directuseofgeneration(industrial&commercial))

November 19, 2024

[46]:
```
# Import necessary libraries
import requests
import pandas as pd

# Prompt user for the API key
api_key = input("mDFlP9SB3qqMh9wyfuWvZpwzFZsElfh0I1YAoiKl")

# Define the API URL and include the API key
api_url = f"https://api.eia.gov/v2/electricity/state-electricity-profiles/
 ↪source-disposition/data/?
 ↪frequency=annual&data[0]=direct-use&facets[state][]=AK&facets[state][]=AL&facets[state][]=A

# Use the GET request to fetch the data
try:
    response = requests.get(api_url)

    # Check for a successful response
    if response.status_code == 200:
        print("Data fetched successfully!")

        # Parse the JSON response
        data = response.json()

        # Verify the structure of the JSON response
        if 'response' in data and 'data' in data['response']:
            # Extract the relevant data
            records = data['response']['data']

            # Convert the records to a DataFrame
            df = pd.DataFrame(records)

            # Display a preview of the data
            print("Preview of the data:")
            print(df.head())

            # Save the data to a CSV file
            df.to_csv('eia_electric_power_operational_data.csv', index=False)
```

```
            print("Data saved to 'eia_electric_power_operational_data.csv'")
        else:
            print("Error: Unexpected structure in the JSON response.")
    else:
        print(f"Failed to fetch data. HTTP Status Code: {response.status_code}")
        print("Error message:", response.text)
except requests.exceptions.RequestException as e:
    print(f"An error occurred: {e}")
```

```
mDFlP9SB3qqMh9wyfuWvZpwzFZsElfh0I1YAoiKl
mDFlP9SB3qqMh9wyfuWvZpwzFZsElfh0I1YAoiKl

Data fetched successfully!
Preview of the data:
  period state stateDescription direct-use direct-use-units
0  2023    AK           Alaska     257915    megawatthours
1  2023    AL          Alabama    5491093    megawatthours
2  2023    NE         Nebraska     419370    megawatthours
3  2023    ND     North Dakota     179455    megawatthours
4  2023    NC   North Carolina    1688675    megawatthours
Data saved to 'eia_electric_power_operational_data.csv'
```

[48]:
```
import os
os.listdir()
from IPython.display import FileLink

# Provide a link to download the CSV file
FileLink('eia_electric_power_operational_data.csv')
```

[48]:
```
/home/6d18b5fb-9902-4a13-ad68-
44cc764f347e/eia_electric_power_operational_data.csv
```

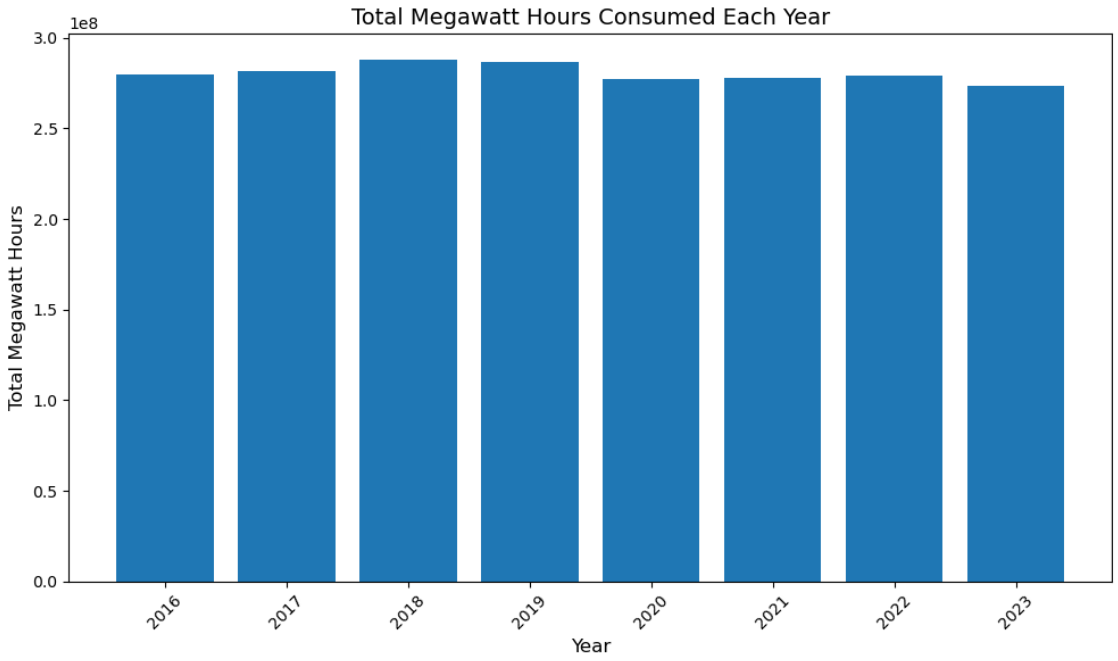
[52]:
```
# Convert 'direct-use' to numeric, in case it's stored as a string
df['direct-use'] = pd.to_numeric(df['direct-use'], errors='coerce')

# Group by 'period' (year) and calculate the total consumption for each year
total_consumption_per_year = df.groupby('period')['direct-use'].sum()

# Display the results
print("Total Megawatt Hours Consumed Each Year:")
print(total_consumption_per_year)

# Optional: Convert the result to a DataFrame for better display in Jupyter
total_consumption_df = total_consumption_per_year.reset_index()
total_consumption_df.columns = ['Year', 'Total Megawatt Hours']

# Display as a grid in Jupyter
display(total_consumption_df)
```

```
Total Megawatt Hours Consumed Each Year:
period
2016    279673398
2017    281918778
2018    287807462
2019    286540676
2020    277405080
2021    277830136
2022    279451402
2023    273836310
Name: direct-use, dtype: int64
```

	Year	Total Megawatt Hours
0	2016	279673398
1	2017	281918778
2	2018	287807462
3	2019	286540676
4	2020	277405080
5	2021	277830136
6	2022	279451402
7	2023	273836310

[54]:
```python
import matplotlib.pyplot as plt

# Create the bar graph
plt.figure(figsize=(10, 6))
plt.bar(total_consumption_df['Year'], total_consumption_df['Total Megawatt␣
↪Hours'])

# Add titles and labels
plt.title('Total Megawatt Hours Consumed Each Year', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Total Megawatt Hours', fontsize=12)

# Rotate x-axis labels for better readability
plt.xticks(total_consumption_df['Year'], rotation=45)

# Display the bar graph
plt.tight_layout()
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
```



[56]:
```
# Group by 'state' and calculate the average energy consumption
average_consumption_per_state = df.groupby('state')['direct-use'].mean()

# Convert the result to a DataFrame for better display
average_consumption_df = average_consumption_per_state.reset_index()
average_consumption_df.columns = ['State', 'Average Megawatt Hours']

# Display the results
print("Average Megawatt Hours Consumed Per State:")
display(average_consumption_df)
```

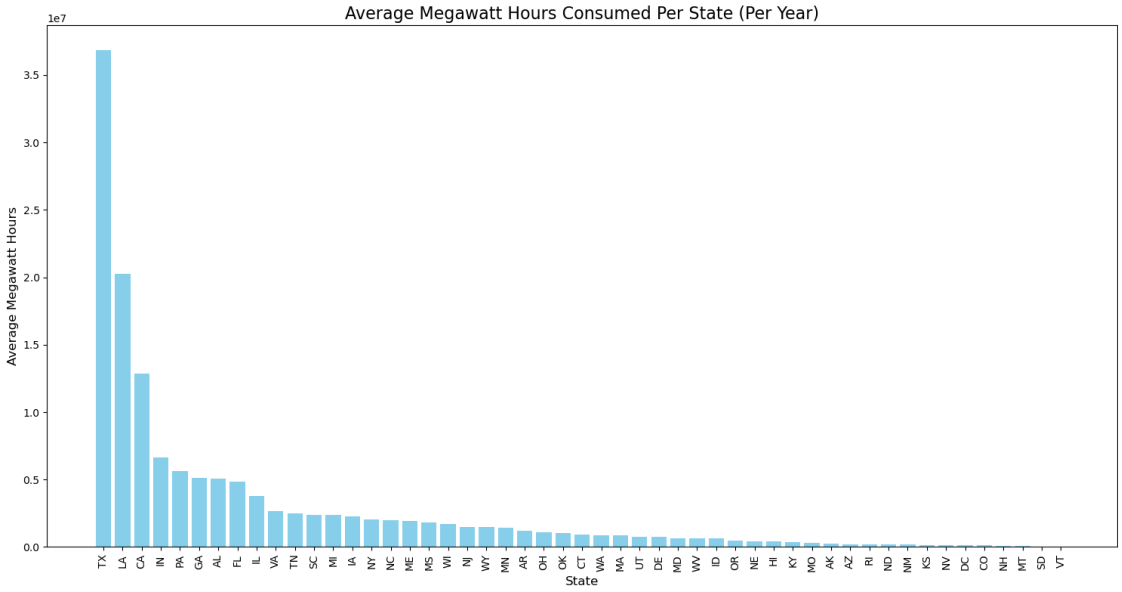
```
Average Megawatt Hours Consumed Per State:
```

	State	Average Megawatt Hours
0	AK	2.496556e+05
1	AL	5.069289e+06
2	AR	1.229580e+06
3	AZ	2.109465e+05
4	CA	1.283285e+07
5	CO	1.186476e+05
6	CT	9.203232e+05
7	DC	1.352262e+05
8	DE	7.536828e+05
9	FL	4.859366e+06
10	GA	5.119340e+06
11	HI	4.287955e+05

12	IA	2.270643e+06
13	ID	6.201221e+05
14	IL	3.777541e+06
15	IN	6.612057e+06
16	KS	1.611266e+05
17	KY	3.579454e+05
18	LA	2.023649e+07
19	MA	8.482760e+05
20	MD	6.456728e+05
21	ME	1.915654e+06
22	MI	2.376553e+06
23	MN	1.447308e+06
24	MO	2.894341e+05
25	MS	1.804781e+06
26	MT	6.270088e+04
27	NC	2.007654e+06
28	ND	1.843561e+05
29	NE	4.363898e+05
30	NH	9.384412e+04
31	NJ	1.500825e+06
32	NM	1.776611e+05
33	NV	1.361702e+05
34	NY	2.063172e+06
35	OH	1.119034e+06
36	OK	1.035920e+06
37	OR	4.879712e+05
38	PA	5.620968e+06
39	RI	1.909059e+05
40	SC	2.392093e+06
41	SD	2.822062e+04
42	TN	2.476971e+06
43	TX	3.683709e+07
44	US	1.402790e+08
45	UT	7.633645e+05
46	VA	2.638126e+06
47	VT	5.819000e+03
48	WA	8.768182e+05
49	WI	1.723626e+06
50	WV	6.389289e+05
51	WY	1.489009e+06

[60]:
```
import matplotlib.pyplot as plt

# Filter out the United States row
average_consumption_df_filtered =␣
  ↪average_consumption_df[average_consumption_df['State'] != 'US']
```

```
# Sort the DataFrame for better visualization (optional)
average_consumption_df_filtered = average_consumption_df_filtered.
 ↪sort_values('Average Megawatt Hours', ascending=False)

# Create the bar graph
plt.figure(figsize=(15, 8))
plt.bar(average_consumption_df_filtered['State'],␣
 ↪average_consumption_df_filtered['Average Megawatt Hours'], color='skyblue')

# Add titles and labels
plt.title('Average Megawatt Hours Consumed Per State (Per Year)', fontsize=16)
plt.xlabel('State', fontsize=12)
plt.ylabel('Average Megawatt Hours', fontsize=12)

# Rotate x-axis labels for better readability
plt.xticks(rotation=90)

# Display the bar graph
plt.tight_layout()
plt.show()
```



[62]:
```
# Step 1: Calculate the average energy consumed from 2016 to 2023
average_energy_consumed = df['direct-use'].sum() / len(df['period'].unique())

# Step 2: Calculate the total energy needed to meet a 30% increase by 2030
target_energy_consumption = average_energy_consumed * 1.30
```

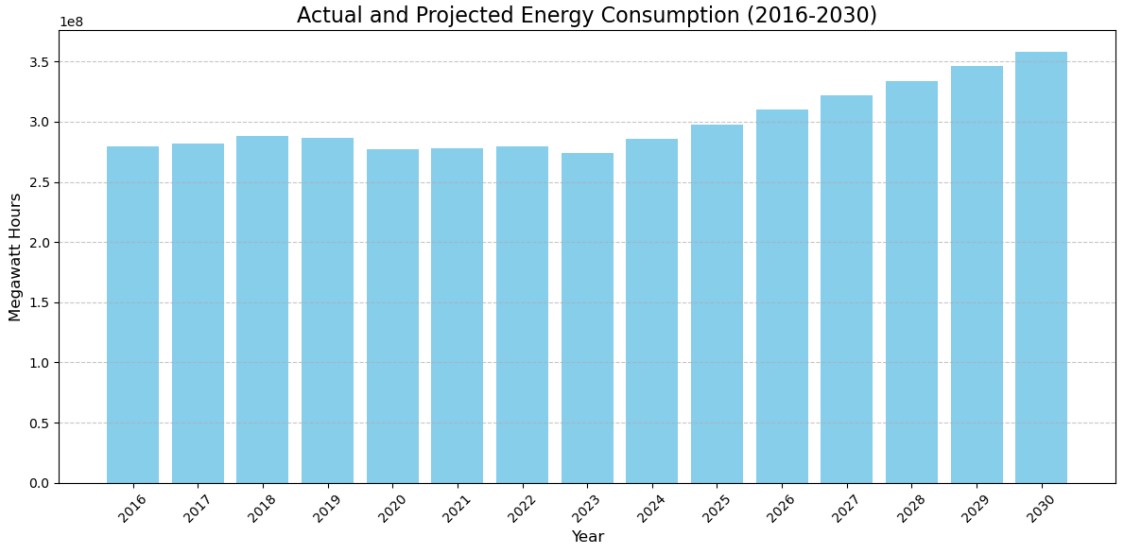
```
# Step 3: Calculate the additional energy needed
additional_energy_needed = target_energy_consumption - average_energy_consumed

# Step 4: Calculate the required annual increase
years_to_2030 = 7  # 2030 is 7 years from 2023
annual_increase_needed = additional_energy_needed / years_to_2030

# Step 5: Create a DataFrame to display the results
results = {
    "Metric": [
        "Average Energy Consumed (2016-2023)",
        "Target Energy Consumption by 2030 (+30%)",
        "Additional Energy Needed by 2030",
        "Annual Increase Needed (to meet target)",
    ],
    "Megawatt Hours": [
        average_energy_consumed,
        target_energy_consumption,
        additional_energy_needed,
        annual_increase_needed,
    ],
}

results_df = pd.DataFrame(results)

# Display the results as a table
print("Energy Production Calculations:")
display(results_df)
```

```
Energy Production Calculations:
```

	Metric	Megawatt Hours
0	Average Energy Consumed (2016-2023)	2.805579e+08
1	Target Energy Consumption by 2030 (+30%)	3.647253e+08
2	Additional Energy Needed by 2030	8.416737e+07
3	Annual Increase Needed (to meet target)	1.202391e+07

[66]:
```
import matplotlib.pyplot as plt

# Recalculate necessary variables
average_energy_consumed = df['direct-use'].sum() / len(df['period'].unique())
target_energy_consumption = average_energy_consumed * 1.30
additional_energy_needed = target_energy_consumption - average_energy_consumed
years_to_2030 = 7
annual_increase_needed = additional_energy_needed / years_to_2030

# Step 1: Calculate actual data (2016-2023)
actual_years = sorted(df['period'].unique())
```

```
actual_consumption = df.groupby('period')['direct-use'].sum()

# Step 2: Calculate prospective data (2024-2030)
prospective_years = list(range(2024, 2031))
last_actual_year_consumption = actual_consumption.iloc[-1]
prospective_consumption = []

# Incremental increase each year to meet the target
for i in range(1, len(prospective_years) + 1):
    increase = annual_increase_needed * i
    prospective_consumption.append(last_actual_year_consumption + increase)

# Combine data
all_years = [str(year) for year in (actual_years + prospective_years)]  #␣
 ↪Convert years to strings
all_consumption = list(actual_consumption) + prospective_consumption

# Step 3: Plot the data
plt.figure(figsize=(12, 6))
plt.bar(all_years, all_consumption, color='skyblue')

# Add labels and title
plt.title('Actual and Projected Energy Consumption (2016-2030)', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Megawatt Hours', fontsize=12)
plt.xticks(rotation=45)

# Show gridlines for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the bar graph
plt.tight_layout()
plt.show()
```


Actual and Projected Energy Consumption (2016-2030)
1e8
Megawatt Hours
Year
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030

[ ]: