**The Merge Strategy:**

*# Process each timeframe in this order*

timeframe\_order = ['daily', '1hour', '30min', '15min', '5min', '1min']

The code follows this logic:

**1. Start with 1-minute data as base (if available):**

Python -;

if '1min' in dfs\_by\_timeframe:

result = dfs\_by\_timeframe['1min'].copy()

**2. Fill gaps using lower frequency data:**

For each timeframe (starting from daily), it:

* Resamples the data to 1-minute frequency
* Uses .update() to fill only the gaps, not overwrite existing data

*# This only fills gaps, doesn't overwrite existing values*

Python -> result[col].update(resampled)

**3. Example scenario:**

Let's say for ES you have:

* **1min data**: 2015-2025 (but missing some days)
* **5min data**: 2010-2025
* **Daily data**: 2007-2025

The merge process:

1. Starts with 1min data (2015-2025)
2. Uses 5min data to fill gaps in the 1min data
3. Uses daily data to fill any remaining gaps (especially 2007-2014)
4. Forward fills to create continuous 1-minute bars

**Visual Example:**

Time 1min 5min Daily -> Final Result

09:30:00 4500 4500 4498 4500 (uses 1min)

09:31:00 4501 - - 4501 (uses 1min)

09:32:00 NaN - - 4501 (forward filled)

09:33:00 NaN - - 4501 (forward filled)

09:34:00 NaN - - 4501 (forward filled)

09:35:00 NaN 4502 - 4502 (uses 5min)

...

[Gap in data]

2010-01-04 NaN NaN 3500 3500 (uses daily)

**Why this approach?**

1. **Maximum data coverage**: Uses all available data sources
2. **Highest granularity where possible**: Prefers 1-minute data when available
3. **No data loss**: Doesn't throw away lower frequency data
4. **Historical completeness**: Daily data often goes back further than intraday

**The forward fill:**

*# After merging all timeframes, forward fill*

Python -> result = result.ffill()

This ensures continuous 1-minute data even when the actual trades might have gaps.