EDA - Course Project

Author: Aadarsh Agarwal

Brief description of the dataset

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. This data file includes all needed information to find out more about hosts, geographical availability, necessary metrics to make predictions and draw conclusions. This data has been taken from Kaggle.

In [1]:

```
%pylab inline

%config InlineBackend.figure_formats = ['svg']

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy import stats
```

```
Populating the interactive namespace from numpy and matplotlib
```

Loading the dataset

In [2]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.

if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
    endpoint_cc5b1ceecefc44b58186cada9dd645ef = 'https://s3.ap.cloud-object-storage.appdomain.cloud'
else:
    endpoint_cc5b1ceecefc44b58186cada9dd645ef = 'https://s3.private.ap.cloud-object-storage.appdomain.cloud'

client_cc5b1ceecefc44b58186cada9dd645ef = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Idz7Zz4PmLMdn-9VVoOjZyrslrbWIpvrsuyHhN3L_rlR',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_cc5b1ceecefc44b58186cada9dd645ef)

body = client_cc5b1ceecefc44b58186cada9dd645ef.get_object(Bucket='exploratorydataanalysis-donotdelete-pr-qqmnz7bsobuldp',Key='AB_NYC_2019.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head(10)
```

Out[2]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitu
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.647
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.753
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.809
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.685
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.798
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill	40.747
6	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.686
7	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Hell's Kitchen	40.764
8	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40.801
9	5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Chinatown	40.713

Attributes

Lets begin by examining the data

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

Data Cleaning

Before moving on with analyzing the data, we will restrict ourselves to the rows important to the goal of predicting the room price.

In [4]:

```
df.drop(['latitude','longitude','last_review','neighbourhood'],axis=1,inplace=True)
```

In [5]:

```
df.head(10)
```

Out[5]:

	id	name	host_id	host_name	neighbourhood_group	room_type	price	mir
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	Entire home/apt	80	
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Entire home/apt	200	
6	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Private room	60	
7	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Private room	79	
8	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Private room	79	
9	5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Entire home/apt	150	

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   room_type                       48895 non-null  object 
 6   price                           48895 non-null  int64  
 7   minimum_nights                  48895 non-null  int64  
 8   number_of_reviews               48895 non-null  int64  
 9   reviews_per_month               38843 non-null  float64
 10  calculated_host_listings_count  48895 non-null  int64  
 11  availability_365                48895 non-null  int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 4.5+ MB
```

From above it can be seen that there are three columns with null values, namely **name**, **host_name** and **reviews_per_month**. Now, for both **name** and **host_name**, there is a respective **id** and **host_id**, and hence no need to worry about those particular as names in particular are not much imporatant while modelling.

Also, from the snippet below, we can see that **reviews_per_month** has null values corresponding to **number_of_reviews** being 0. Hence we can conclude that **reviews_per_month** is also 0 in those places.

In [7]:

```
df[df['reviews_per_month'].isna()][['number_of_reviews','reviews_per_month']]
```

Out[7]:

	number_of_reviews	reviews_per_month
2	0	NaN
19	0	NaN
26	0	NaN
36	0	NaN
38	0	NaN
...	...	...
48890	0	NaN
48891	0	NaN
48892	0	NaN
48893	0	NaN
48894	0	NaN

10052 rows × 2 columns

Therefore, lets fill 0 wherever **reviews_per_month** is null.

In [8]:

```
df['reviews_per_month'].fillna(0, inplace=True)
df.head(10)
```

Out[8]:

	id	name	host_id	host_name	neighbourhood_group	room_type	price	mir
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	Entire home/apt	80	
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Entire home/apt	200	
6	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Private room	60	
7	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Private room	79	
8	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Private room	79	
9	5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Entire home/apt	150	

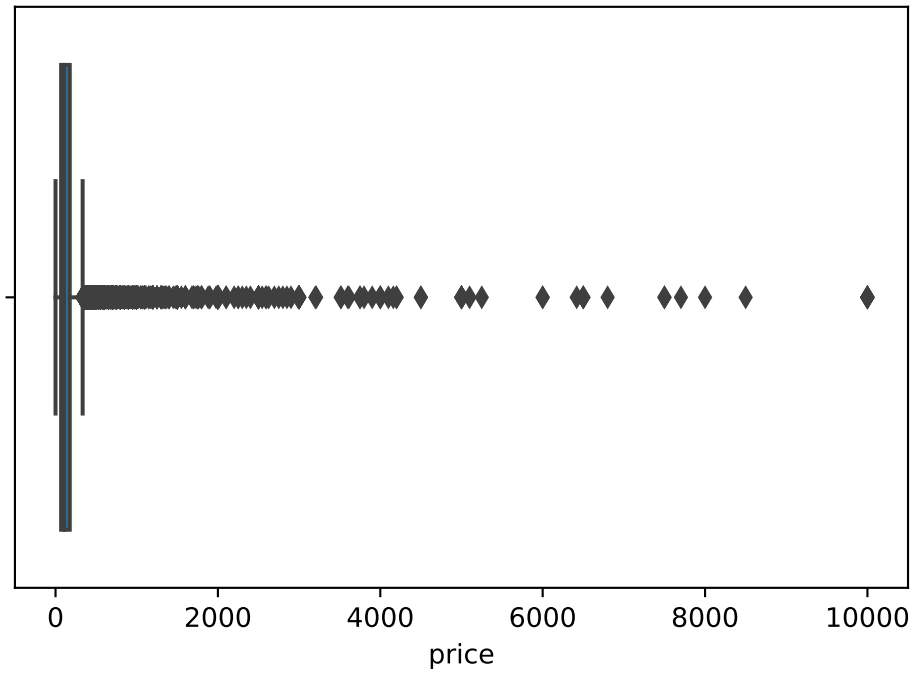
Handling Outliers

In [9]:

```
sns.boxplot(x=df['price'])
```

Out[9]:

```
<AxesSubplot:xlabel='price'>
```



From the given data and the figure above it is evident that price of some rooms was zero which is not possible. Hence we won't be considering these rows.

In [10]:

```
df = df.loc[df["price"] != 0]
print(df[df['price']>334]['price'].shape)
```

```
(2972,)
```

Another thing evident is that there are quite a few outliers, so we can remove them before proceeding further. As shown above the ouliers make up a very insignificant portion of the data and hence removing them won't disrupt our dataset

In [11]:

```
q1=np.percentile(df['price'],25)
q3=np.percentile(df['price'],75)
upr_bound=q3+1.5*(q3-q1)
df=df[df['price']<=334]
```

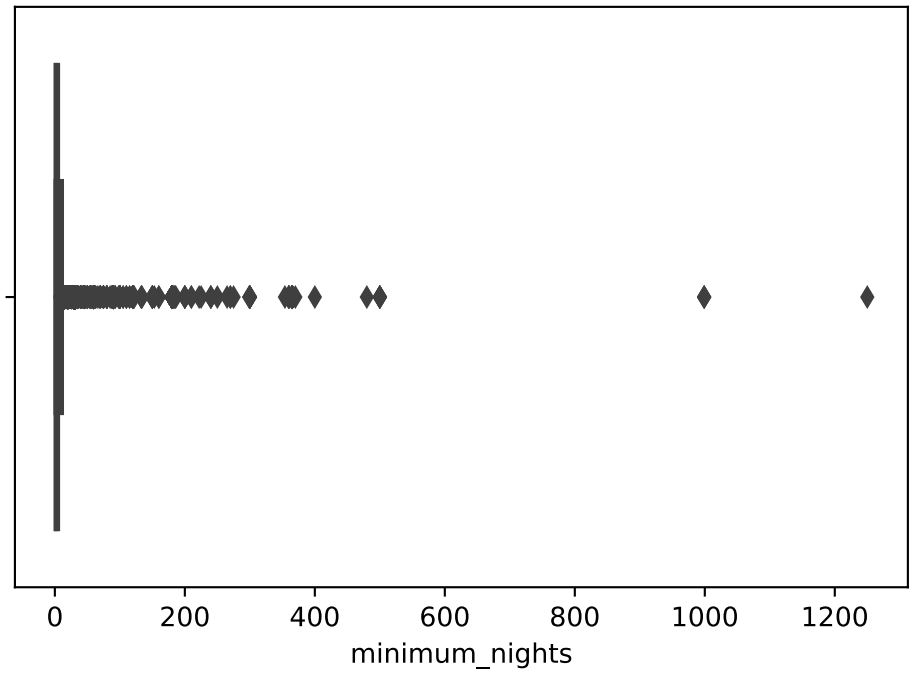
Same thing can be done for other features as well

In [12]:

```
sns.boxplot(x=df['minimum_nights'])
```

Out[12]:

<AxesSubplot:xlabel='minimum_nights'>



In [13]:

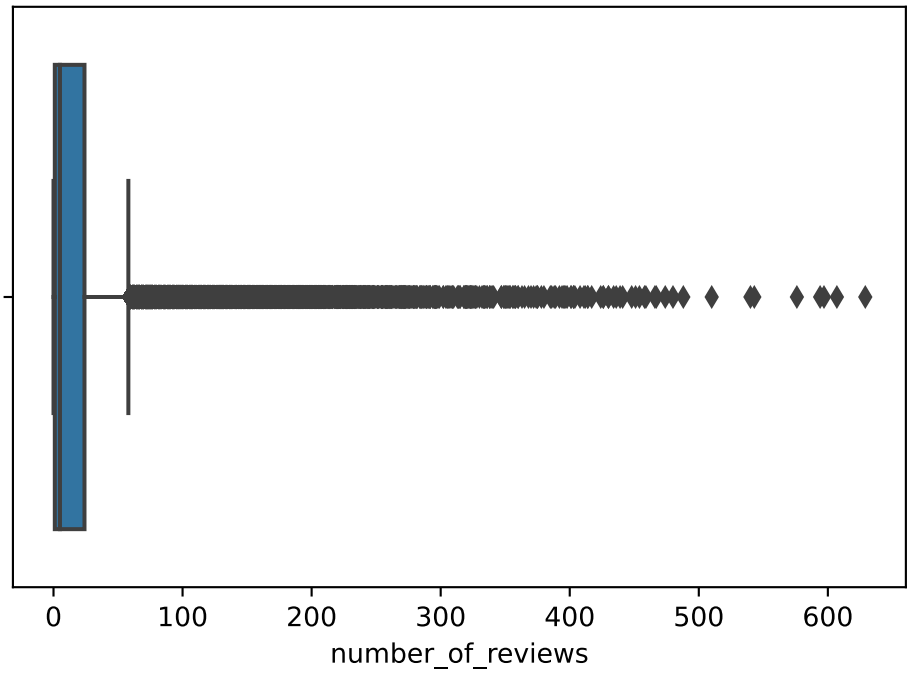
```
df=df[df['minimum_nights']<600]
```

In [14]:

```
sns.boxplot(x=df['number_of_reviews'])
```

Out[14]:

```
<AxesSubplot:xlabel='number_of_reviews'>
```



In [15]:

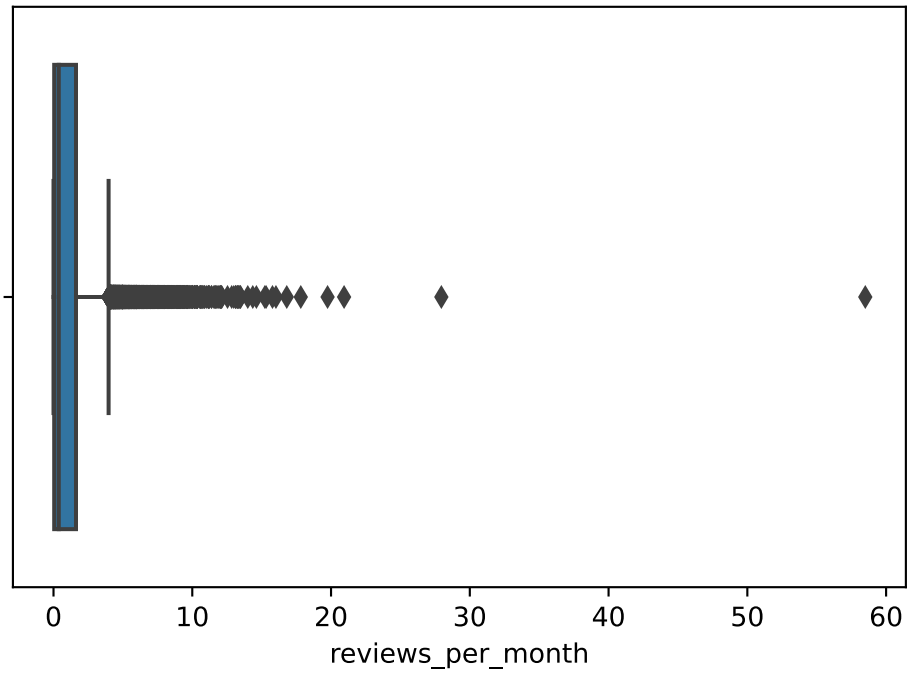
```
df=df[df['number_of_reviews']<=500]
```

In [16]:

```
sns.boxplot(x=df['reviews_per_month'])
```

Out[16]:

```
<AxesSubplot:xlabel='reviews_per_month'>
```



In [17]:

```
df=df[df['reviews_per_month']<=20]
```

In [18]:

```
df.info()
```

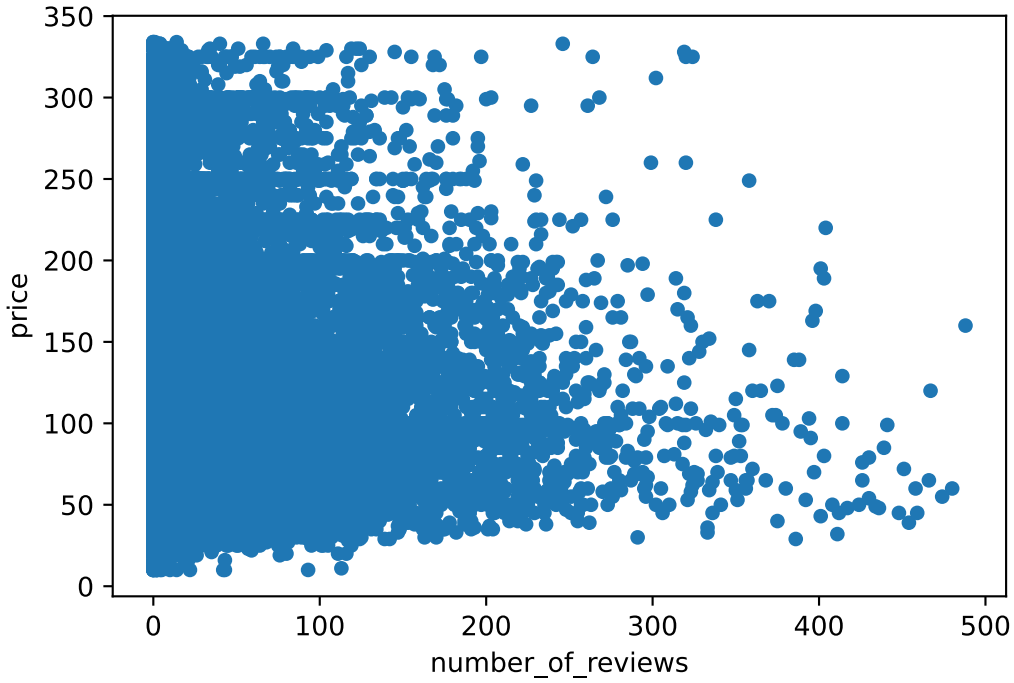
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45897 entries, 0 to 48894
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              45897 non-null  int64  
 1   name                            45882 non-null  object 
 2   host_id                         45897 non-null  int64  
 3   host_name                       45876 non-null  object 
 4   neighbourhood_group             45897 non-null  object 
 5   room_type                       45897 non-null  object 
 6   price                           45897 non-null  int64  
 7   minimum_nights                  45897 non-null  int64  
 8   number_of_reviews               45897 non-null  int64  
 9   reviews_per_month               45897 non-null  float64
 10  calculated_host_listings_count  45897 non-null  int64  
 11  availability_365                45897 non-null  int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 4.6+ MB
```

Scaling

At this point if I plot a scatter plot for any two features, it would be somewhat like below which makes scaling all the more neccessary.

In [19]:

```
df.plot.scatter(x='number_of_reviews',y='price')
plt.show()
```



In [20]:

```
from sklearn.preprocessing import RobustScaler

df_1=df.copy()
cols=['price','minimum_nights','number_of_reviews','reviews_per_month','calculated_host
_listings_count','availability_365']
trans = RobustScaler(quantile_range=(20, 80))
da = trans.fit_transform(df_1[cols].values)
df_1[cols]=da
df_1
```

Out[20]:

	id	name	host_id	host_name	neighbourhood_group	room_type
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Private room
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Entire home/apt
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Private room
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Entire home/apt
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	Entire home/apt
...	...	...	...	...	...	...
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Private room
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Private room
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Entire home/apt
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Shared room
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Private room

45897 rows × 12 columns

Moving on, we'll use this dataset without features **name** and **host_name** before starting with Feature Engineering.

In [21]:

```
df_1=df_1.drop(['name','host_name'],axis=1)
df_1.head(10)
```

Out[21]:

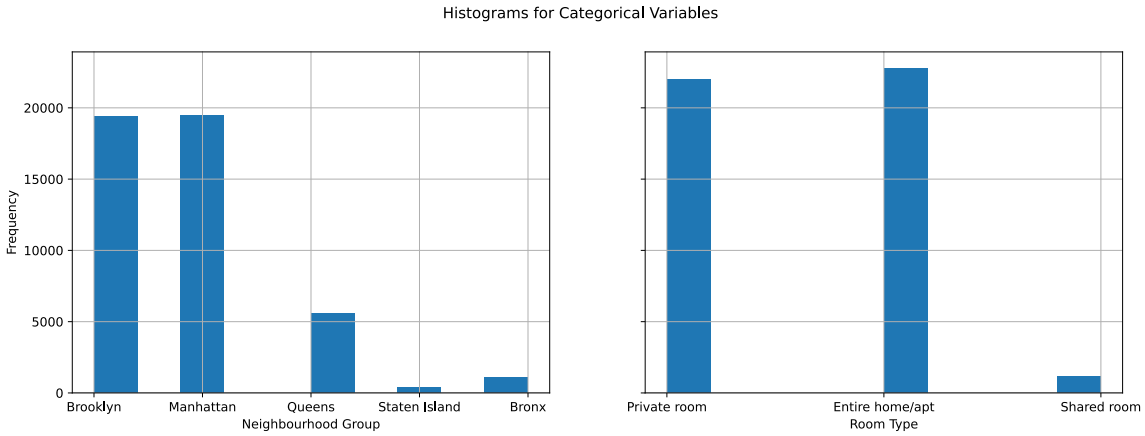
	id	host_id	neighbourhood_group	room_type	price	minimum_nights	number_of_r
0	2539	2787	Brooklyn	Private room	0.418803	-0.2	0.
1	2595	2845	Manhattan	Entire home/apt	1.068376	-0.2	1.
2	3647	4632	Manhattan	Private room	0.427350	0.2	-0.
3	3831	4869	Brooklyn	Entire home/apt	-0.094017	-0.2	8.
4	5022	7192	Manhattan	Entire home/apt	-0.170940	1.6	0.
5	5099	7322	Manhattan	Entire home/apt	0.854701	0.2	2.
6	5121	7356	Brooklyn	Private room	-0.341880	8.6	1.
7	5178	8967	Manhattan	Private room	-0.179487	0.0	12.
8	5203	7490	Manhattan	Private room	-0.179487	0.0	3.
9	5238	7549	Manhattan	Entire home/apt	0.427350	-0.2	4.

Feature Engineering

Since we are left with only two non-numerical data, we will try to analyze them first.

In [22]:

```
fig, axes = plt.subplots(1, 2, sharey=True, figsize=(15,5))
fig.suptitle('Histograms for Categorical Variables')
df_1['neighbourhood_group'].hist(ax=axes[0])
df_1['room_type'].hist(ax=axes[1])
axes[0].set_xlabel("Neighbourhood Group")
axes[0].set_ylabel("Frequency")
axes[1].set_xlabel("Room Type")
fig.show()
```



As seen from the graph above, there are **5** neighbourhoods. The count of both **Brooklyn** and **Manhattan** is much more than the others. So we can club the rest together and use one-hot encoding for the variable. The same can be done for **Room Type**.

In [23]:

```
df_1['neighbourhood_group'].replace(['Queens','Staten Island','Bronx'],'Others',True)
```

In [24]:

