Perform Clustering (K Means & DBSCAN) for the crime data and identify the number of clusters formed and draw inferences.

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

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import silhouette_score, calinski_harabasz_score, silhouette_samples

import warnings
warnings.filterwarnings('ignore')
```

```
crime_data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/crime_data.csv')
crime_data
```

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0
3	Arkansas	8.8	190	50	19.5
4	California	9.0	276	91	40.6
5	Colorado	7.9	204	78	38.7
6	Connecticut	3.3	110	77	11.1
7	Delaware	5.9	238	72	15.8
8	Florida	15.4	335	80	31.9
9	Georgia	17.4	211	60	25.8
10	Hawaii	5.3	46	83	20.2
11	Idaho	2.6	120	54	14.2
12	Illinois	10.4	249	83	24.0
13	Indiana	7.2	113	65	21.0
14	Iowa	2.2	56	57	11.3
15	Kansas	6.0	115	66	18.0
16	Kentucky	9.7	109	52	16.3
17	Louisiana	15.4	249	66	22.2
18	Maine	2.1	83	51	7.8
19	Maryland	11.3	300	67	27.8
20	Massachusetts	4.4	149	85	16.3
21	Michigan	12.1	255	74	35.1

```
crime_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  50 non-null     object 
 1   Murder      50 non-null     float64
 2   Assault     50 non-null     int64  
 3   UrbanPop    50 non-null     int64  
 4   Rape        50 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 2.1+ KB
```

```
crime_data.describe()
```

	Murder	Assault	UrbanPop	Rape
count	50.00000	50.000000	50.000000	50.000000
mean	7.78800	170.760000	65.540000	21.232000
std	4.35551	83.337661	14.474763	9.366385
min	0.80000	45.000000	32.000000	7.300000
25%	4.07500	109.000000	54.500000	15.075000
50%	7.25000	159.000000	66.000000	20.100000
75%	11.25000	249.000000	77.750000	26.175000
max	17.40000	337.000000	91.000000	46.000000
41	Tennessee	13.2	188	59	26.9

```
crime_data.head()
```

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0

15.6

```
crime_data.isnull().sum()
```

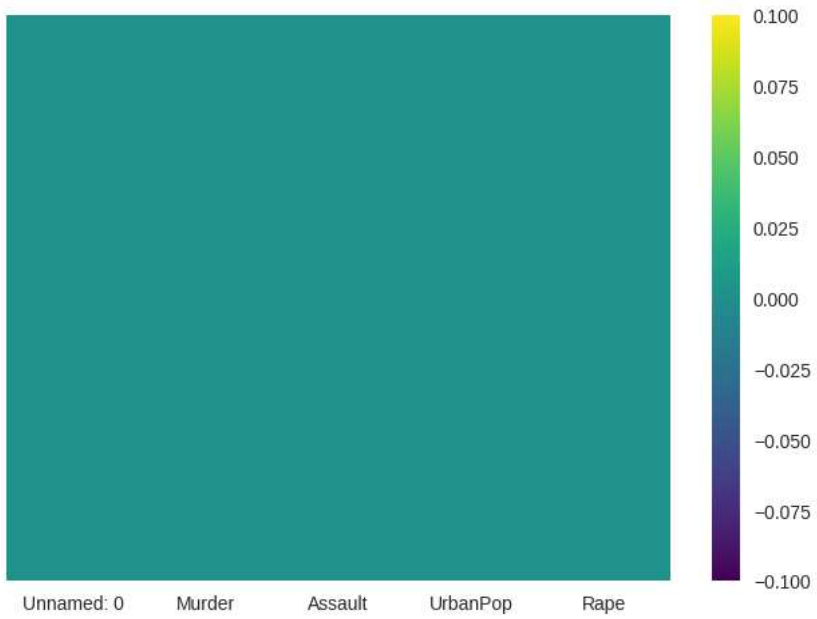
```
Unnamed: 0    0
Murder        0
Assault       0
UrbanPop      0
Rape          0
dtype: int64
```

```
crime_data.dtypes
```

```
Unnamed: 0     object
Murder        float64
Assault         int64
UrbanPop        int64
Rape          float64
dtype: object
```

```
sns.heatmap(crime_data.isnull(),yticklabels=False,cmap='viridis')
```

```
<Axes: >
```




```
crime_data.columns
```

```
Index(['Unnamed: 0', 'Murder', 'Assault', 'UrbanPop', 'Rape'], dtype='object')
```

```
### correlation
plt.figure(figsize=(12, 10))
correlation = crime_data.corr()
sns.heatmap(correlation, annot=True)
plt.show()
```


```
plt.figure(figsize=(20, 20))

columns = ['Murder', 'Assault', 'UrbanPop', 'Rape']
crime_data.boxplot()
plt.show()
```

```
df = crime_data.drop('Unnamed: 0', axis=1)
df.head()
```

	Murder	Assault	UrbanPop	Rape
0	13.2	236	58	21.2
1	10.0	263	48	44.5
2	8.1	294	80	31.0
3	8.8	190	50	19.5
4	9.0	276	91	40.6

```
X_numerics = df[['Murder', 'Assault', 'UrbanPop', 'Rape']]
```

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
sc.fit(X_numerics)

X_scaled = sc.transform(X_numerics)
```

```
X_scaled
```

```
array([[ 1.25517927,  0.79078716, -0.52619514, -0.00345116],
       [ 0.51301858,  1.11805959, -1.22406668,  2.50942392],
       [ 0.07236067,  1.49381682,  1.00912225,  1.05346626],
       [ 0.23470832,  0.23321191, -1.08449238, -0.18679398],
       [ 0.28109336,  1.2756352 ,  1.77678094,  2.08881393],
       [ 0.02597562,  0.40290872,  0.86954794,  1.88390137],
       [-1.04088037, -0.73648418,  0.79976079, -1.09272319],
       [-0.43787481,  0.81502956,  0.45082502, -0.58583422],
       [ 1.76541475,  1.99078607,  1.00912225,  1.1505301 ],
       [ 2.22926518,  0.48775713, -0.38662083,  0.49265293],
       [-0.57702994, -1.51224105,  1.21848371, -0.11129987],
       [-1.20322802, -0.61527217, -0.80534376, -0.75839217],
       [ 0.60578867,  0.94836277,  1.21848371,  0.29852525],
       [-0.13637203, -0.70012057, -0.03768506, -0.0250209 ],
       [-1.29599811, -1.39102904, -0.5959823 , -1.07115345],
       [-0.41468229, -0.67587817,  0.03210209, -0.34856705],
       [ 0.44344101, -0.74860538, -0.94491807, -0.53190987],
       [ 1.76541475,  0.94836277,  0.03210209,  0.10439756],
       [-1.31919063, -1.06375661, -1.01470522, -1.44862395],
       [ 0.81452136,  1.56654403,  0.10188925,  0.70835037],
       [-0.78576263, -0.26375734,  1.35805802, -0.53190987],
       [ 1.00006153,  1.02108998,  0.59039932,  1.49564599],
       [-1.1800355 , -1.19708982,  0.03210209, -0.68289807],
       [ 1.9277624 ,  1.06957478, -1.5032153 , -0.44563089],
       [ 0.28109336,  0.0877575 ,  0.31125071,  0.75148985],
       [-0.41468229, -0.74860538, -0.87513091, -0.521125  ],
       [-0.80895515, -0.83345379, -0.24704653, -0.51034012],
       [ 1.02325405,  0.98472638,  1.0789094 ,  2.671197  ],
       [-1.31919063, -1.37890783, -0.66576945, -1.26528114],
       [-0.08998698, -0.14254532,  1.63720664, -0.26228808],
       [ 0.83771388,  1.38472601,  0.31125071,  1.17209984],
       [ 0.76813632,  1.00896878,  1.42784517,  0.52500755],
       [ 1.20879423,  2.01502847, -1.43342815, -0.55347961],
       [-1.62069341, -1.52436225, -1.5032153 , -1.50254831],
       [-0.11317951, -0.61527217,  0.66018648,  0.01811858],
       [-0.27552716, -0.23951493,  0.1716764 , -0.13286962],
       [-0.66980002, -0.14254532,  0.10188925,  0.87012344],
       [-0.34510472, -0.78496898,  0.45082502, -0.68289807],
       [-1.01768785,  0.03927269,  1.49763233, -1.39469959],
       [ 1.53348953,  1.3119988 , -1.22406668,  0.13675217],
       [-0.92491776, -1.027393  , -1.43342815, -0.90938037],
       [ 1.25517927,  0.20896951, -0.45640799,  0.61128652],
       [ 1.13921666,  0.36654512,  1.00912225,  0.46029832],
       [-1.06407289, -0.61527217,  1.00912225,  0.17989166],
       [-1.29599811, -1.48799864, -2.34066115, -1.08193832],
       [ 0.16513075, -0.17890893, -0.17725937, -0.05737552],
       [-0.87853272, -0.31224214,  0.52061217,  0.53579242],
       [-0.48425985, -1.08799901, -1.85215107, -1.28685088],
       [-1.20322802, -1.42739264,  0.03210209, -1.1250778 ],
       [-0.22914211, -0.11830292, -0.38662083, -0.60740397]])
```

```
std_df = pd.DataFrame(X_scaled)
std_df
```

	0	1	2	3
0	1.255179	0.790787	-0.526195	-0.003451
1	0.513019	1.118060	-1.224067	2.509424
2	0.072361	1.493817	1.009122	1.053466
3	0.234708	0.233212	-1.084492	-0.186794
4	0.281093	1.275635	1.776781	2.088814
5	0.025976	0.402909	0.869548	1.883901
6	-1.040880	-0.736484	0.799761	-1.092723
7	-0.437875	0.815030	0.450825	-0.585834
8	1.765415	1.990786	1.009122	1.150530
9	2.229265	0.487757	-0.386621	0.492653
10	-0.577030	-1.512241	1.218484	-0.111300
11	-1.203228	-0.615272	-0.805344	-0.758392
12	0.605789	0.948363	1.218484	0.298525
13	-0.136372	-0.700121	-0.037685	-0.025021
14	-1.295998	-1.391029	-0.595982	-1.071153
15	-0.414682	-0.675878	0.032102	-0.348567
16	0.443441	-0.748605	-0.944918	-0.531910
17	1.765415	0.948363	0.032102	0.104398
18	-1.319191	-1.063757	-1.014705	-1.448624
19	0.814521	1.566544	0.101889	0.708350
20	-0.785763	-0.263757	1.358058	-0.531910
21	1.000062	1.021090	0.590399	1.495646

```
X = crime_data.drop('Unnamed: 0', axis=1)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
```

```
KMeans(n_clusters=3, random_state=42)
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

```
from yellowbrick.cluster import KElbowVisualizer
model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10))
visualizer.fit(X_numerics)
visualizer.show()
plt.show()
```


```
model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10), metric='silhouette')
visualizer.fit(X_numerics)
visualizer.show()
plt.show()
```