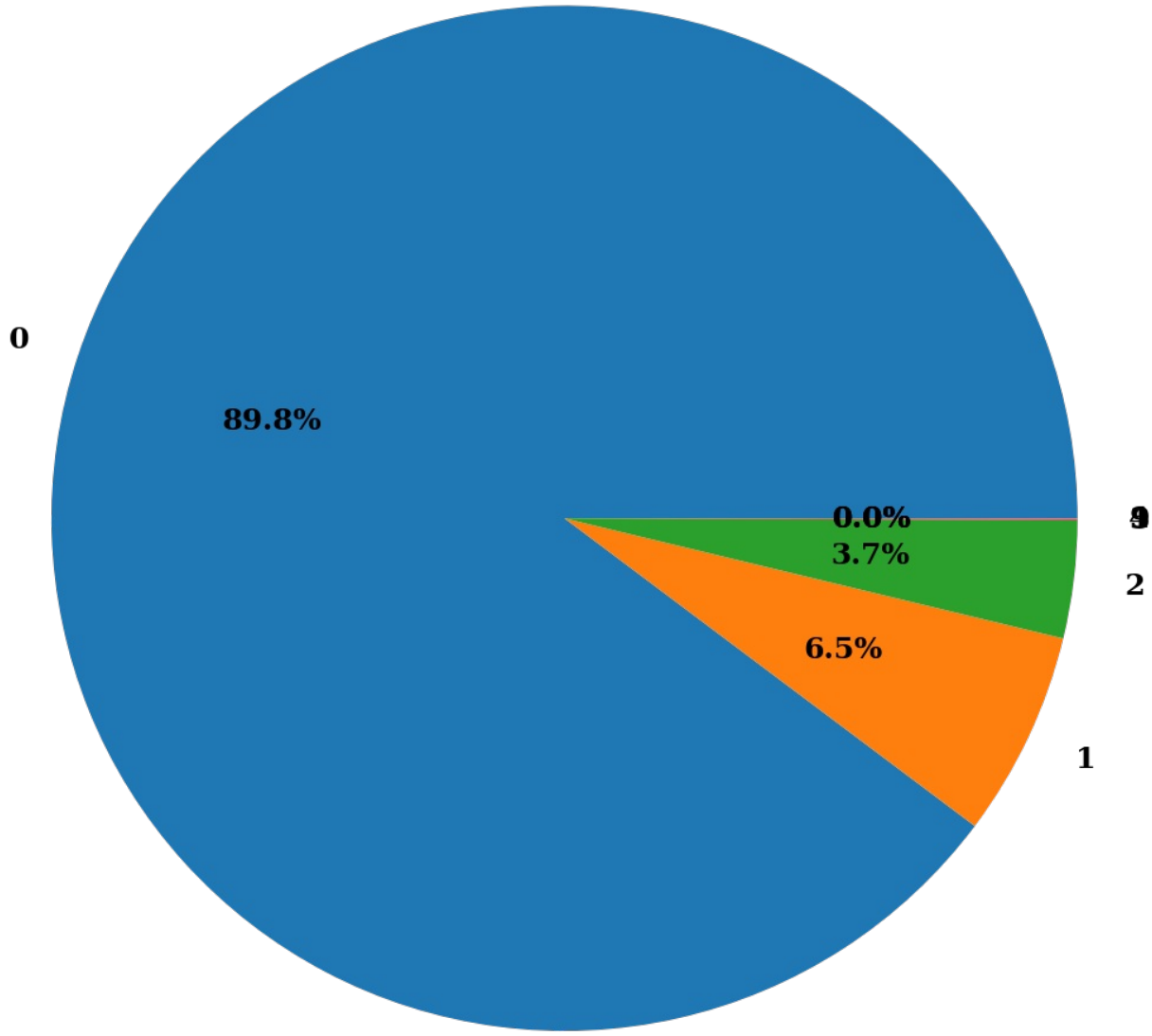


In [16]:
```
for i in df.columns:
    plt.figure(figsize=(30,20))
    plt.pie(df[i].value_counts(), labels=df[i].value_counts().index, autopct='%1.1f%%', textprops={ 'fontsize': 2
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
    hfont = {'fontname':'serif', 'weight': 'bold'}
    plt.title(i, size=20, **hfont)
    plt.show()
```

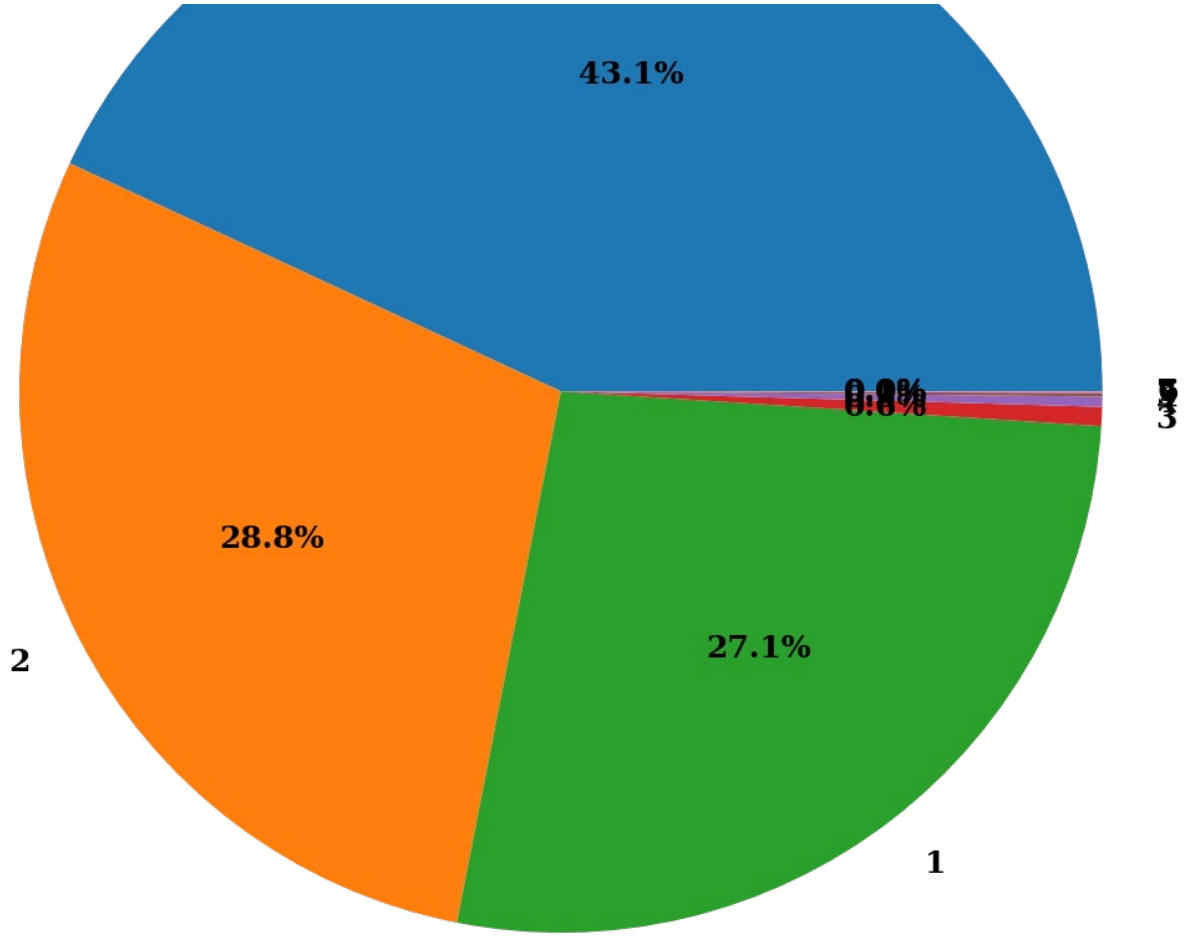




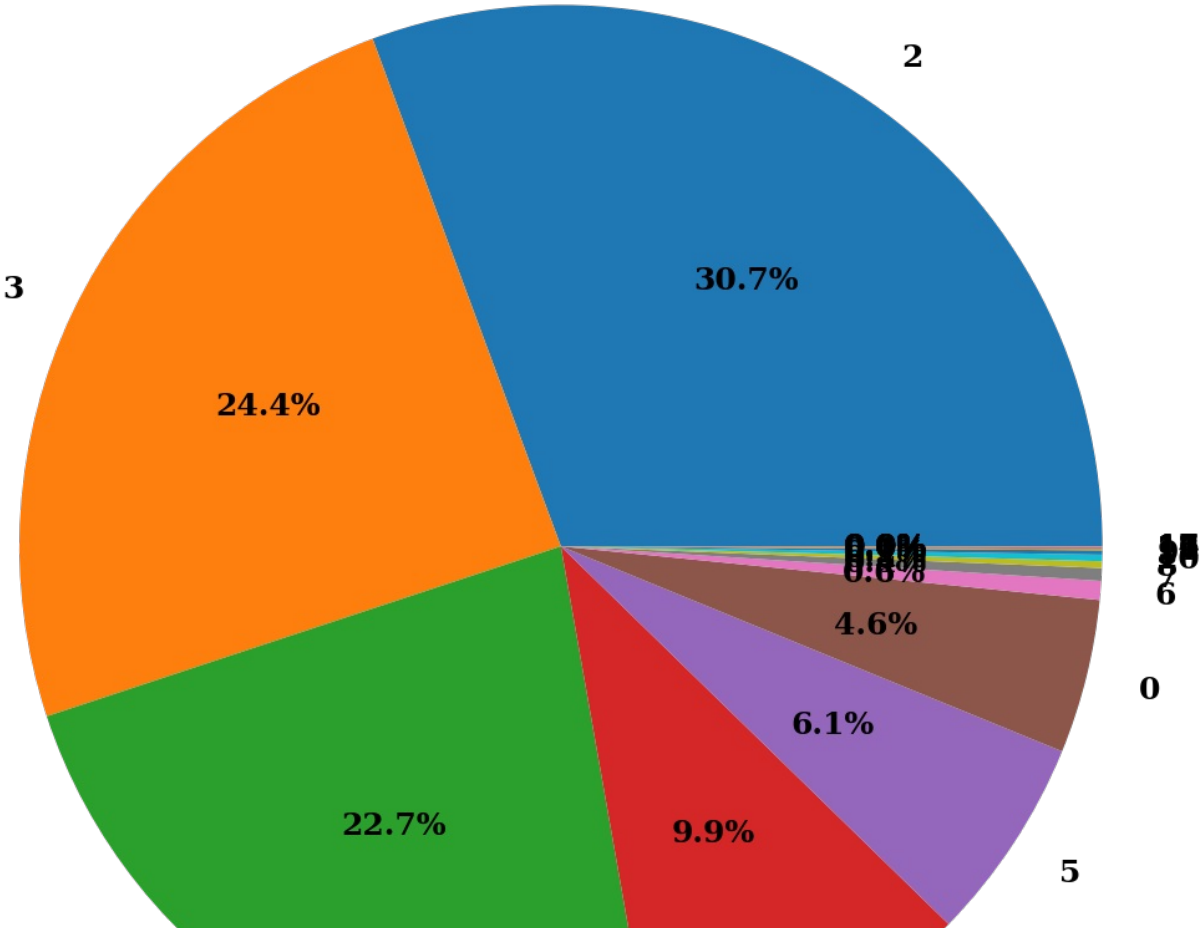
no_of_children

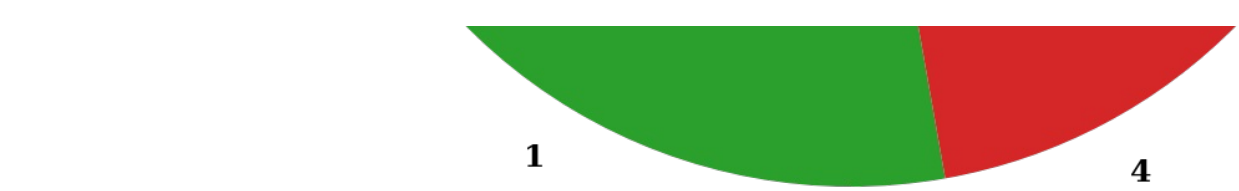


no_of_weekend_nights

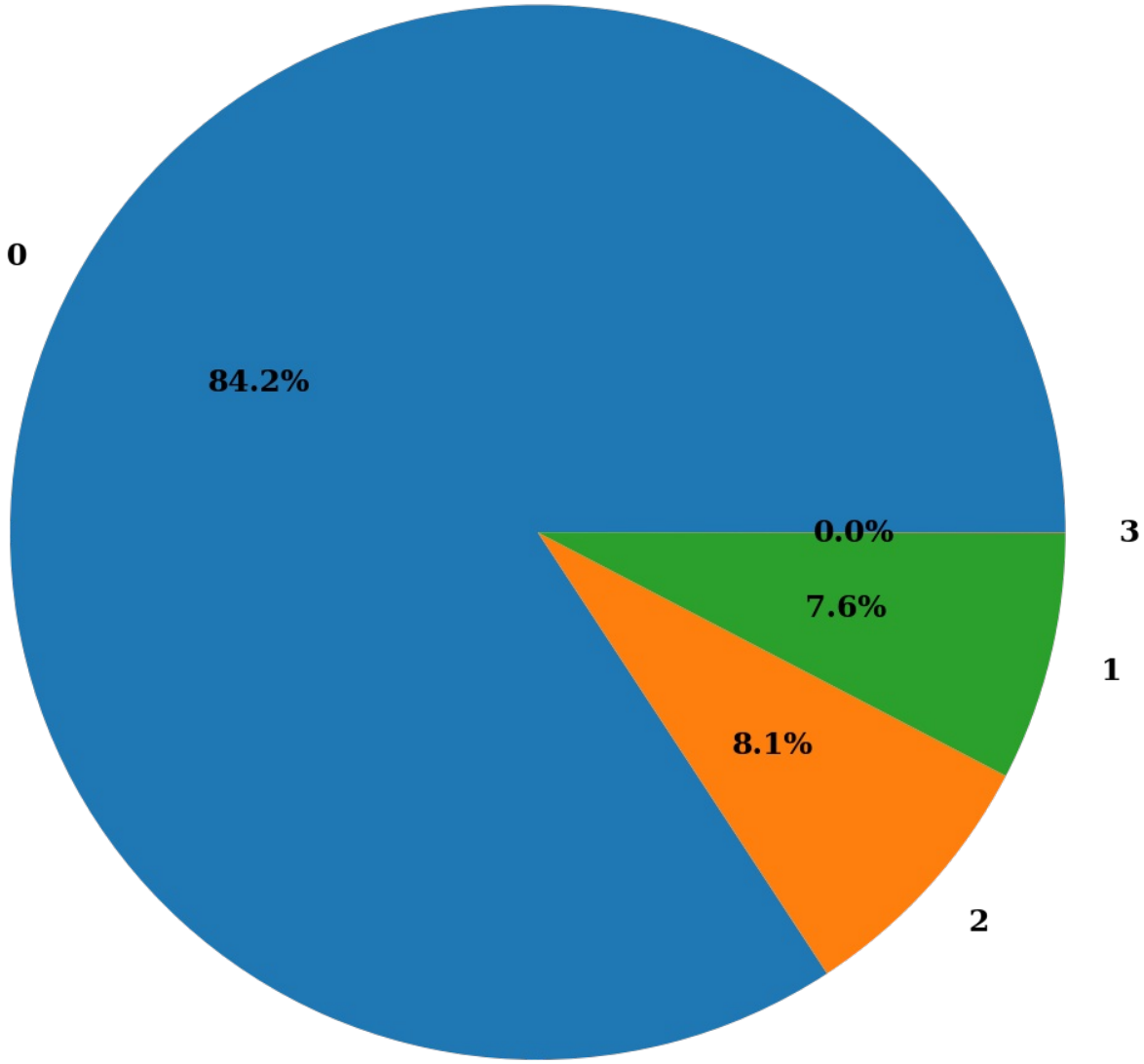


no_of_week_nights

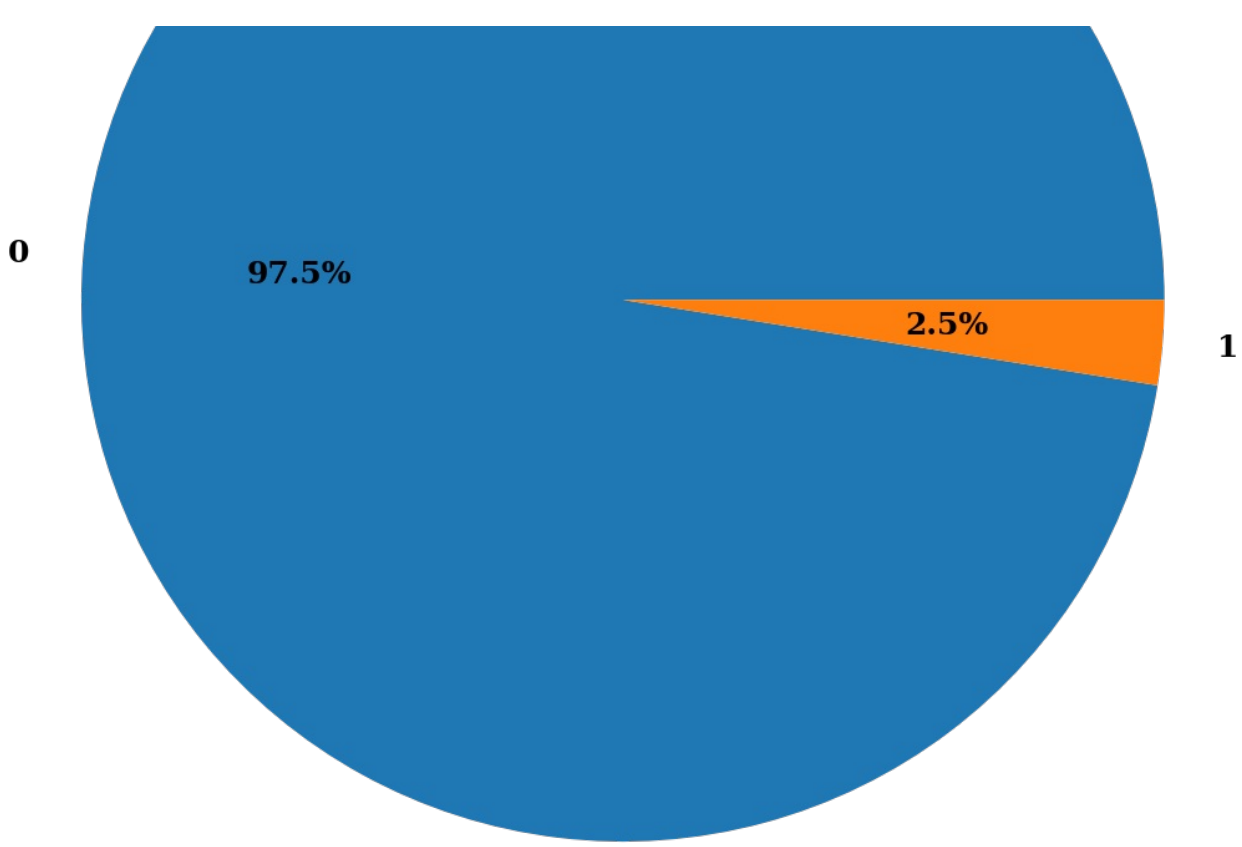




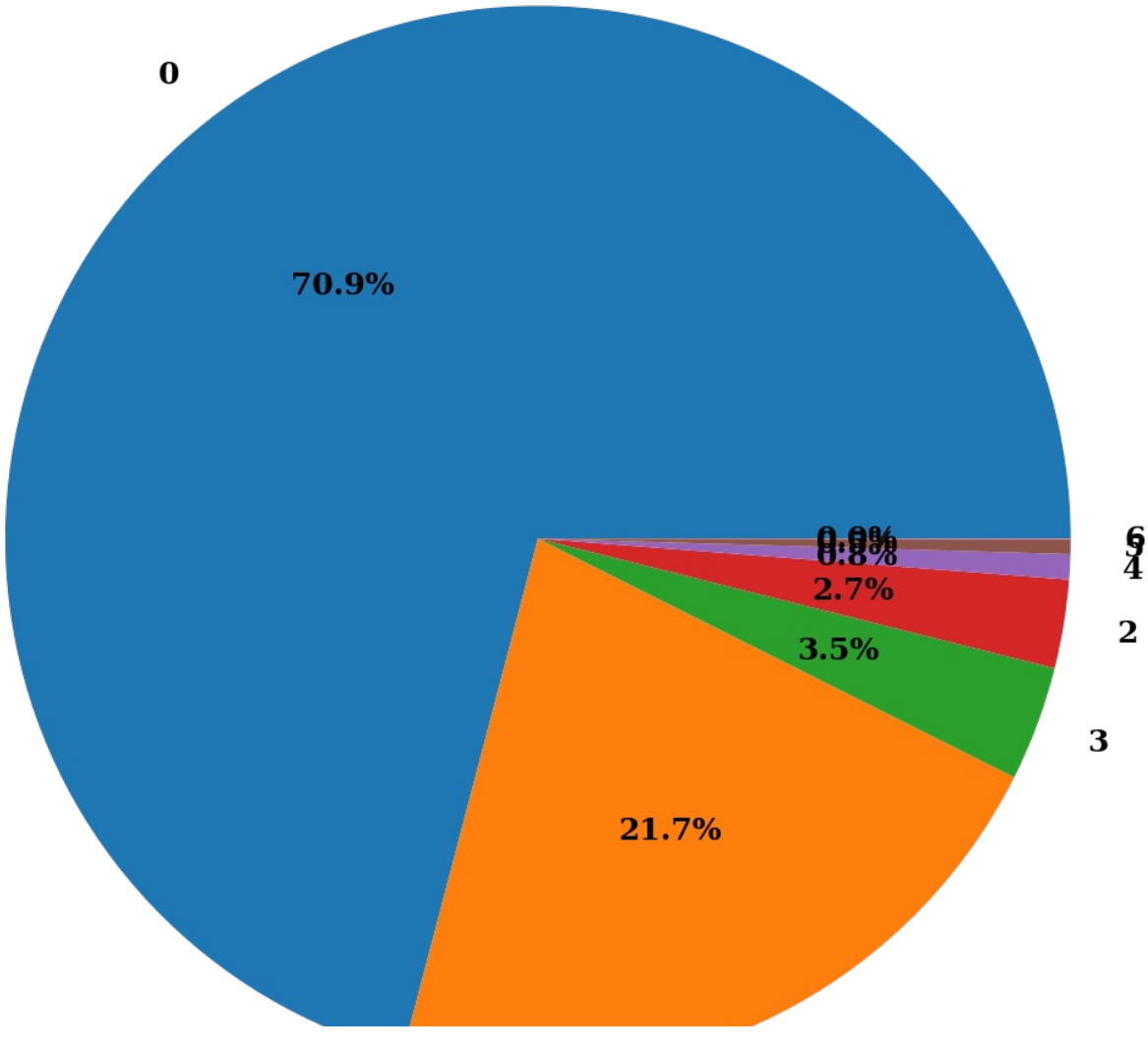
type_of_meal_plan

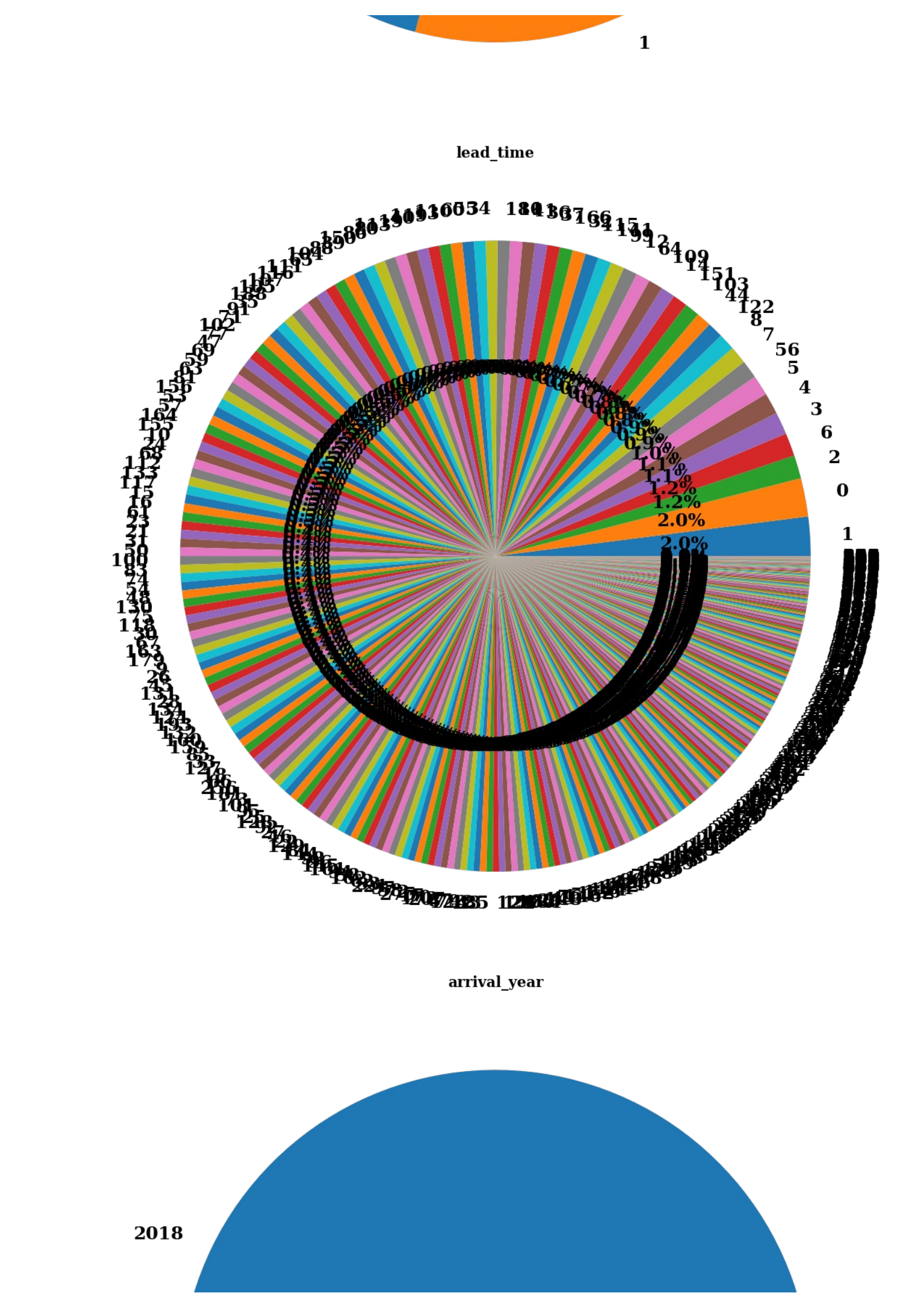


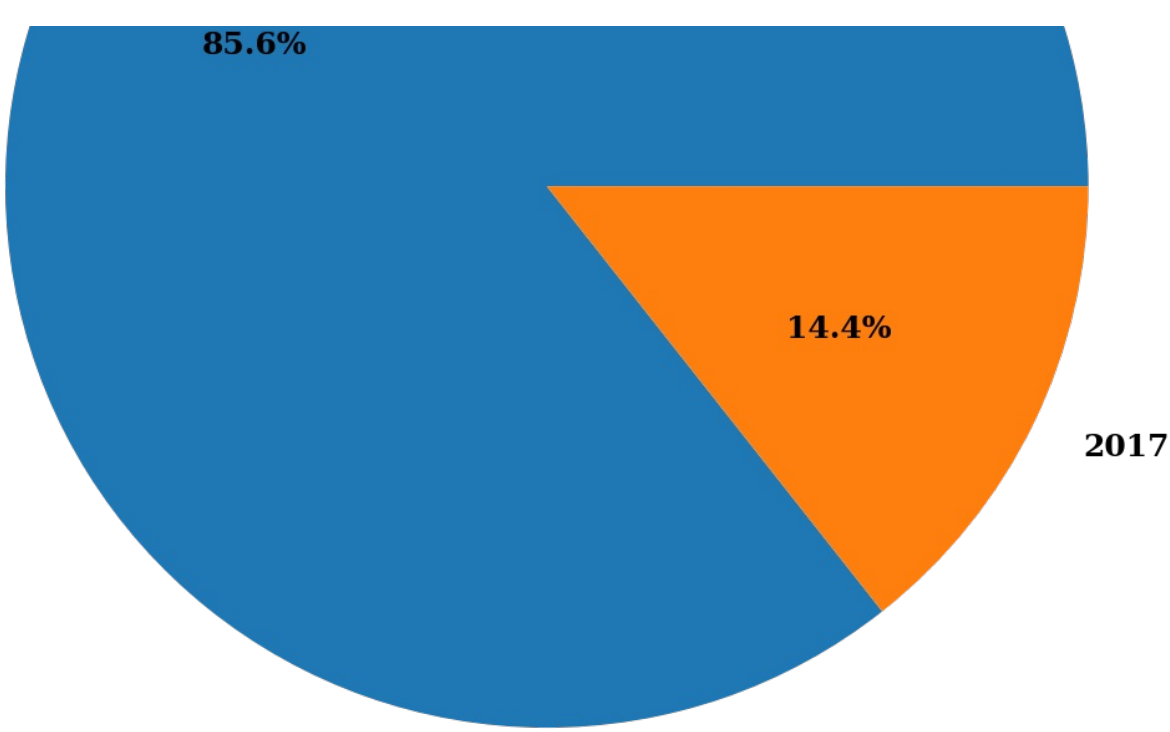
required_car_parking_space

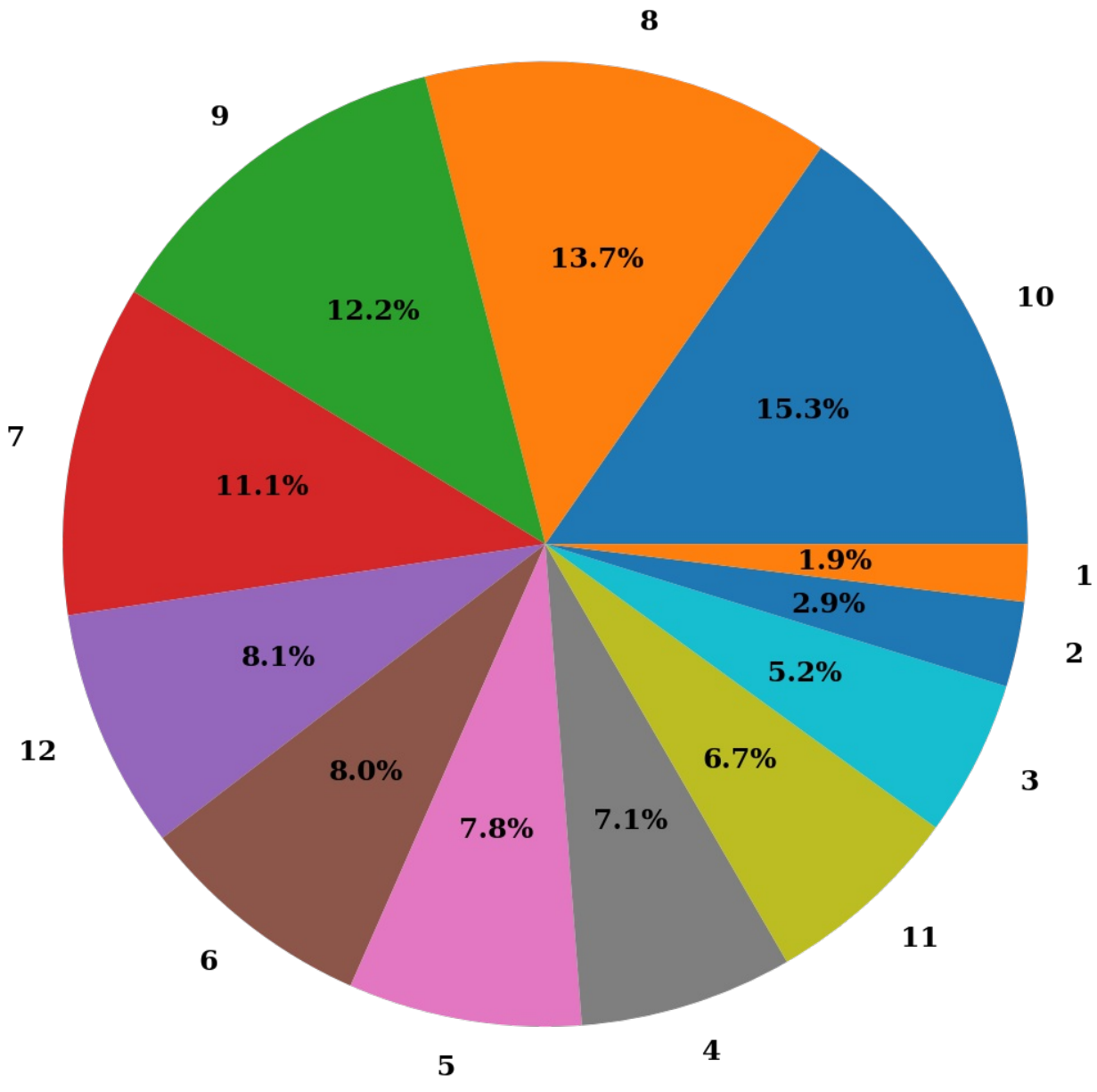


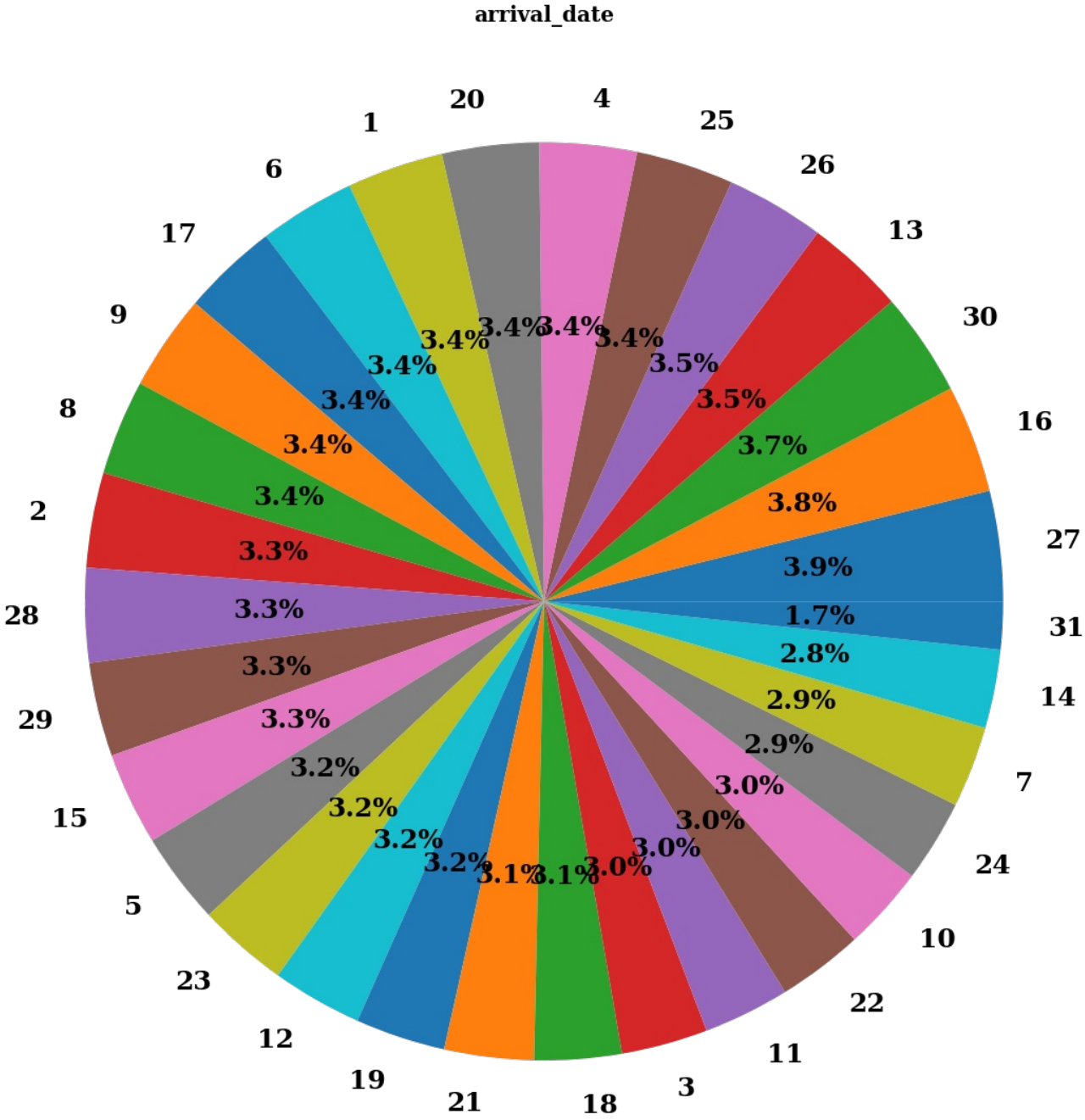
room_type_reserved

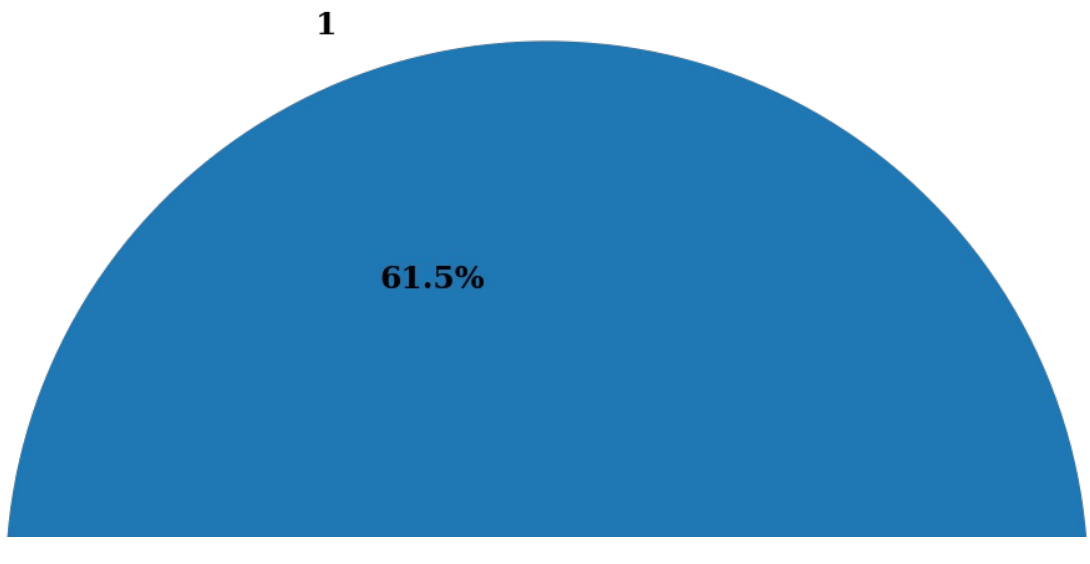


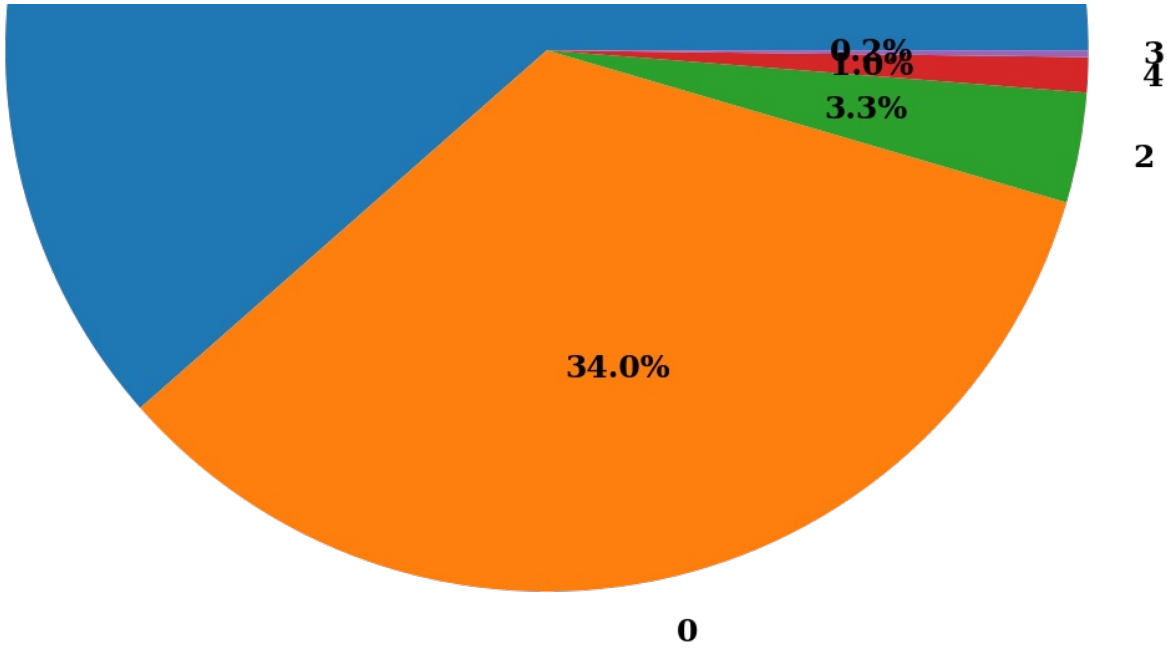

1
lead_time
1.2%
2.0%
arrival_year
2018


85.6%
14.4%
2017

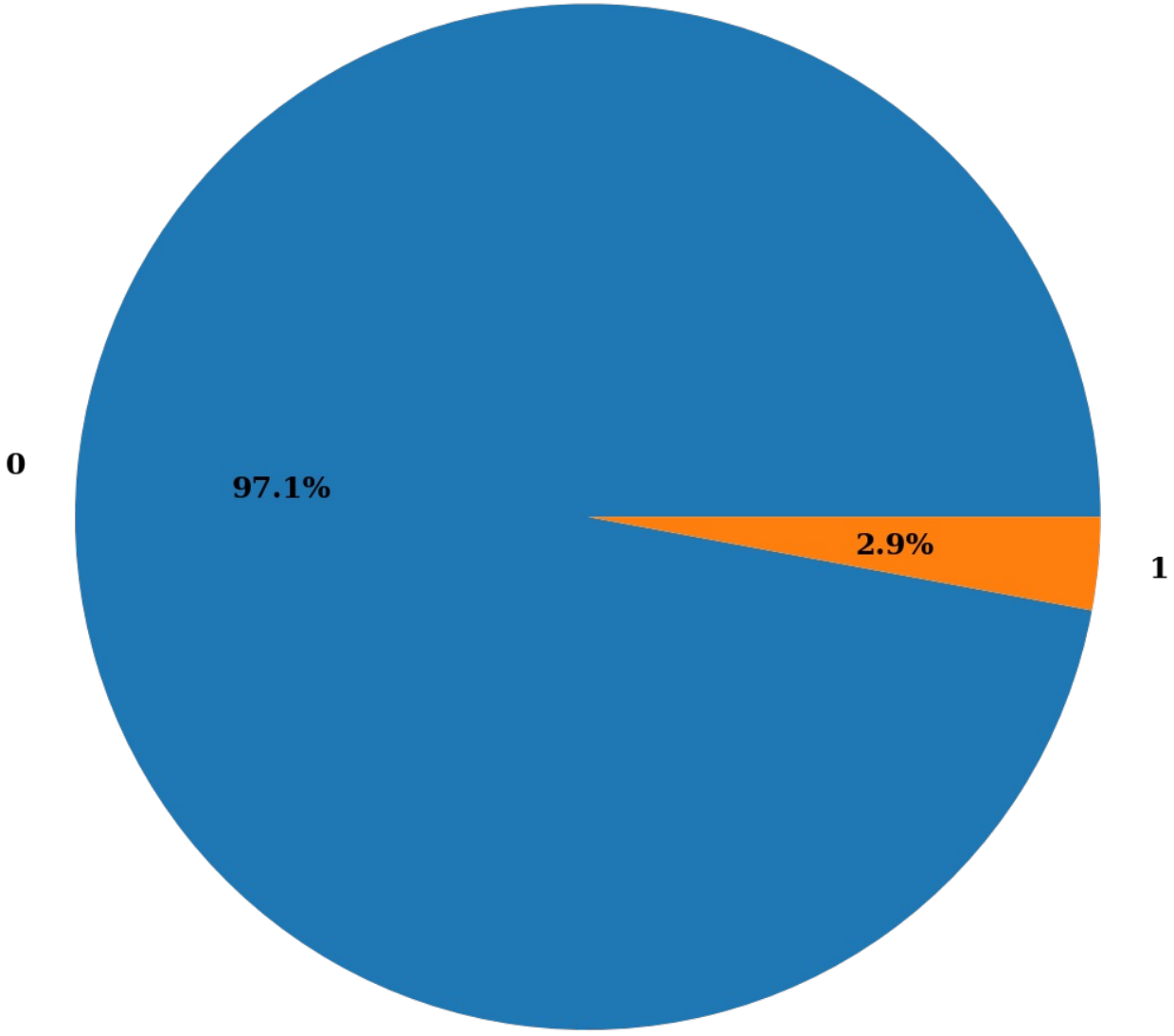

arrival_month
8
9
10
7
12
1
2
3
11
6
5
4
13.7%
12.2%
15.3%
11.1%
1.9%
2.9%
8.1%
5.2%
8.0%
6.7%
7.8%
7.1%

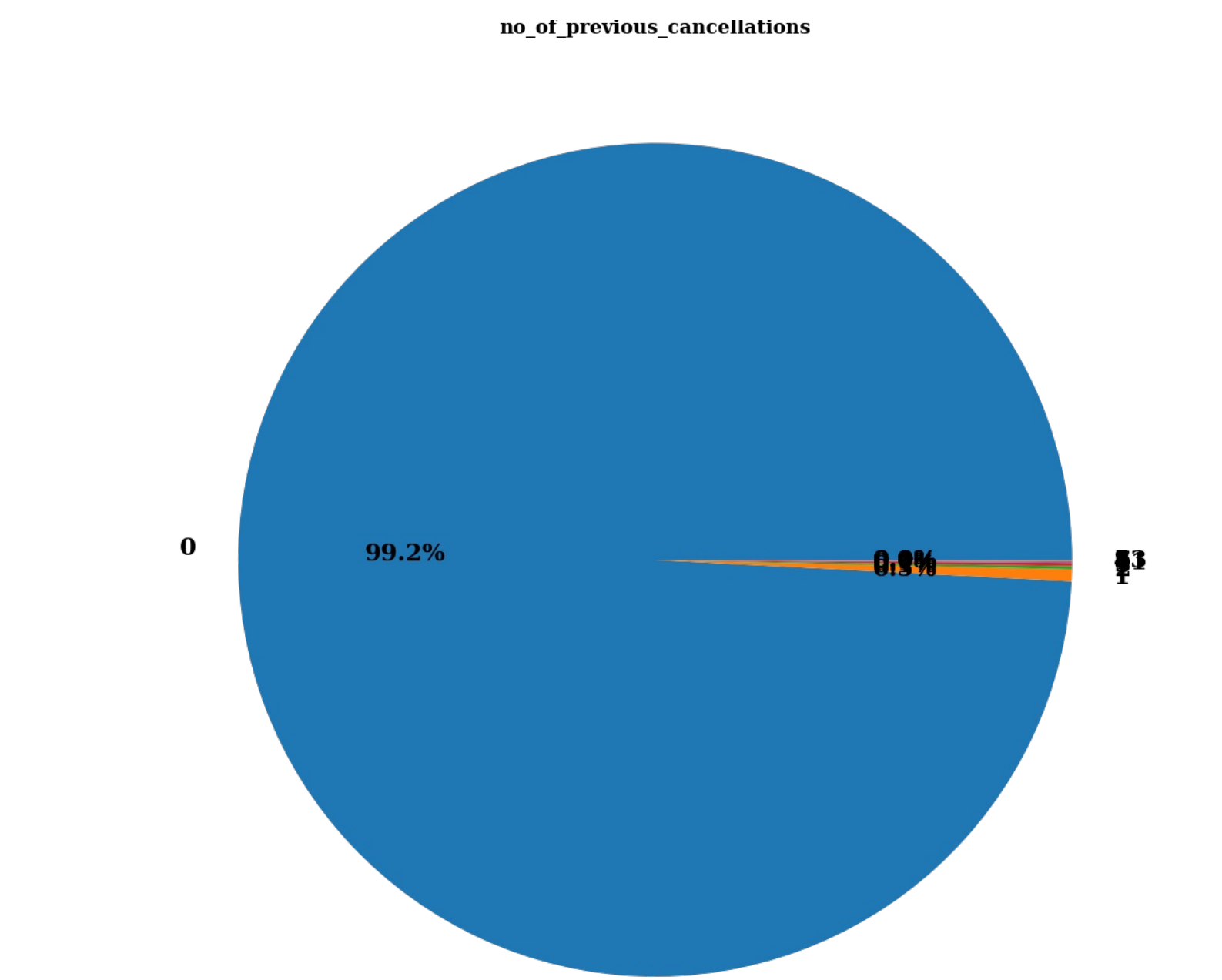

arrival_date
20
4
1
25
6
26
17
13
9
30
8
16
2
27
28
31
29
14
15
7
5
24
23
10
12
22
19
11
21
18
3
3.4%
3.4%
3.4%
3.4%
3.4%
3.4%
3.4%
3.4%
3.5%
3.5%
3.7%
3.8%
3.9%
1.7%
2.8%
2.9%
2.9%
3.0%
3.0%
3.0%
3.0%
3.1%
3.1%
3.2%
3.2%
3.2%
3.2%
3.3%
3.3%
3.3%
3.3%

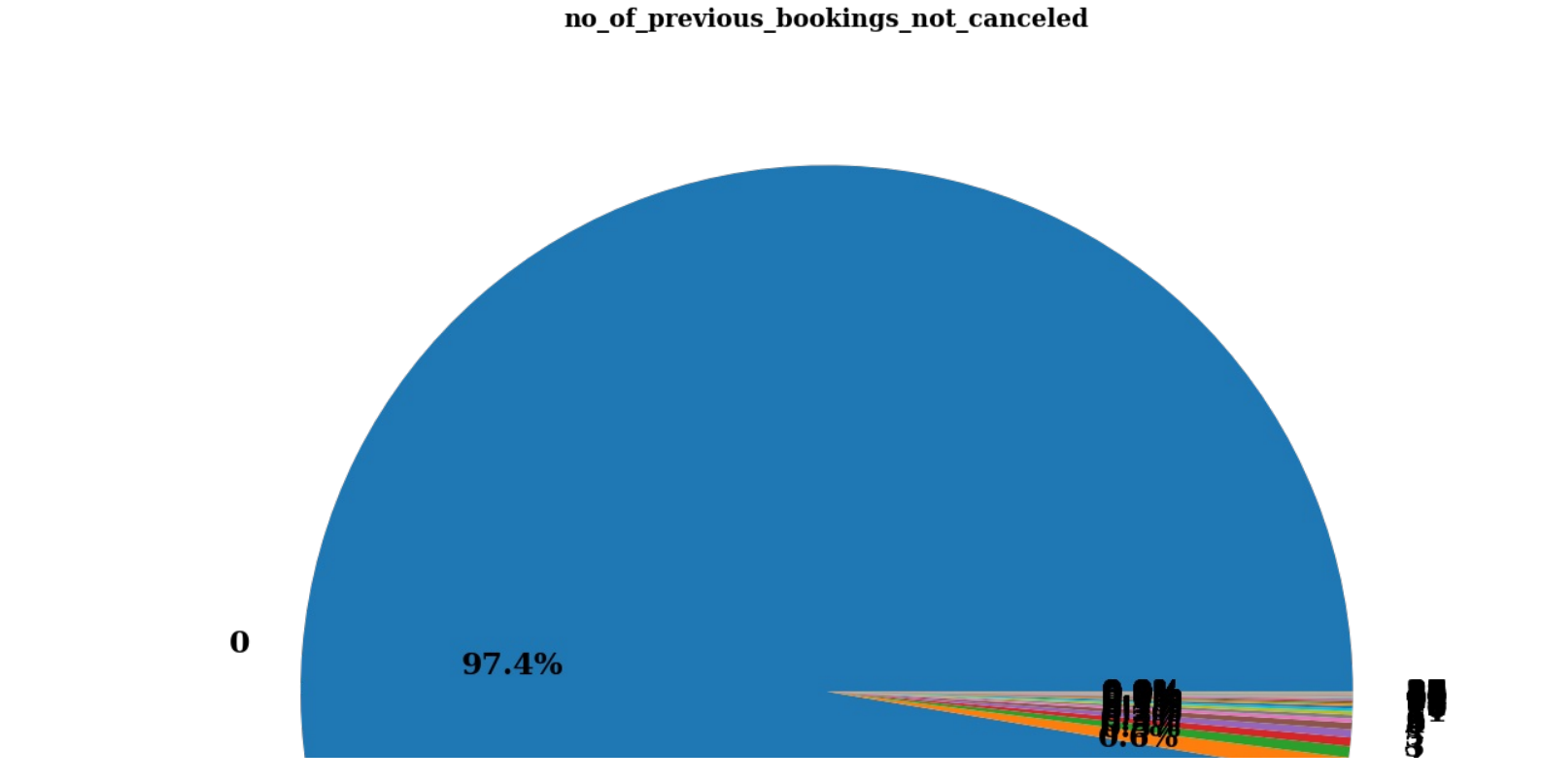

market_segment_type
1
61.5%



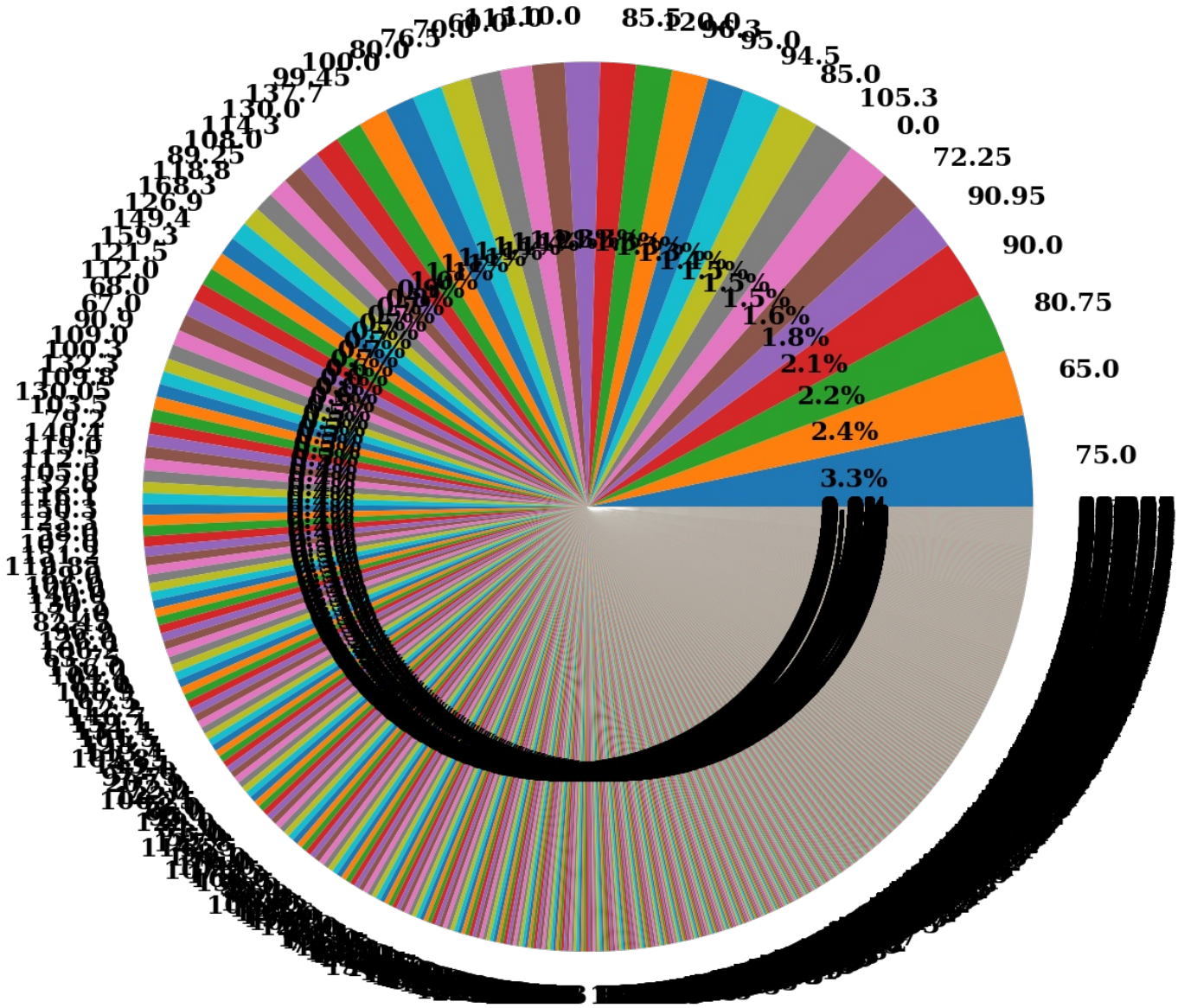
repeated_guest



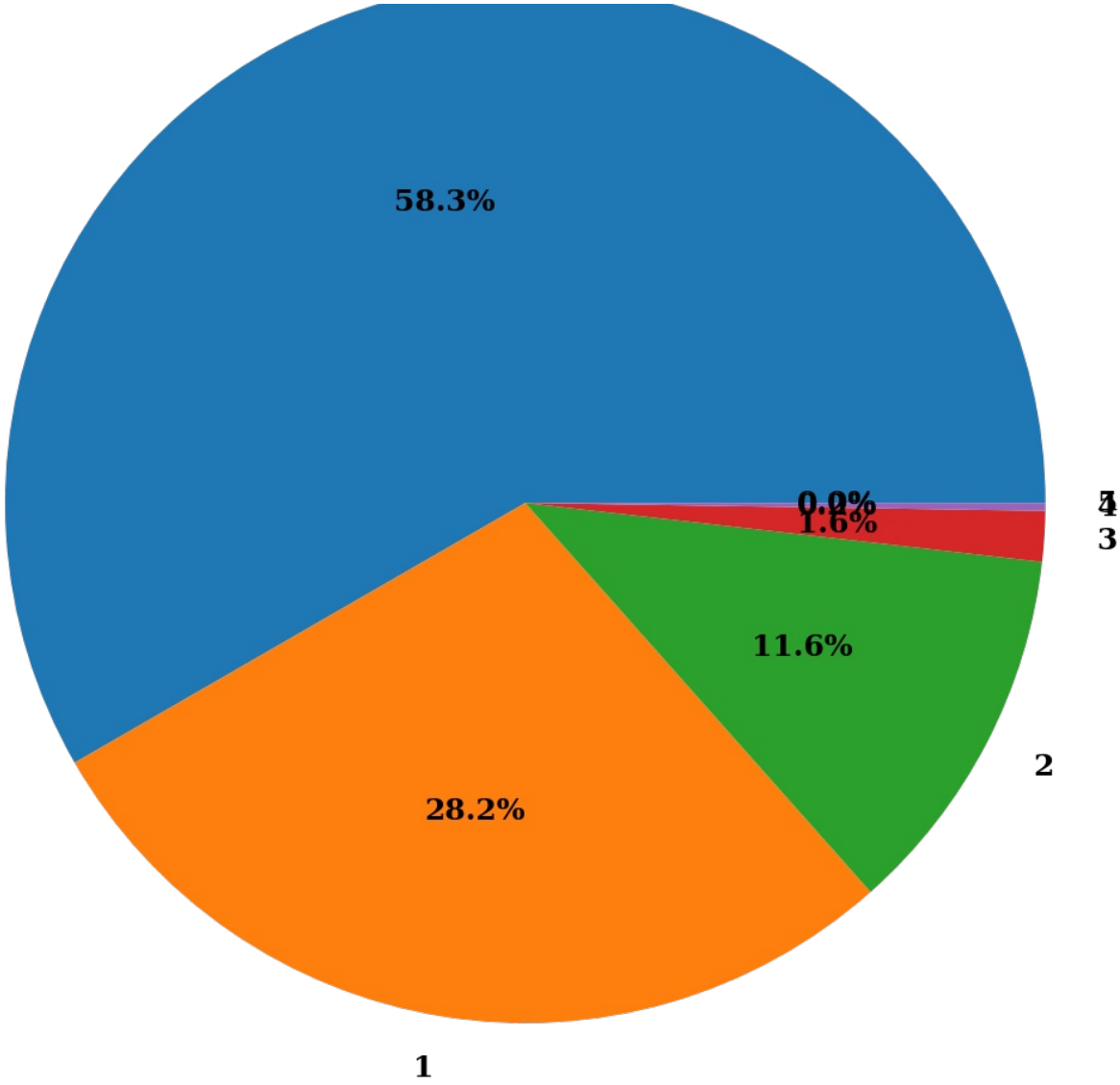

no_of_previous_cancellations
0
99.2%


no_of_previous_bookings_not_canceled
0
97.4%

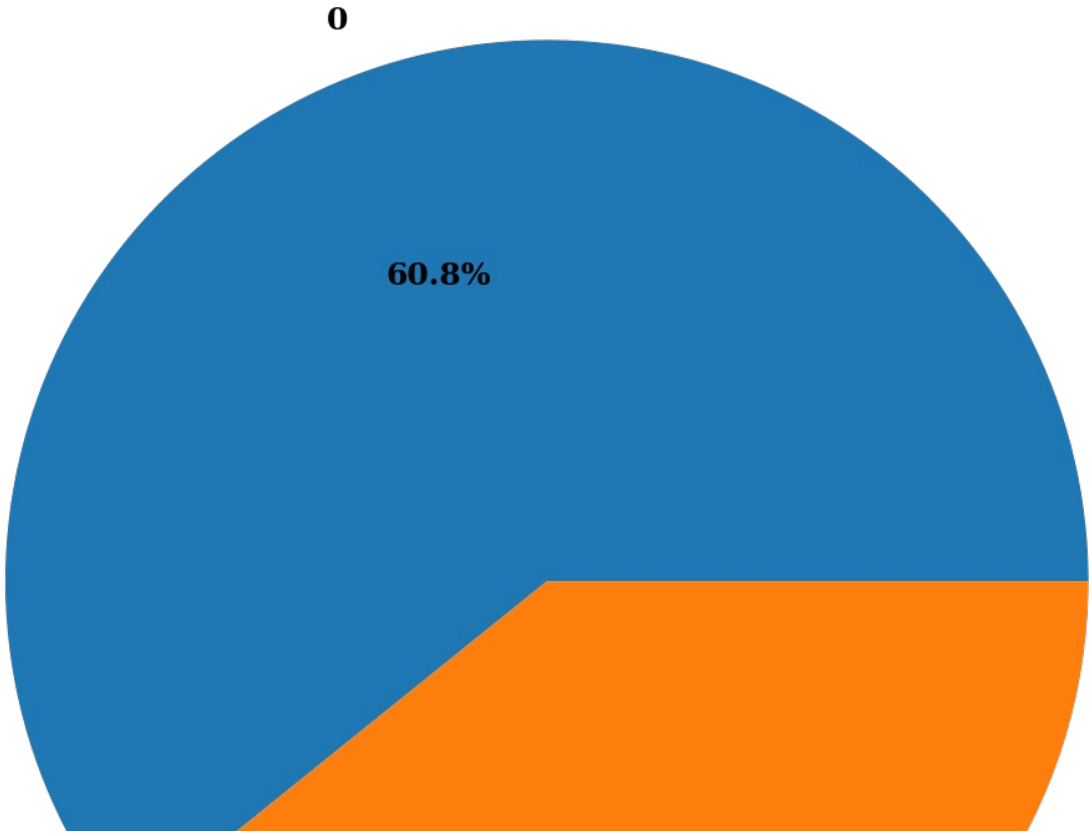
avg_price_per_room

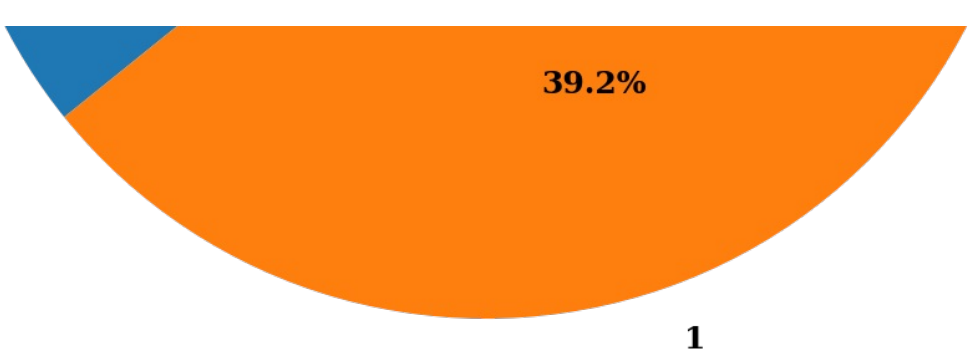


no_of_special_requests

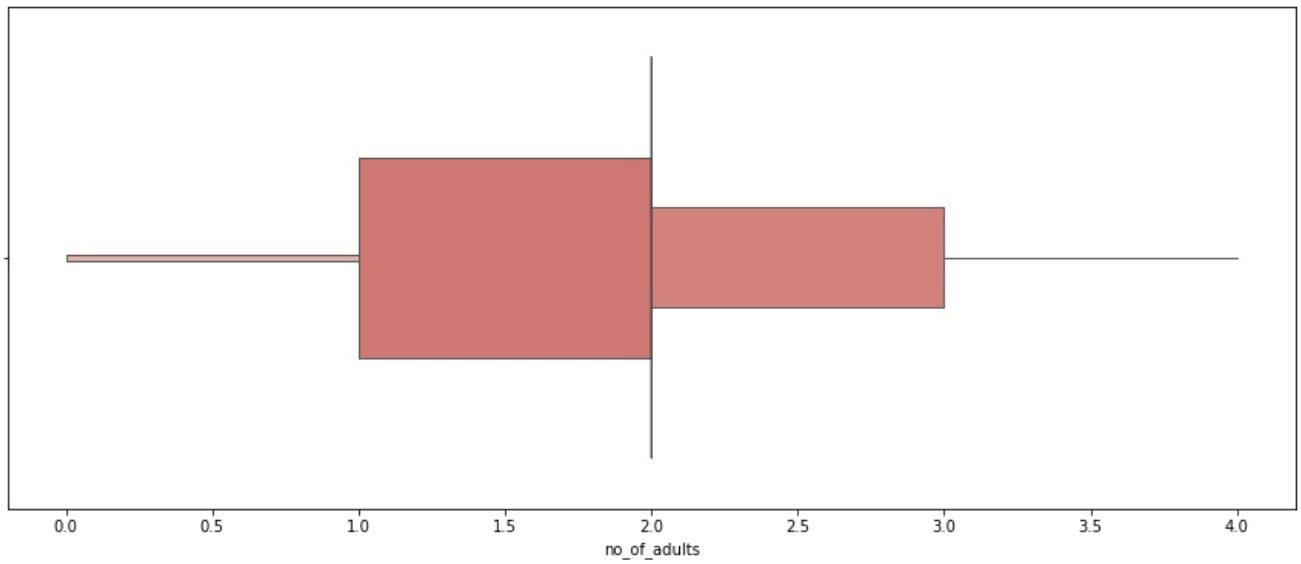


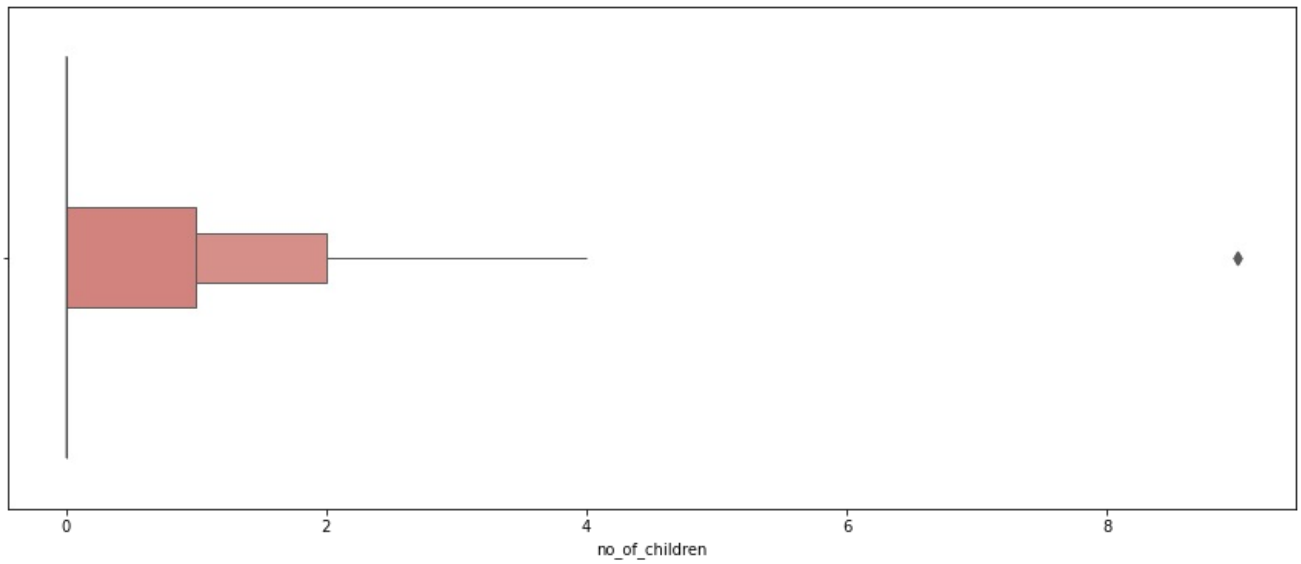
booking_status

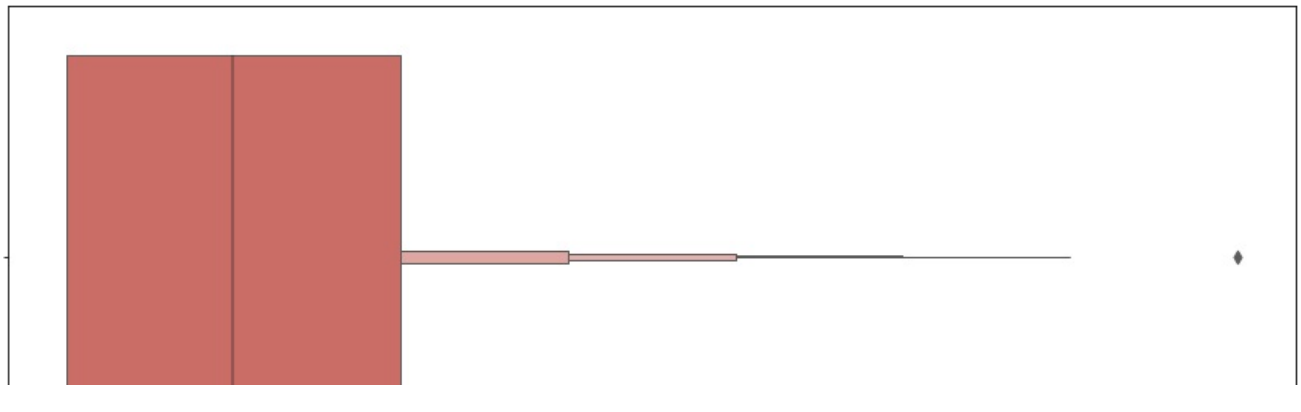


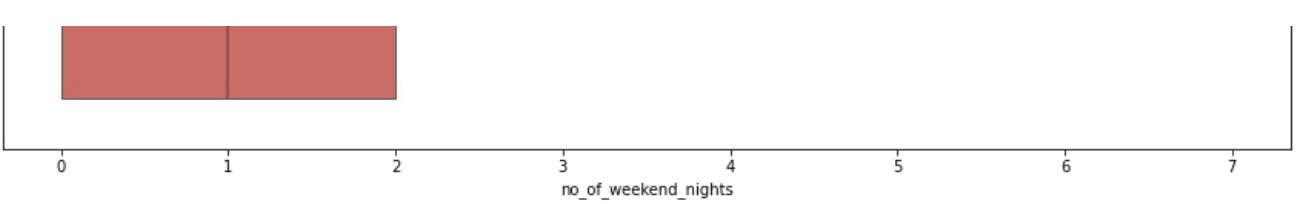


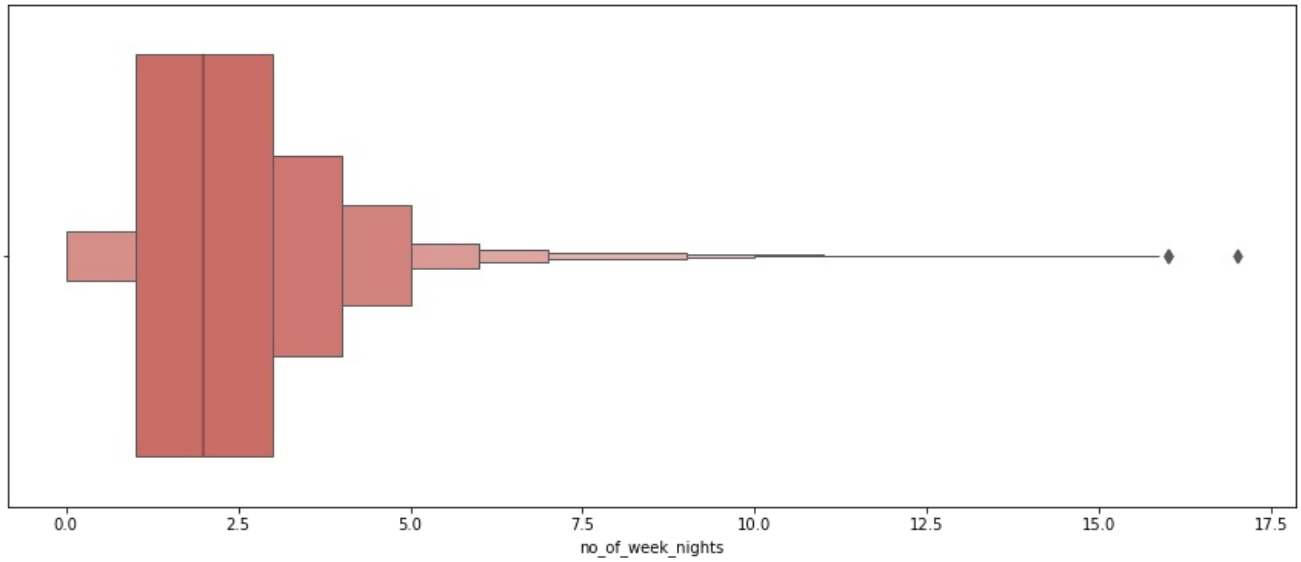
In [17]:
```
for i in df:
    plt.figure(figsize=(15,6))
    sns.boxenplot(x =df[i], data = df, palette = 'hls')
    plt.show()
```

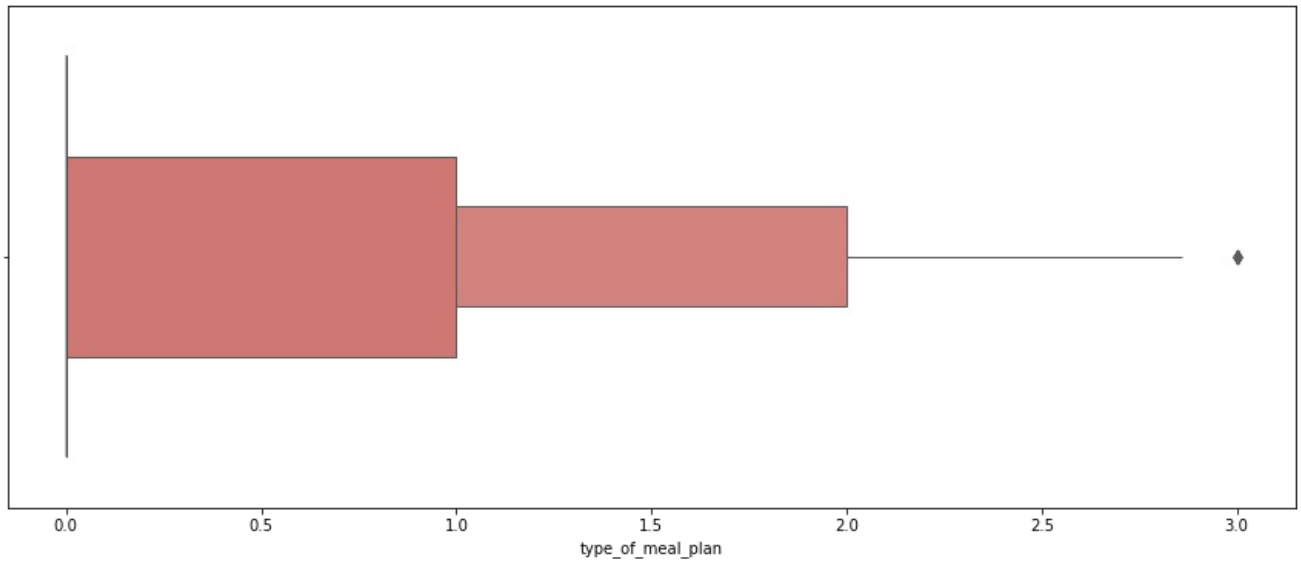


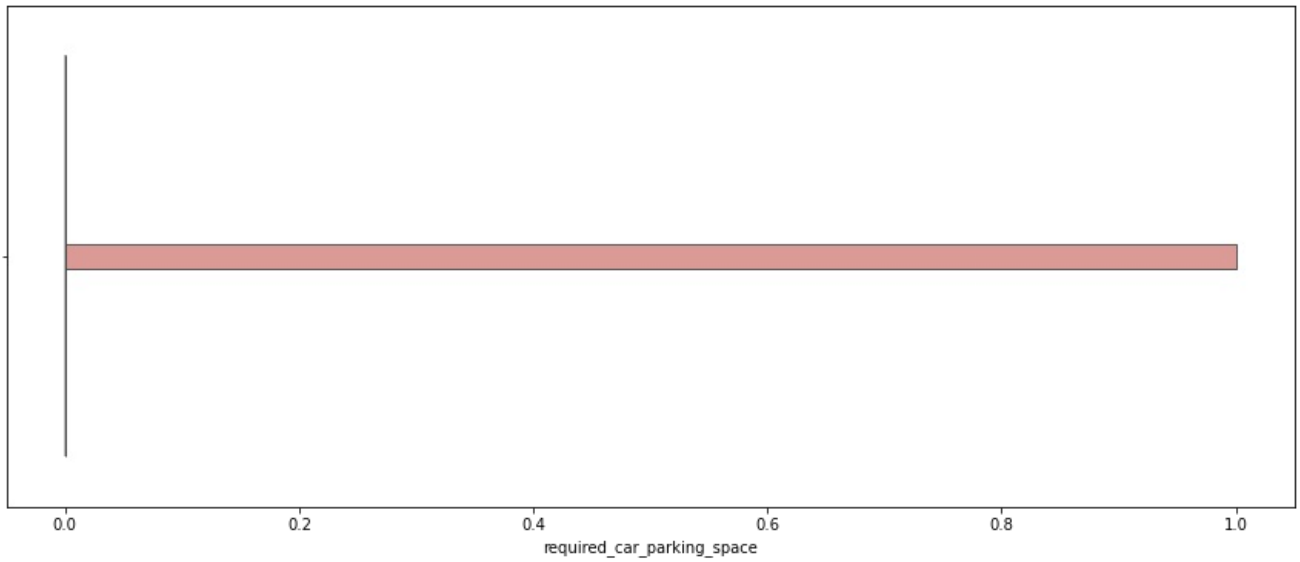


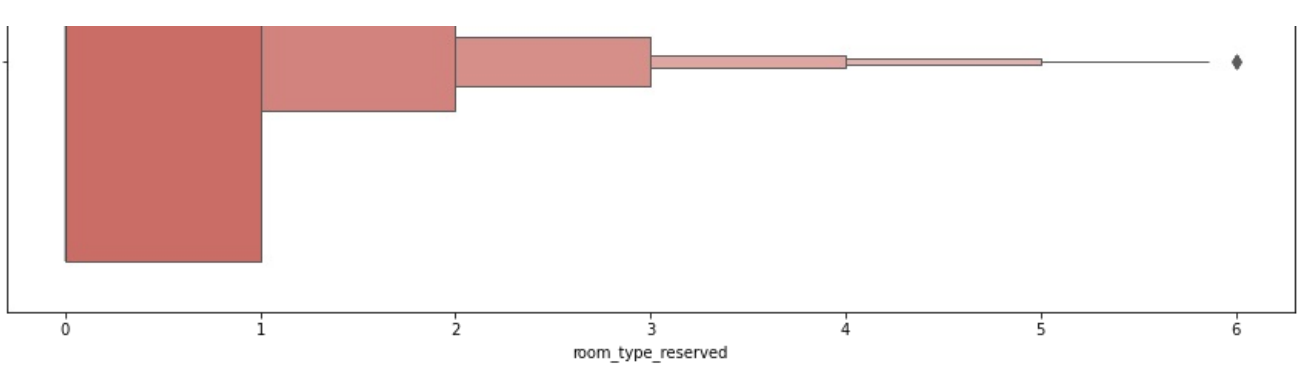


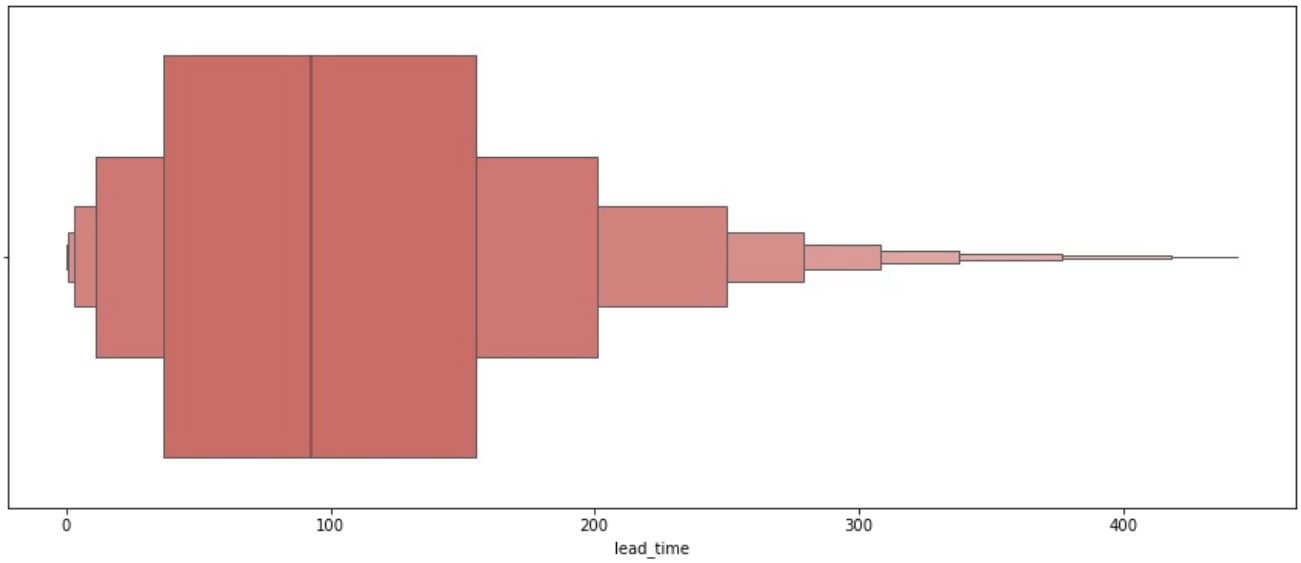

0
1
2
3
4
5
6
7
no_of_weekend_nights

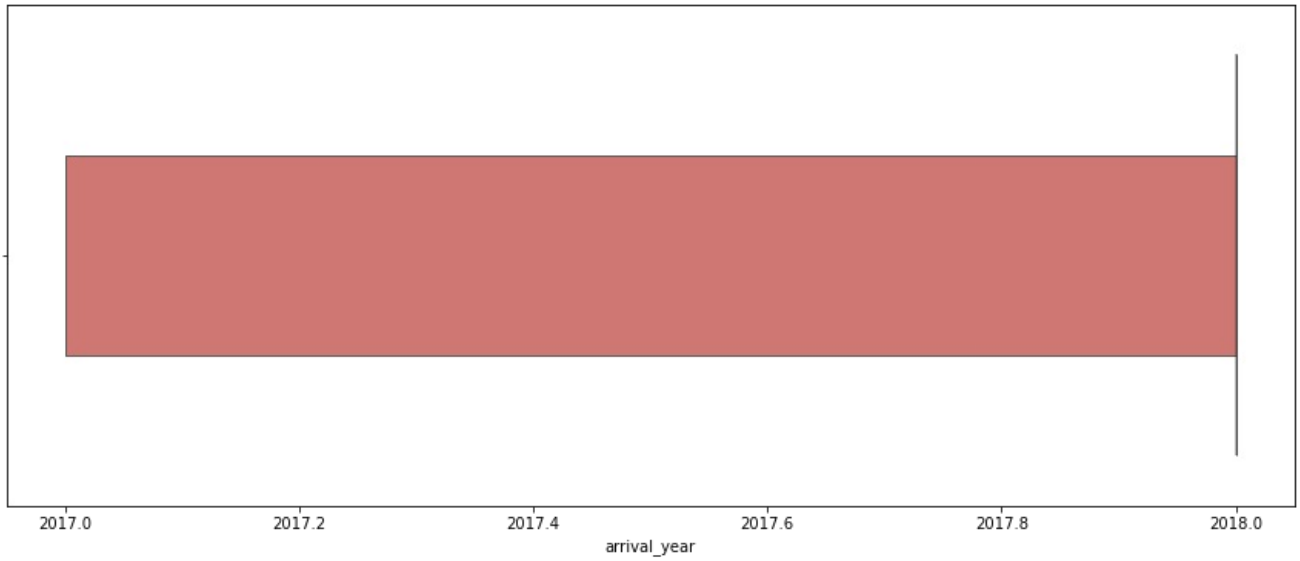

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
no_of_week_nights

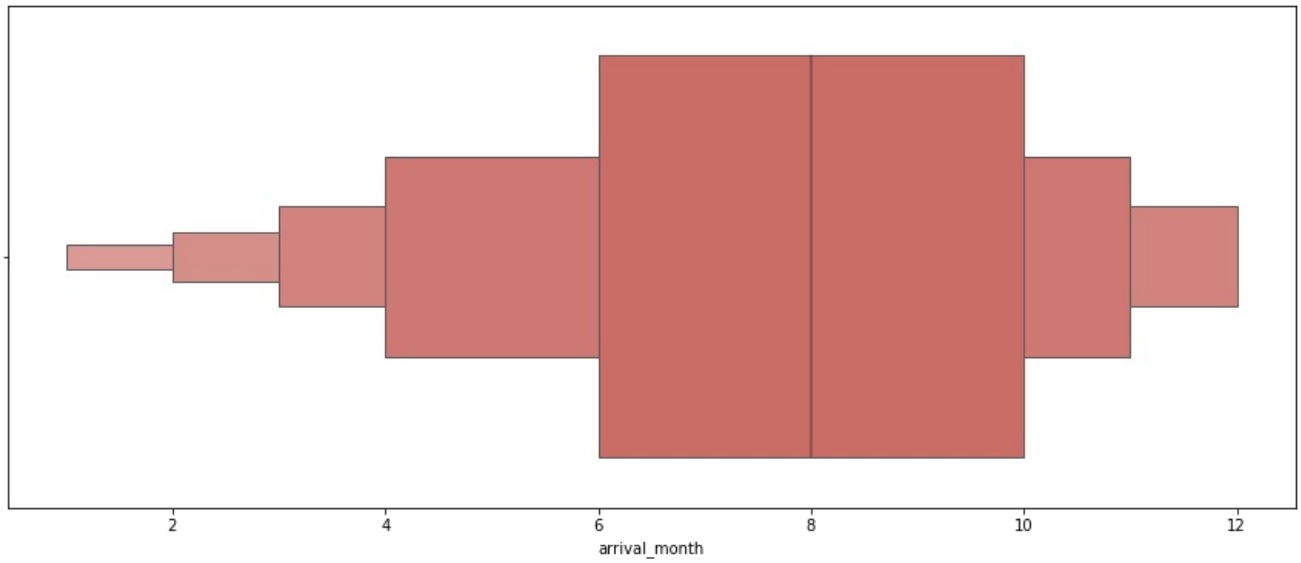

0.0
0.5
1.0
1.5
2.0
2.5
3.0
type_of_meal_plan

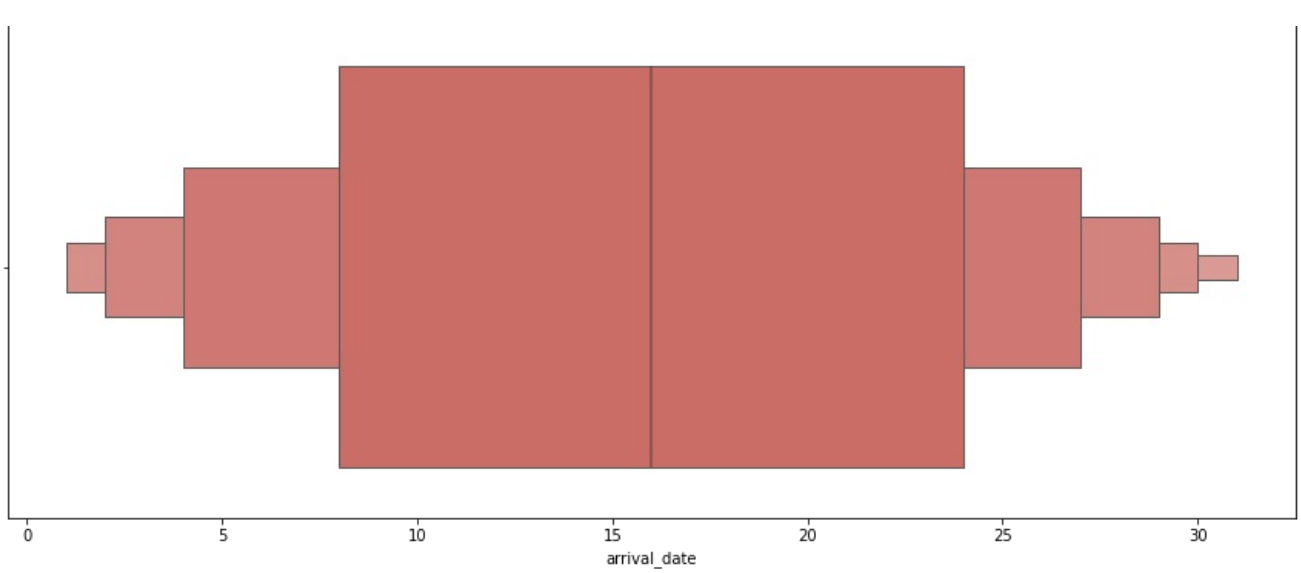

0.0
0.2
0.4
0.6
0.8
1.0
required_car_parking_space

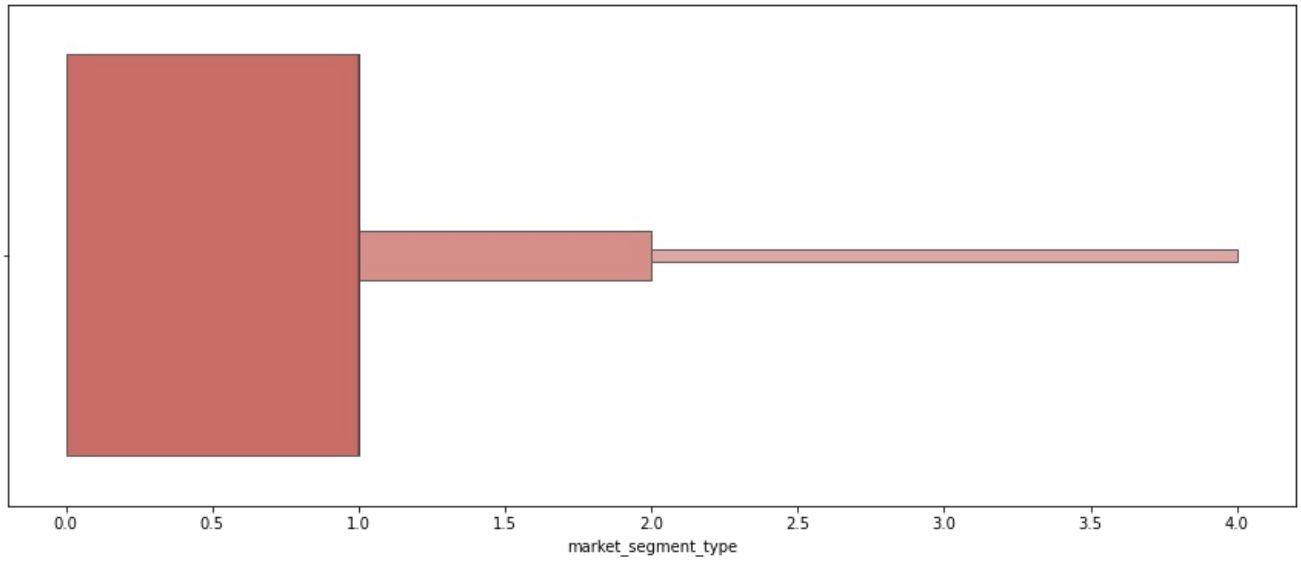

0
1
2
3
4
5
6
room_type_reserved

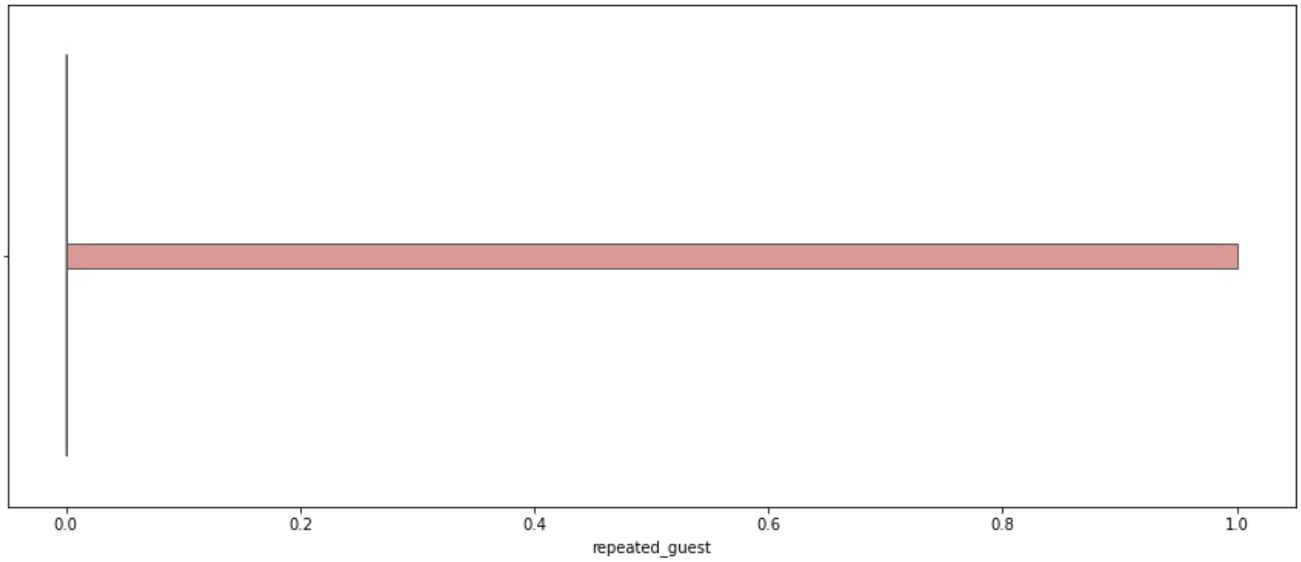

0
100
200
300
400
lead_time

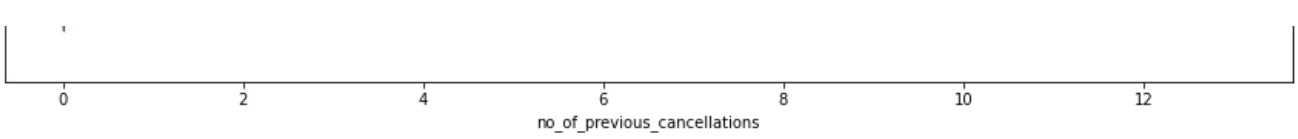

2017.0
2017.2
2017.4
2017.6
2017.8
2018.0
arrival_year

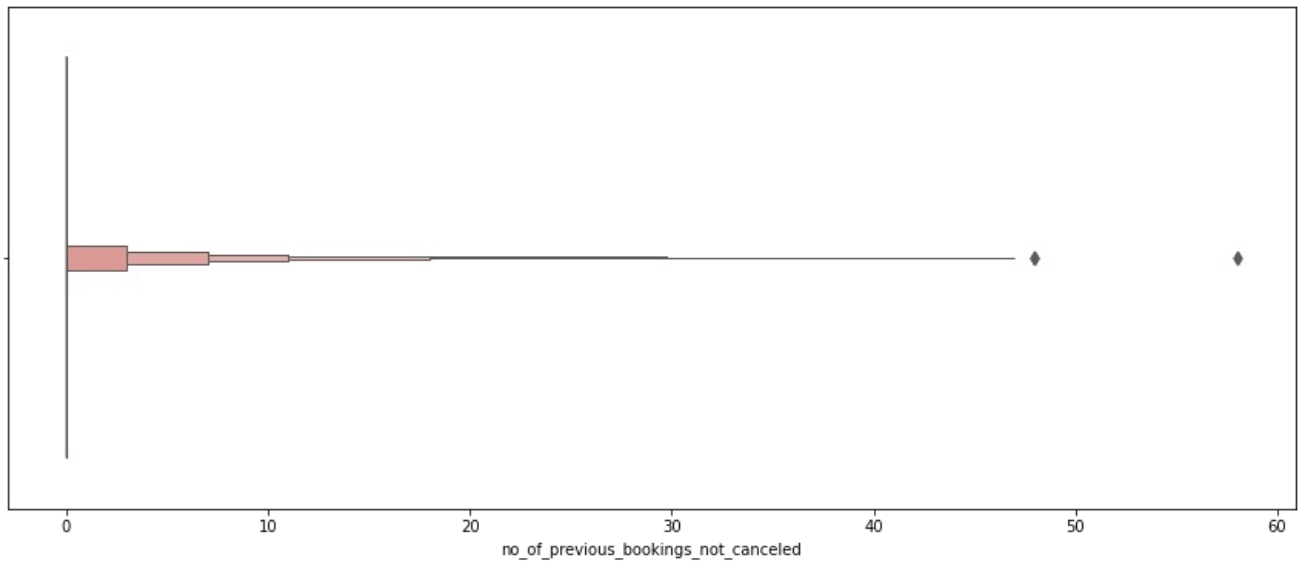

2
4
6
8
10
12
arrival_month

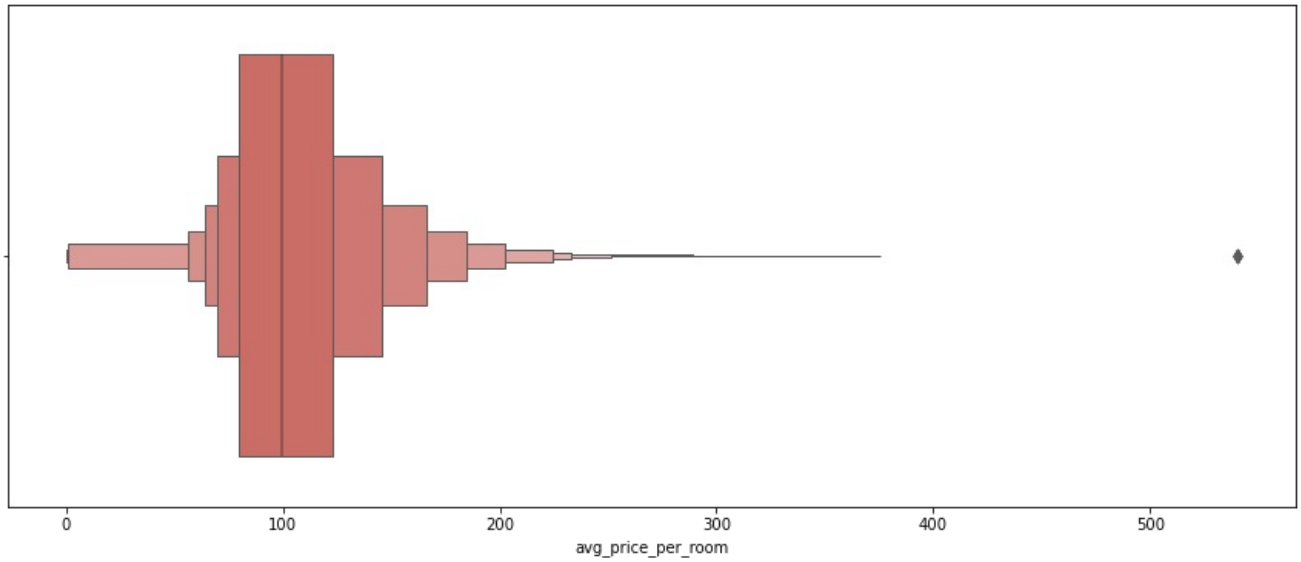

0
5
10
15
20
25
30
arrival_date

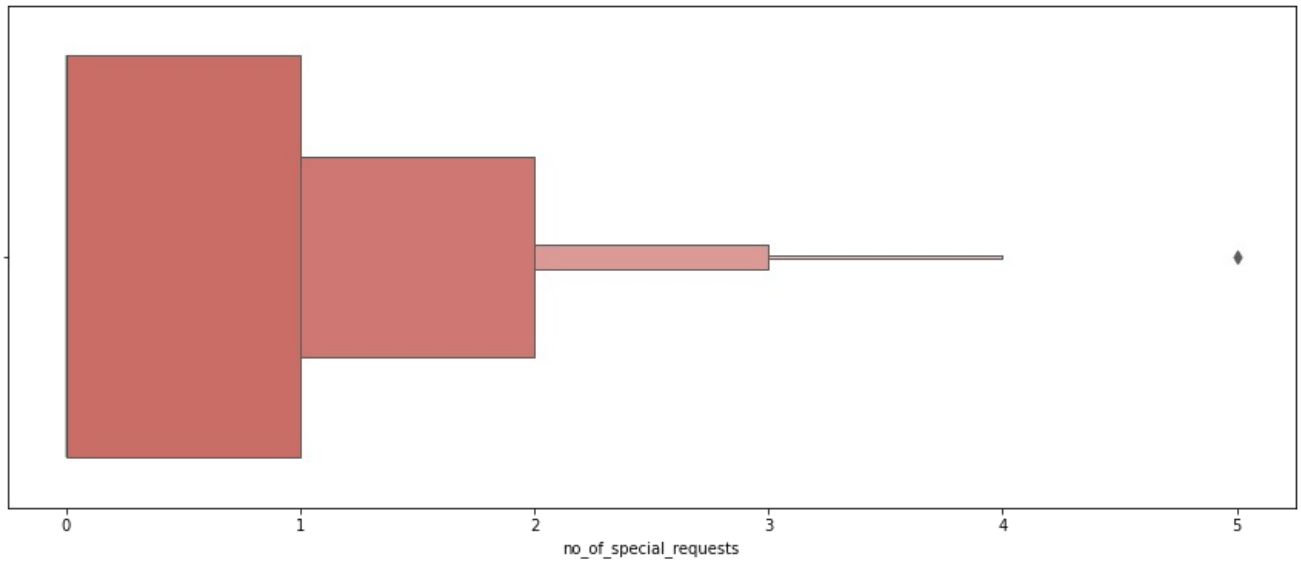

0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
market_segment_type

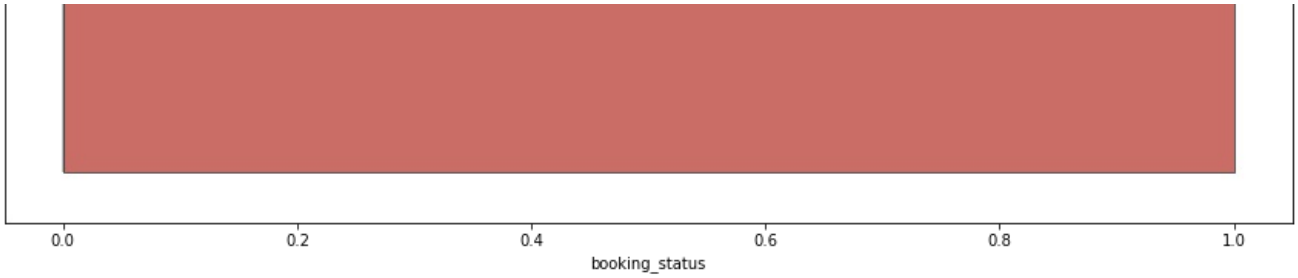

0.0
0.2
0.4
0.6
0.8
1.0
repeated_guest


0
2
4
6
8
10
12
no_of_previous_cancellations

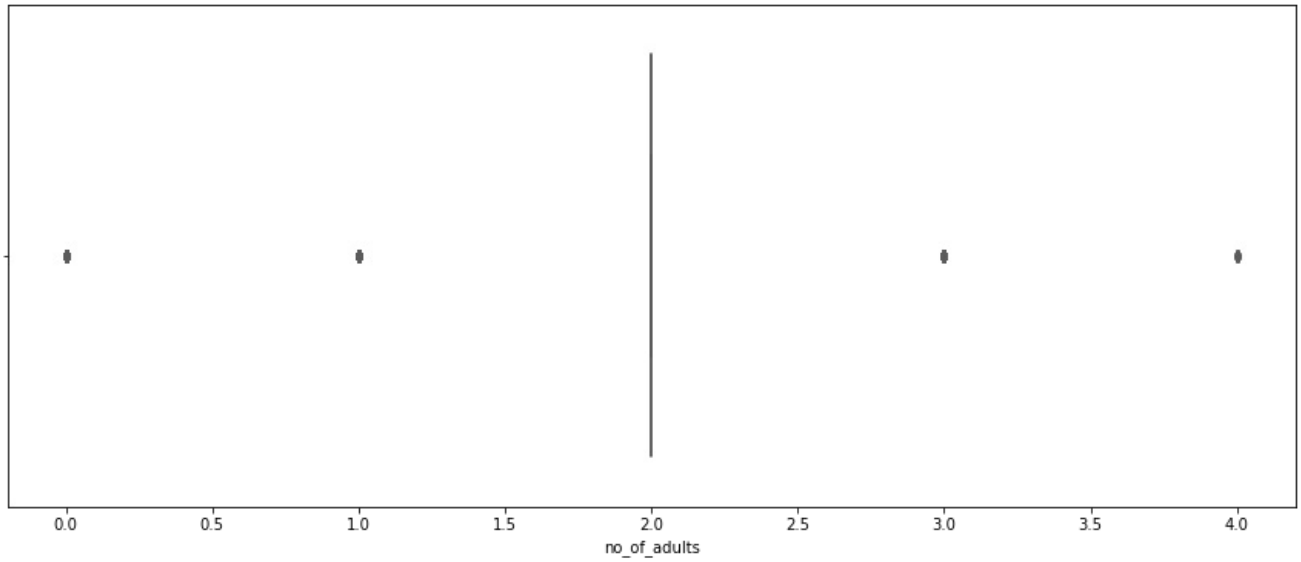

0
10
20
30
40
50
60
no_of_previous_bookings_not_canceled

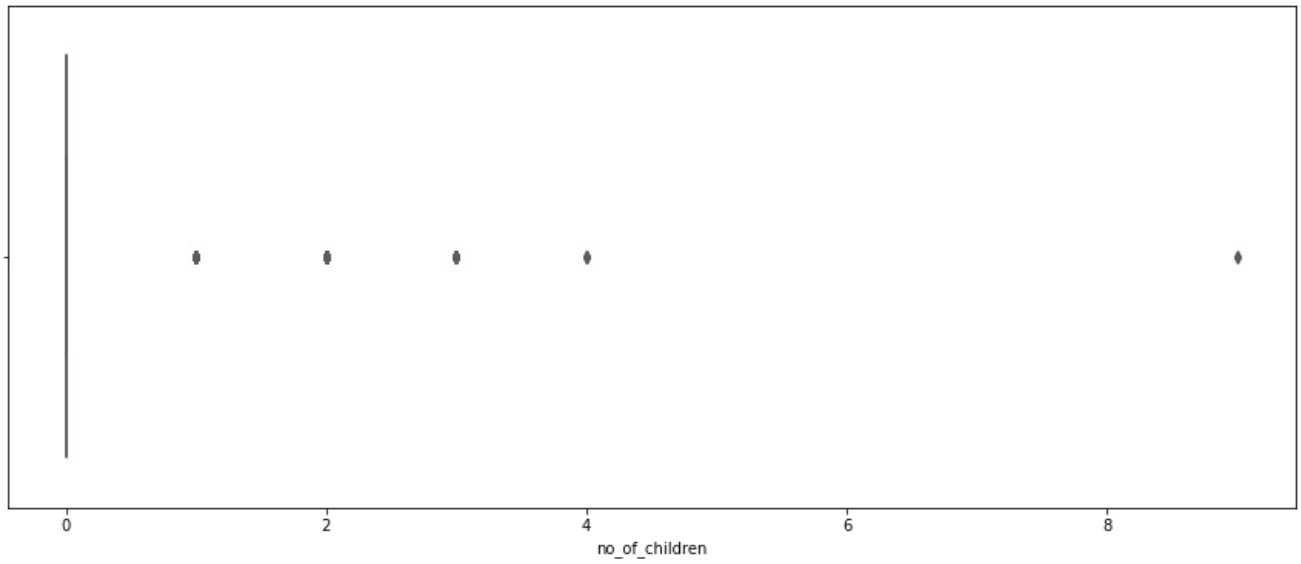

0
100
200
300
400
500
avg_price_per_room

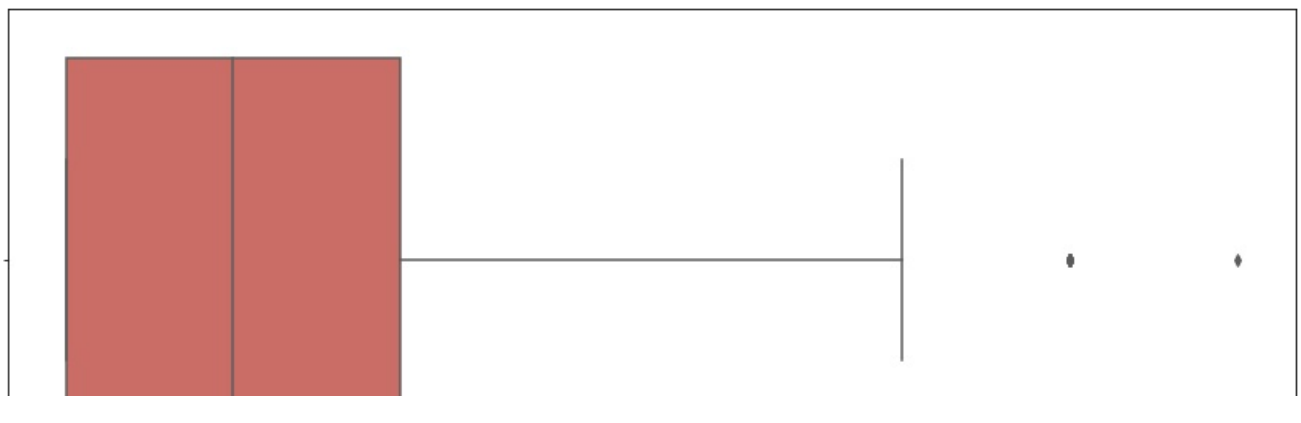

0
1
2
3
4
5
no_of_special_requests

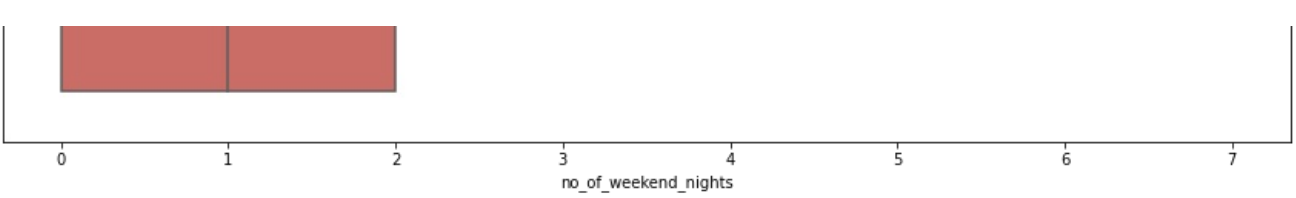


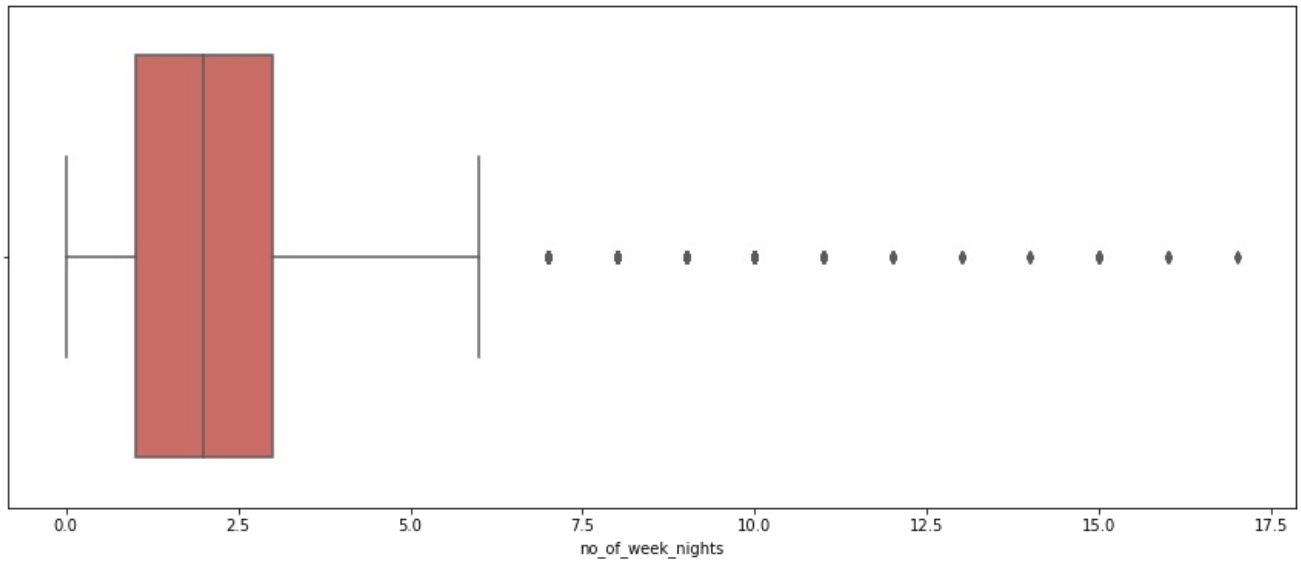
In [18]:
```
for i in df:
    plt.figure(figsize=(15,6))
    sns.boxplot( x=df[i], data = df, palette = 'hls')
    plt.show()
```

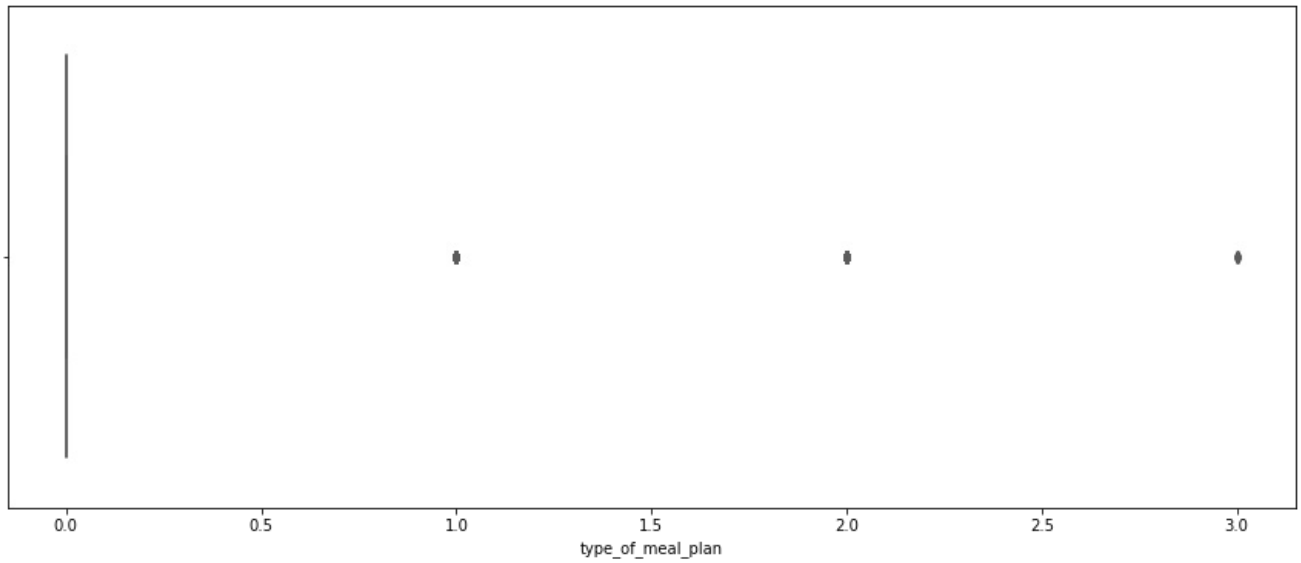


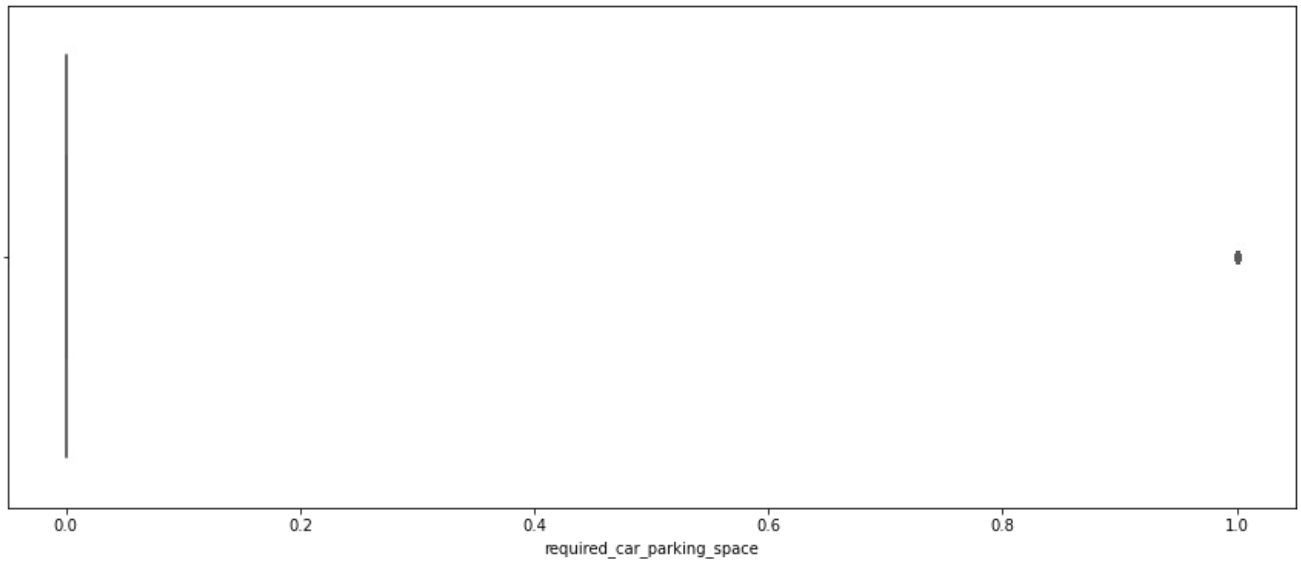


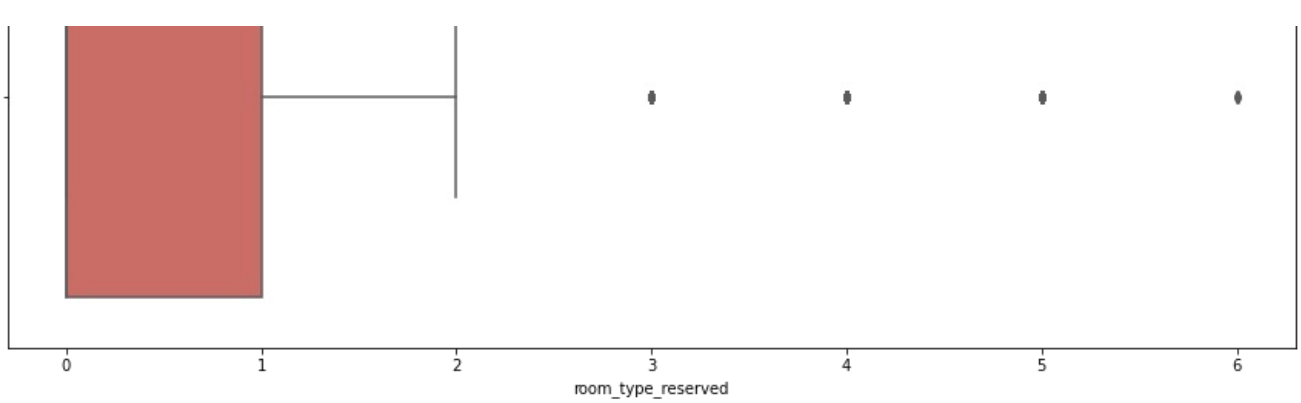


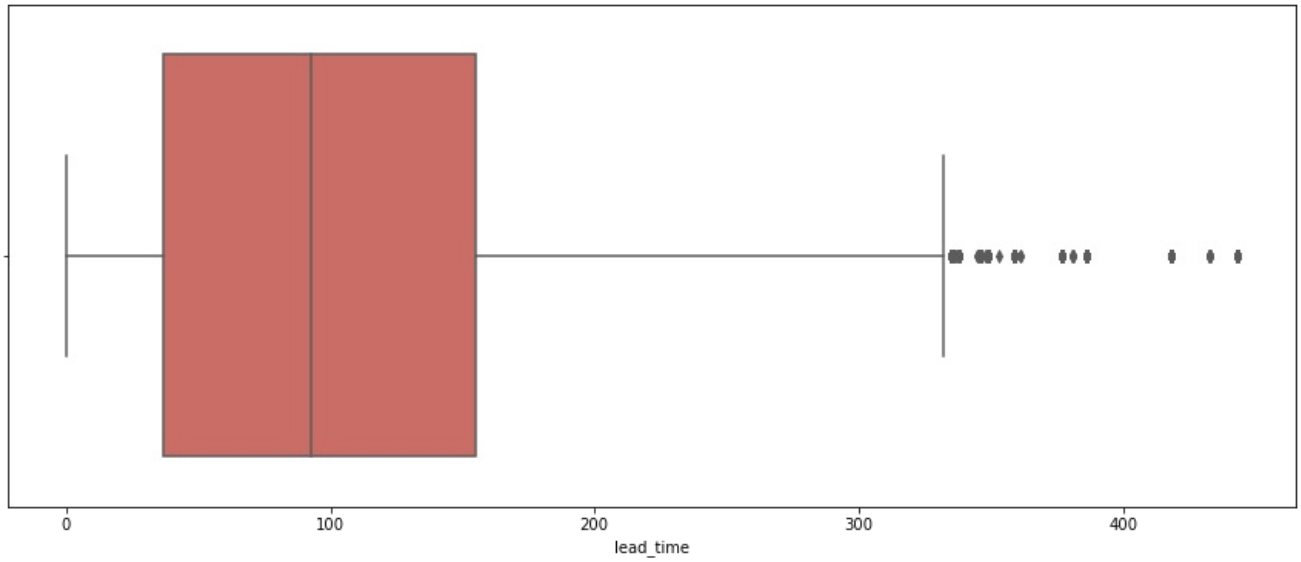

0
1
2
3
4
5
6
7
no_of_weekend_nights

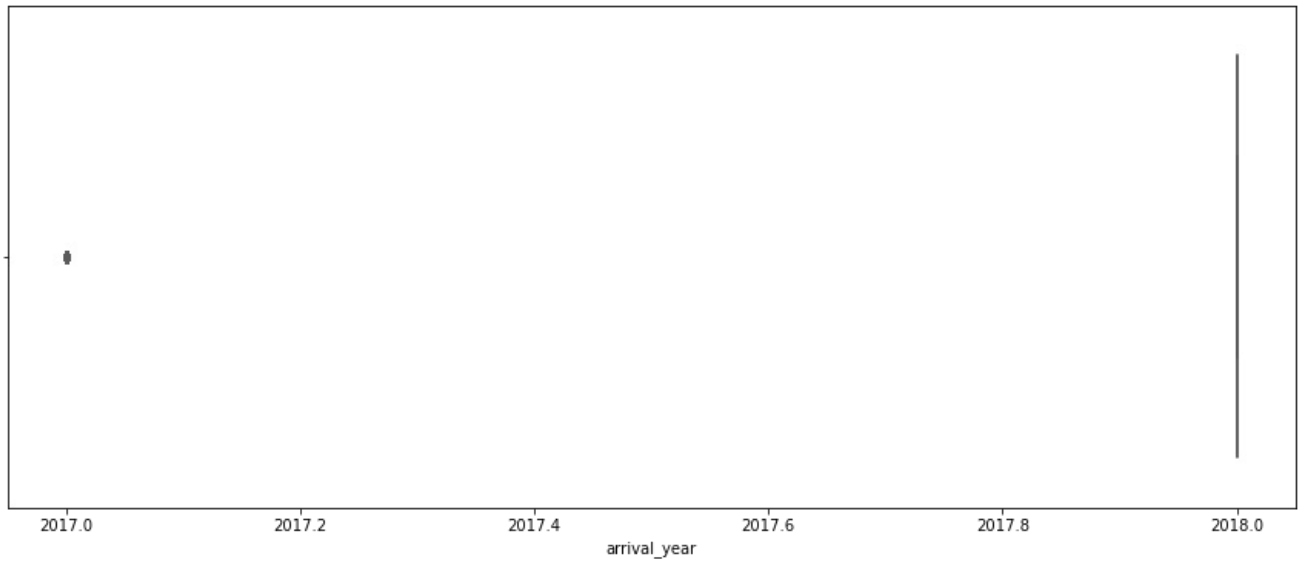

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
no_of_week_nights

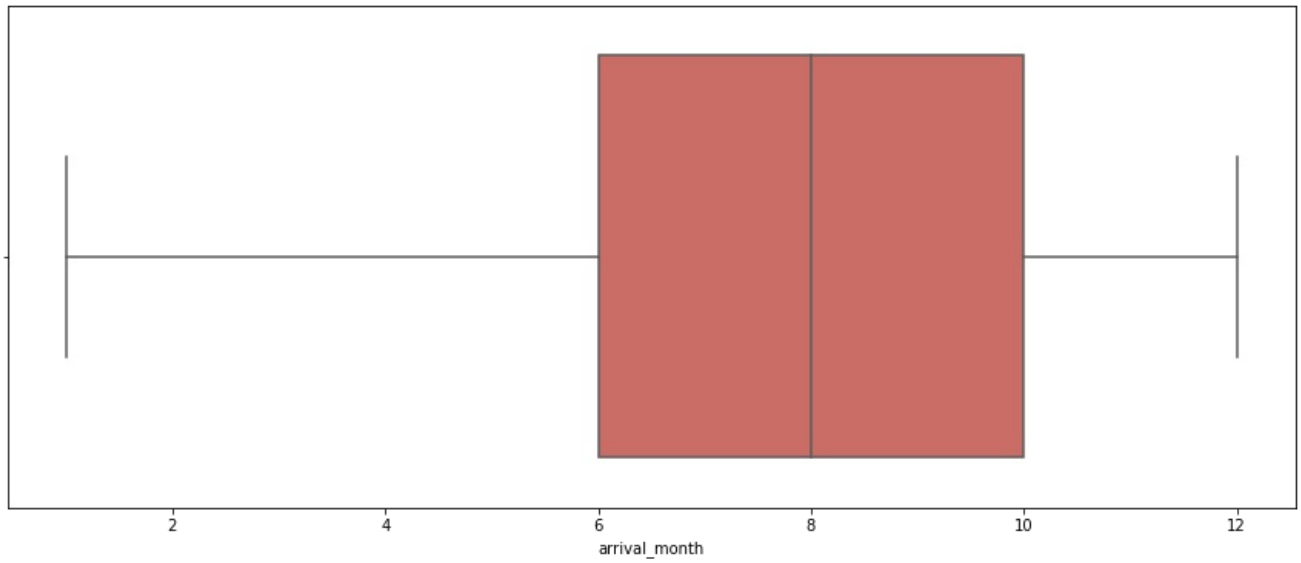

0.0
0.5
1.0
1.5
2.0
2.5
3.0
type_of_meal_plan

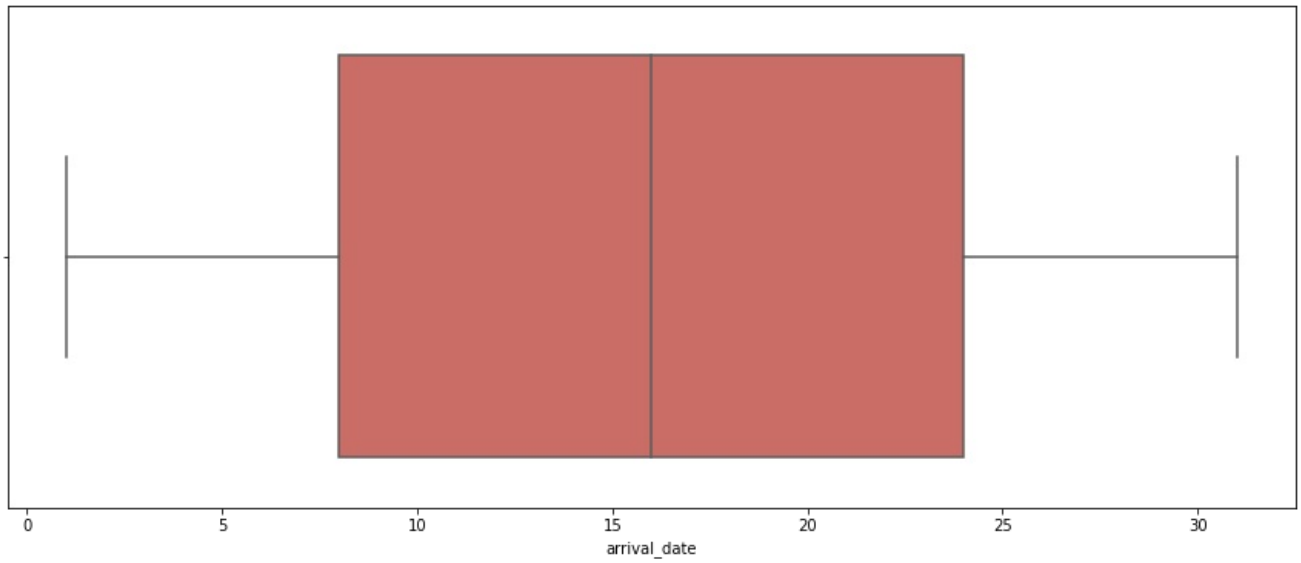

0.0
0.2
0.4
0.6
0.8
1.0
required_car_parking_space

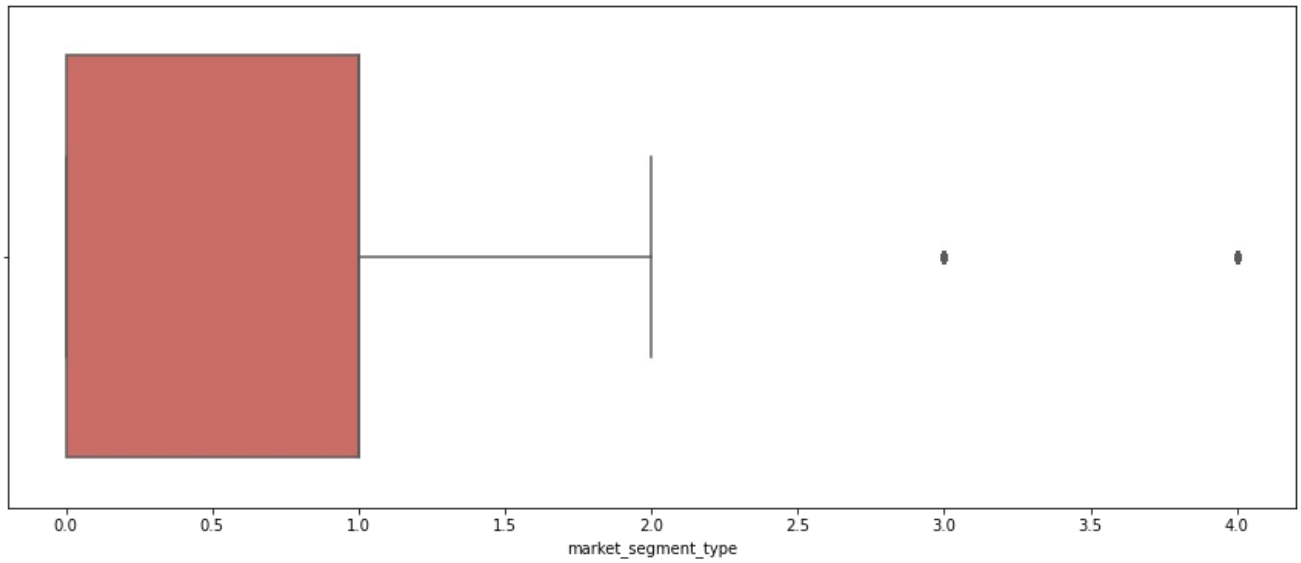

0
1
2
3
4
5
6
room_type_reserved

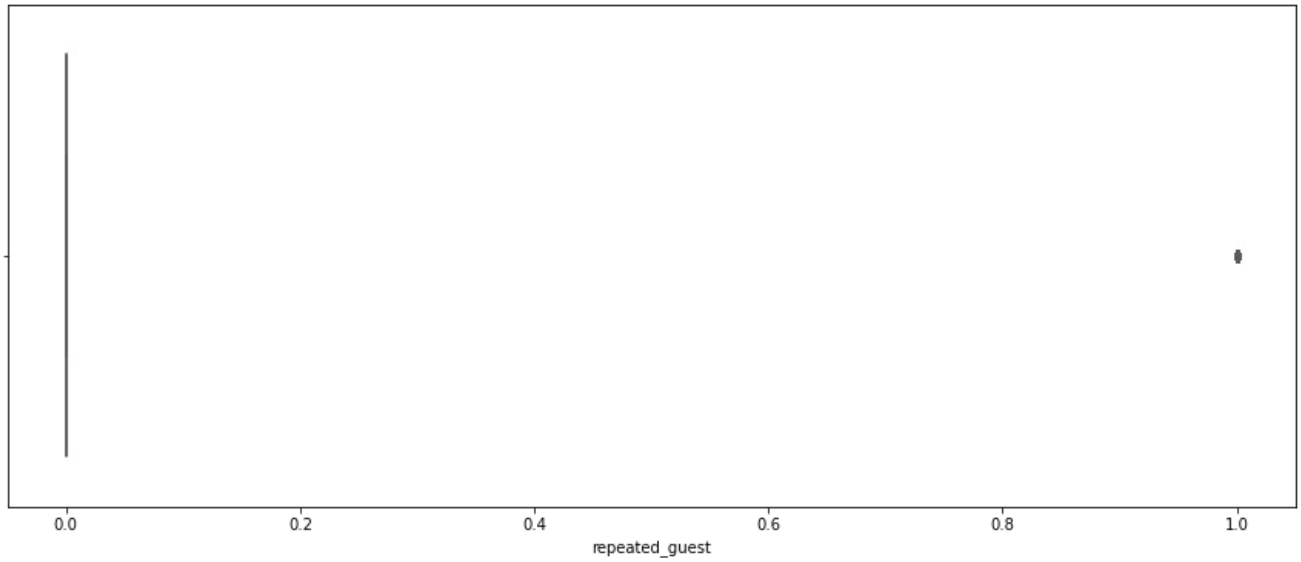

0
100
200
300
400
lead_time

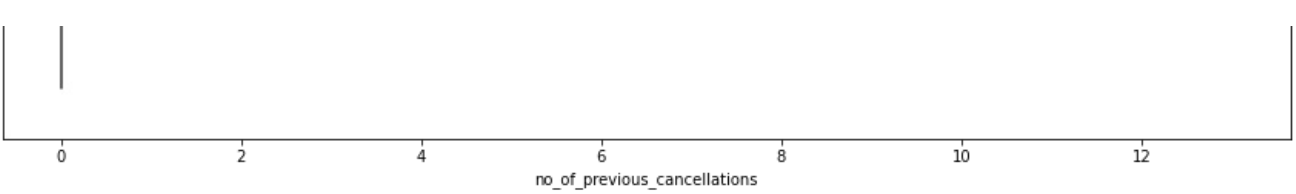

2017.0
2017.2
2017.4
2017.6
2017.8
2018.0
arrival_year

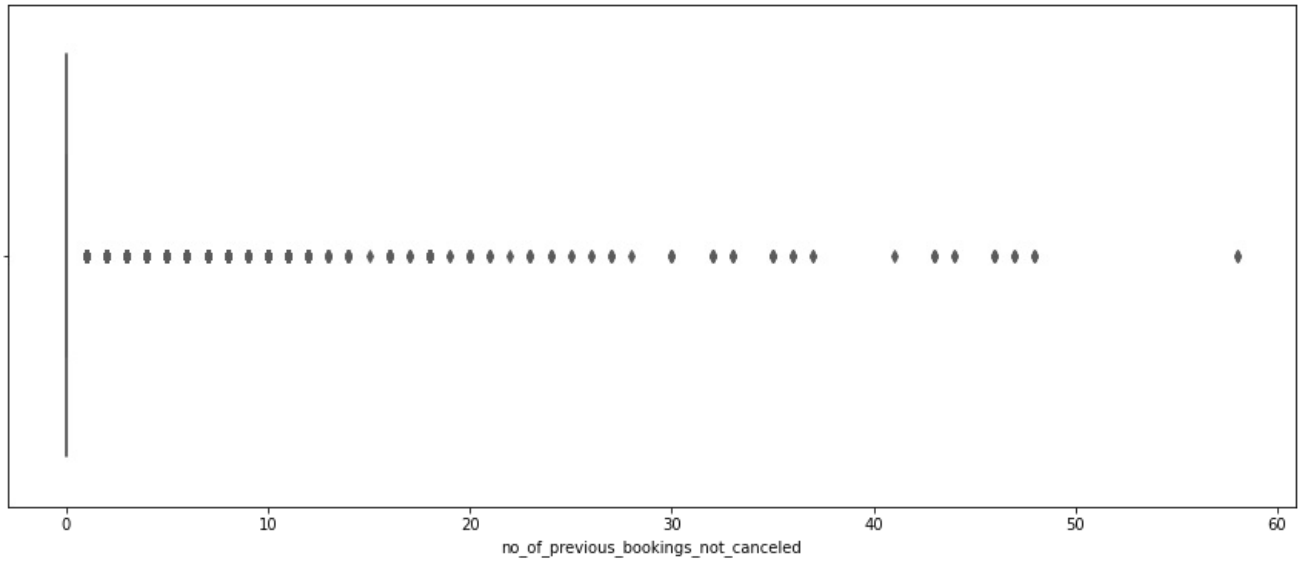

2
4
6
8
10
12
arrival_month

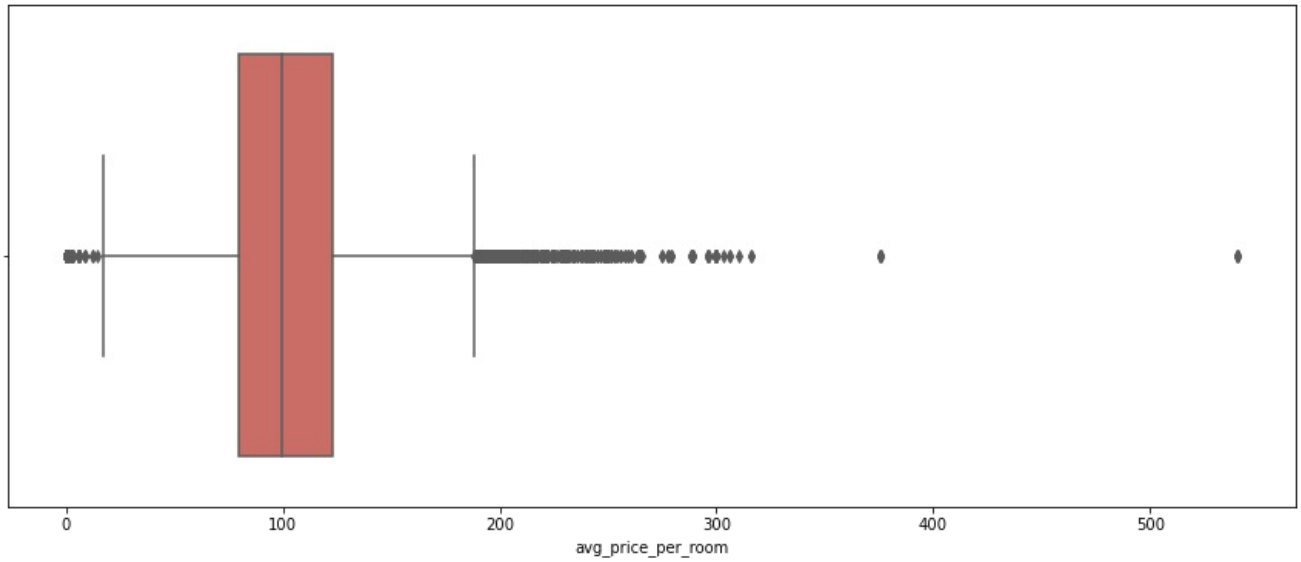

0
5
10
15
20
25
30
arrival_date

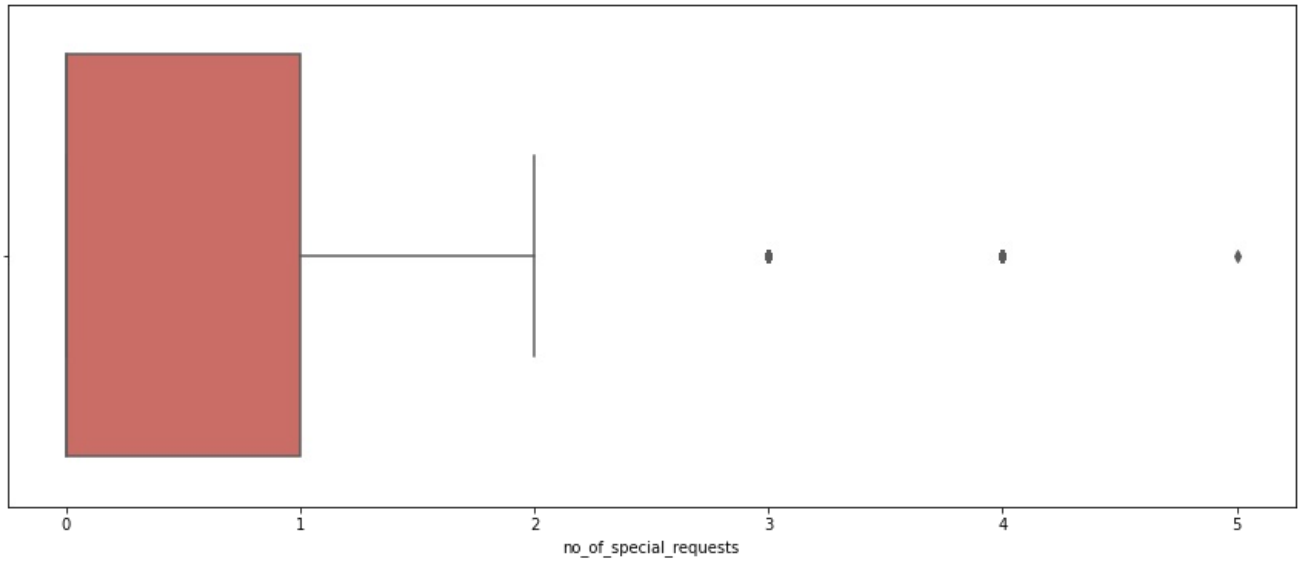

0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
market_segment_type

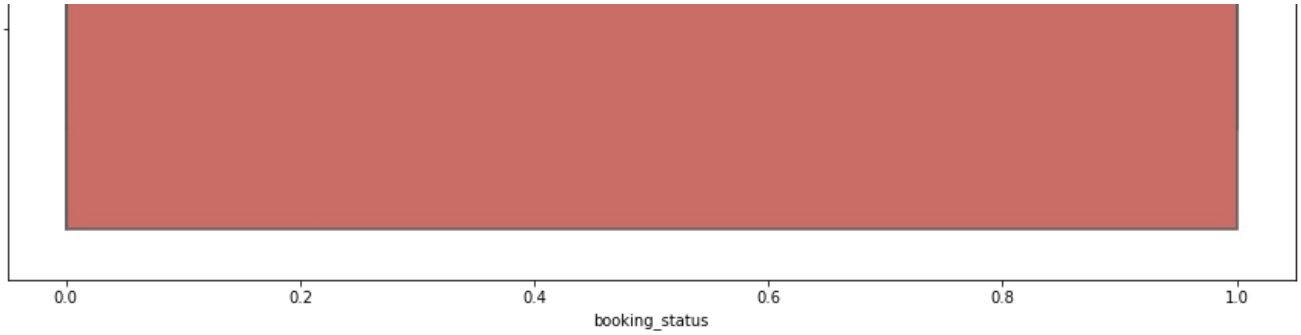

0.0
0.2
0.4
0.6
0.8
1.0
repeated_guest


0
2
4
6
8
10
12
no_of_previous_cancellations

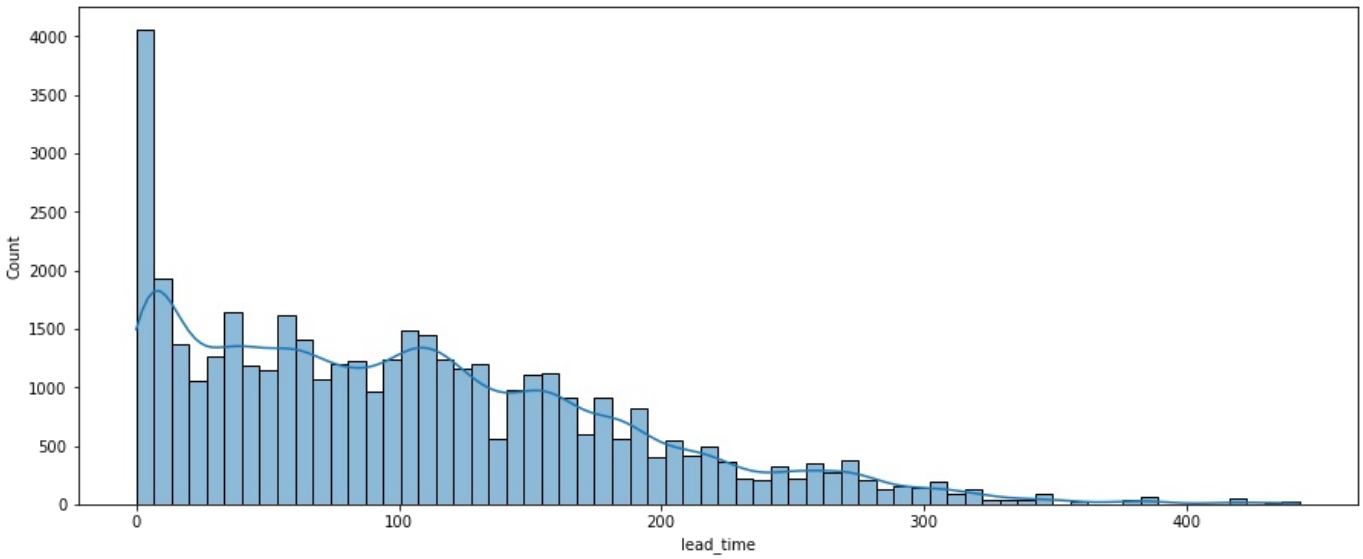

0
10
20
30
40
50
60
no_of_previous_bookings_not_canceled


0
100
200
300
400
500
avg_price_per_room

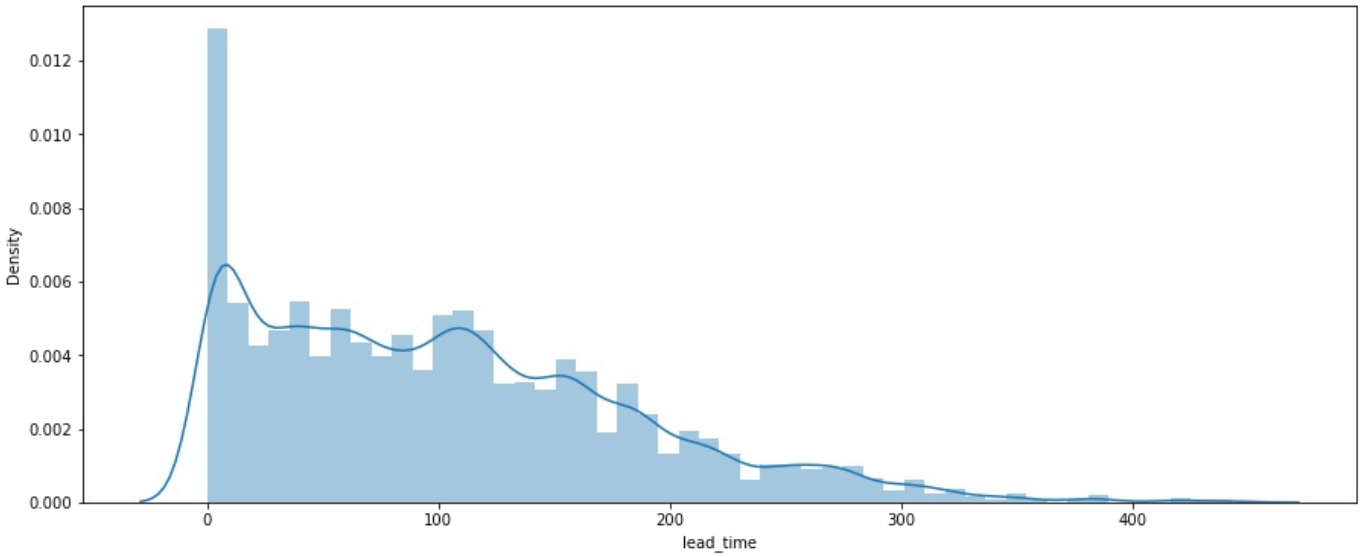

0
1
2
3
4
5
no_of_special_requests



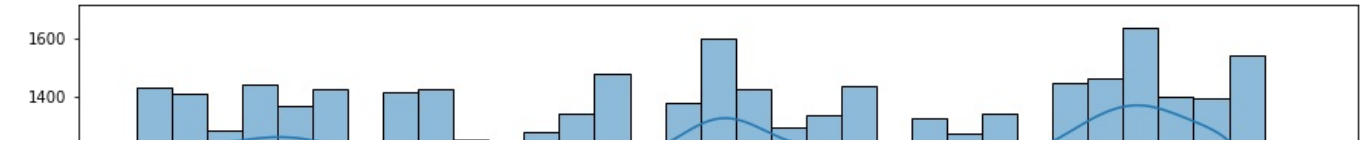
In [19]:
```
plt.figure(figsize=(15,6))
sns.histplot(df['lead_time'],kde = True, palette = 'hls')
plt.show()
```

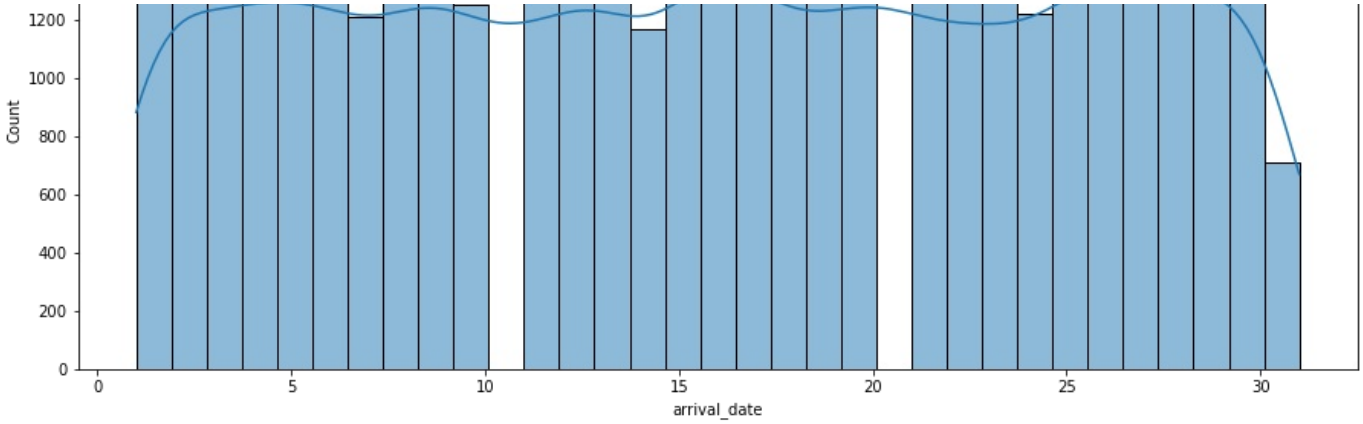


In [20]:
```
plt.figure(figsize=(15,6))
sns.distplot(df['lead_time'],kde = True)
plt.show()
```

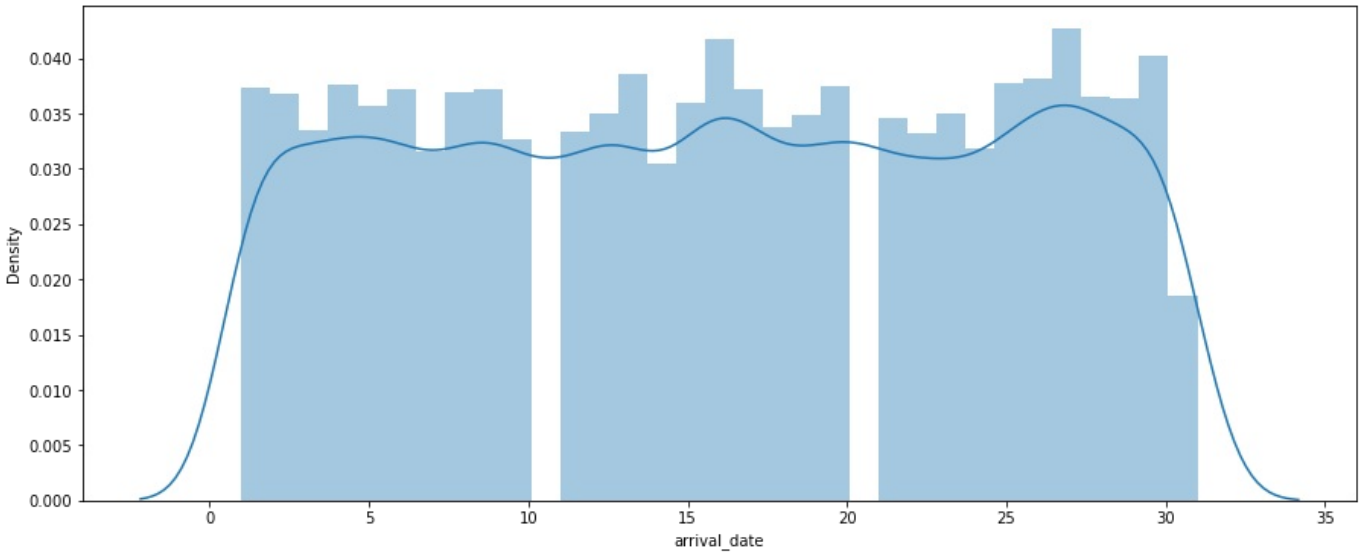


In [21]:
```
plt.figure(figsize=(15,6))
sns.histplot(df['arrival_date'], kde = True, palette = 'hls')
plt.show()
```

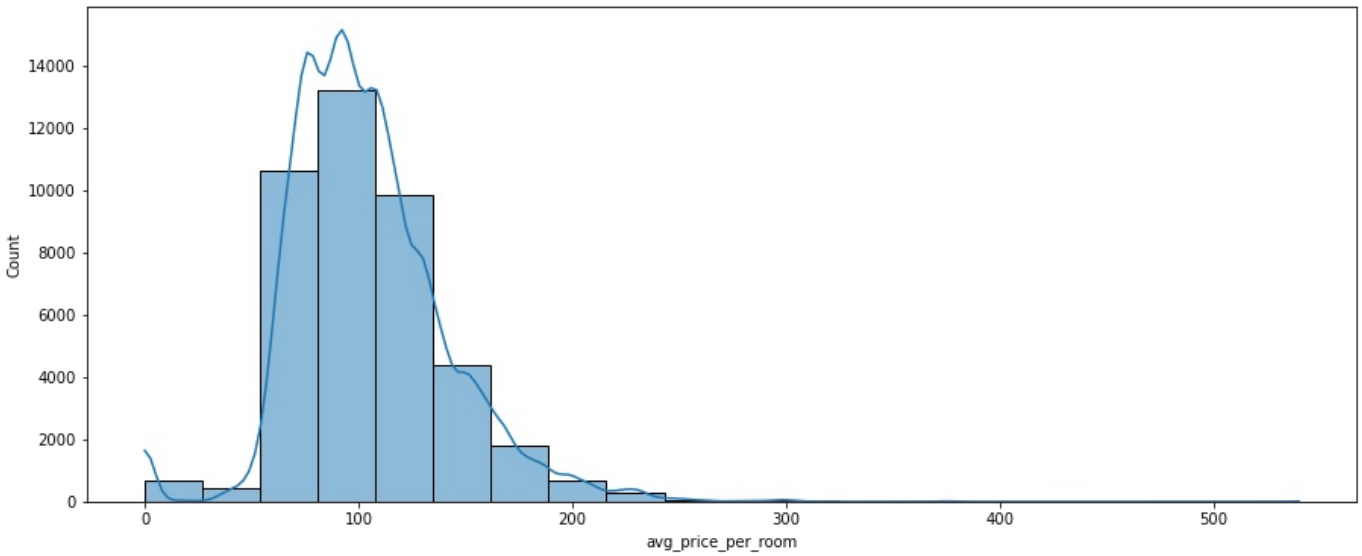




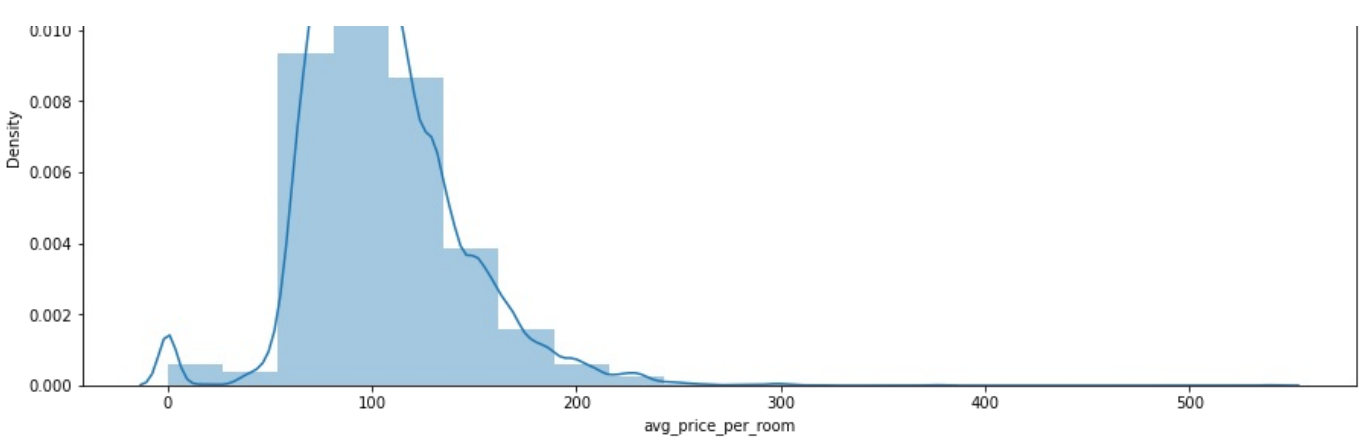
In [22]:
```
plt.figure(figsize=(15,6))
sns.distplot(df['arrival_date'], kde = True)
plt.show()
```



