In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import pandas as pd


df = pd.read_csv('Clean-Water-and-Sanitation-Dataset-Sheet7.csv')


print(df)
```

```
                                                 Time    Unnamed: 1  \
0             Pacific Island Countries and territories  Urbanization
1    Indicator: Goal 06: Water and Sanitation > 6.1...           NaN
2                                           Melanesia           NaN
3                                             ·  Fiji      National
4                                             ·  Fiji         Urban
..                                                ...           ...
163                                           ·  Samoa      National
164                                           ·  Tonga      National
165                                          ·  Tuvalu      National
166                                               NaN           NaN
167                 [object Object] Terms & conditions           NaN

     Unnamed: 2   2016   2017   2018   2019   2020   2021   2022  \
0           NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN
1           NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN
2           NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN
3           NaN  95.28  95.31  95.35  95.38  95.42  95.45  95.49
4           NaN  98.58  98.60  98.61  98.62  98.64  98.65  98.66
..          ...    ...    ...    ...    ...    ...    ...    ...
163         NaN   0.00   6.60   0.00   0.00  46.70   0.00   0.00
164         NaN   0.00   0.00   0.00   0.00  28.60   0.00   0.00
165         NaN   0.00   0.00   0.00   0.00   2.00   0.00   0.00
166         NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN
167         NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN

                                        Unnamed: 10
0                                               NaN
1                                               NaN
2                                               NaN
3                                               NaN
4                                               NaN
..                                              ...
163                                             NaN
164                                             NaN
165                                             NaN
166                                             NaN
167  Sustainable Development Goal 06 - Clean Water ...

[168 rows x 11 columns]
```

In [3]:
```
df = pd.read_csv('Clean1 (1).csv')
df
```

Out[3]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: Goal 06: Water and Sanitation > 6.1...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Melanesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Fiji	National	95.28	95.31	95.35	95.38	95.42	95.45	95.49
4	Fiji	Urban	98.58	98.60	98.61	98.62	98.64	98.65	98.66
5	Fiji	Rural	91.20	91.18	91.15	91.13	91.11	91.08	91.06
6	New Caledonia	National	95.73	95.96	96.18	96.41	96.64	96.87	96.87
7	Papua New Guinea	National	43.99	45.02	46.05	47.08	48.11	49.17	50.24
8	Papua New Guinea	Urban	85.69	86.02	86.34	86.66	86.98	86.96	86.94
9	Papua New Guinea	Rural	37.73	38.84	39.94	41.03	42.12	43.29	44.47
10	Solomon Islands	National	68.54	67.78	67.01	67.15	67.30	67.45	NaN
11	Solomon Islands	Urban	91.34	91.41	91.41	91.41	91.41	91.41	NaN
12	Solomon Islands	Rural	61.80	60.60	59.41	59.41	59.41	59.41	59.41
13	Vanuatu	National	88.80	89.22	89.64	90.07	90.49	90.91	91.34
14	Vanuatu	Urban	98.33	98.47	98.62	98.76	98.91	99.05	99.20
15	Vanuatu	Rural	85.61	86.11	86.61	87.11	87.60	88.10	88.60
16	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	Kiribati	National	71.99	72.59	73.18	73.75	74.31	74.86	75.69
18	Kiribati	Urban	85.84	86.11	86.38	86.64	86.90	87.14	87.86
19	Kiribati	Rural	56.72	57.18	57.64	58.10	58.55	59.01	59.53
20	Marshall Islands	National	86.40	86.17	85.94	85.72	85.49	85.28	85.06
21	Marshall Islands	Urban	85.11	85.00	84.90	84.79	84.68	84.58	84.47
22	Marshall Islands	Rural	90.55	90.00	89.44	88.89	88.34	87.78	87.23
23	Micronesia (Federated States of)	National	90.11	90.11	90.11	90.11	90.11	NaN	NaN
24	Nauru	National	97.00	97.00	97.00	97.00	100.00	NaN	NaN
25	Nauru	Urban	97.00	97.00	97.00	97.00	100.00	NaN	NaN
26	Northern Mariana Islands	National	89.85	90.24	90.64	90.64	90.64	90.64	90.64
27	Palau	National	99.53	99.54	99.54	99.55	99.56	99.56	99.57
28	Palau	Urban	99.62	99.62	99.62	99.62	99.62	99.62	99.61
29	Palau	Rural	99.18	99.21	99.24	99.27	99.30	99.34	99.37
30	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	American Samoa	National	89.62	89.85	89.85	89.85	89.85	89.85	NaN
32	Cook Islands	National	99.96	99.97	99.97	99.97	99.97	99.97	99.97
33	French Polynesia	National	86.92	86.07	85.22	84.37	83.52	82.66	81.81
34	Niue	National	97.46	97.31	97.16	97.01	97.01	97.01	97.01
35	Samoa	National	93.29	94.35	95.41	96.42	97.40	98.27	99.04
36	Samoa	Urban	93.33	95.02	96.73	98.15	99.36	100.00	100.00
37	Samoa	Rural	93.28	94.20	95.12	96.04	96.97	97.90	98.83
38	Tonga	National	98.59	98.63	98.66	98.70	98.74	98.77	98.81
39	Tonga	Urban	99.06	99.14	99.23	99.32	99.41	99.49	99.58
40	Tonga	Rural	98.45	98.47	98.49	98.51	98.54	98.56	98.58
41	Tuvalu	National	99.14	99.16	99.18	99.20	99.22	99.24	99.26
42	Tuvalu	Urban	98.88	98.90	98.92	98.94	98.96	98.98	99.00
43	Tuvalu	Rural	99.54	99.57	99.61	99.64	99.67	99.71	99.74
44	Wallis and Futuna	National	68.95	68.93	68.92	68.91	68.89	68.88	68.88

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
45	Wallis and Futuna	Rural	68.95	68.93	68.92	68.91	68.89	68.88	68.88

In [4]:
```
df = pd.read_csv('Clean2.csv')
df
```

Out[4]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.1.1 Population using an improved ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Kiribati	National	0.0	0.0	82.3	0.0	0.0	0.0	0.0
4	Kiribati	Urban	0.0	0.0	96.5	0.0	0.0	0.0	0.0
5	Kiribati	Rural	0.0	0.0	65.8	0.0	0.0	0.0	0.0
6	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	Cook Islands	National	99.9	0.0	0.0	0.0	0.0	0.0	0.0
8	Cook Islands	Urban	100.0	0.0	0.0	0.0	0.0	0.0	0.0
9	Cook Islands	Rural	99.9	0.0	0.0	0.0	0.0	0.0	0.0
10	Niue	National	96.3	0.0	0.0	0.0	0.0	0.0	0.0
11	Tonga	National	91.7	0.0	0.0	0.0	0.0	0.0	0.0
12	Tuvalu	National	99.9	0.0	0.0	0.0	0.0	0.0	0.0

In [5]:
```
df = pd.read_csv('Clean3.csv')
df
```

Out[5]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.2.1 Population practicing open de...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Melanesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	· Fiji	National	0.15	0.13	0.11	0.08	0.00	0.00	0.00
4	· Fiji	Urban	0.06	0.05	0.05	0.05	0.00	0.00	0.00
5	· Fiji	Rural	0.27	0.22	0.18	0.13	0.00	0.00	0.00
6	· Papua New Guinea	National	15.33	15.53	15.73	15.93	16.13	16.12	16.10
7	· Papua New Guinea	Urban	4.00	4.01	4.03	4.04	4.05	4.05	4.05
8	· Papua New Guinea	Rural	17.03	17.27	17.51	17.75	17.99	17.99	17.99
9	· Solomon Islands	National	46.64	45.53	45.29	45.04	44.79	44.55	NaN
10	· Solomon Islands	Urban	4.73	4.36	4.36	4.36	4.36	4.36	NaN
11	· Solomon Islands	Rural	59.04	58.03	58.03	58.03	58.03	58.03	NaN
12	· Vanuatu	National	0.58	0.44	0.30	0.16	0.06	0.06	0.06
13	· Vanuatu	Urban	0.28	0.27	0.26	0.25	0.24	0.23	0.22
14	· Vanuatu	Rural	0.68	0.50	0.32	0.13	0.00	0.00	0.00
15	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	· Kiribati	National	35.62	35.13	34.64	34.16	33.68	33.22	32.75
17	· Kiribati	Urban	23.18	22.93	22.68	22.43	22.19	21.94	21.69
18	· Kiribati	Rural	49.35	49.03	48.71	48.40	48.08	47.76	47.45
19	· Marshall Islands	National	9.53	9.37	9.21	9.07	8.93	8.80	8.68
20	· Marshall Islands	Urban	4.41	4.55	4.69	4.83	4.97	5.11	5.25
21	· Marshall Islands	Rural	25.97	25.18	24.38	23.59	22.80	22.01	21.22
22	· Nauru	National	2.50	2.59	2.59	2.59	2.59	2.59	0.00
23	· Nauru	Urban	2.50	2.59	2.59	2.59	2.59	2.59	0.00
24	· Palau	National	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25	· Palau	Urban	0.00	0.00	0.00	0.00	0.00	0.00	0.00
26	· Palau	Rural	0.00	0.00	0.00	0.00	0.00	0.00	0.00
27	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	· Cook Islands	National	0.48	0.48	0.48	0.48	0.48	0.48	0.48
29	· Niue	National	0.50	0.50	0.50	0.50	0.50	0.50	0.50
30	· Samoa	National	0.10	0.11	0.11	0.11	0.12	0.12	0.13
31	· Samoa	Urban	0.01	0.01	0.01	0.01	0.01	0.01	0.01
32	· Samoa	Rural	0.12	0.13	0.13	0.14	0.14	0.15	0.15
33	· Tonga	National	0.12	0.12	0.11	0.11	0.10	0.10	0.09
34	· Tonga	Urban	0.08	0.08	0.09	0.09	0.09	0.09	0.09
35	· Tonga	Rural	0.13	0.13	0.12	0.11	0.11	0.10	0.09
36	· Tuvalu	National	4.27	3.98	3.70	3.42	3.15	2.88	2.62
37	· Tuvalu	Urban	3.04	2.89	2.73	2.58	2.42	2.26	2.11
38	· Tuvalu	Rural	6.15	5.72	5.30	4.88	4.45	4.03	3.60

In [6]:
```
df = pd.read_csv('Clean4.csv')
df
```

Out[6]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.2.1 Population with basic handwas...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Melanesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	· Fiji	National	0.00	86.26	86.31	86.37	86.42	86.47	86.53
4	· Fiji	Urban	0.00	91.12	91.12	91.12	91.12	91.12	91.12
5	· Fiji	Rural	0.00	80.13	80.13	80.13	80.13	80.13	80.13
6	· Papua New Guinea	National	29.65	29.67	29.70	29.73	29.77	29.81	29.86
7	· Papua New Guinea	Urban	62.48	62.48	62.48	62.48	62.48	62.48	62.48
8	· Papua New Guinea	Rural	24.73	24.73	24.73	24.73	24.73	24.73	24.73
9	· Solomon Islands	National	38.05	38.25	38.44	38.64	0.00	0.00	0.00
10	· Solomon Islands	Urban	70.67	70.67	70.67	70.67	0.00	0.00	0.00
11	· Solomon Islands	Rural	28.40	28.40	28.40	28.40	28.40	28.40	28.40
12	· Vanuatu	National	49.46	53.88	58.29	62.70	67.10	71.49	75.88
13	· Vanuatu	Urban	66.67	68.82	70.97	73.12	75.27	77.42	79.57
14	· Vanuatu	Rural	43.71	48.86	54.00	59.15	64.30	69.45	74.60
15	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	· Kiribati	National	55.34	55.40	55.46	55.52	55.58	55.63	55.68
17	· Kiribati	Urban	58.85	58.85	58.85	58.85	58.85	58.85	58.85
18	· Kiribati	Rural	51.48	51.48	51.48	51.48	51.48	51.48	51.48
19	· Marshall Islands	National	84.97	85.00	85.02	85.04	85.07	85.09	0.00
20	· Marshall Islands	Urban	86.44	86.44	86.44	86.44	86.44	86.44	0.00
21	· Marshall Islands	Rural	80.25	80.25	80.25	80.25	80.25	80.25	0.00
22	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	· Samoa	National	72.12	72.08	72.05	72.03	72.00	71.98	71.96
24	· Samoa	Urban	83.77	83.77	83.77	83.77	83.77	83.77	83.77
25	· Samoa	Rural	69.44	69.44	69.44	69.44	69.44	69.44	69.44
26	· Tonga	National	69.56	69.55	69.54	69.54	69.54	69.54	69.54
27	· Tonga	Urban	80.43	80.43	80.43	80.43	80.43	80.43	80.43
28	· Tonga	Rural	66.27	66.27	66.27	66.27	66.27	66.27	66.27
29	· Tuvalu	National	94.13	94.10	94.07	94.04	94.01	93.98	93.96
30	· Tuvalu	Urban	92.72	92.72	92.72	92.72	92.72	92.72	92.72
31	· Tuvalu	Rural	96.32	96.32	96.32	96.32	96.32	96.32	96.32

In [7]:
```
df = pd.read_csv('Clean5.csv')
df
```

Out[7]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.2.1 Population using safely manag...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Melanesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	· Fiji	National	48.42	48.72	49.00	49.27	49.24	49.02	48.79
4	· Fiji	Urban	42.91	42.96	43.00	43.05	42.95	42.81	42.67
5	· Fiji	Rural	55.23	55.97	56.70	57.43	57.65	57.49	57.33
6	· Papua New Guinea	Urban	28.93	28.78	28.63	28.49	28.34	28.34	28.34
7	· Vanuatu	Urban	33.86	33.19	32.52	31.85	31.18	30.51	29.84
8	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	· Kiribati	National	23.61	23.81	24.00	24.17	24.32	24.46	24.79
10	· Kiribati	Urban	24.71	24.65	24.58	24.50	24.42	24.32	24.60
11	· Kiribati	Rural	22.40	22.85	23.31	23.75	24.20	24.64	25.05
12	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	· American Samoa	National	37.87	37.06	37.04	37.01	36.99	36.96	36.99
14	· Samoa	National	46.29	45.71	45.13	44.54	44.00	43.47	42.93
15	· Samoa	Urban	39.48	38.84	38.20	37.55	37.25	37.00	36.74
16	· Samoa	Rural	47.85	47.27	46.67	46.07	45.47	44.86	44.25
17	· Tonga	National	34.29	33.85	33.40	32.95	32.54	32.26	32.05
18	· Tonga	Urban	25.00	24.29	23.58	22.88	22.33	22.37	22.34
19	· Tonga	Rural	37.10	36.73	36.35	35.98	35.61	35.24	34.97
20	· Tuvalu	National	40.21	39.68	39.17	38.66	38.17	37.70	37.23
21	· Tuvalu	Urban	36.06	35.83	35.60	35.37	35.15	34.92	34.70
22	· Tuvalu	Rural	46.62	45.85	45.08	44.32	43.56	42.81	42.05

In [8]:
```
df = pd.read_csv('Clean6.csv')
df
```

Out[8]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.2.1 Population using improved san...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	· Kiribati	National	0.0	0.0	60.6	0.0	0.0	0.0	0.0
4	· Kiribati	Urban	0.0	0.0	74.2	0.0	0.0	0.0	0.0
5	· Kiribati	Rural	0.0	0.0	44.8	0.0	0.0	0.0	0.0
6	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	· Niue	National	99.6	0.0	0.0	0.0	0.0	0.0	0.0
8	· Tonga	National	99.8	0.0	0.0	0.0	0.0	0.0	0.0
9	· Tuvalu	National	97.0	0.0	0.0	0.0	0.0	0.0	0.0

In [9]:
```
df = pd.read_csv('Clean7.csv')
df
```

Out[9]:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.3.1 Proportion of wastewater safe...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	· Kiribati	National	0.0	0.0	0.0	0.0	30.8	0.0	0.0
4	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	· American Samoa	National	0.0	0.0	0.0	0.0	69.0	0.0	0.0
6	· Samoa	National	0.0	6.6	0.0	0.0	46.7	0.0	0.0
7	· Tonga	National	0.0	0.0	0.0	0.0	28.6	0.0	0.0
8	· Tuvalu	National	0.0	0.0	0.0	0.0	2.0	0.0	0.0

In [10]:

```python
import pandas as pd

# Load the data
file_path = 'Clean7.csv'
df = pd.read_csv(file_path)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Drop rows where the first column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels (e.g., 'Urban', 'Rural', 'National')
# We assume 'Unnamed: 1' column contains this information

# Create a new column to store the urbanization level
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Display the cleaned DataFrame with Urbanization levels
print("\nCleaned DataFrame with Urbanization Levels:")
print(df)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Optionally, save these DataFrames to separate CSV files
urban_df.to_csv('Urban_Data.csv', index=False)
rural_df.to_csv('Rural_Data.csv', index=False)
national_df.to_csv('National_Data.csv', index=False)

# Display the separated DataFrames
print("\nUrban Data:")
print(urban_df)
print("\nRural Data:")
print(rural_df)
print("\nNational Data:")
print(national_df)
```

Original DataFrame:

	Time	Unnamed: 1	2016
0	Pacific Island Countries and territories	Urbanization	NaN
1	Indicator: 6.3.1 Proportion of wastewater safe...	NaN	NaN
2	Micronesia	NaN	NaN
3	· Kiribati	National	0.0
4	Polynesia	NaN	NaN
5	· American Samoa	National	0.0
6	· Samoa	National	0.0
7	· Tonga	National	0.0
8	· Tuvalu	National	0.0

	2017	2018	2019	2020	2021	2022
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	0.0	0.0	0.0	30.8	0.0	0.0
4	NaN	NaN	NaN	NaN	NaN	NaN
5	0.0	0.0	0.0	69.0	0.0	0.0
6	6.6	0.0	0.0	46.7	0.0	0.0
7	0.0	0.0	0.0	28.6	0.0	0.0
8	0.0	0.0	0.0	2.0	0.0	0.0

Cleaned DataFrame with Urbanization Levels:

	Time	Unnamed: 1	2016	2017	2018
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN
1	· Kiribati	National	0.0	0.0	0.0
2	· American Samoa	National	0.0	0.0	0.0
3	· Samoa	National	0.0	6.6	0.0
4	· Tonga	National	0.0	0.0	0.0
5	· Tuvalu	National	0.0	0.0	0.0

	2019	2020	2021	2022	Urbanization
0	NaN	NaN	NaN	NaN	Urban
1	0.0	30.8	0.0	0.0	National
2	0.0	69.0	0.0	0.0	National
3	0.0	46.7	0.0	0.0	National
4	0.0	28.6	0.0	0.0	National
5	0.0	2.0	0.0	0.0	National

Urban Data:

	Time	Unnamed: 1	2016	2017	2018
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN

	2019	2020	2021	2022	Urbanization
0	NaN	NaN	NaN	NaN	Urban

Rural Data:

Empty DataFrame

Columns: [Time, Unnamed: 1, 2016, 2017, 2018, 2019, 2020, 2021, 2022, Urbanization]

Index: []

National Data:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
1	· Kiribati	National	0.0	0.0	0.0	0.0	30.8	0.0	0.0
2	· American Samoa	National	0.0	0.0	0.0	0.0	69.0	0.0	0.0
3	· Samoa	National	0.0	6.6	0.0	0.0	46.7	0.0	0.0
4	· Tonga	National	0.0	0.0	0.0	0.0	28.6	0.0	0.0
5	· Tuvalu	National	0.0	0.0	0.0	0.0	2.0	0.0	0.0

	Urbanization
1	National
2	National
3	National
4	National
5	National

In [11]:

```python
import pandas as pd

# Load the data
file_path = 'Clean6.csv'
df = pd.read_csv(file_path)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Drop rows where the first column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels (e.g., 'Urban', 'Rural', 'National')
# We assume 'Unnamed: 1' column contains this information

# Create a new column to store the urbanization level
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Display the cleaned DataFrame with Urbanization levels
print("\nCleaned DataFrame with Urbanization Levels:")
print(df)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Optionally, save these DataFrames to separate CSV files
urban_df.to_csv('Urban_Data.csv', index=False)
rural_df.to_csv('Rural_Data.csv', index=False)
national_df.to_csv('National_Data.csv', index=False)

# Display the separated DataFrames
print("\nUrban Data:")
print(urban_df)
print("\nRural Data:")
print(rural_df)
print("\nNational Data:")
print(national_df)
```

Original DataFrame:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Indicator: 6.2.1 Population using improved san...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Micronesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	· Kiribati	National	0.0	0.0	60.6	0.0	0.0	0.0	0.0
4	· Kiribati	Urban	0.0	0.0	74.2	0.0	0.0	0.0	0.0
5	· Kiribati	Rural	0.0	0.0	44.8	0.0	0.0	0.0	0.0
6	Polynesia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	· Niue	National	99.6	0.0	0.0	0.0	0.0	0.0	0.0
8	· Tonga	National	99.8	0.0	0.0	0.0	0.0	0.0	0.0
9	· Tuvalu	National	97.0	0.0	0.0	0.0	0.0	0.0	0.0

Cleaned DataFrame with Urbanization Levels:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022	Urbanization
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Urban
1	· Kiribati	National	0.0	0.0	60.6	0.0	0.0	0.0	0.0	National
2	· Kiribati	Urban	0.0	0.0	74.2	0.0	0.0	0.0	0.0	Urban
3	· Kiribati	Rural	0.0	0.0	44.8	0.0	0.0	0.0	0.0	Rural
4	· Niue	National	99.6	0.0	0.0	0.0	0.0	0.0	0.0	National
5	· Tonga	National	99.8	0.0	0.0	0.0	0.0	0.0	0.0	National
6	· Tuvalu	National	97.0	0.0	0.0	0.0	0.0	0.0	0.0	National

Urban Data:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022	Urbanization
0	Pacific Island Countries and territories	Urbanization	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Urban
2	· Kiribati	Urban	0.0	0.0	74.2	0.0	0.0	0.0	0.0	Urban

Rural Data:

	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022	Urbanization
3	· Kiribati	Rural	0.0	0.0	44.8	0.0	0.0	0.0	0.0	Rural

National Data:

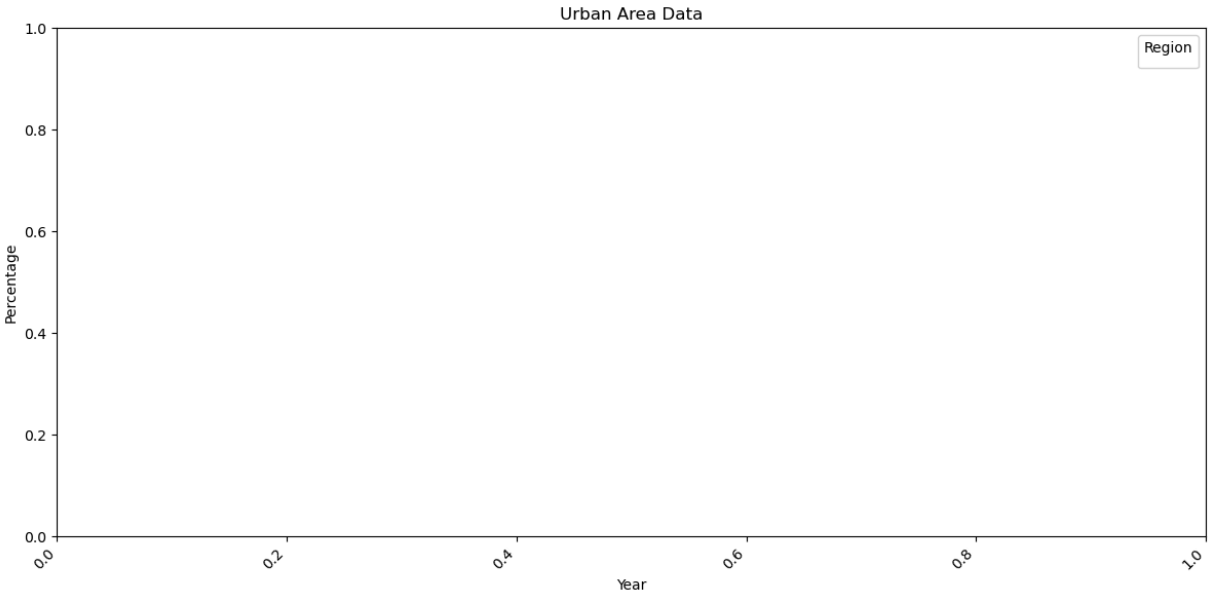
	Time	Unnamed: 1	2016	2017	2018	2019	2020	2021	2022	Urbanization
1	· Kiribati	National	0.0	0.0	60.6	0.0	0.0	0.0	0.0	National
4	· Niue	National	99.6	0.0	0.0	0.0	0.0	0.0	0.0	National
5	· Tonga	National	99.8	0.0	0.0	0.0	0.0	0.0	0.0	National
6	· Tuvalu	National	97.0	0.0	0.0	0.0	0.0	0.0	0.0	National

5	National
6	National

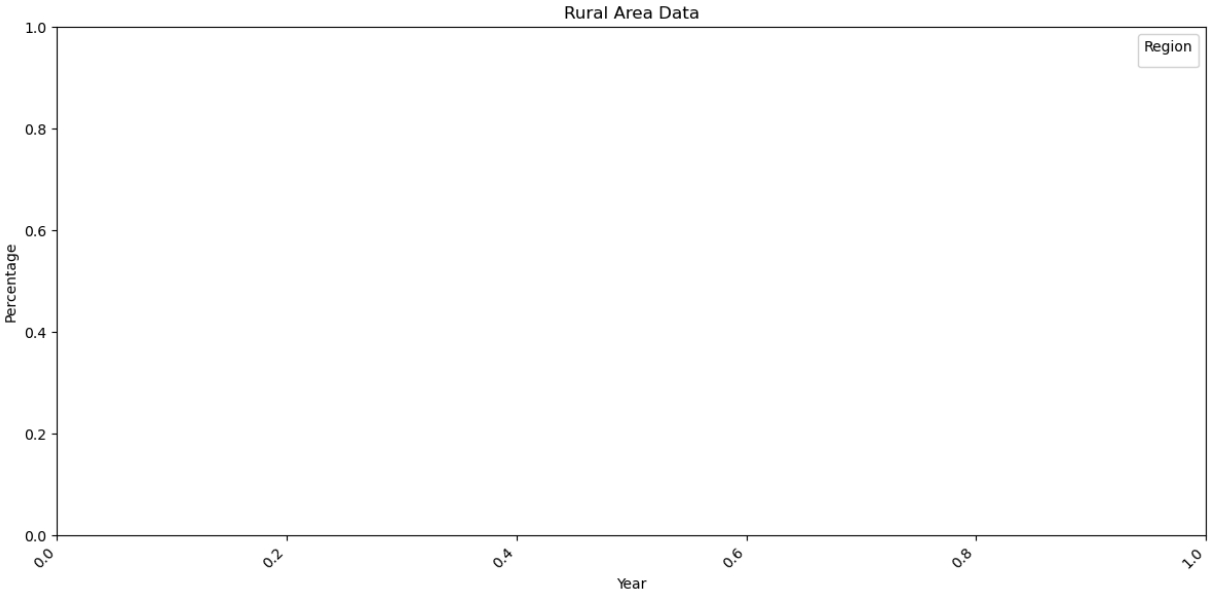
In [12]:

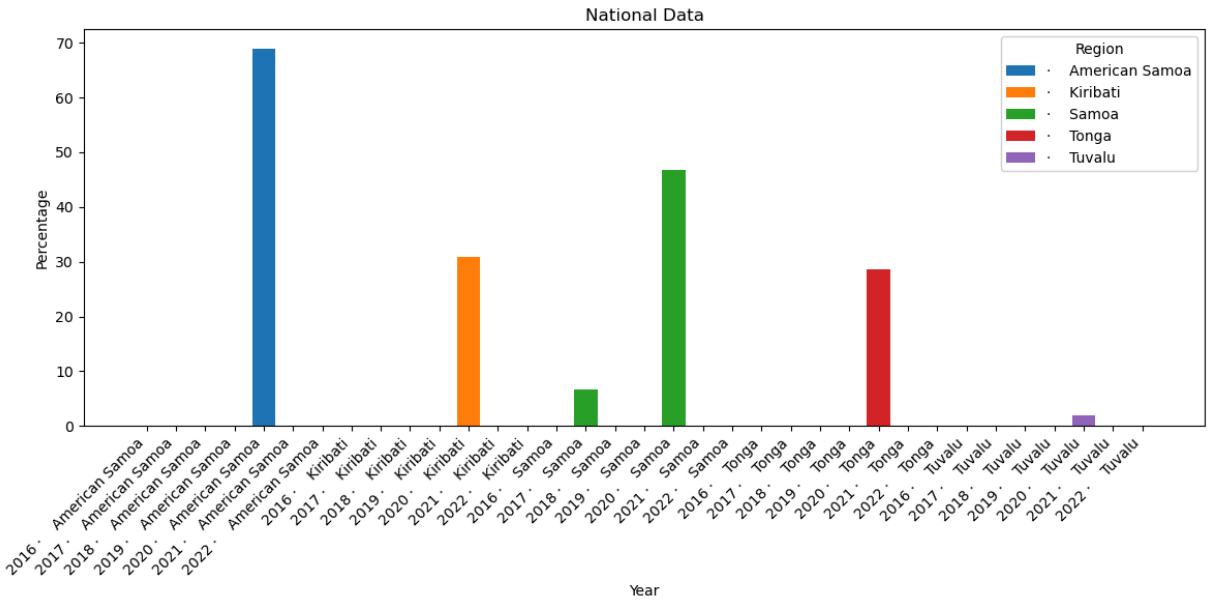
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean7.csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an u
nderscore are ignored when legend() is called with no argument.
```



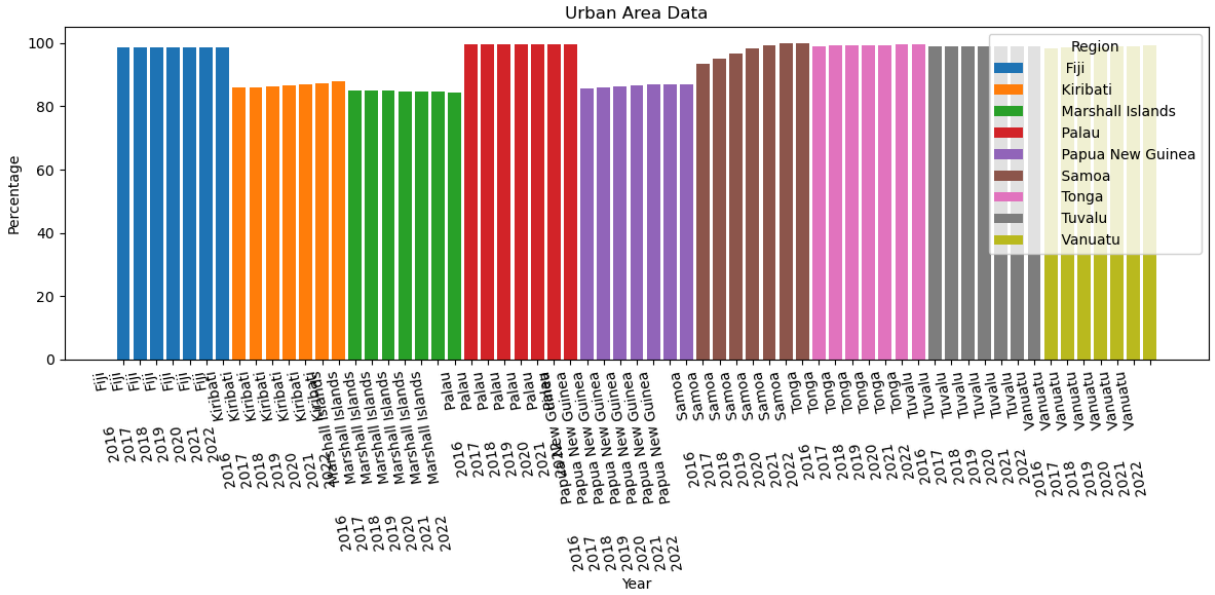

```
No artists with labels found to put in legend.  Note that artists whose label start with an u
nderscore are ignored when legend() is called with no argument.
```

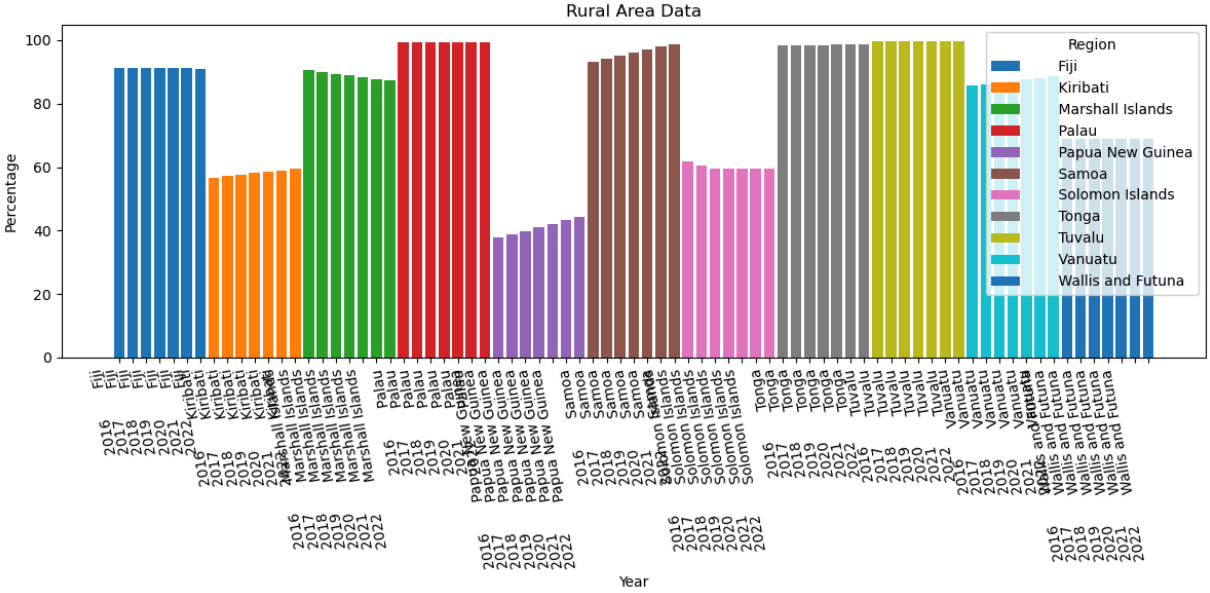


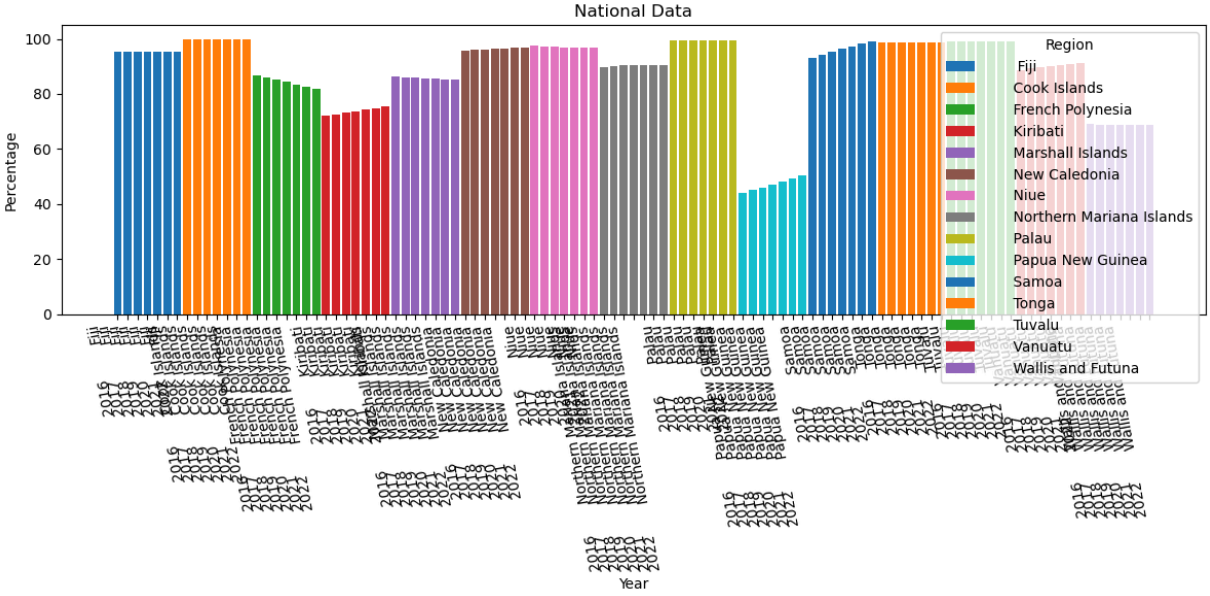


In [13]:

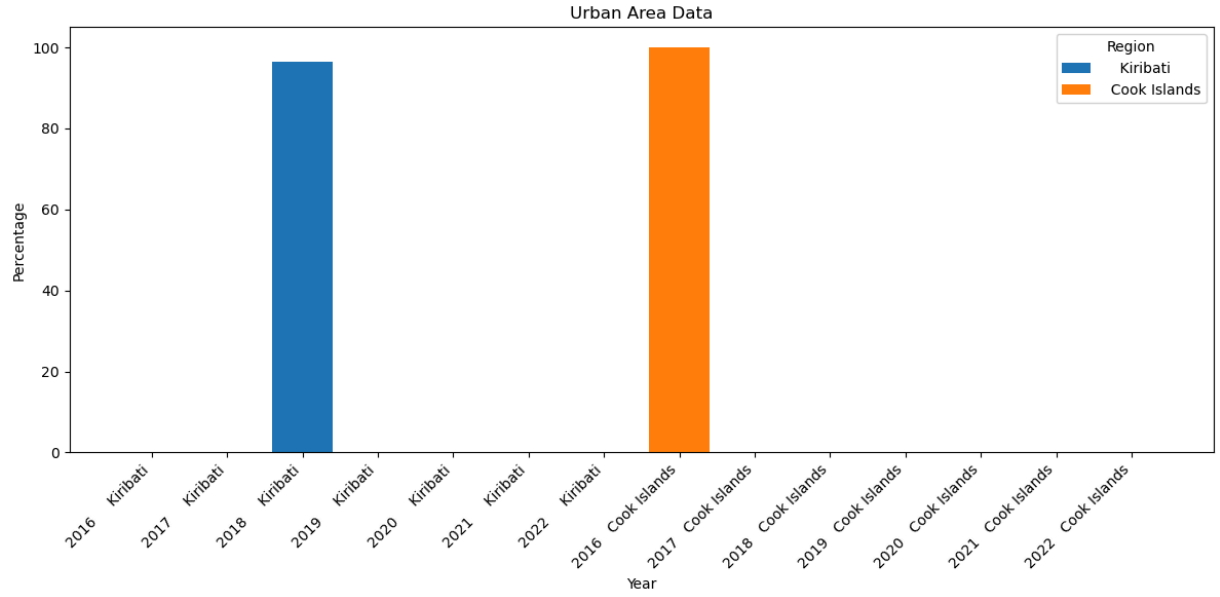
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean1 (1).csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=100, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

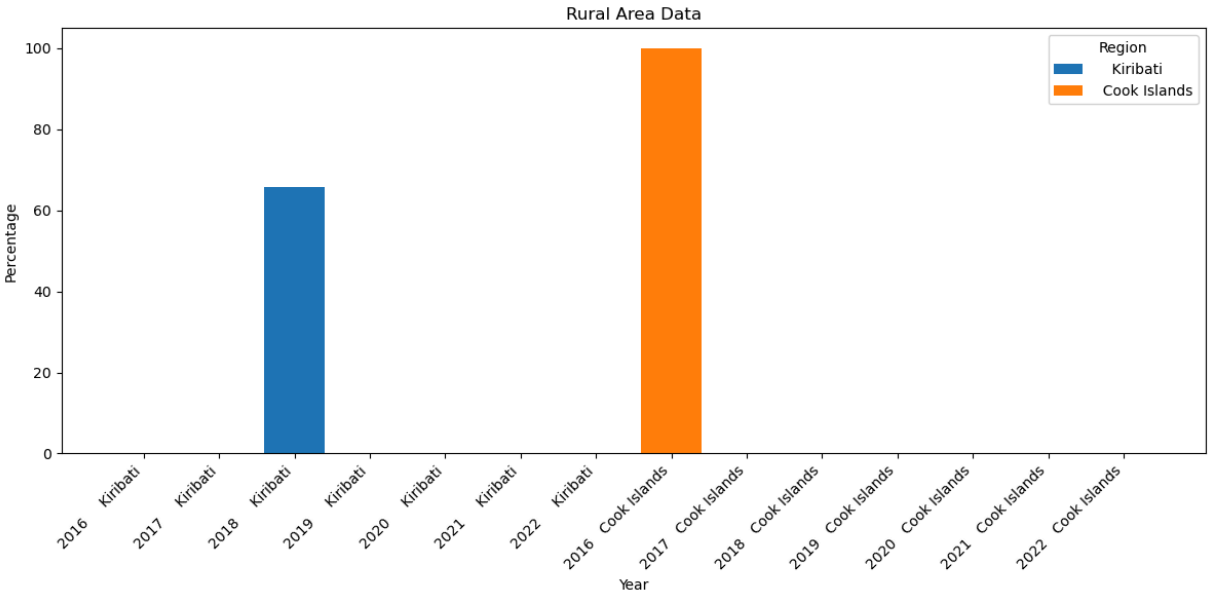


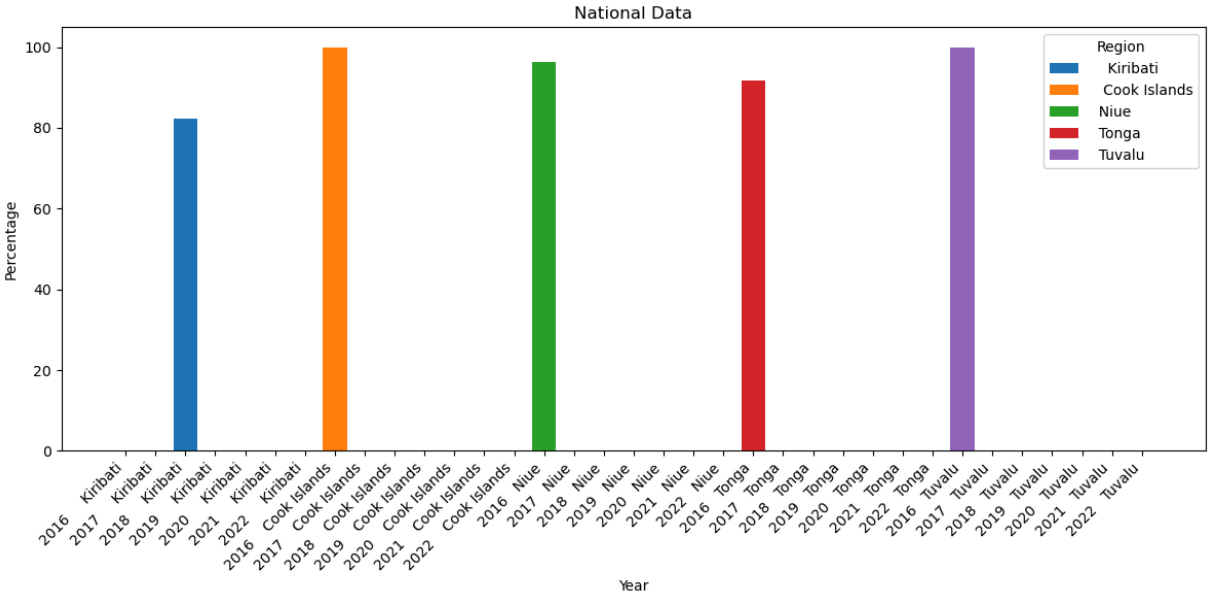

Rural Area Data
Percentage
Year
Region
Fiji
Kiribati
Marshall Islands
Palau
Papua New Guinea
Samoa
Solomon Islands
Tonga
Tuvalu
Vanuatu
Wallis and Futuna


National Data
Percentage
Year
Region
Fiji
Cook Islands
French Polynesia
Kiribati
Marshall Islands
New Caledonia
Niue
Northern Mariana Islands
Palau
Papua New Guinea
Samoa
Tonga
Tuvalu
Vanuatu
Wallis and Futuna

In [14]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean2.csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

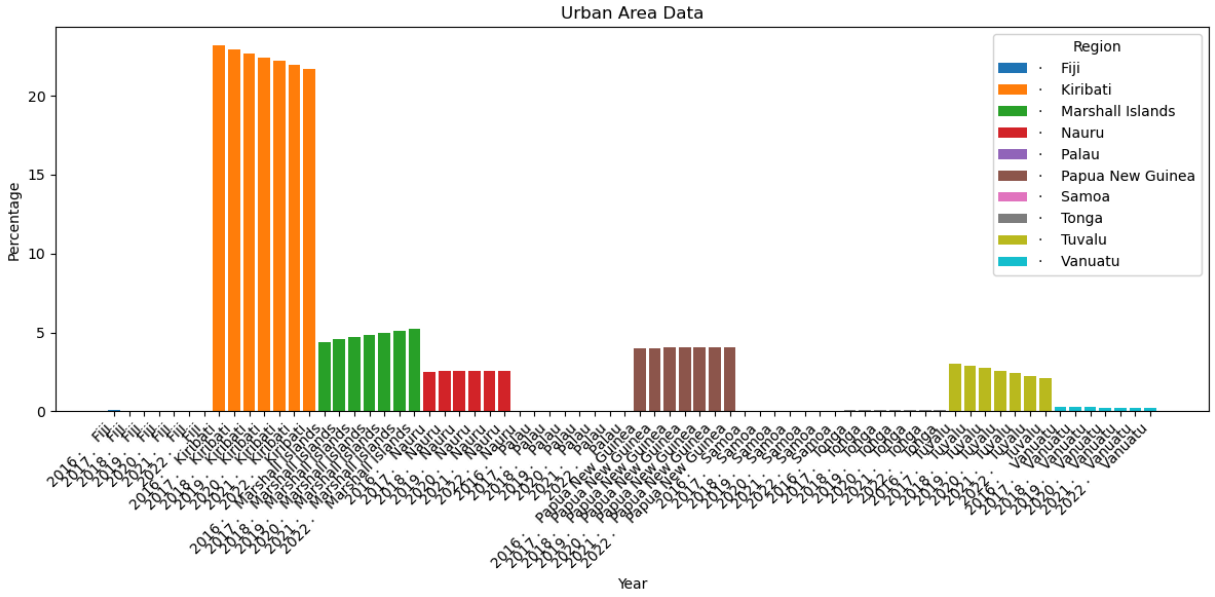


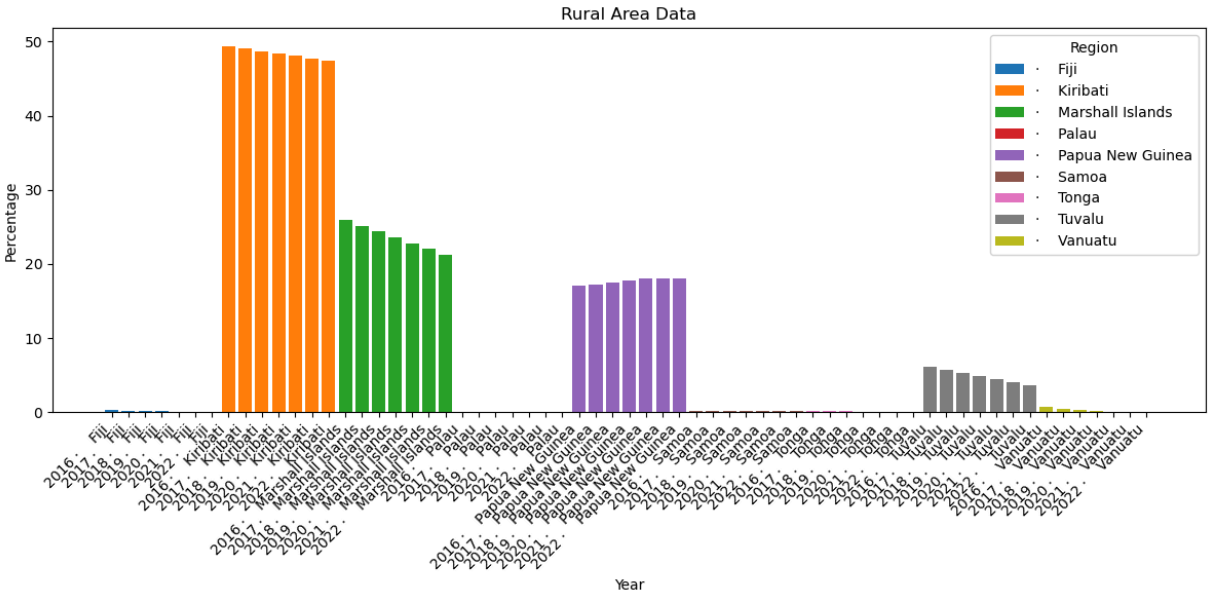

Rural Area Data
Region
Kiribati
Cook Islands
Percentage
Year
0
20
40
60
80
100
2016 Kiribati
2017 Kiribati
2018 Kiribati
2019 Kiribati
2020 Kiribati
2021 Kiribati
2022 Kiribati
2016 Cook Islands
2017 Cook Islands
2018 Cook Islands
2019 Cook Islands
2020 Cook Islands
2021 Cook Islands
2022 Cook Islands

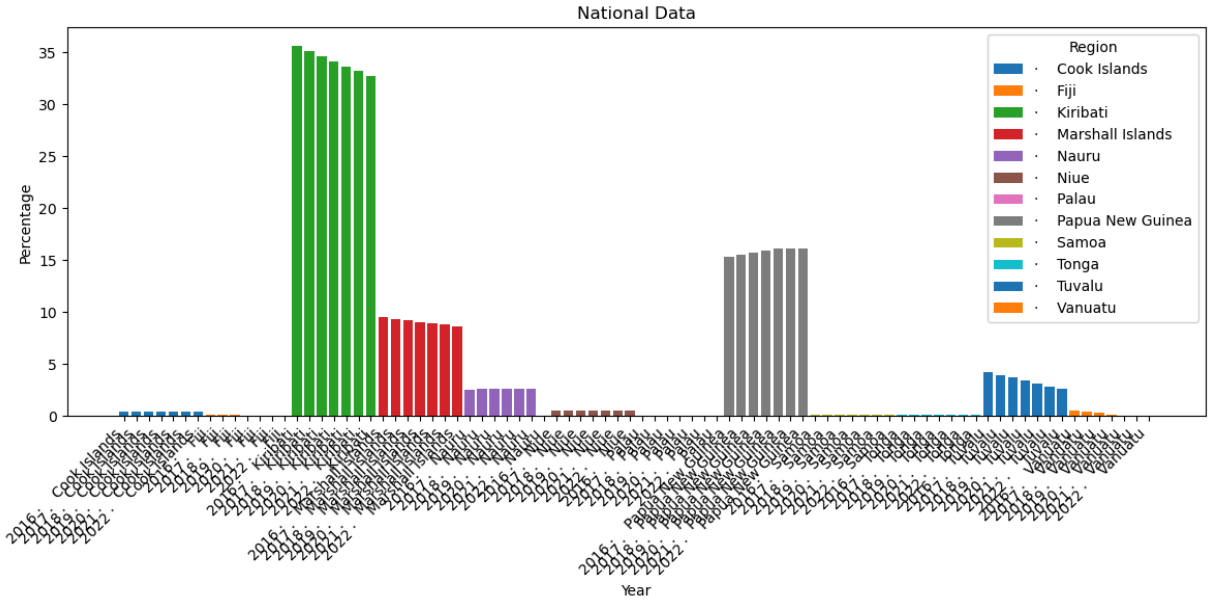

National Data
Region
Kiribati
Cook Islands
Niue
Tonga
Tuvalu
Percentage
Year
0
20
40
60
80
100
2016 Kiribati
2017 Kiribati
2018 Kiribati
2019 Kiribati
2020 Kiribati
2021 Kiribati
2022 Kiribati
2016 Cook Islands
2017 Cook Islands
2018 Cook Islands
2019 Cook Islands
2020 Cook Islands
2021 Cook Islands
2022 Cook Islands
2016 Niue
2017 Niue
2018 Niue
2019 Niue
2020 Niue
2021 Niue
2022 Niue
2016 Tonga
2017 Tonga
2018 Tonga
2019 Tonga
2020 Tonga
2021 Tonga
2022 Tonga
2016 Tuvalu
2017 Tuvalu
2018 Tuvalu
2019 Tuvalu
2020 Tuvalu
2021 Tuvalu
2022 Tuvalu

In [15]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean3.csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

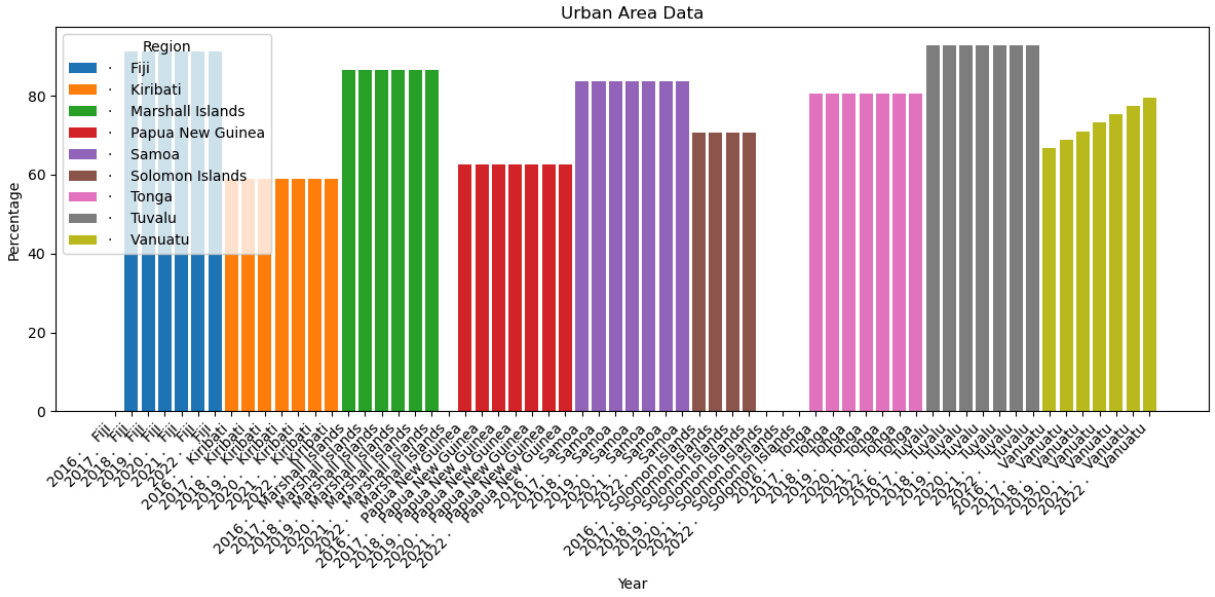


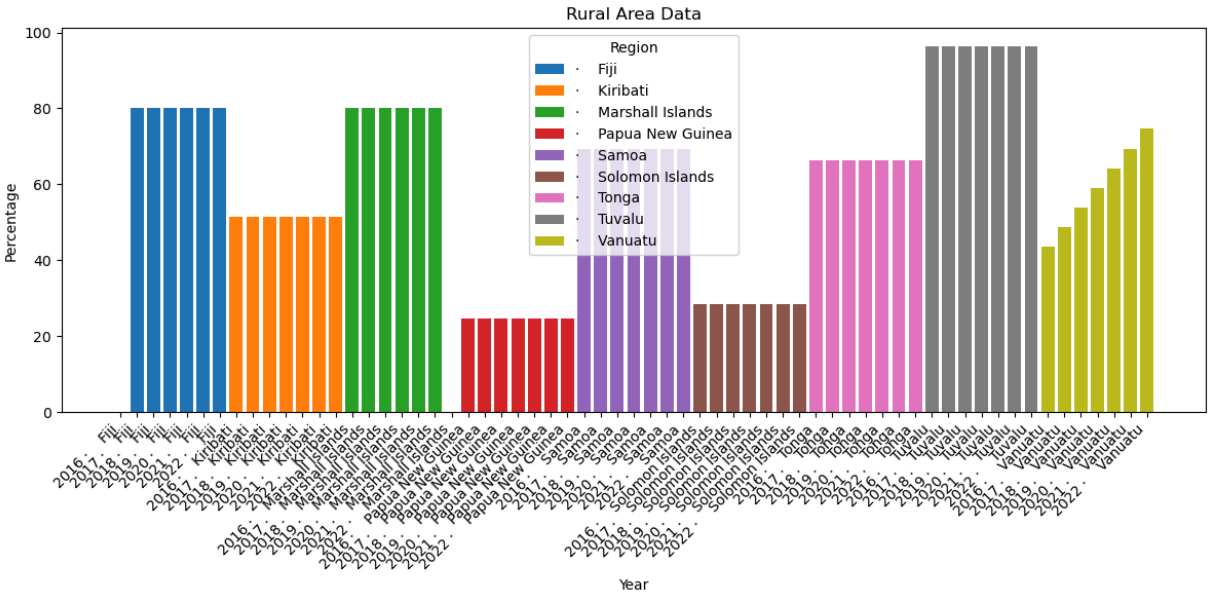

Rural Area Data
Percentage
Year
Region
Fiji
Kiribati
Marshall Islands
Palau
Papua New Guinea
Samoa
Tonga
Tuvalu
Vanuatu

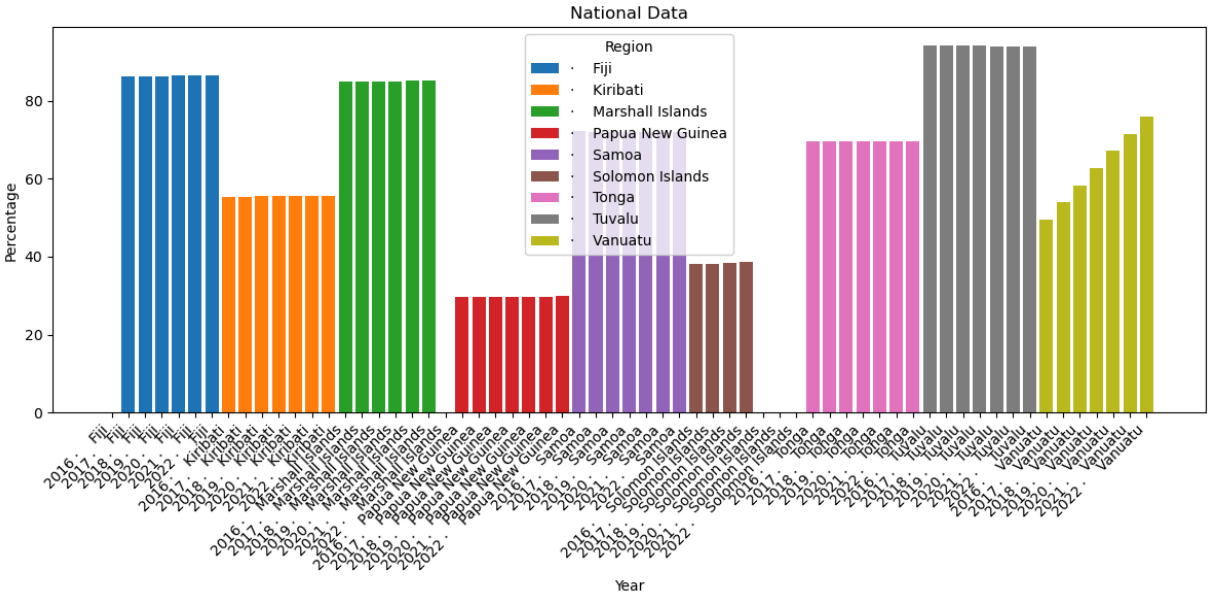

National Data
Percentage
Year
Region
Cook Islands
Fiji
Kiribati
Marshall Islands
Nauru
Niue
Palau
Papua New Guinea
Samoa
Tonga
Tuvalu
Vanuatu

In [16]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean4.csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

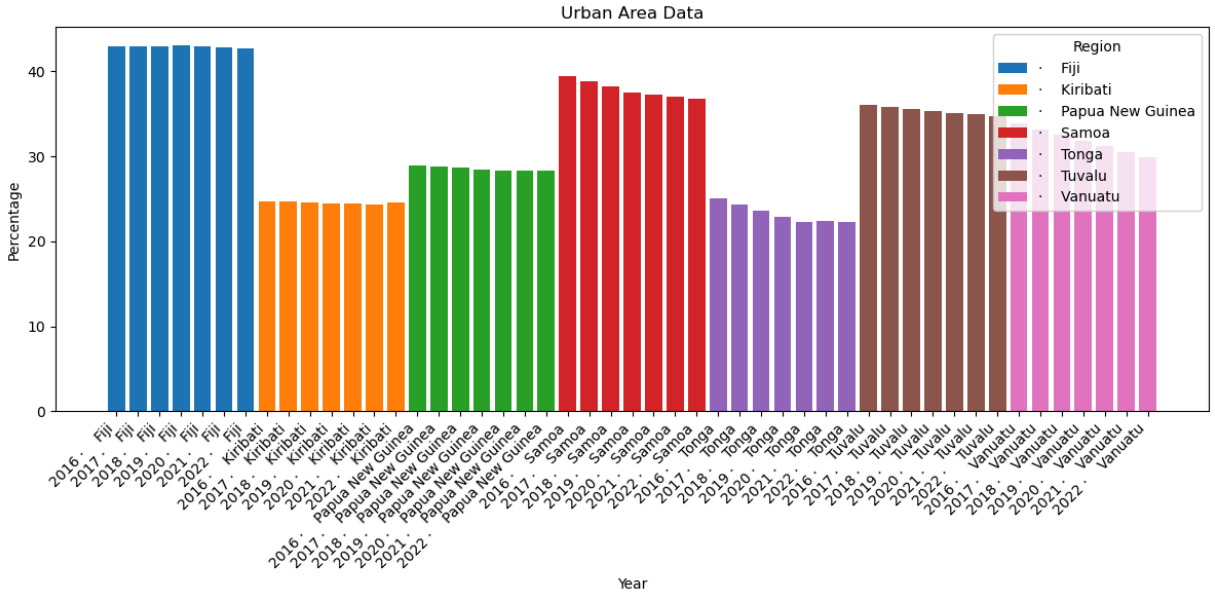


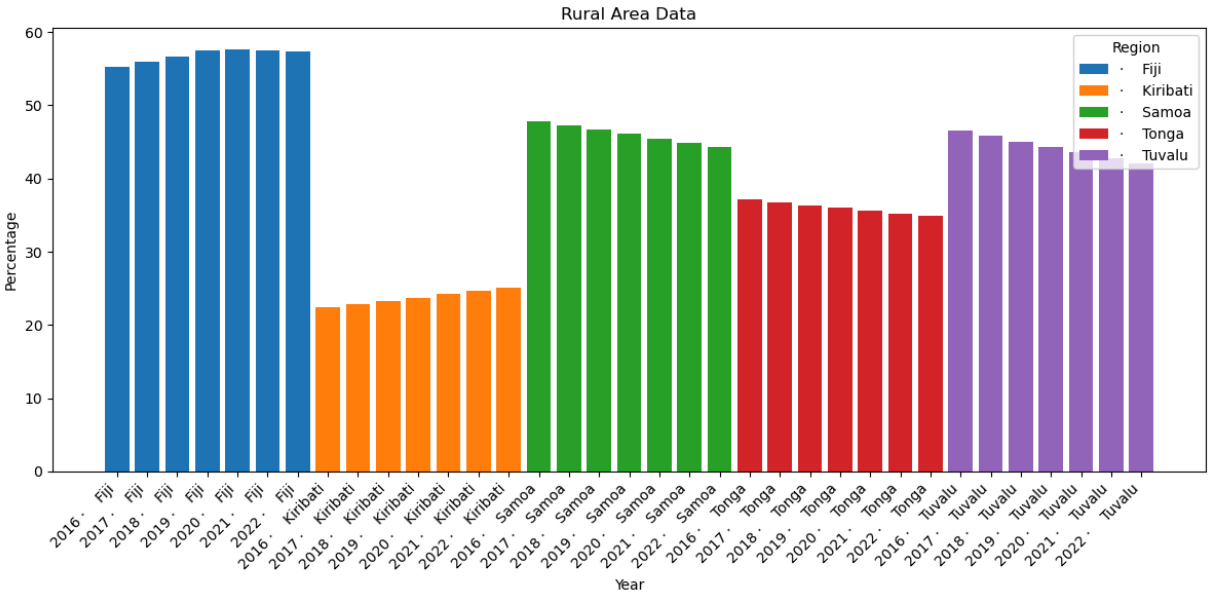

Rural Area Data
Region
Fiji
Kiribati
Marshall Islands
Papua New Guinea
Samoa
Solomon Islands
Tonga
Tuvalu
Vanuatu
Percentage
Year

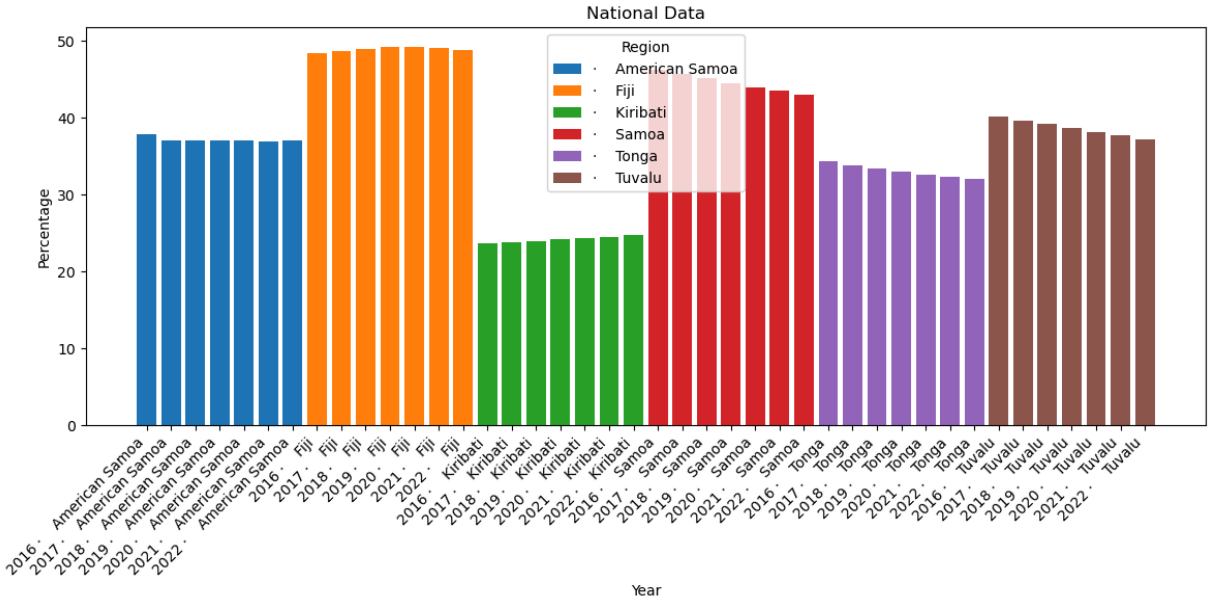

National Data
Region
Fiji
Kiribati
Marshall Islands
Papua New Guinea
Samoa
Solomon Islands
Tonga
Tuvalu
Vanuatu
Percentage
Year

In [17]:

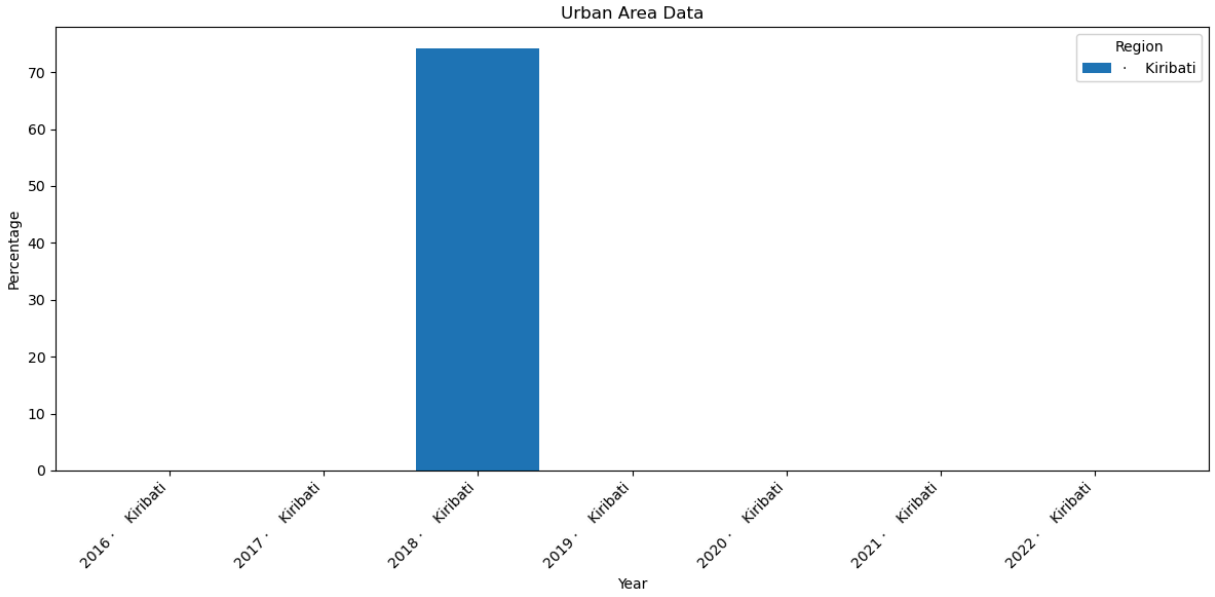
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean5.csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

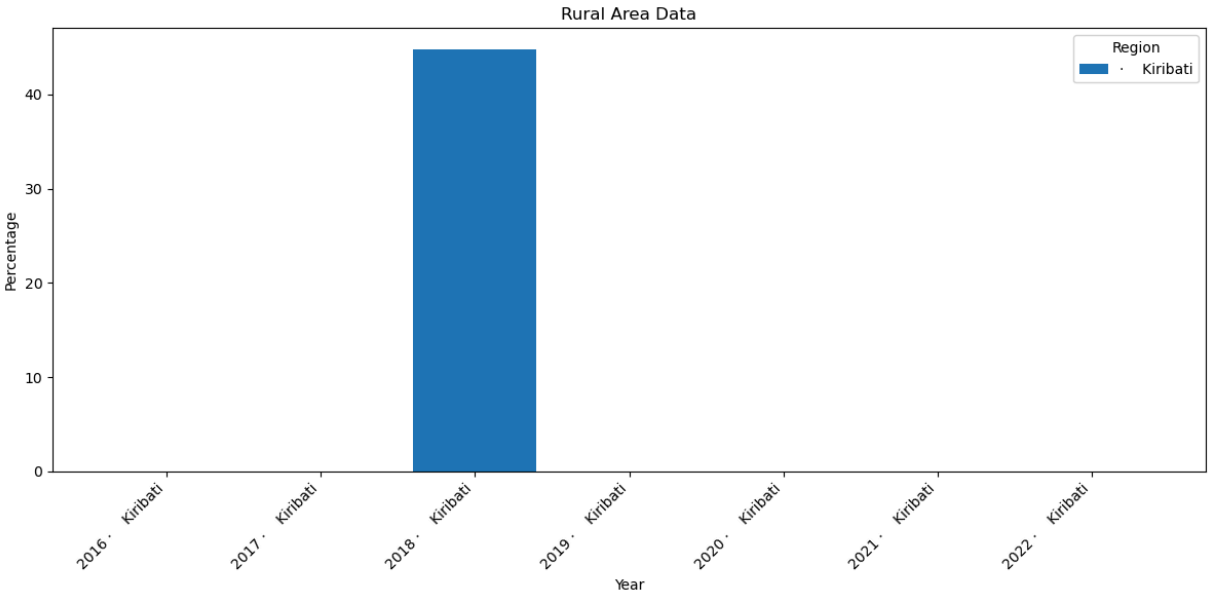


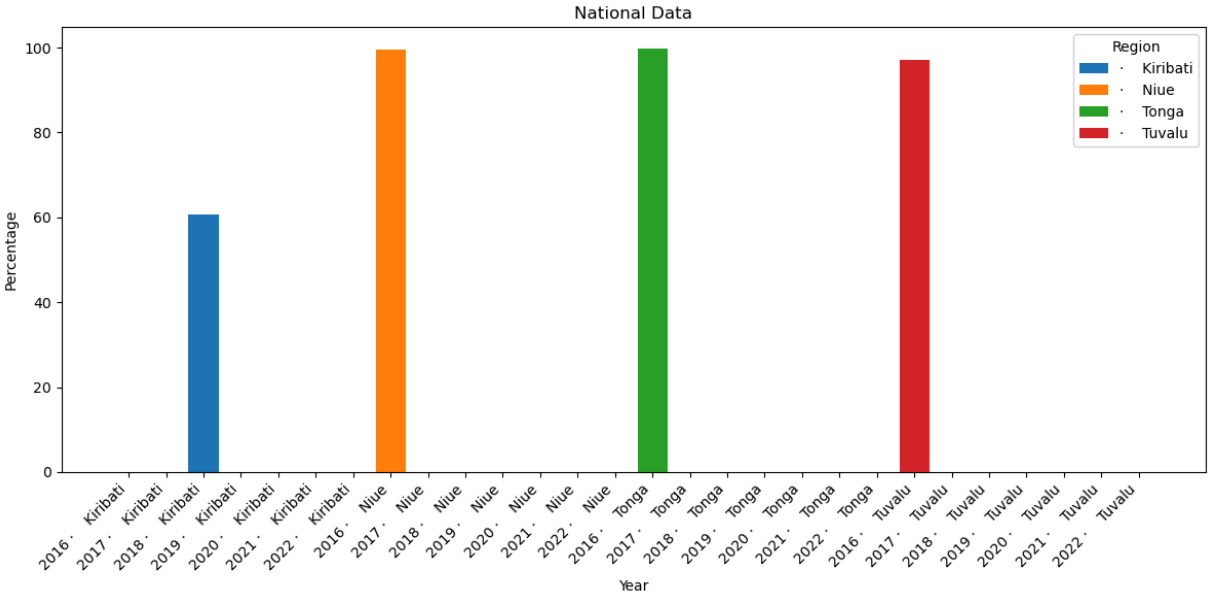

Rural Area Data
Region
Fiji
Kiribati
Samoa
Tonga
Tuvalu
Percentage
Year


National Data
Region
American Samoa
Fiji
Kiribati
Samoa
Tonga
Tuvalu
Percentage
Year

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'Clean6.csv'
df = pd.read_csv(file_path)

# Drop rows where the 'Time' column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Plotting function
def plot_bar_graph(df, title):
    df = df.dropna(subset=['2016', '2017', '2018', '2019', '2020', '2021', '2022'])
    df = df.melt(id_vars=['Time', 'Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    plt.figure(figsize=(12, 6))
    for label, group_df in df.groupby('Time'):
        plt.bar(group_df['Year'] + ' ' + label, group_df['Percentage'], label=label)
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.title(title)
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Region')
    plt.tight_layout()
    plt.show()

# Plotting bar graphs for each urbanization level
plot_bar_graph(urban_df, 'Urban Area Data')
plot_bar_graph(rural_df, 'Rural Area Data')
plot_bar_graph(national_df, 'National Data')
```

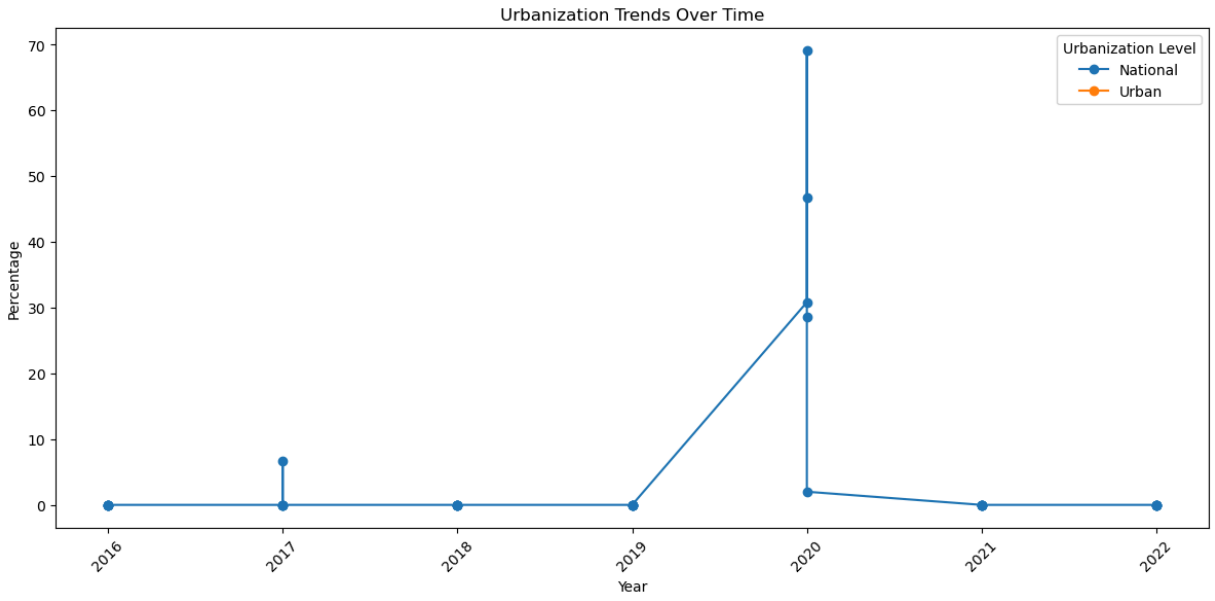



Rural Area Data
Region
Kiribati
Percentage
0
10
20
30
40
2016 · Kiribati
2017 · Kiribati
2018 · Kiribati
2019 · Kiribati
2020 · Kiribati
2021 · Kiribati
2022 · Kiribati
Year


National Data
Region
Kiribati
Niue
Tonga
Tuvalu
Percentage
0
20
40
60
80
100
2016 · Kiribati
2017 · Kiribati
2018 · Kiribati
2019 · Kiribati
2020 · Kiribati
2021 · Kiribati
2022 · Kiribati
2016 · Niue
2017 · Niue
2018 · Niue
2019 · Niue
2020 · Niue
2021 · Niue
2022 · Niue
2016 · Tonga
2017 · Tonga
2018 · Tonga
2019 · Tonga
2020 · Tonga
2021 · Tonga
2022 · Tonga
2016 · Tuvalu
2017 · Tuvalu
2018 · Tuvalu
2019 · Tuvalu
2020 · Tuvalu
2021 · Tuvalu
2022 · Tuvalu
Year

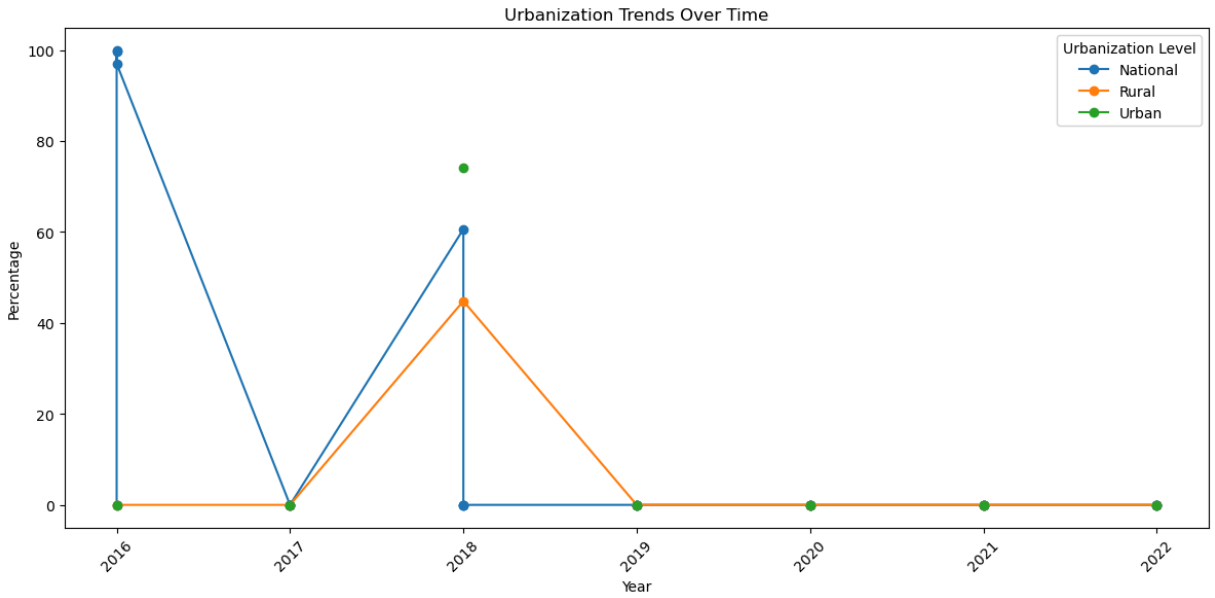
In [19]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean7.csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```



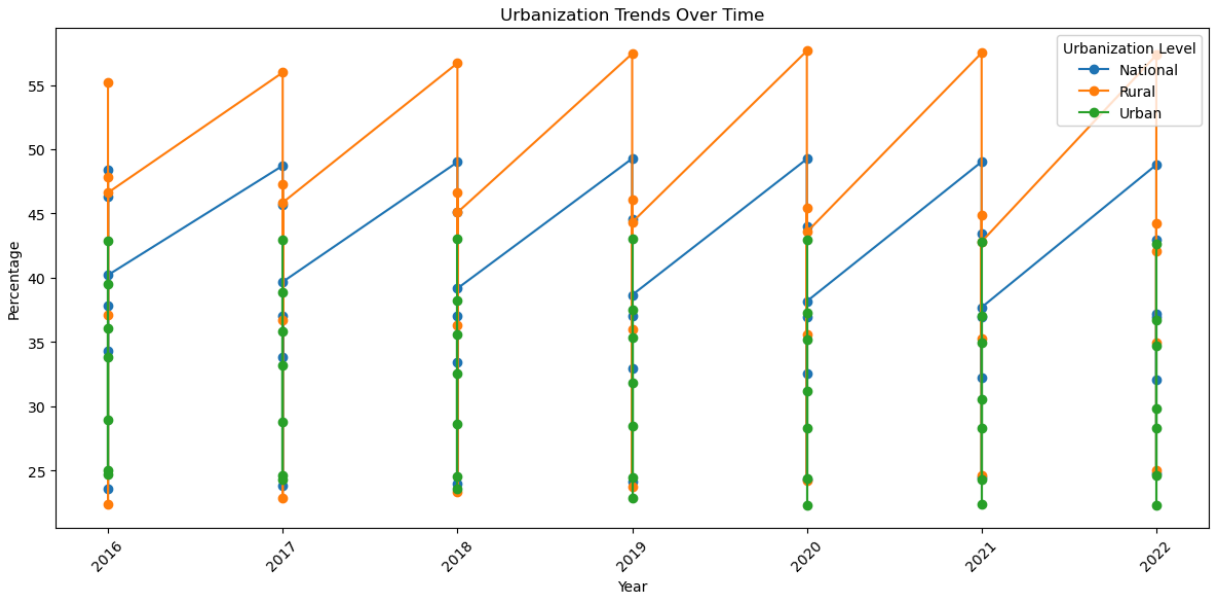
In [20]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean6.csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```



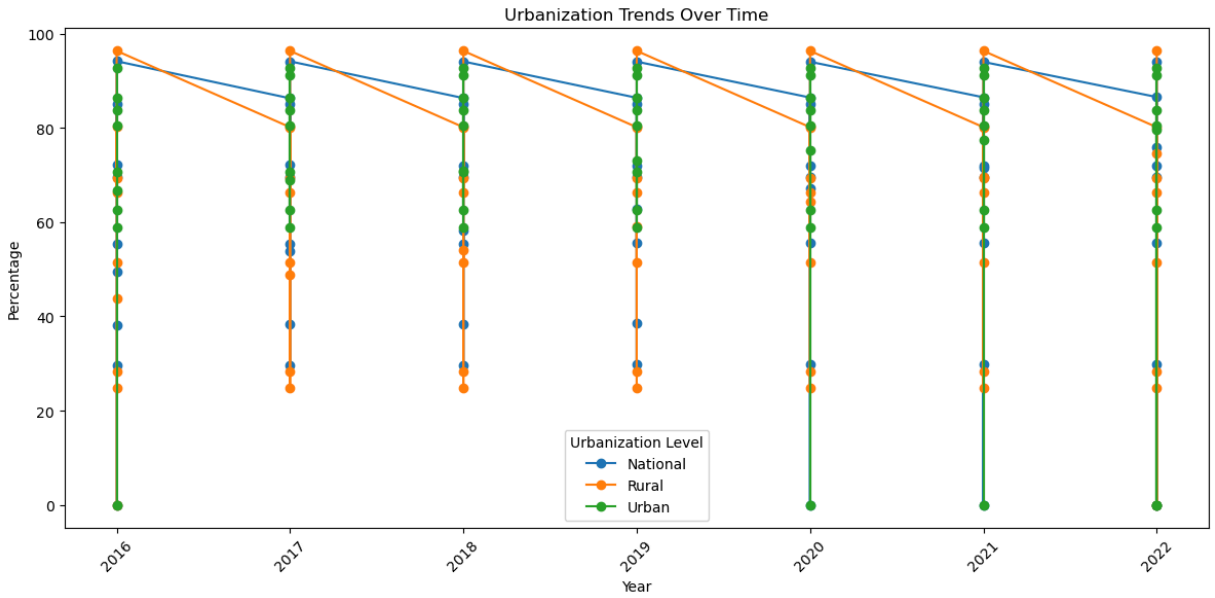
In [21]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean5.csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```

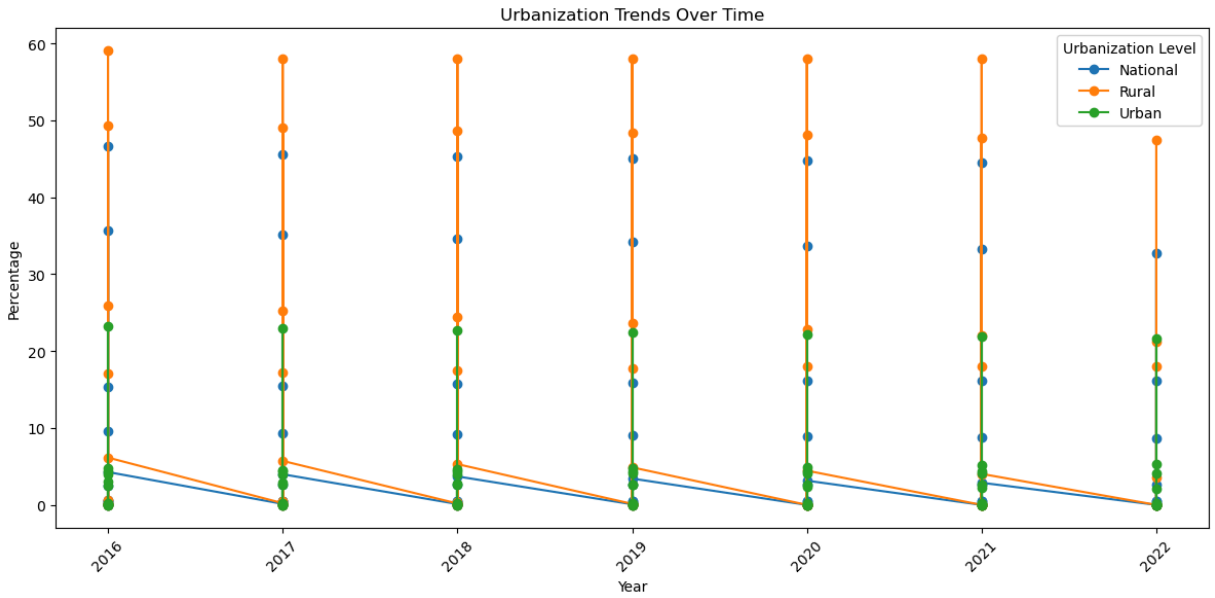


In [22]:

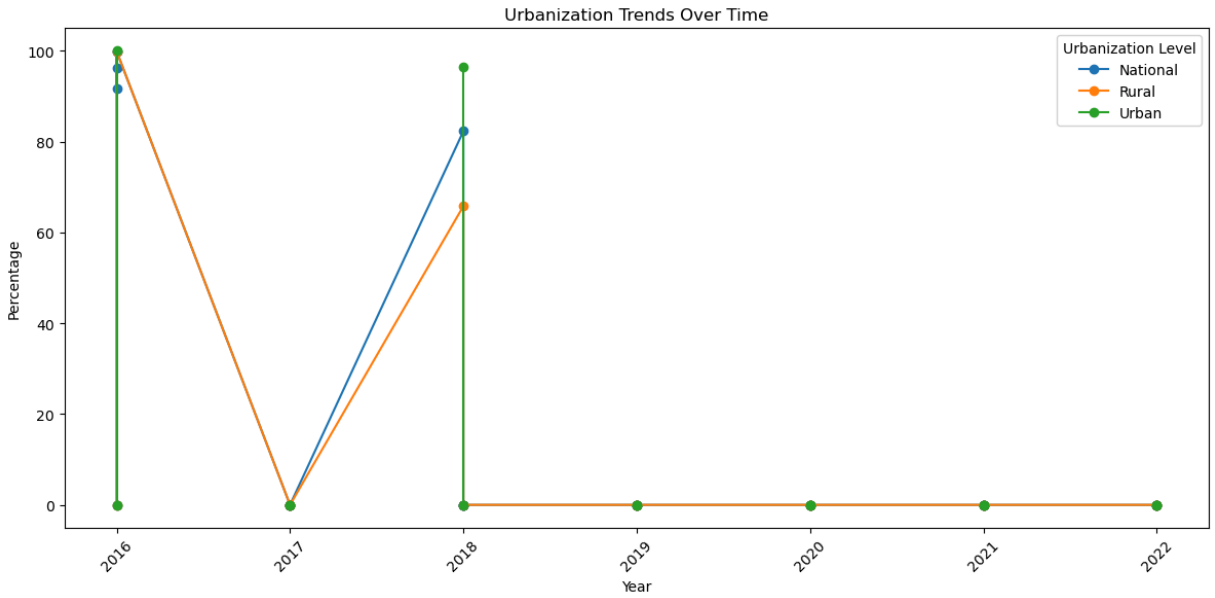
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean4.csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```



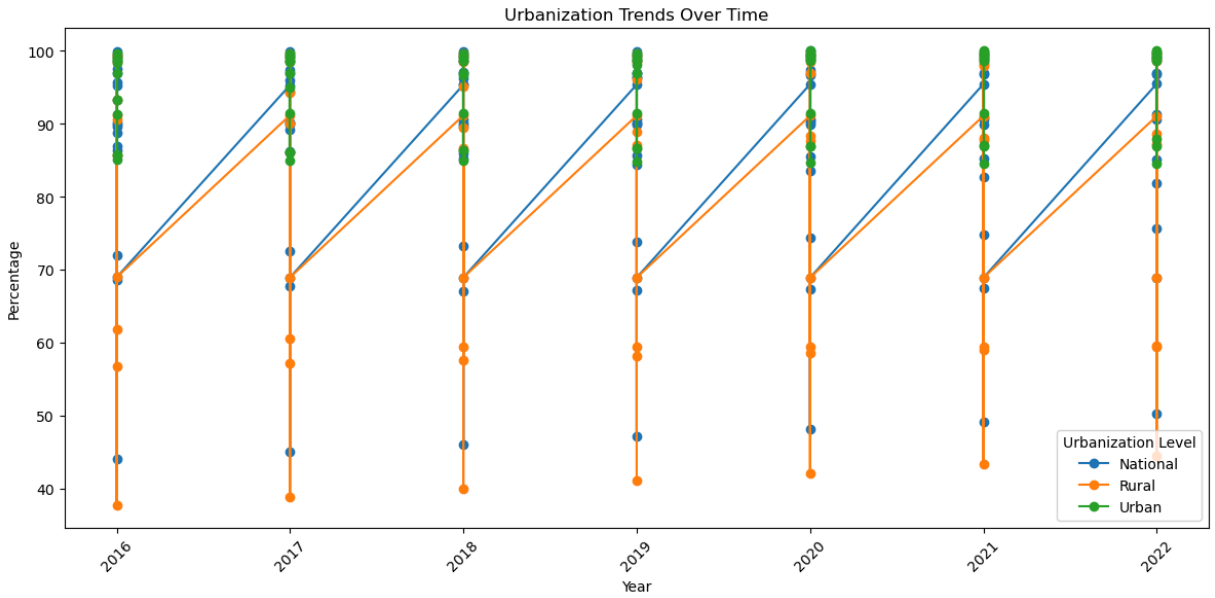
In [23]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean3.csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```



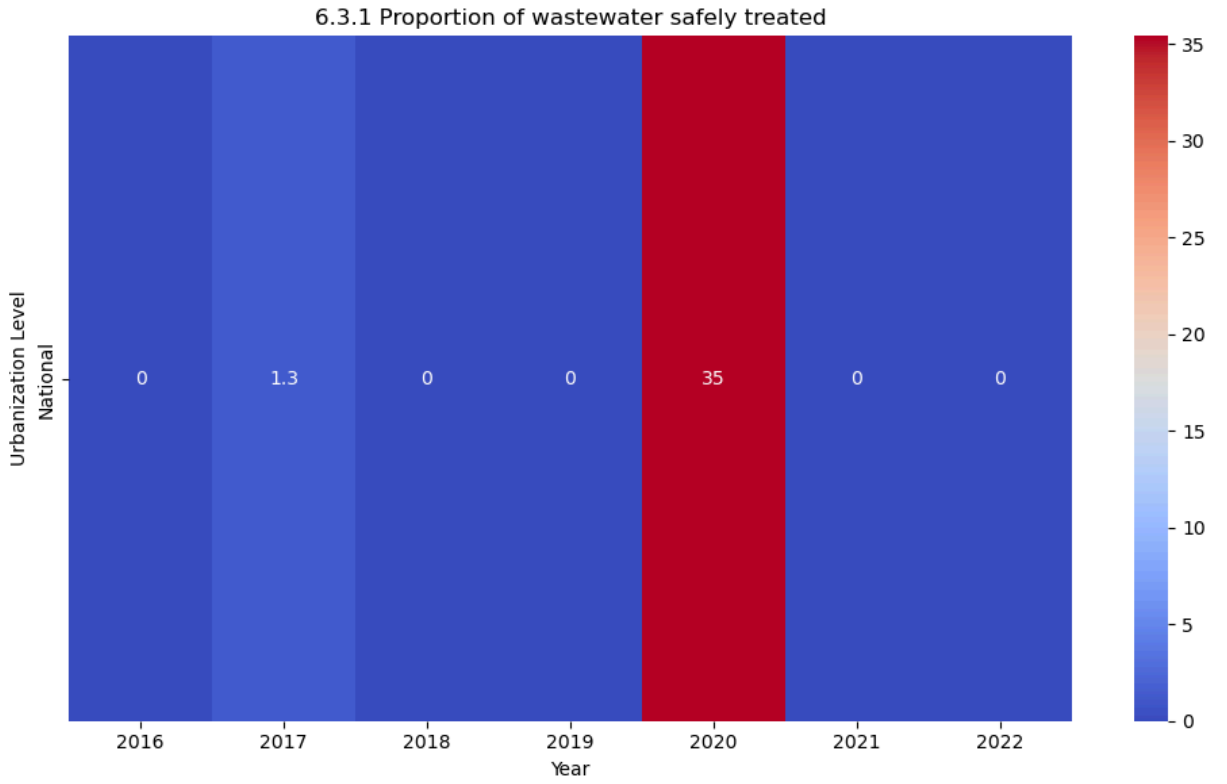
In [24]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean2.csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```



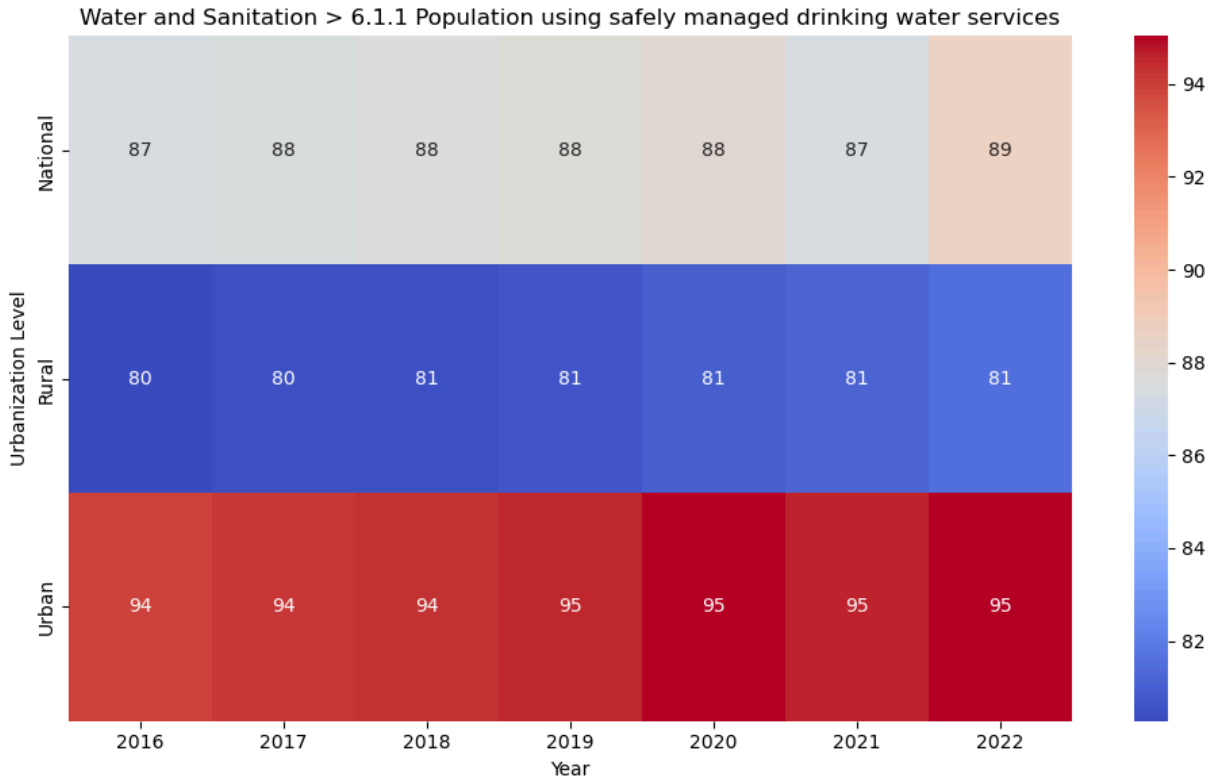
In [25]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('Clean1 (1).csv')

# Clean the data as described before
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for grouped bar chart
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Plot
plt.figure(figsize=(12, 6))
for label, group_df in df_melted.groupby('Urbanization'):
    plt.plot(group_df['Year'], group_df['Percentage'], marker='o', label=label)
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Urbanization Trends Over Time')
plt.xticks(rotation=45)
plt.legend(title='Urbanization Level')
plt.tight_layout()
plt.show()
```



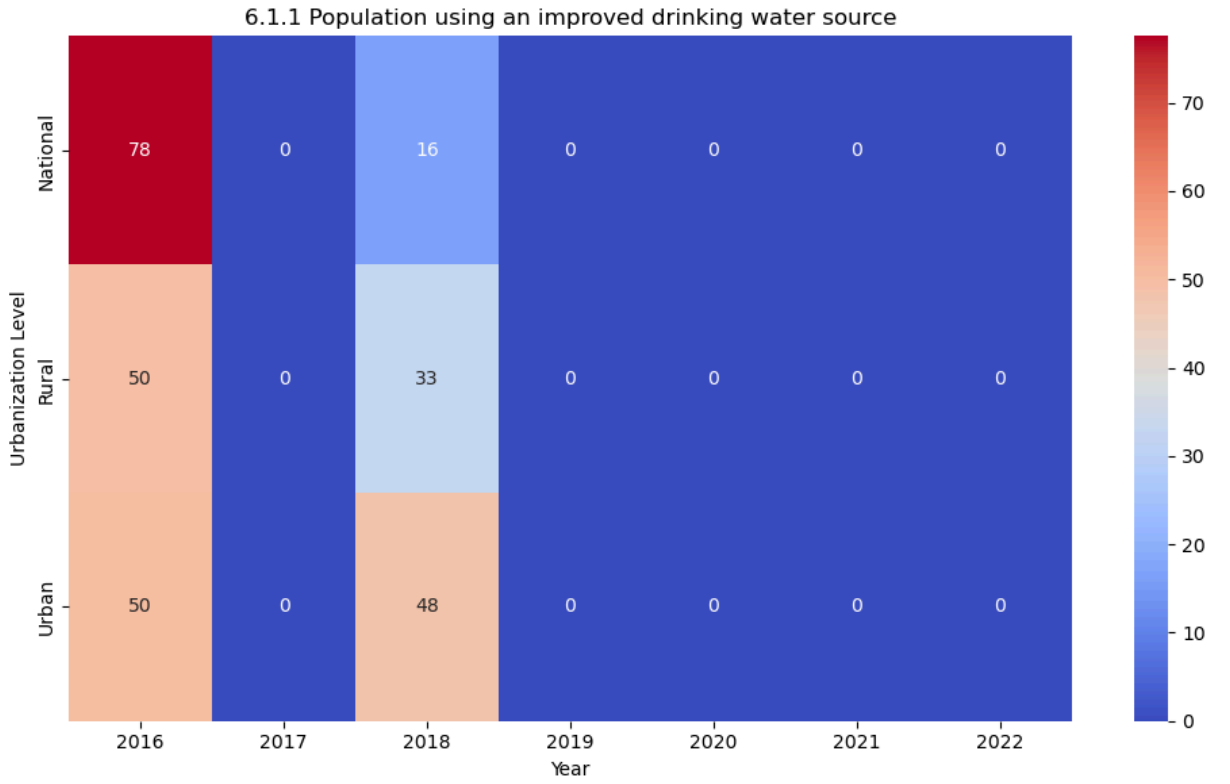
In [26]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv('Clean7.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('6.3.1 Proportion of wastewater safely treated')
plt.tight_layout()
plt.show()
```



In [27]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv('Clean1 (1).csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('Water and Sanitation > 6.1.1 Population using safely managed drinking water service
plt.tight_layout()
plt.show()
```

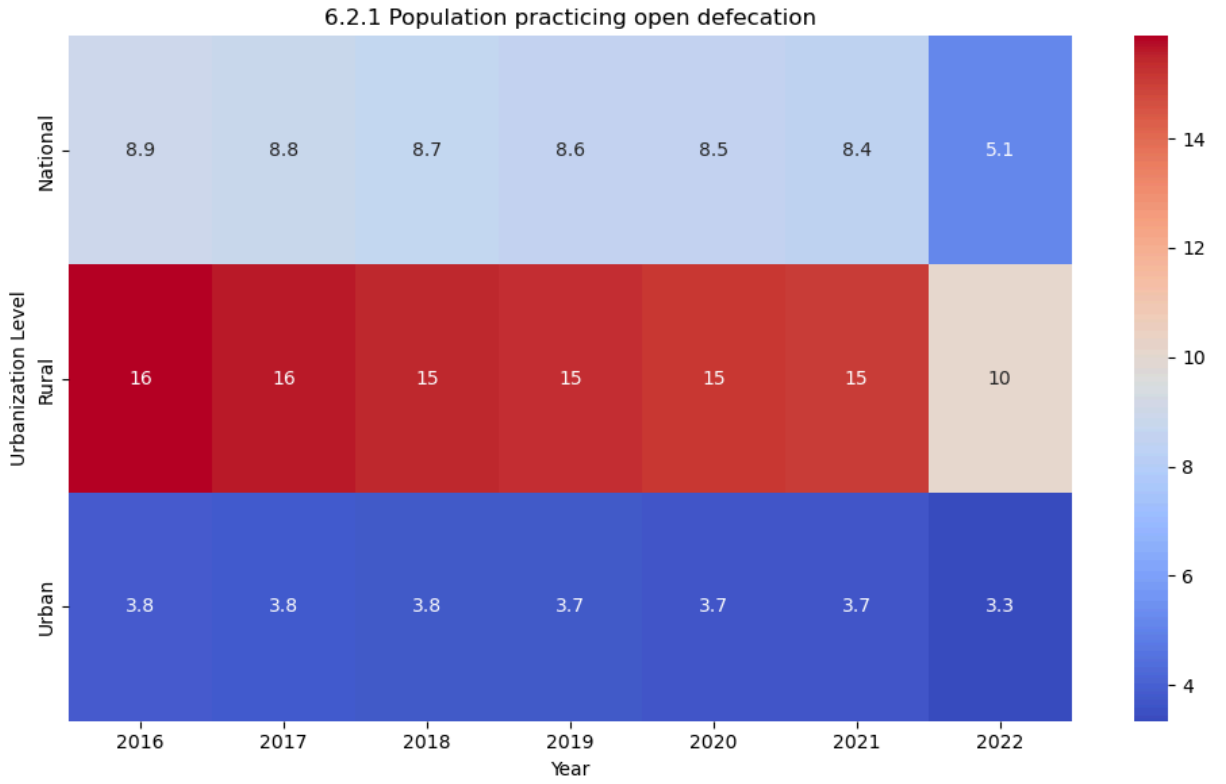


In [28]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv('Clean2.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('6.1.1 Population using an improved drinking water source')
plt.tight_layout()
plt.show()
```



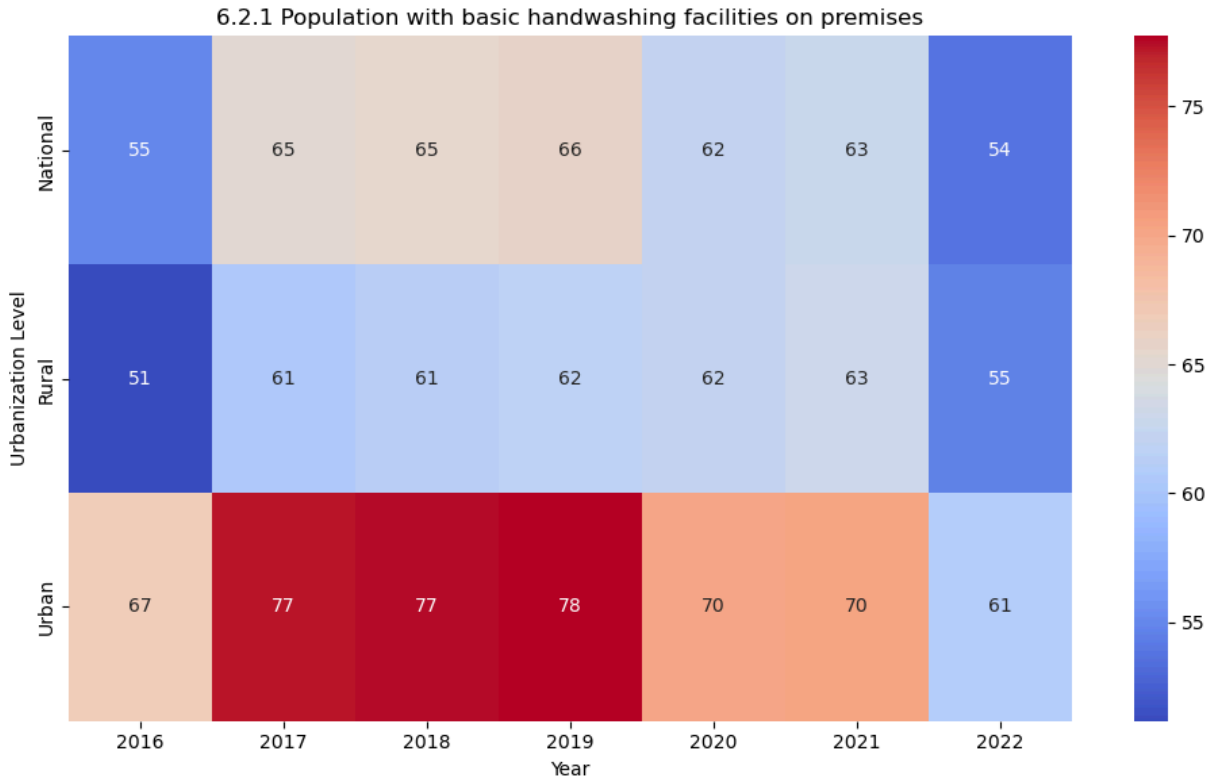
In [29]:

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

# Load the data
df = pd.read_csv('Clean3.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('6.2.1 Population practicing open defecation')
plt.tight_layout()
plt.show()
```



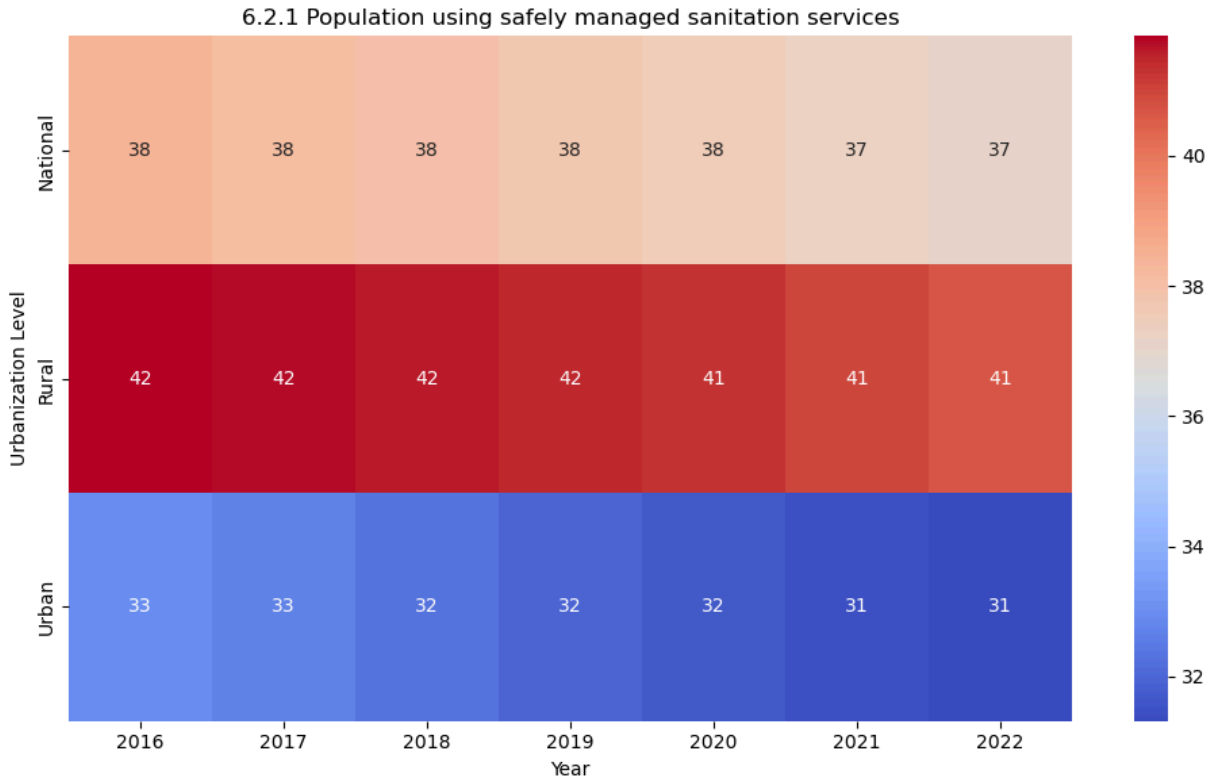
In [30]:

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

# Load the data
df = pd.read_csv('Clean4.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('6.2.1 Population with basic handwashing facilities on premises')
plt.tight_layout()
plt.show()
```

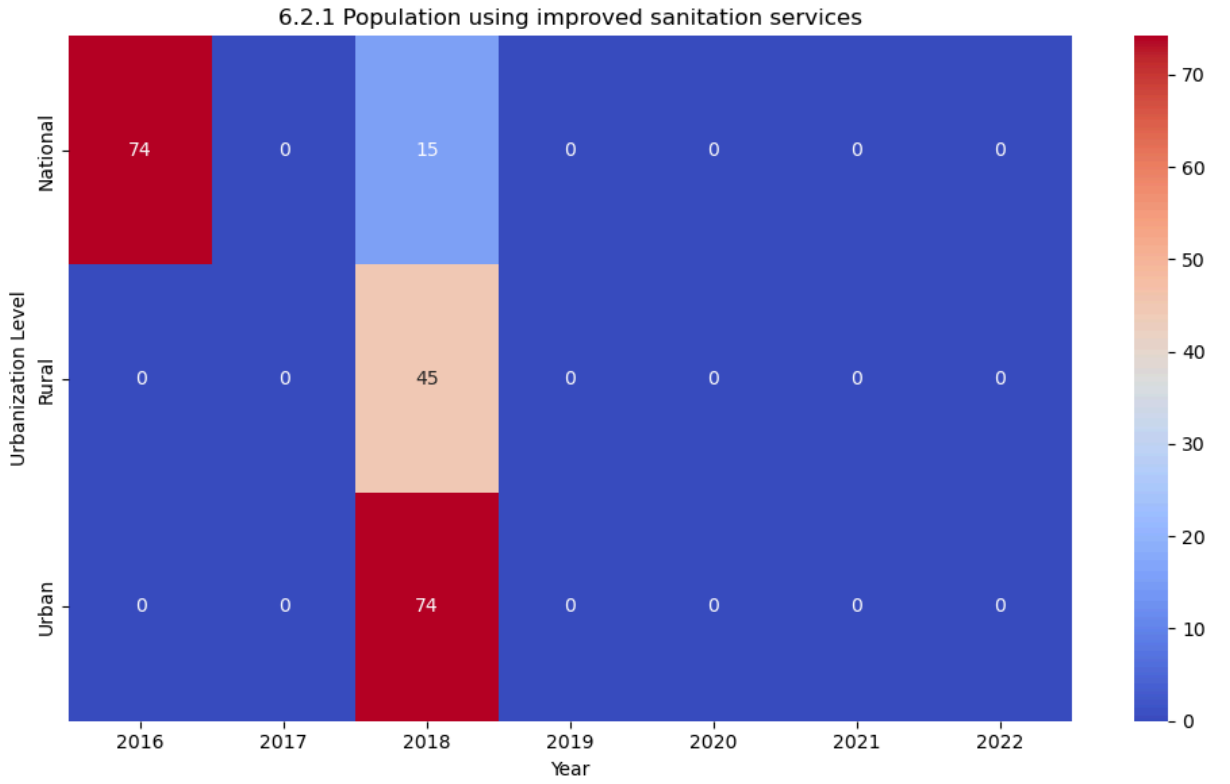


In [31]:

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

# Load the data
df = pd.read_csv('Clean5.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('6.2.1 Population using safely managed sanitation services')
plt.tight_layout()
plt.show()
```



In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv('Clean6.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('6.2.1 Population using improved sanitation services')
plt.tight_layout()
plt.show()
```

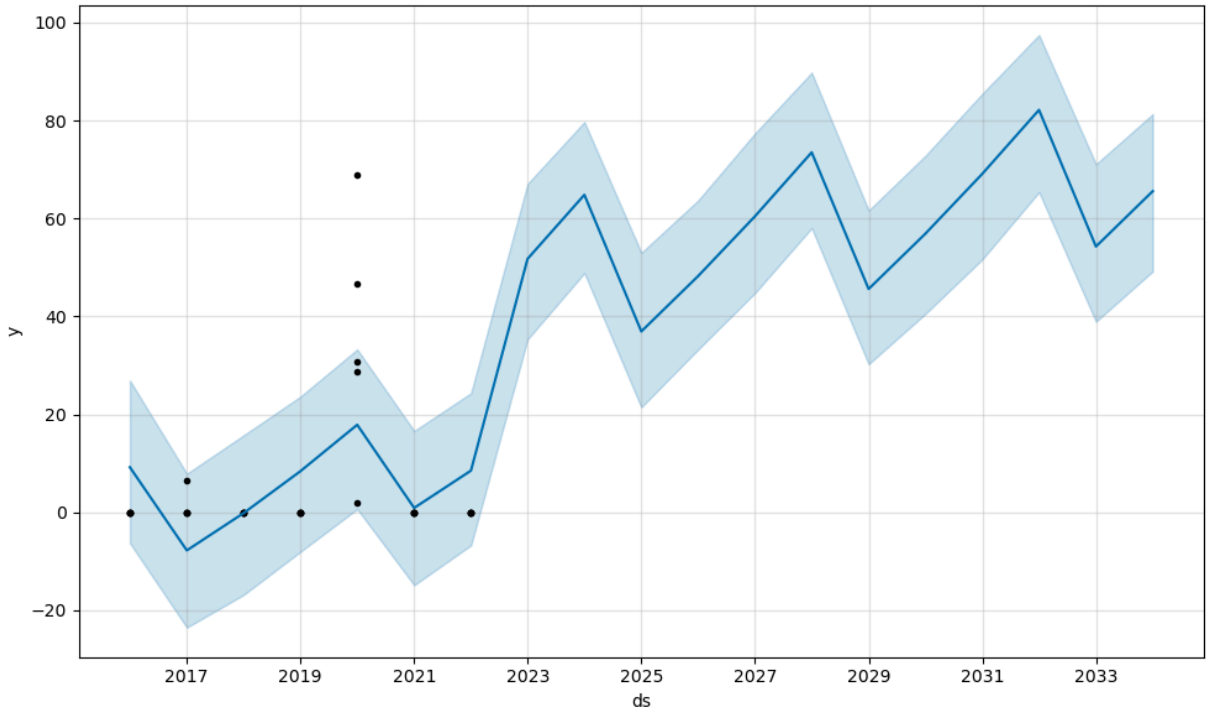


In [33]:
```
pip install prophet
```

```
Requirement already satisfied: prophet in c:\users\vaishnavi\anaconda3\lib\site-packages (1.
1.5)
Requirement already satisfied: pandas>=1.0.4 in c:\users\vaishnavi\anaconda3\lib\site-package
s (from prophet) (1.4.4)
Requirement already satisfied: holidays>=0.25 in c:\users\vaishnavi\anaconda3\lib\site-packag
es (from prophet) (0.53)
Requirement already satisfied: tqdm>=4.36.1 in c:\users\vaishnavi\anaconda3\lib\site-packages
(from prophet) (4.64.1)
Requirement already satisfied: cmdstanpy>=1.0.4 in c:\users\vaishnavi\anaconda3\lib\site-pack
ages (from prophet) (1.2.4)
Requirement already satisfied: importlib-resources in c:\users\vaishnavi\anaconda3\lib\site-p
ackages (from prophet) (6.4.0)
Requirement already satisfied: matplotlib>=2.0.0 in c:\users\vaishnavi\anaconda3\lib\site-pac
kages (from prophet) (3.5.2)
Requirement already satisfied: numpy>=1.15.4 in c:\users\vaishnavi\anaconda3\lib\site-package
s (from prophet) (1.21.5)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in c:\users\vaishnavi\anaconda3\lib\site-
packages (from cmdstanpy>=1.0.4->prophet) (0.5.1)
Requirement already satisfied: python-dateutil in c:\users\vaishnavi\anaconda3\lib\site-packa
ges (from holidays>=0.25->prophet) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\vaishnavi\anaconda3\lib\site-package
s (from matplotlib>=2.0.0->prophet) (9.2.0)
Requirement already satisfied: packaging>=20.0 in c:\users\vaishnavi\anaconda3\lib\site-packa
ges (from matplotlib>=2.0.0->prophet) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\vaishnavi\anaconda3\lib\site-pack
ages (from matplotlib>=2.0.0->prophet) (3.0.9)
Requirement already satisfied: cycler>=0.10 in c:\users\vaishnavi\anaconda3\lib\site-packages
(from matplotlib>=2.0.0->prophet) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\vaishnavi\anaconda3\lib\site-pac
kages (from matplotlib>=2.0.0->prophet) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\vaishnavi\anaconda3\lib\site-pac
kages (from matplotlib>=2.0.0->prophet) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\vaishnavi\anaconda3\lib\site-packages
(from pandas>=1.0.4->prophet) (2022.1)
Requirement already satisfied: colorama in c:\users\vaishnavi\anaconda3\lib\site-packages (fr
om tqdm>=4.36.1->prophet) (0.4.5)
Requirement already satisfied: zipp>=3.1.0 in c:\users\vaishnavi\anaconda3\lib\site-packages
(from importlib-resources->prophet) (3.8.0)
Requirement already satisfied: six>=1.5 in c:\users\vaishnavi\anaconda3\lib\site-packages (fr
om python-dateutil->holidays>=0.25->prophet) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

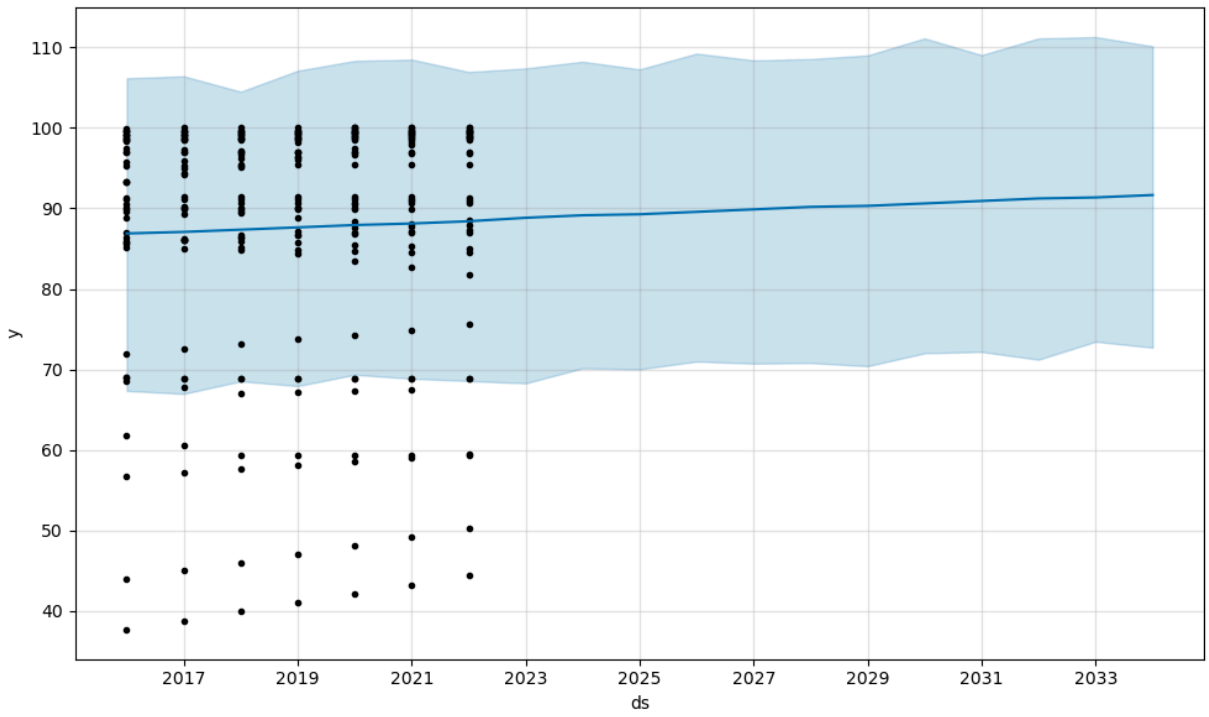
In [34]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean7.csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:47:41 - cmdstanpy - INFO - Chain [1] start processing
21:47:42 - cmdstanpy - INFO - Chain [1] done processing
```



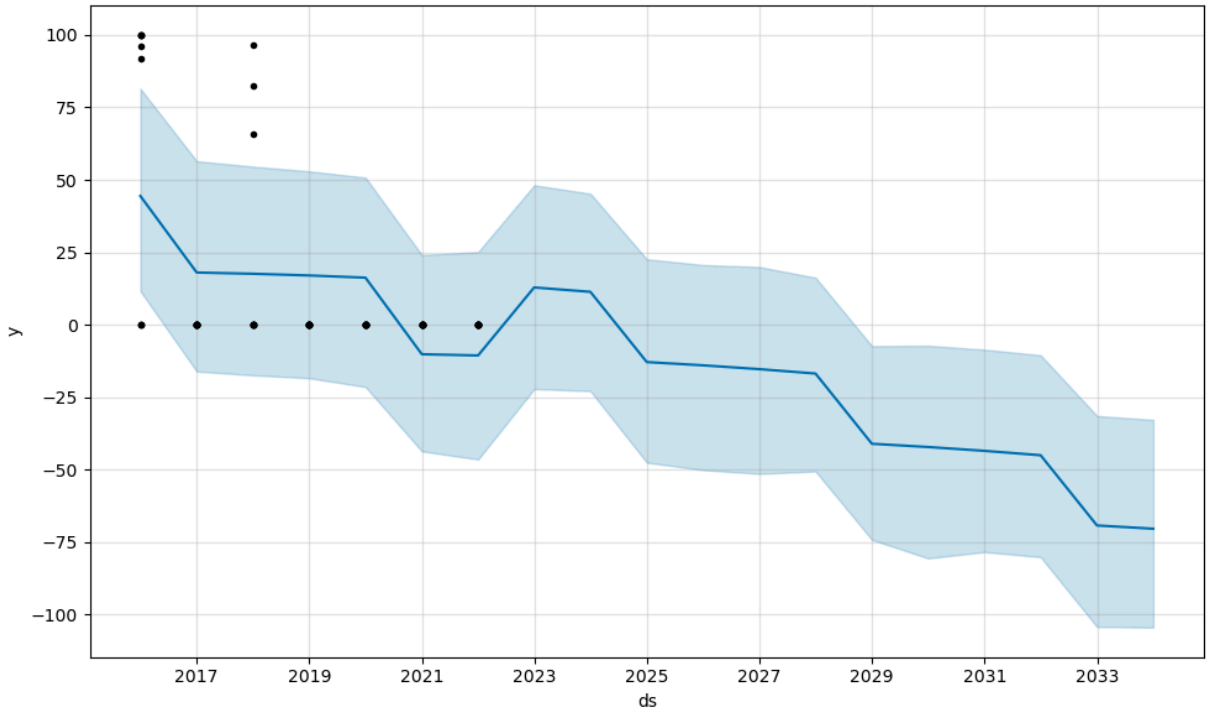
In [35]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean1 (1).csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:47:42 - cmdstanpy - INFO - Chain [1] start processing
21:47:42 - cmdstanpy - INFO - Chain [1] done processing
```



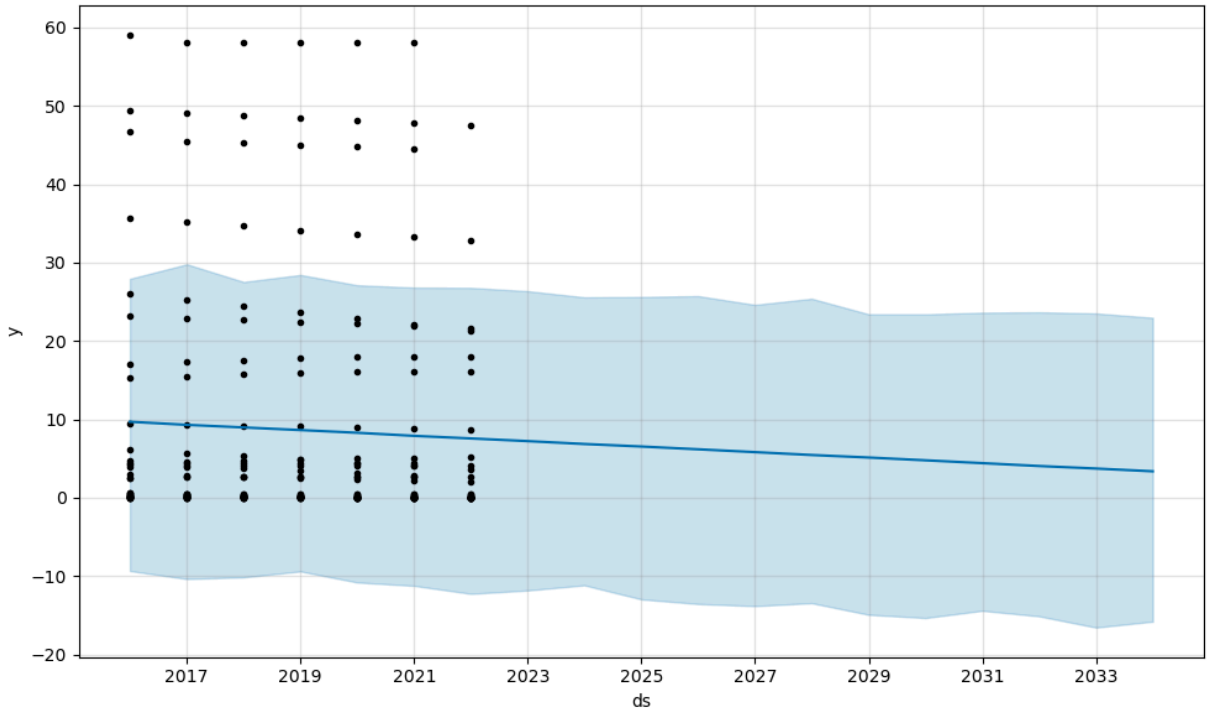
In [36]:
```
mport pandas as pd
rom prophet import Prophet
mport matplotlib.pyplot as plt

 Load the dataset
ile_name = 'Clean2.csv'
f = pd.read_csv(file_name)

 Clean and prepare data for Prophet
f['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                     'Rural' if 'Rural' in str(x) else
                                                     'National' if 'National' in str(x) else
                                                     None)
f = df.dropna(subset=['Urbanization'])
f_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '2020'
f_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

 Forecasting with Prophet
odel = Prophet()
odel.fit(df_prophet)
uture = model.make_future_dataframe(periods=12, freq='Y')
orecast = model.predict(future)

 Plot forecast
odel.plot(forecast)
lt.show()
```

```
21:47:43 - cmdstanpy - INFO - Chain [1] start processing
21:47:43 - cmdstanpy - INFO - Chain [1] done processing
```



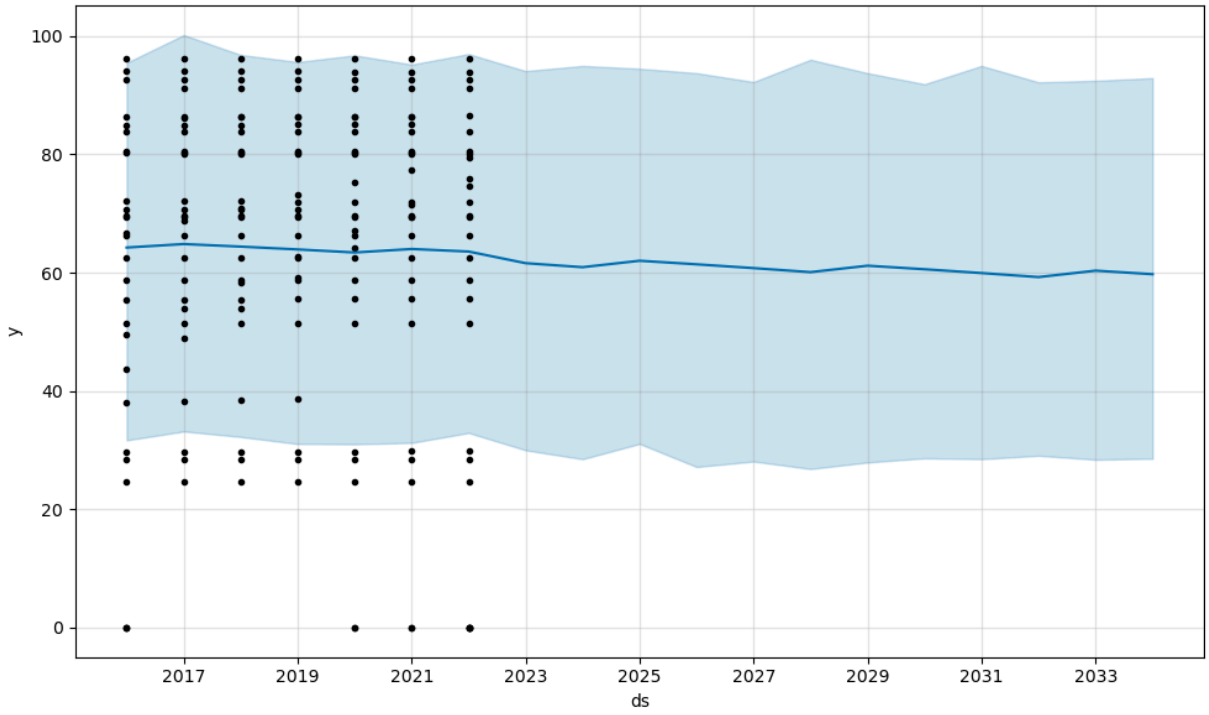
In [37]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean3.csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:47:44 - cmdstanpy - INFO - Chain [1] start processing
21:47:44 - cmdstanpy - INFO - Chain [1] done processing
```



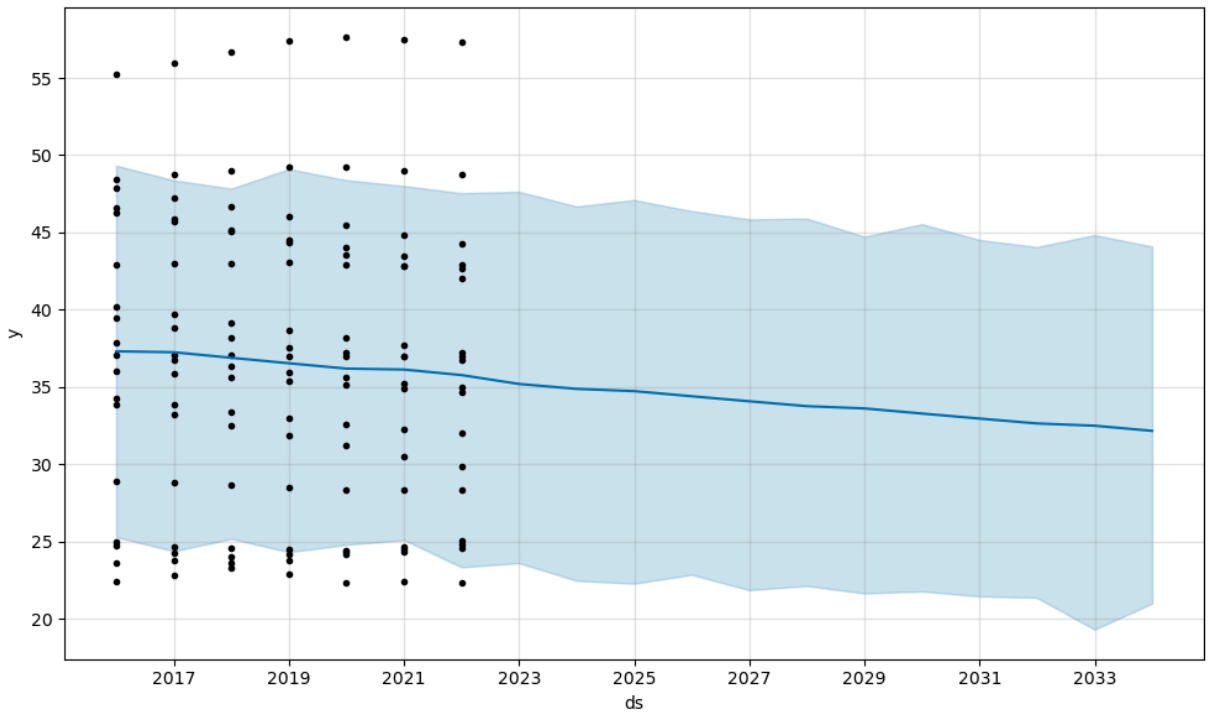
In [38]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean4.csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:47:44 - cmdstanpy - INFO - Chain [1] start processing
21:47:44 - cmdstanpy - INFO - Chain [1] done processing
```



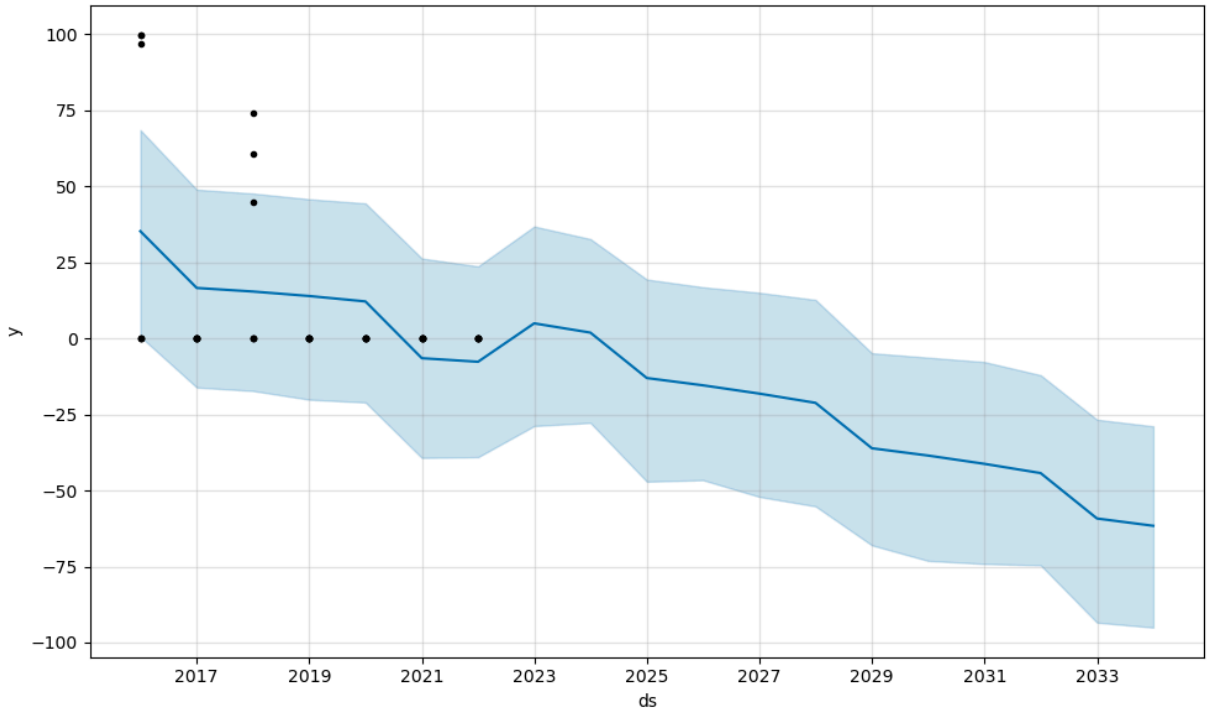
In [39]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean5.csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:47:45 - cmdstanpy - INFO - Chain [1] start processing
21:47:45 - cmdstanpy - INFO - Chain [1] done processing
```



In [40]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean6.csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:47:46 - cmdstanpy - INFO - Chain [1] start processing
21:47:46 - cmdstanpy - INFO - Chain [1] done processing
```



In [41]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean7.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.5
```

In [42]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean1 (1).csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.3333333333333333
```

In [43]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean2.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.5
```

In [44]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean3.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.2857142857142857
```

In [45]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean4.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.0
```

In [46]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean5.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.25
```

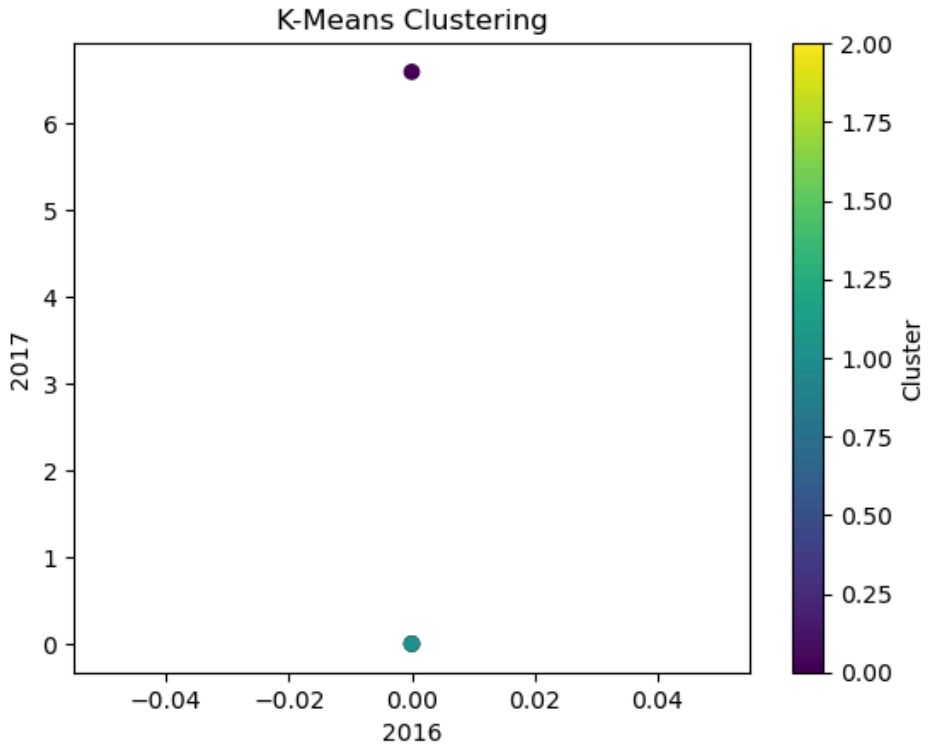
In [47]:

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean6.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.0
```

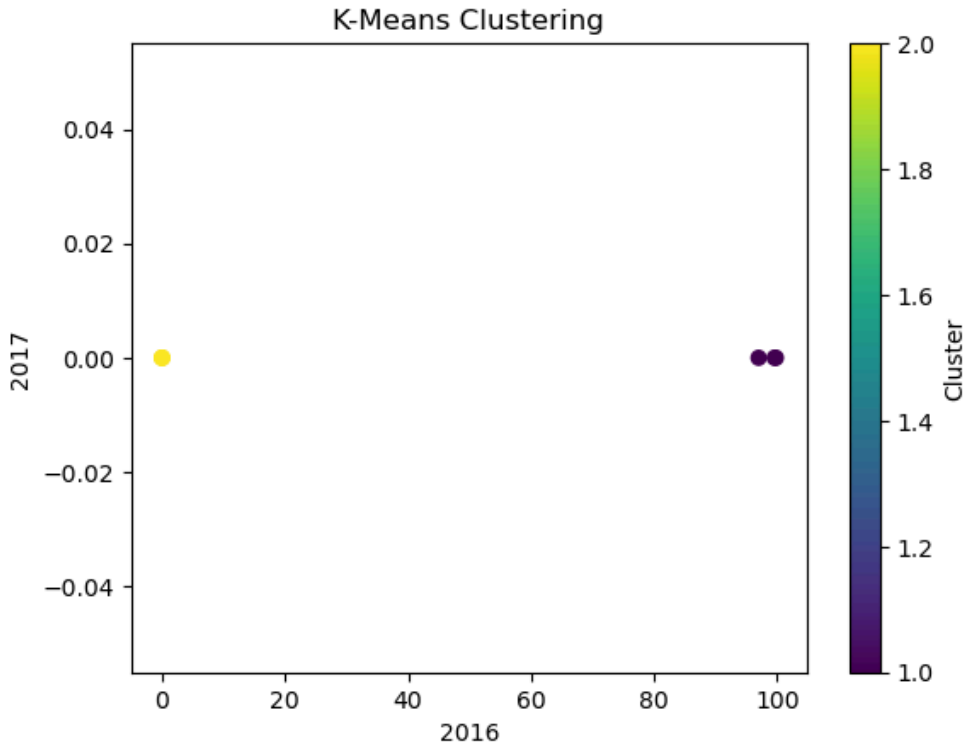
In [48]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean7.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



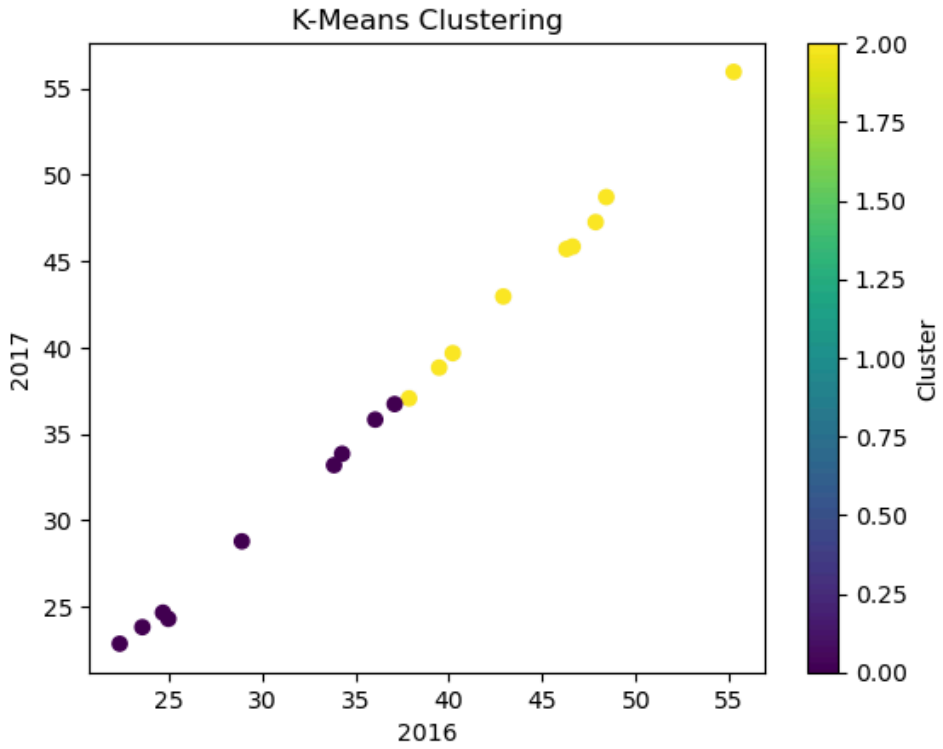
In [49]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean6.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



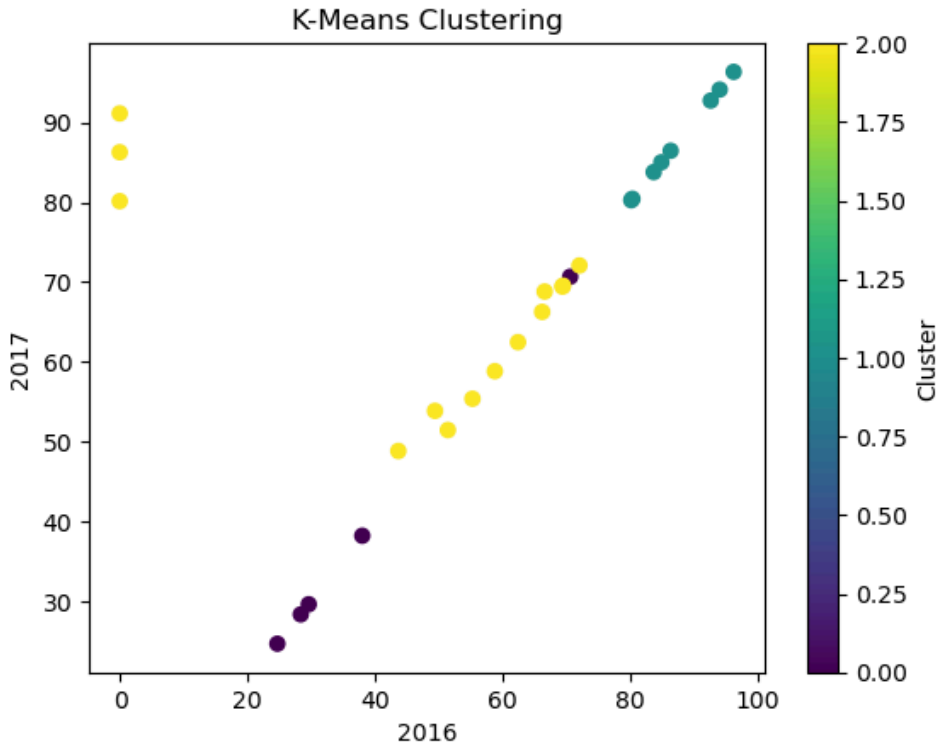
In [50]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean5.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



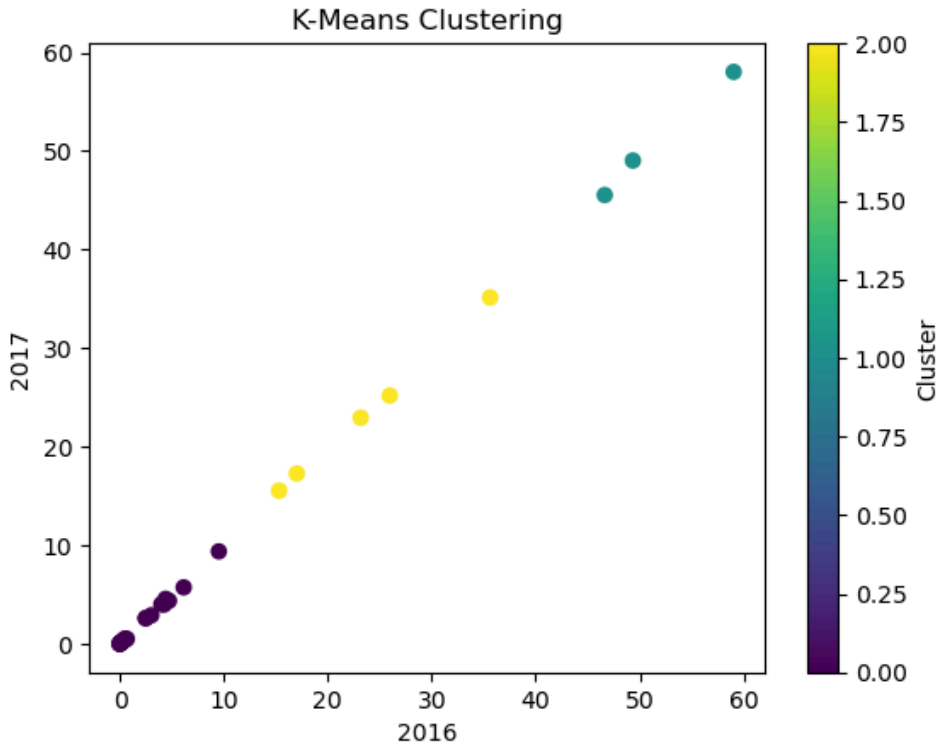
In [51]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean4.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



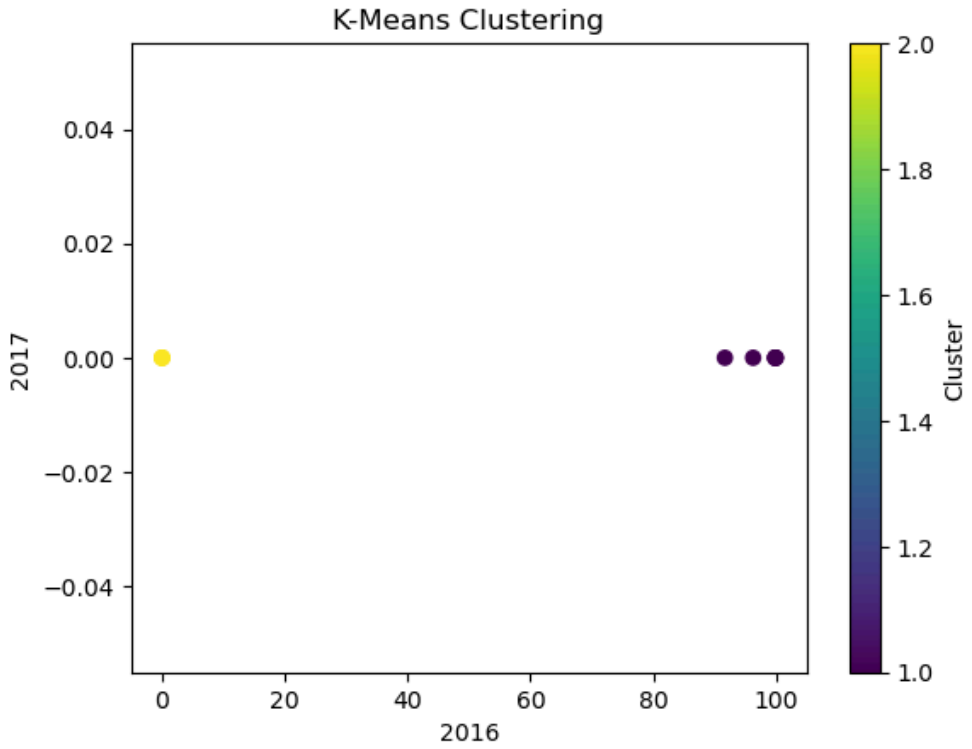
In [52]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean3.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



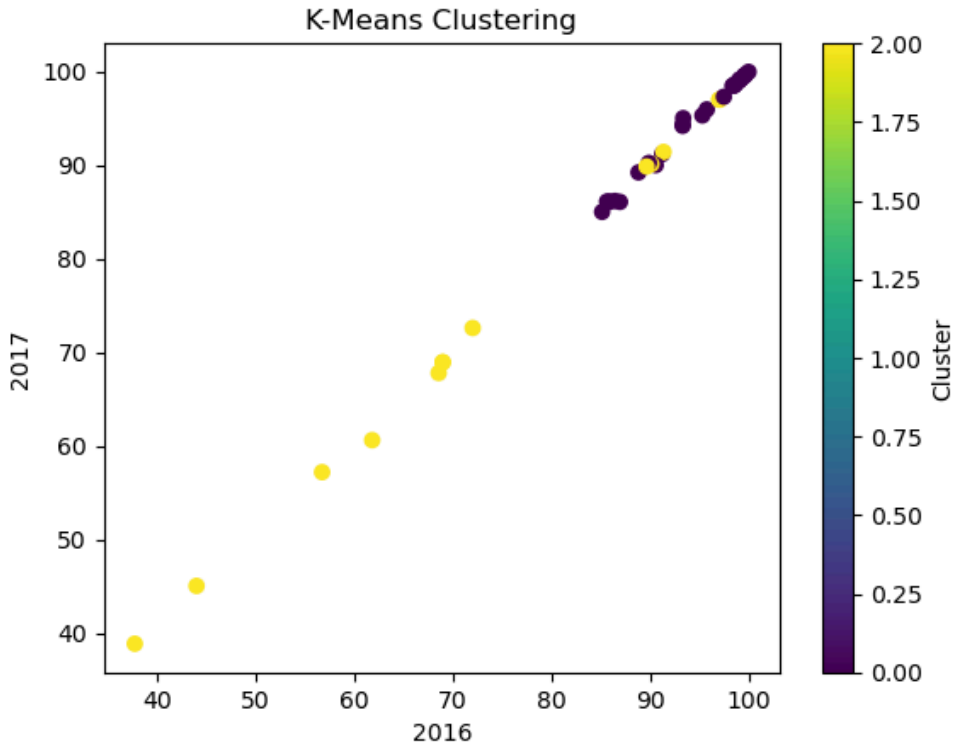
In [53]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean2.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



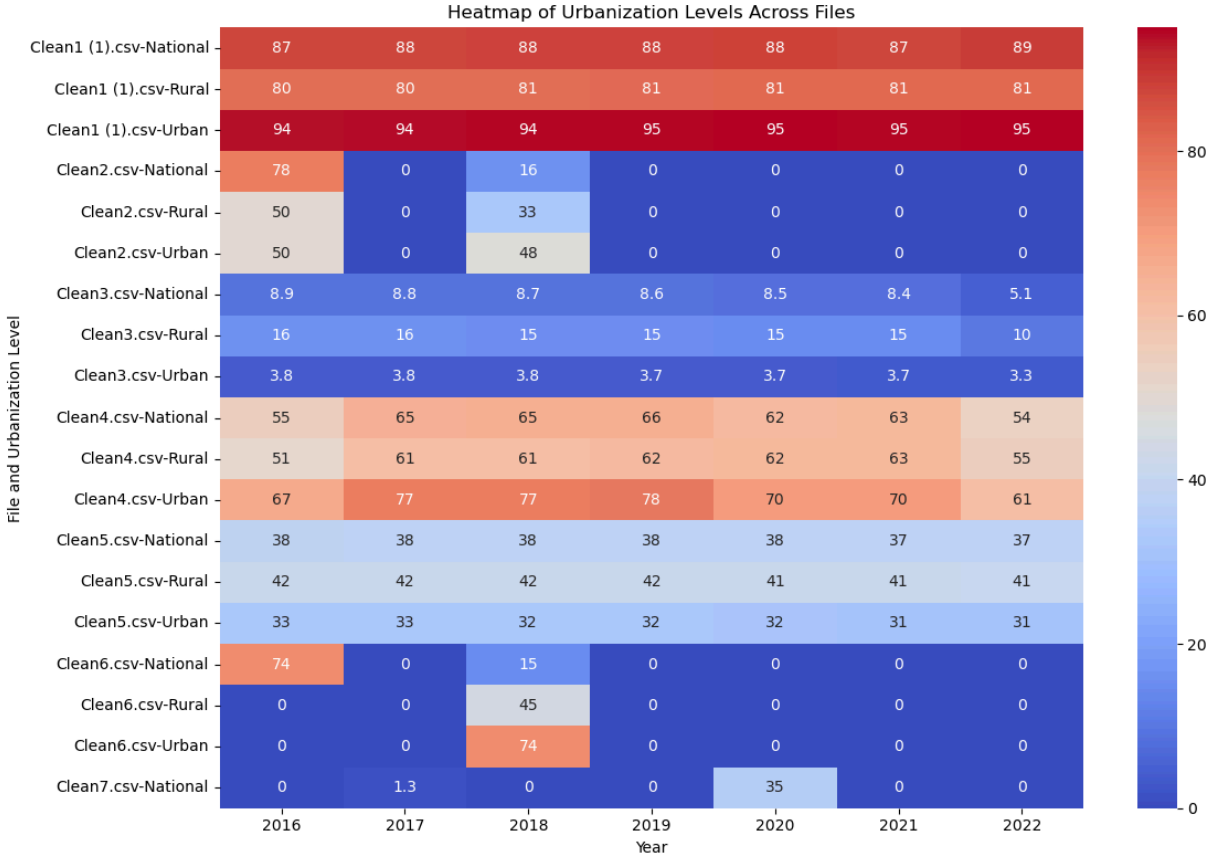
In [54]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean1 (1).csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



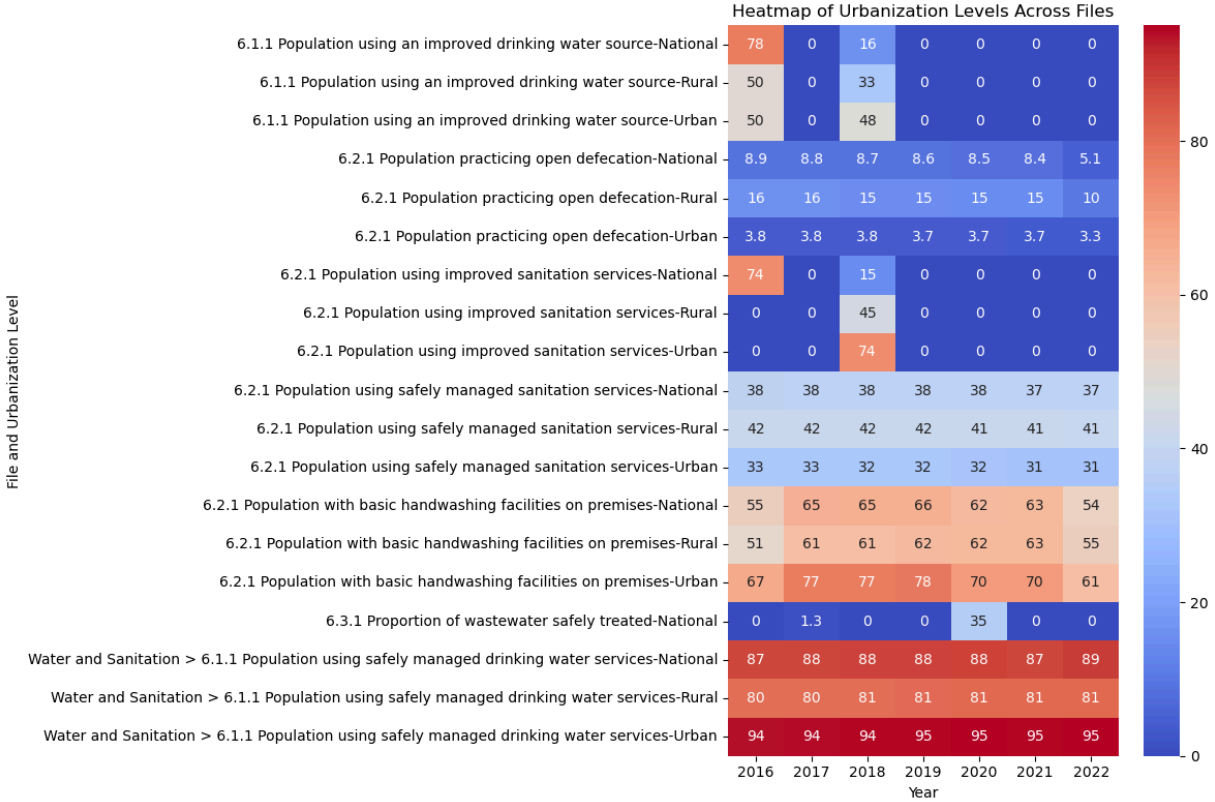
In [55]:

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

# List of filenames
files = ['Clean1 (1).csv', 'Clean2.csv', 'Clean3.csv', 'Clean4.csv', 'Clean5.csv', 'Clean6.csv

# List to hold DataFrames
dfs = []

# Load and preprocess each file
for file in files:
    df = pd.read_csv(file)

    # Clean the data
    df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                          'Rural' if 'Rural' in str(x) else
                                                          'National' if 'National' in str(x)
                                                          None)
    df = df.dropna(subset=['Urbanization'])
    df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    df_melted['File'] = file  # Add a column to identify the source file
    dfs.append(df_melted)

# Combine all DataFrames
combined_df = pd.concat(dfs, ignore_index=True)

# Pivot the DataFrame for heatmap
df_heatmap = combined_df.pivot_table(index=['File', 'Urbanization'], columns='Year', values='P

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('File and Urbanization Level')
plt.title('Heatmap of Urbanization Levels Across Files')
plt.tight_layout()
plt.show()
```


Heatmap of Urbanization Levels Across Files
Clean1 (1).csv-National 87 88 88 88 88 87 89
Clean1 (1).csv-Rural 80 80 81 81 81 81 81
Clean1 (1).csv-Urban 94 94 94 95 95 95 95
Clean2.csv-National 78 0 16 0 0 0 0
Clean2.csv-Rural 50 0 33 0 0 0 0
Clean2.csv-Urban 50 0 48 0 0 0 0
Clean3.csv-National 8.9 8.8 8.7 8.6 8.5 8.4 5.1
Clean3.csv-Rural 16 16 15 15 15 15 10
Clean3.csv-Urban 3.8 3.8 3.8 3.7 3.7 3.7 3.3
Clean4.csv-National 55 65 65 66 62 63 54
Clean4.csv-Rural 51 61 61 62 62 63 55
Clean4.csv-Urban 67 77 77 78 70 70 61
Clean5.csv-National 38 38 38 38 38 37 37
Clean5.csv-Rural 42 42 42 42 41 41 41
Clean5.csv-Urban 33 33 32 32 32 31 31
Clean6.csv-National 74 0 15 0 0 0 0
Clean6.csv-Rural 0 0 45 0 0 0 0
Clean6.csv-Urban 0 0 74 0 0 0 0
Clean7.csv-National 0 1.3 0 0 35 0 0
File and Urbanization Level
2016 2017 2018 2019 2020 2021 2022
Year
80 60 40 20 0

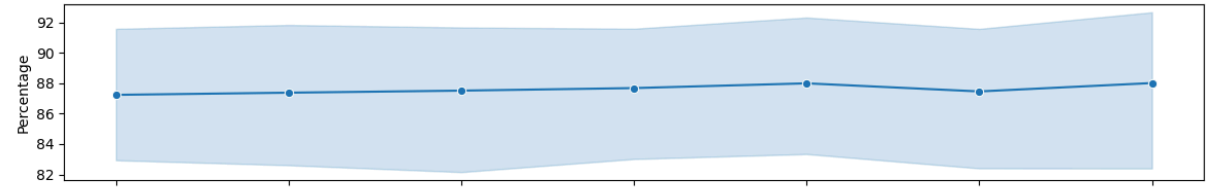
In [56]:

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

# List of filenames and their custom labels
files = ['Clean1 (1).csv', 'Clean2.csv', 'Clean3.csv', 'Clean4.csv', 'Clean5.csv', 'Clean6.csv
labels = ['Water and Sanitation > 6.1.1 Population using safely managed drinking water service

# Create a dictionary to map filenames to custom labels
file_label_map = dict(zip(files, labels))

# List to hold DataFrames
dfs = []

# Load and preprocess each file
for file in files:
    df = pd.read_csv(file)

    # Clean the data
    df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                          'Rural' if 'Rural' in str(x) else
                                                          'National' if 'National' in str(x)
                                                          None)
    df = df.dropna(subset=['Urbanization'])
    df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019',
    df_melted['File'] = file  # Add a column to identify the source file
    dfs.append(df_melted)

# Combine all DataFrames
combined_df = pd.concat(dfs, ignore_index=True)

# Replace filenames with custom labels
combined_df['File'] = combined_df['File'].map(file_label_map)

# Pivot the DataFrame for heatmap
df_heatmap = combined_df.pivot_table(index=['File', 'Urbanization'], columns='Year', values='P

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('File and Urbanization Level')
plt.title('Heatmap of Urbanization Levels Across Files')
plt.tight_layout()
plt.show()
```

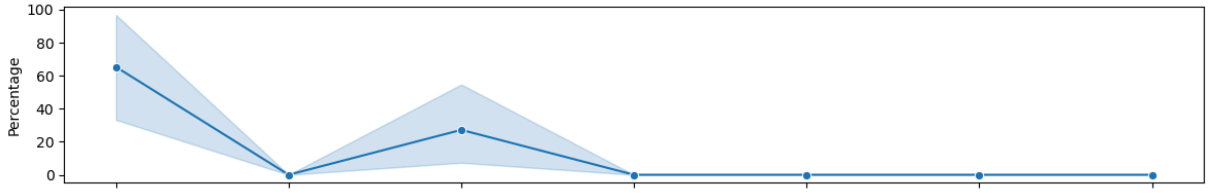

Heatmap of Urbanization Levels Across Files
File and Urbanization Level
6.1.1 Population using an improved drinking water source-National
6.1.1 Population using an improved drinking water source-Rural
6.1.1 Population using an improved drinking water source-Urban
6.2.1 Population practicing open defecation-National
6.2.1 Population practicing open defecation-Rural
6.2.1 Population practicing open defecation-Urban
6.2.1 Population using improved sanitation services-National
6.2.1 Population using improved sanitation services-Rural
6.2.1 Population using improved sanitation services-Urban
6.2.1 Population using safely managed sanitation services-National
6.2.1 Population using safely managed sanitation services-Rural
6.2.1 Population using safely managed sanitation services-Urban
6.2.1 Population with basic handwashing facilities on premises-National
6.2.1 Population with basic handwashing facilities on premises-Rural
6.2.1 Population with basic handwashing facilities on premises-Urban
6.3.1 Proportion of wastewater safely treated-National
Water and Sanitation > 6.1.1 Population using safely managed drinking water services-National
Water and Sanitation > 6.1.1 Population using safely managed drinking water services-Rural
Water and Sanitation > 6.1.1 Population using safely managed drinking water services-Urban
78 0 16 0 0 0 0
50 0 33 0 0 0 0
50 0 48 0 0 0 0
8.9 8.8 8.7 8.6 8.5 8.4 5.1
16 16 15 15 15 15 10
3.8 3.8 3.8 3.7 3.7 3.7 3.3
74 0 15 0 0 0 0
0 0 45 0 0 0 0
0 0 74 0 0 0 0
38 38 38 38 38 37 37
42 42 42 42 41 41 41
33 33 32 32 32 31 31
55 65 65 66 62 63 54
51 61 61 62 62 63 55
67 77 77 78 70 70 61
0 1.3 0 0 35 0 0
87 88 88 88 88 87 89
80 80 81 81 81 81 81
94 94 94 95 95 95 95
2016 2017 2018 2019 2020 2021 2022
Year
0 20 40 60 80

In [57]:

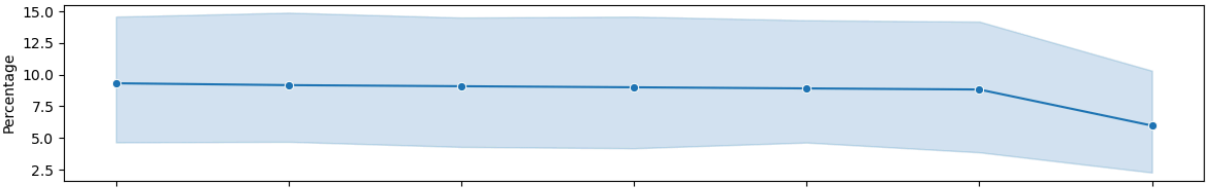
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# List of file names with mixed naming patterns
files = [
    'Clean1 (1).csv',
    'Clean2.csv',
    'Clean3.csv',
    'Clean4.csv',
    'Clean5.csv',
    'Clean6.csv',
    'Clean7.csv'
]

# Corresponding labels for each file
labels = [
    'Water and Sanitation > 6.1.1 Population using safely managed drinking water services',
    '6.1.1 Population using an improved drinking water source',
    '6.2.1 Population practicing open defecation',
    '6.2.1 Population with basic handwashing facilities on premises',
    '6.2.1 Population using safely managed sanitation services',
    '6.2.1 Population using improved sanitation services',
    '6.3.1 Proportion of wastewater safely treated'
]

# Create a list to store DataFrames
dfs = []

# Load each file into a DataFrame and append to the list
for file in files:
    df = pd.read_csv(file)

    # Example of assuming the file structure; adjust as necessary
    df = df.melt(id_vars=['Time'], value_vars=['2016', '2017', '2018', '2019', '2020', '2021',
                 var_name='Year', value_name='Percentage')
    df['File'] = file  # Add a column to identify the source file
    dfs.append(df)

# Combine all DataFrames into one
combined_df = pd.concat(dfs, ignore_index=True)

# Create a figure and subplots
fig, axes = plt.subplots(nrows=len(files), ncols=1, figsize=(12, 18), sharex=True)

# Plot each DataFrame in a subplot
for i, file in enumerate(files):
    ax = axes[i]
    df_filtered = combined_df[combined_df['File'] == file]
    sns.lineplot(data=df_filtered, x='Year', y='Percentage', ax=ax, marker='o')
    ax.set_xlabel('Year')
    ax.set_ylabel('Percentage')

# Add custom labels at the bottom of each subplot
for i, ax in enumerate(axes):
    ax.text(0.5, -0.15, labels[i], ha='center', va='top', transform=ax.transAxes, fontsize=12,

# Improve layout
plt.tight_layout(rect=[0, 0.05, 1, 0.95])  # Adjust the layout to make space for the labels

# Save or display the figure
plt.savefig('combined_line_graphs_with_labels.png')  # Save to a file
plt.show()  # Display the plot
```



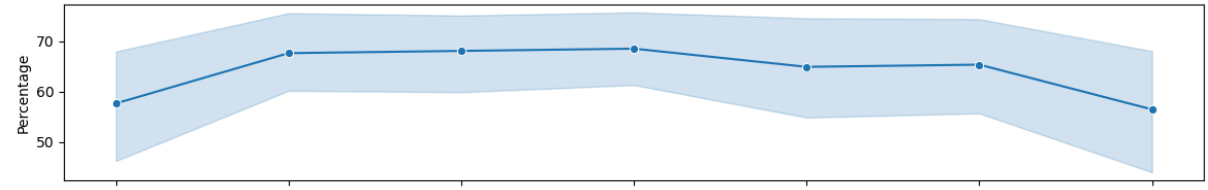
Water and Sanitation > 6.1.1 Population using safely managed drinking water services



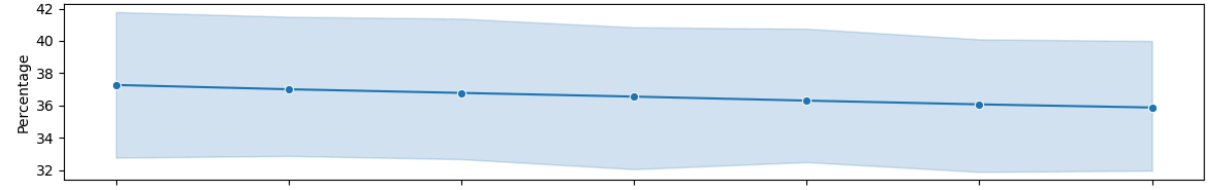
6.1.1 Population using an improved drinking water source



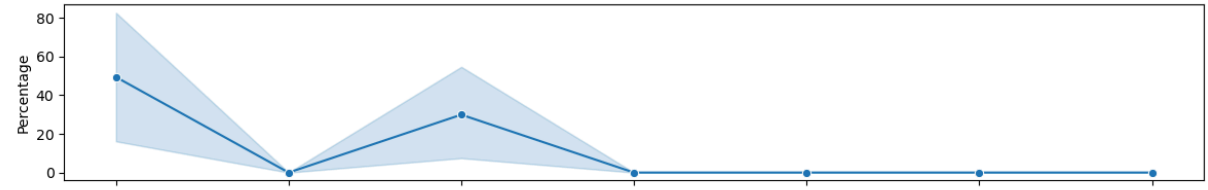
6.2.1 Population practicing open defecation



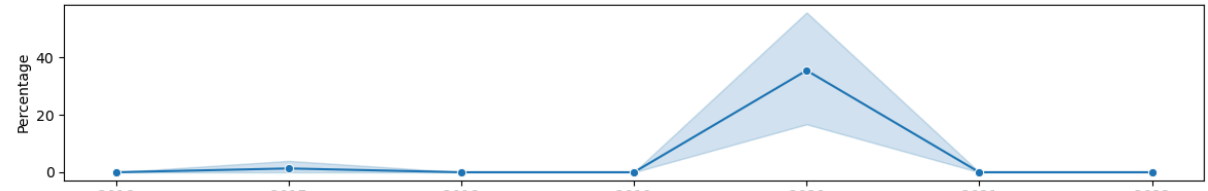
6.2.1 Population with basic handwashing facilities on premises



6.2.1 Population using safely managed sanitation services



6.2.1 Population using improved sanitation services

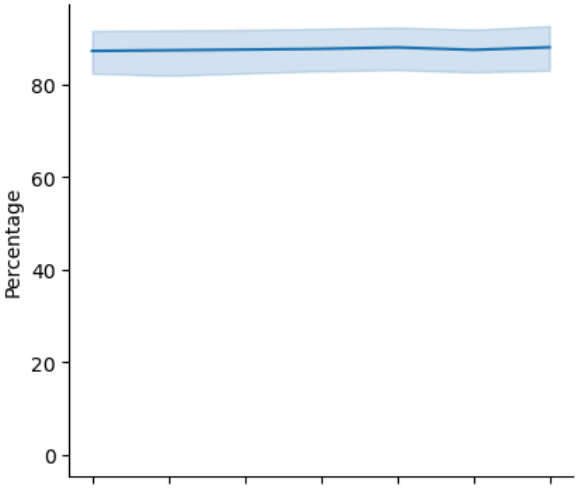


6.3.1 Proportion of wastewater safely treated

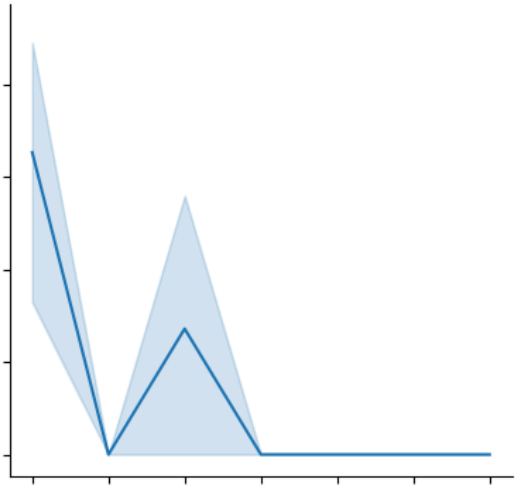
In [58]:

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

# List of file names with mixed naming patterns
files = [
    'Clean1 (1).csv',
    'Clean2.csv',
    'Clean3.csv',
    'Clean4.csv',
    'Clean5.csv',
    'Clean6.csv',
    'Clean7.csv'
]

# Labels for each file
labels = [
    'Water and Sanitation > 6.1.1 Population using safely managed drinking water services',
    '6.1.1 Population using an improved drinking water source',
    '6.2.1 Population practicing open defecation',
    '6.2.1 Population with basic handwashing facilities on premises',
    '6.2.1 Population using safely managed sanitation services',
    '6.2.1 Population using improved sanitation services',
    '6.3.1 Proportion of wastewater safely treated'
]

# Create a list to store DataFrames
dfs = []

# Load each file into a DataFrame and append to the list
for file in files:
    df = pd.read_csv(file)

    # Example of assuming the file structure; adjust as necessary
    df = df.melt(id_vars=['Time'], value_vars=['2016', '2017', '2018', '2019', '2020', '2021',
                 var_name='Year', value_name='Percentage')
    df['File'] = file  # Add a column to identify the source file
    dfs.append(df)

# Combine all DataFrames into one
combined_df = pd.concat(dfs, ignore_index=True)

# Create a FacetGrid with custom titles
g = sns.FacetGrid(combined_df, col='File', col_wrap=2, height=4)
g.map(sns.lineplot, 'Year', 'Percentage')
g.add_legend()

# Set custom titles for each subplot
for ax, label in zip(g.axes.flat, labels):
    ax.set_title(label, fontsize=9)
    ax.set_xlabel('Year')
    ax.set_ylabel('Percentage')

# Add custom labels at the bottom of each subplot
for i, ax in enumerate(axes):
    ax.text(0.5, -0.15, labels[i], ha='center', va='top', transform=ax.transAxes, fontsize=12,

# Improve layout
plt.tight_layout(rect=[0, 0.05, 1, 0.95])  # Adjust the layout to make space for the labels


plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```

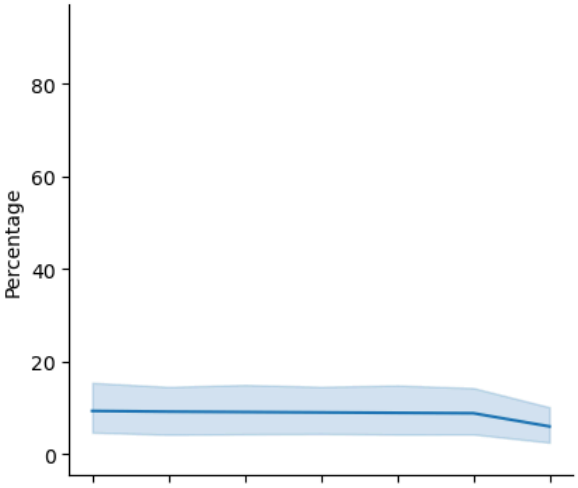
Water and Sanitation > 6.1.1 Population using safely managed drinking water services



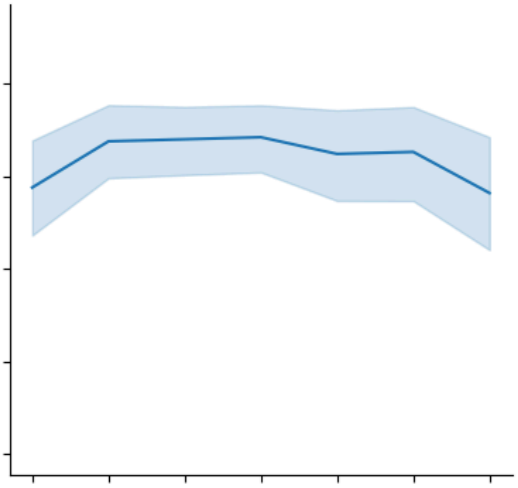
6.1.1 Population using an improved drinking water source



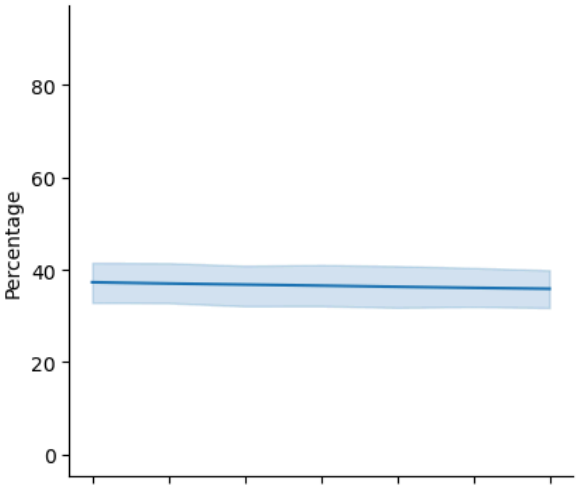
6.2.1 Population practicing open defecation



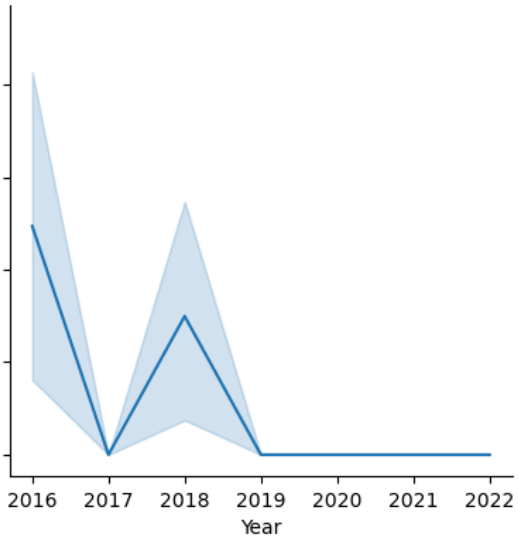
6.2.1 Population with basic handwashing facilities on premises



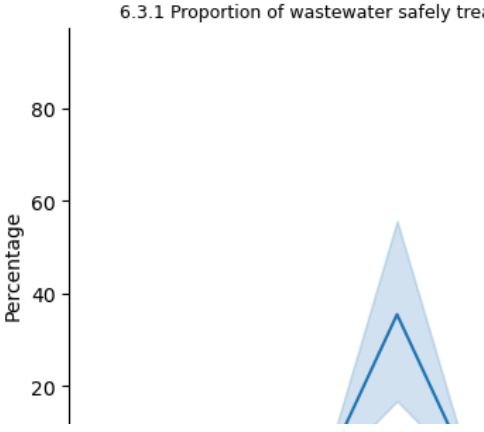
6.2.1 Population using safely managed sanitation services

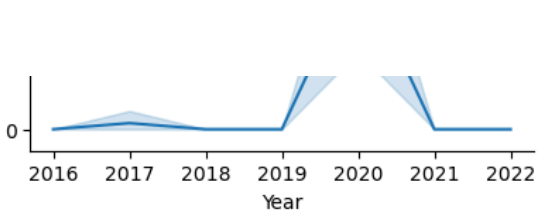


6.2.1 Population using improved sanitation services



6.3.1 Proportion of wastewater safely treated




0
2016
2017
2018
2019
2020
2021
2022
Year

In [59]:

```
import pandas as pd
from statsmodels.tsa.arima_model import ARIMA
import matplotlib.pyplot as plt

# Load and prepare the data
df = pd.read_csv('Clean7.csv')

# Assuming 'Year' is the time index and 'Percentage' is the target
df['Year'] = pd.to_datetime(df['Year'], format='%Y')
df.set_index('Year', inplace=True)

# Prepare data for ARIMA (example with 'Percentage' column)
data = df['Percentage'].fillna(0)  # Handle missing values

# Fit ARIMA model
model = ARIMA(data, order=(5, 1, 0))  # Example order; you may need to tune this
model_fit = model.fit(disp=0)

# Make predictions
forecast = model_fit.forecast(steps=5)  # Forecast the next 5 time points

# Plot results
plt.figure(figsize=(10, 6))
plt.plot(data.index, data, label='Observed')
plt.plot(pd.date_range(start=data.index[-1], periods=6, freq='Y')[1:], forecast[0], label='For
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Time Series Forecasting with ARIMA')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, toler
ance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_lo
c()

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_lo
c()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item
()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item
()

KeyError: 'Year'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13564\2036558835.py in <module>
      7
      8 # Assuming 'Year' is the time index and 'Percentage' is the target
----> 9 df['Year'] = pd.to_datetime(df['Year'], format='%Y')
     10 df.set_index('Year', inplace=True)
     11

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, toler
ance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error will raise

KeyError: 'Year'
```

In [60]:
```
print(df.columns)
```

```
Index(['Time', 'Unnamed: 1', '2016', '2017', '2018', '2019', '2020', '2021',
       '2022'],
      dtype='object')
```

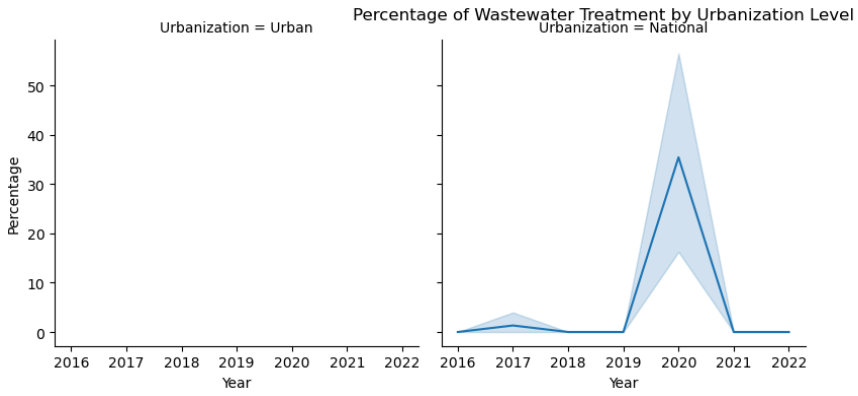
In [61]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Clean7.csv')

# Print column names to verify
print(df.columns)
```

```
Index(['Time', 'Unnamed: 1', '2016', '2017', '2018', '2019', '2020', '2021',
       '2022'],
      dtype='object')
```

In [62]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Clean7.csv')

# Print column names to verify
print(df.columns)

# Print the first few rows to understand the structure
print(df.head())
```

```
Index(['Time', 'Unnamed: 1', '2016', '2017', '2018', '2019', '2020', '2021',
       '2022'],
      dtype='object')
                                                Time    Unnamed: 1  2016  \
0           Pacific Island Countries and territories  Urbanization   NaN
1  Indicator: 6.3.1 Proportion of wastewater safe...           NaN   NaN
2                                          Micronesia           NaN   NaN
3                                        · Kiribati      National   0.0
4                                          Polynesia           NaN   NaN

   2017  2018  2019  2020  2021  2022
0   NaN   NaN   NaN   NaN   NaN   NaN
1   NaN   NaN   NaN   NaN   NaN   NaN
2   NaN   NaN   NaN   NaN   NaN   NaN
3   0.0   0.0   0.0  30.8   0.0   0.0
4   NaN   NaN   NaN   NaN   NaN   NaN
```

In [63]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a FacetGrid for plotting
g = sns.FacetGrid(df_melted, col='Urbanization', col_wrap=3, height=4)
g.map(sns.lineplot, 'Year', 'Percentage')
g.add_legend()

# Add labels and title
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Percentage of Wastewater Treatment by Urbanization Level')

# Show the plot
plt.show()
```

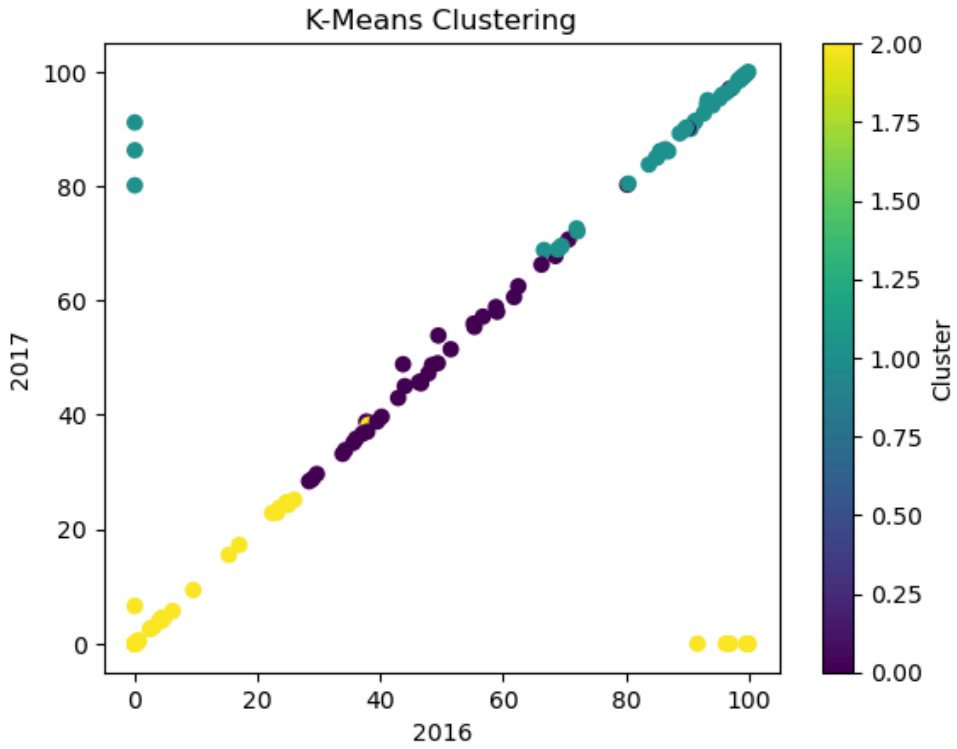


In [64]:
```
import pandas as pd

# Load the data
file_path = 'Clean-Water-and-Sanitation-Dataset-Sheet7.csv'
df = pd.read_csv(file_path)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Drop rows where the first column is NaN
df = df.dropna(subset=['Time'])

# Filter rows based on urbanization levels (e.g., 'Urban', 'Rural', 'National')
# We assume 'Unnamed: 1' column contains this information

# Create a new column to store the urbanization level
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)

# Drop rows where 'Urbanization' is None
df = df.dropna(subset=['Urbanization'])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Display the cleaned DataFrame with Urbanization levels
print("\nCleaned DataFrame with Urbanization Levels:")
print(df)

# Separate the data based on Urbanization levels
urban_df = df[df['Urbanization'] == 'Urban']
rural_df = df[df['Urbanization'] == 'Rural']
national_df = df[df['Urbanization'] == 'National']

# Optionally, save these DataFrames to separate CSV files
urban_df.to_csv('Urban_Data.csv', index=False)
rural_df.to_csv('Rural_Data.csv', index=False)
national_df.to_csv('National_Data.csv', index=False)

# Display the separated DataFrames
print("\nUrban Data:")
print(urban_df)
print("\nRural Data:")
print(rural_df)
print("\nNational Data:")
print(national_df)
```

```
136          · Kiribati   National         NaN   0.00   0.00   0.00   0.00
137    · American Samoa   National         NaN   0.00   0.00   0.00   0.00
138             · Samoa   National         NaN   0.00   6.60   0.00   0.00
139             · Tonga   National         NaN   0.00   0.00   0.00   0.00
140            · Tuvalu   National         NaN   0.00   0.00   0.00   0.00

      2020   2021   2022 Unnamed: 10 Urbanization
1    95.42  95.45  95.49         NaN     National
4    96.64  96.87  96.87         NaN     National
5    48.11  49.17  50.24         NaN     National
8    67.30  67.45   0.00         NaN     National
11   90.49  90.91  91.34         NaN     National
..     ...    ...    ...         ...          ...
136  30.80   0.00   0.00         NaN     National
137  69.00   0.00   0.00         NaN     National
138  46.70   0.00   0.00         NaN     National
139  28.60   0.00   0.00         NaN     National
140   2.00   0.00   0.00         NaN     National

[61 rows x 12 columns]
```

In [65]:

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load and preprocess data
df = pd.read_csv('Clean-Water-and-Sanitation-Dataset-Sheet7.csv')
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)

# Perform clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)
df['Cluster'] = clusters

# Plot clusters
plt.scatter(df['2016'], df['2017'], c=clusters, cmap='viridis')
plt.xlabel('2016')
plt.ylabel('2017')
plt.title('K-Means Clustering')
plt.colorbar(label='Cluster')
plt.show()
```



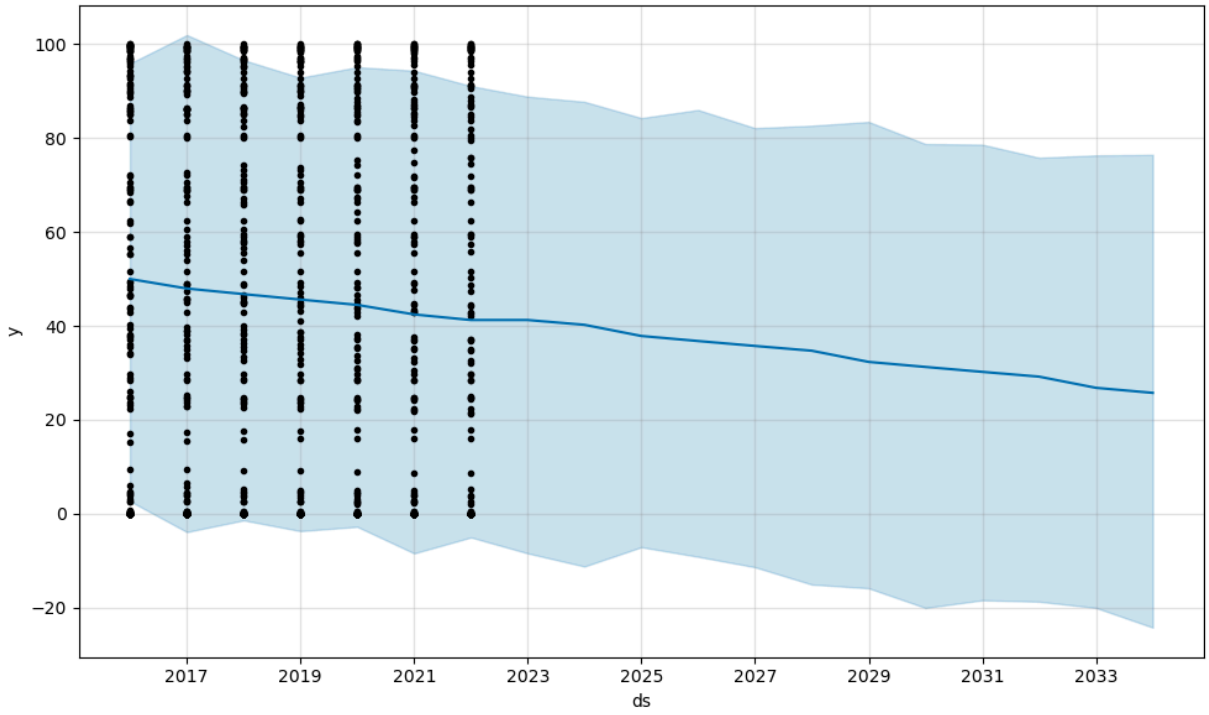
In [66]:

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and preprocess data
df = pd.read_csv('Clean-Water-and-Sanitation-Dataset-Sheet7.csv')
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
X = df[['2016', '2017', '2018', '2019', '2020', '2021', '2022']].fillna(0)
y = df['Urbanization']

# Train and evaluate model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

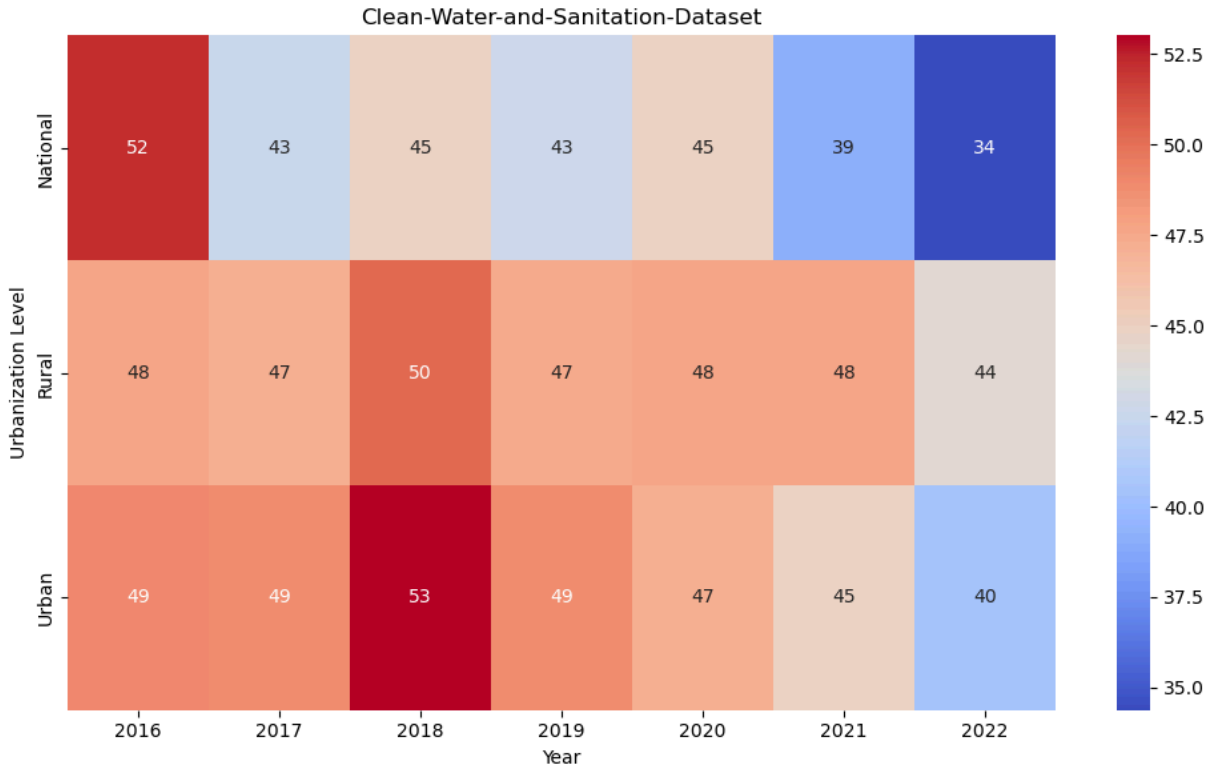
```
Accuracy: 0.3793103448275862
```

In [67]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Load the dataset
file_name = 'Clean-Water-and-Sanitation-Dataset-Sheet7.csv'
df = pd.read_csv(file_name)

# Clean and prepare data for Prophet
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202
df_prophet = df_melted.rename(columns={'Year': 'ds', 'Percentage': 'y'})

# Forecasting with Prophet
model = Prophet()
model.fit(df_prophet)
future = model.make_future_dataframe(periods=12, freq='Y')
forecast = model.predict(future)

# Plot forecast
model.plot(forecast)
plt.show()
```

```
21:48:22 - cmdstanpy - INFO - Chain [1] start processing
21:48:22 - cmdstanpy - INFO - Chain [1] done processing
```



In [68]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv('Clean-Water-and-Sanitation-Dataset-Sheet7.csv')

# Clean the data
df['Urbanization'] = df['Unnamed: 1'].apply(lambda x: 'Urban' if 'Urban' in str(x) else
                                                      'Rural' if 'Rural' in str(x) else
                                                      'National' if 'National' in str(x) els
                                                      None)
df = df.dropna(subset=['Urbanization'])
df.reset_index(drop=True, inplace=True)

# Melt the DataFrame for heatmap
df_melted = df.melt(id_vars=['Urbanization'], value_vars=['2016', '2017', '2018', '2019', '202

# Pivot table for heatmap
df_heatmap = df_melted.pivot_table(index='Urbanization', columns='Year', values='Percentage')

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_heatmap, annot=True, cmap='coolwarm', cbar=True)
plt.xlabel('Year')
plt.ylabel('Urbanization Level')
plt.title('Clean-Water-and-Sanitation-Dataset')
plt.tight_layout()
plt.show()
```



In [69]:
```
df.size
```

Out[69]: 1692

In [70]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141 entries, 0 to 140
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Time          141 non-null    object 
 1   Unnamed: 1    141 non-null    object 
 2   Unnamed: 2    0 non-null      float64
 3   2016          140 non-null    float64
 4   2017          140 non-null    float64
 5   2018          140 non-null    float64
 6   2019          140 non-null    float64
 7   2020          140 non-null    float64
 8   2021          140 non-null    float64
 9   2022          140 non-null    float64
 10  Unnamed: 10   0 non-null      object 
 11  Urbanization  141 non-null    object 
dtypes: float64(8), object(4)
memory usage: 13.3+ KB
```

In [71]:
```
df.describe()
```

Out[71]:

	Unnamed: 2	2016	2017	2018	2019	2020	2021	2022
count	0.0	140.000000	140.000000	140.000000	140.000000	140.000000	140.000000	140.000000
mean	NaN	50.033143	45.661857	48.719929	45.775000	46.381786	43.119857	38.824357
std	NaN	38.599952	38.163475	37.495439	38.391175	38.175159	38.741722	39.101616
min	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	NaN	4.202500	0.500000	3.947500	0.422500	2.315000	0.097500	0.000000
50%	NaN	48.885000	45.275000	48.855000	44.430000	44.395000	37.350000	29.120000
75%	NaN	90.220000	86.110000	86.350000	86.387500	86.425000	84.707500	80.205000
max	NaN	100.000000	99.970000	99.970000	99.970000	100.000000	100.000000	100.000000

In [72]:
```
df.nunique
```

Out[72]:
```
<bound method DataFrame.nunique of                                                  Time    Unnamed:
1  Unnamed: 2  \
0    Pacific Island Countries and territories  Urbanization         NaN
1                                     · Fiji      National         NaN
2                                     · Fiji         Urban         NaN
3                                     · Fiji         Rural         NaN
4                              · New Caledonia      National         NaN
..                                         ...          ...         ...
136                              · Kiribati      National         NaN
137                          · American Samoa      National         NaN
138                                   · Samoa      National         NaN
139                                   · Tonga      National         NaN
140                                  · Tuvalu      National         NaN

      2016   2017   2018   2019   2020   2021   2022 Unnamed: 10 Urbanization
0      NaN    NaN    NaN    NaN    NaN    NaN    NaN         NaN        Urban
1    95.28  95.31  95.35  95.38  95.42  95.45  95.49         NaN     National
2    98.58  98.60  98.61  98.62  98.64  98.65  98.66         NaN        Urban
3    91.20  91.18  91.15  91.13  91.11  91.08  91.06         NaN        Rural
4    95.73  95.96  96.18  96.41  96.64  96.87  96.87         NaN     National
..     ...    ...    ...    ...    ...    ...    ...         ...          ...
136   0.00   0.00   0.00   0.00  30.80   0.00   0.00         NaN     National
137   0.00   0.00   0.00   0.00  69.00   0.00   0.00         NaN     National
138   0.00   6.60   0.00   0.00  46.70   0.00   0.00         NaN     National
139   0.00   0.00   0.00   0.00  28.60   0.00   0.00         NaN     National
140   0.00   0.00   0.00   0.00   2.00   0.00   0.00         NaN     National

[141 rows x 12 columns]>
```

In [73]:
```
df.memory_usage( )
```

Out[73]:
```
Index           128
Time           1128
Unnamed: 1     1128
Unnamed: 2     1128
2016           1128
2017           1128
2018           1128
2019           1128
2020           1128
2021           1128
2022           1128
Unnamed: 10    1128
Urbanization   1128
dtype: int64
```

In [74]:
```
df.sample
```

Out[74]:
```
<bound method NDFrame.sample of                                         Time    Unnamed: 1
Unnamed: 2  \
0    Pacific Island Countries and territories  Urbanization         NaN
1                                      · Fiji      National         NaN
2                                      · Fiji         Urban         NaN
3                                      · Fiji         Rural         NaN
4                              · New Caledonia      National         NaN
..                                         ...           ...         ...
136                              · Kiribati      National         NaN
137                           · American Samoa      National         NaN
138                                    · Samoa      National         NaN
139                                    · Tonga      National         NaN
140                                   · Tuvalu      National         NaN

      2016   2017   2018   2019   2020   2021   2022 Unnamed: 10 Urbanization
0      NaN    NaN    NaN    NaN    NaN    NaN    NaN         NaN        Urban
1    95.28  95.31  95.35  95.38  95.42  95.45  95.49         NaN     National
2    98.58  98.60  98.61  98.62  98.64  98.65  98.66         NaN        Urban
3    91.20  91.18  91.15  91.13  91.11  91.08  91.06         NaN        Rural
4    95.73  95.96  96.18  96.41  96.64  96.87  96.87         NaN     National
..     ...    ...    ...    ...    ...    ...    ...         ...          ...
136   0.00   0.00   0.00   0.00  30.80   0.00   0.00         NaN     National
137   0.00   0.00   0.00   0.00  69.00   0.00   0.00         NaN     National
138   0.00   6.60   0.00   0.00  46.70   0.00   0.00         NaN     National
139   0.00   0.00   0.00   0.00  28.60   0.00   0.00         NaN     National
140   0.00   0.00   0.00   0.00   2.00   0.00   0.00         NaN     National

[141 rows x 12 columns]>
```

In [76]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# List of file names with mixed naming patterns
files = [
    'Clean1 (1).csv',
    'Clean2.csv',
    'Clean3.csv',
    'Clean4.csv',
    'Clean5.csv',
    'Clean6.csv',
    'Clean7.csv'
]
labels = ['Water and Sanitation > 6.1.1 Population using safely managed drinking water service
          '6.1.1 Population using an improved drinking water source',
          '6.2.1 Population practicing open defecation',
          '6.2.1 Population with basic handwashing facilities on premises',
          '6.2.1 Population using safely managed sanitation services',
          '6.2.1 Population using improved sanitation services',
          '6.3.1 Proportion of wastewater safely treated']

# Create a list to store DataFrames
dfs = []

# Load each file into a DataFrame and append to the list
for file in files:
    df = pd.read_csv(file)
    df = df.melt(id_vars=['Time'], value_vars=['2016', '2017', '2018', '2019', '2020', '2021',
                 var_name='Year', value_name='Percentage')
    df['File'] = file  # Add a column to identify the source file
    dfs.append(df)

# Combine all DataFrames into one
combined_df = pd.concat(dfs, ignore_index=True)

# Create and save plots
for i, file in enumerate(files):
    df_filtered = combined_df[combined_df['File'] == file]
    plt.figure(figsize=(12, 6))
    sns.lineplot(data=df_filtered, x='Year', y='Percentage', marker='o')
    plt.title(labels[i])
    plt.xlabel('Year')
    plt.ylabel('Percentage')
    plt.savefig(f'graph_{i+1}.png')
    plt.close()
```

In [ ]:

In [ ]: