In [3]:
```
from google.colab import files
uploaded=files.upload()
```

Choose Files No file selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Airbnb_data.csv to Airbnb_data.csv
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
df=pd.read_csv("Airbnb_data.csv")
df
```

Out[5]:

	id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_p
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3	1.0	Real Bed	
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7	1.0	Real Bed	
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5	1.0	Real Bed	moc
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4	1.0	Real Bed	fl
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2	1.0	Real Bed	moc
...	...	...	...	...	...	...	...	...	
74106	14549287	4.605170	Apartment	Private room	{}	1	1.0	Real Bed	fl
74107	13281809	5.043425	Apartment	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4	2.0	Real Bed	moc
74108	18688039	5.220356	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	5	1.0	Real Bed	moc
74109	17045948	5.273000	Apartment	Entire home/apt	{TV,"Wireless Internet","Air conditioning",Kit...	2	1.0	Real Bed	
74110	3534845	4.852030	Boat	Entire home/apt	{TV,Internet,"Wireless Internet",Kitchen,"Free...	4	1.0	Real Bed	moc

74111 rows × 31 columns

In [6]:
```
#columns
df.columns
```

Out[6]:
```
Index(['id', 'log_price', 'property_type', 'room_type', 'amenities',
       'accommodates', 'bathrooms', 'bed_type', 'cancellation_policy',
       'cleaning_fee', 'city', 'description', 'first_review',
       'host_has_profile_pic', 'host_identity_verified', 'host_response_rate',
       'host_since', 'instant_bookable', 'last_review', 'latitude',
       'longitude', 'name', 'neighbourhood', 'number_of_reviews',
       'review_scores_rating', 'thumbnail_url', 'zipcode', 'bedrooms', 'beds',
       'Unnamed: 29', 'Unnamed: 30'],
      dtype='object')
```

In [7]:
```
#head
df.head()
```

Out[7]:

	id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_policy
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3	1.0	Real Bed	strict
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7	1.0	Real Bed	strict
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5	1.0	Real Bed	moderate
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4	1.0	Real Bed	flexible
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2	1.0	Real Bed	moderate

5 rows × 31 columns

In [8]:
```
#information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74111 entries, 0 to 74110
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       74111 non-null  int64  
 1   log_price                74111 non-null  float64
 2   property_type            74111 non-null  object 
 3   room_type                74111 non-null  object 
 4   amenities                74111 non-null  object 
 5   accommodates             74111 non-null  int64  
 6   bathrooms                73911 non-null  float64
 7   bed_type                 74111 non-null  object 
 8   cancellation_policy      74111 non-null  object 
 9   cleaning_fee             74111 non-null  bool   
 10  city                     74111 non-null  object 
 11  description              74111 non-null  object 
 12  first_review             58247 non-null  object 
 13  host_has_profile_pic     73923 non-null  object 
 14  host_identity_verified   73923 non-null  object 
 15  host_response_rate       55812 non-null  object 
 16  host_since               73923 non-null  object 
 17  instant_bookable         74111 non-null  object 
 18  last_review              58284 non-null  object 
 19  latitude                 74111 non-null  float64
 20  longitude                74111 non-null  float64
 21  name                     74111 non-null  object 
 22  neighbourhood            67239 non-null  object 
 23  number_of_reviews        74111 non-null  int64  
 24  review_scores_rating     57389 non-null  float64
 25  thumbnail_url            65895 non-null  object 
 26  zipcode                  73143 non-null  object 
 27  bedrooms                 74020 non-null  float64
 28  beds                     73980 non-null  float64
 29  Unnamed: 29              0 non-null      float64
 30  Unnamed: 30              0 non-null      float64
dtypes: bool(1), float64(9), int64(3), object(18)
memory usage: 17.0+ MB
```

In [9]:
```
#null values
df.isnull().sum()
```

Out[9]:

	0
id	0
log_price	0
property_type	0
room_type	0
amenities	0
accommodates	0
bathrooms	200
bed_type	0
cancellation_policy	0
cleaning_fee	0
city	0
description	0
first_review	15864
host_has_profile_pic	188
host_identity_verified	188
host_response_rate	18299
host_since	188
instant_bookable	0
last_review	15827
latitude	0
longitude	0
name	0
neighbourhood	6872
number_of_reviews	0
review_scores_rating	16722
thumbnail_url	8216
zipcode	968
bedrooms	91
beds	131
Unnamed: 29	74111
Unnamed: 30	74111

dtype: int64

In [10]:
```
#Insights
#1.Several columns have the high missing values like 'review_scores_rating', 'neighbourhood','Zipcode', 'host_r
#2.there are unwanted columns also present in the list which are not useful in regression like 'thumbnail_url',
```

In [11]:
```
#drop rows with no useful data
df1=df.drop(['thumbnail_url','Unnamed: 29','Unnamed: 30','name','id','amenities','description','zipcode','first_
df1
```

Out[11]:

	log_price	property_type	room_type	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	city	host_ha
0	5.010635	Apartment	Entire home/apt	3	1.0	Real Bed	strict	True	NYC	
1	5.129899	Apartment	Entire home/apt	7	1.0	Real Bed	strict	True	NYC	
2	4.976734	Apartment	Entire home/apt	5	1.0	Real Bed	moderate	True	NYC	
3	6.620073	House	Entire home/apt	4	1.0	Real Bed	flexible	True	SF	
4	4.744932	Apartment	Entire home/apt	2	1.0	Real Bed	moderate	True	DC	
...	...	...	...	...	...	...	...	...	...	
74106	4.605170	Apartment	Private room	1	1.0	Real Bed	flexible	False	NYC	
74107	5.043425	Apartment	Entire home/apt	4	2.0	Real Bed	moderate	True	LA	
74108	5.220356	Apartment	Entire home/apt	5	1.0	Real Bed	moderate	True	NYC	
74109	5.273000	Apartment	Entire home/apt	2	1.0	Real Bed	strict	True	NYC	
74110	4.852030	Boat	Entire home/apt	4	1.0	Real Bed	moderate	False	LA	

74111 rows × 19 columns

In [12]:
```
#information
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74111 entries, 0 to 74110
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   log_price               74111 non-null  float64
 1   property_type           74111 non-null  object 
 2   room_type               74111 non-null  object 
 3   accommodates            74111 non-null  int64  
 4   bathrooms               73911 non-null  float64
 5   bed_type                74111 non-null  object 
 6   cancellation_policy     74111 non-null  object 
 7   cleaning_fee            74111 non-null  bool   
 8   city                    74111 non-null  object 
 9   host_has_profile_pic    73923 non-null  object 
 10  host_identity_verified  73923 non-null  object 
 11  instant_bookable        74111 non-null  object 
 12  latitude                74111 non-null  float64
 13  longitude               74111 non-null  float64
 14  neighbourhood           67239 non-null  object 
 15  number_of_reviews       74111 non-null  int64  
 16  review_scores_rating    57389 non-null  float64
 17  bedrooms                74020 non-null  float64
 18  beds                    73980 non-null  float64
dtypes: bool(1), float64(7), int64(2), object(9)
memory usage: 10.2+ MB
```

In [7]:
```
#Handle the missing numerical columns
numerical_cols=['bathrooms','beds','bedrooms','review_scores_rating']
for col in numerical_cols:
    df1[col].fillna(df1[col].mean(),inplace=True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 4
      2 numerical_cols=['bathrooms','beds','bedrooms','review_scores_rating']
      3 for col in numerical_cols:
----> 4     df1[col].fillna(df1[col].mean(),inplace=True)

NameError: name 'df1' is not defined
```

In [14]:
```
#missing values
df1.isnull().sum()
```

Out[14]:

	0
log_price	0
property_type	0
room_type	0
accommodates	0
bathrooms	0
bed_type	0
cancellation_policy	0
cleaning_fee	0
city	0
host_has_profile_pic	188
host_identity_verified	188
instant_bookable	0
latitude	0
longitude	0
neighbourhood	6872
number_of_reviews	0
review_scores_rating	0
bedrooms	0
beds	0

dtype: int64

In [15]:
```
#drop rows where essential categorical field are missing
df1.dropna(subset=['neighbourhood','host_has_profile_pic','host_identity_verified'],inplace=True)
```

In [16]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 67054 entries, 0 to 74110
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   log_price               67054 non-null  float64
 1   property_type           67054 non-null  object 
 2   room_type               67054 non-null  object 
 3   accommodates            67054 non-null  int64  
 4   bathrooms               67054 non-null  float64
 5   bed_type                67054 non-null  object 
 6   cancellation_policy     67054 non-null  object 
 7   cleaning_fee            67054 non-null  bool   
 8   city                    67054 non-null  object 
 9   host_has_profile_pic    67054 non-null  object 
 10  host_identity_verified  67054 non-null  object 
 11  instant_bookable        67054 non-null  object 
 12  latitude                67054 non-null  float64
 13  longitude               67054 non-null  float64
 14  neighbourhood           67054 non-null  object 
 15  number_of_reviews       67054 non-null  int64  
 16  review_scores_rating    67054 non-null  float64
 17  bedrooms                67054 non-null  float64
 18  beds                    67054 non-null  float64
dtypes: bool(1), float64(7), int64(2), object(9)
memory usage: 9.8+ MB
```

In [17]:
```
#valuecount
df1['property_type'].value_counts()
```

Out[17]:

property_type	count
Apartment	45628
House	13820
Condominium	2392
Townhouse	1483
Loft	1150
Other	548
Guesthouse	437
Bed & Breakfast	401
Bungalow	310
Dorm	127
Guest suite	120
Villa	110
Timeshare	76
In-law	71
Boutique hotel	69
Hostel	63
Camper/RV	59
Cabin	55
Boat	50
Serviced apartment	19
Castle	12
Tent	11
Vacation home	10
Yurt	7
Treehouse	6
Chalet	5
Hut	4
Tipi	3
Earth House	3
Cave	2
Casa particular	1
Parking Space	1
Train	1

dtype: int64

In [18]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['property_type']=LE.fit_transform(df1['property_type'])
df1['property_type']
```

Out[18]:

	property_type
0	0
1	0
2	0
3	17
4	0
...	...
74106	0
74107	0
74108	0
74109	0
74110	2

67054 rows × 1 columns

dtype: int64

In [19]:

```
#valuecount
df1['property_type'].value_counts()
```

Out[19]:

property_type	count
0	45628
17	13820
11	2392
27	1483
20	1150
21	548
15	437
1	401
4	310
12	127
14	120
31	110
25	76
19	71
3	69
16	63
6	59
5	55
2	50
23	19
8	12
24	11
30	10
32	7
29	6
10	5
18	4
26	3
13	3
9	2
7	1
22	1
28	1

dtype: int64

In [21]:
```
#valuecount
df1['neighbourhood'].value_counts()
```

Out[21]:

neighbourhood	count
Williamsburg	2855
Bedford-Stuyvesant	2157
Bushwick	1595
Mid-Wilshire	1392
Upper West Side	1386
...	...
Artesia	1
La Habra	1
Irwindale	1
Rolling Hills	1
Grant City	1

619 rows × 1 columns

dtype: int64

In [22]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['neighbourhood']=LE.fit_transform(df1['neighbourhood'])
df1['neighbourhood']
```

Out[22]:

	neighbourhood
0	77
1	252
2	247
3	325
4	119
...	...
74106	605
74107	255
74108	605
74109	592
74110	319

67054 rows × 1 columns

dtype: int64

In [23]:
```
#valuecount
df1['neighbourhood'].value_counts()
```

Out[23]:

neighbourhood	count
605	2855
44	2157
84	1595
348	1392
556	1386
...	...
20	1
294	1
275	1
464	1
234	1

619 rows × 1 columns

dtype: int64

In [25]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['cleaning_fee']=LE.fit_transform(df1['cleaning_fee'])
df1['cleaning_fee']
```

Out[25]:

	cleaning_fee
0	1
1	1
2	1
3	1
4	1
...	...
74106	0
74107	1
74108	1
74109	1
74110	0

67054 rows × 1 columns

dtype: int64

In [26]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['cancellation_policy']=LE.fit_transform(df1['cancellation_policy'])
df1['cancellation_policy']
```

Out[26]:

	cancellation_policy
0	2
1	2
2	1
3	0
4	1
...	...
74106	0
74107	1
74108	1
74109	2
74110	1

67054 rows × 1 columns

dtype: int64

In [27]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['bed_type']=LE.fit_transform(df1['bed_type'])
df1['bed_type']
```

Out[27]:

	bed_type
0	4
1	4
2	4
3	4
4	4
...	...
74106	4
74107	4
74108	4
74109	4
74110	4

67054 rows × 1 columns

dtype: int64

In [28]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['city']=LE.fit_transform(df1['city'])
df1['city']
```

Out[28]:

	city
0	4
1	4
2	4
3	5
4	2
...	...
74106	4
74107	3
74108	4
74109	4
74110	3

67054 rows × 1 columns

dtype: int64

In [29]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['host_has_profile_pic']=LE.fit_transform(df1['host_has_profile_pic'])
df1['host_has_profile_pic']
```

Out[29]:

	host_has_profile_pic
0	1
1	1
2	1
3	1
4	1
...	...
74106	1
74107	1
74108	1
74109	1
74110	1

67054 rows × 1 columns

dtype: int64

In [30]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['host_identity_verified']=LE.fit_transform(df1['host_identity_verified'])
df1['host_identity_verified']
```

Out[30]:

	host_identity_verified
0	1
1	0
2	1
3	1
4	1
...	...
74106	1
74107	0
74108	1
74109	0
74110	1

67054 rows × 1 columns

dtype: int64

In [31]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['instant_bookable']=LE.fit_transform(df1['instant_bookable'])
df1['instant_bookable']
```

Out[31]:

	instant_bookable
0	0
1	1
2	1
3	0
4	1
...	...
74106	0
74107	0
74108	1
74109	1
74110	0

67054 rows × 1 columns

dtype: int64

In [34]:
```
#data transformation
from sklearn.preprocessing import LabelEncoder
LE= LabelEncoder()
df1['room_type']=LE.fit_transform(df1['room_type'])
df1['room_type']
```

Out[34]:

	room_type
0	0
1	0
2	0
3	0
4	0
...	...
74106	1
74107	0
74108	0
74109	0
74110	0

67054 rows × 1 columns

dtype: int64

In [35]:
```
#shape
df1.shape
```

Out[35]: (67054, 19)

In [39]:
```
#head
df1.head()
```

Out[39]:

	log_price	property_type	room_type	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	city	host_has_pro
0	5.010635	0	0	3	1.0	4	2	1	4	
1	5.129899	0	0	7	1.0	4	2	1	4	
2	4.976734	0	0	5	1.0	4	1	1	4	
3	6.620073	17	0	4	1.0	4	0	1	5	
4	4.744932	0	0	2	1.0	4	1	1	2	

In [40]:
```
#scaling
df2=df1[['latitude']]
from sklearn.preprocessing import StandardScaler
SS=StandardScaler()
SS_cont=SS.fit_transform(df2)
SS_cont=pd.DataFrame(SS_cont)
SS_cont.columns=[['latitude']]
SS_cont
```

Out[40]:

	latitude
0	0.658366
1	0.681956
2	0.696191
3	-0.332960
4	0.058084
...	...
67049	0.662604
67050	-1.655101
67051	0.661832
67052	0.672607
67053	-1.692541

67054 rows × 1 columns

In [41]:
```
#scaling
df2=df1[['longitude']]
from sklearn.preprocessing import StandardScaler
SS=StandardScaler()
```

```
SS_cont=SS.fit_transform(df2)
SS_cont=pd.DataFrame(SS_cont)
SS_cont.columns=[['longitude']]
SS_cont
```

Out[41]:

	longitude
0	0.774242
1	0.774362
2	0.776475
3	-1.486132
4	0.632246
...	...
67049	0.776678
67050	-1.297819
67051	0.776539
67052	0.773843
67053	-1.288316

67054 rows × 1 columns

In [46]:
```
#head
df1.head()
```

Out[46]:

	log_price	property_type	room_type	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	city	host_has_pro
0	5.010635	0	0	3	1.0	4	2	1	4	
1	5.129899	0	0	7	1.0	4	2	1	4	
2	4.976734	0	0	5	1.0	4	1	1	4	
3	6.620073	17	0	4	1.0	4	0	1	5	
4	4.744932	0	0	2	1.0	4	1	1	2	

In [45]:
```
#scaling
df2=df1[['review_scores_rating']]
from sklearn.preprocessing import StandardScaler
SS=StandardScaler()
SS_cont=SS.fit_transform(df2)
SS_cont=pd.DataFrame(SS_cont)
SS_cont.columns=[['review_scores_rating']]
SS_cont
```

Out[45]:

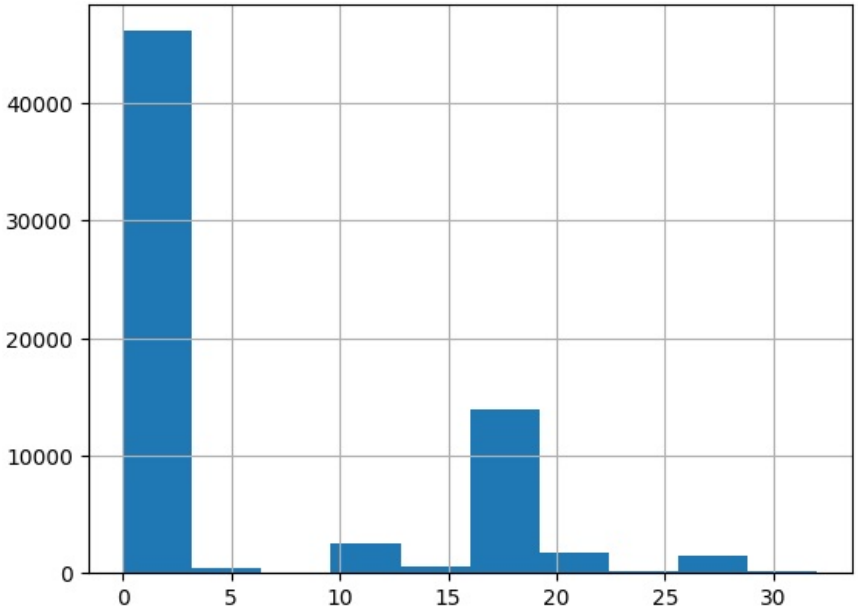
	review_scores_rating
0	0.869224
1	-0.157806
2	-0.304525
3	-0.001204
4	-7.933895
...	...
67049	-0.001204
67050	-0.157806
67051	-0.011088
67052	-0.001204
67053	0.282350

67054 rows × 1 columns

In [47]:
```
#Insights
#1.shape after cleaning the data 67,054 rows
#2.Handled all the missing values with mean
#3.text-heavy and unused columns are being dropped
#4.we have used level encoder for the categorical variable
#5.we have also used standard scaling for continuous variables
```

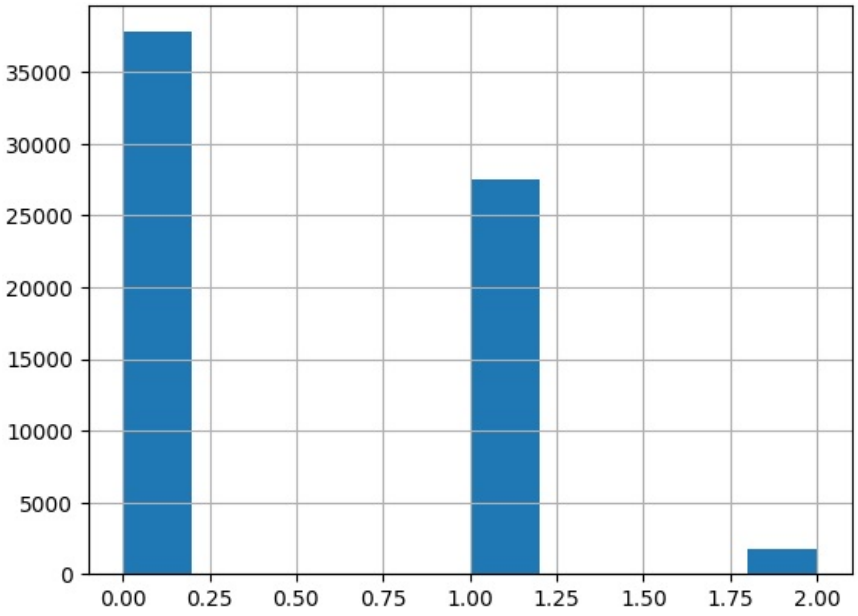
In [50]:
```
#histogram
df1['property_type'].hist()
```

Out[50]: <Axes: >



In [51]:
```
#histogram
df1['room_type'].hist()
```

Out[51]: <Axes: >



In [52]:
```
#correlation
df1.corr()
```

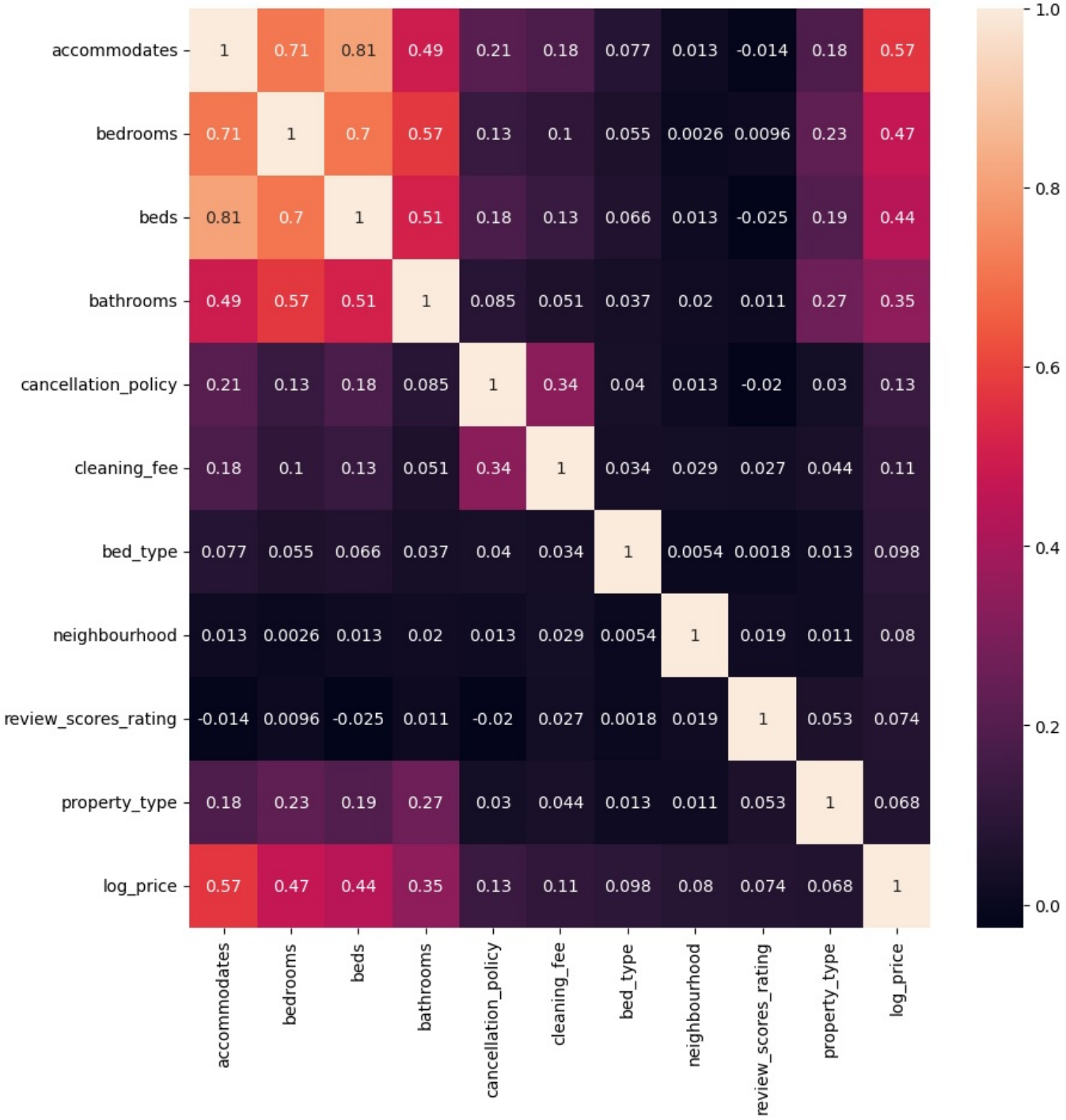
Out[52]:

	log_price	property_type	room_type	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee
log_price	1.000000	0.068300	-0.603475	0.569273	0.346301	0.097634	0.132936	0.110449
property_type	0.068300	1.000000	0.061218	0.183542	0.267106	0.013213	0.030457	0.044435
room_type	-0.603475	0.061218	1.000000	-0.456095	-0.104788	-0.154856	-0.172698	-0.208020
accommodates	0.569273	0.183542	-0.456095	1.000000	0.493649	0.077112	0.207212	0.181826
bathrooms	0.346301	0.267106	-0.104788	0.493649	1.000000	0.037066	0.084506	0.051193
bed_type	0.097634	0.013213	-0.154856	0.077112	0.037066	1.000000	0.040272	0.033810
cancellation_policy	0.132936	0.030457	-0.172698	0.207212	0.084506	0.040272	1.000000	0.338146
cleaning_fee	0.110449	0.044435	-0.208020	0.181826	0.051193	0.033810	0.338146	1.000000
city	0.024852	-0.102333	0.066223	-0.103165	-0.078996	0.007590	-0.022711	-0.020903
host_has_profile_pic	-0.014373	0.005513	-0.000385	-0.005285	-0.005098	-0.001109	0.027921	0.024698
host_identity_verified	0.017769	0.025193	-0.058794	0.051042	0.014674	0.010214	0.156014	0.158714
instant_bookable	-0.040710	0.021418	0.029114	0.052075	0.002609	0.026698	0.006891	0.005266
latitude	-0.037576	-0.295796	0.056462	-0.085687	-0.144328	0.005454	0.004797	-0.070038
longitude	-0.084953	-0.285329	0.059156	-0.090873	-0.139165	0.005353	-0.004206	-0.075077
neighbourhood	0.080256	0.011184	-0.034328	0.013065	0.019504	0.005436	0.012868	0.028525
number_of_reviews	-0.035907	0.053410	-0.023195	0.044292	-0.039582	0.007032	0.193122	0.113281
review_scores_rating	0.073753	0.053216	-0.036949	-0.014444	0.010883	0.001826	-0.020191	0.026691
bedrooms	0.470899	0.229259	-0.238396	0.707893	0.573090	0.054912	0.127455	0.104004
beds	0.440553	0.191329	-0.313147	0.809392	0.513110	0.066212	0.175229	0.129378

In [55]:
```
#get top 10 most correlated with log_price(excluding log_price itself)
corr_matrix=df1.corr()
corr_matrix['log_price'].sort_values(ascending=False)[1:11].index.tolist()
```

Out[55]:
```
['accommodates',
 'bedrooms',
 'beds',
 'bathrooms',
 'cancellation_policy',
 'cleaning_fee',
 'bed_type',
 'neighbourhood',
 'review_scores_rating',
 'property_type']
```

In [56]:
```
#plot heatmap of those features + log_price
plt.figure(figsize=(10,10))
sns.heatmap(df1[corr_matrix['log_price'].sort_values(ascending=False)[1:11].index.tolist()+['log_price']].corr(
plt.show()
```



In [57]:
```
#Insights
#1.from correlation we found the features that are correlated with log_price
#2.positive correlation means if the features will increase then the log price will also get increased
```

In [58]:
```
#feature
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error

#split the dataset
X=df1.drop(['log_price'],axis=1)
y=df1['log_price']
```

In [59]:
```
#train_test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [60]:
```
#Initialize and train a linear regression model
model=LinearRegression()
model.fit(X_train,y_train)
```

Out[60]:
```
▾ LinearRegression
LinearRegression()
```

In [62]:
```
#predict on the test set
```

```
y_pred=model.predict(X_test)
y_pred
```

Out[62]:
```
array([4.96439812, 5.17461749, 5.089127  , ..., 4.41382101, 4.30301095,
       4.99403089])
```

In [63]:
```
#Evaluate the model
print('R2 score:',r2_score(y_test,y_pred))
print('MSE:',mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
R2 score: 0.5423080038938426
MSE: 0.2309363732917148
RMSE: 0.4805583973792517
```

In [64]:
```
#Insights
#1.An Rsquare of 0.54 is descent for real world data like Airbnb listing
#2.For making data more accurate for can go for Random Forest or Gradient Boosting
```

In [65]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
```

In [66]:
```
#train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [67]:
```
#train_random_forest_regressor
rf=RandomForestRegressor()
rf.fit(X_train,y_train)
```

Out[67]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [68]:
```
#predict and evaluate
y_pred=rf.predict(X_test)
print('R2 score:',r2_score(y_test,y_pred))
print('MSE:',mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
R2 score: 0.6976416665166087
MSE: 0.15256010059871322
RMSE: 0.3905894271466052
```

In [ ]:
```
#Insights
#1.It explains of about 69.76% of the variance in log_price, a significant improvement over linear regression
#2. RMSE of 0.39 lower then then linear regression of 0.48 indicates the better predictive accuracy
```

In [ ]:
```

```