ASSIGNMENT NO.-5

Campus Energy Usage Dashboard – Project Report

NAME-ARYAN

ROLL NO. – 2501730276

COURSE- BTECH CSE (AIML)

SECTION-C

SUBJECT- PROBLEM SOLVING USING PYTHON

FACULTY-SAMEER FAROOQ

Introduction

Efficient energy management is a critical concern for campus facilities. Understanding electricity usage patterns across buildings allows administrators to identify high-consumption areas and implement energy-saving strategies. This project develops a **data pipeline and dashboard** for analyzing building-level electricity consumption, providing both visual and textual insights to support decision-making.

2. Objectives

- Read and validate multiple electricity usage datasets for different buildings.
- Aggregate and summarize consumption data on daily and weekly intervals.
- Model buildings and meter readings using **object-oriented programming**.
- Visualize trends using line charts, bar charts, and scatter plots.
- Generate cleaned datasets, building summaries, and an executive report for administrators.
-

3. Methodology

Data Ingestion and Validation

- Multiple CSV files from the /data/ directory were read into a single **combined DataFrame**.
- Missing columns and bad data lines were handled to avoid errors.

- Metadata, including building name and timestamp, was added if missing.

Aggregation

- **Daily totals:** Calculated using groupby and resample('D') for each building.
- **Weekly totals:** Aggregated weekly electricity usage for trend analysis.
- **Building summaries:** Total, mean, minimum, and maximum consumption per building.

Object-Oriented Modeling

- **MeterReading** class stores timestamp and kWh data.
- **Building** class manages a list of readings and calculates total consumption.
- **BuildingManager** handles multiple buildings and generates textual reports.

4. VisualizationsDaily Consumption Trend

- Line chart showing daily electricity usage for all buildings.
- Peaks indicate periods of high activity; troughs correspond to low usage.

Average Weekly Consumption

- Bar chart comparing weekly average usage across buildings.
- Highlights high-consumption buildings that may benefit from efficiency measures.

Peak-Hour Consumption

- Scatter plot identifying the maximum consumption timestamp per building.
- Helps pinpoint times of peak load for operational planning.

Dashboard Layout

- All three plots are combined in a single figure (dashboard.png) for easy interpretation.

Data Export

- **Cleaned dataset:** cleaned_energy_data.csv
- **Building summary:** building_summary.csv
- **Executive report:** summary.txt
- **Dashboard figure:** dashboard.png

Insights & Recommendations

- BuildingA consistently shows the highest electricity usage and should be prioritized for efficiency improvements.
- Peak-hour analysis identifies times for potential demand management strategies.
- Weekly trends reveal opportunities to reduce off-peak usage and optimize schedules.
- Visual dashboards allow administrators to quickly interpret consumption patterns and make informed energy-saving decisions.

Conclusion

The project demonstrates a full **data pipeline for campus energy analysis**, including ingestion, validation, aggregation, OOP modeling, visualization, and report generation. It provides actionable insights that can help the facilities team reduce energy waste, optimize consumption, and support sustainability initiatives.

CODE:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path

data_path = Path("data")
all_files = list(data_path.glob("*.csv"))
df_list = []

for file in all_files:
    try:
        temp_df = pd.read_csv(file, on_bad_lines='skip')
        if 'Building' not in temp_df.columns:
            temp_df['Building'] = file.stem
        if 'Timestamp' not in temp_df.columns or 'kWh' not in temp_df.columns:
            continue
        df_list.append(temp_df)
    except FileNotFoundError:
        print(f"{file} not found, skipping.")
    except Exception as e:
        print(f"Error reading {file}: {e}")

df_combined = pd.concat(df_list, ignore_index=True)
df_combined['Timestamp'] = pd.to_datetime(df_combined['Timestamp'])
df_combined = df_combined.sort_values('Timestamp').reset_index(drop=True)

def calculate_daily_totals(df):
    return df.groupby(['Building', pd.Grouper(key='Timestamp', freq='D')])['kWh'].sum().reset_index()

def calculate_weekly_aggregates(df):
    return df.groupby(['Building', pd.Grouper(key='Timestamp', freq='W')])['kWh'].sum().reset_index()

def building_wise_summary(df):
    summary = df.groupby('Building')['kWh'].agg(['sum','mean','min','max']).reset_index()
    return summary

daily_totals = calculate_daily_totals(df_combined)
weekly_totals = calculate_weekly_aggregates(df_combined)
building_summary = building_wise_summary(df_combined)

class MeterReading:
    def __init__(self, timestamp, kwh):
        self.timestamp = timestamp
        self.kwh = kwh

class Building:
    def __init__(self, name):
        self.name = name
        self.meter_readings = []
    def add_reading(self, reading):
        self.meter_readings.append(reading)
    def calculate_total_consumption(self):
        return sum(r.kwh for r in self.meter_readings)
    def generate_report(self):
        total = self.calculate_total_consumption()
```

```
        def generate_report(self):
            if self.meter_readings:
                peak = max(self.meter_readings, key=lambda r: r.kwh)
                return f"{self.name}: Total={total:.2f} kWh, Peak={peak.kwh:.2f} at {peak.timestamp}"
            return f"{self.name}: No readings"

    class BuildingManager:
        def __init__(self):
            self.buildings = {}
        def add_building_reading(self, building_name, timestamp, kwh):
            if building_name not in self.buildings:
                self.buildings[building_name] = Building(building_name)
            self.buildings[building_name].add_reading(MeterReading(timestamp, kwh))
        def generate_all_reports(self):
            return [b.generate_report() for b in self.buildings.values()]

    manager = BuildingManager()
    for _, row in df_combined.iterrows():
        manager.add_building_reading(row['Building'], row['Timestamp'], row['kWh'])

    reports = manager.generate_all_reports()
    for r in reports:
        print(r)

    buildings = df_combined['Building'].unique()
    fig, axs = plt.subplots(3,1,figsize=(12,15))

    for b in buildings:
        b_data = daily_totals[daily_totals['Building']==b]
        axs[0].plot(b_data['Timestamp'], b_data['kWh'], label=b)
    axs[0].set_title("Daily Consumption Trend")
    axs[0].set_xlabel("Date")
    axs[0].set_ylabel("kWh")
    axs[0].legend()

    weekly_avg = weekly_totals.groupby('Building')['kWh'].mean()
    axs[1].bar(weekly_avg.index, weekly_avg.values, color='orange')
    axs[1].set_title("Average Weekly Consumption per Building")
    axs[1].set_ylabel("kWh")

    peak_hours = df_combined.groupby('Building').apply(lambda x: x.loc[x['kWh'].idxmax()])
    axs[2].scatter(peak_hours['Timestamp'], peak_hours['kWh'], color='green')
    for i, b in enumerate(peak_hours['Building']):
        axs[2].annotate(b, (peak_hours['Timestamp'].iloc[i], peak_hours['kWh'].iloc[i]))
    axs[2].set_title("Peak-Hour Consumption")
    axs[2].set_ylabel("kWh")
    axs[2].set_xlabel("Timestamp")

    plt.tight_layout()
    plt.savefig("dashboard.png")
    plt.show()

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

```
    total_consumption = df_combined['kWh'].sum()
    highest_building = building_summary.loc[building_summary['sum'].idxmax(),'Building']
    peak_row = df_combined.loc[df_combined['kWh'].idxmax()]
    peak_time = peak_row['Timestamp']

    with open("summary.txt","w") as f:
        f.write(f"Total Campus Consumption: {total_consumption:.2f} kWh\n")
        f.write(f"Highest Consuming Building: {highest_building}\n")
        f.write(f"Peak Load Time: {peak_time}, Consumption: {peak_row['kWh']:.2f} kWh\n")
        f.write("\nWeekly Trends:\n")
        f.write(weekly_totals.to_string(index=False))
        f.write("\n\nDaily Trends:\n")
        f.write(daily_totals.to_string(index=False))

    print("\nData export complete. Dashboard.png, CSVs, and summary.txt generated.")
```

OUTPUT:-

```
BuildingA: Total=3456.75 kWh, Peak=180.50 at 2025-06-12 15:00:00
BuildingB: Total=2890.40 kWh, Peak=160.20 at 2025-07-05 14:30:00
BuildingC: Total=3120.80 kWh, Peak=170.75 at 2025-08-20 16:00:00

Data export complete. Dashboard.png, CSVs, and summary.txt generated.
```