Best Local LLMs for Data Analysis (2026)

February 8, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** For data analysis, Qwen 2.5 Coder 32B on a 24GB card is the best local option — excellent pandas, SQL, and data reasoning. On 12GB, DeepSeek Coder V2 16B handles SQL generation and data wrangling reliably. On 8GB, Qwen 2.5 Coder 7B covers basic analysis. The workflow: describe your data and schema to the model, ask for analysis code, run it in Python, paste results back for interpretation. Always include column names and sample rows in your prompts — models hallucinate schema when they have to guess.

📚 **Related:** Best Coding Models · Qwen Models Guide · VRAM Requirements · Best LLMs for Math

"Data analysis" covers a lot of ground. Here's what local LLMs are actually good at:

- **Writing pandas/Python code** to process CSV and JSON files
- **Generating SQL queries** from natural language
- **Summarizing datasets** — distributions, outliers, trends
- **Interpreting results** — explaining what the numbers mean
- **Cleaning data** — handling missing values, type conversions, deduplication

What they're not good at: training ML models (different task entirely), processing datasets larger than their context window (a few thousand rows at most), or replacing a statistician's judgment on methodology.

The right model depends on whether you're writing code or interpreting data — and those are different skills.

Top Picks

Qwen 2.5 Coder 32B — Best Overall

The strongest local model for data work. Qwen 2.5 Coder generates clean pandas code, handles complex SQL with joins and window functions, and can reason about what results mean — not just produce them.

What it does well:

- Multi-step pandas transformations (groupby, merge, pivot, rolling windows)
- SQL with CTEs, subqueries, and window functions
- Explaining analysis results in plain English
- Catching data quality issues in code

VRAM: ~20GB at Q4. Needs a 24GB card (RTX 3090 or 4090).

Context: 128K tokens — enough to include substantial schema descriptions and sample data.

```
ollama pull qwen2.5-coder:32b
```

If you have 24GB VRAM and data analysis is a primary use case, this is the model to run.

DeepSeek Coder V2 16B — Best Mid-Size

DeepSeek Coder V2 is a MoE (Mixture of Experts) model — 16B total parameters with fewer active per token, making it faster than a dense 16B. Particularly strong at SQL generation, where it rivals models twice its size.

What it does well:

- SQL query generation from natural language descriptions
- Database schema understanding
- Basic-to-intermediate pandas code
- Data type conversions and cleaning scripts

VRAM: ~9GB at Q4. Comfortable on 12GB cards (RTX 3060).

```
ollama pull deepseek-coder-v2:16b
```

The sweet spot for data analysis on mid-range hardware. SQL quality is its standout feature.

Qwen 2.5 14B — Best Balance

Not a coding-specific model, but Qwen 2.5 14B is strong enough at code generation and data reasoning to handle most analysis tasks. The advantage over dedicated coding models: it's better at interpreting and explaining results, not just producing code.

What it does well:

- Writing analysis code and explaining what it does
- Reasoning about data patterns ("revenue dropped in Q3 because…")
- Generating both code and narrative reports
- Handling ambiguous requests ("explore this dataset")

VRAM: ~9GB at Q4. Runs on 12GB cards.

```
ollama pull qwen2.5:14b
```

Choose this over DeepSeek Coder when you need the model to think about your data, not just write code for it.

Phi-4 14B — Best for Math-Heavy Analysis

Phi-4 excels at the quantitative reasoning side of data analysis. Statistical calculations, formula derivation, mathematical pattern recognition — anywhere the task is more math than code, Phi-4 outperforms models at its size.

What it does well:

- Statistical calculations and interpretations
- Mathematical pattern recognition
- Structured data processing
- Formula-based data transformations

What it doesn't: Creative interpretation, long-context analysis (16K limit), multilingual data.

VRAM: ~10GB at Q4.

```
ollama pull phi4
```

A specialist pick. If your analysis is heavy on statistics and light on complex code, Phi-4's math foundation gives it an edge.

By Task: Which Model for What

SQL Generation

Best: Qwen 2.5 Coder 32B > DeepSeek Coder V2 16B > Qwen 2.5 Coder 7B

SQL is where coding models shine. They handle joins, aggregations, window functions, CTEs, and subqueries reliably — as long as you give them the schema.

Critical: Always include your table schema in the prompt. Without it, models guess column names and get them wrong.

```
Given this schema:

CREATE TABLE sales (
  id INT PRIMARY KEY,
  product_name VARCHAR(100),
  category VARCHAR(50),
  amount DECIMAL(10,2),
  sale_date DATE,
  region VARCHAR(50)
);

Write a SQL query to find the top 5 product categories by total revenue
for each region in Q4 2025, including month-over-month growth rates.
```

This produces accurate SQL. "Write a query to find top products" without schema produces garbage.

pandas / Python Data Wrangling

Best: Qwen 2.5 Coder 32B > Qwen 2.5 Coder 7B > DeepSeek Coder V2 16B

Include sample data in your prompt — even just 3-5 rows. Models need to see column names, data types, and value formats to write correct code.

```
I have a CSV with these columns and sample data:

date,product,units_sold,price,region
```

```
2025-01-15,Widget A,150,29.99,North
2025-01-15,Widget B,89,49.99,South
2025-01-16,Widget A,203,29.99,North
2025-02-01,Widget C,45,99.99,East

Write pandas code to:
1. Load this CSV
2. Add a revenue column (units_sold * price)
3. Create a monthly summary with total revenue per product
4. Find which product had the highest month-over-month growth
```

Data Interpretation and Reporting

Best: Qwen 2.5 32B > Qwen 2.5 14B > Phi-4 14B

General models outperform coding models here. When the task is "what does this data mean?" rather than "write code to process this data," you want a model with strong reasoning and natural language ability.

```
Here are the monthly revenue figures for our SaaS product:

Jan: $45,200
Feb: $43,800
Mar: $51,300
Apr: $49,100
May: $62,400
Jun: $58,900
Jul: $71,200
Aug: $68,500

What patterns do you see? What might explain the Q2 acceleration?
Are there any concerning trends?
```

Coding models will offer to write analysis code. General models will actually analyze.

Data Cleaning

Best: Qwen 2.5 Coder (any size) > DeepSeek Coder V2

Handling missing values, standardizing formats, deduplication, type conversions — these are bread-and-butter coding tasks. Even the 7B coding models handle them well.

```
This CSV has messy data:
- Dates in mixed formats (MM/DD/YYYY, YYYY-MM-DD, "January 5, 2025")
- Phone numbers with inconsistent formatting
- Duplicate rows where name differs by whitespace
```

```
- Missing values represented as "", "N/A", "null", and actual NaN

Write pandas code to clean all of these issues.
```

The Practical Workflow

Local LLMs don't replace Python for data analysis — they accelerate it. Here's the workflow that actually works:

Step 1: Describe Your Data

Tell the model what you have. Include:

- Column names and types
- 3-5 sample rows
- The size of the dataset
- Any known issues (missing values, mixed types)

Step 2: Ask for Analysis Code

Be specific about what you want. "Analyze this data" produces generic code. "Calculate the rolling 7-day average of daily sales, grouped by region, and flag days where sales dropped more than 20% from the previous week" produces useful code.

Step 3: Run the Code

Copy the generated code into a Python script or Jupyter notebook. Run it on your actual data.

Step 4: Paste Results Back

Take the output — summary statistics, error messages, charts data — and paste it back to the model:

```
I ran your code and got this output:

Region    Avg Revenue    Flagged Days
North     $12,340        3
South     $8,920         7
East      $15,100        1
West      $11,200        5
```

```
The South region has the most flagged days. What might explain this?
What additional analysis would you recommend?
```

Step 5: Iterate

The model suggests next steps. You run more code. You paste more results. Each round refines the analysis.

This loop is where local LLMs outperform notebooks-only workflows. Instead of figuring out the right pandas function from documentation, you describe what you want and get working code in seconds.

Prompting Strategies

Include Schema, Always

The single biggest improvement you can make:

Prompt Quality	What Happens
"Write SQL to find top customers"	Model guesses table/column names. 50% chance of wrong schema.
"Using the `orders` table with columns `customer_id`, `amount`, `order_date`, find the top 10 customers by total spend in 2025"	Correct SQL on first try.

Show Sample Data

For pandas work, paste 3-5 representative rows. The model needs to see:

- Actual column names (not guessed)
- Data types (are prices strings or floats?)
- Value formats (dates as ISO or American? IDs as integers or UUIDs?)
- Edge cases (any nulls, special characters, mixed types?)

Specify Output Format

```
# Vague
"Analyze sales trends"
```

```
# Specific
"Write pandas code that outputs a DataFrame with columns:
month, total_revenue, pct_change_from_previous, is_above_average
Print the result as a formatted table."
```

Break Complex Analysis Into Steps

Don't ask for everything at once. A 7B model that handles three-step analysis reliably will fumble a ten-step request. Break it up:

1. "Load and clean the data"
2. "Calculate summary statistics by category"
3. "Identify outliers using IQR method"
4. "Generate the final report"

Each step builds on the previous output. Simpler prompts produce more reliable code.

Limitations

Context Window vs Dataset Size

A model with 128K context can process roughly 90K words of input — that's a lot of text but maybe 5,000-10,000 CSV rows. For larger datasets, you can't paste the whole thing. Instead:

- Provide schema + sample rows
- Ask for code that processes the full file from disk
- Paste summary statistics, not raw data

Hallucinated Column Names

If you don't provide schema, the model will invent column names that seem reasonable but don't exist in your data. The generated code runs, throws a `KeyError`, and you waste time debugging something the model made up. Always provide column names.

Code Quality

Model-generated code usually works but isn't always optimal. Common issues:

- Inefficient loops where vectorized operations would be faster
- Missing error handling for edge cases

- Hardcoded values that should be parameters
- Deprecated pandas syntax (`.append()` instead of `pd.concat()`)

Review and test all generated code. Use it as a starting point, not a final product.

Statistical Rigor

LLMs can calculate means and standard deviations. They should not design your statistical methodology. If your analysis requires hypothesis testing, regression assumptions, or causal inference, consult a statistics reference — don't trust the model to choose the right test.

Tools That Help

Open Interpreter

Open Interpreter lets the LLM write and execute Python code directly. Ask "analyze this CSV" and it writes the code, runs it, sees the output, and iterates — all automatically.

```
pip install open-interpreter
interpreter --model ollama/qwen2.5-coder:32b
```

Powerful but risky — it executes arbitrary code on your machine. Review what it's doing.

Jupyter + Ollama

Use Ollama's API directly in Jupyter notebooks for an interactive analysis loop:

```
import requests

def ask_llm(prompt, model="qwen2.5-coder:32b"):
    response = requests.post(
        "http://localhost:11434/api/generate",
        json={"model": model, "prompt": prompt, "stream": False}
    )
    return response.json()["response"]

# In a notebook cell:
schema = """Columns: date, product, units, price, region
```

```
Sample: 2025-01-15, Widget A, 150, 29.99, North"""

code = ask_llm(f"Given this data:\n{schema}\n\nWrite pandas code to calculate monthly revenue by
print(code)
# Copy to next cell, run, inspect output, iterate
```

VRAM Recommendations

Your VRAM	Best Model	What You Get
8GB	Qwen 2.5 Coder 7B	Basic pandas, simple SQL, data cleaning
12GB	DeepSeek Coder V2 16B	Strong SQL, intermediate pandas, decent reasoning
12GB (alternative)	Qwen 2.5 14B	Better interpretation, solid code generation
16GB	Qwen 2.5 Coder 14B	Good balance of code + reasoning
24GB	Qwen 2.5 Coder 32B	Best overall — complex analysis, multi-step workflows

→ Use our Planning Tool to check exact VRAM for your setup.

If you split time between data analysis and other tasks (chat, writing, general coding), Qwen 2.5 14B or 32B (non-Coder) gives you a versatile single model. If data work is your primary use case, the Coder variants are worth the specialization.

The Bottom Line

Local LLMs won't replace pandas, SQL, or your analytical judgment. They accelerate the loop: describe what you want → get working code → run it → interpret results → iterate. A task that takes 30 minutes of Stack Overflow browsing and documentation reading takes 5 minutes with a good local model.

Start with the workflow: schema in the prompt, sample data included, specific questions, iterative refinement. The model choice matters less than the prompting strategy — even a 7B model produces useful analysis code when you tell it exactly what your data looks like.

📚 **Model guides:** Best Coding Models · Qwen Models Guide · Phi Models Guide · Best LLMs for Math

📚 **Hardware:** VRAM Requirements · 12GB VRAM Guide · 24GB VRAM Guide

Get notified when we publish new guides.

Subscribe — free, no spam
