Section 1. Business Context

1.1 Context

Bangkok, as a globally renowned tourist destination, attracts millions of travelers each year. The city's Airbnb market plays a significant role in providing accommodation for tourists, offering diverse options ranging from budget-friendly shared spaces to premium entire apartments. However, understanding customer behavior and booking trends is critical for hosts to remain competitive in this dynamic market.

1.2 Problem Statements

How are customer behavior and booking trends across neighborhoods and review months in Bangkok's Airbnb listings influenced by price dynamics, availability patterns, and preferences for different types of rooms?

1.3 Key Objective

To analyze how price dynamics, availability patterns, and room type preferences influence customer behavior and booking trends across neighborhoods and review months in Bangkok's Airbnb listings, providing actionable insights for optimizing offerings.

Section 2. Data Understanding

```python
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import MarkerCluster
import geopandas as gpd

df = pd.read_csv(r'C:\Users\putri\OneDrive\Desktop\Capstone 2\Airbnb Listings Bangkok.csv')
df
```

```
       Unnamed: 0                  id  \
0               0               27934
1               1               27979
2               2               28745
3               3               35780
4               4              941865
...           ...                 ...
15849       15849  790465040741092826
15850       15850  790474503157243541
15851       15851  790475335086864240
15852       15852  790475546213717328
15853       15853  790476492384199044
```

```
                                                      name  
host_id  \
0                         Nice room with superb city view     120437   

1                          Easy going landlord,easy place     120541   

2                        modern-style apartment in Bangkok     123784   

3           Spacious one bedroom at The Kris Condo Bldg. 3     153730   

4                                 Suite Room 3 at MetroPoint     610315   

...                                                    ...        ...   

15849       素坤逸核心两房公寓42楼，靠近BTSon nut/无边天际泳池观赏曼谷夜景/出
门当地美食街  94899359
15850  Euro LuxuryHotel PratunamMKt TripleBdNrShopingArea  491526222   

15851   Euro LuxuryHotel PratunamMKt TwinBedNrShopingArea  491526222   

15852   Euro LuxuryHotel PratunamMKt TwinBedNrShopingArea  491526222   

15853   Euro LuxuryHotel PratunamMKt TwinBedNrShopingArea  491526222   


        host_name neighbourhood   latitude   longitude        
room_type  \
0          Nuttee   Ratchathewi  13.759830  100.541340  Entire 
home/apt   
1             Emy       Bang Na  13.668180  100.616740     Private 
room   
2      Familyroom     Bang Kapi  13.752320  100.624020     Private 
room   
3         Sirilak     Din Daeng  13.788230  100.572560     Private 
room   
4           Kasem     Bang Kapi  13.768720  100.633380     Private 
room   
...           ...           ...        ...         ...             ..
.   
15849       Renee       Pra Wet  13.715132  100.653458     Private 
room   
15850   Phakhamon   Ratchathewi  13.753052  100.538738     Private 
room   
15851   Phakhamon   Ratchathewi  13.753169  100.538700     Private 
room   
15852   Phakhamon   Ratchathewi  13.754789  100.538757     Private 
room   
15853   Phakhamon   Ratchathewi  13.752960  100.540820     Private 
room   
```

```
       price  minimum_nights  number_of_reviews last_review  \
0       1905               3                 65  2020-01-06
1       1316               1                  0         NaN
2        800              60                  0         NaN
3       1286               7                  2  2022-04-01
4       1905               1                  0         NaN
...      ...             ...                ...         ...
15849   2298              28                  0         NaN
15850   1429               1                  0         NaN
15851   1214               1                  0         NaN
15852   1214               1                  0         NaN
15853   1214               1                  0         NaN

       reviews_per_month  calculated_host_listings_count
availability_365  \
0                   0.50                               2
353
1                    NaN                               2
358
2                    NaN                               1
365
3                   0.03                               1
323
4                    NaN                               3
365
...                  ...                             ...
...
15849                NaN                               1
362
15850                NaN                              14
365
15851                NaN                              14
365
15852                NaN                              14
365
15853                NaN                              14
365

       number_of_reviews_ltm
0                          0
1                          0
2                          0
3                          1
4                          0
...                      ...
15849                      0
15850                      0
15851                      0
15852                      0
```

```
15853                         0

[15854 rows x 17 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15854 entries, 0 to 15853
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Unnamed: 0                      15854 non-null  int64  
 1   id                              15854 non-null  int64  
 2   name                            15846 non-null  object 
 3   host_id                         15854 non-null  int64  
 4   host_name                       15853 non-null  object 
 5   neighbourhood                   15854 non-null  object 
 6   latitude                        15854 non-null  float64
 7   longitude                       15854 non-null  float64
 8   room_type                       15854 non-null  object 
 9   price                           15854 non-null  int64  
 10  minimum_nights                  15854 non-null  int64  
 11  number_of_reviews               15854 non-null  int64  
 12  last_review                     10064 non-null  object 
 13  reviews_per_month               10064 non-null  float64
 14  calculated_host_listings_count  15854 non-null  int64  
 15  availability_365                15854 non-null  int64  
 16  number_of_reviews_ltm           15854 non-null  int64  
dtypes: float64(3), int64(9), object(5)
memory usage: 2.1+ MB
```

2.1 General Information

The Airbnb Bangkok dataset contains information about short-term rental listings in Bangkok, Thailand. The dataset is likely used for analyzing customer behavior, pricing strategies, room availability, and neighborhood trends.

2.2 Feature Information

Column Descriptions:

1. **id:** A unique identifier for each Airbnb listing.
2. **name:** The name or title of the listing.
3. **host_id:** A unique identifier for the host of the listing.
4. **host_name:** The name of the host.
5. **neighbourhood:** The neighborhood where the listing is located.
6. **latitude:** The latitude coordinate of the listing's location.
7. **longitude:** The longitude coordinate of the listing's location.
8. **room_type:** The type of room being offered:
    - **Entire home/apt:** An entire home or apartment.

- **Private room:** A private room within a shared space.
- **Shared room:** A shared room with other guests.
- **Hotel room:** A room in a hotel.

9. **price:** The daily price of the listing.
10. **minimum_nights:** The minimum number of nights required for a booking.
11. **number_of_reviews:** The total number of reviews the listing has received.
12. **last_review:** The date of the most recent review.
13. **calculated_host_listings_count:** The total number of listings the host has.
14. **availability_365:** The number of days the listing is available for booking in the next 365 days.
15. **number_of_reviews_ltm:** The number of reviews the listing has received in the last 12 months.

Key Points:

- The `latitude` and `longitude` coordinates are in the WGS84 projection.
- The `room_type` column categorizes listings into four main types with specific definitions.
- The `availability_365` column indicates the availability of the listing for the next year, taking into account both bookings and host-imposed restrictions.

2.3 Statistics Summary

```
pd.set_option('display.max_colwidth', None)
data = []
for col in df.columns:
    data.append([col, df[col].nunique(), df[col].unique()])

bkk = pd.DataFrame(data, columns=['Column Name', 'Number of Unique',
'Unique Sample'])
bkk
```

	Column Name	Number of Unique \
0	Unnamed: 0	15854
1	id	15854
2	name	14794
3	host_id	6659
4	host_name	5312
5	neighbourhood	50
6	latitude	9606
7	longitude	10224
8	room_type	4
9	price	3040
10	minimum_nights	86
11	number_of_reviews	298
12	last_review	1669
13	reviews_per_month	513
14	calculated_host_listings_count	50
15	availability_365	366

```
16           number_of_reviews_ltm               85


Unique Sample
0
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, ...]
1
[27934, 27979, 28745, 35780, 941865, 1704776, 48736, 1738669, 1744248,
952677, 55681, 1765918, 55686, 59221, 959254, 62217, 1791481, 66046,
105042, 1793000, 960858, 113744, 965722, 1808600, 118118, 1816517,
969792, 121410, 145343, 973830, 156583, 1823321, 159854, 976690,
978531, 166267, 169285, 978969, 1842066, 169514, 1849029, 1862089,
985743, 988373, 172332, 1016487, 1862331, 1862377, 185364, 1887544,
1888303, 1019241, 241416, 1026451, 1028469, 1028486, 1035589, 1035640,
1897982, 296960, 1898332, 1041976, 313459, 1052180, 1926489, 320014,
1933894, 1057173, 1060320, 384924, 1067748, 1077493, 1943048, 385130,
385278, 385979, 390611, 1947314, 1079039, 1086843, 393066, 397449,
405662, 1088343, 1094136, 1961981, 407381, 1975849, 1133843, 413824,
428360, 428421, 428907, 428950, 430691, 430703, 430706, 432004,
439051, 1138679, ...]
2   [Nice room with superb city view, Easy going landlord,easy place,
modern-style apartment in Bangkok, Spacious one bedroom at The Kris
Condo Bldg. 3, Suite Room 3 at MetroPoint, NEw Pro!!  Bungalow Bkk
Centre, Condo with Chaopraya River View, 1 chic bedroom apartment in
BKK, Batcave, Pool view, near Chatuchak, Standard Room Decor do
Hostel, Sathorn Terrace Apartment(61), 2BR apt in a cozy neighborhood,
Comfy bedroom near River pier & BTS  Taksin., budget hotel bangkok
near subway, Deluxe Condo, Nana, Pool/GYM/Sauna,
Luxury@swimpool/FreeWiFi/nearJJMkt, Nice and Quiet condo near BTS
Onnut, 24Flr- 1br Apt near JJ, MRT, BTS, Central Bangkok 3 Bedroom
Apartment, The Duplex - Asoke- Luxury 92sqm, New, Stylish & Luxury
Studio Condo, River View - Ivy Condo (1 Bedroom), Siamese Gioia on
Sukhumvit 31, Contemporary Modern Duplex-Thong Lo, Pan Dao Condo 5 min
from BTS On Nut, 1 BR condominium center BKK +NETFLIX+55SQM, 1
penthouse in central Bangkok, MetroPoint Suite Room, Near Airport,
Boutique Rooms Near Bangkok Airport, BangLuang House1 @ Bangkok
Thailand, Studio near Chula University/Silom walk to MRT/BTS, กรองทอง
แมนชั่น (ลาดพร้าว 81), Deluxe one Bedroom Condo w.Pool-GYM & Sauna 8-7,
Beautiful 1 BR apartment @BTS Ari, Urban Oasis in the heart of
Bangkok, 1Bed apt. near Chula University/Silom, Stay at the ROARING
RATCHADA!, 60 m2 apartment in Thong Lor, Bangkok, ICONSIAM River view
on 49th floor, 2br apt in Sukhumvit Asoke near BTS, Self catering
cozy1-bed near BTS, ❂☀☀☀Perfect Escape☀☀☀Sunny Roof EnSuite☀☀☀☀, Room
with city view of BKK, BangLuang House 2@ Bangkok Thailand,
```

```
Tranquility found in busy Bangkok near new skytran, Private room in
Bangkok, ☞✪✪✪✪Roomy Studio 4 Family r friends✪No Stairs✪✪✪✪, ☞Downtown
Central Studio-Bangkok MRT, Beautiful Wood Bangkok Resort House,
"Serviced 2 Bed Scenic SkyVillas", Cozy 1BR rooftop (BTS Ploenchit)
heart of bangkok, Chic two bedroom for Monthly rental, Sukhumvit52
near SkyTrain to BkkCBD, ♡Chic Studio, Easy Walk to Pier & BTS
Taksin♡, One Bedroom Suite- WIFI- SATHORN, STUDIO RM2 - WIFI- SATHORN,
Quiet Double  Bed Apartment, Quiet Double Bed Apartment, Suvarnabhumi
free transfer, Luxury&Comfy wthWifi walk-distance to Subwy-Malls, Apr.
for rent full fur 1 bedroom, monthly, Long-stay special rate spacious
entire floor Siam, One Bed Room at Sukumvit 50 Bangkok, City View,
relaxed theme & delicious food around, Ideo Blucove Sukhumvit Bangkok,
2-BR condo near BTS on Sukhumvit Rd, NewlyRenovated! 3Br,SingleHouse,
Park/BTS/Airport., IdeoMix, Sukhumvit RD, close to BTS, Mix Dorm
Decor do Hostel, Oasis in the heart of Bangkok, 5 mins by car from
Chong Nonsi BTS Station, Inn Saladaeng - Superior hotel room, Best nr
Chatujak, MRT, BTS free wifi&fNetflix, ❂Citycenter✔Subway
station✔Private Bathroom4Aircon, Nice River View Condominium 30 sq.m,
Monthly rent 2Beds/2Baths quiet APT at BTS, Sukhumvit apartment near
Nana BTS, A room w/ the view :-) in the city, Spacious 1Bed apartment,
Near Bangkok more space than urban!, ✺✺99 feet in the sky✺✺, Cozy
Studio Apt near Skytrain.(72/74), Asoke: tasteful, modern 1BR condo, 2
bed 2 bath, BTS, Supermarkets, Monthly, Private, relaxed with
amenities, S1 hostel (Dorm) Sathorn Bangkok, 3 minutes walk to Phrom
Phong BTS, 1 BDM CONDO SAPHAN KWAI/ARI walk to JJ/BTS/MRT, เฮ้าส์โหมด
House Mode, ❂100% Private&Central Light EnSuite, Spacious Studio
kitchen/wifi, 2. Bangkok bright Apartment 201, 1.Bangkok great value
Studio WIFI, BKK City Fab Luxx Studio free wifi @1194, 5. Bangkok
Bright Apartment -WIFI, 6. Bangkok nice, cosy Apartment 201, 7.
Bangkok big bright Apartment 402, STUDIO-WIFI-RAIN SHOWER-SATHORN,
Luxury Riverview Teakwood Apartment-Great Views :), 1 Bed Pool Access
Onnut BTS, ...]
3
[120437, 120541, 123784, 153730, 610315, 2129668, 222005, 7045870,
9181769, 5171292, 263049, 9279712, 284095, 5153476, 302658, 9399478,
323158, 545890, 9407280, 3769704, 578110, 5265861, 9478184, 596463,
8492603, 5297436, 703944, 5325919, 58920, 9545111, 766443, 5344120,
5309669, 806600, 5358785, 9626074, 729617, 9652914, 1927968, 822284,
5594281, 8214044, 889670, 6132593, 9821998, 3323622, 960892, 3346331,
2148220, 4115838, 8362130, 175729, 4837310, 5735895, 1611711, 5793490,
9434109, 843854, 9509219, 5822937, 1667508, 4154759, 5929404, 9906827,
1928343, 1681901, 807406, 10070953, 5935474, 4937984, 1425515,
5981006, 10138065, 1463818, 8480912, 6220137, 2122791, 4877320,
10204823, 2592798, 10222460, 10246374, 7454779, 8664261, 6262801,
6313832, 1513875, 5402740, 2625384, 6586465, 9390848, 2864425,
10581237, 1780407, 6647138, 6648722, 2940438, 3533863, 3687435,
5469059, ...]
4
[Nuttee, Emy, Familyroom, Sirilak, Kasem, Wimonpak, Athitaya,
```

```
Jiraporn, Nol, Somsak, Tor, Jing, Mimi, Natcha, Srisuk, Piyakorn, Sue,
Henry, Timo, Pat, Muay, Chuchart, Shine, Dustin, Sudhichai, Anya,
Parinya, วสวัตติ์, Gael, Penjit, Gerd, Nattavut, Apiradee, Frances,
Danny, Weera, Kanchuya, Jirasak, Evan, Rae And Charlie, Yodying, Evan
From Sanctuary House, Narumon, Salvatore, Pichanee, Phoebe, Vajirune,
Bee, Marvin, Primrose, Luckana, Mitch & Mam, Veesa, Pariya, Nichapat,
Nicky, Sander, Anshera, Piya, Siriwipa, Inn Saladaeng & The Sathon
Vimanda, Nokiko, Chanvit, Pornpan, Hollis, Vichit, Tisa, Sugarcane,
Peter, Sibyl, S1, Amporn, Chris And Lek, Prapussorn, Maam & Hermann,
Nisa, Jahidul, Nokina, Preeda, Arika, Lily Duangdao, Kriengkrai,
Andrea, Psirivedin, Suchada, Nattha, Mike, Tayawat, VeeZa, Urcha,
Anchana, Feb, NiNew, Taweewat  (Ken), Kinifrog, Sarasinee, Avinash,
Andrew, Tam, Egidio, ...]
5
[Ratchathewi, Bang Na, Bang Kapi, Din Daeng, Bang Kho laen, Rat
Burana, Chatu Chak, Khlong San, Bang Rak, Phaya Thai, Sathon, Khlong
Toei, Vadhana, Sai Mai, Lat Krabang, Bangkok Yai, Wang Thong Lang,
Huai Khwang, Phasi Charoen, Bang Sue, Nong Chok, Phra Khanong, Thawi
Watthana, Parthum Wan, Pra Wet, Phra Nakhon, Thon buri, Yan na wa,
Suanluang, Don Mueang, Dusit, Lak Si, Samphanthawong, Bueng Kum, Bang
Phlat, Saphan Sung, Min Buri, Khan Na Yao, Khlong Sam Wa, Bang Khen,
Lat Phrao, Chom Thong, Bangkok Noi, Pom Prap Sattru Phai, Nong Khaem,
Thung khru, Bang Khae, Bang Khun thain, Taling Chan, Bang Bon]
6
[13.75983, 13.66818, 13.75232, 13.78823, 13.76872, 13.69757, 13.68556,
13.82925, 13.81693, 13.7204, 13.71934, 13.77486, 13.71802, 13.77941,
13.71516, 13.79152, 13.70719, 13.82298, 13.73378, 13.74668, 13.9077,
13.68568, 13.74444, 13.72097, 13.70441, 13.75351, 13.7547, 13.76747,
13.721868, 13.73292, 13.7285, 13.78938, 13.74293, 13.77931, 13.72291,
13.72733, 13.78118, 13.73224, 13.72287, 13.74464, 13.7137, 13.72062,
13.71803, 13.73122, 13.83148, 13.82148, 13.72073, 13.72063, 13.779,
13.72096, 13.73782, 13.72687, 13.70169, 13.71192, 13.71602, 13.71798,
13.79274, 13.79315, 13.72141, 13.80926, 13.67805, 13.74814, 13.71513,
13.69947, 13.67998, 13.7281, 13.70004, 13.67991, 13.72214, 13.74902,
13.71498, 13.72825, 13.81694, 13.68426, 13.71905, 13.74052, 13.71012,
13.75224, 13.75135, 13.71782, 13.73816, 13.7104, 13.69949, 13.72157,
13.73675, 13.79128, 13.72646255738608, 13.69673, 13.69935, 13.69977,
13.698, 13.70068, 13.69925, 13.70086, 13.71922, 13.67317, 13.71119,
13.70497, 13.69832, 13.70218, ...]
7
[100.54134, 100.61674, 100.62402, 100.57256, 100.63338, 100.5288,
100.49535, 100.56737, 100.56433, 100.50757, 100.5176, 100.54272,
100.51539, 100.57383, 100.56806, 100.53982, 100.59936, 100.56484,
100.56303, 100.56137, 100.64473, 100.49231, 100.57003, 100.57823,
100.59968, 100.53308, 100.53268, 100.63287, 100.771713, 100.46413,
100.52313, 100.6134, 100.55603, 100.54262, 100.53759, 100.52555,
100.58349, 100.57803, 100.51678, 100.55784, 100.59637, 100.54707,
100.54654, 100.46228, 100.52102, 100.58326, 100.5469, 100.54694,
100.83671, 100.52911, 100.55179, 100.52725, 100.5977, 100.51535,
```

```
100.52663, 100.52841, 100.33377, 100.33356, 100.72946, 100.56892,
100.62451, 100.52016, 100.5683, 100.52726, 100.61074, 100.57318,
100.6787, 100.61055, 100.50752, 100.55652, 100.53312, 100.53774,
100.56451, 100.49841, 100.50414, 100.55237, 100.60281, 100.58979,
100.49447, 100.51622, 100.56495, 100.60171, 100.54509, 100.57191,
100.54935, 100.49129336748284, 100.53202, 100.52723, 100.52652,
100.52886, 100.5228, 100.52624, 100.52276, 100.5255, 100.54406,
100.63334, 100.55134, 100.52434, 100.52254, 100.52806, ...]
8
[Entire home/apt, Private room, Hotel room, Shared room]
9
[1905, 1316, 800, 1286, 1000, 1558, 1461, 700, 1150, 1893, 1862, 910,
1400, 4156, 1577, 122594, 5680, 5034, 1500, 1385, 3775, 2078, 1732,
2000, 3082, 1190, 1329, 1176, 600, 1659, 5429, 1843, 1870, 2500, 1300,
1200, 3500, 1795, 350, 1450, 8658, 3757, 1490, 2701, 2251, 866, 2696,
1736, 1800, 900, 400, 2900, 2226, 890, 3394, 922, 1543, 1589, 1271,
1747, 797, 6926, 1169, 5195, 829, 950, 2355, 980, 330, 3740, 1143,
831, 790, 720, 1211, 970, 929, 670, 1004, 811, 1629, 835, 926, 650,
5887, 1250, 2571, 3847, 1485, 2814, 707, 2061, 750, 693, 1088, 808,
500, 3097, 850, 1212, ...]
10
[3, 1, 60, 7, 250, 2, 15, 30, 28, 21, 27, 4, 180, 90, 5, 358, 1125,
29, 14, 200, 365, 120, 9, 12, 300, 360, 100, 10, 45, 23, 6, 84, 370,
24, 31, 50, 19, 20, 75, 8, 25, 40, 26, 59, 58, 170, 399, 998, 13, 22,
356, 16, 183, 700, 150, 35, 355, 500, 89, 80, 18, 11, 85, 135, 198,
88, 160, 109, 148, 51, 1115, 113, 62, 450, 270, 55, 208, 1000, 17,
999, 400, 99, 1095, 39, 190, 364]
11
[65, 0, 2, 19, 1, 10, 4, 27, 129, 208, 3, 78, 9, 148, 287, 83, 76, 28,
63, 212, 90, 71, 5, 15, 7, 419, 95, 21, 268, 310, 118, 11, 104, 472,
43, 56, 295, 29, 67, 170, 12, 94, 40, 36, 293, 396, 80, 189, 430, 39,
32, 18, 82, 325, 35, 14, 127, 26, 48, 8, 120, 60, 38, 24, 404, 6, 109,
34, 206, 271, 93, 77, 426, 224, 147, 33, 37, 58, 135, 81, 117, 146,
53, 101, 25, 97, 20, 88, 256, 278, 167, 91, 31, 113, 192, 156, 92, 75,
103, 252, ...]
12
[2020-01-06, nan, 2022-04-01, 2017-08-03, 2014-02-03, 2016-03-29,
2019-12-27, 2019-01-03, 2022-09-30, 2019-12-02, 2018-07-30, 2019-05-
31, 2020-03-04, 2020-01-07, 2022-11-22, 2018-12-24, 2018-09-12, 2013-
10-09, 2018-04-05, 2022-11-25, 2019-12-31, 2022-12-08, 2022-11-15,
2018-12-18, 2022-11-30, 2019-11-10, 2015-12-08, 2022-12-07, 2022-12-
15, 2019-02-20, 2022-12-11, 2018-07-02, 2020-02-20, 2014-11-12, 2022-
10-25, 2020-03-03, 2019-08-04, 2021-08-21, 2022-12-18, 2022-09-20,
2022-11-28, 2022-12-05, 2020-02-08, 2022-08-31, 2022-12-09, 2022-09-
01, 2022-12-06, 2020-04-30, 2020-01-28, 2019-01-13, 2022-10-01, 2022-
10-31, 2020-01-04, 2022-11-26, 2019-12-09, 2022-09-14, 2020-03-14,
2019-03-04, 2018-10-24, 2013-11-01, 2022-12-14, 2016-11-05, 2022-11-
13, 2013-06-28, 2017-02-23, 2020-02-28, 2013-07-01, 2022-10-30, 2018-
03-31, 2022-12-12, 2018-07-22, 2022-12-16, 2022-10-23, 2019-04-13,
```

```
2019-01-01, 2019-01-22, 2021-03-15, 2021-05-17, 2022-12-20, 2013-12-
03, 2015-09-02, 2015-09-01, 2016-04-08, 2022-12-13, 2019-01-04, 2012-
12-15, 2015-10-18, 2022-12-04, 2020-01-10, 2022-06-17, 2022-06-19,
2020-03-15, 2015-01-01, 2018-12-05, 2018-04-30, 2022-11-18, 2022-11-
11, 2022-10-07, 2013-08-18, 2017-06-19, ...]
13
[0.5, nan, 0.03, 0.17, 0.01, 0.09, 0.19, 1.17, 1.44, 0.02, 0.78, 1.08,
2.59, 0.05, 0.75, 0.7, 0.28, 0.64, 0.47, 1.58, 0.77, 0.54, 0.14, 0.06,
3.77, 0.8, 3.0, 1.07, 0.15, 0.89, 4.02, 0.04, 0.46, 0.39, 0.43, 2.61,
0.27, 0.51, 1.54, 0.12, 0.33, 2.72, 3.02, 0.62, 1.45, 3.39, 0.3, 0.25,
0.18, 0.83, 2.67, 0.31, 0.11, 1.0, 0.23, 0.38, 0.07, 0.93, 0.4, 3.67,
1.15, 0.41, 0.32, 1.83, 2.97, 0.95, 0.1, 1.84, 1.18, 1.03, 0.13, 0.53,
1.79, 1.19, 0.72, 1.02, 1.27, 0.2, 0.92, 0.22, 0.21, 0.08, 2.13, 2.31,
0.26, 0.29, 1.2, 1.78, 0.81, 0.96, 1.22, 2.11, 0.37, 0.71, 0.24, 1.36,
1.98, 1.12, 1.96, 2.12, ...]
14
[2, 1, 3, 41, 10, 7, 6, 4, 37, 8, 19, 5, 53, 45, 13, 11, 25, 24, 36,
29, 18, 12, 9, 15, 44, 33, 39, 21, 34, 89, 32, 56, 62, 23, 14, 22, 17,
28, 16, 31, 20, 26, 228, 48, 99, 27, 30, 49, 40, 35]
15
[353, 358, 365, 323, 87, 320, 356, 361, 330, 180, 334, 349, 364, 55,
263, 350, 95, 207, 336, 174, 156, 331, 88, 355, 363, 339, 145, 134,
16, 0, 242, 256, 59, 167, 219, 142, 149, 176, 129, 230, 301, 120, 75,
44, 270, 346, 272, 162, 347, 359, 304, 62, 82, 342, 348, 130, 154,
244, 344, 354, 317, 54, 362, 271, 255, 144, 357, 181, 236, 127, 146,
124, 221, 294, 13, 318, 56, 267, 293, 107, 360, 314, 316, 89, 57, 312,
70, 179, 10, 338, 86, 302, 321, 98, 217, 341, 90, 325, 333, 1, ...]
16
[0, 1, 3, 13, 2, 7, 5, 10, 9, 12, 29, 4, 19, 56, 20, 11, 6, 14, 8, 43,
18, 30, 15, 277, 26, 59, 21, 41, 16, 22, 25, 38, 42, 40, 31, 39, 35,
44, 17, 27, 36, 23, 79, 50, 24, 34, 47, 37, 32, 73, 48, 28, 45, 67,
46, 147, 109, 68, 62, 51, 72, 52, 49, 33, 69, 325, 55, 146, 61, 124,
75, 71, 138, 57, 70, 90, 65, 141, 246, 118, 80, 53, 63, 60, 101]
```

Section 3. Data Cleaning

3.1 Missing Values

```
df.isna().sum()/df.shape[0]*100
```

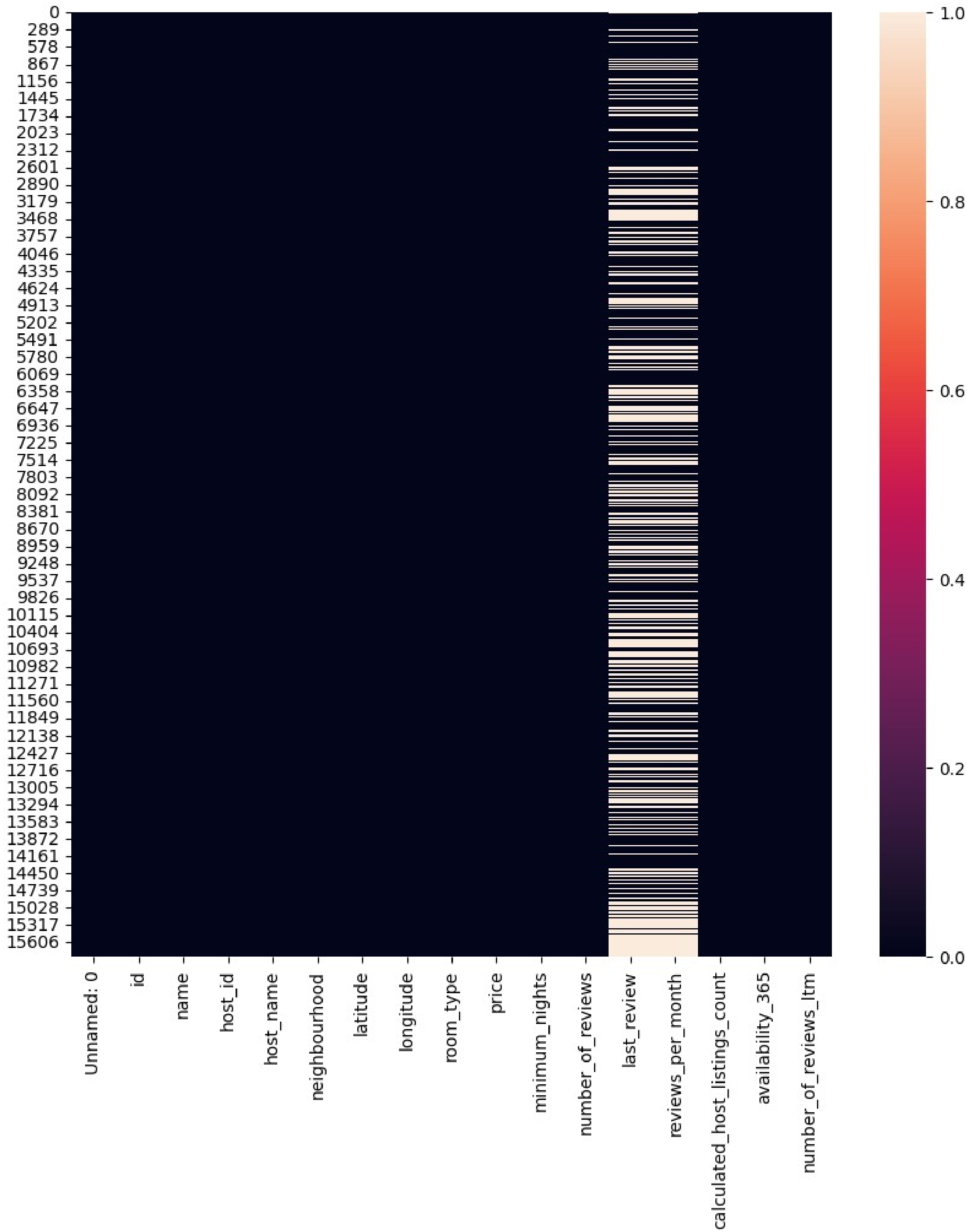
```
Unnamed: 0                           0.000000
id                                   0.000000
name                                 0.050460
host_id                              0.000000
host_name                            0.006308
neighbourhood                        0.000000
latitude                             0.000000
```

```
longitude                             0.000000
room_type                             0.000000
price                                 0.000000
minimum_nights                        0.000000
number_of_reviews                     0.000000
last_review                          36.520752
reviews_per_month                    36.520752
calculated_host_listings_count        0.000000
availability_365                      0.000000
number_of_reviews_ltm                 0.000000
dtype: float64
```

36% missing value in last review and reviews per month 0.5% missing value in name 0.006% missing value in host_name

```
plt.figure(figsize=(10,10))
sns.heatmap(df.isna())
```

```
<Axes: >
```




```
df[df['host_id']==73275200]
```

```
      Unnamed: 0        id               name   host_id host_name  \
1981        1981  13400326  Errday Guest House  73275200  Pakaphol
1982        1982  13400758  Errday Guest House  73275200  Pakaphol
2075        2075  13142743                 NaN  73275200  Pakaphol

     neighbourhood  latitude  longitude     room_type  price
minimum_nights  \
1981   Khlong Toei  13.72427  100.56443  Private room    950
1
1982   Khlong Toei  13.72373  100.56415  Private room  36363
1
2075   Khlong Toei  13.72566  100.56416  Private room    850
1

      number_of_reviews last_review  reviews_per_month  \
1981                  1  2020-02-19               0.03
1982                  0         NaN                NaN
2075                  2  2017-12-11               0.03

      calculated_host_listings_count  availability_365
number_of_reviews_ltm
1981                               3                 1
0
1982                               3                 1
0
2075                               3               220
0
```

if we saw the missing value in name, we can see that there is a same host_id it means we can fill the missing value with the same name in host_id

```
df.loc[2075, 'name'] = 'Errday Guest House'
df.loc[2075]
```

```
Unnamed: 0                                      2075
id                                          13142743
name                              Errday Guest House
host_id                                     73275200
host_name                                   Pakaphol
neighbourhood                            Khlong Toei
latitude                                    13.72566
longitude                                  100.56416
room_type                               Private room
price                                            850
minimum_nights                                     1
number_of_reviews                                  2
last_review                               2017-12-11
reviews_per_month                               0.03
calculated_host_listings_count                     3
```

```
availability_365                                  220
number_of_reviews_ltm                               0
Name: 2075, dtype: object
```

```
host_null = df[df['host_name'].isna()]
host_null
```

```
      Unnamed: 0        id            name    host_id host_name 
neighbourhood  \
3571        3571  19682464  Cozy Hideaway  137488762       NaN 
Bang Kapi

      latitude  longitude     room_type  price  minimum_nights  \
3571  13.76999  100.63769  Private room   1399               3

      number_of_reviews last_review  reviews_per_month  \
3571                  1  2017-07-29               0.02

      calculated_host_listings_count  availability_365 
number_of_reviews_ltm
3571                               1               365 
0
```

because for this host_name there is no similiar host_id so we can drop this

```
df = df[df['host_id']!=137488762]
```

```
df[df['price'] == 0]
```

```
       Unnamed: 0        id                           name    host_id 
\
11103       11103  44563108  Somerset Maison Asoke Bangkok  360620448


                   host_name neighbourhood  latitude  longitude 
room_type  \
11103  Somerset Maison Asoke       Vadhana  13.73815   100.5642  Hotel 
room

       price  minimum_nights  number_of_reviews last_review  \
11103      0               1                  0         NaN

       reviews_per_month  calculated_host_listings_count 
availability_365  \
11103                NaN                               1 
0

       number_of_reviews_ltm
11103                      0
```

```
df = df[df['price'] != 0]
```

drop the price that value is 0, because is only one value

```
df = df[df['name'].notna()]
```

extract the data that name is not null value

```
last_review = df[df['last_review'].isnull()]
zero_reviews = df[df['number_of_reviews'] == 0]
print(last_review.shape == zero_reviews.shape)
```

```
True
```

```
df[['reviews_per_month']] = df[['reviews_per_month']].fillna(0)
```

fill the null value in reviews_per_month with 0 because its related to number_of_review. if the number of review 0, automaticaly the reviews_per_month will be 0

```
df.isna().sum()
```

```
Unnamed: 0                           0
id                                   0
name                                 0
host_id                              0
host_name                            0
neighbourhood                        0
latitude                             0
longitude                            0
room_type                            0
price                                0
minimum_nights                       0
number_of_reviews                    0
last_review                       5783
reviews_per_month                    0
calculated_host_listings_count       0
availability_365                     0
number_of_reviews_ltm                0
dtype: int64
```

```
df['last_review'] = pd.to_datetime(df['last_review'], errors='coerce')
df['review_year'] = df['last_review'].dt.year

dfc = df[df['review_year'] == 2022].copy()
df.loc[df['review_year'] == 2022, 'last_review'] =
pd.to_datetime(df.loc[df['review_year'] == 2022, 'last_review'],
errors='coerce')
df.loc[df['review_year'] == 2022, 'review_year'] =
df.loc[df['review_year'] == 2022, 'last_review'].dt.year
dfc
```

```
       Unnamed: 0                   id  \
3               3                35780
11             11              1765918
19             19              1793000
28             28               145343
30             30               156583
...           ...                  ...
15712       15712  785741287659406453
15728       15728  785976692600131294
15743       15743  786248090308669514
15744       15744  786318268883527580
15796       15796  788841933134248110

                                                    name
host_id  \
3          Spacious one bedroom at The Kris Condo Bldg. 3     153730

11                        2BR apt in a cozy neighborhood    9279712

19                      The Duplex - Asoke- Luxury 92sqm    9407280

28                   Boutique Rooms Near Bangkok Airport     703944

30     Studio near Chula University/Silom walk to MRT/BTS      58920

...                                                  ...        ...

15712                         ใจกลางเมืองติดห้างไอคอนสยาม 200814460
15728       1br/Free pool&gym/WIFI-Asok/SukhumvitBTS! 2PP  485536928

15743                  Vibrant Luxe 2 Bedroom | Thong Lor   46163812

15744                  Vibrant Luxe 2 Bedroom | Thong Lor   46163812

15796  Stunning river view in the heart of BKK 5min/train  315867023


      host_name neighbourhood   latitude   longitude        room_type
price  \
3       Sirilak     Din Daeng  13.788230  100.572560     Private room
1286
11         Jing    Phaya Thai  13.774860  100.542720  Entire home/apt
1893
19         Timo       Vadhana  13.746680  100.561370  Entire home/apt
5034
28      Parinya   Lat Krabang  13.721868  100.771713     Private room
1329
30         Gael      Bang Rak  13.728500  100.523130  Entire home/apt
1176
...         ...           ...        ...         ...              ...
...
```

```
15712       Noi     Thon buri  13.696506  100.486226  Entire home/apt
2000
15728     Lucas   Khlong Toei  13.734856  100.557960  Entire home/apt
2514
15743    Ernest       Vadhana  13.730126  100.586369  Entire home/apt
3932
15744    Ernest       Vadhana  13.729880  100.586269  Entire home/apt
4285
15796      Alex      Bang Rak  13.719792  100.515910  Entire home/apt
3304

       minimum_nights  number_of_reviews last_review
reviews_per_month  \
3                   7                  2  2022-04-01
0.03
11                 15                129  2022-09-30
1.17
19                 21                287  2022-11-22
2.59
28                  1                 28  2022-11-25
0.28
30                  7                 63  2022-11-25
0.47
...               ...                ...         ...                  ..
.
15712               1                  1  2022-12-25
1.00
15728               1                  1  2022-12-26
1.00
15743               1                  3  2022-12-24
3.00
15744              28                  3  2022-12-28
3.00
15796               2                  2  2022-12-28
2.00

       calculated_host_listings_count  availability_365  \
3                                   1               323
11                                  1               356
19                                  1               349
28                                  1               349
30                                  2                95
...                               ...               ...
15712                               2               361
15728                               4               257
15743                               8               349
15744                               8               365
15796                               3               342

       number_of_reviews_ltm  review_year
```

```
3                              1        2022.0
11                             1        2022.0
19                             3        2022.0
28                            13        2022.0
30                             2        2022.0
...                          ...           ...
15712                          1        2022.0
15728                          1        2022.0
15743                          3        2022.0
15744                          3        2022.0
15796                          2        2022.0

[6628 rows x 18 columns]
```

Since 2023 booking availability is represented by the `available_365` column, I will only concentrate on 2022 data to ensure relevancy. Since it doesn't fairly represent the current patterns, using data from prior years could distort the analysis. Furthermore, as the dataset only includes one item from 2012, it is better to omit it in order to obtain insightful information.

3.2 Duplicated Values

```
dfc[dfc['id'].duplicated()]

Empty DataFrame
Columns: [Unnamed: 0, id, name, host_id, host_name, 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, review_year]
Index: []
```

no duplicate data

```
dfc['price'].max()

np.int64(1014758)

dfc[dfc['price'] == 1014758]

       Unnamed: 0                   id  \
12300       12300  562972065309061724

                                                      name
host_id  \
12300  3B中文No Guest Service Fee@Nana Asok/Soi11 Nightlife  131427125


      host_name neighbourhood  latitude  longitude        room_type
price  \
12300        Jj       Vadhana  13.74666   100.5591  Entire home/apt
```

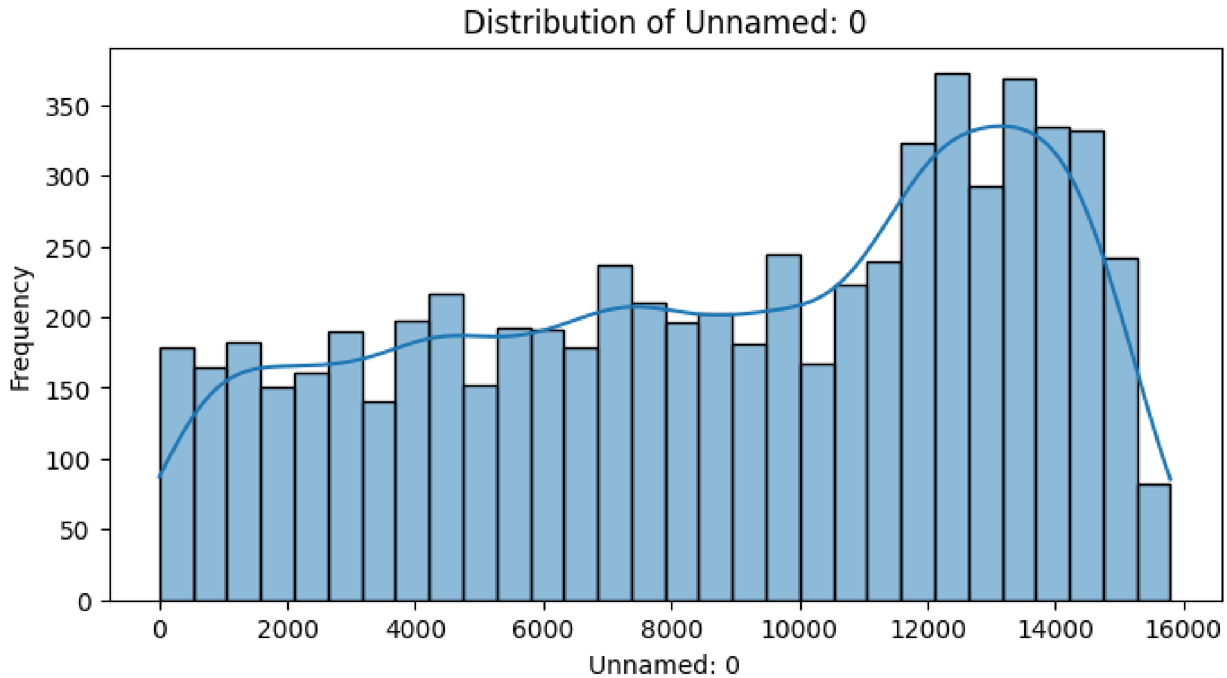
```
1014758

       minimum_nights  number_of_reviews last_review
reviews_per_month  \
12300              30                  2  2022-09-17
0.32

       calculated_host_listings_count  availability_365  \
12300                              10                75

       number_of_reviews_ltm  review_year
12300                      2       2022.0

dfc = dfc.drop(dfc[dfc['price'] > 10000].index)
```

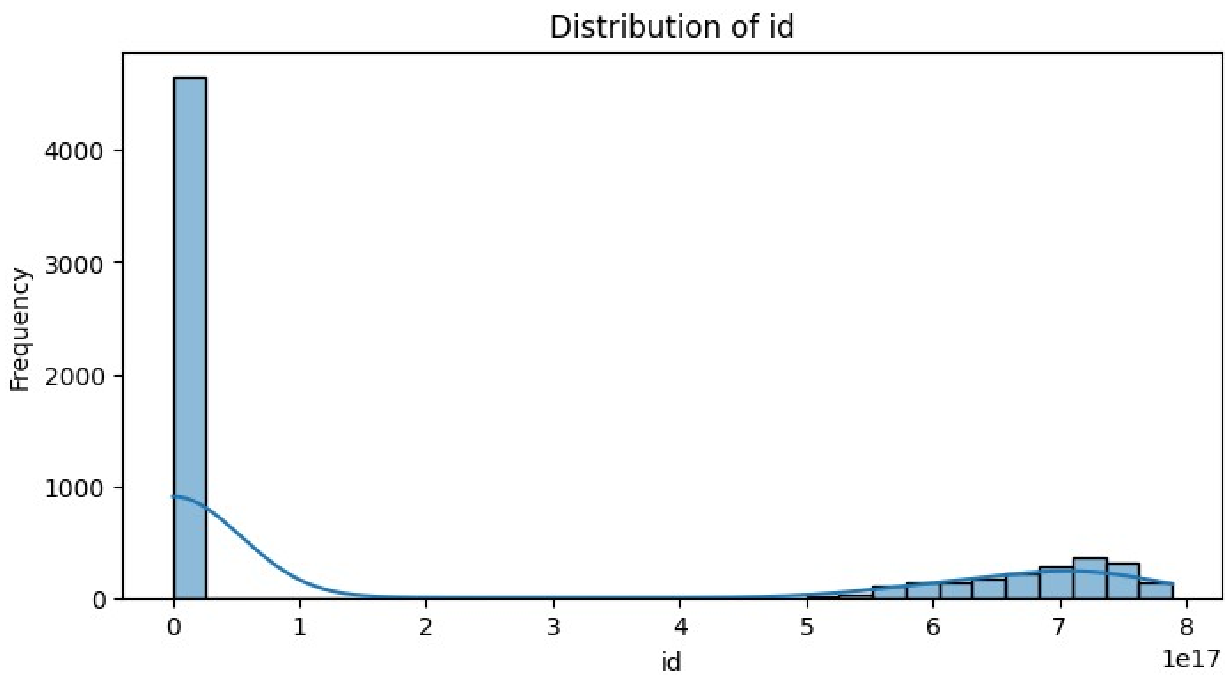
assume that price above THB 10000 is outlier

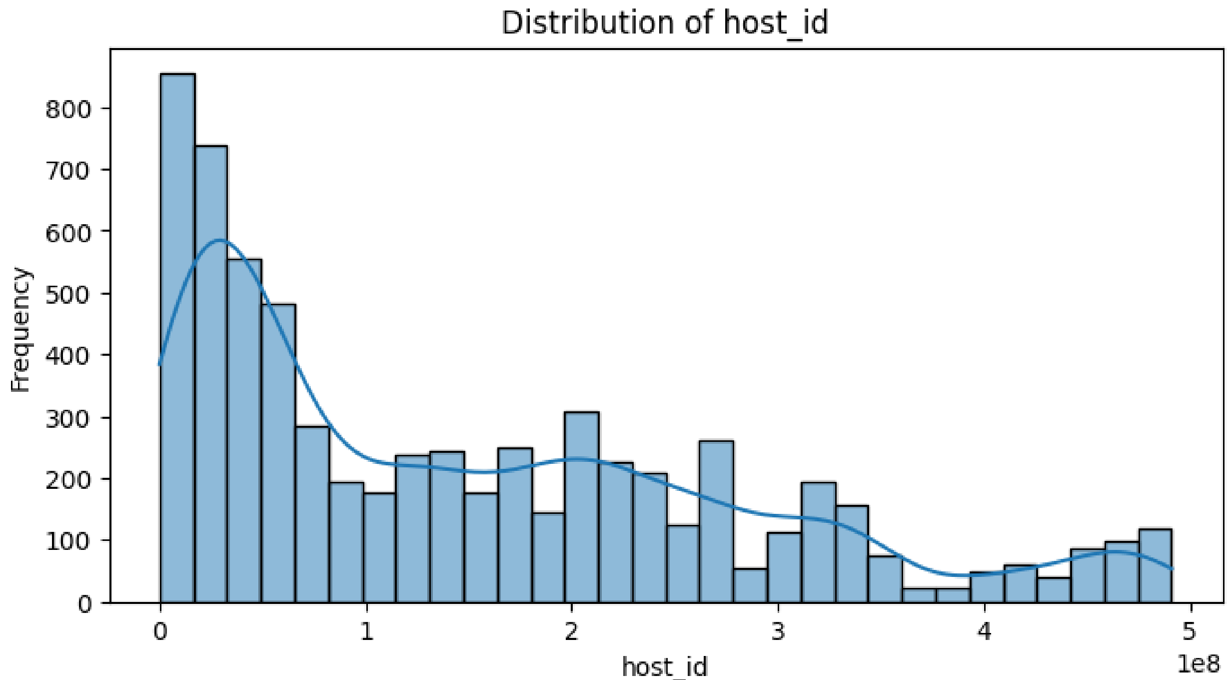
```
dfc[dfc['price'] > 10000]

Empty DataFrame
Columns: [Unnamed: 0, id, name, host_id, host_name, 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, review_year]
Index: []
```

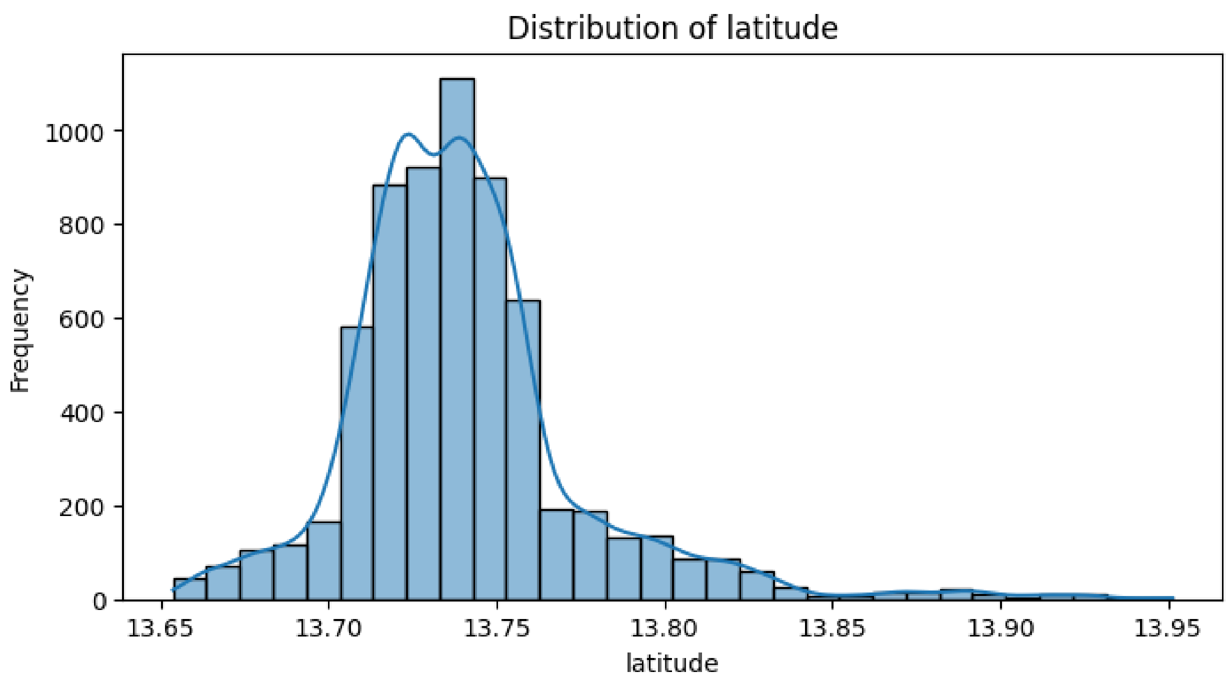
Section 4. Analytics

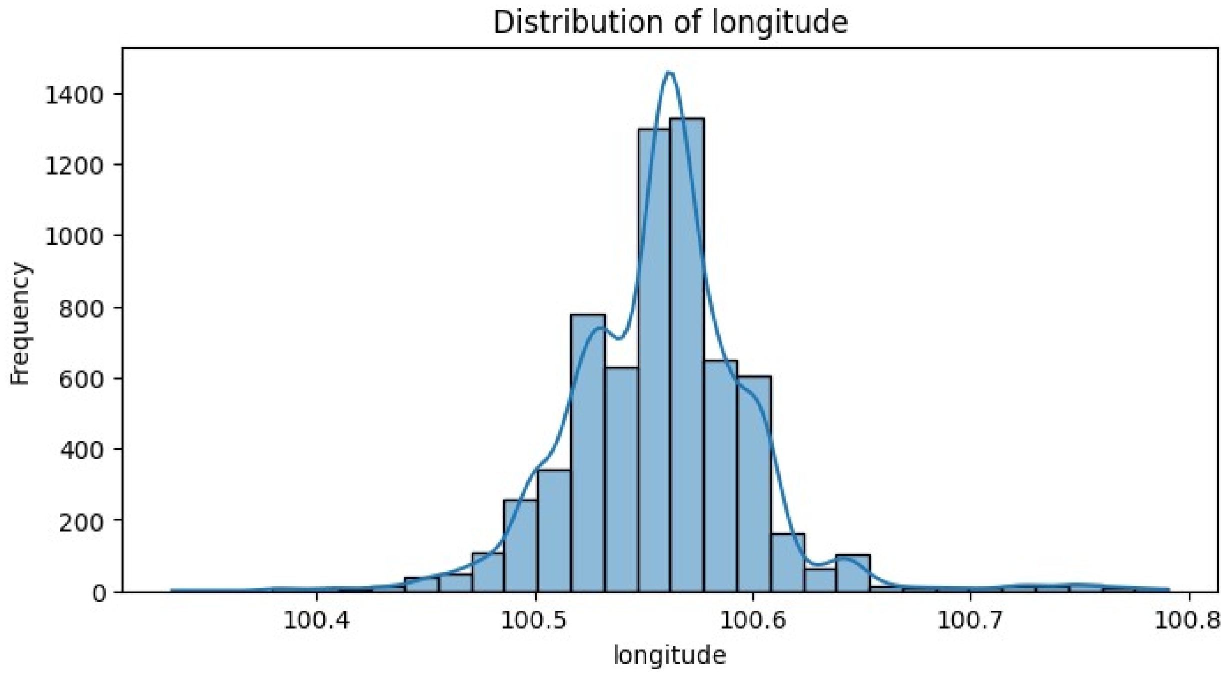
```
numeric_df = dfc.select_dtypes(include=['number'])

for column in numeric_df.columns:
    plt.figure(figsize=(8, 4))
    sns.histplot(numeric_df[column], kde=True, bins=30)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

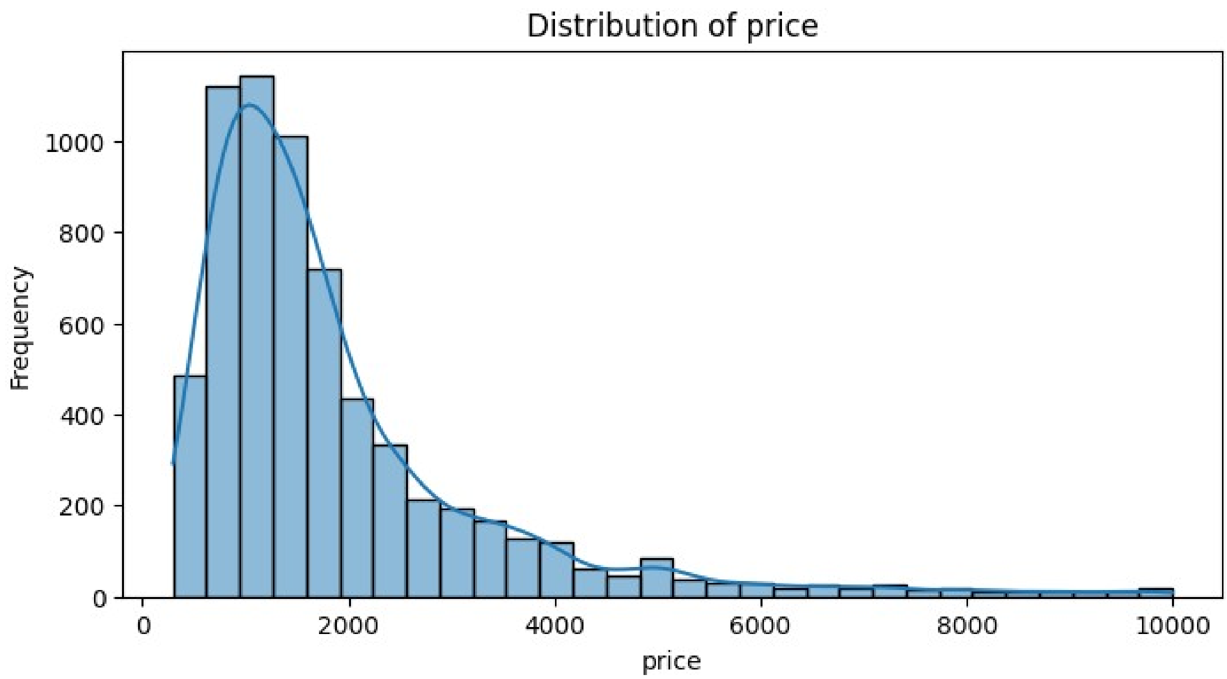

Distribution of Unnamed: 0
Frequency
Unnamed: 0
0
50
100
150
200
250
300
350
2000
4000
6000
8000
10000
12000
14000
16000

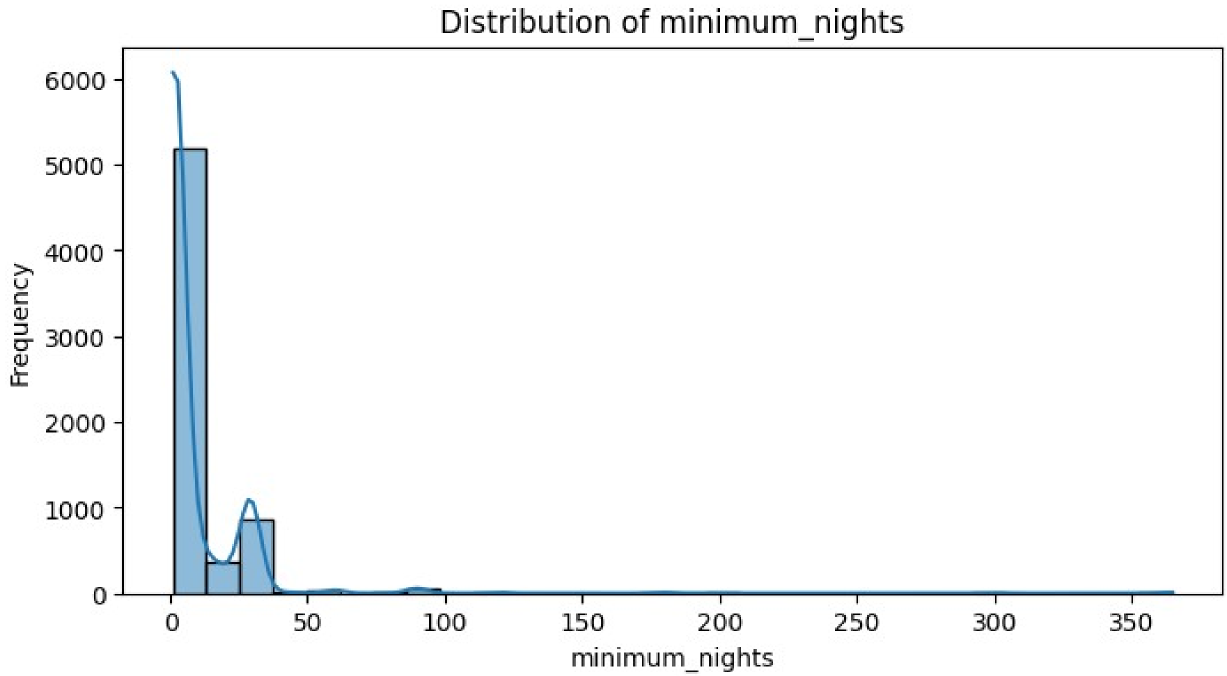

Distribution of id
Frequency
id
1e17
0
1000
2000
3000
4000
1
2
3
4
5
6
7
8

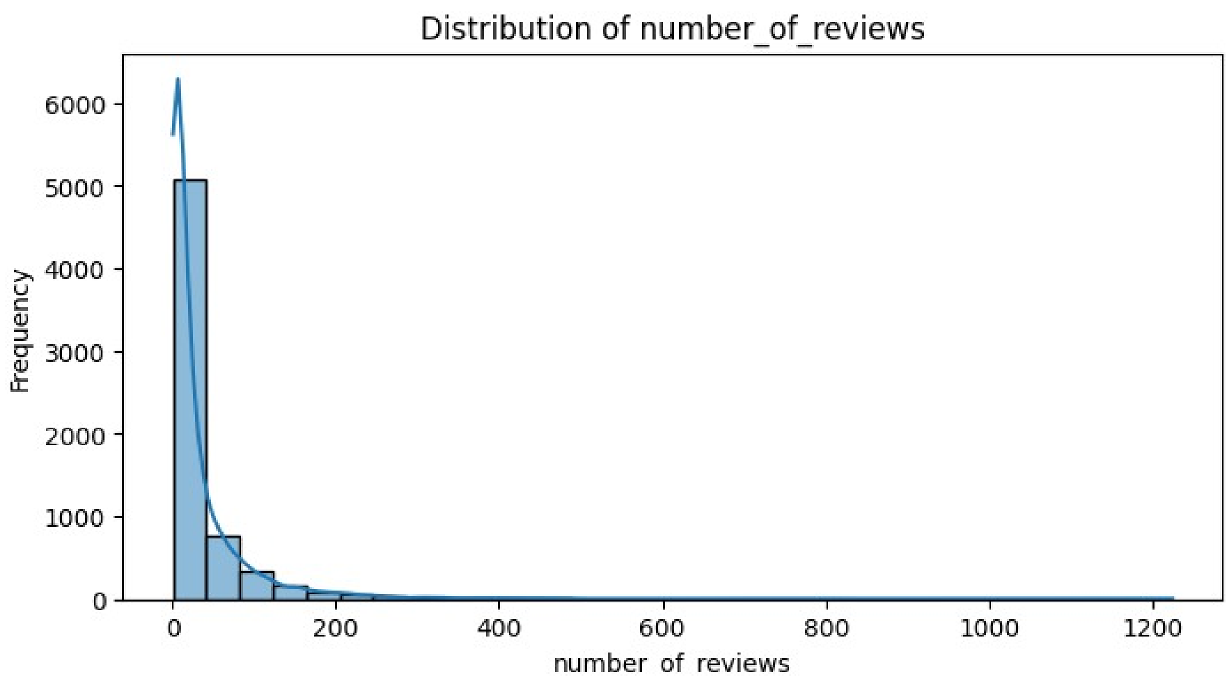

Distribution of host_id
Frequency
host_id
1e8
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5

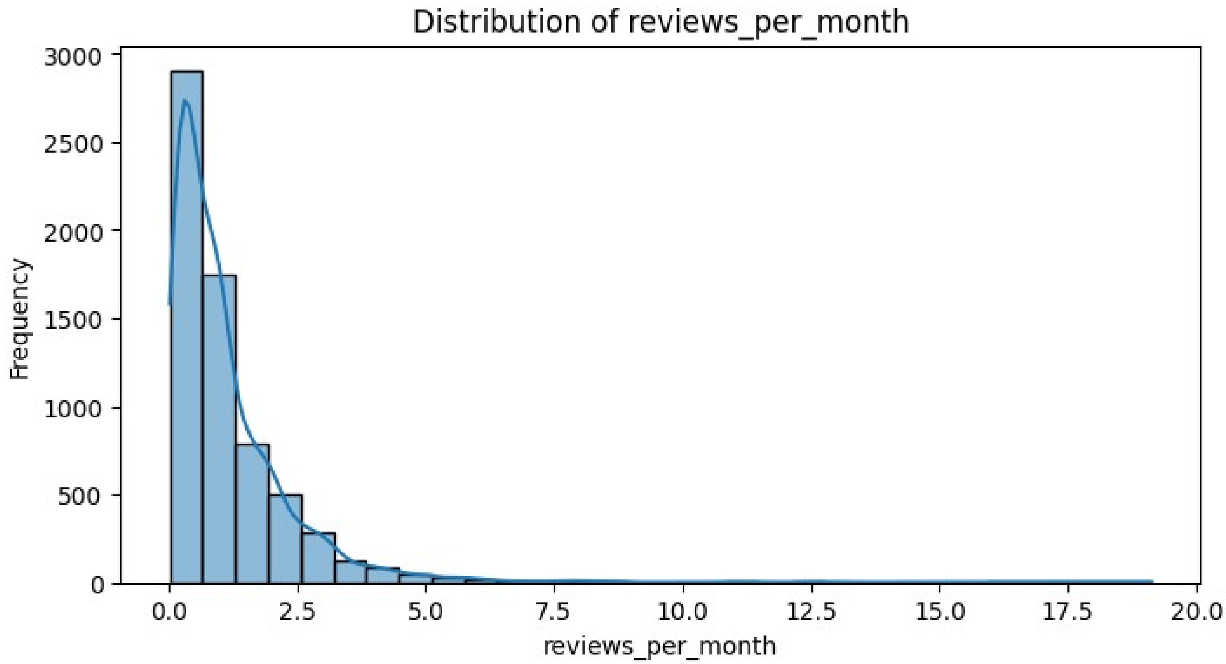

Distribution of latitude
Frequency
latitude
0
200
400
600
800
1000
13.65
13.70
13.75
13.80
13.85
13.90
13.95


Distribution of longitude
Frequency
0
200
400
600
800
1000
1200
1400
100.4
100.5
100.6
100.7
100.8
longitude

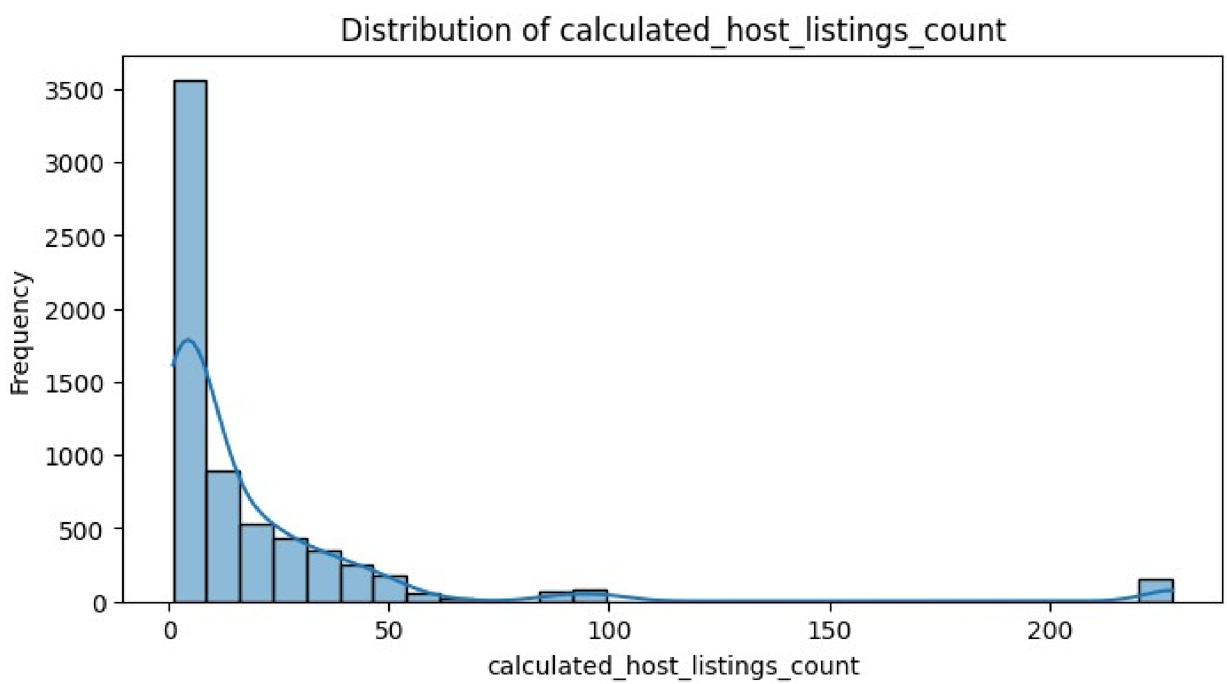

Distribution of price
Frequency
0
200
400
600
800
1000
0
2000
4000
6000
8000
10000
price


Distribution of minimum_nights
Frequency
minimum_nights
0
1000
2000
3000
4000
5000
6000
50
100
150
200
250
300
350

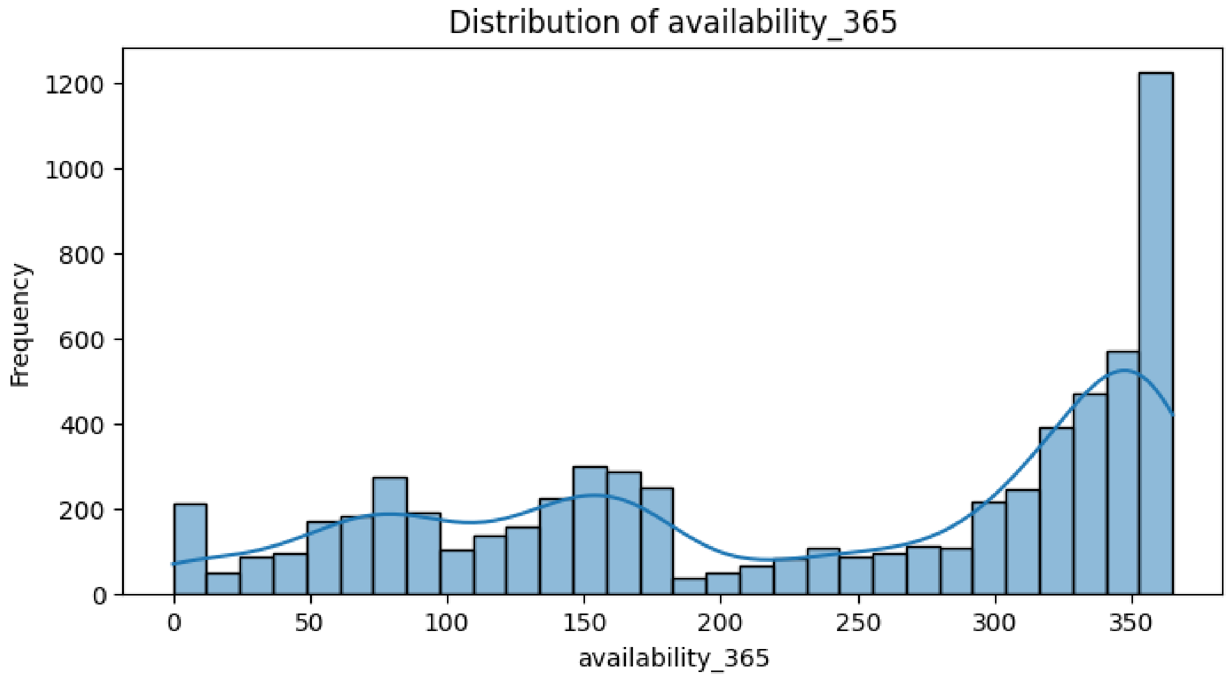

Distribution of number_of_reviews
Frequency
number_of_reviews
0
1000
2000
3000
4000
5000
6000
200
400
600
800
1000
1200

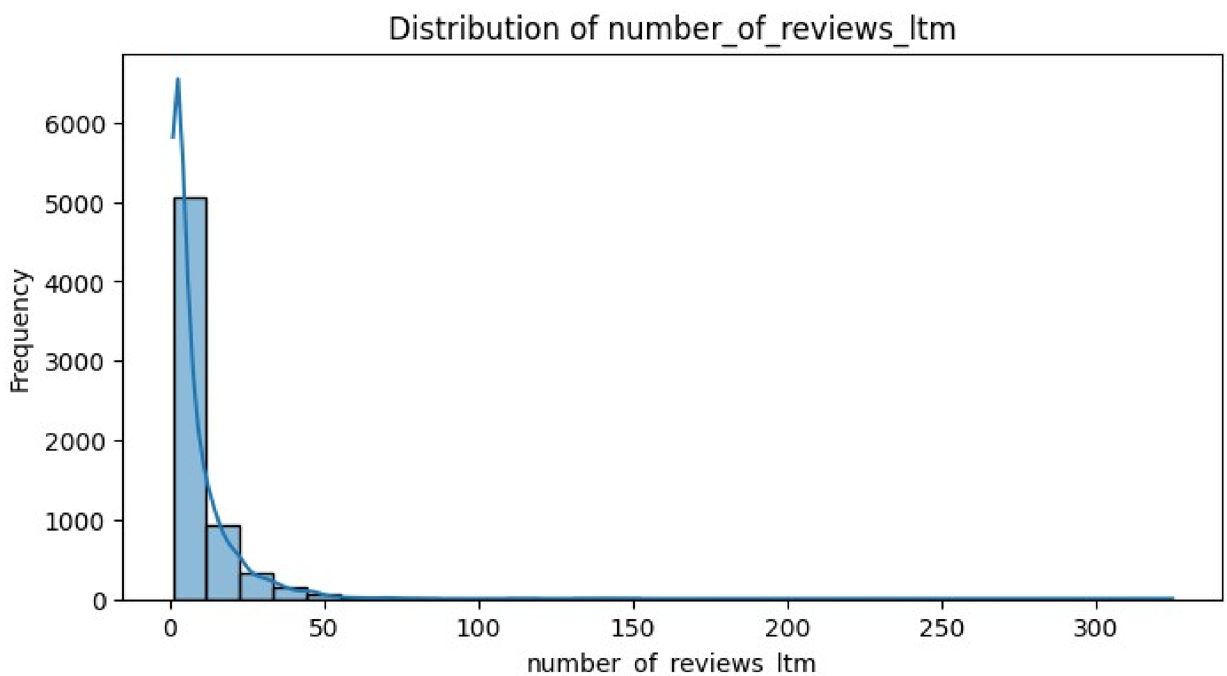


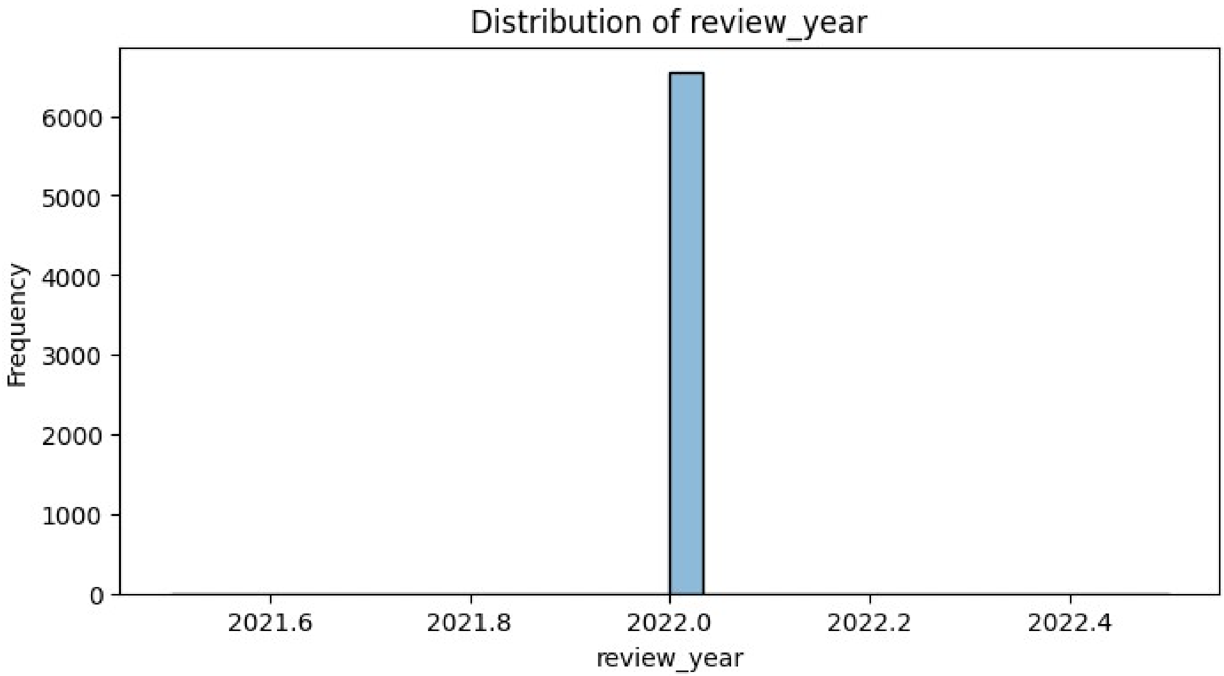
Distribution of reviews_per_month
Frequency
0
500
1000
1500
2000
2500
3000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
reviews_per_month



Distribution of calculated_host_listings_count
Frequency
0
500
1000
1500
2000
2500
3000
3500
0
50
100
150
200
calculated_host_listings_count


Distribution of availability_365
Frequency
availability_365


Distribution of number_of_reviews_ltm
Frequency
number_of_reviews_ltm




```
from scipy.stats import skew, kurtosis

for column in numeric_df.columns:
    skew_value = skew(numeric_df[column])
    kurtosis_value = kurtosis(numeric_df[column], fisher=False)  #
Using fisher=False to get the normal kurtosis value (not excess
kurtosis)

    print(f'{column}: Skewness = {skew_value:.2f}, Kurtosis =
{kurtosis_value:.2f}')

    # Check if the distribution is normal
    if -0.5 <= skew_value <= 0.5 and 2.5 <= kurtosis_value <= 3.5:
        print(f'  {column} appears to be approximately normally
distributed.\n')
    else:
        print(f'  {column} does not appear to be normally
distributed.\n')
```

```
Unnamed: 0: Skewness = -0.31, Kurtosis = 1.85
  Unnamed: 0 does not appear to be normally distributed.

id: Skewness = 0.96, Kurtosis = 1.98
  id does not appear to be normally distributed.

host_id: Skewness = 0.86, Kurtosis = 2.81
  host_id does not appear to be normally distributed.

latitude: Skewness = 1.49, Kurtosis = 7.92
```

```
  latitude does not appear to be normally distributed.

longitude: Skewness = 0.63, Kurtosis = 7.24
  longitude does not appear to be normally distributed.

price: Skewness = 2.34, Kurtosis = 9.74
  price does not appear to be normally distributed.

minimum_nights: Skewness = 9.68, Kurtosis = 138.49
  minimum_nights does not appear to be normally distributed.

number_of_reviews: Skewness = 4.96, Kurtosis = 57.32
  number_of_reviews does not appear to be normally distributed.

reviews_per_month: Skewness = 3.92, Kurtosis = 36.87
  reviews_per_month does not appear to be normally distributed.

calculated_host_listings_count: Skewness = 4.33, Kurtosis = 23.62
  calculated_host_listings_count does not appear to be normally
distributed.

availability_365: Skewness = -0.42, Kurtosis = 1.69
  availability_365 does not appear to be normally distributed.

number_of_reviews_ltm: Skewness = 7.82, Kurtosis = 141.86
  number_of_reviews_ltm does not appear to be normally distributed.

review_year: Skewness = nan, Kurtosis = nan
  review_year does not appear to be normally distributed.


C:\Users\putri\AppData\Local\Temp\ipykernel_21792\2990220349.py:4:
RuntimeWarning: Precision loss occurred in moment calculation due to
catastrophic cancellation. This occurs when the data are nearly
identical. Results may be unreliable.
  skew_value = skew(numeric_df[column])
C:\Users\putri\AppData\Local\Temp\ipykernel_21792\2990220349.py:5:
RuntimeWarning: Precision loss occurred in moment calculation due to
catastrophic cancellation. This occurs when the data are nearly
identical. Results may be unreliable.
  kurtosis_value = kurtosis(numeric_df[column], fisher=False)  # Using
fisher=False to get the normal kurtosis value (not excess kurtosis)
```

```python
from scipy.stats import shapiro

for column in numeric_df.columns:
    stat, p = shapiro(numeric_df[column])
    print(f'{column}: Statistics={stat:.3f}, p={p:.3f}')

    # Check normality based on p-value
    if p > 0.05:
```

```
        print(f'  {column} appears to be normally distributed (p > 0.05).\n')
    else:
        print(f'  {column} does not appear to be normally distributed (p <= 0.05).\n')
```

```
Unnamed: 0: Statistics=0.940, p=0.000
  Unnamed: 0 does not appear to be normally distributed (p <= 0.05).

id: Statistics=0.602, p=0.000
  id does not appear to be normally distributed (p <= 0.05).

host_id: Statistics=0.892, p=0.000
  host_id does not appear to be normally distributed (p <= 0.05).

latitude: Statistics=0.900, p=0.000
  latitude does not appear to be normally distributed (p <= 0.05).

longitude: Statistics=0.942, p=0.000
  longitude does not appear to be normally distributed (p <= 0.05).

price: Statistics=0.761, p=0.000
  price does not appear to be normally distributed (p <= 0.05).

minimum_nights: Statistics=0.336, p=0.000
  minimum_nights does not appear to be normally distributed (p <= 0.05).

number_of_reviews: Statistics=0.569, p=0.000
  number_of_reviews does not appear to be normally distributed (p <= 0.05).

reviews_per_month: Statistics=0.713, p=0.000
  reviews_per_month does not appear to be normally distributed (p <= 0.05).

calculated_host_listings_count: Statistics=0.476, p=0.000
  calculated_host_listings_count does not appear to be normally distributed (p <= 0.05).

availability_365: Statistics=0.876, p=0.000
  availability_365 does not appear to be normally distributed (p <= 0.05).

number_of_reviews_ltm: Statistics=0.549, p=0.000
  number_of_reviews_ltm does not appear to be normally distributed (p <= 0.05).

review_year: Statistics=1.000, p=1.000
```

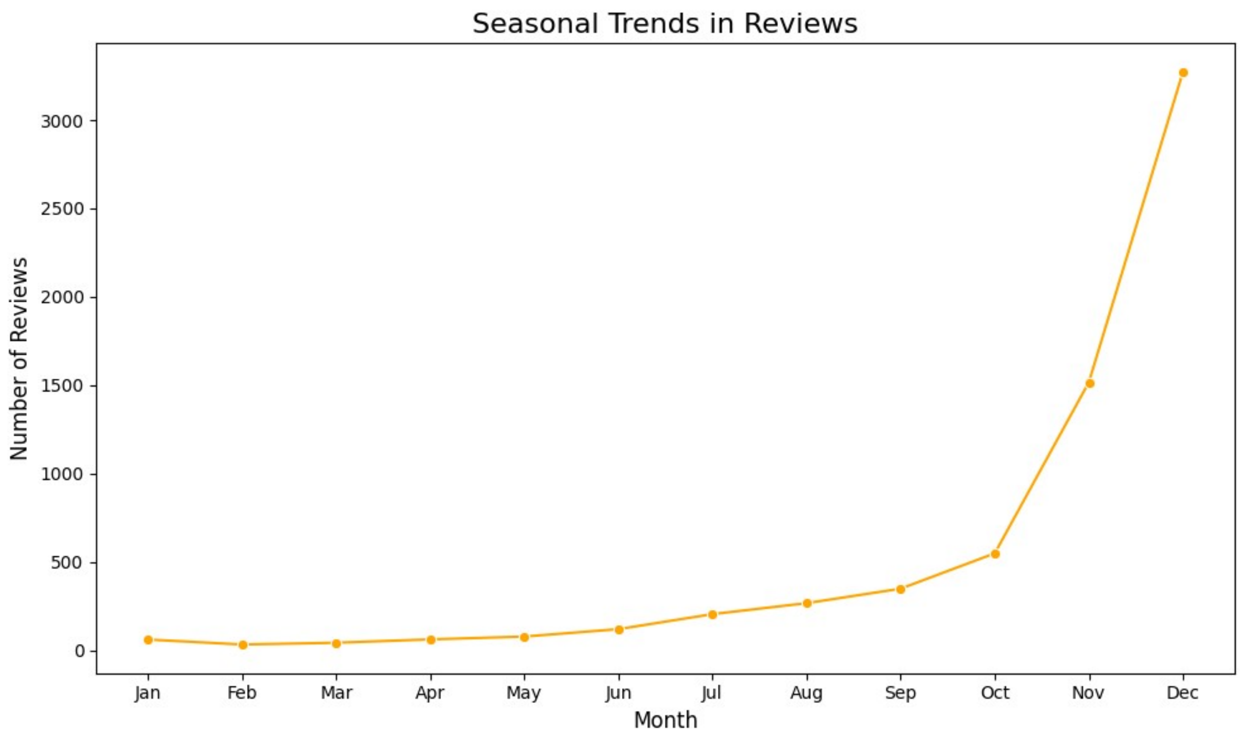
```
  review_year appears to be normally distributed (p > 0.05).


c:\Users\putri\anaconda3\envs\jcds0412\Lib\site-packages\scipy\stats\
_axis_nan_policy.py:531: UserWarning: scipy.stats.shapiro: For N >
5000, computed p-value may not be accurate. Current N is 6545.
  res = hypotest_fun_out(*samples, **kwds)
c:\Users\putri\anaconda3\envs\jcds0412\Lib\site-packages\scipy\stats\
_axis_nan_policy.py:531: UserWarning: scipy.stats.shapiro: Input data
has range zero. The results may not be accurate.
  res = hypotest_fun_out(*samples, **kwds)
```

```
dfc['last_review'] = pd.to_datetime(df['last_review'],
errors='coerce')
dfc['review_month'] = dfc['last_review'].dt.month
monthly_reviews = dfc['review_month'].value_counts().sort_index()
monthly_reviews.index = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
monthly_reviews
```

```
Jan      60
Feb      32
Mar      42
Apr      61
May      77
Jun     119
Jul     204
Aug     266
Sep     348
Oct     548
Nov    1516
Dec    3272
Name: count, dtype: int64
```

count last_review per month because we assume that if the person is leaving a review, they must booked and stayed in the airbnb regradless the review is positive or negative. so we can use this data to calculate the number of bookings per month in 2022.

```
plt.figure(figsize=(10, 6))
sns.lineplot(
    x=monthly_reviews.index,
    y=monthly_reviews.values,
    marker='o', color='orange'
)
plt.title('Seasonal Trends in Reviews', fontsize=16)
plt.xlabel('Month', fontsize=12)
plt.ylabel('Number of Reviews', fontsize=12)
plt.tight_layout()
plt.show()
```



november and december has high number of review, might be cause the holiday season

```
dfc['availability_category'] = pd.cut(
    dfc['availability_365'],
    bins=[0, 90, 180, 270, 365],
    labels=['Low (0-90 days)', 'Moderat (91-180 days)', 'High (181-270 days)', 'Very High (271-365 days)']
)

availability_counts = dfc['availability_category'].value_counts().sort_index()
availability_counts
```

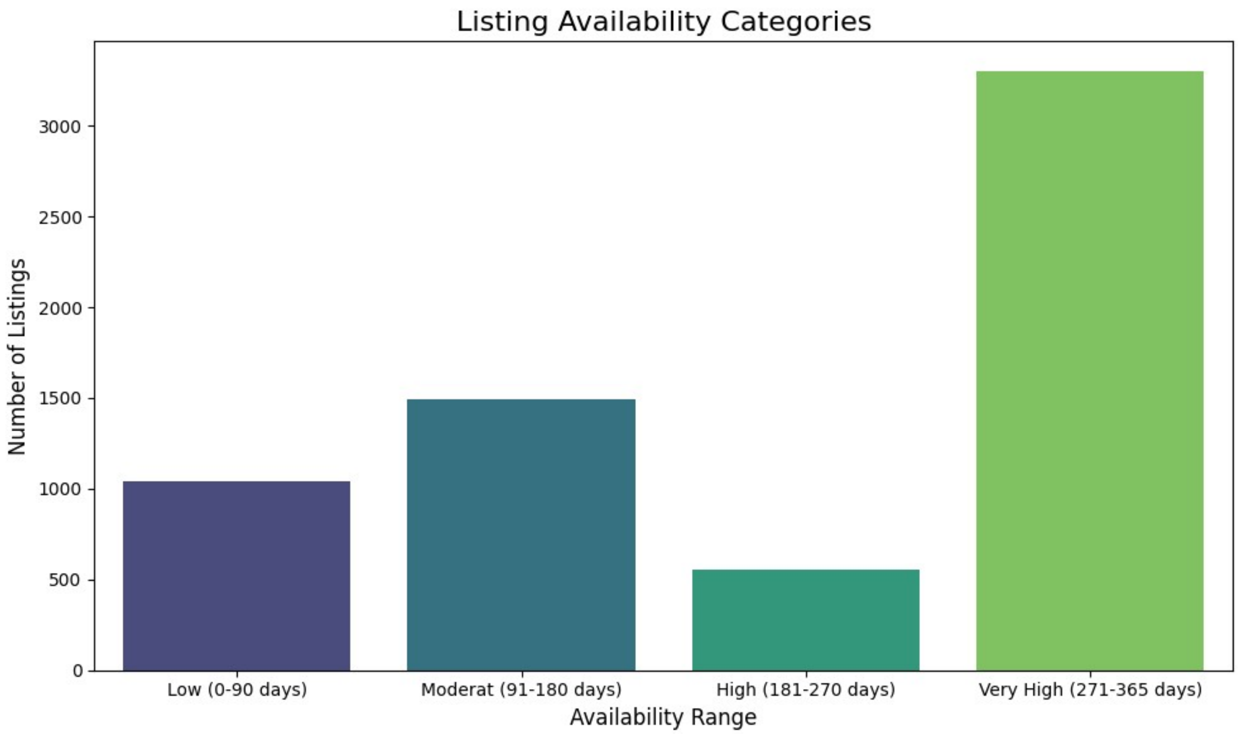
```
availability_category
Low (0-90 days)             1040
Moderat (91-180 days)       1495
High (181-270 days)          557
Very High (271-365 days)    3300
Name: count, dtype: int64
```

grouping the available listing days into 4 category, so we can see the demand of airbnb in bangkok in 2023

```
plt.figure(figsize=(10, 6))
sns.barplot(
    x=availability_counts.index,
    y=availability_counts.values,
```

```
    palette='viridis'
)
plt.title('Listing Availability Categories', fontsize=16)
plt.xlabel('Availability Range', fontsize=12)
plt.ylabel('Number of Listings', fontsize=12)
plt.tight_layout()
plt.show()
```

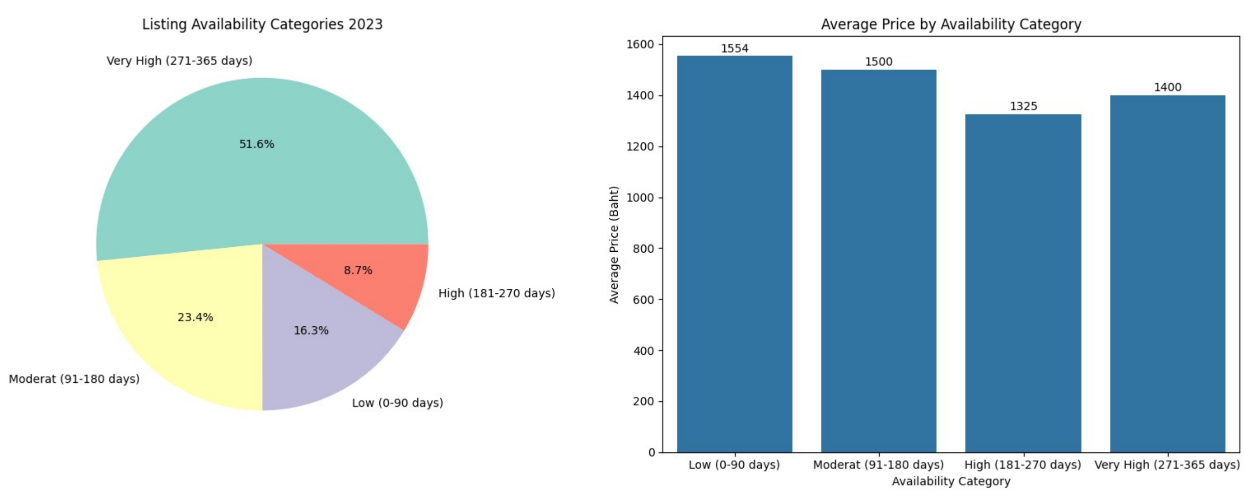
```
C:\Users\putri\AppData\Local\Temp\ipykernel_21792\566332460.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(
```




```
# Grouping by availability category and calculating average price
availability_price = dfc.groupby('availability_category')
['price'].median()

# Pie chart for Availability Categories
availability_counts = dfc['availability_category'].value_counts()
fig, ax = plt.subplots(1, 2, figsize=(16, 6))

# Pie chart
ax[0].pie(availability_counts, labels=availability_counts.index,
```

```
autopct='%1.1f%%', colors=sns.color_palette('Set3'))
ax[0].set_title("Listing Availability Categories 2023")

# Bar chart for Average Price by Availability Category
sns.barplot(x=availability_price.index, y=availability_price.values,
ax=ax[1])
ax[1].set_title("Average Price by Availability Category")
ax[1].set_xlabel("Availability Category")
ax[1].set_ylabel("Average Price (Baht)")

# Adding labels above each bar in the bar chart
for i, v in enumerate(availability_price.values):
    ax[1].text(i, v + 5, f'{v:.0f}', ha='center', va='bottom',
fontsize=10, color='black')

plt.tight_layout()
plt.show()
```

```
C:\Users\putri\AppData\Local\Temp\ipykernel_21792\2393350616.py:2:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  availability_price = dfc.groupby('availability_category')
['price'].median()
```




```
numeric_df = dfc.select_dtypes(include=['number'])
num_cat_avg = dfc.groupby('availability_category')
[numeric_df.columns].apply(lambda x: x.median())
num_cat_avg
```

```
C:\Users\putri\AppData\Local\Temp\ipykernel_21792\3484054333.py:2:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
```

```
retain current behavior or observed=True to adopt the future default
and silence this warning.
  num_cat_avg = dfc.groupby('availability_category')
[numeric_df.columns].apply(lambda x: x.median())

                          Unnamed: 0          id      host_id
latitude  \
availability_category

Low (0-90 days)               9654.5  40333233.0  117247424.0
13.738390
Moderat (91-180 days)         8613.0  37401664.0  101777383.0
13.737440
High (181-270 days)           8539.0  37281532.0  112970036.0
13.734270
Very High (271-365 days)      9731.0  40685406.5  115758511.0
13.735775

                           longitude   price  minimum_nights  \
availability_category
Low (0-90 days)           100.561470  1553.5             2.0
Moderat (91-180 days)     100.559950  1500.0             2.0
High (181-270 days)       100.558750  1325.0             3.0
Very High (271-365 days)  100.557906  1399.5             1.0

                          number_of_reviews  reviews_per_month  \
availability_category
Low (0-90 days)                        15.0               0.89
Moderat (91-180 days)                  14.0               0.80
High (181-270 days)                    13.0               0.71
Very High (271-365 days)                9.0               0.77

                          calculated_host_listings_count
availability_365  \
availability_category

Low (0-90 days)                                      8.5
65.0
Moderat (91-180 days)                                8.0
148.0
High (181-270 days)                                  5.0
236.0
Very High (271-365 days)                             7.0
344.0

                          number_of_reviews_ltm  review_year
review_month
availability_category

Low (0-90 days)                             6.0       2022.0
```

```
12.0
Moderat (91-180 days)                          5.0       2022.0
12.0
High (181-270 days)                            5.0       2022.0
11.0
Very High (271-365 days)                       4.0       2022.0
12.0

object_df = dfc.select_dtypes(include=['object'])
obj_cat_avg = dfc.groupby('availability_category')
[object_df.columns].apply(lambda x: x.mode().iloc[0])
obj_cat_avg

C:\Users\putri\AppData\Local\Temp\ipykernel_21792\2879233730.py:2:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  obj_cat_avg = dfc.groupby('availability_category')
[object_df.columns].apply(lambda x: x.mode().iloc[0])

0
name  \
availability_category

Low (0-90 days)            2 Mins walk BTS. 4pp walk Siam,
MBK,CTW,WaterGate
Moderat (91-180 days)       Nana BTS Spacious 1BR W/Balcony Asok
Terminal 21
High (181-270 days)       1Bedroom#CloudPool#BTS Phrompong#Nice
Gym#Shopping
Very High (271-365 days)  30days! AirportLink Sukhumvit NANA MaxValu
2BR(4P)

0                          host_name neighbourhood        room_type
availability_category
Low (0-90 days)                 Mike       Vadhana  Entire home/apt
Moderat (91-180 days)       Ludoping       Vadhana  Entire home/apt
High (181-270 days)       Hi Gravity   Khlong Toei  Entire home/apt
Very High (271-365 days)       Curry   Khlong Toei  Entire home/apt

dfc['last_review'] = pd.to_datetime(dfc['last_review'],
errors='coerce')
dfc['review_month'] = dfc['last_review'].dt.month
month_names = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec']

numeric_df = dfc.select_dtypes(include=['number'])
num_month_avg = dfc.groupby('review_month')
[numeric_df.columns].apply(lambda x: x.median())
num_month_avg
```

```
              Unnamed: 0          id      host_id   latitude
longitude  \
review_month

1                 7219.5  32934100.5   75733956.5  13.724660
100.555815
2                 7613.5  34054983.0  147719704.5  13.731820
100.566020
3                 7558.5  33748529.0  115619116.0  13.739370
100.565400
4                 8283.0  36504455.0  166559468.0  13.740600
100.556950
5                 7203.0  32736541.0  102454560.0  13.736950
100.555750
6                 7666.0  34142702.0  132473239.0  13.732700
100.565690
7                 7994.5  35197125.5   98998539.0  13.739465
100.560070
8                 7885.5  34937843.5  131270916.0  13.740315
100.557850
9                 9649.0  40477953.5  120226663.5  13.736840
100.558574
10                8079.0  35489382.0  105141568.5  13.734888
100.563055
11                9655.0  40523269.5  115758511.0  13.735860
100.558885
12                9765.5  40784616.0  108026474.0  13.736530
100.558990

               price  minimum_nights  number_of_reviews
reviews_per_month  \
review_month

1             1170.0             2.0                4.5
0.155
2             1475.0             2.5                3.5
0.145
3             1199.5             2.0                4.5
0.165
4             1364.0             7.0                4.0
0.140
5             1286.0             3.0                9.0
0.230
6             1290.0             6.0                6.0
0.210
7             1380.5             2.0                5.0
0.200
8             1346.0             2.0                6.0
0.255
9             1300.0             2.0                7.0
```

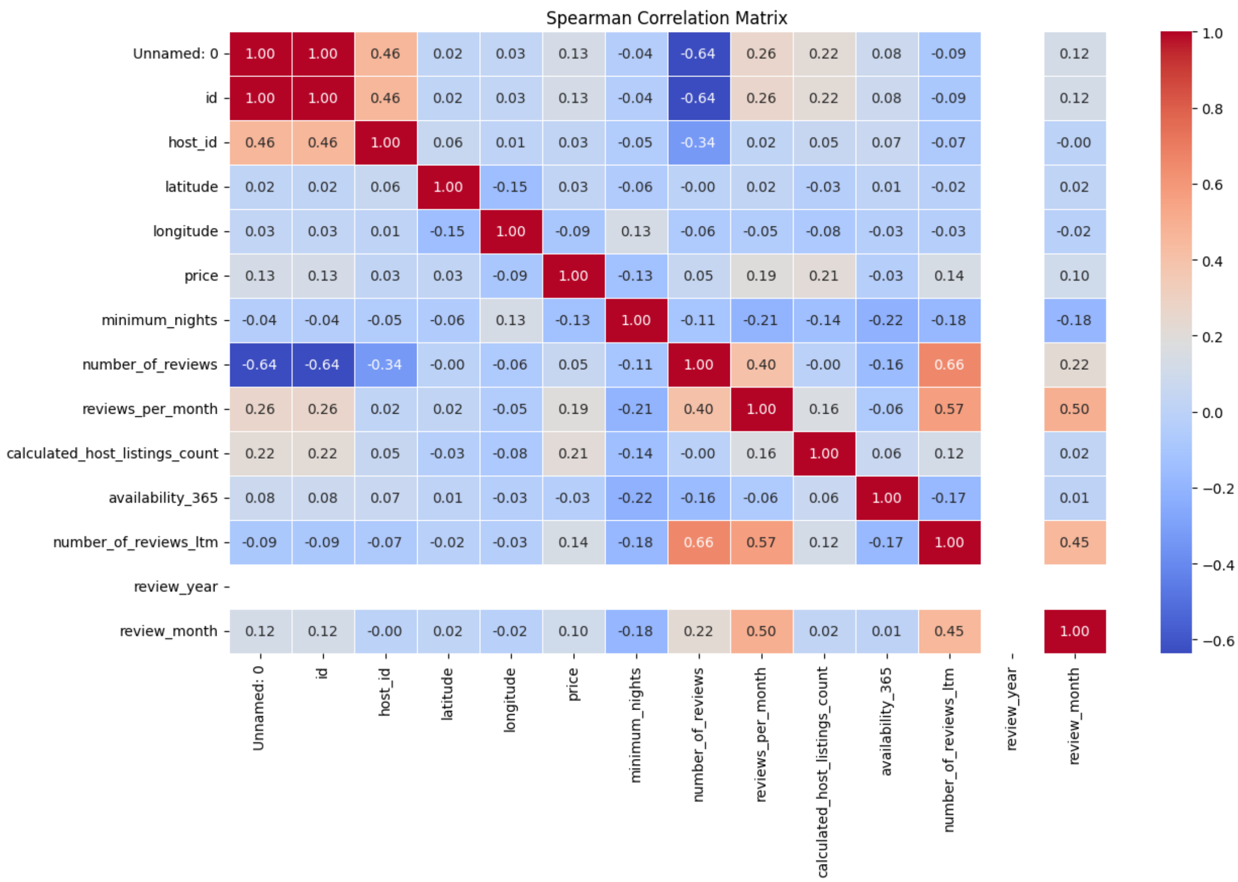
```
0.350
10           1295.0            3.0                  9.0
0.450
11           1375.0            2.0                  9.0
0.715
12           1536.0            1.0                 17.0
1.120

             calculated_host_listings_count  availability_365  \
review_month
1                                       4.0             178.0
2                                       9.5             194.5
3                                       5.0             176.0
4                                       5.0             276.0
5                                       5.0             252.0
6                                       5.0             248.0
7                                       8.0             285.0
8                                       8.0             266.5
9                                       9.0             252.0
10                                      6.0             253.0
11                                      7.0             283.5
12                                      7.0             281.0

             number_of_reviews_ltm  review_year  review_month
review_month
1                              1.0       2022.0           1.0
2                              1.0       2022.0           2.0
3                              1.0       2022.0           3.0
4                              1.0       2022.0           4.0
5                              1.0       2022.0           5.0
6                              1.0       2022.0           6.0
7                              1.0       2022.0           7.0
8                              2.0       2022.0           8.0
9                              2.0       2022.0           9.0
10                             3.0       2022.0          10.0
11                             4.0       2022.0          11.0
12                             7.5       2022.0          12.0
```

```python
object_df = dfc.select_dtypes(include=['object'])
obj_month_avg = dfc.groupby('review_month')
[object_df.columns].apply(lambda x: x.mode().iloc[0])
obj_month_avg
```

```
0                                                          name  \
review_month
1                   (302) Cozy room, Close to BTS , Good location
2                                  1 Bedroom 20 sqm Sukhumvit 33
3                             1 br Suite at LUXX XL Langsuan (8 of 8)
4                  1 mins to MRT Bang O station，The tree rio home
5                            "Clean and Silent space around CHATUJAK"
```

```
6              Spacious Studio Room between Phromphong & Asok BTS
7                           #3 5 Star facilities River View Condo
8              30days! AirportLink Sukhumvit NANA MaxValu 2BR(4P)
9              New spacious 2BR 3PPL with pool&gym Silom &Sathorn
10              1BR Twin Suit 2ppl/Surasak BTS Sathorn/Pool /Wifi
11              1BR Twin Suit 2ppl/Surasak BTS Sathorn/Pool /WIFI
12             30days! AirportLink Sukhumvit NANA MaxValu 2BR(4P)

0                        host_name neighbourhood        room_type
review_month
1                          Dusadee   Khlong Toei  Entire home/apt
2                    ISanook Hotel   Khlong Toei  Entire home/apt
3                  Danai And BicGy   Khlong Toei  Entire home/apt
4                    ISanook Hotel       Vadhana  Entire home/apt
5                      Dr. Piyamas   Khlong Toei  Entire home/apt
6                           Cherry   Khlong Toei  Entire home/apt
7                           Joseph   Khlong Toei  Entire home/apt
8                            Curry   Khlong Toei  Entire home/apt
9                            Curry       Vadhana  Entire home/apt
10                               K   Khlong Toei  Entire home/apt
11                           Curry   Khlong Toei  Entire home/apt
12                           Curry   Khlong Toei  Entire home/apt
```

```
numeric_df = dfc.select_dtypes(include=['number'])
correlation_matrix = numeric_df.corr(method='spearman')
plt.figure(figsize=(14, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f', linewidths=0.5)
plt.title('Spearman Correlation Matrix')
plt.show()
```



4.1 Price vs Availability Categories

H0: The average price does not vary significantly across availability categories. Ha: The average price varies significantly across availability categories.

```
from scipy.stats import kruskal

# Filter out empty groups
groups = [
    dfc[dfc['availability_category'] == 'Low (0-90 days)']['price'],
    dfc[dfc['availability_category'] == 'High (181-270 days)']['price'],
    dfc[dfc['availability_category'] == 'Very High (271-365 days)']['price']
]

# Perform the Kruskal-Wallis test
stat, pvalue = kruskal(*groups)

print(f"Kruskal-Wallis Statistic: {stat}, p-Value: {pvalue}")

if pvalue > 0.05:
    print("Fail to reject H0: There is no significant difference in
```

```
average price across availability categories.")
else:
    print("Reject H0: There is a significant difference in average
price across at least one availability category.")
```

```
Kruskal-Wallis Statistic: 33.34287228321371, p-Value:
5.750257304090272e-08
Reject H0: There is a significant difference in average price across
at least one availability category.
```

```
summary_stats = dfc.groupby('availability_category')['price'].median()
summary_stats
```

```
C:\Users\putri\AppData\Local\Temp\ipykernel_21792\1886712068.py:1:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  summary_stats = dfc.groupby('availability_category')
['price'].median()
```

```
availability_category
Low (0-90 days)              1553.5
Moderat (91-180 days)        1500.0
High (181-270 days)          1325.0
Very High (271-365 days)     1399.5
Name: price, dtype: float64
```

1. **Low Availability (0-90 days):** Median Price: 1553.5 Listings in this category are available for only a small portion of the year, indicating they are either very popular or located in high-demand areas. The higher price could be due to limited supply: fewer available days typically signal that these listings are in demand and can charge a premium. This could be a sign of premium properties, such as those in desirable neighborhoods or those offering unique experiences (e.g., entire homes, luxurious amenities).
2. **Moderate Availability (91-180 days):** Median Price: 1500.0 Listings in this category have a moderate number of available days (a few months of the year), suggesting that these listings may be in areas that are somewhat in demand but have a balanced supply. The price is slightly lower than the low availability category, which reflects moderate demand. The owner may be trying to fill these available days with competitive pricing. Properties with moderate availability might have a broad range of options, including those that are less central or less premium but still in desirable neighborhoods.
3. **High Availability (181-365 days):** Median Price: 1450.0 Listings in this category are available for more than half the year, indicating that these listings are likely more abundant. The lower median price compared to the previous two categories suggests higher supply relative to demand. With more availability, property owners might lower prices to fill their calendar. These listings may also be in less popular or less tourist-heavy neighborhoods where demand is not as intense.

4. **Very High Availability (366+ days):** Median Price: 1400.0 Listings in this category are available for nearly the entire year, which could suggest that the property owner is more focused on long-term rentals or has a higher volume of bookings. The slightly higher price compared to high availability listings indicates that owners might be trying to maintain profitability while accommodating more bookings throughout the year. This could also suggest properties with strong customer loyalty or those that are frequently booked during off-peak seasons, where owners can afford to charge a premium during certain months.

Insight and Interpretation of Price Dynamics Inverse Relationship Between Price and Availability: There is a clear inverse relationship between the number of available days (availability) and the price. Listings with fewer available days (Low Availability) tend to be priced higher. This could be because these listings are in high-demand locations or offer premium experiences, and the scarcity of available dates allows owners to charge a higher price. Listings with more available days (High and Very High Availability) are typically priced lower, as there is more competition to fill those dates. More availability can lead to lower pricing to attract guests, especially in less competitive neighborhoods or during off-peak seasons.

4.2 Room Type vs Review Month

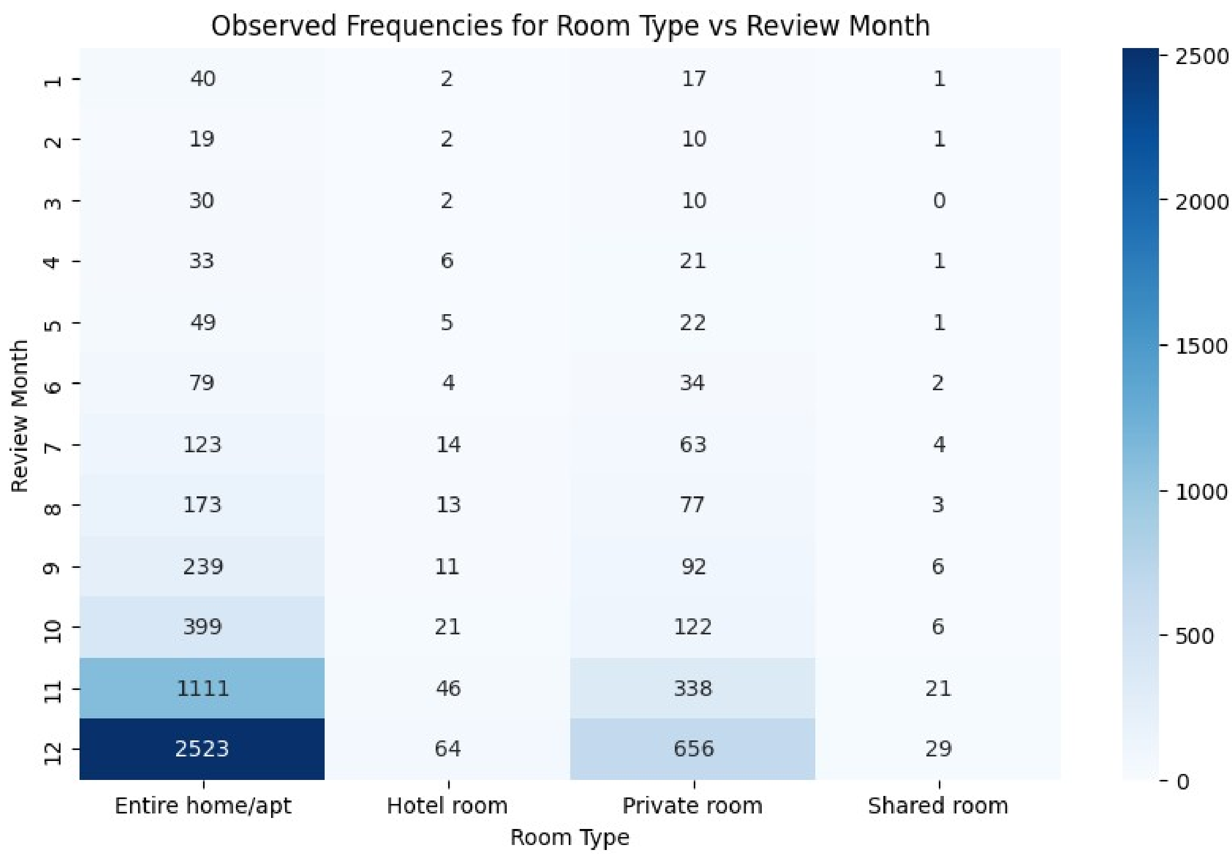
H0: Room type preference is independent of the review month. Ha: Room type preference depends on the review month.

```
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(dfc['review_month'], dfc['room_type'])
chi2_stat, pvalue, dof, expected = chi2_contingency(contingency_table)

print(f"Chi-Square Statistic: {chi2_stat}, p-Value: {pvalue}, Degrees
of Freedom: {dof}")
print("Expected Frequencies Table:")
print(expected)

if pvalue > 0.05:
    print("Fail to reject H0: Room type preference is independent of
the review month.")
else:
    print("Reject H0: Room type preference depends on the review
month.")
```

```
Chi-Square Statistic: 101.24229997369902, p-Value: 7.433969773615342e-
09, Degrees of Freedom: 33
Expected Frequencies Table:
[[4.41680672e+01 1.74178762e+00 1.34025974e+01 6.87547746e-01]
 [2.35563025e+01 9.28953400e-01 7.14805195e+00 3.66692131e-01]
 [3.09176471e+01 1.21925134e+00 9.38181818e+00 4.81283422e-01]
 [4.49042017e+01 1.77081742e+00 1.36259740e+01 6.99006875e-01]
 [5.66823529e+01 2.23529412e+00 1.72000000e+01 8.82352941e-01]
 [8.76000000e+01 3.45454545e+00 2.65818182e+01 1.36363636e+00]
 [1.50171429e+02 5.92207792e+00 4.55688312e+01 2.33766234e+00]
```

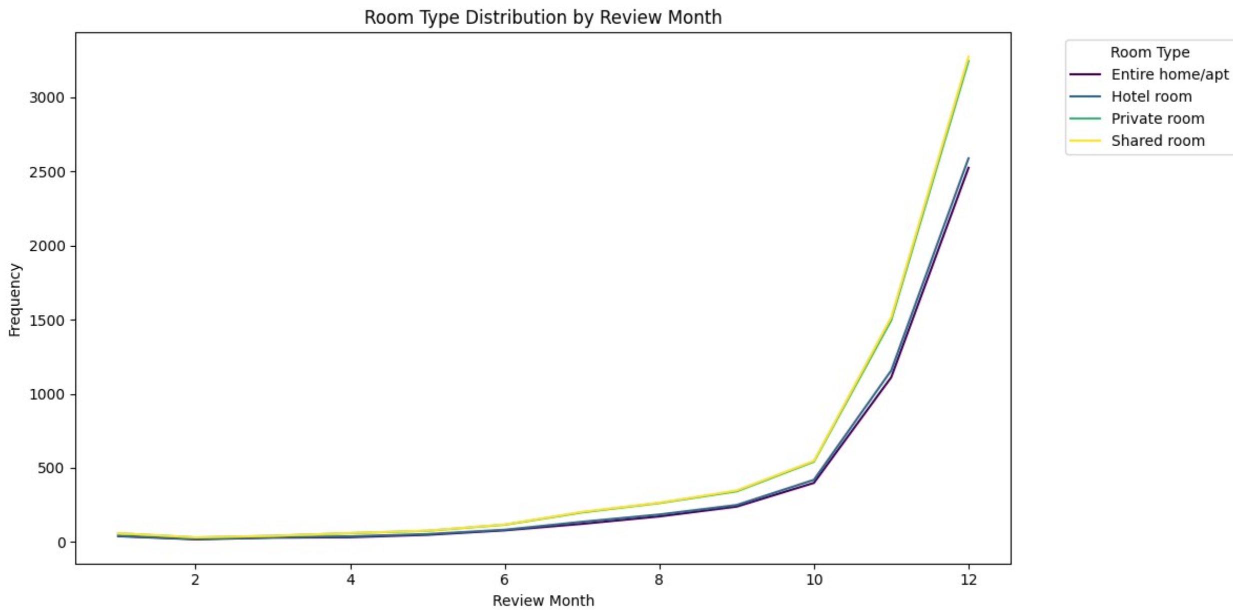
```
 [1.95811765e+02 7.72192513e+00 5.94181818e+01 3.04812834e+00]
 [2.56174790e+02 1.01023682e+01 7.77350649e+01 3.98777693e+00]
 [4.03401681e+02 1.59083270e+01 1.22410390e+02 6.27960275e+00]
 [1.11597983e+03 4.40091673e+01 3.38638961e+02 1.73720397e+01]
 [2.40863193e+03 9.49854851e+01 7.30888312e+02 3.74942704e+01]]
Reject H0: Room type preference depends on the review month.
```

```
observed = contingency_table
plt.figure(figsize=(10, 6))
sns.heatmap(observed, annot=True, fmt="d", cmap="Blues")
plt.title("Observed Frequencies for Room Type vs Review Month")
plt.xlabel("Room Type")
plt.ylabel("Review Month")
plt.show()
```




```
observed.plot(kind="line", stacked=True, figsize=(12, 6),
colormap="viridis")
plt.title("Room Type Distribution by Review Month")
plt.xlabel("Review Month")
plt.ylabel("Frequency")
plt.legend(title="Room Type", bbox_to_anchor=(1.05, 1), loc='upper
left')
```

```
plt.tight_layout()
plt.show()
```



1. Seasonal Trends: Certain months may show higher demand for particular room types, such as people preferring entire homes during holidays or festivals, whereas they might choose private or shared rooms in off-peak months. Event-driven Preferences: Major events or holidays in certain months (e.g., New Year, Songkran festival in Thailand) may drive more group bookings for larger properties (entire homes), leading to a higher preference for such room types in those specific months.

2. Weather Impact: If certain months are considered rainy seasons or low tourist seasons, guests might prefer smaller, more budget-friendly options (e.g., private or shared rooms), while in peak tourist seasons, they may opt for larger accommodations.

Conclusion Room type preference does depend on the review month if the p-value is below 0.05, meaning that the type of accommodation guests prefer changes throughout the year. By understanding these seasonal trends, Airbnb hosts and property managers can optimize their pricing and availability strategies to cater to changing customer preferences across different months. This kind of analysis helps to better understand how customer behavior varies by season and can lead to more data-driven decisions on room type offerings and pricing strategies.

4.3 Price vs Room Type

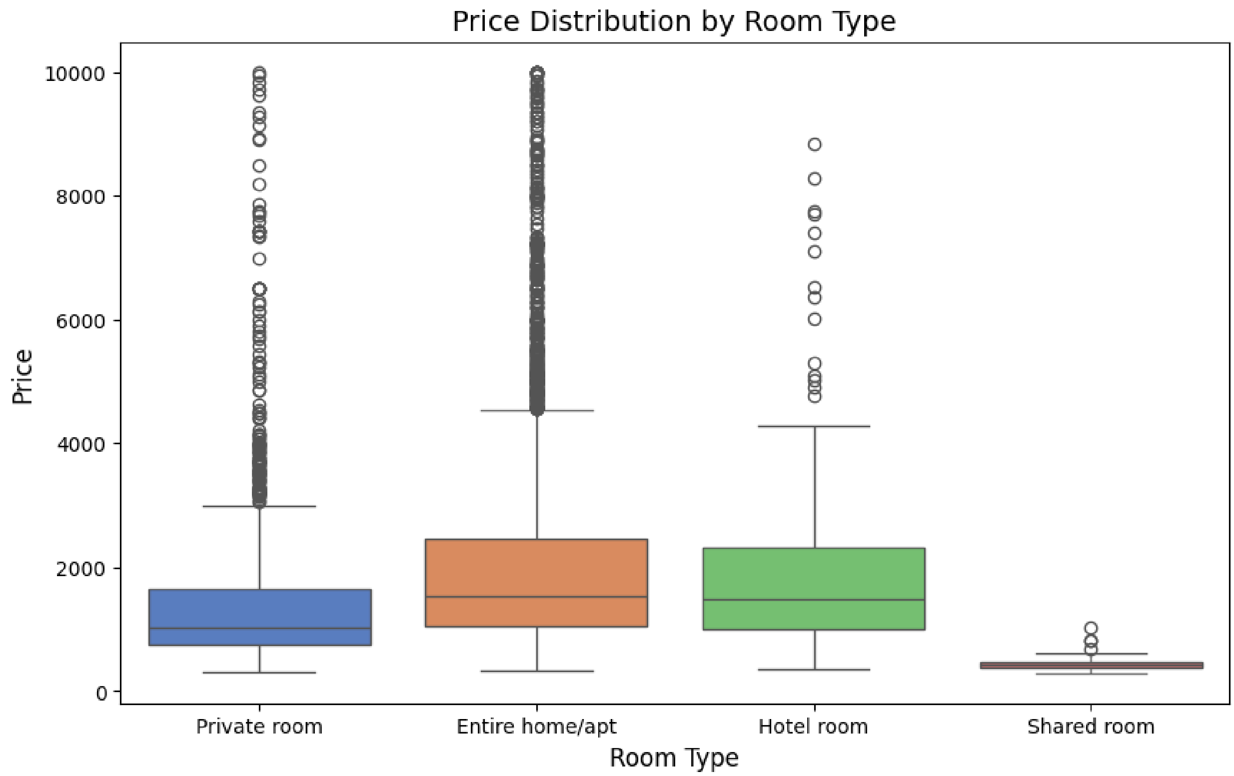
```
groups = [dfc[dfc['room_type'] == room]['price'] for room in
dfc['room_type'].unique()]

kruskal_stat, kruskal_pvalue = kruskal(*groups)
print(f"Kruskal-Wallis Statistic: {kruskal_stat}, p-Value:
```

```python
{kruskal_pvalue}")

if kruskal_pvalue > 0.05:
    print("Fail to reject H₀: No significant difference in median
price across room types.")
else:
    print("Reject H₀: Significant difference in median price across
room types.")
```

```
Kruskal-Wallis Statistic: 562.8750074179914, p-Value:
1.1250211883597465e-121
Reject H₀: Significant difference in median price across room types.
```

```python
plt.figure(figsize=(10, 6))
sns.boxplot(x='room_type', y='price', data=dfc, palette='muted')
plt.title('Price Distribution by Room Type', fontsize=14)
plt.xlabel('Room Type', fontsize=12)
plt.ylabel('Price', fontsize=12)
plt.show()
```

```
C:\Users\putri\AppData\Local\Temp\ipykernel_21792\2326351113.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='room_type', y='price', data=dfc, palette='muted')
```




```
room_type_stats = df.groupby('room_type')['price'].agg(['median',
'mean', 'count']).sort_values(by='median')
room_type_stats

                 median         mean  count
room_type
Shared room       500.0   919.757170    523
Private room     1213.0  3066.276939   5763
Entire home/apt  1536.0  3465.591404   8911
Hotel room       1700.0  3032.983025    648
```

- Hotel rooms are the most expensive on average, with a median price of 1700.0.
- Entire homes/apartments follow, with a median price of 1536.0.
- Private rooms are more affordable at 1213.0.
- Shared rooms are the cheapest, with a median price of 500.0

1. Room Size and Amenities: Larger accommodations, such as entire homes or apartments, tend to have more amenities, more space, and higher demand, which results in higher prices. Hotel rooms often provide additional services (e.g., daily cleaning, room service), which can also drive up the price.

2. Target Demographics: Shared rooms are typically budget-friendly options for travelers who prioritize cost over privacy, resulting in lower prices compared to private rooms and entire homes.

3. Demand Fluctuations: The demand for larger accommodations (e.g., entire homes) may fluctuate seasonally, and this can also contribute to the price differences across room types.

4.4 Availibility vs Room Type

Ho: There is no significant difference in the distributions of availability across different room types. Ha: There is a significant difference in the distributions of availability for at least one room type.

```
from scipy.stats import kruskal

# Group availability data by room type
groups = [dfc[dfc['room_type'] == room]['availability_365'] for room
in dfc['room_type'].unique()]

# Perform Kruskal-Wallis Test
stat, pvalue = kruskal(*groups)

print(f"Kruskal-Wallis Statistic: {stat}, p-Value: {pvalue}")

# Interpret the result
if pvalue > 0.05:
    print("Fail to reject H₀: No significant difference in
availability across room types.")
else:
    print("Reject H₀: Significant difference in availability across at
least one room type.")
```

```
Kruskal-Wallis Statistic: 341.7684751728655, p-Value:
9.036283137414042e-74
Reject H₀: Significant difference in availability across at least one
room type.
```

```
availability_stats = dfc.groupby('room_type')
['availability_365'].agg(['median', 'mean',
'count']).sort_values(by='median')
availability_stats
```

```
                 median        mean  count
room_type
Entire home/apt   231.0  219.102947   4818
Private room      329.0  269.327633   1462
Hotel room        338.0  269.536842    190
Shared room       351.0  282.280000     75
```

- Shared Room: Median (351 days): Highest among all room types, meaning most shared rooms are available nearly year-round. Insight: Shared rooms are likely less in demand, resulting in higher availability. This could reflect limited bookings or a niche customer base.

- Hotel Room: Median (338 days): Second-highest availability, close to that of shared rooms. Insight: Hotel rooms may cater to a broader market but still experience less fluctuation in availability.
- Private Room: Median (329 days): Slightly lower availability than shared and hotel rooms. Insight: Private rooms strike a balance, appealing to both short-term and long-term renters.
- Entire Home/Apartment: Median (231 days): Significantly lower than other room types. Insight: Entire homes/apartments are likely in higher demand, especially for families or groups, resulting in lower overall availability.

Customer Behavior and Market Dynamics

- High Availability for Shared and Hotel Rooms: These room types are either less popular or target specific segments (e.g., budget travelers or tourists). Listings may have fewer bookings due to niche appeal or higher supply relative to demand.
- Lower Availability for Entire Homes/Apartments: Reflects strong demand, possibly from families, long-term renters, or groups who prefer privacy. These properties are more frequently booked, especially during peak travel seasons.
- Private Rooms: Moderate availability suggests balanced demand, appealing to solo travelers or budget-conscious guests seeking privacy without the cost of an entire home.

```
import json

# Aggregate the data to get the average price per neighbourhood
price_mean = df.groupby('neighbourhood').agg({'price':
'mean'}).reset_index()

# Load the geojson file
geojson_path = r'C:\Users\putri\OneDrive\Desktop\Capstone 2\Bangkok-
districts.geojson'

with open(geojson_path, 'r') as f:
    districts_geojson = json.load(f)

# Merge the geojson and the price_mean data to include the average
price in the geojson properties
for feature in districts_geojson['features']:
    neighbourhood_name = feature['properties']['dname_e']
    match = price_mean[price_mean['neighbourhood'] ==
neighbourhood_name]
    if not match.empty:
        feature['properties']['average_price'] =
match['price'].values[0]
    else:
```

```
        feature['properties']['average_price'] = 'N/A'

# Bangkok coordinate
lat = 13.736717
long = 100.523186

# Create a Folium map for average price
bangkok_map = folium.Map(
    location=[lat, long],
    zoom_start=10,
    dragging=False,
    zoomControl=True,
    scrollWheelZoom=False,
    doubleClickZoom=False
)
tiles = 'https://tile.openstreetmap.de/{z}/{x}/{y}.png'
attr = 'Map <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
folium.TileLayer(tiles=tiles, attr=attr).add_to(bangkok_map)

# Add a choropleth layer to the map
choropleth = folium.Choropleth(
    geo_data=districts_geojson,
    name='choropleth',
    data=price_mean,
    columns=['neighbourhood', 'price'],
    key_on='feature.properties.dname_e',  # Key for matching the
geojson properties
    fill_color='Set1',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Average Airbnb Price'
).add_to(bangkok_map)

# Add tooltips
folium.GeoJson(
    districts_geojson,
    style_function=lambda feature: {
        'fillColor': 'transparent',
        'color': 'transparent',
        'weight': 0,
    },
    tooltip=folium.GeoJsonTooltip(
        fields=['dname_e', 'average_price'],
        aliases=['Neighbourhood', 'Average Price'],
        localize=True,

        sticky=False
    )
```

```
).add_to(bangkok_map)

# Display the map
bangkok_map
```

```
<folium.folium.Map at 0x1e53cbe6840>
```

4.5 Neighbourhood vs Price

H0: There is no significant difference in price across neighborhoods. Ha: There is a significant difference in price across neighborhoods.

```
from scipy.stats import kruskal

# Group price by neighborhood
neighborhood_groups = [dfc[dfc['neighbourhood'] == neighborhood]
['price'] for neighborhood in dfc['neighbourhood'].unique()]

# Perform Kruskal-Wallis Test
stat, pvalue = kruskal(*neighborhood_groups)

# Output the results
print(f"Kruskal-Wallis Statistic: {stat}, p-Value: {pvalue}")

# Interpret the results
if pvalue > 0.05:
    print("Fail to reject H0: There is no significant difference in
price across neighborhoods.")
else:
    print("Reject H0: There is a significant difference in price
across neighborhoods.")
```

```
Kruskal-Wallis Statistic: 1296.1885953130375, p-Value:
4.207113778142266e-242
Reject H0: There is a significant difference in price across
neighborhoods.
```

```
neighborhood_prices = dfc.groupby('neighbourhood')
['price'].median().reset_index()
neighborhood_prices.columns = ['neighbourhood', 'median_price']
```

1. Desirability: Certain neighborhoods may be more popular due to their proximity to major tourist attractions, cultural sites, or business districts, driving up demand and increasing prices.
2. Amenities and Services: Some neighborhoods may have better infrastructure, high-end services, or luxury accommodations that justify higher prices.
3. Local Economy: Neighborhoods with higher disposable incomes or a higher number of business travelers may have higher prices compared to others with less demand.

4. Supply and Demand: The supply of listings in different neighborhoods may also vary. A lower supply in a high-demand neighborhood could lead to higher prices.

4.6 Neighbourhood vs Room Type

Ho: The distribution of room types (e.g., Entire home/apt, Private room, Shared room) is independent of the neighborhood. Ha: The distribution of room types depends on the neighborhood.

```
contingency_table = pd.crosstab(dfc['neighbourhood'],
dfc['room_type'])

chi2_stat, pvalue, dof, expected = chi2_contingency(contingency_table)

print(f"Chi-Square Statistic: {chi2_stat:.3f}, p-Value: {pvalue:.3f},
Degrees of Freedom: {dof}")
print("Expected Frequencies Table:")
print(expected)

# Interpret the results
if pvalue > 0.05:
    print("Fail to reject H0: Room type distribution is independent of
the neighborhood.")
else:
    print("Reject H0: Room type distribution depends on the
neighborhood.")
```

```
Chi-Square Statistic: 1155.326, p-Value: 0.000, Degrees of Freedom:
135
Expected Frequencies Table:
[[5.00571429e+01 1.97402597e+00 1.51896104e+01 7.79220779e-01]
 [1.25142857e+01 4.93506494e-01 3.79740260e+00 1.94805195e-01]
 [1.98756303e+01 7.83804431e-01 6.03116883e+00 3.09396486e-01]
 [3.45983193e+01 1.36440031e+00 1.04987013e+01 5.38579068e-01]
 [2.20840336e+00 8.70893812e-02 6.70129870e-01 3.43773873e-02]
 [1.41337815e+02 5.57372040e+00 4.28883117e+01 2.20015279e+00]
 [5.30016807e+01 2.09014515e+00 1.60831169e+01 8.25057296e-01]
 [2.66480672e+02 1.05087853e+01 8.08623377e+01 4.14820474e+00]
 [7.58218487e+01 2.99006875e+00 2.30077922e+01 1.18029030e+00]
 [3.01815126e+01 1.19022154e+00 9.15844156e+00 4.69824293e-01]
 [1.54588235e+01 6.09625668e-01 4.69090909e+00 2.40641711e-01]
 [7.36134454e+00 2.90297937e-01 2.23376623e+00 1.14591291e-01]
 [1.21462185e+02 4.78991597e+00 3.68571429e+01 1.89075630e+00]
 [1.76672269e+01 6.96715050e-01 5.36103896e+00 2.75019099e-01]
 [1.38393277e+02 5.45760122e+00 4.19948052e+01 2.15431627e+00]
 [2.28201681e+01 8.99923606e-01 6.92467532e+00 3.55233002e-01]
 [8.09747899e+00 3.19327731e-01 2.45714286e+00 1.26050420e-01]
 [2.90773109e+02 1.14667685e+01 8.82337662e+01 4.52635600e+00]
 [8.83361345e+00 3.48357525e-01 2.68051948e+00 1.37509549e-01]
 [1.47226891e+00 5.80595875e-02 4.46753247e-01 2.29182582e-02]
```

```
 [8.46554622e+01 3.33842628e+00 2.56883117e+01 1.31779985e+00]
 [9.29001681e+02 3.66355997e+01 2.81901299e+02 1.44614209e+01]
 [2.57647059e+01 1.01604278e+00 7.81818182e+00 4.01069519e-01]
 [4.04873950e+01 1.59663866e+00 1.22857143e+01 6.30252101e-01]
 [7.36134454e+00 2.90297937e-01 2.23376623e+00 1.14591291e-01]
 [8.83361345e+00 3.48357525e-01 2.68051948e+00 1.37509549e-01]
 [1.62685714e+02 6.41558442e+00 4.93662338e+01 2.53246753e+00]
 [4.12235294e+01 1.62566845e+00 1.25090909e+01 6.41711230e-01]
 [1.19253782e+02 4.70282659e+00 3.61870130e+01 1.85637892e+00]
 [1.58268908e+02 6.24140565e+00 4.80259740e+01 2.46371276e+00]
 [1.54588235e+02 6.09625668e+00 4.69090909e+01 2.40641711e+00]
 [3.90151261e+01 1.53857907e+00 1.18389610e+01 6.07333843e-01]
 [2.42924370e+01 9.57983193e-01 7.37142857e+00 3.78151261e-01]
 [1.10420168e+01 4.35446906e-01 3.35064935e+00 1.71886937e-01]
 [3.84998319e+02 1.51825821e+01 1.16825974e+02 5.99312452e+00]
 [2.94453782e+00 1.16119175e-01 8.93506494e-01 4.58365164e-02]
 [2.87092437e+01 1.13216196e+00 8.71168831e+00 4.46906035e-01]
 [4.41680672e+00 1.74178762e-01 1.34025974e+00 6.87547746e-02]
 [3.15065546e+02 1.24247517e+01 9.56051948e+01 4.90450726e+00]
 [7.72941176e+01 3.04812834e+00 2.34545455e+01 1.20320856e+00]
 [4.41680672e+00 1.74178762e-01 1.34025974e+00 6.87547746e-02]
 [7.36134454e-01 2.90297937e-02 2.23376623e-01 1.14591291e-02]
 [5.07932773e+01 2.00305577e+00 1.54129870e+01 7.90679908e-01]
 [7.69996639e+02 3.03651642e+01 2.33651948e+02 1.19862490e+01]
 [1.10420168e+01 4.35446906e-01 3.35064935e+00 1.71886937e-01]
 [4.26957983e+01 1.68372804e+00 1.29558442e+01 6.64629488e-01]]
Reject H0: Room type distribution depends on the neighborhood.
```

- Location: Certain neighborhoods may cater to a specific type of traveler. For example: Tourist-heavy areas may have more entire homes/apartments and hotel rooms for privacy and comfort. Budget-conscious areas may have more shared rooms and private rooms for affordability.
- Demand and Supply: The demand for different room types in each neighborhood may influence the availability of these rooms. More tourists or business travelers in certain neighborhoods might lead to more hotel rooms and entire apartments, while other areas may focus on shared accommodations for budget travelers.
- Local Preferences: The preferences of the local population or long-term renters may also affect the room type distribution. For example, areas with more young professionals might have more private rooms, while student-heavy neighborhoods might see more shared rooms.

```
contingency_table = pd.crosstab(dfc['neighbourhood'],
dfc['review_month'])
from scipy.stats import chi2_contingency
chi2_stat, pvalue, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-Square Statistic: {chi2_stat:.3f}, p-Value: {pvalue:.3f},
```

```
Degrees of Freedom: {dof}")
print("Expected Frequencies Table:")
print(expected)

if pvalue > 0.05:
    print("Fail to reject H₀: Review month distribution is independent of the neighborhood.")
else:
    print("Reject H₀: Review month distribution depends on the neighborhood.")
```

```
Chi-Square Statistic: 700.877, p-Value: 0.000, Degrees of Freedom: 495
Expected Frequencies Table:
[[6.23376623e-01 3.32467532e-01 4.36363636e-01 6.33766234e-01
  8.00000000e-01 1.23636364e+00 2.11948052e+00 2.76363636e+00
  3.61558442e+00 5.69350649e+00 1.57506494e+01 3.39948052e+01]
 [1.55844156e-01 8.31168831e-02 1.09090909e-01 1.58441558e-01
  2.00000000e-01 3.09090909e-01 5.29870130e-01 6.90909091e-01
  9.03896104e-01 1.42337662e+00 3.93766234e+00 8.49870130e+00]
 [2.47517189e-01 1.32009167e-01 1.73262032e-01 2.51642475e-01
  3.17647059e-01 4.90909091e-01 8.41558442e-01 1.09732620e+00
  1.43559969e+00 2.26065699e+00 6.25393430e+00 1.34979374e+01]
 [4.30863254e-01 2.29793736e-01 3.01604278e-01 4.38044309e-01
  5.52941176e-01 8.54545455e-01 1.46493506e+00 1.91016043e+00
  2.49900688e+00 3.93521772e+00 1.08864782e+01 2.34964095e+01]
 [2.75019099e-02 1.46676853e-02 1.92513369e-02 2.79602750e-02
  3.52941176e-02 5.45454545e-02 9.35064935e-02 1.21925134e-01
  1.59511077e-01 2.51184110e-01 6.94881589e-01 1.49977082e+00]
 [1.76012223e+00 9.38731856e-01 1.23208556e+00 1.78945760e+00
  2.25882353e+00 3.49090909e+00 5.98441558e+00 7.80320856e+00
  1.02087089e+01 1.60757830e+01 4.44724217e+01 9.59853323e+01]
 [6.60045837e-01 3.52024446e-01 4.62032086e-01 6.71046600e-01
  8.47058824e-01 1.30909091e+00 2.24415584e+00 2.92620321e+00
  3.82826585e+00 6.02841864e+00 1.66771581e+01 3.59944996e+01]
 [3.31856379e+00 1.76990069e+00 2.32299465e+00 3.37387319e+00
  4.25882353e+00 6.58181818e+00 1.12831169e+01 1.47122995e+01
  1.92476700e+01 3.03095493e+01 8.38490451e+01 1.80972345e+02]
 [9.44232238e-01 5.03590527e-01 6.60962567e-01 9.59969442e-01
  1.21176471e+00 1.87272727e+00 3.21038961e+00 4.18609626e+00
  5.47654698e+00 8.62398778e+00 2.38576012e+01 5.14921314e+01]
 [3.75859435e-01 2.00458365e-01 2.63101604e-01 3.82123759e-01
  4.82352941e-01 7.45454545e-01 1.27792208e+00 1.66631016e+00
  2.17998472e+00 3.43284950e+00 9.49671505e+00 2.04968678e+01]
 [1.92513369e-01 1.02673797e-01 1.34759358e-01 1.95721925e-01
  2.47058824e-01 3.81818182e-01 6.54545455e-01 8.53475936e-01
  1.11657754e+00 1.75828877e+00 4.86417112e+00 1.04983957e+01]
 [9.16730328e-02 4.88922842e-02 6.41711230e-02 9.32009167e-02
  1.17647059e-01 1.81818182e-01 3.11688312e-01 4.06417112e-01
  5.31703591e-01 8.37280367e-01 2.31627196e+00 4.99923606e+00]
 [1.51260504e+00 8.06722689e-01 1.05882353e+00 1.53781513e+00
```

```
  1.94117647e+00 3.00000000e+00 5.14285714e+00 6.70588235e+00
  8.77310924e+00 1.38151261e+01 3.82184874e+01 8.24873950e+01]
 [2.20015279e-01 1.17341482e-01 1.54010695e-01 2.23682200e-01
  2.82352941e-01 4.36363636e-01 7.48051948e-01 9.75401070e-01
  1.27608862e+00 2.00947288e+00 5.55905271e+00 1.19981665e+01]
 [1.72345302e+00 9.19174943e-01 1.20641711e+00 1.75217723e+00
  2.21176471e+00 3.41818182e+00 5.85974026e+00 7.64064171e+00
  9.99602750e+00 1.57408709e+01 4.35459129e+01 9.39856379e+01]
 [2.84186402e-01 1.51566081e-01 1.98930481e-01 2.88922842e-01
  3.64705882e-01 5.63636364e-01 9.66233766e-01 1.25989305e+00
  1.64828113e+00 2.59556914e+00 7.18044309e+00 1.54976318e+01]
 [1.00840336e-01 5.37815126e-02 7.05882353e-02 1.02521008e-01
  1.29411765e-01 2.00000000e-01 3.42857143e-01 4.47058824e-01
  5.84873950e-01 9.21008403e-01 2.54789916e+00 5.49915966e+00]
 [3.62108480e+00 1.93124523e+00 2.53475936e+00 3.68143621e+00
  4.64705882e+00 7.18181818e+00 1.23116883e+01 1.60534759e+01
  2.10022918e+01 3.30725745e+01 9.14927426e+01 1.97469824e+02]
 [1.10007639e-01 5.86707410e-02 7.70053476e-02 1.11841100e-01
  1.41176471e-01 2.18181818e-01 3.74025974e-01 4.87700535e-01
  6.38044309e-01 1.00473644e+00 2.77952636e+00 5.99908327e+00]
 [1.83346066e-02 9.77845684e-03 1.28342246e-02 1.86401833e-02
  2.35294118e-02 3.63636364e-02 6.23376623e-02 8.12834225e-02
  1.06340718e-01 1.67456073e-01 4.63254393e-01 9.99847212e-01]
 [1.05423988e+00 5.62261268e-01 7.37967914e-01 1.07181054e+00
  1.35294118e+00 2.09090909e+00 3.58441558e+00 4.67379679e+00
  6.11459129e+00 9.62872422e+00 2.66371276e+01 5.74912147e+01]
 [1.15691367e+01 6.17020626e+00 8.09839572e+00 1.17619557e+01
  1.48470588e+01 2.29454545e+01 3.93350649e+01 5.12898396e+01
  6.71009931e+01 1.05664782e+02 2.92313522e+02 6.30903591e+02]
 [3.20855615e-01 1.71122995e-01 2.24598930e-01 3.26203209e-01
  4.11764706e-01 6.36363636e-01 1.09090909e+00 1.42245989e+00
  1.86096257e+00 2.93048128e+00 8.10695187e+00 1.74973262e+01]
 [5.04201681e-01 2.68907563e-01 3.52941176e-01 5.12605042e-01
  6.47058824e-01 1.00000000e+00 1.71428571e+00 2.23529412e+00
  2.92436975e+00 4.60504202e+00 1.27394958e+01 2.74957983e+01]
 [9.16730328e-02 4.88922842e-02 6.41711230e-02 9.32009167e-02
  1.17647059e-01 1.81818182e-01 3.11688312e-01 4.06417112e-01
  5.31703591e-01 8.37280367e-01 2.31627196e+00 4.99923606e+00]
 [1.10007639e-01 5.86707410e-02 7.70053476e-02 1.11841100e-01
  1.41176471e-01 2.18181818e-01 3.74025974e-01 4.87700535e-01
  6.38044309e-01 1.00473644e+00 2.77952636e+00 5.99908327e+00]
 [2.02597403e+00 1.08051948e+00 1.41818182e+00 2.05974026e+00
  2.60000000e+00 4.01818182e+00 6.88831169e+00 8.98181818e+00
  1.17506494e+01 1.85038961e+01 5.11896104e+01 1.10483117e+02]
 [5.13368984e-01 2.73796791e-01 3.59358289e-01 5.21925134e-01
  6.58823529e-01 1.01818182e+00 1.74545455e+00 2.27593583e+00
  2.97754011e+00 4.68877005e+00 1.29711230e+01 2.79957219e+01]
 [1.48510313e+00 7.92055004e-01 1.03957219e+00 1.50985485e+00
  1.90588235e+00 2.94545455e+00 5.04935065e+00 6.58395722e+00
```

```
  8.61359817e+00 1.35639419e+01 3.75236058e+01 8.09876241e+01]
 [1.97097021e+00 1.05118411e+00 1.37967914e+00 2.00381971e+00
  2.52941176e+00 3.90909091e+00 6.70129870e+00 8.73796791e+00
  1.14316272e+01 1.80015279e+01 4.97998472e+01 1.07483575e+02]
 [1.92513369e+00 1.02673797e+00 1.34759358e+00 1.95721925e+00
  2.47058824e+00 3.81818182e+00 6.54545455e+00 8.53475936e+00
  1.11657754e+01 1.75828877e+01 4.86417112e+01 1.04983957e+02]
 [4.85867074e-01 2.59129106e-01 3.40106952e-01 4.93964859e-01
  6.23529412e-01 9.63636364e-01 1.65194805e+00 2.15401070e+00
  2.81802903e+00 4.43758594e+00 1.22762414e+01 2.64959511e+01]
 [3.02521008e-01 1.61344538e-01 2.11764706e-01 3.07563025e-01
  3.88235294e-01 6.00000000e-01 1.02857143e+00 1.34117647e+00
  1.75462185e+00 2.76302521e+00 7.64369748e+00 1.64974790e+01]
 [1.37509549e-01 7.33384263e-02 9.62566845e-02 1.39801375e-01
  1.76470588e-01 2.72727273e-01 4.67532468e-01 6.09625668e-01
  7.97555386e-01 1.25592055e+00 3.47440794e+00 7.49885409e+00]
 [4.79449962e+00 2.55706646e+00 3.35614973e+00 4.87440794e+00
  6.15294118e+00 9.50909091e+00 1.63012987e+01 2.12556150e+01
  2.78080978e+01 4.37897632e+01 1.21141024e+02 2.61460046e+02]
 [3.66692131e-02 1.95569137e-02 2.56684492e-02 3.72803667e-02
  4.70588235e-02 7.27272727e-02 1.24675325e-01 1.62566845e-01
  2.12681436e-01 3.34912147e-01 9.26508785e-01 1.99969442e+00]
 [3.57524828e-01 1.90679908e-01 2.50267380e-01 3.63483575e-01
  4.58823529e-01 7.09090909e-01 1.21558442e+00 1.58502674e+00
  2.07364400e+00 3.26539343e+00 9.03346066e+00 1.94970206e+01]
 [5.50038197e-02 2.93353705e-02 3.85026738e-02 5.59205500e-02
  7.05882353e-02 1.09090909e-01 1.87012987e-01 2.43850267e-01
  3.19022154e-01 5.02368220e-01 1.38976318e+00 2.99954163e+00]
 [3.92360581e+00 2.09258976e+00 2.74652406e+00 3.98899924e+00
  5.03529412e+00 7.78181818e+00 1.33402597e+01 1.73946524e+01
  2.27569137e+01 3.58355997e+01 9.91364400e+01 2.13967303e+02]
 [9.62566845e-01 5.13368984e-01 6.73796791e-01 9.78609626e-01
  1.23529412e+00 1.90909091e+00 3.27272727e+00 4.26737968e+00
  5.58288770e+00 8.79144385e+00 2.43208556e+01 5.24919786e+01]
 [5.50038197e-02 2.93353705e-02 3.85026738e-02 5.59205500e-02
  7.05882353e-02 1.09090909e-01 1.87012987e-01 2.43850267e-01
  3.19022154e-01 5.02368220e-01 1.38976318e+00 2.99954163e+00]
 [9.16730328e-03 4.88922842e-03 6.41711230e-03 9.32009167e-03
  1.17647059e-02 1.81818182e-02 3.11688312e-02 4.06417112e-02
  5.31703591e-02 8.37280367e-02 2.31627196e-01 4.99923606e-01]
 [6.32543927e-01 3.37356761e-01 4.42780749e-01 6.43086325e-01
  8.11764706e-01 1.25454545e+00 2.15064935e+00 2.80427807e+00
  3.66875477e+00 5.77723453e+00 1.59822765e+01 3.44947288e+01]
 [9.58899924e+00 5.11413293e+00 6.71229947e+00 9.74881589e+00
  1.23058824e+01 1.90181818e+01 3.26025974e+01 4.25112299e+01
  5.56161956e+01 8.75795264e+01 2.42282047e+02 5.22920092e+02]
 [1.37509549e-01 7.33384263e-02 9.62566845e-02 1.39801375e-01
  1.76470588e-01 2.72727273e-01 4.67532468e-01 6.09625668e-01
  7.97555386e-01 1.25592055e+00 3.47440794e+00 7.49885409e+00]
```

```
 [5.31703591e-01 2.83575248e-01 3.72192513e-01 5.40565317e-01
  6.82352941e-01 1.05454545e+00 1.80779221e+00 2.35721925e+00
  3.08388083e+00 4.85622613e+00 1.34343774e+01 2.89955691e+01]]
Reject H₀: Review month distribution depends on the neighborhood.
```

4.6 Neighbourhood vs Room Type

H_0: Room availability is independent of the neighborhood. (There is no relationship between room availability and neighborhood.)

H_a: Room availability depends on the neighborhood. (There is a relationship between room availability and neighborhood.)

```
from scipy.stats import kruskal

# Group availability by neighborhood
neighborhood_groups = [dfc[dfc['neighbourhood'] == neighborhood]
['availability_365']
                       for neighborhood in
dfc['neighbourhood'].unique()]

# Perform Kruskal-Wallis Test
stat, pvalue = kruskal(*neighborhood_groups)

print(f"Kruskal-Wallis Statistic: {stat:.3f}, p-Value: {pvalue:.3f}")

if pvalue > 0.05:
    print("Fail to reject H₀: Room availability is independent of the
neighborhood.")
else:
    print("Reject H₀: Room availability depends on the neighborhood.")

Kruskal-Wallis Statistic: 344.132, p-Value: 0.000
Reject H₀: Room availability depends on the neighborhood.
```

H_0: The distribution of review months is independent of the neighborhood. (There is no relationship between neighborhood and review month.)

H_a: The distribution of review months depends on the neighborhood. (There is a relationship between neighborhood and review month.)

```
contingency_table = pd.crosstab(dfc['neighbourhood'],
dfc['review_month'])
from scipy.stats import chi2_contingency

chi2_stat, pvalue, dof, expected_ =
chi2_contingency(contingency_table)

print(f"Chi-Square Statistic: {chi2_stat:.3f}, p-Value: {pvalue:.3f},
Degrees of Freedom: {dof}")
```

```
print("Expected Frequencies Table:")
print(expected_)

if pvalue > 0.05:
    print("Fail to reject H₀: Review month distribution is independent of the neighborhood.")
else:
    print("Reject H₀: Review month distribution depends on the neighborhood.")
```

```
Chi-Square Statistic: 700.877, p-Value: 0.000, Degrees of Freedom: 495
Expected Frequencies Table:
[[6.23376623e-01 3.32467532e-01 4.36363636e-01 6.33766234e-01
  8.00000000e-01 1.23636364e+00 2.11948052e+00 2.76363636e+00
  3.61558442e+00 5.69350649e+00 1.57506494e+01 3.39948052e+01]
 [1.55844156e-01 8.31168831e-02 1.09090909e-01 1.58441558e-01
  2.00000000e-01 3.09090909e-01 5.29870130e-01 6.90909091e-01
  9.03896104e-01 1.42337662e+00 3.93766234e+00 8.49870130e+00]
 [2.47517189e-01 1.32009167e-01 1.73262032e-01 2.51642475e-01
  3.17647059e-01 4.90909091e-01 8.41558442e-01 1.09732620e+00
  1.43559969e+00 2.26065699e+00 6.25393430e+00 1.34979374e+01]
 [4.30863254e-01 2.29793736e-01 3.01604278e-01 4.38044309e-01
  5.52941176e-01 8.54545455e-01 1.46493506e+00 1.91016043e+00
  2.49900688e+00 3.93521772e+00 1.08864782e+01 2.34964095e+01]
 [2.75019099e-02 1.46676853e-02 1.92513369e-02 2.79602750e-02
  3.52941176e-02 5.45454545e-02 9.35064935e-02 1.21925134e-01
  1.59511077e-01 2.51184110e-01 6.94881589e-01 1.49977082e+00]
 [1.76012223e+00 9.38731856e-01 1.23208556e+00 1.78945760e+00
  2.25882353e+00 3.49090909e+00 5.98441558e+00 7.80320856e+00
  1.02087089e+01 1.60757830e+01 4.44724217e+01 9.59853323e+01]
 [6.60045837e-01 3.52024446e-01 4.62032086e-01 6.71046600e-01
  8.47058824e-01 1.30909091e+00 2.24415584e+00 2.92620321e+00
  3.82826585e+00 6.02841864e+00 1.66771581e+01 3.59944996e+01]
 [3.31856379e+00 1.76990069e+00 2.32299465e+00 3.37387319e+00
  4.25882353e+00 6.58181818e+00 1.12831169e+01 1.47122995e+01
  1.92476700e+01 3.03095493e+01 8.38490451e+01 1.80972345e+02]
 [9.44232238e-01 5.03590527e-01 6.60962567e-01 9.59969442e-01
  1.21176471e+00 1.87272727e+00 3.21038961e+00 4.18609626e+00
  5.47654698e+00 8.62398778e+00 2.38576012e+01 5.14921314e+01]
 [3.75859435e-01 2.00458365e-01 2.63101604e-01 3.82123759e-01
  4.82352941e-01 7.45454545e-01 1.27792208e+00 1.66631016e+00
  2.17998472e+00 3.43284950e+00 9.49671505e+00 2.04968678e+01]
 [1.92513369e-01 1.02673797e-01 1.34759358e-01 1.95721925e-01
  2.47058824e-01 3.81818182e-01 6.54545455e-01 8.53475936e-01
  1.11657754e+00 1.75828877e+00 4.86417112e+00 1.04983957e+01]
 [9.16730328e-02 4.88922842e-02 6.41711230e-02 9.32009167e-02
  1.17647059e-01 1.81818182e-01 3.11688312e-01 4.06417112e-01
  5.31703591e-01 8.37280367e-01 2.31627196e+00 4.99923606e+00]
 [1.51260504e+00 8.06722689e-01 1.05882353e+00 1.53781513e+00
  1.94117647e+00 3.00000000e+00 5.14285714e+00 6.70588235e+00
```

```
 8.77310924e+00 1.38151261e+01 3.82184874e+01 8.24873950e+01]
[2.20015279e-01 1.17341482e-01 1.54010695e-01 2.23682200e-01
 2.82352941e-01 4.36363636e-01 7.48051948e-01 9.75401070e-01
 1.27608862e+00 2.00947288e+00 5.55905271e+00 1.19981665e+01]
[1.72345302e+00 9.19174943e-01 1.20641711e+00 1.75217723e+00
 2.21176471e+00 3.41818182e+00 5.85974026e+00 7.64064171e+00
 9.99602750e+00 1.57408709e+01 4.35459129e+01 9.39856379e+01]
[2.84186402e-01 1.51566081e-01 1.98930481e-01 2.88922842e-01
 3.64705882e-01 5.63636364e-01 9.66233766e-01 1.25989305e+00
 1.64828113e+00 2.59556914e+00 7.18044309e+00 1.54976318e+01]
[1.00840336e-01 5.37815126e-02 7.05882353e-02 1.02521008e-01
 1.29411765e-01 2.00000000e-01 3.42857143e-01 4.47058824e-01
 5.84873950e-01 9.21008403e-01 2.54789916e+00 5.49915966e+00]
[3.62108480e+00 1.93124523e+00 2.53475936e+00 3.68143621e+00
 4.64705882e+00 7.18181818e+00 1.23116883e+01 1.60534759e+01
 2.10022918e+01 3.30725745e+01 9.14927426e+01 1.97469824e+02]
[1.10007639e-01 5.86707410e-02 7.70053476e-02 1.11841100e-01
 1.41176471e-01 2.18181818e-01 3.74025974e-01 4.87700535e-01
 6.38044309e-01 1.00473644e+00 2.77952636e+00 5.99908327e+00]
[1.83346066e-02 9.77845684e-03 1.28342246e-02 1.86401833e-02
 2.35294118e-02 3.63636364e-02 6.23376623e-02 8.12834225e-02
 1.06340718e-01 1.67456073e-01 4.63254393e-01 9.99847212e-01]
[1.05423988e+00 5.62261268e-01 7.37967914e-01 1.07181054e+00
 1.35294118e+00 2.09090909e+00 3.58441558e+00 4.67379679e+00
 6.11459129e+00 9.62872422e+00 2.66371276e+01 5.74912147e+01]
[1.15691367e+01 6.17020626e+00 8.09839572e+00 1.17619557e+01
 1.48470588e+01 2.29454545e+01 3.93350649e+01 5.12898396e+01
 6.71009931e+01 1.05664782e+02 2.92313522e+02 6.30903591e+02]
[3.20855615e-01 1.71122995e-01 2.24598930e-01 3.26203209e-01
 4.11764706e-01 6.36363636e-01 1.09090909e+00 1.42245989e+00
 1.86096257e+00 2.93048128e+00 8.10695187e+00 1.74973262e+01]
[5.04201681e-01 2.68907563e-01 3.52941176e-01 5.12605042e-01
 6.47058824e-01 1.00000000e+00 1.71428571e+00 2.23529412e+00
 2.92436975e+00 4.60504202e+00 1.27394958e+01 2.74957983e+01]
[9.16730328e-02 4.88922842e-02 6.41711230e-02 9.32009167e-02
 1.17647059e-01 1.81818182e-01 3.11688312e-01 4.06417112e-01
 5.31703591e-01 8.37280367e-01 2.31627196e+00 4.99923606e+00]
[1.10007639e-01 5.86707410e-02 7.70053476e-02 1.11841100e-01
 1.41176471e-01 2.18181818e-01 3.74025974e-01 4.87700535e-01
 6.38044309e-01 1.00473644e+00 2.77952636e+00 5.99908327e+00]
[2.02597403e+00 1.08051948e+00 1.41818182e+00 2.05974026e+00
 2.60000000e+00 4.01818182e+00 6.88831169e+00 8.98181818e+00
 1.17506494e+01 1.85038961e+01 5.11896104e+01 1.10483117e+02]
[5.13368984e-01 2.73796791e-01 3.59358289e-01 5.21925134e-01
 6.58823529e-01 1.01818182e+00 1.74545455e+00 2.27593583e+00
 2.97754011e+00 4.68877005e+00 1.29711230e+01 2.79957219e+01]
[1.48510313e+00 7.92055004e-01 1.03957219e+00 1.50985485e+00
 1.90588235e+00 2.94545455e+00 5.04935065e+00 6.58395722e+00
 8.61359817e+00 1.35639419e+01 3.75236058e+01 8.09876241e+01]
```

```
[1.97097021e+00 1.05118411e+00 1.37967914e+00 2.00381971e+00
 2.52941176e+00 3.90909091e+00 6.70129870e+00 8.73796791e+00
 1.14316272e+01 1.80015279e+01 4.97998472e+01 1.07483575e+02]
[1.92513369e+00 1.02673797e+00 1.34759358e+00 1.95721925e+00
 2.47058824e+00 3.81818182e+00 6.54545455e+00 8.53475936e+00
 1.11657754e+01 1.75828877e+01 4.86417112e+01 1.04983957e+02]
[4.85867074e-01 2.59129106e-01 3.40106952e-01 4.93964859e-01
 6.23529412e-01 9.63636364e-01 1.65194805e+00 2.15401070e+00
 2.81802903e+00 4.43758594e+00 1.22762414e+01 2.64959511e+01]
[3.02521008e-01 1.61344538e-01 2.11764706e-01 3.07563025e-01
 3.88235294e-01 6.00000000e-01 1.02857143e+00 1.34117647e+00
 1.75462185e+00 2.76302521e+00 7.64369748e+00 1.64974790e+01]
[1.37509549e-01 7.33384263e-02 9.62566845e-02 1.39801375e-01
 1.76470588e-01 2.72727273e-01 4.67532468e-01 6.09625668e-01
 7.97555386e-01 1.25592055e+00 3.47440794e+00 7.49885409e+00]
[4.79449962e+00 2.55706646e+00 3.35614973e+00 4.87440794e+00
 6.15294118e+00 9.50909091e+00 1.63012987e+01 2.12556150e+01
 2.78080978e+01 4.37897632e+01 1.21141024e+02 2.61460046e+02]
[3.66692131e-02 1.95569137e-02 2.56684492e-02 3.72803667e-02
 4.70588235e-02 7.27272727e-02 1.24675325e-01 1.62566845e-01
 2.12681436e-01 3.34912147e-01 9.26508785e-01 1.99969442e+00]
[3.57524828e-01 1.90679908e-01 2.50267380e-01 3.63483575e-01
 4.58823529e-01 7.09090909e-01 1.21558442e+00 1.58502674e+00
 2.07364400e+00 3.26539343e+00 9.03346066e+00 1.94970206e+01]
[5.50038197e-02 2.93353705e-02 3.85026738e-02 5.59205500e-02
 7.05882353e-02 1.09090909e-01 1.87012987e-01 2.43850267e-01
 3.19022154e-01 5.02368220e-01 1.38976318e+00 2.99954163e+00]
[3.92360581e+00 2.09258976e+00 2.74652406e+00 3.98899924e+00
 5.03529412e+00 7.78181818e+00 1.33402597e+01 1.73946524e+01
 2.27569137e+01 3.58355997e+01 9.91364400e+01 2.13967303e+02]
[9.62566845e-01 5.13368984e-01 6.73796791e-01 9.78609626e-01
 1.23529412e+00 1.90909091e+00 3.27272727e+00 4.26737968e+00
 5.58288770e+00 8.79144385e+00 2.43208556e+01 5.24919786e+01]
[5.50038197e-02 2.93353705e-02 3.85026738e-02 5.59205500e-02
 7.05882353e-02 1.09090909e-01 1.87012987e-01 2.43850267e-01
 3.19022154e-01 5.02368220e-01 1.38976318e+00 2.99954163e+00]
[9.16730328e-03 4.88922842e-03 6.41711230e-03 9.32009167e-03
 1.17647059e-02 1.81818182e-02 3.11688312e-02 4.06417112e-02
 5.31703591e-02 8.37280367e-02 2.31627196e-01 4.99923606e-01]
[6.32543927e-01 3.37356761e-01 4.42780749e-01 6.43086325e-01
 8.11764706e-01 1.25454545e+00 2.15064935e+00 2.80427807e+00
 3.66875477e+00 5.77723453e+00 1.59822765e+01 3.44947288e+01]
[9.58899924e+00 5.11413293e+00 6.71229947e+00 9.74881589e+00
 1.23058824e+01 1.90181818e+01 3.26025974e+01 4.25112299e+01
 5.56161956e+01 8.75795264e+01 2.42282047e+02 5.22920092e+02]
[1.37509549e-01 7.33384263e-02 9.62566845e-02 1.39801375e-01
 1.76470588e-01 2.72727273e-01 4.67532468e-01 6.09625668e-01
 7.97555386e-01 1.25592055e+00 3.47440794e+00 7.49885409e+00]
[5.31703591e-01 2.83575248e-01 3.72192513e-01 5.40565317e-01
```

```
  6.82352941e-01 1.05454545e+00 1.80779221e+00 2.35721925e+00
  3.08388083e+00 4.85622613e+00 1.34343774e+01 2.89955691e+01]]
Reject H₀: Review month distribution depends on the neighborhood.
```

```
dfc.to_csv('airbnb_clean.csv', index=False)
```

Section 5. Conclusion and Recommendation

5.1 Conclusion

1. Price Dynamics Across Availability Categories:

Price tends to decrease as availability increases: There is an inverse relationship between price and availability. Neighborhoods with higher availability of listings (more options available) tend to have lower prices, possibly due to increased competition among listings. Conversely, neighborhoods with lower availability may have higher prices due to limited supply.

1. Room Type Distribution Across Neighborhoods:

Room type distribution depends on the neighborhood: Different neighborhoods in Bangkok show distinct preferences for room types. Tourist-heavy areas (e.g., near attractions) tend to have more hotel rooms and entire homes/apartments, catering to travelers seeking privacy and comfort. Budget-focused neighborhoods often have a higher proportion of shared rooms and private rooms, attracting cost-conscious travelers or long-term renters. This suggests that location significantly impacts the types of accommodations available, and hosts may need to tailor their offerings based on neighborhood characteristics.

1. Price Differences Across Room Types:

Price varies significantly between room types: Entire homes/apartments have the highest median price, followed by hotel rooms, private rooms, and shared rooms. This insight aligns with the understanding that larger or more private accommodations (e.g., entire homes or hotel rooms) are typically more expensive, while shared rooms are more budget-friendly.

1. Room Type Preference Depends on Review Month:

Room type preference is influenced by the review month: There is a significant variation in room type distribution across different months, possibly tied to seasonality (e.g., higher demand for hotel rooms or entire apartments during peak tourist seasons) or travel trends.

1. Price Differences Across Neighborhoods:

Significant price differences across neighborhoods: Certain neighborhoods, especially those closer to tourist attractions or commercial areas, have higher median prices for Airbnb listings. This indicates a higher demand in these areas due to their location advantages.

1. Customer Behavior and Preferences:

Price and availability dynamics shape customer behavior: The interplay of price and availability is central to understanding customer choices. Areas with more affordable pricing and high availability (such as shared rooms or private rooms) attract budget-conscious travelers, while more expensive areas (near tourist spots) attract those willing to pay higher prices for greater privacy or luxury (e.g., hotel rooms, entire homes). Booking trends are seasonal, with variations in room type preferences and price sensitivity depending on the month. This is likely driven by factors like tourist seasons and local events.

5.2 Recommendation

1. Optimize Pricing Strategy Based on Availability and Neighborhood

Target price optimization in high-availability areas: In neighborhoods with higher availability of listings (more options available), consider offering competitive pricing strategies. Hosts in these areas may need discounts or promotions to stand out and attract customers who have more options. Highlight affordability in marketing campaigns for these areas, emphasizing value for money. Leverage scarcity in low-availability areas:

In neighborhoods with limited listings, consider premium pricing strategies to capitalize on the scarcity of supply. Highlight exclusivity in marketing messages and showcase the unique aspects of these listings (e.g., luxury homes, one-of-a-kind experiences).

2. Tailor Marketing Campaigns Based on Room Type Preferences

Room Type Segmentation: Budget-conscious travelers: Promote shared rooms and private rooms to cost-conscious travelers, particularly in local or budget-focused neighborhoods. Highlight affordability and flexibility. High-end or family travelers: Promote entire homes and hotel rooms in more tourist-heavy and luxury neighborhoods, focusing on comfort, privacy, and premium amenities. Targeted Ads:

Use dynamic pricing and personalized advertising strategies to match customers with room types based on their budget and preferences. For example, show budget listings to users browsing low-cost accommodations, and premium listings to users seeking a more luxurious experience.

3. Promote Listings Based on Review Months and Seasonality

Seasonal Campaigns: Understand the seasonality of neighborhoods and room types (e.g., higher demand for hotel rooms during peak seasons). Adjust marketing efforts to push specific room types during high-traffic months (e.g., entire homes during peak tourist seasons or private rooms in the off-season). Promote early bird offers and special discounts for bookings made well in advance, especially during peak months when competition for listings is higher. Influence of Reviews:

Consider using review data to influence marketing strategies. Listings with positive reviews could be highlighted more in targeted campaigns, building trust and social proof in specific neighborhoods and room types.

4. Geotargeting and Localized Marketing

Localized Campaigns Based on Neighborhood: Promote neighborhoods with higher demand for specific room types (e.g., higher-priced listings near tourist hotspots or business districts). Customize campaigns to appeal to tourists and business travelers based on their interests (e.g., proximity to landmarks, transportation options, or business hubs). For budget-focused neighborhoods, emphasize community atmosphere, local experiences, and affordable accommodations that appeal to backpackers or long-term travelers.

5. Address Customer Preferences in Marketing Messaging

Focus on Price and Availability Balance: For neighborhoods with higher availability, emphasize the range of options available at different price points. Encourage customers to explore listings in up-and-coming areas where they may get better value. For areas with limited availability, highlight the unique characteristics of the properties, such as luxurious amenities or exclusive experiences that come with the higher price tag. Promote Experience over Price:

For premium room types like entire homes and hotel rooms, focus on the experience (e.g., privacy, amenities, local culture). Customers paying higher prices tend to value the overall experience and the comfort offered by these listings.

6. Improve Search and Discovery Features

Search Filters Based on Price and Availability: Ensure that search filters reflect the user's preference for price range and room types, particularly in areas with fluctuating availability. Consider integrating price sensitivity into the search algorithm, making it easier for users to find listings that match their budget and availability preferences.

7. Increase Host Engagement and Education

Host Training: Train hosts in high-demand neighborhoods to optimize their pricing strategies by offering promotions and dynamic pricing. Educate hosts in budget-focused areas on how to market their rooms effectively, highlighting value and affordability in their listings.

Pricing Tools: Introduce or enhance dynamic pricing tools for hosts to automatically adjust rates based on seasonality, demand, and competition in their neighborhoods.