“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.

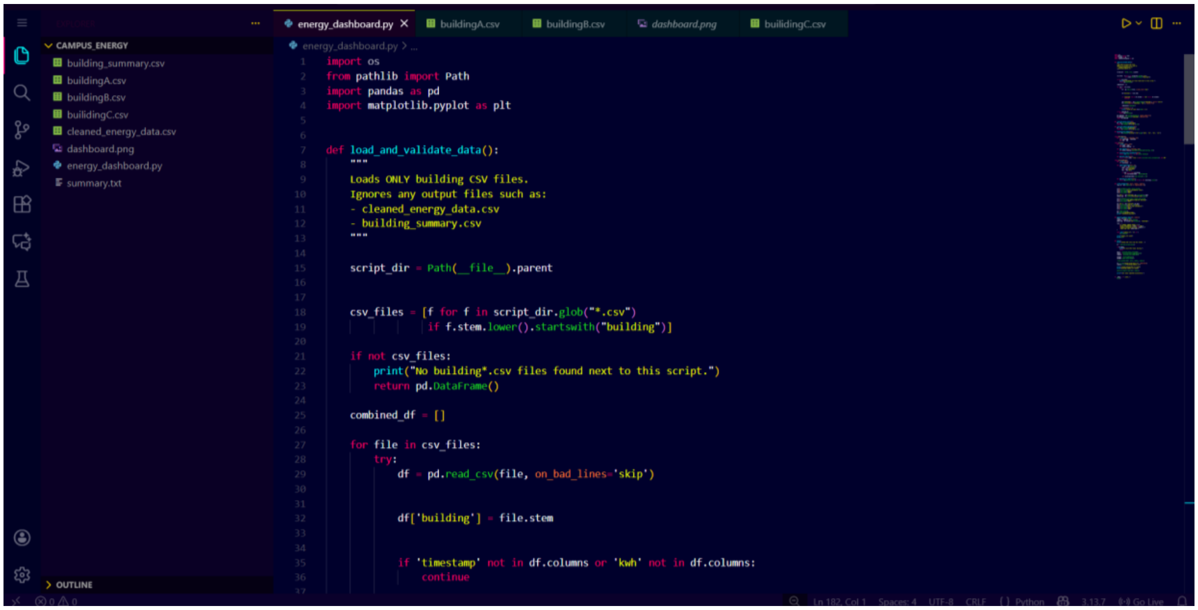
-To generate summary files for administrative insights.

Program Description

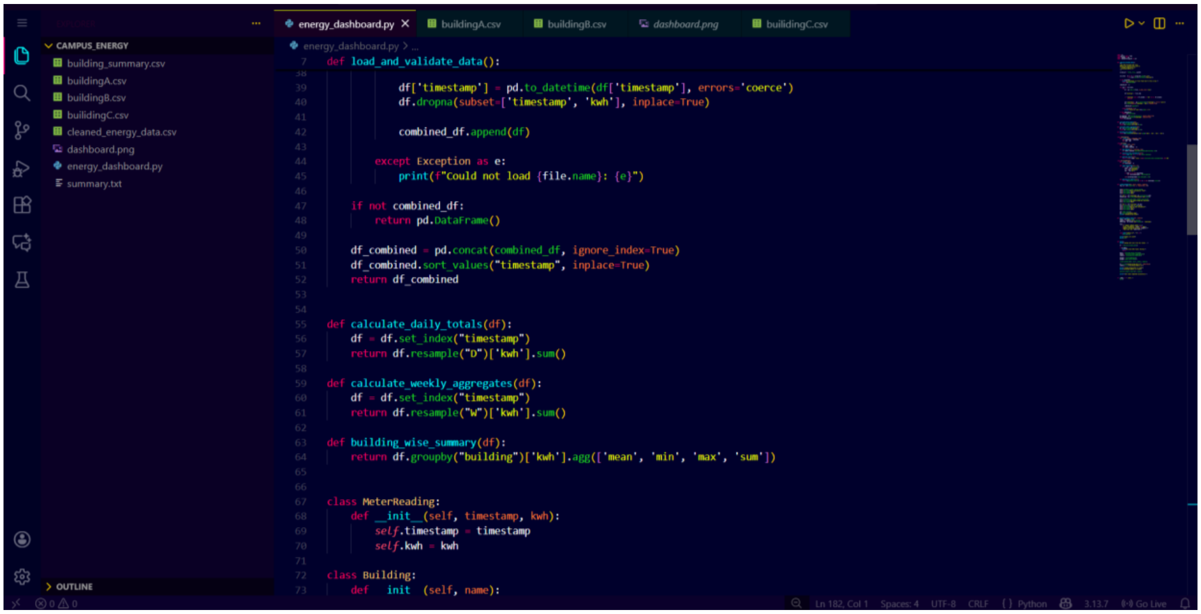
The Python script loads all building CSV files from the same folder, cleans the data, and combines them into a single dataset. It calculates daily and weekly totals using Pandas and summarises consumption for each building.

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.

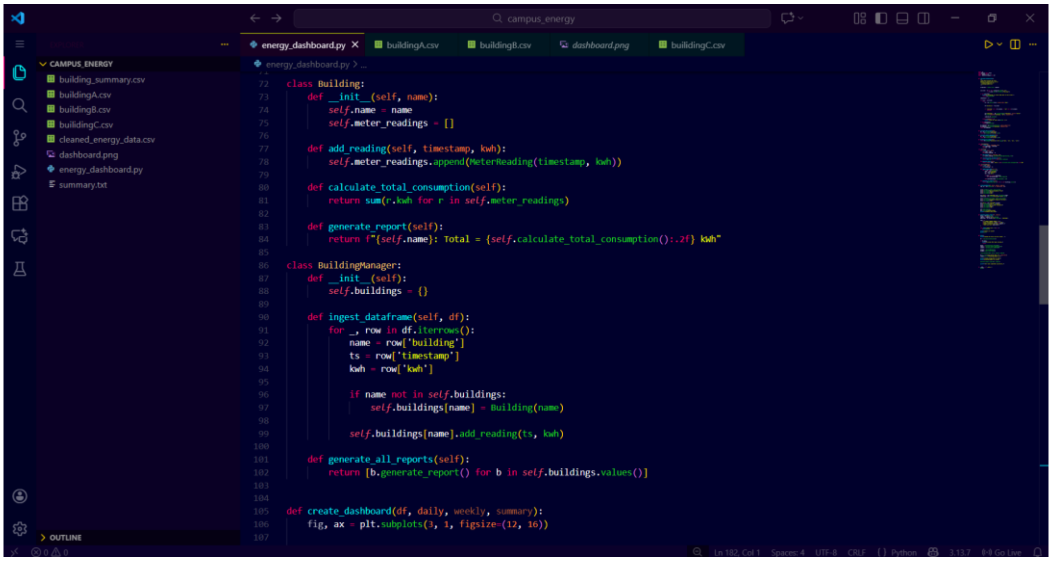
Program Code



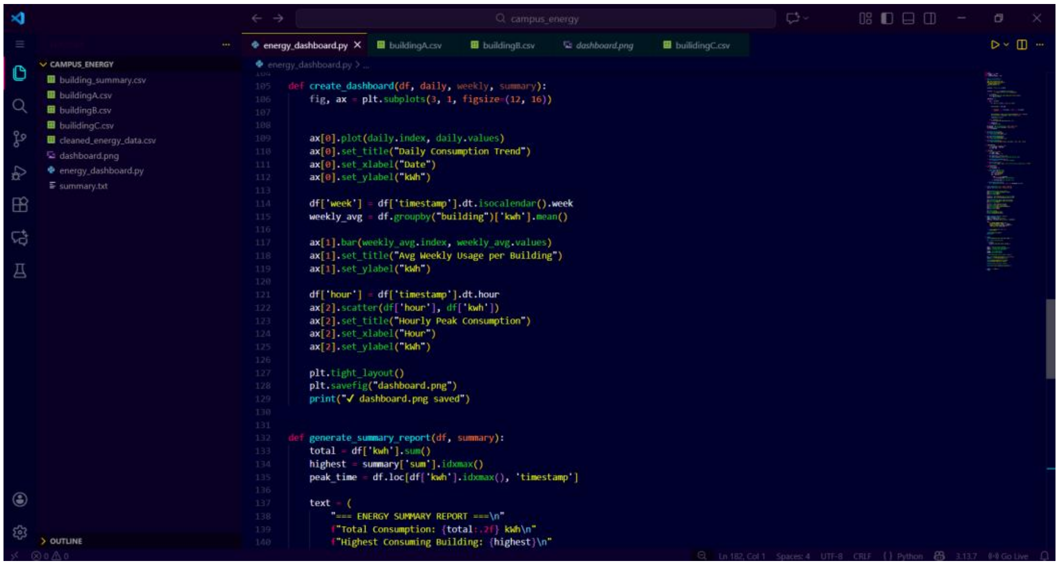
```
import os
from pathlib import Path
import pandas as pd
import matplotlib.pyplot as plt


def load_and_validate_data():
    """
    Loads ONLY building CSV files.
    Ignores any output files such as:
    - cleaned_energy_data.csv
    - building_summary.csv
    """

    script_dir = Path(__file__).parent


    csv_files = [f for f in script_dir.glob("*.csv")
                 if f.stem.lower().startswith("building")]

    if not csv_files:
        print("No building*.csv files found next to this script.")
        return pd.DataFrame()

    combined_df = []

    for file in csv_files:
        try:
            df = pd.read_csv(file, on_bad_lines='skip')


            df['building'] = file.stem


            if 'timestamp' not in df.columns or 'kwh' not in df.columns:
                continue
```



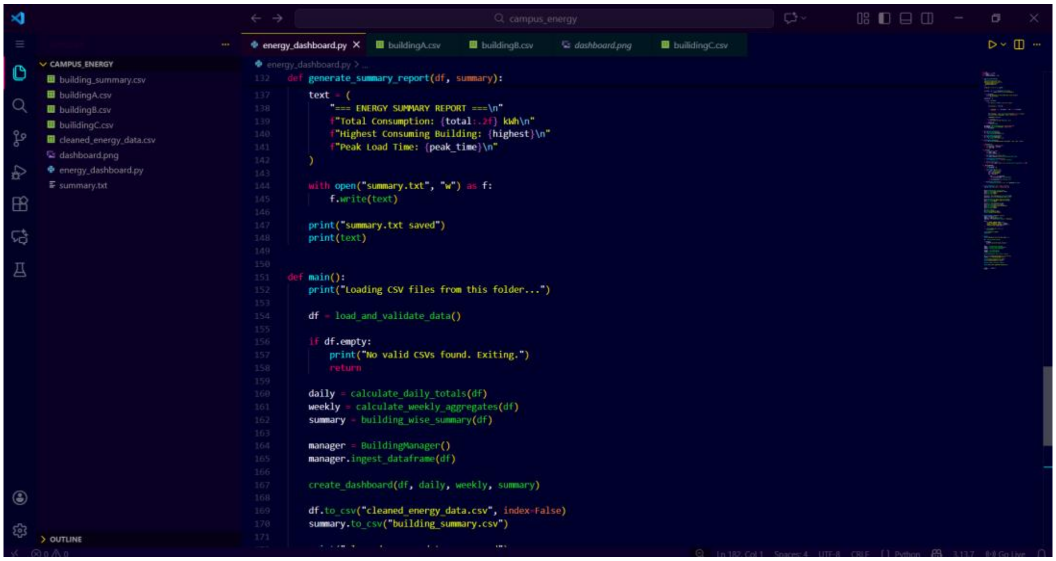
```
def load_and_validate_data():

            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):
```



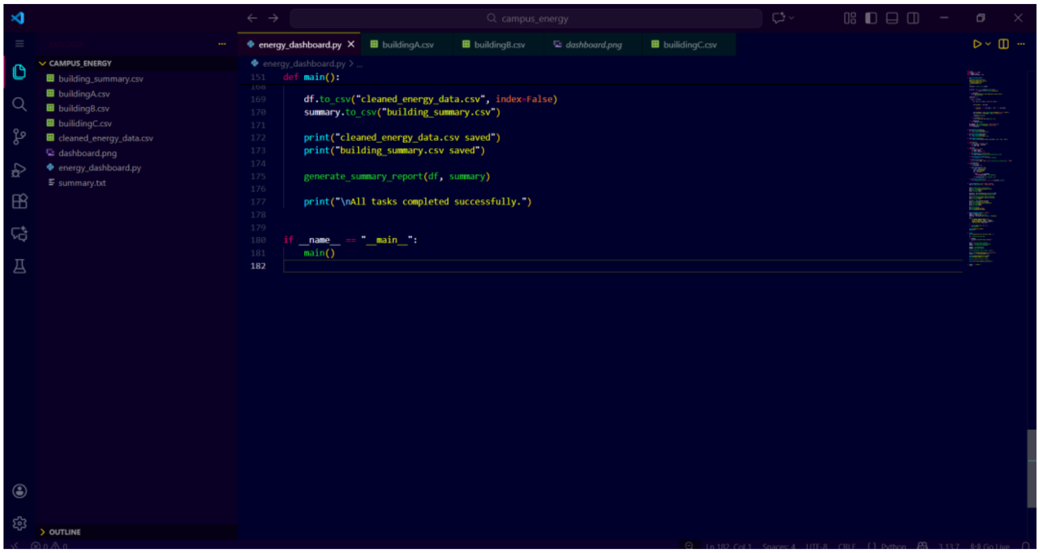
```python
class Building:
    def __init__(self, name):
        self.name = name
        self.meter_readings = []

    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():.2f} 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))

```



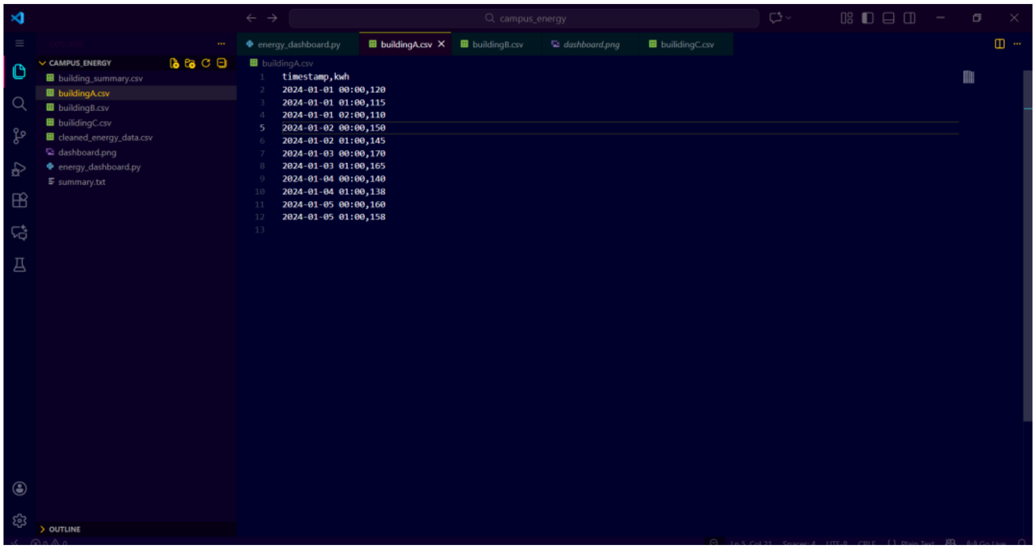
```python
def create_dashboard(df, daily, weekly, summary):
    fig, ax = plt.subplots(3, 1, figsize=(12, 16))


    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_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.png 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"
```



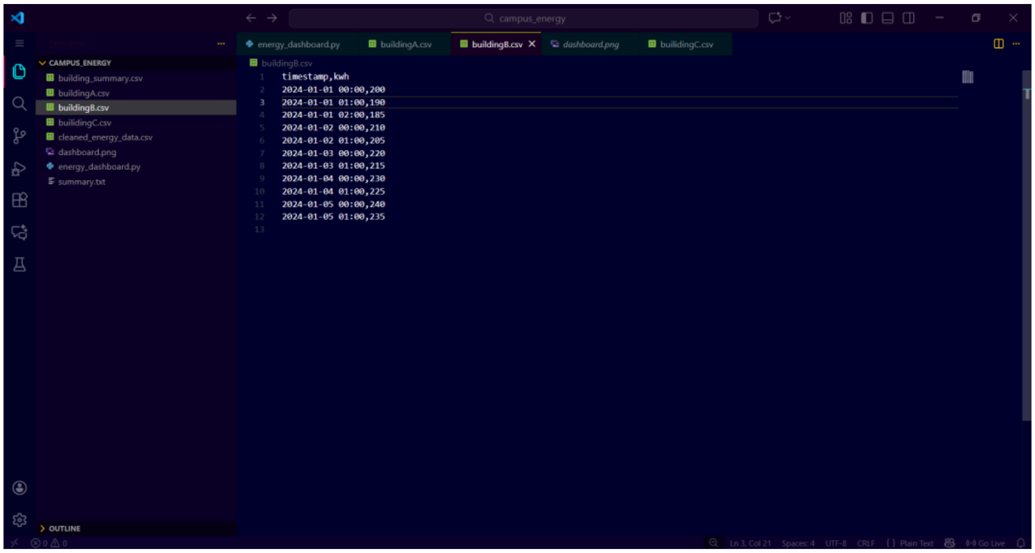
```python
def generate_summary_report(df, summary):
    text = (
        "=== ENERGY SUMMARY REPORT ===\n"
        f"Total Consumption: {total:.2f} kWh\n"
        f"Highest Consuming Building: {highest}\n"
        f"Peak Load Time: {peak_time}\n"
    )

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

    print("summary.txt saved")
    print(text)


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

    df = load_and_validate_data()

    if df.empty:
        print("No valid CSVs 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")

```



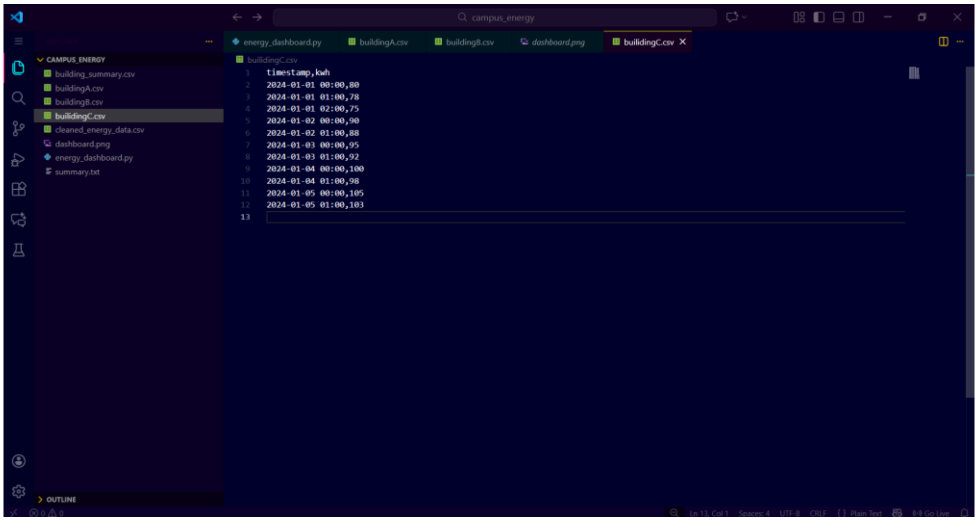
```python
def main():

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

    print("cleaned_energy_data.csv saved")
    print("building_summary.csv saved")

    generate_summary_report(df, summary)

    print("\nAll tasks completed successfully.")


if __name__ == "__main__":
    main()
```



```
timestamp,kwh
2024-01-01 00:00,120
2024-01-01 01:00,115
2024-01-01 02:00,110
2024-01-02 00:00,150
2024-01-02 01:00,145
2024-01-03 00:00,170
2024-01-03 01:00,165
2024-01-04 00:00,140
2024-01-04 01:00,138
2024-01-05 00:00,160
2024-01-05 01:00,158
```

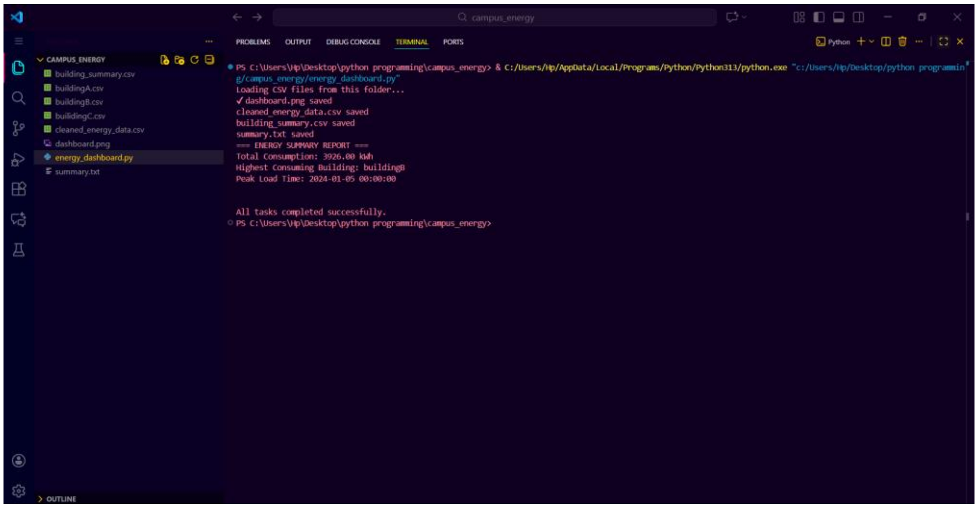


```
timestamp,kwh
2024-01-01 00:00,200
2024-01-01 01:00,190
2024-01-01 02:00,185
2024-01-02 00:00,210
2024-01-02 01:00,205
2024-01-03 00:00,220
2024-01-03 01:00,215
2024-01-04 00:00,230
2024-01-04 01:00,225
2024-01-05 00:00,240
2024-01-05 01:00,235
```

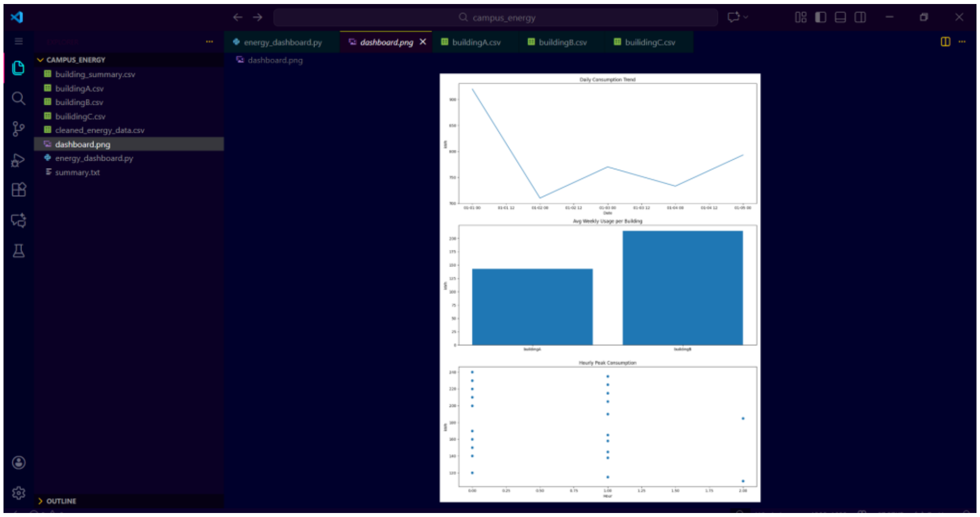


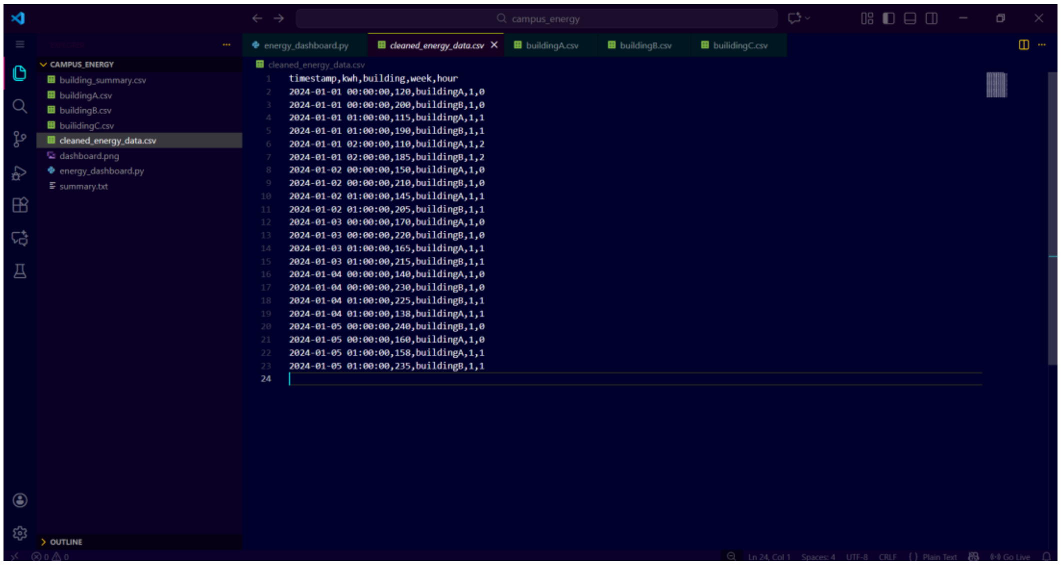
```
timestamp,kwh
2024-01-01 00:00,80
2024-01-01 01:00,78
2024-01-01 02:00,75
2024-01-02 00:00,90
2024-01-02 01:00,88
2024-01-03 00:00,95
2024-01-03 01:00,92
2024-01-04 00:00,100
2024-01-04 01:00,98
2024-01-05 00:00,105
2024-01-05 01:00,103
```

Sample Output

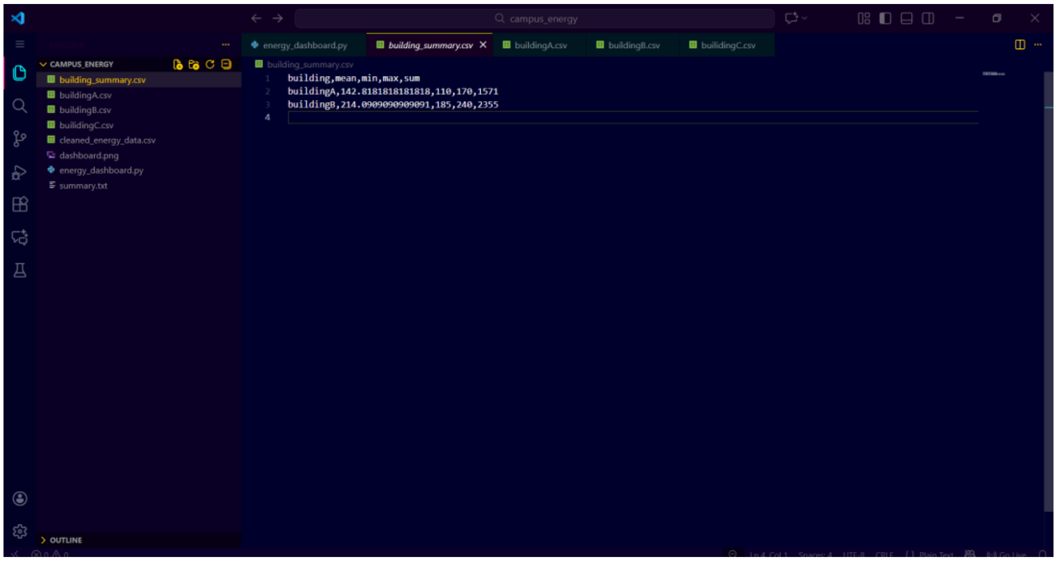


```
PS C:\Users\Hp\Desktop\python programming\campus_energy> & C:/Users/Hp/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Hp/Desktop/python programmin
g/campus_energy/energy_dashboard.py"
Loading CSV files from this folder...
✓ dashboard.png saved
cleaned_energy_data.csv saved
building_summary.csv saved
summary.txt saved
=== ENERGY SUMMARY REPORT ===
Total Consumption: 3926.00 kWh
Highest Consuming Building: buildingB
Peak Load Time: 2024-01-05 00:00:00


All tasks completed successfully.
PS C:\Users\Hp\Desktop\python programming\campus_energy>
```

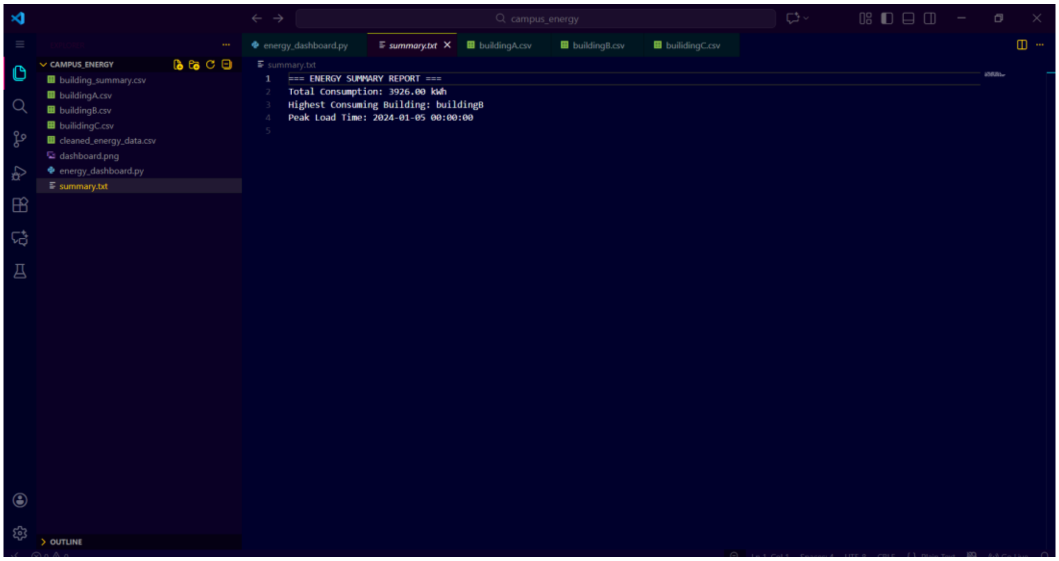




```
timestamp,kwh,building,week,hour
2024-01-01 00:00:00,120,buildingA,1,0
2024-01-01 00:00:00,200,buildingB,1,0
2024-01-01 01:00:00,115,buildingA,1,1
2024-01-01 01:00:00,190,buildingB,1,1
2024-01-01 02:00:00,110,buildingA,1,2
2024-01-01 02:00:00,185,buildingB,1,2
2024-01-02 00:00:00,150,buildingA,1,0
2024-01-02 00:00:00,210,buildingB,1,0
2024-01-02 01:00:00,145,buildingA,1,1
2024-01-02 01:00:00,205,buildingB,1,1
2024-01-03 00:00:00,170,buildingA,1,0
2024-01-03 00:00:00,220,buildingB,1,0
2024-01-03 01:00:00,165,buildingA,1,1
2024-01-03 01:00:00,215,buildingB,1,1
2024-01-04 00:00:00,140,buildingA,1,0
2024-01-04 00:00:00,230,buildingB,1,0
2024-01-04 01:00:00,225,buildingB,1,1
2024-01-04 01:00:00,138,buildingA,1,1
2024-01-05 00:00:00,240,buildingB,1,0
2024-01-05 00:00:00,160,buildingA,1,0
2024-01-05 01:00:00,158,buildingA,1,1
2024-01-05 01:00:00,235,buildingB,1,1
```



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



```
=== ENERGY SUMMARY REPORT ===
Total Consumption: 3926.00 kWh
Highest Consuming Building: buildingB
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.