Project 5: Cluster Analysis & Visualization on U.S. Cities Parks & Recreation Data

Author/s: Gauri Vaidya

Date of Submission: 21st March 2025

For CS 215, Prof Brocks

I would like to acknowledge the use of Stack Overflow and the numerous resources available via Google insights and code examples as well as the Google Colab Python Helpdesk, which helped me resolve technical challenges that came up during this project. Additionally, I did refer to some of the modules in datacamp (the ones on box plot) and combining columns.

Part 1: Cluster analysis of parks & facilities data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.cluster import KMeans

# Loading the dataset
from google.colab import files
uploaded = files.upload("/Users/gaurivaidya/Desktop/")
```

Choose Files parks_data_2024.csv

- **parks_data_2024.csv**(text/csv) - 35720 bytes, last modified: 3/21/2025 - 100% done

```
Saving parks_data_2024.csv to /Users/gaurivaidya/Desktop/parks_data_2024 (4).csv
```

Generate 10 random numbers using numpy Close

```
#Step 1

df = pd.read_csv("/Users/gaurivaidya/Desktop/parks_data_2024 (4).csv")

# Displaying the first few rows
print(df.head())

# Checking dataset dimensions, info, and summary statistics for the first step
print(df.shape)
df.info()
```

```
print(df.describe())
```

```
              City  Population  Acres per 1,000 people  \
0  Albuquerque, NM    553345.0               38.905204   
1      Anaheim, CA    345538.0               13.344408   
2    Anchorage, AK    288464.0             3022.196184   
3    Arlington, TX    397158.0               10.869729   
4    Arlington, VA    245695.0                7.263477   

   Parks per 10,000 residents  Parks as % City Area  Fields/ Diamonds  \
0                    5.692651              0.189322          3.560166   
1                    1.881125              0.143306          2.604634   
2                    7.765267              0.801559          3.154640   
3                    2.517890              0.070975          2.366816   
4                    6.023729              0.112112          4.436395   

   Tennis_dedicdated  Pickleball_dedicated  Pickleball_combined      Hoops  \
0           3.397519              1.012027             2.891505   5.060134   
1           1.504900              1.447019             1.447019   1.504900   
2           2.357313              0.693327             0.693327   1.975983   
3           1.208587              0.553936             1.410018   7.830637   
4           7.326156              0.000000             1.628035  17.094365   

   Community_garden_sites  Dog_parks  Playgrounds  Rec_senior_centers  \
0                0.000000   3.975820     3.307159            1.120458   
1                0.011576   1.157615     1.736423            0.289404   
2                0.017333   2.773310     3.119973            0.207998   
3                0.002518   0.755367     4.230055            0.654651   
4                0.040701   4.070087     5.453916            1.221026   

   Restrooms  Skateparks  Splashpads  Swimming_pools  Disc_golf_courses  \
0   1.301177    0.000000    0.903595        0.000000           0.000000   
1   1.736423    2.604634    0.578808        0.000000           0.000000   
2   1.559987    1.733319    0.000000        1.733319           0.693327   
3   1.938775    1.007156    2.014312        1.510734           0.503578   
4   2.401351    0.407009    2.849061        2.035043           0.407009   

   investment_dollars  
0          220.434307  
1           76.151009  
2           68.927449  
3          112.913565  
4          260.647703  
(100, 20)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   City                        100 non-null    object 
 1   Population                  100 non-null    float64
 2   Acres per 1,000 people      100 non-null    float64
 3   Parks per 10,000 residents  100 non-null    float64
 4   Parks as % City Area        100 non-null    float64
 5   Fields/ Diamonds            100 non-null    float64
```

```
 6   Tennis_dedicdated          100 non-null    float64
 7   Pickleball_dedicated       100 non-null    float64
 8   Pickleball_combined        100 non-null    float64
 9   Hoops                      100 non-null    float64
```

```
# Step 2

# Selecting numeric columns
numeric_cols = df.select_dtypes(include=[np.number]).columns
df_numeric = df[numeric_cols]

# Normalizing the numeric features (each value divided by the column standard deviat
df_normalized = df_numeric / df_numeric.std()

# Displaying the first few rows of the normalized data
print(df_normalized.head())
```

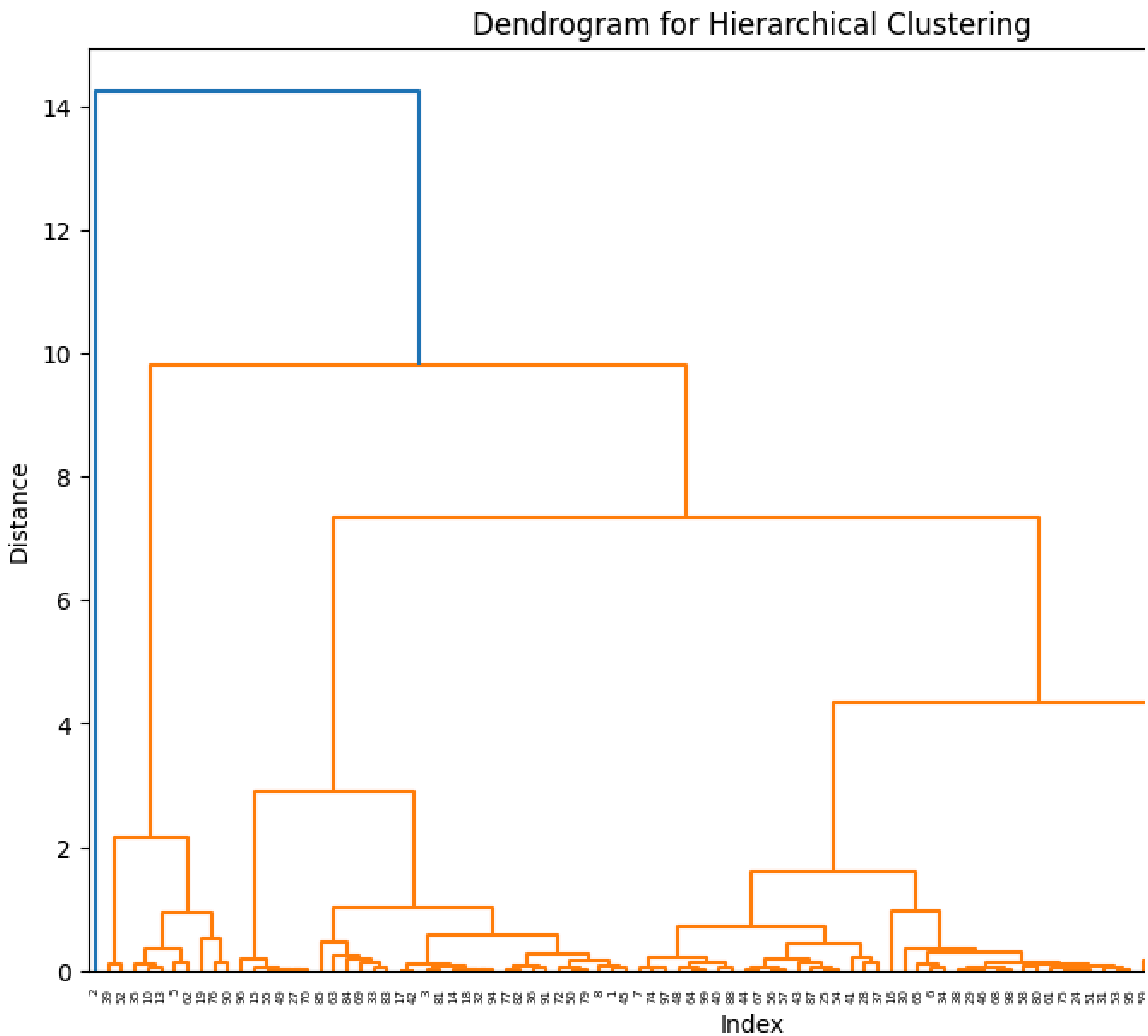
```
   Population  Acres per 1,000 people  Parks per 10,000 residents  \
0    0.567914                0.129078                    2.618802
1    0.354636                0.044274                    0.865378
2    0.296059               10.026918                    3.572272
3    0.407615                0.036063                    1.158310
4    0.252164                0.024098                    2.771109

   Parks as % City Area  Fields/ Diamonds  Tennis_dedicdated  \
0              1.932526          2.675051           1.693940
1              1.462808          1.957079           0.750315
2              8.182003          2.370345           1.175312
3              0.724488          1.778387           0.602579
4              1.144397          3.333435           3.652685

   Pickleball_dedicated  Pickleball_combined     Hoops  \
0              1.470240             2.638993  1.333337
1              2.102183             1.320652  0.396539
2              1.007244             0.632780  0.520668
3              0.804740             1.286883  2.063360
4              0.000000             1.485860  4.504337

   Community_garden_sites  Dog_parks  Playgrounds  Rec_senior_centers  \
0                0.000000   2.950869     1.787147            2.013198
1                0.357604   0.859187     0.938341            0.519990
2                0.535447   2.058361     1.685994            0.373724
3                0.077781   0.560636     2.285868            1.176253
4                1.257309   3.020835     2.947228            2.193895

   Restrooms  Skateparks  Splashpads  Swimming_pools  Disc_golf_courses  \
0   1.045278    0.000000    0.352550        0.000000           0.000000
1   1.394925    3.495339    0.225830        0.000000           0.000000
2   1.253188    2.326060    0.000000        1.007757           1.502728
3   1.557481    1.351572    0.785911        0.878346           1.091462
4   1.929084    0.546193    1.111600        1.183181           0.882157

   investment_dollars
0            1.742835
```

```
1          0.602078
2          0.544966
3          0.892736
4          2.060777
```

```
print("Available columns:", df.columns.tolist())
```

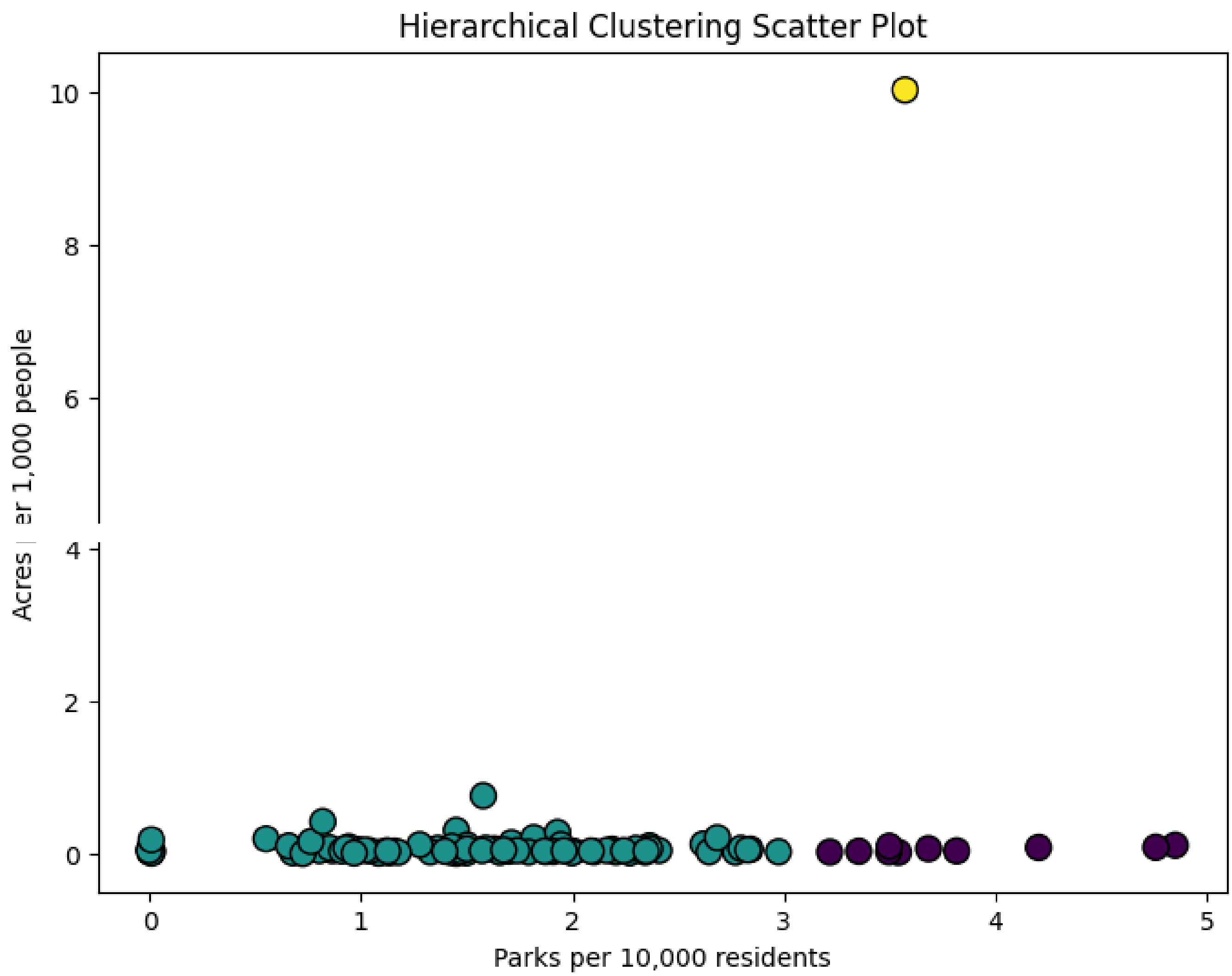
```
arks', 'Splashpads', 'Swimming_pools', 'Disc_golf_courses', 'investment_dollars']
```

```
# Step 3
# Part 1

#The dendogram
# Choosing two features for hierarchical clustering
features_hierarchy = ["Parks per 10,000 residents", "Acres per 1,000 people"]
data_hierarchy = df_normalized[features_hierarchy]

# Computing the linkage matrix using the 'ward' method
Z = linkage(data_hierarchy, method="ward")

# Plotting the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title("Dendrogram for Hierarchical Clustering")
plt.xlabel("Index")
plt.ylabel("Distance")
plt.show()
```



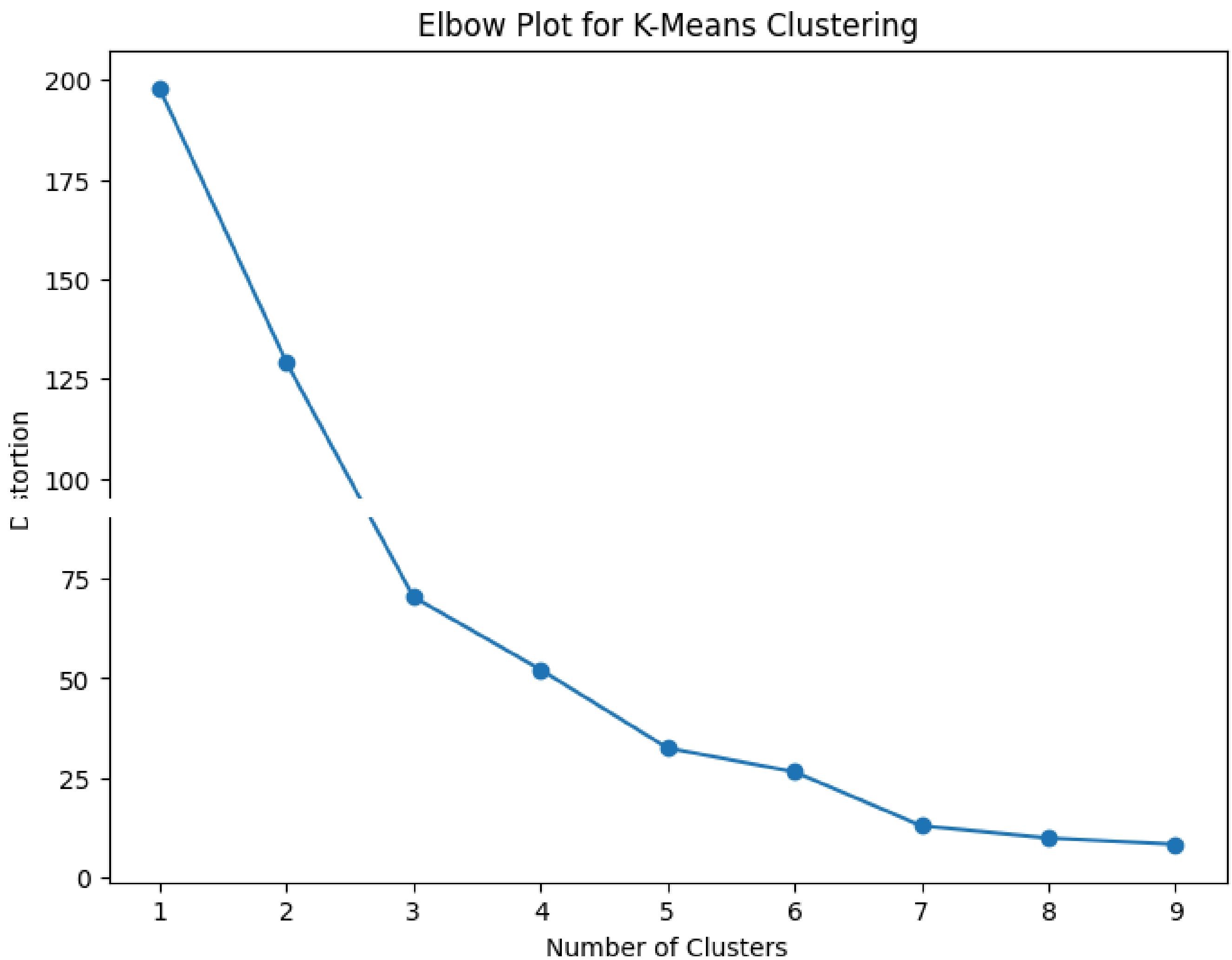

```
# Step 3
# Part 2

# Assigning clusters using the linkage matrix Z and a maxclust criterion (here, 3 cl
clusters_hierarchy = fcluster(Z, t=3, criterion="maxclust")

# Adding the cluster labels to the original dataframe
df["HierCluster"] = clusters_hierarchy

# Creating a scatterplot for the two features with points colored by their cluster l
plt.figure(figsize=(8,6))
plt.scatter(data_hierarchy["Parks per 10,000 residents"],
            data_hierarchy["Acres per 1,000 people"],
            c=clusters_hierarchy, cmap="viridis", edgecolor="k", s=100)
plt.xlabel("Parks per 10,000 residents")
plt.ylabel("Acres per 1,000 people")
```

```
plt.title("Hierarchical Clustering Scatter Plot")

plt.show()
```



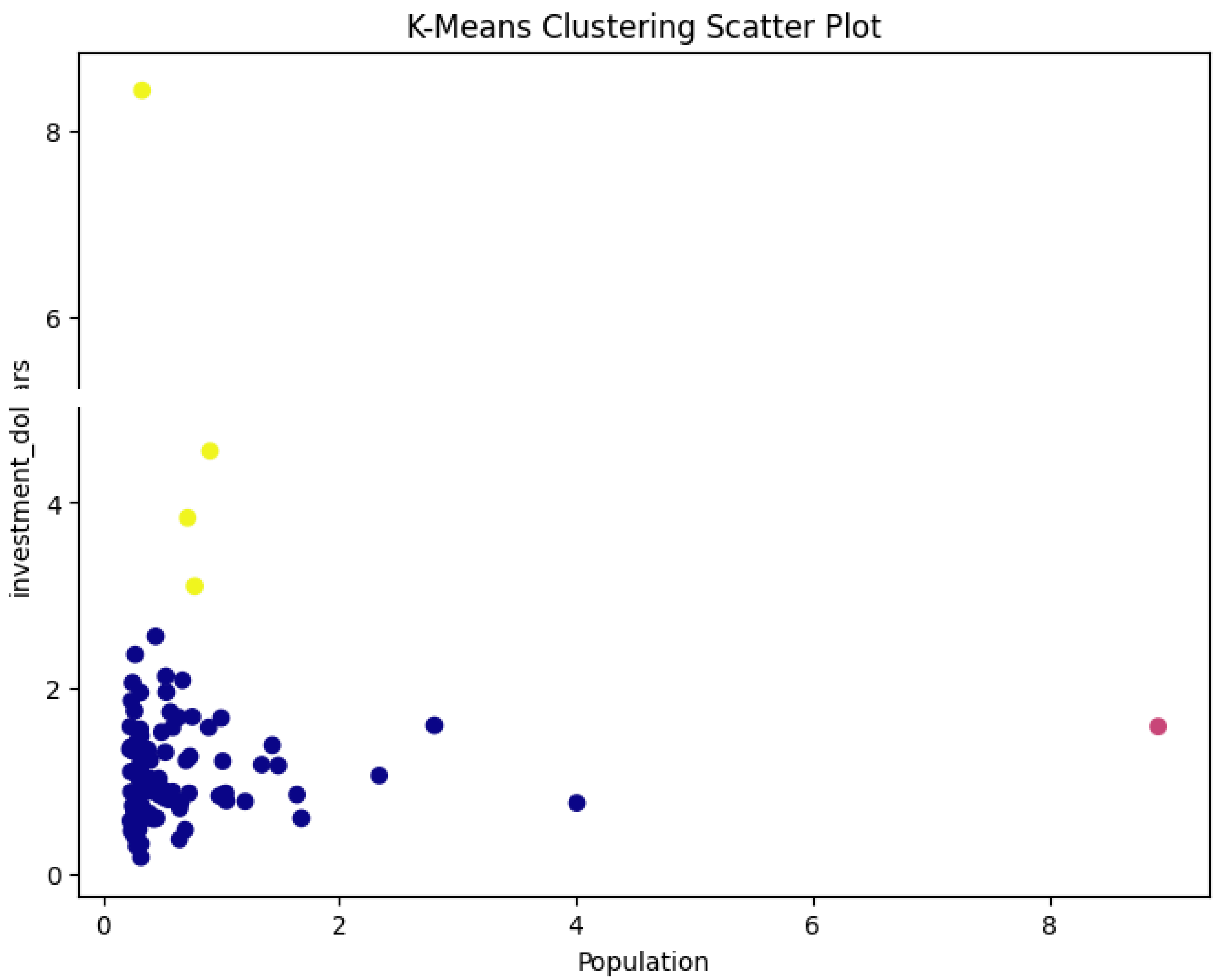

```
# Step 4
# Part 1

# Selecting features for k-means clustering
features_kmeans = ["Population", "investment_dollars"]
data_kmeans = df_normalized[features_kmeans]

# Running k-means with a range of cluster numbers and record the distortions (inerti
distortions = []
K = range(1, 10)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_kmeans)
    distortions.append(kmeans.inertia_)
```

```
# Plot the elbow curve
plt.figure(figsize=(8,6))
plt.plot(K, distortions, marker="o")
plt.xlabel("Number of Clusters")
plt.ylabel("Distortion")
plt.title("Elbow Plot for K-Means Clustering")
plt.show()
```




```
# Step 4
# Part 2

# Performing k-means clustering using the chosen optimal number (e.g., 3)
optimal_k = 3
kmeans_model = KMeans(n_clusters=optimal_k, random_state=42)
clusters_kmeans = kmeans_model.fit_predict(data_kmeans)
df["KMeansCluster"] = clusters_kmeans

# Scattering plot for the selected features with cluster colors
plt.figure(figsize=(8,6))
plt.scatter(data_kmeans[features_kmeans[0]], data_kmeans[features_kmeans[1]], c=clus
```

```
plt.xlabel(features_kmeans[0])
plt.ylabel(features_kmeans[1])
plt.title("K-Means Clustering Scatter Plot")
plt.show()
```




```
# Step 5

# Using all normalized numeric features for clustering
data_all = df_normalized.copy()

# Choosing the number of clusters (using 3 as an example)
optimal_k_all = 3
kmeans_all = KMeans(n_clusters=optimal_k_all, random_state=42)
clusters_all = kmeans_all.fit_predict(data_all)
df["AllNumericCluster"] = clusters_all

# Displaying cluster summary statistics (mean values for each cluster)
cluster_summary = df.groupby("AllNumericCluster")[numeric_cols].mean()
print("Cluster Summary (mean values):")
```

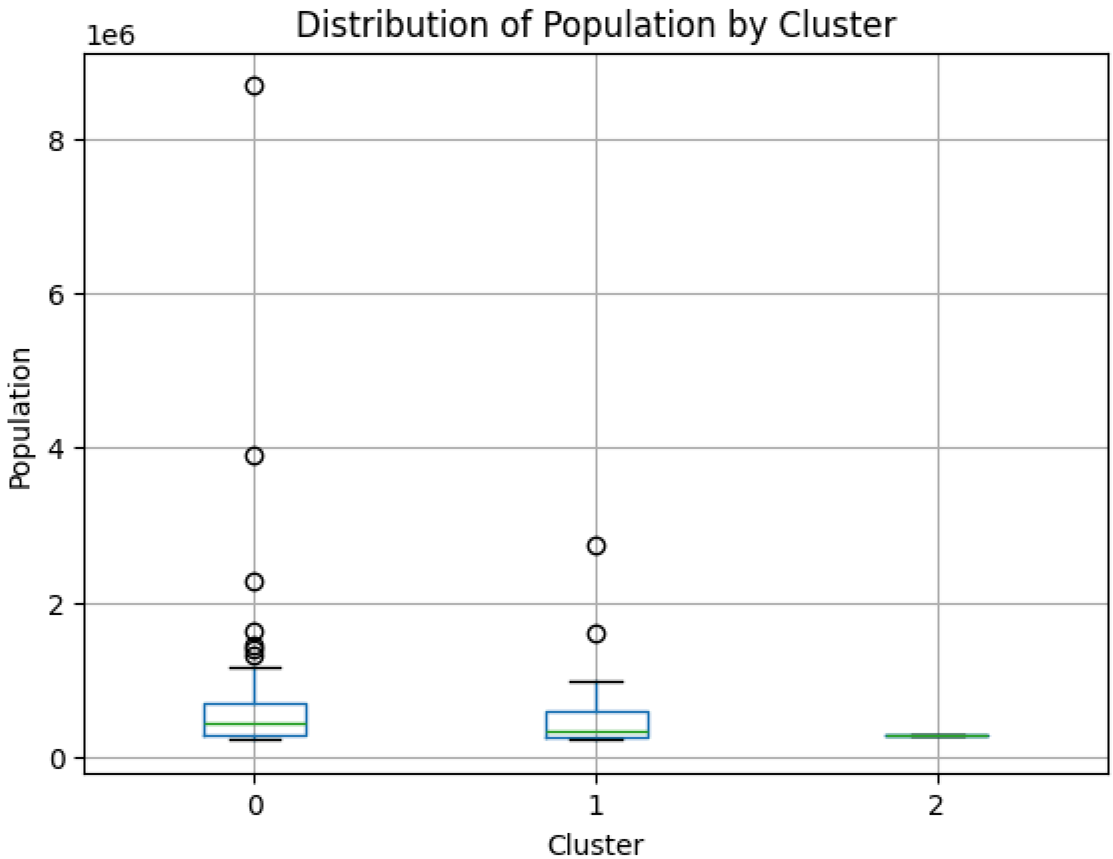
```
print(cluster_summary)
```

```
Cluster Summary (mean values):
                    Population  Acres per 1,000 people  \
AllNumericCluster
0                 741197.68254               22.000975
1                 514859.75000               21.934219
2                 288464.00000             3022.196184

                  Parks per 10,000 residents  Parks as % City Area  \
AllNumericCluster
0                                   3.080971              0.104068
1                                   5.124600              0.119555
2                                   7.765267              0.801559

                  Fields/ Diamonds  Tennis_dedicdated  Pickleball_dedicated  \
AllNumericCluster
0                         2.319738           2.451289              0.698688
1                         3.848218           4.607563              0.781413
2                         3.154640           2.357313              0.693327

                  Pickleball_combined     Hoops  Community_garden_sites  \
AllNumericCluster
0                            1.169792  3.366194                0.009595
1                            1.431529  8.204559                0.036324
2                            0.693327  1.975983                0.017333

                  Dog_parks  Playgrounds  Rec_senior_centers  Restrooms  \
AllNumericCluster
0                  1.336096     2.412073            0.746431   1.583417
1                  2.446469     4.263818            1.239007   2.695555
2                  2.773310     3.119973            0.207998   1.559987

                  Skateparks  Splashpads  Swimming_pools  Disc_golf_courses  \
AllNumericCluster
0                   0.800088    1.489849        1.693058           0.478711
1                   0.998201    4.088085        3.381543           0.619891
2                   1.733319    0.000000        1.733319           0.693327

                  investment_dollars
AllNumericCluster
0                         123.686214
1                         213.462223
2                          68.927449
```

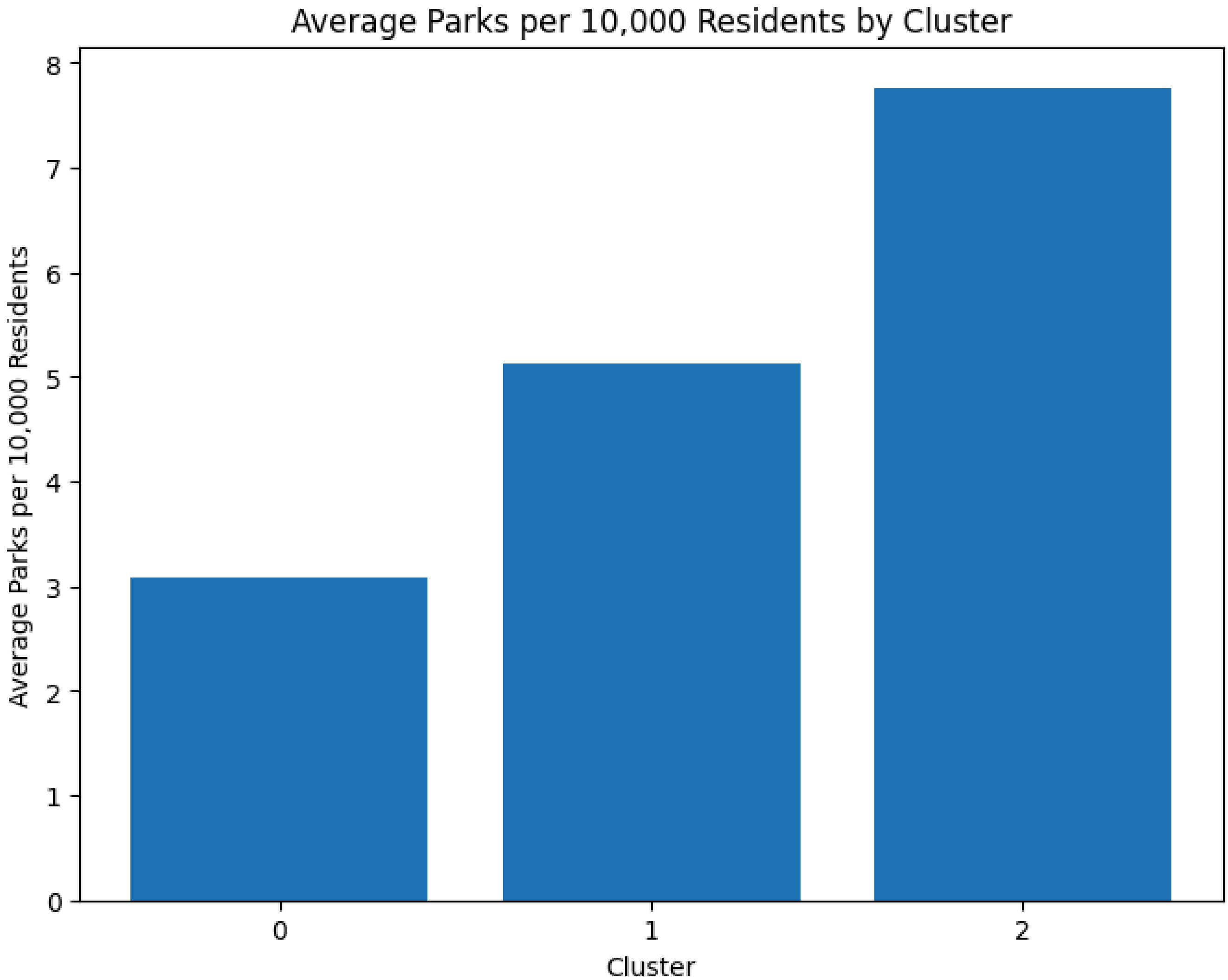
```
# Step 5
# Part 1

# Boxplot of distribution by cluster

plt.figure(figsize=(8,6))
df.boxplot(column="Population", by="AllNumericCluster")
plt.title("Distribution of Population by Cluster")
```

```
plt.suptitle("")
plt.xlabel("Cluster")
plt.ylabel("Population")
plt.show()
```

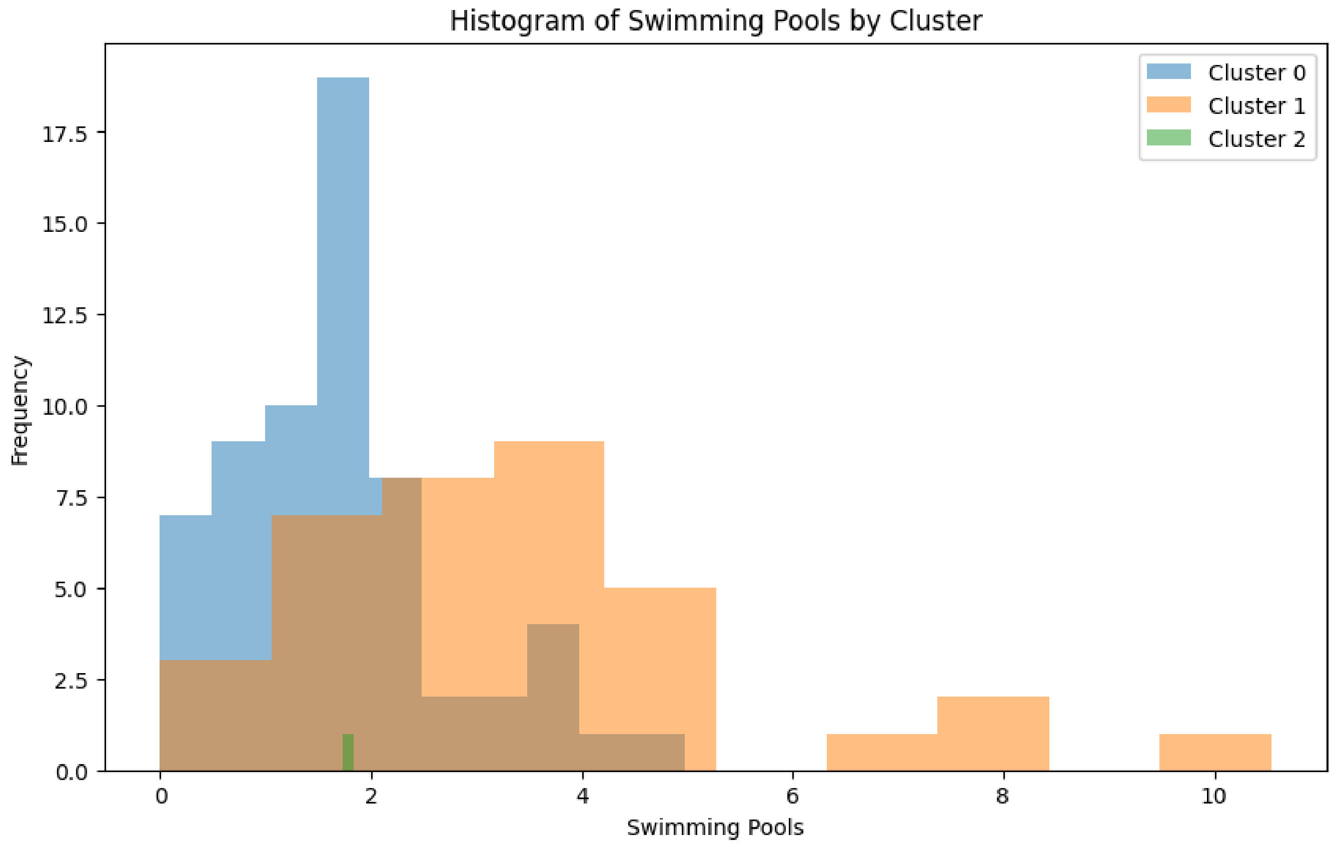
<Figure size 800x600 with 0 Axes>




```
# Bar charrt: Average parks per 10,000 residents by cluster

cluster_means = df.groupby("AllNumericCluster")["Parks per 10,000 residents"].mean()
plt.figure(figsize=(8,6))
plt.bar(cluster_means.index.astype(str), cluster_means.values)
plt.xlabel("Cluster")
plt.ylabel("Average Parks per 10,000 Residents")
plt.title("Average Parks per 10,000 Residents by Cluster")
plt.show()
```




```
# Historgram of swimming pools by cluster

clusters = sorted(df["AllNumericCluster"].unique())
plt.figure(figsize=(10, 6))
for cluster in clusters:
    subset = df[df["AllNumericCluster"] == cluster]
    plt.hist(subset["Swimming_pools"], alpha=0.5, label=f"Cluster {cluster}")
plt.xlabel("Swimming Pools")
plt.ylabel("Frequency")
plt.title("Histogram of Swimming Pools by Cluster")
plt.legend()
plt.show()
```



Since we don’t have an individual “Facilities” column, I created a new metric by summing several facility-related columns:

Facility-related columns:

"Fields/ Diamonds", "Tennis_dedicdated", "Pickleball_dedicated", "Pickleball_combined", "Hoops", "Community_garden_sites", "Dog_parks", "Playgrounds", "Rec_senior_centers", "Restrooms", "Skateparks", "Splashpads", "Swimming_pools", "Disc_golf_courses"

We then calculate Total_Facilities per 10,000 residents as:

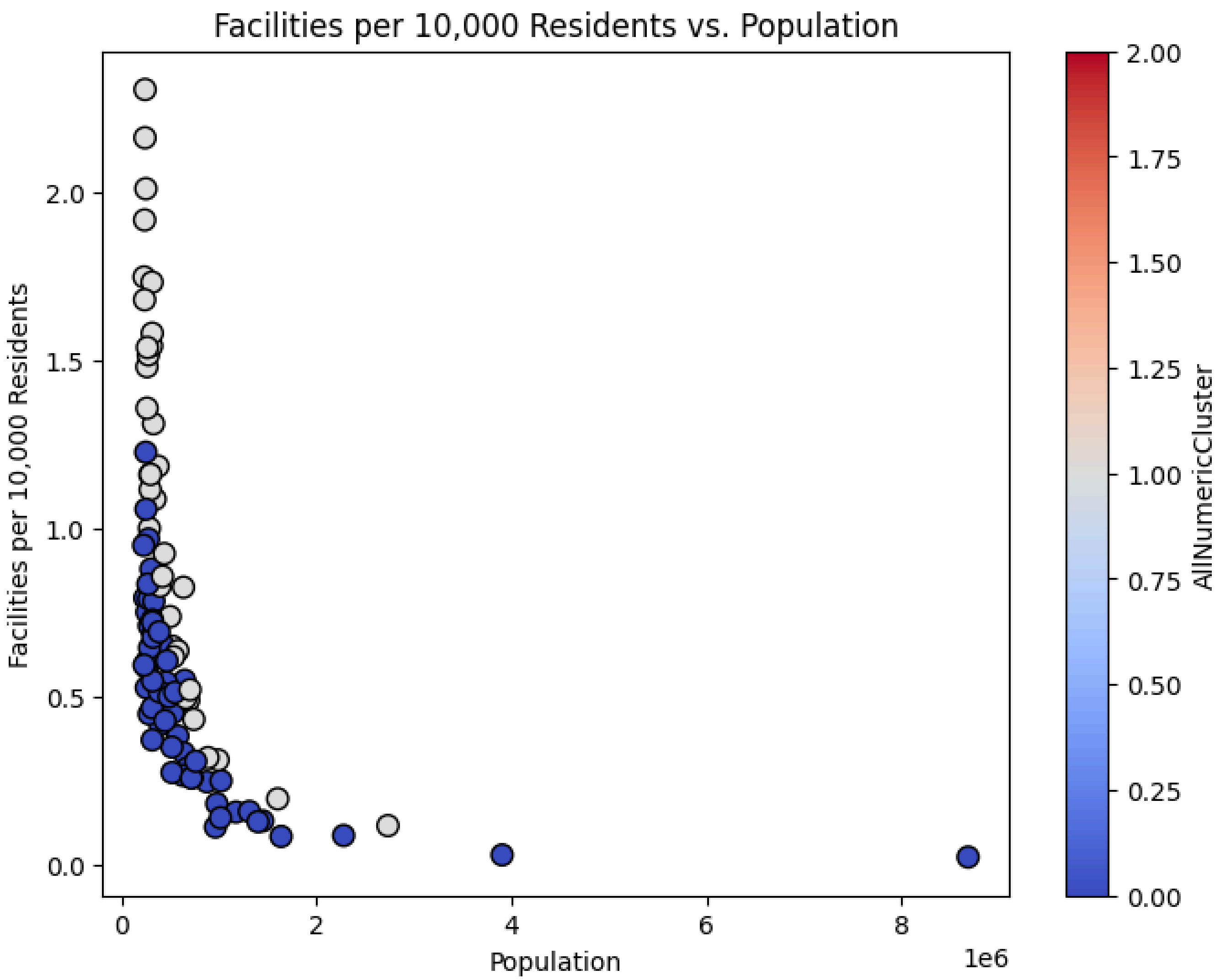
Total Facilities per 10,000 = Total Facilities/ Population × 10000

```
# Listing of columns that represent different facility counts
facility_cols = [
    "Fields/ Diamonds",
    "Tennis_dedicdated",
    "Pickleball_dedicated",
    "Pickleball_combined",
    "Hoops",
    "Community_garden_sites",
    "Dog_parks",
```

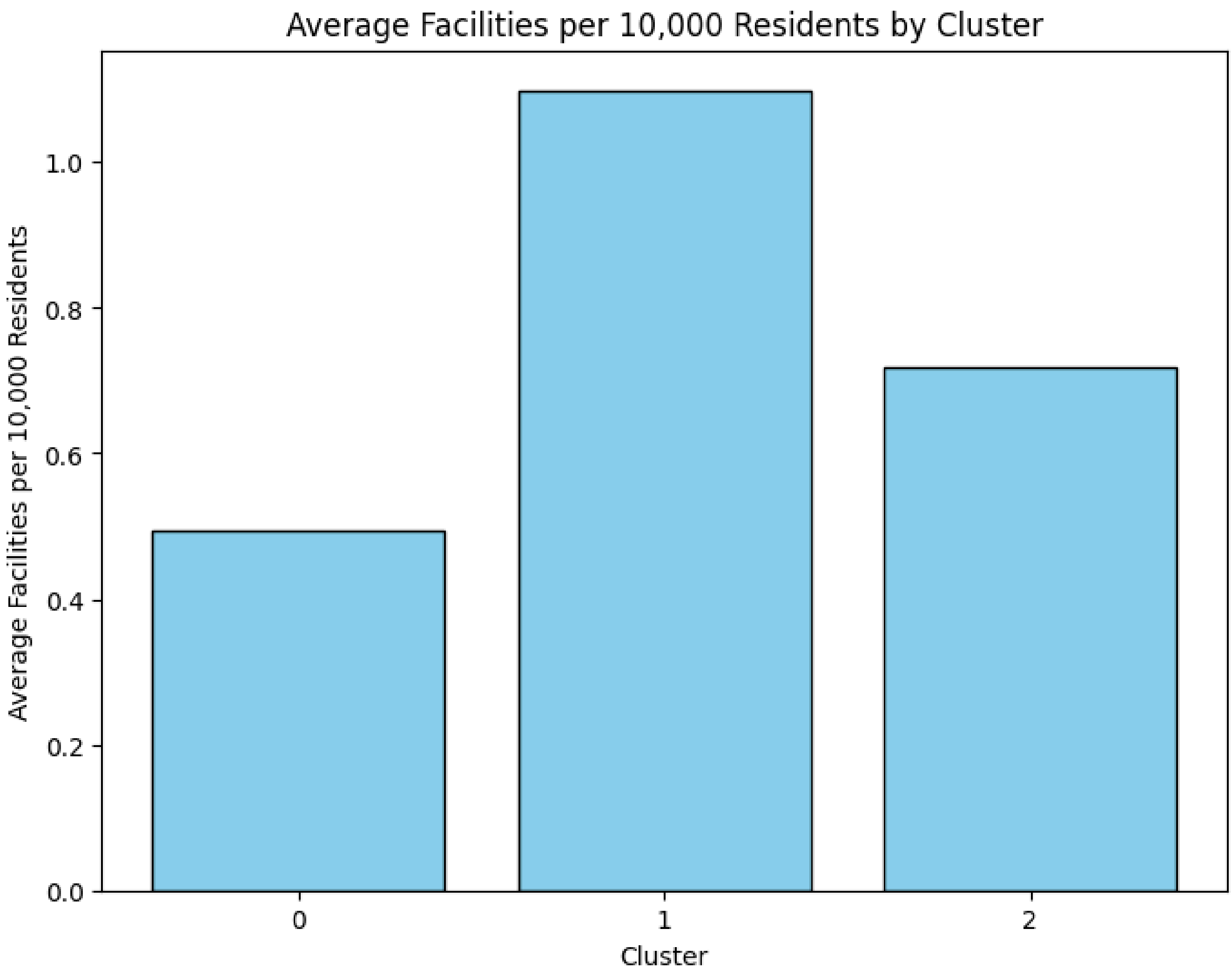
```
    "Playgrounds",
    "Rec_senior_centers",
    "Restrooms",
    "Skateparks",
    "Splashpads",
    "Swimming_pools",
    "Disc_golf_courses"
]

# Creating a new column for Total Facilities by summing all facility-related columns
df["Total_Facilities"] = df[facility_cols].sum(axis=1)

# Calculating Facilities per 10,000 residents (to normalize by population)
df["Facilities per 10,000 residents"] = (df["Total_Facilities"] / df["Population"])

# Displaying a few rows to confirm the new columns
df[["City", "Population", "Total_Facilities", "Facilities per 10,000 residents"]].he
```

	City	Population	Total_Facilities	Facilities per 10,000 residents
0	Albuquerque, NM	553345.0	26.529561	0.479440
1	Anaheim, CA	345538.0	16.623353	0.481086
2	Anchorage, AK	288464.0	20.713157	0.718050
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
# Visualizing it
# 1 scatterplot
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
scatter = plt.scatter(
    df["Population"],
    df["Facilities per 10,000 residents"],
    c=df["AllNumericCluster"],
    cmap="coolwarm",
    edgecolor="k",
    s=70
)
plt.xlabel("Population")
plt.ylabel("Facilities per 10,000 Residents")
plt.title("Facilities per 10,000 Residents vs. the recordedPopulation")
plt.colorbar(scatter, label="AllNumericCluster")
plt.show()
```



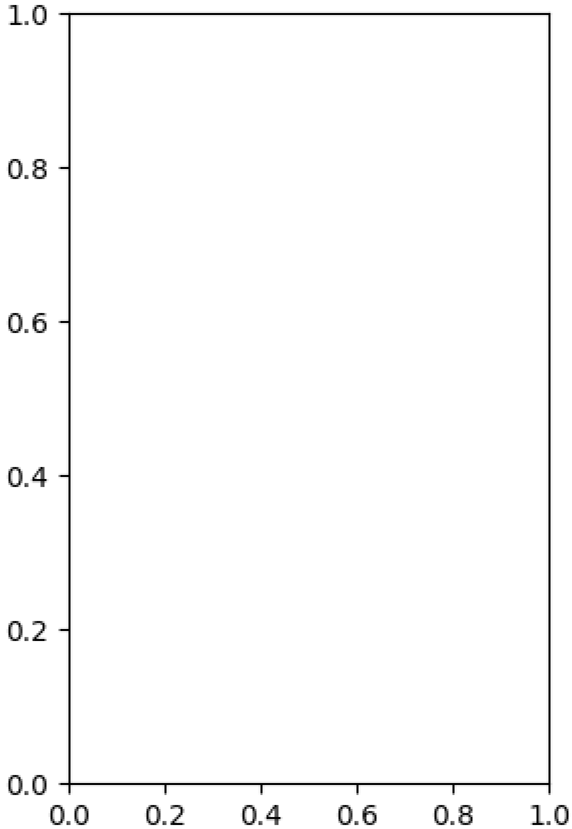

```
# Barchart

cluster_avg_facilities = df.groupby("AllNumericCluster")["Facilities per 10,000 resi

plt.figure(figsize=(8,6))
bars = plt.bar(cluster_avg_facilities.index.astype(str),
               cluster_avg_facilities.values,
               color="skyblue", edgecolor="k")
plt.xlabel("Cluster")
plt.ylabel("Average Facilities per 10,000 Residents")
plt.title("Average Facilities per 10,000 Residents by Cluster")
plt.show()

print("Average Facilities per 10,000 residents by cluster:")
print(cluster_avg_facilities)
```

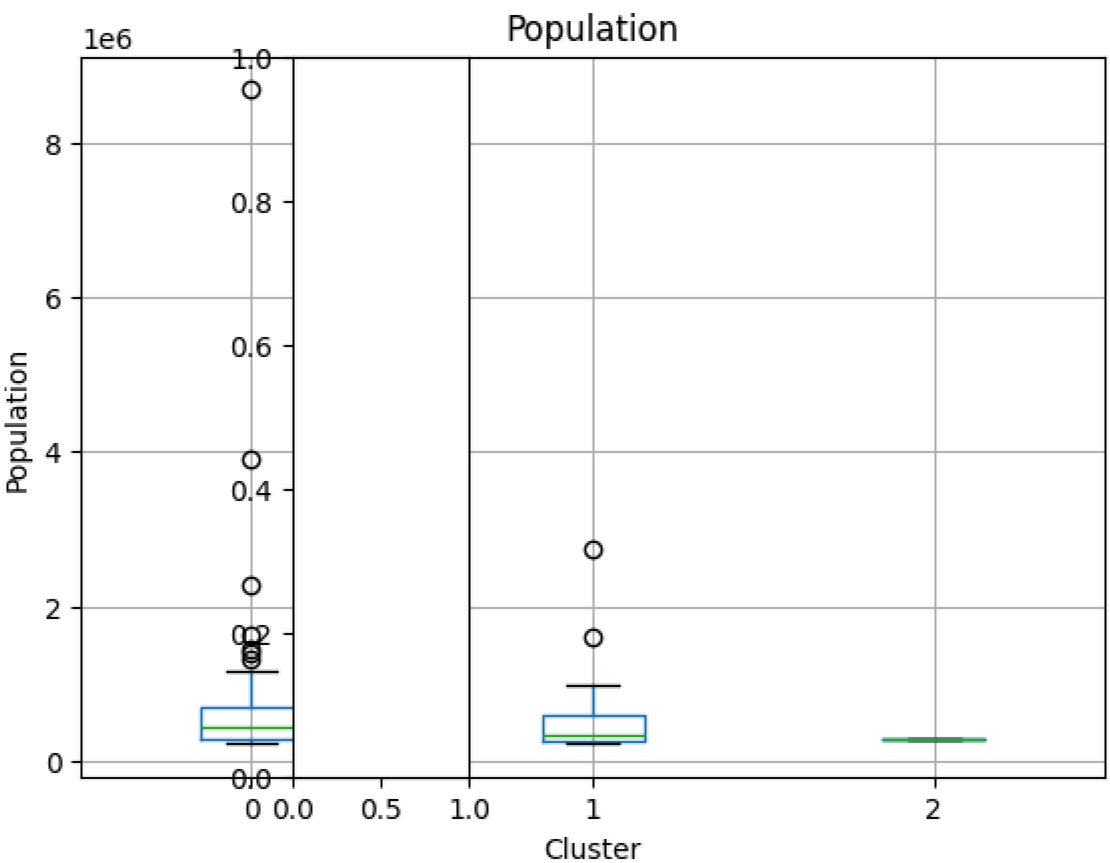


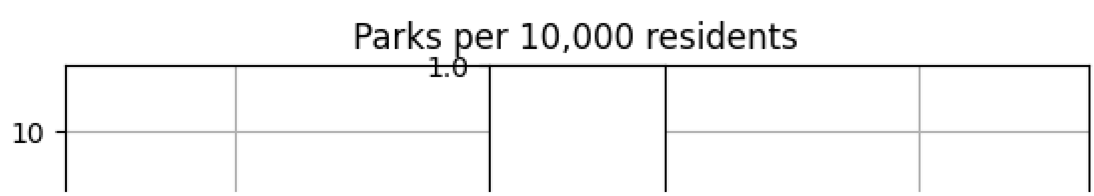

```
Average Facilities per 10,000 residents by cluster:
AllNumericCluster
0    0.493647
1    1.098971
2    0.718050
Name: Facilities per 10,000 residents, dtype: float64
```

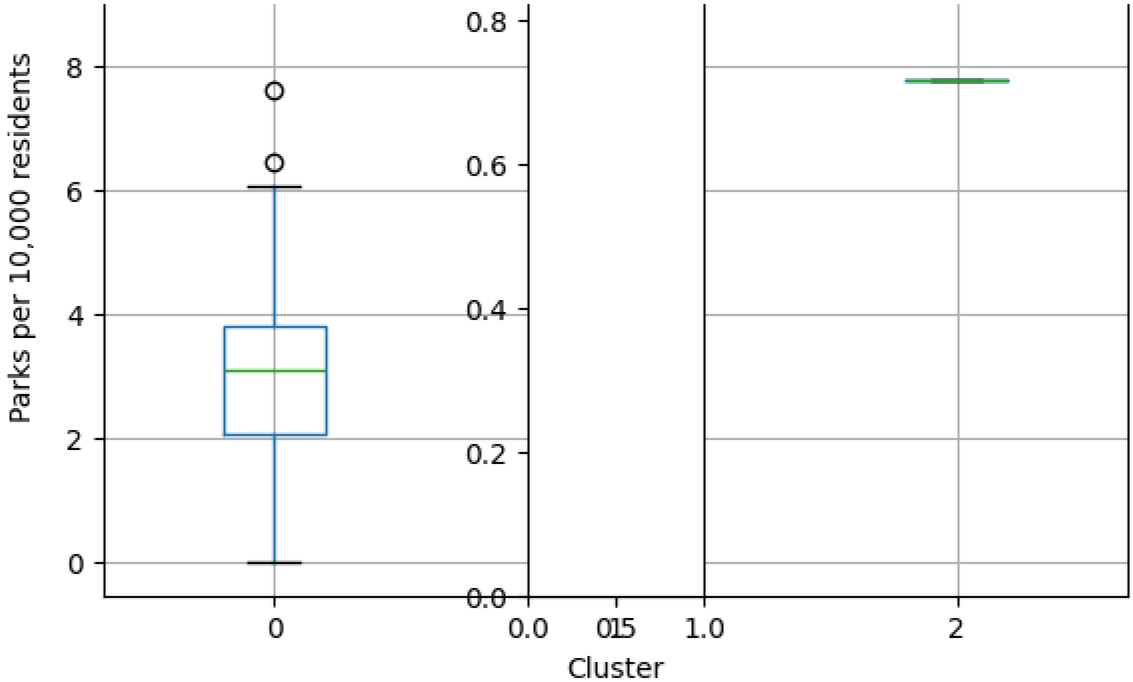
```
# Boxplot of features across all

# Defining the list of numeric features you want to explore
numeric_vars = [
    "Population",
    "Parks per 10,000 residents",
    "Acres per 1,000 people",
    "investment_dollars",
    "Facilities per 10,000 residents"
]

# Number of features
num_plots = len(numeric_vars)

# Creating subplots for each numeric feature's distribution by cluster
```

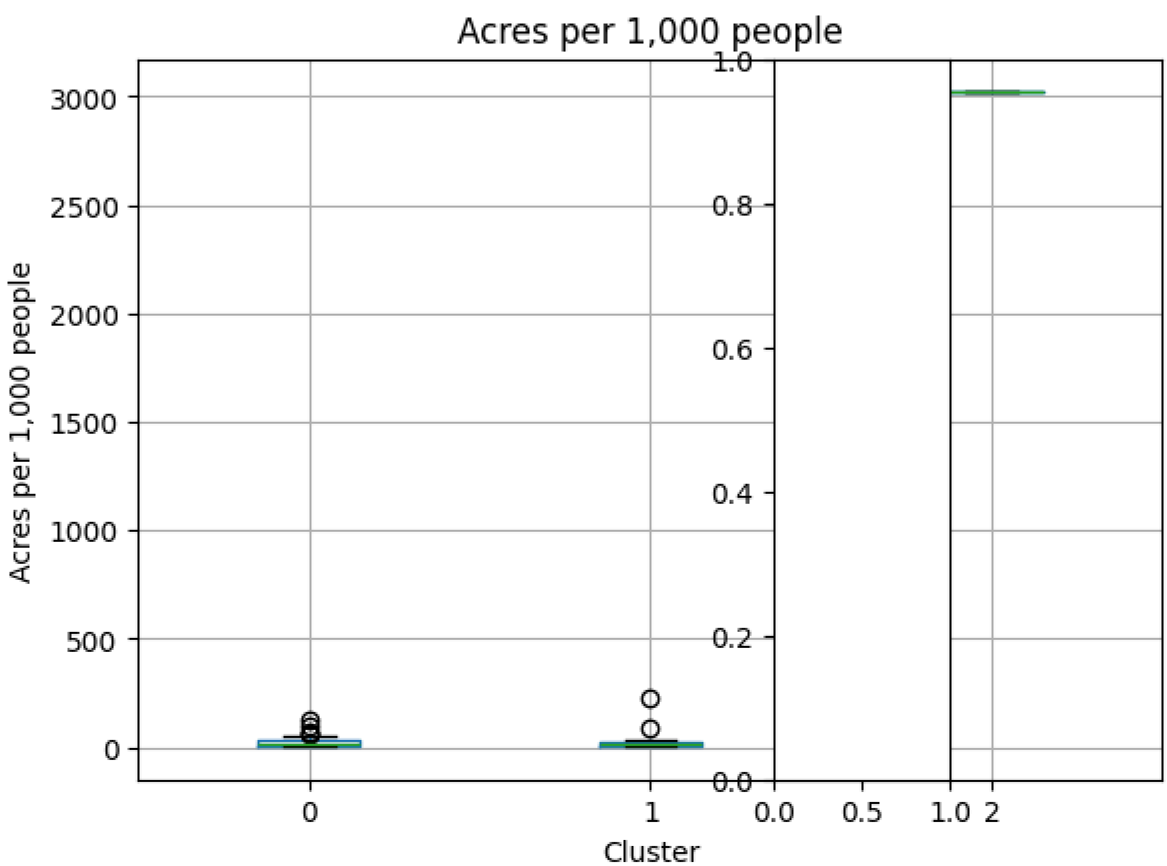
```
plt.figure(figsize=(18, 5))

for i, var in enumerate(numeric_vars, 1):
    plt.subplot(1, num_plots, i)
    df.boxplot(column=var, by="AllNumericCluster")
    plt.title(var)
    plt.xlabel("Cluster")
    plt.ylabel(var)
    # Remove the automatic subtitle that pandas adds
    plt.suptitle("")

plt.tight_layout()
plt.show()
```

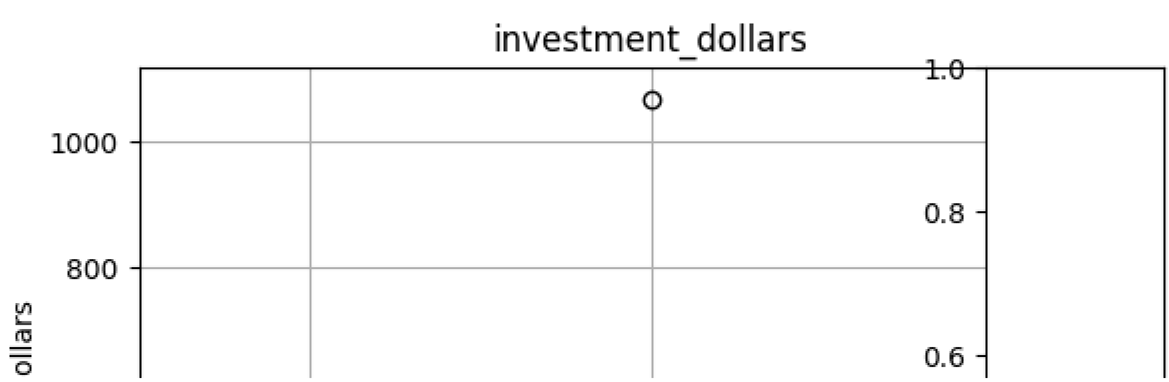

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

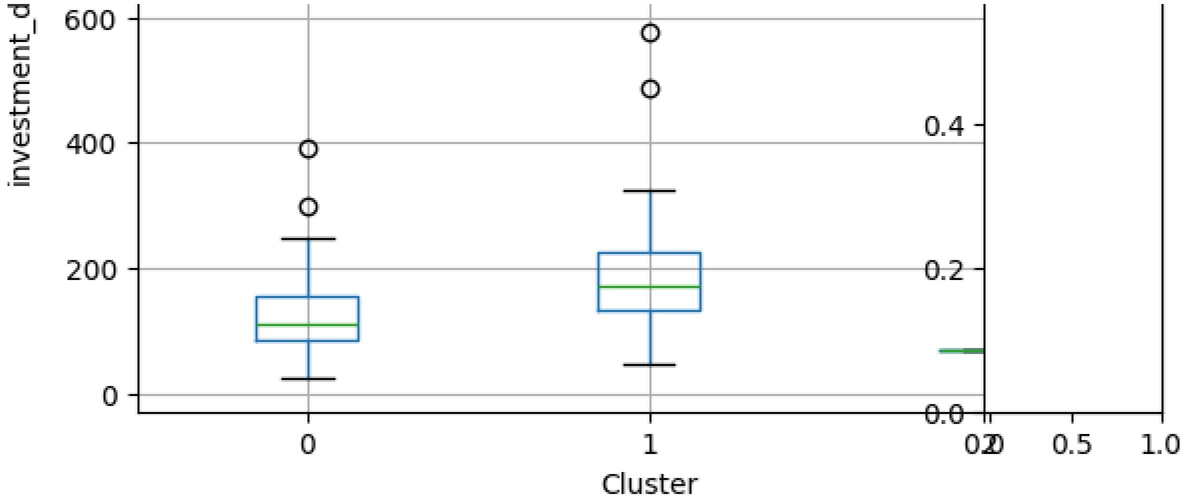

Population
1e6
Population
Cluster
0
1
2


Parks per 10,000 residents
10

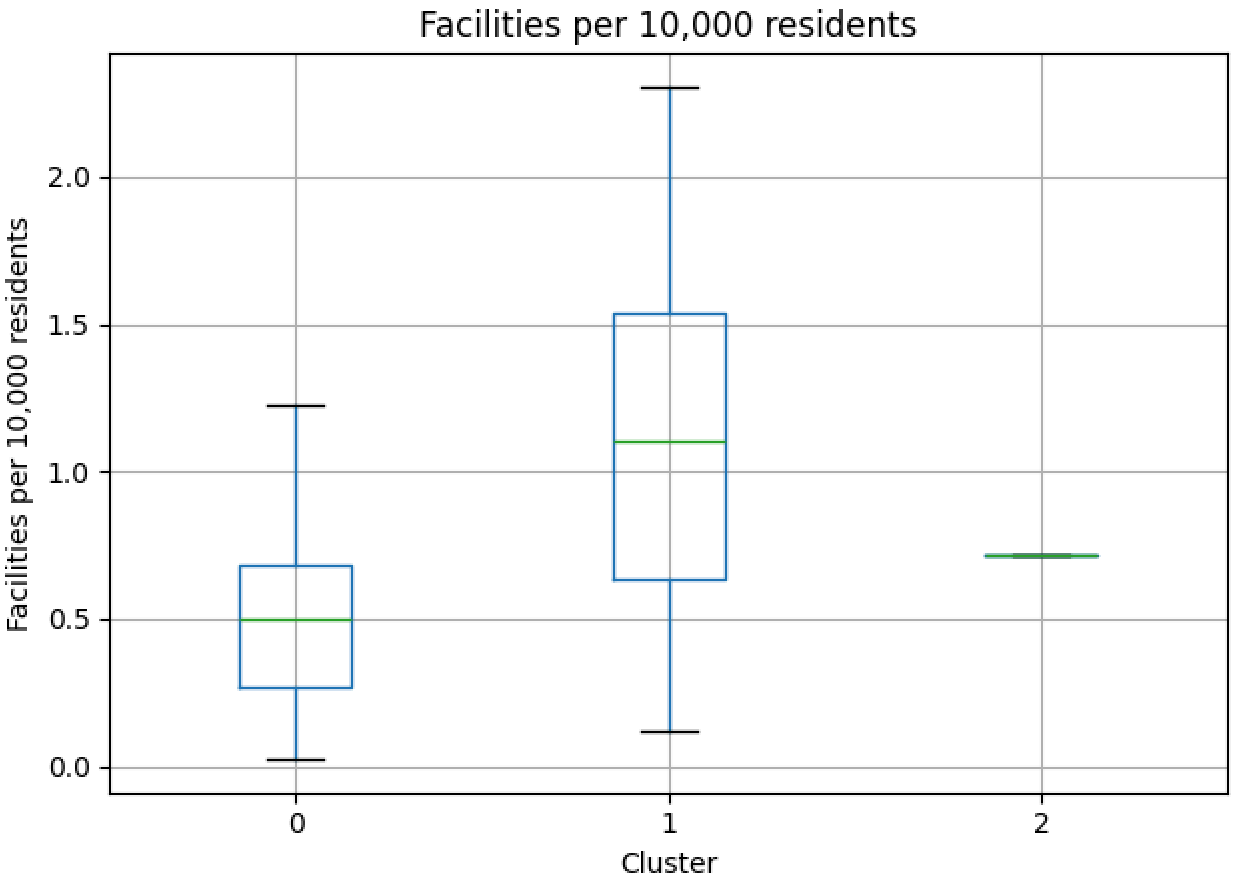

Parks per 10,000 residents
Cluster


Acres per 1,000 people
Acres per 1,000 people
Cluster


investment_dollars
ollars



investment_d
600
400
200
0
0
1
Cluster
0.4
0.2
0.0
0.0
0.5
1.0



Facilities per 10,000 residents
Facilities per 10,000 residents
2.0
1.5
1.0
0.5
0.0
0
1
2
Cluster

The scatter plot shows that as population increases, facilities per 10,000 residents tend to drop, which is interesting however it ends up suggesting that larger cities might struggle to keep up on a per-capita basis. The bar chart then clearly highlights which clusters have, on average, higher or lower facilities per 10,000 residents, making it easy to see which groups are better served and developed. Finally, the box plots break down the distributions of key metrics—like population and park acreage—across clusters, revealing not only typical values but also the range and outliers, which gives a more detailed picture of the diversity within each single cluster.

Step 7

Through my analysis, I discovered that the clustering techniques reveal meaningful differences among the different recordd cities: the hierarchical clustering using “Parks per 10,000 residents” and “Acres per 1,000 people” grouped cities in a way that highlighted variations in park availability and size, while the k-means clustering on “Population” and “investment_dollars” underscored differences in city scale and financial commitment that was being made to parks. Clustering on all numeric features further refined these insights, showing nuanced profiles where larger cities often have lower facilities per capita (which was an interesting finding in my opinioN). The extended exploration—with scatter, bar, and box plots of the newly derived “Facilities per 10,000 residents” metric—confirmed that some smaller cities tend to provide more recreational resources per resident compared to their larger counterparts, offering a comprehensive picture of how urban planning, investment, and resource allocation vary across the public land services dataset.

Part 2

The question I wanted to explore is

Do cities that invest more in parks and recreation on a per capita basis tend to provide more recreational facilities per resident?

To investigate this, I first tried to created a new metric - Investment per 10k Residents - by normalizing the raw investment dollars by each city’s population. I then compared this with the already derived Facilities per 10,000 Residents metric using a scatterplot, with points colored by their previously determined cluster.

From this scatterplot, we see that cities with higher **Investment per 10k Residents** often have higher **Facilities per 10,000 Residents**, suggesting a generally positive relationship between per-capita funding and facility availability. However, there are clear outliers—some cities have relatively high investment but do not provide proportionally more facilities, while others manage more facilities despite lower investment levels. Coloring by cluster also reveals that certain groups of