Importing All the Required Library

In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

First Importing the Future data from excel sheet. Then Removing all the unwanted Coulmns like 'Aj_Oi' from the dataframe. Converted the 1 min data to 15 minutes because it would take hours to backtest 1 min data for 1 year. Using groupby function , converted intraday data seperately to 15 min and then merged all of them.

In [2]:
```
df = pd.read_csv("BANKNIFTY_2017_FUT.csv")
df = df[df['Contract'] == 'I']
df.drop( columns = { 'OI' , 'Volume', 'Adj_Open' , 'Adj_High' , 'Adj_Low' , 'Adj_Close
df.set_index(df.index , inplace = True)
list = np.arange(0 , len(df))
ind = pd.Index(list)
df = df.set_index(ind)
intra = df.groupby('Date')
j = []
for intr_day in intra :
    timer = 1
    intr_day = intr_day[1]
    df_5 = [ intr_day.iloc[0: 1] ]
    for i in range (1 , len(intr_day)) :
        if timer != 16 :
            timer = timer + 1
        if timer == 16 :
            df_5.append(intr_day.iloc[i:i+1])
            timer = 1

    df_5_min = pd.DataFrame()
    for i in range (len(df_5)) :
        df_5_min = pd.concat(  [df_5_min , df_5[i] ] , axis = 0   )
    j.append(df_5_min)




t =pd.concat( j , ignore_index = True)
df = t
df
```

Out[2]:

	Date	Time	Open	High	Low	Close	Contract	Expiry
0	2017-01-02	09:15:59	18210.05	18261.20	18210.00	18213.85	I	2017-01-25
1	2017-01-02	09:30:59	18108.85	18108.85	18085.00	18085.65	I	2017-01-25
2	2017-01-02	09:45:59	18083.35	18085.50	18067.60	18067.60	I	2017-01-25
3	2017-01-02	10:00:59	18056.00	18065.00	18051.30	18060.10	I	2017-01-25
4	2017-01-02	10:15:59	18060.20	18065.40	18060.20	18063.00	I	2017-01-25
...	...	...	...	...	...	...	...	...
6356	2017-12-29	14:15:59	25553.95	25557.50	25553.50	25553.50	I	2018-01-25
6357	2017-12-29	14:30:59	25548.85	25552.65	25548.85	25552.55	I	2018-01-25
6358	2017-12-29	14:45:59	25555.00	25556.95	25552.70	25552.70	I	2018-01-25
6359	2017-12-29	15:00:59	25538.50	25546.95	25534.50	25546.95	I	2018-01-25
6360	2017-12-29	15:15:59	25559.05	25562.00	25556.40	25562.00	I	2018-01-25

6361 rows × 8 columns

Imported the options data from excel to a dataframe dff. We only required weekly expiry contract so we deleted all the other contracts. Also Delted unwanted columns like 'Adj_Low' etc .

In [3]:
```
dff = pd.read_csv("BANKNIFTY_2017_OPT.csv")
dff = dff[ (  (dff['Contract_Monthly'] == 'W') | (dff['Contract_Monthly'] == 'I') )  &
dff.drop( columns = { 'OI' , 'Volume', 'Adj_Open' , 'Adj_High' , 'Adj_Low' , 'Adj_Clos
dff['Strike'] = dff['Strike'].astype(float).astype(int)
#dff[ (dff['Contract_Monthly'] == 'I')  &  (dff['Contract_Weekly'] == 'I') ].iloc[2000
dff
```

Out[3]:

	Date	Time	Open	High	Low	Close	Type	Contract_Monthly	Contract_Weekly	S
1	2017-01-02	09:15:59	7.00	12.00	5.75	5.80	CE	W	I	1
2	2017-01-02	09:15:59	4.40	6.60	4.15	4.20	CE	W	I	1
3	2017-01-02	09:15:59	3.00	3.00	3.00	3.00	CE	W	I	1
4	2017-01-02	09:15:59	1.50	1.50	1.50	1.50	CE	W	I	1
24	2017-01-02	09:15:59	11.35	13.90	8.75	9.20	CE	W	I	1
...	...	...	...	...	...	...	...	...	...	
5283591	2017-12-29	15:29:59	163.25	164.45	154.10	154.10	PE	W	I	2
5283592	2017-12-29	15:29:59	95.20	99.00	93.35	93.35	CE	W	I	2
5283593	2017-12-29	15:29:59	472.05	472.05	462.00	462.00	PE	W	I	2
5283595	2017-12-29	15:30:59	99.00	99.00	99.00	99.00	CE	W	I	2
5283596	2017-12-29	15:30:59	20.50	20.50	20.50	20.50	CE	W	I	2

2576856 rows × 12 columns

Here the main Backtetsting Occurs . Initialized all the varibales like capital,trade etc and created a cols array with all the information required in the TradeReport. Looped over the future data and found the premium collected when the entry condition is met. After it all the premium is updated as market data moves forward. stop loss and take profit is kept at entry and strikes are choosen atm_strike of price. Options data are collected from the options excel sheet for particular date , time, strike.

As the trade ends either beacuse of expiration or sl or tp level , TradeReport is updated and filled with all information. Option selling margin for atm legs are calculated based on span , exposure margin and hedge otm legs by their premium.

If any data cannot be found in options excel sheet provided , then we try any strike price close to it or any data close to that tick data.

In [4]:
```
import math
trade = 0
pl = 0
strike_multiple = 100
cols = ['Date' , 'Expiry' , 'En_Date' , 'Ce_En_Date' , 'Pe_En_Date' , 'Ce_Ex_Date' ,
capital = 1000000
```

```
trade_report = []
sl = 0
tp = 0
Current_Total_Premium = 0
for i in range (3406 ,3600) :
    if (trade == 0) & (df['Time'].iloc[i] == '10:30:59') :
        atm_strike = strike_multiple * round(df['Close'].iloc[i] / strike_multiple)
        hedge_strike_ce = strike_multiple*round( ( atm_strike + (atm_strike*0.02) ) /
        hedge_strike_pe = strike_multiple*round( ( atm_strike - (atm_strike*0.02) ) /
        #print(df['Date'].iloc[i])
        #print(df['Time'].iloc[i])
        #print(hedge_strike_ce)
        ml = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time'].i
        hl1 = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time'].
        si = 100
        while np.shape(hl1)[0] == 0 :
            hl1 = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time
            si = si + 100
        hl2 = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time'].
        ssii = 100
        while np.shape(hl2)[0] == 0 :
            hl2 = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time
            ssii = ssii + 100

        Total_premium = ml['Close'].iloc[0] + ml['Close'].iloc[1] - hl1['Close'].iloc[
        print(Total_premium)
        #print(hl1['Close'].iloc[0])
        sl = Total_premium +Total_premium*0.3
        tp = Total_premium - Total_premium*0.8
        Date = df['Date'].iloc[i]
        Expiry = ml['Expiry'].iloc[0]
        En_Date = df['Date'].iloc[i]
        Ce_En_Date = df['Date'].iloc[i]
        Pe_En_Date = df['Date'].iloc[i]
        Fut_En_Price = df['Close'].iloc[i]
        Atm_Strike = ml['Strike'].iloc[0]
        Ce_Short_Strike = ml['Strike'].iloc[0]
        Pe_Short_Strike = ml['Strike'].iloc[0]
        Ce_Long_Strike = hl1['Strike'].iloc[0]
        Pe_Long_Strike = hl2['Strike'].iloc[0]
        Ce_Short_En_Price = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] =
        Pe_Short_En_Price = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] =
        Ce_Long_En_Price = hl1['Close'].iloc[0]
        Pe_Long_En_Price = hl2['Close'].iloc[0]
        notional_value = Atm_Strike * 25
        Span_Margin = notional_value * 0.01
        Exposure_Margin = notional_value*0.06
        one_leg_margin = Span_Margin + Exposure_Margin
        one_leg_margin_hedge = one_leg_margin*0.03
        Straddle_Margin = 2*one_leg_margin_hedge
        hedge1 = 25*Ce_Long_En_Price
        hedge2 = 25*Pe_Long_En_Price
        hedge_leg = hedge1 + hedge2
        Quantity = math.floor(capital/ ( Straddle_Margin + hedge_leg) )
        qq = Quantity/ 25
        Quantity = math.floor(qq)*25
        trade = 1
        print(sl)
        print(tp)
```

```
    if trade == 1:
        f =dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time'].iloc
        g = 1
        while (np.shape(f)[0] != 2) :
            f =dff[  (dff['Date'] == df['Date'].iloc[i-g]) & (dff['Time'] == df['Time
            g = g + 1
        ff = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time'].i
        sii = 100
        sii_ = 0
        while np.shape(ff)[0] ==  0 :
            ff = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time
            sii = sii + 100
            sii_ = sii_ + 1
            if sii_ == 30 :
                break
        g = 0
        while np.shape(ff)[0] ==  0 :
            ff = dff[  (dff['Date'] == df['Date'].iloc[i-g]) & (dff['Time'] == df['Tim
            g = g+ 1

        fff = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time'].i
        siii = 100
        siii_ = 0
        while np.shape(fff)[0] ==  0 :
            fff = dff[  (dff['Date'] == df['Date'].iloc[i]) & (dff['Time'] == df['Time
            siii = siii + 100
            siii_ = siii_ + 1
            if siii_ == 30 :
                break
        g = 0
        while np.shape(fff)[0] ==  0 :
            fff = dff[  (dff['Date'] == df['Date'].iloc[i-g]) & (dff['Time'] == df['Ti
            g = g + 1

        Current_Total_Premium = f['Close'].iloc[0] + f['Close'].iloc[1] - ff['Close'].
        print(Current_Total_Premium)

    if (trade == 1)  & (Current_Total_Premium > sl ) :
        Ce_Ex_Date = df['Date'].iloc[i]
        Pe_Ex_Date = df['Date'].iloc[i]
        Ce_Short_Ex_Price = f['Close'].iloc[0]
        Pe_Short_Ex_Price = f['Close'].iloc[1]
        Ce_Long_Ex_Price = ff['Close'].iloc[0]
        pe_Long_Ex_Price = fff['Close'].iloc[0]
        Returns_Abs = Total_premium - Current_Total_Premium
        Total_Abs = Returns_Abs* Quantity
        Lot_Size = 25
        capital = capital + Total_Abs

        #rade_report.appendExpiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_Ex_Date,
        trade_report.append([ Date,Expiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_
        print([ Date,Expiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_Ex_Date,Fut_Er
        trade = 0

    if (trade == 1)  & (Current_Total_Premium <tp) :
        Ce_Ex_Date = df['Date'].iloc[i]
        Pe_Ex_Date = df['Date'].iloc[i]
        Ce_Short_Ex_Price = f['Close'].iloc[0]
        Pe_Short_Ex_Price = f['Close'].iloc[1]
        Ce_Long_Ex_Price = ff['Close'].iloc[0]
```

```
        pe_Long_Ex_Price = fff['Close'].iloc[0]
        Returns_Abs = Total_premium - Current_Total_Premium
        Total_Abs = Returns_Abs* Quantity
        Lot_Size = 25
        capital = capital + Total_Abs

        trade_report.append([ Date,Expiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_
        print([ Date,Expiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_Ex_Date,Fut_Er
        trade = 0

    if  (trade == 1) and  ( df['Date'].iloc[i] == Expiry)  and ( df['Time'].iloc[i] ==
        Ce_Ex_Date = df['Date'].iloc[i]
        Pe_Ex_Date = df['Date'].iloc[i]
        Ce_Short_Ex_Price = f['Close'].iloc[0]
        Pe_Short_Ex_Price = f['Close'].iloc[1]
        Ce_Long_Ex_Price = ff['Close'].iloc[0]
        pe_Long_Ex_Price = fff['Close'].iloc[0]
        Returns_Abs = Total_premium - Current_Total_Premium
        Total_Abs = Returns_Abs*Quantity
        Lot_Size = 25
        capital = capital + Total_Abs

        trade_report.append([ Date,Expiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_
        print([ Date,Expiry,En_Date,Ce_En_Date,Pe_En_Date,Ce_Ex_Date,Pe_Ex_Date,Fut_Er
        trade = 0
Trade_Report = pd.DataFrame(trade_report, columns =cols)
print(Trade_Report)
```

```
236.7
307.71
47.339999999999975
236.7
235.69999999999996
235.85
234.2
237.4
237.85000000000002
244.64999999999998
246.85000000000002
247.85000000000002
253.25
252.65
249.95
244.00000000000003
244.89999999999998
243.19999999999996
245.95
240.55
242.49999999999997
256.35
248.99999999999997
164.99999999999997
243.7
246.55
238.05
242.99999999999997
243.35000000000002
246.14999999999998
252.95000000000005
255.89999999999998
253.45
250.20000000000005
248.4
249.8
247.19999999999996
249.94999999999996
252.50000000000003
253.35000000000002
234.99999999999997
252.89999999999998
251.10000000000002
248.65000000000003
252.7
258.65000000000003
258.09999999999997
262.45
252.60000000000002
226.95
259.70000000000005
280.85
282.74999999999994
304.3
306.1
302.25
303.34999999999997
287.3
288.25
298.25
```

```
297.5
303.4
294.29999999999995
291.6
289.9
290.45000000000005
289.8
309.70000000000005
['2017-07-14', '2017-07-20', '2017-07-14', '2017-07-14', '2017-07-14', '2017-07-18',
'2017-07-18', 23822.0, 23800, 23800, 23800, 24300, 23300, 140.9, 322.0, 122.75, 22.8
5, 11.85, 31.45, 15.1, 3.7, -73.00000000000006, 300, 25, -21900.00000000002, 978100.
0]
154.9
201.37
30.97999999999999
154.9
155.10000000000002
152.75
151.79999999999998
153.15
154.54999999999998
153.20000000000002
149.15
150.45
150.05
148.6
148.64999999999998
149.5
139.15
141.15
135.1
142.5
144.55
153.35
145.35
79.1
137.0
145.39999999999998
140.55
138.05000000000004
135.05
135.3
141.5
148.5
170.95
157.9
152.9
149.1
144.35
109.80000000000001
119.4
113.25
137.14999999999998
141.49999999999997
136.70000000000002
106.25000000000001
111.30000000000001
91.80000000000001
117.60000000000001
121.4
```

```
121.4
276.65
['2017-07-19', '2017-07-20', '2017-07-19', '2017-07-19', '2017-07-19', '2017-07-21',
'2017-07-21', 24146.0, 24100, 24100, 24100, 24600, 23600, 94.95, 232.2, 66.0, 89.45,
1.95, 30.0, 4.1, 15.0, -121.74999999999997, 350, 25, -42612.49999999999, 935487.5]
246.8
320.84000000000003
49.359999999999985
246.8
244.45
247.45
248.2
248.0
247.95
247.75000000000003
244.40000000000003
244.4
247.95
242.60000000000002
241.25
243.64999999999998
233.10000000000002
239.3
236.0
235.25000000000003
234.29999999999998
234.54999999999998
229.99999999999997
188.45
225.30000000000004
240.35
241.95
239.95000000000002
236.54999999999998
244.65
246.99999999999997
250.6
248.85000000000005
242.54999999999998
246.6
244.75
235.0
252.7
255.30000000000004
256.55
250.29999999999998
250.54999999999998
254.85000000000002
258.8
260.3
262.8
258.45
248.89999999999998
251.39999999999998
221.84999999999997
297.09999999999997
305.05
311.9
287.34999999999997
287.35
```

```
284.84999999999997
286.15
294.79999999999995
293.95
294.05
287.8
287.95
288.00000000000006
         Date      Expiry     En_Date  Ce_En_Date  Pe_En_Date  Ce_Ex_Date  \
0  2017-07-14  2017-07-20  2017-07-14  2017-07-14  2017-07-14  2017-07-18
1  2017-07-19  2017-07-20  2017-07-19  2017-07-19  2017-07-19  2017-07-21

   Pe_Ex_Date  Fut_En_Price  Atm_Strike  Ce_Short_Strike  ...  \
0  2017-07-18       23822.0       23800            23800  ...
1  2017-07-21       24146.0       24100            24100  ...

   Pe_Short_Ex_Price  Ce_Long_En_Price  Ce_Long_Ex_Price  Pe_Long_En_Price  \
0              22.85             11.85             31.45              15.1
1              89.45              1.95             30.00               4.1

   Pe_Long_Ex_price  Returns_Abs  Quantity  Lot_Size  Total_Abs  \
0               3.7       -73.00       300        25   -21900.0
1              15.0      -121.75       350        25   -42612.5

   Total_Capital_Returns
0               978100.0
1               935487.5

[2 rows x 26 columns]
```

HERE ONLY FEW TICK DATA BACKTEST RESULT IS SHOWN . WHICH IS SOOME MIDDLE PART. ENTIRE BACKTEST RESULT IS NOT PRINTED AS IT WORLD TAKE LOT OF SPACE . BUT ONCE FULLY BACKTESTED RESULT IS SHOWN IN TRADE REPORT BY STORING AND THEN IMPORTING IT FROM EXCEL SHEET TO DATAFRAME.

Entire TradeReport is printed and Drawdown and Capital percentage returns are calculated.

In [6]:
```
seven_m = pd.read_excel('Trade_Report_7_Month.xlsx')
five_m = pd.read_excel('Trade_Report_3_Month.xlsx')
TradeReport = pd.concat([seven_m , five_m] , ignore_index = True )
TradeReport.drop('Unnamed: 0' , axis = 1,inplace = True)
TradeReport['Total_Pct_Returns'] = ( TradeReport['Total_Abs']/TradeReport['Total_Capit
TradeReport['Total_Pct_Returns'].iloc[0] = (TradeReport['Total_Abs'].iloc[0] / 1000000
TradeReport['Returns'] =( (TradeReport['Total_Capital_Returns'] - TradeReport['Total_C
TradeReport['Returns'].iloc[0] = ( (TradeReport['Total_Capital_Returns'].iloc[0] - 100
TradeReport['Drawdown'] = 0
TradeReport['Max Drawdown'] = 0
for i in range (len(TradeReport)) :
    max =np.max( np.array(TradeReport['Returns'][0 : i+1]) )
    TradeReport['Drawdown'].iloc[i] = TradeReport['Returns'].iloc[i] - max
    TradeReport['Max Drawdown'].iloc[i] = max

#TradeReport = TradeReport.style.set_table_attributes("style='display:inline'").set_ca
TradeReport
```

Out[6]:

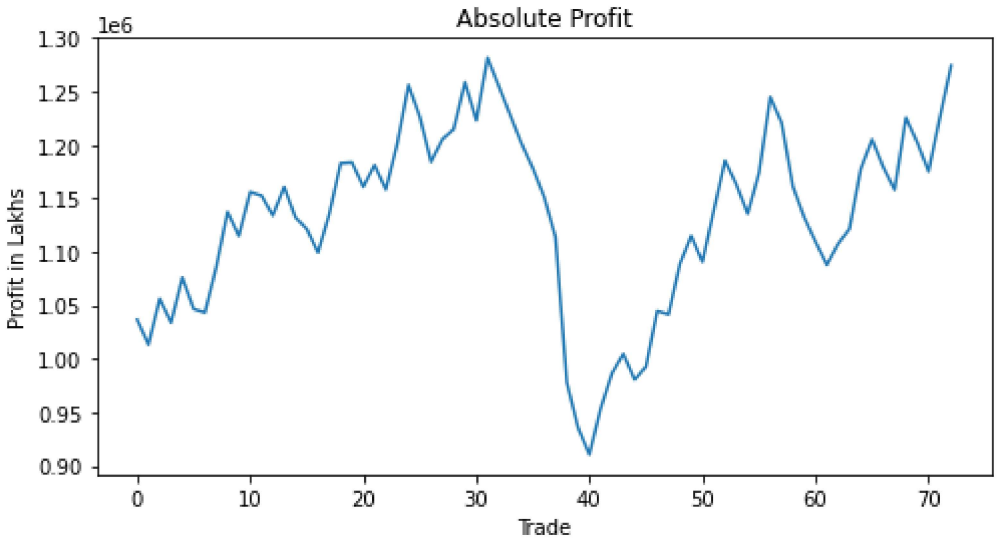
	Date	Expiry	En_Date	Ce_En_Date	Pe_En_Date	Ce_Ex_Date	Pe_Ex_Date	Fut_En_Price	Atm_Strik
0	2017-01-02	2017-01-05	2017-01-02	2017-01-02	2017-01-02	2017-01-05	2017-01-05	18047.25	1800
1	2017-01-06	2017-01-12	2017-01-06	2017-01-06	2017-01-06	2017-01-11	2017-01-11	18333.45	1830
2	2017-01-12	2017-01-12	2017-01-12	2017-01-12	2017-01-12	2017-01-12	2017-01-12	18860.55	1890
3	2017-01-13	2017-01-19	2017-01-13	2017-01-13	2017-01-13	2017-01-16	2017-01-16	18832.90	1880
4	2017-01-17	2017-01-19	2017-01-17	2017-01-17	2017-01-17	2017-01-19	2017-01-19	19154.90	1920
...	...	...	...	...	...	...	...	...	.
68	2017-12-05	2017-12-07	2017-12-05	2017-12-05	2017-12-05	2017-12-07	2017-12-07	25094.60	2510
69	2017-12-08	2017-12-14	2017-12-08	2017-12-08	2017-12-08	2017-12-14	2017-12-14	25320.00	2530
70	2017-12-15	2017-12-21	2017-12-15	2017-12-15	2017-12-15	2017-12-18	2017-12-18	25500.35	2550
71	2017-12-18	2017-12-21	2017-12-18	2017-12-18	2017-12-18	2017-12-21	2017-12-21	25643.95	2560
72	2017-12-22	2017-12-28	2017-12-22	2017-12-22	2017-12-22	2017-12-28	2017-12-28	25642.20	2560

73 rows × 30 columns

Graph of How Capital Is grown is shown

In [7]:
```
TradeReport['Total_Capital_Returns'].plot(figsize = (8,4))
plt.title("Absolute Profit")
plt.xlabel('Trade')
plt.ylabel('Profit in Lakhs')
```

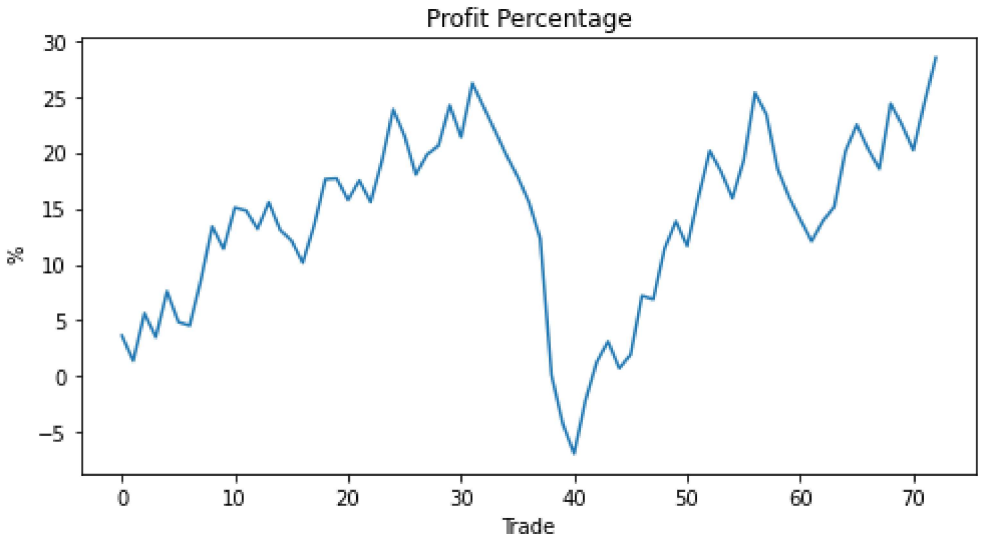
Out[7]:
```
Text(0, 0.5, 'Profit in Lakhs')
```



Graph of cumulative returns of strategy is plotted vs trades .

In [8]:
```
TradeReport['Returns'].cumsum().plot(figsize = (8,4))
plt.title("Profit Percentage")
plt.xlabel('Trade')
plt.ylabel(' %')
```

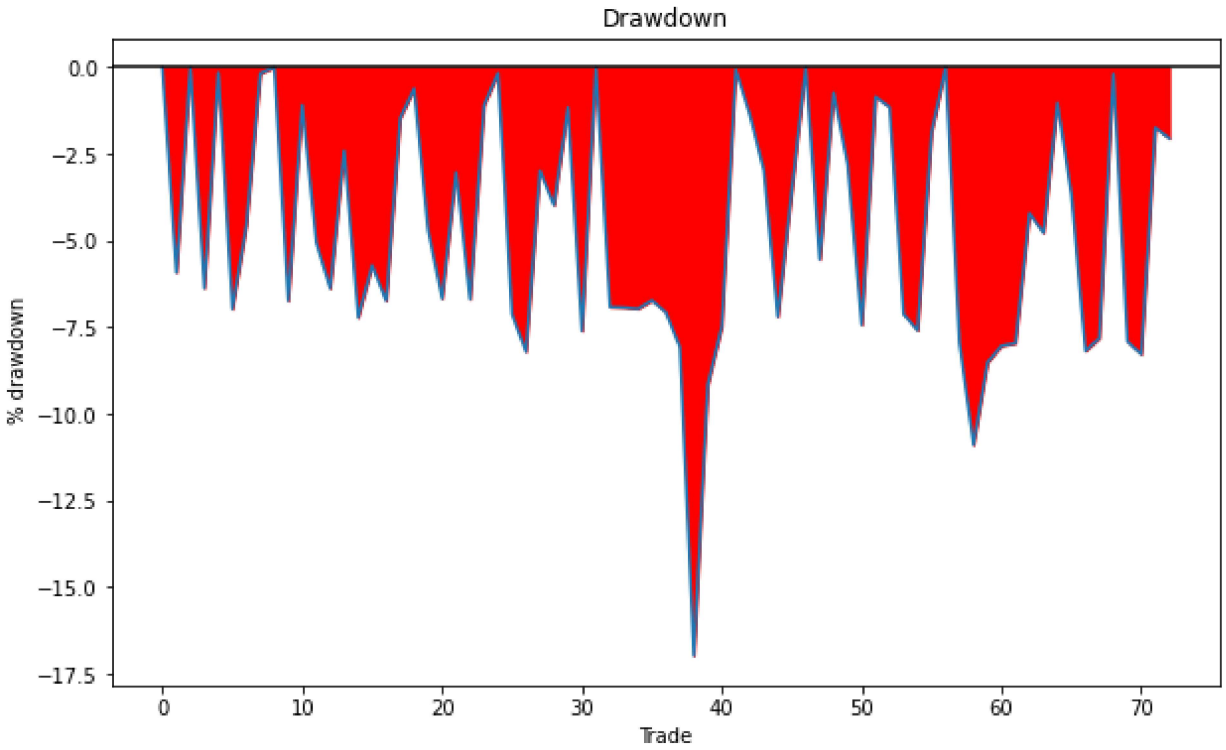
Out[8]:
```
Text(0, 0.5, ' %')
```



Drawdown of startegy in % is plotted to show the risk of the strategy.

In [9]:
```
TradeReport['Drawdown'].plot(figsize = (10,6))
plt.axhline(0 , label ='Line3', color ='black')
plt.fill_between(TradeReport.index, TradeReport['Drawdown'], color='red')
plt.title("Drawdown")
plt.xlabel('Trade')
plt.ylabel(' % drawdown')
```

Out[9]:
```
Text(0, 0.5, ' % drawdown')
```



Sample Performace Of the Entire Strategy is listed and updated

In [10]:
```
stats =pd.read_csv('SamplePerformanceStats.csv')
stats['Percentage Return'].iloc[1] = len(TradeReport)
stats['Percentage Return'].iloc[2] = len(TradeReport[TradeReport['Returns'] >= 0])
stats['Percentage Return'].iloc[3] = len(TradeReport[TradeReport['Returns']<= 0])
stats['Percentage Return'].iloc[4] = (stats['Percentage Return'].iloc[2] /stats['Perce
stats['Percentage Return'].iloc[5] =  TradeReport[TradeReport['Returns'] >= 0]['Return
stats['Percentage Return'].iloc[6] =  TradeReport['Returns'].cumsum()[len(TradeReport)
stats['Percentage Return'].iloc[7] = TradeReport['Returns'].cumsum()[len(TradeReport)-
stats['Percentage Return'].iloc[8] = TradeReport[TradeReport['Returns'] >= 0]['Returns
stats['Percentage Return'].iloc[9] = TradeReport[TradeReport['Returns'] <= 0]['Returns
stats['Percentage Return'].iloc[10] = TradeReport[TradeReport['Returns'] >= 0]['Return
stats['Percentage Return'].iloc[11] = TradeReport[TradeReport['Returns'] <= 0]['Return
stats['Percentage Return'].iloc[12] =  TradeReport['Returns'].cumsum()[len(TradeReport
stats['Percentage Return'].iloc[13]  =  TradeReport['Returns'].cumsum()[len(TradeRepor
stats['Percentage Return'].iloc[14]  =   TradeReport['Returns'].cumsum()[len(TradeRepo
stats['Percentage Return'].iloc[15]  =   TradeReport['Returns'].cumsum()[len(TradeRepo
stats['Percentage Return'].iloc[16]  =    np.min( np.array( TradeReport['Drawdown'] ) )
stats['Absolute Return'].iloc[7] = (TradeReport['Total_Capital_Returns'].iloc[len(Trad
stats['Absolute Return'].iloc[8] =  TradeReport[TradeReport['Returns'] >= 0]['Total_Ab
stats['Absolute Return'].iloc[9] = TradeReport[TradeReport['Returns'] <= 0]['Total_Abs
stats['Absolute Return'].iloc[10] = TradeReport[TradeReport['Returns'] >= 0]['Total_Ab
stats['Absolute Return'].iloc[11] = TradeReport[TradeReport['Returns'] <= 0]['Total_Ab
stats['Absolute Return'].iloc[12] = TradeReport['Total_Abs'].cumsum()[len(TradeReport)
stats['Absolute Return'].iloc[13] = TradeReport['Total_Abs'].cumsum()[len(TradeReport)
stats['Absolute Return'].iloc[14] = TradeReport['Total_Abs'].cumsum()[len(TradeReport)
stats['Absolute Return'].iloc[15] = TradeReport['Total_Abs'].cumsum()[len(TradeReport)
stats['Absolute Return'].iloc[16] =  np.min( np.array( TradeReport['Max Drawdown'] ) )
stats
```

Out[10]:

	Stats	Percentage Return	Absolute Return
0	Capital	1000000	NaN
1	Total Trades	73	NaN
2	Profitable Trades	35	NaN
3	Losing Trades	38	NaN
4	Hit Ratio	47.945205	NaN
5	Risk Reward	1.41922	NaN
6	Calmar ratio	1.677104	NaN
7	Avg Return Per Trade	0.390047	3756.198630
8	Avg Profit Per Trade	3.461934	38225.892857
9	Avg Loss Per Trade	-2.439322	-24999.342105
10	Max Profit Per Trade	6.020855	70682.500000
11	Max Loss Per Trade	-12.177898	-59510.000000
12	Avg Return Per Month	2.372787	32327.604167
13	Avg Return Per Year	28.47344	387931.250000
14	Total Return	28.47344	387931.250000
15	CAGR Return	28.47344	387931.250000
16	Max Drawdown	-16.977747	3.659500

In [ ]: