Analyzing AirBNB Listings In New York

Importing Libraries & Loading Data

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline
```

In [2]:

```
air = pd.read_csv('Airbnb_Open_Data.csv')
air.head()
```

```
C:\Users\1992729\AppData\Local\Temp\ipykernel_15704\2524591296.py:1: DtypeWarning: Columns (25) have mixed types. Specify dtype option on import or set low_memory=False.
  air = pd.read_csv('Airbnb_Open_Data.csv')
```

Out[2]:

	id	NAME	host id	host_identity_verified	host name	neighbourhood group	neighbourhood	lat	long
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	73.97237
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Manhattan	Midtown	40.75362	73.98377
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Elise	Manhattan	Harlem	40.80902	73.94190
3	1002755	NaN	85098326012	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	73.95976
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Manhattan	East Harlem	40.79851	73.94399

5 rows × 26 columns

Basic Checks

In [3]:

```
air.describe().T
```

Out[3]:

	count	mean	std	min	25%	50%	75%	max

	count	mean	std	min	25%	50%	75%	max
id	102599.0	2.914623e+07	1.625751e+07	1.001254e+06	1.508581e+07	2.913660e+07	4.320120e+07	5.736742e+07
host id	102599.0	4.925411e+10	2.853900e+10	1.236005e+08	2.458333e+10	4.911774e+10	7.399650e+10	9.876313e+10
lat	102591.0	4.072809e+01	5.585652e-02	4.049979e+01	4.068874e+01	4.072229e+01	4.076276e+01	4.091697e+01
long	102591.0	-7.394964e+01	4.952126e-02	-7.424984e+01	-7.398258e+01	-7.395444e+01	-7.393235e+01	-7.370522e+01
Construction year	102385.0	2.012487e+03	5.765556e+00	2.003000e+03	2.007000e+03	2.012000e+03	2.017000e+03	2.022000e+03
minimum nights	102190.0	8.135845e+00	3.055378e+01	-1.223000e+03	2.000000e+00	3.000000e+00	5.000000e+00	5.645000e+03
number of reviews	102416.0	2.748374e+01	4.950895e+01	0.000000e+00	1.000000e+00	7.000000e+00	3.000000e+01	1.024000e+03
reviews per month	86720.0	1.374022e+00	1.746621e+00	1.000000e-02	2.200000e-01	7.400000e-01	2.000000e+00	9.000000e+01
review rate number	102273.0	3.279106e+00	1.284657e+00	1.000000e+00	2.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
calculated host listings count	102280.0	7.936605e+00	3.221878e+01	1.000000e+00	1.000000e+00	1.000000e+00	2.000000e+00	3.320000e+02
availability 365	102151.0	1.411333e+02	1.354350e+02	-1.000000e+01	3.000000e+00	9.600000e+01	2.690000e+02	3.677000e+03

In [4]:

```
air.dtypes
```

Out[4]:

```
id                                int64
NAME                             object
host id                           int64
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
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 [5]:

```
air.isnull().sum()
```

Out[5]:

```
id                                0
NAME                            250
host id                           0
host_identity_verified          289
host name                       406
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                    409
number of reviews                 183
last review                     15893
reviews per month               15879
review rate number                326
calculated host listings count    319
availability 365                  448
house_rules                     52131
license                        102597
dtype: int64
```

In [6]:

```
print("unique countries: ", air['country'].unique())
print("unique countries code: ", air['country code'].unique())
```

```
unique countries:  ['United States' nan]
unique countries code:  ['US' nan]
```

Data Cleaning

Dropping Unnecessary Columns

In [7]:

```
air.drop(labels=['host name', 'id', 'license', 'country', 'country code', 'last review',
'reviews per month'], axis=1, inplace=True)
air.head()
```

Out[7]:

	NAME	host id	host_identity_verified	neighbourhood group	neighbourhood	lat	long	instant_bookable
0	Clean & quiet apt home by the park	80014485718	unconfirmed	Brooklyn	Kensington	40.64749	-73.97237	False
1	Skylit Midtown Castle	52335172823	verified	Manhattan	Midtown	40.75362	-73.98377	False
2	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Manhattan	Harlem	40.80902	-73.94190	True
3	NaN	85098326012	unconfirmed	Brooklyn	Clinton Hill	40.68514	-73.95976	True
4	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Manhattan	East Harlem	40.79851	-73.94399	False

Price column contains object values, and we need them in floats to do better analysis. Removing dollar sign and converting type.

In [8]:

```
def cleanedPrice(price):
    #print("price type is : ", type(price))
    if(len(price.split("$")) > 1):

        return "".join(price.split("$")[1].replace(",",""))
    else:
        return "".join(price.replace(",",""))
```

In [9]:

```
air['price'] = air['price'].astype('str')
air['price'] = air['price'].apply(cleanedPrice)

air['price'] = air['price'].astype('float')
air['price'].head()
```

Out[9]:

```
0    966.0
1    142.0
2    620.0
3    368.0
4    204.0
Name: price, dtype: float64
```

service fee column contains object data type, and it starts with $ sign, we need to clean it for numerical analysis.

In [10]:

```
air['service fee'] = air['service fee'].astype('str')
air['service fee'] = air['service fee'].apply(cleanedPrice)

air['service fee'] = air['service fee'].astype('float')
air['service fee'].head()
```

Out[10]:

```
0    193.0
1     28.0
2    124.0
3     74.0
4     41.0
Name: service fee, dtype: float64
```

Replacing null values in review rate number column with mean values

In [11]:

```
air['review rate number'] = air['review rate number'].fillna(air['review rate number'].me
dian())
air['review rate number'].isnull().sum()
```

Out[11]:

```
0
```

Replacing null values in Construction year column with median values

In [12]:

```
air['Construction year'] = air['Construction year'].fillna(air['Construction year'].media
```

```
n())
air['Construction year'].isnull().sum()
```

Out[12]:

```
0
```

Replacing null values in price column with mean values

In [13]:

```
air['price'] = air['price'].fillna(air['price'].median())
air['price'].isnull().sum()
```

Out[13]:

```
0
```

Replacing null values in service fee with Median values

In [14]:

```
air['service fee'] = air['service fee'].fillna(air['service fee'].median())
air['service fee'].isnull().sum()
```

Out[14]:

```
0
```

Replacing null values in minimum nights with Median values

In [15]:

```
air['minimum nights'] = air['minimum nights'].fillna(air['minimum nights'].median())
air['minimum nights'].isnull().sum()
```

Out[15]:

```
0
```

Replacing null values in number of reviews with median values

In [16]:

```
air['number of reviews']=air['number of reviews'].fillna(air['number of reviews'].median(
))
air['number of reviews'].isnull().sum()
```

Out[16]:

```
0
```

In [17]:

```
air['neighbourhood group'].value_counts()
#air['neighbourhood group'] = air['neighbourhood group'].apply(lambda nei: if nei=='brook
ln': return 'Brooklyn')
```

Out[17]:

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

There are some spelling errors in the neighborhood group column, so fixing them is necessary to get better

picture on neighborhood groups.

In [18]:

```
def cleanNeigh(nei):
    if(nei=='brookln'):
        return 'Brooklyn'
    elif(nei == 'manhatan'):
        return 'Manhattan'
    else:
        return nei
air['neighbourhood group'] = air['neighbourhood group'].apply(cleanNeigh)
```

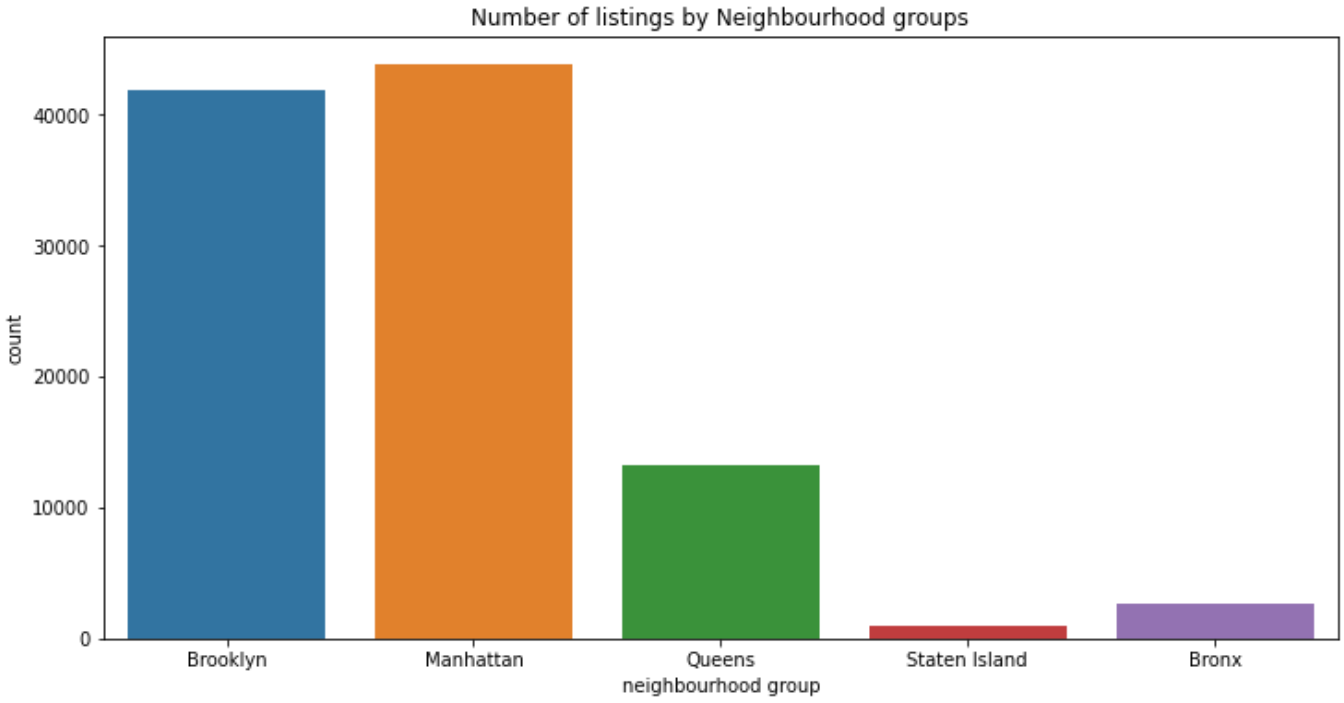
Which Neighborhood Group Has The Most Listings?

In [19]:

```
plt.figure(figsize=(12,6))
sns.countplot(data=air, x='neighbourhood group')
plt.title("Number of listings by Neighbourhood groups")
```

Out[19]:

```
Text(0.5, 1.0, 'Number of listings by Neighbourhood groups')
```



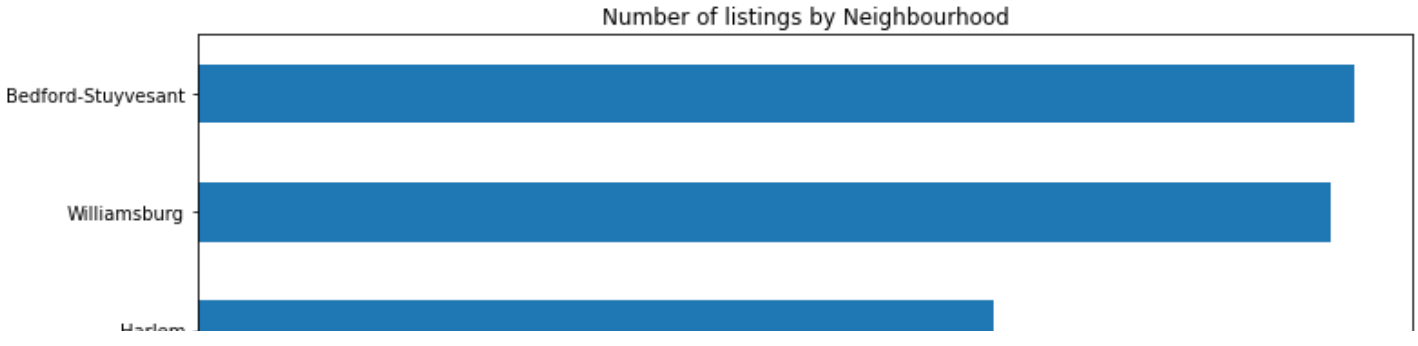
Which neighborhood has the most listings?

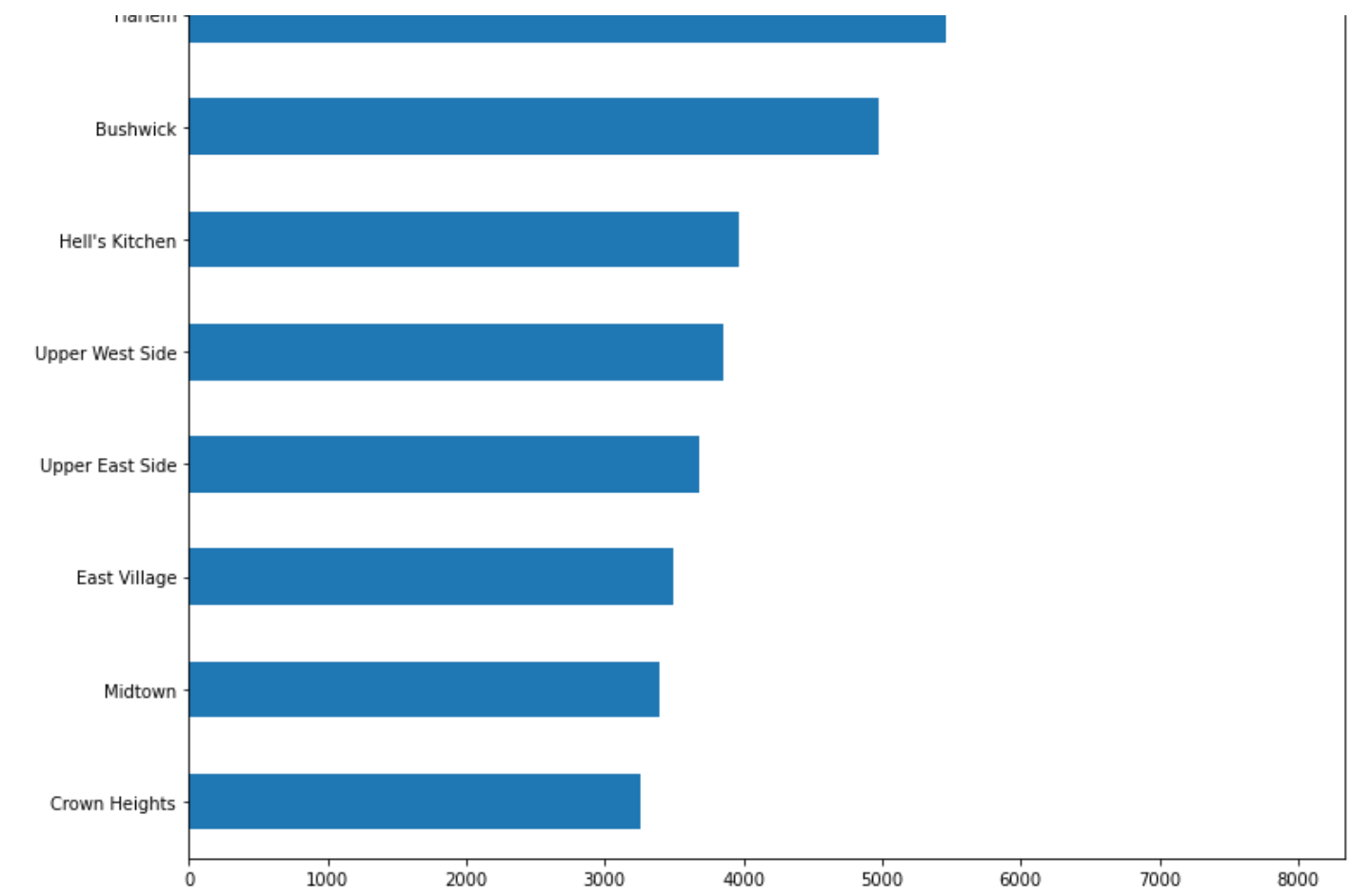
In [20]:

```
plt.figure(figsize=(12,12))
air['neighbourhood'].value_counts()[:10].sort_values(ascending=True).plot(kind="barh")
plt.title("Number of listings by Neighbourhood")
```

Out[20]:

```
Text(0.5, 1.0, 'Number of listings by Neighbourhood')
```





Distribution of Property's Constructed Year

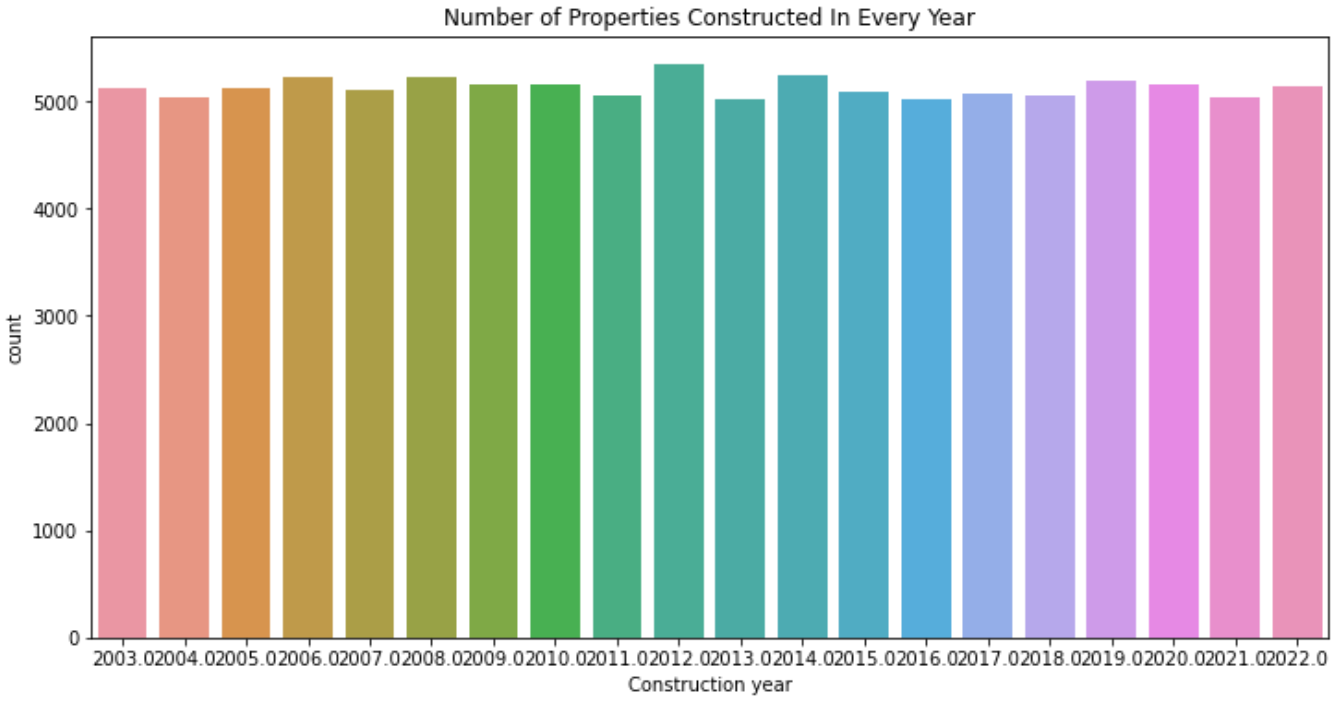
In [21]:

```
plt.figure(figsize=(12,6))
sns.countplot(air['Construction year'])
plt.title("Number of Properties Constructed In Every Year")
```

```
C:\Users\1992729\AppData\Roaming\Python\Python39\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without an expli
cit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[21]:

```
Text(0.5, 1.0, 'Number of Properties Constructed In Every Year')
```



Average price of houses in neighbourhood groups

In [22]:

```
groupedDf = air.groupby(by='neighbourhood group').mean().reset_index()
groupedDf.head()
```

Out[22]:

	neighbourhood group	host id	lat	long	Construction year	price	service fee	minimum nights	number of reviews	review rate number
0	Bronx	4.883886e+10	40.849289	-73.883191	2012.476401	627.756637	125.449484	5.129056	31.653392	3.331121
1	Brooklyn	4.915650e+10	40.683825	-73.950496	2012.514399	626.555386	125.249098	7.284970	28.478336	3.258347
2	Manhattan	4.944571e+10	40.765288	-73.974404	2012.476697	622.440436	124.478478	9.635284	24.097413	3.275683
3	Queens	4.896106e+10	40.728520	-73.867637	2012.483907	630.192206	126.046959	6.481495	33.628477	3.329615
4	Staten Island	4.984976e+10	40.611668	-74.105042	2011.737173	624.489005	124.875393	5.900524	35.704712	3.403141

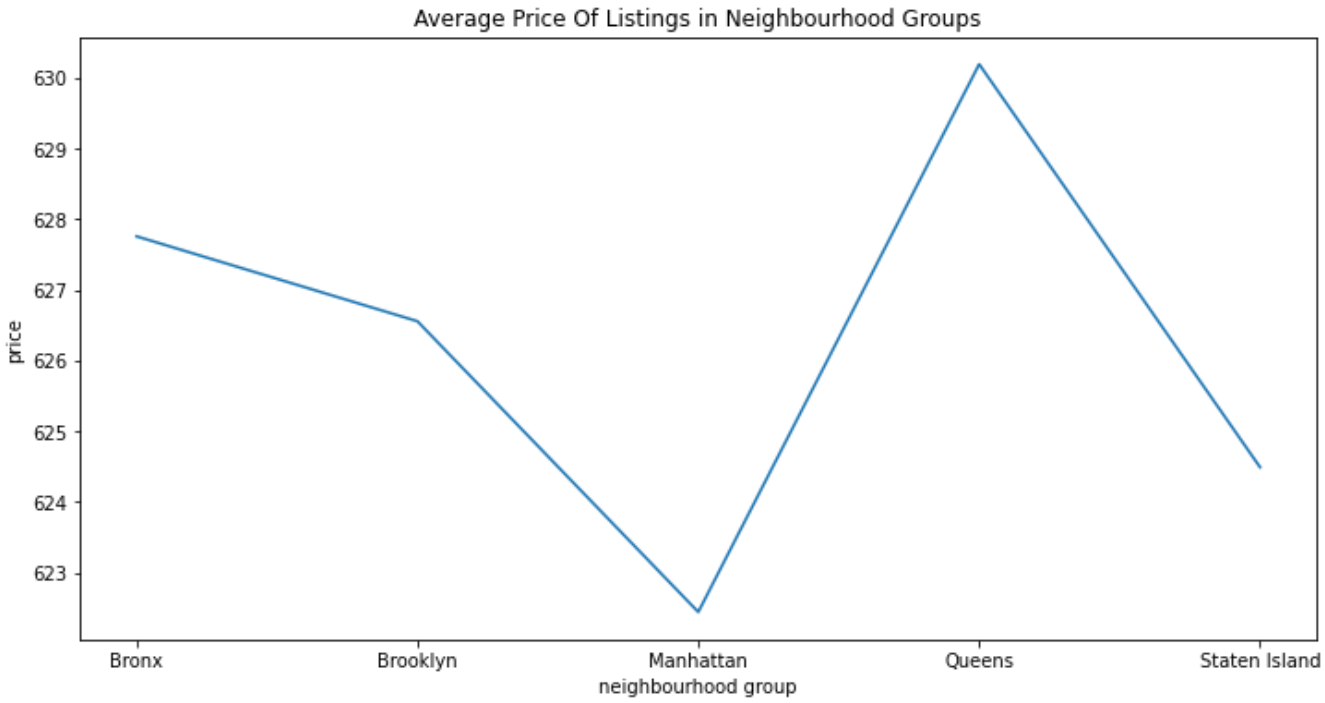
In [23]:

```
plt.figure(figsize=(12,6))
newdf = pd.DataFrame(groupedDf[['neighbourhood group', 'price']])
sns.lineplot(x='neighbourhood group', y ='price', data=newdf)

plt.title('Average Price Of Listings in Neighbourhood Groups')
```

Out[23]:

```
Text(0.5, 1.0, 'Average Price Of Listings in Neighbourhood Groups')
```

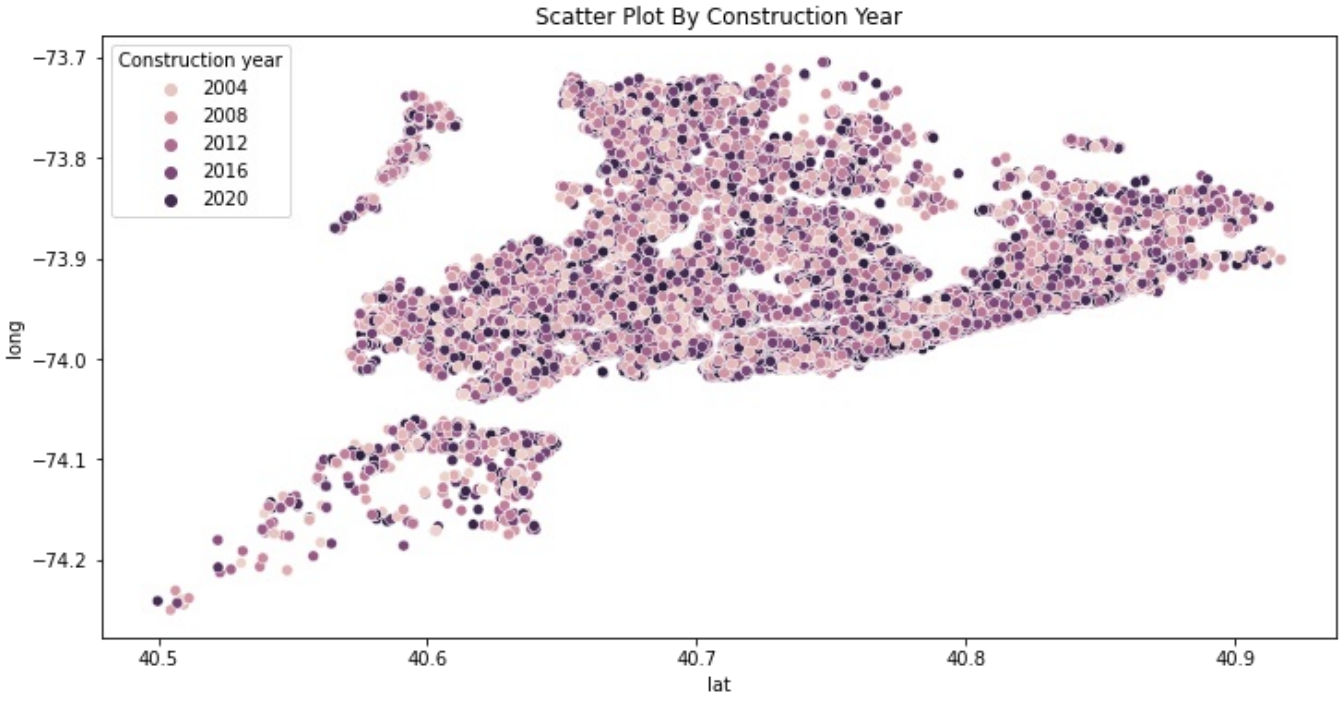


In [24]:

```
plt.figure(figsize=(12,6))
sns.scatterplot(data=air, x='lat', y='long', hue='Construction year')
plt.title("Scatter Plot By Construction Year")
```

Out[24]:

```
Text(0.5, 1.0, 'Scatter Plot By Construction Year')
```

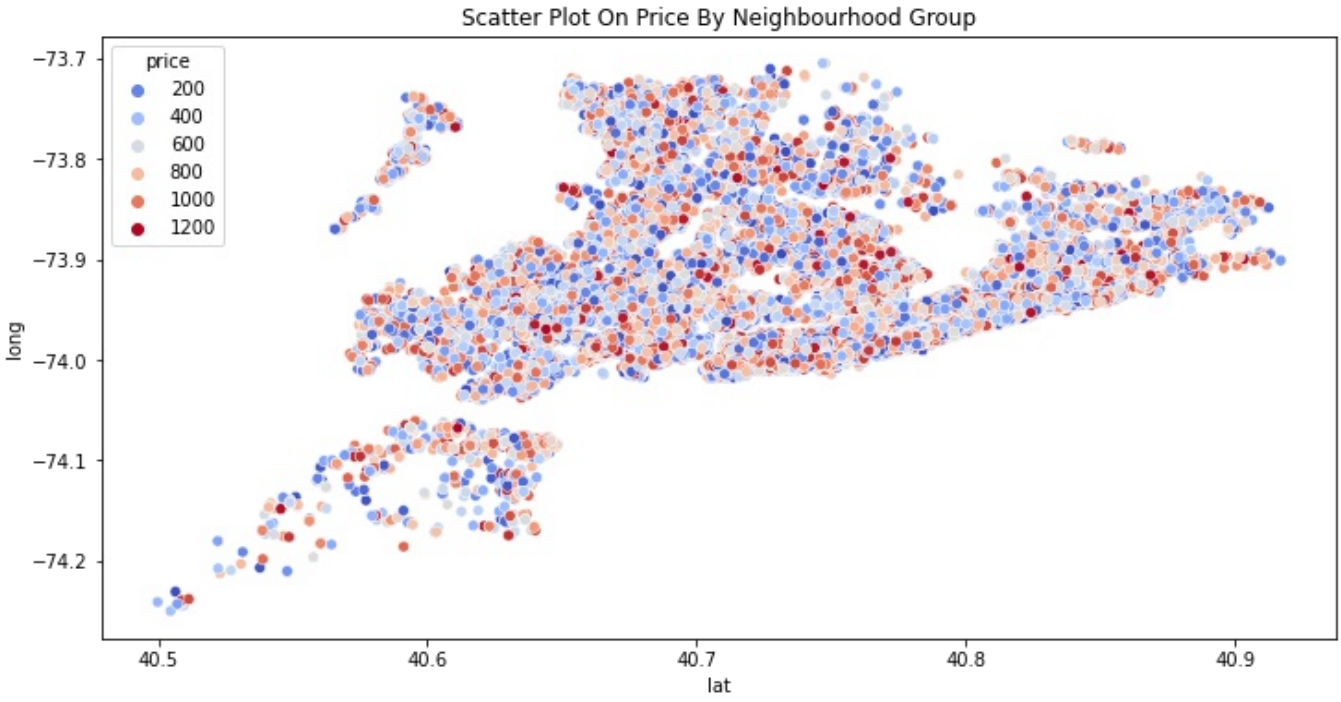


In [26]:

```
plt.figure(figsize=(12,6))
sns.scatterplot(data=air, x='lat', y='long', hue='price', palette='coolwarm')
plt.title("Scatter Plot On Price By Neighbourhood Group")
```

Out[26]:

```
Text(0.5, 1.0, 'Scatter Plot On Price By Neighbourhood Group')
```



Cleaning host identity column

In [31]:

```
air['host_identity_verified'].fillna('unconfirmed', inplace=True)
air['host_identity_verified'].unique()
```

Out[31]:

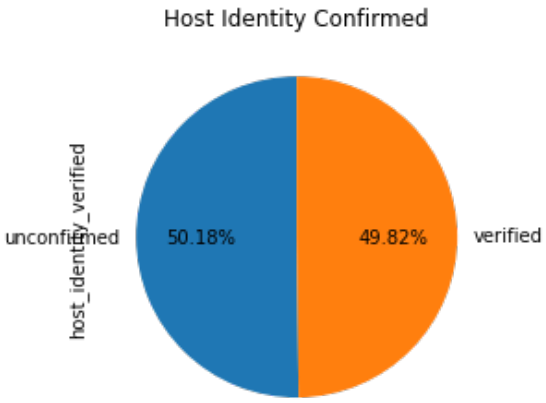
```
array(['unconfirmed', 'verified'], dtype=object)
```

In [41]:

```
air['host_identity_verified'].value_counts().plot(kind='pie', autopct='%.2f%%', startangle=90)
#plt.legend()
plt.title('Host Identity Confirmed')
```

Out[41]:

```
Text(0.5, 1.0, 'Host Identity Confirmed')
```

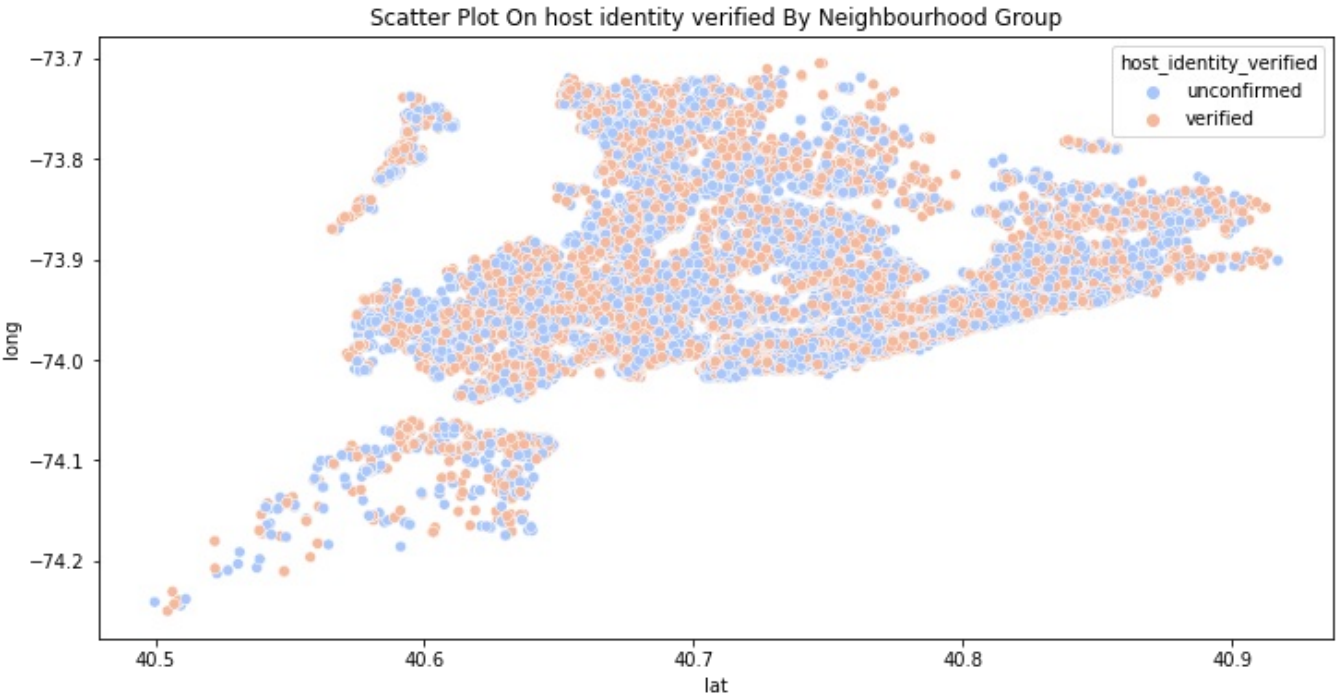


In [42]:

```
plt.figure(figsize=(12,6))
sns.scatterplot(data=air, x='lat', y='long', hue='host_identity_verified', palette='coolwarm')
plt.title("Scatter Plot On host identity verified By Neighbourhood Group")
```

Out[42]:

```
Text(0.5, 1.0, 'Scatter Plot On host identity verified By Neighbourhood Group')
```

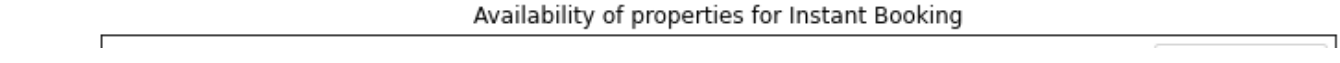


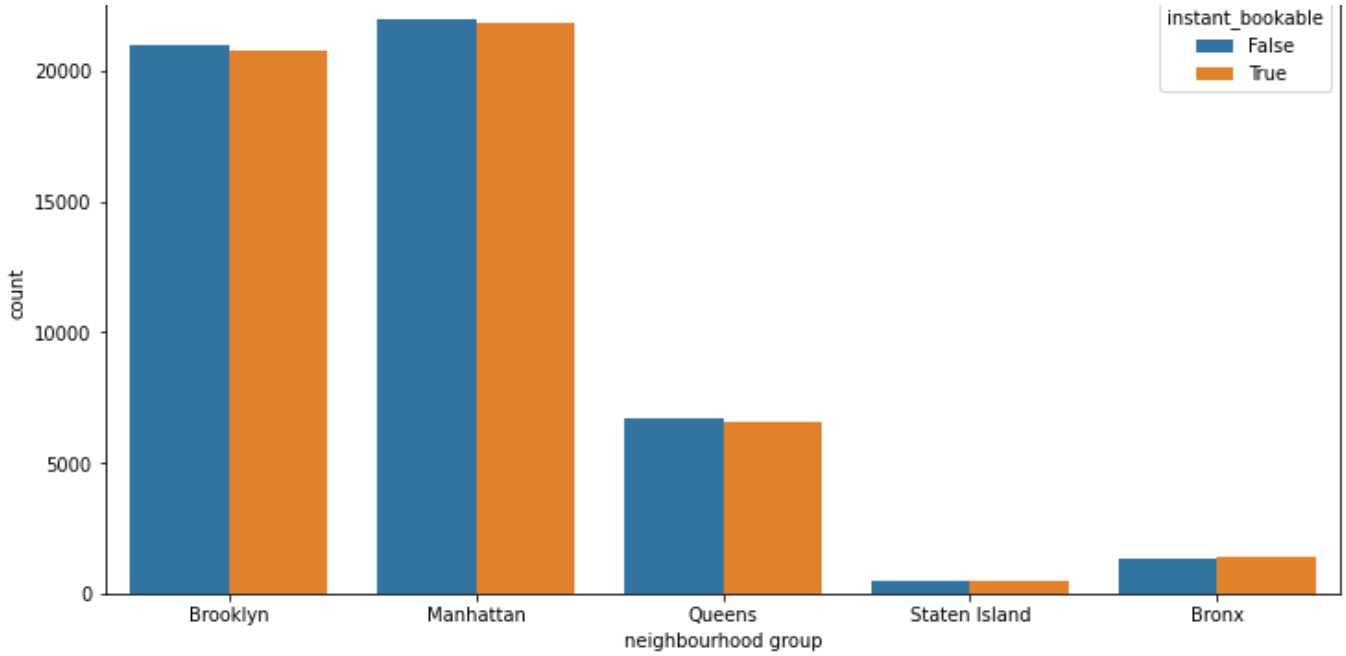
In [45]:

```
plt.figure(figsize=(12,6))
sns.countplot(data=air, hue='instant_bookable', x='neighbourhood group')
plt.title('Availability of properties for Instant Booking')
```

Out[45]:

```
Text(0.5, 1.0, 'Availability of properties for Instant Booking')
```





In [49]:

```
air['cancellation_policy'].fillna('No Policy', inplace= True)

air.head()
```

Out[49]:

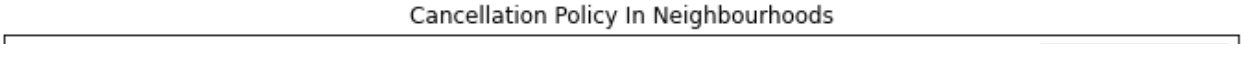
	NAME	host id	host_identity_verified	neighbourhood group	neighbourhood	lat	long	instant_bookable
0	Clean & quiet apt home by the park	80014485718	unconfirmed	Brooklyn	Kensington	40.64749	-73.97237	False
1	Skylit Midtown Castle	52335172823	verified	Manhattan	Midtown	40.75362	-73.98377	False
2	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	unconfirmed	Manhattan	Harlem	40.80902	-73.94190	True
3	NaN	85098326012	unconfirmed	Brooklyn	Clinton Hill	40.68514	-73.95976	True
4	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Manhattan	East Harlem	40.79851	-73.94399	False

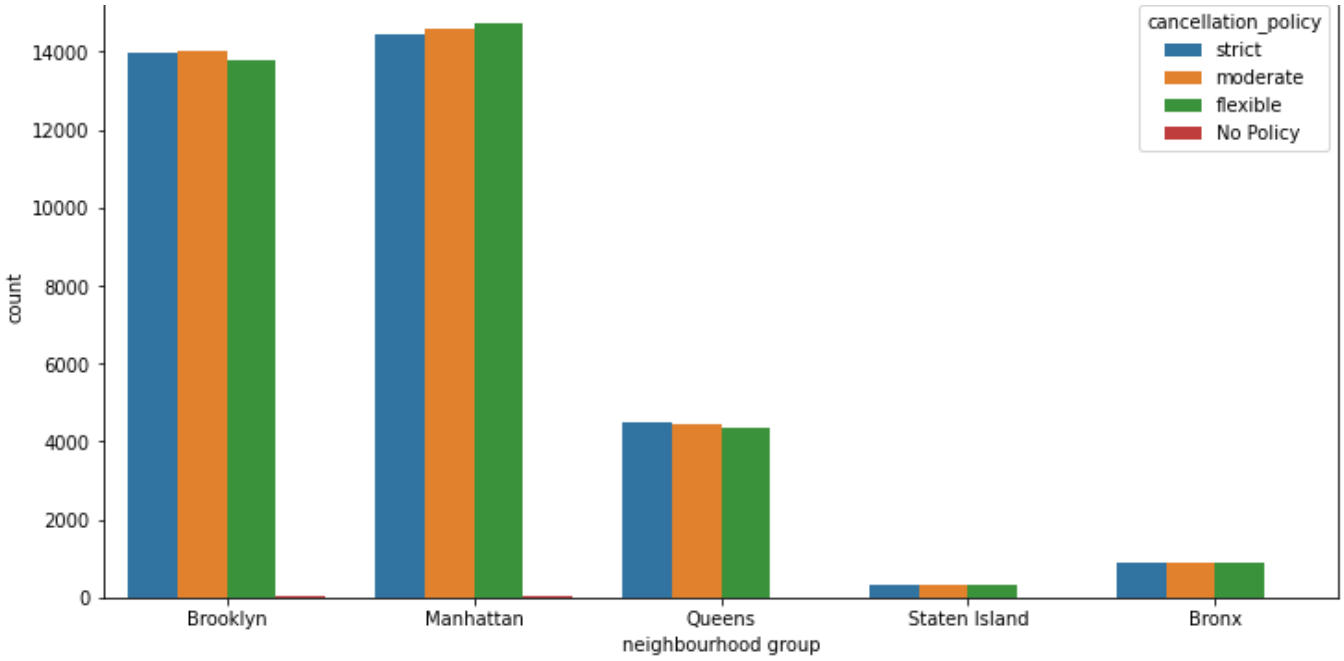
In [50]:

```
plt.figure(figsize=(12,6))
sns.countplot(data=air, hue='cancellation_policy', x='neighbourhood group')
plt.title('Cancellation Policy In Neighbourhoods')
```

Out[50]:

```
Text(0.5, 1.0, 'Cancellation Policy In Neighbourhoods')
```



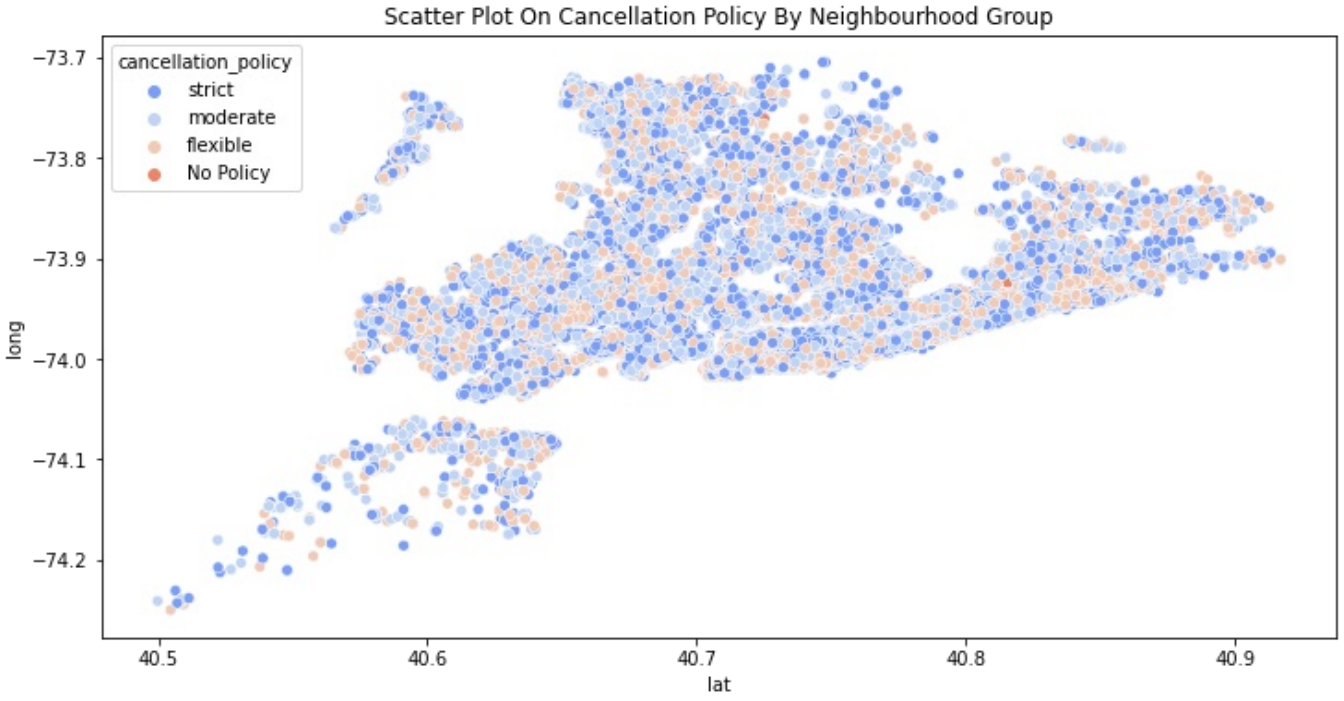


In [52]:

```
plt.figure(figsize=(12,6))
sns.scatterplot(data=air, x='lat', y='long', hue='cancellation_policy', palette='coolwarm')
plt.title("Scatter Plot On Cancellation Policy By Neighbourhood Group")
```

Out[52]:

```
Text(0.5, 1.0, 'Scatter Plot On Cancellation Policy By Neighbourhood Group')
```



How many number of listings are available under each room type?

In [56]:

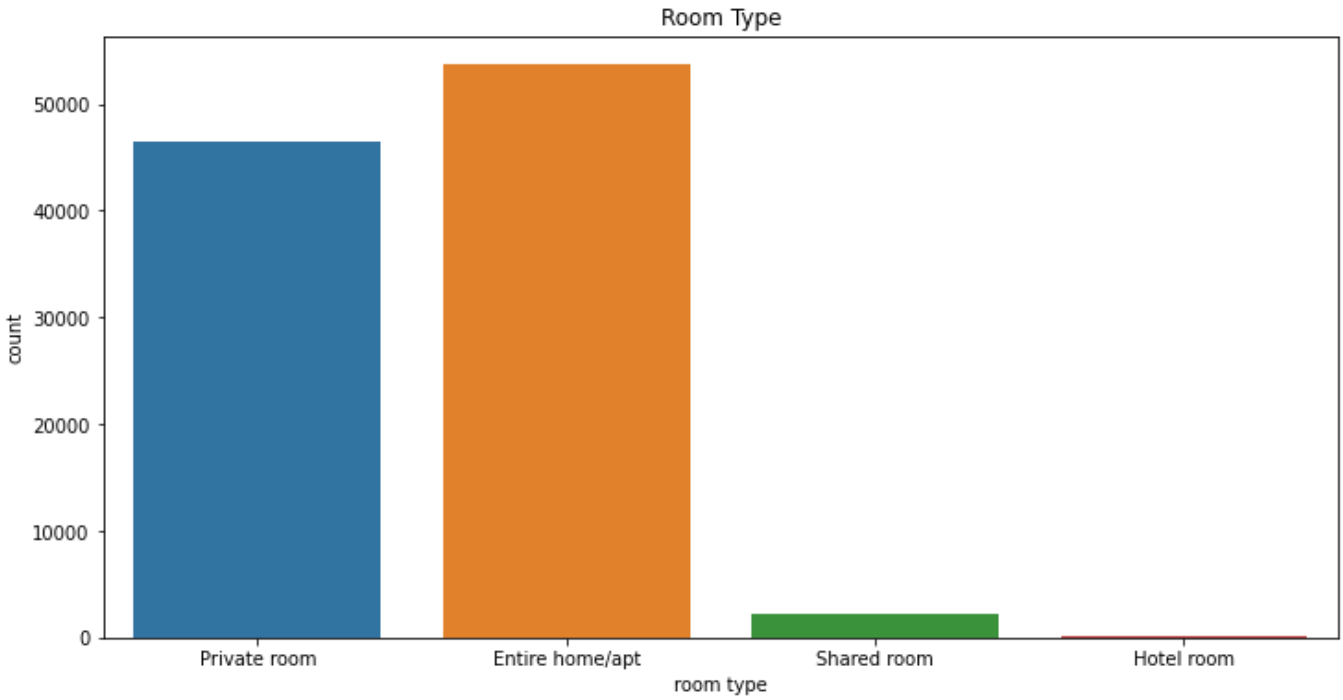
```
plt.figure(figsize=(12,6))
sns.countplot(air['room type'])
plt.title("Room Type")
```

