Problem 2.3

In [9]:
```
import pandas as pd
import numpy as np
```

In [10]:
```
df = pd.read_excel('Crude Oil Data-2.xlsx')
df = df.set_index(df['Unnamed: 1'])
df = df.drop(columns = ['Unnamed: 0', 'Unnamed: 1'])
df.index.names = ['Date']
df.head()
```

Out[10]:

	Crude Oil
Date	
2006-02-13	64.01
2006-02-14	63.28
2006-02-15	62.17
2006-02-16	63.04
2006-02-17	63.92

a)

In [11]:
```
two_day_std = np.std(df['Crude Oil'].diff())*np.sqrt(2)
round(two_day_std, 2)
```

Out[11]:
```
2.23
```

The estimate for the two-day standard deviation is $2.23.

b)

This problem is asking about the 99% VaR. To do this, we calculate $-2.33 * \sigma * \sqrt{T}$

In [12]:
```
VaR = -2.33 * 2.23
VaR
```

Out[12]:
```
-5.1959
```

Conclusion: So, in a worst case scenario (1% probability), crude oil may fall $5.19 in two days.

The exchange should set the maintenance margin to $5,190, as the contract size is 1000 barrels.

c)

In [13]:
```
two_day_returns = df['Crude Oil'].diff(periods = 2)
np.sum(two_day_returns < -5.19)
```

Out[13]:
```
24
```

Out of this data, we can see that there are 24 instances of crude oil dropping over $5.19 in two days.

This is inconsistent with our assumption of a 1% chance. The data has 1040 observations and therefore this represents 2.3% rather than 1%.

d)

In [20]:
```
maintenance_margin = 5190
initial_margin = maintenance_margin / 0.75
```

In [21]:
```
contract_values = df['Crude Oil'] * 1000
```

In [25]:
```
# Define the maintenance margin and calculate the initial margin
maintenance_margin = 5190
initial_margin = maintenance_margin / 0.75

# Create a list of contract values (crude oil prices * 1000)
contract_values = [0, 500, -1500, -1000, 300, -500, -2200, -1700, -200, -300]

# Initialize lists to store the table data
initial_margin_list = [initial_margin]
starting_margin_balance_list = [initial_margin]
change_in_contract_value_list = []
balance_after_change_in_value_list = []
deposit_if_margin_call_list = []
withdrawal_if_excess_list = []
true_margin_balance_list = []
default_margin_balance_list = []

# Calculate the table data based on the contract values
for value in contract_values:
    change_in_contract_value_list.append(value)
    balance_after_change_in_value = starting_margin_balance_list[-1] + value
    balance_after_change_in_value_list.append(balance_after_change_in_value)

    if balance_after_change_in_value < maintenance_margin:
        deposit = maintenance_margin - balance_after_change_in_value
        withdrawal = 0
    else:
        deposit = 0
        withdrawal = balance_after_change_in_value - maintenance_margin

    deposit_if_margin_call_list.append(deposit)
    withdrawal_if_excess_list.append(withdrawal)

    true_margin_balance = initial_margin_list[-1] + sum(withdrawal_if_excess_list) - sum(deposit_if_margin_call_list)
    true_margin_balance_list.append(true_margin_balance)

    default_margin_balance = true_margin_balance + value
    default_margin_balance_list.append(default_margin_balance)

    # Update the starting margin balance for the next iteration
    starting_margin_balance_list.append(balance_after_change_in_value)

# Create a DataFrame to display the table
data = {
    'Initial Margin': initial_margin_list,
    'Starting Margin Balance': starting_margin_balance_list[:-1],
    'Change in Contract Value': change_in_contract_value_list,
    'Balance after Change in Value': balance_after_change_in_value_list,
    'Deposit if Margin Call': deposit_if_margin_call_list,
    'Withdrawal if Excess': withdrawal_if_excess_list,
    'True Margin Balance': true_margin_balance_list,
    'Default Margin Balance': default_margin_balance_list,
    'Formula for Default Margin Balance': [''] * len(contract_values)
}

df = pd.DataFrame(data)
np.sum(df['Default Margin Balance'] > 0)
```

Out[25]:
```
13
```

After replicating the chart in Python, we counted 13 defaults.

3.2

b)

In [1]:
```
rf = 0.004
div = 0.015
index_val = 4450
beta = 0.6
index_val_2m = [3500, 4000, 5000, 5500]
```

In [3]:
```
for price in index_val_2m:

    excess_market = (price-index_val)/index_val - rf + div
    excess_port = excess_market * beta
    exp_port = excess_port + rf
    print(f"With the index at {price}, the expected return of the portfolio is {round(exp_port*100, 2)}%")
```

```
With the index at 3500, the expected return of the portfolio is -11.75%
With the index at 4000, the expected return of the portfolio is -5.01%
With the index at 5000, the expected return of the portfolio is 8.48%
With the index at 5500, the expected return of the portfolio is 15.22%
```