Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01

August 29, 2022

1 Introduction

2 Airbnb, as in "Air Bed and Breakfast," is a service that lets property owners rent out their spaces to travelers looking for a place to stay. Travelers can rent a space for multiple people to share, a shared space with private rooms, or the entire property for themselves. The model also gives you the opportunity to customize and personalize your guests' experience the way you want. Airbnb was started in 2008 by Brian Chesky and Joe Gebbia, based in San Fransisco California.The platform is accessible via website and mobile app.

3 Problem Statement

3.1 Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present a more unique, personalized way of experiencing the world. Today, Airbnb became one of a kind service that is used and recognized by the whole world. Data analysis on millions of listings provided through Airbnb is a crucial factor for the company. These millions of listings generate a lot of data - data that can be analyzed and used for security, business decisions, understanding of customers' and providers' (hosts) behavior and performance on the platform, guiding marketing initiatives, implementation of innovative additional services and much more.

3.2 This dataset has around 49,000 observations in it with 16 columns and it is a mix between categorical and numeric values.

3.3 Explore and analyze the data to discover key understandings (not limited to these) such as :

- What can we learn about different hosts and areas?
- What can we learn from predictions? (ex: locations, prices, reviews, etc)

- Which hosts are the busiest and why?
- Is there any noticeable difference of traffic among different areas and what could be the reason for it?

3.4 Capstone Project_01 - 'Exploratory Data Analysis of Airbnb booking dataset'

3.4.1 We are going to find answers to the following questions -

1. Which is the prefered location according to average best price?
2. Where are most of the hosts located?
3. The highest and lowest rent paying locations by customers
4. Most Popular/demanded host based on reviews and availability 365 days
5. Establishing relation between neighbourhood group and availability of rooms.
6. Which are the top hosts, neighbourhoods, neighbourhood groups based on their turnover?
7. Room type selection based on price, availability on 365 days.
8. Top ten neighbourhood based on listing price.
9. Distribution of properties based on Mandatory stays.
10. Type of Visit based on Mandatory stay allowed for single booking.

[27]:
```
# Import the necessary python libraries
import numpy as np                              # Handles arrays and␣
↪mathematical operations
import matplotlib.pyplot as plt                 # Creates 2D graphs and arrays
import pandas as pd                             # Data handling and wrangling
import seaborn as sns                           # Statistical graphical␣
↪distributions
```

[28]:
```
# Mount Google Drive to read data available
from google.colab import drive
drive.mount('/content/drive', force_remount = True)
```

```
Mounted at /content/drive
```

[30]:
```
# From pandas read csv file
df = pd.read_csv('/content/drive/MyDrive/Colab_Notebooks/
↪Capstone_Project01_EDA_Siddhartha_Pasayat/Airbnb_NYC_2019.csv')
```

[31]:
```
# Check first 5 instances of data
df.head()
```

[31]:
```
     id                                              name  host_id  \
0  2539                Clean & quiet apt home by the park     2787
1  2595                             Skylit Midtown Castle     2845
2  3647               THE VILLAGE OF HARLEM…NEW YORK !      4632
3  3831                   Cozy Entire Floor of Brownstone     4869
4  5022  Entire Apt: Spacious Studio/Loft by central park     7192
```

```
     host_name neighbourhood_group neighbourhood  latitude  longitude  \
0         John            Brooklyn    Kensington  40.64749  -73.97237
1     Jennifer           Manhattan       Midtown  40.75362  -73.98377
2    Elisabeth           Manhattan        Harlem  40.80902  -73.94190
3  LisaRoxanne            Brooklyn  Clinton Hill  40.68514  -73.95976
4        Laura           Manhattan   East Harlem  40.79851  -73.94399

         room_type  price  minimum_nights  number_of_reviews last_review  \
0     Private room    149               1                  9  2018-10-19
1  Entire home/apt    225               1                 45  2019-05-21
2     Private room    150               3                  0         NaN
3  Entire home/apt     89               1                270  2019-07-05
4  Entire home/apt     80              10                  9  2018-11-19

   reviews_per_month  calculated_host_listings_count  availability_365
0               0.21                               6               365
1               0.38                               2               355
2                NaN                               1               365
3               4.64                               1               194
4               0.10                               1                 0
```

[32]:
```
# Check the size of Dataset
df.shape
```

[32]:
```
(48895, 16)
```

[33]:
```
# Check non- null count, data type in columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   name                            48879 non-null  object
 2   host_id                         48895 non-null  int64
 3   host_name                       48874 non-null  object
 4   neighbourhood_group             48895 non-null  object
 5   neighbourhood                   48895 non-null  object
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object
 9   price                           48895 non-null  int64
 10  minimum_nights                  48895 non-null  int64
 11  number_of_reviews               48895 non-null  int64
```

```
 12  last_review                     38843 non-null  object
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

[34]:
```python
# Check column names in dataset
df.columns
```

[34]:
```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365'],
      dtype='object')
```

4 Data set has following features -

1) id - Unique id identifying Airbnb listing
2) name - Represents accomodation
3) host_id - Unique id identifying Airbnb Host
4) host_name - Name under whom host is registered
5) neighbourhood_group - A group of area
6) neighbourhood - neighbourhood_group falls under area
7) latitude- coordinates of listing
8) longitude- coordinates of listing
9) room_type- types of accomodation present
10) price- tariff of listing
11) minimum_nights- minimum nights required to stay during single visit
12) number_of_reviews - total count of reviews given by customers
13) last_review - date of last reviews given
14) review_per_month - reviews recieved per month
15) calculated_host_listings_count - total number of listing registered under host name
16) availability_365 - number of days host/property is available throughout the year

[35]:
```python
# Check for any null values if present in columns
df.isnull().sum()
```

[35]:
```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

4.0.1 Columns like name, host name, last_review and reviews_per_month have null values.

[36]:
```
# Use fillna() method to replace the NULL values with a specified value.
df.fillna(0, inplace=True)
```

[37]:
```
# Check  again for null values
df.isnull().sum()
```

[37]:
```
id                                0
name                              0
host_id                           0
host_name                         0
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
last_review                       0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

[38]:
```
# Describe function is used to get a descriptive status of the dataframe.
df.describe()
```

[38]:
```
                 id       host_id      latitude     longitude         price  \
count  4.889500e+04  4.889500e+04  48895.000000  48895.000000  48895.000000
mean   1.901714e+07  6.762001e+07     40.728949    -73.952170    152.720687
std    1.098311e+07  7.861097e+07      0.054530      0.046157    240.154170
min    2.539000e+03  2.438000e+03     40.499790    -74.244420      0.000000
25%    9.471945e+06  7.822033e+06     40.690100    -73.983070     69.000000
50%    1.967728e+07  3.079382e+07     40.723070    -73.955680    106.000000
75%    2.915218e+07  1.074344e+08     40.763115    -73.936275    175.000000
max    3.648724e+07  2.743213e+08     40.913060    -73.712990  10000.000000

       minimum_nights  number_of_reviews  reviews_per_month  \
count    48895.000000       48895.000000       48895.000000
mean         7.029962          23.274466           1.090910
std         20.510550          44.550582           1.597283
min          1.000000           0.000000           0.000000
25%          1.000000           1.000000           0.040000
50%          3.000000           5.000000           0.370000
75%          5.000000          24.000000           1.580000
max       1250.000000         629.000000          58.500000

       calculated_host_listings_count  availability_365
count                    48895.000000      48895.000000
mean                         7.143982        112.781327
std                         32.952519        131.622289
min                          1.000000          0.000000
25%                          1.000000          0.000000
50%                          1.000000         45.000000
75%                          2.000000        227.000000
max                        327.000000        365.000000
```

4.0.2 We see that minimum price is zero which is not possible and max value of minimum nights is 1250 which is not possible.So we assign 100$ to minimum price and setting a limit of minimum_nights not exceeding 365.

[39]:
```
# Use dropna() to remove rows having null values
df.dropna().head()
```

[39]:
```
     id                                              name  host_id  \
0  2539                Clean & quiet apt home by the park     2787
1  2595                             Skylit Midtown Castle     2845
2  3647               THE VILLAGE OF HARLEM…NEW YORK !      4632
3  3831                   Cozy Entire Floor of Brownstone     4869
4  5022  Entire Apt: Spacious Studio/Loft by central park     7192
```

```
     host_name neighbourhood_group neighbourhood  latitude  longitude  \
0         John            Brooklyn    Kensington  40.64749  -73.97237
1     Jennifer           Manhattan       Midtown  40.75362  -73.98377
2    Elisabeth           Manhattan        Harlem  40.80902  -73.94190
3  LisaRoxanne            Brooklyn  Clinton Hill  40.68514  -73.95976
4        Laura           Manhattan   East Harlem  40.79851  -73.94399

         room_type  price  minimum_nights  number_of_reviews last_review  \
0     Private room    149               1                  9  2018-10-19
1  Entire home/apt    225               1                 45  2019-05-21
2     Private room    150               3                  0           0
3  Entire home/apt     89               1                270  2019-07-05
4  Entire home/apt     80              10                  9  2018-11-19

   reviews_per_month  calculated_host_listings_count  availability_365
0               0.21                               6               365
1               0.38                               2               355
2               0.00                               1               365
3               4.64                               1               194
4               0.10                               1                 0
```

[40]:
```python
# Define a function to correct minimum price , we replace where price is zero␣
↪to 100$
def price_correction(z):
    if z==0:
        return 100
    else:
        return z
```

[41]:
```python
# Identify Rows which have 'price'=0
df[df['price']==0]
```

[41]:
```
             id                                               name    host_id  \
23161  18750597  Huge Brooklyn Brownstone Living, Close to it all.    8993084
25433  20333471       Hostel Style Room | Ideal Traveling Buddies  131697576
25634  20523843    MARTIAL LOFT 3: REDEMPTION (upstairs, 2nd room)   15787004
25753  20608117                       Sunny, Quiet Room in Greenpoint    1641537
25778  20624541      Modern apartment in the heart of Williamsburg   10132166
25794  20639628  Spacious comfortable master bedroom with nice …   86327101
25795  20639792  Contemporary bedroom in brownstone with nice view   86327101
25796  20639914       Cozy yet spacious private brownstone bedroom   86327101
26259  20933849                              the best you can find   13709292
26841  21291569  Coliving in Brooklyn! Modern design / Shared room  101970559
26866  21304320             Best Coliving space ever! Shared room.  101970559

          host_name neighbourhood_group       neighbourhood  latitude  \
```

23161	Kimberly	Brooklyn	Bedford-Stuyvesant	40.69023
25433	Anisha	Bronx	East Morrisania	40.83296
25634	Martial Loft	Brooklyn	Bushwick	40.69467
25753	Lauren	Brooklyn	Greenpoint	40.72462
25778	Aymeric	Brooklyn	Williamsburg	40.70838
25794	Adeyemi	Brooklyn	Bedford-Stuyvesant	40.68173
25795	Adeyemi	Brooklyn	Bedford-Stuyvesant	40.68279
25796	Adeyemi	Brooklyn	Bedford-Stuyvesant	40.68258
26259	Qiuchi	Manhattan	Murray Hill	40.75091
26841	Sergii	Brooklyn	Bushwick	40.69211
26866	Sergii	Brooklyn	Bushwick	40.69166

	longitude	room_type	price	minimum_nights	number_of_reviews \
23161	-73.95428	Private room	0	4	1
25433	-73.88668	Private room	0	2	55
25634	-73.92433	Private room	0	2	16
25753	-73.94072	Private room	0	2	12
25778	-73.94645	Entire home/apt	0	5	3
25794	-73.91342	Private room	0	1	93
25795	-73.91170	Private room	0	1	95
25796	-73.91284	Private room	0	1	95
26259	-73.97597	Entire home/apt	0	3	0
26841	-73.90670	Shared room	0	30	2
26866	-73.90928	Shared room	0	30	5

	last_review	reviews_per_month	calculated_host_listings_count \
23161	2018-01-06	0.05	4
25433	2019-06-24	2.56	4
25634	2019-05-18	0.71	5
25753	2017-10-27	0.53	2
25778	2018-01-02	0.15	1
25794	2019-06-15	4.28	6
25795	2019-06-21	4.37	6
25796	2019-06-23	4.35	6
26259	0	0.00	1
26841	2019-06-22	0.11	6
26866	2019-05-24	0.26	6

	availability_365
23161	28
25433	127
25634	0
25753	0
25778	73
25794	176
25795	232
25796	222

```
26259                  0
26841                333
26866                139
```

[42]:
```python
# Replace all price = 0 by price = 100 $
df['price']=df['price'].apply(price_correction)
```

[43]:
```python
df['price'].isnull().sum()
```

[43]: 0

[44]:
```python
# Maximum stay can't be  greater than 365 days hence we have to define a␣
↪function to set maximum of minimum_night to 365
def minimum_night_count(y):
 if y > 365:
    y==365
 else:
    y==y
    return y
```

[45]:
```python
# Apply Maximum of Minimum_nights to 365
df['minimum_nights']= df['minimum_nights'].apply(minimum_night_count)
```

[46]:
```python
# Check whether the corrected values in the particular features have been␣
↪updated in dataframe
df.describe()
```

[46]:

	id	host_id	latitude	longitude	price \
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.743184
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.144546
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	10.000000
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000

	minimum_nights	number_of_reviews	reviews_per_month \
count	48881.000000	48895.000000	48895.000000
mean	6.840429	23.274466	1.090910
std	16.452017	44.550582	1.597283
min	1.000000	0.000000	0.000000
25%	1.000000	1.000000	0.040000
50%	3.000000	5.000000	0.370000
75%	5.000000	24.000000	1.580000
max	365.000000	629.000000	58.500000

```
       calculated_host_listings_count  availability_365
count                    48895.000000      48895.000000
mean                         7.143982        112.781327
std                         32.952519        131.622289
min                          1.000000          0.000000
25%                          1.000000          0.000000
50%                          1.000000         45.000000
75%                          2.000000        227.000000
max                        327.000000        365.000000
```

4.0.3 Now the above data in the dataframe is ready for analysis.

5 1.Which is the prefered location according to average best price?

[47]:
```
# Get the prefered location by using groupby method
avg_preffered_price_df = df.groupby(['neighbourhood_group','room_type'],␣
↪as_index=False)['price'].mean()
avg_preffered_price_df.columns= [x.replace('neighbourhood_group','location')␣
↪for x in list(avg_preffered_price_df.columns)]
avg_preffered_price_df
```

[47]:

	location	room_type	price
0	Bronx	Entire home/apt	127.506596
1	Bronx	Private room	66.941718
2	Bronx	Shared room	59.800000
3	Brooklyn	Entire home/apt	178.338006
4	Brooklyn	Private room	76.559317
5	Brooklyn	Shared room	51.012107
6	Manhattan	Entire home/apt	249.246685
7	Manhattan	Private room	116.776622
8	Manhattan	Shared room	88.977083
9	Queens	Entire home/apt	147.050573
10	Queens	Private room	71.762456
11	Queens	Shared room	69.020202
12	Staten Island	Entire home/apt	173.846591
13	Staten Island	Private room	62.292553
14	Staten Island	Shared room	57.444444

[48]:
```
avg_preffered_price_df.sort_values('price', ascending=False)[0:5]
```

[48]:

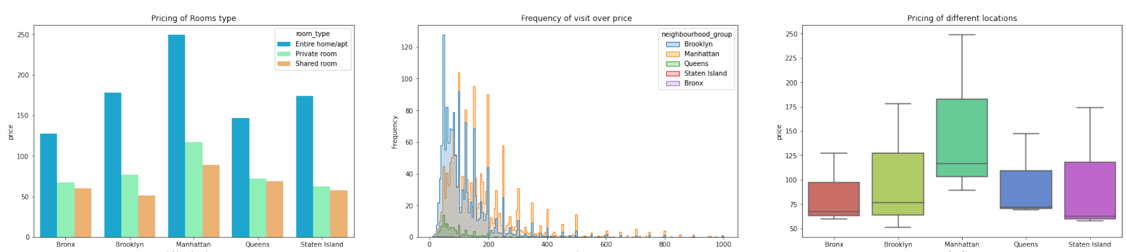
	location	room_type	price
6	Manhattan	Entire home/apt	249.246685
3	Brooklyn	Entire home/apt	178.338006
12	Staten Island	Entire home/apt	173.846591

```
9          Queens  Entire home/apt  147.050573
0           Bronx  Entire home/apt  127.506596
```

5.0.1 Inferences-

1. Top 5 locations based on average price are Manhatton, Brooklyn, Staten Island, Queens and Bronx

[49]:
```python
# let us plot the various graphs to find out relation between neighbourhood groups , room types and price
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(30, 6))
ax = axes.flatten()
mean_price_df = df.groupby(['neighbourhood_group', 'room_type'], as_index=False)[['price']].mean()
sns.barplot(data=mean_price_df, x='neighbourhood_group', y ='price', hue='room_type', ax=ax[0], palette='rainbow')
sns.boxplot(x='location', y='price', data= avg_preffered_price_df, palette='hls')
sns.histplot(data=df[df['price'] < 1000], x="price", hue="neighbourhood_group", stat='frequency', element='step', ax=ax[1])
ax[0].set_title("Pricing of Rooms type");
ax[1].set_title("Frequency of visit over price ");
ax[2].set_title("Pricing of different locations");
```

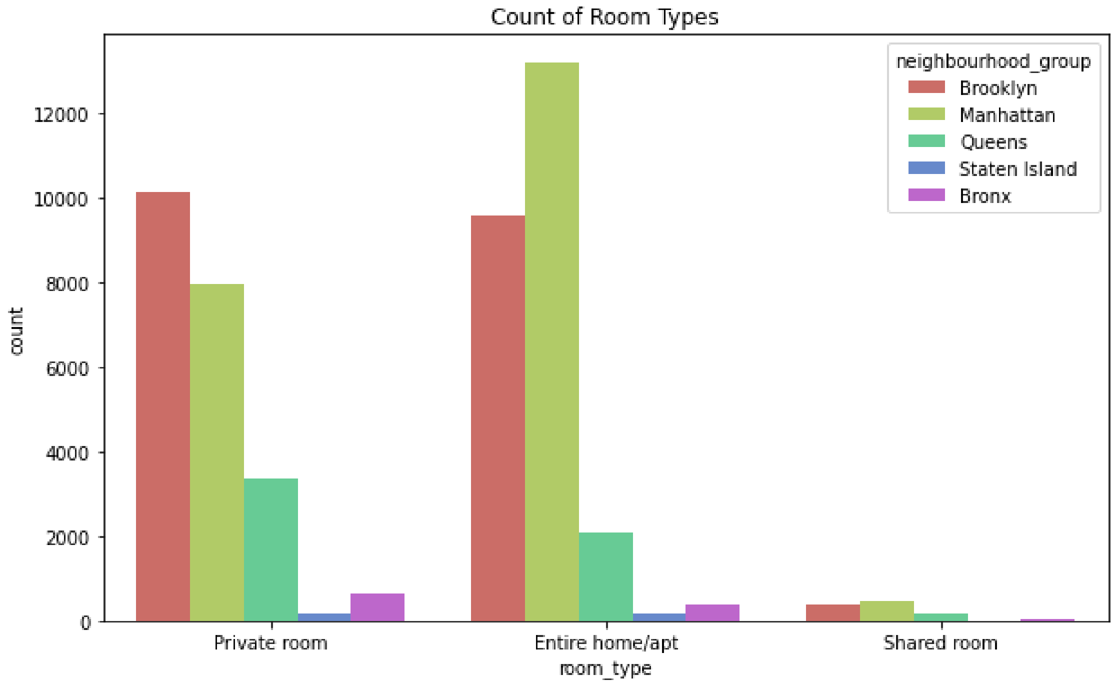


[50]:
```python
room_type_data=df['room_type'].value_counts()
room_type_data
```

[50]:
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

[51]:
```python
# Plot countplot for the visualisation
plt.figure(figsize=(10,6))
sns.countplot(x=df['room_type'],hue=df['neighbourhood_group'], palette='hls')
plt.title('Count of Room Types')
```

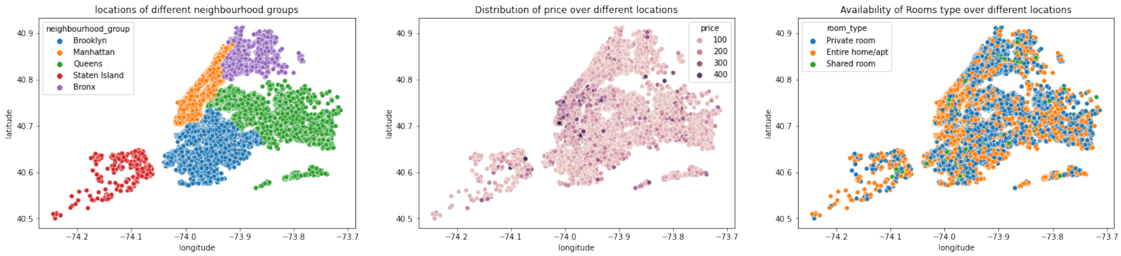
[51]: Text(0.5, 1.0, 'Count of Room Types')



5.0.2 Inferences-

1. Manhattan (Neighbourhood Group) is more prefered in all types of rooms
2. Pricing of Manhattan group is high as compared to other groups
3. Pricing and count of Entire home/ apartment is high as compared to shared room and private rooms in all locations
4. Count of Private room is more in Brooklyn than entire home or apartment

[52]:
```
# Plot scatterplots for the visualisation
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(25, 5))
ax = axes.flatten()
sns.scatterplot(data=df, x='longitude', y='latitude', hue='neighbourhood_group'␣
 ↪,ax=ax[0])
sns.scatterplot(data=df[df['price']<500], x='longitude', y='latitude',␣
 ↪hue='price', ax=ax[1])
sns.scatterplot(data=df, x='longitude', y='latitude', hue='room_type', ax=ax[2])
ax[0].set_title("locations of different neighbourhood groups");
ax[1].set_title("Distribution of price over different locations");
ax[2].set_title("Availability of Rooms type over different locations");
```



[53]:
```
#A pairplot plot a pairwise relationships in a dataset. here we can see␣
↪distribution of each pair with neighbourhood groups.
sns.pairplot(data=df, hue='neighbourhood_group')
plt.title('Pair plot with different Location')
```

Output hidden; open in https://colab.research.google.com to view.

5.0.3 Inferences-

1. From the above pair plot we can conclude that most of the customers are visiting to Manhattan followed by Brooklyn
2. Pricing, minimum night stay, average listing price is more for Manhattan region.

6 2. Where are most of the hosts located?

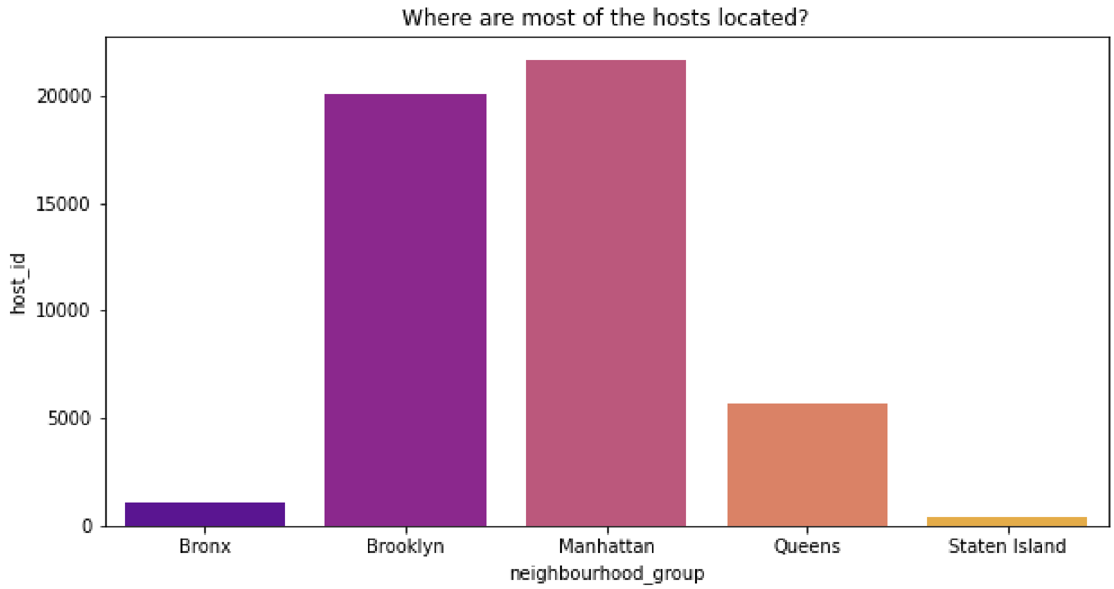
[54]:
```
# Find most active hosts by groupby method
active_host= df.groupby('neighbourhood_group', as_index= False)['host_id'].
↪count()
active_host.sort_values('host_id', ascending=False)
```

[54]:

	neighbourhood_group	host_id
2	Manhattan	21661
1	Brooklyn	20104
3	Queens	5666
0	Bronx	1091
4	Staten Island	373

[55]:
```
# Plot barplot for the visualisation
plt.figure(figsize=(10,5))
sns.barplot(y='host_id',x= 'neighbourhood_group', data= active_host,␣
↪palette='plasma')
plt.title('Where are most of the hosts located?')
```

[55]:
```
Text(0.5, 1.0, 'Where are most of the hosts located?')
```

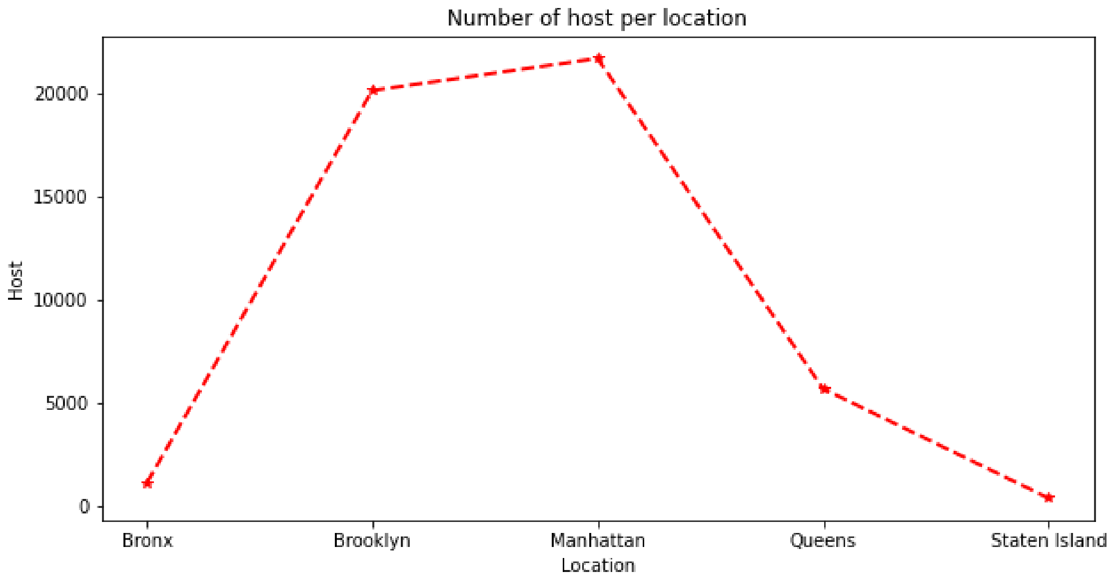


[56]:
```
# Identify active hosts using groupby method
no_of_active_host= df.groupby('neighbourhood_group')['host_id'].count()
no_of_active_host
```

[56]:
```
neighbourhood_group
Bronx             1091
Brooklyn         20104
Manhattan        21661
Queens            5666
Staten Island      373
Name: host_id, dtype: int64
```

[57]:
```
# Graph
plt.figure(figsize=(10,5))
plt.plot(no_of_active_host, 'r*--', lw=2)
plt.title('Number of host per location')
plt.ylabel('Host')
plt.xlabel('Location')
```

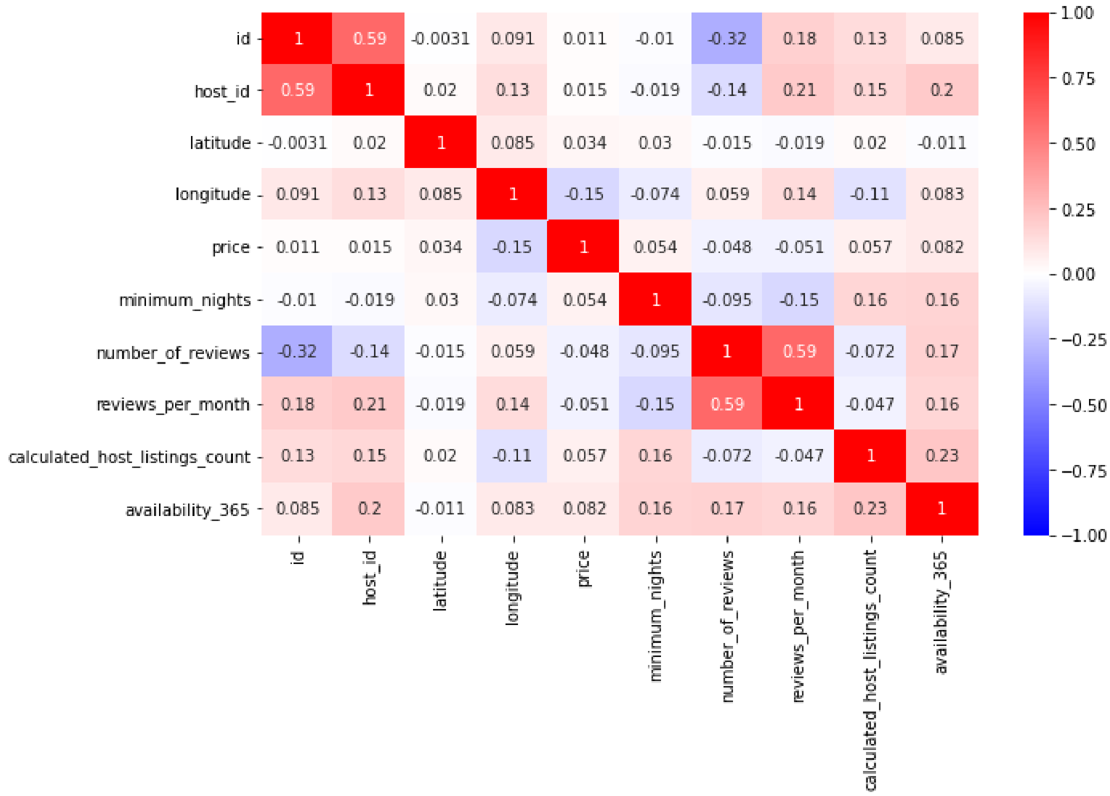
[57]:
```
Text(0.5, 0, 'Location')
```



6.0.1 Inferences-

1. Manhattan has highest numbers of hosts (21661) followed by Brooklyn (20104)

[58]:
```
# Let us plot a heatmap of correction of all variables in dataset. Use of␣
↪colourbar is for getting highest and lowest correlation
plt.figure(figsize=(10, 6))
heatmap = sns.heatmap(df.corr(), linewidths=0, vmin=-1, annot=True, cmap="bwr")
plt.show()
```



6.0.2 Inferences-

1. High correlation number represents high correlation between two variables eg. number of reviews and reviews per month has correction factor as 0.59 which represents they are highly correlated.
2. low correlation number represents less correlation between two variables. eg. host id and minimun nights has correlation factor -0.019 which represents they are not much dependent on each other

7 3. The highest and lowest rent paying locations by customers

[59]:
```python
#Get the highest rent according to location using groupby method
max_price_df = df.groupby('neighbourhood_group',as_index=False)['price'].max().sort_values(['price'],ascending = False).rename(columns = {'price':'Maximum price','neighbourhood_group':'Location'})
#Get the lowest rent according to location using gorupby method
min_price_df = df.groupby('neighbourhood_group',as_index=False)['price'].min().sort_values(['price'],ascending = True).rename(columns = {'price':'Minimum price','neighbourhood_group':'Location'})
```

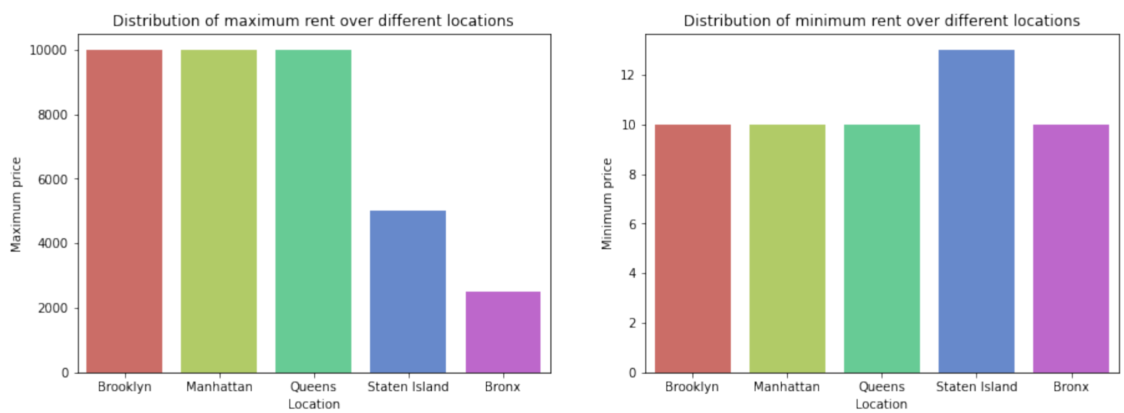
```
price_df= pd.merge(max_price_df, min_price_df, how= 'inner')
price_df
```

[59]:

	Location	Maximum price	Minimum price
0	Brooklyn	10000	10
1	Manhattan	10000	10
2	Queens	10000	10
3	Staten Island	5000	13
4	Bronx	2500	10

[60]:

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))
ax = axes.flatten()
sns.barplot(x='Location',y='Minimum price', data=price_df, ax=ax[1],␣
↪palette='hls')
sns.barplot(x='Location',y='Maximum price', data=price_df, ax=ax[0], palette=␣
↪'hls')
ax[0].set_title("Distribution of maximum rent over different locations");
ax[1].set_title("Distribution of minimum rent over different locations");
```



7.0.1 Inferences-

1. Customers are paying highest rent price of 10000 and lowest rent price of 10 at Manhattan ,Brooklyn and Queens location.

8 4.Most Popular/demanded host based on reviews and calculated host listings count

[61]:
```
#Get the host based on number of reviews using groupby method
host_based_on_review_df = df.
 ↪groupby(['host_id','host_name','calculated_host_listings_count'],as_index=False)['number_of
 ↪sum()

#Get the host based on availability in a year
host_based_on_availability_df = df.
 ↪groupby(['host_id','host_name','calculated_host_listings_count'],as_index=False)['calculate
 ↪count().sort_values(['calculated_host_listings_count'],ascending = False)

top_host_df= pd.merge(host_based_on_availability_df, host_based_on_review_df,␣
 ↪how='inner').
 ↪sort_values(['calculated_host_listings_count','number_of_reviews'],␣
 ↪ascending= False)
top_host_df.head()
```

[61]:
```
     host_id       host_name  calculated_host_listings_count  \
0  219517861    Sonder (NYC)                             327
1  107434423      Blueground                             232
2   30283594            Kara                             121
3  137358866          Kazuya                             103
4   16098958  Jeremy & Laura                              96

   number_of_reviews
0               1281
1                 29
2                 65
3                 87
4                138
```

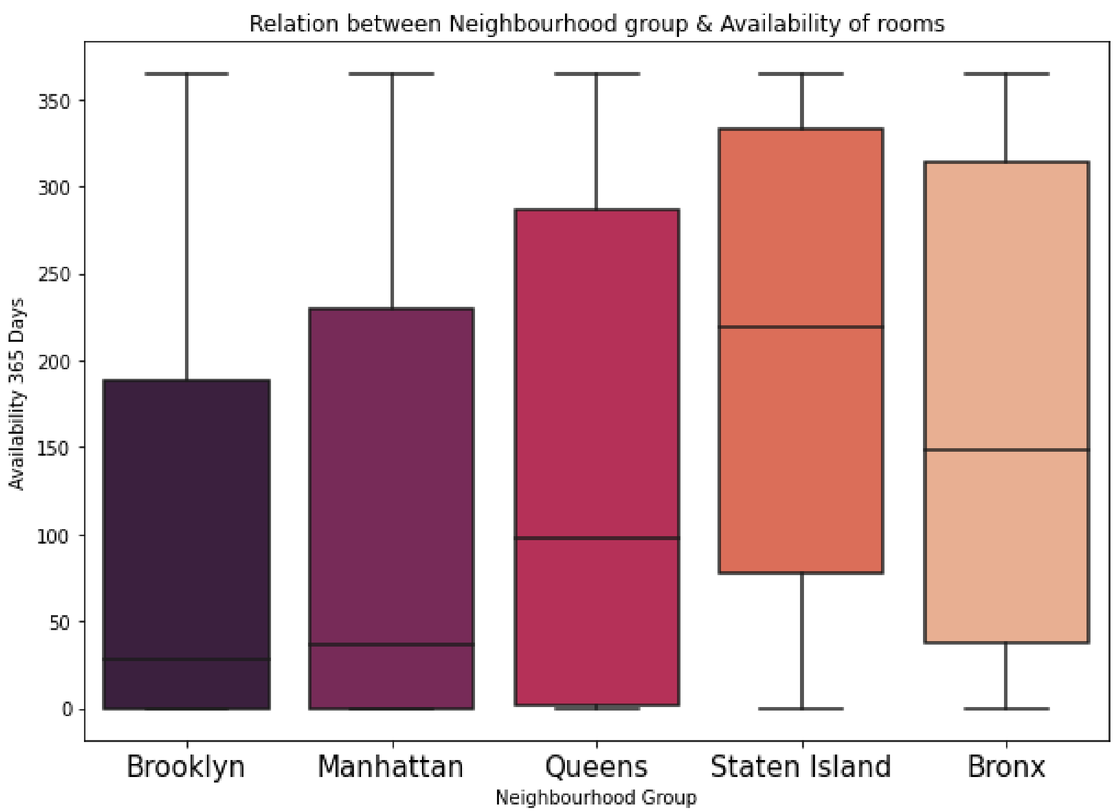
8.0.1 Inferences-

Top hosts based on reviews and calculated host listing count are Sonder, Blueground, Kara, Kazuya, Jeremy & Laura

9 5.Finding Relation between neighbourhood group and availability of rooms

[62]:
```
plt.figure(figsize=(10,7))
ax = sns.boxplot(data=df,␣
↪x='neighbourhood_group',y='availability_365',palette='rocket')

# Naming the Chart
ax.set_title('Relation between Neighbourhood group & Availability of rooms')

# Naming X & Y axis
ax.set_ylabel('Availability 365 Days')
ax.set_xlabel('Neighbourhood Group')

#Adjusting Bar labels
ax.set_xticklabels(ax.get_xticklabels(), size = '15')
```

[62]:
```
[Text(0, 0, 'Brooklyn'),
 Text(0, 0, 'Manhattan'),
 Text(0, 0, 'Queens'),
 Text(0, 0, 'Staten Island'),
 Text(0, 0, 'Bronx')]
```



9.0.1 Inferences-

1. Staten island has highest availability of rooms over 365 days followed by bronx.
2. Brooklyn and manhattan has least availability of rooms

10 6. Who are the top Hosts and which are the top Neighbourhoods, and Neighbourhood groups based on their turnover?

[63]:
```
# Find out Top hosts, neighbourhoods, neighbourhood groups based on turnover
top_host = df.groupby(['host_name','host_id'], as_index= False)['price'].sum().
↪reset_index().sort_values('price', ascending= False)
top_host.head()
```

[63]:

	index	host_name	host_id	price
33240	33240	Sonder (NYC)	219517861	82795
4876	4876	Blueground	107434423	70331
31247	31247	Sally	156158778	37097
29859	29859	Red Awning	205031545	35294
18986	18986	Kara	30283594	33581

10.0.1 Inferences-

Top hosts based on turnover are Sonder(NYC), Blueground, Sally, Red Awning and Kara.

[64]:
```
top_host_neighbourhood = df.groupby(['neighbourhood','host_id'], as_index=␣
↪False)['price'].sum().reset_index().sort_values('price', ascending= False)
top_host_neighbourhood.head()
```

[64]:

	index	neighbourhood	host_id	price
14252	14252	Financial District	219517861	57738
24660	24660	Midtown	205031545	35294
6912	6912	Chelsea	3750764	18780
31514	31514	Upper West Side	836168	15000
8144	8144	Clinton Hill	1177497	14850

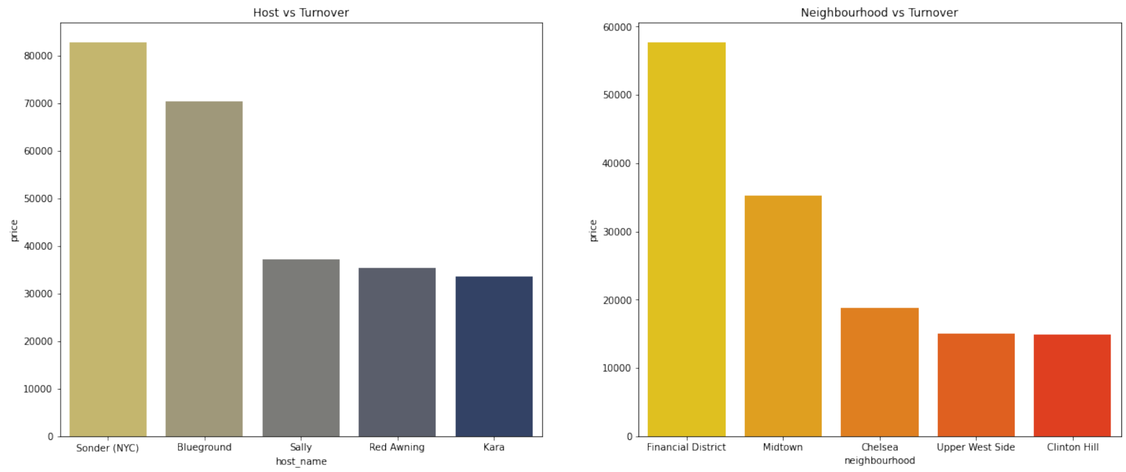
10.0.2 Inferences

1)Top neighbourhood are Financial District, Midtown, Chelsea, Upper West Side and Clinton Hill

2)All these neighbourhood belong to Manhattan neighbourhood group.

```
[65]: # Plot Barplot for visualization of the above analysis
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,8))
ax = axes.flatten()
sns.barplot(x='host_name', y='price', data = top_host.head() , ax=ax[0],␣
 ↪palette='cividis_r').set(title ='Host vs Turnover')
sns.barplot(x='neighbourhood', y='price', data = top_host_neighbourhood.head(),␣
 ↪ax=ax[1], palette='autumn_r').set(title ='Neighbourhood vs Turnover')
```

```
[65]: [Text(0.5, 1.0, 'Neighbourhood vs Turnover')]
```



10.0.3 Inferences-

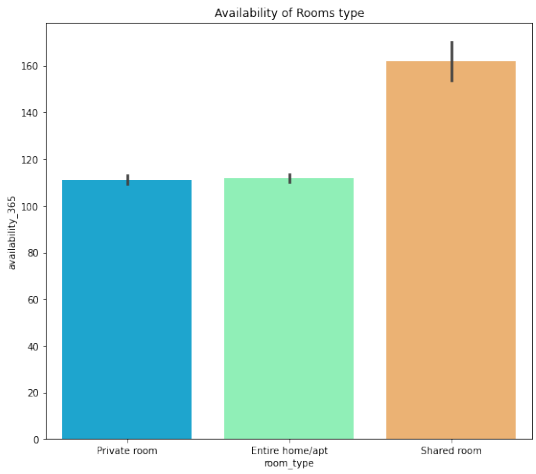
Financial District being Manhattan city's buzzing heart very aptly coincides with our analysis to be on the top in case of Turnover.

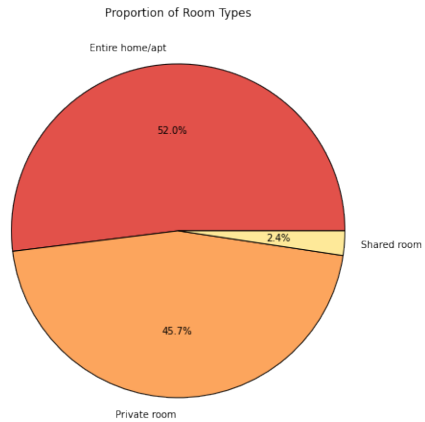
11 7. Room type selection based on price and it's availability on 365 days

```
[66]: # Plot Barplot and Pie Chart for Data Visualization
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20, 8))
ax = axes.flatten()

#define Seaborn color palette to use
colors = sns.color_palette('Spectral')[0:5]

sns.barplot(data=df, x='room_type', y='availability_365', ax=ax[0], palette =␣
 ↪'rainbow' )
ax[0].set_title("Availability of Rooms type");
```

```
labels = df['room_type'].value_counts().index
sizes = df['room_type'].value_counts().values
ax[1].pie(sizes, labels=labels, autopct='%1.1f%%',colors = colors,␣
↪wedgeprops={"edgecolor": "black"}, frame=False)
ax[1].set_title('Proportion of Room Types')
```

[66]: Text(0.5, 1.0, 'Proportion of Room Types')



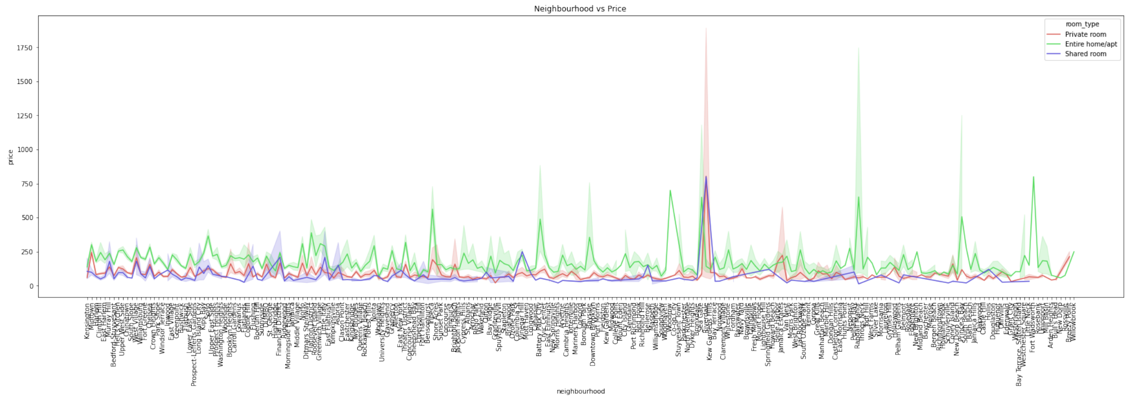


11.0.1 Inferences

1) Shared rooms are more available throughout the year as compared to Private rroms and Entire Home/Appartment making this easier for students or daily workers for their hault.
2) Entire Home/Apt and Private Rooms take a lion share in terms of their cummulative presence in the neighbourhoods.
3) Shared rooms have a meagre presence indicating not much demand of these rooms as nowadays less people are preferring to share common space and ammenities hence limiting it to a particular sect of customers.

[67]:
```
# Plot the line plot of dataframe for neighbourhood vs price
fig = plt.figure(figsize=(30, 8))
sns.lineplot(data=df, x='neighbourhood', y='price',␣
↪hue='room_type',palette="hls")
plt.xticks(rotation=90)
plt.title('Neighbourhood vs Price')
```

[67]: Text(0.5, 1.0, 'Neighbourhood vs Price')

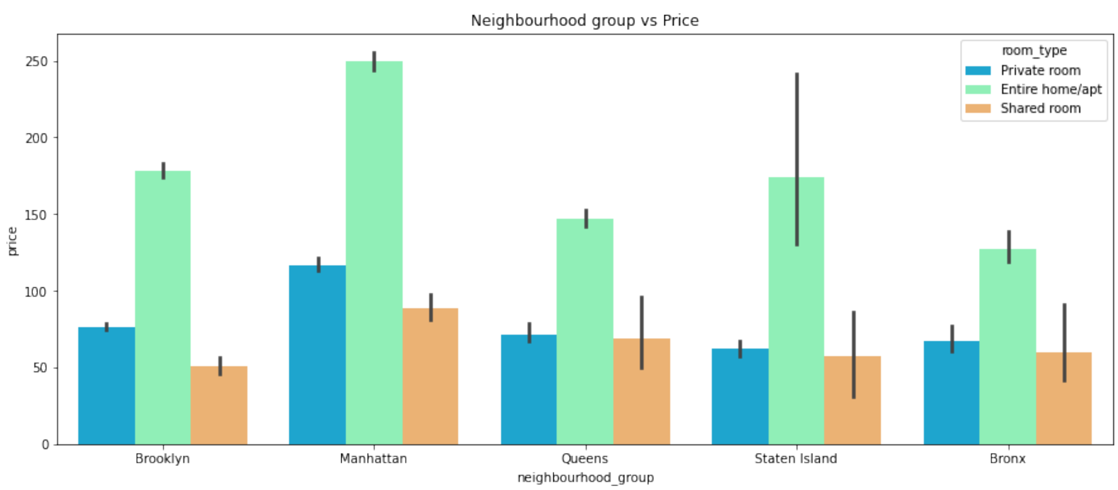


11.0.2 Inferences-

With this plot it is quite evident that Entire room/apt has all time high price throughout the neighbourhood.

[68]:
```
# Plot a barplot to visualize the neighbourhood and prices of various rooms
fig = plt.figure(figsize=(15, 6))
sns.barplot(data=df, x='neighbourhood_group', y='price', hue='room_type',␣
 ↪palette="rainbow")
plt.title('Neighbourhood group vs Price')
```

[68]:
```
Text(0.5, 1.0, 'Neighbourhood group vs Price')
```

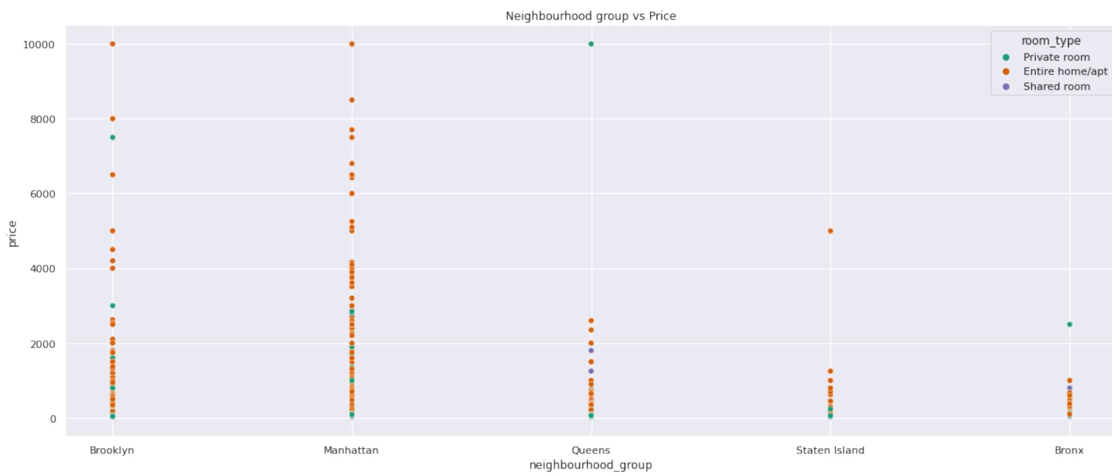


11.0.3 Inference

Entire home/apt has maintained higher price in all neighbourhoods and it is highest in Manhattan

[69]:
```
fig = plt.figure(figsize=(20, 8))
sns.set_theme(style="darkgrid")
sns.scatterplot(data=df, x='neighbourhood_group', y='price', hue='room_type',␣
 ↪palette="Dark2")
plt.title('Neighbourhood group vs Price')
```

[69]:
```
Text(0.5, 1.0, 'Neighbourhood group vs Price')
```



11.0.4 Inference

1) Manhattan and Brooklyn are posh areas with high end properties available.
2) The high end properties are mostly Entire Home/Apt.

12 8. Top ten neighbourbourhood based on listing price

[70]:
```
# Get top 10 neighbourhoods based on groupby method
top_ten_neighborhoods=df.groupby('neighbourhood')['price'].agg('median').
 ↪nlargest(n=10).sort_values(ascending = True)
top_ten_neighborhoods
```

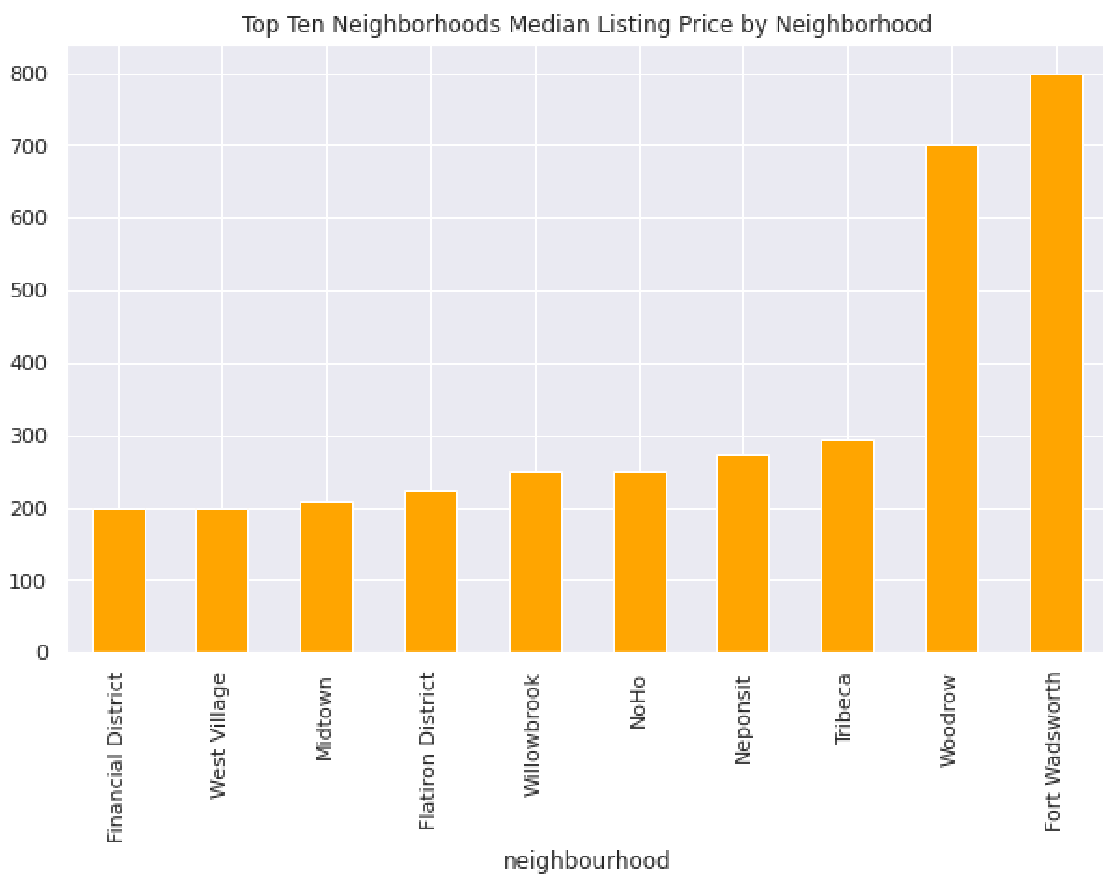
[70]:
```
neighbourhood
Financial District     200.0
West Village           200.0
Midtown                210.0
```

```
Flatiron District      225.0
Willowbrook            249.0
NoHo                   250.0
Neponsit               274.0
Tribeca                295.0
Woodrow                700.0
Fort Wadsworth         800.0
Name: price, dtype: float64
```

[71]:

```
# Plot a bar graph for the above visualization
top_ten_neighborhoods.plot(kind = 'bar', title = 'Top Ten Neighborhoods Median␣
↪Listing Price by Neighborhood', figsize=(10,6), color='orange')
```

[71]: <matplotlib.axes._subplots.AxesSubplot at 0x7effbbae8a90>



12.0.1 Inferences-

Fort Wadsworth and Woodrow are the two most expensive neighbourhoods listed belonging to Staten Island.

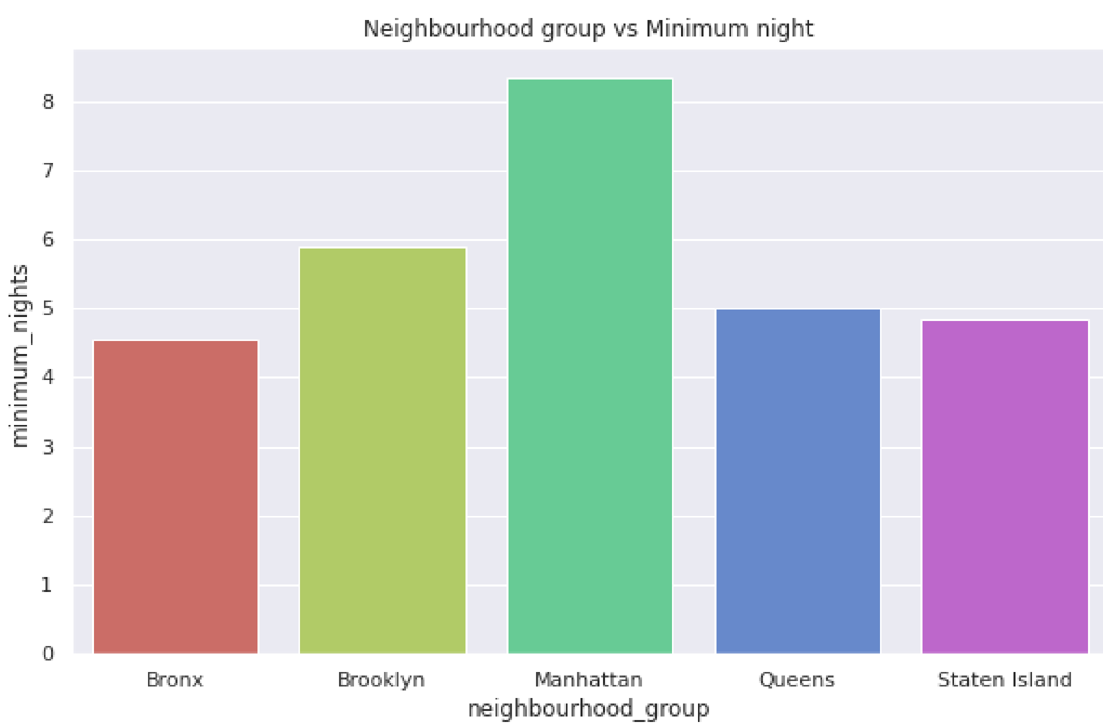
13 9. Distribution of neighbourhoods based on properties/hosts mandatory stays

[72]:
```
# Location where customers spends maximum mandatory nights
minimum_stay_df = df.groupby(('neighbourhood_group'),␣
 ↪as_index=False)['minimum_nights'].mean()
minimum_stay_df
```

[72]:
```
  neighbourhood_group  minimum_nights
0               Bronx        4.560953
1            Brooklyn        5.895711
2           Manhattan        8.345371
3              Queens        5.010240
4       Staten Island        4.831099
```

[73]:
```
# Plot a barplot for the above visualization
fig = plt.figure(figsize=(10, 6))
sns.barplot(data= minimum_stay_df, x='neighbourhood_group', y='minimum_nights',␣
 ↪palette="hls")
plt.title('Neighbourhood group vs Minimum night')
```

[73]:
```
Text(0.5, 1.0, 'Neighbourhood group vs Minimum night')
```



13.0.1 Inferences-

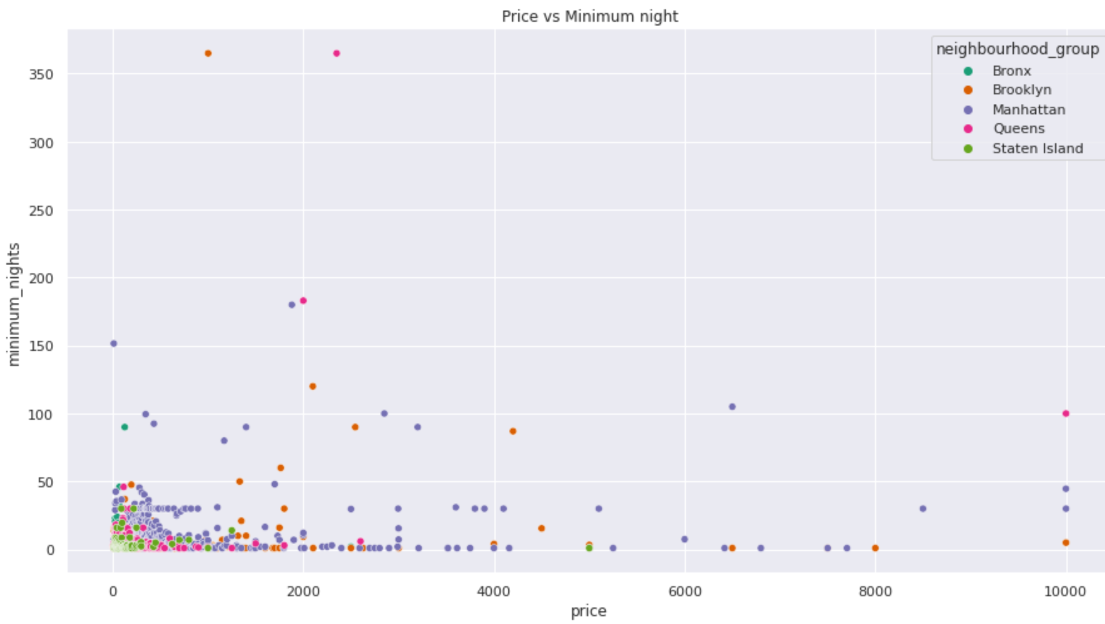
1) Most hosts allow mandatory stays less than 5 nights.
2) Manhattan has generally a higher average for mandatory nights required to stay followed by Brooklyn and Queens.

[74]:
```
# Location where customers spend mandatory nights along with its price and␣
↪neighbourhood
minimum_stayprice_df = df.groupby(['neighbourhood_group','price'],␣
↪as_index=False)['minimum_nights'].mean()
minimum_stayprice_df
```

[74]:
```
     neighbourhood_group  price  minimum_nights
0                  Bronx     10        1.000000
1                  Bronx     20        6.166667
2                  Bronx     21        1.000000
3                  Bronx     22        2.000000
4                  Bronx     23        2.000000
…                      …    …               …
1534       Staten Island    700        7.000000
1535       Staten Island    800        7.000000
1536       Staten Island   1000        1.000000
1537       Staten Island   1250       14.000000
1538       Staten Island   5000        1.000000

[1539 rows x 3 columns]
```

[75]:
```
# Plot a scatterplot for the visualization
fig = plt.figure(figsize=(15, 8))
sns.set_theme(style="darkgrid")
sns.scatterplot(data= minimum_stayprice_df, x='price', y='minimum_nights', hue␣
↪='neighbourhood_group', palette="Dark2",marker = 'o')
plt.title('Price vs Minimum night')
```

[75]:
```
Text(0.5, 1.0, 'Price vs Minimum night')
```



13.0.2 Inferences

1) Generally customers prefer to stay in accomodation having criteria for minimum number of mandatory stay and paying lesser price.
2) Manhattan has a wide spread of offerings both in terms of highest mandatory stay required to expensive listed properties.

14 10. Types of Visit based on Mandatory Stays allowed for a single booking.

[76]:
```
# Identify the type of visits allowed
visit_df = df.groupby(['neighbourhood_group'],␣
↪as_index=False)['minimum_nights'].mean()
visit_df
```

[76]:

	neighbourhood_group	minimum_nights
0	Bronx	4.560953
1	Brooklyn	5.895711
2	Manhattan	8.345371
3	Queens	5.010240
4	Staten Island	4.831099

[77]:
```
# Generate type of visit based upon stays allowed
visit_df['minimum_nights']
Trav_L = []
for i in visit_df['minimum_nights']:
  if i <= 5:
    Trav_L.append('short_term_visit')                     # Less than or equal to 5␣
↪days is Short Term Visit - For Business/Lesiure/Personal
  elif i > 5 and i <= 90:
    Trav_L.append('mid_term_visit')                       # Less than or equal to␣
↪90 days is Mid Term Visit - For Bagpackers
  else:
    Trav_L.append('long_term_visit')                      # More than 90 days is␣
↪Long Term Visit - For Nirvana (Soul Searching)
```

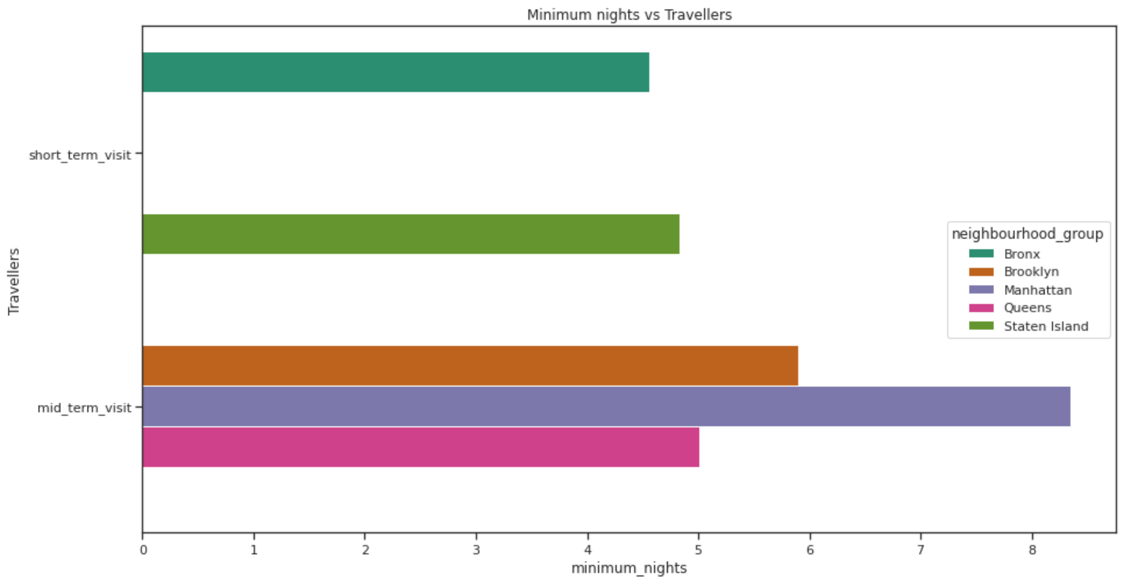
[78]:
```
# Add the column of Visit
visit_df['Travellers'] = Trav_L
visit_df
```

[78]:

	neighbourhood_group	minimum_nights	Travellers
0	Bronx	4.560953	short_term_visit
1	Brooklyn	5.895711	mid_term_visit
2	Manhattan	8.345371	mid_term_visit
3	Queens	5.010240	mid_term_visit
4	Staten Island	4.831099	short_term_visit

[79]:
```
# Plot barplot for the above visualization
fig = plt.figure(figsize=(15, 8))
sns.set_theme(style="ticks")
sns.barplot(data= visit_df, x='minimum_nights', y='Travellers', hue␣
↪='neighbourhood_group', palette="Dark2")
plt.title('Minimum nights vs Travellers')
```

[79]: Text(0.5, 1.0, 'Minimum nights vs Travellers')



14.0.1 Inferences

1) On the basis of hosts allowing minimum mandatory stay Manhattan, Queens and Brooklyn hosts prefer customers having a minimum 'Mid-term visit' whereas hosts in Bronx and Staten Island prefer customers having a minimum 'Short-term visit'.

2) Bronx and Staten Island can be preferred for shorter stays over other neighbourhoods making it budget friendly to some extent.

3) Manhattan and Brooklyn being posh areas and the implementaion of higher mandatory stays for single booking will be make these trips/visits expensive.

4) Different marketing initiatives can be rolled out based on the mandatory stay period in following neighbourhoods.

15 Scope and Limitations:

1. Datasets have limiting attributes to classify various categories of properties.
2. Customer experiential and Category wise ratings for Hosts seemed to be missing which could have played an important role in identifying Star Hosts.
3. A lot of guest information were missing like Purpose of Visit, Number of Guests, which could have given a sense of understanding about
4. the relation of customer footfall and neighbourhoods. Key attributes of properties like Number of Beds, Closets, Bathrooms, Gym, Sauna, Property Age, Distances from nearest Hospitals, Shopping Complexes, Airport, Station were missing.

16 Conclusion:

Manhattan and Brooklyn are the posh areas in NY as there is maximum footfall and properties based on prices and listings are are on the higher side. Manhattan and Brooklyn have the highest number of hosts. Manhattan has highest number of Private rooms and Entire House/Apt. in culmination followed by Brooklyn. Highest accommodations of 10,000 USD are available at Manhattan, Brooklyn and Queens. Most popular hosts are Sonder, Blueground ,Kara to name a few based on number of reviews and calculated host listing counts. Staten Island seems more to be available for booking throughout the year compared to other neighbourhoods. Sonder,Blueground ,Sally are some of the top hosts based on their turnover. Financial District, Midtown, Chelsea are some of the top neighbourhood based on their turnover. Shared rooms are mostly available over other room types and Entire Home /Apt which has the highest proportion of room share are mostly on the expensive ends. Fort Wadsworth and Woodrow are expensive neighbourhood based on median listed price belonging to Staten Island. Most hosts allow a minimum 5 nights mandatory stay for single booking but the average increases in case of Manhattan, Brooklyn and Queens. Bronx and Staten Island are mostly preferred for Shorter visits and onwards and others are for slightly longer stays.

17 Colab to Pdf Conversion

[80]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
from google.colab import drive
drive.mount('/content/drive', force_remount = True)
!cp /content/drive/MyDrive/Colab_Notebooks/
 ↪Capstone_Project01_EDA_Siddhartha_Pasayat/
 ↪Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01.ipynb  ./
!jupyter nbconvert --to PDF␣
 ↪'Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01.ipynb'
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
```

```
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 20 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.9 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
```

```
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.16 [5,093 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.16 [2,265 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.12 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.12 [3,073 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
```

```
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive all
2017.20180305-1 [14.4 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
all 2017.20180305-1 [4,026 kB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 146 MB in 4s (34.8 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 155676 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb …
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2_all.deb …
Unpacking fonts-lato (2.0-2) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.8-2_all.deb …
Unpacking poppler-data (0.4.8-2) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.09_all.deb …
Unpacking tex-common (6.09) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/04-fonts-lmodern_2.004.5-3_all.deb …
Unpacking fonts-lmodern (2.004.5-3) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/05-fonts-noto-mono_20171026-2_all.deb …
Unpacking fonts-noto-mono (20171026-2) …
Selecting previously unselected package fonts-texgyre.
```

```
Preparing to unpack …/06-fonts-texgyre_20160520-1_all.deb …
Unpacking fonts-texgyre (20160520-1) …
Selecting previously unselected package javascript-common.
Preparing to unpack …/07-javascript-common_11_all.deb …
Unpacking javascript-common (11) …
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack …/08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb …
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack …/09-libcupsimage2_2.2.7-1ubuntu2.9_amd64.deb …
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/10-libijs-0.35_0.35-13_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-13) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/11-libjbig2dec0_0.13-6_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.13-6) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.16_all.deb
…
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.16) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.16_amd64.deb …
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.16) …
Selecting previously unselected package libjs-jquery.
Preparing to unpack …/14-libjs-jquery_3.2.1-1_all.deb …
Unpacking libjs-jquery (3.2.1-1) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libpotrace0.
Preparing to unpack …/16-libpotrace0_1.14-2_amd64.deb …
Unpacking libpotrace0 (1.14-2) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/18-rubygems-integration_1.11_all.deb …
Unpacking rubygems-integration (1.11) …
Selecting previously unselected package ruby2.5.
Preparing to unpack …/19-ruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking ruby2.5 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package ruby.
Preparing to unpack …/20-ruby_1%3a2.5.1_amd64.deb …
Unpacking ruby (1:2.5.1) …
Selecting previously unselected package rake.
```

```
Preparing to unpack …/21-rake_12.3.1-1ubuntu0.1_all.deb …
Unpacking rake (12.3.1-1ubuntu0.1) …
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack …/22-ruby-did-you-mean_1.2.0-2_all.deb …
Unpacking ruby-did-you-mean (1.2.0-2) …
Selecting previously unselected package ruby-minitest.
Preparing to unpack …/23-ruby-minitest_5.10.3-1_all.deb …
Unpacking ruby-minitest (5.10.3-1) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/24-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-power-assert.
Preparing to unpack …/25-ruby-power-assert_0.3.0-1_all.deb …
Unpacking ruby-power-assert (0.3.0-1) …
Selecting previously unselected package ruby-test-unit.
Preparing to unpack …/26-ruby-test-unit_3.2.5-1_all.deb …
Unpacking ruby-test-unit (3.2.5-1) …
Selecting previously unselected package libruby2.5:amd64.
Preparing to unpack …/27-libruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack …/28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack …/29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Selecting previously unselected package lmodern.
Preparing to unpack …/32-lmodern_2.004.5-3_all.deb …
Unpacking lmodern (2.004.5-3) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/33-preview-latex-style_11.91-1ubuntu1_all.deb …
Unpacking preview-latex-style (11.91-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/34-t1utils_1.41-2_amd64.deb …
Unpacking t1utils (1.41-2) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/35-tex-gyre_20160520-1_all.deb …
Unpacking tex-gyre (20160520-1) …
Selecting previously unselected package texlive-binaries.
```

```
Preparing to unpack …/36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/37-texlive-base_2017.20180305-1_all.deb …
Unpacking texlive-base (2017.20180305-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/38-texlive-fonts-recommended_2017.20180305-1_all.deb …
Unpacking texlive-fonts-recommended (2017.20180305-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/39-texlive-latex-base_2017.20180305-1_all.deb …
Unpacking texlive-latex-base (2017.20180305-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/40-texlive-latex-recommended_2017.20180305-1_all.deb …
Unpacking texlive-latex-recommended (2017.20180305-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/41-texlive_2017.20180305-1_all.deb …
Unpacking texlive (2017.20180305-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/42-texlive-pictures_2017.20180305-1_all.deb …
Unpacking texlive-pictures (2017.20180305-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/43-texlive-latex-extra_2017.20180305-2_all.deb …
Unpacking texlive-latex-extra (2017.20180305-2) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/44-texlive-plain-generic_2017.20180305-2_all.deb …
Unpacking texlive-plain-generic (2017.20180305-2) …
Selecting previously unselected package tipa.
Preparing to unpack …/45-tipa_2%3a1.3-20_all.deb …
Unpacking tipa (2:1.3-20) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/46-texlive-xetex_2017.20180305-1_all.deb …
Unpacking texlive-xetex (2017.20180305-1) …
Setting up libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.16) …
Setting up libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libjs-jquery (3.2.1-1) …
Setting up libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1) …
Setting up libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up tex-common (6.09) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up poppler-data (0.4.8-2) …
Setting up tex-gyre (20160520-1) …
Setting up preview-latex-style (11.91-1ubuntu1) …
Setting up fonts-texgyre (20160520-1) …
Setting up fonts-noto-mono (20171026-2) …
Setting up fonts-lato (2.0-2) …
```

```
Setting up libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Setting up libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Setting up libjbig2dec0:amd64 (0.13-6) …
Setting up ruby-did-you-mean (1.2.0-2) …
Setting up t1utils (1.41-2) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up libijs-0.35:amd64 (0.35-13) …
Setting up rubygems-integration (1.11) …
Setting up libpotrace0 (1.14-2) …
Setting up javascript-common (11) …
Setting up ruby-minitest (5.10.3-1) …
Setting up libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Setting up libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.16) …
Setting up libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-lmodern (2.004.5-3) …
Setting up ruby-power-assert (0.3.0-1) …
Setting up texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up texlive-base (2017.20180305-1) …
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4:
/var/lib/texmf/tex/generic/config/pdftexconfig.tex
Setting up texlive-fonts-recommended (2017.20180305-1) …
Setting up texlive-plain-generic (2017.20180305-2) …
Setting up texlive-latex-base (2017.20180305-1) …
Setting up lmodern (2.004.5-3) …
Setting up texlive-latex-recommended (2017.20180305-1) …
Setting up texlive-pictures (2017.20180305-1) …
Setting up tipa (2:1.3-20) …
Regenerating '/var/lib/texmf/fmtutil.cnf-DEBIAN'… done.
Regenerating '/var/lib/texmf/fmtutil.cnf-TEXLIVEDIST'… done.
update-fmtutil has updated the following file(s):
        /var/lib/texmf/fmtutil.cnf-DEBIAN
        /var/lib/texmf/fmtutil.cnf-TEXLIVEDIST
If you want to activate the changes in the above file(s),
you should run fmtutil-sys or fmtutil.
Setting up texlive (2017.20180305-1) …
```

```
Setting up texlive-latex-extra (2017.20180305-2) …
Setting up texlive-xetex (2017.20180305-1) …
Setting up ruby2.5 (2.5.1-1ubuntu1.12) …
Setting up ruby (1:2.5.1) …
Setting up ruby-test-unit (3.2.5-1) …
Setting up rake (12.3.1-1ubuntu0.1) …
Setting up libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Processing triggers for mime-support (3.60ubuntu1) …
Processing triggers for libc-bin (2.27-3ubuntu1.5) …
Processing triggers for man-db (2.8.3-2ubuntu0.1) …
Processing triggers for fontconfig (2.12.6-0ubuntu2) …
Processing triggers for tex-common (6.09) …
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting pypandoc
  Downloading pypandoc-1.8.1-py3-none-any.whl (20 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.8.1
Mounted at /content/drive
[NbConvertApp] Converting notebook
Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01.ipynb to PDF
[NbConvertApp] Support files will be in
Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files/
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
```

```
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Making directory
./Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01_files
[NbConvertApp] Writing 141877 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 943392 bytes to
Team_Notebook_Airbnb_Bookings_Analysis_Capstone_Project_01.pdf
```