

Python Project EDA & Data Viz - Airbnb Listing 2024(New York)

STEPS

importing all dependencies (lib)

loading datasets

initial exploration

Data cleaning

Data Analysis

TASK1 IMPORTING ALL DEPENDENCIES

In [31]

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

TASK2 LOADING DATASETS

In [45]

```
data = pd.read_csv('datasets.csv', encoding_errors='ignore')
```

TASK3 INTIAL EXPLORATION

In [46]

```
data.head(3)
```

Out[46]

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	...	last_review	reviews_per_month	calculated_host_listings_count	availability_365	number
0	1312228e+06	Rental unit in Brooklyn	7130382	Walter	Brooklyn	Clinton Hill	40.683710	-73.964610	Private room	55.0	...	2012/15	0.03		1.0	0.0
1	4.527754e+07	Rental unit in New York	51501835	Jennifer	Manhattan	Hell's Kitchen	40.766610	-73.988100	Entire home/apt	144.0	...	01/05/23	0.24		139.0	364.0
2	9.710000e+17	Rental unit in New York	528871354	Joshua	Manhattan	Chelsea	40.750764	-73.994605	Entire home/apt	187.0	...	18/12/23	1.67		1.0	343.0

3 rows x 22 columns

In [48]

```
data.tail(3)
```

Out[48]

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	...	last_review	reviews_per_month	calculated_host_listings_count	availability_365	num
20767	5.385272e+07	Rental unit in New York	304317395	Jeff	Manhattan	Hell's Kitchen	40.757350	-73.992430	Entire home/apt	299.0	...	08/12/23	2.09		1.0	0.0
20768	7.830000e+17	Rental unit in New York	163083101	Marissa	Manhattan	Chinatown	40.713750	-73.991470	Entire home/apt	115.0	...	17/09/23	0.91		1.0	361.0
20769	5.660000e+17	Rental unit in Queens	93827372	Glenroy	Queens	Rosedale	40.658874	-73.728651	Private room	102.0	...	10/12/23	4.50		1.0	0.0

3 rows x 22 columns

In [49]

```
data.shape
```

Out[49]

```
(20770, 22)
```

In [111]

```
data.info()
```

Out[111]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20770 entries, 0 to 20769
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   id                   20770 non-null  float64
 1   name                 20770 non-null  object
 2   host_id              20770 non-null  int64
 3   host_name            20770 non-null  object
 4   neighbourhood_group  20770 non-null  object
 5   neighbourhood         20770 non-null  object
 6   latitude             20770 non-null  float64
 7   longitude            20770 non-null  float64
 8   room_type           20770 non-null  object
 9   price               20770 non-null  float64
10  minimum_nights      20770 non-null  float64
11  number_of_reviews   20770 non-null  float64
12  last_review         20770 non-null  object
13  reviews_per_month  20770 non-null  float64
14  calculated_host_listings_count  20770 non-null  float64
15  review_per_month    20770 non-null  float64
16  availability_365     20770 non-null  float64
17  number_of_reviews_ltm  20770 non-null  float64
18  license             20770 non-null  object
19  rating              20770 non-null  float64
20  bedrooms            20770 non-null  int64
21  beds                20770 non-null  int64
22  baths               20770 non-null  float64
dtypes: float64(18), int64(2), object(10)
memory usage: 3.1+ MB
```

In [143]

```
data.columns
```

Out[143]

```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'number_of_reviews_ltm', 'license', 'rating',
       'bedrooms', 'beds', 'baths'],
      dtype='object')
```

STATISTICAL SUMMARY

In [145]

```
data.describe()
```

Out[145]

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	number_of_reviews_ltm	beds
count	2.077000e+04	2.077000e+04	40.726801	-73.939179	187.714940	28.558493	42.610605	1.257589	18.866666	206.067957	20763.000000	20770.000000
mean	3.033858e+17	1.749049e+08	0.066293	0.061403	1023.245124	33.532697	73.523401	1.904472	70.921443	135.077259	213.54876	1.213592
std	3.901221e+17	1.725657e+08	0.066293	0.061403	1023.245124	33.532697	73.523401	1.904472	70.921443	135.077259	213.54876	1.213592
min	2.595000e+03	1.678000e+03	40.500314	-74.249840	10.000000	1.000000	1.000000	0.010000	1.000000	0.000000	0.000000	1.000000
25%	2.707260e+07	1.041184e+07	40.684159	-73.980755	30.000000	30.000000	4.000000	0.210000	1.000000	87.000000	1.000000	1.000000
50%	4.992852e+07	1.088990e+08	40.722890	-73.949597	125.000000	30.000000	14.000000	0.630000	2.000000	215.000000	3.000000	1.000000
75%	7.222000e+07	1.143997e+08	40.763106	-73.917475	199.000000	30.000000	49.000000	1.800000	5.000000	353.000000	15.000000	2.000000
max	1.050000e+18	1.504035e+08	40.911147	-73.713650	100000.000000	1250.000000	1865.000000	75.400000	713.000000	365.000000	1075.000000	42.000000

TASK 4 DATA CLEANING

In [200]

```
# to identify missing values and here we are dropping those records
# data.fillna() to fill those missing records

data.dropna(inplace=True)
data.isnull().sum()
```

Out[200]

```
id                0
name              0
host_id           0
host_name         0
neighbourhood_group  0
neighbourhood     0
latitude          0
longitude         0
room_type        0
price            0
minimum_nights    0
number_of_reviews  0
last_review       0
reviews_per_month  0
calculated_host_listings_count  0
availability_365   0
number_of_reviews_ltm  0
license           0
rating            0
bedrooms         0
beds             0
baths            0
dtype: int64
```

In [211]

```
data.shape
```

Out[211]

```
(20736, 22)
```

In [380]

```
# Dealing with duplicate records
data.duplicated().sum()

# to save the duplicate records
data[data.duplicated()==False]

# to delete all the duplicate records
data.drop_duplicates(inplace=True)

# to check duplicates
data.duplicated().sum()
```

Out[380]

```
np.int64(0)
```

In [381]

```
# TYPE CASTING
# to check and correct the data types or change the data types
data.dtypes

data['id']=data['id'].astype(object)
data['host_id']=data['host_id'].astype(object)
data.dtypes
```

Out[381]

```
id                object
name              object
host_id           object
host_name         object
neighbourhood_group  object
neighbourhood     object
latitude          float64
longitude         float64
room_type        object
price            float64
minimum_nights    float64
number_of_reviews  float64
last_review       object
reviews_per_month  float64
calculated_host_listings_count  float64
availability_365   float64
number_of_reviews_ltm  float64
license           object
rating            object
bedrooms         object
beds             int64
baths            object
dtype: object
```

EDA

TASK5 DATA ANALYSIS

UNIVARIABLE ANALYSIS

Analysis of each column and value and there distributions

In [411]

```
# PRICE DISTRIBUTIONS
data['price']
```

Out[411]

```
0      55.0
1      34.0
2     147.0
3     120.0
4      45.0
...
20765    45.0
20766    385.0
20767    259.0
20768    115.0
20769    181.0
Name: price, Length: 20724, dtype: float64
```

In [511]

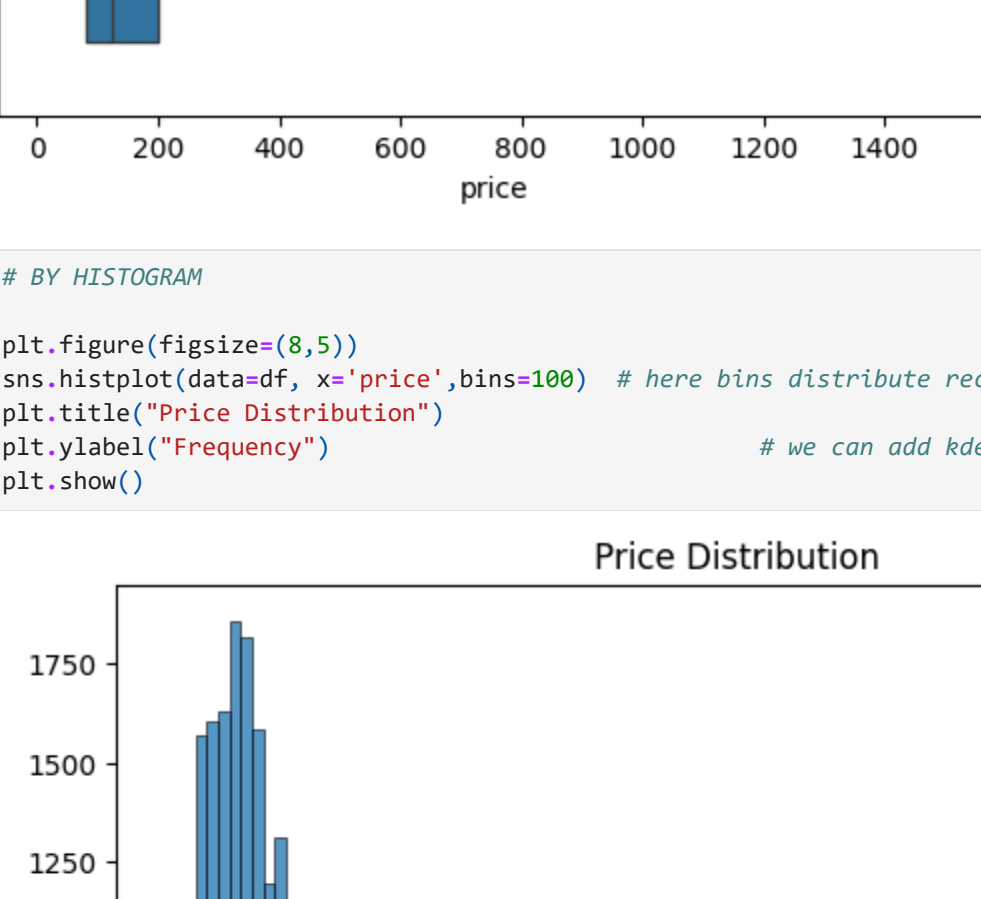
```
# IDENTIFYING OUTLIERS IN PRICE
# BOXPLOT

# sns.boxplot(data=data, x='price') here we first used this,,we got some outliers to deal with them,,we did down method

df = data[data['price']>1500]
sns.boxplot(data=df, x='price')
```

Out[511]

<Axes: xlabel='price'>



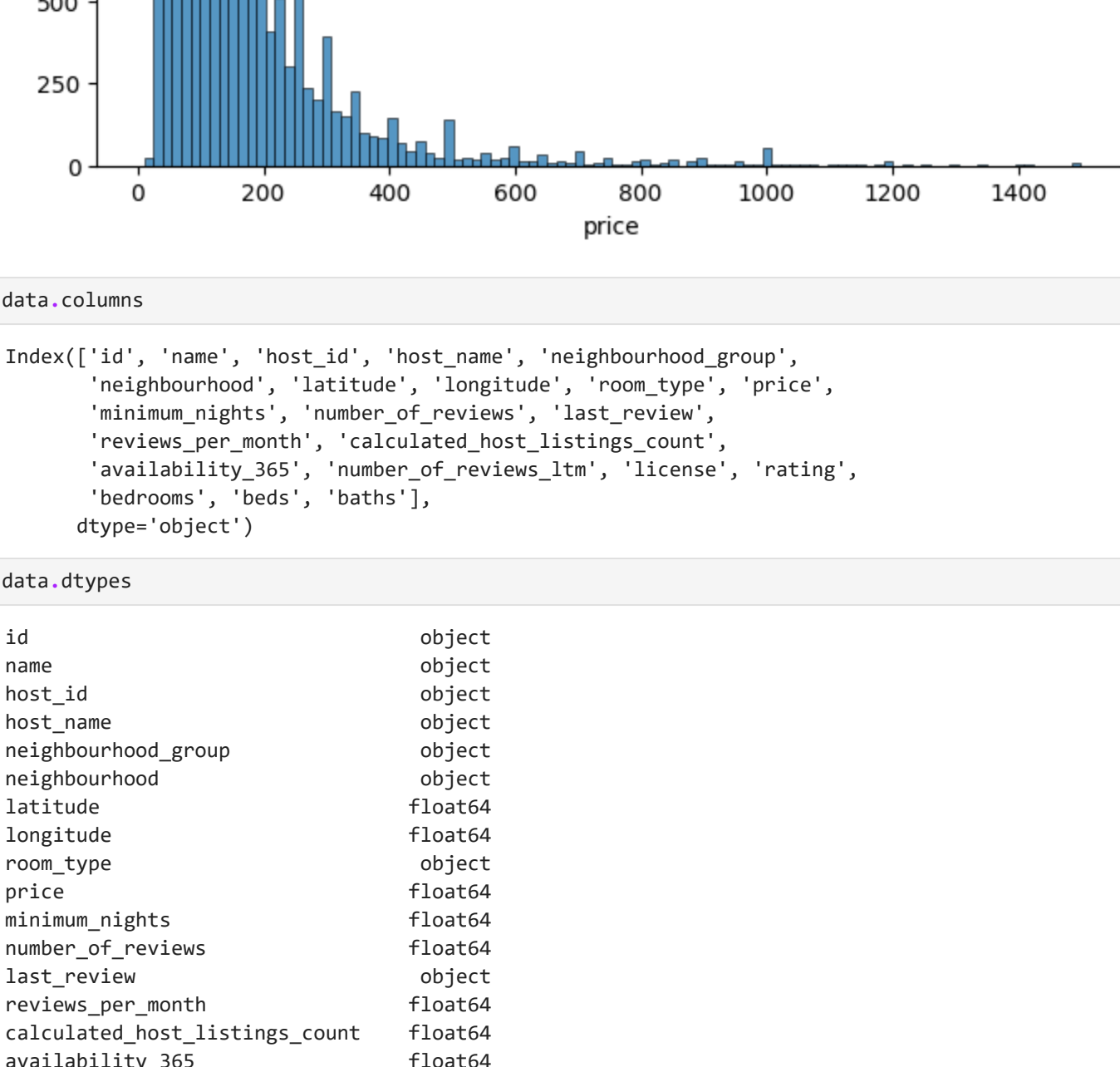
A box plot showing the distribution of 'price'. The x-axis is labeled 'price' and ranges from 0 to 1400. The y-axis represents frequency. The plot shows a long right tail with many outliers represented by small circles to the right of the whiskers.

In [641]

```
# BY HISTOGRAM
plt.figure(figsize=(8,5))
sns.histplot(data=df, x='price', bins=100) # here bins distribute records based on then here 100,,the change can be seen in y axis
plt.title('Price Distribution')
plt.xlabel('price')
plt.ylabel('frequency')
plt.show()
```

Out[641]

Price Distribution



A histogram showing the frequency distribution of 'price'. The x-axis is labeled 'price' and ranges from 0 to 1400. The y-axis is labeled 'frequency' and ranges from 0 to 1750. The distribution is right-skewed, with a peak frequency of approximately 1750 at a price of around 100.

In [591]

```
data.columns
```

Out[591]

```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'number_of_reviews_ltm', 'license', 'rating',
       'bedrooms', 'beds', 'baths'],
      dtype='object')
```

In [601]

```
data.dtypes
```

Out[601]

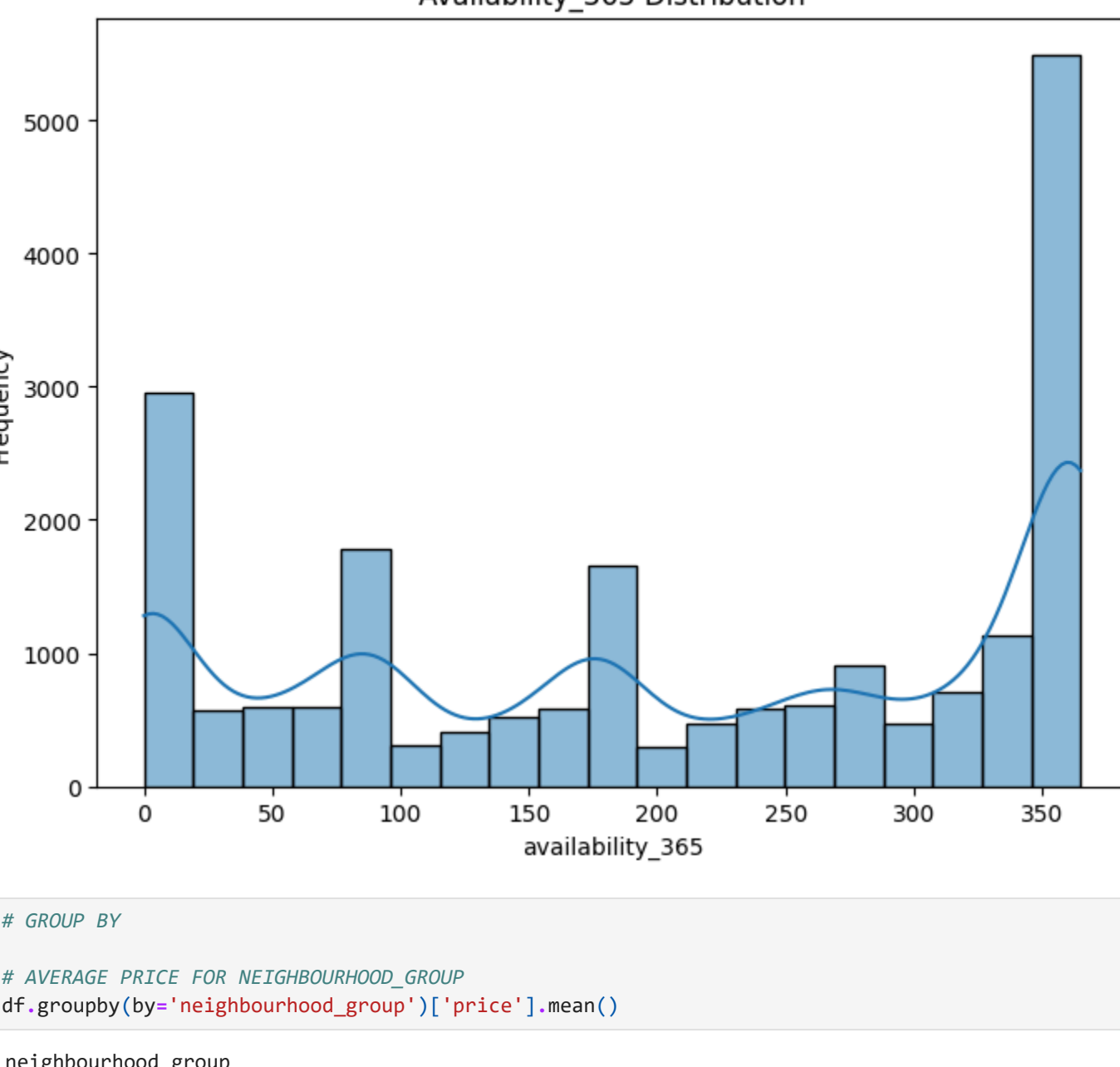
```
id                object
name              object
host_id           object
host_name         object
neighbourhood_group  object
neighbourhood     object
latitude          float64
longitude         float64
room_type        object
price            float64
minimum_nights    float64
number_of_reviews  float64
last_review       object
reviews_per_month  float64
calculated_host_listings_count  float64
availability_365   float64
number_of_reviews_ltm  float64
license           object
rating            object
bedrooms         object
beds             int64
baths            object
dtype: object
```

In [651]

```
plt.figure(figsize=(8,6))
sns.histplot(data=df, x='availability_365', kde=True)
plt.title('Availability_365 Distribution')
plt.xlabel('availability_365')
plt.ylabel('frequency')
plt.show()
```

Out[651]

Availability\_365 Distribution



A histogram showing the frequency distribution of 'availability\_365'. The x-axis is labeled 'availability\_365' and ranges from 0 to 350. The y-axis is labeled 'frequency' and ranges from 0 to 5000. The distribution is right-skewed, with a peak frequency of approximately 5000 at an availability of around 350.

In [681]

```
# GROUP BY
# AVERAGE PRICE FOR NEIGHBOURHOOD_GROUP
df.groupby('neighbourhood_group')['price'].mean()
```

Out[681]

```
neighbourhood_group
Bronx          367.998486
Brooklyn       355.138137
Manhattan      286.360884
Queens         121.683339
Staten Island  118.788889
Name: price, dtype: float64
```

FEATURE ENGINEERING

In [711]

```
# CREATING NEW COLUMN : RETURNS PRICE PER BED
df['price_per_bed']=df['price']/df['beds']
df.head()
```

Out[711]

```
id      name      host_id  host_name  neighbourhood_group  neighbourhood  latitude  longitude  room_type  price  ...  availability_365  number_of_reviews_ltm  license  rating  bedrooms  beds  ba
0      1312228e+06  Rental unit in Brooklyn  7130382  Walter  Brooklyn  Clinton Hill  40.683710  -73.964610  Private room  55.0  ...  0.0  0.0  No License  5  1  1
1      4.5277537e+07  Rental unit in New York  51501835  Jennifer  Manhattan  Hell's Kitchen  40.766610  -73.988100  Entire home/apt  144.0  ...  364.0  2.0  No License  4.67  2  1
2      9710000000000000000.0  Rental unit in New York  528871354  Joshua  Manhattan  Chelsea  40.750764  -73.994605  Entire home/apt  187.0  ...  343.0  6.0  Exempt  4.17  1  2
3      3857863.0  Rental unit in New York  19902271  John And Catherine  Manhattan  Washington Heights  40.835600  -73.942500  Private room  120.0  ...  363.0  12.0  No License  4.64  1  1
4      40896611.0  Condo in New York  61391963  Stay With Vibe  Manhattan  Murray Hill  40.751120  -73.978600  Entire home/apt  85.0  ...  335.0  3.0  No License  4.91  Studio  1
```

5 rows x 25 columns

In [731]

```
# GROUP BY
# AVERAGE PRICE PER BED FOR NEIGHBOURHOOD_GROUP
df.groupby('neighbourhood_group')['price_per_bed'].mean()
```

Out[731]

```
neighbourhood_group
Bronx          74.713639
Brooklyn       89.788493
Manhattan      76.336228
Queens         67.728181
Staten Island  67.728181
Name: price_per_bed, dtype: float64
```

BI VARIABLE ANALYSIS

Analysis based on how one column is related with another column,,relationship between columns,,dependency between columns,,how one variable dependent in another variable

In [741]

```
df.columns
```

Out[741]

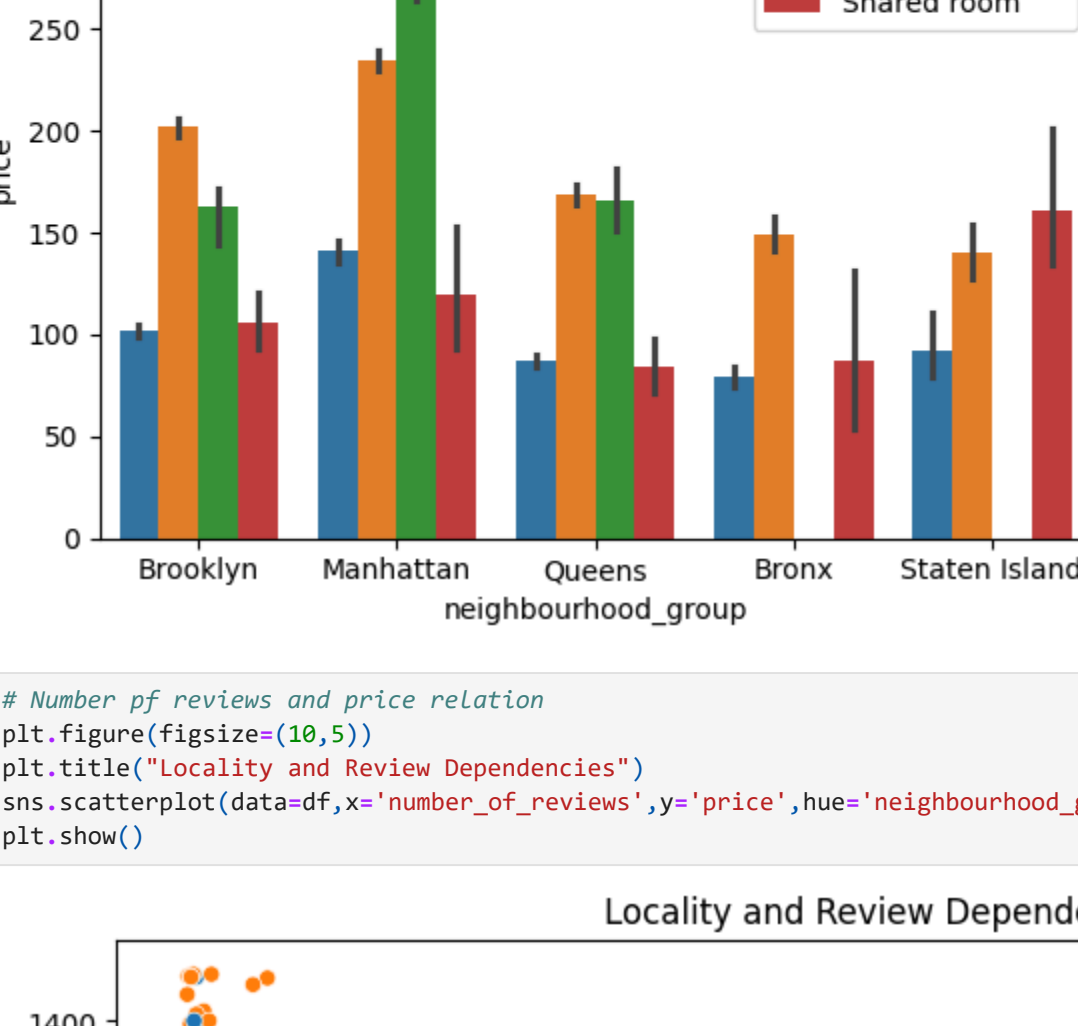
```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'number_of_reviews_ltm', 'license', 'rating',
       'bedrooms', 'beds', 'baths', 'price_per_bed', 'price_per_bed',
       'price_per_bed'],
      dtype='object')
```

In [ ]

```
# Price dependencies on neighbourhood_group
sns.pairplot(data=df, x='neighbourhood_group', y='price', hue='room_type')
```

Out[ ]

<Axes: xlabel='neighbourhood\_group', ylabel='price'>




A pair plot showing the relationship between 'neighbourhood\_group' and 'price'. The x-axis is labeled 'neighbourhood\_group' and has categories: Brooklyn, Manhattan, Queens, Bronx, and Staten Island. The y-axis is labeled 'price' and ranges from 0 to 350. The plot shows four different room types: Private room (blue), Entire home/apt (orange), Hotel room (green), and Shared room (red). The plot shows that 'price' is generally higher for 'Entire home/apt' and 'Hotel room' compared to 'Private room' and 'Shared room'.

In [811]

```
# Number of reviews and price relation
plt.figure(figsize=(8,5))
plt.title('Locality and Review Dependencies')
sns.scatterplot(data=df, x='number_of_reviews', y='price', hue='neighbourhood_group')
plt.show()
```

Out[811]

Locality and Review Dependencies




A scatter plot showing the relationship between 'number\_of\_reviews' and 'price'. The x-axis is labeled 'number\_of\_reviews' and ranges from 0 to 1750. The y-axis is labeled 'price' and ranges from 0 to 1400. The plot shows four different room types: Private room (blue), Entire home/apt (orange), Hotel room (green), and Shared room (red). The plot shows that 'price' is generally higher for 'Entire home/apt' and 'Hotel room' compared to 'Private room' and 'Shared room'.

In [871]

```
# PRED PLOT
sns.pairplot(data=df, vars=['price', 'minimum_nights', 'number_of_reviews', 'availability_365'], hue='room_type')
plt.show()
```

Out[871]

<Axes: >



A pair plot showing the relationship between 'price', 'minimum\_nights', 'number\_of\_reviews', and 'availability\_365'. The plot shows four different room types: Private room (blue), Entire home/apt (orange), Hotel room (green), and Shared room (red). The plot shows that 'price' is generally higher for 'Entire home/apt' and 'Hotel room' compared to 'Private room' and 'Shared room'.

In [911]

```
# GEOGRAPHICAL CHART
# Geographical distributions of AIRBNB listings
plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x='longitude', y='latitude', hue='room_type')
plt.title('Geographical distributions of AIRBNB listings')
plt.show()
```

Out[911]

Geographical distributions of AIRBNB listings



A scatter plot showing the geographical distribution of Airbnb listings. The x-axis is labeled 'longitude' and ranges from -74.2 to -73.7. The y-axis is labeled 'latitude' and ranges from 40.5 to 40.9. The plot shows four different room types: Private room (blue), Entire home/apt (orange), Hotel room (green), and Shared room (red). The plot shows that 'price' is generally higher for 'Entire home/apt' and 'Hotel room' compared to 'Private room' and 'Shared room'.

In [961]

```
# CORRELATION BETWEEN COLUMNS
# Heatmap : correlation of one variable with others for a numerical column
# Heat map only works on numerical column ,so we need to name and sub dataframe with only numerical records
# correlation value will always remain between -1 to +1
# -1 = negative relationship,,one increase another decrease or viceversa
# 0 = no relationship,,one increase another increase or decrease
# +1 = positive relationship,,one increase another also increase ,,
# one decrease another also decrease

corr = df[['latitude', 'longitude', 'price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'availability_365', 'beds']].corr()
corr
```

Out[961]

	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	availability_365	beds
latitude	1.000000	0.047369	0.012688	0.004590	-0.047953	-0.041673	-0.059941	0.017852
longitude	0.047369	1.000000	-0.193728	0.023890	0.004820	-0.041720	0.063523	0.041823
price	0.012688	-0.193728	1.000000	-0.044635	-0.104513	-0.017775	0.048636	0.415278
minimum_nights	0.004590	0.023890	-0.044635	1.000000	-0.059049	-0.122509	0.035466	0.025852
number_of_reviews	-0.047953	0.004820	-0.104513	-0.059049	1.000000	0.631005	-0.049656	0.040071
reviews_per_month	-0.041673	0.017720	-0.017775	-0.122509	0.631005	1.000000	-0.040116	0.053496
availability_365	-0.059941	0.063523	0.048636	0.035466	-0.049656	-0.040116	1.000000	0.065985
beds	0.017852	0.041822	0.415278	-0.025852	0.040071	0.053496	0.065985	1.000000

In [1100]

```
plt.figure(figsize=(8,5))
sns.pairplot(data=df, vars=['price', 'minimum_nights', 'number_of_reviews', 'availability_365'], hue='room_type')
plt.show()
```

Out[1100]

<Axes: >



