```
import numpy as np
import pandas as pd

# In[2]:

data=pd.read_csv('311_Service_Requests_from_2010_to_Present.csv',low_memory=False)

# In[3]:

df_data=pd.DataFrame(data)

# In[4]:

df_data.head()

# In[5]:

df_data.fillna(0)

# In[6]:

df_data.columns

# In[15]:

df_data.drop('School Not Found',axis=1)

# In[16]:

df_data.shape

# In[17]:

df_data['Created Date']=pd.to_datetime(df_data['Created
Date']) df_data['Closed Date']=pd.to_datetime(df_data['Closed
Date']) data1=df_data['Closed Date']-df_data['Created Date']

# In[18]:

data1.head()
```

```
# In[19]:


df_data['request_time']=data1

# In[20]:

df_data.head()

# In[24]:

df_data.info()

# In[21]:

import matplotlib.pyplot as plt

# In[22]:

get_ipython().run_line_magic('matplotlib', 'inline')

# In[23]:

import seaborn as sn
sn.displot(df_data['Status'])
plt.show()

# In[25]:

df_data.loc[:,['Complaint Type','City']]

# In[26]:

major=df_data.loc[:,['Complaint Type']]

# In[27]:

major

# In[28]:

top=major.value_counts()
top

# In[29]:
```

```
top10_complaint=top.head(10)
top10_complaint


# In[30]:


import seaborn as sn
sn.displot(top10_complaint)
plt.show()


# In[31]:


top10_complaint.plot(kind='bar',title='The major complaint types and their count')


# In[32]:


major_cities=df_data.loc[:,['City']]
major_cities


# In[33]:


top_cities=major_cities.value_counts()
top_cities


# In[34]:


top10_cities=top_cities.head(10)


# In[35]:


top10_cities


# In[36]:


top10_cities.plot(kind='bar',title='Cities which have higher Complaints')


# In[37]:


loc_type=df_data.loc[:,'Location Type']
location_count=loc_type.value_counts()
location_count


# In[39]:


location_count.head(10).plot(kind='bar',title='Cities which have higher Complaints')


# In[40]:
```

```
df_data.loc[:,['Complaint Type','City','Location','request_time']]
```

```
# In[41]:
```

```
mean_data = df_data[['City','Complaint Type','request_time']]
mean_data.dropna(subset = ['City','Complaint Type','request_time'], inplace = True)
mean_data['Mean'] = np.around( (mean_data['request_time'].astype(np.int64)/
                                        (pow(10,9)*3600) ), decimals=2)
```

```
# In[42]:
```

```
mean_data['Mean']
```

```
# In[43]:
```

```
mean_data.head(6)
```

```
# In[44]:
```

```
Average_time = np.around((mean_data['Mean'].mean()),decimals=2)
```

```
# In[45]:
```

```
Average_time
```

```
# In[46]:
```

```
mean_data.columns
```

```
# In[47]:
```

```
mean_data['Complaint Type']
```

```
# In[48]:
```

```
Complaint_AvgTime = mean_data.groupby(['Complaint Type']).agg({'Mean':'mean'})
Complaint_AvgTime = pd.DataFrame(Complaint_AvgTime)
Complaint_AvgTime = Complaint_AvgTime.sort_values(['Mean']).reset_index()
Complaint_AvgTime
```

```
# In[49]:
```

```
sample1=Complaint_AvgTime.sample(frac=.5)
```

```
# In[50]:
```

```
sample1
```

```
# In[51]:
```

```
sample2 = Complaint_AvgTime.drop(sample1.index)
sample2
```

```
# In[52]:
```

```
print('Mean of 1st sample =',np.around(float(sample1['Mean'].mean()),decimals=2))
print('Standard dev. of 1st sample =',np.around(float(sample1['Mean'].std()),decimals=2))
print('Mean of 2nd sample =',np.around(float(sample2['Mean'].mean()),decimals=2))
print('Standard dev. of 2nd sample =',np.around(float(sample2['Mean'].std()),decimals=2))
```

```
# In[53]:
```

```
import scipy.stats as stat
```

```
# In[54]:
```

```
ttest_2sp, p_val = stat.ttest_ind(sample1['Mean'],sample2['Mean'])
print('T-statistic is =',ttest_2sp)
print('p value is =',np.around(p_val,decimals=2))
```

```
# In[55]:
```

```
if (p_val<0.05):
    print('Null hypothesis is rejected since p value ({}) is less than 0.05'.format(np.around(p_val,decimals=2)))
else:
    print('Null hypothesis is accepted since p value ({}) is greater than 0.05'.format(np.around(p_val,decimals=2)))
```

```
# ## we can't reject the null hypothesis. null hypothesis-there is no relation between the average response time and complaint type
```

```
# In[ ]:
```