Background

Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present more unique, personalized way of experiencing the world. This dataset describes the listing activity and metrics in NYC, NY for 2019. 自2008年以来，房客和房东都使用Airbnb来扩展旅行的可能性，并呈现更独特、个性化的体验世界的方式。该数据集描述了2019年纽约市的上市活动和指标

Content

This data file includes all needed information to find out more about hosts, geographical availability, necessary metrics to make predictions and draw conclusions. 此数据文件包含所有需要的信息，以查找有关房东的更多信息、地理可用性、进行预测和得出结论的必要指标。

Acknowledgements

This public dataset is part of Airbnb, and the original source can be found on this website. 这个公共数据集是Airbnb的一部分，原始来源可以在这个网站上找到。

Inspiration

What can we learn about different hosts and areas?

关于不同的房东和区域，我们能了解到什么?

What can we learn from predictions? (ex: locations, prices, reviews, etc)

我们能从预测中学到什么?(例如:地点、价格、评论等)

Which hosts are the busiest and why?

哪些房东最忙，为什么?

Is there any noticeable difference of traffic among different areas and what could be the reason for it?

不同地区之间的交通是否有明显的差异，原因是什么?

Acquaring and Loading Data

获取和加载数据

In [182]:
```
#importing necessery libraries for future analysis of the dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import seaborn as sns
import datetime
#setting figure size for future visualizations
sns.set(rc={'figure.figsize':(10,8)})
sns.set_style('white')
```

In [183]:
```
airbnb = pd.read_csv('./AB_NYC_2019.csv', header=0)
airbnb.head(3)
```

Out[183]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902

In [184]:
```
#checking amount of rows in given dataset to understand the size we ar
len(airbnb)
```

Out[184]:
```
48895
```

In [185]:
```
#checking type of every column in the dataset
airbnb.dtypes
```

Out[185]:
```
id                                  int64
name                               object
host_id                             int64
host_name                          object
neighbourhood_group                object
neighbourhood                      object
latitude                          float64
longitude                         float64
room_type                          object
price                               int64
minimum_nights                      int64
number_of_reviews                   int64
last_review                        object
reviews_per_month                 float64
calculated_host_listings_count      int64
availability_365                    int64
dtype: object
```

```
After loading the dataset in and from the head of AB_2019_NYC dataset
we can see a number of things.
在从AB_2019_NYC数据集的头部加载数据集之后，我们可以看到许多东西。

These 16 columns provide a very rich amount of information for deep
data exploration we can do on this dataset.
```

这16列提供了非常丰富的信息，我们可以在这个数据集上进行深度数据探索。

We do already see some missing values, which will require cleaning and handling of NaN values.
我们已经看到了一些缺失的值，这将需要清理和处理NaN值。

Later, we may need to continue with mapping certain values to ones and zeros for predictive analytics.
稍后，我们可能需要继续将某些值映射为1和0以进行预测分析。

Understadning, Wrangling and Cleaning Data

理解、整理和清理数据

In [186]:
```
#after looking at the head of the dataset we already were able to noti

#looking to find out first what columns have null values
#using 'sum' function will show us how many nulls are found in each co
#airbnb.isnull()
airbnb.isnull().sum()
```

Out[186]:
```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

In our case, missing data that is observed does not need too much special treatment.
在我们的例子中，观测到的缺失数据不需要太多的特殊处理。

Looking into the nature of our dataset we can state further things: columns "name" and "host_name" are irrelevant and insignificant to our data analysis, columns "last_review" and "review_per_month" need very simple handling.
查看我们数据集的性质，我们可以进一步说明:列“name”和“host_name”与我们的数据分析无关且无关紧要，列“last_review”和“review_per_month”需要非常简单的处理。

To elaborate, "last_review" is date; if there were no reviews for the listing - date simply will not exist.
为了详细说明，“last_review”是日期;如果没有评论，日期根本就不存在。

In our case, this column is irrelevant and insignificant therefore appending those values is not needed.
在我们的例子中，这一列是无关紧要的，因此不需要附加这些值。

For "review_per_month" column we can simply append it with 0.0 for missing values; we can see that in "number_of_review" that column will have a 0, therefore following this logic with 0 total reviews there will be 0.0 rate of reviews per month.
对于“review_per_month”列，我们可以简单地为缺失的值添加0.0;我们可以看到，在“number_of_review”中，该列的值为0，因此，按照这个逻辑，总评论数为0，那么每月的评论数将为0.0。

Therefore, let's proceed with removing columns that are not important and handling of missing data.
因此，让我们继续删除不重要的列并处理丢失的数据。

In [187]:
```
# data=data.dropna(axis=0, how='any')
# data=data.dropna(axis=0, how='all')
# data=data.dropna(axis=1, how='any')
# data=data.dropna(axis=1, how='all')
# data=data.fillna(value=20)
# data
```

In [189]:
```
def to_today(date1):
    date2= datetime.datetime.strptime('2019-12-31',"%Y-%m-%d")
    if date1<date2:
        num=(date2-date1).days
        return num
    else:
        return 100000
airbnb['last_review'] = pd.to_datetime(airbnb['last_review'])
airbnb['last_review_to_today'] = airbnb['last_review'].apply(lambda x:
#dropping columns that are not significant or could be unethical to us
airbnb.drop(['id','host_name','last_review'], axis=1, inplace=True)
#examing the changes
airbnb.head(3)
```

Out[189]:

	name	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	Clean & quiet apt home by the park	2787	Brooklyn	Kensington	40.64749	-73.97237	Pri rc
1	Skylit Midtown Castle	2845	Manhattan	Midtown	40.75362	-73.98377	Er home,
2	THE VILLAGE OF HARLEM....NEW YORK !	4632	Manhattan	Harlem	40.80902	-73.94190	Pri rc

Please note that we are dropping'host_name' not only because it is insignificant but also for ethical reasons.
请注意，我们删除'host_name'不仅因为它无关紧要，而且出于道德原因。

There should be no reasoning to continue data exploration and model training (which we will be doing later) towards specific individuals based on their names.
不应该有理由根据特定个人的名字继续进行数据探索和模型训练(我们将在后面做)。

Why is that?
为什么呢?

Those names are assigned to actual humans, also they present no security threat or military/governmental interest based on the nature of the dataset, therefore names are unimportant to us.
这些名字是分配给真实的人类的，而且基于数据集的性质，他们不会构成安全威胁或军事/政府利益，因此名字对我们来说不重要。

In [190]:
```
#replacing all NaN values in 'reviews_per_month' with 0
airbnb.fillna({'reviews_per_month':0}, inplace=True)
#examing changes
airbnb.reviews_per_month.isnull().sum()
```

Out[190]:
```
0
```

In [191]:
```
#let's proceed with examing some interesting categorical unique values

#examining the unique values of n_group as this column will appear ver
airbnb.neighbourhood_group.unique()
```

Out[191]:
```
array(['Brooklyn', 'Manhattan', 'Queens', 'Staten Island', 'Bronx'],
      dtype=object)
```

In [192]:
```
#examining the unique values of neighbourhood as this column will appe
len(airbnb.neighbourhood.unique())
```

Out[192]:
```
221
```

In [193]:
```
#examining the unique values of room_type as this column will appear v
airbnb.room_type.unique()
```

Out[193]:
```
array(['Private room', 'Entire home/apt', 'Shared room'], dtype=obje
ct)
```

Understanding unique values and categorical data that we have in our dataset was the last step we had to do.
理解数据集中的唯一值和分类数据是我们必须做的最后一步。

It looks like for those columns' values we will be doing some mapping to prepare the dataset for predictive analysis.
看起来对于这些列的值，我们将做一些映射来为预测分析准备数据集。

Exploring and Visualizing Data

Now that we are ready for an exploration of our data, we can make a rule that we are going to be working from left to right.
现在我们已经准备好探索我们的数据，我们可以制定一个规则，我们将从左到右工作。

The reason some may prefer to do this is due to its set approach - some datasets have a big number of attributes, plus this way we will remember to explore each column individually to make sure we learn as much as we can about our dataset.
有些人可能更喜欢这样做的原因是由于它的集合方法——一些数据集有大量的属性，加上这种方法，我们将记住单独探索每一列，以确保我们尽可能多地了解我们的数据集。

```
In [194]: airbnb
```

Out[194]:

	name	host_id	neighbourhood_group	neighbourhood	latitude	longitude
0	Clean & quiet apt home by the park	2787	Brooklyn	Kensington	40.64749	-73.97237
1	Skylit Midtown Castle	2845	Manhattan	Midtown	40.75362	-73.98377
2	THE VILLAGE OF HARLEM....NEW YORK !	4632	Manhattan	Harlem	40.80902	-73.94190
3	Cozy Entire Floor of Brownstone	4869	Brooklyn	Clinton Hill	40.68514	-73.95976
4	Entire Apt: Spacious Studio/Loft by central park	7192	Manhattan	East Harlem	40.79851	-73.94399
...	...	...	...	...	...	...
48890	Charming one bedroom - newly renovated rowhouse	8232441	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995
48891	Affordable room in Bushwick/East Williamsburg	6570630	Brooklyn	Bushwick	40.70184	-73.93317
48892	Sunny Studio at Historical Neighborhood	23492952	Manhattan	Harlem	40.81475	-73.94867
48893	43rd St. Time Square-cozy single bed	30985759	Manhattan	Hell's Kitchen	40.75751	-73.99112
48894	Trendy duplex in the very heart of Hell's Kitchen	68119814	Manhattan	Hell's Kitchen	40.76404	-73.98933

48895 rows × 14 columns

Category Features 的计数情况

host_id

In [195]:
```
#we will skip first column for now and begin from host_id

#let's see what hosts (IDs) have the most listings on Airbnb platform
top_host=airbnb.host_id.value_counts().head(10)
top_host
```

Out[195]:
```
219517861    327
107434423    232
30283594     121
137358866    103
16098958      96
12243051      96
61391963      91
22541573      87
200380610     65
7503643       52
Name: host_id, dtype: int64
```

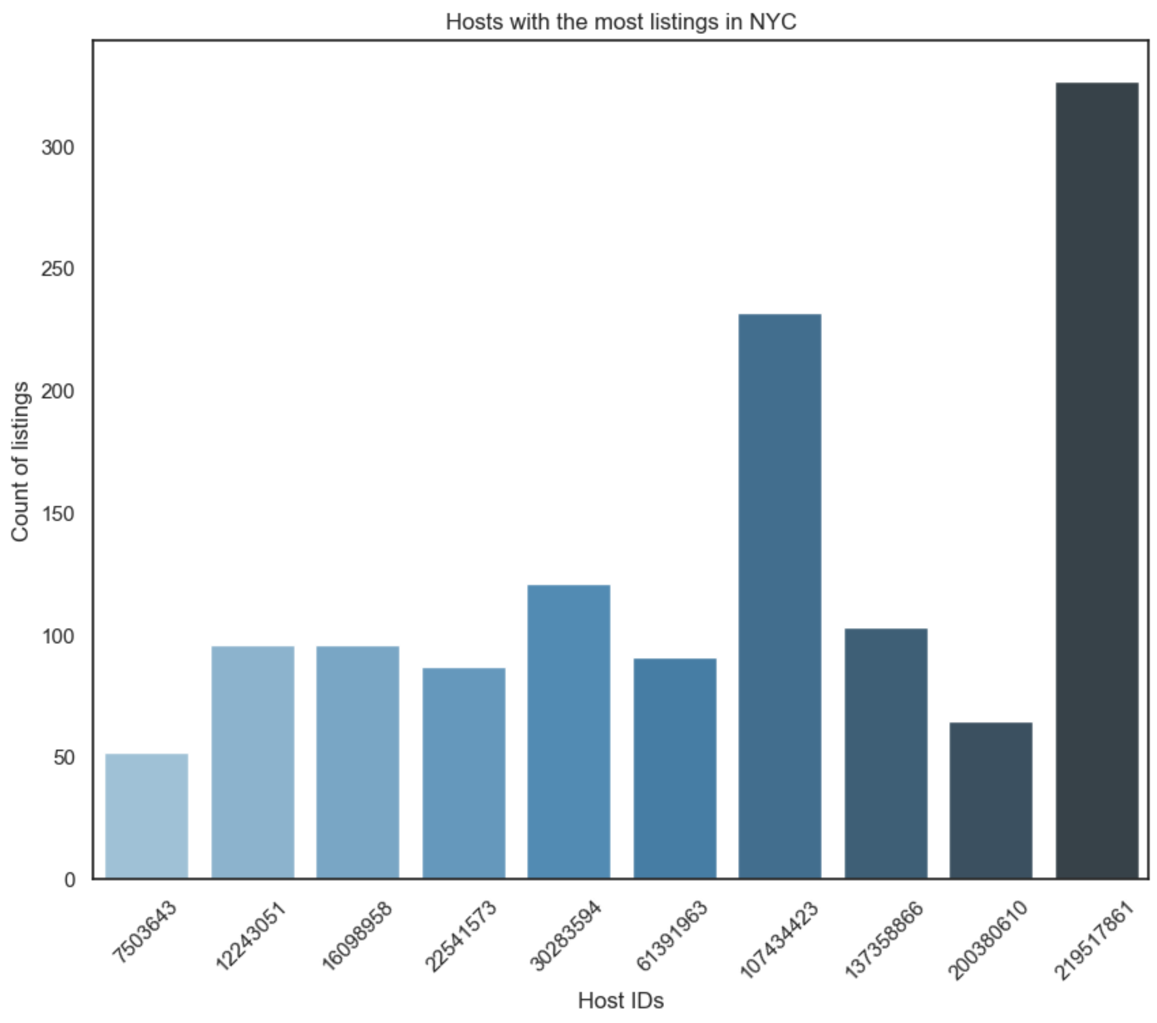
In [196]:
```
top_host_df=pd.DataFrame(top_host)
top_host_df.reset_index(inplace=True)
top_host_df.rename(columns={'index':'Host_ID', 'host_id':'P_Count'}, i
top_host_df
```

Out[196]:

	Host_ID	P_Count
0	219517861	327
1	107434423	232
2	30283594	121
3	137358866	103
4	16098958	96
5	12243051	96
6	61391963	91
7	22541573	87
8	200380610	65
9	7503643	52

```
In [197]: viz_1=sns.barplot(x="Host_ID", y="P_Count", data=top_host_df,
                           palette='Blues_d')
          viz_1.set_title('Hosts with the most listings in NYC')
          viz_1.set_ylabel('Count of listings')
          viz_1.set_xlabel('Host IDs')
          viz_1.set_xticklabels(viz_1.get_xticklabels(), rotation=45)
```

```
Out[197]: [Text(0, 0, '7503643'),
           Text(1, 0, '12243051'),
           Text(2, 0, '16098958'),
           Text(3, 0, '22541573'),
           Text(4, 0, '30283594'),
           Text(5, 0, '61391963'),
           Text(6, 0, '107434423'),
           Text(7, 0, '137358866'),
           Text(8, 0, '200380610'),
           Text(9, 0, '219517861')]
```



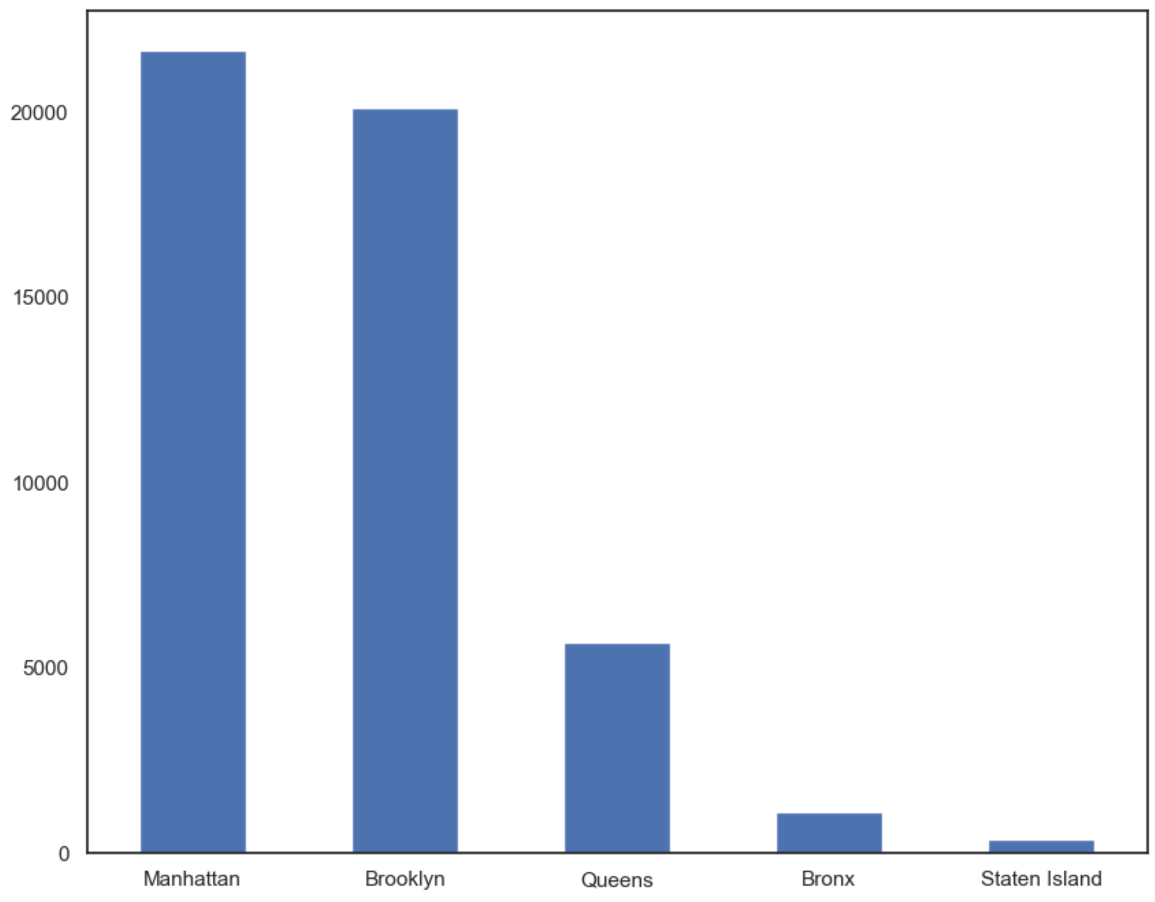
Interesting, we can see that there is a good distribution between top 10 hosts with the most listings.
有趣的是，我们可以看到排名前10的房东之间有一个很好的分布。

First host has more than 300+ listings.
第一个房东有300多个房源。

neighbourhood_group

```
In [198]: viz_nbhg = airbnb.neighbourhood_group.value_counts().plot(kind = 'bar'
          viz_nbhg.set_xticklabels(viz_nbhg.get_xticklabels(), rotation=0)
```

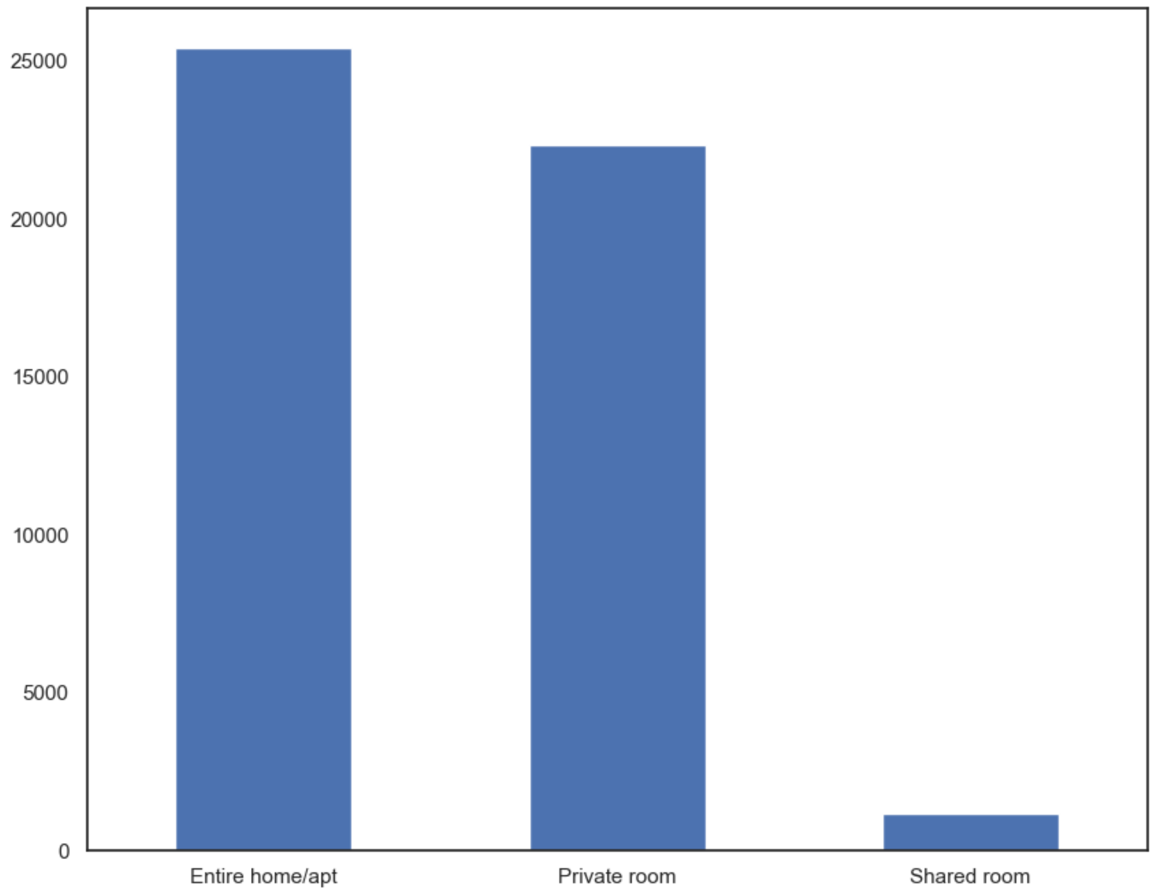
```
Out[198]: [Text(0, 0, 'Manhattan'),
           Text(1, 0, 'Brooklyn'),
           Text(2, 0, 'Queens'),
           Text(3, 0, 'Bronx'),
           Text(4, 0, 'Staten Island')]
```



room_type

```
In [199]: viz_rm = airbnb.room_type.value_counts().plot(kind=  'bar')
          viz_rm.set_xticklabels(viz_rm.get_xticklabels(), rotation = 0)
```

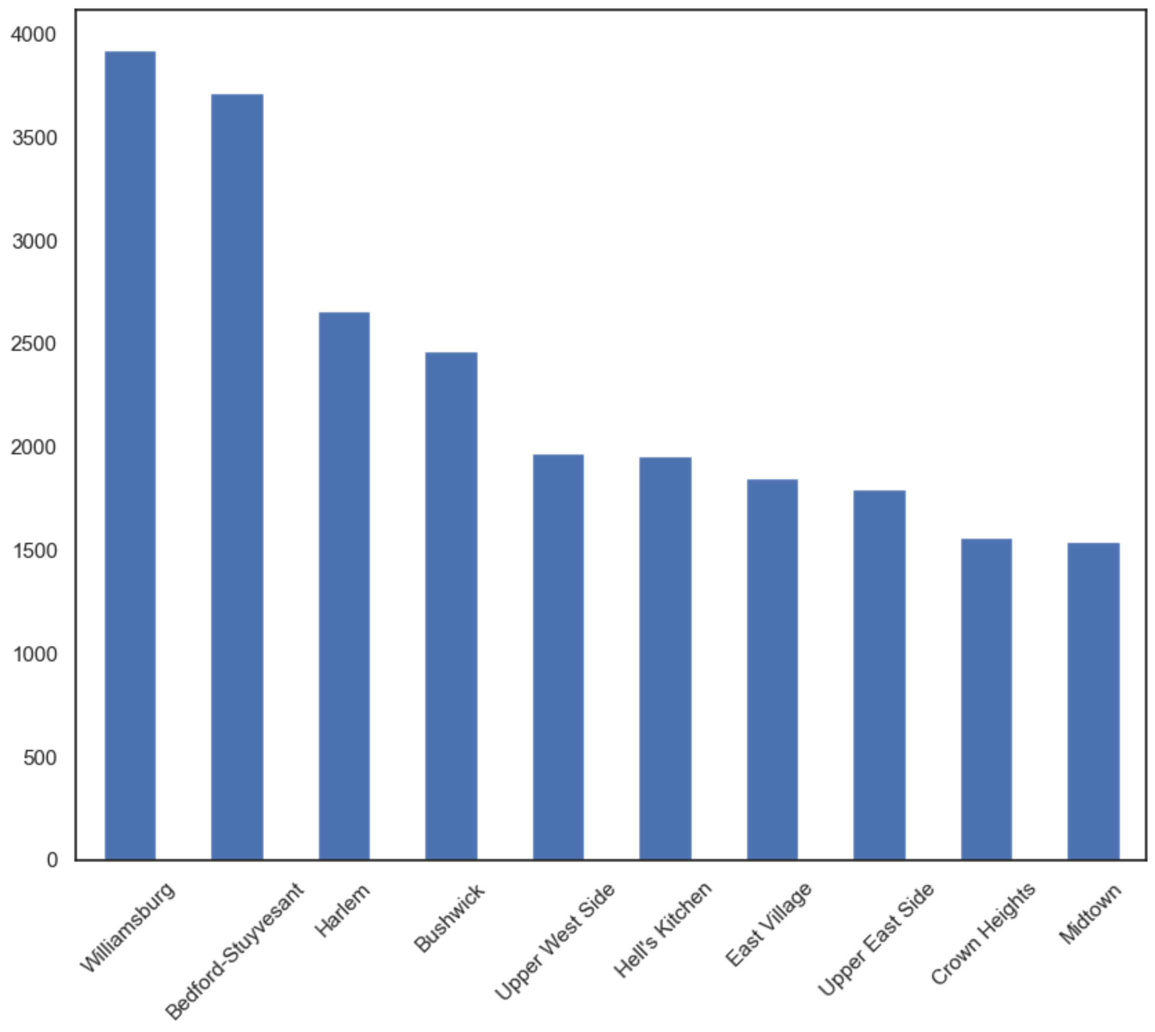
```
Out[199]: [Text(0, 0, 'Entire home/apt'),
           Text(1, 0, 'Private room'),
           Text(2, 0, 'Shared room')]
```



neighbourhood

```
In [200]: viz_nbh = airbnb.neighbourhood.value_counts().head(10).plot(kind= 'bar
          viz_nbh.set_xticklabels(viz_nbh.get_xticklabels(), rotation=45)
```

```
Out[200]: [Text(0, 0, 'Williamsburg'),
           Text(1, 0, 'Bedford-Stuyvesant'),
           Text(2, 0, 'Harlem'),
           Text(3, 0, 'Bushwick'),
           Text(4, 0, 'Upper West Side'),
           Text(5, 0, "Hell's Kitchen"),
           Text(6, 0, 'East Village'),
           Text(7, 0, 'Upper East Side'),
           Text(8, 0, 'Crown Heights'),
           Text(9, 0, 'Midtown')]
```



```
In [201]: #coming back to our dataset we can confirm our fidnings with already e
          top_host_check=airbnb.calculated_host_listings_count.max()
          top_host_check
```

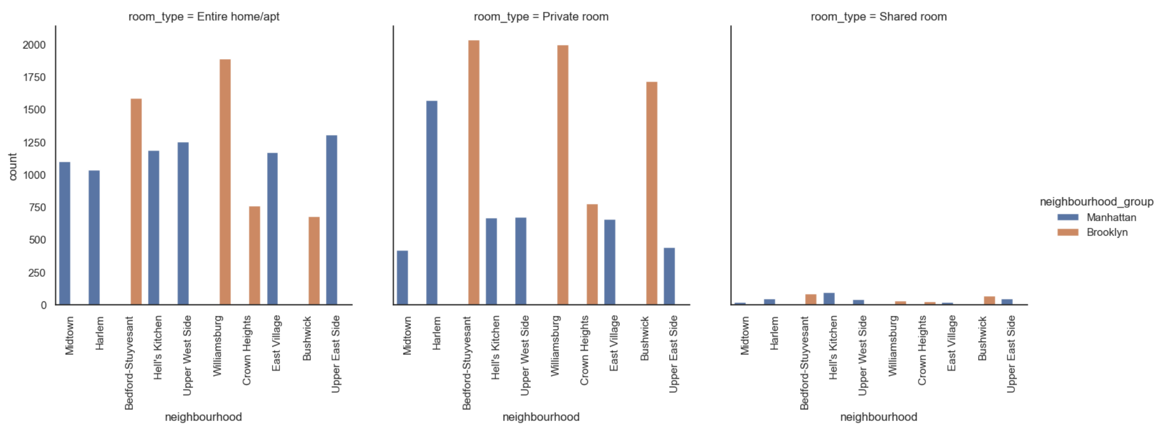
```
Out[201]: 327
```

In [202]:
```
#as we saw earlier from unique values for neighbourhood there are way
#therefore, let's grab just top 10 neighbourhoods that have the most l

#finding out top 10 neighbourhoods
airbnb.neighbourhood.value_counts().head(10)
```

Out[202]:
```
Williamsburg          3920
Bedford-Stuyvesant    3714
Harlem                2658
Bushwick              2465
Upper West Side       1971
Hell's Kitchen        1958
East Village          1853
Upper East Side       1798
Crown Heights         1564
Midtown               1545
Name: neighbourhood, dtype: int64
```

In [211]:
```
#let's now combine this with our boroughs and room type for a rich vis

#grabbing top 10 neighbourhoods for sub-dataframe
sub_7=airbnb.loc[airbnb['neighbourhood'].isin(['Williamsburg','Bedford
                 'Upper West Side','Hell\'s Kitchen','East Village','U
#using catplot to represent multiple interesting attributes together a
viz_3=sns.catplot(x='neighbourhood', hue='neighbourhood_group', col='r
viz_3.set_xticklabels(rotation=90)
```

Out[211]:
```
<seaborn.axisgrid.FacetGrid at 0x188b570a0>
```



Airbnb订单主要集中在 Manhattan 和 Brooklyn，Queen 约为头部二者的一半，Bronx、Staten Island 地区订单数量极少

Entire home/apt 类型房间产生订单最多，Private room 类型房间其次，Shared room 类型房间订单极少

neighbourhood 长尾效应较为明显，第十位约为第一位的一半，共221个中，203个的计数值大于3

Amazing, but let' breakdown on what we can see from this plot.
太棒了，但让我们来分析一下我们从这幅图中看到的东西。

First, we can see that our plot consists of 3 subplots - that is the power of using catplot; with such output, we can easily proceed with comparing distributions among interesting attributes.
首先，我们可以看到我们的图由3个子图组成——这是catplot的强大之处;有了这样的输出，我们就可以很容易地比较感兴趣的属性之间的分布。

```
Y and X axes stay exactly the same for each subplot, Y-axis
represents a count of observations and X-axis observations we want to
count.
Y轴和X轴对每个子图保持完全相同，Y轴表示观测值的计数，X轴表示我们要计数的观测值。

However, there are 2 more important elements: column and hue; those 2
differentiate subplots.
然而，还有两个更重要的元素:列和色调;这两个区分子图。

After we specify the column and determined hue we are able to observe
and compare our Y and X axes among specified column as well as color-
coded.
在我们指定了列和确定的色调之后，我们就可以观察和比较不同颜色的Y轴和X轴

So, what do we learn from this?
那么，我们从中学到什么呢?

The observation that is definitely contrasted the most is that
'Shared room' type Airbnb listing is barely available among 10 most
listing-populated neighborhoods.
对比最明显的观察结果是，“共享房间”类型的Airbnb房源在房源最多的10个社区中几乎找不
到。

Then, we can see that for these 10 neighborhoods only 2 boroughs are
represented: Manhattan and Brooklyn; that was somewhat expected as
Manhattan and Brooklyn are one of the most traveled destinations,
therefore would have the most listing availability.
然后，我们可以看到，在这10个社区中，只有两个区被代表:曼哈顿和布鲁克林;这在一定程度
上是意料之中的，因为曼哈顿和布鲁克林是游客最多的目的地之一，因此会有最多的房源。

We can also observe that Bedford-Stuyvesant and Williamsburg are the
most popular for Manhattan borough, and Harlem for Brooklyn.
我们还可以观察到，贝德福德-史岱文森和威廉斯堡是曼哈顿区最受欢迎的，哈莱姆区是布鲁
克林区最受欢迎的。
```

Number Features

latitude - longitude

price

minimum_nights

number_of_reviews

reviews_per_month

calculated_host_listings_count

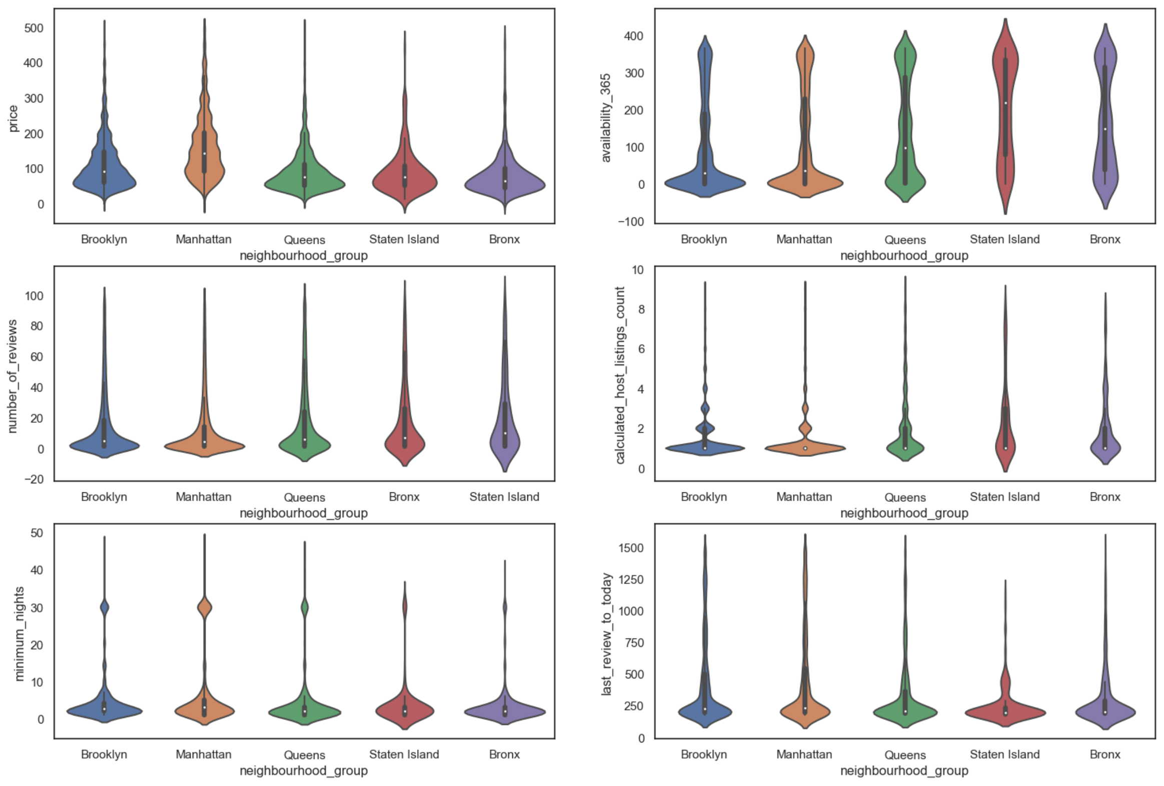
availability_365

last_review_to_today

In [207]:
```
fig=plt.figure(4,figsize=(18,12))
ax1=fig.add_subplot(3,2,1)
ax2=fig.add_subplot(3,2,2)
ax3=fig.add_subplot(3,2,3)
ax4=fig.add_subplot(3,2,4)
ax5=fig.add_subplot(3,2,5)
ax6=fig.add_subplot(3,2,6)

ax_1_per = 100*len(airbnb[airbnb.price<500])/len(airbnb)
ax_2_per = 100*len(airbnb)/len(airbnb)
ax_3_per = 100*len(airbnb[airbnb.number_of_reviews<100])/len(airbnb)
ax_4_per = 100*len(airbnb[airbnb.calculated_host_listings_count<10])/l
ax_5_per = 100*len(airbnb[airbnb.minimum_nights < 50])/len(airbnb)
ax_6_per = 100*len(airbnb[airbnb.last_review_to_today<1500])/len(airbn
print('图中数据集占比分别为: %.2f%%, %.2f%%, %.2f%%, %.2f%%,%.2f%%,%.2f%%
sns.violinplot(x = 'neighbourhood_group', y = 'price',data = airbnb[ai
sns.violinplot(x = 'neighbourhood_group', y = 'availability_365',data
sns.violinplot(x = 'neighbourhood_group', y = 'number_of_reviews',data
sns.violinplot(x = 'neighbourhood_group', y = 'calculated_host_listing
sns.violinplot(x = 'neighbourhood_group', y = 'minimum_nights',data =
sns.violinplot(x = 'neighbourhood_group', y = 'last_review_to_today',d
```

```
图中数据集占比分别为: 97.47%, 100.00%, 93.77%, 93.50%,99.04%,76.54%,
```

Out[207]:
```
<AxesSubplot:xlabel='neighbourhood_group', ylabel='last_review_to_to
day'>
```



纽约各个行政区中，Manhattan 价格没有呈现集中趋势，各个区间分布均匀，其他几个区均有集中在60左右

availability_365 整体呈现两端分布多，中间分布少的特点。在 Brooklyn 和 Manhattan 呈现出类似的分布，集中分布在0天左右，推测这两个区的Airbnb登记房主要为自住房，其他三个区首尾分布近似，自住与出租房占比接近

number_of_reviews 依旧在 Brooklyn 和 Manhattan 呈现出类似的分布，房屋的评论数多集中在0-10之间，结合各个区域房屋数量，可以看到在 Brooklyn 和 Manhattan 由于房屋数量众多，住房和评论比较难出现集中的情况，而越是在房屋少的地区，越容易出现评论数分布集中的情况

calculated_host_listings_count 依旧在 Brooklyn 和 Manhattan 呈现出类似的分布。用户在这两个区都有在订单集中在1、2次的情况，推测这两个区为旅游区，一次性的游客性质租客较多

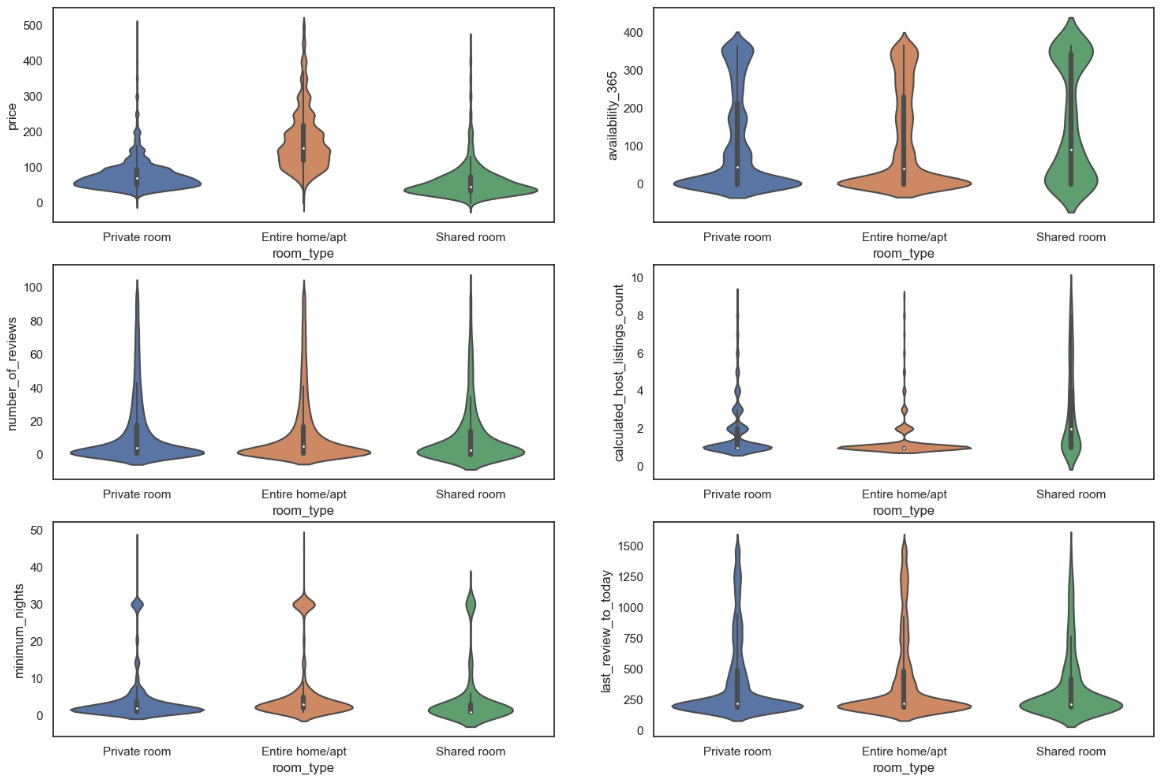
minimum_nights 在各个区的分布状况类似，主要集中分布在0-10天内，但 Manhattan 区约30左右分布明显较多，说明 Manhattan 的部分房屋有议价权，租赁条件较高，属于卖方市场

last_review_to_today 为我们从 last_review 字段派生出来的计算字段，其中空值被我们设为默认值100000。从分布看，最后一次评论时间都集中在约200天前，即 2019/06 左右。Staten Island 分布最为集中且最大约为1200，即2016/09左右，其他地区均早已出现评论，推测 Airbnb 在此地区用户较少，或短期租房需求较少

In [208]:
```python
fig=plt.figure(4,figsize=(18,12))
ax1=fig.add_subplot(3,2,1)
ax2=fig.add_subplot(3,2,2)
ax3=fig.add_subplot(3,2,3)
ax4=fig.add_subplot(3,2,4)
ax5=fig.add_subplot(3,2,5)
ax6=fig.add_subplot(3,2,6)

ax_1_per = 100*len(airbnb[airbnb.price<500])/len(airbnb)
ax_2_per = 100*len(airbnb)/len(airbnb)
ax_3_per = 100*len(airbnb[airbnb.number_of_reviews<100])/len(airbnb)
ax_4_per = 100*len(airbnb[airbnb.calculated_host_listings_count<10])/l
ax_5_per = 100*len(airbnb[airbnb.minimum_nights < 50])/len(airbnb)
ax_6_per = 100*len(airbnb[airbnb.last_review_to_today<1500])/len(airbn
print('图中数据集占比分别为: %.2f%%, %.2f%%, %.2f%%, %.2f%%,%.2f%%,%.2f%%
sns.violinplot(x = 'room_type', y = 'price',data = airbnb[airbnb.price
sns.violinplot(x = 'room_type', y = 'availability_365',data = airbnb,
sns.violinplot(x = 'room_type', y = 'number_of_reviews',data = airbnb|
sns.violinplot(x = 'room_type', y = 'calculated_host_listings_count',d
sns.violinplot(x = 'room_type', y = 'minimum_nights',data = airbnb[air
sns.violinplot(x = 'room_type', y = 'last_review_to_today',data = airb
```

```
图中数据集占比分别为: 97.47%, 100.00%, 93.77%, 93.50%,99.04%,76.54%,
```

Out[208]:
```
<AxesSubplot:xlabel='room_type', ylabel='last_review_to_today'>
```



Private room 与 Shared room 在价格分布上近似，集中在50左右，而按常理讲 Entire Home/apt 面积更大，所以价格更高，整体分布平缓不少，高价房间也较多

availability_365 在 Private room 与 Entire Home/apt 分布近似，可用时间主要集中在0-50天之间，而 Shared room 则呈现两头分布的情况，推测出租 Shared room 的用户属于常驻人口，且对于隐私敏感性较低

number_of_reviews 在三个 room_type 的分布情况类似，均呈现在0-10之间的集中分布

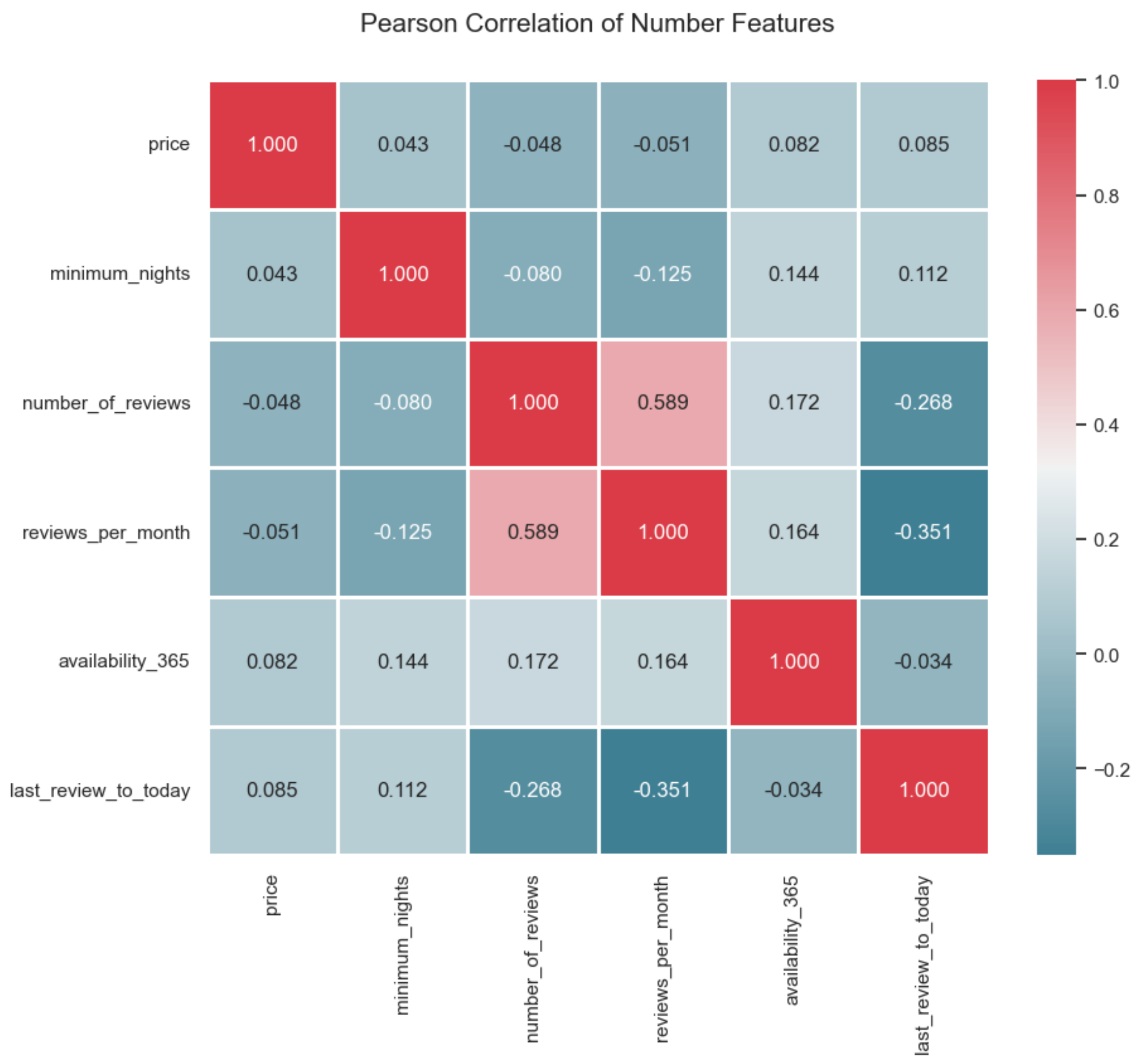
calculated_host_listings_count在 Entire Home/apt 集中趋势明显，显然这种价格较高、面积较大的房屋更适于全家出游，故而租赁次数只有1次的用户最多；在 Private room 的分布平缓很多，但仍然主要集中在0-5次之间；而 Shared room 中已经很平缓，推测可能是住此类房间的客户人数较少，或住此类房间的客户有着订单数目多的特点，需要高频率的在 Airbnb 上租赁，而价格较为低廉的 Shared room 成为了他们的首选

minimum_nights 在三个 room_type 的分布情况类似，主要集中于0-10之间，在30左右有部分集中。其中 Private room 在0-10之间分布最集中，Entire Home/apt 在30日左右集中分布最多，推测可能是由于私人房间出租时有短期随机性（出差），搬离、清洁成本较低，而全套整租成本高，搬离、清洁成本也较高，需要长租来减少边际成本，获得更大利润

last_review_to_today 在 Private room 与 Entire Home/apt 分布近似，Shared room 不集中且较大值少

In [210]:
```
num_field = ['price','minimum_nights','number_of_reviews','reviews_per
f,ax = plt.subplots(figsize=(10, 8))
sns.heatmap(airbnb[num_field].corr(),annot=True,
                    cmap = sns.diverging_palette(220, 10, as_cmap = Tr
                    linewidths=.9, fmt= '.3f',ax = ax)
plt.title('Pearson Correlation of Number Features', y=1.05, size=15)
```

Out[210]:
```
Text(0.5, 1.05, 'Pearson Correlation of Number Features')
```

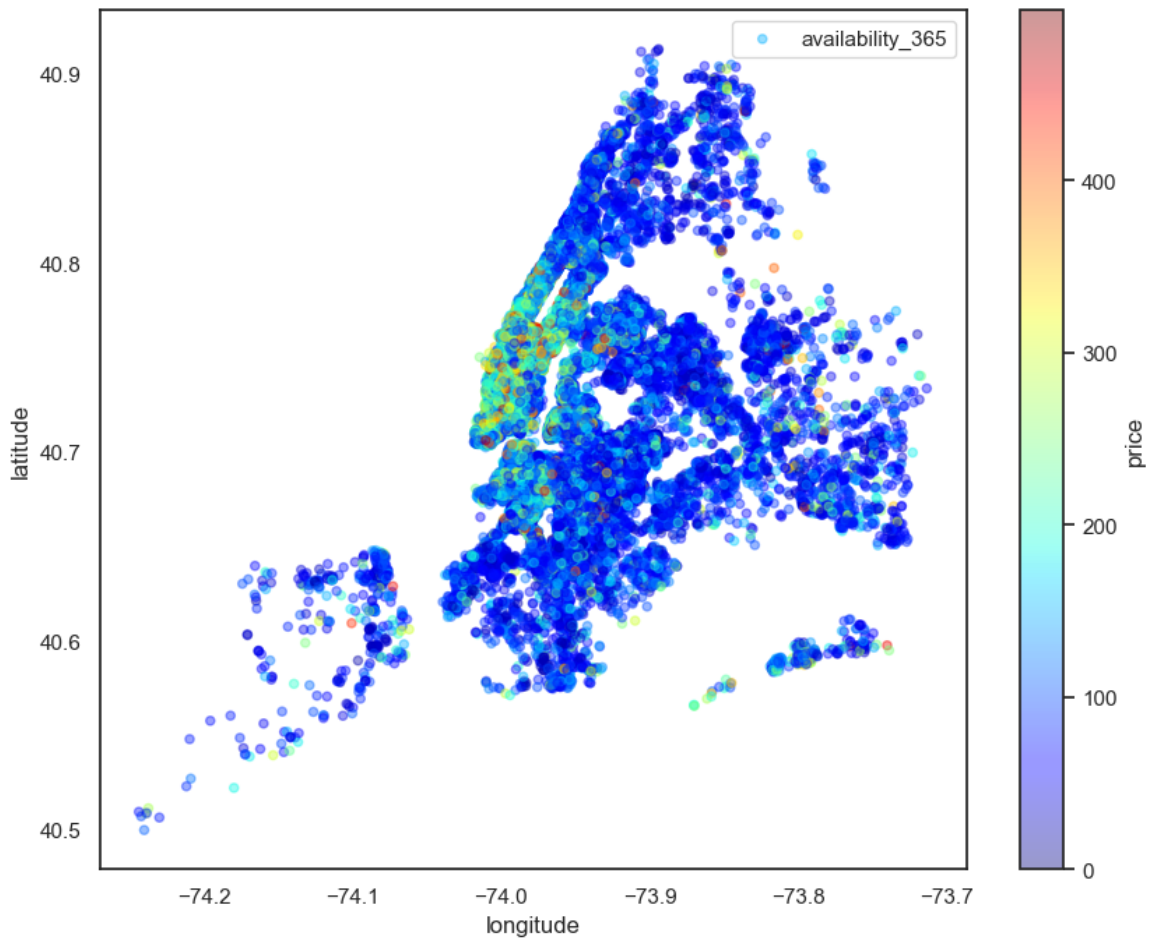


从热力图可以看出，没有字段有强相关，这对于机器学习来说是一个好消息

目前相关性最强的两组字段为：number_of_review 和 reviews_per_month 正相关，last_review_to_today 和 reviews_per_month 负相关。reviews_per_month 由 number_of_review 计算生成，相关性差异主要由于月份影响；last_review_to_today 和 reviews_per_month 可以理解为评论距今的时间越长，月均评论数越少。有可能是我们对 last_review_to_today 缺失默认值的处理导致这样的情况出现

In [87]:
```
#let's what we can do with our given longtitude and latitude columns

#let's see how scatterplot will come out
viz_4=sub_6.plot(kind='scatter', x='longitude', y='latitude', label='a
                        cmap=plt.get_cmap('jet'), colorbar=True, alpha=0.4,
viz_4.legend()
```

Out[87]:
```
<matplotlib.legend.Legend at 0x18258deb0>
```

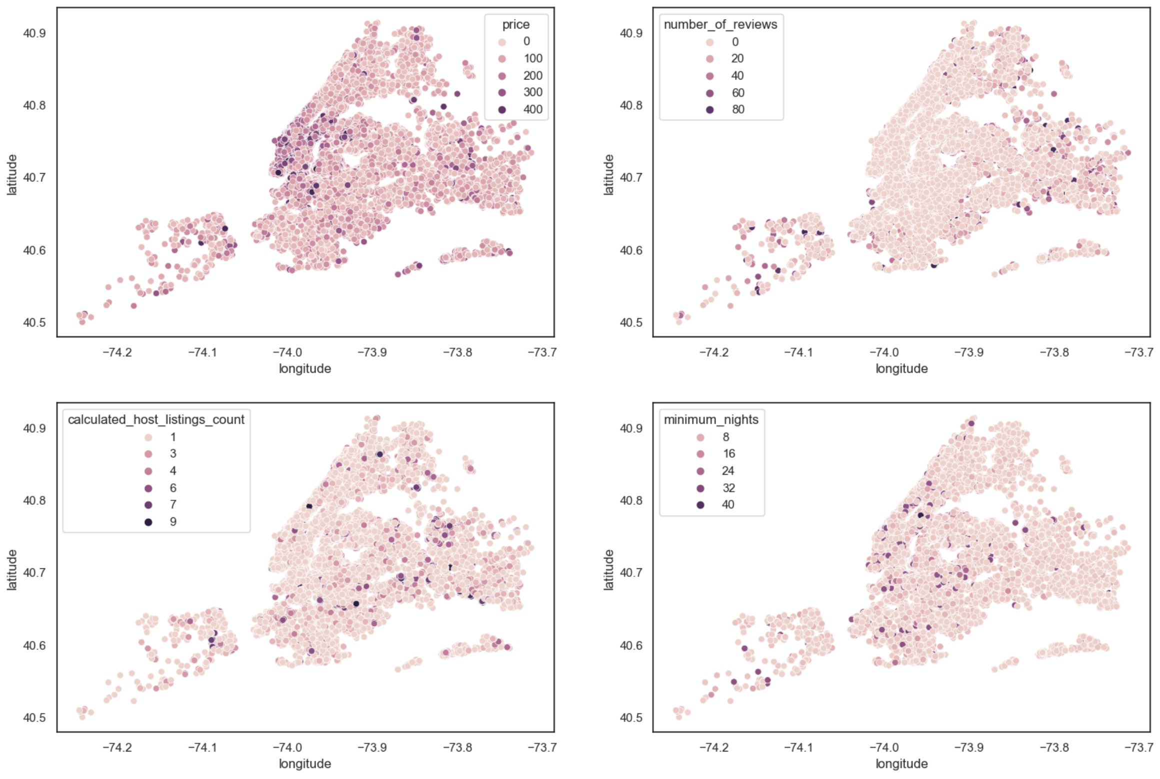


Good, scatterplot worked just fine to output our latitude and longitude points.
很好，散点图可以很好地输出经纬度点。

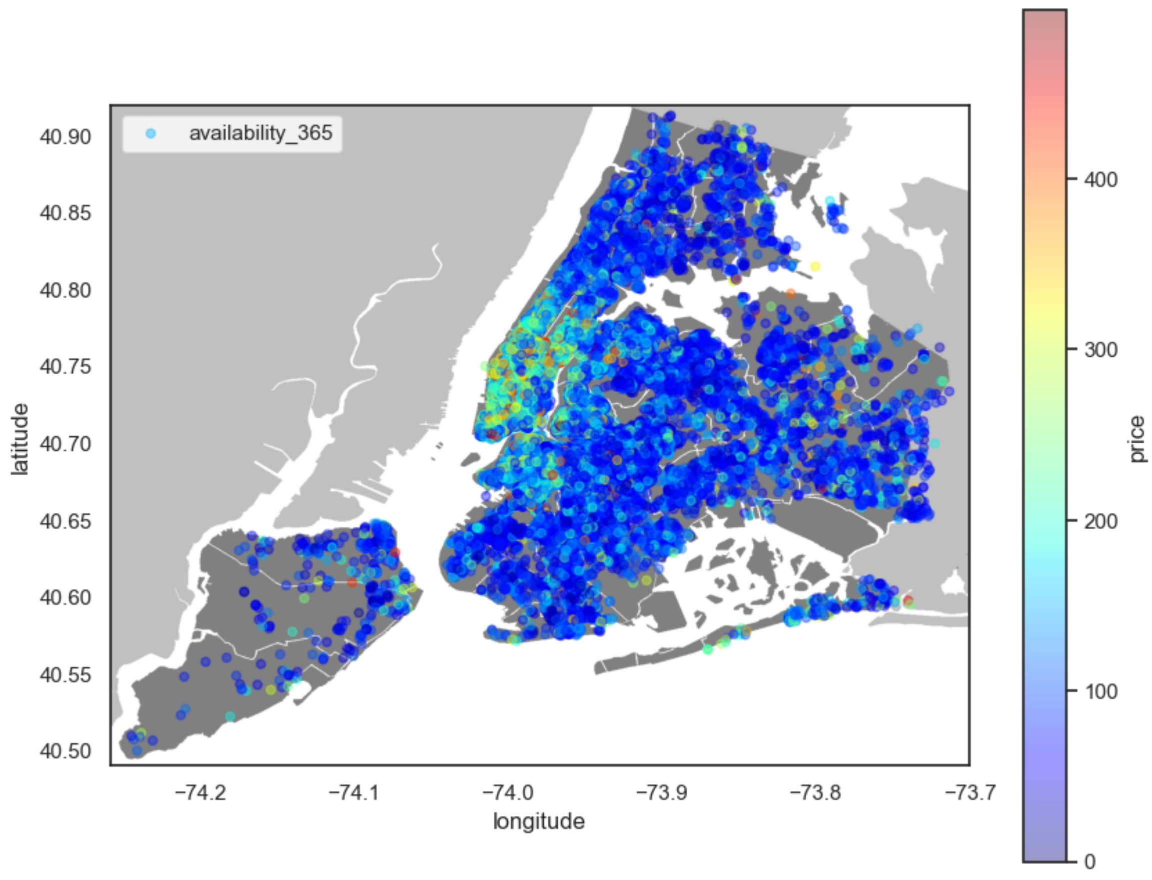
However, it would be nice to have a map bellow for fully immersive heatmap in ourcase - let's see what we can do!
然而，在我们的情况下，如果有一个完全沉浸式的热图，那就太好了——让我们看看我们能做些什么！

In [213]:
```python
fig=plt.figure(4,figsize=(18,12))

ax1=fig.add_subplot(2,2,1)
ax2=fig.add_subplot(2,2,2)
ax3=fig.add_subplot(2,2,3)
ax4=fig.add_subplot(2,2,4)

sns.scatterplot(x= 'longitude', y='latitude', hue = 'price', data = ai
sns.scatterplot(x= 'longitude', y='latitude', hue = 'number_of_reviews
sns.scatterplot(x= 'longitude', y='latitude', hue = 'calculated_host_l
sns.scatterplot(x= 'longitude', y='latitude', hue = 'minimum_nights',
```

Out[213]:
```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



```
In [96]: import urllib.request
         import io
         import matplotlib.pyplot as plt

         # Initializing the figure size
         plt.figure(figsize=(10,8))

         # Loading the PNG NYC image from the web
         url = 'https://upload.wikimedia.org/wikipedia/commons/e/ec/Neighbourhc
         with urllib.request.urlopen(url) as response:
             nyc_img_data = response.read()  # Read data from URL response
             nyc_img = plt.imread(io.BytesIO(nyc_img_data))  # Read image into

         # Scaling the image based on the latitude and longitude max and mins f
         plt.imshow(nyc_img, zorder=0, extent=[-74.258, -73.7, 40.49, 40.92])
         ax = plt.gca()

         # Assuming sub_6 is a DataFrame with the necessary columns
         # Replace 'sub_6' with your actual DataFrame name
         sub_6.plot(kind='scatter', x='longitude', y='latitude', label='availab
                    cmap=plt.get_cmap('jet'), colorbar=True, alpha=0.4, zorder=

         plt.legend()
         plt.show()
```



Fantastic! After scaling our image the best we can, we observe that we end up with a very immersive heatmap.]
太棒了!在尽我们所能地缩放图像后，我们观察到我们最终得到了一个非常逼真的热图

Using latitude and longitude points were able to visualize all NYC listings.
使用纬度和经度点，我们可以可视化纽约市的所有房源

Also, we added a color-coded range for each point on the map based on the price of the listing.
此外，我们还根据清单的价格为地图上的每个点添加了颜色编码的范围。

However, it is important to note that we had to drop some extremely high values as they are treated as outliers for our analysis.
然而，重要的是要注意，我们必须删除一些极高的值，因为它们被视为我们分析的异常值。

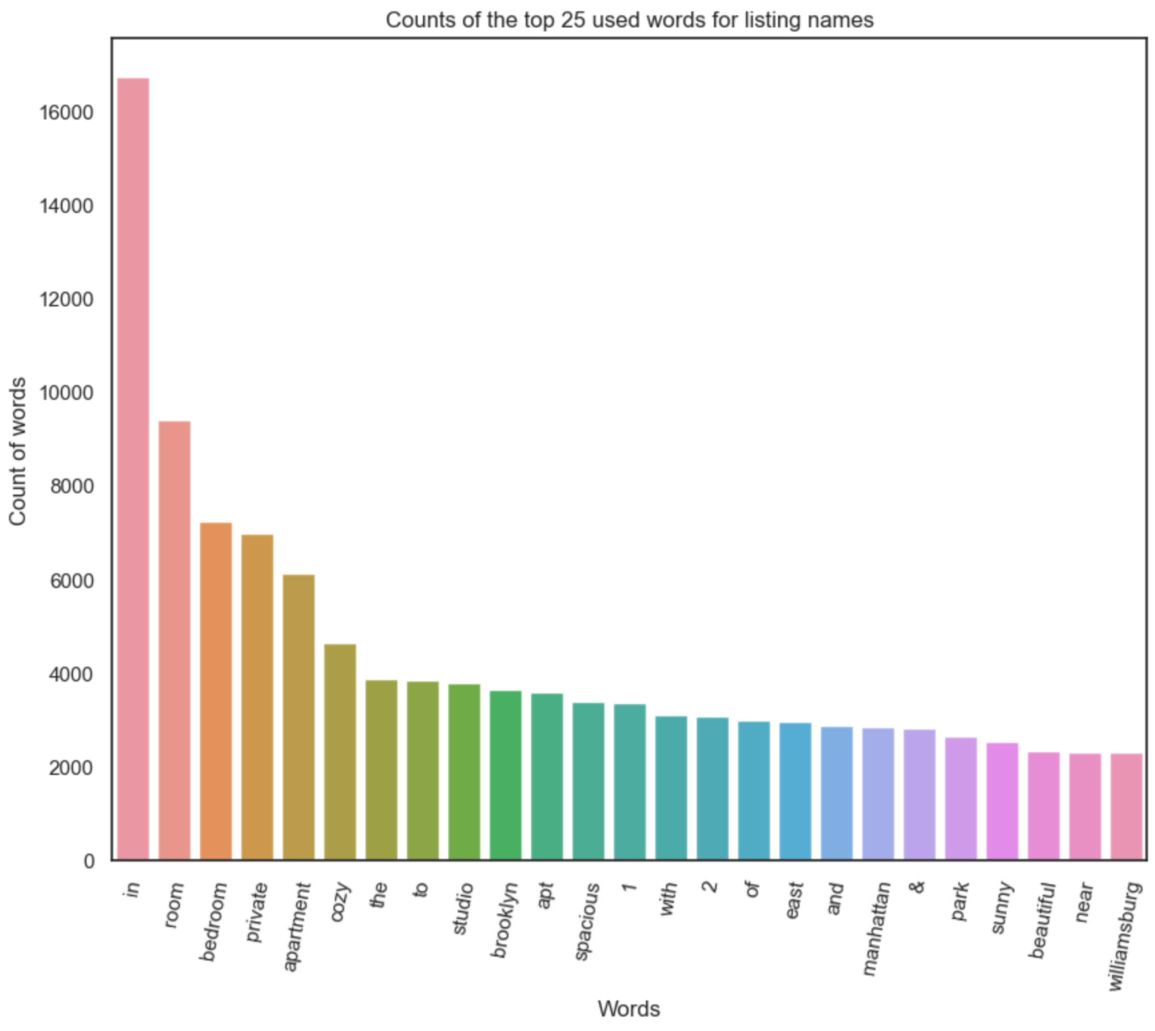
In [89]:
```
#let's comeback now to the 'name' column as it will require litte bit

#initializing empty list where we are going to put our name strings
_names_=[]
#getting name strings from the column and appending it to the list
for name in airbnb.name:
    _names_.append(name)
#setting a function that will split those name strings into separate w
def split_name(name):
    spl=str(name).split()
    return spl
#initializing empty list where we are going to have words counted
_names_for_count_=[]
#getting name string from our list and using split function, later app
for x in _names_:
    for word in split_name(x):
        word=word.lower()
        _names_for_count_.append(word)
```

In [90]:
```
#we are going to use counter
from collections import Counter
#let's see top 25 used words by host to name their listing
_top_25_w=Counter(_names_for_count_).most_common()
_top_25_w=_top_25_w[0:25]
```

In [91]:
```
#now let's put our findings in dataframe for further visualizations
sub_w=pd.DataFrame(_top_25_w)
sub_w.rename(columns={0:'Words', 1:'Count'}, inplace=True)
```

In [92]:
```
#we are going to use barplot for this visualization
viz_5=sns.barplot(x='Words', y='Count', data=sub_w)
viz_5.set_title('Counts of the top 25 used words for listing names')
viz_5.set_ylabel('Count of words')
viz_5.set_xlabel('Words')
viz_5.set_xticklabels(viz_5.get_xticklabels(), rotation=80)
```

Out[92]:
```
[Text(0, 0, 'in'),
 Text(1, 0, 'room'),
 Text(2, 0, 'bedroom'),
 Text(3, 0, 'private'),
 Text(4, 0, 'apartment'),
 Text(5, 0, 'cozy'),
 Text(6, 0, 'the'),
 Text(7, 0, 'to'),
 Text(8, 0, 'studio'),
 Text(9, 0, 'brooklyn'),
 Text(10, 0, 'apt'),
 Text(11, 0, 'spacious'),
 Text(12, 0, '1'),
 Text(13, 0, 'with'),
 Text(14, 0, '2'),
 Text(15, 0, 'of'),
 Text(16, 0, 'east'),
 Text(17, 0, 'and'),
 Text(18, 0, 'manhattan'),
 Text(19, 0, '&'),
 Text(20, 0, 'park'),
 Text(21, 0, 'sunny'),
 Text(22, 0, 'beautiful'),
 Text(23, 0, 'near'),
 Text(24, 0, 'williamsburg')]
```



```
We can observe that finding out and going over top 25 used listings'
name words - we are able to see one clear trend.
我们可以观察到，查找并查看前25个使用列表的名称词-我们能够看到一个明显的趋势。

It shows that hosts are simply describing their listing in a short
form with very specific terms for easier search by a potential
traveler.
它表明，房东只是简单地用非常具体的术语简短地描述他们的房源，以便潜在的旅行者更容易
搜索。

Such wors are 'room', 'bedroom', 'private', 'apartment', 'studio'.
这些词包括room, bedroom, private, apartment, studio。

This shows that there are no catchphrases or 'popular/trending' terms
that are used for names; hosts use very simple terms describing the
space and the area where the listing is.
这表明没有流行语或“流行/趋势”术语用于名称;房东使用非常简单的术语来描述列表所在的空
间和区域。

This technique was somewhat expected as dealing with multilingual
customers can be tricky and you definitely want to describe your
space in a concise and understood form as much as possible.
这种技术在某种程度上是意料之中的，因为处理多语言客户可能会很棘手，而且你肯定希望尽
可能以简洁易懂的形式描述你的空间。
```

In [93]:
```
#last column we need to look at is 'number_of_reviews'

#let's grab 10 most reviewed listings in NYC
top_reviewed_listings=airbnb.nlargest(10,'number_of_reviews')
top_reviewed_listings
```

Out[93]:

	name	host_id	neighbourhood_group	neighbourhood	latitude	long
11759	Room near JFK Queen Bed	47621202	Queens	Jamaica	40.66730	-73.7
2031	Great Bedroom in Manhattan	4734398	Manhattan	Harlem	40.82085	-73.9
2030	Beautiful Bedroom in Manhattan	4734398	Manhattan	Harlem	40.82124	-73.9
2015	Private Bedroom in Manhattan	4734398	Manhattan	Harlem	40.82264	-73.9
13495	Room Near JFK Twin Beds	47621202	Queens	Jamaica	40.66939	-73.7
10623	Steps away from Laguardia airport	37312959	Queens	East Elmhurst	40.77006	-73.8
1879	Manhattan Lux Loft.Like.Love.Lots.Look !	2369681	Manhattan	Lower East Side	40.71921	-73.9
20403	Cozy Room Family Home LGA Airport NO CLEANING FEE	26432133	Queens	East Elmhurst	40.76335	-73.8
4870	Private brownstone studio Brooklyn	12949460	Brooklyn	Park Slope	40.67926	-73.9
471	LG Private Room/Family Friendly	792159	Brooklyn	Bushwick	40.70283	-73.9

In [94]:
```
price_avrg=top_reviewed_listings.price.mean()
print('Average price per night: {}'.format(price_avrg))
```

```
Average price per night: 65.4
```

There is no reason to visualize this as table format would be the most suitable output for better reading.
没有理由将其可视化，因为表格格式将是最适合的输出，以便更好地阅读。

From this table output, we can observe that top 10 most reviewed listings on Airbnb for NYC has price average of $65 with most of the listings under \$50, and 9/10 of them are 'Private room' type; top reviewed listing has 629 reviews.
从这个表的输出中，我们可以观察到，纽约市Airbnb上评论最多的前10个房源的平均价格为65美元，大多数房源在50美元以下，其中9/10是“私人房间”类型;热门评论列表有629条评论。

Conclusion

This Airbnb ('AB_NYC_2019') dataset for the 2019 year appeared to be a very rich dataset with a variety of columns that allowed us to do deep data exploration on each significant column presented.
这个2019年的Airbnb ('AB_NYC_2019')数据集似乎是一个非常丰富的数据集，包含各种列，使我们能够对每个重要的列进行深入的数据探索。

First, we have found hosts that take good advantage of the Airbnb platform and provide the most listings; we found that our top host has 327 listings.
首先，我们找到了充分利用Airbnb平台并提供最多房源的房东;我们发现排名靠前的房东有327个房源。

After that, we proceeded with analyzing boroughs and neighborhood listing densities and what areas were more popular than another.
在那之后，我们继续分析各区和社区的挂牌密度，以及哪些地区比其他地区更受欢迎。

Next, we put good use of our latitude and longitude columns and used to create a geographical heatmap color-coded by the price of listings.
接下来，我们充分利用纬度和经度列，并使用它们来创建按房源价格进行颜色编码的地理热图。

Further, we came back to the first column with name strings and had to do a bit more coding to parse each title and analyze existing trends on how listings are named as well as what was the count for the most used words by hosts.
此外，我们回到带有名称字符串的第一列，并且必须做更多的编码来解析每个标题，并分析清单如何命名的现有趋势，以及房东使用最多的单词的计数。

Lastly, we found the most reviewed listings and analyzed some additional attributes.
最后，我们找到了评论最多的列表，并分析了一些附加属性。

For our data exploration purposes, it also would be nice to have couple additional features, such as positive and negative numeric (0-5 stars) reviews or 0-5 star average review for each listing; addition of these features would help to determine the best-reviewed hosts for NYC along with 'number_of_review' column that is provided.
为了我们的数据探索目的，它还应该有一些额外的功能，如正面和负面的数字(0-5星)评论或0-5星的平均评论;添加这些功能将有助于确定纽约市最受好评的主机以及提供的“number_of_review”列。

Overall, we discovered a very good number of interesting relationships between features and explained each step of the process.
总的来说，我们发现了许多功能之间有趣的关系，并解释了过程的每一步。

This data analytics is very much mimicked on a higher level on Airbnb Data/Machine Learning team for better business decisions, control over the platform, marketing initiatives, implementation of new features and much more.
Airbnb数据/机器学习团队在更高层次上模仿这种数据分析，以实现更好的业务决策和控制

Therefore, I hope this kernel helps everyone!

```
In [ ]:
```