AirBnb Bookings Analysis

Project Type - Exploratory Data Analysis

Project Summary -

This project envolves conducting an analysis of a dataset consisting of around 49,000 observations. The primary objective is to extract meaningful insights that will equip management and stakeholders with crucial information for making informed decisions on how to expand or enhance the business, ultimately driving growth.

Through this analysis, we aim to identify patterns within the data, which will guide recommendations for strategic improvements. Not only will this analysis help guests make better choices, but it will also assist hosts in implementing necessary changes to foster business growth.

The information we have includes:

- Listing counts
- Distribution of data by specific neighborhood groups
- Prices
- Review data
- Room type preferences

From this data, we can uncover insights such as:

- Guest preferences for hosts
- Preferred room types
- Desired price ranges
- Most favored neighborhoods

GitHub Link -

Provide your GitHub Link here.

Problem Statement

The task of this project is to derive insights from the given dataset so that it can be used by the stake holders for business improvements

1. Knowing the Data

Import Libraries

```
# Import Libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import math
import seaborn as sns
import openpyxl
pd.set_option('display.max_columns', 200)
```

Loading Dataset

```
# Load Dataset
'''Creating a dataframe of the given dataset'''
airbnb_df = pd.DataFrame(pd.read_csv("Airbnb NYC 2019_1.csv"))
```

Dataset First View

```
# Dataset First Look
airbnb_df.head(5)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nig
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	

Next steps: View recommended plots New interactive sheet

As we can see here is the first look of our dataset on which we will be working, let's dive deep into it.

```
# Let's check for all the columns we have in our dataset.
airbnb_df.columns
```

```
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')
```

Here is the list of all our columns in the dataset.

Dataset Rows & Columns count

```
# Dataset Rows & Columns count
airbnb_df.shape
```

```
(48895, 16)
```

Looking at the shape of our dataset we can see that the number of rows is significantly high as compared to the number of columns. We have:- rows = 48895 columns = 16

Dataset Information

```
# Dataset Info
airbnb_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
```

Here we can see the division of Categorical and Numerical values in our dataset, We can see:-

- 3 columns have float64 data values (Numerical)
- 7 columns have int64 data type values (Numerical)
- 6 columns have object data type values (Categorical)

Duplicate Values

```
# Dataset Duplicate Value Count
duplicated_values = airbnb_df[airbnb_df.duplicated()]
duplicated_values
```

```
id  name  host_id  host_name  neighbourhood_group  neighbourhood  latitude  longitude  room_type  price  minimum_nights  number_of_
```

With this we can see that there are no exact duplicate values, however we need to check for other columns as well, like id and name, as these values are not likely to be same.

```
# Let us first check for 'id' column.
duplicated_id = airbnb_df[airbnb_df.duplicated(subset=['id'])]
duplicated_id
```

```
id  name  host_id  host_name  neighbourhood_group  neighbourhood  latitude  longitude  room_type  price  minimum_nights  number_of_
```

As we can see there is no dulicated 'id's in our dataset.

```
# Now let us check for 'name' column.
duplicated_name = airbnb_df[airbnb_df.duplicated(subset=['name'])]
duplicated_name
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum
330	81739	Loft w/ Terrace @ Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73842	-73.95312	Private room	249	
339	84010	Superior @ Box House	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73813	-73.95394	Private room	179	
580	219818	✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿	1138692	Keera (Jena)	Manhattan	Lower East Side	40.71892	-73.98401	Entire home/apt	199	
661	250537	The Lenox in Harlem	1313306	Yvette	Manhattan	Harlem	40.81122	-73.94279	Entire home/apt	400	
669	253471	Loft Suite @ The Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73641	-73.95330	Entire home/apt	199	
...	...	...	...	...	...	...	...	...	...	...	
48684	36382847	Comfort home	266211707	Yan	Brooklyn	Sunset Park	40.64439	-74.01816	Private room	185	
48735	36412461	Sunny, Cozy, Private Room In The Heart of Bush...	147515897	Flávia	Brooklyn	Bushwick	40.70366	-73.92728	Private room	84	
48759	36420404	Home Sweet Home	273656890	Liana	Manhattan	East Harlem	40.79266	-73.94740	Private room	50	
48791	36427922	Home away from home	238163900	Lucy	Queens	Cambria Heights	40.68557	-73.72731	Private room	50	
48826	36449743	Brooklyn's finest	66084717	Tim	Brooklyn	East Flatbush	40.65170	-73.92580	Entire home/apt	200	

998 rows × 16 columns

Now as we can see we have 998 rows that are duplicates, this might be due to some changes to get an idea for the reason of these duplicated values, let us observe any one of the rows.

```
# We can check for any specific row having the name value included in the duplicated data, using the query command.
# Let's check for where name is 'Superior @ Box House', which is in our duplicated data.
airbnb_df.query('name == "Superior @ Box House"')
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights
321	77765	Superior @ Box House	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73749	-73.95292	Private room	179	3
339	84010	Superior @ Box House	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73813	-73.95394	Private room	179	3
682	253846	Superior @ Box House	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73731	-73.95450	Private room	179	3

Here we can observe that almost all the values are same except for the id, latitude, longitude, number_of_reviews, reviews_per_month, availability_365. However these columns can be different, and they are not so important.

```
# To tackle this issue we will create a new data frame in which we will include
# only those columns which are really important for us.
airbnb_df = airbnb_df[['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']].copy()
```

Now that we have created this dataframe we will be able to handle duplicated values more efficiently.

```
# Let's check for the duplicated values of any other row.
airbnb_df.query('name == "Loft w/ Terrace @ Box House Hotel"')
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights
328	80700	Loft w/ Terrace @ Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73738	-73.95482	Private room	349	3
330	81739	Loft w/ Terrace @ Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73842	-73.95312	Private room	249	3
680	253839	Loft w/ Terrace @ Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73783	-73.95259	Private room	249	3

As we can see here each and every value is almost the same, except for the prices, there might be a chances that the prices were altered as per the market condition due however the hotel is the same.

```
# In this case we will seek out the duplicated values for those columns which
# are concerning and may effect the data if duplicated.

# Checking for the values where - 'name', 'host_name','neighbourhood_group', 'neighbourhood', 'room_type' are duplicated.
airbnb_df.loc[airbnb_df.duplicated(subset= ['name', 'host_name', 'neighbourhood_group', 'neighbourhood', 'room_type'])]
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minim
330	81739	Loft w/ Terrace @ Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73842	-73.95312	Private room	249	
339	84010	Superior @ Box House	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73813	-73.95394	Private room	179	
580	219818	✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿	1138692	Keera (Jena)	Manhattan	Lower East Side	40.71892	-73.98401	Entire home/apt	199	
669	253471	Loft Suite @ The Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73641	-73.95330	Entire home/apt	199	
670	253475	Loft Suite @ The Box House Hotel	417504	The Box House Hotel	Brooklyn	Greenpoint	40.73794	-73.95254	Entire home/apt	199	
...	...	...	...	...	...	...	...	...	...	...	
47876	35966653	Bright, contemporary and best location	24232061	Tracy	Manhattan	Upper East Side	40.77297	-73.95530	Private room	122	
48026	36039574	★Premier Queen Room with Balcony ★	270874051	Hotel Vetiver	Queens	Long Island City	40.75300	-73.93485	Private room	99	
48207	36139806	30 mins to Times Square!! 15 mins LGA, 25mins ...	260209224	Lotay	Queens	Jackson Heights	40.75077	-73.87020	Entire home/apt	67	
48662	36372006	Very Clean Private Room Near Buses & Restauran...	118405437	PengYu	Queens	Woodhaven	40.69411	-73.86877	Private room	66	
48684	36382847	Comfort home	266211707	Yan	Brooklyn	Sunset Park	40.64439	-74.01816	Private room	185	

240 rows × 16 columns

```
# Let's check for any specific value.
airbnb_df.query('name == "✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿"')
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_night
579	219793	✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿	1138692	Keera (Jena)	Manhattan	Lower East Side	40.71813	-73.98416	Entire home/apt	199	
580	219818	✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿	1138692	Keera (Jena)	Manhattan	Lower East Side	40.71892	-73.98401	Entire home/apt	199	

In this now we can see that all the values are same except for last_review and reviews_per_month, which are not so important regarding duplicacy.

Now to drop the duplicated values we will sort out dataframe on the basis of latest entries, for which we need to have a timestamp in our dataset, in our dataset we can use the 'last_review' column as the timestamp for our dataset.

```
# Converting the 'last_review' column in a datetime format.
airbnb_df['last_review'] = pd.to_datetime(airbnb_df['last_review'])
```

```
# Now let us sort the data by the last_review column
airbnb_df = airbnb_df.sort_values(by='last_review', ascending=False).reset_index(drop=True)
```

As we can see there are few NA values in our dataframe in the date column let's fill these values with the latest dates we are having in our dataframe.

```
# Replacing NA Values
airbnb_df['last_review'].replace(np.nan,airbnb_df['last_review'].max(), inplace=True)

# Let's sort the values again
airbnb_df = airbnb_df.sort_values(by='last_review', ascending=False).reset_index(drop=True)
```

```
<ipython-input-17-3e053ca8e633>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  airbnb_df['last_review'].replace(np.nan,airbnb_df['last_review'].max(), inplace=True)
```

```
# Dropping duplicated values
airbnb_df = airbnb_df.drop_duplicates(subset=['name', 'host_name', 'neighbourhood_group', 'neighbourhood', 'room_type'], keep='first').re

# Now the duplicated columns have been dropped, let's check the current shape of our data frame.
airbnb_df.shape # (48655, 14)

# Let's check if there are any duplicated values now
airbnb_df[airbnb_df.duplicated(subset=['name', 'host_name', 'neighbourhood_group', 'neighbourhood', 'room_type'])] # No values

airbnb_df.query('name == "✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿"') # The latest value shows up
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nigl
48029	219818	✿✿✿ COUNTRY COTTAGE IN THE CITY✿✿✿	1138692	Keera (Jena)	Manhattan	Lower East Side	40.71892	-73.98401	Entire home/apt	199	

We have successfully dropped the duplicated values and now we only have latest data values.

Missing Values/Null Values

```
# Missing Values/Null Values Count
null_values = airbnb_df.isnull()
null_value_count = null_values.sum()

# Visualizing the missing values
null_value_count
```

	0
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	0
reviews_per_month	9989
calculated_host_listings_count	0
availability_365	0

dtype: int64

As we can see we were having missing values in name and host_name columns, however as we are having their respective ids we will still be able to manage the data, however let us check for outliers.

```
# To check the outliers we need to check the columns which are having numerical data.
airbnb_df.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	last_review	rev
count	4.865500e+04	4.865500e+04	48655.000000	48655.000000	48655.000000	48655.000000	48655.000000	48655	
mean	1.900642e+07	6.746743e+07	40.728925	-73.952195	152.671606	7.002939	23.326626	2018-11-30 00:30:20.162367744	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	2011-03-28 00:00:00	
25%	9.464228e+06	7.800355e+06	40.689990	-73.983070	69.000000	1.000000	1.000000	2018-11-04 00:00:00	
50%	1.966033e+07	3.070277e+07	40.722990	-73.955660	105.000000	3.000000	5.000000	2019-06-14 00:00:00	
75%	2.913448e+07	1.074344e+08	40.763130	-73.936265	175.000000	5.000000	24.000000	2019-07-04 00:00:00	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	2019-07-08 00:00:00	
std	1.098021e+07	7.851896e+07	0.054571	0.046102	240.433184	20.536747	44.621609	NaN	

If we look into this data we can see that the price and the minimum_nights columns are the most concerning ones, in these 2 columns we need to find the outliers and drop them.

```
# As we see above the column_name 'calculated_host_listings_count' is quiet long
# Let's change it to 'listings'
airbnb_df.rename(columns={
    "calculated_host_listings_count": 'listings'
}, inplace=True)

# Lets check the price columns once
airbnb_df['price'].describe()

# As we can see the min price is 0 which is not likely to happen.
# According to the current website the price range starts from $25.
# In this case we will check for the prices which are below $25 and we will
# replace their values with 25 so that we can handle the outliers in price column.

airbnb_df['price'].replace(range(0, 25), 25, inplace=True) # replacing values
airbnb_df['price'].describe()
```

```
<ipython-input-21-5e31f3d75c36>:15: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  airbnb_df['price'].replace(range(0, 25), 25, inplace=True) # replacing values
```

	price
count	48655.000000
mean	152.693228
std	240.421079
min	25.000000
25%	69.000000
50%	105.000000
75%	175.000000
max	10000.000000

dtype: float64

As the few price values were less than $25, which is not likely to happen, we have converted those values into 25, so that we may handle the outliers more efficiently, as there was other crucial info present in those rows.

```
# Now let's check the minimum_nights column.
airbnb_df['minimum_nights'].describe()

# As the minimum value in this column is 1, we can work on only the upper limit.
# In this case if any value is more than 365 days, we will mark it as 365.

airbnb_df['minimum_nights'].replace(range(366, 1251), 365, inplace=True) # replacing values
airbnb_df['minimum_nights'].describe()
```

```
<ipython-input-22-6d516405d7f4>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  airbnb_df['minimum_nights'].replace(range(366, 1251), 365, inplace=True) # replacing values
```

	minimum_nights
count	48655.000000
mean	6.915528
std	17.544873
min	1.000000
25%	1.000000
50%	3.000000
75%	5.000000
max	365.000000

dtype: float64

Now we have modified the outliers, we can move ahead with our dataset ready to be wrangled for deriving insights.

2. Understanding Your Variables

```
# Looking on our data once
airbnb_df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	m
0	36455809	Cozy Private Room in Bushwick, Brooklyn	74162901	Christine	Brooklyn	Bushwick	40.69805	-73.92801	Private room	30	
1	15968426	Comfy spacious Astoria aptmt 15m from Manhattan	2753243	Jwanah	Queens	Astoria	40.76248	-73.92422	Private room	56	
2	16000062	Charming Carriage House near Prospect Park	4366974	Ariana	Brooklyn	Crown Heights	40.67320	-73.96195	Entire home/apt	300	
3	15998231	Cozy Room in Brownstone	20327528	Miller	Brooklyn	Bedford-Stuyvesant	40.68631	-73.95597	Private room	35	
4	15987034	near Williamsburg/Manhattan/ Greenpoint - 1 b/1b	19803201	Gino	Brooklyn	Greenpoint	40.72388	-73.94040	Entire home/apt	150	

Next steps: View recommended plots New interactive sheet

```
# Dataset Columns
df_columns = airbnb_df.columns
df_columns # All the columns of our cleaned data.
```

```
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', 'listings', 'availability_365'],
      dtype='object')
```

There are basically 3 types of variables according to their roles:-

- Numerical Variables: These variables represent quantitative data and can be further categorized into:-
    - Continuous Variables: These variables can take any value within the given number of range.
    - Discrete Variables: These variables are having a specific value and are related to a specific identity.
- Categorical Variables: These variables represent qualitative data and can be further categorized into:-
    - Nominal Variables: These variables are random and they do not follow any order or ranking.
    - Ordinal Variables: These variables are according to an order they can be ranked as well.
- Time Variables: These variables are basically date and time variables having a timestamp.

```
# Dataset Describe
df_describe = airbnb_df.describe()
df_describe # These are all the numerical variables in our dataset.
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	last_review	rev
count	4.865500e+04	4.865500e+04	48655.000000	48655.000000	48655.000000	48655.000000	48655.000000	48655	
mean	1.900642e+07	6.746743e+07	40.728925	-73.952195	152.693228	6.915528	23.326626	2018-11-30 00:30:20.162367744	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	25.000000	1.000000	0.000000	2011-03-28 00:00:00	
25%	9.464228e+06	7.800355e+06	40.689990	-73.983070	69.000000	1.000000	1.000000	2018-11-04 00:00:00	
50%	1.966033e+07	3.070277e+07	40.722990	-73.955660	105.000000	3.000000	5.000000	2019-06-14 00:00:00	
75%	2.913448e+07	1.074344e+08	40.763130	-73.936265	175.000000	5.000000	24.000000	2019-07-04 00:00:00	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	365.000000	629.000000	2019-07-08 00:00:00	
std	1.098021e+07	7.851896e+07	0.054571	0.046102	240.421079	17.544873	44.621609	NaN	

Next steps: View recommended plots New interactive sheet

Variables Description

As we can see that we are having various outputs that can be studied so as to attain certain measures and draw conclusions on the basis of the data. For example: we have a wide range of prices available, however the average price range prefered by the customers are around 150, which gives us an idea on the budget of the guests and their preference.

Check Unique Values for each variable.

```
# Check Unique Values for each variable.
def unique_value(df):
  for column in df.columns:
    unique_values = df[column].unique()
    print(f"Unique values for '{column}': {unique_values}")
```

3. ***Data Wrangling***

Data Wrangling Code

As our Data cleaning, Data transformation, and Handling outliers has been completed, now we will be working on "Feature Engineering".

```
# Let's create a new column with the price range distribution.
# We will use it while working with the price column.
start = 0
end = 10000
breakpoints = np.linspace(start, end, num=101)
breakpoints = breakpoints.astype(int)
def price_range(amt):
    bp = breakpoints
    for i in range(len(breakpoints)-1):
        if bp[i] <= amt <= bp[i+1]:
            return f"{bp[i]} - {bp[i+1]}"


airbnb_df['price_range'] = airbnb_df['price'].apply(lambda amt: price_range(amt))
airbnb_df['price_range']
```

```
       price_range
    0      0 - 100
    1      0 - 100
    2    200 - 300
    3      0 - 100
    4    100 - 200
  ...          ...
48650      0 - 100
48651    100 - 200
48652      0 - 100
48653    200 - 300
48654      0 - 100
48655 rows × 1 columns

dtype: object
```

Creating this price range column can provide us these benefits:-

- Data Summarization: As this will provide an effective summary of the price distribution in our dataset.
- Visualization: on a price range instead of individual price column is much more effective.
- Segmentation and Analysis: As it will divide the prices into different segments it would be really easy to compare the prices into different segments or ranges.
- Decision Making: It can help us in various decisions making, into different scenarios, like we can use this info to provide recommendations to the customers according to their budget and rerquirements.
- Communicating Insights: Price range can easily communicate the budget and preferences of our customers and it can also tell us where does the majority of our cutomers lies, according to their purchasing power.

```
# Let's take a look at our dataset once.
airbnb_df.head()
```

```
# As we can see our dataset is sorted according to the last review, let us sort our dataset according to the price.
airbnb_df.sort_values(by='price', ascending=False, inplace=True)

# Let's take a look at our price sorted dataset once.
airbnb_df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_r
42899	13894339	Luxury 1 bedroom apt. - stunning Manhattan views	5143901	Erin	Brooklyn	Greenpoint	40.73260	-73.95739	Entire home/apt	10000	
46628	7003697	Furnished room in Astoria apartment	20582832	Kathrine	Queens	Astoria	40.76810	-73.91651	Private room	10000	
1809	22436899	1-BR Lincoln Center	72390391	Jelena	Manhattan	Upper West Side	40.77213	-73.98665	Entire home/apt	10000	
48357	4737930	Spanish Harlem Apt	1235070	Olson	Manhattan	East Harlem	40.79264	-73.93898	Entire home/apt	9999	
46967	9528920	Quiet, Clean, Lit @ LES & Chinatown	3906464	Amy	Manhattan	Lower East Side	40.71355	-73.98507	Private room	9999	

Next steps: View recommended plots New interactive sheet

```
# Let us check our columns and filter those columns that will work as our features.
airbnb_df.columns
```

```
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', 'listings', 'availability_365', 'price_range'],
      dtype='object')
```

We can consider these columns to be our features so that we can work with these:-

- id
- host_id
- neighbourhood_group
- neighbourhood
- room_type
- price
- price_range
- minimum_nights
- number_of_reviews
- reviews_per_month
- listings
- availability_365

```
# Let filter the dataframe with only the required columns.
feature_df = airbnb_df[['id', 'name', 'host_id', 'neighbourhood_group',
       'neighbourhood', 'room_type', 'price', 'minimum_nights',
       'number_of_reviews', 'reviews_per_month', 'listings',
       'availability_365', 'price_range']]

feature_df = feature_df.reset_index(drop=True)
feature_df.head()
```

	id	name	host_id	neighbourhood_group	neighbourhood	room_type	price	minimum_nights	number_of_reviews	reviews_p
0	13894339	Luxury 1 bedroom apt. - stunning Manhattan views	5143901	Brooklyn	Greenpoint	Entire home/apt	10000	5	5	
1	7003697	Furnished room in Astoria apartment	20582832	Queens	Astoria	Private room	10000	100	2	
2	22436899	1-BR Lincoln Center	72390391	Manhattan	Upper West Side	Entire home/apt	10000	30	0	
3	4737930	Spanish Harlem Apt	1235070	Manhattan	East Harlem	Entire home/apt	9999	5	1	
4	9528920	Quiet, Clean, Lit @ LES & Chinatown	3906464	Manhattan	Lower East Side	Private room	9999	99	6	

Next steps: View recommended plots New interactive sheet

```
feature_df.head()
```

	id	name	host_id	neighbourhood_group	neighbourhood	room_type	price	minimum_nights	number_of_reviews	reviews_p
0	13894339	Luxury 1 bedroom apt. - stunning Manhattan views	5143901	Brooklyn	Greenpoint	Entire home/apt	10000	5	5	
1	7003697	Furnished room in Astoria apartment	20582832	Queens	Astoria	Private room	10000	100	2	
2	22436899	1-BR Lincoln Center	72390391	Manhattan	Upper West Side	Entire home/apt	10000	30	0	
3	4737930	Spanish Harlem Apt	1235070	Manhattan	East Harlem	Entire home/apt	9999	5	1	
4	9528920	Quiet, Clean, Lit @ LES & Chinatown	3906464	Manhattan	Lower East Side	Private room	9999	99	6	

Next steps: View recommended plots New interactive sheet

As we can see we are having few groups in our datasets, let us check what all insights we can derive from them, by seeking information according to the groups.

```
# Let's check the top performing Host as per the total listing count
host_groups = feature_df.groupby('name')
hosts = []
no_of_listings = []
host_prices = []
for host, data in host_groups:
  hosts.append(host)
  no_of_listings.append(data['listings'].sum())
  host_prices.append(data['price'].mean())

host_df = pd.DataFrame({
    'Host Name': hosts,
    'Total Listings': no_of_listings,
    'Price': host_prices
})
host_df = host_df.sort_values(by='Total Listings', ascending=False).reset_index(drop=True)
host_df = host_df.drop_duplicates(subset='Total Listings').reset_index(drop=True)
top_10_hosts = host_df.head(10)
```

```
host_df['Revenue'] = (host_df['Total Listings'])*(host_df['Price'])
top_host_revenue = host_df.sort_values(by='Revenue', ascending=False).reset_index(drop=True)
top_host_revenue.head(10)
```

	Host Name	Total Listings	Price	Revenue
0	Pleasant 1BR in Midtown East by Sonder	654	197.000000	128838.0
1	Central Herald Sq.1BR w/ Roofdeck, Gym next to...	232	334.000000	77488.0
2	Sonder | Stock Exchange | Classic Studio + Lau...	327	201.000000	65727.0
3	West 55th street, Lux 1bd Serviced Apartment	261	251.666667	65685.0
4	West 15th Street Cozy Chelsea 1bd Serviced Apt	174	200.000000	34800.0
5	NYC modern 1 bedroom apt. near Carnegie Hall!	121	239.000000	28919.0
6	TRUE2BR-PRIME MIDTOWN EAST~53rd&3rd	114	232.500000	26505.0
7	LUXURY DELUXE 1BR IN HELL'S KITCHEN-DOORMAN/GYM	65	350.000000	22750.0
8	32 FLR VIEWS!LINCOLN SQR-LUXURY MIDTOWN WEST 60TH	98	229.500000	22491.0
9	East 40th Street, Lux Serviced Studio Apt	87	189.000000	16443.0

Next steps: View recommended plots New interactive sheet

```
# First let's check how many neighbourhood_groups are there.
feature_df['neighbourhood_group'].unique() # There are 5 different groups.

# ['Brooklyn', 'Queens', 'Manhattan', 'Staten Island', 'Bronx']
#  let's group our dataset according to the neighbourhood groups.

n_groups = feature_df.groupby('neighbourhood_group')

# Let's check which group is most prefered as per the listings, and no. of reviews.
groups = [] # To save the groups
listings = []
reviews = []
max_price = []
min_price = []
for group, data in n_groups:
  groups.append(group)
  listings.append(data['listings'].sum())
  reviews.append(data['number_of_reviews'].sum())
  max_price.append(data['price'].max())
  min_price.append(data['price'].min())

group_feat_df = pd.DataFrame({
    'Group': groups,
    'Listing Count': listings,
    'No._of_reviews': reviews,
    'Min Price': min_price,
    'Max Price': max_price
})

group_feat_df
# Here we can see that Manhattan group is the most prefered group as per the listings.
# However the most reviews are given to the Brooklyn group.

# Now let's divide our dataset as per the room types, and create their groups.
# First let's check how room_types are there.

feature_df['room_type'].unique() # There are 3 room types.

# ['Entire home/apt', 'Private room', 'Shared room']
#  let's group our dataset according to the room types.

room_groups = feature_df.groupby('room_type')
rooms = []
room_listings = []
room_reviews = []
max_room_price = []
min_room_price = []
for room_type, room_data in room_groups:
    rooms.append(room_type)
    room_listings.append(room_data['listings'].sum())
    max_room_price.append(room_data['price'].max())
    min_room_price.append(room_data['price'].min())

room_feat_df = pd.DataFrame({
    'Group': rooms,
    'Listing Count': room_listings,
    'Min Price': min_room_price,
    'Max Price': max_room_price
})

room_feat_df
# Here we can see the most prefered room type is the Entire home/apt.
# The most reviews are also given to the Entire home/apt.
# Shared rooms are the least prefered.
```

	Group	Listing Count	Min Price	Max Price
0	Entire home/apt	263986	25	10000
1	Private room	70761	25	10000
2	Shared room	5323	25	1800

Next steps: View recommended plots New interactive sheet

```
# Now let us check how many different neighbourhoods are there in total.
feature_df['neighbourhood'].count() # There are 48655 neighbourhoods

# Now let us check what are the top 10 most prefered neighbourhoods.
# Also, let's check their average pricing and average price range.
area_groups = feature_df.groupby('neighbourhood')
areas = []
listing_count = []
avg_price = []
for area, n_data in area_groups:
  areas.append(area)
  listing_count.append(n_data['listings'].sum())
  avg_price.append(round(n_data['price'].mean(), 2))

area_feat_df = pd.DataFrame({
    'Area': areas,
    'Listing Count': listing_count,
    'Average Price': avg_price,
})

area_feat_df = area_feat_df.sort_values(by='Listing Count', ascending=False).reset_index(drop=True)
area_feat_df.head(10)
```

	Area	Listing Count	Average Price
0	Financial District	84942	226.03
1	Hell's Kitchen	24754	205.33
2	Murray Hill	24726	221.18
3	Midtown	24647	282.66
4	Chelsea	17483	248.36
5	Theater District	16151	242.67
6	Upper East Side	14909	189.10
7	Upper West Side	13201	211.25
8	Bedford-Stuyvesant	9605	107.77
9	Tribeca	7519	492.23

Next steps: View recommended plots New interactive sheet

In the above manipulations we have created and worked with few groups that were there in our dataset, based upon which we were able to derive few insights.

```
# Let us now work with relationships.
# Relationship: Price vs. Room Type
# Checking the distribution of prices for different room types.
# "room_groups" is our grouped dataframe we will be using this.

avg_room_price = []
for room, data in room_groups:
  avg_room_price.append(data['price'].mean())

room_vs_price = pd.DataFrame({
    'Room Type': rooms,
    'Avg_Price': avg_room_price
})
```

```
room_vs_price

# As we can see here the Entire home/apt is having the highest pricing.
```

```
      Room Type   Avg_Price
0  Entire home/apt  211.692566
1     Private room   89.640793
2      Shared room   70.536395
```

Next steps: View recommended plots New interactive sheet

```
# Relationship: Reviews per Month vs. Room Type
# Comparing the distribution of reviews per month for different room types.

total_reviews = []
for room, data in room_groups:
  total_reviews.append(data['reviews_per_month'].sum())

room_vs_reviews = pd.DataFrame({
    'Room Type': rooms,
    'Reviews': total_reviews
})

room_vs_reviews
# As we can see here the Entire home/apt is having the highest no. of reviews per month.
```

```
      Room Type   Reviews
0  Entire home/apt  26525.13
1     Private room  25400.23
2      Shared room   1232.65
```

Next steps: View recommended plots New interactive sheet

Following Manipulations were done:

This dataset was having good amount of well distributed information, which was a plus point for our study. As this dataset if of Airbnb, the locations play a very crucial role in these values and the inforamtion was distributed well as per the locations and their respective areas.

We have done number of manipulations to seek information which can be beneficial for the stake holders to make decisions for the business, not only that our hosts will also get good idea about the preferences of their customers, as we promised in our agenda.

We divided the complete manipulation into 2 major parts, as per the features and the relationships, so that we can draw insights accordingly.

These are the manipulations we followed:-

- Created a new price range column: As this columns will help us to categorize the price distribution and to simplify the judgement for us to get an idea of the budget of our customer, we have tried to keep it as precise as possible.
- Sorted the dataset as per price: As the data wasn't sorted and was not having any significance regaring any value, we sorted it as per the pricing, keep the most expensive ones on the top.
- We filtered the data so that it becomes simple and easy to understand(less chaotic) for us to work only with those columns that are required for the data analysis and feature engineering.
- Divided the data accoring to the categorical groups to derive insights accordingly, it helped us to identify the outcomes in a structed manner with respect to the considered groups, it provided us with more specific information about the different groups we created so that we can identify insights for each group individually. These are the diferent groups we created and worked with:-
  - Created Nighbourhood Group: This group helped us to get an idea of which neighbourhood is the most prefered one in all of the neighbourhoods.
  - Created room type groups: This group helped us to identify which room type is the most prefered by our customers.
  - Created Neighbourhood area groups: This group helped us to identify which is the most prefered area within the neighbourhoods.

We also worked on few of the relationships that helped us in few comparisons that will help our stake holders to make decisions accordingly. These are the relationships we worked with:-

- Relationship: Price vs. Room Type
  - Checked the distribution of prices for different room types.
  - Determined which room type is having the most expensive price distribution.

- Relationship: Reviews per Month vs. Room Type
    - Compared the distribution of reviews per month for different room types.
    - Explored the level of engagement and satisfaction of our customers as per different room types.
- Relationship: Price vs. Neighbourhood
    - Compared the distribution of prices across different neighbourhoods.
    - Identified neighbourhoods with higher or lower average prices and explored price variations as per the areas.

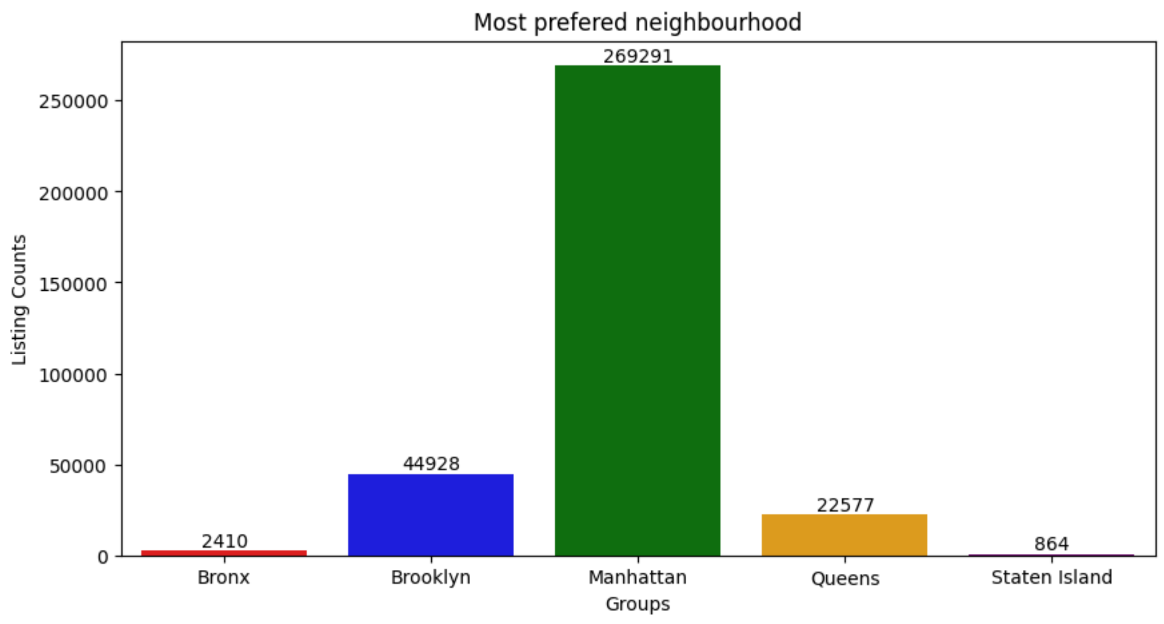
These are the manipulations that we did in order to derive useful information so that it can contribute into the growth of our business and also the busniness of our hosts, and ultimately grow and improve tavelling experience for our customers.

The specifilly derived insights will be mentioned with the visualisations so that it would be better to understand and visualize at the same time.

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart - 1

```
# Chart - 1 visualization code
# Let's visualize the most prefered neighbourhood group with a bar chart

group_feat_df # As we have already created a dataframe for the groups

plt.figure(figsize=(10,5))
# plt.plot(groups,listings, color='black', marker = "o", markerfacecolor = 'blue', markeredgecolor='blue',linestyle='-')
colors = ['red', 'blue', 'green', 'orange', 'purple'] # To make each bar with a different color
sns.barplot(x="Group", y='Listing Count', data=group_feat_df, hue='Group', palette=colors, legend=False)
plt.xlabel('Groups')
plt.ylabel('Listing Counts')
plt.title('Most prefered neighbourhood')
for x, y in zip(range(len(groups)), listings):
    plt.text(x, y, f'{y}', ha='center', va='bottom') # To annotate each bar with the exact value
```



1. Why did you pick the specific chart?

As we need to check which of the groups is having the highest preference, it was quite certain that we need to check the value attained by each group and we also wanted to see the comparison between all the groups, for doing this, bar chart is the best option.

2. What is/are the insight(s) found from the chart?

As we can clearly see that Manhattan is the most prefered group among all and it is outperforming all the other groups with a huge difference, Brooklyn is the 2nd prefernce of our customers after Manhattan.

3. Will the gained insights help creating a positive business impact?

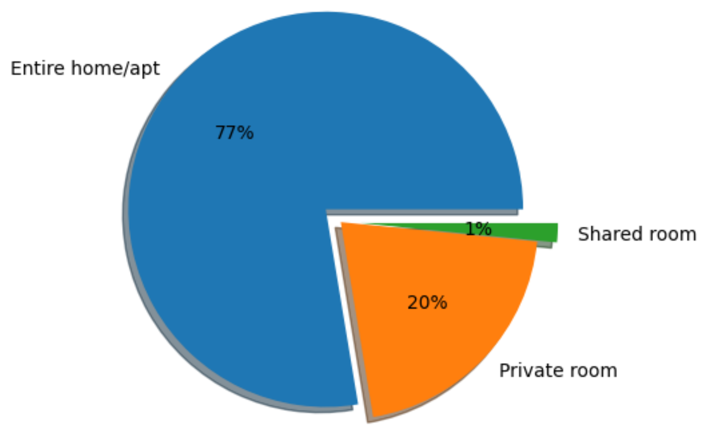
Are there any insights that lead to negative growth? Justify with specific reason.

Absolutely, as we know what are the preferences of our customers we will be able to work on those things that are mostly in demand and we will be able to meet their requirements for their satisfaction, as here in this case we know that the most prefered group is Manhattan, we can target our customers with the availabilites in that area.

If we talk about the negative growth, well it is quite there for our other neighbourhood groups, as there is low demand in the other neighbourhoods, however we will be able to come up with a solution if we try to find out why is that, the most prefered group is Manhattan, if we do that we will be able to identify the cause for low demad in those areas, we need to focus on the reasons and the difference so that we can get to the roots of this.

Chart - 2

```
# Chart - 2 visualization code
# Let's visualize the most prefered room type using a pie chart

room_feat_df # Our grouped df created earlier

# Plotting the data into a pie chart
plt.pie(room_feat_df['Listing Count'], labels=room_feat_df['Group'], autopct="%1i%%", explode=(0.1, 0, 0.1), shadow=True)
plt.show()
```



1. Why did you pick the specific chart?

As we were having only 3 types of rooms it was better to use a pie chart as it would be really easy to see the distribution of the listings on a pie chart and also to see the difference between the room types. In this we are also able to see the difference in percentage wich gave us a wider view on the data outcome.

2. What is/are the insight(s) found from the chart?

As we can clearly see that the "Entire Home/Apt" room type is the most prefered one, this clearly indicates that the people are prioritising their privacy and they would wanna stay all by themselves without having any type interference, as we see that the shared rooms are the least prefered, they are mostly prefered by the students that come from outside so that their accommodation can be affordable. This information can be easily used for more specific target approaches.

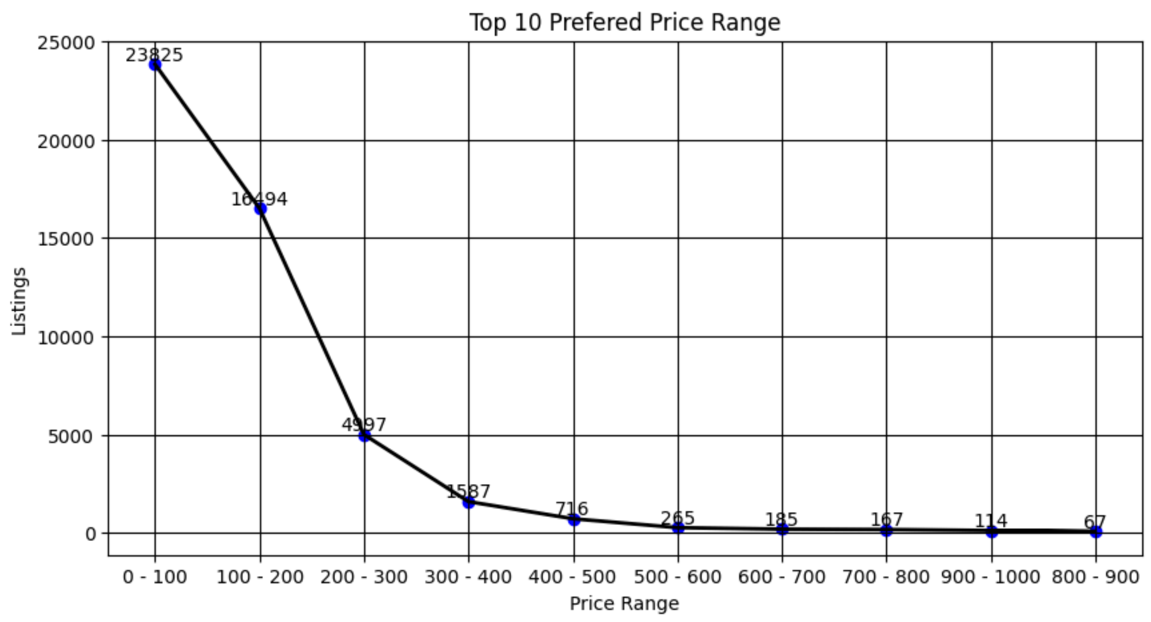
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

As we are able to see what is prefered by the majority of our customer we will be able to spend money on advertisements more efficiently which will help us to minimize our cost wastage, not only for the majority as what sort of customers prefer the other room types, we will be able to target them with their needs which will increase our consumer market even in the low demand sectors, which can work as a decoy for us.

Chart - 3

```
# Chart - 3 visualization code
# Now let's check which price range has the most number of listings, it would give us an idea for the most prefered price range of our c
# Let's plot the top 10 most prefered price range on chart.
prefered_range = airbnb_df['price_range'].value_counts().head(10)
plt.figure(figsize=(10,5))

prefered_range_sorted = prefered_range.sort_index()
plt.scatter(prefered_range.index, prefered_range, color='blue', marker='o')
plt.plot(prefered_range_sorted.index, prefered_range_sorted, color='black', linestyle='-', linewidth=2, label='Line Connecting Dots')

for x, y in zip(prefered_range_sorted.index, prefered_range_sorted): # To annotate each plotted value
    plt.text(x, y, f'{y}', ha='center', va='bottom')
plt.title("Top 10 Prefered Price Range")
plt.xlabel("Price Range")
plt.ylabel("Listings")
plt.grid(color='black')
```



1. Why did you pick the specific chart?

The plot chart can give us the graphical representation of the preferences of our customers regarding prices and how do they go through each of the price ranges. As we can see the chart is giving us a fall in the preferences as the price range increases.

2. What is/are the insight(s) found from the chart?

Looking at this chart we can cleary see the budget preferences of our customers and we can see that the most prefered price range is the 0-100, however there is no 0 values as we have already dropped them. This can give us an idea on the spending power of our customers which will help us in setting better and more specific prices, also at the time of listings we can even make recommendations to our customers with their prefered price ranges.

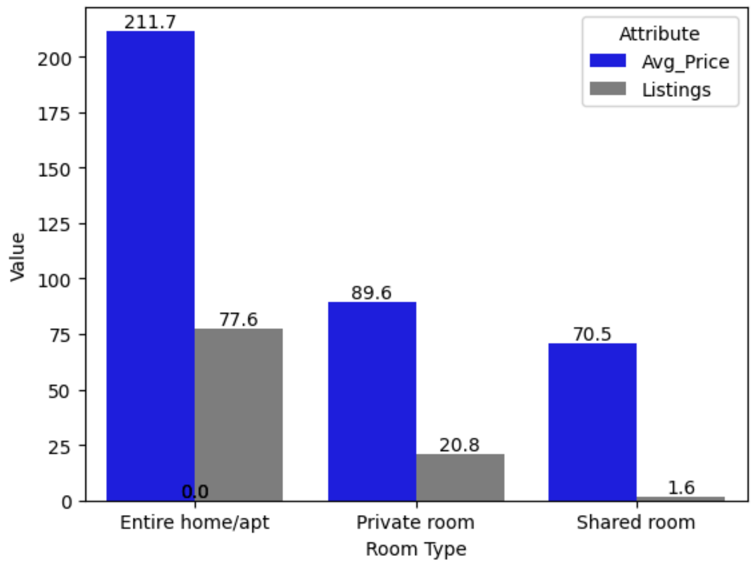
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes definitely, as customers like personalized interfaces and options, if we will focus on providing them the options that are there for them only, it would be really appreciated by them as they will not have to go through a lot while searching for things they are looking for specifically.

Chart - 4

```
# Chart - 4 visualization code
# Let us now visualize the average price division with room types.

data = {
    'Room Type': room_vs_price['Room Type'],
    'Avg_Price': room_vs_price['Avg_Price'],
    'Listings': ((room_feat_df['Listing Count']/room_feat_df['Listing Count'].sum())*100)
}

# Creating a DataFrame
df = pd.DataFrame(data)

df_melted = pd.melt(df, id_vars='Room Type', var_name='Attribute', value_name='Value')
# Creating a strip plot
sns.barplot(x='Room Type', y='Value', data=df_melted, hue="Attribute", palette=['blue', 'grey'], legend=True)

# Applying annotations on the values
for p in plt.gca().patches:
    plt.gca().annotate('{:.1f}'.format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                       textcoords='offset points')

plt.show()
```



1. Why did you pick the specific chart?

As we are only having 3 types it was better to use the bar chart for better visualizing the divisions of the average price and the percentage of the listings divided among these values altogether.

2. What is/are the insight(s) found from the chart?

As we can clearly see that the relationship between the average pricing the listing counts is direct, the highest pricing is in the Entire Home/Apt room type, and slo the listings division, which can be quite using in making decisions like pricing.

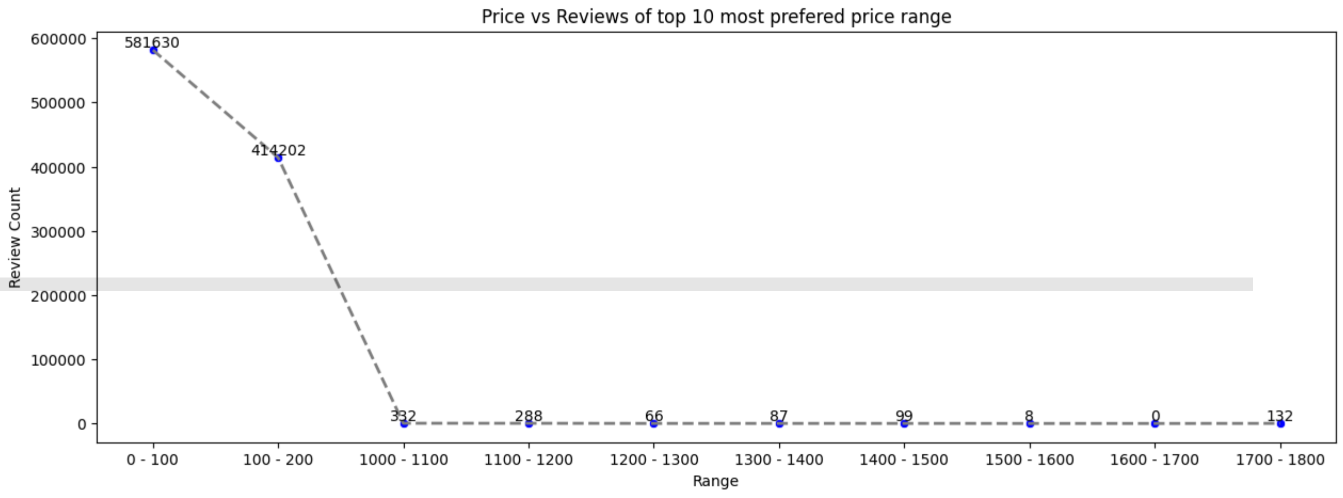
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

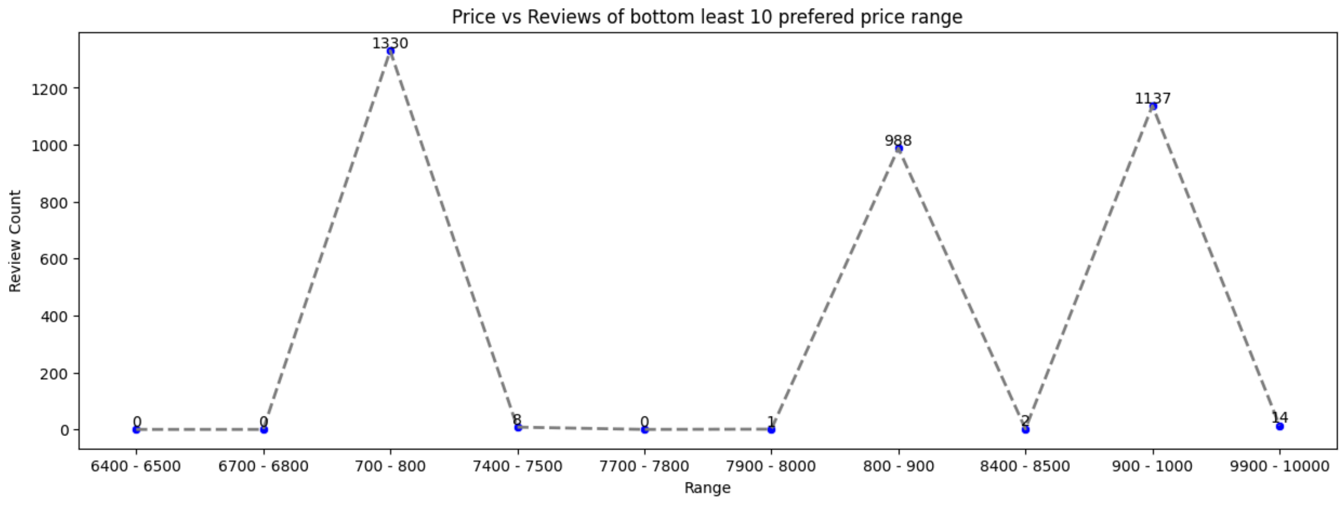
Yes, as we can see what percentage is ready to pay which amount for their preferences we will be able to make better pricing strategies which will definitely assist us in the growth.

Chart - 5

```
# Chart - 5 visualization code
# Let's visualize the relationship between listing price and the number of reviews received.
# As we are having huge number of rows for this we will be taking only top 10 price ranges.
# For the listing prices we will take price_range

sorted_range = feature_df.sort_values(by='price_range', ascending=False)
sorted_range = sorted_range.groupby('price_range')
range_group = []
review_count = []
for range, data in sorted_range:
  range_group.append(range)
  review_count.append(data['number_of_reviews'].sum())

range_df = pd.DataFrame({
    'Range': range_group,
    'Review Count': review_count
})
plt.figure(figsize=(15, 5))
sns.scatterplot(x='Range', y='Review Count', data=range_df.head(10), color='blue', legend=False, marker='o')
plt.plot(range_df['Range'].head(10), range_df['Review Count'].head(10), color='grey', linestyle='--', linewidth=2, label='Line Connectir
plt.title("Price vs Reviews of top 10 most prefered price range")
for x, y in zip(range_df['Range'].head(10), range_df['Review Count'].head(10)):
  plt.text(x, y, f'{y}', ha='center', va='bottom')
```



```
# Let us also check for the least 10 prefered price ranges
plt.figure(figsize=(15, 5))
sns.scatterplot(x='Range', y='Review Count', data=range_df.tail(10), color='blue', legend=False, marker='o')
plt.plot(range_df['Range'].tail(10), range_df['Review Count'].tail(10), color='grey', linestyle='--', linewidth=2, label='Line Connectir
plt.title("Price vs Reviews of bottom least 10 prefered price range")
for x, y in zip(range_df['Range'].tail(10), range_df['Review Count'].tail(10)):
  plt.text(x, y, f'{y}', ha='center', va='bottom')
```



1. Why did you pick the specific chart?

As the difference between the no. of reviews in the price range is very huge, using a plot chart is quiet handy so that the pointers can be seen clearly with respect to their values and also their differences.

2. What is/are the insight(s) found from the chart?

As we can see the price range and the no. of reviews are having inverse relationship, it cleary states that the budget of our majortiy customer base lies between the range 0 - 200.

We can also see that there are few preferences in the higher budget section as well where there is a competetion in the prices.

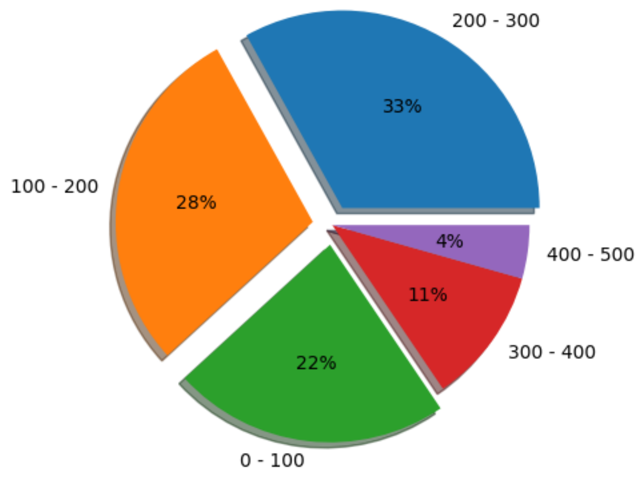
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

This suggests that we will be able to make our pricing policies more targeted and more specific resulting in increasing customer base by providing them with the prices that are under their budget.

Chart - 6

```
# Chart - 6 visualization code
# Let us know the price range having the highest listings so as to get more specific idea on price preference.


sorted_range # Sorted Price Range grouped
range_group # List of all the price range groups
range_listing_count = []

for range, data in sorted_range:
  range_listing_count.append(data['listings'].sum())


range_list_df = pd.DataFrame({
    'Range': range_group,
    'Listing Count': range_listing_count
})

range_list_df = range_list_df.sort_values(by='Listing Count', ascending=False).reset_index(drop=True)

# We will be taking on the top 5 prefered ranges.
plt.pie(range_list_df['Listing Count'].head(), labels=range_list_df['Range'].head(), autopct="%1i%%", explode=(0.1, 0.1, 0.1, 0, 0), sha
plt.show()
```



1. Why did you pick the specific chart?

Pie chart is an effective chat to clearly see the distrubutions and preferences, it becomes easier to notice the division of listngs by the price range.

2. What is/are the insight(s) found from the chart?

As we can see form the chart:- The top 3 price ranges that are having the most listings are:-

- 200-300: 33%
- 100-200: 28%

- 0-100: 22%

3. Will the gained insights help creating a positive business impact?

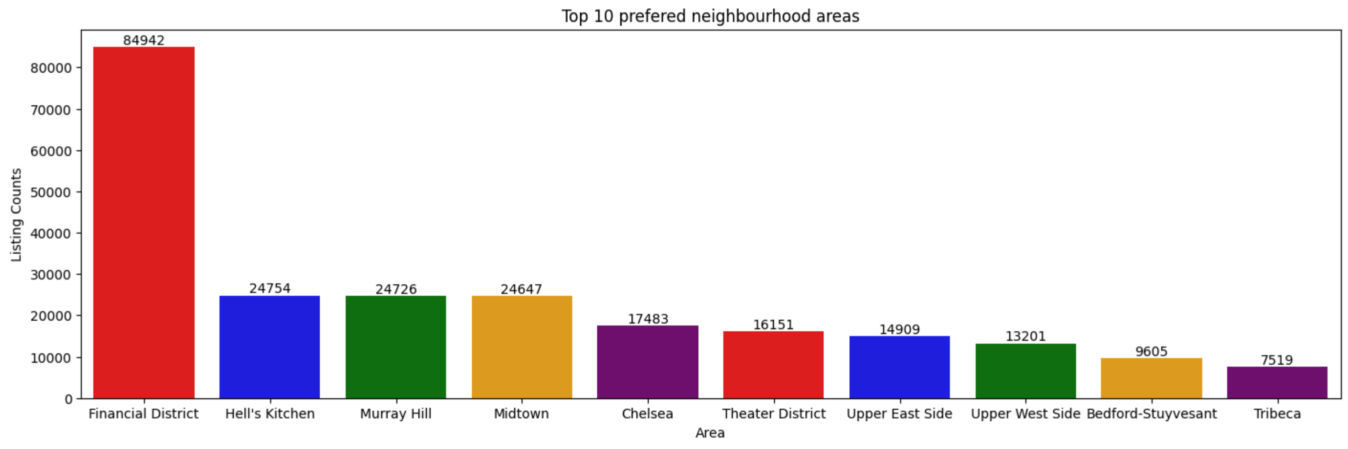
Are there any insights that lead to negative growth? Justify with specific reason.

We can see the number of listings are closely divided among these sectors and this can be a great was to keep a track of all the hostels that are within this price range and create recommendations according to their prefered neighbourhoods.

Chart - 7

```
# Chart - 7 visualization code
# As now we are having the price ranges with the most listings let us check the neighbourhoods with the most listings.

areas = []
area_listing = []
area_group = feature_df.groupby('neighbourhood')
for area, data in area_group:
    areas.append(area)
    area_listing.append(data['listings'].sum())

area_data_df = pd.DataFrame({
    "Area": areas,
    "Values": area_listing
})

area_data_df = area_data_df.sort_values(by='Values', ascending=False).reset_index(drop=True)
area_data_df
plt.figure(figsize=(17,5))
# plt.plot(groups,listings, color='black', marker = "o", markerfacecolor = 'blue', markeredgecolor='blue',linestyle='-')
colors = ['red', 'blue', 'green', 'orange', 'purple']
sns.barplot(x="Area", y='Values', data=area_data_df.head(10), hue="Area", palette=colors, legend=False)
plt.title('Top 10 prefered neighbourhood areas')
plt.xlabel('Area')
plt.ylabel('Listing Counts')
for x, y in zip(area_data_df['Area'].head(10), area_data_df['Values'].head(10)):
    plt.text(x, y, f'{y}', ha='center', va='bottom')
```

```
<ipython-input-44-30fc849892e1>:21: UserWarning:
The palette list has fewer values (5) than needed (10) and will cycle, which may produce an uninterpretable plot.
  sns.barplot(x="Area", y='Values', data=area_data_df.head(10), hue="Area", palette=colors, legend=False)
```



1. Why did you pick the specific chart?

The bar chart clearly reflects the differece between the areas and the gap between them as per the listing counts, we can see the top ten most listed neighbourhoods.

2. What is/are the insight(s) found from the chart?

We can see that the most listed neighbourhood area is the 'Financial District' and it is also having a major gap between the other ones in the list, we can clearly see that the Financial District is the most prefered neighbourhood of our customers.

3. Will the gained insights help creating a positive business impact?

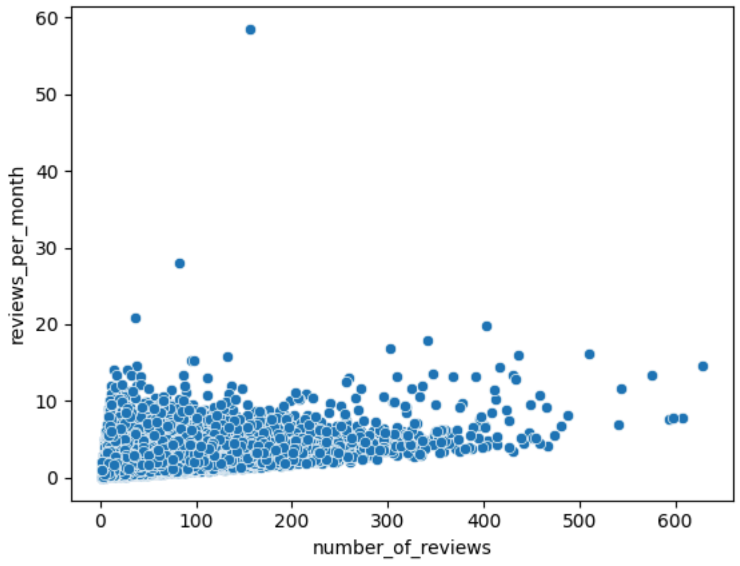
Are there any insights that lead to negative growth? Justify with specific reason.

↳ 1 cell hidden

Chart - 8

```
# Chart - 9 visualization code
# Reviews vs Review per month
sns.scatterplot(x='number_of_reviews', y='reviews_per_month', data=feature_df)
```

```
<Axes: xlabel='number_of_reviews', ylabel='reviews_per_month'>
```



1. Why did you pick the specific chart?

In this case using a scatter plot can easily tell us the frequency, as we can see the review traffic is not that scattere and is rather collective.

2. What is/are the insight(s) found from the chart?

As we can clearly see that there is an outlier as well, however the relationship between the reviews and reviews per month is quiet direct which means that the customer are getting satisfied with the outcomes, where the number of review is higher and the host is also getting good reach towards consumer market.

3. Will the gained insights help creating a positive business impact?

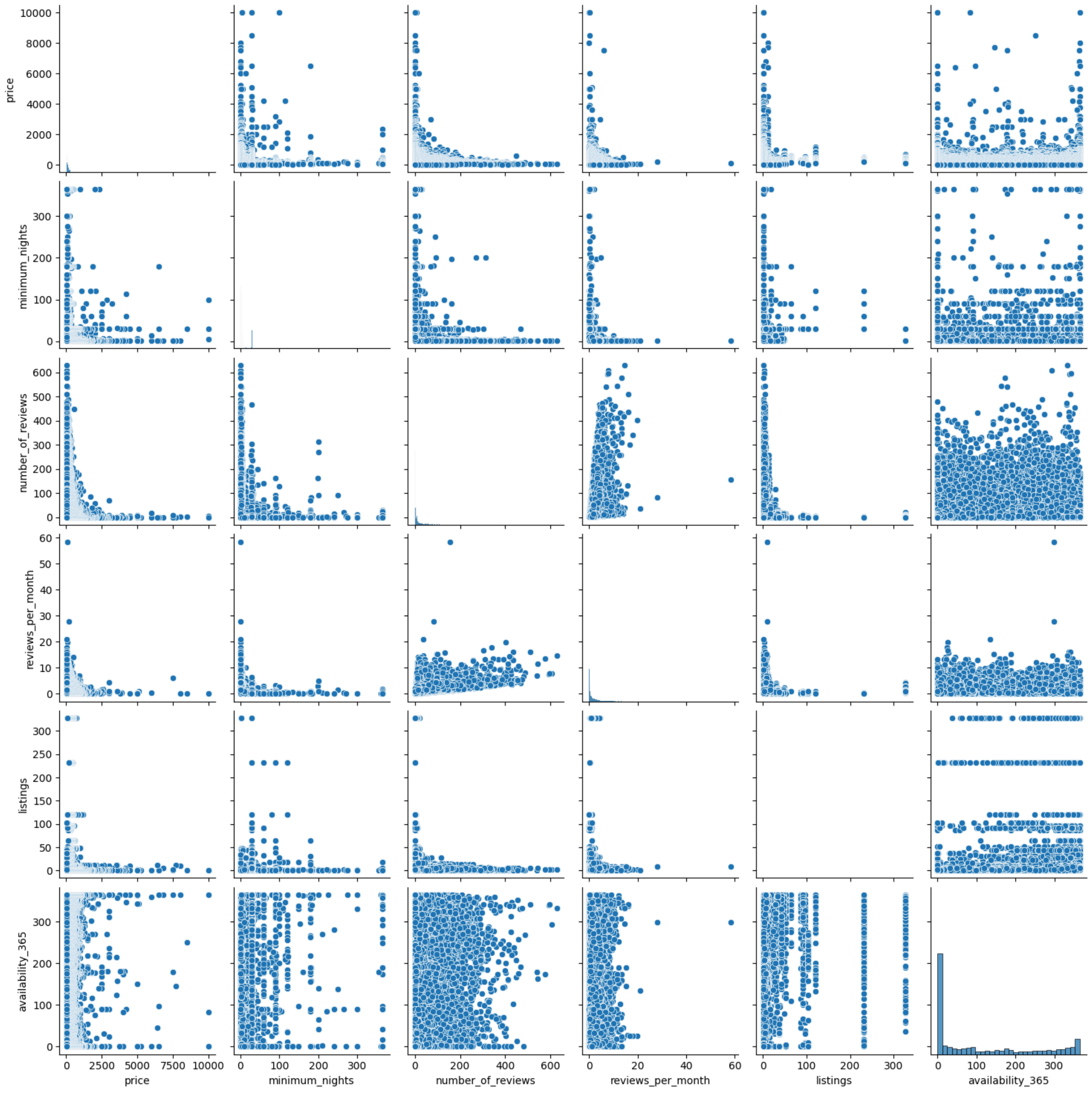
Are there any insights that lead to negative growth? Justify with specific reason.

Yes as we can seek out the difference from the given reviews and make specific recommendations to our hosts about what changes can be done by them in order to increase their reach to the customers. This can be one of our premium services that can widely help our hosts.

Chart - 9 - Pair Plot

```
# Chart - 9 visualization code
# Let's create a pairplot to know the relationship between our few variables.

pair_df = feature_df[['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'listings', 'availability_365', 'price_range
sns.pairplot(pair_df)
```

```
<seaborn.axisgrid.PairGrid at 0x7e63c56a7010>
```



1. Why did you pick the specific chart?

Here we can easily see the distribution of our numerical variables, it will also give us an overview of our dataset and the relationships between our variables.

2. What is/are the insight(s) found from the chart?

Yes, we can see that the utmost majority of our complete data set is within the price range of 0-2500.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

This insight can give us an overview of our dataset so that we can have an idea about the ranges of the numerical values in which we are