



“Python Programming”

Assignment-5(Capstone)

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

Submitted by – Priyam Sharma

Roll no - 2501730184

Course – B. Tech CSE (AI & ML)

Section –B

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.

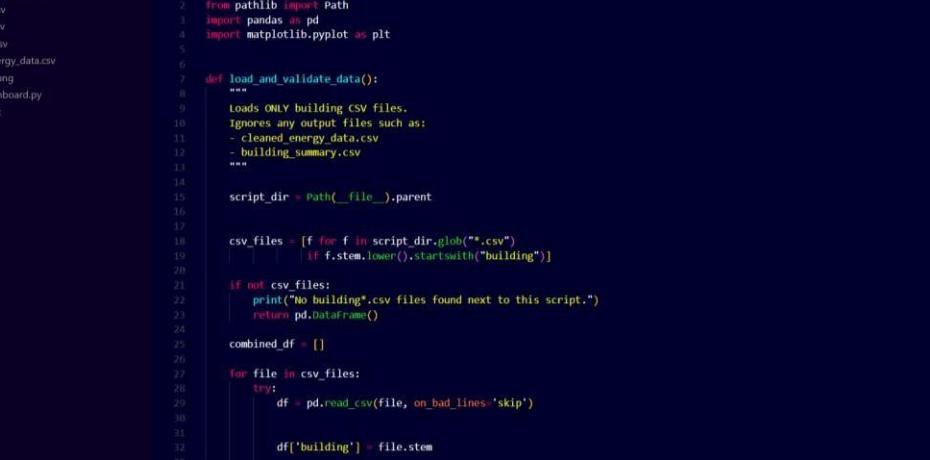
Program Description

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



The screenshot shows a Jupyter Notebook environment with the following details:

- File Explorer:** On the left, it shows a tree view of files under the folder "CAMPUS_ENERGY". The visible files include "building_summary.csv", "buildingA.csv", "buildingB.csv", "buildingC.csv", "buildings", "cleaned_energy_data.csv", "dashboard.png", "energy_dashboard.py", and "summary.txt".
- Code Editor:** The main area displays a Python script named "energy_dashboard.py". The code is as follows:

```
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
17    csv_files = [f for f in script_dir.glob("*.csv")
18                if f.stem.lower().startswith("building")]
19
20    if not csv_files:
21        print("No building*.csv files found next to this script.")
22        return pd.DataFrame()
23
24    combined_df = []
25
26    for file in csv_files:
27        try:
28            df = pd.read_csv(file, on_bad_lines='skip')
29
30            df['building'] = file.stem
31
32            if 'timestamp' not in df.columns or 'kwh' not in df.columns:
33                continue
34
35        except:
36            continue
37
38    return combined_df
```

The notebook interface includes standard Jupyter controls like back/forward, search, and help, along with a status bar at the bottom indicating the current cell, column, and row.



The screenshot shows a Jupyter Notebook environment with several tabs open at the top: 'energy_dashboard.py x', 'buildingA.csv', 'buildingB.csv', 'dashboard.png', and 'buildingC.csv'. The main area displays Python code for data processing:

```
...  
    def load_and_validate_data():  
        df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')  
        df.dropna(subset=['timestamp', 'kwh'], inplace=True)  
  
        combined_df.append(df)  
  
    except Exception as e:  
        print(f"Could not load {file.name}: {e}")  
  
if not combined_df:  
    return pd.DataFrame()  
  
df_combined = pd.concat(combined_df, ignore_index=True)  
df_combined.sort_values("timestamp", inplace=True)  
return df_combined  
  
def calculate_daily_totals(df):  
    df = df.set_index("timestamp")  
    return df.resample("D")['kwh'].sum()  
  
def calculate_weekly_aggregates(df):  
    df = df.set_index("timestamp")  
    return df.resample("W")['kwh'].sum()  
  
def building_wise_summary(df):  
    return df.groupby("building")['kwh'].agg(['mean', 'min', 'max', 'sum'])  
  
class MeterReading:  
    def __init__(self, timestamp, kwh):  
        self.timestamp = timestamp  
        self.kwh = kwh  
  
class Building:  
    def __init__(self, name):
```

```
class Building:
    def __init__(self, name):
        self.name = name
        self.meter_readings = []

    def add_reading(self, timestamp, kwh):
        self.meter_readings.append(MeterReading(timestamp, kwh))

    def calculate_total_consumption(self):
        return sum(r.kwh for r in self.meter_readings)

    def generate_report(self):
        return f"[{self.name}]: Total = {self.calculate_total_consumption()} kWh"

class BuildingManager:
    def __init__(self):
        self.buildings = {}

    def ingest_dataframe(self, df):
        for _, row in df.iterrows():
            name = row['building']
            ts = row['timestamp']
            kwh = row['kwh']

            if name not in self.buildings:
                self.buildings[name] = Building(name)

            self.buildings[name].add_reading(ts, kwh)

    def generate_all_reports(self):
        return [b.generate_report() for b in self.buildings.values()]

def create_dashboard(df, daily, weekly, summary):
    fig, ax = plt.subplots(3, 1, figsize=(12, 16))

    ax[0].plot(daily.index, daily.values)
    ax[0].set_title("Daily Consumption Trend")
    ax[0].set_xlabel("Date")
    ax[0].set_ylabel("kWh")

    df['week'] = df['timestamp'].dt.isocalendar().week
    weekly_avg = df.groupby('building')['kwh'].mean()

    ax[1].bar(weekly_avg.index, weekly_avg.values)
    ax[1].set_title("Avg Weekly Usage per Building")
    ax[1].set_xlabel("Building")
    ax[1].set_ylabel("kWh")

    df['hour'] = df['timestamp'].dt.hour
    ax[2].scatter(df['hour'], df['kwh'])
    ax[2].set_title("Hourly Peak Consumption")
    ax[2].set_xlabel("Hour")
    ax[2].set_ylabel("kWh")

    plt.tight_layout()
    plt.savefig("dashboard.png")
    print("Dashboard saved")

def generate_summary_report(df, summary):
    total = df['kwh'].sum()
    highest = summary['sum'].idxmax()
    peak_time = df.loc[df['kwh'].idxmax(), 'timestamp']

    text = (
        "==== ENERGY SUMMARY REPORT ====\n"
        f"Total Consumption: {total:.2f} kWh\n"
        f"Highest Consuming Building: {highest}\n"
        f"Peak Load Time: {peak_time}\n"
    )

    with open("summary.txt", "w") as f:
        f.write(text)

    print("Summary saved")
```

```
def create_dashboard(df, daily, weekly, summary):
    fig, ax = plt.subplots(3, 1, figsize=(12, 16))

    ax[0].plot(daily.index, daily.values)
    ax[0].set_title("Daily Consumption Trend")
    ax[0].set_xlabel("Date")
    ax[0].set_ylabel("kWh")

    df['week'] = df['timestamp'].dt.isocalendar().week
    weekly_avg = df.groupby('building')['kwh'].mean()

    ax[1].bar(weekly_avg.index, weekly_avg.values)
    ax[1].set_title("Avg Weekly Usage per Building")
    ax[1].set_xlabel("Building")
    ax[1].set_ylabel("kWh")

    df['hour'] = df['timestamp'].dt.hour
    ax[2].scatter(df['hour'], df['kwh'])
    ax[2].set_title("Hourly Peak Consumption")
    ax[2].set_xlabel("Hour")
    ax[2].set_ylabel("kWh")

    plt.tight_layout()
    plt.savefig("dashboard.png")
    print("Dashboard saved")

def generate_summary_report(df, summary):
    total = df['kwh'].sum()
    highest = summary['sum'].idxmax()
    peak_time = df.loc[df['kwh'].idxmax(), 'timestamp']

    text = (
        "==== ENERGY SUMMARY REPORT ====\n"
        f"Total Consumption: {total:.2f} kWh\n"
        f"Highest Consuming Building: {highest}\n"
        f"Peak Load Time: {peak_time}\n"
    )

    with open("summary.txt", "w") as f:
        f.write(text)

    print("Summary saved")
```

```
def generate_summary_report(df, summary):
    text = (
        "==== ENERGY SUMMARY REPORT ====\n"
        f"Total Consumption: {total:.2f} kWh\n"
        f"Highest Consuming Building: {highest}\n"
        f"Peak Load Time: {peak_time}\n"
    )

    with open("summary.txt", "w") as f:
        f.write(text)

    print("Summary saved")
```

```
def main():
    print("Loading CSV files from this folder...")

    df = load_and_validate_data()

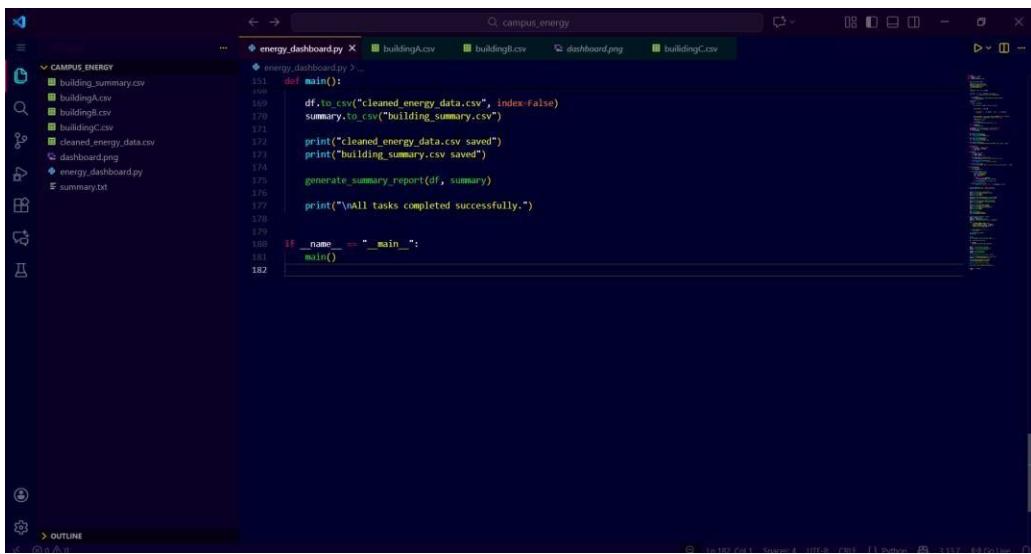
    if df.empty:
        print("No valid CSV found. Exiting.")
        return

    daily = calculate_daily_totals(df)
    weekly = calculate_weekly_aggregates(df)
    summary = building_wise_summary(df)

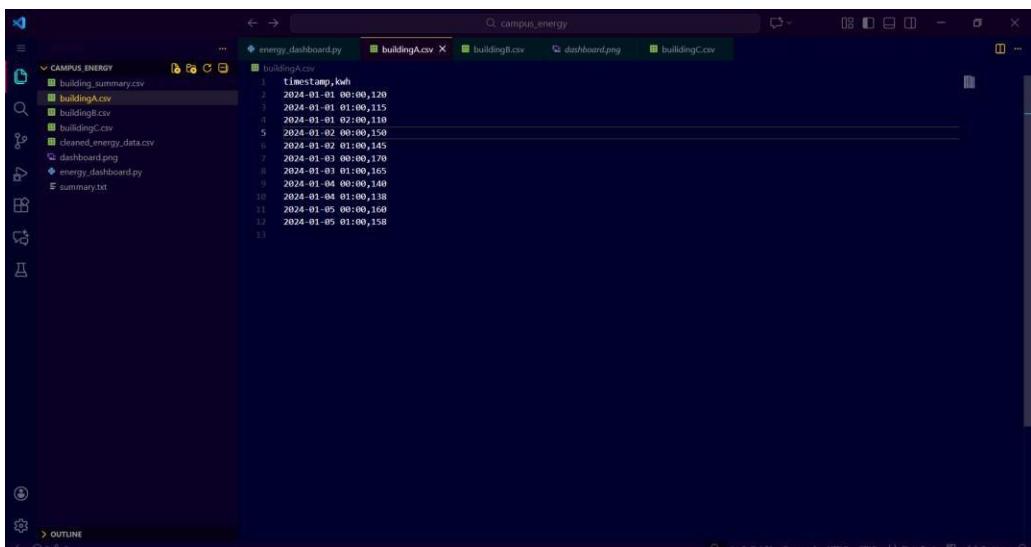
    manager = BuildingManager()
    manager.ingest_dataframe(df)

    create_dashboard(df, daily, weekly, summary)

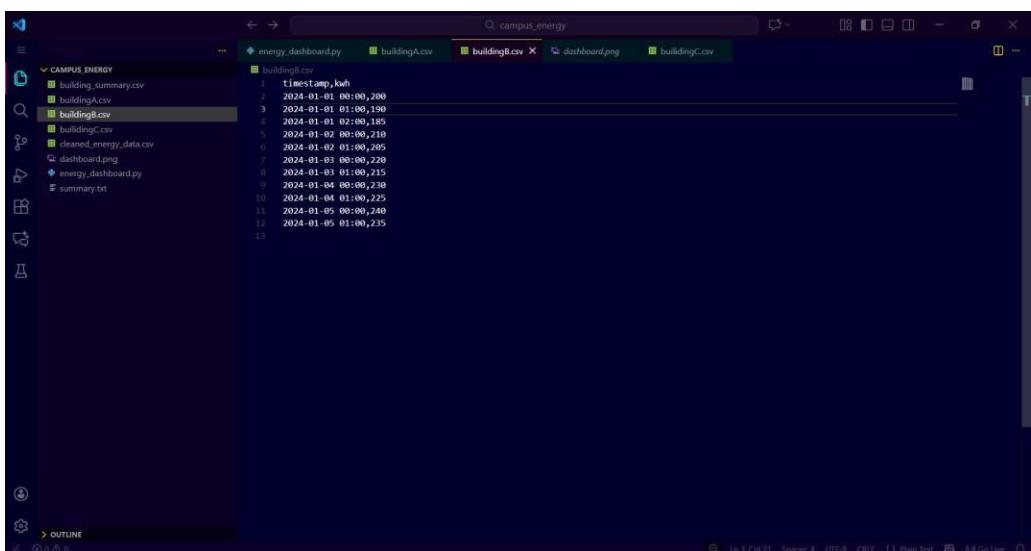
    df.to_csv("cleaned_energy_data.csv", index=False)
    summary.to_csv("building_summary.csv")
```



```
energy_dashboard.py
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
```



```
buildingA.csv
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
```



```
buildingB.csv
1 timestamp,kwh
2 2024-01-01 00:00,200
3 2024-01-01 01:00,190
4 2024-01-01 02:00,180
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 00:00,80
3 2024-01-01 01:00,98
4 2024-01-01 02:00,75
5 2024-01-02 00:00,90
6 2024-01-02 01:00,88
7 2024-01-01 06:00,95
8 2024-01-03 01:00,92
9 2024-01-04 00:00,100
10 2024-01-04 01:00,98
11 2024-01-05 00:00,105
12 2024-01-05 01:00,103

```

Sample Output

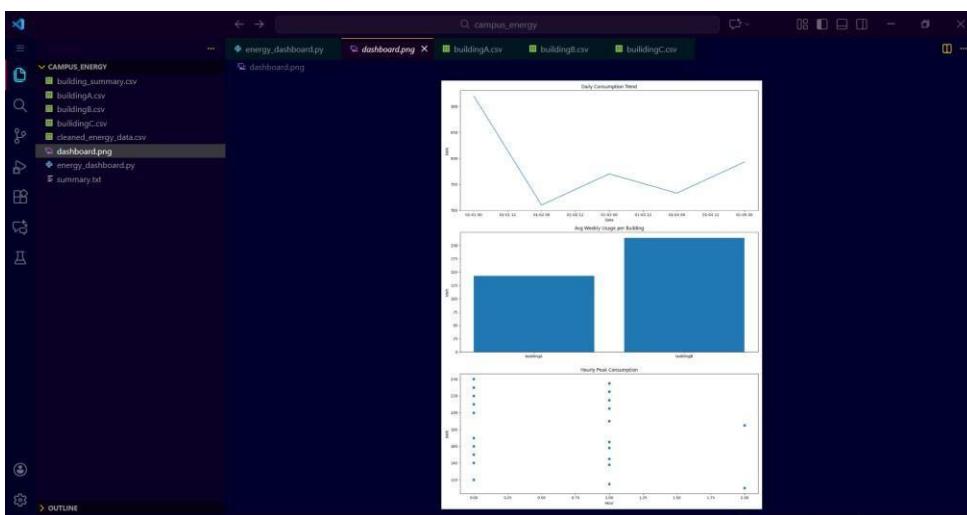
```

PS C:\Users\hp\Desktop\python programming\campus_energy> & C:/Users/hp/appData/local/Programs/Python/Python311/python.exe "c:/Users/hp/Desktop/python programming/campus_energy/energy_dashboard.py"
Loading CSV Files from this folder...
✓ dashboard.png saved
✓ buildingA.csv saved
✓ buildingG.csv saved
✓ buildingL.csv saved
✓ buildingC.csv saved
building_summary.csv saved
summary.txt saved
== ENERGY SUMMARY REPORT ==
Total Consumption: 2000.00 kWh
Highest Consumption Building: buildingG
Peak Load Time: 2024-01-05 00:00:00

All tasks completed successfully.

PS C:\Users\hp\Desktop\python programming\campus_energy>

```



The screenshot shows the VS Code interface with the 'campus_energy' workspace open. The 'cleaned_energy_data.csv' file is the active editor, displaying a list of timestamped energy consumption data for buildings A and B across four weeks. The data includes columns for timestamp, kWh, building, week, and hour.

```
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 01:00:00,130,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 01:00:00,150,buildingA,1,1
23 2024-01-05 01:00:00,235,buildingB,1,1
```

The screenshot shows the VS Code interface with the 'campus_energy' workspace open. The 'building_summary.csv' file is the active editor, displaying summary statistics for buildings A and B. The data includes columns for building, mean, min, max, and sum.

```
1 building,mean,min,max,sum
2 buildingA,142.8181818181818,110,170,1571
3 buildingB,214.0909090909091,185,240,2355
```

The screenshot shows the VS Code interface with the 'campus_energy' workspace open. The 'summary.txt' file is the active editor, displaying a text-based summary report. The report includes the total consumption, the highest consuming building, and the peak load time.

```
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
```

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.