```
df_2 = pd.get_dummies(df_1, columns=['neighbourhood_group','room_type'])
df_2
```

Out[24]:

	id	host_id	price	minimum_nights	number_of_reviews	reviews_per_mo
0	2539	2787	0.418803	-0.2	0.121212	-0.084
1	2595	2845	1.068376	-0.2	1.212121	0.000
2	3647	4632	0.427350	0.2	-0.151515	-0.189
3	3831	4869	-0.094017	-0.2	8.030303	2.119
4	5022	7192	-0.170940	1.6	0.121212	-0.139
...	...	...	...	...	...	
48890	36484665	8232441	-0.256410	0.0	-0.151515	-0.189
48891	36485057	6570630	-0.512821	0.4	-0.151515	-0.189
48892	36485431	23492952	0.128205	1.6	-0.151515	-0.189
48893	36485609	30985759	-0.384615	-0.2	-0.151515	-0.189
48894	36487245	68119814	-0.085470	1.0	-0.151515	-0.189

45897 rows × 14 columns

Hypothesis

1. Manhattan Listings are most expensive.
2. More the minimum number of nights, more is the price.
3. Private rooms have the most availability.

I'll be working on the first hypothesis only in this case. So let me cleary defne the hypothesis.

Null Hypothesis: Manhattan Listings have same prices.

Alternate Hypothesis: Manhattan Listings are the most expensive.

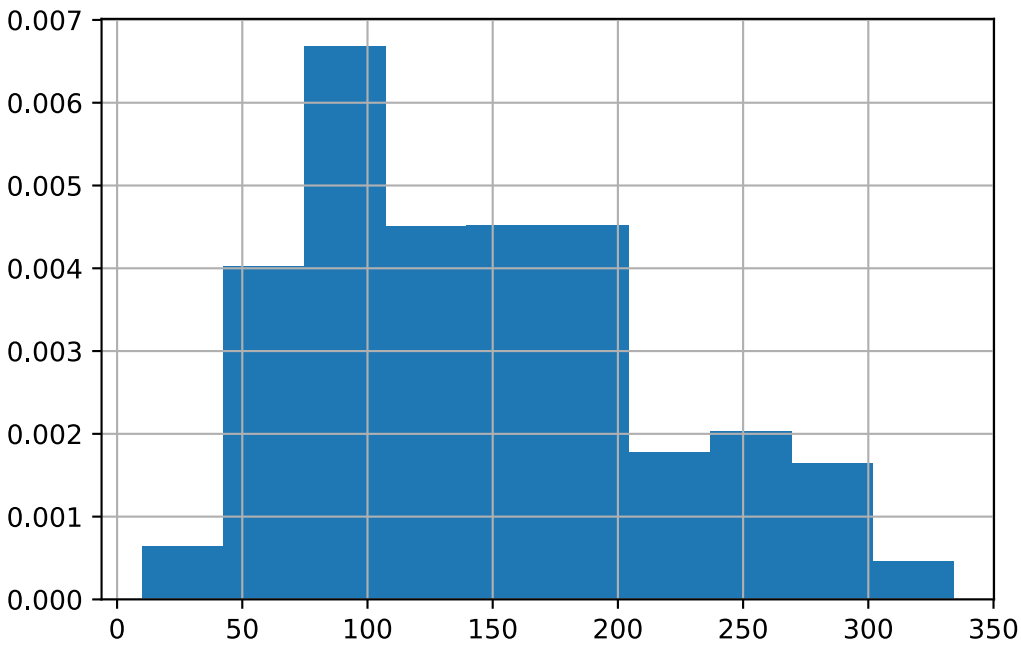
The test statistic here is the price of listings.

In [25]:

```
df[df['neighbourhood_group']=='Manhattan']['price'].hist(density=True)
```

Out[25]:

```
<AxesSubplot:>
```



Hypothesis Testing

Performing Z-test

In [26]:

```
mu=df['price'].mean()
sd=df['price'].std()
smu=df[df['neighbourhood_group']=='Manhattan']['price'].mean()
n=df[df['neighbourhood_group']=='Manhattan']['price'].count()
```

In [27]:

```
from statsmodels.stats import weightstats as stests
ztest ,pval = stests.ztest(df[df['neighbourhood_group']=='Manhattan']['price'], x2=None
, value=mu)
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.0
reject null hypothesis
```

Since our sample size is so large, the probability we get in the above case for our null case is 0. Thus we reject the null hypotheis and accept the alternate hypothesis that **Manhattan Listings are the most expensive.**

Suggestions and Quality of Data

The dataset is of average quality in my regards. There is no strong correlattion between any attributes inspite of extensive cleaning. Despite being a large dataset, the values are repetitive and hence are not of much use. Additional attributes such as area, services offered or any physical characteristics of listings might better help making any kind of predictions from the data.

In [ ]: