Cleaning Data

Data cleaning is the process of fixing or removing incorrect, corrupted, duplicate, or incomplete data within a dataset. Messy data leads to unreliable outcomes. Cleaning data is an essential part of data analysis, and demonstrating your data cleaning skills is key to landing a job.Here we will work on a data set and will show how it should be done.

Import Library

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

Import Dataset

```
ny=pd.read_csv('/AB_NYC_2019.csv')
```

Will Display the first 10 rows

```
ny.head(10)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill	40
6	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40
7	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Hell's Kitchen	40
8	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40
9	5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Chinatown	40

Next steps: Generate code with ny View recommended plots

Data Integrity

For correct data types we'll use:

```
ny.dtypes
```

```
id                                 int64
name                              object
host_id                            int64
host_name                         object
neighbourhood_group               object
neighbourhood                     object
latitude                         float64
longitude                        float64
room_type                         object
price                              int64
minimum_nights                     int64
number_of_reviews                  int64
last_review                       object
reviews_per_month                float64
calculated_host_listings_count     int64
availability_365                   int64
dtype: object
```

converting date columns into datetime

```
ny['last_review']=pd.to_datetime(ny['last_review'],errors='coerce')
```

Detailed Statistics

```
ny.describe(include='all')
```

bourhood_group	neighbourhood	latitude	longitude	room_type	
48895	48895	48895.000000	48895.000000	48895	4889
5	221	NaN	NaN	3	
Manhattan	Williamsburg	NaN	NaN	Entire home/apt	
21661	3920	NaN	NaN	25409	
NaN	NaN	40.728949	-73.952170	NaN	15
NaN	NaN	40.499790	-74.244420	NaN	
NaN	NaN	40.690100	-73.983070	NaN	6
NaN	NaN	40.723070	-73.955680	NaN	10
NaN	NaN	40.763115	-73.936275	NaN	17
NaN	NaN	40.913060	-73.712990	NaN	1000
NaN	NaN	0.054530	0.046157	NaN	24

Checking for missing values if any

```
ny.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
```

now handle missing values

for example:by using fillna to fill missing values in 'reviews_per_month' with the median

```
ny['reviews_per_month']=ny['reviews_per_month'].fillna(ny['reviews_per_month'].median(),inplace=True)
```

Now we'll drop rows where 'last_review' is missing

```
ny.dropna(subset=['last_review'],inplace=True)
```

Now we will remove the **Duplicates**

```
duplicates=ny.duplicated()
```

```
ny.drop_duplicates(inplace=True)
```

```
f"number of duplicates: {duplicates.sum()}"
```

```
'number of duplicates: 0'
```

Now Standardization

We will Standardize the 'price' To float

```
ny['price']=ny['price'].astype(float)
```

For example standardize 'room_type'to lowercase

```
ny['room_type'] = ny['room_type'].str.lower()
```

To check consistency displaying unique values

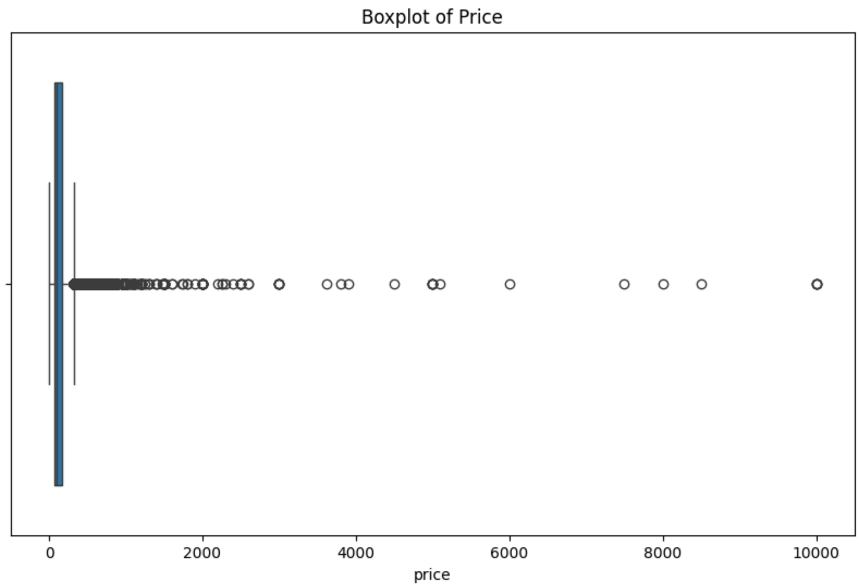
```
ny['room_type'].unique()
```

```
array(['private room', 'entire home/apt', 'shared room'], dtype=object)
```

Now Outlier Detection

Visualising outliers in the price column using a boxplot

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=ny['price'])
plt.title('Boxplot of Price')
plt.show()
```



For example removing outliers outside 1.5*IQR range

```
Q1=ny['price'].quantile(0.25)
```

```
Q3=ny['price'].quantile(0.75)
```

```
IQR=Q3-Q1
```

```
Lower_limit=Q1-1.5*IQR
```

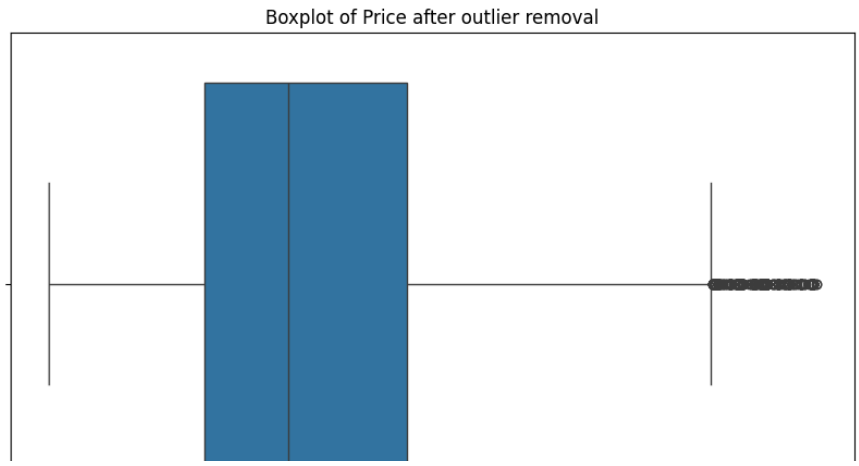
```
Upper_limit=Q3+1.5*IQR
```

Now filtering out outliers

```
ny=ny[(ny['price']>=Lower_limit) & (ny['price']<=Upper_limit)]
```

Now we'll check final changes

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=ny['price'])
plt.title('Boxplot of Price after outlier removal')
plt.show()
```


Boxplot of Price after outlier removal