Coding & Development / Python Prompts

Generate pandas code for data cleaning, transformation, and visualization based on a described dataset.
Difficulty: Intermediate → Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Data Analysis, Pandas, Visualization
Updated: May 2026
Why This Prompt Exists
Most data analysis code is repetitive — loading, cleaning, transforming, visualizing.

You get:

- manual data cleaning (slow, error-prone)
- inconsistent column naming
- missing value handling that’s not documented
- visualizations that don’t tell the right story
- analysis that can’t be reproduced

But data analysis is not one-off scripting.

It is a reproducible pipeline.

- Load data: from CSV, Excel, SQL, or API
- Clean data: handle missing values, rename columns, filter outliers
- Transform data: group, aggregate, pivot, merge
- Analyze: statistics, trends, correlations
- Visualize: charts that answer specific questions

Without a structured approach, analysis is messy.

This framework forces AI to generate reproducible analysis scripts.

The Prompt

```
Assume the role of a data analyst who writes clean, reproducible
pandas code.

Your task is to generate a data analysis script.

Generate:

1. IMPORTS
   - pandas, numpy, matplotlib/seaborn

2. DATA LOADING
   - Read data from source
   - Initial inspection (head, info, describe)

3. DATA CLEANING
   - Handle missing values (drop or fill)
   - Rename columns (snake_case)
   - Fix data types
   - Remove duplicates

4. DATA TRANSFORMATION
   - Filtering, grouping, aggregating
   - Creating new calculated columns

5. ANALYSIS & VISUALIZATION
```

   - Summary statistics
   - Key charts (histograms, bar plots, scatter plots)

6. INSIGHTS SUMMARY
   - Key findings in plain English

INPUTS:

Data Source Description:
[E.G., "CSV file with columns: date, product, sales, region"]

Data Cleaning Needs:
[E.G., "Dates are strings, sales has missing values, region has typos"]

Analysis Questions (2-3):
[LIST]

Desired Visualizations:
[E.G., "Sales by month," "Sales by region," "Top 5 products"]

Output Format:
[JUPYTER NOTEBOOK / PYTHON SCRIPT]

RULES:
- Use pandas for data manipulation
- Use matplotlib/seaborn for visualization
- Add comments explaining each step
- Handle common data issues (missing values, wrong types)

```
- Make visualizations publication-ready (labels, titles, legends)
- Print summary statistics before and after cleaning
```

How To Use It

- Describe your data schema clearly — column names, types, meanings.
- List specific cleaning needs — missing values, outliers, typos.
- The analysis questions drive what visualizations to create.
- Test the generated script on a sample of your data first.
- Add your own domain-specific logic after generation.

Example Input

Data Source Description: CSV file with columns: Date (string), Product (string), Sales (number), Region (string: East, West, North, South)

Data Cleaning Needs: Date column needs conversion to datetime, Sales has missing values (fill with median), Region has typos (“Eat” instead of “East”)

Analysis Questions: “Which month had the highest sales?”, “Which region has the best performance?”, “What are the top 3 products?”

Desired Visualizations: Sales by month (line chart), Sales by region (bar chart), Top 5 products (horizontal bar chart)

Output Format: JUPYTER NOTEBOOK

Why It Works
Most data analysis is messy and unreproducible.

This framework improves outcomes by forcing:

- reproducible pipeline (load → clean → transform → analyze)

- explicit cleaning steps (handles edge cases)
- visualization best practices (labels, titles, legends)
- insights summary (plain English findings)
- pandas best practices (vectorized operations)

Great data analysis scripts don't just produce numbers — they tell a story and can be rerun.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, Python frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also The Python Function Generator