```
#Importing necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer

# Load the data Set
df = pd.read_csv('D:\Internship data\AB_NYC_2019.csv')

#Taking a look at the top 5 rows of the data
df.head()
```

```
     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
```

```
# Checking column types
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
```

```
# Reviewing data columns
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')
```

```
# Looking for missing values
df.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
```

```
# Assessing unique values
df.nunique()
```

```
id                                48895
name                              47905
host_id                           37457
host_name                         11452
neighbourhood_group                   5
neighbourhood                       221
latitude                          19048
longitude                         14718
room_type                             3
price                               674
minimum_nights                      109
number_of_reviews                   394
last_review                        1764
reviews_per_month                   937
calculated_host_listings_count       47
availability_365                    366
dtype: int64
```

```
# Impute missing values with most frequent value for categorical
columns
imputer = SimpleImputer(strategy='mean')
df[['price', 'minimum_nights', 'number_of_reviews']] =
imputer.fit_transform(df[['price', 'minimum_nights',
'number_of_reviews']])

print("Duplicates:", df.duplicated().sum())
```

```
Duplicates: 0
```

```
# Convert date column to datetime format
df['last_review'] = pd.to_datetime(df['last_review'])
```

```python
# Print the DataFrame to see the converted datetime column
print(df['last_review'].head())
```

```
0   2018-10-19
1   2019-05-21
2          NaT
3   2019-07-05
4   2018-11-19
Name: last_review, dtype: datetime64[ns]
```

```python
# Encode categorical columns
from category_encoders import OrdinalEncoder
encoder = OrdinalEncoder()
df['neighbourhood_encoded'] =
encoder.fit_transform(df['neighbourhood'])
df['room_type_encoded'] = encoder.fit_transform(df['room_type'])

# Print the DataFrame to see the encoded values
print(df['neighbourhood_encoded'].describe())
print(df['room_type_encoded'].describe())
```

```
count    45923.000000
mean        31.246325
std         34.836710
min          1.000000
25%          9.000000
50%         20.000000
75%         37.000000
max        219.000000
Name: neighbourhood_encoded, dtype: float64
count    45923.000000
mean         1.545805
std          0.545408
min          1.000000
25%          1.000000
50%          2.000000
75%          2.000000
max          3.000000
Name: room_type_encoded, dtype: float64
```

```python
# Check for invalid values
print("Invalid values:", df['price'][~df['price'].between(0, 1000)])
```

```
Invalid values: 496      2000.0
762      1300.0
946      3000.0
1105     1300.0
1480     2000.0
          ...
48080    1308.0
48304    2999.0
```

```
48305    1999.0
48523    1369.0
48535    1749.0
Name: price, Length: 239, dtype: float64
```

```
# Check for inconsistencies
print("Inconsistencies:", df['minimum_nights'][df['minimum_nights'] <
1])
```

```
Inconsistencies: Series([], Name: minimum_nights, dtype: float64)
```

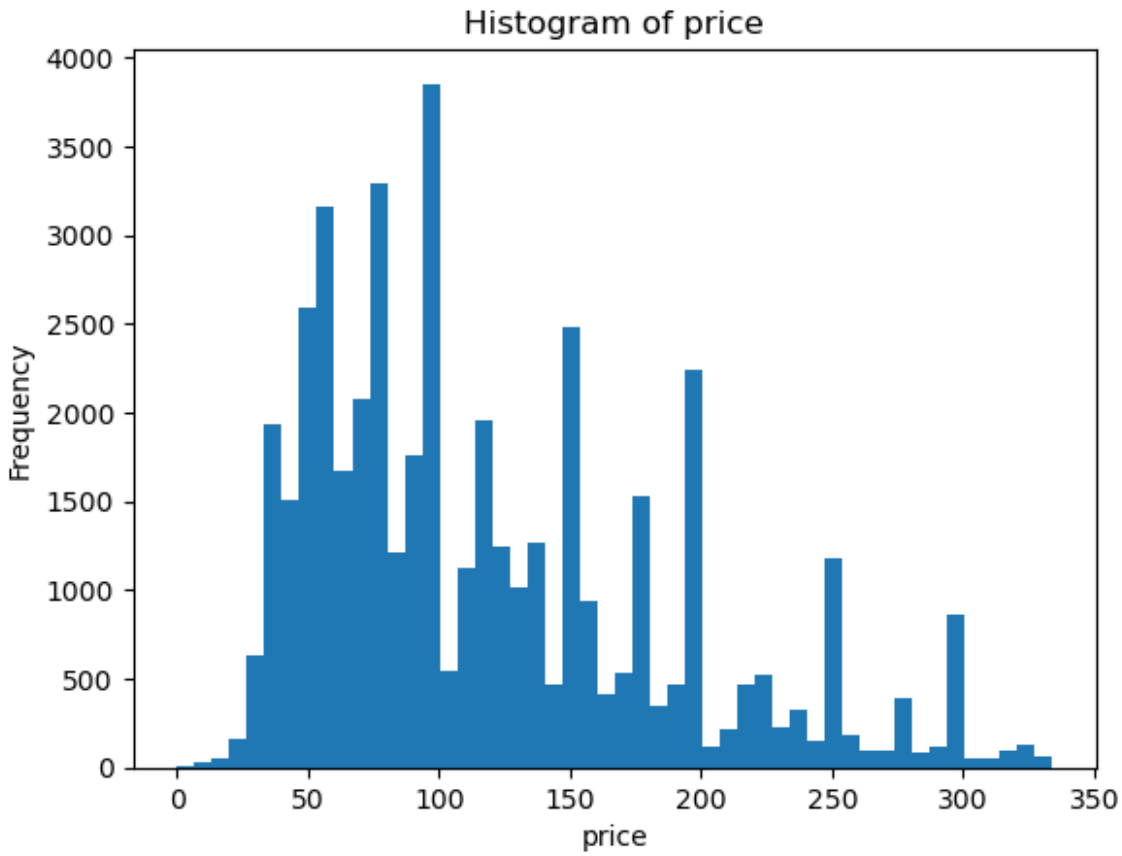
```
# Check for outliers in price column
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1
print("Outliers:", df['price'][~((df['price'] >= Q1 - 1.5 * IQR) &
(df['price'] <= Q3 + 1.5 * IQR))])
```

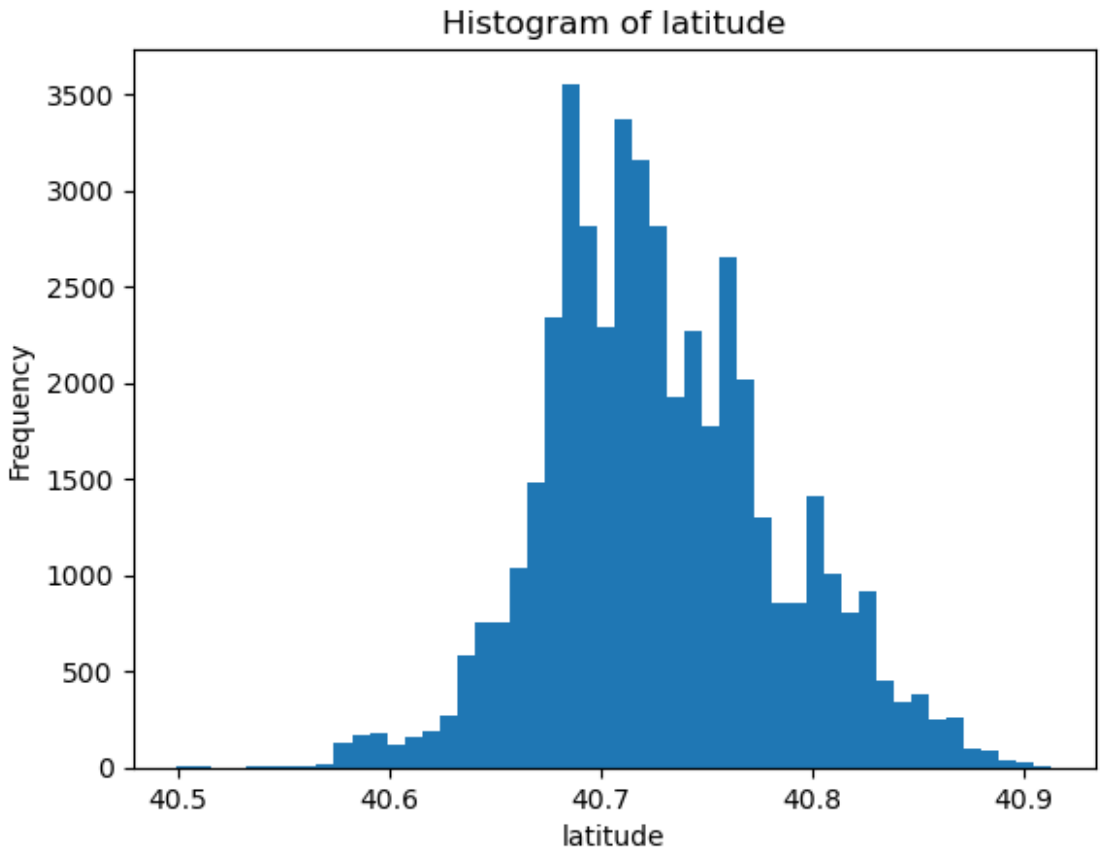
```
Outliers: 61       375.0
85       800.0
103      500.0
114      350.0
121      400.0
         ...
48758    350.0
48833    475.0
48839    800.0
48842    350.0
48856    345.0
Name: price, Length: 2972, dtype: float64
```

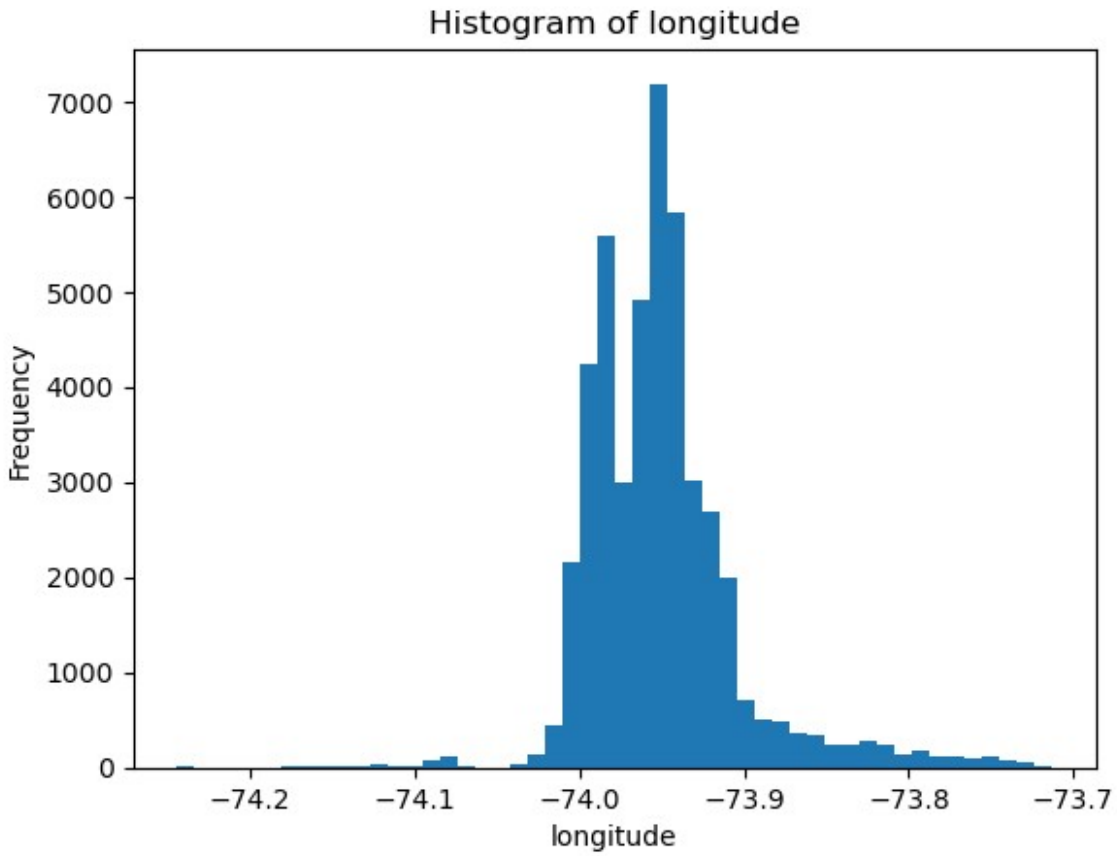
```
# Remove outliers
df = df[~((df['price'] < Q1 - 1.5 * IQR) | (df['price'] > Q3 + 1.5 *
IQR))]
```

```
# To visualize the data make graphs
# Select Numeric Columns
numeric_cols = ['price', 'latitude', 'longitude']
```

```
# Histograms
for col in numeric_cols:
    plt.hist(df[col], bins=50)
    plt.title(f"Histogram of {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()
```

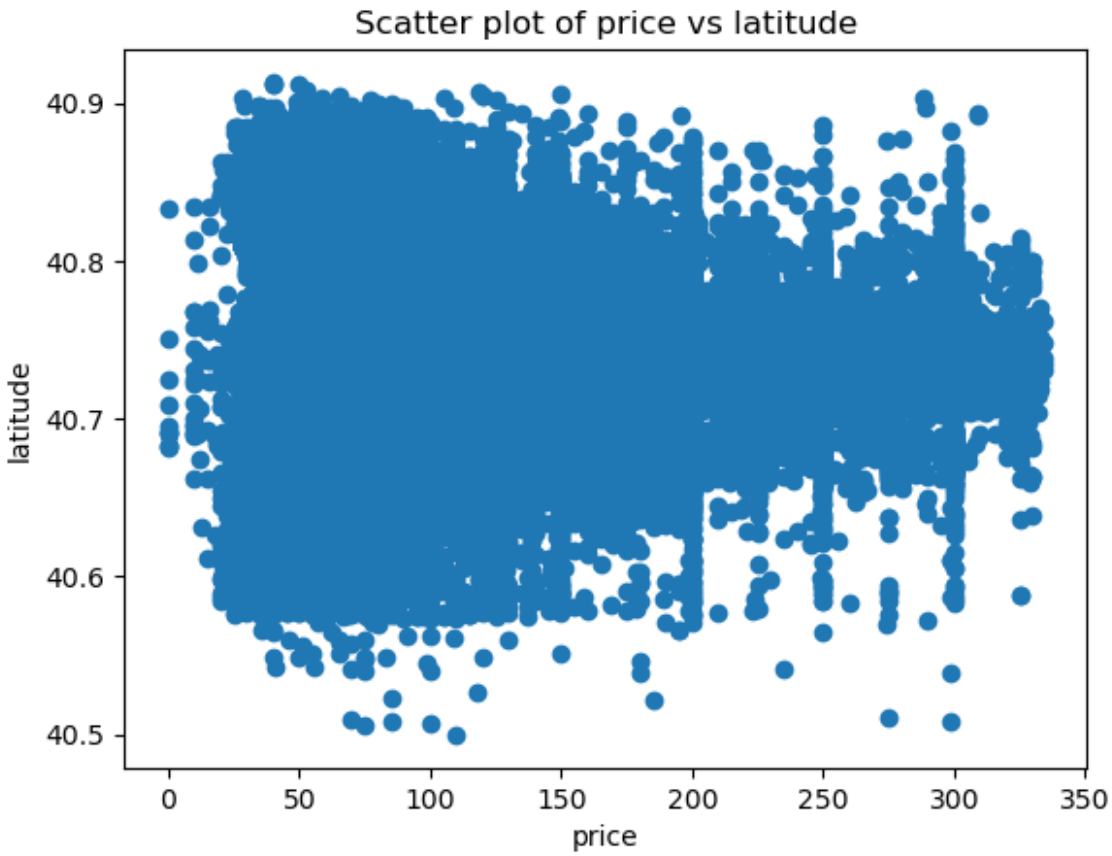

Histogram of price
Frequency
price
0
500
1000
1500
2000
2500
3000
3500
4000
0
50
100
150
200
250
300
350


Histogram of latitude
Frequency
3500
3000
2500
2000
1500
1000
500
0
40.5
40.6
40.7
40.8
40.9
latitude




```
#  Select Two Numeric Columns
x_col = 'price'
y_col = 'latitude'

# For Scatter Plot
plt.scatter(df[x_col], df[y_col])
plt.title(f"Scatter plot of {x_col} vs {y_col}")
plt.xlabel(x_col)
plt.ylabel(y_col)
plt.show()
```




```
# Select a Numeric Column
col = 'price'

# Calculate IQR
q1 = df[col].quantile(0.25)
q3 = df[col].quantile(0.75)
iqr = q3 - q1

# Create a Box Plot
plt.figure(figsize=(10, 6))
plt.boxplot(df[col].values, patch_artist=True, vert=False)
plt.title(f"Box Plot of {col}")
plt.xlabel("Value")
plt.ylabel(col)
plt.grid(True)

# Outliers
outliers = df[(df[col] < q1 - 1.5 * iqr) | (df[col] > q3 + 1.5 * iqr)]
plt.scatter(outliers[col], [1] * len(outliers), marker='x',
color='red', label='Outliers')

# Median and quartiles
median = df[col].median()
```

```
plt.axvline(median, color='blue', label='Median')
plt.axvline(q1, color='green', label='Q1')
plt.axvline(q3, color='green', label='Q3')

plt.legend()

plt.show()
```