```
import numpy as np
import pandas as pd

from google.colab import files

## downloaded csv file from url and importing it into notebook
import_data = files.upload()
```

```
import io

## using pandas to load and process csv file
city_data = pd.read_csv(io.BytesIO(import_data['AB_NYC_2019.csv'])) ## loading informa
```

```
## Problem 1 - Cleaning Data

city_data["name"].fillna("No Name", inplace = True)
city_data["host_name"].fillna("No Host Name", inplace = True)
city_data["last_review"].fillna("No Last Review", inplace = True)
city_data["reviews_per_month"].fillna(0, inplace = True)

## Fixing the null values within columns name, host_name, last_review, and reviews_per
```

```
df = pd.DataFrame(city_data)
df["count"] = 1 ## setting a count column initialized to 1, will be useful later for c
```

```
## Problem 2 Part A
```

```
copy_df = df

v = copy_df.neighbourhood.value_counts() ## getting the count of each neighborhood

gl = copy_df[copy_df.neighbourhood.isin(v.index[v.gt(5)])] ## only keeping track of ne

grouped_data = gl.groupby('neighbourhood')["price"].mean() ## mean of the neighbourhoc
grouped_data = grouped_data.sort_values(ascending=True)

print(grouped_data)
```

```
neighbourhood
Bull's Head           47.333333
Hunts Point           50.500000
Tremont               51.545455
Soundview             53.466667
Bronxdale             57.105263
                         ...
Flatiron District    341.925000
Battery Park City    367.557143
Riverdale            442.090909
Sea Gate             487.857143
Tribeca              490.638418
Name: price, Length: 190, dtype: float64
```

```
## problem2 part B
import matplotlib.pyplot as plt

grouped_data2 = df.groupby('neighbourhood_group') ## grouping the neighborhood_group t

grouped_data2 = grouped_data2.agg({'price': 'mean'}) ## average out all the prices of


fig = plt.figure()
x_axis = ["Bronx", "Brooklyn", "Manhattan", "Queens", "Staten Island"] ## setting x-va
y_axis = grouped_data2["price"] ## setting y-var

ax = fig.add_axes([0,0,1,1]) ## showing the resutls
plt.pie(y_axis, labels = x_axis, startangle = 90)
plt.show()
```

```
## Problem 3
from scipy.stats import pearsonr
## what two variables are needed

corr, _ = pearsonr(df["reviews_per_month"], df["number_of_reviews"])
print("Correlation: " + str(corr) + " reviews_per_month v number_of_reviews")

corr, _ = pearsonr(df["availability_365"], df["calculated_host_listings_count"])
print("Correlation: " + str(corr)+ " availability_365 v calculated_host_listings_count

corr, _ = pearsonr(df["availability_365"], df["reviews_per_month"])
print("Correlation: " + str(corr) + " availability_365 v reviews_per_month")

corr, _ = pearsonr(df["availability_365"], df["number_of_reviews"])
print("Correlation: " + str(corr) + " availability_365 v number_of_reviews")

corr, _ = pearsonr(df["number_of_reviews"],df["calculated_host_listings_count"])
print("Correlation: " + str(corr) + " number_of_reviews v calculated_host_listings_cou

corr, _ = pearsonr(df["reviews_per_month"], df["calculated_host_listings_count"])
print("Correlation: " + str(corr) + " reviews_per_month v calculated_host_listings_cou

## I looked at correlation coefficient of the combination of all 4 of these colums and

## NEED TO MAKE HEAT-MAP
```

```
Correlation: 0.5894072970835142 reviews_per_month v number_of_reviews
Correlation: 0.22570137219113479 availability_365 v calculated_host_listings_coun
Correlation: 0.16373167028253466 availability_365 v reviews_per_month
Correlation: 0.172027581462931 availability_365 v number_of_reviews
Correlation: -0.07237606054177578 number_of_reviews v calculated_host_listings_c
Correlation: -0.04731205587843853 reviews_per_month v calculated_host_listings_c
```

```
## problem 4a

color_dict = {'Bronx':'red', 'Brooklyn':'green', 'Manhattan':'orange', 'Queens': 'blue

for index, row in df.iterrows():
    plt.scatter(x=row["longitude"], y=row["latitude"], c = color_dict[row["neighbourho

## for index, row in df.iterrows():
plt.show()
```

```
## problem 4b

test = df.head(10000)

for index, row in test.iterrows():
    if (row["price"] < 200):
        plt.scatter(x=row["longitude"], y=row["latitude"], c = "red")
    elif (row["price"] < 400):
        plt.scatter(x=row["longitude"], y=row["latitude"], c = "blue")
    elif (row["price"] < 600):
        plt.scatter(x=row["longitude"], y=row["latitude"], c = "green")
    elif (row["price"] < 800):
        plt.scatter(x=row["longitude"], y=row["latitude"], c = "yellow")
    elif (row["price"] < 1000):
        plt.scatter(x=row["longitude"], y=row["latitude"], c = "blue")

plt.show()
```

```
## Problem 5
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

text = df.name.tolist() ## convert all the description words into a list
text = ' '.join(text).lower() ## make lowercase

wordcloud = WordCloud(stopwords = STOPWORDS,
                      collocations=True).generate(text) ## create wordcloud from list
plt.imshow(wordcloud) ## plot the wordcloud
plt.axis('off') ## no axis
```

```
plt.show()

## Results show that private, central park, beautiful, spacious are more of the commor
```

```
## Problem 6

## First we will analyze the Financial District neighborhood
Financial_District = df[df.neighbourhood == "Financial District"]


## "number_of_reviews"
## "availability_365"
## "price"
```

```
fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(Financial_District["calculated_host_listings_count"], Financial_District['
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v availability_365")

corr, _ = pearsonr(Financial_District["calculated_host_listings_count"], Financial_Dis
print("Correlation: " + str(corr) + "   calculated_host_listings_count v availability_
```

```
fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(Financial_District["calculated_host_listings_count"], Financial_District['
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v price")

corr, _ = pearsonr(Financial_District["calculated_host_listings_count"], Financial_Dis
print("Correlation: " + str(corr) + "     calculated_host_listings_count v price")
```

```
fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(Financial_District["calculated_host_listings_count"], Financial_District['
```

```
plt.scatter(Financial_District["calculated_host_listings_count"], Financial_District[
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v availability")

corr, _ = pearsonr(Financial_District["calculated_host_listings_count"], Financial_Dis
print("Correlation: " + str(corr) + "     calculated_host_listings_count v minimum_nig

## for the Financial District, there is a moderate correlation between host listings a
```

```
PortMorris = df[df.neighbourhood == "Port Morris"]


fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(PortMorris["calculated_host_listings_count"], PortMorris["availability_365
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v availability_365")

corr, _ = pearsonr(PortMorris["calculated_host_listings_count"], PortMorris["availabil
print("Correlation: " + str(corr) + "   calculated_host_listings_count v availability_


fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(PortMorris["calculated_host_listings_count"], PortMorris["price"])
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v price")

corr, _ = pearsonr(PortMorris["calculated_host_listings_count"], PortMorris["price"])
print("Correlation: " + str(corr) + "     calculated_host_listings_count v price")



fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(PortMorris["calculated_host_listings_count"], PortMorris["minimum_nights"]
plt.xlabel("calculated_host_listings_count")
```

```
plt.ylabel("availability")
plt.title("calculated_host_listings_count v minimum_nights")

corr, _ = pearsonr(PortMorris["calculated_host_listings_count"], PortMorris["minimum_r
print("Correlation: " + str(corr) + "     calculated_host_listings_count v minimum_nig

## For the neighborhood Port Morris, there is a negative tight correlation between lis
```

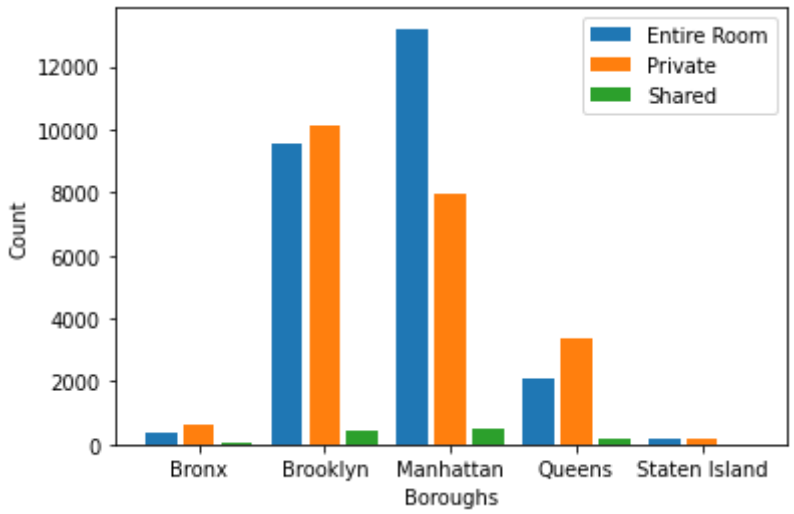
```
## Analyzing Ridgewood

Ridgewood = df[df.neighbourhood == "Ridgewood"]


## "number_of_reviews"
## "availability_365"
## "price"



fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(Ridgewood["calculated_host_listings_count"], Ridgewood["availability_365"]
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v availability")

corr, _ = pearsonr(Ridgewood["calculated_host_listings_count"], Ridgewood["availabilit
print("Correlation: " + str(corr) + "   calculated_host_listings_count v availability_


fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(Ridgewood["calculated_host_listings_count"], Ridgewood["price"])
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v price")

corr, _ = pearsonr(Ridgewood["calculated_host_listings_count"], Ridgewood["price"])
print("Correlation: " + str(corr) + "     calculated_host_listings_count v price")
```

```
fig, ax = plt.subplots(figsize=(10, 6))
plt.scatter(Ridgewood["calculated_host_listings_count"], Ridgewood["minimum_nights"])
plt.xlabel("calculated_host_listings_count")
plt.ylabel("availability")
plt.title("calculated_host_listings_count v minimum_nights")

corr, _ = pearsonr(Ridgewood["calculated_host_listings_count"], Ridgewood["minimum_nig
print("Correlation: " + str(corr) + "     calculated_host_listings_count v minimum_nig

## for the Ridgewood community, there is a moderate correlation between host listings
```

```
## Part 7a

## I want to create a graph between the price range for private room, vs entire home


df_copy = df

df_copy2 = df_copy.groupby(["neighbourhood_group", "room_type"]) ## group between thes

grouped_sum = df_copy2.price.mean() ## getting mean

x = np.arange(5)
entire = [grouped_sum["Bronx"][0], grouped_sum["Brooklyn"][0], grouped_sum["Manhattan'
private = [grouped_sum["Bronx"][1], grouped_sum["Brooklyn"][1], grouped_sum["Manhattar
shared = [grouped_sum["Bronx"][2], grouped_sum["Brooklyn"][2], grouped_sum["Manhattan'
width = 0.25

## making bar chart

plt.bar(x-0.3, entire, width)
plt.bar(x, private, width)
plt.bar(x+0.3, shared, width)
```

```
plt.xticks(x, ["Bronx", "Brooklyn", "Manhattan", "Queens", "Staten Island"])
plt.xlabel("Boroughs")
plt.ylabel("Price")
plt.legend(["Entire Room", "Private", "Shared"])
plt.show()


## Entire Room is most expensive, then private, then shared, which ultimately does mak
```

```
## Part 7b

## find the total representation of the 5 boroughs in the data

df_copy = df

df_copy2 = df_copy.groupby(["neighbourhood_group", "room_type"]) ## group by these var

grouped_sum = df_copy2.agg({'count': 'sum'}) ## find the sum of each group

x = np.arange(5)
entire = [grouped_sum["count"][0], grouped_sum["count"][3],grouped_sum["count"][6], gr
private = [grouped_sum["count"][1], grouped_sum["count"][4],grouped_sum["count"][7], g
shared = [grouped_sum["count"][2], grouped_sum["count"][5],grouped_sum["count"][8], gr
width = 0.25

plt.bar(x-0.3, entire, width)
plt.bar(x, private, width)
plt.bar(x+0.3, shared, width)

plt.xticks(x, ["Bronx", "Brooklyn", "Manhattan", "Queens", "Staten Island"])
plt.xlabel("Boroughs")
plt.ylabel("Count")
plt.legend(["Entire Room", "Private", "Shared"])
plt.show()
```



✓ 2s completed at 11:30 PM