Quantitative Trading Strategy Project

Introduction

This project focuses on designing a momentum-based short-selling strategy around **index rebalancing events**, particularly targeting **one-off events**. The strategy is built using quantitative analysis, technical indicators, and strict liquidity constraints to capture price distortions caused by index additions. Risk management is integrated through a **SPY hedge** to mitigate market-wide movements. The main technical indicators used in this strategy include **RSI**, **MACD**, **Bollinger Bands**, and **ATR**, which are well-suited for identifying overbought conditions and momentum shifts based on my personal equity trading experience.

Data Extraction, Cleaning, and Liquidity Filtering

Initial Data Inspection and Cleaning

I began by loading the dataset and addressing missing values using `pandas` and defining appropriate `na_values`. Initial attempts to fill missing fields (like `$MM to Trade` and `Shares to Trade`) were later abandoned as most stocks with missing data were delisted or invalid. For conciseness, I removed these tickers and cleaned the data thoroughly.

Special characters in the **Ticker** column (e.g., `*`, `-W`, `-V`) were stripped, and spaces were removed. The validity of each ticker was confirmed by fetching historical data from Yahoo Finance (`yfinance`), ensuring that only tickers with valid data were retained.

Event Classification: One-off vs. Scheduled Events

The dataset was classified into:

- **One-off Events**: Events occurring outside regularly scheduled rebalancing months.
- **Scheduled Events**: Events occurring in March, June, September, and December.

This allowed for filtering based on event type, with a focus on **one-off events** for strategy development.

Liquidity Constraints

To minimize market impact, I imposed a **liquidity constraint** based on the 20-day average trading volume, ensuring that no position size exceeded 1% of a stock's average volume, with a **minimum threshold of 50,000 shares**. This constraint was designed to ensure smooth trade execution without moving the market.

1. **Mathematical Justification**: In managing a 5,000,000 portfolio, I allocate 76% of the total capital to my aggressive short strategy, splitting this evenly across four stocks, meaning each stock receives 19% of the portfolio. The remaining 24% is allocated to a SPY hedge.To avoid creating market distortions while deploying substantial capital, I ensure that my strategy does not exceed 1% of the 20-day average trading volume for each stock. Given my capital and liquidity constraints, this balance is critical in ensuring smooth trade execution without becoming a "whale" in the market.

For context, the maximum position I could theoretically take would be 50,000 shares of a stock priced at 100 dollars. However, given that only 76% of my capital is allocated to the aggressive strategy, my total allocation per stock is limited to 19% of 5,000,000 or 950,000 per stock.

This calculation allows me to trade substantial volumes without impacting the market. For instance:

If a stock's price is 100 dollars, I can purchase 9,500 shares (950,000 / 100), which is well below the typical 50,000-share daily average.

Even if a stock’s price were lower (e.g., 50 per share), my 950,000 allocation would enable me to purchase 19,000 shares, which would still be only a fraction of the daily trading volume (about 38% of a typical 50,000-share daily average).

By adhering to these liquidity constraints, I can confidently deploy large sums of capital in stocks without exceeding 1% of the daily trading volume. This ensures that my trades don’t move the market or attract unwanted attention from larger players or algorithms, allowing me to act efficiently without becoming a "whale" and disrupting normal market behavior.

1. **Market Dynamics**: In high-frequency environments, large trades can attract unwanted attention from algorithms. The liquidity constraint ensures minimal market disruption.

Data Extraction for Trading Strategy

After filtering for liquidity, I extracted 100 days of **Open**, **Close**, **High**, **Low**, and **Volume** data starting from the event announcement date. This data formed the basis for technical analysis and signal generation.

Creation of Baseline Analysis Techniques: Leveraging Technical Analysis from Personal Equity Trading Experience

Intraday vs. Daily Data

Where available, I utilized **intraday data** for signal generation, as index rebalancing events often lead to short-term price distortions. For stocks without intraday data, I relied on **daily**

data. Intraday data provides more precise entries and exits, especially in volatile conditions.

Technical Indicators and Their Role

I chose **RSI**, **Bollinger Bands**, **MACD**, and **ATR** for detecting overbought conditions and momentum shifts. Here's how each indicator contributes to the strategy:

- **RSI**: Identifies overbought conditions, signaling when to enter a short position.
- **Bollinger Bands**: Detect price deviations from average volatility, triggering short signals when prices exceed the upper band.
- **MACD**: Confirms bearish momentum, particularly when it crosses below the signal line.
- **ATR**: Measures volatility and sets stop-loss levels, avoiding premature exits during high volatility.

Strategy Insights from Personal Trading Experience

Drawing from my trading experience, I often observed retail traders "**holding the bag**"—continuing to hold stocks after their fair value had been distorted by market events. This strategy exploits such distortions by shorting **one-off event** stocks after their prices temporarily rise due to index additions.

Volume Dynamics in One-off Events

Index additions often trigger a **spike in volume**. This temporary volume increase allows my strategy to capitalize on price distortions. The strategy benefits from the natural price reversion that occurs once the initial excitement surrounding index inclusion fades.

Trading Constraints and Strategy Adjustments

Given the restriction against **intraday buying and selling**, I adopted a conservative approach—focusing on **shorting overbought stocks** rather than long positions. I named this strategy **"Mr. Bear"**, as it exploits overbought conditions to capture price reversals. The strategy is designed to re-enter after stop-losses if overbought conditions persist, offering flexibility.

Simplicity over Complexity

Initially, I considered more complex approaches such as **pairs trading** and **cointegration**. However, I chose to focus on a simpler, momentum-based approach first, which yielded strong results. Simplicity often proves highly effective when backed by a deep understanding of market dynamics.

Although I did not implement predictive models in this strategy, I would explore more complex techniques (e.g., **regularized regression with technical indicators** and **machine learning models**) to predict future events or test the strategy on unseen data in a real-world environment. If you're interested in my predictive models using **ML algorithms** and

Statistical Computing techniques, feel free to check out my **Quantitative Research Project** on my GitHub.

Statistical Assumptions and Validation

In this strategy, I assumed that **price reversion** would naturally occur after index inclusion due to overbought conditions. This assumption relies on **mean reversion** theory, which posits that prices will revert to their average over time. While my focus was on one-off events, further testing would involve ensuring the strategy's robustness by testing it on:

- **Different market conditions**: How does the strategy perform during a bull market or a bear market?
- **Cross-validation**: Implementing walk-forward optimization techniques to ensure that the parameters (e.g., RSI thresholds) aren't overfitted to the historical data.
- **Out-of-sample testing**: Applying the strategy to unseen data to validate its predictive power.

Robustness and Risk Management

Future iterations would include **robustness checks** by adding:

- **Monte Carlo simulations**: To evaluate the strategy's performance under different random market scenarios.
- **Backtesting on additional indices or asset classes**: To see if the strategy generalizes well beyond the S&P 500 index.
- **Hedging enhancements**: Exploring dynamic hedging strategies to adjust the SPY hedge based on market conditions.

Strategy Workflow:

1. **Data Cleaning**:
    - Handled missing values and cleaned ticker symbols.
    - Validated tickers using `yfinance`.
2. **Liquidity Filtering**:
    - Set a minimum liquidity threshold of 50,000 shares, representing 1% of the 20-day average trading volume.
3. **Data Extraction**:
    - Extracted 100 days of Open, Close, High, Low, and Volume data for **One-off** and **Scheduled Events**.
4. **Technical Indicators**:

- **RSI** identified overbought conditions for short entries.
- **Bollinger Bands** and **MACD** confirmed momentum, while **ATR** informed stop-loss levels and presence of volatility.

5. **Strategy Execution**:
    - **Signal Generation**: Short positions triggered by overbought conditions and excessive price moves.
    - **Exit Strategy**: Exited below the lower Bollinger Band or at 20% profit; stop-losses were triggered by continued momentum.
6. **Portfolio Allocation**:
    - Allocated 19% of the portfolio to each liquid stock and reserved 24% for the SPY hedge.
    - The Mathematical reasoning for this is that I wanted my aggressive bearish strategy to dominate, but also consider instances where I may be getting stopped out repeatedly and in these instances, I am able to reduce the losses or offset them with the SPY hedge.
7. **SPY Hedge**:
    - Used a long SPY position to hedge against market risk and mitigate losses during uptrends.
8. **Transaction and Overnight Costs**:
    - Included transaction costs ($0.01 per share) and overnight holding costs (based on the **Fed Funds Rate**).
9. **Performance Evaluation**:
    - Assessed profit and loss for each stock, calculated cumulative returns, and factored in all costs.

Key Takeaways:

1. **Liquidity Constraints**: Ensured smooth execution and minimal market impact.
2. **Effective Use of Indicators**: The combination of RSI and Bollinger Bands effectively captured overbought conditions for short entries.
3. **SPY Hedge**: Provided protection during market-wide uptrends, offsetting potential short strategy losses.
4. **Cost Management**: Factored in transaction and overnight costs for accurate profit calculations.
5. **Simplicity Wins**: A straightforward strategy focusing on technical analysis proved highly effective. Future iterations could include more complex models, but simplicity often works well with a strong understanding of the market.

Conclusion:

This project demonstrates how quantitative strategies can exploit price distortions caused by index rebalancing events. By focusing on **One-off Events** and using technical indicators, I developed a robust short-selling strategy with integrated risk management. While this approach was backward-looking and only applied to historical data, future improvements could involve using predictive models, regularized regression, or machine learning techniques to forecast events. Simplicity in this strategy yielded effective results, but further complexity can be explored if necessary. Check out my **Quantitative Research Project** on my GitHub for more complex statistical analysis.

Code for Data Extraction, Exploration and Cleaning

In [162…

```
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from fredapi import Fred
import numpy as np

# Step 1: Load the Data
na_values = ["Null", "NA", "Nan", "NaN", "-", "", "N/A"]
file_path = "/Users/aidanashrafi/Downloads/Index_Add_Event_Data.xlsx"
dfs = pd.read_excel(file_path, sheet_name=None, na_values=na_values)

# Access the main data sheet
data_df = dfs.get("Data")

# Step 2: Drop rows with any missing values and clean the ticker column
def clean_ticker_column(df):
    df['Ticker'] = df['Ticker'].str.replace('*', '', regex=False)
    df['Ticker'] = df['Ticker'].str.replace('-W', '', regex=False)
    df['Ticker'] = df['Ticker'].str.replace('-V', '', regex=False)
    df['Ticker'] = df['Ticker'].str.replace('-', '', regex=False)
    df['Ticker'] = df['Ticker'].apply(lambda x: x[:-2] if x.endswith('US') else
    df['Ticker'] = df['Ticker'].str.strip().str[:4]  # Strip spaces and limit
    return df

data_df_cleaned = clean_ticker_column(data_df.dropna().copy())

# Step 3: Validate tickers by checking if they have valid data in YFinance
def check_valid_ticker(ticker):
    stock = yf.Ticker(ticker)
    hist = stock.history(period='1mo')
    return not hist.empty

# Filter for valid tickers
data_df_cleaned['Valid_Ticker'] = data_df_cleaned['Ticker'].apply(check_valid_
data_df_cleaned = data_df_cleaned[data_df_cleaned['Valid_Ticker']].drop(columns

# Step 4: Remove scheduled events (March, June, September, December)
scheduled_months = [3, 6, 9, 12]
data_df_cleaned['Month'] = pd.to_datetime(data_df_cleaned['Announced']).dt.mon
one_off_filtered_df = data_df_cleaned[~data_df_cleaned['Month'].isin(scheduled_
scheduled_df = data_df_cleaned[data_df_cleaned['Month'].isin(scheduled_months)]

# Step 5: Calculate 20-day average volume for one-off and scheduled events
```

```
def get_20_day_avg_volume(ticker, announcement_date):
    stock = yf.Ticker(ticker)
    hist = stock.history(start=announcement_date - pd.Timedelta(days=40), end=a
    return hist['Volume'].rolling(window=20).mean().iloc[-1] if not hist.empty

one_off_filtered_df['20_Day_Avg_Volume'] = one_off_filtered_df.apply(
    lambda row: get_20_day_avg_volume(row['Ticker'], row['Announced']), axis=1
one_off_filtered_df['Liquidity_Constrained_Position_Size'] = one_off_filtered_d
    lambda avg_volume: 0.01 * avg_volume if avg_volume else 0)

scheduled_df['20_Day_Avg_Volume'] = scheduled_df.apply(
    lambda row: get_20_day_avg_volume(row['Ticker'], row['Announced']), axis=1
scheduled_df['Liquidity_Constrained_Position_Size'] = scheduled_df['20_Day_Avg_
    lambda avg_volume: 0.01 * avg_volume if avg_volume else 0)

# Step 6: Filter out illiquid stocks for both one-off and scheduled events
position_threshold = 50000

# For one-off events
one_off_liquid_stocks_df = one_off_filtered_df[one_off_filtered_df['Liquidity_(

# For scheduled events
scheduled_liquid_stocks_df = scheduled_df[scheduled_df['Liquidity_Constrained_P

# Combine liquid stocks from both one-off and scheduled events
liquid_stocks_df = pd.concat([one_off_liquid_stocks_df, scheduled_liquid_stocks
print("\n Number of Liquid Stocks: ", len(liquid_stocks_df))
print("\n Number of Liquid Stocks with One-Off Events: ", len(one_off_liquid_st
# Step 7: Extract Open and Close Prices with column labels formatted with '_Cl
def extract_prices(tickers, start_dates, days=100):
    close_prices, open_prices, combined_prices = [], [], []
    for ticker, start_date in zip(tickers, start_dates):
        stock = yf.Ticker(ticker)
        hist = stock.history(start=start_date, end=start_date + pd.Timedelta(da
        if not hist.empty:
            close_series = hist['Close'].reset_index(drop=True).rename(f"{ticke
            open_series = hist['Open'].reset_index(drop=True).rename(f"{ticker}
            close_prices.append(close_series)
            open_prices.append(open_series)

            # Combined open and close prices
            combined_series = pd.DataFrame({
                f"{ticker}_Open": open_series,
                f"{ticker}_Close": close_series
            })
            combined_prices.append(combined_series)

    close_prices_df = pd.concat(close_prices, axis=1) if close_prices else pd.D
    open_prices_df = pd.concat(open_prices, axis=1) if open_prices else pd.Data
    combined_prices_df = pd.concat(combined_prices, axis=1) if combined_prices

    return close_prices_df, open_prices_df, combined_prices_df

# Extract prices for both one-off and scheduled events
tickers_one_off = one_off_liquid_stocks_df['Ticker'].tolist()
start_dates_one_off = one_off_liquid_stocks_df['Announced'].tolist()
close_prices_one_off, open_prices_one_off, combined_prices_one_off = extract_p

tickers_scheduled = scheduled_liquid_stocks_df['Ticker'].tolist()
```

```
start_dates_scheduled = scheduled_liquid_stocks_df['Announced'].tolist()
close_prices_scheduled, open_prices_scheduled, combined_prices_scheduled = ext
```

```
RUSH: No price data found, symbol may be delisted (period=1mo)
PACW: No data found, symbol may be delisted
NEX: No data found, symbol may be delisted
FBHS: No data found, symbol may be delisted
CBTX: No data found, symbol may be delisted
AVID: No data found, symbol may be delisted
HTA: No data found, symbol may be delisted
 Number of Liquid Stocks:  9

 Number of Liquid Stocks with One-Off Events:  4
```

In [162…

```
# Check to make sure I have what I want
liquid_stocks_df
```

Out[1624]:

	Announced	Trade Date	Index Change	Ticker	Action	Last Px	Sector	Shs to Trade	$MM to Trade	Tr
1	2024-07-23	2024-07-25	S&P 400	AVTR	Add	21.17	Healthcare	74808923.0	1583.7	1(
8	2024-05-03	2024-05-07	S&P 600	MARA	Add	17.52	Info Tech	29030300.0	508.6	
42	2023-08-21	2023-08-24	S&P 500	KVUE	Add	22.90	Cons Stap	300389422.0	6878.9	
119	2022-10-17	2022-10-19	S&P 400	AR	Add	33.93	Energy	30910807.0	1048.8	
15	2024-03-26	2024-03-28	S&P 400	ROIV	Add	10.08	Healthcare	37158708.0	374.6	
64	2023-06-05	2023-06-16	S&P 500	PANW	Add	217.24	Information Technology	44093704.0	9578.9	
106	2022-12-19	2022-12-21	S&P 400	SMCI	Add	78.29	Info Tech	5054602.0	395.7	
125	2022-09-23	2022-09-30	S&P 500	PCG	Add	12.61	Utilities	279658925.0	3526.5	1(
155	2022-06-03	2022-06-07	S&P 500	VICI	Add	32.45	Real Estate	146440651.0	4752.0	

Simple Correlation Analysis

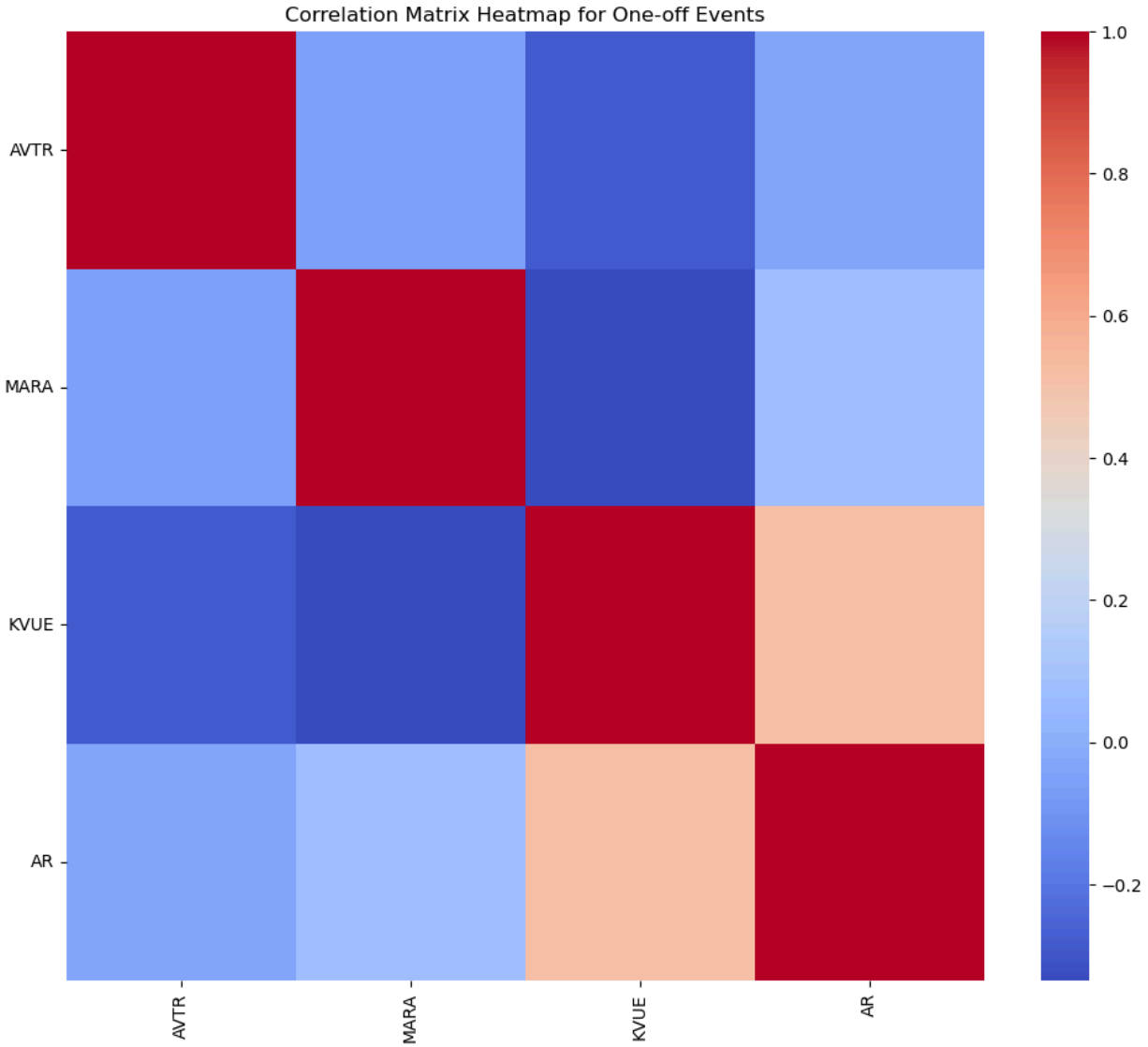
In [162…

```
# Generalized Correlation Analysis Function
def calculate_correlation(df, close_prices_df, event_type="Event"):
    # Get the tickers for the event type
    desired_tickers = df['Ticker'].tolist()

    # Append '_Close' to each ticker to match the columns in `close_prices_df`
    desired_columns = [ticker + '_Close' for ticker in desired_tickers]

    # Filter `close_prices_df` to only include these tickers
```

```
    filtered_close_prices_df = close_prices_df[desired_columns]

    # Calculate the correlation matrix
    correlation_matrix = filtered_close_prices_df.corr()

    # Visualize the correlation matrix using a heatmap
    plt.figure(figsize=(12, 10))
    sns.heatmap(correlation_matrix, annot=False, cmap="coolwarm", xticklabels=(
    plt.title(f"Correlation Matrix Heatmap for {event_type}")
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    plt.show()

    # Print the correlation matrix
    print(f"Full Correlation Matrix for {event_type}:")
    print(correlation_matrix)

    return correlation_matrix
```

In [162…

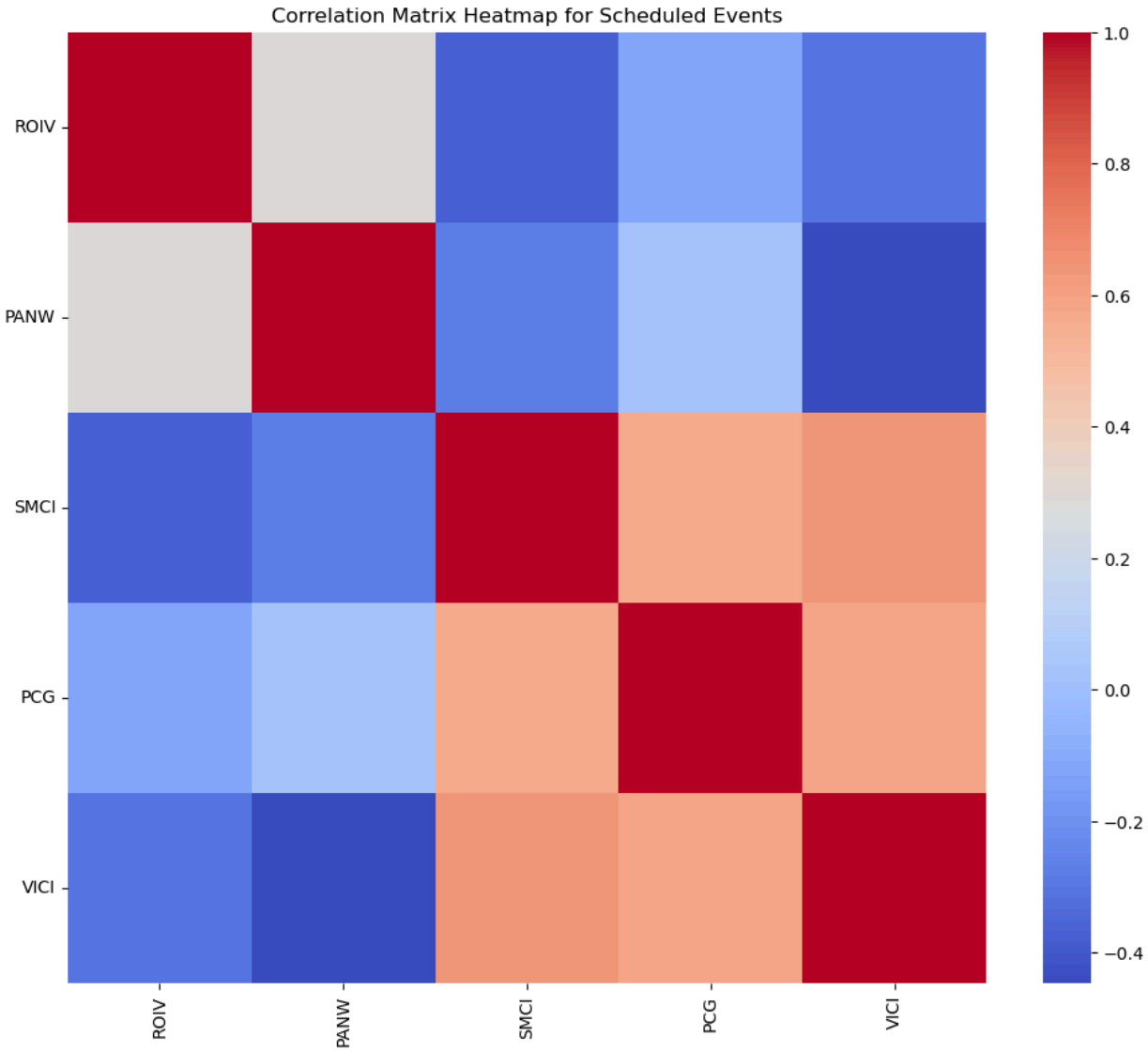
```
# Call the generalized correlation function for one-off events
one_off_correlation_matrix = calculate_correlation(one_off_liquid_stocks_df, c
```



```
Full Correlation Matrix for One-off Events:
            AVTR_Close  MARA_Close  KVUE_Close  AR_Close
AVTR_Close    1.000000   -0.053166   -0.284631 -0.038350
MARA_Close   -0.053166    1.000000   -0.335444  0.077621
KVUE_Close   -0.284631   -0.335444    1.000000  0.521008
AR_Close     -0.038350    0.077621    0.521008  1.000000
```

In [162…

```
# Call the generalized correlation function for scheduled events
scheduled_correlation_matrix = calculate_correlation(scheduled_liquid_stocks_d
```




```
Full Correlation Matrix for Scheduled Events:
            ROIV_Close  PANW_Close  SMCI_Close  PCG_Close  VICI_Close
ROIV_Close    1.000000    0.295145   -0.376189  -0.114666   -0.301854
PANW_Close    0.295145    1.000000   -0.278827   0.032413   -0.446576
SMCI_Close   -0.376189   -0.278827    1.000000   0.571259    0.635941
PCG_Close    -0.114666    0.032413    0.571259   1.000000    0.591788
VICI_Close   -0.301854   -0.446576    0.635941   0.591788    1.000000
```

FRED Data Extraction (Targeted for Stocks I selected)

In [162…

```
# FRED DATA EXTRACTION
fred = Fred(api_key='91ee23f35b975fbad796eab7bfe974a9')

# Get the first and last announcement dates from your cleaned dataset
```

```
first_announcement_date = one_off_liquid_stocks_df['Announced'].min()
last_announcement_date = one_off_liquid_stocks_df['Announced'].max()

# Define the extended end date (last announcement date + 100 days)
extended_end_date = last_announcement_date + pd.Timedelta(days=100)

# Fetch Fed Funds rate data from FRED for the entire date range
fed_funds_data = fred.get_series('FEDFUNDS', observation_start=first_announceme

# Convert the fetched data into a DataFrame for easier handling
fed_funds_df = pd.DataFrame(fed_funds_data, columns=['Fed Funds Rate'])
fed_funds_df['Date'] = fed_funds_df.index
fed_funds_df.reset_index(drop=True, inplace=True)

# Calculate long and short position costs
fed_funds_df['Long Position Cost'] = (fed_funds_df['Fed Funds Rate'] + 1.5)/365
fed_funds_df['Short Position Cost'] = (fed_funds_df['Fed Funds Rate'] + 1.0)/36

# Display the first few rows to verify the data
print(fed_funds_df.head(n=10))
print("\n")
print(fed_funds_df.tail())  # Check last few rows to ensure data is complete fo
```

```
   Fed Funds Rate       Date  Long Position Cost  Short Position Cost
0            3.08 2022-10-01            0.012548             0.011178
1            3.78 2022-11-01            0.014466             0.013096
2            4.10 2022-12-01            0.015342             0.013973
3            4.33 2023-01-01            0.015973             0.014603
4            4.57 2023-02-01            0.016630             0.015260
5            4.65 2023-03-01            0.016849             0.015479
6            4.83 2023-04-01            0.017342             0.015973
7            5.06 2023-05-01            0.017973             0.016603
8            5.08 2023-06-01            0.018027             0.016658
9            5.12 2023-07-01            0.018137             0.016767


    Fed Funds Rate       Date  Long Position Cost  Short Position Cost
19            5.33 2024-05-01            0.018712             0.017342
20            5.33 2024-06-01            0.018712             0.017342
21            5.33 2024-07-01            0.018712             0.017342
22            5.33 2024-08-01            0.018712             0.017342
23            5.13 2024-09-01            0.018164             0.016795
```

Technical Analysis Indicators Construction

In [162…

```
# Technical Analysis Indicators Construction
import yfinance as yf
import pandas as pd
import talib  # TA-Lib for technical analysis

# Function to add customized RSI, MACD, ATR, and Bollinger Bands (BB) using TA-
def add_ta_indicators(df, is_daily=False):
    # Adjust indicator time periods based on whether it's intraday or daily da
    if is_daily:
        rsi_period = 2  # Use longer periods for daily data
        macd_fast = 2
        macd_slow = 4
        macd_signal = 2
        atr_period = 2
```

```
        bb_period = 2
    else:
        rsi_period = 6  # Use shorter periods for intraday data
        macd_fast = 6
        macd_slow = 12
        macd_signal = 6
        atr_period = 12
        bb_period = 12

    df['RSI'] = talib.RSI(df['Close'], timeperiod=rsi_period)
    df['MACD'], df['MACD_Signal'], _ = talib.MACD(df['Close'], fastperiod=macd_
    df['ATR'] = talib.ATR(df['High'], df['Low'], df['Close'], timeperiod=atr_p
    df['BB_upper'], df['BB_middle'], df['BB_lower'] = talib.BBANDS(df['Close']

    return df

# Function to fetch intraday or daily data based on available range
def fetch_stock_data(ticker, start_date, end_date, is_daily=False):
    try:
        if not is_daily:
            # Attempt to fetch 1-hour intraday data
            stock_data = yf.download(ticker, start=start_date, end=end_date, i
            if stock_data.empty:
                raise ValueError(f"No intraday data for {ticker}, switching to
        else:
            # Fetch daily data
            stock_data = yf.download(ticker, start=start_date, end=end_date, i
            print(f"Fetched daily data for {ticker}.")
    except Exception as e:
        print(e)
        # Fallback to daily data if needed
        stock_data = yf.download(ticker, start=start_date, end=end_date, inter
        print(f"\nFetched daily data for {ticker} instead.")
    return stock_data

# Iterate over each ticker and determine if it should use intraday or daily da
indicator_dfs = {}
for index, row in one_off_liquid_stocks_df.iterrows():
    ticker = row['Ticker']
    start_date = pd.to_datetime(row['Announced'])  # Announcement date
    end_date = start_date + pd.Timedelta(days=100)  # 100-day window

    # Check if the ticker requires daily data (due to the intraday limit)
    is_daily = ticker == 'AR'

    # Fetch stock data with the appropriate interval
    stock_data = fetch_stock_data(ticker, start_date, end_date, is_daily=is_da

    # Check if data is available
    if not stock_data.empty:
        # Add indicators to stock data, adjusting for intraday/daily
        stock_data_with_indicators = add_ta_indicators(stock_data, is_daily=is_

        # Save the data with indicators
        indicator_dfs[ticker] = stock_data_with_indicators

        # Display the first few rows of indicators for each ticker
        print(f"\nTechnical indicators for {ticker}:")
        print(stock_data_with_indicators.head(n=20))
```

```
    else:
        print(f"No data found for {ticker}.")
```

```
[**********100%%**********]  1 of 1 completed
```

Technical indicators for AVTR:

Datetime	Open	High	Low	Close	Adj Close	Volume	RSI	MACD	MACD_Signal	ATR	BB_upper	BB_middle
2024-07-23 09:30:00-04:00	22.000000	22.209999	21.139999	21.360001	21.360001	2795359	NaN	NaN	NaN	NaN	NaN	NaN
2024-07-23 10:30:00-04:00	21.360001	21.500000	21.340000	21.389999	21.389999	945806	NaN	NaN	NaN	NaN	NaN	NaN
2024-07-23 11:30:00-04:00	21.385000	21.514999	21.379999	21.455000	21.455000	1109701	NaN	NaN	NaN	NaN	NaN	NaN
2024-07-23 12:30:00-04:00	21.455000	21.510000	21.410000	21.469999	21.469999	2820658	NaN	NaN	NaN	NaN	NaN	NaN
2024-07-23 13:30:00-04:00	21.469999	21.469999	21.299999	21.309999	21.309999	770420	NaN	NaN	NaN	NaN	NaN	NaN
2024-07-23 14:30:00-04:00	21.315001	21.315001	21.139999	21.165001	21.165001	1066442	NaN	NaN	NaN	NaN	NaN	NaN
2024-07-23 15:30:00-04:00	21.155001	21.190001	21.120001	21.160000	21.160000	1888429	26.190282	NaN	NaN	NaN	NaN	NaN
2024-07-24 09:30:00-04:00	22.570000	22.570000	22.030001	22.450001	22.450001	7537419	84.247989	NaN	NaN	NaN	NaN	NaN
2024-07-24 10:30:00-04:00	22.450001	22.695000	22.410000	22.584999	22.584999	3684691	85.664081	NaN	NaN	NaN	NaN	NaN
2024-07-24 11:30:00-04:00	22.590000	22.660000	22.455000	22.639999	22.639999	2766845	86.267638	NaN	NaN	NaN	NaN	NaN
2024-07-24 12:30:00-04:00	22.629999	22.709999	22.490000	22.590000	22.590000	2085174	82.479560	NaN	NaN	NaN	NaN	NaN
2024-07-24 13:30:00-04:00	22.590000	22.705000	22.490000	22.570000	22.570000	2158120	80.776945	NaN	NaN	NaN	NaN	NaN
2024-07-24 14:30:00-04:00	22.565001	22.584999	22.320000	22.434999	22.434999	2521788	69.205528	NaN	NaN	NaN	NaN	NaN
2024-07-24 15:30:00-04:00	22.430000	22.565001	22.360001	22.365000	22.365000	5221524	63.541810	NaN	NaN	NaN	NaN	NaN
2024-07-25 09:30:00-04:00	22.100000	22.770000	22.060101	22.629999	22.629999	4797158	73.422793	NaN	NaN	NaN	NaN	NaN
2024-07-25 10:30:00-04:00	22.639999	22.730000	22.379999	22.674999	22.674999	3814303	74.813763	NaN	NaN	NaN	NaN	NaN
2024-07-25 11:30:00-04:00	22.674999	22.969999	22.620001	22.955000	22.955000	2948580	81.890607	0.329168	NaN	NaN	NaN	NaN
2024-07-25 12:30:00-04:00	22.950001	23.070000	22.895000	23.010000	23.010000	3270116	83.015512	0.328026	NaN	NaN	NaN	NaN
2024-07-25 13:30:00-04:00	23.014999	23.014999	22.711000	22.809999	22.809999	3500075	65.312210	0.286544	NaN	NaN	NaN	NaN
2024-07-25 14:30:00-04:00	22.809999	22.959999	22.639999	22.805000	22.805000	5147255	64.897096	0.248217	NaN	NaN	NaN	NaN

```
2024-07-24 13:30:00-04:00          NaN       NaN  23.080882  21.845417
2024-07-24 14:30:00-04:00          NaN  0.284167  23.172578  21.935000
2024-07-24 15:30:00-04:00          NaN  0.277569  23.227785  22.016250
2024-07-25 09:30:00-04:00          NaN  0.313597  23.318338  22.114166
2024-07-25 10:30:00-04:00          NaN  0.316631  23.387708  22.214583
2024-07-25 11:30:00-04:00     0.377001  0.319411  23.452138  22.351666
2024-07-25 12:30:00-04:00     0.363008  0.307377  23.395112  22.505416
2024-07-25 13:30:00-04:00     0.341161  0.307095  23.021696  22.642916
2024-07-25 14:30:00-04:00     0.314606  0.308171  23.041722  22.672500

                             BB_lower
Datetime
2024-07-23 09:30:00-04:00         NaN
2024-07-23 10:30:00-04:00         NaN
2024-07-23 11:30:00-04:00         NaN
2024-07-23 12:30:00-04:00         NaN
2024-07-23 13:30:00-04:00         NaN
2024-07-23 14:30:00-04:00         NaN
2024-07-23 15:30:00-04:00         NaN
2024-07-24 09:30:00-04:00         NaN
2024-07-24 10:30:00-04:00         NaN
2024-07-24 11:30:00-04:00         NaN
2024-07-24 12:30:00-04:00         NaN
2024-07-24 13:30:00-04:00   20.609952
2024-07-24 14:30:00-04:00   20.697422
2024-07-24 15:30:00-04:00   20.804714
2024-07-25 09:30:00-04:00   20.909995
2024-07-25 10:30:00-04:00   21.041458
2024-07-25 11:30:00-04:00   21.251195
2024-07-25 12:30:00-04:00   21.615721
2024-07-25 13:30:00-04:00   22.264136
2024-07-25 14:30:00-04:00   22.303277
[**********************100%%**********************]  1 of 1 completed
```

Technical indicators for MARA:

Datetime	Open	High	Low	Close
2024-05-03 09:30:00-04:00	17.837999	18.209999	17.469999	17.684999
2024-05-03 10:30:00-04:00	17.690001	17.799999	17.129999	17.215000
2024-05-03 11:30:00-04:00	17.219999	17.615000	17.190001	17.455000
2024-05-03 12:30:00-04:00	17.477600	17.674999	17.299999	17.469900
2024-05-03 13:30:00-04:00	17.459999	17.575001	17.320000	17.525000
2024-05-03 14:30:00-04:00	17.525000	17.584999	17.370001	17.495001
2024-05-03 15:30:00-04:00	17.495001	17.559999	17.420000	17.500000
2024-05-06 09:30:00-04:00	18.410000	19.860001	18.230000	19.780001
2024-05-06 10:30:00-04:00	19.725000	20.299999	17.520000	20.295000
2024-05-06 11:30:00-04:00	20.299999	20.900000	20.230000	20.620001
2024-05-06 12:30:00-04:00	20.620001	21.027201	20.514999	20.840000
2024-05-06 13:30:00-04:00	20.838800	20.850000	20.480000	20.580000
2024-05-06 14:30:00-04:00	20.590000	20.689899	20.309999	20.555000
2024-05-06 15:30:00-04:00	20.549999	20.760000	20.490000	20.695000
2024-05-07 09:30:00-04:00	20.090000	20.680000	20.030001	20.100000
2024-05-07 10:30:00-04:00	20.105000	20.518499	19.910000	20.135000
2024-05-07 11:30:00-04:00	20.120001	20.495001	20.070000	20.275000
2024-05-07 12:30:00-04:00	20.275000	20.278500	19.820000	19.905001
2024-05-07 13:30:00-04:00	19.905001	20.215000	19.900000	19.969900
2024-05-07 14:30:00-04:00	19.965000	20.055000	19.700001	19.905001

Datetime	Adj Close	Volume	RSI	MACD
2024-05-03 09:30:00-04:00	17.684999	16548580	NaN	NaN
2024-05-03 10:30:00-04:00	17.215000	5797519	NaN	NaN
2024-05-03 11:30:00-04:00	17.455000	3923802	NaN	NaN
2024-05-03 12:30:00-04:00	17.469900	2903120	NaN	NaN
2024-05-03 13:30:00-04:00	17.525000	2623998	NaN	NaN
2024-05-03 14:30:00-04:00	17.495001	1830869	NaN	NaN
2024-05-03 15:30:00-04:00	17.500000	2271296	38.650294	NaN
2024-05-06 09:30:00-04:00	19.780001	21990585	85.919503	NaN
2024-05-06 10:30:00-04:00	20.295000	17653613	88.352083	NaN
2024-05-06 11:30:00-04:00	20.620001	15343513	89.699677	NaN
2024-05-06 12:30:00-04:00	20.840000	9990756	90.584529	NaN
2024-05-06 13:30:00-04:00	20.580000	7666281	80.747121	NaN
2024-05-06 14:30:00-04:00	20.555000	6582275	79.747842	NaN
2024-05-06 15:30:00-04:00	20.695000	6400592	81.302772	NaN
2024-05-07 09:30:00-04:00	20.100000	14697881	58.425152	NaN
2024-05-07 10:30:00-04:00	20.135000	8849878	59.234855	NaN
2024-05-07 11:30:00-04:00	20.275000	4311152	62.719934	0.604688
2024-05-07 12:30:00-04:00	19.905001	5401297	49.341832	0.469802
2024-05-07 13:30:00-04:00	19.969900	4053673	51.518474	0.376186
2024-05-07 14:30:00-04:00	19.905001	5456314	48.992392	0.294510

Datetime	MACD_Signal	ATR	BB_upper	BB_middle
2024-05-03 09:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-03 10:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-03 11:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-03 12:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-03 13:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-03 14:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-03 15:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-06 09:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-06 10:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-06 11:30:00-04:00	NaN	NaN	NaN	NaN
2024-05-06 12:30:00-04:00	NaN	NaN	NaN	NaN

```
2024-05-06 13:30:00-04:00          NaN       NaN  21.653540  18.704992
2024-05-06 14:30:00-04:00          NaN  0.762675  21.987049  18.944159
2024-05-06 15:30:00-04:00          NaN  0.721619  22.225468  19.234159
2024-05-07 09:30:00-04:00          NaN  0.716900  22.273858  19.454575
2024-05-07 10:30:00-04:00          NaN  0.707867  22.244235  19.676667
2024-05-07 11:30:00-04:00     0.922404  0.684295  22.132590  19.905834
2024-05-07 12:30:00-04:00     0.793089  0.665479  21.797743  20.106667
2024-05-07 13:30:00-04:00     0.673974  0.636272  20.969465  20.312492
2024-05-07 14:30:00-04:00     0.565556  0.612833  20.949019  20.322909

                              BB_lower
Datetime
2024-05-03 09:30:00-04:00          NaN
2024-05-03 10:30:00-04:00          NaN
2024-05-03 11:30:00-04:00          NaN
2024-05-03 12:30:00-04:00          NaN
2024-05-03 13:30:00-04:00          NaN
2024-05-03 14:30:00-04:00          NaN
2024-05-03 15:30:00-04:00          NaN
2024-05-06 09:30:00-04:00          NaN
2024-05-06 10:30:00-04:00          NaN
2024-05-06 11:30:00-04:00          NaN
2024-05-06 12:30:00-04:00          NaN
2024-05-06 13:30:00-04:00    15.756444
2024-05-06 14:30:00-04:00    15.901268
2024-05-06 15:30:00-04:00    16.242849
2024-05-07 09:30:00-04:00    16.635292
2024-05-07 10:30:00-04:00    17.109099
2024-05-07 11:30:00-04:00    17.679077
2024-05-07 12:30:00-04:00    18.415591
2024-05-07 13:30:00-04:00    19.655518
2024-05-07 14:30:00-04:00    19.696798
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
```

```
Technical indicators for KVUE:
                                Open       High        Low      Close  \
Datetime                                                              
2023-08-21 09:30:00-04:00  23.170000  23.690001  23.030001  23.219999   
2023-08-21 10:30:00-04:00  23.219999  23.309999  23.000000  23.105000   
2023-08-21 11:30:00-04:00  23.108200  23.160000  22.809999  23.110001   
2023-08-21 12:30:00-04:00  23.115000  23.215000  23.000000  23.200001   
2023-08-21 13:30:00-04:00  23.200001  23.219999  23.000000  23.035000   
2023-08-21 14:30:00-04:00  23.030001  23.090000  22.910000  23.004999   
2023-08-21 15:30:00-04:00  23.004999  23.030001  22.900000  22.915001   
2023-08-22 09:30:00-04:00  23.240000  23.510000  23.120001  23.485001   
2023-08-22 10:30:00-04:00  23.480000  23.790001  23.420000  23.650000   
2023-08-22 11:30:00-04:00  23.652201  23.740000  23.490000  23.589899   
2023-08-22 12:30:00-04:00  23.580000  23.790001  23.549999  23.775000   
2023-08-22 13:30:00-04:00  23.775000  23.820000  23.660000  23.735001   
2023-08-22 14:30:00-04:00  23.735001  23.760000  23.549999  23.705000   
2023-08-22 15:30:00-04:00  23.705000  23.780001  23.639999  23.740000   
2023-08-23 09:30:00-04:00  23.850000  24.049999  23.809999  23.995001   
2023-08-23 10:30:00-04:00  24.000000  24.030001  23.760000  23.934999   
2023-08-23 11:30:00-04:00  23.934999  24.040001  23.850000  23.885000   
2023-08-23 12:30:00-04:00  23.889999  24.010000  23.850000  23.975000   
2023-08-23 13:30:00-04:00  23.975000  24.010000  23.910000  23.980000   
2023-08-23 14:30:00-04:00  23.975000  23.990000  23.860001  23.910000   

                           Adj Close    Volume        RSI      MACD  \
Datetime                                                             
2023-08-21 09:30:00-04:00  23.219999  56821714        NaN       NaN   
2023-08-21 10:30:00-04:00  23.105000  23536399        NaN       NaN   
2023-08-21 11:30:00-04:00  23.110001  16240989        NaN       NaN   
2023-08-21 12:30:00-04:00  23.200001  13580429        NaN       NaN   
2023-08-21 13:30:00-04:00  23.035000  13730587        NaN       NaN   
2023-08-21 14:30:00-04:00  23.004999  11360584        NaN       NaN   
2023-08-21 15:30:00-04:00  22.915001  13579082  19.192133       NaN   
2023-08-22 09:30:00-04:00  23.485001  20511423  66.072989       NaN   
2023-08-22 10:30:00-04:00  23.650000  13045031  71.763385       NaN   
2023-08-22 11:30:00-04:00  23.589899   7479446  66.861623       NaN   
2023-08-22 12:30:00-04:00  23.775000   5764865  73.540966       NaN   
2023-08-22 13:30:00-04:00  23.735001   9058206  69.888142       NaN   
2023-08-22 14:30:00-04:00  23.705000  10421158  66.897447       NaN   
2023-08-22 15:30:00-04:00  23.740000  13044199  68.768466       NaN   
2023-08-23 09:30:00-04:00  23.995001  15434574  79.097696       NaN   
2023-08-23 10:30:00-04:00  23.934999  14410756  72.342064       NaN   
2023-08-23 11:30:00-04:00  23.885000   8730234  66.649808  0.185518   
2023-08-23 12:30:00-04:00  23.975000   8322502  71.494663  0.177968   
2023-08-23 13:30:00-04:00  23.980000   7555487  71.768038  0.166242   
2023-08-23 14:30:00-04:00  23.910000   8156240  61.808070  0.142616   

                           MACD_Signal       ATR   BB_upper  BB_middle  \
Datetime                                                                 
2023-08-21 09:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-21 10:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-21 11:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-21 12:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-21 13:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-21 14:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-21 15:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-22 09:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-22 10:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-22 11:30:00-04:00          NaN       NaN        NaN        NaN   
2023-08-22 12:30:00-04:00          NaN       NaN        NaN        NaN   
```

```
2023-08-22 13:30:00-04:00          NaN       NaN  23.909617  23.318742
2023-08-22 14:30:00-04:00          NaN  0.269167  23.982922  23.359158
2023-08-22 15:30:00-04:00          NaN  0.258403  24.048232  23.412075
2023-08-23 09:30:00-04:00          NaN  0.262703  24.168313  23.485825
2023-08-23 10:30:00-04:00          NaN  0.263311  24.247650  23.547075
2023-08-23 11:30:00-04:00     0.198786  0.257202  24.267056  23.617908
2023-08-23 12:30:00-04:00     0.192838  0.249102  24.257797  23.698742
2023-08-23 13:30:00-04:00     0.185239  0.236677  24.107890  23.787492
2023-08-23 14:30:00-04:00     0.173061  0.227787  24.091497  23.822908

                            BB_lower
Datetime
2023-08-21 09:30:00-04:00        NaN
2023-08-21 10:30:00-04:00        NaN
2023-08-21 11:30:00-04:00        NaN
2023-08-21 12:30:00-04:00        NaN
2023-08-21 13:30:00-04:00        NaN
2023-08-21 14:30:00-04:00        NaN
2023-08-21 15:30:00-04:00        NaN
2023-08-22 09:30:00-04:00        NaN
2023-08-22 10:30:00-04:00        NaN
2023-08-22 11:30:00-04:00        NaN
2023-08-22 12:30:00-04:00        NaN
2023-08-22 13:30:00-04:00  22.727866
2023-08-22 14:30:00-04:00  22.735394
2023-08-22 15:30:00-04:00  22.775918
2023-08-23 09:30:00-04:00  22.803337
2023-08-23 10:30:00-04:00  22.846500
2023-08-23 11:30:00-04:00  22.968761
2023-08-23 12:30:00-04:00  23.139687
2023-08-23 13:30:00-04:00  23.467094
2023-08-23 14:30:00-04:00  23.554319
Fetched daily data for AR.

Technical indicators for AR:
                 Open       High        Low      Close  Adj Close    Volume  \
Date
2022-10-17  33.790001  34.490002  32.759998  33.930000  33.930000   7866000
2022-10-18  35.549999  36.869999  35.209999  35.799999  35.799999  19956200
2022-10-19  35.459999  36.830002  35.430000  36.799999  36.799999  50900200
2022-10-20  36.860001  37.090000  34.820000  34.970001  34.970001  10000300
2022-10-21  34.270000  34.310001  32.020000  33.470001  33.470001  12941400
2022-10-24  33.549999  33.900002  32.650002  33.090000  33.090000   6893700
2022-10-25  33.209999  35.700001  32.740002  35.369999  35.369999   6828900
2022-10-26  35.310001  36.209999  35.130001  35.470001  35.470001   7688300
2022-10-27  35.320000  37.290001  34.610001  35.779999  35.779999  14320000
2022-10-28  35.849998  36.990002  34.200001  35.200001  35.200001   7292700
2022-10-31  36.130001  38.119999  35.849998  36.660000  36.660000   9058600
2022-11-01  37.000000  37.570000  35.680000  36.040001  36.040001   5970200
2022-11-02  36.470001  37.070000  35.270000  35.439999  35.439999   6582000
2022-11-03  35.099998  36.150002  34.540001  35.400002  35.400002   5236500
2022-11-04  36.330002  37.509998  35.910000  37.220001  37.220001   5740600
2022-11-07  38.560001  40.189999  38.450001  39.330002  39.330002   6608900
2022-11-08  38.270000  38.799999  37.230000  38.630001  38.630001   6357700
2022-11-09  37.650002  38.169998  35.639999  35.730000  35.730000   6285000
2022-11-10  36.750000  37.799999  35.919998  37.630001  37.630001   4557200
2022-11-11  38.119999  38.490002  36.700001  37.240002  37.240002   7846600

                  RSI      MACD  MACD_Signal       ATR   BB_upper  BB_middle
\
```

```
Date
2022-10-17         NaN       NaN          NaN       NaN        NaN        NaN
2022-10-18         NaN       NaN          NaN       NaN  36.734999  34.865000
2022-10-19  100.000000       NaN          NaN  2.170000  37.299999  36.299999
2022-10-20   43.951013       NaN          NaN  2.220000  37.714998  35.885000
2022-10-21   22.905029 -0.338000     0.086000  2.585001  35.720001  34.220001
2022-10-24   18.432873 -0.518800    -0.317200  1.917500  33.660002  33.280001
2022-10-25   75.600368  0.191386     0.021858  2.438750  36.509998  34.230000
2022-10-26   77.013590  0.309054     0.213322  1.759374  35.520002  35.420000
2022-10-27   83.086934  0.332840     0.293000  2.219687  35.934998  35.625000
2022-10-28   41.779980  0.094173     0.160449  2.504844  36.069998  35.490000
2022-10-31   83.379562  0.410660     0.327256  2.712421  37.389999  35.930000
2022-11-01   51.889909  0.199115     0.241829  2.301210  36.969999  36.350000
2022-11-02   29.977339 -0.056292     0.043082  2.050605  36.340002  35.740000
2022-11-03   28.379543 -0.103028    -0.054325  1.830303  35.459997  35.420000
2022-11-04   87.758575  0.400432     0.248847  1.970150  38.130001  36.310001
2022-11-07   95.811133  0.957009     0.720955  2.470074  40.385002  38.275002
2022-11-08   66.699322  0.626455     0.657955  2.285038  39.680002  38.980001
2022-11-09   18.961717 -0.380044    -0.034044  2.637520  40.080002  37.180000
2022-11-10   58.180950  0.026668     0.006431  2.353760  38.580002  36.680000
2022-11-11   48.537614 -0.003101     0.000076  2.071880  37.825001  37.435001

            BB_lower
Date
2022-10-17       NaN
2022-10-18  32.995001
2022-10-19  35.299999
2022-10-20  34.055002
2022-10-21  32.720001
2022-10-24  32.900000
2022-10-25  31.950001
2022-10-26  35.319998
2022-10-27  35.315002
2022-10-28  34.910002
2022-10-31  34.470001
2022-11-01  35.730001
2022-11-02  35.139997
2022-11-03  35.380003
2022-11-04  34.490002
2022-11-07  36.165001
2022-11-08  38.280001
2022-11-09  34.279999
2022-11-10  34.779999
2022-11-11  37.045002
```

SPY Data Extraction for Hedge

In [163…

```python
# Spy data extraction for hedge
import yfinance as yf
import pandas as pd

# Extract SPY data
def get_spy_data(start_date, end_date):
    spy_data = yf.download('SPY', start=start_date, end=end_date, interval='1d
    return spy_data[['Open', 'Close']]

# Example usage based on first and last announcement dates
first_announcement_date = one_off_liquid_stocks_df['Announced'].min()
```

```
last_announcement_date = one_off_liquid_stocks_df['Announced'].max()
extended_end_date = last_announcement_date + pd.Timedelta(days=100)

# Get SPY data
spy_data = get_spy_data(first_announcement_date, extended_end_date)
print(spy_data.head())

# Allocate Portfolio
portfolio_size = 5000000
spy_allocation = portfolio_size * 0.24  # 24% allocation to SPY
aggressive_allocation = portfolio_size * 0.76  # 76% to aggressive short strat

# SPY Shares held throughout
spy_shares = spy_allocation / spy_data.iloc[0]['Open']  # Buy SPY at the first

spy_data = spy_data.copy()  # Explicitly create a copy of the DataFrame

# Track SPY value over time
spy_data.loc[:, 'SPY_Value'] = spy_shares * spy_data['Close']

# Display the first few rows to verify
print(spy_data.head())
print("\n", spy_data.tail())
```

```
[**********************100%%**********************]  1 of 1 completed
                 Open       Close
Date
2022-10-17  364.010010  366.820007
2022-10-18  375.130005  371.130005
2022-10-19  368.989990  368.500000
2022-10-20  368.029999  365.410004
2022-10-21  365.119995  374.290009
                 Open       Close     SPY_Value
Date
2022-10-17  364.010010  366.820007  1.209263e+06
2022-10-18  375.130005  371.130005  1.223472e+06
2022-10-19  368.989990  368.500000  1.214802e+06
2022-10-20  368.029999  365.410004  1.204615e+06
2022-10-21  365.119995  374.290009  1.233889e+06

                  Open       Close     SPY_Value
Date
2024-10-14  581.219971  584.320007  1.926277e+06
2024-10-15  584.590027  579.780029  1.911310e+06
2024-10-16  579.780029  582.299988  1.919618e+06
2024-10-17  585.909973  582.349976  1.919782e+06
2024-10-18  584.070007  584.590027  1.927167e+06
```

Portfolio Construction and Returns

Trade Signal Generation, Cost Analysis, Profit/Loss Analysis

In [163…

```
# Portfolio allocation - 19% of portfolio per ticker in the aggressive short s

aggressive_allocation_per_ticker = aggressive_allocation / 4  # 19% allocation

# Transaction cost per share
```

```
transaction_cost_per_share = 0.01
total_net_pnl = 0
# Initialize variables for global totals
total_winning_trades = 0
total_losing_trades = 0
total_trades_all_tickers = 0
total_percentage_return = 0
total_average_return_per_trade = 0
win_rate = 0
# Function to calculate transaction costs
def calculate_transaction_costs(num_shares):
    return num_shares * transaction_cost_per_share

# Function to calculate overnight costs (Fed Funds Rate)
def calculate_short_overnight_costs(num_shares, days_held, fed_funds_rate):
    daily_cost_rate = (fed_funds_rate + 1.0) / 365  # Fed funds rate + 1% for s
    return num_shares * days_held * daily_cost_rate

# Function to generate entry signals based on the open price
def generate_trade_signals(df):
    df['Signal'] = 0  # Default no signal

    # Entry conditions based on the open price (only enter at the open)
    condition_1 = df['RSI'] > 70  # RSI overbought condition
    condition_2 = (df['MACD'] > df['MACD_Signal']) & (df['MACD'] > 0)  # MACD >
    condition_3 = df['Open'] > df['BB_middle']  # Price above upper Bollinger l

    # Generate short signals when all conditions are met at the open
    df.loc[condition_1 & condition_2 & condition_3, 'Signal'] = -1  # Short sig

    return df

# Function to generate exit signals (at the next open price or close)
def generate_exit_signals(df, entry_price):

    df['Exit_Signal'] = 0  # Default no exit signal

    # Profit exit condition: Open price < Bollinger Band lower or 20% return or
    profit_exit_condition = (df['Open'] < df['BB_lower']) | (df['Open'] < (ent

    # Loss exit condition
    loss_exit_condition = (df['Close'] > (entry_price * 1.10)) & (df['RSI'] > (

    # Set exit signals
    df.loc[profit_exit_condition, 'Exit_Signal'] = 1  # Exit for profit
    df.loc[loss_exit_condition, 'Exit_Signal'] = 2  # Exit for loss

    return df
```

In [163…

```
percentage_returns_all = []

spy_win = 1  # Since SPY is a winning trade
spy_trade = 1  # One trade for SPY

# Function to simulate P&L with costs (transaction and overnight costs)
def simulate_pnl_with_costs(df, fed_funds_df, ticker):
    winning_trades = 0
    losing_trades = 0
    entry_price = None
    pnl = 0
```

```
    total_costs = 0
    total_trades = 0
    percentage_returns = []
    trade_active = False
    num_shares = 0
    completed_trades = 0

    print(f"\nTicker: {ticker}")

    for i, row in df.iterrows():
        # Enter short position only at open prices
        if row['Signal'] == -1 and not trade_active:
            entry_price = row['Open']
            print(f"Entered at {entry_price} on {i}")

            num_shares = aggressive_allocation_per_ticker // entry_price
            trade_active = True
            total_trades += 1
            transaction_cost = calculate_transaction_costs(num_shares)
            total_costs += transaction_cost
            entry_date = row.name

        # Exit position based on open price for profit or loss
        if row['Exit_Signal'] in [1, 2] and trade_active:
            exit_price = row['Open']
            trade_active = False
            profit = num_shares * (entry_price - exit_price)

            pct_return = (profit / (num_shares * entry_price)) * 100
            percentage_returns.append(pct_return)
            percentage_returns_all.append(pct_return)

            if pct_return > 0:
                winning_trades += 1
            else:
                losing_trades += 1

            transaction_cost = calculate_transaction_costs(num_shares)
            total_costs += transaction_cost

            days_held = (row.name - entry_date).days
            if days_held == 0:
                days_held = 1

            trade_date = pd.to_datetime(i).floor('D').tz_localize(None)
            fed_funds_rate = fed_funds_df.loc[trade_date, 'Fed Funds Rate'] if
            overnight_cost = calculate_short_overnight_costs(num_shares, days_
            total_costs += overnight_cost
            pnl += profit

            print(f"Exited at {exit_price} on {i}, P/L: {profit}, Return: {pct_

            completed_trades += 1

    # Compute overall metrics after processing all rows
    cumulative_return = sum(percentage_returns)
    if completed_trades > 0:
        win_rate = (winning_trades / (winning_trades + losing_trades)) * 100
        avg_return_per_trade = cumulative_return / completed_trades
    else:
```

```
        win_rate = 0
        avg_return_per_trade = 0

    # Print summary of key metrics
    print(f"\n--- Summary for {ticker} ---")
    print(f"Total P/L: {pnl}")
    print(f"Total Trades: {completed_trades}")
    print(f"Winning Trades: {winning_trades}")
    print(f"Losing Trades: {losing_trades}")
    print(f"Win Rate: {win_rate:.2f}%")
    print(f"Average Return per Trade: {avg_return_per_trade:.2f}%")
    print(f"Total Costs: {total_costs}")
    print(f"Cumulative Percentage Return: {cumulative_return:.2f}%")

    return pnl, cumulative_return, completed_trades, total_costs, winning_trad
```

In [163…

```
# Function to calculate the last trade date from the aggressive strategy
def get_last_trade_date(indicator_dfs):
    last_trade_date = None

    # Iterate through all the tickers
    for ticker, df in indicator_dfs.items():
        # Generate signals for entry
        df_with_signals = generate_trade_signals(df)

        # Check if any entry signals exist
        if (df_with_signals['Signal'] == -1).any():
            # Take the first entry price from signals
            entry_price = df_with_signals.loc[df_with_signals['Signal'] == -1,

            # Generate exit signals based on open/close rules
            df_with_signals = generate_exit_signals(df_with_signals, entry_pri

            # Check the last date when an exit signal was generated
            last_exit_date = df_with_signals[df_with_signals['Exit_Signal'] !=

            # Ensure the last_exit_date is timezone-naive
            last_exit_date = last_exit_date.tz_localize(None)

            # If there's no existing last_trade_date, or the current ticker's
            if last_trade_date is None or last_exit_date > last_trade_date:
                last_trade_date = last_exit_date

    return last_trade_date

# Example usage:
last_trade_date = get_last_trade_date(indicator_dfs)
print(f"Last Trade Date: {last_trade_date}")
```

```
Last Trade Date: 2024-10-16 09:30:00
```

In [163…

```
# Spy Returns
# Normalize the last_trade_date to remove any time component
last_trade_date_naive = last_trade_date.normalize()

# Ensure both SPY data and Fed Funds data have timezone-naive datetime indexes
spy_data.index = spy_data.index.tz_localize(None)

# Convert 'Date' column in fed_funds_df to datetime and set it as index
fed_funds_df['Date'] = pd.to_datetime(fed_funds_df['Date'])  # Ensure 'Date' is
```

```
fed_funds_df.set_index('Date', inplace=True)  # Set 'Date' as the index
fed_funds_df.index = fed_funds_df.index.tz_localize(None)  # Make the index ti

# Forward-fill the Fed Funds Rate data to ensure there is no missing data for 
fed_funds_df.ffill(inplace=True)

# Function to calculate SPY profit/loss based on the last trade date with trans
def calculate_spy_pnl(spy_data, last_trade_date, spy_shares, fed_funds_df):
    # Sell SPY at the close price on the last trade date
    spy_sell_price = spy_data.loc[last_trade_date, 'Close']

    # Calculate SPY profit
    spy_pnl = spy_shares * (spy_sell_price - spy_data.iloc[0]['Open'])

    # Calculate percentage return for SPY
    spy_pct_return = ((spy_sell_price - spy_data.iloc[0]['Open']) / spy_data.i

    # Transaction cost for SPY (buy + sell)
    spy_transaction_cost = spy_shares * 0.01 * 2  # Buy and sell cost

    # Accumulate overnight costs for long SPY position (Fed Funds Rate + 1.5%)
    total_overnight_cost = 0
    for date in pd.date_range(spy_data.index[0], last_trade_date):
        # Align with Fed Funds data (accessing the index, not as a column)
        if date in fed_funds_df.index:
            fed_funds_rate = fed_funds_df.loc[date, 'Fed Funds Rate']
        else:
            fed_funds_rate = fed_funds_df['Fed Funds Rate'].iloc[-1]  # Use las

        daily_cost_rate = (fed_funds_rate + 1.5) / 365  # Long position cost (
        total_overnight_cost += spy_shares * daily_cost_rate

    # Subtract total costs (transaction + overnight) from the profit
    total_costs = spy_transaction_cost + total_overnight_cost
    net_spy_pnl = spy_pnl - total_costs

    return net_spy_pnl, spy_pct_return, spy_transaction_cost, total_overnight_

# Calculate SPY profit/loss
spy_pnl, spy_pct_return, spy_transaction_cost, total_overnight_cost = calculat

spy_net_pnl = spy_pnl - spy_transaction_cost - total_overnight_cost
# Print SPY P/L, transaction cost, overnight cost, and returns
print(f"SPY Profit/Loss: {spy_pnl:.2f}")
print(f"SPY Percentage Return: {spy_pct_return:.2f}%")
print(f"SPY Transaction Cost: {spy_transaction_cost:.2f}")
print(f"SPY Total Overnight Cost: {total_overnight_cost:.2f}")
print(f"SPY Net Profit/Loss after Costs: {(spy_pnl - spy_transaction_cost - to
```

```
SPY Profit/Loss: 675797.12
SPY Percentage Return: 59.97%
SPY Transaction Cost: 65.93
SPY Total Overnight Cost: 43754.45
SPY Net Profit/Loss after Costs: 631976.74
```

In [163…

```
# Adding the SPY hedge trade
def include_spy_in_totals(spy_pct_return, spy_net_pnl):
    global total_winning_trades, total_trades_all_tickers, total_percentage_re

    # Since SPY was a winning trade, we add it as 1 winning trade and 1 total 
```

```
    total_winning_trades += spy_win
    total_trades_all_tickers += spy_trade
    total_percentage_return += spy_pct_return
    total_net_pnl += spy_net_pnl

# Loop over each ticker in your data
for ticker, df in indicator_dfs.items():
    df_with_signals = generate_trade_signals(df)

    if (df_with_signals['Signal'] == -1).any():
        entry_price = df_with_signals.loc[df_with_signals['Signal'] == -1, 'Op
        df_with_signals = generate_exit_signals(df_with_signals, entry_price)
        total_pnl, cumulative_return, total_trades, total_costs, wins, losses,
        total_net_pnl += (total_pnl - total_costs)
        total_winning_trades += wins
        total_losing_trades += losses
        total_trades_all_tickers += total_trades
        total_percentage_return += cumulative_return

# Now include the SPY hedge returns
include_spy_in_totals(spy_pct_return, spy_net_pnl)

# Final calculations with SPY hedge
overall_win_rate = (total_winning_trades / total_trades_all_tickers) * 100 if
overall_avg_return_per_trade = total_percentage_return / total_trades_all_tick
```

```
Ticker: AVTR
Entered at 24.110000610351562 on 2024-07-26 09:30:00-04:00
Exited at 24.969999313354492 on 2024-09-04 09:30:00-04:00, P/L: -33885.6688957
21436, Return: -3.57%, Days Held: 40, Overnight Cost: 4318.027397260274, Trans
action Cost: 394.02
Entered at 25.364999771118164 on 2024-09-05 14:30:00-04:00
Exited at 26.809999465942383 on 2024-09-17 14:30:00-04:00, P/L: -54119.5735702
51465, Return: -5.70%, Days Held: 12, Overnight Cost: 1231.331506849315, Trans
action Cost: 374.53000000000003
Entered at 27.100000381469727 on 2024-09-18 13:30:00-04:00
Exited at 25.170000076293945 on 2024-10-02 09:30:00-04:00, P/L: 67656.16069793
701, Return: 7.12%, Days Held: 13, Overnight Cost: 1248.5342465753424, Transac
tion Cost: 350.55
Entered at 24.700000762939453 on 2024-10-14 09:30:00-04:00
Exited at 24.770000457763672 on 2024-10-16 09:30:00-04:00, P/L: -2692.25826263
42773, Return: -0.28%, Days Held: 2, Overnight Cost: 210.74520547945207, Trans
action Cost: 384.61

--- Summary for AVTR ---
Total P/L: -23041.340030670166
Total Trades: 4
Winning Trades: 1
Losing Trades: 3
Win Rate: 25.00%
Average Return per Trade: -0.61%
Total Costs: 10016.058356164383
Cumulative Percentage Return: -2.43%

Ticker: MARA
Entered at 18.030000686645508 on 2024-05-15 10:30:00-04:00
Exited at 20.479999542236328 on 2024-05-28 14:30:00-04:00, P/L: -129087.989702
22473, Return: -13.59%, Days Held: 13, Overnight Cost: 1876.5945205479452, Tra
nsaction Cost: 526.89
Entered at 20.149999618530273 on 2024-06-04 11:30:00-04:00
Exited at 18.44499969482422 on 2024-06-11 10:30:00-04:00, P/L: 80383.926403045
65, Return: 8.46%, Days Held: 6, Overnight Cost: 775.0027397260274, Transactio
n Cost: 471.46000000000004
Entered at 20.315000534057617 on 2024-06-12 09:30:00-04:00
Exited at 19.084999084472656 on 2024-06-14 12:30:00-04:00, P/L: 57518.55778694
153, Return: 6.05%, Days Held: 2, Overnight Cost: 256.23561643835615, Transact
ion Cost: 467.63
Entered at 19.84000015258789 on 2024-06-17 13:30:00-04:00
Exited at 19.860000610351562 on 2024-07-05 09:30:00-04:00, P/L: -957.681919097
9004, Return: -0.10%, Days Held: 17, Overnight Cost: 2230.167123287671, Transa
ction Cost: 478.83
Entered at 19.219999313354492 on 2024-07-12 09:30:00-04:00
Exited at 20.165000915527344 on 2024-07-25 09:30:00-04:00, P/L: -46708.5941905
97534, Return: -4.92%, Days Held: 13, Overnight Cost: 1760.413698630137, Trans
action Cost: 494.27000000000004
Entered at 21.375 on 2024-07-26 12:30:00-04:00
Exited at 13.780099868774414 on 2024-08-05 09:30:00-04:00, P/L: 337547.7414321
8994, Return: 35.53%, Days Held: 9, Overnight Cost: 1095.8794520547945, Transa
ction Cost: 444.44
Entered at 17.21500015258789 on 2024-08-08 12:30:00-04:00

--- Summary for MARA ---
Total P/L: 298695.95981025696
Total Trades: 6
Winning Trades: 3
Losing Trades: 3
```

```
Win Rate: 50.00%
Average Return per Trade: 5.24%
Total Costs: 14313.173150684934
Cumulative Percentage Return: 31.44%

Ticker: KVUE
Entered at 22.1299991607666 on 2023-09-11 09:30:00-04:00
Exited at 20.825000762939453 on 2023-09-19 10:30:00-04:00, P/L: 56020.97122192
383, Return: 5.90%, Days Held: 8, Overnight Cost: 940.8876712328768, Transacti
on Cost: 429.28000000000003
Entered at 21.209999084472656 on 2023-09-20 11:30:00-04:00
Exited at 19.600000381469727 on 2023-10-04 10:30:00-04:00, P/L: 72111.84190750
122, Return: 7.59%, Days Held: 13, Overnight Cost: 1595.2602739726028, Transac
tion Cost: 447.90000000000003
Entered at 20.049999237060547 on 2023-10-06 11:30:00-04:00
Exited at 19.395000457763672 on 2023-10-11 10:30:00-04:00, P/L: 31034.49716186
5234, Return: 3.27%, Days Held: 4, Overnight Cost: 519.2438356164383, Transact
ion Cost: 473.81
Entered at 19.489999771118164 on 2023-10-16 11:30:00-04:00
Exited at 19.520000457763672 on 2023-10-23 09:30:00-04:00, P/L: -1462.29346847
53418, Return: -0.15%, Days Held: 6, Overnight Cost: 801.2383561643836, Transa
ction Cost: 487.42
Entered at 19.934999465942383 on 2023-10-25 10:30:00-04:00
Exited at 18.850000381469727 on 2023-10-26 09:30:00-04:00, P/L: 51704.54637145
996, Return: 5.44%, Days Held: 1, Overnight Cost: 130.55890410958904, Transact
ion Cost: 476.54
Entered at 18.700000762939453 on 2023-11-01 12:30:00-04:00
Exited at 19.0049991607666 on 2023-11-13 10:30:00-05:00, P/L: -15494.528606414
795, Return: -1.63%, Days Held: 11, Overnight Cost: 1531.0191780821917, Transa
ction Cost: 508.02000000000004
Entered at 19.239999771118164 on 2023-11-14 09:30:00-05:00

--- Summary for KVUE ---
Total P/L: 193915.0345878601
Total Trades: 6
Winning Trades: 4
Losing Trades: 2
Win Rate: 66.67%
Average Return per Trade: 3.40%
Total Costs: 11657.908219178084
Cumulative Percentage Return: 20.41%

Ticker: AR
Entered at 36.130001068115234 on 2022-10-31 00:00:00
Exited at 35.099998474121094 on 2022-11-03 00:00:00, P/L: 27081.85820388794, R
eturn: 2.85%, Days Held: 3, Overnight Cost: 216.1068493150685, Transaction Cos
t: 262.93
Entered at 36.33000183105469 on 2022-11-04 00:00:00
Exited at 38.27000045776367 on 2022-11-08 00:00:00, P/L: -50729.02408981323, R
eturn: -5.34%, Days Held: 4, Overnight Cost: 286.56438356164387, Transaction C
ost: 261.49
Entered at 37.68000030517578 on 2022-11-23 00:00:00
Exited at 33.5 on 2022-12-16 00:00:00, P/L: 105386.1676940918, Return: 11.09%,
Days Held: 23, Overnight Cost: 1588.7013698630137, Transaction Cost: 252.12
Entered at 34.189998626708984 on 2022-12-21 00:00:00
Exited at 30.5 on 2022-12-29 00:00:00, P/L: 102526.61184310913, Return: 10.7
9%, Days Held: 8, Overnight Cost: 608.986301369863, Transaction Cost: 277.85
Entered at 29.31999969482422 on 2023-01-09 00:00:00
Exited at 29.559999465942383 on 2023-01-13 00:00:00, P/L: -7776.232583999634,
Return: -0.82%, Days Held: 4, Overnight Cost: 355.0794520547945, Transaction C
```

```
ost: 324.01
Entered at 30.649999618530273 on 2023-01-17 00:00:00
Exited at 28.729999542236328 on 2023-01-19 00:00:00, P/L: 59510.402364730835,
Return: 6.26%, Days Held: 2, Overnight Cost: 169.83561643835617, Transaction C
ost: 309.95

--- Summary for AR ---
Total P/L: 235999.78343200684
Total Trades: 6
Winning Trades: 4
Losing Trades: 2
Win Rate: 66.67%
Average Return per Trade: 4.14%
Total Costs: 6601.973972602739
Cumulative Percentage Return: 24.84%
Exited at 20.165000915527344 on 2024-07-25 09:30:00-04:00, P/L: -46708.5941905
97534, Return: -4.92%, Days Held: 13, Overnight Cost: 1760.413698630137, Trans
action Cost: 494.27000000000004
Entered at 21.375 on 2024-07-26 12:30:00-04:00
Exited at 13.780099868774414 on 2024-08-05 09:30:00-04:00, P/L: 337547.7414321
8994, Return: 35.53%, Days Held: 9, Overnight Cost: 1095.8794520547945, Transa
ction Cost: 444.44
Entered at 17.21500015258789 on 2024-08-08 12:30:00-04:00

--- Summary for MARA ---
Total P/L: 298695.95981025696
Total Trades: 6
Winning Trades: 3
Losing Trades: 3
Win Rate: 50.00%
Average Return per Trade: 5.24%
Total Costs: 14313.173150684934
Cumulative Percentage Return: 31.44%

Ticker: KVUE
Entered at 22.1299991607666 on 2023-09-11 09:30:00-04:00
Exited at 20.825000762939453 on 2023-09-19 10:30:00-04:00, P/L: 56020.97122192
383, Return: 5.90%, Days Held: 8, Overnight Cost: 940.8876712328768, Transacti
on Cost: 429.28000000000003
Entered at 21.209999084472656 on 2023-09-20 11:30:00-04:00
Exited at 19.600000381469727 on 2023-10-04 10:30:00-04:00, P/L: 72111.84190750
122, Return: 7.59%, Days Held: 13, Overnight Cost: 1595.2602739726028, Transac
tion Cost: 447.90000000000003
Entered at 20.049999237060547 on 2023-10-06 11:30:00-04:00
Exited at 19.395000457763672 on 2023-10-11 10:30:00-04:00, P/L: 31034.49716186
5234, Return: 3.27%, Days Held: 4, Overnight Cost: 519.2438356164383, Transact
ion Cost: 473.81
Entered at 19.489999771118164 on 2023-10-16 11:30:00-04:00
Exited at 19.520000457763672 on 2023-10-23 09:30:00-04:00, P/L: -1462.29346847
53418, Return: -0.15%, Days Held: 6, Overnight Cost: 801.2383561643836, Transa
ction Cost: 487.42
Entered at 19.934999465942383 on 2023-10-25 10:30:00-04:00
Exited at 18.850000381469727 on 2023-10-26 09:30:00-04:00, P/L: 51704.54637145
996, Return: 5.44%, Days Held: 1, Overnight Cost: 130.55890410958904, Transact
ion Cost: 476.54
Entered at 18.700000762939453 on 2023-11-01 12:30:00-04:00
Exited at 19.0049991607666 on 2023-11-13 10:30:00-05:00, P/L: -15494.528606414
795, Return: -1.63%, Days Held: 11, Overnight Cost: 1531.0191780821917, Transa
ction Cost: 508.02000000000004
Entered at 19.239999771118164 on 2023-11-14 09:30:00-05:00
```

```
--- Summary for KVUE ---
Total P/L: 193915.0345878601
Total Trades: 6
Winning Trades: 4
Losing Trades: 2
Win Rate: 66.67%
Average Return per Trade: 3.40%
Total Costs: 11657.908219178084
Cumulative Percentage Return: 20.41%

Ticker: AR
Entered at 36.130001068115234 on 2022-10-31 00:00:00
Exited at 35.099998474121094 on 2022-11-03 00:00:00, P/L: 27081.85820388794, R
eturn: 2.85%, Days Held: 3, Overnight Cost: 216.1068493150685, Transaction Cos
t: 262.93
Entered at 36.33000183105469 on 2022-11-04 00:00:00
Exited at 38.27000045776367 on 2022-11-08 00:00:00, P/L: -50729.02408981323, R
eturn: -5.34%, Days Held: 4, Overnight Cost: 286.56438356164387, Transaction C
ost: 261.49
Entered at 37.68000030517578 on 2022-11-23 00:00:00
Exited at 33.5 on 2022-12-16 00:00:00, P/L: 105386.1676940918, Return: 11.09%,
Days Held: 23, Overnight Cost: 1588.7013698630137, Transaction Cost: 252.12
Entered at 34.189998626708984 on 2022-12-21 00:00:00
Exited at 30.5 on 2022-12-29 00:00:00, P/L: 102526.61184310913, Return: 10.7
9%, Days Held: 8, Overnight Cost: 608.986301369863, Transaction Cost: 277.85
Entered at 29.31999969482422 on 2023-01-09 00:00:00
Exited at 29.559999465942383 on 2023-01-13 00:00:00, P/L: -7776.232583999634,
Return: -0.82%, Days Held: 4, Overnight Cost: 355.0794520547945, Transaction C
ost: 324.01
Entered at 30.649999618530273 on 2023-01-17 00:00:00
Exited at 28.729999542236328 on 2023-01-19 00:00:00, P/L: 59510.402364730835,
Return: 6.26%, Days Held: 2, Overnight Cost: 169.83561643835617, Transaction C
ost: 309.95

--- Summary for AR ---
Total P/L: 235999.78343200684
Total Trades: 6
Winning Trades: 4
Losing Trades: 2
Win Rate: 66.67%
Average Return per Trade: 4.14%
Total Costs: 6601.973972602739
Cumulative Percentage Return: 24.84%
```

Summary of Returns

In [163…

```
print(f"Hedge: SPY Net P/L after Costs: {spy_net_pnl:.2f}")
print(f"\nTotal Net P/L after Costs for Aggressive Bearish Strategy: {total_ne
print(f"\nTotal Net P/L after Costs: {total_net_pnl:.2f}")
print(f"\nTotal Net Percentage Return without compounding II: {100*(total_net_
# Print the overall metrics
print(f"\nTotal Winning Trades: {total_winning_trades}")
print(f"\nTotal Losing Trades: {total_losing_trades}")
print(f"\nOverall Win Rate: {overall_win_rate:.2f}%")
print(f"\nOverall Average Return per Trade: {overall_avg_return_per_trade:.2f}

import numpy as np
```

```
# Convert percentage_returns_all to a numpy array
returns_array = np.array(percentage_returns_all)

print(f"\nWorst Trade: {returns_array.min():.2f}%")
print(f"\nBest Trade: {returns_array.max():.2f}%")

# Calculate the standard deviation of returns
std_dev_returns = np.std(returns_array)
risk_free_rate = 0.0  # Assumed zero for simplicity
sharpe_ratio = (overall_avg_return_per_trade - risk_free_rate) / std_dev_returı
print(f"\nSharpe Ratio: {sharpe_ratio:.2f}")
```

```
Hedge: SPY Net P/L after Costs: 631976.74

Total Net P/L after Costs for Aggressive Bearish Strategy: 662980.32

Total Net P/L after Costs: 1294957.06

Total Net Percentage Return without compounding II: 25.90%

Total Winning Trades: 13

Total Losing Trades: 10

Overall Win Rate: 56.52%

Overall Average Return per Trade: 5.84%

Worst Trade: -13.59%

Best Trade: 35.53%

Sharpe Ratio: 0.63
```

Results Summary and Performance Evaluation

Key Results:

- **Total Net Profit Across All Trades**: $1,294,957.06
- **Number of Trades**: 23
- **Cumulative Return**: 25.90% cumulative return on the $5,000,000 portfolio.
- **Average Return per Trade**: On average, each trade generated a 5.84% return.
- **Max Drawdown**: The largest observed drawdown was -13.59%, representing the largest portfolio drop during testing.
- **Sharpe Ratio**: The strategy achieved a Sharpe ratio of 0.63, which is decent, but still below the ideal level of 1.0 for a professional quantitative strategy.

SPY Hedge Performance:

- **Hedge Allocation**: 24% of the portfolio was allocated to SPY as a hedge.
- **Hedge Contribution**: The SPY hedge helped to offset losses during broad market uptrends, contributing $631,976.74 to the total portfolio return.

- **Net P&L from Hedge**: The long SPY position contributed 48.82% of the total portfolio return.

Trade Metrics:

- **Win Rate**: 56.52% of trades were profitable, highlighting a balanced strategy with a slight edge in favor of winning trades.
- **Best Trade**: The most profitable trade yielded a 35.53% return.
- **Worst Trade**: The largest loss incurred was -13.59%, signaling the need for better stop-loss measures to mitigate large drawdowns.

Risk and Volatility Management:

- **SPY Hedge Efficiency**: The 24% allocation to SPY as a hedge reduced portfolio volatility and provided consistent protection during market uptrends. It played a significant role in cushioning losses when the market moved against short positions.
- **Risk-Adjusted Returns**: While the Sharpe ratio of 0.63 demonstrates that the strategy provides positive risk-adjusted returns, refining risk management measures such as stop-losses and dynamic hedging could improve this significantly.

Strategy Refinement Considerations:

- **Stop-Loss Improvements**: The -13.59% loss highlights the need for improved stop-loss rules. Incorporating a **dynamic ATR-based stop-loss** would minimize the impact of such large losses, however for such a simplistic strategy it could also deteriorate the effectiveness given the purposeful risk tolerance.
- **Future Enhancements**: Adding more complex strategies such as **cointegration** or **pairs trading**, as well as **predictive machine learning models**, could improve signal generation and the robustness of the strategy. These enhancements could improve the Sharpe ratio and reduce drawdowns in future iterations.

Final Conclusion:

The momentum-based short strategy demonstrated a solid performance, with a **25.90% cumulative return** and an effective **SPY hedge** that reduced overall portfolio risk. The Sharpe ratio of 0.63 suggests a reasonable risk-adjusted return, though there is room for improvement.

To make this strategy more competitive in professional quantitative settings, enhancing **stop-loss mechanisms** and introducing more sophisticated statistical techniques such as **cointegration** or **machine learning models** for predictive analysis would further optimize results. This could potentially push the Sharpe ratio above 1.0, positioning the strategy for use in more demanding environments.

Further out-of-sample testing and backtesting on different asset classes would also be important steps toward verifying the robustness of the model and ensuring its viability in live market conditions.

For more advanced modeling techniques and quantitative research, feel free to explore my **Quantitative Research Project** on GitHub, where I delve deeper into **predictive modeling** and **machine learning algorithms**.