In [23]:
```
plt.figure(figsize=(15,6))
sns.histplot(df['avg_price_per_room'], kde = True, bins = 20, palette = 'hls')
plt.show()
```



In [24]:
```
plt.figure(figsize=(15,6))
sns.distplot(df['avg_price_per_room'], kde = True, bins = 20)
plt.show()
```



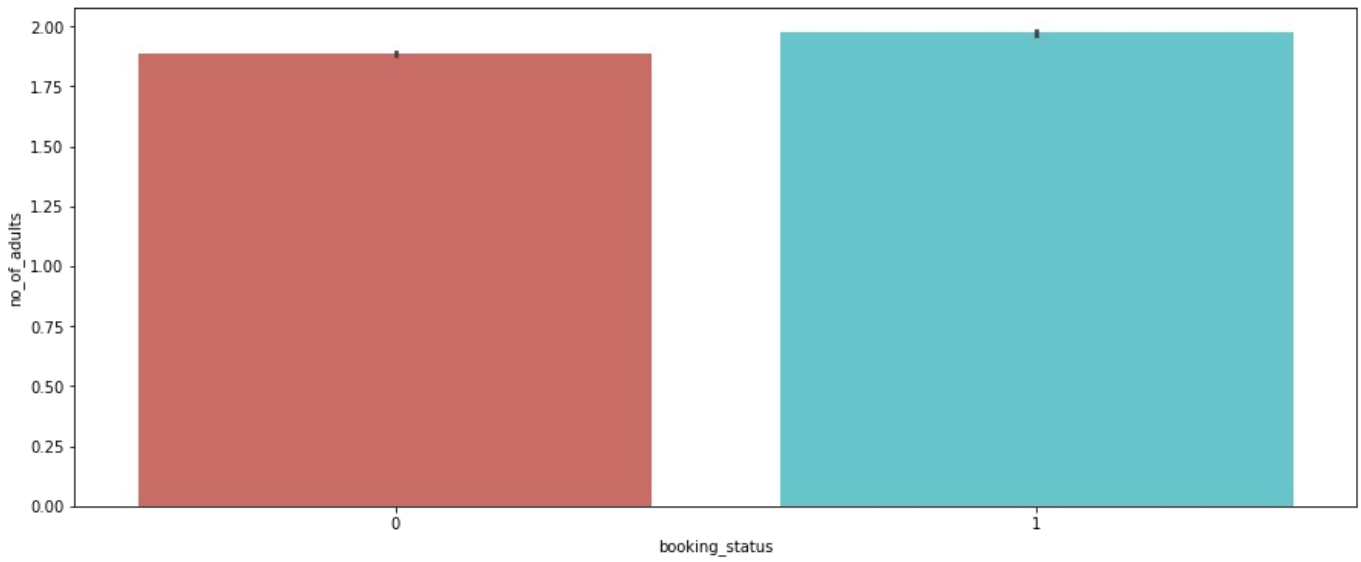
In [25]:
```
df.head()
```

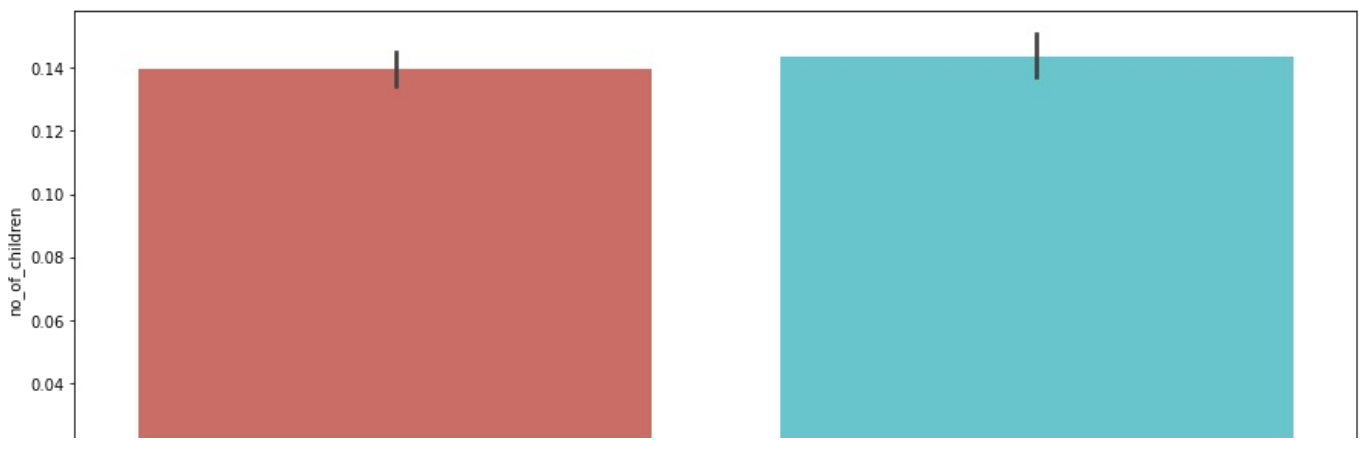
Out[25]:

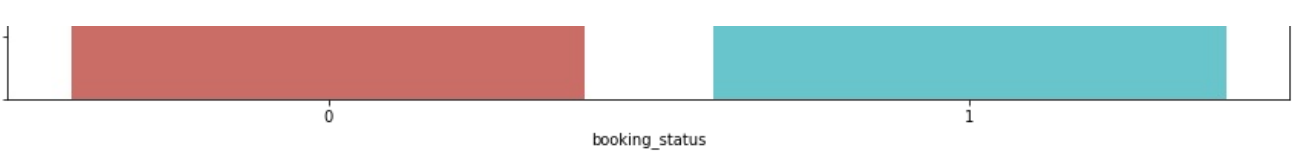
	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_type_reserved
0	2	0	0	2	1	0	0
1	2	0	1	2	0	0	0
2	2	0	0	1	0	0	0
3	1	0	0	2	1	0	0
4	2	0	1	0	0	0	0

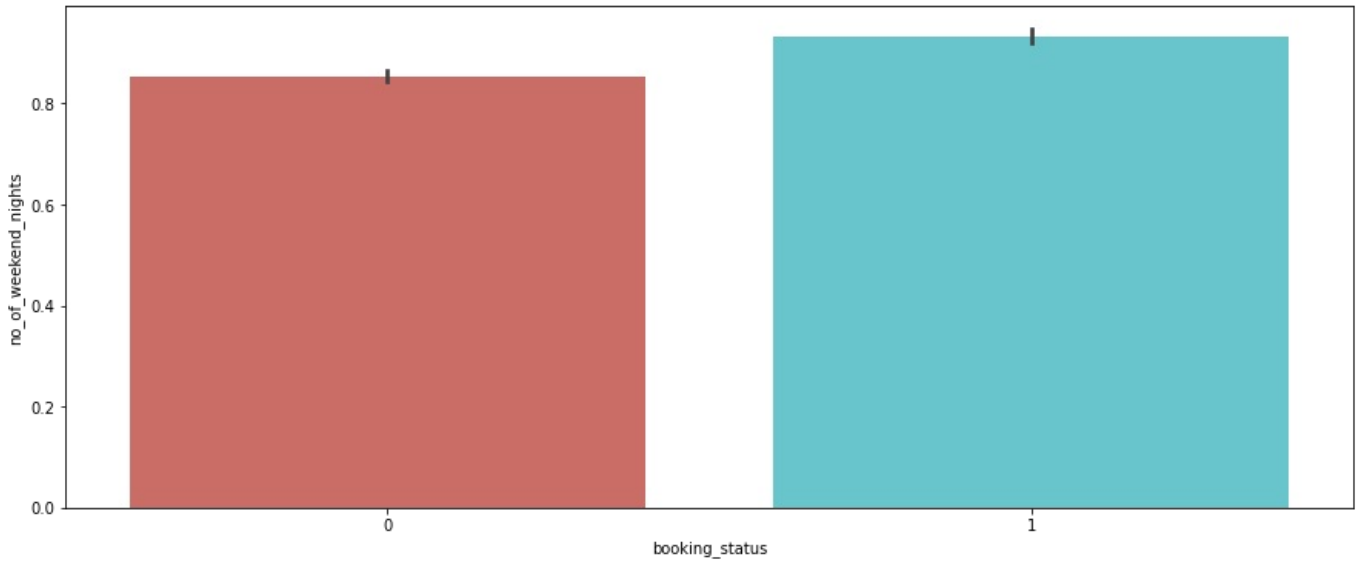
In [26]:
```
df2 = df[['no_of_adults', 'no_of_children', 'no_of_weekend_nights',
       'no_of_week_nights', 'type_of_meal_plan', 'required_car_parking_space',
       'room_type_reserved', 'lead_time', 'arrival_year', 'arrival_month',
       'arrival_date', 'market_segment_type', 'repeated_guest',
       'no_of_previous_cancellations', 'no_of_previous_bookings_not_canceled',
       'avg_price_per_room', 'no_of_special_requests']]
```

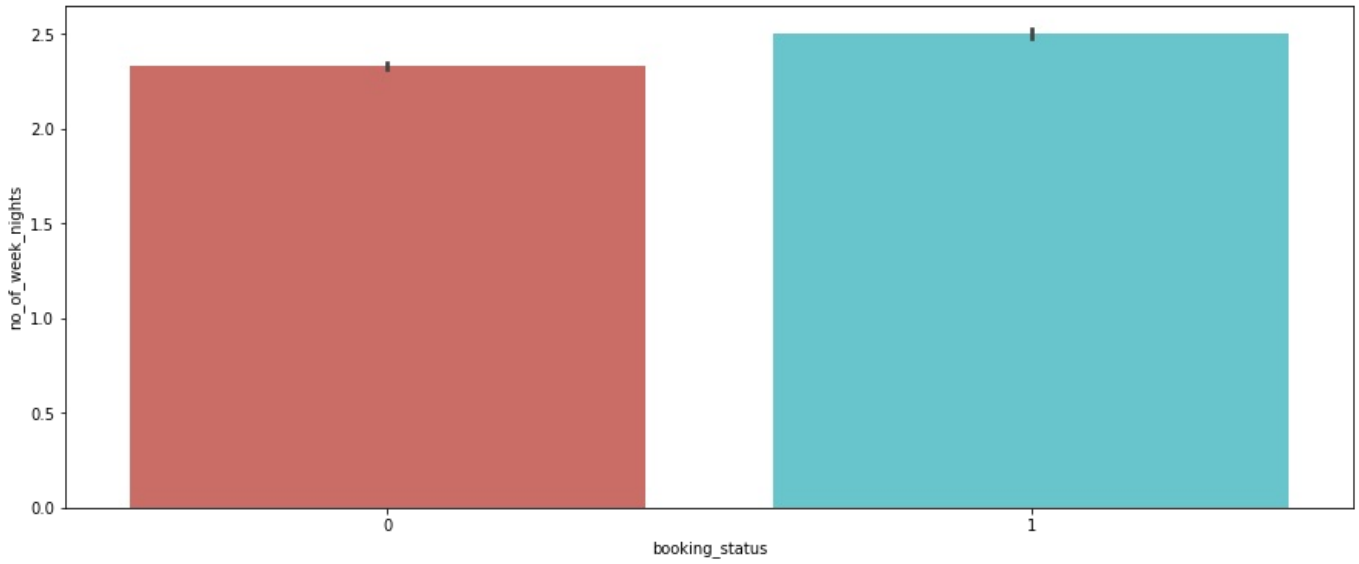
In [27]:
```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.barplot(x=df['booking_status'], y = df[i], palette = 'hls')
    plt.show()
```

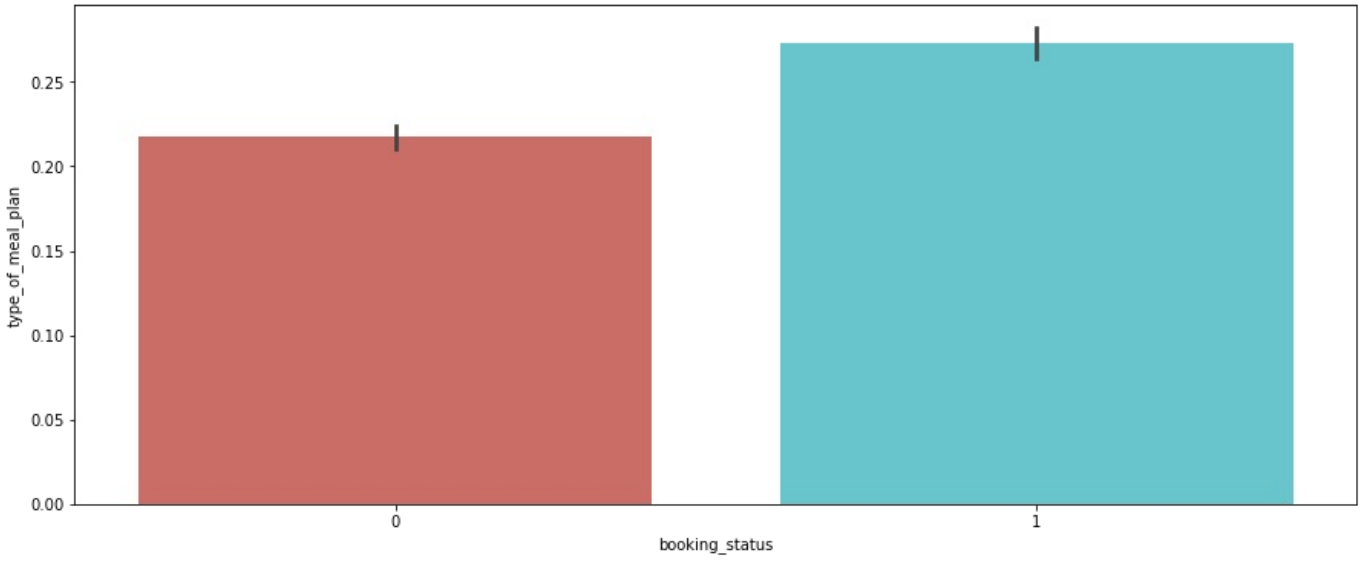


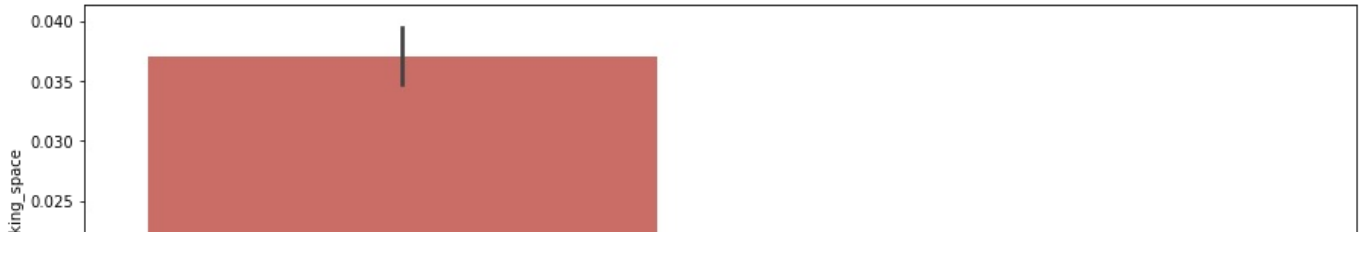



0.02
0.00
0
1
booking_status

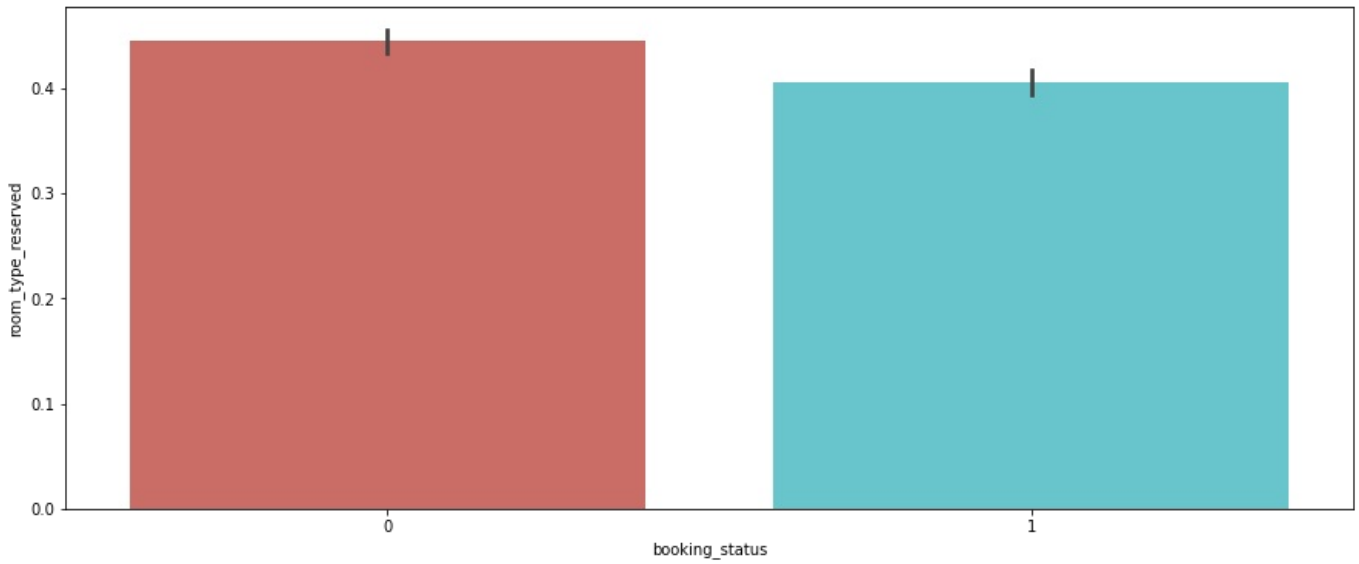

no_of_weekend_nights
0.8
0.6
0.4
0.2
0.0
0
1
booking_status

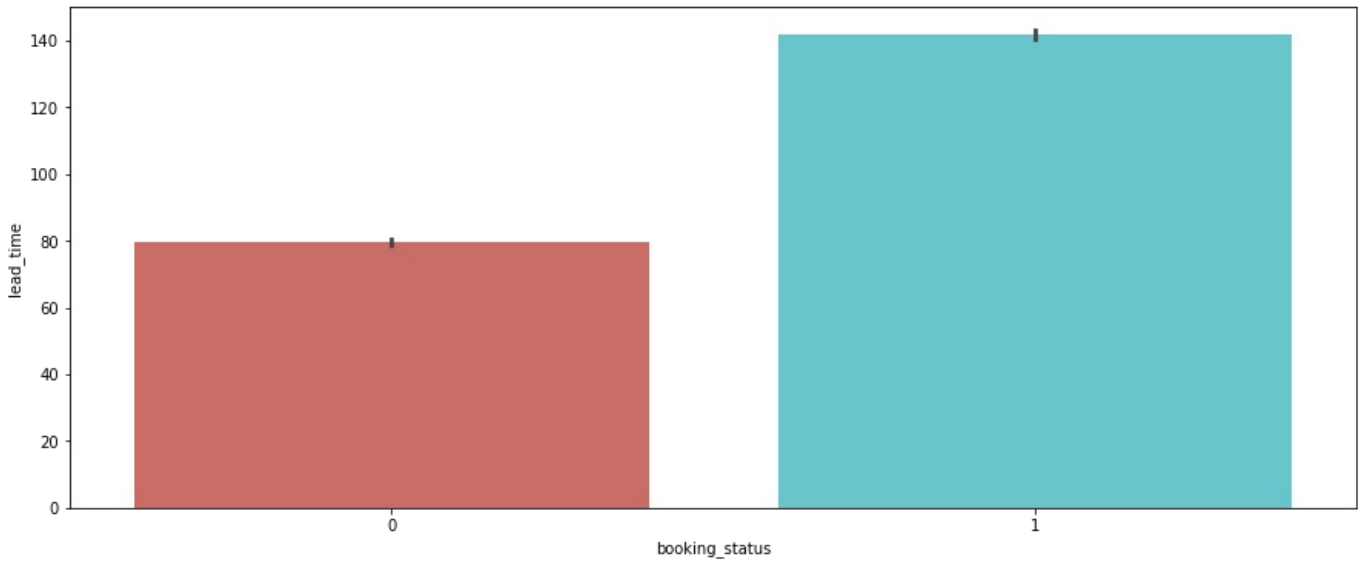

no_of_week_nights
2.5
2.0
1.5
1.0
0.5
0.0
0
1
booking_status

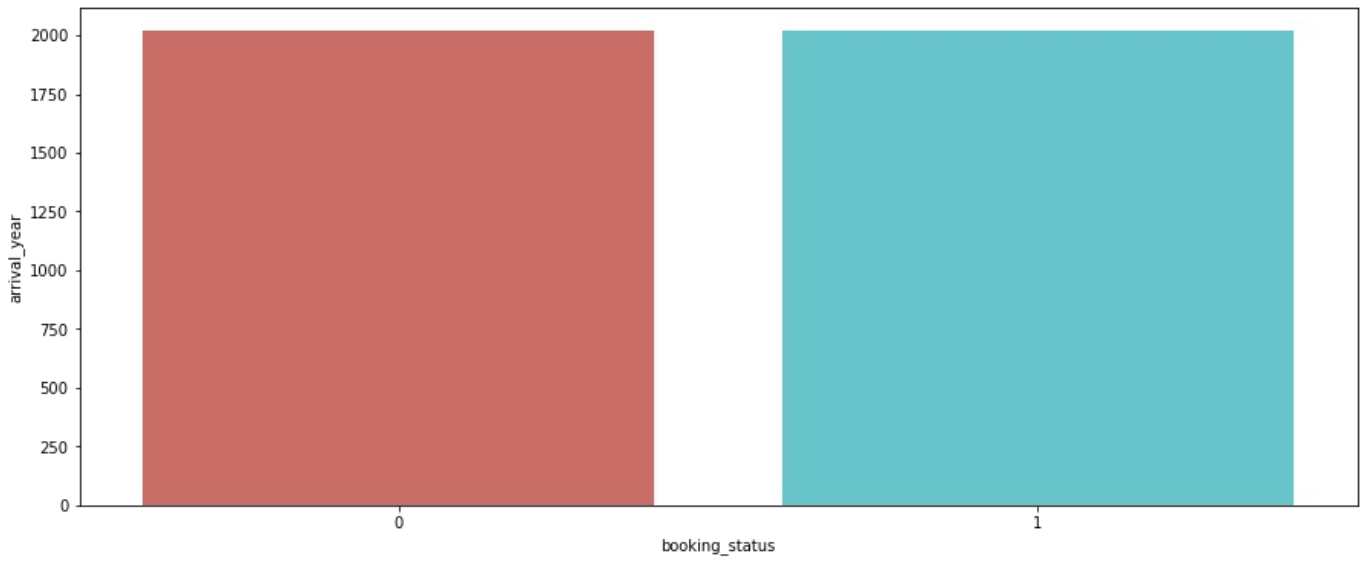

type_of_meal_plan
0.25
0.20
0.15
0.10
0.05
0.00
0
1
booking_status


0.040
0.035
0.030
0.025

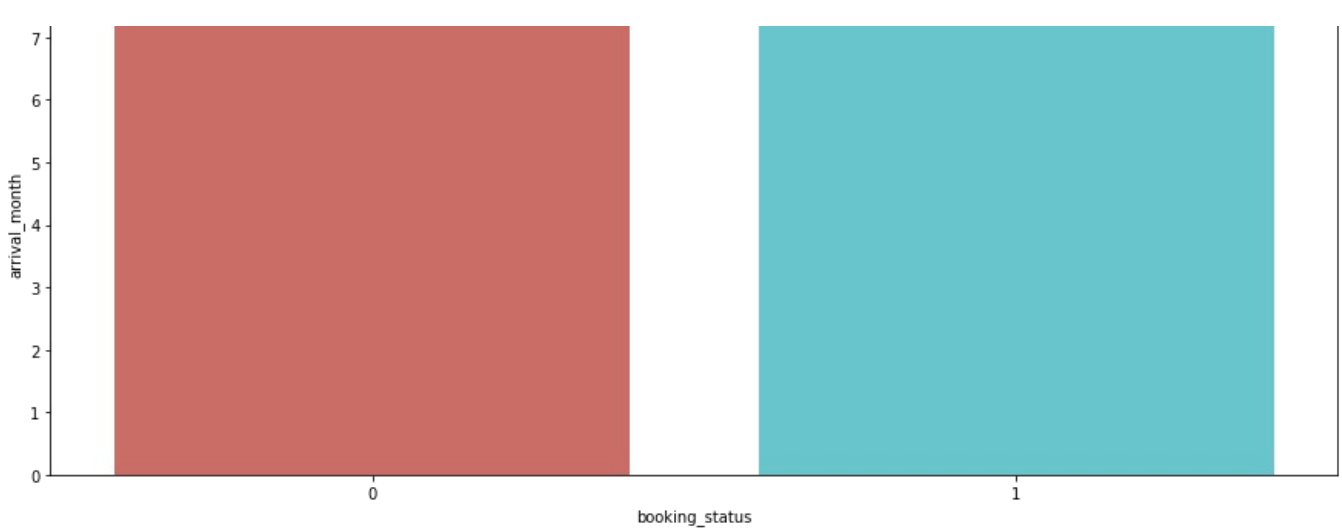

required_car_parl
0.020
0.015
0.010
0.005
0.000
0
1
booking_status

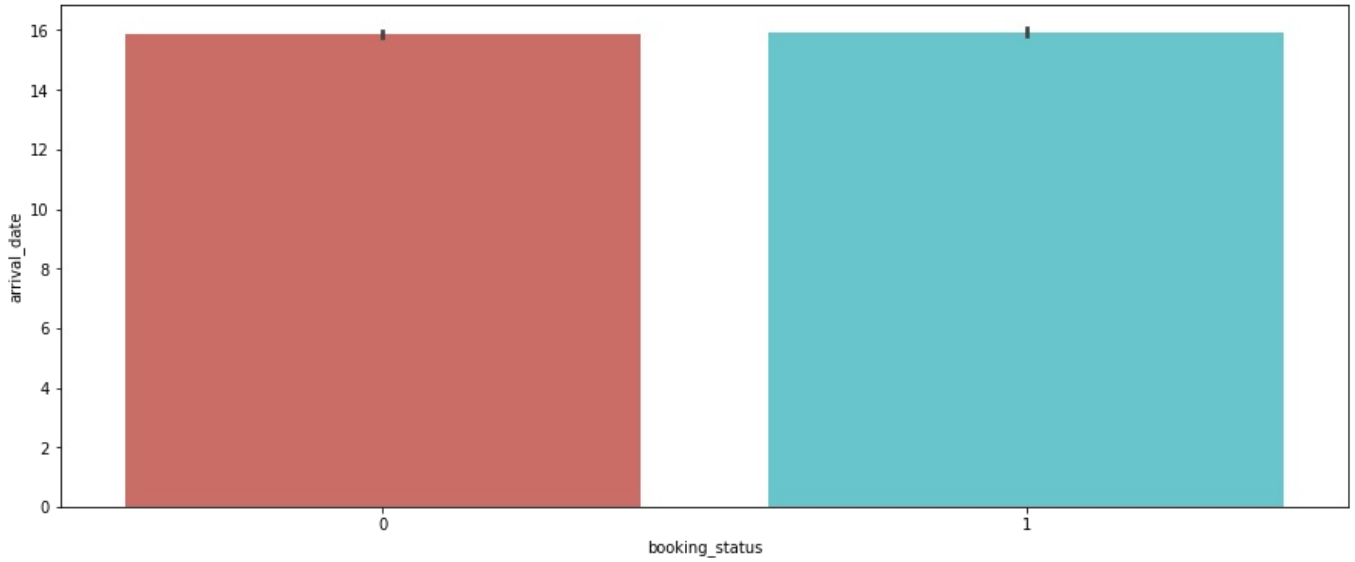

room_type_reserved
0.4
0.3
0.2
0.1
0.0
0
1
booking_status

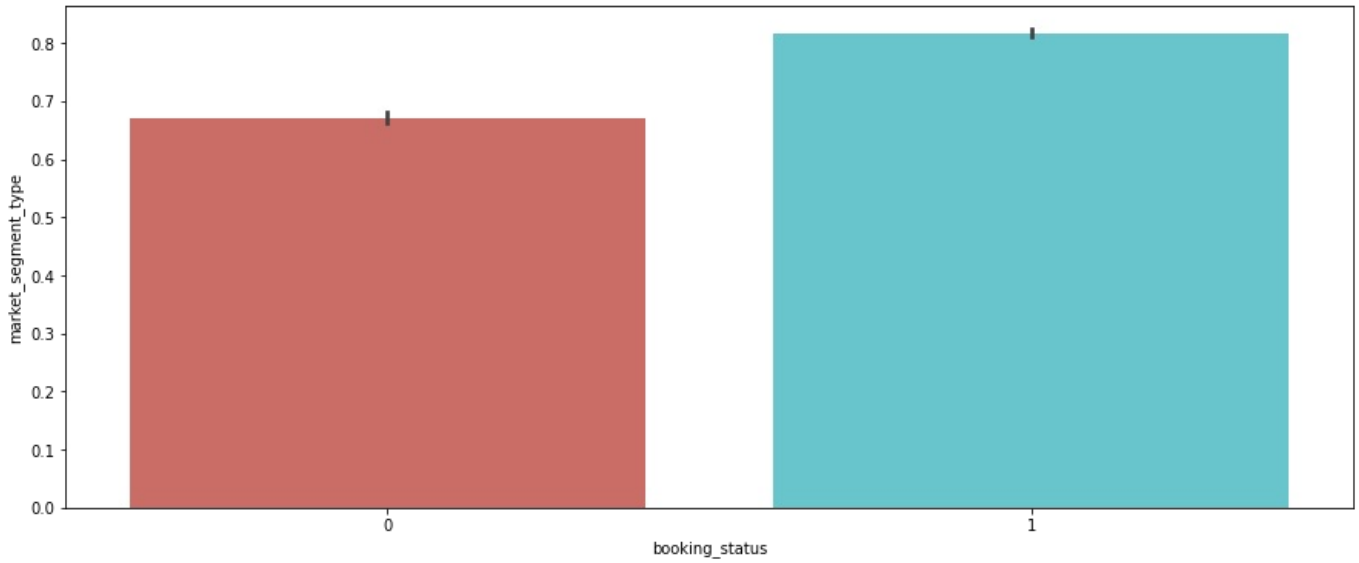

lead_time
140
120
100
80
60
40
20
0
0
1
booking_status


arrival_year
2000
1750
1500
1250
1000
750
500
250
0
0
1
booking_status


8

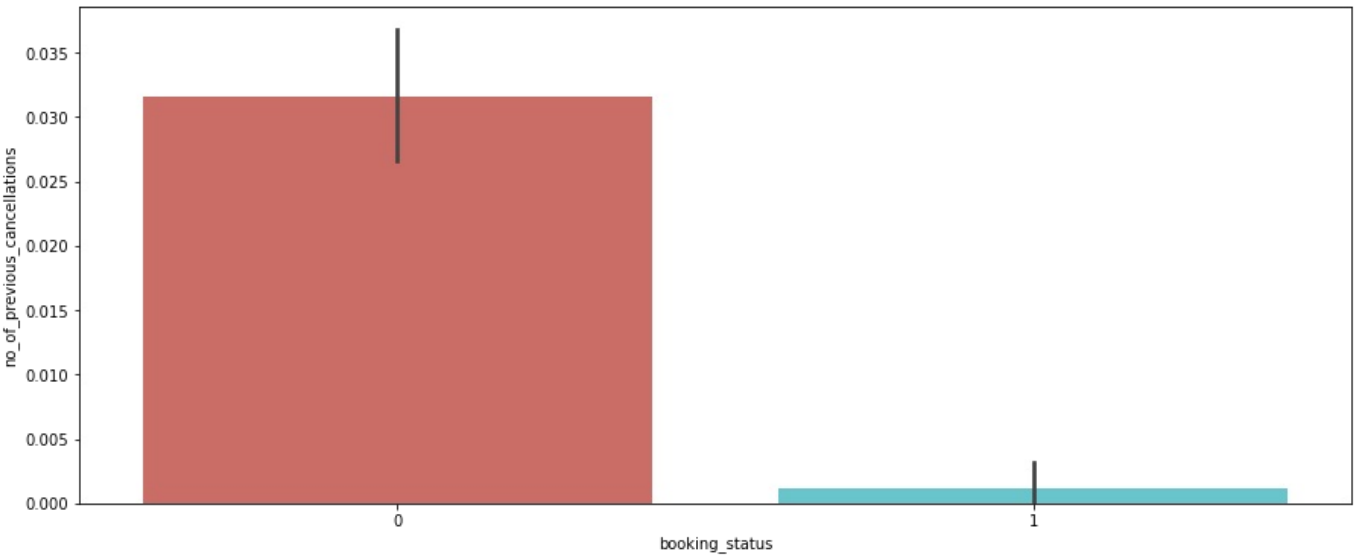

arrival_month
booking_status
0
1

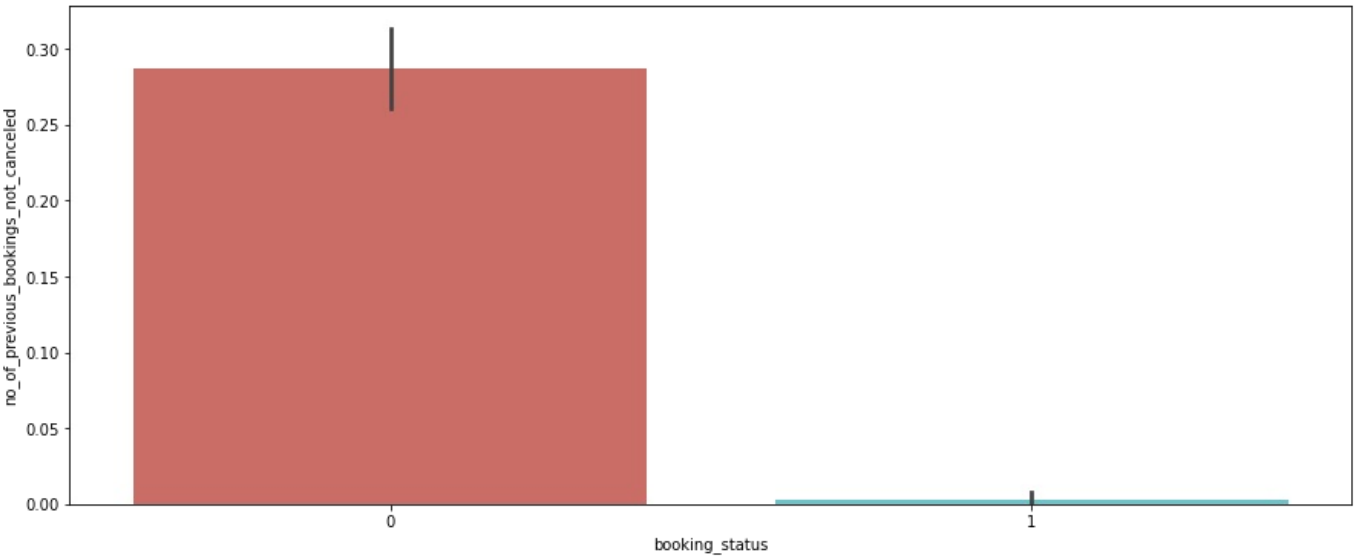

arrival_date
booking_status
0
1

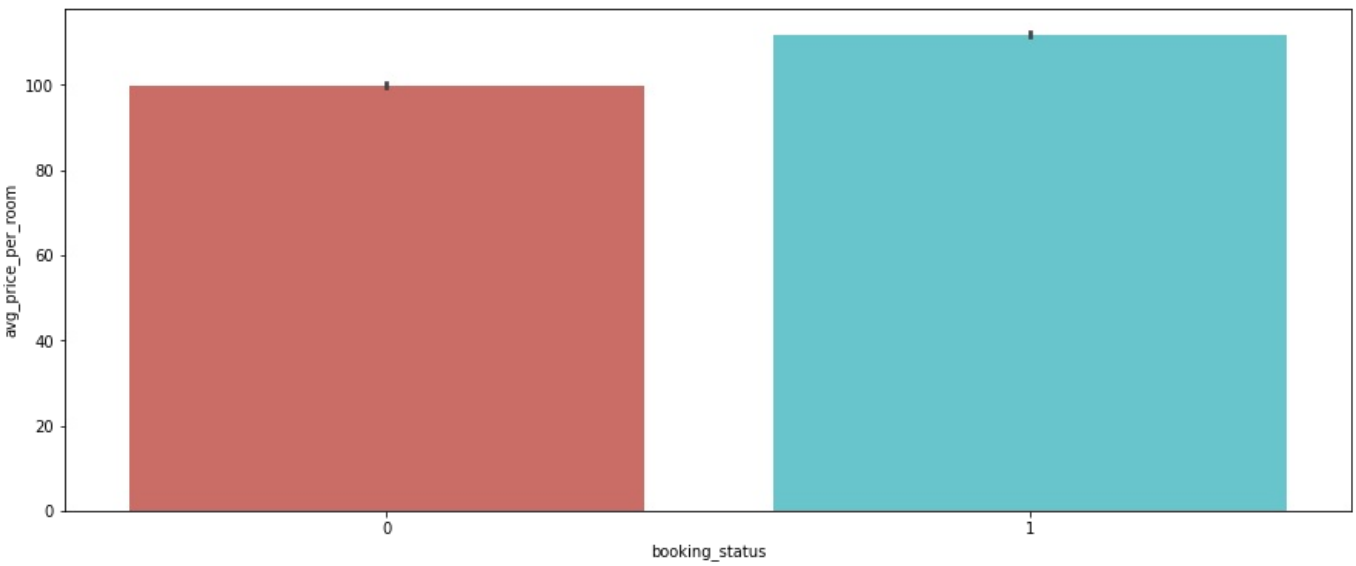

market_segment_type
booking_status
0
1


repeated_guest

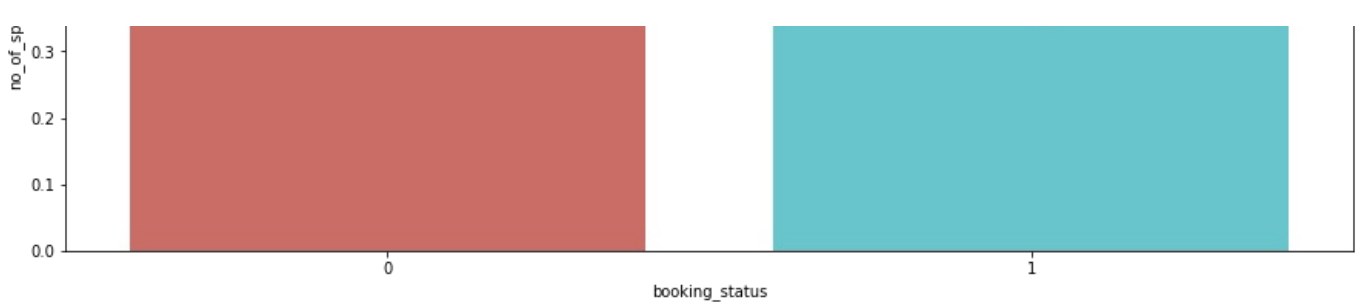

0.00
0
1
booking_status

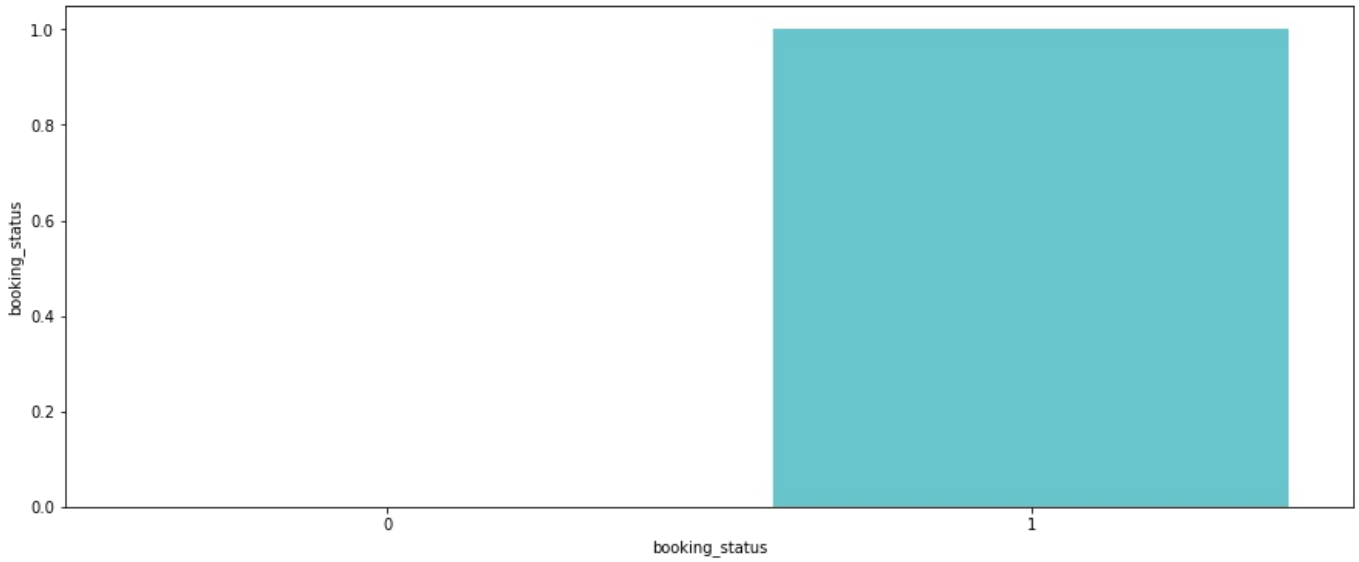

no_of_previous_cancellations
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
1
booking_status


no_of_previous_bookings_not_canceled
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
booking_status

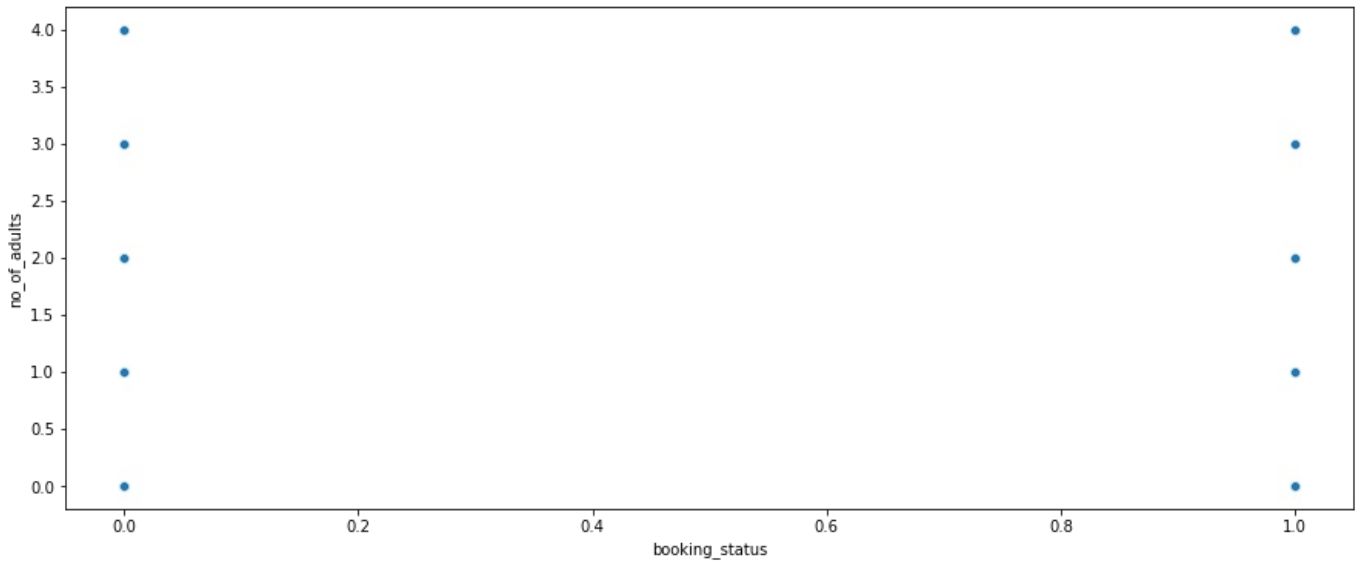

avg_price_per_room
100
80
60
40
20
0
0
1
booking_status

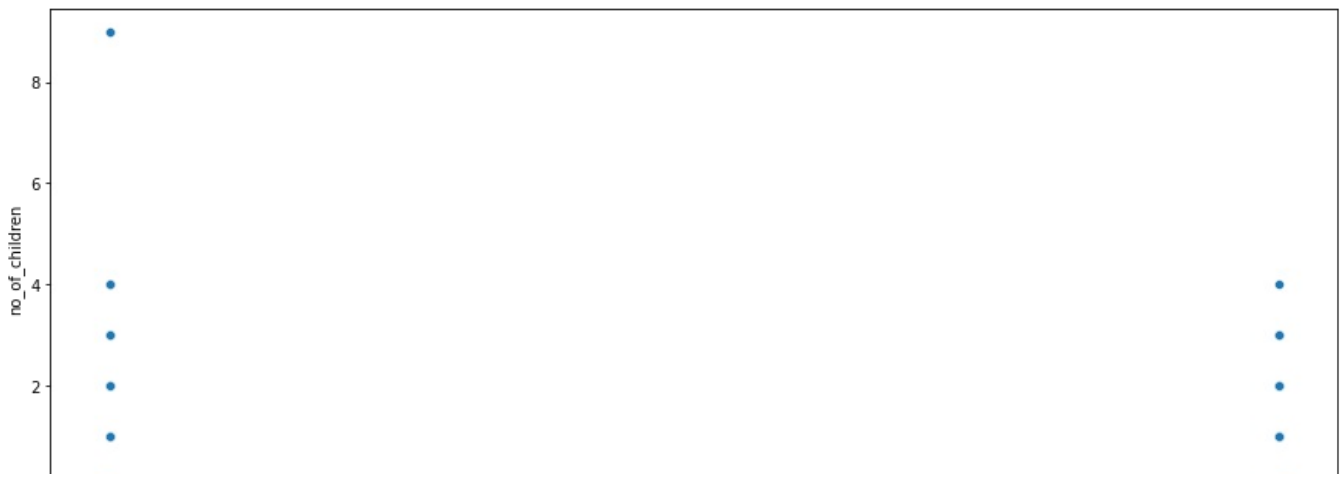

0.7
0.6
0.5
0.4



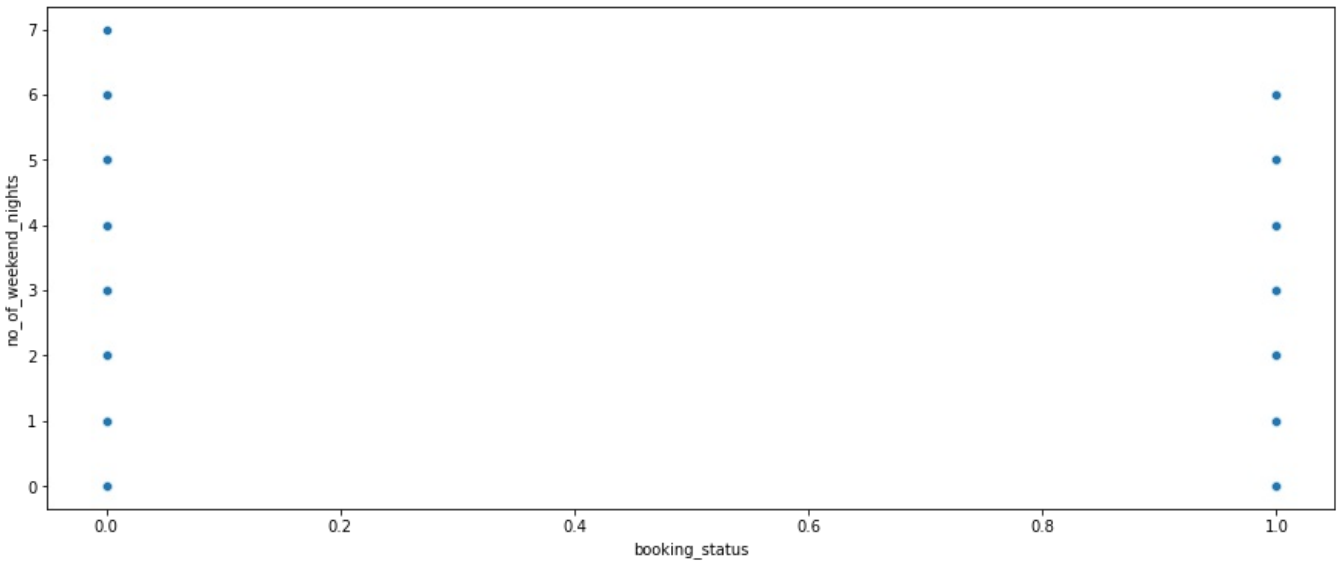


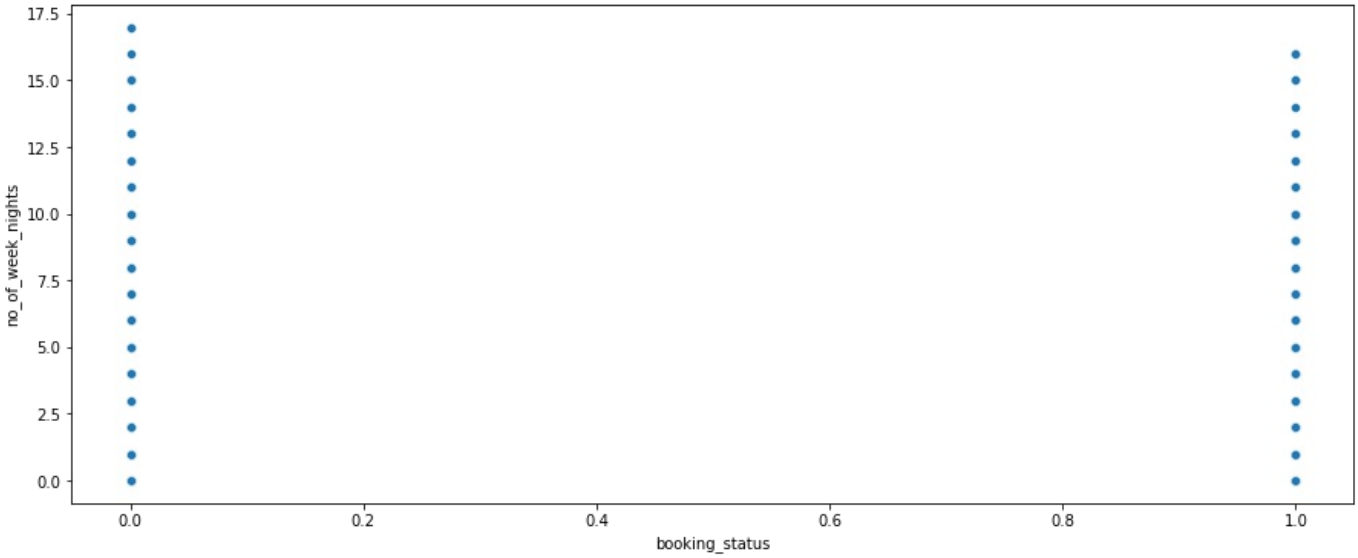
In [28]:
```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.scatterplot(x = df['booking_status'], y = df2[i], palette = 'hls')
    plt.show()
```

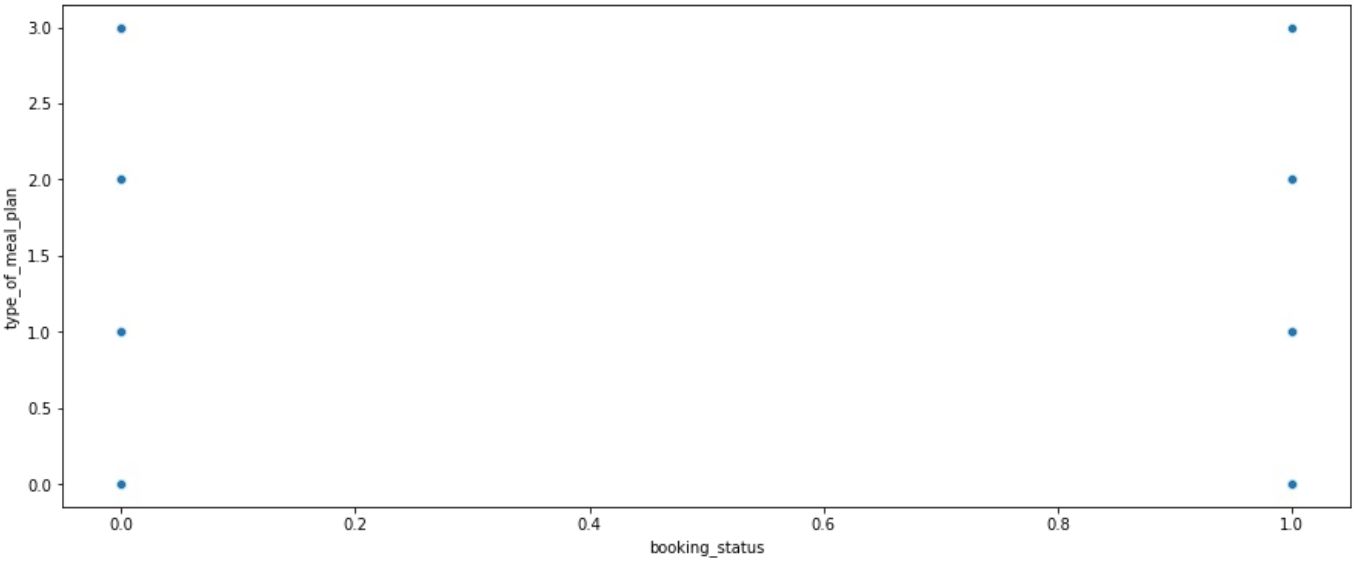




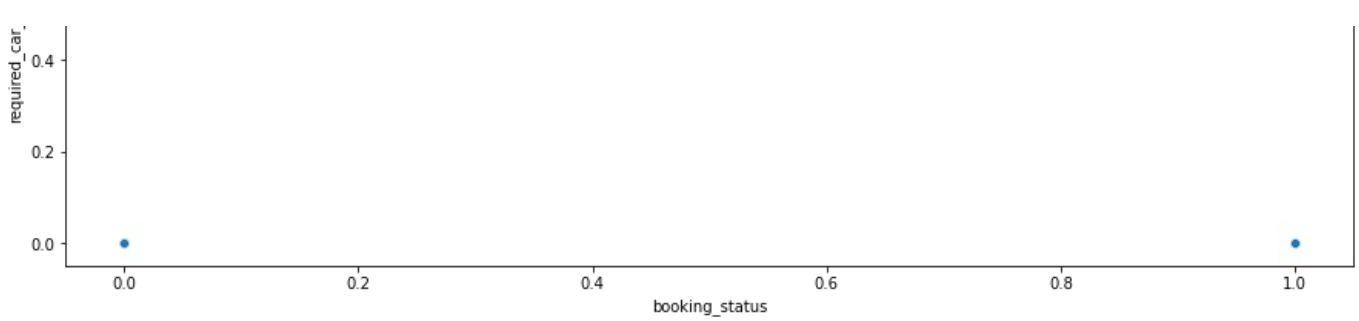

0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

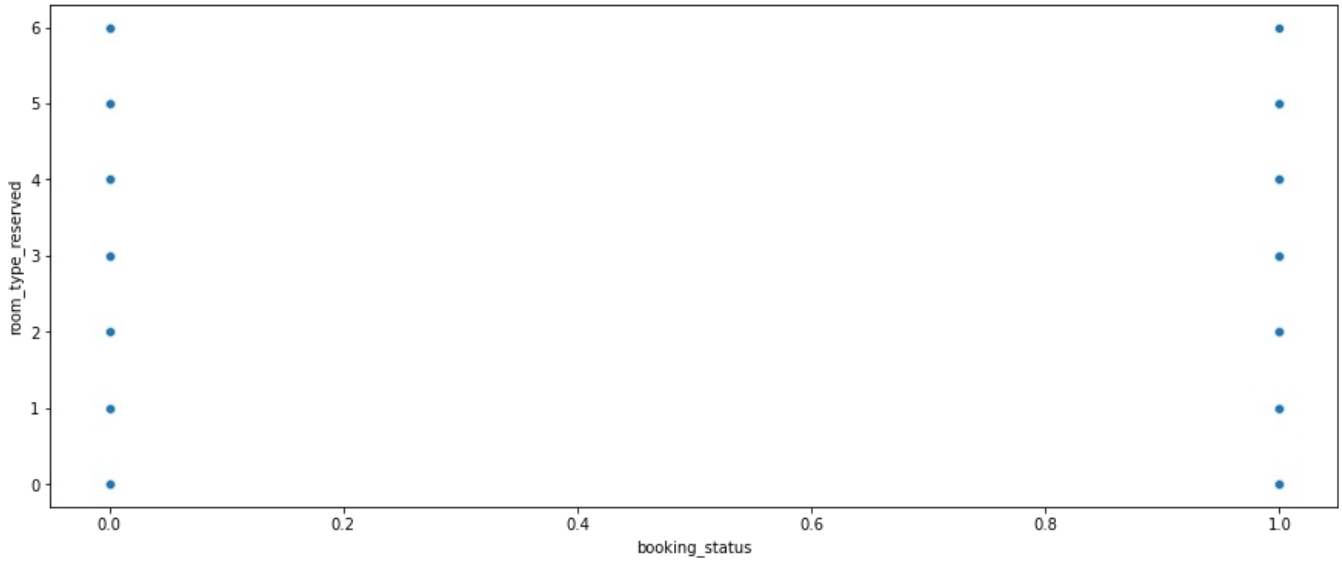

no_of_weekend_nights
7
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

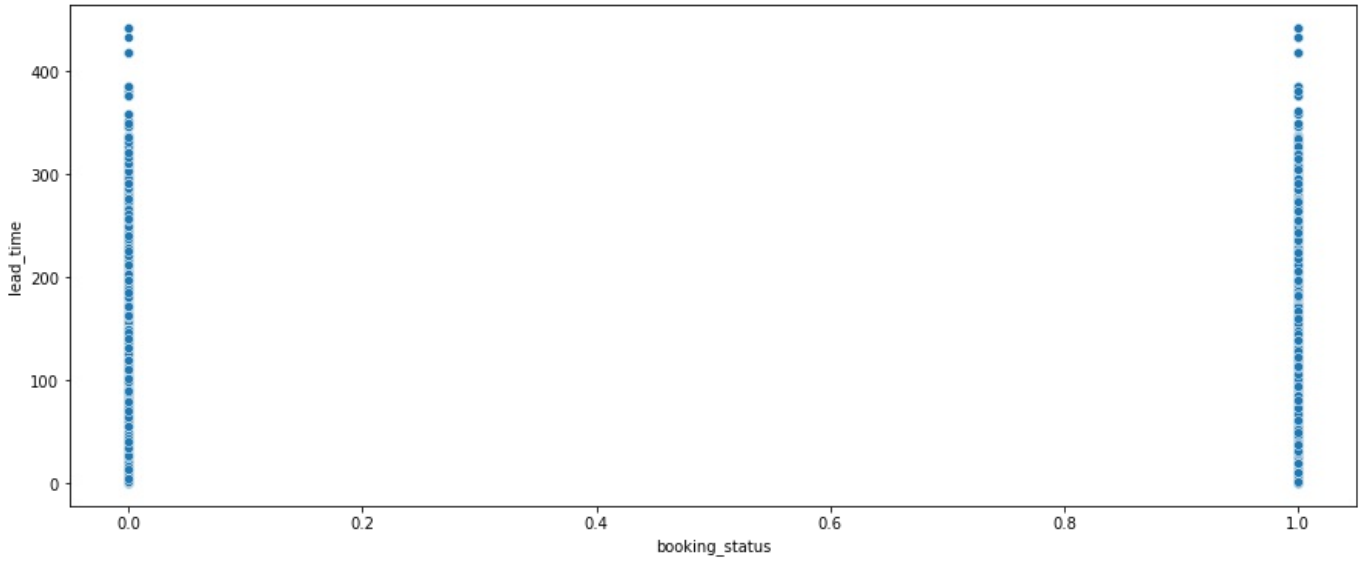

no_of_week_nights
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

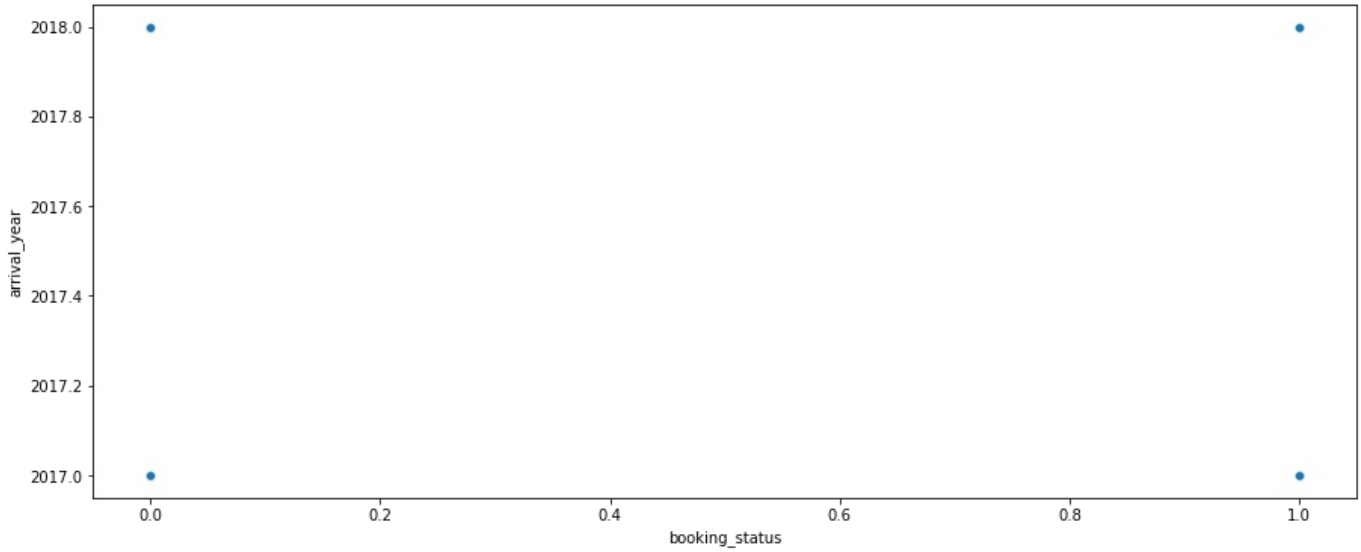

type_of_meal_plan
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


_parking_space
1.0
0.8
0.6

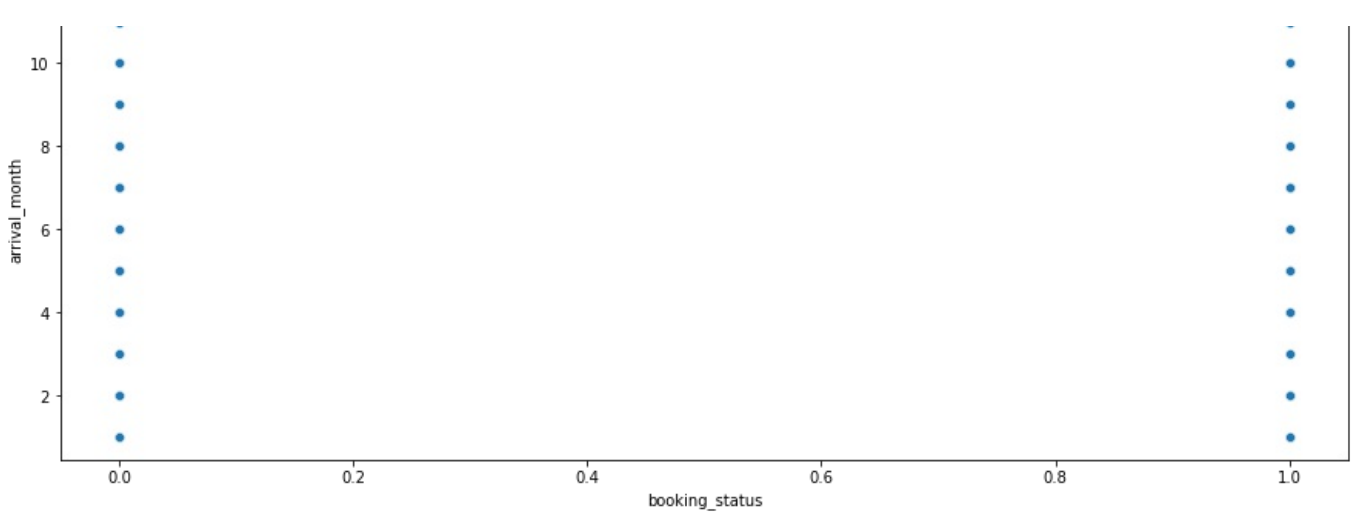

required_car
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

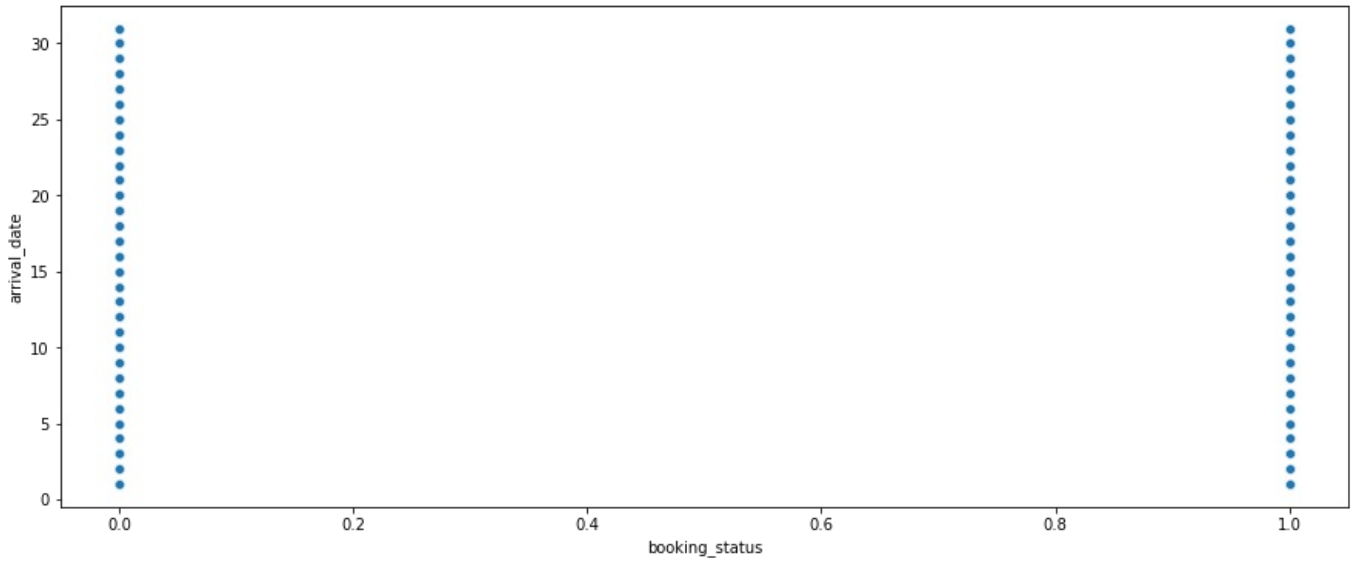

room_type_reserved
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

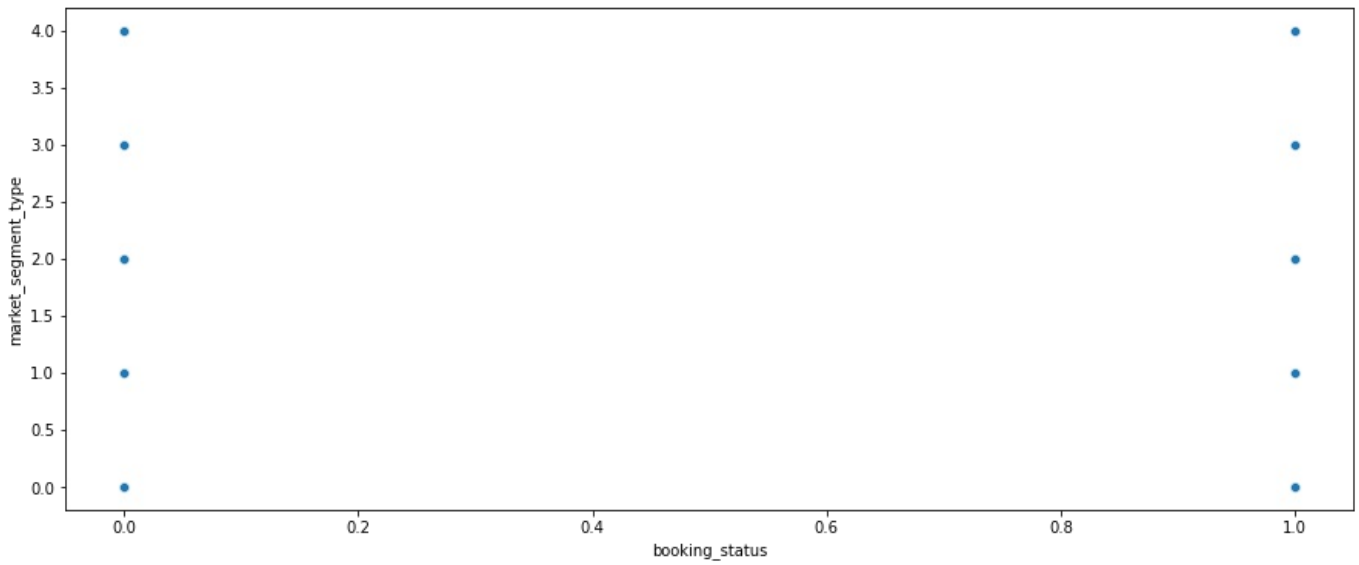

lead_time
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

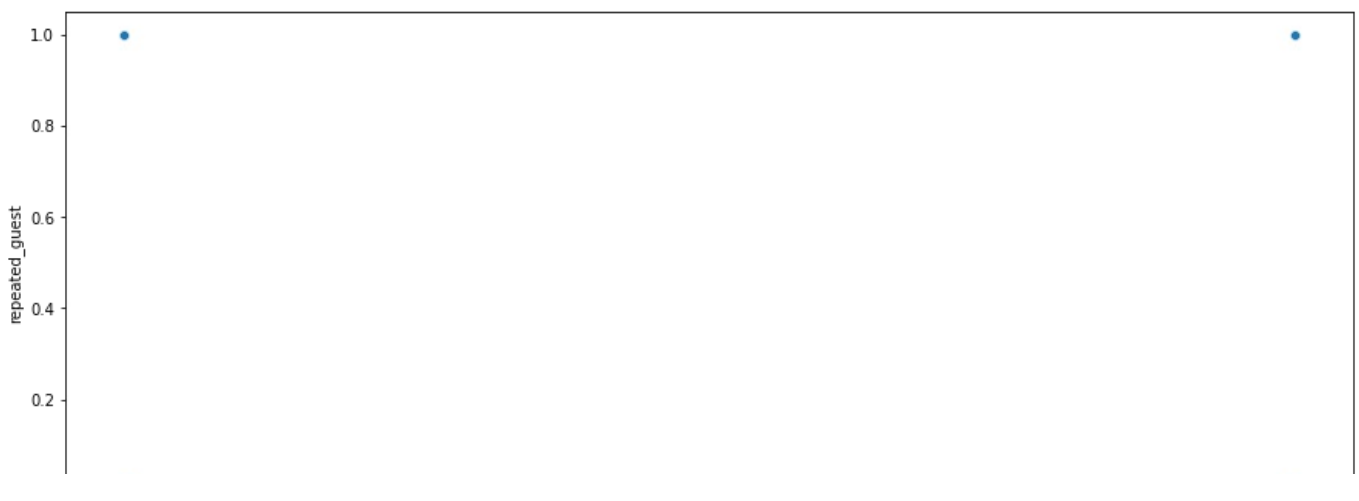

arrival_year
2018.0
2017.8
2017.6
2017.4
2017.2
2017.0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

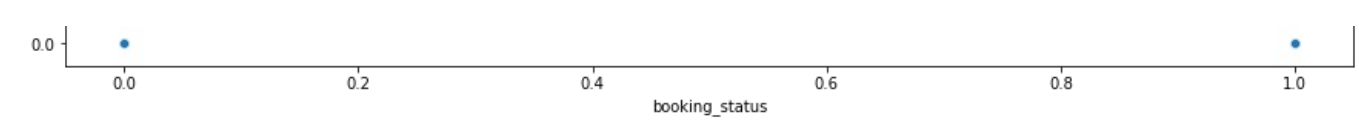

12

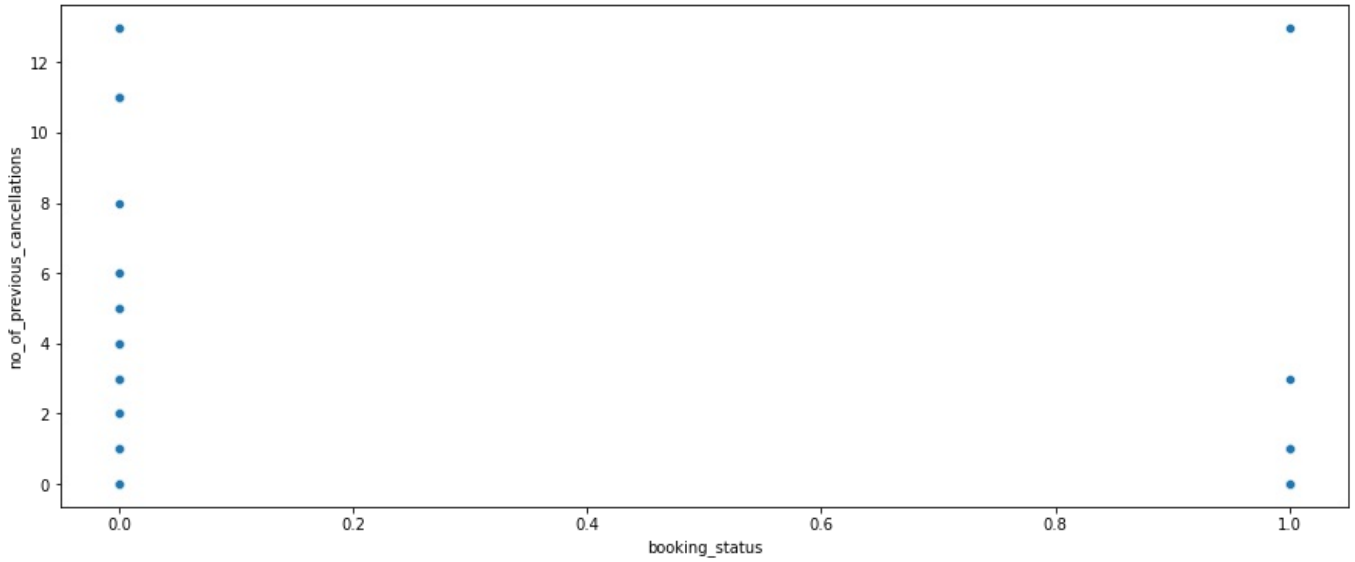

arrival_month
booking_status

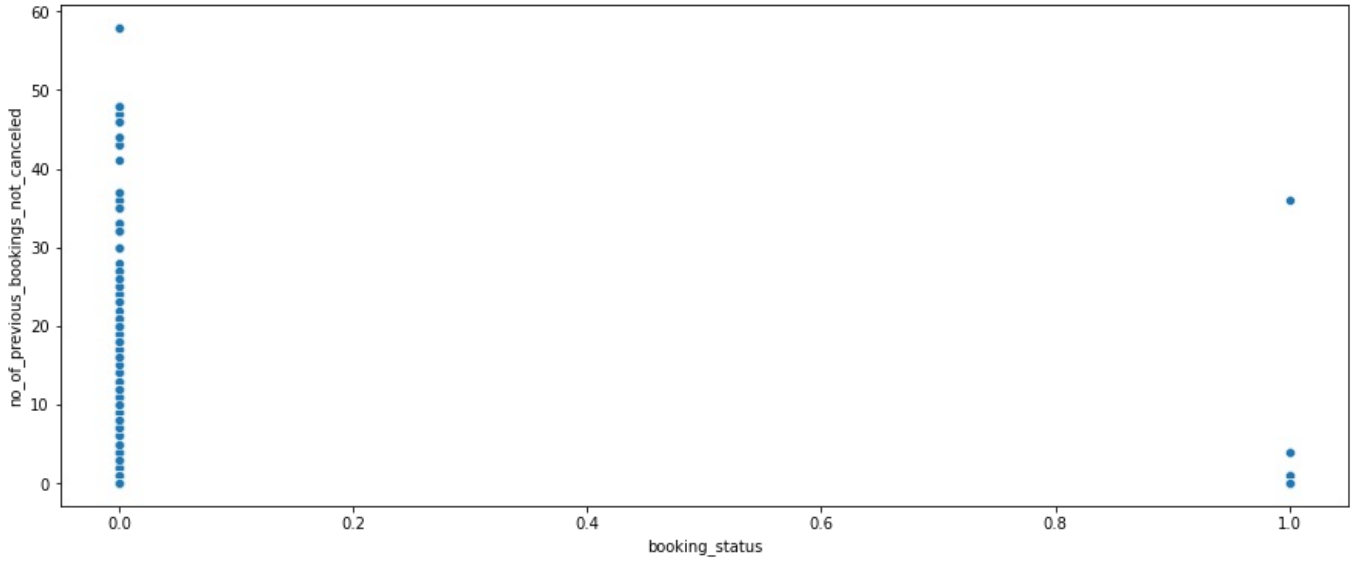

arrival_date
booking_status

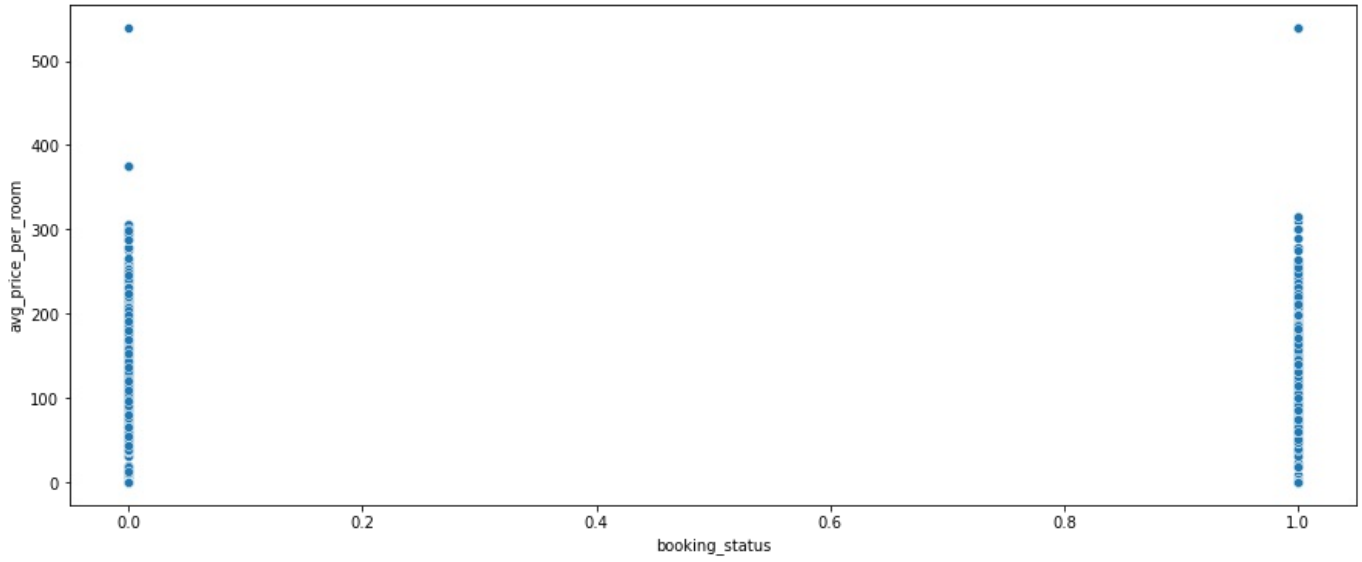

market_segment_type
booking_status


repeated_guest

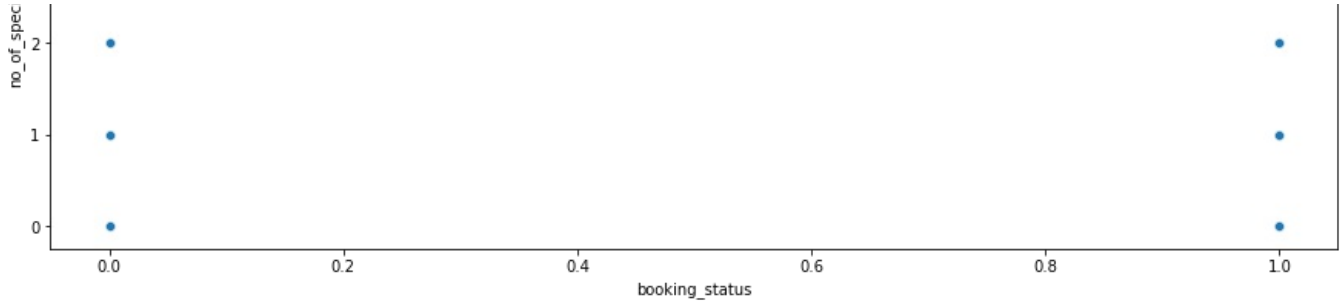

0.0
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


no_of_previous_cancellations
0
2
4
6
8
10
12
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


no_of_previous_bookings_not_canceled
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


avg_price_per_room
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

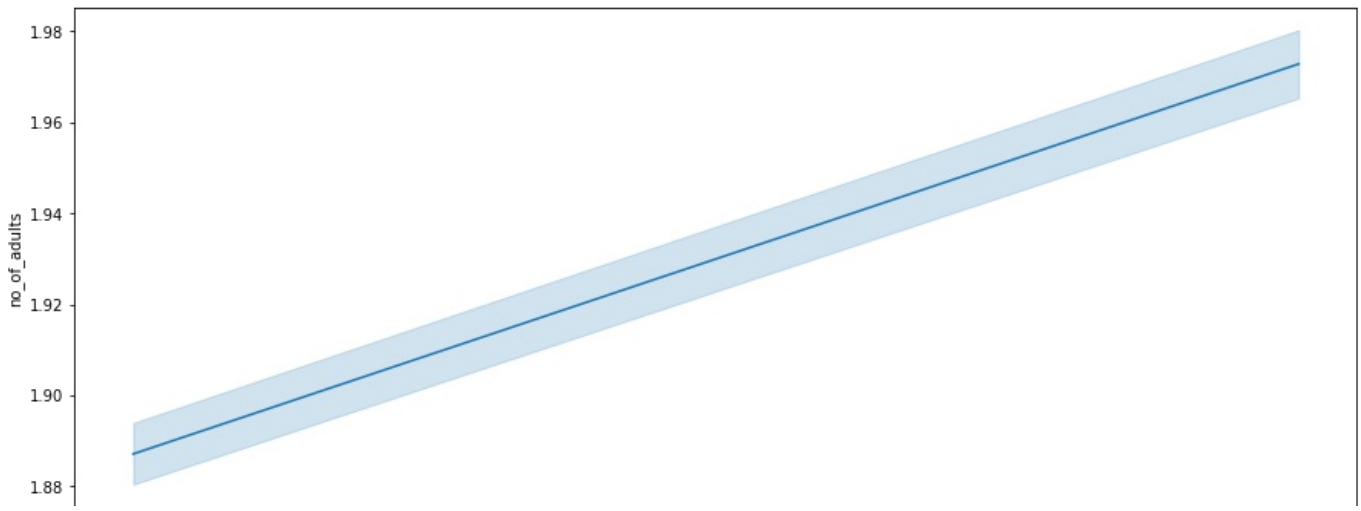

ial_requests
3
4
5

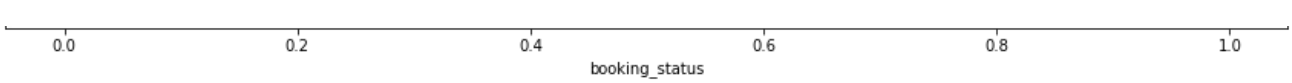


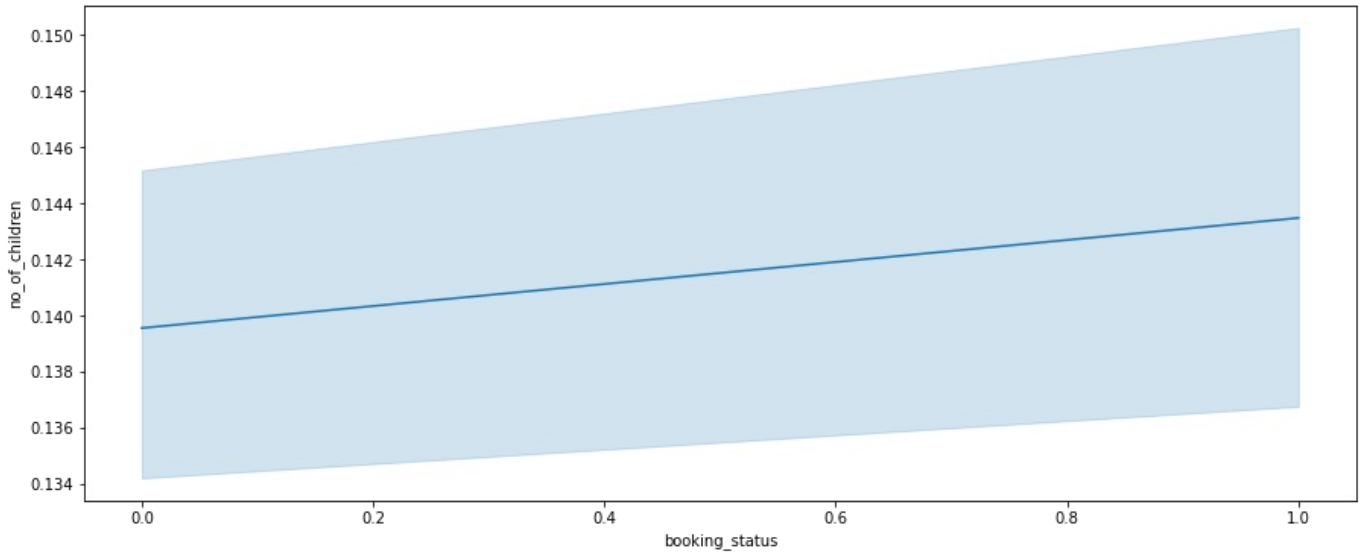

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2894             try:
-> 2895                 return self._engine.get_loc(casted_key)
   2896             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'booking_status'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-28-5f01b7b0ddd8> in <module>
      1 for i in df.columns:
      2     plt.figure(figsize=(15,6))
----> 3     sns.scatterplot(x = df['booking_status'], y = df2[i], palette = 'hls')
      4     plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2900             if self.columns.nlevels > 1:
   2901                 return self._getitem_multilevel(key)
-> 2902             indexer = self.columns.get_loc(key)
   2903             if is_integer(indexer):
   2904                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2895                 return self._engine.get_loc(casted_key)
   2896             except KeyError as err:
-> 2897                 raise KeyError(key) from err
   2898
   2899         if tolerance is not None:

KeyError: 'booking_status'
```

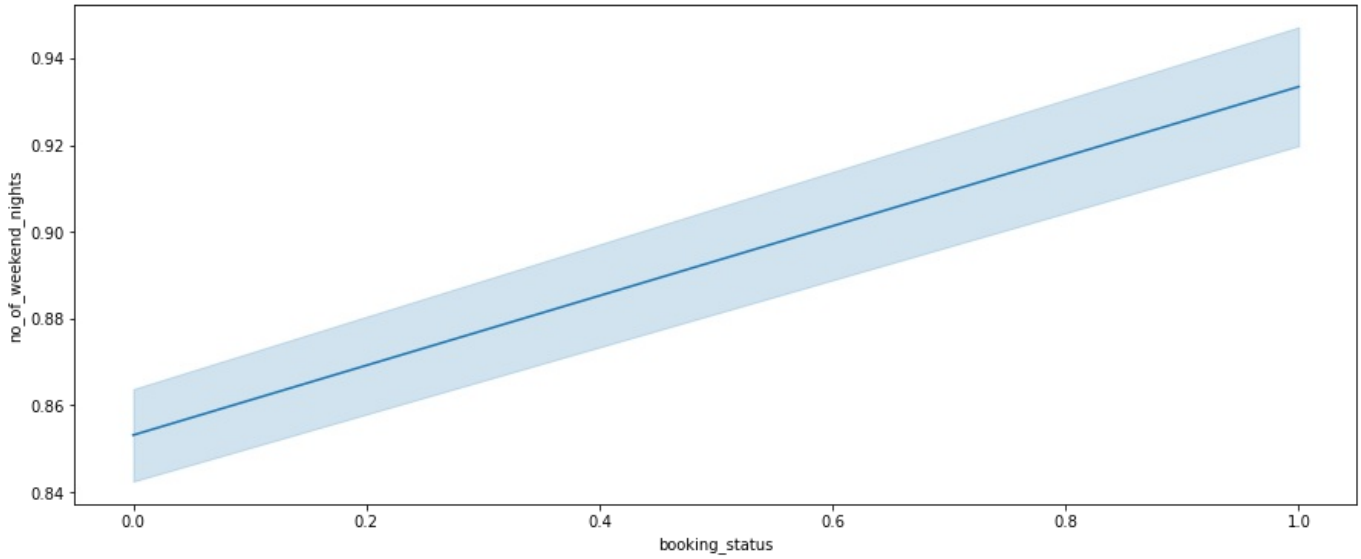
```
<Figure size 1080x432 with 0 Axes>
```

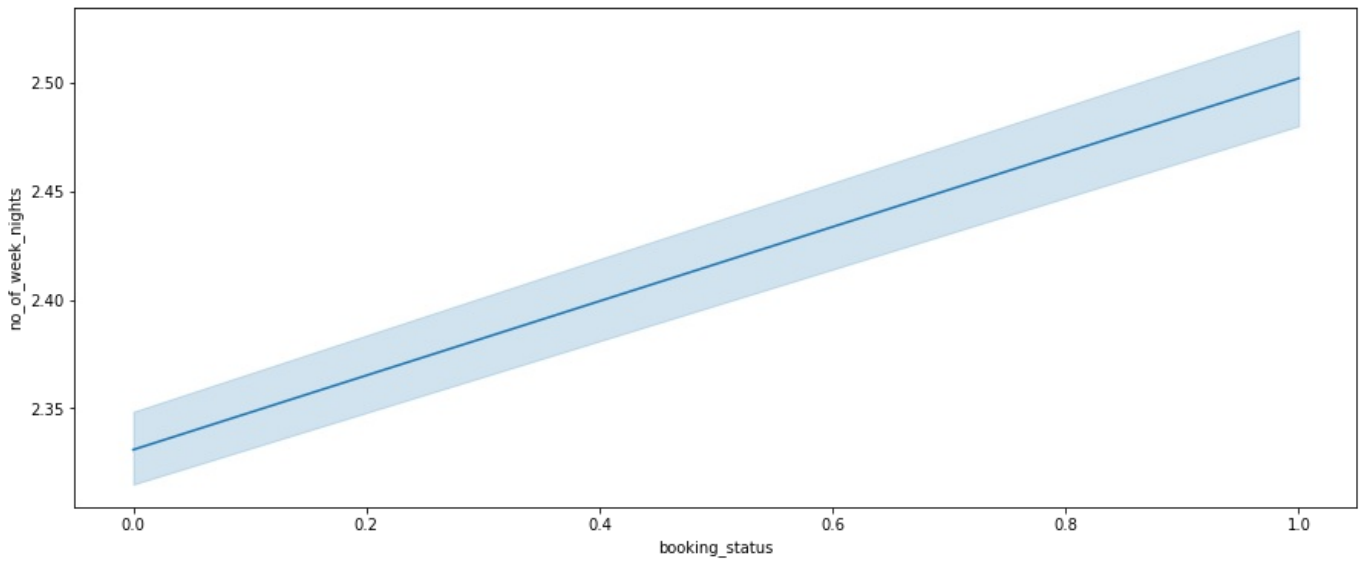
In [29]:
```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.lineplot(x = df['booking_status'], y = df[i], palette = 'hls')
    plt.show()
```

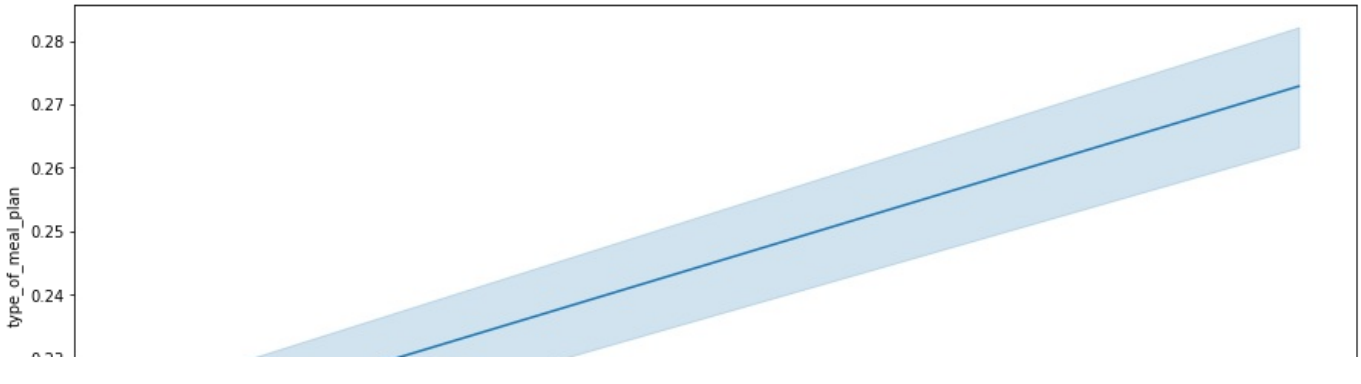



0.0
0.2
0.4
0.6
0.8
1.0
booking_status

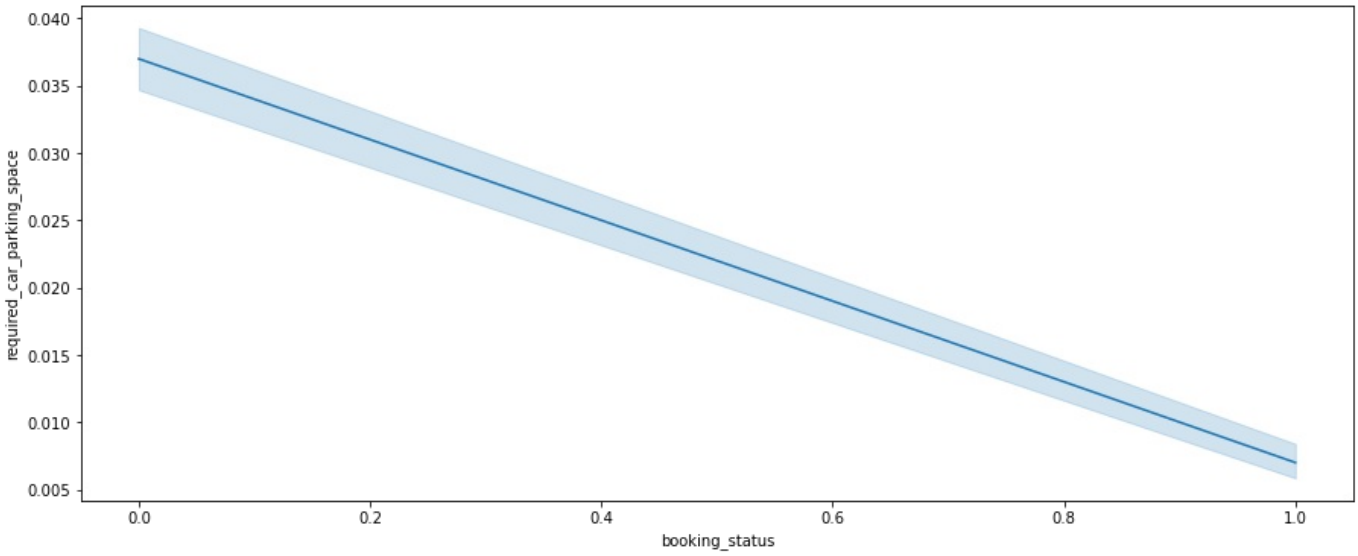

0.150
0.148
0.146
0.144
0.142
0.140
0.138
0.136
0.134
no_of_children
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

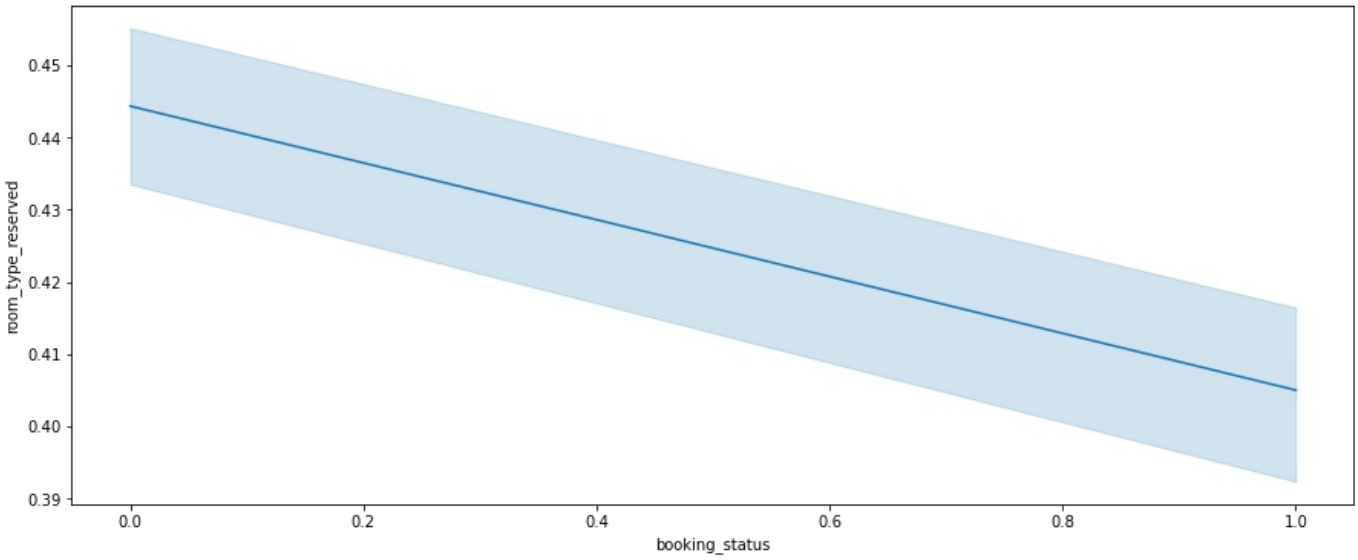

0.94
0.92
0.90
0.88
0.86
0.84
no_of_weekend_nights
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

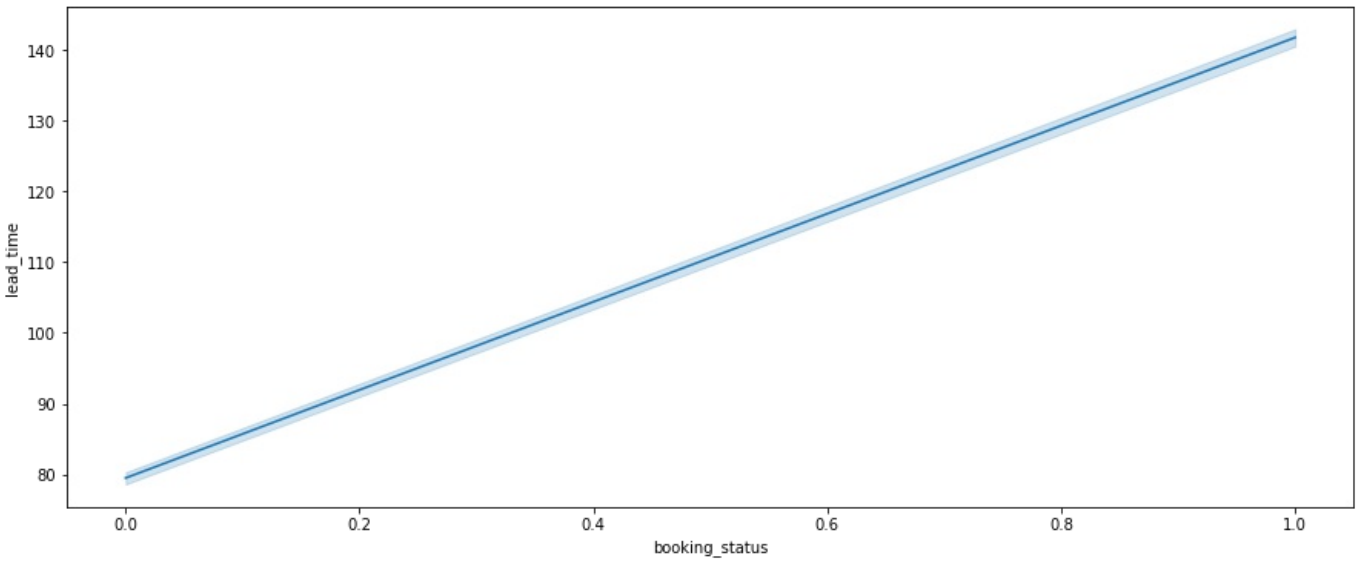

2.50
2.45
2.40
2.35
no_of_week_nights
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


0.28
0.27
0.26
0.25
0.24
type_of_meal_plan

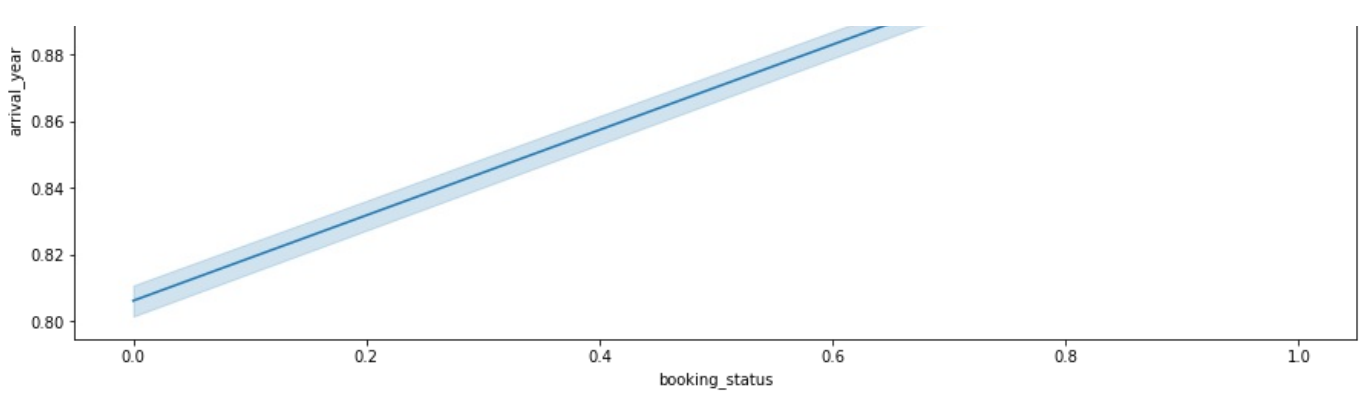

0.23
0.22
0.21
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

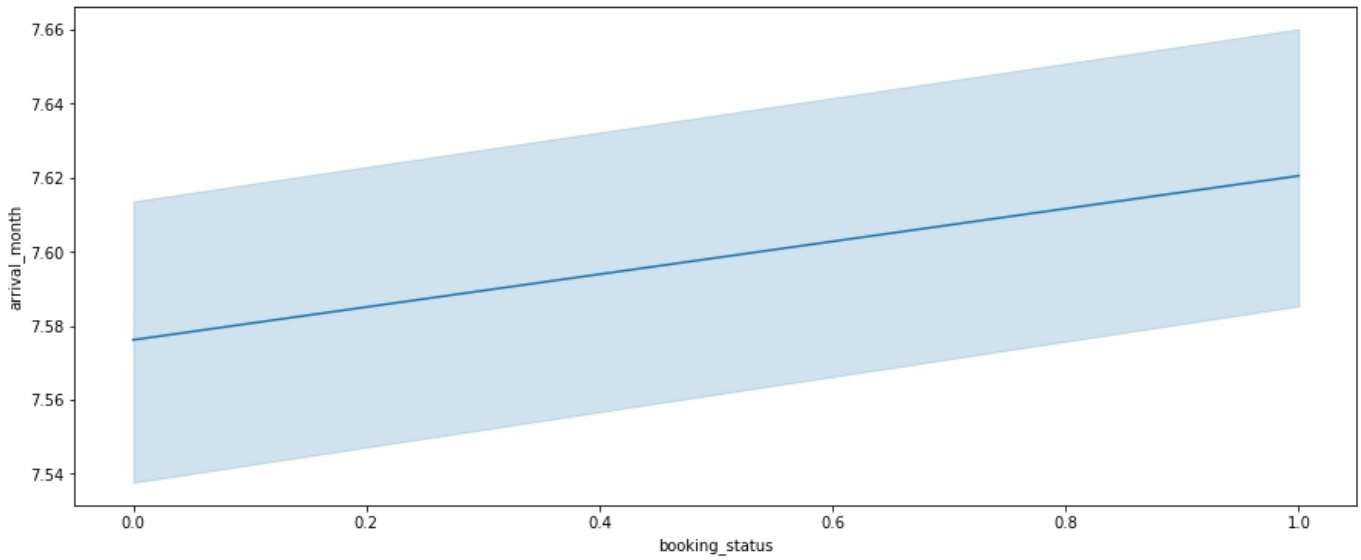

0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
required_car_parking_space
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

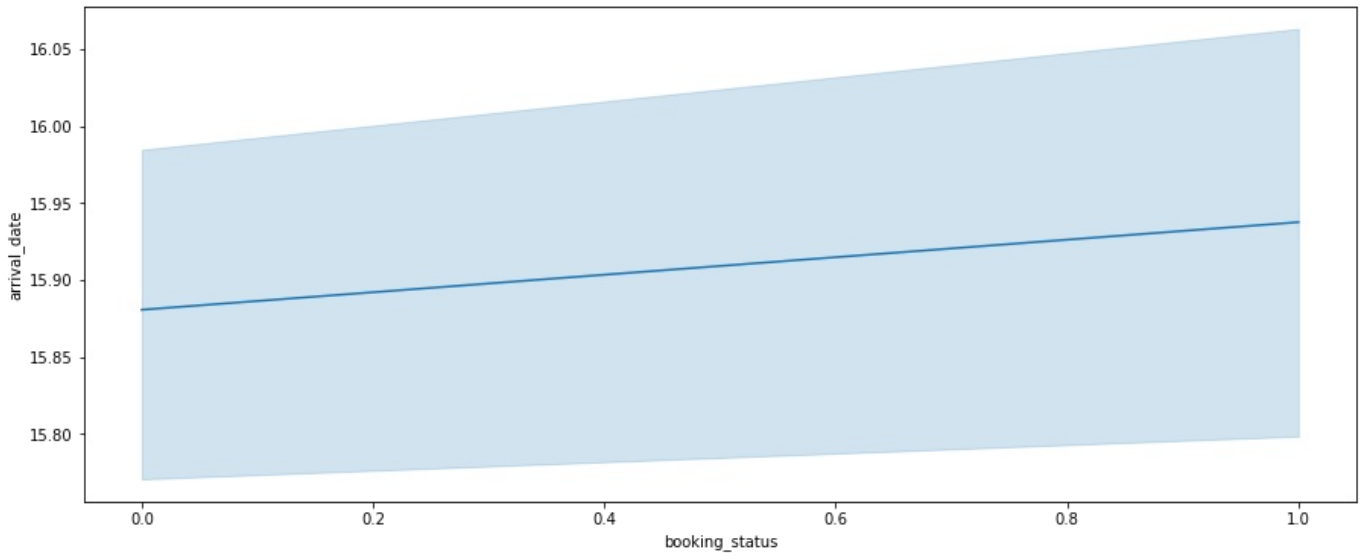

0.45
0.44
0.43
0.42
0.41
0.40
0.39
room_type_reserved
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

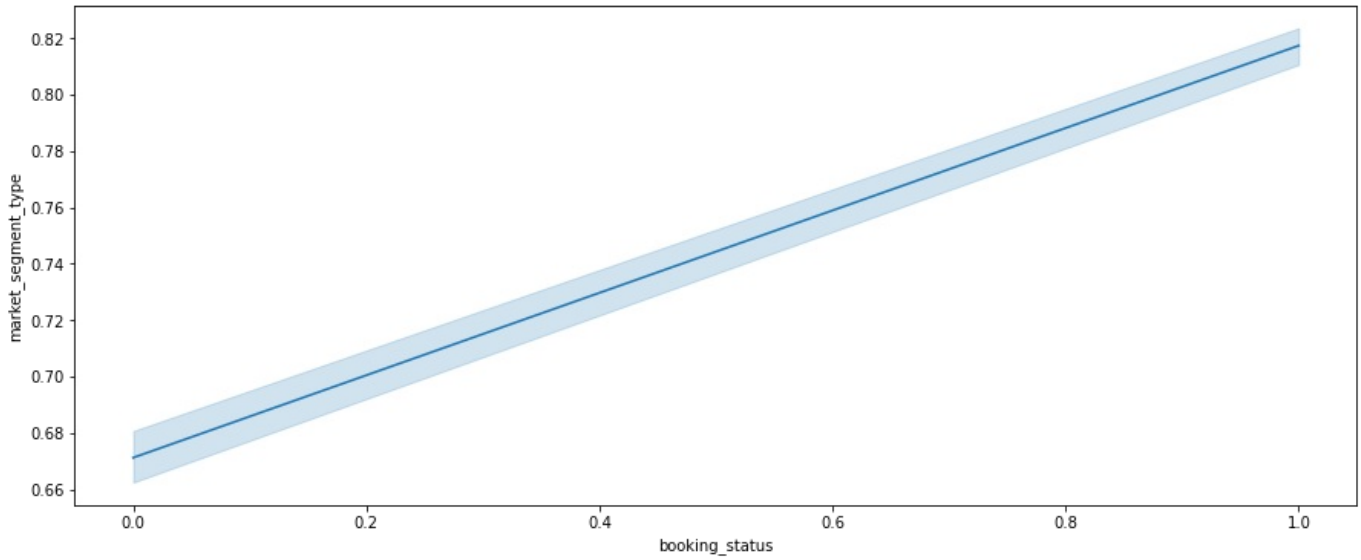

140
130
120
110
100
90
80
lead_time
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


+2.017e3
0.94
0.92
0.90

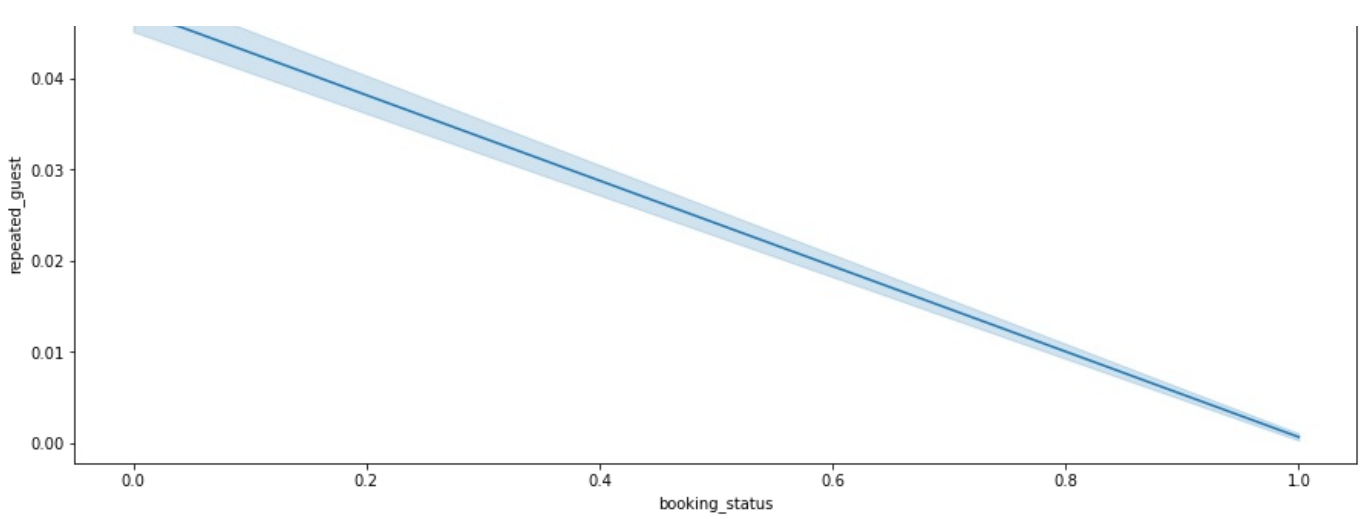

arrival_year
0.88
0.86
0.84
0.82
0.80
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

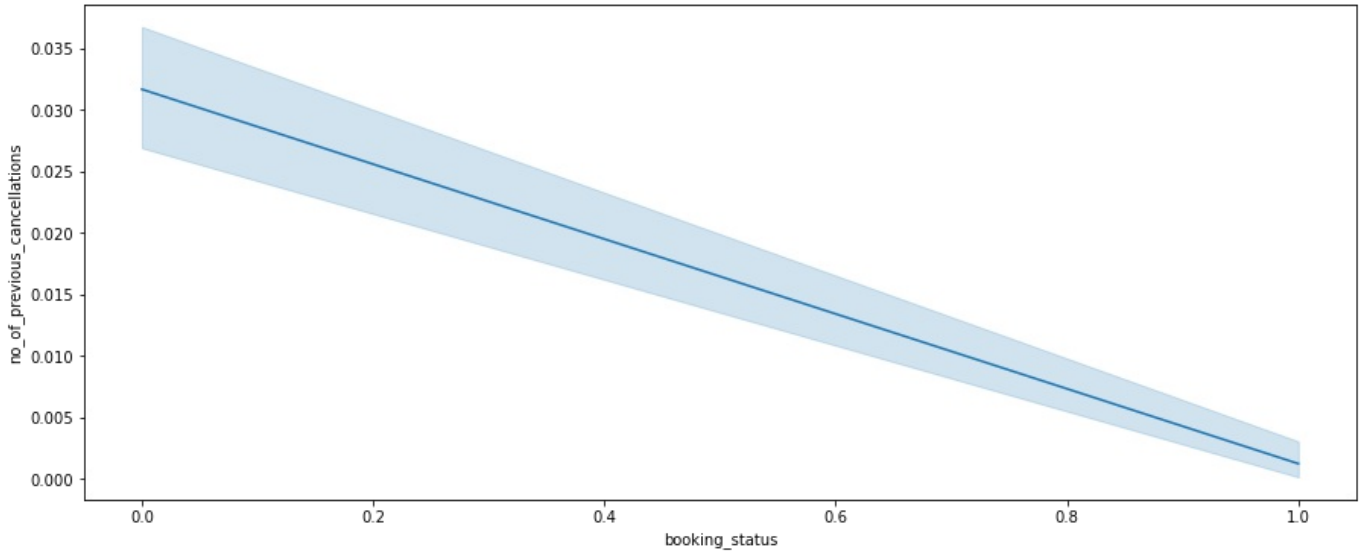

arrival_month
7.66
7.64
7.62
7.60
7.58
7.56
7.54
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

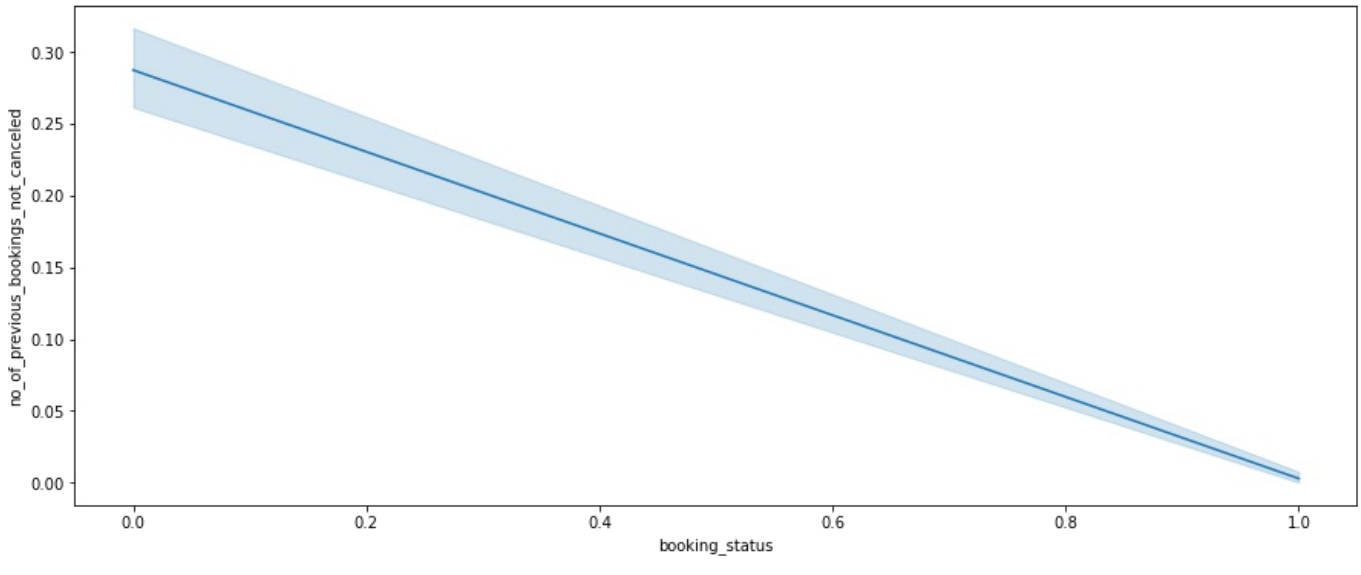

arrival_date
16.05
16.00
15.95
15.90
15.85
15.80
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

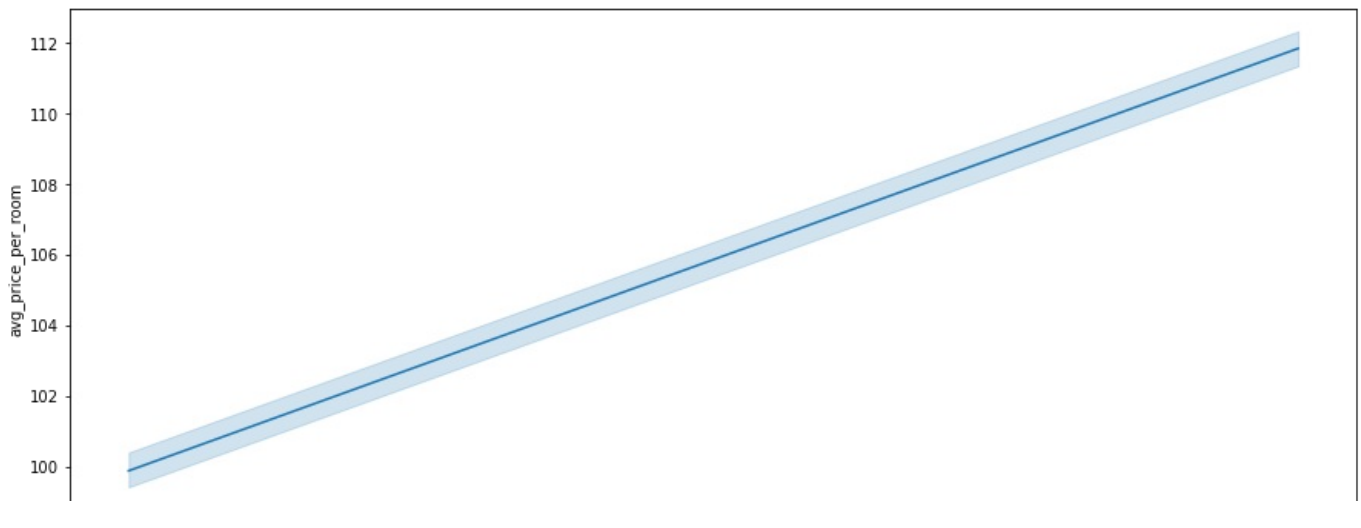

market_segment_type
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

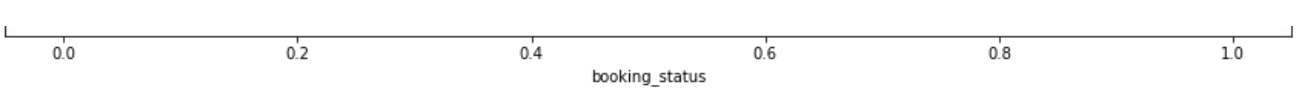

0.05

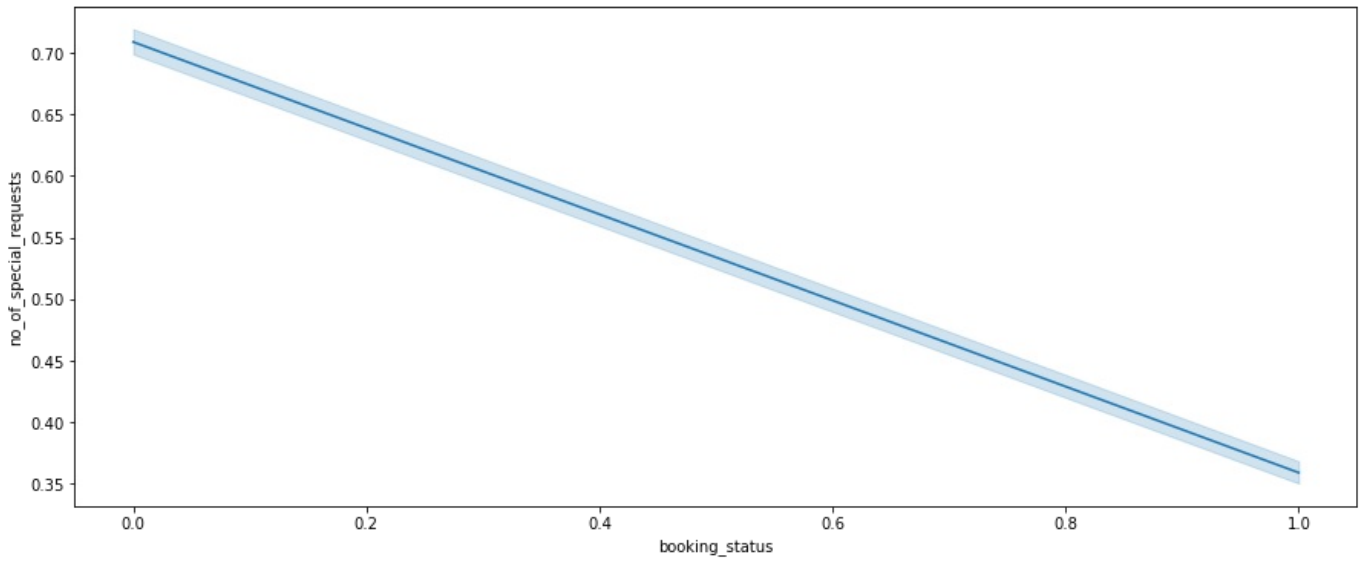

repeated_guest
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

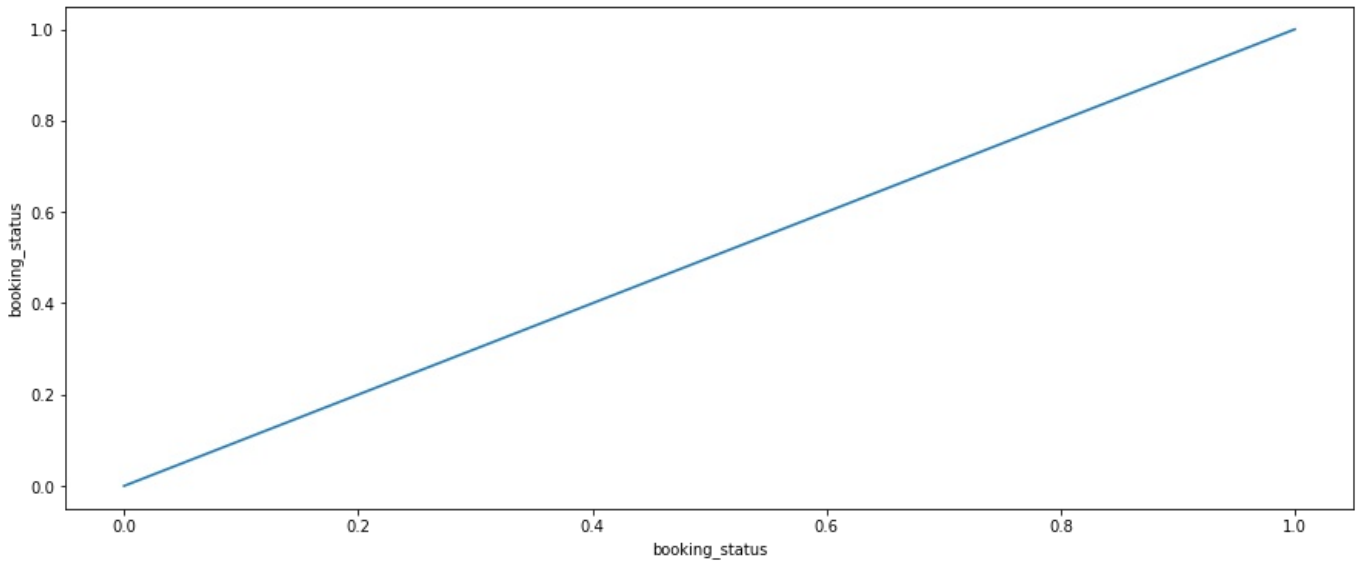

no_of_previous_cancellations
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.0
0.2
0.4
0.6
0.8
1.0
booking_status


no_of_previous_bookings_not_canceled
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.0
0.2
0.4
0.6
0.8
1.0
booking_status

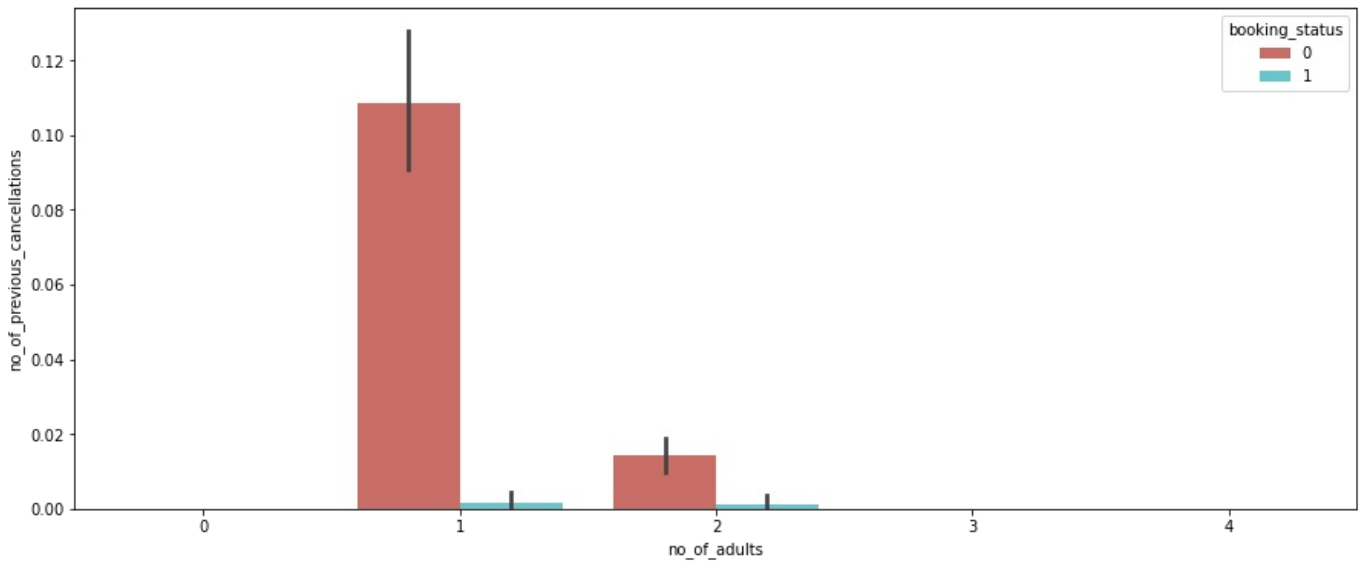

avg_price_per_room
112
110
108
106
104
102
100



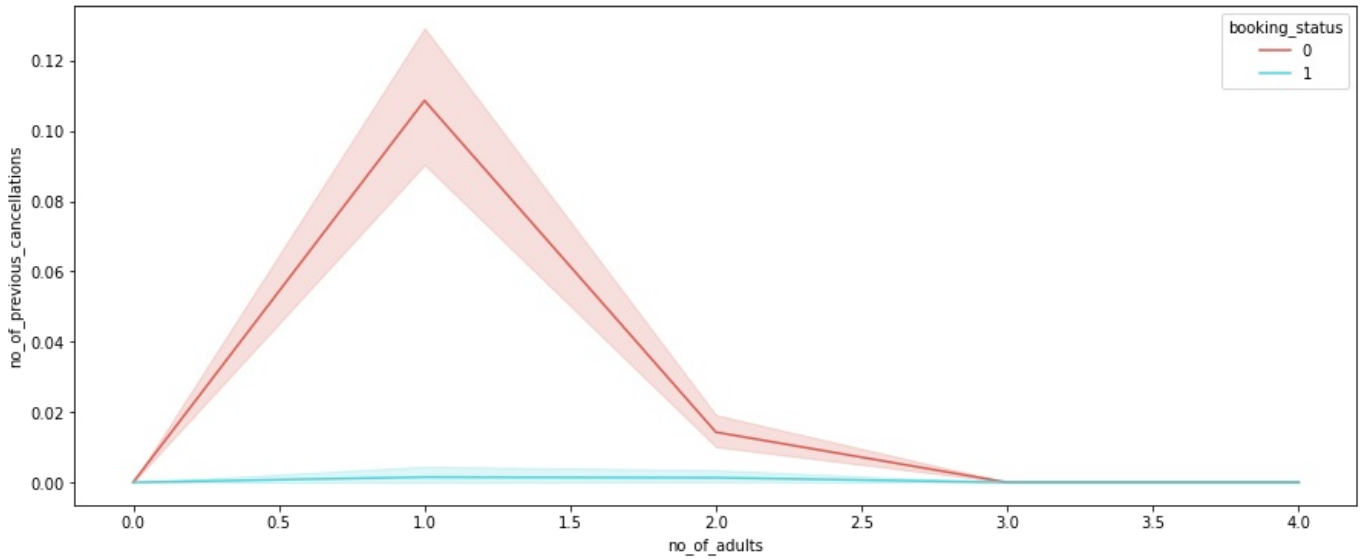




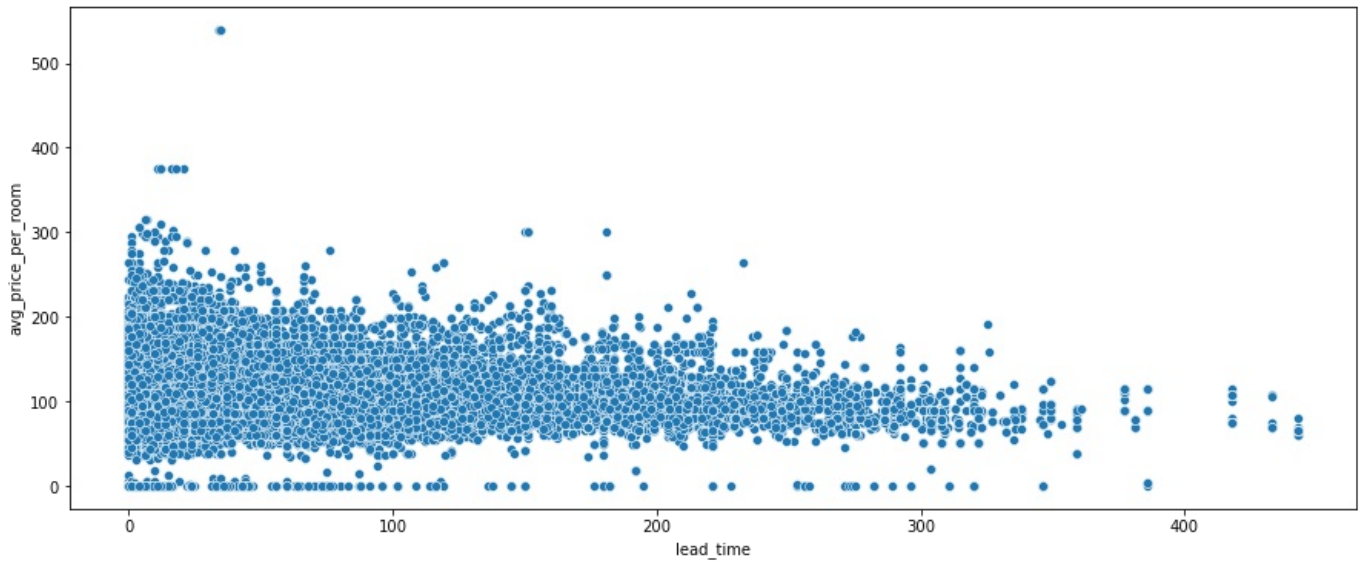
In [30]:
```
plt.figure(figsize=(15,6))
sns.barplot(x = df['no_of_adults'], y = df['no_of_previous_cancellations'], hue = df['booking_status'], palette =
plt.show()
```



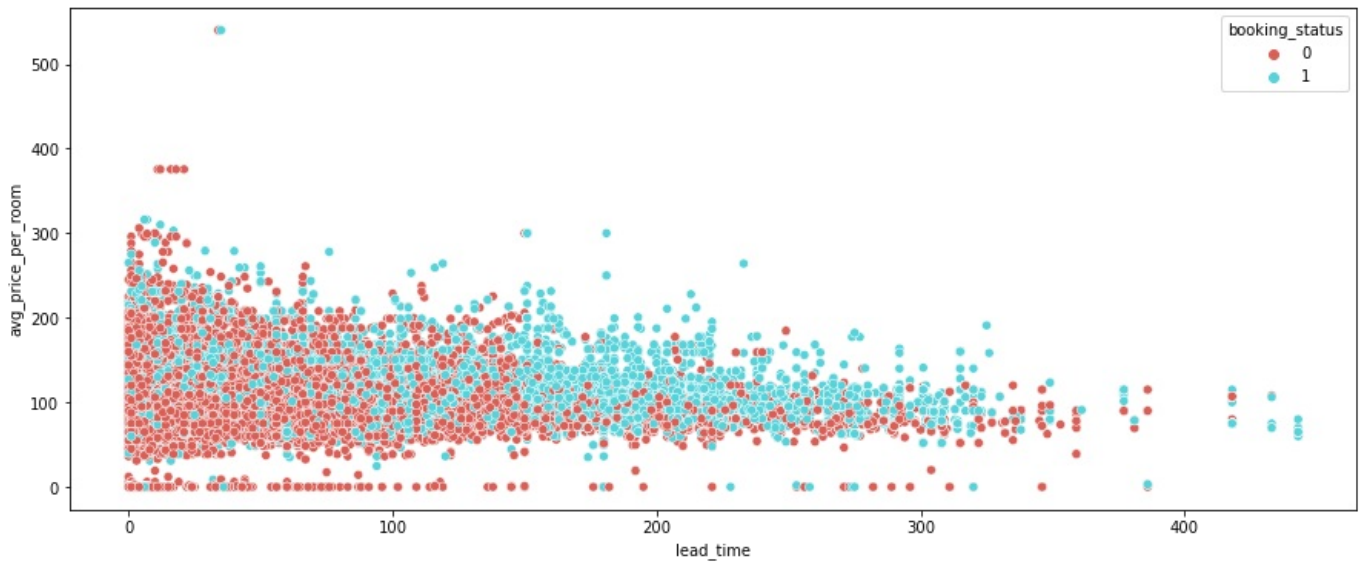
In [31]:
```
plt.figure(figsize=(15,6))
sns.lineplot(x = df['no_of_adults'], y = df['no_of_previous_cancellations'], hue = df['booking_status'], palette
plt.show()
```



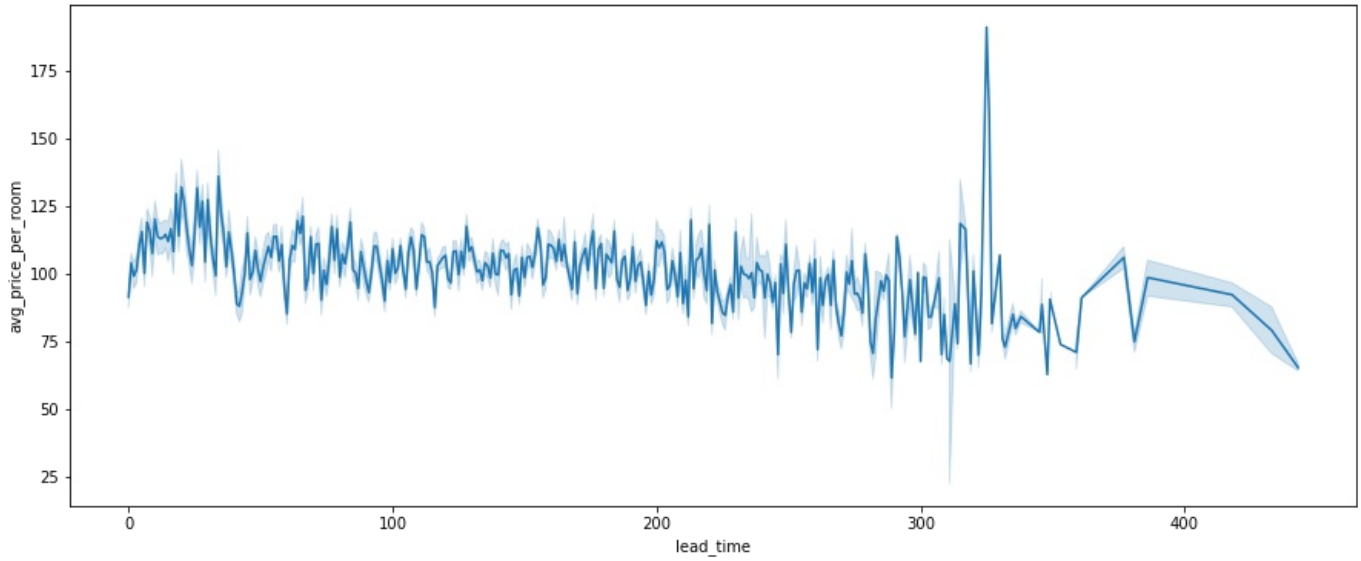
In [32]:
```
plt.figure(figsize=(15,6))
sns.scatterplot(x = df['lead_time'], y = df['avg_price_per_room'], palette = 'hls')
plt.show()
```



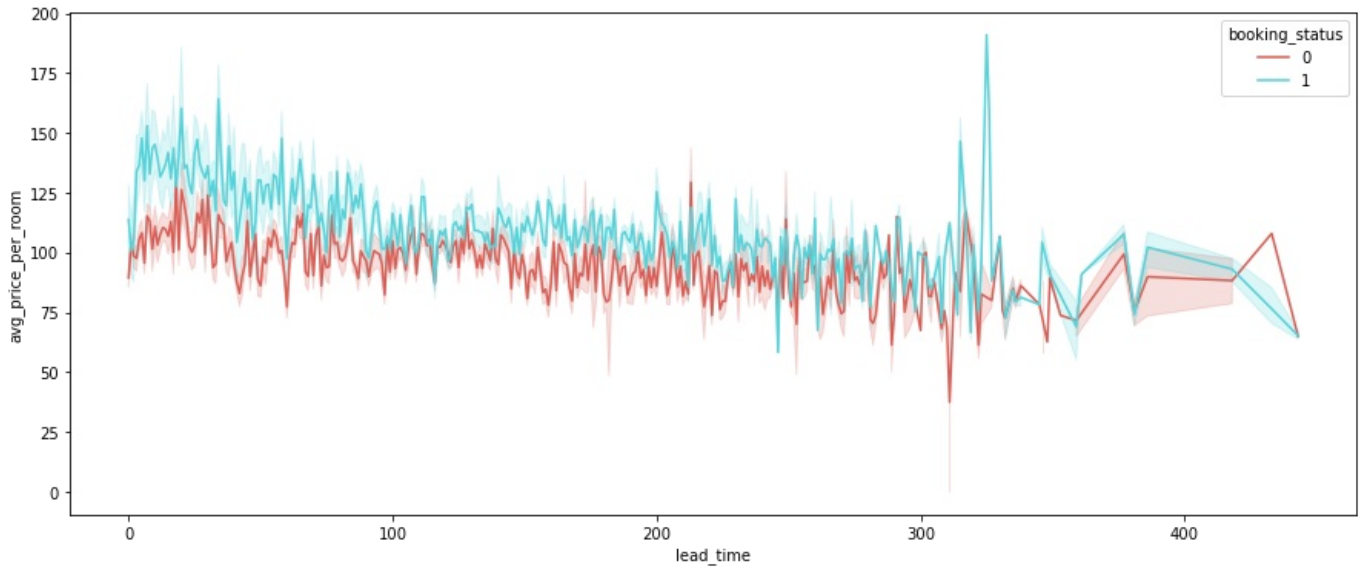
In [33]:
```
plt.figure(figsize=(15,6))
sns.scatterplot(x = df['lead_time'], y = df['avg_price_per_room'], hue = df['booking_status'], palette = 'hls')
plt.show()
```



In [34]:
```
plt.figure(figsize=(15,6))
sns.lineplot(x = df['lead_time'], y = df['avg_price_per_room'], palette = 'hls')
plt.show()
```



In [35]:
```
plt.figure(figsize=(15,6))
sns.lineplot(x = df['lead_time'], y = df['avg_price_per_room'], hue = df['booking_status'], palette = 'hls')
plt.show()
```



In [36]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
no_of_adults                           0.0
no_of_children                         0.0
no_of_weekend_nights                   2.0
no_of_week_nights                      2.0
type_of_meal_plan                      0.0
required_car_parking_space             0.0
room_type_reserved                     1.0
lead_time                            118.0
arrival_year                           0.0
arrival_month                          4.0
arrival_date                          16.0
market_segment_type                    1.0
repeated_guest                         0.0
no_of_previous_cancellations           0.0
no_of_previous_bookings_not_canceled   0.0
avg_price_per_room                    43.3
no_of_special_requests                 1.0
booking_status                         1.0
dtype: float64
```

In [37]:
```
df_new = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis = 1)]
```

In [38]:
```
df_new.shape
```

Out[38]:
```
(16701, 18)
```

In [39]:
```
df.shape
```

Out[39]:
```
(42100, 18)
```

In [40]:
```
import numpy as np

def detect_outliers_zscore(data, threshold=3):
    """
    Detect outliers using the Z-score method.

    Parameters:
    -----------
    data : array-like
        Input data.
    threshold : float, optional (default=3)
        The threshold value for the Z-score.

    Returns:
    --------
    A boolean array with True for the positions of outliers.
    """
    z_scores = np.abs((data - np.mean(data)) / np.std(data))
    return z_scores > threshold
```

In [41]:
```
import numpy as np

# Generate some random data
data = np.random.normal(loc=0, scale=1, size=100)

# Add an outlier
data[0] = 10

# Detect outliers using the Z-score method
outliers = detect_outliers_zscore(data)

# Print the indices of the outliers
print(np.where(outliers))
```

```
(array([0], dtype=int64),)
```

In [42]:
```
df_corr= df.corr()
```

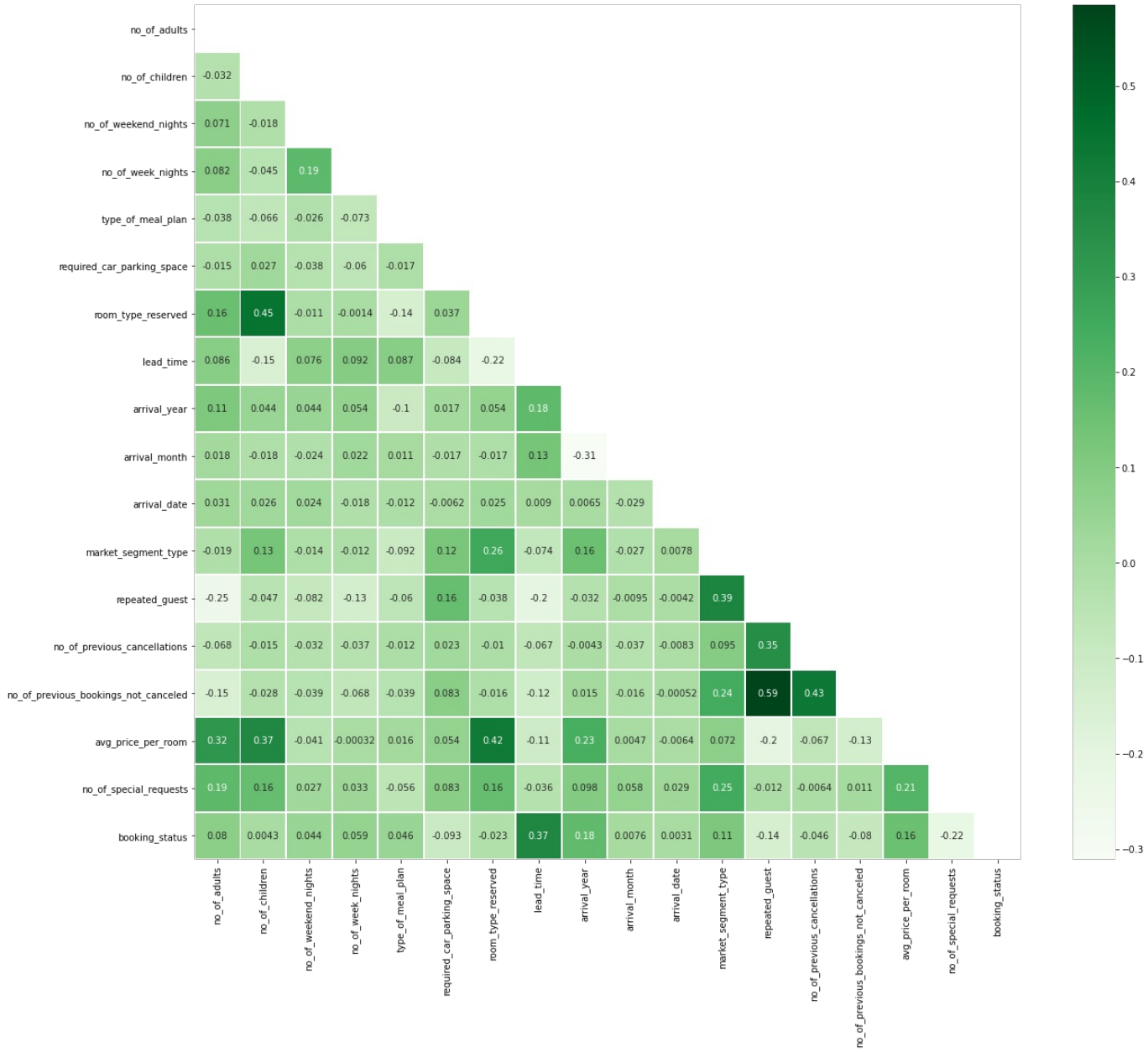
In [43]:
```
df_corr
```

Out[43]:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_ca
no_of_adults	1.000000	-0.031669	0.071417	0.082347	-0.038004	
no_of_children	-0.031669	1.000000	-0.018453	-0.044519	-0.066252	
no_of_weekend_nights	0.071417	-0.018453	1.000000	0.188200	-0.025613	
no_of_week_nights	0.082347	-0.044519	0.188200	1.000000	-0.072577	
type_of_meal_plan	-0.038004	-0.066252	-0.025613	-0.072577	1.000000	
required_car_parking_space	-0.014628	0.026916	-0.037671	-0.060155	-0.016814	
room_type_reserved	0.160038	0.446015	-0.010716	-0.001363	-0.139149	
lead_time	0.085969	-0.151382	0.076146	0.091935	0.087172	
arrival_year	0.111617	0.043867	0.043962	0.054174	-0.100922	
arrival_month	0.018059	-0.018420	-0.024495	0.022302	0.010629	
arrival_date	0.030865	0.026428	0.024276	-0.018058	-0.011635	
market_segment_type	-0.019303	0.131166	-0.013913	-0.011606	-0.092322	
repeated_guest	-0.249581	-0.047146	-0.082078	-0.132183	-0.059776	
no_of_previous_cancellations	-0.068073	-0.014755	-0.031709	-0.036995	-0.012222	
no_of_previous_bookings_not_canceled	-0.152986	-0.027848	-0.039084	-0.068045	-0.039064	
avg_price_per_room	0.320082	0.370766	-0.041083	-0.000323	0.015905	

no_of_special_requests	0.185158	0.164969	0.026503	0.032639	-0.056030
booking_status	0.079760	0.004260	0.044279	0.058506	0.045980

In [44]:
```
plt.figure(figsize=(20, 17))
matrix = np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap="Greens");
plt.show()
```



In [45]:
```
X = df.drop('booking_status',axis=1)
y =df['booking_status']
```

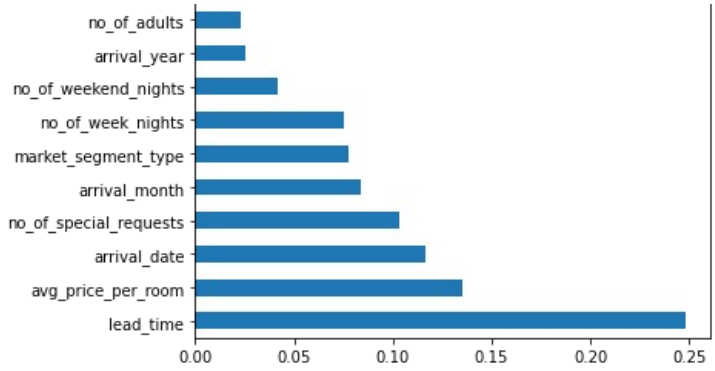
In [46]:
```
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
```

```
[0.02325276 0.01078019 0.04154382 0.07496878 0.01614862 0.00748892
 0.02277477 0.2483604  0.02519298 0.08407937 0.11654404 0.07753804
 0.01145534 0.00037451 0.00098236 0.13531048 0.10320463]
```

In [48]:
```
X = df.iloc[:,:-1]
```

In [49]:
```
import pandas as pd
```

In [50]:
```
feat_importances = pd.Series(model.feature_importances_,index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



In [51]:
```
top_10 = pd.DataFrame({'Feature Importance': feat_importances.nlargest(10)})
```

In [52]:
```
top_10
```

Out[52]:

	Feature Importance
lead_time	0.248360
avg_price_per_room	0.135310
arrival_date	0.116544
no_of_special_requests	0.103205
arrival_month	0.084079
market_segment_type	0.077538
no_of_week_nights	0.074969
no_of_weekend_nights	0.041544
arrival_year	0.025193
no_of_adults	0.023253

In [53]:
```
X = X[['lead_time', 'avg_price_per_room', 'arrival_date', 'no_of_special_requests', 'arrival_month', 'market_segm
       'no_of_week_nights', 'arrival_year', 'no_of_adults']]
```

In [54]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [55]:
```
X = scaler.fit_transform(X)
```

In [56]:
```
from  sklearn.model_selection import train_test_split
```

In [57]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.25, random_state= 42, stratify = y)
```

In [58]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classifica
```

In [59]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Out[59]:
```
LogisticRegression()
```

In [60]:
```
y_pred = lr.predict(X_test)
```

In [61]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7593349168646081
```

In [62]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[62]:
```
DecisionTreeClassifier()
```

In [63]:
```
y_pred = dt.predict(X_test)
```

In [64]:
```
print("Accuracy:", accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7406175771971496
```

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

Out[65]:
```
RandomForestClassifier()
```

In [66]:
```
y_pred = rfc.predict(X_test)
```

In [67]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7972446555819478
```

In [68]:
```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

In [69]:
```
folds = StratifiedKFold(n_splits = 5, shuffle = True, random_state = 40)
```

In [70]:
```
def grid_search(model,folds,params,scoring):
    grid_search = GridSearchCV(model,
                               cv=folds,
                               param_grid=params,
                               scoring=scoring,
                               n_jobs=-1, verbose=1)
    return grid_search
```

In [71]:
```
def print_best_score_params(model):
    print("Best Score: ", model.best_score_)
    print("Best Hyperparameters: ", model.best_params_)
```

In [75]:
```
##log_reg = LogisticRegression()
##log_params = {'C': [0.01, 1, 10],
          'penalty': ['l1', 'l2'],
          'solver': ['liblinear','newton-cg','saga']
         }
##grid_search_log = grid_search(log_reg, folds, log_params, scoring=None)
##grid_search_log.fit(X_train, y_train)
##print_best_score_params(grid_search_log)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   16.9s
[Parallel(n_jobs=-1)]: Done  90 out of  90 | elapsed:   24.1s finished
```

```
Best Score:  0.7605700712589074
Best Hyperparameters:  {'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}
```

In [72]:
```
lr = LogisticRegression(C = 1, penalty = 'l2', solver = 'liblinear')
lr.fit(X_train, y_train)
```

Out[72]:
```
LogisticRegression(C=1, solver='liblinear')
```

In [73]:
```
y_pred = lr.predict(X_test)
```

In [74]:
```
print("Accuracy:",accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7595249406175772
```

In [75]:
```
dtc = DecisionTreeClassifier(random_state=40)
dtc_params = {
    'max_depth': [5,10,20,30],
    'min_samples_leaf': [5,10,20,30]
}
grid_search_dtc = grid_search(dtc, folds, dtc_params, scoring='roc_auc_ovr')
```

```
grid_search_dtc.fit(X_train, y_train)
print_best_score_params(grid_search_dtc)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    9.0s
```

```
Best Score:  0.8687916926957321
Best Hyperparameters:  {'max_depth': 10, 'min_samples_leaf': 30}
```

```
[Parallel(n_jobs=-1)]: Done  80 out of  80 | elapsed:   54.4s finished
```

In [76]:
```
dt = DecisionTreeClassifier(max_depth = 10, min_samples_leaf = 30)
dt.fit(X_train, y_train)
```

Out[76]:
```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=30)
```

In [77]:
```
y_pred = dt.predict(X_test)
```

In [78]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8036104513064133
```

In [79]:
```
rfc = RandomForestClassifier(random_state=40, n_jobs = -1,oob_score=True)
rfc_params = {'max_depth': [10,20,30,40],
          'min_samples_leaf': [5,10,15,20,30],
          'n_estimators': [100,200,500,700]
        }
grid_search_rfc = grid_search(rfc, folds, rfc_params, scoring='roc_auc_ovr')
grid_search_rfc.fit(X_train, y_train)
print('OOB SCORE :',grid_search_rfc.best_estimator_.oob_score_)
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:  6.4min
[Parallel(n_jobs=-1)]: Done 400 out of 400 | elapsed: 30.9min finished
```

```
OOB SCORE : 0.8157719714964371
```

In [80]:
```
print_best_score_params(grid_search_rfc)
```

```
Best Score:  0.886033376027453
Best Hyperparameters:  {'max_depth': 20, 'min_samples_leaf': 10, 'n_estimators': 700}
```

In [81]:
```
rfc = RandomForestClassifier(max_depth = 20, min_samples_leaf = 10, n_estimators = 700)
rfc.fit(X_train, y_train)
```

Out[81]:
```
RandomForestClassifier(max_depth=20, min_samples_leaf=10, n_estimators=700)
```

In [82]:
```
ypred = rfc.predict(X_test)
```

In [83]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8036104513064133
```

In [86]:
```
from sklearn.naive_bayes import GaussianNB
```

In [87]:
```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Out[87]:
```
GaussianNB()
```

In [88]:
```
y_pred = gnb.predict(X_test)
```

In [89]:
```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.7351068883610451
```

In [90]:
```
param_grid = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]
}
```

In [91]:
```
grid_search = GridSearchCV(gnb, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Out[91]:
```
GridSearchCV(cv=5, estimator=GaussianNB(),
             param_grid={'var_smoothing': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05,
                                           0.0001, 0.001, 0.01, 0.1]})
```

In [92]:
```
print("Best hyperparameters: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best hyperparameters:  {'var_smoothing': 1e-09}
Best accuracy:  0.7320348376880443
```

In [93]:
```
gnb = GaussianNB(var_smoothing = 1e-09)
gnb.fit(X_train,y_train)
```

Out[93]:
```
GaussianNB()
```

In [94]:
```
y_pred = gnb.predict(X_test)
```

In [95]:
```
accuracy = accuracy_score(y_test,y_pred)
print('Accuracy:',accuracy)
```

```
Accuracy: 0.7351068883610451
```

In [96]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [97]:
```
gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)
```

Out[97]:
```
GradientBoostingClassifier()
```

In [98]:
```
y_pred = gbc.predict(X_test)
```

In [99]:
```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.8058907363420428
```

In [100…
```
import xgboost as xgb
```

In [101…
```
model = xgb.XGBClassifier()
model.fit(X_train,y_train)
```

```
[15:12:34] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[101…
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
```

```
monotone_constraints='()', n_estimators=100, n_jobs=8,
num_parallel_tree=1, predictor='auto', random_state=0,
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='exact', validate_parameters=1, verbosity=None)
```

In [102…]
```
y_pred = model.predict(X_test)
```

In [103…]
```
accuracy = accuracy_score(y_test,y_pred)
print('Accuracy:',accuracy)
```

```
Accuracy: 0.8158669833729216
```

In [ ]: