



## **"Python Programming"**

### **Assignment-5(Capstone)**

**Topic** – End-to-End Energy Consumption Analysis and Visualisation

**Submitted by** – Meghna Kumar

**Roll no** - 2501730214

**Course** – B. Tech CSE (AI & ML)

**Section** – A

**Faculty name**- Mr Sameer Farooq

## **Introduction**

This project focuses on analysing electricity usage across multiple campus buildings. By automating data loading, cleaning, aggregation, and visualisation, the system helps identify usage patterns, peak hours, and high-consumption buildings. The goal is to support better energy management and decision-making.

## **Objectives**

- To read and validate building-wise energy CSV files automatically.
- To compute daily, weekly, and building-level energy statistics.
- To use object-oriented programming for structured data handling.
- To create a dashboard showing energy trends and comparisons.
- To generate summary files for administrative insights.

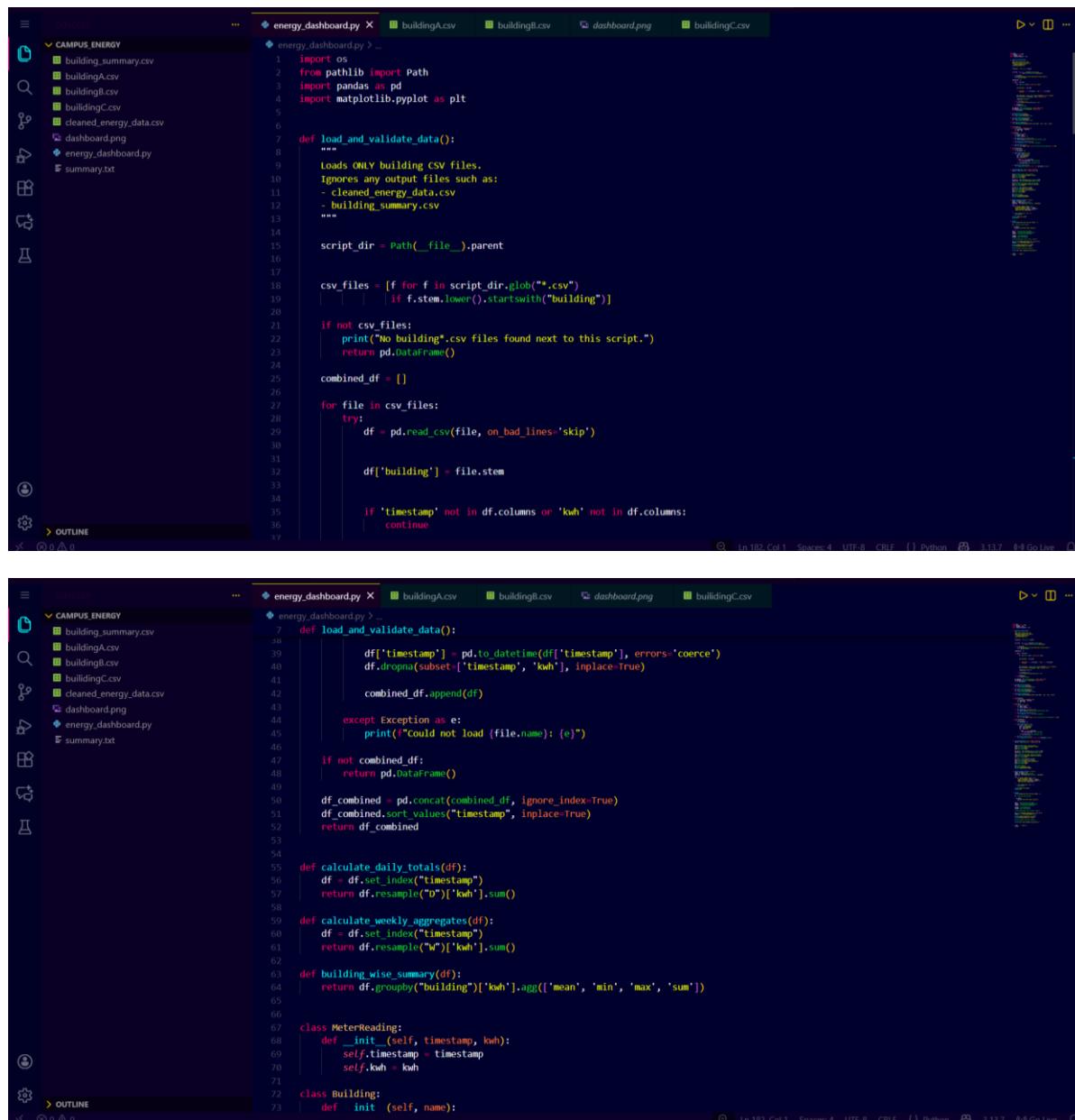
## **Program Description**

The Python script loads all building CSV files from the same folder, cleans the data, and combines them into a single dataset. It calculates daily and weekly totals using Pandas and summarises consumption for each building.

OOP classes are used to model buildings and meter readings. A dashboard is created using Matplotlib, containing a daily trend line, weekly comparison bar chart, and hourly scatter plot. The

program finally exports a cleaned dataset, building summary, and a text-based executive summary.

## Program Code

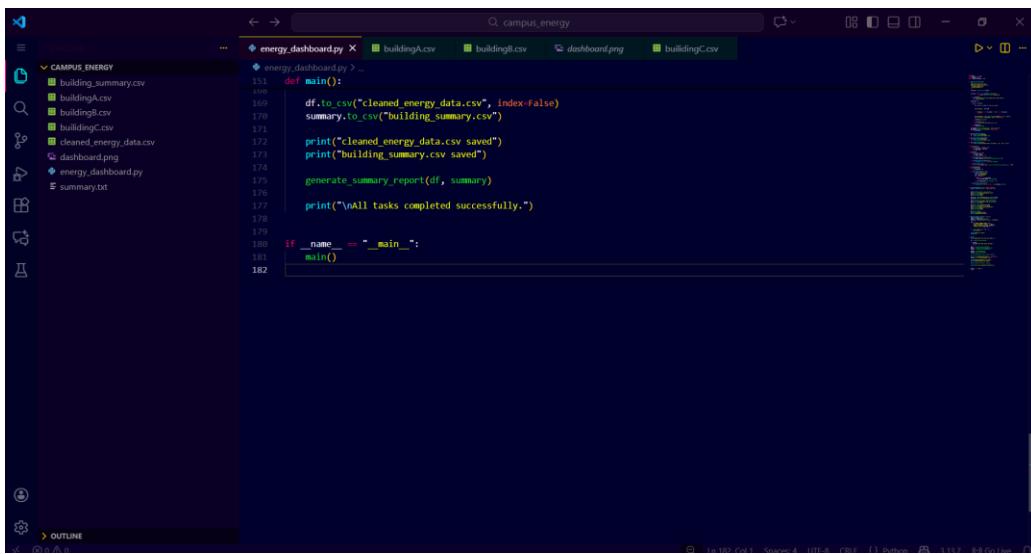


```
energy_dashboard.py X buildingA.csv buildingB.csv dashboard.png buildingC.csv
◆ energy_dashboard.py > ...
1 import os
2 from pathlib import Path
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 def load_and_validate_data():
7     """
8         Loads ONLY building CSV files.
9         Ignores any output files such as:
10        - cleaned_energy_data.csv
11        - building_summary.csv
12    """
13
14    script_dir = Path(__file__).parent
15
16    csv_files = [f for f in script_dir.glob("*.csv")
17                 if f.stem.lower().startswith("building")]
18
19    if not csv_files:
20        print("No building*.csv files found next to this script.")
21        return pd.DataFrame()
22
23    combined_df = []
24
25    for file in csv_files:
26        try:
27            df = pd.read_csv(file, on_bad_lines='skip')
28
29            df['building'] = file.stem
30
31            if 'timestamp' not in df.columns or 'kwh' not in df.columns:
32                continue
33
34
35        except Exception as e:
36            print(f"Could not load {file.name}: {e}")
37
38    if not combined_df:
39        return pd.DataFrame()
40
41    df_combined = pd.concat(combined_df, ignore_index=True)
42    df_combined.sort_values("timestamp", inplace=True)
43
44    combined_df.append(df)
45
46
47    def calculate_daily_totals(df):
48        df = df.set_index("timestamp")
49        return df.resample("D")['kwh'].sum()
50
51    def calculate_weekly_aggregates(df):
52        df = df.set_index("timestamp")
53        return df.resample("W")['kwh'].sum()
54
55    def building_wise_summary(df):
56        return df.groupby("building")['kwh'].agg(['mean', 'min', 'max', 'sum'])
57
58
59    class MeterReading:
60        def __init__(self, timestamp, kwh):
61            self.timestamp = timestamp
62            self.kwh = kwh
63
64    class Building:
65        def __init__(self, name):
66
67
68
69
70
71
72
73
```

```
... energy_dashboard.py ... campus_energy
...
1  #!/usr/bin/python
2
3  import pandas as pd
4
5  class Building:
6      def __init__(self, name):
7          self.name = name
8          self.meter_readings = []
9
10     def add_reading(self, timestamp, kwh):
11         self.meter_readings.append(MeterReading(timestamp, kwh))
12
13     def calculate_total_consumption(self):
14         return sum(r.kwh for r in self.meter_readings)
15
16     def generate_report(self):
17         return f"({self.name}): Total = {self.calculate_total_consumption():.2f} kWh"
18
19
20  class BuildingManager:
21      def __init__(self):
22          self.buildings = {}
23
24      def ingest_dataframe(self, df):
25          for _, row in df.iterrows():
26              name = row['building']
27              ts = row['timestamp']
28              kwh = row['kwh']
29
30              if name not in self.buildings:
31                  self.buildings[name] = Building(name)
32
33              self.buildings[name].add_reading(ts, kwh)
34
35      def generate_all_reports(self):
36          return [b.generate_report() for b in self.buildings.values()]
37
38
39  def create_dashboard(df, daily, weekly, summary):
40      fig, ax = plt.subplots(3, 1, figsize=(12, 16))
41
42      df['date'] = pd.to_datetime(df['timestamp'])
43      df['year'] = df['date'].dt.year
44      df['month'] = df['date'].dt.month
45      df['week'] = df['date'].dt.isocalendar().week
46
47      df['hour'] = df['timestamp'].dt.hour
48
49      ax[0].plot(df['date'], df['kwh'])
50      ax[0].set_title("Daily Consumption Trend")
51      ax[0].set_xlabel("Date")
52      ax[0].set_ylabel("kWh")
53
54      df['avg'] = df.groupby('building')['kwh'].mean()
55
56      ax[1].bar(df['year'], df['avg'])
57      ax[1].set_title("Avg Yearly Usage per Building")
58      ax[1].set_xlabel("Year")
59      ax[1].set_ylabel("kWh")
60
61      df['hour'] = df['timestamp'].dt.hour
62      df['hour'] = df['hour'].map(lambda x: x % 24)
63
64      ax[2].scatter(df['hour'], df['kwh'])
65      ax[2].set_title("Hourly Peak Consumption")
66      ax[2].set_xlabel("Hour")
67      ax[2].set_ylabel("kWh")
68
69      plt.tight_layout()
70      plt.savefig("dashboard.png")
71      print("Dashboard saved")
72
73
74  def generate_summary_report(df, summary):
75      total = df['kwh'].sum()
76      highest = summary['sum'].idxmax()
77      peak_time = df.loc[df['kwh'].idxmax(), 'timestamp']
78
79      text = (
80          "==== ENERGY SUMMARY REPORT ====\n"
81          f"Total Consumption: ({total:.2f}) kWh\n"
82          f"Highest Consuming Building: {highest}\n"
83          f"Peak Load Time: {peak_time}\n"
84      )
85
86      with open("summary.txt", "w") as f:
87          f.write(text)
88
89      print("Summary.txt saved")
90
91
92  def main():
93      print("Loading CSV files from this folder...")
94
95      df = load_and_validate_data()
96
97      if df.empty:
98          print("No valid CSV found. Exiting.")
99          return
100
101      daily = calculate_daily_totals(df)
102      weekly = calculate_weekly_aggregates(df)
103      summary = building_wise_summary(df)
104
105      manager = BuildingManager()
106      manager.ingest_dataframe(df)
107
108      create_dashboard(df, daily, weekly, summary)
109
110      df.to_csv("cleaned_energy_data.csv", index=False)
111      summary.to_csv("building_summary.csv")
112
113
114
```

```
... energy_dashboard.py ... campus_energy
...
1  #!/usr/bin/python
2
3  import pandas as pd
4
5  class Building:
6      def __init__(self, name):
7          self.name = name
8          self.meter_readings = []
9
10     def add_reading(self, timestamp, kwh):
11         self.meter_readings.append(MeterReading(timestamp, kwh))
12
13     def calculate_total_consumption(self):
14         return sum(r.kwh for r in self.meter_readings)
15
16     def generate_report(self):
17         return f"({self.name}): Total = {self.calculate_total_consumption():.2f} kWh"
18
19
20  class BuildingManager:
21      def __init__(self):
22          self.buildings = {}
23
24      def ingest_dataframe(self, df):
25          for _, row in df.iterrows():
26              name = row['building']
27              ts = row['timestamp']
28              kwh = row['kwh']
29
30              if name not in self.buildings:
31                  self.buildings[name] = Building(name)
32
33              self.buildings[name].add_reading(ts, kwh)
34
35      def generate_all_reports(self):
36          return [b.generate_report() for b in self.buildings.values()]
37
38
39  def create_dashboard(df, daily, weekly, summary):
40      fig, ax = plt.subplots(3, 1, figsize=(12, 16))
41
42      df['date'] = pd.to_datetime(df['timestamp'])
43      df['year'] = df['date'].dt.year
44      df['month'] = df['date'].dt.month
45      df['week'] = df['date'].dt.isocalendar().week
46
47      df['hour'] = df['timestamp'].dt.hour
48
49      ax[0].plot(df['date'], df['kwh'])
50      ax[0].set_title("Daily Consumption Trend")
51      ax[0].set_xlabel("Date")
52      ax[0].set_ylabel("kWh")
53
54      df['avg'] = df.groupby('building')['kwh'].mean()
55
56      ax[1].bar(df['year'], df['avg'])
57      ax[1].set_title("Avg Yearly Usage per Building")
58      ax[1].set_xlabel("Year")
59      ax[1].set_ylabel("kWh")
60
61      df['hour'] = df['timestamp'].dt.hour
62      df['hour'] = df['hour'].map(lambda x: x % 24)
63
64      ax[2].scatter(df['hour'], df['kwh'])
65      ax[2].set_title("Hourly Peak Consumption")
66      ax[2].set_xlabel("Hour")
67      ax[2].set_ylabel("kWh")
68
69      plt.tight_layout()
70      plt.savefig("dashboard.png")
71      print("Dashboard saved")
72
73
74  def generate_summary_report(df, summary):
75      total = df['kwh'].sum()
76      highest = summary['sum'].idxmax()
77      peak_time = df.loc[df['kwh'].idxmax(), 'timestamp']
78
79      text = (
80          "==== ENERGY SUMMARY REPORT ====\n"
81          f"Total Consumption: ({total:.2f}) kWh\n"
82          f"Highest Consuming Building: {highest}\n"
83          f"Peak Load Time: {peak_time}\n"
84      )
85
86      with open("summary.txt", "w") as f:
87          f.write(text)
88
89      print("Summary.txt saved")
90
91
92  def main():
93      print("Loading CSV files from this folder...")
94
95      df = load_and_validate_data()
96
97      if df.empty:
98          print("No valid CSV found. Exiting.")
99          return
100
101      daily = calculate_daily_totals(df)
102      weekly = calculate_weekly_aggregates(df)
103      summary = building_wise_summary(df)
104
105      manager = BuildingManager()
106      manager.ingest_dataframe(df)
107
108      create_dashboard(df, daily, weekly, summary)
109
110      df.to_csv("cleaned_energy_data.csv", index=False)
111      summary.to_csv("building_summary.csv")
112
113
114
```

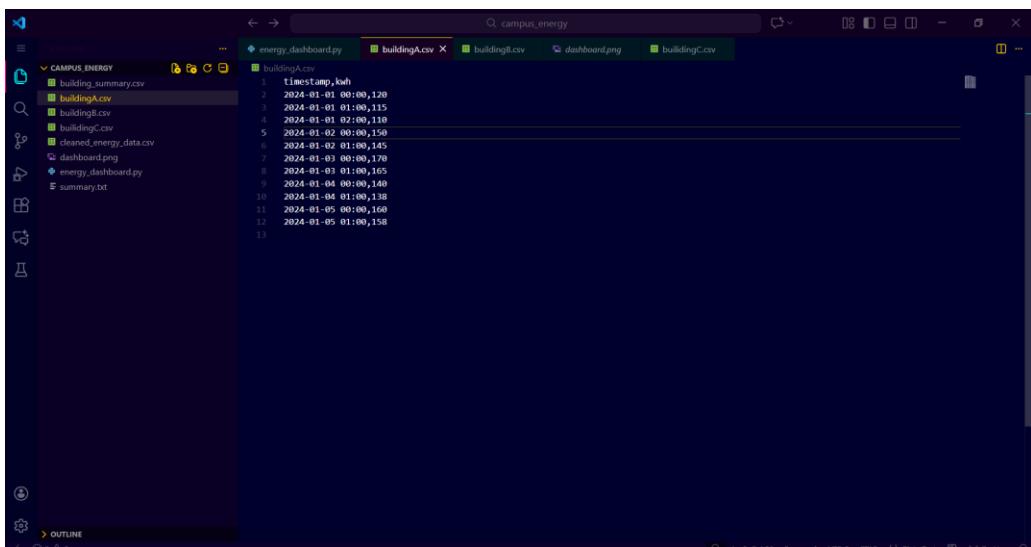
```
... energy_dashboard.py ... campus_energy
...
1  #!/usr/bin/python
2
3  import pandas as pd
4
5  class Building:
6      def __init__(self, name):
7          self.name = name
8          self.meter_readings = []
9
10     def add_reading(self, timestamp, kwh):
11         self.meter_readings.append(MeterReading(timestamp, kwh))
12
13     def calculate_total_consumption(self):
14         return sum(r.kwh for r in self.meter_readings)
15
16     def generate_report(self):
17         return f"({self.name}): Total = {self.calculate_total_consumption():.2f} kWh"
18
19
20  class BuildingManager:
21      def __init__(self):
22          self.buildings = {}
23
24      def ingest_dataframe(self, df):
25          for _, row in df.iterrows():
26              name = row['building']
27              ts = row['timestamp']
28              kwh = row['kwh']
29
30              if name not in self.buildings:
31                  self.buildings[name] = Building(name)
32
33              self.buildings[name].add_reading(ts, kwh)
34
35      def generate_all_reports(self):
36          return [b.generate_report() for b in self.buildings.values()]
37
38
39  def calculate_daily_totals(df):
40      daily = df.groupby('date').sum()
41      daily['date'] = pd.to_datetime(daily.index)
42
43      return daily
44
45
46  def calculate_weekly_aggregates(df):
47      weekly = df.groupby('week').sum()
48
49      return weekly
50
51
52  def building_wise_summary(df):
53      summary = df.groupby('building').sum()
54
55      return summary
56
57
58  def generate_summary_report(df, summary):
59      total = df['kwh'].sum()
60      highest = summary['sum'].idxmax()
61      peak_time = df.loc[df['kwh'].idxmax(), 'timestamp']
62
63      text = (
64          "==== ENERGY SUMMARY REPORT ====\n"
65          f"Total Consumption: ({total:.2f}) kWh\n"
66          f"Highest Consuming Building: {highest}\n"
67          f"Peak Load Time: {peak_time}\n"
68      )
69
70      with open("summary.txt", "w") as f:
71          f.write(text)
72
73      print("Summary.txt saved")
74
75
76  def main():
77      print("Loading CSV files from this folder...")
78
79      df = load_and_validate_data()
80
81      if df.empty:
82          print("No valid CSV found. Exiting.")
83          return
84
85      daily = calculate_daily_totals(df)
86      weekly = calculate_weekly_aggregates(df)
87      summary = building_wise_summary(df)
88
89      manager = BuildingManager()
90      manager.ingest_dataframe(df)
91
92      create_dashboard(df, daily, weekly, summary)
93
94      df.to_csv("cleaned_energy_data.csv", index=False)
95      summary.to_csv("building_summary.csv")
96
97
98
```



```
campus.energy
energy_dashboard.py
buildingA.csv
buildingB.csv
buildingC.csv
cleaned_energy_data.csv
dashboard.png
energy.dashboard.py
summary.txt

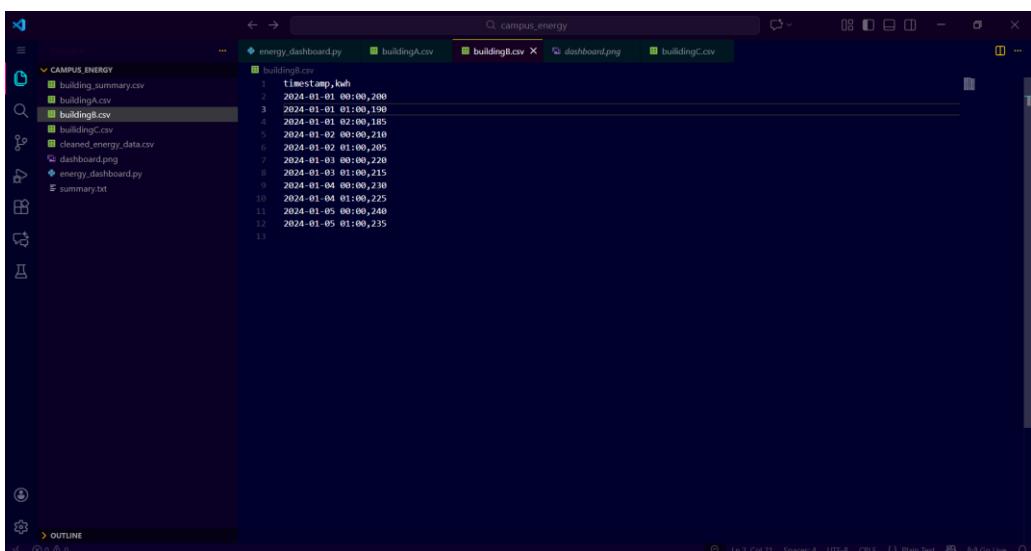
151     def main():
152         df.to_csv("cleaned_energy_data.csv", index=False)
153         summary.to_csv("building_summary.csv")
154
155         print("cleaned energy data.csv saved")
156         print("building_summary.csv saved")
157
158         generate_summary_report(df, summary)
159
160         print("\nAll tasks completed successfully.")
161
162 if __name__ == "__main__":
163     main()

164
```



```
campus.energy
buildingA.csv
buildingB.csv
buildingC.csv
cleaned_energy_data.csv
dashboard.png
energy.dashboard.py
summary.txt

1 timestamp,kwh
2 2024-01-01 00:00,120
3 2024-01-01 01:00,115
4 2024-01-01 02:00,110
5 2024-01-02 00:00,150
6 2024-01-02 01:00,145
7 2024-01-03 00:00,170
8 2024-01-03 01:00,165
9 2024-01-04 00:00,140
10 2024-01-04 01:00,138
11 2024-01-05 00:00,160
12 2024-01-05 01:00,158
13
```



```
campus.energy
buildingA.csv
buildingB.csv
buildingC.csv
cleaned_energy_data.csv
dashboard.png
energy.dashboard.py
summary.txt

1 timestamp,kwh
2 2024-01-01 00:00,200
3 2024-01-01 01:00,190
4 2024-01-01 02:00,185
5 2024-01-02 00:00,210
6 2024-01-02 01:00,205
7 2024-01-03 00:00,220
8 2024-01-03 01:00,215
9 2024-01-04 00:00,230
10 2024-01-04 01:00,225
11 2024-01-05 00:00,240
12 2024-01-05 01:00,235
13
```

```

1 timestamp,kwh
2 2024-01-01 00:00:00
3 2024-01-01 01:00:00
4 2024-01-01 02:00:00
5 2024-01-01 03:00:00
6 2024-01-01 04:00:00
7 2024-01-01 05:00:00
8 2024-01-01 06:00:00
9 2024-01-01 07:00:00
10 2024-01-01 08:00:00
11 2024-01-01 09:00:00
12 2024-01-01 10:00:00
13 2024-01-01 11:00:00
14 2024-01-01 12:00:00
15 2024-01-01 13:00:00
16 2024-01-01 14:00:00
17 2024-01-01 15:00:00
18 2024-01-01 16:00:00
19 2024-01-01 17:00:00
20 2024-01-01 18:00:00
21 2024-01-01 19:00:00
22 2024-01-01 20:00:00
23 2024-01-01 21:00:00
24 2024-01-01 22:00:00
25 2024-01-01 23:00:00
26 2024-01-02 00:00:00
27 2024-01-02 01:00:00
28 2024-01-02 02:00:00
29 2024-01-02 03:00:00
30 2024-01-02 04:00:00
31 2024-01-02 05:00:00
32 2024-01-02 06:00:00
33 2024-01-02 07:00:00
34 2024-01-02 08:00:00
35 2024-01-02 09:00:00
36 2024-01-02 10:00:00
37 2024-01-02 11:00:00
38 2024-01-02 12:00:00
39 2024-01-02 13:00:00
40 2024-01-02 14:00:00
41 2024-01-02 15:00:00
42 2024-01-02 16:00:00
43 2024-01-02 17:00:00
44 2024-01-02 18:00:00
45 2024-01-02 19:00:00
46 2024-01-02 20:00:00
47 2024-01-02 21:00:00
48 2024-01-02 22:00:00
49 2024-01-02 23:00:00
50 2024-01-03 00:00:00
51 2024-01-03 01:00:00
52 2024-01-03 02:00:00
53 2024-01-03 03:00:00
54 2024-01-03 04:00:00
55 2024-01-03 05:00:00
56 2024-01-03 06:00:00
57 2024-01-03 07:00:00
58 2024-01-03 08:00:00
59 2024-01-03 09:00:00
60 2024-01-03 10:00:00
61 2024-01-03 11:00:00
62 2024-01-03 12:00:00
63 2024-01-03 13:00:00
64 2024-01-03 14:00:00
65 2024-01-03 15:00:00
66 2024-01-03 16:00:00
67 2024-01-03 17:00:00
68 2024-01-03 18:00:00
69 2024-01-03 19:00:00
70 2024-01-03 20:00:00
71 2024-01-03 21:00:00
72 2024-01-03 22:00:00
73 2024-01-03 23:00:00
74 2024-01-04 00:00:00
75 2024-01-04 01:00:00
76 2024-01-04 02:00:00
77 2024-01-04 03:00:00
78 2024-01-04 04:00:00
79 2024-01-04 05:00:00
80 2024-01-04 06:00:00
81 2024-01-04 07:00:00
82 2024-01-04 08:00:00
83 2024-01-04 09:00:00
84 2024-01-04 10:00:00
85 2024-01-04 11:00:00
86 2024-01-04 12:00:00
87 2024-01-04 13:00:00
88 2024-01-04 14:00:00
89 2024-01-04 15:00:00
90 2024-01-04 16:00:00
91 2024-01-04 17:00:00
92 2024-01-04 18:00:00
93 2024-01-04 19:00:00
94 2024-01-04 20:00:00
95 2024-01-04 21:00:00
96 2024-01-04 22:00:00
97 2024-01-04 23:00:00
98 2024-01-05 00:00:00
99 2024-01-05 01:00:00
100 2024-01-05 02:00:00
101 2024-01-05 03:00:00
102 2024-01-05 04:00:00
103 2024-01-05 05:00:00

```

## Sample Output

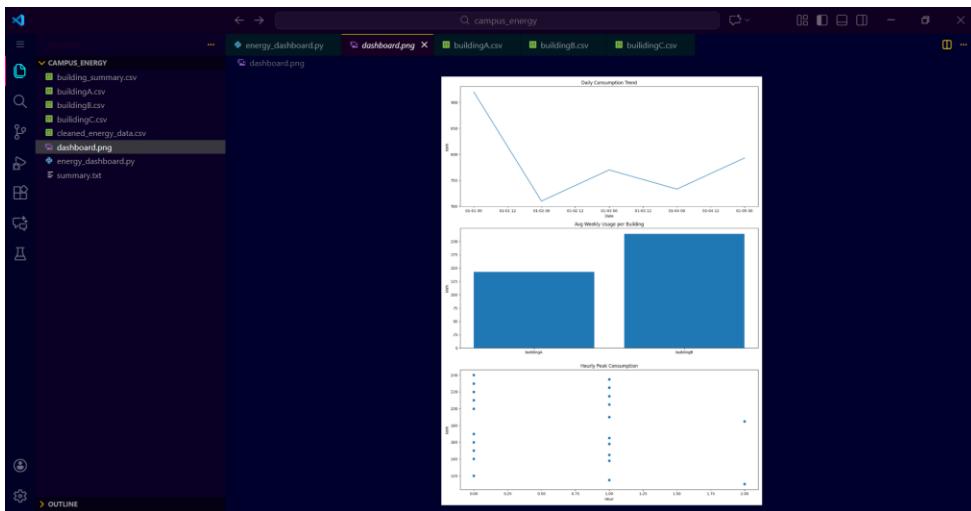
```

PS C:\Users\ip\Desktop\python_programming\campus_energy> & C:/Users/Ip/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Ip/Desktop/python_programming/campus_energy/energy_dashboard.py"
loading CSV files from this folder...
✓ dashboard.prj saved
✓ buildingA.csv saved
✓ buildingB.csv saved
✓ buildingC.csv saved
summary.txt saved
--- ENERGY SUMMARY REPORT ---
Total Consumption: 1000 kWh
Highest Consuming Building: buildingB
Peak Load Time: 2024-01-05 00:00:00

All tasks completed successfully.

PS C:\Users\ip\Desktop\python_programming\campus_energy>

```



The screenshot shows a code editor window with the title bar "campus energy". The left sidebar shows a project structure under "CAMPUS ENERGY" with files like "building\_summary.csv", "buildingA.csv", "buildingB.csv", "buildingC.csv", "cleaned\_energy\_data.csv", "dashboard.png", "energy\_dashboard.py", and "summary.txt". The main pane displays the content of "cleaned\_energy\_data.csv":

```
1 timestamp,kwh,building,week,hour
2 2024-01-01 00:00:00,120,buildingA,1,0
3 2024-01-01 00:00:00,200,buildingB,1,0
4 2024-01-01 01:00:00,115,buildingA,1,1
5 2024-01-01 01:00:00,196,buildingB,1,1
6 2024-01-01 02:00:00,110,buildingA,1,2
7 2024-01-01 02:00:00,185,buildingB,1,2
8 2024-01-02 00:00:00,150,buildingA,1,0
9 2024-01-02 00:00:00,210,buildingB,1,0
10 2024-01-02 01:00:00,105,buildingA,1,1
11 2024-01-02 01:00:00,205,buildingB,1,1
12 2024-01-03 00:00:00,170,buildingA,1,0
13 2024-01-03 00:00:00,220,buildingB,1,0
14 2024-01-03 01:00:00,165,buildingA,1,1
15 2024-01-03 01:00:00,215,buildingB,1,1
16 2024-01-04 00:00:00,140,buildingA,1,0
17 2024-01-04 00:00:00,230,buildingB,1,0
18 2024-01-04 01:00:00,225,buildingA,1,1
19 2024-01-04 01:00:00,138,buildingB,1,1
20 2024-01-05 00:00:00,240,buildingA,1,0
21 2024-01-05 00:00:00,160,buildingB,1,0
22 2024-01-05 01:00:00,158,buildingA,1,1
23 2024-01-05 01:00:00,235,buildingB,1,1
24
```

The screenshot shows a code editor window with the title bar "campus energy". The left sidebar shows a project structure under "CAMPUS ENERGY" with files like "building\_summary.csv", "buildingA.csv", "buildingB.csv", "buildingC.csv", "cleaned\_energy\_data.csv", "dashboard.png", "energy\_dashboard.py", and "summary.txt". The main pane displays the content of "building\_summary.csv":

```
1 building,mean,min,max,sum
2 buildingA,142.8181818181816,110,170,1571
3 buildingB,214.0909090909091,185,240,2355
4
```

The screenshot shows a code editor window with the title bar "campus energy". The left sidebar shows a project structure under "CAMPUS ENERGY" with files like "building\_summary.csv", "buildingA.csv", "buildingB.csv", "buildingC.csv", "cleaned\_energy\_data.csv", "dashboard.png", "energy\_dashboard.py", and "summary.txt". The main pane displays the content of "summary.txt":

```
1 == ENERGY SUMMARY REPORT ==
2 Total Consumption: 3926.00 kWh
3 Highest Consuming Building: buildingB
4 Peak Load Time: 2024-01-05 00:00:00
5
```

## **Conclusion**

The project successfully analyzes campus electricity usage and produces clear visual and textual insights. It automates data processing, identifies key consumption trends, and helps administrators make informed decisions for energy conservation.