AIRBNB Case Study

Methodology Document PPT 1:

In the case study we have used Jupiter notebook to perform initial analysis of the data and Tableau for data analysis and visualization.

Initial Analysis using Jupiter Notebook: Data Set Used: AB_NYC_2019.csv

Number of Rows: 48895

Number of Columns: 16

```
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#Remove warnings in kernel while running a cell
import warnings
warnings.filterwarnings('ignore')

#notebook setting to display all the rowns and columns .
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('display.max_columns',None)
```

```
# Importing df_Leads.csv and vewing the dataframe

df_nyc = pd.read_csv('AB_NYC_2019.csv')
df_nyc.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-1
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-0
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-0
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-1

```
df_nyc.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
```

```
df_nyc.shape
```

```
(48895, 16)
```

```
df_nyc.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	a
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	

Data Cleaning

Duplicate Value Check

```
# Checking Duplicates
df_nyc.duplicated().sum()
```

```
0
```

Missing values check

```
List =  list(df_nyc.columns)
print(List)
```

```
['id', 'name', 'host_id', 'host_name', 'neighbourhood_group', 'neighbourhood', 'latitude', 'longitude', 'room_type', 'price', 'minimum_nights', 'number_o
f_reviews', 'last_review', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365']
```

```
#looking to find out first what columns have null values
#using 'sum' function will show us how many nulls are found in each column in dataset
df_nyc.isnull().sum()
```

```
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
```

```
# Checking Null Values Percentage in the dataset
(df_nyc.isnull().sum()/len(df_nyc)*100).sort_values(ascending=False)
```

```
last_review                       20.558339
reviews_per_month                 20.558339
host_name                          0.042949
name                               0.032723
id                                 0.000000
host_id                            0.000000
neighbourhood_group                0.000000
neighbourhood                      0.000000
latitude                           0.000000
longitude                          0.000000
room_type                          0.000000
price                              0.000000
minimum_nights                     0.000000
number_of_reviews                  0.000000
calculated_host_listings_count     0.000000
availability_365                   0.000000
dtype: float64
```

```
# checking row wise missing values
((df_nyc.isnull().sum(axis=1))/(df_nyc.shape[1])*100).sort_values(ascending=False)
```

```
6605     18.75
38992    18.75
18047    18.75
6567     18.75
16071    18.75
         ...
17948     0.00
17949     0.00
17950     0.00
17951     0.00
24447     0.00
Length: 48895, dtype: float64
```

```
for i in List:
    print(i)
```

```
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
```

```
df_nyc.neighbourhood_group.value_counts()
```

```
neighbourhood_group
Manhattan        21661
Brooklyn         20104
Queens            5666
Bronx             1091
Staten Island      373
Name: count, dtype: int64
```

```
df_nyc.neighbourhood.value_counts()
```

```
neighbourhood
Williamsburg                3920
Bedford-Stuyvesant          3714
Harlem                      2658
Bushwick                    2465
Upper West Side             1971
Hell's Kitchen              1958
East Village                1853
Upper East Side             1798
Crown Heights               1564
Midtown                     1545
East Harlem                 1117
Greenpoint                  1115
Chelsea                     1113
Lower East Side              911
Astoria                      900
Washington Heights           899
West Village                 768
```

```
df_nyc.room_type.value_counts()
```

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

```
df_nyc[df_nyc.last_review.isna() == True]
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_revi
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	
19	7750	Huge 2 BR Upper East Cental Park	17985	Sing	Manhattan	East Harlem	40.79685	-73.94872	Entire home/apt	190	7	
26	8700	Magnifique Suite au N de Manhattan - vue Cloitres	26394	Claude & Sophie	Manhattan	Inwood	40.86754	-73.92639	Private room	80	4	
36	11452	Clean and Quiet in Brooklyn	7355	Vt	Brooklyn	Bedford-Stuyvesant	40.68876	-73.94312	Private room	35	60	
38	11943	Country space in the city	45445	Harriet	Brooklyn	Flatbush	40.63702	-73.96327	Private room	150	1	
...	...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	

10052 rows × 16 columns

```
df_nyc.last_review.value_counts()
```

```
last_review
23-06-2019    1413
01-07-2019    1359
30-06-2019    1341
24-06-2019     875
07-07-2019     718
              ...
25-12-2012       1
01-10-2013       1
29-05-2014       1
19-04-2014       1
29-03-2018       1
Name: count, Length: 1764, dtype: int64
```

```
df_nyc['last_review'] = pd.to_datetime(df_nyc['last_review'])
```

```
df_nyc.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  datetime64[ns]
 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: datetime64[ns](1), float64(3), int64(7), object(5)
memory usage: 6.0+ MB
```

- We removed the columns like Name, Host Name which was not giving much information.

```
df_nyc.drop(['name','host_name'], axis=1, inplace=True)
df_nyc.head()
```

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month
0	2539	2787	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-19	0.21
1	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	0.38
2	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaT	NaN
3	3831	4869	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	4.64
4	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	0.10

```
Q1 = df_nyc["reviews_per_month"].quantile(0.25)
Q3 = df_nyc["reviews_per_month"].quantile(0.75)
Q4 = df_nyc["reviews_per_month"].quantile(0.95)
IQR = Q3 - Q1
upper = Q3 + 1.5*IQR
lower = Q1 - 1.5*IQR
```

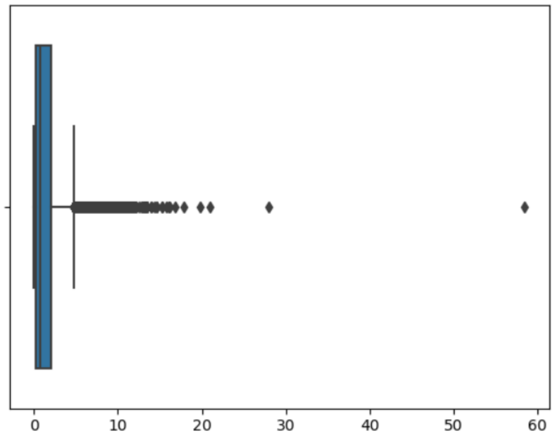
```
print(Q1)
print(Q4)
print(Q3)
print(IQR)
print(upper)
print(lower)
```

```
0.19
4.64
2.02
1.83
4.765000000000001
-2.555
```

```python
if (df_nyc["reviews_per_month"].max() - upper == 0) or (abs(df_nyc["reviews_per_month"].min()) - abs(lower)== 0):
    print(True)
```

```python
sns.boxplot(data=df_nyc,x="reviews_per_month")
```

```
<Axes: xlabel='reviews_per_month'>
```




```python
df_nyc['reviews_per_month'] = df_nyc['reviews_per_month'].fillna(df_nyc['reviews_per_month'].median())
```

```python
df_nyc.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_h
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	2018-10-04 01:47:23.910099456	1.238930	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	2011-03-28 00:00:00	0.010000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	2018-07-08 00:00:00	0.280000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	2019-05-19 00:00:00	0.720000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2019-06-23 00:00:00	1.580000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	2019-07-08 00:00:00	58.500000	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	NaN	1.520861	

```
df_nyc
```

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_pe
0	2539	2787	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-19	
1	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	
2	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaT	
3	3831	4869	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	
4	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	
...	...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	8232441	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	NaT	
48891	36485057	6570630	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	NaT	
48892	36485431	23492952	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	NaT	
48893	36485609	30985759	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	NaT	
48894	36487245	68119814	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	NaT	

48895 rows × 14 columns

```
df_nyc.to_csv("Final_Nyc.csv")
```

Step 2: Data Wrangling:

- Checked the Duplicate rows in our dataset and no duplicate data was found.
- Checked the Null Values in our dataset. Columns like name, host-name, last review and review-per-month have null values.
- We've dropped the column name as missing values are less and dropping it won't have significant impact on analysis.
- Checked the formatting in our dataset.
- Identified and review outliers.

Data Analysis and Visualizations using Tableau:

- We have used tableau to visualize the data for the assignment. Below are the detailed steps used for each visualization.

1) Top 10 Host:

- We identified the top 10 Host Ids, Host Name with count of Host Ids using the tree map.

2) Preferred Room type with respect to Neighbourhood group:

- We created a pie chart for understanding the percentage of room type preferred w r t neighbourhood group
- We added Room Type to the colours Marks card to highlight the different Room Type in different colours and count of Host Id to the size

3) For Variance of price with Neighbourhood Groups:

- We used a box and whisker's plot with Neighbourhood Groups in Columns and Price in Rows.
- We changed the Price from a Sum Measure to the median measure.

4) Average price of Neighbourhood groups:

- We created a bubble chart with Neighbourhood Groups in Columns and Price column in Rows.
- We added the Neighbourhood Groups to the colours Marks card to highlight the different neighbourhood Groups in different colours. Also Put Avg price in Label.

5) Customer Booking w r t minimum nights:

- We created the bin for Minimum nights as shown below.

- The bins were used to display the distribution of minimum nights based on the number of ids booked for each neighbourhood group.

6) Popular Neighbourhoods:

- We took neighbourhood in rows and sum of reviews in column and took neighbourhood groups in colour.
- We used filter to show Top 20 neighbours as per the sum of reviews.

7) Neighbourhood vs Availability:

- We created a dual axis chart using bar chart for availability 365 and line chart for price for top 10 neighbourhood group sorted by price.

<u>Methodology Document PPT 2:</u>

1) Room type with respect to Neighbourhood group:

- We created a pie chart for understanding the percentage of room type preferred w r t neighbourhood group
- We added Room Type to the colours Marks card to highlight the different Room Type in different colours and count of Host Id to the size

2) Customer Booking with respect to minimum nights:

- We created the bin for Minimum nights as shown below.

- The bins were used to display the distribution of minimum nights based on the number of ids booked for each neighbourhood group.

3) Neighbourhood vs Availability:

- We created a dual axis chart using bar chart for availability 365 and line chart for price for top 10 neighbourhood group sorted by price.

4) Price range preferred by Customers:

- We have taken pricing preference based on volume of bookings done in a price range and no of Ids to create a bar chart. We have created bin for Price column with interval of $20.

5) Understanding Price variation w.r.t Room Type & Neighbourhood:

- We created Highlights Table chat by taking Room Type in rows & Neighbourhood Group in column.
- We took the average price in colour Marks card to highlight the different Room Type in different colours.

6) Price variation w r t Geography:

- We used Geo location chart to plot neighbourhood, neighbourhood Group in map to show case the variation of prices across.

7) Popular Neighbourhoods:

- We took neighbourhood in rows and sum of reviews in column and took neighbourhood groups in colour.
- We used filter to show Top 20 neighbours as per the sum of reviews.

8) Tools used:

- Data cleaning and preparation: Jupyter notebook – Python
- Visualization and analysis: Tableau
- Data Storytelling: Microsoft PPT