```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('/content/hotel_bookings.csv')
```

```
df.head()
```

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

5 rows × 32 columns

```
df2 = df.copy()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

```
df.describe()
```

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

```
print(df.isnull().sum())
```

```
hotel                               0
is_canceled                         0
lead_time                           0
arrival_date_year                   0
arrival_date_month                  0
arrival_date_week_number            0
arrival_date_day_of_month           0
stays_in_weekend_nights             0
stays_in_week_nights                0
adults                              0
children                            4
babies                              0
meal                                0
country                           488
market_segment                      0
distribution_channel                0
is_repeated_guest                   0
previous_cancellations              0
previous_bookings_not_canceled      0
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
agent                           16340
company                        112593
days_in_waiting_list                0
customer_type                       0
adr                                 0
required_car_parking_spaces         0
total_of_special_requests           0
reservation_status                  0
reservation_status_date             0
dtype: int64
```

```
percent_missing = df.isnull().sum() * 100 / len(df)
percent_missing
```

	0
hotel	0.000000
is_canceled	0.000000
lead_time	0.000000
arrival_date_year	0.000000
arrival_date_month	0.000000
arrival_date_week_number	0.000000
arrival_date_day_of_month	0.000000
stays_in_weekend_nights	0.000000
stays_in_week_nights	0.000000
adults	0.000000
children	0.003350
babies	0.000000
meal	0.000000
country	0.408744
market_segment	0.000000
distribution_channel	0.000000
is_repeated_guest	0.000000
previous_cancellations	0.000000
previous_bookings_not_canceled	0.000000
reserved_room_type	0.000000
assigned_room_type	0.000000
booking_changes	0.000000
deposit_type	0.000000
agent	13.686238
company	94.306893
days_in_waiting_list	0.000000
customer_type	0.000000
adr	0.000000
required_car_parking_spaces	0.000000
total_of_special_requests	0.000000
reservation_status	0.000000
reservation_status_date	0.000000

dtype: float64

Cleaning

```
#fill numerical columns with mean
df['adr'].fillna(df['adr'].mean(), inplace=True)
df['required_car_parking_spaces'].fillna(df['required_car_parking_spaces'].mean(), inplace=True)
df['total_of_special_requests'].fillna(df['total_of_special_requests'].mean(), inplace=True)
```

```
/tmp/ipython-input-1403421669.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  df['adr'].fillna(df['adr'].mean(), inplace=True)
/tmp/ipython-input-1403421669.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  df['required_car_parking_spaces'].fillna(df['required_car_parking_spaces'].mean(), inplace=True)
/tmp/ipython-input-1403421669.py:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col
```

```
  df['total_of_special_requests'].fillna(df['total_of_special_requests'].mean(), inplace=True)
```

```
#fill country & agent with mode
df['country'].fillna(df['country'].mode()[0], inplace=True)
df['agent'].fillna(df['agent'].mode()[0], inplace=True)
# For columns with 1 missing value, use forward fill or backward fill
df['reservation_status'].fillna(method='ffill', inplace=True)
df['reservation_status_date'].fillna(method='ffill', inplace=True)
```

```
/tmp/ipython-input-131549627.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col

  df['country'].fillna(df['country'].mode()[0], inplace=True)
/tmp/ipython-input-131549627.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col

  df['agent'].fillna(df['agent'].mode()[0], inplace=True)
/tmp/ipython-input-131549627.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col

  df['reservation_status'].fillna(method='ffill', inplace=True)
/tmp/ipython-input-131549627.py:5: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use
  df['reservation_status'].fillna(method='ffill', inplace=True)
/tmp/ipython-input-131549627.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col

  df['reservation_status_date'].fillna(method='ffill', inplace=True)
/tmp/ipython-input-131549627.py:6: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use
  df['reservation_status_date'].fillna(method='ffill', inplace=True)
```

```
df.drop(columns='company', inplace=True)
```

```
print('the data is clean will done roowiii ;D' if df.isnull().sum().sum() == 0 else 'there are still missing values')
```

```
there are still missing values
```

changing data type

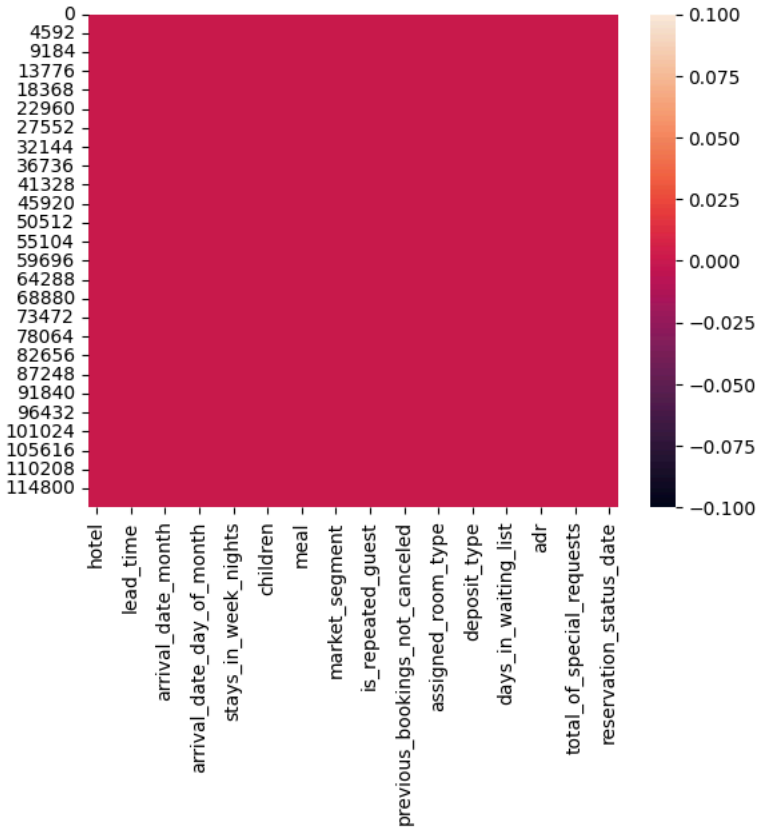
```
df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'])
df['children'] = df['children'].fillna(0).astype(int)
df['agent'] = df['agent'].astype(str)
```

```
df['arrival_date_month'].unique()
```

```
array(['July', 'August', 'September', 'October', 'November', 'December',
       'January', 'February', 'March', 'April', 'May', 'June'],
      dtype=object)
```

```
sns.heatmap(df.isnull())
```

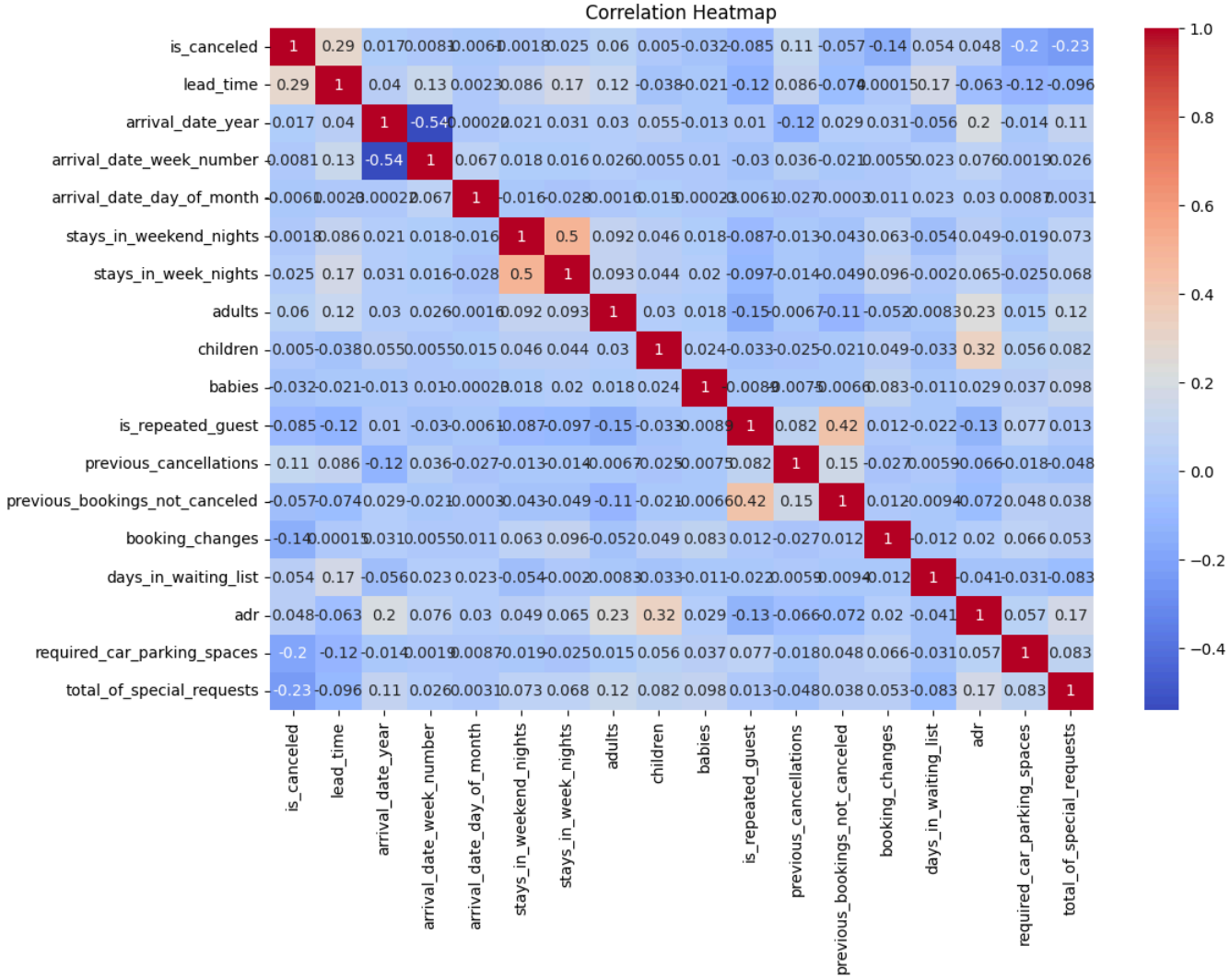
```
<Axes: >
```




```
print(df.select_dtypes(include='object').columns)
```

```
Index(['hotel', 'arrival_date_month', 'meal', 'country', 'market_segment',
       'distribution_channel', 'reserved_room_type', 'assigned_room_type',
       'deposit_type', 'agent', 'customer_type', 'reservation_status'],
      dtype='object')
```

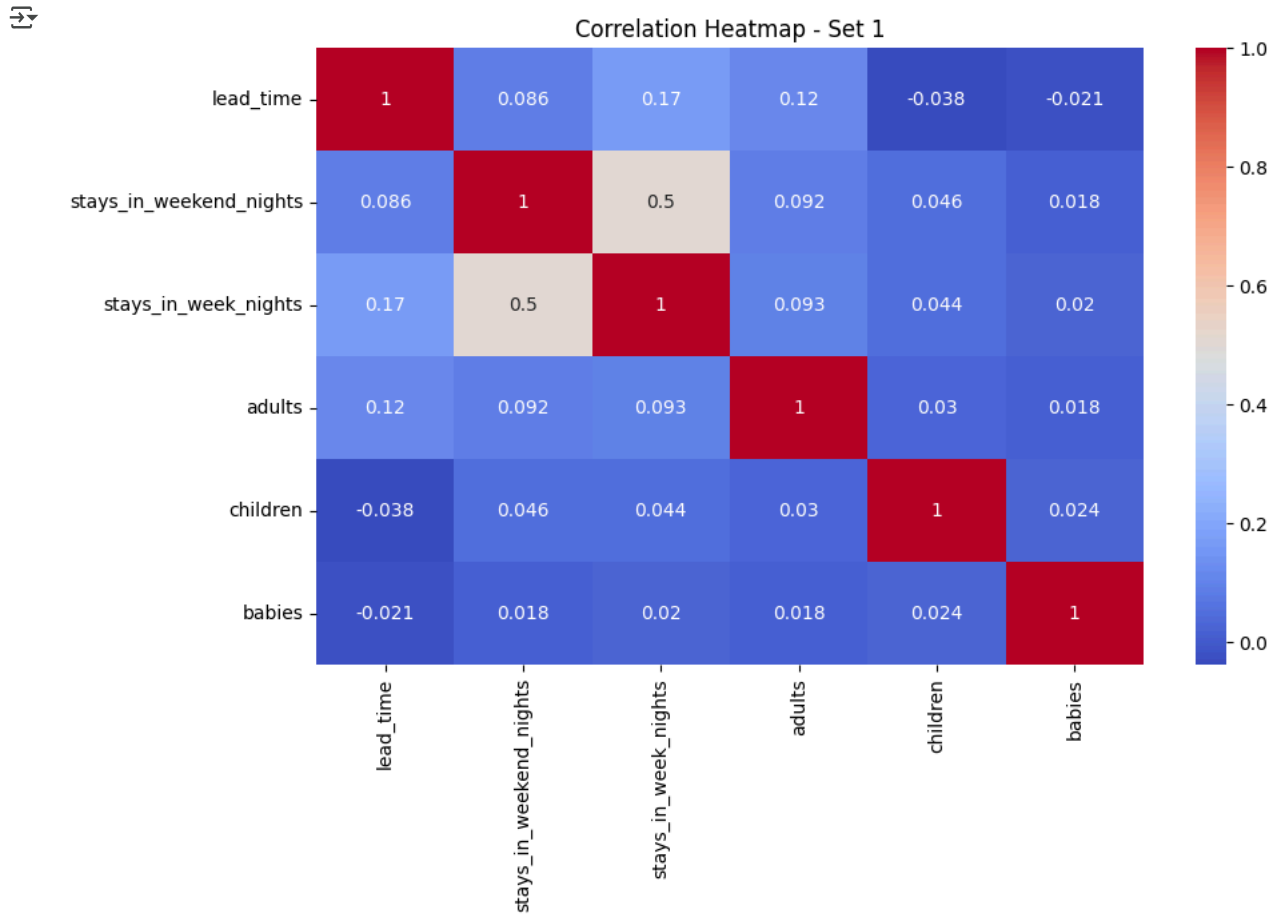
```
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

plt.figure(figsize=(12,8))
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



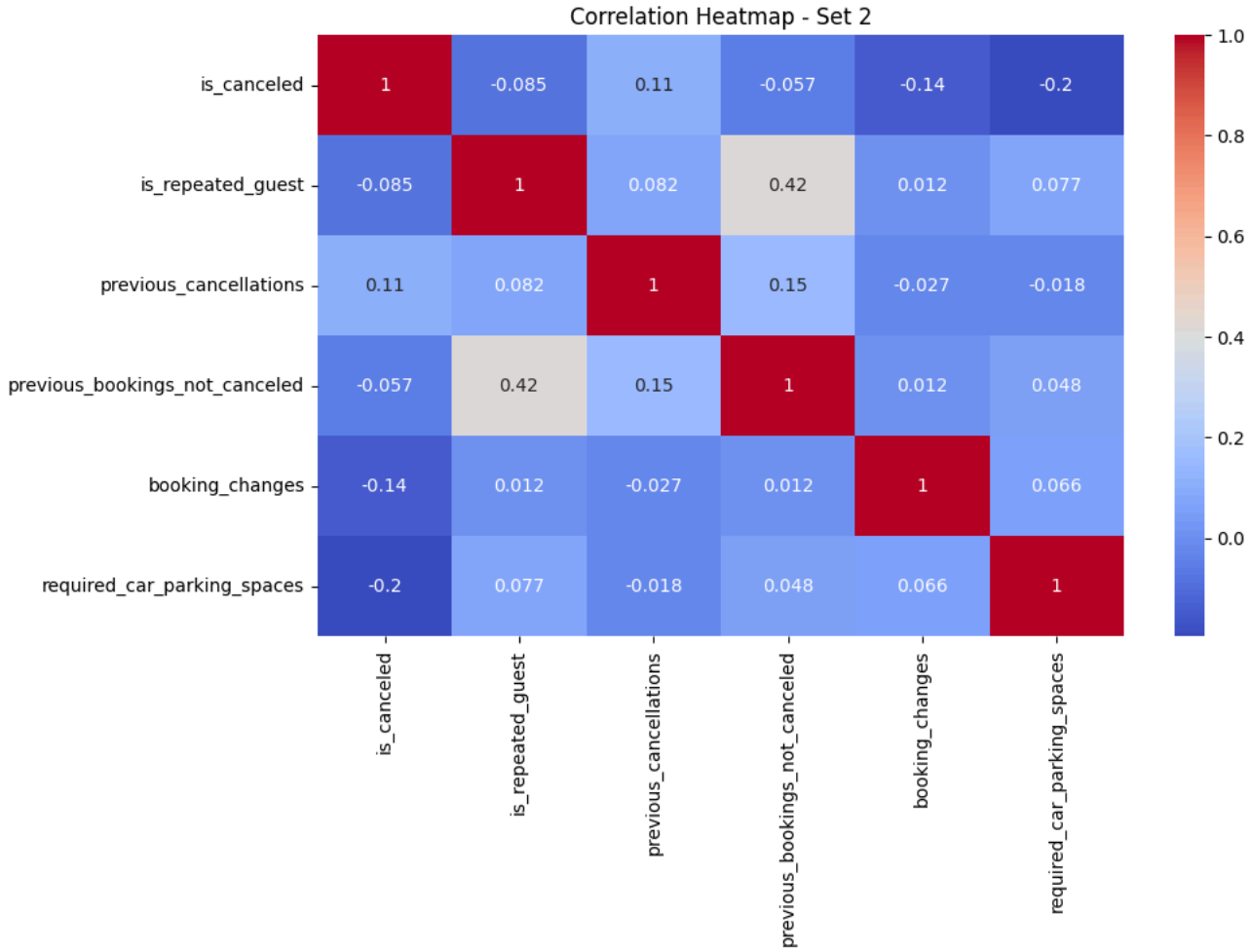

```
numeric_cols = df.select_dtypes(include='number').columns.tolist()
print(numeric_cols)

group1 = ['lead_time', 'stays_in_weekend_nights', 'stays_in_week_nights', 'adults', 'children', 'babies']
group2 = ['is_canceled', 'is_repeated_guest', 'previous_cancellations', 'previous_bookings_not_canceled', 'booking_changes', 'required_
group3 = ['adr', 'days_in_waiting_list', 'total_of_special_requests']
```

```
['is_canceled', 'lead_time', 'arrival_date_year', 'arrival_date_week_number', 'arrival_date_day_of_month', 'stays_in_weekend_nights
```

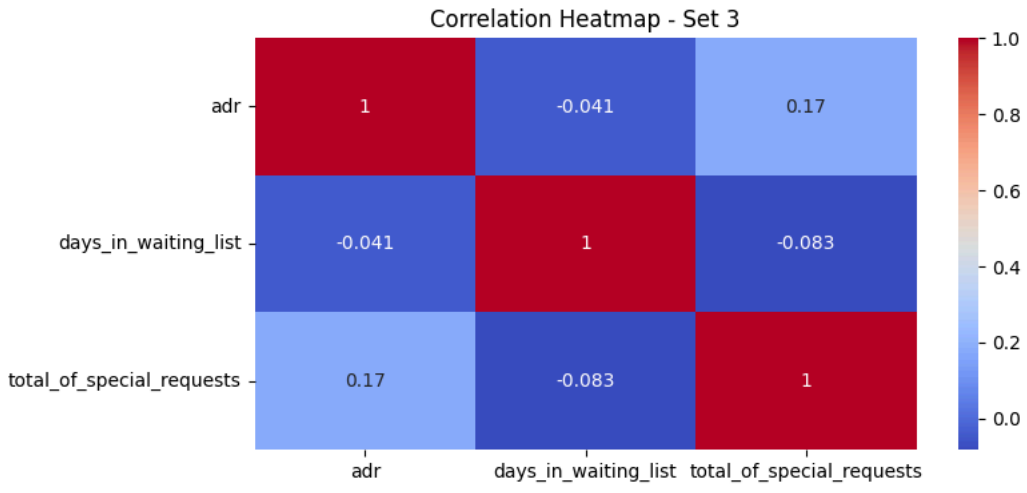
```
plt.figure(figsize=(10, 6))
sns.heatmap(df[group1].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap - Set 1')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.heatmap(df[group2].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap - Set 2')
plt.show()
```

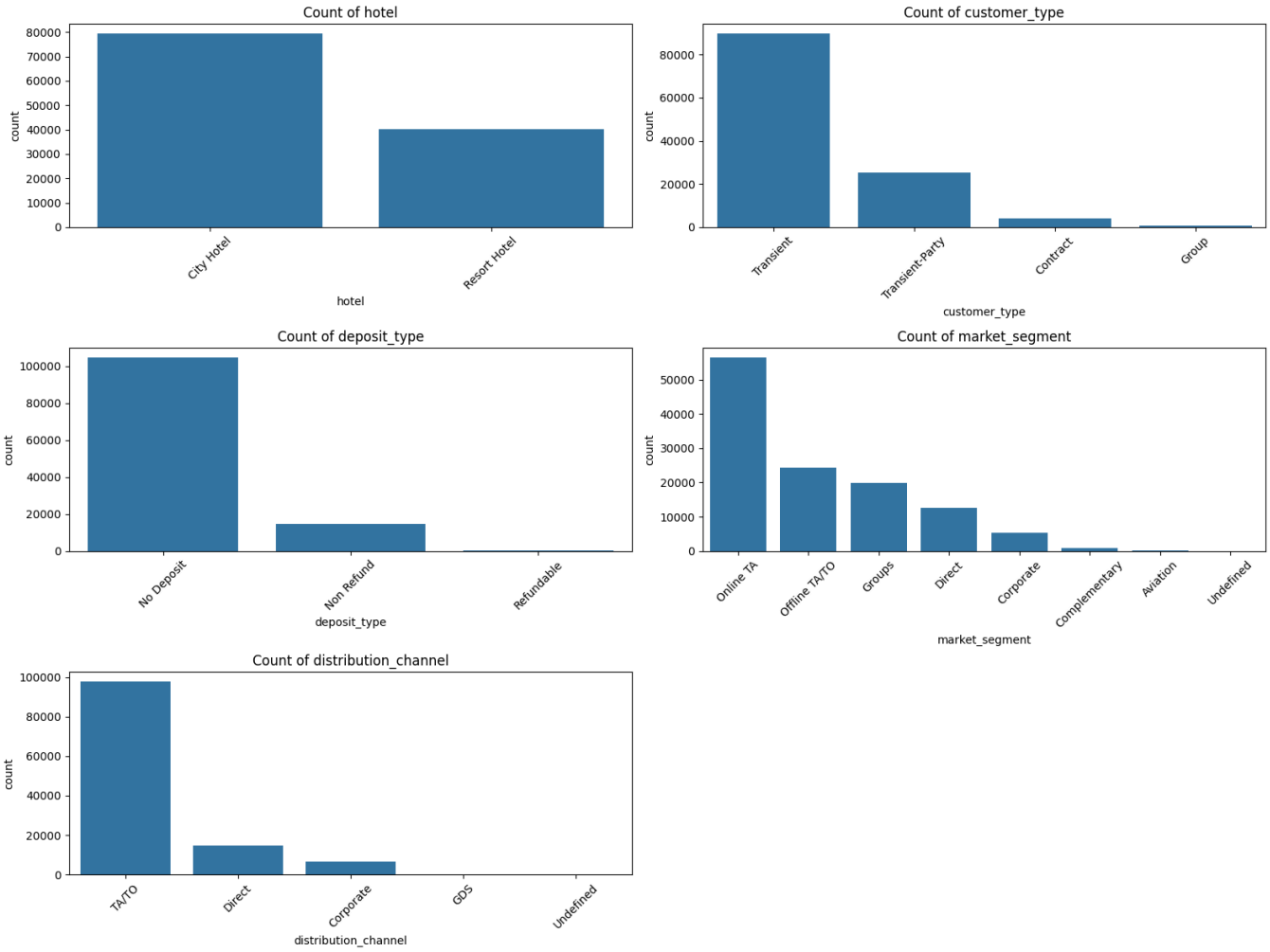



```
plt.figure(figsize=(8, 4))
sns.heatmap(df[group3].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap - Set 3')
plt.show()
```

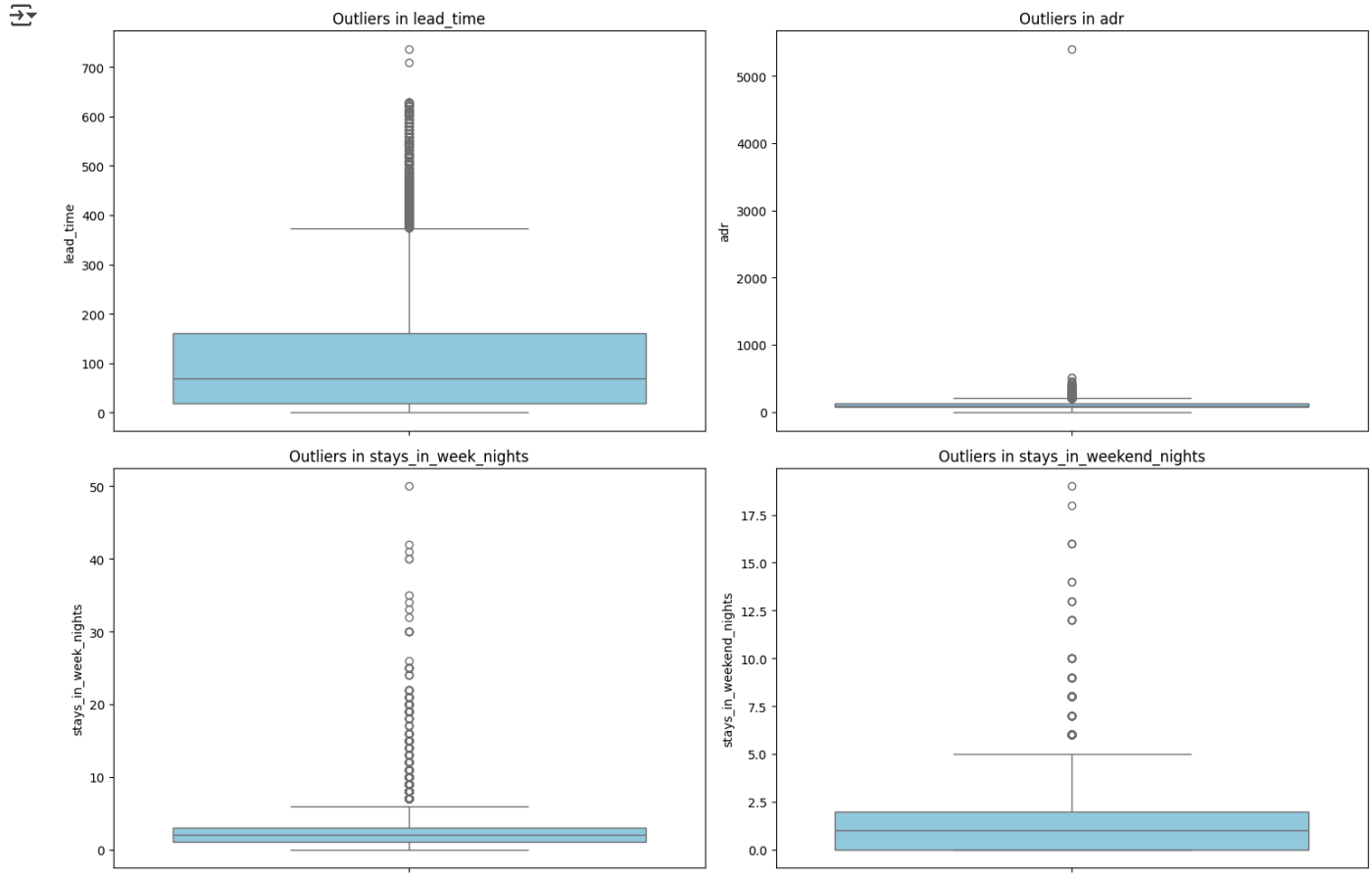



```
cat_cols = ['hotel', 'customer_type', 'deposit_type', 'market_segment', 'distribution_channel']

fig, axs = plt.subplots(3, 2, figsize=(16, 12))

for i, col in enumerate(cat_cols):
    row = i // 2
    col_idx = i % 2
    sns.countplot(data=df, x=col, ax=axs[row, col_idx], order=df[col].value_counts().index)
    axs[row, col_idx].set_title(f'Count of {col}')
    axs[row, col_idx].tick_params(axis='x', rotation=45)

if len(cat_cols) % 2 != 0:
    axs[2, 1].axis('off')
```

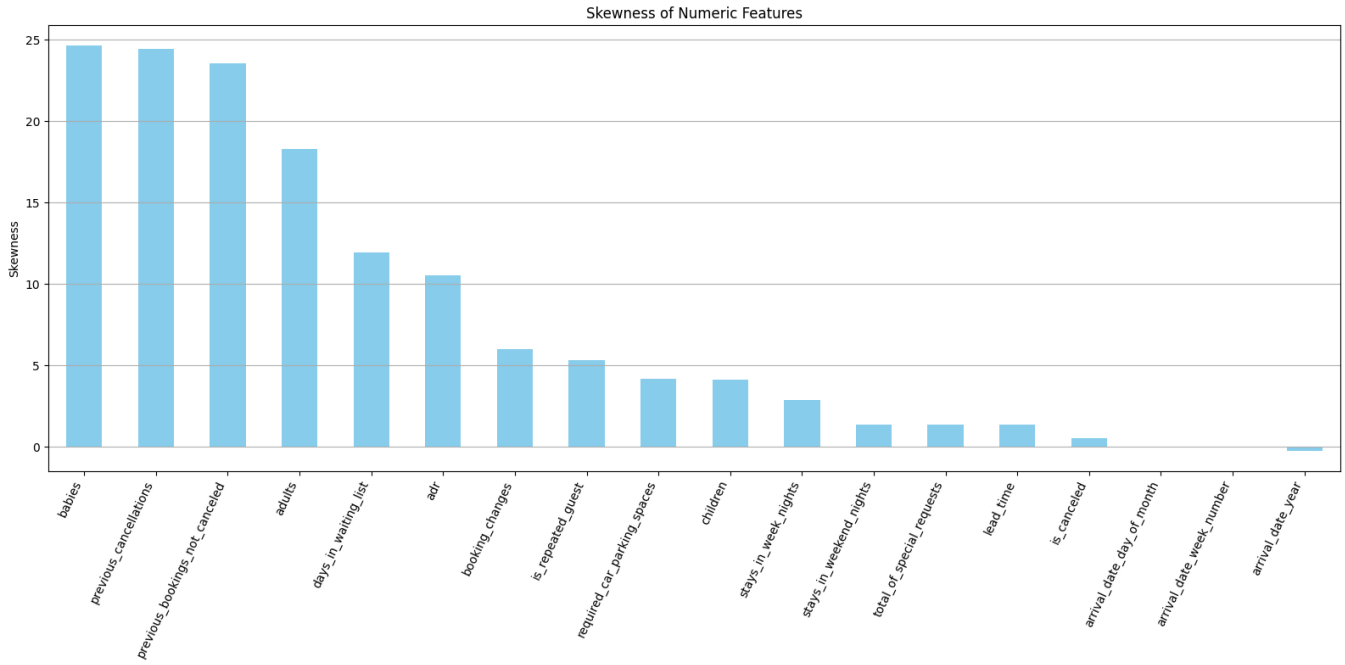
```
plt.tight_layout()
plt.show()
```




```
columns_to_check = ['lead_time', 'adr', 'stays_in_week_nights', 'stays_in_weekend_nights']
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
axes = axes.flatten()

for i, col in enumerate(columns_to_check):
    sns.boxplot(data=df, y=col, ax=axes[i], color='skyblue')
    axes[i].set_title(f'Outliers in {col}')

plt.tight_layout()
plt.show()
```

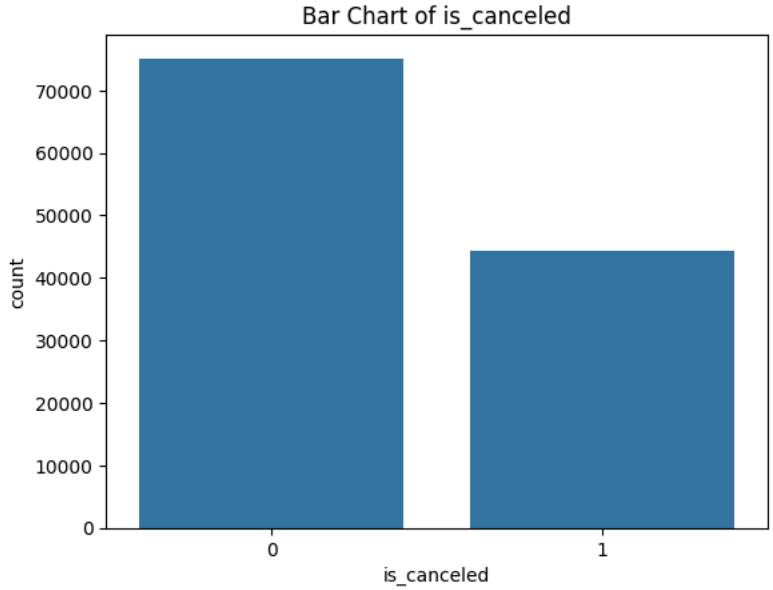



```
plt.figure(figsize=(16, 8))
skew_vals.plot(kind='bar', color='skyblue')
plt.title('Skewness of Numeric Features')
plt.ylabel('Skewness')
plt.xticks(rotation=65, ha='right')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

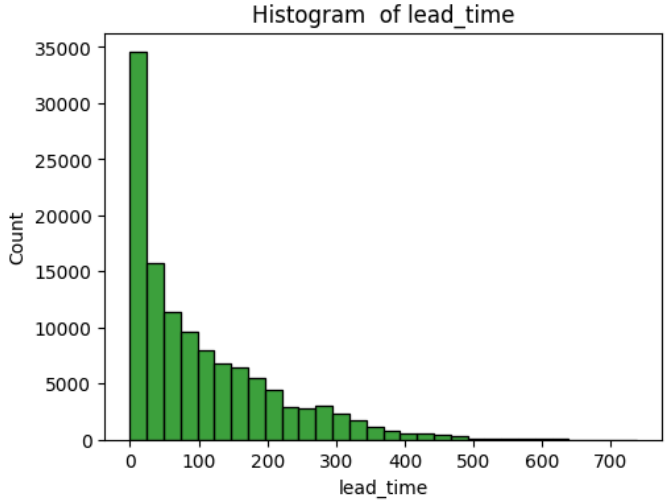


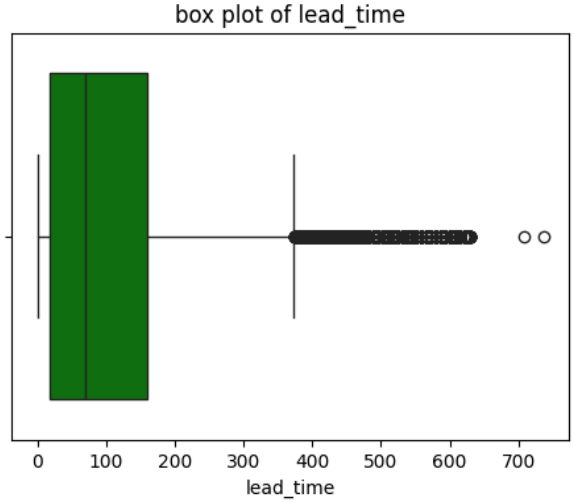

```
numeric_columns = df.select_dtypes(include=["float64", "int64"]).columns.tolist()
for col in numeric_columns:
    unique_vals = df[col].nunique()
    print(f"{col} has {unique_vals} unique values.")
    if unique_vals <= 20:
        sns.countplot(data=df, x=col)
        plt.title(f"Bar Chart of {col}")
        plt.show()
    else:
        plt.figure(figsize=(12, 4))
        plt.subplot(1,2,1)
        sns.histplot(df[col], bins=30, color="g")
        plt.title(f"Histogram  of {col}")

        plt.subplot(1,2,2)
        sns.boxplot(x=df[col], color="g")
        plt.title(f"box plot of {col}")
        plt.show()
```

```
is_canceled has 2 unique values.
```

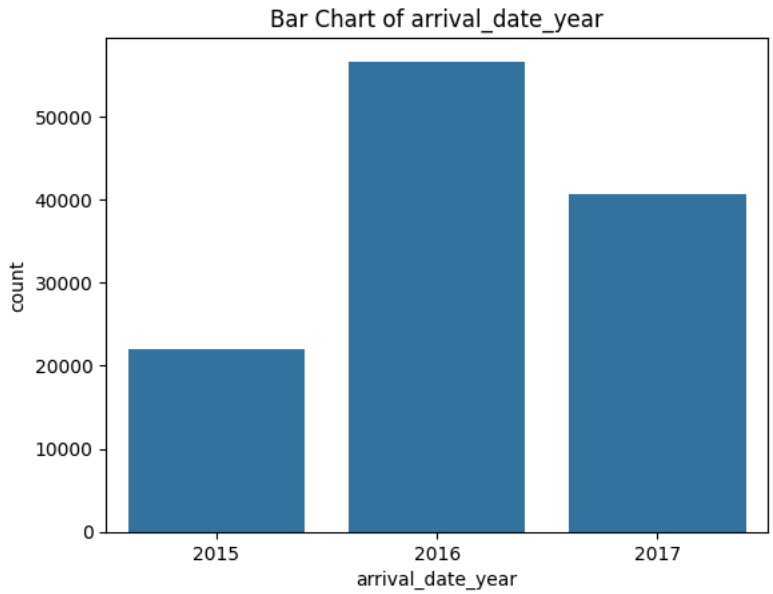



```
lead_time has 479 unique values.
```

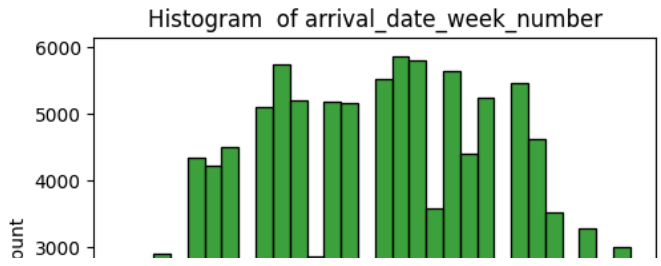


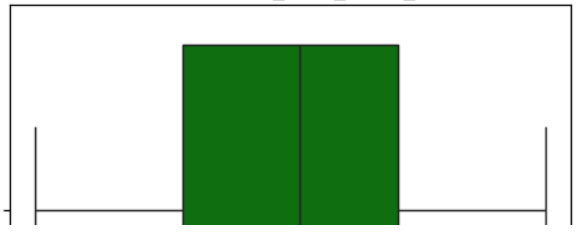


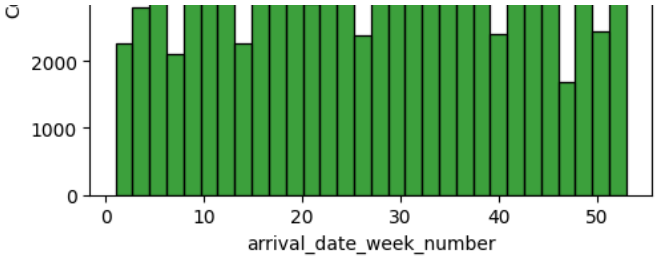

```
arrival_date_year has 3 unique values.
```

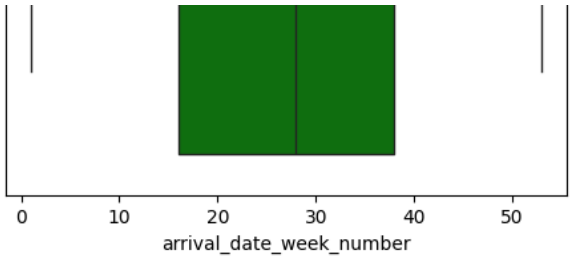



```
arrival_date_week_number has 53 unique values.
```

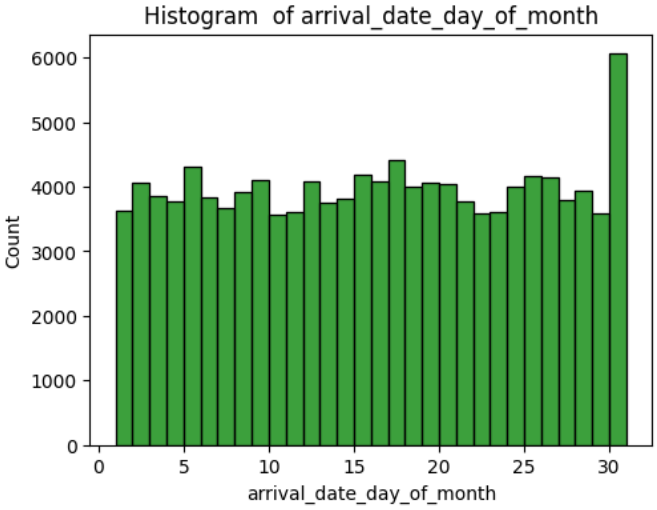


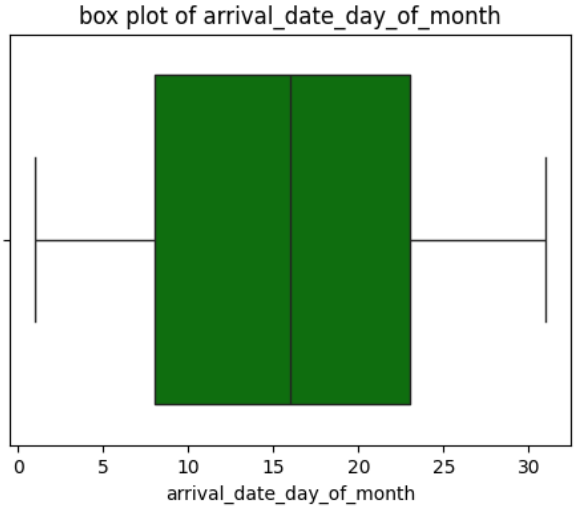




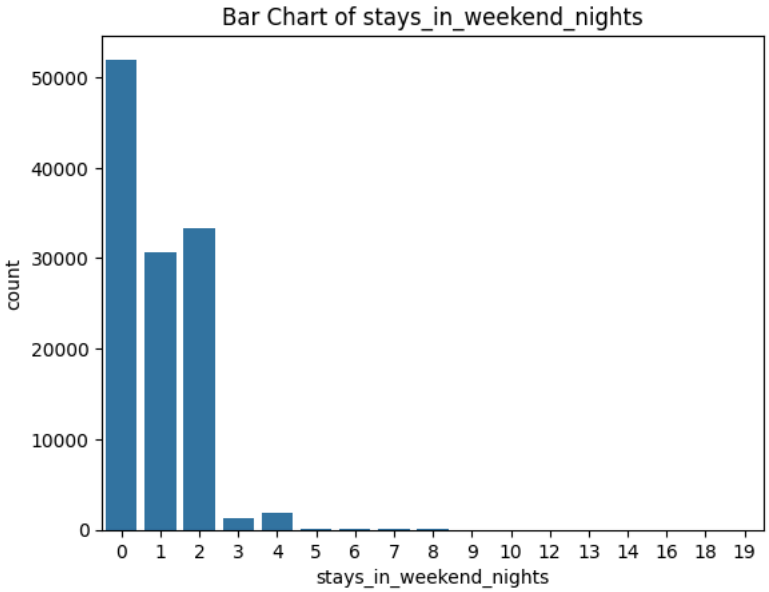


arrival_date_day_of_month has 31 unique values.

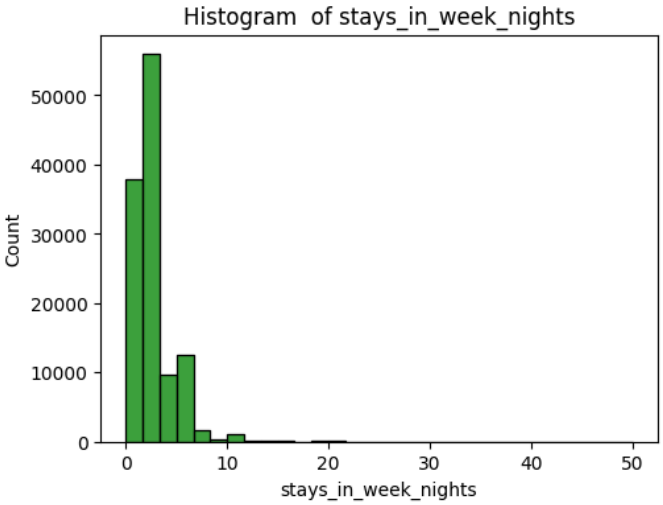


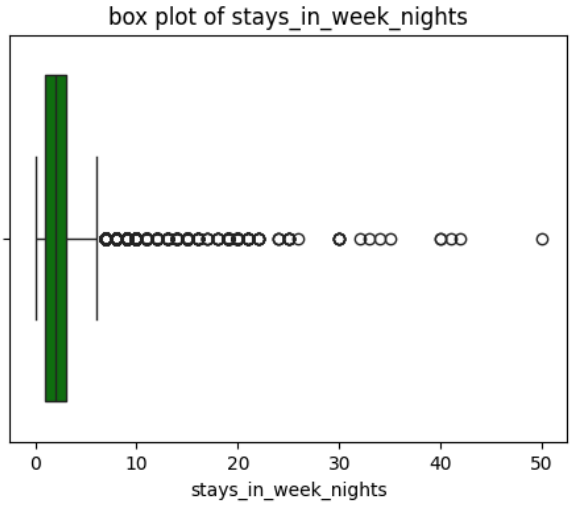


stays_in_weekend_nights has 17 unique values.

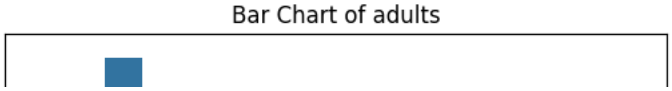


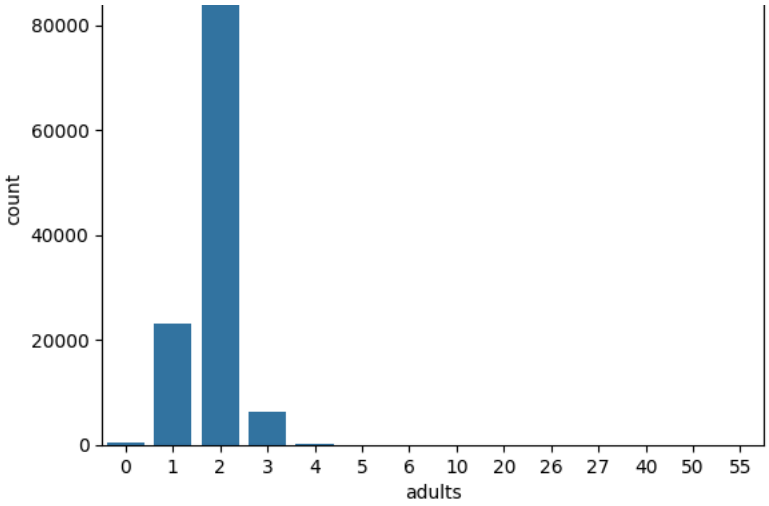
stays_in_week_nights has 35 unique values.



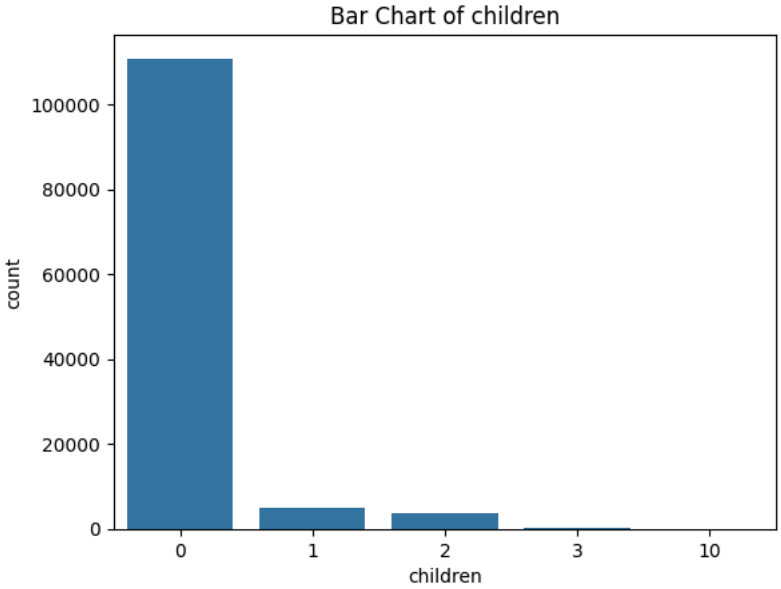


adults has 14 unique values.

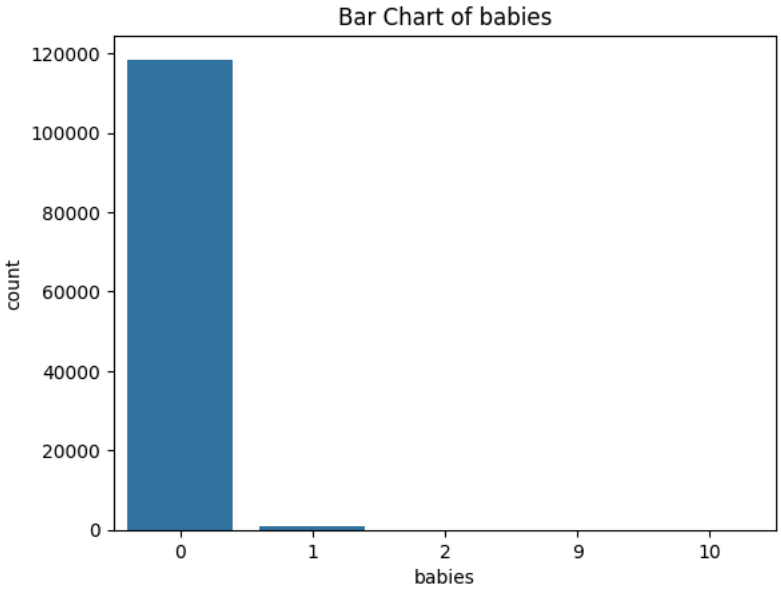




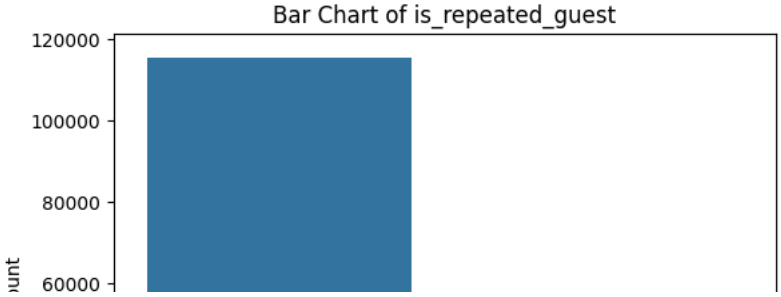
children has 5 unique values.

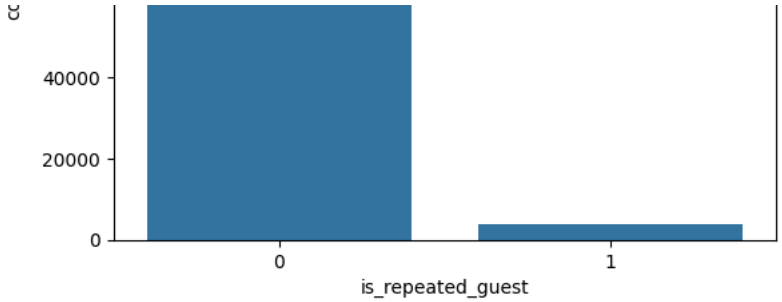


babies has 5 unique values.

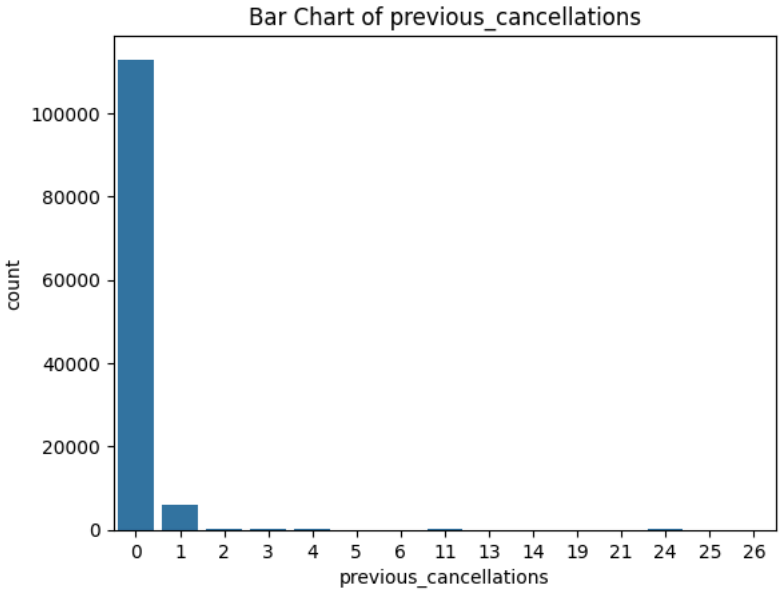


is_repeated_guest has 2 unique values.

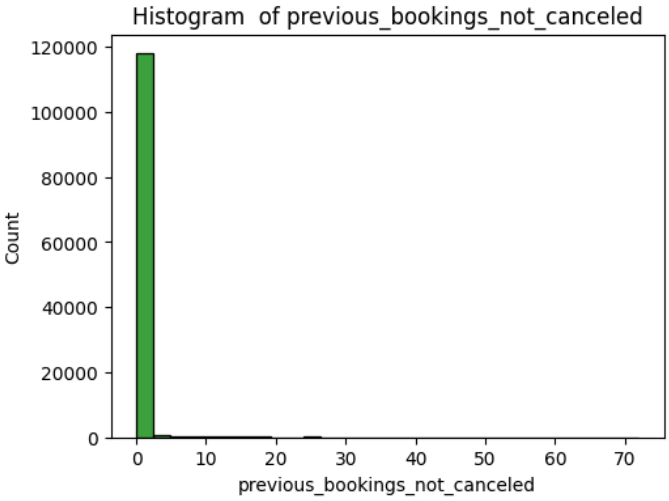


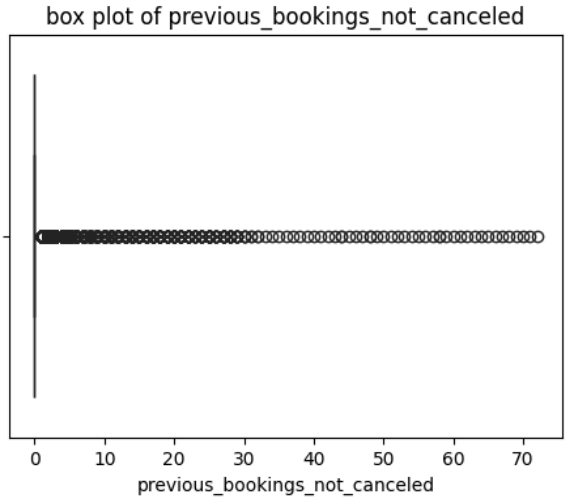


previous_cancellations has 15 unique values.

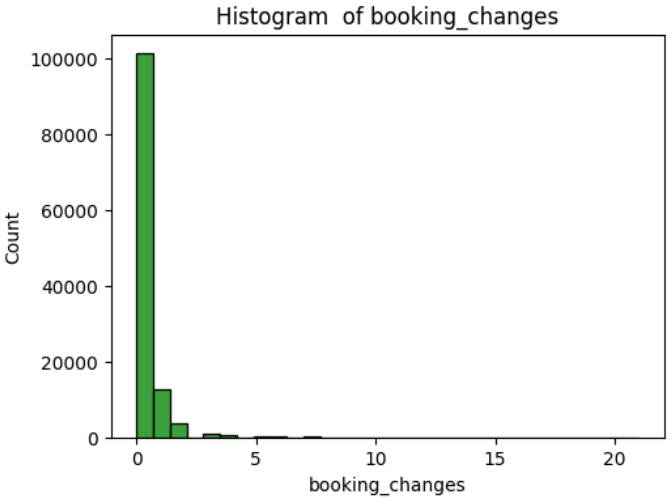


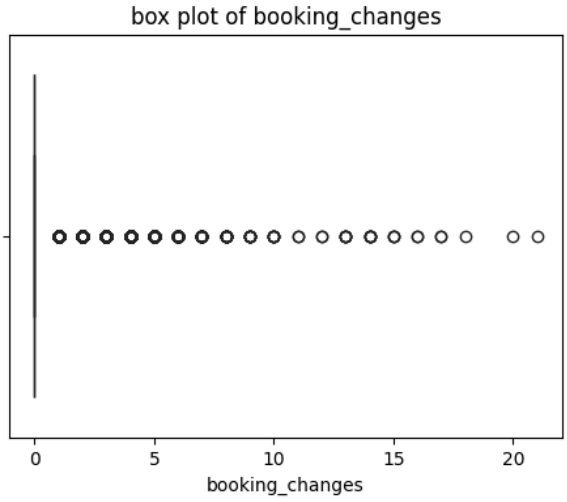
previous_bookings_not_canceled has 73 unique values.



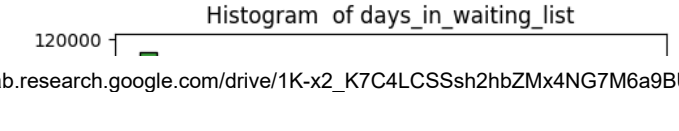


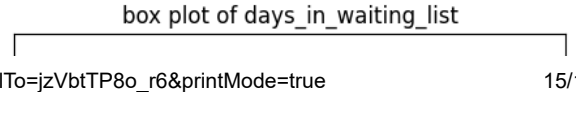
booking_changes has 21 unique values.

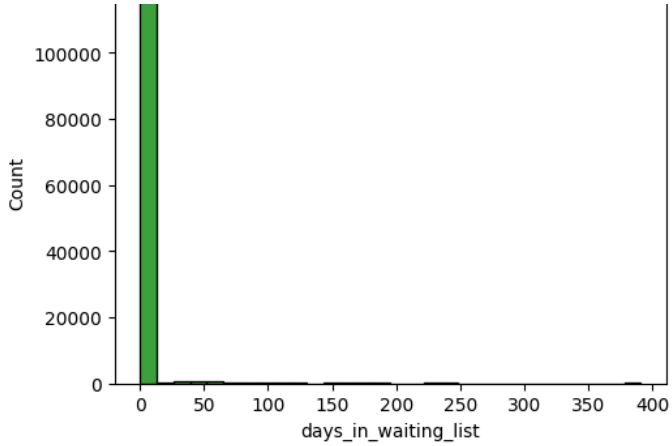


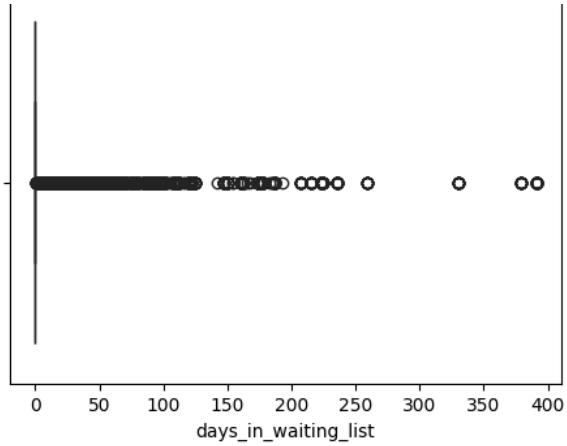


days_in_waiting_list has 128 unique values.

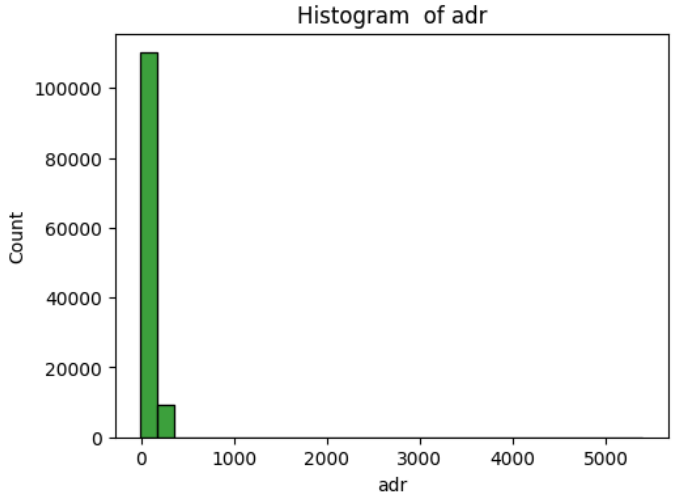


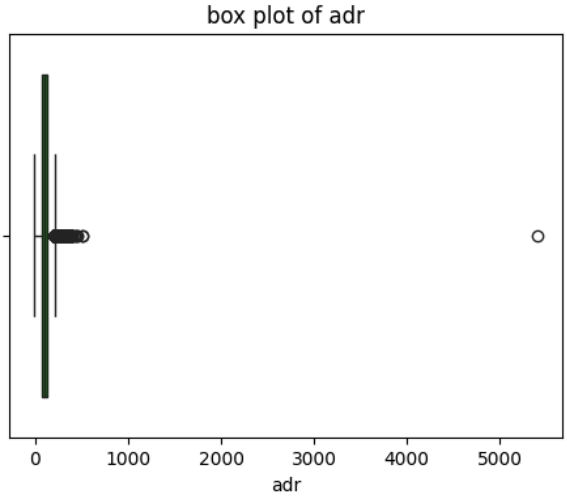




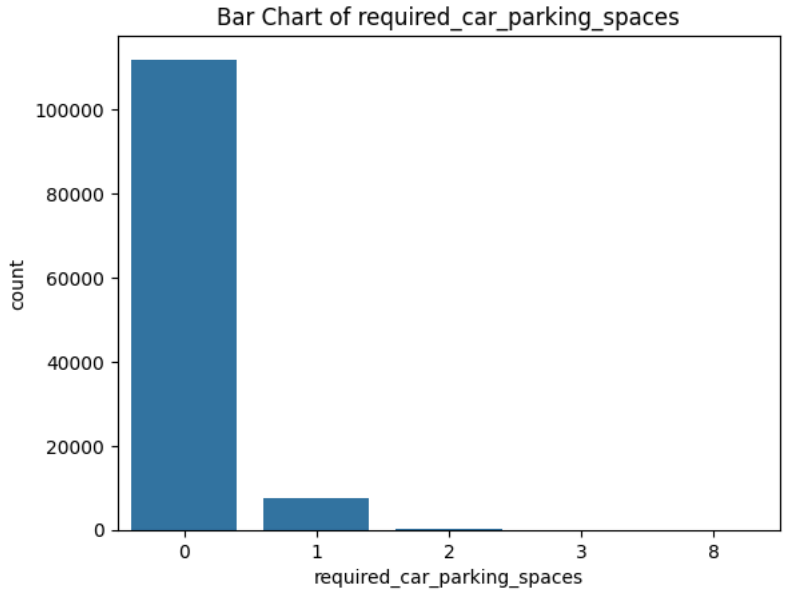



```
adr has 8879 unique values.
```

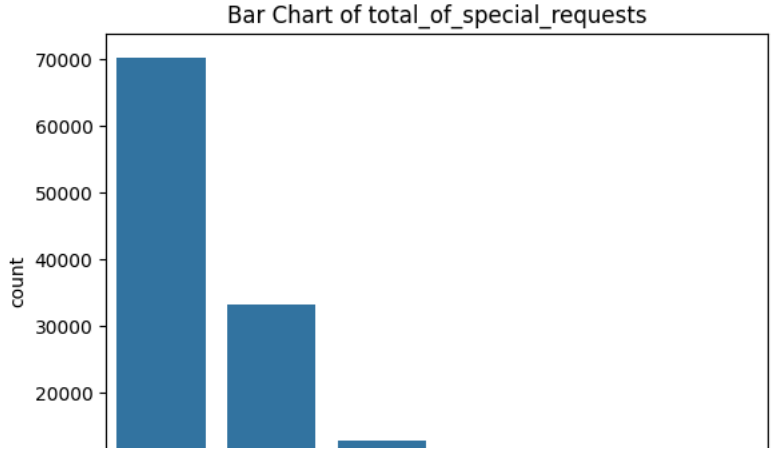


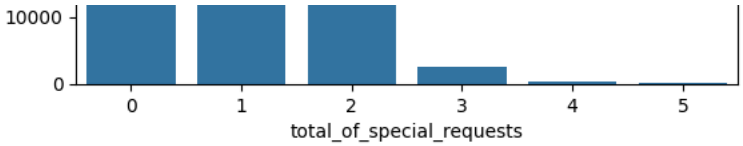



```
required_car_parking_spaces has 5 unique values.
```




```
total_of_special_requests has 6 unique values.
```





DETECT OUTLIER

```
from scipy.stats import zscore
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
z_scores = zscore(df[numeric_cols])
outliers = (abs(z_scores) > 3)

outlier_count = outliers.sum(axis=0)
print(outlier_count)
```

```
[   0 1454    0    0    0 2199 1669  481 3729  917 3810  317  936 1570
 1871 1138 7416 2877]
```

```
def detect_outliers_iqr(df):
    outliers = pd.DataFrame()
    numeric_df = df.select_dtypes(include=['number'])

    for column in numeric_df.columns:
        Q1 = numeric_df[column].quantile(0.25)
        Q3 = numeric_df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers[column] = (numeric_df[column] < lower_bound) | (numeric_df[column] > upper_bound)

    return outliers

outliers_iqr = detect_outliers_iqr(df)

print("Outliers detected using IQR:")
print(outliers_iqr.sum())
```

```
Outliers detected using IQR:
is_canceled                         0
lead_time                        3005
arrival_date_year                   0
arrival_date_week_number            0
```