Quant Developer Screening Assignment

Overview

This assignment is designed to evaluate candidates applying for the **Quant Developer Intern position**.

The goal is to assess your ability to:

- Write **efficient and clean code**
- Apply **statistical reasoning**
- Handle **imperfect real-world data**
- Design and test **systematic trading strategies**
- Work under **constraints and ambiguity**

This is **not a textbook exercise**. There is no single correct answer. We are interested in **your reasoning process and implementation decisions**.

Expected completion time: **8–12 hours**

Submission deadline: **48 hours after receiving the dataset**

Problem Statement

You are given **minute-level historical price data for multiple assets**.

Your task is to **identify exploitable statistical relationships between assets and design a systematic trading strategy to capture those opportunities.**

You must:

1. Analyze the dataset
2. Identify candidate trading relationships
3. Design a trading signal
4. Implement a backtesting system
5. Evaluate strategy performance

Dataset Description

The dataset contains **minute-level prices for multiple assets**.

Notes:

The dataset intentionally contains:

- Missing values
- Outliers
- Non-stationary periods
- Possible structural breaks

Your system should handle these appropriately.

Requirements

1. Data Processing

You must implement a pipeline that:

- Loads the dataset
- Handles missing values
- Detects or mitigates extreme outliers
- Ensures time alignment between assets

You may not manually edit the dataset.

All preprocessing must occur **programmatically**.

---

2. Relationship Discovery

Your system should identify **statistically meaningful relationships between assets**.

Examples of approaches include (but are not limited to):

- correlation analysis
- spread analysis
- stationarity testing
- cointegration testing
- clustering

You must justify **why the relationship is suitable for trading**.

---

3. Trading Strategy Design

Design a systematic trading strategy based on the discovered relationship.

Your strategy should define:

- Entry signals
- Exit signals
- Position sizing
- Risk controls

Your strategy must include **clear rules that can be executed programmatically**.

Example signal framework (illustrative only):

spread = asset_A - beta * asset_B

z_score = (spread - mean) / std

Entry and exit thresholds should be **chosen and justified**.

---

4. Backtesting Engine

You must implement a simple backtesting system that:

Tracks:

- open positions
- portfolio value
- realized profit and loss
- number of trades

Your backtest should include:

- transaction costs
- slippage assumption

Assume:

transaction_cost = 0.01% per trade

You may choose a reasonable slippage model.

---

5. Performance Evaluation

Your final system must calculate the following metrics:

- Total Return
- Annualized Return
- Sharpe Ratio
- Maximum Drawdown
- Win Rate
- Average Trade Duration

Provide visualizations including:

- Price series
- Strategy signals
- Equity curve
- Drawdown curve

Constraints

To ensure fair evaluation:

1. Your solution must run in **under 30 seconds** on a standard laptop.
2. Hardcoding results is strictly prohibited.
3. Loops over large datasets should be avoided where possible.
4. Code must be modular and readable.

Deliverables

Please submit a GitHub repository containing:

```
project/
    data_loader.py
    strategy.py
    backtester.py
    analysis.py
    results/
    README.md
```

Your README should include:

- explanation of your strategy
- assumptions made
- limitations
- potential improvements

Limit your written explanation to **two pages maximum**.

Bonus Challenges

Candidates who complete the base task may optionally attempt the following:

1. Implement **parameter optimization** without overfitting.
2. Add **out-of-sample testing**.
3. Implement **multiple simultaneous pairs**.
4. Design **risk allocation across trades**.

Bonus work should be clearly separated from the main solution.

Evaluation Criteria

Your submission will be evaluated across the following dimensions.

Category	Weight
Code Quality	25%
Statistical Reasoning	25%
Strategy Design	20%
Efficiency & Performance	15%
Robustness to Data Issues	10%
Documentation	5%

Important Notes

This assignment intentionally contains **open-ended elements**.

We do not expect perfect performance.

Instead we look for:

- clear thinking
- good engineering practices
- thoughtful experimentation
- awareness of limitations

Academic Integrity

You may consult documentation and public resources.

However:

- your code must be your own
- do not use full pre-built backtesting frameworks
- large language model generated solutions without modification will be easily identifiable

---

Submission

Submit your GitHub repository link along with a short email describing:

- your approach
- total time spent
- the biggest challenge you encountered

Here's an **additional section you can insert into the document** to clarify the **AI usage policy** while still enforcing understanding and ownership. I wrote it in a **professional tone suitable for a hiring assignment document**.

Use of AI Tools

We recognize that modern developers frequently use **AI tools (such as code assistants or large language models)** as part of their workflow. For this assignment:

Use of AI tools is allowed and encouraged.

However, there is an important expectation:

You must remain the architect of your solution. AI should act as the worker, not the decision-maker.

This means:

You are responsible for:

- Designing the system architecture
- Choosing the modeling approach
- Defining the trading logic
- Implementing the evaluation framework
- Ensuring the correctness of the implementation

AI tools may assist with:

- writing boilerplate code
- debugging syntax errors
- explaining library usage
- accelerating development

However, you **must fully understand every line of code in your submission**.

During follow-up discussions or interviews, you may be asked to:

- Explain specific parts of your code
- Modify your implementation live
- Justify design decisions
- Identify weaknesses or limitations in your strategy

If you cannot explain or modify your own code, it will be assumed that the implementation was **not authored or understood by you**.

AI Usage Disclosure

In your README.md, include a short section titled:

AI Assistance

Briefly describe:

- Which tools you used (if any)
- What parts of the project they helped with

- What design decisions were made independently

Example format:

AI Assistance

Tools used: ChatGPT, GitHub Copilot

Used for:

- pandas syntax assistance

- debugging vectorized operations

All strategy design, architecture decisions, and evaluation logic were implemented independently.

This will **not negatively affect evaluation**. Transparency is encouraged.

Follow-Up Technical Discussion

Shortlisted candidates will participate in a **technical discussion** where they may be asked to:

- walk through their architecture
- implement a small modification
- debug a component of their system
- extend the strategy