Part 1. Data gathering component

Imports

In [7]:
```
from typing import List, Tuple

import pandas as pd

import yfinance as yf
```

1. Connect to source

I used the `yfinance` library to connect to Yahoo Finance. For bonus, this program is capable of downloading multiple assets, combining them with the associated time column, and saving the data into a csv or Excel file.

In [3]:
```
START_DATE = "2025-02-13"
END_DATE = "2025-02-15"

FIRST_OPTION_DATE = "2025-02-21"
SECOND_OPTION_DATE = "2025-03-21"
THIRD_OPTION_DATE = "2025-04-18"
```

In [4]:
```
def fetch_market_data(tickers: List[str], start_date: str, end_date: str, op
    """
    Fetches historical market data and options chain data for given tickers

    Parameters:
    tickers (List[str]): A list of ticker symbols.
    start_date (str): The start date in 'YYYY-MM-DD' format.
    end_date (str): The end date in 'YYYY-MM-DD' format.
    option_dates (List[str]): A list of option expiration dates in 'YYYY-MM-

    Returns:
    Tuple[pd.DataFrame, pd.DataFrame]: A tuple containing two DataFrames:
        - Historical market data for all tickers with 1-minute intervals.
        - Combined options chain data for the specified expiration dates for
    """
    # Fetch historical market data with 1-minute intervals
    historical_data = yf.download(
        tickers=tickers,
        start=start_date,
        end=end_date,
        interval="1m",
        group_by="ticker",
        prepost=False,
        auto_adjust=True,
        threads=True
```

```
    )
    # Extract Close prices only
    historical_data = historical_data.xs("Close", axis=1, level=1)

    # Drop any rows with missing values (i.e. outside trading hours)
    historical_data = historical_data.dropna(how='any')

    # Initialize a list to collect options data
    options_data_list = []

    # Fetch options chain data for each ticker and specified expiration date
    for ticker in tickers:
        try:
            stock = yf.Ticker(ticker)
            # Get available expiration dates for the ticker
            available_expirations = stock.options
            # print(ticker)
            # print(available_expirations)
            # Filter the provided option_dates to include only those availab
            # If the exact option date isn't available, use the closest avai
            valid_expirations = []
            for date in option_dates:
                if date in available_expirations:
                    valid_expirations.append(date)
                else:
                    closest_date = min(available_expirations, key=lambda x:
                    valid_expirations.append(closest_date)
            for exp_date in valid_expirations:
                opt = stock.option_chain(exp_date)
                calls = opt.calls
                puts = opt.puts
                calls['optionType'] = 'call'
                puts['optionType'] = 'put'
                combined_options = pd.concat([calls, puts])
                combined_options['expirationDate'] = exp_date
                combined_options['ticker'] = ticker
                options_data_list.append(combined_options)
        except Exception as e:
            print(f"Could not retrieve options data for {ticker} on {exp_dat

    # Combine all options data into a single DataFrame
    if options_data_list:
        options_data = pd.concat(options_data_list, ignore_index=True)
    else:
        options_data = pd.DataFrame()

    historical_data.to_csv('historical_data.csv')
    options_data.to_csv('options_data.csv')
    return historical_data, options_data
```

2. Download data on options and equity for symbols

In [5]:
```
df_hist, df_opt = fetch_market_data(["NVDA", "SPY", "^VIX"], START_DATE, END
df_hist.head()
```

```
[**********100%**********]  3 of 3 completed
```

Out[5]:

Ticker	NVDA	SPY	^VIX
Datetime			
2025-02-13 14:31:00+00:00	132.070007	604.789978	15.65
2025-02-13 14:32:00+00:00	132.439896	604.825012	15.66
2025-02-13 14:34:00+00:00	133.059296	604.554993	15.67
2025-02-13 14:35:00+00:00	132.990005	604.520020	15.64
2025-02-13 14:36:00+00:00	132.977905	604.469971	15.67

In [6]:
```
df_opt.head()
```

Out[6]:

	contractSymbol	lastTradeDate	strike	lastPrice	bid	ask	change	p
0	NVDA250221C00000500	2025-02-14 19:45:44+00:00	0.5	137.30	137.70	138.90	2.540008	
1	NVDA250221C00001000	2025-02-13 19:22:16+00:00	1.0	134.05	137.30	138.60	0.000000	
2	NVDA250221C00001500	2025-02-13 19:22:16+00:00	1.5	133.54	136.50	137.85	0.000000	
3	NVDA250221C00002000	2025-02-10 15:23:36+00:00	2.0	132.70	136.15	138.05	0.000000	
4	NVDA250221C00002500	2025-02-10 15:24:36+00:00	2.5	132.15	135.55	136.90	0.000000	

Why are there so many maturities available?

Contracts are issued by the market when there are buyers i.e. demand. So why would someone want to buy an option with a different maturity than the traditional third Friday of monthly options?

Specifically, many of these options are weekly options, which are issued on Thursdays and expire the following Friday. In the final week of monthly options, their prices experience accelerated theta and are strongly influenced by short-term news, like earnings reports. Traders who would like to take advantage of these properties of options are served well by weekly options and other short-dated options.

This only works when there's enough volume and open interest in the options to shrink the spread to the point where a trader can apply the same strategies they use with

monthly options. NVDA, SPY, and ^VIX are all high volume tickers that are constantly influenced by short-term news, so they are good candidates for underlyings that have options with additional maturities.

3. Symbols explanation

NVDA

NVDA is the symbol of NVIDIA, the technology company that designs high-performance GPUs among other products. It is traded on the NASDAQ exchange.

SPY

SPY is the exchange-traded fund (ETF) that tracks the performance of the S&P 500 index. This index represents the 500 largest publicly traded companies in the U.S. This index can be considered representative of market conditions. SPY, as an ETF, can be traded with high liquidity, allowing investors to dynamically adjust their positions in SPY in order to adjust their market exposure.

^VIX

^VIX is the Chicago Board Options Exchange Volatility Index. It weights the implied volatility S&P 500 Index (SPX) options prices expiring between 23 and 37 days in the future on the CBOE to measure the 30-day implied volatility of the S&P 500. As a measure of volatility, it increases in times of investor uncertainty, leading to its moniker "the fear gauge". Although VIX is not itself tradeable, it can be traded on through ETFs like VXX and derivatives like options and futures.

Options Information

`NVDA250221C00028000`

Here's an example option contract symbol. Let's break it down.

- The first letters are the ticker of the underlying, i.e. `NVDA` for NVIDIA.
- The next six numbers are the expiration of date of the option, in the format YYMMDD i.e. `250221` for Februray 21, 2025.
- The next letter indicates whether the option is a call or a put by `C` or `P`.
- The next eight numbers indicate the strike price. If interpreted as a number, you can divide it by 1000 to get the strike price, i.e. `00028000` → $28.

4. Additional Records

Current asset values at time of download

Asset	Ticker	Price (USD)
Nvidia	NVDA	138.86
S&P 500 ETF	SPY	609.70
CBOE Volatility Index	^VIX	14.85

Data sourced from Yahoo Finance.

In [27]:
```
NVDA_CURR = df_hist['NVDA'].iloc[-1]
SPY_CURR = df_hist['SPY'].iloc[-1]
VIX_CURR = df_hist['^VIX'].iloc[-1]
NVDA_CURR, SPY_CURR, VIX_CURR
```

Out[27]:
```
(np.float64(138.86000061035156),
 np.float64(609.7000122070312),
 np.float64(14.850000381469727))
```

Short-term interest rate

Federal funds (effective): 4.33% = 0.0433

In [28]:
```
FED_FUNDS_RATE = 0.0433
```