In [100…

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

In [101…

```
df = pd.read_csv("airbnb data.csv")
df
```

```
C:\Users\KIIT\AppData\Local\Temp\ipykernel_4108\1292283203.py:1: DtypeWarning: Colum
ns (25) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("airbnb data.csv")
```

Out[101…

	id	NAME	host id	host_identity_verified	host name	neighbou
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Br
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Mar
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Elise	Mar
3	1002755	NaN	85098326012	unconfirmed	Garry	Br
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Mar
...	...	...	...	...	...	
102594	6092437	Spare room in Williamsburg	12312296767	verified	Krik	Br
102595	6092990	Best Location near Columbia U	77864383453	unconfirmed	Mifan	Mar
102596	6093542	Comfy, bright room in Brooklyn	69050334417	unconfirmed	Megan	Br
102597	6094094	Big Studio-One Stop from Midtown	11160591270	unconfirmed	Christopher	C
102598	6094647	585 sf Luxury Studio	68170633372	unconfirmed	Rebecca	Mar

102599 rows × 26 columns

Data Cleaning

In [101…
```
print('Initailly number of duplicates in the data:',df.duplicated().sum())
```

```
Initailly number of duplicates in the data: 541
```

In [101…
```
df.drop_duplicates(inplace = True)
```

In [101…
```
print("After removing number of duplicates",df.duplicated().sum())
```

```
After removing number of duplicates 0
```

In [102…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 102058 entries, 0 to 102057
Data columns (total 26 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              102058 non-null  int64
 1   NAME                            101808 non-null  object
 2   host id                         102058 non-null  int64
 3   host_identity_verified          101769 non-null  object
 4   host name                       101654 non-null  object
 5   neighbourhood group             102029 non-null  object
 6   neighbourhood                   102042 non-null  object
 7   lat                             102050 non-null  float64
 8   long                            102050 non-null  float64
 9   country                         101526 non-null  object
 10  country code                    101927 non-null  object
 11  instant_bookable                101953 non-null  object
 12  cancellation_policy             101982 non-null  object
 13  room type                       102058 non-null  object
 14  Construction year               101844 non-null  float64
 15  price                           101811 non-null  object
 16  service fee                     101785 non-null  object
 17  minimum nights                  101658 non-null  float64
 18  number of reviews               101875 non-null  float64
 19  last review                     86226 non-null   object
 20  reviews per month               86240 non-null   float64
 21  review rate number              101739 non-null  float64
 22  calculated host listings count  101739 non-null  float64
 23  availability 365                101610 non-null  float64
 24  house_rules                     50216 non-null   object
 25  license                         2 non-null       object
dtypes: float64(9), int64(2), object(15)
memory usage: 21.0+ MB
```

In [102…
```
# last review is actually date in our data but in info here it is object that means
```

In [102…
```
df['last review'] = pd.to_datetime(df['last review'], errors='coerce')
```

In [102…
```
#If the column has invalid dates, use errors='coerce' to replace them with NaT (Not
```

In [102…
```
#constructure year is in float -----> datetime
```

In [103…
```
#df['Construction year'] =  pd.to_datetime(df['Construction year'])
#df['Construction year'] = df['Construction year'].dt.year
```

In [103…
```
# type of price (object) ----> to (float)
```

In [103…
```
# df['price'].astype(int) this shows error because this also contains some non nume
```

In [103…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 102058 entries, 0 to 102057
Data columns (total 26 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              102058 non-null  int64
 1   NAME                            101808 non-null  object
 2   host id                         102058 non-null  int64
 3   host_identity_verified          101769 non-null  object
 4   host name                       101654 non-null  object
 5   neighbourhood group             102029 non-null  object
 6   neighbourhood                   102042 non-null  object
 7   lat                             102050 non-null  float64
 8   long                            102050 non-null  float64
 9   country                         101526 non-null  object
 10  country code                    101927 non-null  object
 11  instant_bookable                101953 non-null  object
 12  cancellation_policy             101982 non-null  object
 13  room type                       102058 non-null  object
 14  Construction year               101844 non-null  float64
 15  price                           101811 non-null  object
 16  service fee                     101785 non-null  object
 17  minimum nights                  101658 non-null  float64
 18  number of reviews               101875 non-null  float64
 19  last review                     86226 non-null   datetime64[ns]
 20  reviews per month               86240 non-null   float64
 21  review rate number              101739 non-null  float64
 22  calculated host listings count  101739 non-null  float64
 23  availability 365                101610 non-null  float64
 24  house_rules                     50216 non-null   object
 25  license                         2 non-null       object
dtypes: datetime64[ns](1), float64(9), int64(2), object(14)
memory usage: 21.0+ MB
```

In [103…
```
print("Total number of null values each column have:\n",df.isna().sum())
```

```
Total number of null values each column have:
 id                                     0
NAME                                  250
host id                                 0
host_identity_verified                289
host name                             404
neighbourhood group                    29
neighbourhood                          16
lat                                     8
long                                    8
country                               532
country code                          131
instant_bookable                      105
cancellation_policy                    76
room type                               0
Construction year                     214
price                                 247
service fee                           273
minimum nights                        400
number of reviews                     183
last review                         15832
reviews per month                   15818
review rate number                    319
calculated host listings count        319
availability 365                      448
house_rules                         51842
license                            102056
dtype: int64
```

In [104…

```
# licencse and house rule are not in our feature selction we can neglect it , means
#can directly talk to the coustomer about the rules otherwise .
#licence is also not as much important as it can be cocluded from the dataset as ou
# so both of the features are not important as much so i neglected these features:-
```

In [104…

```
df.drop(columns =['house_rules','license'], inplace = True)
```

In [104…

```
# also in 1lakh dataset the null values of 100 to 1000 can not affect that much it
```

In [104…

```
#Name column have name of the house or by which owner wants to rent the house and H
```

In [104…

```
df.dropna(subset = ['NAME', 'host name'], inplace = True)
```

In [105…

```
#now the most important feature is the review feature in which we cant use deopna c
#than 10k,so we need to fill this value stastiscally or use the previous value or n
```

In [105…

```
df.fillna({'review per month':df['reviews per month'].mean()} , inplace = True)
```

In [105…

```
#last review consist of the ddate at which review was done , but more than 10K colu
#date instead of NaT
```

In [105…

```
df.fillna({'last review' : df['last review'].min()}, inplace = True)
```

In [105…
```
# here we can see that construction year is having 200 nan ,
# i have decide to replace the nan value with mode of construction year , logically
# the year which has maximum construction of house
```

In [106…
```
df.fillna({'Construction year': df['Construction year'].mode()[0]}, inplace = True)
```

In [106…
```
#mode can be multiple thats why only .mode() will return the series i have choosed
```

In [106…
```
#removing dollar sign and converting it into floating point of price feature
```

In [106…
```
df['price'] = df['price'].astype(str).replace(r'[$,]', '', regex=True).astype(float
df['service fee'] = df['service fee'].astype(str).replace(r'[$,]', '', regex=True).
```

In [106…
```
#now fillig the NAN values of price and service fee with the mean
```

In [107…
```
df.fillna({'price' : df['price'].mean()},inplace= True)
```

In [107…
```
df.fillna({'service fee' : df['service fee'].mean()},inplace= True)
```

In [107…
```
#after cleaning
```

In [107…
```
print('After dropping unuseful columns and NaN \n',df.isnull().sum())
```

```
After dropping unuseful columns and NaN
 id                                    0
NAME                                  0
host id                               0
host_identity_verified              276
host name                             0
neighbourhood group                  26
neighbourhood                        16
lat                                   8
long                                  8
country                             526
country code                        122
instant_bookable                     96
cancellation_policy                  70
room type                             0
Construction year                     0
price                                 0
service fee                           0
minimum nights                      394
number of reviews                   182
last review                           0
reviews per month                 15703
review rate number                  307
calculated host listings count      318
availability 365                    420
dtype: int64
```

In [107…
```
df.head()
```

Out[107…

	id	NAME	host id	host_identity_verified	host name	neighbourhood group
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Brooklyn
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Manhattan
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Elise	Manhattan
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Manhattan
5	1004098	Large Cozy 1 BR Apartment In Midtown East	45498551794	verified	Michelle	Manhattan

5 rows × 24 columns

In [108…

```
# now data cleanig , validation is done next is data visulisation
```

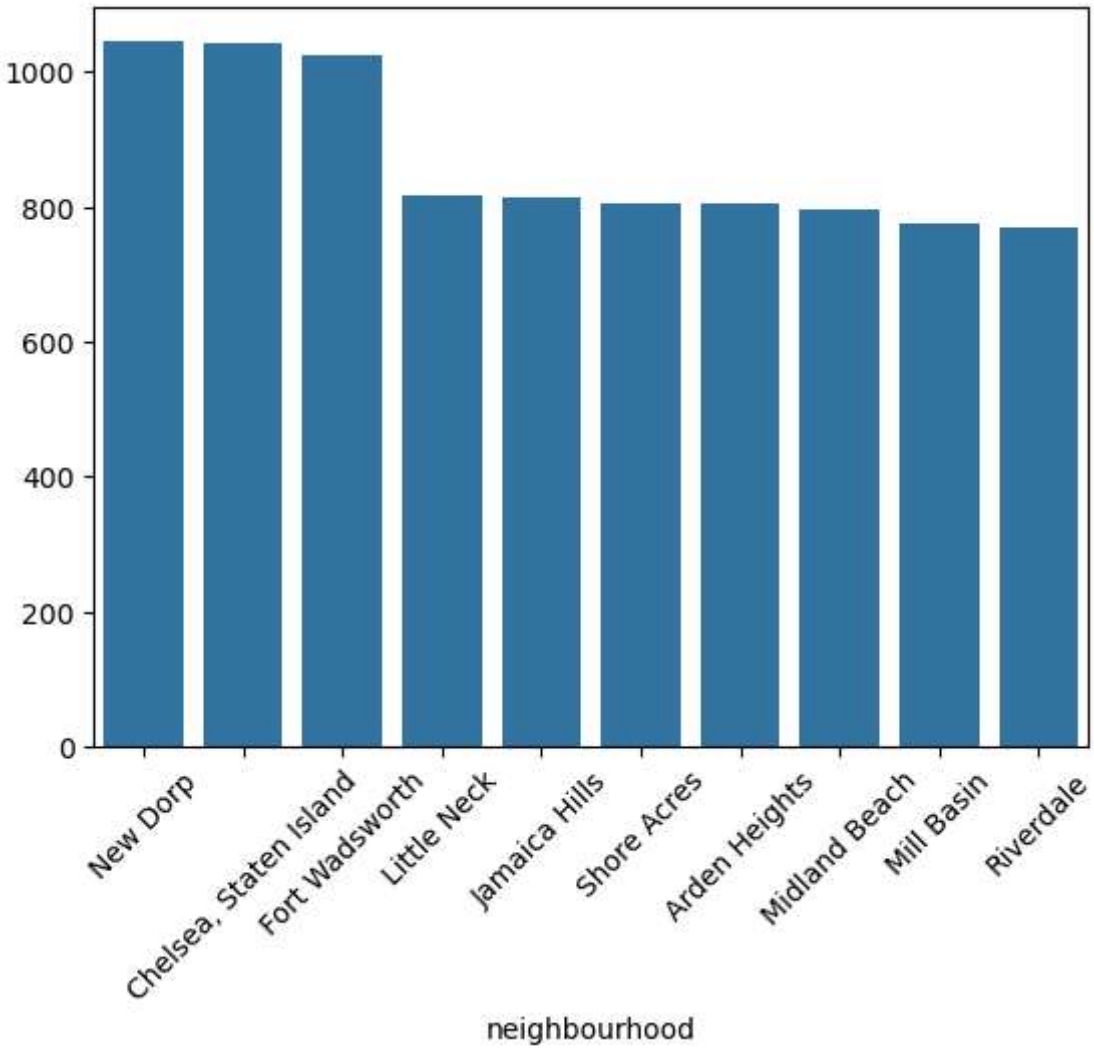
Top 10 Neighbourhood by Average price

In [108…

```
avg_price=df.groupby('neighbourhood')['price'].mean().sort_values(ascending = False
sns.barplot(x=avg_price.index, y =avg_price.values)
plt.xticks(rotation=45)
```

Out[108…

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'New Dorp'),
  Text(1, 0, 'Chelsea, Staten Island'),
  Text(2, 0, 'Fort Wadsworth'),
  Text(3, 0, 'Little Neck'),
  Text(4, 0, 'Jamaica Hills'),
  Text(5, 0, 'Shore Acres'),
  Text(6, 0, 'Arden Heights'),
  Text(7, 0, 'Midland Beach'),
  Text(8, 0, 'Mill Basin'),
  Text(9, 0, 'Riverdale')])
```



Distribution of listing price

In [108…

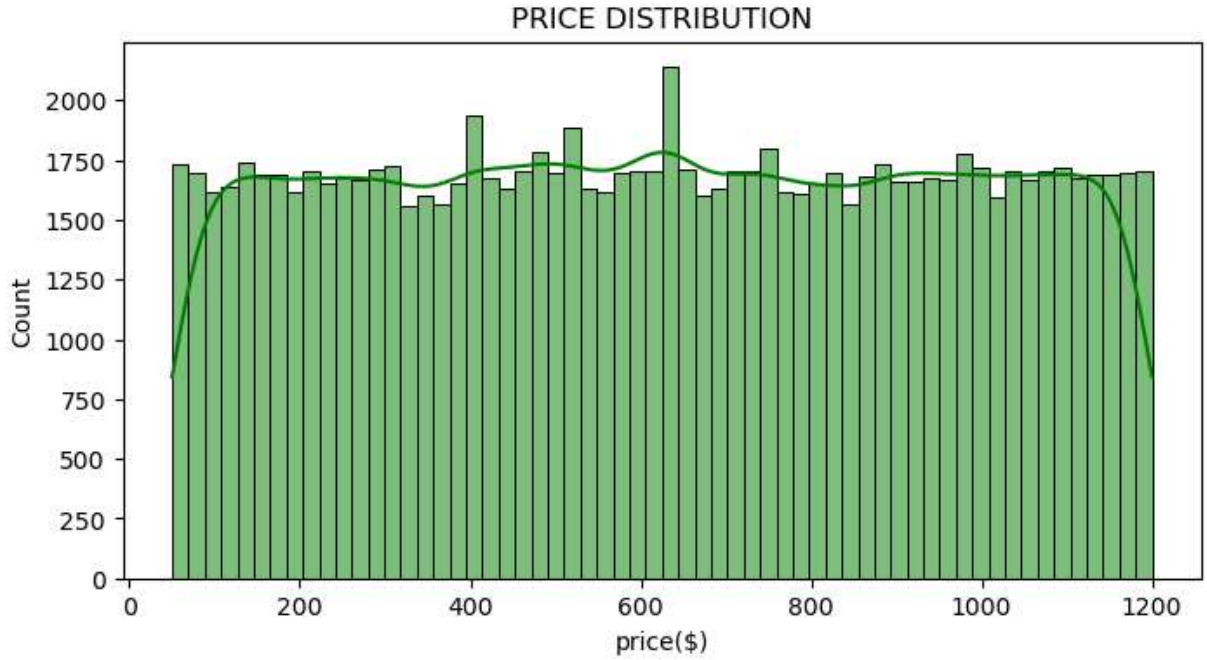
```
df['price'].describe()
```

Out[108…

```
count    101410.000000
mean        625.381008
std         331.218114
min          50.000000
25%         341.000000
50%         625.381008
75%         912.000000
max        1200.000000
Name: price, dtype: float64
```

In [108…

```
plt.figure(figsize=(8,4))
sns.histplot(df['price'] , bins = 60 , kde = True, color = 'green')
plt.xlabel('price($)')
plt.title('PRICE DISTRIBUTION')
plt.show()
```



In [108…

```
#------>> this confirming that the price of the houses have wide range
```

Distribution of Room Type

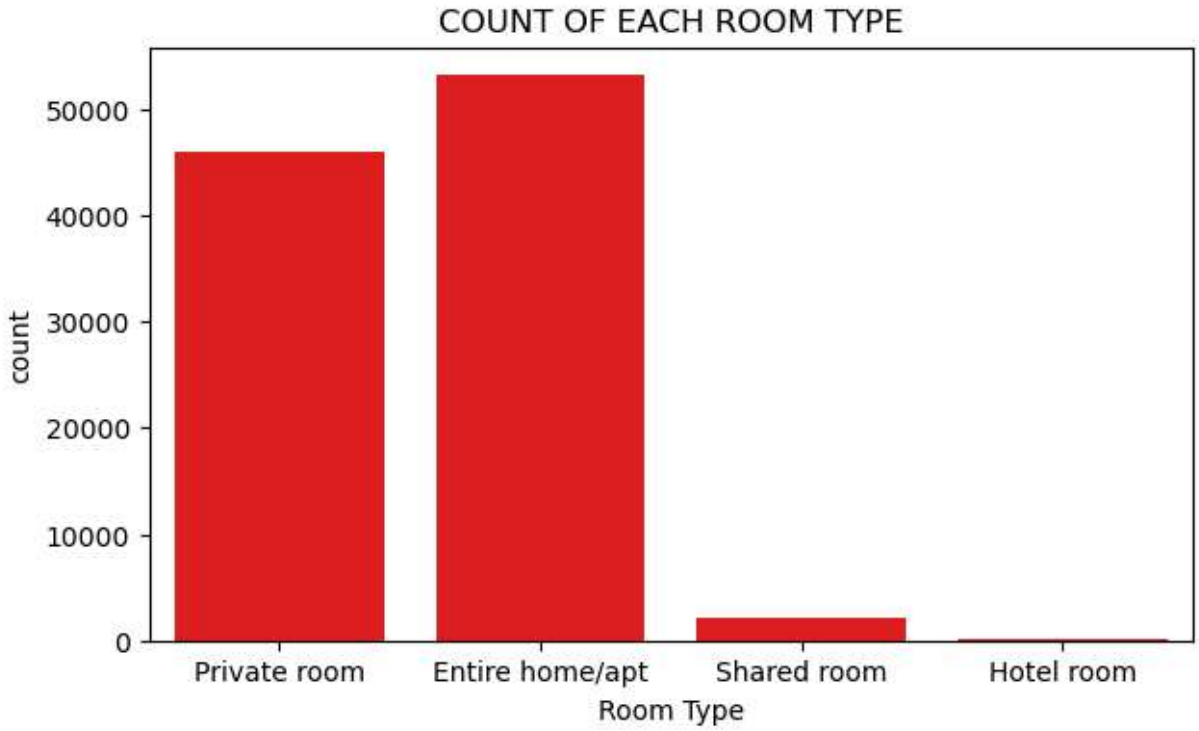
In [109…

```
print('Types of Room AIRBNB have\n',df['room type'])
```

```
Types of Room AIRBNB have
 0            Private room
1         Entire home/apt
2            Private room
4         Entire home/apt
5         Entire home/apt
               ...
102053       Private room
102054       Private room
102055    Entire home/apt
102056       Private room
102057    Entire home/apt
Name: room type, Length: 101410, dtype: object
```

In [109…

```
plt.figure(figsize=(7,4))
sns.countplot(x= df['room type'], color='red')
plt.title('COUNT OF EACH ROOM TYPE')
plt.xlabel('Room Type')
plt.show()
```



In [109…

```
#-----> this conclued the amount of room avalable as well as popularity of room typ
```

Distribution of listing (houses) in Areas

In [110…

```
df['neighbourhood group']
```

Out[110…

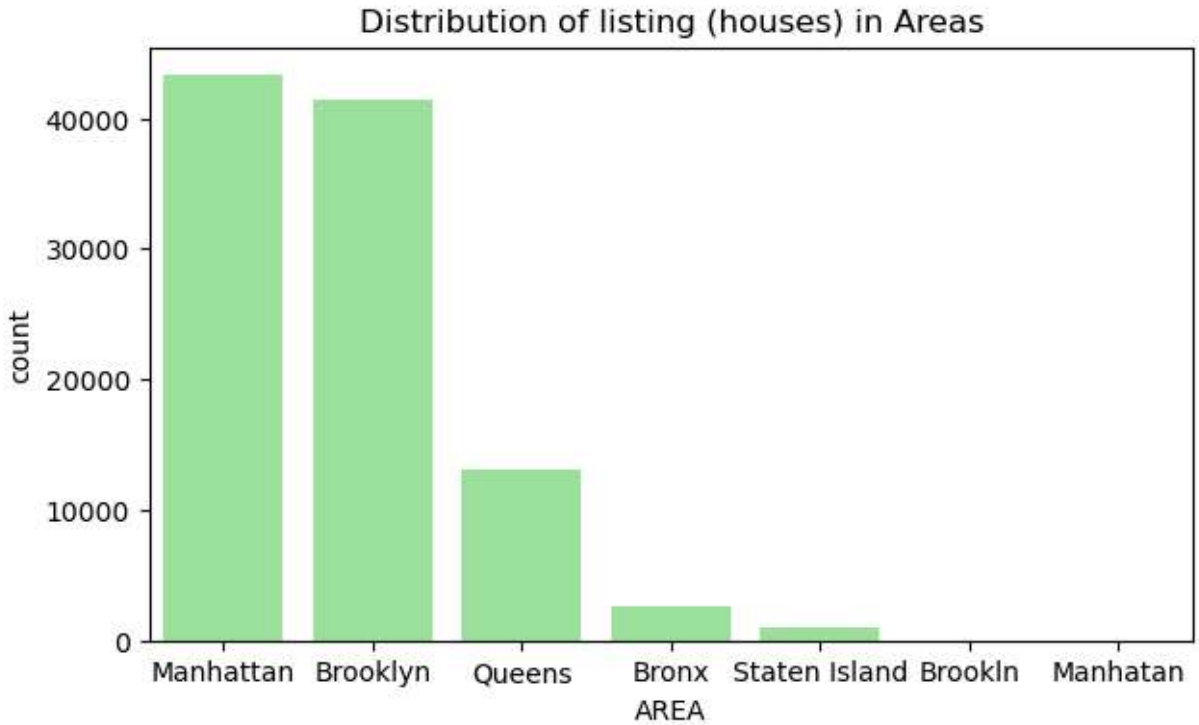
```
0          Brooklyn
1         Manhattan
2         Manhattan
4         Manhattan
5         Manhattan
             ...
102053     Brooklyn
102054     Brooklyn
102055     Brooklyn
102056    Manhattan
102057    Manhattan
Name: neighbourhood group, Length: 101410, dtype: object
```

In [110…

```
#df['neighbourhood group'].value_counts()
#Counts occurrences of each unique value in the column.Returns a Series with values
#order=Ensures that Seaborn plots the categories from most to least frequent.
```

In [110…

```
plt.figure(figsize=(7,4))
df['neighbourhood group']=df['neighbourhood group'].str.title()
sns.countplot(x= df['neighbourhood group'], color='lightgreen', order= df['neighbou
plt.title('Distribution of listing (houses) in Areas')
plt.xlabel('AREA')
plt.show()
```



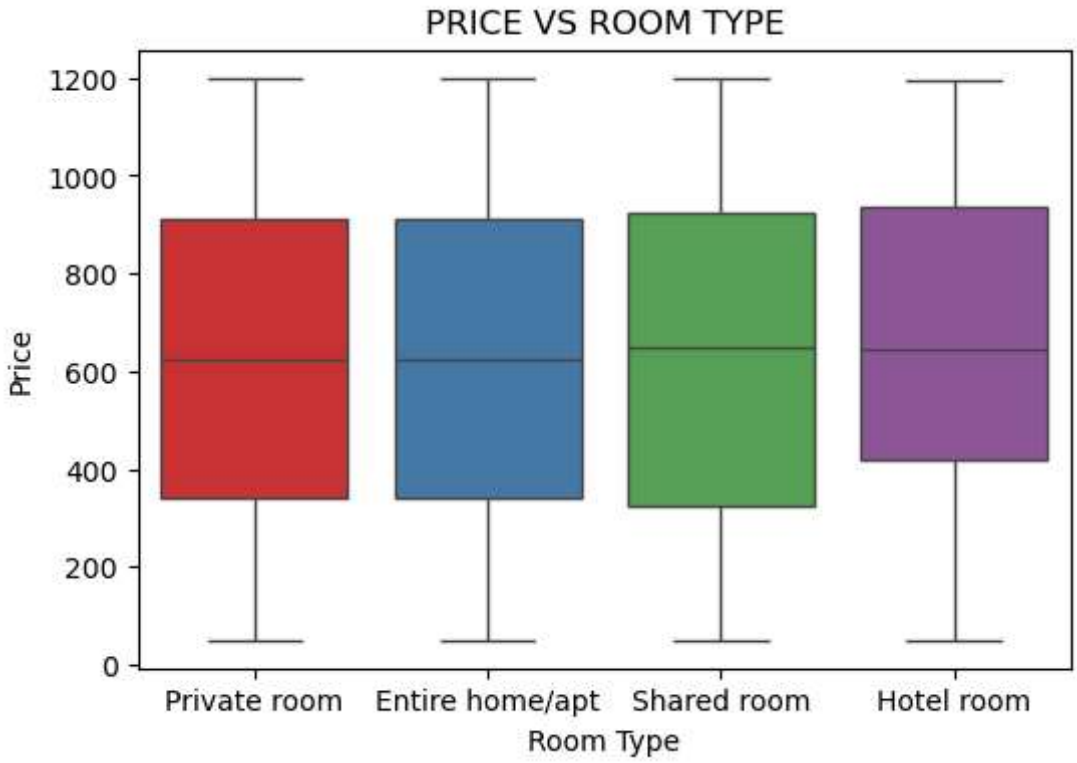
In [110…

```
#-----> this shows the most of the booking are happing in the manhattan and brookly
#prime locations whereas Bronx area and staten island have least
```

Distribution of Price against differnt Room

In [110…

```
plt.figure(figsize=(6,4))
sns.boxplot(x='room type', y='price', hue= 'room type', data= df, palette= 'Set1')
plt.title('PRICE VS ROOM TYPE')
plt.xlabel('Room Type')
plt.ylabel('Price')
plt.show()
```



Review change over time

In [111…

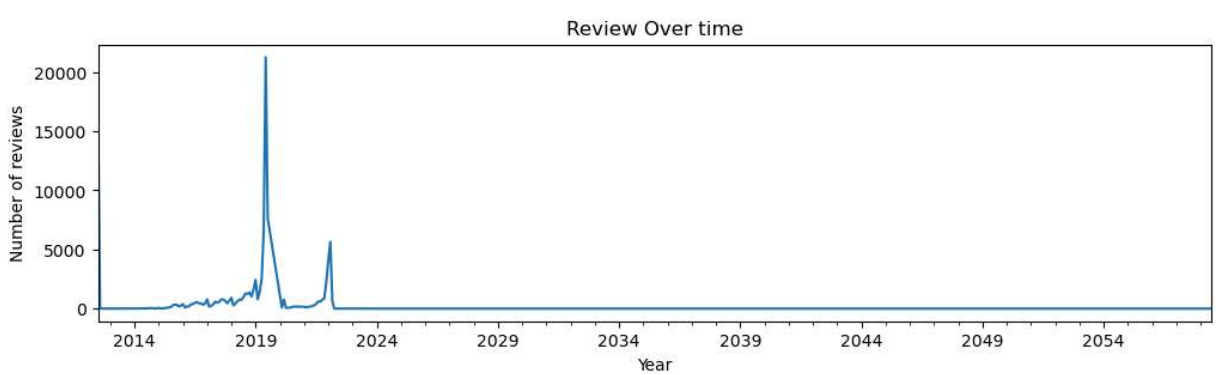
```
# want to see change in review over month time frame
review_over_time= df.groupby(df['last review'].dt.to_period('M')).size()
```

In [111…

```
plt.figure(figsize=(12,3))
review_over_time.plot(kind='line')
plt.xlabel('Year')
plt.ylabel('Number of reviews')
plt.title('Review Over time')
```

Out[111…

```
Text(0.5, 1.0, 'Review Over time')
```



In [111…

```
#----> there was peak in the review in year 2019to2020
```

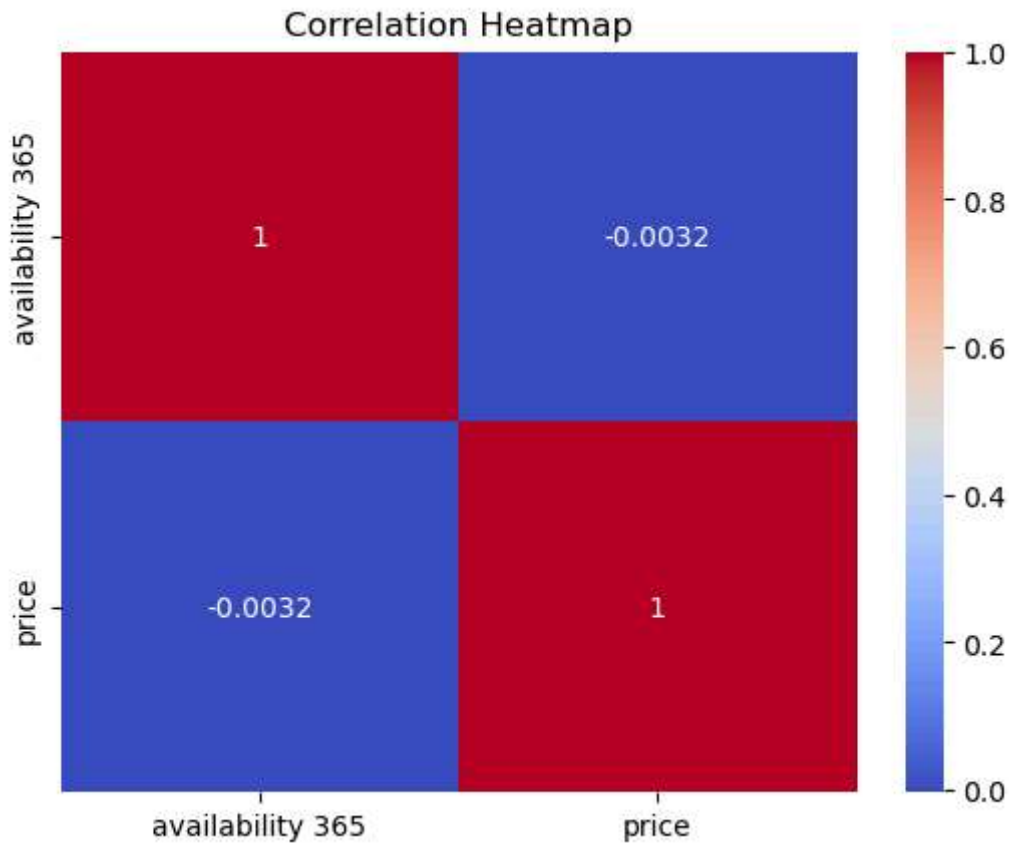
correlation between Avalability , Price and Review

In [111…

```
sns.heatmap(df[['availability 365','price']].corr(), annot=True,cmap='coolwarm')
plt.title('Correlation Heatmap')
```

Out[111…

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



In [112…

```
#-----> here we can se that correlation between different fetures
# postive correlation show postive relation that means directly propotional with fe
# negative  correlation show neg relation that means inversely propotional with fea
# it means when we compare availability 365 vs price they are neg correlated means
```

Relationship between Minimum nights and Number of reviews

In [112…

```
df['minimum nights'].describe()
```

Out[112…

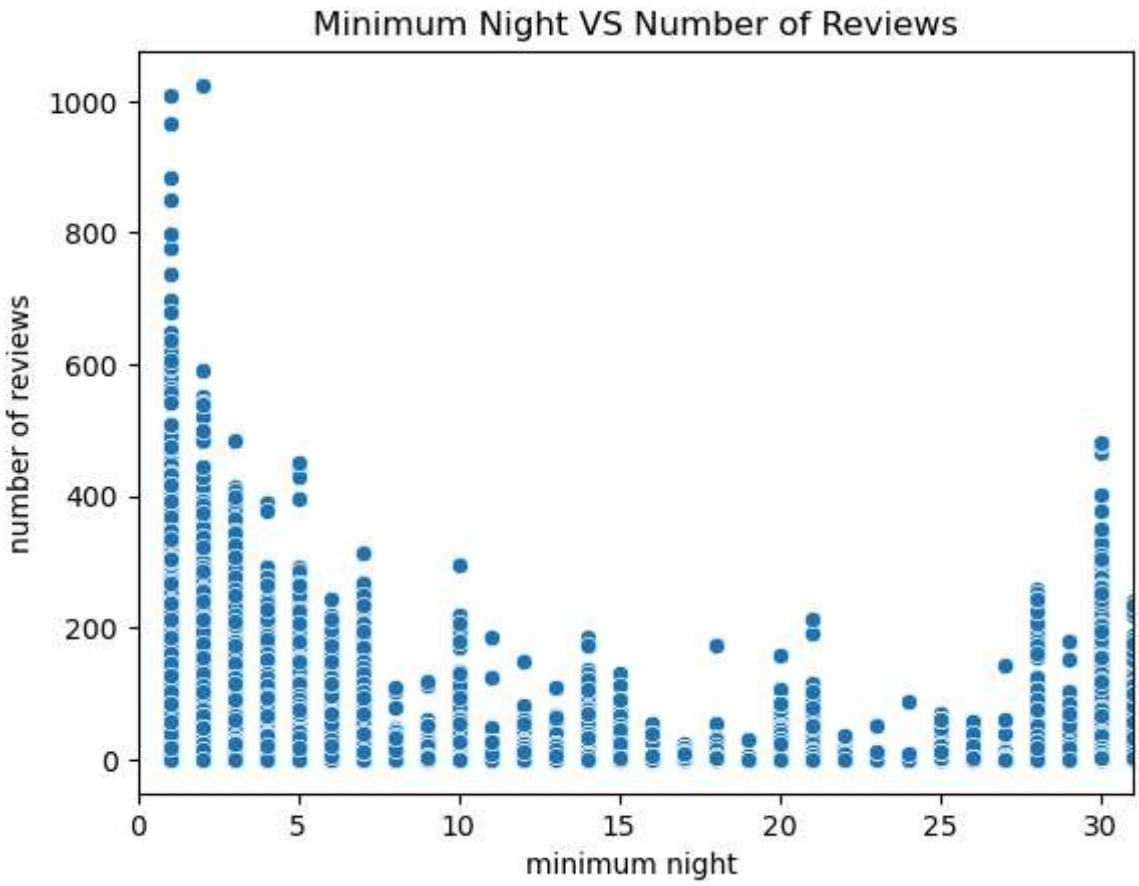
```
count    101016.000000
mean          8.113744
std          30.378014
min       -1223.000000
25%           2.000000
50%           3.000000
75%           5.000000
max        5645.000000
Name: minimum nights, dtype: float64
```

In [112…

```
sns.scatterplot(data =df, x='minimum nights', y= 'number of reviews')
plt.xlim(0,31) # since booking is done mostly 0_30 days max
plt.title('Minimum Night VS Number of Reviews')
plt.xlabel('minimum night ')
```

Out[112…

```
Text(0.5, 0, 'minimum night ')
```



Avg Price by min night stayed

In [112…

```
df_short_stays = df[df['minimum nights'] <= 30].copy()#ccopy insure that it does no
# Define custom bins within 0-30 range
bins = [0, 1, 3, 5, 10, 15, 20, 25, 30]
labels = ['1 day', '2-3', '4-5', '6-10', '11-15', '16-20', '21-25', '26-30']
df_short_stays['min_night_range'] = pd.cut(df_short_stays['minimum nights'], bins=b

# Plot
```

```
plt.figure(figsize=(4, 4))
sns.barplot(data=df_short_stays, x='min_night_range', y='price',color='red')
plt.title("Average Price by Minimum Night Range (1–30 Nights)")
plt.xlabel("Minimum Night Range")
plt.ylabel("Average Price")
plt.xticks(rotation=45)
plt.show()
```

Customer satisfaction by Cancellation Policy vs Review Score

In [113…

```
df['cancellation_policy']
```

Out[113…

```
0           strict
1         moderate
2         flexible
4         moderate
5         flexible
            ...
102053    moderate
102054    flexible
102055    moderate
102056      strict
102057    flexible
Name: cancellation_policy, Length: 101410, dtype: object
```

In [113…

```
plt.figure(figsize=(12, 5))
sns.boxplot(data=df, x='cancellation_policy', y='review rate number',hue='cancellat
```

```
plt.title("Review Score by Cancellation Policy")
plt.xlabel("Cancellation Policy")
plt.ylabel("Review Rating")
plt.legend(loc='upper right')
plt.show()
```

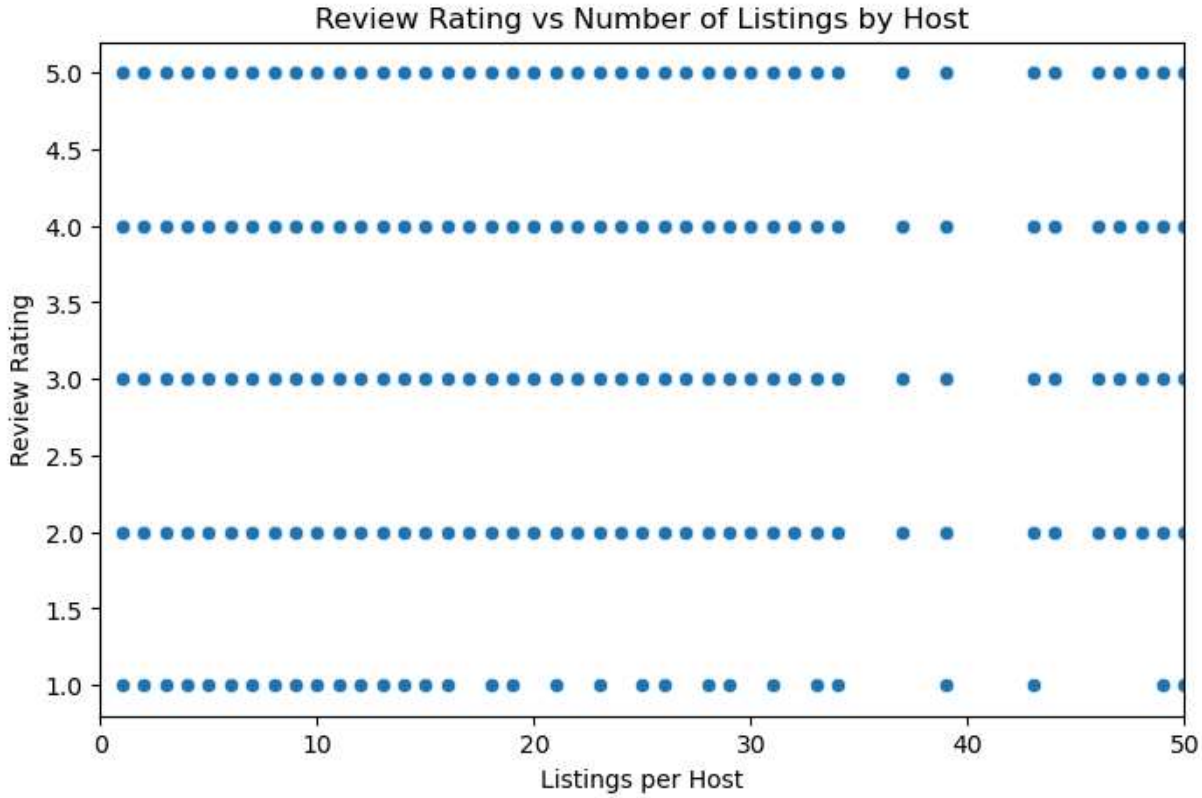
In [113…

```
#----> this shows that the customer satisfactonn is not harmed in this as all the c
```

Listing Count per Host

In [113…

```
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='calculated host listings count', y='review rate number'
plt.xlim(0,50)
plt.title("Review Rating vs Number of Listings by Host")
plt.xlabel("Listings per Host")
plt.ylabel("Review Rating")
plt.show()
```



In [113…

```
#---> this is showing the same person have more listing vs its review it means it i
#if host with many listing maintain the quality or not
```

In [ ]: