Data Cleaning in Python

In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('D:\Airbnb_Open_Data.csv')
```

```
C:\Users\Naushad Saifi\AppData\Local\Temp\ipykernel_16716\3613834805.py:1: Dtyp
eWarning: Columns (25) have mixed types. Specify dtype option on import or set
low_memory=False.
  df = pd.read_csv('D:\Airbnb_Open_Data.csv')
```

In [4]:

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

In [5]:

```
df.head()
```

Out[5]:

	id	NAME	host id	host_identity	host name	neighbourhood group	neighbourho
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Brooklyn	Kensing
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Manhattan	Midtc
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Elise	Manhattan	Harl
3	1002755	NaN	85098326012	unconfirmed	Garry	Brooklyn	Clinton
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Manhattan	East Harl

5 rows × 26 columns

In [6]:

```
df.columns
```

```
Out[6]: Index(['id', 'NAME', 'host id', 'host_identity', 'host name',
               'neighbourhood group', 'neighbourhood', 'lat', 'long', 'country',
               'country code', 'instant_bookable', 'cancellation_policy', 'room type',
               'Construction year', 'price', 'service fee', 'minimum nights',
               'number of reviews', 'last review', 'reviews per month',
               'review rate number', 'calculated host listings count',
               'availability 365', 'house_rules', 'license'],
             dtype='object')
```

```
In [11]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102599 entries, 0 to 102598
Data columns (total 26 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              102599 non-null  int64
 1   NAME                            102349 non-null  object
 2   host id                         102599 non-null  int64
 3   host_identity                   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                           102352 non-null  object
 16  service fee                     102326 non-null  object
 17  minimum nights                  102190 non-null  float64
 18  number of reviews               102416 non-null  float64
 19  last review                     86706 non-null   datetime64[ns]
 20  reviews per month               86720 non-null   float64
 21  review rate number              102273 non-null  float64
 22  calculated host listings count  102280 non-null  float64
 23  availability 365                102151 non-null  float64
 24  house_rules                     50468 non-null   object
 25  license                         2 non-null       object
dtypes: datetime64[ns](1), float64(9), int64(2), object(14)
memory usage: 20.4+ MB
```

Checking Missing Values

```
In [20]: print(df.isnull().sum())
```

```
id                                0
NAME                              0
host id                           0
host_identity                   276
host name                         0
neighbourhood group              26
neighbourhood                    16
lat                               8
long                              8
country                         526
country code                    122
instant_bookable                 96
cancellation_policy              70
room type                         0
Construction year               200
price                           239
service fee                     268
minimum nights                  403
number of reviews               182
last review                       0
reviews per month                 0
review rate number              314
calculated host listings count  318
availability 365                420
dtype: int64
```

Handling Missing Values

```
In [10]: df['last review'] = pd.to_datetime(df['last review'], errors = 'coerce')
```

```
In [14]: df.fillna({'reviews per month' : 0, 'last review' : df['last review'].min()}, in
```

```
C:\Users\Naushad Saifi\AppData\Local\Temp\ipykernel_16716\1608659002.py:1: Futu
reWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set t
he values inplace instead of always setting a new array. To retain the old beha
vior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `
df.isetitem(i, newvals)`
  df.fillna({'reviews per month' : 0, 'last review' : df['last review'].min()},
inplace = True)
```

```
In [16]: df.dropna(subset = ['NAME','host name'], inplace = True)
```

```
In [19]: df = df.drop(columns = ['license', 'house_rules'], errors = 'ignore')
```

```
In [21]: df.head()
```

Out[21]:

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

5 rows × 24 columns

In [25]:

```
#remove $signs and convert to float
df['price'] = df['price'].replace('[\$,\s]','', regex=True).astype(float)
df['service fee'] = df['service fee'].replace('[\$,\s]','', regex=True).astype(f

#Explanation:
#[\$,\s]: This regex pattern matches:
#\$: The dollar sign (escaped because $ has a special meaning in regex).
#,: The comma.
#\s: Any whitespace (spaces, tabs, etc.).
#replace('[\$,\s]','', regex=True): Removes all instances of dollar signs, comma
#astype(float): Converts the cleaned string to a float.
```

In [24]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101949 entries, 0 to 102598
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              101949 non-null  int64
 1   NAME                            101949 non-null  object
 2   host id                         101949 non-null  int64
 3   host_identity                   101673 non-null  object
 4   host name                       101949 non-null  object
 5   neighbourhood group             101923 non-null  object
 6   neighbourhood                   101933 non-null  object
 7   lat                             101941 non-null  float64
 8   long                            101941 non-null  float64
 9   country                         101423 non-null  object
 10  country code                    101827 non-null  object
 11  instant_bookable                101853 non-null  object
 12  cancellation_policy             101879 non-null  object
 13  room type                       101949 non-null  object
 14  Construction year               101749 non-null  float64
 15  price                           101710 non-null  float64
 16  service fee                     101681 non-null  float64
 17  minimum nights                  101546 non-null  float64
 18  number of reviews               101767 non-null  float64
 19  last review                     101949 non-null  datetime64[ns]
 20  reviews per month               101949 non-null  float64
 21  review rate number              101635 non-null  float64
 22  calculated host listings count  101631 non-null  float64
 23  availability 365                101529 non-null  float64
dtypes: datetime64[ns](1), float64(11), int64(2), object(10)
memory usage: 19.4+ MB
```

```
In [26]: df.head()
```

Out[26]:

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

5 rows × 24 columns

Remove Duplicates

In [27]:

```
df.drop_duplicates(inplace=True)
```

In [28]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101410 entries, 0 to 102057
Data columns (total 24 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              101410 non-null  int64
 1   NAME                            101410 non-null  object
 2   host id                         101410 non-null  int64
 3   host_identity                   101134 non-null  object
 4   host name                       101410 non-null  object
 5   neighbourhood group             101384 non-null  object
 6   neighbourhood                   101394 non-null  object
 7   lat                             101402 non-null  float64
 8   long                            101402 non-null  float64
 9   country                         100884 non-null  object
 10  country code                    101288 non-null  object
 11  instant_bookable                101314 non-null  object
 12  cancellation_policy             101340 non-null  object
 13  room type                       101410 non-null  object
 14  Construction year               101210 non-null  float64
 15  price                           101171 non-null  float64
 16  service fee                     101142 non-null  float64
 17  minimum nights                  101016 non-null  float64
 18  number of reviews               101228 non-null  float64
 19  last review                     101410 non-null  datetime64[ns]
 20  reviews per month               101410 non-null  float64
 21  review rate number              101103 non-null  float64
 22  calculated host listings count  101092 non-null  float64
 23  availability 365                100990 non-null  float64
dtypes: datetime64[ns](1), float64(11), int64(2), object(10)
memory usage: 19.3+ MB
```

Descriptive Statistics

In [29]:

```
df.describe()
```

Out[29]:

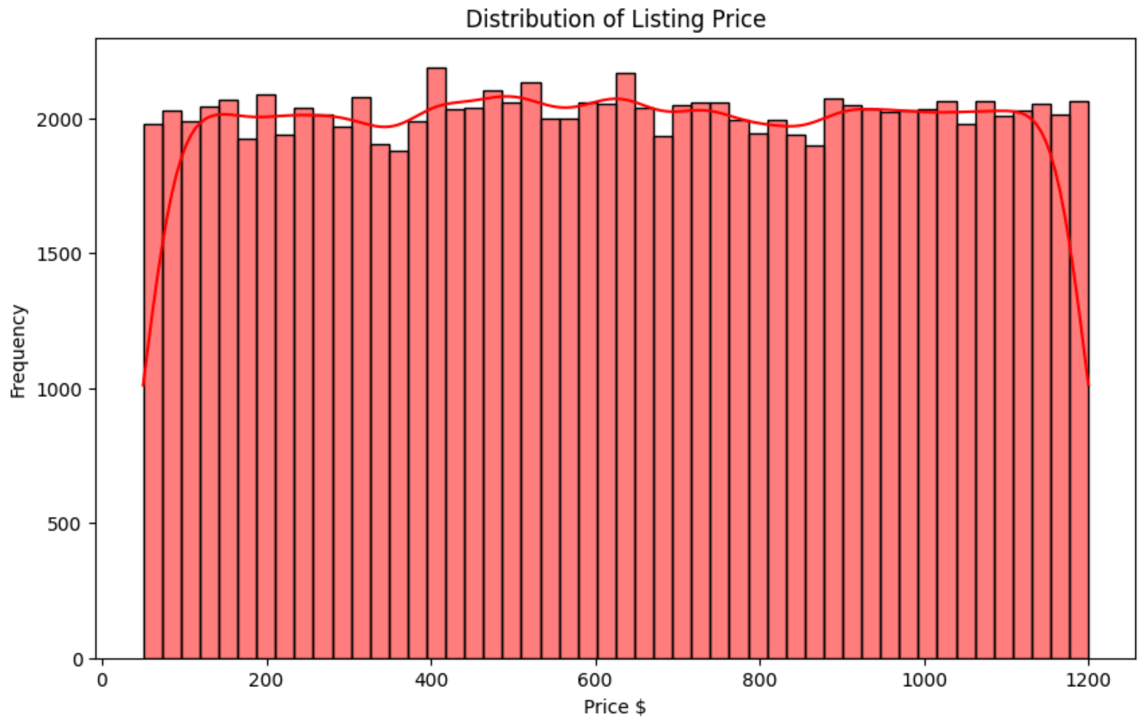
	id	host id	lat	long	Construction year	price
count	1.014100e+05	1.014100e+05	101402.000000	101402.000000	101210.0	101171.000000
mean	2.920959e+07	4.926155e+10	40.728082	-73.949663	1905.0	625.381008
std	1.626820e+07	2.853703e+10	0.055850	0.049474	0.0	331.609111
min	1.001254e+06	1.236005e+08	40.499790	-74.249840	1905.0	50.000000
25%	1.507574e+07	2.459183e+10	40.688730	-73.982570	1905.0	340.000000
50%	2.922911e+07	4.912069e+10	40.722300	-73.954440	1905.0	625.000000
75%	4.328308e+07	7.399747e+10	40.762750	-73.932340	1905.0	913.000000
max	5.736742e+07	9.876313e+10	40.916970	-73.705220	1905.0	1200.000000

Visualization

what is the distribution of listing prices?

In [33]:

```
plt.figure(figsize = (10,6))
sns.histplot(df['price'], bins=50, kde=True, color = 'red')
plt.title('Distribution of Listing Price')
plt.xlabel('Price $')
plt.ylabel('Frequency')
plt.show()
```



How Diffrent Room Types Distributed?

In [37]:

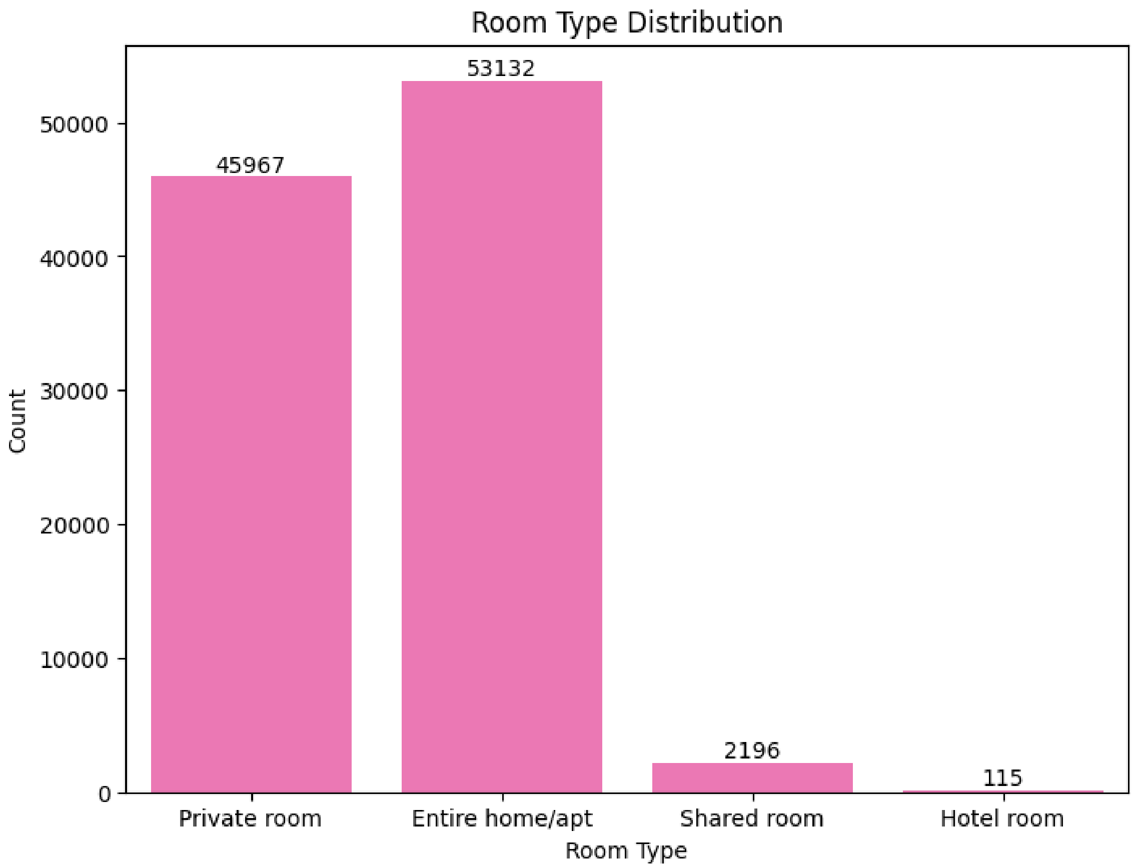
```
df['room type'].value_counts()
```

Out[37]:

```
Entire home/apt    53132
Private room       45967
Shared room         2196
Hotel room           115
Name: room type, dtype: int64
```

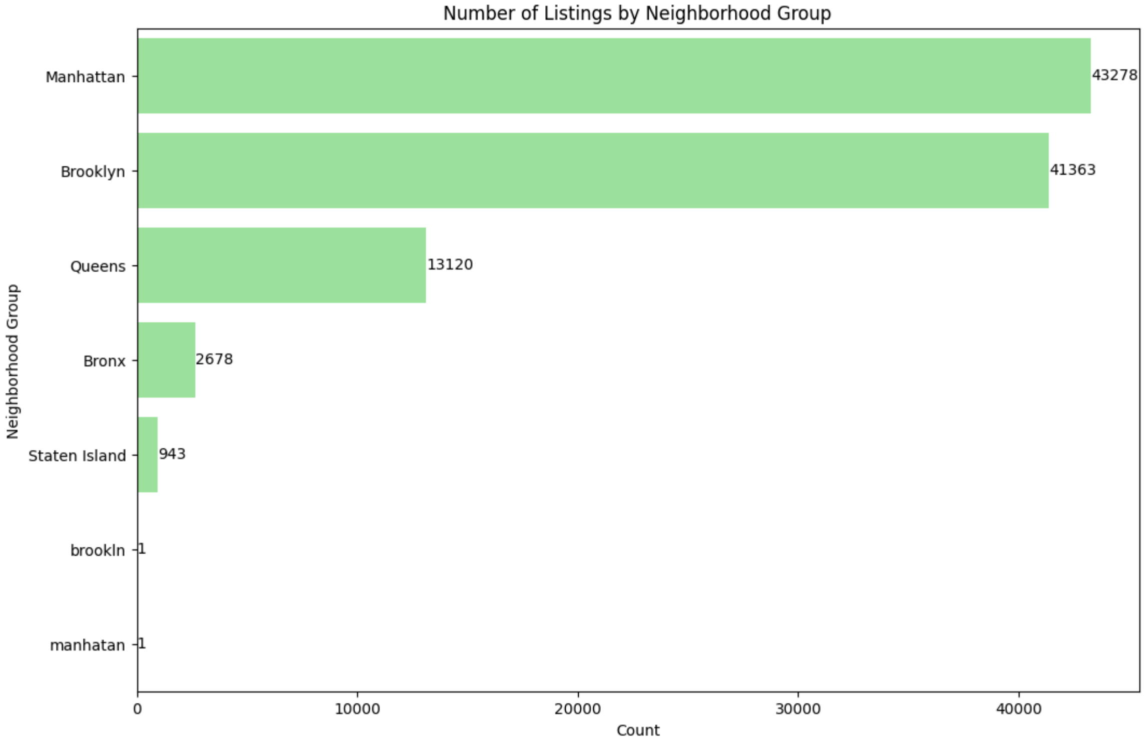
In [39]:

```
plt.figure(figsize = (8,6))
ax = sns.countplot(x = 'room type', data = df, color = 'hotpink')
for bars in ax.containers:
    ax.bar_label(bars)
plt.title('Room Type Distribution')
plt.xlabel('Room Type')
plt.ylabel('Count')
plt.show()
```



How are Listings distributed across different neighboehoods?

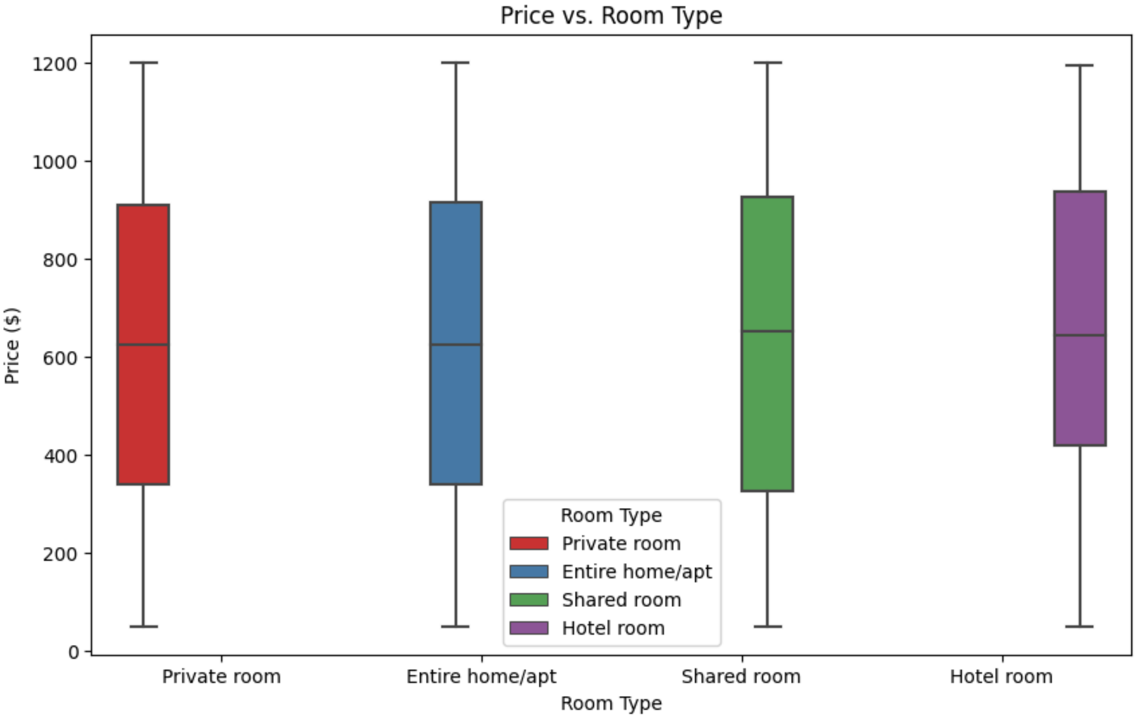
```
In [44]: plt.figure(figsize = (12,8))
         ax = sns.countplot(y = 'neighbourhood group', data = df, color = 'lightgreen',
                            order = df['neighbourhood group'].value_counts().index)
         for bars in ax.containers:
             ax.bar_label(bars)
         plt.title('Number of Listings by Neighborhood Group')
         plt.xlabel('Count')
         plt.ylabel('Neighborhood Group')
         plt.show()
```



What the Relationship b/w Price and Room Type?

In [45]:

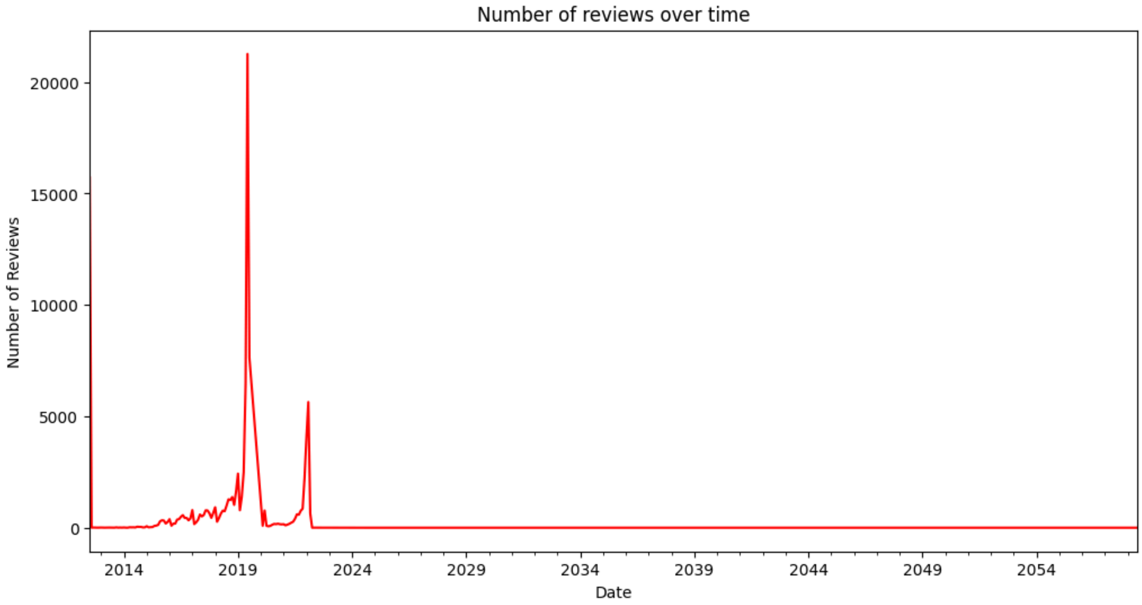
```
plt.figure(figsize = (10,6))
sns.boxplot(x= 'room type', y= 'price', hue = 'room type', data=df, palette = 'S
plt.title('Price vs. Room Type')
plt.xlabel('Room Type')
plt.ylabel('Price ($)')
plt.legend(title='Room Type')
plt.show()
```



How Has the Number Reviews change Over Time?

In [46]:
```
df['last review'] = pd.to_datetime(df['last review'])
reviews_over_time = df.groupby(df['last review'].dt.to_period('M')).size()

plt.figure(figsize = (12,6))
reviews_over_time.plot(kind = 'line',color='red')
plt.title('Number of reviews over time')
plt.xlabel('Date')
plt.ylabel('Number of Reviews')
plt.show()
```



In [47]:
```
df.to_csv('cleaned_data.csv', index=False)
```

In [ ]: