QUESTION 1 import pandas as pd

```
from yahoo_fin import stock_info as si


# Function to compute portfolio weights def
compute_weights(corr_matrix):
num_stocks = len(corr_matrix)
   weights = []


   for i in range(num_stocks):
      if i == 0:
         weights.append(1 / corr_matrix.iloc[i, i]**2)
      else:
         weights.append(1 / corr_matrix.iloc[i, i]**2 / sum(1 / corr_matrix.iloc[:i, :i].sum(axis=1)))


   return weights


# Fetch historical stock data for the S&P 500
def fetching_stock_data():     tickers =
si.tickers_sp500()     start_date = "2012-01-
01"     end_date = "2022-12-31"


   stock_data = pd.DataFrame()


   for ticker in tickers:
      data = si.get_data(ticker, start_date, end_date)
stock_data[ticker] = data["adjclose"]

   return stock_data
```

```
# Initializing an empty DataFrame to accumulate portfolio values portfolio_values
= pd.DataFrame()

# Looping through the 10-year period and perform portfolio management every three months
for year in range(2012, 2023):     for quarter in range(1, 5):
    # Fetching historical stock data for the past 6 months
stock_data = fetch_stock_data()

    # Computing correlation matrix
correlation_matrix = stock_data.corr()

    # Using the function to compute weights
weights = compute_weights(correlation_matrix)

    # Selecting the stocks based on weights        selected_stocks = [stock_data.columns[i] for i in
range(len(stock_data.columns)) if weights[i] > 0

    # Buy and hold for three months (example: assuming equal investment in each stock)
portfolio_values = pd.concat([portfolio_values, stock_data[selected_stocks].iloc[-1]])

# Display final selected stocks, weights, and cumulative portfolio values
print("Selected Stocks:", selected_stocks) print("Portfolio Weights:",
weights) print("Cumulative Portfolio Values:")
print(portfolio_values.sum(axis=1))
```

question 3 import

numpy as np

```
# Function to compute maximum drawdown def
compute_max_drawdown(portfolio_values):
   peak, trough = np.argmax(np.maximum.accumulate(portfolio_values) - portfolio_values),
np.argmax(portfolio_values)      return (portfolio_values[trough] - portfolio_values[peak]) /
portfolio_values[peak]

# Function to compute daily and annual Sharpe ratio def
compute_sharpe_ratio(portfolio_values, risk_free_rate=0):
   daily_returns = portfolio_values.pct_change().dropna()     sharpe_ratio =
(daily_returns.mean() - risk_free_rate) / daily_returns.std()     annual_sharpe_ratio =
sharpe_ratio * np.sqrt(252)  # Assuming 252 trading days in a year     return
annual_sharpe_ratio

# Assuming 'portfolio_values' is the cumulative portfolio values DataFrame
max_drawdown          =          compute_max_drawdown(portfolio_values)
sharpe_ratio = compute_sharpe_ratio(portfolio_values)

# Displaying computed metrics print("Maximum
Drawdown:", max_drawdown) print("Annual
Sharpe Ratio:", sharpe_ratio)
```

question 4

```
# Function to compute ATR (Average True Range) for the portfolio def
compute_portfolio_atr(wealth_process, window=14):     portfolio_returns
= wealth_process.pct_change().dropna()     high_low =
wealth_process.diff().abs().rolling(window=window).mean()     atr =
high_low.rolling(window=window).mean()
   return atr

# Function to apply risk management def
apply_risk_management(portfolio_values, gamma, delta, atr):
    stop_loss = gamma * atr
take_profit = delta * atr


    exit_points = portfolio_values / portfolio_values.shift(1)


    for i in range(len(portfolio_values)):         if
exit_points.iloc[i] < 1 - stop_loss.iloc[i]:
            # Implement stop loss            print(f"Exiting due to stop loss
on {portfolio_values.index[i]}")
            # Take a vacation and wait for the next starting month



        elif exit_points.iloc[i] > 1 + take_profit.iloc[i]:
            # Implement take profit            print(f"Exiting due to take profit
on {portfolio_values.index[i]}")
            # Take a vacation and wait for the next starting month


# Assume you have a DataFrame 'portfolio_values' containing daily portfolio values
gamma, delta = compute_optimal_parameters(historical_data)  # Implement this function based on
your requirements


# Assume 'wealth_process' is the wealth process of 1 GOOG + 2 AAPL portfolio_atr
= compute_portfolio_atr(wealth_process)


apply_risk_management(portfolio_values, gamma, delta, portfolio_atr)

# Display final portfolio and weights with risk management
# ...
# Computing maximum drawdown, Sharpe ratio, etc. max_drawdown_with_risk
= compute_max_drawdown(portfolio_values) sharpe_ratio_with_risk =
compute_sharpe_ratio(portfolio_values)

# Displaying computed metrics with risk management print("Maximum
Drawdown with Risk Management:", max_drawdown_with_risk) print("Sharpe
Ratio with Risk Management:", sharpe_ratio_with_risk)
```