```
import pandas as pd
import numpy as py
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

df = pd.read_csv('house_hold.csv')
```

```
#water usage for one month
df1=df[df['Month']==1]
df1
```

	Year	Month	Household	Bathing(L)	Cooking(L)	Washing(L)	Gardening(L)	Dr
0	2023	1	Household_1	1274	432	1412.0	774.0	
1	2023	1	Household_2	1219	590	1844.0	1185.0	
2	2023	1	Household_3	1076	589	NaN	1023.0	
3	2023	1	Household_4	1486	465	1766.0	992.0	
4	2023	1	Household_5	1422	470	1721.0	615.0	
5	2023	1	Household_6	1249	447	1832.0	759.0	
6	2023	1	Household_7	1095	582	1377.0	1060.0	
7	2023	1	Household_8	1455	557	1960.0	890.0	
8	2023	1	Household_9	975	456	1480.0	NaN	
9	2023	1	Household_10	1378	574	1854.0	722.0	

Next steps: Generate code with df1 View recommended plots New interactive sheet

```
df1=df1.drop(columns=["Total_Usage(L)"])
df1
```

	Year	Month	Household	Bathing(L)	Cooking(L)	Washing(L)	Gardening(L)	Dr

0	2023	1	Household_1	1274	432	1412.0	774.0
1	2023	1	Household_2	1219	590	1844.0	1185.0
2	2023	1	Household_3	1076	589	NaN	1023.0
3	2023	1	Household_4	1486	465	1766.0	992.0
4	2023	1	Household_5	1422	470	1721.0	615.0
5	2023	1	Household_6	1249	447	1832.0	759.0
6	2023	1	Household_7	1095	582	1377.0	1060.0
7	2023	1	Household_8	1455	557	1960.0	890.0
8	2023	1	Household_9	975	456	1480.0	NaN
9	2023	1	Household_10	1378	574	1854.0	722.0

Next steps: Generate code with df1 View recommended plots New interactive sheet

```
#fill the null values
df1= df1.fillna(df1.mean(numeric_only=True))
df1
```

	Year	Month	Household	Bathing(L)	Cooking(L)	Washing(L)	Gardening(L)	Dr
0	2023	1	Household_1	1274	432	1412.0	774.000000	
1	2023	1	Household_2	1219	590	1844.0	1185.000000	
2	2023	1	Household_3	1076	589	1694.0	1023.000000	
3	2023	1	Household_4	1486	465	1766.0	992.000000	
4	2023	1	Household_5	1422	470	1721.0	615.000000	
5	2023	1	Household_6	1249	447	1832.0	759.000000	
6	2023	1	Household_7	1095	582	1377.0	1060.000000	
7	2023	1	Household_8	1455	557	1960.0	890.000000	
8	2023	1	Household_9	975	456	1480.0	891.111111	
9	2023	1	Household_10	1378	574	1854.0	722.000000	

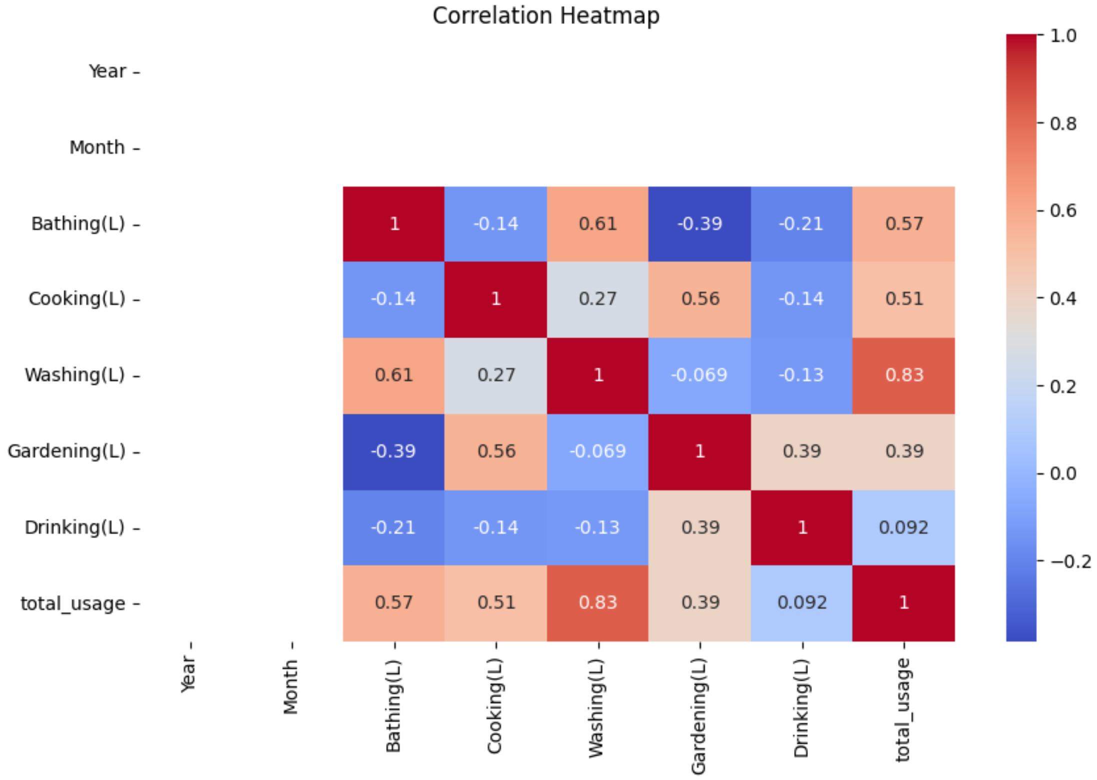
Next steps: Generate code with df1 View recommended plots New interactive sheet

```
df1["total_usage"] = df1["Bathing(L)"] + df1["Cooking(L)"] + df1["Washing(L)"] + df1["Gardening(L)"] + df1["Drinking(L)"]
df1
```

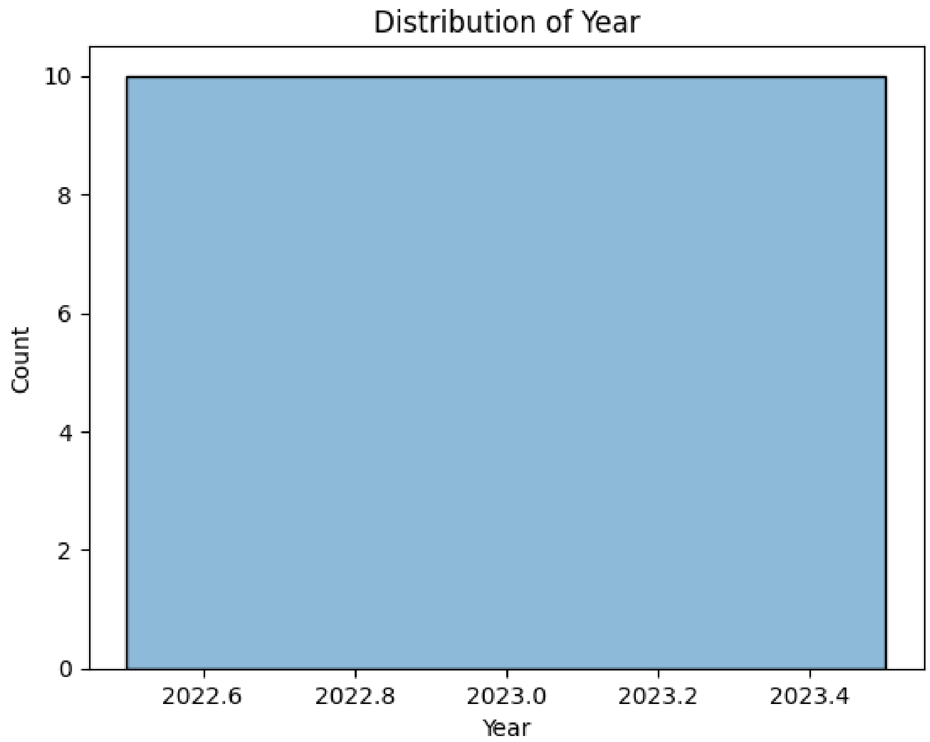
	Year	Month	Household	Bathing(L)	Cooking(L)	Washing(L)	Gardening(L)	Dr
0	2023	1	Household_1	1274	432	1412.0	774.000000	
1	2023	1	Household_2	1219	590	1844.0	1185.000000	
2	2023	1	Household_3	1076	589	1694.0	1023.000000	
3	2023	1	Household_4	1486	465	1766.0	992.000000	
4	2023	1	Household_5	1422	470	1721.0	615.000000	
5	2023	1	Household_6	1249	447	1832.0	759.000000	
6	2023	1	Household_7	1095	582	1377.0	1060.000000	
7	2023	1	Household_8	1455	557	1960.0	890.000000	
8	2023	1	Household_9	975	456	1480.0	891.111111	
9	2023	1	Household_10	1378	574	1854.0	722.000000	

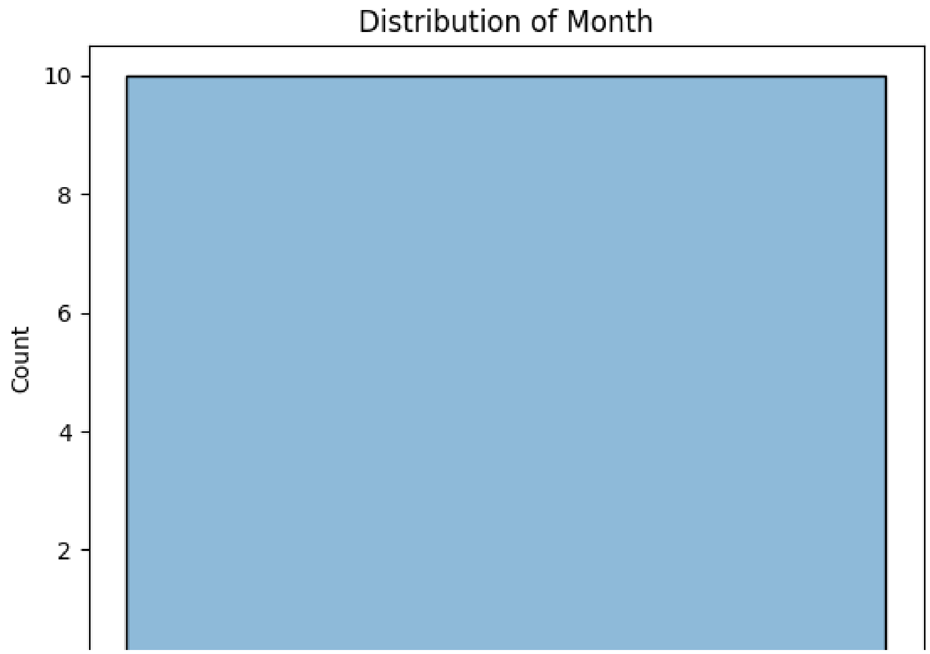
Next steps: Generate code with df1 View recommended plots New interactive sheet

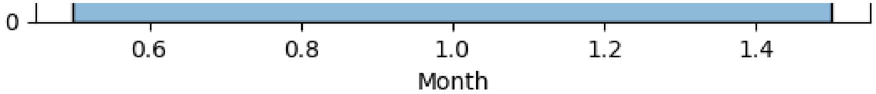
```
#Heatmap

plt.figure(figsize=(10,6))
sns.heatmap(df1.corr(numeric_only=True), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

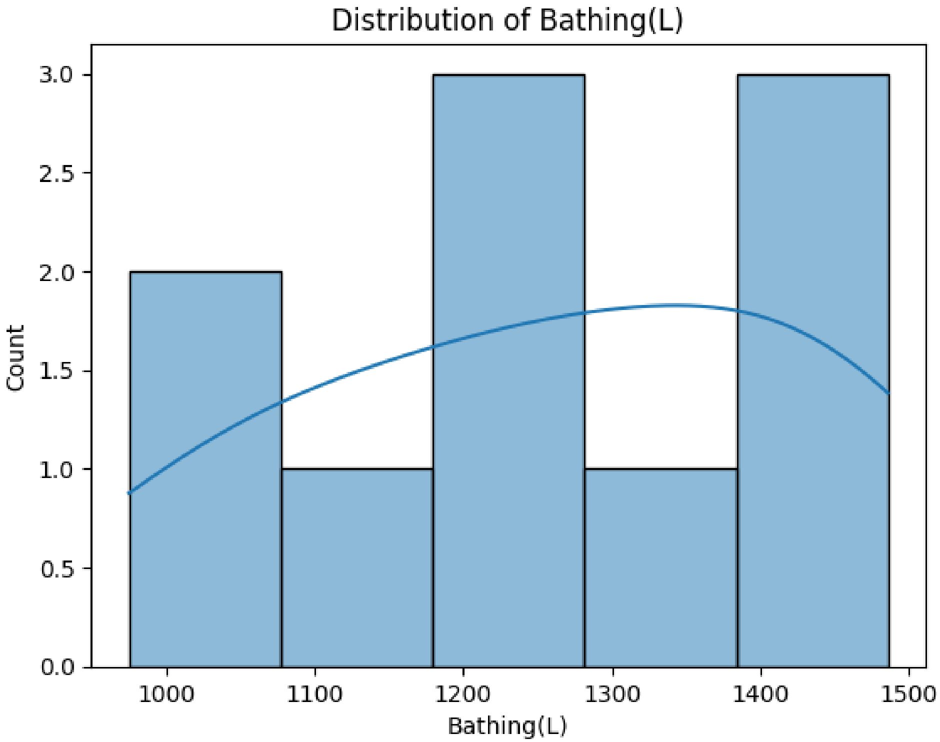


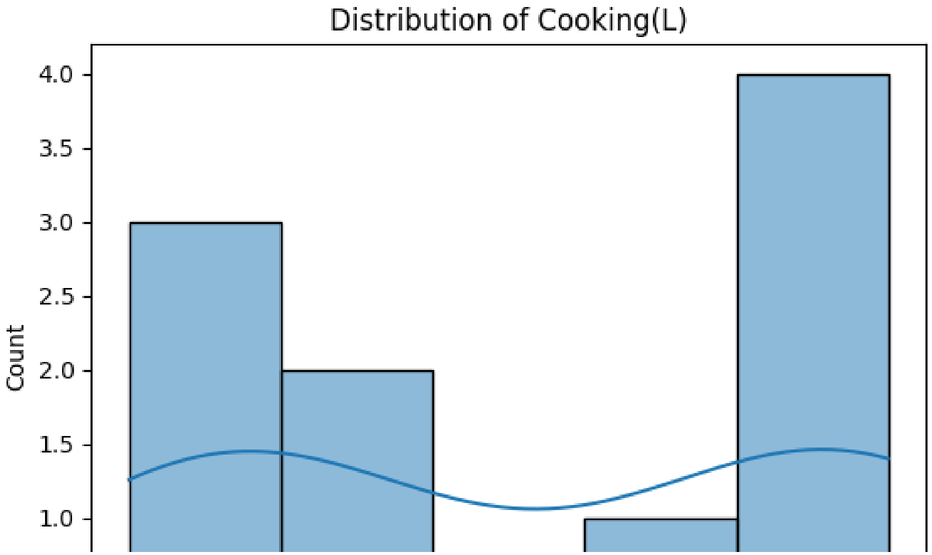

```
# Distribution plots for numeric columns
for col in df1.select_dtypes(include=[py.number]).columns:
        plt.figure()
        sns.histplot(df1[col].dropna(), kde=True)
        plt.title(f"Distribution of {col}")
        plt.show()
```

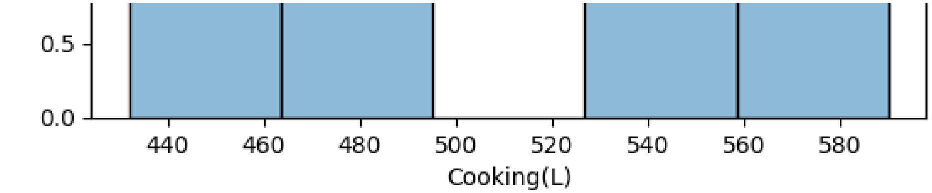

Distribution of Year
Count
10
8
6
4
2
0
2022.6
2022.8
2023.0
2023.2
2023.4
Year

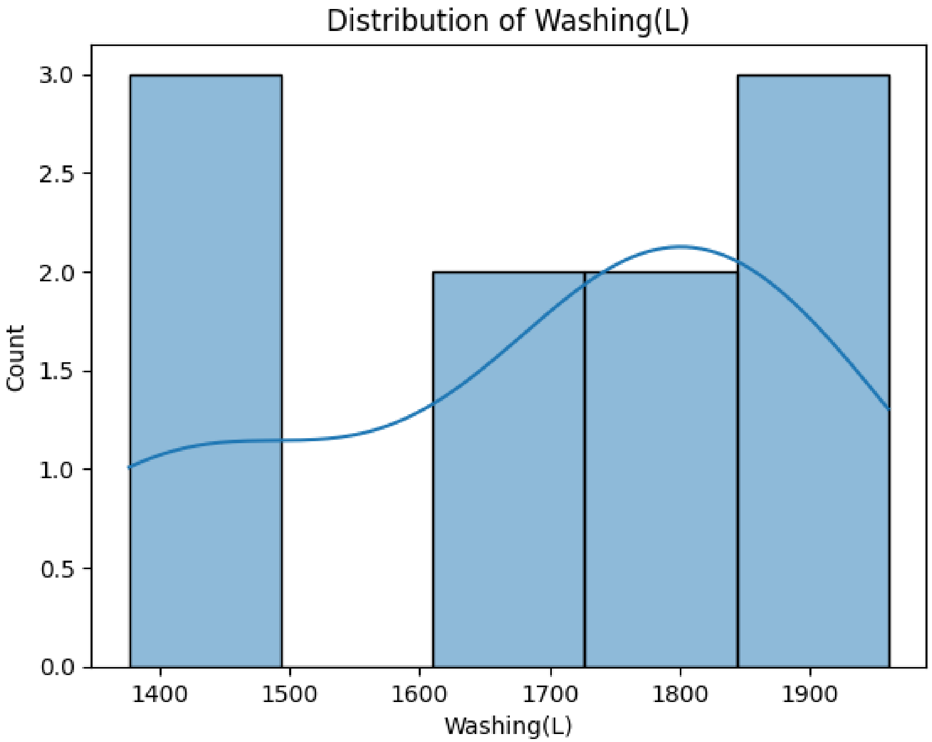

Distribution of Month
Count
10
8
6
4
2

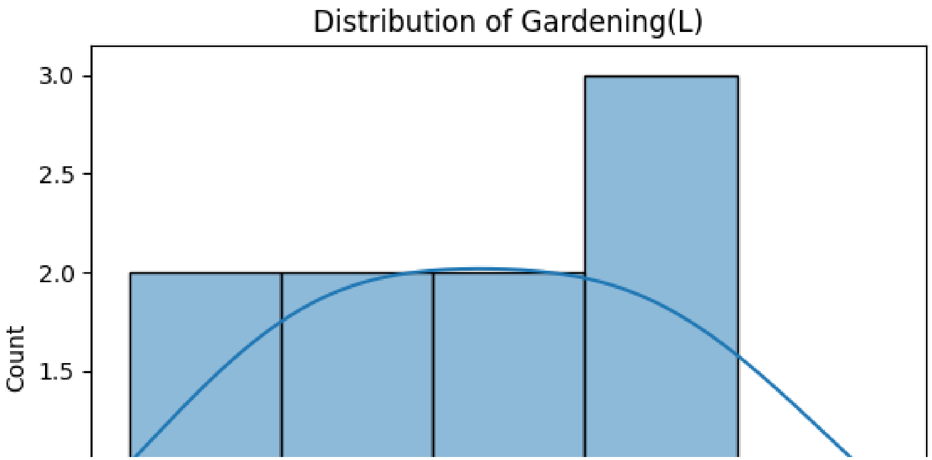

0
0.6
0.8
1.0
1.2
1.4
Month

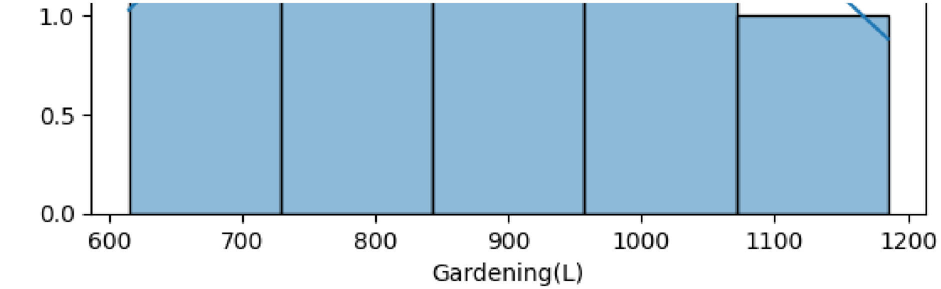

Distribution of Bathing(L)
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1000
1100
1200
1300
1400
1500
Bathing(L)

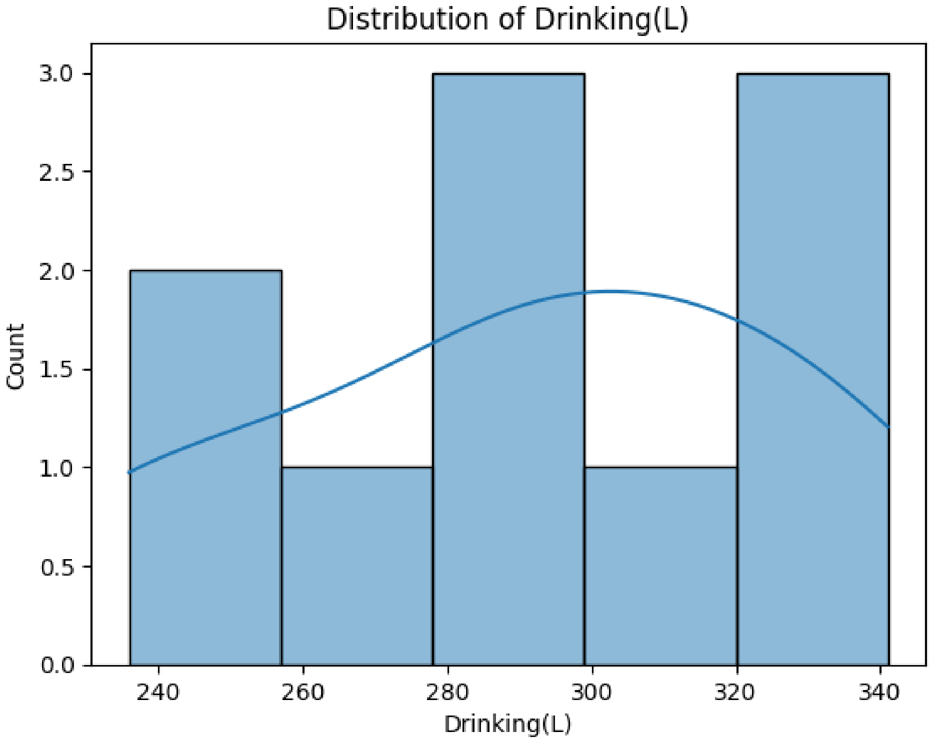

Distribution of Cooking(L)
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0

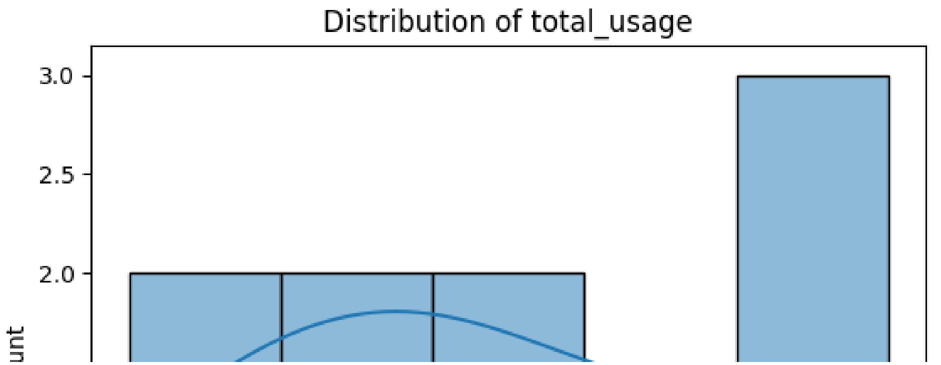

0.5
0.0
440
460
480
500
520
540
560
580
Cooking(L)

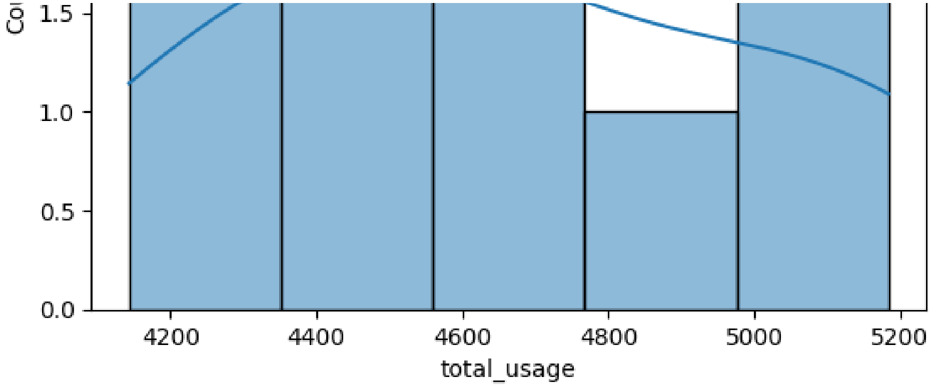

Distribution of Washing(L)
Count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1400
1500
1600
1700
1800
1900
Washing(L)


Distribution of Gardening(L)
Count
3.0
2.5
2.0
1.5


1.0
0.5
0.0
600
700
800
900
1000
1100
1200
Gardening(L)


Distribution of Drinking(L)
Count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
240
260
280
300
320
340
Drinking(L)


Distribution of total_usage
3.0
2.5
2.0

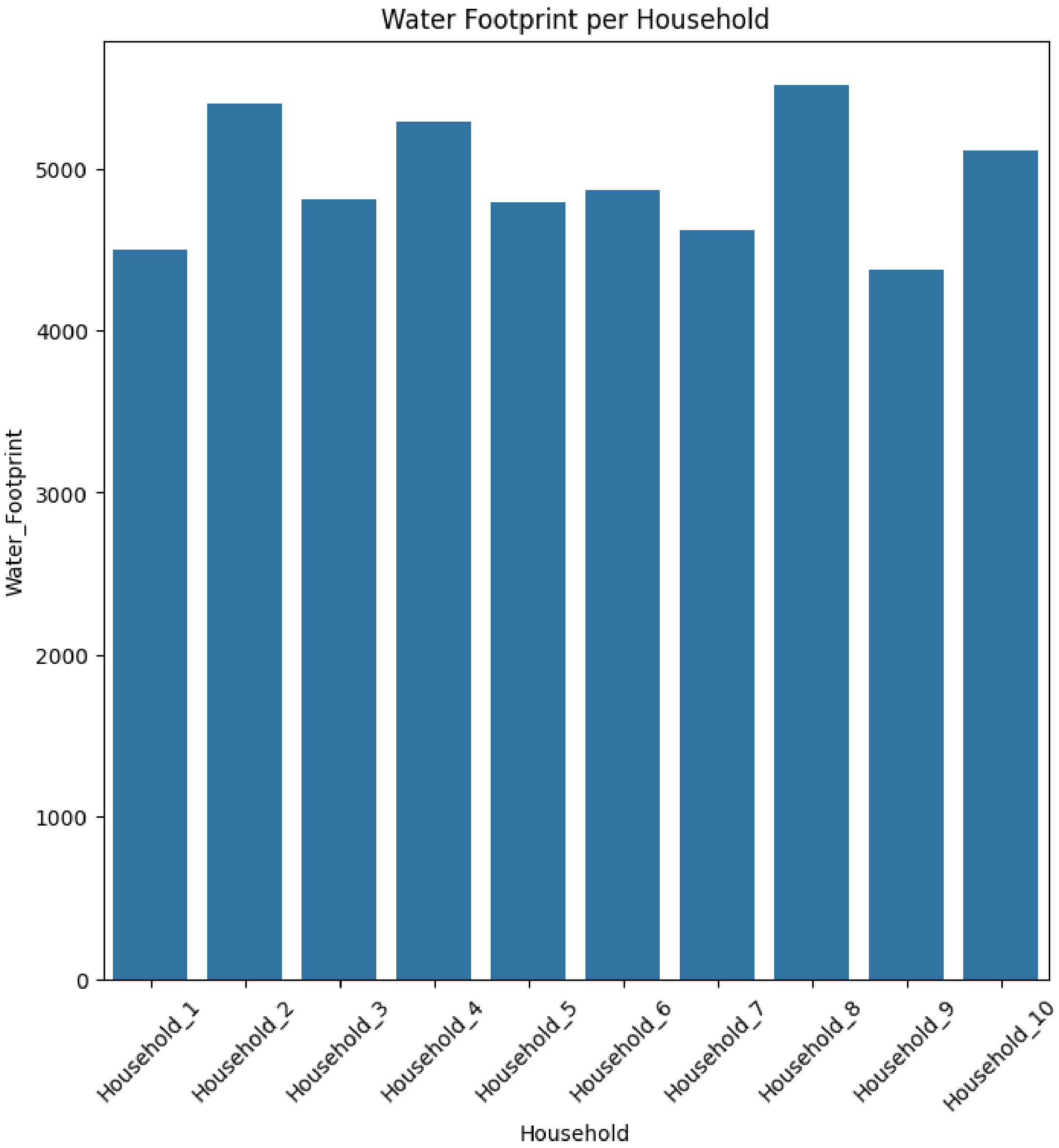



```
# Assign weights to activities to calculate water_footprints
weights = {
    "Bathing(L)": 1.2,
    "Cooking(L)": 1.1,
    "Washing(L)": 1.0,
    "Gardening(L)": 0.8,
    "Drinking(L)": 1.5
}

df1["Water_Footprint"] = (
    df1["Bathing(L)"] * weights["Bathing(L)"] +
    df1["Cooking(L)"] * weights["Cooking(L)"] +
    df1["Washing(L)"] * weights["Washing(L)"] +
    df1["Gardening(L)"] * weights["Gardening(L)"] +
    df1["Drinking(L)"] * weights["Drinking(L)"]
)
df1
```

	Year	Month	Household	Bathing(L)	Cooking(L)	Washing(L)	Gardening(L)	Dr
0	2023	1	Household_1	1274	432	1412.0	774.000000	
1	2023	1	Household_2	1219	590	1844.0	1185.000000	
2	2023	1	Household_3	1076	589	1694.0	1023.000000	
3	2023	1	Household_4	1486	465	1766.0	992.000000	
4	2023	1	Household_5	1422	470	1721.0	615.000000	
5	2023	1	Household_6	1249	447	1832.0	759.000000	
6	2023	1	Household_7	1095	582	1377.0	1060.000000	
7	2023	1	Household_8	1455	557	1960.0	890.000000	
8	2023	1	Household_9	975	456	1480.0	891.111111	
9	2023	1	Household_10	1378	574	1854.0	722.000000	

Next steps: Generate code with df1 View recommended plots New interactive sheet

