```
#Importing libraries
import pandas as pd
from pandas import DataFrame
import numpy as np
import tkinter as tk
from tkinter import *
from tkinter import filedialog
import openpyxl
from openpyxl.styles import PatternFill
from openpyxl.styles.borders import Border, Side
from openpyxl.chart import (
 PieChart,
 ProjectedPieChart,
 Reference,
 BarChart)
from openpyxl import Workbook
from openpyxl.utils.dataframe import dataframe_to_rows
from openpyxl.chart.series import DataPoint
from openpyxl.chart.label import DataLabelList
import os
filename_=[] #file inputted in UploadFile function
#Line 14 to 110 is the Tkinter module and file-input-program to get the inspection files
def UploadFile(event=None): #Function to upload file
    filename = filedialog.askopenfilename()
    filename_.append(filename)

if os.path.isfile('ntnu1.ico'):
 logo = os.getcwd()
 logo1 = logo + '/ntnu1.ico'
else:
 logo1 = ''

#Function with button that outputs information about the Excelfile template. The E, W, S and Y function outputs information about the values from ASME B31.3 that is used in the formulas.
def Excelinfo():
 win = tk.Toplevel()
 win.wm_title("Excel format")
 win.iconbitmap(logo1)
 l = tk.Label(win, text="The format of the Inspection Excel file should be with the following column-grid format. This is to ensure that there are no Errors due to wrong formatting. \n COLUMNS SHOULD BE: \n Line No./TAG No. Insp. TAG No. Clock pos. Diameter Nom. WT Corr. Allowance Des. pressure (bar) \n\n Op. temp. (°C) Material Spec. No. Inspection year Min. WT year Uncorroded WT year")
 l.grid(row=0, column=0, padx=5, pady=5)

 b = tk.Button(win, text="Close", command=win.destroy)
 b.grid(row=1, column=0)
def E():
    win = tk.Toplevel()
    win.wm_title("Quality factor")
    win.iconbitmap(logo1)
    l = tk.Label(win, text="The E-value is the quality factor, this factor is found in Table A-1A or Table A-1B in ASME B31.3 \n This program has data on the quality factor from the following specs: \n\n A106B A47 A48 A126 A197 A278 A426 B26 API5L A53 A106 \n\n A179 A333 A334 A369 A381 A524 A671 A672 A691 A268 A269 \n\n A270 A312 A358 A376 A789 A790 A982 B42 B43 B68 B75 \n\n B88 B280 B466 B161 B163 B165 B167 B407 B444 B474 B622 \n\n B668 B690 B729 B804 B338 B861 B862 B523 B658 B210 B241 \n\n B547 A395 A536 A571 A216 A352 A217 A351 A487 B61 B62 \n\n B148 B584 A494 B367 A134 A139 A249 B464 B514 B515 B675 \n\n B704 B705 A135 A587 A409 A813 A814 B467 B619 B626 B725 \n\n A451")
    l.grid(row=0, column=0, padx=5, pady=5)

    b = tk.Button(win, text="Close", command=win.destroy)
    b.grid(row=1, column=0)
def W():
    win = tk.Toplevel()
    win.wm_title("Weld joint strength reduction factor")
    win.iconbitmap(logo1)

    l = tk.Label(win, text="The W-value is the weld joint strength reduction factor in accordance with para. 302.3.5(e) in ASME B31.3 \n This value is set by two parametres, temperature and steel group. \n The steel groups are the specification listed in the Material-column in the Excel-sheet. \n The following groups are included in this program: \n Carbon Steel, CrMo, CSEF (N + T), CSEF. Austenitic welds in austenitic stainless grade 3xx, and N088xx and N066xx nickel alloys. \n\n If the material is NOT in this list, the W-value is automatically set to 1, be sure to check if this is correct before proceding with the inspection results.")
    l.grid(row=0, column=0, padx=5, pady=5)

    b = tk.Button(win, text="Close", command=win.destroy)
    b.grid(row=1, column=0)
def Y():
    win = tk.Toplevel()
    win.wm_title("Y-value")
    win.iconbitmap(logo1)

    l = tk.Label(win, text="The Y-value is the coefficient from Table 304.1.1 in ASME B31.3. This value is valid for t<D/6 and for materials shown. \n For t >= D/6, the Y value is calculated by the following formula: \n Y = (d+2c)/(D+d+2c)")
    l.grid(row=0, column=0)

    b = tk.Button(win, text="Close", command=win.destroy)
    b.grid(row=1, column=0)
def S():
    win = tk.Toplevel()
    win.wm_title("Stress value")
    win.iconbitmap(logo1)

    l = tk.Label(win, text="The S-value is the stress value for materials from Table A-1 or Table A-1M in ASME B31.3. \n This program has limited its S-value data to temperatures ranging from 0-400 Fahrenheit, and to the following material specs: \n\n A106B A47 A48 A126 A197 A278 A426 B26 API5L A53 A106 \n\n A179 A333 A334 A369 A381 A524 A671 A672 A691 A268 A269 \n\n A270 A312 A358 A376 A789 A790 A982 B42 B43 B68 B75 \n\n B88 B280 B466 B161 B163 B165 B167 B407 B444 B474 B622 \n\n B668 B690 B729 B804 B338 B861 B862 B523 B658 B210 B241 \n\n B547 A395 A536 A571 A216 A352 A217 A351 A487 B61 B62 \n\n B148 B584 A494 B367 A134 A139 A249 B464 B514 B515 B675 \n\n B704 B705 A135 A587 A409 A813 A814 B467 B619 B626 B725 \n\n A451")
    l.grid(row=0, column=0)

    b = tk.Button(win, text="Close", command=win.destroy)
    b.grid(row=1, column=0)


root = tk.Tk()
greet = tk.Label(text="Input Inspection-file:")
greet.grid(row=0, column=1)

button_E = tk.Button(root, text="E-Coefficient", command=E)
button_E.grid(row = 5, column = 1, padx=5, pady=5)
button_Y = tk.Button(root, text="Y-Coefficient", command=Y)
button_Y.grid(row = 6, column = 1, padx=5, pady=5)
button_W = tk.Button(root, text="W-Coefficient", command=W)
button_W.grid(row = 7, column = 1, padx=5, pady=5)
```

```
button_S = tk.Button(root, text="S-Coefficient", command=S)
button_S.grid(row = 8, column = 1, padx=5, pady=5)
Button_Ex = tk.Button(root, text="Excel formatting", command=Excelinfo)
Button_Ex.grid(row=9, column=1, padx=5, pady=5)

e = Entry(root, width = 50)
e.grid(row = 3, column = 1)

Excelfile_=[] #Append Excelfile here to save the filename for later.

def Click(): #Button that runs the program after file and inspection filename is inputed.
 Excelfile_.append(e.get())
 root.destroy()

greet1 = tk.Label(text="Filename result-file:")
greet1.grid(row=2, column=1)

button = tk.Button(root, text='Open', command=UploadFile)
button.grid(row=1,column=1)

button1 = tk.Button(root, text='Apply', command=Click)
button1.grid(row=3,column=2)

root.iconbitmap(logo1)
root.title('Inspection - Dataset')

info = tk.Label(text="READ THIS FIRST TIME USING THE PROGRAM \n \n  This program has some limitations when it comes to materials and temperatures. \n All limitations are provided in the different button windows underneath. \n This program is based of ASME B31.3, we recommend that you have a basic  \n knowledge of this standard")
info.grid(row=4, column=1)

root.mainloop()


#Define the input file as a dataframe
df = pd.read_excel(str(filename_[0]), na_values=['-', 'OK'])

#Function that sets the Weldfactor for all rows in the inspection document
#This is a digitalized way to set the values accordingly to material spec and temperature, this is from the ASME B31.3 table
#The Weldfactor uses the Temperature and Material columns to see what the factor should be.
#Since mostpart of the values in the W-diagram in ASME are 1, we set the value as 1, then change it if necessary. This saves us alot of extra code.
#Is takes all arguments for temperature = Ambigous and sets it to roomtemperature (20 degrees)
def Weldfactor():
 df_weld=df[['Op. temp. (°C)', 'Material']]
 df_weld1 = df_weld.replace(['amb','Amb'], 20)
 df_weld1['Op. temp. (°C)'].astype('float') #Sets the temperature as float datatype
 df_weld1['Material'].astype('|S') #Sets the material as string datatype
 df_weld1 = df_weld1.assign(W=1)
 df_weld1['W'].astype('float')

 for index, row in df_weld1.iterrows():
  if (427<row['Op. temp. (°C)']<454) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.95
  elif (454<row['Op. temp. (°C)']<482) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.91
  elif (482<row['Op. temp. (°C)']<510) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.86
  elif (510<row['Op. temp. (°C)']<538) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.82
  elif (538<row['Op. temp. (°C)']<566) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.77
  elif (566<row['Op. temp. (°C)']<593) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.73
  elif (593<row['Op. temp. (°C)']<621) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.68
  elif (621<row['Op. temp. (°C)']) & (row['Material']=='CrMo'):
   df_weld1.loc[index, ['W']] = 0.64

  elif (510<row['Op. temp. (°C)']<538) & (row['Material']=='CSEF (N + T)'):
   df_weld1.loc[index, ['W']] = 0.95
  elif (538<row['Op. temp. (°C)']<566) & (row['Material']=='CSEF (N + T)'):
   df_weld1.loc[index, ['W']] = 0.91
  elif (566<row['Op. temp. (°C)']<593) & (row['Material']=='CSEF (N + T)'):
   df_weld1.loc[index, ['W']] = 0.86
  elif (593<row['Op. temp. (°C)']<621) & (row['Material']=='CSEF (N + T)'):
   df_weld1.loc[index, ['W']] = 0.82
  elif (621<row['Op. temp. (°C)']) & (row['Material']=='CSEF (N + T)'):
   df_weld1.loc[index, ['W']] = 0.77

  elif (482<row['Op. temp. (°C)']) & (row['Material']=='CSEF'):
   df_weld1.loc[index, ['W']] = 0.5

  elif (510<row['Op. temp. (°C)']<538) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.95
  elif (538<row['Op. temp. (°C)']<566) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.91
  elif (566<row['Op. temp. (°C)']<593) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.86
  elif (593<row['Op. temp. (°C)']<621) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.82
  elif (621<row['Op. temp. (°C)']<649) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.77
  elif (649<row['Op. temp. (°C)']<677) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.73
  elif (677<row['Op. temp. (°C)']<704) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.68
  elif (704<row['Op. temp. (°C)']<732) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.64
  elif (732<row['Op. temp. (°C)']<760) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.59
  elif (760<row['Op. temp. (°C)']<788) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.55
  elif (788<row['Op. temp. (°C)']) & (row['Material']=='Austenitic stainless grade 3xx'):
   df_weld1.loc[index, ['W']] = 0.5

  elif row['Material']=='Other Materials':
   print("IF NO W-VALUE EXIST, SET W=1") #Prints it in the python terminal, not in the software. Be sure to doublecheck the terminal output if "Other Materials" is the case.
```

```
 df['W']=df_weld1['W']
Weldfactor()

#Function that sets the E-coefficient to every row.
def E_coeff():
 list_1 = ['A106 B','A47', 'A48', 'A126', 'A197', 'A278', 'A426', 'B26', 'API 5L','A53', 'A106', 'A179', 'A333', 'A334', 'A369', 'A381', 'A524', 'A671', 'A672', 'A691', 'A268', 'A269', 'A270', 'A312', 'A358', 'A376', 'A789',
'A790', 'A982', 'B42', 'B43', 'B68', 'B75', 'B88', 'B280', 'B466', 'B161', 'B163', 'B165', 'B167', 'B407', 'B444', 'B474', 'B622', 'B668', 'B690', 'B729', 'B804', 'B338', 'B861', 'B862', 'B523', 'B658', 'B210', 'B241',
'B547']
 list_080 = ['A395', 'A536', 'A571', 'A216', 'A352', 'A217', 'A351', 'A487', 'B61', 'B62', 'B148', 'B584', 'A494', 'B367', 'A134', 'A139', 'A249', 'B464', 'B514', 'B515', 'B675', 'B704', 'B705']
 list_085 = ['A135','A587','A409','A813','A814','B467','B619','B626','B725']
 list_090 = ['A451']

 df_E = df
 df_E1=df.assign(E=1)

 for index, row in df_E1.iterrows():
  if row['Spec. No.'] in list_1:
   df_E1.loc[index, 'E'] = 1
  elif row['Spec. No.'] in list_090:
   df_E1.loc[index, 'E'] = 0.9
  elif row['Spec. No.'] in list_085:
   df_E1.loc[index, 'E'] = 0.85
  elif row['Spec. No.'] in list_080:
   df_E1.loc[index, 'E'] = 0.80
  else:
   print('Material at row: '+ str(index+2)+ ' in Excelfile not in list')
   df_E1.loc[index, 'E'] = "ERROR MATERIAL NOT IN LIST"

 df['E'] = df_E1['E']
E_coeff()

def S_value():
 dfwitht=df.assign(S=1)
 dfwitht = dfwitht.replace(['amb','Amb'], 20)

 A106B = [20,20,20,19.9]
 A47 = [10,10,10,10]
 A48 = [6,6,6,6]
 A126 = [4,4,4,4]
 A197 = [8,8,8,8]
 A278 = [6,6,6,6]
 A426 = [30,30,30,30]
 B26 = [4.7,4.7,4.7,3.5]
 API5L = [30,30,30,30]
 A53 = [20,20,20,19.9]
 A106 = [23.3,23.3,23.3,22.8]
 A179 = [8,8,8,8]
 A333 = [21.7,21.7,20.6,19.9]
 A334 = [21.7,21.4,20.6,19.9]
 A369 = [20,18.7,18.2,18]
 A381 = [25,25,25,25]
 A524 = [20,20,20,19.9]
 A671 = [25,25,24.8,22.8]
 A672 = [25,25,25,25]
 A691 = [28.3,28.3,28.3,28.2]
 A268 = [23.3,23.3,23.3,23.3]
 A269 = [20,20,20,19.3]
 A270 = [20,20,20,19.3]
 A312 = [30,30,30,29.6]
 A358 = [20,20,20,20]
 A376 = [20,20,20,20]
 A789 = [38.7,38.6,36.8,35.6]
 A790 = [38.7,38.6,36.8,35.6]
 A928 = [38.7,38.5,36.4,35.1]
 B42 = [15,13.6,12.6,4.3]
 B43 = [8,7.9,7.9,5]
 B68 = [6,4.9,4.7,3]
 B75 = [15,13.6,12.6,4.3]
 B88 = [12,10.9,10,9.5]
 B280 = [6,4.9,4.7,3]
 B466 =[12,11.6,11.3,11]
 B161 =[21.7,21.7,21.6,21.6]
 B163 =[26.7,26.7,26.7,26.7]
 B165 =[28.3,28.3,28.3,28.3]
 B167 =[23.3,23.3,23.3,23.3]
 B407 =[26.7,26.7,26.7,26.7]
 B444 =[40,40,39.6,39.2]
 B474 =[23.3,23.3,23.3,23.3]
 B622 =[36.3,30.9,28.1,26.1]
 B668 =[20.7,20.7,20.7,20.7]
 B690 =[30,30,30,20.6]
 B729 =[23.3,23.3,23.3,23.3]
 B804 =[30,30,29.9,28.6]
 B338 =[23.3,21.8,18.9,16.7]
 B861 =[23.3,21.8,18.9,16.7]
 B862 =[23.3,21.8,18.9,16.7]
 B523 =[26.7,22.1,18.9,16.7]
 B658 =[26.7,22.1,18.9,16.7]
 B210 =[14,14,11.7,5.2]
 B241 =[12.7,12.7,10.5,5.2]
 B547 =[12,12,12,12]
 A395 =[20,19,17.9,16.9]
 A536 =[21.7,21.7,21.7,21.7]
 A571 =[20,20,20,20]
 A216 =[23.3,23.3,23.3,22.8]
 A352 =[23.3,23.3,23.3,22.8]
 A217 =[30,29.9,29.1,28.8]
 A351 =[26.7,26.7,26.7,26.7]
 A487 =[36.7,36.7,35.9,35.3]
 B61 =[10.7,9.5,9.2,8.6]
 B62 =[9.3,9.2,8.1,7.4]
 B148 =[26.7,26.7,26.7,26.7]
 B584 =[36.7,36.7,36.7,36.7]
 A494 =[26.7,26.7,26.7,26.7]
 B367 =[16.7,13.8,11.4,9.5]
 A134 =[21.7,21.7,21.7,21.7]
```

```
 A139 = [22,22,22,22]
 A249 = [30,30,29.5,27.3]
 B464 = [23.3,23.3,23.3,23.3]
 B514 = [26.7,26.7,26.7,26.7]
 B515 = [26.7,26.7,26.7,26.7]
 B675 = [30,30,30,29.6]
 B704 = [23.3,23.3,23.3,23.3]
 B705 = [23.3,23.3,23.3,23.3]
 A135 = [20,20,20,19.9]
 A587 = [16,16,16,16]
 A409 = [20,20,20,20]
 A813 = [30,30,30,29.6]
 A814 = [30,30,30,29.6]
 B467 = [13.3,12.6,12,11.5]
 B619 = [36.3,30.9,28.1,26.1]
 B626 = [36.3,30.9,28.1,26.1]
 B725 = [28.3,28.3,28.3,28.3]
 A451 = [26.7,26.7,26.7,26.7]

 list_all_S =
[A106B,A47,A48,A126,A197,A278,A426,B26,API5L,A53,A106,A179,A333,A334,A369,A381,A524,A671,A672,A691,A268,A269,A270,A312,A358,A376,A789,A790,A928,B42,B43,B68,B75,B88,B280,B466,B161

 list_all_Ss=['A106B','A47','A48','A126','A197','A278','A426','B26','API5L','A53','A106','A179','A333','A334','A369','A381','A524','A671','A672','A691','A268','A269','A270','A312','A358', 'A376', 'A789', 'A790',
'A982', 'B42', 'B43', 'B68', 'B75', 'B88', 'B280', 'B466', 'B161', 'B163', 'B165', 'B167', 'B407', 'B444', 'B474', 'B622', 'B668', 'B690', 'B729', 'B804', 'B338', 'B861', 'B862', 'B523', 'B658', 'B210', 'B241',
'B547','A395', 'A536', 'A571', 'A216', 'A352', 'A217', 'A351', 'A487', 'B61', 'B62', 'B148', 'B584', 'A494', 'B367', 'A134', 'A139', 'A249', 'B464', 'B514', 'B515', 'B675', 'B704',
'B705','A135','A587','A409','A813','A814','B467','B619','B626','B725','A451']

 specs=[]

 dfwitht['Spec. No.'] = dfwitht['Spec. No.'].str.replace(" ","")

 for index, row in dfwitht.iterrows():
  if row['Spec. No.'] in list_all_Ss:
   specs.append(list_all_Ss.index(row['Spec. No.']))
  else:
   print("No data for this spec")

 for spec in specs:
  for index, row in dfwitht.iterrows():
   if row['Op. temp. (°C)']<37.777778:
    dfwitht.loc[index, ['S']] = list_all_S[spec][0]
   elif 37.777778<row['Op. temp. (°C)']<93.33333:
    dfwitht.loc[index, ['S']] = list_all_S[spec][1]
   elif 93.33333<row['Op. temp. (°C)']<148.88889:
    dfwitht.loc[index, ['S']] = list_all_S[spec][2]
   elif 148.88889<row['Op. temp. (°C)']<204.44444:
    dfwitht.loc[index, ['S']] = list_all_S[spec][3]
   elif row['Op. temp. (°C)']>204.44444:
    print("No data for temperatures over 400 farenheit")
  break
 df['S']=dfwitht['S']*6.89475908677537 #Converting from KSI to MPa
S_value()

#Function that decides the Y-coefficient for the values in the inspection-document
def Ycoeff():
 t_val=[]
 df_y1=df[['Material','Diameter', 'Nom. WT','Corr. Allowance','Des. pressure (bar)', 'Op. temp. (°C)','W', 'S', 'E']]
 df_y1 = df_y1.replace(['amb','Amb'], 20)
 for index, row in df_y1.iterrows():
  t_val.append((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*0.4))

 t_df=DataFrame(t_val, columns=['t'])
 df_yval=pd.concat([df_y1, t_df], axis=1)
 df_yval1=df_yval.assign(Y=0.4) #Sets the Y-values to 0.4, since the mostpart of the Y-values in ASME B31.3 is 0.4, saves us alot of code.

 for index, row in df_yval1.iterrows():
  if (row['t']<row['Diameter']/6):
   if (510<row['Op. temp. (°C)']<538) & (row['Material']=='Ferritic steels'):
    df_yval1.loc[index, ['Y']] = 0.5
   elif (538<row['Op. temp. (°C)']) & (row['Material']=='Ferritic steels'):
    df_yval1.loc[index, ['Y']] = 0.7

   elif (566<row['Op. temp. (°C)']<593) & (row['Material']=='Austenitic steels'):
    df_yval1.loc[index, ['Y']] = 0.5
   elif (593<row['Op. temp. (°C)']) & (row['Material']=='Austenitic steels'):
    df_yval1.loc[index, ['Y']] = 0.7

   elif (621<row['Op. temp. (°C)']<649) & (row['Material']=='Nickel alloys'):
    df_yval1.loc[index, ['Y']] = 0.5
   elif (649<row['Op. temp. (°C)']) & (row['Material']=='Nickel alloys'):
    df_yval1.loc[index, ['Y']] = 0.7

   elif (row['Material']=='Gray iron'):
    df_yval1.loc[index, ['Y']] = 0

  elif (row['t']>row['Diameter']/6):
   df_yval1.loc[index, ['Y']] = (((row['Diameter']-2*row['Nom. WT'])+2*row['Corr. Allowance'])/(row['Diameter']+(row['Diameter']-2*row['Nom. WT'])+2*row['Corr. Allowance']))

 df['Y']=df_yval1['Y']
Ycoeff()

def Redvalues():
 global red_values
 req_t = []
 t_req_c = []
 t_req_c1 = []
 t_req_c_sf = []
 t_wo_c =[]
 other = []
 red = []
 red_idx = []
 Inspection_red = []
 Line_No= []
 Insp_Tag=[]
 for index, row in df.iterrows():
  req_t.append((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y']))
```

```
 t_req_c.append(((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y'])) + row['Corr. Allowance'])
 df_t=DataFrame(req_t, columns=['t'])
 df_tm=DataFrame(t_req_c, columns=['tm'])



 df_insps = df.filter(regex='Min. WT')

 Inspection_columns = [col for col in df.columns if 'Inspection' in col]
 Year = [item.replace('Inspection ', '') for item in Inspection_columns]




 df_insps = pd.concat([df_insps, df_tm, df['Line No./TAG No.'],df['Insp. TAG No.'],df_t], axis=1)
 for i in Year:
  for index, row in df_insps.iterrows():
   if (row['t'])>=row['Min. WT '+str(i)]:
    red.append(row['Min. WT '+str(i)])
    t_req_c1.append(row['tm'])
    t_req_c_sf.append(row['tm']/0.875)
    t_wo_c.append(row['t'])
    red_idx.append(index+2)
    Inspection_red.append('Inspection No. '+str(i))
    Line_No.append(row['Line No./TAG No.'])
    Insp_Tag.append(row['Insp. TAG No.'])
   else:
    pass
red_dataframe = pd.DataFrame({'idx':red_idx, 'Wall Thickness':red,'Required Wall Thickness':t_wo_c,'Required Wall Thickness w/ CA':t_req_c1, 'Required Wall Thickness + SF & CA':t_req_c_sf,
'Inspection':Inspection_red,'Line No./TAG No.':Line_No, 'Insp. TAG No.':Insp_Tag})
red_values = red_dataframe.sort_values(['idx', 'Wall Thickness','Required Wall Thickness','Required Wall Thickness w/ CA','Required Wall Thickness + SF & CA', 'Inspection','Line No./TAG No.','Insp. TAG
No.'], ascending=True).set_index('idx')

def Yellowvalues():
 global yellow_values
 req_t = []
 t_req_c = []
 t_req_c1 = []
 t_req_c_sf = []
 t_wo_c = []
 other = []
 yellow = []
 yellow_idx = []
 Inspection_yellow = []
 Line_No= []
 Insp_Tag=[]
 for index, row in df.iterrows():
  req_t.append((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y']))
  t_req_c.append(((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y'])) + row['Corr. Allowance'])
  #t_req_c_sf.append(((((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y'])) + row['Corr. Allowance'])/0.875)
  #other.append(((row['Des. pressure (bar)']*(row['Diameter']+2*row['Nom. WT']))/(2*(row['S']*row['E']*row['W']-(row['Des. pressure (bar)']*(1-row['Y']))))))
 df_t=DataFrame(req_t, columns=['t'])
 df_tm=DataFrame(t_req_c, columns=['tm'])
 df_other = DataFrame(other, columns=['tother'])
 df['req_t']=df_t['t']

 df_insps = df.filter(regex='Min. WT')
 lastcol = df_insps.iloc[:, -1:]
 title = list(lastcol)
 Inspection_columns = [col for col in df.columns if 'Inspection' in col]
 Year = [item.replace('Inspection ', '') for item in Inspection_columns]

 df_insps = pd.concat([df_insps, df_tm,df['Line No./TAG No.'],df['Insp. TAG No.'],df['Corr. Allowance'], df_t], axis=1)

 for i in Year:
  for index, row in df_insps.iterrows():
   if row['Corr. Allowance']==0:
    if (row['tm'])<row['Min. WT '+str(i)]<(row['tm']/0.875):
     yellow.append(row['Min. WT '+str(i)])
     yellow_idx.append(index+2)
     t_req_c1.append(row['tm'])
     t_req_c_sf.append(row['tm']/0.875)
     t_wo_c.append(row['t'])
     Inspection_yellow.append('Inspection No. '+str(i))
     Line_No.append(row['Line No./TAG No.'])
     Insp_Tag.append(row['Insp. TAG No.'])
    else:
     pass
   else:
    if (row['t'])<row['Min. WT '+str(i)]<(row['tm']):
     yellow.append(row['Min. WT '+str(i)])
     yellow_idx.append(index+2)
     t_req_c1.append(row['tm'])
     t_req_c_sf.append(row['tm']/0.875)
     t_wo_c.append(row['t'])
     Inspection_yellow.append('Inspection No. '+str(i))
     Line_No.append(row['Line No./TAG No.'])
     Insp_Tag.append(row['Insp. TAG No.'])
    else:
     pass
 yellow_dataframe = pd.DataFrame({'idx':yellow_idx, 'Wall Thickness':yellow,'Required Wall Thickness':t_wo_c,'Required Wall Thickness w/ CA':t_req_c1, 'Required Wall Thickness + SF & CA':t_req_c_sf ,
'Inspection':Inspection_yellow,'Line No./TAG No.':Line_No, 'Insp. TAG No.':Insp_Tag})
 yellow_values = yellow_dataframe.sort_values(['idx', 'Wall Thickness', 'Required Wall Thickness','Required Wall Thickness w/ CA', 'Required Wall Thickness + SF & CA', 'Inspection','Line No./TAG No.','Insp.
TAG No.'], ascending=True).set_index('idx')

def Greenvalues():
 global green_values
 req_t = []
 t_req_c = []
 t_req_c1 = []
 t_req_c_sf = []
 t_wo_c = []
 other = []
 green = []
 green_idx = []
 Inspection_green = []
```

```
Line_No= []
Insp_Tag=[]
for index, row in df.iterrows():
 req_t.append((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y']))
 t_req_c.append(((row['Des. pressure (bar)']*0.1*row['Diameter'])/(2*(row['S']*row['E']*row['W'])+row['Des. pressure (bar)']*0.1*row['Y']))+row['Corr. Allowance'])
df_t=DataFrame(req_t, columns=['t'])
df_tm=DataFrame(t_req_c, columns=['tm'])
df['req_t']=df_t['t']

df_insps = df.filter(regex='Min. WT')
lastcol = df_insps.iloc[:, -1:]
title = list(lastcol)

Inspection_columns = [col for col in df.columns if 'Inspection' in col]
Year = [item.replace('Inspection ', '') for item in Inspection_columns]

df_insps = pd.concat([df_insps, df_tm,df['Line No./TAG No.'],df['Insp. TAG No.'],df_t], axis=1)
for i in Year:
 for index, row in df_insps.iterrows():
  if row['Min. WT '+str(i)]>(row['tm']):
   green.append(row['Min. WT '+str(i)])
   green_idx.append(index+2)
   Inspection_green.append('Inspection No. '+str(i))
   t_req_c1.append(row['tm'])
   t_req_c_sf.append(row['tm']/0.875)
   t_wo_c.append(row['t'])
   Line_No.append(row['Line No./TAG No.'])
   Insp_Tag.append(row['Insp. TAG No.'])
  else:
   pass
green_dataframe = pd.DataFrame({'idx':green_idx, 'Wall Thickness':green,'Required Wall Thickness':t_wo_c,'Required Wall Thickness w/ CA':t_req_c1, 'Required Wall Thickness + SF & CA':t_req_c_sf,
'Inspection':Inspection_green,'Line No./TAG No.':Line_No, 'Insp. TAG No.':Insp_Tag})
green_values = green_dataframe.sort_values(['idx', 'Wall Thickness','Required Wall Thickness' ,'Required Wall Thickness w/ CA','Required Wall Thickness + SF & CA', 'Inspection','Line No./TAG No.','Insp.
TAG No.'], ascending=True).set_index('idx')

Redvalues()
Yellowvalues()
Greenvalues()

reddata = (len(red_values['Wall Thickness']))
yellowdata = (len(yellow_values['Wall Thickness']))
greendata = (len(green_values['Wall Thickness']))

redspool = (len(red_values[red_values['Insp. TAG No.'].str.contains("SPO")]))
redweld = (len(red_values[red_values['Insp. TAG No.'].str.contains("WEL")]))
redbend = (len(red_values[red_values['Insp. TAG No.'].str.contains("BND")]))
redtee = (len(red_values[red_values['Insp. TAG No.'].str.contains("TEE")]))

yellowspool = (len(yellow_values[yellow_values['Insp. TAG No.'].str.contains("SPO")]))
yellowweld = (len(yellow_values[yellow_values['Insp. TAG No.'].str.contains("WEL")]))
yellowbend = (len(yellow_values[yellow_values['Insp. TAG No.'].str.contains("BND")]))
yellowtee = (len(yellow_values[yellow_values['Insp. TAG No.'].str.contains("TEE")]))

greenspool = (len(green_values[green_values['Insp. TAG No.'].str.contains("SPO")]))
greenweld = (len(green_values[green_values['Insp. TAG No.'].str.contains("WEL")]))
greenbend = (len(green_values[green_values['Insp. TAG No.'].str.contains("BND")]))
greentee = (len(green_values[green_values['Insp. TAG No.'].str.contains("TEE")]))

emptydata = {'Inspection grading':['Red', 'Yellow', 'Green', np.nan], 'Number of notes':[reddata, yellowdata,greendata, np.nan], 'Part':['SPOOL', 'WELD', 'BEND', 'TEE'], 'Red':[redspool, redweld, redbend,
redtee], 'Yellow':[yellowspool, yellowweld, yellowbend, yellowtee], 'Green':[greenspool, greenweld, greenbend, greentee]}
emptydf = pd.DataFrame(data=emptydata)
writer = pd.ExcelWriter(str(Excelfile_[0])+'.xlsx', engine='xlsxwriter')



dfl = df.drop(['W','Y','E','S','req_t'], axis=1)
dfl.to_excel(writer,index=False,sheet_name='Sheet 1')
emptydf.to_excel(writer,index=False, sheet_name='Charts')
red_values.to_excel(writer, sheet_name='Red Values')
yellow_values.to_excel(writer, sheet_name='Yellow Values')
green_values.to_excel(writer, sheet_name='Green Values')

writer.save()

fillred = PatternFill(patternType='solid', fgColor='FC3D3D')
fillyellow = PatternFill(patternType='solid', fgColor='FAF330')
fillgreen = PatternFill(patternType='solid', fgColor='3DCF14')

wb = openpyxl.load_workbook(str(Excelfile_[0])+'.xlsx')

wss = wb['Sheet 1']
wsc = wb['Charts']
wsr = wb['Red Values']
wsy = wb['Yellow Values']
wsg = wb['Green Values']

chart = BarChart()
chart.type = "col"
chart.style = 10
chart.title = "Inspection location"
chart.y_axis.title = "Number of notes"
chart.x_axis.title = "Inspection color grade"
bardata = Reference(wsc, min_col=4, min_row=1, max_row=5, max_col=6)
coldata = Reference(wsc, min_col=3, min_row=2, max_row=5)
chart.add_data(bardata, titles_from_data=True)
chart.set_categories(coldata)
chart.shape = 4
wsc.add_chart(chart, "A"+str(wsc.max_row + 1))

chart_Pie = PieChart()
chart_Pie.style = 10
pielabel = Reference(wsc, min_col=1, min_row=2, max_row=4)
piedata = Reference(wsc, min_col=2, min_row=1, max_row=4)
chart_Pie.add_data(piedata, titles_from_data=True)
chart_Pie.set_categories(pielabel)
chart_Pie.title = "Inspections"
chart_Pie.dataLabels = DataLabelList()
```

```
chart_Pie.dataLabels.showPercent = True
wsc.add_chart(chart_Pie, 'G'+str(wsc.max_row + 1))

thin_border = Border(left=Side(style='thin'),
                     right=Side(style='thin'),
                     top=Side(style='thin'),
                     bottom=Side(style='thin'))

for cell in wsr['B']:
 if cell.value != np.nan:
  cell.fill = fillred
  cell.border = thin_border


for cell in wsy['B']:
 if cell.value != np.nan:
  cell.fill = fillyellow
  cell.border = thin_border

for cell in wsg['B']:
 if cell.value != np.nan:
  cell.fill = fillgreen
  cell.border = thin_border


for sheet in wb.worksheets:
 for col in sheet.columns:
  max_length = 0
  column = col[0].column_letter
  for cell in col:
   try:
    if len(str(cell.value)) > max_length:
     max_length = len(cell.value)
   except:
    pass
  adjusted_width = (max_length + 2) * 1.2
  sheet.column_dimensions[column].width = adjusted_width

wb.save(str(Excelfile_[0])+'.xlsx')
```