Shreyank Shetty

Coding Assignment: Tax Profile 2018 and 2019

Problem Statement: Generate Tax profile for 2018 and 2019 from the list of transaction given, by combining trades and calculating short term gains/losses and long term gains/losses.

Input: Assignment.xlsx (list of trades from 2018 -2019)

Output:
Short Term Gains/Losses 2018 : -2,732,805.5210000006 USD
Short Term Gains/Losses 2019 : 9,610,551.166000001 USD
Long Term Gains/Losses 2019 : 3,292,543.3978999997 USD

Code Execution Time:
Average: between 14-15 seconds

Objectives:
- Minimise realised gains.
- Avoid Wash Sales.
- Long terms gains prioritised over Short term gains.

Assumptions Made:
- Capital gains/losses are calculated for a sell or short sale orders.
- Trades execute before 01/01/2019 will be categorised under gains/losses for 2018.
- A sell order can only be combined with buy orders executed before the sell trade.
- A BTC order can only be combined with buy orders executed after the BTC trade. There are no timeframe for when a short sale can be closed. Any buy order made after BTC trade can be matched.
- If sell/BTC order does not have enough buy orders to match quantity, match the existing quantity. Ignore the gains/losses from remaining quantity.
  Stocks where this assumption was applied :
    Sell order : 'ECOL', 'WSO' ,'IAA'
    BTC order : 'IWO' , 'AQUA', 'IWM', 'POOL'
- Trades can only be combined for the same stock.

Approach Used:
1. Gather all trades in the trade sheet for a particular stock.
2. Scan from the first trade (trade date) for the first sell order.
3. Get the buy order with highest trade prices executed before the sell.
4. Match the trades. (Adjust the quantities of the sell and buy order in the dataset)
5. Calculate the capital gains/losses. (Add to cumulative)
6. Continue scan for next sell.
7. After all trades done, pick next stock and repeat from step 1.

Short Sales (BTC) Trades:
Assumption used to handle BTC trades :
A BTC trade is matched by a 'Buy' order that was executed ***after*** the BTC trade.

However there were only 26 BTC trades out of 2738 trades. Out of these 22 belonged to stocks 'IWO' and 'IWM' that had only BTC trades. Hence based on the above assumption of matching, these 22 would be left out.

Hence I've submitted two versions of the solution.

Version 1: Handles BTC trades based on the assumption made above.

Version 2: Ignores all BTC trades completely

Pseudo Code:

Below is the Pseudo code that runs through the main logic of the code. It explains the algorithm for obtaining tax profile by calculating gains/losses for 2018.

Note:

Calculating tax profile for 2019 is also similar with few adjustments. For readability and better understanding, I've left out the code for calculating tax profile for 2019 in the pseudo code.

```
load the assignment.xlsx file

# to work with the data on python
df = Convert excel sheet to pandas dataframe

Add new column to the data frame, called 'isMatch'. This is to keep track of trades that are
matched

uniqueStocks = Define function to get list of unique stocks from the list of trades.

For every stock in unique stock ;
    gains/losses = calculate the tax profile.


Define function to calculate tax profile
calculateTaxProfile:
    df2 = from dataframe  get only trades for the particular stock called. ( new sub data frame)
    date = 01/01/2019
    check if df2 has any SELL or BTC order:
    if yes :
        #loop through trades in df2
        for trades in df2:
            if trade is a 'sell' or 'btc' and TradeDate < date :
                execute recursive function to calculate gains for 2018
            else:
                execute recursive function to calculate gains for 2019


    return gains for 2018,2019
```

```
#define recursive function to calculate gain 2018:
Recursivefunction for 2018 :

    if quantity == 0:
        #exit the recursive call
        return gains
    else:
        highest Buy order =  call function get highest buy order executed before sell.


        if sell quantity > buy quantity :

            short term gain = (sell trade price * buy quantity ) - ( buy trade price * buy quantity)

            add gains to total gains.
            reduced quantity by quantity matched
            mark buy trade as matched.
            call recursivefunction 2018 again ()


        else if sell quantity < buy quantity :

            short term gain = (sell trade price * sell quantity ) - ( buy trade price * sell quantity)

            add gains to total gains.

            reduced quantity to zero
            mark sell trade as matched.


            call recursivefunction 2018 again ()


#function to get highest buy order
HigestBuyOrder:

    HighestTradePrice = 0
    HighestTradeIndex = 0 ( for data frame )

    for trades in df2 ( stock data frame)
        if trade is a ‘buy’ and not already matched before :
            if trade happened before sell and TradePrice > HighestTradePrice
                check if possibility for wash sale
                no wash sale :
                    highest trade (HighestTradeIndex) = trade

    return HighestTradeIndex
```