```
import pandas as pd
from queue import Queue

# Load data from SPY_2016_2021.xlsx
df = pd.read_excel('/Users/wenqicheng/Downloads/
SPY_2016_2021_HI.xlsx')
# Prompt user to enter file name
#filename = input("Enter the name of the financial excel data file
(including extension): ")

# Load data from excel file
#df = pd.read_excel(filename)
print(len(df))

# Set up parameters
short_window = 12
long_window = 26
signal_period = 9
commission = 1/800
starting_capital = 100000

# Initialize variables
trades = Queue()
holdings = 0
holdings_bs=0
balance = starting_capital

# Determine when to buy and sell using the Buy-Hold-Sell strategy
if df['Close'][len(df)-1] > df['Close'][0]:
    # Buy at the beginning and hold until the end
    shares = balance // df['Close'][0]
    cost = shares * df['Close'][0] * (1 + commission)
    holdings_bs += shares
    balance -= cost
    trades.put({'Type': 'Buy', 'Shares': shares, 'Price':
df['Close'][0], 'Balance': balance})
# Sell at the end
proceeds = holdings_bs * df['Close'][len(df)-1] * (1 - commission)
balance += proceeds
trades.put({'Type': 'Sell_bs', 'Shares': holdings_bs, 'Price':
df['Close'][len(df)-1], 'Balance': balance})

# Calculate profit/loss

buy_hold_sell_profit = (holdings_bs * df['Close'][len(df)-1] * (1 -
commission)) - starting_capital
print(buy_hold_sell_profit)



# Choose between SMA or EMA
while True:
    method = input("Select method (SMA/EMA): ").upper()
    if method in ["SMA", "EMA"]:
```

```
        break
    print("Invalid method. Please choose either SMA or EMA.")

if method == "SMA":
    # Calculate short and long SMAs
    df['Short_SMA'] =
df['Close'].rolling(window=short_window+1).mean()
    df['Long_SMA'] =
df['Close'].rolling(window=long_window+1).mean()

    # Calculate MACD
    df['MACD'] = df['Short_SMA'] - df['Long_SMA']

    # Calculate signal line
    df['Signal'] = df['MACD'].rolling(window=signal_period).mean()

    # Calculate MACD histogram
    df['Histogram'] = df['MACD'] - df['Signal']

elif method == "EMA":
    # Calculate short and long EMAs
    df['Short_EMA'] = df['Close'].ewm(span=short_window).mean()
    df['Long_EMA'] = df['Close'].ewm(span=long_window).mean()

    # Calculate MACD
    df['MACD'] = df['Short_EMA'] - df['Long_EMA']

    # Calculate signal line
    df['Signal'] = df['MACD'].ewm(span=signal_period).mean()

    # Calculate MACD histogram
    df['Histogram'] = df['MACD'] - df['Signal']

# Determine when to buy and sell
#for i in range(long_window, len(df)):
for i in range(len(df)):
    if df['Signal'][i] > df['MACD'][i] and df['Signal'][i-1] <=
df['MACD'][i-1]:
        # Buy
        shares = balance // df['Close'][i]
        cost = shares * df['Close'][i] * (1 + commission)
        holdings += shares
        balance -= cost
        trades.put({'Type': 'Buy', 'Shares': shares, 'Price':
df['Close'][i], 'Balance': balance})

    elif df['Signal'][i] < df['MACD'][i] and df['Signal'][i-1] >=
df['MACD'][i-1]:
        # Sell
        proceeds = holdings * df['Close'][i] * (1 - commission)
        balance += proceeds
        trades.put({'Type': 'Sell', 'Shares': holdings, 'Price':
df['Close'][i], 'Balance': balance})
        holdings = 0
```

```
# Calculate profit/loss using the MACD strategy
macd_profit = balance + (holdings * df['Close'][len(df)-1] * (1 -
commission))


# Compute number of trades and average return per trade
if df['Close'][len(df)-1]:
    trades.task_done()
    #num_trades = len(trades)

    num_trades = trades.qsize()
    macd_return_per_trade = macd_profit / num_trades
    buy_hold_sell_return_per_trade = buy_hold_sell_profit /
num_trades

# Compute relative gain/loss against Buy-Hold-Sell strategy
    relative_gain_loss = (macd_profit / buy_hold_sell_profit) - 1

# Print results
    print("Number of trades made using MACD:", num_trades)
    print("Average return per trade using MACD:",
macd_return_per_trade)
    print("Average return per trade using Buy-Hold-Sell:",
buy_hold_sell_return_per_trade)
    print("Relative gain/loss against Buy-Hold-Sell:",
relative_gain_loss)

#Write records to text file
    with open('records_1.txt', 'w') as f:
     while not trades.empty():
        record = trades.get()
        f.write(f"Type: {record['Type']}, Holdings:
{record['Shares']}")
        f.write('\n')
```