Project 1: US Weather + Energy Analysis Pipeline

Overview

Duration: 4-5 weeks
Time Commitment: 12-15 hours (with AI assistance)
Coding Estimate: 3-4k lines of code

Business Context

Energy companies lose millions of dollars due to inaccurate demand forecasting. By combining weather data with energy consumption patterns, data scientists can help utilities optimize power generation, reduce waste, and lower costs. This project demonstrates your ability to build production-ready data pipelines that deliver real business value.

Project Deliverables

1. Automated Data Pipeline

What I need from you:

- A Python script that runs daily (via cron or manual execution) to fetch fresh data
- Weather data for these 5 cities: New York, Chicago, Houston, Phoenix, and Seattle
- Pull temperature (high/low) for each city
- Energy consumption data for the same regions from EIA
- Store everything in a structured format with clear date/time stamps
- If an API fails, the pipeline should log the error and continue with other data sources
- Include a script to fetch 90 days of historical data for initial analysis

City Reference Table:

City	State	NOAA Station ID	EIA Region Code
New York	New York	GHCND:USW00094728	NYIS
Chicago	Illinois	GHCND:USW00094846	PJM
Houston	Texas	GHCND:USW00012960	ERCO
Phoenix	Arizona	GHCND:USW00023183	AZPS
Seattle	Washington	GHCND:USW00024233	SCL

2. Data Quality Report

What I need from you:

- Automated checks that tell:
    - How many missing values we have and where
    - Any outliers (temperatures above 130°F or below -50°F, negative energy consumption)
    - Data freshness (flag if we're getting old data)
- A simple dashboard or report showing data quality metrics over time
- Documentation of what each quality check does and why it matters

3. Analysis Dashboard with Specific Visualizations

What I need from you:

Build a single Streamlit dashboard that I can scroll through with these 4 visualizations:

Visualization 1 - Geographic Overview

- Interactive map of the US showing all 5 cities
- Each city should display: current temperature, today's energy usage, % change from yesterday
- Use color coding: red for high energy usage, green for low
- Include a title and timestamp showing when data was last updated

Visualization 2 - Time Series Analysis

- Dual-axis line chart showing temperature and energy consumption over the last 90 days
- Temperature on left axis (solid line), energy on right axis (dotted line)
- Include a dropdown to select which city to view (or "All Cities" option)
- Highlight weekends with shaded regions
- Include proper axis labels and a legend

Visualization 3 - Correlation Analysis

- Scatter plot of temperature (x-axis) vs energy consumption (y-axis)
- Include all cities' data points, color-coded by city
- Add a regression line with the equation displayed
- Show R-squared value and correlation coefficient prominently
- Add hover tooltips showing exact values and dates

Visualization 4 - Usage Patterns Heatmap

- Heatmap showing average energy usage by temperature range (y-axis) and day of week (x-axis)
- Temperature ranges: <50°F, 50-60°F, 60-70°F, 70-80°F, 80-90°F, >90°F
- Use a color scale from blue (low usage) to red (high usage)
- Allow filtering by city

- Include text annotations on cells showing actual values
- Add a color bar legend

Technical Requirements for Streamlit Dashboard:

- Use Streamlit's native components (st.plotly_chart, st.map, etc.)
- Add a sidebar with:
    - Date range selector
    - City filter (multiselect)

4. Production-Ready Code

What I need from you:

- Organized Python modules, not a messy notebook
- A config file where I can easily change cities
- Logging that helps me debug issues without diving into code
- A README that my junior developer can follow to run everything
- Comments explaining any complex logic or business rules

Data Sources

API Base URLs:

- NOAA Climate Data Online: https://www.ncei.noaa.gov/cdo-web/api/v2
- EIA Energy: https://api.eia.gov/v2/electricity/

Registration Requirements:

- NOAA Climate Data Online: Register at https://www.ncdc.noaa.gov/cdo-web/token
- EIA Energy API: Register at https://www.eia.gov/opendata/register.php

Example API Calls:

```
# NOAA Climate Data Online - Get daily weather data

# Requires token in header: {'token': 'YOUR_TOKEN_HERE'}

weather_url = "https://www.ncei.noaa.gov/cdo-web/api/v2/data"

params = {

    'datasetid': 'GHCND',

    'stationid': 'GHCND:USW00094728',  # NYC station
```

```
    'startdate': '2024-01-01',
    'enddate': '2024-01-31',
    'datatypeid': 'TMAX,TMIN'  # Only temperature data needed
}

# EIA Energy - Daily electricity demand
energy_url = "https://api.eia.gov/v2/electricity/rto/daily-region-data/data/"
```

API Notes:

- NOAA CDO provides historical weather data (up to several years back)
- Request data types: TMAX (max temp), TMIN (min temp)
- Temperature values are in tenths of degrees Celsius (convert to Fahrenheit)
- EIA API requires specific region codes (see table above)
- **Important**: You will need to fetch historical data (90+ days) for proper analysis

Backup Datasets (if APIs are down):

- Weather: https://www.ncdc.noaa.gov/cdo-web/datasets
- Energy: https://www.eia.gov/electricity/data/browser/

Expected Findings

- Strong correlation (r > 0.7) between temperature extremes and energy usage
- Seasonal patterns in energy consumption
- Weekend vs weekday consumption differences
- Regional variations in weather-energy relationships

AI Usage Guidelines

Recommended AI Prompts

Good Prompt Example:

"I need to create a Python function that fetches weather data from NOAA API for multiple cities, handles rate limiting, implements exponential backoff for retries, and logs all errors. The function should return a pandas DataFrame with consistent column names regardless of the city."

Poor Prompt Example:

"Write code to get weather data"

Required AI Documentation

Create an `AI_USAGE.md` file documenting:

- Which AI tools you used (Gemini, ChatGPT, Claude, GitHub Copilot, etc.)
- Most effective prompts
- AI mistakes you found and fixed (if any)
- Time saved estimate

Learning from AI-Generated Code

If you found a mistake in the AI-generated code, explain:

- How you discovered the issue
- What the fix was
- What you learned from debugging it

This shows real problem-solving skills that employers value.

Video Requirements (3 minutes)

Script Framework

0:00-0:30: Problem Statement

- "Energy companies need accurate demand forecasting..."
- Quantify the business impact

0:30-2:00: Technical Walkthrough

- Show your pipeline architecture diagram
- Demo live API call with error handling
- Explain your most interesting piece of code
- Show data quality checks in action

2:00-2:30: Results & Insights

- Demo the dashboard
- Highlight the strongest correlation found
- Show one unexpected insight

2:30-3:00: AI Collaboration & Learning

- Explain how AI helped you solve a specific challenge
- Mention any AI mistakes you caught (if applicable)
- What would you do differently next time?

Video Technical Requirements

- Screen recording with your voice
- Show actual code, not just slides
- Include live demonstration of the dashboard
- Upload to YouTube (unlisted) or Loom

Suggested Repository Structure

```
project1-energy-analysis/
├── README.md                 # Business-focused project summary
├── AI_USAGE.md               # AI assistance documentation
├── pyproject.toml            # Dependencies (using uv)
├── config/
│   └── config.yaml           # API keys, cities list
├── src/
│   ├── data_fetcher.py       # API interaction module
│   ├── data_processor.py     # Cleaning and transformation
│   ├── analysis.py           # Statistical analysis
│   └── pipeline.py           # Main orchestration
├── dashboards/
│   └── app.py                # Streamlit application
├── logs/
│   └── pipeline.log          # Execution logs
├── data/
│   ├── raw/                  # Original API responses
│   └── processed/            # Clean, analysis-ready data
├── notebooks/
│   └── exploration.ipynb     # Initial analysis (optional)
├── tests/
│   └── test_pipeline.py      # Basic unit tests
└── video_link.md             # Link to your presentation
```

Grading Rubric (Employer Perspective)

Technical Excellence (60%)

- [ ] Pipeline runs successfully end-to-end
- [ ] Proper error handling and logging
- [ ] Code is modular and follows Python best practices
- [ ] Dashboard is interactive and insightful
- [ ] Statistical analysis is correct

Communication (25%)

- [ ] Video clearly explains the problem and solution
- [ ] Code is well-documented
- [ ] README provides clear setup instructions
- [ ] Technical concepts explained clearly

Business Value (15%)

- [ ] Clear understanding of use case
- [ ] Actionable insights identified
- [ ] Professional presentation

Troubleshooting Guide

NOAA API Issues:

- Make sure you're using Climate Data Online, not weather.gov
- Include your token in the request headers
- Station IDs must match exactly (see table)
- Date format must be YYYY-MM-DD
- Temperature data comes in tenths of degrees Celsius

No data returned from API?

- Check your API parameters carefully
- Verify the date range is valid (not future dates)
- Look at the raw API response structure
- Some stations may have gaps in historical data

EIA API issues?

- Verify your region code matches the table above
- Check that your API key is included correctly
- Use the correct endpoint for daily vs hourly data

Common Pitfalls to Avoid

1. **API Rate Limiting**: Implement delays between requests
2. **Missing Data**: Handle gracefully, don't let pipeline crash
3. **Time Zones**: Be consistent (recommend using UTC)
4. **Git Secrets**: Never commit API keys (use .env file)
5. **Overcomplicating**: Start simple, then add features

Peer Review Requirement

Before submission:

1. Exchange repositories with a partner of your choice
2. Partner runs your pipeline and provides feedback
3. Partner watches your video and gives presentation feedback
4. Include partner feedback in your final submission

Submission Checklist

- [ ] Pipeline fetches fresh data successfully
- [ ] Dashboard runs without errors
- [ ] All code is in src/ folder (not just notebooks)
- [ ] AI_USAGE.md documents your AI collaboration
- [ ] Video is exactly 3 minutes (±10 seconds)
- [ ] Partner review completed
- [ ] Repository is public on GitHub

Resources

- NOAA API Documentation
- EIA API Documentation
- Streamlit Documentation
- Streamlit Caching Guide

Questions?

Post in the WhatsApp channel