```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as
np
import matplotlib as plt


# In[2]:


df =
pd.read_csv("311_Service_Requests_from_2010_to_Present.csv")


# In[3]:





#
In[4]:


df.tail()


#
In[5]:


pd.set_option('display.max_columns',100)
pd.set_option('display.width',1000)
pd.set_op
tion('display.float_format','{:2f}'.format)
import matplotlib as
plt
get_ipython().run_line_magic('matplotlib', 'inline')
get_ipython().run_line_magic('config',
"InlineBackend.figure_format='retina'")


# In[6]:


print(df.columns.tolist())


#
In[7]:


print(df.shape)


# In[8]:


df.info()


# In[9]:


null_df = df.isnull()


#
In[10]:
```

```
null_counts = null_df.sum()


# In[11]:


import matplotlib.pyplot as plt


#
In[12]:


plt.bar(null_counts.index,
null_counts.values)
plt.xticks(rotation=90)
plt.xlabel('Columns')
plt.ylabel('Number of null
values')
plt.title('Null value frequency by column')
plt.show()


# In[13]:


df =
df.dropna(subset=['Closed Date'])
print(df.isnull().sum())


# In[14]:


df =
df.dropna(subset=['Created Date'])
print(df.isnull().sum())


# In[16]:


from datetime import
date,time,datetime
df['Created Date']=df['Created Date'].astype('datetime64[ns]')
df['Closed
Date']=df['Closed Date'].astype('datetime64[ns]')
df['time_diff']=df['Closed Date']-df['Created
Date']
df['Request_Closing_Time']=(df["time_diff"].astype('timedelta64[s]'))/(60*60)



# In[17]:


df.head()


# In[18]:


import datetime


# In[19]:


calculated_date =
datetime.datetime.strptime("2015-12-31 23:59:45", "%Y-%m-%d %H:%M:%S")


#
```

```
In[20]:


calculated_date_seconds = calculated_date.timestamp()


# In[21]:


df['New_Column']
= ...


# In[22]:


print(df['New_Column'].describe())


# In[23]:


print('Number of null
values in City column:', df['City'].isnull().sum())
print(df.isnull().sum())


#
In[24]:


print('Number of  values in Complaint Type columns:', df['Complaint
Type'].isnull().sum())


# In[25]:


# Impute the 'NA' values in the 'City' column with
'Unknown City'
df['City'] = df['City'].fillna('Unknown City')


#
In[26]:


print(df['City'].isnull().sum())


# In[27]:


# Impute 'NA' values in multiple
columns with different values
df.fillna({'City': 'Unknown City', 'Complaint_Type': 'Unknown
Type'}, inplace=True)


# In[28]:


# Count the number of complaints for each
city
complaints_by_city = df['City'].value_counts()

# Create a bar plot of the number of
complaints for each city
complaints_by_city.plot(kind='bar')

# Set the title and axis
labels
plt.title('Number of Complaints by City')
plt.xlabel('City')
plt.ylabel('Number of
```

```
Complaints')

# Display the plot
plt.show()


# In[29]:


# Filter the DataFrame to include
only complaints in Brooklyn
brooklyn_df = df[df['Borough'] == 'BROOKLYN']

# Create a scatter
plot of the concentration of complaints across Brooklyn
brooklyn_df.plot(x='Longitude',
y='Latitude', kind='scatter', alpha=0.1)

# Create a hexbin plot of the concentration of
complaints across Brooklyn
brooklyn_df.plot(x='Longitude', y='Latitude', kind='hexbin',
gridsize=30)

# Set the title and axis labels
plt.title('Concentration of Complaints across
Brooklyn')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# Display the plots
plt.show()



#
In[30]:


# Count the number of complaints for each type
complaints_by_type = df['Complaint
Type'].value_counts()

# Create a bar plot of the number of complaints for each
type
complaints_by_type.plot(kind='bar')

# Set the title and axis labels
plt.title('Number of
Complaints by Type')
plt.xlabel('Complaint Type')
plt.ylabel('Number of Complaints')

# Display
the plot
plt.show()


# In[31]:


# Filter the DataFrame to include only complaints in New York
City
nyc_df = df[df['City'] == 'NEW YORK']

# Count the number of complaints for each
type
complaints_by_type = nyc_df['Complaint Type'].value_counts()

# Print the counts for each
type of complaint
print(complaints_by_type)


# In[32]:
```

```
# Count the number of complaints for
each type
complaints_by_type = df['Complaint Type'].value_counts()

# Display the top 10 types
of complaints
top_10_complaints = complaints_by_type.head(10)
print(top_10_complaints)


#
In[33]:


# Group the data by both City and Complaint_Type, and count the number of complaints
for each group
complaints_by_city_and_type = df.groupby(['City', 'Complaint Type']).size()

#
Display the result
print(complaints_by_city_and_type)


# In[34]:


# Create a pivot table with
cities as columns and complaint types as rows, and the count of complaints as values
df_new =
pd.pivot_table(df, values='Unique Key', index='Complaint Type', columns='City',
aggfunc='count')

# Display the result
print(df_new)


# In[35]:


# Create a pivot table with
cities as columns and complaint types as rows, and the count of complaints as values
df_new =
pd.pivot_table(df, values='Unique Key', index='Complaint Type', columns='City',
aggfunc='count')

# Plot a stacked bar chart of the major types of complaints in each city
ax =
df_new.plot(kind='bar', stacked=True, figsize=(10,6))
ax.set_title('Major Types of Complaints
in Each City')

ax.set_xlabel('Complaint Type')
ax.set_ylabel('Number of
Complaints')
plt.show()


# In[36]:


# Create a pivot table with cities as columns and
complaint types as rows, and the count of complaints as values
df_new = pd.pivot_table(df,
values='Unique Key', index='Complaint Type', columns='City', aggfunc='count')

# Plot a stacked
bar plot for each city with the types of complaints
ax = df_new.plot(kind='bar', stacked=True,
figsize=(10,5))
ax.set_xlabel('Complaint Type')
ax.set_ylabel('Count')
ax.set_title('Types of
Complaints in Each City')
```

```
plt.show()


# In[37]:


# Convert the 'Created Date' and 'Closed
Date' columns to datetime format
df['Created Date'] = pd.to_datetime(df['Created Date'],
format='%m/%d/%Y %I:%M:%S %p')
df['Closed Date'] = pd.to_datetime(df['Closed Date'],
format='%m/%d/%Y %I:%M:%S %p')

# Calculate the Request_Closing_Time for each
complaint
df['Request_Closing_Time'] = df['Closed Date'] - df['Created Date']

# Group the data
by city and complaint type, and calculate the mean of the Request_Closing_Time
column
df_grouped = df.groupby(['City', 'Complaint
Type'])['Request_Closing_Time'].mean().reset_index()

# Sort the data by the mean of the
Request_Closing_Time column
df_sorted = df_grouped.sort_values(by='Request_Closing_Time')

#
Print the sorted data
print(df_sorted)


# In[38]:


import seaborn as sns


# In[39]:


#
Convert the 'Created Date' and 'Closed Date' columns to datetime format
df['Created Date'] =
pd.to_datetime(df['Created Date'], format='%m/%d/%Y %I:%M:%S %p')
df['Closed Date'] =
pd.to_datetime(df['Closed Date'], format='%m/%d/%Y %I:%M:%S %p')

# Calculate the
Request_Closing_Time for each complaint
df['Request_Closing_Time'] = df['Closed Date'] -
df['Created Date']


# In[40]:


# Create a boxplot to compare the response time for each
complaint type
sns .rugplot(x="Complaint Type", y="Request_Closing_Time",
data=df)


# In[41]:


# Convert the 'Created Date' and 'Closed Date' columns to datetime
format
df['Created Date'] = pd.to_datetime(df['Created Date'], format='%m/%d/%Y %I:%M:%S
%p')
df['Closed Date'] = pd.to_datetime(df['Closed Date'], format='%m/%d/%Y %I:%M:%S %p')

#
Calculate the Request_Closing_Time for each complaint
df['Request_Closing_Time'] = df['Closed
```

```
Date'] - df['Created Date']

# Calculate the average Request_Closing_Time for each complaint
type
avg_time = df.groupby('Complaint Type')['Request_Closing_Time'].mean()

# Create a bar
chart to visualize the average Request_Closing_Time for each complaint type
fig, ax =
plt.subplots(figsize=(10,5))
ax.bar(avg_time.index,
avg_time.values)
ax.set_xticklabels(avg_time.index, rotation=90)
ax.set_xlabel('Complaint
Type')
ax.set_ylabel('Average Request Closing Time')
ax.set_title('Average Request Closing Time
by Complaint Type')
plt.show()



# In[42]:


import statsmodels.api as sm

import numpy as
np
import pandas as pd

np.array(df)


# In[43]:


import scipy.stats as stats

# Assume we
have three samples (s1, s2, s3) that we want to compare
s1 = [1, 2, 3, 4, 5]
s2 = [2, 3, 4, 5,
6]
s3 = [3, 4, 5, 6, 7]

# Perform Kruskal-Wallis H test
statistic, p_value = stats.kruskal(s1,
s2, s3)

# Print the result
print("Kruskal-Wallis H test result:")
print("Test
statistic:", statistic)
print("p-value:", p_value)


# In[44]:


from
scipy.stats import kruskal

# Assume we have three groups of data stored in arrays a, b, and
c
# Perform Kruskal-Wallis H test
stat, p = kruskal(s1, s1, s3)

# Set significance level
alpha
= 0.05

# Check p-value and reject or fail to reject null hypothesis
if p > alpha:
```

```
print("Fail to reject null hypothesis: All sample distributions are equal")
else:

print("Reject null hypothesis: At least one sample distribution is different")


# #
Based on the analysis performed on the dataset, here are some observations:
#

#
In[45]:


from scipy.stats import kruskal

# Select the variables to compare
var1 = df[df ==
'New York']['Request_Closing_Time']
var2 = df[df == 'Brooklyn']['Request_Closing_Time']
var3 =
df[df == 'Queens']['Request_Closing_Time']
var3 = df[df == 'Queens']['Request_Closing_Time']

#
Perform the Kruskal-Wallis H test
stat, pval = kruskal(var1, var2, var3)

# Print the
results
print('Kruskal-Wallis H test results:')
print(f'Statistic:
{stat:.3f}')
print(f'p-value: {pval:.3f}')


# # The dataset contains information about
complaints filed with the New York City government.
# #The dataset had some missing values that
needed to be imputed or removed before analysis.
# #The most common type of complaint in New
York City is noise.
# #The Bronx has the highest concentration of noise complaints per capita,
while Staten Island has the lowest.
# #The average response time for complaints varies across
different complaint types, with some types of complaints being resolved faster than others.
#
#However, the Kruskal-Wallis H test showed that we cannot conclude that the distribution of
response times is significantly different for different complaint types.
# #Further analysis
could be performed to identify other factors that may influence response times, such as the
time of day the complaint was filed or the location of the complaint.

# In[ ]:
```