Mapping crimes committed in France

Les Incodables : Eva Canzerini, Anthony Wensierski, Jeanne Robart

Interest of subject

- Crime Index in France : 55,26
- Informing institutions about the location of crimes
- Find out which “Cours d’appel” hear the most felony appeals.

Table of contents

INCODABLES TIME

Stage 1 : The Scrapping

Scrapping

Scrapping refers to the automated process of extracting data from websites or online sources for various purposes such as analysis, research, or aggregation.

Scraping site selection

To scrap, we used the Cour de Cassation database to collect decisions handed down by the Criminal Division of the Cour de Cassation from 2020 to 2024.

JUDILIBRE
COUR DE CASSATION

Web page inspection

Identification of the location in the page structure of the button providing access to the decision.

Collecting decisions

To collecting decisions, it is important to creating a list to store decision URLs and then creating a loop to iterate through the pages of search results

Recreating the links of 1500 decisions

After analyzing the decision URLs, we recreate the links that lead directly to the decisions we're interested in.

```
# I.1. Retrieve the URLs of the pages containing the text of each decision

# Installing and importing necessary libraries
pip install requests
pip install bs4
import requests
from bs4 import BeautifulSoup

# Creating a list to store decision URLs
list1 = []
```

```
# Creating a loop to iterate through the pages of search results
for i in range(501):
  # Constructing URL with page number to iterate through every page number from 1 to 500
  url = "https://www.courdecassation.fr/recherche-judilibre?search_api_full%20%20%20%20...:%20text=&op=Rechercher&date_du=2020-01-0
  webpage = requests.get(url)
  # Parsing the webpage content
  soup = BeautifulSoup(webpage.content)
  # Finding all decision items "Lire" on the page, containing decision URLs
  aas = soup.find_all("div", class_="decision-item")
  # Looping through each decision item
  for a in aas:
    # Extracting decision URL
    href = a.find("a").get("href")
    sublist = [href]
    # Appending decision URL to the list
    list1.append(sublist)
```

+ str(i)

Scrapping

Scrapping refers to the automated process of extracting data from websites or online sources for various purposes such as analysis, research, or aggregation.

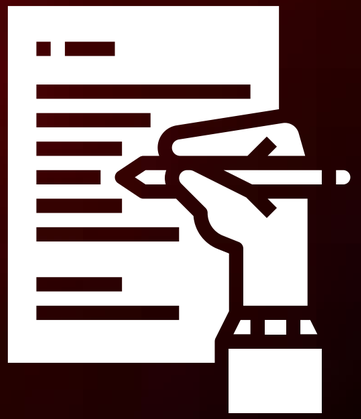
Extract the decision text

First, it’s important to create a list to store decision texts then looping through decision URLs to extract decision text

Checking if the decision exist

We added this condition because we had a bug in the code, we assumed there was a problem with the text of a decision.

Extraction of decision text



```
#I. 2. Retrieve the text of each decision

# Installing and importing necessary libraries
pip install regex
import regex as re

# Creating a list to store decision texts
decision_texts = []

# Looping through decision URLs to extract decision texts
for sublist in list1:
  href = sublist[0]
  decision_url = "https://www.courdecassation.fr" + href
  decision_page = requests.get(decision_url)
  decision_soup = BeautifulSoup(decision_page.content, "html.parser")
  decision_content = decision_soup.find("div", class_="decision-content decision-content--main")
  # Checking if decision content exists
  # We added this condition because we had a bug in the code, we assumed there was a problem with the text of a decision.
  if decision_content:
    # Extracting decision text
    decision_text = decision_content.getText()
    decision_texts.append(decision_text)
  else:
    print(f"Decision content not found for URL: {decision_url}")
```

Stage 2 : The Creation of the CSV files

For creating a CSV file with certain informations about the decisions, we have :

- imported the necessary library

```
import csv
```

- created a list of crimes, a list of Courts and a list of sexes (“Monsieur” ou “Madame”)

List the crimes to be investigated

ABUSE OF TRUST

KIDNAPPING

CRIMINAL CONSPIRACY

MURDER

VOYEURISM

CONSPIRACY

SEXUAL ASSAULT

PIMPIMG

FORGERY AND FORGERY REGISTRATION

ARMED ROBBERY

MISUSE OF CORPORATE ASSETS

SEXUAL EXHIBITION

VIOLENCE RESULTING IN UNINTENTIONAL DEATH

MONEY LAUNDERING

CORRUPTION AND INFLUENCE PEDDLING

ABUSE OF WEAKNESS

ROBBERY

TERRORISM

ENDANGERING THE LIFE OF OTHERS

RAPE

USURPATION

THREAT TO LIFE

CONCEALMENT

TORTURE

RACKETEERING

POISONING

EMBEZZLEMENT

SWINDLING

BANKRUPTCY

USURPATION

ILLEGAL CONFLICTS OF INTEREST

UNLAWFUL TRESPASSING

BANKRUPTCY

INSIDER TRADING

Appeal court location

Versailles

Metz

Besançon

Rennes

Agen

Toulouse

Bastia

Nîmes

Rouen

Cayenne

Montpellier

Aix en Provence

Bordeaux

Orléans

Limoges

Nancy

Grenoble

Reims

Saint-Pierre-et-Miquelon

Douai

Basse-Terre

Colmar

Lyon

Dijon

Pau

Paris

Amiens

Papeete

Chambéry

Fort-de-France

St Denis de la Réunion

Bourges

Angers

Poitiers

Caen

Nouméa

Riom

Then, we have :

- constructed regex patterns from the lists

```
pattern1 = "|".join(words1)
pattern2 = "|".join(words2)
pattern3 = "|".join(re.escape(word) for word in words3)
```

- and created a regex pattern for date

```
date_pattern = r"\b\d{1,2}\s*(?:janvier|février|mars|avril|mai|juin|juillet|août|septembre|octobre|novembre|décembre)\s*\d{4}\b"
```

After that, we have :

- created a list to store results, looped through decision texts to find matches of every crime, Court, and sex and checked if all matches are found before retrieving informations

```
# Creating a list to store results
results = []

# Looping through decision texts to find matches of every crime, Court, and sex
for decision_text in decision_texts:
  first_match = re.search(pattern1, decision_text)
  second_match = re.search(pattern2, decision_text)
  third_match = re.search(pattern3, decision_text)
  date_match = re.search(date_pattern, decision_text)
  # Checking if all matches are found before retrieving informations
  if first_match and second_match and third_match and date_match:
    results.append((first_match.group(), second_match.group(), third_match.group(), date_match.group()))
```

And so, we have :

- written the results to a CSV file, named the CSV file and its columns

```
# Writing results to a CSV file
if results:
  # Naming the CSV file
  csv_filename = "Informations_decisions.csv"
  with open(csv_filename, "w", newline="") as csvfile:
    writer = csv.writer(csvfile)
    # Naming the columns
    writer.writerow(["Crime", "Cour d'appel", "Sexe", "Date"])
    writer.writerows(results)
  print(f"Results saved to {csv_filename}")
else:
  print("No matches found.")
```

Informations_decisions

Crime	Cour d'appel	Sexe	Date
vol	cour d'appel de Douai	M. [	27 février 2024
viol	cour d'appel de Reims	M. [	27 février 2024
escroquerie	cour d'appel de Limoges	Mme [	27 février 2024
vol	cour d'appel de Versailles	M. [	27 février 2024
viol	cour d'appel de Reims	M. [	27 février 2024
blanchiment	cour d'appel de Lyon	M. [	27 février 2024
viol	cour d'appel de Nancy	M. [	27 février 2024
viol	cour d'appel d'Orléans	M. [	27 février 2024
escroquerie	cour d'appel de Paris	M. [	27 février 2024
viol	cour d'appel d'Amiens	M. [	27 février 2024
vol	cour d'appel de Dijon	Mme [	27 février 2024
assassinat	cour d'appel de Douai	M. [	27 février 2024
meurtre	cour d'appel de Cayenne	Mme [	27 février 2024
viol	cour d'appel de Riom	M. [	27 février 2024
vol	cour d'appel de Riom	M. [	27 février 2024
vol	cour d'appel de Bourges	Mme [	27 février 2024
vol	cour d'appel de Rennes	M. [	27 février 2024
meurtre	cour d'appel de Lyon	Mme [	27 février 2024
viol	cour d'appel de Lyon	M. [	27 février 2024
viol	cour d'appel de Lyon	M. [	27 février 2024
viol	cour d'appel de Fort-de-France	M. [	27 février 2024
viol	cour d'appel de Montpellier	M. [	15 février 2024
viol	cour d'appel de Rennes	Mme [	15 février 2024
viol	cour d'appel de Rouen	Mme [	15 février 2024
viol	cour d'appel de Montpellier	M. [	15 février 2024
terrorisme	cour d'appel de Limoges	M. [	15 février 2024
terrorisme	cour d'appel de Montpellier	M. [	15 février 2024
viol	cour d'appel de Versailles	M. [	15 février 2024
viol	cour d'appel de Paris	M. [	15 février 2024
viol	cour d'appel de Bordeaux	M. [	15 février 2024
viol	cour d'appel de Grenoble	Mme [	15 février 2024
viol	cour d'appel de Bourges	M. [	15 février 2024
viol	cour d'appel de Paris	Mme [	15 février 2024
viol	cour d'appel de Toulouse	M. [	15 février 2024
viol	cour d'appel de Montpellier	Mme [	15 février 2024
viol	cour d'appel de Montpellier	M. [	15 février 2024
viol	cour d'appel de Grenoble	M. [	15 février 2024
terrorisme	cour d'appel de Lyon	M. [	15 février 2024
viol	cour d'appel de Nîmes	M. [	15 février 2024
viol	cour d'appel de Paris	M. [	15 février 2024
vol	cour d'appel de Dijon	Mme [	15 février 2024
viol	cour d'appel de Versailles	M. [	15 février 2024
terrorisme	cour d'appel de Paris	Mme [	15 février 2024
terrorisme	cour d'appel de Paris	Mme [	15 février 2024
vol	cour d'appel de Nouméa	Mme [	15 février 2024
viol	cour d'appel de Riom	M. [	15 février 2024
viol	cour d'appel de Rennes	Mme [	15 février 2024
viol	cour d'appel de Paris	M. [	15 février 2024
viol	cour d'appel de Paris	M. [	15 février 2024
vol	cour d'appel de Paris	M. [	15 février 2024
viol	cour d'appel de Riom	M. [	15 février 2024
viol	cour d'appel de Rennes	Mme [	15 février 2024
viol	cour d'appel de Paris	M. [	15 février 2024
viol	cour d'appel de Nancy	Mme [	15 février 2024
viol	cour d'appel de Paris	M. [	15 février 2024
viol	cour d'appel de Paris	Mme [	15 février 2024
viol	cour d'appel de Douai	M. [	14 février 2024
viol	cour d'appel de Pau	Mme [	14 février 2024
viol	cour d'appel de Versailles	Mme [	14 février 2024

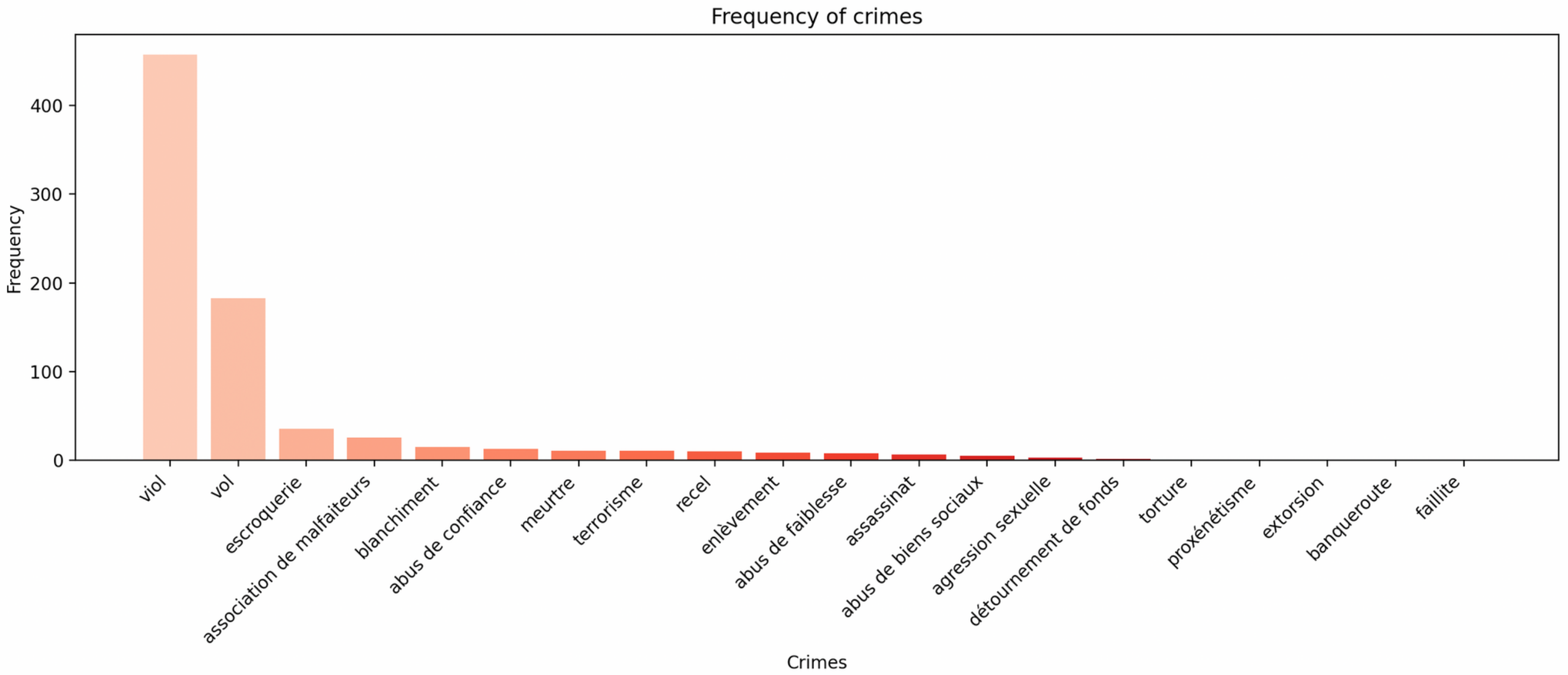
Stage 3 : The Creation of the graphics

Appetizers

```
# Installing and importing necessary libraries
pip install pandas
pip install matplotlib
pip install numpy
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

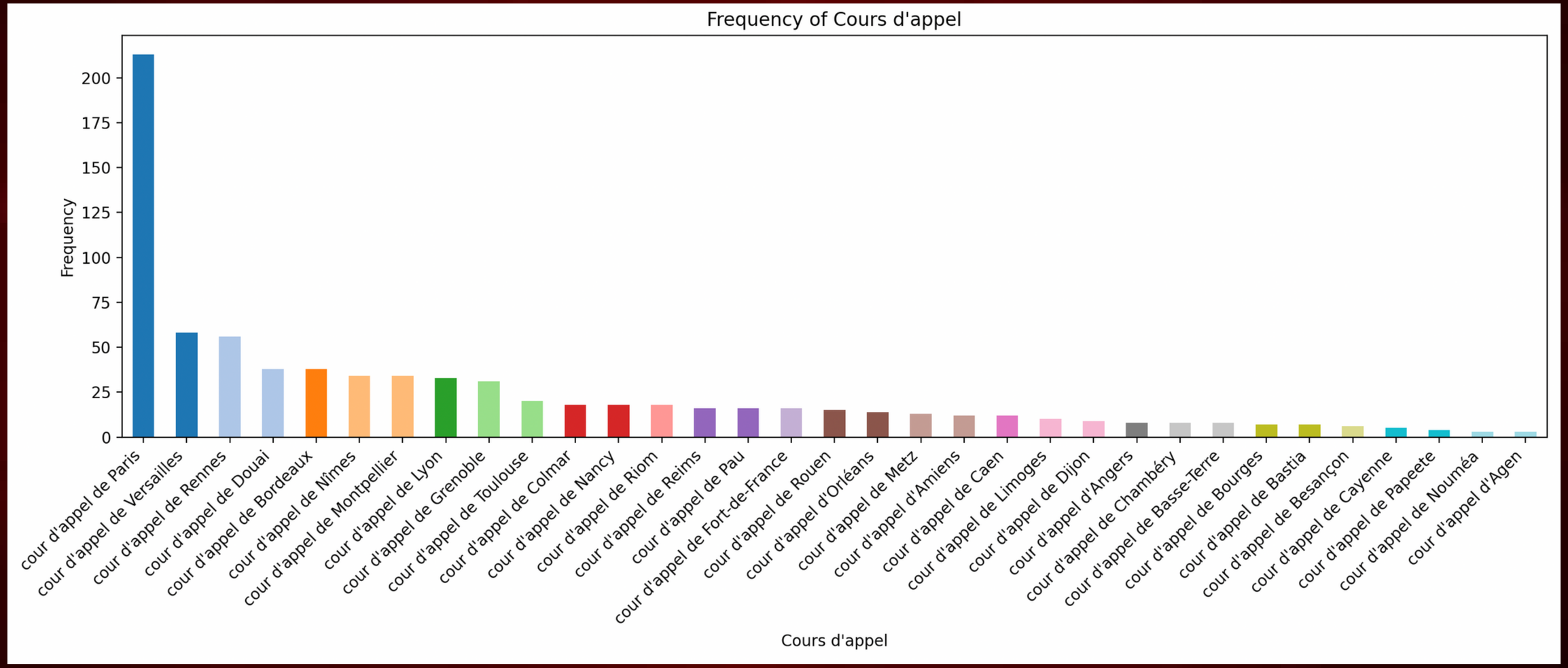
Graph about frequency of crimes

```
# Reading data from CSV file
df = pd.read_csv('Informations_decisions.csv')
# Couting frequency of crimes
crime_counts = df["Crime"].value_counts()
num_unique_values = len(crime_counts)
# Generating color range for the plot
colors = plt.cm.Reds(np.linspace(0.2, 1, num_unique_values))
# Creating bar plot for frequency of crimes
plt.figure(figsize=(15, 6))
plt.bar(crime_counts.index, crime_counts.values, color=colors)
plt.xlabel("Crimes")
plt.ylabel("Frequency")
plt.title("Frequency of crimes")
plt.xticks(rotation=45, ha='right')
plt.show()
```


Frequency of crimes
Frequency
0
100
200
300
400
viol
vol
escroquerie
association de malfaiteurs
blanchiment
abus de confiance
meurtre
terrorisme
recel
enlèvement
abus de faiblesse
assassinat
abus de biens sociaux
agression sexuelle
détournement de fonds
torture
proxénétisme
extorsion
banqueroute
faillite
Crimes

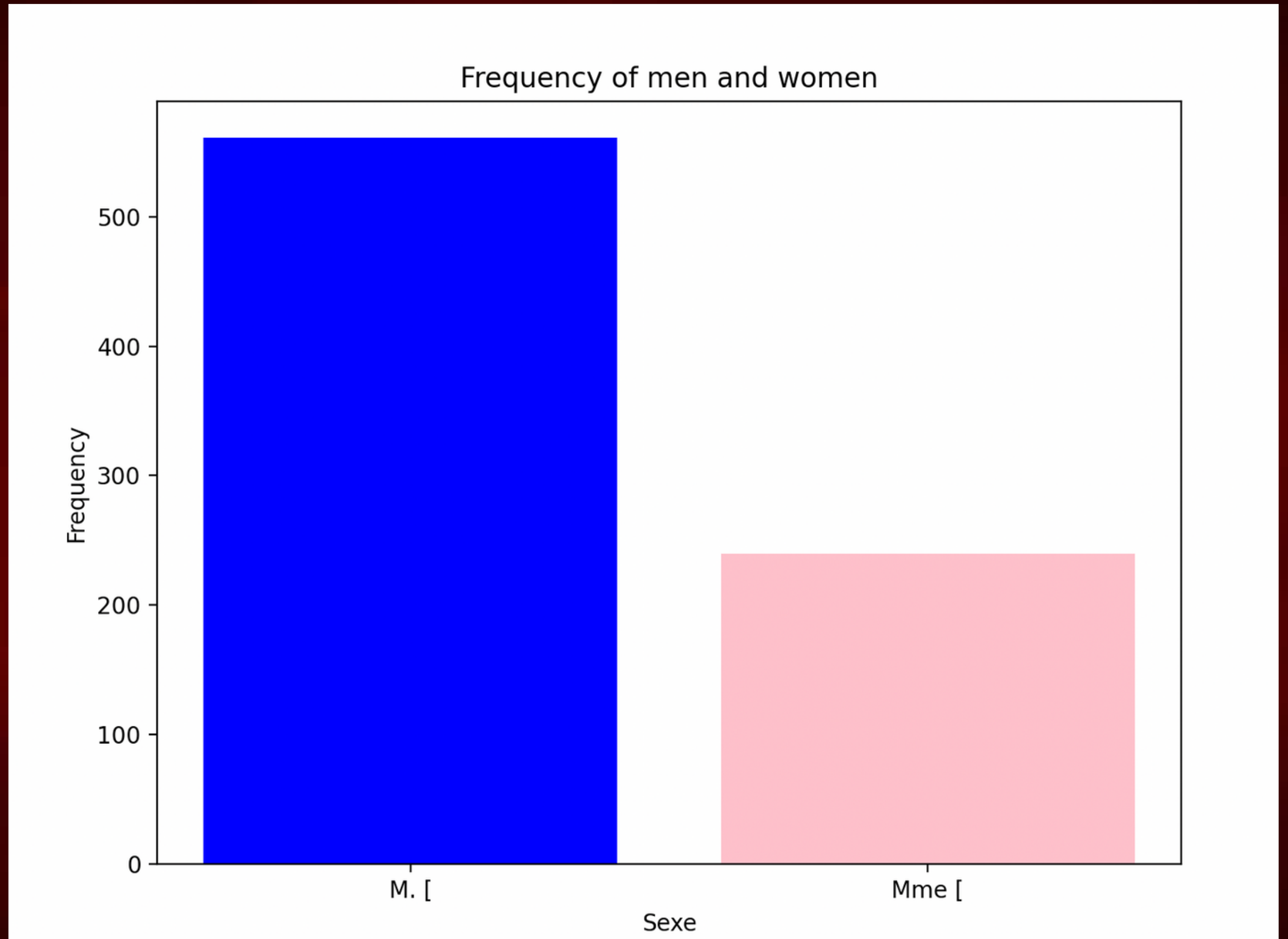
Graph about frequency of Cours d’appel

```
# Importing necessary libraries
from matplotlib.colors import LinearSegmentedColormap

# Reading data from CSV file
df = pd.read_csv('Informations_decisions.csv')

# Counting frequency of Cours d'appel
pattern_counts = df["Cour d'appel"].value_counts()
pattern_counts = df["Cour d'appel"].value_counts()

# Generating colors for the plot
num_colors = len(pattern_counts)
custom_colors = plt.cm.get_cmap("tab20", num_colors)

# Creating bar plot for frequency of Cours d'appel
plt.figure(figsize=(14, 6))
bar_plot = pattern_counts.plot(kind="bar", color=custom_colors.colors)
plt.title("Frequency of Cours d'appel")
plt.xlabel("Cours d'appel")
plt.ylabel('Frequency')
bar_plot.set_xticklabels(bar_plot.get_xticklabels(), rotation=45, ha='right')
plt.tight_layout()
plt.show()
```


Frequency of Cours d'appel
Frequency
0
25
50
75
100
125
150
175
200
Cours d'appel
cour d'appel de Paris
cour d'appel de Versailles
cour d'appel de Rennes
cour d'appel de Douai
cour d'appel de Bordeaux
cour d'appel de Nîmes
cour d'appel de Montpellier
cour d'appel de Lyon
cour d'appel de Grenoble
cour d'appel de Toulouse
cour d'appel de Colmar
cour d'appel de Nancy
cour d'appel de Riom
cour d'appel de Reims
cour d'appel de Pau
cour d'appel de Fort-de-France
cour d'appel de Rouen
cour d'appel d'Orléans
cour d'appel de Metz
cour d'appel d'Amiens
cour d'appel de Caen
cour d'appel de Limoges
cour d'appel de Dijon
cour d'appel d'Angers
cour d'appel de Chambéry
cour d'appel de Basse-Terre
cour d'appel de Bourges
cour d'appel de Bastia
cour d'appel de Besançon
cour d'appel de Cayenne
cour d'appel de Papeete
cour d'appel de Nouméa
cour d'appel d'Agen

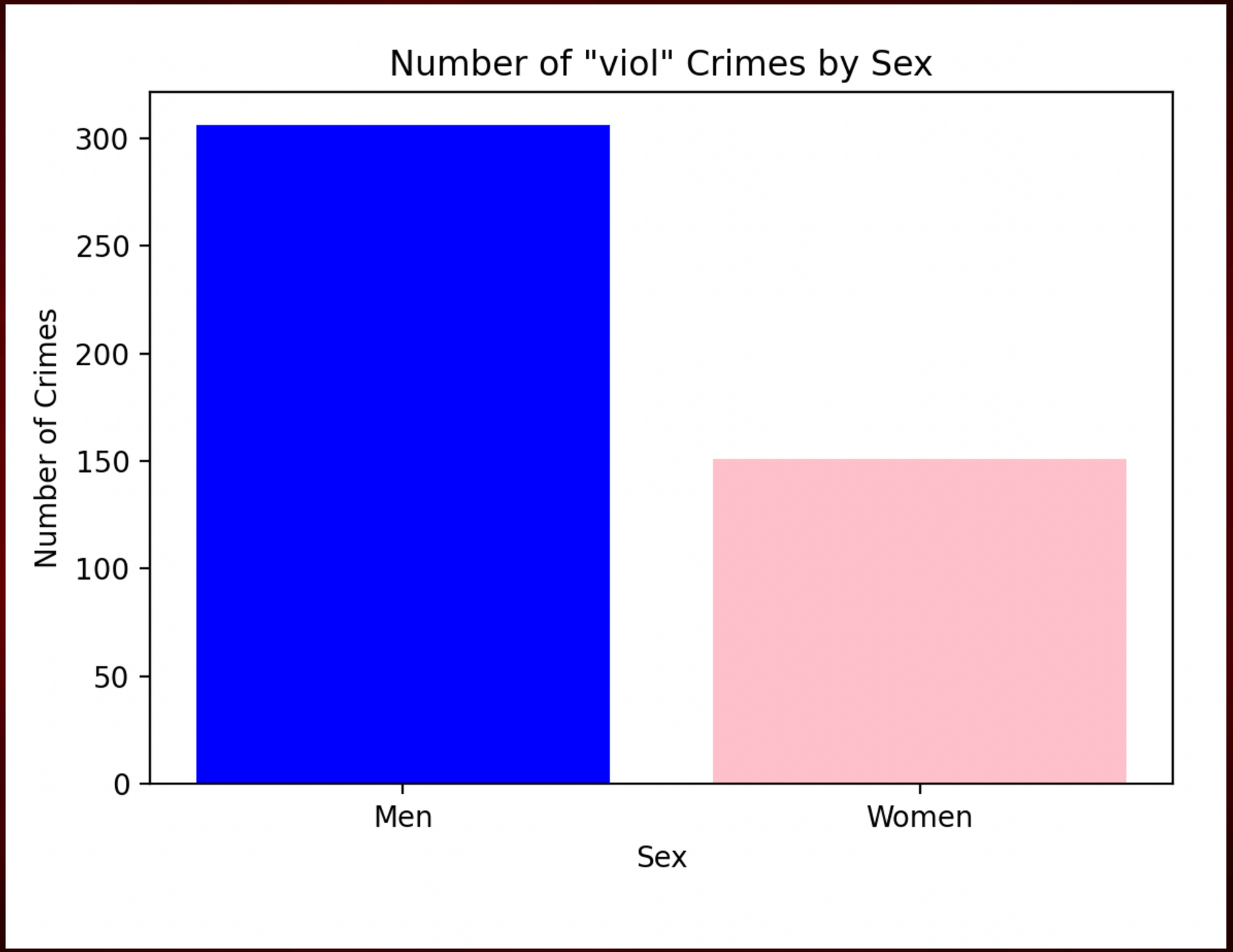
Graph about frequency of men & women

```
# Reading data from CSV
df = pd.read_csv('Informations_decisions.csv')

# Counting frequency of men and women
sex_counts = df["Sexe"].value_counts()

# Creating bar plot for frequency of men and women
plt.figure(figsize=(8, 6))
plt.bar(sex_counts.index, sex_counts.values, color=["blue", "pink"])
plt.xlabel("Sexe")
plt.ylabel("Frequency")
plt.title("Frequency of men and women")
plt.show()
```


Frequency of men and women
Frequency
500
400
300
200
100
0
M. [
Mme [
Sexe

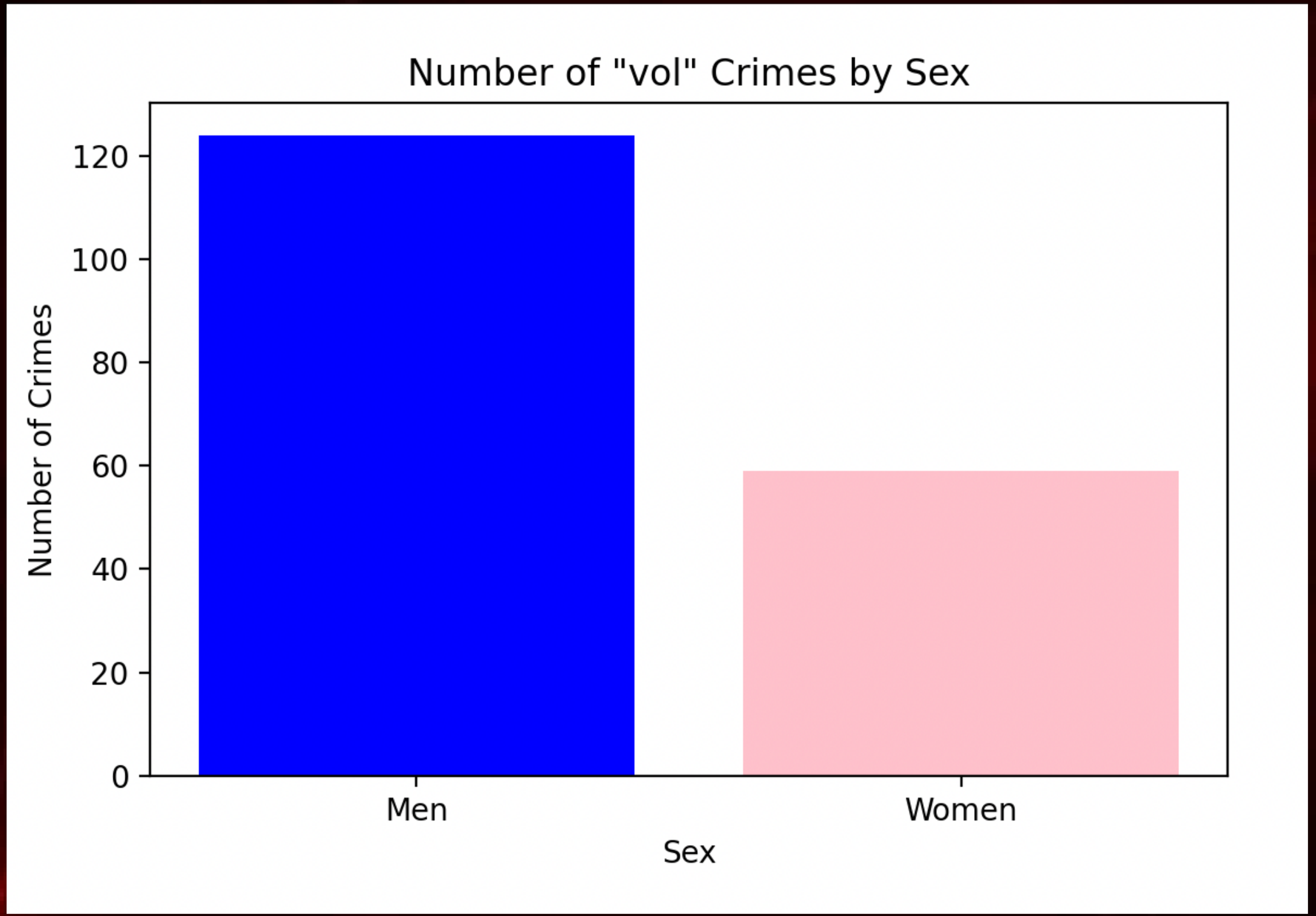
Graph about frequency of “viol” by sex

```
# Reading crime data from CSV
crime_data = pd.read_csv("Informations_decisions.csv")

# Filtering data for keeping only "viol" crimes
viol_data = crime_data[crime_data['Crime'] == 'viol']

# Counting the number of "viol" crime by men and by women
sex_counts = viol_data['Sexe'].value_counts()
men_count = sex_counts.get("M. [", 0)
women_count = sex_counts.get("Mme [", 0)

# Creating bar plot for number of "viol" crimes by sex
plt.bar(['Men', 'Women'], [men_count, women_count], color=['blue', 'pink'])
plt.xlabel('Sex')
plt.ylabel('Number of Crimes')
plt.title('Number of "viol" Crimes by Sex')
plt.show()
```


Number of "viol" Crimes by Sex
Number of Crimes
300
250
200
150
100
50
0
Men
Women
Sex

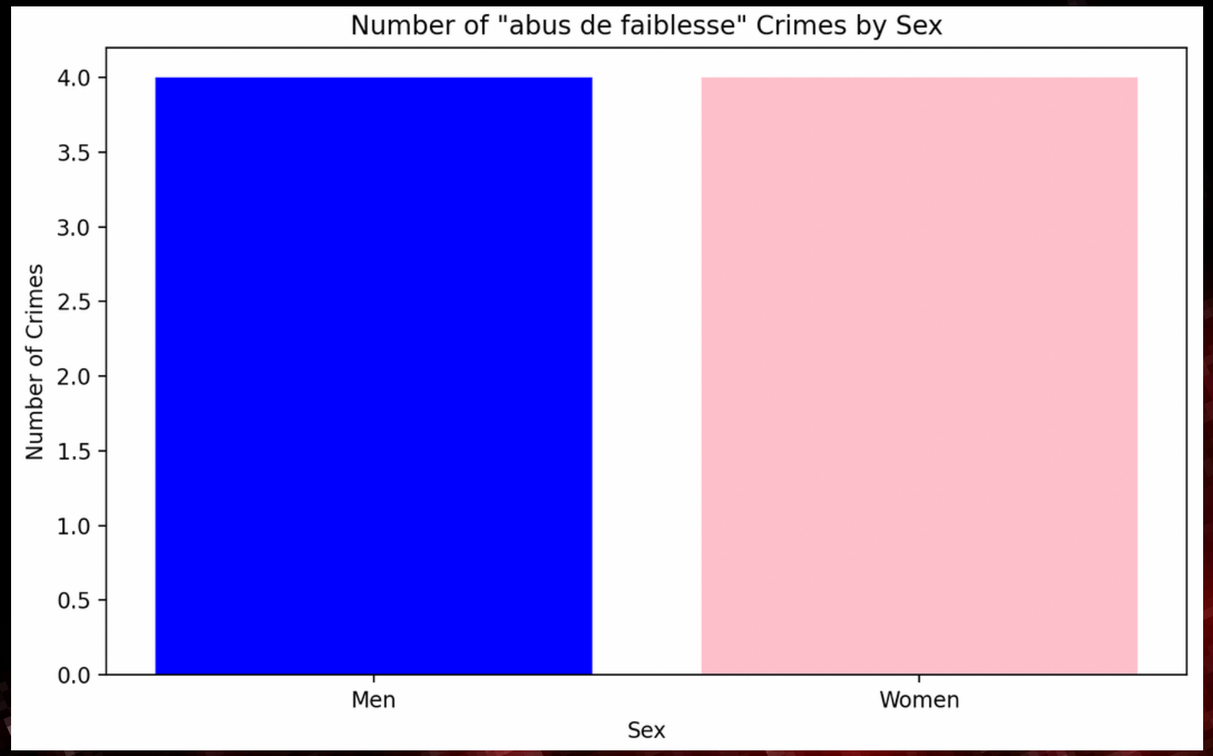
Graph about frequency of “vol” by sex

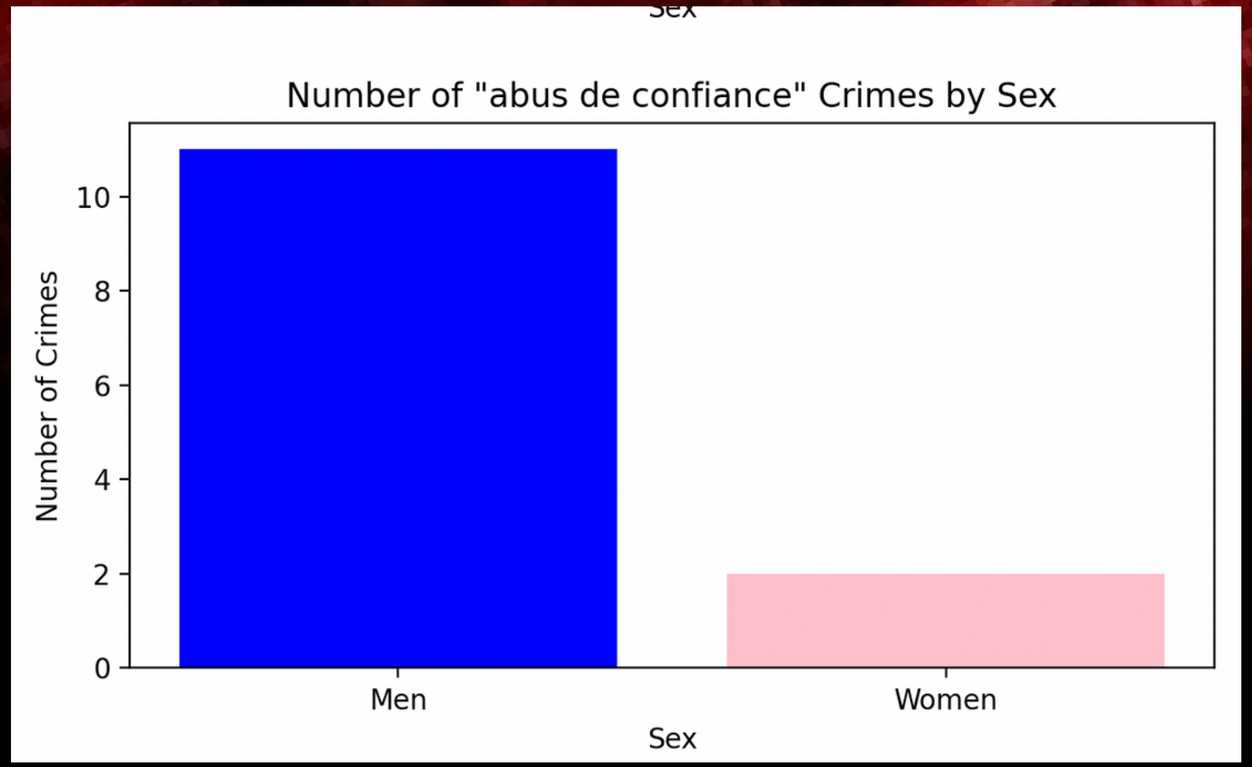
```
# Idem for "vol" crimes
crime_data = pd.read_csv("Informations_decisions.csv")
vol_data = crime_data[crime_data['Crime'] == 'vol']
sex_counts = vol_data['Sexe'].value_counts()
men_count = sex_counts.get("M. [", 0)
women_count = sex_counts.get("Mme [", 0)
plt.bar(['Men', 'Women'], [men_count, women_count], color=['blue', 'pink'])
plt.xlabel('Sex')
plt.ylabel('Number of Crimes')
plt.title('Number of "vol" Crimes by Sex')
plt.show()
```


Number of "vol" Crimes by Sex
Number of Crimes
120
100
80
60
40
20
0
Men
Women
Sex

Graph about frequency of “abus de confiance” and “abus de faiblesse” by sex

```
# Idem for "abus de confiance" and "abus de faiblesse" crimes
crime_data = pd.read_csv("Informations_decisions.csv")
abus_faiblesse_data = crime_data[crime_data['Crime'] == 'abus de faiblesse']
abus_faiblesse_sex_counts = abus_faiblesse_data['Sexe'].value_counts()
abus_faiblesse_men_count = abus_faiblesse_sex_counts.get("M. [", 0)
abus_faiblesse_women_count = abus_faiblesse_sex_counts.get("Mme [", 0)
abus_confiance_data = crime_data[crime_data['Crime'] == 'abus de confiance']
abus_confiance_sex_counts = abus_confiance_data['Sexe'].value_counts()
abus_confiance_men_count = abus_confiance_sex_counts.get("M. [", 0)
abus_confiance_women_count = abus_confiance_sex_counts.get("Mme [", 0)
fig, ax = plt.subplots(2, figsize=(8, 10))
ax[0].bar(['Men', 'Women'], [abus_faiblesse_men_count, abus_faiblesse_women_count], color=['blue', 'pink'])
ax[0].set_xlabel('Sex')
ax[0].set_ylabel('Number of Crimes')
ax[0].set_title('Number of "abus de faiblesse" Crimes by Sex')
ax[1].bar(['Men', 'Women'], [abus_confiance_men_count, abus_confiance_women_count], color=['blue', 'pink'])
ax[1].set_xlabel('Sex')
ax[1].set_ylabel('Number of Crimes')
ax[1].set_title('Number of "abus de confiance" Crimes by Sex')
plt.tight_layout()
plt.show()
```

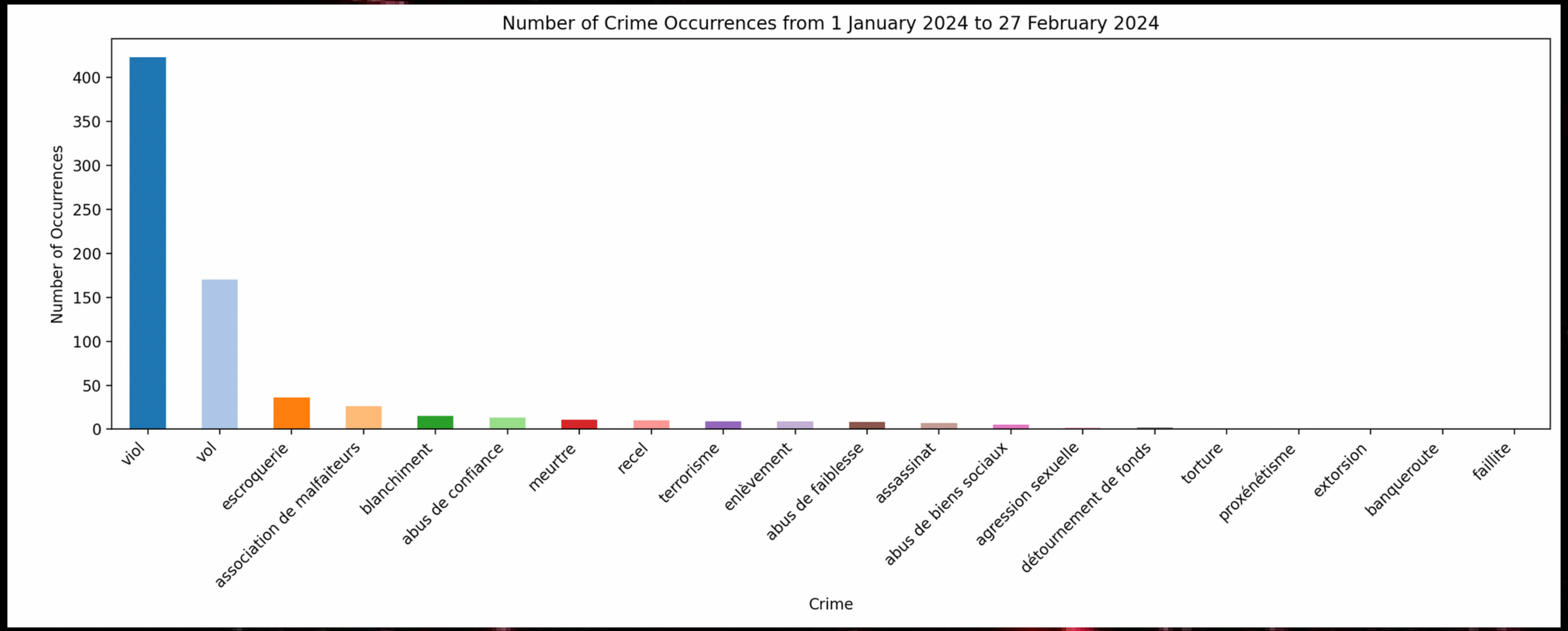

Number of "abus de faiblesse" Crimes by Sex
Number of Crimes
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Men
Women
Sex


Sex
Number of "abus de confiance" Crimes by Sex
Number of Crimes
0
2
4
6
8
10
Men
Women
Sex

Graph showing the number of crimes committed for each crime type from January 1st 2024 to February 27th 2024

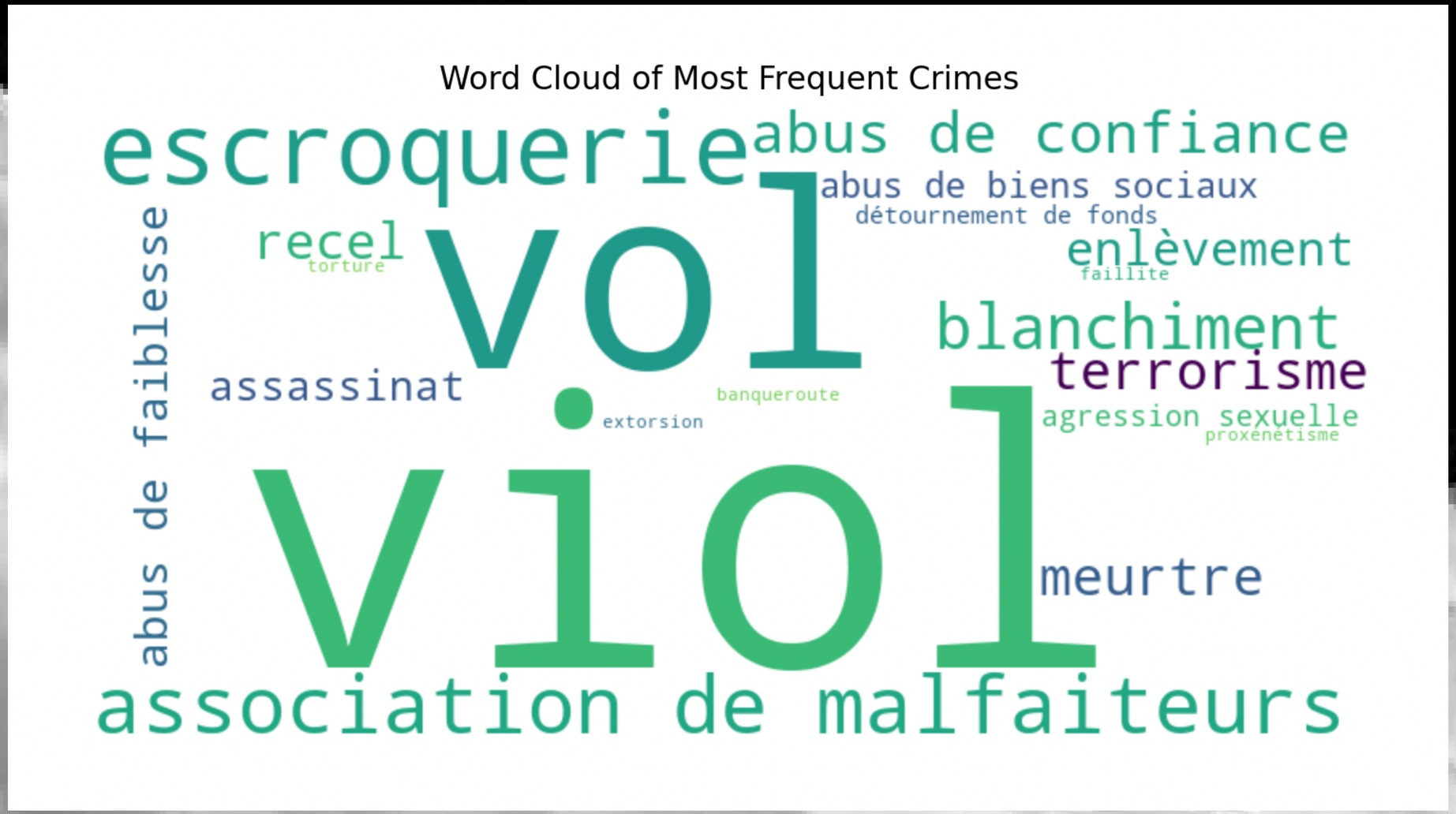
```
# Reading the csv file
df = pd.read_csv('Informations_decisions.csv')

# Defining a custom mapping for month names in French because the locale parameter is not available in the pd.to_datetime() function in the ver
month_mapping = {'janvier': 'January', 'février': 'February', 'mars': 'March', 'avril': 'April', 'mai': 'May', 'juin': 'June', 'juillet': 'July

# Replacing French month names with English month names
df['Date'] = df['Date'].replace(month_mapping, regex=True)

# Converting to datetime
df['Date'] = pd.to_datetime(df['Date'], format='%d %B %Y')

# Filting data for the specified date range
start_date = pd.to_datetime("2024-01-01")
end_date = pd.to_datetime("2024-02-27")
filtered_df = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]

# Grouping data by "Crime" and counting occurrences
crime_counts = filtered_df['Crime'].value_counts()
```

```
# Ploting the graph with colors
plt.figure(figsize=(15, 6))
colors = plt.cm.tab20(range(len(crime_counts)))
crime_counts.plot(kind='bar', color=colors)
plt.xlabel('Crime')
plt.ylabel('Number of Occurrences')
plt.title('Number of Crime Occurrences from 1 January 2024 to 27 February 2024')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```


Number of Crime Occurrences from 1 January 2024 to 27 February 2024
Number of Occurrences
0
50
100
150
200
250
300
350
400
viol
vol
escroquerie
association de malfaiteurs
blanchiment
abus de confiance
meurtre
recel
terrorisme
enlèvement
abus de faiblesse
assassinat
abus de biens sociaux
agression sexuelle
détournement de fonds
torture
proxénétisme
extorsion
banqueroute
faillite
Crime

THE CLOUD

```
# Installing and importing necessary libraries
import wordcloud
from wordcloud import WordCloud

# Generating word cloud from the 'Crime' column
df = pd.read_csv('Informations_decisions.csv')
wordcloud = WordCloud(width=800, height=400, background_color='white')
wordcloud.generate_from_frequencies(df['Crime'].value_counts())

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Most Frequent Crimes')
plt.show()
```


Word Cloud of Most Frequent Crimes
escroquerie
abus de confiance
abus de biens sociaux
détournement de fonds
recel
torture
vol
enlèvement
faillite
blanchiment
terrorisme
assassinat
banqueroute
extorsion
agression sexuelle
proxénétisme
abus de faiblesse
viol
meurtre
association de malfaiteurs

The masterclass

The map about the location of the crimes in France

```
# Installing and importing necessary libraries
pip install folium
import folium
import pandas as pd
from geopy.geocoders import Nominatim

# Reading crime data from CSV file
crime_data = pd.read_csv("Informations_decisions.csv")
# Initializing geolocator to localize every Cour d'Appel on a map
geolocator = Nominatim(user_agent="crime_mapping")
```

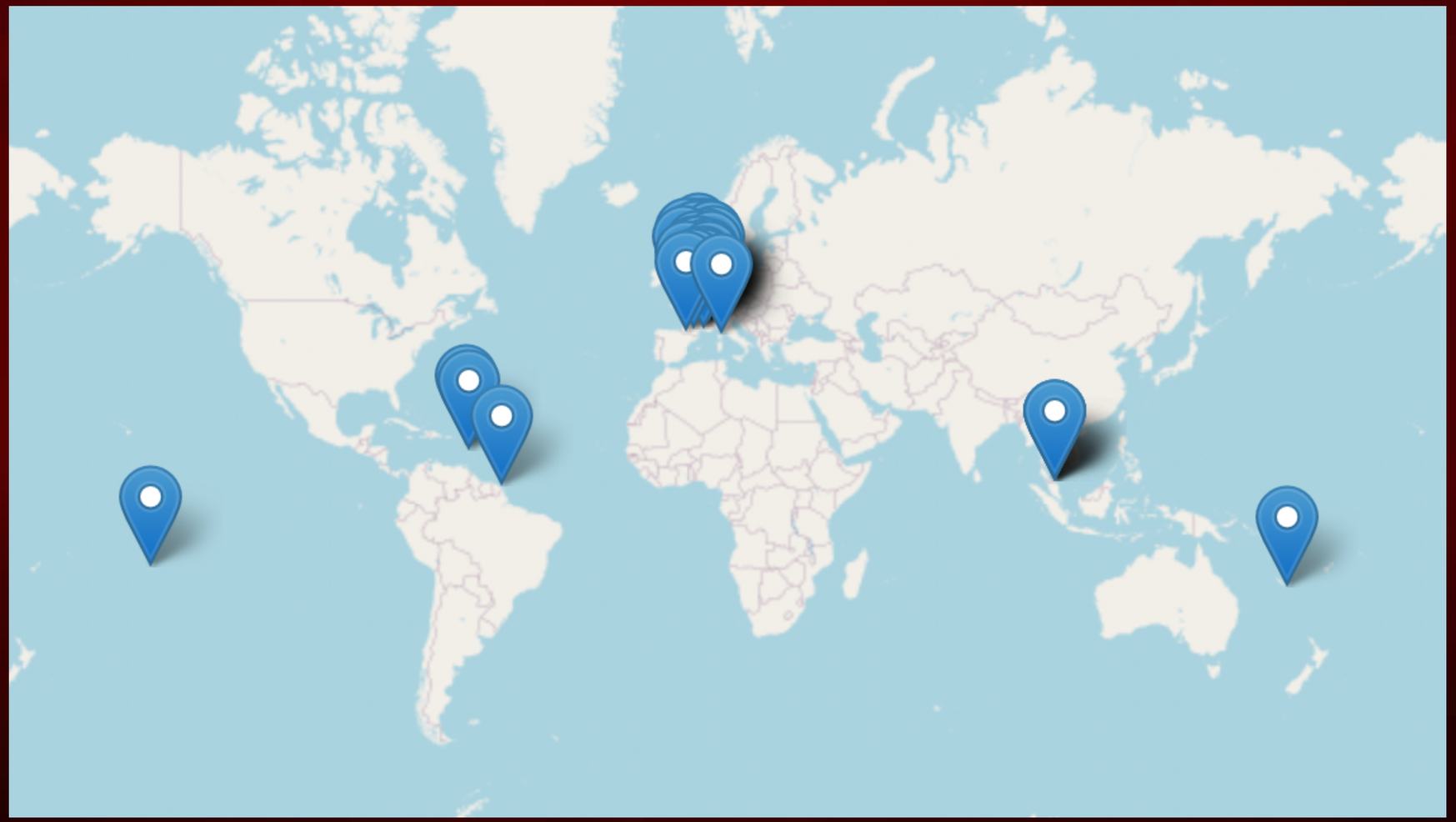
```python
# Adding latitude and longitude columns to the crime data
crime_data['latitude'], crime_data['longitude'] = zip(*crime_data['Cour d\'appel'].apply(geocode_location))
location_df = crime_data[['Cour d\'appel', 'latitude', 'longitude']]
location_df.columns = ['Location', 'Latitude', 'Longitude']

# Finding most common crime for each location
most_common_crime = crime_data.groupby('Cour d\'appel')['Crime'].agg(pd.Series.mode).reset_index()
most_common_crime.columns = ['Location', 'Most Common Crime']
location_df = pd.merge(location_df, most_common_crime, on='Location')
location_df = location_df.drop_duplicates(subset=['Location'])

# Saving location data (longitude and latitude) to CSV file
location_df.to_csv('locations_with_crime.csv', index=False)
print("The CSV file 'locations_with_crime.csv' has been created with location names, coordinates, and the most common crime.")

# Reading location data from CSV file
location_df = pd.read_csv('locations_with_crime.csv')

# Setting center coordinated for the map (on Paris, because it is were we are)
center_latitude = 48.8566
center_longitude = 2.3522
m = folium.Map(location=[center_latitude, center_longitude], zoom_start=10)

# Adding markers for each location on the map and adding to the name of the marker the most common crime for this location
for index, row in location_df.iterrows():
  folium.Marker([row['Latitude'], row['Longitude']], popup=f"{row['Location']}: {row['Most Common Crime']}").add_to(m)

# Saving the map as an HTML file
m.save('crime_map.html')
print("The HTML file 'crime_map.html' has been created.")
```

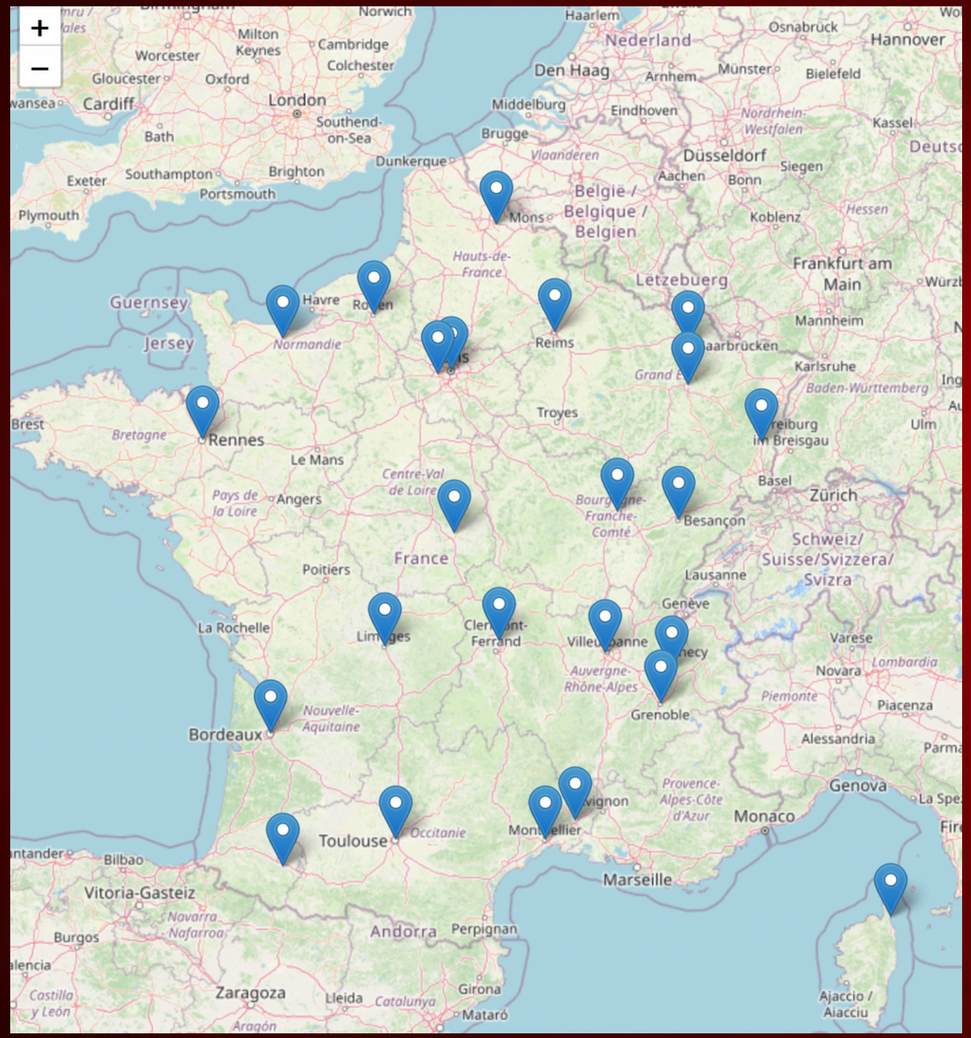
```python
# Creating a function to geocode location
# We isolated the location of the column "Cour d'appel" which contained not only the location but also "Cour d'appel de" or "Cour d'appel d'"
def geocode_location(location):
  try:
    parts = location.split()
    if parts[-1].startswith("d'"):
      location = parts[-2]
    elif parts[-1] == "de":
      location = parts[-2]
    else:
      location = parts[-1]

    location_info = geolocator.geocode(location)
    if location_info:
      return location_info.latitude, location_info.longitude
    else:
      return None, None
  except Exception as e:
    print(f"Error geocoding location '{location}': {e}")
    return None, None
```

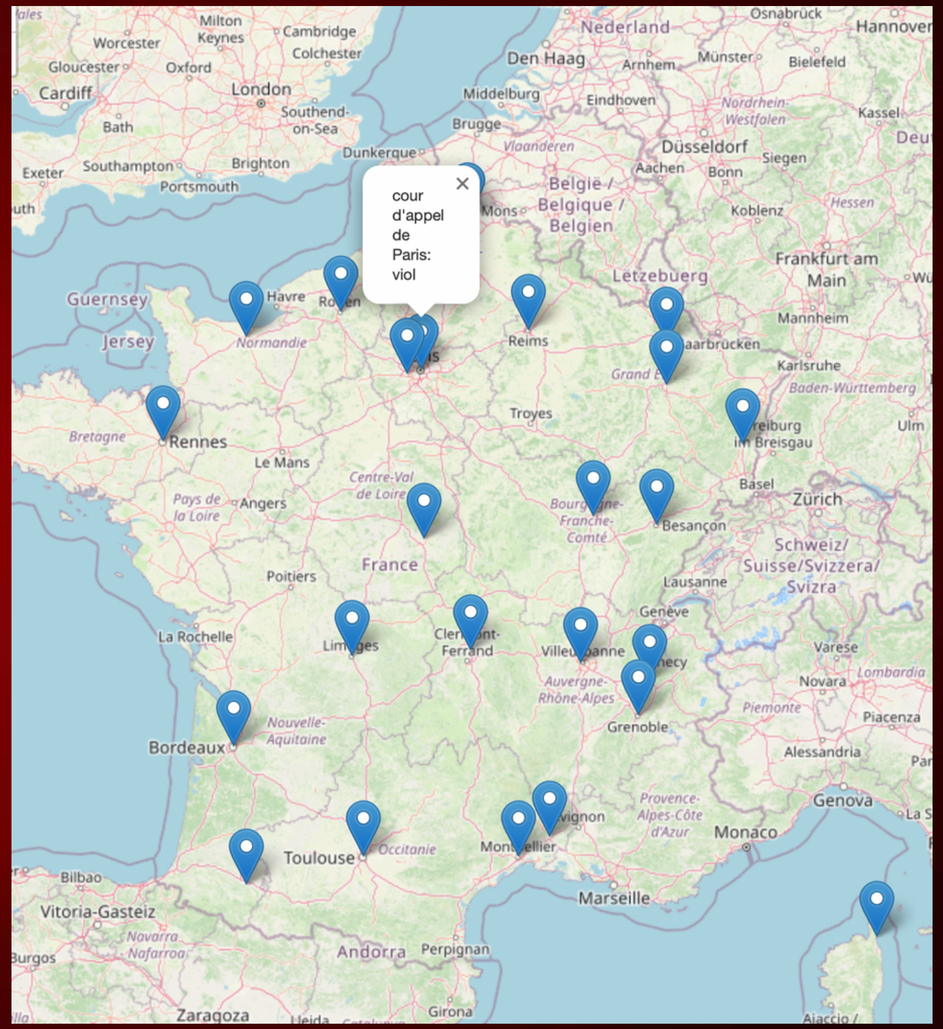
locations_with_crime

Location	Latitude	Longitude	Most Common Crime
cour d'appel de Douai	50.3675677	3.0804641	viol
cour d'appel de Reims	49.2577886	4.031926	viol
cour d'appel de Limoges	45.8354243	1.2644847	viol
cour d'appel de Versailles	48.8035403	2.1266886	viol
cour d'appel de Lyon	45.7578137	4.8320114	viol
cour d'appel de Nancy	48.6937223	6.1834097	viol
cour d'appel d'Orléans	6.129816	102.240212	viol
cour d'appel de Paris	48.8534951	2.3483915	viol
cour d'appel d'Amiens	6.129816	102.240212	viol
cour d'appel de Dijon	47.3215806	5.0414701	viol
cour d'appel de Cayenne	4.9371544	-52.3258736	vol
cour d'appel de Riom	45.893068	3.1136667	viol
cour d'appel de Bourges	47.0811658	2.399125	viol
cour d'appel de Rennes	48.1113387	-1.6800198	viol
cour d'appel de Fort-de-France	14.6027962	-61.0676724	viol
cour d'appel de Montpellier	43.6112422	3.8767337	viol
cour d'appel de Rouen	49.4404591	1.0939658	viol
cour d'appel de Bordeaux	44.841225	-0.5800364	viol
cour d'appel de Grenoble	45.1875602	5.7357819	viol
cour d'appel de Toulouse	43.6044622	1.4442469	viol
cour d'appel de Nîmes	43.8374249	4.3600687	viol
cour d'appel de Nouméa	-22.2745264	166.442419	viol
cour d'appel de Pau	43.2957547	-0.3685668	viol
cour d'appel de Colmar	48.0777517	7.3579641	viol
cour d'appel de Caen	49.1813403	-0.3635615	viol
cour d'appel de Papeete	-17.5373835	-149.5659964	viol
cour d'appel de Besançon	47.2380222	6.0243622	viol
cour d'appel d'Angers	6.129816	102.240212	viol
cour d'appel de Metz	49.1196964	6.1763552	vol
cour d'appel de Chambéry	45.5662672	5.9203636	viol
cour d'appel de Basse-Terre	16.0000778	-61.7333373	viol
cour d'appel de Bastia	42.6993979	9.4509187	viol
cour d'appel d'Agen	6.129816	102.240212	vol

```
In [69]: crime_data['latitude'], crime_data['longitude'] = zip(*crime_data['Cour d\'appel'].apply(ge
    ...: ocode_location))
Error geocoding location 'Lyon': HTTPSConnectionPool(host='nominatim.openstreetmap.org', port=443):
Max retries exceeded with url: /search?q=Lyon&format=json&limit=1 (Caused by ReadTimeoutError("HTTPS
ConnectionPool(host='nominatim.openstreetmap.org', port=443): Read timed out. (read timeout=1)"))

In [70]:
```




+
−
Nederland
Den Haag
Arnhem
Münster
Osnabrück
Hannover
Bielefeld
Eindhoven
Middelburg
Brugge
Dunkerque
Vlaanderen
Nordrhein-Westfalen
Kassel
Düsseldorf
Aachen
Bonn
Siegen
Koblenz
Hessen
België / Belgique / Belgien
Mons
Hauts-de-France
Frankfurt am Main
Mannheim
Letzebuerg
Saarbrücken
Karlsruhe
Baden-Württemberg
Grand Est
Troyes
Freiburg im Breisgau
Ulm
Basel
Zürich
Schweiz/ Suisse/Svizzera/ Svizra
Lausanne
Genève
Besançon
Bourgogne-Franche-Comté
Centre-Val de Loire
Pays de la Loire
Angers
Le Mans
Rennes
Bretagne
Brest
Guernsey
Jersey
Havre
Rouen
Normandie
Reims
France
Poitiers
La Rochelle
Limoges
Clermont-Ferrand
Villeurbanne
Auvergne-Rhône-Alpes
Grenoble
Varese
Lombardia
Novara
Piemonte
Piacenza
Alessandria
Parma
Nouvelle-Aquitaine
Bordeaux
Provence-Alpes-Côte d'Azur
Avignon
Montpellier
Occitanie
Toulouse
Monaco
Genova
Marseille
Bilbao
Vitoria-Gasteiz
Navarra
Nafarroa
Burgos
Andorra
Perpignan
Zaragoza
Lleida
Catalunya
Girona
Mataró
Castilla y León
Aragón
Ajaccio / Aiacciu


cour d'appel de Paris: viol
Nederland
Den Haag
Arnhem
Münster
Osnabrück
Hannover
Bielefeld
Eindhoven
Middelburg
Brugge
Dunkerque
Vlaanderen
Nordrhein-Westfalen
Kassel
Düsseldorf
Aachen
Bonn
Siegen
Koblenz
Hessen
België / Belgique / Belgien
Mons
Frankfurt am Main
Mannheim
Letzebuerg
Saarbrücken
Karlsruhe
Baden-Württemberg
Grand Est
Troyes
Freiburg im Breisgau
Ulm
Basel
Zürich
Schweiz/ Suisse/Svizzera/ Svizra
Lausanne
Genève
Besançon
Bourgogne-Franche-Comté
Centre-Val de Loire
Pays de la Loire
Angers
Le Mans
Rennes
Bretagne
Guernsey
Jersey
Havre
Rouen
Normandie
Reims
France
Poitiers
La Rochelle
Limoges
Clermont-Ferrand
Villeurbanne
Auvergne-Rhône-Alpes
Grenoble
Varese
Lombardia
Novara
Piemonte
Piacenza
Alessandria
Nouvelle-Aquitaine
Bordeaux
Provence-Alpes-Côte d'Azur
Avignon
Montpellier
Occitanie
Toulouse
Monaco
Genova
Marseille
Bilbao
Vitoria-Gasteiz
Navarra
Nafarroa
Burgos
Andorra
Perpignan
Zaragoza
Girona

Les Incodables