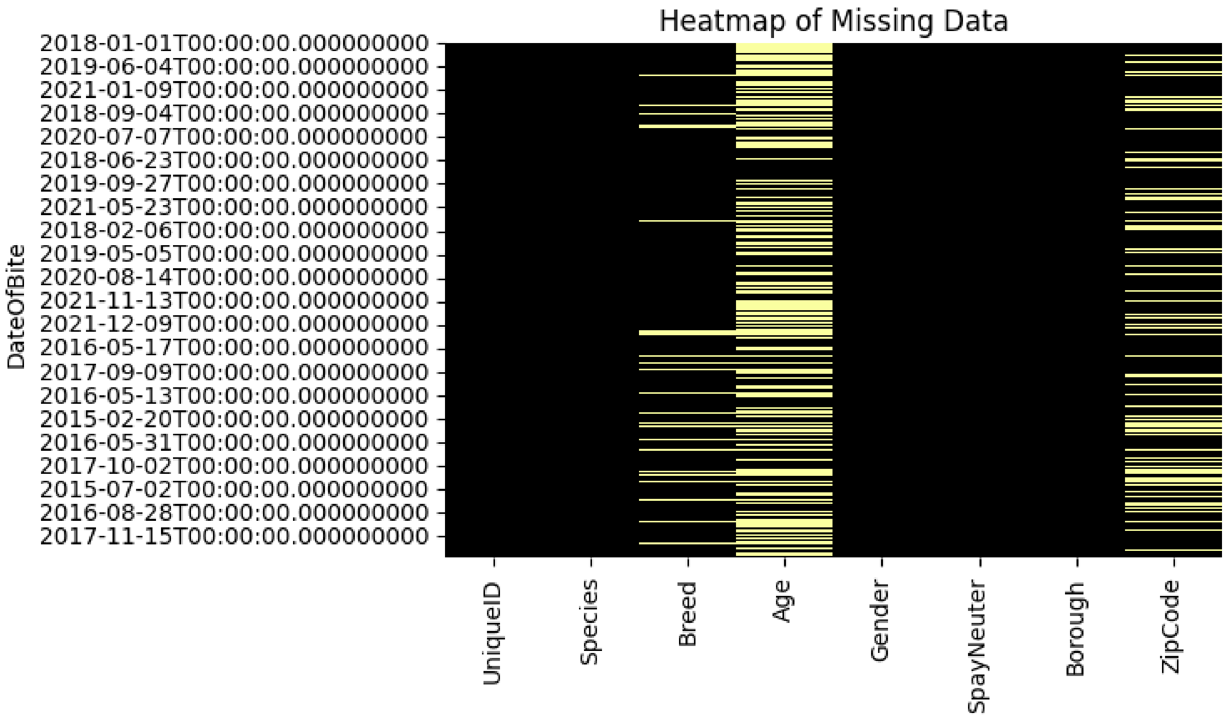
Homework 5 Notebook

```
## Dog Bites in New York City
```

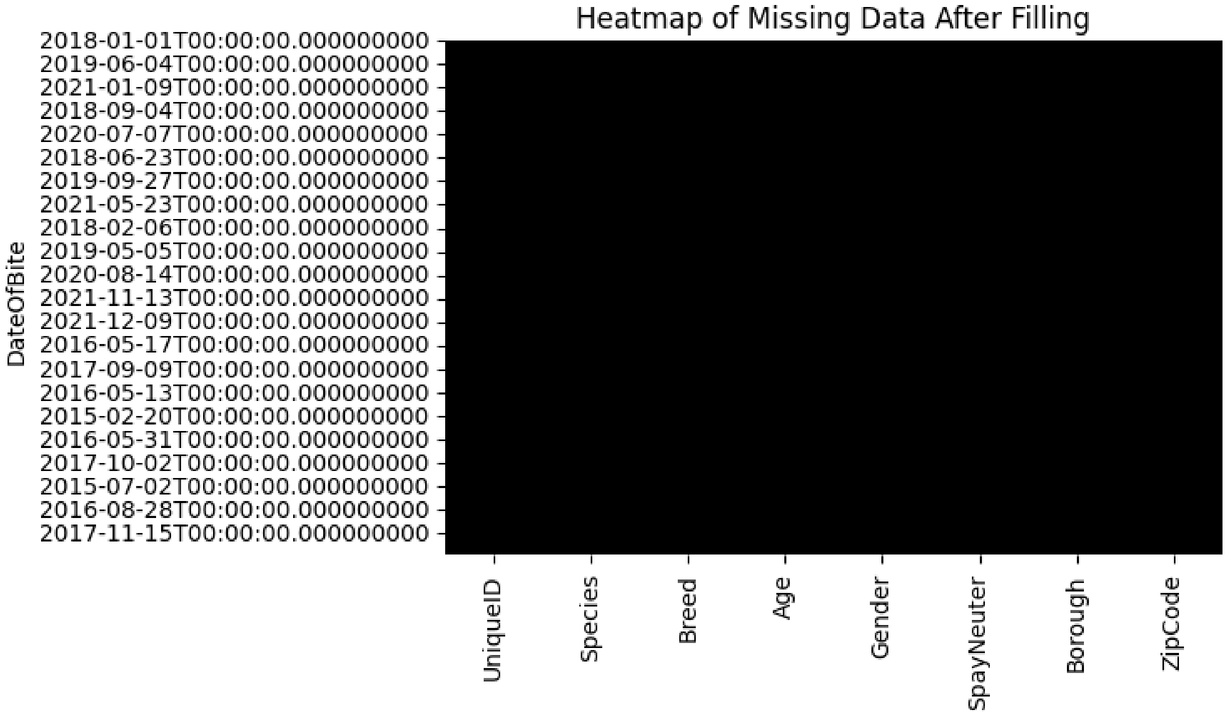
```
### Data Exploration, Cleaning, and Preparation
```

```python
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
import plotly.express as px
import json
import plotly.graph_objects as go

# for formatting
headline1 = "\n----------|"
headline2 = "|----------\n"

data = pd.read_csv("Dog_Bites_Data.csv")
data['DateOfBite'] = pd.to_datetime(data['DateOfBite'])
data.set_index('DateOfBite', inplace=True)

# heatmap of missing values
plt.figure(figsize=(6,4))
sns.heatmap(data.isnull(), cbar=False, cmap="inferno")
plt.title("Heatmap of Missing Data")
plt.show()
```



```python
# fixing datatypes & filling missing values

# filling missing breeds with 'UNKNOWN'
data['Breed'] = data['Breed'].fillna('UNKNOWN')

# age to numeric
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')

# fill missing age values with their median
data['Age'] = data['Age'].fillna(data['Age'].median())

# zipcode to numeric
data['ZipCode'] = pd.to_numeric(data['ZipCode'], errors='coerce')

# fill missing zip codes with 'UNKNOWN' to mirror Breed
data['ZipCode'] = data['ZipCode'].fillna('UNKNOWN')

# correcting unknown 'mixed' breeds into unknown to only keep known mixes
corrections = {
   "Mixed/Other": "UNKNOWN",
   "MIXED BREED": "UNKNOWN",
   "MIXED": "UNKNOWN"
}
data['Breed'] = data['Breed'].replace(corrections)
```

```
# heatmap of missing values (after filling)
plt.figure(figsize=(6,4))
sns.heatmap(data.isnull(), cbar=False, cmap="inferno")
plt.title("Heatmap of Missing Data After Filling")
plt.show()
```



```
# summary statistics of dataset

# to print datatypes
print(f"{headline1} Datatypes {headline2}{data.dtypes}")

# # of rows
print(f"{headline1} Rows {headline2}{len(data.axes[0])}")

# # of columns
print(f"{headline1} Columns {headline2}{len(data.axes[1])}")

# # of missing values
print(f"{headline1} Missing Data Count {headline2}{data.isnull().sum()}")
```

```
# for loop to go through each column and display numerical statistics or
categorical statistics
for column in data.columns:
    if data[column].dtype == 'int64':
        print(f"{headline1} {column} {headline2}")
        print(f"median: {data[column].median()}") # median added to
.describe since it was specified in project instructions
        print(data[column].describe())
    else:
        print(f"{headline1} {column} {headline2}")
        print(data[column].value_counts())
```

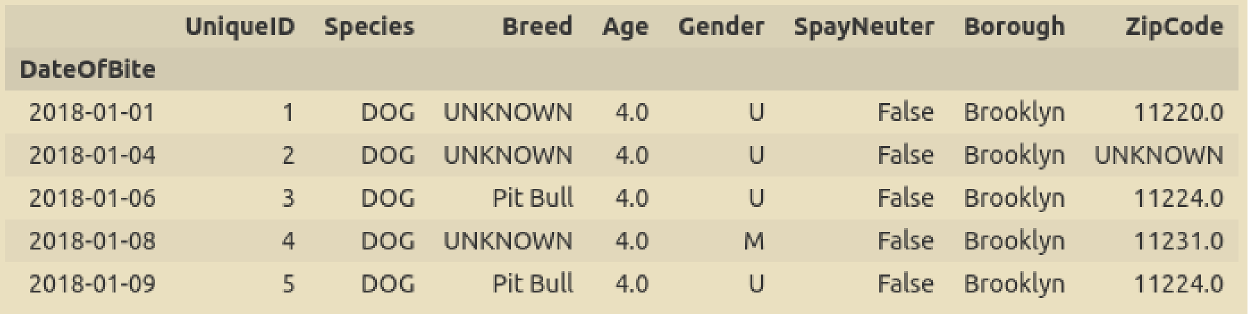
```
----------| Datatypes |----------
UniqueID        int64
Species        object
Breed          object
Age           float64
Gender         object
SpayNeuter       bool
Borough        object
ZipCode        object
dtype: object

----------| Rows |----------
22663

----------| Columns |----------
8

----------| Missing Data Count |----------
UniqueID      0
Species       0
Breed         0
Age           0
Gender        0
SpayNeuter    0
Borough       0
ZipCode       0
dtype: int64

----------| UniqueID |----------
```

```
median: 5666.0
count    22663.000000
mean      5715.036668
std       3354.278369
min          1.000000
25%       2833.500000
50%       5666.000000
75%       8499.000000
max      12383.000000
Name: UniqueID, dtype: float64

----------| Species |----------

Species
DOG    22663
Name: count, dtype: int64

----------| Breed |----------

Breed
UNKNOWN                              5865
Pit Bull                             4004
Shih Tzu                              731
Chihuahua                             646
German Shepherd                       622
                                     ... 
GOLDEN RETRIEVER / POODLE MIX           1
CHOCOLATE LAB & AMERICAN STAFFORD       1
LHASA APSO / SHIH TZU MIX               1
AMERICAN BULL MIX                       1
CHIHUAHUA / YORKIE MIX                  1
Name: count, Length: 1648, dtype: int64

----------| Age |----------

Age
4.0     13611
2.0      1624
3.0      1504
1.0      1365
5.0      1040
6.0       795
7.0       655
8.0       569
9.0       375
10.0      361
11.0      242
```

```
12.0      188
13.0      140
14.0       75
15.0       51
16.0       22
17.0        9
2.5         7
1.5         4
19.0        3
3.5         3
20.0        2
4.5         2
1.6         2
4.6         2
1.3         2
41.0        1
0.6         1
21.0        1
15.5        1
0.2         1
1.8         1
2.6         1
3.6         1
6.5         1
10.5        1
Name: count, dtype: int64

----------| Gender |----------

Gender
U    10535
M     8739
F     3389
Name: count, dtype: int64

----------| SpayNeuter |----------

SpayNeuter
False    16787
True      5876
Name: count, dtype: int64

----------| Borough |----------

Borough
Queens           5773
Manhattan        5270
```

```
Brooklyn         4985
Bronx            3782
Staten Island    1872
Other             981
Name: count, dtype: int64

----------| ZipCode |----------

ZipCode
UNKNOWN    5859
10029.0     369
11208.0     261
11368.0     226
10065.0     221
           ...
11772.0       1
6460.0        1
2633.0        1
18301.0       1
7036.0        1
Name: count, Length: 513, dtype: int64
```

```
data.head()
```

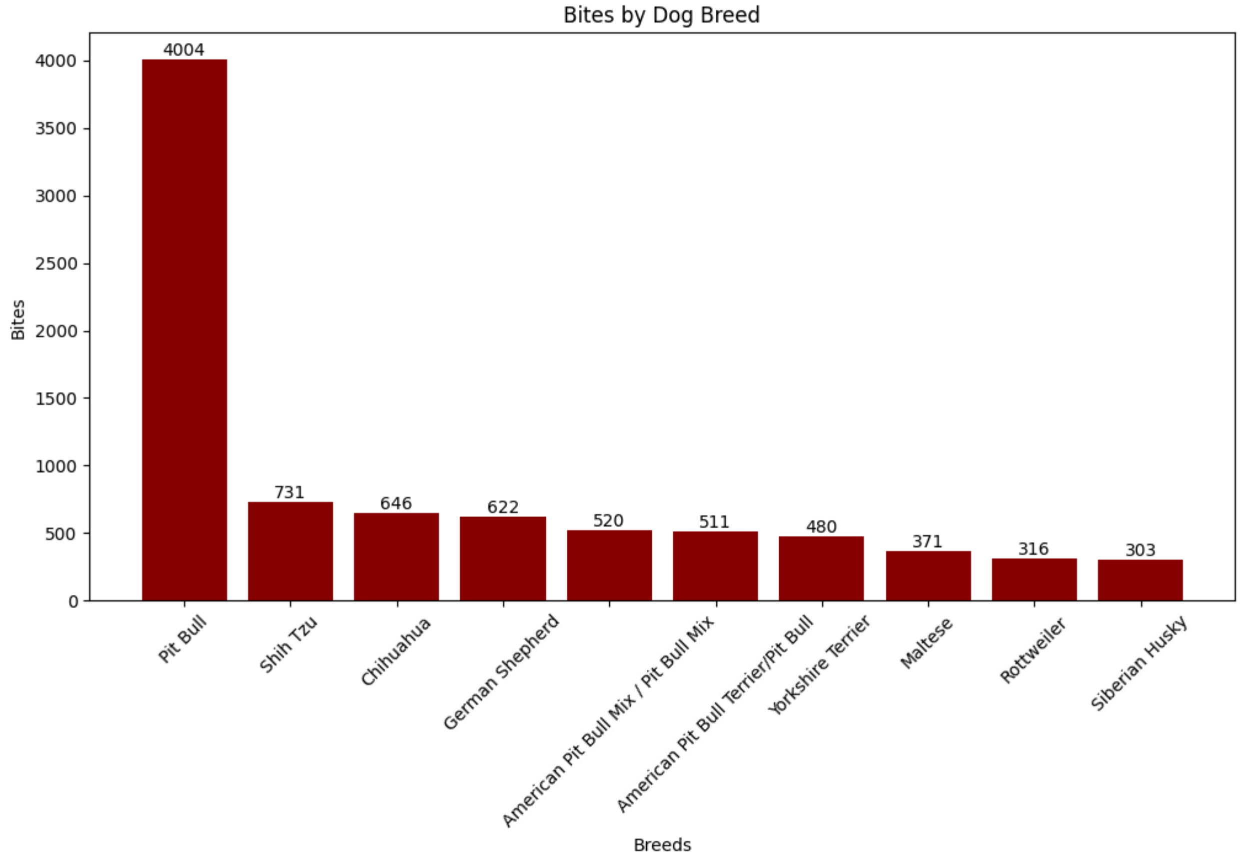


	UniqueID	Species	Breed	Age	Gender	SpayNeuter	Borough	ZipCode
DateOfBite								
2018-01-01	1	DOG	UNKNOWN	4.0	U	False	Brooklyn	11220.0
2018-01-04	2	DOG	UNKNOWN	4.0	U	False	Brooklyn	UNKNOWN
2018-01-06	3	DOG	Pit Bull	4.0	U	False	Brooklyn	11224.0
2018-01-08	4	DOG	UNKNOWN	4.0	M	False	Brooklyn	11231.0
2018-01-09	5	DOG	Pit Bull	4.0	U	False	Brooklyn	11224.0

```
### Data Analysis & Maps
```

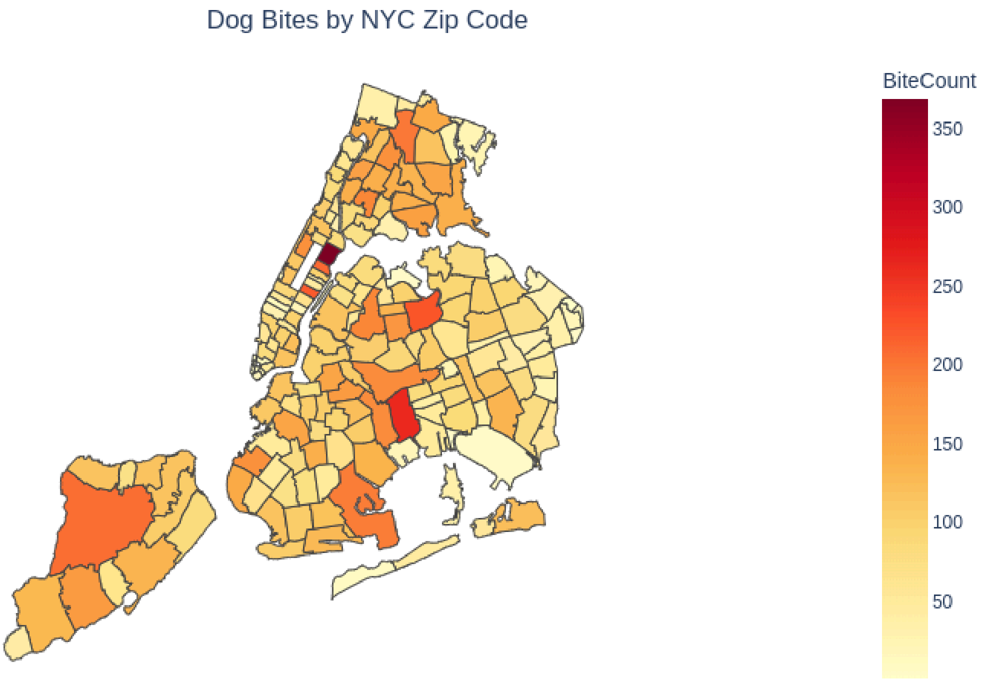
```
# getting the top ten breeds with most bites from dataset
grouped_breeds = data['Breed'].value_counts()    # grouping
grouped_breeds = grouped_breeds.iloc[1:]         # removing first row of
'UNKNOWN' breed
top_ten_breeds = grouped_breeds.head(10)         # keeping only the top ten

# converting to dataframe
```

```
top_ten_breeds = top_ten_breeds.reset_index()
top_ten_breeds.columns = ['Breed', 'Bites']

# plotting bar chart
plt.figure(figsize=(12,6))

# bar labels
bars = plt.bar(top_ten_breeds['Breed'], top_ten_breeds['Bites'],
color='darkred')
for bar in bars:
   plt.text(
   bar.get_x() + bar.get_width() / 2, # X position: center of each bar
   bar.get_height(), # Y position: height of each bar
   bar.get_height(), # Text to display (the height of the bar)
   ha='center', # Horizontal alignment
   va='bottom' # Vertical alignment
 )

plt.title("Bites by Dog Breed")
plt.xticks(rotation=45)
plt.xlabel('Breeds')
plt.ylabel('Bites')
plt.show()
```

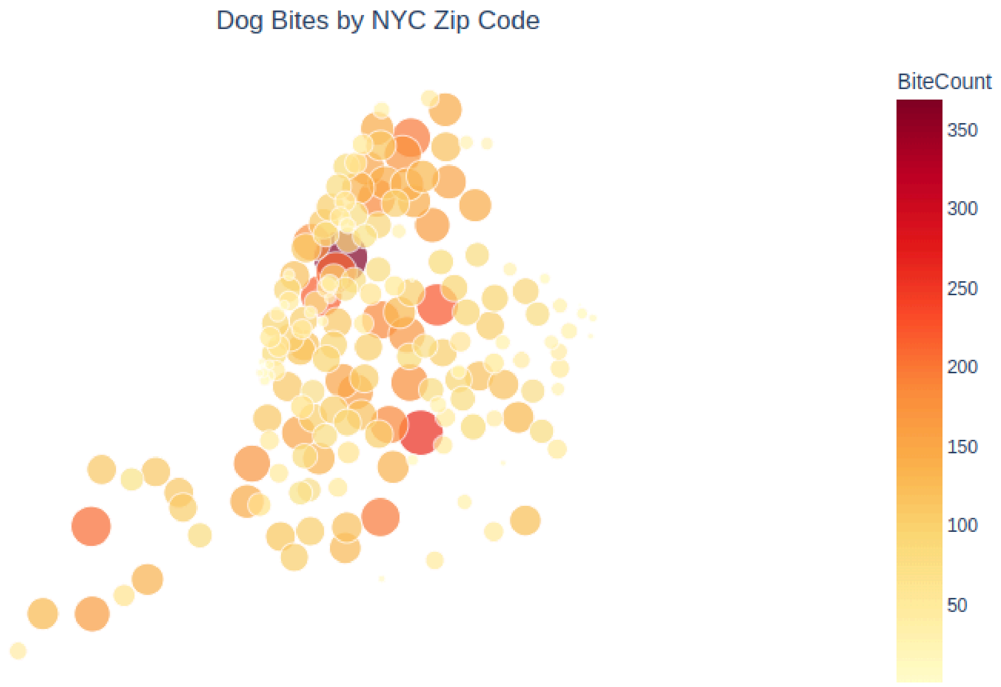


```
# load geojson with nyc zip codes found online
with open("nyc_zipcodes.geojson") as f:
   ny_zip_json = json.load(f)

# filter out and retain only the valid nyc_zips from geojson
valid_nyc_zips = set(f['properties']['postalCode'] for f in
ny_zip_json['features'])

# preprocess dog bites in nyc
zip_codes = data['ZipCode'].value_counts().reset_index()
zip_codes.columns = ['ZipCode', 'BiteCount']


############
# cleaning zip codes

# convert to string and remove .0 to normalize
```

```
zip_codes['ZipCode'] = zip_codes['ZipCode'].astype(str).str.replace('.0',
'')

# remove any zip codes longer than 5 digits
zip_codes = zip_codes[zip_codes['ZipCode'].str.len() <= 5]

# pad with leading zeros to ensure 5 digits
zip_codes['ZipCode'] = zip_codes['ZipCode'].str.zfill(5)

# keep only NYC zip codes
zip_codes = zip_codes[zip_codes['ZipCode'].isin(valid_nyc_zips)]

############


color_scale = "YlOrRd"
# color_scale = "geyser"        # really nice color scale but not the best
fit for this map. Keeping it here for a future project


# choropleth map
fig = px.choropleth(zip_codes,
                    geojson=ny_zip_json,
                    locations='ZipCode',
                    featureidkey="properties.postalCode",
                    color='BiteCount',
                    color_continuous_scale=color_scale,
                    projection="mercator",
                    title="Dog Bites by NYC Zip Code"
)

# fit map on zipcodes and create space for centered title
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(
   margin={"r":0,"t":30,"l":0,"b":0},  # top margin for title
   title_x=0.5                         # centers title
)
fig.show()
```



```
#  bubble map
fig = px.scatter_geo(zip_codes,
                    geojson=ny_zip_json,
                    locations='ZipCode',
                    featureidkey="properties.postalCode",
                    size='BiteCount',
                    size_max=25,
                    color='BiteCount',
                    color_continuous_scale=color_scale,
                    projection="mercator",
                    title='Dog Bites by NYC Zip Code')


# fit map on zipcodes and create space for centered title
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(
   margin={"r":0,"t":30,"l":0,"b":0},   # top margin for title
   title_x=0.5                          # centers title
)
fig.show()
```


Dog Bites by NYC Zip Code
BiteCount
350
300
250
200
150
100
50