```
C:\Users\1992729\AppData\Roaming\Python\Python39\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[56]:

Out[56]:

```
Text(0.5, 1.0, 'Room Type')
```



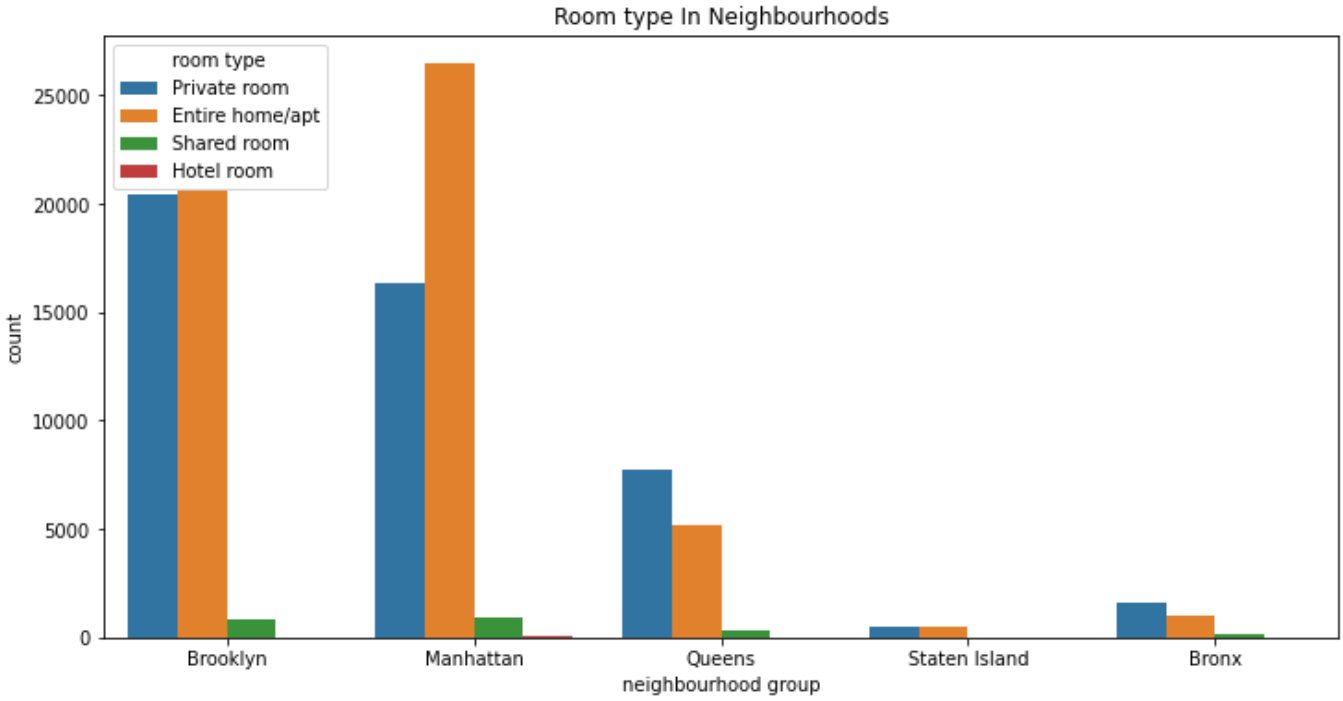
How are neighbourhood groups composed, based on room type?

In [58]:

```
plt.figure(figsize=(12,6))
sns.countplot(data=air, hue='room type', x='neighbourhood group')
plt.title('Room type In Neighbourhoods')
```

Out[58]:

```
Text(0.5, 1.0, 'Room type In Neighbourhoods')
```



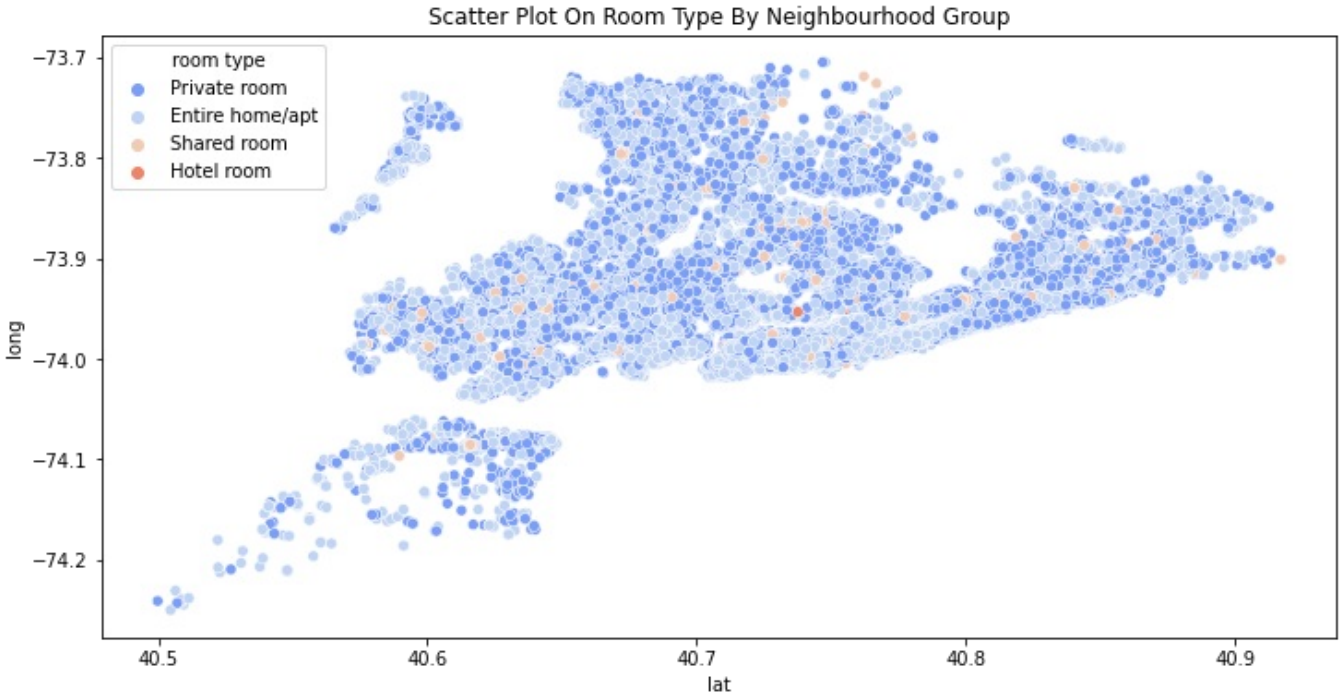
How are room types scattered across New York?

In [54]:

```
plt.figure(figsize=(12,6))
sns.scatterplot(data=air, x='lat', y='long', hue='room type', palette='coolwarm')
plt.title("Scatter Plot On Room Type By Neighbourhood Group")
```

Out[54]:

```
Text(0.5, 1.0, 'Scatter Plot On Room Type By Neighbourhood Group')
```



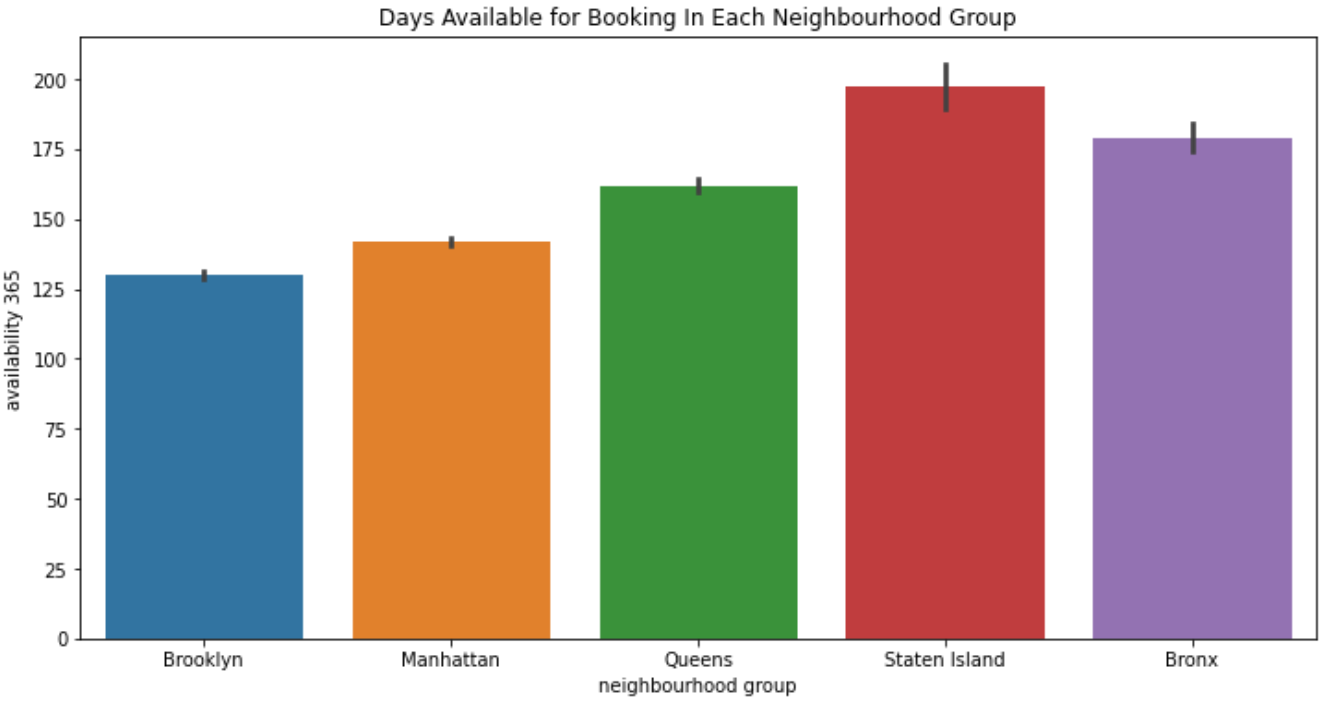
Which neighbourhood group is the least busy?

In [73]:

```
plt.figure(figsize=(12,6))
sns.barplot(x='neighbourhood group', y='availability 365', data=air)
plt.title('Days Available for Booking In Each Neighbourhood Group')
```

Out[73]:

```
Text(0.5, 1.0, 'Days Available for Booking In Each Neighbourhood Group')
```



Which neighbourhood is least busy?

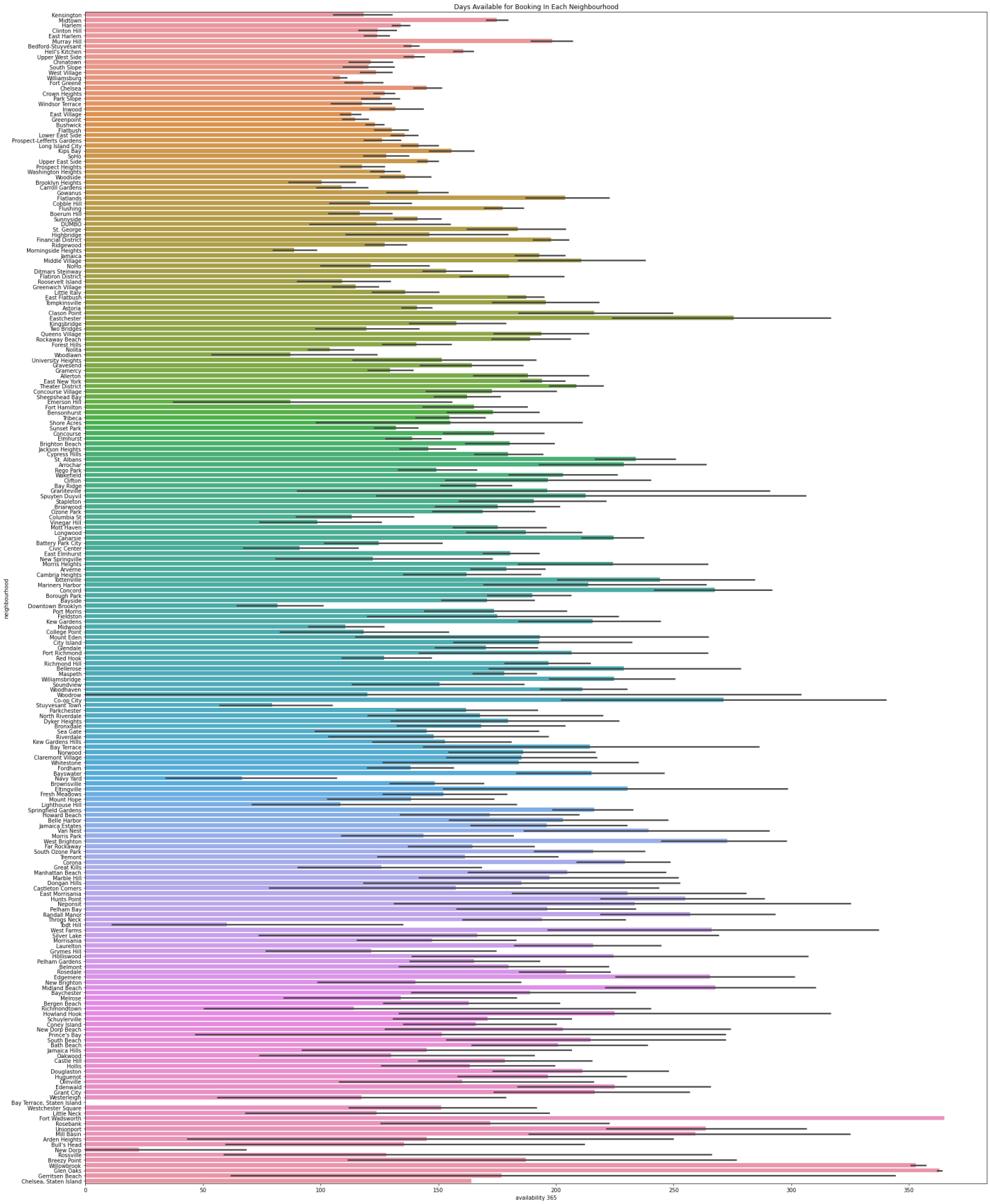
In [80]:

```
plt.figure(figsize=(30,40))
sns.barplot(y='neighbourhood', x='availability 365', data=air, orient='h')
plt.title('Days Available for Booking In Each Neighbourhood')
```

```
#plt.xlabel(rot=90)
```

Out[80]:

```
Text(0.5, 1.0, 'Days Available for Booking In Each Neighbourhood')
```



In [105]:

```
air.head()
```

Out[105]:

	NAME	host id	host_identity_verified	neighbourhood group	neighbourhood	lat	long	instant_bookable

	NAME	host id	host_identity_verified	neighbourhood group	neighbourhood	lat	long	instant_bookable
0	Clean & quiet apt home by the park	80014485718	unconfirmed	Brooklyn	Kensington	40.64749	-73.97237	False
1	Skylit Midtown Castle	52335172823	verified	Manhattan	Midtown	40.75362	-73.98377	False
2	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	unconfirmed	Manhattan	Harlem	40.80902	-73.94190	True
3	NaN	85098326012	unconfirmed	Brooklyn	Clinton Hill	40.68514	-73.95976	True
4	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Manhattan	East Harlem	40.79851	-73.94399	False