EDA- Air Bnb

"The dataset is obtained from Kaggle and mainly focuses on data cleaning, addressing missing values, inconsistent data, and outliers."

In [1]:
```
# importing libaries

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

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [2]:
```
plt.rcParams["figure.figsize"] = (10, 5)
```

In [3]:
```
# Reading the csv.

df = pd.read_csv('D:\Relevel\SQL Project/Air BNB/Airbnb_Open_Data.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	id	NAME	host id	host_identity_verified	host name	neighbourhood group	neighbou
0	1001254.0	Clean & quiet apt home by the park	8.001449e+10	unconfirmed	Madaline	Brooklyn	Kens
1	1002102.0	Skylit Midtown Castle	5.233517e+10	verified	Jenna	Manhattan	M
2	1002403.0	THE VILLAGE OF HARLEM....NEW YORK !	7.882924e+10	NaN	Elise	Manhattan	
3	1002755.0	NaN	8.509833e+10	unconfirmed	Garry	Brooklyn	Clint
4	1003689.0	Entire Apt: Spacious Studio/Loft by central park	9.203760e+10	verified	Lyndon	Manhattan	East

5 rows × 27 columns

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

Out[5]:
```
(102603, 27)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102603 entries, 0 to 102602
Data columns (total 27 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   id                              102599 non-null  float64
 1   NAME                            102349 non-null  object 
 2   host id                         102599 non-null  float64
 3   host_identity_verified          102310 non-null  object 
 4   host name                       102193 non-null  object 
 5   neighbourhood group             102570 non-null  object 
 6   neighbourhood                   102583 non-null  object 
 7   lat                             102591 non-null  float64
 8   long                            102591 non-null  float64
 9   country                         102067 non-null  object 
 10  country code                    102468 non-null  object 
 11  instant_bookable                102494 non-null  object 
 12  cancellation_policy             102523 non-null  object 
 13  room type                       102599 non-null  object 
 14  Construction year               102385 non-null  float64
 15  price                           102354 non-null  object 
 16  service fee                     102326 non-null  object 
 17  minimum nights                  102190 non-null  float64
 18  Avg_per_night                   102599 non-null  object 
 19  number of reviews               102416 non-null  float64
 20  last review                     86706 non-null   object 
 21  reviews per month               86720 non-null   float64
 22  review rate number              102273 non-null  float64
 23  calculated host listings count  102280 non-null  float64
 24  availability 365                102151 non-null  float64
 25  house_rules                     50468 non-null   object 
 26  license                         2 non-null       object 
dtypes: float64(11), object(16)
memory usage: 21.1+ MB
```

In [7]:
```
df.dtypes
```

Out[7]:

```
id                                float64
NAME                               object
host id                           float64
host_identity_verified             object
host name                          object
neighbourhood group                object
neighbourhood                      object
lat                               float64
long                              float64
country                            object
country code                       object
instant_bookable                   object
cancellation_policy                object
room type                          object
Construction year                 float64
price                              object
service fee                        object
minimum nights                    float64
Avg_per_night                      object
number of reviews                 float64
last review                        object
reviews per month                 float64
review rate number                float64
calculated host listings count    float64
availability 365                  float64
house_rules                        object
license                            object
dtype: object
```

In [8]:

```
df.describe(include = 'all').T
```

Out[8]:

	count	unique	top	freq	mean	std
id	102599.0	NaN	NaN	NaN	29146234.52213	16257505.607309
NAME	102349	61281	Home away from home	33	NaN	NaN
host id	102599.0	NaN	NaN	NaN	49254111474.328667	28538996644.374817
host_identity_verified	102310	2	unconfirmed	51200	NaN	NaN
host name	102193	13190	Michael	881	NaN	NaN
neighbourhood group	102570	7	Manhattan	43792	NaN	NaN
neighbourhood	102583	224	Bedford-Stuyvesant	7937	NaN	NaN
lat	102591.0	NaN	NaN	NaN	40.728094	0.055857
long	102591.0	NaN	NaN	NaN	-73.949644	0.049521
country	102067	1	United States	102067	NaN	NaN
country code	102468	1	US	102468	NaN	NaN
instant_bookable	102494	2	False	51474	NaN	NaN
cancellation_policy	102523	3	moderate	34343	NaN	NaN
room type	102599	4	Entire home/apt	53701	NaN	NaN
Construction year	102385.0	NaN	NaN	NaN	2012.487464	5.765556
price	102354	1151	$206	137	NaN	NaN
service fee	102326	231	$41	526	NaN	NaN
minimum nights	102190.0	NaN	NaN	NaN	8.135845	30.553781
Avg_per_night	102599	2073	$27.00	440	NaN	NaN
number of reviews	102416.0	NaN	NaN	NaN	27.483743	49.508954
last review	86706	2477	6/23/2019	2443	NaN	NaN
reviews per month	86720.0	NaN	NaN	NaN	1.374022	1.746621
review rate number	102273.0	NaN	NaN	NaN	3.279106	1.284657
calculated host listings count	102280.0	NaN	NaN	NaN	7.936605	32.21878
availability 365	102151.0	NaN	NaN	NaN	141.133254	135.435024
house_rules	50468	1976	#NAME?	2712	NaN	NaN
license	2	1	41662/AL	2	NaN	NaN

In [9]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[9]:
```
license                           102601
house_rules                        52135
last review                        15897
reviews per month                  15883
country                              536
availability 365                     452
minimum nights                       413
host name                            410
review rate number                   330
calculated host listings count       323
host_identity_verified               293
service fee                          277
NAME                                 254
price                                249
Construction year                    218
number of reviews                    187
country code                         135
instant_bookable                     109
cancellation_policy                   80
neighbourhood group                   33
neighbourhood                         20
long                                  12
lat                                   12
id                                     4
Avg_per_night                          4
host id                                4
room type                              4
dtype: int64
```

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

Out[10]:
```
id                                102058
NAME                               61281
host id                           102057
host_identity_verified                 2
host name                          13190
neighbourhood group                    7
neighbourhood                        224
lat                                21991
long                               17774
country                                1
country code                           1
instant_bookable                       2
cancellation_policy                    3
room type                              4
Construction year                     20
price                               1151
service fee                          231
minimum nights                       153
Avg_per_night                       2073
number of reviews                    476
last review                         2477
reviews per month                   1016
review rate number                     5
calculated host listings count        78
availability 365                     438
house_rules                         1976
license                                1
dtype: int64
```

In [11]:
```
# Copying tha data to df1

df1 = df.copy()
```

Data Cleaning

In [12]:
```
# Converting Price and service fee from Object(sting) to numeric

df1['price'] = df1['price'].str.replace('$', '').str.replace(',', '').str.replace(' '
df1['price'] = pd.to_numeric(df1['price'])
df1['service fee'] = df1['service fee'].str.replace('$', '').str.replace(',', '').str.
df1['service fee'] = pd.to_numeric(df1['service fee'])
```

In [13]:
```
df1.describe(include = 'all').T
```

Out[13]:

	count	unique	top	freq	mean	std
id	102599.0	NaN	NaN	NaN	29146234.52213	16257505.607309
NAME	102349	61281	Home away from home	33	NaN	NaN
host id	102599.0	NaN	NaN	NaN	49254111474.328667	28538996644.374817
host_identity_verified	102310	2	unconfirmed	51200	NaN	NaN
host name	102193	13190	Michael	881	NaN	NaN
neighbourhood group	102570	7	Manhattan	43792	NaN	NaN
neighbourhood	102583	224	Bedford-Stuyvesant	7937	NaN	NaN
lat	102591.0	NaN	NaN	NaN	40.728094	0.055857
long	102591.0	NaN	NaN	NaN	-73.949644	0.049521
country	102067	1	United States	102067	NaN	NaN
country code	102468	1	US	102468	NaN	NaN
instant_bookable	102494	2	False	51474	NaN	NaN
cancellation_policy	102523	3	moderate	34343	NaN	NaN
room type	102599	4	Entire home/apt	53701	NaN	NaN
Construction year	102385.0	NaN	NaN	NaN	2012.487464	5.765556
price	102354.0	NaN	NaN	NaN	625.293521	331.668373
service fee	102326.0	NaN	NaN	NaN	125.026924	66.325739
minimum nights	102190.0	NaN	NaN	NaN	8.135845	30.553781
Avg_per_night	102599	2073	$27.00	440	NaN	NaN
number of reviews	102416.0	NaN	NaN	NaN	27.483743	49.508954
last review	86706	2477	6/23/2019	2443	NaN	NaN
reviews per month	86720.0	NaN	NaN	NaN	1.374022	1.746621
review rate number	102273.0	NaN	NaN	NaN	3.279106	1.284657
calculated host listings count	102280.0	NaN	NaN	NaN	7.936605	32.21878
availability 365	102151.0	NaN	NaN	NaN	141.133254	135.435024
house_rules	50468	1976	#NAME?	2712	NaN	NaN
license	2	1	41662/AL	2	NaN	NaN

In [14]:
```
df1.columns = [col.lower().replace(" ","_") for col in df1.columns]
df1.columns
```

Out[14]:
```
Index(['id', 'name', 'host_id', 'host_identity_verified', 'host_name',
       'neighbourhood_group', 'neighbourhood', 'lat', 'long', 'country',
       'country_code', 'instant_bookable', 'cancellation_policy', 'room_type',
       'construction_year', 'price', 'service_fee', 'minimum_nights',
       'avg_per_night', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'review_rate_number',
       'calculated_host_listings_count', 'availability_365', 'house_rules',
       'license'],
      dtype='object')
```

Checking the null values in data set

In [15]:
```
df1.isnull().sum().sort_values(ascending = False)
```

Out[15]:
```
license                           102601
house_rules                        52135
last_review                        15897
reviews_per_month                  15883
country                              536
availability_365                     452
minimum_nights                       413
host_name                            410
review_rate_number                   330
calculated_host_listings_count       323
host_identity_verified               293
service_fee                          277
name                                 254
price                                249
construction_year                    218
number_of_reviews                    187
country_code                         135
instant_bookable                     109
cancellation_policy                   80
neighbourhood_group                   33
neighbourhood                         20
long                                  12
lat                                   12
id                                     4
avg_per_night                          4
host_id                                4
room_type                              4
dtype: int64
```

Droping the license column because almost all the rows is having null values

In [16]:
```
df1.drop('license', axis = 1, inplace = True)
```

Filling blank to all null values in house rules columns as it might differ from owner to owner

In [17]:
```
df1['house_rules'].fillna('Blank', inplace = True)
```

Converting the last review column to Date and Time

In [18]:
```
df1['last_review']
```

Out[18]:
```
0          10/19/2021
1           5/21/2022
2                 NaN
3           7/5/2019
4          11/19/2018
              ...
102598            NaN
102599            NaN
102600            NaN
102601            NaN
102602            NaN
Name: last_review, Length: 102603, dtype: object
```

In [19]:
```
df1['last_review'] = pd.to_datetime(df1['last_review'])
```

In [20]:
```
df1['last_review'].min(), df1['last_review'].max()
```

Out[20]:
```
(Timestamp('2012-07-11 00:00:00'), Timestamp('2058-06-16 00:00:00'))
```

Certain columns have future dates, so converting those dates to the median of the data set

In [21]:
```
df1[df1['last_review'].apply(lambda x:x.year) > 2023]
```

Out[21]:

	id	name	host_id	host_identity_verified	host_name	neighbourhood_group	r
127	1071478.0	Garden studio in the Upper East Sid	7.717256e+10	unconfirmed	Miller	Manhattan	
191	1106825.0	LUX APT IN TIMES SQUARE NEW BUILDING	9.372536e+10	unconfirmed	Aiden	Manhattan	
255	1142173.0	Beautiful Landmarked Duplex	8.794478e+10	NaN	Baker	Brooklyn	
318	1176967.0	NaN	7.008447e+10	verified	Barnes	Brooklyn	
483	1268097.0	Modern Space in Charming Pre-war	1.374659e+10	verified	Adelaide	Manhattan	

5 rows × 26 columns

In [22]:
```
df1[df1['last_review'].apply(lambda x:x.year) > 2023] = df1['last_review'].median()
df1['last_review'].fillna(df1['last_review'].median, inplace = True)
```

To remove dupicates if any

In [23]:
```
df1.drop_duplicates(keep = 'first', inplace = True)
```

In [24]:
```
# Checking how many mull values is pending

df1.isnull().sum().sort_values(ascending = False)
```

Out[24]:
```
reviews_per_month                 15821
country                             535
availability_365                    450
host_name                           407
minimum_nights                      402
review_rate_number                  322
calculated_host_listings_count      321
host_identity_verified              291
service_fee                         276
name                                252
price                               248
construction_year                   215
number_of_reviews                   186
country_code                        134
instant_bookable                    108
cancellation_policy                  79
neighbourhood_group                  32
neighbourhood                        19
long                                 11
lat                                  11
id                                    3
room_type                             3
avg_per_night                         3
host_id                               3
last_review                           0
house_rules                           0
dtype: int64
```

In [25]:
```
print(df1['reviews_per_month'].dtype)
```

```
object
```

Converting reviews_per_month to numeric to plot the data

In [26]:
```
print(df1['reviews_per_month'].unique())
```

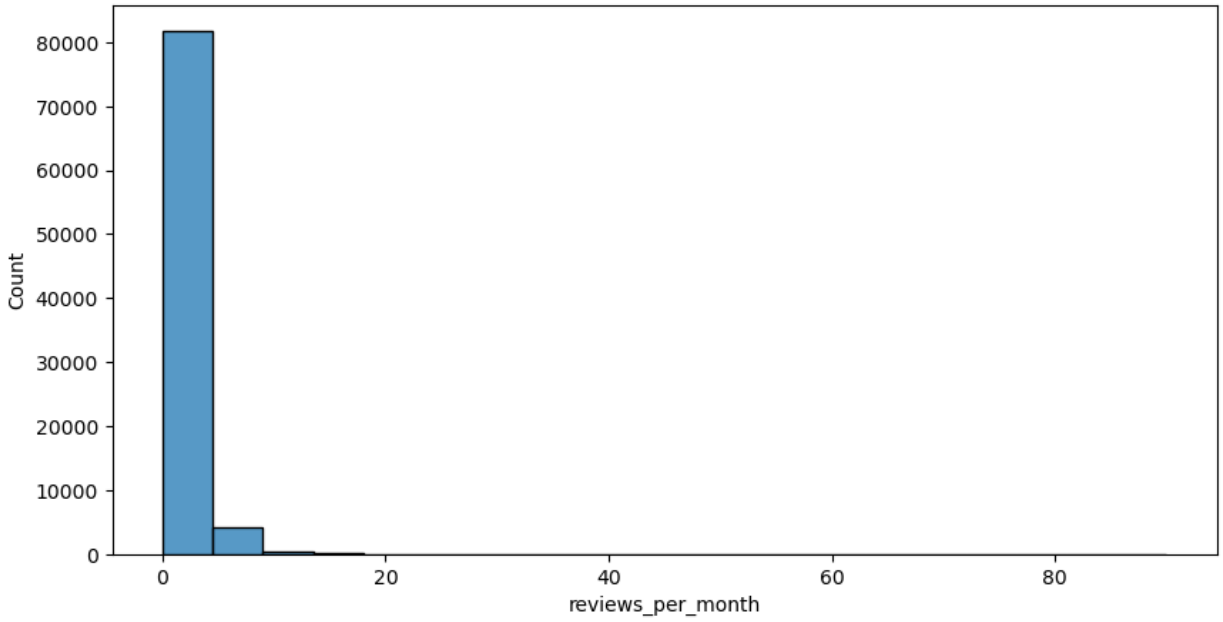
```
[0.21 0.38 nan ... 7.73 24.49 33.08]
```

In [27]:
```
df1['reviews_per_month'] = df1['reviews_per_month'].astype(str)
```

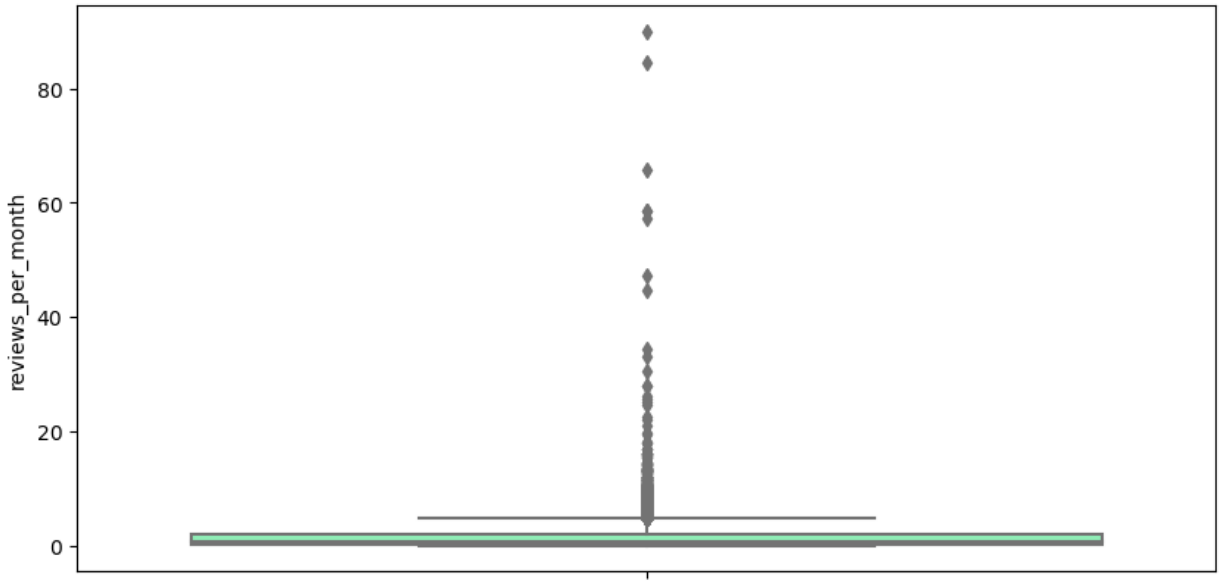
In [28]:
```
df1['reviews_per_month'] = df1['reviews_per_month'].str.replace(',', '')
```

In [29]:
```
df1['reviews_per_month'] = pd.to_numeric(df1['reviews_per_month'], errors='coerce')
```

In [30]:
```
sns.histplot(df1['reviews_per_month'], bins = 20);
```



In [31]:
```
sns.boxplot(data = df1, y = 'reviews_per_month', palette = 'rainbow');
```



In [32]:
```
df1['reviews_per_month'].median()
```

Out[32]:
```
0.74
```

There are so many outliers in the 'reviews_per_month' column, so we will take the median to fill the null values. This ensures that our end output remains unchanged.

In [33]:
```
df1['reviews_per_month'].fillna(df1['reviews_per_month'].median(), inplace = True)
```

Dropping the column country and country code as data set which we are using is of USA.

In [34]:
```
df1.drop(['country','country_code'],axis = 1, inplace = True)
```

In [35]:
```
df['neighbourhood group'].value_counts()
```

Out[35]:
```
Manhattan        43792
Brooklyn         41842
Queens           13267
Bronx             2712
Staten Island      955
brookln              1
manhatan             1
Name: neighbourhood group, dtype: int64
```

There is spelling and lower case mistake

In [36]:
```
df1['neighbourhood_group'] = df1['neighbourhood_group'].replace({'brookln':'Brooklyn',
```

In [37]:
```
df1['neighbourhood_group'].value_counts()
```

Out[37]:
```
Manhattan              43555
Brooklyn               41629
Queens                 13197
Bronx                   2694
Staten Island            949
2019-06-14 00:00:00        1
Name: neighbourhood_group, dtype: int64
```

There is an unknown row in the 'neighbourhood_group' column. I will check it and proceed with dropping it.

In [38]:
```
df1[df1['neighbourhood_group'] == '2019-06-14 00:00:00' ]
```

Out[38]:

id	name	host_id	host_identity_verified	host_name	neighbourhood_group	neighbourhood	lat	lo

0 rows × 24 columns

In [39]:
```
# It show datafram doesn't have value of "2019-06-14 00:00:00", it means this is data

print(df1[df1['neighbourhood_group'] == '2019-06-14 00:00:00'])
```

```
Empty DataFrame
Columns: [id, name, host_id, host_identity_verified, host_name, neighbourhood_group,
neighbourhood, lat, long, instant_bookable, cancellation_policy, room_type, construct
ion_year, price, service_fee, minimum_nights, avg_per_night, number_of_reviews, last_
review, reviews_per_month, review_rate_number, calculated_host_listings_count, availa
bility_365, house_rules]
Index: []

[0 rows x 24 columns]
```

In [40]:
```
print(df1['neighbourhood_group'].unique())
```

```
['Brooklyn' 'Manhattan' 'Queens' nan Timestamp('2019-06-14 00:00:00')
 'Staten Island' 'Bronx']
```

In [41]:
```
df1['neighbourhood_group'].dtype
```

Out[41]:
```
dtype('O')
```

In [42]:
```
df1[df1['neighbourhood_group'].astype(str) == '2019-06-14 00:00:00']
```

Out[42]:

	id	name	host_id	host_identity_verified	host_name	neighbourhood_group	neighbourh
127	2019-06-14 00:00:00	2019-06-14 00:00:00	2019-06-14 00:00:00	2019-06-14 00:00:00	2019-06-14 00:00:00	2019-06-14 00:00:00	2019-06 00:0

1 rows × 24 columns

In [43]:
```
# Drop the row by index

df1.drop(index= 127 , axis=0, inplace=True)
```

In [44]:
```
df1['neighbourhood_group'].value_counts()
```

Out[44]:
```
Manhattan        43555
Brooklyn         41629
Queens           13197
Bronx             2694
Staten Island      949
Name: neighbourhood_group, dtype: int64
```

Filling null values of column Name and Host_name by "Blank", Host_id by "0" and host_Identity_Verified by "Unconfirmed"

In [45]:
```
df1['name'].fillna('Blank', inplace = True)
```

In [46]:
```
df1['host_id'].fillna('0', inplace = True)
```

In [47]:
```
df1['host_identity_verified'].fillna('unconfirmed',inplace = True)
```

In [48]:
```
df1['host_name'].fillna('Blank', inplace = True)
```

Neighbourhood_Group and Neighbourhood columns are related to each other as both contain places. To fill the values of the Neighbourhood_Group column, I have grouped the Neighbourhood_Group by the unique values in the Neighbourhood column and stored in dictionary.

In [49]:
```
df_non_null = df1[df1['neighbourhood_group'].notnull()]
```

In [50]:
```
grouped_neighbourhoods = df_non_null.groupby('neighbourhood_group')['neighbourhood'].u
```

In [51]:
```
grouped_neighbourhoods
```

Out[51]:
```
{'Bronx': array(['Highbridge', 'Clason Point', 'Kingsbridge', 'Woodlawn',
       'University Heights', 'Allerton', 'Concourse Village', 'Concourse',
       'Wakefield', 'Spuyten Duyvil', 'Mott Haven', 'Longwood',
       'Morris Heights', 'Port Morris', 'Fieldston', 'Mount Eden',
       'City Island', 'Williamsbridge', 'Soundview', 'Co-op City',
       'Parkchester', 'North Riverdale', 'Bronxdale', 'Riverdale',
       'Norwood', 'Claremont Village', 'Fordham', 'Mount Hope',
       'Eastchester', 'Van Nest', 'Morris Park', 'Tremont',
       'East Morrisania', 'Hunts Point', 'Pelham Bay', 'Throgs Neck',
       'West Farms', 'Morrisania', 'Pelham Gardens', 'Belmont',
       'Baychester', 'Melrose', 'Schuylerville', 'Castle Hill',
       'Olinville', 'Edenwald', 'Westchester Square', 'Unionport'],
      dtype=object),
 'Brooklyn': array(['Kensington', 'Clinton Hill', 'Bedford-Stuyvesant', 'South Slop
e',
       'Williamsburg', 'Fort Greene', 'Crown Heights', 'Park Slope',
       'Windsor Terrace', 'Greenpoint', 'Bushwick', 'Flatbush',
       'Prospect-Lefferts Gardens', 'Prospect Heights',
       'Brooklyn Heights', 'Carroll Gardens', 'Gowanus', 'Flatlands',
       'Cobble Hill', 'Boerum Hill', 'DUMBO', 'East Flatbush',
       'Gravesend', nan, 'East New York', 'Sheepshead Bay',
       'Fort Hamilton', 'Bensonhurst', 'Sunset Park', 'Brighton Beach',
       'Cypress Hills', 'Bay Ridge', 'Columbia St', 'Vinegar Hill',
       'Canarsie', 'Borough Park', 'Downtown Brooklyn', 'Midwood',
       'Red Hook', 'Dyker Heights', 'Sea Gate', 'Navy Yard',
       'Brownsville', 'Manhattan Beach', 'Bergen Beach', 'Coney Island',
       'Bath Beach', 'Mill Basin', 'Gerritsen Beach'], dtype=object),
 'Manhattan': array(['Midtown', 'Harlem', 'East Harlem', 'Murray Hill',
       "Hell's Kitchen", 'Upper West Side', 'Chinatown', 'West Village',
       'Chelsea', 'Inwood', 'East Village', 'Lower East Side', 'Kips Bay',
       'SoHo', 'Upper East Side', 'Washington Heights',
       'Financial District', 'Morningside Heights', 'NoHo',
       'Flatiron District', 'Roosevelt Island', 'Greenwich Village',
       'Little Italy', 'Two Bridges', 'Nolita', 'Gramercy', nan,
       'Theater District', 'Tribeca', 'Battery Park City', 'Civic Center',
       'Stuyvesant Town', 'Marble Hill'], dtype=object),
 'Queens': array(['Long Island City', 'Flushing', 'Sunnyside', 'Ridgewood',
       'Jamaica', 'Middle Village', 'Ditmars Steinway', 'Astoria',
       'Rockaway Beach', 'Forest Hills', 'Elmhurst', 'Jackson Heights',
       'St. Albans', 'Rego Park', 'Woodside', 'Briarwood', 'Ozone Park',
       'East Elmhurst', 'Arverne', 'Cambria Heights', 'Bayside',
       'Kew Gardens', 'College Point', 'Glendale', 'Richmond Hill',
       'Queens Village', 'Bellerose', 'Maspeth', 'Woodhaven',
       'Kew Gardens Hills', 'Bay Terrace', 'Whitestone', 'Bayswater',
       'Fresh Meadows', 'Springfield Gardens', 'Howard Beach',
       'Belle Harbor', 'Jamaica Estates', 'Far Rockaway',
       'South Ozone Park', 'Corona', 'Neponsit', 'Laurelton',
       'Holliswood', 'Rosedale', 'Edgemere', 'Jamaica Hills', 'Hollis',
       'Douglaston', 'Little Neck', 'Breezy Point', 'Glen Oaks'],
      dtype=object),
 'Staten Island': array(['St. George', 'Tompkinsville', 'Emerson Hill', 'Shore Acre
s',
       'Arrochar', 'Clifton', 'Graniteville', 'Stapleton',
       'New Springville', 'Tottenville', 'Mariners Harbor', 'Concord',
       'Port Richmond', 'Woodrow', 'Eltingville', 'Lighthouse Hill',
       'West Brighton', 'Great Kills', 'Dongan Hills',
       'Castleton Corners', 'Randall Manor', 'Todt Hill', 'Silver Lake',
       'Grymes Hill', 'New Brighton', 'Midland Beach', 'Richmondtown',
       'Howland Hook', 'New Dorp Beach', "Prince's Bay", 'South Beach',
       'Oakwood', 'Huguenot', 'Grant City', 'Westerleigh',
```

```
'Bay Terrace, Staten Island', 'Fort Wadsworth', 'Rosebank',
'Arden Heights', "Bull's Head", 'New Dorp', 'Rossville',
'Willowbrook', 'Chelsea, Staten Island'], dtype=object)}
```

I used a for loop to fill the null values in the 'Neighbourhood Group' column by matching the values with a dictionary. If a match is found, the corresponding key is assigned as the output in the 'Neighbourhood Group' column.

In [52]:
```
for index, row in df1.iterrows():
    if pd.isnull(row['neighbourhood_group']):
        neighbourhood = row['neighbourhood']
        for group, neighbourhoods in grouped_neighbourhoods.items():
            if neighbourhood in neighbourhoods:
                df1.at[index, 'neighbourhood_group'] = group
                break
```

In [53]:
```
# Drop the column neighbourhood

df1.drop('neighbourhood', axis =1 , inplace = True )
```

In [54]:
```
# Diriving the percentage of instant_bookable

ratio_true = df1['instant_bookable'].value_counts(normalize=True)
ratio_false = 1 - ratio_true
ratio_true
```

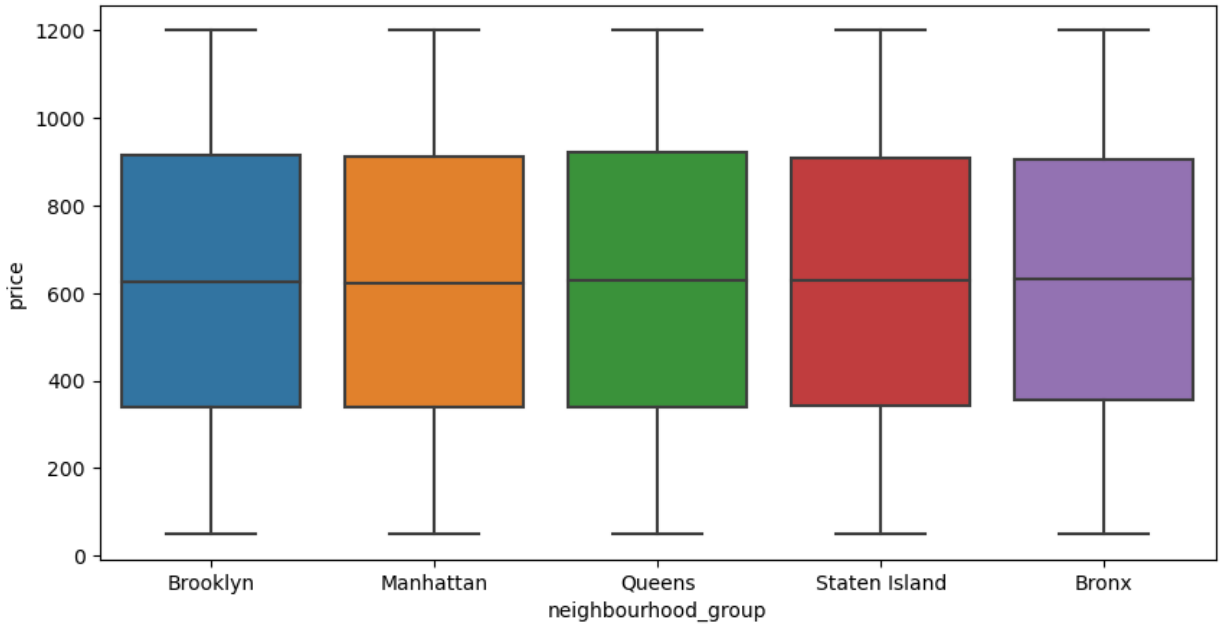
Out[54]:
```
False    0.50205
True     0.49795
Name: instant_bookable, dtype: float64
```

Assigning the value 'Not available' to the null values in the 'instant_bookable' column since it may vary from host to host. However, in the 'cancellation_policy' column, the benefit of the doubt is given to the host and assigning the value 'Moderate' to the null values.

In [55]:
```
df1['instant_bookable'].fillna('Not available', inplace=True)
df1['cancellation_policy'].fillna('Moderate', inplace=True)
```

Checking outlier in price

In [56]:
```
sns.boxplot(data = df1, x = 'neighbourhood_group', y= 'price');
```



In [57]:
```
df1['price'].mean()
```

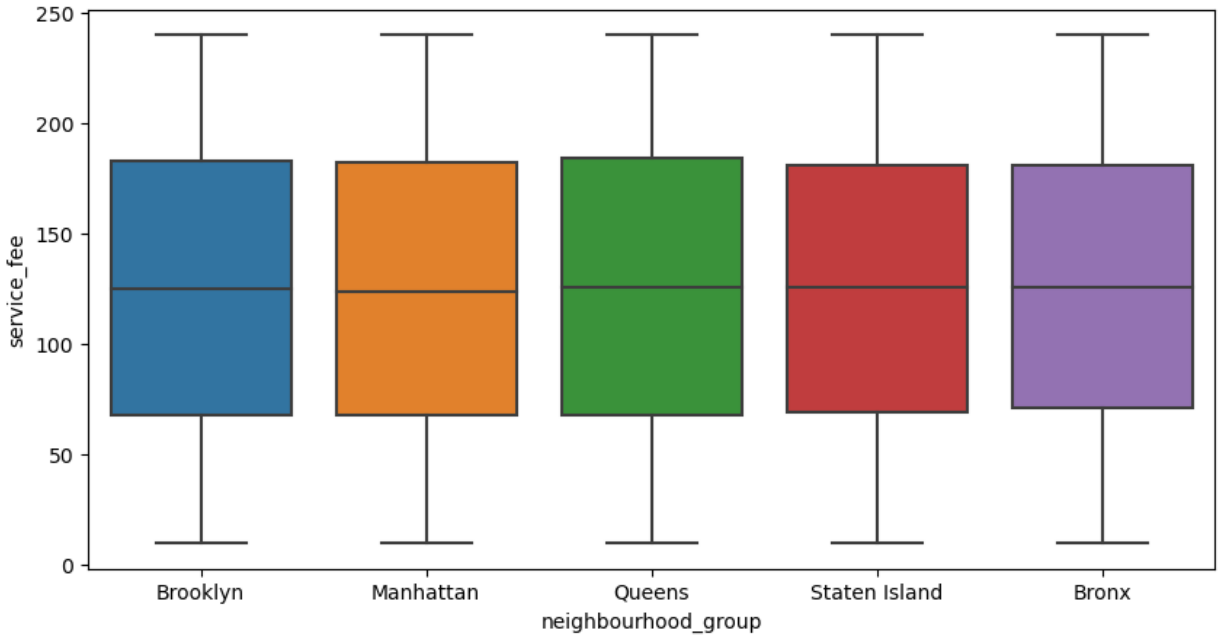
Out[57]:
```
625.3509842055635
```

In [58]:
```
df1['price'].median()
```

Out[58]:
```
624.5
```

Checking outlier in service_fee

In [59]:
```
sns.boxplot(data = df1, x = 'neighbourhood_group', y= 'service_fee');
```



Filling null value with median of price and median of service_fee

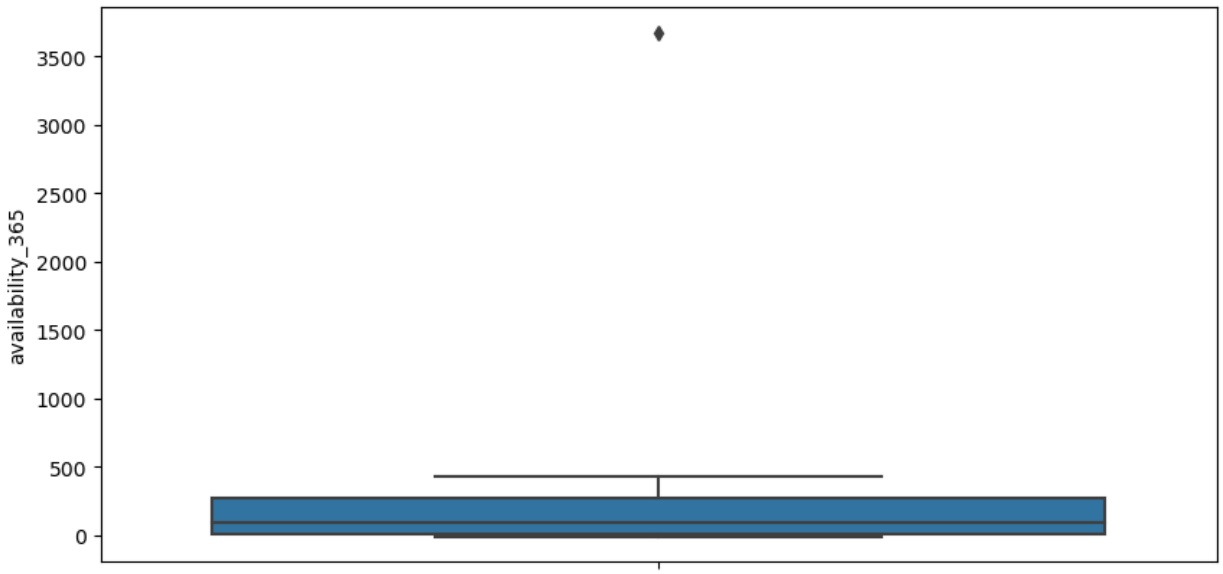
In [60]:
```
df1['price'].fillna(df1['price'].median(), inplace = True)
df1['service_fee'].fillna(df1['service_fee'].median(), inplace = True)
```

In [61]:
```
df1.isnull().sum().sort_values(ascending = False)
```

Out[61]:
```
availability_365                450
minimum_nights                  402
review_rate_number              322
calculated_host_listings_count  321
construction_year               215
number_of_reviews               186
lat                              11
long                             11
room_type                         3
avg_per_night                     3
id                                3
neighbourhood_group               3
name                              0
price                             0
service_fee                       0
cancellation_policy               0
instant_bookable                  0
last_review                       0
reviews_per_month                 0
host_name                         0
host_identity_verified            0
host_id                           0
house_rules                       0
dtype: int64
```

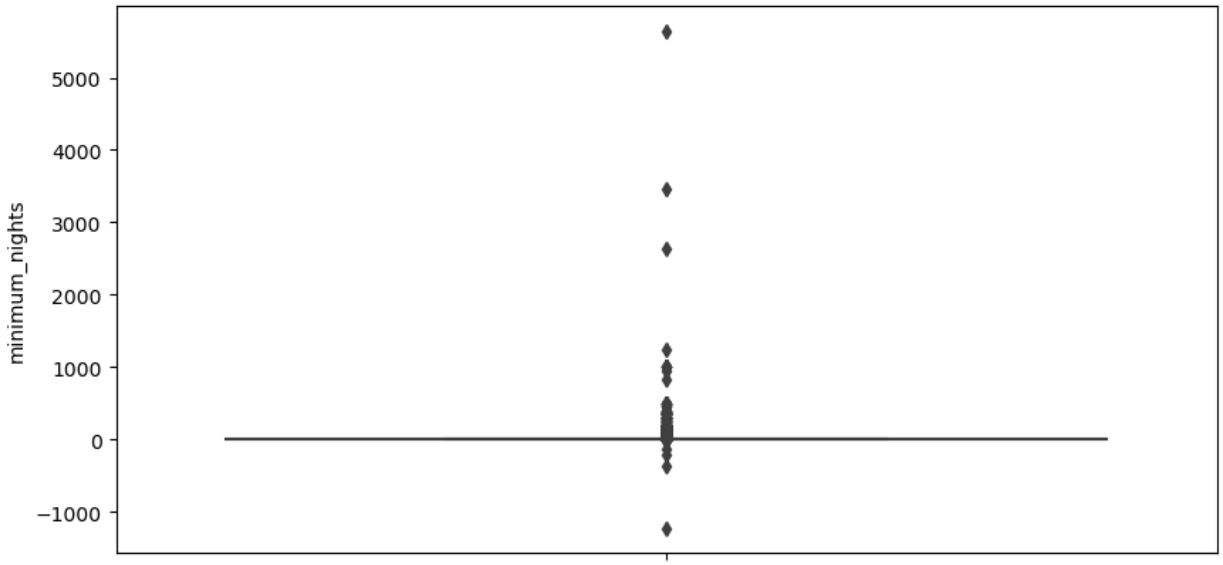
Checking for outlier in rest of the column

In [62]:
```
sns.boxplot(data = df1, y = 'availability_365');
```



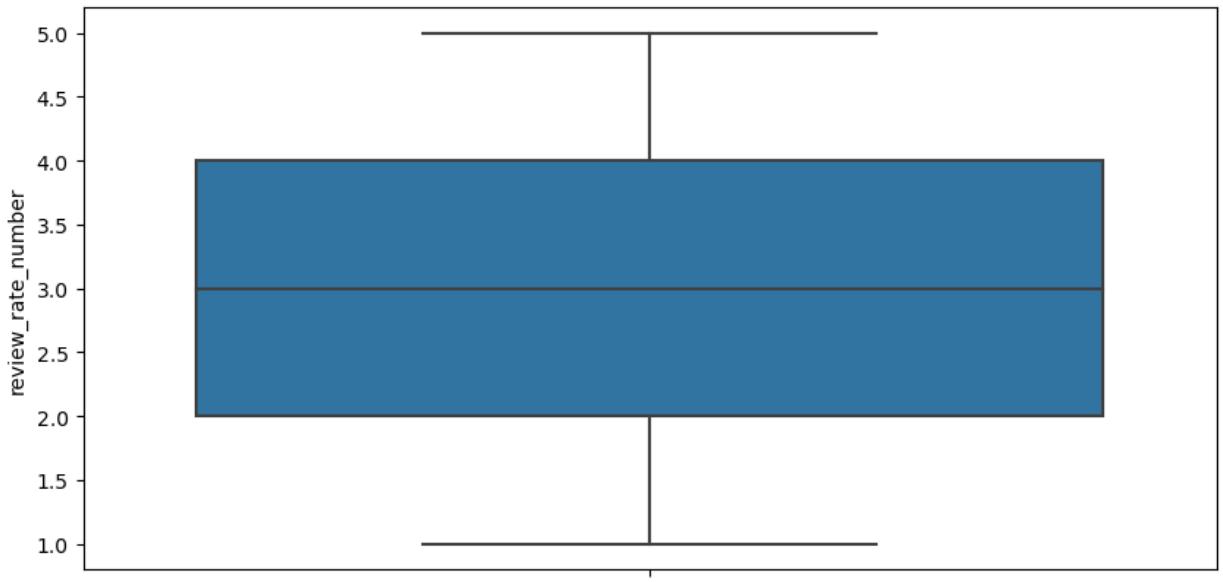
There are some outliers in availability_365 column, so filling the null values with the median of availability_365.

In [63]:
```
sns.boxplot(data = df1, y = 'minimum_nights');
```

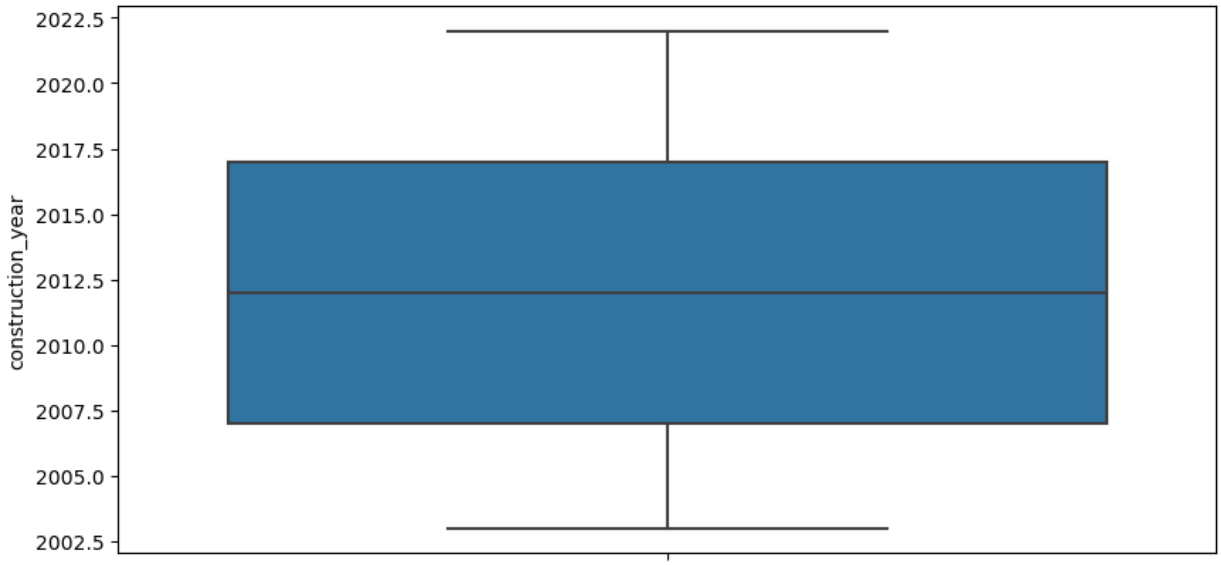


There are outliers in minimum_nights column, so filloing the null values with median of minimum_nights.

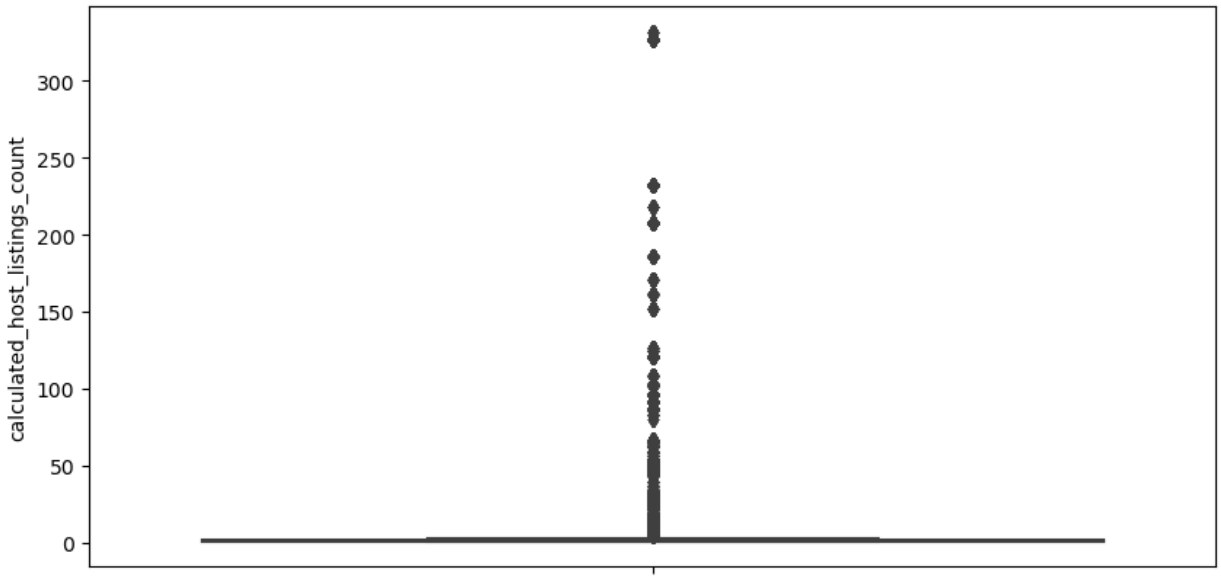
In [64]:
```
sns.boxplot(data = df1, y = 'review_rate_number');
```



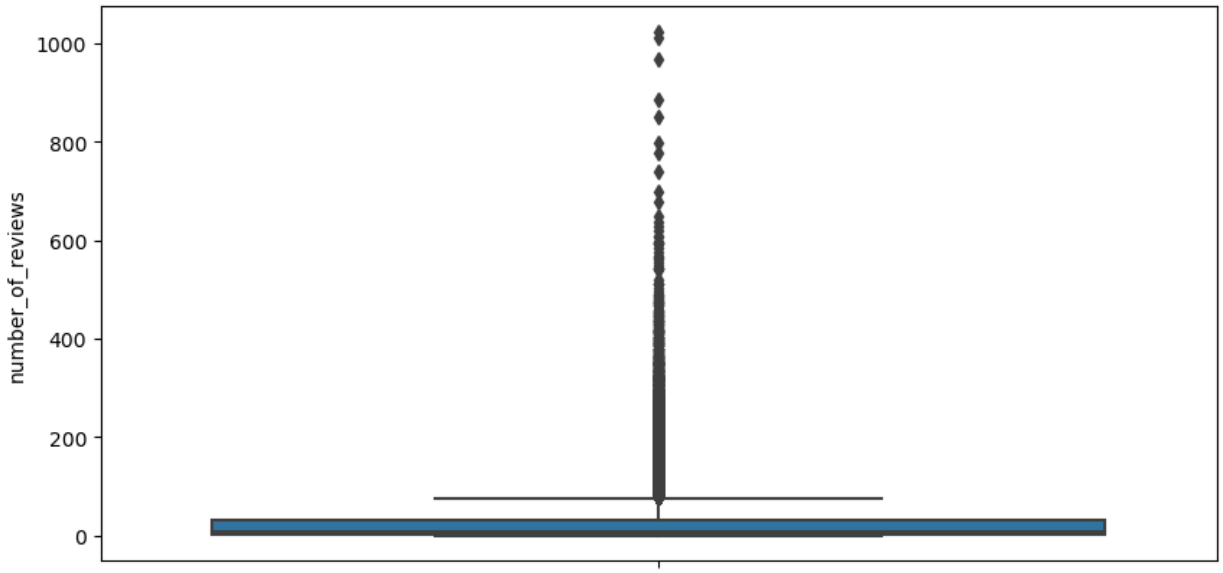
In [65]:
```
sns.boxplot(data = df1, y = 'construction_year');
```



In [66]:
```
sns.boxplot(data = df1, y = 'calculated_host_listings_count');
```



In [67]:
```
sns.boxplot(data = df1, y = 'number_of_reviews');
```



Columns such as availability_365, minimum_nights, calculated_host_listings_count, number_of_reviews have some outliers. To handle the null values in these columns, fill them with the median value of their respective columns. Additionally, fill the null values in other columns such as review_rate_number and construction_year with their respective column medians.

In [68]:
```
df1['availability_365'].fillna(df1['availability_365'].median(), inplace = True)
df1['minimum_nights'].fillna(df1['minimum_nights'].median(), inplace = True)
df1['review_rate_number'].fillna(df1['review_rate_number'].median(), inplace = True)
df1['construction_year'].fillna(df1['construction_year'].median(), inplace = True)
df1['calculated_host_listings_count'].fillna(df1['calculated_host_listings_count'].mec
df1['number_of_reviews'].fillna(df1['number_of_reviews'].median(), inplace = True)
```

In [69]:
```
df1.isnull().sum().sort_values(ascending = False)
```

Out[69]:
```
lat                                11
long                               11
id                                  3
neighbourhood_group                 3
room_type                           3
avg_per_night                       3
minimum_nights                      0
availability_365                    0
calculated_host_listings_count      0
review_rate_number                  0
reviews_per_month                   0
last_review                         0
number_of_reviews                   0
construction_year                   0
service_fee                         0
price                               0
name                                0
cancellation_policy                 0
instant_bookable                    0
host_name                           0
host_identity_verified              0
host_id                             0
house_rules                         0
dtype: int64
```

Dropping rows of neighbourhood_group and long which is having null values

In [70]:
```
df1.dropna(subset=['neighbourhood_group'], inplace=True)
```

In [71]:
```
df1.dropna(subset = 'long', inplace = True)
```

In [72]:
```
df1.isnull().sum().sort_values(ascending = False)
```

Out[72]:
```
id                                 0
price                              0
availability_365                   0
calculated_host_listings_count     0
review_rate_number                 0
reviews_per_month                  0
last_review                        0
number_of_reviews                  0
avg_per_night                      0
minimum_nights                     0
service_fee                        0
construction_year                  0
name                               0
room_type                          0
cancellation_policy                0
instant_bookable                   0
long                               0
lat                                0
neighbourhood_group                0
host_name                          0
host_identity_verified             0
host_id                            0
house_rules                        0
dtype: int64
```

Since all the data is cleaned, this data is now ready for further analysis and to derive useful insights out of it.