### **Dataset Exploration**

* We plotted the entire dataset, revealing a dense and fluctuating spread landscape.
* In the **zoomed-in chart** (100 timestamps), we observed clearer price volatility and visible bid-ask spread changes.
* Price levels were mostly centered around a base price of ~10,000 units, with bid-ask spreads varying frequently but consistently.

This visual exploration helped identify key market behavior: **spread persistence with noisy mid-price oscillation.**

**Strategy Inspiration**

The strategy is inspired by **statistical arbitrage**, particularly **mean-reversion**. By tracking the **moving average of the mid-price** and its **spread standard deviation**, the bot identifies:

* Overpriced asks to sell into.
* Underpriced bids to buy from.

This exploits short-term inefficiencies in the spread while maintaining symmetric exposure.

### **Core Logic**

1. **Track bid/ask prices** at each timestamp:  
   * Compute mid\_price = (best\_bid + best\_ask) / 2
   * Compute spread = best\_ask - best\_bid
   * Record mid-price and spread in history.
2. **Compute statistical boundaries**:  
   * Use np.mean(mid\_h) as the fair mid-price.
   * Use np.std(spread\_h) as an indicator of volatility.
3. **Order placement logic**:  
   * **Buy** when price is below the lower bound → buy\_price = min(mean(mid\_h) - spread\_std, best\_ask)
   * **Sell** when price is above upper bound → sell\_price = max(mean(mid\_h) + spread\_std, best\_bid)

This logic adapts to market conditions and uses recent history to detect favorable price deviations.

**Experiments & Insights**

* **Zooming into 100 timestamps** confirmed that bid/ask prices fluctuate enough to frequently trigger spread-based trades.
* Using **spread standard deviation** as a dynamic threshold avoids rigid rules and adapts to volatility.