# Exploiting LLMs for Better Code

An example of using Claude to re-design and refactor an ETL pipeline
4 July 2025

# Overview

It’s not all Vibing (yet)

- Before the LLMs (preparatory work)
- Starting small with Claude
- Assisted refactoring with Claude
- Assisted planning with Claude
- Assisted vibing with Claude Code for refactoring
- Interactive vibing with Claude for EDA

# Definitions

Vibing Coding:

Coding primarily with natural language prompts.  In this context, providing the LLM with prompts and letting the LLM write the code.

AI-Assisted Coding:

Includes vibing.  But in this context, a combination of human and LLM coding.

# Before The LLM

What did we do?

- A colleague and I wrote code to extract, clean, and analyze the code.
- No LLMs, just pure human coding.

Benefit

- This provided us with key insights into the data.
- Identified issues with current data collection methodology.
- LLMs are not quite ready for starting from scratch (but they’re close)

# Starting Small with Claude

What did we do?

Provide Claude with context for research,

- Research context
- Related research papers
- Raw data examples
- Processed data examples
- Examples of issues in the data
- Team member skill-level and experience

Ask Claude

- Debug errors
- Check code for correctness
- Refactor single functions

Benefits → understanding of what Claude needs to work best, how to supervise Claude to get the best results.

# Assisted Refactoring with Claude

What did we do?

- Provide Claude with same context.
- Ask Claude to check for bugs, issues
- Ask Claude to refactor code and provide better visualizations.
- Review code changes, review outputs
- It’s still an iterative process.

Benefits

- Claude provided new ideas (html report was Claude’s idea)
- Better quality code, but it’s still OUR (mostly) code.

# Now what?

What we've done so far:

- Re-envision our data collection strategy, building a web app to collect data.
- Use current script to analyze extracted web data
- Manually build code to filter web data to work with previous code.

What's next?

- Merge all of our code into an etl pipeline (one command does it all)
- Using Claude Code.

# Assisted Planning with Claude

What did we do?

- Provide Claude with context and existing code.
- Ask Claude for advise (DO NOT CODE YET)  on building a more robust, production quality ETL pipeline.
- Iterate (a lot) with Claude, reviewing advise, suggesting improvements
- (Important) providing Claude with use case examples.
- Several hours of work.

Benefit

- I now have a plan and i’m confident it addresses all use cases.
- I’m ready to refactor.

# Assisted Vibing with Claude Code

What did we do?

- Install Claude Code (I'm paying $200/month for the max plan)
- Copy all prompts and responses into the new repo.
- Copy old code into temp_old_code subdirectory.
- Create a long, context laden prompt, referencing old code, and planning.
- Iterate with Claude to generate claude.md context file (DO NOT CODE YET)
- Ask Claude to start coding

Iterate–iterate–iterate

- Check files and directory structure (not code).  (Iterate)
- Check all generated artifacts.  (Iterate)
- Check all logs. (Iterate)
- Run scripts and check outputs. (Iterate)
- And, iterate.

# Interactive Vibring with Claude for EDA

Now that I verified all the artifacts, datasets, and logs are correct:

- Ask Claude to add more exploratory data analysis (EDA) artifacts to report
- Analyze report.
- Ask Claude for more changes, more tables, more plots.
- Analyze report.
- Ask Claude for more changes
- Etc.

# Summary & Conclusions

Summary

- We started with existing code and snippets and successively added more LLM generated code to our code base, refactoring from research code to production quality code in a short time.
- Once refactored, we used Claude to assist with exploratory data analysis.

Conclusions

- Claude (LLM used almost exclusively) is able to plan and write production quality code. But it still requires a lot of iterations to get it right.
- Better prompting is the key. Context is key. It would be useful to explore best practices w.r.t. prompting to improve LLM code automation.
- Coding with LLMs has gone from simple plots and debugging to fully refactored, production quality coding in less than three years. It's not 100% autonomous yet. But, at this rate, I expect it will be in less than a few years.