SI 206
Fall 2024
Isaac Servin
Joseph Marino
Matthew Cappo

Link to GitHub repository: https://github.com/isaac-servin/final_project.git

1. The primary goal of our project was to explore the relationship between rising temperatures and energy production, focusing on finding a positive correlation between temperature increases and energy output. We aimed to understand how temperature changes affect production trends, particularly in renewable and non-renewable energy sources. To achieve this, we planned to collect data from the National Temperature Index for temperature trends, Electricity Maps for real-time and historical electricity production by energy source, and the U.S. Energy Information Administration for detailed energy production and distribution data. Combining these datasets, we aimed to analyze how temperature increases impact energy production and identify key patterns.

2. We successfully achieved several key goals in our project by working with the Electricity Maps API, as well as the scraped data from the U.S. Energy Information Administration. However, instead of using the National Temperature Index, we pivoted to using the Open Meteo Weather API to gather temperature data. This allowed us to collect data on maximum and minimum temperatures over several years, providing a detailed view of temperature trends. Additionally, we successfully gathered data on electricity consumption through the Electricity Maps API, which included a breakdown by energy source. We also scraped data from the U.S. Energy Information Administration to supplement our analysis with more comprehensive energy production figures. These datasets provided the necessary foundation to analyze how temperature variations correlate with energy production and consumption, enabling us to meet the primary objectives of our project.

3. During the project, we encountered several challenges that forced adjustments to our original plans. One important problem was with the original weather API we intended to use for temperature data, which only provided a CSV file instead of a functional API. This limitation forced us to pivot and use the Open Meteo Weather API instead, which provided more flexibility and allowed us to gather the temperature data we needed. Another challenge was adapting to different Python libraries to effectively display and analyze the data. Visualizing data clearly and insightfully required experimenting with various libraries, which took additional time and effort as we worked to improve our skills working with the various Python libraries. Finally, because we relied on data from

different sources, we faced difficulties in drawing consistent conclusions. The variability in the data formats and the types of insights each dataset offered sometimes led to conflicting interpretations, requiring us to find the discrepancies and ensure our conclusions were well-supported. These challenges highlighted the importance of flexibility and critical thinking in a data analysis project like this. Finally, after our grading presentation we made additional updates to our code and overall project[1].

calculations.txt

```
Average Energy Production by Source:
Coal: 80905.11
Petroleum: 11175.34
Natural_Gas: 155582.26
Other_Fossil_Gas: 1050.28
Nuclear: 65190.62
Hydroelectric_Conventional: 31349.31
Other_Renewable_Sources: 51524.09
Hydroelectric_Pumped_Storage: 14334.99
Other_Energy_Sources: 1636.27
Utility_Total: 368351.69
Estimated_Photovoltaic: 11852.35

Total Electricity Consumption by Sector:
Residential: 31599110152.00
Commercial: 29855119392.00
Industrial: 21845778820.00
Transportation: 159361578.00
Total: 83459369942.00
Direct_Use: 1545434961.00
Total_End_Use: 85004804903.00

```

Figure 1: Calculations From Data in Our Database

[1] Reference Page 10

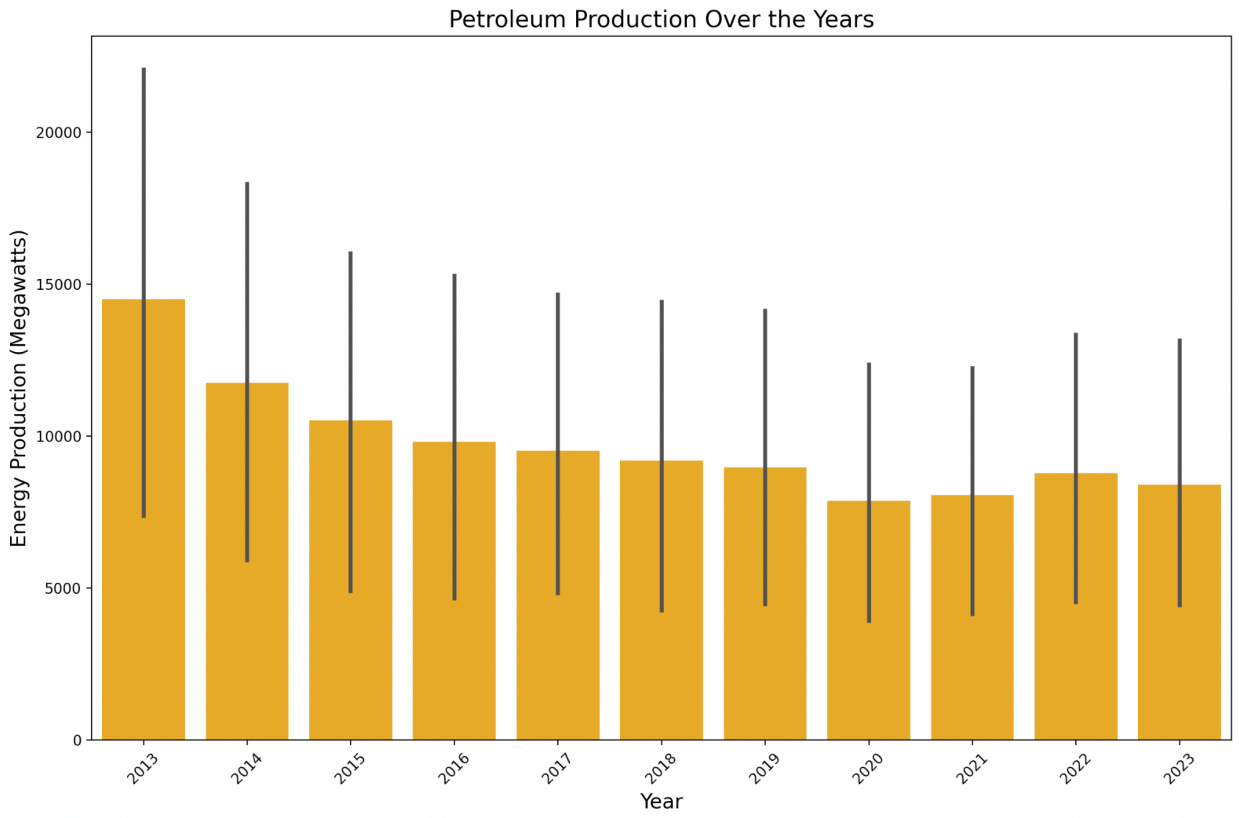

Figure 2: Petroleum Production Over the Years[2]

[2] A long confidence interval indicates a high degree of uncertainty about the true population parameter, meaning the range of values within which the true value could fall is wide. This suggests that the sample data does not provide a very precise estimate of the population mean; essentially, the more spread out the interval, the less precise the estimate is considered to be. In this context, it means that as time has passed, the accuracy of the estimate in which energy is to be used per year has decreased.

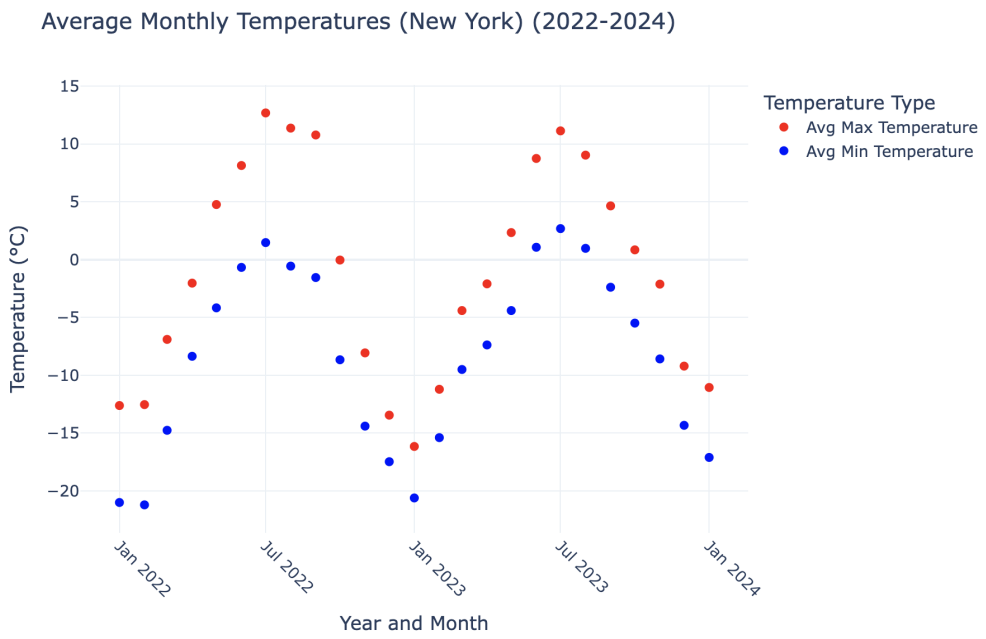

Figure 3: Scatterplot of Average Monthly Temperatures in New York

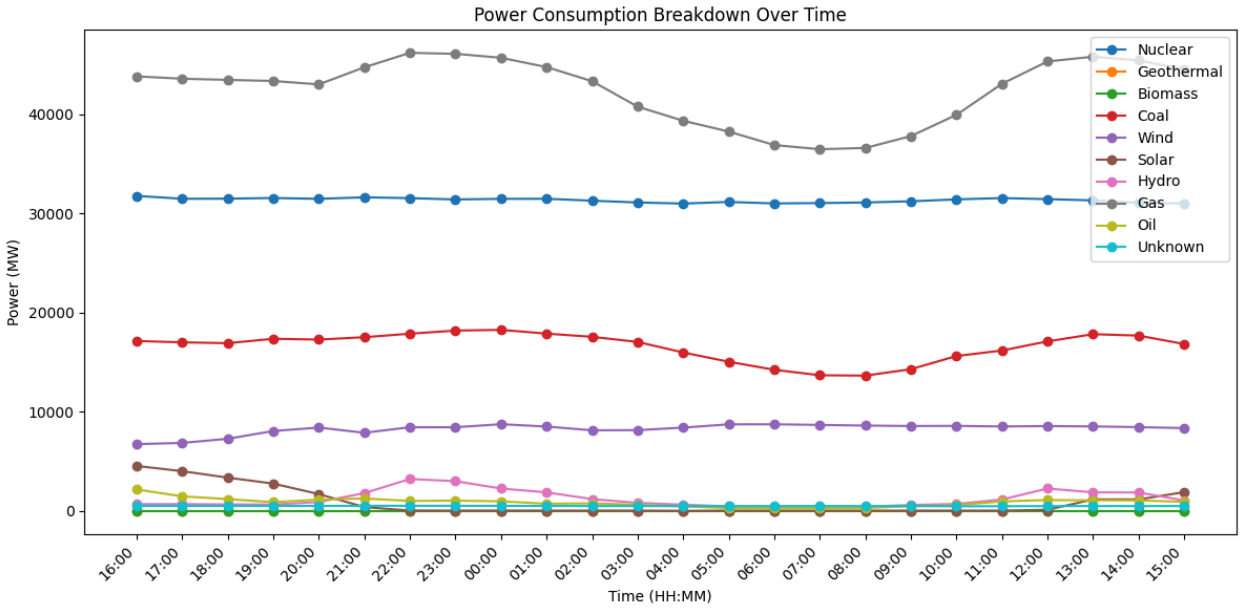

Figure 4: Matplotlib Line Graph of Zone US-MIDW-MISO (MI, WI, MN, IA, IL, IN, AR, LA)

Instructions:

The code must be run 20 times to get all the data loaded into their respective tables (energy_data, electicity_map_data, weather_data). The other tables (LastProcessed, last_inserted_datetime, last_inserted_date, sqlite_sequence) shown in the browser were created for serving as checkpoints in data collection. The table ‘Years’ shows the integer key which corresponds to years in the table ‘energy_data’.

Function Descriptions

1. fetch_eia_data(url, table_name, num_cols)

Purpose: Fetches data from a URL, parses it, and returns rows of data based on a defined number of columns (num_cols). It processes data incrementally using a last_index stored in the database.

- **Steps**:
    1. Connects to the database and ensures the LastProcessed table exists.
    2. Fetches the last processed index (last_index) from the LastProcessed table.
    3. Makes an HTTP request to the provided url and parses the HTML with BeautifulSoup.
    4. Extracts data in chunks of num_cols from the HTML table cells, starting from last_index.
    5. Updates last_index in the database to track progress for subsequent runs.

---

2. get_last_saved_index(table_name, conn)

Purpose: Retrieves the last processed index for a given table from the LastProcessed table.

- **Steps**:
    1. Queries the LastProcessed table for the last_index associated with the provided table_name.
    2. If no record exists, initializes last_index to 0 and inserts it into the database.
    3. Returns the retrieved or initialized last_index.

---

3. update_last_index(table_name, last_index, conn)

Purpose: Updates the LastProcessed table with the latest last_index for the specified table.

- **Steps**:
    1. Executes an UPDATE statement to set the last_index for the given table_name.

---

4. store_data(data, table_name)

Purpose: Stores fetched data into the energy_data table in the database. Handles both energy production (eia_data) and electricity consumption (electricity_data).

- **Steps**:
    1. Ensures the Years and energy_data tables exist in the database.
    2. Inserts unique year values into the Years table and retrieves their year_id.
    3. Inserts data rows into the energy_data table, filling unused columns with NULL for either eia_data or electricity_data.
    4. Uses INSERT OR IGNORE to prevent duplicate entries for year_id.

---

5. fetch_data_from_db()

Purpose: Reads all data from the energy_data table and returns it as two Pandas DataFrames: one for eia_data and one for electricity_data.

- **Steps**:
    1. Joins energy_data with Years to include year information.
    2. Splits the combined data into two DataFrames: one with columns related to energy production (eia_data) and another with electricity consumption (electricity_data).
    3. Returns the two DataFrames.

---

6. write_calculations(eia_df, electricity_df)

Purpose: Performs calculations on energy production and consumption data, then writes the results to a file.

- **Steps**:
    1. Converts production and consumption columns to numeric data types.
    2. Calculates average energy production for each source and total electricity consumption for each sector.
    3. Write the results to calculations.txt.

---

7. visualize_data(eia_df, electricity_df)

Purpose: Creates a visualization of energy production and consumption data using Matplotlib and Seaborn.

- **Steps**:
    1. Queries the database for Petroleum production data and electricity consumption data by sector.
    2. Merges the two datasets on the Year column.
    3. Generates a bar plot showing Petroleum production over the years.

---

8. load_weather_data(json_file, max_items=11)

Purpose: Loads weather data from a JSON file into the database incrementally.

- **Steps**:
    1. Ensures the weather_data and last_inserted_date tables exist.
    2. Reads data from the JSON file, tracking the last processed date to avoid re-insertion.
    3. Inserts up to max_items rows into the weather_data table.

---

9. calculate_and_write_results()

Purpose: Computes summary statistics for weather data and joins it with electricity consumption data to generate a report.

- **Steps**:
    1. Calculates average temperatures from weather_data.
    2. Computes total residential electricity usage from energy_data.
    3. Joins weather data with electricity data based on year and writes the results to weather_calculation_results.txt.

---

10. fetch_electricity_map_data(max_items_per_zone=24)

Purpose: Fetches electricity consumption breakdown data for various zones and stores it in the database.

- **Steps**:
    1. Ensures the electricity_map_data and last_inserted_datetime tables exist.
    2. Fetches data incrementally for each zone, based on the last inserted datetime.
    3. Inserts the fetched data into the electricity_map_data table.

__main__

Purpose: Coordinates the workflow of fetching, storing, processing, and visualizing data.

- **Steps**:
    1. Fetches and stores eia_data and electricity_data.
    2. Loads weather data from a JSON file.
    3. Fetches electricity map data.
    4. Generates visualizations and calculations.
    5. Outputs progress and results to the console and files.

12/10	Introduction to Plotly	Lecture 23	It helped with creating a scatter plot for figure 2A which assisted in conveying data temperature by year and month.

12/09	JSON and APIS	Lecture 17	This lecture helped with understanding how to convert API data into a json

12/11	Matplotlib and Github	Lecture 21	This lecture provided us with information that was instrumental in creating our visualizations to effectively convey our data.

12/11	Midterm and BS	Lecture 15	This lecture provided us with the necessary tools to properly scrape websites using BeautifulSoup, and obtain the specific information relevant to our analysis.

12/11	Merging Visualization	Lecture 20 - DB Joins https://www.stratascratch.com/blog/types-of-pandas-joins-and-how-to-use-them-in-python/	It helped with understanding how to JOIN/merge visualization.

Updates After Grading Presentation:

Calculations.txt file:
At the time our calculation was not returning any data. Here is our calculation.txt file which appears when one runs our code.

calculations.txt

```
Average Energy Production by Source:
Coal: 80905.11
Petroleum: 11175.34
Natural_Gas: 155582.26
Other_Fossil_Gas: 1050.28
Nuclear: 65190.62
Hydroelectric_Conventional: 31349.31
Other_Renewable_Sources: 51524.09
Hydroelectric_Pumped_Storage: 14334.99
Other_Energy_Sources: 1636.27
Utility_Total: 368351.69
Estimated_Photovoltaic: 11852.35

Total Electricity Consumption by Sector:
Residential: 31599110152.00
Commercial: 29855119392.00
Industrial: 21845778820.00
Transportation: 159361578.00
Total: 83459369942.00
Direct_Use: 1545434961.00
Total_End_Use: 85004804903.00

```

JOIN:
At the time of our grading session presentation we had not used JOIN. Here is where we have used JOIN in our coding.

```
    # Join weather_data and energy_data
    cur.execute("""
    SELECT w.date, w.temperature_2m_max, e.Residential
    FROM weather_data w
    JOIN Years y ON substr(w.date, 1, 4) = y.year
    JOIN energy_data e ON y.id = e.year_id
    LIMIT 10
    """)
```

SELECT:

At the time of our grading session presentation, it was not clear where we were using SELECT to select items from the tables and calculate something from the data. Here is an example of where we use SELECT.

```
366  def calculate_and_write_results():
367      conn = sqlite3.connect('project_database.db')
368      cur = conn.cursor()
369
370      # Calculate average temperatures
371      cur.execute("SELECT AVG(temperature_2m_max), AVG(temperature_2m_min) FROM weather_data")
372      result = cur.fetchone()
373      avg_max, avg_min = result if result else (None, None)
```

100 Entries of an API:

At the time of our grading session presentation we had not run our code prior to showing proof of 100 rows of data from one of our API's. Here are 100 rows of data from our API for electricity_maps_data. All of our websites/API's have 100 rows of data which are limited to a max of 25 items each time a file is run to gather data.

	zone	datetime	nuclear	geothermal	biomass	coal	wind	solar	hydro	gas	oil	unknown
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	US-MIDW-MISO	2024-12-13T00:00:00.000Z	10628.0	0.0	0.0	25795.0	8448.0	0.0	1930.0	31052.0	1.0	357.0
2	US-MIDA-PJM	2024-12-13T00:00:00.000Z	33226.0	0.0	0.0	19983.0	1798.0	0.0	1849.0	47473.0	373.0	527.0
3	US-NY-NYIS	2024-12-13T00:00:00.000Z	3881.0	0.0	13.0	507.0	1971.0	0.0	4209.0	9986.0	9.0	264.0
4	US-NE-ISNE	2024-12-13T00:00:00.000Z	3667.0	0.0	601.0	41.0	1051.0	0.0	3866.0	6944.0	1.0	114.0
5	US-CENT-SWPP	2024-12-13T00:00:00.000Z	2043.0	0.0	11.0	10244.0	10803.0	0.0	440.0	11950.0	0.0	59.0
6	US-MIDW-MISO	2024-12-12T23:00:00.00...	11549.0	0.0	1.0	25642.0	8585.0	0.0	2256.0	31431.0	7.0	358.0
7	US-MIDA-PJM	2024-12-12T23:00:00.00...	33238.0	0.0	0.0	20367.0	1644.0	0.0	2516.0	49488...	481.0	518.0
8	US-NY-NYIS	2024-12-12T23:00:00.00...	4058.0	0.0	12.0	503.0	2045.0	0.0	4298.0	10100.0	12.0	264.0
9	US-NE-ISNE	2024-12-12T23:00:00.00...	3702.0	0.0	588.0	47.0	1118.0	0.0	3977.0	7278.0	22.0	114.0
10	US-CENT-SWPP	2024-12-12T23:00:00.00...	2047.0	0.0	11.0	10489.0	10290.0	57.0	503.0	12065...	1.0	63.0
11	US-MIDW-MISO	2024-12-12T22:00:00.00...	11699.0	0.0	2.0	24694.0	8836.0	610.0	2150.0	28829...	6.0	351.0
12	US-MIDA-PJM	2024-12-12T22:00:00.00...	33301.0	0.0	0.0	19787.0	1615.0	661.0	2501.0	48953...	445.0	521.0
13	US-NY-NYIS	2024-12-12T22:00:00.00...	4001.0	0.0	7.0	451.0	1792.0	15.0	4327.0	10571.0	10.0	239.0
14	US-NE-ISNE	2024-12-12T22:00:00.00...	3704.0	0.0	590.0	43.0	1105.0	3.0	4033.0	7322.0	2.0	217.0
15	US-CENT-SWPP	2024-12-12T22:00:00.00...	2029.0	0.0	11.0	10435.0	10264.0	68.0	499.0	11994.0	1.0	64.0
16	US-MIDW-MISO	2024-12-12T21:00:00.000Z	11914.0	0.0	1.0	20799.0	22815.0	342...	1143.0	25849...	6.0	334.0
17	US-MIDA-PJM	2024-12-12T21:00:00.000Z	33418.0	0.0	0.0	19835.0	1619.0	692.0	2479.0	49030...	446.0	523.0
18	US-NY-NYIS	2024-12-12T21:00:00.000Z	3886.0	0.0	0.0	422.0	1970.0	15.0	3749.0	10067.0	9.0	257.0
19	US-NE-ISNE	2024-12-12T21:00:00.000Z	3687.0	0.0	591.0	45.0	1149.0	6.0	3476.0	6987.0	10.0	249.0
20	US-CENT-SWPP	2024-12-12T21:00:00.000Z	2021.0	0.0	11.0	9956.0	10495.0	230.0	734.0	10005...	0.0	70.0
21	US-MIDW-MISO	2024-12-12T20:00:00.00...	11860.0	0.0	1.0	20221.0	22932.0	449...	986.0	24602...	7.0	322.0
22	US-MIDA-PJM	2024-12-12T20:00:00.00...	33363.0	0.0	0.0	18762.0	1995.0	347...	1100.0	45442...	456.0	517.0
23	US-NY-NYIS	2024-12-12T20:00:00.00...	3796.0	0.0	0.0	398.0	1995.0	75.0	3348.0	9292.0	10.0	370.0
24	US-NE-ISNE	2024-12-12T20:00:00.00...	3686.0	0.0	592.0	48.0	1119.0	97.0	2805.0	6129.0	1.0	273.0
25	US-CENT-SWPP	2024-12-12T20:00:00.00...	2046.0	0.0	10.0	9856.0	11039.0	260.0	808.0	9239.0	1.0	70.0
26	US-MIDW-MISO	2024-12-12T19:00:00.000Z	11929.0	0.0	1.0	20611.0	22689.0	444...	941.0	25743.0	10.0	322.0
27	US-MIDA-PJM	2024-12-12T19:00:00.000Z	33099.0	0.0	0.0	18325.0	2261.0	529...	968.0	43778...	525.0	523.0
28	US-NY-NYIS	2024-12-12T19:00:00.000Z	3952.0	0.0	0.0	460.0	2011.0	135.0	3072.0	8473.0	13.0	411.0
29	US-NE-ISNE	2024-12-12T19:00:00.000Z	3716.0	0.0	595.0	59.0	1123.0	284.0	2665.0	4799.0	10.0	344.0
30	US-CENT-SWPP	2024-12-12T19:00:00.000Z	2063.0	0.0	9.0	10260.0	10716.0	270.0	830.0	9639.0	0.0	69.0
31	US-MIDW-MISO	2024-12-12T18:00:00.000Z	11977.0	0.0	1.0	21369.0	21967.0	430...	938.0	26784...	11.0	326.0
32	US-MIDA-PJM	2024-12-12T18:00:00.000Z	33029.0	0.0	0.0	18228.0	2532.0	587...	944.0	42389...	522.0	498.0
33	US-NY-NYIS	2024-12-12T18:00:00.000Z	4096.0	0.0	0.0	432.0	2142.0	148.0	3184.0	8072.0	12.0	418.0
34	US-NE-ISNE	2024-12-12T18:00:00.000Z	3684.0	0.0	577.0	42.0	1112.0	469.0	2514.0	4328.0	8.0	170.0
35	US-CENT-SWPP	2024-12-12T18:00:00.000Z	2054.0	0.0	10.0	10350.0	10913.0	213.0	894.0	9894.0	1.0	66.0
36	US-MIDW-MISO	2024-12-12T17:00:00.000Z	11805.0	0.0	1.0	24191.0	8574.0	472...	725.0	26445...	13.0	342.0
37	US-MIDA-PJM	2024-12-12T17:00:00.000Z	32697.0	0.0	0.0	18370.0	2545.0	588...	929.0	41369.0	508.0	485.0
38	US-NY-NYIS	2024-12-12T17:00:00.000Z	4067.0	0.0	0.0	506.0	2119.0	171.0	3058.0	7754.0	14.0	420.0

1 - 38 of 100 Go to: 1