แนวทางการวิเคราะห์ เนื่องจากข้อจำกัดด้านปริมาณข้อมูลที่มีเพียง 10 วันสำหรับการทดลองและตรวจสอบ การวิเคราะห์จะเป็นการสิเคราะห์พฤติกรรมหุ้นและตลาดโดนรวมมากกว่าการวิเคราะห์ลักษณะการเคลื่อนไหวเฉพาะเจาะจงเป็นรายวัน เพราะต้องการให้กลยุทธ์นี้ใช้กับหุ้นตัวเดิมได้ในทุกวันมากกว่าการหาสถานการณ์เฉพาะทั่วๆไปเพื่อจะได้เข้าใจพฤติกรรมของหุ้น ตลาด หรือคนในตลาด

In [65]:
```python
import pandas as pd
import os

# 📌 โหลดข้อมูลจากหลายไฟล์
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",
}

df_list = {}
for date, file in file_paths.items():
    df = pd.read_csv(file, low_memory=False)
    df["timestamp"] = pd.to_datetime(df["Date"] + " " + df["Time_"])

    # 📌 เติมค่าหายไปด้วยค่าก่อนหน้า (Backward Fill)
    df.fillna(method="bfill", inplace=True)

    df_list[date] = df
```

```
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/1275224698.p
y:22: FutureWarning: DataFrame.fillna with 'method' is deprecated and will r
aise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="bfill", inplace=True)
```

In [67]:

```
import pandas as pd

# 🟢 โหลดข้อมูลและเรียงลำดับเวลา
file_path = "S50M24_20240502_20240516.csv"  # เปลี่ยนเป็นพาธที่ถูกต้อง
df = pd.read_csv(file_path)

# 🟢 กำหนดช่วง Training & Blind Test
train_start = "2024-05-02"
train_end = "2024-05-14"  # Training: 2-14 พ.ค.
blind_start = "2024-05-15"
blind_end = "2024-05-16"   # Blind Test: 15-16 พ.ค.

# 🟢 แบ่งข้อมูลโดยใช้ Masking (แทน .loc[] เพื่อไม่ต้องใช้ index)
df_train = df[(df['Date'] >= train_start) & (df['Date'] <= train_end)]
df_blind = df[(df['Date'] >= blind_start) & (df['Date'] <= blind_end)]
df = df_train

# รวม Date และ Time_ เพื่อสร้าง Timestamp ที่ถูกต้อง
df['timestamp'] = pd.to_datetime(df['Date'] + ' ' + df['Time_'])

# ตั้งค่า timestamp เป็น Index
df.set_index('timestamp', inplace=True)
```

```
# ลบคอลัมน์ Date และ Time_ ที่ไม่จำเป็นแล้ว
df.drop(columns=['Date', 'Time_'], inplace=True)
#df
```

Step 1: Data Inspection & Cleaning

In [70]:
```
import mplfinance as mpf

# ✅ รวม Date และ Time_ เป็น timestamp

# ✅ ตรวจสอบข้อมูลก่อน Resample
print(df[['BidTrade', 'AskTrade', 'BidVolume', 'AskVolume']].head())

# ✅ ใช้ Flat Aggregation เพื่อหลีกเลี่ยง Nested Renamer
df_resampled = df.resample('15T').agg({
    'BidTrade': ['first', 'last', 'min', 'max'],
    'AskTrade': ['first', 'last', 'min', 'max'],
    'BidVolume': 'sum',
    'AskVolume': 'sum'
}).dropna()

# ✅ เปลี่ยนชื่อคอลัมน์ให้เป็น Single Index
df_resampled.columns = ['Open_Bid', 'Close_Bid', 'Low_Bid', 'High_Bid',
                        'Open_Ask', 'Close_Ask', 'Low_Ask', 'High_Ask',
                        'BidVolume', 'AskVolume']

# ✅ คำนวณ Open, Close, High, Low โดยเฉลี่ยจากฝั่ง Bid และ Ask
df_resampled['Open'] = df_resampled[['Open_Bid', 'Open_Ask']].mean(axis=1)
df_resampled['Close'] = df_resampled[['Close_Bid', 'Close_Ask']].mean(axis=1
df_resampled['High'] = df_resampled[['High_Bid', 'High_Ask']].max(axis=1)
df_resampled['Low'] = df_resampled[['Low_Bid', 'Low_Ask']].min(axis=1)
df_resampled['Volume'] = df_resampled['BidVolume'] + df_resampled['AskVolume

# ✅ เลือกเฉพาะคอลัมน์ที่ต้องใช้
df_resampled = df_resampled[['Open', 'High', 'Low', 'Close', 'Volume']]

# ✅ ตรวจสอบโครงสร้าง DataFrame หลังการ Resample
#print(df_resampled.head())

# ✅ พล็อตกราฟแท่งเทียน
mpf.plot(df_resampled, type='candle', volume=True, title='HFT Executed Trade
```

```
                           BidTrade  AskTrade  BidVolume  AskVolume
timestamp
2024-05-02 09:45:00.001003    836.53       NaN       12.0        NaN
2024-05-02 09:45:00.001003       NaN       NaN        NaN        NaN
2024-05-02 09:45:00.001003       NaN    836.76        NaN       70.0
2024-05-02 09:45:00.034358       NaN       NaN        NaN        NaN
2024-05-02 09:45:00.034402    836.50       NaN       23.0        NaN
```

```
/var/folders/hj/vmbgf9dd4tq0dvgjdngp58kw0000gn/T/ipykernel_8975/2190547681.p
y:9: FutureWarning: 'T' is deprecated and will be removed in a future versio
n, please use 'min' instead.
  df_resampled = df.resample('15T').agg({
```

- มีช่วงที่ราคา ขึ้นแรง และ ลงแรง สลับกัน - ภาพรวมราคามีแนวโน้ม ขาขึ้น (Uptrend) แต่มีการ Pullback เป็นระยะ กลยุทธ์ที่อาจเหมาะสม - Momentum Trading ใช้ข้อมูล Volume Spikes เป็นสัญญาณยืนยัน Trend เข้า Position ตามการเคลื่อนไหวของตลาด - Mean Reversion หากราคาพุ่งขึ้นเร็วเกินไปและตามมาด้วยแรงขายหนัก → อาจเกิด Mean Reversion ดูช่วงที่มี Volume Spikes แล้วราคากลับทิศ เพื่อหาจังหวะเปิด Short หรืออาจจะเกิดจากปัจจัยภายนอก เช่น ข่าวสำคัญ Breakout Trading - เมื่อเห็นช่วงที่มี Volume สูงผิดปกติ → อาจเป็นสัญญาณ Breakout - ใช้ร่วมกับแนวต้าน/แนวรับ หรือ VWAP เพื่อยืนยันจุดเข้าออก

Data Inspection & Cleaning

In [73]:
```python
import pandas as pd
import os

# 📌 ตั้งค่าพื้นฐาน
data_folder = "split_data"
file_list = sorted([f for f in os.listdir(data_folder) if f.endswith(".csv")

# 📌 ตรวจสอบโครงสร้างข้อมูลและ Missing Values
inspection_results = []

for file in file_list:
    file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)

    # ✅ เติมค่า NaN
    df.fillna({"BidTrade": 0, "AskTrade": 0, "BidVolume": 0, "AskVolume": 0}
```

```
    # ✅ ตรวจสอบโครงสร้างข้อมูล
    missing_values = df.isna().sum().sum()
    total_rows = len(df)

    inspection_results.append({
        "File": file,
        "Total Rows": total_rows,
        "Missing Values": missing_values,
        "Missing Percentage": (missing_values / (total_rows * len(df.columns
    })

# 📌 แสดงผลลัพธ์
df_inspection_summary = pd.DataFrame(inspection_results)
import ace_tools_open as ace
ace.display_dataframe_to_user(name="Data Inspection Summary", dataframe=df_i
```

Data Inspection Summary

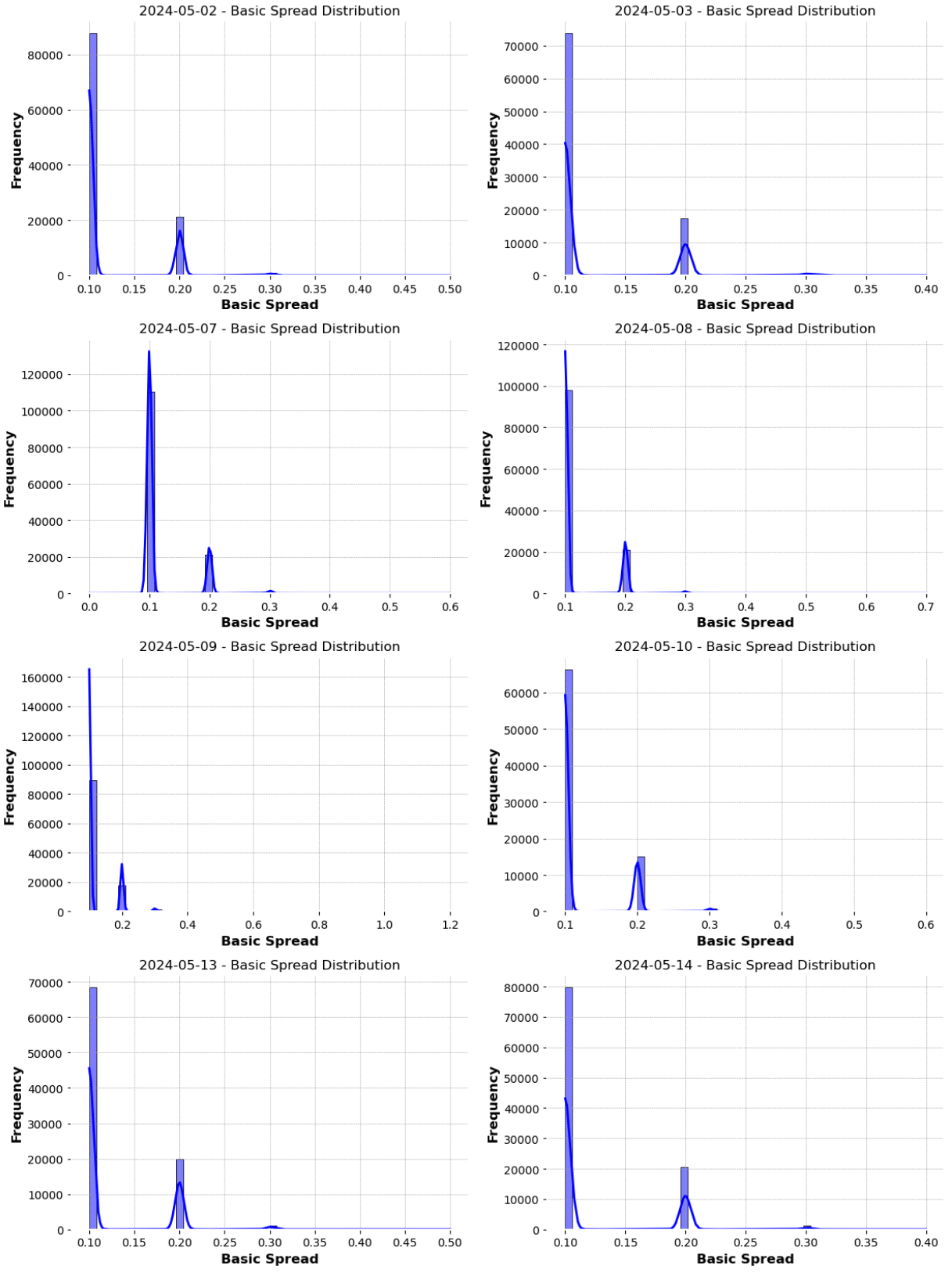
File	Total Rows	Missing Values	Missing Percentage
S50M24_2024-05-02.csv	109724	0	0
S50M24_2024-05-03.csv	92053	0	0
S50M24_2024-05-07.csv	132764	0	0
S50M24_2024-05-08.csv	119857	0	0
S50M24_2024-05-09.csv	108281	0	0
S50M24_2024-05-10.csv	82334	0	0
S50M24_2024-05-13.csv	89686	0	0
S50M24_2024-05-14.csv	101324	0	0

Step 2 : Market Structure Analysis (วิเคราะห์โครงสร้างตลาด)

Basic Spread Distribution

In [92]:
```
import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(4, 2, figsize=(12, 16))

for ax, (date, df) in zip(axes.flat, df_list.items()):
    df["Basic Spread"] = df["Ask1"] - df["Bid1"]
    sns.histplot(df["Basic Spread"].dropna(), bins=50, kde=True, color="blue
    ax.set_title(f"{date} - Basic Spread Distribution")
    ax.set_xlabel("Basic Spread")
```

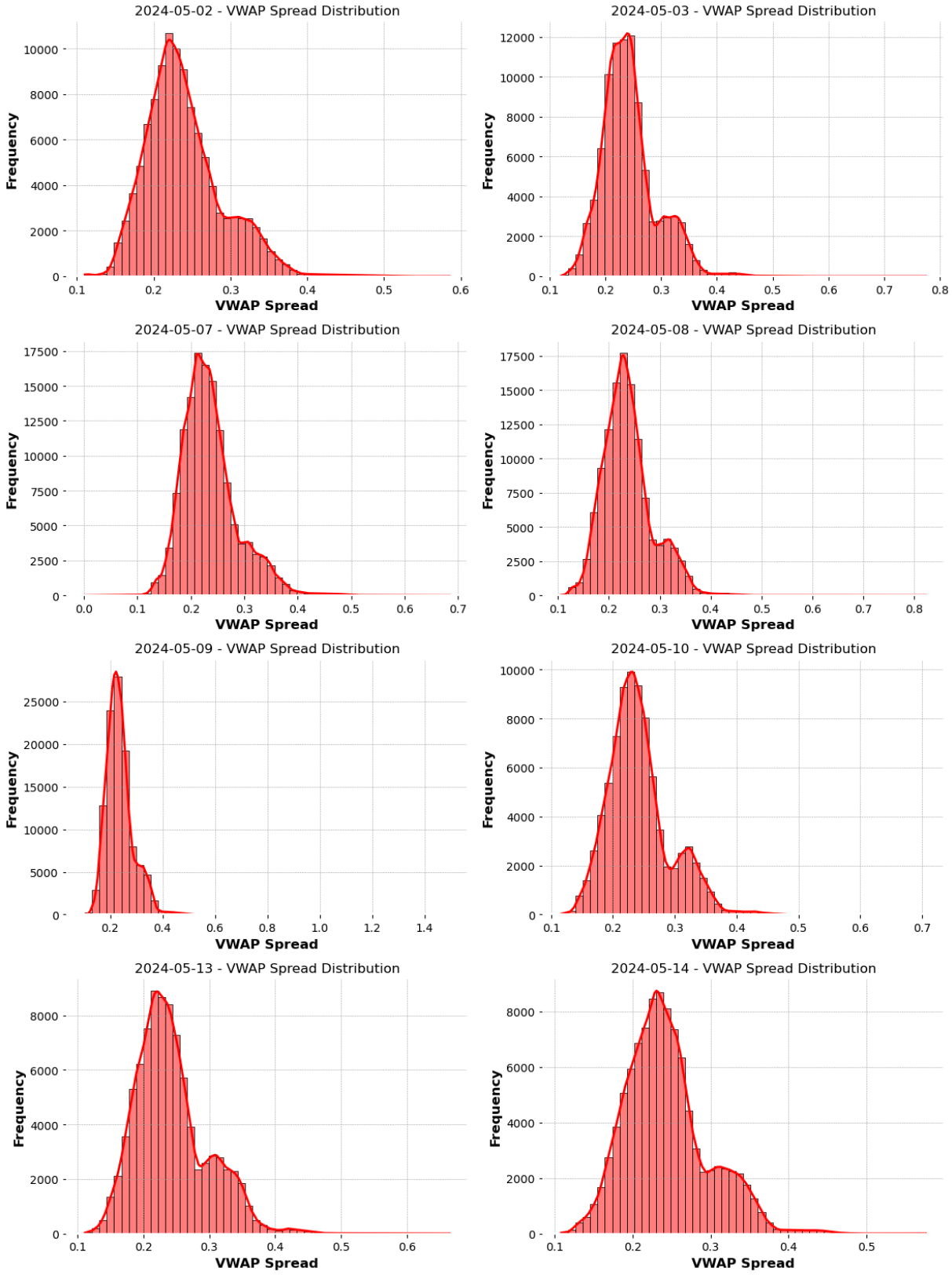
```
    ax.set_ylabel("Frequency")

plt.tight_layout()
plt.show()
```



VWAP Spread Distribution

In [94]:

```
fig, axes = plt.subplots(4, 2, figsize=(12, 16))

for ax, (date, df) in zip(axes.flat, df_list.items()):
    df["VWAP_Bid"] = (df["Bid1"] * df["vBid1"] + df["Bid2"] * df["vBid2"]) /
    df["VWAP_Ask"] = (df["Ask1"] * df["vAsk1"] + df["Ask2"] * df["vAsk2"]) /
    df["VWAP_Spread"] = df["VWAP_Ask"] - df["VWAP_Bid"]

    sns.histplot(df["VWAP_Spread"].dropna(), bins=50, kde=True, color="red",
    ax.set_title(f"{date} - VWAP Spread Distribution")
    ax.set_xlabel("VWAP Spread")
    ax.set_ylabel("Frequency")

plt.tight_layout()
plt.show()
```



VWAP Spread Distribution #ส่วนต่างระหว่างราคาเทรดจริงกับ VWAP ส่วนต่างระหว่างราคาซื้อขายกับ VWAP ที่แท้จริง การกระจายตัวของ Spread มีลักษณะคล้ายกันในแต่ละวัน แปลว่าหุ้นมีพฤติกรรมค่อนข้างคล้ายกัน ส่วนใหญ่การกระจายตัวมีความหนาแน่นอยู่ที่ Spread ต่ำกว่า 0.5 แปลว่ามีสภาพคล่องสูง กลยุทธ์ที่อาจเหมาะสม Mean Reversion Strategy - หาก Spread กว้างขึ้นกว่าปกติ อาจหมายถึงการเคลื่อนตัวของราคาที่เร็วเกินไป - สามารถใช้ VWAP Spread เป็น Threshold ในการเข้าเทรด เช่น หาก Spread สูงเกินค่าเฉลี่ย (0.35 - 0.5) → อาจเป็นสัญญาณ Short #น่าสนใจ หาก Spread ต่ำผิดปกติ → อาจเป็นสัญญาณ Long #น่าสนใจ Liquidity & Market Impact Estimation - Spread ที่กว้างขึ้นอาจบ่งบอกถึง Liquidity ต่ำ หรือ Market Impact สูง - ควรพิจารณา ขนาดของออเดอร์

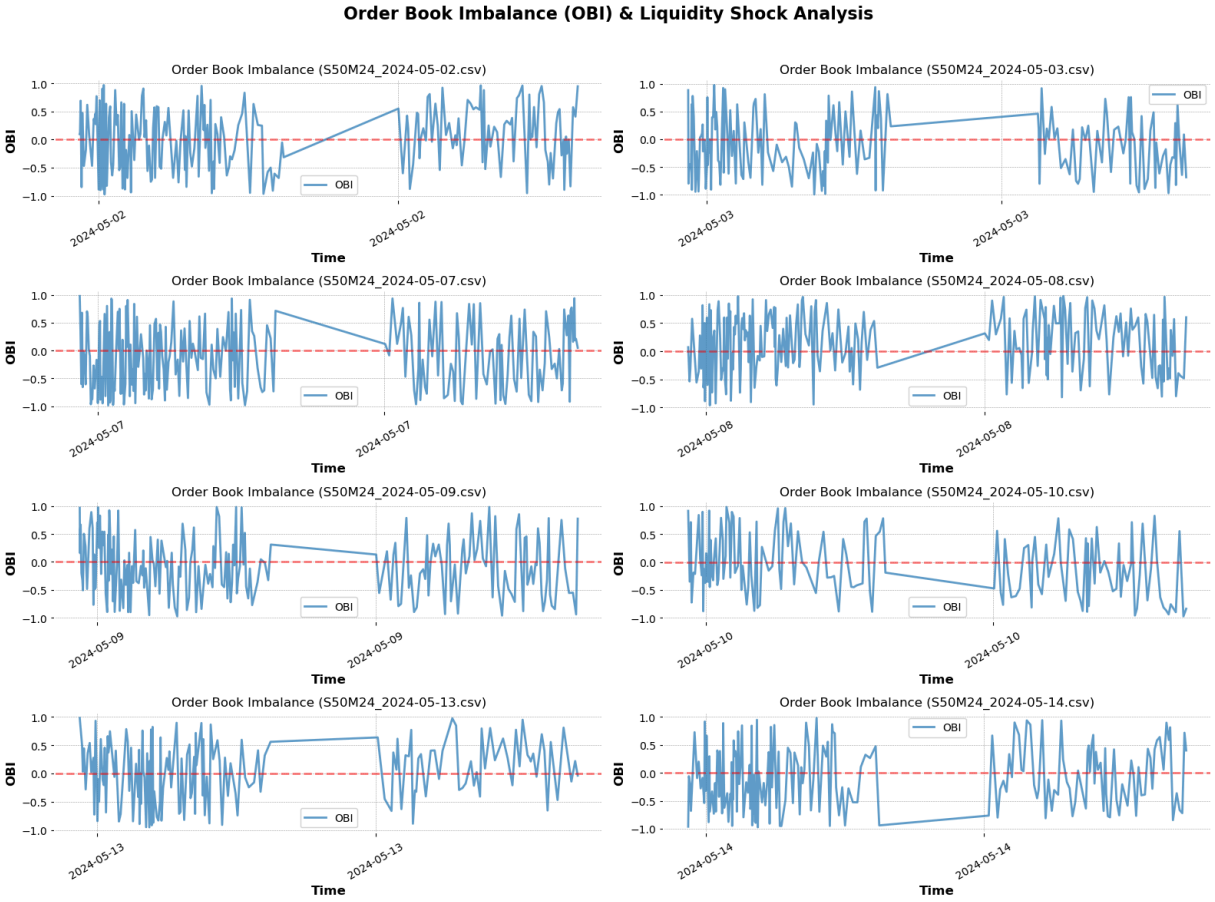
ที่ใช้เพื่อหลีกเลี่ยง Slippage VWAP-Based Execution Strategy - สามารถใช้ VWAP Spread เป็นตัวกำหนด Timing ของคำสั่งซื้อขาย - หาก Spread แคบ → การเข้าเทรดจะมีประสิทธิภาพมากกว่า

Step 3 : Order Flow & Liquidity Analysis (วิเคราะห์กระแสคำสั่งซื้อขายและสภาพคล่อง)

Order Book Imbalance (OBI) & Liquidity Shock

In [96]:
```
# 📌 เก็บผลลัพธ์
obi_stats = []

# 📌 กำหนดขนาดกราฟ (4x2 Multi-Plot)
fig, axes = plt.subplots(4, 2, figsize=(16, 12))
fig.suptitle("Order Book Imbalance (OBI) & Liquidity Shock Analysis", fontsi

for idx, file in enumerate(file_list):
    file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)

    # ✅ คำนวณ Order Book Imbalance (OBI)
    df["OBI"] = (df["vBid1"] - df["vAsk1"]) / (df["vBid1"] + df["vAsk1"])

    # ✅ คำนวณ % การเปลี่ยนแปลงของ Volume (Liquidity Shock)
    df["Liquidity Shock"] = df["vBid1"].pct_change() + df["vAsk1"].pct_chang

    # ✅ ค่าสถิติของ OBI
    obi_data = {
        "File": file,
        "Avg OBI": df["OBI"].mean(),
        "Std OBI": df["OBI"].std(),
        "Avg Liquidity Shock": df["Liquidity Shock"].mean(),
        "Max Liquidity Shock": df["Liquidity Shock"].max()
    }
    obi_stats.append(obi_data)

    # ✅ ลดขนาดข้อมูลโดยสุ่มเลือก 1000 จุด (ถ้ามีมากกว่านั้น) และสร้างสำเนาใหม่
    df_sampled = df.iloc[::500].copy() if len(df) > 5000 else df.copy()

    # ✅ ใช้ .loc[] เพื่อแปลง timestamp เป็น datetime
    df_sampled.loc[:, "timestamp"] = pd.to_datetime(df_sampled["timestamp"])

    # ✅ วาดกราฟ OBI
    ax = axes[idx // 2, idx % 2]
    ax.plot(df_sampled["timestamp"], df_sampled["OBI"], label="OBI", alpha=0
    ax.axhline(0, color="red", linestyle="--", alpha=0.5)
    ax.set_title(f"Order Book Imbalance ({file})")
    ax.set_xlabel("Time")
    ax.set_ylabel("OBI")
    ax.legend()

    # ✅ ปรับแกนเวลาให้โหลดเร็วขึ้น
    ax.set_xticks(ax.get_xticks()[::4])
    ax.tick_params(axis="x", rotation=30)
```

```python
# 📌 ปรับ Layout และแสดงกราฟ
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# 📌 แสดงผลลัพธ์ OBI Summary
df_obi_summary = pd.DataFrame(obi_stats)
import ace_tools_open as ace
ace.display_dataframe_to_user(name="Order Book Imbalance Summary", dataframe
```



Order Book Imbalance Summary

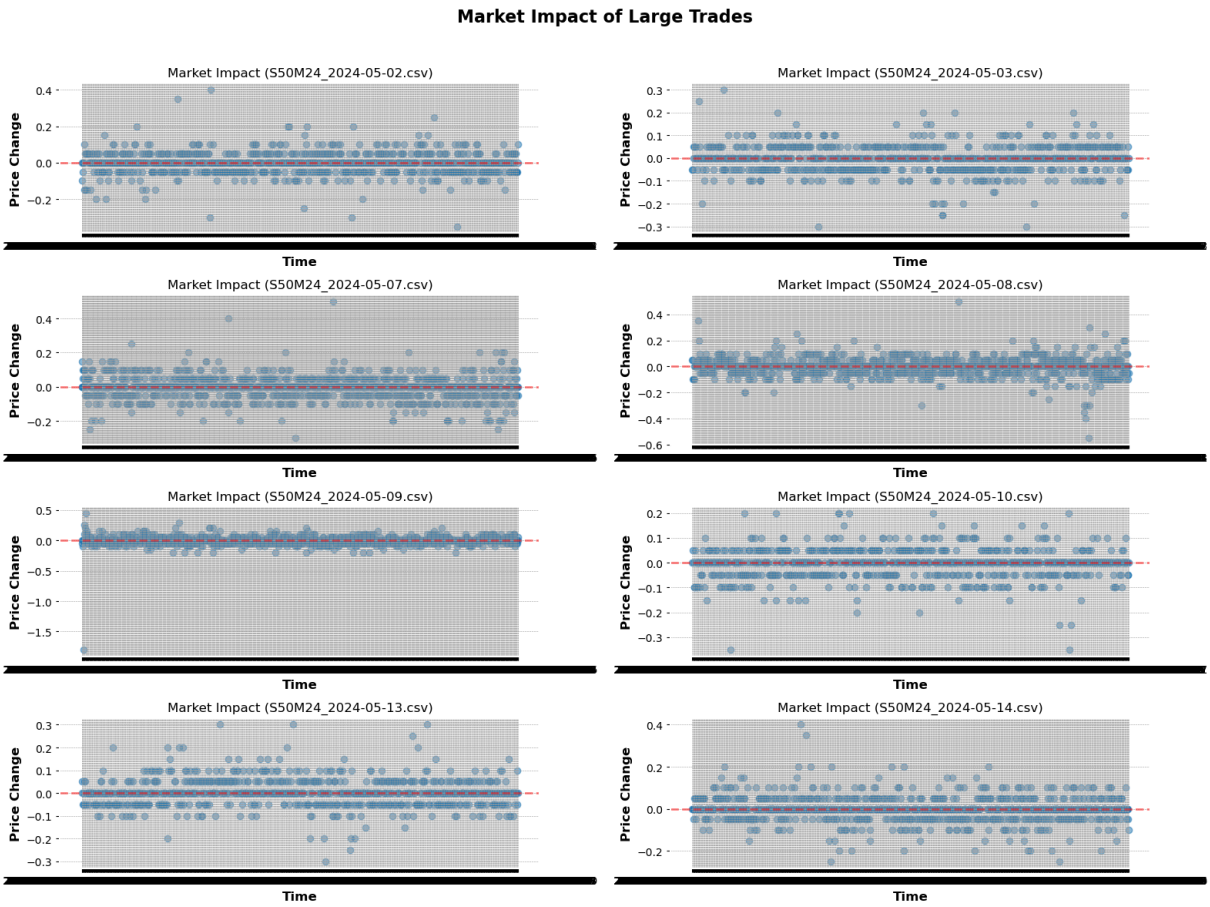
File	Avg OBI	Std OBI	Avg Liquidity Shock	Max Liqu
S50M24_2024-05-02.csv	0.018034	0.541975	1.143072	
S50M24_2024-05-03.csv	-0.092826	0.559873	1.408921	
S50M24_2024-05-07.csv	-0.093288	0.548849	1.14764	
S50M24_2024-05-08.csv	0.120063	0.543857	1.197388	
S50M24_2024-05-09.csv	-0.050494	0.536232	0.945363	
S50M24_2024-05-10.csv	-0.082225	0.535869	0.964849	
S50M24_2024-05-13.csv	0.043367	0.536334	1.01543	
S50M24_2024-05-14.csv	-0.057095	0.548809	1.251141	

ลักษณะของพฤติกรรม #ความไม่สมดุลของคำสั่งซื้อ ค่าเฉลี่ย OBI อยู่ใกล้ศูนย์ → ไม่มี Bias ของฝั่งซื้อหรือขายที่เด่นชัดในระยะยาว ค่า Std OBI ค่อนข้างสูง → บ่งบอกว่าความไม่สมดุลของคำสั่งซื้อขายมีความผันผวนสูง และ Order Flow เปลี่ยนแปลงเร็ว การเคลื่อนไหวของราคาอาจได้รับอิทธิพลจาก Order Imbalance ในช่วงสั้น ๆ สภาพคล่องอาจเปลี่ยนแปลงเร็ว → หมายความว่าแม้ราคาจะดูนิ่งโดยรวม แต่บางช่วงอาจมี Slippage สูงหากมีคำสั่งขนาดใหญ่เข้ามา ตลาดอาจเหมาะกับ HFT หรือ Scalping มากกว่า Swing Trading → เพราะ Order Flow มีการเปลี่ยนแปลงเร็ว การถือ Position นานเกิน ไปอาจทำให้เกิดความเสี่ยงจาก Liquidity Shock กลยุทธ์ที่เหมาะสม Order Flow Scalping : OBI Indicator เพื่อตรวจจับความไม่สมดุลของ Order Book และเข้า Position สั้น ๆ ตาม Flow Market Making with Dynamic Spread ใช้ OBI และ Liquidity Shock เป็นตัวกำหนด Spread ที่จะใช้ในการทำ Market Making หาก OBI ผันผวนสูง → ปรับ Spread ให้กว้างขึ้นเพื่อป้องกัน Slippage หาก OBI ต่ำและ Liquidity สูง → ลด Spread เพื่อเพิ่มโอกาส Fill Orders

Market Impact of Large Trades

In [98]:
```
# 📌 เก็บผลลัพธ์
impact_stats = []

# 📌 กำหนดขนาดกราฟ (4x2 Multi-Plot)
fig, axes = plt.subplots(4, 2, figsize=(16, 12))
fig.suptitle("Market Impact of Large Trades", fontsize=16)

for idx, file in enumerate(file_list):
    file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)

    # ✅ คำนวณ Mid-Price
    df["MidPrice"] = (df["Bid1"] + df["Ask1"]) / 2

    # ✅ หาค่าเปลี่ยนแปลงของ Mid-Price หลังจาก Large Trade
    df["Price Change"] = df["MidPrice"].diff()

    # ✅ คัดเลือก Large Trades (Top 5% ของ Volume)
    large_trades = df[(df["BidVolume"] > df["BidVolume"].quantile(0.95)) |
                      (df["AskVolume"] > df["AskVolume"].quantile(0.95))]

    avg_price_change = large_trades["Price Change"].mean()

    impact_stats.append({
```

```
        "File": file,
        "Avg Price Change After Large Trade": avg_price_change
    })

    # ✅ Plot the impact
    ax = axes[idx // 2, idx % 2]
    ax.scatter(large_trades["timestamp"], large_trades["Price Change"], alph
    ax.axhline(0, color="red", linestyle="--", alpha=0.5)
    ax.set_title(f"Market Impact ({file})")
    ax.set_xlabel("Time")
    ax.set_ylabel("Price Change")

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# 📌 แสดงผลลัพธ์ Market Impact Summary
df_impact_summary = pd.DataFrame(impact_stats)
ace.display_dataframe_to_user(name="Market Impact Analysis", dataframe=df_im
```



Market Impact Analysis

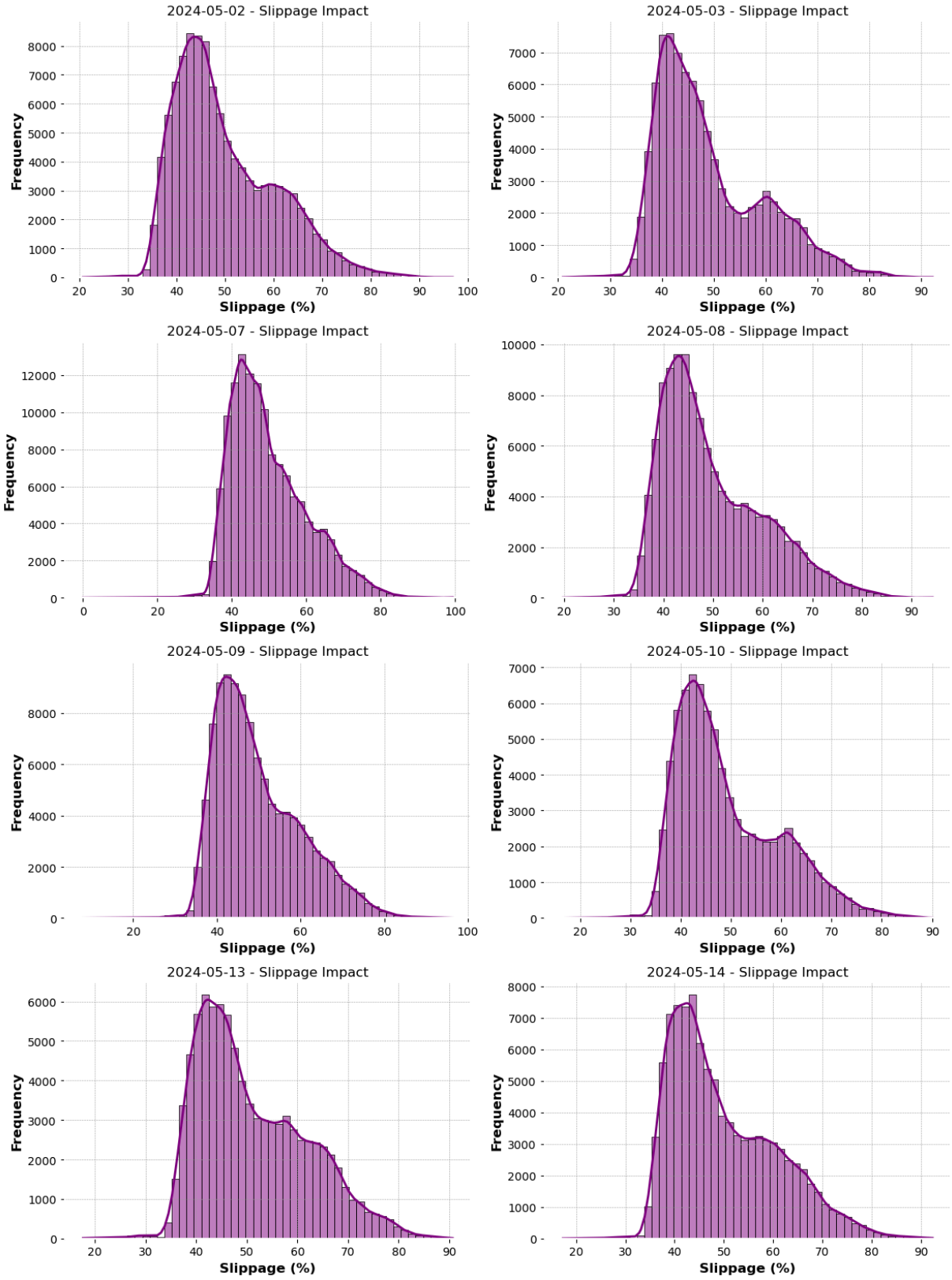
File	Avg Price Change After Large Trade
S50M24_2024-05-02.csv	-0.000481
S50M24_2024-05-03.csv	-0.002768
S50M24_2024-05-07.csv	-0.005321
S50M24_2024-05-08.csv	0.001161
S50M24_2024-05-09.csv	-0.002626
S50M24_2024-05-10.csv	-0.004401
S50M24_2024-05-13.csv	0.003297
S50M24_2024-05-14.csv	-0.000934

ลักษณะของ Market Impact # ผลกระทบจาก Large Order Price Change หลังจาก Large Trades มีค่าเฉลี่ยใกล้ศูนย์ → แสดงว่าหลังจากมีคำสั่งขนาดใหญ่ ราคามักจะกลับคืนสู่ค่าเฉลี่ย มากกว่าจะเกิด Trend Liquidity Shock อาจเป็นตัวกำหนดการเปลี่ยนแปลงของ Market Impact → หากตลาดมีสภาพคล่องต่ำ Large Trade อาจทำให้เกิดการ เคลื่อนไหวรุนแรงชั่วคราว แต่ไม่นานนัก ไม่มี Momentum Effect หลังจาก Large Trade → นั่นหมายความว่าไม่สามารถใช้กลยุทธ์ Trend Following ที่อิงกับ Large Trade ได้ Large Trades ไม่ได้สร้าง Momentum ชัดเจน → ราคามีแนวโน้มกลับเข้าสู่ Mean หลังจากการซื้อขายขนาดใหญ่ การกระจายตัวของ Price Change ค่อนข้างแคบ ในหลายวัน → บ่งบอกว่า Large Trades ไม่ได้ส่งผลกระทบอย่างมีนัยสำคัญในระยะยาว พฤติกรรมของ Order Flow หลังจาก Large Trades บางวันเห็นได้ชัดว่าราคาเปลี่ยนแปลงไปแล้วกลับมาที่ Mean → บ่งบอกว่าตลาดมี Mean Reversion Behavior หาก Spread ขยายตัวหลังจาก Large Trades และหดกลับอย่างรวดเร็ว → อาจมีโอกาสสำหรับ Liquidity-Based Market Making กลยุทธ์ที่เหมาะสม Mean Reversion Trade หลัง Large Trade เมื่อมี Large Trade ราคามักจะกลับเข้าสู่ระดับก่อนหน้า → สามารถใช้กลยุทธ์ Fade หรือ Mean Reversion เพื่อทำกำไร ใช้ VWAP หรือ Moving Average เป็นจุดอ้างอิงสำหรับ Mean Reversion Market Making ปรับตัวตาม Market Impact หากพบว่าหลัง Large Trade ราคามีแนวโน้มกลับเข้าสู่ Midpoint → สามารถปรับ Spread ให้อยู่ในตำแหน่งที่ช่วยให้ Fill Orders ได้ดีขึ้น Large Trade Detection เป็น Signal หากพบ Large Trade ที่มี Market Impact ผิดปกติ อาจเป็นสัญญาณของ Liquidity Shock หรือ Algorithmic Execution ของนักลงทุน รายใหญ่ใช้ข้อมูลนี้เป็นตัวกำหนดการปรับ Position หรือปรับ Risk Management ในระยะสั้น

Slippage Impact

In [107…

```
fig, axes = plt.subplots(4, 2, figsize=(12, 16))

for ax, (date, df) in zip(axes.flat, df_list.items()):
    df["Slippage Impact"] = ((df["Ask1"] - df["Bid1"]) / df["VWAP_Spread"])
    sns.histplot(df["Slippage Impact"].dropna(), bins=50, kde=True, color="p
    ax.set_title(f"{date} - Slippage Impact")
    ax.set_xlabel("Slippage (%)")
    ax.set_ylabel("Frequency")

plt.tight_layout()
plt.show()
```



Slippage Impact ส่วนต่างระหว่างราคาที่เราต้องการและราคาจริงที่คำสั่งถูกดำเนินการ Slippage เฉลี่ยอยู่ที่ 40% - 50% ในทุก ๆ วัน มีบางวัน Slippage ขยายตัวสูงถึง 90% (Extreme Cases) การกระจายตัวของ Slippage ค่อนข้าง เบ้ไปทางขวา (Right-Skewed Distribution) หมายความว่ามีบางเหตุการณ์ที่ Slippage สูงกว่าปกติ กลยุทธ์ที่อาจเหมาะสม VWAP-Based Execution (ลด Slippage) ใช้ VWAP หรือ Limit Orders เพื่อลด Slippage

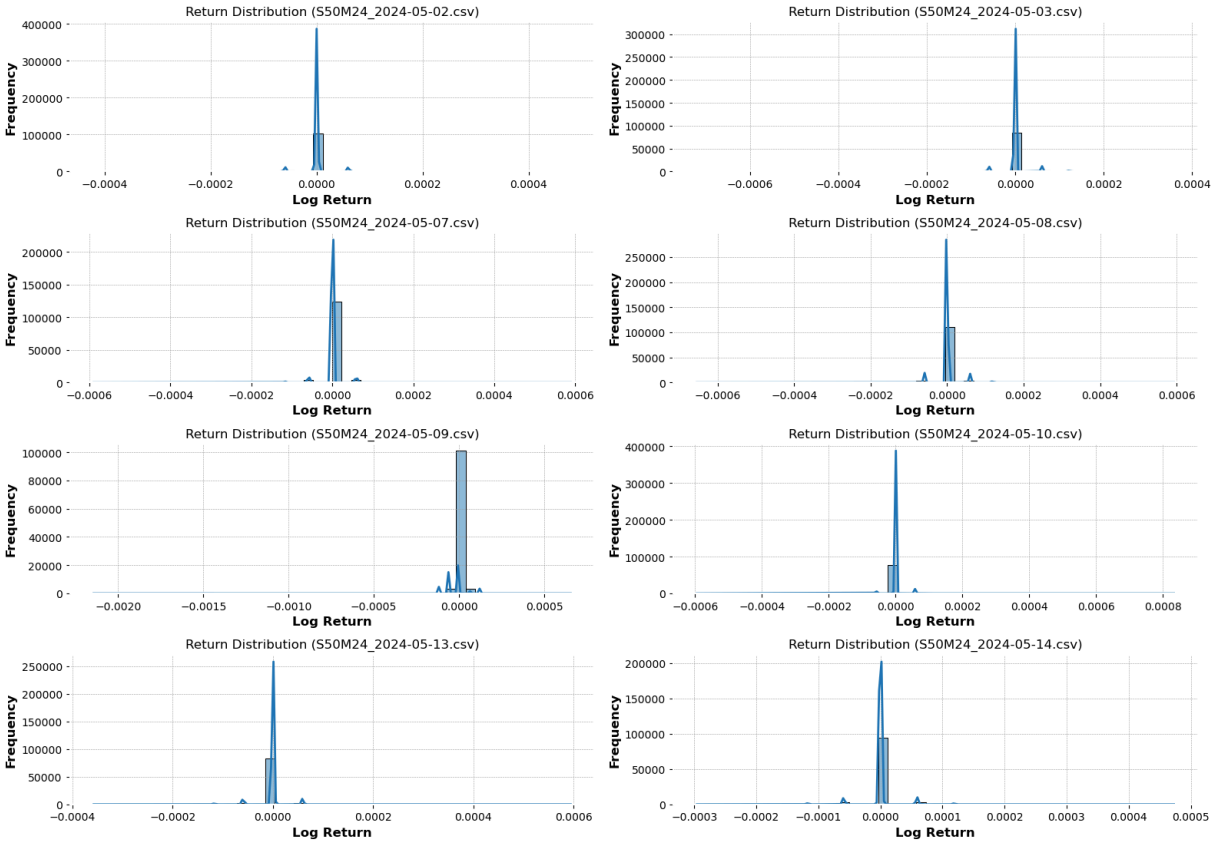
Step 4 : Price Behavior & Statistical Analysis (วิเคราะห์พฤติกรรมราคาและสถิติ)

Return Distribution & Volatility Analysis

In [100…

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 📌 เก็บผลลัพธ์
returns_summary = []
fig, axes = plt.subplots(4, 2, figsize=(16, 12))
fig.suptitle("Return Distribution & Volatility Analysis", fontsize=16)

for idx, file in enumerate(file_list):
    file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)

    # ✅ คำนวณ Log Returns
    df["MidPrice"] = (df["Bid1"] + df["Ask1"]) / 2
    df["LogReturn"] = np.log(df["MidPrice"] / df["MidPrice"].shift(1))

    # ✅ คำนวณสถิติของ Returns
    mean_return = df["LogReturn"].mean()
    std_return = df["LogReturn"].std()

    returns_summary.append({
        "File": file,
        "Mean Return": mean_return,
        "Std Dev Return": std_return
    })

    # ✅ วาด Histogram ของ Returns
    ax = axes[idx // 2, idx % 2]
    sns.histplot(df["LogReturn"].dropna(), bins=50, kde=True, ax=ax)
    ax.set_title(f"Return Distribution ({file})")
    ax.set_xlabel("Log Return")
    ax.set_ylabel("Frequency")

# 📌 แสดงกราฟ
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# 📌 แสดงผลลัพธ์ตาราง Return Summary
df_returns_summary = pd.DataFrame(returns_summary)
ace.display_dataframe_to_user(name="Return & Volatility Summary", dataframe=
```

Return Distribution & Volatility Analysis



Return & Volatility Summary

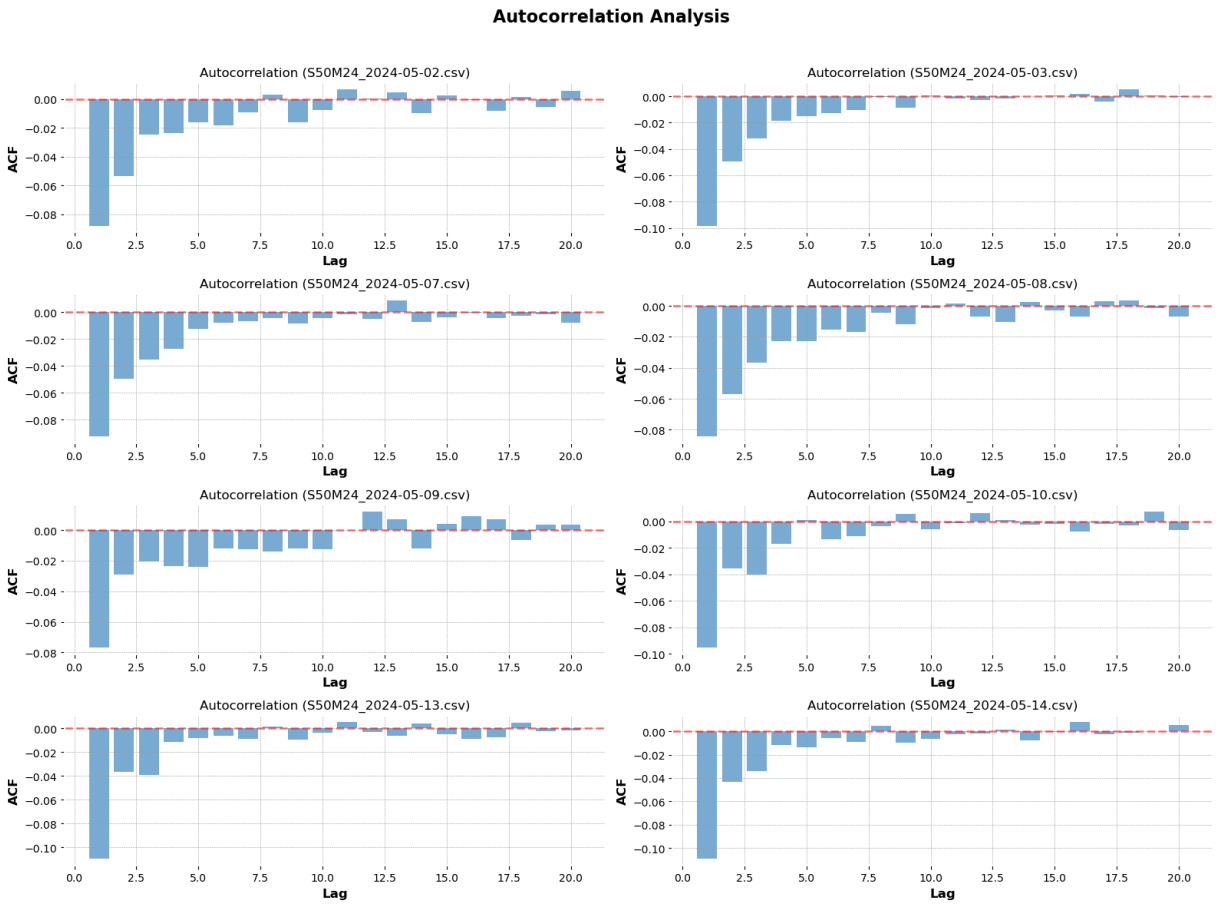
File	Mean Return	Std Dev Return
S50M24_2024-05-02.csv	-5.846966e-8	0.000019
S50M24_2024-05-03.csv	2.859653e-8	0.00002
S50M24_2024-05-07.csv	3.930251e-8	0.000019
S50M24_2024-05-08.csv	-3.313112e-8	0.000021
S50M24_2024-05-09.csv	-2.638821e-8	0.000021
S50M24_2024-05-10.csv	-1.44198e-9	0.00002
S50M24_2024-05-13.csv	1.323851e-8	0.00002
S50M24_2024-05-14.csv	3.739531e-8	0.000019

ลักษณะของหุ้นตัวนี้ #วัดโดย MidPrice Return Distribution: การกระจายของผลตอบแทนมีลักษณะที่แคบมากบ่งบอกว่าหุ้นมีการเคลื่อนไหวของราคาในช่วงแคบ ๆ เป็นหลัก Mean Return: ค่าเฉลี่ยของผลตอบแทนรายวันมีค่าต่ำมากซึ่งหมายความว่าไม่มีแนวโน้มที่ชัดเจนในระยะสั้น Volatility (Std Dev Return): ความผันผวนค่อนข้างคงที่ในระดับต่ำ ซึ่งสอดคล้องกับการเคลื่อนไหวแคบของราคาหุ้น Low Volatility Environment: หุ้นตัวนี้อาจมีแนวโน้มเคลื่อนไหวในช่วงแคบอย่างต่อเนื่อง Mean Reversion Behavior: ค่าเฉลี่ยผลตอบแทนต่ำและการกระจายของ return ค่อนข้างสมมาตร หุ้นตัวนี้อาจมีพฤติกรรมของ Mean Reversion กลยุทธ์ที่เหมาะสม Mean Reversion Strategies : เนื่องจากลักษณะแคบและมีการเคลื่อนไหวกลับไปหาค่าเฉลี่ย กลยุทธ์ที่เหมาะสมคือ Market Making,Pairs Low Volatility Scalping : เนื่องจากความผันผวนต่ำ Scalping อาจทำกำไรได้ดีโดยอาศัยการเข้าออกอย่างรวดเร็วในช่วงราคาที่จำกัด เช่น ใช้ Order Book Analysis เพื่อจับ Liquidity Imbalance และเข้าออกตาม Microstructure Flow

Autocorrelation Analysis (ACF)

In [102…

```python
from statsmodels.tsa.stattools import acf, pacf

# 📌 เก็บผลลัพธ์
autocorr_summary = []
fig, axes = plt.subplots(4, 2, figsize=(16, 12))
fig.suptitle("Autocorrelation Analysis", fontsize=16)

for idx, file in enumerate(file_list):
    file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)

    # ✅ คำนวณ Autocorrelation
    df["MidPrice"] = (df["Bid1"] + df["Ask1"]) / 2
    df["LogReturn"] = np.log(df["MidPrice"] / df["MidPrice"].shift(1))

    acf_values = acf(df["LogReturn"].dropna(), nlags=20)

    # ✅ เก็บค่า Autocorrelation ของ Lag 1-5
    autocorr_summary.append({
        "File": file,
        "ACF Lag 1": acf_values[1],
        "ACF Lag 2": acf_values[2],
        "ACF Lag 3": acf_values[3]
    })

    # ✅ วาดกราฟ Autocorrelation
    ax = axes[idx // 2, idx % 2]
    ax.bar(range(1, len(acf_values)), acf_values[1:], alpha=0.6)
    ax.axhline(0, color="red", linestyle="--", alpha=0.5)
    ax.set_title(f"Autocorrelation ({file})")
    ax.set_xlabel("Lag")
    ax.set_ylabel("ACF")

# 📌 แสดงกราฟ
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# 📌 แสดงผลลัพธ์ Autocorrelation Summary
df_autocorr_summary = pd.DataFrame(autocorr_summary)
ace.display_dataframe_to_user(name="Autocorrelation Summary", dataframe=df_a
```



Autocorrelation Summary

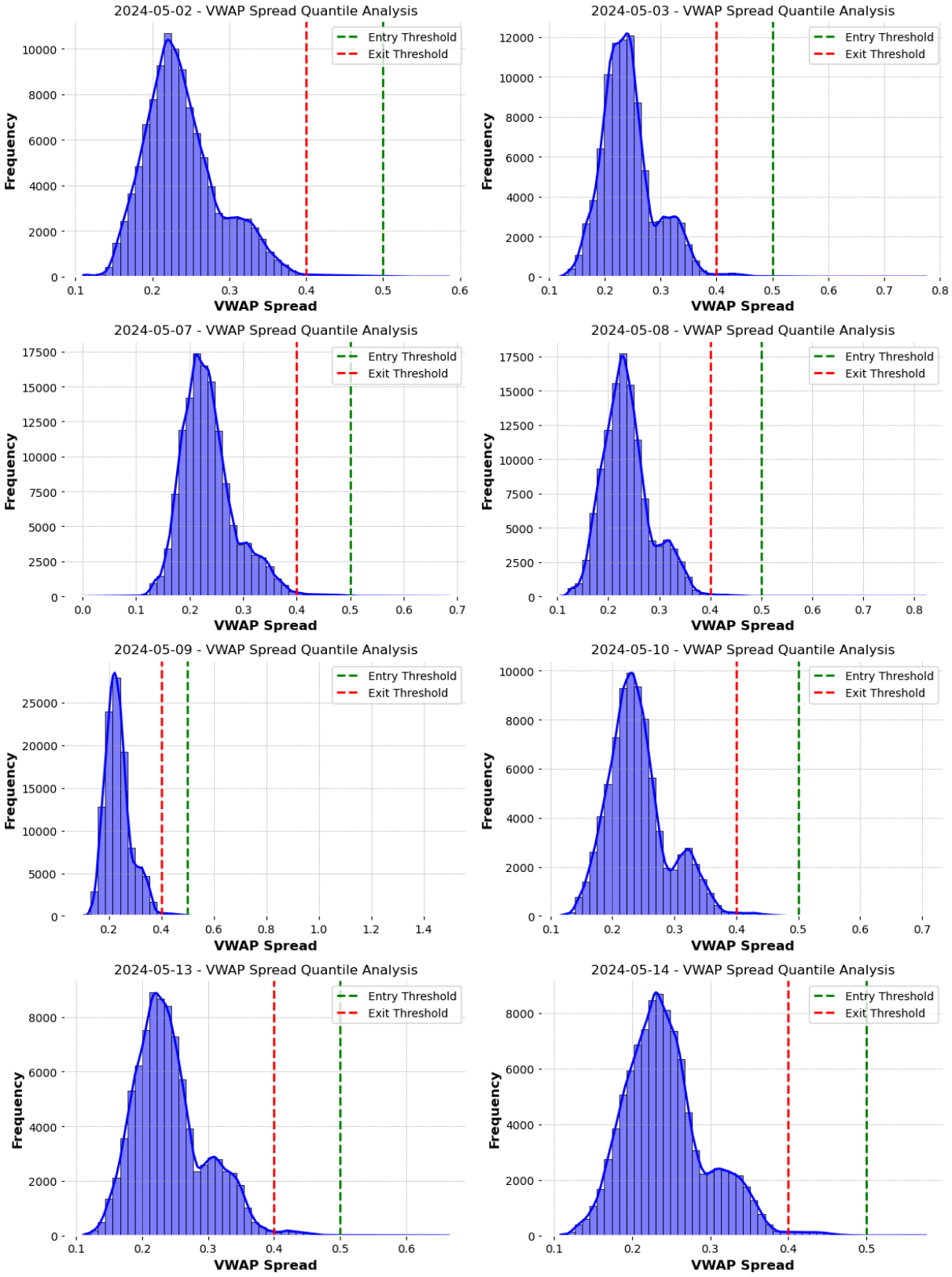
File	ACF Lag 1	ACF Lag 2	ACF Lag 3
S50M24_2024-05-02.csv	-0.087834	-0.053242	-0.024336
S50M24_2024-05-03.csv	-0.098125	-0.049249	-0.032167
S50M24_2024-05-07.csv	-0.09252	-0.049269	-0.034955
S50M24_2024-05-08.csv	-0.084507	-0.057068	-0.036659
S50M24_2024-05-09.csv	-0.076856	-0.029261	-0.020371
S50M24_2024-05-10.csv	-0.095256	-0.035438	-0.040114
S50M24_2024-05-13.csv	-0.109529	-0.036746	-0.039113
S50M24_2024-05-14.csv	-0.108991	-0.042994	-0.034151

Autocorrelation Function (ACF) คืออะไร? วัดว่าค่าของ Series (ราคาหุ้น) มีความสัมพันธ์กับค่าก่อนหน้า (Lagged Values) อย่างไร ค่า ACF เป็นบวก → มี Momentum (แนวโน้มต่อเนื่อง) ค่า ACF เป็นลบ → มี Mean Reversion (ราคากลับเข้าสู่ค่าเฉลี่ย) ค่า ACF ลดลงใกล้ศูนย์ ไม่มีโครงสร้างชัดเจน ค่า ACF เป็นบวกลบเป็นช่วงๆ Cyclical Behaviorลักษณะของพฤติกรรม ค่า ACF เป็นลบในช่วง Lag 1 - Lag 3 → บ่งชี้ว่าราคามีแนวโน้มจะเคลื่อนที่ในลักษณะ Mean Reversion มากกว่าที่จะเป็น Trend Following ACF ลดลงเรื่อย ๆ และไม่เด้งกลับขึ้นมา → ไม่มีโครงสร้างของ Momentum ชัดเจน ACF ค่าต่ำและค่อนข้างคงที่ทุกวัน → พฤติกรรมของราคามีความเสถียรในแง่ของ Mean Reversion พฤติกรรมอาจคล้ายกับ Market Making Environment ที่ราคาถูกกดให้อยู่ในกรอบแคบ กลยุทธ์ที่เหมาะสม Mean Reversion Trading Liquidity-Based Market Making : เนื่องจากพฤติกรรมราคามี Mean Reversion และ ACF เป็นลบ Market Making สามารถทำกำไรได้ Scalping ในช่วงราคาสั้น ๆ : พฤติกรรม Mean Reversion + Low Volatility → Scalping ได้ผลดี

VWAP Spread Quantile Analysis

In [104…

```
fig, axes = plt.subplots(4, 2, figsize=(12, 16))

entry_threshold = 0.5
exit_threshold = 0.4

for ax, (date, df) in zip(axes.flat, df_list.items()):
    sns.histplot(df["VWAP_Spread"].dropna(), bins=50, kde=True, color="blue"
    ax.axvline(entry_threshold, color="green", linestyle="--", label="Entry
    ax.axvline(exit_threshold, color="red", linestyle="--", label="Exit Thre
    ax.legend()
    ax.set_title(f"{date} - VWAP Spread Quantile Analysis")
    ax.set_xlabel("VWAP Spread")
    ax.set_ylabel("Frequency")

plt.tight_layout()
plt.show()
```



VWAP Spread Distribution ส่วนต่างระหว่างราคาที่ Volume ถูกถัวเฉลี่ยและราคาตลาด ณ ขณะนั้น บ่งชี้ว่าราคามีความผันผวนหรือไม่ บางวันกว้างบางวันแคบ เส้น Entry และ Exit Threshold เส้นสีแดง (Entry Threshold) → จุดที่กำหนดให้เข้าเทรด (เช่น Quantile 80%) เส้นสีเขียว (Exit Threshold) → จุดที่กำหนดให้ปิดสถานะ (เช่น Quantile 30%) Thresholds ถูกวางไว้โดยอ้างอิงจากการกระจายตัวของ VWAP Spread กลยุทธ์ที่อาจเหมาะสม - Quantile-Based Entry/Exit Strategy กำหนดจุดเข้าและออกโดยอิงจาก Quantile 80% และ 30% ถ้า Spread สูงกว่าระดับ Entry Threshold → เปิดสถานะ ถ้า Spread ลดลงต่ำกว่าระดับ Exit Threshold → ปิดสถานะ - Mean Reversion Trading (เล่นการกลับสู่ค่าเฉลี่ย) ถ้า Spread กว้างขึ้นผิดปกติ → ตลาดอาจมี Mean Reversion สามารถใช้กลยุทธ์ Fade the

Extremes (เปิดสถานะตรงข้ามเมื่อ Spread ขยายตัว) - Adaptive Thresholding (ปรับ Threshold ตามตลาด) Threshold ควรปรับตามสภาพตลาด (Dynamic Quantile) ใช้วิธีปรับจุด Entry/Exit ตามค่าเบี่ยงเบนมาตรฐานของ Spread

Step 5 : Choosing the Strategy

จากการวิเคราะห์ในหลายมิติ ทั้งด้านโครงสร้างตลาด (Market Microstructure), พฤติกรรมของราคา (Price Behavior), สภาพคล่อง (Liquidity), และกระแสคำสั่ง (Order Flow) เราสามารถสรุปสาระสำคัญที่สะท้อนลักษณะเฉพาะของหุ้นตัวนี้ได้ ดังนี้: 1. ตลาดมีลักษณะ “Mean-Reverting” ชัดเจน สัญญาณจาก Autocorrelation, การกลับสู่ Mean หลังจาก Large Trade และ Return Distribution ที่แคบ ล้วนบ่งชี้ว่าราคามีแนวโน้มจะเคลื่อนไหวในกรอบ และมักย้อนกลับสู่ค่าเฉลี่ยในระยะสั้น สิ่งนี้ส่งผลให้กลยุทธ์ที่อาศัย Momentum หรือ Trend Following มีแนวโน้มที่จะ ไม่ได้ผลหรืออาจนำไปสู่ False Signal ได้ง่าย 2. โครงสร้างของ Order Flow มีความผันผวนสูง แม้จะสมดุลในระยะยาว ค่าเฉลี่ยของ Order Book Imbalance อาจดูใกล้ศูนย์ แต่ Standard Deviation ที่สูงแสดงถึง ความไม่แน่นอน และการสลับฝั่งของแรงซื้อแรงขายอย่างรวดเร็ว ความไม่แน่นอนนี้ส่งผลให้ต้อง บริหารความเสี่ยงแบบ Dynamic โดยเฉพาะอย่างยิ่งในกลยุทธ์ที่เกี่ยวข้องกับ Volume ขนาดใหญ่ 3. Slippage สูงและมีลักษณะเบ้ขวา (Skewed) ชี้ให้เห็นถึงความเสี่ยงเชิงปฏิบัติในการวางคำสั่งซื้อขาย โดยเฉพาะในช่วงเวลาที่สภาพคล่องต่ำหรือคำสั่งขนาดใหญ่ การบริหารขนาดคำสั่ง (Order Sizing) และการเลือกช่วงเวลาในการเข้าตลาด (Timing) จึงเป็น ปัจจัยสำคัญที่ต้องควบคุมอย่างเข้มงวดจาก Insight ข้างต้น เราสามารถระบุแนวทางกลยุทธ์ที่สอดคล้องกับลักษณะตลาด ได้แก่: Mean Reversion Strategy: ใช้ Quantile ของ VWAP Spread เป็นจุดเข้าซื้อ/ขาย เพื่อทำกำไรจากการกลับเข้าสู่ค่าเฉลี่ย Market Making: ใช้ประโยชน์จากความสม่ำเสมอของ Spread และ Return Distribution ที่คงที่ ในการวางคำสั่งฝั่งซื้อขายพร้อมกันใน Bid/Ask Scalping หรือ HFT Style: อาศัยจังหวะการสลับของ Order Imbalance และการเปลี่ยนทิศของ Microstructure เพื่อทำกำไรในระยะเวลาอันสั้นสรุปแนวทางการเทรดที่เหมาะสมกับหุ้นตัวนี้ ตกลงที่จะลองใช้ Mean Reversion Strategies,Market Making,หรือ Mean Reversion Scalping ในการทดลองเป็นฐาน และเพิ่มกลยุทธ์ signal หรืออื่นๆ ประกอบกันในภายหลังจากการทดลองทำ Backtest กับข้อมูลที่แบ่งไว้สำหรับ ทดสอบ(8 ใน 10 วัน และอีก 2 วันแบ่งไปทำ Blind test)

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

EDA เพิ่มเติม

AVG Slippage

In [129…

```
import pandas as pd
import numpy as np

# 📌 รายชื่อไฟล์ข้อมูล (สามารถเพิ่มให้ครบ 8 ไฟล์ได้)
file_paths = [
    "split_data/S50M24_2024-05-02.csv",
    "split_data/S50M24_2024-05-03.csv",
    "split_data/S50M24_2024-05-07.csv",
    "split_data/S50M24_2024-05-08.csv",
    "split_data/S50M24_2024-05-09.csv",
    "split_data/S50M24_2024-05-10.csv",
    "split_data/S50M24_2024-05-13.csv",
```

```
    "split_data/S50M24_2024-05-14.csv",
]

# 📌 ฟังก์ชันคำนวณ Slippage
def calculate_slippage(df):
    df["Slippage"] = df["Ask1"] - df["Bid1"]
    return df["Slippage"].mean()  # ค่าเฉลี่ยของ Slippage ที่ใช้ใน Backtest

# 📌 เก็บผลลัพธ์ Slippage
results_slippage = []

# 📌 วนลูปประมวลผลไฟล์ทั้งหมด
for file_path in file_paths:
    df = pd.read_csv(file_path)

    # ✅ ตรวจสอบว่ามีคอลัมน์ที่ต้องใช้หรือไม่
    required_columns = {"Bid1", "Ask1"}
    if not required_columns.issubset(df.columns):
        print(f"❌ Missing required columns in {file_path}")
        continue

    # ✅ คำนวณค่าเฉลี่ยของ Slippage
    avg_slippage = calculate_slippage(df)

    # ✅ เก็บค่าผลลัพธ์สำหรับ Backtest
    results_slippage.append({
        "File": file_path.split("/")[-1],
        "Avg Slippage (%)": round(avg_slippage, 5)
    })

# 📌 แสดงผลลัพธ์ใน DataFrame
df_results_slippage = pd.DataFrame(results_slippage)
tools.display_dataframe_to_user(name="Backtest Slippage Summary", dataframe=
```

Backtest Slippage Summary

File	Avg Slippage (%)
S50M24_2024-05-02.csv	0.12088
S50M24_2024-05-03.csv	0.12084
S50M24_2024-05-07.csv	0.1182
S50M24_2024-05-08.csv	0.11909
S50M24_2024-05-09.csv	0.11934
S50M24_2024-05-10.csv	0.12046
S50M24_2024-05-13.csv	0.12516
S50M24_2024-05-14.csv	0.12245

ค่าเฉลี่ย Slippage ค่อนข้างคงที่และไม่สูงมาก การจำลอง จำลองโดยการ ซื้อที่โวลุ่มต่ำ และมีการใช้ Dept Orderbook เพื่อชดเชยการจำลอง Slippage แทนแล้ว

AVG Market Impact

In [135…

```python
# 📌 ฟังก์ชันคำนวณ Market Impact
def calculate_market_impact(df):
    df["MidPrice"] = (df["Bid1"] + df["Ask1"]) / 2
    df["MarketImpact"] = df["MidPrice"].diff()
    return df["MarketImpact"].mean() * 100  # คิดเป็นเปอร์เซ็นต์

# 📌 เก็บผลลัพธ์ Market Impact
results_market_impact = []

# 📌 วนลูปประมวลผลไฟล์ทั้งหมด
for file_path in file_paths:
    df = pd.read_csv(file_path)

    # ✅ ตรวจสอบว่ามีคอลัมน์ที่ต้องใช้หรือไม่
    required_columns = {"Bid1", "Ask1"}
    if not required_columns.issubset(df.columns):
        print(f"❌ Missing required columns in {file_path}")
        continue

    # ✅ คำนวณค่าเฉลี่ยของ Market Impact
    avg_market_impact = calculate_market_impact(df)

    # ✅ เก็บค่าผลลัพธ์สำหรับ Backtest
    results_market_impact.append({
        "File": file_path.split("/")[-1],
        "Avg Market Impact (%)": round(avg_market_impact, 5)
    })

# 📌 แสดงผลลัพธ์ใน DataFrame
df_results_market_impact = pd.DataFrame(results_market_impact)
tools.display_dataframe_to_user(name="Backtest Market Impact Summary", dataf
```

Backtest Market Impact Summary

File	Avg Market Impact (%)
S50M24_2024-05-02.csv	-0.00488
S50M24_2024-05-03.csv	0.00239
S50M24_2024-05-07.csv	0.00331
S50M24_2024-05-08.csv	-0.0028
S50M24_2024-05-09.csv	-0.00222
S50M24_2024-05-10.csv	-0.00012
S50M24_2024-05-13.csv	0.00112
S50M24_2024-05-14.csv	0.00316

ค่าเฉลี่ย Market Impact มีค่าที่ต่ำมาก นอกจากนี้จากพฤติกรรมการวิเคราะห์ EDA ที่ผ่านมา Large Order ไม่ส่งผลกระทบต่อตลาดมากนัก และการซื้อในระบบใช้โวลุ่มค่อนข้างต่ำ

Fill Rate

In [38]:
```python
import ace_tools_open as tools
# 📌 ฟังก์ชันคำนวณ Fill Rate สำหรับแต่ละไฟล์
def calculate_fill_rate(order_book_path, trade_log_path):
    try:
        # 📌 โหลดไฟล์
        order_book_df = pd.read_csv(order_book_path)
        trade_log_df = pd.read_csv(trade_log_path)

        # 📌 ปรับชื่อคอลัมน์ให้เป็นตัวพิมพ์เล็กเพื่อป้องกัน Case-Sensitivity
        order_book_df.columns = order_book_df.columns.str.strip().str.lower(
        trade_log_df.columns = trade_log_df.columns.str.strip().str.lower()

        # ✅ ตรวจสอบว่ามีคอลัมน์ที่ต้องใช้ครบหรือไม่
        required_columns_trade_log = {"timestamp", "type"}
        required_columns_order_book = {"timestamp", "bidtrade", "asktrade"}

        if not required_columns_trade_log.issubset(trade_log_df.columns) or
            print(f"❌ Missing required columns in {order_book_path} or {tra
            return None

        # ✅ รวม Order Book กับ Trade Log ตาม Timestamp (ไม่ใช้ Price)
        merged_df = trade_log_df.merge(order_book_df[["timestamp", "bidtrade

        # ✅ คำนวณ Fill Rate
        total_orders = len(trade_log_df)

        # ตรวจสอบ Fill Rate ที่ Bid1 และ Ask1 (ใช้แค่ Timestamp + BidTrade/AskTr
        #bid1_trades = merged_df[merged_df["bidtrade"] > 0]
        #ask1_trades = merged_df[merged_df["asktrade"] > 0]

        #bid1_fill_rate = len(bid1_trades) / total_orders * 100
        #ask1_fill_rate = len(ask1_trades) / total_orders * 100

        # ตรวจสอบ Fill Rate ตามประเภทคำสั่ง (ใช้แค่ Timestamp + BidTrade/AskTrad
        open_long_fill = len(merged_df[(merged_df["type"] == "OPEN_LONG") &
        close_short_fill = len(merged_df[(merged_df["type"] == "CLOSE_SHORT"
        open_short_fill = len(merged_df[(merged_df["type"] == "OPEN_SHORT")
        close_long_fill = len(merged_df[(merged_df["type"] == "CLOSE_LONG")

        return {
            "File": trade_log_path.split("/")[-1],
            #"Bid1 Fill Rate (%)": round(bid1_fill_rate, 2),
            #"Ask1 Fill Rate (%)": round(ask1_fill_rate, 2),
            "OPEN_LONG Fill Rate (%)": round(open_long_fill, 2),
            "CLOSE_SHORT Fill Rate (%)": round(close_short_fill, 2),
            "OPEN_SHORT Fill Rate (%)": round(open_short_fill, 2),
            "CLOSE_LONG Fill Rate (%)": round(close_long_fill, 2),
            "Total Orders Checked": total_orders
        }
```

```
    except Exception as e:
        print(f"❌ Error processing {trade_log_path}: {e}")
        return None

# 📌 รายการไฟล์ที่ต้องประมวลผล
file_pairs = [
    ("split_data/S50M24_2024-05-02.csv", "Trade_Log/Trade_Log_2024-05-02.csv
    ("split_data/S50M24_2024-05-03.csv", "Trade_Log/Trade_Log_2024-05-03.csv
    ("split_data/S50M24_2024-05-07.csv", "Trade_Log/Trade_Log_2024-05-07.csv
    ("split_data/S50M24_2024-05-08.csv", "Trade_Log/Trade_Log_2024-05-08.csv
    ("split_data/S50M24_2024-05-09.csv", "Trade_Log/Trade_Log_2024-05-09.csv
    ("split_data/S50M24_2024-05-10.csv", "Trade_Log/Trade_Log_2024-05-10.csv
    ("split_data/S50M24_2024-05-13.csv", "Trade_Log/Trade_Log_2024-05-13.csv
    ("split_data/S50M24_2024-05-14.csv", "Trade_Log/Trade_Log_2024-05-14.csv
]

# ✅ คำนวณ Fill Rate สำหรับทุกไฟล์
fill_rate_results = [calculate_fill_rate(order_book, trade_log) for order_bo

# ✅ สร้าง DataFrame แสดงผลลัพธ์
df_results_fill_rate = pd.DataFrame(fill_rate_results)

# ✅ แสดงผล
tools.display_dataframe_to_user(name="Backtest Fill Rate Summary", dataframe
```

Backtest Fill Rate Summary

File	OPEN_LONG Fill Rate (%)	CLOSE_SHORT Fill Rate (%)
Trade_Log_2024-05-02.csv	99.84	99
Trade_Log_2024-05-03.csv	99.9	99.
Trade_Log_2024-05-07.csv	99.77	99.
Trade_Log_2024-05-08.csv	99.81	98
Trade_Log_2024-05-09.csv	99.85	1
Trade_Log_2024-05-10.csv	99.86	99.
Trade_Log_2024-05-13.csv	99.88	99.
Trade_Log_2024-05-14.csv	99.88	9