LAB SUBMISSION Unsupervised learning: Clustering. Detect suspicious consumptions from households.

Important Information:

- This lab report must be completed **individually**.
- Please upload your completed report to Atenea.
- You have until 9th january to submit the report.

Objective: The goal of this exercise is to identify households with atypical or suspicious consumption patterns compared to other users. These unusual patterns may not always indicate fraud but could be linked to system malfunctions (e.g., water leaks in toilets causing abnormally high water usage) or discrepancies such as a higher actual number of residents than recorded.

By detecting these anomalies, we aim to uncover potential issues and inefficiencies, allowing for timely interventions. The objective is to develop a clustering model to identify households exhibiting these atypical consumption behaviors. Analyze the results to determine which features contribute most to the detection of anomalies.

Dataset Description

The dataset is located in: `Data/Labsubmission_suspicious_consumption_dataset.xlsx`

This shyntetic dataset for identifying suspicious consumption patterns in households includes the following features:

- `Household_ID`: Unique identifier for each household.
- `Daily_Electricity_kWh`: Daily electricity consumption in kWh.
- `Daily_Water_Liters`: Daily water consumption in liters.
- `Daily_Gas_CubicMeters`: Daily gas consumption in cubic meters.
- `Num_Residents`: Number of residents in the household.
- `Temperature_C`: Average daily temperature in Celsius.
- `Humidity_Percent`: Average daily humidity in percentage.

Import libraries

In [49]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

```
# To suppress all warnings
warnings.filterwarnings("ignore")
```

Load dataset

In [50]:
```
url = 'https://raw.githubusercontent.com/sbarja/course2425_datascience4energysys
dataset = pd.read_excel(url)
```

In [51]:
```
dataset.head(5)
```

Out[51]:

	Household_ID	Daily_Electricity_kWh	Daily_Water_Liters	Daily_Gas_CubicMeters	Num
0	1	34.967142	219.967772	3.649643	
1	2	28.617357	196.231684	4.710963	
2	3	36.476885	152.981518	3.415160	
3	4	45.230299	117.653161	4.384077	
4	5	27.658466	184.911166	1.212771	

In [52]:
```
print(dataset.dtypes)
print()
print(dataset.isna().sum())
```

```
Household_ID              int64
Daily_Electricity_kWh    float64
Daily_Water_Liters       float64
Daily_Gas_CubicMeters    float64
Num_Residents              int64
Temperature_C            float64
Humidity_Percent         float64
dtype: object

Household_ID             0
Daily_Electricity_kWh    0
Daily_Water_Liters       0
Daily_Gas_CubicMeters    0
Num_Residents            0
Temperature_C            0
Humidity_Percent         0
dtype: int64
```

1. Understanding the data

It is necessary to visualize and understand the data we are going to work with, as well as to know its characteristics.

- How many households are in the dataset? How many attributes has each household?
- Is there any missing data?

- Statistical summary of the input data set (Statistics Table)

How many households that the dataset have? How many features there are?

In [53]:
```
print(len(dataset))
print()
print(len(dataset.columns))
print('Actually there are 6 features considering one column is Household_ID')
```

```
1000

7
Actually there are 6 features considering one column is Household_ID
```

Is there any missing data?

In [54]:
```
dataset.isna().sum()
```

Out[54]:
```
Household_ID             0
Daily_Electricity_kWh    0
Daily_Water_Liters       0
Daily_Gas_CubicMeters    0
Num_Residents            0
Temperature_C            0
Humidity_Percent         0
dtype: int64
```

Statistical summary of the data set.

In [55]:
```
dataset.describe()
```

Out[55]:

	Household_ID	Daily_Electricity_kWh	Daily_Water_Liters	Daily_Gas_CubicMeters	I
count	1000.000000	1000.000000	1000.000000	1000.000000	
mean	500.500000	37.420246	191.867931	6.199373	
std	288.819436	27.322164	143.810457	4.658273	
min	1.000000	-2.412673	2.980568	-1.039024	
25%	250.750000	24.371303	123.501764	3.857672	
50%	500.500000	31.520594	159.724794	5.315209	
75%	750.250000	39.175583	200.650271	6.776582	
max	1000.000000	253.381810	1146.323523	34.284502	

Set the household_ID as the index

In [56]:
```
dataset.set_index('Household_ID', inplace=True)
```

In [57]:
```
dataset.head(3)
```

Out[57]:

Household_ID	Daily_Electricity_kWh	Daily_Water_Liters	Daily_Gas_CubicMeters	Num_Re
1	34.967142	219.967772	3.649643	
2	28.617357	196.231684	4.710963	
3	36.476885	152.981518	3.415160	

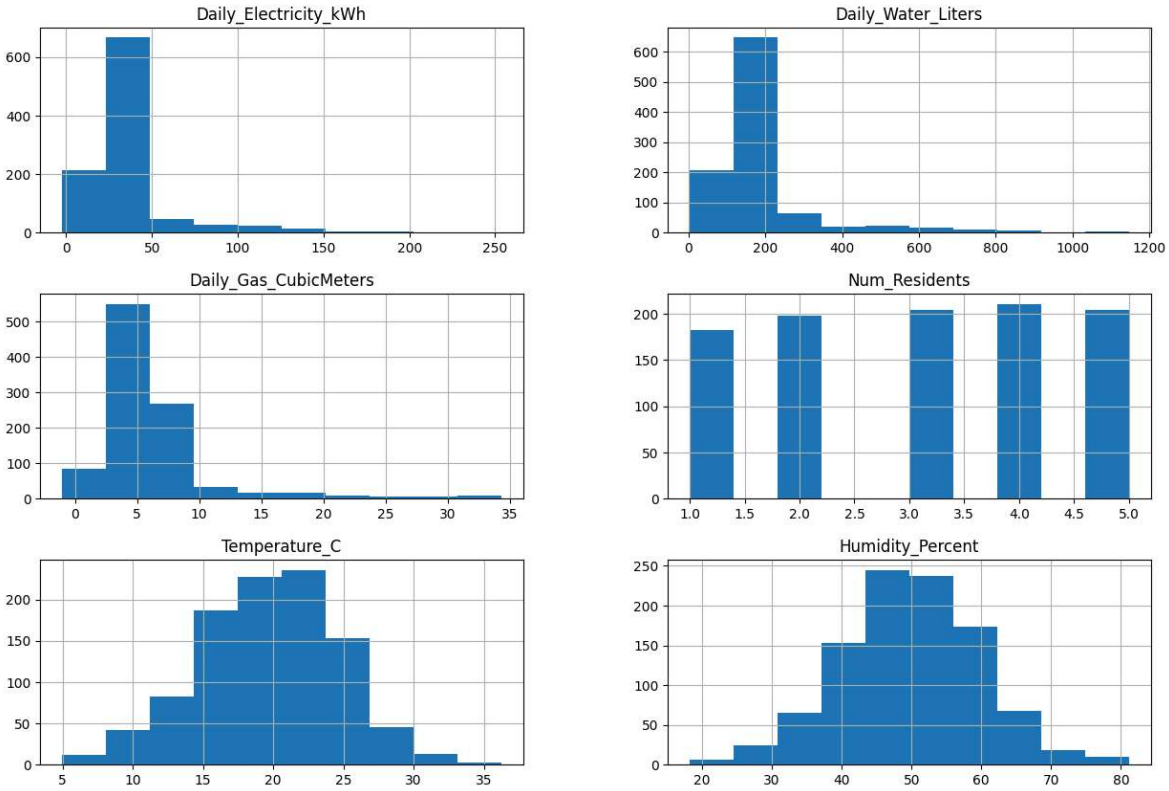
2. Visualize the data.

A visual way to understand the input data.

- Histogram
- Density curve
- Boxplots
- Scatter plots
- Correlation matrix

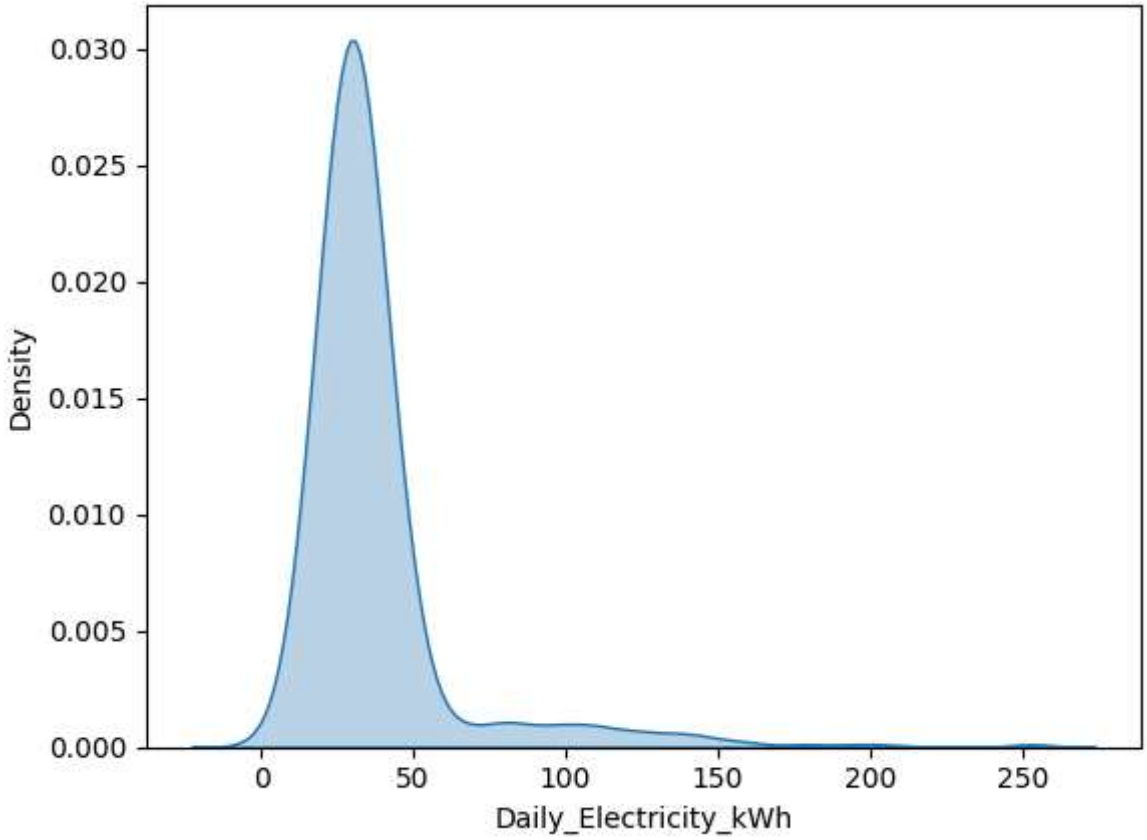
Plot the histogram of each feature

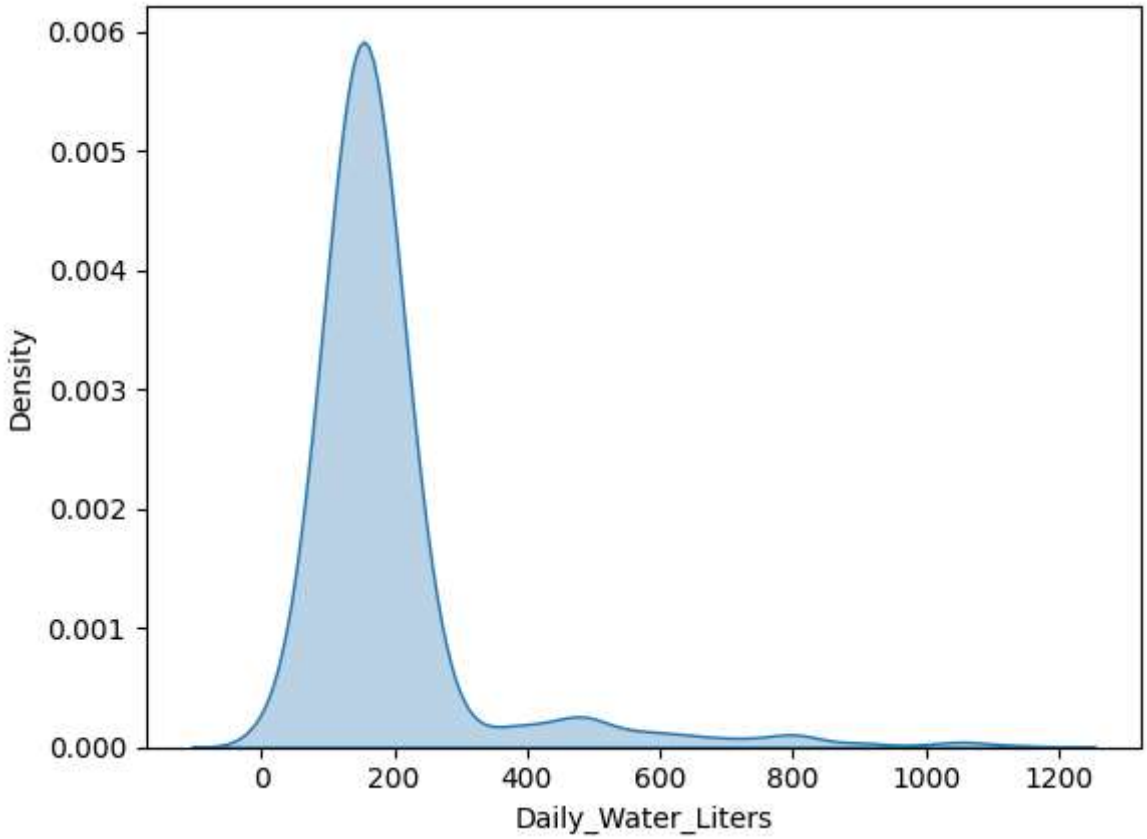
In [58]:
```
#Histogram plot
dataset.hist(figsize=(15,10))
plt.show()
```

