# Capstone Tasks – Smart Home Energy Usage Tracker

### 1. Automate Smart Home Energy Report Generation Weekly

• Set up a scheduled process (MySQL stored procedure / Databricks job).  
• Configure it to run once a week to compute **room-wise and device-wise energy usage**.  
• The job will read device logs, generate weekly summaries, and store them in **CSV/Delta files** for reporting.

### 2. Pipeline to Fetch, Clean, and Summarize Energy Logs

The pipeline has sequential stages:  
• **Fetch Stage** – Pull new smart home energy logs from CSV/API/MongoDB.  
• **Clean Stage** – Apply Python preprocessing scripts to remove nulls, duplicates, or malformed sensor readings.  
• **Summarize Stage** – Use Pandas/PySpark to generate **daily and weekly summaries** (average, peak/off-peak usage per device/room).  
• Each stage runs automatically when triggered, ensuring **continuous updates of energy data**.

### 3. Alert or Log if Device Usage Crosses a Threshold (>10 kWh per Device per Day)

• Add a Python/PySpark validation step after summarization to check thresholds.  
• If any device exceeds **10 kWh/day**, log a warning in the system or raise an alert.  
• Alerts can be sent through **email notifications, dashboards, or monitoring tools**, helping users identify abnormal energy consumption.

### Steps to Create Smart Home Energy Processing Pipeline

1. **Create Databases in MySQL & MongoDB**  
   • Use MySQL to store structured data (devices, rooms, and usage logs).  
   • Use MongoDB to store raw sensor logs in JSON format.
2. **Push Scripts to Repository**  
   • Maintain Python/PySpark scripts for data fetch, clean, summarize, and alerting.  
   • Store them in a Git repository for version control.
3. **Create a New Processing Pipeline**  
   • Define the pipeline to run sequentially – **fetch → clean → summarize → alert check**.  
   • Integrate with Databricks or a scheduler to ensure automation.
4. **Define Pipeline Stages in Code**  
   • Specify data ingestion, cleaning, summarization, and alert-checking steps.  
   • Ensure each stage passes its output to the next stage.
5. **Run the Pipeline**  
   • Save and execute the scripts.  
   • The system processes logs step by step – fetching raw data, cleaning it, summarizing results, and finally generating alerts.