OneChronos Analytics
Technical Challenge

This challenge evaluates your ability to work with multidimensional market microstructure data and communicate what patterns you see. A successful submission will involve:

- Constructing your dataset efficiently and effectively
- Using quantitative methods to understand what your dataset says, and doesn't say
- Writing clear code (in any language you like) with documentation and modularity
- Presenting your findings with clarity and precision

Submission Guidelines

- Please complete and return this challenge **within three hours of receipt.**
- Please submit a written document with your analysis and all code/scripts used. You are welcome to use any programming language. Please ensure you make clear how your code contributes to your results.
- A GitHub repository with your work is welcome, but optional.

Data

You are given one trading day's worth of Level I **quote data** (top-of-book quotes) and **trade data** for four equity instruments:

Symbol ID	**Ticker**	**Description**
2178	CMG	Chipotle Mexican Grill (Consumer Discretionary)
7614	NVDA	NVIDIA Corporation (Technology/Semiconductors)
9574	SIRI	Sirius XM Holdings (Communication Services)
10407	TLT	iShares 20+ Year Treasury Bond ETF (Fixed Income)

Quotes Dataset (`quotes.csv`)

National Best Bid and Offer (NBBO) snapshots representing the consolidated top-of-book at each update event. Each row is characterized by:

- `transaction_timestamp`: time of quote (nanoseconds, UTC)
- `symbol_id`: identifier of security quoted
- `bid_price`: best (highest) buy price at time of quote
- `bid_size`: number of shares that could be bought at the bid price
- `ask_price`: best (lowest) sell price at time of quote
- `ask_size`: number of shares that could be sold at the ask price

- `price_decimal`: decimal scaling factor for prices (price = raw value $\times 10^{-\text{price\_decimal}}$). If `bid_price` = 123400 and `price_decimal` = 4, then the actual bid is \$12.34.

Trades Dataset (`trades.csv`)

Individual trade executions reported across market:

- `transaction_timestamp`: time of execution (nanoseconds, UTC)
- `symbol_id`: identifier of security executed
- `trade_exchange_code`: exchange or facility where the execution occurred
- `trade_price`: execution price (raw integer, apply `price_decimal`)
- `trade_size`: number of shares executed
- `price_decimal`: decimal scaling factor for prices (see above).

Challenge

Part 1: Analyze

- Ingest both datasets into a reasonably performative data structure (DataFrame, database, etc.).
- Construct a derived dataset that enriches each trade with the prevailing NBBO at execution time. You'll want to perform an *as-of* (point-in-time) join to match each trade with the **most recent quote relative to the trade timestamp.** See the end of this document for more information on as-of joins.

Part 2: Synthesize

Use your derived dataset to characterize trading behavior across these four symbols. The questions below are meant to guide your thinking. **Feel free to explore additional angles you find interesting. There is no ideal question or ideal answer to this challenge. Tell us what interests you about the data, and why.**

- How are these instruments quoted relative to one another? What explains any differences?
- How are these instruments traded relative to one another? What explains any differences?
- If you were an institutional investor and this data is your worldview, how would you trade these instruments? Why would you trade them that way? Are there market conditions in which you would want to trade these instruments? Are there market conditions in which you wouldn't?
- If you were a trading venue designing a marketplace where these instruments trade, and this data is your worldview, how would you design your marketplace? Why would you design it that way?

If the time constraint gets in your way, please briefly explain what you would do if you had more time.

As-Of Joins

An *as-of join* matches each record in one dataset to the most recent record in another dataset based on a shared column.

For this task, you'll want to match each trade to the most recent quote that occurred at or before the trade timestamp. You'll need to choose the correct join strategy that accomplishes this.

Both implementations require sorted data. The `by` parameter allows you to partition the join by additional columns (you probably shouldn't join trade and quote data for different symbols!)

As-of joins can be implemented in pandas:

```python
import pandas as pd

left = left.sort_values("SORT_COLUMN")
right = right.sort_values("SORT_COLUMN")

out = pd.merge_asof(
    left,
    right,
    on="SORT_COLUMN",
    by="OPTIONAL_COLUMN",
    direction="DIRECTION"
)
```

Or in polars:

```python
import polars as pl

left = left.sort("SORT_COLUMN")
right = right.sort("SORT_COLUMN")

out = left.join_asof(
    right,
    on="SORT_COLUMN",
    by="OPTIONAL_COLUMN",
    strategy="DIRECTION"
)
```

More details on these functions are available here:

- `https://pandas.pydata.org/docs/reference/api/pandas.merge_asof.html`
- `https://docs.pola.rs/api/python/stable/reference/dataframe/api/polars.DataFrame.join_asof.html`