```
 1 import pandas as pd
 2 import seaborn as sns
 3 import numpy as np
 4 import requests
 5 %matplotlib inline
 6 import plotly.express as px
 7 import folium
 8 from folium import plugins
 9 from folium.plugins import HeatMap
10 from folium.plugins import MarkerCluster
11 from folium.plugins import MeasureControl
```

```
 1 from google.colab import drive
 2 drive.mount('/content/drive')
 3
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```
 1 BASE_PATH = '/content/drive/MyDrive/Cleveland crime A26/data'
 2
```

```
 1 import os
 2 os.listdir(BASE_PATH)
 3
```

```
['2023', '2024', '2025']
```

```
 1 os.listdir(f'{BASE_PATH}/2023')
 2
```

```
['2023-04-cleveland-street.csv',
 '2023-02-cleveland-street.csv',
 '2023-04-cleveland-outcomes.csv',
 '2023-01-cleveland-street.csv',
 '2023-02-cleveland-outcomes.csv',
 '2023-04-cleveland-stop-and-search.csv',
 '2023-03-cleveland-street.csv',
 '2023-09-cleveland-stop-and-search.csv',
 '2023-03-cleveland-stop-and-search.csv',
 '2023-03-cleveland-outcomes.csv',
 '2023-01-cleveland-outcomes.csv',
 '2023-11-cleveland-stop-and-search.csv',
 '2023-10-cleveland-stop-and-search.csv',
 '2023-08-cleveland-stop-and-search.csv',
 '2023-05-cleveland-stop-and-search.csv',
 '2023-07-cleveland-stop-and-search.csv',
 '2023-12-cleveland-stop-and-search.csv',
 '2023-01-cleveland-stop-and-search.csv',
 '2023-02-cleveland-stop-and-search.csv',
 '2023-06-cleveland-stop-and-search.csv',
 '2023-05-cleveland-outcomes.csv',
 '2023-05-cleveland-street.csv',
 '2023-06-cleveland-outcomes.csv',
 '2023-08-cleveland-outcomes.csv',
 '2023-07-cleveland-street.csv',
 '2023-07-cleveland-outcomes.csv',
 '2023-06-cleveland-street.csv',
 '2023-12-cleveland-street.csv',
 '2023-11-cleveland-outcomes.csv',
 '2023-12-cleveland-outcomes.csv',
 '2023-11-cleveland-street.csv',
 '2023-08-cleveland-street.csv',
 '2023-09-cleveland-outcomes.csv',
 '2023-10-cleveland-street.csv',
 '2023-10-cleveland-outcomes.csv',
 '2023-09-cleveland-street.csv']
```

```
 1 os.listdir(f'{BASE_PATH}/2024')
 2
```

```
['2024-11-cleveland-stop-and-search.csv',
 '2024-10-cleveland-stop-and-search.csv',
 '2024-12-cleveland-stop-and-search.csv',
 '2024-03-cleveland-stop-and-search.csv',
 '2024-05-cleveland-stop-and-search.csv',
 '2024-07-cleveland-stop-and-search.csv',
 '2024-01-cleveland-stop-and-search.csv',
 '2024-08-cleveland-stop-and-search.csv',
 '2024-06-cleveland-stop-and-search.csv',
 '2024-02-cleveland-stop-and-search.csv',
 '2024-04-cleveland-stop-and-search.csv',
```

```
 '2024-09-cleveland-stop-and-search.csv',
 '2024-01-cleveland-outcomes.csv',
 '2024-06-cleveland-outcomes.csv',
 '2024-05-cleveland-street.csv',
 '2024-02-cleveland-street.csv',
 '2024-02-cleveland-outcomes.csv',
 '2024-05-cleveland-outcomes.csv',
 '2024-03-cleveland-outcomes.csv',
 '2024-04-cleveland-street.csv',
 '2024-01-cleveland-street.csv',
 '2024-04-cleveland-outcomes.csv',
 '2024-03-cleveland-street.csv',
 '2024-08-cleveland-outcomes.csv',
 '2024-11-cleveland-outcomes.csv',
 '2024-12-cleveland-street.csv',
 '2024-09-cleveland-street.csv',
 '2024-10-cleveland-street.csv',
 '2024-10-cleveland-outcomes.csv',
 '2024-12-cleveland-outcomes.csv',
 '2024-07-cleveland-outcomes.csv',
 '2024-09-cleveland-outcomes.csv',
 '2024-08-cleveland-street.csv',
 '2024-11-cleveland-street.csv',
 '2024-06-cleveland-street.csv',
 '2024-07-cleveland-street.csv']
```

```
os.listdir(f'{BASE_PATH}/2025')

```

```
['2025-05-cleveland-stop-and-search.csv',
 '2025-09-cleveland-stop-and-search.csv',
 '2025-04-cleveland-stop-and-search.csv',
 '2025-03-cleveland-stop-and-search.csv',
 '2025-10-cleveland-stop-and-search.csv',
 '2025-11-cleveland-stop-and-search.csv',
 '2025-01-cleveland-stop-and-search.csv',
 '2025-07-cleveland-stop-and-search.csv',
 '2025-06-cleveland-stop-and-search.csv',
 '2025-02-cleveland-stop-and-search.csv',
 '2025-08-cleveland-stop-and-search.csv',
 '2025-02-cleveland-street.csv',
 '2025-01-cleveland-outcomes.csv',
 '2025-04-cleveland-street.csv',
 '2025-01-cleveland-street.csv',
 '2025-04-cleveland-outcomes.csv',
 '2025-03-cleveland-street.csv',
 '2025-03-cleveland-outcomes.csv',
 '2025-02-cleveland-outcomes.csv',
 '2025-09-cleveland-outcomes.csv',
 '2025-08-cleveland-outcomes.csv',
 '2025-08-cleveland-street.csv',
 '2025-05-cleveland-outcomes.csv',
 '2025-06-cleveland-outcomes.csv',
 '2025-07-cleveland-outcomes.csv',
 '2025-07-cleveland-street.csv',
 '2025-06-cleveland-street.csv',
 '2025-05-cleveland-street.csv',
 '2025-09-cleveland-street.csv',
 '2025-11-cleveland-street.csv',
 '2025-11-cleveland-outcomes.csv',
 '2025-10-cleveland-street.csv',
 '2025-10-cleveland-outcomes.csv']
```

```
Start coding or generate with AI.
```

```
import pandas as pd
import os

def load_year_data(year):
    folder = f'{BASE_PATH}/{year}'
    files = [f for f in os.listdir(folder) if f.endswith('.csv')]

    df = pd.concat(
        [pd.read_csv(f'{folder}/{file}') for file in files],
        ignore_index=True
    )

    df['year'] = year
    return df

df_2023 = load_year_data(2023)
df_2024 = load_year_data(2024)
df_2025 = load_year_data(2025)

```

```
df_2023.head()
```

	Crime ID	Month	Reported by	Falls within	Longitude	Latitude	Location	LSOA code
0	NaN	2023-04	Cleveland Police	Cleveland Police	-1.235660	54.710526	On or near Dobson Place	E01011954
1	d6de0a0661244386645bc3435c8319194fd27234272d29...	2023-04	Cleveland Police	Cleveland Police	-1.233941	54.711980	On or near Wolsingham Road	E01011954
2	7ea9cd133d761ac9c85a4367bab92d7e390b475949936e...	2023-04	Cleveland Police	Cleveland Police	-1.239173	54.711061	On or near Butterwick Road	E01011954
3	0913b9503bc8873b9134c84b61410c5d1a4e72069a1e35...	2023-04	Cleveland Police	Cleveland Police	-1.239498	54.711980	On or near King Oswy Drive	E01011954
4	ecb46016e7cf86b64caba450a6b913ab1e4778cc3dee5f...	2023-04	Cleveland Police	Cleveland Police	-1.239865	54.710589	On or near Marshall Close	E01011954

5 rows × 27 columns

```
df_2023.shape
df_2024.shape
df_2025.shape

```

```
(155505, 27)
```

```
df_all = pd.concat(
    [df_2023, df_2024, df_2025],
    ignore_index=True
)

df_all.shape

```

```
(513391, 27)
```

```
Start coding or generate with AI.
```

```
df_count_gp = df_all.groupby(['Crime type', 'LSOA code']).size().to_frame(name='count_each_LSOA').reset_index()
df_count_gp
```

	Crime type	LSOA code	count_each_LSOA
0	Anti-social behaviour	E01011949	103
1	Anti-social behaviour	E01011950	200
2	Anti-social behaviour	E01011951	181
3	Anti-social behaviour	E01011952	182
4	Anti-social behaviour	E01011953	167
...	...	...	...
4746	Violence and sexual offences	E01035200	336
4747	Violence and sexual offences	E01035201	160
4748	Violence and sexual offences	E01035202	207
4749	Violence and sexual offences	E01035203	71
4750	Violence and sexual offences	E01035204	58

4751 rows × 3 columns

Next steps: Generate code with df_count_gp | New interactive sheet

```
df_all.columns
```

```
Index(['Crime ID', 'Month', 'Reported by', 'Falls within', 'Longitude',
       'Latitude', 'Location', 'LSOA code', 'LSOA name', 'Crime type',
       'Last outcome category', 'Context', 'Outcome type', 'Type', 'Date',
       'Part of a policing operation', 'Policing operation', 'Gender',
       'Age range', 'Self-defined ethnicity', 'Officer-defined ethnicity',
       'Legislation', 'Object of search', 'Outcome',
       'Outcome linked to object of search',
       'Removal of more than just outer clothing', 'year'],
      dtype='object')
```

```
df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 513391 entries, 0 to 513390
Data columns (total 27 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   Crime ID                                440867 non-null  object 
 1   Month                                   491269 non-null  object 
 2   Reported by                             491269 non-null  object 
 3   Falls within                            491269 non-null  object 
 4   Longitude                               512221 non-null  float64
 5   Latitude                                512221 non-null  float64
 6   Location                                491269 non-null  object 
 7   LSOA code                               491269 non-null  object 
 8   LSOA name                               491269 non-null  object 
 9   Crime type                              269438 non-null  object 
 10  Last outcome category                   219036 non-null  object 
 11  Context                                 0 non-null       float64
 12  Outcome type                            221831 non-null  object 
 13  Type                                    22122 non-null   object 
 14  Date                                    22122 non-null   object 
 15  Part of a policing operation            0 non-null       float64
 16  Policing operation                      0 non-null       float64
 17  Gender                                  19215 non-null   object 
 18  Age range                               18878 non-null   object 
 19  Self-defined ethnicity                  17635 non-null   object 
 20  Officer-defined ethnicity               17400 non-null   object 
 21  Legislation                             21865 non-null   object 
 22  Object of search                        21617 non-null   object 
 23  Outcome                                 19969 non-null   object 
 24  Outcome linked to object of search      5715 non-null    object 
 25  Removal of more than just outer clothing  0 non-null     float64
 26  year                                    513391 non-null  int64  
dtypes: float64(6), int64(1), object(20)
memory usage: 105.8+ MB
```

```
df_all.isna().sum().sort_values(ascending=False).head(15)
```

	0
Policing operation	513391
Part of a policing operation	513391
Context	513391
Removal of more than just outer clothing	513391
Outcome linked to object of search	507676
Officer-defined ethnicity	495991
Self-defined ethnicity	495756
Age range	494513
Gender	494176
Outcome	493422
Object of search	491774
Legislation	491526
Type	491269
Date	491269
Last outcome category	294355

dtype: int64

```
vso_all = vso_all.dropna(subset=["Latitude", "Longitude"])

```

```
key_cols = ["date", "month", "year", "offence", "crime_type", "latitude", "longitude", "lsoa", "postcode"]
[col for col in key_cols if col in [c.lower() for c in df_all.columns]]

```

```
['date', 'month', 'year', 'latitude', 'longitude']
```

```
[c for c in df_all.columns if "date" in c.lower() or "time" in c.lower()]

```

```
['Date']
```

```
df_all["Date"] = pd.to_datetime(df_all["Date"], errors="coerce")
df_all["year"] = df_all["Date"].dt.year
df_all["month"] = df_all["Date"].dt.month

```

```
df_all["Crime type"].value_counts()
```

Crime type	count
Violence and sexual offences	87956
Anti-social behaviour	50402
Criminal damage and arson	27153
Shoplifting	24049
Public order	19424
Other theft	14003
Burglary	12280
Vehicle crime	11526
Drugs	8087
Other crime	7002
Robbery	2818
Possession of weapons	2137
Bicycle theft	1626
Theft from the person	975

dtype: int64

```
gp_by_crime = (
    df_all
    .groupby(["Month", "Crime type"])
    .size()
    .rename("incidents")
    .reset_index()
)
gp_by_crime.head()

```

	Month	Crime type	incidents
0	2023-01	Anti-social behaviour	1161
1	2023-01	Bicycle theft	40
2	2023-01	Burglary	416
3	2023-01	Criminal damage and arson	770
4	2023-01	Drugs	217

Next steps: Generate code with gp_by_crime | New interactive sheet

```
gp_by_crime = df_all.groupby(["Month", "Crime type"])['Reported by'].count()
gp_by_crime
```

		Reported by
Month	Crime type	
2023-01	Anti-social behaviour	1161
	Bicycle theft	40
	Burglary	416
	Criminal damage and arson	770
	Drugs	217
...	...	...
2025-11	Robbery	120
	Shoplifting	900
	Theft from the person	30
	Vehicle crime	304
	Violence and sexual offences	2352

490 rows × 1 columns

dtype: int64

```
cross_tab_crime = pd.crosstab(
    index=df_all["Crime type"],
    columns=df_all["Month"]
)

```

```
cross_tab_crime = pd.crosstab(df_all['Crime type'], df_all['Month'])
cross_tab_crime
```

Month	2023-01	2023-02	2023-03	2023-04	2023-05	2023-06	2023-07	2023-08	2023-09	2023-10	...	2025-02	2025-03	2025-04	2025-05	2025-06	2025-07
Crime type																	
Anti-social behaviour	1161	1402	1559	1700	1639	1834	1696	1722	1403	1459	...	1181	1621	1703	1749	1672	166
Bicycle theft	40	69	72	42	63	56	83	56	56	63	...	27	36	34	50	42	6
Burglary	416	383	488	438	407	441	415	450	403	413	...	235	290	277	320	322	32
Criminal damage and arson	770	793	894	868	830	803	886	881	849	835	...	590	706	703	754	808	77
Drugs	217	189	175	175	239	202	204	215	200	206	...	261	290	281	224	211	23
Other crime	163	181	202	155	200	206	191	185	187	176	...	231	263	202	199	233	27
Other theft	383	414	446	487	435	491	494	461	496	429	...	300	323	344	360	372	38
Possession of weapons	64	54	58	58	72	69	65	55	53	61	...	51	64	71	68	46	6
Public order	635	726	733	727	743	732	692	584	561	579	...	374	464	471	505	514	56
Robbery	71	78	70	85	67	83	83	65	69	59	...	65	82	96	93	133	12
Shoplifting	592	709	774	722	596	507	613	645	744	719	...	654	740	658	674	650	71
Theft from the person	43	41	42	56	65	34	28	23	29	26	...	22	25	10	24	31	2
Vehicle crime	359	365	391	440	354	345	380	385	438	487	...	202	275	225	324	261	23
Violence and sexual offences	2749	2665	2822	2754	3054	2916	2726	2572	2603	2636	...	2044	2404	2363	2455	2512	273

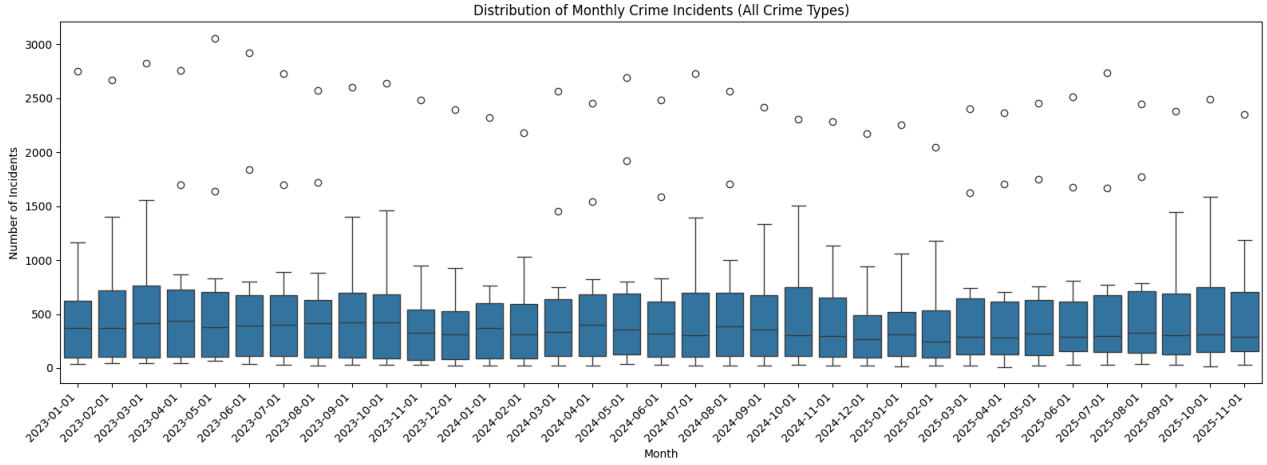
14 rows × 35 columns

```
box_df = (
    cross_tab_crime
    .reset_index()
    .melt(
```

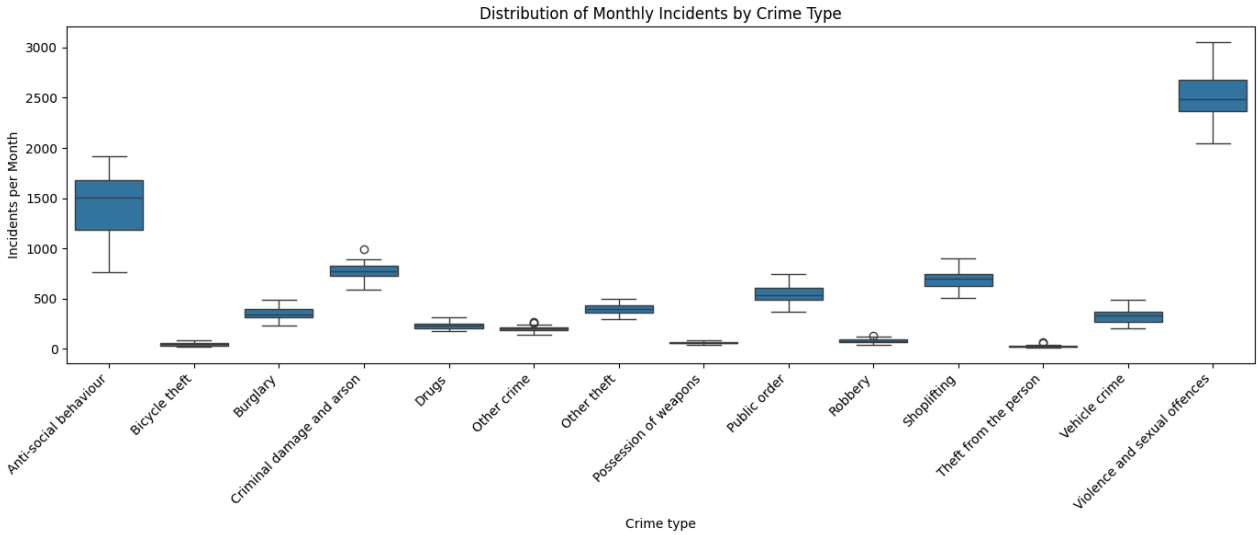
```
        id_vars="Crime type",
        var_name="Month",
        value_name="Incidents"
    )
)

```

```
box_df["Month"] = pd.to_datetime(box_df["Month"], format="%Y-%m")

```

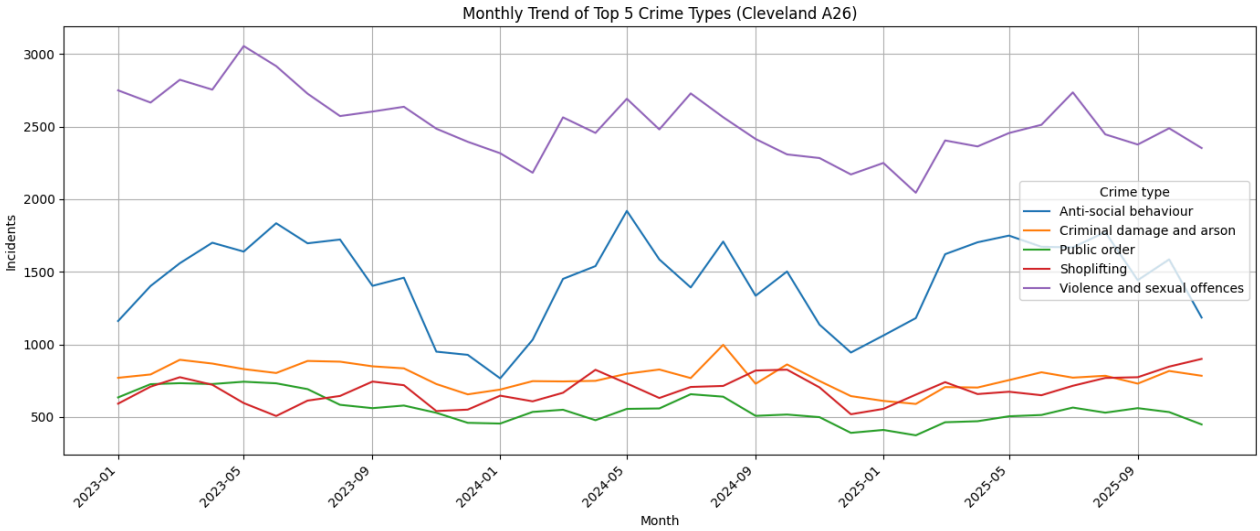
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(16,6))
sns.boxplot(
    data=box_df,
    x="Month",
    y="Incidents"
)

plt.xticks(rotation=45, ha="right")
plt.xlabel("Month")
plt.ylabel("Number of Incidents")
plt.title("Distribution of Monthly Crime Incidents (All Crime Types)")
plt.tight_layout()
plt.show()

```




```
plt.figure(figsize=(14,6))
sns.boxplot(
    data=box_df,
    x="Crime type",
    y="Incidents"
)

plt.xticks(rotation=45, ha="right")
plt.title("Distribution of Monthly Incidents by Crime Type")
plt.xlabel("Crime type")
plt.ylabel("Incidents per Month")
plt.tight_layout()
plt.show()

```



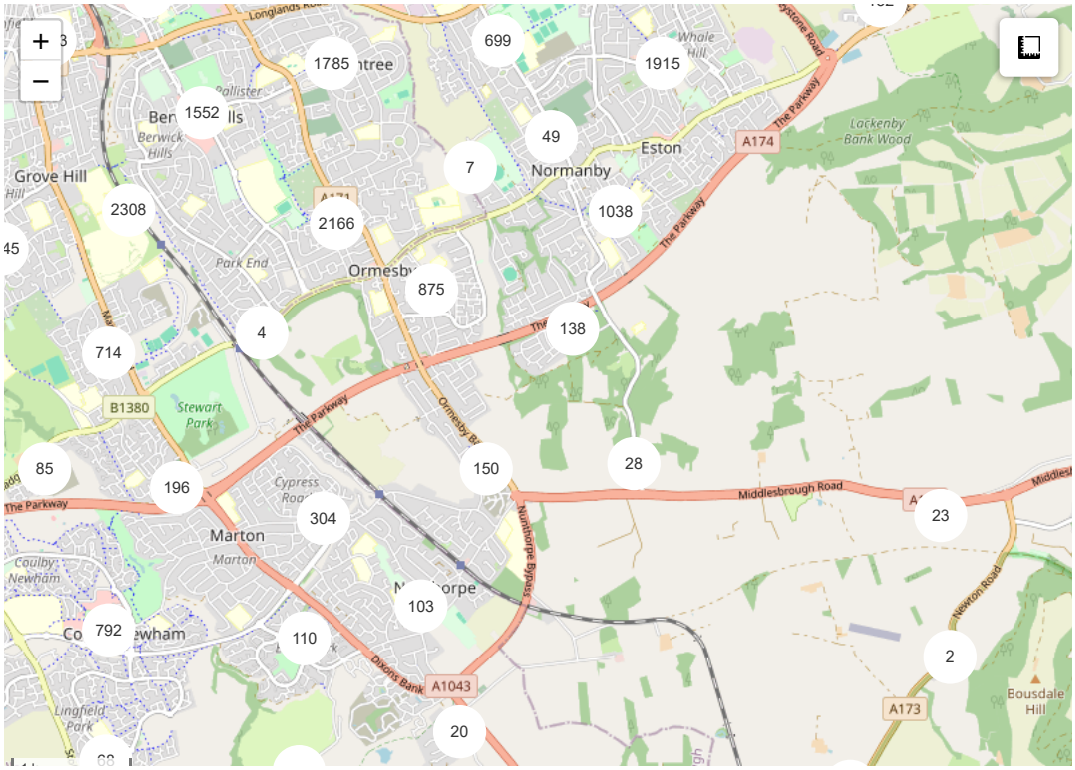

```
monthly_crime = (
    df_all
    .groupby(["Month", "Crime type"])
    .size()
    .reset_index(name="Incidents")
)

monthly_crime["Month"] = pd.to_datetime(monthly_crime["Month"])

top5 = (
    df_all["Crime type"]
    .value_counts()
    .head(5)
    .index
)

plt.figure(figsize=(14,6))
sns.lineplot(
    data=monthly_crime[monthly_crime["Crime type"].isin(top5)],
    x="Month",
    y="Incidents",
    hue="Crime type"
)

plt.xticks(rotation=45, ha="right")
plt.title("Monthly Trend of Top 5 Crime Types (Cleveland A26)")
plt.xlabel("Month")
plt.ylabel("Incidents")
plt.grid(True)
plt.tight_layout()
plt.show()

```



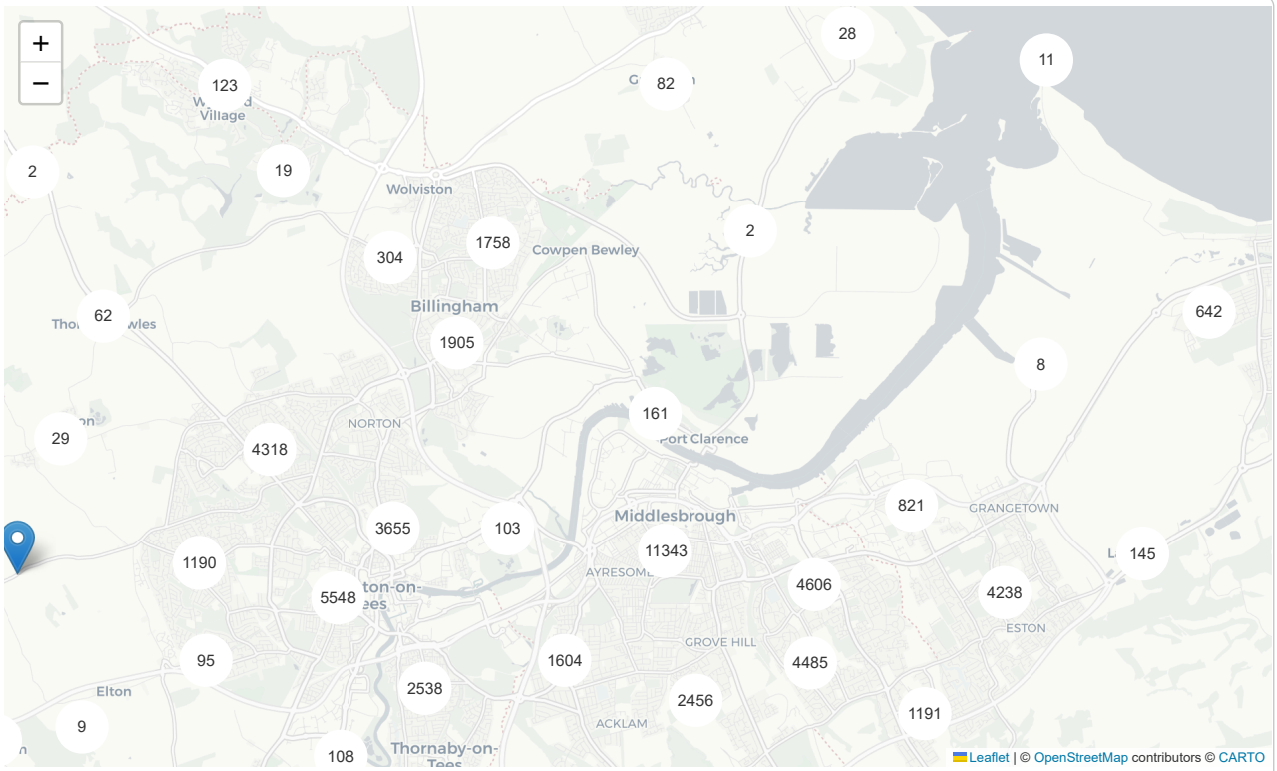

```
import folium
from folium import plugins
from folium.plugins import HeatMap
from folium.plugins import MarkerCluster
from folium.plugins import MeasureControl
```

```
crime_map = folium.Map(width=800, height=600, location=[54.54, -1.16], control_scale = True, zoom_start=13)
crime_map.add_child(MeasureControl())
crime_map
```

```
vso_map_cda = df_all.loc[df_all['Crime type'] == 'Criminal damage and arson']
vso_map_cda = vso_all[['Latitude', 'Longitude']].to_numpy()
crime_map.add_child(plugins.MarkerCluster(vso_map))
crime_map
```




```
import folium
from folium import plugins

# Filter offence type
vso_all = df_all[df_all["Crime type"] == "Violence and sexual offences"]
vso_all = vso_all.dropna(subset=["Latitude", "Longitude"])

# Create base map centred on data
crime_map = folium.Map(
    location=[vso_all["Latitude"].mean(), vso_all["Longitude"].mean()],
    zoom_start=12,
    tiles="CartoDB positron"
)

# Create marker cluster
marker_cluster = plugins.MarkerCluster(name="Violence & Sexual Offences")

# Add points with popups
for _, row in vso_all.iterrows():
    folium.Marker(
        location=[row["Latitude"], row["Longitude"]],
        popup=f"""
        <b>Crime type:</b> {row['Crime type']}<br>
        <b>Month:</b> {row['Month']}
        """,
    ).add_to(marker_cluster)

crime_map.add_child(marker_cluster)
folium.LayerControl().add_to(crime_map)
heat_data = vso_all[["Latitude", "Longitude"]].values.tolist()

plugins.HeatMap(
    heat_data,
    radius=12,
    blur=15,
    min_opacity=0.4,
    name="VSO Heatmap"
).add_to(crime_map)

crime_map


```



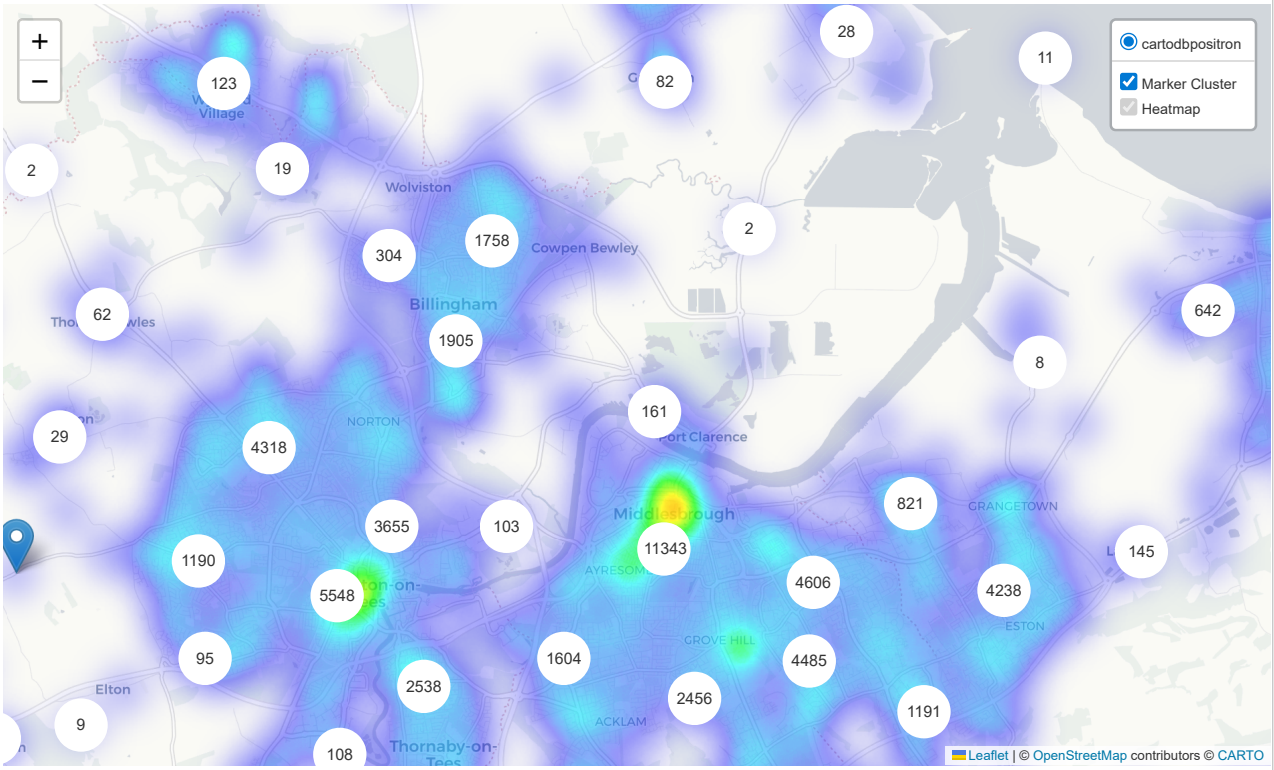

```
vso_all = df_all[df_all["Crime type"] == "Violence and sexual offences"].copy()

vso_all["Latitude"] = pd.to_numeric(vso_all["Latitude"], errors="coerce")
vso_all["Longitude"] = pd.to_numeric(vso_all["Longitude"], errors="coerce")

vso_all = vso_all.dropna(subset=["Latitude", "Longitude"])

```

Spatial Analysis: Violence and Sexual Offences (2023–2025)

This section visualises the geographic distribution of recorded violence and sexual offences across the Cleveland A26 policing area. Marker clustering highlights spatial concentration points, while heatmap intensity reveals persistent hotspots rather than isolated incidents. The analysis supports targeted patrol planning and preventative intervention strategies in high-risk locations.

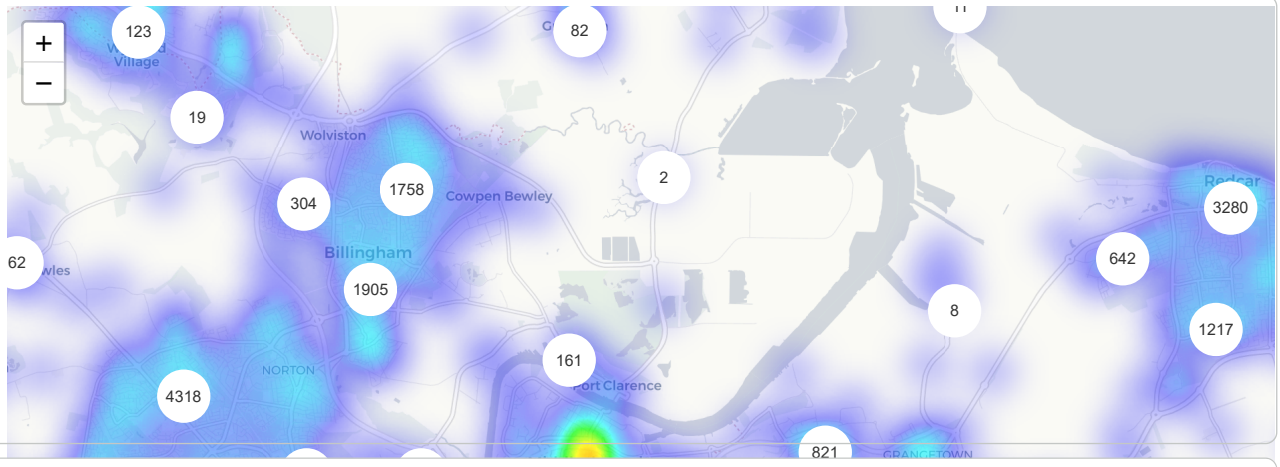
```
import folium
from folium import plugins
from folium.plugins import HeatMap

# centre map on your filtered data
crime_map = folium.Map(
    location=[vso_all["Latitude"].mean(), vso_all["Longitude"].mean()],
    zoom_start=12,
    tiles="CartoDB positron"
)

# -------- Marker Cluster layer --------
cluster = plugins.MarkerCluster(name="Marker Cluster")

for lat, lon in zip(vso_all["Latitude"], vso_all["Longitude"]):
    folium.Marker([lat, lon]).add_to(cluster)

cluster.add_to(crime_map)

# -------- HeatMap layer --------
heat_data = vso_all[["Latitude", "Longitude"]].values.tolist()

HeatMap(
    heat_data,
    name="Heatmap",
    radius=18,
    blur=25,
    min_opacity=0.3,
```

```
    max_zoom=1
).add_to(crime_map)

# layer control so you can toggle layers
folium.LayerControl(collapsed=False).add_to(crime_map)

crime_map

```

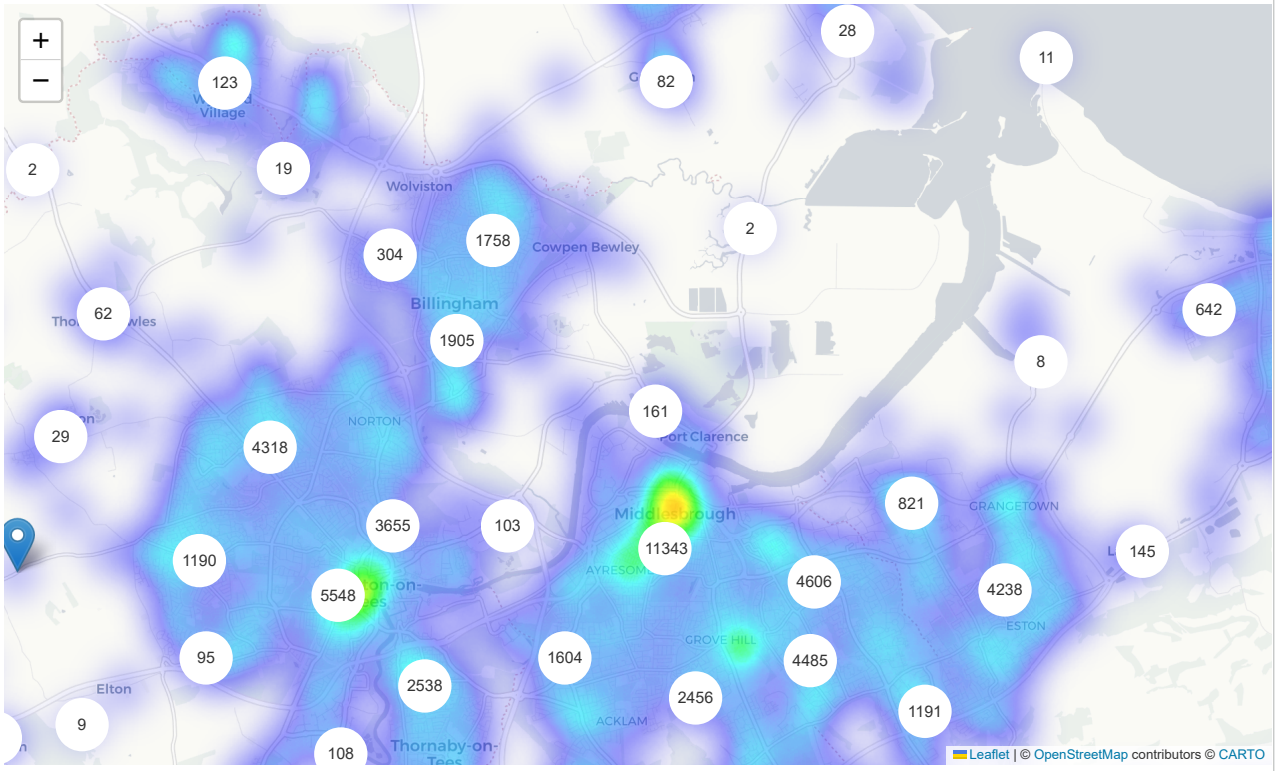



```
Start coding or generate with AI.
```

```
crime_types = ["Violence and sexual offences", "Burglary"]

for ct in crime_types:
    subset = df_all[df_all["Crime type"] == ct].dropna(subset=["Latitude","Longitude"])
    heat_data = subset[["Latitude","Longitude"]].values.tolist()

    HeatMap(
        heat_data,
        name=ct,
        radius=20,
        blur=25,
        min_opacity=0.4
    ).add_to(crime_map)

folium.LayerControl(collapsed=False).add_to(crime_map)
crime_map

```




```
for year in ["2023", "2024", "2025"]:
    yearly = vso_all[vso_all["Month"].str.startswith(year)]
    heat_data = yearly[["Latitude","Longitude"]].values.tolist()

    HeatMap(
        heat_data,
        name=f"VSO {year}",
        radius=20,
        blur=25,
        min_opacity=0.4
    ).add_to(crime_map)

folium.LayerControl(collapsed=False).add_to(crime_map)
crime_map

```




```
Start coding or generate with AI.
```

crime forecasting

```
Start coding or generate with AI.
```

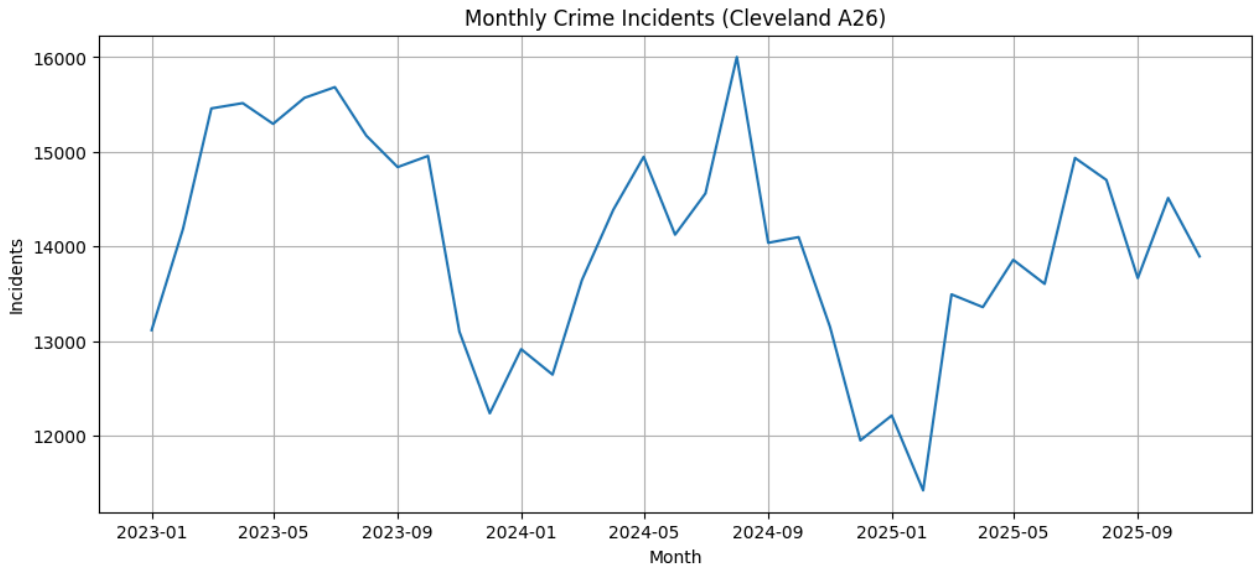
```
df_all["Month"] = pd.to_datetime(df_all["Month"], format="%Y-%m")

```

```
monthly_ts = (
    df_all
    .groupby("Month")
    .size()
    .asfreq("MS")   # Monthly Start frequency
)

monthly_ts.head()

```

Month	0
2023-01-01	13116
2023-02-01	14186
2023-03-01	15459
2023-04-01	15515
2023-05-01	15296

dtype: int64

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
plt.plot(monthly_ts)
plt.title("Monthly Crime Incidents (Cleveland A26)")
plt.xlabel("Month")
plt.ylabel("Incidents")
plt.grid(True)
plt.show()

```




```
train = monthly_ts[:-6]
test  = monthly_ts[-6:]

naive_forecast = train.iloc[-1]

print("Naive forecast for next months:", naive_forecast)

```

```
Naive forecast for next months: 13859
```

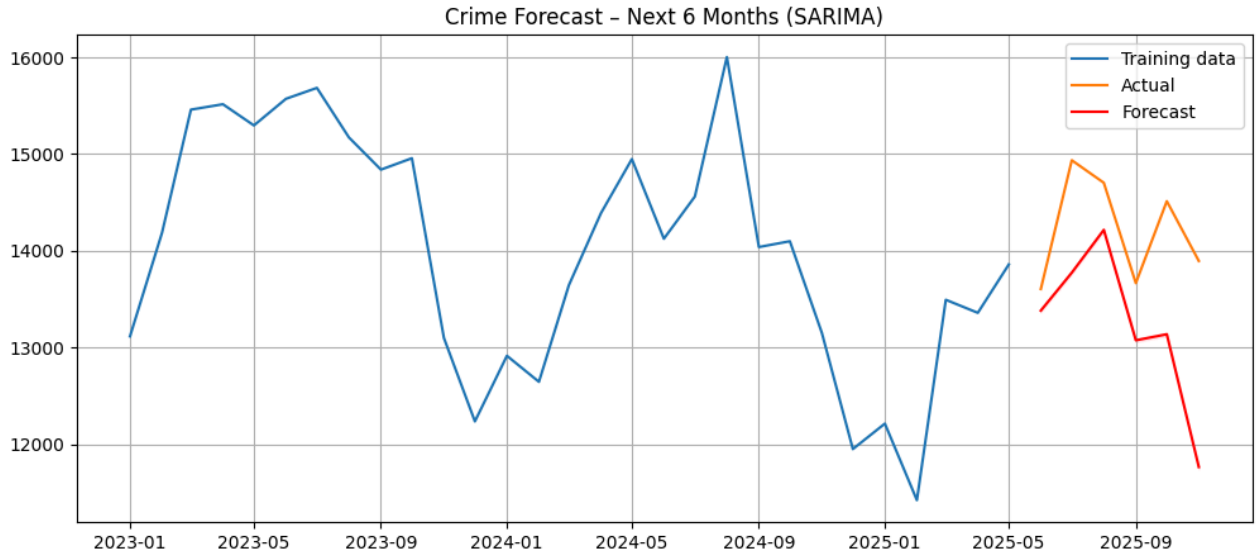
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

model = SARIMAX(
    train,
    order=(1,1,1),
    seasonal_order=(1,1,1,12),
    enforce_stationarity=False,
    enforce_invertibility=False
)
```

```
10
11 results = model.fit()
12 print(results.summary())
13
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:866: UserWarning:

Too few observations to estimate starting parameters for seasonal ARMA. All parameters except for variances will be set to z

                                SARIMAX Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                   29
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood                -6.129
Date:                            Wed, 04 Feb 2026   AIC                            22.258
Time:                                    05:50:14   BIC                            15.724
Sample:                                01-01-2023   HQIC                            8.593
                                     - 05-01-2025
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5783      0.015    -39.772      0.000      -0.607      -0.550
ma.L1         -0.0416      0.003    -15.794      0.000      -0.047      -0.036
ar.S.L12       0.0020      0.000      5.513      0.000       0.001       0.003
ma.S.L12      -0.9400   5.39e-05  -1.74e+04      0.000      -0.940      -0.940
sigma2        27.4606   3.29e-06   8.34e+06      0.000      27.461      27.461
===================================================================================
Ljung-Box (L1) (Q):                   2.00   Jarque-Bera (JB):                 0.33
Prob(Q):                              0.16   Prob(JB):                         0.85
Heteroskedasticity (H):                nan   Skew:                            -0.00
Prob(H) (two-sided):                   nan   Kurtosis:                         1.00
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.55e+23. Standard errors may be unstable.
/usr/local/lib/python3.12/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/mlemodel.py:3160: UserWarning:

Early subset of data for variable 0 has too few non-missing observations to calculate test statistic.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/mlemodel.py:3160: UserWarning:

Later subset of data for variable 0 has too few non-missing observations to calculate test statistic.
```

```
1 forecast = results.get_forecast(steps=6)
2 forecast_mean = forecast.predicted_mean
3 forecast_ci = forecast.conf_int()
4
```

```
 1 plt.figure(figsize=(12,5))
 2
 3 plt.plot(train, label="Training data")
 4 plt.plot(test, label="Actual")
 5 plt.plot(forecast_mean, label="Forecast", color="red")
 6
 7 plt.fill_between(
 8     forecast_ci.index,
 9     forecast_ci.iloc[:,0],
10     forecast_ci.iloc[:,1],
11     color="pink",
12     alpha=0.3
13 )
14
15 plt.legend()
16 plt.title("Crime Forecast – Next 6 Months (SARIMA)")
17 plt.grid(True)
18 plt.show()
19
```




```
1 from sklearn.metrics import mean_absolute_error, mean_squared_error
2 import numpy as np
3
4 mae = mean_absolute_error(test, forecast_mean)
5 rmse = np.sqrt(mean_squared_error(test, forecast_mean))
6
7 mae, rmse
8
```

```
(994.5270142249128, np.float64(1184.032433672397))
```

```
1 vso_ts = (
2     df_all[df_all["Crime type"] == "Violence and sexual offences"]
3     .groupby("Month")
4     .size()
5     .asfreq("MS")
6 )
7
```

```
1 Start coding or generate with AI.
```

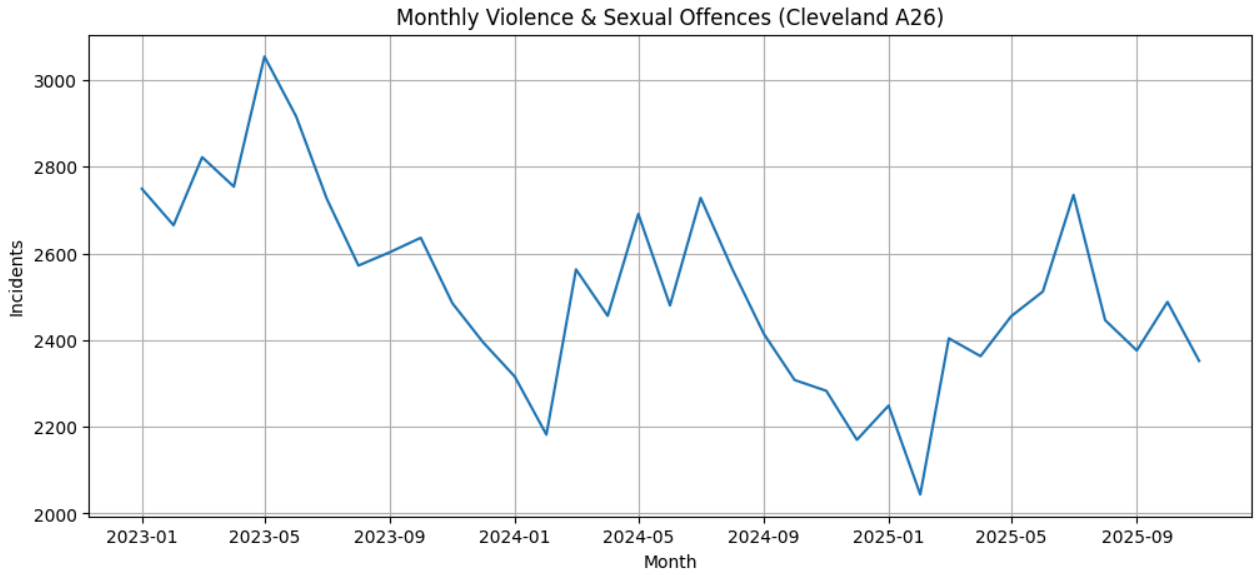
```
1 # ensure Month is datetime
2 df_all["Month"] = pd.to_datetime(df_all["Month"], format="%Y-%m")
3
4 # filter Violence & Sexual Offences
5 vso_df = df_all[df_all["Crime type"] == "Violence and sexual offences"]
6
7 # aggregate to monthly counts
8 vso_ts = (
9     vso_df
10     .groupby("Month")
11     .size()
12     .asfreq("MS")   # monthly frequency
13 )
14
15 vso_ts.head()
16
```

Month	0
2023-01-01	2749
2023-02-01	2665
2023-03-01	2822
2023-04-01	2754
2023-05-01	3054

dtype: int64

```
1 Start coding or generate with AI.
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
plt.plot(vso_ts)
plt.title("Monthly Violence & Sexual Offences (Cleveland A26)")
plt.xlabel("Month")
plt.ylabel("Incidents")
plt.grid(True)
plt.show()

```




```
Start coding or generate with AI.
```

```
train_vso = vso_ts[:-6]
test_vso  = vso_ts[-6:]

train_vso.tail(), test_vso

```

```
(Month
2025-01-01    2249
2025-02-01    2044
2025-03-01    2404
2025-04-01    2363
2025-05-01    2455
Freq: MS, dtype: int64,
Month
2025-06-01    2512
2025-07-01    2735
2025-08-01    2446
2025-09-01    2376
2025-10-01    2488
2025-11-01    2352
Freq: MS, dtype: int64)
```

```
naive_forecast = train_vso.iloc[-1]
naive_forecast

```

```
np.int64(2455)
```

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

sarima_vso = SARIMAX(
    train_vso,
    order=(1,1,1),
    seasonal_order=(1,1,1,12),
    enforce_stationarity=False,
    enforce_invertibility=False
)

sarima_results = sarima_vso.fit()
print(sarima_results.summary())

```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/sarimax.py:866: UserWarning:

Too few observations to estimate starting parameters for seasonal ARMA. All parameters except for variances will be set to z

                                SARIMAX Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                   29
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood                0.395
Date:                            Wed, 04 Feb 2026   AIC                           9.210
Time:                                    05:50:16   BIC                           2.675
Sample:                                01-01-2023   HQIC                         -4.455
                                     - 05-01-2025
Covariance Type:                             opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.4692      0.011   -133.093      0.000      -1.491      -1.448
ma.L1          0.5217      0.016     32.575      0.000       0.490       0.553
ar.S.L12       0.3787      0.003    132.207      0.000       0.373       0.384
ma.S.L12       1.0361      0.000   4846.177      0.000       1.036       1.037
sigma2         0.0222      0.010      2.297      0.022       0.003       0.041
===================================================================================
Ljung-Box (L1) (Q):                   2.00   Jarque-Bera (JB):                 0.33
Prob(Q):                              0.16   Prob(JB):                         0.85
Heteroskedasticity (H):                nan   Skew:                             0.00
Prob(H) (two-sided):                   nan   Kurtosis:                         1.00
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.07e+19. Standard errors may be unstable.
/usr/local/lib/python3.12/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/mlemodel.py:3160: UserWarning:

Early subset of data for variable 0 has too few non-missing observations to calculate test statistic.

/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/statespace/mlemodel.py:3160: UserWarning:

Later subset of data for variable 0 has too few non-missing observations to calculate test statistic.
```

```
vso_forecast = sarima_results.get_forecast(steps=6)

vso_forecast_mean = vso_forecast.predicted_mean
vso_forecast_ci = vso_forecast.conf_int()

vso_forecast_mean

```

	predicted_mean
2025-06-01	2390.423410
2025-07-01	2548.559533
2025-08-01	2756.537878
2025-09-01	1985.892754
2025-10-01	2638.406392
2025-11-01	1469.336051

dtype: float64

```
plt.figure(figsize=(12,5))

plt.plot(train_vso, label="Training data")
plt.plot(test_vso, label="Actual (last 6 months)")
plt.plot(vso_forecast_mean, label="Forecast", color="red")

plt.fill_between(
    vso_forecast_ci.index,
    vso_forecast_ci.iloc[:,0],
    vso_forecast_ci.iloc[:,1],
    color="pink",
    alpha=0.3
)

plt.title("Forecast of Violence & Sexual Offences (SARIMA)")
plt.xlabel("Month")
plt.ylabel("Incidents")
```

```
plt.legend()
plt.grid(True)
plt.show()

```