Libraries

```
# Data manipulation
import pandas as pd
import numpy as np

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

Hurricane Sandy 311 Dataset

Load in Dataset

```
from google.colab import drive
import pandas as pd

# Mount Google Drive
drive.mount('/content/drive')

# Path to the CSV file in Google Drive
hurricane_data_path = '/content/drive/MyDrive/311data.csv'

# Read the CSV file into a pandas DataFrame
hurricane_data = pd.read_csv(hurricane_data_path)

# Display the first few rows of the DataFrame to verify
print(hurricane_data.head())
```

```
Mounted at /content/drive
         id location_type     incident_address       city    borough  \
0  23531046           NaN  111 LAWRENCE STREET   BROOKLYN   BROOKLYN
1  23534651        Street        5614 BROADWAY      BRONX      BRONX
2  23535556           NaN                  NaN   NEW YORK  MANHATTAN
3  23536817           NaN      41-15 45 STREET  SUNNYSIDE     QUEENS
4  23536826           NaN   140 EAST 46 STREET   NEW YORK  MANHATTAN

   latitude  longitude    created_date      closed_date agency  \
0  40.69283  -73.98623   7/1/2012 3:48  7/26/2012 12:29    TLC
1  40.87983  -73.90414  7/2/2012 14:05  7/17/2012 12:35    DOT
2  40.73986  -73.97774  7/2/2012 15:28   7/2/2012 21:00    DEP
3  40.74641  -73.91898  7/2/2012 11:20    7/6/2012 0:00    DOB
4  40.75378  -73.97372   7/2/2012 9:59   9/19/2012 0:00    DOB

                            agency_name     complaint_type  \
0          Taxi and Limousine Commission     Taxi Complaint
1           Department of Transportation  Broken Muni Meter
2  Department of Environmental Protection       Water System
3                Department of Buildings    Scaffold Safety
4                Department of Buildings       Building/Use

                                       description  resolution_date
0                                Driver Complaint  7/26/2012 12:29
1                                     No Receipt  7/17/2012 12:35
2                         Hydrant Running Full (WA4)   7/2/2012 21:00
3  Suspended (Hanging) Scaffolds - No Pmt/Lic/Dan...    7/6/2012 0:00
4   Illegal Conversion Of Residential Building/Space   9/19/2012 0:00
```

EDA

```
# Display basic information about the DataFrame
print(hurricane_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 815538 entries, 0 to 815537
Data columns (total 14 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   id                815538 non-null  int64  
 1   location_type     630641 non-null  object 
 2   incident_address  682999 non-null  object 
 3   city              815506 non-null  object 
 4   borough           815538 non-null  object 
 5   latitude          815538 non-null  float64
 6   longitude         815538 non-null  float64
 7   created_date      815538 non-null  object 
 8   closed_date       792891 non-null  object 
 9   agency            815538 non-null  object 
 10  agency_name       815538 non-null  object 
 11  complaint_type    815538 non-null  object 
 12  description       812917 non-null  object 
 13  resolution_date   804159 non-null  object 
dtypes: float64(2), int64(1), object(11)
memory usage: 87.1+ MB
None
```

```
# Display summary statistics for numerical columns
print(hurricane_data.describe())
```

```
                 id       latitude      longitude
count  8.155380e+05  815538.000000  815538.000000
mean   2.413080e+07      40.728566     -73.920981
std    3.540918e+05       0.087142       0.082524
min    2.352531e+07      40.498630     -74.254710
25%    2.382137e+07      40.664480     -73.964080
50%    2.414827e+07      40.720130     -73.924950
75%    2.442339e+07      40.807350     -73.874920
max    3.071714e+07      40.912870     -73.700390
```

```
print(hurricane_data.dtypes)
```

```
id                   int64
location_type       object
incident_address    object
city                object
borough             object
latitude           float64
longitude          float64
created_date        object
closed_date         object
agency              object
agency_name         object
complaint_type      object
description         object
resolution_date     object
dtype: object
```

```
print(hurricane_data.shape)
```

```
(815538, 14)
```

Cleaning

```
# Check for missing values
missing_values = hurricane_data.isnull().sum()
print(missing_values)
```

```
id                      0
location_type      184897
incident_address   132539
city                   32
borough                 0
latitude                0
longitude               0
created_date            0
closed_date         22647
agency                  0
agency_name             0
complaint_type          0
description          2621
resolution_date     11379
dtype: int64
```

```
# Handle missing values by dropping rows with missing values
cleaned_data = hurricane_data.dropna()


# Check for duplicate IDs
duplicate_ids = cleaned_data[cleaned_data.duplicated(subset='id', keep=False)]
print(f'Number of duplicate IDs: {duplicate_ids.shape[0]}')
```

```
Number of duplicate IDs: 0
```

```
# Check for duplicates based on "Created Date" and "Incident Address"
duplicate_created_address = cleaned_data[cleaned_data.duplicated(subset=['created_
print(f'Number of duplicate "Created Date" and "Incident Address": {duplicate_cre
```

```
Number of duplicate "Created Date" and "Incident Address": 221441
```

```
duplicate_full = duplicate_created_address[duplicate_created_address.duplicated(s
print(f'Number of duplicates with same "Created Date", "Incident Address", "Compl
```

```
Number of duplicates with same "Created Date", "Incident Address", "Complaint
```

Because there are no duplicate IDs (unique identifier) we will not be removing these "duplicates"

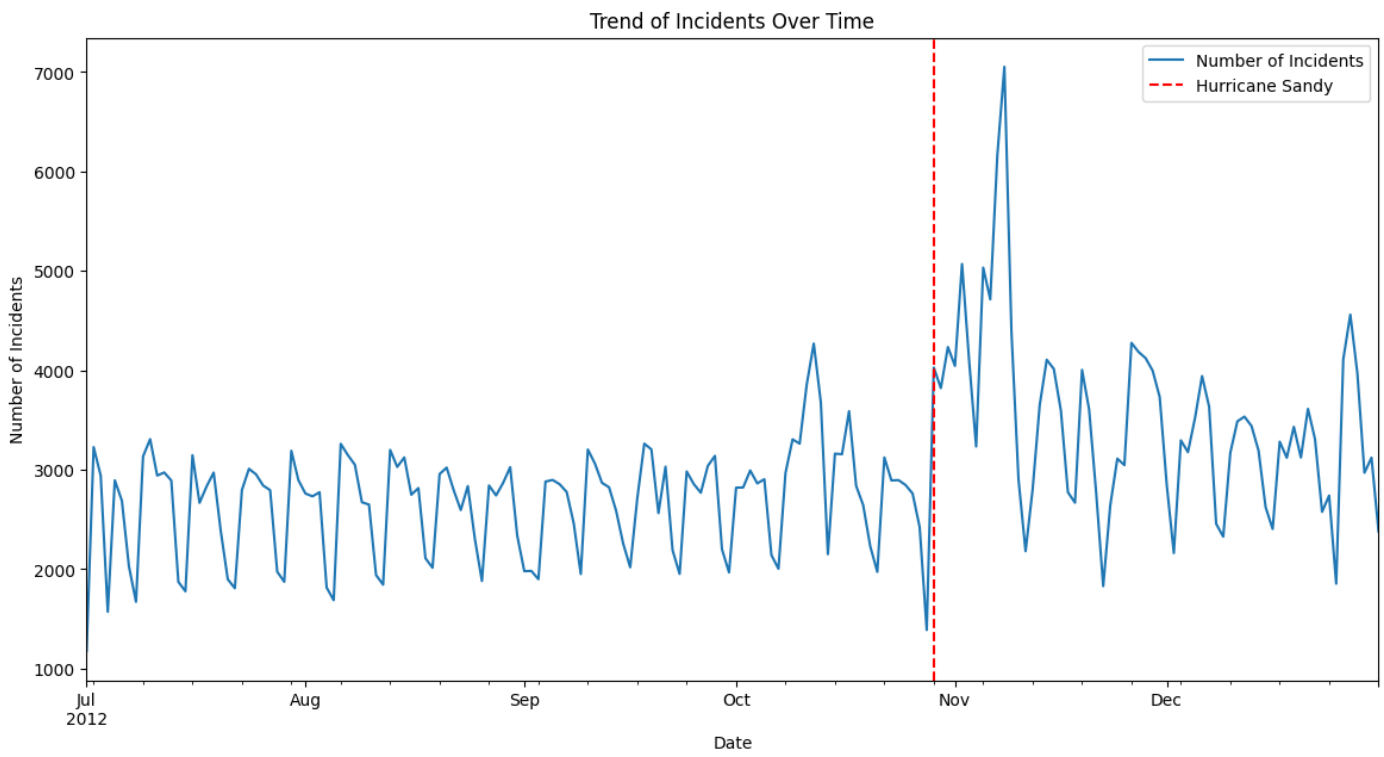
```
# Convert date columns to datetime format
cleaned_data['created_date'] = pd.to_datetime(cleaned_data['created_date'])
cleaned_data['closed_date'] = pd.to_datetime(cleaned_data['closed_date'])
cleaned_data['resolution_date'] = pd.to_datetime(cleaned_data['resolution_date'])

# Print the earliest and latest dates for 'created_date'
earliest_created_date = cleaned_data['created_date'].min()
latest_created_date = cleaned_data['created_date'].max()
print(f'Earliest Created Date: {earliest_created_date}')
print(f'Latest Created Date: {latest_created_date}')

# Print the earliest and latest dates for 'closed_date'
earliest_closed_date = cleaned_data['closed_date'].min()
latest_closed_date = cleaned_data['closed_date'].max()
print(f'Earliest Closed Date: {earliest_closed_date}')
print(f'Latest Closed Date: {latest_closed_date}')

# Print the earliest and latest dates for 'resolution_date'
earliest_resolution_date = cleaned_data['resolution_date'].min()
```

```
latest_resolution_date = cleaned_data['resolution_date'].max()
print(f'Earliest Resolution Date: {earliest_resolution_date}')
print(f'Latest Resolution Date: {latest_resolution_date}')
```

```
<ipython-input-13-69d7948a67dc>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  cleaned_data['created_date'] = pd.to_datetime(cleaned_data['created_date'])
<ipython-input-13-69d7948a67dc>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  cleaned_data['closed_date'] = pd.to_datetime(cleaned_data['closed_date'])
Earliest Created Date: 2012-07-01 00:00:00
Latest Created Date: 2012-12-31 00:00:00
Earliest Closed Date: 1900-01-01 00:00:00
Latest Closed Date: 2016-05-13 11:43:00
Earliest Resolution Date: 2012-05-21 00:00:00
Latest Resolution Date: 2016-05-13 11:43:00
<ipython-input-13-69d7948a67dc>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  cleaned_data['resolution_date'] = pd.to_datetime(cleaned_data['resolution_da
```

```
# Define the periods before and after Hurricane Sandy
before_sandy_start = '2012-09-01'
before_sandy_end = '2012-10-28'
after_sandy_start = '2012-10-29'
after_sandy_end = '2012-12-31'
```

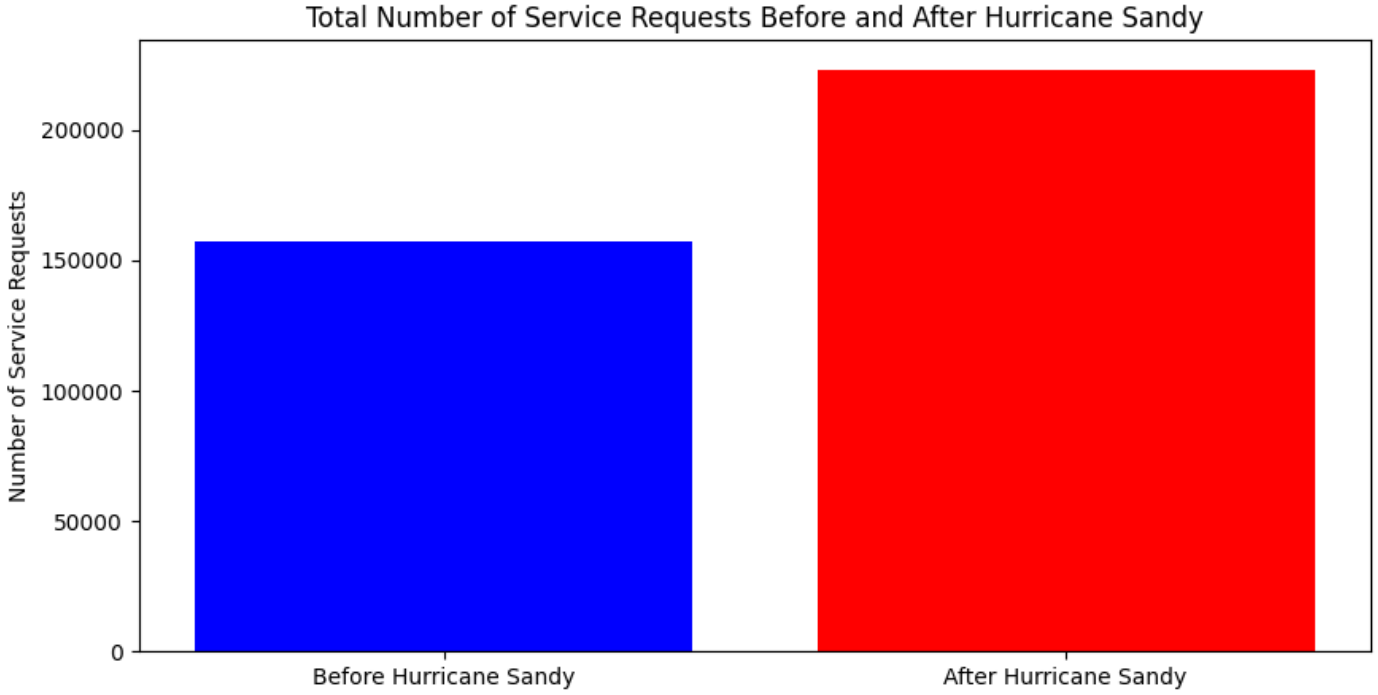
```
plt.figure(figsize=(14, 7))
cleaned_data.resample('D', on='created_date').size().plot(label='Number of Incide
plt.axvline(pd.Timestamp('2012-10-29'), color='r', linestyle='--', label='Hurrica
plt.title('Trend of Incidents Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Incidents')
plt.legend()
plt.show()
```



How did the number of service requests change before and after Hurricane Sandy? To compare, consider the total number of service requests during one month before the hurricane (i.e., from 2012-09-28 to 2012-10-29) and one month after the hurricane (i.e. from 2012-10-29 to 2012-11-29).

```
# Filter the data for the periods
before_sandy = cleaned_data[(cleaned_data['created_date'] >= before_sandy_start) &
after_sandy = cleaned_data[(cleaned_data['created_date'] >= after_sandy_start) &

# Count the number of service requests
before_sandy_count = before_sandy.shape[0]
after_sandy_count = after_sandy.shape[0]

print(f'Total number of service requests one month before Hurricane Sandy: {befor
print(f'Total number of service requests one month after Hurricane Sandy: {after_

# Plot the comparison
plt.figure(figsize=(10, 5))
plt.bar(['Before Hurricane Sandy', 'After Hurricane Sandy'], [before_sandy_count,
plt.title('Total Number of Service Requests Before and After Hurricane Sandy')
plt.ylabel('Number of Service Requests')
plt.show()
```

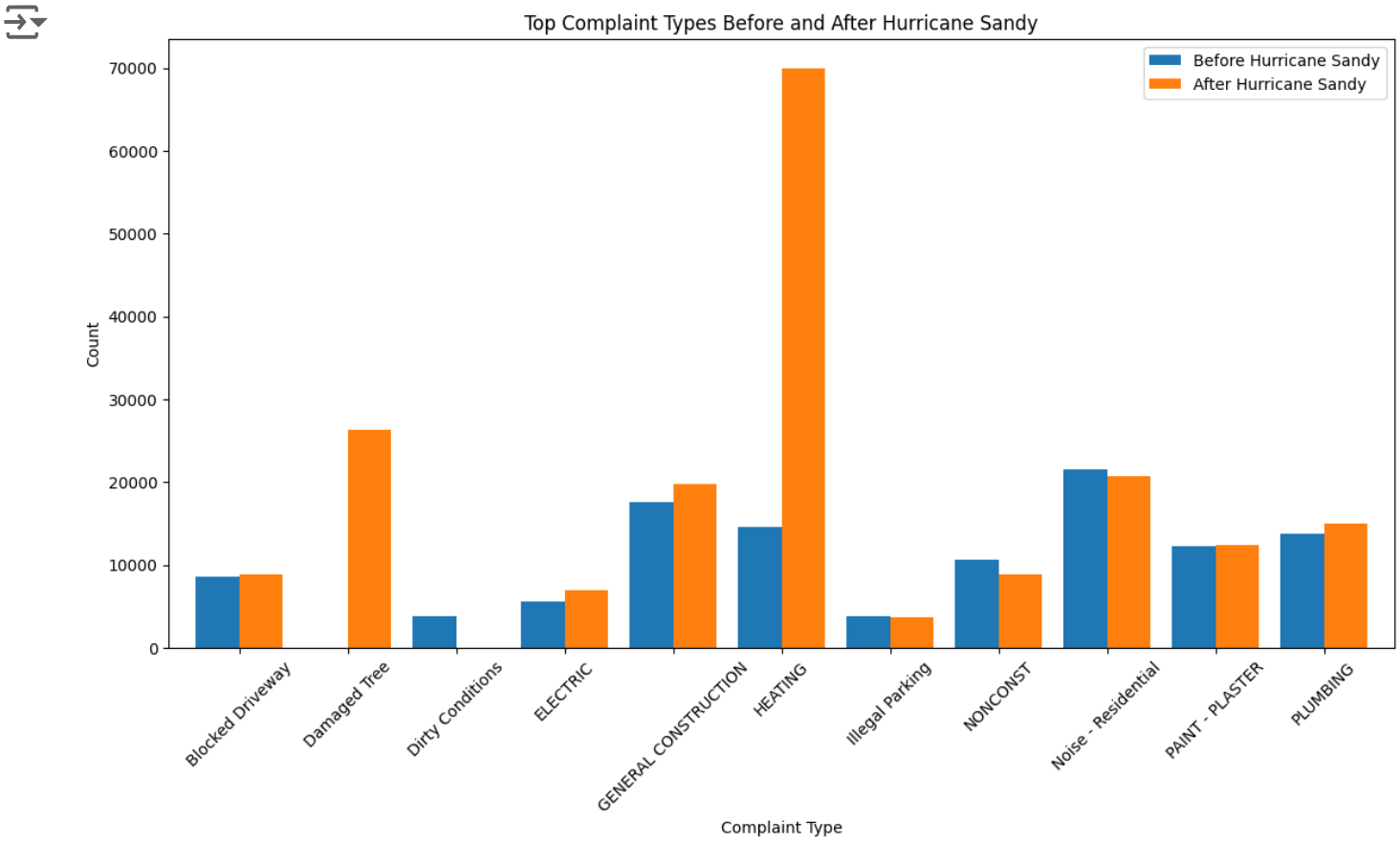
```
Total number of service requests one month before Hurricane Sandy: 157187
Total number of service requests one month after Hurricane Sandy: 223183
```



What types of service requests were most common after Hurricane Sandy, and how did they differ from other periods?

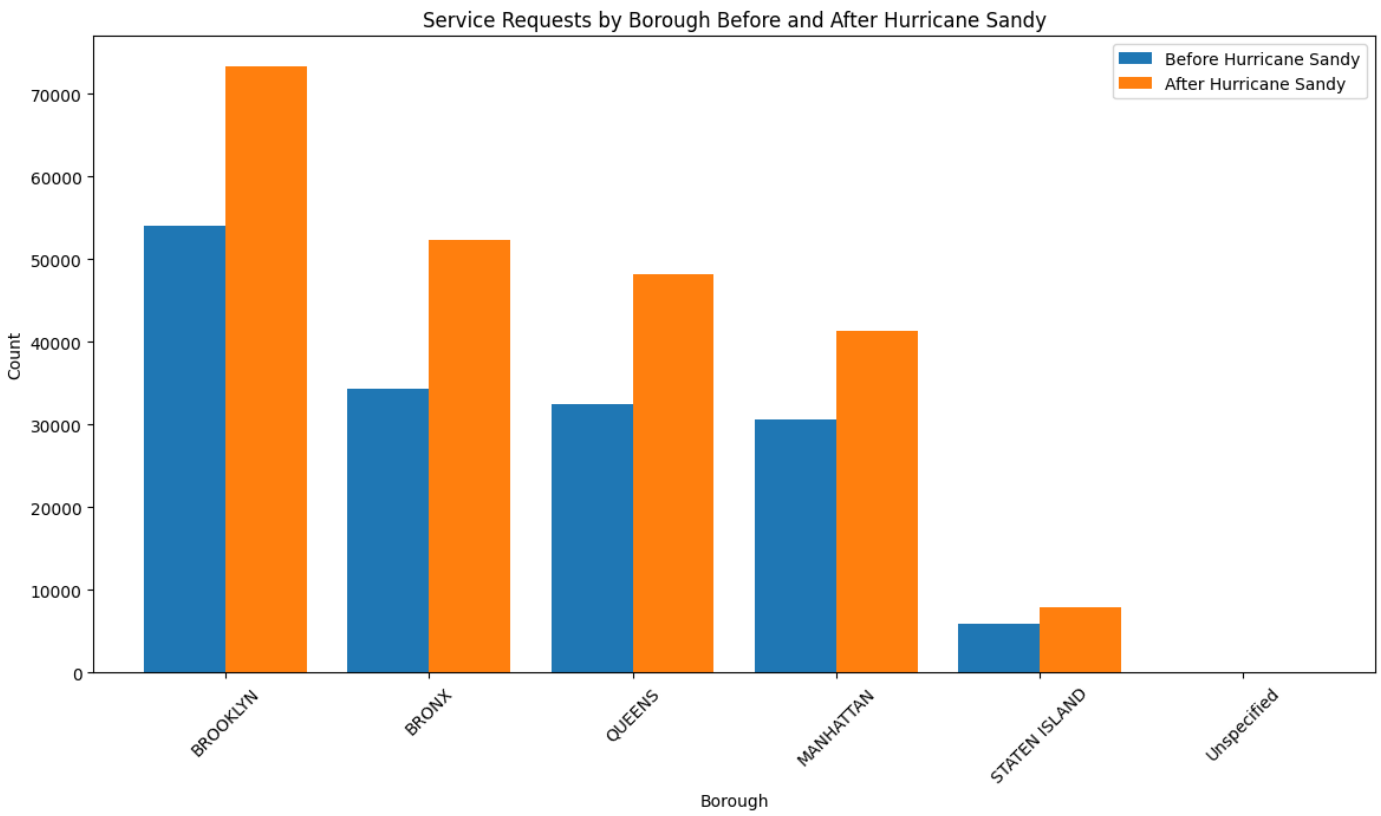
```
# Prepare data for the top 10 complaint types before and after Hurricane Sandy
top_complaints_before_sandy = before_sandy['complaint_type'].value_counts().nlarg
top_complaints_after_sandy = after_sandy['complaint_type'].value_counts().nlarges

# Create a DataFrame with both before and after counts
complaints_comparison = pd.DataFrame({
    'Before Hurricane Sandy': top_complaints_before_sandy,
    'After Hurricane Sandy': top_complaints_after_sandy
}).fillna(0)  # Fill NaN with 0 for complaint types that may not appear in both p

# Plot the comparison
complaints_comparison.plot(kind='bar', figsize=(14, 7), width=0.8)
plt.title('Top Complaint Types Before and After Hurricane Sandy')
```

```
plt.xlabel('Complaint Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



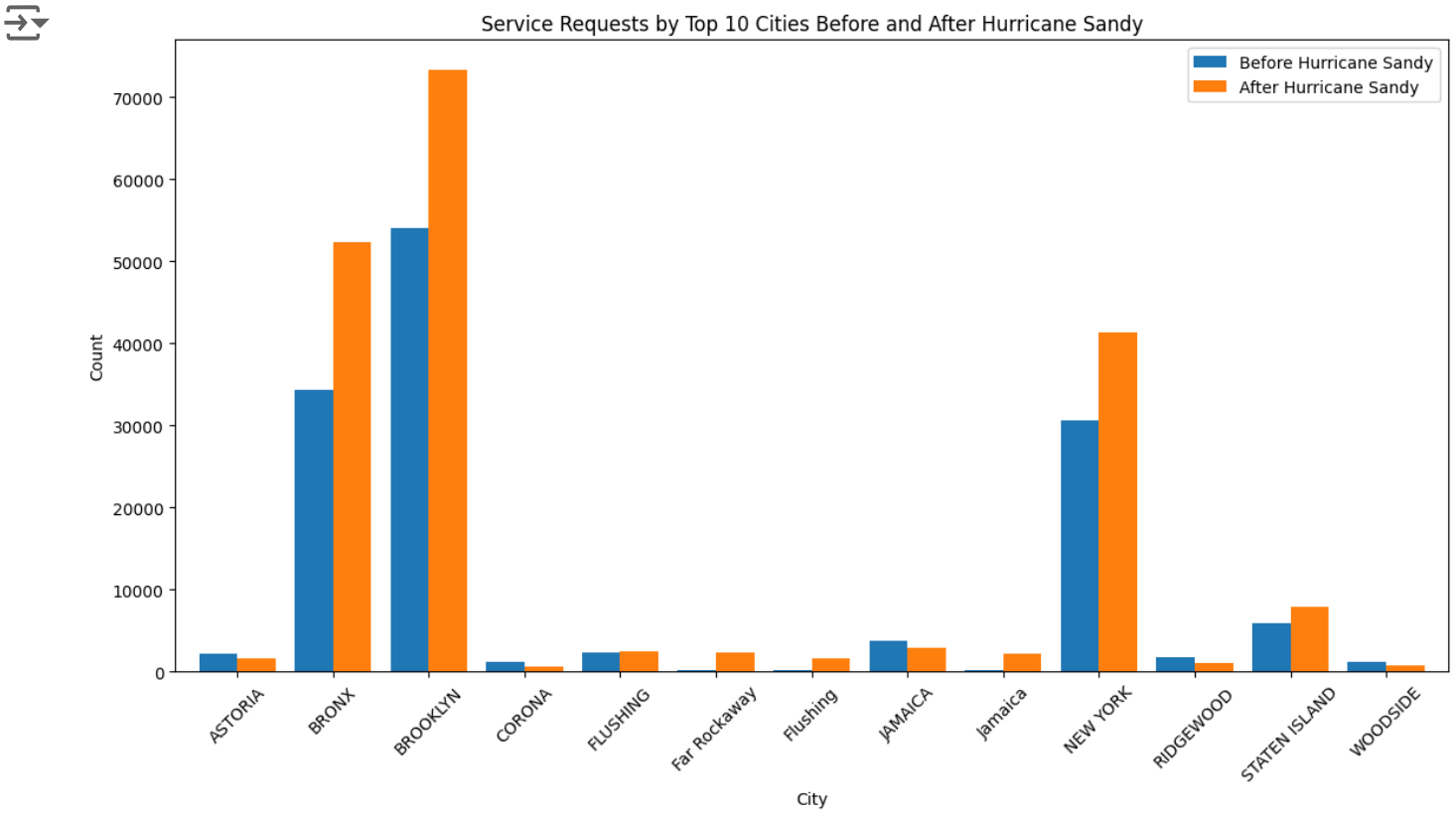
Which areas of New York City were most affected by Hurricane Sandy, based on the frequency and type of service requests?

```
# Group by borough and count the number of service requests
borough_counts_before = before_sandy['borough'].value_counts()
borough_counts_after = after_sandy['borough'].value_counts()

# Create a DataFrame with both before and after counts
borough_comparison = pd.DataFrame({
    'Before Hurricane Sandy': borough_counts_before,
    'After Hurricane Sandy': borough_counts_after
}).fillna(0)  # Fill NaN with 0 for boroughs that may not appear in both periods

# Plot the comparison
borough_comparison.plot(kind='bar', figsize=(14, 7), width=0.8)
plt.title('Service Requests by Borough Before and After Hurricane Sandy')
plt.xlabel('Borough')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



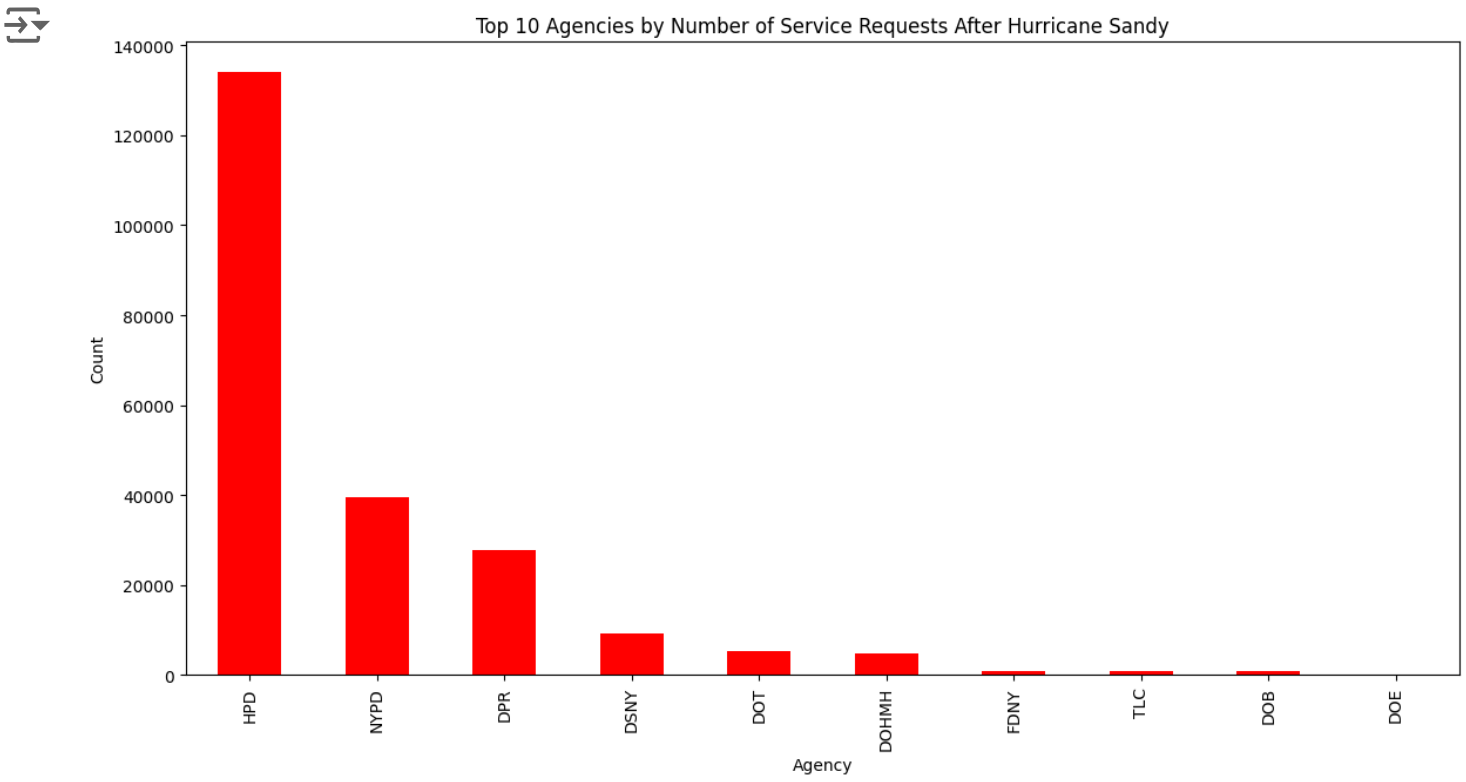

```
# Group by city and count the number of service requests
city_counts_before = before_sandy['city'].value_counts().nlargest(10)
city_counts_after = after_sandy['city'].value_counts().nlargest(10)

# Combine the top cities from both periods
top_cities = pd.concat([city_counts_before, city_counts_after], axis=1, keys=['Be

# Filter the original data to include only these top cities
filtered_before_sandy = before_sandy[before_sandy['city'].isin(top_cities.index)]
filtered_after_sandy = after_sandy[after_sandy['city'].isin(top_cities.index)]

# Recount the service requests for the filtered data
```

```
filtered_city_counts_before = filtered_before_sandy['city'].value_counts()
filtered_city_counts_after = filtered_after_sandy['city'].value_counts()

# Create a DataFrame with both before and after counts
city_comparison = pd.DataFrame({
    'Before Hurricane Sandy': filtered_city_counts_before,
    'After Hurricane Sandy': filtered_city_counts_after
}).fillna(0)

# Plot the comparison
city_comparison.plot(kind='bar', figsize=(14, 7), width=0.8)
plt.title('Service Requests by Top 10 Cities Before and After Hurricane Sandy')
plt.xlabel('City')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



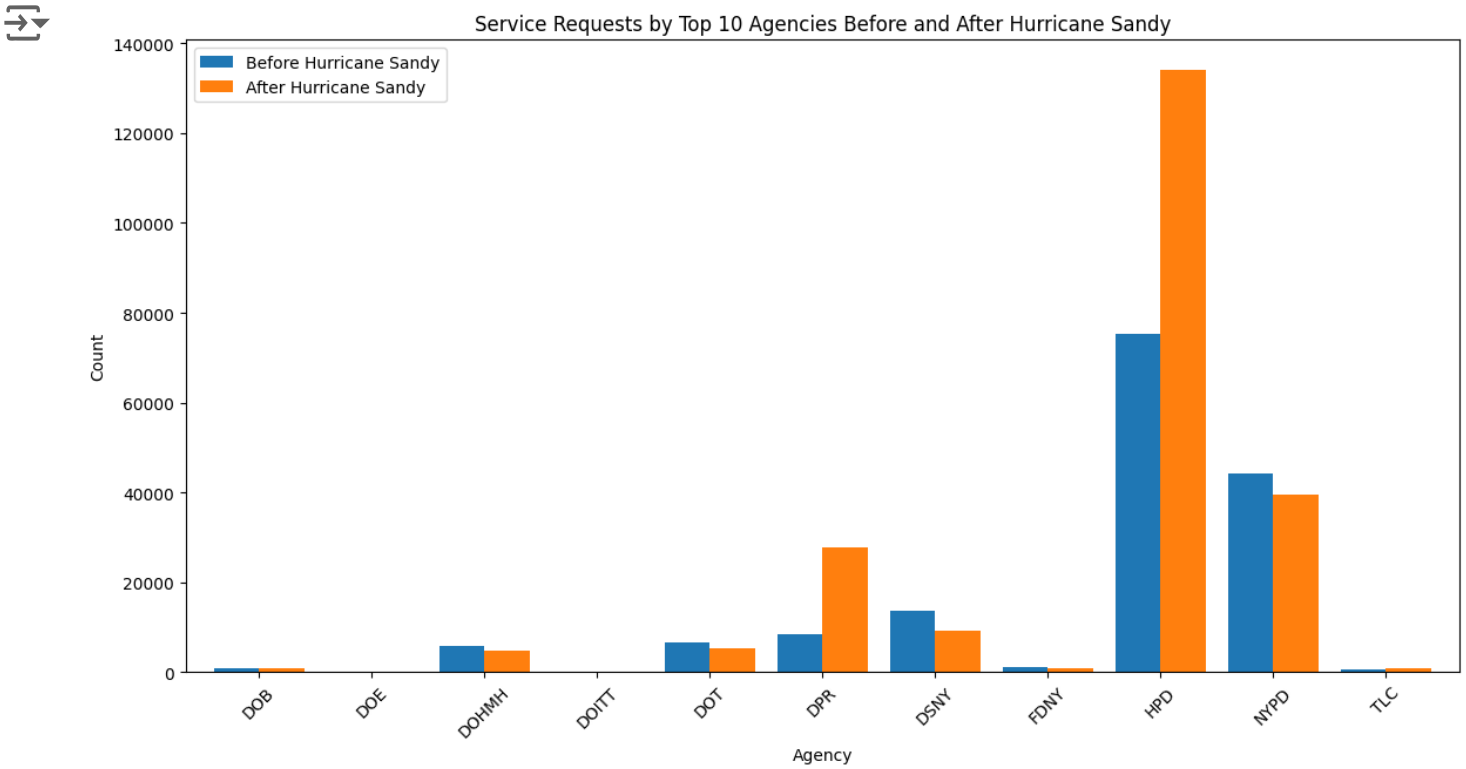
Which agencies received the highest service requests after the hurricane (again, consider a one-month timeframe)? You can plot the top 10 agencies on a bar chart.

```
# Get the top 10 agencies by number of service requests after Hurricane Sandy
top_agencies_after_sandy = after_sandy['agency'].value_counts().nlargest(10)

# Plot the top 10 agencies
plt.figure(figsize=(14, 7))
top_agencies_after_sandy.plot(kind='bar', color='red')
plt.title('Top 10 Agencies by Number of Service Requests After Hurricane Sandy')
plt.xlabel('Agency')
plt.ylabel('Count')
plt.show()
```



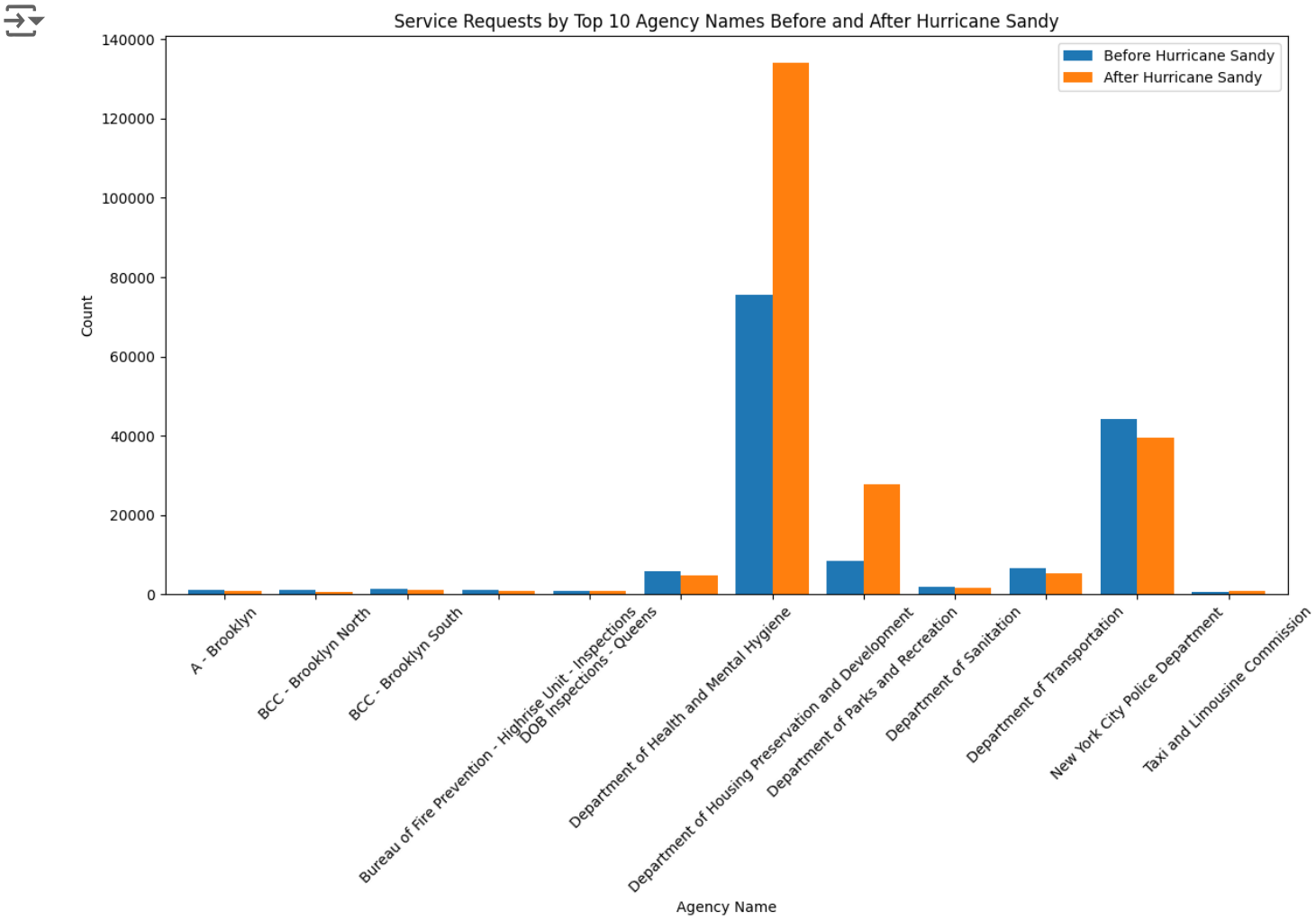
Extra visulizations for analysis

```
# Group by agency and count the number of service requests
agency_counts_before = before_sandy['agency'].value_counts().nlargest(10)
agency_counts_after = after_sandy['agency'].value_counts().nlargest(10)

# Combine the top agencies from both periods
top_agencies = pd.concat([agency_counts_before, agency_counts_after], axis=1, key

# Filter the original data to include only these top agencies
filtered_before_sandy = before_sandy[before_sandy['agency'].isin(top_agencies.ind
filtered_after_sandy = after_sandy[after_sandy['agency'].isin(top_agencies.index)

# Recount the service requests for the filtered data
filtered_agency_counts_before = filtered_before_sandy['agency'].value_counts()
filtered_agency_counts_after = filtered_after_sandy['agency'].value_counts()

# Create a DataFrame with both before and after counts
agency_comparison = pd.DataFrame({
    'Before Hurricane Sandy': filtered_agency_counts_before,
    'After Hurricane Sandy': filtered_agency_counts_after
}).fillna(0)

# Plot the comparison
agency_comparison.plot(kind='bar', figsize=(14, 7), width=0.8)
plt.title('Service Requests by Top 10 Agencies Before and After Hurricane Sandy')
plt.xlabel('Agency')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```




```
# Group by agency_name and count the number of service requests
agency_name_counts_before = before_sandy['agency_name'].value_counts().nlargest(1
agency_name_counts_after = after_sandy['agency_name'].value_counts().nlargest(10)

# Combine the top agency names from both periods
top_agency_names = pd.concat([agency_name_counts_before, agency_name_counts_after

# Filter the original data to include only these top agency names
filtered_before_sandy = before_sandy[before_sandy['agency_name'].isin(top_agency_
filtered_after_sandy = after_sandy[after_sandy['agency_name'].isin(top_agency_nam

# Recount the service requests for the filtered data
filtered_agency_name_counts_before = filtered_before_sandy['agency_name'].value_c
filtered_agency_name_counts_after = filtered_after_sandy['agency_name'].value_cou
```

```
# Create a DataFrame with both before and after counts
agency_name_comparison = pd.DataFrame({
    'Before Hurricane Sandy': filtered_agency_name_counts_before,
    'After Hurricane Sandy': filtered_agency_name_counts_after
}).fillna(0)

# Plot the comparison
agency_name_comparison.plot(kind='bar', figsize=(14, 7), width=0.8)
plt.title('Service Requests by Top 10 Agency Names Before and After Hurricane San
plt.xlabel('Agency Name')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

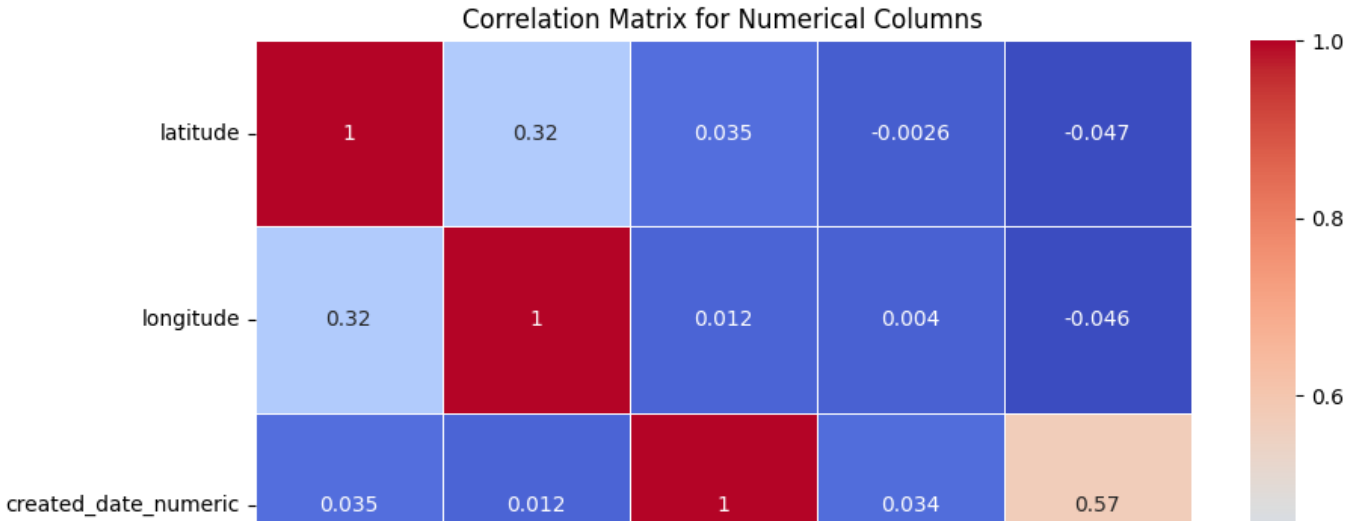


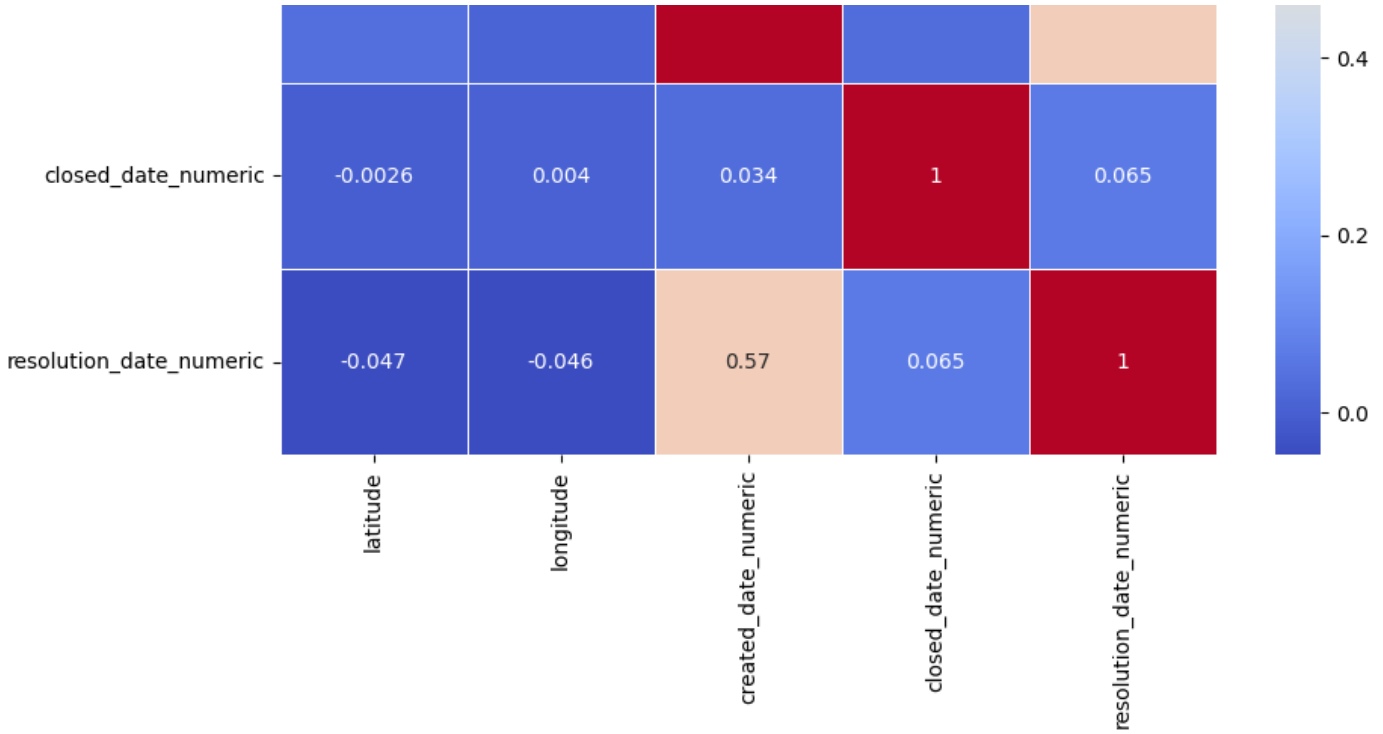
Analysis

```
# Convert date columns to numeric values (number of days since a reference date)
```

```
cleaned_data['created_date_numeric'] = (cleaned_data['created_date'] - pd.Timestar
cleaned_data['closed_date_numeric'] = (cleaned_data['closed_date'] - pd.Timestamp
cleaned_data['resolution_date_numeric'] = (cleaned_data['resolution_date'] - pd.T

# Select numerical columns
numerical_columns = ['latitude', 'longitude', 'created_date_numeric', 'closed_dat

# Calculate correlation matrix for numerical columns
correlation_matrix = cleaned_data[numerical_columns].corr()

# Plot the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix for Numerical Columns')
plt.show()
```

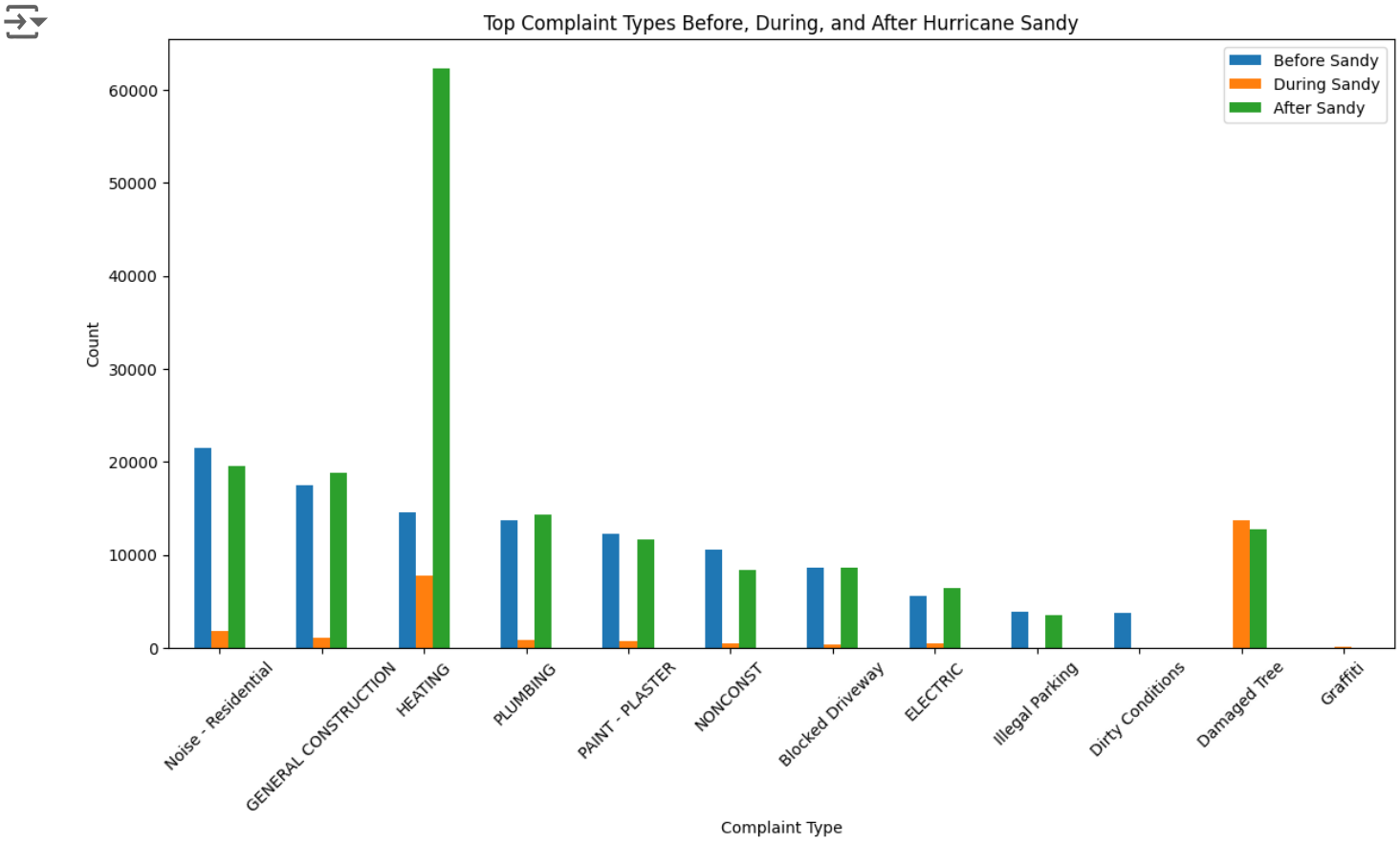
```
<ipython-input-23-b996fe14e155>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  cleaned_data['created_date_numeric'] = (cleaned_data['created_date'] - pd.Ti
<ipython-input-23-b996fe14e155>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  cleaned_data['closed_date_numeric'] = (cleaned_data['closed_date'] - pd.Time
<ipython-input-23-b996fe14e155>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  cleaned_data['resolution_date_numeric'] = (cleaned_data['resolution_date'] -
```






```
# Define the periods before, during, and after Hurricane Sandy
before_period = cleaned_data[(cleaned_data['created_date'] >= '2012-09-01') & (cl
during_period = cleaned_data[(cleaned_data['created_date'] >= '2012-10-28') & (cl
after_period = cleaned_data[(cleaned_data['created_date'] > '2012-11-04') & (clea

# Get the top 10 complaint types before, during, and after Hurricane Sandy
top_complaints_before = before_period['complaint_type'].value_counts().nlargest(1
top_complaints_during = during_period['complaint_type'].value_counts().nlargest(1
top_complaints_after = after_period['complaint_type'].value_counts().nlargest(10)

# Combine the top complaint types from all periods
```

```
top_complaints = pd.concat([top_complaints_before, top_complaints_during, top_com

# Plot the comparison
top_complaints.plot(kind='bar', figsize=(14, 7))
plt.title('Top Complaint Types Before, During, and After Hurricane Sandy')
plt.xlabel('Complaint Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit