Experiment 2b:

Aim: To fit a straight line for force vs displacement data for a spring loaded with masses and find the value of spring constant.

Software packages used: language- Python3, packages- Pandas, Numpy, Matplotlib

Theory: The least square fit method minimizes the sum of the squares of the errors or deviations between the observed values and the fitted values.

The formula for the line of best fit is $y = ax + b$, where a is the slope and b is the y-intercept. In this case, the slope can be found out using the formula:

$$a = \frac{(n\Sigma x_i y_i) - (\Sigma x_i)(\Sigma y_i)}{(n\Sigma x_i^2) - (\Sigma x_i)^2}$$

The errror associated with the slope is

$$a_{err} = \sqrt{\frac{\Sigma(y_i - ax_i - b)^2}{n-2}} \times \sqrt{\frac{n}{(n\Sigma x_i^2) - (\Sigma x_i)^2}}$$

The y-intercept b can be found out using:

$$b = \frac{(\Sigma x_i^2)(\Sigma y_i) - (\Sigma x_i)(\Sigma x_i y_i)}{(n\Sigma x_i^2) - (\Sigma x_i)^2}$$

The error associated with the y-intecept is

$$b_{err} = \sqrt{\frac{\Sigma(y_i - ax_i - b)^2}{n-2}} \times \sqrt{\frac{\Sigma x_i^2}{(n\Sigma x_i^2) - (\Sigma x_i)^2}}$$

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

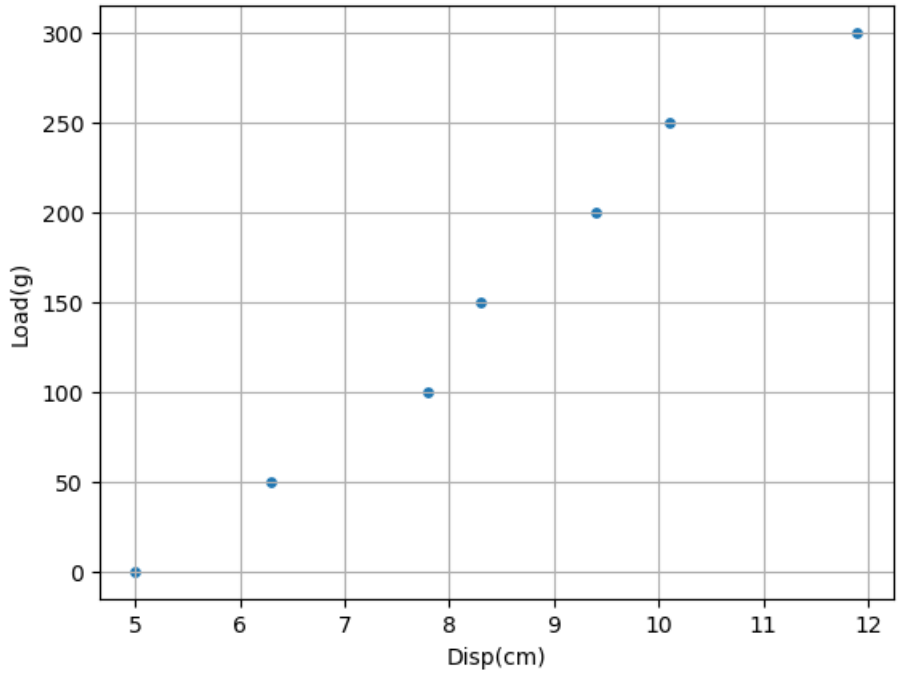
The data for load vs displacemnt is stored in the file 'load_vs_disp.csv'

In [30]:
```
df = pd.read_csv('load_vs_disp.csv')
```

In [31]:
```
display(df)
```

	Load(g)	Disp(cm)
0	0	5.0
1	50	6.3
2	100	7.8
3	150	8.3
4	200	9.4
5	250	10.1
6	300	11.9

In [34]:
```
df.plot(kind='scatter',y = 'Load(g)', x= 'Disp(cm)', s = 15, grid = True)
plt.show()
```



In [73]:
```
n=len(df['Load(g)'])
x_i = df['Disp(cm)']
y_i = df['Load(g)']

# slope calcualation
a = (n*((x_i*y_i).sum()) -((x_i.sum())*(y_i.sum()))  )/((n*((x_i)**2).sum()) -((x_i.sum())**2) )

# y-intercept calculation
b = (((x_i**2).sum())*(y_i.sum()) -  (x_i.sum())*((x_i*y_i).sum())) /((n*((x_i)**2).sum()) -((x_i.s

print("slope, a =", a )
print("y-intercept, b =",b)
```

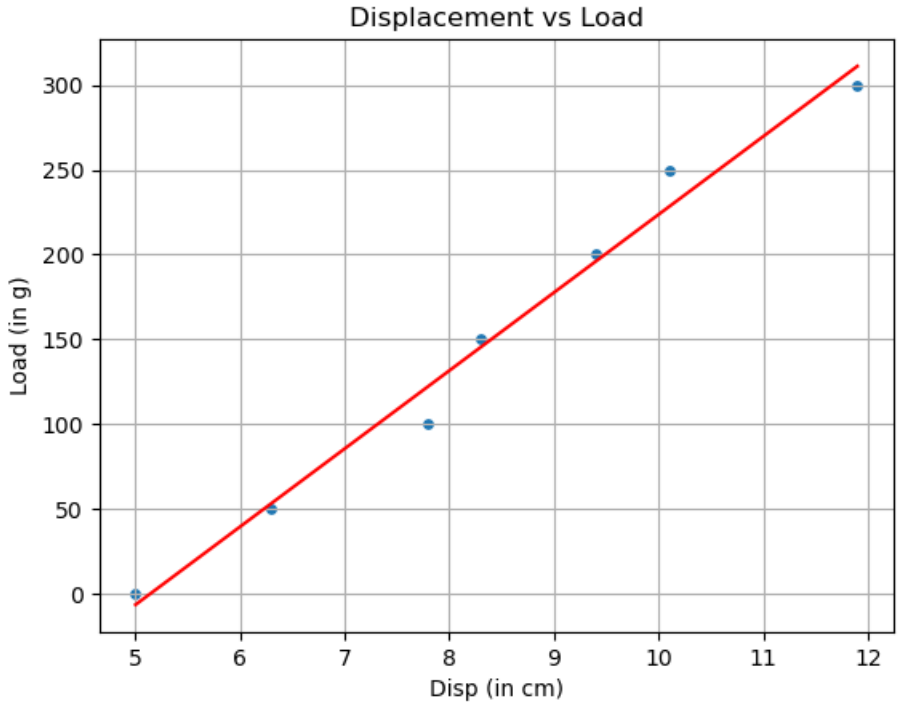
```
slope, a = 46.028325123152655
y-intercept, b = -236.6379310344821
```

In [76]:
```
# slope error-calculation
a_error = ((((y_i - a*x_i -b)**2).sum()/(n-2))**0.5)*((n)/((n*((x_i)**2).sum()) -((x_i.sum())**2))

# y-intercept error-calculation
b_error = ((((y_i - a*x_i -b)**2).sum()/(n-2))**0.5)*((((x_i**2).sum())/((n*((x_i)**2).sum()) -((x_

print("slope error, a_error =", a_error )
print("y-intercept error, b_error =",b_error)
```

```
slope error, a_error = 2.704281795614227
y-intercept error, b_error = 23.450972906530044
```

In [77]:
```
# generating points for straight line
# y = ax + b

y = a*df['Disp(cm)'] + b
```

In [78]:
```
# plotting data points and straight line

plt.plot(df["Disp(cm)"], y, color = 'r')
plt.scatter(df['Disp(cm)'], df["Load(g)"], s = 15)

plt.title("Displacement vs Load")
plt.xlabel("Disp (in cm)")
plt.ylabel("Load (in g)")
plt.grid()

plt.show()
```



In [81]:
```
#calcualtion of spring constant

k = a * ((9.81 * 0.001)/(0.01))

k_error = a_error * ((9.81 * 0.001)/(0.01))
```

In [82]:
```
print("spring constant = %0.2f" %(k), u"\u00B1", "%0.2f" %(k_error) )
```

```
spring constant = 45.15 ± 2.65
```