```
import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
data·=·pd.read_csv('/content/drive/MyDrive/AB_NYC_2019.csv')
```

```
data
```

	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
...	...	...	...	...	...	...	.
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.6785
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.7018
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.8147
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.7575
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.7640

48895 rows × 16 columns

```
#Removing duplicate
data.drop_duplicates(inplace=True)
```

```
#·checking·the·number·of·missing·values·in·each·column
print(data.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
```

```
# impute missing values in the name and host_name columns
data[['name', 'host_name']] = data[['name', 'host_name']].fillna('Unknown')
```

```
print(data.isnull().sum())
```

```
id                                    0
name                                  0
host_id                               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
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

```
# impute missing values in the last_review and reviews_per_month columns
data['last_review'] = pd.to_datetime(data['last_review'])
median_last_review = data['last_review'].median()
data['last_review'] = data['last_review'].fillna(median_last_review)

median_reviews_per_month = data['reviews_per_month'].median()
data['reviews_per_month'] = data['reviews_per_month'].fillna(median_reviews_per_month)
```

```
print(data.isnull().sum())
```

```
id                                0
name                              0
host_id                           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
last_review                       0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

```
# removing 'id', 'host_id', and 'last_review' unnecessary columns
data.drop(['id', 'host_id', 'last_review'], axis=1, inplace=True)
```

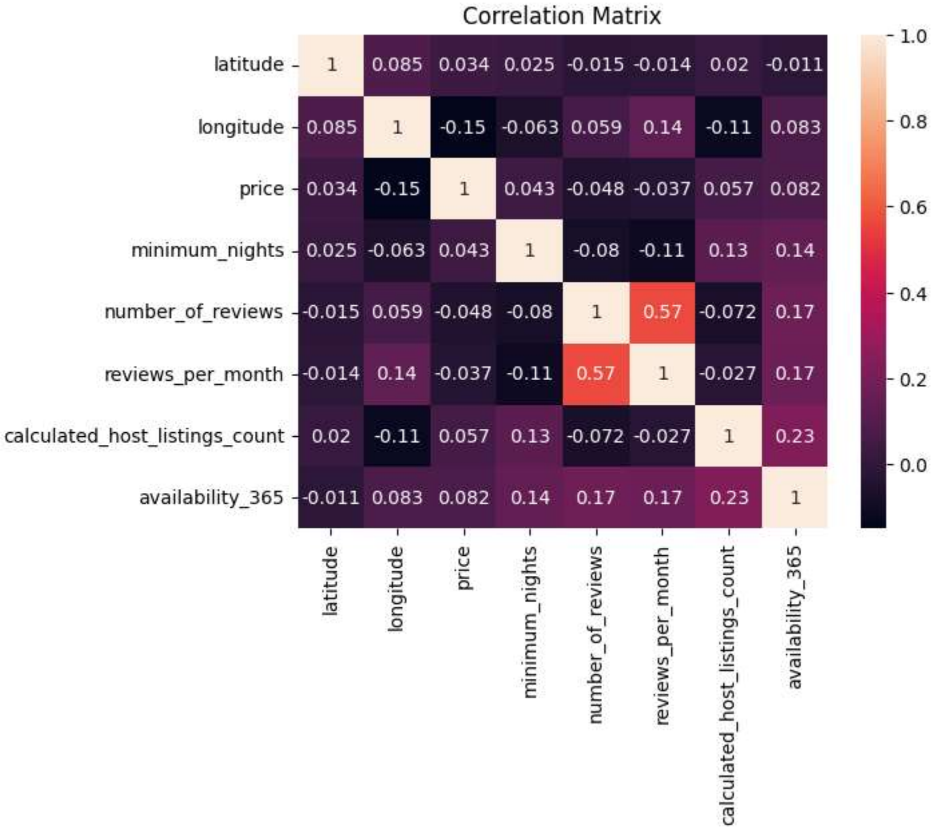
```
data.describe()
```

	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_mont
count	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	48895.0000
mean	40.728949	-73.952170	152.720687	7.029962	23.274466	1.2389
std	0.054530	0.046157	240.154170	20.510550	44.550582	1.5208
min	40.499790	-74.244420	0.000000	1.000000	0.000000	0.0100

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
```

```
#·Correlation·analysis
corr_matrix·=·data.corr(numeric_only=True)
corr_matrix·=·data.corr()
sns.heatmap(corr_matrix,·annot=True)
plt.title("Correlation·Matrix")
plt.show()
```

```
<ipython-input-36-5fc9a567e083>:3: FutureWarning: The default value of numeric_only in DataFrame.c
  corr_matrix = data.corr()
```



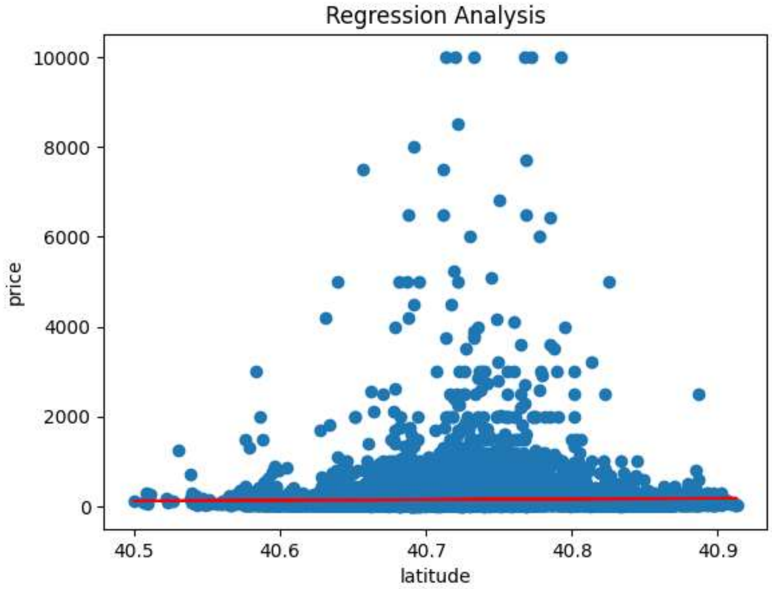
```
#·Calculate·Pearson's·correlation·coefficient·and·p-value·between·two·variables
corr_coeff,·p_value·=·pearsonr(data['price'],·data['availability_365'])
print("Pearson's·correlation·coefficient:",·corr_coeff)
print("p-value:",·p_value)
```

```
Pearson's correlation coefficient: 0.08182882742168793
p-value: 2.056743270408171e-73
```

```
corr_coeff, p_value = pearsonr(data['latitude'], data['longitude'])
print("Pearson's correlation coefficient:", corr_coeff)
print("p-value:", p_value)
```

```
    Pearson's correlation coefficient: 0.08478836838914451
    p-value: 1.0614406457426472e-78
```

```
# Regression analysis
X = data['latitude'].values.reshape(-1, 1)
y = data['price'].values.reshape(-1, 1)
regressor = LinearRegression()
regressor.fit(X, y)
y_pred = regressor.predict(X)
plt.scatter(X, y)
plt.plot(X, y_pred, color='red')
plt.xlabel('latitude')
plt.ylabel('price')
plt.title('Regression Analysis')
plt.show()
```



```
print("Intercept:", regressor.intercept_)
print("Slope:", regressor.coef_)
```

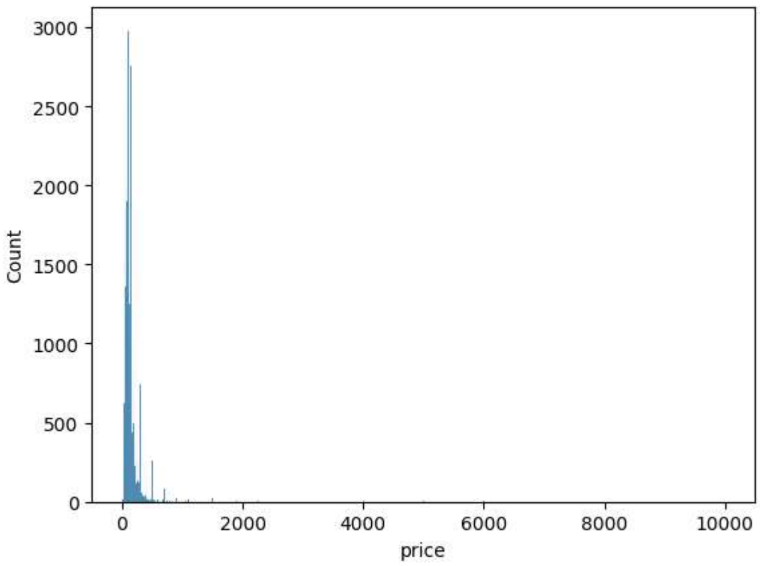
```
    Intercept: [-5934.96204863]
    Slope: [[149.46820144]]
```

```
# Inferential analysis
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())
```

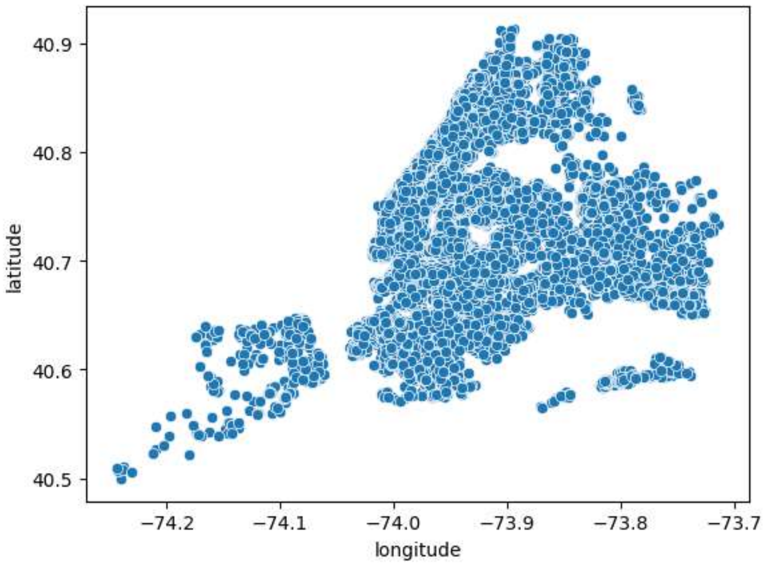
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     56.38
Date:                Thu, 04 May 2023   Prob (F-statistic):           6.07e-14
Time:                        13:50:27   Log-Likelihood:            -3.3736e+05
No. Observations:               48895   AIC:                         6.747e+05
Df Residuals:                   48893   BIC:                         6.747e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -5934.9620    810.744     -7.320      0.000   -7524.031   -4345.893
x1           149.4682     19.906      7.509      0.000     110.453     188.484
==============================================================================
Omnibus:                   105137.688   Durbin-Watson:                   1.836
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        704029629.724
Skew:                          19.142   Prob(JB):                         0.00
Kurtosis:                     589.605   Cond. No.                     3.04e+04
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.04e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

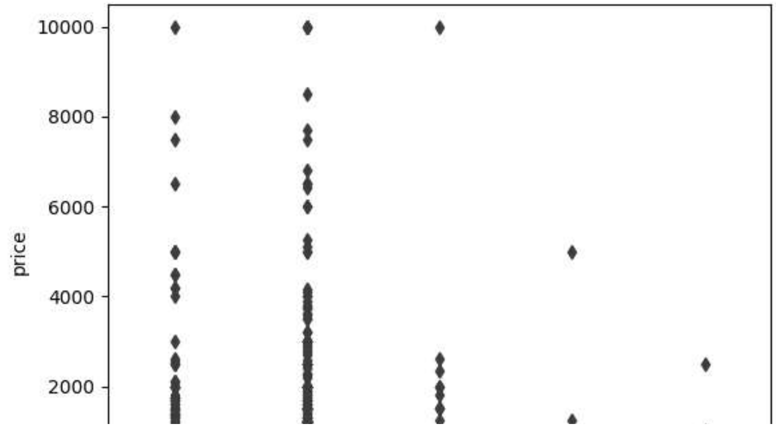
```
import matplotlib.pyplot as plt
import seaborn as sns

# histogram of price
sns.histplot(data['price'], kde=False)
plt.show()
```



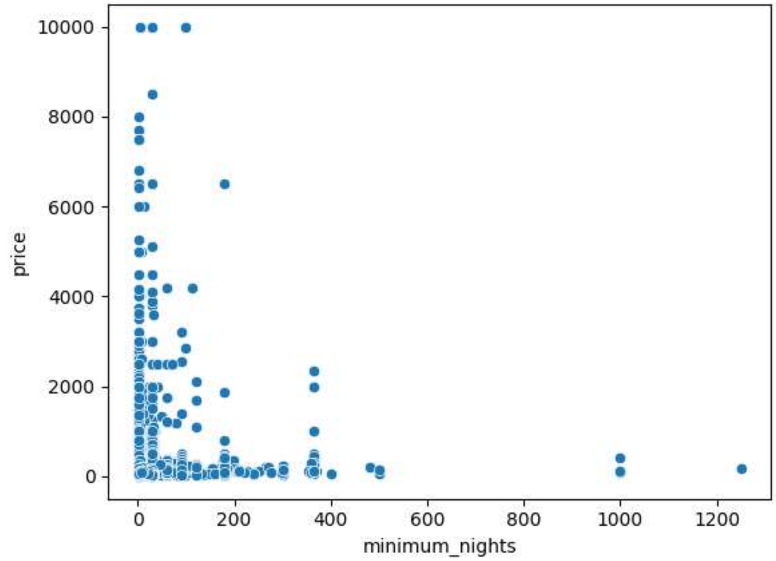
```
# scatterplot of latitude vs. longitude
sns.scatterplot(x='longitude', y='latitude', data=data)
plt.show()
```



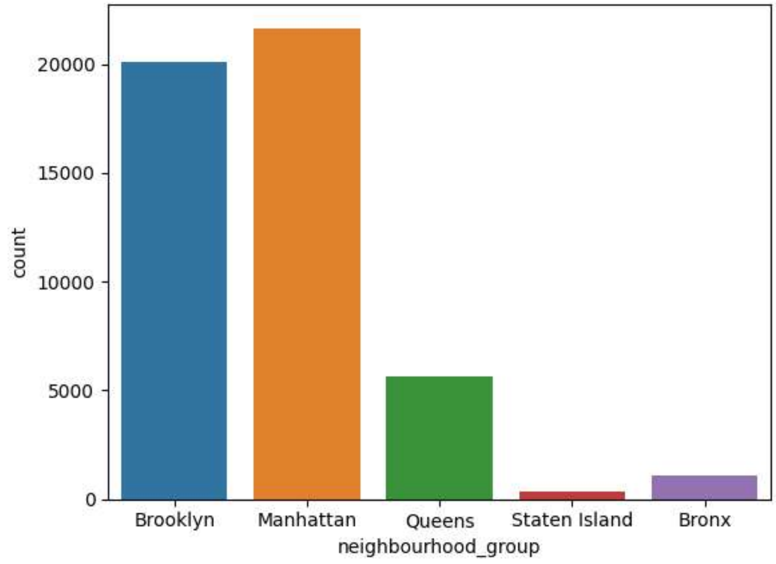
```
# boxplot of price vs. neighbourhood_group
sns.boxplot(x='neighbourhood_group', y='price', data=data)
plt.show()
```



```
import·seaborn·as·sns
import·matplotlib.pyplot·as·plt

sns.scatterplot(x='minimum_nights',·y='price',·data=data)
plt.show()
```

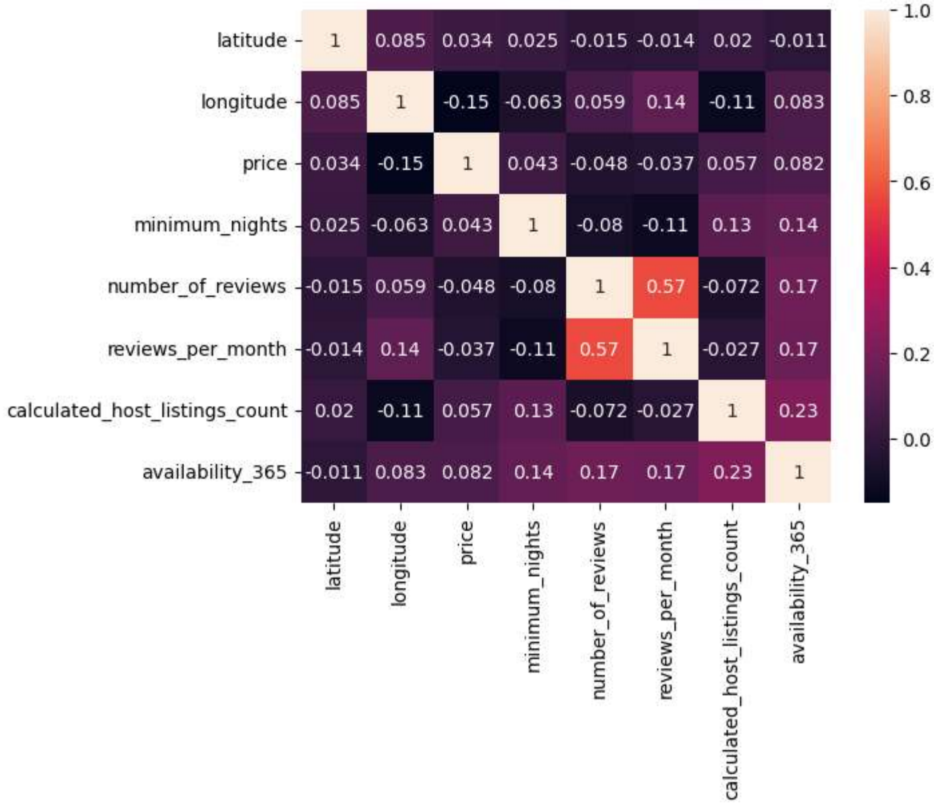


```
sns.countplot(x='neighbourhood_group',·data=data)
plt.show()
```



```
sns.heatmap(data.corr(), annot=True)
plt.show()
```

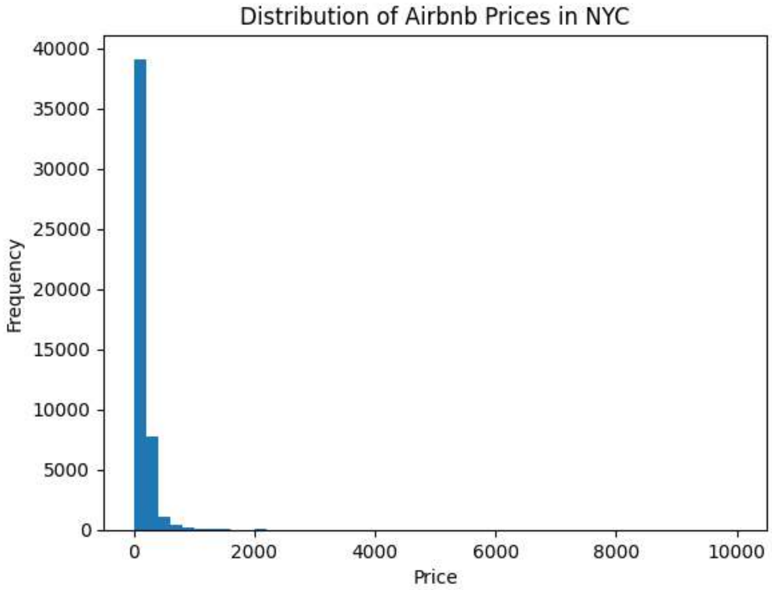
```
    <ipython-input-25-9fa67090a602>:1: FutureWarning: The default value of numeric_only in DataFrame.cc
      sns.heatmap(data.corr(), annot=True)
```



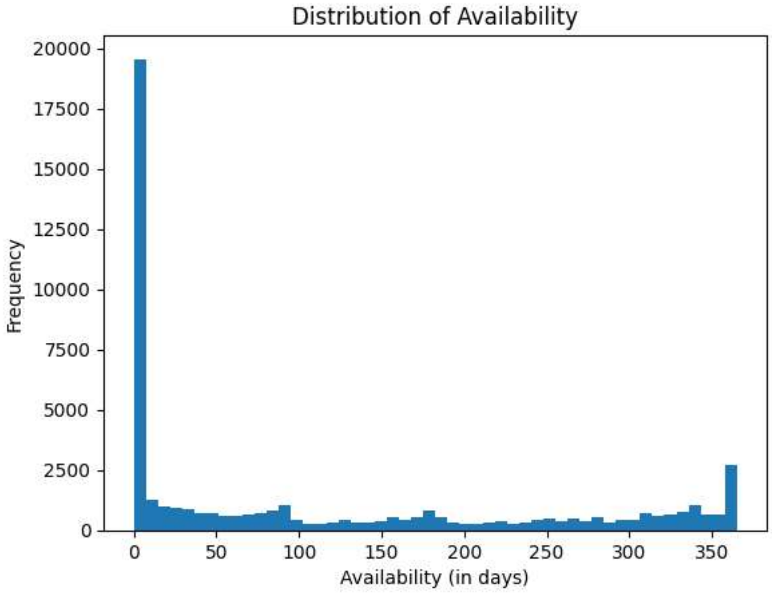
```
avg_price = data.groupby('neighbourhood_group')['price'].mean().reset_index()
```

```
import·seaborn·as·sns
import·matplotlib.pyplot·as·plt

#·Create·bar·plot·of·average·price·by·neighbourhood·group
sns.barplot(data=avg_price,·x='neighbourhood_group',·y='price')
plt.title('Average·Price·by·Neighbourhood·Group')
plt.show()
```

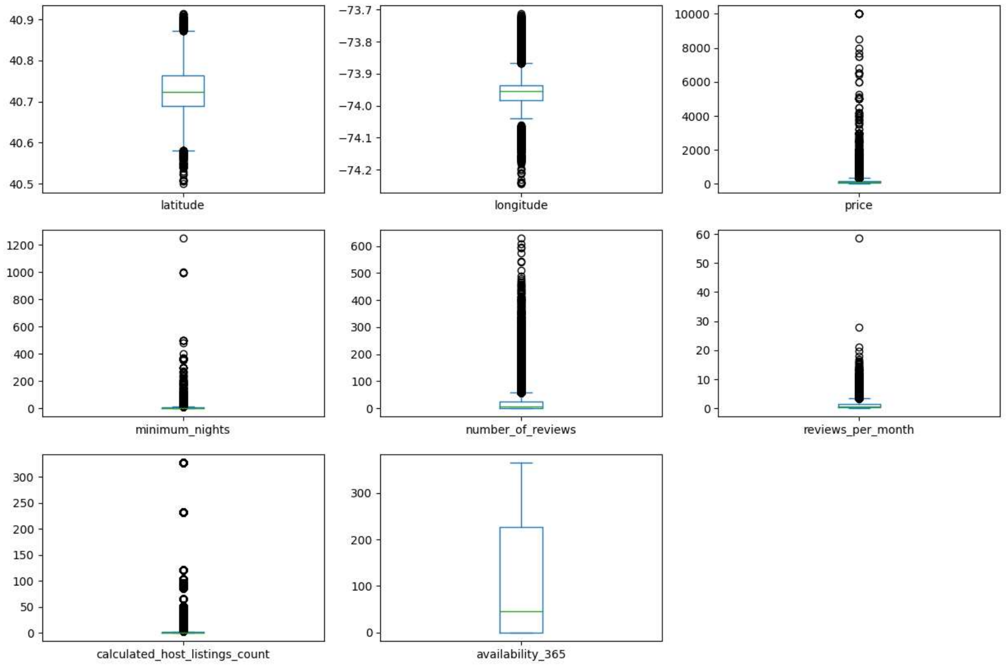
```
import matplotlib.pyplot as plt

plt.hist(data['price'], bins=50)
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.title('Distribution of Airbnb Prices in NYC')
plt.show()
```



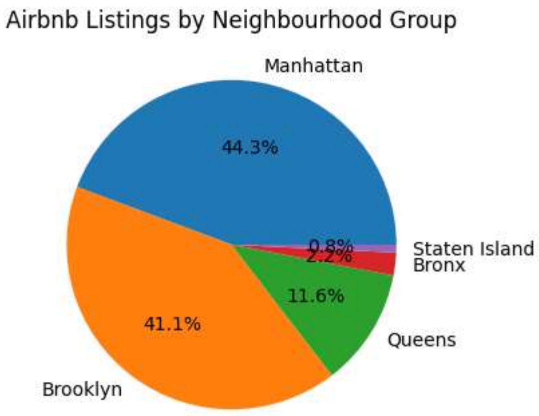
```
import matplotlib.pyplot as plt

plt.hist(data['availability_365'], bins=50)
plt.xlabel('Availability (in days)')
plt.ylabel('Frequency')
plt.title('Distribution of Availability')
plt.show()
```



```
#·Create·box·plots·for·each·feature
data.plot(kind='box',·subplots=True,·layout=(3,3),·figsize=(15,10))
plt.show()
```

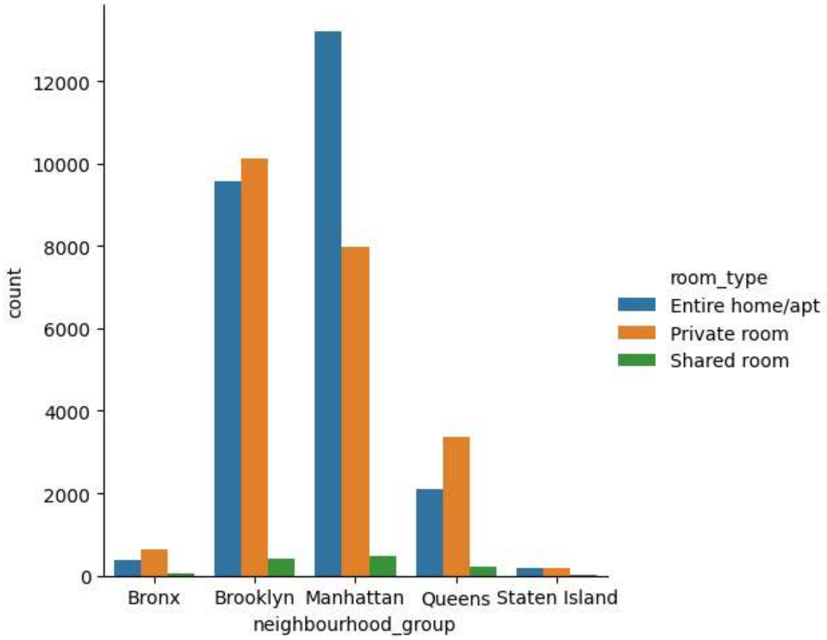


```
# Count the number of listings by neighbourhood group
counts = data['neighbourhood_group'].value_counts()

# Create a pie chart
plt.figure(figsize=(6,4))
plt.pie(counts.values, labels=counts.index, autopct='%1.1f%%')
plt.title('Airbnb Listings by Neighbourhood Group')
plt.show()
```

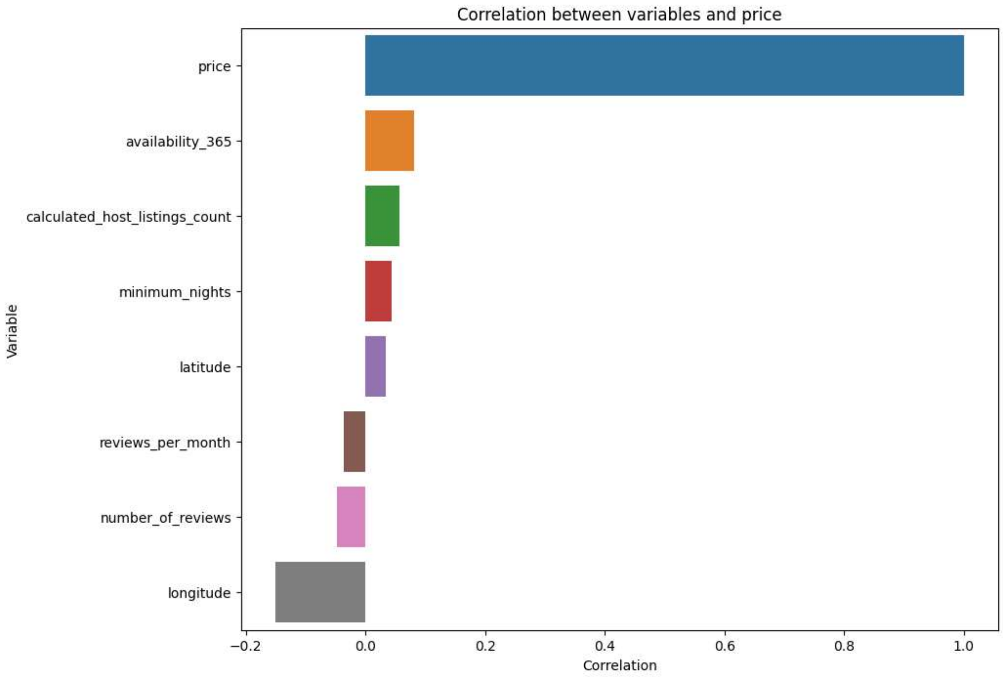


```
#Analyze the most popular types of listings and the neighborhoods they are located in
grouped_data = data.groupby(['neighbourhood_group', 'room_type']).size().reset_index(name='count')

# Plot the data
sns.catplot(x='neighbourhood_group', y='count', hue='room_type', data=grouped_data, kind='bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f096406eda0>
```



```
#Identify the most important amenities that guests are looking for in an Airbnb listing in New York City

# Select relevant columns
selected_cols = ['price', 'latitude', 'longitude', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_coun
data = data[selected_cols]

# Calculate correlations between variables and price
corr_matrix = data.corr()
corr_with_price = corr_matrix['price'].sort_values(ascending=False)

# Plot the correlations
plt.figure(figsize=(10,8))
sns.barplot(x=corr_with_price.values, y=corr_with_price.index, orient='h')
plt.title('Correlation between variables and price')
plt.xlabel('Correlation')
plt.ylabel('Variable')
plt.show()
```



Correlation between variables and price

✓ 0s completed at 9:03 PM