```
#Bar graph for Water_footprints
plt.figure(figsize=(8,8))
sns.barplot(x="Household", y="Water_Footprint", data=df1)
plt.xticks(rotation=45)
plt.title("Water Footprint per Household")
plt.show()
```




```
# Encode categorical variables
```

```
label_encoders = {}
for col in df1.select_dtypes(include=['object']).columns:
          le = LabelEncoder()
          df1[col] = le.fit_transform(df1[col].astype(str))
          label_encoders[col] = le
print("Categorical columns encoded successfully")
print(df1.head())
```

```
Categorical columns encoded successfully
   Year  Month  Household  Bathing(L)  Cooking(L)  Washing(L)  Gardening(L)  \
0  2023      1          0        1274         432      1412.0         774.0
1  2023      1          2        1219         590      1844.0        1185.0
2  2023      1          3        1076         589      1694.0        1023.0
3  2023      1          4        1486         465      1766.0         992.0
4  2023      1          5        1422         470      1721.0         615.0

   Drinking(L)  total_usage  Water_Footprint
0       311.00      4203.00         4501.700
1       331.00      5169.00         5400.300
2       238.00      4620.00         4808.500
3       291.25      5000.25         5291.175
4       236.00      4464.00         4790.400
```

```
# Scale numeric features
scaler = StandardScaler()
numeric_cols = df1.select_dtypes(include=[py.number]).columns
df1[numeric_cols] = scaler.fit_transform(df1[numeric_cols])
print("Numeric columns scaled successfully")
print(df1.head())
```

```
Numeric columns scaled successfully
   Year  Month  Household  Bathing(L)  Cooking(L)  Washing(L)  Gardening(L)  \
0   0.0    0.0  -1.566699    0.067120   -1.325703   -1.466798     -0.701789
1   0.0    0.0  -0.870388   -0.265457    1.161958    0.780212      1.761131
2   0.0    0.0  -0.522233   -1.130159    1.146214    0.000000      0.790345
3   0.0    0.0  -0.174078    1.349055   -0.806128    0.374502      0.604577
4   0.0    0.0   0.174078    0.962056   -0.727405    0.140438     -1.654598

   Drinking(L)  total_usage  Water_Footprint
0     0.569197    -1.277213        -1.156140
1     1.145600     1.449624         1.286256
2    -1.534671    -0.100100        -0.322257
3     0.000000     0.973274         0.989654
4    -1.592312    -0.540459        -0.371453
```

```
#Training ML Model

activity_cols = ['Bathing(L)','Cooking(L)','Washing(L)','Gardening(L)','Drinking(L)']
X=df1[activity_cols]
y=df1["Water_Footprint"]
```

```
scalar=StandardScaler()
X_scaled=scalar.fit_transform(X)
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

```
# Random Forest
rf = RandomForestRegressor(random_state=42, n_estimators=100)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
# Model Evaluation

print("Linear Regression Performance:")
print("MSE:", mean_squared_error(y_test, y_pred_lr))
print("R²:", r2_score(y_test, y_pred_lr))

print("Random Forest Performance:")
print("MSE:", mean_squared_error(y_test, y_pred_rf))
print("R²:", r2_score(y_test, y_pred_rf))
```

```
Linear Regression Performance:
MSE: 1.232595164407831e-30
R²: 1.0
Random Forest Performance:
MSE: 0.8753722236896062
R²: 0.548258816747192
```

What can I help you build?