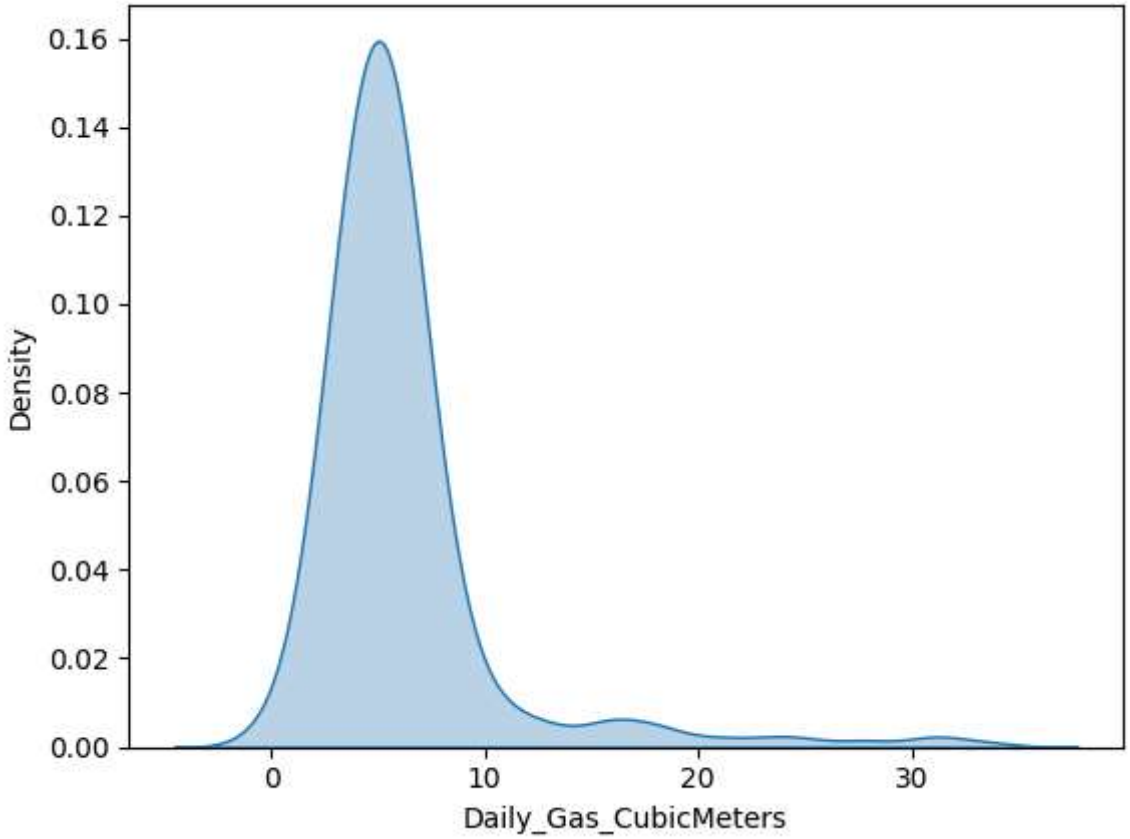


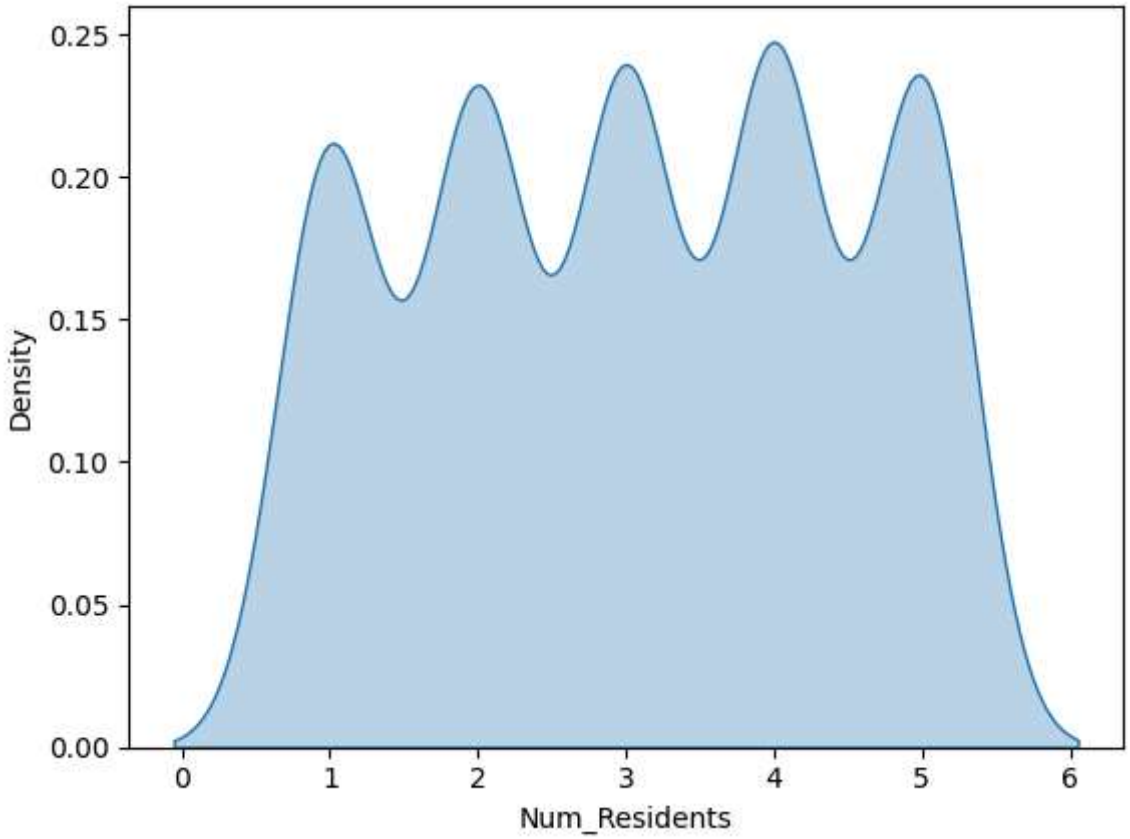
Plot the density plot of each feature

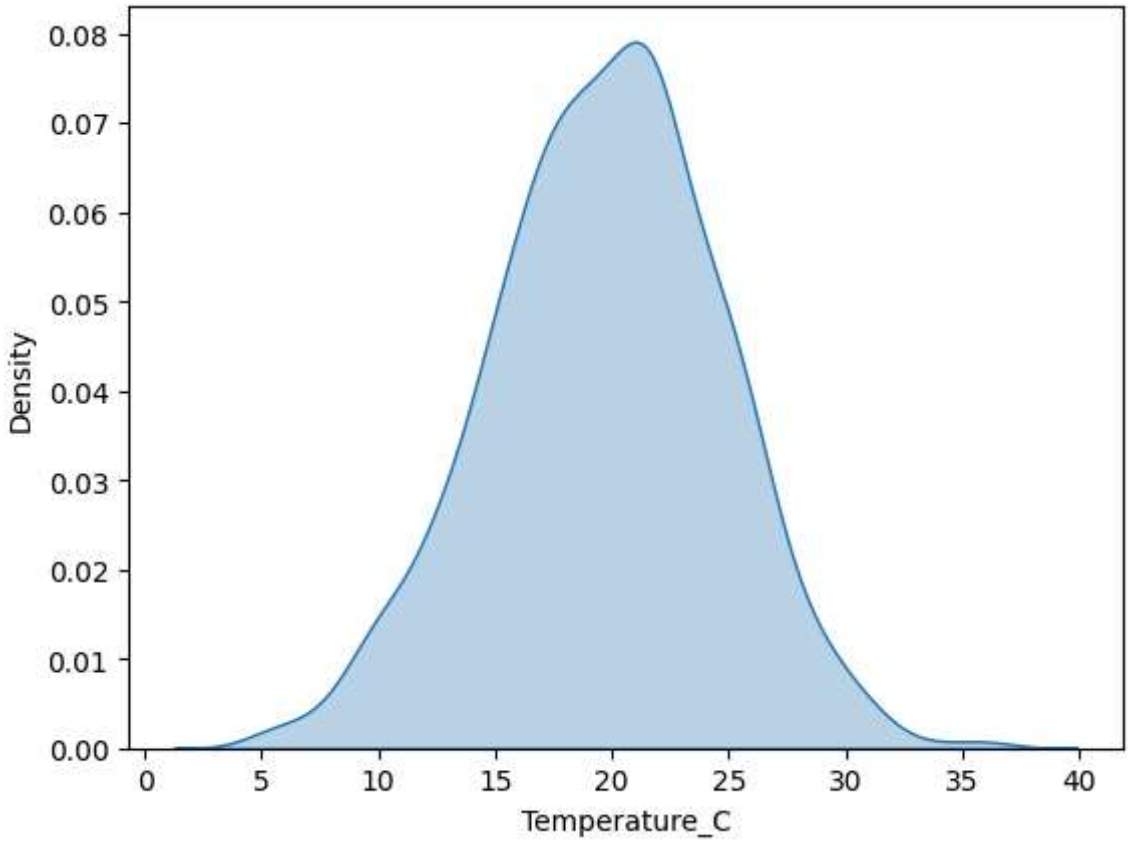
In [59]:
```
#Density plot

variables = dataset.columns

for variables in variables:
  sns.kdeplot(data=dataset, x=variables, fill=True, alpha=0.3, label='Overall Di
  plt.show()
```

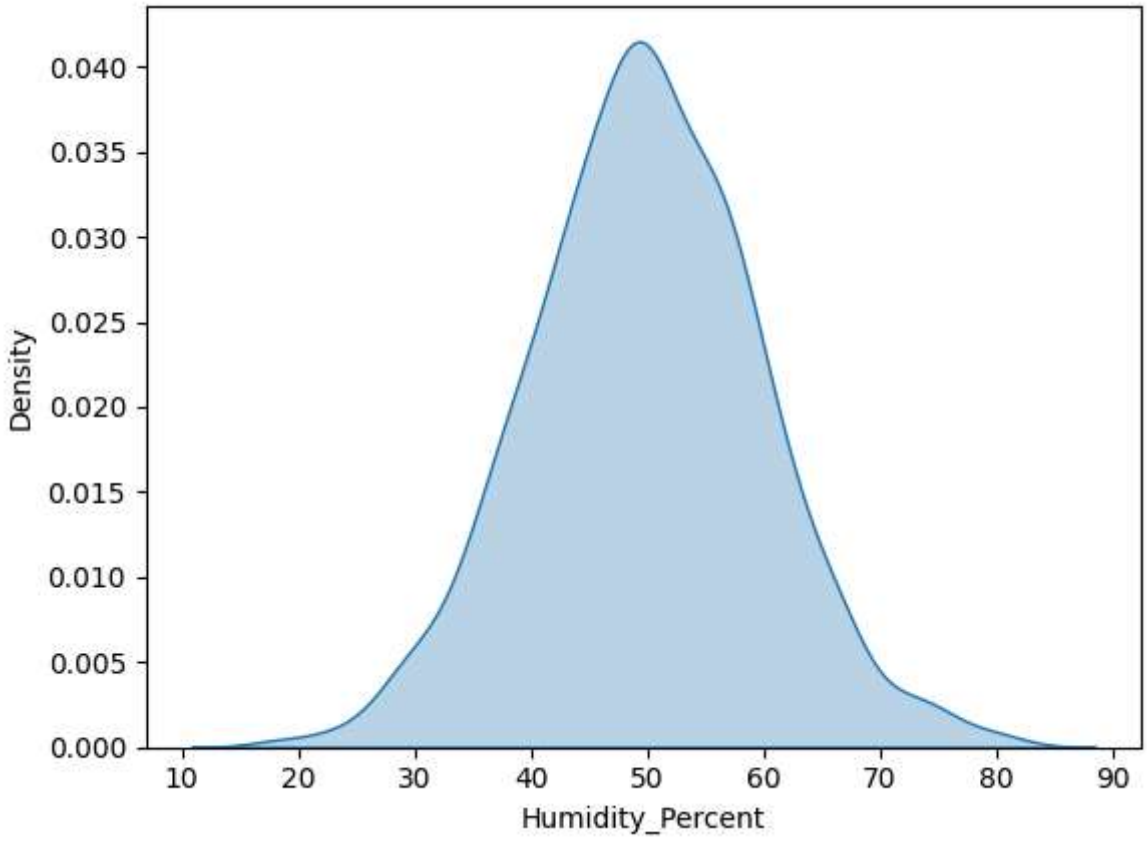

0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
0
50
100
150
200
250
Daily_Electricity_kWh


0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
0
200
400
600
800
1000
1200
Daily_Water_Liters


0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0
10
20
30
Daily_Gas_CubicMeters

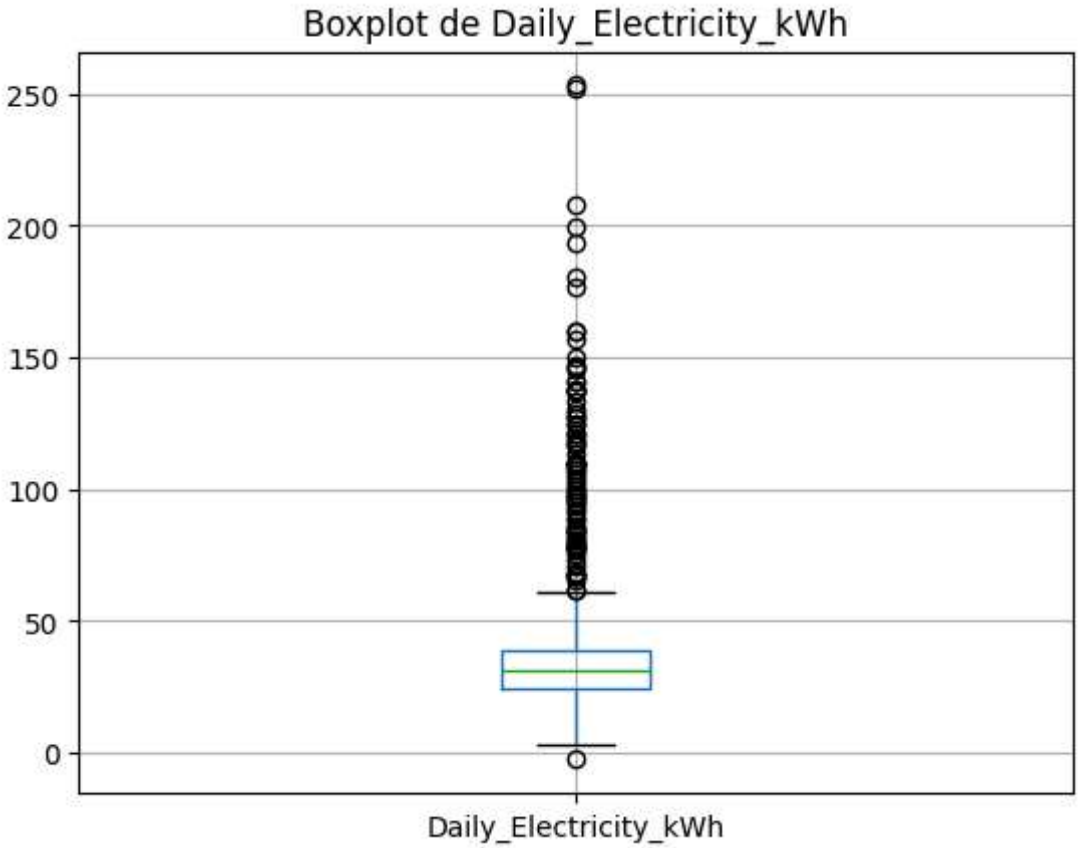

0.25
0.20
0.15
0.10
0.05
0.00
Density
0
1
2
3
4
5
6
Num_Residents

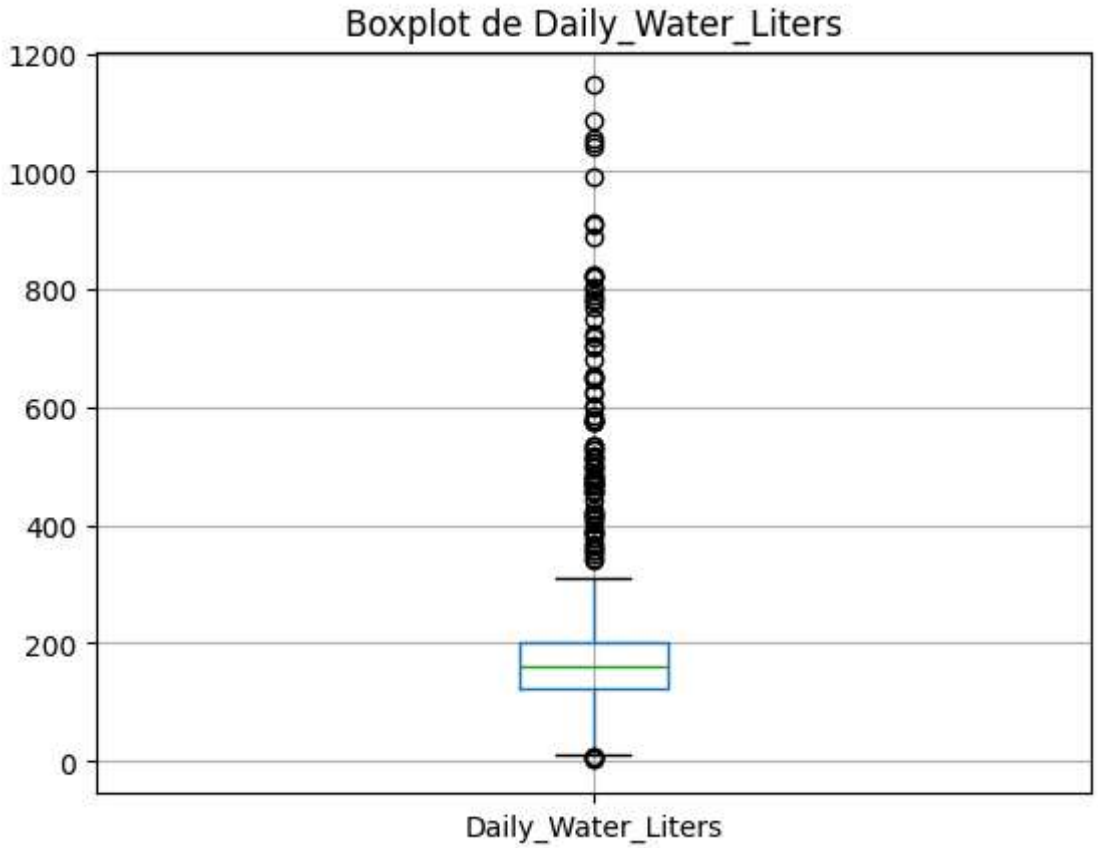


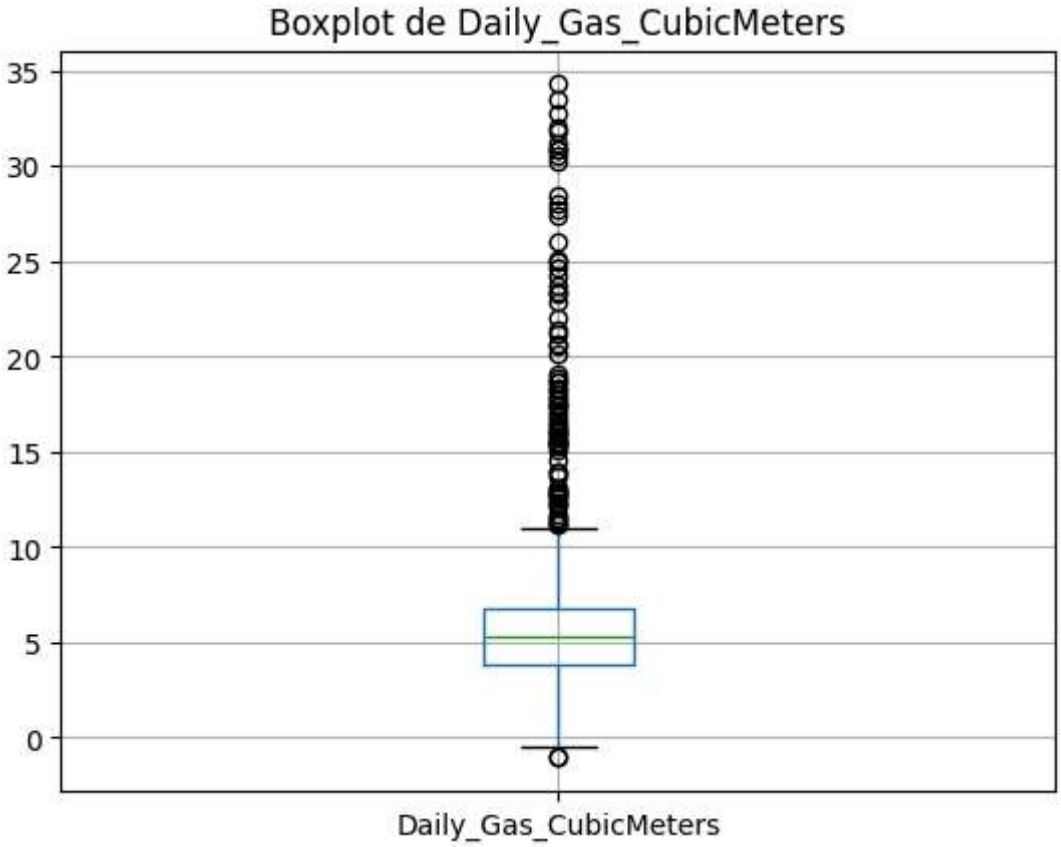


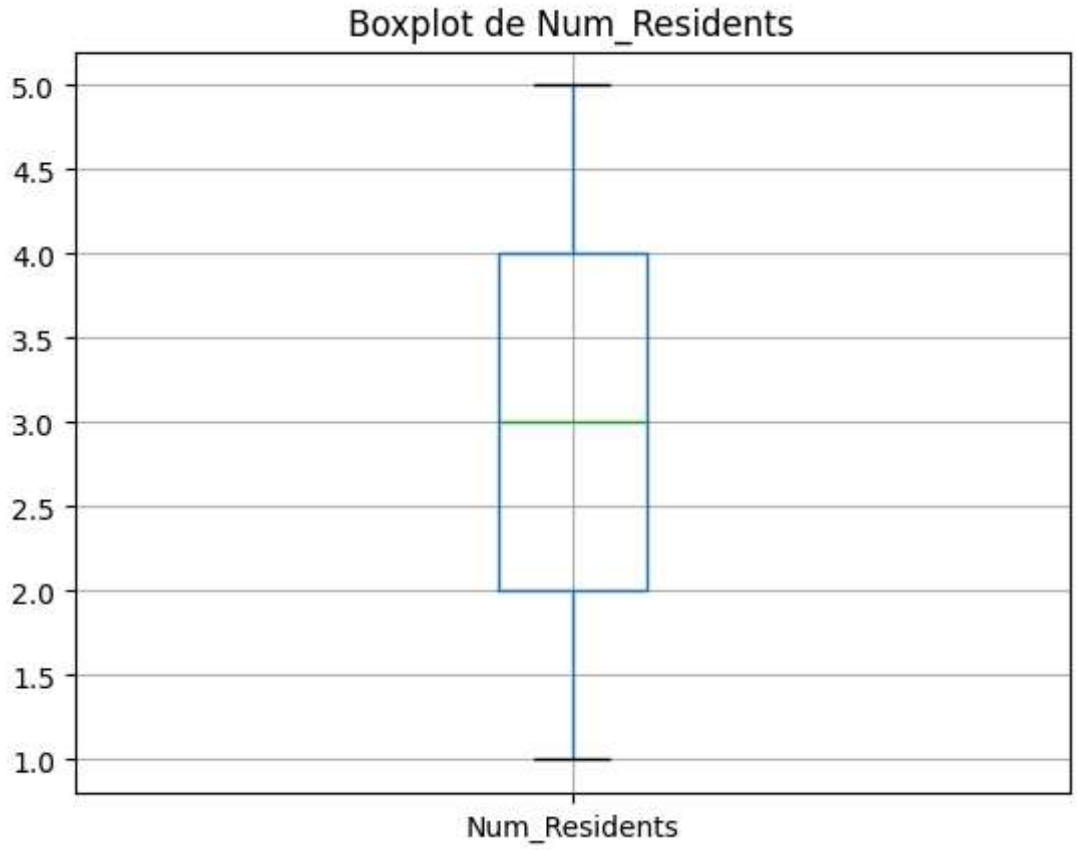
Boxplot of each feature. Should we delete the outliers?

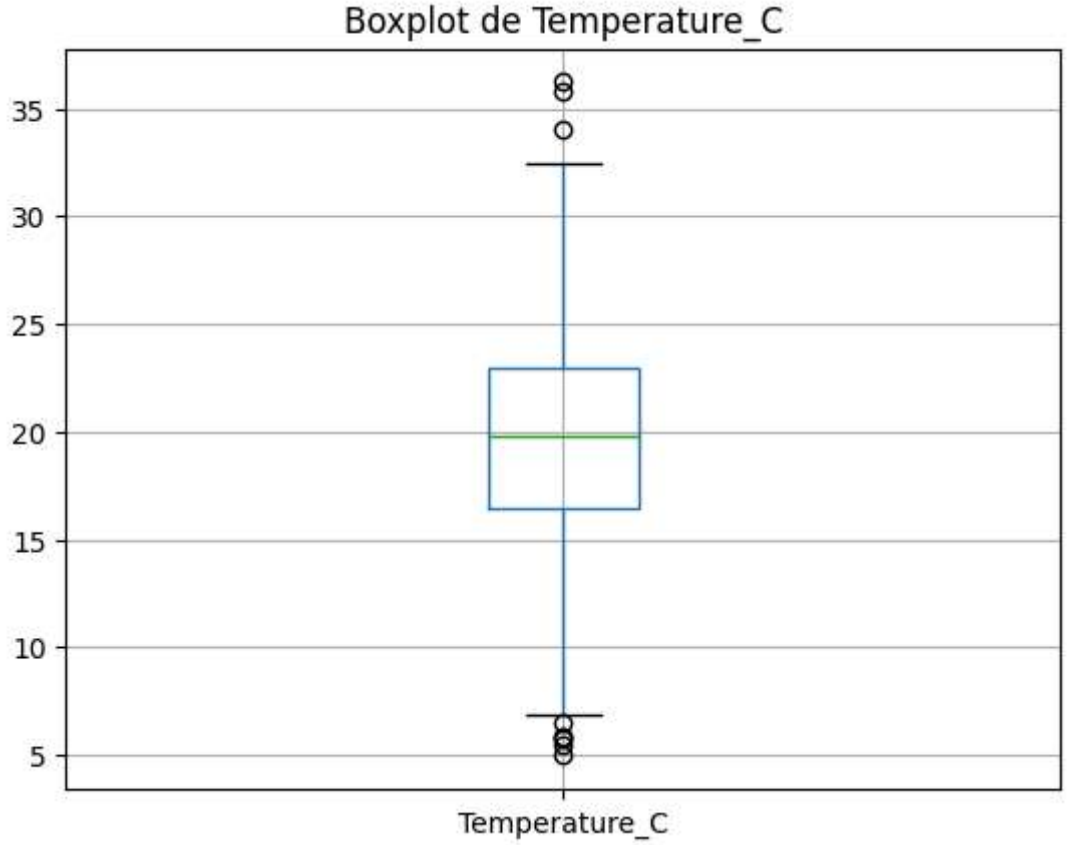
In [60]:
```
#Boxplots

variables = dataset.columns

for var in variables:
    dataset.boxplot(column=var)
    plt.title(f'Boxplot de {var}')
    plt.show()
```

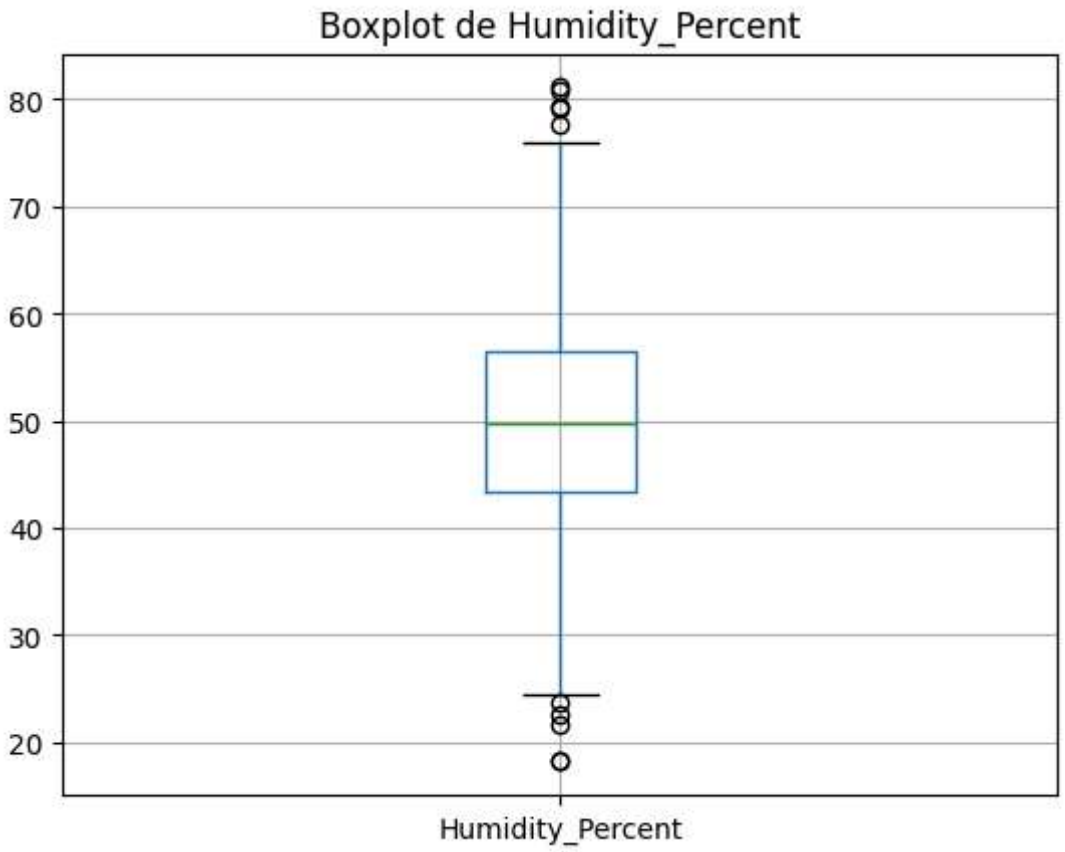



Boxplot de Daily_Water_Liters
1200
1000
800
600
400
200
0
Daily_Water_Liters


Boxplot de Daily_Gas_CubicMeters
35
30
25
20
15
10
5
0
Daily_Gas_CubicMeters

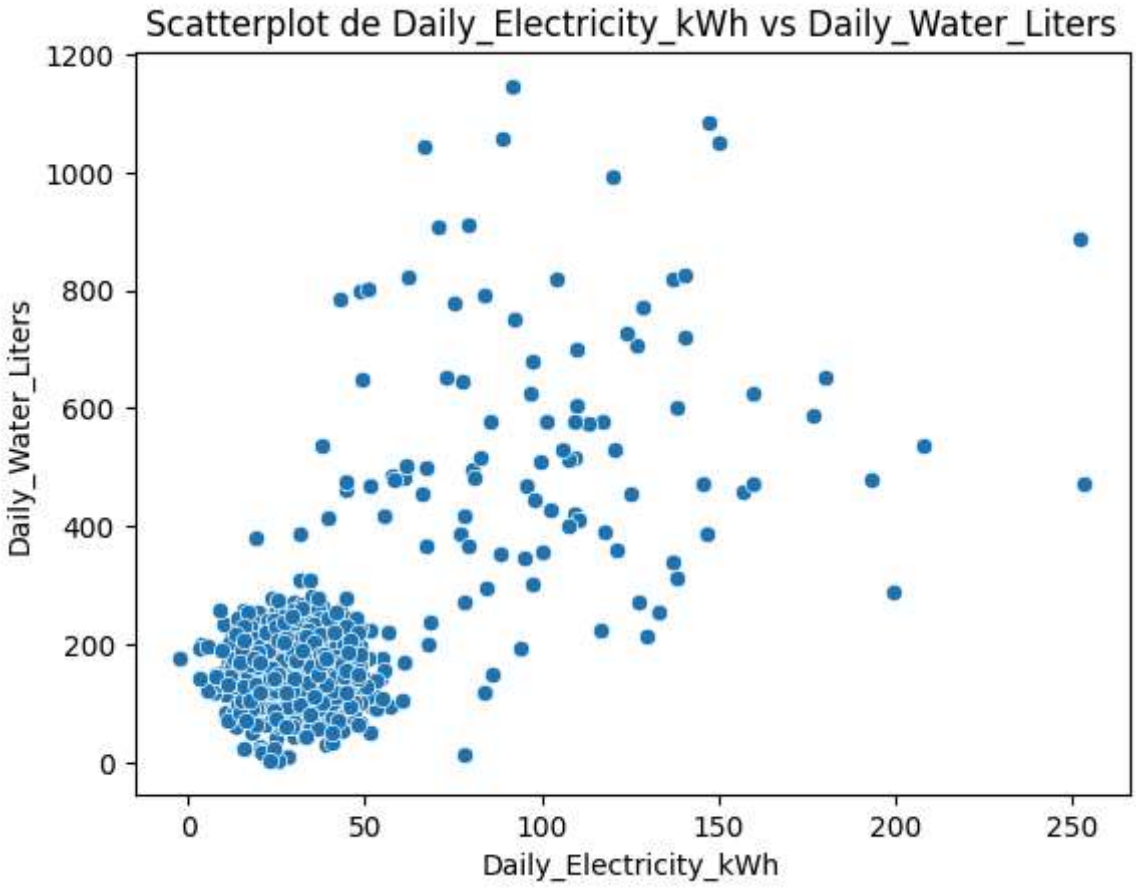

Boxplot de Num_Residents
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Num_Residents

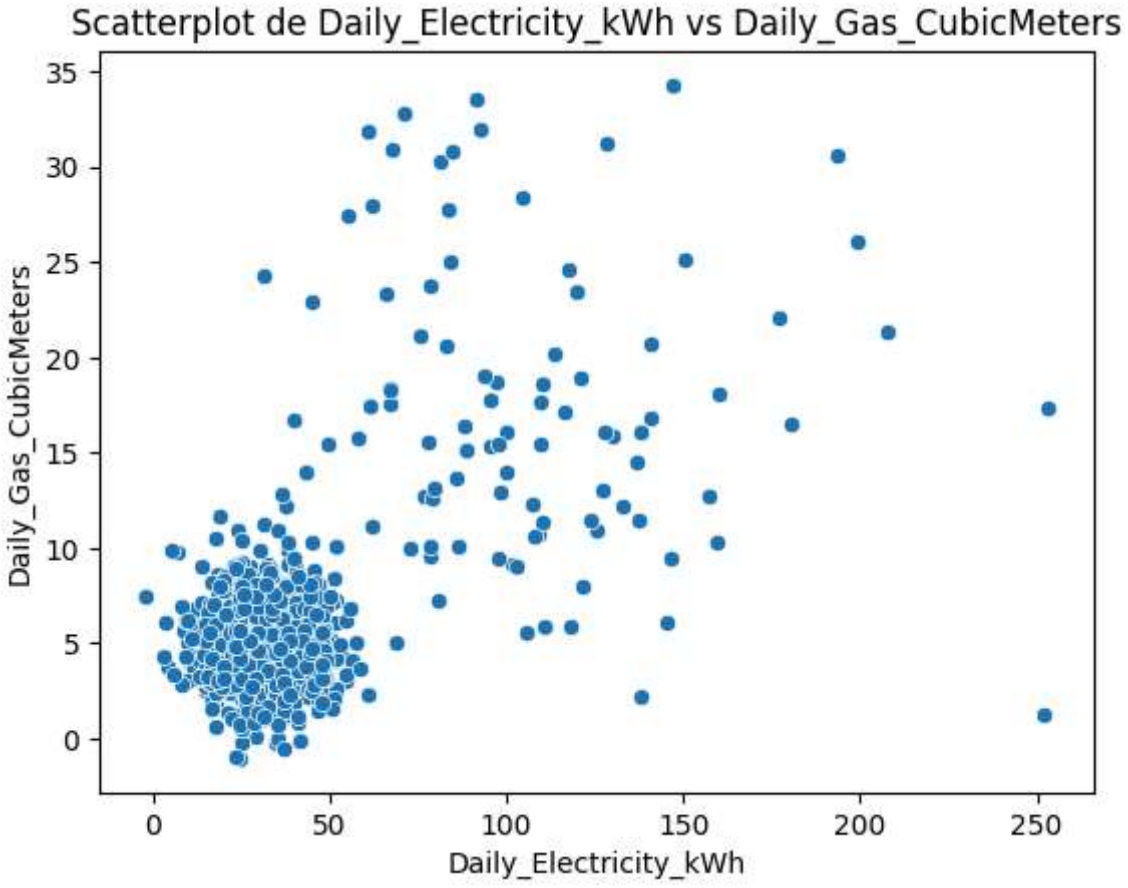

Boxplot de Temperature_C
35
30
25
20
15
10
5
Temperature_C

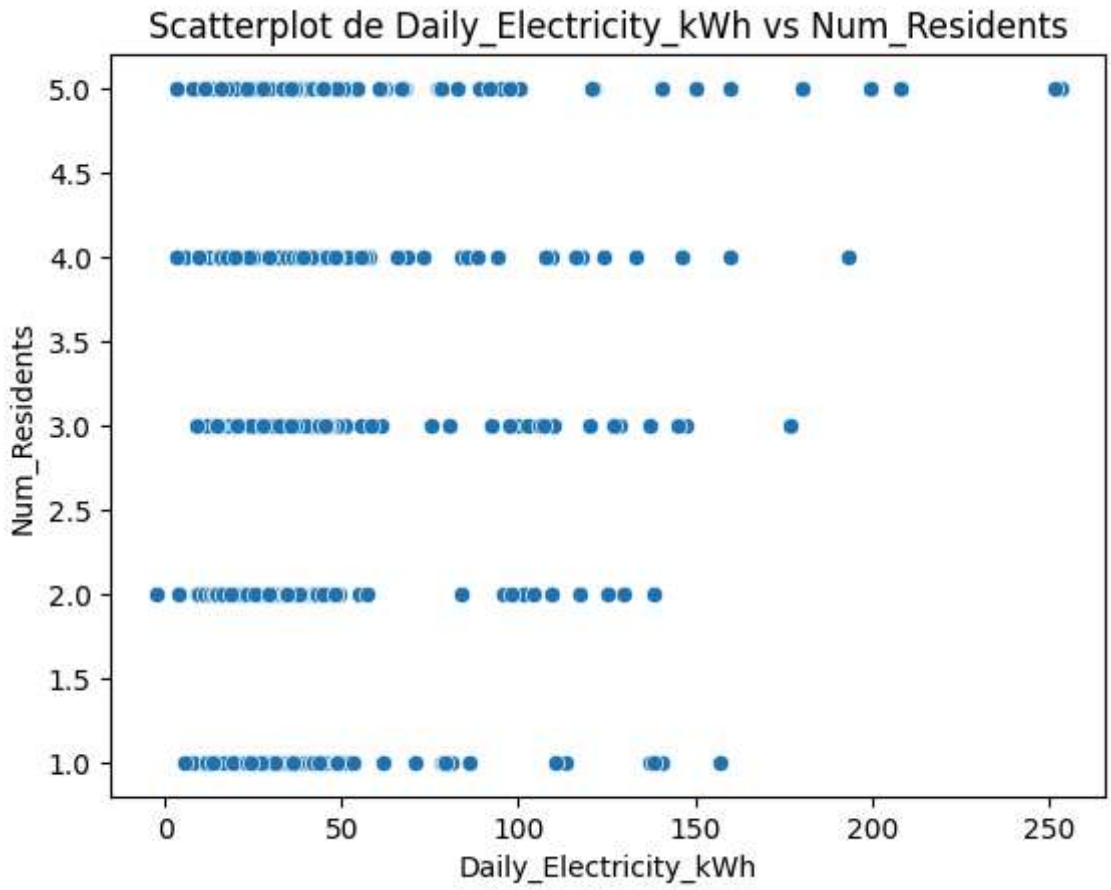


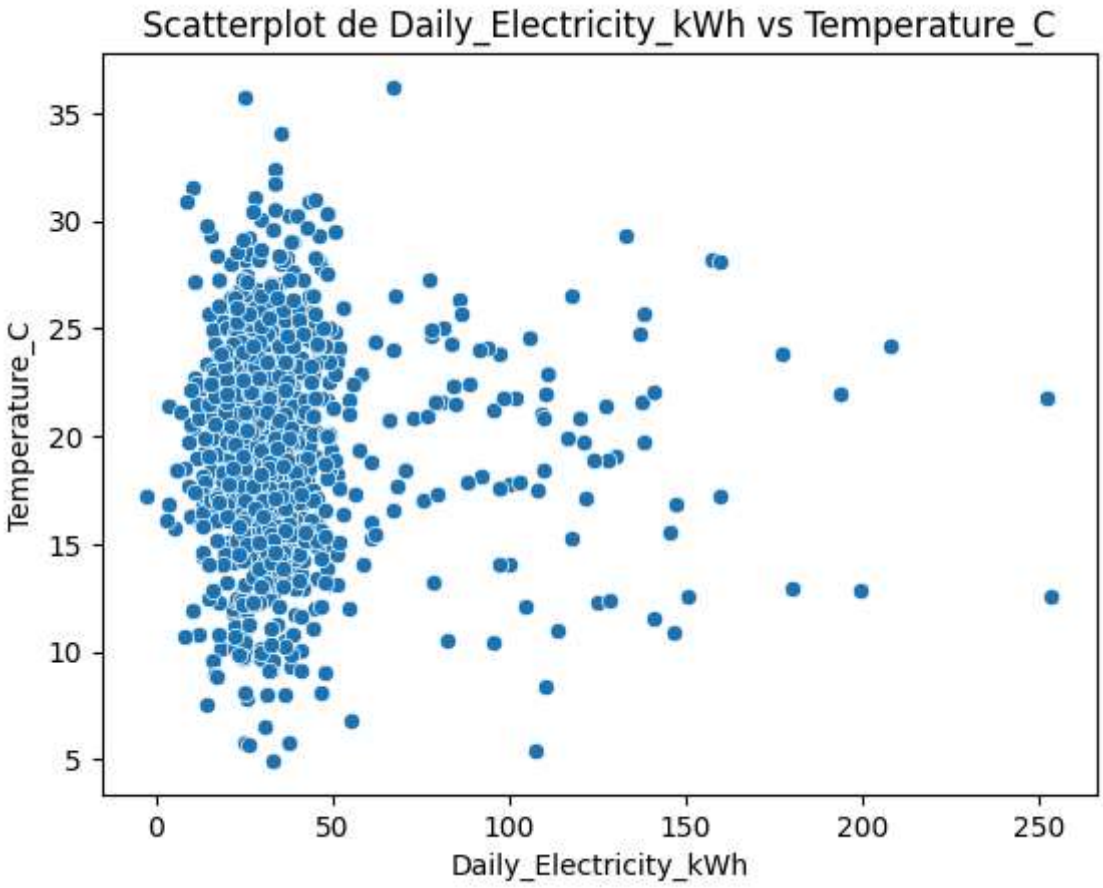
Multivariate plot: Scatter plot

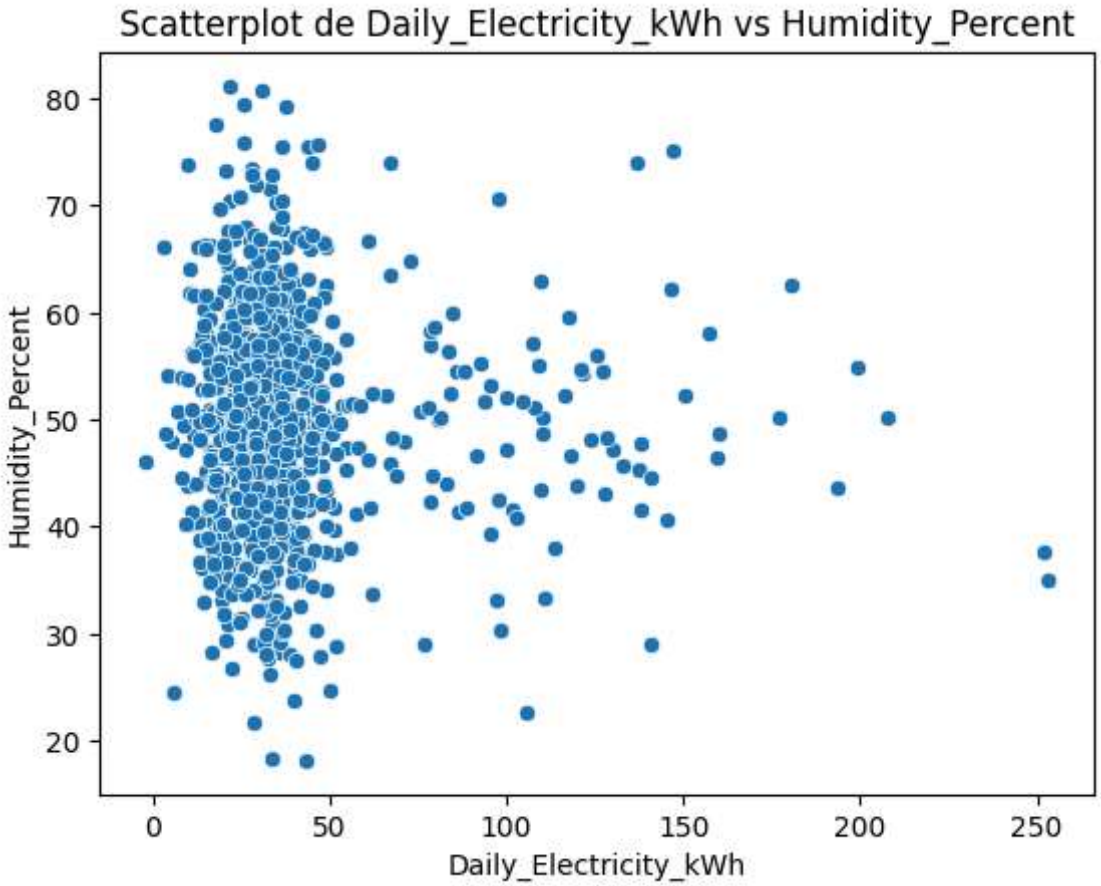
In [61]:
```
variables = dataset.columns

for i, var1 in enumerate(variables):
    for var2 in variables[i+1:]:
        sns.scatterplot(data=dataset, x=var1, y=var2)
        plt.title(f'Scatterplot de {var1} vs {var2}')
        plt.show()
```

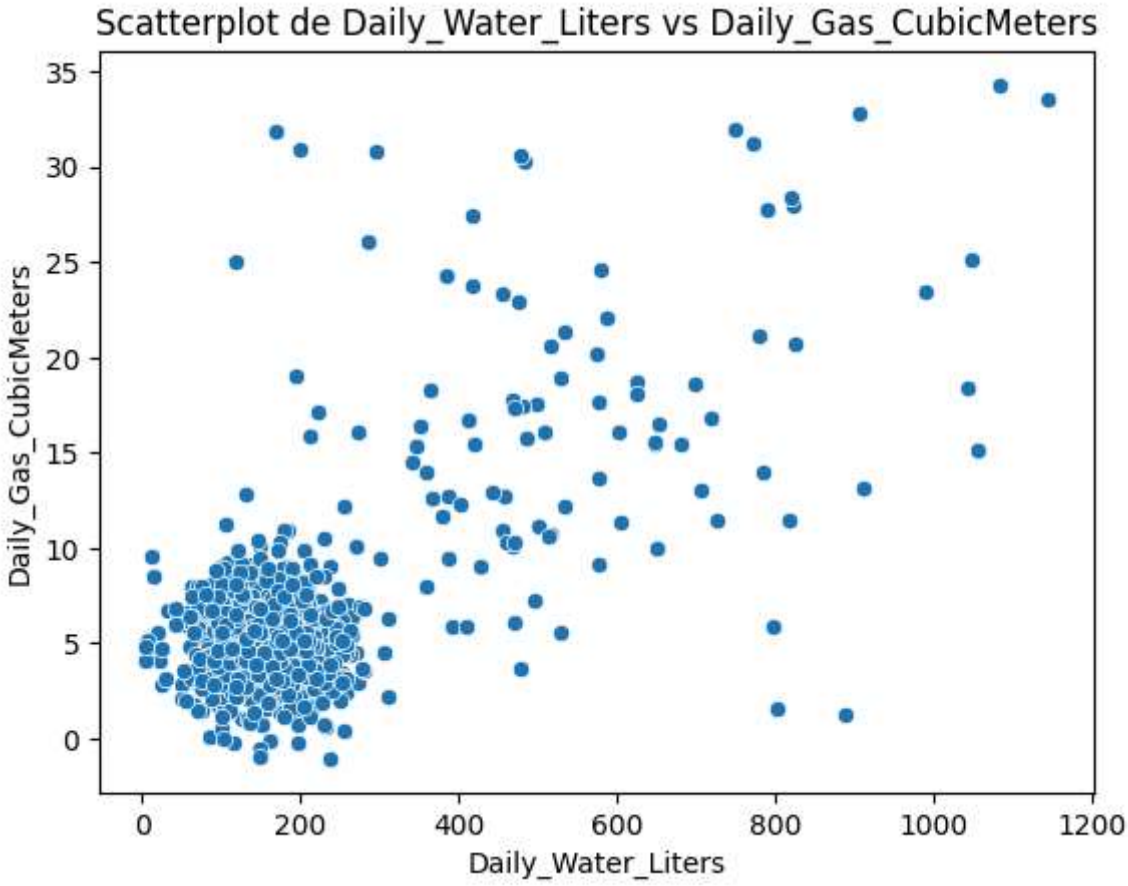

Scatterplot de Daily_Electricity_kWh vs Daily_Water_Liters
Daily_Water_Liters
Daily_Electricity_kWh
0
200
400
600
800
1000
1200
0
50
100
150
200
250

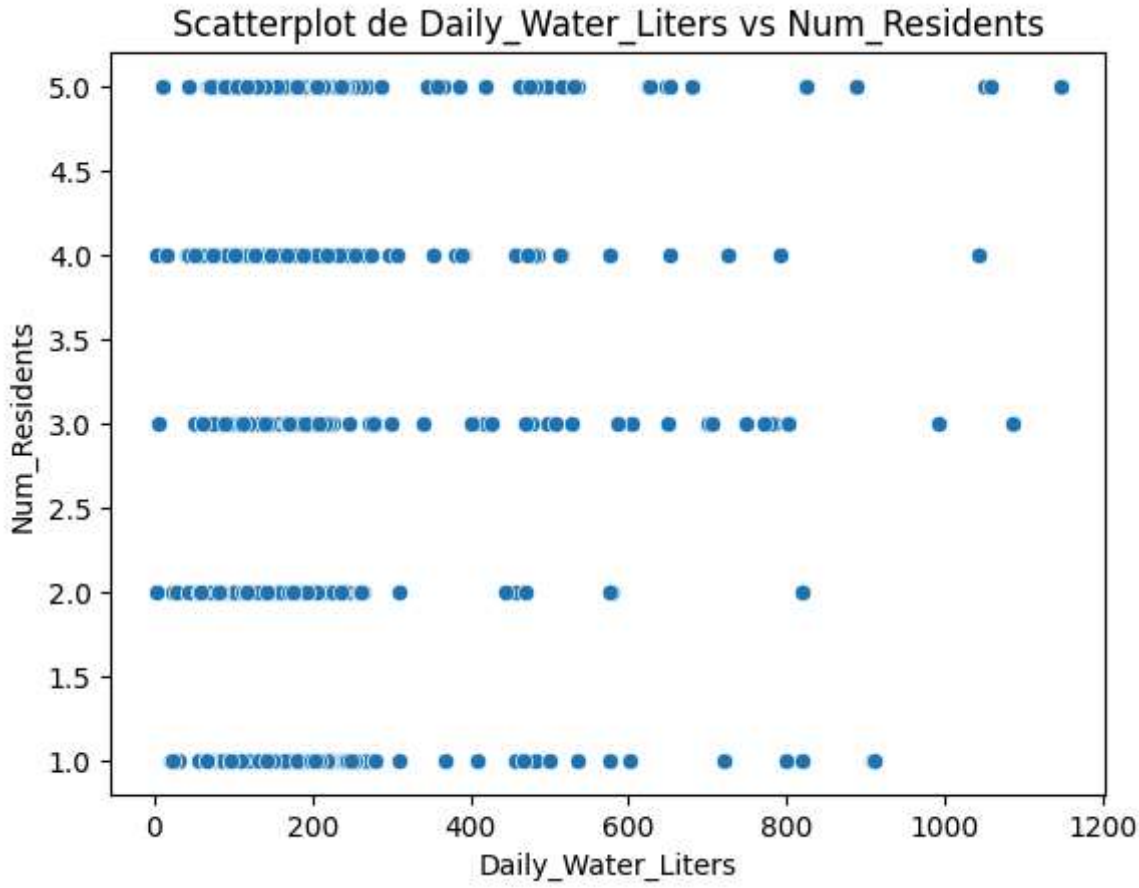

Scatterplot de Daily_Electricity_kWh vs Daily_Gas_CubicMeters
Daily_Gas_CubicMeters
Daily_Electricity_kWh
0
5
10
15
20
25
30
35
0
50
100
150
200
250

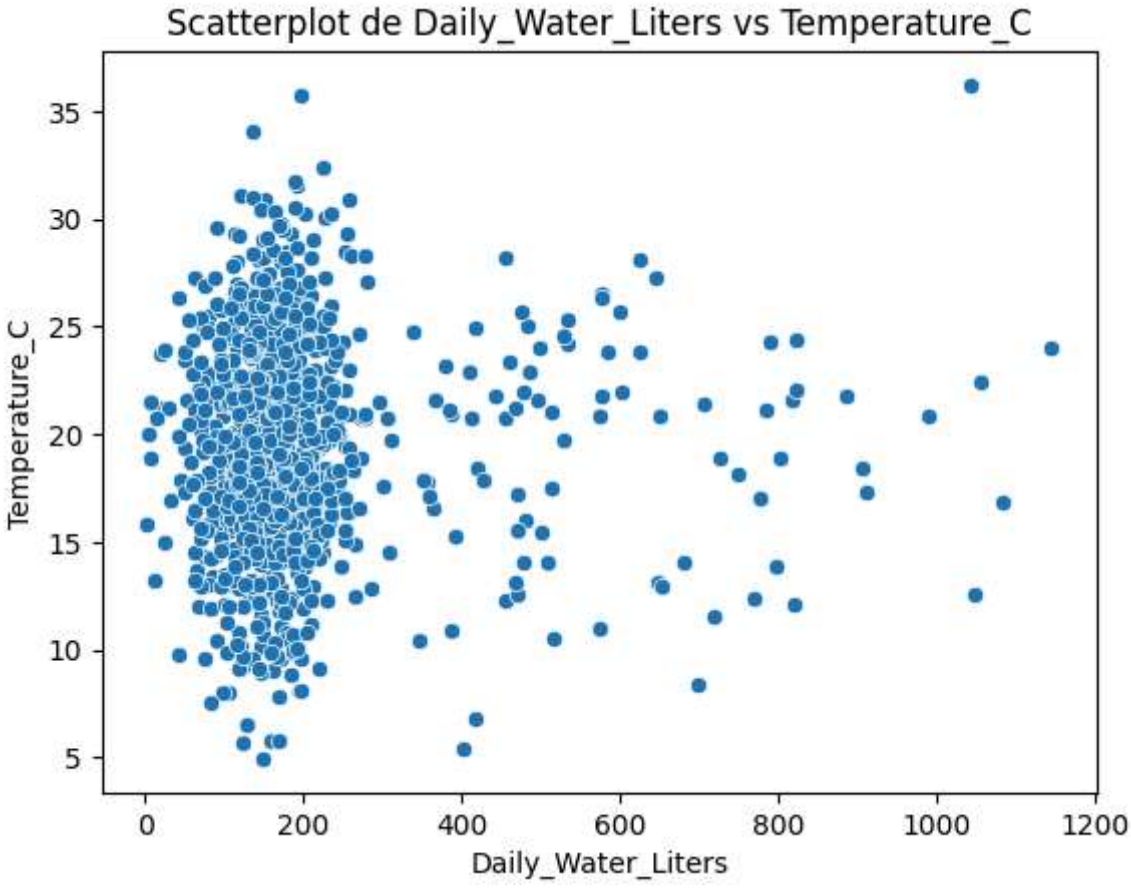

Scatterplot de Daily_Electricity_kWh vs Num_Residents
Num_Residents
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
50
100
150
200
250
Daily_Electricity_kWh

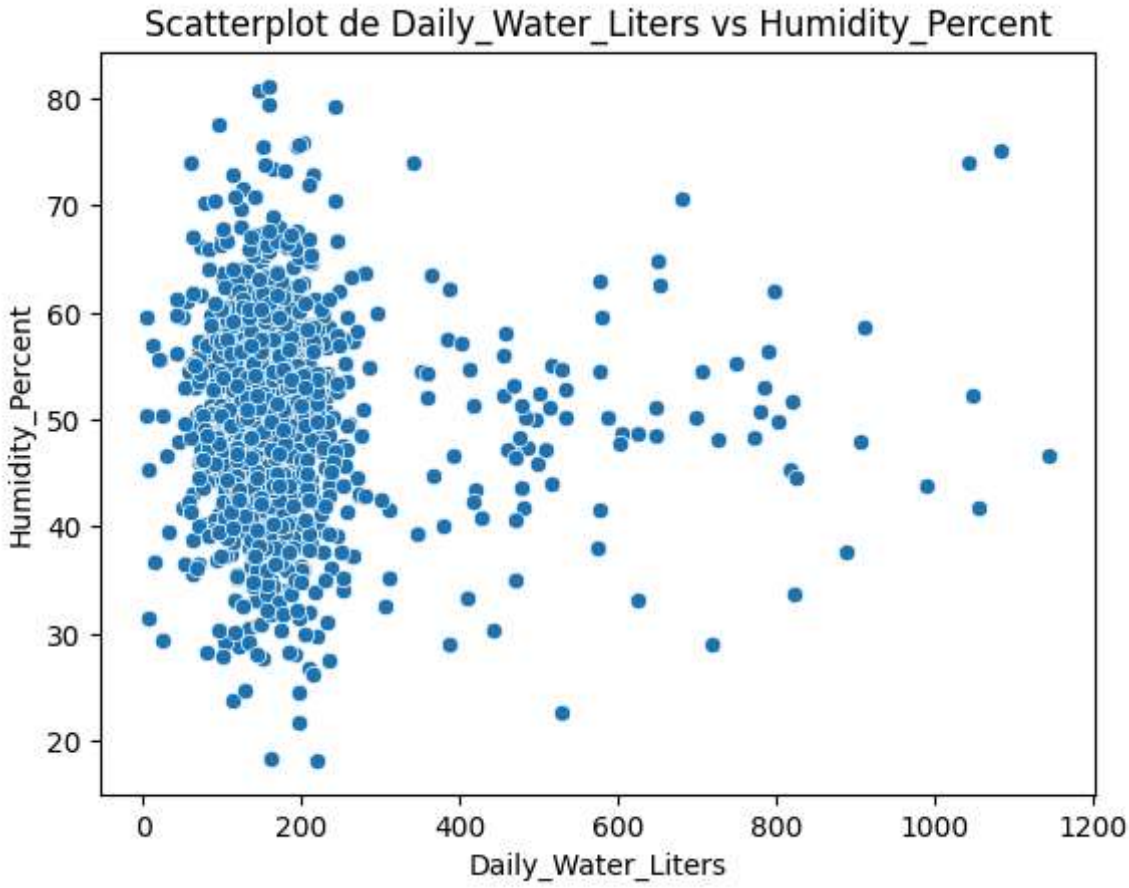

Scatterplot de Daily_Electricity_kWh vs Temperature_C
Temperature_C
35
30
25
20
15
10
5
0
50
100
150
200
250
Daily_Electricity_kWh

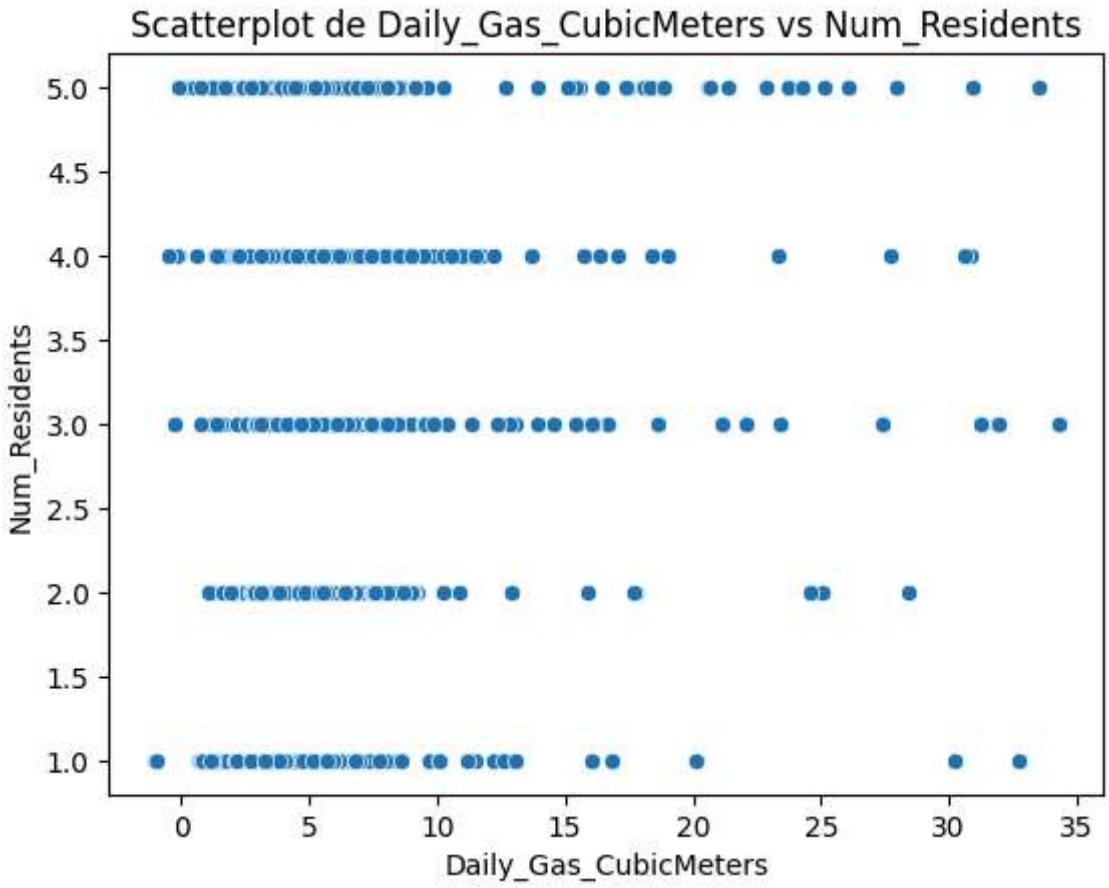

Scatterplot de Daily_Electricity_kWh vs Humidity_Percent
Humidity_Percent
Daily_Electricity_kWh
80
70
60
50
40
30
20
0
50
100
150
200
250

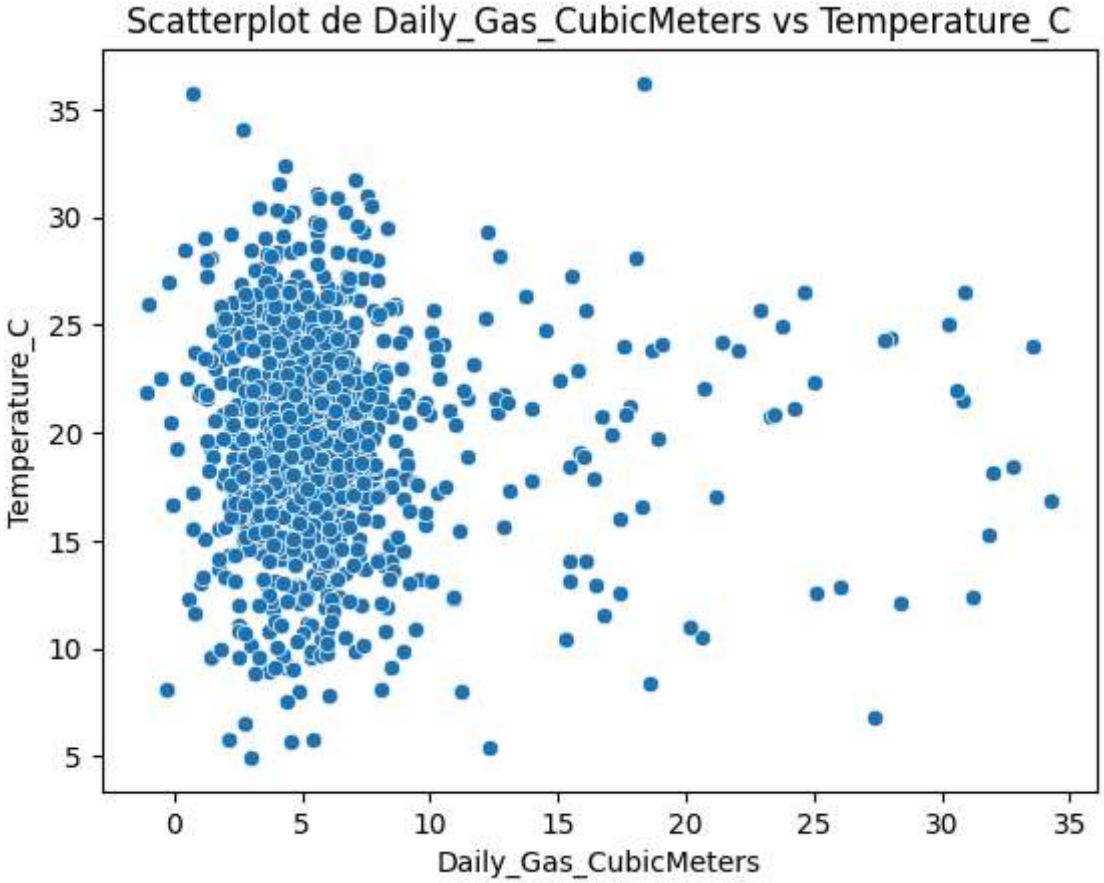

Scatterplot de Daily_Water_Liters vs Daily_Gas_CubicMeters
Daily_Gas_CubicMeters
Daily_Water_Liters
35
30
25
20
15
10
5
0
0
200
400
600
800
1000
1200

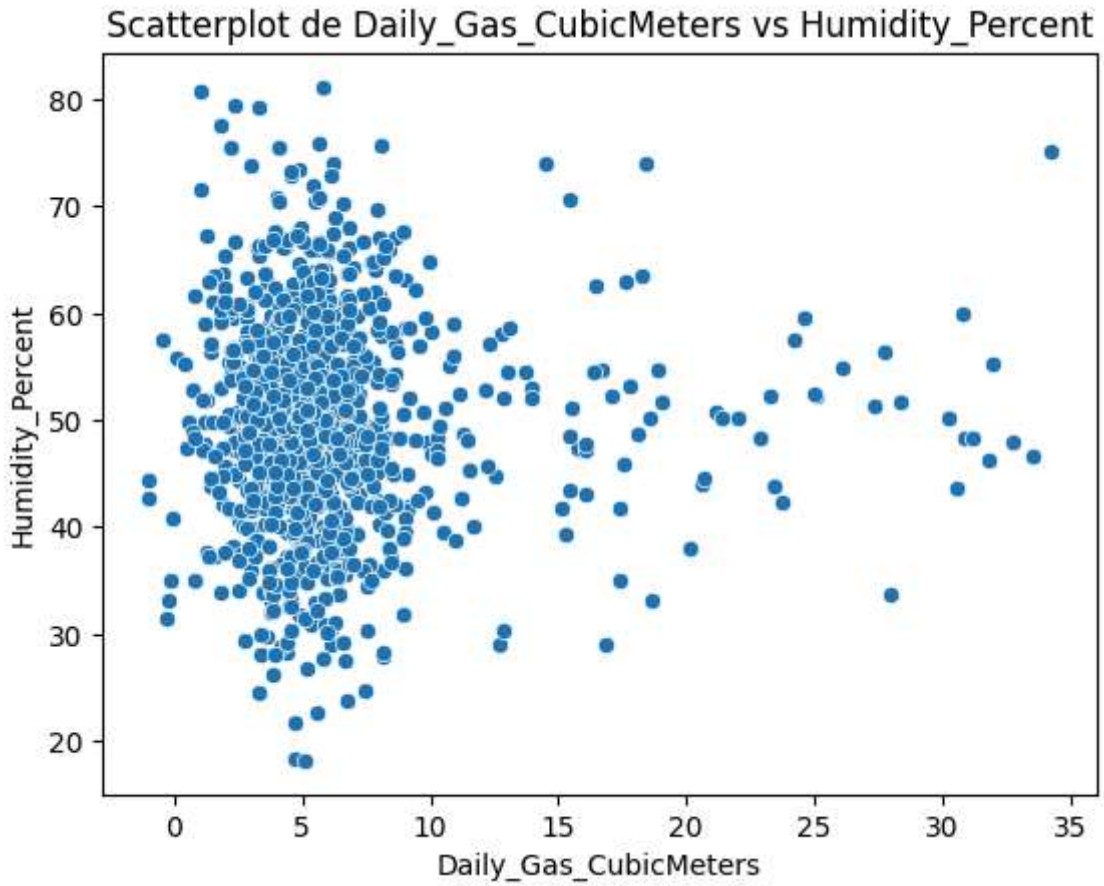

Scatterplot de Daily_Water_Liters vs Num_Residents
Num_Residents
Daily_Water_Liters
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
200
400
600
800
1000
1200

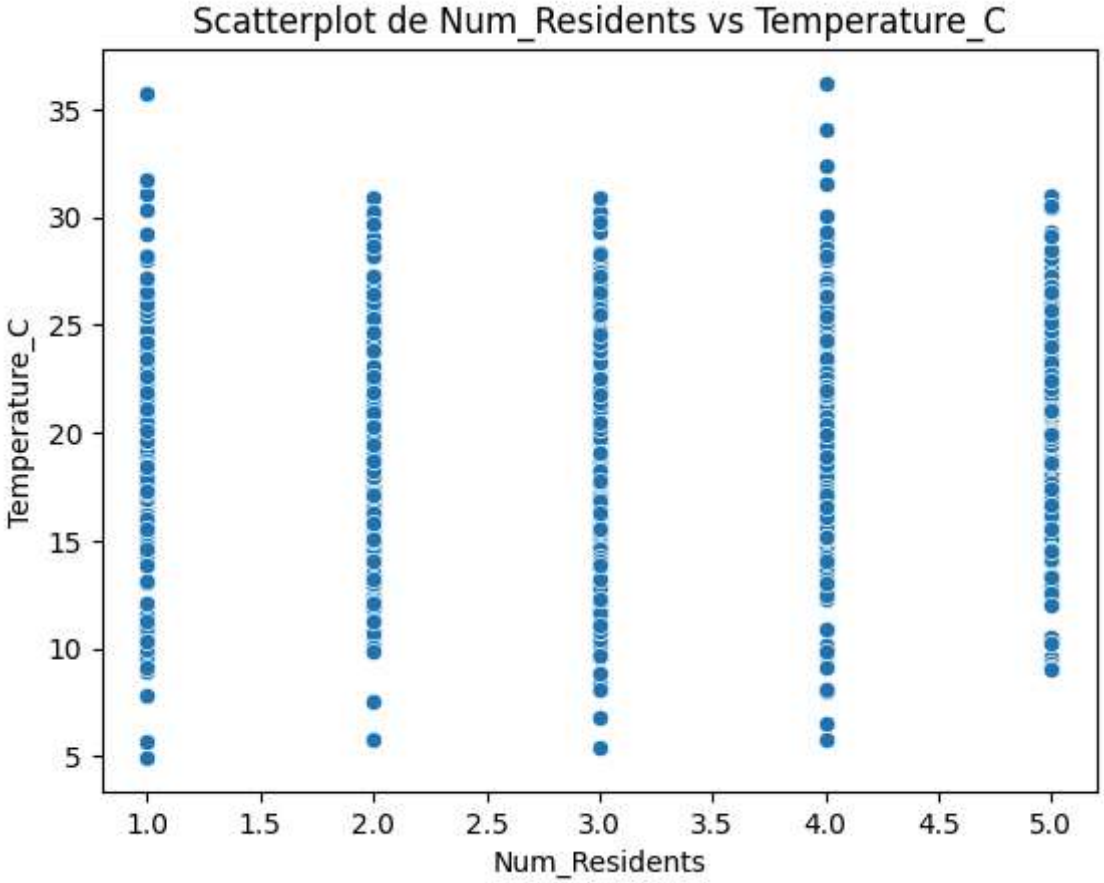

Scatterplot de Daily_Water_Liters vs Temperature_C
Temperature_C
Daily_Water_Liters
35
30
25
20
15
10
5
0
200
400
600
800
1000
1200

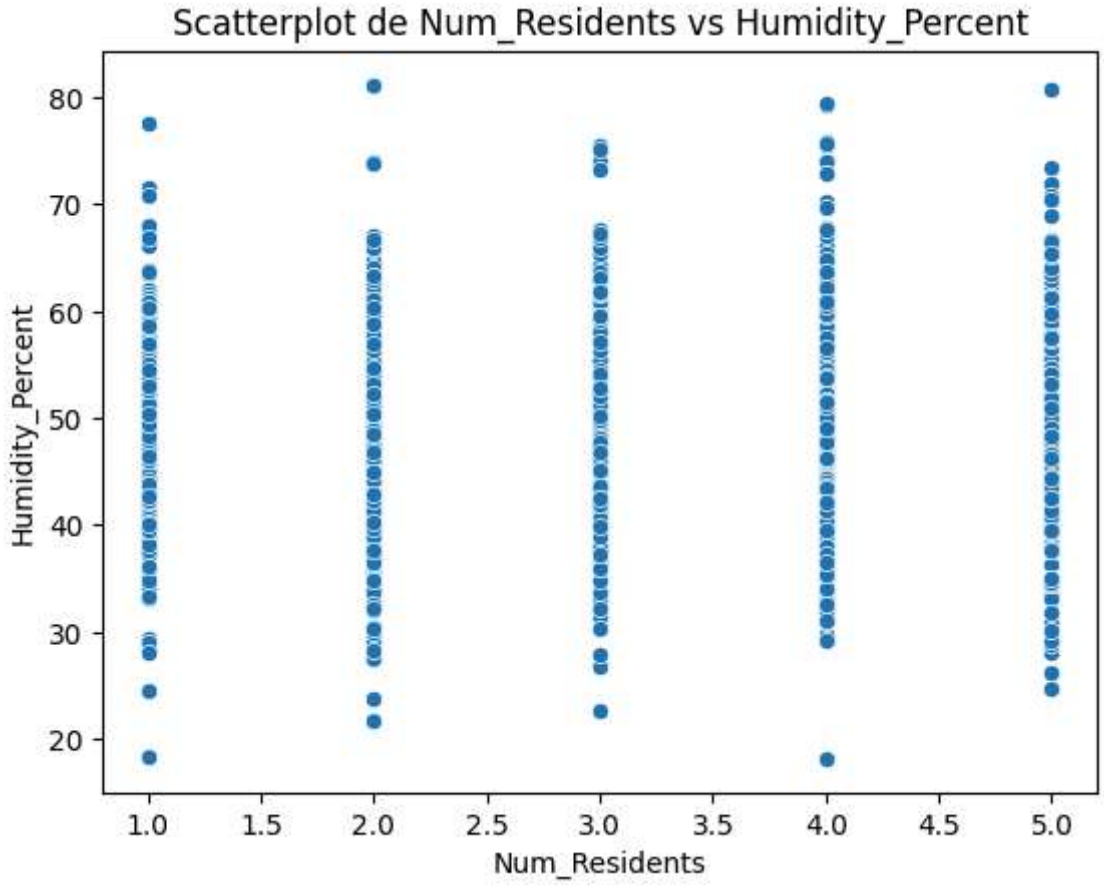

Scatterplot de Daily_Water_Liters vs Humidity_Percent
Humidity_Percent
80
70
60
50
40
30
20
0
200
400
600
800
1000
1200
Daily_Water_Liters

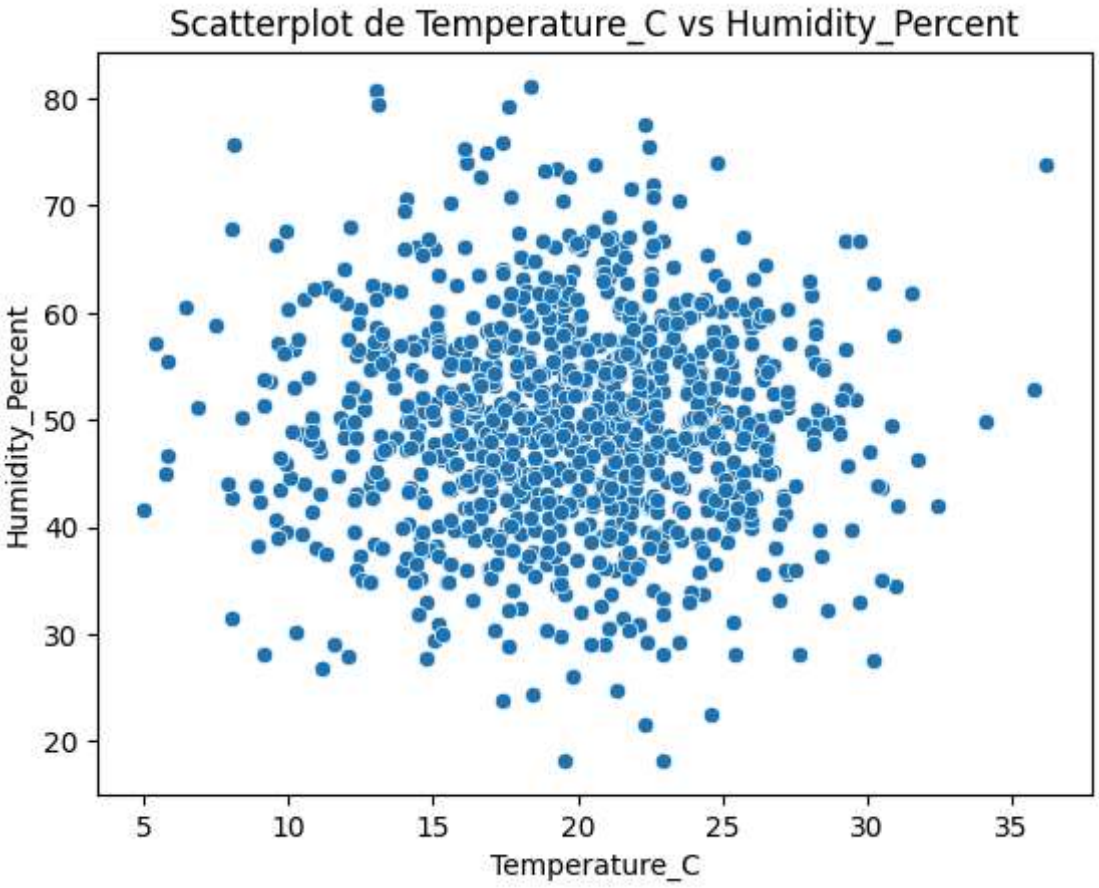

Scatterplot de Daily_Gas_CubicMeters vs Num_Residents
Num_Residents
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
5
10
15
20
25
30
35
Daily_Gas_CubicMeters


Scatterplot de Daily_Gas_CubicMeters vs Temperature_C
Temperature_C
Daily_Gas_CubicMeters
5
10
15
20
25
30
35
0
5
10
15
20
25
30
35


Scatterplot de Daily_Gas_CubicMeters vs Humidity_Percent
Humidity_Percent
Daily_Gas_CubicMeters
20
30
40
50
60
70
80
0
5
10
15
20
25
30
35

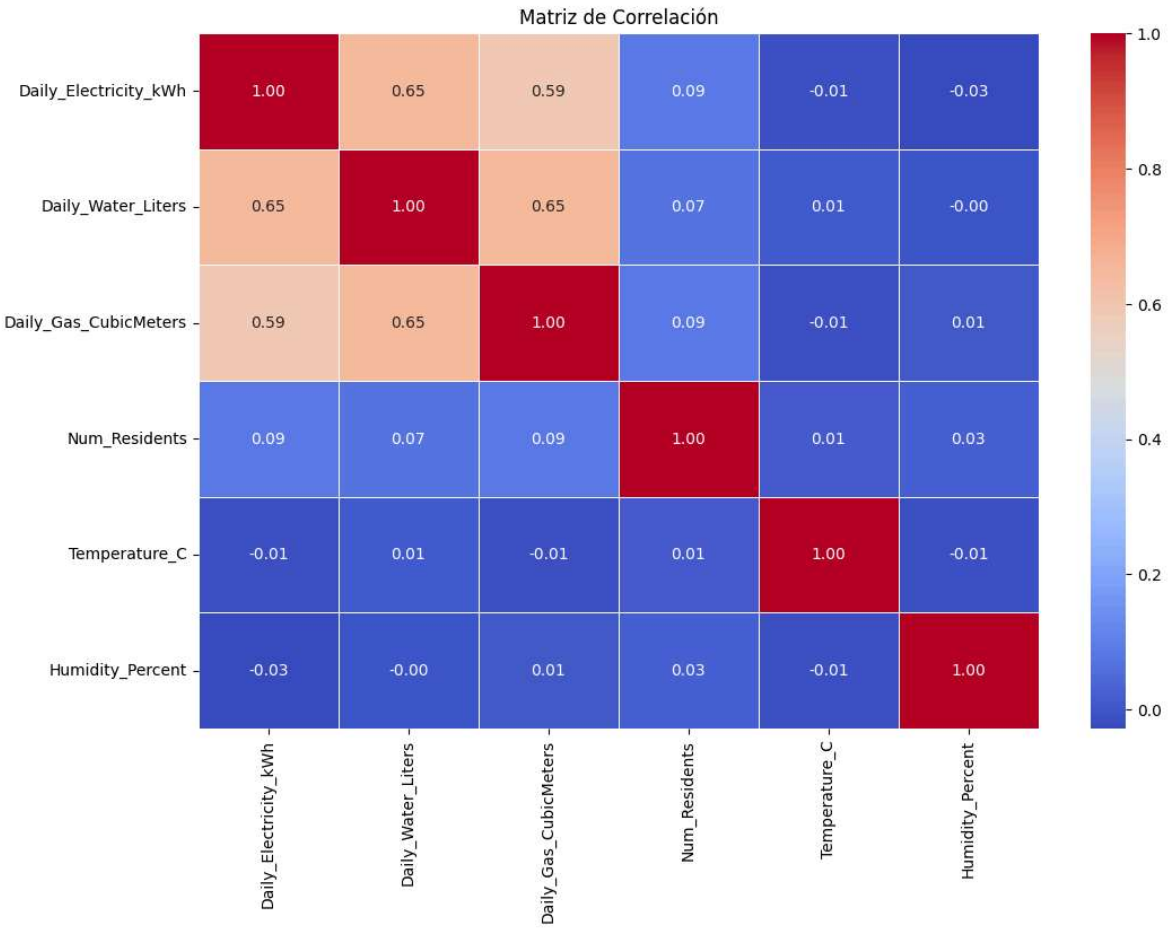

Scatterplot de Num_Residents vs Temperature_C
Temperature_C
35
30
25
20
15
10
5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Num_Residents


Scatterplot de Num_Residents vs Humidity_Percent
Humidity_Percent
80
70
60
50
40
30
20
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Num_Residents



Multivariate plot: Correlation matrix.

In [62]:
```
# calculate the correlation matrix
corr_matrix = dataset.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Matriz de Correlación')
plt.show()
```



3. **Prepare the data.**

- Take care of missing data (if necessary)
- Scale the data

Take care of missing data (if necessary)

In [63]:
```
dataset.isna().sum()
```

Out[63]:
```
Daily_Electricity_kWh    0
Daily_Water_Liters       0
Daily_Gas_CubicMeters    0
Num_Residents            0
Temperature_C            0
Humidity_Percent         0
dtype: int64
```

In [64]:
```
dataset.dropna(inplace=True)
```

Scale the data using StandardScaler()

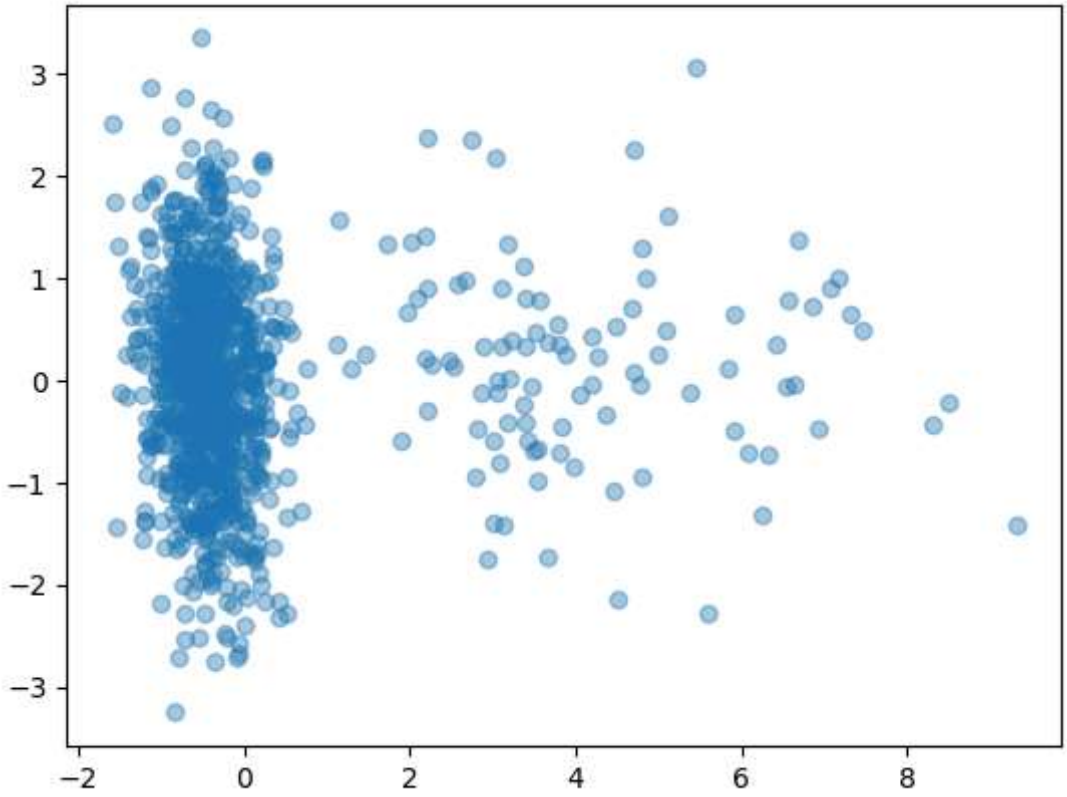
In [65]:
```
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
dataset_scaled = scaler.fit_transform(dataset)
```

Dimensionality reduction with PCA: Reduce the scaled dataset to 2 principal components and plot them on a 2D graph.

In [66]:
```
# PCA for dimensionality reduction
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
results_pca = pca.fit_transform(dataset_scaled)
print(results_pca)
```

```
[[-0.15448712  1.19625805]
 [-0.40846319  2.6529965 ]
 [-0.68526015  0.24141834]
 ...
 [ 3.0631933  -0.12151972]
 [-0.72676747 -0.87949027]
 [-0.53012821  0.88047627]]
```

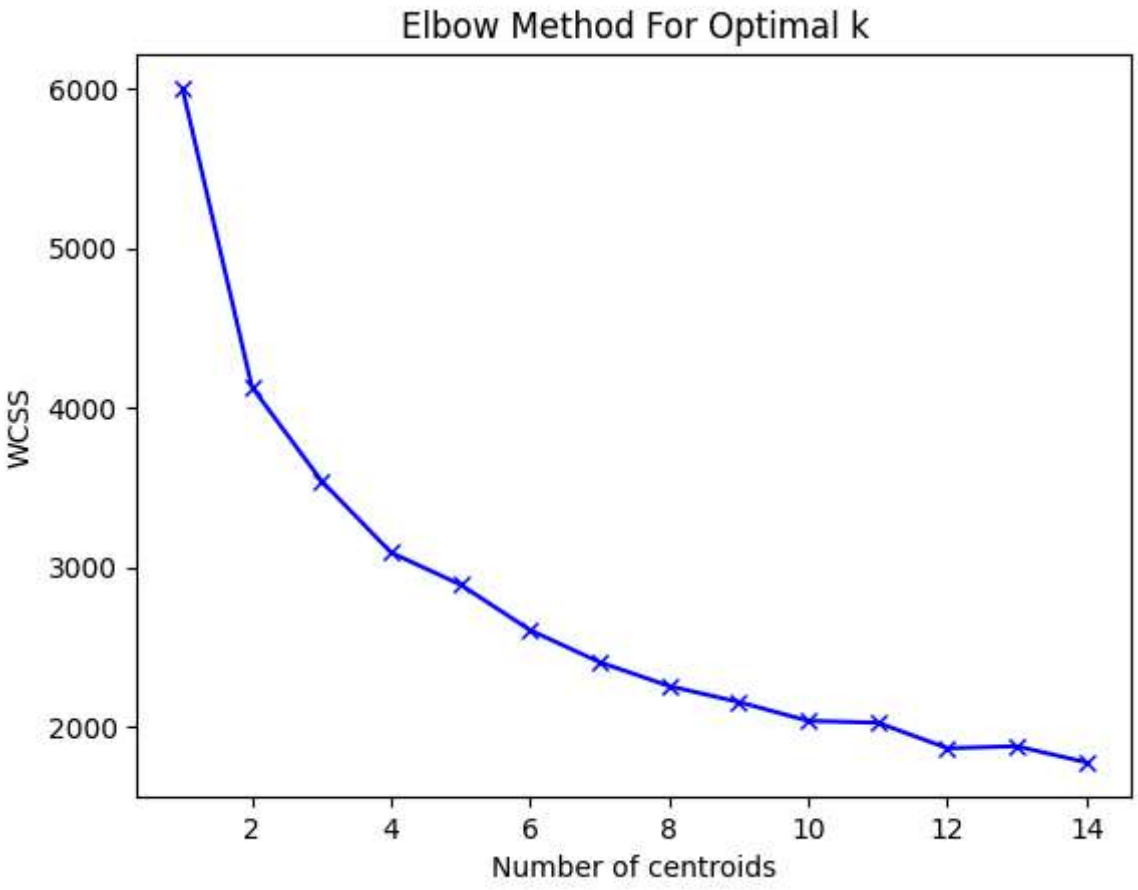
In [67]:
```
# Visualize the principal components in a plot
plt.scatter(results_pca[:, 0], results_pca[:, 1],alpha=0.4)
plt.show()
```



4. Unsupervised Learning Model Building: Houses Clustering using K-means

Find the optimal number of clusters using the Elbow method

In [68]:
```
# Elbow method
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

dataset_scaled_4 = dataset_scaled.copy()

wcss = []
K = range(1,15)

for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(dataset_scaled_4)
    wcss.append(km.inertia_)
plt.plot(K, wcss, 'bx-')
plt.xlabel('Number of centroids')
plt.ylabel('WCSS')
plt.title('Elbow Method For Optimal k')
plt.show()
```



Use K-means to find the cluster for each household

In [69]:
```
# Train the K-means for the optimal number of clusters given the result of the E
kmeans = KMeans(n_clusters=2, random_state=0).fit(dataset_scaled_4)
y_kmeans = kmeans.predict(dataset_scaled_4)
```

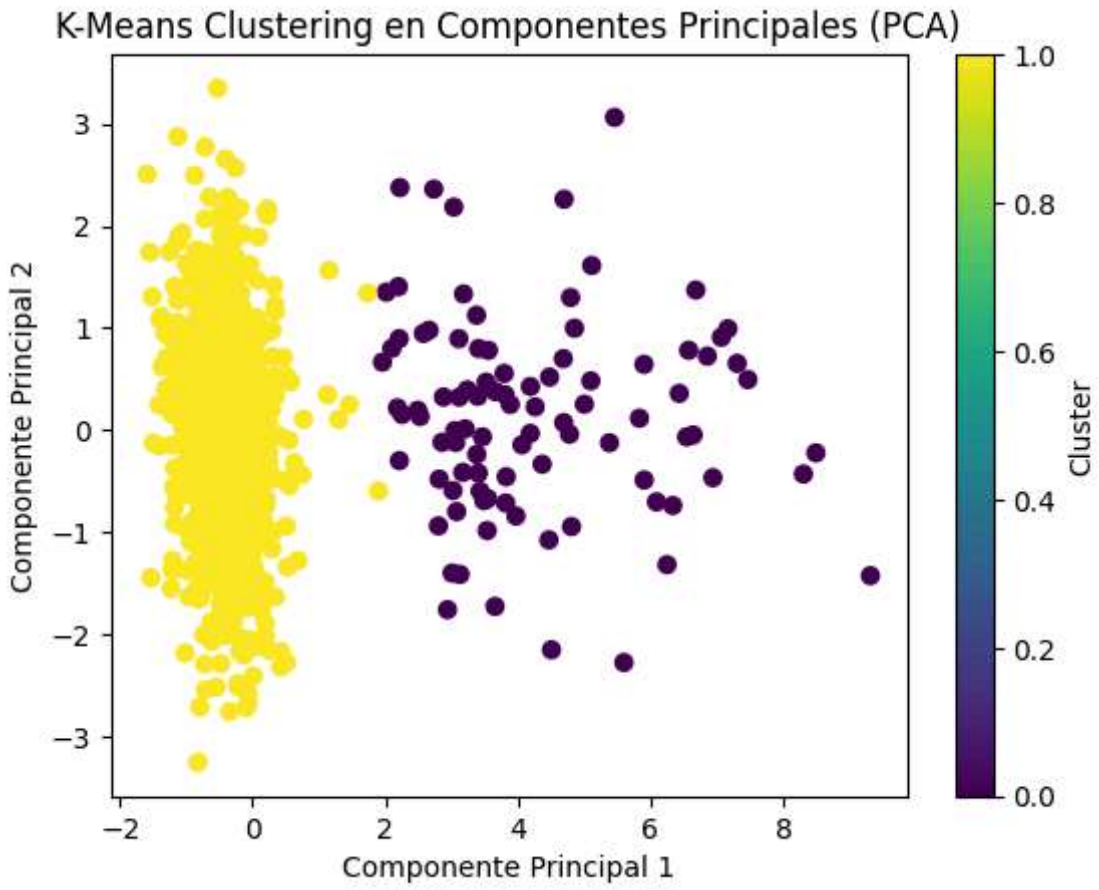
Print the clusters labels

In [70]:
```
print(y_kmeans)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1
 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1
 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1
 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1
 1]
```

Visualize the K-Means Clustering results obtained before on the PCA Components plot.

In [71]:
```
# Visualize K-Means Clustering on PCA Components (each data point should be colo

# Crear el scatter plot
plt.scatter(results_pca[:, 0], results_pca[:, 1], c=y_kmeans, cmap='viridis')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('K-Means Clustering en Componentes Principales (PCA)')
plt.colorbar(label='Cluster')
plt.show()
```



Add cluster labels back to the original dataset (create a new column named "cluster" in the original dataset)

In [72]:
```
# Add cluster labels to the original dataset (create a new column)
dataset['cluster'] = y_kmeans
dataset.head(3)
```

Out[72]:

Household_ID	Daily_Electricity_kWh	Daily_Water_Liters	Daily_Gas_CubicMeters	Num_Re
1	34.967142	219.967772	3.649643	
2	28.617357	196.231684	4.710963	
3	36.476885	152.981518	3.415160	

Group the data by cluster label (.groupby('cluster')) and compute the mean values (.mean()). and store that into a DataFrame that displays the mean values of each feature for clusters labeled as 0 and 1.

In [73]:
```
dataset_grupmean = dataset.copy()
dataset_grupmean = dataset_grupmean.groupby('cluster').mean()
```

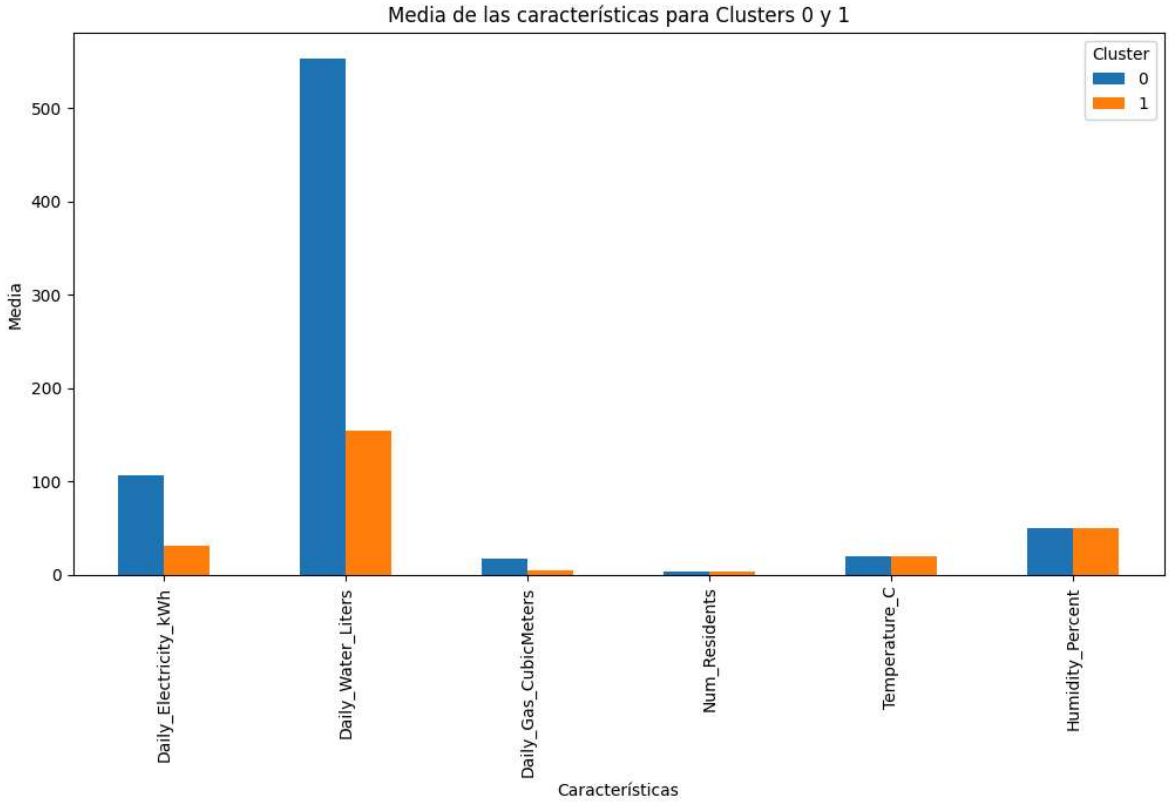
In [74]:
```
dataset_grupmean.head(3)
```

Out[74]:

cluster	Daily_Electricity_kWh	Daily_Water_Liters	Daily_Gas_CubicMeters	Num_Residents
0	105.794123	553.348417	17.219082	3.387097
1	30.409474	154.803228	5.069458	3.020948

Create a plot to visualize the average feature values for clusters 0 and 1 using bar plots and identify which features are most meaningful for detecting suspicious consumption.

In [75]:
```
# Identify potential anomalous households based on extreme feature averages

clusters_01 = dataset_grupmean.loc[[0, 1]]

clusters_01.T.plot(kind='bar', figsize=(12, 6))
plt.title('Media de las características para Clusters 0 y 1')
plt.xlabel('Características')
plt.ylabel('Media')
plt.legend(title='Cluster')
plt.show()
```



In [ ]:

In [ ]:

Final conclusion

Summarize the outcomes of the code developed, including a brief overview of the clustering process and the results obtained. Highlight any key findings or patterns observed after clustering the households, such as distinctive characteristics that might lead to suspicious consumption

Conclusions:

This report addresses a problem of identifying suspicious or atypical consumption patterns in households using unsupervised learning techniques.

First, the quality of the dataset was analysed as well as the study of individual variables. Then, clustering techniques such as PCA, K-means and the elbow method were used to determine the ideal number of clusters. Finally, the cluster results were plotted in order to extract conclusions in a more convenient way.

It can be seen that the means of number of residents, temperature and humidity are not too different between clusters. On the other hand, there is a large difference in the consumption of water, electricity and gas. This could indicate a malfunctioning of some device, of some sector of the system or even some user doing fraud.

I think it would be interesting to study the cases where this difference between cluster 0 and cluster 1 of the consumption for the three most relevant features is higher in order to study case by case. I also think it would be interesting to have more granularity of the data to be able to know if these atypical consumptions are made at atypical times or however it is continuously, this would help to know if it is a malfunction or leakage, a fraud or misuse of the facilities or a particular consumption pattern of certain users.

In [ ]:

In [ ]: