Internship project ¶

Title: Data Cleaning Analysis Report

Subtitle: Ensuring Data Quality for Reliable Insights

Presented by: Faleye Doyin Opeyemi

Date: 12-8-2025

1. Data Loading & Data collection

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# load dataset
df = pd.read_csv("C:\\Users\\FALEYE DOYINSOLA\\project 3 cleaning analysis.cs
```

In [5]:
```
# display first few rows
df.head()
```

Out[5]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitud
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6474
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7536
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8090
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6851
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7985

2. Data integrity Check

In [6]:
```
# check dataset information
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
```

In [7]:
```
# checking for datatype of the dataset
df.dtypes
```

Out[7]:
```
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
```

In [8]:
```
# summary statistics
df.describe()
```

Out[8]:

	id	host_id	latitude	longitude	price	minimum_nights
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000

In [9]:
```
# Checking for the rows and columns
df.shape
```

Out[9]:
```
(48895, 16)
```

In [10]:
```
df.columns
```

Out[10]:
```
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')
```

In [11]:
```
# removal of inrelevant columns
df.drop(['id','host_id'], axis =1 , inplace = True)
```

In [12]:
```
df.head()
```

Out[12]:

	name	host_name	neighbourhood_group	neighbourhood	latitude	longitude	ro
0	Clean & quiet apt home by the park	John	Brooklyn	Kensington	40.64749	-73.97237	
1	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	40.75362	-73.98377	
2	THE VILLAGE OF HARLEM....NEW YORK !	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	
3	Cozy Entire Floor of Brownstone	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	
4	Entire Apt: Spacious Studio/Loft by central park	Laura	Manhattan	East Harlem	40.79851	-73.94399	

In [13]:
```
df.shape
```

Out[13]:
```
(48895, 14)
```

2. Missing Data Handling

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
name                                 16
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
```

In [15]:
```
# checking for the unique data of 'last_review' column
df['last_review'].unique()
```

Out[15]:
```
array(['2018-10-19', '2019-05-21', nan, ..., '2017-12-23', '2018-01-29',
       '2018-03-29'], dtype=object)
```

In [16]:
```
# Handling the missing data
df.drop(['last_review','reviews_per_month'], axis =1 , inplace = True)
```

In [17]:
```
df.head()
```

Out[17]:

	name	host_name	neighbourhood_group	neighbourhood	latitude	longitude	ro
0	Clean & quiet apt home by the park	John	Brooklyn	Kensington	40.64749	-73.97237	
1	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	40.75362	-73.98377	
2	THE VILLAGE OF HARLEM....NEW YORK !	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	
3	Cozy Entire Floor of Brownstone	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	
4	Entire Apt: Spacious Studio/Loft by central park	Laura	Manhattan	East Harlem	40.79851	-73.94399	

In [18]:
```
# checking for the accuracy of the rows and columns after droping some colum
df.shape
```

Out[18]:
```
(48895, 12)
```

In [19]:
```
# checking for missing values
df.isnull().sum()
```

Out[19]:
```
name                              16
host_name                         21
neighbourhood_group                0
neighbourhood                      0
latitude                           0
longitude                          0
room_type                          0
price                              0
minimum_nights                     0
number_of_reviews                  0
calculated_host_listings_count     0
availability_365                   0
dtype: int64
```

In [20]:
```
# drop missing value
df.dropna(inplace=True)
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
name                              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
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

3. Duplicate Removal

In [22]:
```
#check for duplicate
duplicated = df.duplicated()
```

In [23]:
```
duplicated.sum() # there is no duplicate
```

Out[23]:
```
0
```

In [24]:
```
df.head()
```

Out[24]:

	name	host_name	neighbourhood_group	neighbourhood	latitude	longitude	ro
0	Clean & quiet apt home by the park	John	Brooklyn	Kensington	40.64749	-73.97237	
1	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	40.75362	-73.98377	
2	THE VILLAGE OF HARLEM....NEW YORK !	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	
3	Cozy Entire Floor of Brownstone	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	
4	Entire Apt: Spacious Studio/Loft by central park	Laura	Manhattan	East Harlem	40.79851	-73.94399	

4. Standardization

In [25]:
```
#standardized all text column
textcol = ['name','host_name','neighbourhood_group','neighbourhood','room_typ
```

In [26]:
```
textcol
```

Out[26]:
```
['name', 'host_name', 'neighbourhood_group', 'neighbourhood', 'room_type']
```

In [27]:
```
df[textcol]= df[textcol].apply(lambda col: col.str.strip().str.lower())
```

In [28]:
```
df.head()
```

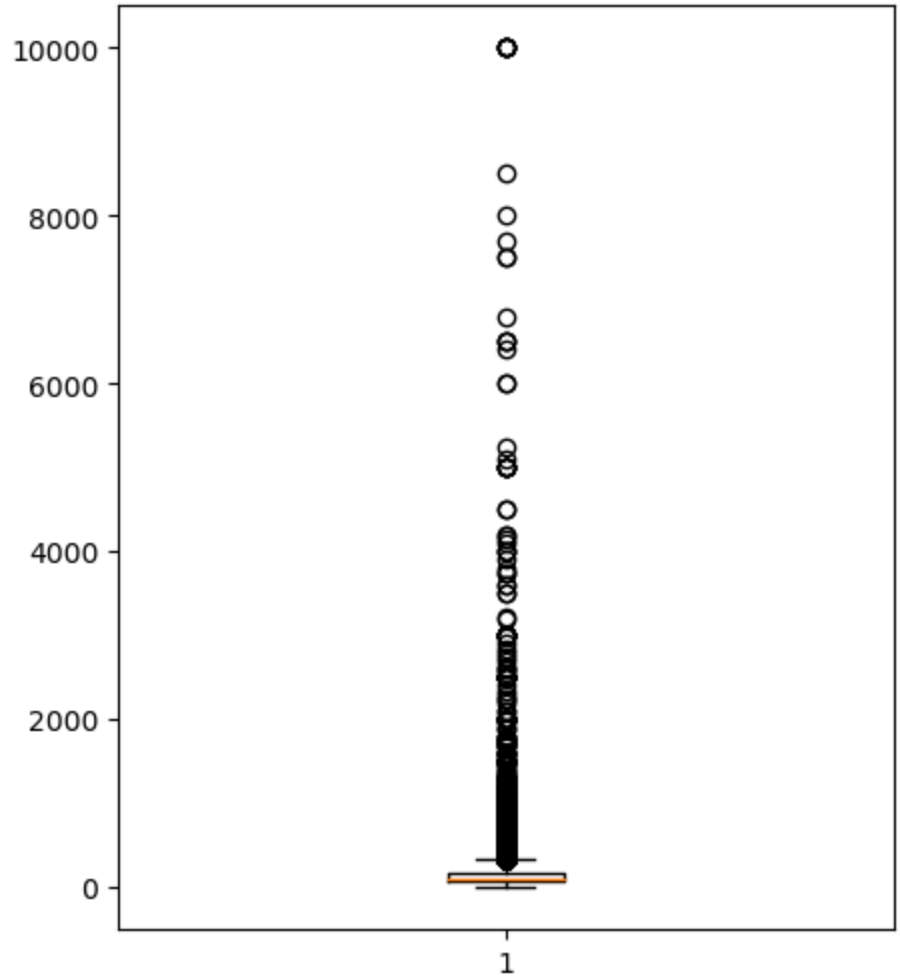
Out[28]:

	name	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_
0	clean & quiet apt home by the park	john	brooklyn	kensington	40.64749	-73.97237	pr r
1	skylit midtown castle	jennifer	manhattan	midtown	40.75362	-73.98377	e home
2	the village of harlem....new york !	elisabeth	manhattan	harlem	40.80902	-73.94190	pr r
3	cozy entire floor of brownstone	lisaroxanne	brooklyn	clinton hill	40.68514	-73.95976	e home
4	entire apt: spacious studio/loft by central park	laura	manhattan	east harlem	40.79851	-73.94399	e home

5. Outlier Detection & Removal

In [29]:
```
# to detect or determine the outlier we plot a boxplot
plt.figure(figsize=(5,6))
plt.boxplot(df['price'])
plt.show
```

Out[29]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [30]:
```
# outlier formular -•
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3-Q1
```

In [31]:
```
# threhold formular
threshold = 1.5
```

In [32]:
```
# upperbound and lowerbound formular aferthe outlier
lowerbound = Q1 - threshold * IQR
upperbound = Q3 + threshold * IQR
```

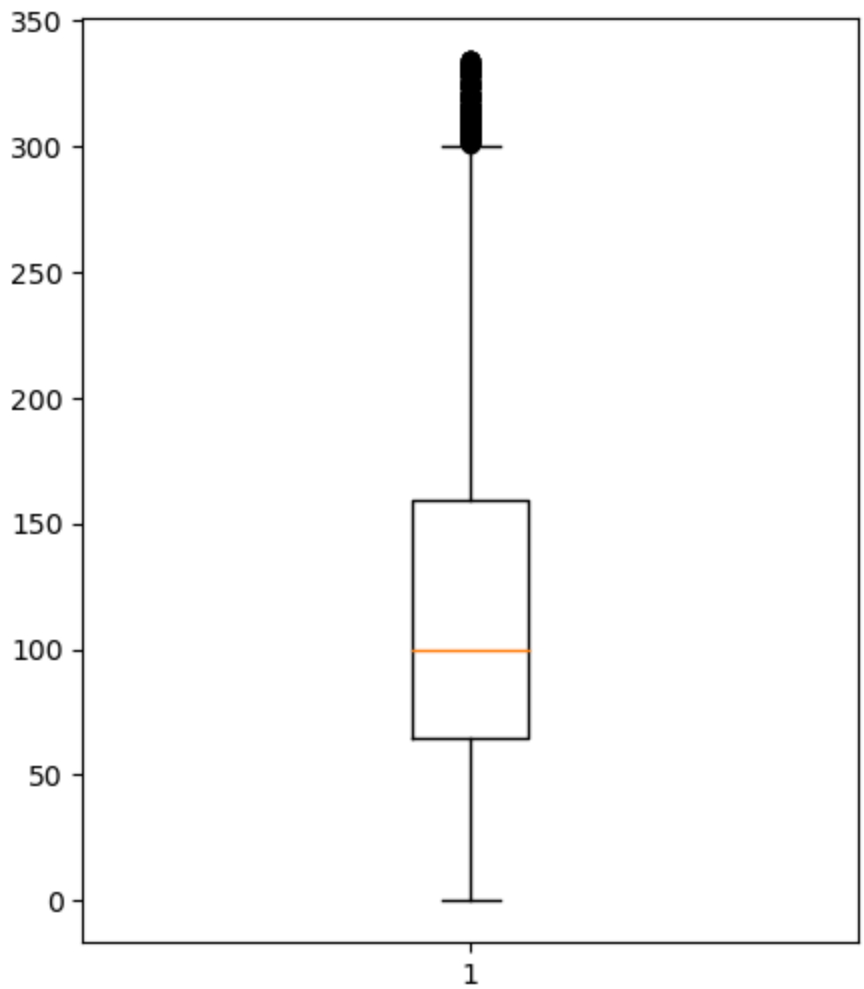
In [33]:
```
#remove outlier from df price
df = df[(df['price'] >= lowerbound) & (df['price'] <= upperbound)]
```

In [34]:
```
# checking for the accuracy of the rows and columns  after cleaning
df.shape
```

Out[34]: (45887, 12)

In [35]:
```
# re-detecting after the removal of outlier
plt.figure(figsize=(5,6))
plt.boxplot(df['price'])
plt.show
```

Out[35]: <function matplotlib.pyplot.show(close=None, block=None)>



In [ ]: