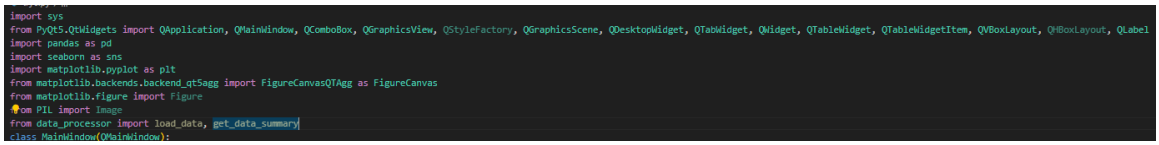
Abed Alrahman Alrabie': GH1033903
Hadi ali chebli:GH1033528

Problem statement:

The problem is analyzing street crime data in Camden to provide better insight and visualizations that help local law enforcement, city planners, and community organizations. By prioritizing this problem to be solved, these agencies can work towards better crime prevention, have more efficient resource planning, and improve community safety.

Code explanation:

1. Importing the proper libraries:

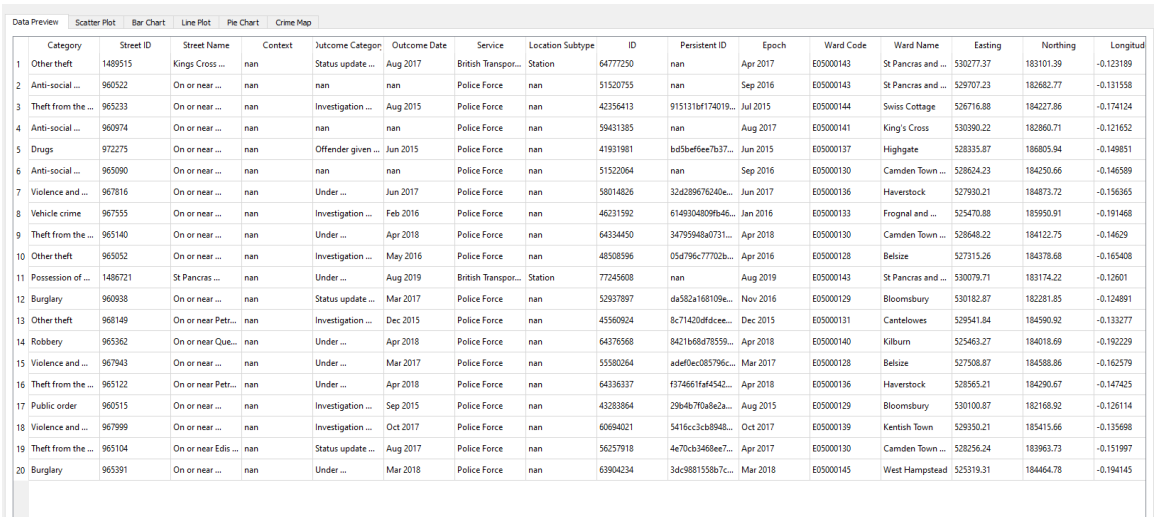


2. Main window class:
    - Starts the main window and its other parts

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.data1 = None
        self.setWindowTitle("Dataset Analyzer")
        screen_geometry = QDesktopWidget().screenGeometry()
        self.setGeometry(screen_geometry)
        self.base_scene = QGraphicsScene(self)
        self.plot_scene = QGraphicsScene(self)
        self.baseComponent = QGraphicsView(self.base_scene, self)
        self.header_dropdown = QComboBox(self)
        self.header_dropdown.setMinimumWidth(200)
        self.header_dropdown.addItem("Select Header Field")
        self.header_dropdown.currentIndexChanged.connect(self.header_selected)
        self.tab_widget = QTabWidget(self)
```

    - Loads the file and handles possible file loading errors and sets up different tabs for different data representation.

```
    try:
        data = self.open_file()
        self.tab_widget.addTab(self.dummy_data_tab(data), "Data Preview")
        scatter_tab = ScatterPlotTab(data, "Street ID", "Easting")
        self.tab_widget.addTab(scatter_tab, "Scatter Plot")
        bar_chart_tab = BarChartTab(data, "Category", "Ward Name")
        self.tab_widget.addTab(bar_chart_tab, "Bar Chart")
        line_plot_tab = LinePlotTab(data, "Epoch", "Easting")
        self.tab_widget.addTab(line_plot_tab, "Line Plot")
        pie_chart_tab = PieChartTab(data)
        self.tab_widget.addTab(pie_chart_tab, "Pie Chart")
        map_tab = MapTab(data)
        self.tab_widget.addTab(map_tab, "Crime Map")
    except Exception as e:
        error_label = QLabel(f"Failed to load data: {str(e)}")
        error_tab = QWidget()
        layout = QVBoxLayout(error_tab)
        layout.addWidget(error_label)
        error_tab.setLayout(layout)
        self.tab_widget.addTab(error_tab, "Error")

    available_height = self.height() - self.menuBar().height() - 50
    self.baseComponent.setGeometry(10, self.menuBar().height(), 230, available_height)
    self.tab_widget.setGeometry(250, self.menuBar().height(), self.width() - 300, available_height)
    self.header_dropdown.move(20, 10 + self.menuBar().height())

def header_selected(self, index):
    selected_field = self.header_dropdown.currentText()
    if selected_field != "Select Header Field":
        print(f"Selected header field: {selected_field}")

def open_file(self):
    file_path = "data/On_Street_Crime_In_Camden.csv"
    if file_path:
        data = pd.read_csv(file_path, low_memory=False)
        print(f"Header: {data.columns}")
        self.header_dropdown.addItems(data.columns)
        print("----------------------------------------")
        data['Latitude'] = pd.to_numeric(data['Latitude'], errors='coerce')
        data['Longitude'] = pd.to_numeric(data['Longitude'], errors='coerce')
        data = data.dropna(subset=['Latitude', 'Longitude'])

        print(f"Number of valid entries: {len(data)}")
        print(f"Latitude range: {data['Latitude'].min()} to {data['Latitude'].max()}")
        print(f"Longitude range: {data['Longitude'].min()} to {data['Longitude'].max()}")

        return data
```

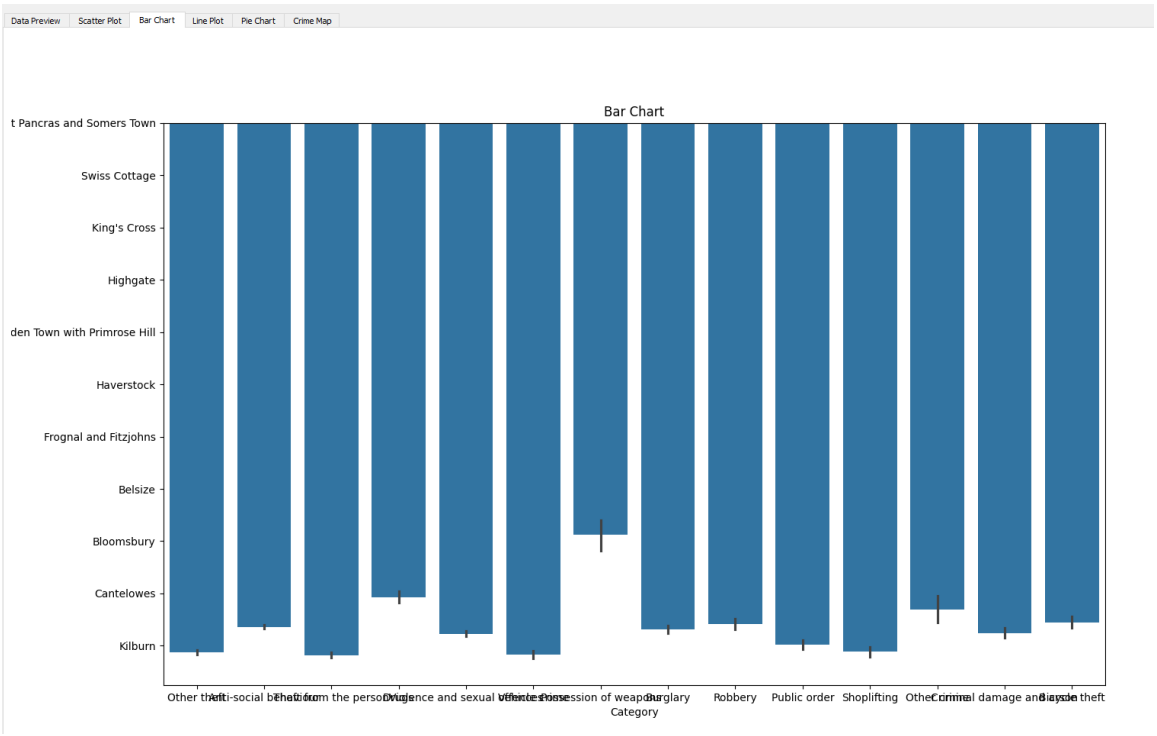


Data Preview | Scatter Plot | Bar Chart | Line Plot | Pie Chart | Crime Map

	Category	Street ID	Street Name	Context	Outcome Categor	Outcome Date	Service	Location Subtype	ID	Persistent ID	Epoch	Ward Code	Ward Name	Easting	Northing	Longitud
1	Other theft	1489515	Kings Cross ...	nan	Status update ...	Aug 2017	British Transpor...	Station	64777250	nan	Apr 2017	E05000143	St Pancras and ...	530277.37	183101.39	-0.123189
2	Anti-social ...	960522	On or near ...	nan	nan	nan	Police Force	nan	51520755	nan	Sep 2016	E05000143	St Pancras and ...	529707.23	182682.77	-0.131558
3	Theft from the ...	965233	On or near ...	nan	Investigation ...	Aug 2015	Police Force	nan	42356413	915131bf174019...	Jul 2015	E05000144	Swiss Cottage	526716.88	184227.86	-0.174124
4	Anti-social ...	960974	On or near ...	nan	nan	nan	Police Force	nan	59431385	nan	Aug 2017	E05000141	King's Cross	530390.22	182860.71	-0.121652
5	Drugs	972275	On or near ...	nan	Offender given ...	Jun 2015	Police Force	nan	41931981	bd5bef6ee7b37...	Jun 2015	E05000137	Highgate	528335.87	186805.94	-0.149851
6	Anti-social ...	965090	On or near ...	nan	nan	nan	Police Force	nan	51522064	nan	Sep 2016	E05000130	Camden Town ...	528624.23	184250.66	-0.146589
7	Violence and ...	967816	On or near ...	nan	Under ...	Jun 2017	Police Force	nan	58014826	32d289676240e...	Jun 2017	E05000136	Haverstock	527930.21	184873.72	-0.156365
8	Vehicle crime	967555	On or near ...	nan	Investigation ...	Feb 2016	Police Force	nan	46231592	6149304809fb46...	Jan 2016	E05000133	Frognal and ...	525470.88	185950.91	-0.191468
9	Theft from the ...	965140	On or near ...	nan	Under ...	Apr 2018	Police Force	nan	64334450	34795948a0731...	Apr 2018	E05000130	Camden Town ...	528648.22	184122.75	-0.14629
10	Other theft	965052	On or near ...	nan	Investigation ...	May 2016	Police Force	nan	48508596	05d796c77702b...	Apr 2016	E05000128	Belsize	527315.26	184378.68	-0.165408
11	Possession of ...	1486721	St Pancras ...	nan	Under ...	Aug 2019	British Transpor...	Station	77245608	nan	Aug 2019	E05000143	St Pancras and ...	530079.71	183174.22	-0.12601
12	Burglary	960938	On or near ...	nan	Status update ...	Mar 2017	Police Force	nan	52937897	da582a168109e...	Nov 2016	E05000129	Bloomsbury	530182.87	182281.85	-0.124891
13	Other theft	968149	On or near Petr...	nan	Investigation ...	Dec 2015	Police Force	nan	45560924	8c71420dfdcee...	Dec 2015	E05000131	Cantelowes	529541.84	184590.92	-0.133277
14	Robbery	965362	On or near Que...	nan	Under ...	Apr 2018	Police Force	nan	64376568	8421b68d78559...	Apr 2018	E05000140	Kilburn	525463.27	184018.69	-0.192229
15	Violence and ...	967943	On or near ...	nan	Under ...	Mar 2017	Police Force	nan	55580264	adef0ec085796c...	Mar 2017	E05000128	Belsize	527508.87	184588.86	-0.162579
16	Theft from the ...	965122	On or near Petr...	nan	Under ...	Apr 2018	Police Force	nan	64336337	f374661faf4542...	Apr 2018	E05000136	Haverstock	528565.21	184290.67	-0.147425
17	Public order	960515	On or near ...	nan	Investigation ...	Sep 2015	Police Force	nan	43283864	29b4b7f0a8e2a...	Aug 2015	E05000129	Bloomsbury	530100.87	182168.92	-0.126114
18	Violence and ...	967999	On or near ...	nan	Investigation ...	Oct 2017	Police Force	nan	60694021	5416cc3cb8948...	Oct 2017	E05000139	Kentish Town	529350.21	185415.66	-0.135698
19	Theft from the ...	965104	On or near Edis ...	nan	Status update ...	Aug 2017	Police Force	nan	56257918	4e70cb3468ee7...	Apr 2017	E05000130	Camden Town ...	528256.24	183963.73	-0.151997
20	Burglary	965391	On or near ...	nan	Under ...	Mar 2018	Police Force	nan	63904234	3dc9881558b7c...	Mar 2018	E05000145	West Hampstead	525319.31	184464.78	-0.194145

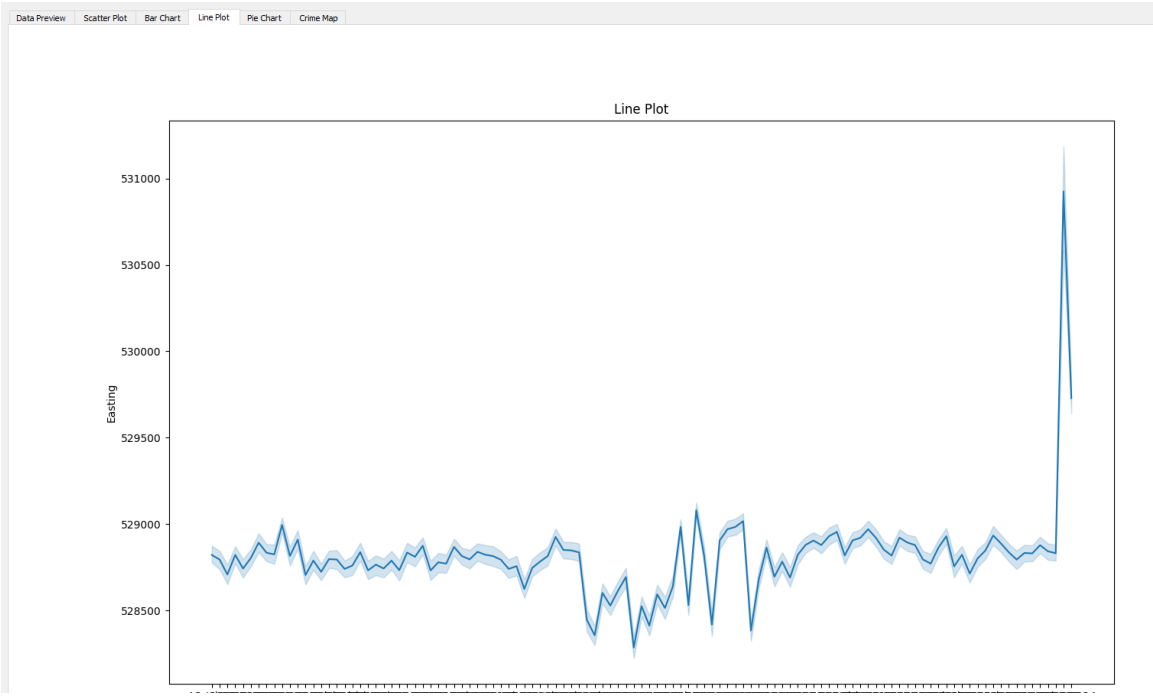
3. Tab for bar chart:
    - Displays bar chart from the data

```
class BarChartTab(QWidget):
    def __init__(self, data, x_column, y_column):
        super().__init__()
        self.data = data
        self.x_column = x_column
        self.y_column = y_column
        self.init_ui()

    def init_ui(self):
        try:
            self.fig, self.ax = plt.subplots()
            sns.barplot(x=self.x_column, y=self.y_column, data=self.data)
            self.ax.set_xlabel(self.x_column)
            self.ax.set_ylabel(self.y_column)
            self.ax.set_title("Bar Chart")
            self.layout = QVBoxLayout(self)
            self.layout.addWidget(FigureCanvas(self.fig))
            self.setLayout(self.layout)
        except Exception as e:
            error_label = QLabel(f"Failed to create bar chart: {str(e)}")
            layout = QVBoxLayout(self)
            layout.addWidget(error_label)
            self.setLayout(layout)
```



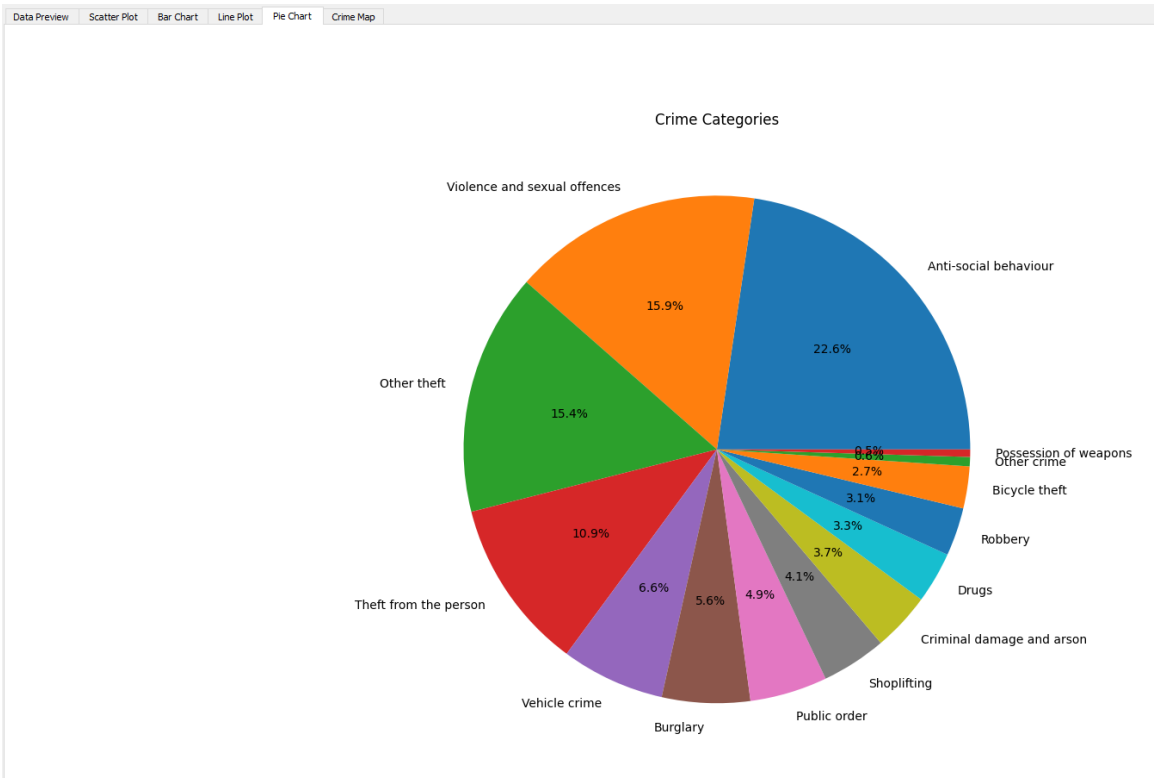
4. Class for line plot:
    - Displayiong line chart:

```python
class LinePlotTab(QWidget):
    def __init__(self, data, x_column, y_column):
        super().__init__()
        self.data = data
        self.x_column = x_column
        self.y_column = y_column
        self.init_ui()

    def init_ui(self):
        try:
            self.fig, self.ax = plt.subplots()
            sns.lineplot(x=self.x_column, y=self.y_column, data=self.data)
            self.ax.set_xlabel(self.x_column)
            self.ax.set_ylabel(self.y_column)
            self.ax.set_title("Line Plot")
            self.layout = QVBoxLayout(self)
            self.layout.addWidget(FigureCanvas(self.fig))
            self.setLayout(self.layout)
        except Exception as e:
            error_label = QLabel(f"Failed to create line plot: {str(e)}")
            layout = QVBoxLayout(self)
            layout.addWidget(error_label)
            self.setLayout(layout)
```



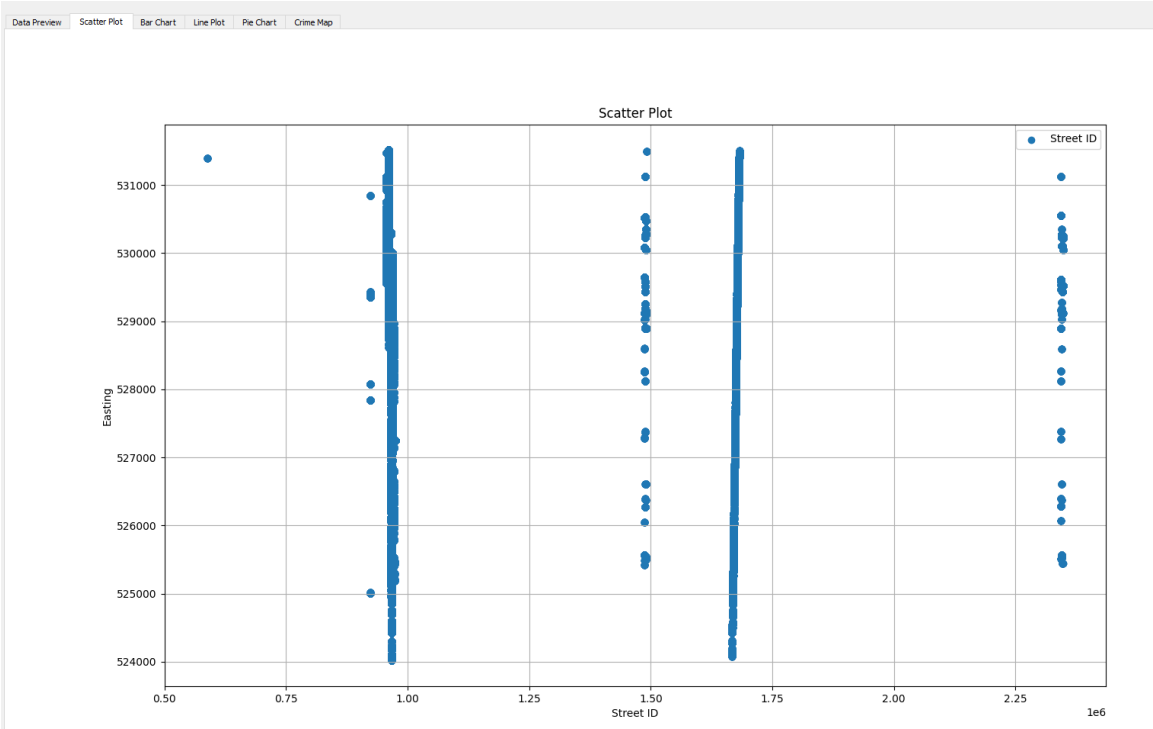
5. Pie chart class:
    - Class to plot the pie chart

```python
class PieChartTab(QWidget):
    def __init__(self, data):
        super().__init__()
        self.data = data
        self.init_ui()

    def init_ui(self):
        try:
            self.fig, self.ax = plt.subplots()
            category_counts = self.data['Category'].value_counts()
            self.ax.pie(category_counts.values, labels=category_counts.index, autopct='%1.1f%%')
            self.ax.set_title("Crime Categories")
            self.layout = QVBoxLayout(self)
            self.layout.addWidget(FigureCanvas(self.fig))
            self.setLayout(self.layout)
        except Exception as e:
            error_label = QLabel(f"Failed to create pie chart: {str(e)}")
            layout = QVBoxLayout(self)
            layout.addWidget(error_label)
            self.setLayout(layout)
```



6. Class for scatter plot
    - Its for displaying scatter plot

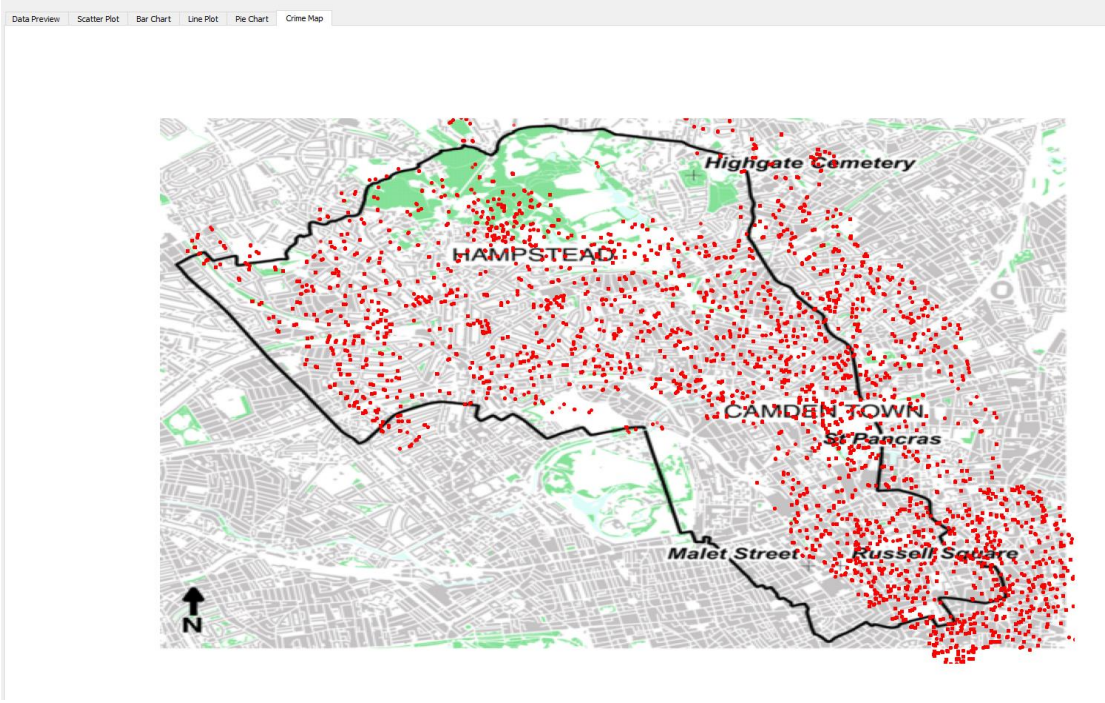
```python
class ScatterPlotTab(QWidget):
    def __init__(self, data, x_column, y_column):
        super().__init__()
        self.data = data
        self.x_column = x_column
        self.y_column = y_column
        self.init_ui()

    def init_ui(self):
        try:
            self.fig, self.ax = plt.subplots()
            self.ax.scatter(self.data[self.x_column], self.data[self.y_column], marker='o', label=self.x_column)
            plt.title("Scatter Plot")
            plt.grid(True)
            plt.legend()
            self.ax.set_xlabel(self.x_column)
            self.ax.set_ylabel(self.y_column)
            self.ax.set_title("Scatter Plot")
            self.layout = QVBoxLayout(self)
            self.layout.addWidget(FigureCanvas(self.fig))
            self.setLayout(self.layout)
        except Exception as e:
            error_label = QLabel(f"Failed to create scatter plot: {str(e)}")
            layout = QVBoxLayout(self)
            layout.addWidget(error_label)
            self.setLayout(layout)
```



7. Map class:

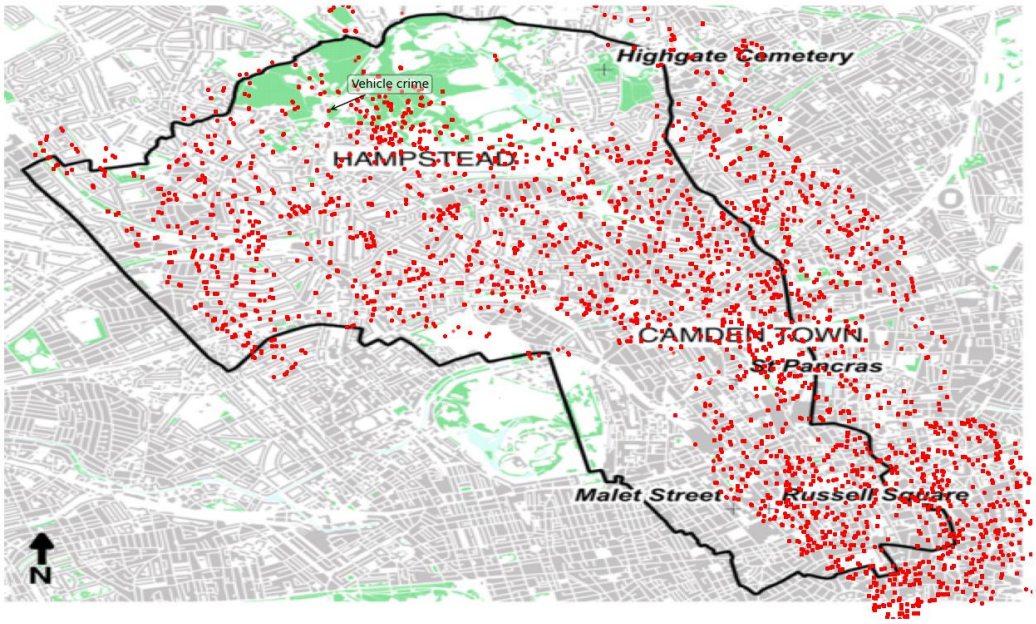
- Plots points on the map according to the longitude and latitude data from

```python
class MapTab(QWidget):
    def __init__(self, data):
        super().__init__()
        self.data = data
        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        try:
            self.fig, self.ax = plt.subplots(figsize=(10, 8))
            self.canvas = FigureCanvas(self.fig)
            layout.addWidget(self.canvas)

            img = plt.imread("mytry/A-map-of-the-London-Borough-of-Camden-and-location-in-UK-right-Field-locations-are.p
            self.ax.imshow(img, extent=[0, 1, 0, 1], aspect='auto')

            lat_min, lat_max = self.data['Latitude'].min(), self.data['Latitude'].max()
            lon_min, lon_max = self.data['Longitude'].min(), self.data['Longitude'].max()

            x_adjust = 0.03
            y_adjust = 1.01
            x_scale = 1.05
            y_scale = -1.1

            x = (self.data['Longitude'] - lon_min) / (lon_max - lon_min)
            y = (self.data['Latitude'] - lat_min) / (lat_max - lat_min)

            x = x * x_scale + x_adjust
            y = y * y_scale + y_adjust
            y = 1 - y

            self.scatter = self.ax.scatter(x, y, c='red', alpha=0.5, s=5)
            self.ax.set_xlim(0, 1)
            self.ax.set_ylim(0, 1)
            self.ax.axis('off')

            self.annot = self.ax.annotate("", xy=(0,0), xytext=(20,20),
                                          textcoords="offset points",
                                          bbox=dict(boxstyle="round", fc="w"),
                                          arrowprops=dict(arrowstyle="->"))
            self.annot.set_visible(False)
            self.fig.canvas.mpl_connect("motion_notify_event", self.hover)
        except Exception as e:
            error_label = QLabel(f"Failed to load map: {str(e)}")
            layout.addWidget(error_label)
```

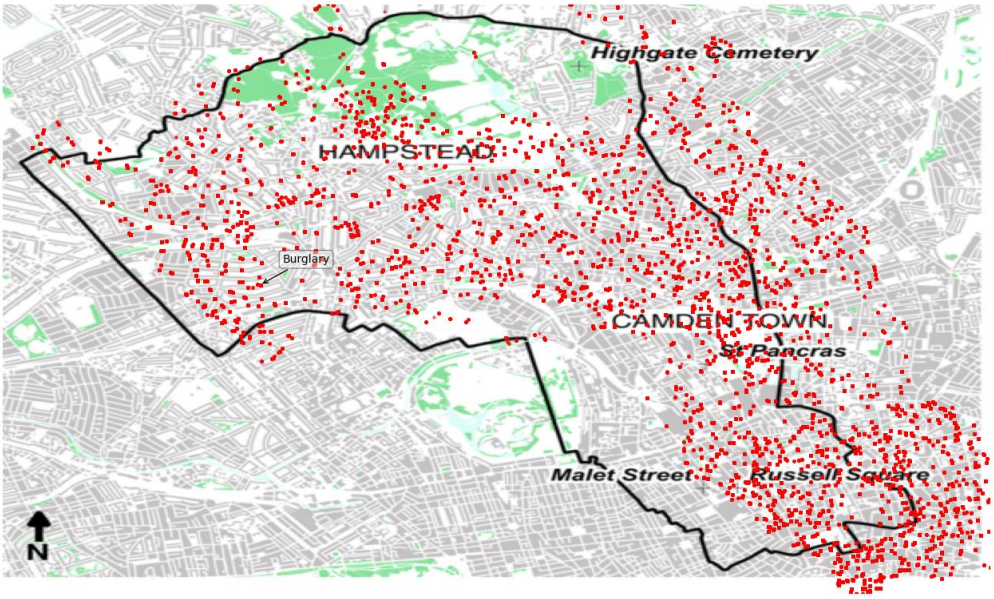


- When you hover over a point on the map a pop uo will show what type of crime there is in that area.

```
def hover(self, event):
    vis = self.annot.get_visible()
    if event.inaxes == self.ax:
        cont, ind = self.scatter.contains(event)
        if cont:
            self.update_annot(ind)
            self.annot.set_visible(True)
            self.fig.canvas.draw_idle()
        else:
            if vis:
                self.annot.set_visible(False)
                self.fig.canvas.draw_idle()

def update_annot(self, ind):
    pos = self.scatter.get_offsets()[ind["ind"][0]]
    self.annot.xy = pos
    text = f"{self.data.iloc[ind['ind'][0]]['Category']}"
    self.annot.set_text(text)
    self.annot.get_bbox_patch().set_alpha(0.4)
```



-



8. Initialises and runs the main program:

```python
if __name__ == "__main__":
    try:
        app = QApplication(sys.argv)
        window = MainWindow()
        window.show()
        sys.exit(app.exec_())
    except Exception as e:
        print(f"Failed to start application: {str(e)}")
```

9. Unit testing:
    - We created a new file test_data_processor and data processor to do unit tests

```python
import pandas as pd

def load_data(file_path):
    try:
        data = pd.read_csv(file_path, low_memory=False)
        data['Latitude'] = pd.to_numeric(data['Latitude'], errors='coerce')
        data['Longitude'] = pd.to_numeric(data['Longitude'], errors='coerce')
        data = data.dropna(subset=['Latitude', 'Longitude'])
        return data
    except Exception as e:
        raise Exception(f"Failed to load data: {str(e)}")

def get_data_summary(data):
    try:
        summary = {
            'num_entries': len(data),
            'lat_range': (data['Latitude'].min(), data['Latitude'].max()),
            'lon_range': (data['Longitude'].min(), data['Longitude'].max())
        }
        return summary
    except Exception as e:
        raise Exception(f"Failed to summarize data: {str(e)}")
```

```python
import unittest
from unittest.mock import patch, mock_open
import pandas as pd
from data_processor import load_data, get_data_summary

class TestDataProcessor(unittest.TestCase):
    @patch("builtins.open", new_callable=mock_open, read_data="Latitude,Longitude\n51.5,-0.12\n51.51,-0.
    @patch("pandas.read_csv")
    def test_load_data(self, mock_read_csv, mock_file):
        mock_read_csv.return_value = pd.DataFrame({
            'Latitude': [51.5, 51.51],
            'Longitude': [-0.12, -0.13]
        })

        data = load_data("data\On_Street_Crime_In_Camden.csv")
        self.assertEqual(len(data), 2)
        self.assertIn('Latitude', data.columns)
        self.assertIn('Longitude', data.columns)

    @patch("pandas.read_csv", side_effect=Exception("File not found"))
    def test_load_data_failure(self, mock_read_csv):
        with self.assertRaises(Exception) as context:
            load_data("data\On_Street_Crime_In_Camden.csv")
        self.assertTrue("Failed to load data" in str(context.exception))

    def test_get_data_summary(self):
        data = pd.DataFrame({
            'Latitude': [51.5, 51.51],
            'Longitude': [-0.12, -0.13]
        })

        summary = get_data_summary(data)
        self.assertEqual(summary['num_entries'], 2)
        self.assertEqual(summary['lat_range'], (51.5, 51.51))
        self.assertEqual(summary['lon_range'], (-0.13, -0.12))

    def test_get_data_summary_failure(self):
        data = pd.DataFrame({
            'Latitude': ['invalid', 51.51],
            'Longitude': [-0.12, 'invalid']
        })

        with self.assertRaises(Exception) as context:
            get_data_summary(data)
        self.assertTrue("Failed to summarize data" in str(context.exception))

if __name__ == "__main__":
    unittest.main()
```

-
```
  load_data("data\On_Street_Crime_In_Camden.csv")
....
----------------------------------------------------------------------
Ran 4 tests in 0.008s

OK
```

-
```
  load_data("data\On_Street_Crime_In_Camden.csv")
....
----------------------------------------------------
Ran 4 tests in 0.011s

OK
```

GUI usage has been optimized to give the user an optimal experience and also to show representations of the data in the csv file to serve understandable visuals of the data to be used for improving and defending against crime which increases the quality of life in Camden.

Error handeling to make sure of when the user has done a specific mistake it is given and or shown to the user via a message to make sure the gui stays functioning for the user.

Data processing is when specific data is chosen to be represented in different tabs of the program depending on relevance and especially the map would be usefull for data analysis and maybe crime prediction and or prevention.

Unit testing has been incorporated , to make sure that the data processing and retrieval is working.

Conclusion:

This application shows data in many ways aiding the benefactors from this program to improve the way of life for many people.

For future work this program can be improved by using more data sets and visualizing them properly. Add more interaction features that allow the user to better see and understand the data.make the data processing faster or better especially if we want to use bigger csv files.Upload the program to the internet maybe using a website by using flask and a proper hosting service for it or maybe a mobile application to make it usable on the go for maybe for more mainstream users.In the future we could be able to use machine learning techniques to maybe predict crime in the area to make and allocate resources according to the predicted data.

Github link: https://github.com/Hadialishibli/advanced-programming-assesment.git

Data set link required to run the application:
https://data.europa.eu/data/datasets/on-street-crime-in-camden?locale=en