```
! git clone https://github.com/Neurocomputation-Lab-NCAI-
NEDUET/INSTRUX_AI.git
```

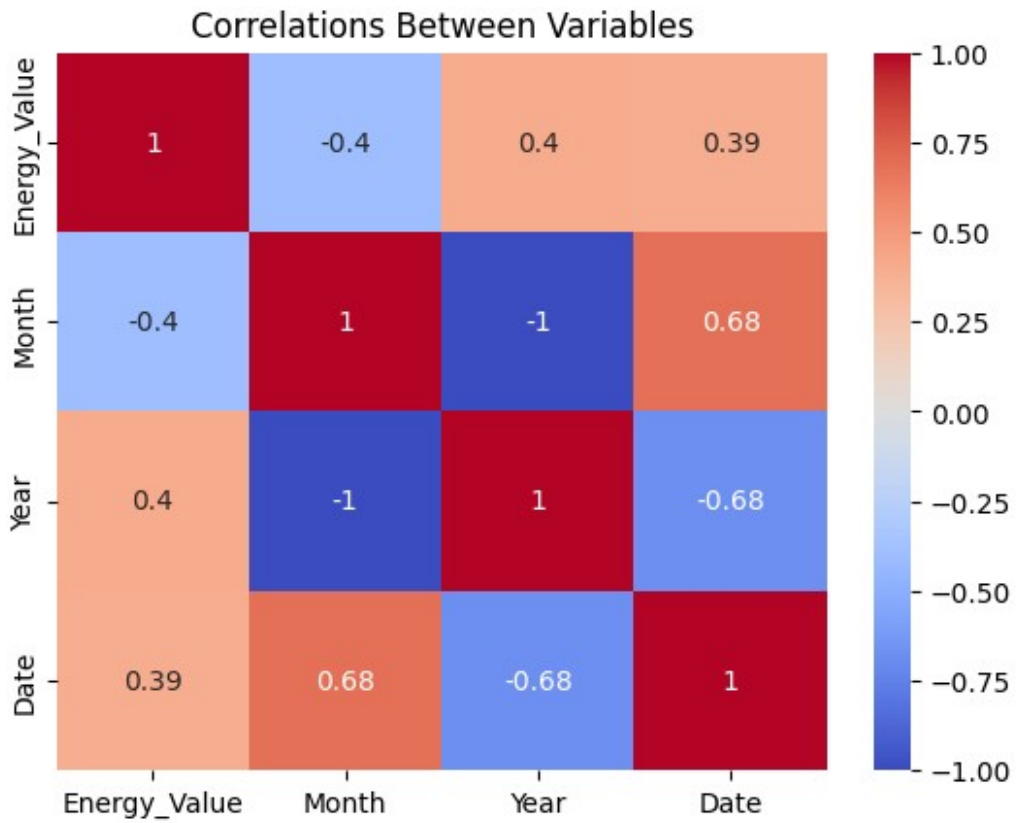
```
Cloning into 'INSTRUX_AI'...
remote: Enumerating objects: 28, done.ote: Counting objects: 100%
(28/28), done.ote: Compressing objects: 100% (27/27), done.ote: Total
28 (delta 6), reused 17 (delta 1), pack-reused 0
```

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

data = pd.read_csv('/content/INSTRUX_AI/data/instrux_3.csv')

data.head()
```

	Energy_Value	Month	Year	Date	Time	Day
0	5102980.5	12	2022	13	12:33:42	Tuesday
1	5103799.0	12	2022	13	15:07:57	Tuesday
2	5104085.0	12	2022	13	16:00:07	Tuesday
3	5104383.0	12	2022	13	17:00:07	Tuesday
4	5104698.0	12	2022	13	18:00:06	Tuesday

```
# Compute the correlations between variables
correlations = data.corr()
# Heatmap (for correlations)
sns.heatmap(correlations, annot=True, cmap='coolwarm')
plt.title('Correlations Between Variables')
plt.show()
```

```
<ipython-input-5-f07a7e4ee333>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  correlations = data.corr()
```



1. Energy_Value and Year: The correlation coefficient is 0.4, which indicates a weak to moderate positive correlation between these two variables. As the year increases, the energy consumption tends to increase.

2. Energy_Value and Date: The correlation coefficient is 0.39, which indicates a weak to moderate positive correlation between these two variables. As the Date increases, the energy consumption also tends to increase.

3. Energy_Value and Month: The correlation coefficient is -0.4, which indicates a weak to moderate negative correlation between these two variables. As the Month increases, the energy consumption decreases and vice versa.

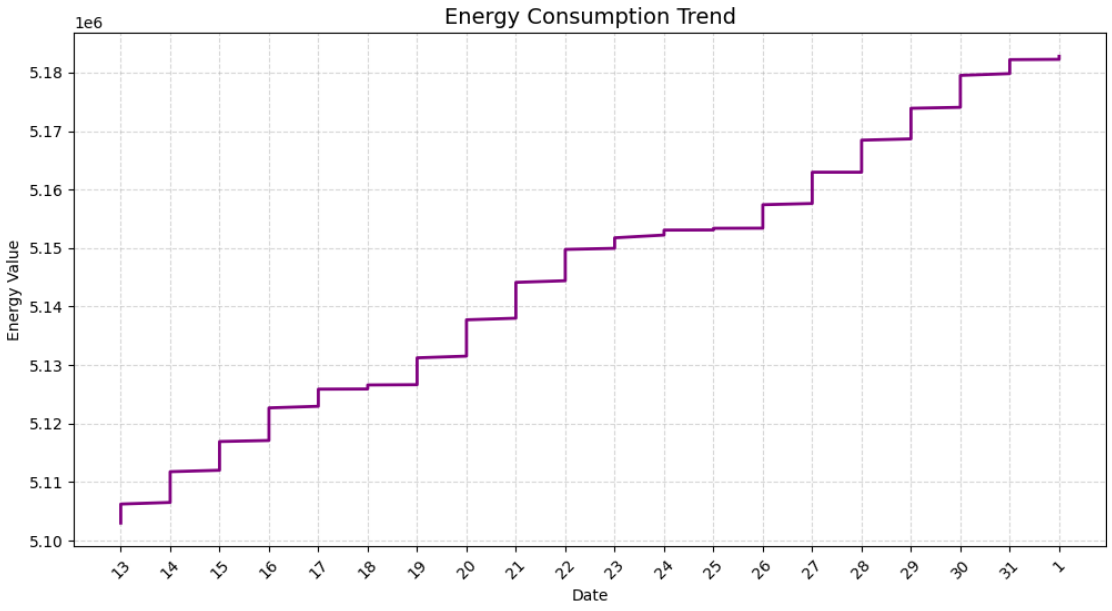
4. Month and Year: The correlation coefficient is -1, which indicates a perfect negative correlation. This is expected, as the dataset seems to be mostly from December 2022, and this negative correlation represents the transition from 2022 to 2023.

5. Month and Date: The correlation coefficient is 0.68, which indicates a moderate positive correlation between these two variables. As the month value increases, the date value also tends to increase.

6. Year and Date: The correlation coefficient is -0.680, which indicates a moderate negative correlation between these two variables. As the year value increases, the date value tends to decrease.

NOTE: I didn't include Time and Day as they are non numeric, and they have very less correlation. Also by including them my heatmap was being affected so I excluded them.

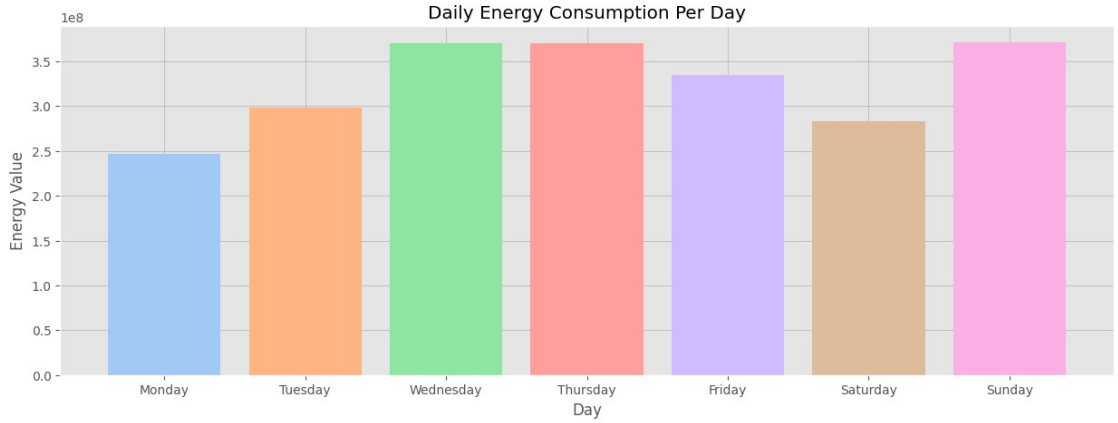
```
#Energy consumption trend
data = pd.read_csv('/content/INSTRUX_AI/data/instrux_3.csv',
parse_dates=['Date']) #Used parse dates to ensure date is interpreted
correctly for plotting, didnt define above because it was messing with
corr.
# Set the plot size
plt.figure(figsize=(12, 6))

# Plot the data with a custom line color and style
plt.plot(data['Date'], data['Energy_Value'], color='purple',
linestyle='-', linewidth=2)

plt.xlabel('Date')
plt.ylabel('Energy Value')
plt.title('Energy Consumption Trend', fontsize=14)

# Add a grid to the plot
plt.grid(True, linestyle='--', alpha=0.5)

plt.xticks(rotation=45)
plt.show()
```



As we've already discovered the correlation between Date and Energy Value, here is the visualization. As date is increasing, energy is also increasing.

```
# Energy consumption per day
daily_consumption = data.groupby('Day')['Energy_Value'].sum()
days_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
```

```
'Saturday', 'Sunday']
daily_consumption = daily_consumption.reindex(days_order)
# Set plot style
plt.style.use('ggplot')
# Set custom colors for the bars
colors = sns.color_palette('pastel', 7)
# Create the bar plot
plt.figure(figsize=(15, 5))
plt.bar(daily_consumption.index, daily_consumption.values,
color=colors)
plt.xlabel('Day')
plt.ylabel('Energy Value')
plt.title('Daily Energy Consumption Per Day')
# Customize the grid appearance
plt.grid(color='gray', linestyle='-', linewidth=0.5, alpha=0.5)

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

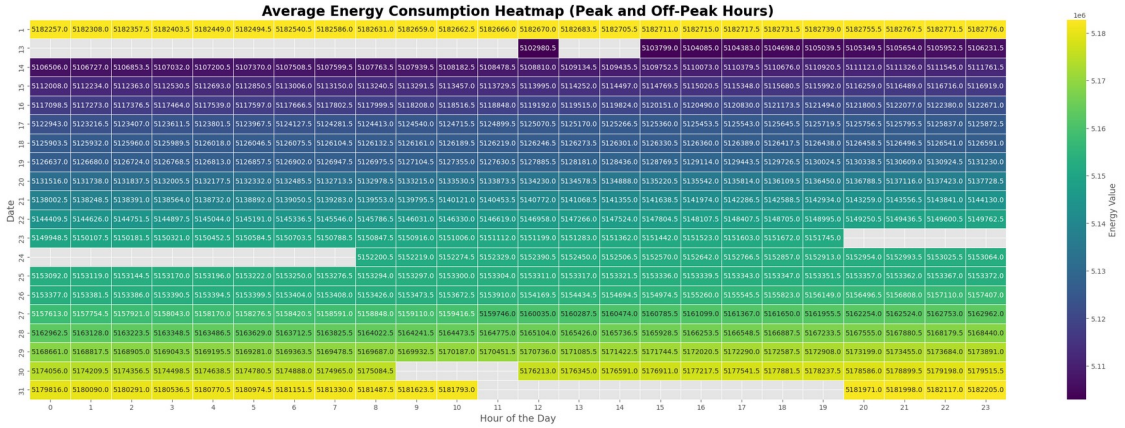


Energy consumed on per day basis. It is a sum of all days energy consumption, for example take Monday, it is pointing to 2.4, it means we sum up all the energy values of all Mondays in the dataset (there were 2 Mondays), hence we got 246334544.

```
# Converting 'Time' column to a datetime object and extracting the
hour
data['Hour'] = pd.to_datetime(data['Time']).dt.hour

# Group by date and hour, and calculate the average energy consumption
date_hour_energy = data.groupby(['Date', 'Hour'])
['Energy_Value'].mean().reset_index()

# Pivot the grouped data to create a matrix of average energy
consumption
energy_matrix = date_hour_energy.pivot('Date', 'Hour', 'Energy_Value')

# Create the heatmap with annotations (numbers) and larger cells
plt.figure(figsize=(32, 10))
```

```
sns.heatmap(energy_matrix, cmap='viridis', linewidths=.5, annot=True,
fmt='.1f', cbar_kws={'label': 'Energy Value'})
plt.xlabel('Hour of the Day', fontsize=14)
plt.ylabel('Date', fontsize=14)
plt.title('Average Energy Consumption Heatmap (Peak and Off-Peak
Hours)', fontsize=20, fontweight='bold')
plt.show()
```

```
<ipython-input-8-a48dda67c34e>:8: FutureWarning: In a future version
of pandas all arguments of DataFrame.pivot will be keyword-only.
  energy_matrix = date_hour_energy.pivot('Date', 'Hour',
'Energy_Value')
```



First, let's see what colours define:

1. Dark purple: Lower energy consumption values.
2. Light purple/blue: Moderate energy consumption values.
3. Green: Higher energy consumption values.
4. Yellow: The highest energy consumption values.

Now as you can observe, each cell represents the average energy consumption for a particular date and hour. The color intensity of the cell indicates how high or low the energy consumption is for that time. Darker colors indicate lower energy consumption (off peak hours), while lighter colors (towards yellow) represent higher energy consumptiomn (Peak hours)

It is observed that from 13-11-22 (start of dataset), low energy is consumed, but as the date is increasing energy value is also increasing, hence on 1-01-23 (end of dataset), it is the highest.

But if we look from hourly perspective, we can see that majority cells have darker shade (off peak hours) in the starting hours and lighter shade at midnight.

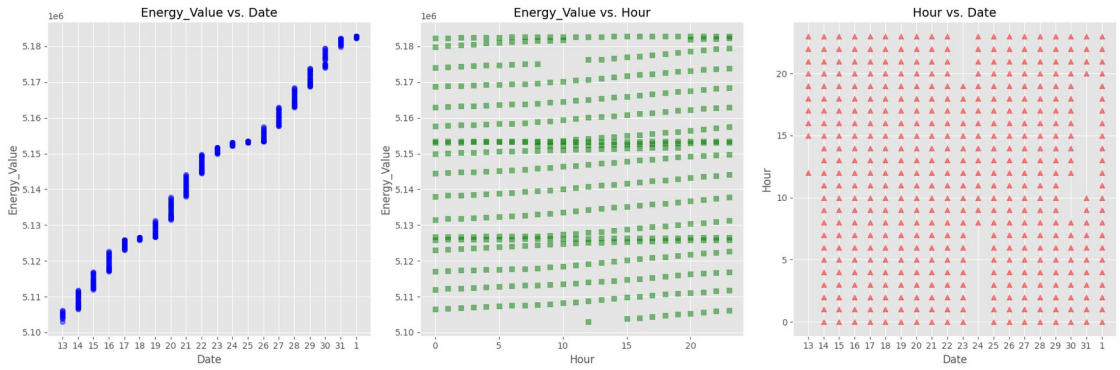
My Theories:

1. The increase from early dates to ending dates could be because the expansion of production (Maybe they're expanding their business), addition of new machinery, or increasing demand for textile products.

2. It suggests that the Rainbow Hosiery has a night shift or operates more intensively during nighttime hours. It opens at 8:00am and closes at 6:00am so it might be the case. This could be a strategy to take advantage of lower electricity rates during off-peak hours, reduce daytime noise or heat generation, or accommodate worker schedules. The dataset covers December and January, peak hours are from 6:00 pm to 10:00 pm. It is observed that higher energy consumption are outside these peak hours, especially towards midnight, it confirms that the industry is strategically shifting its operations to minimize energy costs.

NOTE: There are some missing hours as you can still see some colourless cells. I will fill these with interpolation after analysing whether I should use linear interpolation of non linear.

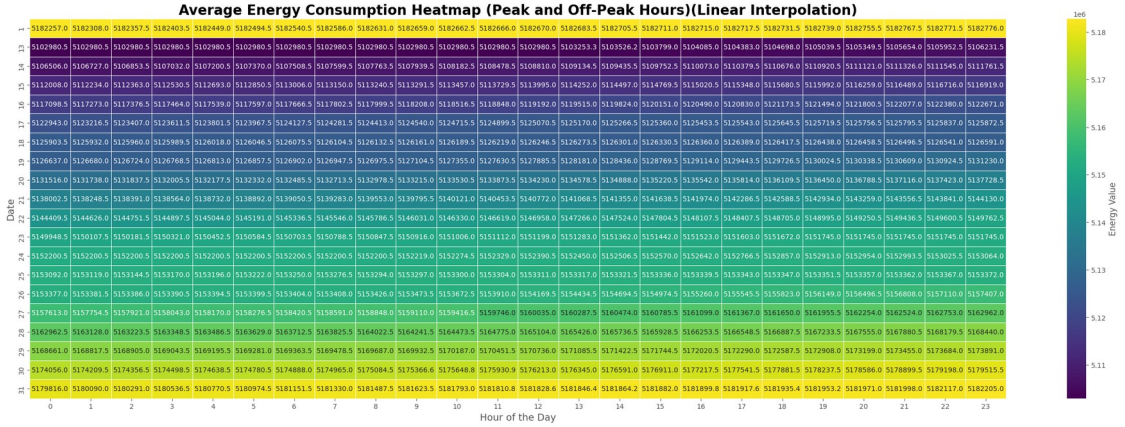
```
#Checking trend

# Create a larger figure
plt.figure(figsize=(18, 6))

# Energy_Value and Date
plt.subplot(1, 3, 1)
plt.scatter(data['Date'], data['Energy_Value'], c='blue', marker='o',
alpha=0.5)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Energy_Value', fontsize=12)
plt.title('Energy_Value vs. Date', fontsize=14)

# Energy_Value and Time (Hour)
plt.subplot(1, 3, 2)
plt.scatter(data['Hour'], data['Energy_Value'], c='green', marker='s',
alpha=0.5)
plt.xlabel('Hour', fontsize=12)
plt.ylabel('Energy_Value', fontsize=12)
plt.title('Energy_Value vs. Hour', fontsize=14)

# Date and Time (Hour)
plt.subplot(1, 3, 3)
plt.scatter(data['Date'], data['Hour'], c='red', marker='^',
alpha=0.5)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Hour', fontsize=12)
plt.title('Hour vs. Date', fontsize=14)

# Adjust the layout and show the plots
plt.tight_layout()
plt.show()
```



This is just to confirm the relevant variables (Energy Value vs Date)are linear so now I can implement linear interpolation.

```
#LINEAR INTERPOLATION

# Convert the 'Time' column to a datetime object and extract the hour
data['Hour'] = pd.to_datetime(data['Time']).dt.hour

# Group by date and hour, and calculate the average energy consumption
date_hour_energy = data.groupby(['Date', 'Hour'])
['Energy_Value'].mean().reset_index()

# Pivot the grouped data to create a matrix of average energy
consumption
energy_matrix = date_hour_energy.pivot('Date', 'Hour', 'Energy_Value')

# Fill missing values using linear interpolation
energy_matrix.interpolate(method='linear', axis=1, inplace=True)

# There were still few values missing so I use the fillna() function
to fill them with the nearest valid value
energy_matrix.fillna(method='bfill', axis=1, inplace=True)
energy_matrix.fillna(method='ffill', axis=1, inplace=True)

# Create the heatmap with annotations (numbers) and larger cells
plt.figure(figsize=(32, 10))
sns.heatmap(energy_matrix, cmap='viridis', linewidths=.5, annot=True,
fmt='.1f', cbar_kws={'label': 'Energy Value'})
plt.xlabel('Hour of the Day', fontsize=14)
plt.ylabel('Date', fontsize=14)
plt.title('Average Energy Consumption Heatmap (Peak and Off-Peak
Hours)(Linear Interpolation)', fontsize=20, fontweight='bold')
plt.show()
```

```
<ipython-input-10-4bdfae5761b4>:10: FutureWarning: In a future version
of pandas all arguments of DataFrame.pivot will be keyword-only.
  energy_matrix = date_hour_energy.pivot('Date', 'Hour',
'Energy_Value')
```



After analyzing scatter plot patterns, I used linear interpolation and this is the result. Linear interpolation is basically an estimation of an unknown value that falls within two known values.

```
import datetime
# Find the peak and off-peak hours with energy values for each date
peak_and_off_peak = energy_matrix.apply(lambda row: {
    'peak_hour': row.idxmax(),
    'peak_energy_value': row.max(),
    'off_peak_hour': row.idxmin(),
    'off_peak_energy_value': row.min()
}, axis=1)

# Create a DataFrame from the result
peak_and_off_peak_df = pd.DataFrame(peak_and_off_peak.tolist(),
index=energy_matrix.index)
peak_and_off_peak_df.reset_index(inplace=True)
peak_and_off_peak_df.columns = ['Date', 'Peak Hour', 'Peak Energy
Value', 'Off-Peak Hour', 'Off-Peak Energy Value']

# Convert the hour columns to the AM-PM format
peak_and_off_peak_df['Peak Hour'] = peak_and_off_peak_df['Peak
Hour'].apply(lambda x: datetime.datetime.strptime(str(x),
"%H").strftime("%I %p"))
peak_and_off_peak_df['Off-Peak Hour'] = peak_and_off_peak_df['Off-Peak
Hour'].apply(lambda x: datetime.datetime.strptime(str(x),
"%H").strftime("%I %p"))

# Display the DataFrame
print(peak_and_off_peak_df)
```

```
   Date Peak Hour  Peak Energy Value Off-Peak Hour  Off-Peak Energy
Value
0     1     11 PM          5182776.0         12 AM
5182257.0
1    13     11 PM          5106231.5         12 AM
```

```
5102980.5
2    14     11 PM          5111761.5          12 AM
5106506.0
3    15     11 PM          5116919.0          12 AM
5112008.0
4    16     11 PM          5122671.0          12 AM
5117098.5
5    17     11 PM          5125872.5          12 AM
5122943.0
6    18     11 PM          5126591.0          12 AM
5125903.5
7    19     11 PM          5131230.0          12 AM
5126637.0
8    20     11 PM          5137728.5          12 AM
5131516.0
9    21     11 PM          5144130.0          12 AM
5138002.5
10   22     11 PM          5149762.5          12 AM
5144409.5
11   23     07 PM          5151745.0          12 AM
5149948.5
12   24     11 PM          5153064.0          12 AM
5152200.5
13   25     11 PM          5153372.0          12 AM
5153092.0
14   26     11 PM          5157407.0          12 AM
5153377.0
15   27     11 PM          5162962.0          12 AM
5157613.0
16   28     11 PM          5168440.0          12 AM
5162962.5
17   29     11 PM          5173891.0          12 AM
5168661.0
18   30     11 PM          5179515.5          12 AM
5174056.0
19   31     11 PM          5182205.0          12 AM
5179816.0
```

NOTE: Please note that here the first date is 1, It is 01-01-23 while from 13 it is, 13-12-22.

I've summarized the heatmap here, now we can clearly observe that highest energy consumption is at 23rd hour (11pm), after peak hours (i.e 6:00pm-10:00pm) and lowest energy consumption is at starting hour of industry i.e 12am.

```
# Convert the 'Time' column to a datetime object
data['Time'] = pd.to_datetime(data['Time'])

# Group by date, and calculate the first and last recorded times
first_last_times = data.groupby(['Date'])['Time'].agg(['min',
'max']).reset_index()
```

```
# Rename columns
first_last_times.columns = ['Date', 'First Recorded Time', 'Last
Recorded Time']

# Extract just the time part and add day of the week
first_last_times['First Recorded Time'] = first_last_times['First
Recorded Time'].dt.strftime('%I:%M:%S %p')
first_last_times['Last Recorded Time'] = first_last_times['Last
Recorded Time'].dt.strftime('%I:%M:%S %p')
first_last_times['Day'] = pd.to_datetime(first_last_times['Date'],
format="%d").dt.day_name()

# Reorder the columns
first_last_times = first_last_times[['Date', 'Day', 'First Recorded
Time', 'Last Recorded Time']]

# Set display options for neatness
pd.set_option('display.width', 200)
pd.set_option('display.max_columns', 10)
pd.set_option('display.colheader_justify', 'center')

# Display the DataFrame
print(first_last_times)
```

	Date	Day	First Recorded Time	Last Recorded Time
0	1	Monday	12:00:38 AM	11:00:05 PM
1	13	Saturday	12:33:42 PM	11:00:06 PM
2	14	Sunday	12:00:06 AM	11:00:06 PM
3	15	Monday	12:00:07 AM	11:00:02 PM
4	16	Tuesday	12:00:07 AM	11:00:03 PM
5	17	Wednesday	12:00:02 AM	11:00:01 PM
6	18	Thursday	12:00:01 AM	11:00:00 PM
7	19	Friday	12:00:00 AM	11:00:05 PM
8	20	Saturday	12:00:05 AM	11:00:07 PM
9	21	Sunday	12:00:09 AM	11:00:06 PM
10	22	Monday	12:00:06 AM	11:00:00 PM
11	23	Tuesday	12:00:00 AM	07:00:06 PM
12	24	Wednesday	08:17:24 AM	11:00:08 PM
13	25	Thursday	12:00:28 AM	11:00:05 PM
14	26	Friday	12:00:15 AM	11:00:09 PM
15	27	Saturday	12:00:09 AM	11:59:46 PM
16	28	Sunday	12:00:06 AM	11:00:07 PM
17	29	Monday	12:00:08 AM	11:00:05 PM
18	30	Tuesday	12:00:05 AM	11:00:08 PM
19	31	Wednesday	12:00:01 AM	11:00:08 PM

My Observations and Theories:

1. Rainbow Hosiery operates every day, meaning the production demand is high and factory needs to run on most days to meet those demands.
2. On 13th Saturday and 24th Wednesday, work started at unsual time compared to rest, there could be many reasons like factory scheduled a half day, or maintenance activities, power outage, delay in raw material supply. Same assumptions for 23rd Tuesday for early timing.