# Energy Analytics Portfolio Repo

**Project:** Data Center Load Growth, Renewables Buildout, Market Impacts, and Project Finance

This document is a detailed build specification you can give to Claude Code or Codex. It defines the scope, data sources, pipeline architecture, analytics modules, dashboard requirements, quality gates, and a deliverables roadmap. The goal is a resume-ready repository that looks like real energy analytics work: ingestion - QA - modeling - economics - visualization - documentation.

## Executive Summary

You will build an end-to-end analytics repository that quantifies how data center load growth changes regional power markets and how renewables + storage can meet that demand. The project combines (1) load and demand forecasting, (2) interconnection queue analytics to estimate what supply will actually be built, (3) energy market metrics like capture prices and congestion proxies, and (4) project finance pro-forma modeling to translate market conditions into IRR/NPV outcomes. Outputs include a reproducible pipeline, curated datasets, a dashboard (Streamlit or similar), and a concise written report with assumptions and results.

### Primary Learning Outcomes

- Data engineering: automated ingestion from multiple public sources, schema normalization, and validation tests.
- Time-series analytics: load shaping, weather sensitivity, scenario forecasting, and uncertainty communication.
- Energy markets: LMP-based metrics, capture price, basis and congestion indicators, and nodal vs zonal reasoning.
- Project finance: LCOE, revenue stack, merchant vs contracted cases, NPV/IRR/DSCR, and sensitivity analysis.
- Product mindset: dashboard design, clear documentation, and reproducible research practices.

## Target Audience and Resume Positioning

Design the repo to appeal to energy analytics, market analytics, DER/VPP analytics, utility planning, and data center energy procurement teams. The end product should support resume bullets like:

- Built a reproducible pipeline integrating ISO/RTO load and price data, interconnection queues, and weather signals to forecast regional load growth and supply additions.
- Modeled renewables capture prices and congestion proxies and translated market outcomes into project finance metrics (NPV, IRR, DSCR) under multiple contracting structures.
- Deployed an interactive dashboard to explore scenarios, assumptions, and outputs with automated QA checks and documentation.

## 1. Problem Statement and Research Questions

Large-load growth (especially data centers and AI compute) is reshaping regional demand profiles. At the same time, renewables and storage buildout depends on interconnection and transmission constraints, and wholesale market prices and congestion drive realized project economics. This project answers:

- Demand: How fast is load growing by region and what are plausible scenarios to 2030/2035?
- Supply: Based on interconnection queues, what capacity is likely to reach commercial operation, and when?
- Markets: How do prices, congestion proxies, and renewables capture prices evolve with load and supply scenarios?
- Finance: Under merchant or contracted structures, what do those market conditions imply for IRR/NPV/DSCR?

### Non-goals (to keep scope realistic)

- No proprietary power flow modeling. Use market-based proxies and public planning outputs instead.
- No claims of 'official forecasts' - focus on transparent, scenario-based analytics with stated assumptions.
- Avoid overfitting. Prefer interpretable models and robust validation.

## 2. Repository Deliverables

- A fully reproducible data pipeline (Makefile or task runner) that produces cleaned datasets in a /data/processed directory.
- A dashboard (Streamlit recommended) with scenario toggles, charts, and downloadable outputs.
- A report (Markdown or PDF) summarizing assumptions, methods, results, and limitations.
- Unit tests + data validation checks (Great Expectations or custom tests) for key tables.
- A set of 'starter notebooks' for EDA and model development, with final logic moved into scripts.

### Suggested Tech Stack

- Python: pandas, polars (optional), numpy, scipy, statsmodels, scikit-learn, prophet or pmdarima (optional), pyarrow, duckdb.
- SQL: DuckDB or Postgres; store raw/processed tables and run analytics queries.
- R (optional): ggplot2, forecast, fable, tidymodels for comparison notebooks.
- Visualization: Streamlit + Plotly (or Altair).
- Packaging: uv or poetry; pre-commit; ruff; black; mypy (light).
- CI: GitHub Actions to run tests and a small pipeline sample.

# 3. Data Sources and Acquisition Plan

Use only public, redistributable sources. Store raw downloads as references (URLs and retrieval metadata) and prefer programmatic pulls. Where licensing prevents redistribution, store ingestion scripts and small samples only.

## 3.1 Core datasets (minimum viable)

- ISO/RTO hourly load time series (region level).
- ISO/RTO day-ahead or real-time prices (hub or zonal LMP).
- Interconnection queues (solar, wind, storage) with status, MW, location, and target COD.
- Weather data (temperature and possibly humidity) for key load zones (NOAA or reanalysis).
- Fuel price proxies (Henry Hub gas, coal index where available) or EIA series for context.

## 3.2 Optional datasets (stretch goals)

- Curtailment reports (where published) or inferred curtailment via negative prices and generation data.
- Emissions intensity proxies (ISO published or eGRID-style averages) to estimate marginal emissions trends.
- Transmission constraints proxies: congestion components, basis differentials, or queue saturation metrics.
- Public data center location lists (only if licensing allows) or use synthetic load adders by county/region.

## 3.3 Data provenance requirements

- Every dataset must have a source entry in data_sources.yml with: name, URL, retrieval date, license notes, and fields used.
- Raw tables must include ingestion metadata columns: source, retrieval_timestamp, and transformation_version.
- Maintain a changelog for assumptions and schema changes.

# 4. Pipeline Architecture and Folder Structure

Use a layered architecture: raw -> staged -> curated -> analytics outputs.

| Layer | Purpose | Example contents |
|---|---|---|
| data/raw | Immutable raw pulls and small samples | CSV/JSON snapshots, API responses, URL manifests |
| data/staged | Normalized schemas, basic cleaning | standardized column names, parsed datetimes |
| data/curated | Analysis-ready tables | hourly load/price panel, queue features, weather features |
| data/marts | Dashboard-ready outputs | scenario results, finance outputs, summary metrics |

## 4.1 Orchestration

Implement a task runner that can: (a) fetch data, (b) run transforms, (c) run validation, (d) build models, (e) export dashboard artifacts. Options: Makefile, Invoke, or a lightweight orchestration tool. Keep it simple.

- Command: make all (or python -m pipeline run-all).
- Deterministic outputs: pin dependency versions and set random seeds.
- Cache expensive steps and store intermediate parquet files.

## 4.2 Data QA (required)

- Schema checks: expected columns and types for curated tables.
- Range checks: prices, load, temperature within plausible bounds.
- Completeness checks: missing hours and duplicates; enforce 8760-ish coverage.
- Reconciliation checks: compare monthly totals to published summaries where possible.

# 5. Analytics Modules (build in this order)

## Module A - Load and Data Center Growth Scenarios

Create a baseline regional load model and overlay data center growth scenarios. Treat the data center component as an additive load driver with commissioning lags and uncertainty bands.

- Inputs: hourly load, temperature, calendar effects; optional economic indicators.
- Baseline model: temperature response + day-of-week + seasonality; validate with backtesting.
- Scenario engine: define Low/Base/High data center growth paths (MW) and map them to regions.
- Outputs: hourly load forecasts per region + peak demand projections; uncertainty intervals.

### Recommended methods

- Interpretable: regression with splines or GAM-like terms; or gradient boosting with SHAP explanations.
- Forecast evaluation: rolling-origin backtest with MAPE/MAE and peak-error metrics.
- Document assumptions: commissioning schedule, load factor, and sensitivity to temperature.

## Module B - Interconnection Queue: Expected Online Capacity

Ingest and normalize interconnection queue data across one or more ISOs. Estimate probability of completion and expected COD distribution by technology and region.

- Standardize fields: project_id, iso, zone, county/state (if available), tech, MW, status, queue_date, target_COD.
- Feature engineering: age in queue, status category, withdrawn flags, cluster phase, interconnection cost (if present).
- Model: survival or classification model to estimate completion probability; calibrate to historical cohorts.
- Output: expected online MW by year (P50/P90) and by tech; separate solar/wind/storage.

### Validation

- Cohort checks: compare predicted build rates to historical build-out (where data exists).
- Sensitivity: show how results change if completion probabilities shift +/- 10 to 20 percentage points.

## Module C - Market Metrics and Renewable Economics

Quantify market conditions relevant to renewables and large loads. Build metrics that are intuitive and defensible.

- Price level and volatility: monthly average, peak hours, distribution tails.
- Congestion proxy: hub vs zone basis; or congestion component where available.
- Capture price: generation-weighted average price for solar and wind (using profile shapes).
- Curtailment proxy: negative price hours, low price hours during high renewable output.

- Emissions proxy: region average or marginal proxy where published; show directional impacts.

### Outputs

- Time-series plots of capture price vs hub price; capture ratio trends.
- Scenario results: how capture ratio changes under more load vs more renewables vs more transmission constraint.
- A 'zone attractiveness' score for building new renewables or pairing storage.

## Module D - Project Finance Pro-forma

Translate market outcomes into project economics. Implement a transparent pro-forma with merchant and contracted cases.

- Project types: utility-scale solar, onshore wind, and 2 to 4-hour battery storage (simple model).
- Capex/opex: parameterized assumptions with sensitivity ranges (no hardcoding).
- Revenue stack: energy revenue (capture price), optional capacity/REC proxy, storage arbitrage revenue proxy.
- Financing: debt/equity split, cost of debt, tenor, DSCR constraint, tax assumptions simplified (document).
- Metrics: LCOE, NPV, IRR, payback, DSCR trajectory.

### Contracting cases

- Merchant: floating revenue based on modeled prices and capture ratio.
- Fixed PPA: fixed price with optional escalation; compare to merchant distribution.
- Hybrid: fixed floor + merchant upside.

### Deliverable charts

- Waterfall chart of revenue drivers (energy, curtailment loss, storage uplift).
- Sensitivity tornado: IRR vs capex, capture price, curtailment, debt rate.
- Scenario matrix: IRR across (Load growth scenario x Supply buildout scenario).

## 6. Dashboard Specification

Build a single interactive interface that lets a reviewer explore assumptions and results quickly. Prefer Streamlit for speed.

### Required pages

- Overview: key KPIs, scenario selector, headline charts.
- Load: historical load, model fit, forecast scenarios, peak projections.
- Supply: queue pipeline, expected online MW by year, tech mix, regional comparison.
- Markets: prices, basis proxy, capture prices, negative price hours.
- Finance: project pro-forma inputs, IRR/NPV/DSCR outputs, sensitivity charts.
- Downloads: export scenario outputs as CSV/parquet and generate a PDF summary report.

### User controls

- Scenario toggles: Load (Low/Base/High), Queue completion (P50/P90), and Market tightness (low/high congestion).
- Finance knobs: capex, opex, WACC, debt rate, PPA price, degradation assumptions.
- Region selector: ISO/zone; consistent naming across all pages.

### Design requirements

- All charts labeled with units and definitions. No unexplained acronyms.
- Every metric includes a tooltip or footnote definition.
- The dashboard must run from processed data only (no long fetch during app runtime).

# 7. Engineering Standards and Documentation

Treat this like a professional analytics codebase.

## 7.1 Code quality

- Use typed function signatures where feasible; keep modules small and testable.
- Centralize constants and assumptions in a config file (YAML/TOML).
- Implement logging with clear step-level messages.
- Use parquet for processed tables; document schemas in /docs/data_dictionary.md.

## 7.2 Reproducibility checklist

- One-command build: creates all curated tables and dashboard artifacts.
- Pin dependencies; provide a lockfile.
- Provide small sample datasets so reviewers can run the pipeline without waiting.
- CI runs unit tests and a small 'smoke pipeline' on sample data.

## 7.3 Documentation artifacts

- README: problem framing, architecture diagram, how to run, and key findings.
- Method notes: separate docs for (a) load model, (b) queue model, (c) finance model.
- Assumptions: a single assumptions table with values, rationale, and sensitivity ranges.

## 8. Milestone Roadmap (Suggested)

Build iteratively. Each milestone should produce something demo-able.

### Milestone 1 - Data foundation (1 to 3 days)

- Pick 1 region (e.g., ERCOT or PJM) for the MVP.
- Ingest hourly load + hub prices + weather and produce a clean hourly panel table.
- Add QA checks and generate first charts.

### Milestone 2 - Queue normalization (2 to 5 days)

- Ingest interconnection queue, normalize schema, produce expected online MW by year with a simple heuristic.
- Replace heuristic with a simple statistical model if historical statuses allow.

### Milestone 3 - Markets metrics (2 to 5 days)

- Compute capture prices using standard solar and wind profiles and price series.
- Add congestion proxy and negative price hour metrics.
- Write a short findings note with 3 key insights.

### Milestone 4 - Project finance module (3 to 7 days)

- Implement solar pro-forma first. Add wind and storage as extensions.
- Run scenario matrix and sensitivity charts. Export results as tidy tables.

### Milestone 5 - Dashboard + polish (2 to 5 days)

- Build dashboard pages and scenario controls.
- Add downloads and an auto-generated summary report page.
- Finalize README with architecture and usage.

# 9. Operating Instructions for Claude Code / Codex

Use the LLM as a co-developer, but enforce structure and QA. The agent should proceed step-by-step, committing code frequently and keeping tasks small.

## 9.1 Rules for the agent

- Before writing code, propose the file changes and functions to create.
- Prefer small, composable functions. Avoid huge notebooks for final logic.
- After each ingestion step, create a schema summary and basic data validation checks.
- After each milestone, update the README with what now works and how to run it.
- Never silently change assumptions. Update config and document changes.

## 9.2 Step-by-step agent task list

- Initialize repo with python packaging, ruff/black, pre-commit, and a Makefile.
- Create data_sources.yml and a metadata logging utility.
- Implement ingestion scripts for load, prices, weather, and queue data.
- Build transformation scripts to produce curated parquet tables.
- Implement QA tests and a smoke pipeline on sample data.
- Implement modeling modules and export scenario outputs.
- Build Streamlit dashboard and connect to marts tables.
- Write a concise report with findings and limitations.

## 9.3 Suggested prompts to use while building

Use this repo spec as the source of truth. Start with Milestone 1.
1) Propose a repo skeleton and Makefile commands.
2) Identify public sources for hourly load and hub prices for the chosen ISO (prefer programmatic).
3) Implement ingestion -> staged -> curated transforms, saving parquet outputs.
4) Add QA tests for missing hours, duplicates, and basic ranges.
5) Produce 3 initial charts and a short markdown findings note.

## Appendix A - Example Resume Bullets

- Built an end-to-end energy analytics pipeline combining ISO hourly load and price data, interconnection queue records, and weather drivers; implemented automated QA checks and reproducible parquet data marts.
- Developed scenario-based forecasts for regional load growth incorporating data center commissioning assumptions and temperature sensitivity; evaluated performance with rolling backtests.
- Estimated expected online renewable and storage capacity using normalized queue datasets and completion probability modeling; produced P50/P90 buildout trajectories by technology and region.
- Computed market metrics including capture price and basis proxies and translated price outcomes into project finance results (LCOE, NPV, IRR, DSCR) for solar, wind, and storage under merchant and contracted structures.
- Deployed a Streamlit dashboard to explore scenarios, assumptions, and outputs, enabling exportable tables and automated report generation for stakeholder-ready analysis.

## Appendix B - Repo Checklist for 'Portfolio Ready'

- README includes: problem statement, data sources, architecture diagram, how to run, and key findings.
- A /docs folder includes: assumptions table, data dictionary, and method notes.
- Dashboard runs locally in under 60 seconds using processed data.
- CI passes: lint, tests, and smoke pipeline.
- At least one region end-to-end is complete; second region is a clearly labeled extension task.