Strategy

การเลือกกลยุทธ์ นอกจากแนวคิดพื้นฐานว่าควรจะพัฒนากลยุทธ์การเทรดบนพื้นฐานแบบไหน เช่น HFT, Mean Reavers หลังจากนั้นจะเป็นการลองโค้ดดูว่าแต่ละกลยุทธ์เมื่อใช้ร่วมกับ Signal หรือ Tachical หรือ Indicator บางอย่างจะได้ผลลัพธ์ดีหรือไม่อย่างไร เพื่อหากลยุทธ์ที่เหมาะสมกับพฤติกรรมของหุ้นตัวนี้มากที่สุด เช่น Mean Reavers + Bollinger Band, Mean Reavers + OFI, etc. จนได้กลยุทธ์ที่โอเคในความเห็นของฉัน

Plan of Attack

เหตุผลในการเลือกกลยุทธ์ จากการวิเคราะห์ EDA ทำให้เลือกกลยุทธ์พื้นฐานบางตัวมาเพื่อทดลองทดสอบ จากทั้งหมด Mean Reversion น่าสนใจที่สุด ทั้งในแง่การวิเคราะห์ EDA หรือพฤติกรรมของหุ้นไทย รวมถึงคำถามในแบบทดสอบก่อนหน้า ว่าบริษัทน่าจะมีการ concern เกี่ยวกับประเด็ก latency อยู่บ้าง การใช้ Mean Reversion ที่ไม่ได้ต้องการความเร็วที่สูงมาก เท่ากับพวก HFT อาจจะเป็นทางเลือกที่ดีกว่า กลยุทธ์ที่พัฒนาขึ้นคือ Order Flow Imbalance Matching - Reverse (OMR) ซึ่งอาศัยแนวคิดของ Order Flow Imbalance (OFI) ในการหาสัญญาณซื้อขาย แต่แตกต่างจากแนวทางดั้งเดิมโดยใช้ Reverse Strategy แทนที่จะเทรดตามแนวโน้มของ OFI เราใช้กลยุทธ์ สวนทางกับแรงดันตลาด (Contrarian Approach) กลยุทธ์นี้เหมาะสำหรับตลาดที่มี Mean Reversion หรือที่มีการเคลื่อนไหวแบบ Noise มากกว่าการมี Trend แข็งแกร่ง ที่จากการทดลองรอบก่อนๆมีการใช้ OFI ธรรมดาแล้วมีเปอเซนต์ชนะน้อยมาก ถ้ามองกลับกันก็หมายความว่าถ้าใช้แบบกลับด้านอาจจะมีผลลัพธ์ที่ดี เนื่องจากพฤติกรรมของหุ้นตัวนี้ ไม่สามารถเลือกเทรนได้ มีพฤติกรรม Mean Reversion แม้จะมี ออเดอร์ขนาดใหญ่เข้ามาก็ตาม แสดงว่ามีสภาพคล่องที่สูงมาก และเกิดการเปลี่ยนแปลงระดับของราคาได้ ราคาจะไม่ผันผวนมากในแค่ของมูลค่า Market Inefficiencies ที่ใช้ประโยชน์ Liquidity Frictions & Order Book Imbalance → ใช้ข้อมูล Market Depth เพื่อดูว่าฝั่งไหนมี Order Imbalance มากเกินไป Short-Term Reversals → กลยุทธ์นี้อาศัยการที่ราคาเด้งกลับจาก OFI ที่สูงหรือต่ำกว่าค่าปกติ Slippage & Market Impact Control → จำกัดขนาดการซื้อขายที่ 30% ของ Volume ใน LOB เพื่อไม่ให้ Order กระทบต่อตลาดมากเกินไป

หลักการของกลยุทธ์

1. Reverse Order Flow Imbalance (OFI) ปกติ OFI ใช้วิเคราะห์แรงซื้อ/ขายจากการเปลี่ยนแปลงของปริมาณ Bid และ Ask แต่ในกลยุทธ์นี้ใช้ Reverse OFI ซึ่งเป็นการสวนทางกับสัญญาณปกติ หากตลาดมีแรงซื้อเข้า (OFI > Quantile 80%) → มีแรงซื้อมาก → ราคามีโอกาสมา Triger ที่ Ask1 สูง → เปิดสถานะ Short, ปิด Long ที่ราคาสูง หรือ Ask ได้ หากตลาดมีแรงขายเข้า (OFI < Quantile 30%) → ทีแรงขายมาก → ราคามีโอกาสมา Triger ที่ Bid1 สูง → เปิดสถานะ Long, ปิดสถานะ Short ที่ราคาต่ำ หรือ Bid ได้ 2. Mean Reversion Strategy กลยุทธ์นี้ใช้สมมติฐานว่าราคามีแนวโน้มจะกลับเข้าสู่ค่าเฉลี่ย (Mean) เมื่อเกิดความผันผวนสูง ใช้ค่า Quantile (เปอร์เซ็นไทล์) ของ OFI เพื่อกำหนดขอบเขตการเข้าเทรด กำหนดให้ Quantile 80% เป็นจุดเข้า Short และ Quantile 30% เป็นจุดเข้า Long 3. Execution ผ่าน Market Depth (VWAP Matching) ใช้ Volume-Weighted Average Price (VWAP) เพื่อคำนวณราคาซื้อ-ขายที่แท้จริง จำลองการซื้อขายโดยคำนวณจาก Bid/Ask Depth ของตลาด 4. การปิดสถานะ เมื่อได้รับสัญญาณตรงข้าม → ปิดสถานะเดิมก่อน แล้วเปิดสถานะใหม่ หากถึง สิ้นวัน และยังมีสถานะค้างอยู่ → ปิดทั้งหมดเพื่อป้องกัน Overnight Risk

การจำลองระบบการซื้อขาย (Trade Execution Simulation)

- ระบบนี้ใช้ข้อมูลตลาดจากไฟล์ CSV ซึ่งมีรายละเอียดของ Orderbook เช่น Timestamp Bid Ask Price - ใช้ Market Depth Matching ในการจำลองการซื้อขาย โดยใช้ราคาที่เกิดขึ้นจริงในตลาด - ระบบคำนวณเงินทุน (Capital Management) แบ่งเงินออกเป็น "Capital" และ "Cash" Capital → มูลค่ารวมของพอร์ต (Cash + มูลค่าสัญญา) Cash → เงินสดที่ใช้เปิดสัญญา (จำกัดการซื้อขายให้ไม่เกิน Cash) เมื่อเปิดสถานะ Capital จะลดลงเฉพาะค่าธรรมเนียม แต่ Cash จะลดลงตามต้นทุนที่ใช้ซื้อ เมื่อปิดสถานะ Cash จะได้รับคืนจากการขาย และ Capital จะปรับตามกำไรขาดทุนจริง - การคำนวณต้นทุนที่สมจริง (Realistic Cost Calculation) มีการคำนวณค่าธรรมเนียมซื้อขายตาม อัตรา Commission Rate (0.00007) ที่แท้จริงของบุคคลธรรมดาในตลาด TFEX ซึ่งถ้าเป็นบริษัท Quantfilm อาจจะได้เรทที่ดีกว่านี้ และกลยุทธ์อาจจะได้ผลลัพธ์ดีกว่านี้ - ใช้ Market Depth และ VWAP (Volume Weighted Average Price) เพื่อให้ได้ราคาซื้อขายตามสภาพคล่องของตลาด - ไม่ใช้ราคา Mid-price หรือราคาปัจจุบันที่อาจทำให้เกิด Backtest Leakage - ระบบบังคับปิดสถานะ (Forced Liquidation) หากหมดวัน (End of Day) จะบังคับปิดสถานะทั้งหมด ใช้วิธีการ Match ราคากับ Market Depth จริง ก่อนปิด - ระบบนี้ป้องกัน Overnight Risk และ Slippage ที่ไม่สามารถจำลองได้ ยังไม่รองรับ - การคำนวณการปิดหลายออเดอร์ปริมาณมาก ๆ พร้อมกันที่อาจจะเกิดโวลุ่มไม่พอได้ แต่กับบรรเท่าได้ด้วยการ ใช้สัญญาณ OPEN_LONG = CLOSE_SHORT , OPEN_SHORT = CLOSE_LONG - ไม่มีการคำนวณ Slippage เพราะมีการใช้ Market Depth ป้องกันปัญญาเบื้องต้น และค่าเฉลี่ย Slippage มีค่าต่ำมากจนไม่ต้องนำมาคิดในการจำลองก็ได้ - ไม่มีการจำลอง Market Impact ด้วยเหตุผลเดียวกันคือค่าเฉลี่ย Market impact มีค่าต่ำมากจนอาจจะไม่มีผลต่อการจำลอง แต่ก็มีความเสี่ยงในกรณีที่มีออเดอร์ที่ใหญ่มากกว่าปกติมากๆ - ไม่มีการจำลอง Latency เพราะไม่มีข้อมูลเพียงพอ ถือเป็นหนึ่งในสาเหตุที่ใช้ MeanRevers เพราะไม่ใช่กลยุทธ์ที่ต้องใช้ความเร็วมาก และเหมาะสมกับ Home trade ดังนั้นไม่ใช้

ปัญหาสำหรับบริษัทที่มีสภาพแวดล้อมแบบ QuantFirm - ไม่มีการจำลองการถูกแย่งซื้อจากบุคคนอื่น เช่น การสุ่มลดโวลุ่มเมื่อเกิดการเทรด

Strategy Implementation & Backtesting

การออกแบบกลยุทธ์ ใช้ OFI Quantiles (q80, q20) เป็น Thresholds เมื่อ OFI > q80 → เปิดสถานะ Short เมื่อ OFI < q20 → เปิดสถานะ Long ใช้ Market Depth Matching เพื่อให้แน่ใจว่าราคาที่ใช้ซื้อขายเป็นราคาจริงจากตลาด จำกัด Order ที่ 30% ของ Volume ที่ระดับราคาดีที่สุด(Lavel 1) เพื่อป้องกัน Slippage และ Market Impact การทดสอบกลยุทธ์ (Backtesting) แยกข้อมูลเป็น In-Sample และ Out-of-Sample ใช้ 8 วันแรกเป็น In-Sample และ 3 วันสุดท้ายเป็น Out-of-Sample นำค่าธรรมเนียม (Commission), Slippage และ Market Impact มาคำนวณจริง ใช้ VWAP-Based Execution → ซื้อขายที่ราคาเฉลี่ยของ Order Book ตาม Volume ที่ตลาดรองรับ มี Forced Liquidation ก่อนหมดวัน → บังคับปิดสถานะทุกวันเพื่อลด Overnight Risk Performance Metrics ที่วัดผล Final PnL = กำไรขาดทุนสุดท้าย ROI (%) = ผลตอบแทนคิดเป็นเปอร์เซ็นต์ Maximum Drawdown (MDD) = ขนาดการลดลงสูงสุดของพอร์ต Sharpe Ratio = อัตราส่วนผลตอบแทนต่อความเสี่ยง Profit Factor = อัตราส่วนของกำไรต่อขาดทุน Win Rate = อัตราการชนะของกลยุทธ์ Total Trades = จำนวนธุรกรรมทั้งหมด Total Slippage Cost = ค่าที่เกิดจาก Slippage (ต่ำมาก) Total Fee Cost = ค่าธรรมเนียมทั้งหมดที่เสียไป

Code

โครงสร้างโค้ด Load Data: โหลดข้อมูลตลาดจากไฟล์ CSV Generate Signals: คำนวณ Order Flow Imbalance (OFI) และกำหนดสัญญาณซื้อขาย Portfolio Management: บริหารสถานะพอร์ต (เปิด-ปิดสถานะ) VWAP Execution: ใช้ Volume Weighted Average Price (VWAP) เพื่อคำนวณราคาซื้อขาย Logging & Performance Metrics: บันทึกธุรกรรมและวิเคราะห์ผลลัพธ์หลังการ BacktestFunction : get_vwap_and_levels - คำนวณราคา VWAP และระดับของออเดอร์ที่ถูกใช้ - ใช้ข้อมูลปริมาณออเดอร์ในระดับต่าง ๆ ของ Bid หรือ Ask - ไล่ระดับราคาเพื่อให้ได้ VWAP (Volume Weighted Average Price) process_file - ประมวลผลข้อมูลการซื้อขายย้อนหลังจากไฟล์ CSV - โหลดข้อมูลและตรวจสอบว่ามีคอลัมน์ที่จำเป็นครบหรือไม่ - คำนวณค่า OFI และกำหนดสัญญาณซื้อขาย (Signal) - บริหารสถานะพอร์ต (ปิดสถานะที่เปิดอยู่และเปิดสถานะใหม่ตามสัญญาณ) - บันทึกผลลัพธ์การซื้อขายและคำนวณค่าประสิทธิภาพของกลยุทธ์Order Flow Imbalance (OFI) คำนวณนจาก df["OFI"] = (df["BidTrade"] * df["BidVolume"]) - (df["AskTrade"] * df["AskVolume"]) VWAP Execution เพื่อให้การซื้อขายเกิดขึ้นที่ราคาถัวเฉลี่ยของปริมาณออเดอร์ที่มีอยู่ Maximum Drawdown - MDD คำนวณจากความกว้างสูงสุดที่เคยเกิดจากสุดสูงสุดไปยังจุดต่ำสุด ณ ช่วงเวลาหนึ่งๆ Final PnL - Net Profit ดดยรวมของวัน ROI_IN (%) - กำไรทั้งหมดเมื่อคิดเป็น % จากเงินต้นทั้งหมด Final Capital - เงินสุทธิทั้งหมด Sharpe Ratio - Mean_return/std_return Profit Factor - total_profit/total_loss Win-Rate - จำนวนการเทรดที่ชนะทั้งหมด(แบบหักค่าธรรมเนียมทั้งค่าเปิดและปิดเรียบร้อนแล้ว) Total Trades - จำนวนการเทรดแบบแยกเป้นการเปิดและปิดสัญญา Total Fee Cost - ค่าธรรมเนียมทั้งหมด

UXUI

In [8]:
```python
import pandas as pd
import ipywidgets as widgets
import os
from IPython.display import display, clear_output

def create_csv_viewer(folder_path):
    """
    📌 ฟังก์ชันสำหรับสร้าง UI ดูไฟล์ CSV ในโฟลเดอร์ที่ระบุ
    folder_path: str -> โฟลเดอร์ที่เก็บไฟล์ CSV
    """

    # 📌 ดึงรายชื่อไฟล์ทั้งหมดจากโฟลเดอร์
    file_list = [f for f in os.listdir(folder_path) if f.endswith(".csv")]

    # 📌 ตัวเลือกเลือกไฟล์และจำนวนแถว
    file_picker = widgets.Dropdown(
        options=file_list,
        value=file_list[0] if file_list else None,
        description="Select File:",
```

```
        style={'description_width': 'initial'}
    )

    rows_per_page = widgets.Dropdown(
        options=[5, 10, 20, 50, 100],
        value=10,
        description="Rows per Page:",
        style={'description_width': 'initial'}
    )

    # 📌 ตัวแปรเก็บข้อมูลของตาราง
    df = pd.DataFrame()
    current_page = 0

    # 📌 ปุ่มเปลี่ยนหน้า
    prev_button = widgets.Button(description="⏮ Previous", disabled=True)
    next_button = widgets.Button(description="Next ⏭")
    page_label = widgets.Label(value="Page 1")
    jump_to_page = widgets.IntText(value=1, description="Jump to Page:", mir
    jump_button = widgets.Button(description="Go")

    # 📌 พื้นที่แสดงข้อมูล
    output = widgets.Output()

    def load_data():
        """โหลดข้อมูลจากไฟล์ CSV ที่เลือก"""
        nonlocal df, current_page
        file_path = os.path.join(folder_path, file_picker.value)
        df = pd.read_csv(file_path)
        current_page = 0
        update_table()

    def update_table():
        """อัปเดตข้อมูลในตาราง"""
        with output:
            clear_output(wait=True)
            start = current_page * rows_per_page.value
            end = start + rows_per_page.value
            display(df.iloc[start:end])

            # 📌 อัปเดตหมายเลขหน้า
            total_pages = max(1, (len(df) - 1) // rows_per_page.value + 1)
            page_label.value = f"Page {current_page + 1} of {total_pages}"

            # 📌 อัปเดตค่าต่ำสุดสูงสุดของ Jump to Page
            jump_to_page.min = 1
            jump_to_page.max = total_pages

            # 📌 อัปเดตสถานะปุ่ม
            prev_button.disabled = current_page == 0
            next_button.disabled = end >= len(df)

    def prev_page(_):
        """ย้อนกลับไปหน้าก่อนหน้า"""
        nonlocal current_page
        if current_page > 0:
```

```
            current_page -= 1
            update_table()

    def next_page(_):
        """ไปยังหน้าถัดไป"""
        nonlocal current_page
        if (current_page + 1) * rows_per_page.value < len(df):
            current_page += 1
            update_table()

    def jump_to_selected_page(_):
        """ข้ามไปยังหน้าที่ต้องการ"""
        nonlocal current_page
        total_pages = max(1, (len(df) - 1) // rows_per_page.value + 1)
        if 1 <= jump_to_page.value <= total_pages:
            current_page = jump_to_page.value - 1
            update_table()

    # 📌 เชื่อมโยงปุ่มกับฟังก์ชัน
    prev_button.on_click(prev_page)
    next_button.on_click(next_page)
    jump_button.on_click(jump_to_selected_page)

    # 📌 อัปเดตเมื่อเลือกไฟล์หรือเปลี่ยนจำนวนแถว
    def on_change(change):
        load_data()

    file_picker.observe(on_change, names="value")
    rows_per_page.observe(on_change, names="value")

    # 📌 จัดวาง UI
    ui_top = widgets.HBox([file_picker, rows_per_page], layout=widgets.Layou
    ui_bottom = widgets.HBox([
        prev_button,
        page_label,
        next_button,
        jump_to_page,
        jump_button
    ], layout=widgets.Layout(justify_content="flex-end"))

    display(ui_top, output, ui_bottom)
    load_data()
```

MeanReversion - ReverseOrderFlowImbalance - Strategy

In [19]:

```
import pandas as pd
import numpy as np
import os


file_paths = {
    "2024-05-02": "split_data/S50M24_2024-05-02.csv",
```

```
    "2024-05-03": "split_data/S50M24_2024-05-03.csv",
    "2024-05-07": "split_data/S50M24_2024-05-07.csv",
    "2024-05-08": "split_data/S50M24_2024-05-08.csv",
    "2024-05-09": "split_data/S50M24_2024-05-09.csv",
    "2024-05-10": "split_data/S50M24_2024-05-10.csv",
    "2024-05-13": "split_data/S50M24_2024-05-13.csv",
    "2024-05-14": "split_data/S50M24_2024-05-14.csv",
}

# 📌 โฟลเดอร์เก็บ Log
log_folder = "results/Trade_Logs"
transaction_folder = "results/Transaction_Pairs"
os.makedirs(log_folder, exist_ok=True)
os.makedirs(transaction_folder, exist_ok=True)

# 📌 ฟังก์ชันสำหรับไล่ระดับราคา (Market Depth Matching) VWAP สำหรับ โวลุ่มและราคาของอ
def get_vwap_and_levels(volume_needed, price_levels, volume_levels, side):
    total_value = 0
    total_size = 0
    used_levels = []

    for i in range(len(price_levels)):
        price = price_levels[i]
        available_volume = volume_levels[i]

        if available_volume <= 0:
            continue

        size_to_use = min(volume_needed, available_volume)
        total_value += size_to_use * price
        total_size += size_to_use
        used_levels.append(side + str(i + 1))

        volume_needed -= size_to_use
        if volume_needed <= 0:
            break

    vwap_price = total_value / total_size if total_size > 0 else 0
    return vwap_price, used_levels

# 📌 ฟังก์ชันสำหรับประมวลผลไฟล์
def process_file(file_path, date):
    df = pd.read_csv(file_path, low_memory=False)

    if "timestamp" in df.columns:
        df.rename(columns={"timestamp": "Timestamp"}, inplace=True)

    required_columns = {"BidTrade", "AskTrade", "BidVolume", "AskVolume", "E
                        "vBid1", "vAsk1", "Bid2", "vBid2", "Bid3", "vBid3",
    if missing_columns := required_columns - set(df.columns):
        print(f"❌ Skipping {file_path} - Missing columns: {missing_columns}
        return None

    df.fillna({"BidTrade": 0, "AskTrade": 0, "BidVolume": 0, "AskVolume": 0}
```

```python
#OFI คำนวณ จากทั้งหมดอาจเกิด data Leak
#df["OFI"] = (df["BidTrade"] * df["BidVolume"]) - (df["AskTrade"] * df["
#q80, q20 = df["OFI"].quantile(0.80), df["OFI"].quantile(0.2)

#df["Signal"] = 0
#df.loc[df["OFI"] > q80, "Signal"] = 1
#df.loc[df["OFI"] < q20, "Signal"] = -1

# คำนวณ OFI
df["OFI"] = (df["BidTrade"] * df["BidVolume"]) - (df["AskTrade"] * df["A

# เพิ่มคอลัมน์สำหรับ Rolling Quantile
df["q80"] = None
df["q20"] = None
df["Signal"] = 0
df["q80"] = df["OFI"].expanding().quantile(0.80)
df["q20"] = df["OFI"].expanding().quantile(0.20)

df["Signal"] = 0
df.loc[df["OFI"] > df["q80"], "Signal"] = 1
df.loc[df["OFI"] < df["q20"], "Signal"] = -1



log_file_path = os.path.join(log_folder, f"Trade_Log_{date}.csv")
transaction_file_path = os.path.join(transaction_folder, f"Transaction_F

#ระบบซื้อขายและบันทึกผล
log_data = []
transaction_data = []

open_long_positions = []
open_short_positions = []
commission_rate = 0.00007
initial_capital = 1_000_000
capital = initial_capital
cash = initial_capital
peak_capital = initial_capital
total_fees = 0

for row in df.itertuples(index=False):
    signal, timestamp = row.Signal, row.Timestamp

    bid_prices, bid_volumes = [row.Bid1, row.Bid2, row.Bid3, row.Bid4],
    ask_prices, ask_volumes = [row.Ask1, row.Ask2, row.Ask3, row.Ask4],

    #วนลูปตาม Signal
    if signal == -1:
        if open_short_positions:
            for pos in open_short_positions:
                vwap_price, used_levels = get_vwap_and_levels(pos["Size"
                profit = (pos["EntryPrice"] - vwap_price) * pos["Size"]
                fee_close = pos["Size"] * vwap_price * commission_rate
                net_profit = profit - (pos["Fee"] + fee_close)
                capital += profit - fee_close
                cash -= pos["Size"] * vwap_price + fee_close
```

```
                total_fees += fee_close
                peak_capital = max(peak_capital, capital)
                drawdown = peak_capital - capital

                log_data.append([timestamp, "CLOSE_SHORT", pos["Size"],
                transaction_data.append([pos["Timestamp"], timestamp, pc

            open_short_positions.clear()
        else:
            size = 0.3 * row.vBid1
            vwap_price, used_levels = get_vwap_and_levels(size, bid_pric
            fee = size * vwap_price * commission_rate

            if cash >= size * vwap_price + fee: # Check cash
                cash -= size * vwap_price + fee
                capital -= fee
                total_fees += fee
                peak_capital = max(peak_capital, capital)
                drawdown = peak_capital - capital

                open_long_positions.append({"Size": size, "EntryPrice":
                log_data.append([timestamp, "OPEN_LONG", size, vwap_pric

    elif signal == 1:
        if open_long_positions:
            for pos in open_long_positions:
                vwap_price, used_levels = get_vwap_and_levels(pos["Size"
                profit = (vwap_price - pos["EntryPrice"]) * pos["Size"]
                fee_close = pos["Size"] * vwap_price * commission_rate
                net_profit = profit - (pos["Fee"] + fee_close)
                capital += profit - fee_close
                cash += pos["Size"] * vwap_price - fee_close
                total_fees += fee_close
                peak_capital = max(peak_capital, capital)
                drawdown = peak_capital - capital

                log_data.append([timestamp, "CLOSE_LONG", pos["Size"], v
                transaction_data.append([pos["Timestamp"], timestamp, pc

            open_long_positions.clear()
        else:
            size = 0.3 * row.vAsk1
            vwap_price, used_levels = get_vwap_and_levels(size, ask_pric
            fee = size * vwap_price * commission_rate

            if cash >= size * vwap_price + fee:
                cash += size * vwap_price - fee  # เปิด Short ต้องได้เงินสดเ
                capital -= fee
                total_fees += fee
                peak_capital = max(peak_capital, capital)
                drawdown = peak_capital - capital

                open_short_positions.append({"Size": size, "EntryPrice":
                log_data.append([timestamp, "OPEN_SHORT", size, vwap_pri

# ✅ บังคับปิดสถานะทั้งหมดก่อนหมดวัน
```

```python
    if open_long_positions or open_short_positions:
        final_row = df.iloc[-1]
        bid_prices = [final_row.Bid1, final_row.Bid2, final_row.Bid3]
        bid_volumes = [final_row.vBid1, final_row.vBid2, final_row.vBid3]

        ask_prices = [final_row.Ask1, final_row.Ask2, final_row.Ask3]
        ask_volumes = [final_row.vAsk1, final_row.vAsk2, final_row.vAsk3]

        # ✅ บังคับปิด LONG
        for pos in open_long_positions:
            vwap_price, used_levels = get_vwap_and_levels(pos["Size"], ask_p
            profit = (vwap_price - pos["EntryPrice"]) * pos["Size"]
            fee = pos["Size"] * vwap_price * commission_rate
            net_profit = profit - (pos["Fee"] + fee)

            capital += profit - fee
            cash += pos["Size"] * vwap_price - fee  # เงินสดกลับเข้า Cash
            total_fees += fee
            peak_capital = max(peak_capital, capital)
            drawdown = peak_capital - capital

            log_data.append([timestamp, "FORCED_CLOSE_LONG", pos["Size"], vw
            transaction_data.append([pos["Timestamp"], timestamp, pos["Size"

        # ✅ บังคับปิด SHORT
        for pos in open_short_positions:
            vwap_price, used_levels = get_vwap_and_levels(pos["Size"], bid_p
            profit = (pos["EntryPrice"] - vwap_price) * pos["Size"]
            fee = pos["Size"] * vwap_price * commission_rate
            net_profit = profit - (pos["Fee"] + fee)

            capital += profit - fee
            cash -= pos["Size"] * vwap_price + fee  # หักเงินสดออกตอนปิด Short
            total_fees += fee
            peak_capital = max(peak_capital, capital)
            drawdown = peak_capital - capital

            log_data.append([timestamp, "FORCED_CLOSE_SHORT", pos["Size"], v
            transaction_data.append([pos["Timestamp"], timestamp, pos["Size"

    # ✅ คำนวณ Maximum Drawdown (MDD)
    #df_log = pd.DataFrame(log_data, columns=["Timestamp", "Type", "Size", "


    df_log = pd.DataFrame(log_data, columns=["Timestamp", "Type", "Size", "F
    max_drawdown = df_log["Drawdown"].max()
    df_transaction = pd.DataFrame(transaction_data, columns=["Timestamp_Oper
    df_log.to_csv(log_file_path, index=False)
    df_transaction.to_csv(transaction_file_path, index=False)

    #Mean_return = df_log["Profit"].mean()
    Mean_return = df_transaction["Net_Profit"].mean()

    #std_return = df_log["Profit"].std()
    std_return = df_transaction["Net_Profit"].std()
```

```
    #total_profit = df_log[df_log["Profit"] > 0]["Profit"].sum()
    total_profit = df_transaction[df_transaction["Net_Profit"] > 0]["Net_Pro

    #total_loss = df_log[df_log["Profit"] < 0]["Profit"].sum()
    total_loss = df_transaction[df_transaction["Net_Profit"] < 0]["Net_Profi

    Sharpe_Ratio = Mean_return/std_return
    ProfitFactor = int(total_profit)/(int(total_loss)*(-1))
    #Profit_count = df_log[df_log["Profit"] > 0]["Profit"].count() #แบบไม่หัก
    Profit_count = df_transaction[df_transaction["Net_Profit"] > 0]["Net_Pro
    #Loss_count = df_log[df_log["Profit"] < 0]["Profit"].count() #แบบไม่หักค่า
    Loss_count = df_transaction[df_transaction["Net_Profit"] < 0]["Net_Profi
    Win_rate = (Profit_count)/(Profit_count+Loss_count)

    Total_Trades = len(df_log)

    #Total_Capital_used = df_transaction["Size"] * df_transaction["Price_Ope
    #Total_Capital_used = total_capital_used.sum()

    #Slippage Cost
    df_transaction["Slippage"] = abs(df_transaction["Price_Open"] - df_trans
    Total_Slippage_Cost = df_transaction["Slippage"].sum()

    #Market Impact Cost
    #df_transaction["Market_Impact"] = abs(df_transaction["Price_Open"] - df
    #total_market_impact_cost = df_transaction["Market_Impact"].sum()
    #AVG
    #avg_market_impact = df_transaction["Market_Impact"].mean()


    #Commission & Fee Cost
    total_fee_cost = df_transaction["Fee_Open"].sum() + df_transaction["Fee_

    #Drawdown Duration ไม่ถูก และยังนึกไม่ออกว่าจะใช้อะไรคำนวณ
    #df_log["Peak_Capital"] = df_log["Capital"].cummax()
    #df_log["Drawdown_Flag"] = df_log["Capital"] < df_log["Peak_Capital"]
    #drawdown_duration = df_log["Drawdown_Flag"].sum()




    return {"Date": date,
            "Final PnL": round(capital - initial_capital, 2),
            "ROI_IN (%)": round((capital - initial_capital) * 100 / initial_
            "Final Capital": round(capital, 2),
            "MDD": round(max_drawdown, 2),
            "Sharpe Ratio" : round(Sharpe_Ratio, 2) ,
            "Profit Factor" : round(ProfitFactor, 2),
            "Win-Rate " : round(Win_rate, 2),
            "Total Trades" : round(Total_Trades, 2),
            "Total Slippage Cost " : round(Total_Slippage_Cost, 2),
            #"Total Market Impact Cost " : round(total_market_impact_cost, 2
            #"AVG Market Impact " : round(avg_market_impact, 2),
            "Total Fee Cost " : round(total_fee_cost, 2),
            #"Drawdown Duration " : round(drawdown_duration, 2),
           }
```

```
# 📌 ประมวลผล
df_results = pd.DataFrame([process_file(path, date) for date, path in file_p
df_results.to_csv("results/Performance_Report.csv", index=False)
#df_results
```

In [21]:
```
df_results
```

Out[21]:

	Date	Final PnL	ROI_IN (%)	Final Capital	MDD	Sharpe Ratio	Profit Factor	Win-Rate	Total Trades	T Slipp C
0	2024-05-02	15507.44	1.55	1015507.44	114.53	0.38	10.67	0.55	29704	2409
1	2024-05-03	15665.50	1.57	1015665.50	101.01	0.39	13.35	0.56	25598	207
2	2024-05-07	22039.66	2.20	1022039.66	183.03	0.33	7.41	0.50	39514	305
3	2024-05-08	16875.78	1.69	1016875.78	91.57	0.40	8.64	0.54	32536	259
4	2024-05-09	18528.08	1.85	1018528.08	164.84	0.44	9.80	0.55	32518	265
5	2024-05-10	11517.02	1.15	1011517.02	74.05	0.44	10.46	0.56	24150	196
6	2024-05-13	14293.06	1.43	1014293.06	69.99	0.38	12.53	0.59	25400	217
7	2024-05-14	17709.11	1.77	1017709.11	166.40	0.26	11.69	0.55	29156	236

In [23]:
```
create_csv_viewer("results/Trade_Logs/")
```

```
HBox(children=(Dropdown(description='Select File:', options=('Trade_Log_2024
-05-09.csv', 'Trade_Log_2024-05-08…
Output()
HBox(children=(Button(description='⏮ Previous', disabled=True, style=ButtonS
tyle()), Label(value='Page 1'), Bu…
```

In [25]:
```
create_csv_viewer("results/Transaction_Pairs")
```

```
HBox(children=(Dropdown(description='Select File:', options=('Transaction_Pa
ir_2024-05-08.csv', 'Transaction_P…
Output()
HBox(children=(Button(description='⏮ Previous', disabled=True, style=ButtonS
tyle()), Label(value='Page 1'), Bu…
```

ผลลัพธ์จากการเทรด - มีกำไรไม่สูงมากแต่สามารถทำกำไรได้ทุกวันซึ่งเหมาะกับการข้อมูลที่มีเพียง 10 วัน โดยที่ไม่สามารถใช้ข้อมูลอื่นเพิ่มเพื่อ หา Singnal ที่นานกว่านี้ได้ - MDD มีบางวันที่ค่อยข้างสูงแต่อยู่ในจุดที่รับได้เมื่อเทียบกับผลลัพธ์อาจจะต้องเพิ่ม SL เพื่อลด MDD หรือหากพฤติกรรมมีการเปลี่ยนแปลง - SR มีค่าต่ำมาก คือผลตอบแทนไม่คุ้มค่ากับความเสี่ยงแปลว่าเป็นกลยุทธ์ที่มีความ

เสี่ยงสูงมากกว่ากำไรที่ได้ - ProfitFactor อยู่ในระดับที่ค่อนข้างดี แปลว่าสามารถทำกำไรได้จริง - Total Trades และ Total Fee Cost ค่อนข้างสูงแต่ก้เหมาะกับการเทรดแบบ Quant Firm เพราะสามารถเพิ่ม สภาพคล่องของตลาดได้ หรือแม้แต่การที่ต่อรองค่าธรรมเนียมพิเศษกับโบรกเกอร์เนื่องจากทำการเทรดเยอะ

Blind Test Data

In [27]:
```python
import pandas as pd
import numpy as np
import os

file_paths = {
    "2024-05-15": "split_data/S50M24_2024-05-15.csv",
    "2024-05-16": "split_data/S50M24_2024-05-16.csv",
}

# 📌 โฟลเดอร์เก็บ Log
log_folder = "results/Trade_Logs_Blind_Test"
transaction_folder = "results/Transaction_Pairs_Blind_Test"
os.makedirs(log_folder, exist_ok=True)
os.makedirs(transaction_folder, exist_ok=True)

# 📌 ฟังก์ชันสำหรับไล่ระดับราคา (Market Depth Matching)
def get_vwap_and_levels(volume_needed, price_levels, volume_levels, side):
    total_value = 0
    total_size = 0
    used_levels = []

    for i in range(len(price_levels)):
        price = price_levels[i]
        available_volume = volume_levels[i]

        if available_volume <= 0:
            continue

        size_to_use = min(volume_needed, available_volume)
        total_value += size_to_use * price
        total_size += size_to_use
        used_levels.append(side + str(i + 1))

        volume_needed -= size_to_use
        if volume_needed <= 0:
            break

    vwap_price = total_value / total_size if total_size > 0 else 0
    return vwap_price, used_levels

# 📌 ฟังก์ชันสำหรับประมวลผลไฟล์
def process_file(file_path, date):
    df = pd.read_csv(file_path, low_memory=False)

    if "timestamp" in df.columns:
        df.rename(columns={"timestamp": "Timestamp"}, inplace=True)

    required_columns = {"BidTrade", "AskTrade", "BidVolume", "AskVolume", "E
```

```
                        "vBid1", "vAsk1", "Bid2", "vBid2", "Bid3", "vBid3",
    if missing_columns := required_columns - set(df.columns):
        print(f"❌ Skipping {file_path} - Missing columns: {missing_columns}
        return None

    df.fillna({"BidTrade": 0, "AskTrade": 0, "BidVolume": 0, "AskVolume": 0}

    #df["OFI"] = (df["BidTrade"] * df["BidVolume"]) - (df["AskTrade"] * df["
    #q80, q20 = df["OFI"].quantile(0.8), df["OFI"].quantile(0.2)

    #df["Signal"] = 0
    #df.loc[df["OFI"] > q80, "Signal"] = 1
    #df.loc[df["OFI"] < q20, "Signal"] = -1

    # คำนวณ OFI
    df["OFI"] = (df["BidTrade"] * df["BidVolume"]) - (df["AskTrade"] * df["A

    # เพิ่มคอลัมน์สำหรับ Rolling Quantile
    df["q80"] = None
    df["q20"] = None
    df["Signal"] = 0

    # ใช้ For Loop ไล่คำนวณ q80/q20 ตั้งแต่ Record แรก
    #for i in range(1, len(df) + 1):  # เริ่มจาก 1 ไปจนถึงแถวสุดท้าย
    #    past_data = df.iloc[:i]  # ใช้ข้อมูลตั้งแต่แถวแรกถึงปัจจุบัน
    #    df.loc[df.index[i - 1], "q80"] = past_data["OFI"].quantile(0.80)
    #    df.loc[df.index[i - 1], "q20"] = past_data["OFI"].quantile(0.20)

    # กำหนด Signal ตาม Quantile ที่คำนวณได้
    #df.loc[df["OFI"] > df["q80"], "Signal"] = 1
    #df.loc[df["OFI"] < df["q20"], "Signal"] = -1
    df["q80"] = df["OFI"].expanding().quantile(0.80)
    df["q20"] = df["OFI"].expanding().quantile(0.20)

    df["Signal"] = 0
    df.loc[df["OFI"] > df["q80"], "Signal"] = 1
    df.loc[df["OFI"] < df["q20"], "Signal"] = -1


    log_file_path = os.path.join(log_folder, f"Trade_Log_{date}.csv")
    transaction_file_path = os.path.join(transaction_folder, f"Transaction_F

    log_data = []
    transaction_data = []

    open_long_positions = []
    open_short_positions = []
    commission_rate = 0.00007
    initial_capital = 1_000_000
    capital = initial_capital
    cash = initial_capital
    peak_capital = initial_capital
    total_fees = 0

    for row in df.itertuples(index=False):
        signal, timestamp = row.Signal, row.Timestamp
```

```
        bid_prices, bid_volumes = [row.Bid1, row.Bid2, row.Bid3, row.Bid4],
        ask_prices, ask_volumes = [row.Ask1, row.Ask2, row.Ask3, row.Ask4],

        if signal == -1:
            if open_short_positions:
                for pos in open_short_positions:
                    vwap_price, used_levels = get_vwap_and_levels(pos["Size"
                    profit = (pos["EntryPrice"] - vwap_price) * pos["Size"]
                    fee_close = pos["Size"] * vwap_price * commission_rate
                    net_profit = profit - (pos["Fee"] + fee_close)
                    capital += profit - fee_close
                    cash -= pos["Size"] * vwap_price + fee_close
                    total_fees += fee_close
                    peak_capital = max(peak_capital, capital)
                    drawdown = peak_capital - capital

                    log_data.append([timestamp, "CLOSE_SHORT", pos["Size"],
                    transaction_data.append([pos["Timestamp"], timestamp, pc

                open_short_positions.clear()
            else:
                size = 0.3 * row.vBid1
                vwap_price, used_levels = get_vwap_and_levels(size, bid_pric
                fee = size * vwap_price * commission_rate

                if cash >= size * vwap_price + fee:
                    cash -= size * vwap_price + fee
                    capital -= fee
                    total_fees += fee
                    peak_capital = max(peak_capital, capital)
                    drawdown = peak_capital - capital

                    open_long_positions.append({"Size": size, "EntryPrice":
                    log_data.append([timestamp, "OPEN_LONG", size, vwap_pric

        elif signal == 1:
            if open_long_positions:
                for pos in open_long_positions:
                    vwap_price, used_levels = get_vwap_and_levels(pos["Size"
                    profit = (vwap_price - pos["EntryPrice"]) * pos["Size"]
                    fee_close = pos["Size"] * vwap_price * commission_rate
                    net_profit = profit - (pos["Fee"] + fee_close)
                    capital += profit - fee_close
                    cash += pos["Size"] * vwap_price - fee_close
                    total_fees += fee_close
                    peak_capital = max(peak_capital, capital)
                    drawdown = peak_capital - capital

                    log_data.append([timestamp, "CLOSE_LONG", pos["Size"], v
                    transaction_data.append([pos["Timestamp"], timestamp, pc

                open_long_positions.clear()
            else:
                size = 0.3 * row.vAsk1
                vwap_price, used_levels = get_vwap_and_levels(size, ask_pric
```

```
                fee = size * vwap_price * commission_rate

                if cash >= size * vwap_price + fee:
                    cash += size * vwap_price - fee  # เปิด Short ต้องได้เงินสดเ
                    capital -= fee
                    total_fees += fee
                    peak_capital = max(peak_capital, capital)
                    drawdown = peak_capital - capital

                    open_short_positions.append({"Size": size, "EntryPrice":
                    log_data.append([timestamp, "OPEN_SHORT", size, vwap_pri

    # ✅ บังคับปิดสถานะทั้งหมดก่อนหมดวัน
    if open_long_positions or open_short_positions:
        final_row = df.iloc[-1]
        bid_prices = [final_row.Bid1, final_row.Bid2, final_row.Bid3]
        bid_volumes = [final_row.vBid1, final_row.vBid2, final_row.vBid3]

        ask_prices = [final_row.Ask1, final_row.Ask2, final_row.Ask3]
        ask_volumes = [final_row.vAsk1, final_row.vAsk2, final_row.vAsk3]

        # ✅ บังคับปิด LONG
        for pos in open_long_positions:
            vwap_price, used_levels = get_vwap_and_levels(pos["Size"], ask_p
            profit = (vwap_price - pos["EntryPrice"]) * pos["Size"]
            fee = pos["Size"] * vwap_price * commission_rate
            net_profit = profit - (pos["Fee"] + fee)

            capital += profit - fee
            cash += pos["Size"] * vwap_price - fee  # เงินสดกลับเข้า Cash
            total_fees += fee
            peak_capital = max(peak_capital, capital)
            drawdown = peak_capital - capital

            log_data.append([timestamp, "FORCED_CLOSE_LONG", pos["Size"], vw
            transaction_data.append([pos["Timestamp"], timestamp, pos["Size"

        # ✅ บังคับปิด SHORT
        for pos in open_short_positions:
            vwap_price, used_levels = get_vwap_and_levels(pos["Size"], bid_p
            profit = (pos["EntryPrice"] - vwap_price) * pos["Size"]
            fee = pos["Size"] * vwap_price * commission_rate
            net_profit = profit - (pos["Fee"] + fee)

            capital += profit - fee
            cash -= pos["Size"] * vwap_price + fee  # หักเงินสดออกตอนปิด Short
            total_fees += fee
            peak_capital = max(peak_capital, capital)
            drawdown = peak_capital - capital

            log_data.append([timestamp, "FORCED_CLOSE_SHORT", pos["Size"], v
            transaction_data.append([pos["Timestamp"], timestamp, pos["Size"

    # ✅ คำนวณ Maximum Drawdown (MDD)
    #df_log = pd.DataFrame(log_data, columns=["Timestamp", "Type", "Size", "
```

```
df_log = pd.DataFrame(log_data, columns=["Timestamp", "Type", "Size", "P
max_drawdown = df_log["Drawdown"].max()
df_transaction = pd.DataFrame(transaction_data, columns=["Timestamp_Oper
df_log.to_csv(log_file_path, index=False)
df_transaction.to_csv(transaction_file_path, index=False)

#Mean_return = df_log["Profit"].mean()
Mean_return = df_transaction["Net_Profit"].mean()

#std_return = df_log["Profit"].std()
std_return = df_transaction["Net_Profit"].std()

#total_profit = df_log[df_log["Profit"] > 0]["Profit"].sum()
total_profit = df_transaction[df_transaction["Net_Profit"] > 0]["Net_Pro

#total_loss = df_log[df_log["Profit"] < 0]["Profit"].sum()
total_loss = df_transaction[df_transaction["Net_Profit"] < 0]["Net_Profi

Sharpe_Ratio = Mean_return/std_return
ProfitFactor = int(total_profit)/(int(total_loss)*(-1))
Profit_count = df_log[df_log["Profit"] > 0]["Profit"].count() #แบบไม่หักค่า
Profit_count = df_transaction[df_transaction["Net_Profit"] > 0]["Net_Pro
Loss_count = df_log[df_log["Profit"] < 0]["Profit"].count() #แบบไม่หักค่าธร
Loss_count = df_transaction[df_transaction["Net_Profit"] < 0]["Net_Profi
Win_rate = (Profit_count)/(Profit_count+Loss_count)

Total_Trades = len(df_log)

#Total_Capital_used = df_transaction["Size"] * df_transaction["Price_Ope
#Total_Capital_used = total_capital_used.sum()

#Slippage Cost
df_transaction["Slippage"] = abs(df_transaction["Price_Open"] - df_trans
Total_Slippage_Cost = df_transaction["Slippage"].sum()

#Market Impact Cost
#df_transaction["Market_Impact"] = abs(df_transaction["Price_Open"] - df
#total_market_impact_cost = df_transaction["Market_Impact"].sum()
#AVG
#avg_market_impact = df_transaction["Market_Impact"].mean()


#Commission & Fee Cost
total_fee_cost = df_transaction["Fee_Open"].sum() + df_transaction["Fee_

#Drawdown Duration ไม่ถูก และยังนึกไม่ออกว่าจะใช้อะไรคำนวณ
#df_log["Peak_Capital"] = df_log["Capital"].cummax()
#df_log["Drawdown_Flag"] = df_log["Capital"] < df_log["Peak_Capital"]
#drawdown_duration = df_log["Drawdown_Flag"].sum()



return {"Date": date,
        "Final PnL": round(capital - initial_capital, 2),
```

```
            "ROI_IN (%)": round((capital - initial_capital) * 100 / initial_
            "Final Capital": round(capital, 2),
            "MDD": round(max_drawdown, 2),
            "Sharpe Ratio" : round(Sharpe_Ratio, 2) ,
            "Profit Factor" : round(ProfitFactor, 2),
            "Win-Rate " : round(Win_rate, 2),
            "Total Trades" : round(Total_Trades, 2),
            "Total Slippage Cost " : round(Total_Slippage_Cost, 2),
            #"Total Market Impact Cost " : round(total_market_impact_cost, 2
            #"AVG Market Impact " : round(avg_market_impact, 2),
            "Total Fee Cost " : round(total_fee_cost, 2),
            #"Drawdown Duration " : round(drawdown_duration, 2),
           }

# 📌 ประมวลผล
df_results_blind_test = pd.DataFrame([process_file(path, date) for date, pat
df_results_blind_test.to_csv("results/Performance_Report_Blind_Test.csv", in
#df_results_blind_test
```

In [29]:
```
df_results_blind_test
```

Out[29]:

	Date	Final PnL	ROI_IN (%)	Final Capital	MDD	Sharpe Ratio	Profit Factor	Win-Rate	Total Trades	To Slippa Co
0	2024-05-15	12057.38	1.21	1012057.38	142.92	0.47	9.94	0.58	25284	2106.
1	2024-05-16	18997.94	1.90	1018997.94	164.44	0.33	6.66	0.48	37652	2859

In [31]:
```
create_csv_viewer("results/Trade_Logs_Blind_Test")
```

```
HBox(children=(Dropdown(description='Select File:', options=('Trade_Log_2024
-05-16.csv', 'Trade_Log_2024-05-15…
Output()
HBox(children=(Button(description='⏮ Previous', disabled=True, style=ButtonS
tyle()), Label(value='Page 1'), Bu…
```

In [33]:
```
create_csv_viewer("results/Transaction_Pairs_Blind_Test")
```

```
HBox(children=(Dropdown(description='Select File:', options=('Transaction_Pa
ir_2024-05-15.csv', 'Transaction_P…
Output()
HBox(children=(Button(description='⏮ Previous', disabled=True, style=ButtonS
tyle()), Label(value='Page 1'), Bu…
```

ผลลัพธ์ยังอยู่ในระดับเดียวกันหรือใกล้เคียงกับชุดข้อมูลทดลองแปลว่าเป็นกลยุทธีที่เหมาะสมกับพฤติกรรมของหุ้นตัวนี้ในช่วงระยะเวลาหนึ่ง

In [ ]:
```

```